



このコンテンツは公開から3年以上経過しており内容が古い可能性があります  
最新情報については[サービス別資料](#)もしくはサービスのドキュメントをご確認ください

# [AWS Black Belt Online Seminar]

## EC2 Image Builder

サービスカットシリーズ

Solutions Architect 柳 嘉起

2020/8/25

AWS 公式 Webinar

<https://amzn.to/JPWebinar>



過去資料

<https://amzn.to/JPArchive>



# 自己紹介

## 柳 嘉起 (やなぎ よしき)

エンタープライズソリューション本部  
ソリューションアーキテクト

### 経歴

- 国内SIerでインフラアーキテクトとして勤務
- スポーツ/ギャンブル系のシステムを担当

### 好きなAWSサービス

- Amazon S3
- マネジメント&ガバナンスサービス



# “Design with Ops in Mind”

# AWS Black Belt Online Seminar とは

「サービス別」「ソリューション別」「業種別」のそれぞれのテーマに分かれて、アマゾンウェブ サービス ジャパン株式会社が主催するオンラインセミナーシリーズです。

## 質問を投げることができます！

- 書き込んだ質問は、主催者にしか見えません
- 今後のロードマップに関するご質問は  
お答えできませんのでご了承下さい

- ① 吹き出しをクリック
- ② 質問を入力
- ③ Sendをクリック



Twitter ハッシュタグは以下をご利用ください  
#awsblackbelt

# 内容についての注意点

- 本資料では2020年8月25日現在のサービス内容および価格についてご説明しています。最新の情報はAWS公式ウェブサイト(<http://aws.amazon.com>)にてご確認ください。
- 資料作成には十分注意しておりますが、資料内の価格とAWS公式ウェブサイト記載の価格に相違があった場合、AWS公式ウェブサイトの価格を優先とさせていただきます。
- 価格は税抜表記となっております。日本居住者のお客様には別途消費税をご請求させていただきます。
- AWS does not offer binding price quotes. AWS pricing is publicly available and is subject to change in accordance with the AWS Customer Agreement available at <http://aws.amazon.com/agreement/>. Any pricing information included in this document is provided only as an estimate of usage charges for AWS services based on certain information that you have provided. Monthly charges will be based on your actual use of AWS services, and may vary from the estimates provided.

# アジェンダと想定聴講者

## アジェンダ

1. EC2とAMI
2. EC2 Image Builder 概要
3. EC2 Image Builder 動作画面
4. ユースケース & Tips
5. トラブルシューティングとデバッグ
6. 補足情報
7. まとめ

## 想定聴講者

- EC2 Image Builder を使ったことが無い方
- EC2 Image Builder の使い方/使いどころや、実行時のエラー対処にお悩みの方
- セキュアなサーバーイメージの作成や更新に苦労されている方

# 本日本お伝えしたいこと

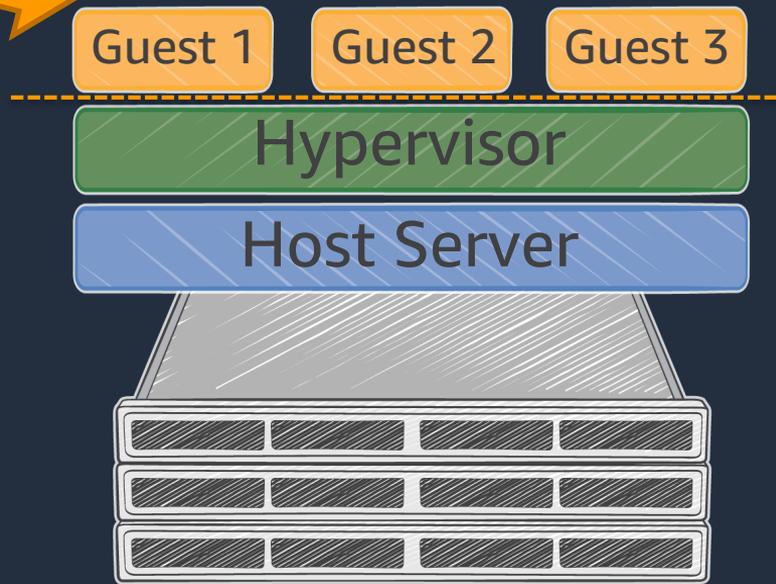
- EC2 Image Builder の概要と使い方
- このサービスによって得られるベネフィット

# EC2とAMI (Amazon Machine Image)

# Amazon EC2

EC2インスタンス

- 数分で起動し、1時間または秒単位の従量課金で利用可能なAWSクラウド上の仮想サーバー (インスタンスによってはベアメタルサーバも選択可)
- サーバーの追加・削除、マシンスペック変更も数分で可能
- 管理者権限(root / Administrator) で利用可能



# EC2 インスタンス起動に必要な設定

インスタンスタイプ

...

仮想サーバのスペック  
(CPU、メモリ、NW)

+

AMI  
(Amazon Machine Image)

...

作成する仮想サーバの OS イメージ

+

その他の設定

...

- アベイラビリティゾーン/サブネット
  - セキュリティグループ
- など

# EC2 インスタンス起動に必要な設定

インスタンスタイプ

...

仮想サーバのスペック  
(CPU、メモリ、NW)

+

AMI  
(Amazon Machine Image)

...

作成する仮想サーバの OS イメージ

+

その他の設定

...

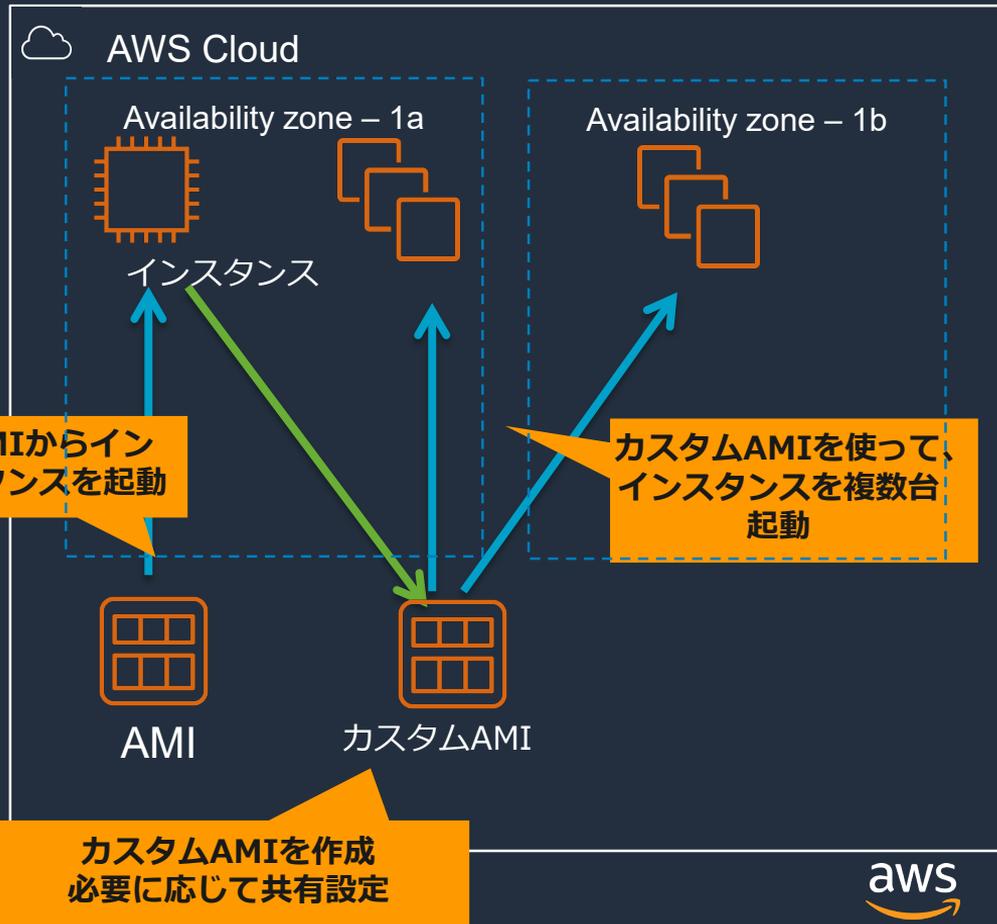
- アベイラビリティゾーン/サブネット
  - セキュリティグループ
- など

# AMI (Amazon Machine Image) とは

AMIは事前構成されたOSイメージで、EC2インスタンスを作成するために必要なOSとソフトウェアを含んでいる。

自由にカスタムAMIを作成可能

- 作成したAMIは別アカウントと共有可能
- カスタムAMIから何台でもEC2インスタンスを起動可能
- 別リージョンへのコピーも可能



# ゴールデンイメージ

サービスが迅速に提供できるように、また組織内で共通利用できるように、事前に構成された環境をマシンイメージ化し、**テンプレートとして使える**ようにしたもの



ゴールデンイメージにどのスタックまで含めるかは、トレードオフがあるため戦略的に考える

- ・ 初期イメージに近いほど、ゴールデンイメージをデプロイしてからReadyな状態にするまでの時間がかかる
- ・ カスタマイズ量が多いほど、ゴールデンイメージを作成する時間はかかる
- ・ アプリケーションやミドルウェアの更新頻度が高いのであれば、イメージに含めるスタックが増えるのに比例してイメージを作り直す頻度も高くなる

# ゴールデンイメージ

サービスが迅速に提供できるように、もしくは組織内で共通利用できるように、ある程度（もしくは完全に）準備された環境をマシンイメージ化し、**テンプレートとして使えるようにしたもの**



ゴールデンイメージにどのスタックまで含めるかは、トレードオフがあるため戦略的に考える

ゴールデンイメージに含むものが増えるほど  
作成時間と作成頻度が問題になる

# ゴールデンイメージ

サービスが迅速に提供できるように、もしくは組織内で共通利用できるように、ある程度（もしくは完全に）準備された環境をマシンイメージ化し、**テンプレートとして使えるようにしたもの**



ゴールデンイメージにどのスタックまで含めるかは、トレードオフがあるため戦略的に考える

## ゴールデンイメージの作成をCI/CD化する

# ゴールデンイメージを定期的につくる際の課題



ゴールデンイメージのCI/CDには、ビルドオートメーションの高いスキルが求められる



既存の自動化ツールやスクリプトを使用するお客様は、デプロイ環境を維持運用する必要がある



ソフトウェアの更新のつど、新しいイメージを手動で構築およびテストしなければならない



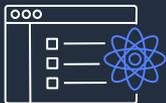
事前に問題を検出できず、本番運転で問題が発生する



# EC2 Image Builder

OS イメージのビルド / カスタマイズ / デプロイを容易にする  
完全マネージド型サービス

# EC2 Image Builder の機能



OSイメージのビルド / カスタマイズ / デプロイを行う  
自動化されたパイプラインを作成



セキュリティとコンプライアンスを満たした最新のイメージを作成



AWS VM Import/Export (VMIE) と組み合わせることで、  
AWS環境だけでなくオンプレミスで使用できる



イメージを実稼働環境で使用する前に検証する仕組みの組み込み



リビジョン管理と  
AWS アカウント間での自動化スクリプト、レシピ、および イメージの共有

# EC2 Image Builder の用語



EC2 Image Builder

- コンポーネント
- イメージレシピ
- イメージパイプライン

# EC2 Image Builder コンポーネント

コンポーネントは「ビルドコンポーネント」と「テストコンポーネント」の2種類があります。

## ビルドコンポーネント

ソフトウェアパッケージのダウンロード、インストール、および構成の手順を定義するドキュメント

## テストコンポーネント

ソフトウェアパッケージで実行するテストを定義するドキュメント

- ・コンポーネントはPhasesとStepsで構成

### Phases:

- ・ Stepをまとめたもの
- ・ ドキュメント内で一意に名づける
- ・ Image Builderでは build/validate/testが使える

### Steps:

- ・ 個々の作業単位で、実行するアクションを定義
- ・ ひとつのPhases内では一意に名づける
- ・ 上から順次実行される
- ・ Stepのinputとoutputを後続のStepのinputに使える
- ・ 各ステップでアクションモジュールを使い、終了コードを返す

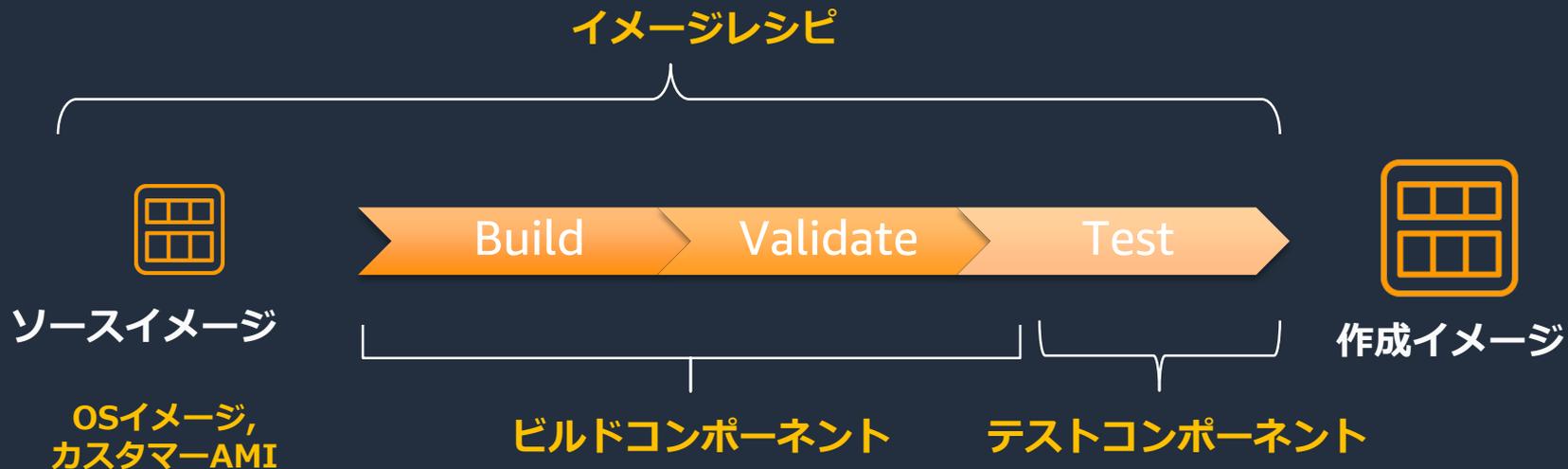
- ・ YAMLで記述されたドキュメント

<https://docs.aws.amazon.com/imagebuilder/latest/userguide/image-builder-component-manager.html>

```
phases:
-
  name: 'build'
  steps:
  -
    name: SampleS3Download
    action: S3Download
    timeoutSeconds: 60
    onFailure: Abort
    maxAttempts: 3
    inputs:
    -
      source: 's3://sample-bucket/sample1.ps1'
      destination: 'C:\Temp\sample1.ps1'
    -
      source: 's3://sample-bucket/sample2.ps1'
      destination: 'C:\Temp\sample2.ps1'
```

# EC2 Image Builder イメージレシピ

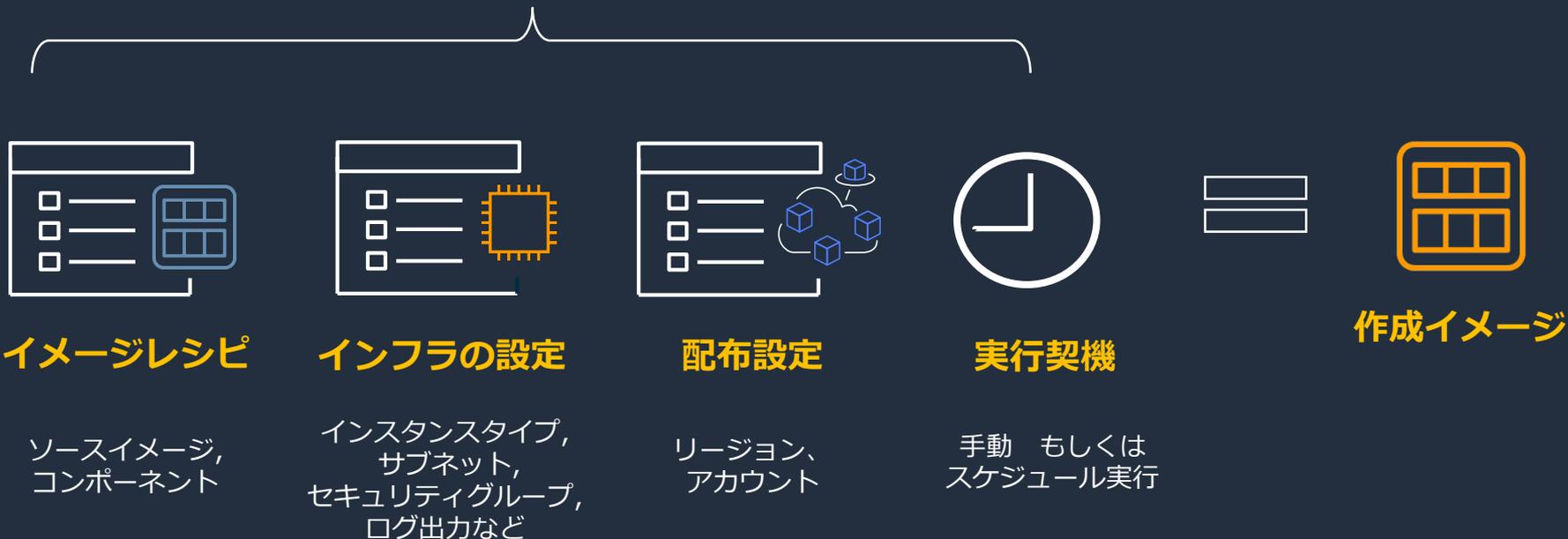
イメージレシピは作成するイメージの設計書です。  
ソースイメージと、それに適用するコンポーネントを定義します。



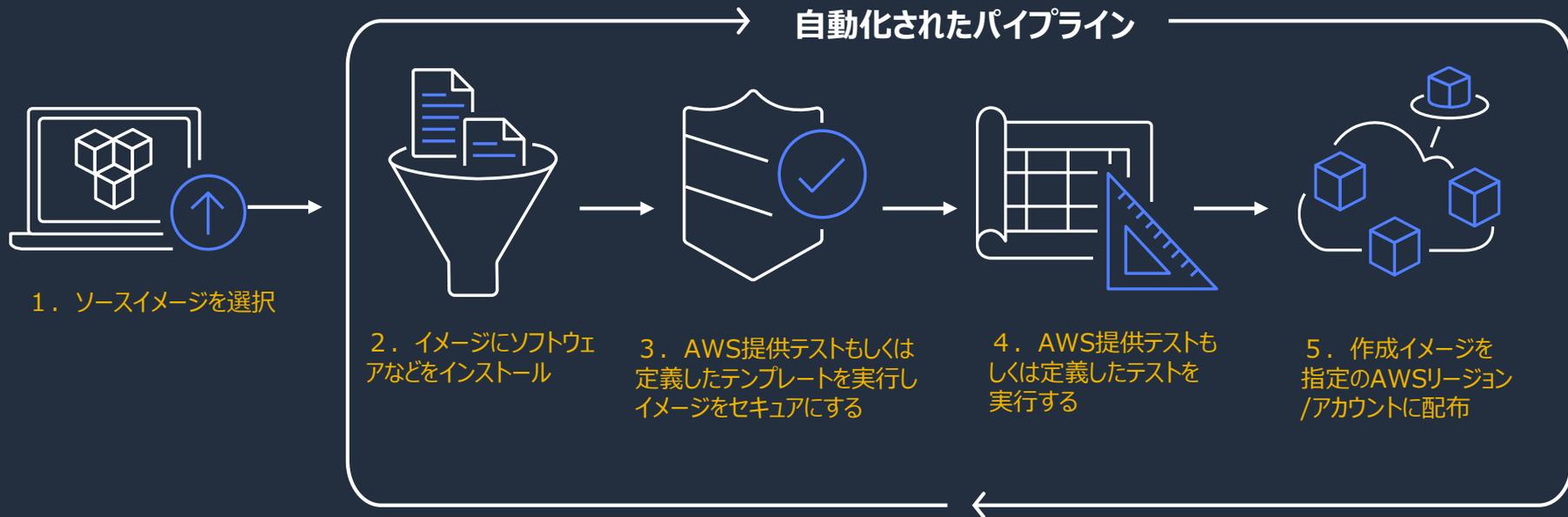
# EC2 Image Builder イメージパイプライン

イメージパイプラインは、イメージ作成を自動化するための設定です。  
レシピ、インフラの設定、配布設定と、それを実行する契機をパイプラインとして定義します。

## パイプライン



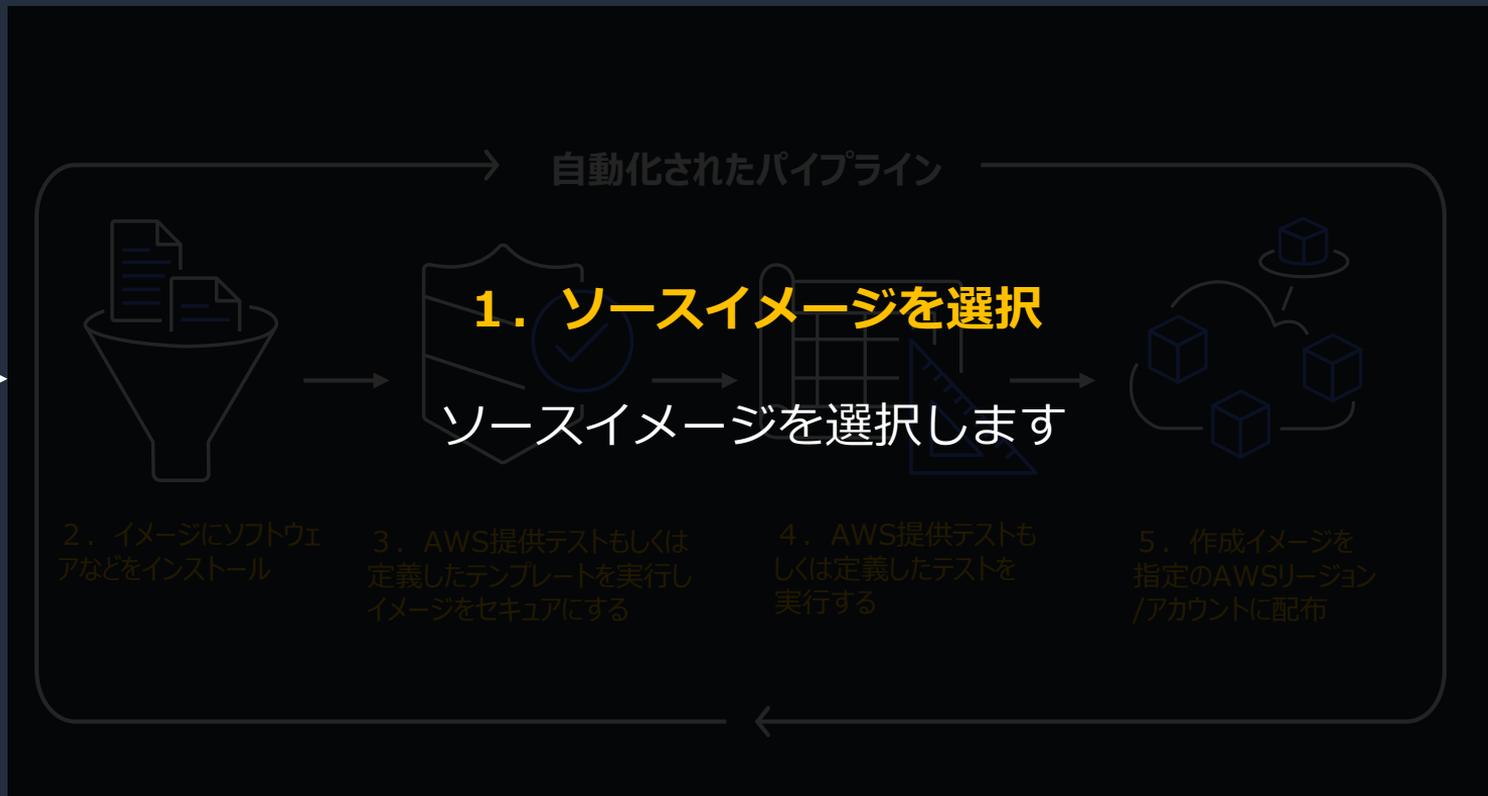
# EC2 Image Builder の動作



# EC2 Image Builder の動作



1. ソースイメージを選択



# EC2 Image Builder の動作



1. ソースイメージを選択



2. イメージにソフトウェアなどをインストール

## 2. イメージにソフトウェアなどをインストール

自動化されたパイプライン

イメージでインストールされるソフトウェアのカスタマイズ例

- 1/ ソフトウェア・ミドルウェア
- 2/ アプリケーション
- 3/ OS のアップデート
- 4/ パッチ

3. AWS提供テストもしくは定義したテストを実行しイメージをセキュアにする

4. AWS提供テストもしくは定義したテストを実行する

5. 作成イメージを指定のAWSリージョン/アカウントに配布

# EC2 Image Builder の動作

## 3. AWS提供テストもしくは定義したテンプレートを実行しイメージをセキュアにする

1. ソースイメージを選択

2. イメージにソフトウェアなどをインストール

3. AWS提供テストもしくは定義したテンプレートを実行しイメージをセキュアにする



## イメージのセキュア化の例

- 1/ セキュリティパッチの適用
- 2/ 強力なパスワードを要求
- 3/ ディスク全体の暗号化を実施
- 4/ 不要なポートを閉じる
- 5/ ソフトウェアFWを有効にする
- 6/ ログ出力と監査を有効にする

# EC2 Image Builder の動作

## 4. AWS提供テストもしくは定義したテストを実行する

AWS が提供するテストとユーザー独自のテスト例

- 1/ AMI がブートすることを確認するテスト
- 2/ サンプルアプリケーションが動作するかどうかのテスト
- 3/ 特定のパッチが適用されているかどうかのテスト
- 4/ セキュリティポリシーのテスト

自動化されたパイプライン



4. AWS提供テストもしくは定義したテストを実行する



5. 作成イメージを指定のAWSリージョン/アカウントに配布

# EC2 Image Builder の動作

## 5. 作成イメージを指定のAWSリージョン/アカウントに配布

テストに合格した後、作成したイメージを選択したAWSリージョン/アカウントに配布します。

1. ソースイメージを選択

2. イメージにソフトウェアなどをインストール

3. AWS提供テストもしくは定義したテンプレートを実行しイメージをセキュアにする

4. AWS提供テストもしくは定義したテストを実行する

5. 作成イメージを指定のAWSリージョン/アカウントに配布

自動化されたパイプライン



# EC2 Image Builder パイプライン実行時の内部動作

1	レシピで定義されたソースイメージとインスタンスタイプから EC2インスタンスが起動
2	EC2インスタンスにレシピで指定したビルドコンポーネントがダウンロードされ、 EC2インスタンス上で実行
3	EC2インスタンスが停止しAMIが作られ、その後インスタンスは <b>terminate</b>
4	作成されたAMIからEC2インスタンスを起動
5	EC2インスタンスにレシピで指定したテストコンポーネントがダウンロードされ、 EC2インスタンス上で実行される。テスト後インスタンスは <b>terminate</b>
6	上記の工程が正常に終了すると、 Output images の Status が「Available」になり完了

実行制御はAWS Systems Manager Automation によって行われている

# EC2 Image Builder 対応OSと出力フォーマット

## • 対応OS

- Amazon Linux 2
- Windows Server 2012、2016、2019、version 1909
- Ubuntu Server 16、18
- Red Hat Enterprise Linux (RHEL) 7、8
- Cent OS 7、8※
- SUSE Linux Enterprise Server (SLES) 15

※CentOS 8はAWS MarketplaceからパブリックAMIとして利用できないため、AWS VM Import/Export (VMIE) を使用してご自身のCentOS 8 AMIを持ち込む必要があります。

## • 出力フォーマット

- AMI
- VHDX、VMDK、OVF

# EC2 Image Builder 前提条件

イメージレシピの内容に応じて、IAMロールが必要になります。  
最低限必要なIAMロールのポリシーは以下です。

- **EC2InstanceProfileForImageBuilder** (コンポーネントのビルドとテスト実行に必要なため)
- **AmazonSSMManagedInstanceCore** (パイプラインは、SSM Automation により管理されるため)

その他レシピ内で利用するサービスに応じてIAMロールの設定が必要になります。  
必要なIAMロールは以下をご確認ください。

※ログ出力にも権限設定が必要です (後述)

<https://docs.aws.amazon.com/imagebuilder/latest/userguide/image-builder-service-linked-role.html>

EC2 Image Builderは、以下のインタフェースから利用可能です

- マネジメントコンソール
- AWS CLI
- AWS Tools for SDK
- AWS CloudFormation

# EC2 Image Builder ログ

- ログ出力

- (1) Amazon S3 (指定した場所)
- (2) Amazon CloudWatch Logs (下記)

- CloudWatch Logs のストリーム先

```
LogGroup : "/aws/imagebuilder/<ImageName>
```

```
LogStream : <ImageVersion>/<ImageBuildVersion>["x.x.x/x"]
```

Amazon CloudWatch Logs へのログ出力は、  
デフォルトで“有効”になっています。  
停止したいときはIAMロールから、下記の権限を削除します。

```
"Statement": [  
  {  
    "Effect": "Allow",  
    "Action": [  
      "logs:CreateLogStream",  
      "logs:CreateLogGroup",  
      "logs:PutLogEvents"  
    ],  
    "Resource": "arn:aws:logs:*:*:log-group:/aws/imagebuilder/*"  
  }  
]
```

## EC2 Image Builder ログ ※S3にログを出力したいとき

Amazon S3 へのログ出力は、権限設定が必要です。  
IAMロールに下記の権限例を参考に付与してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::bucket-name/*"
    }
  ]
}
```

# レシピ内で利用するサービスに応じた権限設定の例

例えばAWSが提供するテストコンポーネントの中に、"ebs-volume-usage-test-linux"があります。同コンポーネントは以下の工程でテストを行います。

- (1) EBS ボリュームを作成
- (2) インスタンスにアタッチ
- (3) ボリューム上にテンポラリファイルを作成
- (4) ボリュームをデタッチ
- (5) ボリュームを再アタッチ
- (6) ファイルが存在することを検証
- (7) ボリュームをデタッチして削除

上記の操作を行う上で、実行権限として

**ec2:AttachVolume, ec2:CreateTags, ec2:CreateVolume, ec2>DeleteVolume, ec2:DescribeVolumes, ec2:DetachVolume** が設定されたIAMポリシーが必要になります。

名前	概要	OS	種類
ebs-volume-usage-test-linux  Version 1.0.1	The EBS volume usage test creates an EBS volume and attaches it to the instance. It creates a temporary file on the volume and detaches the volume. It reattaches the volume and validates that the file exists. It then detaches and deletes the volume. To perform this test, an IAM policy with the following actions is required: ec2:AttachVolume, ec2:CreateTags, ec2:CreateVolume, ec2>DeleteVolume, ec2:DescribeVolumes, and ec2:DetachVolume.	Linux	TEST

# EC2 Image Builder コストについて

- ✓ EC2 Image Builder自体は無料
- ✓ イメージの作成、保存、共有するために使用した分のAWSリソース料金が別途課金

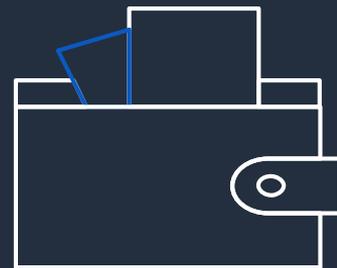
例：

EC2インスタンス料金

Amazon S3 / Amazon CloudWatch Logs

EBSスナップショット

Amazon Inspector



# EC2 Image Builder 動作画面

Step 1

**Define recipe**

---

Step 2

Configure pipeline

---

Step 3

Configure additional settings -  
*optional*

---

Step 4

Review and create

**Step 1 : レシピの定義**

**Step 2 : パイプラインの構成**

**Step 3 : 追加設定 (オプション)**

**Step 4 : 確認・作成**

# Step 1: レシピ定義 1. ソースイメージを選択

## Define recipe

A recipe specifies the activities needed to make changes to the source image. A recipe cannot be modified once

### Source image [Info](#)

#### Image operating system (OS)

Image Builder supports Amazon Linux, Windows, Ubuntu, CentOS, RHEL, and SLES.

Amazon Linux  
Amazon Linux 2



Windows  
Windows Server 1909,  
2004, 2012R2, 2016, and  
2019



Ubuntu  
Ubuntu 16, 18 and 20



CentOS  
CentOS 7 and 8



Red Hat Enterprise  
Linux (RHEL)  
RHEL 7 and 8



SUSE Linux Enterprise  
Server (SLES)  
SLES 15



- **AWS提供AMI**  
(managed images)

- **自分で作成したAMI**
- **共有されたAMI**  
(カスタムAMI)

上記からソースイメージを指定

# Step 1: レシピ定義

## 1. ソースイメージを選択

### Storage (Volumes)

The storage device settings for your pipeline.

▼ EBS Volume 1 (AMI Root)

Device name	Snapshot - Optional	Volume type
<input type="text" value="/dev/xvda"/>	<input type="text" value="snap-081e8433c7"/>	<input type="text" value="General Purp..."/>
Size (GiB)	IOPS	Encryption (KMS Alias)
<input type="text" value="8"/>	<input type="text" value="100"/>	<input type="text" value="Do not enable"/>

Delete on termination

### ストレージ設定

- Device name
- Snapshot
- Volume type
- Size(GiB)
- IOPS
- Encryption(KMS Alias)

# Step 1: レシピ定義 2. ビルドコンポーネントを指定

### Build components [Info](#)

Build components contain software, settings, and configurations to be installed or applied. Build components are included in the recipe and are run during the process of building custom images.

[Create build component](#)

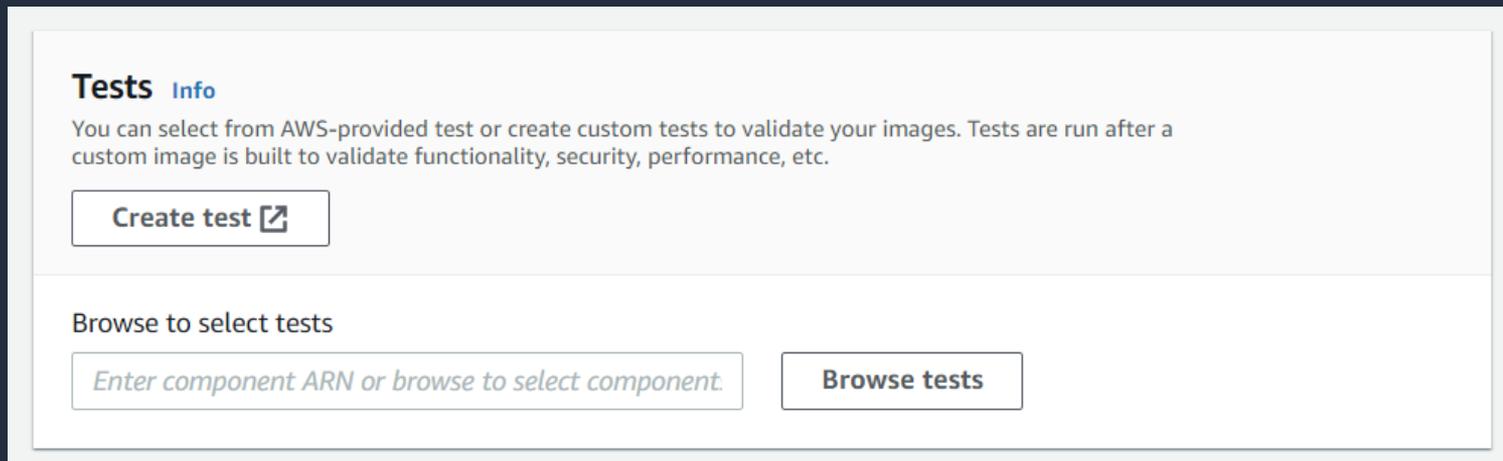
Browse to select build components

[Browse build components](#)

- イメージに適用するビルドコンポーネント（インストールすべき追加ソフトウェアを定義したもの）を指定する
- **“Create build component”** で自ら新規に定義（YAML記述）すること  
も、**“Browse build components”** であらかじめ作成されたものを適用することも可能
- 複数のビルドコンポーネントを指定可能

コンポーネントは選択した順にインストールされます。  
コンポーネントを選択した後は、コンポーネントを並べ替えることはできません。

# Step 1: レシピ定義 3. 実行するテストを構成



- テストコンポーネント（イメージの検証のために実行されるテスト）を定義する
- “**Create test**” で自ら新規に定義（YAML記述）することも、“**Browse tests**” であらかじめ作成されたものを使用することも可能
- 複数のテストコンポーネントを指定可能

# Step 2 :パイプラインの構成

## Configure pipeline

Define the pipeline infrastructure and build schedule.

### Pipeline details

Name

Custom naming allowed. Max 70 characters. Letters, numbers, space, -, and \_ allowed. Don't use previously used names.

Description

Custom description allowed, Maximum 1024 characters.

IAM role [Info](#)

Select a role to associate with the instance profile. Image Builder spins up EC2 instances in your account to customize images and run validation tests. This setting specifies the IAM role used for this purpose.

Choose IAM role ▼



[Create new instance profile role](#)

Enhanced metadata collection

EC2 Image Builder uses AWS Systems Manager Inventory to collect additional information about the images you create. This information is checked before the creation of an image to ensure compatibility between components and images.

Enable enhanced metadata collection

Deselecting this field disables the collection of additional information from images created for this pipeline.

- パイプラインの名前や、Descriptionを入力
- IAMロールを指定

- **EC2InstanceProfileForImageBuilder**
- **AmazonSSMManagedInstanceCore**

をもとにIAMロールを設定する。

前者はコンポーネントのビルドとテスト実行に必要な十分な権限を持つポリシーで、後者はインスタンス操作におけるアクション（EC2 Image BuilderはSSMにて実行）を許可する権限を持つポリシーである。

※ロギングを行う場合はs3 : PutObjectが必要

<https://docs.aws.amazon.com/imagebuilder/latest/userguide/security-iam.html>

# Step 2 :パイプラインの構成

## Build schedule

Select the frequency at which the image pipeline produces new images with the specified customization. All schedules use UTC time zone and the minimum precision is 1 hour.

**Manual**

Manually run the pipeline by clicking "Run pipeline" on pipeline detail page.

**Schedule builder**

Automatically run the pipeline using a job scheduler.

**CRON expression**

Automatically run the pipeline using a syntax that specifies the time and intervals to run it.

- **パイプラインの実行契機を指定する**
  - **Manual (手動)**
  - **Schedule builder (日/週/月と時刻)**
  - **CRON expression (cron書式)**

# Step 2 :パイプラインの構成

- ビルドの実行契機を指定する
  - Manual（手動）
  - **Schedule builder**（日/週/月と時刻）
  - CRON expression（cron書式）

**Build schedule**

Select the frequency at which the image pipeline produces new images with the specified customization and validation with tests in the recipe. All schedules use UTC time zone and the minimum precision is 1 hour.

Manual  
Manually run the pipeline by clicking "Run pipeline" on pipeline detail page.

**Schedule builder**  
Automatically run the pipeline using a job scheduler.

CRON expression  
Automatically run the pipeline using a syntax that specifies the time and intervals to run it.

Run pipeline every  on  at  UTC

**Infrastructure settings - optional**  
Image Builder launches EC2 Instances in your account to customize in... validation tests. These settings specify infrastructure details for instar...

Instance type [Info](#)  
Select one or more instance types used to customize your image. We... specific defaults if you do not provide an entry.

[EC2 default instance types](#)

• 毎日→何時に実行するか  
• 毎週→何曜日の何時に実行するか  
• 毎月→何日の何時に実行するか  
を指定する。複数指定は不可

時刻がUTC（協定世界時 Coordinated Universal Time）であることに注意

# Step 2 :パイプラインの構成

- ビルドの実行契機を指定する
  - Manual（手動）
  - Schedule builder（日/週/月と時刻）
  - **CRON expression（cron書式）**

**Build schedule**

Select the frequency at which the image pipeline produces new images with the specified customization and validation with tests in the recipe. All schedules use UTC time zone and the minimum precision is 1 hour.

Manual  
Manually run the pipeline by clicking "Run pipeline" on pipeline detail page.

Schedule builder  
Automatically run the pipeline using a job scheduler.

CRON expression  
Automatically run the pipeline using a syntax that specifies the time and intervals to run it.

**CRON書式例** <https://docs.aws.amazon.com/imagebuilder/latest/userguide/image-builder-cron.html>

分	時間	何日か	月	何曜日か	意味
0	10	*	*	?	毎日 AM10:00(UTC)に起動
15	12	*	*	?	毎日 PM12:15(UTC)に起動
0	18	?	*	MON-FRI	月-金 PM6:00(UTC)に起動
0	8	1	*	?	毎月 1日 AM8:00(UTC)に起動

# Step 2 :パイプラインの構成

## Infrastructure settings - optional

Image Builder launches EC2 Instances in your account to customize images and run validation tests. These settings specify infrastructure details for instances that run in your AWS account.

### Instance type [Info](#)

Select one or more instance types used to customize your image. We will choose service specific defaults if you do not select any.

Choose one or more instance types

[EC2 default instance types](#)

### SNS topic [Info](#)

Select an SNS topic to receive notifications and alerts from EC2 Image Builder.

Choose SNS topic

[Create new SNS topic](#)

### ▶ VPC, subnet and security groups

Specify advanced settings to launch your instance used to customize your image.

### ▶ Troubleshooting settings [Info](#)

Specify settings to troubleshoot issues with building your image.

## ビルド用インスタンスの設定を行う

- ・ インスタンスタイプを指定
- ・ 通知先SNS Topicを指定
- ・ VPC、subnet、SGsを指定
- ・ トラブルシューティング用の設定

# Step 2 :パイプラインの構成

- インスタンスタイプを指定
- 通知先SNS Topicを指定
- **VPC、subnet、SGsを指定**
- **トラブルシューティング用の設定**

▼ VPC, subnet and security groups  
Specify advanced settings to launch your instance used to customize your image.

 We will choose service specific defaults if you do not provide an entry.

Virtual Private Cloud (VPC) [Info](#)  
Select a VPC to launch your instance.

vpc-07503ed30 [redacted] ▼ 

[Create new VPC](#)

Subnet ID  
The subnet associated with the selected VPC.

subnet-0d6ea0f [redacted] (Private-a) ▼ 

[Create new subnet](#)

Security groups  
The security group associated with the selected VPC

Choose security group ▼ 

sg-03621db [redacted] X  
launch-wizard-4 created 2020-07-13T08:57:09.043+09:00

[Create new security group](#)

インスタンスを作成するsubnetはインターネット接続が可能、もしくはPrivateLinkが作成されている必要があります（後述）

# Step 2 :パイプラインの構成

- ビルド用インスタンスのインスタンスタイプを指定
- 通知先SNS Topicを指定
- VPC、subnet、SGsを指定
- **トラブルシューティング用の設定**

## ▼ Troubleshooting settings [Info](#)

Specify settings to troubleshoot issues with building your image.

### Instance settings [Info](#)

Terminate instance on failure

### Key pair [Info](#)

Choose a key pair, which allows you to securely connect to your instance.

Choose a key pair name

[Create key pair](#)

### Logs

Select a location to save logs.

### S3 location

s3://

## Terminate instance on failure

イメージのビルドが失敗したときにインスタンスをトラブルシューティングできるようにする場合は、チェックボックスがオフになっていることを確認してください。

(チェックしていると失敗時にインスタンスは削除されます)

## Key Pair

作成インスタンスにアクセスするためのAmazon EC2キーペアを指定します

## Logs

ログを出力するS3ロケーションを指定します

# Step 3 :追加設定（オプション）

## Configure additional settings - *optional*

This page allows you to configure post-build activities.

### Associate license configuration to AMI [info](#)

Attach license configurations, a construct of AWS License Manager, to images built with Image Builder. License configurations can be associated with your AMI.

Associate license configuration to AMI  
The unique ID of the license configuration created in AWS License Manager.

[Create new license configuration](#)

### Output AMI

The image that contains the OS and preinstalled software to deploy EC2 instances.

**Name** [info](#)  
Enter a name for your AMI. This will be the AMI name once the pipeline is run.

Max 100 characters

**AMI tags** [info](#)  
Assign your own metadata to each resource in the form of tags.

Key	Value - <i>optional</i>
<input type="text" value="Enter key"/>	<input type="text" value="Enter value"/>

### AMI distribution settings

Select the AWS regions to distribute AMIs and AWS accounts that can launch the AMI.

Select AWS region(s) to distribute the AMI  
Your current region is included by default.

Region:

Encryption:  Configured in storage options

オプションの追加設定で行えることは、以下になります。

- ・ ライセンス構成をAMIに関連付ける  
（AWS License Manager 連携）
- ・ 出力AMIの設定
- ・ AMIの配信設定

## Step 3 :追加設定（オプション）

- ・ ライセンス構成をAMIに関連付ける
- ・ 出力AMIの設定
- ・ AMIの配信設定

**Associate license configuration to AMI** [Info](#)

Attach license configurations, a construct of AWS License Manager, to images built with Image Builder. License configurations contain configurations that were associated with your AMI.

Associate license configuration to AMI  
The unique ID of the license configuration created in AWS License Manager.

Choose a license configuration

Q

SQL Server 2017 Example	arn:aws:license-manager:ap-northeast-1:350010-1b82890665be8a3e7c8f
Windows SQL 2016 Server	arn:aws:license-manager:ap-northeast-1:350010-ae6329d6bc0a9e679670d

AWS License Managerで作成したライセンス構成を、作成するAMIに関連付けることができます。AMIにライセンス数管理が必要なミドルウェアが含まれている場合はこのオプションを適用することにより、ライセンスの使用状況をLicense Managerで追跡できるようになります。

# Step 3 :追加設定 (オプション)

- ・ ライセンス構成をAMIに関連付ける
- ・ 出力AMIの設定
- ・ AMIの配信設定

## Output AMI

The image that contains the OS and preinstalled software to deploy EC2 instances.

### Name [Info](#)

Enter a name for your AMI. This will be the AMI name once the pipeline is

Max 100 characters

### AMI tags [Info](#)

Assign your own metadata to each resource in the form of tags.

Key

Value - *optional*

Remove tag

Add tag

<Name>

作成するAMIの名前を指定します

<AMI tags>

作成するAMIに付与するタグを指定します

# Step 3 :追加設定 (オプション)

- ・ ライセンス構成をAMIに関連付ける
- ・ 出力AMIの設定
- ・ AMIの配信設定

## AMI distribution settings

Select the AWS regions to distribute AMIs and AWS accounts that can launch the AMI.

Select AWS region(s) to distribute the AMI

Your current region is included by default.

Region

ap-northeast-1 ▼

Encryption

Configured in storage options

Add region

Launch permissions

Add/remove AWS user account number to set launch permission default.

Private

Public

Enter account number

Add

### <Select AWS region(s) to distribute the AMI>

AMIを配布するregionを選択します。現在操作を行っているregionはデフォルトで含まれています。

### <Launch permissions>

作成するAMIイメージをPrivate(デフォルト)にするか、Publicにするかを選択します。Privateの場合は、特定のAWSアカウントに権限を付与できます。Publicの場合は、すべてのAWSユーザーがAMIにアクセス可能になります。

# Step 4 :確認・作成

最後にこれまでの設定の確認画面が出て、問題無ければ作成を行います。

Step 1  
[Define recipe](#)

---

Step 2  
[Configure pipeline](#)

---

Step 3  
[Configure additional settings - optional](#)

---

Step 4  
**Review and create**

## Review and create

Step 1: Image recipe Edit

### Recipe details

Recipe name (default) EC2IBBB-recipe	Recipe version (default) 1.0.0	Image OS type Linux	Parent image Amazon Linux 2 x86
Parent image Amazon Linux 2 x86	Build components python-3-linux   1.0.2	Test components ebs-volume-usage- test-linux   1.0.1	

### Storage details

Device name	Snapshot ID	Size (GiB)	Volume type	IOPS	Delete on termination	Encryption (KMS Alias)
/dev/xvda (root)	snap-081e8433c7be73324	8	gp2	100	Enabled	Do not enable

Step 2: Pipeline configuration Edit

# パイプライン作成完了と実行

作成が完了したパイプラインは、[Actions]から[Run pipeline]を選択することで手動実行することができます。

The screenshot shows the AWS Management Console interface for EC2 Image Builder. A green notification banner at the top states: "Image pipeline EC2IBBB was successfully created" with the ARN: arn:aws:imagebuilder:ap-northeast-1:3[redacted]:image-pipeline/ec2ibbb. The main content area displays the "Image pipelines" section, including a search bar and a table of pipelines. The "Run pipeline" button in the "Actions" dropdown menu is circled in red.

**Image pipeline EC2IBBB was successfully created**  
Image pipeline ARN: arn:aws:imagebuilder:ap-northeast-1:3[redacted]:image-pipeline/ec2ibbb

EC2 Image Builder > Image pipelines

**Image pipelines**  
The image pipeline in Image Builder defines all aspects of the process to customize images. It consists of the image recipe, infrastructure configuration, distribution, and test settings.

View details Actions ▲  
Run pipeline  
Disable pipeline  
Delete

Find pipelines by name. Press enter to search all results. Any Status ▼

<input checked="" type="checkbox"/>	Pipeline name	Date created	Version	Status	Last run	ARN
<input checked="" type="checkbox"/>	EC2IBBB	Aug 13, 2020 1:14 AM	1.0.0	Enabled	-	arn:aws:imag 1 [redacted] pipeline/ec2i

# ユースケース & Tips

- ・コンポーネントをGUIから作成する
- ・Image Builder コンポーネントをローカルで開発する
- ・EC2 Image Builder で Ansible playbookを実行する
- ・CloudFormationを使ってリソースを作成する
- ・リソースを他アカウントや組織で共有する
- ・オンプレミスの仮想マシンからAMIを作成する
- ・オンプレミスの仮想マシンイメージを作成する
- ・イメージ作成用EC2を Private Subnet に作成する
- ・カスケード パイプライン で常に最新バージョンを作成する
- ・STIG コンポーネントを適用する
- ・AWS Systems Manager によるAMI作成との使い分け

# コンポーネントをGUIから作成する

EC2 Image Builderにはコンポーネントを作成するためのエディタ画面が用意されています。

EC2 Image Builder > Components > Create component

## Create build component / test

Specify details for build components (software to install, scripts to run, and settings to apply) or tests to run to validate images.

### Component details

#### Image operating system (OS)

Specify the OS the component is compatible with.

Linux ▼

#### Compatible OS Versions

Specify the OS Version(s) that this component is compatible with.

▼

#### Component name

Name of component

Maximum of 128 characters. Letters, numbers, spaces, -, and \_ allowed

#### Description - optional

Maximum of 1024 characters

#### KMS keys - optional

Specify the KMS key to encrypt the component with. Only customer managed keys that have an alias are shown.

Image Builder service key (enabled by default) ▼ 

#### Component version

Component version

Format: major.minor.patch

#### Change description - optional

Details about changes made to the specific version of the component.

Maximum of 1024 characters

# コンポーネントをGUIから作成する

**Definition document** [Info](#)

This defines the actions for Image Builder to perform on your image. It uses the YAML format to list customizations steps.

Define document content  
Specify content in YAML.

Use build component example  
Content can be edited inline.

**Content**

```
1 name: HelloWorldTestingDocument
2 description: This is hello world testing document.
3 schemaVersion: 1.0
4
5 phases:
6   - name: build
7     steps:
8       - name: HelloWorldStep
9         action: ExecuteBash
10        inputs:
11          commands:
12            - echo "Hello World! Build."
13
14   - name: validate
15     steps:
16       - name: HelloWorldStep
17         action: ExecuteBash
18        inputs:
19          commands:
20            - echo "Hello World! Validate."
21
22   - name: test
23     steps:
24       - name: HelloWorldStep
25         action: ExecuteBash
26        inputs:
27          commands:
28            - echo "Hello World! Test."
29
```

ビルドコンポーネント定義の例

**Definition document** [Info](#)

This defines the actions for Image Builder to perform on your image. It uses the Y

Define document content  
Specify content in YAML.

Use build c  
Content can f

**Content**

```
1 name: HelloWorldTestingDocument
2 description: This is hello world testing document.
3 schemaVersion: 1.0
4
5 phases:
6   - name: test
7     steps:
8       - name: HelloWorldStep
9         action: ExecuteBash
10        inputs:
11          commands:
12            - echo "Hello World! Test."
13
```

テストコンポーネント定義の例

## 【参考】アクションモジュール

その他ドキュメント内で使える便利なアクションモジュールとして、以下のものがあります。

アクションモジュール	内容
ExecuteBinary	バイナリを実行
ExecutePowerShell	PowerShellスクリプトを実行
ExecuteBash	Bashスクリプトを実行
Reboot	インスタンスを再起動する
UpdateOS	WindowsとLinuxのアップデートをインストールする
S3Upload	ローカルファイルをS3オブジェクトにコピー
S3Download	S3オブジェクトをローカルファイルにコピー
SetRegistry	Windowsレジストリキーの値を設定

# アクションモジュールの使用例

## ExecuteBash

## Bashスクリプトを実行

```
name: InstallAndValidateCorretto
action: ExecuteBash
inputs:
  commands:
    - sudo yum install java-11-amazon-corretto-headless -y
    - |
      function fail_with_message() {
        1>&2 echo $1
        exit 1
      }

      ARCH=`/usr/bin/arch`

      JAVA_PATH=/usr/lib/jvm/java-11-amazon-corretto.$ARCH/bin/java
      if [ -x $JAVA_PATH ]; then
        echo "Amazon Corretto 11 JRE is installed."
      else
        fail_with_message "Amazon Corretto 11 JRE is not installed. Failing."
      fi

      JAVAC_PATH=/usr/lib/jvm/java-11-amazon-corretto.$ARCH/bin/javac
      if [ -x $JAVAC_PATH ]; then
        echo "Amazon Corretto 11 JDK is installed."
      else
        fail_with_message "Amazon Corretto 11 JDK is not installed. Failing."
      fi
```

# ソフトウェアの追加と削除を行う

AWS提供のビルドコンポーネント（補足資料参照）もスクリプトを確認可能。これを参考に、新しいソフトウェアの追加/削除コンポーネントを作成できる。

EC2 Image Builder

Image pipelines

Recipes

**Components**

Images

EC2 Image Builder > Components

**Components**

Both build components and tests are part of a build process. They contain software, settings, and configurations that are used during the process of building custom images. Tests are used to validate functionality, security, performance, and other aspects of the build process.

Find components by name. Press enter to search.

Component name
<input type="checkbox"/> amazon-cloudwatch-agent-linux

EC2 Image Builder > Components > amazon-cloudwatch-agent-linux

## amazon-cloudwatch-agent-linux

**Content**

```
77 |         echo {{ build.OperatingSystemRelease.outputs.stdout }}
78 |     ;;
79 |     esac
80 |
81 | - name: FileExtension
82 |   action: ExecuteBash
83 |   inputs:
84 |     commands:
85 |     - |
86 |       RELEASE='{{ build.OperatingSystemRelease.outputs.stdout }}'
87 |       if [ `echo "$RELEASE" | grep -E '^(amzn|centos|rhel|sles)'` ]; then
88 |         echo 'rpm'
89 |       elif [ `echo "$RELEASE" | grep -E '^(debian|ubuntu)'` ]; then
90 |         echo 'deb'
91 |       else
92 |         echo "The Operating System $RELEASE does not have a file extension specified. Failing build."
93 |         exit {{ build.Fail.outputs.stdout }}
94 |       fi
95 |
96 | - name: Source
97 |   action: ExecuteBash
98 |   inputs:
99 |     commands:
100 |    - |
101 |      RELEASE='{{ build.OperatingSystemRelease.outputs.stdout }}'
102 |      TYPE='{{ build.OperatingSystemArchitecture.outputs.stdout }}'
```



# Image Builder コンポーネントをローカルで開発する

- ・ **コンポーネント管理アプリケーション (AWSTOE:AWS Task Orchestrator and Executor)** を使うことで、ローカル環境でもImage Builder コンポーネントの開発・トラブルシューティングを行うことができます
- ・ **AWSTOEはスタンドアロンアプリケーション**なので、任意のクラウドもしくはオンプレミス環境で実行可能です。また、ダウンロードファイル自体が実行ファイルになっています (インストールコマンド等は不要。パスを通せばそのまま使用できる)

## 【手順】

- (1) AWSTOEをインストールする / 必要に応じてAWS認証情報を設定
- (2) インストールしたマシンで、Image Builder コンポーネント ドキュメントを作成
- (3) 作成したコンポーネントを検証する
- (4) コンポーネントを実行する

次ページから、Linuxでの使用例を記載します。

# Image Builder コンポーネントをローカルで開発する

## (1) AWSTOEをインストールする / 必要に応じてAWS認証情報を設定

AWSTOEは下記のアーキテクチャ、OSに対応しています。

アーキテクチャ	対応OS
386	Amazon Linux2,RHEL 8 and 7, Ubuntu 18 and 16, CentOS 7, SUSE 15
AMD64	Windows Server 2019/2016/2012 R2/version 1909 Amazon Linux2, RHEL 8 and 7, Ubuntu 18 and 16, CentOS 7, SUSE 15
ARM64	Amazon Linux2, RHEL 8 and 7, Ubuntu 18 and 16, CentOS 7, SUSE 15

AWSTOEのダウンロードリンクは下記に記載されています。

<https://docs.aws.amazon.com/imagebuilder/latest/userguide/image-builder-component-manager.html>

必要に応じて、S3,CloudWatchにロギングするためのAWS認証情報を設定します。

```
$ export AWS_ACCESS_KEY_ID=your_access_key_id
```

```
$ export AWS_SECRET_ACCESS_KEY=your_secret_access_key
```

※AWS認証情報はロギングに必要な権限設定が行われているIAMロールから作成する

※EC2インスタンスで実行している場合は、IAMロールの割り当てでOK

# Image Builder コンポーネントをローカルで開発する

## (2) インストールしたマシンで、Image Builder コンポーネント ドキュメントを作成

```
$ vi hello-world-linux.yml
```

```
name: Hello World
description: This is hello world testing document for Linux.
schemaVersion: 1.0
phases:
  - name: build
    steps:
      - name: HelloWorldStep
        action: ExecuteBash
        inputs:
          commands:
            - echo 'Hello World from the build phase.'
  - name: validate
    steps:
      - name: HelloWorldStep
        action: ExecuteBash
        inputs:
          commands:
            - echo 'Hello World from the validate phase.'
  - name: test
    steps:
      - name: HelloWorldStep
        action: ExecuteBash
        inputs:
          commands:
            - echo 'Hello World from the test phase.'
```

# Image Builder コンポーネントをローカルで開発する

## (3) インストールしたマシンで、Image Builder コンポーネント ドキュメントを検証

```
$ awstoe validate --documents hello-world-linux.yml
```

### 【結果：成功例】

```
{
  "validationStatus": "success",
  "message": "Document(s) [hello-world-linux.yml] is/are valid."
}
```

### 【結果：NG例】

```
{
  "validationStatus": "failed",
  "message": "validate document failed for hello-world-linux-bad.yml. Error : Parsing step 'HelloWorldStep' in phase 'build' failed. Error: line 1: cannot unmarshal string `echo 'H...` into []string."
}
```

# Image Builder コンポーネントをローカルで開発する

## (4) コンポーネントを実行する

### 【フェーズを指定して実行 build/test】

```
$ awstoe run --documents hello-world-linux.yml --phases build
```

### 【すべてのフェーズを実行（ドキュメント全体）】

```
$ awstoe run --documents documentName.yaml
```

### 【結果：成功例】

```
{
  "executionId": "c264c9ac-e3a3-11ea-9704-06a88c4af07a",
  "status": "success",
  "failedStepCount": 0,
  "executedStepCount": 1,
  "failureMessage": "",
  "logUrl": "/home/ec2-user/TOE_2020-08-21_11-45-06.UTC-0_c264c9ac-e3a3-11ea-9704-06a88c4af07a"
}
```

※ --trace を使うことで、トレースログを出力可能

© 2020, Amazon Web Services, Inc. or its Affiliates.



# EC2 Image Builder で Ansible playbookを実行する

以前から構成管理ツールである“Ansible”を使っている場合、Ansible playbookをEC2 Image builder パイプラインから実行することによって、Ansible資産を活用できます

## ■ Ansible playbookを実行するための要件

- ・ Image Builder内でAnsibleを実行する場合、リモートではなくローカルホスト上で実行するようにplaybookを（再）構成する必要がある
    - ・ hostを 127.0.0.1 にする
    - ・ connection を local にする
- また、gather\_facts は false で問題ありません。

## ■ Ansible playbook 実行手順

1. Ansibleをインストールするアクションの作成 (**ExecuteBash**)
2. playbookをダウンロードするアクションの作成 (**S3Download**)
3. playbookの実行 (**ExecuteBinary**)
4. playbookの削除 (**ExecuteBash**)
5. 実行内容に対する検証の実行 (**ExecuteBash**)

次ページから、playbookの実行例を用いて説明します

# EC2 Image Builder で Ansible playbookを実行する

ここからは、playbookを実行するビルドコンポーネントを記述 (YAML)していきます。まず最初に、ドキュメントにおけるトップレベルのプロパティを宣言します。

```
name: 'Ansible Playbook Execution on Amazon Linux 2'  
description: 'This is a sample component that demonstrates how to download and  
execute an Ansible playbook against Amazon Linux 2.'  
schemaVersion: 1.0  
phases:  
  - name: build  
    steps:
```

続けて、1～5を記述していきます。

## 1. Ansibleをインストールするアクションの作成 (ExecuteBash)

```
- name: InstallAnsible  
  action: ExecuteBash  
  inputs:  
    commands:  
      - sudo amazon-linux-extras install -y ansible2
```

# EC2 Image Builder で Ansible playbook を実行する

## 2. playbook をダウンロードするアクションの作成 (S3Download)

```
- name: DownloadPlaybook
  action: S3Download
  inputs:
    - source: 's3://mybucket/my-playbook.yml'
      destination: '/tmp/my-playbook.yml'
```

## 3. playbook の実行 (ExecuteBinary)

```
- name: InvokeAnsible
  action: ExecuteBinary
  inputs:
    path: ansible-playbook
  arguments:
    - '{{build.DownloadPlaybook.inputs[0].destination}}'
```

コンポーネント管理アプリケーションの機能 (chain) で、  
前段のステップにある `'/tmp/my-playbook.yml'` が渡される

# EC2 Image Builder で Ansible playbook を実行する

## 4. playbook の削除 (ExecuteBash)

```
- name: DeletePlaybook
  action: ExecuteBash
  inputs:
    commands:
      - rm '{{build.DownloadPlaybook.inputs[0].destination}}'
```

## 5. 実行内容に対する検証の実行 (ExecuteBash)

```
- name: validate
  steps:
    - name: ValidateResponse
      action: ExecuteBash
      inputs:
        commands:
          - curl -s http://127.0.0.1 | grep "Hello world from EC2 Image
            Builder and Ansible"
```

これで1~5まで実装完了し、Ansible playbook を実行するビルドコンポーネントが完成

完全版はこちら

<https://aws.amazon.com/jp/blogs/compute/executing-ansible-playbooks-in-your-amazon-ec2-image-builder-pipeline/>

# CloudFormationを使ってリソースを作成する

AWS CloudFormation で、EC2 Image Builderのリソース（コンポーネント、レシピ、パイプライン）を作成することが可能です。

Resource types	内容
AWS::ImageBuilder::Component	コンポーネントの定義
AWS::ImageBuilder::DistributionConfiguration	出力AMIと、配布先の定義
AWS::ImageBuilder::Image	イメージのビルドに関する情報
AWS::ImageBuilder::ImagePipeline	イメージパイプラインの定義
AWS::ImageBuilder::ImageRecipe	イメージレシピの定義
AWS::ImageBuilder::InfrastructureConfiguration	一時的に構築するEC2の定義

[https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/AWS\\_ImageBuilder.html](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/AWS_ImageBuilder.html)

# CloudFormationを使ってリソースを作成する

## CloudFormation サンプル

```
AWSTemplateFormatVersion: 2010-09-09
```

```
Resources:
```

### ImagePipeline:

```
Type: 'AWS::ImageBuilder::ImagePipeline'  
Properties:  
  Name: 'image-pipeline-Blackbelt'  
  Description: 'description'  
  ImageRecipeArn: !Ref ImageRecipe  
  InfrastructureConfigurationArn: !Ref InfrastructureConfiguration  
  ImageTestsConfiguration:  
    ImageTestsEnabled: false  
    TimeoutMinutes: 90  
  Schedule:  
    ScheduleExpression: 'cron(0 0 * * 0)'  
    PipelineExecutionStartCondition: 'EXPRESSION_MATCH_ONLY'  
  Status: 'DISABLED'
```

### InfrastructureConfiguration:

```
Type: 'AWS::ImageBuilder::InfrastructureConfiguration'  
Properties:  
  Name: 'infrastructure-configuration-BB'  
  InstanceProfileName: !Ref InstanceProfile  
  Description: 'EC2 ImageBuilder BlackBelt'  
  InstanceTypes:  
    - 'm5.large'  
  SubnetId: 'subnet-9d2dddd5'  
  SecurityGroupIds:  
    - 'sg-0b6924425c099683d'  
  TerminateInstanceOnFailure: True
```

### InstanceProfile:

```
Type: AWS::IAM::InstanceProfile  
Properties:  
  InstanceProfileName: ImageBuilder-BB  
  Roles:  
    - EC2ImageBuilder-08
```

### Component:

```
Type: 'AWS::ImageBuilder::Component'  
Properties:  
  Name: 'component-BB'  
  Platform: 'Linux'  
  Version: "1.0.0"  
  Description: 'Hello Image Builder'  
  Data: |  
    name: HelloWorldTestingLinuxDoc - InlineData  
    description: This is hello world testing doc  
    schemaVersion: 1.0  
  
  phases:  
    - name: build  
      steps:  
        - name: HelloWorldStep  
          action: ExecuteBash  
          inputs:  
            commands:  
              - echo "Hello World! Build."  
    - name: validate  
      steps:  
        - name: HelloWorldStep  
          action: ExecuteBash  
          inputs:  
            commands:  
              - echo "Hello World! Validate."  
    - name: test  
      steps:  
        - name: HelloWorldStep  
          action: ExecuteBash  
          inputs:  
            commands:  
              - echo "Hello World! Test."
```

### ImageRecipe:

```
Type: 'AWS::ImageBuilder::ImageRecipe'  
Properties:  
  Name: 'image-recipe-Blackbelt08'  
  Version: '1.0.0'  
  ParentImage: arn:aws:imagebuilder:ap-northeast-1:aws:image/amazon-linux-2-  
x86_2020.7.24  
Components:  
  - ComponentArn: !Ref Component
```

# リソースを他アカウントや組織で共有する

## (1) EC2 Image Builderで作成イメージを共有する

- ・ AWS EC2 Image Builder パイプライン内で設定  
=>参照 Step 3 :追加設定 (オプション) AMIの配信設定

## (2) AWS Resource Access Manager (AWS RAM) でリソースを共有

- ・ 他のAWSアカウントやAWS Organizations 内 (組織全体、OU、特定のアカウント) で共有できます。(詳細は次ページ)

### 【前提条件】

- ・ RAMを実行するAWSアカウントが、上記リソースの所有者であること (共有されているリソースを共有することはできない)
- ・ リソースが暗号化されている場合は、AWS KMSのCMKも明示的に共有する
- ・ AWS Organizationsで共有する場合は、AWS Organizationsの設定から"AWS Resource Access Manager"を有効化する

# リソースを他アカウントや組織で共有する

## (2) AWS Resource Access Manager (AWS RAM) でリソースを共有

EC2 Image Builder のリソース（イメージ / イメージレシピ / コンポーネント）は **AWS Resource Access Manager** を使ってのアカウント間の共有が可能です。

Resource Access Manager > 自分が共有: リソースの共有 > リソースの共有の作成

### リソースの共有の作成

リソースの共有を作成し、AWS アカウントや組織ユニット、組織にリソースへのアクセス権限を付与します。

**説明**

**名前**  
このリソースの共有にわかりやすい名前を付けます

リソースの共有名を追加

**リソース - オプション**  
リソースの共有に追加するリソースの選択

リソースタイプを選択します

サブネット

Q

- Aurora DB クラスター
- CodeBuild プロジェクト
- CodeBuild レポートグループ
- Glue カタログ
- Glue データベース
- Glue データベース
- Image Builder イメージ
- Image Builder イメージレシピ
- Image Builder コンポーネント

Image Builder イメージ

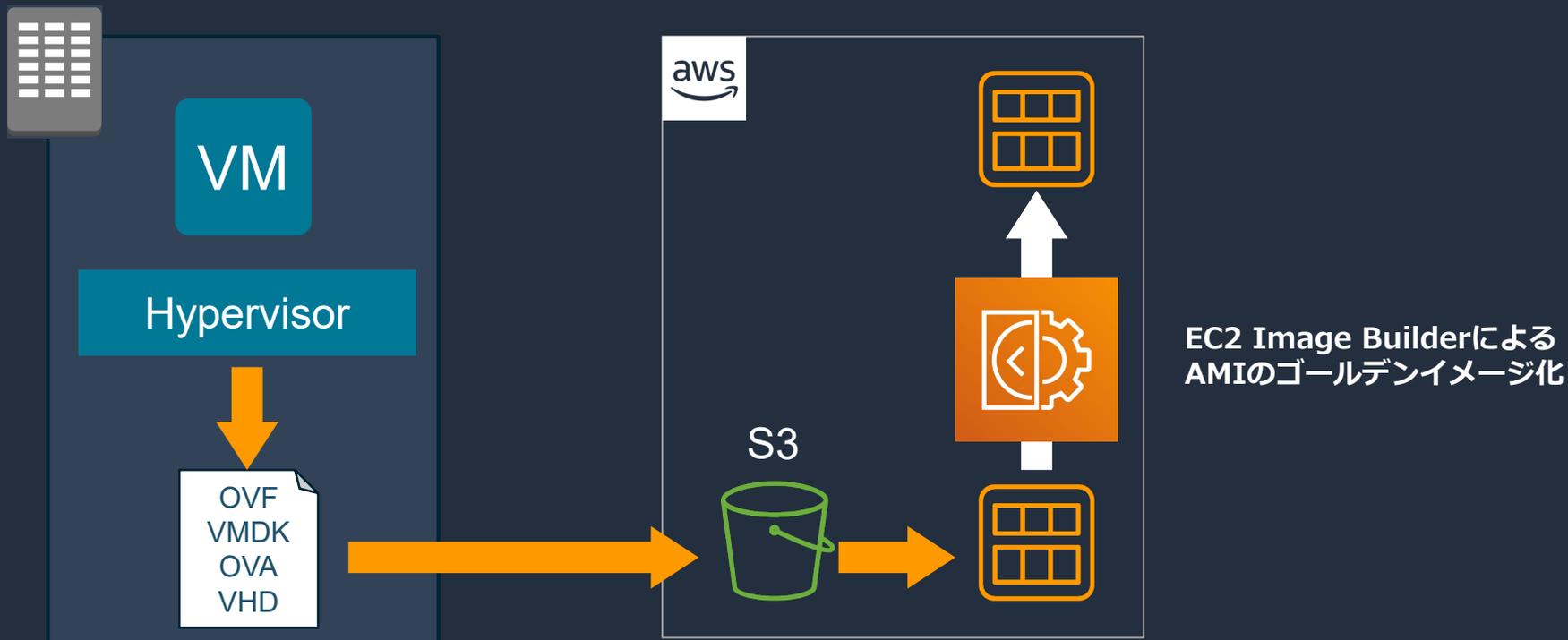
Image Builder イメージレシピ

Image Builder コンポーネント

<https://docs.aws.amazon.com/imagebuilder/latest/userguide/image-builder-resource-sharing.html>

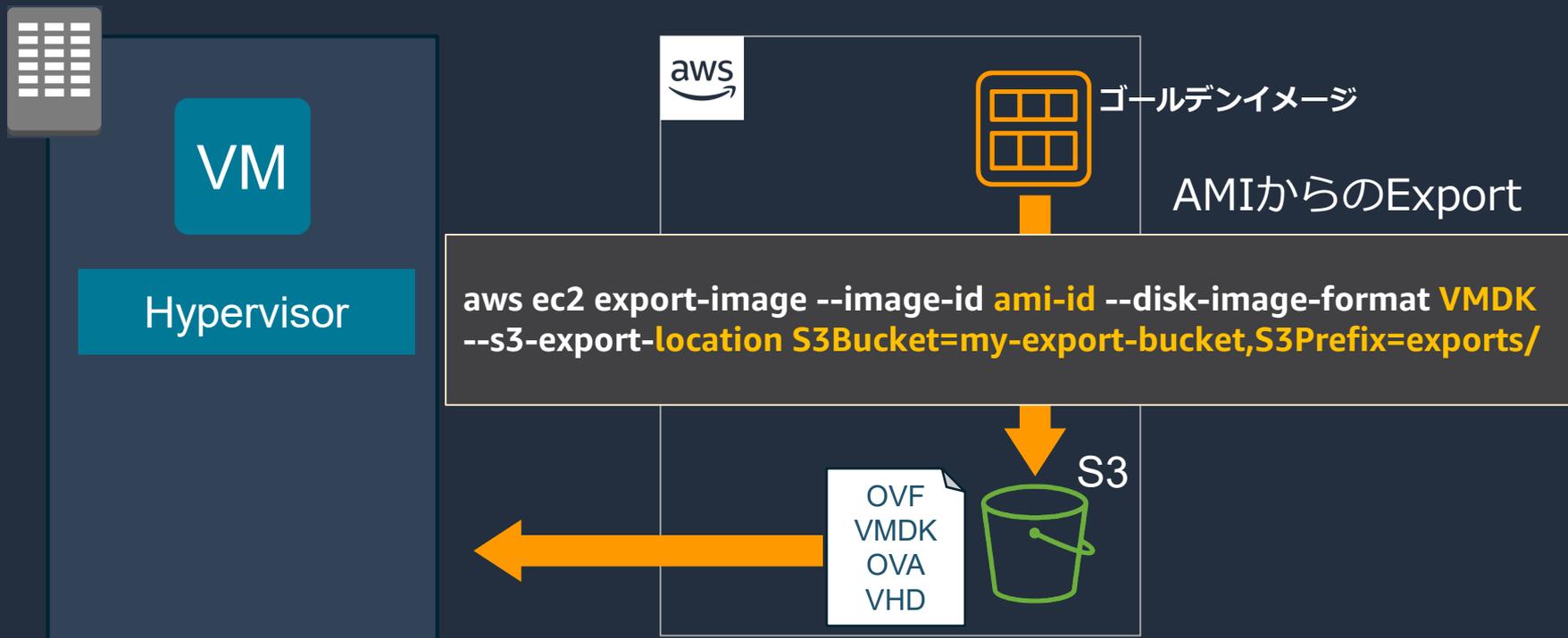
# オンプレミスの仮想マシンからAMIを作成する

EC2 Image Builder と AWS VM Import/Export (VMIE) を組み合わせることで、オンプレミスからの仮想マシンイメージも作成可能



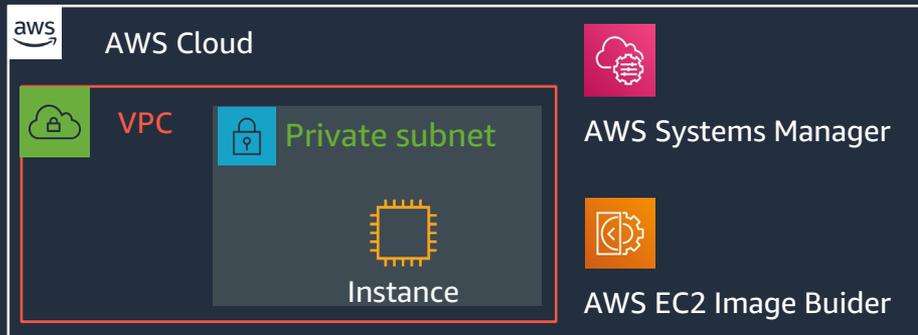
# オンプレミスの仮想マシンイメージを作成する

EC2 Image Builder と AWS VM Import/Export (VMIE) を組み合わせることで、オンプレミスの仮想マシンイメージも作成可能



# イメージ作成用EC2を Private Subnet に作成する

Image BuilderはAMIを作る過程で、イメージソースとなるEC2を作成します。EC2はVPC外のサービスとのアクセスが必須であるため、Private Subnet に作成する場合は以下の対応を行います。



通信要件が満たせていない時はSSM実行ステップ  
(RunCommandなど) でタイムアウトエラーが発生

ステップ #	ステップ名	アクション	ステータス
1	WaitForInstanceToSpinUp	aws:waitForAwsResourceProperty	成功
2	VerifySSMAgentBranch	aws:branch	成功
3	VerifySSMAgentLinux	aws:runCommand	タイムアウト

## 【対応方法 1】

- NAT Gatewayを作成し、ルーティングする

## 【対応方法 2】

- EC2を作成するPrivate subnetに、アクセス先のサービスのPrivate Linkを設定する

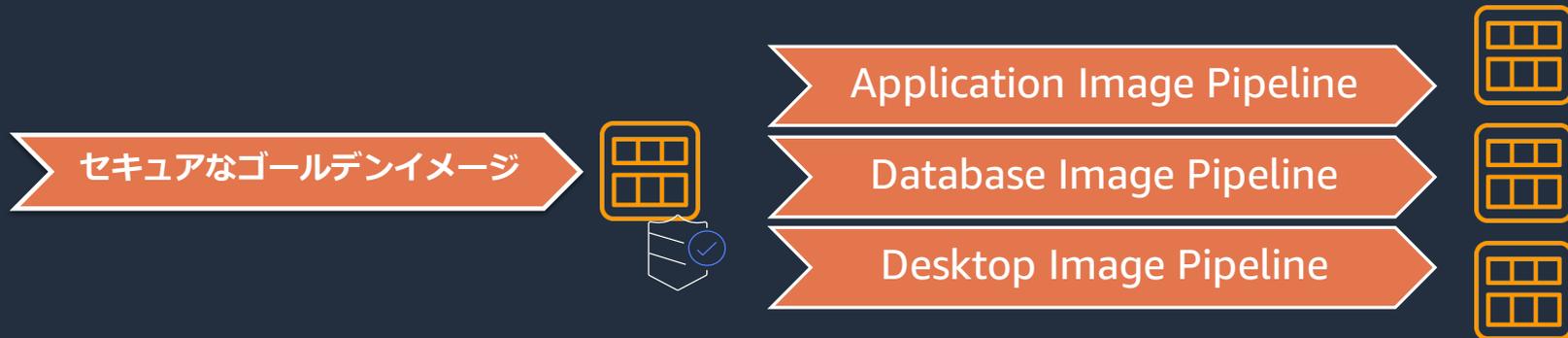
EC2 Image Builderに関するPrivateLink: imagebuilder

Systems Managerに関するPrivateLink: ssm, ssmmessages, ec2messages

+必要に応じてS3などのPrivate Linkを設置。

# カスケード パイプライン で常に最新バージョンを作成する

[Always build latest version option] オプションを使用してイメージをバージョン管理します。下流パイプラインは、上流のパイプラインで作成された最新バージョンの作成イメージを使用します。



<input type="checkbox"/>	Recipe name	Version	Image OS	Source image
<input type="checkbox"/>	<a href="#">MyBasicLinuxRecipe</a>	1.0.0	Linux	arn:aws:imagebuilder:us-east-1:aws:image/amazon-linux-2-x86/2020.1.8
<input type="checkbox"/>	<a href="#">MyBasicLinuxRecipe</a>	1.0.1	Linux	arn:aws:imagebuilder:us-east-1:aws:image/amazon-linux-2-x86/2020.1.8
<input type="checkbox"/>	<a href="#">Web-Server2029-IIS</a>	1.0.5	Windows	arn:aws:imagebuilder:us-east- <span style="background-color: black; color: black;">XXXXXXXXXX</span> /windows2019-stig-low-recipe-6ef66b4c9974/x.x.x
<input type="checkbox"/>	<a href="#">Web-Server2029-IIS</a>	1.0.7	Windows	arn:aws:imagebuilder:us-east- <span style="background-color: black; color: black;">XXXXXXXXXX</span> /windows2019-stig-low-recipe-6ef66b4c9974/x.x.x
<input type="checkbox"/>	<a href="#">Web-Server2029-IIS</a>	1.0.8	Windows	arn:aws:imagebuilder:us-east- <span style="background-color: black; color: black;">XXXXXXXXXX</span> /windows2019-stig-low-recipe-6ef66b4c9974/x.x.x
<input type="checkbox"/>	<a href="#">Windows2019_STIG_Low-recipe-6ef66b4c9974</a>	1.0.0	Windows	arn:aws:imagebuilder:us-east-1:aws:image/windows-server-2019-english-full-base-x86/2020.3.11

# STIG コンポーネントを適用する

- セキュリティ技術実装ガイド (STIG) は、防衛情報システム局 (DISA) が情報システムとソフトウェアを保護するために作成した設定標準です
- システムを STIG 標準に準拠させるには、さまざまなセキュリティ設定をインストール、設定、およびテストする必要があります (STIG コンプライアンス用に構成された Amazon EC2 Windows Server AMI は、160 以上の必須セキュリティ項目が事前に設定されています)
- **EC2 Image Builder の Windows STIG コンポーネント** を使うことで、STIG で規定されているセキュリティ設定を自動的に反映 (構成ミスのスキャンして、修正スクリプトを実行) させることが可能です

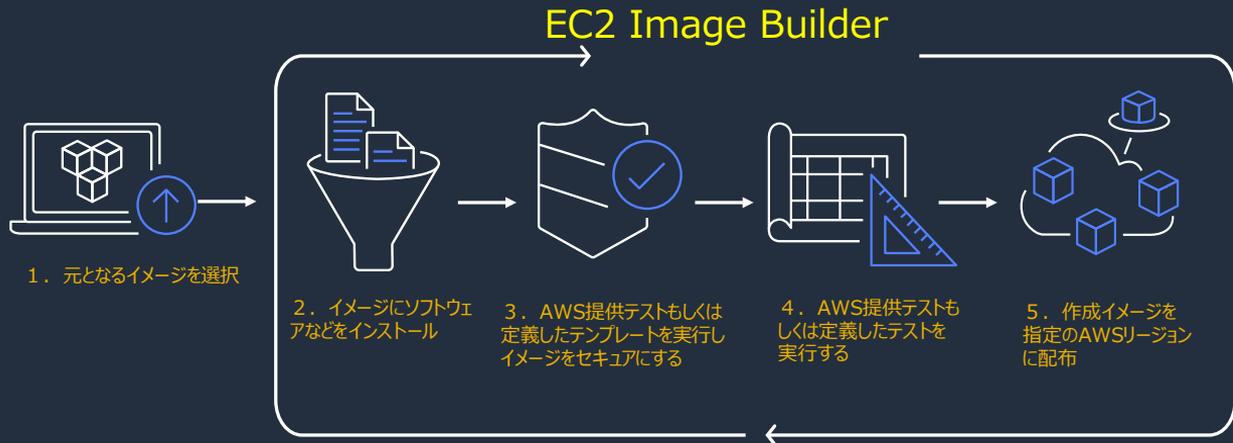
## STIG コンポーネントの例

stig-build-windows-medium   Version 1.1.0	Applies the medium and low severity STIG settings to Windows instances. For more information, see <a href="https://docs.aws.amazon.com/imagebuilder/latest/userguide/image-builder-stig.html">https://docs.aws.amazon.com/imagebuilder/latest/userguide/image-builder-stig.html</a> .	Windows	BUILD
---	---	---------	-------

<https://docs.aws.amazon.com/imagebuilder/latest/userguide/image-builder-stig.html>

# AWS Systems Manager によるAMI作成との使い分け

OSイメージを定期的作成する際、AWS Systems Manager を用いる方法もあります。EC2 Image Builder も Systems Manager Automation で制御実行されていますが、両者の使い分けについて考え方を整理しておきます。



・ EC2 Image Builder はイメージ作成の自動化パイプラインをワンストップで構築するマネージドサービスです。特に比較ポイントになるのは、**EC2 Image Builderには、イメージを実稼働環境で使用する前に検証する仕組みが組み込まれていること、そのためのコンポーネントがあらかじめ準備されていることです。**

・ **EC2 ImageBuilderのインプットとなるソースはOSイメージ (AMI)に限定されるため、稼働状態のEC2のAMIを取得したい、という要件の場合は、Systems Manager で “AWS-CreateImage” などを実行する運用の出番になります。**

・ Systems Manager で作成されたAMIをEC2 Image Builderのソースイメージにして、ゴールデンイメージ（稼働環境に素早くデプロイするためのテンプレートイメージ。AMI をクリーンアップしたり、ソフトウェアやアプリを最新化して共通的に利用できる状態にする）化するパイプラインを作成するのもよいでしょう。

# トラブルシューティングと デバッグ

# トラブルシューティングとデバッグを行う

よく起きる問題のパターンと、シューティングアプローチは以下になります。

パターン	トラブルシューティング
権限に問題	<p>“AccessDenied : Access Denied status code : 403” や “UnauthorizedOperation” などのメッセージが出力されている時は、適切なIAMポリシーが割り当てられないことが原因であるケースが多い。</p> <p>【対応】</p> <p>コンポーネントが使用するAPIやリソースにアクセスする権限や、ロギング（S3アクセスなど）に必要な権限が設定されているかを確認します。</p>
NW疎通に問題	<p>Build fails with “status = ‘TimedOut’” や “failure message = ‘Step timed out while step is verifying the SSM Agent availability on the target instance(s)’” のメッセージが出力されている時は、NW疎通もしくは権限に問題が発生しているケースがある。</p> <p>【対応】</p> <ul style="list-style-type: none"><li>・イメージビルドのためのEC2 インスタンスが作成されるサブネットは、インターネットアクセスもしくはPrivateLinkが設定されている必要がある。</li></ul> <p>Private subnetの場合は、NATゲートウェイもしくは Image Builder / Sysmtems Manager / Amazon S3 / CloudWatchのPrivateLinkエンドポイントを必要に応じて作成する。</p> <ul style="list-style-type: none"><li>・ EC2InstanceProfileForImageBuilderおよびAmazonSSMManagedInstanceCore と利用サービスに則ったロールが適用されていることを確認</li></ul>
自作コンポーネントに問題	<p>自作したコンポーネント実行箇所でエラーが出ている際は、EC2 インスタンスにアクセスしないと原因/詳細が分からないことが多い。</p> <p>【対応】</p> <p>EC2 インスタンスにアクセスし、ログを確認、対応する。</p>

# トラブルシューティングとデバッグを行う

エラー等が発生したときの調査方法 この流れで見ていくことをおすすめします

## (1) EC2 Image Builderが出力するログの確認

- ・ CloudWatch Logs
- ・ S3
- ・ CloudTrail (必要に応じて)
- ・ EC2 Image Builder

## (2) Systems Manager Automation 実行履歴の確認

## (3) ビルドコンポーネント実行中に起動されたEC2の調査



# トラブルシューティングとデバッグを行う

## (1) EC2 Image Builderが出力するログの確認

## CloudWatch Logs

EC2上の標準出力・標準エラーが  
ストリームされる

CloudWatch > CloudWatch Logs > Log groups > /aws/imagebuilder/EIB-4-recipe-33a27ef80cf3 > 1.0.0/1

CloudWatch Logs インサイトを試す  
CloudWatch Logs insights allow

Log events

イベントをフィルター

タイムスタンプ	メッセージ
	ロードする古いイベントがあります。さらにロードします。
2020-08-20T01:18:35.382+09:00	Info Stdout: dotnet-runtime-3.1 x86_64 3.1.7-1 packages-microsoft-com-prod 29 M
Info Stdout: dotnet-runtime-3.1	x86_64 3.1.7-1 packages-microsoft-com-prod 29 M
2020-08-20T01:18:35.382+09:00	Info Stdout: dotnet-runtime-deps-3.1 x86_64 3.1.7-1 packages-microsoft-com-prod 2.8 k
Info Stdout: dotnet-runtime-deps-3.1	x86_64 3.1.7-1 packages-microsoft-com-prod 2.8 k
2020-08-20T01:18:35.382+09:00	Info Stdout: dotnet-targeting-pack-3.1 x86_64 3.1.0-1 packages-microsoft-com-prod 3.4 M
Info Stdout: dotnet-targeting-pack-3.1	x86_64 3.1.0-1 packages-microsoft-com-prod 3.4 M
2020-08-20T01:18:35.382+09:00	Info Stdout: netstandard-targeting-pack-2.1
Info Stdout: netstandard-targeting-pack-2.1	
2020-08-20T01:18:35.382+09:00	Info Stdout: x86_64 2.1.0-1 packages-microsoft-com-prod 2.1 M

LogGroup : `"/aws/imagebuilder/<ImageName>`

LogStream : `<ImageVersion>/<ImageBuildVersion>["x.x.x/x"]`

# トラブルシューティングとデバッグを行う

## (1) EC2 Image Builderが出力するログの確認 CloudWatch Logs

▶	2020-08-20T01:34:51.557+09:00	Info Step Execute_Inspector_Assessment
▶	2020-08-20T01:34:51.874+09:00	Info Command execution resulted in an error
▶	2020-08-20T01:34:51.874+09:00	Info Stderr: <b>Unable to tag instance. Error: UnauthorizedOperation:</b> You are not authorized to perform this operation
▶	2020-08-20T01:34:51.874+09:00	Info ExitCode 1
▶	2020-08-20T01:34:52.875+09:00	Info TOE has completed execution with failure - <b>Execution failed!</b>

エラー発生時の例（上記メッセージをクリックすることで全体表示）

Info Stderr: **Unable to tag instance. Error: UnauthorizedOperation: You are not authorized to perform this operation. Encoded authorization failure message: sV0iG-31m7MTCAjskqFTvaewshgLE**（中略）  
LkpjxMRbnNnjeHe8JnSjPnlQj5urnahPKXYvuI3100mx8FtmCQRWGacy41jFbvJCdqwKP  
ciYGcc7AfnmFPCVsJuEjWoOK3CLhjpLYoe--**status code: 403, request id: d4ea7bbf-f326-4aca-9\*\*5-8\*\*\*60412266**

# トラブルシューティングとデバッグを行う

## (1) EC2 Image Builderが出力するログの確認 S3

特に application.log がトラブルシューティングに有用

<input type="checkbox"/>  0__dotnet-core-runtime-linux__3.1.2_1.yml		選択したビルドコンポーネントの実物
<input type="checkbox"/>  1__inspector-test-linux__1.0.1_1.yml		選択したテストコンポーネントの実物
<input type="checkbox"/>  application.log		AWSTOEに関するdebugログ
<input type="checkbox"/>  chaining.json		ビルド/テストコンポーネントの実行ログ
<input type="checkbox"/>  console.log		標準出力/標準エラーログ (CWLと同等)
<input type="checkbox"/>  detailedoutput.json		オーケストレーションに関する すべての詳細情報を記述したファイル

# トラブルシューティングとデバッグを行う

## (1) EC2 Image Builderが出力するログの確認 CloudTrail

- ・ イベントソース: `ssm.amazonaws.com` でフィルター  
SSM Automationの実行が開始されていることがわかる

フィルター:	イベントソース ▼	ssm.amazonaws.com ✕	時間範囲:	時間範囲の選択
イベント時間	ユーザー名	イベント名		
▼ 2020-08-21, 07:56:27 AM	imagebuilderdeea79e5-9e00-44b7-bc72-3ece0015b299	StartAutomationExecution		

- ・ ImageBuilderのユーザ名を見つけたら、ユーザ名でフィルターし検索

フィルター:	ユーザー名 ▼	imagebuilderdeea79e5-9... ✕	時間範囲:	時間範囲の選択
イベント時間	ユーザー名	イベント名		
▶ 2020-08-21, 08:09:30 AM	imagebuilderdeea79e5-9e00-44b7-bc72-3ece0015b299	DescribeImages		
▶ 2020-08-21, 08:09:20 AM	imagebuilderdeea79e5-9e00-44b7-bc72-3ece0015b299	DescribeImages		
▶ 2020-08-21, 08:09:16 AM	imagebuilderdeea79e5-9e00-44b7-bc72-3ece0015b299	CreateImage		
▶ 2020-08-21, 08:08:48 AM	imagebuilderdeea79e5-9e00-44b7-bc72-3ece0015b299	DescribeInstanceStatus		
▶ 2020-08-21, 08:08:38 AM	imagebuilderdeea79e5-9e00-44b7-bc72-3ece0015b299	DescribeInstanceStatus		
▶ 2020-08-21, 08:08:35 AM	imagebuilderdeea79e5-9e00-44b7-bc72-3ece0015b299	DescribeInstanceStatus		

イベントの表示

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAVC7R5IQ7ESHILT6IA:imagebuilderdeea79e5-9e00-44b7-bc72-3ece0015b299",
    "arn": "arn:aws:sts::[redacted]:assumed-role/AWSServiceRoleForImageBuilder/imagebuilderdeea",
    "accountId": "[redacted]",
    "accessKeyId": "[redacted]",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "[redacted]",
        "arn": "arn:aws:iam::[redacted]:role/aws-service-role/imagebuilder.amazonaws.com/Ak",
        "accountId": "[redacted]",
        "userName": "AWSServiceRoleForImageBuilder"
      }
    }
  }
}
```

APIの実行履歴をもとに、トラブルシューティングを行う



# トラブルシューティングとデバッグを行う

## (1) EC2 Image Builderが出力するログの確認 EC2 Image Builder

Output image Configuration Image recipe

EC2 Image Builderから実行しているSSMでエラーが発生が発生していることがわかる

Output images  
Output images of the pipeline

Find output images

Version	Date created	Status	Reason for failure
1.0.0/1	Aug 20, 2020 1:14 AM	Failed	SSM execution 'd366feec-a1c9-4218-876e-bb53ae05a421' failed with status = 'Failed' in state = 'TESTING' and failure message = 'Document arn:aws:imagebuilder:ap-northeast-1:255485124026:component/inspector-test-linux/1.0.1/1 failed!'

SSM execution 'd366feec-a1c9-4218-876e-bb53ae05a421' failed with status = 'Failed' in state = 'TESTING' and failure message = 'Document arn:aws:imagebuilder:ap-northeast-1:255485124026:component/inspector-test-linux/1.0.1/1 failed!'

# トラブルシューティングとデバッグを行う

## (2) Systems Manager (SSM) Automation 実行履歴の確認

EC2 Image Builder  
のエラーメッセージをもとに  
SSM Automation のログを確認

SSM execution 'd366feec-a1c9-4218-876e-bb53ae05a421' failed with status = 'Failed' in state = 'TESTING' and failure message = 'Document arn:aws:imagebuilder:ap-northeast-1:255485124026:component/inspector-test-linux/1.0.1/1 failed!'

AWS Systems Manager > 自動化

Executions | Preferences

自動化の実行

実行 ID: d366feec-a1c9-4218-876e-bb53ae05a421 × Clear filters

実行 ID	ドキュメント名	ステータス
d366feec-a1c9-4218-876e-bb53ae05a421	arn:aws:ssm:ap-northeast-1:332908494474:document/ImageBuilderTestImageDocument	⊗ 失敗

### ポイント

SSM Automationで、ドキュメント  
“ImageBuilderTestImageDocument  
”が実行されており、  
そこで失敗していることがわかる。

実行IDをクリックして、  
詳細を表示させる（次ページ）

# トラブルシューティングとデバッグを行う

## (2) Systems Manager Automation 実行履歴の確認

実行ステータス			
全体的なステータス	すべての実行したステップ	# 成功	# 失敗
⊗ 失敗	8	7	1

### 実行したステップ (20)

ステップ ID	ステップ #	ステップ名	アクション	ステータス
f7beb528-880b-4e43-addf-7ae135eb1484	1	WaitForInstanceToSpinUp	aws:waitForAwsResourceProperty	🟢 成功
7b7d161d-0651-451e-8273-09e0dd4c091d	2	VerifySSMAgentBranch	aws:branch	🟢 成功
dc6cc9b4-e9da-46e9-ace8-0e1271487fd6	3	VerifySSMAgentLinux	aws:runCommand	🟢 成功
46500e57-cce2-488a-92f1-87645d48b76d	4	EndOfVerifySSMAgentBranch	aws:sleep	🟢 成功
71ca6f36-d0fe-4682-befc-d90e84352a9c	5	RunTestsBranch	aws:branch	🟢 成功
c59ae451-798e-44cd-ba9f-d3bcd396d4d8	6	RunTestsWithLogging	aws:runCommand	⊗ 失敗
cbe06bfa-ccda-48da-834c-d8838c246fc2	7	FailureHandling	aws:branch	🟢 成功
6fa7d731-1942-4c83-ab6f-020baadfb61c	8	EndOfFailureHandling	aws:sleep	🟢 成功
49781eed-3e75-496b-a70e-6a1d83e58caf	-	WaitBeforeDescribingInstanceAgain	aws:sleep	🕒 保留中

実行 ID

d366feec-a1c9-4218-876e-bb53ae05a421

### ポイント

EC2 Image Builder によって実行されている SSM Automation の全体フローと、問題発生箇所及び内容が確認できる。

# 【参考】 EC2 Image Builderで実行されているSSMドキュメント

実行されているドキュメントは、SSMドキュメントから確認可能

AWS Systems Manager > ドキュメント

Amazon が所有 | 自己所有 | Shared with me | **すべてのドキュメント**

ドキュメント

Q

ドキュメント名のプレフィックス: Equals: Image X

Clear filters

ImageBuilderTestImageDocument ○

ドキュメントタイプ 所有者  
Automation 332908494474

プラットフォームタイプ  
Windows, Linux

デフォルトバージョン  
9

ImageBuilderBuildImageDocument

ドキュメントタイプ 所有者  
Automation 332908494474

プラットフォームタイプ  
Windows, Linux

デフォルトバージョン  
13

## ImageBuilderBuildImageDocument

イメージをビルドする際に実行される。  
37ステップから成るドキュメントで、レシピの内容に応じて実行されるステップが判定される。

## ImageBuilderTestImageDocument

イメージをテストする際に実行される。  
20ステップから成るドキュメントで、レシピの内容に応じて実行されるステップが判定される。

# トラブルシューティングとデバッグを行う

## (3) ビルドコンポーネント実行中に起動されたEC2の調査

前述の通り、イメージをビルドする際にEC2 インスタンスが起動されます。そのインスタンスにアクセス（SSH、RDP、または SSM Session Manager）することで、仮想マシン内のログやダウンロードファイルの実行状況などを確認することができます。

▼ Troubleshooting settings [Info](#)  
Specify settings to troubleshoot issues with building your image.

**Instance settings** [Info](#)  
 Terminate instance on failure

**Key pair** [Info](#)  
Choose a key pair, which allows you to securely connect to your instance.

[Create key pair](#)

**Logs**  
Select a location to save logs.

**S3 location**  
 [Browse S3](#)

### ポイント

EC2 の調査を行う際には、ビルド時に作成されるインスタンスの終了を防ぐために、「Terminate Instance on Failure」オプションのチェックを外します。

# OSイメージ構築/利用に 組み合わせて使うAWSサービス

# User data (ユーザーデータ)

起動時にスクリプト実行を行う機能

2種類の形式でインスタンスに渡す

- シェルスクリプト
- cloud-initディレクティブ

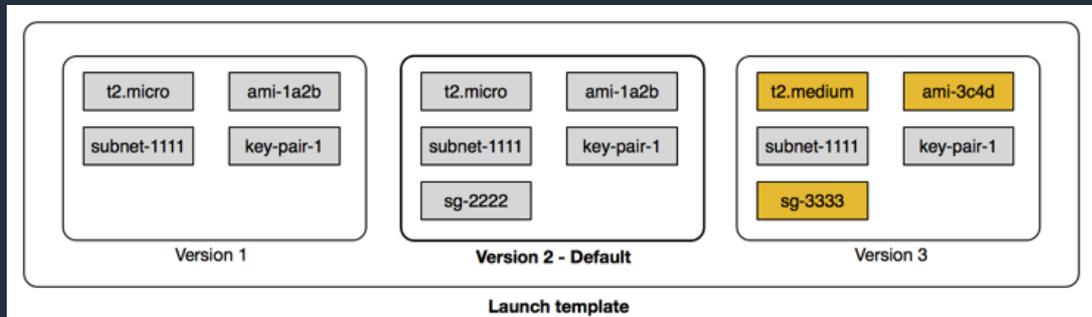
利用用途

- AMIでカバーできない起動時の設定変更
- 起動時に実行するスクリプトやchef, puppetへ、外部からパラメータとして値を渡す

```
#!/bin/bash
yum update -y
amazon-linux-extras install -y lamp-mariadb10.2-php7.2 php7.2
yum install -y httpd mariadb-server
systemctl start httpd
systemctl enable httpd
usermod -a -G apache ec2-user
chown -R ec2-user:apache /var/www
chmod 2775 /var/www
find /var/www -type d -exec chmod 2775 {} \;;
find /var/www -type f -exec chmod 0664 {} \;;
echo "<?php phpinfo(); ?>" > /var/www/html/phpinfo.php
```

# Launch Template (起動テンプレート)

- EC2インスタンス起動時に設定すべき項目（AMI ID やインスタンスタイプ、通常インスタンスの起動に使用しているネットワーク設定）をテンプレート化して、インスタンスの起動をシンプル化することができる
- オートスケーリングやスポットフリート、オンデマンドインスタンスでサポートされる
- 全社標準の設定を適用したり、ベストプラクティスに従ったインスタンスの起動を容易にする



[https://docs.aws.amazon.com/ja\\_jp/AWSEC2/latest/UserGuide/ec2-launch-templates.html](https://docs.aws.amazon.com/ja_jp/AWSEC2/latest/UserGuide/ec2-launch-templates.html)

# 補足情報

# EC2 Image Builder サービスクォータについて

リソース	デフォルト値
現在のリージョンのこのアカウントで進行できる同時ビルドの最大数。	アカウントあたり、リージョンあたり 100 個のビルド
現在のリージョンのアカウントで作成できる EC2 Image Builder コンポーネントの最大数。	アカウントあたり、リージョンあたり 1,000 個のコンポーネント
EC2 Image Builder コンポーネントのデータフィールドの最大サイズ。	16 KB
現在のリージョンのアカウントで作成できる EC2 Image Builder イメージパイプラインの最大数。	アカウントあたり、リージョンあたり 75 個のイメージパイプライン
現在のリージョンのアカウントで作成できる EC2 Image Builder イメージレシピの最大数。	アカウントあたり、リージョンあたり 1,000 個のイメージレシピ
単一の EC2 Image Builder イメージレシピに関連付けることができる EC2 Image Builder コンポーネントの最大数。	イメージあたり、リージョンあたり 20 個のコンポーネント
現在のリージョンのアカウントで作成できる EC2 Image Builder インフラストラクチャ設定の最大数。	アカウントあたり、リージョンあたり 1,000 個の設定
現在のリージョンのアカウントで作成できる EC2 Image Builder ディストリビューション設定の最大数。	アカウントあたり、リージョンあたり 1,000 個の設定

# EC2 Image Builder 発表以降のアップデート

**EC2 Image Builder コンポーネント がローカルで開発可能に**

<https://aws.amazon.com/jp/about-aws/whats-new/2020/08/ec2-image-builder-コンポーネント-can-now-be-developed-locally/>

**EC2 Image Builder から CloudWatch へのログストリーミングが可能に**

<https://aws.amazon.com/jp/about-aws/whats-new/2020/07/ec2-image-builder-can-now-stream-logs-to-cloudwatch/>

**EC2 Image Builder での暗号化された AMI の作成および配信が可能に**

<https://aws.amazon.com/jp/about-aws/whats-new/2020/07/ec2-image-builder-produce-distribute-encrypted-amis/>

**EC2 Image Builder が AWS PrivateLink を介した接続のサポートを開始**

<https://aws.amazon.com/jp/about-aws/whats-new/2020/06/ec2-image-builder-now-supports-connectivity-through-aws-privatelink/>

**EC2 Image Builder に AWS CloudFormation へのサポートを追加**

<https://aws.amazon.com/jp/about-aws/whats-new/2020/05/ec2-image-builder-now-includes-support-for-aws-CloudFormation/>

**EC2 Image Builder が Ubuntu、RHEL、CentOS、SLES のサポートを追加**

<https://aws.amazon.com/jp/about-aws/whats-new/2020/04/ec2-image-builder-adds-support-ubuntu-rhel-centos-sles/>

# まとめ

AWS 公式 Webinar  
<https://amzn.to/JPWebinar>



過去資料  
<https://amzn.to/JPArchive>



# EC2 Image Builder まとめ



## イメージを自動的に作成する仕組みの構築

コードを記述することなく、GUI ウィザードから  
イメージを作成する自動化パイプラインが構築可能



## セキュリティとAMI品質の向上

セキュアなイメージの作成と最新化の実現  
セキュリティ設定の再利用  
本番環境に展開する前にテストを実行して問題を検出



## ハイブリッド対応

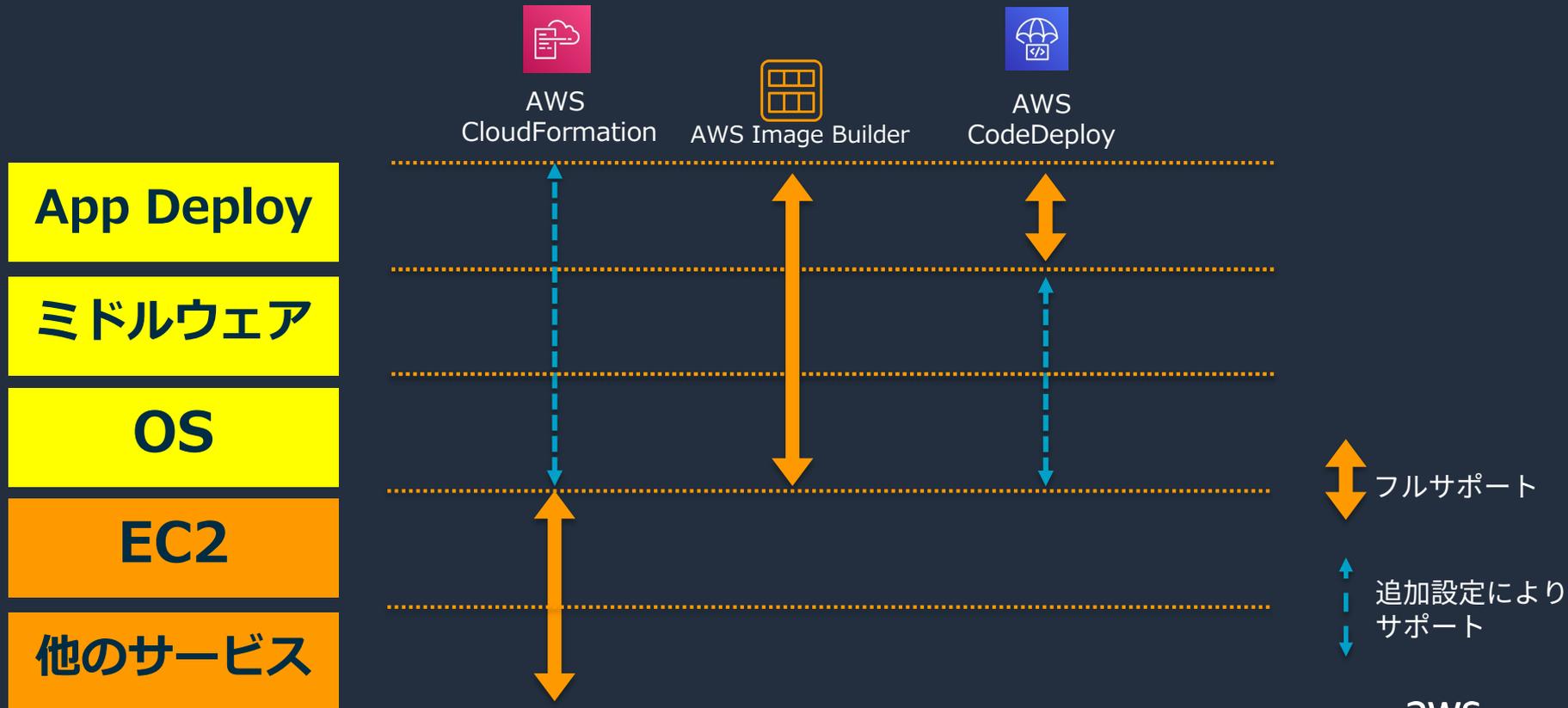
AWSで用いるAMIと、オンプレミスで用いるVMイメージが作成可能



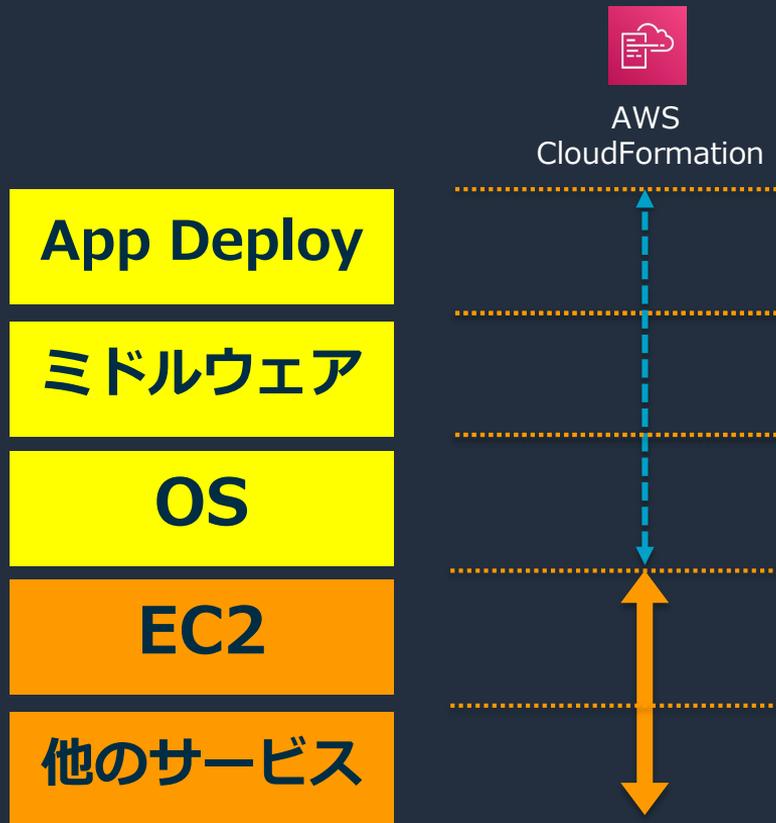
## コスト不要

EC2 Image Builder 自体は無料で利用可能  
イメージの作成、保存、共有するために使用した分の  
AWSリソース料金が別途課金

# AWSのプロビジョニングサービス



# AWSのプロビジョニングサービス



## 8月のBlack Belt Online Seminar



8/26 (水) 18:00-19:00 AWS CloudFormation

AWS CloudFormation はAWSリソースの環境構築を、設定ファイルを元に自動化できるサービスです。本セミナーではAWS CloudFormationの基本的な機能をはじめ、ユースケース、アップデート情報についてもご紹介いたします。

# Q&A

お答えできなかったご質問については

AWS Japan Blog 「<https://aws.amazon.com/jp/blogs/news/>」にて

後日掲載します。

# AWS の日本語資料の場所「AWS 資料」で検索



日本担当チームへお問い合わせ サポート 日本語 ▾ アカウント ▾

コンソールにサインイン

製品 ソリューション 料金 ドキュメント 学習 パートナー AWS Marketplace その他 🔍

## AWS クラウドサービス活用資料集トップ

アマゾン ウェブ サービス (AWS) は安全なクラウドサービスプラットフォームで、ビジネスのスケールと成長をサポートする処理能力、データベースストレージ、およびその他多種多様な機能を提供します。お客様は必要なサービスを選択し、必要な分だけご利用いただけます。それらを活用するために役立つ日本語資料、動画コンテンツを多数ご提供しております。(本サイトは主に、AWS Webinar で使用した資料およびオンデマンドセミナー情報を掲載しています。)

[AWS Webinar お申込 »](#)

[AWS 初心者向け »](#)

[業種・ソリューション別資料 »](#)

[サービス別資料 »](#)

<https://amzn.to/JPArchive>



# AWS Well-Architected 個別技術相談会

毎週“W-A個別技術相談会”を実施中

- AWSのソリューションアーキテクト(SA)に  
対策などを相談することも可能

- **申込みはイベント告知サイトから**

(<https://aws.amazon.com/jp/about-aws/events/>)

AWS イベント

で[検索]



# ご視聴ありがとうございました

AWS 公式 Webinar  
<https://amzn.to/JPWebinar>



過去資料  
<https://amzn.to/JPArchive>



# 参考資料

## Amazon が提供する EC2 Image Builder コンポーネント

AWS 公式 Webinar  
<https://amzn.to/JPWebinar>



過去資料  
<https://amzn.to/JPArchive>



# Amazonが提供するビルドコンポーネント Linux

#	名前	概要	OS	種類
1	dotnet-core-sdk-linux   Version 3.1.0	stalls the Microsoft .NET Core 3.1 SDK and its dependencies from the Microsoft yum repository.	Linux	BUILD
2	dotnet-core-runtime-linux   Version 3.1.2	Installs the Microsoft .NET Core Runtime version 3.1 and its dependencies from the Microsoft package repository. For more information, review the .NET Core 3.1 download page at <a href="https://dotnet.microsoft.com/download/dotnet-core/3.1">https://dotnet.microsoft.com/download/dotnet-core/3.1</a> .	Linux	BUILD
3	dotnet-core-runtime-linux   Version 3.1.1	Description Installs the Microsoft .NET Core Runtime version 3.1 and its dependencies from the Microsoft package repository. For more information, review the .NET Core 3.1 download page at <a href="https://dotnet.microsoft.com/download/dotnet-core/3.1">https://dotnet.microsoft.com/download/dotnet-core/3.1</a> .	Linux	BUILD
4	docker-ce-linux   Version 1.0.0	Description Install the latest Docker Community Edition from Amazon Linux Extras, and enable the ec2-user user to manage docker without using sudo.	Linux	BUILD



# Amazonが提供するビルドコンポーネント Linux

#	名前	概要	OS	種類
5	aws-cli-version-2-linux   Version 1.0.0	Installs the latest version of the AWS CLI version 2, and creates the symlink /usr/bin/aws pointing to the installed application. For more information, review the user guide at <a href="https://docs.aws.amazon.com/cli/latest/userguide/">https://docs.aws.amazon.com/cli/latest/userguide/</a> .	Linux	BUILD
6	amazon-corretto-8-jdk   Version 1.0.0	Installs Amazon Corretto 8 JDK from Amazon Linux Extras	Linux	BUILD
7	amazon-corretto-11   Version 1.0.0	Installs Amazon Corretto 11	Linux	BUILD
8	amazon-corretto-11-headless   Version 1.0.0	Installs Amazon Corretto 11 Headless	Linux	BUILD
9	amazon-cloudwatch-agent-linux   Version 1.0.0	Installs the latest version of the Amazon CloudWatch agent. This component installs only the agent. You must take additional steps to configure and use the Amazon CloudWatch agent. For more information, see the documentation at <a href="https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/install-CloudWatch-Agent-on-EC2-Instance.html">https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/install-CloudWatch-Agent-on-EC2-Instance.html</a> .	Linux	BUILD

# Amazonが提供するビルドコンポーネント Linux

#	名前	概要	OS	種類
10	php-7_4-linux   Version 1.0.0	Installs PHP 7.4 from Amazon Linux Extras.	Linux	BUILD
11	php-7_3-linux   Version 1.0.0	Install PHP 7.3 from Amazon Linux Extras	Linux	BUILD
12	php-7_2-ubuntu   Version 1.0.0	Installs PHP 7.2 from the Ubuntu package repository.	Linux	BUILD
13	php-7_2-linux   Version 1.0.0	Install PHP 7.2 from Amazon Linux Extras	Linux	BUILD
14	php-7_1-linux   Version 1.0.0	Install PHP 7.1 from Amazon Linux Extras	Linux	BUILD
15	mono-linux   Version 1.0.1	Installs the latest version of the Mono framework. Follows the instructions found at <a href="https://www.mono-project.com/">https://www.mono-project.com/</a> .	Linux	BUILD
16	mono-linux   Version 1.0.0	Installs the latest version of the Mono framework. Follows the instructions found at <a href="https://www.mono-project.com/">https://www.mono-project.com/</a> .	Linux	BUILD

# Amazonが提供するビルドコンポーネント Linux

#	名前	概要	OS	種類
17	mate-de-linux   Version 1.20.0	Installs the MATE Desktop Environment, xrdp, TigerVNC server, and enables the xrdp service.	Linux	BUILD
18	dotnet-core-sdk-linux   Version 3.1.2	Installs the Microsoft .NET Core SDK version 3.1 and its dependencies from the Microsoft package repository. For more information, review the .NET Core 3.1 download page at <a href="https://dotnet.microsoft.com/download/dotnet-core/3.1">https://dotnet.microsoft.com/download/dotnet-core/3.1</a> .	Linux	BUILD
19	dotnet-core-sdk-linux   Version 3.1.1	Installs the Microsoft .NET Core SDK version 3.1 and its dependencies from the Microsoft package repository. For more information, review the .NET Core 3.1 download page at <a href="https://dotnet.microsoft.com/download/dotnet-core/3.1">https://dotnet.microsoft.com/download/dotnet-core/3.1</a> .	Linux	BUILD
20	reboot-linux   Version 1.0.1	Reboots the system.	Linux	BUILD
21	python-3-linux   Version 1.0.2	Installs the Python 3 package using apt, yum, or zypper.	Linux	BUILD

# Amazonが提供するビルドコンポーネント Linux

#	名前	概要	OS	種類
22	python-3-linux   Version 1.0.1	Installs the Python 3 package using apt, yum, or zypper.	Linux	BUILD
23	python-3-linux   Version 1.0.0	Installs the latest version of Python 3	Linux	BUILD
24	powershell-yum   Version 1.0.0	Installs the latest version of PowerShell from the Microsoft RedHat repository.	Linux	BUILD
25	powershell-core-linux   Version 6.2.4	Installs PowerShell Core 6.2.4 from the GitHub release package. This follows the 'Installation via Direct Download - Red Hat Enterprise Linux (RHEL) 7' instructions listed here: <a href="https://docs.microsoft.com/en-us/powershell/scripting/install/installing-powershell-core-on-linux?view=powershell-6#red-hat-enterprise-linux-rhel-7">https://docs.microsoft.com/en-us/powershell/scripting/install/installing-powershell-core-on-linux?view=powershell-6#red-hat-enterprise-linux-rhel-7</a> .	Linux	BUILD

# Amazonが提供するビルドコンポーネント Linux

#	名前	概要	OS	種類
26	powershell-core-linux   Version 6.2.3	Installs PowerShell Core 6.2.3 from the Microsoft yum repository. Follows the RHEL instructions listed here: <a href="https://docs.microsoft.com/en-us/powershell/scripting/install/installing-powershell-core-on-linux?view=powershell-6#red-hat-enterprise-linux-rhel-7">https://docs.microsoft.com/en-us/powershell/scripting/install/installing-powershell-core-on-linux?view=powershell-6#red-hat-enterprise-linux-rhel-7</a> .	Linux	BUILD
27	update-linux   Version 1.0.0	Updates Linux with the latest security updates.	Linux	BUILD
28	update-linux-kernel-mainline   Version 1.0.0	Installs the latest mainline release of the Linux kernel. For Amazon Linux 2, this will install the 'kernel-ng' package from Amazon Linux Extras. For CentOS 7 and Red Hat Enterprise Linux 7 and 8, this will install the 'kernel-ml' package from <a href="https://www.elrepo.org">https://www.elrepo.org</a> .	Linux	BUILD

# Amazonが提供するビルドコンポーネント Linux

#	名前	概要	OS	種類
29	stig-build-linux-medium   Version 2.6.0	Applies the medium and low severity STIG settings for Red Hat Enterprise Linux (RHEL) to Amazon Linux 2 instances. For more information, see <a href="https://docs.aws.amazon.com/imagebuilder/latest/userguide/image-builder-stig.html">https://docs.aws.amazon.com/imagebuilder/latest/userguide/image-builder-stig.html</a> .	Linux	BUILD
30	stig-build-linux-medium   Version 2.5.0	Applies the medium severity STIG settings to Amazon Linux 2. Uses the STIG settings from Red Hat Enterprise Linux (RHEL).	Linux	BUILD
31	stig-build-linux-low   Version 2.6.0	Applies the low severity STIG settings for Red Hat Enterprise Linux (RHEL) to Amazon Linux 2 instances. For more information, see <a href="https://docs.aws.amazon.com/imagebuilder/latest/userguide/image-builder-stig.html">https://docs.aws.amazon.com/imagebuilder/latest/userguide/image-builder-stig.html</a> .	Linux	BUILD
32	stig-build-linux-low   Version 2.5.0	Applies the low severity STIG settings to Amazon Linux 2. Uses the STIG settings from Red Hat Enterprise Linux (RHEL).	Linux	BUILD

# Amazonが提供するビルドコンポーネント Linux

#	名前	概要	OS	種類
33	stig-build-linux-high   Version 2.6.0	Applies the high, medium, and low severity STIG settings for Red Hat Enterprise Linux (RHEL) to Amazon Linux 2 instances. For more information, see <a href="https://docs.aws.amazon.com/imagebuilder/latest/userguide/image-builder-stig.html">https://docs.aws.amazon.com/imagebuilder/latest/userguide/image-builder-stig.html</a> .	Linux	BUILD
34	stig-build-linux-high   Version 2.5.0	Applies the high severity STIG settings to Amazon Linux 2. Uses the STIG settings from Red Hat Enterprise Linux (RHEL).	Linux	BUILD

# Amazonが提供するビルドコンポーネント Windows

#	名前	概要	OS	種類
1	dotnet-core-runtime-windows   Version 3.1.0	Installs the Microsoft .NET Core Runtime version 3.1.2. For more information, review the .NET Core 3.1 download page at <a href="https://dotnet.microsoft.com/download/dotnet-core/3.1">https://dotnet.microsoft.com/download/dotnet-core/3.1</a> .	Windows	BUILD
2	dotnet-core-hosting-bundle-windows   Version 3.1.0	Installs the ASP.NET Core Hosting Bundle version 3.1.2. The hosting bundle includes the .NET Core Runtime and IIS support. For more information, review the .NET Core 3.1 download page at <a href="https://dotnet.microsoft.com/download/dotnet-core/3.1">https://dotnet.microsoft.com/download/dotnet-core/3.1</a> .	Windows	BUILD
3	aws-cli-version-2-windows   Version 1.0.0	Installs the latest version of the AWS CLI version 2. For more information, review the user guide at <a href="https://docs.aws.amazon.com/cli/latest/userguide/">https://docs.aws.amazon.com/cli/latest/userguide/</a> .	Windows	BUILD
4	anaconda-windows   Version 2019.10.0	Installs the Anaconda distribution and environments for Tensorflow, PyTorch, and MXNet.	Windows	BUILD

# Amazonが提供するビルドコンポーネント Windows

#	名前	概要	OS	種類
5	amazon-corretto-8-windows   Version 1.0.0	Installs Amazon Corretto 8 for Windows in accordance with the Amazon Corretto 8 User Guide at <a href="https://docs.aws.amazon.com/corretto/latest/corretto-8-ug/windows-7-install.html">https://docs.aws.amazon.com/corretto/latest/corretto-8-ug/windows-7-install.html</a> .	Windows	BUILD
6	amazon-corretto-11-windows   Version 1.0.0	Installs Amazon Corretto 11 for Windows in accordance with the Amazon Corretto 11 User Guide at <a href="https://docs.aws.amazon.com/corretto/latest/corretto-11-ug/windows-7-install.html">https://docs.aws.amazon.com/corretto/latest/corretto-11-ug/windows-7-install.html</a> .	Windows	BUILD
7	amazon-cloudwatch-agent-windows   Version 1.0.0	Installs the latest version of the Amazon CloudWatch agent. This component installs only the agent. You must take additional steps to configure and use the Amazon CloudWatch agent. For more information, see the documentation at <a href="https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/install-CloudWatch-Agent-on-EC2-Instance.html">https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/install-CloudWatch-Agent-on-EC2-Instance.html</a> .	Windows	BUILD

# Amazonが提供するビルドコンポーネント Windows

#	名前	概要	OS	種類
8	dotnet-core-sdk-windows   Version 3.1.0	Installs the Microsoft .NET Core SDK version 3.1.102. The installation includes version 3.1.2 of the ASP.NET Core Runtime, the .NET Core Runtime, and the Desktop Runtime. For more information, review the .NET Core 3.1 download page at <a href="https://dotnet.microsoft.com/download/dotnet-core/3.1">https://dotnet.microsoft.com/download/dotnet-core/3.1</a> .	Windows	BUILD
9	reboot-windows   Version 1.0.1	Reboots the system.	Windows	BUILD
10	python-3-windows   Version 3.8.2	Installs Python 3.8.2 for Windows.	Windows	BUILD
11	PuTTY   Version 0.73.1	Installs PuTTY version 0.73 from <a href="https://www.chiark.greenend.org.uk/~sgtatham/putty/releases/0.73.html">https://www.chiark.greenend.org.uk/~sgtatham/putty/releases/0.73.html</a> .	Windows	BUILD
12	PuTTY   Version 0.73.0	Description Installs PuTTY	Windows	BUILD

# Amazonが提供するビルドコンポーネント Windows

#	名前	概要	OS	種類
13	powershell-windows   Version 7.0.2	Installs PowerShell version 7.0.2 from the GitHub repository at <a href="https://github.com/PowerShell/PowerShell">https://github.com/PowerShell/PowerShell</a> .	Windows	BUILD
14	powershell-windows   Version 7.0.1	Installs PowerShell version 7.0.1 from the GitHub repository at <a href="https://github.com/PowerShell/PowerShell">https://github.com/PowerShell/PowerShell</a> .	Windows	BUILD
15	powershell-windows   Version 7.0.0	Installs PowerShell version 7.0.0 from the GitHub repository at <a href="https://github.com/PowerShell/PowerShell">https://github.com/PowerShell/PowerShell</a> .	Windows	BUILD
16	powershell-core-windows   Version 6.2.4	Performs a default installation of PowerShell Core 6.2.4 from the <a href="https://github.com/PowerShell/PowerShell">https://github.com/PowerShell/PowerShell</a> repository.	Windows	BUILD
17	powershell-core-windows   Version 6.2.3	Performs a default installation of PowerShell Core 6.2.3 from the <a href="https://github.com/PowerShell/PowerShell">https://github.com/PowerShell/PowerShell</a> repository.	Windows	BUILD

# Amazonが提供するビルドコンポーネント Windows

#	名前	概要	OS	種類
18	update-windows   Version 1.0.0	Updates Windows with the latest security updates.	Windows	BUILD
19	stig-build-windows-medium   Version 1.1.0	Applies the medium and low severity STIG settings to Windows instances. For more information, see <a href="https://docs.aws.amazon.com/imagebuilder/latest/userguide/image-builder-stig.html">https://docs.aws.amazon.com/imagebuilder/latest/userguide/image-builder-stig.html</a> .	Windows	BUILD
20	stig-build-windows-medium   Version 1.0.0	Applies the medium severity STIG settings to Windows.	Windows	BUILD
21	stig-build-windows-low   Version 1.1.0	Applies the low severity STIG settings to Windows instances. For more information, see <a href="https://docs.aws.amazon.com/imagebuilder/latest/userguide/image-builder-stig.html">https://docs.aws.amazon.com/imagebuilder/latest/userguide/image-builder-stig.html</a> .	Windows	BUILD
22	stig-build-windows-low   Version 1.0.0	Applies the low severity STIG settings to Windows	Windows	BUILD

# Amazonが提供するビルドコンポーネント Windows

#	名前	概要	OS	種類
23	stig-build-windows-high   Version 1.1.0	Applies the high, medium, and low severity STIG settings to Windows instances. For more information, see <a href="https://docs.aws.amazon.com/imagebuilder/latest/userguide/image-builder-stig.html">https://docs.aws.amazon.com/imagebuilder/latest/userguide/image-builder-stig.html</a> .	Windows	BUILD
24	stig-build-windows-high   Version 1.0.0	Applies the high severity STIG settings to Windows.	Windows	BUILD

# Amazonが提供するテストコンポーネント Linux

#	名前	概要	OS	種類
1	eni-attachment-test-linux   Version 1.0.1	The ENI attachment test creates an ENI and attaches it to the instance. It validates that the attached ENI has an IP address. It then detaches and deletes the ENI. To perform this test, an IAM policy with the following actions is required: ec2:AttachNetworkInterface, ec2:CreateNetworkInterface, ec2:CreateTags, ec2>DeleteNetworkInterface, ec2:DescribeNetworkInterface, ec2:DescribeNetworkInterfaceAttribute, and ec2:DetachNetworkInterface.	Linux	TEST
2	eni-attachment-test-linux   Version 1.0.0	The ENI attachment test creates an ENI and attaches it to the instance. It validates the attached ENI has an IP address. It then detaches and deletes the ENI. In order to perform this test, an IAM policy with ec2:AttachNetworkInterface, ec2:CreateTags, ec2:DescribeNetworkInterfaceAttribute, ec2:DetachNetworkInterface, eni:CreateNetworkInterface, eni>DeleteNetworkInterface and eni:DescribeNetworkInterface actions is required.	Linux	TEST

# Amazonが提供するテストコンポーネント Linux

#	名前	概要	OS	種類
3	simple-boot-test-linux   Version 1.0.0	Executes a simple boot test.	Linux	TEST
4	reboot-test-linux   Version 1.0.0	Tests whether the system can reboot successfully	Linux	TEST
5	reboot-linux   Version 1.0.0	Reboots the system.	Linux	TEST
6	validate-ssh-public-key-linux   Version 1.0.0	Checks if the latest ssh public key provided by the Instance Metadata is present on instance. This test does not assert that public key from Instance Metadata is the only key present.	Linux	TEST
7	validate-ssh-host-key-generation-linux   Version 1.0.0	Checks if the ssh host key was generated after latest boot.	Linux	TEST

# Amazonが提供するテストコンポーネント Linux

#	名前	概要	OS	種類
8	ebs-volume-usage-test-linux  Version 1.0.0	The EBS volume usage test creates an EBS volume and attaches it to the instance. It creates a temporary file on the volume and detaches the volume. It reattaches the volume and validates the file exists. It then detaches and deletes the volume. In order to perform this test, an IAM policy with ec2:AttachVolume, ec2:CreateTags, ec2:DetachVolume, ebs:CreateVolume, ebs>DeleteVolume and ebs:DescribeVolumes actions is required.	Linux	TEST
9	ebs-volume-usage-test-linux  Version 1.0.1	The EBS volume usage test creates an EBS volume and attaches it to the instance. It creates a temporary file on the volume and detaches the volume. It reattaches the volume and validates that the file exists. It then detaches and deletes the volume. To perform this test, an IAM policy with the following actions is required: ec2:AttachVolume, ec2:CreateTags, ec2:CreateVolume, ec2>DeleteVolume, ec2:DescribeVolumes, and ec2:DetachVolume.	Linux	TEST

# Amazonが提供するテストコンポーネント Windows

#	名前	概要	OS	種類
1	eni-attachment-test-windows   Version 1.0.1	The ENI attachment test creates an ENI and attaches it to the instance. It validates that the attached ENI has an IP address. It then detaches and deletes the ENI. To perform this test, an IAM policy with the following actions is required: ec2:AttachNetworkInterface, ec2:CreateNetworkInterface, ec2:CreateTags, ec2>DeleteNetworkInterface, ec2:DescribeNetworkInterface, ec2:DescribeNetworkInterfaceAttribute, and ec2:DetachNetworkInterface.	Windows	TEST
2	eni-attachment-test-windows   Version 1.0.0	The ENI attachment test creates an ENI and attaches it to the instance. It validates the attached ENI has an IP address. It then detaches and deletes the ENI. In order to perform this test, an IAM policy with ec2:AttachNetworkInterface, ec2:CreateTags, ec2:DescribeNetworkInterfaceAttribute, ec2:DetachNetworkInterface, eni:CreateNetworkInterface, eni>DeleteNetworkInterface and eni:DescribeNetworkInterface actions is required.	Windows	TEST

# Amazonが提供するテストコンポーネント Windows

#	名前	概要	OS	種類
3	ec2-network-route-test-windows   Version 1.0.0	Test to ensure all required EC2 network routes exist in the route table.	Windows	TEST
4	ebs-volume-usage-test-windows   Version 1.0.1	The EBS volume usage test creates an EBS volume and attaches it to the instance. It creates a temporary file on the volume and detaches the volume. It reattaches the volume and validates that the file exists. It then detaches and deletes the volume. To perform this test, an IAM policy with the following actions is required: ec2:AttachVolume, ec2:CreateTags, ec2:CreateVolume, ec2>DeleteVolume, ec2:DescribeVolumes, and ec2:DetachVolume.	Windows	TEST

# Amazonが提供するテストコンポーネント Windows

#	名前	概要	OS	種類
5	ebs-volume-usage-test-windows   Version 1.0.0	The EBS volume usage test creates an EBS volume and attaches it to the instance. It creates a temporary file on the volume and detaches the volume. It reattaches the volume and validates the file exists. It then detaches and deletes the volume. In order to perform this test, an IAM policy with ec2:AttachVolume, ec2:CreateTags, ec2:DetachVolume, ebs:CreateVolume, ebs>DeleteVolume and ebs:DescribeVolumes actions is required.	Windows	TEST
6	reboot-windows   Version 1.0.0	Reboots the system.	Windows	TEST
7	reboot-test-windows   Version 1.0.0	Tests whether the system can reboot successfully	Windows	TEST
8	windows-activation-test   Version 1.0.0	Checks the Common Information Model for Windows license status	Windows	TEST
9	simple-boot-test-windows   Version 1.0.0	Executes a simple boot test.	Windows	TEST

# Thank You

