
AWS Marketplace

Seller Guide



AWS Marketplace: Seller Guide

Copyright © Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

| | |
|---|----|
| What is AWS Marketplace? | 1 |
| Using AWS Marketplace as a seller | 1 |
| Pricing | 2 |
| Getting started as a seller | 4 |
| Seller requirements for publishing free software products | 4 |
| Additional seller requirements for paid products | 4 |
| Eligible jurisdictions for paid products | 5 |
| AWS Marketplace Management Portal | 5 |
| Seller registration process | 6 |
| Creating your public profile | 7 |
| Providing tax information | 7 |
| Providing US bank account information | 7 |
| Completing the Know Your Customer process | 8 |
| Completing bank account verification process | 9 |
| (Optional) Add secondary users for the Know Your Customer procedure | 10 |
| Enrolling in the AWS Marketplace Tax Calculation Service | 11 |
| Disbursement and buyer billing | 12 |
| Already a seller? | 12 |
| Complaints handling policy – Amazon Payments Europe | 13 |
| Seller toolkit | 14 |
| AWS Marketplace Commerce Analytics Service | 14 |
| AWS Marketplace Field Demonstration Program | 25 |
| Product Support Connection | 25 |
| More resources in AWS Marketplace Management Portal | 34 |
| Preparing your product | 35 |
| Product delivery | 35 |
| Product pricing | 37 |
| Pricing models | 38 |
| Changing pricing models | 40 |
| Changing prices | 41 |
| Private offers | 41 |
| Refunds | 41 |
| Regions and countries | 43 |
| AWS Regions | 43 |
| Countries | 43 |
| Private offers | 44 |
| How private offers work | 44 |
| Private offers through consulting partners | 45 |
| Notes about private offers | 45 |
| Supported product types | 45 |
| Offer submission process | 46 |
| Reporting for private offers | 47 |
| Flexible payment scheduler | 48 |
| Consulting partner private offers | 49 |
| Private offer upgrades and renewals | 52 |
| Standardized license terms | 54 |
| Standard Contract for AWS Marketplace (SCMP) | 54 |
| Enterprise Contract for AWS Marketplace (ECMP) | 55 |
| Categories and metadata | 56 |
| Naming and describing your product | 56 |
| Choosing categories and keywords | 57 |
| Search engine optimization for products | 58 |
| Keywords | 58 |
| Software categories | 59 |

| | |
|--|-----|
| Highlights section | 59 |
| Short description | 59 |
| AWS Marketplace for Desktop Applications (AMDA) | 60 |
| Starting the onboarding process | 60 |
| Product submission and packaging | 60 |
| Application packaging types | 61 |
| Building the AMDA package | 61 |
| Application metadata | 62 |
| Ingestion and new version updates | 62 |
| AMI-based products | 63 |
| AMI-based product delivery methods | 63 |
| Understanding AMI-based products | 63 |
| Product lifecycle | 64 |
| AMI product codes | 65 |
| Change requests | 65 |
| Product Load Forms | 66 |
| Single-AMI products | 66 |
| Prerequisites | 66 |
| Creating your product | 67 |
| Creating a change request | 68 |
| Getting status of a request | 69 |
| Updating product information | 69 |
| Updating version information | 70 |
| Adding a new version | 70 |
| Giving AWS Marketplace access to your AMI | 71 |
| Restricting a version | 72 |
| Removing a product from AWS Marketplace | 73 |
| Common errors when submitting change requests | 74 |
| AMI-based Delivery Using CloudFormation | 74 |
| Building your product listing | 75 |
| Preparing your AWS CloudFormation template | 75 |
| Getting the cost estimate for your template infrastructure | 77 |
| Topology diagram | 77 |
| Meeting the submission requirements | 77 |
| Submitting your product request | 78 |
| Adding serverless application components | 79 |
| Private images | 84 |
| Package group requirements | 85 |
| Submitting your package group | 86 |
| Scan status | 86 |
| Submitting your product to AWS Marketplace | 87 |
| Best practices for building AMIs | 88 |
| Verifying your AMI | 88 |
| Securing resell rights | 88 |
| Building an AMI | 88 |
| Verifying your software is running on your AWS Marketplace AMI | 89 |
| Securing an AMI | 90 |
| AMI product pricing | 91 |
| AMI pricing models | 91 |
| AWS charges and software charges | 93 |
| Contract pricing for AMI products | 94 |
| AWS Marketplace Metering Service integration | 96 |
| Metering service concepts | 96 |
| Pricing your software | 98 |
| Adding your product to AWS Marketplace | 98 |
| Modifying your software to use the Metering Service | 100 |
| Call AWS Marketplace Metering Service | 101 |

| | |
|---|-----|
| Failure handling | 102 |
| Limitations | 102 |
| Vendor-metered tagging (Optional) | 103 |
| Code example | 104 |
| AWS License Manager integration | 105 |
| License models | 106 |
| Integration workflow | 109 |
| License Manager integration prerequisites | 109 |
| Integrating an AMI product with License Manager | 110 |
| License renewals and upgrades | 110 |
| Amazon SNS notifications for AMI products | 111 |
| Amazon SNS topic: <code>aws-mp-subscription-notification</code> | 111 |
| Subscribing an Amazon SQS queue to the Amazon SNS topic | 111 |
| AMI product checklist | 112 |
| AMI security policies | 113 |
| Security policies | 113 |
| Access policies | 113 |
| Customer information policies | 114 |
| Product usage policies | 114 |
| Architecture policies | 115 |
| Container-based products | 116 |
| Getting help | 116 |
| Getting started with container products | 116 |
| Prerequisites | 117 |
| Creating a container product | 117 |
| Creating the product ID for your container product | 118 |
| Creating or updating pricing details for container products | 118 |
| Integrating AWS Marketplace Metering Service for your container product | 119 |
| Integrating AWS License Manager for your container product | 119 |
| Adding a new version of your product | 119 |
| Updating version information | 124 |
| Creating or updating product information for your container product | 125 |
| Publishing container products | 126 |
| Container product scans for security issues | 126 |
| Container-based product requirements | 126 |
| Security requirements | 127 |
| Access requirements | 127 |
| Customer information requirements | 127 |
| Product usage requirements | 127 |
| Architecture requirements | 128 |
| Container product pricing | 129 |
| Container pricing models | 129 |
| Contract pricing for container products | 131 |
| Billing, metering, and licensing integrations | 134 |
| Hourly and custom metering with AWS Marketplace Metering Service | 134 |
| Contract pricing with AWS License Manager | 135 |
| Hourly metering with AWS Marketplace Metering Service | 136 |
| Custom metering with AWS Marketplace Metering Service | 143 |
| Contract pricing with AWS License Manager | 152 |
| Amazon SNS notifications for container products | 174 |
| Amazon SNS topic: <code>aws-mp-subscription-notification</code> | 174 |
| Subscribing an Amazon SQS queue to the Amazon SNS topic | 174 |
| Machine learning products | 176 |
| Getting started with machine learning products | 176 |
| SageMaker model package | 176 |
| SageMaker algorithm | 176 |
| Deploying an inference model | 176 |

| | |
|--|-----|
| Security and intellectual property | 177 |
| Protecting intellectual property | 177 |
| No network access | 177 |
| Security of customer data | 177 |
| Machine learning product pricing | 178 |
| Infrastructure pricing | 178 |
| Software pricing | 178 |
| Prepare your product in SageMaker | 180 |
| Packaging your code into images | 180 |
| Uploading your images | 194 |
| Creating your Amazon SageMaker resource | 196 |
| Publishing your product in AWS Marketplace | 200 |
| Overview of publishing process | 200 |
| Permissions required | 200 |
| Creating your product listing | 201 |
| Testing your product | 203 |
| Updating your product | 203 |
| Requirements and best practices for creating machine learning products | 204 |
| Required assets | 204 |
| General best practices for ML products | 205 |
| Requirements for usage information | 205 |
| Requirements for inputs and outputs | 205 |
| Requirements for Jupyter notebook | 206 |
| Summary of requirements and recommendations for ML product listings | 207 |
| Service restrictions and quotas | 209 |
| Network isolation | 209 |
| Image size | 209 |
| Storage size | 209 |
| Instance size | 210 |
| Payload size for inference | 210 |
| Processing time for inference | 210 |
| Service quotas | 210 |
| Managed spot training | 210 |
| Docker images and AWS accounts | 210 |
| Publishing model packages from built-in algorithms or AWS Marketplace | 210 |
| Supported AWS Regions for publishing | 211 |
| Troubleshooting | 211 |
| Reporting | 212 |
| Daily business report | 212 |
| Monthly revenue report | 212 |
| Disbursement report | 213 |
| Other reports and analysis | 213 |
| Software as a service (SaaS)–based products | 214 |
| Getting started with SaaS products | 214 |
| Prerequisites | 214 |
| Creating a SaaS product | 215 |
| Create an initial SaaS product page | 216 |
| Integrate your SaaS subscription product | 216 |
| Integrate your SaaS contract product | 219 |
| Integrate your SaaS contract with pay-as-you-go product | 221 |
| Deploy a serverless SaaS integration solution | 223 |
| Plan your SaaS product | 224 |
| Plan your pricing | 224 |
| Plan your billing integration | 225 |
| Plan your Amazon SNS integration | 225 |
| Plan how customers will access your product | 225 |
| SaaS product guidelines | 226 |

| | |
|---|-----|
| Product setup guidelines | 226 |
| Customer information requirements | 226 |
| Product usage guidelines | 226 |
| Architecture guidelines | 227 |
| SaaS product pricing | 227 |
| | 228 |
| Pricing for SaaS subscriptions | 228 |
| Pricing for SaaS contracts | 229 |
| SaaS customer onboarding | 232 |
| Configuring your SaaS product to accept new buyers | 233 |
| Amazon SNS notifications for SaaS products | 234 |
| Amazon SNS topic: <code>aws-mp-entitlement-notification</code> | 234 |
| Amazon SNS topic: <code>aws-mp-subscription-notification</code> | 235 |
| Subscribing an SQS queue to the SNS topic | 236 |
| Accessing the AWS Marketplace Metering and Entitlement Service APIs | 236 |
| Metering for usage | 236 |
| Checking entitlements | 239 |
| SaaS product integration checklist | 240 |
| Reporting | 242 |
| SaaS code examples | 243 |
| ResolveCustomer code example | 243 |
| GetEntitlement code example | 244 |
| BatchMeterUsage code example | 245 |
| BatchMeterUsage with usage allocation tagging code example (Optional) | 246 |
| Using AWS PrivateLink with AWS Marketplace | 247 |
| Introduction | 247 |
| Configuring your product | 248 |
| Submitting your product to AWS Marketplace | 249 |
| Buyer access to VPC endpoints | 249 |
| Appendix: Checklists | 250 |
| Professional services products | 251 |
| Getting help | 251 |
| Getting started with professional services products | 251 |
| Prerequisites | 252 |
| Creating a professional services product | 252 |
| Creating private offers | 253 |
| Editing product information | 254 |
| Editing product pricing | 254 |
| Editing product visibility | 255 |
| Removing a professional services product | 255 |
| Product details | 256 |
| Product descriptions | 256 |
| Additional resources | 257 |
| Support information | 257 |
| Pricing dimensions | 257 |
| Product visibility | 258 |
| Product requirements | 258 |
| Product setup guidelines | 258 |
| Customer information requirements | 258 |
| Product usage guidelines | 259 |
| Architecture guidelines | 259 |
| Professional services product pricing | 259 |
| Data products | 260 |
| Submitting your product | 261 |
| Using the Products tab | 262 |
| Company and product logo requirements | 262 |
| Requirements for submitting paid repackaged software | 263 |

| | |
|--|-----|
| AWS CloudFormation-launched product (free or paid) or usage-based paid AMI product | 263 |
| Submitting your product | 264 |
| Updating your product | 264 |
| Product changes and updates | 264 |
| Timing and expectations | 265 |
| Submitting AMIs to AWS Marketplace | 265 |
| AMI self-service scanning | 265 |
| AMI cloning and product code assignment | 266 |
| Final checklist | 266 |
| Marketing your product | 268 |
| 180-day GTM Academy | 268 |
| Announcing your product's availability | 268 |
| AWS Marketplace messaging | 268 |
| Reviews on AWS Marketplace | 269 |
| Linking to AWS Marketplace | 270 |
| Using the AWS Marketplace logo | 270 |
| Linking directly to your product on AWS Marketplace | 270 |
| Press releases | 270 |
| AWS Marketplace trademark usage guidelines | 271 |
| Seller reports and data feeds | 273 |
| Seller reports | 273 |
| Accessing reports | 273 |
| Daily business report | 274 |
| Daily customer subscriber report | 281 |
| Disbursement report | 283 |
| Monthly billed revenue report | 288 |
| Sales compensation report | 294 |
| US sales and use tax report | 296 |
| Data feeds | 299 |
| Storage and structure of data feeds | 299 |
| Accessing data feeds | 301 |
| Using data feeds | 303 |
| Data feed tables overview | 303 |
| Data feed query examples | 310 |
| Account data feed | 320 |
| Address data feed | 321 |
| Billing event data feed | 323 |
| Legacy mapping data feed | 332 |
| Offer data feed | 332 |
| Offer product data feed | 333 |
| Offer target data feed | 334 |
| Product data feed | 335 |
| Tax item data feed | 336 |
| Security | 113 |
| IAM for AWS Marketplace | 343 |
| Creating users | 343 |
| Creating or using groups | 344 |
| Signing in as an IAM user | 345 |
| Policies and permissions for AWS Marketplace sellers | 346 |
| Policies | 346 |
| Permissions | 346 |
| AWS managed policies | 349 |
| AWSMarketplaceAmiIngestion | 350 |
| AWSMarketplaceFullAccess | 350 |
| AWSMarketplaceGetEntitlements | 352 |
| AWSMarketplaceMeteringFullAccess | 352 |
| AWSMarketplaceMeteringRegisterUsage | 353 |

| | |
|--|-----|
| AWSMarketplaceSellerFullAccess | 353 |
| AWSMarketplaceSellerProductsFullAccess | 354 |
| AWSMarketplaceSellerProductsReadOnly | 355 |
| Policy updates | 356 |
| AWS Marketplace Commerce Analytics Service account permissions | 356 |
| AWS Marketplace Product Support Connection account permissions | 357 |
| Amazon SQS permissions | 357 |
| AWS Marketplace metering and entitlement API permissions | 358 |
| IAM policy for SaaS products | 358 |
| IAM policy for AMI products | 359 |
| IAM policy for container products | 359 |
| Logging AWS Marketplace API calls with AWS CloudTrail | 359 |
| AWS Marketplace log file entry examples | 360 |
| Related Topics | 363 |
| Document history | 364 |
| AWS glossary | 370 |

What is AWS Marketplace?

AWS Marketplace is a curated digital catalog that customers can use to find, buy, deploy, and manage third-party software, data, and services to build solutions and run their businesses. AWS Marketplace includes thousands of software listings from popular categories such as security, business applications, machine learning, and data products across specific industries, such as healthcare, financial services, and telecommunications. Customers can quickly launch preconfigured software, and choose software solutions in Amazon Machine Images (AMIs), software as a service (SaaS), and other formats. Professional services are also available to help customers configure, deploy, and manage third-party software. For a complete list of delivery methods, see [Product delivery](#).

You can use AWS Marketplace as a buyer (subscriber), seller (provider), or both. Anyone with an AWS account can use AWS Marketplace as a buyer, and can register to become a seller. A seller can be an independent software vendor (ISV), consulting partner, managed services provider (MSP), or individual who has something to offer that works with AWS products and services.

Note

Data product providers must meet the AWS Data Exchange eligibility requirements. For more information, see [Providing data products on AWS Data Exchange](#) in the *AWS Data Exchange User Guide*.

Eligible partners can programmatically list AWS Marketplace products outside of AWS Marketplace. For more information about becoming an eligible partner, contact your AWS Marketplace business development partner.

The following video explains more about selling on AWS Marketplace.

[Introduction to AWS Marketplace](#)

Using AWS Marketplace as a seller

The following diagram shows the process for selling a software product on AWS Marketplace.



Seller process

| Step | Action | Description |
|------|---------------------------------|---|
| 1 | Register (p. 6) | As a seller, you start by registering for the AWS Marketplace Management Portal (AMMP). We recommend that you implement a new dedicated AWS account that can be easily linked with an existing AWS organization. Verify that the AWS Partner's tax information meets the jurisdictional eligibility criteria. For AWS Partners without an existing U.S. bank account, you can create one at no additional cost by using Hyperwallet . |
| 2 | Decide product type | Decide on the type of product that you want to sell. For more information about creating the product types in AWS Marketplace, see the following: |

| Step | Action | Description |
|------|--|---|
| | | <ul style="list-style-type: none"> • AWS Marketplace for Desktop Applications (AMDA) (p. 60) • AMI-based products (p. 63) • Container-based products (p. 116) • Machine learning products (p. 176) • Software as a service (SaaS)–based products (p. 214) • Professional services products (p. 251) • Data products (For more information about data products, see What is AWS Data Exchange? in the <i>AWS Data Exchange User Guide</i>.) |
| 3 | Prepare product (p. 35) | Configure your package, set a pricing scheme, determine the relevant categories in which to list your product, and add keywords so your product appears in relevant searches. To simplify the procurement process, you can use standardized license terms for both public product listings and private offers. |
| 4 | Submit product (p. 261) | Use the product submission process to make your products available on AWS Marketplace. Products can be simple, for example, a single Amazon Machine Image (AMI) that has one price structure. Or, products can be complicated, with multiple AMIs, AWS CloudFormation templates, and complex pricing options and payment schedules. |
| 5 | Market product (p. 268) | Contribute to the success of your product by driving awareness of AWS Marketplace and by driving traffic directly to your product pages on AWS Marketplace. |
| 6 | View reports and data feeds (p. 273) | After you're registered as a seller, use the AWS Marketplace Management Portal to access usage reports for your products. AWS Marketplace provides tools for collecting and analyzing information about your product sales. |
| 7 | Manage products | Use the AWS Marketplace Management Portal to manage your account and product pages. |

As a seller, go to the [AWS Marketplace Management Portal](#) to register. If you're charging for use of your product, you must also provide tax and banking information as part of your registration. When you register, you create a profile for your company or for yourself that is discoverable on AWS Marketplace. You also use the [AWS Marketplace Management Portal](#) to create and manage product pages for your products.

Pricing

Products can be free to use or can have associated charges. The charge becomes part of the buyer's AWS bill, and after the buyer pays, AWS pays the seller. Products can take many forms. For example, a product can be offered as an Amazon Machine Image (AMI) that is instantiated using a buyer's AWS account. Products can also be configured to use CloudFormation templates for delivery to the buyer. Products can also be SaaS offerings from an ISV, web access control lists (web ACL), sets of rules, or conditions for AWS WAF. Products can also be professional services from an ISV, consulting partners, or MSP.

Flexible pricing options include free trial, hourly, monthly, annual, multi-year, and Bring Your Own License model (BYOL), and being billed from one source. AWS handles billing and payments, and charges appear on customers' AWS bill.

Software products can be purchased at the listed price using the ISV's standard end user license agreement (EULA). In addition, software products can be offered with custom pricing and EULA through private offers. Products can also be purchased under a contract with specified time or usage boundaries. After subscribing to a product, the buyer can copy the product to their AWS Service Catalog to manage how the product is accessed and used in the buyer's organization. For more information about pricing, see [the section called "Product pricing" \(p. 37\)](#).

Getting started as a seller

If you want to sell your software in AWS Marketplace, review the requirements and then follow the steps to register as a seller. There are different registration requirements based on where you reside and what type of products you're selling. To register as a seller in AWS Marketplace, you can use an existing AWS account or create a new account. All AWS Marketplace interactions are tied to the account that you choose.

Notes

- Registering as an AWS Marketplace seller is a prerequisite to listing data products on AWS Data Exchange and making them available on AWS Marketplace. For more information about these requirements, see [Providing Data Products on AWS Data Exchange](#) in the *AWS Data Exchange User Guide*.
- For information about the permissions that AWS Marketplace sellers need, see [Policies and permissions for AWS Marketplace sellers \(p. 346\)](#).

Seller requirements for publishing free software products

Regardless of whether you charge for your product when you offer it in AWS Marketplace, you're selling that product. The cost to the customer is \$0.00, but you and the customer agree to a mutual contract for use of the product. If you offer only free products, you don't have to provide banking information to AWS Marketplace. To create and offer free products in AWS Marketplace, you must:

- Sell publicly available, full-feature production-ready software.
- Have a defined customer support process and support organization.
- Provide a means to keep software regularly updated and free of vulnerabilities.
- Follow best practices and guidelines when marketing your product in AWS Marketplace.
- Be an AWS customer in good standing and meet the requirements in the terms and conditions for AWS Marketplace sellers.

Additional seller requirements for paid products

If you charge for your products or offer Bring Your Own License model (BYOL) products, you must also meet the following requirements and provide this additional information:

- You must be a permanent resident or citizen in an [eligible jurisdiction \(p. 5\)](#), or a business entity organized or incorporated in one of those areas.
- You must provide tax and bank account information. For US-based entities, a W-9 form and a banking account from a US-based bank are required.
- Non-US sellers are required to provide a (i) W-8 form, value-added tax (VAT) or goods and services tax (GST) registration number, and (ii) US bank information. If you don't have a US bank account, you can register for a virtual US bank account from [Hyperwallet](#).
- To provide data products, you must also request on-boarding through the [Create case](#) wizard for AWS Support.
- To sell products to customers whose AWS accounts are based in countries and territories in Europe, the Middle East, and Africa (EMEA) (excluding Turkey and South Africa) through Amazon Web Services EMEA SARL, you must [complete the Know Your Customer process \(p. 8\)](#). In addition:

- You receive up to two disbursements (for transactions through AWS Inc. and Amazon Web Services EMEA SARL).
- You may be taxed on the listing fee for certain transactions, depending on location. For more information about taxes, see the [AWS Marketplace Sellers Tax](#) help page. If value-added tax (VAT) on your listing fee is assessed, AWS Marketplace will provide a tax-compliant invoice.
- For more information about Amazon Web Services EMEA SARL, see *AWS EMEA Marketplace - Sellers* on the [Amazon Web Services Europe FAQs](#) website.

To sell into the AWS GovCloud (US) Region, sellers must have an [AWS GovCloud \(US\) account](#). For details on ITAR requirements, see the [AWS GovCloud \(US\) User Guide](#).

For questions about AWS Marketplace seller requirements or the registration process, contact the [AWS Marketplace Seller Operations](#) team.

Eligible jurisdictions for paid products

To sell paid software in AWS Marketplace, you must be a permanent resident or citizen in one of the following countries or SARs, or a business entity organized or incorporated therein:

- Australia¹
- Bahrain^{1 2}
- European Union (EU) member state¹
- Hong Kong SAR
- Japan^{2 3}
- New Zealand¹
- Norway^{1 2}
- Qatar
- Switzerland^{1 2}
- United Arab Emirates (UAE)^{1 2}
- United Kingdom (UK)¹
- United States (US)

¹ Sellers of paid products in these countries must provide VAT registration information in country of establishment.

² If you as a seller are located in the same country as the buyer, you may be responsible for tax invoicing, collections, and remittances. Please consult with your tax advisor.

³ Sellers based in Japan have an obligation to self-account for the Japanese Consumption Tax (JCT) on the listing fee charges. Sellers based in other jurisdictions may have similar obligations. Please consult with your tax advisor.

For more information about VAT, invoicing, and your tax obligations as a seller, see [AWS Marketplace Sellers](#) on [Amazon Web Service Tax Help](#).

AWS Marketplace Management Portal

The [AWS Marketplace Management Portal](#) is the tool that you use to register as an AWS Marketplace seller. Then, you can use the portal to manage the products that you sell in AWS Marketplace. You can complete the following tasks on the portal:

- Register as an AWS Marketplace seller.

- Use the **Products** page to submit new software products and update existing software products.
- Monitor the status of your requests.
- Upload files needed to create and manage your new software products.
- Manage your software products into incremental channel revenue by taking advantage of the go-to-market activities.
- Measure the results of your marketing efforts within hours of launch, including the usage and revenue driven by your campaigns.
- Enable customer service representatives to retrieve customer data in real time.
- Initiate an automatic Amazon Machine Image (AMI) scan to detect vulnerabilities.

Note

Data products are published and managed from the AWS Data Exchange console. AWS Data Exchange providers can use the AWS Marketplace Management Portal to register as a seller, request AWS Data Exchange on-boarding, access seller reports, and submit refund requests.

All registered sellers can access the AWS Marketplace Management Portal using their AWS credentials for the account that they used to create their products. The account that you use is defined as the seller of record when a customer subscribes to your product. If you need help determining the specific account that is the seller of record for your products, contact the [AWS Marketplace Seller Operations](#) team.

AWS Marketplace strongly recommends using AWS Identity and Access Management (IAM) roles to sign in to the AWS Marketplace Management Portal rather than using your root account credentials. For more information, see [IAM Users](#) in the *IAM User Guide*.

To enable people in your company to sign in to the AWS Marketplace Management Portal, create an IAM user for each person you want to have access and define access permissions to the AWS Marketplace Management Portal. We also recommend creating an AWS account root user role to use for access. For more information, see [AWS account root user credentials and IAM user credentials](#) in the *AWS General Reference*.

Seller registration process

By registering as a seller for AWS Marketplace, you can sell your products and services to other AWS Marketplace customers.

Registering as a seller requires the following steps:

1. **Create your public profile** – You provide the information that is displayed in AWS Marketplace to buyers that tells them about your company, such as your company name and logo. After you complete this process, you can sell products for free. To sell paid products, you must complete steps two and three.
2. **Provide your tax information** – To appropriately assess, report, and (where applicable) withhold taxes on your paid sales, you must provide your tax and value added tax (VAT) information.
3. **Provide your banking information** – You provide your US bank information so that AWS Marketplace can pay you for your sales.
4. (Optional) **Enroll in the US tax calculation service** – You can optionally enroll in this service to calculate your US state sales and use tax for products you sell on AWS Marketplace.

These steps are described in more detail in the following sections.

After you have completed registering your account as a seller, you can create products to sell to buyers through AWS Marketplace. For more information, see [Preparing your product \(p. 35\)](#).

You can use AWS Identity and Access Management (IAM) to configure your primary AWS account to allow multiple users with various permissions to access the AWS Marketplace Management Portal. For more information, visit [the section called "IAM for AWS Marketplace" \(p. 343\)](#).

Creating your public profile

The first step to register is to select the AWS account to use as your primary AWS Marketplace account, and provide the information that is displayed to potential buyers in the AWS Marketplace console.

Note

Once you use an AWS account to list a product on AWS Marketplace, you can't change the account associated with the product. You can use an existing account or register a new account. This account will be the seller of record for your products in AWS Marketplace and will be used for reporting, disbursement, and communication from AWS Marketplace to you.

To create your public profile

1. From the [AWS Marketplace Management Portal](#) (AMMP), choose **Register now** and then sign in to your chosen seller AWS account.
2. Select **Add public profile** to provide your seller information.

After you have completed the public profile, you can publish and sell free products. To sell paid products, you must provide your tax and banking information.

Providing tax information

You must provide your tax, and value added tax (VAT) where applicable, information so that AWS Marketplace can accurately report and withhold taxes on your product sales.

To provide your tax information

1. Sign in to the [AWS Marketplace Management Portal](#), and choose **Settings**.
2. Select **Complete tax information** in the **Payment Information** section.
3. After you have completed the tax information, return to the **Settings** page and select **Complete VAT information**, if it is available.

Note

The VAT information section is only available if you are in an AWS Region that supports VAT.

Providing US bank account information

A US bank account is required for all sellers who want to sell paid products in AWS Marketplace. AWS Marketplace only disburses to US bank accounts.

Note

For a list of countries where you can offer paid products in AWS Marketplace, see [Eligible jurisdictions for paid products \(p. 5\)](#).

To provide US bank information

1. Sign in to the [AWS Marketplace Management Portal](#), and choose **Settings**.
2. Select **Complete banking information** in the **Payment Information** section.
3. Provide the required information about your US bank account.

Note

If you have not yet provided your tax information (and value added tax information, where applicable), you will not be able to provide your banking information.

If you do not already have a US bank account, you may be able to obtain one through Hyperwallet. Hyperwallet can provide you with a US account, which you can provide to AWS Marketplace for your AWS Marketplace disbursements.

Hyperwallet is an independent service provider that may enable you to transfer funds to another bank account in a supported currency. For a limited time, you will not be required to pay certain Hyperwallet service fees in connection with AWS Marketplace disbursements.

- By adding your Hyperwallet account details to your AWS Marketplace seller account, you agree and acknowledge that AWS Marketplace will share your name, email address, and account number with Hyperwallet to confirm your status as an AWS Marketplace seller.
- Additional fees may apply to your use of Hyperwallet services (including transfer fees and foreign exchange fees required to transfer funds into your local currency), as well as foreign exchange rates. Hyperwallet's service fee will be waived for a limited time, and only with respect to AWS Marketplace disbursements of the proceeds from your Paid products into your Hyperwallet account. Consult the Fees section of the Hyperwallet site or contact Hyperwallet for more information and to review applicable fees. Visit the [Hyperwallet support site](#) to learn more about their services.

To begin registration with Hyperwallet and obtain your US bank account information

1. Sign in to the [AWS Marketplace Management Portal](#), and choose **Settings**, then select **Complete banking information** in the **Payment Information** section.
2. If you do not have a Hyperwallet account, and need one for use in AWS Marketplace, choose **No** in response to **Do you have a US bank account?** and **Are you registered with Hyperwallet?** You will be provided with a personal identification number (PIN) and link to sign up for Hyperwallet.
3. After you have activated your Hyperwallet account, follow the steps described on the Hyperwallet registration portal to complete registration and receive your deposit account information.
4. When you have obtained an account from Hyperwallet, add your Hyperwallet account information to your AWS account by signing in to the [AWS Marketplace Management Portal](#). Then, choose **Settings**, then select **Complete banking information** in the **Payment Information** section.

Completing the Know Your Customer process

Know Your Customer (KYC) is a compliance requirement used by financial institutions and online businesses to verify the identity of their customers. This requirement is due to the revised Payment Services Directive (PSD 2) and European Union anti-money laundering Directives that govern financial institutions such as banks and other payment institutions.

AWS Marketplace transactions through Amazon Web Services EMEA SARL are processed through Amazon Payments Europe, S.C.A. (APE), a licensed electronic money institution in Luxembourg which requires verification of your identity in order to use the payment service.

For you as a seller to transact through Amazon Web Services EMEA SARL, you are required to complete the KYC process. This process involves providing additional information about your company, key points of contact, beneficial ownership, and supporting documentation.

To complete the KYC process

1. On the AWS Marketplace Management Portal, choose **Settings**.
2. In the **Account Summary** section, confirm that the **Country** that is shown is correct.

Note

Choose the **Info** link to see how to change your country.

3. Choose **Go to KYC information** or select the **Know your customer (KYC)** tab and then choose **Start KYC Compliance** and you will be re-directed to the KYC registration portal.

For more information about how information is used and shared by AWS Marketplace, see the [Amazon Payments Europe Privacy Notice](#).

4. Choose **Go to KYC overview**.
5. On the **Know Your Customer (KYC) Overview**, read through the list of required information and documentation and gather the required documentation (if you haven't yet done so). Then choose **Continue to KYC compliance**.
6. Enter the **Basic details** as directed. After you review the Amazon Payments Europe Terms & Conditions, choose **Agree and continue**.

When you continue to the next page or next step in the KYC process, that action indicates that you accept the Amazon Payments Europe Terms & Conditions.

If you have questions, refer to **Frequently Asked Questions (FAQ)** located on the right side of the console.

7. Enter the required **Business information** as directed, and then choose **Next**.

Note

Your information is saved every time you chose **Next** to go to the next step.

8. Enter the required **Point of contact information** as directed, and then choose **Next**.
9. Choose whether the **Beneficial owner** is the same as the point of contact, add beneficial owners (up to four) if necessary, confirm your additions, and then choose **Next**.
10. For **Additional documents**, upload your business license, identity document, and letter of authorization (if applicable).
11. On **Review and Submit**, review and verify all of the information that you have entered.

You can select **Edit** to return to any previous section if necessary.

12. Choose **Submit for verification**.

The status of your KYC compliance will be reviewed (typically within 24 hours). You will be notified through an email message after the review is complete. The entire KYC process typically takes approximately 2 weeks.

You can return to the **Settings** tab to view the status of your KYC compliance on the **Account Summary** card. For more information about your KYC status, choose the **Know your customer (KYC)** tab under the **Account Summary** card. It will display **Under review** until the review has been completed.

After your KYC is verified, you must provide a bank statement on the **Payment information** tab before you can receive disbursements through APE.

Completing bank account verification process

To receive disbursements from Amazon Payments Europe (APE), you must provide additional information to verify your disbursement bank account that is listed in the **Payment Information** tab in the AWS Marketplace Management Portal.

Providing additional bank information

To provide additional bank information

1. Sign in to the AWS Marketplace Management Portal, and then choose **Settings**.

2. Select **Update banking information** in the **Payment Information** section.
3. Select the appropriate disbursement account.

The **Verification status** displays **Not Verified**.

4. Choose **Verify**.
5. You will be re-directed to the **Bank Account Verification** registration portal where you can upload and submit your bank statement.

If you use Hyperwallet's virtual bank account solution, see [the section called "Downloading your bank statement from Hyperwallet"](#) (p. 10).

6. In the portal, choose **Upload bank document** and then choose **Submit**.

Downloading your bank statement from Hyperwallet

For sellers who use [Hyperwallet's virtual bank account solution](#), you can download the Hyperwallet bank statement by using the following procedure. Then, you can upload the bank document as directed in [Completing bank account verification process](#) (p. 9).

To download your bank statement from Hyperwallet

1. Sign into your [Hyperwallet account](#).
2. Go to the [Deposit Account Information](#) page.
3. Download your **Bank Account Validation Statement**.

(Optional) Add secondary users for the Know Your Customer procedure

Secondary users (IAM users) are individuals who can amend KYC information, control the flow of funds or refunds, and change financial information such as bank account details.

Only secondary users that are KYC verified can make the aforementioned updates. These secondary users are subject to the same ongoing screening controls as the root account owner.

To become KYC verified, secondary users must complete the procedure in [Completing the Know Your Customer process](#) (p. 8).

To add secondary users for the Know Your Customer procedure

1. Ask the IAM user to sign in to the AWS Marketplace Management Portal.
2. Navigate to the **Settings** tab.
3. Choose the **Know Your Customer (KYC)** tab and see the section for **Secondary user information**.
4. Choose **Complete secondary user information**.

You are re-directed to the **Secondary User** registration portal.

5. In the **Secondary User** registration portal, complete the required fields, and then choose **Next**.
6. On the **Review and Submit** page, upload a copy of the identity document (**Upload Passport**) and proof of address (**Upload Document**).
7. Choose **Submit for Verification**.

The status of your KYC compliance will be reviewed (typically within 24 hours). You will be notified through an email message after the review is complete. The entire KYC process typically takes approximately 2 weeks.

Enrolling in the AWS Marketplace Tax Calculation Service

AWS Marketplace Tax Calculation Service provides the ability to calculate and collect US sales and use tax for existing and new products. Some states are not eligible for Tax Calculation Service because AWS Marketplace is required by law to collect and remit applicable sales tax attributable to taxable sales of your products to customers based in these states. To use the service, configure your tax nexus settings for your seller profile, and then assign product tax codes to your products.

Note

We recommend that you review the [AWS Marketplace Tax Methodology](#) and [AWS Marketplace Product Tax Code Guidance](#) in their entirety before completing this process.

To configure your tax nexus settings

1. Open the [AWS Marketplace Management Portal](#)
2. Under the **Settings** tab, configure the applicable tax nexus settings.
3. Assign product tax codes (PTCs) to your products through the [AWS Marketplace Management Portal](#).
4. For product types not supported by the **Products** tab submission process, download a product load form by choosing **File Upload** from the **Assets** tab. You must edit and upload the updated product load form.

After you have completed these steps, US sales and use tax calculation will be enabled.

Note

- Activation of your tax nexus settings takes from 5–48 hours.
- Tax nexus settings must be configured before you can assign PTCs.
- PTC assignment happens 24 hours after the AWS Marketplace team approves and publishes your product, which may take 3–5 days from the time you submit your product change request.
- When tax calculation begins, estimated sales tax charges are included in customer invoices. Sales tax is calculated based on factors including, but not limited to, the customer's billing address, the tax code of your product, and your tax nexus settings. The resulting sales tax charge, if applicable, is included in the customer's invoice and identified as a US sales tax charge under the specific product sold by your company. Customer invoices show your company's Legal Name, which you provided when you registered to become an AWS Marketplace seller.
- The collected sales tax funds are sent with your monthly disbursement. The US Sales and Use Tax Report is available to you on the fifteenth of the month, detailing what taxes were collected. You are responsible for remitting your own taxes.

If you enroll for the AWS Marketplace Tax Calculation Service, we also recommend that you register for the [Amazon Tax Exemption Program](#) (ATEP). You are not required to use this service. However, we recommend that all AWS Marketplace sellers who use the Tax Calculation Service participate in ATEP. Participation helps to reduce the number of tax-only refunds that will need to be processed to qualified customers registered in ATEP.

You can edit or delete the tax nexus information on the [Tax Calculation Service Settings](#) page in the AWS Marketplace Management Portal.

For more information, visit [AWS Marketplace Sellers](#) on [Amazon Web Service Tax Help](#) to learn more about where AWS collects sales tax, VAT, or GST on your sales and remits such taxes to the local tax authorities, in the name of AWS, Inc.

Note

Your use of the Tax Calculation service is governed by the [AWS Marketplace US Tax Collection Support Terms and Conditions](#)

Disbursement and buyer billing

AWS acts as the billing mechanism on your behalf. The two most common payment options available to buyers are *credit card* and *invoicing*.

The following is information about the billing for AWS Marketplace subscriptions:

- Purchases with upfront payments are billed immediately upon subscription.
- Billing schedules for private offers are agreed upon between the buyer and seller.
- Invoice payment terms (including bill due date) are agreed upon between the buyer and AWS. The terms are *not* disclosed to vendors.
- Private offers using the flexible payment scheduler are required to be on *invoicing* as the payment option.
- You can validate the invoicing using the [Monthly billed revenue report \(p. 288\)](#). This report summarizes invoicing by AWS on your behalf. This report contains a Transaction Reference key to match and provide visibility to the invoice creation date and invoice due date.

The following is information about how you as the seller get your disbursement:

- A valid [payment method](#), a [registered US bank account](#), and a submitted W9 form are required for disbursement.
- Sellers of paid products are required to provide a W-8, value added tax (VAT) or good and services tax (GST) registration number, and a US bank account. [Hyperwallet](#) can provide you with a US bank account, which you can provide to AWS Marketplace for your AWS Marketplace disbursements.
- AWS disburses payments monthly directly to the bank account associated with the seller account, minus AWS Marketplace service fees.
- AWS disburses payment via ACH transfer after the buyer pays an invoice.
- AWS disbursements are made once a month between the 7th and 10th of the month. The date will be the same for a seller each month. The [Disbursement report \(p. 283\)](#) will reflect your disbursement date.
- AWS disbursements cover a rolling monthly period (starting when the seller account was created).
- Funds are disbursed only after they are collected from the customer.
- Payments take approximately 1–2 business days to arrive in the seller's bank following the disbursement date. The exact timing is subject to the bank and the time zone.
- The disbursement report is updated in the AWS Marketplace Management Portal 3–5 days after the disbursement.
- Details about disbursed funds and uncollected funds are available in the monthly disbursement report, including any open account receivables.
- If you participate in the AWS Marketplace Tax Calculation Service, any US sales and use tax collected from customers will be included in your monthly disbursement.

Already a seller?

Manage your products into incremental channel revenue by taking advantage of the go-to-market activities made available in the [AWS Marketplace Management Portal](#). Activities include the following:

- Measure the results of your marketing efforts within hours, including the usage and revenue driven by your campaigns.

- Enable customer service representatives to retrieve customer data in real time.
- Upload files needed to create and manage your products, and monitor progress as we process them.

Complaints handling policy – Amazon Payments Europe

If you have any issues with the services provided by Amazon Payments Europe (APE), let us know. Your feedback helps us create a better experience for you and all of our buyers and sellers.

Note

Only complaints specific to AWS Marketplace will be addressed through the following procedure. Services provided by Amazon Payments Europe S.C.A. include, among others, processing of payment transactions, verifying the errors that may appear in the fee charges, and disbursements of funds.

Submitting a complaint

If you have an AWS Marketplace account with Amazon Payments Europe S.C.A., your complaint will be handled by Amazon Payments Europe S.C.A.

To submit a complaint

1. Sign in to your [AWS Marketplace](#) Seller account.
2. Go to **Contact Us**.
3. Select **Commercial Marketplace, Seller Account, Registration**.
4. Provide details about your complaint and choose **Submit**.

Amazon Payments Europe Complaint resolution time frames

Amazon Payments Europe S.C.A. (APE) will respond with an update to your complaint within 15 business days following the day on which it received your complaint. In exceptional circumstances beyond the control of APE, the resolution of the complaint may be extended up to 35 business days, following the day on which APE first received that complaint.

Complaint escalation

If you aren't satisfied with our response, you may choose to escalate your complaint by contacting the following:

- Amazon Payments Europe senior management

Submit your complaints by sending an email message to senior management at <management-ape@amazon.lu>. We will consider your comments carefully and respond within 15 business days following the day on which senior management received your complaint. In exceptional circumstances beyond the control of Amazon Payment Europe, the resolution of the complaint may be extended up to 35 business days, following the day on which senior management first received the complaint.

- Commission de Surveillance du Secteur Financier (CSSF)

The CSSF is the authority responsible for the prudential supervision of companies in the financial sector in Luxembourg. You can contact the CSSF at 110 Route d'Arlon L-2991 Luxembourg or use the **Contact page** at: <https://www.cssf.lu/contacts/>. To obtain further information regarding the CSSF and how to contact them, see [Customer complaints](#) on the CSSF website.

- Online Dispute Resolution

If you opened your account online in the EU, you may also have the option to refer your complaint to the CSSF by using the Online Dispute Resolution platform. This option is available because Amazon Payments Europe S.C.A. provides financial services and the CSSF is the authority responsible for its licence. For more information, see the [Online Dispute Resolution](#) platform on the European Commission website.

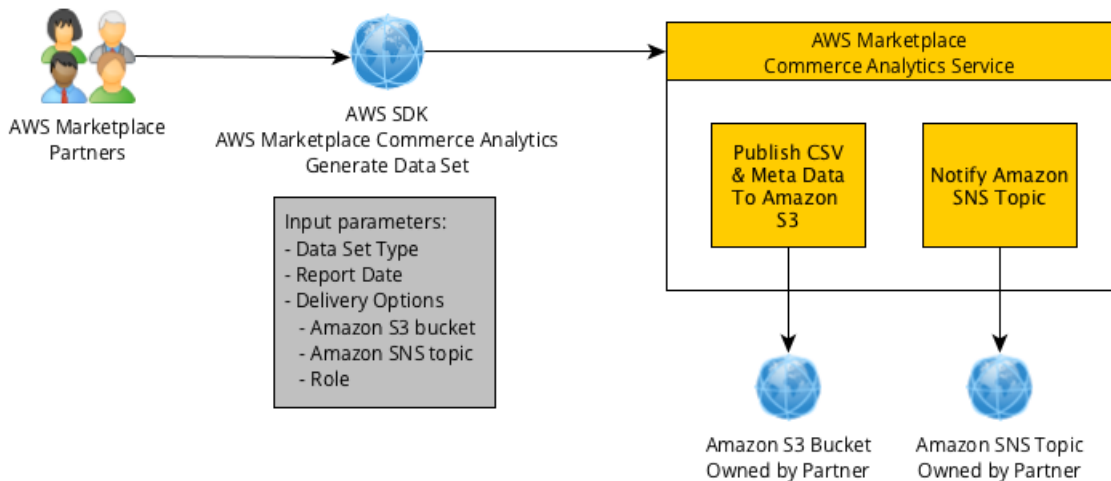
Seller toolkit

The [AWS Marketplace Management Portal](#) is your primary tool for selling products on AWS Marketplace. The following additional tools can give you more insight into your customer base and help you better understand your sales.

- [AWS Marketplace Commerce Analytics Service \(p. 14\)](#)
- [AWS Marketplace Field Demonstration Program \(p. 25\)](#)
- [Product Support Connection \(p. 25\)](#)
- [Seller reports and data feeds \(p. 273\)](#)
- [More resources in AWS Marketplace Management Portal \(p. 34\)](#)

AWS Marketplace Commerce Analytics Service

The AWS Marketplace Commerce Analytics Service lets you programmatically access product and customer data through AWS Marketplace. After you enroll in the service, you can access your usage, subscription, and billing reports through the AWS SDK.



The data you request using the SDK tools is delivered to your AWS account as datasets. Most of the datasets correspond to the same data as the text-based reports available on the [AWS Marketplace Management Portal](#). You can request datasets for a specific date, and the data is delivered to the provided Amazon S3 bucket. Notification of data delivery is provided by the Amazon Simple Notification Service (Amazon SNS).

Terms and conditions

These AWS Marketplace Commerce Analytics Service Terms and Conditions (these "**CAS Terms**") contain the terms and conditions specific to your use of and access to the AWS Marketplace Commerce Analytics

Service ("**CA Service**") and are effective as of the date you click an "I Accept" button or check box presented with these CAS Terms or, if earlier, when you use any CA Service offerings. These CAS Terms are an addendum to the Terms and Conditions for AWS Marketplace Sellers (the "**AWS Marketplace Seller Terms**") between you and Amazon Web Services, Inc. ("**AWS**," "**we**," "**us**" or "**our**"), the terms of which are hereby incorporated herein. In the event of a conflict between these CAS Terms and the AWS Marketplace Seller Terms, the terms and conditions of these CAS Terms apply, but only to the extent of such conflict and solely with respect to your use of the CA Service. Capitalized terms used herein but not defined herein shall have the meanings set forth in the AWS Marketplace Seller Terms.

1. **CA Services and CAS Data.** To qualify for access to the CA Service, you must be an AWS Marketplace Seller bound by existing AWS Marketplace Seller Terms. Information and data you receive or have access to in connection with the CA Service ("**CAS Data**") constitutes Subscriber Information and is subject to the restrictions and obligations set forth in the AWS Marketplace Seller Terms. You may use CAS Data on a confidential basis to improve and target marketing and other promotional activities related to Your AWS Marketplace Content provided that you do not (a) disclose CAS Data to any third party; (b) use any CAS Data in any way inconsistent with applicable privacy policies or law; (c) contact a subscriber to influence them to make an alternative purchase outside of the AWS Marketplace; (d) disparage us, our affiliates, or any of their or our respective products; or (e) target communications of any kind on the basis of the intended recipient being an AWS Marketplace subscriber.
2. **CA Service Limitations and Security.** You will only access (or attempt to access) the CA Service by the means described in the CA Service documentation. You will not misrepresent or mask your identity or your client's identity when using the CA Service. We reserve the right, in our sole discretion, to set and enforce limits on your use of the CA Service, including, without limitation, with respect to the number of connections, calls and servers permitted to access the CA Service during any period of time. You agree to, and will not attempt to circumvent such limitations. We reserve the right to restrict, suspend or terminate your right to access the CA Service if we believe that you may be in breach of these CAS Terms or are misusing the CA Service.
3. **CA Service Credential Confidentiality and Security.** CA Service credentials (such as passwords, keys and client IDs) are intended to be used by you to identify your API client. You are solely responsible for keeping your credentials confidential and will take all reasonable measures to avoid disclosure, dissemination or unauthorized use of such credentials, including, at a minimum, those measures you take to protect your own confidential information of a similar nature. CA Service credentials may not be embedded on open source projects. You are solely responsible for any and all access to the CA Service with your credentials.
4. **Modification.** We may modify these CAS Terms at any time by posting a revised version on the AWS Site or providing you with notice in accordance with the AWS Marketplace Seller Terms. The modified terms will become effective upon posting or, if we notify you by email, as stated in the email message. By continuing use or access the CA Service after the effective date of any modifications to these CAS Terms, you agree to be bound by the modified terms.
5. **Termination.** These CAS Terms and the rights to use CAS Data granted herein will terminate, with or without notice to you upon termination of your AWS Marketplace Seller Terms for any reason. In addition, we may stop providing the CA Services or terminate your access to the CA Services at any time for any or no reason.

Onboarding guide

You must configure your AWS account and AWS services to use the AWS Marketplace Commerce Analytics Service.

To use the AWS Marketplace Commerce Analytics Service

1. [Set up your AWS account with permissions \(p. 16\).](#)
2. [Create a destination Amazon S3 bucket \(p. 16\).](#)
3. [Configure an Amazon SNS topic for response notifications \(p. 16\).](#)

4. [Enroll in the Commerce Analytics Service program \(p. 16\)](#).
5. [Verify your configuration \(p. 16\)](#).

Set up your AWS account with permissions

AWS Marketplace **strongly** recommends using AWS Identity and Access Management (IAM) roles to sign in to the AWS Marketplace Management Portal rather than using your root account credentials. See [the section called "Policies and permissions for AWS Marketplace sellers" \(p. 346\)](#) for specific IAM permissions for AWS Marketplace Commerce Analytics Service permissions. See [Create IAM Users](#) for details. By creating individual IAM users for people accessing your account, you can give each IAM user a unique set of security credentials. You can also grant different permissions to each IAM user. If necessary, you can change or revoke an IAM user's permissions any time.

Create a destination Amazon S3 bucket

The Commerce Analytics Service delivers the data you request to an Amazon S3 bucket that you specify. If you already have an Amazon S3 bucket to use, proceed to the next step.

If you don't have an Amazon S3 bucket or you want to create an Amazon S3 bucket specifically for this data, see [How do I Create an S3 Bucket](#).

Configure an Amazon SNS topic for response notifications

The Commerce Analytics Service delivers response notifications using Amazon SNS. The service publishes messages to this topic to notify you when your datasets are available or if an error occurred. If you already have an Amazon SNS topic for this purpose, proceed to the next step.

If you don't have an Amazon SNS topic configured for this service, configure one now. For instructions, see [Create a Topic](#).

Record the topic Amazon Resource Name (ARN) for the topic you created, because the ARN is required to call the service.

Enroll in the Commerce Analytics Service program

The Commerce Analytics Service accesses the Amazon S3 bucket and Amazon SNS topic after you configure the service with the ARN for the topic and name of the bucket.

To enable access

1. Log in to the [AWS Marketplace Management Portal](#) with the AWS account you use to manage your AWS Marketplace products.
2. Navigate to the [Commerce Analytics Service enrollment page](#).
3. Enter the Amazon S3 bucket name and Amazon SNS topic ARN, and choose **Enroll**.
4. On the permissions page, choose **Allow**.
5. On the AWS Marketplace Management Portal, record the **Role Name ARN** in the success message. You need the ARN to call the service.

Note

Onboarding onto Commerce Analytics Service creates an IAM role in your AWS account that allows AWS Marketplace to write to the Amazon S3 bucket and publish notifications to the Amazon SNS Topic.

Verify your configuration

The last step is to verify that your configuration works as expected.

To test your configuration

1. Download, install, and configure the [AWS Command Line Interface \(AWS CLI\)](#).
2. Using the AWS CLI, run this command.

```
aws marketplacecommerceanalytics generate-data-set \  
--data-set-type "customer_subscriber_hourly_monthly_subscriptions" \  
--data-set-publication-date "{TODAY'S-DATE}" \  
--role-name-arn "{YOUR-ROLE-NAME-ARN}" \  
--destination-s3-bucket-name "{YOUR-S3-BUCKET}" \  
--destination-s3-prefix "test-prefix" \  
--sns-topic-arn "{YOUR-SNS-TOPIC-ARN}"
```

- For `--data-set-publication-date`, replace `{TODAY'S DATE}` with the current date using ISO-8601 format, `YYYY-MM-DDT00:00:00Z`, where `YYYY` is the four-digit year, `MM` is the two-digit month, and `DD` is the two-digit day.
- For `--role-name-arn`, replace `{YOUR-ROLE-NAME-ARN}` with the ARN of the role you received from the enrollment process in [Enroll in the Commerce Analytics Service program \(p. 16\)](#).
- For `--destination-s3-bucket-name`, replace `{YOUR-S3-BUCKET}` with the Amazon S3 bucket you created in [Create a destination Amazon S3 bucket \(p. 16\)](#).
- For `--sns-topic-arn`, replace `{YOUR-SNS-TOPIC-ARN}` with the Amazon SNS topic you created in [Configure an Amazon SNS topic for response notifications \(p. 16\)](#).

If you receive a response including the `dataSetRequestId` response from the service, you've completed the on-boarding process. A successful response looks like this:

```
{  
  "dataSetRequestId": "646dd4ed-6806-11e5-a6d8-fd5dbcaa74ab"  
}
```

Technical implementation guide

The AWS Marketplace Commerce Analytics Service is provided through the [AWS SDK](#). This guide shows you how to interact with the service using the [AWS CLI](#) and the [AWS SDK for Java](#).

IAM policy for Commerce Analytics Service

To allow your IAM users to use the Commerce Analytics Service, attach the following inline policy to your users.

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": "marketplacecommerceanalytics:GenerateDataSet",  
      "Resource": "*"   
    }  
  ]  
}
```

For more information, see [Creating Policies in the IAM console](#) in the *IAM User Guide*.

Making Requests with the AWS CLI

To get started, download the [AWS CLI](#). The following AWS CLI example makes a request for the **Hourly/Monthly Subscriptions** dataset for October 1, 2017. This dataset is published to the **demo-bucket** Amazon S3 bucket using the prefix **demo-prefix**, and the notification message is delivered to the **demo-topic** Amazon SNS topic.

```
aws marketplacecommerceanalytics generate-data-set \  
--data-set-type "customer_subscriber_hourly_monthly_subscriptions" \  
--data-set-publication-date "2017-10-01T00:00:00Z" \  
--role-name-arn "arn:aws:iam::123412341234:role/MarketplaceCommerceAnalyticsRole" \  
--destination-s3-bucket-name "demo-bucket" \  
--destination-s3-prefix "demo-prefix" \  
--sns-topic-arn "arn:aws:sns:us-west-2:123412341234:demo-topic"
```

This request returns an identifier that is unique for each request. You can use this identifier to correlate requests with notifications published to your Amazon SNS topic. The following example is an example of this identifier.

```
{  
  "dataSetRequestId": "646dd4ed-6806-11e5-a6d8-fd5dbcaa74ab"  
}
```

Making requests with the AWS SDK for Java

To start, download the [AWS Java SDK](#). The following AWS SDK for Java example makes a request for the **Hourly/Monthly Subscriptions** dataset for October 1, 2015. This dataset is published to the **demo-bucket** Amazon S3 bucket using the prefix **demo-prefix**, and the notification message is delivered to the **demo-topic** Amazon SNS topic.

```
/*  
 * Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.  
 *  
 * Licensed under the Apache License, Version 2.0 (the "License").  
 * You may not use this file except in compliance with the License.  
 * A copy of the License is located at  
 *  
 * http://aws.amazon.com/apache2.0  
 *  
 * or in the "license" file accompanying this file. This file is distributed  
 * on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either  
 * express or implied. See the License for the specific language governing  
 * permissions and limitations under the License.  
 */  
import java.text.DateFormat;  
import java.text.ParseException;  
import java.text.SimpleDateFormat;  
import java.util.Date;  
import java.util.TimeZone;  
import com.amazonaws.AmazonClientException;  
import com.amazonaws.AmazonServiceException;  
import com.amazonaws.auth.AWSCredentials;  
import com.amazonaws.auth.profile.ProfileCredentialsProvider;  
import com.amazonaws.regions.Region;  
import com.amazonaws.regions.Regions;
```

```

import
    com.amazonaws.services.marketplacecommerceanalytics.AWSMarketplaceCommerceAnalyticsClient;
import com.amazonaws.services.marketplacecommerceanalytics.model.GenerateDataSetRequest;
import com.amazonaws.services.marketplacecommerceanalytics.model.GenerateDataSetResult;
/**
 * This sample demonstrates how to make basic requests to the AWS Marketplace Commerce
 * Analytics service using the AWS SDK for Java.
 * <p>
 * <b>Prerequisites:</b> Follow the on-boarding guide: {URL OR SOMETHING}
 * <p>
 * Fill in your AWS access credentials in the provided credentials file
 * template, and be sure to move the file to the default location
 * (~/.aws/credentials) where the sample code will load the credentials from.
 * <p>
 * <b>WARNING:</b> To avoid accidental leakage of your credentials, DO NOT keep
 * the credentials file in your source directory.
 * <p>
 * http://aws.amazon.com/security-credentials
 */
public class MarketplaceCommerceAnalyticsSample {
    public static void main(String[] args) throws ParseException {
        /**
         * The ProfileCredentialsProvider will return your [default]
         * credential profile by reading from the credentials file located at
         * (~/.aws/credentials).
         */
        AWSCredentials credentials = null;
        try {
            credentials = new ProfileCredentialsProvider().getCredentials();
        } catch (Exception e) {
            throw new AmazonClientException("Cannot load the credentials from the credential profiles "
                + "file. Make sure that your credentials file is at the correct "
                + "location (~/.aws/credentials), and is in valid
                format.", e);
        }
        AWSMarketplaceCommerceAnalyticsClient client = new
            AWSMarketplaceCommerceAnalyticsClient(credentials);
        Region usEast1 = Region.getRegion(Regions.US_EAST_1);
        client.setRegion(usEast1);
        System.out.println("=====");
        System.out.println("Getting Started with AWS Marketplace Commerce Analytics Service");
        System.out.println("=====\n");
        // Create a data set request with the desired parameters
        GenerateDataSetRequest request = new GenerateDataSetRequest();
        request.setDataSetType("customer_subscriber_hourly_monthly_subscriptions");
        request.setDataSetPublicationDate(convertIso8601StringToDateUtc("2014-06-09T00:00:00Z"));
        request.setRoleNameArn("arn:aws:iam:864545609859:role/MarketplaceCommerceAnalyticsRole");
        request.setDestinationS3BucketName("awsmp-goldmine-seller");
        request.setDestinationS3Prefix("java-sdk-test");
        request.setSnsTopicArn("arn:aws:sns:us-west-2:864545609859:awsmp-goldmine-seller-topic");
        System.out.println(
            String.format("Creating a request for data set %s for publication date %s.",
                request.getDataSetType(), request.getDataSetPublicationDate()));
        try {
            // Make the request to the service
            GenerateDataSetResult result = client.generateDataSet(request);
            // The Data Set Request ID is a unique identifier that you can use to correlate the
            // request with responses on your Amazon SNS topic
            System.out.println("Request successful, unique ID: " + result.getDataSetRequestId());
        } catch (AmazonServiceException ase) {
            System.out.println("Caught an AmazonServiceException, which means your request made it "
                + "to the AWS Marketplace Commerce Analytics service, but was rejected with an "
                + "error response for some reason.");
            System.out.println("Error Message: " + ase.getMessage());
            System.out.println("HTTP Status Code: " + ase.getStatusCode());
            System.out.println("AWS Error Code: " + ase.getErrorCode());
        }
    }
}

```

```

System.out.println("Error Type: " + ase.getErrorType());
System.out.println("Request ID: " + ase.getRequestId());
} catch (AmazonClientException ace) {
System.out.println("Caught an AmazonClientException, which means the client encountered "
+ "a serious internal problem while trying to communicate with the AWS Marketplace"
+ "Commerce Analytics service, such as not being able to access the "
+ "network.");
System.out.println("Error Message: " + ace.getMessage());
}
}
private static Date convertIso8601StringToDateUtc(String dateIso8601) throws ParseException
{
TimeZone utcTimeZone = TimeZone.getTimeZone("UTC");
DateFormat utcDateFormat = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ssX");
utcDateFormat.setTimeZone(utcTimeZone);
return utcDateFormat.parse(dateIso8601);
}
}

```

You should expect results similar to this example.

```

=====
Getting Started with AWS Marketplace Commerce Analytics Service
=====
Creating a request for data set customer_subscriber_hourly_monthly_subscriptions for
publication
date Sun Jun 08 17:00:00 PDT 2014.
Request successful, unique ID: c59aff81-6875-11e5-a6d8-fd5dbcaa74ab

```

Technical documentation

The service exposes one method, `GenerateDataSet`, which enables you to request datasets to be published to your Amazon S3 bucket. The following table lists the parameters for `GenerateDataSet`.

Dataset parameters

| Field | Description |
|-----------------------------------|---|
| Data Set Type | This dataset will be returned as the result of the request. |
| Data Set Publication Date | The date a dataset was published. For daily datasets, provide a date with day-level granularity for the desired day. For monthly datasets, provide a date with month-level granularity for the desired month. The day value is ignored. |
| Role Name ARN | The ARN of the role with an attached permissions policy that provides the service with access to your resources. |
| Destination Amazon S3 Bucket Name | The name (the friendly name, not the ARN) of the destination Amazon S3 bucket. Your datasets are published to this location. |

| Field | Description |
|------------------------------|--|
| Destination Amazon S3 Prefix | <p>(Optional) The Amazon S3 prefix for the published dataset, similar to a directory path in standard file systems.</p> <p>For example, if given the bucket name <code>mybucket</code> and the prefix <code>myprefix/mydatasets</code>, the output file is published to <code>s3://DOC-EXAMPLE-BUCKET/myprefix/mydatasets/outputfile</code>.</p> <p>If the prefix directory structure doesn't exist, it's created.</p> <p>If no prefix is provided, the dataset is published to the Amazon S3 bucket root.</p> |
| SNS Topic ARN | The ARN for the Amazon SNS topic that is notified when the dataset has been published or if an error occurs. |

Responses

The AWS Marketplace Commerce Analytics service returns two responses. The first is synchronous, which is returned immediately, and the second is asynchronous, which is returned using the Amazon SNS. The synchronous response is similar to this example.

Data set parameters

| Field | Description |
|---------------------|--|
| Data Set Request ID | A unique identifier representing a specific request to the service. This identifier can be used to correlate a request with notifications on the Amazon SNS topic. |

The asynchronous response is posted as a JSON-formatted document to your Amazon SNS topic and is similar to this example.

Dataset parameters

| Field | Description |
|--------------------------------|--|
| Data Set S3 Location | The bucket name and key for the delivered dataset. |
| Data Set Meta Data S3 Location | The bucket name and key for the delivered dataset metadata file. |
| Data Set Request ID | A unique identifier representing a specific request to the service. This identifier can be used to correlate a request with notifications on the Amazon SNS topic. |
| Success | "True" if the operation succeeded; "false" if not. |

| Field | Description |
|---------|---|
| Message | (Optional) If an error occurred (for example, "Success" is "false"), this message contains information about the failure. |

Example JSON-formatted asynchronous response

```
{
  "dataSetS3Location":{
    "bucketName":"demo-bucket",
    "key":"demo-prefix/customer_subscriber_hourly_monthly_subscriptions_2014-06-09.csv"
  },
  "dataSetMetaDataSetS3Location":{
    "bucketName":"demo-bucket",
    "key":"demo-prefix/customer_subscriber_hourly_monthly_subscriptions_2014-06-09.meta.json"
  },
  "dataSetRequestId":"f65b7244-6862-11e5-80e2-c5127e17c023",
  "success":true
}
```

Outputs

After a successful request, the requested dataset is delivered to your Amazon S3 bucket as a .csv file. A JSON-formatted metadata file is published to the same location as the dataset file. The metadata file provides useful information about the dataset and original request parameters. The metadata file has the same name as the dataset file, but ends with the extension .meta.json. The following table lists the metadata fields in the .csv file.

Metadata fields

| Field | Description |
|--------------------------------|--|
| Data Set Request ID | A unique identifier representing a specific request to the service. This identifier can be used to correlate a request with notifications on the Amazon SNS topic. |
| Data Set Coverage Range | Defines the start date/time and end date/time for the data coverage range. These dates are in ISO 8601 format. |
| Data Set Request Parameters | The original request parameters to the GenerateDataSet method. |
| Data Set S3 Location | The bucket name and key for the delivered dataset. |
| Data Set Meta Data S3 Location | The bucket name and key for the delivered dataset metadata file. |

Following is an example of JSON-formatted metadata contents.

```
{
```

```
"dataSetRequestId": "43d7137b-8a94-4042-a09d-c41e87f371c1",
"dataSetCoverageRange": {
  "startDateTime": "2014-06-08T00:00:00.000Z",
  "endDateTime": "2014-06-08T23:59:59.000Z"
},
"dataSetRequestParameters": {
  "sellerAccountId": "123412341234",
  "dataSetType": "customer_subscriber_hourly_monthly_subscriptions",
  "dataSetPublicationDate": "2014-06-09T00:00:00.000Z",
  "roleNameArn": "arn:aws:iam::123412341234:role/MarketplaceCommerceAnalyticsRole",
  "destinationS3BucketName": "demo-bucket",
  "destinationS3Prefix": "demo_prefix/customer_subscriber_hourly_monthly_subscriptions",
  "snsTopicArn": "arn:aws:sns:us-west-2:123412341234:demo-topic"
},
"dataSetS3Location": {
  "bucketName": "demo-bucket",
  "key": "demo_prefix/customer_subscriber_hourly_monthly_subscriptions_2014-06-09.csv"
},
"dataSetMetaDataS3Location": {
  "bucketName": "demo-bucket",
  "key": "demo_prefix/customer_subscriber_hourly_monthly_subscriptions_2014-06-09.meta.json"
}
}
```

For a complete list of available datasets, including availability dates, refer to the [AWS SDK documentation](#).

Troubleshooting

This sections describes solutions to issues you may encounter with using the AWS Marketplace Commerce Analytics Service.

I can't access the service because of an allow list issue.

If you're not yet registered as a seller on the AWS Marketplace, visit [AWS Marketplace Management Portal](#) to register. If you have already registered as a seller on AWS Marketplace, contact the [AWS Marketplace Seller Operations](#) team.

I can't request datasets for a date in the past, even though the SDK documentation says it should be available for this date.

Even though datasets are listed as being available for certain dates in the past, we have data only since the time that you joined AWS Marketplace. If you believe that this is in error, contact the [AWS Marketplace Seller Operations](#) team.

When I call the service, I receive the error message "Could not connect to the endpoint URL: https://marketplacecommerceanalytics.eu-central-1.amazonaws.com/"

The AWS Marketplace Commerce Analytics Service is available only in the US East (N. Virginia) Region. You must make all calls to the Commerce Analytics Service to the `us-east-1` endpoint.

If you're using the AWS CLI, add the `--region` flag to each call and specify the AWS Region as `us-east-1`, as shown in the following example.

```
aws marketplacecommerceanalytics generate-data-set \
--data-set-type "customer_subscriber_hourly_monthly_subscriptions" \
--data-set-publication-date "2016-04-21T00:00:00Z" \
--role-name-arn "arn:aws:iam::138136086619:role/MarketplaceCommerceAnalyticsRole" \
--destination-s3-bucket-name "marketplace-analytics-service" \
--destination-s3-prefix "test-prefix" \
```



```
--sns-topic-arn "arn:aws:sns:eu-central-1:138136086619:Marketplace_Analytics_Service_Notice" \  
--region us-east-1
```

I want to use a different Amazon S3 bucket or Amazon SNS topic than the ones I selected when I went through the on-boarding process.

When enrolling in the AWS Marketplace Commerce Analytics Service, you specified an Amazon S3 bucket and Amazon SNS topic. The onboarding process configures your IAM permissions to allow the service access to only these specific resources. To use different resources, you need to modify your IAM policy:

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. Choose **Roles** on the left side of the IAM console.
3. Choose **MarketplaceCommerceAnalyticsRole**.
4. Expand the **Inline Roles** section, if not already expanded.
5. Locate the policy with a name that starts with *oneClick_MarketplaceCommerceAnalyticsRole* and choose **Edit Policy**.
6. In the policy document, locate the section that specifies actions related to the service that you want to modify. For example, to change your Amazon S3 bucket, locate the section that includes the actions that start with **s3:** and change their respective **Resource** selection to specify your new Amazon S3 bucket.

For additional information about IAM policies, see the following guide: https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies.html

I get an `AccessDeniedException` error when I call the `GenerateDataSet` action

This can happen if your IAM user doesn't have the permissions necessary to call `GenerateDataSet`. The following procedure outlines the steps needed to update an IAM policy with those permissions using the IAM console.

To get the `GenerateDataSet` permissions

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. From the navigation pane on the right, choose **Users**.
3. Choose the IAM user whose credentials you want to use for the `marketplacecommerceanalytics` AWS CLI commands to open the **Summary** page.
4. From the **Permissions** tab, choose **Add inline policy**
5. Open the **JSON** tab and paste the following code:

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": "marketplacecommerceanalytics:GenerateDataSet",  
      "Resource": "*" }  
  ]  
}
```

6. Choose **Review policy**, provide the inline policy with a descriptive name, like *GenerateDataSetPolicy*, and choose **Create policy**.

After updating the permissions, run the AWS CLI command again with the same credentials as this IAM user to complete the action.

For more information, see [Creating Policies in the IAM console](#) in the *IAM User Guide*.

My problem isn't listed here.

Contact the [AWS Marketplace Seller Operations](#) team.

AWS Marketplace Field Demonstration Program

The AWS Marketplace Field Demonstration Program (FDP) allows the AWS field team (internally approved AWS employees) to use some products and solutions through AWS Marketplace at no charge.

Examples of approved AWS employees may include solutions architects and sales and marketing professionals. The FDP allows these employees to demonstrate product capabilities for education and potential inclusion in customer workloads.

The following product types are supported:

- [Amazon Machine Images \(AMIs\)](#) (p. 63)
- [Containers](#) (p. 116)
- [Machine learning algorithms and model packages \(SageMaker\)](#) (p. 176)
- [Data sets \(AWS Data Exchange\)](#)

Note

For AWS Data Exchange products, the FDP only applies to products with a public offer of \$0 (free).

For AWS Data Exchange products that have subscription verification enabled, providers need to approve the subscription request. For more information about subscription verification, see [Subscription verification for subscribers](#) in the *AWS Data Exchange User Guide*.

You're automatically enrolled in the FDP program when you sign up as an AWS Marketplace seller. To opt out, submit a support request to the [Managed Catalog Operations \(MCO\)](#) team.

To view information about product usage under this program, see the [AWS field demonstration usage](#) (p. 293) section of the [Monthly billed revenue report](#) (p. 288).

Product Support Connection

AWS Marketplace Product Support Connection (PSC) is a feature that enables AWS Marketplace customers to provide contact information in the AWS Marketplace website for the purposes of obtaining and accessing product support from AWS Marketplace Sellers. AWS Marketplace shares the provided data with participating Sellers via an API to enable a better support experience. Customers can choose to add contact details during or after a purchase of PSC-enabled AWS Marketplace products, and Sellers can retrieve the Customer contact data, along with relevant product subscription details, by calling a pull-based API.

Your staff can use the Customer Support Eligibility tool to access near-real-time information about a customer's subscription to your products and provide fast, personalized service. AWS Marketplace Management Portal makes it easy to get started: Enter a customer's AWS account ID to retrieve subscription and usage information from their account.

You also have the option to enroll your products in AWS Marketplace PSC. For products that are enrolled in PSC, AWS Marketplace customers can choose to provide contact information (including name, organization, email address, and phone number) through the AWS Marketplace website for the purposes of obtaining and accessing product support. If you enroll in PSC, AWS Marketplace shares the provided data with you through an API to help enable a more seamless support experience.

Note

Data products don't support this feature.

Technical implementation guide

This section covers API specification details and how to onboard with the PSC feature. The PSC `start-support-data-export` API operation is part of the AWS Marketplace Commerce Analytics Service (CAS). To integrate with the API for PSC, you must first enroll in CAS. If you are already enrolled in CAS, use the same AWS Identity and Access Management (IAM) role that you created when you onboarded.

IAM policy for PSC

To allow your IAM users to access the AWS Marketplace PSC feature, you must attach the following inline policy to your users.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "marketplacecommerceanalytics:StartSupportDataExport",
      "Resource": "*"
    }
  ]
}
```

For more information, see [Creating Policies in the IAM console](#) in the *IAM User Guide*.

Making requests with the AWS CLI

You can request an export of the PSC data using the [AWS CLI](#) or any of the [AWS Software Development Kits \(SDKs\)](#).

If you have already been using CAS to call the `generate-data-set` operation, you must use the same IAM role for both `generate-data-set` and `start-support-data-export`.

To ensure the security of the customer contact data available through the Product Support Connection program, we recommend that the Amazon Simple Storage Service (Amazon S3) bucket you use for `start-support-data-export` be separate from the S3 bucket you use for `generate-data-set`. Verify the permissions on your IAM role allow access to all S3 buckets you intend to use.

```
aws marketplacecommerceanalytics start-support-data-export
--data-set-type "test_customer_support_contacts_data" \
--from-date "{START-DATE}" \
--role-name-arn "{YOUR-ROLE-NAME-ARN}" \
--destination-s3-bucket-name "{YOUR-S3-BUCKET}" \
--destination-s3-prefix "test-prefix" \
--sns-topic-arn "{YOUR-SNS-TOPIC-ARN}"
```

A successful response from the service returns the `dataSetRequestId` of the request.

Example

```
{
  "dataSetRequestId":
    "646dd4ed-6806-11e5-a6d8-fd5dbcaa74ab"
```

}

API request parameters and responses

StartSupportDataExport method

The `StartSupportDataExport` method allows you to request contact details that customers have submitted for your PSC-enabled products. Data is exported from the start date specified in the request up to 15 minutes prior to the time of the request. A successful request results in the dataset being published to the Amazon Simple Storage Service(Amazon S3) bucket specified.

At this time, you can query the API to request the `test_customer_support_contacts_data` dataset. This will export a static test dataset containing data that does not correspond to any real customer data. You should use the test data for testing and integration. The `customer_support_contacts_data` option, which will return the real customer contact data for your PSC-enabled products, will not be available until after the General Availability of this feature later in 2016.

Request parameters

| Input | Description |
|---------------|--|
| Data Set Type | <p>The type of dataset requested to be exported. Valid options for datasets are:</p> <p><code>test_customer_support_contacts_data</code></p> <p><code>customer_support_contacts_data</code></p> <p>The <code>test_customer_support_contacts_data</code> dataset provides sample data for testing and integration purposes and is available immediately. The <code>customer_support_contacts_data</code> dataset is currently unavailable. This option will contain actual customer data and be available upon general availability of PSC.</p> |
| From Date | <p>The earliest date of data to be exported. The exported data will contain information from the specified From Date to 15 minutes prior to the time of the request.</p> <p>The From Date must be expressed as an ISO 8601 date/time string.</p> <p>If you would like to receive the full data set, as opposed to a set of updates, specify any date prior to the date when you onboarded to the program. To receive only incremental data since your last request, specify the <code>endDateTime</code> from the <code>dataSetCoverageRange</code> from the metadata JSON file resulting from your previous request. See below for more information about the metadata JSON file.</p> |
| Role Name ARN | <p>The Amazon Resource Name (ARN) of the IAM role with an attached permissions policy which provides the service with access to your resources.</p> |

| Input | Description |
|----------------------------|--|
| Destination S3 Bucket Name | The name (friendly name, not ARN) of the destination Amazon S3 bucket . Your datasets will be published to this location. |
| Destination S3 Prefix | <p>(Optional) The desired Amazon S3 prefix for the published dataset, similar to a directory path in standard file systems.</p> <p>For example, if given the bucket name "mybucket" and the prefix "myprefix/mydatasets", the output file "outputfile" would be published to "s3://<i>DOC-EXAMPLE-BUCKET</i>/myprefix/mydatasets/outputfile".</p> <p>If the prefix directory structure does not exist, it will be created.</p> <p>If no prefix is provided, the data set will be published to the Amazon S3 bucket root.</p> |
| SNS Topic ARN | The Amazon Resource Name (ARN) for the Amazon SNS topic that will be notified when the data set has been published, or if an error occurs. |

Responses

Calls to the API will immediately return a response with the Data Set Request ID.

| Field | Description |
|---------------------|--|
| Data Set Request ID | A unique identifier representing a specific request to the service. This identifier can be used to correlate a request with notifications on the Amazon SNS topic. |

An additional response containing metadata will be posted to the Amazon Simple Notification Service (Amazon SNS) topic specified in the original request. The contents of the post are detailed in the following table.

| Field | Description |
|--------------------------------|--|
| Data Set S3 Location | The bucket name and key for the delivered dataset. |
| Data Set Meta Data S3 Location | The bucket name and key for the delivered dataset meta data file. |
| Data Set Request ID | A unique identifier representing a specific request to the service. This identifier can be used to correlate a request with notifications on the Amazon SNS topic. |
| Success | "True" if the operation succeeded; "false " if not. |

| Field | Description |
|---------|---|
| Message | (Optional) If an error occurred (for example, "Success" is "false"), this message will contain information about the failure. |

The metadata file is JSON-formatted and contains the following fields.

| Field | Description |
|--------------------------------|--|
| Data Set Request ID | A unique identifier representing a specific request to the service. This identifier can be used to correlate a request with notifications on the Amazon SNS topic. |
| Data Set Coverage Range | Defines the start date / time and end date / time for the data coverage range. These dates are in ISO 8601 format. |
| Data Set Request Parameters | The original request parameters to the GenerateDataSet method. |
| Data Set S3 Location | The bucket name and key for the delivered dataset. |
| Data Set Meta Data S3 Location | The bucket name and key for the delivered dataset metadata file. |
| Request Received Date Time | The date/time at which the request was received, in ISO 8601 format. |
| Request Completed Date Time | The date/time at which the request was completed, in ISO 8601 format. |

Example JSON-formatted metadata contents

```
{
  "dataSetRequestId": "c3c84ee0-5aba-11e6-8d9c-235dc080841d",
  "dataSetCoverageRange": {
    "startDateTime": "2016-08-18T00:00:00.000Z",
    "endDateTime": "2016-08-05T03:14:50.334Z"
  },
  "dataSetRequestParameters": {
    "fromDate": "2016-08-18T00:00:00.000Z",
    "dataSetType": "test_customer_support_contacts_data",
    "roleNameArn": "arn:aws:iam::123456789012:role/MarketplaceCommerceAnalyticsRole",
    "destinationS3BucketName": "mybucket",
    "destinationS3Prefix": "mydata",
    "snsTopicArn": "arn:aws:sns:us-west-2:123456789012:mynotification"
  },
  "dataSetS3Location": {
    "bucketName": "mybucket",
    "key": "mydata/
test_customer_support_contacts_data_2015-01-18T00-00-00Z_to_2016-08-05T03-14-50Z.csv"
  },
  "dataSetMetaDataSetS3Location": {
    "bucketName": "mybucket",
    "key": "mydata/
test_customer_support_contacts_data_2015-01-18T00-00-00Z_to_2016-08-05T03-14-50Z.meta.json"
  }
}
```

```

    },
    "requestReceivedDateTime": "2016-08-05T03:14:50.108Z",
    "requestCompletedDateTime": "2016-08-05T03:14:50.334Z"
  }

```

Output data format

The output data contains customer contact records, product code, product ID, subscription start date, and the AWS account ID of the customer. A summary of the fields is shown in the following table. Each output file contains a comma-separated header, followed by the records containing customer data and subscription information. Each record contains a "Create", "Update", or "Delete" operation type to indicate whether the record is newly created, modified, or deleted since the "From Date" indicated in the API request. The overall file format adheres to the RFC4180 standard.

If multiple operations have occurred on a record in the time frame specified by the "from-date" parameter API request, only the most recent data will be reflected or exported. For example, if a customer creates and then updates a record, the record returned will be different depending on the specified "from-date". If the "from-date" is prior to the date at which the record was created, only a CREATE record will be passed in the output data set, and the record will reflect the most recently entered details. If the "from-date" is after the record was created, but before it was updated, only an UPDATE record will be passed in the output data set. If the from-date is after the record was updated, no record will be passed. Likewise, if a customer creates and then deletes a record, only the "DELETE" will appear in the output file.

If you would like to receive the full dataset, as opposed to a set of updates, specify any date prior to the date when you onboarded to the program. To receive only incremental data since your last request, specify the `endTime` from the `dataSetCoverageRange` from the metadata JSON file resulting from your previous request.

| Field | Format | Description |
|-------------------|----------------------------------|---|
| Product ID | 36-character hexadecimal string | Unique identifier for the product in AWS Marketplace (GUID). Required field; always appears in every record. |
| Product Code | 25-character alphanumeric string | Unique identifier for the product, associated with billing and available in Amazon Elastic Compute Cloud (Amazon EC2) instance metadata. Required field; always appears in every record. |
| Customer Guid | 36-character hexadecimal string | Unique GUID identifying the customer contact data record. This will be unique for each record that appears in the output file. Required field; always appears in every record. |
| Subscription Guid | 36-character hexadecimal string | Unique GUID corresponding to the customer's product subscription. A customer can |

| Field | Format | Description |
|-------------------------|---|--|
| | | <p>have multiple subscriptions to the same product.</p> <p>Required field; always appears in every record.</p> |
| Subscription Start Date | <p>ISO 8601 date/time, with UTC time zone.</p> <p>The format is YYYY-MM-DDTHH:mm:ss.nnnZ, where YYYY is year, MM is month, DD is day, HH is hour from 00-23, mm is minute of hour from 00-59, ss is second of minute from 00-59, and nnn is millisecond of second from 000-9999, such as "2016-04-07T14:05:15.275Z"</p> | <p>Start date of the customer's product subscription.</p> <p>Required field; always appears in every record.</p> |
| Organization | String with a maximum length of 255 characters | <p>Organization name provided by the customer.</p> <p>Always appears in records with operation type "Update" or "Create." Does not appear in records with operation type "Delete."</p> |
| AWS Customer Id | 12-digit numeric string which may include leading zeroes | <p>The AWS customer ID for the customer subscribed to the product.</p> <p>Required field; always appears in every record.</p> |
| Given Name | String with a maximum length of 100 characters | <p>Given name or first name for the point of contact provided by the customer.</p> <p>Always appears in records with operation type "Update" or "Create." Does not appear in records with operation type "Delete."</p> |
| Surname | String with a maximum length of 100 characters | <p>Surname (family name or last name) for the point of contact provided by the customer.</p> <p>Always appears in records with operation type "Update" or "Create." Does not appear in records with operation type "Delete."</p> |

| Field | Format | Description |
|------------------|---|---|
| Telephone Number | String with a maximum length of 25 characters. May include international phone numbers. | <p>Telephone number provided by the customer.</p> <p>Always appears in records with operation type "Update" or "Create." Does not appear in records with operation type "Delete."</p> |
| Email | String with a maximum length of 254 characters | <p>Email address provided by the customer.</p> <p>Always appears in records with operation type "Update" or "Create." Does not appear in records with operation type "Delete."</p> |
| Title | String with a maximum length of 255 characters | <p>Job title provided by the customer.</p> <p>Optional field. Will sometimes occur in records with operation type "Update" or "Create." Does not appear in records with operation type "Delete."</p> |
| Country Code | 2-character ISO 3166 country code | <p>Country code provided by the customer.</p> <p>Optional field. Will sometimes occur in records with operation type "Update" or "Create." Does not appear in records with operation type "Delete."</p> |
| ZIP Code | 5-digit string | <p>Zip code provided by the customer; applicable to USA only.</p> <p>Optional field. Will sometimes occur in records with operation type "Update" or "Create." Does not appear in records with operation type "Delete."</p> |

| Field | Format | Description |
|----------------|--|---|
| Operation Time | ISO 8601 date/time, with UTC time zone. The format is YYYY-MM-DDTHH:mm:ss.nnnZ (YYYY is year, MM is month, DD is day of month, HH is hour of day from 00-23, mm is minute of hour from 00-59, ss is second of minute from 00-59 and nnn is millisecond of second from 000-999), such as "2016-04-07T14:05:15.275Z" | Indicates the date/time when the record was most recently created, updated, or deleted by the customer. Required field; always appears in every record. |
| Operation Type | String; possible values are "CREATE", "UPDATE", or "DELETE" | CREATE: Indicates that the record has been newly created since the from-date specified in the API request. UPDATE: Indicates that the record has been updated since the from-date specified in the API request. DELETE: Indicates that the record has been deleted since the from-date specified in the API request. Required field; always appears in every record. |

An example of the output file format is shown below.

Product Id,Product Code,Customer Guid,Subscription Guid,Subscription Start Date,Organization,AWS Customer Id,Given Name,Surname,Telephone Number,Email,Title,Country Code,ZIP Code,Operation Time,Operation Type

4b898955-84fa-4cfb-8f43-98287ad69c06,4gzp2symm0v9zidfrn9f854w6,ba1d75cc-d984-4f07-bb14-ae04b952afbc,cad371fb-6f2c-4537-a054-1a7afc6312fd,2016-05-27T00:00:00.000Z,Example Inc -- Service Division,000011112222,Eugene,Thietmar,555-947-8228,eugethi@example.org,,,,,2016-05-12T03:54:46.143Z,CREATE

4b898955-84fa-4cfb-8f43-98287ad69c06,4gzp2symm0v9zidfrn9f854w6,1b4a2b5f-2c5d-4779-b0c7-2878b0f45cfc,cad371fb-6f2c-4537-a054-1a7afc6312fd,2016-05-19T00:00:00.000Z,Example Inc -- Service Division,000011112222,Angela,Doe,555-294-4528,adoe@example.com,,US,02201,2016-05-19T18:21:06.834Z,CREATE

cade58ff-ff82-4770-b84b-0bd399bf1c6d,c0dcyyqczbk5uc62acmp6450t,6c83ff14-5167-43cc-bb9f-24865a78db72,c2f40319-8fc2-409a-884b-2f85adf9e29c,2015-12-01T00:00:00.000Z,Example Inc -- European Sales Division,111122223333,Ravi,Smith,555-111-1010,ravis@example.com,Head of IT,ES,,2016-04-07T14:05:15.145Z,CREATE

4b898955-84fa-4cfb-8f43-98287ad69c06,4gzp2symm0v9zidfrn9f854w6,1b4a2b5f-2c5d-4779-b0c7-2878b0f45cfc,cad371fb-6f2c-4537-a054-1a7afc6312fd,2016-05-01T00:00:00.000Z,,000011112222,,,,,,2016-04-22T14:36:24.054Z,DELETE

3f4300eb-bfa0-4610-8d68-d8ba71baaa50,3qtu9xydxldrj8c5jyldy1lqo,91c72621-6cf4-4d69-8ebe-073ff4f8ab9e,d118eb96-55ce-4752-909c-eedcfdcd6647,2015-11-30T00:00:00.000Z,Example Inc -- Design Division,333344445555,Nathan,Zhenyuan,555-2222-1010,nathanz@example.com,Sr. Program Manager,US,98109,2016-04-07T14:05:15.275Z,CREATE

3f4300eb-bfa0-4610-8d68-d8ba71baaa50,3qtu9xydxldrj8c5jyldy1lqo,2ae0be12-7397-4fdb-a1c7-ead17967002c,d118eb96-55ce-4752-909c-eedcfdcd6647,2016-05-01T00:00:00.000Z,Example Inc -- Design Division,333344445555,Abdul,Alves,555-676-8989,abdal@example.com,,,,2016-05-11T05:26:51.000Z,UPDATE

...

Note

When a customer deletes their contact information from the PSC program, you will see a record in the output .csv file that indicates an operation type "DELETE." After a customer deletes their data, the API no longer transmits contact information such as name, telephone number, email, and so forth. Each delete record consists of the data required to uniquely identify the record to be deleted. Delete records contain product ID, product code, operation time, customer GUID, subscription GUID, subscription start date, AWS Customer ID, operation time, and operation type.

If a customer opts out of Product Support Connection by deleting their contact information, you should also remove the contact information from your records. Because the customer contact data will not be included in the DELETE record, you will need to look up the record in your system by using the unique Customer GUID.

A delete record will also be sent if a customer terminates a subscription.

If you have questions or would like more information about participating in AWS Marketplace Product Support Connection, contact the [AWS Marketplace Seller Operations](#) team.

More resources in AWS Marketplace Management Portal

There are more resources available to you in the AWS Marketplace Management Portal. If you open the [AWS Marketplace Management Portal](#) and sign in, you can see links to additional resources on the home page, in the **Marketplace Resources** section.

For example, to get support for marketing your product in the 90 days leading up to launch and the 90 days after launch, you can see the [180-day GTM Academy \(p. 268\)](#) that is linked from the AWS Marketplace Management Portal home page under **Marketplace Resources**.

Preparing your product

Preparing to publish a product on AWS Marketplace includes configuring your package, setting a pricing scheme, determining the relevant categories in which to list your product, and adding keywords so your product appears in relevant searches.

Topics

- [Product delivery \(p. 35\)](#)
- [Product pricing \(p. 37\)](#)
- [Regions and countries for your AWS Marketplace product \(p. 43\)](#)
- [Private offers \(p. 44\)](#)
- [Standardized license terms \(p. 54\)](#)
- [Categories and metadata \(p. 56\)](#)
- [Search engine optimization for products \(p. 58\)](#)

Product delivery

Each product delivery method has several options for packaging, pricing, and delivery. Some methods aren't available to you as a seller on AWS Marketplace until you register for the program supporting it.

You can create products with a standard list price and end user license agreement (EULA). You can also create private offers for individual customers with custom pricing and EULAs. If you need to make additional changes to the terms of the contract, you can work with the AWS Marketplace team to create a custom private offer.

Tip

To simplify the procurement process, you can use [standardized license terms \(p. 54\)](#) for both public product listings and private offers.

The following table lists the methods that you can use to deliver software products and how AWS Marketplace buyers find each type of deliverable in the AWS Marketplace console.

Product delivery methods

| Product delivery method | Delivery Method filter on the console | Description |
|-------------------------|---------------------------------------|---|
| Single AMI | Amazon Machine Image (AMI) | <p>You deliver a single custom Amazon Machine Image (AMI) for your product. The AMI provides the information required to launch an Amazon Elastic Compute Cloud (Amazon EC2) instance.</p> <p>Buyers can use the single AMI to create Amazon EC2 instances with your product already installed and ready to use.</p> <p>For more information, see AMI-based products (p. 63).</p> |

| Product delivery method | Delivery Method filter on the console | Description |
|--|---------------------------------------|---|
| AMI delivered using AWS CloudFormation templates | CloudFormation Template | <p>You can list AMI-based products that are delivered to AWS Marketplace buyers by using CloudFormation templates.</p> <p>Buyers can purchase a single solution that entitles them to all of the AMIs in that product.</p> <p>For more information about delivering AMIs as an CloudFormation template, see AMI-based delivery using AWS CloudFormation.</p> <p>For more information about CloudFormation templates, see AWS CloudFormation concepts in the <i>AWS CloudFormation User Guide</i>.</p> |
| Private image build | Private Image Build | <p>You offer products in a way that lets buyers install your product on a base gold image that meets their internal standards for operating system configuration.</p> <p>For more information, see Private images (p. 84).</p> |
| Container-based product or application | Container | <p>You deliver products packaged in container images. Container products consist of options, which are a set of container images and deployment templates that work together.</p> <p>For more information, see Container-based products (p. 116).</p> |
| Data products | AWS Data Exchange | <p>You use AWS Data Exchange to create data products.</p> <p>For information about publishing and managing data products and offers through AWS Data Exchange, see Providing data products on AWS Data Exchange in the <i>AWS Data Exchange User Guide</i>.</p> |

| Product delivery method | Delivery Method filter on the console | Description |
|--|---------------------------------------|---|
| Machine learning algorithms and model packages | SageMaker Model | <p>You use Amazon SageMaker to create the algorithm or model package, and then publish it on AWS Marketplace.</p> <p>For more information about delivering machine learning algorithms and model packages, see Machine learning products (p. 176).</p> <p>For information about SageMaker, see What is SageMaker? in the <i>Amazon SageMaker Developer Guide</i>.</p> |
| Software as a service (SaaS) | SaaS | <p>You can offer SaaS products with subscription- based, contract-based, or contract with consumption pricing models.</p> <p>For more information, see Software as a service (SaaS)-based products (p. 214).</p> |
| Professional services | Professional Services | <p>You can offer professional services that support or work with other AWS Marketplace products.</p> |

Product pricing

This topic provides general pricing information about software products in AWS Marketplace. All pricing is based on US dollars (USD).

For paid products, AWS Marketplace collects software charges from the customer.

There is no service fee for free or open-source Software that is made available to customers without charge.

For information about refunds, see [Refunds](#) (p. 41).

Topics

- [Pricing models](#) (p. 38)
- [Changing pricing models](#) (p. 40)
- [Changing prices](#) (p. 41)
- [Private offers](#) (p. 41)
- [Refunds](#) (p. 41)

Pricing models

The following topics provide general information about the pricing models available in AWS Marketplace.

Topics

- [Annual pricing \(p. 38\)](#)
- [Usage pricing \(p. 39\)](#)
- [Contract pricing \(p. 40\)](#)
- [Bring Your Own License pricing \(p. 40\)](#)

For information about the pricing models for specific product delivery methods, see:

- [AMI product pricing \(p. 91\)](#)
- [Container product pricing \(p. 129\)](#)
- [Machine learning product pricing \(p. 178\)](#)
- [SaaS product pricing \(p. 227\)](#)
- [Professional services product pricing \(p. 259\)](#)

Annual pricing

An annual pricing model enables you to offer products to customers who can purchase a 12-month subscription. The subscription pricing can provide up to 40 percent savings compared to running the same product hourly for extended periods. The customer is invoiced for the full amount of the contract at the time of subscription. For more information about how annual subscriptions are presented to customers, see [AMI Subscriptions](#).

Considerations when working with an annual subscription include the following:

- Annual pricing is defined per instance type. It can be the same for all Amazon Elastic Compute Cloud (Amazon EC2) instance types or different for each instance type.
- All Annual instance types must also have an Hourly instance type defined. AWS Marketplace doesn't offer Annual-only pricing or Hourly without Annual on the same product. For any product offering Annual pricing, Hourly pricing also needs to be specified.
- A \$0 Annual price is allowed on a specific instance type, if the Hourly price is also \$0 and there are other non-\$0 Annual instance types defined.
- At the end of the annual subscription period, the customer will start being charged at the hourly price.
- If a customer buys X Annual subscriptions but is running Y software on Y instances, then the customer is charged at Hourly software price for (Y-X) instances which are not covered by Annual subscriptions. As such, an Hourly rate must be included for all Annual pricing instance types.
- Using seller private offers, you can offer a multi-year (up to 3 years) or custom duration AMI with upfront payment, or a flexible payment schedule. For more information about multi-year and custom duration contracts, see [Private offers \(p. 44\)](#) and [the section called "Flexible payment scheduler" \(p. 48\)](#).

If you offer an Annual product in AWS Marketplace, you agree to the specific refund policies for Annual products, located in the **File Uploader** documents section in the [AWS Marketplace Management Portal](#).

Price change

You can change annual prices (the \$ value, for example \$1,000/year to \$1,200/year) whenever you want. However, you must give 90 days' notice to existing customers of annual pricing. The new price will apply to new subscriptions but will have no impact on existing subscriptions.

Price changes will be effective for auto-renewals only if the price was changed at least 90 days before the auto-renewal date. The customer will receive an email message prior to auto-renewal that includes the new price.

End user license agreement

An AWS customer's usage of software for 12 months under an annual subscription is covered by the EULA that you provide on your product's details page on AWS Marketplace.

Usage pricing

A usage pricing model, also known as *pay as you go* pricing, enables you to offer products to customers who only pay for what they use.

As a seller, you can choose one of the following usage categories:

- **Users**
- **Hosts**
- **Bandwidth**
- **Data**
- **Tiers**
- **Units** (for custom categories)

You can also define up to 24 dimensions for the product. Charges are measured and reported when the API is called by the software. We recommend that sellers configure the API to be called once per hour as a best practice, depending on their use case. All usage is calculated monthly and billed monthly using the same mechanism as existing AWS Marketplace software.

Using the AWS Marketplace Metering Service, you can handle several new pricing scenarios.

Example Charge by Host

If your software monitors hosts, you can charge for each host monitored and set different pricing based on the host size.

Example Charge by User

If your software allows multiple users across an organization, you can charge by user. Each hour, the customer is charged for the total number of provisioned users.

Note

In the Product Load Form (PLF), relevant columns are preceded with "FCP" (Flexible Consumption Pricing). For example: **FCP Category (Custom Pricing Category)**.

For AWS Marketplace Metering Service products, note the following:

- If your software is already on AWS Marketplace, you will need to create a product to enable an alternate usage dimension. You can't convert a standard product to use the AWS Marketplace Metering Service. After the new product is published, you can remove the old product or keep both on the website.
- The AWS Marketplace Metering Service requires that your software reports usage every hour, recording the customer usage for the hour. If there is a failure in the transmission or receipt of metering service

records, AWS will be unable to bill for such usage. You are responsible for ensuring the successful receipt of metering records.

- Products that use the AWS Marketplace Metering Service don't support 1-Click. Buyers are required to launch your software with an AWS Identity and Access Management (IAM) role with specific permissions and have an internet gateway.
- Free Trial and Annual Pricing aren't compatible with the AWS Marketplace Metering Service.
- Changing dimension (user, hosts, bandwidth, and data) or dimension name isn't supported. You will need to create a new product.

Contract pricing

Using the contract pricing model, you can offer upfront pricing to customers that enables them to buy a license for 1 month, 12 months, 24 months, or 36 months.

Contract pricing is available for the following products:

- Single AMI-based products and AMI with AWS CloudFormation template-based products. For more information, see [Contract pricing for AMI products \(p. 94\)](#)
- Container-based products. For more information, see [Contract pricing for container products \(p. 131\)](#).
- Software as a service (SaaS)-based products. For more information, see [Pricing for SaaS contracts \(p. 229\)](#).

Note

Contract pricing for AMI and container-based products is only for new products.

If you have an existing AMI or container-based product and want to use contract pricing, create a new listing and then apply the contract pricing model by using the Product Load Form (PLF) to add different dimensions, integrate the AMI or container-based product with AWS License Manager, and then publish the AMI or container-based product.

When a customer purchases a product with contract pricing, a license is created by AWS Marketplace in the customer AWS account that your software can check using the License Manager API. Customers will need an IAM role to launch an instance of the AMI or container-based product.

Bring Your Own License pricing

There is no service fee for Bring Your Own License (BYOL) products on AWS Marketplace.

To deliver on our customer promise of selection, we require that all BYOL products also have a paid option. This is so that customers who don't have existing licenses have the option to purchase and use the products.

For BYOL products, we realize that the online purchase of software is a departure from how some companies do business. Therefore, for the first 90 days after launch, we will relax the requirement that this software is accompanied by a version available for purchase on AWS Marketplace. During this time, the AWS Marketplace account management teams will work with you to address challenges. The team can help you to determine if and how the software can be made available for purchase on AWS Marketplace.

Changing pricing models

Changes to pricing models must be reviewed and approved by AWS Marketplace to ensure a positive customer experience and reduced risk to all parties. Discuss the pricing model changes you want to make by contacting the [AWS Marketplace Managed Catalog Operations \(MCO\)](#) team.

All requests for pricing model changes can take 30–90 days to process and review.

Changing prices

You can update prices and metadata through the AWS Marketplace Management Portal.

To change prices

1. Sign in to the [AWS Marketplace Management Portal](#).
2. In the **Products** tab, a list of current products that you created is available. In the table for your current products, choose the **Action** column to edit your product.

Private offers

In the AWS Marketplace Seller Private Offer program, AWS Marketplace sellers can negotiate custom pricing and EULAs with individual AWS Marketplace customers (buyers). For more information, see [Private offers \(p. 44\)](#).

Refunds

All paid products, regardless of pricing model, must have a stated refund policy for software charges. The policy must include the terms of the refund as well as a method of contacting the seller to request a refund. Although the details of the refund policy are up to you, we encourage you to offer customers some manner of refund for usage of the product. You must comply with your posted refund policies.

Refund request types

Customers can request different types of refunds for AWS Marketplace products. If a customer requests a software refund directly from AWS, we instruct them to contact you using your posted support contact information for the product in question. Refunds of any AWS infrastructure charges are up to the discretion of AWS and are handled independently of software refunds.

If you use the AWS Marketplace Tax Calculation Service, customers might contact you to request a tax-only refund. If a customer requests a tax-only refund, you can, at your discretion, grant either a tax-only refund or a full software refund plus tax.

Refund policy and approvals

The following list outlines the AWS Marketplace refund policy and whether your approval is needed:

- **Free trials**

If you list your software as a free trial product, AWS can issue refunds on your behalf for software charges accruing within seven days of a conversion from a free trial to a paid subscription. Refunds issued in connection with free trial conversions require no action on your part. By enabling a free trial on a product, you are agreeing to this policy.

- **Private offers**

All refunds for private offers must be authorized by you before AWS can process them.

- **Software metering refunds**

If you meter the usage of your software by using the AWS Marketplace Metering Service, AWS can issue refunds on your behalf for software charges resulting from software metering errors. If these

errors are common across multiple customers, AWS reserves the right to determine an appropriate refund for each customer and apply it directly to each customer. Refunds issued in connection with the AWS Marketplace Metering Service must be confirmed with the seller one time, but does not require the seller to confirm each individual refund. By using the AWS Marketplace Metering Service with a product, you are agreeing to this policy.

- **Subscription cancellation within 48 hours of purchase**

If a buyer cancels their subscription within 48 hours of a non-private offer purchase, AWS will issue a full refund (cancel with 100 percent refund). Refunds issued in connection with cancellation within 48 hours of purchase require no action on your part. After 48 hours, such buyer request is at your discretion. By listing your product on AWS Marketplace, you are agreeing to this policy.

- **Subscription upgrade**

If a buyer replaces an existing non-private offer subscription with a more expensive subscription or a subscription of equal value, AWS can issue refunds on your behalf for the lower-tier subscription. This is a two-step process for the buyer: Buy a new subscription and then request cancellation of the old subscription with a refund.

- **Subscription downgrade**

All downgrade subscription refund requests must be authorized by you before AWS can process them.

All AWS authorized refunds are processed automatically and require no action on your part.

Refund process

You can initiate refunds for your product software usage by submitting a [Refund Request Form](#). Once received by the AWS Marketplace Buyer Support Team, a related support case will be created in the [AWS Support Center Console](#), with the refund status noted in the subject line. Refund-related support is facilitated directly through these cases. For more information, see [Accessing AWS Support](#).

The following procedure outlines how to request a refund for an external customer or an internal testing account.

To initiate a software refund for a customer

1. Gather the following information from the customer:
 - The customer's email address that is associated with their AWS account.
 - The customer's AWS account number of the account used to subscribe to your product. Remind your customer that if they are the payer of an organization, they need to provide you with the AWS account ID for the linked account subscribed to your product.
 - The billing periods for which the customer would like a refund.
2. Sign in to your AWS account and then navigate to the [Refund Request Form](#).
3. Enter the customer's information in the form.
4. Enter the Product ID for the product that your customer is requesting a refund for. You can find the Product ID in your [daily customer subscriber report \(p. 281\)](#).
5. For annual products where a customer is requesting a refund, upgrade, or downgrade, you must perform the following tasks:
 - a. Verify the customer has purchased an annual subscription using your daily customer subscriber report (there might be a 24-hour delay).
 - b. Provide a **Subscription Cancellation Date** in the comments field.
 - c. Provide a description of the change that you're authorizing (refund, upgrade, or downgrade) in the comments field.

6. Submit the form. We'll be notified and will begin to process the refund and issue it to the customer.
7. An outbound case will be created in the [AWS Support Center Console](#) with status information on the refund request. The subject line will contain one of the following:
 - **Completed** – The refund was processed and no further action is required.
 - **Pending** – The refund will be processed once the current billing cycle ends.
 - **Action Required** – The request could not be processed, and we need additional information from you. You can respond directly to the support case; however, you will also need to submit a new refund request form.
8. Once a refund is successfully processed, it will reflect on the customer's account within 24–48 hours. However, it can take up to five business days for the funds to appear in the customer's financial account.

Regions and countries for your AWS Marketplace product

When you create a product in AWS Marketplace, you choose the AWS Regions where it is available. You also choose the countries where buyers can purchase your product from. These two properties are similar, but they are not the same. For example, a buyer might be located in, and purchasing from, the United States but is installing your product in the Europe (Frankfurt) Region. In order for this buyer to purchase your product, you must include both the United States in your list of countries, and the Europe (Frankfurt) Region in your list of Regions.

AWS Regions

When creating or editing server or machine learning product information, you can limit your product to specific AWS Regions where your users can install and use the product.

For server products, including Amazon Machine Image (AMI)-, container-, and AWS CloudFormation-based products, you can select specific Regions where the product is available. You can also choose to automatically make your product available in new US Regions, non-US Regions, or all Regions as they become available.

For machine learning products, you can either select specific Regions, or all Regions including future Regions as they become available.

For more information about AWS Regions, see [AWS service endpoints](#) in the AWS General Reference.

Countries

By default, your product is available to buyers in all countries where AWS Marketplace is available. For new and existing server and software as a service (SaaS) products, you can control product availability in specific countries for tax, compliance, support, or marketing purposes.

There are exceptions to this functionality:

- **Previous purchases** – After updating your product with a new list of countries, buyers that have already subscribed to your product will still have access while their subscription is active.
- **Private offers** – When you limit your product to buyers in specific countries, it does not limit private offers. When you create a private offer to a specific buyer, it is available to that buyer, even if they are in a country that you did not include in your specified countries.

Note

Customer eligibility is determined at an AWS linked account level. For more information, see [How does AWS determine the Location of your account?](#) Customers that share their entitlement can only activate the entitlement in a region you have allowed. For more information about managing entitlements, see [Sharing subscriptions in an organization](#) in the *AWS Marketplace Buyer Guide*.

Private offers

Private offers are a purchasing program that allows sellers and buyers to negotiate custom prices and end user licensing agreement (EULA) terms for software purchases in AWS Marketplace.

Tip

You can negotiate EULA terms for each private offer, or you can use or amend [standardized license terms \(p. 54\)](#) to simplify the procurement process.

How private offers work

You can create and manage all of your private offers from the **Offers** page in the [AWS Marketplace Management Portal](#). You specify the product that the offer is being made for and the AWS account ID (or IDs) for the buyer you're creating the offer for. AWS Marketplace Management Portal generates a unique ID and URL for the offer. For instructions on creating private offers, see [Consulting partner creates](#).

When you create a private offer, you can extend the offer to up to 25 accounts. The offer is visible only to the accounts that you create the offer for. Buyers can't view the offer unless you extend the offer to either their linked account or to their management account. You can't force service limits in the offer, so the buyer can use as much of your product at the negotiated prices as they want, unless the product already has a limit.

AWS Marketplace buyers can access third-party financing for private offers. For more information, see [Customer financing is now available in AWS Marketplace](#).

Note

The buyer isn't notified that you created a private offer. You can provide the URL for the custom offer to the buyer, or they can navigate to your product through AWS Marketplace.

When the buyer navigates to your product's subscription page, a banner indicates that a private offer is available. After the buyer accepts the offer, they're invoiced for the purchase using the same portal tools used for all AWS Marketplace transactions. Accepted offers become *agreements*, and are also referred to as *contracts* or *subscriptions*.

For software as a service (SaaS) contracts and SaaS contracts with consumption products, you can offer upgrades and renewals on agreements that were made when buyers accepted private offers. For example, you can do this to grant new entitlements, offer pricing discounts, adjust payment schedules, or change the end user license agreement (EULA) to use standardized license terms. For more information, see [Private offer upgrades and renewals \(p. 52\)](#).

Private offers are tracked in seller reports. For more information, see [Reporting for private offers \(p. 47\)](#) and the [Seller reports guide](#).

Private offer experience for buyer

After you create a private offer and notify the potential buyer, they will have steps they must perform to accept the offer. For more information about the buyer experience for private offers, see [Private offers](#) in the *AWS Marketplace Buyer Guide*.

To receive the terms of the offer, the buyer must accept the offer before the offer expiration date. After the offer expires, the terms are no longer valid. You must re-create the private offer for the buyer to accept the terms. As the seller, you can provide a URL to the fulfillment page for the offer, or the customer can navigate to your product page on AWS Marketplace and choose the link in the banner to view the private offer.

Private offers through consulting partners

If you are a consulting partner, you can negotiate special terms with an ISV to offer their products to buyers. With this type of offer, you are listed as seller-of-record.

For more information, see [Consulting partner private offers \(p. 49\)](#).

Notes about private offers

When working with private offers, keep the following in mind:

- You can't create private offers for second party, Amazon Machine Image (AMI) monthly, or multi-AMI-based delivery using AWS CloudFormation products, or for limiting customer usage.
- For private offers with the flexible payment scheduler, it is possible to break upfront payments into multiple payments over time if buyers are on invoicing terms with AWS.
- If the buyer account for your private offer is managed through a private marketplace, you must include both the buyer's account and the account that includes their private marketplace administrator in the offer.
- Private offers don't support the Bring Your Own License model (BYOL) or BYOL product types.

Supported product types

AMI, container, professional services, and SaaS products are supported for private offers.

Private offers for AMI products

You can provide private offers pricing for AMI products.

The offer can be any custom duration for the following:

- AMI hourly or AMI annual private offers: up to 3 years (1,095 days)
- AMI contract private offers: up to 5 years (60 months)

For AMI contracts, private offers don't monitor usage.

Buyers can manually upgrade to new contract levels at any time. However, it is up to the independent software vendor (ISV) to define contract tiers, enforce service limitations, and advise buyers to manually upgrade to higher contract tiers when needed. The contract duration of the private offer can match the public product listing, or can be a custom duration in months (up to 60).

License entitlements begin on the date the buyer accepts the private offer.

For AMI private offers with flexible payment schedules, you can set the number of annual instance types agreed to in the contract, for the duration of the contract.

Note

Private offers are not available for monthly billing contracts.

Private offers for container products

You can provide private offers pricing for container-based product contracts.

The offer can be any custom duration for the following:

- Container hourly or AMI annual private offers – Up to 3 years (1,095 days)
- Container contract private offers – Up to 5 years (60 months)

For Container contracts, private offers don't monitor usage.

Buyers can manually upgrade to new contract levels at any time. However, the independent software vendor (ISV) defines the contract tiers, enforces service limitations, and advises buyers to manually upgrade to higher contract tiers when needed. The contract duration of the private offer can match the public product listing, or it can be a custom duration in months (up to 60 months).

License entitlements begin on the date the buyer accepts the private offer. For container private offers with flexible payment schedules, you can set the number of units agreed to in the contract, for the duration of the contract. You can also define a custom hourly price for those same units if the buyer uses more.

Note

Private offers are not available for monthly billing contracts.

Private offers for professional services products

All professional services product offerings are done through private offers. For more information, see [Creating private offers \(p. 253\)](#).

Private offers for SaaS products

Software as a service (SaaS) private offer products can't change the pricing level for a given pricing tier based on timing. For example, an offer can't charge \$0.80/hour for three months and then change pricing to \$0.60/hour thereafter for the same pricing tier. For SaaS contracts, private offers don't monitor usage.

Buyers can manually upgrade to new contract levels at any time. However, the independent software vendor (ISV) defines contract tiers, enforces service limitations, and advises buyers to manually upgrade to higher contract tiers when needed. The contract duration of the private offer can match the public product listing, or it can be a custom duration in months (up to 60 months).

Offer submission process

You can create simple private offers using the AWS Marketplace Management Portal by using the following procedure.

To create a private offer

1. Sign in to the [AWS Marketplace Management Portal](#) and choose **Offers**.
2. On the **Manage Private Offer** page, choose **CREATE AN OFFER**.
3. On the **Create Private Offer** page, select the product from the dropdown list and enter the AWS account ID (or IDs) of the AWS Marketplace buyer. If your buyer is paying for the product in installments, select **Allow buyers to pay for this product in installments**. Verify the information that you entered, and then choose **NEXT**.

Note

Selecting **Allow buyers to pay for this product in installments (ISV only)** enables you to offer your buyer a payment schedule with annual payments that aren't evenly distributed, multiple payments for a multi-year deal, or quarterly payments. Buyers must be on invoicing terms with AWS to receive a flexible payment schedule on their private offer. For more information, see [the section called "Flexible payment scheduler" \(p. 48\)](#).

4. On the **Create an Offer** page, verify the product name and buyer ID.
5. Specify the **Contract duration**:
 - a. If the product offer is for an AMI hourly or AMI annual pricing model, choose a **Duration** option or enter a custom duration in number of days.

Note

The duration of the private offer can be up to 1,095 days for the AMI hourly or the AMI annual pricing model.

- b. If the product offer is for a SaaS contract pricing model, AMI contract pricing model, or container contract pricing model, choose a **Duration** option or enter a custom duration in number of months.

Note

The duration of the private offer can be up to 60 months for the SaaS contract pricing model, AMI contract pricing model, or container contract pricing model.

6. In **Input offer price**, enter the pricing information that you negotiated with the customer. If you have installment payments for the private offer, specify the number of units and the payment schedule for the contract duration. For more information about installment payments, see [the section called "Flexible payment scheduler" \(p. 48\)](#).
7. In **Upload End User License Agreement**, select from available options or upload your EULA .pdf file.
8. In **Offer Expiration and Acceptance Date**, enter the number of days that the offer is valid for.

Note

This is the number of days after the customer accepts the offer that the terms of the agreement are active. After the number of days has lapsed, the price and EULA revert to the terms provided in the public offering.

9. For **Buyer needs to accept the offer by**, enter the date when the offer is no longer available if not accepted.

Note

This is the date that the offer becomes null and void. On that date, the buyer won't be able to accept the offer under the custom terms that you have specified.

10. Choose **REVIEW OFFER**.
11. On the **Review Offer** page, verify the offer information and the .pdf file, and then do one of the following:
 - If the offer is correct, choose **EXTEND OFFER**.
 - If the offer is incorrect, choose **EDIT OFFER** and make any required changes.

The offer should appear on the **Manage Private Offer** page in approximately 45 minutes. To view the offer, sign in to the AWS Marketplace Management Portal and choose **Private Offer**. This opens the **Manage Private Offer** landing page.

Reporting for private offers

Private offers appear on the existing seller reports and in the reports relevant to the offer. The [Monthly billed revenue report \(p. 288\)](#) is generated every month and has offer visibility and offer ID information. When an invoice is generated for a buyer, it appears in the report covering the appropriate billing period. For more information, see the [Seller reports guide](#).

The **Offer ID** field contains the unique offer ID generated for the private offer. It's blank unless the report entry is for a private offer. The **Offer Visibility** field indicates whether the report entry is a public or private offer. For all private offers, the entry is marked private.

Flexible payment scheduler

Flexible payment scheduler enables you to extend private offers with a custom payment schedule. The schedule can be spread over up to three years, and the customer makes payments in regular installments. After they are subscribed, your customers can see all the payments on the schedule and on their AWS invoice, helping them track their spending. Flexible payment scheduler is available for private offers on AMI multi-year and SaaS contracts products.

Any customer on invoice terms, for example net-30 or net-60 terms, can subscribe to a private offer with a flexible payment schedule. Customers who pay their AWS bill using a credit card can't. If you try to create a private offer with a custom payment schedule for a customer who isn't on invoice terms, you receive an error.

Creating a payment schedule

The process for creating a custom payment schedule using flexible payment scheduler is part of the process for creating a private offer. While creating the private offer, as you are adding product and buyer account information, choose **Allow Buyers to pay for this product in installments**. This enables you to create an offer with a flexible payment schedule. When you choose **Next** to continue, the flexible payment scheduler feature validates that any AWS account that you added is an account on invoice terms. If you have provided an account that isn't on invoice terms, you receive an error message.

Note

If the account is in an AWS Organizations billing family, the targeted account can be any account that is on net payment terms with AWS. For more information, see [Consolidated Billing for AWS Organizations](#) in the *AWS Billing and Cost Management User Guide*.

After the AWS account or accounts are confirmed, customize your offer details on the next page. Choose the contract duration for this offer and specify the offer details accordingly.

Note

For private offers with flexible payment scheduler, for multi-year and custom duration Amazon Machine Image (AMI) products, set the number of instances for each instance type included in the offer and the hourly pricing for any additional launched instances. After the customer launches the specified number of instances, any additional instances launched are charged at the hourly rate specified in the private offer.

Under **Payment Schedule**, add the invoice dates and invoice amounts for all of the installments that the customer will make. You can add up to 60 installments. Each time you add an installment, **Total amount due from buyer** is updated.

Note

The invoice date for the first installment is the first time that the customer is invoiced for your private offer. You receive the payment for that first invoice after AWS Marketplace receives the payment from the customer.

The flexible payment scheduler feature validates that the invoice dates fall within the contract duration. If your last invoice date is after the duration of the contract, you receive an error message.

After you have added all invoice dates and amounts, confirm that **Total amount due from buyer** matches the total price that you want your customer to pay over the course of the private offer. To finish creating the private offer, upload the end user license agreement (EULA) for the customer and set the offer acceptance date.

Note

Only one invoice date can occur before the offer acceptance date that you're extending to your customer.

Your customer is invoiced based on the schedule that you defined, and invoices start after they accept the offer. If the first invoice date is scheduled before the offer is accepted, this invoice is processed immediately after the offer is accepted.

Note

You can't modify the payment schedule on a private offer that has been extended to and subscribed by a buyer. To make changes, you must create a new offer.

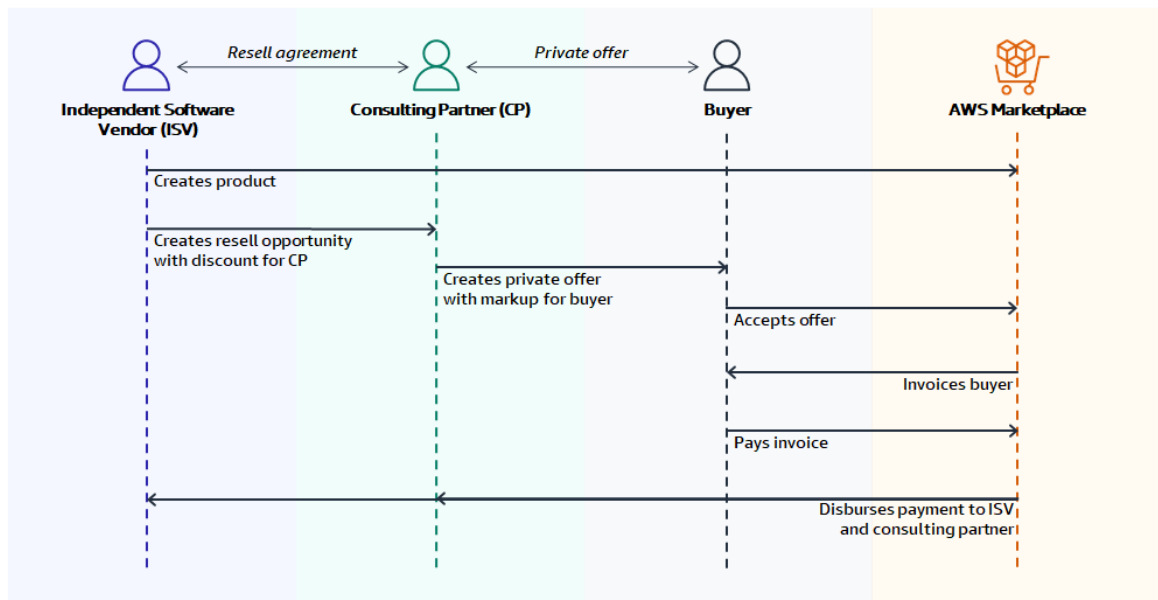
Reporting for flexible payment scheduler

Reporting for private offers with flexible payment schedules is in the [Section 4: Contracts with flexible payment schedule \(p. 293\)](#), of the monthly billed revenue report.

Consulting partner private offers

AWS Marketplace consulting partner private offers allow consulting partners to resell independent software vendors' (ISVs) products on AWS Marketplace. The consulting partner and ISV establish an agreement to resell one or more of the ISV's products, and then they extend a private offer to the buyer for that product.

The following diagram shows this relationship between an ISV, a consulting partner, and a buyer.



Note

For more information on creating a resell opportunity for a consulting partner, as an ISV, see [ISV setup of resell opportunities \(p. 50\)](#).

For more information on creating a private offer for a buyer, as a consulting partner, see [Consulting partner setup of resell opportunities \(p. 51\)](#).

Each consulting partner private offer is visible only to a single buyer, with customized pricing and unique commercial terms to meet that buyer's needs. When creating a private offer, you start from a wholesale cost set by the ISV. Then you mark up that price to create the buyer's offer price. The wholesale cost is determined in one of two ways:

- **Recurring discount** – An ISV authorizes the consulting partner to resell their product or products at an agreed-to discount from their list price with a recurring opportunity. This discount allows the consulting partner to continue to resell the product without further price negotiation with the ISV. This

discount can be set up to last until a specified date, or indefinitely, until ended by either the ISV or the consulting partner.

- **Non-recurring discount** – The opportunity that the ISV gives the consulting partner is a one-time discount intended to be used only with a specific buyer.

In both cases, after the buyer pays for the private offer, AWS Marketplace uses the standard process to distribute the funds to the consulting partner and the ISV based on the agreed-to pricing.

Tip

As an ISV or a consulting partner, you can view opportunities that you have granted or received from the **Partners** menu of the [AWS Marketplace Management Portal](#).

For detailed instructions about creating private offers, see [Consulting partner creates](#).

For information about third-party financing for private offers, see [Customer financing is now available in AWS Marketplace](#).

Additional information

For additional information and questions, we encourage ISVs and consulting partners to connect with the AWS Marketplace channel team. If you don't know who to contact specifically, send an email message to <aws-mp-channel@amazon.com>, and someone on the team will respond to you within one business day.

ISV setup of resell opportunities

As an ISV, you can authorize consulting partners to resell your products by creating a resale *opportunity* for that partner. You can specify a discount percentage or custom price per product dimension to create a wholesale price for the consulting partner. The partner can mark up the wholesale price when creating their consulting partner private offer for a buyer. For more information about consulting partner private offers, see [Extending a private offer based on an opportunity \(p. 51\)](#).

Note

If the particular terms of the authorization that you want to create are not possible using the AWS Marketplace Management Portal, you can fill out an *AWS Marketplace Reseller Author form*. To request and return the form, reach out to your AWS Marketplace channel account manager or send an email message to mpcustdesk@amazon.com.

The following procedure outlines how ISVs can create an opportunity for a consulting partner. To use this feature, you must have permissions to use the **Partners** tab in the AWS Marketplace Management Portal. For more information, see [Policies for AWS Marketplace sellers \(p. 346\)](#).

To create an opportunity for a consulting partner

1. Sign in to the [AWS Marketplace Management Portal](#).
2. Choose the **Partners** tab, and then choose **Create opportunity**.
3. Enter the **Opportunity name** and **Opportunity description**.

Note

The information you enter in **Opportunity name** and **Opportunity description** will be visible to consulting partners in their seller reports.

4. Choose the **Resellers** that you want to authorize. You can select resellers by name or account ID.

Note

If a reseller doesn't appear in the list, they may need to register first. Only registered resellers can be authorized for an opportunity. For more information, see [Consulting partner setup of resell opportunities \(p. 51\)](#).

5. Select which of your **Products** are part of this opportunity and the **Discount** that you want to apply.

6. Select the **Duration** of the opportunity.
7. (Optional) For SaaS contract products, add or remove custom **Product dimensions** and modify the **Additional usage fees** to customize your opportunity.
8. (Optional) Set one or more **Buyer account IDs** to specify that the opportunity is only for those buyers.
9. (Optional) Select the **End User License Agreement (EULA)** version or upload the EULA to be included in the opportunity.
10. Select **Review opportunity**, and make sure that the information is correct.
11. Select **Create opportunity** to finalize the opportunity and authorize the consulting partners.

Once created, opportunities can't have their dates extended. However, you can revoke an opportunity and recreate it at any time. When you revoke an opportunity, new offers can't make use of that discount. Any existing offers are unaffected and retain their opportunity discount.

Consulting partner setup of resell opportunities

To create a consulting partner private offer, you must be registered through the [AWS Marketplace Management Portal](#) as an AWS Marketplace seller. The following topics can help you get started as an AWS Marketplace seller.

- [Getting started as a seller \(p. 4\)](#)
- [Preparing your product \(p. 35\)](#)
- [Submitting your product for publication \(p. 261\)](#)
- [Seller reports and data feeds \(p. 273\)](#)

If you're new to providing products on AWS Marketplace, the following topics can help you better understand the kinds of products available:

- [AMI-based products \(p. 63\)](#)
- [Software as a service \(SaaS\)-based products \(p. 214\)](#)

Putting an agreement in place with an ISV

Before you as a consulting partner can create a private offer for a product, the ISV must authorize you to resell their product. The ISV does this by creating an *opportunity* for you. For more information, see [ISV setup of resell opportunities \(p. 50\)](#). To create an opportunity, the ISV must provide:

- The product or products that they authorize you to resell.
- The price reduction that they want to offer you.
- The AWS account ID that you used to register as an AWS Marketplace seller.

After the opportunity has been created, you will be an authorized reseller for that product. Then, you can extend private offers that are marked up from the price given you by the ISV.

Extending a private offer based on an opportunity

For recurring discount private offers, an ISV authorizes a consulting partner to resell one or more of their products on AWS Marketplace. The discount, called the wholesale price, is an agreed-to price or percentage discount off the product's list price. Consulting partners can use the discount with any number of buyers.

The following procedure outlines how a consulting partner can extend a private offer based on a recurring discount.

To extend a private offer based on a recurring discount

1. Determine what your offer price will be by marking up the wholesale price by a percentage.
2. Sign into the [AWS Marketplace Management Portal](#).
3. Choose **Partner** from the menu at the top of the screen.
4. From the **Opportunities you received** list, choose the opportunity with the product and discount you want to offer to the buyer, and then select **Create Offer**.
5. Step through the **Create Private Offer** screens, entering required information, including the buyer's account, the pricing details of the offer (including the marked-up price by a percentage), the End User License Agreement (EULA), and the dates for the offer. Then, choose **Review Offer**.
6. Review the offer, and then choose **Extend Offer**.

The publishing process for this offer can take up to 45 minutes to complete. After it's completed, the offer is visible on the **Manage Offers** page.

Accepted offers

After the buyer accepts the private offer from the consulting partner, the offer and any disbursement of funds occur in the same manner:

1. AWS Marketplace invoices the buyer on their existing AWS bill per the terms of the private offer. If the private offer is extended to a linked account, the invoiced amount appears on the payer account associated with that linked account.
2. The buyer pays their AWS bill in accordance with the net payment terms that they agreed to with AWS. The private offer process enables custom terms for each transaction, but net payment terms aren't customizable.
3. After AWS receives payment from the buyer, AWS disburses payment to you and the ISV. The ISV receives the wholesale cost minus the AWS Marketplace fee. You receive your markup minus the AWS Marketplace processing fee. All fees are percentages applied to the transaction amounts listed. If you're not sure of the fee percentages and need this information for quoting purposes, contact your AWS Marketplace channel account manager. If you don't know who that is, send an email message to the AWS Marketplace channel team at <aws-mp-channel@amazon.com>, and someone on the team will respond to you within 24 hours.
4. AWS Marketplace provides electronic reports to the ISV and to you using the [AWS Marketplace Management Portal](#). These reports have the following differences depending on the type of private offer:
 - For recurring discount private offers, the ISV sees you as the buyer and you see the subscriber as the buyer.
 - For non-recurring discount private offers, the ISV and the consulting partner see the subscriber as the buyer.

For more information about AWS Marketplace reporting, see [Seller Reporting](#).

Private offer upgrades and renewals

For software as a service (SaaS) contract and SaaS contract with consumption products, you can offer upgrades and renewals by using a private offer on any active agreements. For example, you can do this to grant new entitlements, offer pricing discounts, adjust payment schedules, or change the end user license agreement (EULA) to use [standardized license terms \(p. 54\)](#). You can also change the number of units and payment schedule, and add a custom end date.

The difference between an *offer* and an *agreement* is whether the buyer accepted its terms:

- An **offer** is a set of terms for a buyer's use of a product. Offers can be public or private.
- An **agreement** is an offer that a buyer accepted. Agreements include purchased and free products that a seller made available via a public or private offer.

This feature is available to all AWS Marketplace sellers, including independent software vendors (ISVs) and consulting partners. You can't amend an agreement to specify a seller of record that's different from the seller of record from the original agreement.

To use this feature, you must have permissions to use the **Agreements** tab in the AWS Marketplace Management Portal. For information, see [Permissions for AWS Marketplace sellers \(p. 346\)](#).

Supported product types

The following product types support private offer renewals and upgrades:

- SaaS contracts
- SaaS contracts with consumption

Submission process for upgrades and renewals

You can create private offer upgrades and renewals from the AWS Marketplace Management Portal by using the following procedure.

To create private offer upgrades and renewals

1. Sign in to the [AWS Marketplace Management Portal](#) and choose **Agreements**.
2. On the **Agreements** page, create an upgrade or renewal private offer in one of the following ways:
 - Choose a check box next to an agreement, and then choose **Create agreement-based offer**.
 - Choose an agreement ID to view the agreement details. On the **Agreement summary** page, review the agreement's existing information and terms to verify that this is the agreement you want to amend, and then choose **Create agreement-based offer**.
3. On the **Agreement offer details** page, enter a custom offer name.

Tip

Entering descriptive custom offer names can help you distinguish between your active offers on the **Offers** page. Custom offer names are also visible to buyers. AWS recommends that you specify a custom offer name that includes any additional identifying details, such as your own IDs and purchase order numbers. Using high-level descriptions like **upgrade** or **renewal** and custom company names are also recommended. Don't use any personally identifiable data (for example, first or last names, phone numbers, or addresses). You can enter up to 150 characters for this field.

4. Edit the information for any dates, dimensions, payment schedule, and EULA that you want to change. Then choose **Next**.
5. On the **Review and create** page, review the information. When ready, choose **Create agreement-based offer**.

The new private offer appears on the **Manage Private Offer** page in approximately 45 minutes. To view the offer, sign in to the AWS Marketplace Management Portal and choose **Offers** to open the **Manage Private Offer** page.

Similar to the process for creating a private offer, the buyer isn't notified that you created a new private offer. Instead, you provide the URL for the new private offer to the buyer. From there, the buyer has the option to accept it or to continue to operate under the original agreement:

- If the buyer accepts the private offer upgrade or renewal, the new agreement takes effect immediately and the agreement is listed on the **Agreements** page in the AWS Marketplace Management Portal. Any remaining scheduled payments from previous agreements are cancelled.

Buyers accept agreement-based private offers the same way they accept private offers. For more information about the buyer experience for private offers, see [Private offers](#) in the *AWS Marketplace Buyer Guide*.

- If the buyer doesn't accept the private offer upgrade or renewal before it expires, the original agreement remains in effect with no changes.

Reporting for upgrades and renewals

Upgrade and renewal private offers appear on the existing seller reports and in the reports relevant to the offer. The [Daily customer subscriber report \(p. 281\)](#) report and [Daily business report \(p. 274\)](#) report are generated daily. The [Monthly billed revenue report \(p. 288\)](#) report is generated monthly.

In the Daily customer subscriber report, the **Subscription intent** field indicates whether the report entry is a new private offer. The **Previous offer ID** field indicates the ID of the offer that preceded the new offer, if one exists. For all private offers, the entry is marked "private".

Agreements data is not shown in data feeds.

Standardized license terms

As you go through the process of [preparing your product \(p. 35\)](#), you need to determine what to include in the end user license agreement (EULA) for your product. You can create and customize your own EULAs, or you can use the standardized license terms that AWS Marketplace offers, which help speed up transactions and simplify procurement.

AWS Marketplace offers the following options for providing standardized license terms for your product listings:

- [Standard Contract for AWS Marketplace \(SCMP\) \(p. 54\)](#) – These license terms are intended to meet the fundamental requirements of buyer and seller.

This option is available to all AWS Marketplace sellers and is accessible to all buyers.

- [Enterprise Contract for AWS Marketplace \(ECMP\) \(p. 55\)](#) – These license terms are intended to meet the more stringent requirements of enterprise buyers.

This option is available for all AWS Marketplace sellers to enroll in, and it's accessible only to enrolled enterprise buyers. When you enroll as a seller in the ECMP program, you can still offer SCMP to non-enterprise buyers.

Disclaimer

You are responsible for determining whether these documents meet your specific requirements. These documents should not be construed as legal advice for any particular facts or circumstances.

Standard Contract for AWS Marketplace (SCMP)

SCMP is a standardized set of license terms that govern usage and define obligations of buyers and sellers. AWS Marketplace sellers can offer SCMP as the EULA for public product listings. Buyers can search for, buy, and quickly deploy software from sellers that offer the terms of the standard contract.

For [private offers \(p. 44\)](#), you can amend the SCMP to address custom transaction requirements as agreed upon by both parties.

Getting started with SCMP

This section describes how to review terms and offer SCMP to buyers.

To use SCMP for new and existing AWS Marketplace listings

1. Review the terms of the [Standard Contract for AWS Marketplace](#).
2. Sign in to the [AWS Marketplace Management Portal](#).
3. As you create a product or edit an existing listing, choose **SCMP EULA** as the EULA.

For more information about creating products, see [Submitting your product for publication \(p. 261\)](#).

To request assistance in updating the EULA to the SCMP

1. From the lower-left corner of the AWS Marketplace Management Portal, choose [Contact us](#).
2. Enter your email address, and then complete the rest of the form as follows:
 - For the subject of your question, choose **Commercial Marketplace**.
 - For the category, choose **Product Listing**.
 - For the subcategory, choose **Standard Contract Request**.
 - In the text box for providing request details, type `Enable SCMP for AWS Marketplace product listings`.

If you have questions about the SCMP program, contact `<aws-mp-standardcontract@amazon.com>`.

Enterprise Contract for AWS Marketplace (ECMP)

Like SCMP, ECMP is standardized set of license terms that govern usage and define obligations of buyers and sellers. For [private offers \(p. 44\)](#), you can amend the ECMP to address custom transaction requirements as agreed upon by both parties.

The differences between ECMP and SCMP are as follows:

- ECMP is designed to address the more stringent requirements of large enterprises.
- To offer the ECMP on your listings, you must enroll in the program.
- You can offer ECMP as the EULA on public product listings and private offers *only* to enrolled enterprise buyers.

After enrolling in the ECMP program, you can still offer SCMP as your public EULA to address license requirements of non-enterprise customers.

Getting started with ECMP

This section describes how to review terms and enroll in the ECMP program.

To enable ECMP for AWS Marketplace listings

1. Review the terms of the [Enterprise Contract for AWS Marketplace](#).
2. Complete the [enrollment form](#) and choose **Register Now**.

If you have questions about enrollment in the ECMP program, contact aws-mp-enterprisecontract@amazon.com.

Categories and metadata

Here are best practices and information for supplying product metadata. AWS Marketplace revises product metadata solely for quality assurance and error correction.

Naming and describing your product

The information that you provide about your product is visible to buyers. Ensure that potential buyers have enough information to make informed decisions about buying your product.

Creating the product name

Keep the following guidelines in mind as you create the product name:

- Use title case (capitalize the first letter of each important word)
- Ensure that a buyer can identify the product by the name alone
- Use the name of the brand or manufacturer
- Avoid descriptive data or hyperbole

Example product name: Smart Solution Load Balancer - Premium Edition.

Writing the product description

The product description lists the product's features, benefits, and usage. It can also provide other relevant, specific product information. The description can be up to 350 characters long.

Keep the following guidelines in mind as you write the product description:

- Avoid unnecessary capitalization
- Avoid unnecessary punctuation marks
- Don't include redirect information
- Check spelling and grammar
- Include only critical, useful information

Example product solution: Smart Solution automatically distributes incoming application traffic across multiple Amazon EC2 instances. It enables you to achieve even greater fault tolerance in your applications, providing the amount of load-balancing capacity you need to respond to incoming application traffic. Smart Solution detects unhealthy instances in a pool and automatically reroutes traffic to healthy instances until the unhealthy instances are restored. You can enable Smart Solution in a single AWS Availability Zone or across multiple Availability Zones to ensure more consistent application performance.

Writing the product highlights

The product information page displays up to three product highlight bullet points. Use these bullet points to briefly describe the product's primary selling points.

Example product highlight: Projecting costs: With Smart Solution, you pay only for what you use. You're charged for each hour or partial hour that Smart Solution is running.

Writing the release notes

Each time you update an AMI product, you must provide a description of the changes in the release notes. The release notes should contain specific information to help the user decide whether to install the update. Use clear labels for the update, such as "Critical" for a security update or "Important" or "Optional" for other types of updates.

Writing the usage instructions

Provide usage instructions that help ensure that the buyer can successfully configure and run the software. The usage instructions you provide are shown during the AMI configuration process.

To write effective usage instructions, follow these guidelines:

- Write them with a new or moderately technical audience.
- Don't assume that the user has prior experience with or extensive knowledge of the product, computer operating systems, engineering, or IT operations.
- Take the buyer from launching to using the product, including any configuration or special steps to get the application running.

Example usage instructions:

1. Launch the product via 1-Click.
2. Use a web browser to access the application at https://<EC2_Instance_Public_DNS>/index.html.
3. Sign in using the following credentials:
 - Username: user
 - Password: the instance_id of the instance

Writing the upgrade instructions

Provide details on how buyer can upgrade from an earlier version of the product. Include information on how to preserve data and settings when creating another instance. If there is no upgrade path, edit this field to specifically mention that.

Example upgrade instructions:

1. Do ***, and then ***.
2. Check that all plugins used by your project are compatible with version *.* , by doing ***. If they aren't compatible, do ***.
3. Make a backup of your data, by doing ***.

Choosing categories and keywords

When you list your product, you can choose up to three software categories and corresponding subcategories for your product. This helps buyers discover your product as they browse or search for products on AWS Marketplace. Choose only categories that are relevant to your product; in most cases, only one category applies. The product load form and the **Products** tab both contain a complete list of categories.

Categories aren't the same as keywords. The categories and subcategories available are predefined for AWS Marketplace, and you decide which ones apply to your product by selecting them from a list during

the product request process. Keywords aren't predefined, but are created during the process. You don't need to add the category as a keyword.

Creating search keywords

During the product request process, you can enter up to three keywords (single words or phrases) to help buyers discover your product through site searches. The keywords field can contain a maximum of 250 characters.

The following tips can help you to create a relevant set of search keywords:

- Use relevant terms.
- Don't use the names of products published by other sellers or use other sellers' names.
- Choose keywords from your buyer's vocabulary—that is, words and phrases that buyers are likely to use when thinking about your type of product.
- Create keywords based on specific features in your product.
- Don't use the product title as a keyword. The product title is already indexed in searches.

Note: Keywords aren't the same as software categories. Keywords are more specific terms that are related to your product.

Search engine optimization for products

The AWS Marketplace website ranks the results of search queries using search optimization techniques similar to those used across the industry. By understanding how AWS Marketplace ranks and returns search results, you can create product details optimized for the AWS Marketplace search engine. We recommend taking this guidance into consideration when you create your product detail pages.

Keywords

During the product creation process, you can submit up to three keywords (single words or phrases) to help customers discover your product through site searches. The keywords text box can contain up to 250 characters.

Use the following tips to create search keywords:

- Use terms that are relevant so that customers can easily find your products.
- Choose keywords from your customers' vocabulary—that is, words and phrases that they're likely to use when thinking about your type of product.
- Create keywords based on specific features in your product.
- Don't include the product title in the terms that you submit. The product title is already indexed in the search.

Note

Keywords aren't the same as software categories. Keywords are more specific terms that are related to your product.

You can edit keywords after you create a product by editing the metadata for the product. For products that you created using a Product Load Form (PLF), you also use the PLF to make changes to those products. For products that you created using the **Products** tab in AWS Marketplace Management Portal, you also use the **Products** tab to make changes. For more information, see [Product changes and updates \(p. 264\)](#).

The AWS Marketplace Managed Catalog Operations team helps redirect queries with similar-sounding words or words with similar meanings. For example, when customers search for *automobile* when you expect them to search for *car*.

Software categories

When you list your product, you can choose up to three software categories and corresponding subcategories for your product. This helps customers discover your product as they browse or search the products on AWS Marketplace. Choose only categories that are relevant to your product. In most cases, only one category applies. Both the product load form and the **Products** pages contain a complete list of categories.

Note

Categories aren't the same as keywords. The available categories and subcategories are predefined for AWS Marketplace. You decide which of them apply to your product by choosing them from a list. Keywords aren't predefined, but they are created during the process.

Highlights section

The product details page displays up to three product highlights as bullet points. Customers can search for products by highlights, so include highlights when you create a product. A highlight should describe the product's primary selling points in brief and informative language.

Example Highlights

- Projecting costs: With AnyCompany's product, you pay only for what you use. You're charged for each hour or partial hour that it's running.

Short description

The product description lists the product's features, benefits, and usage instructions, along with other relevant and specific product information. Keep the following guidelines in mind as you create the product description:

- Avoid unnecessary capitalization and punctuation marks
- Don't include redirect information
- Check spelling and grammar
- Include only critical and useful information

Example Short description

AnyCompany's product automatically distributes incoming application traffic across multiple Amazon EC2 instances. It enables you to improve fault tolerance in your applications by seamlessly providing the load balancing capacity that you need to respond to incoming application traffic. AnyCompany's product detects unhealthy instances in a pool and automatically reroutes traffic to healthy instances until the unhealthy instances have been restored. Customers can enable it in a single AWS Availability Zone or across multiple Availability Zones to enable more consistent application performance.

AWS Marketplace for Desktop Applications (AMDA)

AWS Marketplace for Desktop Applications (AMDA) is a catalog of virtualized desktop applications that run on Amazon WorkSpaces. By using AMDA, you can find and subscribe to free and paid applications across 11 software categories. Applications run in virtualized containers as if they were natively installed and buyers are charged on a per-user, per-month basis.

Buyers use the Amazon WorkSpaces Application Manager (Amazon WAM) console to deploy desktop applications to their WorkSpaces. The applications are delivered to each WorkSpace through the Amazon WAM client application.

The virtualization technology enables fast delivery of programs, often without a reboot, so that users can quickly launch and use their subscribed applications. Users are charged only for those applications they have been assigned, and charges accrue monthly from when they are first launched until the assignment is revoked. For more information, see the following resources:

- [WorkSpaces product pages](#)
- [WorkSpaces testimonials](#)
- [AMDA help pages and frequently asked questions](#)
- [AWS Marketplace for Desktop Applications catalog](#)

Starting the onboarding process

Under the terms of the AWS Marketplace for Desktop Applications Publisher Addendum (the “AMDA Addendum”), Amazon Web Services, Inc. is the seller of record for applications you choose to make available through the AMDA channel. As the seller of record, AWS needs to know the price you charge AWS for the products you plan to have on AWS Marketplace. Pricing should be on a per-month basis, per user. AWS can help you to determine the final price to buyers.

In order to have your product published in non-US AWS Regions, AWS also needs you to provide certain export classification information, including the applicable [Export Control Classification Number](#) (ECCN).

During and after the initial testing of your product, a member of the AMDA Business Development team is available to answer your questions. You will then receive an email message from <aws-mp-amda-contract@amazon.com> with the AMDA Addendum for you to fill out, sign, and return for counter signature. The AMDA Addendum is an addendum to the Terms and Conditions for AWS Marketplace Sellers. Therefore, you must establish an AWS Marketplace seller account and click through these terms prior to beginning the onboarding process to include your applications on AMDA.

Product submission and packaging

Virtualization and packaging are handled by the AWS Marketplace Managed Catalog Operations (MCO) team. AMDA vendors provide the software installer, installation instructions, and product metadata. MCO will work with you to complete the packaging and complete the process for AMDA. Currently, all AMDA software must be packaged by using an MCO administrative account with permissions to the Amazon Simple Storage Service (Amazon S3) bucket that will store the package. AWS is unable to accept shared

packages. Review the following guidelines before you submit your product. MCO will start processing your packaging request upon receipt of these items:

1. Software installer and license key:
 - a. Amazon S3 bucket or external URL for the hosted Installer file (for example, .msi or .exe)
 - b. Server license key that is compatible with Windows Server 2008 R2
2. Installation instructions:
 - a. Known issues for Windows Server 2008 R2
 - b. Silent install command line arguments
 - c. Licensing mechanism notes:
 - i. Where is the license stored?
 - ii. How is the license verified?
 - iii. Which actions trigger a license check?
 - d. Auto-update
 - i. If enabled, describe how to disable this function.
 - e. Services or Registry requirements:
 - i. List each required service or registry key, with a brief description of its purpose.
3. Test servers, data files, and additional external elements
 - a. If required for installation, provide a test environment for external components (for example, SQL Server).
 - b. If your program processes data files, include test files so we can assess performance and functionality.
4. List all program dependencies, for example:
 - a. C++ redistributables
 - b. Java, QuickTime, etc.
 - c. GPU/hardware requirements
5. Program technical contacts
 - a. Who is the point of contact for technical questions or issues encountered during testing and packaging?

Application packaging types

AMDA packaging can be completed in two ways: virtualized installation or silent installation.

Virtualized installation relies on AMDA packaging tools to monitor all file changes during the installation process. AWS will point to the installer executable and click **Install**, which monitors all file changes. AWS then makes custom changes to the registry, services, and file structure to assess program stability and performance.

Some advanced programs require a silent installation mechanism. In this case, AMDA virtualizes only the installer files so that the software is physically installed only when the application is first launched on the user's Workspace. Additional steps are required to script the removal of silent installation programs.

Building the AMDA package

The packaging process relies on creating a diff of the target installation machine, which is a Windows Server 2008 R2 virtual machine (VM). The packaging tool monitors the VM during the installation process, creates a manifest of the changed files, and rolls this into a package to be ingested.

After capturing the changes programmatically, an AWS technician inspects the files, services, and registry entries to ensure all changes were accurately captured. During this process, the technician removes all uninstall and auto-update references to ensure the application stays within the confines of the virtualized package.

Programs that rely on specific Windows services (for example, background-running Windows services and .dll requirements) might require additional testing and packaging. By default, all program properties are virtualized to run on demand. Some services might require elevation to ensure they are available to the program at runtime.

License keys are captured during the packaging process to help ensure a seamless, one-click experience for end users. If your program requires the license keys on first launch, include detailed notes about how to manually add the license to the applications files.

Application metadata

The current data load form is available at: <https://s3.amazonaws.com/aws-mp-vendor-guide/AMDAProductDataLoad.xlsx> Enter the following application metadata into the AMDAProductDataLoad.xlsx load form and include it with your application submission:

- Title – The title of the product.
- Full Description – Description appears on the product detail page.
- Short Description – Description appears on the search results page.
- End User License Agreement – The EULA that applies to the buyer's use of the product.
- Image – The product image or logo that appears on the product detail page, in search results, and elsewhere on the AMDA website. Provide a URL to a square-formatted image logo.
- Categories – The software category for the product. See the AMDA home page to view the available categories.
- Software By – The software developer that is displayed on the product page, which is usually your company name.
- Vendor URL – The link to your website or a specific page that displays more information on the product.
 - Support text/email/URL – Only one field is required, but multiple contact points are encouraged.

Ingestion and new version updates

Ingestion of each AMDA product is handled by the AWS Marketplace MCO team. The current pipeline supports releases on Thursday. AWS locks on metadata and final packaging on Tuesday at noon PST. Requests after Tuesday noon PST are eligible on the following week's publishing day. New version updates are made on the same schedule.

If no metadata updates are requested, only the installer and associated files are required.

If you are updating metadata, send an updated product data load form to the [AWS Marketplace Seller Operations](#) team.

AMI-based products

One way of delivering your products to buyers is with [Amazon Machine Images \(AMIs\)](#). An AMI provides the information required to launch an Amazon Elastic Compute Cloud (Amazon EC2) instance. You create a custom AMI for your product, and buyers can use it to create EC2 instances with your product already installed and ready to use.

When buyers use the AMI that you provide, they're billed for instances that they create, following the pricing and metering options that you create for your product. Buyers can use your product AMI in the same way that they use other AMIs in AWS, including making new custom versions of the AMI. EC2 instances created from the AMI are still billed as your product, based on the AMI product code.

See the following resources:

- For more information about pricing AWS Marketplace products, see [Product pricing \(p. 37\)](#).
- For more information about creating custom metering for your product, see [AWS Marketplace Metering Service integration \(p. 96\)](#).

AMI-based product delivery methods

You can deliver your AMI-based product in one of three ways:

- **Single AMI** – Buyers select and use the AMI as a template for an EC2 instance. Buyers can find these products using the **Amazon Machine Image** delivery method filter.

For more information, see [Single-AMI products \(p. 66\)](#).

- **AWS CloudFormation templates** – You create templates that allow buyers to install a system of multiple instances with different roles as a single unit. Buyers can find these products using the **CloudFormation** delivery method filter.

For more information, see [AMI-based delivery using AWS CloudFormation \(p. 74\)](#).

- **Private image build** – This approach allows buyers to install your product on a base gold image that meets their internal needs for operating system configuration. They create a new AMI, with your product code for tracking and billing. Buyers can find these products using the **Private Amazon Machine Image** delivery method filter.

For more information, see [Private images \(p. 84\)](#).

See the following resources:

- For more information about how your AMIs are tracked as buyers use them, see [AMI product codes \(p. 65\)](#).
- For more information about the details of AMI-based products, and their lifecycle, see [Understanding AMI-based products \(p. 63\)](#).

Understanding AMI-based products

This section outlines key concepts in working with AMI-based products.

Topics

- [Product lifecycle \(p. 64\)](#)

- [AMI product codes \(p. 65\)](#)
- [Change requests \(p. 65\)](#)
- [Product Load Forms \(p. 66\)](#)

Product lifecycle

AMI-based products include a set of one or more versions of the software, and metadata about the product as a whole. When you create the product, you configure its properties in AWS Marketplace including your product's name, description, and pricing. You also determine the appropriate categories for your product and add keywords so your product appears in relevant searches.

You also create the first version of the software. Depending on how you are delivering your software, this might be a single AMI, a set of one or more AMIs with AWS CloudFormation templates, or software packages for your buyer to use in creating their own AMIs. For more information, see [AMI-based product delivery methods \(p. 63\)](#).

For paid products, buyers are billed for the number of installed instances. To meter on a different dimension that your software tracks (for example, number of users of the product), integrate your product with the AWS Marketplace Metering Service. For more information, see [AWS Marketplace Metering Service integration \(p. 96\)](#).

When you create your product and the first version of your software, it's initially published in a limited scope so that only your account can access it. When you're ready, you can publish it to the AWS Marketplace catalog to allow buyers to subscribe and purchase your product.

The lifecycle of an AMI-based product for AWS Marketplace does not end after you publish the first version. You should keep your product up to date with new versions of your software and with security patches for the base operating system.

As an example of a complete AMI-based product lifecycle, imagine a seller wants to sell their AMI-based product on AWS Marketplace. Following is how the seller creates and maintains the product over time:

1. **Create a product** – The seller creates the product, and publishes version 1.0.0 to AWS Marketplace. Buyers can create instances of version 1.0.0 and use it.
2. **Add a new version** – Later, the seller adds a new feature to the product, and adds a new version, 1.1.0, that includes the feature. Buyers can still use the original version, 1.0.0, or they can choose the new version, 1.1.0.

Note

Unlike new products, new versions are published to full public availability. You can only test them in AWS Marketplace without customers seeing them if the product as a whole is in limited release.

3. **Update product information** – With version 1.1.0 available, the seller lets buyers know about the new feature by updating the product information with new highlight text describing the feature.
4. **Add a minor version** – When the seller fixes a bug in version 1.1.0, they release it by adding a new version 1.1.1. Buyers now have the choice of using version 1.0.0, 1.1.0, or 1.1.1.
5. **Restrict a version** – The seller decides that the bug is serious enough that they don't want buyers to be able to use version 1.1.0, so they restrict that version. No new customers can then buy 1.1.0 (they can only choose 1.0.0 or 1.1.1), although existing buyers still have access to it.
6. **Update version information** – To help those existing buyers, the seller updates the version information for 1.1.0 with a suggestion to upgrade to version 1.1.1.
7. **Monitor usage** – As buyers purchase and use the product, the seller monitors sales, usage, and other metrics using the AWS Marketplace [Seller reports and data feeds \(p. 273\)](#).
8. **Remove the product** – When the product is no longer needed, the seller removes it from AWS Marketplace.

In this example, the seller created three different versions of the AMI in the product, but only two were available to new buyers (prior to removing the product).

To make modifications to versions or the product information, you create [Change requests \(p. 65\)](#) in the AWS Marketplace Management Portal.

For detailed instructions on the steps to create and manage your AMI-based product, see [Single-AMI products \(p. 66\)](#).

AMI product codes

A unique product code is assigned to your product when you create it in AWS Marketplace. That product code is associated with the AMIs for your product and is used to track usage of your product. Product codes are propagated automatically as buyers work with the software. For example, a customer subscribes and launches an AMI, configures it, and produces a new AMI. The new AMI still contains the original product code, so correct usage tracking and permissions remains in place.

Note

The product *code* is different than the product *ID* for your product. Each product in AWS Marketplace is assigned a unique product ID. The product ID is used to identify your product in the AWS Marketplace catalog, in customer billing, and in seller reports. The product code is attached to instances created from your AMI as instance metadata. When an AMI with that product code is used to create an instance, the customer will get a bill that shows the associated product ID. After you create your product, find the product code and the product ID in the AWS Marketplace Management Portal page for your product.

As a seller, your software can get the product code for the running Amazon Elastic Compute Cloud (Amazon EC2) instance at runtime from the instance metadata. You can use the product code for extra security, such as validating the product code at product start. You can't make API calls to an AMI's product code until the product has been published into a limited state for testing. For more information about verifying the product code, see [Verifying your software is running on your AWS Marketplace AMI \(p. 89\)](#).

Change requests

To make changes to a product or version in AWS Marketplace, you submit a **change request** through the AWS Marketplace Management Portal. Change requests are added to a queue and can take from minutes to days to resolve, depending on the type of request. You can see the status of requests in the AWS Marketplace Management Portal.

The types of changes you can request for AMI-based products include:

- Update product information displayed to buyers.
- Update version information displayed to buyers.
- Add a new version of your product.
- Restrict a version so that new buyers can no longer access that version.
- Update the AWS Regions that a product is available in.
- Update the pricing and instance types for a product.
- Remove a product from AWS Marketplace.

For more information, see [Creating a change request \(p. 68\)](#).

Note

Some change requests require you to use product load forms to create the request. See the following section.

Product Load Forms

Typically, when you create or edit your product, you work within the AWS Marketplace Management Portal user interface to make the changes that you want. However, a few operations direct you to use a *Product Load Form* (PLF).

A PLF is a spreadsheet that contains all the information about a product. There are several ways that you can get the PLF:

- You can download the PLF for an existing product from the product's details page in the AWS Marketplace Management Portal.
- You are prompted to download the PLF when you select a menu item for an action that requires it. For example, if you choose to create a new monthly billed server product, you will be prompted to download the appropriate PLF.

If the action is an edit to an existing product, the PLF is pre-populated with the information for that product, so you only need to change the details that you are updating.

- If you need a new, blank PLF, there are links to PLFs, based on the type of product you want to create, on the AWS Marketplace Management Portal [File upload](#) page.

After you have completed your PLF, upload it to the AWS Marketplace Management Portal [File upload](#) page. The PLF itself has more detailed instructions in the **Instructions** tab.

Single-AMI products

This section discusses how you can work with products in AWS Marketplace that are made up of a single Amazon Machine Instance (AMI). Customers can use AMIs to create Amazon EC2 instances with your product already installed and configured.

Topics

- [Prerequisites \(p. 66\)](#)
- [Creating your product \(p. 67\)](#)
- [Creating a change request \(p. 68\)](#)
- [Getting status of a request \(p. 69\)](#)
- [Updating product information \(p. 69\)](#)
- [Updating version information \(p. 70\)](#)
- [Adding a new version \(p. 70\)](#)
- [Giving AWS Marketplace access to your AMI \(p. 71\)](#)
- [Restricting a version \(p. 72\)](#)
- [Removing a product from AWS Marketplace \(p. 73\)](#)
- [Common errors when submitting change requests \(p. 74\)](#)

Prerequisites

Before you get started, you must complete the following prerequisites:

1. Have access to the AWS Marketplace Management Portal. This is the tool that you use to register as a seller and manage the product that you sell on AWS Marketplace. To learn more about getting

- access to the AWS Marketplace Management Portal, see [Policies and permissions for AWS Marketplace sellers \(p. 346\)](#).
2. Register as a seller and, if you want to charge for your products, submit your tax and banking information. To learn more about becoming a seller, see [Getting started as a seller \(p. 4\)](#).
 3. Have a product that you want to sell. For AMI-based products, this typically means you have created or modified server software, and you have created an AMI for your customers to use. To learn more about preparing an AMI for use in AWS Marketplace, see [Best practices for building AMIs \(p. 88\)](#).

Creating your product

Create AMI-based products by using the AWS Marketplace Management Portal.

To create a single-AMI product

1. Open the AWS Marketplace Management Portal at <https://aws.amazon.com/marketplace/management/tour/>, and then sign in to your seller account.
2. From the **Products** menu, choose **Server**. Or, you can go directly to the [Server Products](#) page.
3. From the **Current Server Products** tab, select **Create server product** and then select one of the licensing types for single AMI products:
 - **Bring your own license (BYOL)** – A product that the user gets a license from you outside of AWS Marketplace. It can be either a paid or free license.
 - **Free** – A product that is free for your subscribers to use. (They will still pay charges for any associated Amazon Elastic Compute Cloud (Amazon EC2) instance or other AWS resources.)
 - **Paid hourly or hourly-annual** – A product that the buyer pays for either on an hourly basis or hourly with an annual contract. AWS does the metering based on the product code on the AMI.
 - **Paid monthly** – A product that the buyer is billed for monthly by AWS.

Note

There is one other type of licensing for AMI-based products: Usage-based. This licensing type applies when your product integrates with the AWS Marketplace Metering Service to provide custom metering based on your customers' usage. To create a product that has usage-based pricing, you must download, complete, and upload a Product Load Form (PLF).

For more information about PLFs, see [Product Load Forms \(p. 66\)](#).

For more information about the different types of licensing, see [AMI pricing models \(p. 91\)](#).

4. Based on your selection, fill out the information for the new product, and choose **Submit**.

Note

If you select **Paid monthly**, you will be asked to download a Product Load Form (PLF).

5. Verify that the request appears on the **Requests** tab with the **Under Review** status. You can return to this page to see the status of your request as it is processed.

Note

Product verification and publication is a manual process, handled by the AWS Marketplace Seller Operations team. It can take 3–5 days to publish your initial product version, if there are no errors. For more details about timing, see [Timing and expectations \(p. 265\)](#).

When your product is initially published, it's only accessible to your AWS account (the one you used to create the product). If you view the product from the **Server products** page, you can select **View on AWS Marketplace** to view the product details as it will appear in AWS Marketplace for buyers. This detail listing isn't visible to other AWS Marketplace users.

This capability allows you to test your product (and even publish multiple versions for testing) before releasing it publicly. If you need to make the product available to additional test accounts, or to publish your product publicly, contact the [AWS Marketplace Seller Operations team](#).

For more information about preparing your product information and submitting it for publication, see the following resources:

- [Preparing your product \(p. 35\)](#)
- [Submitting your product for publication \(p. 261\)](#)

For more information about preparing your AMI for submission to AWS Marketplace, see the following resources:

- [Best practices for building AMIs \(p. 88\)](#)
- [AMI product checklist \(p. 112\)](#)
- [AMI security policies \(p. 113\)](#)

Creating a change request

To make modifications to versions or the product information, you create a *change request* in the AWS Marketplace Management Portal.

Note

You can also create change requests using the [AWS Marketplace Catalog API](#).

To create a change request

1. Open the AWS Marketplace Management Portal at <https://aws.amazon.com/marketplace/management/tour/>, and sign in to your seller account, then go to the [Server products](#) page.
2. On the **Current server products** tab, select the product you want to modify.
3. Choose an option from the **Request changes** dropdown.

For most change requests, you simply fill out the UI form and submit. However, for certain changes, you must download, complete, and then upload a Product Load Form (PLF). This is a spreadsheet that contains a form for you to fill out with the required information. When you choose one of these change requests, you will be prompted to download the correct PLF for the request you are attempting to create. The PLF is pre-populated with information from your existing product details. You can upload your completed PLF to the AWS Marketplace Management Portal [File upload](#) page.

Note

We strongly recommend that you download and use the most recent PLF. The form is regularly updated with new information, including instance types and Regions as they become available. You can find the latest PLF for a product from the **Server products** page, by selecting the product and then choosing **Download Product Load Form**.

For more information about the status of a change request, see [the section called "Getting status of a request" \(p. 69\)](#). For insight into potential issues with change requests, see [Common errors when submitting change requests \(p. 74\)](#).

For more details about specific change requests, see the following resources:

- [Updating product information \(p. 69\)](#)
- [Updating version information \(p. 70\)](#)
- [Adding a new version \(p. 70\)](#)
- [Restricting a version \(p. 72\)](#)

Getting status of a request

After you submit a change request, you can see the status of your request from the **Requests** tab of the [Server products](#) page of the AWS Marketplace Management Portal. The status could be any of the following:

- **Under review** means that your request is being reviewed. Some requests require manual review by the AWS Marketplace team but most are reviewed automatically in the system.
- **Succeeded** means that your request is complete. Your product or version has been updated as you requested.
- **Action required** means that you need to update your request to fix an issue or answer a question about the request. Select the request to see the details, including any issues.
- **Failed** means that something went wrong with the request, and you should create a new request for the change, with the same data.

Updating product information

After you have created your product, you may want to change some of the information associated with it in AWS Marketplace. For example, if a new version modifies the description or highlights of the product, you can edit the product information with the new data.

To update product information

1. Open the AWS Marketplace Management Portal at <https://aws.amazon.com/marketplace/management/tour/>, and then sign in to your seller account.
2. Go to the [Server Products](#) page, and on the **Current server products** tab, select the product you want to modify.
3. From the **Request changes** dropdown, choose **Update product information**.
4. Update any of the following fields that you need to change:
 - **Product title**
 - **SKU**
 - **Short description**
 - **Long description**
 - **Product logo image URL**
 - **Highlights**
 - **Product categories**
 - **Keywords**
 - **Product video URL**
 - **Resources**
 - **Support information**

Note

For details about the logo format, see [Company and product logo requirements \(p. 262\)](#).

5. Select **Submit**.
6. Verify that the request appears on the **Requests** tab with the **Under review** status. You may need to refresh the page to see the request on the list.

You can check the status of your request at any time from the **Requests** tab of the [Server Products](#) page. For more information, see [Getting status of a request \(p. 69\)](#).

Updating version information

After a version is created, it can be helpful to provide updated information to your buyers by modifying the information associated with the version. For example, if you plan to restrict version 1.0 after version 1.1 is released, you can update the description of version 1.0 to direct buyers to version 1.1, with the date that the version will be restricted. You update the version information from the AWS Marketplace Management Portal.

To update version information

1. Open the AWS Marketplace Management Portal at <https://aws.amazon.com/marketplace/management/tour/>, and then sign in to your seller account.
2. Go to the [Server Products](#) page, and on the **Current server products** tab, select the product you want to modify.
3. From the **Request changes** dropdown, choose **Update version information**.
4. On the **Update version** page, select the version that you want to update.
5. Update any of the following information that you need to modify:
 - **Release notes**
 - **Usage instructions**
 - **64-bit (x86) Amazon Machine Image (AMI)** – Details on usage and security group
6. Select **Submit**.
7. Verify that the request appears on the **Requests** tab with the **Under review** status.

Note

You can't use this procedure to update the version title, or the AMI associated with the version. In this case, [create a new version \(p. 70\)](#) and [restrict the previous one \(p. 72\)](#).

You can check the status of your request at any time from the **Requests** tab of the [Server Products](#) page. For more information, see [Getting status of a request \(p. 69\)](#).

Adding a new version

You can add a new version of your product when you make changes to the product, the base image, or any other time you need to modify the AMI for the product. Add a new version of your product from the AWS Marketplace Management Portal.

Note

For information about creating an AMI for AWS Marketplace, see [Best practices for building AMIs \(p. 88\)](#).

To add a new version

1. Open the AWS Marketplace Management Portal at <https://aws.amazon.com/marketplace/management/tour/>, and then sign in to your seller account.
2. Go to the [Server products](#) page, and on the **Current server products** tab, select the product you want to modify.
3. From the **Request changes** dropdown, choose **Add new version**. The **Add a new version** form appears, pre-populated with the information from your most recent version.
4. In the **Version information** section, provide the following information:
 - **Version title** – Enter a valid string (for example **1.1** or **Version 2.0**). It must be unique across the product.

- **Release notes** – Enter text to describe details about this version.
5. In the **New Amazon Machine Image (AMI)** section, provide the following information:
 - **Amazon Machine Image ID** – Enter the AMI ID for the AMI that you want to use for this version. You can find the AMI ID from the [list of AMIs in the console](#). The AMI must exist in the US East (N. Virginia) Region, and in your AWS Marketplace Seller account.
 - **IAM access role ARN** – Enter the Amazon Resource Name (ARN) for an AWS Identity and Access Management (IAM) role that allows AWS Marketplace to gain access to your AMI. For instructions on how to create the IAM role, see [Giving AWS Marketplace access to your AMI \(p. 71\)](#). Use the standard format for an IAM ARN, for example: `arn:aws:iam::123456789012:role/RoleName`. The ARN must exist in your AWS Marketplace Seller account.
 - **OS user name** – For Linux-based AMIs, enter the name of a user that can be used to sign into the instance. We recommend using `ec2-user`.
 - **Scanning port** – Enter the port number that can be used to log into the operating system: the SSH port for a Linux AMI or the RDP port for a Windows AMI.
 6. If it is not already, expand the **Configuration settings to publish the AMI to the AWS Marketplace customer website** section, then provide the following information:
 - **Usage instructions** – Enter instructions for using the AMI or a link to more information about using the AMI. For example: *To get started with the product, navigate to <https://example.com/usage.htm>.*
 - **Endpoint URL** – Provide information about how the buyer can access the software after they create an instance. Enter the **Protocol** (`https` or `http`), the **Relative URL** (for example, `/index.html`), and the **Port** (for example, `443`) that buyers can use to access your product. (The host name depends on the EC2 instance, so you only need to provide the relative path).
 - **Operating system (OS)** – Enter the name of the OS used by the AMI (for example, *Amazon Linux*).
 - **OS version** – Enter the specific version of the OS in the AMI.
 - **Recommended instance type** – Choose the instance type that buyers get by default.
 - **Security group recommendations** – Enter the information for one or more recommendations, including the protocol (`TCP` or `UDP`), range of ports to allow, and list of IPv4 CIDR IPs (in the form `xxx.xxx.xxx.xxx/nn`, for example, `192.0.2.0/24`).
 7. Select **Submit** to submit the request to add your new version.
 8. Verify that the request appears on the **Requests** tab with the **Under review** status. If there are errors to fix, the page displays the errors in a table at the top of the page, and the specific fields that need to be updated display in red.

You can check the status of your request at any time from the **Requests** tab of the [Server Products](#) page. The new version will be reviewed and, if successful, published as a new public version of your product. If there is an issue, the status may show **Action required**. Select the request to see details, including any issues.

If your request is successful, your existing users will receive an email notifying them the new version is available, linking to the version's release notes, and suggesting that they upgrade to the latest version. You will also receive a copy of the email in your root AWS account email.

Giving AWS Marketplace access to your AMI

When you create a request that includes adding a new AMI to AWS Marketplace, the AMI must be copied into the AWS Marketplace system and then scanned for security issues. You must give AWS Marketplace access to the AMI by creating an AWS Identity and Access Management (IAM) role with permissions to perform actions on your AMI and a trust policy that allows AWS Marketplace to assume the role. You only need to create the IAM role once.

To create a role for AWS Marketplace AMI assets ingestion

1. Sign in to the AWS Management Console, open the IAM console and go to the [Roles page](#).
2. Select **Create role**.
3. On the **Create role** page, make the following selections:
 - **Select type of trusted entity** – Choose **AWS Service**.
 - **Choose a use case** – Choose **AWS Marketplace**.
 - **Select your use case** – Choose **Marketplace – AMI Assets Ingestion**.
 - To move to the next page, select **Next: Permissions**.
4. Select the **AWSMarketplaceAmiIngestion** policy. Add a permissions boundary if required, and then select **Next: Tags** to continue.

Note

You can use permissions boundaries to limit the access that you give AWS Marketplace with this role. For more information, see [Permissions boundaries for IAM entities](#) in the *AWS Identity and Access Management User Guide*.

5. To continue, select **Next: Review**.
6. Provide a name for the role, and select **Create role**.
7. You should see "The role *rolename* has been created" at the top of the page, and the role should appear in the list of roles.

On this page, when you select the role that you just created, you can see its ARN in the form `arn:aws:iam::123456789012:role/exampleRole`. Use the ARN for the **IAM access role ARN** when you create change requests, for example, when [adding a new version \(p. 70\)](#) to your product.

Restricting a version

If you want to prevent buyers from accessing a specific version of your public product, you can restrict that version.

Note

Buyers that previously subscribed to the product will still be able to use the version. AWS Marketplace guidelines require that you continue to offer support to existing buyers for 90 days after restricting the version.

To restrict a version

1. Open the AWS Marketplace Management Portal at <https://aws.amazon.com/marketplace/management/tour/>, and then sign in to your seller account.
2. Go to the [Server Products](#) page, and on the **Current server products** tab, select the product you want to modify.
3. From the **Request changes** dropdown, choose **Restrict version**.
4. On the **Restrict version** page, select the version (or versions) that you want to restrict.
5. Select **Submit** to submit your request for review.
6. Verify that the **Requests** tab show the **Request status** as **Under review**. When the request completes, the status is **Succeeded**.

Note

You can't restrict all versions of a product. If you try to restrict the last remaining public version of a product, you will receive an error. To completely remove a product, see [the section called "Removing a product from AWS Marketplace" \(p. 73\)](#).

You can check the status of your request at any time from the **Requests** tab of the [Server Products](#) page. For more information, see [Getting status of a request \(p. 69\)](#).

Note

Restricting a version can take up to 3 days to complete.

If your request is successful, your existing users will receive an email notifying them of the version restriction, and suggesting that they use the most recent version available. You will also receive a copy of the email in your root AWS account email.

Removing a product from AWS Marketplace

After your product is published, you can remove (also referred to as *sunset*) it from AWS Marketplace. To remove a product, identify the product and submit a request to remove it, along with a reason for removal and a contact email address for you. You can also provide a replacement product ID if you're replacing the current product with a new one. After you request product removal, new customers will no longer be able to subscribe to the product. You're required to support any existing customers for a minimum of 90 days. We process requests for product removal from AWS Marketplace with the following conditions:

- The product is removed from AWS Marketplace search, browse, and other discovery tools. Any **Subscribe** button or functionality is disabled, and messaging on the page clearly indicates the product is no longer available. Note that the product detail page is still accessible using the URL and may be indexed in public search engines.
- A reason for removal must be specified (for example, end of support, end of product updates, or replacement product). For the requirements for continuing support for removed products, see [Terms and Conditions for AWS Marketplace Sellers](#).
- AWS Marketplace contacts current buyers via email message informing them of the product removal, reasons for the removal, and to provide seller contact information.
- Current buyers *do* retain access to the software until they cancel their subscription. They aren't affected in any way by the product's removal.

To remove a product created using the AWS Marketplace Management Portal

1. Open the AWS Marketplace Management Portal at <https://aws.amazon.com/marketplace/management/tour/>, and then sign in to your seller account.
2. Choose the **Products** tab, and then choose **Server**.
3. On your product page, under **Current server products**, locate the product that you want to remove. From the **Actions** column on the **Select action** menu, choose **Unpublish product**.
4. On the **Unpublish Product** page, for **Request Reason**, enter the reason that you're requesting the product's removal.
5. (Optional) Provide a **Replacement Product ID**, if there is another product that will take the place of the product you are removing.
6. For **Contact Information**, enter the email address that AWS can use to contact you with any questions.
7. Review the information for accuracy, and then choose **Submit Sunset Request**.

A **What's next** informational page displays after you submit the product removal request. The AWS Marketplace Seller Operations team reviews and processes your request. Check the status of your submission by viewing **Requests**.

After your product is removed, the product appears in the **Current Products** list in the AWS Marketplace Management Portal. In **Current Products**, the only action that you can perform is downloading the spreadsheet for the product. You can't edit or submit another sunset request.

If you have questions about product removals, contact the [AWS Marketplace Seller Operations team](#).

Common errors when submitting change requests

When you make changes to your product's information, you sometimes run into errors. Following are some common issues and suggestions for how to fix them:

- **Scanning your AMI** – Several issues could happen when scanning your AMI:
 - You have not granted AWS Marketplace permissions to scan your AMI. Grant AWS Marketplace permissions to access it. Or you have granted permissions, but the permissions boundary is too restrictive. For more information, see [Giving AWS Marketplace access to your AMI \(p. 71\)](#).
 - If scanning finds security issues or Common Vulnerabilities and Exposures (CVEs) in your AMI, make sure you're using the latest patches for the operating system in your image. For more information, see [AMI security policies \(p. 113\)](#).

For general guidelines about building an AMI, see [Best practices for building AMIs \(p. 88\)](#).

- **AWS Marketplace Management Portal fields** – Some fields in the AWS Marketplace Management Portal require very specific information:
 - If you are unsure about what the field is requesting, try checking the details in the console. Most fields have text descriptions above the field, and formatting requirements below the field.
 - If you try to submit a form with one or more invalid fields, a list of issues is shown. A recommended action is provided to help you fix the issue.
 - If you're asked to provide an ARN, you will typically find it elsewhere in the console. For example, the ARN for the IAM role that you created to give AWS Marketplace access to your AMI is found on the [Roles page](#) in the IAM console. ARNs all have a similar format. For example, an IAM role ARN is in the form `arn:aws:iam::123456789012:role/exampleRole`.
 - Your logos and videos must be provided as a URL directly to the content. For more information about logo formats, see [Company and product logo requirements \(p. 262\)](#).

For more information about submitting products and version change requests, see [Submitting your product for publication \(p. 261\)](#).

- **Product Load Form (PLF) issues** – PLFs contain instructions that are included in the spreadsheet. Overall instructions are provided in the Instructions table. Each field has instructions for how to fill it out—select the field to reveal the instructions.
- **Request in Progress** – Some requests cannot happen in parallel. You can only have one request to update specific information in progress for a product at a time. You can see all of your requests still under review on the **Requests** tab of the **Server products** page in AWS Marketplace Management Portal. If you have a pending request that you did not intend, you can cancel it and then submit a new request with the change that you want to make.
 - You can't update version information when an update (to add or restrict) a version is ongoing.
 - If there is a request pending from the AWS Marketplace Seller Operations team, you can't submit any new changes.
- **Unexplained error** – If your submission fails with no explanation, try again. Occasionally, server load causes a submission to fail.

If you're still having problems with a change request, contact the [AWS Marketplace Seller Operations team](#).

AMI-based delivery using AWS CloudFormation

AWS Marketplace sellers can list AMI-based products that are delivered to AWS Marketplace buyers by using AWS CloudFormation templates. You can use the templates to define a cluster or distributed

architecture for the products or to select different AMI combinations or product configurations. The AWS CloudFormation templates can be configured to deliver a single Amazon Machine Image (AMI) or multiple AMIs along with associated config files and Lambda functions. Buyers can browse the selection of solutions on AWS Marketplace, buy with one click, and deploy by using AWS CloudFormation templates that you provide.

Multi-AMI solutions can contain up to 20 AMIs and up to 20 AWS CloudFormation templates. Each AWS CloudFormation template can reference any combination or subset of the AMIs contained in the solution. The buyer purchases a single solution that entitles them to all of the AMIs in that product. When the product has multiple AMIs, each AMI has its own unique product code and can be priced and metered separately. However, individual components of a solution aren't discoverable or procurable outside the context of the product.

If you have existing single-AMI products, you can't migrate or combine them into a new multi-AMI listing. However, your new solution can feature the same software or copies of AMIs used by existing products. Each listing created on AWS Marketplace is a listing with new product codes.

You can also include Lambda functions in a Serverless Application with your AMI so that buyers can deploy them through CloudFormation. For instructions on how to include Lambda functions and serverless applications with your AMI, see [Adding serverless application components \(p. 79\)](#).

Building your product listing

To submit your product, you need to prepare and validate your AMI(s), create your AWS CloudFormation template(s), create a topology diagram, complete the product load form, and submit the materials to AWS Marketplace. We recommend that you start by creating and validating your AMI(s) and then complete and validate the AWS CloudFormation template(s). After you complete those steps, you should create a topology diagram and estimate the software and infrastructure price. AWS Marketplace validates your submission and works with you to make your product public. Use the [AWS Pricing Calculator](#) to help estimate the infrastructure cost for your template. Provide AWS Marketplace with a link to your saved calculator configuration. The following are limitations of multi-AMI solution products:

- Updating existing AWS Marketplace products from a standalone product to a multi-AMI product isn't supported. To make a product available in a multi-AMI product, copy the AMI and submit it as a component to a new multi-AMI product. The resulting AMI has a unique product code that's different from the previous product's code.
- Multi-AMI solutions aren't visible on the **AWS Marketplace** tab of the **Launch Instance** page in the Amazon Elastic Compute Cloud (Amazon EC2) console.
- An AWS CloudFormation template must not launch AMIs outside of those listed in the multi-AMI solution.
- AWS CloudFormation templates must be submitted in the form of a public URL. All nested template URLs contained in the template must also be publicly accessible.

Preparing your AWS CloudFormation template

To build your AWS CloudFormation templates, you must meet the template prerequisites and provide the required input and security parameters. When submitting your AWS CloudFormation template, use the guidelines in the following sections.

Template prerequisites

- Verify that the template is launched successfully through the AWS CloudFormation console **in all Regions enabled for your product**. You can use this tool to test your templates: <https://github.com/aws-quickstart/taskcat>.
- If you are creating a single-AMI product, the template must contain only one AMI.

- AMIs must be in a [mapping table](#) for each Region. The AWS Marketplace team updates the AMI IDs after they're cloned. Your source AMI must be in `us-east-1` and the other regions can use placeholders. See the following YAML example.

```
Mappings:
  RegionMap:
    us-east-1:
      ImageId: ami-0123456789abcdef0
    us-west-1:
      ImageId: ami-xxxxxxxxxxxxxxxxxxx
    eu-west-1:
      ImageId: ami-xxxxxxxxxxxxxxxxxxx
    ap-southeast-1:
      ImageId: ami-xxxxxxxxxxxxxxxxxxx
```

- Build templates so that they do not depend on the use in a particular availability zone (AZ). Not all customers have access to all AZs, and AZs are mapped differently for different accounts.
- You can include dependencies such as Lambda functions, config files, and scripts with your AMI. For more information, see [Create a serverless application \(p. 79\)](#).
- If you're building a clustered solution using an Auto Scaling group, we recommend that you account for a scaling event. The new node should join the running cluster automatically.
- Even for single-node products, we recommend using an [Auto Scaling group](#).
- If your solution involves a cluster of multiple instances, consider using placement groups if you want low network latency, high network throughput, or both among the instances.
- If your solution involves Docker containers, you must incorporate the Docker images into the AMI.
- For ease of review by the AWS Marketplace team and transparency to the customer, we recommend that you add comments in your **UserData** section.

Template input parameters

- Input parameters to the template must not include the AWS Marketplace customer's AWS credentials (such as passwords, public keys, private keys, or certificates) or personal information such as email address.
- Do not set defaults for parameters such as remote access, CIDR/IP, or passwords for databases. The customer must provide these as input parameters.
- For sensitive inputs such as passwords, choose the `NoEcho` property and enable stronger regular expression. For other inputs, set the most common inputs along with appropriate helper text.
- Use AWS CloudFormation parameter types for inputs where available.
- Use `AWS::CloudFormation::Interface` to group and sort input parameters.

Network and security parameters

- Ensure that the default SSH port (22) or RDP port (3389) isn't open to 0.0.0.0.
- Instead of using the default virtual private cloud (VPC), we recommend that you build a VPC with appropriate access control lists (ACLs) and security groups. Only AWS accounts created before December 4, 2013, support EC2-Classic.
- Access to the customer's AWS environment should be enabled using an IAM role to call [AssumeRole](#) from the AWS Security Token Service.
- Set IAM roles and policies to [grant the least privilege](#) and enable write access only when absolutely necessary. For example, if your application needs only `S3:GET`, `PUT`, and `DELETE` operations, specify those actions only. We don't recommend the use of `S3:*` in this case.

After your template is received, AWS Marketplace validates the product configuration and information and provides feedback for any required revisions.

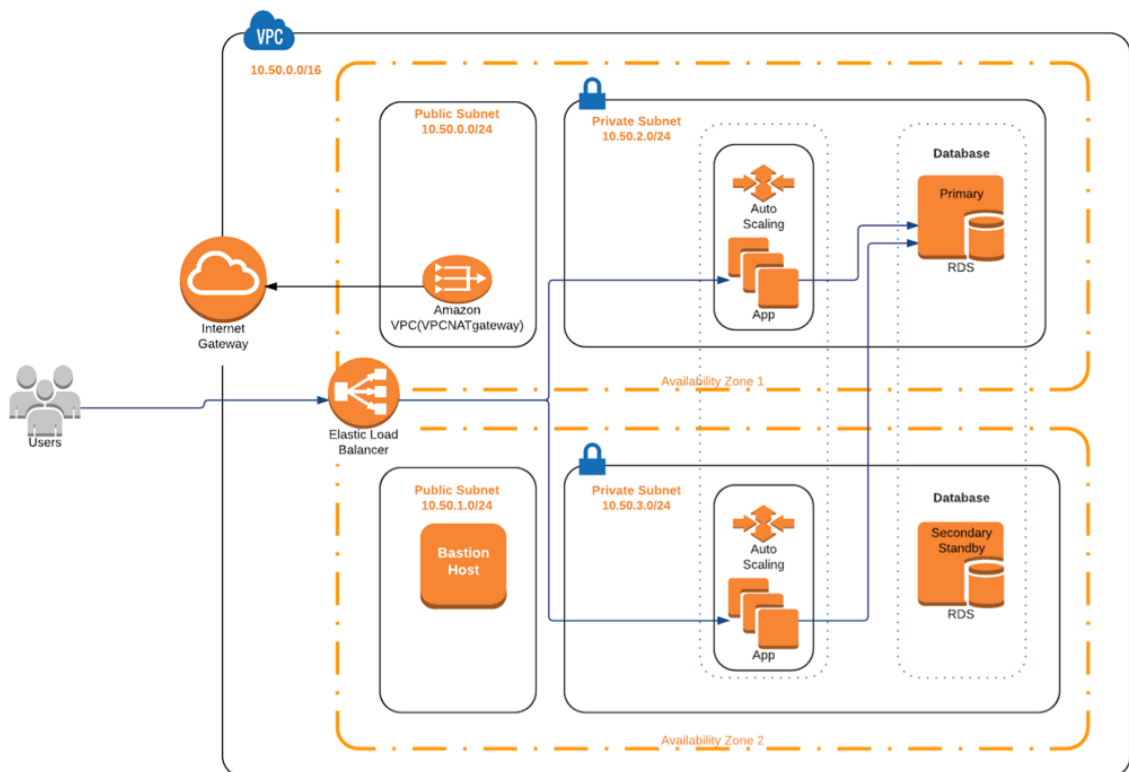
Getting the cost estimate for your template infrastructure

The infrastructure cost estimate for each template displayed to customers is based on an estimate that you provide by using the [AWS Pricing Calculator](#). The estimation should include the list of services to be deployed as part of the template, along with the default values for a typical deployment.

After you calculate the template's estimated monthly cost, provide AWS Marketplace with the **Save and Share** link for the US East (N. Virginia) Region. This is part of the submission process.

Topology diagram

You must provide a topology diagram for each template. The diagram must use the [AWS product icons](#) for each AWS service deployed through the AWS CloudFormation template, and it must include metadata for the services. The diagram must be 1100 x 700 pixels in size. Make sure that your diagram meets this sizing requirement to avoid cropping or stretching, as shown in the following image.



Meeting the submission requirements

To submit products delivered by using AWS CloudFormation templates, you must provide the following resources:

- AWS CloudFormation template or templates

- A single-AMI product can have one to three AWS CloudFormation templates
- A multi-AMI product can have up to 20 AWS CloudFormation templates
- The estimated infrastructure price for the default configuration of each template
- A topology diagram and topology metadata
- Completed product form (available from the [AWS Marketplace Management Portal](#))
 - For single-AMI products, use the [Commercial Product](#) form
 - For multi-AMI products, use the [Multi-AMI Product form](#)

The product forms include example submissions for your reference.

For each product, most of the required product data and metadata are the same as for traditional single-AMI products. Therefore, each AMI that is delivered by using an AWS CloudFormation template must continue to meet the standards and requirements described for AWS Marketplace.

For each AWS CloudFormation template, you must also provide the following information.

| Field | Description | Restrictions |
|-------------------|---|-----------------|
| Title | Title of the topology. This appears on the detail and fulfillment pages and the pop-up that shows the topology details. | 50 characters |
| Short description | This appears on the detail and fulfillment pages. | 200 characters |
| Long description | This appears in the topology details pop-up. | 2000 characters |

For multi-AMI products, the following fields are required:

- Solution title
- Solution short description
- Solution long description
- For AWS CloudFormation templates (up to 20 per solution)
 - Deployment title (per template)
 - Short description (per template)
 - Long description (per template)
 - Architecture diagram (per template)
 - Infrastructure pricing estimate (per template)
 - List of products/components contained in this AWS CloudFormation template
 - List of regions supported by this AWS CloudFormation template

Submitting your product request

Use the [AWS Marketplace Management Portal](#) to submit your product. On the **Assets** tab, choose **File Upload**. Upload any files you want to submit and enter a brief description. Allow three to five weeks for request processing, including:

- Review of the AWS CloudFormation template, AMI, and metadata for the AMI and AWS CloudFormation template
- Publication of your AWS CloudFormation template to AWS Marketplace products

Adding serverless application components

You can create a product that includes one or more Amazon Machine Images (AMIs), delivered using one or more AWS CloudFormation templates, with serverless components incorporated into the product. For example, create a product with one AMI configured as a controller server and another AMI configured as a worker server, delivered as a AWS CloudFormation stack. The AWS CloudFormation template used to create the stack can include the definition to set up an AWS Lambda function that is triggered by an event in one of the servers.

When you use this approach to design your product, you can simplify the architecture and make it easier for your buyers to launch. This approach can also make it easier for you to update your product.

For information about creating AMIs for your product, see [AMI-based products \(p. 63\)](#). For information about completing AWS CloudFormation templates for your product, see [AMI-based delivery using AWS CloudFormation \(p. 74\)](#).

When you define your serverless application, you use an AWS Serverless Application Model (AWS SAM) template that you store in the AWS Serverless Application Repository. AWS SAM is an open-source framework for building serverless applications. During deployment, AWS SAM transforms and expands the AWS Serverless Application Model syntax into AWS CloudFormation syntax. The AWS Serverless Application Repository is a managed repository for serverless applications. It makes it possible for you to store and share reusable applications so buyers can assemble and deploy serverless architectures. To create and offer this type of product, complete the following steps:

Steps

- [Create a serverless application \(p. 79\)](#)
- [Publish your application to the repository \(p. 80\)](#)
- [Create the CloudFormation template \(p. 81\)](#)
- [Submit your CloudFormation template and configuration files \(p. 83\)](#)
- [Update your AWS Serverless Application Repository application permissions \(p. 83\)](#)
- [Share your AMI \(p. 84\)](#)
- [Submit your CloudFormation product with AMI and serverless application \(p. 84\)](#)

AWS Marketplace reviews and validates your product before your listing is created. If there are issues you must resolve before the offer is listed, we will send you an email message.

As part of fulfilling a subscription, we copy the AMIs, serverless applications, and AWS CloudFormation templates to an AWS Marketplace-owned repository in each AWS Region. When a buyer subscribes to your product, we give them access, and also notify them when you update your software.

Create a serverless application

Your first step is to package the AWS Lambda functions used to create your serverless application. Your application is a combination of Lambda functions, event sources, and other resources that work together to perform tasks. A serverless application can be as simple as one Lambda function or contain multiple functions with other resources, such as APIs, databases, and event source mappings.

Use the AWS SAM to define a model for your serverless application. For descriptions of property names and types, see [AWS::Serverless::Application](#) in AWS Labs on GitHub. The following is an example of an AWS SAM template with a single Lambda function and AWS Identity and Access Management (IAM) role.


```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Description: An example of SAM template with Lambda function and IAM role

Resources:
  SampleFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: 'com.sampleproject.SampleHandler::handleRequest'
      Runtime: java8
      CodeUri: 's3://DOC-EXAMPLE-BUCKET/2EXAMPLE-1234-4b12-ac37-515EXAMPLEe5-lambda.zip'
      Description: Sample Lambda function
      Timeout: 120
      MemorySize: 1024
      Role:
        Fn::GetAtt: [SampleFunctionRole, Arn]

# Role to execute the Lambda function
SampleFunctionRole:
  Type: "AWS::IAM::Role"
  Properties:
    AssumeRolePolicyDocument:
      Statement:
        - Effect: "Allow"
          Principal:
            Service:
              - "lambda.amazonaws.com"
          Action: "sts:AssumeRole"
    ManagedPolicyArns:
      - "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
    Policies:
      - PolicyName: SFNXDeployWorkflowDefinitionPolicy
        PolicyDocument:
          Statement:
            - Effect: "Allow"
              Action:
                - "s3:Get*"
              Resource: "*"
    RoleName: "SampleFunctionRole"
```

Publish your application to the repository

To publish an application, you first upload the application code. Store your code artifacts (for example, Lambda functions, scripts, configuration files) in an Amazon S3 bucket that your account owns. When you upload your application, it's initially set to private, meaning that it's only available to the AWS account that created it. You must create an IAM policy that grants AWS Serverless Application Repository permissions to access the artifacts you uploaded.

To publish your serverless application to the serverless application repository

1. Open the Amazon S3 console at <https://console.aws.amazon.com/s3/>.
2. Choose the Amazon S3 bucket that you used to package your application.
3. Choose the **Permissions** tab.
4. Choose **Bucket Policy**.
5. Copy and paste the following example policy statement.

Note

The example policy statement will produce an error until values for `aws:SourceAccount` and `Resource` are updated in following steps.

```
{
```

```
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Principal": {
      "Service": "serverlessrepo.amazonaws.com"
    },
    "Action": "s3:GetObject",
    "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
    "Condition": {
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      }
    }
  }
]
}
```

- a. Replace `DOC-EXAMPLE-BUCKET` in the Resource property value with the bucket name for your bucket.
 - b. Replace `123456789012` in the Condition element with your AWS account ID. The Condition element ensures that the AWS Serverless Application Repository only has permission to access applications from the specified AWS account.
6. Choose **Save**.
 7. Open the AWS Serverless Application Repository console at <https://console.aws.amazon.com/serverlessrepo>.
 8. On the **My Applications** page, choose **Publish application**.
 9. Complete the required fields and any optional field, as appropriate. The required fields are:
 - **Application name**
 - **Author**
 - **Description**
 - **Source code URL**
 - **SAM template**
 10. Choose **Publish Application**.

To publish subsequent versions of your application

1. Open the AWS Serverless Application Repository console at <https://console.aws.amazon.com/serverlessrepo>.
2. In the navigation pane, from **My Applications**, choose the application.
3. Choose **Publish new version**.

For more information, see [Publishing serverless Applications Using the AWS SAM CLI](#).

Create the CloudFormation template

To build your CloudFormation templates, you must meet the template prerequisites and provide the required input and security parameters. For more information, see [Template anatomy](#) in the *AWS CloudFormation User Guide*.

In your CloudFormation template, you can reference your serverless application and your AMI. You can also use nested CloudFormation templates and reference serverless applications both in the root template and the nested templates. To reference the serverless application, you use the AWS SAM

template. You can automatically generate the AWS SAM template for your application from the AWS Serverless Application Repository. The following is an example template.

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Description: An example root template for a SAR application

Resources:
  SampleSARApplication:
    Type: AWS::Serverless::Application
    Properties:
      Location:
        ApplicationId: arn:aws:serverlessrepo:us-east-1:1234567890:applications/
TestApplication
  SemanticVersion: 1.0.0
SampleEC2Instance:
  Type: AWS::EC2::Instance
  Properties:
    ImageId: "ami-79fd7eee"
    KeyName: "testkey"
    BlockDeviceMappings:
      - DeviceName: "/dev/sdm"
        Ebs:
          VolumeType: "io1"
          Iops: "200"
          DeleteOnTermination: "false"
          VolumeSize: "20"
      - DeviceName: "/dev/sdk"
        NoDevice: {}
```

The AWS SAM template contains the following elements:

- **ApplicationID** – Your application's Amazon Resource Name (ARN). This information is located in the **My Applications** section of the AWS Serverless Application Repository.
- **SemanticVersion** – The version of your serverless application. You can find this from the **My Applications** section of the AWS Serverless Application Repository.
- **Parameter (optional)** – Application parameters.

Note

For **ApplicationID** and **SemanticVersion**, [intrinsic functions](#) aren't supported. You must hardcode those strings. The **ApplicationID** is updated when it's cloned by AWS Marketplace.

If you're planning to reference configuration and script files in your CloudFormation template, use the following format. For nested templates (`AWS::CloudFormation::Stack`), only `TemplateURLs` without intrinsic functions are supported. Note the `Parameters` content in the template.

```
AWSTemplateFormatVersion: '2010-09-09'
Metadata:
  Name: Seller test product
Parameters:
  CFRefFilesBucket:
    Type: String
    Default: "seller-bucket"
  CFRefFilesBucketKeyPrefix:
    Type: String
    Default: "cftsolutionFolder/additionCFfiles"
Resources:
  TestEc2:
    Type: AWS::EC2::Instance
    Metadata:
      AWS::CloudFormation::Init:
```

```
addCloudAccount:
  files:
    /etc/cfn/set-ai-a-settings.sh:
      source:
        Fn::Sub:
          - https://${CFTRefFilesBucket}.${S3Region}amazonaws.com/
            ${CFTRefFilesBucketKeyPrefix}/sampleScript.sh
          - S3Region:
              !If
                - GovCloudCondition
                - s3-us-gov-west-1
                - s3
            owner: root
            mode: '000700'
            authentication: S3AccessCreds
      ..
      ..
      ..
SampleNestedStack:
  Type: AWS::CloudFormation::Stack
  Properties:
    TemplateURL: 'https://sellerbucket.s3.amazonaws.com/sellerproductfolder/
nestedCft.template'
    Parameters:
      SampleParameter: 'test'
  Transform: AWS::Serverless-2016-10-31
```

Submit your CloudFormation template and configuration files

To submit your CloudFormation template and configuration and scripts files, grant AWS Marketplace permissions to read the Amazon S3 bucket where these files are stored. To do so, update your bucket policy to include the following permissions.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "assets.marketplace.amazonaws.com"
      },
      "Action": ["s3:GetObject", "s3:ListBucket"],
      "Resource": ["arn:aws:s3:::DOC-EXAMPLE-BUCKET",
        "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"]
    }
  ]
}
```

Update your AWS Serverless Application Repository application permissions

To submit your AWS Serverless Application Repository application to AWS Marketplace, you must grant AWS Marketplace permissions to read your application. To do that, add permissions to a policy associated with your serverless application. There are two ways to update your application policy:

- Go to the [AWS Serverless Application Repository](#). Choose your serverless application from the list. Select the **Sharing** tab, and choose **Create Statement**. On the **Statement configuration** page, enter the following service principal, `assets.marketplace.amazonaws.com`, in the **Account Ids** field. Then choose **Save**.
- Use the following AWS CLI command to update your application policy.

```
aws serverlessrepo put-application-policy \  
--region region \  
--application-id application-arn \  
--statements Principals=assets.marketplace.amazonaws.com,Actions=Deploy
```

Share your AMI

All AMIs built and submitted to AWS Marketplace must adhere to all product policies. Self-service AMI scanning is available in the AWS Marketplace Management Portal. With this feature, you can initiate scans of your AMIs. You receive scanning results quickly (typically, in less than an hour) with clear feedback in a single location. After your AMI has been successfully scanned, submit the AMI for processing by the AWS Marketplace Seller and Catalog Operations team by uploading your product load form.

Submit your CloudFormation product with AMI and serverless application

Keep the following in mind before you submit your product:

- You must provide a topology diagram for each template. The diagram must use the AWS product icons for each AWS service deployed through the CloudFormation template. Also, the diagram must include metadata for the services. To download our official AWS architecture icons, see [AWS Architecture Icons](#).
- The infrastructure cost estimate for each template displayed to buyers is based on an estimate that you provide by using the [AWS Pricing Calculator](#). In the estimate, include the list of services to be deployed as part of the template, along with the default values for a typical deployment.
- Complete the product load form. You can find the product load form from the AWS Marketplace Management Portal. A different product load form is required for single AMI products and multiple AMI products. In the product load form, you will provide a public URL to your CloudFormation template. CloudFormation templates must be submitted in the form of a public URL.
- Use the AWS Marketplace Management Portal to submit your listing. From **Assets**, choose **File upload**, attach your file, and then choose **Upload**. After we receive your template and metadata, AWS starts processing your request.

After you submit your listing, AWS Marketplace reviews and validates the product load form. Additionally, AWS Marketplace regionalizes AMIs and serverless applications, and updates the regional mapping for your AWS CloudFormation template on your behalf. If any issues occur, the AWS Marketplace Seller and Catalog Operations team contacts you by email.

Private images

You can use private image builds to let buyers purchase your installable software products through AWS Marketplace, and then install those products on a gold image or Amazon Machine Image (AMI) they choose from the images available to their AWS account. A *gold image* is a buyer-provided server image that includes a base operating system with modifications applied to help ensure the software adheres to the buyer's IT standards. Gold images allow buyers to better meet their internal security, compliance, and management requirements.

This topic describes how to use the [AWS Marketplace Management Portal](#) (AMMP) to upload your software binaries and/or scripts and create an installable package group for each operating system (OS) your software will run on. AWS Marketplace does a test build by installing the package group on a base

OS you specify and scans the resulting image for certain known vulnerabilities. After the image build and scan completes, you can use the AMMP to submit your product.

The following diagram shows the private image build flow.



1. You upload an installable software package to AWS Marketplace.
2. A buyer selects your product from AWS Marketplace, but wants to use their own gold image for the OS on the AMI.
3. The buyer requests a new AMI from the AWS Marketplace Private Image Build Service, specifying their gold image and your installable software package.
4. The AWS Marketplace Private Image Build Service creates a new private image that the buyer can use in their AWS account only. They can then launch the AMI from the private image configuration panes or within Amazon Elastic Compute Cloud (Amazon EC2).

Package group requirements

You can submit your package group for use on either AWS Marketplace base Linux AMIs or AWS Marketplace base Windows Server AMIs.

When you select the OS platform for your product, you will have the option to select multiple OSs and OS versions on which your package group will run. Windows Server packages will not run on Linux OSs and vice versa, so if you want your product to support private images for both OS platforms, you will need to define at least two package groups. When you define your package group, you upload the installation packages or scripts and AMMP will build and scan a test image for each OS you choose.

For your package group to successfully complete the build and scan process, you must adhere to these guidelines:

- The package group must have one of the packages or scripts marked as the installer. For example, the installer may be a batch file or script that orchestrates the installation of the other packages and provides the required parameters for an unattended installation.
- For Windows Server-based packages, the supported installer types have .msi, .ps1, .bat, and .exe extensions.
- For Linux/UNIX (or any POSIX-compliant) systems, the supported installer types have .exe, .rpm, .deb, .sh, and .run extensions.
- The entire installation process must be unattended. It cannot require any interactive input, and all parameters or switches must be included in the installer.
- The packages must install without downloading patches or configuration files (be complete) from another website.

- The installer/installation script must be synchronous. For example, the script must not exit until the packages are completely installed.
- The installer must exit with exit status 0 when the installation is successful. Any value other than 0 is used for unsuccessful installations.
- The installer cannot require a reboot during the installation. A reboot would stop the agent that tracks the test and scan process for packages. If your installer reboots, the agent is stopped and the test and scan will fail.
- The installer must not affect the network routing on the instance in such a way that the host becomes unreachable.

Submitting your package group

To submit a package group to AWS Marketplace for use with Private Image Build

1. From the [AMMP](#), choose **Assets, Private image build**.
2. Under **Manage packages**, choose **Start package**.
3. In **Enter a unique name for your package group**, type the name of your product. The name must be less than 100 characters and can only contain alphanumeric characters, underscores, and dashes. Each product name associated with the AWS account used to create and publish package groups must be unique. After you've used a name (even if the build is unsuccessful), you can't use the name again. We recommend using a naming convention with a revision number included in the file name. For example:
`[product_group_name]<product_name><version><platform><revision_number>`
4. In **Select one or more packages**, select a package from the dropdown list or choose **Browse** to locate and select the package group you want to upload.
5. Under **Select supported operating system platform**, choose either **AWS Marketplace base Linux AMIs** or **AWS Marketplace base Windows AMIs**.
6. Under **Select supported operating systems**, choose all the OSs that your package group will support, and then choose **Submit**.

For each package group you submit, a build process is completed for each OS version you chose. After you submit your package group, you are redirected to the **Scan status** page, where you can check progress of the image building and scanning process for each package group.

Scan status

After you submit your package group, you can check the current status on the **Scan status** tab. Each package group you've submitted is listed. Choose the arrow next to the package group to expand the list and show the build and scan status for each package group you selected.

Each entry will show the AMI ID, date you submitted the package group, and the status of the package group (or build). During the process, you can track the state of package groups and individual builds you have submitted. There are four states your package group submission can be in, and five states individual builds can be in.

Package group state

The package group state updates as automated steps complete. You can return to the **Scan status** page to check on progress, or if the page is open you can choose **Refresh status** to update the information on the page. The package group states are:

- **Building** – You have submitted your package group and the corresponding image(s) are being built.

- **Scanning** – You have submitted your package group and the corresponding images(s) are being scanned.
- **Successful** – All builds associated with your package group were successfully scanned. Submit your product load form.
- **Issues Found** – One or more builds for your submission failed that require your attention. Choose the **information** icon next to the status for additional troubleshooting information.
- **Investigating** – There was a problem uncovered during the build and scan process. AWS Marketplace is investigating.

Note

If your status remains in the **Investigating** state for four or more business days, contact the [AWS Marketplace Seller Operations team](#).

OS build state

On the **Scan status** page, you can choose the arrow next to the package group name to expand the entry to show each OS build that is part of the package group. The OS build states are:

- **Building** – The build of your software on the OS is in progress. This might take up to an hour to complete for each build.
- **Scanning** – The build process completed successfully and the scan is in progress. This might take several hours to complete.
- **Successful** – The build and scan process completed successfully. No further action on your part.
- **Issues found** – There was a problem with the build or the scan process that require your attention. Choose the **Information** icon next to the status for additional troubleshooting information.
- **Investigating** – The build or scan process failed. AWS Marketplace is investigating.

Note

If your status remains in the **Investigating** state for four or more business days, contact the [AWS Marketplace Seller Operations team](#).

When your package group shows a status of **Successful**, this phase is complete. Next, you can publish your package group as a new fulfillment option for your product on AWS Marketplace.

Submitting your product to AWS Marketplace

After you upload a package group to AWS Marketplace you can submit a product load form to publish it as a new fulfillment option for your product, or as a new product if it does not already exist. The load form is an Excel spreadsheet. The first tab of the spreadsheet provides instructions for providing the metadata needed to publish your product on AWS Marketplace.

To download and complete the load form

1. From the [AMMP](#), under the **Assets** tab, choose **File upload**.
2. On the **File Uploads** page, under **Product load forms and seller guides**, choose **Private Image Form**.
3. Download the product load form.
4. Complete the form.
5. From the **AMMP Assets** tab, choose **File upload**.
6. Choose the files you want to submit and enter a brief description.

AWS Marketplace creates or update your product entry. If there are any questions on your submission, AWS Marketplace will contact you for clarification. Your product is typically added or updated within five business days.

When adding a package group as a new fulfillment option for your product, consider the following options:

- Add the package group as an additional fulfillment option to an existing software version, on an existing public product on AWS Marketplace. With this approach, the software version on the AMI and package fulfillment options must match. AWS Marketplace cannot replace an AMI on an existing software version.
- If the package group has different software than what currently exists on AWS Marketplace, you can list the package group as a new software version on an existing product. Using this approach, you must provide a successfully built and scanned AMI from the AMMP **Packages** tab. You will have the option to test package fulfillment before making the new package group public. However, the AMI will be visible to buyers right away. This is consistent with the current experience for new software versions.

Best practices for building AMIs

This topic provides some best practices and references to help you build Amazon Machine Images (AMIs) for use with AWS Marketplace. AMIs built and submitted to AWS Marketplace must adhere to all AWS Marketplace product policies.

Verifying your AMI

To help verify your AMI before submitting it as a new product or version, you can use self-service scanning.

From the AWS Marketplace Management Portal, choose **Amazon Machine Image** from the **Assets** menu. Click **Add AMI** to start the scanning process. You can see the scan status of AMIs by returning to this page.

Note

To learn about giving AWS Marketplace access to your AMI, see [Giving AWS Marketplace access to your AMI \(p. 71\)](#).

Securing resell rights

You are responsible for securing resell rights for non-free Linux distributions, with the exception of AWS-provided Amazon Linux, RHEL, SUSE, and Windows AMIs.

Building an AMI

Use the following guidelines for building AMIs:

- Ensure that your AMI meets all AWS Marketplace policies, including disabling root login.
- Create your AMI in the US East (N. Virginia) Region.
- Create products from existing, well-maintained AMIs backed by Amazon Elastic Block Store (Amazon EBS) with a clearly defined lifecycle provided by trusted, reputable sources such as AWS Marketplace.
- Build AMIs using the most up-to-date operating systems, packages, and software.
- Ensure that all AMIs must start with a public AMI that uses hardware virtual machine (HVM) virtualization and 64-bit architecture.
- Develop a repeatable process for building, updating, and republishing AMIs.

- Use a consistent operating system (OS) user name across all versions and products. We recommend **ec2-user**.
- Configure a running instance from your final AMI to the end-user experience you want, and test all installation methods, features, and performance *before* submission to AWS Marketplace.
- Check port settings as follows:
 - Linux-based AMIs – Ensure that a valid SSH port is open. The default SSH port is 22.
 - Windows-based AMIs – Ensure that an RDP port is open. The default RDP port is 3389. Also, the WinRM port (5985 by default) must be open to 10.0.0.0/16.

For more information about creating an AMI, see the following resources:

[Creating Your Own AMI](#) in the *Amazon EC2 User Guide for Linux Instances*

[Creating a Custom Windows AMI](#) in the *Amazon EC2 User Guide for Windows Instances*

[How do I create an Amazon Machine Image \(AMI\) from an EBS-backed instance?](#)

[Amazon Linux AMI](#)

[Amazon EC2 Instance Types](#) and [Instance Types](#)

Verifying your software is running on your AWS Marketplace AMI

You may wish to have your software verify at runtime that it is running on an Amazon EC2 instance created from your AMI product.

To verify the Amazon EC2 instance is created from your AMI product, use the instance metadata service built into Amazon EC2. The following steps take you through this validation. For more information about using the metadata service, see [Instance metadata and user data](#) in the *Amazon Elastic Compute Cloud User Guide*.

1. Obtain the instance identity document

Each running instance has an identity document accessible from the instance that provides data about the instance itself. The following example shows using curl from the instance to retrieve the instance identity document.

```
curl http://169.254.169.254/latest/dynamic/instance-identity/document
{
  "accountId" : "0123456789",
  "architecture" : "x86_64",
  "availabilityZone" : "us-east-1e",
  "billingProducts" : null,
  "devpayProductCodes" : null,
  "marketplaceProductCodes" : [ "0vg00000000000000000000000000000000" ],
  "imageId" : "ami-0123456789abcdef1",
  "instanceId" : "i-0123456789abcdef0",
  "instanceType" : "t2.medium",
  "kernelId" : null,
  "pendingTime" : "2020-02-25T20:23:14Z",
  "privateIp" : "10.0.0.2",
  "ramdiskId" : null,
  "region" : "us-east-1",
  "version" : "2017-09-30"
}
```

2. Verify the instance identity document

You can verify that the instance identity is correct using the signature. For details about this process, see [Instance identity documents](#) in the *Amazon Elastic Compute Cloud User guide*.

3. Verify the product code

When you initially submit your AMI product for publishing, your product is assigned a [product code](#) by AWS Marketplace. You can verify the product code by checking the `marketplaceProductCodes` field in the instance identity document, or you can get it directly from the metadata service:

```
curl http://169.254.169.254/latest/meta-data/product-codes  
0vg00000000000000000000000000000000
```

If the product code matches the one for your AMI product, then the instance was created from your product.

You may also wish to verify other information from the instance identity document, such as the `instanceId` and the `instance privateIp`.

Securing an AMI

We recommend the following guidelines for creating secure AMIs:

- Architect your AMI to deploy as a minimum installation to reduce the attack surface. Disable or remove unnecessary services and programs.
- Whenever possible, use end-to-end encryption for network traffic. For example, use Secure Sockets Layer (SSL) to secure HTTP sessions between you and your buyers. Ensure that your service uses only valid and up-to-date certificates.
- When adding a new version to your AMI product, configure security groups to control inbound traffic access to your instance. Ensure that your security groups are configured to allow access only to the minimum set of ports required to provide necessary functionality for your services. Allow administrative access only to the minimum set of ports and source IP address ranges necessary. For more information about how to add a new version to your AMI product, see [Adding a new version](#) (p. 70).
- Consider performing a penetration test against your AWS computing environment at regular intervals, or consider employing a third party to conduct such tests on your behalf. For more information, including a penetration testing request form, see [AWS Penetration Testing](#).
- Be aware of the top 10 vulnerabilities for web applications, and build your applications accordingly. To learn more, see [Open Web Application Security Project \(OWASP\) - Top 10 Web Application Security Risks](#). When new internet vulnerabilities are discovered, promptly update any web applications that ship in your AMI. Examples of resources that include this information are [SecurityFocus](#) and the [NIST National Vulnerability Database](#).

For more information related to security, see the following resources:

- [Guidelines for Shared Linux AMIs](#) in the *Amazon EC2 User Guide for Linux Instances*
- [AWS Cloud Security](#)
- [The Center for Internet Security \(CIS\): Security Benchmarks](#)
- [The Open Web Application Security Project \(OWASP\): Secure Coding Practices - Quick Reference Guide](#)
- [OWASP Top 10 Web Application Security Risks](#)
- [SANS \(SysAdmin, Audit, Networking, and Security\) Common Weakness Enumeration \(CWE\) Top 25 Most Dangerous Software Errors](#)
- [Security Focus](#)

- [NIST National Vulnerability Database](#)

AMI product pricing

AWS Marketplace has multiple pricing models for Amazon Machine Image (AMI) products. With seller private offers, there are options available for multi-year and custom duration contracts.

For more information about multi-year and custom duration contracts, see [Private offers \(p. 44\)](#) and [Flexible payment scheduler \(p. 48\)](#). The following table provides general information about pricing models.

Note

You must be able to provide a W-9 tax form (for US based entities) or a W-8 form (for EU-based entities) as described in [Seller registration process \(p. 6\)](#).

AMI pricing models

The following table provides general information about pricing models for AMI-based products.

| Pricing model | Description |
|-------------------------------|--|
| Free | <p>Customers can run as many instances as Amazon Elastic Compute Cloud (Amazon EC2) supports with no additional software charges incurred.</p> <p>Note Free Trial and Annual pricing can't be combined with Monthly pricing.</p> |
| Bring your own license (BYOL) | <p>AWS Marketplace doesn't charge customers for usage of the software, but customers must supply a license key to activate the product. This key is purchased outside of AWS Marketplace. The entitlement and licensing enforcement, in addition to all pricing and billing, are handled by you.</p> |
| Paid hourly or hourly-annual | <p>Hourly – Software is charged by the hour. Each instance type can be priced differently (but it isn't required to be), and usage is rounded up to the nearest whole hour.</p> <p>Hourly with Free Trial – Customers are limited to running exactly one instance of the software without incurring a charge. You define the duration, between 5 and 30 days. The free trial applies to the most expensive instance type that is running, and any concurrent usage outside the 1 instance is billed at the hourly rate. NOTE: This is a different model than the AWS Free Tier for Amazon EC2 usage whereby customers are given 750 hours of free usage each month.</p> <p>Hourly with Monthly – Both hourly and monthly charges are applied independently. The monthly fee is charged every month regardless of usage, and the hourly fee is applied based on hourly usage only.</p> <p>Hourly with Annual – Customers have the option to purchase a year's worth of usage upfront for one Amazon EC2 instance of one instance type. You set the pricing for each instance type and can offer net savings over the hourly price. Any customer</p> |

| Pricing model | Description |
|---------------|---|
| | <p>usage above the number of annual subscriptions purchased is billed at the hourly rate you set for that instance type.</p> <p>Hourly with Multi-Annual and Custom Duration – This type of offer is only available through seller private offers. Using seller private offers, you specify a custom contract duration, up to 3 years. You can specify upfront payment, or include a flexible payment schedule. You set the pricing for each instance type. If there is a flexible payment schedule in the offer, you also set the invoice dates, payment amounts, and number of instances for each instance type included in the offer. For an active seller private offer with a flexible payment schedule, after the customer launches the specified number of instances, any additional instances launched are charged at the hourly rate specified in the seller private offer. For more information about multi-year and custom duration contracts, see Private offers (p. 44) and the section called “Flexible payment scheduler” (p. 48).</p> <p>Hourly with Free Trial and Annual – This is identical to the Hourly model with an Annual option, except it includes a Free Trial allowing a customer to run one instance of any instance type for free for a set number of days that you determine. Annual subscriptions can be purchased at any time, and they are combined with the Free Trial subscription.</p> <p>Annual with Hourly – Same as the Hourly with Annual pricing model. Customers have the option to purchase a year’s worth of usage upfront for one Amazon EC2 instance of one instance type. You set the pricing for each instance type and can offer net savings over the hourly price, but offering savings isn’t required. Any customer usage above the number of annual subscriptions purchased is billed at the hourly rate you set for that instance type.</p> <p>Multi-Annual and Custom Duration with Hourly – This is only available through Private offers (p. 44). Using seller private offers, you can specify a custom duration contract of up to three years. You can require upfront payment, or you can offer a flexible payment schedule to the customer. You set the pricing for each instance type for the duration of the contract, and the hourly pricing for additional instances launched. If you offer a flexible payment schedule, you also set the invoice dates, payment amounts, and number of instances for each instance type included in the offer. For an active private offer with a flexible payment schedule, after the specified number of instances have been launched, any additional instances the customer launches are charged at the hourly rate specified in the private offer. For more information about multi-year and custom duration contracts, see Private offers (p. 44) and the section called “Flexible payment scheduler” (p. 48).</p> <p>Note Free Trial and Annual pricing can't be combined with Monthly pricing.</p> |

| Pricing model | Description |
|------------------------|---|
| Paid monthly | <p>Monthly – Software is paid for on a fixed monthly basis, regardless of the number of instances the customer runs. Monthly charges are pro-rated at sign-up and upon cancellation. Example: A customer who subscribes for 1 day of the month will be charged for 1/30th of the month.</p> <p>Monthly with Hourly – Both Hourly and Monthly charges are applied independently. The monthly fee is charged every month regardless of usage, and the hourly fee is applied based on hourly usage only.</p> <p>Note Free Trial and Annual pricing can't be combined with Monthly pricing.</p> |
| Paid usage pricing | <p>Software is directly charged for the value you provide along with one of four usage categories: users, data, bandwidth, or hosts. You can define up to 24 dimensions for the product. All charges are still incurred hourly by the customer.</p> <p>All usage is calculated monthly and billed monthly using the same mechanism as existing AWS Marketplace software. Usage pricing is also referred to as AWS Marketplace Metering Service.</p> |
| Contract pricing model | <p>AMI with contract pricing – A Single AMI product or Single AMI with AWS CloudFormation stack that the buyer pays an upfront fee for.</p> |

AWS charges and software charges

Amazon Machine Image (AMI)-based product charges fall into two categories:

- **Infrastructure Pricing Details** – All AMI-based products incur associated AWS infrastructure charges depending on the services and infrastructure used. These rates and fees are defined and controlled by AWS, and can vary between AWS Regions. For more information, see [Amazon EC2 Pricing](#).
- **Software Pricing Details** – For Paid products, the seller defines the charges for using the software.

These two product charge categories are displayed separately on the AWS Marketplace detail pages to help buyers understand the potential cost of using the products.

Free trial for single AMI hourly products

Single AMI hourly products are eligible for the optional Free trial program. In a Free trial, a customer can subscribe to the product and use a single instance for up to 31 days without paying software charges on the product. Applicable AWS infrastructure charges still apply. To offer a single AMI hourly product, define the duration of the trial period and notify the [AWS Marketplace Managed Catalog Operations \(MCO\)](#) team. The trial period can be 5–31 days.

When customers subscribe to a Free trial product, they receive a welcome email message that includes the term of the Free trial, a calculated expiration date, and details on unsubscribing. A reminder email message is sent three days before the expiration date.

If you offer a Free trial product in AWS Marketplace, you agree to the specific refund policies described under **Refund Policy**.

Note

Free trials are available for single AMI products. AMI products that use AWS CloudFormation don't allow free trials.

Contract pricing for AMI products

For Amazon Machine Image (AMI)-based products with contract pricing, AWS Marketplace bills your customers upfront or by the payment schedule that you define, based on the contract between you and your customer. After that point, they're entitled to use those resources.

To set your pricing, choose one or more contract durations that you offer customers. You can enter different prices for each contract duration. Your options are 1-month, 12-months, 24-month, and 36-month durations. For private offers, you can specify a custom duration in months (up to 60 months).

Choose the category that best describes your product's pricing. The pricing category appears to customers on the AWS Marketplace website. You can choose from **Bandwidth** (GB/s, MB/s), **Data** (GB, MB, TB), **Hosts**, **Requests**, **Tiers**, or **Users**. If none of the predefined categories fit your needs, you can choose the more generic **Units** category.

The offer allows for up to 24 dimensions to be added to it. Each dimension requires the following data:

- **Contracts Category** – The contract category is used to measure or meter your product if the product supports consumption-based metering on top of contract pricing. For contract products with no consumption-based pricing, you can choose a category which most closely resembles the category of dimension in the contract or choose **Units** if no values resemble the units for the dimension in the contract
- **Contracts Unit** – The Contract Unit is used along with the Contracts Category for metering if the product supports consumption-based metering. Choose one of the available values for the units that closely matches your dimensions based on the Category selected.
- **Contracts Dimension Allow Multiple Purchases** – This field is used to indicate whether an offer is a tiered pricing offer or a non-tiered offer:

Tiered offer – Allows the buyer to subscribe to only one of the available dimensions in the offer. Dimensions in a tiered offer don't have the concept of quantities. Signing a contract with a specific dimension essentially indicates that the buyer has chosen the specific feature indicated by that dimension.

Non-tiered offer – Allows the customer to procure more than one dimensions as part of the contract and allows them to procure multiple units of each such dimension.

Setting a value of *true* for this field indicates that the offer is a non-tiered offer. Setting a value of *false* for this field indicates that the offer is a tiered offer.

When using the Product Load Form (PLF) to create the contracts for your AMI product, you must define the following fields for your pricing dimensions:

- **Contracts DimensionX API Name** – The name that should appear in the license generated in the buyer's AWS License Manager account. This name is also used as the value for `Name` in `Entitlement` in the `CheckoutLicense` API call.
- **Contracts DimensionX Display Name** – The customer-facing name of the dimension that will be displayed on the product detail and procurement pages of the AWS Marketplace website. Create a name that is be user-friendly. The name's maximum length is 24 characters. After the listing is public, the value of `Name` can't be changed.
- **Contracts DimensionX Description** – The customer-facing description of a dimension that provides additional information about the dimension for the product, such as the capabilities that the specific dimension provides. The maximum length for the description is 70 characters.

- **Contracts DimensionX Quantity** – This is used to calculate proration in cases of agreement amendments to a product. This value of this field should be set to 1 for all contract offers. It should not be edited.
- **Contracts DimensionX 1-Month Rate** – The contract rate to be charged for onemonth of entitlements against this dimension. For non-tiered offers, this rate is charged for each unit of the dimension that is procured. This field supports three decimal places.
- **Contracts DimensionX 12-Month Rate** – The contract rate to be charged for 12 months of entitlements against the dimension. For non-tiered offers, this rate is charged for each unit of the dimension that is procured. This field supports three decimal places.
- **Contracts DimensionX 24-Month Rate** – The contract rate to be charged for 24 months of entitlements against the dimension. For non-tiered offers, this rate is charged for each unit of the dimension that is procured. This field supports three decimal places.
- **Contracts DimensionX 36-Month Rate** – The contract rate to be charged for 36 months of entitlements against the dimension. For non-tiered offers, this rate is charged for each unit of the dimension that is procured. This field supports three decimal places.

Example: Data storage application

| | 1-month price | 12-month price | 24-month price | 36-month price |
|-----------------------|---------------|----------------|----------------|----------------|
| Unencrypted data (GB) | \$1.50/GB | \$16.00/GB | \$30.00/GB | \$60.00/GB |
| Encrypted data (GB) | \$1.55/GB | \$16.60/GB | \$31.20/GB | \$61.20/GB |

Example: Log monitoring product

| | 1-month price | 12-month price | 24-month price | 36-month price |
|--|---------------|----------------|----------------|----------------|
| Basic (10 hosts monitored, 5 containers monitored) | \$100 | \$1000 | \$2000 | \$4000 |
| Standard (20 hosts monitored, 10 containers monitored) | \$200 | \$2000 | \$4000 | \$8000 |
| Pro (40 hosts monitored, 20 containers monitored) | \$400 | \$4000 | \$8000 | \$16,000 |
| Additional hosts monitored per hour | \$10 | \$100 | \$200 | \$400 |
| Additional containers monitored per hour | \$10 | \$100 | \$200 | \$400 |

Note

The prices can be for the following durations: 1 month, 12 months, 24 months, or 36 months. You can choose to offer one or more of these options for your product. The durations must be the same across each dimension.

Example

For example, in a case where you have `ReadOnlyUsers` and `AdminUsers` dimensions, if you offer a yearly price for `ReadOnlyUsers`, you must offer a yearly price for `AdminUsers`, too.

Automatic renewals

When a customer purchases your product through AWS Marketplace using AMI contracts, they can agree to automatic renewal of the contract terms. The customer continues to pay for the entitlements every month or for 1, 2, or 3 years. The customer always has the option to modify the renewal settings. They can cancel the renewal or renew the contract different quantities and durations.

AWS Marketplace Metering Service integration

The AWS Marketplace Metering Service is a pricing and metering feature that sellers can use to directly charge for their software by usage category. There are five usage categories: users, data, bandwidth, hosts, or unit. You can use the Metering Service with Amazon Machine Image (AMI)-based, container-based, and software as a service (SaaS)-based products. For more information, see the [AWS Marketplace Metering Service API Reference](#).

All AMI-based software that uses the Metering Service must meet the following requirements:

- Your software must be launched from AWS Marketplace through an Amazon Machine Image (AMI).
- If you have an existing product in AWS Marketplace, you must submit a new AMI and create a new product to enable this feature.
- All software must be provisioned with an AWS Identity and Access Management (IAM) role. The end customer must add an IAM role to the Amazon Elastic Compute Cloud (Amazon EC2) instance the user is provisioning with the software. The use of an IAM role is optional when you deploy software through AWS Marketplace. It's required when you deploy AWS Marketplace Metering Service software.
- Your software must be able to determine consumption in some way.

Products that use the Metering Service must charge customers by a single usage category, but you can define up to 24 dimensions of a single category. Depending on the category, software can be priced by provisioned resources, concurrent resources, or accumulated resource consumption. All charges are still incurred hourly by the customer. All usage is calculated and billed monthly using the same mechanism as existing AWS Marketplace software.

The AWS Marketplace Metering Service enables several new scenarios. For example, if your software monitors hosts, you can charge for each host monitored. You can have different prices based on the host size, and charge for the number of concurrent hosts monitored each hour. Similarly, if your software allows many users across an organization to sign in, you can charge by the number of users. Each hour, the customer is charged for the total number of provisioned users.

Metering service concepts

The AWS Marketplace Metering Service enables software sellers to modify their software to send metering records to an endpoint to capture usage. Sellers can select a usage category and define up

to 24 dimensions of that one category. These dimensions are metered once per hour, aggregated, and charged against a price plan defined by the seller. As a seller, you must determine which dimension you want to use. After the AMI is published, you will not be able to change it. Important service concepts include the following:

- **Usage Category** – Any software product priced through the use of the Metering Service is categorized according to one usage category, which determines the appropriate way to charge customers. Usage categories include but aren't limited to:
 - **Users** – A defined set of permissions associated with a single identifier. This category is appropriate for software in which a customer's users connect to the software directly (for example, for customer-relationship management or business intelligence reporting).
 - **Hosts** – Any server, node, instance, endpoint, or other part of a computing system. This category is appropriate for software that monitors or scans many customer-owned instances (for example, performance or security monitoring).
 - **Data** – Storage or information, measured in MB, GB, or TB. This category is appropriate for software that manages stored data or processes data in batches.
 - **Bandwidth** – Measured in Mbps or Gbps. This category is appropriate for software that allows customers to specify an amount of bandwidth to provision.
 - **Unit** – Unit of measurement; see the examples described next.
- **Usage Unit** – A software product's specific usage unit corresponds to the selected usage category. This usage unit describes the unit your software will charge on. Examples include:
 - **NodesHrs** (corresponding to the Hosts category)
 - **UserHrs** (corresponding to the User category)
 - **GBStored** (corresponding to the Data category)
- **Consumption** – Software products priced through the use of the Metering Service charge for consumption in one of three ways:
 - **Provisioned** – The software allows customers to configure a specific amount of resources for use (for example, number of users or a fixed amount of bandwidth). Each hour, customers pay for what they have provisioned.
 - **Concurrent** – The software allows any number of distinct hosts or users to connect to the software. Each hour, customers pay based on the number of hosts or users who accessed the software.
 - **Accumulated** – The software allows customers to use any amount of data, either processed or stored. Each hour, customers pay for the aggregated amount.
- **Pricing** – Software products priced through the use of the Metering Service must specify either a single price or define up to 24 dimensions, each with their own price. Details about the pricing options include:
 - **Single dimension** – This is the simplest pricing option. Customers pay a single price per resource unit per hour, regardless of size or volume (for example, \$0.014 per user per hour, or \$0.070 per host per hour).
 - **Multiple dimensions** – This pricing option is appropriate when the selected usage category varies along multiple axes. For example, for host monitoring, a different price could be set depending on the size of the host. Or, for user-based pricing, a different price could be set based on the type of user (for example, admin, power user, and read-only user).
- **Metering** – All usage is recorded as a metering event, once each hour. Your software must be configured to send the appropriate dimension and usage amount to the AWS Marketplace Metering Service.
 - **Allocations** – Optionally, you may distribute the usage into allocations by properties that you track. These allocations are represented as tags to the buyer. The tags allow the buyer to view their costs split into usage by tag. For example, if you charge by the user, and users have a "Department" property, you could create usage allocations with tags that have a key of "Department", and one allocation per value. This approach doesn't change the price, dimensions, or the total usage that you report. However, it allows your customer to view their costs by categories appropriate to your product.

Pricing your software

When pricing your software with the AWS Marketplace Metering Service, you must first decide on a usage category and how it will be consumed. The service supports six distinct pricing scenarios. You must select only one of these for your product:

- Provisioned user (per hour)
- Concurrent user (per hour)
- Provisioned host (per hour)
- Concurrent host (per hour)
- Provisioned bandwidth (per hour)
- Accumulated data (per hour)

Next, you must decide how to price the selected usage category:

- Single price
- Multiple dimensions (up to 24)

[Adding your product to AWS Marketplace \(p. 98\)](#) describes how to provide a customer-friendly description of your dimension and pricing.

Example: Provisioned bandwidth with nonlinear pricing

Imagine you offer network appliance software. You choose to bill by provisioned bandwidth. For your usage category, select **Bandwidth**. In addition to charging by bandwidth, you want to charge a different price as buyers scale up. You can define multiple dimensions within the bandwidth category. You can define a distinct price for 25 Mbps, 100 Mbps, and 1 Gbps.

Example: Concurrent hosts with multiple dimensions

Imagine you offer software that monitors other Amazon EC2 instances. You choose to bill by the number of hosts that are being monitored. For your usage category, select **Host**. In addition to charging by host, you want to charge for the extra value for monitoring larger hosts. You can use multiple dimensions within the host category. You can define a distinct price for micro, small, medium, large, x-large, 2XL, 4XL, and 8XL instances. Your software is responsible for mapping each particular host to one of your defined dimensions. Your software is responsible for sending a separate metering record for each dimension of your usage category if applicable.

Adding your product to AWS Marketplace

To take advantage of the Metering Service, you must create a new product for AWS Marketplace to list. If your product is already on the AWS Marketplace, you will need to decide whether the new AWS Marketplace Metering Service product will be made available in addition to your current product, or if it will replace your current product as the only version available to new users. If you choose replacement, the existing product will be removed from the AWS Marketplace so that it is no longer available for new buyers. Existing customers will continue to have access to their old product and instances, but they can migrate to the new product at their convenience. The new product must meter usage to the AWS Marketplace Metering Service, as described in [Modifying your software to use the Metering Service \(p. 100\)](#).

After you have your AMI, follow the standard process to share and scan your AMI using the self-service tool. In addition to using the template available on the management portal, fill out the product load form and upload it to start the ingestion process.

Use the following definitions to complete the fields of the Product Load Form for the AWS Marketplace Metering Service. On the Product Load Form, these fields are labeled as **Flexible Consumption Pricing (FCP)** to differentiate them from hourly and monthly priced products.

- **Title** – If you already have a product on AWS Marketplace and you're adding the same product with the AWS Marketplace Metering Service, include the FCP category and dimension in parentheses to differentiate them (for example, "PRODUCT TITLE (Data)").
- **Pricing Model** – From the dropdown list, choose **Usage**.
- **FCP Category** – The category in which customers are charged for paid products with a **Usage** pricing component. From the dropdown list, choose **Users**, **Hosts**, **Data**, or **Bandwidth**.
- **FCP Unit** – The unit of measurement on which customers are charged for paid products with a **Usage** pricing component. Options will appear in the dropdown list based on the FCP category you selected. The following table lists the valid units for each category.

| Category | Valid units |
|-----------|-------------|
| Users | UserHrs |
| Hosts | HostHrs |
| Data | MB, GB, TB |
| Bandwidth | Mbps, Gbps |

- **FCP Dimension Name** – The name used when sending metering records by calling the `MeterUsage` operation. It is visible in billing reports. However, because it isn't external-facing, the name doesn't need to be user-friendly. The name can be no more than 15 characters and can only include alphanumeric and underscore characters. After you set the name and make the product public, you can't change it. Changing the name requires a new AMI.
- **FCP Dimension Description** – The customer-facing statement that describes the dimension for the product. The description (can be no more than 70 characters and should be user-friendly. Examples of descriptions include: Administrators per hour and Per Mbps bandwidth provisioned. After the product is published, you can't change this description.
- **FCP Rate** – The software charge per unit for this product. This field supports three decimal places.

Notes:

- You don't need to fill out hourly and annual pricing fields.
- Free trial and annual pricing aren't compatible.
- Products that use multiple AMIs and the Clusters and AWS Resources feature can't use the AWS Marketplace Metering Service.
- Price, instance type, or AWS Region change will follow the same process as other AWS Marketplace products.
- Products with the AWS Marketplace Metering Service can't be converted to other pricing models such as hourly, monthly, or Bring Your Own License (BYOL).
- AWS Marketplace recommends adding IAM policy information in your usage instructions or document.
- You can include up to 24 FCP dimensions in total. Once created and published, you can't modify existing dimensions, but you can add new ones (up to the limit of 24).

If you have questions, contact the [AWS Marketplace Seller Operations](#) team.

Modifying your software to use the Metering Service

You will need to modify your software to record customer usage, send hourly usage reports to the Metering Service, and handle new failure modes. The software operates independently of pricing, but the software will need to know about the usage category, how it's consumed, and any dimensions.

Measuring consumption

Your software must determine how much of the selected usage category and which dimensions the customer has consumed. This value will be sent, once each hour, to the Metering Service. In all cases, it's assumed that your software has the ability to measure, record, and read consumption of resources for the purpose of sending it on an hourly basis to the Metering Service.

For provisioned consumption, this will typically be read from the software configuration as a sampled value, but might also be a maximum configured value, recorded each hour. For concurrent consumption, this might be either a periodic sample or a maximum value recorded each hour. For accumulated consumption, this will be a value that is accumulated each hour.

For pricing on multiple dimensions, multiple values must be measured and sent to the Metering Service, one per dimension. This requires your software to be programmed or configured with the known set of dimensions when you provide the AMI. The set of dimensions can't change after a product is created.

For each pricing scenario, the following table describes recommended ways for measuring consumption each hour.

| Scenario | How to measure |
|------------------|---|
| Provisioned user | Current number of provisioned users (sampled). -OR- Maximum number of provisioned users (seen that hour). |
| Concurrent user | Current number of concurrent users (sampled). -OR- Maximum number of concurrent users (seen that hour). -OR- Total number of distinct users (seen that hour). |
| Provisioned host | Current number of provisioned hosts (sampled). -OR- Maximum number of provisioned hosts (seen that hour). |
| Concurrent host | Current number of concurrent hosts (sampled). -OR- Maximum number of concurrent hosts (seen that hour). |

| Scenario | How to measure |
|-----------------------|--|
| | -OR- Total number of distinct hosts (seen that hour). |
| Provisioned bandwidth | Current provisioned bandwidth setting (sampled). -OR- Maximum provisioned bandwidth (seen that hour). |
| Accumulated data | Current GB of data stored (sampled). -OR- Maximum GB of data stored (seen that hour). -OR- Total GB of data added or processed that hour. -OR- Total GB of data processed that hour. |

Call AWS Marketplace Metering Service

Your software must call the Metering Service hourly and record the consumption value for that hour.

When your software starts, it should record the minute-of-the-hour at which it started. This is referred to as the *start-minute*. Every hour on the start-minute, your software must retrieve the consumption value for that hour and call the Metering Service. For information about how to obtain this value, see the [Measuring consumption \(p. 100\)](#) section.

To wake up each hour at the start-minute, your software must use one of the following approaches:

- A thread within your software.
- A daemon process that starts up with the instance or software.
- A cron job that is configured during application startup.

Note

Your software must call the AWS Marketplace Metering Service using the IAM role configured on the customer's instance and specify the consumption dimension and amount.

Your software can use the AWS SDK to call the AWS Marketplace Metering Service, similar to the following example implementation:

1. Use the instance profile to create a service client. This requires the role configured for the EC2 instance. The role credentials are refreshed by the SDK automatically.
2. Each hour, read your software configuration and state to determine consumption values for that hour. This might include collecting a value-per-dimension.
3. Call the `meterUsage` method on the SDK client with the following parameters (call additionally for each dimension that has usage):
 - `timestamp` – Timestamp of the hour being recorded (in UTC).
 - `productCode` – Product code assigned to the software.

- `dimension` – Dimension (or dimensions) assigned to the software.
- `quantity` – Consumption value for the hour.
- `allocations` – (Optional) You may provide allocations for the usage across properties that you track. These allocations must add up to the total consumption in the record. To the buyer, these display as potential cost allocation tags in their billing tools (such as the AWS Billing and Cost Management console). The buyer must activate the tags in their account in order to track their cost using these tags.

In addition, your software must call an in-Region AWS Marketplace Metering Service endpoint. Your product must have a correct Regional endpoint set up, so `us-east-1` sends records to a `us-east-1` endpoint, and `us-west-2` sends records to a `us-west-2` endpoint. Making in-Region calls provides buyers with a more stable experience and prevents situations in which an unrelated Region's availability could impact software running in another Region.

When you send metering records to the service, you must connect to the AWS Marketplace Metering Service in your Region. Use the `getCurrentRegion()` helper method to determine the Region in which the EC2 instance is running, and then pass this Region information to the `MeteringServiceClient` constructor. If you don't specify an AWS Region in the SDK constructor, the default `us-east-1` Region is used. If your application attempts to make cross-Region calls to the service, the calls are rejected. For more information, see [Determining an Application's Current Region](#) and [getCurrentRegion\(\)](#).

Failure handling

Your product must send metering records to the service, a public internet endpoint, so that usage can be captured and billed. Because it's possible for a customer to modify network settings in a way that prevents your metering records from being delivered, your product should account for this by choosing a failure mode.

Note

Some metering failures may be transient issues in connecting to the AWS Marketplace Metering Service. AWS Marketplace strongly recommends implementing retries for up to 30 minutes, with exponential back off, to avoid short-term outages or network issues.

Typically, software can fail open (provide a warning message but maintain full functionality) or fail closed (disable all functionality in the application until a connection has been reestablished). You can choose to fail open, closed, or something specific to your application. We strongly recommend that you refrain from failing closed after less than two hours of metering failures.

As an example of failing partially open, you could continue to allow access to the software but not allow the buyer to modify the software settings. Or, a buyer could still access the software but would not be able to create additional users. Your software is responsible for defining and enforcing this failure mode. Your software's failure mode must be included when your AMI is submitted, and it can't be changed later.

Limitations

Keep these limitations in mind when designing and submitting your Metering Service-enabled software:

- **IAM role and internet gateway requirements for your customers** – Your customers must have an internet gateway and must launch your software with an IAM role with specific permissions. For more information, see [AWS Marketplace metering and entitlement API permissions \(p. 358\)](#). Your software can't connect to the Metering Service if these two conditions aren't met.
- **Inability to add new or change usage category to existing Metering Service product** – When customers subscribe to your software product, they're agreeing to terms and conditions. Changing the usage categories in products with the Metering Service requires a new product and a new subscription.
- **Inability to change dimensions to existing Metering Service product** – When customers subscribe to your software product, they're agreeing to terms and conditions. Changing the dimensions in products

with the Metering Service requires a new product and a new subscription. You *can* add new dimensions to existing products, up to the limit of 24.

- **Lack of free trial and annual subscriptions** – Metering Service products don't support free trials and annual subscriptions at launch.
- **Multi-instance or cluster-based deployment considerations** – Some software is deployed as part of a multi-instance deployment. When you design your software, consider how and where consumption is measured and where metering records are emitted.

Vendor-metered tagging (Optional)

Vendor-metered tagging helps Independent Software Vendors (ISVs) give the buyer more granular insight into their software usage and can help them perform cost allocation.

There are many ways to tag a buyer's software usage. One way is to first ask your buyers what they want to see in their cost allocation. Then you can split the usage across properties that you track for the buyer's account. Examples of properties include `Account ID`, `Business Unit`, `Cost Centers`, and other relevant metadata for your product. These properties are exposed to the buyer as tags. Using tags, buyers can view their costs split into usage by the tag values in their AWS Billing Console (<https://console.aws.amazon.com/billing/>). Vendor-metered tagging doesn't change the price, dimensions, or the total usage that you report. It allows your customer to view their costs by categories appropriate to your product.

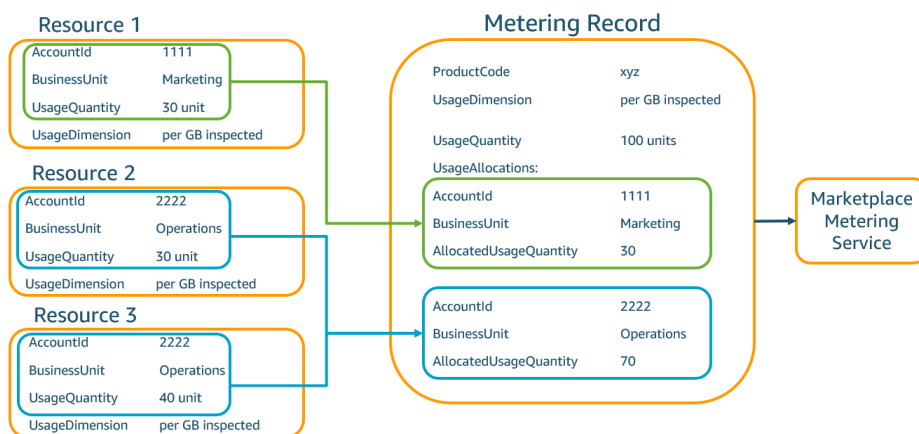
In a common use case, a buyer subscribes to your product with one AWS account. The buyer also has numerous user accounts associated with the same product subscription. You can create usage allocations with tags that have a key of `Account ID`, and then allocate usage to each user account. In this case, buyers can activate the `Account ID` tag in their Billing and Cost Management console and analyze individual user account usage.

Seller experience

Sellers can aggregate the metering records for resources with the same set of tags instead of aggregating usage for all resources. For example, sellers can construct the metering record that includes different buckets of `UsageAllocations`. Each bucket represents `UsageQuantity` for a set of tags, such as `AccountId` and `BusinessUnit`.

In the following diagram, **Resource 1** has a unique set of `AccountId` and `BusinessUnit` tags, and appears in the **Metering Record** as a single entry.

Resource 2 and **Resource 3** both have the same `AccountId` tag, 2222, and the same `BusinessUnit` tag, `Operations`. As a result, they're combined into a single `UsageAllocations` entry in the **Metering Record**.



Buyer experience

The following table shows an example of the buyer experience after a buyer activates the `AccountId` and `BusinessUnit` vendor tags.

In this example, the buyer can see allocated usage in their **Cost Usage Report**. The vendor-metered tags use the prefix `aws:marketplace:isv`. Buyers can activate them in the Billing and Cost Management, under **Cost Allocation Tags, AWS-generated cost allocation tags**.

The first and last rows of the **Cost Usage Report** are relevant to what the Seller sends to the Metering Service (as shown in the [Seller experience \(p. 147\)](#) example).

Cost Usage Report (Simplified)

| ProductCode | Buyer | UsageDimension | UsageQuantity | aws:marketplace:isv:BusinessUnit | aws:marketplace:isv:BusinessUnit |
|-------------|--------------|-----------------------------|---------------|----------------------------------|----------------------------------|
| xyz | 111122223333 | Network: per (GB) inspected | 70 | 2222 | Operations |
| xyz | 111122223333 | Network: per (GB) inspected | 30 | 3333 | Finance |
| xyz | 111122223333 | Network: per (GB) inspected | 20 | 4444 | IT |
| xyz | 111122223333 | Network: per (GB) inspected | 20 | 5555 | Marketing |
| xyz | 111122223333 | Network: per (GB) inspected | 30 | 1111 | Marketing |

For a code example, see [MeterUsage with usage allocation tagging \(Optional\) \(p. 104\)](#)

Code example

The following code example is provided to help you integrate your AMI product with the AWS Marketplace APIs required for publishing and maintaining your product.

MeterUsage with usage allocation tagging (Optional)

The following code example is relevant for AMI products with consumption pricing models. The Python example sends a metering record with appropriate usage allocation tags to AWS Marketplace to charge your customers for pay-as-you-go fees.

```
# NOTE: Your application will need to aggregate usage for the
#       customer for the hour and set the quantity as seen below.
#       AWS Marketplace can only accept records for up to an hour in the past.
#
# productCode is supplied after the AWS Marketplace Ops team has
# published the product to limited

# Import AWS Python SDK
import boto3
import time

usageRecord = [
    {
```

```
        "AllocatedUsageQuantity": 2,
        "Tags":
          [
            { "Key": "BusinessUnit", "Value": "IT" },
            { "Key": "AccountId", "Value": "123456789" },
          ]
      },
      {
        "AllocatedUsageQuantity": 1,
        "Tags":
          [
            { "Key": "BusinessUnit", "Value": "Finance" },
            { "Key": "AccountId", "Value": "987654321" },
          ]
      }
    ]
  ]

marketplaceClient = boto3.client("meteringmarketplace")

response = marketplaceClient.meter_usage(
    ProductCode="testProduct",
    Timestamp=int(time.time()),
    UsageDimension="Dimension1",
    UsageQuantity=3,
    DryRun=False,
    UsageAllocations=usageRecord
)
```

For more information about `MeterUsage`, see [MeterUsage](#) in the *AWS Marketplace Metering Service API Reference*.

Example response

```
{ "MeteringRecordId": "string" }
```

AWS License Manager integration

For Amazon Machine Image (AMI)-based products with contract pricing, you use AWS License Manager to associate licenses with your product.

AWS License Manager is a license management tool that enables your application to track and update licenses (also known as entitlements) that have been purchased by a customer. This section provides information about how to integrate your product with AWS License Manager. After the integration is complete, you can publish your product listing on AWS Marketplace.

For more information about AWS License Manager, see the [AWS License Manager User Guide](#) and the [AWS License Manager](#) section of the *AWS CLI Command Reference*.

Note

- Customers can't launch new instances of the AMI after the contract expiry period. However, during the contract duration, they can launch any number of instances. These licenses are not node-locked or tied to particular instances.
- **Private Offer Creation**– Sellers can generate private offers for the products using the Private offer creation tool in the AWS Marketplace Management Portal.

- **Reporting** – You can set up data feeds by setting up an Amazon S3 bucket in the **Report** section in the AWS Marketplace Management Portal. For more information, see [Seller reports and data feeds](#) (p. 273).

License models

AWS Marketplace integration with AWS License Manager supports two license models:

- [Configurable license model](#) (p. 106)
- [Tiered license model](#) (p. 108)

Configurable license model

The configurable license model (also known as the quantifiable license model) entitles a buyer to a specific quantity of resources after a buyer has procured a license.

You set a pricing dimension and a per unit price. Then, buyer can choose the quantity of the resources that they want to purchase.

Example of pricing dimension and per unit price

You can set a pricing dimension (such as data backup) and per unit price (such as \$30/unit)

The buyer can choose to purchase 5, 10, or 20 units.

Your product tracks and meters usage to measure the quantity of resources consumed.

With the configuration model, the entitlements are counted in one of two ways:

- [Drawdown licenses](#) (p. 106)
- [Floating licenses](#) (p. 107)

Drawdown licenses

The license is drawn from the pool of allowed amount of licenses upon use. That entitlement is checked out permanently and can't be returned to the license pool.

Example of processing a limited amount of data

A user is entitled to process 500GB of data. As they continue to process data, the quantity is drawn from the pool of 500GB until all 500GB licenses are consumed.

For drawdown licenses, you can use the `CheckoutLicense` API operation to check out license units that are consumed.

Example of backup to S3 for a number of units/year

You have a storage product that allows backup to Amazon Simple Storage Service (Amazon S3) for up to 1024 units for data for one year. Your application can be launched by using multiple Amazon EC2 instances. Your application has a mechanism to track and aggregate data. Your software calls the `CheckoutLicense` API operation with the Product ID upon every backup or at fixed intervals to update the consumed quantities.

In this example, your software calls `CheckoutLicense` to check out 10 units of data. When the total capacity reaches the backup limit that the customer has purchased, the API call fails.

Request

```
linux-machine ~]$ aws license-manager checkout-license\  
--product-sku "2205b290-19e6-4c76-9eea-377d6bf71a47" \  
--checkout-type "PERPETUAL" \  
--key-fingerprint "aws:294406891311:AWS/Marketplace:issuer-fingerprint" \  
--entitlements "Name=DataConsumption, Value=10, Unit=Count" \  
--client-token "AKIAIOSFODNN7EXAMPLE"
```

Response

```
{  
  "CheckoutType": "PERPETUAL",  
  "EntitlementsAllowed": [  
    {  
      "Name": "DataConsumption",  
      "Count": 10,  
      "Units": "Count",  
      "Value": "Enabled"  
    }  
  ],  
  "Expiration": "2021-04-22T19:02:36",  
  "IssuedAt": "2021-04-22T18:02:36",  
  "LicenseArn": "arn:aws:license-manager::294406891311:license:l-16bf01b...",  
  "LicenseConsumptionToken": "AKIAIOSFODNN7EXAMPLE"  
}
```

Floating licenses

The license is returned to the pool of the allowed amount of licenses after use.

Example of number of users from a fixed upper limit

A user is entitled to 500 simultaneous users on the application. As users log in and log out, the users are drawn and returned to the pool of 500 users. However, the application can't draw more than 500 users from the pool because 500 simultaneous users is the fixed upper limit.

For floating licenses, you can use the `CheckInLicense` API operation to return the license units to the entitlement pool.

Example of number of concurrent users for one year

Your product is priced based on number of concurrent users. The customer purchases a license for 10 users for one year. The customer launches the software by providing AWS Identity and Access Management (IAM) permissions. When a user logs in, your application calls the `CheckoutLicense` API operation to reduce the quantity by 1. When the user logs out, the application returns that license to the pool by calling the `CheckInLicense` API operation. If you don't call `CheckInLicense`, the license unit will be automatically checked in after 1 hour.

Note

In the following Request, the `key-fingerprint` isn't a placeholder value but the actual value of the fingerprint with which all licenses will be published.

Request

```
linux-machine ~]$ aws license-manager checkout-license\  
--product-sku "2205b290-19e6-4c76-9eea-377d6bf71a47" \  
--checkout-type "PROVISIONAL" \  
--key-fingerprint "aws:294406891311:AWS/Marketplace:issuer-fingerprint" \  
--entitlements "Name=ReadOnlyUSers, Value=10, Unit=Count" \  
--client-token "AKIAIOSFODNN7EXAMPLE"
```

Response

```
{
  "CheckoutType": "PROVISIONAL",
  "EntitlementsAllowed": [
    {
      "Name": "ReadOnlyUsers",
      "Count": 10,
      "Units": "Count",
      "Value": "Enabled"
    }
  ],
  "Expiration": "2021-04-22T19:02:36",
  "IssuedAt": "2021-04-22T18:02:36",
  "LicenseArn": "arn:aws:license-manager::294406891311:license:l-16bf01b...",
  "LicenseConsumptionToken": "AKIAIOSFODNN7EXAMPLE"
}
```

Tiered license model

The tiered license model entitles a buyer to a specific level, or tier, of application features after a buyer has procured a license.

You create tiers for your product, such as Basic, Intermediate, and Premium. The buyer then selects one of the predefined tiers.

The application doesn't need to track or meter usage of the application.

With the tiered license model, the entitlements aren't counted but instead signify a tier of service that was procured by the customer.

If you want to offer bundled features together, we recommend using the tiered license model.

Example of Basic, Intermediate, and Premium tiers

A customer can sign a contract for one of three possible tiers of the software: Basic, Intermediate, or Premium. Each of these tiers has its own pricing. Your software can identify the tier that the customer has signed up for by invoking the `CheckoutLicense` API operation and specifying all possible tiers in the request.

The response of the request contains the entitlement corresponding to the tier the customer has procured. Based on this information, the software can provision the appropriate customer experience.

Request

```
linux-machine ~]$ aws license-manager checkout-license\
--product-sku "2205b290-19e6-4c76-9eea-377d6bf71a47" \
--checkout-type "PROVISIONAL" \
--key-fingerprint "aws:294406891311:AWS/Marketplace:issuer-fingerprint" \
--entitlements "Name=BasicTier, Unit=None" "Name=IntermediateTier, Unit=None" \
"Name=PremiumTier, Unit=None"
```

Response

```
{
  "CheckoutType": "PROVISIONAL",
  "EntitlementsAllowed": [
    {
      "Name": "IntermediateTier",
      "Units": "None"
    }
  ]
}
```

```
},  
  "Expiration": "2021-04-22T19:02:36",  
  "IssuedAt": "2021-04-22T18:02:36",  
  "LicenseArn": "arn:aws:license-manager::294406891311:license:l-16bf01b...",  
  "LicenseConsumptionToken": "AKIAIOSFODNN7EXAMPLE"  
}
```

Integration workflow

The following steps show the workflow for integrating your AMI product with AWS License Manager:

1. Seller creates a product with AWS License Manager integration.
2. Seller lists the product on AWS Marketplace.
3. Buyer finds the product on AWS Marketplace and purchases it.
4. A license is sent to the buyer in their AWS account.
5. Buyer uses the software by launching the Amazon Elastic Compute Cloud (Amazon EC2) instance, Amazon Elastic Container Service (Amazon ECS) task, or Amazon Elastic Kubernetes Service (Amazon EKS) pod software. The customer deploys by using an IAM role.
6. Software reads the license in the buyer's AWS License Manager account, discovers the entitlements purchased, and provisions the features accordingly.

Note

License Manager doesn't do any tracking or updates; this is done by the seller's application.

License Manager integration prerequisites

Before publishing the product, you must do the following:

1. Create a new AMI product in the AWS Marketplace Management Portal, and make a note of its product code.
2. Fill out the Product Load Form (PLF) with the necessary price information, and return it to us for processing.
3. Use an IAM role for the task or pod running your application with the IAM permissions necessary to call `CheckoutLicense`, `ExtendLicenseConsumption`, and `CheckInLicense`.

The required IAM permissions are detailed in the following IAM policy.

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "VisualEditor0",  
      "Effect": "Allow",  
      "Action": [  
        "license-manager:CheckoutLicense",  
        "license-manager:GetLicense",  
        "license-manager:CheckInLicense",  
        "license-manager:ExtendLicenseConsumption",  
        "license-manager:ListReceivedLicenses"  
      ],  
      "Resource": "*"   
    }  
  ]  
}
```

4. Make a test call to the `RegisterUsage` API operation with a record for all of the pricing dimensions you define.

Integrating an AMI product with License Manager

To integrate your AMI-based product with License Manager

1. Set IAM permissions to call License Manager. For more information, see [License Manager integration prerequisites](#) (p. 109).
2. Download the AWS SDK.

Don't configure AWS credentials within your software. AWS credentials for the buyer are automatically obtained at runtime when your AMI is running within an Amazon EC2 instance, Amazon ECS task, or Amazon EKS pod.

3. Add license checks to your product.

Your product can call the `CheckoutLicense` API operation wherever the license check should be performed. For checking the license, your product has to know:

1. The trusted issuer of the license (AWS Marketplace)
2. The application's Product SKU (Product ID)
3. The entitlement to check for this application

The API calls vary based on what kind of pricing licenses you set up.

4. Publish your product listing on AWS Marketplace.

License Manager API calls

To manage the licenses stored in the customer's License Manager account, your software can use the following API calls:

- `GetLicense` – Gets the status of a purchased license on whether the license is expired or about to be expired to send a notification to the customer.
- `CheckoutLicense` – Discovers licenses that the user has purchased. You can also use it to update the license quantity when the user has consumed some quantity of licenses. With `CheckoutLicense`, you can keep checking out the quantities of the licenses used by the customer. When the customer exhausts all the licenses, this call returns an error. For information about the suggested cadence to run `CheckoutLicense`, see [the section called "License renewals and upgrades"](#) (p. 110).
- `ExtendLicenseConsumption` – In case of floating dimensions, when the software checks out a license, it will return the license to the pool automatically after 60 minutes. If you want to extend the time the license remains checked out, your software can call `ExtendLicenseConsumption` to extend the license for another 60 minutes.
- `CheckInLicense` – In case of floating dimensions, when you want to return the license to the entitlement pool, use `CheckInLicense`.
- `ListReceivedLicenses` – Lists licenses purchased by the buyer.

License renewals and upgrades

Customers can renew or upgrade their licenses on the AWS Marketplace Management Portal. After they make an additional purchase, AWS Marketplace generates a new version of the license that reflects the new entitlements. Your software reads the new entitlements using the same API calls. You don't have to do anything different in terms of License Manager Integration to handle renewals and upgrades.

Due to license renewals, upgrades, cancellations, and so on, we recommend that your product performs the `CheckoutLicense` API call at a regular cadence while the product is in use. By using the

CheckoutLicense API operation at a regular cadence, the product can detect changes in entitlements such as upgrades and expiry.

We recommend that you perform the CheckoutLicense API call every 15 minutes.

Amazon SNS notifications for AMI products

To receive notifications, you subscribe to the AWS Marketplace Amazon Simple Notification Service (Amazon SNS) topics provided to you during product creation. The topics provide notifications about changes to customers' subscriptions for your products. For example, you can know when customers accept a private offer.

The following Amazon SNS topic is available for AMI products:

- [Amazon SNS topic: aws-mp-subscription-notification \(p. 111\)](#) – This topic notifies you when a buyer subscribes or unsubscribes to a product. This notification is available for hourly pricing models, including hourly and hourly with annual.

Amazon SNS topic: aws-mp-subscription-notification

Each message in the aws-mp-subscription-notification topic has the following format.

```
{
  "action": "<action-name>",
  "customer-identifier": " X01EXAMPLE",
  "product-code": "n0123EXAMPLEXXXXXXXXXXXX",
  "offer-identifier": "offer-abcexample123"
}
```

The `<action-name>` will vary depending on the notification. Possible actions are:

- subscribe-success
- subscribe-fail
- unsubscribe-pending
- unsubscribe-success

The `offer-identifier` only appears in the notification if the offer is a *private offer*.

Subscribing an Amazon SQS queue to the Amazon SNS topic

We recommend subscribing an Amazon SQS queue to the provided SNS topics. For detailed instructions on creating an SQS queue and subscribing the queue to a topic, see [Subscribing an Amazon SQS queue to an Amazon SNS topic](#) in the *Amazon Simple Notification Service Developer Guide*.

Note

You can only subscribe to AWS Marketplace SNS topics from the AWS account used to sell the products. However, you can forward the messages to a different account. For more information, see [Sending Amazon SNS messages to an Amazon SQS queue in a different account](#) in the *Amazon Simple Notification Service Developer Guide*.

Polling the SQS queue for notifications

After you subscribe your SQS queue to an SNS topic, the messages are stored in SQS. You must define a service that continually polls the queue, looks for messages, and handles them accordingly.

AMI product checklist

Before submitting your AMI product request to AWS Marketplace, review this checklist. Validating this information will help to make sure your submission goes through the publication process smoothly.

Product usage:

- Your AMI must be production-ready.
- Your AMI can't restrict product usage by time or any other measurements.
- Your AMI must be compatible with the 1-Click fulfillment experience.
- Everything required to use the product is in the software, including client applications.
- The default user uses a randomized password, or creating the initial user requires verification that the buyer is authorized to use the instance using a value unique to the instance such as instance ID.

For free or paid products:

- No additional license is required to use the product.
- The buyer doesn't have to provide personally identifiable information (for example, their email address) to use the product.

AMI preparation:

- Uses HVM virtualization and 64-bit architecture
- Doesn't contain any known vulnerabilities, malware or viruses
- Buyers have OS-level administration access to the AMI
- Run your AMI through AMI Self-Service Scanning

For Windows AMIs:

- Uses the most recent version of [EC2ConfigService](#)
- `Ec2SetPassword`, `Ec2WindowsActivate`, and `Ec2HandleUserData` are enabled in your AMI
- No guest accounts or remote desktop users are present

For Linux AMIs:

- Root login is locked or disabled
- No authorized keys, default passwords, or other credentials are included

Product Load Form or Product tab

- All required fields are completed
- All values are within specified character limits
- All URLs load without error
- The product image is at least 110 pixels wide and between a 1:1 and 2:1 ratio

- Pricing is specified for all enabled instance types (for hourly, hourly-based monthly pricing, and hourly-based annual pricing models)
- Monthly pricing is specified (for hourly-based monthly and monthly pricing models)

AMI security policies

AWS Marketplace maintains the following policies for all Amazon Machine Image (AMI) products and offerings in AWS Marketplace. The policies promote a safe, secure, and trustworthy platform for our customers.

All products and their related metadata are reviewed when they're submitted to ensure that they meet or exceed current AWS Marketplace policies. These policies are reviewed and adjusted to meet evolving security guidelines. AWS Marketplace continuously scans your products to verify that they meet changes to the security guidelines. If products fall out of compliance, AWS Marketplace will contact you to update your AMI product to meet new standards. Likewise, if a newly discovered vulnerability is found to affect the AMI, we will ask you to provide an updated AMI with the relevant updates in place. You must use the [self-service AMI scanning tool](#) before submitting your AMI. This tool helps ensure that the AMI meets AWS Marketplace policies.

Security policies

All AMIs must adhere to the following security policies:

- AMIs must not contain any known vulnerabilities, malware, or viruses as detected by the [self-service AMI scanning tool](#) or AWS Security.
- AMIs must use currently supported operating systems and other software packages. Any version of an AMI with an End-of-Life (EoL) operating system or other software packages will be delisted from the AWS Marketplace. You can build a new AMI with updated packages and publish it as a new version to AWS Marketplace.
- All instance authentication must use key pair access, not password-based authentication, even if the password is generated, reset, or defined by the user at launch. AMIs must not contain passwords, authentication keys, key pairs, security keys, or other credentials for any reason.
- AMIs must not request or use access or secret keys from users to access AWS resources. If your AMI application requires access to the user account, it must be achieved through an AWS Identity and Access Management (IAM) role instantiated through AWS CloudFormation, which creates the instance and associates the appropriate role. When single-AMI launch is enabled for products with an AWS CloudFormation delivery method, corresponding usage instructions must include clear guidance for creating minimally privileged IAM roles. For more information, see [the section called "AMI-based Delivery Using CloudFormation" \(p. 74\)](#).
- Linux-based AMIs must not allow SSH password authentication. Disable password authentication via your `sshd_config` file by setting `PasswordAuthentication` to `NO`.

Access policies

There are three categories of access policies: general, Linux-specific, and Windows-specific policies.

General access policies

All AMIs must adhere to the following general access policies:

- AMIs must allow operating system (OS)-level administration capabilities to allow for compliance requirements, vulnerability updates, and log file access. Linux-based AMIs use SSH, and Windows-based AMIs use RDP.

- AMIs must not contain authorized passwords or authorized keys.
- AMIs must not use fixed passwords for administrative access. AMIs must use a randomized password instead. An alternative implementation is to retrieve the instance metadata and use the `instance_id` as the password. The administrator must be prompted for this randomized password before being permitted to set or change their own credentials. For information about retrieving instance metadata, see [Instance Metadata and User Data](#) in the *Amazon EC2 User Guide for Linux Instances*.
- You must not have access to the customer's running instances. The customer has to explicitly enable any outside access, and any accessibility built into the AMI must be off by default.

Linux-specific access policies

Linux-based AMIs must adhere to the following access policies, as well as the general access policies:

- Linux-based AMIs must [disable password-based remote logins](#) for root access and allow only sudo access through a user account, not root. Users must use sudo access through a user account and can't use root access. Sudo access allows the administrator to control which users are allowed to perform root functions. It also logs the activity for an audit trail. AMIs must not contain authorized passwords or authorized keys.
- Linux-based AMIs must not have blank or null root passwords.

Windows-specific access policies

Windows-based AMIs must adhere to the following access policies, as well as the general access policies:

- For Windows Server 2016 and later, use `EC2Launch`.
- For Windows Server 2012 R2 and earlier, use the most recent version of `Ec2ConfigService` and enable `Ec2SetPassword`, `Ec2WindowsActivate`, and `Ec2HandleUserData`.
- Remove guest accounts and remote desktop users, none of which are allowed.

Customer information policies

All AMIs must adhere to the following customer information policies:

- Software must not collect or export customer data without the customer's knowledge and express consent except as required by BYOL (Bring Your Own License). Applications that collect or export customer data must follow these guidelines:
 - The collection of the customer data must be self-service, automated, and secure. Buyers must not need to wait for sellers to approve to deploy the software.
 - The requirements for customer data must be clearly stated in the description or the usage instructions of the listing. This includes what is collected, the location of where the customer data will be stored, and how it will be used. For example, *This product collects your name and email address. This information is sent to and stored by the <company name>. This information will only be used to contact the buyer in regards to the <product name>.*
 - Payment information must not be collected.

Product usage policies

All AMIs must adhere to the following product usage policies:

- Products must not restrict access to the product or product functionality by time, number of users, or other restrictions. Beta and prerelease products, or products whose sole purpose is to offer trial or

evaluation functionality, are not supported. Developer, Community, and BYOL editions of commercial software are supported, provided an equivalent paid version is also available in AWS Marketplace.

- All AMIs must be compatible with either the Launch from Website experience or AMI-based delivery through AWS CloudFormation. For Launch from Website, the AMI can't require customer or user data at instance creation to function correctly.
- AMIs and their software must be deployable in a self-service manner and must not require additional payment methods or costs. Applications that require external dependencies on deployment must follow these guidelines:
 - The requirement must be disclosed in the description or the usage instructions of the listing. For example, *This product requires an internet connection to deploy properly. The following packages are downloaded on deployment: <list of package>*.
 - Sellers are responsible for the use of and ensuring the availability and security of all external dependencies.
 - If the external dependencies are no longer available, the product must be removed from AWS Marketplace as well.
 - The external dependencies must not require additional payment methods or costs.
- For all products except BYOL, the fulfillment process must not require the customer to leave AWS Marketplace.
- AMIs that require an ongoing connection to external resources not under the direct control of the buyer—for example, external APIs or AWS services managed by the seller or a third party—must follow these guidelines:
 - The requirement must be disclosed in the description or the usage instructions of the listing. For example, *This product requires an ongoing internet connection. The following ongoing external services are required to properly function: <list of resources>*.
 - Sellers are responsible for the use of and ensuring the availability and security of all external resources.
 - If the external resources are no longer available, the product must be removed from AWS Marketplace as well.
 - The external resources must not require additional payment methods or costs and the setup of the connection must be automated.
- Product software and metadata must not contain language that redirects users to other cloud platforms, additional products, or upsell services that aren't available in AWS Marketplace.
- If your product is an add-on to another ISV's product, your product description must indicate that it extends the functionality of the other product and that without it, your product has very limited utility. For example, *This product extends the functionality of <product name> and without it, this product has very limited utility. Please note that <product name> might require its own license for full functionality with this listing.*

Architecture policies

All AMIs must adhere to the following architecture policies:

- Source AMIs for AWS Marketplace must be provided in the US East (N. Virginia) Region.
- AMIs must use HVM virtualization.
- AMIs must use 64-bit or 64-bit ARM architecture.
- AMIs must be AMIs backed by Amazon Elastic Block Store (Amazon EBS). We don't support AMIs backed by Amazon Simple Storage Service (Amazon S3).
- AMIs must not use encrypted file systems.
- AMIs must be built so that they can run in all AWS Regions and are Region-agnostic. AMIs built differently for different Regions aren't allowed.

Container-based products

AWS Marketplace supports software products that use Docker containers. Container products consist of delivery options that are a set of container images and deployment templates that go together. You submit at least one delivery option for your product, with up to a maximum of four. For each delivery option, you provide a set of container images, usage instructions, and links to deployment templates for customers to launch that delivery option.

AWS Marketplace buyers see the available delivery options on the published product detail pages that are available to them. After they subscribe to the product and choose their preferred delivery option, buyers see information and instructions for launching and using the product. For Container image delivery options, buyers see links to the available deployment templates and container image URLs. They also receive instructions for how to pull the individual container images. For Helm chart delivery options, buyers will see step-by-step instructions for launching using Helm.

For a walkthrough of the buying experience, you can refer to this video: [Deploying AWS Marketplace Containers on Amazon ECS Clusters \(3:34\)](#).

You can find, subscribe to, and deploy third-party Kubernetes applications from AWS Marketplace on any Kubernetes cluster in any environment. You can deploy third-party Kubernetes applications on Amazon Elastic Container Service (Amazon ECS), Amazon Elastic Kubernetes Service (Amazon EKS), AWS Fargate, and on-premises using Amazon EKS Anywhere (EKS Anywhere). You can also deploy them on self-managed Kubernetes clusters on-premises or in Amazon Elastic Compute Cloud (Amazon EC2).

You can run Free and Bring Your Own License model (BYOL) container products on any Docker-compatible runtime.

Topics

- [Getting help \(p. 116\)](#)
- [Getting started with container products \(p. 116\)](#)
- [Container-based product requirements \(p. 126\)](#)
- [Container product pricing \(p. 129\)](#)
- [Billing, metering, and licensing integrations \(p. 134\)](#)
- [Amazon SNS notifications for container products \(p. 174\)](#)

Getting help

For assistance with your container products, contact your business development partner for AWS Marketplace or [contact us](#).

Getting started with container products

This topic describes all the steps related to creating, testing, and publishing your first container product for AWS Marketplace.

Topics

- [Prerequisites \(p. 117\)](#)
- [Creating a container product \(p. 117\)](#)
- [Creating the product ID for your container product \(p. 118\)](#)

- [Creating or updating pricing details for container products \(p. 118\)](#)
- [Integrating AWS Marketplace Metering Service for your container product \(p. 119\)](#)
- [Integrating AWS License Manager for your container product \(p. 119\)](#)
- [Adding a new version of your product \(p. 119\)](#)
- [Updating version information \(p. 124\)](#)
- [Creating or updating product information for your container product \(p. 125\)](#)
- [Publishing container products \(p. 126\)](#)
- [Container product scans for security issues \(p. 126\)](#)

Prerequisites

Before you get started, you must complete the following prerequisites:

1. Access and use the [AWS Marketplace Management Portal](#). This is the tool that you use to register as a seller and manage the products that you sell on AWS Marketplace. For more information, see [AWS Marketplace Management Portal](#).
2. Register as a seller, and submit your tax and banking information. For more information, see [Seller registration process \(p. 6\)](#).
3. Create at least one container in Amazon Elastic Container Service (Amazon ECS), Amazon Elastic Kubernetes Service (Amazon EKS), or AWS Fargate. Make sure that you have links for the associated images.
4. Plan how you'll create and integrate your container product in AWS Marketplace.

We recommend that you plan your pricing, entitlement, and metering strategy well in advance of publicly publishing your product.

- For information about the requirements for container-based products, see [Container-based product requirements \(p. 126\)](#).
- For information about setting the pricing for your product, see [Container product pricing \(p. 129\)](#).
- For information about custom metering for your paid container-based product, see [Hourly and custom metering with AWS Marketplace Metering Service \(p. 134\)](#).

Creating a container product

Creating a container product involves the following steps:

1. [Create the product ID \(p. 118\)](#).
2. [Create the pricing details \(p. 118\)](#).
3. For paid products, [integrate metering into your product \(p. 119\)](#).
4. [Add a new version of your product \(p. 119\)](#), including:
 - a. Add repositories for your containers.
 - b. Upload the final containers into the repositories.
 - c. Create the first version of the product with your first container images.
5. [Update the product version information \(p. 124\)](#).
6. [Publish the product for buyers \(p. 126\)](#).

The first two steps must happen before you can edit any other details in AWS Marketplace. However, you can perform the other steps in the order that makes the most sense to you or in parallel.

The following topics describe each of these steps.

Creating the product ID for your container product

To get started with a container product, you must create a product ID record in AWS Marketplace. The product ID is used to track your product throughout its lifecycle.

Use the following procedure to create a new container product in the AWS Marketplace Management Portal, and generate the product ID.

To create the container product ID

1. Open a web browser and sign into the [AWS Marketplace Management Portal](#).
2. From the menu bar, expand **Assets**, and choose **Container**.
3. Enter a customer-facing name for your product, and choose **Create**. If necessary, you can change this name later.
4. Make a note of the **Product ID** for your later use when you create or update the product pricing details.

Tip

If you lose your product ID, you can find it in the AWS Marketplace Management Portal by choosing **Container** from the **Assets** menu. The **Containers** page shows a list of your products with their associated product IDs.

You now have your initial container product and product ID. Next, add pricing details for your product.

Creating or updating pricing details for container products

To update the pricing details for your container product, you must use a product load form (PLF). The PLF for your product is a spreadsheet that contains information about your product. The following procedure outlines using the PLF to update information about your product, including pricing details.

Note

For more information about pricing models for container products, see [Container product pricing \(p. 129\)](#).

Your pricing and metering must be aligned. For more information about metering with container products, see [Hourly and custom metering with AWS Marketplace Metering Service \(p. 134\)](#).

To update pricing for your container product by using the product load form

1. Open a web browser and sign into the [AWS Marketplace Management Portal](#).
2. From the menu bar, expand **Assets**, and choose **File upload**.
3. From **Product load forms for download** on the right side, choose **Containers Product Load Form**.
4. Open the PLF spreadsheet on your computer, and fill out the fields to define your product. This information includes your product ID that you made note of when you created your container product.

Tip

When viewing the PLF in Microsoft Excel, hover over each of the fields to show comments that provide guidance about how to fill in each field.

Provide pricing and metering dimensions, based on your pricing model for your product. For more information, see the following:

- [Product load form for custom metering \(p. 144\)](#)
- [Product load form for hourly metering \(p. 137\)](#)

Note

Required fields have a red header with the word **REQUIRED** in the spreadsheet. Make sure that all of these fields are filled out to avoid delays in processing your request.

5. Save your PLF.
6. If it's not still open, open a web browser and sign into the [AWS Marketplace Management Portal](#).
7. From the menu bar, expand **Assets**, and choose **File Upload**.
8. In **Upload File**, browse your computer and choose the PLF you saved for this container product.
9. Provide a brief description for this PLF to help you identify it among the other PLFs you upload.
10. Choose **Upload**. Your uploaded PLF appears in a table at the bottom of the page.

Your pricing details are reviewed and updated manually by the AWS Marketplace Seller Operations team. It typically takes a few business days to complete the update. You can check the status by choosing **Container** from the **Assets** menu in the AWS Marketplace Management Portal. An email message is sent to you when the review of your product pricing details is complete.

Note

Your container product is now created, in a limited state. Your account can view the product for testing and modify it. To make it visible to other test accounts, or when it's ready to be made publicly available, see [Publishing container products \(p. 126\)](#).

You can edit your container product pricing by following this same procedure, until you publicly publish the product.

After you create the pricing details for your product, you can add other product details, integrate metering into your product, and create a software version for your product.

Integrating AWS Marketplace Metering Service for your container product

For container-based products with usage pricing, you use the [AWS Marketplace Metering Service](#) for both checking entitlement to use your product and metering usage for billing. You must meter for the pricing model that you created when setting your pricing information. For more information, see [Hourly and custom metering with AWS Marketplace Metering Service \(p. 134\)](#).

Integrating AWS License Manager for your container product

For container-based products with contract pricing, you use the AWS License Manager to associate licenses with your product.

For more information about integrating with AWS License Manager, see [Contract pricing with AWS License Manager \(p. 152\)](#).

Adding a new version of your product

Your product might have several versions over its lifetime. Each version has a set of container images that are specific to that version.

Note

You can't add a version to your product until you have created the product ID and the pricing for your product. For more information about these steps, see [Creating the product ID for your container product \(p. 118\)](#), and [Creating or updating pricing details for container products \(p. 118\)](#).

Creating a version of your product involves the following steps:

Topics

- [Step 1: Adding repositories](#) (p. 120)
- [Step 2: Uploading container images and artifacts to repositories](#) (p. 120)
- [Step 3: Adding a new version to your container product](#) (p. 122)

Your container images and other artifacts for your product are stored in repositories in AWS Marketplace. Typically, you create one repository for each artifact needed, but the repository can store multiple versions of the artifact (with different tags).

Note

All images in your product deployment must use images from the AWS Marketplace repositories.

Step 1: Adding repositories

The following procedure describes how to add any needed repositories in AWS Marketplace.

To add repositories

1. Sign in to the [AWS Marketplace Management Portal](#).
2. Select **Server** from the **Products** menu.
3. On the **Server products** tab, select the product you want to modify, and then choose **Add repositories** from the **Request changes** dropdown.
4. Enter the name for the repository that you want to create. If you want to create more than one new repository, choose **Add new repository** for each additional repository, and give it a unique name.

Note

The repository will have this structure: `<repositoryID>.dkr.ecr.us-east-1.amazonaws.com/<sellerName>/<repositoryName>`. When you add items to the repository (in the following procedure), they will get a tag and have this structure: `<repositoryID>.dkr.ecr.us-east-1.amazonaws.com/<sellerName>/<repositoryName>:<tag>`. The `repositoryID` is an internal ID for AWS Marketplace. The `sellerName` is based on the name you created for your seller account. You define the `repositoryName` in this step. The `tag` is set when you upload an artifact to the repository.

5. Select **Submit**.

Note

You can have up to 50 repositories per product.

A new request is created and shown on the **Requests** tab. When it's completed, within minutes, you can start adding container images and other artifacts to the repositories you have created.

Step 2: Uploading container images and artifacts to repositories

To upload container images and artifacts to repositories

1. Sign in to the [AWS Marketplace Management Portal](#).
2. From the **Products** menu, choose **Server**.
3. On the **Server products** tab, select the product you want to modify.
4. Choose **Add repositories** from the **Request changes** dropdown.
5. Choose **View existing repositories**.

6. Select the repository to which you want to upload.
7. Select **View push commands** to open a list of instructions, including commands you can use to push Docker container images and Helm charts to that repository.

For general information about how to push container images and other artifacts to repositories, refer to [Pushing an image](#) in the *Amazon Elastic Container Registry User Guide*.

Note

You can use the following Amazon Elastic Container Registry (Amazon ECR) API operations when calling `docker pull` or `docker push`:

- `DescribeImages` – Use this to review the metadata about the images in a repository.
 - `GetAuthorizationToken` – Use to authenticate before uploading artifacts to the repository, then use `docker pull` or `docker push` commands.
 - `ListImages` – Use to view a list of images you pushed.
8. Use the commands listed to push any needed artifacts from your local repository to the AWS Marketplace repository for your product.

Note

The `tag` that you provide in the `push` commands is used to differentiate the version of the artifact that you are uploading to the repository. Use a tag that makes sense for the version the artifacts are a part of.

9. Repeat for each container image or artifact you need in your version.

Note

Your version can include up to 50 container images or artifacts in each delivery option. Refer to the following procedure for more information about delivery options.

After you upload your artifacts, you're ready to create the version of your product.

Note

Your container images are scanned automatically to see if they meet the [Container-based product requirements \(p. 126\)](#). For more information, refer to [Container product scans for security issues \(p. 126\)](#).

Adding a new delivery option without a template

1. To add a new delivery option without a template, choose **Add delivery option**. After adding an option, follow the instructions in the following steps to configure it.
2. Choose a delivery method for the delivery option. The delivery method determines how buyers will launch your software.
 - For a **Container image** delivery option, provide paths to container images in an Amazon Elastic Container Registry (Amazon ECR) repository that was created in the AWS Marketplace console. Buyers use the container image paths to launch the software by pulling the images directly into their environments.
 - For a **Helm chart** delivery option, provide paths to Helm charts in an Amazon ECR repository that was created in the AWS Marketplace console. Buyers install the Helm charts in their deployment environment to launch the software.
3. To add a **Container image** delivery option, perform the following steps:
 - a. In **Container images**, add the Amazon ECR URL to the container images that contain the product version software.
 - b. In **Delivery option title** and **Deployment option description**, enter a title and description for this delivery option.
 - c. In **Usage instructions**, enter detailed information to help your buyers use your software after launching it.

- d. In **Supported services**, select the environments that buyers can launch the software in.
 - e. In **Deployment templates**, add resources that buyers can use to launch the software. Enter a title and a URL to the resource for each template.
4. To add a **Helm chart** delivery option, perform the following steps:
 - a. In **Helm chart**, add the Amazon ECR URL to the Helm chart that buyers will install in their deployment environment to launch your software.
 - b. In **Container images**, add the Amazon ECR URL to the container images that contain the product version software.
 - c. In **Delivery option title** and **Deployment option description**, enter a title and description for this delivery option.
 - d. In **Usage instructions**, enter detailed information to help your buyers use your software after launching it.
 - e. In **Supported services**, select the environments that buyers can launch the software in.
 - f. *Optional* - In **Helm release name**, enter the name of the Kubernetes namespace where the Helm chart will be installed.
 - g. *Optional* - In **Helm installation namespace**, enter the name for the Helm release that will be used by the `helm install` command.
 - h. *Optional* - In **Kubernetes service account name**, enter the name of the Kubernetes service account that will be used to connect to AWS Identity and Access Management (IAM). The Kubernetes service account calls AWS services such as licensing or metering.
 - i. Choose to enable **QuickLaunch** on this product version. QuickLaunch is a feature in AWS Marketplace. Buyers can use QuickLaunch to create an Amazon EKS cluster quickly and launch your software on it by using AWS CloudFormation. For more information, see [QuickLaunch in AWS Marketplace](#).
 - j. In **Override parameters**, enter parameters that will be used in the Helm CLI commands that launch the software. Buyers can override the provided default values. If you have enabled QuickLaunch, also enter a parameter name and description for the CloudFormation form.
 - k. Choose **Hide passwords and secrets** to mask sensitive information in consoles, command line tools, and APIs. For more information, see the `NoEcho` parameter documentation in [Parameters](#) in the *AWS CloudFormation User Guide*.
 5. If needed, choose **Add delivery option** to add additional delivery options and perform the instructions in the previous steps to configure them.
 6. Select **Submit**.

Step 3: Adding a new version to your container product

To add a new version to your container product

1. Sign in to the [AWS Marketplace Management Portal](#).
2. Choose **Server** from the **Products** menu.
3. On the **Server products** tab, select the product you want to add a version to. Then choose **Add new version** from the **Request changes** dropdown.
4. On the **Add new version** page, enter the **Version title** and **Release notes** for your version.
5. After entering the version details, the next step is to add delivery options. Delivery options are sets of instructions and information that buyers can use to launch the software from your product version. Delivery options are known as *fulfillment options* to buyers.

Note

Your product can support multiple platforms with different container images (for example, Kubernetes and Ubuntu deployments). You can create one delivery option for each way that customers can set up your product, up to four delivery options for a product.

- a. If the product already has delivery options in other versions, you can use the existing option as a template to add a delivery option to the new version. In **Delivery options**, choose the delivery option that you want to add from the list. You can edit the option using the instructions in the following steps.
 - b. To add a new delivery option without a template, choose **Add delivery option**. After adding an option, follow the instructions in the following steps to configure it.
6. Choose a delivery method for the delivery option. The delivery method determines how buyers will launch your software.
- For a **Container image** delivery option, provide paths to container images in an Amazon Elastic Container Registry (Amazon ECR) repository that was created in the AWS Marketplace console. Buyers use the container image paths to launch the software by pulling the images directly into their environments.
 - For a **Helm chart** delivery option, provide paths to Helm charts in an Amazon ECR repository that was created in the AWS Marketplace console. Buyers install the Helm charts in their deployment environment to launch the software.
7. To add a **Container image** delivery option, perform the following steps:
- a. In **Container images**, add the Amazon ECR URL to the container images that contain the product version software.
 - b. In **Delivery option title** and **Deployment option description**, enter a title and description for this delivery option.
 - c. In **Usage instructions**, enter detailed information to help your buyers use your software after launching it.
 - d. In **Supported services**, select the environments that buyers can launch the software in.
 - e. In **Deployment templates**, add resources that buyers can use to launch the software. Enter a title and a URL to the resource for each template.
8. To add a **Helm chart** delivery option, perform the following steps:
- a. In **Helm chart**, add the Amazon ECR URL to the Helm chart that buyers will install in their deployment environment to launch your software.
 - b. In **Container images**, add the Amazon ECR URL to the container images that contain the product version software.
 - c. In **Delivery option title** and **Deployment option description**, enter a title and description for this delivery option.
 - d. In **Usage instructions**, enter detailed information to help your buyers use your software after launching it.
 - e. In **Supported services**, select the environments that buyers can launch the software in.
 - f. *Optional* - In **Helm release name**, enter the name of the Kubernetes namespace where the Helm chart will be installed.
 - g. *Optional* - In **Helm installation namespace**, enter the name for the Helm release that will be used by the `helm install` command.
 - h. *Optional* - In **Kubernetes service account name**, enter the name of the Kubernetes service account that will be used to connect to AWS Identity and Access Management (IAM). The Kubernetes service account calls AWS services such as licensing or metering.
 - i. Choose to enable **QuickLaunch** on this product version. QuickLaunch is a feature in AWS Marketplace. Buyers can use QuickLaunch to create an Amazon EKS cluster quickly and launch your software on it by using AWS CloudFormation. For more information, see [QuickLaunch in AWS Marketplace](#).
 - j. In **Override parameters**, enter parameters that will be used in the Helm CLI commands that launch the software. Buyers can override the provided default values. If you have enabled QuickLaunch, also enter a parameter name and description for the CloudFormation form.

- k. Choose **Hide passwords and secrets** to mask sensitive information in consoles, command line tools, and APIs. For more information, see the `NoEcho` parameter documentation in [Parameters](#) in the *AWS CloudFormation User Guide*.
9. If needed, choose **Add delivery option** to add additional delivery options and perform the instructions in the previous steps to configure them.
10. Select **Submit**.

Your request for a new version is created and should complete within minutes. You can track the request from the **Requests** tab of the **Server products** page.

Note

Your new version is available to all of your buyers. If your product is currently set to limited availability, your version is available to the set of buyers that the product is available for. If your product is currently set to public availability, then your new version is available to all AWS Marketplace buyers.

If this was your first version set, your product is now ready to be published. For information about how to publish a product, see [Publishing container products \(p. 126\)](#).

Your request for a new version is created and should complete within minutes. You can track the request from the **Requests** tab of the **Server products** page.

Note

Your new version is available to all of your buyers. If your product is currently set to limited availability, your version is available to the set of buyers that the product is available for. If your product is currently set to public availability, then your new version is available to all AWS Marketplace buyers.

If this was your first version set, your product is now ready to be published. The next topic describes publishing your product.

If this was your first version set, your product is now ready to be published, see [the section called "Publishing container products" \(p. 126\)](#).

Updating version information

After a version is created, it can be helpful to provide updated information to your buyers by modifying the information associated with the version. For example, if you plan to restrict version 1.0 after version 1.1 is released, you can update the description of version 1.0 to direct buyers to version 1.1. Provide the date that version 1.0 will be restricted. You update the version information from the AWS Marketplace Management Portal.

To update version information

1. Sign in to the [AWS Marketplace Management Portal](#).
2. Select **Server** from the **Products** menu.
3. On the **Server products** tab, select the product that you want to modify.
4. From the **Request changes** dropdown, choose **Update version information**.
5. On the **Update version** page, select the version that you want to update and choose **Submit**.
6. Make updates to the selected version. The fields that are available for updating depend on the status of the product version or delivery option.
 - a. For all versions, you can update the **Release notes**.
 - b. For versions that are not yet publicly available, you can update the **Version title**.
 - c. For delivery options that haven't been restricted, you can update the following fields:

- **Description**
 - **Usage instructions**
 - **Supported services**
- d. For delivery options in versions that are not yet publicly available, you can update the following fields:
- **Delivery option titles**
 - **Helm chart** (for **Helm chart** delivery options only)
 - **Container images**
 - **Deployment resources**
7. Choose **Submit**.
8. Verify that the request appears on the **Requests** tab with the **Under review** status.

You can check the status of your request at any time from the **Requests** tab of the [Server Products](#) page.

Creating or updating product information for your container product

After you have created your product ID and set the pricing, you can edit your product information, including what customers will see about your container product in the AWS Marketplace. The following procedure outlines creating the product details for your product.

To create or update product details for your container product

1. Sign in to the [AWS Marketplace Management Portal](#).
2. Select **Server** from the **Products** menu.
3. On the **Server products** tab, select the product that you want to modify.
4. From the **Request changes** dropdown, choose **Update product information**.
5. Update any of the following fields that you need to change:
 - **Product title**
 - **SKU**
 - **Short description**
 - **Long description**
 - **Product logo image URL**
 - **Highlights**
 - **Product categories**
 - **Keywords**
 - **Product video URL**
 - **Resources**
 - **Support information**

Note

Image URLs must be in an Amazon S3 bucket that is publicly accessible. For more details about the logo format, see [Company and product logo requirements \(p. 262\)](#).

6. Select **Submit**.
7. Verify that the request appears on the **Requests** tab with the **Under review** status. You might need to refresh the page to see the request on the list.

You can check the status of your request at any time from the **Requests** tab of the [Server Products](#) page.

Publishing container products

When you initially create your product, its availability is limited to just your account. Once your product is ready for testing (including having product details filled in and the first version created), you can make it available to other accounts for testing, or to all accounts as a public product.

Note

Before publishing publicly, you should review your product to ensure accuracy, including image links, deployment templates, descriptions, and pricing. Your pricing model can't be changed for publicly published products.

To publish your limited product to additional accounts or for public availability, contact the [AWS Marketplace Seller Operations team](#). In your request, provide the product ID and describe the changes that you want to make.

Note

You can also choose to restrict a version that you no longer want available to buyers. You can include this in a request to publish a product publicly, to avoid test versions appearing in public products.

You can't restrict a version if it will leave your public product with no public versions.

When you publicly publish a container product, you make it visible to all AWS customers who can then subscribe and launch your product. The AWS Marketplace Seller Operations team reviews the data in your product information, as well as your test calls to the AWS Marketplace Metering Service.

Container product scans for security issues

When you submit a container image URL, we scan it and check for security vulnerabilities. We examine the images you provide for known security vulnerabilities. To do this, we perform a layer-by-layer static scan on the image. If we find critical vulnerabilities with remotely exploitable risk vectors, we provide you with a list of found issues. We strongly recommend that you perform your own security analysis using a container image scanner such as Clair, Twistlock, Aqua Security, or Trend Micro to avoid delays in the ingestion and publishing process.

Your choice of base image for building your container images can have a significant influence on the security profile of the final image. If you choose a base image that already has known critical vulnerabilities, they will be flagged because of the base layer, even if your application software layers are clean. We recommend that you verify that you're starting with a vulnerability-free base container before you build your images and submit them to AWS Marketplace.

Container-based product requirements

AWS Marketplace maintains the following requirements for all container-based products and offerings on AWS Marketplace. These requirements help to promote a safe, secure, and trustworthy catalog for our customers. We also encourage sellers to review implementation of additional controls and protocols as applicable to meet the needs of their specific products.

All products and their related metadata are reviewed when submitted to ensure that they meet or exceed current AWS Marketplace requirements. We review and adjust these policies to meet our evolving security and other usage requirements. AWS Marketplace continuously verifies that existing products continue to meet any changes to these requirements. If products fall out of compliance, AWS Marketplace will contact you to update your product. In some cases, your product might temporarily be unavailable to new subscribers until issues are resolved.

Security requirements

All container-based products must adhere to the following security requirements:

- Docker container images must be free from any known malware, viruses, or vulnerabilities. The self-service container images ingestion tool can detect vulnerabilities. To use the container scanning tool, sign into the AWS Marketplace Management Portal, select **Container** from the **Assets** menu, and submit an image for your product.
- If your container-based products requires access to manage AWS resources, it must be achieved through [IAM roles for service accounts](#) (if run through Amazon Elastic Kubernetes Service (Amazon EKS)) or [IAM roles for tasks](#) (if run through Amazon Elastic Container Service (Amazon ECS)) instead of requesting an access key from users.
- Container-based products must only require least privileges to run. For more information, see [ECS security](#) and [EKS security](#).
- Container images should be configured to run with non-root privileges by default.

Access requirements

All container-based products must adhere to the following access requirements:

- Container-based products must use an initial randomized password. Container-based products must not use initial fixed or blank passwords for external administrative access (for example, to log in to the application via a web interface). The buyer must be prompted for this randomized password before being permitted to set or change their own credentials.
- Any outside access to the application must be explicitly agreed to and enabled by customers.

Customer information requirements

All container-based products must adhere to the following customer information requirements:

- Software must not collect or export customer data without the customer's knowledge and express consent except as required by BYOL (Bring Your Own License). Applications that collect or export customer data must follow these guidelines:
 - The collection of the customer data must be self-service, automated, and secure. Buyers must not need to wait for sellers to approve to deploy the software.
 - The requirements for customer data must be clearly stated in the description or the usage instructions of the listing. This includes what is collected, the location of where the customer data will be stored, and how it will be used. For example, *This product collects your name and email address. This information is sent to and stored by the <company name>. This information will only be used to contact the buyer in regards to the <product name>.*
- Payment information must not be collected.

Product usage requirements

All container-based products must adhere to the following product usage requirements:

- Sellers can only list fully functioning products. Beta or prerelease products for trial or evaluation purposes are not allowed. Developer, community, and BYOL editions of commercial software are supported if the seller provides an equivalent paid version on AWS Marketplace within 90 days of providing the free edition.

- All of a container-based product's usage instructions must include all steps to deploy container-based products. Usage instructions must provide commands and deployment resources pointing to the corresponding container images on AWS Marketplace.
- Container-based products must include all container images that a subscriber needs to use the software. In addition, container-based products must not require a user to launch the product using any images from outside AWS Marketplace (for example, container images from third-party repositories).
- Containers and their software must be deployable in a self-service manner and must not require additional payment methods or costs. Applications that require external dependencies on deployment must follow these guidelines:
 - The requirement must be disclosed in the description or the usage instructions of the listing. For example, *This product requires an internet connection to deploy properly. The following packages are downloaded on deployment: <list of package>*.
 - Sellers are responsible for the use of and ensuring the availability and security of all external dependencies.
 - If the external dependencies are no longer available, the product must be removed from AWS Marketplace as well.
 - The external dependencies must not require additional payment methods or costs.
- Containers that require an ongoing connection to external resources not under the direct control of the buyer—for example, external APIs or AWS services managed by the seller or a third party—must follow these guidelines:
 - The requirement must be disclosed in the description or the usage instructions of the listing. For example, *This product requires an ongoing internet connection. The following ongoing external services are required to properly function: <list of resources>*.
 - Sellers are responsible for the use of and ensuring the availability and security of all external resources.
 - If the external resources are no longer available, the product must be removed from AWS Marketplace as well.
 - The external resources must not require additional payment methods or costs and the setup of the connection must be automated.
- Product software and metadata must not contain language that redirects users to other cloud platforms, additional products, or upsell services that aren't available on AWS Marketplace.
- If your product is an add-on to another ISV's product, your product description must indicate that it extends the functionality of the other product and that without it, your product has very limited utility. For example, *This product extends the functionality of <product name> and without it, this product has very limited utility. Please note that <product name> might require its own license for full functionality with this listing.*

Architecture requirements

All container-based products must adhere to the following architecture requirements:

- Source container images for AWS Marketplace must be pushed to the Amazon Elastic Container Registry (Amazon ECR) repository owned by AWS Marketplace. You can create these repositories in the AWS Marketplace Management Portal under server products for each of your container product listings.
- Container images must be based on Linux.
- Paid container-based products must be able to be deployed on [Amazon ECS](#), [Amazon EKS](#), or [AWS Fargate](#).
- Paid container-based products with contract pricing and an integration with AWS License Manager should deploy on Amazon EKS, Amazon ECS, AWS Fargate, Amazon EKS Anywhere, Amazon ECS

Anywhere, Red Hat OpenShift Service on AWS (ROSA), self-managed Kubernetes clusters on-premises, or on Amazon Elastic Compute Cloud.

Container product pricing

This section outlines the available pricing models for container products. You can list free products, Bring Your Own License model (BYOL) products, and paid products for Amazon Elastic Container Service (Amazon ECS), Amazon Elastic Kubernetes Service (Amazon EKS), and AWS Fargate. You can set only one price per product.

Note

You use the [AWS Marketplace Metering Service](#) to enforce entitlement and meter usage for your paid products. For per task or per pod pricing, usage is metered automatically by AWS.

The price you set for a container product applies to all AWS Regions. Whenever you lower the price for a container product, the new price is implemented for your buyers immediately. For price increases, existing buyers are notified about the change 90 days before it impacts their billing. New buyers are billed the new amount.

Container pricing models

AWS Marketplace has multiple pricing models for container products.

The following table provides general information about pricing models for container-based products.

Pricing models for container products

| Pricing model | Description |
|----------------------------------|---|
| Bring Your Own License (BYOL) | BYOL is managed outside of AWS Marketplace through an external billing relationship that you maintain with the buyer. |
| Monthly | <p>Fixed monthly price A fixed monthly price that provides users with unlimited product usage during the following month.</p> <p>Example: You set the price for your product at \$99 per month. Your product includes three different container images that are deployed using an Amazon ECS task definition.</p> <p>After a buyer subscribes to your product, they're immediately charged \$99, which repeats each month until they cancel the subscription. The buyer also gets unlimited usage of the product. The buyer also pays separately for any infrastructure that the tasks run on. While subscribed, they can access your container images. They can launch and run any number of containers from those images on Amazon ECS or Amazon EKS in any configuration.</p> <p>If the buyer cancels their subscription in the middle of a month, they lose access to the Amazon ECR repository where AWS Marketplace stores the container images. The buyer might have pulled and stored the original images. However, they can no longer pull new container image versions that you make available through AWS Marketplace. The buyer is refunded for the unused portion of the final month. You're paid based on the buyer's usage minus the agreed-to AWS Marketplace fee.</p> |
| Custom metric pricing dimensions | Custom metered prices based off of dimensions you define (for example users, nodes, repositories, or GB), up to 24 dimensions per product. |

| Pricing model | Description |
|---|--|
| | <p>Example: Your product charges by users. You have admin users and regular users, and you define the pricing as \$2 for admin users and \$1 for regular users. You can set them up as separate dimensions when listing your product. You charge by users logged in per day and you meter that usage per day.</p> |
| <p>Per task or per pod hourly price</p> | <p>Amazon ECS task or Amazon EKS pod</p> <p>Per Amazon ECS task or per Amazon EKS pod pricing that we measure to the second with the price set per hour.</p> <p>Example: Your product includes three different container images: a controller node, a worker node, and an analytics node. Because your product isn't functional or useful without the controller node, you decide that is the image that you want to charge usage for. You set a price of \$6 per hour.</p> <p>You modify the software in the container image for the controller node to integrate with the AWS Marketplace Metering Service RegisterUsage API operation. This ensures that only buyers with an active subscription can launch and run that container image and that its usage is metered based on how long it runs.</p> <p>The buyer is charged \$6 per hour of usage for each Amazon EKS controller pod running. If the buyer launches five Amazon EKS controller pods that include the controller node container, they're charged \$30 per hour (\$6 per pod). The buyer also pays separately for any infrastructure that the pods run on.</p> <p>For hourly pricing, billing is per-second with a 1-minute minimum. If the customer runs this controller container for 20 minutes and 30 seconds, they're charged $20 \times (\\$6/60) + 30 \times (\\$6/60/60) = \\$2 + \\$0.05 = \\$2.05$. You're paid based on the buyer's usage minus the agreed-to AWS Marketplace fee.</p> |

| Pricing model | Description |
|--------------------------------------|--|
| Hourly/usage with long-term contract | <p>A long-term contract, at a reduced price, paid up front or in regular installments. A long-term contract can be added to an existing product that has custom metered pricing, or per task and per pod pricing. Buyers pay the metered prices when they consume more than what they purchased in the long term contract.</p> <p>Example: For metered pricing models, you can add a long-term contract price for buyers to get a discount for committing upfront. Say that you normally charge \$1 per some unit consumed. A buyer using 1 unit per hour would pay \$8760 per year (365 days x 24 hours x \$1 per hour). You could enable a contract that enables the buyer to use 1 unit per hour for those 365 days at half that price (\$4380). In this case, the buyer commits to pay upfront for the one-year contract, and the price drops from \$1 per unit to \$0.5 per unit. You could also enable the buyer to purchase multiple of these contracts. If the quantity that is metered showed that the buyer consumed 10 units in an hour, and they had two contracts, then 2 units will be included in the 2 contracts. The 8 additional units would be billed at the regular \$1 per hour, for a total of \$8 in that hour.</p> <p>For the per task or per pod example, you can also add a long-term contract price for buyers to get a discount for committing upfront. If you normally charge \$6 per pod, you could set a long-term contract duration of 365 days with a price of \$13,140 (365 days x 24 hours x \$3 per pod per hour). One contract would then entitle the customer to 1 pod per hour during those 365 days. Customers can choose to purchase multiple contracts. For example, a customer can purchase two contracts which entitles them to 2 pods per hour. If the customer runs more pods per hour than the entitled contracts, then excess pods will be billed at your normal hourly price.</p> <p>In both cases, buyers that purchase long-term contracts will be billed upfront, either as a one-time payment or regularly scheduled future payments. Buyers will also be billed for any additional usage above their contract at the metered rate.</p> |
| Container contract pricing | <p>AMI with contract pricing – A container-based product that the buyer pays an upfront fee for.</p> |

Contract pricing for container products

For container-based products with contract pricing, AWS Marketplace bills your customers upfront or by the payment schedule that you define, based on the contract between you and your customer. After that point, they're entitled to use those resources.

To set your pricing, choose one or more contract durations that you offer customers. You can enter different prices for each contract duration. Your options are 1-month, 12-months, 24-month, and 36-month durations. For private offers, you can specify a custom duration in months (up to 60 months).

Choose the category that best describes your product's pricing. The pricing category appears to customers on the AWS Marketplace website. You can choose from **Bandwidth** (GB/s, MB/s), **Data** (GB, MB, TB), **Hosts**, **Requests**, **Tiers**, or **Users**. If none of the predefined categories fit your needs, you can choose the more generic **Units** category.

The offer allows for up to 24 dimensions to be added to it. Each dimension requires the following data:

- **Contracts Category** – The contract category is used to measure or meter your product if the product supports consumption based metering on top of contract pricing. For contract products with no consumption based pricing, you can choose a category which most closely resembles the category of

dimension in the contract. If no values resemble the units for the dimension in the contract, choose `Units`.

- **Contracts Unit** – The contract unit is used along with category for metering if the product supports consumption based metering. Choose one of the available values for the units that closely matches your dimensions based on the category selected.
- **Contracts Dimension Allow Multiple Purchases** – This field is used to indicate whether an offer is a tiered pricing offer or a non-tiered offer which allows for purchase of multiple dimensions.

Tiered offer – Allows the buyer to subscribe to only one of the available dimensions in the offer. Dimensions in a tiered offer don't have the concept of quantities. Signing a contract with a specific dimension essentially indicates that the buyer has chosen the specific feature indicated by that dimension.

Non-tiered offer – Allows the customer to procure more than one dimensions as part of the contract and allows them to procure multiple units of each such dimension.

Setting a value of `true` for this field indicates that the offer is a non-tiered offer. Setting a value of `false` for this field indicates that the offer is a tiered offer.

When using the Product Load Form (PLF) to create the contracts for your Container product, you must define the following fields for your pricing dimensions:

- **Contracts DimensionX API Name** – The name that should appear in the license generated in the buyer's AWS License Manager account. This name is also used as the value for `Name` in `Entitlement` in the `CheckoutLicense` API call.
- **Contracts DimensionX Display Name** – The customer-facing name of the dimension that will be displayed on the product detail and procurement pages of the AWS Marketplace website. Create a name that is user-friendly. The name's maximum length is 24 characters. After the listing is public, the value of `Name` can't be changed.
- **Contracts DimensionX Description** – The customer-facing description of a dimension that provides additional information about the dimension for the product, such as the capabilities that the specific dimension provides. The maximum length for the description is 70 characters.
- **Contracts DimensionX Quantity** – This is used to calculate proration in cases of agreement amendments to a product. This value of this field should be set to 1 for all contract offers. It should not be edited.
- **Contracts DimensionX 1-Month Rate** – The contract rate to be charged for 1-month of entitlements against this dimension. For non-tiered offers, this rate is charged for each unit of the dimension that is procured. This field supports three decimal places.
- **Contracts DimensionX 12-Month Rate** – The contract rate to be charged for 12 months of entitlements against the dimension. For non-tiered offers, this rate is charged for each unit of the dimension that is procured. This field supports three decimal places.
- **Contracts DimensionX 24-Month Rate** – The contract rate to be charged for 24 months of entitlements against the dimension. For non-tiered offers, this rate is charged for each unit of the dimension that is procured. This field supports three decimal places.
- **Contracts DimensionX 36-Month Rate** – The contract rate to be charged for 36 months of entitlements against the dimension. For non-tiered offers, this rate is charged for each unit of the dimension that is procured. This field supports three decimal places.

Example: Data storage application

| | 1-month price | 12-month price | 24-month price | 36-month price |
|-----------------------|---------------|----------------|----------------|----------------|
| Unencrypted data (GB) | \$1.50/GB | \$16.00/GB | \$30.00/GB | \$60.00/GB |

| | 1-month price | 12-month price | 24-month price | P36-month price |
|---------------------|---------------|----------------|----------------|-----------------|
| Encrypted data (GB) | \$1.55/GB | \$16.60/GB | \$31.20/GB | \$61.20/GB |

Example: Log monitoring product

| | 1-month price | 12-month price | 24-month price | 36-month price |
|--|---------------|----------------|----------------|----------------|
| Basic (10 hosts monitored, 5 containers monitored) | \$100 | \$1000 | \$2000 | \$4000 |
| Standard (20 hosts monitored, 10 containers monitored) | \$200 | \$2000 | \$4000 | \$8000 |
| Pro (40 hosts monitored, 20 containers monitored) | \$400 | \$4000 | \$8000 | \$16,000 |
| Additional hosts monitored per hour | \$10 | \$100 | \$200 | \$400 |
| Additional containers monitored per hour | \$10 | \$100 | \$200 | \$400 |

Note

The prices can be for the following durations: 1 month, 12 months, 24 months, or 36 months. You can choose to offer one or more of these options for your product. The durations must be the same across each dimension.

Example

For example, in a case where you have `ReadOnlyUsers` and `AdminUsers` dimensions, if you offer a yearly price for `ReadOnlyUsers`, you must offer a yearly price for `AdminUsers`, too.

Automatic renewals

When a customer purchases your product through AWS Marketplace using Container contracts, they can agree to automatic renewal of the contract terms. The customer continues to pay for the entitlements every month or for 1, 2, or 3 years. The customer always has the option to modify the renewal settings. They can cancel the renewal or renew the contract different quantities and durations.

When a container contract ends

A container contract product has a contract expiry. When a contract ends, the following events occur:

1. Your container product receives an `entitlement-updated` notification indicating that the buyer's entitlement has changed, and the AWS Marketplace Entitlement Service returns an empty response.

2. You have one hour to meter any remaining usage for the customer. After this you can no longer send metering records for this customer.

Billing, metering, and licensing integrations

AWS Marketplace integrates with other AWS services to provide both metering and contract-based pricing for your container product.

Hourly and custom metering with AWS Marketplace Metering Service

To both check entitlement to use your product and to meter usage for billing, use the [AWS Marketplace Metering Service](#). If you want to define your own pricing units and meter that usage to us for billing, integrate by using the [MeterUsage](#) API operation. If you want to price your product based on number of tasks or pods used and have AWS meter that usage automatically, integrate by using the [RegisterUsage](#) API operation. For both types of pricing, you can add a long-term contract price without changing how you integrate with the AWS Marketplace Metering Service.

When you create a new container product in the AWS Marketplace Management Portal, we provide a set of product identifiers (the product code and public key) that are used to integrate your product with the AWS Marketplace Metering Service.

Entitlement

Integrating with the AWS Marketplace Metering Service allows you to verify that the customer running your paid software is subscribed to your product on AWS Marketplace, guarding you against unauthorized use at container startup. To verify entitlement, use the [MeterUsage](#) or [RegisterUsage](#) API operations, depending on your pricing model. For hourly and fixed monthly pricing models, use the [RegisterUsage](#) API operation. For custom metering pricing models, use the [MeterUsage](#) API operation.

If a buyer isn't entitled to your product, these API operations return the `CustomerNotEntitledException` exception.

Note

If a buyer unsubscribes from your product while running it, they are entitled to continue running it. However, they can't launch additional containers for your product.

Integration guidelines

As you create and publish your container products and use the [MeterUsage](#) or [RegisterUsage](#) API operations for entitlement and metering, keep the following guidelines in mind:

- Don't configure AWS credentials within your software or the Docker container image. AWS credentials for the buyer are automatically obtained at runtime when your container image is running within an Amazon ECS task or Amazon EKS pod.
- To call the [MeterUsage](#) or [RegisterUsage](#) API operations from Amazon EKS, you must [use a supported AWS SDK](#). To test [MeterUsage](#) or [RegisterUsage](#) integration of Amazon EKS, you must run an Amazon EKS cluster running Kubernetes 1.13.x or greater. Kubernetes 1.13 is required for AWS Identity and Access Management (IAM) roles for pod support, which is a dependency for the running pod to obtain the AWS credentials required to invoke these actions on Amazon EKS.
- You can do local development, but you will get a `PlatformNotSupportedException` exception. This exception won't occur when you launch the container on AWS container services (Amazon ECS, Amazon EKS, and Fargate).

Supported AWS Regions

For a list of all AWS Marketplace supported AWS Regions, see [Region Table](#) on the Global Infrastructure website.

Obtaining the AWS Region for metering

When integrating your container for metering with either the `MeterUsage` or `RegisterUsage` API operation, don't configure the AWS SDK to use a specific AWS Region. The Region must be obtained dynamically at runtime.

Example

For example, a customer launches an Amazon ECS task or Amazon EKS pod. The `RegisterUsage` API operation is called in a Region that differs from the Region where the Amazon ECS task or Amazon EKS pod was launched. Therefore, the `RegisterUsage` API operation throws an `InvalidRegionException` error.

AWS SDK languages don't determine the `AWS_REGION` in a consistent manner. For example, the AWS SDK for Java automatically uses [Amazon EC2 instance metadata](#) (specifically, `ec2InstanceMetadata`) to obtain the Region when environment variables or other configuration aren't present. In this instance, only call `ec2InstanceMetadata` if the `AWS_REGION` environment variable isn't present.

For information about how to dynamically obtain an AWS Region at runtime, refer to the [AWS SDK Developer Guide](#) for your programming language.

Preventing metering modification

Introducing ways for buyers to modify or override calls to `RegisterUsage` or `MeterUsage` might result in undesirable billing and payment issues. We strongly recommend that you integrate the metering and entitlement logic.

When engineering your product to prevent metering modification, keep the following in mind:

- If buyers can insert new image layers that contain `CMD` or `ENTRYPOINT` instructions, directly integrate `RegisterUsage` or `MeterUsage` into the software that the buyer is running through your container image. Otherwise, calls to `RegisterUsage` or `MeterUsage` executed via `CMD` or `ENTRYPOINT` from the base image will likely be overridden by the buyer.
- We recommend that you manage the AWS Marketplace product codes that your software uses as input to `RegisterUsage` or `MeterUsage` in a manner buyers can't modify. However, if your product manages product codes in a manner customers can override, such as AWS CloudFormation, Helm chart, or Kubernetes manifest, you must maintain a list of *trusted* AWS Marketplace product codes. This is to ensure that the product code your software passes as input to `RegisterUsage` or `MeterUsage` is valid.
- If any of your trusted product codes are for free products, ensure that they can't be used in place of a paid product code.

Contract pricing with AWS License Manager

For container-based products with contract pricing, you use AWS License Manager to associate licenses with your product.

AWS License Manager is a license management tool that enables your application to track and update licenses (also known as entitlements) that have been purchased by a customer. This section provides information about how to integrate your product with AWS License Manager. After the integration is complete, you can publish your product listing on AWS Marketplace.

For more information about AWS License Manager, see the [AWS License Manager User Guide](#) and the [AWS License Manager](#) section of the *AWS CLI Command Reference*.

Note

- Customers can't launch new instances of the container after the contract expiry period. However, during the contract duration, they can launch any number of instances. These licenses are not node-locked or tied to particular instances.
- **Private Offer Creation** – Sellers can generate private offers for the products using the Private offer creation tool in the AWS Marketplace Management Portal.
- **Reporting** – You can set up data feeds by setting up an Amazon S3 bucket in the **Report** section in the AWS Marketplace Management Portal. For more information, see [Seller reports and data feeds \(p. 273\)](#).

Integration workflow

The following steps show the workflow for integrating your container product with AWS License Manager:

1. Seller creates a product with AWS License Manager integration.
2. Seller lists the product on AWS Marketplace.
3. Buyer finds the product on AWS Marketplace and purchases it.
4. A license is sent to the buyer in their AWS account.
5. Buyer uses the software by launching the Amazon EC2 instance, Amazon ECS task, or Amazon EKS pod software. The customer deploys using an IAM role.
6. Software reads the license in the buyer's AWS License Manager account, discovers the entitlements purchased, and provisions the features accordingly.

Note

License Manager doesn't do any tracking or updates; this is done by the seller's application.

Hourly metering with AWS Marketplace Metering Service

If your container product uses per-hour per-task or per-pod pricing instead of custom metered pricing dimensions, you don't have to define custom metering dimensions.

The `RegisterUsage` API operation meters software use per Amazon Elastic Container Service (Amazon ECS) task or per Amazon Elastic Kubernetes Service (Amazon EKS) pod, per hour, with usage prorated to the second. A minimum of 1 minute of usage applies to tasks or pods that are short lived. Continuous metering for software use is automatically handled by the AWS Marketplace Metering Control Plane. Your software isn't required to perform any metering specific actions except calling `RegisterUsage` once for metering of software use to commence.

The AWS Marketplace Metering Control Plane continues to bill customers for running Amazon ECS tasks and Amazon EKS pods, regardless of the customer's subscription state. This removes the need for your software to perform entitlement checks after the initial successful launch of the task or pod.

Hourly metering prerequisites

Before publishing the product, you must do the following:

1. Create a new container product in the AWS Marketplace Management Portal, and make a note of its product code.

For more information, see [Creating a container product \(p. 117\)](#).
2. Fill out the product load form (PLF) with the necessary hourly price information, and return it to us for processing.

For more information, see [Creating or updating pricing details for container products \(p. 118\)](#).
3. Use an AWS Identity and Access Management (IAM) role for the task or pod running your application with the IAM permissions necessary to call `RegisterUsage`. The IAM managed policy `AWSMarketplaceMeteringRegisterUsage` has these permissions.
4. (Optional) If you want to see logging, we recommend that you enable AWS CloudTrail logging in the task or pod definition.
5. Make a test call to the `RegisterUsage` API operation with a record for all of the pricing dimensions you define.

Product load form for hourly metering

When filling out the product load form for hourly metering, fill out the following fields for your product, in addition to the other required and optional fields that define your product:

- **Hourly Price** – The price for your product, per hour.
- **Dimension Long Term Rate** – The total software price over a long-term contract when buyers pay upfront.
- **Long Term Duration (Days)** – The duration, in days, for the long-term contract.

Testing integration and preview mode for `RegisterUsage`

Use the `RegisterUsage` API operation to test your integration before submitting your image to AWS Marketplace for publishing.

Preview mode operates identically to production mode, except preview mode does not verify entitlement to use your product. To call `RegisterUsage` in preview mode, call `RegisterUsage` from the container image by running your product on Amazon ECS or Amazon EKS. Use the AWS account that you're using to list the product on AWS Marketplace. Your metering integration must dynamically set the AWS Region, rather than hardcoding it. However, when testing, launch at least one Amazon ECS task or Amazon EKS pod containing your paid container in the US East (N. Virginia) Region. By doing this, the AWS Marketplace operations team can verify your work with the logs in that Region.

Note

If your product supports both Amazon ECS and Amazon EKS, you only need to launch in Amazon EKS for us to validate your integration.

You can't fully test the integration until your product is published with all the required metadata and pricing information. If requested, the AWS Marketplace catalog operations team can verify receipt of your metering records in preview mode.

Error handling for `RegisterUsage`

If your container image integrates with the AWS Marketplace Metering Service and receives an exception other than `ThrottlingException` at container startup, you should terminate the container to prevent unauthorized use.

Exceptions other than `ThrottlingException` are thrown only on the initial call to the `RegisterUsage` API operation. Subsequent calls from the same Amazon ECS task or Amazon EKS pod don't throw `CustomerNotSubscribedException` even if the customer unsubscribes while the task or

pod is still running. These customers are still charged for running containers after they unsubscribe, and their usage is tracked.

The following table describes the errors that the `RegisterUsage` API operation might throw. Each AWS SDK programming language has a set of error handling guidelines that you can refer to for additional information.

| Error | Description |
|--|--|
| <code>InternalServerErrorException</code> | <code>RegisterUsage</code> isn't available. |
| <code>CustomerNotEntitledException</code> | The customer doesn't have a valid subscription for the product. |
| <code>InvalidProductCodeException</code> | The <code>ProductCode</code> value passed in as part of the request doesn't exist. |
| <code>InvalidPublicKeyException</code> | The <code>PublicKeyVersion</code> value passed in as part of the request doesn't exist. |
| <code>PlatformNotSupportedException</code> | AWS Marketplace doesn't support metering usage from the underlying platform. Only Amazon ECS, Amazon EKS, and AWS Fargate are supported. |
| <code>ThrottlingException</code> | The calls to <code>RegisterUsage</code> are throttled. |
| <code>InvalidRegionException</code> | <code>RegisterUsage</code> must be called in the same AWS Region that the Amazon ECS task or Amazon EKS pod was launched in. This prevents a container from choosing a Region (for example, <code>withRegion("us-east-1")</code>) when calling <code>RegisterUsage</code> . |

Integrating your container product with the AWS Marketplace Metering Service using the AWS SDK for Java

The following steps outline an example implementation using the AWS SDK for Java to integrate with the [AWS Marketplace Metering Service's RegisterUsage](#) action. For the full source code, see [RegisterUsage Java example \(p. 140\)](#). Many of these steps apply regardless of the language.

Example steps for AWS Marketplace Metering Service integration

1. Sign into the [AWS Marketplace Management Portal](#).
2. From **Assets** choose **Containers** to start creating a new container product. Creating the product generates the product code for the product to integrate with your container image. For more information about publishing, see [Publishing container products \(p. 126\)](#). For information about setting IAM permissions, see [the section called "AWS Marketplace metering and entitlement API permissions" \(p. 358\)](#).
3. Download the public [AWS Java SDK](#).

Important

To call the metering APIs from Amazon EKS, you must [use a supported AWS SDK](#) and run on an Amazon EKS cluster running Kubernetes 1.13 or later.

4. (Optional) If you're integrating with the `RegisterUsage` action and you want to perform digital signature verification, you need to configure the [BouncyCastle](#) signature verification library in your application classpath.

If you want to use JSON Web Token (JWT), you must also include [JWT Java](#) libraries in your application classpath. Using JWT provides a simpler approach to signature verification but is not required, and you can use standalone BouncyCastle instead. Whether you use JWT or BouncyCastle, you need to use a build system such as Maven to include transitive dependencies of BouncyCastle or JWT in your application classpath.

```
// Required for signature verification using code sample
<dependency>
  <groupId>org.bouncycastle</groupId>
  <artifactId>bcpkix-jdk15on</artifactId>
  <version>1.60</version>
</dependency>

// This one is only required for JWT
<dependency>
  <groupId>com.nimbusds</groupId>
  <artifactId>nimbus-jose-jwt</artifactId>
  <version>6.0</version>
</dependency>
```

5. Call `RegisterUsage` from each paid container image in your product offering. `ProductCode` and `PublicKeyVersion` are required parameters, and all other inputs are optional. The following is an example payload for `RegisterUsage`.

```
{
  "ProductCode" : "string", // (required)
  "PublicKeyVersion": 1,    // (required)
  "Nonce": "string",       // (optional) to scope down the registration
                           //                   to a specific running software
                           //                   instance and guard against
                           //                   replay attacks
}
```

Note

It is possible to see transient issues in connecting to the AWS Marketplace Metering Service. AWS Marketplace strongly recommends implementing retries for up to 30 minutes, with exponential back off, to avoid short-term outages or network issues.

6. `RegisterUsage` generates an RSA-PSS digital signature using SHA-256 that you can use to verify request authenticity. The signature includes the following fields: `ProductCode`, `PublicKeyVersion`, and `Nonce`. To verify the digital signature, you must retain these fields from the request. The following code is an example response to a `RegisterUsage` call.

```
{
  "Signature": "<<JWT Token>>"
}

// Where the JWT Token is composed of 3 dot-separated,
// base-64 URL Encoded sections.
// e.g. eyJhbGcVCj9.eyJzdWIMzkwMjJ9.rrO9Qw0SXRWTe

// Section 1: Header/Algorithm
{
  "alg": "PS256",
  "typ": "JWT"
}

// Section 2: Payload
{
  "ProductCode" : "string",
  "PublicKeyVersion": 1,
```

```
"Nonce": "string",
"iat": date // JWT issued at claim
}

// Section 3: RSA-PSS SHA256 signature
"rrO9Q4FEi3gweH3X41rt2okf5zwIatUUwERlw016wTy_21Nv8S..."
```

7. Rebuild a new version of your Docker container image that includes the `RegisterUsage` call, tag the container, and push it to any Docker registry that is compatible with Amazon ECS or Amazon EKS, such as Amazon ECR or Docker Hub. If you are using Amazon ECR, ensure that the account launching the Amazon ECS task or Amazon EKS pod has permissions on the Amazon ECR repository. Otherwise, execution fails.

Note

If you use a private Docker Hub repository, follow the steps in [Private Registry Authentication for Tasks](#) in the *Amazon Elastic Container Service Developer Guide*.

8. Create an IAM role that grants permission for your container to call `RegisterUsage`, as defined in the following code. You must supply this IAM role in the `Task Role` parameter of the Amazon ECS task or Amazon EKS pod definition.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "aws-marketplace:RegisterUsage"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

9. Create an Amazon ECS task or Amazon EKS pod definition that references the container that has integrated with AWS Marketplace and references the IAM role that you created in step 7. You should enable AWS CloudTrail logging in the task definition if you want to see logging.
10. Create an Amazon ECS or Amazon EKS cluster to execute your task or pod. For more information about creating an Amazon ECS cluster, see [Creating a Cluster](#) in the *Amazon Elastic Container Service Developer Guide*. For more information about creating an Amazon EKS cluster (using Kubernetes version 1.1.3.x or later), see [Creating an Amazon EKS Cluster](#).
11. Configure the Amazon ECS or Amazon EKS cluster and launch the Amazon ECS task definition or Amazon EKS pod that you created, in the us-east-1 AWS Region. It's only during this testing process, before the product is live, that you have to use this region.
12. When you get a valid response back from `RegisterUsage`, you can begin creating your container product. For questions, contact the [AWS Marketplace Seller Operations](#) team.

RegisterUsage Java example

The following example uses the AWS SDK for Java and AWS Marketplace Metering Service to call the `RegisterUsage` operation. Signature verification is optional, but if you want to perform signature verification, you must include the required digital signature verification libraries. This example is for illustrative purposes only.

```
import com.amazonaws.auth.PEM;
import com.amazonaws.services.marketplacemetering.AWSMarketplaceMetering;
import com.amazonaws.services.marketplacemetering.AWSMarketplaceMeteringClientBuilder;
import com.amazonaws.services.marketplacemetering.model.RegisterUsageRequest;
import com.amazonaws.services.marketplacemetering.model.RegisterUsageResult;
```

```
import com.amazonaws.util.json.Jackson;
import com.fasterxml.jackson.databind.JsonNode;
import com.nimbusds.jose.JWSObject;
import com.nimbusds.jose.JWSVerifier;
import com.nimbusds.jose.crypto.RSASSAVerifier;
import java.io.ByteArrayInputStream;
import java.nio.charset.StandardCharsets;
import java.security.PublicKey;
import java.security.Security;
import java.security.Signature;
import java.security.interfaces.RSAPublicKey;
import java.util.Base64;
import java.util.Optional;
import java.util.UUID;
import org.bouncycastle.jce.provider.BouncyCastleProvider;

/**
 * Class for making calls out to AWS Marketplace Metering Service.
 */
class RegisterUsage {

    private static final String PRODUCT_CODE = ".....";

    private final AWSMarketplaceMetering registerUsageClient;
    private final SignatureVerifier signatureVerifier;
    private final int publicKeyVersion;

    public RegisterUsage(final SignatureVerifier signatureVerifier) {
        this.signatureVerifier = signatureVerifier;
        this.publicKeyVersion = PublicKeyProvider.PUBLIC_KEY_VERSION;
        this.registerUsageClient = AWSMarketplaceMeteringClientBuilder.standard().build();
    }

    /**
     * Shows how to call RegisterUsage client and verify digital signature.
     */
    public void callRegisterUsage() {
        RegisterUsageRequest request = new RegisterUsageRequest()
            .withProductCode(PRODUCT_CODE)
            .withPublicKeyVersion(publicKeyVersion)
            .withNonce(UUID.randomUUID().toString());

        // Execute call to RegisterUsage (only need to call once at container startup)
        RegisterUsageResult result = this.registerUsageClient.registerUsage(request);

        // Verify Digital Signature w/o JWT
        boolean isSignatureValid = this.signatureVerifier.verify(request, result);
        if (!isSignatureValid) {
            throw new RuntimeException("Revoke entitlement, digital signature invalid.");
        }
    }
}

/**
 * Signature verification class with both a JWT-library based verification
 * and a non-library based implementation.
 */
class SignatureVerifier {
    private static BouncyCastleProvider BC = new BouncyCastleProvider();

    private static final String SIGNATURE_ALGORITHM = "SHA256withRSA/PSS";

    private final PublicKey publicKey;

    public SignatureVerifier(PublicKeyProvider publicKeyProvider) {
        this.publicKey = publicKeyProvider.getPublicKey().orElse(null);
    }
}
```

```
        Security.addProvider(BC);
    }

    /**
     * Example signature verification using the NimbusJOSEJWT library to verify the JWT
     * Token.
     *
     * @param request RegisterUsage Request.
     * @param result RegisterUsage Result.
     * @return true if the token matches.
     */
    public boolean verifyUsingNimbusJOSEJWT(final RegisterUsageRequest request, final
RegisterUsageResult result) {
        if (!getPublicKey().isPresent()) {
            return false;
        }

        try {
            JWSVerifier verifier = new RSASSAVerifier((RSAPublicKey) getPublicKey().get());
            JWSPayload jwsObject = JWSPayload.parse(result.getSignature());
            return jwsObject.verify(verifier) &&
                validatePayload(jwsObject.getPayload().toString(), request, result);
        } catch (Exception e) {
            // log error
            return false;
        }
    }

    /**
     * Example signature verification without any JWT library support.
     *
     * @param request RegisterUsage Request.
     * @param result RegisterUsage Result.
     * @return true if the token matches.
     */
    public boolean verify(final RegisterUsageRequest request, final RegisterUsageResult
result) {
        if (!getPublicKey().isPresent()) {
            return false;
        }
        try {
            String[] jwtParts = result.getSignature().split("\\.");
            String header = jwtParts[0];
            String payload = jwtParts[1];
            String payloadSignature = jwtParts[2];

            Signature signature = Signature.getInstance(SIGNATURE_ALGORITHM, BC);
            signature.initVerify(getPublicKey().get());
            signature.update(String.format("%s.%s", header,
payload).getBytes(StandardCharsets.UTF_8));
            boolean verified = signature.verify(Base64.getUrlDecoder()
                .decode(payloadSignature.getBytes(StandardCharsets.UTF_8)));

            String decodedPayload = new String(Base64.getUrlDecoder().decode(payload));
            return verified && validatePayload(decodedPayload, request, result);
        } catch (Exception e) {
            // log error
            return false;
        }
    }

    /**
     * Validate each value in the returned payload matches values originally
     * supplied in the request to RegisterUsage. TimeToLiveInMillis and
     * PublicKeyExpirationTimestamp will have the values in the payload compared
     * to values in the signature
     */
}
```

```
    */
    private boolean validatePayload(final String payload, final RegisterUsageRequest
request,
                                final RegisterUsageResult result) {
        try {
            JsonNode payloadJson = Jackson.getMapper().readTree(payload);
            boolean matches = payloadJson.get("productCode")
                .asText()
                .equals(request.getProductCode());
            matches = matches && payloadJson.get("nonce")
                .asText()
                .equals(request.getNonce());
            return matches = matches && payloadJson.get("publicKeyVersion")
                .asText()
                .equals(String.valueOf(request.getPublicKeyVersion()));
        } catch (Exception ex) {
            // log error
            return false;
        }
    }

    private Optional<PublicKey> getPublicKey() {
        return Optional.ofNullable(this.publicKey);
    }
}

/**
 * Public key provider taking advantage of the AWS PEM Utility.
 */
class PublicKeyProvider {
    // Replace with your public key. Ensure there are new-lines ("\n") in the
    // string after "-----BEGIN PUBLIC KEY-----\n" and before "\n-----END PUBLIC KEY-----".
    private static final String PUBLIC_KEY =
        "-----BEGIN PUBLIC KEY-----\n"
        + "MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDdlatRjRjogo3WoJgGHFHYLugd\n"
        + "UWAY9iR3fy4arWNA1KoS8kVw33cJibXr8bvWUAUparCwlvdbH6dvEOfou0/gCFQs\n"
        + "HUFQrSDv+MuSUMAe8jzKE4qW+jK+xQU9a03GUNKHkkle+Q0pX/g6jXZ7r1/xAK5D\n"
        + "o2kQ+X5xK9cipRgEKwIDAQAB\n"
        + "-----END PUBLIC KEY-----";

    public static final int PUBLIC_KEY_VERSION = 1;

    public Optional<PublicKey> getPublicKey() {
        try {
            return Optional.of(PEM.readPublicKey(new ByteArrayInputStream(
                PUBLIC_KEY.getBytes(StandardCharsets.UTF_8))));
        } catch (Exception e) {
            // log error
            return Optional.empty();
        }
    }
}
}
```

Custom metering with AWS Marketplace Metering Service

AWS Marketplace container products can have custom metering on up to 24 different pricing dimensions per product. Each dimension can have a long-term contract price associated with it. To enable custom metering, integrate your container product with AWS Marketplace Metering Service. You can define your own pricing units and custom metering for that usage to AWS for billing using the [MeterUsage](#) API operation.

Price dimensions are defined in two locations, once in the product load form and once through the `MeterUsage` operation. This two-factor method ensures that the subsequent offers are working as intended before they're made available to the public.

To set up custom metering, you'll need to choose the usage category, the unit type, and pricing dimensions:

- **Usage category** – The usage category helps buyers understand what your product is and how to use it.
- **Unit type** – The unit type defines the unit of measure for billing. For example, bandwidth measured in GBps or MBps, the number of hosts, or data measured in MB, GB, or TB.
- **Pricing dimensions** – The pricing dimensions represents a feature or service that you've set a per-unit price for (for example, users, scans, vCPUs, or deployed agents). Pricing dimensions are public. However, you can still define private and Bring Your Own License (BYOL) offers for public products. Don't send pricing in the metering records. You meter the quantity of units, and we use that along with the prices you defined when creating your product to compute the buyer's bill.

If your product pricing doesn't fit with any of the predefined categories or unit types, you can choose the generic **Units** category. Then, use the dimension description to describe what the unit is.

Optionally, you may distribute the usage into allocations by properties that you track. The allocations are represented as tags to the buyer. These tags allow the buyer to view their costs split into usage by tag values. For example, if you charge by the user, and users have a "Department" property, you could create usage allocations with tags that have a key of "Department", and one allocation per value. This does not change the price, dimensions, or the total usage that you report, but allows your customer to view their costs by categories appropriate to your product.

We recommend that you send a metering record every hour. However, you can aggregate usage over daily or monthly periods as well. If you experience an outage, you can aggregate buyer software use and send it in the following hours metering. You can't send more than one record per hour.

Important

Free trial and prepaid entitlement are tracked on an hourly level. As a result, sending these records in separately might lead to the buyer being overcharged.

Custom metering prerequisites

Before publishing the product, you must do the following:

1. Create a new container product in the AWS Marketplace Management Portal, and make a note of its product code.
2. Fill out the product load form with the necessary dimension information, and return it to us for processing.
3. Use an AWS Identity and Access Management (IAM) role for the task or pod running your application with the IAM permissions necessary to call `MeterUsage`. The IAM managed policy `AWSMarketplaceMeteringRegisterUsage` has these permissions.
4. (Optional) We recommend that you enable AWS CloudTrail logging in the task or pod definition if you want to see logging.
5. Make a test call to the `MeterUsage` API operation with a record for all of the pricing dimensions you define.

Product load form for custom metering

When filling out the product load form for custom metering, each product can have up to 24 dimensions. The dimensions are defined in the following fields:

- **Dimension Name** – The name used when your container application is sending metering records to the AWS Marketplace Metering Service. This name indicates which dimension your buyer will use. This name is visible in billing reports. After you set the name, you can't change it.
- **Dimension Description** – The buyer-facing description for the dimension. The description can't exceed 70 characters. After the product is published publicly to buyers, this field can't be changed.
- **Dimension Rate** – The software price per unit for this product when buyers pay as they go. This field supports three decimal places.
- **Dimension Long Term Rate** – The total software price over a long-term contract when buyers pay upfront.
- **Long Term Duration (Days)** – The duration, in days, for the long-term contract.

Testing MeterUsage integration and preview mode

Use the `MeterUsage` operation to test your integration before submitting your image to AWS Marketplace for publishing.

Preview mode operates identically to production mode, except preview mode does not verify entitlement to use your product. To call `MeterUsage` in preview mode, call `MeterUsage` from the container images by running your product on Amazon Elastic Container Service (Amazon ECS) or Amazon Elastic Kubernetes Service (Amazon EKS) with the AWS account you are using to list the product on AWS Marketplace. Your metering integration must dynamically set the AWS Region, rather than hard coding it. However, when testing, launch at least one Amazon ECS task or Amazon EKS pod containing your paid container in the US East (N. Virginia) Region so that the AWS Marketplace operations team can verify your work with the logs in that Region.

Note

If your product supports both Amazon ECS and Amazon EKS, you only need to launch in Amazon EKS for us to validate your integration.

You can't fully test the integration until your product is published with all the required metadata and pricing information. If requested, the AWS Marketplace catalog operations team can verify receipt of your metering records in preview mode.

Error handling for MeterUsage

If your container image integrates with the `MeterUsage` operation and receives an exception other than `ThrottlingException` at container startup, you should terminate the container to prevent unauthorized use.

Exceptions other than `ThrottlingException` are thrown only on the initial call to `MeterUsage`. Subsequent calls from the same Amazon ECS task or Amazon EKS pod do not throw `CustomerNotSubscribedException` even if the customer unsubscribes while the task or pod is still running. These customers are still charged for running containers after they unsubscribe and their usage is tracked.

The following table describes the errors that `MeterUsage` might throw. Each AWS SDK programming language has a set of error handling guidelines that you can refer to for additional information.

| Error | Description |
|--|---|
| <code>DuplicateRequestException</code> | A metering record has already been emitted for the given <code>{usageDimension, timestamp}</code> with a different <code>usageQuantity</code> . |

| Error | Description |
|---|---|
| <code>InvalidUsageDimensionException</code> | The usage dimension does not match one of the <code>UsageDimensions</code> associated with the product. |
| <code>TimestampOutOfBoundsException</code> | The timestamp value passed in the <code>MeterUsage</code> is out of allowed range. |
| <code>InternalServiceErrorException</code> | <code>MeterUsage</code> isn't available. |
| <code>CustomerNotEntitledException</code> | The customer doesn't have a valid subscription for the product. |
| <code>InvalidProductCodeException</code> | The <code>ProductCode</code> value passed in as part of the request doesn't exist. |
| <code>PlatformNotSupportedException</code> | AWS Marketplace doesn't support metering usage from the underlying platform. Only Amazon ECS, Amazon EKS, and AWS Fargate are supported. |
| <code>ThrottlingException</code> | The calls to <code>MeterUsage</code> are throttled. |
| <code>InvalidEndpointRegionException</code> | <code>MeterUsage</code> must be called in the same AWS Region that the Amazon ECS task or Amazon EKS pod was launched in. This prevents a container from choosing a Region (for example, <code>withRegion("us-east-1")</code>) when calling <code>MeterUsage</code> . |
| <code>InvalidTagException</code> | The tags in the <code>UsageAllocations</code> object must have unique keys within each <code>UsageAllocation</code> , have no key-value pairs the same across <code>UsageAllocation</code> objects, and the number of tags must be 5 or less per <code>UsageAllocation</code> . |
| <code>InvalidUsageAllocationsException</code> | The <code>UsageAllocation</code> object allocations do not add up to the <code>UsageQuantity</code> value. |

Vendor-metered tagging (Optional)

Vendor-metered tagging helps Independent Software Vendors (ISVs) give the buyer more granular insight into their software usage and can help them perform cost allocation.

There are many ways to tag a buyer's software usage. One way is to first ask your buyers what they want to see in their cost allocation. Then you can split the usage across properties that you track for the buyer's account. Examples of properties include `Account ID`, `Business Unit`, `Cost Centers`, and other relevant metadata for your product. These properties are exposed to the buyer as tags. Using tags, buyers can view their costs split into usage by the tag values in their AWS Billing Console (<https://console.aws.amazon.com/billing/>). Vendor-metered tagging doesn't change the price, dimensions, or the total usage that you report. It allows your customer to view their costs by categories appropriate to your product.

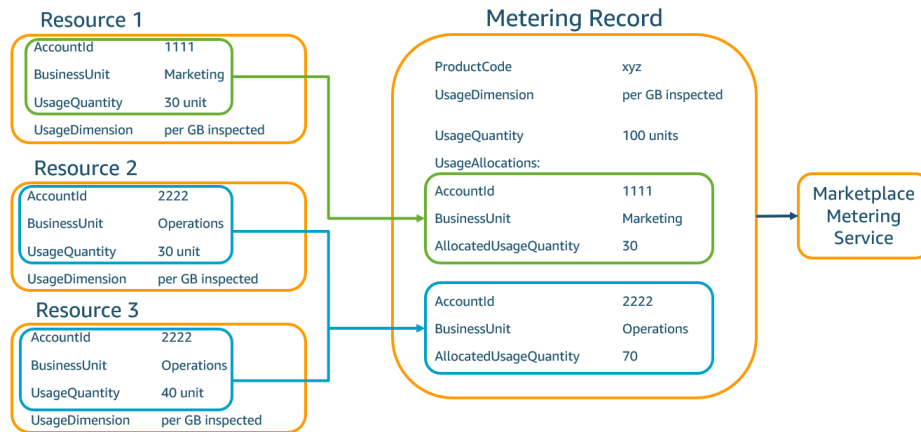
In a common use case, a buyer subscribes to your product with one AWS account. The buyer also has numerous user accounts associated with the same product subscription. You can create usage allocations with tags that have a key of `Account ID`, and then allocate usage to each user account. In this case, buyers can activate the `Account ID` tag in their Billing and Cost Management console and analyze individual user account usage.

Seller experience

Sellers can aggregate the metering records for resources with the same set of tags instead of aggregating usage for all resources. For example, sellers can construct the metering record that includes different buckets of `UsageAllocations`. Each bucket represents `UsageQuantity` for a set of tags, such as `AccountId` and `BusinessUnit`.

In the following diagram, **Resource 1** has a unique set of `AccountId` and `BusinessUnit` tags, and appears in the **Metering Record** as a single entry.

Resource 2 and **Resource 3** both have the same `AccountId` tag, 2222, and the same `BusinessUnit` tag, `Operations`. As a result, they're combined into a single `UsageAllocations` entry in the **Metering Record**.



Buyer experience

The following table shows an example of the buyer experience after a buyer activates the `AccountId` and `BusinessUnit` vendor tags.

In this example, the buyer can see allocated usage in their **Cost Usage Report**. The vendor-metered tags use the prefix `aws:marketplace:isv`. Buyers can activate them in the Billing and Cost Management, under **Cost Allocation Tags, AWS-generated cost allocation tags**.

The first and last rows of the **Cost Usage Report** are relevant to what the Seller sends to the Metering Service (as shown in the [Seller experience](#) (p. 147) example).

Cost Usage Report (Simplified)

| ProductCode | Buyer | UsageDimension | UsageQuantity | aws:marketplace:isv:AccountId | aws:marketplace:isv:BusinessUnit |
|-------------|--------------|-----------------------------|---------------|-------------------------------|----------------------------------|
| xyz | 111122223333 | Network: per (GB) inspected | 70 | 2222 | Operations |
| xyz | 111122223333 | Network: per (GB) inspected | 30 | 3333 | Finance |
| xyz | 111122223333 | Network: per (GB) inspected | 20 | 4444 | IT |
| xyz | 111122223333 | Network: per (GB) inspected | 20 | 5555 | Marketing |
| xyz | 111122223333 | Network: per (GB) inspected | 30 | 1111 | Marketing |

For a code example, see [MeterUsage code example with usage allocation tagging \(Optional\)](#) (p. 148).

Code example

The following code example is provided to help you integrate your container product with the AWS Marketplace APIs required for publishing and maintaining your product.

MeterUsage code example with usage allocation tagging (Optional)

The following code example is relevant for container products with consumption pricing models. The Python example sends a metering record with appropriate usage allocation tags to AWS Marketplace to charge your customers for pay-as-you-go fees.

```
# NOTE: Your application will need to aggregate usage for the
#       customer for the hour and set the quantity as seen below.
#       AWS Marketplace can only accept records for up to an hour in the past.
#
# productCode is supplied after the AWS Marketplace Ops team has
# published the product to limited

# Import AWS Python SDK
import boto3
import time

usageRecord = [
    {
        "AllocatedUsageQuantity": 2,
        "Tags":
            [
                { "Key": "BusinessUnit", "Value": "IT" },
                { "Key": "AccountId", "Value": "123456789" },
            ]
    },
    {
        "AllocatedUsageQuantity": 1,
        "Tags":
            [
                { "Key": "BusinessUnit", "Value": "Finance" },
                { "Key": "AccountId", "Value": "987654321" },
            ]
    }
]

marketplaceClient = boto3.client("meteringmarketplace")

response = marketplaceClient.meter_usage(
    ProductCode="testProduct",
    Timestamp=int(time.time()),
    UsageDimension="Dimension1",
    UsageQuantity=3,
    DryRun=False,
    UsageAllocations=usageRecord
)
```

For more information about MeterUsage, see [MeterUsage](#) in the *AWS Marketplace Metering Service API Reference*.

Example response

```
{ "MeteringRecordId": "string" }
```

Integrating your container product with the AWS Marketplace Metering Service using the AWS SDK for Java

The following example outlines an implementation that uses the AWS SDK for Java to integrate with the [AWS Marketplace Metering Service MeterUsage](#) operation. For complete details, see [MeterUsage Java examples \(p. 150\)](#). Many of the following steps apply regardless of the language.

Example: AWS Marketplace Metering Service integration

1. Sign in to the [AWS Marketplace Management Portal](#).
2. From **Assets**, choose **Containers** to start creating a new container product. Creating the product generates the product code for the product to integrate with your container image. For more information about publishing, see [Publishing container products \(p. 126\)](#). For information about setting AWS Identity and Access Management (IAM) permissions, see [the section called "AWS Marketplace metering and entitlement API permissions" \(p. 358\)](#).
3. Download the public [AWS Java SDK](#).

Important

To call the metering API operations from Amazon Elastic Kubernetes Service (Amazon EKS), you must [use a supported AWS SDK](#) and run on an Amazon EKS cluster running Kubernetes 1.13 or later.

4. Call the `MeterUsage` operation from the task or pod once every hour for each dimension usage. The API operation accepts one metering record for a unique combination of `Dimension`, `Resource`, and `Hour`. The resource is either an Amazon Elastic Container Service (Amazon ECS) task or an Amazon EKS pod.

```
{
  "ProductCode" : "string", // (required)
  "UsageDimension" : "string", // (required)
  "UsageQuantity": int, // (optional) Default is 0. Acceptable value from [0,
2147483647 (INT_MAX)]
  "Timestamp": Date, // (required) Timestamp in UTC. Value can be one hour in the
past.
  "UsageAllocations": List<UsageAllocation> // (optional) UsageAllocations across 1
or more tags.
}
```

Note

It is possible to see transient issues in connecting to the AWS Marketplace Metering Service. AWS Marketplace strongly recommends implementing retries for up to 30 minutes, with exponential back off, to avoid short-term outages or network issues.

5. Rebuild a new version of your Docker container image that includes the `MeterUsage` call, tag the container, and push it to any Docker registry that is compatible with Amazon ECS or Amazon EKS, such as Amazon Elastic Container Registry (Amazon ECR) or Docker Hub. If you are using Amazon ECR, ensure that the account launching the Amazon ECS task or Amazon EKS pod has permissions on the Amazon ECR repository. Otherwise, the operation fails.

Note

If you use a private Docker Hub repository, follow the steps in [Private registry authentication for tasks](#) in the *Amazon Elastic Container Service Developer Guide*.

6. Create an [IAM](#) role that grants permission for your container to call `MeterUsage`, as defined in the following code example. You must supply this AWS Identity and Access Management (IAM) role in the `TaskRole` parameter of the Amazon ECS task or Amazon EKS pod definition.

```
{
  "Version": "2012-10-17",
```

```
"Statement": [
  {
    "Action": [
      "aws-marketplace:MeterUsage"
    ],
    "Effect": "Allow",
    "Resource": "*"
  }
]
```

7. Create an Amazon ECS task or Amazon EKS pod definition that references the container that has integrated with AWS Marketplace and references the IAM role that you created in step 6. If you want to see logging, enable AWS CloudTrail logging in the task definition.
8. Create an Amazon ECS or Amazon EKS cluster to run your task or pod. For more information about creating an Amazon ECS cluster, see [Creating a cluster](#) in the *Amazon Elastic Container Service Developer Guide*. For more information about creating an Amazon EKS cluster (using Kubernetes version 1.13.x or later), see [Creating an Amazon EKS Cluster](#).
9. Configure the Amazon ECS or Amazon EKS cluster and launch the Amazon ECS task definition or Amazon EKS pod that you created in step 8, in the us-east-1 AWS Region. It's only during this testing process, before the product is live, that you have to use this Region.
10. When you get a valid response from `MeterUsage` for each of the dimensions being published for the product, you can begin creating your container product. For questions, contact the [AWS Marketplace Seller Operations](#) team.

MeterUsage Java examples

The following code examples use the AWS SDK for Java and AWS Marketplace Metering Service to call the `MeterUsage` operation.

The following code example calls the `MeterUsage` operation without any `UsageAllocations`.

```
import com.amazonaws.services.marketplacemetering.AWSMarketplaceMetering;
import com.amazonaws.services.marketplacemetering.AWSMarketplaceMeteringClientBuilder;
import com.amazonaws.services.marketplacemetering.model.MeterUsageRequest;
import com.amazonaws.services.marketplacemetering.model.MeterUsageResult;

import java.util.Date;

public class MeterUsage {
    private static final String PRODUCT_CODE = ".....";
    private final AWSMarketplaceMetering awsMarketplaceMetering;

    public MeterUsage() {
        awsMarketplaceMetering = AWSMarketplaceMeteringClientBuilder.standard().build();
    }

    /**
     * Submits metering record for a FCP Dimension. The API accepts 1 metering record per
     * dimension
     * for a given buyer's resource for a given timestamp hour. Ex. If a buyer is running
     * 10 tasks,
     * the API will accepts 1 call to MeterUsage in an hour for a given dimension for each
     * running task.
     *
     * @param dimension - FCP dimension name provided during the publishing of the product.
     * @param quantity - FCP dimension consumption value for the hour.
     * @param timestamp - Timestamp, in UTC, for which the usage is being reported.
     * Timestamp cant be more than 1 hour in the past.
     * Make sure the timestamp value is not before the start of the
     * software usage.
     */
}
```

```
*/
public void callMeterUsage(String dimension, int quantity, Date timestamp) {
    MeterUsageRequest meterUsageRequest = new MeterUsageRequest()
        .withProductCode(PRODUCT_CODE)
        .withUsageDimension(dimension)
        .withUsageQuantity(quantity)
        .withTimestamp(timestamp);
    MeterUsageResult meterUsageResult =
awsMarketplaceMetering.meterUsage(meterUsageRequest);
}
}
```

The following code example calls the `MeterUsage` operation with `UsageAllocations`.

```
private static String callMeterUsageWithAllocationsByTag(AWSMarketplaceMetering
marketplaceMetering) {
    // Tag Keys for the product
    String tagKey1 = "Key1";
    String tagKey2 = "Key2";
    String tagKey3 = "Key3";

    // 1st Usage Allocation bucket which has two Tags [{Key1, Key1Value1},{Key2,
Key2Value1}]
    List<Tag> tagsForUsageAllocation1 = Arrays.asList(new
Tag().withKey(tagKey1).withValue("Key1Value1"),
    new Tag().withKey(tagKey2).withValue("Key2Value1"));
    UsageAllocation usageAllocation1 = new UsageAllocation()
        .withTags(tagsForUsageAllocation1)
        .withAllocatedUsageQuantity(20);

    // 2nd Usage Allocation bucket which has two Tags [{Key1, Key1Value2},{Key2,
Key2Value1}]
    List<Tag> tagsForUsageAllocation2 = Arrays.asList(new
Tag().withKey(tagKey1).withValue("Key1Value2"),
    new Tag().withKey(tagKey2).withValue("Key2Value1"));
    UsageAllocation usageAllocation2 = new UsageAllocation()
        .withTags(tagsForUsageAllocation2)
        .withAllocatedUsageQuantity(20);

    // 3rd Usage Allocation bucket which has two Tags [{Key1, Key1Value2},{Key2,
Key2Value2},{Key3, Key3Value1}]
    List<Tag> tagsForUsageAllocation3 = Arrays.asList(new
Tag().withKey(tagKey1).withValue("Key1Value2"),
    new Tag().withKey(tagKey2).withValue("Key2Value2"),
    new Tag().withKey(tagKey3).withValue("Key3Value1"));
    UsageAllocation usageAllocation3 = new UsageAllocation()
        .withTags(tagsForUsageAllocation3)
        .withAllocatedUsageQuantity(15);

    // 4th Usage Allocation bucket with no tags
    UsageAllocation usageAllocation4 = new UsageAllocation()
        .withAllocatedUsageQuantity(15);

    List<UsageAllocation> usageAllocationList = Arrays.asList(usageAllocation1,
        usageAllocation2,
        usageAllocation3,
        usageAllocation4);

    MeterUsageRequest meterUsageRequest = new MeterUsageRequest()
        .withProductCode("TestProductCode")
        .withUsageDimension("Dimension1")
        .withTimestamp(new Date())
        //UsageQuantity value must match with sum of all AllocatedUsageQuantity
        .withUsageQuantity(70)
        .withUsageAllocations(usageAllocationList);
}
```



```
MeterUsageResult meterUsageResult;
try {
    meterUsageResult = marketplaceMetering.meterUsage(meterUsageRequest);
} catch (Exception e) {
    // Log Error
    throw e;
}

return meterUsageResult.getMeteringRecordId();
}
```

Contract pricing with AWS License Manager

For container-based products with contract pricing, use AWS License Manager to associate licenses with your product.

AWS License Manager is a license management tool that enables your application to track and update licenses (also known as entitlements) that have been purchased by a customer. This section provides information about how to integrate your product with AWS License Manager. After the integration is complete, you can publish your product listing on AWS Marketplace.

If you're integrating License Manager with an AWS Marketplace for Containers Anywhere product for Amazon EKS Anywhere, Amazon ECS Anywhere, Amazon Elastic Compute Cloud (Amazon EC2), or on-premises infrastructure, follow the instructions in [Integrating an AWS Marketplace for Containers Anywhere product with License Manager](#) (p. 157).

For more information about AWS License Manager, see the [AWS License Manager User Guide](#) and the [AWS License Manager](#) section of the *AWS CLI Command Reference*.

License models

AWS Marketplace integration with AWS License Manager supports two license models:

- [Configurable license model](#) (p. 152)
- [Tiered license model](#) (p. 155)

Configurable license model

The configurable license model (also known as the quantifiable license model) entitles a buyer to a specific quantity of resources after a buyer has procured a license.

You set a pricing dimension and a per unit price. Then, the buyer can choose the quantity of the resources that they want to purchase.

Example of pricing dimension and per unit price

You can set a pricing dimension (such as data backup) and per unit price (such as \$30/unit).

The buyer can choose to purchase 5, 10, or 20 units.

Your product tracks and meters usage to measure the quantity of resources consumed.

With the configuration model, the entitlements are counted in one of two ways:

- [Drawdown licenses](#) (p. 153)

- [Floating licenses \(p. 153\)](#)

Drawdown license

The license is drawn from the pool of allowed amount of licenses upon use. That entitlement is checked out permanently and can't be returned to the license pool.

Example of processing a limited amount of data

A user is entitled to process 500 GB of data. As they continue to process data, the quantity is drawn from the pool of 500 GB until all 500 GB licenses are consumed.

For drawdown licenses, you can use the `CheckoutLicense` API operation to check out license units (entitlements) that are consumed.

Example of backup to S3 for a number of units/year

You have a storage product that allows backup to Amazon Simple Storage Service (Amazon S3) for up to 1,024 units for data for one year. Your application can be launched by using multiple Amazon EC2 instances. Your application has a mechanism to track and aggregate data. Your software calls the `CheckoutLicense` API operation with the Product ID upon every backup or at fixed intervals to update the consumed quantities.

In this example, your software calls the `CheckoutLicense` API operation to check out 10 units of data. When the total capacity reaches the backup limit that the customer has purchased, the API call fails.

Request

```
linux-machine ~]$ aws license-manager checkout-license\  
--product-sku "2205b290-19e6-4c76-9eea-377d6bf71a47" \  
--checkout-type "PERPETUAL" \  
--key-fingerprint "aws:294406891311:AWS/Marketplace:issuer-fingerprint" \  
--entitlements "Name=DataConsumption, Value=10, Unit=Count" \  
--client-token "AKIAIOSFODNN7EXAMPLE"
```

Response

```
{  
  "CheckoutType": "PERPETUAL",  
  "EntitlementsAllowed": [  
    {  
      "Name": "DataConsumption",  
      "Count": 10,  
      "Units": "Count",  
      "Value": "Enabled"  
    }  
  ],  
  "Expiration": "2021-04-22T19:02:36",  
  "IssuedAt": "2021-04-22T18:02:36",  
  "LicenseArn": "arn:aws:license-manager::294406891311:license:l-16bf01b...",  
  "LicenseConsumptionToken": "AKIAIOSFODNN7EXAMPLE"  
}
```

Floating licenses

The license is returned to the pool of the allowed amount of licenses after use.

For floating licenses, the application checks out entitlements from the entitlements pool using the `CheckoutLicense` API operation when the resource is being used. The response of the

`CheckoutLicense` API operation includes a license consumption token which is a unique identifier for the checkout. The license consumption token can be used to perform additional actions on the entitlements checked out, such as checking them back into the license or extending the checkout.

To check the entitlement back into the pool, use the `CheckInLicense` API operation when the resource is no longer in use.

```
aws license-manager check-in-license --license-consumption-token
"f1603b3c1f574b7284db84..."
```

In case of failure to check in the entitlement (in case the application crashed), the entitlement checks back into the pool automatically after 60 minutes. If the resource is in use longer than 60 minutes, it is a best practice to keep the entitlement checked out of the pool by using the `ExtendLicenseConsumption` API operation as long as the resource is being used.

```
aws license-manager extend-license-consumption --license-consumption-token
"f1603b3c1f574b7284..."
```

Example of number of users from a fixed upper limit

A user is entitled to 500 simultaneous users on the application. As users log in and log out, the users are drawn and returned to the pool of 500 users. However, the application can't draw more than 500 users from the pool because 500 simultaneous users is the fixed upper limit.

For floating entitlements, you can use the `CheckInLicense` API operation to return the license units to the entitlement pool.

Example of number of concurrent users for one year

Your product is priced based on number of concurrent users. The customer purchases a license for 10 users for one year. The customer launches the software by providing AWS Identity and Access Management (IAM) permissions. When a user logs in, your application calls the `CheckoutLicense` API operation to reduce the quantity by 1. When the user logs out, the application returns that license to the pool by calling the `CheckInLicense` API operation. If you don't call `CheckInLicense`, the license unit will be automatically checked in after 1 hour.

Note

In the following Request, the `key-fingerprint` isn't a placeholder value but the actual value of the fingerprint with which all licenses will be published.

Request

```
aws license-manager checkout-license \
--product-sku "2205b290-19e6-4c76-9eea-377d6bf71a47" \
--checkout-type "PROVISIONAL" \
--key-fingerprint "aws:294406891311:AWS/Marketplace:issuer-fingerprint" \
--entitlements "Name=ReadOnlyUsers, Value=10, Unit=Count" \
--client-token "AKIAIOSFODNN7EXAMPLE"
```

Response

```
{
  "CheckoutType": "PROVISIONAL",
  "EntitlementsAllowed": [
    {
      "Name": "ReadOnlyUsers",
```

```
    "Count": 10,  
    "Units": "Count",  
    "Value": "Enabled"  
  }  
},  
"Expiration": "2021-04-22T19:02:36",  
"IssuedAt": "2021-04-22T18:02:36",  
"LicenseArn": "arn:aws:license-manager::294406891311:license:l-16bf01b...",  
"LicenseConsumptionToken": "AKIAIOSFODNN7EXAMPLE"  
}
```

Tiered license model

The tiered license model entitles a buyer to a specific level, or tier, of application features after a buyer has procured a license.

You create tiers for your product, such as Basic, Intermediate, and Premium. The buyer then selects one of the predefined tiers.

The application doesn't need to track or meter usage of the application.

With the tiered license model, the entitlements aren't counted but instead signify a tier of service that was procured by the customer.

If you want to offer bundled features together, tiers are preferable.

Example of Basic, Intermediate, and Premium tiers

A customer can sign a contract for one of three possible tiers of the software: Basic, Intermediate, or Premium. Each of these tiers has its own pricing. Your software can identify the tier that the customer has signed up for by invoking the `CheckoutLicense` API operation and specifying all possible tiers in the request.

The response of the request contains the entitlement corresponding to the tier that the customer has procured. Based on this information, the software can provision the appropriate customer experience.

Request

```
linux-machine ~]$ aws license-manager checkout-license\  
--product-sku "2205b290-19e6-4c76-9eea-377d6bf71a47" \  
--checkout-type "PROVISIONAL" \  
--key-fingerprint "aws:294406891311:AWS/Marketplace:issuer-fingerprint" \  
--entitlements "Name=BasicTier, Unit=None" "Name=IntermediateTier, Unit=None" \  
"Name=PremiumTier, Unit=None"
```

Response

```
{  
  "CheckoutType": "PROVISIONAL",  
  "EntitlementsAllowed": [  
    {  
      "Name": "IntermediateTier",  
      "Units": "None"  
    }  
  ],  
  "Expiration": "2021-04-22T19:02:36",  
  "IssuedAt": "2021-04-22T18:02:36",  
  "LicenseArn": "arn:aws:license-manager::294406891311:license:l-16bf01b...",  
  "LicenseConsumptionToken": "AKIAIOSFODNN7EXAMPLE"  
}
```

```
}
```

AWS License Manager integration prerequisites

Before publishing the product, you must do the following:

1. Create a new container product in the AWS Marketplace Management Portal, and make a note of its product code.

For more information, see [Creating a container product \(p. 117\)](#).

2. Fill out the product load form (PLF) with the necessary price information, and return it to us for processing.

For more information, see [Creating or updating pricing details for container products \(p. 118\)](#).

3. Use an IAM role for the task or pod running your application with the IAM permissions necessary to call the `CheckoutLicense`, `ExtendLicenseConsumption`, and `CheckInLicense` API operations.

The required IAM permissions are detailed in the following IAM policy.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": [
        "license-manager:CheckoutLicense",
        "license-manager:GetLicense",
        "license-manager:CheckInLicense",
        "license-manager:ExtendLicenseConsumption",
        "license-manager:ListReceivedLicenses"
      ],
      "Resource": "*"
    }
  ]
}
```

4. Make a test call to the `RegisterUsage` API operation with a record for all of the pricing dimensions you define.

Integrating a container product with License Manager

To integrate your container-based product with License Manager

1. Set IAM permissions to call License Manager. For more information, see [AWS License Manager integration prerequisites \(p. 156\)](#).
2. Download the AWS SDK.

Note

Don't configure AWS credentials within your software. AWS credentials for the buyer are automatically obtained at runtime when your container is running within an Amazon EC2 instance, Amazon ECS task, or Amazon EKS pod.

3. Add license checks to your product.

Your product can call the `CheckoutLicense` API operation wherever the license check should be performed. To check the license, your product must know:

1. The trusted issuer of the license (AWS Marketplace)

2. The application's Product SKU (Product ID)
3. The entitlement to check for this application

The API calls vary based on what kind of pricing licenses you set up.

4. Publish your product listing on AWS Marketplace.

License Manager API operations

To manage the licenses stored in the customer's License Manager account, your software can use the following API operations:

- `GetLicense` – Gets the status of a purchased license on whether the license is expired or about to be expired to send a notification to the customer.
- `CheckoutLicense` – Discovers licenses that the user has purchased. You can also use the `CheckoutLicense` API operation to update the license quantity when the user has consumed some quantity of licenses. With `CheckoutLicense`, you can keep checking out the quantities of licenses used by the customer. When the customer exhausts all the licenses, this call returns an error. For information about the suggested cadence to run `CheckoutLicense`, see [the section called “License renewals and upgrades” \(p. 157\)](#).
- `ExtendLicenseConsumption` – In case of floating dimensions, when the software checks out a license, the license will return to the pool automatically after 60 minutes. If you want to extend the time the license remains checked out, use the `ExtendLicenseConsumption` API operation to extend the license for another 60 minutes.
- `CheckInLicense` – In case of floating dimensions, when you want to return the license to the entitlement pool, use the `CheckInLicense` API operation.
- `ListReceivedLicenses` API – Lists licenses purchased by the buyer.

License renewals and upgrades

Customers can renew or upgrade their licenses on the AWS Marketplace Management Portal. After they make an additional purchase, AWS Marketplace generates a new version of the license that reflects the new entitlements. Your software reads the new entitlements by using the same API operations. You don't have to do anything different in terms of License Manager integration to handle renewals and upgrades.

Due to license renewals, upgrades, cancellations, and so on, we recommend that your product calls the `CheckoutLicense` API operation at a regular cadence while the product is in use. By using the `CheckoutLicense` API operation at a regular cadence, the product can detect changes in entitlements such as upgrades and expiry.

We recommend that you perform the `CheckoutLicense` API call every 15 minutes.

Integrating an AWS Marketplace for Containers Anywhere product with License Manager

Follow these instructions to integrate AWS License Manager with an AWS Marketplace for Containers Anywhere product for Amazon EKS Anywhere, Amazon ECS Anywhere, Amazon EC2, or on-premises infrastructure.

For general information about the License Manager integration with AWS Marketplace, including available license models, see [Contract pricing with AWS License Manager \(p. 152\)](#). For more information about AWS License Manager, see the [AWS License Manager User Guide](#) and the [AWS License Manager](#) section of the *AWS CLI Command Reference*.

Integrating an AWS Marketplace for Containers Anywhere product with License Manager

Use the following instructions to integrate your AWS Marketplace for Containers Anywhere product with AWS License Manager.

To integrate your AWS Marketplace for Containers Anywhere product with License Manager

1. Open a web browser and sign into the [AWS Marketplace Management Portal](#).
2. Create a product ID for your container product by performing the following steps. You will use this ID in your container image for license checks in a later step.
 - a. From the menu bar, expand **Assets**, and choose **Container**.
 - b. Enter a customer-facing name for your product, and choose **Create**. You can change this name later.
 - c. Make a note of the **Product ID**. You will use it when you create or update the product pricing details.

Tip

If you lose your product ID, you can find it in the AWS Marketplace Management Portal by choosing **Container** from the **Assets** menu. The **Containers** page shows a list of your products with their associated product IDs.

3. Download the latest public AWS SDK and then include it in your container application. You can find installation instructions for your preferred AWS SDK at [Tools to Build on AWS](#).

Note

To call the License Manager API operations from Amazon EKS Anywhere or a Kubernetes cluster that isn't provided by AWS, you must use a supported AWS SDK. To view a list of supported AWS SDKs, see [Using a supported AWS SDK](#).

4. Create an AWS License Manager client with a custom credential provider so that it can provide credentials to the container application deployed on AWS as well as on-premises. For complete source code for a custom credential provider, `LicenseCredentialsProvider`, see the following sections:

- [LicenseManagerCredentialsProvider - Java implementation \(p. 166\)](#)
- [LicenseManagerCredentialsProvider - Golang implementation \(p. 171\)](#)

`LicenseCredentialsProvider` extends the AWS SDK's default credential provider chain for on-premises use by adding `LicenseManagerTokenCredentialsProvider`. This provides credentials by using License Manager OIDC issued identity tokens in on-premises environments. You must include the source code for `LicenseCredentialsProvider` in your application classpath.

Note

Extending the `DefaultCredentialsProvider` allows the same container application to obtain credentials when running on AWS and when running in an on-premises environment. If the container application already uses a custom credential provider chain instead of the default, it can also be extended by adding `LicenseManagerTokenCredentialsProvider` to the custom chain.

The following code snippet is an example of creating an AWS License Manager client using Java.

```
LicenseManagerClientBuilder clientBuilder =  
    LicenseManagerClient.builder().credentialsProvider(LicenseCredentialsProvider.create());
```

5. Call the `CheckoutLicense` API operation by using the `aws license-manager checkout-license` command from each paid container image in your product offering. This checks that the buyer is entitled to use a license for your application. If the buyer is entitled to the application,

`CheckLicense` succeeds and returns the requested entitlements and their values. If the buyer isn't entitled to the application, `CheckLicense` throws an exception.

The following parameters are required when calling the `CheckoutLicense` API operation:

- `CheckoutType` – The valid values are `PROVISIONAL` or `PERPETUAL`:
 - Use `PERPETUAL` when the quantity of entitlements checked out will be exhausted from the pool.

Example: Buyer is entitled to process 500 GB of data. As they continue to process data, the quantity is drawn down and exhausted from the pool of 500 GB.
 - Use `PROVISIONAL` for floating license entitlements where the entitlements are checked out of the pool and returned after use.

Example: User is entitled to 500 simultaneous users on the application. As users log in or log out, the users are drawn or returned to the pool of 500 users. To learn more about floating license entitlements, see [Floating license entitlements with License Manager \(p. 165\)](#).
- `ClientToken` – A unique, case-sensitive identifier. We recommend using a random UUID for each unique request.
- `Entitlements` – A list of entitlements to be checked out.
 - For feature entitlements, provide the `Name` and `Unit` properties as follows.

```
{
  "Name": "<Entitlement_Name>",
  "Unit": "None"
}
```

- For counted entitlements, provide the `Name`, `Unit`, and `Count` properties as follows.

```
{
  "Name": "<Entitlement_Name>",
  "Unit": "<Entitlement_Unit>",
  "Value": <Desired_Count>
}
```

- `KeyFingerprint` – The key fingerprint for licenses issued by AWS Marketplace is `aws:294406891311:AWS/Marketplace:issuer-fingerprint`. Using this key fingerprint ensures that the license is issued by AWS Marketplace and not by an unreliable entity.
- `ProductSKU` – The Product ID generated on AWS Marketplace Management Portal in previous steps.

The following snippet is an example of a call using the `CheckoutLicense` API operation using the AWS CLI.

```
aws license-manager checkout-license \
--product-sku "2205b290-19e6-4c76-9eea-377d6bf71a47" \
--checkout-type "PROVISIONAL" \
--client-token "79464194dca9429698cc774587a603a1" \
--entitlements "Name=AWS::Marketplace::Usage/Drawdown/DataConsumption, Value=10,
Unit=Gigabytes" \
--key-fingerprint "aws:294406891311:AWS/Marketplace:issuer-fingerprint"
```

Note

To check licenses, container applications require outbound network access to use License Manager. Applications deployed on-premises might experience unreliable or slow outbound network access. These applications should include adequate retries when calling License

Manager. For more information, see [Best practices for integrating with License Manager for on-premises deployments \(p. 165\)](#).

6. Call the `CheckoutLicense` API operation at a regular cadence to identify any changes to customers' licenses due to renewals, upgrades, or cancellations made on AWS Marketplace. The cadence depends on the application. We recommend checking licenses once a day to pick up changes automatically without any buyer intervention.

An application deployed on-premises might have unreliable outbound network access to check licenses on a regular cadence. In such cases, the application should use a cached licenses for sufficient resiliency. For more information, see [Best practices for integrating with License Manager for on-premises deployments \(p. 165\)](#).

7. After you integrate the `CheckoutLicense` call with your container application, build a new version of your Docker container image with the changes.
8. Update your application's Helm chart to accept a Kubernetes secret as optional input that contains configuration to access licenses using License Manager APIs. The configuration secret will contain an identity token issued by License Manager and an AWS Identity and Access Management role which will be used by the custom credential provider described previously to get AWS credentials for calling License Manager APIs when the container application is deployed on-premises. Also, add the AWS Region as an input with a default value of `us-east-1`.

Buyers deploying the container application on-premises can create the Kubernetes secret through the AWS Marketplace buyer experience for container products. The Kubernetes secret name can then be provided as input to the `helm install` command. The configuration secret can be expected in the following format.

```
apiVersion: v1
kind: Secret
metadata:
  name: aws-marketplace-license-config
type: Opaque
stringData:
  license_token: <token_value> // License Manager issued JWT token
  iam_role: <role_arn> // AWS Identity and Access Management role to assume with
  license token
```

9. Update the application deployment template in the Helm chart for container images integrated with AWS License Manager to include the following:
 - Service account for pod – The service account is required for Helm deployments on Amazon EKS. It's used to get permissions to call License Manager API operations by setting up IAM roles for the service account on the container image. For more information about IAM roles for service accounts, see [IAM roles for service accounts](#).
 - License access for on-premises deployments – The license configuration secret is required to provide credentials and appropriate permissions to call License Manager API operations for Helm deployments in on-premises environments. Buyers will generate and provide the license secret to Helm from the AWS Marketplace buyer experience.

The following code snippet is a sample deployment specification with the service account, license configuration, and image pull secret.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: example-app
spec:
  replicas: 1
  selector:
```

```
    matchLabels:
      app: example-app
  template:
    metadata:
      labels:
        app: example-app
  spec:
    // Service account for pod
    serviceAccountName: {{ .Values.serviceAccountName }}
    containers:
      - name: example-app
        image: example-app
        ports:
          - containerPort: 8001
    // Add the following conditional attributes
    {{ - if .Values.awsmpl.licenseConfigSecretName }}
    //Mount the license volume to the container image
    volumeMounts:
      - name: awsmpl-product-license
        mountPath: "/var/run/secrets/product-license"
    //Add following environment variable to container for credential
  provider
  env:
    - name: AWS_WEB_IDENTITY_REFRESH_TOKEN_FILE
      value: "/var/run/secrets/product-license/license_token"
    - name: AWS_ROLE_ARN
      valueFrom:
        secretKeyRef:
          name: {{ .Values.aws.licenseConfigSecretName }}
          key: iam_role
    //Mount the license secret as a volume to the pod
  volumes:
    - name: awsmpl-product-license
      secret:
        secretName: {{ .Values.aws.licenseConfigSecretName }}
        optional: true
  {{ - end }}
```

Note

The license configuration secret is optional. Buyers only use the value for on-premises deployments. For AWS deployments, the deployment specification should include a service account for the License Manager integrated images.

10. Test the License Manager integration locally and on Amazon EKS by performing the steps in the following sections:
 - a. [Testing License Manager integration locally \(p. 161\)](#)
 - b. [Testing License Manager integration on Amazon EKS \(p. 163\)](#)
11. After you successfully verify License Manager integration both on AWS and on-premises, you can create your container product listing by following the steps in [Creating a container product \(p. 117\)](#).

Testing License Manager integration locally

You can use minikube or any other setup to test License Manager integration on any Kubernetes cluster locally. Make sure that the Kubernetes cluster has outbound internet access to call License Manager API operations.

To test a License Manager integration locally

1. Create a test license in a test seller account with desired entitlements. To set up a test license, see [CreateLicense](#) in the *AWS License Manager API Reference*. Or, use the following script to create a test license and then create a license grant to a test buyer account to consume the license. The following script uses test seller account credentials.

```
read -p 'AWS Account for test buyer: ' TEST_BUYER_ACCOUNT_ID
read -p 'License entitlements: ' ENTITLEMENTS

# TEST_SELLER_ACCOUNT_ID="109876543210"
# ENTITLEMENTS="{\"Name\": \"ByData\", \"MaxCount\": 1000, \"Overage\": true, \"Unit\": \"Gigabits\", \"AllowCheckIn\": true}"

# Create License

NOW=$(date +%Y-%m-%dT00:00:00+00:00)

PRODUCT_NAME="My awesome product"
PRODUCT_SKU="c97b7825-44c4-4f42-b025-12baa4c171e0"

LICENSE_BENEFICIARY=" arn:aws:iam::$TEST_BUYER_ACCOUNT_ID:root "
LICENSE_ISSUER_NAME="test-seller"
LICENSE_NAME="test-seller-license"

CLIENT_TOKEN="b3920968-a94f-4547-af07-3dd232319367"
CONSUMPTION_TTL=180
CONSUMPTION_RENEW_TYPE="None"

HOME_REGION="us-east-1"

LICENSE_ARN=$(aws license-manager create-license --license-name
"$LICENSE_NAME" --product-name "$PRODUCT_NAME" --product-sku
"$PRODUCT_SKU" --issuer Name="$LICENSE_ISSUER_NAME" --home-region
"$HOME_REGION" --validity Begin="$NOW" --entitlements "$ENTITLEMENTS"
--beneficiary "$LICENSE_BENEFICIARY" --consumption-configuration
RenewType="$CONSUMPTION_RENEW_TYPE",ProvisionalConfiguration={MaxTimeToLiveInMinutes=
$CONSUMPTION_TTL} --client-token "$CLIENT_TOKEN" | jq -r ".LicenseArn" )

echo "License arn: $LICENSE_ARN"

# Create Grant

GRANT_TOKEN="e9a14140-4fca-4219-8230-57511a6ea6"
GRANT_NAME="test-grant"

GRANT_ARN=$(aws license-manager create-grant --grant-name "$GRANT_NAME" --license-arn
"$LICENSE_ARN" --principals "$LICENSE_BENEFICIARY" --home-region "$HOME_REGION" --
client-token "$GRANT_TOKEN" --allowed-operations "CheckoutLicense" "CheckInLicense"
"ExtendConsumptionLicense" "CreateToken" | jq -r ".GrantArn")

echo "Grant arn: $GRANT_ARN"
```

2. Create a K8 secret with the license token and IAM role using the secret format defined previously. Use the License Manager `CreateToken` API operation to generate a license token. Then, use the IAM `CreateRole` API operation to create an IAM role with permissions and a trust policy. See the example in the following script. The following script uses test buyer account credentials.

```
read -p 'AWS Account for test license: ' TEST_ACCOUNT_ID
read -p 'License Arn' LICENSE_ARN
# Create IAM Role
ROLE_NAME="AWSLicenseManagerConsumptionTestRole"
ROLE_DESCRIPTION="Role to test AWS License Manager integration on-prem"
```

```
ROLE_POLICY_ARN="arn:aws:iam::aws:policy/service-role/
AWSLicenseManagerConsumptionPolicy"
ROLE_TRUST_POLICY="{\"Version\": \"2012-10-17\", \"Statement\": [{ \"Effect\": \"Allow\",
  \"Principal\": { \"Federated\": \"openid-license-manager.amazonaws.com\" }, \"Action
\": \"sts:AssumeRoleWithWebIdentity\", \"Condition\": { \"ForAnyValue:StringLike\":
  { \"openid-license-manager.amazonaws.com:amr\": \"aws:license-manager:token-issuer-
account-id:${TEST_ACCOUNT_ID}\" } } ]}"
ROLE_SESSION_DURATION=3600

ROLE_ARN=$(aws iam create-role --role-name "$ROLE_NAME" --description
"$ROLE_DESCRIPTION" --assume-role-policy-document "$ROLE_TRUST_POLICY" --max-session-
duration $ROLE_SESSION_DURATION | jq ".Role" | jq -r ".Arn")

aws iam attach-role-policy --role-name "$ROLE_NAME" --policy-arn "$ROLE_POLICY_ARN"

echo "Role arn: $ROLE_ARN"

# Create Token
CLIENT_TOKEN="b3920968-a94f-4547-af07-3dd232319367"

TOKEN=$(aws license-manager create-token --license-arn $LICENSE_ARN --role-arns
$ROLE_ARN --client-token $CLIENT_TOKEN | jq '.Token')

echo "License access token: $TOKEN"c
```

3. Set up any Kubernetes cluster hosted outside AWS. Use it to test that the container applications can connect to the AWS License Manager API from environments other than AWS and that the custom credential provider is well integrated in the application.
4. Deploy the license token and IAM role generated previously into the local Kubernetes cluster.

```
kubectl create secret generic "awsmpl-license-access-config" \
--from-literal=license_token=${TOKEN} \
--from-literal=iam_role=${ROLE_ARN}
```

5. Deploy your application through Helm with the secret name as input and verify that the application can call License Manager API operations to perform entitlement checks. For Helm and deployment specification changes, refer to Step 9 in [Integrating an AWS Marketplace for Containers Anywhere product with License Manager \(p. 158\)](#).

Testing License Manager integration on Amazon EKS

You can also test License Manager integration on Amazon EKS. Test to make sure that the application can call License Manager API operations without the license configuration secret. Also make sure that the service account can be used to set up IRSA and provide relevant credentials to the application.

To test a License Manager integration on Amazon EKS

1. Create a test license in a test seller account with the desired entitlements. See [CreateLicense API reference](#) to set up your test license or use the following script to create one and create a license grant to a test buyer account to consume the license. The following script uses test seller account credentials.

```
read -p 'AWS Account for test buyer: ' TEST_BUYER_ACCOUNT_ID
read -p 'License entitlements: ' ENTITLEMENTS

# TEST_SELLER_ACCOUNT_ID="109876543210"
# ENTITLEMENTS="{\"Name\": \"ByData\", \"MaxCount\": 1000, \"Overage\": true, \"Unit\":
  \"Gigabits\", \"AllowCheckIn\": true}"

# Create License
```

```
NOW=$(date +%Y-%m-%dT00:00:00+00:00)

PRODUCT_NAME="My awesome product"
PRODUCT_SKU="c97b7825-44c4-4f42-b025-12baa4c171e0"

LICENSE_BENEFICIARY=" arn:aws:iam::$TEST_BUYER_ACCOUNT_ID:root "
LICENSE_ISSUER_NAME="test-seller"
LICENSE_NAME="test-seller-license"

CLIENT_TOKEN="b3920968-a94f-4547-af07-3dd232319367"
CONSUMPTION_TTL=180
CONSUMPTION_RENEW_TYPE="None"

HOME_REGION="us-east-1"

LICENSE_ARN=$(aws license-manager create-license --license-name
"$LICENSE_NAME" --product-name "$PRODUCT_NAME" --product-sku
"$PRODUCT_SKU" --issuer Name="$LICENSE_ISSUER_NAME" --home-region
"$HOME_REGION" --validity Begin="$NOW" --entitlements "$ENTITLEMENTS"
--beneficiary "$LICENSE_BENEFICIARY" --consumption-configuration
RenewType="$CONSUMPTION_RENEW_TYPE",ProvisionalConfiguration={MaxTimeToLiveInMinutes=
$CONSUMPTION_TTL} --client-token "$CLIENT_TOKEN" | jq -r ".LicenseArn" )

echo "License arn: $LICENSE_ARN"

# Create Grant

GRANT_TOKEN="e9a14140-4fca-4219-8230-57511a6ea6"
GRANT_NAME="test-grant"

GRANT_ARN=$(aws license-manager create-grant --grant-name "$GRANT_NAME" --license-arn
"$LICENSE_ARN" --principals "$LICENSE_BENEFICIARY" --home-region "$HOME_REGION" --
client-token "$GRANT_TOKEN" --allowed-operations "CheckoutLicense" "CheckInLicense"
"ExtendConsumptionLicense" "CreateToken" | jq -r ".GrantArn")

echo "Grant arn: $GRANT_ARN"
```

2. Create a test Amazon EKS cluster of desired configurations, or run the following commands to use a default configuration.

```
aws ec2 create-key-pair --region us-west-2 --key-name eks-key-pair
```

```
eksctl create cluster \
--name awsmpeks-test-example \
--region us-west-2 \
--with-oidc \
--ssh-access \
--ssh-public-key eks-key-pair
```

3. Create a service account for an existing cluster and associate it with an IAM role. The following command creates an IAM role with the `AWSLicenseManagerConsumptionPolicy`. Then, the command attaches it to the `test_sa` service account of the Amazon EKS cluster where the License Manager integrated images should be deployed. As a result, the service account can get appropriate credentials to call License Manager API operations.

```
eksctl create iamserviceaccount \
--name test_sa \
--namespace test_namespace \
--cluster awsmpeks-test-example \
--attach-policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLicenseManagerConsumptionPolicy" \
--approve \
```

```
--override-existing-serviceaccounts
```

4. Deploy the application through Helm in the service account where the IAM role is associated from the previous command. Verify that the application can call License Manager API operations to perform entitlement checks.

Floating license entitlements with License Manager

With floating licenses, as users log into the application, a license is drawn from the pool of available licenses. As users log out, the licenses are added back to the pool of available licenses.

For floating licenses, the application uses the `CheckoutLicense` API operation to check out entitlements from the entitlements pool when the resource is being used. The response of the `CheckoutLicense` API operation includes a license consumption token which is a unique identifier for the checkout. The license consumption token can perform additional actions on the entitlements that are checked out, such as checking them back into the license pool or extending the checkout.

When the resource is no longer in use, the application uses the `CheckInLicense` API operation to check the entitlement back into the pool.

```
aws license-manager check-in-license \  
--license-consumption-token "f1603b3c1f574b7284db84a9e771ee12"
```

If checking a license back into the pool fails, for example, if the application crashes during the operation, the entitlement is checked back into the pool automatically after 60 minutes. Because of this, if the resource is in use longer than 60 minutes, it's a best practice to keep the entitlement checked out of the pool. To do this, use the `ExtendLicenseConsumption` API operation as long as the resource is being used.

```
aws license-manager extend-license-consumption \  
--license-consumption-token "f1603b3c1f574b7284db84a9e771ee12"
```

Best practices for integrating with License Manager for on-premises deployments

Container application deployments in an on-premises environment might encounter unreliable outbound network access. Use the following best practices to add resiliency to avoid service disruption to buyers due to potential issues caused by poor internet connectivity:

- **Adequate retry** – Transient network issues can keep your application from connecting to AWS License Manager. Implement retries for up to 30 minutes, with exponential back off. This can help avoid short-term outages or network issues.
- **Avoid hard limit** – Applications deployed in connected clusters can regularly check licenses to identify any changes due to upgrades or renewals. With unreliable outbound access, the application might not be able to identify those changes. Whenever possible, the application should avoid disruption of service to buyers due to inability to check licenses through License Manager. Applications can fall back on a free-trial or open-source experience when the license expires and they can't check if a license is valid.
- **Notify customers** – When using a cached license, any changes to the license (including renewal or upgrades) are not automatically reflected on the running workload. Notify your customers (that they must allow outbound access to the application again temporarily so the application can update its cached license. For example, notify customers through the application itself or through its documentation. Similarly, when falling back to a lower set of functionalities, notify customers that their entitlements are exhausted or the license is expired. Then, they can choose to either upgrade or renew.

LicenseManagerCredentialsProvider - Java implementation

LicenseManagerCredentialsProvider extends the AWS SDK's default credential provider chain for on-premises use by adding LicenseManagerTokenCredentialsProvider.

LicenseManagerCredentialsProvider

```
package com.amazon.awsmp.license;

import software.amazon.awssdk.auth.credentials.AwsCredentials;
import software.amazon.awssdk.auth.credentials.AwsCredentialsProvider;
import software.amazon.awssdk.auth.credentials.AwsCredentialsProviderChain;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.auth.credentials.internal.LazyAwsCredentialsProvider;
import software.amazon.awssdk.utils.SdkAutoCloseable;

public class LicenseManagerCredentialsProvider implements AwsCredentialsProvider, SdkAutoCloseable
{
    private static final LicenseManagerCredentialsProvider CREDENTIALS_PROVIDER = new
    LicenseManagerCredentialsProvider();
    private final LazyAwsCredentialsProvider providerChain;

    private LicenseManagerCredentialsProvider() {
        this.providerChain = createChain();
    }

    public static LicenseManagerCredentialsProvider create() {
        return CREDENTIALS_PROVIDER;
    }

    @Override
    public AwsCredentials resolveCredentials() {
        return this.providerChain.resolveCredentials();
    }

    @Override
    public void close() {
        this.providerChain.close();
    }

    private LazyAwsCredentialsProvider createChain() {
        return LazyAwsCredentialsProvider.create(() -> {
            AwsCredentialsProvider[] credentialsProviders = new AwsCredentialsProvider[] {
                DefaultCredentialsProvider.create(),
                LicenseManagerTokenCredentialsProvider.create();
            };

            return AwsCredentialsProviderChain.builder().reuseLastProviderEnabled(true)
                .credentialsProviders(credentialsProviders).build();
        });
    }
}
```

LicenseManagerTokenCredentialsProvider

LicenseManagerTokenCredentialsProvider provides credentials by using License Manager OIDC issued identity tokens in on-premises environments. You must include the source code for LicenseManagerCredentialsProvider in your application classpath.

```
package com.amazon.awsmp.license;

import software.amazon.awssdk.auth.credentials.AnonymousCredentialsProvider;
import software.amazon.awssdk.auth.credentials.AwsCredentials;
```

```
import software.amazon.awssdk.auth.credentials.AwsCredentialsProvider;
import software.amazon.awssdk.core.SdkSystemSetting;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryPolicyContext;
import software.amazon.awssdk.core.retry.conditions.OrRetryCondition;
import software.amazon.awssdk.core.retry.conditions.RetryCondition;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.regions.providers.DefaultAwsRegionProviderChain;
import software.amazon.awssdk.services.licensemanager.LicenseManagerClient;
import software.amazon.awssdk.services.licensemanager.model.GetAccessTokenRequest;
import software.amazon.awssdk.services.licensemanager.model.GetAccessTokenResponse;
import software.amazon.awssdk.services.sts.StsClient;
import
    software.amazon.awssdk.services.sts.auth.StsAssumeRoleWithWebIdentityCredentialsProvider;
import software.amazon.awssdk.services.sts.model.AssumeRoleWithWebIdentityRequest;
import software.amazon.awssdk.services.sts.model.IdpCommunicationErrorException;
import software.amazon.awssdk.utils.IoUtils;
import software.amazon.awssdk.utils.SdkAutoCloseable;
import software.amazon.awssdk.utils.StringUtils;
import software.amazon.awssdk.utils.SystemSetting;

import java.io.IOException;
import java.io.InputStream;
import java.io.UncheckedIOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.function.Supplier;

public class LicenseManagerTokenCredentialsProvider implements AwsCredentialsProvider,
    SdkAutoCloseable {

    private final StsAssumeRoleWithWebIdentityCredentialsProvider credentialsProvider;
    private final RuntimeException loadException;

    private Path licenseAccessTokenFile;
    private String roleArn;
    private String roleSessionName;
    private StsClient stsClient;
    private LicenseManagerClient lmClient;

    public static LicenseManagerTokenCredentialsProvider create() {
        return new Builder().build();
    }

    @Override
    public AwsCredentials resolveCredentials() {
        if (this.loadException != null) {
            throw this.loadException;
        }
        return this.credentialsProvider.resolveCredentials();
    }

    @Override
    public void close() {
        IoUtils.closeQuietly(this.credentialsProvider, null);
        IoUtils.closeQuietly(this.stsClient, null);
        IoUtils.closeIfCloseable(this.lmClient, null);
    }

    private LicenseManagerTokenCredentialsProvider(Builder builder) {
        StsAssumeRoleWithWebIdentityCredentialsProvider credentialsProvider = null;
        RuntimeException loadException = null;

        try {
```



```
        this.licenseAccessTokenFile =
Paths.get(StringUtils.trim(LicenseSystemSetting.AWS_WEB_IDENTITY_REFRESH_TOKEN_FILE.getStringValueOrThrow()));
        this.roleArn = SdkSystemSetting.AWS_ROLE_ARN.getStringValueOrThrow();
        this.roleSessionName =
SdkSystemSetting.AWS_ROLE_SESSION_NAME.getStringValue().orElse("aws-sdk-java-" +
System.currentTimeMillis());
        this.stsClient = builder.stsClient != null ? builder.stsClient :
StsClientFactory.create();
        this.lmClient = builder.lmClient != null ? builder.lmClient :
LicenseManagerClientFactory.create();

        AssumeRoleWithWebIdentityRequest request =
AssumeRoleWithWebIdentityRequest.builder()
            .roleArn(this.roleArn).roleSessionName(this.roleSessionName).build();

        Supplier<AssumeRoleWithWebIdentityRequest> supplier = new
AssumeRoleRequestSupplier(request,
            this.licenseAccessTokenFile, this.lmClient);

        credentialsProvider = StsAssumeRoleWithWebIdentityCredentialsProvider.builder()
            .stsClient(this.stsClient).refreshRequest(supplier).build();
    } catch (RuntimeException ex) {
        loadException = ex;
    }

    this.credentialsProvider = credentialsProvider;
    this.loadException = loadException;
}

public static final class Builder {
    private Path licenseAccessTokenFile;
    private String roleArn;
    private String roleSessionName;
    private StsClient stsClient;
    private LicenseManagerClient lmClient;

    public LicenseManagerTokenCredentialsProvider build() {
        return new LicenseManagerTokenCredentialsProvider(this);
    }

    public LicenseManagerTokenCredentialsProvider.Builder licenseAccessTokenFile(Path
licenseAccessTokenFile) {
        this.licenseAccessTokenFile = licenseAccessTokenFile;
        return this;
    }

    public LicenseManagerTokenCredentialsProvider.Builder roleArn(String roleArn) {
        this.roleArn = roleArn;
        return this;
    }

    public LicenseManagerTokenCredentialsProvider.Builder roleSessionName(String
roleSessionName) {
        this.roleSessionName = roleSessionName;
        return this;
    }

    public LicenseManagerTokenCredentialsProvider.Builder stsClient(StsClient
stsClient) {
        this.stsClient = stsClient;
        return this;
    }

    public LicenseManagerTokenCredentialsProvider.Builder lmClient(LicenseManagerClient
lmClient) {
        this.lmClient = lmClient;
    }
}
```

```
        return this;
    }
}

private static final class AssumeRoleRequestSupplier implements Supplier {
    private final LicenseManagerClient lmClient;
    private final AssumeRoleWithWebIdentityRequest request;
    private final Path webIdentityRefreshTokenFile;

    AssumeRoleRequestSupplier(final AssumeRoleWithWebIdentityRequest request,
                              final Path webIdentityRefreshTokenFile,
                              final LicenseManagerClient lmClient) {

        this.lmClient = lmClient;
        this.request = request;
        this.webIdentityRefreshTokenFile = webIdentityRefreshTokenFile;
    }

    public AssumeRoleWithWebIdentityRequest get() {
        return this.request.toBuilder()
            .webIdentityToken(getIdentityToken())
            .build();
    }

    private String getIdentityToken() {
        return refreshIdToken(readRefreshToken(this.webIdentityRefreshTokenFile));
    }

    private String readRefreshToken(Path file) {
        try (InputStream webIdentityRefreshTokenStream = Files.newInputStream(file)) {
            return IoUtils.toUtf8String(webIdentityRefreshTokenStream);
        } catch (IOException e) {
            throw new UncheckedIOException(e);
        }
    }

    private String refreshIdToken(String licenseRefreshToken) {
        final GetAccessTokenRequest request = GetAccessTokenRequest.builder()
            .token(licenseRefreshToken)
            .build();

        GetAccessTokenResponse response = this.lmClient.getAccessToken(request);
        return response.accessToken();
    }
}

private static final class LicenseManagerClientFactory {
    private static final Duration DEFAULT_API_TIMEOUT = Duration.ofSeconds(30);
    private static final Duration DEFAULT_API_ATTEMPT_TIMEOUT = Duration.ofSeconds(10);

    public static LicenseManagerClient create() {
        return getLicenseManagerClient();
    }

    private static LicenseManagerClient getLicenseManagerClient() {
        ClientOverrideConfiguration configuration =
            ClientOverrideConfiguration.builder()
                .apiCallTimeout(DEFAULT_API_TIMEOUT)
                .apiCallAttemptTimeout(DEFAULT_API_ATTEMPT_TIMEOUT)
                .build();

        LicenseManagerClient client = LicenseManagerClient.builder()
            .region(configureLicenseManagerRegion())
            .credentialsProvider(AnonymousCredentialsProvider.create())
            .overrideConfiguration(configuration).build();
        return client;
    }
}
```

```
private static Region configureLicenseManagerRegion() {
    Region defaultRegion = Region.US_EAST_1;

    Region region;
    try {
        region = (new DefaultAwsRegionProviderChain()).getRegion();
    } catch (RuntimeException ex) {
        region = defaultRegion;
    }
    return region;
}

private static final class StsClientFactory {
    private static final Duration DEFAULT_API_TIMEOUT = Duration.ofSeconds(30);
    private static final Duration DEFAULT_API_ATTEMPT_TIMEOUT = Duration.ofSeconds(10);

    public static StsClient create() {
        return getStsClient();
    }

    private static StsClient getStsClient() {
        OrRetryCondition retryCondition = OrRetryCondition.create(new
StsRetryCondition(),
            RetryCondition.defaultRetryCondition());

        ClientOverrideConfiguration configuration =
ClientOverrideConfiguration.builder()
            .apiCallTimeout(DEFAULT_API_TIMEOUT)
            .apiCallAttemptTimeout(DEFAULT_API_ATTEMPT_TIMEOUT)
            .retryPolicy(r -> r.retryCondition(retryCondition))
            .build();

        return StsClient.builder()
            .region(configureStsRegion())
            .credentialsProvider(AnonymousCredentialsProvider.create())
            .overrideConfiguration(configuration).build();
    }

    private static Region configureStsRegion() {
        Region defaultRegion = Region.US_EAST_1;
        Region stsRegion;
        try {
            stsRegion = (new DefaultAwsRegionProviderChain()).getRegion();
        } catch (RuntimeException ex) {
            stsRegion = defaultRegion;
        }
        return stsRegion;
    }

    private static final class StsRetryCondition implements RetryCondition {
        public boolean shouldRetry(RetryPolicyContext context) {
            return context.exception() instanceof IdpCommunicationErrorException;
        }
    }
}

private enum LicenseSystemSetting implements SystemSetting {
    AWS_WEB_IDENTITY_REFRESH_TOKEN_FILE("aws.webIdentityRefreshTokenFile");

    private String systemProperty;
    private String defaultValue = null;

    LicenseSystemSetting(String systemProperty) {
        this.systemProperty = systemProperty;
    }
}
```

```
    }

    @Override
    public String property() {
        return this.systemProperty;
    }

    @Override
    public String environmentVariable() {
        return this.name();
    }

    @Override
    public String defaultValue() {
        return this.defaultValue;
    }
}
}
```

LicenseManagerCredentialsProvider - Golang implementation

LicenseCredentialsProvider

LicenseCredentialsProvider extends the AWS SDK's default credential provider chain for on-premises use by adding LicenseManagerTokenCredentialsProvider.

```
package lib

import (
    "context"
    "fmt"
    "sync"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/config"
)

// LicenseCredentialsProvider is the custom credential provider that can retrieve valid
// temporary aws credentials
type LicenseCredentialsProvider struct {
    fallbackProvider aws.CredentialsProvider
    mux              sync.RWMutex
    licenseCredentials aws.Credentials
    err              error
}

// NewLicenseCredentialsProvider method will create a LicenseCredentialProvider Object
// which contains valid temporary aws credentials
func NewLicenseCredentialsProvider(provider (*LicenseCredentialsProvider, error) {
    licenseCredentialProvider := &LicenseCredentialsProvider{}
    fallbackProvider, err := createCredentialProvider()
    if err != nil {
        return licenseCredentialProvider, fmt.Errorf("failed to create
LicenseCredentialsProvider, %w", err)
    }
    licenseCredentialProvider.fallbackProvider = fallbackProvider
    return licenseCredentialProvider, nil
}

// Retrieve method will retrieve temporary aws credentials from the credential provider
func (l *LicenseCredentialsProvider) Retrieve(ctx context.Context) (aws.Credentials, error) {
    {
        l.mux.RLock()
        defer l.mux.RUnlock()
    }
}
```

```
l.licenseCredentials, l.err = l.fallBackProvider.Retrieve(ctx)
return l.licenseCredentials, l.err
}

func createCredentialProvider() (aws.CredentialsProvider, error) {
// LoadDefaultConfig will examine all "default" credential providers
ctx := context.TODO()
cfg, err := config.LoadDefaultConfig(ctx)
if err != nil {
return nil, fmt.Errorf("failed to create FallBackProvider, %w", err)
}

var useFallbackProvider bool
if cfg.Credentials != nil {
if _, err := cfg.Credentials.Retrieve(ctx); err != nil {
// If the "default" credentials provider cannot retrieve credentials, enable fallback to
customCredentialsProvider.
useFallbackProvider = true
}
} else {
useFallbackProvider = true
}

if useFallbackProvider {
customProvider, err := newLicenseManagerTokenCredentialsProvider()
if err != nil {
return cfg.Credentials, fmt.Errorf("failed to create fallBackProvider, %w", err)
}
// wrap up customProvider with CredentialsCache to enable caching
cfg.Credentials = aws.NewCredentialsCache(customProvider)
}
return cfg.Credentials, nil
}
```

LicenseManagerTokenCredentialsProvider

LicenseManagerTokenCredentialsProvider provides credentials by using License Manager OIDC issued identity tokens in on-premises environments. You must include the source code for LicenseCredentialsProvider in your application classpath.

```
package lib

import (
"context"
"fmt"
"io/ioutil"
"os"
"sync"
"time"

"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/config"
"github.com/aws/aws-sdk-go-v2/service/sts"
)

const awsRefreshTokenFilePathEnvVar = "AWS_LICENSE_ACCESS_FILE"

// licenseManagerTokenCredentialsProvider defines and contains
StsAssumeRoleWithWebIdentityProvider
type licenseManagerTokenCredentialsProvider struct {
stsCredentialProvider *stsAssumeRoleWithWebIdentityProvider
mux                    sync.RWMutex
licenseCredentials    aws.Credentials
err                    error
}
```

```
}

// Retrieve method will retrieve credentials from credential provider.
// Make this method public to make this provider satisfies CredentialProvider interface
func (a *licenseManagerTokenCredentialsProvider) Retrieve(ctx context.Context)
(aws.Credentials, error) {
    a.mux.RLock()
    defer a.mux.RUnlock()
    a.licenseCredentials, a.err = a.stsCredentialProvider.Retrieve(ctx)
    return a.licenseCredentials, a.err
}

// newLicenseManagerTokenCredentialsProvider will create and return
a LicenseManagerTokenCredentialsProvider Object which wraps up
stsAssumeRoleWithWebIdentityProvider
func newLicenseManagerTokenCredentialsProvider() (*licenseManagerTokenCredentialsProvider,
error) {
    // 1. Retrieve variables From yaml environment
    envConfig, err := config.NewEnvConfig()
    if err != nil {
        return &licenseManagerTokenCredentialsProvider{}, fmt.Errorf("failed to create
LicenseManagerTokenCredentialsProvider, %w", err)
    }
    roleArn := envConfig.RoleARN
    var roleSessionName string
    if envConfig.RoleSessionName == "" {
        roleSessionName = fmt.Sprintf("aws-sdk-go-v2-%v", time.Now().UnixNano())
    } else {
        roleSessionName = envConfig.RoleSessionName
    }
    tokenFilePath := os.Getenv(awsRefreshTokenFilePathEnvVar)
    b, err := ioutil.ReadFile(tokenFilePath)
    if err != nil {
        return &licenseManagerTokenCredentialsProvider{}, fmt.Errorf("failed to create
LicenseManagerTokenCredentialsProvider, %w", err)
    }
    refreshToken := aws.String(string(b))

    // 2. Create stsClient
    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        return &licenseManagerTokenCredentialsProvider{}, fmt.Errorf("failed to create
LicenseManagerTokenCredentialsProvider, %w", err)
    }
    stsClient := sts.NewFromConfig(cfg, func(o *sts.Options) {
        o.Region = configureStsClientRegion(cfg.Region)
        o.Credentials = aws.AnonymousCredentials{}
    })

    // 3. Configure StsAssumeRoleWithWebIdentityProvider
    stsCredentialProvider := newStsAssumeRoleWithWebIdentityProvider(stsClient, roleArn,
roleSessionName, refreshToken)

    // 4. Build and return
    return &licenseManagerTokenCredentialsProvider{
        stsCredentialProvider: stsCredentialProvider,
    }, nil
}

func configureStsClientRegion(configRegion string) string {
    defaultRegion := "us-east-1"
    if configRegion == "" {
        return defaultRegion
    } else {
        return configRegion
    }
}
```

```
}
```

Amazon SNS notifications for container products

To receive notifications, you subscribe to the AWS Marketplace Amazon Simple Notification Service (Amazon SNS) topics provided to you during product creation. The topics provide notifications about changes to customers' subscriptions for your products. For example, you can use this to know when customers accept a private offer.

The following Amazon SNS topic is available for container products:

- [Amazon SNS topic: aws-mp-subscription-notification \(p. 174\)](#) – This topic notifies you when a buyer subscribes or unsubscribes to a product. This is available for hourly pricing models, including hourly and hourly with long term.

Amazon SNS topic: aws-mp-subscription-notification

Each message in the aws-mp-subscription-notification topic has the following format.

```
{
  "action": "<action-name>",
  "customer-identifier": " X01EXAMPLE",
  "product-code": "n0123EXAMPLEXXXXXXXXXXXX",
  "offer-identifier": "offer-abcexample123"
}
```

The `<action-name>` will vary depending on the notification. Possible actions are:

- subscribe-success
- subscribe-fail
- unsubscribe-pending
- unsubscribe-success

The `offer-identifier` only appears in the notification if the offer is a *private offer*.

Subscribing an Amazon SQS queue to the Amazon SNS topic

We recommend subscribing an Amazon SQS queue to the provided SNS topics. For detailed instructions on creating an SQS queue and subscribing the queue to a topic, see [Subscribing an Amazon SQS queue to an Amazon SNS topic](#) in the *Amazon Simple Notification Service Developer Guide*.

Note

You can only subscribe to AWS Marketplace SNS topics from the AWS account used to sell the products. However, you can forward the messages to a different account. For more information, see [Sending Amazon SNS messages to an Amazon SQS queue in a different account](#) in the *Amazon Simple Notification Service Developer Guide*.

Polling the SQS queue for notifications

After you subscribe your SQS queue to an SNS topic, the messages are stored in SQS. You must define a service that continually polls the queue, looks for messages, and handles them accordingly.

Machine learning products

As a seller, you can use AWS Marketplace to create machine learning (ML) algorithms and models that your buyers can deploy in AWS. There are two types of Amazon SageMaker products listed in AWS Marketplace:

Model package

A pre-trained model for making predictions that does not require any further training by the buyer.

Algorithm

A model that requires the buyer to supply training data before it makes predictions. The training algorithm is included.

These products are available to buyers through the Amazon SageMaker console or AWS Marketplace. Buyers can review product descriptions, documentation, customer reviews, pricing, and support information. When they subscribe to either a model package product or algorithm product, it's added to their product list on the SageMaker console. Buyers can also use AWS SDKs, the AWS Command Line Interface (AWS CLI), or the SageMaker console to create a fully managed REST inference endpoint or perform inference on batches of data.

For support with creating machine learning products with Amazon SageMaker, contact [AWS Marketplace Seller Operations](#).

Getting started with machine learning products

AWS Marketplace supports two machine learning product types, using Amazon SageMaker. Both types, the model package products and the algorithm products, produce a deployable inference model for making predictions.

SageMaker model package

An [Amazon SageMaker model package](#) product contains a pre-trained model. Pre-trained models can be deployed in SageMaker to make inferences or predictions in real time or in batches. This product contains a trained inference component with model artifacts, if any. As a seller, you can train a model using SageMaker or bring your own model.

SageMaker algorithm

Buyers can use a [SageMaker algorithm](#) product to perform complete machine learning workloads. An algorithm product has two logical components: training and inference. In SageMaker, buyers use their own datasets to create a training job with your training component. When the algorithm in your training component completes, it generates the model artifacts of the machine learning model. SageMaker saves the model artifacts in the buyers' Amazon Simple Storage Service (Amazon S3) bucket. In SageMaker, buyers can then deploy your inference component along with those generated model artifacts to perform inference (or prediction) in real time or in batches.

Deploying an inference model

Whether the inference model is created from a model package or an algorithm, there are two methods to deploy them:

- **Endpoint** – This method uses SageMaker to deploy the model and create an API endpoint. The buyer can use this endpoint as part of their backend service to power their applications. When data is sent to the endpoint, SageMaker passes it to the model container and returns the results in an API response. The endpoint and the container continue to run until stopped by the buyer.

Note

In AWS Marketplace, the endpoint method is referred to as *real-time inference*, and in the SageMaker documentation, it is referred to as *hosting services*. For more information, see [Deploy a Model in Amazon SageMaker](#).

- **Batch transform job** – In this method, a buyer stores datasets for inference in Amazon S3. When the batch transform job starts, SageMaker deploys the model, passes data from an S3 bucket to the model's container, and then returns the results to an S3 bucket. When the job completes, SageMaker stops the job. For more information, see [Get Inferences for an Entire Dataset with Batch Transform](#).

Note

Both methods are transparent to the model because SageMaker passes data to the model and returns results to the buyer.

Security and intellectual property

Amazon SageMaker protects both your intellectual property and buyer data for models and algorithms obtained from AWS Marketplace.

Protecting intellectual property

When you create a product, the code is packaged in Docker container images. For more information, see [Prepare your product in SageMaker \(p. 180\)](#), later in this guide. When you upload a container image, the image and artifacts are encrypted in transit and at rest. The images are also scanned for vulnerabilities before being published.

To help safeguard your intellectual property, SageMaker allows only buyers to access your product through AWS service endpoints. Buyers cannot directly access or pull container images or model artifacts, nor can they access the underlying infrastructure.

No network access

Unlike SageMaker models and algorithms that buyers create, when buyers launch your product from AWS Marketplace, the models and algorithms are deployed without network access. SageMaker deploys images in an environment with no access to the network or AWS service endpoints. For example, a container image can't make outbound API calls to services on the internet, [VPC endpoints](#), or any other AWS services.

Security of customer data

Your product runs in SageMaker within the buyer's AWS account. So, when a buyer uses your product to perform data inference, you as the seller can't access their data.

For algorithm products, model artifacts are outputted by your training image after each training job. Model artifacts are stored in the buyer's account. The model artifacts from the training job are used when the buyer deploys the model with your inference image. To protect any intellectual property that may be contained in the model artifact, encrypt them before outputting them.

Important

This security model prevents your code from accessing the internet during runtime. Therefore, your code can't use resources or libraries from the internet, so package your dependencies in the

Docker container image. This is especially important if you choose to encrypt your outputted artifacts from the training job. The keys to encrypt and decrypt artifacts can't be accessed over the internet at runtime. They must be packaged with your image.

For more information, see [Security in Amazon SageMaker](#).

Machine learning product pricing

You can choose from several available pricing models for your Amazon SageMaker products. Buyers who subscribe to your product run it in SageMaker within their own AWS account. The price for your buyers is a combination of the infrastructure costs for the resources running in their AWS account and the product pricing that you set.

Infrastructure pricing

Buyers are responsible for all the infrastructure costs of SageMaker while using your product. These costs are set by AWS and are available on the [Amazon SageMaker pricing](#) page.

Software pricing

You determine the software prices that AWS Marketplace charges the buyer for using your product. You set the pricing and terms when you are adding your machine learning product to AWS Marketplace.

All infrastructure and software prices per instance type are presented to the buyer on the product listing pages in AWS Marketplace before the buyer subscribes.

Free pricing

You can choose to offer your product for free. In this case, the buyer only pays for infrastructure costs.

Hourly pricing

You can offer your product with a price per hour per instance of your software running in SageMaker. You can charge a different hourly price for each instance type that your software runs on. While a buyer runs your software, AWS Marketplace tracks usage and then bills the buyer accordingly. Usage is prorated to the minute.

For *model package* products, buyer can run your software in two different ways. They can host an endpoint continuously to perform real-time inference or run a batch transform job on a dataset. You can set different pricing for both of the ways a buyer can run your software.

For *algorithm* products, in addition to determining the prices for performing inference, as mentioned earlier, you also determine an hourly price for training jobs. You can charge a different hourly price for each instance type that your training image supports.

Inference pricing

When the buyer runs your software by hosting an endpoint to continuously perform real-time inference, you can choose to set a price per inference.

Note

Batch transform processes always use hourly pricing. Training jobs for algorithm products also always use hourly pricing. You can set these prices independently of the inference pricing, and of each other.

By default, with inference pricing, AWS Marketplace charges your buyer for each invocation of your endpoint. However, in some cases, your software processes a batch of inferences in a single invocation (also known as a *mini-batch*). For an endpoint deployment, you can indicate a custom number of inferences that AWS Marketplace should charge the buyer for that single invocation. To do this, include a custom metering header in the HTTP response headers of your invocation, as in the following example.

```
X-Amzn-Inference-Metering: {"Dimension": "inference.count", "ConsumedUnits": 3}
```

This example shows an invocation that charges the buyer for three inferences.

Note

For inference pricing, AWS Marketplace only charges buyer for requests where the HTTP response code is 2XX.

Free trial

Optionally, you can create a free trial for your product and define the number of days of the free trial. Free trials can be 5–120 days. During the free trial, buyers can run your software as much as they want and aren't charged for your software. Buyers are charged for the infrastructure costs during the free trial. After the trial ends, they are charged your normal software price, along with the infrastructure costs.

Note

You can only create a free trial for offers that are charged hourly. You can't create a free trial for a product with inference pricing.

When buyers subscribe to a product with a free trial, they receive a welcome email message. The message includes the term of the free trial, a calculated expiration date, and details on unsubscribing. A reminder email message is sent three days before the expiration date.

If you offer a free trial for your product in AWS Marketplace, you agree to the specific [refund policy](#) for free trials.

Private offers

You can create private offers for your machine learning products. A private offer gives specific buyers a different price than your publicly displayed price.

Private offers work in one of two ways:

- **Hourly** – Private offers can be an hourly rate that is different from the publicly displayed hourly rate.
- **Contract** – Private offers can be a contract with a fixed upfront fee for a specified number of days. The buyer is allowed to use an unlimited number of instances for the entire duration of the contract. At the end of the contract, any instances that continue to run are billed at the hourly rate that you set in the private offer. For example, you can create a contract with a fixed upfront fee for 365 days of unlimited use. You also set an hourly rate for the private offer. When the buyer accepts this private offer, they pay that upfront fee. When the contract ends, any instances still running are billed at that hourly rate.

The set of terms and agreement between you and the buyer in private offers can differ from the one in the public offer or other private offers.

You can create and extend multiple private offers to a single buyer. Buyers that you extend the private offers to have the option to choose between the private offers and the public offer. Buyers can only be subscribed to one offer at any given time. They can't be subscribed to both a private offer and the public offer at the same time.

To create a private offer for a specific buyer for SageMaker products, contact [AWS Marketplace Seller Operations](#).

Note

For more details about limitations of private offers, see [Notes about private offers \(p. 45\)](#).

Prepare your product in SageMaker

Before publishing your product in AWS Marketplace, you must prepare it in Amazon SageMaker. There are three steps to preparing your product:

1. [Packaging your code into images \(p. 180\)](#) – To prepare a model package or algorithm product, you must create the Docker container images for your product.
2. [Uploading your images \(p. 194\)](#) – After packaging your code in container images and testing them locally, upload the images and scan them for known vulnerabilities. Fix any vulnerabilities before continuing.
3. [Creating your Amazon SageMaker resource \(p. 196\)](#) – After your images are scanned successfully, they can be used to create a model package or algorithm resource in SageMaker.

Packaging your code into images

Machine learning products in AWS Marketplace use Amazon SageMaker to create and run the machine learning logic you provide for buyers. SageMaker runs Docker container images that contain your logic. SageMaker runs these containers in a secure and scalable infrastructure. For more information, see [Security and intellectual property \(p. 177\)](#).

Topics

- [Which type of container image do I create? \(p. 180\)](#)
- [Model package images \(p. 181\)](#)
- [Algorithm images \(p. 185\)](#)

Which type of container image do I create?

The two types of container images are an inference image and a training image.

To create a model package product, you need only an inference image. For detailed instructions, see [Model package images \(p. 181\)](#).

To create an algorithm product, you need both training and inference images. For detailed instructions, see [Algorithm images \(p. 185\)](#).

To package code properly into a container image, the container must adhere to the SageMaker file structure. The container must expose the correct endpoints to ensure that the service can pass data to and from your container. The following sections explain the details of this process.

Important

For security purposes, when a buyer subscribes to your containerized product, the Docker containers run in an isolated environment without an internet connection. When you create your containers, don't rely on outgoing calls over the internet because they will fail. Calls to AWS services will also fail. For more information, see the [Security and intellectual property \(p. 177\)](#) section.

Optionally, when creating your inference and training images, use a container from [Available Deep Learning Containers Images](#) as a starting point. The images are already properly packaged with different machine learning frameworks.

Model package images

An Amazon SageMaker SageMaker model package is a pre-trained model that makes predictions and does not require any further training by the buyer.

A model package includes the following components:

- An inference image stored in [Amazon Elastic Container Registry](#) (Amazon ECR)
- (Optional) Model artifacts, stored separately in [Amazon S3](#)

Note

Model artifacts are files your model uses to make predictions and are generally the result of your own training processes. Artifacts can be any file type that is needed by your model, but must use .tar.gz compression. For model packages, they can either be bundled within your inference image or be stored separately in Amazon S3. The model artifacts stored in Amazon S3 are loaded into the inference container at runtime. When publishing your model package, those artifacts are published and stored in AWS Marketplace owned Amazon S3 buckets that are inaccessible by the buyer directly.

The following is an overview of how buyers use a model package and its components:

1. The buyer subscribes to a model package and deploys the model. SageMaker runs the inference image. Any seller-provided model artifacts not bundled in the inference image are loaded dynamically at runtime.
2. SageMaker passes the buyer's inference data to the container via the container's HTTP endpoints and returns the prediction results.

Note

Your model can be deployed as an endpoint for single inferences or as a batch job to get predictions for an entire dataset at once. For more information, see [Deploy Models for Inference](#).

Creating an inference image for model packages

This section provides a walkthrough for packaging your inference code into an inference image for your model package product.

The inference image is a Docker image containing your inference logic. The container at runtime exposes HTTP endpoints to allow SageMaker to pass data to and from your container.

Note

The following is only one example of packaging code for an inference image. For more information, see [Using Docker containers with SageMaker](#) and the [AWS Marketplace SageMaker examples](#) on GitHub.

The following example uses a web service, [Flask](#), for simplicity, and is not considered production-ready.

Step 1: Creating the container image

For the inference image to be compatible with SageMaker, the Docker image must expose HTTP endpoints. While your container is running, SageMaker passes buyer inputs for inference to the container's HTTP endpoint. The inference results are returned in the body of the HTTP response.

The following walkthrough uses the Docker CLI in a development environment using a Linux Ubuntu distribution.

- [Create the web server script \(p. 182\)](#)
- [Create the script for the container run \(p. 183\)](#)

- [Create the Dockerfile \(p. 183\)](#)
- [Package or upload the model artifacts \(p. 183\)](#)

Create the web server script

This example uses a Python server called [Flask](#), but you can use any web server that works for your framework.

Note

[Flask](#) is used here for simplicity. It is not considered a production-ready web server.

Create a Flask web server script that serves the two HTTP endpoints on TCP port 8080 that SageMaker uses. The following are the two expected endpoints:

- `/ping` – SageMaker makes HTTP GET requests to this endpoint to check if your container is ready. When your container is ready, it responds to HTTP GET requests at this endpoint with an HTTP 200 response code.
- `/invocations` – SageMaker makes HTTP POST requests to this endpoint for inference. The input data for inference is sent in the body of the request. The user-specified content type is passed in the HTTP header. The body of the response is the inference output. For details about timeouts, see [Requirements and best practices for creating machine learning products \(p. 204\)](#).

`./web_app_serve.py`

```
# Import modules
import json
import re
from flask import Flask
from flask import request
app = Flask(__name__)

# Create a path for health checks
@app.route("/ping")
def endpoint_ping():
    return ""

# Create a path for inference
@app.route("/invocations", methods=["POST"])
def endpoint_invocations():

    # Read the input
    input_str = request.get_data().decode("utf8")

    # Add your inference code between these comments.
    #
    #
    #
    # Add your inference code above this comment.

    # Return a response with a prediction
    response = {"prediction": "a", "text": input_str}
    return json.dumps(response)
```

In the previous example, there is no actual inference logic. For your actual inference image, add the inference logic into the web app so it processes the input and returns the actual prediction.

Your inference image must contain all of its required dependencies because it will not have internet access, nor will it be able to make calls to any AWS services.

Note

This same code is called for both real-time and batch inferences

Create the script for the container run

Create a script named `serve` that SageMaker runs when it runs the Docker container image. The following script starts the HTTP web server.

./serve

```
#!/bin/bash

# Run flask server on port 8080 for SageMaker
flask run --host 0.0.0.0 --port 8080
```

Create the Dockerfile

Create a Dockerfile in your build context. This example uses Ubuntu 18.04, but you can start from any base image that works for your framework.

./Dockerfile

```
FROM ubuntu:18.04

# Specify encoding
ENV LC_ALL=C.UTF-8
ENV LANG=C.UTF-8

# Install python-pip
RUN apt-get update \
&& apt-get install -y python3.6 python3-pip \
&& ln -s /usr/bin/python3.6 /usr/bin/python \
&& ln -s /usr/bin/pip3 /usr/bin/pip;

# Install flask server
RUN pip install -U Flask;

# Add a web server script to the image
# Set an environment to tell flask the script to run
COPY /web_app_serve.py /web_app_serve.py
ENV FLASK_APP=/web_app_serve.py

# Add a script that Amazon SageMaker will run
# Set run permissions
# Prepend program directory to $PATH
COPY /serve /opt/program/serve
RUN chmod 755 /opt/program/serve
ENV PATH=/opt/program:${PATH}
```

The Dockerfile adds the two previously created scripts to the image. The directory of the `serve` script is added to the `PATH` so it can run when the container runs.

Package or upload the model artifacts

The two ways to provide the model artifacts from training the model to the inference image are as follows:

- Packaged statically with the inference image.
- Loaded dynamically at runtime. Because it's loaded dynamically, you can use the same image for packaging different machine learning models.

If you want to package your model artifacts with the inference image, include the artifacts in the `Dockerfile`.

If you want to load your model artifacts dynamically, store those artifacts separately in a compressed file (.tar.gz) in Amazon S3. When creating the model package, specify the location of the compressed file, and SageMaker extracts and copies the contents to the container directory `/opt/ml/model/` when running your container. When publishing your model package, those artifacts are published and stored in AWS Marketplace owned Amazon S3 buckets that are inaccessible by the buyer directly.

Step 2: Building and testing the image locally

In the build context, the following files now exist:

- `./Dockerfile`
- `./web_app_serve.py`
- `./serve`
- Your inference logic and (optional) dependencies

Next build, run, and test the container image.

Build the image

Run the Docker command in the build context to build and tag the image. This example uses the tag `my-inference-image`.

```
sudo docker build --tag my-inference-image ./
```

After running this Docker command to build the image, you should see output as Docker builds the image based on each line in your `Dockerfile`. When it finishes, you should see something similar to the following.

```
Successfully built abcdef123456  
Successfully tagged my-inference-image:latest
```

Run locally

After your build has completed, you can test the image locally.

```
sudo docker run \  
  --rm \  
  --publish 8080:8080/tcp \  
  --detach \  
  --name my-inference-container \  
  my-inference-image \  
  serve
```

The following are details about the command:

- `--rm` – Automatically remove the container after it stops.
- `--publish 8080:8080/tcp` – Expose port 8080 to simulate the port that SageMaker sends HTTP requests to.
- `--detach` – Run the container in the background.
- `--name my-inference-container` – Give this running container a name.
- `my-inference-image` – Run the built image.

- `serve` – Run the same script that SageMaker runs when running the container.

After running this command, Docker creates a container from the inference image you built and runs it in the background. The container runs the `serve` script, which launches your web server for testing purposes.

Test the ping HTTP endpoint

When SageMaker runs your container, it periodically pings the endpoint. When the endpoint returns an HTTP response with status code 200, it signals to SageMaker that the container is ready for inference. You can test this by running the following command, which tests the endpoint and includes the response header.

```
curl --include http://127.0.0.1:8080/ping
```

Example output is as follows.

```
HTTP/1.0 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 0
Server: MyServer/0.16.0 Python/3.6.8
Date: Mon, 21 Oct 2019 06:58:54 GMT
```

Test the inference HTTP endpoint

When the container indicates it is ready by returning a 200 status code to your ping, SageMaker passes the inference data to the `/invocations` HTTP endpoint via a `POST` request. Test the inference point by running the following command.

```
curl \
  --request POST \
  --data "hello world" \
  http://127.0.0.1:8080/invocations
```

Example output is as follows.

```
{"prediction": "a", "text": "hello world"}
```

With these two HTTP endpoints working, the inference image is now compatible with SageMaker.

Note

The model of your model package product can be deployed in two ways: real time and batch. In both deployments, SageMaker uses the same HTTP endpoints while running the Docker container.

To stop the container, run the following command.

```
sudo docker container stop my-inference-container
```

When your inference image is ready and tested, you can continue to [Uploading your images \(p. 194\)](#).

Algorithm images

An Amazon SageMaker algorithm requires that the buyer bring their own data to train before it makes predictions.

An algorithm includes the following components:

- A training image stored in [Amazon ECR](#)
- An inference image stored in Amazon Elastic Container Registry (Amazon ECR)

Note

For algorithm products, the training container generates model artifacts that are loaded into the inference container on model deployment.

The following is an overview of how buyers use an algorithm and its components:

1. The buyer creates a training job with a compatible dataset and appropriate hyperparameter values. SageMaker runs the training image and loads the training data and hyperparameters into the training container. When the training job completes, the model artifacts located in `/opt/ml/model/` are compressed and copied to the buyer's [Amazon S3](#) bucket.
2. The buyer creates a model package with the model artifacts from the training stored in Amazon S3 and deploys the model. SageMaker runs the inference image, extracts the compressed model artifacts, and loads the files into the inference container directory path `/opt/ml/model/` where it is consumed by the code that serves the inference.
3. Whether the model deploys as an endpoint or a batch transform job, SageMaker passes the data for inference on behalf of the buyer to the container via the container's HTTP endpoint and returns the prediction results.

Note

For more information, see [Train Models](#).

Creating a training image for algorithms

This section provides a walkthrough for packaging your training code into a training image. A training image is required to create an algorithm product.

A *training image* is a Docker image containing your training algorithm. The container adheres to a specific file structure to allow SageMaker to copy data to and from your container.

Both the training and inference images are required when publishing an algorithm product. After creating your training image, you must create an inference image. The two images can be combined into one image or remain as separate images. Whether to combine the images or separate them is up to you. Typically, inference is simpler than training, and you might want separate images to help with inference performance.

Note

The following is only one example of packaging code for a training image. For more information, see [Use your own algorithms and models with the AWS Marketplace](#) and the [AWS Marketplace SageMaker examples](#) on GitHub.

Step 1: Creating the container image

For the training image to be compatible with Amazon SageMaker, it must adhere to a specific file structure to allow SageMaker to copy the training data and configuration inputs to specific paths in your container. When the training completes, the generated model artifacts are stored in a specific directory path in the container where SageMaker copies from.

The following uses Docker CLI installed in a development environment on an Ubuntu distribution of Linux.

- [Prepare your program to read configuration inputs \(p. 187\)](#)

- [Prepare your program to read data inputs \(p. 187\)](#)
- [Prepare your program to write training outputs \(p. 187\)](#)
- [Create the script for the container run \(p. 188\)](#)
- [Create the `Dockerfile` \(p. 188\)](#)

Prepare your program to read configuration inputs

If your training program requires any buyer-provided configuration input, the following is where those are copied to inside your container when ran. If required, your program must read from those specific file paths.

- `/opt/ml/input/config` is the directory that contains information which controls how your program runs.
 - `hyperparameters.json` is a JSON-formatted dictionary of hyperparameter names and values. The values are strings, so you may need to convert them.
 - `resourceConfig.json` is a JSON-formatted file that describes the network layout used for [distributed training](#). If your training image does not support distributed training, you can ignore this file.

Note

For more information about configuration inputs, see [How Amazon SageMaker Provides Training Information](#).

Prepare your program to read data inputs

Training data can be passed to the container in one of the following two modes. Your training program that runs in the container digests the training data in one of those two modes.

File mode

- `/opt/ml/input/data/<channel_name>/` contains the input data for that channel. The channels are created based on the call to the `CreateTrainingJob` operation, but it's generally important that channels match what the algorithm expects. The files for each channel are copied from [Amazon S3](#) to this directory, preserving the tree structure indicated by the Amazon S3 key structure.

Pipe mode

- `/opt/ml/input/data/<channel_name>_<epoch_number>` is the pipe for a given epoch. Epochs start at zero and increase by one each time you read them. There is no limit to the number of epochs that you can run, but you must close each pipe before reading the next epoch.

Prepare your program to write training outputs

The output of the training is written to the following container directories:

- `/opt/ml/model/` is the directory where you write the model or the model artifacts that your training algorithm generates. Your model can be in any format that you want. It can be a single file or a whole directory tree. SageMaker packages any files in this directory into a compressed file (`.tar.gz`). This file is available at the Amazon S3 location returned by the `DescribeTrainingJob` API operation.
- `/opt/ml/output/` is a directory where the algorithm can write a `failure` file that describes why the job failed. The contents of this file are returned in the `FailureReason` field of the `DescribeTrainingJob` result. For jobs that succeed, there is no reason to write this file because it's ignored.

Create the script for the container run

Create a `train` shell script that SageMaker runs when it runs the Docker container image. When the training completes and the model artifacts are written to their respective directories, exit the script.

`./train`

```
#!/bin/bash

# Run your training program here
#
#
#
```

Create the Dockerfile

Create a `Dockerfile` in your build context. This example uses Ubuntu 18.04 as the base image, but you can start from any base image that works for your framework.

`./Dockerfile`

```
FROM ubuntu:18.04

# Add training dependencies and programs
#
#
#
#
# Add a script that SageMaker will run
# Set run permissions
# Prepend program directory to $PATH
COPY /train /opt/program/train
RUN chmod 755 /opt/program/train
ENV PATH=/opt/program:${PATH}
```

The `Dockerfile` adds the previously created `train` script to the image. The script's directory is added to the `PATH` so it can run when the container runs.

In the previous example, there is no actual training logic. For your actual training image, add the training dependencies to the `Dockerfile`, and add the logic to read the training inputs to train and generate the model artifacts.

Your training image must contain all of its required dependencies because it will not have internet access.

For more information, see [Use your own algorithms and models with the AWS Marketplace](#) and the [AWS Marketplace SageMaker examples](#) on GitHub.

Step 2: Building and testing the image locally

In the build context, the following files now exist:

- `./Dockerfile`
- `./train`
- Your training dependencies and logic

Next you can build, run, and test this container image.

Build the image

Run the Docker command in the build context to build and tag the image. This example uses the tag `my-training-image`.

```
sudo docker build --tag my-training-image ./
```

After running this Docker command to build the image, you should see output as Docker builds the image based on each line in your `Dockerfile`. When it finishes, you should see something similar to the following.

```
Successfully built abcdef123456  
Successfully tagged my-training-image:latest
```

Run locally

After that has completed, test the image locally as shown in the following example.

```
sudo docker run \  
  --rm \  
  --volume '<path_to_input>:/opt/ml/input:ro' \  
  --volume '<path_to_model>:/opt/ml/model' \  
  --volume '<path_to_output>:/opt/ml/output' \  
  --name my-training-container \  
  my-training-image \  
  train
```

The following are command details:

- `--rm` – Automatically remove the container after it stops.
- `--volume '<path_to_input>:/opt/ml/input:ro'` – Make test input directory available to container as read-only.
- `--volume '<path_to_model>:/opt/ml/model'` – Bind mount the path where the model artifacts are stored on the host machine when the training test is complete.
- `--volume '<path_to_output>:/opt/ml/output'` – Bind mount the path where the failure reason in a `failure` file is written to on the host machine.
- `--name my-training-container` – Give this running container a name.
- `my-training-image` – Run the built image.
- `train` – Run the same script SageMaker runs when running the container.

After running this command, Docker creates a container from the training image you built and runs it. The container runs the `train` script, which starts your training program.

After your training program finishes and the container exits, check that the output model artifacts are correct. Additionally, check the log outputs to confirm that they are not producing logs that you do not want, while ensuring enough information is provided about the training job.

This completes packaging your training code for an algorithm product. Because an algorithm product also includes an inference image, continue to the next section, [Creating an inference image for algorithms](#) (p. 189).

Creating an inference image for algorithms

This section provides a walkthrough for packaging your inference code into an inference image for your algorithm product.

The inference image is a Docker image containing your inference logic. The container at runtime exposes HTTP endpoints to allow SageMaker to pass data to and from your container.

Both the training and inference images are required when publishing an algorithm product. If you have not already done so, see the previous section about [Creating a training image for algorithms \(p. 186\)](#). The two images can be combined into one image or remain as separate images. Whether to combine the images or separate them is up to you. Typically, inference is simpler than training, and you might want separate images to help with inference performance.

Note

The following is only one example of packaging code for an inference image. For more information, see [Use your own algorithms and models with the AWS Marketplace](#) and the [AWS Marketplace SageMaker examples](#) on GitHub.

The following example uses a web service, [Flask](#), for simplicity, and is not considered production-ready.

Step 1: Creating the inference image

For the inference image to be compatible with SageMaker, the Docker image must expose HTTP endpoints. While your container is running, SageMaker passes inputs for inference provided by the buyer to your container's HTTP endpoint. The result of the inference is returned in the body of the HTTP response.

The following uses Docker CLI installed in a development environment on an Ubuntu distribution of Linux.

- [Create the web server script \(p. 190\)](#)
- [Create the script for the container run \(p. 191\)](#)
- [Create the Dockerfile \(p. 191\)](#)
- [Preparing your program to dynamically load model artifacts \(p. 192\)](#)

Create the web server script

This example uses a Python server called [Flask](#), but you can use any web server that works for your framework.

Note

[Flask](#) is used here for simplicity. It is not considered a production-ready web server.

Create the Flask web server script that serves the two HTTP endpoints on TCP port 8080 that SageMaker uses. The following are the two expected endpoints:

- `/ping` – SageMaker makes HTTP GET requests to this endpoint to check if your container is ready. When your container is ready, it responds to HTTP GET requests at this endpoint with an HTTP 200 response code.
- `/invocations` – SageMaker makes HTTP POST requests to this endpoint for inference. The input data for inference is sent in the body of the request. The user-specified content type is passed in the HTTP header. The body of the response is the inference output.

`./web_app_serve.py`

```
# Import modules
import json
import re
from flask import Flask
from flask import request
```

```
app = Flask(__name__)

# Create a path for health checks
@app.route("/ping")
def endpoint_ping():
    return ""

# Create a path for inference
@app.route("/invocations", methods=["POST"])
def endpoint_invocations():

    # Read the input
    input_str = request.get_data().decode("utf8")

    # Add your inference code here.
    #
    #
    #
    # Add your inference code here.

    # Return a response with a prediction
    response = {"prediction": "a", "text": input_str}
    return json.dumps(response)
```

In the previous example, there is no actual inference logic. For your actual inference image, add the inference logic into the web app so it processes the input and returns the prediction.

Your inference image must contain all of its required dependencies because it will not have internet access.

Create the script for the container run

Create a script named `serve` that SageMaker runs when it runs the Docker container image. In this script, start the HTTP web server.

./serve

```
#!/bin/bash

# Run flask server on port 8080 for SageMaker
flask run --host 0.0.0.0 --port 8080
```

Create the Dockerfile

Create a `Dockerfile` in your build context. This example uses Ubuntu 18.04, but you can start from any base image that works for your framework.

./Dockerfile

```
FROM ubuntu:18.04

# Specify encoding
ENV LC_ALL=C.UTF-8
ENV LANG=C.UTF-8

# Install python-pip
RUN apt-get update \
&& apt-get install -y python3.6 python3-pip \
&& ln -s /usr/bin/python3.6 /usr/bin/python \
```



```
&& ln -s /usr/bin/pip3 /usr/bin/pip;

# Install flask server
RUN pip install -U Flask;

# Add a web server script to the image
# Set an environment to tell flask the script to run
COPY /web_app_serve.py /web_app_serve.py
ENV FLASK_APP=/web_app_serve.py

# Add a script that Amazon SageMaker will run
# Set run permissions
# Prepend program directory to $PATH
COPY /serve /opt/program/serve
RUN chmod 755 /opt/program/serve
ENV PATH=/opt/program:${PATH}
```

The `Dockerfile` adds the two created previously scripts to the image. The directory of the `serve` script is added to the `PATH` so it can run when the container runs.

Preparing your program to dynamically load model artifacts

For algorithm products, the buyer uses their own datasets with your training image to generate unique model artifacts. When the training process completes, your training container outputs model artifacts to the container directory `/opt/ml/model/`. SageMaker compresses the contents in that directory into a `.tar.gz` file and stores it in the buyer's AWS account in Amazon S3.

When the model deploys, SageMaker runs your inference image, extracts the model artifacts from the `.tar.gz` file stored in the buyer's account in Amazon S3, and loads them into the inference container in the `/opt/ml/model/` directory. At runtime, your inference container code uses the model data.

Note

To protect any intellectual property that might be contained in the model artifact files, you can choose to encrypt the files before outputting them. For more information, see [Security and intellectual property \(p. 177\)](#).

Step 2: Building and testing the image locally

In the build context, the following files now exist:

- `./Dockerfile`
- `./web_app_serve.py`
- `./serve`

Next you can build, run, and test this container image.

Build the image

Run the Docker command to build and tag the image. This example uses the tag `my-inference-image`.

```
sudo docker build --tag my-inference-image ./
```

After running this Docker command to build the image, you should see output as Docker builds the image based on each line in your `Dockerfile`. When it finishes, you should see something similar to the following.

```
Successfully built abcdef123456
```

```
Successfully tagged my-inference-image:latest
```

Run locally

After your build has completed, you can test the image locally.

```
sudo docker run \  
  --rm \  
  --publish 8080:8080/tcp \  
  --volume '<path_to_model>:/opt/ml/model:ro' \  
  --detach \  
  --name my-inference-container \  
  my-inference-image \  
  serve
```

The following are command details:

- `--rm` – Automatically remove the container after it stops.
- `--publish 8080:8080/tcp` – Expose port 8080 to simulate the port SageMaker sends HTTP requests to.
- `--volume '<path_to_model>:/opt/ml/model:ro'` – Bind mount the path to where the test model artifacts are stored on the host machine as read-only to make them available to your inference code in the container.
- `--detach` – Run the container in the background.
- `--name my-inference-container` – Give this running container a name.
- `my-inference-image` – Run the built image.
- `serve` – Run the same script SageMaker runs when running the container.

After running this command, Docker creates a container from the inference image and runs it in the background. The container runs the `serve` script, which starts your web server for testing purposes.

Test the ping HTTP endpoint

When SageMaker runs your container, it periodically pings the endpoint. When the endpoint returns an HTTP response with status code 200, it signals to SageMaker that the container is ready for inference.

Run the following command to test the endpoint and include the response header.

```
curl --include http://127.0.0.1:8080/ping
```

Example output is shown in the following example.

```
HTTP/1.0 200 OK  
Content-Type: text/html; charset=utf-8  
Content-Length: 0  
Server: MyServer/0.16.0 Python/3.6.8  
Date: Mon, 21 Oct 2019 06:58:54 GMT
```

Test the inference HTTP endpoint

When the container indicates it is ready by returning a 200 status code, SageMaker passes the inference data to the `/invocations` HTTP endpoint via a `POST` request.

Run the following command to test the inference endpoint.

```
curl \
  --request POST \
  --data "hello world" \
  http://127.0.0.1:8080/invocations
```

Example output is shown in the following example..

```
{"prediction": "a", "text": "hello world"}
```

With these two HTTP endpoints working, the inference image is now compatible with SageMaker.

Note

The model of your algorithm product can be deployed in two ways: real time and batch. For both deployments, SageMaker uses the same HTTP endpoints while running the Docker container.

To stop the container, run the following command.

```
sudo docker container stop my-inference-container
```

After both your training and inference images for your algorithm product are ready and tested, continue to [Uploading your images \(p. 194\)](#).

Uploading your images

This section provides a walkthrough for uploading your inference and training images to Amazon Elastic Container Registry. [Amazon ECR](#) is a fully managed Docker registry. This is where Amazon SageMaker pulls images from to create a model package for inference or algorithm for training jobs. This is also where AWS Marketplace retrieves the images to publish your model package and algorithm products.

Which images must I upload?

If you're publishing a model package, upload only an inference image. If you're publishing an algorithm, upload both an inference image and a training image. If the inference and training images are combined, upload the combined image only once.

What IAM permissions are required?

The following steps assume that the local machine has the correct AWS credentials for an AWS Identity and Access Management (IAM) role or user in the seller AWS account. The role or user must have the correct policies in place for both AWS Marketplace and Amazon ECR. For example, you could use the following AWS managed policies:

- `AWSMarketplaceSellerProductsFullAccess` – For access to AWS Marketplace
- `AmazonEC2ContainerRegistryFullAccess` – For access to Amazon ECR

Log your Docker client into AWS

Set a variable for the AWS Region that you want to publish from (see [Supported AWS Regions for publishing \(p. 211\)](#)). For this example, use the US East (Ohio) Region.

```
region=us-east-2
```

Run the following command to set a variable with your AWS account ID. This example assumes that the current AWS Command Line Interface (AWS CLI) credentials belong to the seller's AWS account.

```
account=$(aws sts get-caller-identity --query Account --output text)
```

To authenticate your Docker CLI client with your AWS account Amazon ECR Docker registry for your Region, run the following command.

```
aws ecr get-login-password \  
--region ${region} \  
| sudo docker login \  
--username AWS \  
--password-stdin \  
${account}.dkr.ecr.${region}.amazonaws.com
```

Create repository and upload image

Set a variable for the tag of the uploaded image and another variable for the name of the uploaded image repository.

```
image=my-inference-image  
repo=my-inference-image
```

Note

In previous sections of this guide where the inference and training images were built, they were tagged as **my-inference-image** and **my-training-image**, respectively. For this example, create and upload the inference image to a repository with the same name.

Run the following command to create the image repository in Amazon ECR.

```
aws ecr --region ${region} create-repository --repository-name "${repo}"
```

The full name of the Amazon ECR repository location is made up of the following parts: <account-id>.dkr.ecr.<region>.amazonaws.com/<image-repository-name>

To push the image to the repository, you must tag it with the full name of the repository location.

Set a variable for the full name of the image repository location along with the latest tag.

```
fullname="${account}.dkr.ecr.${region}.amazonaws.com/${repo}:latest"
```

Tag the image with the full name.

```
sudo docker tag ${image} ${fullname}
```

Finally, push the inference image to the repository in Amazon ECR.

```
sudo docker push ${fullname}
```

After the upload completes, the image appears in the [repository list of the Amazon ECR console](#) in the Region that you are publishing from. In the previous example, the image was pushed to a repository in the US East (Ohio) Region.

Scan your uploaded image

In the [Amazon ECR console](#), choose the AWS Region that you are publishing from, and open the repository that the image was uploaded to. Select your uploaded image and start a scan to check for known vulnerabilities. AWS Marketplace checks the Amazon ECR scan results of the container images used in your Amazon SageMaker resource before publishing it. Before you can create your product, you must fix container images that have vulnerabilities with either a Critical or High severity.

After your images are scanned successfully, they can be used to create a model package or algorithm resource.

If you believe that your product had errors in the scan that are false positives, contact [AWS Marketplace Seller Operations](#) with information about the error.

Next steps

- See size limits in [Requirements and best practices for creating machine learning products \(p. 204\)](#)
- Continue to [Creating your Amazon SageMaker resource \(p. 196\)](#)

Creating your Amazon SageMaker resource

To publish a model package or algorithm product, you must create the respective [model package resource](#) or [algorithm resource](#) in Amazon SageMaker.

When you create your resource for an AWS Marketplace product, it must be certified through a validation step. The validation step requires that you provide data to test your model package or algorithm resource before it can be published.

Note

If you haven't yet created the images for your product and uploaded them to Amazon Elastic Container Registry (Amazon ECR), see [Packaging your code into images \(p. 180\)](#) and [Uploading your images \(p. 194\)](#) for information about how to do so.

Creating your model package

The following are requirements for creating a model package for AWS Marketplace:

- An inference image stored in [Amazon ECR](#)
- (Optional) Model artifacts, stored separately in [Amazon S3](#)
- Your test data used for inferences, stored in Amazon Simple Storage Service (Amazon S3)

Note

The following is about creating a model package product. For more information about model packages in SageMaker, see [Create a Model Package Resource](#).

Creating the model package resources

The following procedures step you through creating the model package resources.

Step 1: To create the model package resources

1. Open the [Amazon SageMaker console](#).
2. Ensure that you are in the AWS Region that you want to publish from by looking at the top right of the page. For publishing, see the [Supported AWS Regions for publishing \(p. 211\)](#) section. The inference image you uploaded to Amazon ECR in previous steps must be in the same Region.
3. In the left navigation menu, choose **Model packages**.

4. Choose **Create model package**.

After you create the package, you need to set the specifications of the inference package.

Step 2: To set inference specifications

1. Provide a **Name** for your model package (for example, *my-model-package*).
2. For **Location of inference image**, enter the URI of your inference image that was uploaded to Amazon ECR. You can retrieve the URI by locating your image in the [Amazon ECR console](#).
3. If your model artifacts from training are bundled with your logic in your inference image, leave the **Location of model data artifacts** empty. Otherwise, specify the full Amazon S3 location of the compressed file (.tar.gz) of your model artifacts.
4. Using the dropdown box, choose the supported instance types of your inference image for both real-time inference (also known as *endpoint*) and batch-transform jobs.
5. Choose **Next**.

Before your model package can be created and published, validation is necessary to ensure that it functions as expected. This requires that you run a batch transform job with test data for inference that you provide. The validation specifications tell SageMaker how to perform the validation.

Step 3: To set validation specifications

1. Set **Publish this model package in AWS Marketplace** to **Yes**. If you set this to **No**, you can't publish this model package later. Choosing **Yes** *certifies* your model package for AWS Marketplace and requires the validation step.
2. If this is the first time completing this process, choose **Create a new role** for the **IAM role**. Amazon SageMaker uses this role when it deploys your model package. This includes actions, such as pulling images from Amazon ECR and artifacts from Amazon S3. Review the settings, and choose **Create role**. Creating a role here grants permissions described by the [AmazonSageMakerFullAccess](#) IAM policy to the role that you create.
3. Edit the **JSON** in the validation profile. For details about allowed values, see [TransformJobDefinition](#).
 1. `TransformInput.DataSource.S3Uri`: Set to where your test data for inference is stored.
 2. `TransformInput.ContentType`: Specify your test data content type (for example, `application/json`, `text/plain`, `image/png`, or any other value). SageMaker does not validate the actual input data. This value is passed to your container HTTP endpoint in the `Content-type` header value.
 3. `TransformInput.CompressionType`: Set to `None` if your test data for inference in Amazon S3 is not compressed.
 4. `TransformInput.SplitType`: Set to `None` to pass each object in Amazon S3 as a whole for inference.
 5. `TransformOutput.S3OutputPath`: Set to the location that the inference output is stored.
 6. `TransformOutput.AssembleWith`: Set to `None` to output each inference as separate objects in Amazon S3.
4. Choose **Create model package**.

SageMaker pulls the inference image from Amazon ECR, copies any artifacts to the inference container, and runs a batch transform job using your test data for inference. After the validation succeeds, the status changes to **Completed**.

Note

The validation step does not evaluate the accuracy of the model with your test data. The validation step checks if the container runs and responds as expected.

You have completed creating your model product resources. Continue to [Publishing your product in AWS Marketplace \(p. 200\)](#).

Creating your algorithm

The following are requirements for creating an algorithm for AWS Marketplace:

- An inference image, stored in Amazon ECR
- A training image, stored in Amazon ECR
- Your test data for training, stored in Amazon S3
- Your test data for inference, stored in Amazon S3

Note

The following walkthrough creates an algorithm product. For more information, see [Create an Algorithm Resource](#).

Creating the algorithm resources

The following procedures step you through creating the resources in your algorithm package.

Step 1: To create the algorithm resources

1. Open the [Amazon SageMaker console](#).
2. Ensure that you are in the AWS Region that you want to publish from by looking at the top right of the page (see [Supported AWS Regions for publishing \(p. 211\)](#)). The training and inference images you uploaded to Amazon ECR in previous steps must be in this same Region.
3. In the left navigation menu, choose **Algorithms**.
4. Choose **Create algorithm**.

After you have created the algorithm package, you must set the specifications for the training and tuning of your model.

Step 2: To set the training and tuning specifications

1. Enter the **Name** for your algorithm (for example, *my-algorithm*).
2. For **Training image**, paste the full URI location of your training image that was uploaded to Amazon ECR. You can retrieve the URI by locating your image in the [Amazon ECR console](#).
3. Using the dropdown box, choose the **instance types for training** that your training image supports.
4. Under the **Channel specification** section, add a channel for each input dataset that your algorithm supports, up to 20 channels of input sources. For more information, see [Input Data Configuration](#).
5. Choose **Next**.
6. If your algorithm supports hyperparameters and hyperparameter tuning, you must specify the tuning parameters.
7. Choose **Next**.

Note

We highly recommend that your algorithm supports hyperparameter tuning and makes appropriate parameters tunable. This allows data scientists to tune models to get the best results.

After you have set the tuning parameters, if any, you must set the specifications for your inference image.

Step 3: To set inference image specification

1. For **Location of inference image**, paste the URI of the inference image that was uploaded to Amazon ECR. You can retrieve the URI by locating your image in the [Amazon ECR Console](#).
2. Using the dropdown box, choose the supported instance types for your inference image for both real-time inference (also known as *endpoint*) and batch-transform jobs.
3. Choose **Next**.

Before your algorithm can be created and published, validation is necessary to ensure that it functions as expected. This requires that you run both a training job with test data for training and a batch transform job with test data for inference that you provide. The validation specifications tell SageMaker how to perform the validation.

Step 4: To set validation specifications

1. Set **Publish this algorithm in AWS Marketplace** to **Yes**. If you set this to **No**, you can't publish this algorithm later. Choosing **Yes** [certifies](#) your algorithm for AWS Marketplace and requires the validation specification.
2. If this is your first time creating a machine learning package for AWS Marketplace, choose **Create a new role** for the **IAM role**. Amazon SageMaker uses this role when training your algorithm and deploying the subsequent model package. This includes actions such as pulling images from Amazon ECR, storing artifacts in Amazon S3, and copying training data from Amazon S3. Review the settings, and choose **Create role**. Creating a role here grants permissions described by the [AmazonSageMakerFullAccess](#) IAM policy to the role that you create.
3. Edit the **JSON** file in the validation profile for **Training job definition**. For more information about allowed values, see [TrainingJobDefinition](#).
 1. **InputDataConfig**: In this JSON array, add a [Channel object](#) for each channel that you specified in the training-specification step. For each channel, specify where your test data for training is stored.
 2. **OutputDataConfig**: After the training completes, the model artifacts in the training container directory path `/opt/ml/model/` are compressed and copied out to Amazon S3. Specify the Amazon S3 location of where the compressed file (`.tar.gz`) is stored.
4. Edit the JSON file in the validation profile for **Transform job definition**. For more information about allowed values, see [TransformJobDefinition](#).
 1. **TransformInput.DataSource.S3Uri**: Set to where your test data for inference is stored.
 2. **TransformInput.ContentType**: Specify your test data content type. For example, `application/json`, `text/plain`, `image/png`, or any other value. Amazon SageMaker does not validate the actual input data. This value is passed to your container HTTP endpoint in the `Content-type` header value.
 3. **TransformInput.CompressionType**: Set to `None` if your test data for inference in Amazon S3 is not compressed.
 4. **TransformInput.SplitType**: Choose how you want objects in S3 split. For example, `None` passes each object in Amazon S3 as a whole for inference. For more details, see [SplitType](#) in the Amazon SageMaker API Reference.
 5. **TransformOutput.S3OutputPath**: Set to the location where the inference output is stored.
 6. **TransformOutput.AssembleWith**: Set to `None` to output each inference as separate objects in Amazon S3.
5. Choose **Create algorithm package**.

SageMaker pulls the training image from Amazon ECR, runs a test-training job using your data, and stores the model artifacts in Amazon S3. It then pulls the inference image from Amazon ECR, copies the

artifacts from Amazon S3 into the inference container, and runs a batch transform job using your test data for inference. After the validation succeeds, the status changes to **Completed**.

Note

The validation step does not evaluate the accuracy of the training or the model with your test data. The validation step checks if the containers run and respond as expected.

The validation step only validates batch processing. It is up to you to validate that real-time processing works with your product.

You have completed creating your algorithm product resources. Continue to [Publishing your product in AWS Marketplace \(p. 200\)](#).

Publishing your product in AWS Marketplace

Before you can publish your model package or algorithm, the following are required:

- An AWS account that is registered as an AWS Marketplace seller. You can do this in the [AWS Marketplace Management Portal](#).
- A completed seller profile under the [Settings](#) page in the AWS Marketplace Management Portal.
- For publishing paid products, you must complete the tax interview and bank forms. This is not required for publishing free products. For more information, see [Seller registration process](#).
- You must have permissions to access the AWS Marketplace Management Portal and Amazon SageMaker. For more information, see [Permissions required \(p. 200\)](#).

Overview of publishing process

There are four steps in the publishing process:

1. **Submit product** – Create a listing with the description, usage information, and other details of your model package or algorithm product. After you submit your product for publishing, it takes about an hour until the status changes to the next step.
2. **Test product** – Use your AWS account that is registered as an AWS Marketplace seller to preview the listing in the AWS Marketplace, subscribe to it, and test the product. In addition, other allowed AWS accounts can preview and test the product. If any changes are necessary, you can go back and edit the listing details.
3. **Sign off for publishing** – When your product is ready to go live, return to the AWS Marketplace Management Portal, and choose **Sign off and publish**.
4. **Product goes live** – Your product is now live in the AWS Marketplace. You can maintain your product by publishing new versions with updates or product fixes.

Permissions required

To publish an Amazon SageMaker product, the AWS Identity and Access Management (IAM) user or role you are logged in as requires one or both of the following IAM actions:

- **sagemaker:DescribeModelPackage** – For listing a model package
- **sagemaker:DescribeAlgorithm** – For listing an algorithm

For the AWS Marketplace permissions needed, or for managing your seller account, see [Policies and permissions for AWS Marketplace sellers](#).

Creating your product listing

The following is a walkthrough for creating your product listing in the AWS Marketplace for both model package and algorithm products.

Note

Before creating your listing, ensure that you have the required resources specified in [Requirements and best practices for creating machine learning products \(p. 204\)](#).

To create a machine learning product

1. While logged into your seller AWS account, navigate to the [AWS Marketplace Management Portal](#).
2. In the top menu, navigate to **Products** and then **Machine learning**.
3. Choose **Create new listing**.

After you have created your listing, you must provide general product information, set up the launch option for the first version of your product, set up the pricing for your product, and finally submit the product. The following procedures take you through each of these steps.

Step 1: To provide general product information

1. Choose **Add** for **Product descriptions**.
 - a. For the **Product visibility** section, choose one of the following options:
 - **Public** – The product will initially be available to a limited set of AWS accounts for testing. After you sign off and publish it, the product will be publicly discoverable and available for subscription by all customers.
 - **Private** – The product will only be visible to the AWS accounts that you specify. You will not be able to make this product public in the future.
 - b. Enter **Product title**, **Short product description**, **Product overview**, **Product category 1**, and other details. For product descriptions, see [Requirements and best practices for creating machine learning products \(p. 204\)](#).
 - c. Choose **Continue** when complete. You can change these values later.
2. Continue to **Promotional Resources**, and provide a product logo and relevant links. Choose **Continue** when complete. You can change these values later.
3. Continue to **Support Information**, provide support and contact details. Choose **Continue** when complete. You can change these values later.
4. Continue to **Region Availability**, and choose the specific AWS Regions you want to list your product in. The default value is **Make available in all current and future supported Regions**. Choose **Continue** when complete. After you submit your draft for publishing, you can't change this selection.

Next, you're ready to provide the launch option, which is the model or algorithm that you're selling.

Step 2: To add your launch option

1. For **Enter ARN**, paste the Amazon Resource Name (ARN) of your model package or algorithm. You can find these in the Amazon SageMaker console [Model Packages](#) or [Algorithms](#) pages.
 1. An ARN for a model package appears as the following:
`arn:aws:sagemaker:<region>:<account-id>:model-package/<model-package-name>`
 2. An ARN for an algorithm appears as the following:
`arn:aws:sagemaker:<region>:<account-id>:algorithm/<algorithm-name>`

2. Choose **Add**.
3. This step differs depending on if you publish a model package or algorithm product. With the exception of the buyer-facing version number, you can change the version details later.
 1. Provide the version number, release notes, and URLs to the sample Jupyter notebook and GitHub repository.
 2. For *algorithm products* only, on the next page, describe the training data and include an example training data resource along with an overview of the training algorithm. The algorithm metrics, channel specification, and hyperparameters will be automatically displayed on the product detail page based on the values you provided when you created the algorithm resource in SageMaker.
 3. On the next page, provide model input details and URLs for the sample input files.
 4. On the next page, provide the model output details and sample outputs as text or URLs. For usage information, see [Requirements and best practices for creating machine learning products \(p. 204\)](#).
 5. On the next page, set the recommended instances. If this is a model package product, choose the recommended instance type from your supported instances for both the batch transform and real-time deployments. If this is an algorithm product, choose the recommended instance type training jobs, as well. You can't choose instance types that your model package or algorithm resource doesn't support. The supported instance types were selected when you created those resources in Amazon SageMaker.
4. Choose **Continue** when complete.

Note

Clear usage information that describes the expected inputs and outputs of your product (with examples) is crucial for supporting a positive buyer experience. For more information, see [Requirements and best practices for creating machine learning products \(p. 204\)](#).

The next step in publishing your product is to set the pricing and terms.

Step 3: To set the pricing and terms

1. Choose **Add offer**.
2. Set your **Pricing**. You can provide your software for free, set your paid pricing, or enable a free trial period. For more information, see [Machine learning product pricing \(p. 178\)](#).
3. Upload a plaintext file to use as your End User License Agreement (EULA).
4. Choose **Save and close**.

You have provided all the information for your product. The next step is to publish it to limited availability so that you can test the product.

Step 5: To submit your product for publishing

- For **Product overview**, choose **Submit for publishing**. This starts the publishing process by creating a preview listing in AWS Marketplace that you can subscribe to and use for testing.

Note

For more information about testing your machine learning product, see [Testing your product \(p. 203\)](#).

After testing your product, you can redo the steps above if there are any changes that need to be made. When you're ready for your product to be available to buyers in AWS Data Exchange, you can sign off for publishing.

Step 4: To sign off for publishing

- After testing your product and writing your descriptions, pricing, and usage information, return to the product overview, and choose **Sign off and publish**.

Testing your product

After the initial submission of your product, it takes about an hour for your preview listing to be ready. After the status changes to **Test Product**, your seller account and other allow-listed AWS accounts can preview the listing in AWS Marketplace, subscribe to the product, and test it. To see a preview of your listing, in the AWS Marketplace Management Portal, navigate to the product overview page, and choose **Go to staged product**.

To add other AWS accounts to test your product before publishing, contact [AWS Marketplace Seller Operations](#) and provide the AWS account IDs. Allow-listed accounts display a **Limited** badge alongside the product version on the product detail page.

If you want to make changes, choose **Edit product** and follow the same steps as creating your product. When you're ready for your product to be published publicly for all buyers to see, return to the management portal, and choose **Sign off and publish**.

Updating your product

To update your model package or algorithm product, navigate to the [Machine Learning Listings](#) page in the AWS Marketplace Management Portal. You can add new model package or algorithm resources as new versions of your existing product. You can also restrict previous versions or remove your entire product.

Adding new versions

To add new versions of your model package or algorithm resources, navigate to the **Product Overview** of your existing product, and then use the follow procedure.

To add a new version

1. Choose **Edit product**.
2. Under **Launch option**, choose **Edit**.
3. To add the ARN of your resource, navigate to the **Version** page, and choose **Add new version**. For more information about adding a launch option, see [Creating your product listing \(p. 201\)](#).

Note

Usage information is specific to each product version. Continue to follow the [Requirements and best practices for creating machine learning products \(p. 204\)](#) when adding usage information to new versions.

When your buyers launch your product from its AWS Marketplace listing, they can choose different versions. When your buyers launch your product from the Amazon SageMaker console, only the latest version is visible.

Restricting versions

To restrict versions of your model package or algorithm resources, navigate to the **Product Overview** page of your existing product, and then use the following procedure.

To restrict a version

1. Choose **Edit product**.
2. Under **Launch option**, choose **Edit**.
3. On the **Version** page, choose **Restrict version**.
4. Return to the **Product Overview**, and choose **Submit for publishing**.

Note

Buyers that have already subscribed to your product can continue to use restricted versions of your model package or algorithm. However, new buyers will not be able to see those restricted versions as options.

Remove a product

To remove a product, navigate to your list of published products in the [Machine Learning Listings](#) page in the AWS Marketplace Management Portal. Choose the product you want to remove, and in the **Actions** dropdown list, choose **Unpublish listing**.

Provide an email address and a reason to remove your listing, in the event that an AWS Marketplace representative contacts you regarding your request.

Note

When you remove a product from AWS Marketplace, new buyers can no longer subscribe to your product. However, existing buyers can continue using your product, which must be supported for a minimum of 90 days. If you plan to have another product replace the unpublished listing, indicate the new listing in the details of your removal request.

Requirements and best practices for creating machine learning products

It is important that your buyers find it easy to test your model package and algorithm products. The following sections describe the requirements for creating machine learning (ML) product listings and best practices for ML products. For a complete summary of requirements and recommendations, see the [Summary of requirements and recommendations for ML product listings \(p. 207\)](#).

Note

An AWS Marketplace representative might contact you to help you meet these requirements if your published products don't meet them.

Topics

- [Required assets \(p. 204\)](#)
- [General best practices for ML products \(p. 205\)](#)
- [Requirements for usage information \(p. 205\)](#)
- [Requirements for inputs and outputs \(p. 205\)](#)
- [Requirements for Jupyter notebook \(p. 206\)](#)
- [Summary of requirements and recommendations for ML product listings \(p. 207\)](#)

Required assets

Before creating a machine learning product listing, ensure that you have the following required assets:

- **Amazon Resource Name (ARN)** – Provide the ARN of the model package or algorithm resource in the AWS Region that you are publishing from (see [Supported AWS Regions for publishing \(p. 211\)](#)).
 - An ARN for a model package has this form: `arn:aws:sagemaker:<region>:<account-id>:model-package/<model-package-name>`
 - An ARN for an algorithm has this form: `arn:aws:sagemaker:<region>:<account-id>:algorithm/<algorithm-name>`
- the section called “Requirements for usage information” (p. 205) – Provide details about inputs, outputs, and code examples.
- the section called “Requirements for inputs and outputs” (p. 205) – Provide either files or text.
- the section called “Requirements for Jupyter notebook” (p. 206) – Demonstrate complete product usage.

General best practices for ML products

Provide the following information for your machine learning product:

- For product descriptions, include the following:
 - What your model does
 - Who the target customer is
 - What the most important use case is
 - How your model was trained or the amount of data that was used
 - What the performance metrics are and the validation data used
 - If medical, whether or not your model is for diagnostic use
- Optionally, for paid products, offer a free trial of 14–30 days for customers to try your product. For more information, see [Machine learning product pricing \(p. 178\)](#).
- Optionally, for model package products, if you want to enable a real-time product demo on your product listing page, contact [AWS Marketplace Seller Operations](#). The product demo allows a prospective buyer to try your model directly on the listing page without subscribing to or deploying the model themselves.

Requirements for usage information

Clear usage information that describes the expected inputs and outputs of your product (with examples) is crucial for driving a positive buyer experience.

With each new version of your resource that you add to your product listing, you must provide usage information.

To add usage information for a new product that you are publishing for the first time, sign into the AWS Marketplace Management Portal console. From the Products dropdown, choose **Machine learning**. Select your product. In the **Product Overview** under **Launch option**, provide the ARN of your model package or algorithm resource, and choose **Add**.

To edit the existing usage information for a specific version, choose **Edit** under **Launch option** and then **Edit version**.

Requirements for inputs and outputs

A clear explanation of your format, with examples of inputs and outputs, is important to help your buyers to understand and use your product. This understanding helps your buyers to perform any necessary transformations on the input data to get the best inference results.

You will be prompted for the following when adding your SageMaker resource to your product listing.

Inference inputs and outputs

For inference input, provide the input format for both the real-time endpoint and batch transform job. Include code snippets for any necessary preprocessing of the data. Include supported MIME content types (for example, **image/jpeg**, **image/png**, **image/bmp**), descriptions of values if applicable, and limitations. Include input samples hosted on [GitHub](#).

For inference output, provide the output format for the both real-time endpoint and batch transform job. Include output MIME content type (for example, **application/json**, **image/jpeg**) and description of values if applicable. Include output samples hosted on [GitHub](#).

For samples, provide input files that work with your product. If your model performs multiclass classification, provide at least one sample input file for each class.

Training inputs

In the **Information to train a model** section, provide the input data format and code snippets for any necessary preprocessing of the data. Include supported MIME content types (for example, **image/jpeg**, **image/png**, **image/bmp**), description of values if applicable, and limitations. Ensure to include input samples hosted on [GitHub](#).

Explain both optional and mandatory features that can be provided by the buyer, and specify whether the PIPE input mode is supported. If [distributed training](#) (training with more than 1 CPU/GPU instance) is supported, specify this. For tuning, list the recommend hyperparameters.

Requirements for Jupyter notebook

When adding your SageMaker resource to your product listing, provide a link to a sample Jupyter notebook hosted on [GitHub](#) that demonstrates the complete workflow without asking the buyer to upload or find any data.

Use the AWS SDK for Python (Boto). A well-developed sample notebook makes it easier for buyers to try and use your listing.

For model package products, your sample notebook demonstrates the preparation of input data, creation of an endpoint for real-time inference, and performance of batch-transform jobs. For more information, see [Model Package listing and Sample notebook](#) on GitHub. For sample notebooks, see [generic_sample_notebook](#) and [auto_insurance](#). The latter sample notebook works in all Regions, without entering any parameters and without a buyer needing to locate sample data.

Note

An underdeveloped sample Jupyter notebook that does not show multiple possible inputs and data preprocessing steps might make it difficult for the buyer to fully understand your product's value proposition.

For algorithm products, the sample notebook demonstrates complete training, tuning, model creation, the creation of an endpoint for real-time inference, and the performance of batch-transform jobs (see [Algorithm listing and Sample notebook](#) on GitHub). For sample notebooks, see [amazon_demo_product](#) and [automl](#) on GitHub. These sample notebooks work in all Regions without entering any parameters and without a buyer needing to locate sample data.

Note

A lack of example training data might prevent your buyer from running the Jupyter notebook successfully. An underdeveloped sample notebook might prevent your buyers from using your product and hinder adoption.

Summary of requirements and recommendations for ML product listings

The following table provides a summary of the requirements and recommendations for a machine learning product listing page.

| Details | For model package listings | For algorithm listings |
|---|----------------------------|------------------------|
| Product descriptions | | |
| Explain in detail what the product does for supported content types (for example, "detects X in images"). | Required | Required |
| Provide compelling and differentiating information about the product (avoid adjectives like "best" or unsubstantiated claims). | Recommended | Recommended |
| List most important use case(s) for this product. | Required | Required |
| Describe the data (source and size) it was trained on and list any known limitations. | Required | Not applicable |
| Describe the core framework that the model was built on. | Recommended | Recommended |
| Summarize model performance metric on validation data (for example, "XX.YY percent accuracy benchmarked using the Z dataset"). | Required | Not applicable |
| Summarize model latency and/or throughput metrics on recommended instance type. | Required | Not applicable |
| Describe the algorithm category. For example, "This decision forest regression algorithm is based on an ensemble of tree-structured classifiers that are built using the general technique of bootstrap aggregation and a random choice of features." | Not applicable | Required |
| Usage information | | |
| For inference, provide the input format for both the real-time endpoint and batch transform job. Include supported MIME content types (for example, | Required | Required |

| Details | For model package listings | For algorithm listings |
|--|----------------------------|------------------------|
| <p>image/jpeg, image/png, image/bmp), description of values if applicable, and limitations. See Requirements for inputs and outputs (p. 205).</p> | | |
| <p>For inference, provide input samples for both the real-time endpoint and batch transform job. Samples must be hosted on GitHub. See Requirements for inputs and outputs (p. 205).</p> | Required | Required |
| <p>For inference, provide the output format for both the real-time endpoint and batch transform job. Include output MIME content type (for example, application/json, image/jpeg) and description of values if applicable. See Requirements for inputs and outputs (p. 205).</p> | Required | Required |
| <p>For inference, provide output samples for both the real-time endpoint and batch transform job. Samples must be hosted on GitHub. See Requirements for inputs and outputs (p. 205).</p> | Required | Required |
| <p>For inference, provide an example of using an endpoint or batch transform job. Include a code example using the AWS Command Line Interface (AWS CLI) commands or using an AWS SDK.</p> | Required | Required |
| <p>For training, provide input format. Include supported MIME content types (for example, image/jpeg, image/png, image/bmp), description of values if applicable, and limitations (for example, minimum rows of data required). See Requirements for inputs and outputs (p. 205).</p> | Not applicable | Required |
| <p>For training, provide input samples hosted on GitHub. See Requirements for inputs and outputs (p. 205).</p> | Not applicable | Required |

| Details | For model package listings | For algorithm listings |
|--|----------------------------|------------------------|
| For training, provide an example of performing training jobs. Describe the supported hyperparameters, their ranges, and their overall impact. Specify if the algorithm supports hyperparameter tuning, distributed training, or GPU instances. Include code example such as AWS CLI commands or using an AWS SDK, for example. | Not applicable | Required |
| Provide a Jupyter notebook hosted on GitHub demonstrating complete use of your product. See Requirements for Jupyter notebook (p. 206) . | Required | Required |
| Provide technical information related to the usage of the product, including user manuals and sample data. | Recommended | Recommended |

Service restrictions and quotas

This section describes restrictions and quotas on your machine learning (ML) products in AWS Marketplace.

Network isolation

For security purposes, when a buyer subscribes to your containerized product, the Docker containers are run in an isolated environment without network access. When you create your containers, don't rely on making outgoing calls over the internet because they will fail. Calls to AWS services will also fail.

Image size

Your Docker image size is governed by the Amazon Elastic Container Registry (Amazon ECR) [service quotas](#). The Docker image size affects the startup time during training jobs, batch-transform jobs, and endpoint creation. For better performance, maintain an optimal Docker image size.

Storage size

When you create an endpoint, Amazon SageMaker attaches an Amazon Elastic Block Store (Amazon EBS) storage volume to each ML compute instance that hosts the endpoint. (An endpoint is also known as *real-time inference* or *Amazon SageMaker hosting service*.) The size of the storage volume depends on the instance type. For more information, see [Host Instance Storage Volumes](#). For batch transform, keep these limits in mind.

Instance size

SageMaker provides a selection of instance types that are optimized to fit different ML use cases. Instance types are comprised of varying combinations of CPU, GPU, memory, and networking capacity. Instance types give you the flexibility to choose the appropriate mix of resources for building, training, and deploying your ML models. For more information, see [Amazon SageMaker ML Instance Types](#).

Payload size for inference

For an endpoint, the maximum size of the input data per invocation is 25 MB. This value can't be adjusted. If your software expects larger input sizes, your buyers can deploy endpoints configured for [Amazon SageMaker Asynchronous Inference](#). If Asynchronous Inference is needed, provide detailed usage information so that your buyers configure the endpoint properly.

For batch transform, the maximum size of the input data per invocation is 100 MB. This value can't be adjusted.

Processing time for inference

For an endpoint, the maximum processing time per invocation is 60 seconds. If your software requires longer processing times, your buyers can deploy endpoints configured for [Amazon SageMaker Asynchronous Inference](#). If Asynchronous Inference is needed, provide detailed usage information so that your buyers configure the endpoint properly.

For batch transform, the maximum processing time per invocation is 60 minutes. This value can't be adjusted.

Service quotas

For more information about quotas related to training and inference, see [Amazon SageMaker Service Quotas](#).

Managed spot training

For all algorithms from AWS Marketplace, the value of `MaxWaitTimeInSeconds` is set to 3,600 seconds (60 minutes), even if the checkpoint for [managed spot training](#) is implemented. This value can't be adjusted.

Docker images and AWS accounts

For publishing, images must be stored in Amazon ECR repositories owned by the AWS account of the seller. It isn't possible to publish images that are stored in a repository owned by another AWS account.

Publishing model packages from built-in algorithms or AWS Marketplace

Model packages created from training jobs using an [Amazon SageMaker built-in algorithm](#) or an algorithm from an AWS Marketplace subscription can't be published.

You can still use the model artifacts from the training job, but your own inference image is required for publishing model packages.

Supported AWS Regions for publishing

AWS Marketplace supports publishing model package and algorithm resources from AWS Regions where the following are both true:

- A Region that [Amazon SageMaker supports](#)
- An [available Region](#) that is opted-in by default (for example, [describe-regions](#) returns "OptInStatus": "opt-in-not-required")

All assets required for publishing a model package or algorithm product must be stored in the same Region that you choose to publish from. This includes the following:

- Model package and algorithm resources that are created in Amazon SageMaker
- Inference and training images that are uploaded to Amazon ECR repositories
- Model artifacts (if any) that are stored in Amazon Simple Storage Service (Amazon S3) and dynamically loaded during model deployment for model package resources
- Test data for inference and training validation that are stored in Amazon S3

You can develop and train your product in any Region that is supported by SageMaker. But, before you can publish, you must copy all assets to and re-create resources in a Region that AWS Marketplace supports publishing from.

During the listing process, regardless of the AWS Region that you publish from, you can choose the Regions that you want to publish to and make your product available in.

Troubleshooting

This section provides help for some common errors that you might encounter during the publishing process of your machine learning product. If your issue isn't listed, contact [AWS Marketplace Seller Operations](#).

General: I get a 400 error when I add the Amazon Resource Name (ARN) of my model package or algorithm in the AWS Marketplace Management Portal

If you used the Amazon SageMaker console to create your resource, you must choose **Yes** on the final page of the process for **Publish this model package in AWS Marketplace** or **Yes** for **Publish this algorithm in AWS Marketplace**. You can't choose **No** and later publish it. Selecting **Yes** doesn't publish the model package or algorithm. However, it validates your model package or algorithm resource when it is created, which is necessary for use in AWS Marketplace.

If you're using the AWS SDK to [create a model package](#) or [create an algorithm](#), ensure that the parameter `CertifyForMarketplace` is set to `true`.

After you re-create your certified and validated model package or algorithm resource, add the new ARN in the AWS Marketplace Management Portal.

General: I get a 404 error when I add the ARN of my model package or algorithm in the AWS Marketplace Management Portal

This error can happen for several reasons:

- The ARN might be invalid. Ensure that you are using the correct ARN.
 - For model packages, the ARNs should look similar to `arn:aws:sagemaker:us-east-2:000123456789:model-package/my-model-package-name`.

- For algorithms, the ARNs should look similar to `arn:aws:sagemaker:us-east-2:000123456789:algorithm/my-algorithm`.
- The model package or algorithm resource wasn't created in the same AWS account as the seller account. Ensure that all resources and assets for publishing are in the seller account that you are publishing from.
- The AWS Identity and Access Management (IAM) user or role that you use for publishing doesn't have the correct IAM permissions to access the model package or algorithm resource. Ensure that your IAM user or role has the following permissions:
 - For model packages, the action `sagemaker:DescribeModelPackage` on the model package resource must be allowed.
 - For algorithms, the action `sagemaker:DescribeAlgorithm` on the algorithm resource must be allowed.

General: I get a 500 error when I specify the pricing for my algorithm product in the AWS Marketplace Management Portal

This error can happen when you attempt to publish an algorithm resource with only a training image and without an accompanying inference image. Algorithm resources that are published on AWS Marketplace must have both components. For more information, see [Prepare your product in SageMaker \(p. 180\)](#).

Amazon SageMaker: I get a “Client error: Access denied for registry” failure message when I create a model package or algorithm resource

This error can happen when the image that is being used to create the model package or algorithm is stored in an [Amazon ECR](#) repository that belongs to another AWS account. Model package or algorithm validation does not support cross-account images. Copy the image to an Amazon ECR repository owned by the AWS account that you are using to publish. Then, proceed with creating the resource using the new image location.

Amazon SageMaker: I get “Not Started” and “Client error: No scan scheduled...” failure messages when I create a model package or algorithm resource

This error can happen when SageMaker fails to start a scan of your Docker container image stored in Amazon ECR. If this happens, open the [Amazon ECR console](#), find the repository where your image was uploaded to, choose the image, and then choose **Scan**.

Reporting

AWS Marketplace produces reports for your Amazon SageMaker products that include data about buyers, financials, usage, and taxes. All reports are available in the AWS Marketplace Management Portal on the [Reports page](#). For more information, see [Seller Reports](#).

The following subsections summarize how financials for your machine learning products that use SageMaker are reported.

Daily business report

The daily business report provides the instance type, hours of usage, revenue from software charges, and other details for each buyer and product. Buyers are identified by a unique and anonymous Customer Reference ID. For more information, see [Daily business report](#).

Monthly revenue report

The monthly revenue report provides you with the monthly revenue that has been billed to your buyers for using your software. For more information, see [Monthly billed revenue report](#).

Disbursement report

The monthly disbursement report provides a breakdown of all funds collected on your behalf during the settlement period for your software charges. The total settlement amount reflected in the report should match the amount deposited to your bank account. For more information, see [Disbursement report](#).

Other reports and analysis

For other available reports, see [Seller reports](#).

You can also create custom reports using the available [Data feeds \(p. 299\)](#) from AWS Marketplace.

Software as a service (SaaS)–based products

With software as a service (SaaS) products, you deploy software hosted on AWS infrastructure and grant buyers access to the software in your AWS environment. You are responsible for managing customer access, account creation, resource provisioning, and account management within your software.

For assistance with your SaaS products, [contact us](#).

Topics

- [Getting started with SaaS products \(p. 214\)](#)
- [Plan your SaaS product \(p. 224\)](#)
- [SaaS product guidelines \(p. 226\)](#)
- [SaaS product pricing \(p. 227\)](#)
- [SaaS customer onboarding \(p. 232\)](#)
- [Amazon SNS notifications for SaaS products \(p. 234\)](#)
- [Accessing the AWS Marketplace Metering and Entitlement Service APIs \(p. 236\)](#)
- [Reporting \(p. 242\)](#)
- [Code examples for SaaS product integration \(p. 243\)](#)
- [Using AWS PrivateLink with AWS Marketplace \(p. 247\)](#)

Getting started with SaaS products

This chapter outlines how Software as a service (SaaS) products work for sellers who create and maintain them. This section describes how to get your SaaS product on AWS Marketplace and how to integrate it with the appropriate AWS Marketplace APIs, based on the SaaS product's billing model.

Prerequisites

Before you get started, you must complete the following prerequisites:

1. Access and use the [AWS Marketplace Management Portal](#). This is the tool that you use to register as a seller and manage the products that you sell on AWS Marketplace. For more information, see [AWS Marketplace Management Portal \(p. 5\)](#).
2. Register as a seller, and submit your tax and banking information. For more information, see [Seller registration process \(p. 6\)](#).
3. Plan how you'll create and integrate your SaaS product in AWS Marketplace. For more information, see [Plan your SaaS product \(p. 224\)](#).

Topics

- [Creating a SaaS product \(p. 215\)](#)
- [Create an initial SaaS product page \(p. 216\)](#)

- [Integrate your SaaS subscription product \(p. 216\)](#)
- [Integrate your SaaS contract product \(p. 219\)](#)
- [Integrate your SaaS contract with pay-as-you-go product \(p. 221\)](#)
- [Deploy a serverless SaaS integration solution \(p. 223\)](#)

Creating a SaaS product

To sell software as a service (SaaS) products, you create the SaaS product in AWS Marketplace, integrate it with AWS Marketplace, test the integration, and release the product to customers. The following steps explain the process in more detail.

To create a SaaS product in AWS Marketplace

1. Decide to list a SaaS product

Have a SaaS product that you would like to sell in AWS Marketplace. Review and understand how to [Plan your SaaS product \(p. 224\)](#).

2. Determine pricing and offer type

There are three offer types for SaaS products: subscriptions, contracts, and contracts with pay-as-you-go . Your choice of offer type affects how you integrate your SaaS product with AWS Marketplace. For more information, see [Plan your pricing \(p. 224\)](#).

3. Collect assets

Collect the assets that you will need to use to submit your product. Assets for your product include:

- Product logo URL – A publicly accessible URL that contains a clear image of the logo for the product you are providing.
- End User License Agreement (EULA) URL – Your product must have a EULA, and you must provide a link to it for customers to read and review on your product's AWS Marketplace page.
- Product registration URL – This URL is where customers are sent after subscribing to your product in AWS Marketplace.
- Metadata about your product – You provide the metadata in the product creation wizard of the AWS Marketplace Management Portal.
- Support information for your product – This includes email addresses and URLs for your product's support channels.

4. Submit your product for integration

[Create an initial SaaS product page \(p. 216\)](#) from your seller account using AWS Marketplace Management Portal. AWS Marketplace will publish your product as a limited product, which means that it's only available to your accounts to use for integration and testing. The AWS Marketplace Operations Team will send you an email message with your product code, Amazon Simple Notification Service (Amazon SNS) topics, and product page URL. With that information, you will have an environment to use for creating and testing your integration with AWS Marketplace in your product. Use the email message that you received from the AWS Marketplace Operations team for correspondence regarding the product.

5. Integrate with AWS Marketplace

Your product must support customers onboarding and using your product, including validating their subscription before giving them access, and, in some cases, metering for their usage. How you integrate with AWS Marketplace depends on the offer type you're using for your product. For more information about integration, based on offer type, see the following topics:

- [Subscription integration](#)

- [Contract integration](#)
- [Contract with pay-as-you-go integration](#)

The final step of integrating your product with AWS Marketplace is to test it to ensure that the integration works properly.

6. **Submit your product for launch**

After you have verified your integration, and you're ready for the product to be live, submit it to the AWS Marketplace Operations team (using the email case created earlier) for end-to-end testing and launch.

7. **Launch**

After end-to-end testing is complete, you need to review the product page with the original prices. Approve the page by responding to the email case you received when you created your product (see [Creating a SaaS product \(p. 215\)](#)). After your approval, the AWS Marketplace Operations team will make the product page live on AWS Marketplace. At this point, customers can start discovering and subscribing to your product.

Create an initial SaaS product page

Take your software as a service (SaaS) application metadata, and create a new SaaS product in AWS Marketplace catalog, using the AWS Marketplace Management Portal.

1. Sign in to the [AWS Marketplace Management Portal](#).
2. For **Products**, choose **SaaS**.
3. For **Create SaaS product**, choose **SaaS Subscriptions** or **SaaS Contracts**, depending on the type of product you want to create, and then choose **Start**.

Note

If you choose **SaaS Contracts**, you will set a contract price, or a contract with pay-as-you-go pricing when you set the pricing information for your product.

4. Review and complete the product creation wizard using the metadata and product assets you gathered. For assistance with creating your SaaS product, [contact the AWS Marketplace Operations team](#).
5. The AWS Marketplace Operations team publishes your product as a limited product that is visible to you and any AWS accounts you have allowed to view the product.

Note

Prices can be temporarily reduced so that you can test the purchase flow without incurring high charges. For more information, [contact us](#).

6. The AWS Marketplace Operations team sends an email message to the address associated with your AWS account to enable testing of product codes, Amazon Simple Notification Service (Amazon SNS) topics, and product page URLs. This is the first of several tests for your product that are required before the product can go live. You can respond to the email from the AWS Marketplace Operations team for support and requests about your product, for example, requesting that more accounts have access to the product for testing purposes. If you lose this information, or have questions, [contact us](#).

Integrate your SaaS subscription product

Integrating your product with AWS Marketplace is one step in [Creating a SaaS product \(p. 215\)](#). To integrate your software as a service (SaaS) subscription product with AWS Marketplace, you must write code and demonstrate that it can respond successfully to several customer scenarios. The following

sections describe these scenarios, how to respond to them, and provide an overview of testing your integration.

Note

Before you begin, make sure you've chosen the right pricing model for your software as a service (SaaS) product in AWS Marketplace. For more information, see [Plan your SaaS product \(p. 224\)](#).

Topics

- [Scenario: Your service validates new customers \(p. 217\)](#)
- [Scenario: Meter usage \(p. 217\)](#)
- [Scenario: Monitor changes to user subscriptions \(p. 217\)](#)
- [Scenario: Verify customer subscription \(p. 218\)](#)
- [Testing your SaaS subscription product integration \(p. 218\)](#)

Scenario: Your service validates new customers

When a customer subscribes to your product, they are redirected to your registration URL which is an HTTP POST request with a temporary `x-amzn-marketplace-token` token. Respond to this request in the following ways:

1. Exchange the token for a `CustomerIdentifier`, `CustomerAWSAccountId`, and `ProductCode` by calling the `ResolveCustomer` API operation in the AWS Marketplace Metering Service.
2. Persist the `CustomerIdentifier`, `CustomerAWSAccountID`, and `ProductCode` in your system for future calls. You must store whether the customer has a valid subscription, along with whatever information you need about the customer.
3. As a response to the request, you must show your user's first use experience (as applicable for your service).

Scenario: Meter usage

When the customer starts to use your service, you must send metering records hourly. For details on how to meter, see [Metering for usage \(p. 236\)](#).

We recommend that you use AWS CloudTrail to monitor activity to ensure that billing information is being sent to AWS. Keep the following in mind when sending metering records:

- Metering requests are de-duplicated on the hour.
- Records sent every hour are cumulative.
- We strongly recommend as a best practice that, even if there were no records in the last hour, you send metering records every hour, with usage of 0.

Scenario: Monitor changes to user subscriptions

Set up an Amazon Simple Queue Service (Amazon SQS) queue, and subscribe to your product's Amazon SNS topic. Your SNS topic information was included in the email message that you received from the AWS Marketplace Operations team when you created your product. For more information, see [Creating a SaaS product \(p. 215\)](#). By subscribing to your SNS topic, you receive notifications about changes to customer subscriptions, including providing or revoking access for specific customers.

Note

An Amazon SNS topic Amazon Resource Name (ARN) looks like `arn:aws:sns:us-east-1:<account id>:aws-mp-subscription-notification-<product code>`.

The notifications that you must respond to are:

- `subscribe-success` – The customer is subscribed, and you can successfully meter against their customer ID.
- `unsubscribe-pending` – The customer is in the process of unsubscribing. You should send any last metering records.
- `unsubscribe-success` – The customer has unsubscribed. Metering records for the customer will no longer be accepted. Follow your practices for shutting down customer resources, adhering to your retention policies.
- `subscribe-fail` – The customer subscription failed. You should not meter against their customer ID or create resources on behalf of the customer.

Scenario: Verify customer subscription

Before creating resources on the customer's behalf, verify that the customer should have access to your product. Store the latest status of the customer from the notifications you receive via Amazon SQS to know if the customer has access.

Testing your SaaS subscription product integration

After you've integrated your SaaS subscription product with AWS Marketplace, you must conduct in-depth testing to ensure that the integration is successful. The following procedure outlines the steps to verify your product integration.

Note

Use your own accounts to subscribe to your product and test that the integration is successful. Prices can be temporarily reduced so that you can test the purchase flow without incurring high charges in those accounts. For more information about temporarily reducing the prices or allowing additional test accounts to access your product, [contact us](#).

After your product is launched, the service must continue to respond to these scenarios for new customers.

1. Use an allowed account to test the customer experience by subscribing to your product.
2. After you've subscribed with the allowed account, ensure that the account is redirected to the registration URL, and that the redirect is a POST request that includes a temporary token. Make sure that your application persists the customer ID for future calls. This tests part of [Scenario: Your service validates new customers \(p. 217\)](#).
3. After verifying the test account in the previous step, onboard the account into your application. For example, you can have the test customer fill out a form to create a new user account. Or, provide them with other next steps to get access to your SaaS application. This tests part of [Scenario: Your service validates new customers \(p. 217\)](#).
4. After the test customer is onboarded, make requests that will send metering records to AWS for billing purposes by using the `BatchMeterUsage` API operation in the AWS Marketplace Metering Service. This tests [Scenario: Meter usage \(p. 217\)](#).
5. Test for subscription changes. Possible scenarios include unsubscribes, successful subscriptions, and failed subscriptions. This tests [Scenario: Monitor changes to user subscriptions \(p. 217\)](#).
6. Verify a successful subscription. After you receive an Amazon SNS notification for your test account with a successful subscription message, metering can begin. Records that are sent to the AWS Marketplace Metering Service before you receive the Amazon SNS notification aren't metered. This tests [Scenario: Verify customer subscription \(p. 218\)](#).

Note

To prevent billing issues, we strongly recommend programmatically waiting for this notification before launching resources on behalf of your customers.

7. After you have completed all of the integration requirements and tested the solution, notify the AWS Marketplace Operations team. They will run a series of final tests on the solution by verifying that you have successfully sent metered records with the `BatchMeterUsage` API operation.

After your integration and testing is complete, you can perform a final review and list your product on the public AWS Marketplace. For more information, see [Creating a SaaS product \(p. 215\)](#).

Integrate your SaaS contract product

Integrating your product with AWS Marketplace is one step in [Creating a SaaS product \(p. 215\)](#). To integrate your software as a service (SaaS) contract product with AWS Marketplace, you must write code and demonstrate that it can respond successfully to several customer scenarios. The following sections describe these scenarios, how to respond to them, and provide an overview of testing your integration.

Note

Before you begin, make sure you've chosen the right pricing model for your software as a service (SaaS) product in AWS Marketplace. For more information, see [Plan your SaaS product \(p. 224\)](#).

Topics

- [Scenario: Your service validates new customers \(p. 219\)](#)
- [Scenario: Your service handles customer requests \(p. 219\)](#)
- [Scenario: Monitor changes to user subscriptions \(p. 220\)](#)
- [Testing your SaaS contract product integration \(p. 220\)](#)

Scenario: Your service validates new customers

When a customer subscribes to your product, they are redirected to your registration URL, which is an HTTP POST request with a temporary `x-amzn-marketplace-token` token. Respond to this request in the following ways:

1. Exchange the token for a `CustomerIdentifier`, `CustomerAWSAccountId`, and `ProductCode` by calling the [ResolveCustomer](#) API operation in the AWS Marketplace Metering Service.
2. Verify the subscription and quantity (if applicable) the customer has access to by calling the [GetEntitlements](#) API operation in the AWS Marketplace Entitlement Service.
3. Persist the `CustomerIdentifier`, `CustomerAWSAccountId`, and `ProductCode` in your system for future calls. Store whether the customer has a valid subscription, along with whatever information you need about the customer.
4. As a response to the request, you must show your user's first use experience (as applicable for your service).

Scenario: Your service handles customer requests

When a customer makes a request to your service, you must respond to the following scenarios with appropriate actions or messaging:

- They don't have a customer ID in your system. This means that they have not yet subscribed. You should tell the user how to subscribe.
- They have a customer ID, and the [GetEntitlements](#) API operation returns an appropriate entitlement. In this scenario, you should fulfill the request.
- They do have a customer ID, but the [GetEntitlements](#) API operation returns no entitlement, or not enough quantity to fulfill the request. In this scenario, you must determine how to handle access and manage their experience.

Scenario: Monitor changes to user subscriptions

Set up an Amazon Simple Queue Service (Amazon SQS) queue, and subscribe to your product's Amazon SNS topic. Your SNS topic information was included in the email message that you received from the AWS Marketplace Operations Team when you created your product. For more information, see [Creating a SaaS product \(p. 215\)](#). By subscribing to your SNS topic, you receive notifications about changes to customer entitlements, including providing or revoking access for specific customers.

Note

An SNS topic Amazon Resource Name (ARN) looks like `arn:aws:sns:us-east-1:<account id>:aws-mp-entitlement-notification-<product code>`.

The only notification that you must respond to is:

- `entitlement-updated` – The customer entitlement has changed, and you must call the `GetEntitlements` API operation to see the new status. Update your customer store, and, if applicable (for example, the customer's contract has lapsed), follow your practices for shutting down customer resources, adhering to your retention policies.

Note

For additional information, see [Checking entitlements \(p. 239\)](#).

Testing your SaaS contract product integration

After you've integrated your SaaS contract product with AWS Marketplace, you must conduct in-depth testing to ensure that the integration is successful. The following procedure outlines the steps to verify your product integration.

Note

Use your own accounts to subscribe to your product and test that the integration is successful. Prices can be temporarily reduced so that you can test the purchase flow without incurring high charges in those accounts. For more information about temporarily reducing the prices or allowing additional test accounts to access your product, [contact us](#). After your product is launched, the service must continue to respond to these scenarios for new customers.

1. Use an allowed account to test the customer experience by getting a contract for your product.
2. After the account has the contract, ensure that the account is redirected to the registration URL, and that the redirect is a POST request that includes a temporary token. Make sure that your application persists the customer ID for future calls and correctly handles the entitlement the customer has. This tests part of [Scenario: Your service validates new customers \(p. 219\)](#).
3. After verifying the test account in the previous step, onboard the account into your application. For example, you can have the test customer fill out a form to create a new user account. Or, provide them with other next steps to get access to your SaaS application. This tests part of [Scenario: Your service validates new customers \(p. 219\)](#).
4. If no entitlement is returned from the `GetEntitlements` API operation, either during onboarding or in your ongoing verification passes, your application must correctly manage access and the experience for users who are not entitled. This tests [Scenario: Your service handles customer requests \(p. 219\)](#).
5. Test for subscription changes. Verify that your application correctly handles unsubscribes, successful subscription, and failed subscription scenarios. This tests [Scenario: Monitor changes to user subscriptions \(p. 220\)](#).
6. After you have completed all the integration requirements and tested the solution, notify the AWS Marketplace Operations team. They will then test the solution by verifying that you have successfully called the `GetEntitlements` API operation and sufficiently onboarded new customers.

After your integration and testing is complete, you can perform a final review and list your product on the public AWS Marketplace. For more information, see [Creating a SaaS product \(p. 215\)](#).

Integrate your SaaS contract with pay-as-you-go product

Integrating your product with AWS Marketplace is one step in [Creating a SaaS product \(p. 215\)](#). To integrate your software as a service (SaaS) contract product with AWS Marketplace, you must write code and demonstrate that it can respond successfully to several customer scenarios. The following sections describe these scenarios, how to respond to them, and provide an overview of testing your integration.

Note

Before you begin, make sure you've chosen the right pricing model for your software as a service (SaaS) product in AWS Marketplace. For more information, see [Plan your SaaS product \(p. 224\)](#).

Topics

- [Scenario: Your service validates new customers \(p. 221\)](#)
- [Scenario: Your service handles customer requests \(p. 221\)](#)
- [Scenario: Meter usage \(p. 222\)](#)
- [Scenario: Monitor changes to user entitlements \(p. 222\)](#)
- [Testing your SaaS contract product integration \(p. 222\)](#)

Scenario: Your service validates new customers

When a customer subscribes to your product, they are redirected to your registration URL, which is an HTTP POST request with a temporary `x-amzn-marketplace-token` token. Respond to this request in the following ways:

1. Exchange the token for a `CustomerIdentifier`, `CustomerAWSAccountId`, and `ProductCode` by calling the [ResolveCustomer](#) API operation in the AWS Marketplace Metering Service.
2. Verify the subscription and quantity (if applicable) the customer has access to by calling the [GetEntitlements](#) action in the AWS Marketplace Entitlement Service.
3. Persist the `CustomerIdentifier`, `CustomerAWSAccountId`, and `ProductCode` in your system for future calls. Store whether the customer has a valid subscription, along with whatever information you need about the customer.
4. As a response to the request, you must show your user's first use experience (as applicable for your service).

Scenario: Your service handles customer requests

When a customer makes a request to your service, you must respond to the following scenarios with appropriate actions or messaging:

- They don't have a customer ID in your system. This means that they have not yet subscribed. You should give them messaging describing how to subscribe.
- They have a customer ID, and the `GetEntitlements` API operation returns an appropriate entitlement. In this scenario, you should fulfill the request.
- They do have a customer ID, but the `GetEntitlements` API operation returns no entitlement, or not enough quantity to fulfill the request. In this scenario, you must determine how to handle access and manage their experience.

Scenario: Meter usage

When the customer starts to use your service, you must send metering records hourly. For details on how to meter, see [Metering for usage \(p. 236\)](#).

We recommend that you use AWS CloudTrail to monitor activity to ensure that billing information is being sent to AWS. Keep the following in mind when sending metering records:

- Metering requests are de-duplicated on the hour.
- Records sent every hour are cumulative.
- We strongly recommend as a best practice that, even if there were no records in the last hour, you send metering records every hour, with usage of 0.

Scenario: Monitor changes to user entitlements

Set up an Amazon Simple Queue Service (Amazon SQS) queue, and subscribe to your product's Amazon SNS topics—there are two SNS topics, one for entitlement changes and one for subscription changes. Your topic information was included in the email message that you received from the AWS Marketplace Operations Team when you created your product. For more information, see [Creating a SaaS product \(p. 215\)](#). By subscribing to your SNS topics, you receive notifications about changes to customer subscriptions, including providing or revoking access for specific customers.

Note

An SNS topic Amazon Resource Name (ARN) for a subscription change looks like `arn:aws:sns:us-east-1:<account id>:aws-mp-subscription-notification-<product code>`. An SNS topic ARN for entitlement changes looks like `arn:aws:sns:us-east-1:<account id>:aws-mp-entitlement-notification-<product code>`.

The notifications that you must respond to are as follows:

- `entitlement-updated` (in the entitlement SNS topic)– The customer entitlement has changed, and you must call the `GetEntitlements` API operation to see the new status. Update your customer store, and, if applicable (for example, the customer's contract has lapsed), follow your practices for shutting down customer resources, adhering to your retention policies.
- `subscribe-success` (in the subscription SNS topic) – The customer is subscribed, and you can successfully meter against their customer ID.
- `unsubscribe-pending` (in the subscription SNS topic) – The customer is in the process of unsubscribing. You should send any last metering records.
- `unsubscribe-success` (in the subscription SNS topic) – The customer has unsubscribed. Metering records for the customer will no longer be accepted. Follow your practices for shutting down customer resources, adhering to your retention policies.
- `subscribe-fail` (in the subscription SNS topic) – The customer subscription failed. You should not meter against their customer ID or enable resources on behalf of the customer.

Note

For additional information, see [Checking entitlements \(p. 239\)](#).

Testing your SaaS contract product integration

After you've integrated your contract with pay-as-you-go product with AWS Marketplace, you must conduct in-depth testing to ensure that the integration is successful. The following procedure outlines the steps to verify your product integration.

Note

Use your own accounts to subscribe to your product and test that the integration is successful. Prices can be temporarily reduced so that you can test the purchase flow without incurring high charges in those accounts. For more information about temporarily reducing the prices or allowing additional test accounts to access your product, [contact us](#). After your product is launched, the service must continue to respond to these scenarios for new customers.

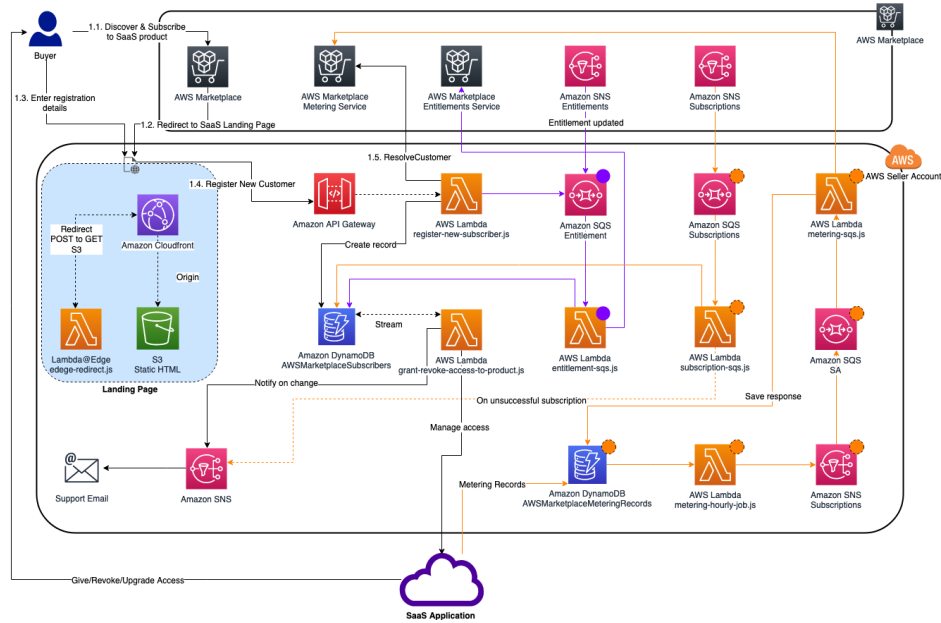
1. Use an allowed account to test the customer experience by getting a contract for your product.
2. After the account has the contract, ensure that the account is redirected to the registration URL, and that the redirect is a POST request that includes a temporary token. Make sure that your application persists the customer ID for future calls and correctly handles the entitlement the customer has. This tests part of [Scenario: Your service validates new customers \(p. 221\)](#).
3. After verifying the test account in the previous step, onboard the account into your application. For example, you can have the test customer fill out a form to create a new user account. Or, provide them with other next steps to get access to your SaaS application. This tests part of [Scenario: Your service validates new customers \(p. 221\)](#).
4. If no entitlement is returned from the `GetEntitlements` API operation, either during onboarding or in your ongoing verification passes, your application must correctly manage access and the experience for users who are not entitled. This tests [Scenario: Your service handles customer requests \(p. 221\)](#).
5. After the test customer is onboarded, make requests that will send metering records to AWS for billing purposes by using the `BatchMeterUsage` API operation in the AWS Marketplace Metering Service. This tests [Scenario: Meter usage \(p. 222\)](#).
6. Test for subscription changes. Verify that your application correctly handles unsubscribes, successful subscription, and failed subscription scenarios. This tests [Scenario: Monitor changes to user entitlements \(p. 222\)](#).
7. After you have completed all the integration requirements and tested the solution, notify the AWS Marketplace Operations team. They will then test the solution by verifying that you have successfully called the `GetEntitlements` API operation and sufficiently onboarded new customers. They will also verify that you have successfully sent metered records with the `BatchMeterUsage` API operation.

After your integration and testing is complete, you can perform a final review and list your product on the public AWS Marketplace. For more information, see [Creating a SaaS product \(p. 215\)](#).

Deploy a serverless SaaS integration solution

The AWS Marketplace serverless SaaS integration deployment fulfills the core capabilities required to successfully integrate a vendor's SaaS solution with its corresponding listing on AWS Marketplace. These capabilities include accepting new customer registrations, granting and revoking customer access, updating customer entitlements, and reporting metered usage.

The following figure shows how the AWS Marketplace serverless SaaS integration on AWS environment sets up the following workflow of events.



For more information about how to deploy a serverless SaaS integration on the AWS Cloud, reference the [AWS Marketplace Serverless SaaS Integration Quick Start Reference Deployment Guide](#). This *Quick Start Reference Guide* is for registered AWS Marketplace sellers who want a lightweight serverless solution for completing the required integration on new SaaS listings.

Plan your SaaS product

Before you add your SaaS product to AWS Marketplace, you must first do some planning. This step is critical to the success of your product. A lack of planning can result in billing issues or you might have to re-create your product in AWS Marketplace.

Important

Most of your product's settings can't be changed after you've configured them. If you need to change them after the product is created in AWS Marketplace, you probably need to create a new product with the correct settings.

Plan your pricing

There are three pricing options for SaaS products on AWS Marketplace. Choosing the right pricing model for your product is the most important decision you will make. Choosing the wrong pricing model can set you back by weeks, because it determines the payment options for your customers and the billing integration code you'll need to write, test, and deploy.

- **SaaS subscriptions** – A pay-as-you-go model where buyers are billed for their hourly usage of your SaaS product.
- **SaaS contracts** – Buyers are either billed in advance for the use of your software, or you can offer them a flexible payment schedule.
- **SaaS contracts with pay-as-you-go** – This option is similar to a standard contract, however your customers can also pay for additional usage above their contract. This is a blended pricing option, that gives your customers the most pricing options and it requires the most integration code on your end.

For more information on pricing, see [SaaS product pricing \(p. 227\)](#).

Plan your billing integration

One of the benefits of having a SaaS product on AWS Marketplace is consolidating billing. In order to take advantage of this benefit, you must integrate with the AWS Marketplace Metering Service or the AWS Marketplace Entitlement Service, depending on your chosen pricing model. These two services help you ensure that your billing and usage reporting is accurate.

After you plan your integration, you must test the integration with your product before it goes live. For more information about integration and testing, see [Accessing the AWS Marketplace Metering and Entitlement Service APIs](#) (p. 236).

Plan your Amazon SNS integration

There are two Amazon SNS topics that you can subscribe to for your SaaS product. These messages can help you programmatically handle changes to subscriptions and contracts initiated by AWS or by your customers. You can use these Amazon SNS notifications as programmatic triggers to enable a customer to register for a new account in on your product registration website, to deny customers with expired subscriptions from accessing your product, depending on how you program the handling of these notifications.

Plan how customers will access your product

This section describes how to make your product accessible to buyers.

Plan your SaaS product registration Website

Customers who buy your SaaS product need to access to it. You must plan and implement how you want your customers to access the product. SaaS products support the following access options:

- AWS PrivateLink
- Your own product website

Using AWS PrivateLink for customers to access your SaaS product

You can use [Using AWS PrivateLink with AWS Marketplace](#) (p. 247) to configure your service as an Amazon Virtual Private Cloud (Amazon VPC) endpoint service. Your customers can create a VPC endpoint and access your software across the AWS Cloud virtual network. Alternatively, you can provide access to your software product through a website you own and maintain, with customers creating a connection across the internet.

Using your own registration website

Your SaaS product is hosted in your environment and it must be accessed over the internet through a public endpoint that you manage and maintain, like a website. Typically, you have a website that customers use to register for your product, sign in to use the product, and access support for your product. For the sake of simplicity, this endpoint will be referred to as your *registration website*.

If you choose this access option and your product doesn't already have a registration website, you need to create one. After you have a registration website, your website must be programmed to validate customers whenever they access your registration page.

To validate customers using your registration website

1. Accept POST requests that includes the temporary token `x-amzn-marketplace-token`.

2. Exchange the token for a `customerID` by calling [ResolveCustomer](#) in the AWS Marketplace Metering Service.
3. After obtaining a `customerID`, persist it in your application for future calls.
4. With the `customerID`, call [GetEntitlement](#) in the AWS Marketplace Entitlement Service to verify which dimension the customer is subscribed to and the quantity.
5. After you've verified your customer's access and entitlement, program your application to ensure that the customer doesn't exceed what they're entitled to.

SaaS product guidelines

AWS Marketplace maintains these guidelines for all SaaS products and offerings on AWS Marketplace to promote a safe, secure, and trustworthy platform for our customers.

All products and their related metadata are reviewed when submitted to ensure that they meet or exceed current AWS Marketplace guidelines. These guidelines are reviewed and adjusted to meet our evolving security requirements. In addition, AWS Marketplace continuously reviews products to verify that they meet any changes to these guidelines. If products fall out of compliance, we might require that you update your product and in some cases your product might temporarily be unavailable to new subscribers until issues are resolved.

Product setup guidelines

All SaaS products must adhere to the following product setup guidelines:

- At least one pricing dimension must have a price greater than \$0.00.
- All pricing dimensions must relate to actual software and cannot include any other products or services unrelated to the software.
- SaaS products offered exclusively in the AWS GovCloud (US) Regions must include `GovCloud` somewhere in the product title.

Customer information requirements

All SaaS products must adhere to the following customer information requirements:

- SaaS products must be billed entirely through the listed dimensions on AWS Marketplace.
- You cannot collect customer payment information for your SaaS product at any time, including credit card and bank account information.

Product usage guidelines

All SaaS products must adhere to the following product usage guidelines:

- After subscribing to the product in AWS Marketplace, customers should be able to create an account within your SaaS application and gain access to a web console within two business days. If the customer cannot gain access to the application immediately, you must provide a message with specific instructions on when they will gain access. When an account has been created, the customer must be sent a notification confirming that their account has been created along with clear next steps.
- If a customer already has an account in the SaaS application, they must have the ability to log in from the fulfillment landing page.

- Customers must be able to see the status of their subscription within the SaaS application, including any relevant contract or subscription usage information.
- Customers must be able to easily get help with issues such as: using the application, troubleshooting, and requesting refunds (if applicable). Support contact options must be specified on the fulfillment landing page.
- Product software and metadata must not contain language that redirects users to other cloud platforms, additional products, or up sell services that aren't available on AWS Marketplace.
- If your product is an add-on to another ISV's product, your product description must indicate that it extends the functionality of the other product and that without it, your product has very limited utility. For example, *This product extends the functionality of <product name> and without it, this product has very limited utility. Please note that <product name> might require its own license for full functionality with this listing.*

Architecture guidelines

All SaaS products must adhere to the following architecture guidelines:

- A portion of your application must be hosted in an AWS account that you own.
- All application components should be hosted in infrastructure you manage. Applications that require additional resources in the customer's infrastructure must follow these guidelines:
 - Provision resources in a secure way, such as using the AWS Security Token Service (AWS STS) or AWS Identity and Access Management (IAM).
 - Provide additional documentation including a description of all provisioned AWS services, IAM policy statements, and how an IAM role or user is deployed and used in the customer's account.
 - Include a notification in the product description that explains that if the customer incurs additional AWS infrastructure charges separate from their AWS Marketplace transaction, they're responsible for paying the additional infrastructure charges.
 - If your product deploys an agent, you must provide instructions to the customer that describe how to deploy it in their AWS account.
 - Applications that require resources running in the customer's infrastructure will have an additional review by AWS Marketplace, which can take 2-4 weeks.
- Successfully call the AWS Marketplace APIs from the AWS account that registered as a provider and submitted the SaaS publishing request. The SaaS pricing model determines which APIs should be called:
 - SaaS contracts – [GetEntitlements](#) in the AWS Marketplace Entitlement Service.
 - SaaS contracts with consumption – [GetEntitlements](#) in the AWS Marketplace Entitlement Service and [BatchMeterUsage](#) in the AWS Marketplace Metering Service.
 - SaaS subscriptions – [BatchMeterUsage](#) in the AWS Marketplace Metering Service.
- SaaS products offered exclusively in the AWS GovCloud (US) Regions must outline the architectural boundaries between other AWS Regions and the AWS GovCloud (US) Regions, use cases for the product, and the workloads not recommended for the product.

SaaS product pricing

After a buyer purchases your software as a service (SaaS) product on AWS Marketplace, AWS Marketplace provides you with their billing identifier. You use the billing identifier to call the AWS Marketplace Entitlement Service and the AWS Marketplace Metering Service. Then, customers access the product in your AWS environment or through a VPC endpoint connection you create.

SaaS pricing models

| Pricing model | Description |
|-----------------------------------|--|
| SaaS subscriptions | A pay-as-you-go model where we bill buyers for their hourly usage of your SaaS product. For more information, see Pricing for SaaS subscriptions (p. 228) |
| SaaS contracts | Buyers are either billed in advance for the use of your software, or you can offer them a flexible payment schedule. Customers can also pay for additional usage above their contract. For more information, see Pricing for SaaS contracts (p. 229) . |
| SaaS contracts with pay-as-you-go | Buyers are either billed in advance for the use of your software, or you can offer them a flexible payment schedule. Buyers are also billed an additional metered rate for usage on top of the contract price. For more information, see Pricing for SaaS contracts (p. 229) . |

To make your SaaS product available on AWS Marketplace, decide whether you want to offer the [SaaS subscriptions pricing model \(p. 228\)](#) or the [SaaS contracts pricing model \(p. 229\)](#).

Pricing for SaaS subscriptions

For software as a service (SaaS) subscriptions, AWS Marketplace bills your customers based on the metering records that you send to us. All charges must be measured and reported every hour from the software deployed in the customer's account. All usage is then calculated monthly and billed monthly using the same mechanism as AMI based AWS Marketplace offerings. Our ability to bill customers for usage of your product is dependent upon receiving metering records from you. You are responsible for ensuring that your product's metering records are successfully transmitted and received.

Before you can publish a SaaS product with subscription pricing, you must do the following:

1. Create a new SaaS product in the AWS Marketplace Management Portal, choose **New SaaS Subscription**.
2. Complete the fields in the **General** tab with the necessary information. Make a note of the product code.
3. On the **Pricing** tab, under **Set Pricing**, select the **Category** that describes your product's pricing most accurately. The pricing category appears to customers on the AWS Marketplace website. You can choose from **Bandwidth** (GBps, MBps), **Data** (GB, MB, TB), **Hosts** (hours), **Requests**, **Tiers** (hours), or **Users** (hours). If none of the predefined categories fit your needs, you can choose the more generic **Units** category.

Next, define your Pricing Dimensions. Each Pricing Dimension represents a feature or service that you can set a per-unit price for. Examples of dimensions include users, hosts scanned, and GB of logs ingested. You can define up to 24 dimensions. For each dimension you define, you must add the following information:

- **Dimension API Name** – The API name used when sending metering records to the [AWS Marketplace Metering Service](#). This name indicates which dimension your customer used. This name is visible in billing reports. The name doesn't need to be reader-friendly because you're the only one with access to your reports. After you set the name, you can't change it.

- **Dimension Description** – The customer-facing statement that describes the dimension for the product. The description can be no more than 70 characters and should be user-friendly. Examples of descriptions are Administrators per hour, and Per Mbps bandwidth provisioned. After the product is published, you can't change this description.
- **Dimension Rate** – The software charge per FCP unit for this product, in USD. This field supports three decimal places.

When a SaaS subscription ends

A customer can unsubscribe from your SaaS subscription product through the AWS Management Console. Key points of the SaaS subscription ending process include the following:

1. Your SaaS product is sent an `unsubscribe-pending` notification through the Amazon SNS topic for that customer.
2. You have one hour to meter any remaining usage for the customer.
3. After this hour, you receive an `unsubscribe-success` notification. At this point, you can no longer send metering records for this customer.

It's up to you to decide how you want to disable functionality in your SaaS product for unsubscribed customers. For example, your product might complete the customer's existing work but prevent them from creating work. You might want to display a message to the customer that their usage has been disabled. Customers can resubscribe to your product through AWS Marketplace.

When a SaaS subscription is cancelled

Key points of the SaaS subscription cancellation process include the following:

1. A customer can cancel their subscription to your SaaS subscription product the **Your Marketplace Software** page of the AWS Marketplace website.

Your SaaS product is sent notification through the Amazon SNS topic for that customer.

2. You have one hour to meter any remaining usage for the customer.
3. You notify the customer from your product that the cancellation is in progress. If a customer indicates that they want to cancel through your product, direct the customer to AWS Marketplace. To guarantee that there will be no future charges, customers should confirm the cancellation with AWS Marketplace.

Pricing for SaaS contracts

For software as a service (SaaS) contracts, the customer initiates a purchase of your software and enters into an agreement with you. Under the agreement, the customer is entitled to a specified quantity of use of your SaaS product. AWS Marketplace communicates these entitlements to your SaaS application. This is done through the AWS Marketplace Entitlement Service. When using the SaaS Contract pricing model, your application never sends metering records. Instead, it verifies entitlement by calling the AWS Marketplace Entitlement Service. You define the usage categories, dimensions, and the length of the contract.

AWS Marketplace bills your customers upfront or by the payment schedule that you define, based on the contract between you and your customer. After that point, they're entitled to use those resources. For additional usage above their contract, AWS Marketplace bills your customers based on the metering records received by us through the AWS Marketplace Entitlement Service.

Before you can publish a SaaS product with contract pricing, you must do the following:

1. Create a new SaaS product in the AWS Marketplace Management Portal, and choose **New SaaS Contract**.
2. Complete the fields in the **General** tab with the necessary information. Make a note of the product code.
3. On the **Pricing** tab:
 - a. For **Set Pricing**, choose the **Contract Duration** you want offer customers. You can enter different prices for each contract duration. You can select one or more of the following options: **Monthly**, **1 year**, **2 Years**, and **3 Years**. If you are creating a private offer, you can choose a custom duration in months (up to 60 months).
 - b. For **Choose the contract type you want to offer**, choose how you want customers to be able to purchase your product from the following options:
 - **Buyer can choose one or more options offered** – Customers can select a quantity for each pricing dimension you offer.
 - **Buyer can choose one tier from multiple tiers offered** – Customers choose a tier from options that include different sets of features, services, and usage amounts.
 - c. Choose the usage unit category that describes your product's pricing most accurately. The pricing category appears to customers on the AWS Marketplace website. You can choose from **Bandwidth** (GBps, MBps), **Data** (GB, MB, TB), **Hosts** (hours), **Requests**, **Tiers** (hours), or **Users** (hours). If none of the predefined categories fit your needs, you can choose the more generic **Units** category.
4. After you choose a category, define your Pricing Dimensions. Each Pricing Dimension represents a feature or service that you can set a per unit price for. Examples of dimensions are users, hosts scanned, and GB of logs ingested. For each dimension you define, you add a name, a description, a price, and an API name. The name, price, and description are displayed to customers. You use the API name for tracking and reporting with AWS Marketplace as follows:
 - Calling the [AWS Marketplace Entitlement Service](#) to retrieve the dimensions your customers have purchased.
 - Calling the [AWS Marketplace Metering Service](#) to indicate which dimensions customers used.

For each pricing dimension in your contract, you can choose to let customers pay as they go for additional usage of that dimension above their contract. You can also add additional dimensions without contract prices that customers only consume by paying as they go.

When using the wizard to create the contracts for your SaaS product, you must define the following fields for your pricing dimensions:

- **Dimension API Name** – The name used when calling the Entitlements API. This name is visible in billing reports and reports that aren't external-facing. The maximum length for the API name is 15 characters. After you set the name, it can't be changed.
- **Dimension Display Name:** – The customer-facing name of a dimension. This name should help the customer understand the dimension for the product. The name should be user-friendly, and its maximum length is 24 characters. This value can be changed.
- **Dimension Description:** – The customer-facing description of a dimension that provides additional information about the dimension for the product. The maximum length for the description is 70 characters.
- **Dimension - Monthly Price** – The software charge per unit for the 1-month option for this dimension. This field supports three decimal places.
- **Dimension - 1 Year Price** – The software charge per unit for the 12-month option for this dimension. This field supports three decimal places. It's not a monthly charge. The price must reflect the 12-month one-time charge price.
- **Dimension - 2 Years Price** – The software charge per unit for the 24-month option for this dimension. This field supports three decimal places.
- **Dimension - 3 Years Price** – The software charge per unit for the 36-month option for this dimension. This field supports three decimal places.

Example: Data storage application

| | Monthly price | 12-month price | 24-month price | Pay-as-you-go price for additional usage |
|-----------------------|---------------|----------------|----------------|--|
| Unencrypted data (GB) | \$1.50/GB | \$16.00/GB | \$30.00/GB | \$0.1/GB per hour |
| Encrypted data (GB) | \$1.55/GB | \$16.60/GB | \$31.20/GB | \$0.11/GB per hour |

Example: Log monitoring product

| | Monthly price | 12-month price | Pay-as-you-go price for additional usage |
|--|---------------|----------------|--|
| Basic (10 hosts monitored, 5 containers monitored) | \$100 | \$1000 | |
| Standard (20 hosts monitored, 10 containers monitored) | \$200 | \$2000 | |
| Pro (40 hosts monitored, 20 containers monitored) | \$400 | \$4000 | |
| Additional hosts monitored per hour | | | \$0.1 |
| Additional containers monitored per hour | | | \$0.2 |

Note

The prices can be for the following durations: 1 month, 12 months, 24 months, or 36 months. You can choose to offer one or more of these options for your product. The durations must be the same across each dimension. For example, assume that you have `ReadOnlyUsers` and `AdminUsers` dimensions. If you offer a yearly price for `ReadOnlyUsers`, you must offer a yearly price for `AdminUsers`, too.

SaaS contract upgrades

Customers can upgrade a contract to one of a higher value except for longer durations. For example, they can upgrade to higher quantities or higher-value entitlements. Customers are given a prorated credit for their existing contract. Customers can't decrease the size of their existing contract. They can only decrease the size at renewal, or cancel their renewal.

Entitlements are verified by your SaaS product, which makes calls to the AWS Marketplace Entitlement Service.

Automatic renewals

When a customer purchases your product through AWS Marketplace using SaaS contracts, they can agree to automatic renewal of the contract terms. The customer continues to pay for the entitlements every

month or for 1, 2, or 3 years. The customer always has the option to modify the renewal settings. They can cancel the renewal or renew the contract for different quantities and durations.

When a SaaS contract ends

A SaaS contract product has a contract expiry. When a contract ends, the following events occur:

1. Your SaaS product receives an `entitlement-updated` notification indicating the buyer's entitlement has changed. The AWS Marketplace Entitlement Service returns an empty response.
2. You have 1 hour to meter any remaining usage for the customer. After this time has elapsed, you can no longer send metering records for this customer.

When a SaaS contract is cancelled

Key points of the SaaS contract cancellation process include the following:

1. Customers can request a cancellation and refund for SaaS contract products through AWS Support.

Customers must request refunds within 48 hours through AWS Support.

The full or prorated refund is typically granted in 3–5 business days.

2. Your SaaS product is sent notification through the Amazon SNS topic for that customer.
3. You have one hour to send a final metering record for the customer for any additional usage charges.
4. You notify the customer from your product that the cancellation is in progress. If a customer indicates that they want to cancel through your product, direct the customer to AWS Marketplace. To guarantee that there will be no future charges, customers should confirm the cancellation with AWS Marketplace.

SaaS customer onboarding

With software as a service (SaaS) subscriptions and SaaS contracts, your customers subscribe to your products through AWS Marketplace but access the product in your AWS environment. After subscribing to the product, your customer is directed to a website you create and manage as a part of your SaaS product to register their account and configure the product.

When creating your SaaS product listing, you provide a URL to your registration landing page. We use that URL to redirect customers to your registration landing page after they subscribe. On your software's registration landing page, you collect whatever information is required to create an account for the customer. We recommend collecting your customer's email addresses if you plan to contact them through email for usage notifications.

The registration landing page must be able to identify and accept the `x-amzn-marketplace-token` token in the form data from AWS Marketplace with the customer's identifier for billing. It should then pass that token value to the AWS Marketplace Metering Service to resolve for the unique customer identifier, customer AWS account Id, and corresponding product code. For a code example, see [ResolveCustomer code example \(p. 243\)](#).

Note

The registration token resolves to a specific subscribed customer and each generated token has an expiration window of 4 hours. As long as the caller is calling the API with the same token, it will keep returning the same response values until the token expires.

Configuring your SaaS product to accept new buyers

You're responsible for correctly configuring your SaaS software to accept new customers and meter them appropriately. The following process outlines one recommended way of identifying, implementing, and metering a new customer's access to your software:

1. When a customer visits your product page on the AWS Marketplace website, they choose to subscribe to your product.
2. The customer's AWS account is subscribed to your product. This means subscription and metering records sent from your product become part of the customer's AWS bill.
3. A registration token is generated for the customer that contains their customer identifier and your product code.
4. The customer is redirected to your software's registration landing page. This page must be able to accept the token with the customer's identifier.
5. The customer's browser sends a POST request to your software's registration landing page URL. The request contains one POST parameter, `x-amzn-marketplace-token`, containing the customer's registration token. From the perspective of your registration website, the customer has submitted a form with this parameter. The registration token is an opaque string.
6. To redeem this registration token for a customer identifier, customer AWS account Id, and product code, your website must call `ResolveCustomer` on the AWS Marketplace Metering Service. The customer identifier isn't the customer's AWS account ID, but it's universal between products and should be saved to an internal source as part of your customer records. The product code is a unique string for your SaaS product that AWS provides to you. Each AWS product has one unique product code, which is assigned to you during registration.

Note

To see an example of a `ResolveCustomer` call, see [ResolveCustomer code example \(p. 243\)](#).

7. Your website validates that the product code returned matches your SaaS product the customer is attempting to access and calls `GetEntitlement` to return the entitlement data for the customer's subscription.
8. The customer is instructed to either create an account in your product or sign in to an existing account.
9. The customer is now signed in to your website using credentials specific to that SaaS product. In your accounts database, you can have an entry for each customer. Your accounts database must have a column for the AWS customer identifier, which you populate with the customer identifier that you obtained in step 6. Verify that no other accounts in your system share this customer identifier. For customers who subscribe to multiple products through AWS Marketplace, the customer identifier will remain the same, with each subscription having a unique product code.
10. During your seller registration process, you subscribe to Amazon SNS topics that notify you when customers subscribe or unsubscribe to your product. These are Amazon SNS notifications in JSON format that inform you of customer actions:
 - Entitlement notification - For products with pricing models that include a contract, you are notified when buyers create a new contract, upgrade it, renew it, or it expires. Your accounts database must have an extra column for the subscription state. For more information, see [Amazon SNS topic: aws-mp-entitlement-notification \(p. 234\)](#).
 - Subscription notification - For products with any pricing model, including contracts and subscriptions, you are notified when a buyer subscribes or unsubscribes to a product. For more information, see [Amazon SNS topic: aws-mp-subscription-notification \(p. 235\)](#).

We recommend that you use Amazon Simple Queue Service (Amazon SQS) to capture these messages. After you receive a subscription notification with `subscribe-success`, the customer account is ready for metering. Records that you send before this notification aren't metered. For

information about how to do this, see [Step 2: Give Permission to the Amazon SNS Topic to Send Messages to the Amazon SQS Queue](#) in the *Amazon Simple Notification Service Developer Guide*.

Note

Do not activate a product subscription unless you receive a `subscribe-success` notification.

11. Use the customer identifier stored in your database to meter for usage through the AWS Marketplace Metering Service or check for entitlements through the AWS Marketplace Entitlement Service.

Security and ordering

As a seller, it's your responsibility to trust only customer identifiers that are immediately returned from AWS or those that your system has signed. We recommend that you resolve the registration token immediately because it may expire after approximately 1 hour. After you resolve the registration token, store the customer identifier as a signed attribute on the customer's browser session until the registration is complete.

Amazon SNS notifications for SaaS products

To receive notifications, you subscribe to the AWS Marketplace Amazon Simple Notification Service (Amazon SNS) topics provided to you during product creation. The topics provide notifications about changes to customers' subscriptions and contract entitlements for your products. This enables you to know when to provide and revoke access for specific customers.

The following Amazon SNS topics are available to software as a service (SaaS) products:

- [Amazon SNS topic: `aws-mp-entitlement-notification` \(p. 234\)](#) – This topic notifies you when buyers create a new contract, upgrade it, renew it, or it expires. This is only available for products with pricing models that include a contract (also known as **SaaS Contracts** and **SaaS Contracts with Consumption (Overages)**).
- [Amazon SNS topic: `aws-mp-subscription-notification` \(p. 235\)](#) – This topic notifies you when a buyer subscribes to or unsubscribes from a product and includes the `offer-identifier` for private offers. This is available for all pricing models, including contracts and subscriptions (also known as **SaaS Subscriptions** and **SaaS Contracts with Consumption (Overages)**).

To learn more about the scenarios in which you respond to these notifications, see the following topics:

- [Integrate your SaaS subscription product \(p. 216\)](#)
- [Integrate your SaaS contract product \(p. 219\)](#)
- [Integrate your SaaS contract with pay-as-you-go product \(p. 221\)](#)

Amazon SNS topic: `aws-mp-entitlement-notification`

Each message in the `aws-mp-entitlement-notification` topic has the following format.

```
{
  "action": "<action-name>",
  "customer-identifier": " X01EXAMPLEX",
  "product-code": "n0123EXAMPLEXXXXXXXXXXXXX",
}
```

The `<action-name>` will always be `entitlement-updated`.

Note

- For entitlement messages, regardless of the action (new, upgrade, renewal, or expired), the message is the same. A subsequent call to `GetEntitlement` is required to discover the content of the update.
- For **SaaS Contract with Consumption (Overages)**, sellers are provided with the [aws-mp-subscription-notification SNS topic \(p. 235\)](#). This is an extra notification that a seller receives when they add on overage pricing. When a seller acquires new customers, instead of only getting `entitlement-updated` (which may refer to any kind of action), the seller receives a `subscribe` message indicating that this is a new customer.

Products with contract pricing (including contracts with pay-as-you-go) must respond to these messages. For more information about how to respond, see [Scenario: Monitor changes to user subscriptions \(p. 220\)](#).

Amazon SNS topic: aws-mp-subscription-notification

Each message in the `aws-mp-subscription-notification` topic has the following format.

```
{
  "action": "<action-name>",
  "customer-identifier": " X01EXAMPLE",
  "product-code": "n0123EXAMPLEXXXXXXXXXXXX",
  "offer-identifier": "offer-abcexample123"
}
```

The `offer-identifier` only appears in the notification if the offer is a *private offer*.

The `<action-name>` will vary depending on the notification. Possible actions are:

- `subscribe-success` – The `subscribe-success` message signals when the seller can begin sending metering records.
- `subscribe-fail` – If the `subscribe-fail` message is generated, payment might have failed even though the buyer has already transitioned from the AWS Marketplace to the seller's SaaS landing page. The seller should wait for the `subscribe-success` message before allowing consumption of the product.
- `unsubscribe-pending` – When a buyer unsubscribes, an `unsubscribe-pending` message is sent first. This indicates that the seller has a limited time (about one hour) to get final metering records sent before the buyer is cancelled completely.
- `unsubscribe-success` – The `unsubscribe-success` message signals the completion of cancellation, after which no further metering records will be accepted.

Note

- If a buyer unsubscribes and then immediately successfully re-subscribes before the final `unsubscribe-success` message is sent, the final `unsubscribe-success` message will not be sent and a `subscribe-success` message will be sent instead.

Products with subscription pricing (including contracts with pay-as-you-go) must respond to these messages. For more information about how to respond, see the following topics:

- [Integrate your SaaS subscription product \(p. 216\)](#)
- [Integrate your SaaS contract with pay-as-you-go product \(p. 221\)](#)

Subscribing an SQS queue to the SNS topic

We recommend subscribing an Amazon SQS queue to the provided SNS topics. For detailed instructions on creating an SQS queue and subscribing the queue to a topic, see [Subscribing an Amazon SQS queue to an Amazon SNS topic](#) in the *Amazon Simple Notification Service Developer Guide*.

Note

You can only subscribe to AWS Marketplace SNS topics from the AWS account used to sell the products. However, you can forward the messages to a different account. For more information, see [Sending Amazon SNS messages to an Amazon SQS queue in a different account](#) in the *Amazon Simple Notification Service Developer Guide*.

Polling the SQS queue for notifications

After you subscribe your SQS queue to an SNS topic, the messages are stored in SQS. You must define a service that continually polls the queue, looks for messages, and handles them accordingly.

Accessing the AWS Marketplace Metering and Entitlement Service APIs

This section outlines the process of integrating with the AWS Marketplace Metering Service or AWS Marketplace Entitlement Service, used to ensure your billing and reporting of customer usage of your software as a service (SaaS) products is accurate. It's assumed that you've submitted a SaaS subscriptions product or a SaaS contracts product that has been published to a limited state. In a limited state, you can use your test accounts to verify proper configuration and function but your product is not available publicly.

Note

If your SaaS product is integrated with another AWS managed service that handles metering in a different way (such as Amazon SageMaker Ground Truth, or AWS WAF), then you do not need to integrate with AWS Marketplace metering service. Metering for your product should only happen in one system to avoid double billing your customer.

Topics

- [Metering for usage \(p. 236\)](#)
- [Checking entitlements \(p. 239\)](#)
- [SaaS product integration checklist \(p. 240\)](#)

For information about setting up the AWS CLI, along with credentials, see [Configuring the AWS CLI](#) in the *AWS Command Line Interface User Guide*. If you're new to the AWS Python SDK, see the [Boto 3 Quickstart](#).

Metering for usage

For software as a service (SaaS) subscriptions, you meter for all usage, and then customers are billed by AWS based on the metering records that you provide. For SaaS contracts, you only meter for usage beyond a customer's contract entitlements. When your application meters usage for a customer, your application is providing AWS with a quantity of usage accrued. Your application meters for the pricing dimensions that you defined when you created your product, such as gigabytes transferred or

hosts scanned in a given hour. For example, if you charge based on the amount of data sent into your application, you can measure the amount of data and send a corresponding metering record once an hour. AWS calculates a customer's bill using the metering data along with the prices that you provided when you created your product.

Note

Optionally, you can split the usage across properties that you track. These properties are exposed to the buyer as tags. These tags allow the buyer to view their costs split into usage by the tag values. For example, if you charge by the user, and users have a `Department` property, you could create a usage allocations with tags that have a key of `Department`, and one allocation per value. This doesn't change the price, dimensions, or the total usage that you report, but allows your customer to view their costs by categories appropriate to your product. For more information, see [Vendor-metered tagging \(Optional\) \(p. 238\)](#).

We recommend that you send a metering record every hour to give customers as much granular visibility into their usage and costs as possible. If you aggregate usage in time periods greater than an hour (for example, one day), continue sending metering records every hour and record a quantity of 0 if there is no usage to report for that hour. Report usage to AWS on an hourly basis for all of your customers, in batches of up to 25 at a time.

AWS can only bill customers for usage of your product upon receiving metering records from you. You're responsible for ensuring that your product's metering records are successfully transmitted and received. You can use AWS CloudTrail to verify the record or records that you send are accurate. You can also use the information to perform audits over time. For more information, see [Logging AWS Marketplace API calls with AWS CloudTrail \(p. 359\)](#).

Note

If your SaaS product is integrated with another AWS managed service that handles metering in a different way (such as Amazon SageMaker Ground Truth, or AWS WAF), then you do not need to integrate with AWS Marketplace metering service. Metering for your product should only happen in one system to avoid double billing your customer.

Configure your product to meter usage

You use the `BatchMeterUsage` operation in the AWS Marketplace Metering Service to deliver metering records to AWS. Keep the following in mind:

- We require sellers to use batching by using the `BatchMeterUsage` operation.
- We deduplicate metering requests on the hour.
 - Requests are deduplicated per product/customer/hour/dimension.
 - You can always retry any request, but if you meter for a different quantity, the original quantity is billed.
 - If you send multiple requests for the same customer/dimension/hour, the records are not aggregated.
- Sellers can send metering records with a timestamp up to 6 hours in the past if the customer is subscribed to your product. If the customer unsubscribes, sellers have to send the metering records within 1 hour of the customer unsubscribing.
- `BatchMeterUsage` payloads must not exceed 1MB. Choose the number of usage records to send in a `BatchMeterUsage` request so that you don't exceed the size of the payload.
- The AWS Marketplace Metering Service is available in the AWS Regions listed in the topic [AWS Marketplace endpoints and quotas](#) in the *AWS General Reference*. By default, the US East (N. Virginia) Region is enabled for SaaS metering products when you request your product. If you intend to use other Regions, contact the [AWS Marketplace Seller Operations](#) team. For more information, see [BatchMeterUsage](#).

For code examples, see [Code examples for SaaS product integration \(p. 243\)](#).

Example: Host scanning

Your product analyzes computing hardware for known security vulnerabilities. Customers manually initiate or schedule these scans of their Amazon Elastic Compute Cloud (Amazon EC2) instances. As your product performs these scans, it tallies the number of unique hosts scanned every hour. In this example, your product uses the **Hosts** category. You can declare multiple dimensions for the types of hosts scanned. For example, you can charge different prices for small, medium, and large hosts.

Example: Log analysis

Your SaaS product digests logs that are generated by customer products, reporting trends, and anomalies. As customers upload logs to your product, you measure the quantity of data received in megabytes, gigabytes, or terabytes. On the tenth minute of every hour, a cron job reads this usage for each customer for the previous hour. The job builds a batch report and uses the `BatchMeterUsage` operation to send it to AWS. In this example, your product uses the **Data** category. Your product can also meter for the amount of log data stored for any given hour. In this case, your product can meter along two dimensions: data received in the hour and total data stored in the hour. You can continue to meter for data stored until the customer deletes this data or it expires.

Vendor-metered tagging (Optional)

Vendor-metered tagging helps Independent Software Vendors (ISVs) give the buyer more granular insight into their software usage and can help them perform cost allocation.

There are many ways to tag a buyer's software usage. One way is to first ask your buyers what they want to see in their cost allocation. Then you can split the usage across properties that you track for the buyer's account. Examples of properties include `Account ID`, `Business Unit`, `Cost Centers`, and other relevant metadata for your product. These properties are exposed to the buyer as tags. Using tags, buyers can view their costs split into usage by the tag values in their AWS Billing Console (<https://console.aws.amazon.com/billing/>). Vendor-metered tagging doesn't change the price, dimensions, or the total usage that you report. It allows your customer to view their costs by categories appropriate to your product.

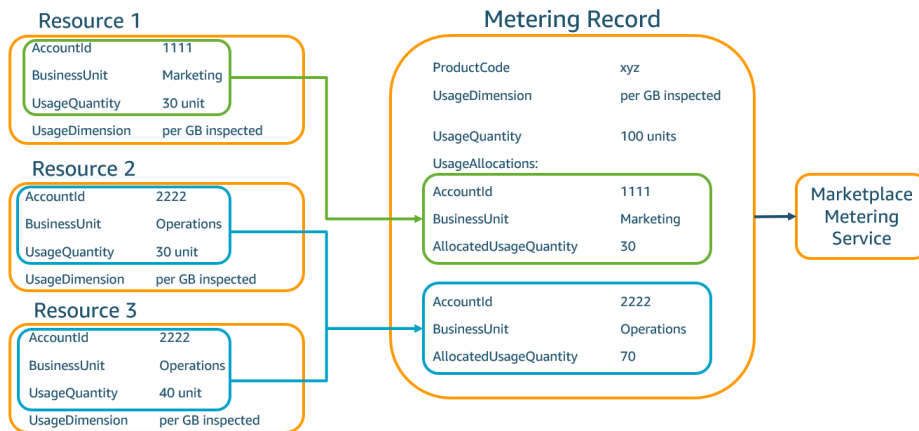
In a common use case, a buyer subscribes to your product with one AWS account. The buyer also has numerous user accounts associated with the same product subscription. You can create usage allocations with tags that have a key of `Account ID`, and then allocate usage to each user account. In this case, buyers can activate the `Account ID` tag in their Billing and Cost Management console and analyze individual user account usage.

Seller experience

Sellers can aggregate the metering records for resources with the same set of tags instead of aggregating usage for all resources. For example, sellers can construct the metering record that includes different buckets of `UsageAllocations`. Each bucket represents `UsageQuantity` for a set of tags, such as `AccountId` and `BusinessUnit`.

In the following diagram, **Resource 1** has a unique set of `AccountId` and `BusinessUnit` tags, and appears in the **Metering Record** as a single entry.

Resource 2 and **Resource 3** both have the same `AccountId` tag, `2222`, and the same `BusinessUnit` tag, `Operations`. As a result, they're combined into a single `UsageAllocations` entry in the **Metering Record**.



Buyer experience

The following table shows an example of the buyer experience after a buyer activates the `AccountId` and `BusinessUnit` vendor tags.

In this example, the buyer can see allocated usage in their **Cost Usage Report**. The vendor-metered tags use the prefix `aws:marketplace:isv`. Buyers can activate them in the Billing and Cost Management, under **Cost Allocation Tags, AWS-generated cost allocation tags**.

The first and last rows of the **Cost Usage Report** are relevant to what the Seller sends to the Metering Service (as shown in the [Seller experience \(p. 147\)](#) example).

Cost Usage Report (Simplified)

| ProductCode | Buyer | UsageDimension | UsageQuantity | aws:marketplace:isv:BusinessUnit | aws:marketplace:isv:BusinessUnit |
|-------------|--------------|-----------------------------|---------------|----------------------------------|----------------------------------|
| xyz | 111122223333 | Network: per (GB) inspected | 70 | 2222 | Operations |
| xyz | 111122223333 | Network: per (GB) inspected | 30 | 3333 | Finance |
| xyz | 111122223333 | Network: per (GB) inspected | 20 | 4444 | IT |
| xyz | 111122223333 | Network: per (GB) inspected | 20 | 5555 | Marketing |
| xyz | 111122223333 | Network: per (GB) inspected | 30 | 1111 | Marketing |

For a code example, see [BatchMeterUsage with usage allocation tagging code example \(Optional\) \(p. 246\)](#).

Checking entitlements

If your product is a SaaS contracts product, your product calls the AWS Marketplace Entitlement Service to retrieve the customer's entitlement using the `GetEntitlements`. Your product should verify subsequent usage on that account against the AWS Marketplace Entitlement Service. For example, if the customer provisions 10 users on the account, your product should check the AWS Marketplace Entitlement Service for entitlement to that capacity.

To verify a customer's entitlement to your product, use the `GetEntitlements` operation in the AWS Marketplace Entitlement Service. The AWS Marketplace Entitlement Service is available only in the US East (N. Virginia) Region, accessible through `entitlement.marketplace.us-east-1.amazonaws.com`.

`GetEntitlements` accepts a customer identifier and dimension as filters. `ProductCode` is a required parameter. The operation returns a paginated list of entitlements. The result has an `ExpirationDate` field that shows the minimum period of time that the entitlement is valid for. If the customer has set up automatic renewal, the date in the `ExpirationDate` field is the renewal date.

For code samples, see [Code examples for SaaS product integration \(p. 243\)](#).

Retrieving entitlement on user actions

The following examples can help you better understand the process for retrieving entitlement on user actions.

Example: User-based product

You offer a product that allows a number of accounts to exist for a given customer. The customer can visit a dashboard to provision new users (for example, to assign credentials). When the customer provisions a new user, your product calls `GetEntitlements` to verify that the capacity exists. If it does not, you can call the AWS Marketplace Metering Service to bill for additional users.

Example: Data storage product

You offer a product that enables customers to store a certain amount of data in encrypted or unencrypted form. The customer can view a dashboard that displays the amount of data existing and allocated in your product. Your dashboard retrieves the allocation amount through `GetEntitlements`.

SaaS product integration checklist

Before your SaaS product goes live, use this checklist to verify that you have completed the required configuration.

| Category | Requirements |
|----------|---|
| Access | Submitted a seller registration form with the desired AWS account for AWS Marketplace usage. |
| Access | Completed the seller registration, including terms and conditions, bank account, and W8 or W9 tax form. |
| Access | Configured cross-account roles for the registered AWS Marketplace account. |
| Product | Completed the product request form in the AWS Marketplace Management Portal. |
| Product | Provided AWS account IDs for testing in the Notes tab of the Create product wizard in the AMMP. |
| Product | Provided a URL of the EULA in .txt format in the Products tab. |
| Product | Received your product code and Amazon SNS topic information from AWS Marketplace. |

| Category | Requirements |
|------------------|---|
| Product | Subscribed to the Amazon SNS topic and created an Amazon SQS queue to subscribe to the Amazon SNS topic. |
| Billing Solution | Validated you can send metering records to the <code>BatchMeterUsage</code> operation each hour for each customer for SaaS subscriptions products. Can send metering records for additional usage by each customer for SaaS contracts products. |
| Billing Solution | Validated you can verify customer entitlements from the AWS Marketplace Entitlement Service for SaaS contracts products. |
| Billing Solution | Validated that the costs appear as expected on bills generated for test accounts. |
| Billing Solution | Tested for situations such as invalid customer IDs and canceled subscriptions. |
| Product | Submitted the product request back to AWS Marketplace for publishing. |
| Registration | Implemented an HTTPS registration page that can accept HTTP POST requests. |
| Registration | Validated you can accept new customer registrations. |
| Registration | Validated you are <i>not</i> storing the registration token in a cookie. |
| Registration | Validated you are using <code>ResolveCustomer</code> to obtain the <code>ProductCode</code> and <code>CustomerIdentifier</code> from the AWS token. |
| Registration | Validated you can resolve the registration token received from AWS with no delays. |
| Registration | Tested that you aren't blocked from registering with email services addresses such as Gmail. |
| Registration | Tested that you can accept incomplete registrations and multiple registration attempts. |
| Subscription | Test that you can handle <code>unsubscribe-pending</code> and <code>unsubscribe-success</code> messages. |
| Subscription | Validated that you send final metering records within an hour of receiving an <code>unsubscribe-pending</code> message. |
| Security | Validated the AWS root account doesn't have API keys, has a strong password, and is associated with a hardware multi-factor authentication (MFA) device. All administrative access is through identities created with AWS Identity and Access Management (IAM). No shared accounts. |

| Category | Requirements |
|-------------|---|
| Security | Validated that IAM roles are used for all programmatic Amazon Elastic Compute Cloud (Amazon EC2) access. Credentials aren't hard-coded into scripts, headers, or source code. |
| Security | Validated you maintain comprehensive logging and log consolidation. |
| Security | Verified you have well-defined public and private subnet boundaries that isolate application services and access to database and file systems. Distinct data class definitions that demarcate sensitive data and segregate public and private data. |
| Security | Verified you have private data encryption in transit and at rest with scheduled key rotation. |
| Security | Validated you have security incident tools and access in place and routinely scheduled incident response exercises that accommodate timely investigation and recovery. |
| Reliability | Verified the system adapts to changes in demand, scaling up and down as required, and employs load balancing to ensure high performance. The system also provides edge-based caching as required. |
| Reliability | Validated recovery time and point objectives are specified, and disaster recovery is scheduled at regular intervals. Component failure is self-healing via automated triggers and notifications. |

Reporting

AWS Marketplace produces reports for your SaaS products that include data on subscribers, financials, usage, and taxes. For more information, see [the section called "Seller reports" \(p. 273\)](#). The following table summarizes how financials for SaaS products are reported.

| Report | SaaS content |
|------------------------|---|
| Daily business report | Upfront contract charges appear in the Fees section. Metered usage charges appear in the Usage section. |
| Monthly revenue report | Upfront contract charges appear in the Annual subscriptions section. Metered usage charges appear in the Billing and revenue data section. |

| Report | SaaS content |
|----------------------------|--|
| Sales compensation report | Upfront contract charges and monthly additional usage charges appear as separate line items. |
| Customer subscriber report | New SaaS contracts appear in the Annual subscriptions section. New SaaS subscriptions appear in the Hourly/monthly subscriptions section. |

Code examples for SaaS product integration

The following code examples can help you integrate your software as a service (SaaS) product with the AWS Marketplace APIs that are required for publishing and maintaining your product.

Topics

- [ResolveCustomer code example \(p. 243\)](#)
- [GetEntitlement code example \(p. 244\)](#)
- [BatchMeterUsage code example \(p. 245\)](#)
- [BatchMeterUsage with usage allocation tagging code example \(Optional\) \(p. 246\)](#)

ResolveCustomer code example

The following code example is relevant for all pricing models. The Python example exchanges a `x-amzn-marketplace-token` token for a `CustomerIdentifier`, `ProductCode`, and `CustomerAWSAccountId`. The `CustomerAWSAccountId` is the AWS account ID associated with the subscription. This code runs in an application on your registration website, when you are redirected there from the AWS Marketplace Management Portal. The redirect is a POST request that includes the token.

For more information about `ResolveCustomer`, see [ResolveCustomer](#) in the *AWS Marketplace Metering Service API Reference*.

```
# Import AWS Python SDK and urllib.parse
import boto3
import urllib.parse as urlparse

# Resolving Customer Registration Token
formFields = urlparse.parse_qs(postBody)
regToken = formFields['x-amzn-marketplace-token']

# If regToken present in POST request, exchange for customerID
if (regToken):
    marketplaceClient = boto3.client('meteringmarketplace')
    customerData = marketplaceClient.resolve_customer(regToken)
    productCode = customerData['ProductCode']
    customerID = customerData['CustomerIdentifier']
    customerAWSAccountId = customerData['CustomerAWSAccountId']

    # TODO: Store customer information
    # TODO: Validate no other accounts share the same customerID
```

Example response

```
{
```

```
'CustomerIdentifier': 'string',  
'CustomerAWSAccountId': 'string',  
'ProductCode': 'string'  
}
```

GetEntitlement code example

The following code example is relevant for SaaS products with the contract and SaaS contract with consumption pricing model. The Python example verifies that a customer has an active entitlement.

For more information about GetEntitlement, see [GetEntitlement](#) in the *AWS Marketplace Entitlement Service API Reference*.

```
# Import AWS Python SDK  
import boto3  
  
marketplaceClient = boto3.client('marketplace-entitlement')  
  
# Filter entitlements for a specific customerID  
#  
# productCode is supplied after the AWS Marketplace Ops team has published  
# the product to limited  
#  
# customerID is obtained from the ResolveCustomer response  
entitlement = marketplaceClient.get_entitlements({  
    'ProductCode': 'productCode',  
    'Filter' : {  
        'CUSTOMER_IDENTIFIER': [  
            'customerID',  
        ]  
    },  
    'NextToken' : 'string',  
    'MaxResults': 123  
})  
  
# TODO: Verify the dimension a customer is subscribed to and the quantity,  
# if applicable
```

Example response

The returned value corresponds to the dimensions created when you created the product in the AWS Marketplace Management Portal.

```
{  
  "Entitlements": [  
    {  
      "CustomerIdentifier": "string",  
      "Dimension": "string",  
      "ExpirationDate": number,  
      "ProductCode": "string",  
      "Value": {  
        "BooleanValue": boolean,  
        "DoubleValue": number,  
        "IntegerValue": number,  
        "StringValue": "string"  
      }  
    }  
  ],  
  "NextToken": "string"  
}
```

BatchMeterUsage code example

The following code example is relevant for SaaS subscription and contract with consumption pricing models, but not for SaaS contract products without consumption. The Python example sends a metering record to AWS Marketplace to charge your customers for pay-as-you-go fees.

```
# NOTE: Your application will need to aggregate usage for the
#       customer for the hour and set the quantity as seen below.
#       AWS Marketplace can only accept records for up to an hour in the past.
#
# productCode is supplied after the AWS Marketplace Ops team has
# published the product to limited
#
# customerID is obtained from the ResolveCustomer response

# Import AWS Python SDK
import boto3

usageRecord = [
    {
        'Timestamp': datetime(2015, 1, 1),
        'CustomerIdentifier': 'customerID',
        'Dimension': 'string',
        'Quantity': 123
    }
]

marketplaceClient = boto3.client('meteringmarketplace')

response = marketplaceClient.batch_meter_usage(usageRecord, productCode)
```

For more information about BatchMeterUsage, see [BatchMeterUsage](#) in the *AWS Marketplace Metering Service API Reference*.

Example response

```
{
  'Results': [
    {
      'UsageRecord': {
        'Timestamp': datetime(2015, 1, 1),
        'CustomerIdentifier': 'string',
        'Dimension': 'string',
        'Quantity': 123
      },
      'MeteringRecordId': 'string',
      'Status': 'Success' | 'CustomerNotSubscribed' | 'DuplicateRecord'
    },
  ],
  'UnprocessedRecords': [
    {
      'Timestamp': datetime(2015, 1, 1),
      'CustomerIdentifier': 'string',
      'Dimension': 'string',
      'Quantity': 123
    }
  ]
}
```

BatchMeterUsage with usage allocation tagging code example (Optional)

The following code example is relevant for a SaaS subscription and contract with consumption pricing models, but not for SaaS contract products without consumption. The Python example sends a metering record with appropriate usage allocation tags to AWS Marketplace to charge your customers for pay-as-you-go fees.

```
# NOTE: Your application will need to aggregate usage for the
#       customer for the hour and set the quantity as seen below.
#       AWS Marketplace can only accept records for up to an hour in the past.
#
# productCode is supplied after the AWS Marketplace Ops team has
# published the product to limited
#
# customerID is obtained from the ResolveCustomer response

# Import AWS Python SDK
import boto3
import time

usageRecords = [
    {
        "Timestamp": int(time.time()),
        "CustomerIdentifier": "customerID",
        "Dimension": "Dimension1",
        "Quantity": 3,
        "UsageAllocations": [
            {
                "AllocatedUsageQuantity": 2,
                "Tags": [
                    { "Key": "BusinessUnit", "Value": "IT" },
                    { "Key": "AccountId", "Value": "123456789" },
                ]
            },
            {
                "AllocatedUsageQuantity": 1,
                "Tags": [
                    { "Key": "BusinessUnit", "Value": "Finance" },
                    { "Key": "AccountId", "Value": "987654321" },
                ]
            },
        ]
    }
]

marketplaceClient = boto3.client('meteringmarketplace')

response = marketplaceClient.batch_meter_usage(UsageRecords=usageRecords,
        ProductCode="testProduct")
```

For more information about BatchMeterUsage, see [BatchMeterUsage](#) in the *AWS Marketplace Metering Service API Reference*.

Example response

```
{
```

```
"Results": [
  {
    "Timestamp": "1634691015",
    "CustomerIdentifier": "customerID",
    "Dimension": "Dimension1",
    "Quantity": 3,
    "UsageAllocations": [
      {
        "AllocatedUsageQuantity": 2,
        "Tags": [
          { "Key": "BusinessUnit", "Value": "IT" },
          { "Key": "AccountId", "Value": "123456789" }
        ]
      },
      {
        "AllocatedUsageQuantity": 1,
        "Tags": [
          { "Key": "BusinessUnit", "Value": "Finance" },
          { "Key": "AccountId", "Value": "987654321" }
        ]
      }
    ],
    "MeteringRecordId": "8fjef98ejf",
    "Status": "Success"
  },
  {
    "Timestamp": "1634691015",
    "CustomerIdentifier": "customerID",
    "Dimension": "Dimension1",
    "Quantity": 3,
    "UsageAllocations": []
  }
]
```

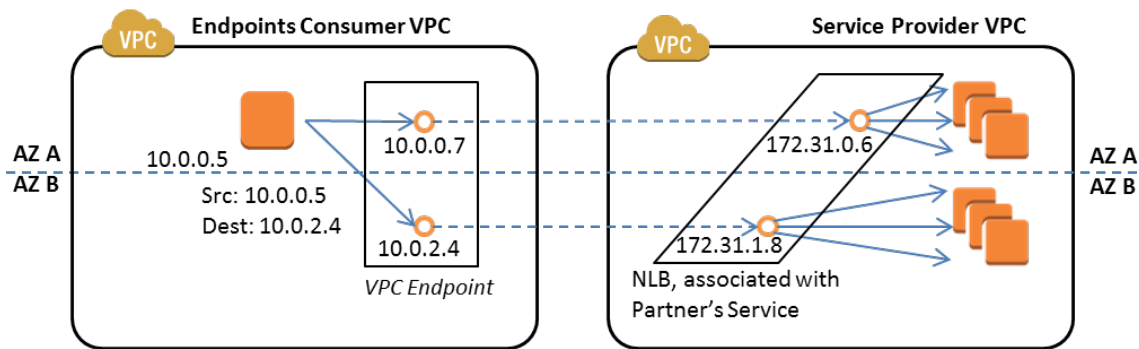
Using AWS PrivateLink with AWS Marketplace

AWS Marketplace supports AWS PrivateLink, a technology that allows you to use the Amazon network to provide buyers with access to products you sell through AWS Marketplace. This document outlines the process for configuring and delivering your products through an Amazon Virtual Private Cloud (VPC) endpoint using AWS PrivateLink technology.

In this document, we assume that you have working knowledge of several AWS services and the AWS Marketplace environment.

Introduction

As an AWS Marketplace seller, you can provide buyers access to your service through an Amazon VPC endpoint. This approach provides buyers with access to your service across the Amazon network using [AWS PrivateLink](#) technology. If you use AWS Marketplace to create and deliver this offering, buyers can discover your service in AWS Marketplace. Your buyers can also find your product in the list of available services for creating a VPC endpoint.



A [VPC endpoint](#) is a virtual device that enables AWS customers to create a private connection between their VPC and another AWS service without requiring access over the internet, through a NAT device, a VPN connection, or AWS Direct Connect. You can create an endpoint service through AWS Marketplace that makes it possible for buyers to use this technology to connect to your service. This connection method is more secure for your buyers because they access your service through the Amazon private network rather than through the Internet.

For each region where you want to offer your service, you create or use existing resources to configure a VPC, set up your service instances, set up a network load balancer, and register your services with the network load balancer by creating a service endpoint. After you complete those steps and test your offering, you provide your configuration information to the [AWS Marketplace Seller Operations](#) team.

AWS recommends that you provide a private DNS name that your buyers can use when they create VPC endpoints.

When buyers create their VPC endpoints, they have the option to enable a private DNS name. By choosing this option, the buyer's VPC service configures a [private hosted zone](#). If you provide the private DNS name, buyers can use it when configuring VPC endpoints to connect to your service. In the buyer's private hosted zone, the private DNS name (api.example.com) will point to the randomly generated DNS name(s) (vpce-1111111111111111-yyy-yyyyy.api.vpce.example.com) created for your endpoint service(s). The buyer's EC2 instances call the same unified DNS name (api.example.com) across different VPCs. Also, if public and private DNS names are same, the buyer can use the same public name when accessing your service from within or outside of the VPC.

For assistance with making your service available through AWS Marketplace, you can contact the [AWS Marketplace Seller Operations](#) team. When an AWS Marketplace buyer subscribes to your service and creates a VPC endpoint, your service is shown under **Your AWS Marketplace Services**. The MCO team uses the user-friendly DNS name for ease of discovery of your service when creating the VPC endpoint.

Your product is created as a software as a service (SaaS) product. Metering and billing is the same as with other AWS Marketplace SaaS products.

Configuring your product

To configure your product to be available through an Amazon VPC endpoint:

1. Create or use an existing [Amazon VPC](#).
2. Create (or use existing) [Amazon EC2](#) instance(s) for your product.
3. Create a [network load balancer](#) in each of the regions where you offer your product. AWS recommends that you include all [Availability Zones](#) (AZs) for a region.
4. Use the Amazon VPC console, the CLI, or supported SDKs to create a VPC endpoint service.
5. Verify that you can access the service through the network load balancer.

6. [Request a certificate from AWS Certificate Manager \(ACM\)](#) for your user-friendly DNS name. Before ACM issues a certificate, it validates that you own or control the domain names in your certificate request.
7. Delegate the subdomain of your user-friendly DNS name, such as `api.vpce.example.com`, to the name servers provided to you by the MCO team. In your DNS system, you must create a name server (NS) resource record to point this subdomain to the Amazon Route 53 name servers provided by the MCO team so that DNS names (such as `vpce-0ac6c347a78c90f8.api.vpce.example.com`) are publicly resolvable.
8. Allow access to your buyers' AWS accounts.

Note: You can use a supported SDK or this CLI command to automate access to accounts: `aws vpcev2 modify-vpc-endpoint-service-permissions --service-id vpce-svc-0123456789abcdef1 --add-allowed-principals arn:aws:iam::111111111111:root arn:aws:iam::222222222222:root`.

Submitting your product to AWS Marketplace

During the process of publishing your service to AWS Marketplace, you work with the [AWS Marketplace Seller Operations](#) team. To submit your PrivateLink-enabled product:

1. Email the following information to the [AWS Marketplace Seller Operations](#) team:
 - a. The endpoint and the AWS account used to create the endpoint. The endpoint is similar to this: `com.amazonaws.vpce.us-east-1.vpce-svc-0daa010345a21646`
 - b. The user-friendly DNS name for your service. This is the DNS name that AWS Marketplace buyers use to access your product.
 - c. The AWS account that you used to request certificates and the private DNS name buyers use to access the VPC endpoint.

The AWS Marketplace MCO team verifies your company's identity and the DNS name to use for the service you are registering (such as `api.vpce.example.com`). After verification, the DNS name overrides the default base endpoint DNS name.

Buyer access to VPC endpoints

AWS Marketplace buyers who are creating a VPC endpoint can discover your service in these situations:

- You followed the seller processes described earlier on this page to create or use an existing product.
- The buyer subscribes to your service.
- You added the buyer's AWS account to your list of allowed accounts.

When the buyer creates the VPC endpoint, they have the option to associate a private hosted zone with their VPC. The hosted zone contains a record set for the default private DNS name for the service that resolves to the private IP address of the endpoint network interfaces in their VPC.

Any buyer-hosted endpoint, including AWS Marketplace services, can provide permissions to all accounts (the "*" permission). However, when you use this approach, the services aren't included in the **Describe** calls or console unless you search by the service name. To display the services in the **Describe** calls, the buyer's AWS account must be explicitly added to the allow list by the service.

To access your service, buyers do the following:

1. Discover and subscribe to your service on AWS Marketplace.
2. Use the AWS Command Line Interface (AWS CLI), API, or the Amazon VPC console to discover your service and then establish a VPC endpoint to connect to your service in the subnets and AZs they use.

The endpoints are shown as elastic network interfaces in the subnets. Local IP addresses and region and zonal DNS names are assigned to the endpoints.

| Client-side DNS name | Name |
|----------------------|---|
| Regional | Vpce<0dc9a211a78c90f8>.api.vpce.example.com |
| IAD2 (1a) | us-east-1a- Vpce<0dc9a211a78c90f8>.api.vpce.example.com |
| IAD2 (1b) | us-east-1b- Vpce<0dc9a211a78c90f8>.api.vpce.example.com |

If you provided a default private DNS name and the buyer chooses **Enable Private DNS Name** (associated a private hosted zone) when creating a VPC endpoint, the buyer sees the regional default private DNS name to connect to your service.

| Name | Alias | Alias hosted zone ID | (Notes) |
|-----------------|--|----------------------|--------------|
| api.example.com | vpce<0dc9a211a78c90f8> api.vpce.example.com | Z00AABBCCDD | IAD1 IAD2 |

Appendix: Checklists

Use the following checklists to ensure that you configure and test your product before you submit it to the MCO team.

Product creation checklist

- Create (or use an existing) VPC and then configure it.
- Create and configure a network load balancer within the VPC.
- Register your service with your network load balancer by creating a VPC endpoint service.
- Provide the AWS account ID you used to configure the VPC endpoint to MCO.
- Provide the default endpoint service name (for example, com.amazonaws.vpce.us-east-1.vpce-svc-0bbb070044a2164) to MCO.
- Provide a user-friendly service DNS name (required) to override the randomly generated service DNS name. Request SSL certificates from ACM for the subdomain used for your user-friendly service DNS name. Provide these certificates and the AWS account ID you used to request them to the MCO team.
- Recommended: Provide a private DNS name.
- Create a process to inform and allow your AWS Marketplace buyers the option to connect to your service using AWS PrivateLink technology. Add AWS account IDs for your buyers to your allowed list of accounts.

Product testing

- Verify that your service is configured and discoverable.
- Verify that your service is discoverable over the network load balancer.
- Verify that a buyer can create a VPC endpoint and access your service. Use an AWS account you own that is not the account you used to set up your service.

Professional services products

As a seller, you can offer professional services to AWS Marketplace buyers. Professional services include services to assess, migrate, support, manage, and train others in how to use AWS services and products in AWS Marketplace. Sellers create a product offering that describes the services they provide, negotiate with customers to create an agreement on terms, and then create a custom offer for services through AWS Marketplace.

Buyers can find professional services products on the AWS Marketplace catalog by selecting **Professional Services** under **Categories**, choosing **Professional Services** under **Delivery methods**, and refine their search by **Publisher**, **Pricing model**, and **Pricing unit**. They're charged for the services in their AWS bill. They can use tools such as AWS Cost Explorer to centralize payments and manage their costs.

For more information about professional services products, see:

- [Getting started with professional services products \(p. 251\)](#)
- [Providing details for a professional services product \(p. 256\)](#)
- [Requirements for professional services products \(p. 258\)](#)

The following video explains more about managing professional services products in AWS Marketplace.

[Manage Professional service products in AWS Marketplace.](#)

Getting help

For assistance with your professional services products, contact your business development partner for AWS Marketplace or [contact us](#).

Getting started with professional services products

This topic describes how to get started with a professional services product, and goes through the steps to create your first product, and how to offer it to your customers. Your product definition tells your customers about the services that you offer and why they should select your company for those services. AWS Marketplace then allows them to contact you. You agree on a contract, and then you create a private offer that allows them to purchase your services for a fixed cost.

Topics

- [Prerequisites \(p. 252\)](#)
- [Creating a professional services product \(p. 252\)](#)
- [Creating private offers \(p. 253\)](#)
- [Editing product information \(p. 254\)](#)
- [Editing product pricing \(p. 254\)](#)
- [Editing product visibility \(p. 255\)](#)

- [Removing a professional services product \(p. 255\)](#)

Prerequisites

To sell professional services on AWS Marketplace, you must complete the following prerequisites:

- Have access to the AWS Marketplace Management Portal. This is the tool that you use to register as a seller and manage the products that you sell on AWS Marketplace. To learn more about getting access to the AWS Marketplace Management Portal, see [Policies and permissions for AWS Marketplace sellers \(p. 346\)](#).
- Register as an AWS Marketplace seller, and, if you want to charge for your products, submit your tax and banking information. To learn more about becoming an seller, see [Getting started as a seller \(p. 4\)](#).
- You must have a professional services product to offer that is related to an AWS service or at least one public product in AWS Marketplace. Your product must either directly support those products, or offer services that drive subscriptions to those products.

Note

Your product must be listed in at least one of these primary categories: Assessments, Implementation, Managed services, Premium support, or Training.

For more information about professional services product guidelines, see [Requirements for professional services products \(p. 258\)](#).

Creating a professional services product

The following procedure describes how to create a new professional services product in the AWS Marketplace Management Portal.

To create a professional services product

1. Open a web browser and sign into the [AWS Marketplace Management Portal](#).
2. From the **Products** menu, select **Professional services**. This page shows you all professional services products that you have already created, as well as any requests you have made for creating or modifying these products.
3. On the **Professional Services products** tab, select **Create professional services product**.
4. On the **Create product** page, provide the information for your product, and select **Submit**. For more information about the details that you must provide, see [Providing details for a professional services product \(p. 256\)](#).
5. (Optional) From the **Products** menu of AWS Marketplace Management Portal, select **Professional services**, then choose the **Requests** tab. Verify that you see your product request with the correct **Product title**, and that the **Request status** is **Under review**. Your product should be created in limited preview mode within a few minutes.

Note

You can return to the **Requests** tab of the **Professional services** page to see the status of your request at any time. Any errors in the creation process will appear here. You can select the request to see the request details or to fix errors.

When your product is initially created, it's only accessible to your AWS account (the one you used to create the product). If you view the product from the **Professional services** page, you can select **View on AWS Marketplace** to view the product details as they appear in AWS Marketplace for buyers. This detail listing isn't available to other AWS Marketplace users, unless you extend a private offer to them.

To learn how to make the product available publicly, see [Editing product visibility \(p. 255\)](#).

Creating private offers

When a potential buyer views your product on AWS Marketplace, they can't purchase it directly. When they attempt to subscribe, they are redirected to request a [private offer](#) from you. AWS Marketplace sends an email message to your AWS Marketplace seller account root user email address, informing you that the customer has requested a private offer. The following procedure describes how to respond to this request.

Note

When you create a private offer higher than \$250,000 through the AWS Marketplace Management Portal, additional approval may be required. For more information, contact your AWS Marketplace Business Development representative, or send an email message with your details to the AWS Marketplace Business operations team at mpcustdesk@amazon.com.

To create a private offer for a professional services product

1. Contact the customer to resolve any questions you have about the request. Agree on the offer terms before creating the private offer in AWS Marketplace. The buyer is not obligated to purchase your product, so it makes sense to agree before creating the offer.
2. Open a web browser and sign into the [AWS Marketplace Management Portal](#).
3. Select **Offers** from the menu, then select **Create private offer**.
4. On the **Create private offer** page, select the product that you want to create a private offer for. Only currently available products are included.
5. Enter the **Buyer account Ids** that you want to extend a private offer to. You can include up to 25 accounts in a single private offer. If the buyer used the request an offer feature, the email message that you received includes the buyer account Id for the requesting account.
6. Choose whether to allow buyers to pay for the product in installments. Typically, short contracts are paid in one payment. Longer contracts may have the option to pay in installments, but this is part of the agreement that you come to with the buyer. Select **Next**.
7. Complete the offer details, including the following information:
 - **Custom offer name** – Provide enough detail that you and the customers will recognize the offer. Include your company or product name and a description of the product. Do not include any personally identifiable information, including names, phone numbers, or addresses.
 - **Agreement end date** – The date that the agreed-to services end. For example, if you are offering support for 1 year, enter a date that is 1 year away from the date that the service will be available.
 - **Product dimensions** – The prices and units for the service that you are offering, as follows:
 - Lump sum payment offers – You can list each of the dimensions with their associated price (for example, you could have dimensions called *Silver*, *Gold*, and *Platinum*). The buyer can choose and pay for their preference.
 - Offers that include a payment schedule – You must choose a single dimension and provide a payment schedule with amounts and dates for each payment.

Note

If you want to create a zero dollar offer, you must select **I want to enable zero dollar prices** for confirmation. This precaution helps to prevent you from accidentally creating a free offer.

- **Service agreement** – Documents that define your service agreement with the customer. The documents that you upload (in text or PDF formats) are appended together into a single PDF document, so make sure that the file name is not required to understand the content.
 - **Offer expiration date** – The date the offer expires. This determines how long the buyer has to accept the offer and is unrelated to when the professional service will be available. You can extend the offer expiration date after your offer has been created.
8. Select **Next** when you're done editing the options.

9. On the **Review offer** page, make sure that the offer details are correct, and then choose **Create offer**.

Note

Your offer may take some time to be published. After it's published, you can view the offer on the **Manage offers** page. If you need to edit an offer (that has not yet been accepted), you can do so from that page.

10. After the offer is published, and available on the **Manage private offers** page, from the **Actions** menu for that offer, select **Copy offer URL**, and then send it in an email message to the buyer to accept.

Editing product information

The following procedure describes how to edit the product information for an existing professional services product in the AWS Marketplace Management Portal.

To edit product information

1. Open a web browser and sign into the [AWS Marketplace Management Portal](#).
2. From the **Products** menu, select **Professional services**. This page shows you all professional services products that you have already created, as well as any requests you have outstanding for creating or modifying these products.
3. Select an existing product that you would like to edit. Then, from the **Request changes** menu, select **Update product information**.
4. Make the changes to the details. For more information about the fields you can edit, see [Providing details for a professional services product \(p. 256\)](#).
5. Select **Submit** to create the request.
6. (Optional) If you are not already on the **Requests** tab of the **Professional services** page, then from the **Products** menu of AWS Marketplace Management Portal, select **Professional services**, then choose the **Requests** tab. Verify that you see your request with the correct **Product title**, and that the **Request status** is **Under review**. Your product will be updated with the changes you requested within a few minutes. If there is an error, you can view it here and resubmit your edits after fixing the errors.

Editing product pricing

The following procedure describes how to edit the pricing information for an existing professional services product in the AWS Marketplace Management Portal.

To edit product pricing

1. Open a web browser and sign into the [AWS Marketplace Management Portal](#).
2. From the **Products** menu, select **Professional services**. This page shows you all professional services products that you have already created, as well as any requests you have made for creating or modifying these products.
3. Select an existing product that you would like to edit, then from the **Request changes** menu, select **Update pricing dimensions**.

Note

You can only add new pricing dimensions through the AWS Marketplace Management Portal. To modify or remove previously created dimensions, contact the [AWS Marketplace Seller Operations team](#) with your request. In your request, include the product ID and details about what dimensions you want to change or remove.

4. Add any new pricing dimensions that you want. For more information about the pricing fields, see [Providing details for a professional services product \(p. 256\)](#).
5. Select **Submit** to create the request.
6. (Optional) From the **Products** menu of AWS Marketplace Management Portal, select **Professional services**, then choose the **Requests** tab. Verify that you see your request with the correct **Product title**, and that the **Request status** is **Under review**. Your product will be updated with the changes you requested within a few minutes. If there is an error, you can view it here and resubmit your edits after fixing the errors.

Editing product visibility

By default, products are created with limited visibility—a new product is only visible from your account. You can add other test accounts, or make the product publicly visible in the AWS Marketplace. The following procedure describes how to edit the visibility of an existing professional services product in the AWS Marketplace Management Portal.

To edit product visibility

1. Open a web browser and sign into the [AWS Marketplace Management Portal](#).
2. From the **Products** menu, select **Professional services**. This page shows you all professional services products that you have already created, as well as any requests you have outstanding for creating or modifying these products.
3. Select an existing product that you would like to edit. Then, from the **Request changes** menu, select **Update product visibility**.
4. Select **Contact us**. This launches a Contact Us page with the product information for the AWS Marketplace Seller Operations team to review and make the change.
5. Add details of your request, including whether you want the product to be public or private. If private, provide the account IDs you want to access, and then select **Submit** to send your request,

Note

To make a product visible in the public AWS Marketplace catalog requires a product review by the AWS Marketplace Seller Operations team to ensure that the product meets the product guidelines (see [Requirements for professional services products \(p. 258\)](#)). The request can take several days to complete.

Removing a professional services product

The following procedure describes how to remove an existing professional services product from the AWS Marketplace Management Portal.

To remove a product

1. Open a web browser and sign into the [AWS Marketplace Management Portal](#).
2. From the **Products** menu, select **Professional services**. This page shows you all professional services products that you have already created, as well as any requests you have outstanding for creating or modifying these products.
3. Select an existing product that you would like to edit. Then, from the **Request changes** menu, select **Unpublish a product**.
4. Select **Contact us**. This launches a **Contact Us** page with the correct information for the AWS Marketplace Seller Operations team to review and make the change.
5. Complete any additional information requested in the **Contact Us** page template, and select **Submit** to send your request.

Note

The request can take several days to complete. Products with active offers will be moved to restricted state until the last active subscription or contract is completed and then removed from AWS Marketplace. Products in restricted state are only visible to customers with active offers and sellers will not be able to extend new offers on these products.

Providing details for a professional services product

When you publish a professional services product on AWS Marketplace, you must provide the product metadata. This topic discusses information that is useful when you prepare or edit your product's details.

Note

For information about guidelines and requirements for professional services products, see [Requirements for professional services products \(p. 258\)](#).

Topics

- [Product descriptions \(p. 256\)](#)
- [Additional resources \(p. 257\)](#)
- [Support information \(p. 257\)](#)
- [Pricing dimensions \(p. 257\)](#)
- [Product visibility \(p. 258\)](#)

Product descriptions

The product descriptions section in the product details is the core of your product. It describes your product to your potential buyers so that they can make a purchasing decision. This section of the product details includes the following data:

- **Product title** – The name of your product. This is used to identify your product; it's visible on the product page and within search results. Provide a meaningful name for your product. It must be unique within AWS Marketplace.
- **SKU** – (Optional) Used to track your products on AWS Marketplace. This information is for your own use; buyers don't see it.
- **Short description** – A concise description of your product that appears on the tiles and underneath the product title in the AWS Marketplace product catalog.
- **Long description** – A longer, formatted description that describes the details of your product to buyers. List the product features, benefits, usage, and other information specific to the product. Use the available formatting to make the information easier to understand and scan.
- **Product logo** – This field is a public S3 URL that points to an image file that represents your product. The file must be in .png, .jpg, or .gif format, with a transparent or white background, under 5MB, and be between 110-10,000 pixels wide and tall. The logo is uploaded during product submission, and stored in AWS Marketplace. Modifying the contents of the URL will not modify the logo in AWS Marketplace after it is submitted.

Note

The S3 URL that you provide must be publicly available. This is a property of the S3 bucket in which the file resides. For more information, see [How do I edit public access settings for S3 buckets?](#) in the Amazon Simple Storage Service Console User Guide.

- **Highlights** – A set of one to three short points about your product, describing its key features or differentiators. At least one highlight is required.

- **Product categories** – The types of service that you provide. You must choose at least one, and up to three, categories. There are many categories to choose from, but professional services products must include at least one of the following:

Assessment

Evaluation of the customer's current operating environment to find the right solutions for their organization.

Implementation

Help with configuration, setup, and deployment of third-party software.

Premium support

Access to guidance and assistance from experts, designed for the customer's needs.

Managed services

End-to-end environment management on the customer's behalf.

Training

Tailored workshops, programs, and educational tools provided by experts to help the customer employees learn best practices.

- **Keywords for search results** – Provide up to three keywords that buyers might use to search for your product. You can list keywords in a comma-separated list, up to 250 characters.
- **Associated products** – *optional* – Include at least one public product from AWS Marketplace that your service either works with or supports. AWS Marketplace uses these products as input when selecting products to show on your product's details page or in **Related products** for those products.

Additional resources

In the **Additional resources** section of the product details, you can provide links to resources that you have created to help your customers. This is an optional set of one to three downloadable resources that are stored online. Examples of resources include product information sheets, whitepapers, or product manuals. For each resource, provide a name and a URL for the resource.

Support information

This section is a formatted text field that allows you to describe the support that you provide for your service.

Customers expect support on issues such as using the services, troubleshooting, and requesting refunds (if applicable). The support description should contain a statement about the level of support a customer can expect. Consider including support details for both pre-purchase questions and post-purchase issues.

Pricing dimensions

Pricing dimensions for professional services are packages that you offer. For example you might offer *Silver*, *Gold*, and *Platinum* support. Or you might offer 10, 20, or 50 hours of consulting. For each dimension you want to offer (at least one, up to 24), specify a name and a description. When you create a private offer for the product by working with a buyer directly, you set the actual prices for these dimensions.

Note

For information about how pricing dimensions are used, and how prices are set, see [Creating private offers \(p. 253\)](#).

Product visibility

Released products can be visible in AWS Marketplace to just your own account, to a small set of test accounts, or to all AWS accounts. By default, the product is published in private release. To change the product visibility, see [Editing product visibility \(p. 255\)](#).

Requirements for professional services products

AWS Marketplace maintains requirements for all products and offerings on AWS Marketplace. These requirements help to promote a safe, secure, and trustworthy curated digital catalog for our customers. We encourage sellers to review the implementation of additional controls and protocols as applicable to meet the needs of their specific products.

AWS Marketplace reviews all products and their related metadata when submitted to ensure that they meet or exceed current AWS Marketplace requirements. We review and adjust these requirements to meet our security requirements. In addition, AWS Marketplace continuously reviews products to verify that they meet any changes to these requirements. If products fall out of compliance, AWS Marketplace contacts you to update your product. In some cases, your products might temporarily be unavailable to new subscribers until issues are resolved.

Product setup guidelines

All professional services products must adhere to the following product setup guidelines:

- Your AWS account must be registered with AWS Marketplace as a paid seller.
- All pricing dimensions must relate to actual services offered and billed using AWS Marketplace.
- Your products must be listed in one of these categories: *Assessments, Implementation, Managed services, Premium support, or Training*.
- Besides the required professional services categories, your product should also be correctly categorized by choosing other appropriate categories that fit into services offered.
- Your product's logo must not be designed to confuse with the AWS logo, or any logo from an unrelated third party.
- Your product details must not contain offensive or explicit material. They must comply with the AWS Acceptable Use Policy available at <https://aws.amazon.com/aup/>.
- Your professional service product must directly support or offer services related to third-party software products listed on AWS Marketplace or help customers achieve specific outcomes related to the adoption or management of their AWS Cloud.

Customer information requirements

All professional services products must adhere to the following customer information requirements:

- Professional services products must be billed entirely through the listed dimensions on AWS Marketplace.
- You are not permitted to collect customer payment information for your professional services product listing on AWS Marketplace at any time, including credit card and bank account information.
- Any subscriber or prospective subscriber information provided by AWS to you in conjunction with your professional services products must be used solely in accordance with the Terms and Conditions for AWS Marketplace Sellers.

Product usage guidelines

All professional services products must adhere to the following product usage guidelines:

- After subscribing to the product in AWS Marketplace, customers should be able to access the service or be contacted by you (as the seller) within two business days of subscription. If the customer can't gain access to the service immediately, you must send an email message to them with specific details about when they will gain access. When the access is granted, you must send the customer an email message confirming that the access has been granted along with clear next steps.
- Customers must be able to easily get help with issues, such as using the services, troubleshooting, and requesting refunds (if applicable). Support contact options must be specified on the fulfillment landing page. The support description should contain a statement about the level of support a customer can expect.
- Your product's overview should include clear value propositions, key features, links to detailed documentation, and clear definitions of pre-purchase and post-purchase support of the services offered.
- Your products should have clear and straightforward service terms.

Architecture guidelines

All professional services products must adhere to the following architecture guidelines:

- Professional services products must be related to an AWS service or at least one public AWS Marketplace product (up to four) in which the product offers services for those related products directly or offers services that drive more subscribers to those related products.
- If the services offered require additional resources in the customer's infrastructure, follow these guidelines:
 - Provision resources in a secure way, such as by using the AWS Security Token Service or AWS Identity and Access Management (IAM).
 - Provide additional documentation including a description of all provisioned AWS services, IAM policy statements, and how an IAM role or user is deployed and used in the customer account.
 - Include a notification in the product description that explains that if the customer incurs additional AWS infrastructure costs, separate from their AWS Marketplace transaction, they're responsible for paying additional infrastructure charges.
 - If your product deploys an agent, provide instructions to the customer that describe how to deploy the agent in their AWS account.

Professional services product pricing

You can set the following product pricing model for your professional services products:

- **Private offers** - With seller private offers, there are options available for multi-year and custom duration contracts. For more information about multi-year and custom duration contracts, see [Private offers \(p. 44\)](#) and [Flexible payment scheduler \(p. 48\)](#).

You can set only one price per product.

For more information about pricing AWS Marketplace products, see [Product pricing \(p. 37\)](#).

Data products

AWS Data Exchange is a service that makes it easy for AWS customers to securely exchange file-based data sets in the AWS Cloud. As a provider, AWS Data Exchange eliminates the need to build and maintain any data delivery, entitlement, or billing technology. Providers in AWS Data Exchange have a secure, transparent, and reliable channel to reach AWS customers and grant existing customers their subscriptions more efficiently. The process for becoming an AWS Data Exchange provider requires a few steps to determine eligibility.

A data product has the following parts:

- **Product details** – This information helps potential subscribers understand what the product is. This includes a name, descriptions (both short and long), a logo image, and support contact information. Product details are filled out by providers.
- **Product offer(s)** – In order to make a product available on AWS Data Exchange, providers must define a public offer. This includes the prices and durations, data subscription agreement, refund policy, and the option to create custom offers.
- **Data set(s)** – A product can contain one or more data sets. A data set is a dynamic set of file-based data content. Data sets are dynamic and are versioned using revisions. Each revision can contain multiple assets.

For more information, including eligibility requirements, see [Providing Data Products on AWS Data Exchange](#) in the *AWS Data Exchange User Guide*.

Submitting your product for publication

You use the product submission process to make your products available on AWS Marketplace. Products can be quite simple, for example, a single Amazon Machine Image (AMI) that has one price structure. Or, products can be quite complicated, with multiple AMIs, AWS CloudFormation templates, and complex pricing options and payment schedules. You define your product offering and submit it through the AWS Marketplace Management Portal in one of two ways:

- Using the **Products** tab – For products that are less complex, you use the **Products** tab to completely define and submit your request.
- Using the **Assets** tab – For products that are more complex and require more definition, you download a product load form (PLF), add product details, and then upload the completed form using the **File upload** option.

Note

Data product providers must use the AWS Data Exchange console to publish products. For more information, see [Publishing Products](#) in the *AWS Data Exchange User Guide*.

We recommend that you start by using the **Products** tab to determine which approach to use. The following table lists configurations and the approach you use to submit your request. The first column is the pricing model for your product, and the other three columns are how the product is deployed to the customer.

| Pricing model | Products launched using single-node AMI | Products launched with AWS CloudFormation | Products launched as software as a service (SaaS) |
|---|---|---|---|
| Bring Your Own License (BYOL) | Products tab | Assets tab | |
| Free | Products tab | Assets tab | |
| Paid Hourly | Products tab | Assets tab | |
| Paid Hourly with Annual | Products tab | Assets tab | |
| Paid Monthly | Products tab | Assets tab | |
| Hourly with Monthly | Assets tab | Assets tab | |
| Paid Usage (AWS Marketplace Metering Service) | Products tab | Assets tab | |
| Contract Pricing | Products tab | | |
| SaaS Subscription | | | Products tab |
| SaaS Contract | | | Products tab |
| SaaS Legacy | | | Assets tab |

You can submit products individually or, if you use a product load form, you can submit multiple products or product updates at the same time. You cannot submit multiple products at the same time using the **Products** tab. If you are unclear on what products can be submitted in what manner, start by using the **Products** tab. If you have any problems making your submissions, contact the [AWS Marketplace Managed Catalog Operations \(MCO\)](#) team.

Using the Products tab

To access the **Products** tab, log in to the AWS Marketplace Management Portal. From the **Products** tab, choose either **Server**, **SaaS**, or **Machine learning**, depending on the type of product you are managing. A dashboard for that product type appears that contains all of your current products. If you choose the **Requests** tab, the dashboard displays any outstanding requests you have and your completed request history. Once you start creating a new product request, you can save your work in progress, and if necessary, create your request in several different sessions.

When you are ready to submit your product request, the request is reviewed by the AWS Marketplace team. You can monitor the status of your request on the product page for the type of product you are requesting. For new products, after your request is approved for publication, you receive a limited listing URL that you can use to preview and approve your submission. Your product offer is not published until you approve the submission. When you request an update to an existing product, the update is published without the need for you to review and approve the change. This includes adding or removing versions, and metadata changes.

You track the status of your requests under the **Requests** tab. The status will be one of the following:

- **Draft** – You have started the request process but have not submitted your request.
- **Submitted** – You have completed and submitted your request, and it is under review.
- **Action Required** – The AWS Marketplace team has reviewed your request and needs more information.
- **Approval Required** – The AWS Marketplace team has created the limited listing URL for your product. You must review and either approve or reject the URL before AWS Marketplace will publish. If you approve, the status changes to **Publishing Pending** while the site gets published. If you reject, the status returns to **Draft** so you can modify the request.
- **Publishing Pending** – You have approved the mock-up of your request and AWS Marketplace is publishing your product.
- **Expired** – You started the request process but did not complete it within six months, so the request expired.

If you have an entry with a status of **Submitted**, you can retract the submission. If you have an entry with a status of **Draft**, you can delete the request. This will allow you to start over. When you delete a **Draft** entry, the entry is moved to the **Request History** tab.

To add your product in the AWS GovCloud (US) AWS Region, you must [have an active AWS GovCloud \(US\) account](#) and comply with the AWS GovCloud (US) requirements, including export control requirements.

Company and product logo requirements

Your company logo and the logo for your products must conform to the following AWS Marketplace guidelines so that the user experience is uniform when browsing AWS Marketplace:

Product logo specifications – Your product logo image should have a transparent or white background and be 120 to 640 pixels in size, with a 1:1 or 2:1 (wide) ratio.

Company logo specifications – Your company logo image should have a transparent background and be 220 x 220 pixels in size, allowing for 10 pixels of padding on each side within.

Requirements for submitting paid repackaged software

If you are submitting a paid listing of either a repackaged open-source software (for example, open source AMI or container products with paid support), or software that was originally created by a vendor other than you (for example, reselling an AMI with Windows operating system), the following requirements must be met before submission:

- The product title must indicate the value added by your repackaging. Examples of product titles include: *Hardened <Product>*, *<Product> with added packages*, *<Product1> on <Product2>*.
- The product title must not contain any other language that is not otherwise supported with documentation. For example, the product title may not use the words *certified*, *original*, or *free* unless these are substantiated in the product details that you provide.
- The product short description must include a clear statement summarizing the product charges. The short description must begin with the phrase *This product has charges associated with it for...* For example, if a product includes charges for support from the seller, then the product description should state: *This product has charges associated with it for seller support*.
- The product logo must be same as the company logo which was used during your seller registration process. The product logo can differ from your company logo only if you use the official software logo, whereby you must receive explicit permission from the original software vendor. If explicit permission is obtained, a link to that documentation must be included in the notes section of the change request (or in the **Enter a brief description** field of the **File Uploads** page when using the product load form).
- For AMI products, the AMI name must not be reused from the original product. The AMI name must begin with the seller name and follow this format: [Seller Name] [name-given-to-ami].

If the paid listing is for a standalone software product that was not created by your company and there is no intellectual property added to the product (for example, bundling additional software libraries or adding special configuration) then, along with the earlier requirements, the following requirements must also be met:

- Product title must include the seller name (along with the value added, as described earlier). The seller name is the name used during seller registration. For example, *<Product> with maintenance support by <seller>*.
- The first line of the product's long description must begin with the phrase *This is a repackaged software product wherein additional charges apply for...* (or, if it's open source, *This is a repackaged open source software product wherein additional charges apply for...*). Then, the long description must include a clear statement summarizing what you are charging for, as well as additional details describing those features. For example, the long description of an open source product charging for additional support might start as: *This is a repackaged open source software product wherein additional charges apply for support with {SLA Details}*.

AWS CloudFormation-launched product (free or paid) or usage-based paid AMI product

Use a product load form (PLF) to submit products that AWS Marketplace customers launch by using AWS CloudFormation templates. The PLF is available through the AWS Marketplace Management Portal (AMMP).

Submitting your product

1. From the [AMMP](#), download the product load form (PLF) for your product.
2. Add your product definition, which includes product information (title, description, highlights), technical information (AMI_ID, Regions, instance types, OS), and pricing details (pricing model, Free Trial).
3. Submit your PLF following the instructions under the Instructions table of the spreadsheet.

The AWS Marketplace team reviews your product for policy and security compliance, software vulnerabilities, and product usability. If there are any questions or issues with a request, the AWS Marketplace team will contact you via an email message to discuss your request. Once approved, a mock-up of your product's page is created. After you review the page, you accept or reject the mock-up. Once approved, we add the page to the AWS Marketplace.

Updating your product

For products that you created by using the product load form (PLF), you also use the PLF to make changes to those products. You can make changes to the original PLF you completed or, if it's not available, you can start with a new PLF. Just like using the **Products** tab, you can add a new version, remove existing versions, and update pricing, instance types, Region availability, and metadata. To make an update, you prepare any updated product the same way you prepare a new product. After the product update is prepared, follow these steps:

1. Use your existing PLF or, from the [AWS Marketplace Management Portal](#), under the **Assets** tab, choose **File upload**. Under **Product load forms and seller guides**, you can download the PLF for your product.
2. Update your product submission in the PLF.
3. From the [AWS Marketplace Management Portal](#), under the **Assets** tab, choose **File Upload**.
4. On the **File Uploads** page, upload your updated PLF and any AWS CloudFormation templates. The file uploader provides a secure transfer mechanism and a history of submitted files. The uploader automatically notifies the AWS Marketplace team to begin processing your request. Include a description of the submission (adding new version, changing price, changing metadata, and so forth).

Your product submission is reviewed for policy and security compliance, software vulnerabilities, and product usability. If there are any questions or issues with a request, the AWS Marketplace team will contact you via an email message. Updates to existing product pages are processed and released directly without additional reviews.

Product changes and updates

Sellers can submit changes to their product at any time, and they will be processed as described earlier. However, some changes can only be made every 90 or 120 days, or when pending changes are in place. Examples include price changes and AWS Region or instance type changes. Common changes include:

- **New Version** – New versions of the software and rollouts of patches or updates. At your request, we can notify customers who have subscribed to your AWS Marketplace content about the availability of new versions or send upgrade instructions on your behalf.
- **Metadata change** – Changes to product information (Description, URLs, and Usage Instructions).
- **Pricing Change** – A change to the pricing amount. A notification to current customers is sent after the request is complete.

- **Pricing Model Change** – A change to the pricing model (for example, Hourly, Free, Hourly_Annual). Not all pricing model changes are supported, and all requests to change models must be reviewed and approved by the AWS Marketplace team. Any change from a free to a paid model presents significant impact to existing customers. An alternative is to propose a new product with additional features and encourage current customers to migrate.
- **Region or Instance change** – Adding or removing instances types or Regions.
- **Product takedown** - Remove a product page from AWS Marketplace to prevent new customers from subscribing. A notification to current customers is sent after the request is complete.

Timing and expectations

While we strive to process requests as quickly as possible, requests can require multiple iterations and review by the seller and AWS Marketplace team. Use the following as guidance for how long it will take to complete the process:

- Total request time normally takes 2–4 weeks of calendar time. More complex requests or products can take longer, due to multiple iterations and adjustments to product metadata and software.
- Review and processing of requests typically requires 3 business days. We will notify you if there are any issues that require additional action.
- We require a completed product request and AMI at least 45 days in advance of any planned events or releases, so we can prioritize the request accordingly.

If you have any questions about your request, contact the [AWS Marketplace Seller Operations](#) team.

Submitting AMIs to AWS Marketplace

All AMIs built and submitted to AWS Marketplace must adhere to all product policies. We suggest a few final checks of your AMI prior to submission:

- Remove all user credentials from the system; for example, all default passwords, authorization keys, key pairs, security keys or other credentials.
- Ensure that root login is disabled or locked. Only sudo access accounts are allowed.
- If you are submitting an AMI to be deployed into the AWS GovCloud (US) Region, you need to [have an active AWS GovCloud account](#) and agree to the [AWS GovCloud Requirements](#), including applicable export control requirements.

AMI self-service scanning

Self-service AMI scanning is available within the AWS Marketplace Management Portal. With this feature, you can initiate scans of your AMIs and receive scanning results quickly—typically in less than an hour—with clear feedback in a single location.

To begin sharing and scanning your AMI with self-service scanning

1. Navigate to <https://aws.amazon.com/marketplace/management/manage-products/>.
2. Select the AMI to share.
3. View your scan results.

After your AMI has successfully been scanned, you can follow the current process to submit it to the AWS Marketplace Seller Operations team by [uploading](#) your product load form (PLF). If you have any issues, contact the [AWS Marketplace Seller Operations](#) team.

To include your AMI in the self-service scanning list, the AMI must be in the `us-east-1` (N. Virginia) Region and owned by your AWS Marketplace seller account. If you need to grant other accounts access to the AWS Marketplace Management Portal, you must register those accounts as sellers. For more information, see [Seller registration process \(p. 6\)](#).

AMI cloning and product code assignment

After your AMI is submitted, AWS Marketplace creates cloned AMIs for each Region that you have indicated that software should be available in. During this cloning and publishing process, AWS Marketplace attaches a product code to the cloned AMIs. The product code is used to both control access and to meter usage. All submissions must go through this AMI cloning process.

Final checklist

To help avoid delays in publishing your product, use this checklist before you submit your product request.

Product usage

- Production-ready.
- Does not restrict product usage by time or other restrictions.
- Compatible with 1-click fulfillment experience.
- Everything required to use the product is contained within the software, including client applications.
- Default user uses a randomized password and/or creation of initial user requires verification that the buyer is authorized to use the instance using a value unique to the instance such as instance ID.

For free or paid products

- No additional license is required to use the product.
- Paid repackaged software meets the AWS Marketplace [Requirements for submitting paid repackaged software \(p. 263\)](#).
- Buyer does not have to provide personally identifiable information (for example, an email address) to use the product.

AMI preparation

- Use hardware virtual machine (HVM) virtualization and 64-bit architecture.
- Does not contain any known vulnerabilities, malware, or viruses.
- Buyers have operating system-level administration access to the AMI.
- Run your AMI through AMI Self-Service Scanning.

For Windows AMIs

- Use the most recent version of `Ec2ConfigService`, as described in [Configuring a Windows Instance Using EC2Config Service](#).
- The `Ec2SetPassword`, `Ec2WindowsActivate`, and `Ec2HandleUserData` plugins are enabled, as described in [Configuring a Windows Instance Using EC2Config Service](#).

- No Guest Accounts or Remote Desktop Users are present.

For Linux AMIs

- Root login is locked and disabled.
- No authorized keys, default passwords, or other credentials are included.
- All required fields are completed.
- All values are within specified character limits.
- All URLs load without error.
- Product image is at least 110px wide and between a 1:1 and 2:1 ratio.
- Pricing is specified for all enabled instance types (for hourly, hourly_monthly, and hourly_annual pricing models).
- Monthly pricing is specified (for hourly_monthly and monthly pricing models).

If you have any questions or comments about automated AMI building, contact the [AWS Marketplace Seller Operations](#) team.

Marketing your product

You can contribute to the success of your product by driving awareness of AWS Marketplace and by driving traffic directly to your product pages on AWS Marketplace. The following provides information and support to help you market the product or products that you have listed on AWS Marketplace.

180-day GTM Academy

The 180-day GTM Academy is available to all AWS Marketplace sellers, and it provides self-service go-to-market (GTM) resources to help you build, activate, and track demand generation campaigns for your offering in AWS Marketplace. You can:

- Fast track demand for your listings.
- Improve marketing return on investment and enhance customer messaging by integrating AWS and AWS Marketplace value proposition into your messaging.
- You can progress toward or within the [AWS Marketplace Go-to-Market Program Guide](#).

You can access the 180-day GTM Academy from the **Marketplace Resources** section of the [AWS Marketplace Management Portal](#).

Announcing your product's availability

We encourage you to broadly announce the availability of your product on AWS Marketplace. You can do this via press releases, tweets, blogs, or any other preferred media channels. We have provided sample text that you can include, along with guidelines and instructions for using our trademarks and issuing press releases.

We will review your blogs, tweets, and other non-press release announcements before going public to ensure consistency with AWS messaging and brand guidelines or voice. Submit your request for review to your AWS account manager. The review takes up to 10 business days to complete. Notify us when you post any tweets, blogs, or press releases, and we will do our best to repost to increase their visibility.

AWS Marketplace messaging

In your customer communications you might want to describe the purpose, goals, and benefits of purchasing your product using AWS Marketplace. Use the following messaging when referring to AWS Marketplace.

What is AWS Marketplace?

AWS Marketplace is an online store that makes it easy for customers to find, compare, and immediately start using the software and services that run on AWS. Visitors to AWS Marketplace can use 1-Click deployment to quickly launch preconfigured software and pay only for what they use, by the hour or month. AWS handles billing and payments, and software charges appear on the customer's AWS bill.

Why would a customer shop on AWS Marketplace?

Finding and deploying software can be challenging. AWS Marketplace features a wide selection of commercial and free IT and business software, including software infrastructure such as databases and application servers, IoT solutions, developer tools, and business applications, from popular sellers. AWS Marketplace enables customers to compare options, read reviews, and quickly find the software they want. Then they can deploy it to their own Amazon Elastic Compute Cloud instance using 1-Click or using the AWS Marketplace Management Portal.

Software prices are clearly posted on the website and customers can purchase most software immediately, with payment instruments already on file with Amazon Web Services. Software charges appear on the same monthly bill as AWS infrastructure charges.

Why would software or SaaS sellers sell on AWS Marketplace?

With AWS Marketplace, software and software as a service (SaaS) sellers with offerings that run on AWS can benefit from increased customer awareness, simplified deployment, and automated billing.

AWS Marketplace helps software and SaaS sellers of software and services that run on AWS find new customers by exposing their products to some of the hundreds of thousands of AWS customers, ranging from individual software developers to large enterprises.

Selling on AWS Marketplace enables independent software vendors (ISVs) to add hourly billing for their software without undertaking costly code changes. They simply upload an Amazon Machine Image (AMI) to AWS and provide the hourly cost. Billing is managed by AWS Marketplace, relieving sellers of the responsibility of metering usage, managing customer accounts, and processing payments, leaving software developers more time to focus on building great software.

Additionally, customers benefit from the ability to easily deploy preconfigured images of the software, simplifying onboarding for new customers.

Reviews on AWS Marketplace

AWS Marketplace provides the ability for customers to submit reviews on your product. We also provide the ability for syndicated reviewers such as G2, a business-to-business marketplace that curates independent product reviews, to integrate their syndicated reviews on AWS Marketplace.

AWS Marketplace customer reviews must meet the review guidelines listed in the user guide for buyers. Review submissions are not released on AWS Marketplace until after the submission is reviewed to verify it meets our review criteria. For more information on review guidelines, see [Product Reviews](#). Syndicated review organizations use their own unique processes to validate their reviews and aren't reviewed by AWS Marketplace before release. If you think that a syndicated review on your product doesn't meet the product review guidelines, or if you think a review on your product contains objectionable content, [contact the seller operations team](#).

The reviewer can also provide a star rating for your product based on a five-star rating system. The ratings are averaged to give the overall star rating for your product. Syndicated reviews can also include a star rating, but star ratings from syndicated reviews are not averaged in with the AWS customer star ratings.

The following are additional key points about the product review feature:

- You can't have a product review removed from AWS Marketplace. However, you can leave a comment on any review as long as the comment meets the review criteria governing product reviews.
- If you think that a review doesn't meet the review guidelines or contains objectionable content, you can [contact the seller operations team](#) and describe your concern.
- AWS customers searching for products in AWS Marketplace can search and filter results based on ratings, verified reviews, and externally sourced reviews. AWS customers see the externally sourced ratings alongside AWS customer ratings in search results.

- Syndicated reviews for your product are automatically added to AWS Marketplace at no cost to you. Because reviews are automatically added, you don't need to submit a request to have a syndicated review added.
- If you don't have any syndicated reviews for your product, you can contact the syndicated reviewer and follow their process for getting your product reviewed. For example, with G2, you can visit their website and claim your product page to start their review process.

Linking to AWS Marketplace

Your company likely has a web presence where it describes and promotes your product. We encourage you to highlight that the product is available to run on AWS and can be purchased using AWS Marketplace. To simplify the process for your customers to discover and deploy your software, we have provided instructions for linking your customers to your product.

Using the AWS Marketplace logo

The AWS Marketplace logo is a way to easily tell your customers that your software runs on AWS and is available in AWS Marketplace. If you would like to promote your software in AWS Marketplace, [download the archived folder \(.zip file\)](#), which contains multiple color treatments and file formats.

Linking directly to your product on AWS Marketplace

You can send your customers directly to the product's information page on AWS Marketplace by including deep links on your website or collateral. Use the following example link structure for browser-based linking.

```
https://aws.amazon.com/marketplace/pp/ASIN
```

Replace ASIN segment of the URL with your product's ASIN.

Example

```
https://aws.amazon.com/marketplace/pp/B00635Y2IW
```

The ASIN appears in the URL when you search for your application on aws.amazon.com/marketplace. Alternatively, you can consult with your account manager to find the ASIN.

Note

Test the links before using them to make sure that they direct your customers to the correct page.

Press releases

We encourage you to announce your product's availability on AWS Marketplace through any channel you prefer. However, all press releases that reference AWS Marketplace must be reviewed and signed off on by Amazon before any publication or announcement is made. While we encourage you to make announcements, we can't support joint press releases with AWS Marketplace sellers. We will, on a case-

by-case basis, support press releases with a quote from AWS. The quote must meet several conditions, including but not limited to: it announces a new product or service listed on AWS Marketplace or it includes a customer reference that uses AWS Marketplace.

All press releases must be drafted by you. We suggest the following headline: [Insert product name] Now Available on AWS Marketplace. Use the messaging in this document for consistency.

The press release **should**:

- Clearly and accurately describe how the announcement relates to Amazon.com
- Clarify your role on AWS and with customers
- Be customer-focused and emphasize the customer benefit

The press release **should not**:

- Use the terms *partners*, *partnership*, or *alliance* to describe the relationship. We prefer *agreement*, *teamed*, or *relationship*.
- Include a quote from an Amazon Web Services executive unless previously agreed upon.
- Include any sales projections or use .com by the merchant unless referring to the website in your company boilerplate.
- Refer to your organization as an associate of Amazon.com because this could be confused with Amazon Associates, our online affiliate program.
- Disclose proprietary information about Amazon.com or refer to our stock ticker symbol.

Have your press release reviewed by submitting it in text format to your account manager. Additionally, review the [Amazon Web Services trademark guidelines](#) before using any AWS trademarks. Guidelines specific to the AWS Marketplace trademark are in the following section.

AWS Marketplace trademark usage guidelines

These Guidelines apply to your use of the AWS Marketplace logo and trademark, (each the “Trademark” and collectively the “Trademarks”) in materials that have been approved in advance by Amazon.com, Inc. and/or its affiliates (“Amazon”). Strict compliance with these Guidelines is required at all times, and any use of a Trademark in violation of these Guidelines will automatically terminate any license related to your use of the Trademarks.

1. You may use the Trademark solely for the purpose expressly authorized by Amazon and your use must: (i) comply with the most up-to-date version of all agreement(s) with Amazon regarding your use of any of the Trademarks (collectively “Agreements”); (ii) comply with the most up-to-date version of these Guidelines; and (iii) comply with any other terms, conditions, or policies that Amazon may issue from time to time that apply to the use of the Trademark.
2. We will supply an approved Trademark image for you to use. You may not alter the Trademark in any manner, including but not limited to, changing the proportion, color, or font of the Trademark, or adding or removing any element(s) from the Trademark.
3. You may not use the Trademark in any manner that implies sponsorship or endorsement by Amazon other than by using the Trademark as specifically authorized under the Agreements.
4. You may not use the Trademark to disparage Amazon, its products or services, or in a manner which, in Amazon’s sole discretion, may diminish or otherwise damage or tarnish Amazon’s goodwill in the Trademark.
5. The Trademark must appear by itself, with reasonable spacing between each side of the Trademark and other visual, graphic or textual elements. Under no circumstance should the Trademark be placed on any background which interferes with the readability or display of the Trademark.

6. You must include the following statement in any materials that display the Trademark: "AWS Marketplace and the AWS Marketplace logo are trademarks of Amazon.com, Inc. or its affiliates.
7. You acknowledge that all rights to the Trademark are the exclusive property of Amazon, and all goodwill generated through your use of the Trademark will inure to the benefit of Amazon. You will not take any action that is in conflict with Amazon's rights in, or ownership of, the Trademark.

Amazon reserves the right, exercisable at its sole discretion, to modify these Guidelines and/or the approved Trademarks at any time and to take appropriate action against any use without permission or any use that does not conform to these Guidelines. If you have questions about these Guidelines, contact trademarks@amazon.com for assistance or write to us at the following address:

Amazon.com, Inc., Attention: Trademarks

PO Box 81226

Seattle, WA 98108-1226

Seller reports and data feeds

AWS Marketplace provides the following tools for collecting and analyzing information about your product sales:

- [Reports \(p. 273\)](#) that are automatically created and are available to all registered AWS Marketplace sellers.
- An [API \(p. 14\)](#) that enables you to access sections of those reports.
- [Data feeds \(p. 299\)](#) that provide additional customer information that you can use to identify customer information for transactions listed in the reports.

AWS Marketplace provides as much data as possible in reports and data feeds while adhering to the following:

- Amazon standards and tenets for protecting customer data.
- The terms and conditions that buyers accept when they buy a product on AWS Marketplace. As a seller, you are contractually bound to securely manage buyer data and to delete data upon buyer's request.

Seller reports

AWS Marketplace provides reports that include information about product usage, buyers, billing, and payment information. Reports are available to all registered AWS Marketplace sellers.

Here are some key points about report generation:

- Reports are generated daily, weekly, or monthly, depending on the report.
- Reports are generated at 00:00 UTC and cover through 24:00 UTC of the previous day.
- Reports are generated as .csv files.
- You can configure Amazon SNS to notify you when data is delivered to your encrypted S3 bucket. After you configure notifications, AWS sends notifications to the email address that is associated with the AWS account that you registered with on AWS Marketplace.

For information on how to configure notifications, see [Getting started with Amazon SNS](#) in the *Amazon Simple Notification Service Developer Guide*.

To cancel getting notification emails, contact the [AWS Marketplace Seller Operations](#) team.

- To learn about each report, you can download [sample reports](#).

Accessing reports

AWS Marketplace provides two ways to configure your reports:

- Using an API interface. The [AWS Marketplace Commerce Analytics Service \(p. 14\)](#) enables you to automatically access the data in your reports through an API interface. You can automate ingesting your information and download a portion of a report instead of the whole report. The service returns data asynchronously to a file in Amazon Simple Storage Service (Amazon S3) rather than directly as with a traditional API. The data is delivered in a machine-readable format so that you can import or incorporate the data into your systems.
- Using the reports dashboard in the [AWS Marketplace Management Portal](#). This dashboard provides reports for previous reporting periods.

You can control access to reports by using AWS Identity and Access Management (IAM) permissions.

Daily business report

The daily business report helps you understand how AWS customers are using your products on a daily basis and the estimated revenue from that usage. You only receive this report if relevant information is available. If you don't receive this report and think that you should have received it, contact the [AWS Marketplace Seller Operations](#) team.

You can access this report at the [AWS Marketplace Management Portal](#). If you are registered for the [the section called "AWS Marketplace Commerce Analytics Service" \(p. 14\)](#), you can also access your reports using the AWS SDK.

You can use a unique identifier for each customer to identify customers over time and across reports. The identifier enables you to track customer usage patterns so that you can estimate customer spend, gain insights into free trial usage, and annual usage trends.

Publication schedule

This report is published daily at 00:00 UTC and covers from 00:00 UTC through 23:59 UTC of the previous day. Any exceptions to the schedule are noted at the introduction of the daily business report section.

Topics

- [Section 1: Usage by instance type \(p. 274\)](#)
- [Section 2: Fees \(p. 276\)](#)
- [Section 3: Free trial conversions \(p. 277\)](#)
- [Section 4: New instances \(p. 277\)](#)
- [Section 5: New product subscribers \(p. 278\)](#)
- [Section 6: Canceled product subscribers \(p. 279\)](#)

Section 1: Usage by instance type

This section lists data with a row for each instance type that the customer uses. For instance, when the customer uses a product on one instance type and the same product on a different instance type, the report includes a row for each of the two instance types.

| Column name | Description |
|-----------------------|---|
| Customer Reference ID | A unique identifier that isn't the account ID. It helps track usage, revenue, and subscriptions by customers. |
| User's State | The billing address state that is associated with the account that is subscribed to the product. |
| User's Country | The two-character country code that is associated with the account that is subscribed to the product. This report uses ISO 3166-1 alpha-2 standard. |
| Product Title | The title of the product. |
| Product Code | The unique identifier for the product. |

| Column name | Description |
|-----------------------------|---|
| Instance Type | The instance type associated with the product usage: for example, t2.micro. |
| Usage Units | The number of units of usage that the customer used during the reporting period. |
| Usage Unit Type | The unit of measurement that meters the customer's usage. For example, hours or days. |
| Offering Description | The description for product offering. For example, the product is offered for hourly usage, free trial usage, or annual usage. |
| Estimated Revenue | The estimated revenue from the product usage. The billing is finalized at the end of the month. |
| Currency | The currency of the transaction. For example, if the transaction is in US dollars, the entry is USD. |
| Offer ID | The identifier for the offer that the buyer signed. |
| Offer Visibility | Whether the offer is a public, private, or enterprise contract offer. |
| Customer AWS Account Number | The ID of the account that the charges are billed to. |
| Customer Country | The two-character country code that is associated with the account that the charges are billed to. |
| Customer State | The billing address state that is associated with the account that the charges are billed to. This report uses ISO 3166-1 alpha-2 standard. |
| Customer City | The billing address city that is associated with the account that charges are billed to. |
| Customer Zip Code | The billing address zip code that is associated with the account that the charges are billed to. |
| Customer Email Domain | The email domain that is associated with the account that the charges are billed to. For example, if the email address is liu-jie@example.com, the entry is example.com. |
| Solution Title | The name of the solution. |
| Solution ID | The unique identifier for the solution. |
| Payer Reference ID | A unique identifier that isn't the account ID. It's associated with the account that fees are billed to. It helps with tracking usage, revenue, and subscriptions by customers across all of the AWS Marketplace financial reports. |
| Payer Address ID | A unique identifier that represents the customer's address. |

Section 2: Fees

This section includes fee-based transactions that are associated with products: for example, annual, monthly, SaaS contracts product fees, and data product subscription fees. The data in this section covers the 24-hour period 72 hours before the time that the report is generated. For example, if the report is generated on May 24, the data covers the 24-hour period for May 21.

| Column name | Description |
|-----------------------------|---|
| Customer Reference ID | A unique identifier that isn't the account ID. It helps track usage, revenue, and subscriptions by customers. |
| User's State | The billing address state that is associated with the account that is subscribed to the product. |
| User's Country | The two-character country code that is associated with the account that is subscribed to the product. This report uses ISO 3166-1 alpha-2 standard. |
| Product Title | The title of the product. |
| Product Code | The unique identifier for the product. |
| Amount | The usage fee. If there is a refund, this value is negative. If this entry is for an AWS Marketplace SaaS contract, the amount represents the fee for the dimension, not the entire contract. |
| Currency | The currency of the transaction. For example, if the transaction is in US dollars, the entry is USD. |
| Fee Description | The reason for the fee: for example, monthly fee, annual fee, or refund. |
| Customer AWS Account Number | The ID of the account that the charges are billed to. |
| Customer Country | The two-character country code that is associated with the account that the charges are billed to. This report uses ISO 3166-1 alpha-2 standard. |
| Customer State | The billing address state that is associated with the account that the charges are billed to. |
| Customer City | The billing address city that is associated with the account that charges are billed to. |
| Customer Zip Code | The billing address zip code that is associated with the account that the charges are billed to. |
| Customer Email Domain | The email domain that is associated with the account that the charges are billed to. For example, if the email address is liu-jie@example.com, the entry is example.com. |
| Start Date | The start date for an AWS Marketplace SaaS contract or data product subscription. |

| Column name | Description |
|--------------------|---|
| End Date | The end date for an AWS Marketplace SaaS contract or data product subscription. |
| Quantity | The number of units for a dimension that the contract specifies. |
| Dimension | The dimension that the contract specifies. |
| Solution Title | The name of the solution. |
| Solution ID | The unique identifier for the solution. |
| Payer Reference ID | A unique identifier that isn't the account ID. It's associated with the account that fees are billed to. It helps with tracking usage, revenue, and subscriptions by customers across all of the AWS Marketplace financial reports. |
| Payer Address ID | A unique identifier that represents the customer's address. |

Section 3: Free trial conversions

This section lists data for free trial starts, conversions and cancellations, and covers the previous 24-hour period.

| Column name | Description |
|---------------------------|---|
| Product Title | The title of the product. |
| Product Code | The unique identifier representing the product. |
| New Free Trials | The number of new free trials that are initiated in the reporting period. |
| Total Current Free Trials | The total number of active free trial subscriptions. |
| Converted Free Trials | The total number of subscriptions that moved from free trial to paid usage during the reporting period. |
| Non-Converted Free Trials | The total number of subscriptions that ended the free trial and didn't convert to paid usage. |
| Solution Title | The name of the solution. |
| Solution ID | The unique identifier for the solution. |

Section 4: New instances

This section lists data for new EC2 instance and instances types, and covers the previous 24-hour period.

| Column name | Description |
|-----------------------------|---|
| Customer Reference ID | A unique identifier that isn't the account ID. It helps track usage, revenue, and subscriptions by customers. |
| User's State | The billing address state that is associated with the account that is subscribed to the product. |
| User's Country | The two-character country code that is associated with the account that is subscribed to the product. This report uses ISO 3166-1 alpha-2 standard. |
| Product Title | The title of the product. |
| Product Code | The unique identifier for the product. |
| Type | The Amazon EC2 instance type. |
| Count | The number of EC2 instances. |
| Customer AWS Account Number | The ID of the account that the charges are billed to. |
| Customer Country | The two-character country code that is associated with the account that the charges are billed to. This report uses ISO 3166-1 alpha-2 standard. |
| Customer State | The billing address state that is associated with the account that the charges are billed to. |
| Customer City | The billing address city that is associated with the account that charges are billed to. |
| Customer Zip Code | The billing address zip code that is associated with the account that the charges are billed to. |
| Customer Email Domain | The email domain that is associated with the account that the charges are billed to. For example, if the email address is liu-jie@example.com, the entry is example.com. |
| Solution Title | The name of the solution. |
| Solution ID | The unique identifier for the solution. |
| Payer Reference ID | A unique identifier that isn't the account ID. It's associated with the account that fees are billed to. It helps with tracking usage, revenue, and subscriptions by customers across all of the AWS Marketplace financial reports. |
| Payer Address ID | A unique identifier that represents the customer's address. |

Section 5: New product subscribers

This section lists data for new buyers, and covers the previous 24-hour period.

| Column name | Description |
|-----------------------|--|
| Customer Reference ID | A unique identifier that isn't the account ID. It helps track usage, revenue, and subscriptions by customers. |
| User's State | The billing address state that is associated with the account that is subscribed to the product. |
| User's Country | The two-character country code that is associated with the account subscribed to the product. This report uses ISO 3166-1 alpha-2 standard. |
| Product Title | The title of the product. |
| Product Code | The unique identifier for the product. |
| Offer ID | The identifier for the offer the buyer signed. |
| Offer Visibility | Whether the offer is a public, private, or enterprise contract offer. |
| Customer Country | The two-character country code that is associated with the account that the charges are billed to. This report uses ISO 3166-1 alpha-2 standard. |
| Customer State | The billing address state that is associated with the account that the charges are billed to. |
| Customer City | The billing address city that is associated with the account that charges are billed to. |
| Customer Zip Code | The billing address zip code that is associated with the account that the charges are billed to. |
| Customer Email Domain | The email domain that is associated with the account that the charges are billed to. For example, if the email address is liu-jie@example.com, the entry is example.com. |
| Solution Title | The name of the solution. |
| Solution ID | The unique identifier for the solution. |
| Payer Reference ID | A unique identifier that isn't the account. It's associated with the account that fees are billed to. It helps with tracking usage, revenue, and subscriptions by customers across all of the AWS Marketplace financial reports. |
| Payer Address ID | A unique identifier that represents the customer's address. |

Section 6: Canceled product subscribers

This section lists data for buyer cancellations, and covers the previous 24-hour period.

| Column name | Description |
|-----------------------------|---|
| Customer Reference ID | A unique identifier that isn't the account ID. It helps track usage, revenue, and subscriptions by customers. |
| User's State | The billing address state that is associated with the account that is subscribed to the product. |
| User's Country | The two-character country code that is associated with the account that is subscribed to the product. This report uses ISO 3166-1 alpha-2 standard. |
| Product Title | The title of the product. |
| Product Code | The unique identifier for the product. |
| Subscribed Date | The date when the subscription started. |
| Offer ID | The identifier for the offer that the buyer signed. |
| Offer Visibility | Whether the offer is a public, private, or enterprise contract offer. |
| Customer AWS Account Number | The ID of the account that the charges are billed to. |
| Customer Country | The two-character country code that is associated with the account that the charges are billed to. This report uses ISO 3166-1 alpha-2 standard. |
| Customer State | The billing address state that is associated with the account that the charges are billed to. |
| Customer City | The billing address city that is associated with the account that charges are billed to. |
| Customer Zip Code | The billing address zip code that is associated with the account that the charges are billed to. |
| Customer Email Domain | The email domain that is associated with the account that the charges are billed to. For example, if the email address is liu-jie@example.com, the entry is example.com. |
| Solution Title | The name of the solution. |
| Solution ID | The unique identifier for the solution. |
| Payer Reference ID | A unique identifier that isn't the account ID. It's associated with the account that fees are billed to. It helps with tracking usage, revenue, and subscriptions by customers across all of the AWS Marketplace financial reports. |
| Payer Address ID | A unique identifier that represents the customer's address. |

Daily customer subscriber report

This report lists data for customers who purchased your products. This report doesn't specify current or past usage, only that a customer is subscribed to your product. You only receive this report if relevant information is available. If you don't receive this report and think that you should have, contact the [AWS Marketplace Seller Operations](#) team.

You can access this report at the [AWS Marketplace Management Portal](#). If you are registered for the [the section called "AWS Marketplace Commerce Analytics Service" \(p. 14\)](#), you can also access your reports using the AWS SDK.

The report has two sections: one for hourly and monthly subscriptions and one for annual subscriptions. The report includes the list of AWS account IDs for all customers who are subscribed to your products.

Publication schedule

This report is published daily at 00:00 UTC and covers from 00:00 UTC through 23:59 UTC of the previous day.

Topics

- [Section 1: Hourly and monthly subscriptions \(p. 281\)](#)
- [Section 2: Variable length subscriptions \(p. 282\)](#)

Section 1: Hourly and monthly subscriptions

This section lists data for all usage-based subscriptions as of the previous day at 23:59:59 UTC.

| Column name | Description |
|-----------------------------|---|
| Customer AWS Account Number | The account that is subscribed to the product. |
| Product Title | The title of the product. |
| Product Id | A unique identifier for the software product. |
| Product Code | The unique identifier for the software product. |
| Subscription Start Date | The start date for the subscription, formatted as YYYY-MM-DD. |
| Offer ID | The identifier for the offer that the buyer signed. |
| Offer Visibility | Whether the offer is a public, private, or enterprise contract offer. |
| Solution Title | The name of the solution. |
| Solution ID | The unique identifier for the solution. |
| Payer Reference ID | A unique identifier that isn't the account ID. It's associated with the account that fees are billed to. It helps with tracking usage, revenue, and subscriptions by customers across all of the AWS Marketplace financial reports. |
| Reseller account ID | The unique identifier for the consulting partner reseller. |

| Column name | Description |
|-----------------------|--|
| Reseller account name | The name of the consulting partner reseller. |

Section 2: Variable length subscriptions

This section lists data for all fee-based subscriptions as of the previous day at 23:59:59 UTC.

| Column name | Description |
|-----------------------------|---|
| Customer AWS Account Number | The ID of the account that is subscribed to the product. |
| Product Title | The title of the product. |
| Product Id | The unique identifier for the software product. |
| Product Code | A unique identifier for the software product. This information is also available as part of the Amazon EC2 instance metadata. |
| Subscription Id | The ID for the subscription. |
| Subscription Quantity | The total number of licenses that the customer purchased. |
| Subscription Type | The type of subscription. |
| Subscription Intent | Whether this offer is an upgrade or renewal of an earlier offer. |
| Offer ID | The identifier for the offer that the buyer signed. |
| Subscription Start Date | The date when the customer subscribed to the product, formatted as YYYY-MM-DD. |
| Previous Offer ID | The ID of the offer that preceded the upgrade or renewal offer, if one exists. |
| Offer Visibility | Whether the offer is a public, private, or enterprise contract offer. |
| Solution Title | The name of the solution. |
| Solution ID | The unique identifier for the solution. |
| Payer Reference ID | A unique identifier that isn't the account ID. It's associated with the account that fees are billed to. It helps with tracking usage, revenue, and subscriptions by customers across all of the AWS Marketplace financial reports. |
| Reseller account ID | The unique identifier for the consulting partner reseller. |
| Reseller account name | The name of the consulting partner reseller. |

Disbursement report

The disbursement report provides information about funds that we collected and disbursed to your bank accounts since the previous disbursement. Disbursements can include customer payments or refunds for a subscription to your product, and some taxes collected or refunded to the customer. You don't receive disbursement of funds until the funds are collected from the customer. Different customers have different payment terms with AWS, so some of the funds in each of the uncollected age categories might not be due from the customer.

Refunds appear as negative amounts because the money is returned to your customer after you authorize a refund.

This report is available on the AWS Marketplace Management Portal under the **Reports** tab. To create your own report similar to this one you can use the AWS Marketplace [Data feeds \(p. 299\)](#), including the [Example 1: Disbursements by product \(p. 310\)](#) as a base report to customize to meet your needs.

Publication schedule

This report is published 3-5 days after a disbursement has been initiated to transfer funds to your bank. In general, this is a report for sellers who receive disbursements on a monthly cadence. If there is no disbursement initiated, no disbursement report is generated.

Topics

- [Section 1: Disbursed amount by product \(p. 283\)](#)
- [Section 2: Disbursed amount by customer geography \(p. 285\)](#)
- [Section 3: Disbursed amount by instance hours \(p. 285\)](#)
- [Section 4: Age of uncollected funds \(p. 286\)](#)
- [Section 5: Age of disbursed funds \(p. 286\)](#)
- [Section 6: Age of past due funds \(p. 287\)](#)
- [Section 7: Uncollected funds breakdown \(p. 287\)](#)

Section 1: Disbursed amount by product

This section lists data for disbursements by product.

| Column name | Description |
|-----------------|--|
| Product | The title of the product. |
| Product Code | The unique identifier for the product. |
| SellerRev | The amount that is billed to the customer for the usage or fees of the product. |
| AWSRefFee | The amount of the AWS Marketplace fee. |
| SellerRevRefund | The amount of the subscription cost that is refunded to customers if any refunds were processed during the data coverage period. |
| AWSRefFeeRefund | The amount of the AWS Marketplace fee that is refunded if any refunds were processed during the data coverage period. |

AWS Marketplace Seller Guide
Disbursement report

| Column name | Description |
|-----------------------------|---|
| SellerRevCredit | The AWS credits that AWS Marketplace placed on the customer's account. |
| AWSRefFeeCredit | The AWS credits that AWS Marketplace placed on your account. |
| Net Amount | The total funds that we disbursed to you. This column is equal to the SellerRev column minus the AWSRefFee column. When a refund is given to a customer, this column is a negative number equal to the SellerRevRefund column minus the AWSRefFeeRefund column. |
| Transaction Reference ID | A unique identifier for the transaction that helps you correlate transactions across AWS Marketplace reports. |
| SellerUSSalesTax | The total amount of US sales and use tax that is billed for this transaction. |
| SellerUSSalesTaxRefund | The total amount of US sales and use tax that is refunded for this transaction if a refund was processed. |
| Customer AWS Account Number | The ID of the account that the charges are billed to. |
| Customer Country | The two-character country code that is associated with the account that the charges are billed to. This report uses ISO 3166-1 alpha-2 standard. |
| Customer State | The billing address state that is associated with the account that the charges are billed to. |
| Customer City | The billing address city that is associated with the account that charges are billed to. |
| Customer Zip Code | The billing address postal code that is associated with the account that the charges are billed to. |
| Customer Email Domain | The email domain that is associated with the account that the charges are billed to. For example, if the email address is liu-jie@example.com, the entry is example.com. |
| Solution Title | The name of the solution. |
| Solution ID | The unique identifier for the solution. |
| Payer Reference ID | A unique identifier that isn't the account ID. It's associated with the account that fees are billed to. It helps with tracking usage, revenue, and subscriptions by customers across all of the AWS Marketplace financial reports. |
| Payer Address ID | A unique identifier that represents the customer's address. |

Section 2: Disbursed amount by customer geography

This section lists data for disbursements by the customer's geographic location.

| Column name | Description |
|------------------------------|---|
| Settlement ID | The unique identifier of the disbursement. |
| Settlement Period Start Date | The starting date and time of the disbursement period. |
| Settlement Period End Date | The ending date and time of the disbursement period. |
| Deposit Date | The date and time when the disbursement occurred. |
| Disbursed Amount | The total amount of the disbursement. |
| Country Code | The two-character country code that is associated with the account that the charges are billed to. This report uses ISO 3166-1 alpha-2 standard. |
| State or Region | The billing address state that is associated with the account that the charges are billed to. |
| City | The billing address city that is associated with the account that charges are billed to. |
| Postal Code | The billing address postal code that is associated with the account that the software charges are billed to. |
| Net Amount by Tax Location | The total funds that are disbursed to the seller by tax location, less AWS Marketplace fees, refunds, and US sales and use tax. |
| Gross Amount by Tax Location | The total funds that are disbursed to the seller by tax location. |
| Seller U.S. Sales Tax | The total amount of US sales and use tax that is billed for this transaction on behalf of the Seller. (That is, related records in US Sales and Tax reports show "tax liable party" == "SELLER".) |
| Seller U.S. Sales Tax Refund | The total amount of US sales and use tax that is refunded for this transaction if a refund was processed, when such taxes were collected on behalf of the Seller. (That is, related records in US Sales and Tax reports show "tax liable party" == "SELLER".) |

Section 3: Disbursed amount by instance hours

This section lists data for disbursements by Amazon EC2 instance hours.

| Column name | Description |
|------------------------|---|
| Product | The title of the product. |
| Product Code | The unique identifier for the product. |
| Usage Type Description | The description of the usage, including offer type, Region, and instance type. |
| Rate | The rate per hour for the offer type, Region, and instance type. |
| User Count | The number of unique customers using the offer type, Region, and instance type. |
| Instance Hours | The number of hours that the instance consumed for the offer type, Region, and instance type. |
| Solution Title | The name of the solution. |
| Solution ID | The unique identifier for the solution. |

Section 4: Age of uncollected funds

This section lists data for uncollected funds, organized by the age. Uncollected funds might include amounts that aren't due yet.

| Column name | Description |
|-----------------------------------|--|
| Uncollected (< 31 days pending) | The total of funds billed but not collected for less than 31 days. |
| Uncollected (31–60 days pending) | The total of funds billed but not collected for between 31–60 days. |
| Uncollected (61–90 days pending) | The total of funds billed but not collected for between 61–90 days. |
| Uncollected (91–120 days pending) | The total of funds billed but not collected for between 91–120 days. |
| Uncollected (> 120 days pending) | The total of funds billed but not collected for more than 120 days. |
| Uncollected (overall) | The total of all funds billed but not collected. |

Section 5: Age of disbursed funds

This section lists data for collected funds since the previous disbursement.

| Column name | Description |
|-------------------------------|--|
| Collected (< 31 days pending) | The total of funds collected that were billed in the 0–31 day range. |

| Column name | Description |
|---------------------------------|---|
| Collected (31–60 days pending) | The total of funds collected that were billed in the 31–60 day range. |
| Collected (61–90 days pending) | The total of funds collected that were billed in the 61–90 days range. |
| Collected (91–120 days pending) | The total of funds collected that were billed in the 91–120 days range. |
| Collected (> 120 days pending) | The total of funds collected that were billed in the greater than 120 days range. |
| Collected (overall) | The total of all collected funds. |

Section 6: Age of past due funds

This section lists data for funds that have been accrued and are payable by the customer, but have not been paid in accordance with the customer's agreement with AWS.

| Column name | Description |
|------------------------|--|
| Past Due (< 31 days) | The total of funds that have accrued in the last 0–31 days and are due but that the customer hasn't paid. |
| Past Due (31–60 days) | The total of funds that have accrued in the last 31–60 days and are due but that the customer hasn't paid. |
| Past Due (61–90 days) | The total of funds that have accrued in the last 61–90 days that are due but that the customer hasn't paid. |
| Past Due (91–120 days) | The total of funds that have accrued in the last 91–120 days and are due but that the customer hasn't paid. |
| Past Due (> 120 days) | The total of funds that have accrued in the last 121 or more days and are due but that the customer hasn't paid. |
| Past Due (overall) | The total of funds that have accrued and are due but that the customer hasn't paid. |

Section 7: Uncollected funds breakdown

This section lists all uncollected funds, sorted by the payment due date.

| Column name | Description |
|--------------------------|--|
| Payer AWS Account Number | The account that the software charges are billed to. |

| Column name | Description |
|--------------------------|---|
| Product Code | The unique identifier for the product. |
| Gross Revenue | The amount that is billed for using the product or the fees for using the product. |
| AWS Revenue Share | The AWS fee amount that is deducted from the billed amount at settlement time. |
| Gross Refunds | The total amount of any refunds for the transaction. |
| AWS Refunds Share | The portion of the AWS fee that is refunded for the transaction. |
| Net Revenue | The net amount that is billed for this transaction, minus AWS fees, refunds, and US sales and use tax. |
| Currency | The currency of the transaction. For example, if the transaction is in US dollars, the entry is USD. |
| AR Period | The month and year of the transaction, in the format of YYYY-MM. |
| Transaction Reference ID | A unique identifier that represents the transaction, which you can use to correlate transactions across AWS Marketplace reports. |
| Opportunity Name | The unique identifier for a registered opportunity. |
| Opportunity Description | Any metadata in the registered opportunity. |
| Solution Title | The name of the solution. |
| Solution ID | The unique identifier of the solution. |
| Payer Reference ID | A unique identifier that isn't the account ID. It's associated with the account that fees are billed to. It helps with tracking usage, revenue, and subscriptions by customers across all of the AWS Marketplace financial reports. |
| Payer Address ID | A unique identifier that represents the customer's address. |
| Payment Due date | The payment due date in the format of YYYY-MM-DD. |

Monthly billed revenue report

The monthly billed revenue report provides you authoritative information about billed revenue every month for accounting and other financial reporting purposes. This report shows the total amounts that AWS bills to customers for hourly, annual, or monthly usage of your products. The report has four sections: billed amounts for hourly usage and monthly fees, variable-length subscriptions, field demonstration usage, and flexible payments.

Important

The amounts in this report reflect only revenue that we billed to customers, not amounts that we collected.

This report is available on the AWS Marketplace Management Portal under the **Reports** tab. If you're enrolled in the AWS Marketplace commerce analytics service, you can use API calls to pull down sections of this report. For more information, see [the section called "AWS Marketplace Commerce Analytics Service" \(p. 14\)](#).

Publication schedule

This report is published monthly on the fifteenth day of each month at 00:00 UTC. The report covers the previous calendar month from the first day of the month at 00:00 UTC through the last day of the month at 23:59 UTC. For example, the report that is published on May 15 covers from April 1 at 00:00 UTC through April 30 at 23:59 UTC.

Topics

- [Section 1: Billing and revenue data \(p. 289\)](#)
- [Section 2: Variable length subscriptions \(p. 291\)](#)
- [Section 3: AWS field demonstration usage \(p. 293\)](#)
- [Section 4: Contracts with flexible payment schedule \(p. 293\)](#)

Section 1: Billing and revenue data

This section lists data for usage billing, refunds, fees, and US sales and use tax that is collected.

| Column name | Description |
|------------------------|--|
| Customer Reference ID | A unique identifier that isn't the account ID. It helps track usage, revenue, and subscriptions by customers. |
| Country | The two-character country code that is associated with the account that the charges are billed to. This report uses ISO 3166-1 alpha-2 standard. |
| State | The billing address state that is associated with the account that the charges are billed to. |
| City | The billing address city that is associated with the account that charges are billed to. |
| Zip Code | The billing address postal code that is associated with the account that the charges are billed to. |
| Product Title | The title of the product. |
| Product Code | The unique identifier for the product. |
| Customer Billed Amount | The amount that is billed to the customer for the usage or monthly fees of the product. |
| AWS Listing Fee | The AWS Marketplace fee amount to be deducted from the billed amount. |

| Column name | Description |
|---------------------------------------|---|
| Refunds Amount | The total amount of the subscription cost refunded to customers if any refunds were processed during the data coverage period. |
| AWS Fee Refund | The portion of the AWS Marketplace fee refunded if any refunds were processed during the data coverage period. |
| Cost | The cost of goods to a reseller: for example, what a reseller pays you when they sell your product. |
| Partner Revenue Amount | The total amount billed for the transaction, net of AWS Marketplace fees, refunds, and US sales and use tax. |
| Currency | The currency of the transaction. For example, if the transaction is in US dollars, the entry is USD. |
| Transaction Reference ID | A unique identifier for the transaction that helps you correlate transactions across AWS Marketplace reports. |
| U.S. Sales Tax Customer Billed Amount | The total amount of US sales and use tax that is billed for this transaction on behalf of the Seller. (That is, related records in US Sales and Tax reports show "tax liable party" == "SELLER".) |
| U.S. Sales Tax Refunds Amount | The total amount of US sales and use tax that is refunded for this transaction if a refund was processed, when such taxes were collected on behalf of the Seller. (That is, related records in US Sales and Tax reports show "tax liable party" == "SELLER".) |
| Offer ID | The identifier for the offer that the buyer signed. |
| Offer Visibility | Whether the offer is a public, private, or enterprise contract offer. |
| Customer AWS Account Number | The ID of the account that the charges are billed to. |
| Customer Email Domain | The email domain that is associated with the account that the charges are billed to. For example, if the email address is <i>liu-jie@example.com</i> , the entry is <i>example.com</i> . |
| Opportunity Name | The unique identifier for a registered opportunity. |
| Opportunity Description | The metadata for the registered opportunity. |
| Solution Title | The name of the solution. |
| Solution ID | The unique identifier for the solution. |

| Column name | Description |
|--------------------|---|
| Payer Reference ID | A unique identifier that isn't the account ID. It's associated with the account that fees are billed to. It helps with tracking usage, revenue, and subscriptions by customers across all of the AWS Marketplace financial reports. |
| Payer Address ID | A unique identifier that represents the customer's address. |

Section 2: Variable length subscriptions

This section lists data for fee-based charges.

| Column name | Description |
|----------------------------|--|
| Customer Reference ID | A unique identifier that isn't the account ID. It helps track usage, revenue, and subscriptions by customers. |
| Country | The two-character country code that is associated with the account that the charges are billed to. This report uses ISO 3166-1 alpha-2 standard. |
| State | The billing address state that is associated with the account that the charges are billed to. |
| City | The billing address city that is associated with the account that charges are billed to. |
| Zip Code | The billing address zip code that is associated with the account that the charges are billed to. |
| Product Title | The title of the product. |
| Product Code | The unique identifier for the product. |
| Subscription Quantity | The number of total licenses that is specified as part of the variable-length subscription purchase. |
| Subscription Start Date | The start date of the variable-length subscription purchase. |
| Subscription End Date | The end date of the variable-length subscription purchase. |
| Subscription Instance Type | The instance type that is associated with the variable-length subscription purchase. |
| Customer Billed Amount | The amount that is billed for the usage, monthly fees, or both. |
| AWS Listing Fee | The AWS Marketplace fee amount that is deducted from the billed amount. |

| Column name | Description |
|---------------------------------------|---|
| Refunds Amount | The total amount refunded to customers if any refunds were processed during the data coverage period. |
| AWS Fee Refund | The portion of the AWS Marketplace fee refunded if any refunds were processed during the data coverage period. |
| Cost | The cost of goods to a reseller: for example, what a reseller pays you when they sell your product. |
| Partner Revenue Amount | The total amount that is billed for this transaction, net of AWS Marketplace fees, refunds, and US sales and use tax. |
| Currency | The currency of the transaction. For example, if the transaction is in US dollars, the entry is USD. |
| Transaction Reference ID | A unique identifier for the transaction that helps you correlate transactions across AWS Marketplace reports. |
| U.S. Sales Tax Customer Billed Amount | The total amount of US sales and use tax that is billed for this transaction on behalf of the Seller. (That is, related records in US Sales and Tax reports show "tax liable party" == "SELLER".) |
| U.S. Sales Tax Refunds Amount | The total amount of US sales and use tax that is refunded for this transaction if a refund was processed, when such taxes were collected on behalf of the Seller. (That is, related records in US Sales and Tax reports show "tax liable party" == "SELLER".) |
| Customer AWS Account Number | The ID of the account that the charges are billed to. |
| Customer Email Domain | The email domain that is associated with the account that the charges are billed to. For example, if the email address is liu-jie@example.com, the entry is example.com. |
| Offer ID | The identifier for the offer that the buyer signed. |
| Offer Visibility | Whether the offer is a public, private, or enterprise contract offer. |
| Contract Start Date | The start date for an AWS Marketplace SaaS contract. |
| Contract End Date | The end date for an AWS Marketplace SaaS contract. |
| Opportunity Name | The unique identifier for a registered opportunity. |
| Opportunity Description | The metadata for the registered opportunity. |
| Solution Title | The name of the solution. |

| Column name | Description |
|--------------------|---|
| Solution ID | The unique identifier for the solution. |
| Payer Reference ID | A unique identifier that isn't the account ID. It's associated with the account that fees are billed to. It helps with tracking usage, revenue, and subscriptions by customers across all of the AWS Marketplace financial reports. |
| Payer Address ID | A unique identifier that represents the customer's address. |

Section 3: AWS field demonstration usage

The section lists data for AWS [field demonstration usage](#) (p. 25) of your product. You can configure your product to allow us to demonstrate your product to potential customers. Any usage from the demonstrations is listed here.

| Column name | Description |
|------------------|--|
| Product Title | The title of the product. |
| Product Code | The unique identifier for the product. |
| Instance Type | The Amazon EC2 instance type that is associated with the field demonstration. |
| Usage Units | The number of units of usage that is associated with the product. |
| Usage Unit Types | The usage units that are associated with the usage unit count: for example, hours. |

Section 4: Contracts with flexible payment schedule

This section lists data for all contracts that you created with a flexible payment schedule in the previous reporting period.

| Column name | Description |
|-----------------------------|--|
| Customer AWS Account Number | The ID of the payer account that the charges are billed to. |
| Customer Country | The two-character country code that is associated with the payer account that the charges are billed to. This report uses ISO 3166-1 alpha-2 standard. |
| Customer State | The billing address state that is associated with the payer account that the charges are billed to. |
| Customer City | The billing address city that is associated with the payer account that charges are billed to. |

| Column name | Description |
|----------------------------|---|
| Customer ZIP Code | The billing address zip code that is associated with the payer account that the charges are billed to. |
| Customer Email Domain | The email domain that is associated with the payer account that the charges are billed to. For example, if the email address is <i>liujie@example.com</i> , the entry is <i>example.com</i> . |
| User Reference ID | The account of the payer account that the charges are billed to. |
| User AWS Account Number | The ID of the account that subscribed to the product. |
| Product ID | The unique identifier for the product. |
| Product Title | The title of the product. |
| Product Type | The type of product. |
| AWS Marketplace Offer ID | The identifier for the offer that the buyer signed. |
| Contract Create Date | The contract creation date, which is the date that an account subscribes to the offer. |
| Contract Expiration Date | The date when the contract expires. |
| Total Contract Value (USD) | The total value of the contract in USD. |
| # of Payments | The number of payments that are scheduled for the contract. |
| Invoice Date | The date the invoice is created. |
| Invoice Amount (USD) | The amount that is billed on the invoice in USD. |
| Payer Reference ID | A unique identifier that isn't the account ID. It's associated with the account that fees are billed to. It helps with tracking usage, revenue, and subscriptions by customers across all of the AWS Marketplace financial reports. |

Sales compensation report

The report lists monthly billed revenue with additional customer information that isn't in the standard [the section called "Monthly billed revenue report" \(p. 288\)](#). The report lists the total amounts that AWS bills to customers for hourly, annual, or monthly usage of your product.

Important

The amounts in this report reflect only revenue that is billed to customers, not amounts that are collected from customers.

The sales compensation report and the information that is shared with you as part of this program constitute Amazon's Confidential Information under our nondisclosure agreement with you or, if no such agreement exists, the Terms and Conditions for AWS Marketplace sellers. You can use this information only for compensating your sales representatives by mapping AWS Marketplace revenue to the representatives by company name, geography, and AWS account ID. You can share this information

with employees who need to know it to understand the source of commissions that is payable to them. Your use and sharing of such information must comply with the obligations in our nondisclosure agreement with you and the terms and conditions for AWS Marketplace sellers, including, without limitation, Section 3.8 of the Terms and Conditions for AWS Marketplace sellers.

To create your own report similar to this one you can use the AWS Marketplace [Data feeds](#) (p. 299), including the [Example 2: Sales compensation report](#) (p. 314) as a base report to customize to meet your needs.

Publication schedule

This report is published monthly, on the fifteenth day of each month at 00:00 UTC. The report covers the previous calendar month from 00:00 UTC through 23:59 UTC of the last calendar day of the month. For example, the report published on May 15 covers from April 1 at 00:00 UTC through April 30 at 23:59 UTC.

Billed revenue

The billed revenue section of this report includes usage and fee-based charges from the previous calendar month. The following are the column names and descriptions.

Note

In this report, *listing fee* is the percentage of transaction proceeds (except for those from resale by authorized resellers of authorized resale products) determined in accordance with the tiered listing fee.

| Column name | Description |
|-----------------------------|--|
| Customer AWS Account Number | The account that the charges are billed to. |
| Country | The two-character country code that is associated with the account that the charges are billed to. This report uses ISO 3166-1 alpha-2 standard. |
| State | The billing address state that is associated with the account that the charges are billed to. |
| City | The billing address city that is associated with the account that the charges are billed to. |
| Zip Code | The billing address zip code that is associated with the account that the charges are billed to. |
| Email Domain | The email domain that is associated with the account that the charges are billed to. For example, if the email address is liu-jie@example.com, the entry is example.com. |
| Product Code | The unique identifier for the product. |
| Product Title | The title of the product. |
| Gross Revenue | The amount that is billed for using the product or the monthly fees for using the product. |
| AWS Revenue Share | The AWS fee amount that is deducted from the billed amount at settlement time. It appears in the the section called "Disbursement report" (p. 283). |

| Column name | Description |
|--------------------------|---|
| Gross Refunds | The total amount of any refunds for the transaction. |
| AWS Refunds Share | The portion of the AWS fee that is refunded for the transaction. |
| Net Revenue | The net amount that is billed for this transaction, minus AWS fees, refunds, and US sales and use tax. |
| Currency | The currency of the transaction. For example, if the transaction is in US dollars, the entry is USD. |
| AR Period | The month and year of the transaction, in the format of YYYY-MM. |
| Transaction Reference ID | A unique identifier that represents the transaction, which you can use to correlate transactions across AWS Marketplace reports. |
| Opportunity Name | The unique identifier for a registered opportunity. |
| Opportunity Description | Any metadata in the registered opportunity. |
| Solution Title | The name of the solution. |
| Solution ID | The unique identifier of the solution. |
| Payer Reference ID | A unique identifier that isn't the account ID. It's associated with the account that fees are billed to. It helps with tracking usage, revenue, and subscriptions by customers across all of the AWS Marketplace financial reports. |
| Payer Address ID | A unique identifier that represents the customer's address. |

US sales and use tax report

This monthly report provides sellers with information about US sales and use tax that AWS collects from sales and use transactions in AWS Marketplace. The report includes both products that sellers enroll in the AWS Marketplace US sales tax collection service and products that AWS is required to collect and remit tax on.

For sales of products enrolled in the tax calculation service, the report includes calculated US sales and use tax for products with a product tax code. Any products without a product tax code appear in this report with a tax value of \$0.00 USD. For sales of products that are not eligible for the tax calculation service because of enacted marketplace facilitator rules, you will see amounts that AWS has collected and remitted as AWS, based on our internal tax decisions. For more information, see [AWS Marketplace Sellers & Tax Collection](#) on Amazon Web Services Tax Help.

To map transactions between the disbursement report and this report, use the `Transaction Reference ID`.

This report is available on the AWS Marketplace Management Portal under the **Reports** tab. If you're enrolled in the AWS Marketplace commerce analytics service, you can use API calls to pull down sections

of this report. For more information, see [the section called "AWS Marketplace Commerce Analytics Service" \(p. 14\)](#).

Publication schedule

This report is published monthly on the fifteenth day of each month at 00:00 UTC. The report covers the previous calendar month from the first day of the month at 00:00 UTC through the last day of the month at 23:59 UTC. For example, the report that is published on May 15 covers from April 1 at 00:00 UTC through April 30 at 23:59 UTC.

US sales and use tax records

This section lists data for US sales tax amounts that result from software charges.

| Column name | Description |
|----------------------|---|
| Line Item ID | A unique identifier for a line item. Refund transactions have the same line item ID as their forward tax transactions. |
| Customer Bill ID | The unique identifier for a customer bill. |
| Product Name | The name of the product purchased. |
| Product Code | The unique identifier for the product. |
| Product Tax Code | A standard code to identify the tax properties for a product. You choose the properties when you create or modify the product. |
| Seller ID | A unique identifier for the seller of record of the transaction. |
| Seller Name | The legal name of the seller. |
| Transaction Date | The date of the transaction. |
| Total Adjusted Price | The final price for the transaction. |
| Total Tax | The total tax that is charged for the transaction. |
| Base Currency Code | The base currency code for all AWS Marketplace transactions. This entry is always USD. |
| Bill to City | The billing address city that is associated with the payer account that we bill software charges to. |
| Bill to State | The billing address zip code that is associated with the payer account that the software charges are billed to. |
| Bill to Postal Code | The billing address postal code that is associated with the payer account that the software charges are billed to. |
| Bill to Country | The two-character country code that is associated with the payer account that the software charges are billed to. This report uses ISO 3166-1 alpha-2 standard. |

| Column name | Description |
|--|--|
| Transaction Type Code | <p>The type code of the transaction. Valid values:</p> <ul style="list-style-type: none"> • AWS: A forward tax transaction • REFUND: A full or partial refund • TAXONLYREFUND: A tax-only refund <p>Refund transactions share the line item ID with their original forward transactions.</p> |
| Display Price Taxability Type | The taxability type for the price that appears to customers. All AWS Marketplace offerings are exclusive. |
| Tax Location Code Taxed Jurisdiction | The vertex geocode that is associated with the taxed location. |
| Tax Type Code | The type of tax that is applied to the transaction. The possible values are <code>None</code> , <code>Sales</code> , and <code>SellerUse</code> . |
| Jurisdiction Level | The jurisdiction level of the address that is used for tax location. The possible values are <code>State</code> , <code>County</code> , <code>City</code> , and <code>District</code> . |
| Taxed Jurisdiction | The name of the taxed jurisdiction. |
| Taxable Sale Amount | The amount of the transaction that is taxable, by jurisdiction level. |
| Nontaxable Sale Amount | The amount of the transaction that is nontaxable, by jurisdiction level. |
| Tax Amount | The tax that is charged at the jurisdiction level. |
| Tax Jurisdiction Tax Rate | The tax rate that is applied at the jurisdiction level. |
| Tax Calculation Reason Code | Whether the transaction is taxable, not taxable, exempt, or zero-rated, organized by the jurisdiction level. |
| Date Used For Tax Calculation | The date that is used for calculating tax on the transaction. |
| Customer Exemption Certificate ID | The certificate ID of the exemption certificate. |
| Customer Exemption Certificate ID Domain | Where the certificate is being stored in Amazon systems. |
| Customer Exemption Certificate Level | The jurisdiction level that supplied the exemption. |
| Customer Exemption Code | The code that specifies the exemption: for example, <code>RESALE</code> . |
| Customer Exemption Domain | The Amazon system that is used to capture the customer exemption information, if information is available. |

| Column name | Description |
|--------------------------|--|
| Customer Reference ID | A unique identifier that isn't the account ID. It helps track usage, revenue and subscriptions by customers. |
| Transaction Reference ID | A unique identifier for the transaction that helps you correlate transactions across AWS Marketplace reports. |
| Payer Reference ID | A unique identifier that isn't the account ID. It's associated with the account that fees are billed to. It helps with tracking usage, revenue, and subscriptions by customers across all of the AWS Marketplace financial reports. |
| Tax Liable Party | This field will either be populated with <code>seller</code> or <code>aws</code> . If the seller is the tax liable party, they are responsible for their own collection and remittance obligations based on their tax decision. If <code>aws</code> is the tax liable party sales tax will be collected and remitted by AWS. For more information, see AWS Marketplace Sellers & Tax Collection on Amazon Web Services Tax Help. |

Data feeds

AWS Marketplace provides a number of data feeds to help sellers collect and analyze information about your product sales. Data feeds are available to all registered AWS Marketplace sellers. Since data feeds are generated within a day, they contain the most current data available.

This page provides an overview of data feeds, and explains how to access and use them. Subsequent pages describe each data feed.

Storage and structure of data feeds

Data feeds collect and deliver comma-separated value (CSV) files to an encrypted Amazon S3 bucket that you provide. The CSV files have the following characteristics:

- They follow [4180 standards](#).
- Character encoding is UTF-8 without BOM.
- Commas are used as separators between values.
- Fields are escaped by double quotation marks.
- `\n` is the line feed character.
- Dates are reported in the UTC time zone, are in ISO 8601 date and time format, and are accurate within 1 second.
- All `*_period_start_date` and `*_period_end_date` values are inclusive, which means that `23:59:59` is the last possible timestamp for any day.
- All monetary fields are preceded with a currency field.
- Monetary fields use a period (.) character as a decimal separator, and don't use a comma (,) as a thousands separator.

Data feeds are generated and stored as follows:

- Data feeds are generated within a day, and contain 24 hours of data from the previous day.
- In the Amazon S3 bucket, data feeds are organized by month using the following format:

bucket-name/data-feed-name_version/year=YYYY/month=MM/data.csv

- As each daily data feed is generated, it is appended to the existing CSV file for that month. When a new month starts, a new CSV file is generated for each data feed.
- Information in data feeds is backfilled from 2010/01/01 to 2020/04/30 (inclusive) and is available in the [CSV file \(p. 299\)](#) in the `year=2010/month=01` subfolder.

You may notice cases where the current month's file for a given data feed contains only column headers, and no data. This means that there were no new entries for that month for the feed. This can happen with data feeds that are updated less frequently, like the product feed. In these cases, data is available in the backfilled folder.

- In Amazon S3, you can create an [Amazon S3 lifecycle policy](#) to manage how long to keep files in the bucket.
- You can configure Amazon SNS to notify you when data is delivered to your encrypted S3 bucket. For information on how to configure notifications, see [Getting started with Amazon SNS](#) in the *Amazon Simple Notification Service Developer Guide*.

Historization of the data

Each data feed includes columns that document the history of the data. Except for `valid_to`, these columns are common to all data feeds. They're included as a common history schema and are useful in querying the data.

| Column name | Description |
|--------------------------|--|
| <code>valid_from</code> | The first date that the value for the primary key is valid for in relation to values for other fields. |
| <code>valid_to</code> | This column is only shown on the Address (p. 321) data feed and is always blank. |
| <code>insert_date</code> | The date a record was inserted into the data feed. |
| <code>update_date</code> | The date the record was last updated. |
| <code>delete_date</code> | This column is always blank. |

The following shows an example of these columns.

| <code>valid_from</code> | <code>valid_to</code> | <code>insert_date</code> | <code>update_date</code> | <code>delete_date</code> |
|-------------------------|-----------------------|--------------------------|--------------------------|--------------------------|
| 2018-12-12T02:00:00Z | | 2018-12-12T02:00:00Z | 2018-12-12T02:00:00Z | |
| 2019-03-29T03:00:00Z | | 2019-03-29T03:00:00Z | 2019-03-29T03:00:00Z | |
| 2019-03-29T03:00:00Z | | 2019-03-29T03:00:00Z | 2019-04-28T03:00:00Z | |

The `valid_from` and `update_date` field together form a *bi-temporal data model*. The `valid_from` field, as it is named, tells you when the item is valid from. If the item was edited, it can have multiple

records in the feed, each with a different `update_date`, but the same `valid_from` date. For example, to find the current value for an item, you would find the record with the most recent `update_date`, from the list of records with the most recent `valid_from` date.

In the example above, the record was originally created 2018-12-12. It was then changed on 2019-03-29 (for example, if the address in the record changed). Later, on 2019-04-28, the address change was corrected (so the `valid_from` didn't change, but the `update_date` did). Correcting the address (a rare event) retroactively changes the record from the original `valid_from` date, so that field didn't change. A query to find the most recent `valid_from` would return two records, the one with the latest `update_date` gives you the actual current record.

Accessing data feeds

To access data feeds, you need to configure your environment to receive data feeds to an encrypted Amazon S3 bucket. AWS Marketplace provides an [AWS CloudFormation template](#) that you can use to simplify configuration.

To use the AWS CloudFormation template to configure your environment to receive data feeds

1. Open a web browser and sign into the [AWS Marketplace Management Portal](#), then go to [Set up customer data storage](#).
2. Choose **Create resources with AWS CloudFormation template** to open the template in the AWS CloudFormation console in another window.
3. In the template, specify the following and then choose **Next**:
 - Stack name – The collection of resources you're creating to enable access to data feeds.
 - Amazon S3 bucket name – The bucket for storing data feeds.
 - (Optional) Amazon SNS topic name – The topic for receiving notifications when AWS delivers new data to the Amazon S3 bucket.
4. On the **Review** page, confirm your entries and choose **Create stack**. This will open a new page with the CloudFormation status and details.
5. From the **Resources** tab, copy Amazon Resource Names (ARNs) for the following resources from the CloudFormation page into the fields on the AWS Marketplace [Set up customer data storage](#) page:
 - Amazon S3 bucket for storing data feeds
 - AWS KMS key for encrypting the Amazon S3 bucket
 - (Optional) Amazon SNS topic for receiving notifications when AWS delivers new data to the Amazon S3 bucket
6. On the **Set up customer data storage** page, choose **Submit**.
7. (Optional) Edit the policies created by the CloudFormation template. See [Data feed policies \(p. 301\)](#) for more details.

You are now subscribed to data feeds. The next time data feeds are generated, you can access the data.

For more information about AWS CloudFormation templates, see [Working with AWS CloudFormation templates](#) in the *AWS CloudFormation User Guide*.

Data feed policies

When your Amazon S3 bucket is created by the CloudFormation template, it will create policies for access attached to that bucket, the AWS KMS key, and the Amazon SNS topic. The policies allow the AWS

Marketplace reports service to write to your bucket and SNS topic with the data feed information. Each policy will have a section like the following (this example is from the Amazon S3 bucket).

```
{
  "Sid": "AwsMarketplaceDataFeedsAccess",
  "Effect": "Allow",
  "Principal": {
    "Service": "reports.marketplace.amazonaws.com"
  },
  "Action": [
    "s3:ListBucket",
    "s3:GetObject",
    "s3:PutObject",
    "s3:GetEncryptionConfiguration",
    "s3:GetBucketAcl",
    "s3:PutObjectAcl"
  ],
  "Resource": [
    "arn:aws:s3:::datafeed-bucket",
    "arn:aws:s3:::datafeed-bucket/*"
  ]
},
```

In this policy, `reports.marketplace.amazonaws.com` is the service principal that AWS Marketplace uses to push data to the Amazon S3 bucket. The `datafeed-bucket` is the bucket that you specified in the CloudFormation template.

When the AWS Marketplace reports service calls Amazon S3, AWS KMS, or Amazon SNS, it will provide the ARN of the data it is intending to write to the bucket when it does. To ensure that the only data written to your bucket is data written on your behalf, you can specify the `aws:SourceArn` in the condition of the policy. In the following example, you must replace the `account-id` with the ID for your AWS account.

```
{
  "Sid": "AwsMarketplaceDataFeedsAccess",
  "Effect": "Allow",
  "Principal": {
    "Service": "reports.marketplace.amazonaws.com"
  },
  "Action": [
    "s3:ListBucket",
    "s3:GetObject",
    "s3:PutObject",
    "s3:GetEncryptionConfiguration",
    "s3:GetBucketAcl",
    "s3:PutObjectAcl"
  ],
  "Resource": [
    "arn:aws:s3:::datafeed-test-bucket",
    "arn:aws:s3:::datafeed-test-bucket/*"
  ],
  "Condition": {
    "StringEquals": {
      "aws:SourceAccount": "account-id",
      "aws:SourceArn": [
        "arn:aws:marketplace::account-id:AWSMarketplace/SellerDataSubscription/DataFeeds_V1",
        "arn:aws:marketplace::account-id:AWSMarketplace/SellerDataSubscription/Example-Report"
      ]
    }
  }
},
```

Using data feeds

When data is available in your Amazon S3 bucket, you can use data feeds in the following ways:

- Download the .CSV files from the Amazon S3 bucket you created in [Accessing data feeds \(p. 301\)](#) so that you can view the data in a spreadsheet.
- Use ETL (extract, transform, and load), SQL query, business analytics tools to collect and analyze the data.

You can use AWS services to collect and analyze data, or any third-party tool that can perform analysis of .CSV-based datasets.

Example: Use AWS services to collect and analyze data

The following procedure assumes that you've already configured your environment to receive data feeds to an Amazon S3 bucket and that the bucket contains data feeds.

To collect and analyze data from data feeds

1. From the [AWS Glue console](#), [create a crawler](#) to connect to the Amazon S3 bucket that stores the data feeds, extract the data you want, and create metadata tables in the AWS Glue Data Catalog.

For more information about AWS Glue, see the [AWS Glue Developer Guide](#).

2. From the [Athena console](#), [run SQL queries on the data in the AWS Glue Data Catalog](#).

For more information about Athena see the [Amazon Athena User Guide](#).

3. From the [Amazon QuickSight console](#), [create an analysis](#) and then [create a visual](#) of the data.

For more information about Amazon QuickSight, see the [Amazon QuickSight User Guide](#).

For a detailed example of one way to use AWS services to collect and analyze data in data feeds, see [Using Seller Data Feed Delivery Service, Amazon Athena, and Amazon QuickSight to create seller reports](#) at the AWS Marketplace Blog.

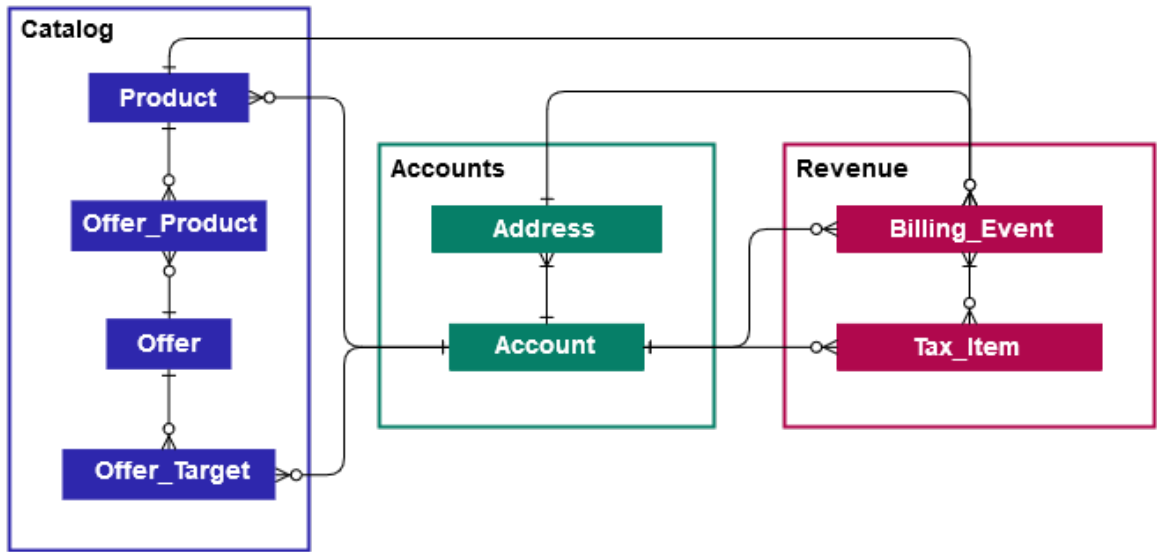
Data feed tables overview

The AWS Marketplace provided data feeds are a set of tables that you can join together to provide more context for your queries.

There are three general domains, or categories of interest, in your data feeds:

- **Catalog** – Includes information about the products and offers in your account.
- **Accounts** – Includes information about the accounts that provide or purchase products on AWS Marketplace (your own accounts or accounts of parties that you work with such as channel partners or buyers).
- **Revenue** – Includes information about billing, disbursements, and taxes.

The following diagram shows the tables in each domain, and how they are related to each other. This diagram shows the Catalog, Accounts, and Revenue domains, including the tables within them.



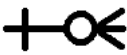
The following sections provide *entity relationship* (ER) diagrams for each domain. Each ER diagram shows the tables and the fields within each table, as well as the fields that you can use to join the tables.

Note

The ER diagrams in this section do not include the common fields for all data feeds. For more information about the common fields, see [Storage and structure of data feeds \(p. 299\)](#).



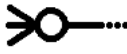
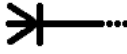
The following table describes the symbols that are used in the ER diagrams.

| Symbol | Description |
|-----------|--|
| PK | Primary key – A primary key for the table. When used with the <code>valid_from</code> and <code>update_date</code> fields, it is unique. For more details about using these fields together, see Historization of the data (p. 300) . If more than one field is marked as primary key, then the fields together form the primary key. |
| FK | Foreign key – A field that represents a primary key in a different table. Not necessarily unique in the table. Note In some cases, the foreign key can be blank if the record in the current table does not have a corresponding record in the foreign table. |
| AK | Alternate key – A key that can be used as a key in the table. Follows the same uniqueness rules as the primary key. |

| Symbol | Description |
|---|---|
|  | Connector – Lines between fields represent a connection, which is two fields that can be used to join tables. The ends of the line represent the type of connection. This example represents a one-to-many connection. |

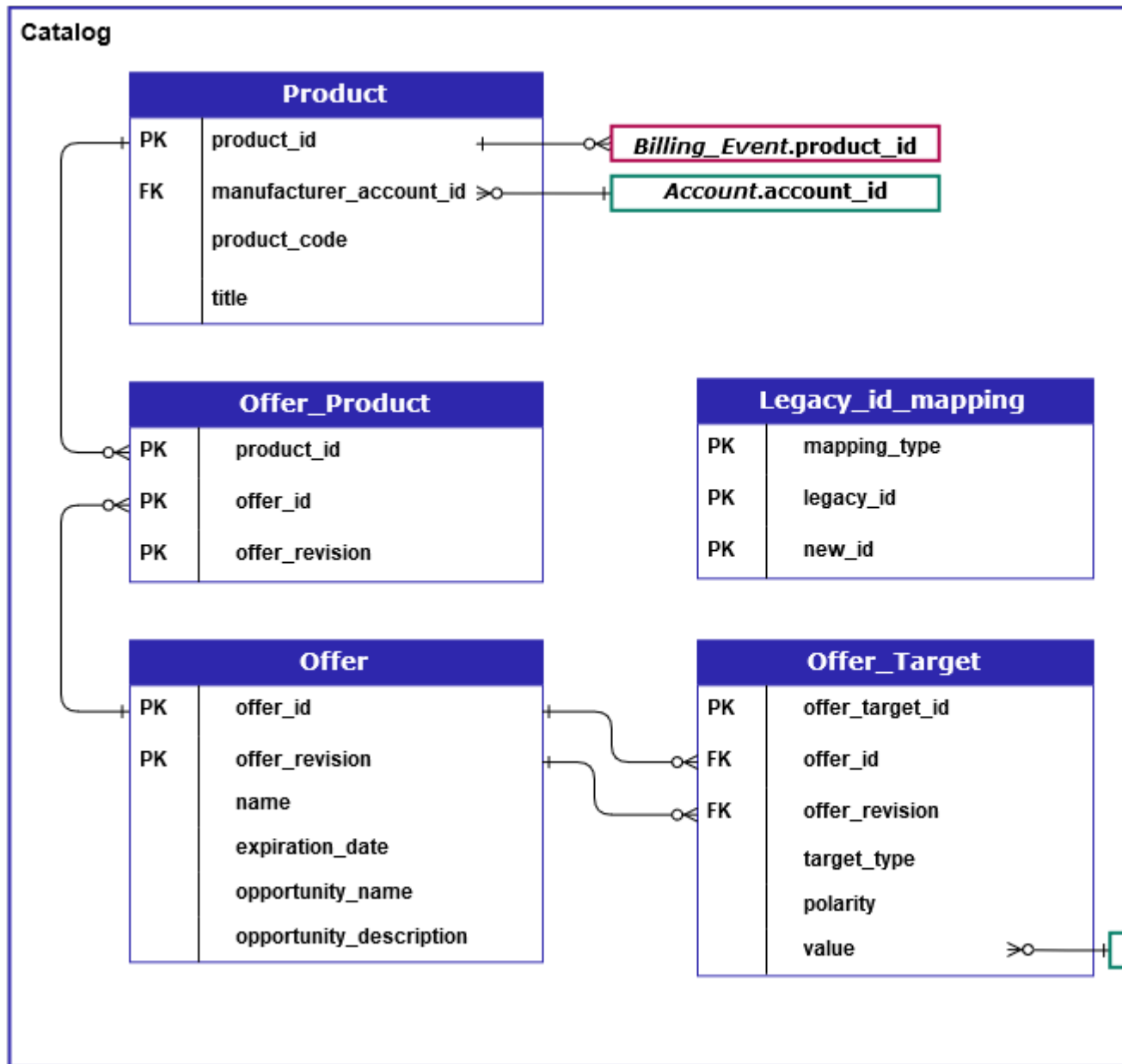
Connector types

The following table shows the types of ends that each connector can have.

| Connector type | Description |
|---|---|
|  | One to n – A connector with this end represents a join that has exactly one value on this side of the join. |
|  | Zero or one to n – A connector with this end represents a join that has zero or one values on this side of the join. |
|  | Zero or more to n – A connector with this end represents a join that has zero, one, or many values on this side of the join. |
|  | One or more to n – A connector with this end represents a join that has one or many values on this side of the join. |

Catalog-related tables

The following diagram shows the relationships between tables in the Catalog domain, as well as the fields within the tables.



The Product, Offer_Product, Offer, Offer_Target, and Legacy_id_mapping tables are in the Catalog domain.

The Offer_Target table includes a value field for the account_id of the target, but only when the target_type value is account.

The Legacy_id_mapping table is not used for current data.

Note

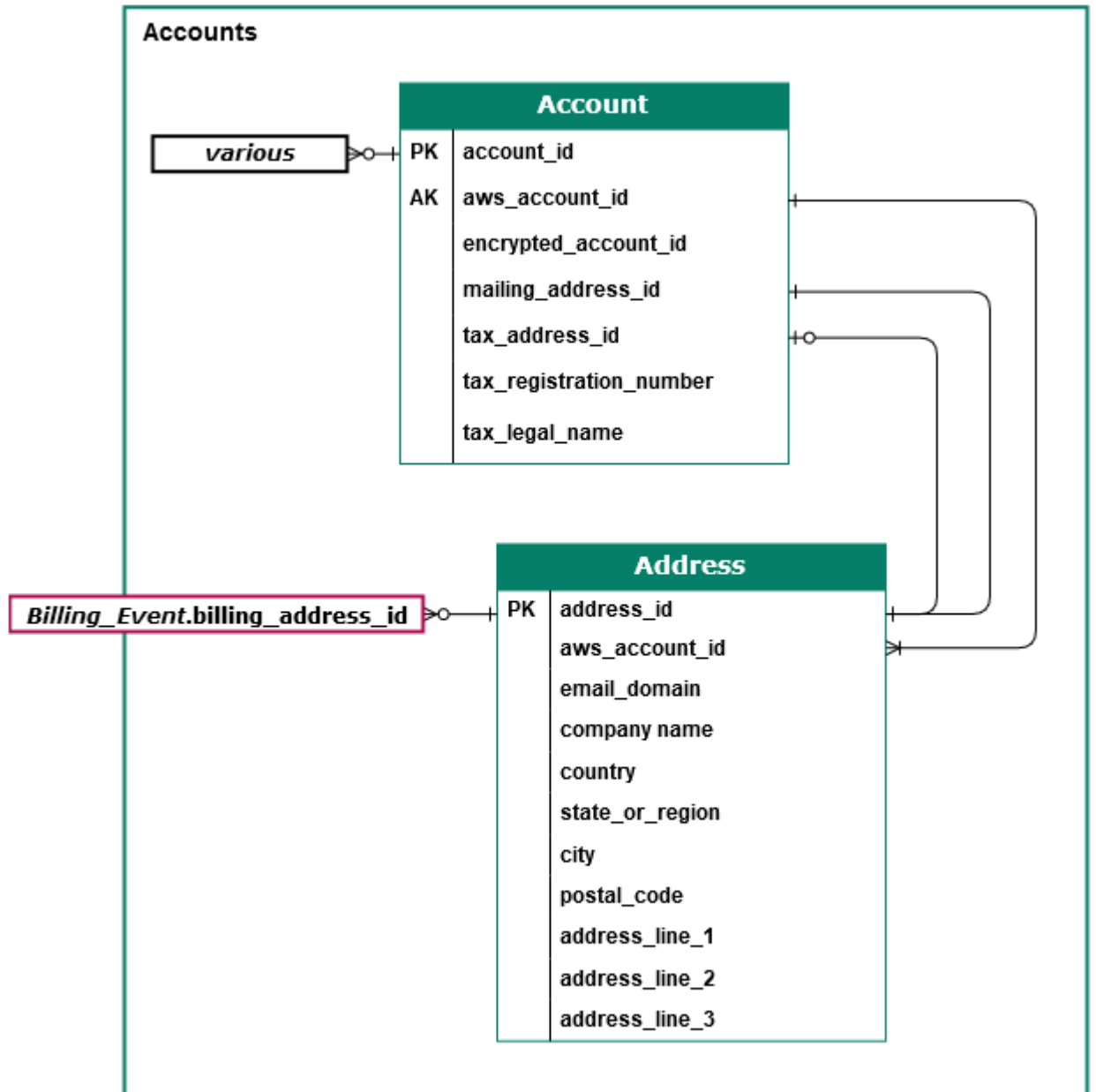
For more information about these tables, including a description of each field in the table and the joins that can be created, see the following topics:

- [Product data feed \(p. 335\)](#)
- [Offer product data feed \(p. 333\)](#)

- Offer data feed (p. 332)
- Offer target data feed (p. 334)
- Legacy mapping data feed (p. 332)

Accounts-related tables

The following diagram shows the relationships between the `Account` and `Address` tables in the Accounts domain, as well as the fields within the tables.



Note

For more information about these tables, including a description of each field in the table and the joins that can be created, see the following topics:

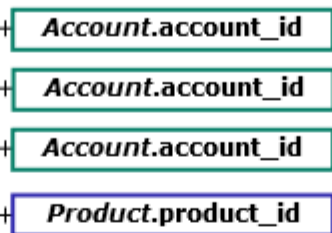
- [Account data feed \(p. 320\)](#)
- [Address data feed \(p. 321\)](#)

Revenue-related tables

The following diagram shows the relationships between the `Billing_Event` and `Tax_Item` tables in the Revenue domain, as well as the fields within the tables. The `Billing_Event` table includes information about disbursements, as well as billing events.

Revenue

| Billing_Event | |
|---------------|-------------------------------|
| PK | billing_event_id |
| FK | from_account_id |
| FK | to_account_id |
| FK | end_user_account_id |
| FK | product_id |
| | action |
| | transaction_type |
| | parent_billing_event_id |
| | disbursement_billing_event_id |
| | amount |
| | currency |
| | balance_impacting |
| | invoice_date |
| | transaction_date |
| | payment_due_date |
| | usage_period_start_date |
| | usage_period_end_date |
| | invoice_id |
| | billing_address_id |
| | transaction_reference_id |
| | bank_trace_id |



| Tax_Item | |
|----------|----------------------------------|
| PK | tax_item_id |
| FK | invoice_id |
| | line_item_id |
| | customer_bill_id |
| | tax_liable_party |
| | transaction_type_code |
| FK | product_id |
| | product_tax_code |
| | invoice_date |
| FK | taxed_customer_account_id |
| | taxed_customer_country |
| | taxed_customer_state_or_province |
| | taxed_customer_city |
| | taxed_customer_postal_code |
| | tax_location_code_taxed |
| | tax_type_code |
| | jurisdiction_level |
| | taxed_jurisdiction |
| | display_price_taxability_type |
| | taxable_amount |
| | non_taxable_amount |
| | tax_amount |
| | tax_jurisdiction_rate |
| | tax_amount |
| | tax_currency |
| | tax_calculation_reason_code |
| | date used for tax calculation |

Note

For more information about these tables, including a description of each field in the table and the joins that can be created, see the following topics:

- [Billing event data feed \(p. 323\)](#)
- [Tax item data feed \(p. 336\)](#)

Data feed query examples

This section gives examples of complex queries using the data feeds provided by AWS Marketplace. These examples are similar to the [Seller reports \(p. 273\)](#) that you get from the AWS Marketplace Management Portal. You can customize these queries to create other reports that you need.

Example 1: Disbursements by product

To find out the amount that's been disbursed by product, you can run a query like the following. This example is comparable to the [Disbursement report \(p. 283\)](#) that you can get as a seller report. However, you can use this sample to build your own queries and customize it to get exactly the report that you need.

This set of sample queries build upon each other to create the final list of product details with disbursements. It also shows how to get the product information at a specific point in time. The comments in the queries explain what the queries are doing, as well as how you can modify them to get different views of the data.

Note

When running this query, we are assuming that the data ingested is using two time axes (the `valid_from` column and the `update_date` column). For more details, see [Storage and structure of data feeds \(p. 299\)](#).

```
-- Get all the products and keep the latest product_id, valid_from tuple
with products_with_uni_temporal_data as (
  select
    *
  from
    (
      select
        *,
        ROW_NUMBER() OVER (PARTITION BY product_id, valid_from
          ORDER BY from_iso8601_timestamp(update_date) desc)
          as row_num
      from
        productfeed_v1
    )
  where
    -- A product_id can appear multiple times with the same
    -- valid_from date but with a different update_date column,
    -- making it effectively bi-temporal. By only taking the most
    -- recent tuple, we are converting to a uni-temporal model.
    row_num = 1
),

-- Gets the latest revision of a product
-- A product can have multiple revisions where some of the
-- columns, like the title, can change.
-- For the purpose of the disbursement report, we want
-- to get the latest revision of a product
products_with_latest_version as (
  select
    *
```

```
from
(
  select
    *,
    ROW_NUMBER() OVER (PARTITION BY product_id
      ORDER BY from_iso8601_timestamp(valid_from) desc)
      as row_num_latest_version
  from
    products_with_uni_temporal_data
)
where
  row_num_latest_version = 1
),

-- Get all the accounts and keep the latest account_id, valid_from tuple
accounts_with_uni_temporal_data as (
  select
    *
  from
    (
      select
        *,
        ROW_NUMBER() OVER (PARTITION BY account_id, valid_from ORDER BY
from_iso8601_timestamp(update_date) desc) as row_num
      from
        accountfeed_v1
    )
  where
    -- An account_id can appear multiple times with the same
    -- valid_from date but with a different update_date column,
    -- making it effectively bi-temporal. By only taking the most
    -- recent tuple, we are converting to a uni-temporal model.
    row_num = 1
),

-- Gets the latest revision of an account
-- An account can have multiple revisions where some of the
-- columns, like the mailing_address_id, can change.
-- For the purpose of the disbursement report, we want
-- to get the latest revision of a product
accounts_with_latest_version as (
  select
    *
  from
    (
      select
        *,
        ROW_NUMBER() OVER (PARTITION BY account_id
      ORDER BY from_iso8601_timestamp(valid_from) desc)
      as row_num_latest_version
    from
      accounts_with_uni_temporal_data
    )
  where
    row_num_latest_version = 1
),

-- Get all the billing events and keep the
-- latest billing_event_id, valid_from tuple:
billing_events_with_uni_temporal_data as (
  select
    *
  from (
    select
      billing_event_id,
      from_iso8601_timestamp(valid_from) as valid_from,
```



```
from_iso8601_timestamp(update_date) as update_date,
from_iso8601_timestamp(invoice_date) as invoice_date,
transaction_type,
transaction_reference_id,
product_id,
disbursement_billing_event_id,
action,
from_account_id,
to_account_id,
end_user_account_id,
CAST(amount as decimal(20, 10)) invoice_amount,
bank_trace_id,
ROW_NUMBER() OVER (PARTITION BY billing_event_id, valid_from
ORDER BY from_iso8601_timestamp(update_date) desc)
as row_num
from
    billingeventfeed_v1
)
where row_num = 1
),

-- Get all the disbursements
-- The billing events data is immutable.
-- It is not required to use time windows based on the
-- valid_from column to get the most recent billing event
disbursement_events as (
    select
        billing_events_raw.billing_event_id as disbursement_id,
        billing_events_raw.invoice_date as disbursement_date,
        billing_events_raw.bank_trace_id
    from
        billing_events_with_uni_temporal_data billing_events_raw
    where
        -- Only interested in disbursements, so filter out
        -- non-disbursements by selecting transaction type
        -- to be DISBURSEMENT:
        billing_events_raw.transaction_type = 'DISBURSEMENT'
        -- Select a time period, you can adjust the dates
        -- below if need be. For billing events use the
        -- invoice date as the point in time of the
        -- disbursement being initiated:
        and billing_events_raw.invoice_date >=
            from_iso8601_timestamp('2020-10-01T00:00:00Z')
        and billing_events_raw.invoice_date <
            from_iso8601_timestamp('2020-11-01T00:00:00Z')
),

-- Get the invoices along with the line items that
-- are part of the above filtered disbursements
disbursed_line_items as (
    select
        line_items.transaction_reference_id,
        line_items.product_id,
        line_items.transaction_type,
        (case
            -- Get the payer of the invoice from any
            -- transaction type that is not AWS and
            -- not BALANCE_ADJUSTMENT.
            -- For AWS and BALANCE_ADJUSTMENT, the billing
            -- event feed will show the "AWS Marketplace"
            -- account as the receiver of the funds and the
            -- seller as the payer. Filter those out.
            when line_items.transaction_type
                not like '%AWS%' and transaction_type
                not like 'BALANCE_ADJUSTMENT'
            then line_items.from_account_id
```

```

end) as payer_account_id,
line_items.end_user_account_id,
invoice_amount,
disbursements.disbursement_date,
disbursements.disbursement_id,
disbursements.bank_trace_id
from
  billing_events_with_uni_temporal_data line_items
  -- Each disbursed line item is linked to the parent
  -- disbursement via the disbursement_billing_event_id
  join disbursement_events disbursements
    on disbursements.disbursement_id
    = line_items.disbursement_billing_event_id
where
  -- we are interested only in the invoice line
  -- items that are DISBURSED
  line_items.action = 'DISBURSED'
),

-- An invoice can contain multiple line items
-- Create a pivot table to calculate the different
-- amounts that are part of an invoice.
-- The new row is aggregated at
-- transaction_reference_id - end_user_account_id level
invoice_amounts_aggregated as (
  select
    transaction_reference_id,
    product_id,
    -- a given disbursement id should have the
    -- same disbursement_date
    max(disbursement_date) as disbursement_date,
    -- Build a pivot table in order to provide all the
    -- data related to a transaction in a single row.
    -- Note that the amounts are negated. This is because
    -- when an invoice is generated, we give you the
    -- positive amounts and the disbursement event
    -- negates the amounts
    sum(case when transaction_type = 'SELLER_REV_SHARE'
      then -invoice_amount else 0 end) as seller_rev_share,
    sum(case when transaction_type = 'AWS_REV_SHARE'
      then -invoice_amount else 0 end) as aws_rev_share,
    sum(case when transaction_type = 'SELLER_REV_SHARE_REFUND'
      then -invoice_amount else 0 end) as seller_rev_refund,
    sum(case when transaction_type = 'AWS_REV_SHARE_REFUND'
      then -invoice_amount else 0 end) as aws_rev_refund,
    sum(case when transaction_type = 'SELLER_REV_SHARE_CREDIT'
      then -invoice_amount else 0 end) as seller_rev_credit,
    sum(case when transaction_type = 'AWS_REV_SHARE_CREDIT'
      then -invoice_amount else 0 end) as aws_rev_credit,
    sum(case when transaction_type = 'SELLER_TAX_SHARE'
      then -invoice_amount else 0 end) as seller_tax_share,
    sum(case when transaction_type = 'SELLER_TAX_SHARE_REFUND'
      then -invoice_amount else 0 end) as seller_tax_refund,
    -- This is the account that pays the invoice:
    max(payer_account_id) as payer_account_id,
    -- This is the account that subscribed to the product:
    end_user_account_id as customer_account_id,
    bank_trace_id
  from
    disbursed_line_items
  group by
    transaction_reference_id,
    product_id,
    disbursement_id,
    -- There might be a different end-user for the same
    -- transaction reference id. Distributed licenses

```

```

        -- is an example
        end_user_account_id,
        bank_trace_id
    ),
disbursed_amount_by_product as (
    select
        products.title as ProductTitle,
        products.product_code as ProductCode,
        -- We are rounding the sums using 2 decimal precision
        -- Note that the rounding method might differ
        -- between SQL implementations.
        -- The disbursement seller report is using
        -- RoundingMode.HALF_UP. This might create
        -- discrepancies between this SQL output
        -- and the disbursement seller report
        round(invoice_amounts.seller_rev_share, 2) as SellerRev,
        round(invoice_amounts.aws_rev_share, 2) as AWSRefFee,
        round(invoice_amounts.seller_rev_refund, 2) as SellerRevRefund,
        round(invoice_amounts.aws_rev_refund, 2) as AWSRefFeeRefund,
        round(invoice_amounts.seller_rev_credit, 2) as SellerRevCredit,
        round(invoice_amounts.aws_rev_credit, 2) as AWSRefFeeCredit,
        (
            round(invoice_amounts.seller_rev_share, 2) +
            round(invoice_amounts.aws_rev_share, 2) +
            round(invoice_amounts.seller_rev_refund, 2) +
            round(invoice_amounts.aws_rev_refund, 2) +
            round(invoice_amounts.seller_rev_credit, 2) +
            round(invoice_amounts.aws_rev_credit, 2)
        ) as NetAmount,
        invoice_amounts.transaction_reference_id
            as TransactionReferenceID,
        round(invoice_amounts.seller_tax_share, 2)
            as SellerSalesTax,
        round(invoice_amounts.seller_tax_refund, 2)
            as SellerSalesTaxRefund,
        payer_info.aws_account_id
            as PayerAwsAccountId,
        customer_info.aws_account_id
            as EndCustomerAwsAccountId,
        invoice_amounts.disbursement_date
            as DisbursementDate,
        invoice_amounts.bank_trace_id
            as BankTraceId
    from
        invoice_amounts_aggregated invoice_amounts
        join products_with_latest_version products
            on products.product_id = invoice_amounts.product_id
        left join accounts_with_latest_version payer_info
            on payer_info.account_id = invoice_amounts.payer_account_id
        left join accounts_with_latest_version customer_info
            on customer_info.account_id = invoice_amounts.customer_account_id
    )
select * from disbursed_amount_by_product;

```

Example 2: Sales compensation report

To find the billed revenue by customer, you can run a query like the following. This example is comparable to the [Sales compensation report \(p. 294\)](#) that you can get as a seller report. However, you can use this sample to build your own queries and customize it to get exactly the report that you need.

This is a set of sample queries that build upon each other to create the final list of customer details with the total amount billed to each customer for usage of your software. The comments in the queries

explain what the queries are doing, as well as how you can modify them to get different views of the data.

Note

When running this query, we are assuming that the data ingested is using two time axes (the `valid_from` column and the `update` column). For more details, see [Storage and structure of data feeds \(p. 299\)](#).

```
-- Gets all the products and keeps the latest product_id,
-- valid_from tuple.
with products_with_uni_temporal_data as (
  select
    *
  from
    (
      select
        *,
        ROW_NUMBER() OVER (PARTITION BY product_id, valid_from
                           ORDER BY from_iso8601_timestamp(update_date) desc)
        as row_num
      from
        productfeed_v1
    )
  where
    -- A product_id can appear multiple times with the same
    -- valid_from date but with a different update_date column,
    -- making it effectively bi-temporal. By only taking the most
    -- recent tuple, we are converting to a uni-temporal model.
    row_num = 1
),

-- Gets the latest revision of a product
-- A product can have multiple revisions where some of the
-- columns, like the title, can change.
-- For the purpose of the sales compensation report, we want
-- to get the latest revision of a product
products_with_latest_revision as (
  select
    *
  from
    (
      select
        *,
        ROW_NUMBER() OVER (PARTITION BY product_id ORDER BY
                           from_iso8601_timestamp(valid_from) desc) as row_num_latest_revision
      from
        products_with_uni_temporal_data
    )
  where
    row_num_latest_revision = 1
),

-- Gets all the addresses and keeps the latest address_id,
-- aws_account_id, and valid_from combination.
-- We're transitioning from a bi-temporal data model to an
-- uni-temporal data_model
piifeed_with_uni_temporal_data as (
  select
    *
  from
    (
      select
        *,
        ROW_NUMBER() OVER (
          PARTITION BY address_id, aws_account_id, valid_from
```

```
        ORDER BY from_iso8601_timestamp(update_date) desc)
        as row_num
    from
        piifeed
    )
    where
        -- An address_id can appear multiple times with the same
        -- valid_from date but with a different update_date column.
        -- We are only interested in the most recent.
        row_num = 1
    ),

-- Gets the latest revision of an address.
-- An address_id can have multiple revisions where some of
-- the columns can change.
-- For the purpose of the sales compensation report, we want to
-- get the latest revision of an address + account_id pair.
pii_with_latest_revision as (
    select
        *
    from
        (
            select
                *,
                ROW_NUMBER() OVER (PARTITION BY address_id, aws_account_id
                    ORDER BY from_iso8601_timestamp(valid_from) desc)
                    as row_num_latest_revision
            from
                piifeed_with_uni_temporal_data
        )
    where
        row_num_latest_revision = 1
),

-- Gets all the accounts and keeps the latest
-- account_id, valid_from tuple.
-- We're transitioning from a bi-temporal data
-- model to an uni-temporal data_model.
accounts_with_uni_temporal_data as (
    select
        *
    from
        (
            select
                *,
                ROW_NUMBER() OVER (PARTITION BY account_id, valid_from
                    ORDER BY from_iso8601_timestamp(update_date) desc)
                    as row_num
            from
                accountfeed_v1
        )
    where
        -- An account_id can appear multiple times with the same
        -- valid_from date but with a different update_date column.
        -- We are only interested in the most recent tuple.
        row_num = 1
),

-- Gets all the historical dates for an account
-- An account can have multiple revisions where some of the
-- columns like the mailing_address_id can change.
accounts_with_history as (
    select
        *,
        -- This interval's begin_date
        case
```

```

when
-- First record for a given account_id
  lag(valid_from, 1) over (partition by account_id
    order by from_iso8601_timestamp(valid_from) asc) is null
then
-- 'force' begin_date a bit earlier because of different
-- data propagation times. We'll subtract one day as one
-- hour is not sufficient
  from_iso8601_timestamp(valid_from) - INTERVAL '1' DAY
else
-- not the first line -> return the real date
  from_iso8601_timestamp(valid_from)
end as begin_date,
-- This interval's end date.
COALESCE(
  LEAD(from_iso8601_timestamp(valid_from), 1)
    OVER (partition by account_id
      ORDER BY from_iso8601_timestamp(valid_from)),
  from_iso8601_timestamp('9999-01-01T00:00:00Z')
) as end_date
from
  accounts_with_uni_temporal_data
),

-- Gets all the billing events and keeps the latest
-- billing_event_id, valid_from tuple.
-- We're transitioning from a bi-temporal data
-- model to an uni-temporal data_model.
billing_events_with_uni_temporal_data as (
  select
    *
  from (
    select
      billing_event_id,
      from_iso8601_timestamp(valid_from) as valid_from,
      from_iso8601_timestamp(update_date) as update_date,
      from_iso8601_timestamp(invoice_date) as invoice_date,
      transaction_type,
      transaction_reference_id,
      product_id,
      disbursement_billing_event_id,
      action,
      currency,
      from_account_id,
      to_account_id,
      end_user_account_id,
      -- convert an empty billing address to null. This will
      -- later be used in a COALESCE call
      case
        when billing_address_id <> '' then billing_address_id else null
      end as billing_address_id,
      CAST(amount as decimal(20, 10)) invoice_amount,
      ROW_NUMBER() OVER (PARTITION BY billing_event_id, valid_from
        ORDER BY from_iso8601_timestamp(update_date) desc)
        as row_num
    from
      billingeventfeed_v1
    where
      -- The Sales Compensation Report does not contain BALANCE
      -- ADJUSTMENTS, so we filter them out here
      transaction_type <> 'BALANCE_ADJUSTMENT'
      -- Keep only the transactions that will affect any
      -- future disbursed amounts.
      and balance_impacting = '1'
  )
  where row_num = 1

```

```
),  
  
-- Gets the billing address for all DISBURSED invoices. This  
-- will be the address of the payer when the invoice was paid.  
-- NOTE: For legal reasons, for CPPO transactions, the  
-- manufacturer will not see the payer's billing address id  
billing_addresses_for_disbursed_invoices as (  
  select  
    billing_events_raw.transaction_reference_id,  
    billing_events_raw.billing_address_id,  
    billing_events_raw.from_account_id  
  from  
    billing_events_with_uni_temporal_data billing_events_raw  
  where  
    -- the disbursed items will contain the billing address id  
    billing_events_raw.action = 'DISBURSED'  
    -- we only want to get the billing address id for the  
    -- transaction line items where the seller is the receiver  
    -- of the amount  
    and billing_events_raw.transaction_type like 'SELLER_%'  
  group by  
    billing_events_raw.transaction_reference_id,  
    billing_events_raw.billing_address_id,  
    billing_events_raw.from_account_id  
),  
  
-- An invoice can contain multiple line items.  
-- We create a pivot table to calculate the different amounts  
-- that are part of an invoice.  
-- The new row is aggregated at  
-- transaction_reference_id - end_user_account_id level  
invoiced_and_forgiven_transactions as (  
  select  
    transaction_reference_id,  
    product_id,  
    -- A transaction will have the same invoice date for all  
    -- of its line items (transaction types)  
    max(invoice_date) as invoice_date,  
    -- A transaction will have the same billing_address_id  
    -- for all of its line items. Remember that the billing event  
    -- is uni temporal and we retrieved only the latest valid_from item  
    max(billing_address_id) as billing_address_id,  
    -- A transaction will have the same currency for all  
    -- of its line items  
    max(currency) as currency,  
    -- We're building a pivot table in order to provide all the  
    -- data related to a transaction in a single row  
    sum(case when transaction_type = 'SELLER_REV_SHARE'  
           then invoice_amount else 0 end) as seller_rev_share,  
    sum(case when transaction_type = 'AWS_REV_SHARE'  
           then invoice_amount else 0 end) as aws_rev_share,  
    sum(case when transaction_type = 'SELLER_REV_SHARE_REFUND'  
           then invoice_amount else 0 end) as seller_rev_refund,  
    sum(case when transaction_type = 'AWS_REV_SHARE_REFUND'  
           then invoice_amount else 0 end) as aws_rev_refund,  
    sum(case when transaction_type = 'SELLER_REV_SHARE_CREDIT'  
           then invoice_amount else 0 end) as seller_rev_credit,  
    sum(case when transaction_type = 'AWS_REV_SHARE_CREDIT'  
           then invoice_amount else 0 end) as aws_rev_credit,  
    sum(case when transaction_type = 'SELLER_TAX_SHARE'  
           then invoice_amount else 0 end) as seller_tax_share,  
    sum(case when transaction_type = 'SELLER_TAX_SHARE_REFUND'  
           then invoice_amount else 0 end) as seller_tax_refund,  
    -- this is the account that pays the invoice.  
    max(case  
      -- Get the payer of the invoice from any transaction type
```

AWS Marketplace Seller Guide
Data feed query examples

```
-- that is not AWS and not BALANCE_ADJUSTMENT.
-- For AWS and BALANCE_ADJUSTMENT, the billing event feed
-- will show the "AWS Marketplace" account as the
-- receiver of the funds and the seller as the payer. We
-- are not interested in this information here.
when
  transaction_type not like '%AWS%'
  and transaction_type not like 'BALANCE_ADJUSTMENT'
  then from_account_id
end) as payer_account_id,
-- this is the account that subscribed to your product
end_user_account_id as customer_account_id
from
  billing_events_with_uni_temporal_data
where
  -- Get invoiced or forgiven items. Disbursements are
  -- not part of the sales compensation report
  action in ('INVOICED', 'FORGIVEN')
group by
  transaction_reference_id,
  product_id,
  -- There might be a different end-user for the same
  -- transaction reference id. Distributed licenses
  -- is an example.
  end_user_account_id
),
invoiced_items_with_product_and_billing_address as (
  select
    invoice_amounts.*,
    products.product_code,
    products.title,
    payer_info.aws_account_id as payer_aws_account_id,
    payer_info.account_id as payer_reference_id,
    customer_info.aws_account_id as end_user_aws_account_id,
    (
      invoice_amounts.seller_rev_share +
      invoice_amounts.aws_rev_share +
      invoice_amounts.seller_rev_refund +
      invoice_amounts.aws_rev_refund +
      invoice_amounts.seller_rev_credit +
      invoice_amounts.aws_rev_credit +
      invoice_amounts.seller_tax_share +
      invoice_amounts.seller_tax_refund
    ) as seller_net_revenue,
    -- Try to get the billing address from the DISBURSED event
    -- (if any). If there is no DISBURSEMENT, get the billing
    -- address from the INVOICED item. If still no billing address,
    -- then default to getting the mailing address of the payer.
    coalesce(billing_add.billing_address_id,
             invoice_amounts.billing_address_id,
             payer_info.mailing_address_id)
             as final_billing_address_id
  from
    invoiced_and_forgiven_transactions invoice_amounts
  join products_with_latest_revision products
    on products.product_id = invoice_amounts.product_id
  left join accounts_with_history payer_info
    on payer_info.account_id = invoice_amounts.payer_account_id
    -- Get the Payer Information at the time of invoice creation
    and payer_info.begin_date <= invoice_amounts.invoice_date
    and invoice_amounts.invoice_date < payer_info.end_date
  left join accounts_with_history customer_info
    on customer_info.account_id = invoice_amounts.customer_account_id
    -- Get the End User Information at the time of invoice creation
    and customer_info.begin_date <= invoice_amounts.invoice_date
```



```

        and invoice_amounts.invoice_date < customer_info.end_date
    left join billing_addresses_for_disbursed_invoices billing_add
        on billing_add.transaction_reference_id =
            invoice_amounts.transaction_reference_id
        and billing_add.from_account_id =
            invoice_amounts.payer_account_id
    ),
invoices_with_full_address as (
    select
        payer_aws_account_id as "Customer AWS Account Number",
        pii_data.country as "Country",
        pii_data.state_or_region as "State",
        pii_data.city as "City",
        pii_data.postal_code as "Zip Code",
        pii_data.email_domain as "Email Domain",
        product_code as "Product Code",
        title as "Product Title",
        seller_rev_share as "Gross Revenue",
        aws_rev_share as "AWS Revenue Share",
        seller_rev_refund as "Gross Refunds",
        aws_rev_refund as "AWS Refunds Share",
        seller_net_revenue as "Net Revenue",
        currency as "Currency",
        date_format(invoice_date, '%Y-%m') as "AR Period",
        transaction_reference_id as "Transaction Reference ID",
        payer_reference_id as "Payer Reference ID",
        end_user_aws_account_id as "End Customer AWS Account ID"
    from
        invoiced_items_with_product_and_billing_address invoice_amounts
        left join pii_with_latest_revision pii_data
            on pii_data.aws_account_id = invoice_amounts.payer_aws_account_id
            and pii_data.address_id = invoice_amounts.final_billing_address_id
        -- Filter out FORGIVEN and Field Demonstration Pricing transactions
        where seller_net_revenue <> 0
    )
select * from invoices_with_full_address;

```

Account data feed

This data feed provides information about all the accounts you interact with: your own, any channel partners you work with, buyers, payers, and all taxed accounts.

Account data is immutable, and it is not associated with a version number. Changes to fields are appended, so this data feed may have several rows with the same `account_id` and different `valid_from` values. For information about data history fields, see [Historization of the data \(p. 300\)](#).

The account data feed is refreshed every 24 hours, so new data is available daily.

The following table explains the names and descriptions of the data feed's columns.

| Column name | Description |
|-------------------------|---|
| <code>account_id</code> | <p>The globally unique identifier (GUID) of the account.</p> <p>Can be used to join to fields in the <code>Product</code>, <code>Offer_Target</code>, <code>Billing_Event</code>, and <code>Tax_Item</code> data feeds. See those data feeds for information about the fields that can be used to join.</p> |

| Column name | Description |
|-------------------------|--|
| aws_account_id | The AWS account number of the seller's AWS account, which is unique by AWS partition. |
| encrypted_account_id | The unique, encrypted ID for an individual buyer of your application. The value for encrypted_account_id is used by the AWS Marketplace Metering Service, for example, as the value for CustomerIdentifier that is returned by the ResolveCustomer action. |
| mailing_address_id | The mailing address reference for this account. |
| tax_address_id | The tax address reference for this account. |
| tax_registration_number | For non-US accounts, the tax registration number for this account. |
| tax_legal_name | For non-US accounts, the legal company name. This is the name used on tax invoices. |

Example of account data feed

The following shows an example of the account data feed. For readability, the data history columns aren't shown. For information about data history fields, see [Historization of the data \(p. 300\)](#).

| account_id | aws_account_id | encrypted_account_id | mailing_address_id | tax_address_id | tax_registration_number | tax_legal_name |
|---------------|----------------|----------------------|---|----------------|-------------------------|----------------|
| xk0CSmiAm6P | 0404E5690007 | zP1kNf6Wp2503Z | 46eN6eViOfiiqtwX8e3kaOiPalUiofjyFa3 | | | |
| 7nyo5jwTRoPly | 555676799907 | z01Qf8pQ88W3 | TjSzMrrF2gV62VjS4MqF83829J5H6pT850M2sBhmyFY5Lr42s8 | | | |
| VleGa2t9j3Mux | 737900888X | SjGAGuarev | ILM5UeiMlGFVDNMBYBv6DpP401558G1E0fM8Dj7uoTC2j7Pm8ZKKN | | | |

Address data feed

This data feed provides contact information for all the accounts you interact with: your own, any channel partners you work with, buyers, payers, and all taxed accounts. Each time a new transaction occurs, the customer address for the transaction is scanned, and if it's not in your data feed, a new entry is added to your data feed file.

Address data is immutable.

The address data feed is refreshed every 24 hours, so new data is available daily.

The following table explains the names and descriptions of the data feed's columns.

| Column name | Description |
|-------------|--|
| address_id | The unique key of the address. Can be used to join from the Billing_Event data feed on the billing_address_id field, or from the Account data feed on the |

| Column name | Description |
|-----------------|--|
| | mailing_address_id or tax_address_id fields. |
| aws_account_id | The AWS account number of this address. Can be used to join to the Account data feed on the aws_account_id field. |
| email_domain | The domain for the email address on file for this account. |
| company_name | The company name on file for this account. |
| country | The ISO 3166 alpha-2 country code on file for this address. |
| state_or_region | The state or region on file for this address. |
| city | The city on file for this address. |
| postal_code | The postal code on file for this address. |
| address_line_1 | The first line of the address on file for this address. |
| address_line_2 | The second line of the address on file for this address. |
| address_line_3 | The third line of the address on file for this address. |

Example of address data feed

The following shows an example of the address data feed. In the data feed, this information is presented in a single table. For readability, the data is shown in two tables here, and the data history columns aren't shown. For information about data history fields, see [Historization of the data \(p. 300\)](#).

| address_id | aws_account_id | email_domain | company_name | country | state_or_region | city | postal_code |
|----------------------------|-----------------|--------------|---------------------------------|---------|-----------------|----------|-------------|
| V5NhBYBiYog442V66600014AFm | 442V66600014AFm | gmgm.com | MTZj7Pm8ZK Jackson's Company | DE | | Hamburg | 67568 |
| G68xdbkZQD555567679955555 | 555567679955555 | syn.com | 0A6NiSMWmH5ViLJ Major's Company | US | OH | Dayton | 57684 |
| NLUc5UeiMIG55567679955555 | 555567679955555 | syn.com | 0A6NiSMWmH5ViLJ Major's Company | US | NY | New York | 89475 |

| address_line_1 | address_line_2 | address_line_3 |
|----------------|----------------|----------------|
| | | |
| | | |

| address_line_1 | address_line_2 | address_line_3 |
|----------------|----------------|----------------|
| | 19th Floor | |

Billing event data feed

This data feed provides information about billing events, including invoicing and disbursements.

For example, you can use this data feed to learn when and what a buyer is invoiced. You can also use the [example SQL queries \(p. 329\)](#) to analyze the data from this data feed.

This data feed contains information associated with billing events for which you are the seller of record. For agreements made via channel partners, this data feed contains information about billing events between the manufacturer and seller of record.

The billing event data feed is refreshed every 24 hours, so new data is available daily.

Billing event data is immutable.

The following table explains the names and descriptions of the data feed's columns.

| Column name | Description |
|---------------------|--|
| billing_event_id | An identifier for a billing event. This ID is unique in the seller's environment. |
| from_account_id | The account that initiated the billing event. If <code>transaction_type</code> is <code>SELLER_REV_SHARE</code> , it is the buyer's payer account. This is a foreign key to the account (p. 320) data feed. Can be used to join to the <code>Account</code> data feed on the <code>account_id</code> field. |
| to_account_id | The account that receives the transaction amount for the product. This is a foreign key to the <code>account</code> data feed. Can be used to join to the <code>Account</code> data feed on the <code>account_id</code> field. |
| end_user_account_id | The account that uses the product. This account may be different from the buyer and payer accounts. Can be used to join to the <code>Account</code> data feed on the <code>account_id</code> field. |
| product_id | The identifier of the product. This is a foreign key to the product (p. 335) data feed. Can be used to join to the <code>Product</code> data feed on the <code>product_id</code> field. |
| action | The type of action for this event. Possible values are as follows: |

| Column name | Description |
|-------------------------|--|
| | <ul style="list-style-type: none"> • INVOICED – The buyer was invoiced for the amount. • FORGIVEN – The buyer was invoiced for the amount, and AWS reverted the charge. • DISBURSED – The seller was paid this amount. This can include a month of invoices, or be an on-demand disbursement. |
| transaction_type | <p>The type of transaction. For examples, see Taxing scenarios (p. 326). Possible values are as follows:</p> <ul style="list-style-type: none"> • SELLER_REV_SHARE – A positive amount; this is the price the seller set in the agreement with buyer. • SELLER_TAX_SHARE – A positive amount; this is the amount added to SELLER_REV_SHARE to cover taxes the seller owes. • AWS_REV_SHARE – A negative amount; this is the listing fee. • AWS_TAX_SHARE – A positive amount; this is the amount of taxes AWS collected in addition to SELLER_REV_SHARE. This amount doesn't affect the seller's balance. This amount is not disbursed and is provided for seller's awareness of taxes invoiced to buyer and remitted to authorities on the seller's behalf. • transaction_type_REFUND – The amount of refund requested by buyer. • transaction_type_CREDIT – The amount AWS credits the buyer. • BALANCE_ADJUSTMENT – An adjustment made by AWS to resolve invoicing issues. • DISBURSEMENT – If <code>action = DISBURSEMENT</code> and <code>balancing_impacting = 1</code>, this is the amount paid to seller. If the value for <code>action</code> is <code>= INVOICED</code>, this record negates the <code>parent_billing_event_id</code> record either in full or in part. In this case, the related disbursement <code>disbursement_billing_event_id</code> is shown and <code>balancing_impacting = 0</code>. • DISBURSEMENT_FAILURE – Negates the transaction. |
| parent_billing_event_id | <p>If the <code>action</code> is DISBURSEMENT or FORGIVEN and the <code>transaction_type</code> is DISBURSEMENT, this is the <code>billing_event_id</code> that initiated this billing event. If <code>action</code> has another value, this field is null.</p> |

| Column name | Description |
|-------------------------------|---|
| disbursement_billing_event_id | <p>The related disbursement when the action is <code>DISBURSED</code> AND one of the following is true:</p> <ul style="list-style-type: none"> • <code>transaction_type</code> like ('SELLER%') • <code>transaction_type</code> like ('AWS%') <p>In all other cases, this value is null.</p> |
| amount | The billing event amount. |
| currency | The ISO 639 currency code. |
| balance_impacting | Whether the amount is taken into account in calculating seller disbursements. A value of 0 indicates the amount is shown for informational purposes and has no effect on the balance. A value of 1 indicates that this amount takes into account in determining seller disbursements. |
| invoice_date | The date the invoice was created. |
| payment_due_date | When the action is <code>INVOICED</code> , the due date for the invoice. |
| usage_period_start_date | The start date for the period in the record. |
| usage_period_end_date | The end date for the period in the record. |
| invoice_id | The AWS invoice ID. |
| billing_address_id | <p>The payer's billing address reference in the address data feed.</p> <p>Can be used to join to the <code>Address</code> data feed on the <code>address_id</code> field.</p> |
| transaction_reference_id | <p>An identifier that allows you to cross-reference data from the following reports:</p> <ul style="list-style-type: none"> • Disbursement report (p. 283) • Monthly billed revenue report (p. 288) • Sales compensation report (p. 294) • US sales and use tax report (p. 296) |
| bank_trace_id | For disbursement transactions (<code>transaction_type = DISBURSEMENT</code> and <code>action = DISBURSED</code>), the trace ID assigned by the bank. The trace ID can be used to correlate with bank-provided reports from the seller bank. |

| Column name | Description |
|-------------|--|
| broker_id | <p>An identifier of the business entity which facilitated the transaction. Possible values are as follows:</p> <ul style="list-style-type: none"> • AWS_INC <ul style="list-style-type: none"> – The identifier for AWS, Inc. (based in the United States). • AWS_EUROPE <ul style="list-style-type: none"> – The identifier for Amazon Web Services EMEA SARL (based in Luxembourg). • NULL <ul style="list-style-type: none"> – Previous transactions without an explicit broker_id were facilitated by AWS_INC. |

Taxing scenarios

The taxation model that is in place for the country and state of the buyer and seller dictates how taxes are collected and remitted. Following are the possible scenarios:

- Taxes are collected and remitted by AWS. In these cases, the `transaction_type` is `AWS_TAX_SHARE`.
- Taxes are collected by AWS, disbursed to the seller, and remitted by the seller to the tax authorities. In these cases, the `transaction_type` is `SELLER_TAX_SHARE`.
- Taxes are not collected by AWS. The seller must calculate the taxes and remit them to the tax authorities. In these cases, AWS Marketplace doesn't perform tax calculations or receive tax information. The seller pays the taxes from the revenue share.

Examples of billing event data feed

This section shows examples of the billing event data period at the time of invoicing and one month later. Note the following for all tables in this section:

- In data feeds, `billing_event_id` values are 40-character alphanumeric strings. They're shown here as two-character strings for readability.
- In the data feed, this information is presented in a single table. For readability, the data is shown in multiple tables here, and all columns aren't shown.

For the examples in this section, assume the following:

- Arnav is the buyer.
 - His account ID is 737399998888.
 - He's located in France, which is subject to marketplace facilitator laws. For more information, see [Amazon Web Service Tax Help](#).
 - He purchased `prod-o4grxfafcx` and was invoiced \$120.60 for his monthly usage of that product.
 - He paid the invoice within the month.
- Jane is the manufacturer.
 - Her account ID is 111122223333.

- Paulo is the seller of record.
 - His account ID is 777788889999.
 - He lives in Kansas, which is not subject to market facilitator laws.

Billing event data feed for seller of record

As the seller of record, Paulo invoices the buyer, Arnav.

The following tables show the relevant information in Paulo's data feed when he invoices Arnav.

| billing_event | from_account | to_account_id | end_user_account | product_id | action | transaction_type |
|---------------|--------------|---------------|------------------|-----------------|----------|------------------|
| I0 | 737399998888 | 777788889999 | 737399998888 | prod-o4grxfafcx | INVOICED | SELLER_REV_SHARE |
| I1 | 737399998888 | AWS | 737399998888 | prod-o4grxfafcx | INVOICED | AWS_TAX_SHARE |
| I2 | 777788889999 | 1112222333 | 737399998888 | prod-o4grxfafcx | INVOICED | SELLER_REV_SHARE |
| I3 | 777788889999 | AWS | 737399998888 | prod-o4grxfafcx | INVOICED | AWS_REV_SHARE |

| parent_billing_event | disbursement_billing_event | amount | currency | invoice_date | invoice_id |
|----------------------|----------------------------|--------|----------|----------------------|-------------|
| | | 100 | USD | 2018-12-31T00:00:00Z | 07800216640 |
| | | 20.6 | USD | 2018-12-31T00:00:00Z | 07800216640 |
| | | -80 | USD | 2018-12-31T00:00:00Z | 0788576665 |
| | | -0.2 | USD | 2018-12-31T00:00:00Z | 0788576665 |

The following tables show the relevant information in Paulo's data feed at the end of the month, after Arnav pays the invoice.

| billing_event | from_account | to_account_id | end_user_account | product_id | action | transaction_type |
|---------------|--------------|---------------|------------------|-----------------|-----------|------------------|
| I10 | 737399998888 | 777788889999 | 737399998888 | | DISBURSED | SELLER_REV_SHARE |
| I12 | 777788889999 | 1112222333 | 737399998888 | | DISBURSED | SELLER_REV_SHARE |
| I13 | 777788889999 | AWS | 737399998888 | prod-o4grxfafcx | DISBURSED | AWS_REV_SHARE |
| I14 | AWS | 777788889999 | | | DISBURSED | DISBURSEMENT |

| parent_billing_event | disbursement_billing_event | amount | currency | invoice_date | invoice_id |
|----------------------|----------------------------|--------|----------|----------------------|-------------|
| I0 | I14 | -100 | USD | 2018-12-31T00:00:00Z | 07800216640 |

| parent_billing_e | disbursement_b | amount | currency | invoice_date | invoice_id |
|------------------|----------------|--------|----------|---------------------|-------------|
| I2 | I14 | 80 | USD | 2018-12-31T00:04:48 | 07488576665 |
| I3 | I14 | 0.2 | USD | 2018-12-31T00:04:48 | 07488576665 |
| | | 19.8 | USD | | |

Billing event data feed for manufacturer

The following tables show the relevant information in the Jane's data feed when Paulo invoices Arnav.

| billing_event | from_account | to_account_id | end_user_acc | product_id | action | transaction_type |
|---------------|--------------|---------------|--------------|-----------------|----------|------------------|
| I5 | 777788889999 | 11122223333 | | prod-o4grxfafcx | INVOICED | SELLER_REV_SHARE |
| I6 | 777788889999 | 11122223333 | | prod-o4grxfafcx | INVOICED | SELLER_TAX_SHARE |
| I7 | 11122223333 | AWS | | prod-o4grxfafcx | INVOICED | AWS_REV_SHARE |

| parent_billing_e | disbursement_b | amount | currency | invoice_date | invoice_id |
|------------------|----------------|--------|----------|---------------------|-------------|
| | | 73.5 | | 2018-12-31T00:04:48 | 07488576665 |
| | | 6.5 | | 2018-12-31T00:04:48 | 07488576665 |
| | | -7.35 | | 2018-12-31T00:04:48 | 07488576665 |

The following tables show the relevant information in Jane's data feed at the end of the month, after the invoice is paid.

| billing_event | from_account | to_account_id | end_user_acc | product_id | action | transaction_type |
|---------------|--------------|---------------|--------------|-----------------|-----------|------------------|
| I30 | 777788889999 | 11122223333 | | prod-o4grxfafcx | DISBURSED | SELLER_REV_SHARE |
| I31 | 777788889999 | 11122223333 | | prod-o4grxfafcx | DISBURSED | SELLER_TAX_SHARE |
| I32 | 11122223333 | AWS | | prod-o4grxfafcx | DISBURSED | AWS_REV_SHARE |
| I33 | AWS | 11122223333 | | | DISBURSED | DISBURSEMENT |

| parent_billing_e | disbursement_b | amount | currency | invoice_date | invoice_id |
|------------------|----------------|--------|----------|--------------|------------|
| I5 | I33 | -73.5 | USD | | |

| parent_billing_id | disbursement_id | amount | currency | invoice_date | invoice_id |
|-------------------|-----------------|--------|----------|--------------|------------|
| I6 | I33 | -6.5 | USD | | |
| I7 | I33 | 7.35 | USD | | |
| | | 72.65 | USD | | |

Example queries

As described in [Using data feeds \(p. 303\)](#), you can use [Athena](#) to run queries on the data that's collected and stored as data feeds in your managed Amazon S3 bucket. This section provides some examples of common ways you might do this. All examples assume that a single currency is used.

Example 1: Amount invoiced, including taxes

To find out how much buyers were invoiced, including taxes, you can run a query as shown in the following example.

```
SELECT sum(amount) FROM billing_event
WHERE
  action = 'INVOICED'
  AND
  (
    (transaction_type in ('SELLER_REV_SHARE', 'SELLER_TAX_SHARE')
      -- to discard SELLER_REV_SHARE from Manufacturer to Channel Partner, aka cost of
      goods
      AND to_account_id='seller-account-id'
    )
    OR transaction_type= 'AWS_TAX_SHARE'
  );
```

Example 2: Amount invoiced to buyers on seller's behalf

To find out how much buyers were invoiced on a seller's behalf, you can run a query as shown in the following example.

```
SELECT sum(amount) FROM billing_event
WHERE
  action = 'INVOICED'
  AND transaction_type in ('SELLER_REV_SHARE', 'SELLER_TAX_SHARE')
  AND to_account_id='seller-account-id'
;
```

Example 3: Amount AWS can collect on seller's behalf

To find out how much AWS can collect on a seller's behalf, minus any refunds, credits, and forgiven accounts, you can run a query as shown in the following example.

```
SELECT sum(amount) FROM billing_event
WHERE
  -- what is invoiced on behalf of SELLER, incl. refunds/ credits and cost of goods
  transaction_type like 'SELLER_%'
  -- FORGIVEN action records will "negate" related INVOICED
  and action in ('INVOICED','FORGIVEN')
;
```

Example 4: Amount seller can collect

To find out how much sellers can collect, you can run a query as shown in the following example. This example removes listing fees and taxes that AWS collects, and adds any exceptional balance adjustments.

```
SELECT sum(amount) FROM billing_event
WHERE
  (transaction_type like 'SELLER_%' -- what is invoiced on behalf of SELLER
  or transaction_type like 'AWS_REV_%' -- what is owed to AWS
  or transaction_type = 'BALANCE_ADJUSTMENT' -- exceptionnal case
  )
  and action in ('INVOICED','FORGIVEN')
;
```

You can also use the following query to collect the same information, as shown in the following example.

```
SELECT sum(amount) FROM billing_event
WHERE
  balance_impacting = 1
  and action in ('INVOICED','FORGIVEN')
;
```

The following example shows the same information, but is restricted to 2018 transactions and assumes all buyers paid their invoices.

```
SELECT sum(amount) FROM billing_event
WHERE
  invoice_date between '2018-01-01' and '2018-12-31'
  and balance_impacting = 1
  and action in ('INVOICED','FORGIVEN')
;
```

Example 5: Amount of disbursements

To find out the amount that's been disbursed, you can run a query as shown in the following example.

```
select sum(amount) FROM billing_event
WHERE
  action ='DISBURSED'
  and transaction_type like 'DISBURSEMENT%'
;
```

Example 6: Amount pending disbursement

To find out the amount that's pending disbursement, you can run a query as shown in the following example. This query removes amounts that have already been disbursed.

```
SELECT sum(amount) FROM billing_event targeted
WHERE
  (transaction_type like 'SELLER_%' -- what is invoiced on behalf of SELLER
  or transaction_type like 'AWS_REV_%' -- what is owed to AWS
  or transaction_type = 'BALANCE_ADJUSTMENT' -- exceptionnal case
  )
  -- DISBURSEMENT action records will "negate" 'INVOICED'
  -- but do not take into account failed disbursements
  AND
  (not exists
   (select 1
    from billing_event disbursement
```

```
        join billing_event failed_disbursement
          on disbursement.billing_event_id=failed_disbursement.parent_billing_event_id
      where
        disbursement.transaction_type='DISBURSEMENT'
        and failed_disbursement.transaction_type='DISBURSEMENT_FAILURE'
        and targeted.disbursement_billing_event_id=disbursement.billing_event_id
    )
)
;
```

Another way to get the same information is to run a query to get the seller's balance, as shown in the following example.

```
SELECT sum(amount) FROM billing_event
WHERE
  balance_impacting = 1
;
```

The following query extends our example. It restricts the results to 2018 transactions and returns additional details about the transactions.

```
select sum(residual_amount_per_transaction)
from
  (SELECT
    max(billed_invoices.amount) invoiced_amount,
    sum(nvl(disbursed_invoices.amount,0)) disbursed_amount,
    -- Exercise left to the reader:
    -- use transaction_type to distinguish listing fee vs seller-owed money
    -- still pending collection
    max(transaction_type) transaction_type,
    max(billed_invoices.amount)
    + sum(nvl(disbursed_invoices.amount,0)) residual_amount_per_transaction
  FROM billing_event billed_invoices
  -- find related disbursements
  left join billing_event disbursed_invoices
    on disbursed_invoices.action='DISBURSED'
    and disbursed_invoices.parent_billing_event_id=billed_invoices.billing_event_id
  WHERE
    billed_invoices.invoice_date between '2018-01-01' and '2018-12-31'
    and billed_invoices.transaction_type like 'SELLER_%' -- invoiced on behalf of SELLER
    and billed_invoices.action in ('INVOICED','FORGIVEN')
    -- do not take into account failed disbursements
    AND not exists
      (select 1 from billing_event failed_disbursement
        where disbursed_invoices.disbursement_billing_event_id =
        failed_disbursement.parent_billing_event_id
      )
  GROUP BY billed_invoices.billing_event_id
);
```

Example 7: Balance of set of invoices

To learn the sum of a set of invoices, you can run a query as shown in the following example.

```
SELECT invoice_id, sum(amount) FROM billing_event targeted
WHERE
  -- invoice_id is only not null for invoiced records AND disbursed records
  -- linking them to related disbursement -> no need to filter more precisely
  invoice_id in ('XXX','YYY')
  -- filter out failed disbursements
  AND not exists
    (select 1
```

```
from billing_event disbursement
  join billing_event failed_disbursement
    on disbursement.billing_event_id=failed_disbursement.parent_billing_event_id
where
  disbursement.transaction_type='DISBURSEMENT'
  and failed_disbursement.transaction_type='DISBURSEMENT_FAILURE'
  and targeted.disbursement_billing_event_id=disbursement.billing_event_id
)
group by invoice_id;
```

Legacy mapping data feed

This data feed lists how product IDs and offer IDs map to legacy globally unique identifiers (GUIDs). The legacy GUIDs were used in older reports, and the new IDs are used in data feeds and in AWS Marketplace APIs.

This data feed provides information about all products you've created as the seller of record and all products you're authorized to resell.

The legacy mapping data feed is refreshed every 24 hours, so new data is available daily.

The following table explains the names and descriptions of the data feed's columns.

| Column name | Description |
|--------------|---|
| mapping_type | Whether this is a product ID or offer ID. |
| legacy_id | The legacy ID for this product or offer. |
| new_id | The friendly ID for this product or offer. This ID is used as the primary key and with all current API actions. |

Example of legacy mapping data feed

The following shows an example of the legacy mapping data feed. For readability, the data history columns aren't shown. For information about data history fields, see [Historization of the data \(p. 300\)](#).

| mapping_type | legacy_id | new_id |
|--------------|--------------------------------------|---------------------|
| OFFER | 8a806c74-dbd6-403e-9362-bb08f417ff37 | offer-dacpxznflfwin |
| PRODUCT | 1368541d-890b-4b6c-9bb9-4a55306ab642 | product-306ab642 |
| OFFER | 558d8382-6b3a-4c75-8345-a627b552f5f1 | offer-gszhml5npzip |

Offer data feed

This data feed provides information about all offers you've created as the seller of record. If a single offer has multiple revisions, all revisions are included in the data feed.

When you make an offer revision and the data in an exposed field changes, a new record is created in the data feed for the same primary key (`offer_id` plus `offer_revision`), but with a different value for

`valid_from` field. For more information about the data feed history columns, see [Historization of the data \(p. 300\)](#).

The offer data feed is refreshed every 24 hours, so new data is available daily.

The following table explains the names and descriptions of the data feed's columns.

| Column name | Description |
|--------------------------------------|--|
| <code>offer_id</code> | The friendly identifier for the offer. Can used to join to the <code>offer_id</code> field of the <code>Offer_Product</code> data feed. |
| <code>offer_revision</code> | The offer revision. This field and the <code>offer_id</code> field combine to form the primary key. With <code>offer_id</code> , can used to join to the <code>offer_id</code> and <code>offer_revision</code> fields of the <code>Target_Offer</code> data feed. |
| <code>name</code> | The seller-defined name of the offer. |
| <code>expiration_date</code> | The date and time that the offer expires. |
| <code>opportunity_name</code> | Any opportunity data linked to this offer. If the offer is bound to an AWS opportunity, this field is populated. |
| <code>opportunity_description</code> | Any descriptive information linked to this offer. If the offer is bound to an AWS opportunity, this field is populated. |

Example of offer data feed

The following shows an example of the offer data feed. For readability, the data history columns aren't shown. For information about data history fields, see [Historization of the data \(p. 300\)](#).

| <code>offer_id</code> | <code>offer_revision</code> | <code>name</code> | <code>expiration_date</code> | <code>opportunity_na</code> | <code>opportunity_description</code> |
|-----------------------|-----------------------------|-----------------------------------|------------------------------|-----------------------------|--------------------------------------|
| offer-dacpxznflfwin | 1 | Enterprise Contract Program Offer | 9999-01-01T00:00:00Z | | |
| offer-gszhmle5npzip | 1 | Private offer created by seller | 2020-10-31T00:00:00Z | | |
| offer-hmzhyle8nphlp | 1 | Enterprise Contract Program Offer | 9999-01-01T00:00:00Z | | |

Offer product data feed

One offer can have several products, and one product can be included in different offers. This data feed lists information about the relationships between offers and products.

This data feed provides information about all product offers you've created as the seller of record.

When you add or remove a product from an offer, you create an offer revision.

The offer product data feed is refreshed every 24 hours, so new data is available daily.

The following table explains the names and descriptions of the data feed's columns. For information about the data feed history columns, see [Historization of the data \(p. 300\)](#).

| Column name | Description | | |
|----------------|---|--|--|
| offer_id | The friendly identifier of this offer. Can used to join to the offer_id field of the Offer data feed. | | |
| offer_revision | Combines with offer_id field to form the foreign key to the offer revision. | | |
| product_id | The friendly identifier of the product, this is the foreign key to the product that this offer exposes. Can used to join to the product_id field of the Product data feed. | | |

Example of Offer product data feed

The following shows an example of the Offer product data feed.

| offer_id | offer_revision | product_id |
|--------------------|----------------|-------------------|
| offer-dacpxznflwin | 10 | prod-o4grxfafcxxx |
| offer-gszhml5npzip | 24 | prod-o4grxfafcxy |

Offer target data feed

This data feed lists targets of an offer's revision for all offers you've created as the seller of record. If a single offer has multiple revisions, all revisions are included in the data feed.

When you make an offer revision and the data in an exposed field changes, a new record is created in the data feed for the same primary key (offer_id plus offer_revision), but with a different value for valid_from field.

The offer target data feed is refreshed every 24 hours, so new data is available daily.

The following table explains the names and descriptions of the data feed's columns.

| Column name | Description |
|-------------------------|--|
| offer_target_id | The primary key of the feed. |
| offer_id+offer_revision | The identifier and revision of the offer. These two columns reference the offer that this target relates to. Can used to join to the offer_id and offer_revision fields of the Target data feed. |
| target_type | Indicates whether the offer recipient is BuyerAccounts, which indicates a private offer, or ParticipatingPrograms. |
| polarity | Indicates whether the offer is intended to be made to the target_type. Acceptable values are as follows: <ul style="list-style-type: none"> PositiveTargeting – The offer applies for this target_type. NegativeTargeting – The offer doesn't apply for this target_type. |
| value | A string that represents the target: either an AWS account ID or a program that can be used with an offer. For example, Standard Contract for AWS Marketplace (SCMP) (p. 54) , Enterprise Contract for AWS Marketplace (ECMP) (p. 55) , or AWS Marketplace Field Demonstration Program (FDP) (p. 25) . |

Example of offer target data feed

The following shows an example of the offer target data feed. For readability, the data history columns aren't shown. For information about data history fields, see [Historization of the data \(p. 300\)](#).

| offer_target_id | offer_id | offer_revision | target_type | polarity | value |
|---|---|----------------|-----------------------|-------------------|--------------------|
| 925ddc73f6a373b7c5544ea3210610803b600 | 8ff6b960d3e14517bcd6e600 dacpxznflwin | 1 | ParticipatingPrograms | PositiveTargeting | EnterpriseContract |
| 471ff22ae3165278ff6b960d3e14517bcd601 | 8ff6b960d3e14517bcd6e600 gszhmle5npzip | 1 | ParticipatingPrograms | PositiveTargeting | FieldDemonstration |
| 511ff22adfj65278ff6b960d3e14517bcd6e602 | 8ff6b960d3e14517bcd6e600 gszhmle5npzip | 1 | ParticipatingPrograms | PositiveTargeting | EnterpriseContract |

Product data feed

This data feed provides information about all products you've created as the seller of record and all products you're authorized to resell.

Product data is mutable. This means that when you change the value for one of the following fields, a new record is created in the data feed with a different value for `valid_from` field. For more information about the data feed history columns, see [Historization of the data \(p. 300\)](#).

The product data feed is refreshed every 24 hours, so new data is available daily.

The following table explains the names and descriptions of the data feed's columns.

| Column name | Description |
|--------------------------------------|---|
| <code>product_id</code> | The friendly identifier of the product. Can used to join to the <code>product_id</code> fields of the <code>Account</code> , <code>Billing_Event</code> , and <code>Offer_Product</code> data feeds. |
| <code>manufacturer_account_id</code> | The identifier of the product owner. This is a foreign key to the Account (p. 320) data feed. Can used to join to the <code>account_id</code> field of the <code>Account</code> data feed. |
| <code>product_code</code> | The existing entitlement product code used to meter the product. This value is also used to join data with a report, or to reference what is provided in AWS Marketplace Metering Service. |
| <code>title</code> | The title of the product. |

Example of product data feed

The following shows an example of the offer target data feed. For readability, the data history columns aren't shown. For information about data history fields, see [Historization of the data \(p. 300\)](#).

| <code>product_id</code> | <code>manufacturer_account_i</code> | <code>product_code</code> | <code>title</code> |
|--------------------------------|-------------------------------------|-----------------------------|--------------------|
| <code>prod-o4grxfafcxxx</code> | 555568000000 | <code>product_code_1</code> | Product1 |
| <code>prod-t3grxfafcxy</code> | 444457000000 | <code>product_code_2</code> | Product2 |
| <code>prod-x8faxxfafcxy</code> | 666678000000 | <code>product_code_3</code> | Product3 |

Tax item data feed

This data feed provides information about tax calculations for a customer invoice.

There can be multiple line items (`line_item_id`) for a given product (`product_id`) of a given customer invoice (`invoice_id`), one or more for each tax jurisdiction. This happens, for example, with usage-based bills for customers who are using different AWS Region rules by different AWS entities (say, the US and Ireland). To learn more about where AWS collects sales tax, VAT, or GST on your sales and remits such taxes to the local tax authorities, in the name of AWS, Inc., see [Amazon Web Service Tax Help](#).

The tax item data feed is refreshed every 24 hours, so new data is available daily.

Tax item data is immutable.

The following table explains the names and descriptions of the data feed's columns. For information about data history columns, see [Historization of the data \(p. 300\)](#).

| Column name | Description |
|---------------------------|--|
| tax_item_id | A unique identifier for a tax item record. |
| invoice_id | The AWS invoice ID. You can use this value with the value of <code>product_id</code> to find related tax billing events. |
| line_item_id | A unique identifier for a customer bill line item. Refund transactions have the same line item ID as their forward tax transactions. |
| customer_bill_id | The unique identifier of the customer bill. Buyers can share this identifier with the seller to help identify and resolve tax calculation questions. |
| tax_liable_party | <p>Either <code>AWS</code> or <code>Seller</code>. If the seller is the tax liable party, taxes are collected. If <code>AWS</code> is the tax liable party, sales tax is collected and remitted by <code>AWS</code>. For more information, see AWS Marketplace Sellers & Tax Collection.</p> <p>If no taxes are collected, there is no value shown here. The seller needs to determine whether some taxes were collected for each invoice, as the seller is liable for tax collection.</p> |
| transaction_type_code | <p>The type of transaction. Possible values are as follows:</p> <ul style="list-style-type: none"> • <code>AWS</code> – A forward tax transaction • <code>REFUND</code> – A full or partial refund • <code>TAXONLYREFUND</code> – A tax-only refund <p>Refund transactions share the line item ID with their original forward transactions.</p> |
| product_id | <p>A foreign key to the product.</p> <p>Can be used to join to the <code>Product</code> data feed on the <code>product_id</code> field.</p> |
| product_tax_code | A standard code to identify the tax properties for a product. Sellers choose the properties when creating or modifying the product. |
| invoice_date | The date the invoice was created. |
| taxed_customer_account_id | <p>A foreign key to the account entity who is taxed.</p> <p>Can be used to join to the <code>Account</code> data feed on the <code>account_id</code> field.</p> |
| taxed_customer_country | The ISO 3166 alpha 2 country code of the address used for tax calculations. |

| Column name | Description |
|--|---|
| taxed_customer_state_or_region | The state, region, or province used for tax calculations. |
| taxed_customer_city | The city used for tax calculations. |
| taxed_customer_postal_code | The postal code used for tax calculations. |
| tax_location_code_taxed_jurisdiction | The vertex geocode that is associated with the taxed location. |
| tax_type_code | The type of tax that is applied to the transaction. The possible values are <code>None</code> , <code>Sales</code> , and <code>SellerUse</code> . |
| jurisdiction_level | The jurisdiction level of the address that is used for tax location. The possible values are <code>State</code> , <code>County</code> , <code>City</code> , and <code>District</code> . |
| taxed_jurisdiction | The name of the tax jurisdiction. |
| display_price_taxability_type | Whether the price that buyers see is inclusive or exclusive of taxes. All AWS Marketplace offerings are exclusive of taxes. |
| taxable_amount | The amount of the transaction that is taxable, at this jurisdiction level. |
| nontaxable_amount | The amount of the transaction that is nontaxable, at this jurisdiction level. |
| tax_jurisdiction_rate | The tax rate that is applied, at this jurisdiction level. |
| tax_amount | The amount of tax that is charged, at this jurisdiction level. |
| tax_currency | The ISO 4217 alpha 3 currency code for above amounts. |
| tax_calculation_reason_code | Whether the transaction is taxable, not taxable, exempt, or zero-rated, organized by the jurisdiction level. |
| date_used_for_tax_calculation | The date that is used for calculating tax on the transaction. |
| customer_exemption_certificate_id | The certificate ID of the exemption certificate. |
| customer_exemption_certificate_id_domain | The location where the certificate is stored on Amazon systems. |
| customer_exemption_certificate_level | The jurisdiction level that supplied the exemption. |
| customer_exemption_code | The code that specifies the exemption; for example, <code>RESALE</code> . |
| customer_exemption_domain | The Amazon system that is used to capture the customer exemption information, if available. |

| Column name | Description |
|--------------------------|--|
| transaction_reference_id | An identifier that allows you to cross-reference data from the following reports: <ul style="list-style-type: none"> Disbursement report (p. 283) Monthly billed revenue report (p. 288) Sales compensation report (p. 294) US sales and use tax report (p. 296) |

Note

Beginning August 5, 2021, international Marketplace Facilitator taxes for AWS Marketplace sales will have entries in the tax item data feed. This means that, beginning August 5, 2021, every AWS_TAX_SHARE and SELLER_TAX_SHARE record in the billing event data feed is expected to have a corresponding record in the tax item data feed.

Example of tax item data feed

The following shows an example of the tax item data feed. In the data feed, this information is presented in a single table. For readability, the data is shown in multiple tables here, and all columns aren't shown.

| tax_item_id | invoice_id | line_item_id | customer_bill_id |
|------------------------------------|-------------|-----------------|----------------------|
| 6p2ni6tu041xagvhbyanbg785k21m6840 | 785k21m6840 | 6txjoav_0001 | 71000000000000000000 |
| 6p2ni6tu041xagvhbyanbg785k21m6840 | 785k21m6840 | 6txjoav_0002 | 53000000000000000000 |
| flr4jobxjzww8czdsrq4noue2508156769 | 2508156769 | 5wxw0k7_0001 | 76400000000000000000 |
| gfkjjobxjzww56jgkrsrqgjt5280336398 | 280336398 | 5gj567d_0002 | 76400000000000000000 |
| wwk1qpvb8ran3geiw8e3m456d32077 | 456d32077 | 7wpkuwhgk1_0001 | 99300000000000000000 |
| wwk1qpvb8ran3geiw8e3m456d32077 | 456d32077 | 7wpkuwhgk1_0002 | 99300000000000000000 |
| fnohdid8kwgqq9lvii2k30sp229897654 | 229897654 | 5e8h75x_0001 | 92100000000000000000 |

| tax_liable_party | transaction_type_code | product_id | product_tax_code | invoice_date |
|------------------|-----------------------|--------------------|----------------------------|----------------------|
| Seller | AWS | prod-o4grxfafcx | AWSMP_SOFTWARE | 2018-12-31T00:00:00Z |
| Seller | AWS | prod-o4grxfafcx | AWSMP_SOFTWARE | 2018-12-31T00:00:00Z |
| Seller | AWS | prod-t3grxfafcxxy | AWS_REMOTE_ACCESS_SOFTWARE | 2018-12-31T00:00:00Z |
| Seller | REFUND | prod-t3grxfafcxxy | AWS_REMOTE_ACCESS_SOFTWARE | 2018-12-31T00:00:00Z |
| Seller | AWS | prod-x8faxxfafcxxy | AWS_REMOTE_ACCESS_SOFTWARE | 2018-12-31T00:00:00Z |

AWS Marketplace Seller Guide
Tax item data feed

| tax_liable_party | transaction_type_code | product_id | product_tax_code | invoice_date |
|------------------|-----------------------|--------------------|---------------------------|----------------------|
| Seller | TAXONLYREFUND | prod-x8faxxfafcxxy | AWS_REMOTE_ACCESS_STORAGE | 2018-07-31T00:00:00Z |
| AWS | AWS | prod-wghj8xfafrhgj | AWS_REMOTE_ACCESS_STORAGE | 2018-07-31T00:00:00Z |

| taxed_customer_account_id | taxed_customer_code | taxed_customer_state | taxed_customer_city | taxed_customer_postal_code |
|---------------------------|---------------------|----------------------|---------------------|----------------------------|
| VleGa2t9j3MuxioH9wL8 | US | GA | MILTON | 48573-4839 |
| VleGa2t9j3MuxioH9wL8 | US | GA | MILTON | 48573-4839 |
| 7nyo5jwTRoPlyX81vx95 | US | NC | DURHAM | 27517-4834 |
| 7nyo5jwTRoPlyX81vx95 | US | NC | DURHAM | 27517-4834 |
| 7nyo5jwTRoPlyX81vx95 | US | TX | NOT APPLICABLE | 75844-1235 |
| 7nyo5jwTRoPlyX81vx95 | US | TX | HOUSTON | 75844-1235 |
| 192a0421313e41f0695 | US | CT | NEW HAVEN | 06002-2948 |

| tax_location_id | tax_type_code | jurisdiction_level | taxed_jurisdiction | display_price | taxable_amount | nontaxable_amount |
|-----------------|---------------|--------------------|--------------------|---------------|----------------|-------------------|
| 460473664 | Sales | State | GA | Exclusive | 100 | 0 |
| 66301164 | Sales | County | FULTON | Exclusive | 0 | 100 |
| 692938178 | SellerUse | State | NC | Exclusive | 58.1 | 523.8 |
| 692938178 | SellerUse | State | NC | Exclusive | -58.1 | 523.8 |
| 356794387 | Sales | State | TX | Exclusive | 1105.14 | 0 |
| 528887443 | Sales | City | HOUSTON | Exclusive | -36 | 0 |
| 171248162 | Sales | State | CT | Exclusive | 0 | 114.55 |

| tax_jurisdiction_rate | tax_amount | tax_currency | tax_calculation_reason | date_used_for_tax_calculation |
|-----------------------|------------|--------------|------------------------|-------------------------------|
| 0.206 | 20.6 | USD | Taxable | 2018-10-31T00:00:00Z |
| 0 | 0 | USD | NonTaxable | 2018-10-31T00:00:00Z |
| 0.1 | 5.8 | USD | Taxable | 2018-07-31T00:00:00Z |
| 0.1 | -5.8 | USD | Taxable | 2018-07-31T00:00:00Z |
| 0.06 | 66.3 | USD | Taxable | 2018-07-31T00:00:00Z |
| 0.01 | -0.36 | USD | NonTaxable | 2018-07-31T00:00:00Z |

AWS Marketplace Seller Guide
Tax item data feed

| tax_jurisdiction_ra | tax_amount | tax_currency | tax_calculation_rea | date_used_for_tax_calculation |
|----------------------------|-------------------|---------------------|----------------------------|--------------------------------------|
| 0 | 0 | USD | Exempt | 2019-06-30T00:00:00Z |

AWS Marketplace security

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The [shared responsibility model](#) describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. The effectiveness of our security is regularly tested and verified by third-party auditors as part of the [AWS compliance programs](#). To learn about the compliance programs that apply to AWS Marketplace, see [AWS Services in Scope by Compliance Program](#).
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You're also responsible for other factors including the sensitivity of your data, your organization's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using AWS Marketplace. The following topics show you how to configure AWS Identity and Access Management to manage access to AWS Marketplace in order to meet your security and compliance objectives. You can also learn how to use other AWS services that can help you to monitor and secure your AWS Marketplace resources.

To learn more about security and other policies regarding the products that you offer in AWS Marketplace, see the following topics:

- [AMI security policies \(p. 113\)](#)
- [Container-based product requirements \(p. 126\)](#)
- [SaaS product guidelines \(p. 226\)](#)
- [Requirements for professional services products \(p. 258\)](#)

Note

To learn about security on AWS Data Exchange for data products, see [Security](#) in the *AWS Data Exchange User Guide*.

To learn about security for buyers in AWS Marketplace, see [Security on AWS Marketplace](#) in the *AWS Marketplace Buyer Guide*.

Topics

- [Controlling access to AWS Marketplace Management Portal \(p. 343\)](#)
- [Policies and permissions for AWS Marketplace sellers \(p. 346\)](#)
- [AWS managed policies for AWS Marketplace sellers \(p. 349\)](#)
- [AWS Marketplace Commerce Analytics Service account permissions \(p. 356\)](#)
- [AWS Marketplace Product Support Connection account permissions \(p. 357\)](#)
- [Amazon SQS permissions \(p. 357\)](#)
- [AWS Marketplace metering and entitlement API permissions \(p. 358\)](#)
- [Logging AWS Marketplace API calls with AWS CloudTrail \(p. 359\)](#)

Controlling access to AWS Marketplace Management Portal

AWS Identity and Access Management (IAM) is an AWS service that helps you control access to AWS resources. If you are an IAM administrator, you control who can be *authenticated* (signed in) and *authorized* (have permissions) to use AWS Marketplace resources. IAM is an AWS service that you can use with no additional charge.

The recommended way to control who can do what in AWS Marketplace Management Portal is to use IAM to create users and groups. Then you add the users to the groups, and manage the groups. For example, if John should be allowed to view your products, create an IAM user for him and add his IAM user to a group you create for read-only access. You can assign a policy or permissions to the group that provide read-only permissions. If you have other users that need read-only access, you can add them to the group you created rather than adding permissions to their user account. If John's role changes and he no longer needs read-only access, you can remove John's user account from the group.

A *policy* is a document that defines the permissions that apply to a user, group, or role. In turn, the permissions determine what users can do in AWS. A policy typically allows access to specific actions, and can optionally grant that the actions are allowed for specific resources, like Amazon EC2 instances, Amazon S3 buckets, and so on. Policies can also explicitly deny access. A *permission* is a statement within a policy that allows or denies access to a particular resource. You can state any permission like this: "A has permission to do B to C." For example, Jane (A) has permission to read messages (B) from John's Amazon Simple Queue Service queue (C). Whenever Jane sends a request to Amazon SQS to use John's queue, the service checks to see if she has permission. It further checks to see if the request satisfies the conditions John specified in the permission.

Important

All of the IAM users that you create authenticate by using their credentials. However, they use the same AWS account. Any change that a user makes can impact the whole account.

AWS Marketplace has permissions defined to control the actions that someone with those permissions can take in AWS Marketplace Management Portal. There are also policies that AWS Marketplace created and manage that combine several permissions. For example, the `aws-marketplace-management:ViewMarketing` permission gives a user access to the **Marketing** tab in AWS Marketplace Management Portal. The `AWSMarketplaceSellerProductsFullAccess` policy gives the user full access to products in the AWS Marketplace Management Portal.

The following resources provide more information about getting started and using IAM.

- [Creating Your First IAM Admin User and Group](#)
- [IAM Best Practices](#)
- [Managing IAM Policies](#)
- [Attaching a Policy to an IAM Group](#)
- [Identities \(Users, Groups, and Roles\)](#)
- [Controlling Access Using Policies](#)

The following provides some high-level guidance for creating users and groups, and logging in as an IAM user.

Creating users

To allow people in your company to sign in to the AWS Marketplace Management Portal, create an IAM user for each person who needs access.

To create IAM users

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Users** and then choose **Create New Users**.
3. In the numbered text boxes, enter a name for each user that you want to create.
4. Clear the **Generate an access key for each user** check box and then choose **Create**.

To assign a password to each user that you just created

1. In the list of users, choose the name of a new user.
2. Choose the **Security Credentials** tab and then choose **Manage Password**.
3. Choose an option for either an auto-generated password or a custom password. Optionally, to require the user to choose a new password at the next sign-in, select the box for **Require user to create a new password at next sign-in**. Choose **Apply**.
4. Choose **Download Credentials** to save the user name, password, and account-specific sign-in URL to a comma-separated values (CSV) file on your computer. Then choose **Close**.

Note

To sign in with the IAM user name and password that you just created, users must navigate to your account-specific sign-in URL. This URL is in the credentials file that you just downloaded and is also available on the IAM console. For more information, see [How IAM Users Sign In to Your AWS Account](#) in the *IAM User Guide*.

Tip

Create a user name and password for yourself as well, even though you're the AWS account owner. It's a recommended best practice for everyone to work in the AWS Marketplace Management Portal as an IAM user, even the account owner. To learn how to create an IAM user for yourself that has administrative permissions, go to [Creating an Administrators Group](#) in the *IAM User Guide*.

Creating or using groups

After you create users, create groups, create permissions to access the pages in the AWS Marketplace Management Portal, add those permissions to the groups, and then add users to the groups.

When you assign permissions to a group, you allow any member of that group to perform specific actions. When you add a new user to the group, that user automatically gains the permissions that are assigned to the group. A group can have permissions for more than one action. We recommend using a managed policy rather than creating your own policy.

To assign a managed policy for AWS Marketplace to a group

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Groups**, and then choose the group you want to attach a policy to.
3. On the summary page for the group, under the **Permissions** tab, choose **Attach Policy**.
4. On the **Attach Policy** page, next to **Filter**: enter **awsmarketplace**.
5. Choose the policy or policies you want to attach, and then choose **Attach Policy**.

To create a policy with AWS Marketplace Management Portal permissions

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.

2. In the navigation pane, choose **Policies** and then choose **Create Policy**.
3. Next to **Policy Generator**, choose **Select**.
4. On the **Edit Permissions** page, do the following:
 - a. For **Effect**, choose **Allow**.
 - b. For **AWS Service**, choose **AWS Marketplace Management Portal**.
 - c. For **Actions**, select the permission or permissions to allow.
 - d. Choose **Add Statement**.
 - e. Choose **Next Step**.
5. On the **Review Policy** page, do the following:
 - a. For **Policy Name**, enter a name for this policy. Take note of the policy name because you need it for a later step.
 - b. (Optional) For **Description**, enter a description for this policy.
 - c. Choose **Create Policy**.

To create an IAM group with appropriate permissions and add users to the group

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Groups** and then choose **Create New Group**.
3. For **Group Name**, type a name for the group. Then choose **Next Step**.
4. On the **Attach Policy** page, do the following:
 - a. For **Filter**, choose **Customer Managed Policies**.
 - b. Select the check box next to the name of the policy that you want to attach to this group. This is typically the policy that you just created.
 - c. Choose **Next Step**.
5. Choose **Create Group**.
6. Find your new group in the **Groups** list and then select the check box next to it. Choose **Group Actions** and then **Add Users to Group**.
7. Select the check box next to each user to add to the group and then choose **Add Users**.

Signing in as an IAM user

After you have created users in IAM, users can sign in with their own user names and passwords. To do so, they need to use the unique URL that is associated with your AWS account. You can get and distribute the sign-in URL to your users.

To get your account's unique sign-in URL

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Dashboard**.
3. Near the top of the content pane, find **IAM users sign-in link**: and take note of the sign-in link, which has a format like this:

```
https://AWS_account_ID.signin.aws.amazon.com/console/
```

Note

If you want the URL for your sign-in page to contain your company name (or other friendly identifier) instead of your AWS account ID, you can create an alias for your account by

choosing **Customize**. For more information, see [Your AWS Account ID and Its Alias](#) in the *IAM User Guide*.

4. Distribute this URL to the people at your company who can work with the AWS Marketplace Management Portal, along with the user name and password that you created for each. Instruct them to use your account's unique sign-in URL to sign in before they access the AWS Marketplace Management Portal.

Policies and permissions for AWS Marketplace sellers

AWS Marketplace has three managed policies you can use with the AWS Marketplace Management Portal. In addition, you can use individual permissions to create your own AWS Identity and Access Management (IAM) policy.

Note

To learn about policies and permissions on AWS Data Exchange for data products, see [Identity and Access Management in AWS Data Exchange](#) in the *AWS Data Exchange User Guide*.

To learn about policies and permissions for AWS Marketplace buyers, see [Controlling access to AWS Marketplace subscriptions](#) in the *AWS Marketplace Buyer Guide*.

Policies for AWS Marketplace sellers

You can use the following managed policies to provide IAM users with controlled access to the AWS Marketplace Management Portal:

AWSMarketplaceSellerFullAccess

Allows full access to all of the pages in the AWS Marketplace Management Portal and other AWS services, such as Amazon Machine Image (AMI) management.

AWSMarketplaceSellerProductsFullAccess

Allows full access to the [Products](#) pages in the AWS Marketplace Management Portal.

AWSMarketplaceSellerProductsReadOnly

Allows read-only access to the [Products](#) pages in the AWS Marketplace Management Portal.

Important

AWS Marketplace buyers can use managed policies to manage the subscriptions they purchase. The managed policies you use with AWS Marketplace Management Portal start with `AWSMarketplaceSeller`. When you search for policies in IAM, make sure to search for policies that start with `AWSMarketplaceSeller`.

AWS Marketplace also provides specialized managed policies for specific scenarios. For a full list of AWS managed policies for AWS Marketplace sellers, as well as descriptions of what permissions they provide, see [AWS managed policies for AWS Marketplace sellers](#) (p. 349).

Permissions for AWS Marketplace sellers

You can use the following permissions in IAM policies for the AWS Marketplace Management Portal:

aws-marketplace-management:viewMarketing

Allows access to the [Marketing](#) page in the AWS Marketplace Management Portal.

aws-marketplace-management:viewSupport

Allows access to the [Customer Support Eligibility](#) page in the AWS Marketplace Management Portal.

aws-marketplace-management:viewReports

Allows access to the [Reports](#) page in the AWS Marketplace Management Portal.

aws-marketplace-management:uploadFiles

Allows access to the [File Upload](#) page in the AWS Marketplace Management Portal.

aws-marketplace-management:viewSettings

Allows access to the [Settings](#) page in the AWS Marketplace Management Portal.

aws-marketplace:ListEntities

Allows access to list objects in AWS Marketplace Management Portal. Required to access the [Offers](#) and [Partners](#) pages in the AWS Marketplace Management Portal.

aws-marketplace:DescribeEntity

Allows access to see details of objects in AWS Marketplace Management Portal. Required to access the [Offers](#), [Partners](#), and [Agreements](#) pages in the AWS Marketplace Management Portal.

aws-marketplace:StartChangeSet

Allows access to create product changes in AWS Marketplace Management Portal. Required to make changes in the [Offers](#), [Partners](#), and [Agreements \(p. 52\)](#) pages in the AWS Marketplace Management Portal.

aws-marketplace:SearchAgreements

Allows viewing the high-level list of agreements on the [Agreements \(p. 52\)](#) page, as well as opportunities between ISVs and consulting partners on the [Partners \(p. 49\)](#) page.

aws-marketplace:DescribeAgreement

Allows viewing of high-level agreement details on the [Agreements](#) page, as well as opportunities between ISVs and consulting partners on the [Partners](#) page.

aws-marketplace:GetAgreementTerms

Allows viewing all agreement term details on the [Agreements](#) page, as well as opportunities between ISVs and consulting partners on the [Partners](#) page.

Note

To enable a user to access the [Manage Products](#) page, you must use either the `AWSMarketplaceSellerProductsFullAccess` or `AWSMarketplaceSellerProductsReadOnly` managed permissions.

You can combine the preceding permissions into a single IAM policy to grant the permissions that you want. See the following examples.

Example 1: Permissions to access the Marketing and File Upload pages.

To grant permissions to both the **Marketing** page and the **File Upload** page, use a policy similar to the following example.

```
{
```

```
"Version": "2012-10-17",
"Statement": [{
  "Effect": "Allow",
  "Action": [
    "aws-marketplace-management:viewMarketing",
    "aws-marketplace-management:uploadFiles"
  ],
  "Resource": ["*"]
}]
}
```

Example 2: Permissions to create upgrades and renewals for private offers

To grant permissions to view and use the **Agreements** page to create upgrades and renewals for private offers, use a policy similar to the following example.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "aws-marketplace:SearchAgreements",
        "aws-marketplace:DescribeAgreement",
        "aws-marketplace:GetAgreementTerms",
        "aws-marketplace:DescribeEntity",
        "aws-marketplace:StartChangeSet"
      ],
      "Effect": "Allow",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws-marketplace:PartyType": "Proposer"
        },
        "ForAllValues:StringEquals": {
          "aws-marketplace:AgreementType": [
            "PurchaseAgreement"
          ]
        }
      }
    }
  ]
}
```

Example 3: Permissions to access the Offers page and create new private offers

To grant permissions to view and use the **Offers** page to view existing and create new private offers, use a policy similar to the following example.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "aws-marketplace:ListEntities",
        "aws-marketplace:DescribeEntity",
        "aws-marketplace:StartChangeSet"
      ],
      "Effect": "Allow",

```

```
    "Resource": "*",  
  }  
]  
}
```

Using IAM groups

Alternatively, you can create separate IAM groups for granting access to each individual page in the AWS Marketplace Management Portal. Users can belong to more than one group. So, if a user needs access to more than one page, you can add the user to all of the appropriate groups. For example, create one IAM group and grant that group permission to access the **Marketing** page, create another group and grant that group permission to access the **File Upload** page, and so on. If a user needs permission to access both the **Marketing** page and the **File Upload** page, add the user to both groups.

For more information about IAM users and groups, see [Identities \(Users, Groups, and Roles\)](#) in the *IAM User Guide*.

AWS managed policies for AWS Marketplace sellers

To add permissions to users, groups, and roles, it is easier to use AWS managed policies than to write policies yourself. It takes time and expertise to [create IAM customer managed policies](#) that provide your team with only the permissions they need. To get started quickly, you can use our AWS managed policies. These policies cover common use cases and are available in your AWS account. For more information about AWS managed policies, see [AWS managed policies](#) in the *IAM User Guide*.

AWS services maintain and update AWS managed policies. You can't change the permissions in AWS managed policies. Services occasionally add additional permissions to an AWS managed policy to support new features. This type of update affects all identities (users, groups, and roles) where the policy is attached. Services are most likely to update an AWS managed policy when a new feature is launched or when new operations become available. Services do not remove permissions from an AWS managed policy, so policy updates won't break your existing permissions.

Additionally, AWS supports managed policies for job functions that span multiple services. For example, the **ViewOnlyAccess** AWS managed policy provides read-only access to many AWS services and resources. When a service launches a new feature, AWS adds read-only permissions for new operations and resources. For a list and descriptions of job function policies, see [AWS managed policies for job functions](#) in the *IAM User Guide*.

This section lists each of the policies used to manage seller access to AWS Marketplace. For information about buyer policies, see [AWS managed policies for AWS Marketplace buyers](#) in the *AWS Marketplace Buyer Guide*.

Topics

- [AWS managed policy: AWSMarketplaceAmiIngestion](#) (p. 350)
- [AWS managed policy: AWSMarketplaceFullAccess](#) (p. 350)
- [AWS managed policy: AWSMarketplaceGetEntitlements](#) (p. 352)
- [AWS managed policy: AWSMarketplaceMeteringFullAccess](#) (p. 352)
- [AWS managed policy: AWSMarketplaceMeteringRegisterUsage](#) (p. 353)
- [AWS managed policy: AWSMarketplaceSellerFullAccess](#) (p. 353)
- [AWS managed policy: AWSMarketplaceSellerProductsFullAccess](#) (p. 354)
- [AWS managed policy: AWSMarketplaceSellerProductsReadOnly](#) (p. 355)
- [AWS Marketplace updates to AWS managed policies](#) (p. 356)

AWS managed policy: AWSMarketplaceAmiIngestion

You can create a service role with this policy that can then be used by AWS Marketplace to perform actions on your behalf. For more information about using AWSMarketplaceAmiIngestion, see [Giving AWS Marketplace access to your AMI \(p. 71\)](#).

This policy is used to grant contributor permissions that allow AWS Marketplace to copy your Amazon Machine Images (AMIs) in order to list them on AWS Marketplace.

Permissions details

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "ec2:ModifySnapshotAttribute"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:ec2:us-east-1::snapshot/snap-*"
    },
    {
      "Action": [
        "ec2:DescribeImageAttribute",
        "ec2:DescribeImages",
        "ec2:DescribeSnapshotAttribute",
        "ec2:ModifyImageAttribute"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

AWS managed policy: AWSMarketplaceFullAccess

You can attach the AWSMarketplaceFullAccess policy to your IAM identities.

This policy grants administrative permissions that allow full access to AWS Marketplace and related services, both as a seller and a buyer. These permissions include the ability to subscribe and unsubscribe to AWS Marketplace software, manage AWS Marketplace software instances from the AWS Marketplace, creating and managing private marketplace in your account, as well as access to Amazon EC2, AWS CloudFormation, and Amazon EC2 Systems Manager.

Permissions details

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "aws-marketplace:*",
        "cloudformation:CreateStack",
        "cloudformation:DescribeStackResource",
        "cloudformation:DescribeStackResources",
        "cloudformation:DescribeStacks",
        "cloudformation:List*",
        "ec2:AuthorizeSecurityGroupEgress",

```

```

    "ec2:AuthorizeSecurityGroupIngress",
    "ec2:CreateSecurityGroup",
    "ec2:CreateTags",
    "ec2:DescribeAccountAttributes",
    "ec2:DescribeAddresses",
    "ec2>DeleteSecurityGroup",
    "ec2:DescribeImages",
    "ec2:DescribeInstances",
    "ec2:DescribeKeyPairs",
    "ec2:DescribeSecurityGroups",
    "ec2:DescribeSubnets",
    "ec2:DescribeTags",
    "ec2:DescribeVpcs",
    "ec2:RunInstances",
    "ec2:StartInstances",
    "ec2:StopInstances",
    "ec2:TerminateInstances"
  ],
  "Resource": "*"
},
{
  "Effect": "Allow",
  "Action": [
    "ec2:CopyImage",
    "ec2:DeregisterImage",
    "ec2:DescribeSnapshots",
    "ec2>DeleteSnapshot",
    "ec2:CreateImage",
    "ec2:DescribeInstanceStatus",
    "ssm:GetAutomationExecution",
    "ssm:UpdateDocumentDefaultVersion",
    "ssm:CreateDocument",
    "ssm:StartAutomationExecution",
    "ssm:ListDocuments",
    "ssm:UpdateDocument",
    "ssm:DescribeDocument",
    "sns:ListTopics",
    "sns:GetTopicAttributes",
    "sns:CreateTopic",
    "iam:GetRole",
    "iam:GetInstanceProfile",
    "iam:ListRoles",
    "iam:ListInstanceProfiles"
  ],
  "Resource": "*"
},
{
  "Effect": "Allow",
  "Action": [
    "s3:ListBucket",
    "s3:GetObject"
  ],
  "Resource": [
    "arn:aws:s3::*image-build*"
  ]
},
{
  "Effect": "Allow",
  "Action": [
    "sns:Publish",
    "sns:setTopicAttributes"
  ],
  "Resource": "arn:aws:sns:*:*:*image-build*"
},
{
  "Effect": "Allow",

```



```
    "Action": [
      "iam:PassRole"
    ],
    "Resource": [
      "*"
    ],
    "Condition": {
      "StringLike": {
        "iam:PassedToService": [
          "ec2.amazonaws.com",
          "ssm.amazonaws.com"
        ]
      }
    }
  }
}
```

AWS managed policy: AWSMarketplaceGetEntitlements

You can attach the `AWSMarketplaceGetEntitlements` policy to your IAM identities.

This policy grants read-only permissions that allow Software-as-a-Service (SaaS) product sellers to check whether a customer has subscribed to their AWS Marketplace SaaS product.

Permissions details

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "aws-marketplace:GetEntitlements"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

AWS managed policy: AWSMarketplaceMeteringFullAccess

You can attach the `AWSMarketplaceMeteringFullAccess` policy to your IAM identities.

This policy grants contributor permissions that allow reporting metered usage that corresponds to AMI and container products with flexible consumption pricing on AWS Marketplace.

Permissions details

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "aws-marketplace:MeterUsage"
      ],

```

```
        "Effect": "Allow",
        "Resource": "*"
    }
  ]
}
```

AWS managed policy: AWSMarketplaceMeteringRegisterUsage

You can attach the `AWSMarketplaceMeteringRegisterUsage` policy to your IAM identities.

This policy grants contributor permissions that allow reporting metered usage that corresponds to container products with hourly pricing on AWS Marketplace.

Permissions details

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "aws-marketplace:RegisterUsage"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

AWS managed policy: AWSMarketplaceSellerFullAccess

You can attach the `AWSMarketplaceSellerFullAccess` policy to your IAM identities.

This policy grants administrative permissions that allow full access to all seller operations on AWS Marketplace, including AWS Marketplace Management Portal, as well as managing the Amazon EC2 Amazon Machine Images (AMI) used in AMI-based products.

Permissions details

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "aws-marketplace-management:uploadFiles",
        "aws-marketplace-management:viewMarketing",
        "aws-marketplace-management:viewReports",
        "aws-marketplace-management:viewSupport",
        "aws-marketplace-management:viewSettings",
        "aws-marketplace:ListChangeSets",
        "aws-marketplace:DescribeChangeSet",
        "aws-marketplace:StartChangeSet",
        "aws-marketplace:CancelChangeSet",
        "aws-marketplace:ListEntities",
        "aws-marketplace:DescribeEntity",

```

```
        "aws-marketplace:ListTasks",
        "aws-marketplace:DescribeTask",
        "aws-marketplace:UpdateTask",
        "aws-marketplace:CompleteTask",
        "ec2:DescribeImages",
        "ec2:DescribeSnapshots",
        "ec2:ModifyImageAttribute",
        "ec2:ModifySnapshotAttribute"
    ],
    "Resource": "*"
  },
  {
    "Action": [
      "aws-marketplace:SearchAgreements",
      "aws-marketplace:DescribeAgreement",
      "aws-marketplace:GetAgreementTerms"
    ],
    "Effect": "Allow",
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "aws-marketplace:PartyType": "Proposer"
      },
      "ForAllValues:StringEquals": {
        "aws-marketplace:AgreementType": [
          "PurchaseAgreement"
        ]
      }
    }
  },
  {
    "Effect": "Allow",
    "Action": [
      "iam:GetRole"
    ],
    "Resource": "arn:aws:iam::*:role/*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "iam:PassRole"
    ],
    "Resource": "arn:aws:iam::*:role/*",
    "Condition": {
      "StringEquals": {
        "iam:PassedToService": "assets.marketplace.amazonaws.com"
      }
    }
  }
}
```

AWS managed policy: AWSMarketplaceSellerProductsFullAccess

You can attach the `AWSMarketplaceSellerProductsFullAccess` policy to your IAM identities.

This policy grants contributor permissions that allow full access to manage products and to the AWS Marketplace Management Portal, as well as managing the Amazon EC2 Amazon Machine Images (AMI) used in AMI-based products.

Permissions details

```
{
```

```
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "aws-marketplace:ListChangeSets",
      "aws-marketplace:DescribeChangeSet",
      "aws-marketplace:StartChangeSet",
      "aws-marketplace:CancelChangeSet",
      "aws-marketplace:ListEntities",
      "aws-marketplace:DescribeEntity",
      "aws-marketplace:ListTasks",
      "aws-marketplace:DescribeTask",
      "aws-marketplace:UpdateTask",
      "aws-marketplace:CompleteTask",
      "ec2:DescribeImages",
      "ec2:DescribeSnapshots",
      "ec2:ModifyImageAttribute",
      "ec2:ModifySnapshotAttribute"
    ],
    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "iam:GetRole"
    ],
    "Resource": "arn:aws:iam::*:role/*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "iam:PassRole"
    ],
    "Resource": "arn:aws:iam::*:role/*",
    "Condition": {
      "StringEquals": {
        "iam:PassedToService": "assets.marketplace.amazonaws.com"
      }
    }
  }
]
```

AWS managed policy: AWSMarketplaceSellerProductsReadOnly

You can attach the `AWSMarketplaceSellerProductsReadOnly` policy to your IAM identities.

This policy grants read-only permissions that allow access to view products on the AWS Marketplace Management Portal, as well as to view the Amazon EC2 Amazon Machine Images (AMI) used in AMI-based products.

Permissions details

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "aws-marketplace:ListChangeSets",
        "aws-marketplace:DescribeChangeSet",
```

```

        "aws-marketplace:ListEntities",
        "aws-marketplace:DescribeEntity",
        "aws-marketplace:ListTasks",
        "aws-marketplace:DescribeTask",
        "ec2:DescribeImages",
        "ec2:DescribeSnapshots"
    ],
    "Resource": "*"
}
]
}

```

AWS Marketplace updates to AWS managed policies

View details about updates to AWS managed policies for AWS Marketplace since this service began tracking these changes. For automatic alerts about changes to this page, subscribe to the RSS feed on the AWS Marketplace [Document history \(p. 364\)](#) page.

| Change | Description | Date |
|--|--|-------------------|
| AWSMarketplaceSellerFullAccess (p. 354) and AWSMarketplaceSellerProductsFullAccess (p. 354) – Update to existing policies | AWS Marketplace updated the policies so that the <code>iam:PassRole</code> Service condition is only applied to <code>iam:PassRole</code> . | November 22, 2021 |
| AWSMarketplaceFullAccess (p. 350) – Update to an existing policy | AWS Marketplace removed a duplicate <code>ec2:DescribeAccountAttributes</code> permission from <code>AWSMarketplaceFullAccess</code> policy. | July 20, 2021 |
| AWS Marketplace started tracking changes | AWS Marketplace started tracking changes for its AWS managed policies. | April 20, 2021 |

AWS Marketplace Commerce Analytics Service account permissions

You can use the following IAM permission policy to allow an IAM user to access the AWS Marketplace Commerce Analytics Service.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "marketplacecommerceanalytics:GenerateDataSet",
      "Resource": "*"
    }
  ]
}

```

```
}
```

For more information about this feature, see [AWS Marketplace Commerce Analytics Service \(p. 14\)](#).

AWS Marketplace Product Support Connection account permissions

The AWS Marketplace product support connection feature makes it possible for customers to provide contact information in the AWS Marketplace website so that you can offer them support for your products. AWS Marketplace shares the data that the customer provides to you through an API. Customers can choose to add contact details during or after they purchase a product that you enrolled in AWS Marketplace product support connection. You use the API to retrieve the customer's contact data, along with relevant product subscription details.

If haven't enrolled in the [the section called "AWS Marketplace Commerce Analytics Service" \(p. 14\)](#), you must configure your account and AWS services to use it. Do the following:

1. (Optional) Create an [IAM user](#).
2. Create a destination [Amazon Simple Storage Service \(Amazon S3\) bucket](#).
3. Create an [Amazon Simple Notification Service \(Amazon SNS\) topic](#) for response notifications.
4. Enroll in the AWS Marketplace Commerce Analytics Service.
5. (Recommended) Make a test call to the service using the [AWS Command Line Interface \(AWS CLI\)](#).

For instructions, see the [Onboarding guide \(p. 15\)](#).

Note

The IAM permissions required for product support connection are different from those required for commerce analytics service. Product support connection requires that the IAM user can call the `marketplacecommerceanalytics:StartSupportDataExport` action.

You can allow an IAM user to call the `StartSupportDataExport` action by using an IAM permission policy.

Example

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "marketplacecommerceanalytics:StartSupportDataExport",
      "Resource": "*"
    }
  ]
}
```

For more information about this feature, see [Product Support Connection \(p. 25\)](#).

Amazon SQS permissions

As part of the SaaS product publication process, AWS Marketplace provides you an Amazon SNS topic you can use to receive notifications if a customer's subscription or entitlement status changes. You

can configure one or more Amazon SQS queues to the topic so that the queues can take action on the notification. For example, if a customer adds more storage to the subscription they have to your SaaS product, the Amazon SNS topic can send a message to an Amazon SQS queue that starts a process to automatically increase the storage capacity available to that customer.

When you subscribe the Amazon Simple Queue Service (Amazon SQS) queue to the provided Amazon SNS topic, permissions are automatically added to allow the topic to publish messages to the queue. However, you still need an IAM policy for granting the AWS Marketplace Metering and Entitlement Service API user access to the queue. This can be applied to the same user if the services run with the same credentials. Create a policy with the following contents and attach it to your IAM user or role.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "sqs:ReceiveMessage", "sqs:DeleteMessage", "sqs:GetQueueAttributes",
"sqs:GetQueueUrl"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:sqs:REGION_HERE:XXXXXXXXXXXX:NAME_HERE"
    }
  ]
}
```

Note

The Resource field is the Amazon Resource Name (ARN) of your Amazon SQS queue.

For more information on message notification and queuing for your SaaS products, see [the section called “Subscribing an SQS queue to the SNS topic” \(p. 236\)](#) and [the section called “Accessing the AWS Marketplace Metering and Entitlement Service APIs” \(p. 236\)](#).

AWS Marketplace metering and entitlement API permissions

Software as a service (SaaS) products, AMI products, and container products can use the AWS Marketplace Metering and Entitlement Service API. Each type requires different IAM permissions. For your product or products, you meter for all usage, and customers are billed by AWS based on the metering records that you provide. To enable the integration required to provide AWS Marketplace your metering records, the service account that the integration is running under needs a constrained IAM policy to enable access. Attach the policy for the product type you are sending metering information for to the IAM user or role that you're using for the integration.

IAM policy for SaaS products

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "aws-marketplace:ResolveCustomer",
        "aws-marketplace:BatchMeterUsage",
        "aws-marketplace:GetEntitlements"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

```
}  
  ]  
}
```

Note

The first permission is required for all SaaS integrations. The second and third permissions are needed for the AWS Marketplace metering service API and the AWS Marketplace entitlement service API, respectively.

IAM policy for AMI products

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Action": [  
        "aws-marketplace:MeterUsage"  
      ],  
      "Effect": "Allow",  
      "Resource": "*"   
    }  
  ]  
}
```

IAM policy for container products

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Action": [  
        "aws-marketplace:RegisterUsage"  
      ],  
      "Effect": "Allow",  
      "Resource": "*"   
    }  
  ]  
}
```

For more information about creating IAM users, see [Creating an IAM User in Your AWS Account](#) in the *IAM User Guide*. For more information about creating and assigning policies, see [Changing Permissions for an IAM User](#).

This policy grants access to the APIs for the IAM role or user that you attach the policy to. For more information on how to enable role assumption by another account for these API calls, see [How to Best Architect Your AWS Marketplace SaaS Subscription Across Multiple AWS Accounts](#) at the AWS Partner Network (APN) Blog.

Logging AWS Marketplace API calls with AWS CloudTrail

AWS Marketplace is integrated with CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in AWS Marketplace. CloudTrail captures API calls for AWS Marketplace as events. The calls captured include calls from the AWS Marketplace console and code calls to the AWS Marketplace API operations.

CloudTrail is enabled on your AWS account when you create the account. When supported event activity occurs in AWS Marketplace, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your account.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or AWS Identity and Access Management (IAM) user credentials.
- Whether the request was made with temporary security credentials for a role or a federated user.
- Whether the request was made by another AWS service.

AWS Marketplace supports logging the `BatchMeterUsage` operation as events in CloudTrail log files.

AWS Marketplace log file entry examples

Example: BatchMeterUsage

The following example shows a CloudTrail log entry that demonstrates the `BatchMeterUsage` action from the AWS Marketplace Metering Service.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::123456789012:user/Alice",
    "accountId": "123456789012",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "userName": "Alice"
  },
  "eventTime": "2018-04-19T16:32:51Z",
  "eventSource": "metering-marketplace.amazonaws.com",
  "eventName": "BatchMeterUsage",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.0.2/24",
  "userAgent": "Coral/Netty14",
  "requestParameters": {
    "usageRecords": [
      {
        "dimension": "Dimension1",
        "timestamp": "Apr 19, 2018 4:32:50 PM",
        "customerIdentifier": "customer1",
        "quantity": 1
      }
    ],
    "productCode": "EXAMPLE_proCode"
  },
  "responseElements": {
    "results": [
      {
        "usageRecord": {
          "dimension": "Dimension1",
          "timestamp": "Apr 19, 2018 4:32:50 PM",
          "customerIdentifier": "customer1",
          "quantity": 1
        },
        "meteringRecordId": "bEXAMPLE-98f0-4e90-8bd2-bf0EXAMPLE1e",
        "status": "Success"
      }
    ]
  }
}
```

```
    ],  
    "unprocessedRecords": [ ]  
  },  
  "requestID": "dEXAMPLE-251d-11e7-8d11-1f3EXAMPLE8b",  
  "eventID": "cEXAMPLE-e6c2-465d-b47f-150EXAMPLE97",  
  "readOnly": false,  
  "eventType": "AwsApiCall",  
  "recipientAccountId": "123456789012"  
}  
]  
}
```

Example: RegisterUsage for containers

The following example shows a CloudTrail log entry that demonstrates the RegisterUsage action from the AWS Marketplace Metering Service.

```
{  
  "eventVersion": "1.05",  
  "userIdentity": {  
    "type": "AssumedRole",  
    "principalId": "EX_PRINCIPAL_ID:botocore-session-1111111111",  
    "arn": "arn:aws:sts::123456789012:assumed-role/Alice/botocore-session-1111111111",  
    "accountId": "123456789012",  
    "accessKeyId": "EXAMPLE_KEY_ID",  
    "sessionContext": {  
      "sessionIssuer": {  
        "type": "Role",  
        "principalId": "EX_PRINCIPAL_ID",  
        "arn": "arn:aws:iam::123456789012:role/Alice",  
        "accountId": "123456789012",  
        "userName": "Alice"  
      },  
      "webIdFederationData": {  
        "federatedProvider": "arn:aws:iam::123456789012:oidc-provider/oidc.eks.us-east-1.amazonaws.com/id/EXAMPLEFA1C58F08CDB049167EXAMPLE",  
        "attributes": {}  
      },  
      "attributes": {  
        "mfaAuthenticated": "false",  
        "creationDate": "2020-07-23T02:19:34Z"  
      }  
    }  
  },  
  "eventTime": "2020-07-23T02:19:46Z",  
  "eventSource": "metering-marketplace.amazonaws.com",  
  "eventName": "RegisterUsage",  
  "awsRegion": "us-east-1",  
  "sourceIPAddress": "1.2.3.4",  
  "userAgent": "aws-cli/1.18.103 Python/3.8.2 Linux/4.14.181-142.260.amzn2.x86_64  
botocore/1.17.26",  
  "requestParameters": {  
    "productCode": "EXAMPLE_proCode",  
    "publicKeyVersion": 1  
  },  
  "responseElements": {  
    "signature": "eyJhbGciOiJIUzI1Ni..."  
  },  
  "requestID": "dEXAMPLE-251d-11e7-8d11-1f3EXAMPLE8b",  
  "eventID": "cEXAMPLE-e6c2-465d-b47f-150EXAMPLE97",  
  "eventType": "AwsApiCall",  
  "recipientAccountId": "123456789012"  
}
```

Example: MeterUsage for containers on Amazon EKS

The following example shows a CloudTrail log entry that demonstrates the MeterUsage action from the AWS Marketplace Metering Service for containers on Amazon EKS.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EX_PRINCIPAL_ID:botocore-session-1111111111",
    "arn": "arn:aws:sts::123456789012:assumed-role/Alice/botocore-session-1111111111",
    "accountId": "123456789012",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:role/Alice",
        "accountId": "123456789012",
        "userName": "Alice"
      },
      "webIdFederationData": {
        "federatedProvider": "arn:aws:iam::123456789012:oidc-provider/oidc.eks.us-east-1.amazonaws.com/id/EXAMPLEFA1C58F08CDB049167EXAMPLE",
        "attributes": {}
      },
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2020-07-23T01:03:26Z"
      }
    }
  },
  "eventTime": "2020-07-23T01:38:13Z",
  "eventSource": "metering-marketplace.amazonaws.com",
  "eventName": "MeterUsage",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "1.2.3.4",
  "userAgent": "aws-cli/1.18.103 Python/3.8.2 Linux/4.14.181-142.260.amzn2.x86_64 botocore/1.17.26",
  "requestParameters": {
    "timestamp": "Jul 23, 2020 1:35:44 AM",
    "usageQuantity": 1,
    "usageDimension": "Dimension1",
    "productCode": "EXAMPLE_proCode"
  },
  "responseElements": {
    "meteringRecordId": "bEXAMPLE-98f0-4e90-8bd2-bf0EXAMPLE1e"
  },
  "requestID": "dEXAMPLE-251d-11e7-8d11-1f3EXAMPLE8b",
  "eventID": "cEXAMPLE-e6c2-465d-b47f-150EXAMPLE97",
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
}
```

Example: MeterUsage on AMIs

The following example shows a CloudTrail log entry that demonstrates the MeterUsage action from the AWS Marketplace Metering Service for AMIs.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
```

```
    "type": "AssumedRole",
    "principalId": "EX_PRINCIPAL_ID:i-exampled859aa775c",
    "arn": "arn:aws:sts::123456789012:assumed-role/Alice/i-exampled859aa775c",
    "accountId": "123456789012",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:role/Alice",
        "accountId": "123456789012",
        "userName": "Alice"
      },
      "webIdFederationData": {},
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2020-07-10T23:05:20Z"
      },
      "ec2RoleDelivery": "1.0"
    }
  },
  "eventTime": "2020-07-10T23:06:42Z",
  "eventSource": "metering-marketplace.amazonaws.com",
  "eventName": "MeterUsage",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "1.2.3.4",
  "userAgent": "aws-cli/1.16.102 Python/2.7.16 Linux/4.14.133-113.112.amzn2.x86_64
botocore/1.12.92",
  "requestParameters": {
    "productCode": "EXAMPLE_proCode",
    "timestamp": "Jul 10, 2020 11:06:41 PM",
    "usageDimension": "Dimension1",
    "usageQuantity": 1,
    "dryRun": false
  },
  "responseElements": {
    "meteringRecordId": "bEXAMPLE-98f0-4e90-8bd2-bf0EXAMPLE1e"
  },
  "requestID": "dEXAMPLE-251d-11e7-8d11-1f3EXAMPLE8b",
  "eventID": "cEXAMPLE-e6c2-465d-b47f-150EXAMPLE97",
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
}
```

Related Topics

For more information, see the following topics in the *AWS CloudTrail User Guide*:

- [Overview for Creating a Trail](#)
- [AWS Service Integrations with CloudTrail Logs](#)
- [Configuring Amazon SNS Notifications for CloudTrail](#)
- [Receiving CloudTrail Log Files from Multiple Regions and Receiving CloudTrail Log Files from Multiple Accounts](#)
- [CloudTrail userIdentity Element.](#)

Document history

The following table describes the documentation for this release of the *AWS Marketplace Seller Guide*.

For notification about updates to this documentation, you can subscribe to the RSS feed.

| update-history-change | update-history-description | update-history-date |
|---|---|---------------------|
| Added a video to Professional services products | Updated the Professional services products landing page with a video containing details on how to manage service products | February 24, 2022 |
| New topic about deploying a serverless SaaS integration solution | New information added for integrating serverless SaaS deployment, including a link to AWS Quick Start for a reference on deployment steps. | February 15, 2022 |
| Minimal updates to Container-based requirements and AMI sections | Minimal updates to policies in Container-based requirements and removed incorrect information for AMI pricing contracts | February 14, 2022 |
| Container versioning update | Documentation-only update to clarify how to push container images and other artifacts to repositories. | February 10, 2022 |
| Update to ResolveCustomer code example for SaaS products | The ResolveCustomer code example for SaaS products has been updated to include CustomerAWSAccountID. | February 3, 2022 |
| Added documentation for integrating AWS License Manager with AWS Marketplace for Containers Anywhere products | Documentation-only update to add detailed guidance on adding contract-pricing to your AWS Marketplace for Containers Anywhere products by integrating with License Manager. | February 1, 2022 |
| SaaS SNS notifications update | Documentation-only update to clarify SaaS notification messages. | January 25, 2022 |
| Ability for sellers to transact with EMEA-based buyers through Amazon Web Services EMEA SARL | Eligible AWS Marketplace sellers can now transact with customers whose AWS accounts are based in countries and territories in Europe, the Middle East, and Africa (EMEA) through Amazon Web Services EMEA SARL. | January 7, 2022 |

| | | |
|--|---|-------------------|
| Added documentation for creating delivery options for container-based products with a Helm chart delivery method | Sellers can now provide delivery options with a Helm chart delivery method. Buyers can use these delivery options to launch a container-based application by installing a seller-provided Helm chart in their launch environment. When providing a Helm chart delivery method, sellers can enable QuickLaunch for buyers. QuickLaunch is a feature that buyers can use to use AWS CloudFormation to quickly create a new Amazon EKS cluster and launch a container-based application on it. | November 29, 2021 |
| Update to existing policies | Security policies for AWS Marketplace sellers have been updated. | November 22, 2021 |
| Contract pricing for AMI and Container-based products | Independent software vendors (ISVs) can now list a new AMI-based product or a Container-based product and offer upfront contract pricing for buyers. | November 17, 2021 |
| Vendor metered tagging | Documentation-only update for vendor metered tagging, including code examples. | November 11, 2021 |
| Amazon Simple Notification Service for AMI or Container products | Independent software vendors (ISVs) can receive notifications when customers subscribe to or unsubscribe from AMI or Container products through the Amazon Simple Notification Service. | November 10, 2021 |
| New seller permissions | AWS Marketplace added new permissions to access the Offers and Partners tabs in the AWS Marketplace Management Portal. | November 9, 2021 |
| Ability to deploy endpoints configured for Asynchronous Inference for machine learning products | For machine learning software that expects a higher payload inference than the maximum, or requires processing times that exceed the maximum processing time per invocation, buyers have the ability to deploy endpoints configured for Amazon SageMaker Asynchronous Inference. | November 8, 2021 |

| | | |
|---|---|-------------------|
| Refund policy and approvals | Documentation-only update to clarify the refund policy and move all refund information to one central location in the <i>AWS Marketplace Seller Guide</i> . | August 20, 2021 |
| Select or upload EULA for consulting partner private offers | Independent software vendors can now select or upload an End User License Agreement (EULA) when creating resale opportunities for consulting partners. | August 17, 2021 |
| Custom product dimensions for SaaS contract products | Independent software vendors (ISVs) can now customize SaaS contract product dimensions when creating resale opportunities for consulting partners. | August 17, 2021 |
| AWS Marketplace Field Demonstration Program | Documentation-only update to clarify the requirements for AWS Data Exchange dataset products for the AWS Marketplace Field Demonstration Program. | August 3, 2021 |
| SaaS product guidelines update | Product guidelines for SaaS products have been updated. | July 29, 2021 |
| Container-based product requirements update | Container-based product requirements have been updated. | July 29, 2021 |
| AMI security policy update | Security policies for AMI products have been updated. | July 29, 2021 |
| More eligible jurisdictions | The following are now eligible to become sellers on AWS Marketplace: Hong Kong SAR and Qatar. | June 23, 2021 |
| Data feeds overview | Documentation-only update to give an overview of the structure of the data feeds available to sellers. | June 23, 2021 |
| Updated the machine learning chapter | Documentation-only update to the information about creating and maintaining machine learning products. | May 27, 2021 |
| Self-service updating for container products | Sellers now have a simpler and faster way to update their container-based products through the AWS Marketplace Management Portal. | December 17, 2020 |

| | | |
|---|---|-------------------|
| Professional services | Sellers can now offer professional services to AWS Marketplace buyers. Added the professional services section to the documentation. | December 3, 2020 |
| Self-service updating for AMI products | Sellers now have a simpler and faster way to update their Amazon Machine Image (AMI) based products through the AWS Marketplace Management Portal. | November 23, 2020 |
| More eligible jurisdictions | The following are now eligible to become sellers on AWS Marketplace: Bahrain, Norway, Switzerland, and the United Arab Emirates (UAE). | June 17, 2020 |
| You can offer upgrades and renewals on accepted private offers | For SaaS contract and SaaS contracts with consumption products, you can offer upgrades and renewals using private offers on previously-accepted private offers. | May 28, 2020 |
| More information is available in data feeds | More information from reports is broken down into smaller data feeds to simplify finding and analyzing data. | May 21, 2020 |
| Standardized license terms are now available | You can offer standardized license terms in place of custom EULAs to simplify the contracting process. | April 28, 2020 |
| Australia and New Zealand are eligible jurisdictions | The following are now eligible to become sellers on AWS Marketplace: (i) Permanent residents and citizens of Australia (AU) New Zealand (NZ) or (ii) business entities organized or incorporated in one of those areas. | April 2, 2020 |
| Container products now support custom metering and pricing enhancements | If you want to define your own pricing units and meter that usage to us for billing, integrate with the AWS Marketplace Metering Service's <code>meterUsage</code> action. | November 14, 2019 |
| AWS Marketplace supports data products through AWS Data Exchange | You can now provide data products in AWS Marketplace. | November 13, 2019 |

| | | |
|--|---|--------------------|
| Introducing the AWS Marketplace Catalog API service (p. 364) | The AWS Marketplace Catalog API service provides an API interface for approved sellers to programmatically manage their products. | November 12, 2019 |
| AWS Marketplace supports paid hourly containers | AWS Marketplace now supports paid hourly containers running on Amazon Elastic Kubernetes Service (Amazon EKS). | September 25, 2019 |
| Updated AMI product functionality | You can now deploy AMIs and Lambda functions together using AWS CloudFormation. | September 11, 2019 |
| Added Security section | Consolidated security content under a new Security section. | May 7, 2019 |
| Updated AMI security policies | Updated the security policies for AMI products | April 11, 2019 |
| Added versioning information to Machine Learning Products section (p. 364) | Added content describing product version capability for machine learning products. | March 21, 2019 |
| Added Machine Learning Products section | Added content for publishing machine learning products | November 28, 2018 |
| Added Container-Based Products section | Added content for publishing container-based products. | November 27, 2018 |
| Updated link for submitting seller help request | Changed email address to webform address. | October 22, 2018 |
| Added SaaS contracts with consumption content | Restructured SaaS content and added content to support release of SaaS contracts with consumption features. | October 18, 2018 |
| Added content about flexible payment schedule for private offers | Added content to support release of flexible payment scheduler for private offers content. | October 15, 2018 |
| Updated IAM permissions content | Added content to support of new IAM permission for AMMP read only access. | October 9, 2018 |
| Added content about consulting partner private offers | Added content to support Consulting Partner Private Offers feature release. | October 9, 2018 |
| Added content about private image builds | Added content to support release of Private Image Build for AMIs feature. | August 13, 2018 |
| Added search engine optimization guidance for sellers. | Added guidance for sellers who want to optimize their product for search. | July 3, 2018 |

[Updated link to AWS Marketplace logos](#)

Updated link to point to new AWS Marketplace logos.

June 12, 2018

[Added seller guides](#)

Converted all PDF seller guides to online content.

May 9, 2018

AWS glossary

For the latest AWS terminology, see the [AWS glossary](#) in the *AWS General Reference*.