Amazon Managed Service for Prometheus (AMP)

堀内 保大

AWS Japan G.K.
Solutions Architect
2023/01

AWS Black Belt Online Seminarとは

- 「サービス別」「ソリューション別」「業種別」などのテーマに分け、アマゾン ウェブ サービス ジャパン合同会社が提供するオンラインセミナーシリーズです
- AWSの技術担当者が、AWSの各サービスやソリューションについてテーマごとに動画を公開します
- 動画を一時停止・スキップすることで、興味がある分野・項目だけの聴講も可能、スキマ時間の学習にもお役立ていただけます
- 以下のURLより、過去のセミナー含めた資料などをダウンロードすることができます
  - https://aws.amazon.com/jp/aws-jp-introduction/aws-jp-webinar-service-cut/
  - https://www.youtube.com/playlist?list=PLzWGOASvSx6FIwIC2X1nObr1KcMCBBlqY

内容についての注意点

- 本資料では2023年1月時点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(https://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います。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

自己紹介

名前：堀内 保大

所属：Solutions Architect, DNB-IM

経歴： SIerで性能関連の技術支援

好きなAWSサービス：

- Amazon Elastic Kubernetes Service (EKS)
- AWS Fargate
- Amazon Managed Service for Prometheus (AMP)

本セミナーの対象者

前提知識

- Prometheus についての基礎的な知識がある

対象者

- Amazon Managed Service for Prometheus をこれからご利用予定の方
- Amazon ECS や Amazon EKS 等のコンテナサービスの監視に興味のある方

アジェンダ

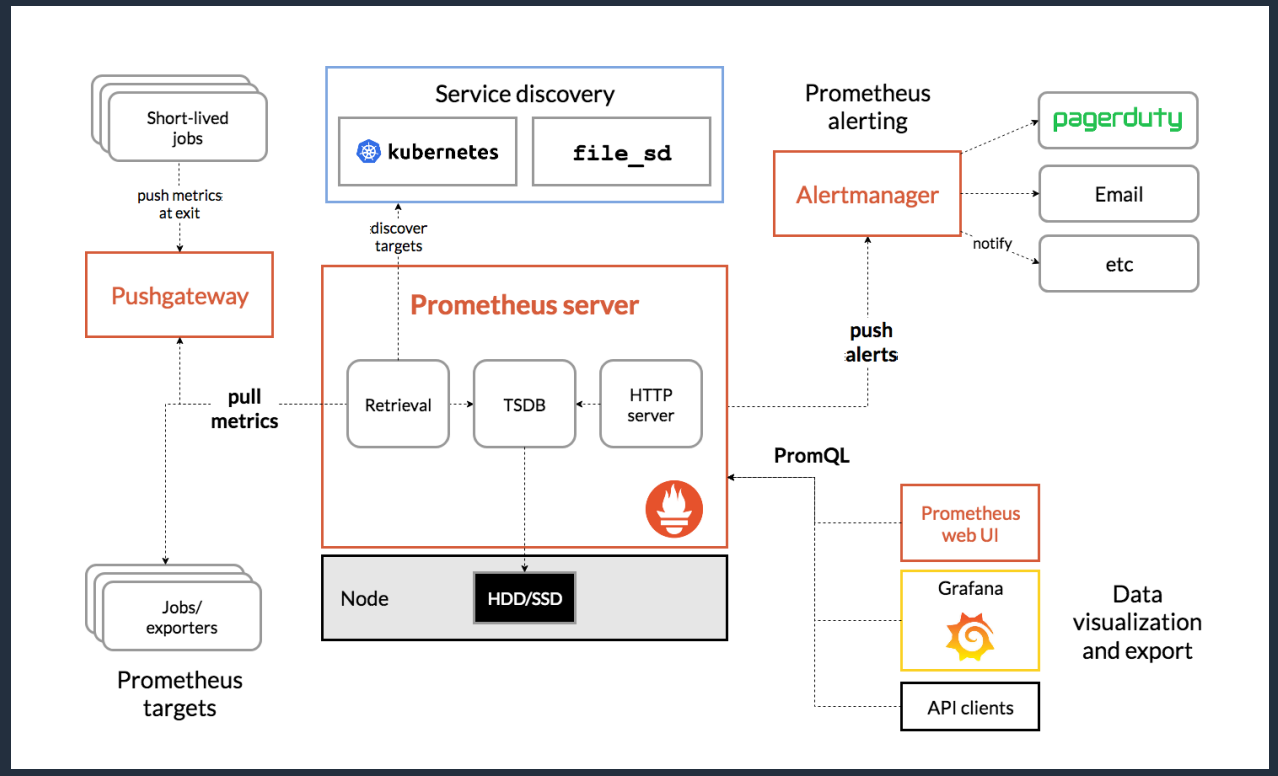
1. Prometheus とは？
2. Amazon Managed Service for Prometheus (AMP)とは？
3. 主要な機能
4. Amazon ECS や Amazon EKS との連携
5. Tips
6. クオータと料金
7. まとめ

Prometheus とは？

Prometheus の利点

モニタリング・アラートのためのオープンソースのソリューション

2016年にKubernetesに次ぐ2番目のプロジェクトとしてCloud Native Computing Foundation (CNCF)に参加、2018年にはGraduatedに。



- エフェメラルなコンテナと相性が良い
  - Service Discovery でメトリクス収集の対象を発見
- プル型のメトリクス収集
- Kubernetes と一緒に広く使われている
  - Kubernetes および多くのアドオンが Prometheus 形式でメトリクスを公開している
- 活発なコニュミティ
  - 多様な exporter やクライアントライブラリ
- 強力なクエリ言語 (PromQL)

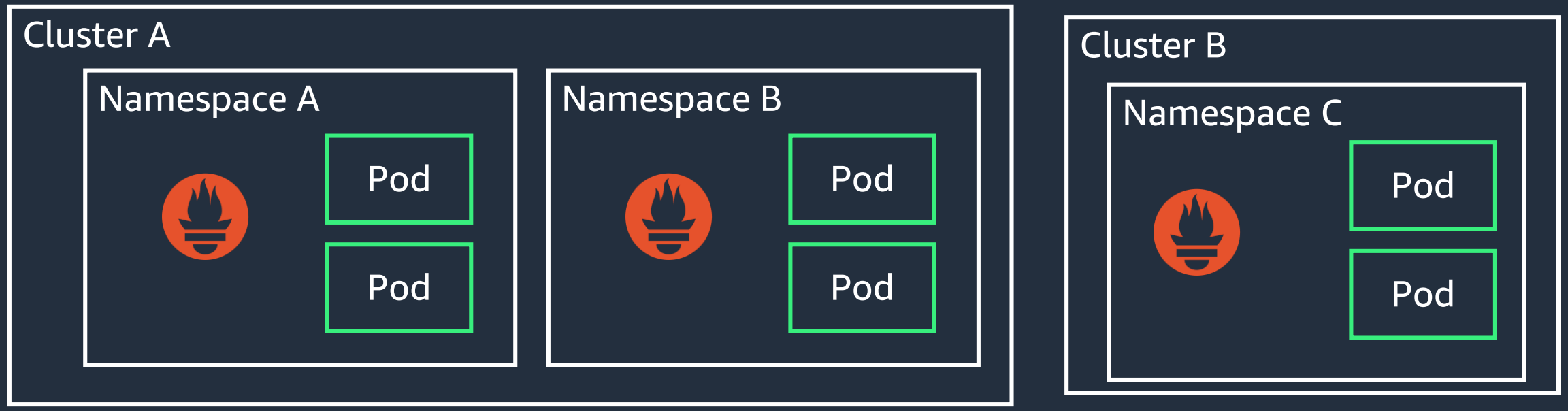
https://prometheus.io/docs/introduction/overview/

Prometheus の運用における課題 (1/2)

1. ストレージおよび API の可用性・スケーラビリティの確保が困難

→水平にスケールする機能を持たないデザイン

- 保存するメトリクスが増えるとリソース消費量が増える
- Prometheusサーバーを増強するか、アプリ、リージョン、チーム毎に別のPrometheusサーバーを用意する

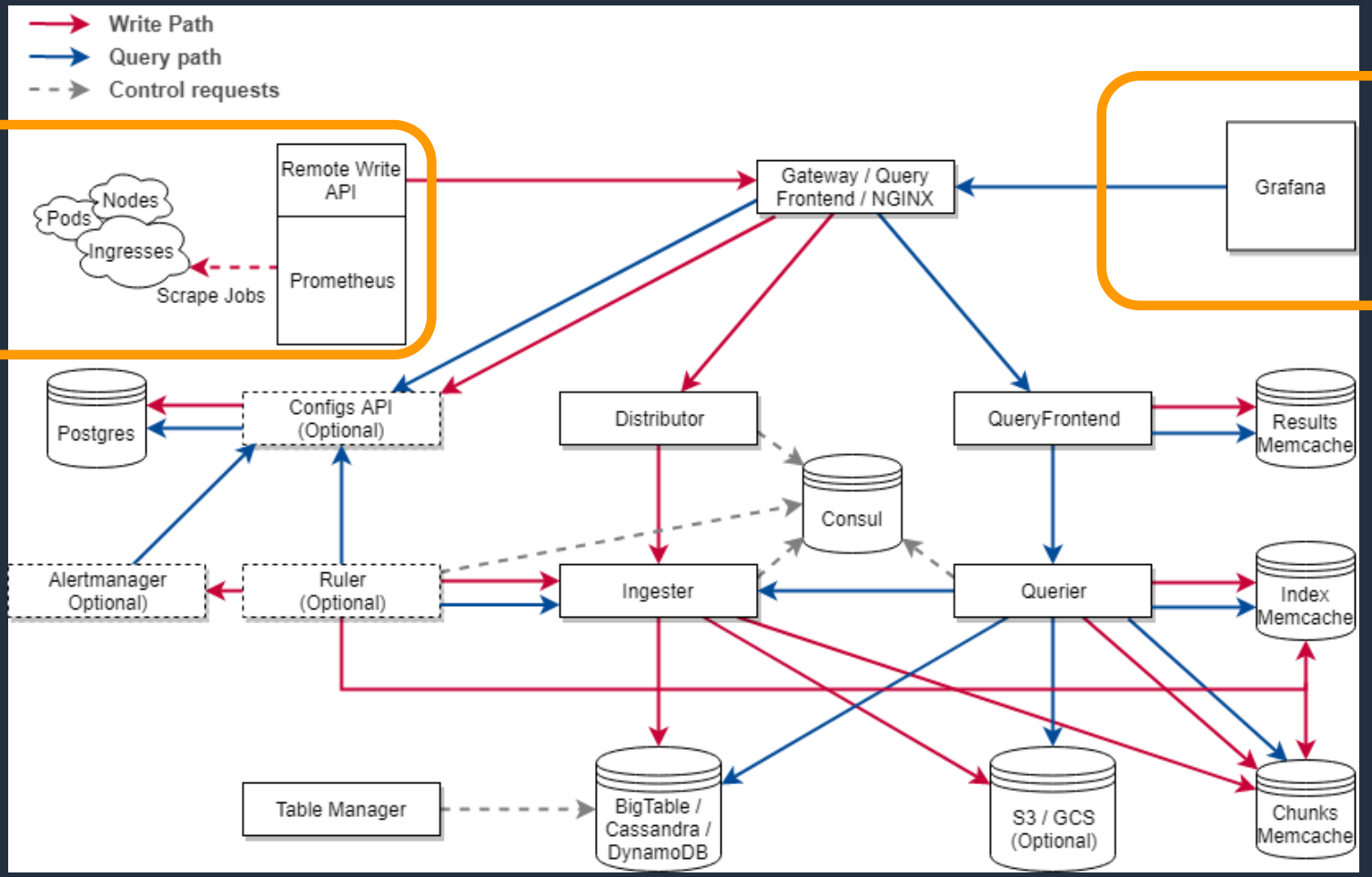


Prometheus の運用における課題 (1/2)
Prometheus リモートストレージによる解決

- Prometheus には Remote Write という外部ストレージにメトリクスを書き込む機能が存在
- 長期保存、可用性、スケーラビリティについては、Prometheus 互換のリモートストレージにデータを書き出すことにより解決が可能
  - https://prometheus.io/docs/prometheus/latest/storage/#remote-storage-integrations

Prometheus の運用における課題 (1/2)
Prometheus リモートストレージによる解決

例：Cortex によるスケールする Prometheus 構成



https://cortexmetrics.io/docs/architecture/

Prometheus の運用における課題 (2/2)

2. 運用上考慮すべき事項があり、運用工数がかかる

- クエリ応答時間やコスト削減のためのメモリ・ストレージの最適化
- Prometheus サーバーの継続的なメンテナンス
- メトリクスの収集・クエリに対する認可・統制の仕組みの実装

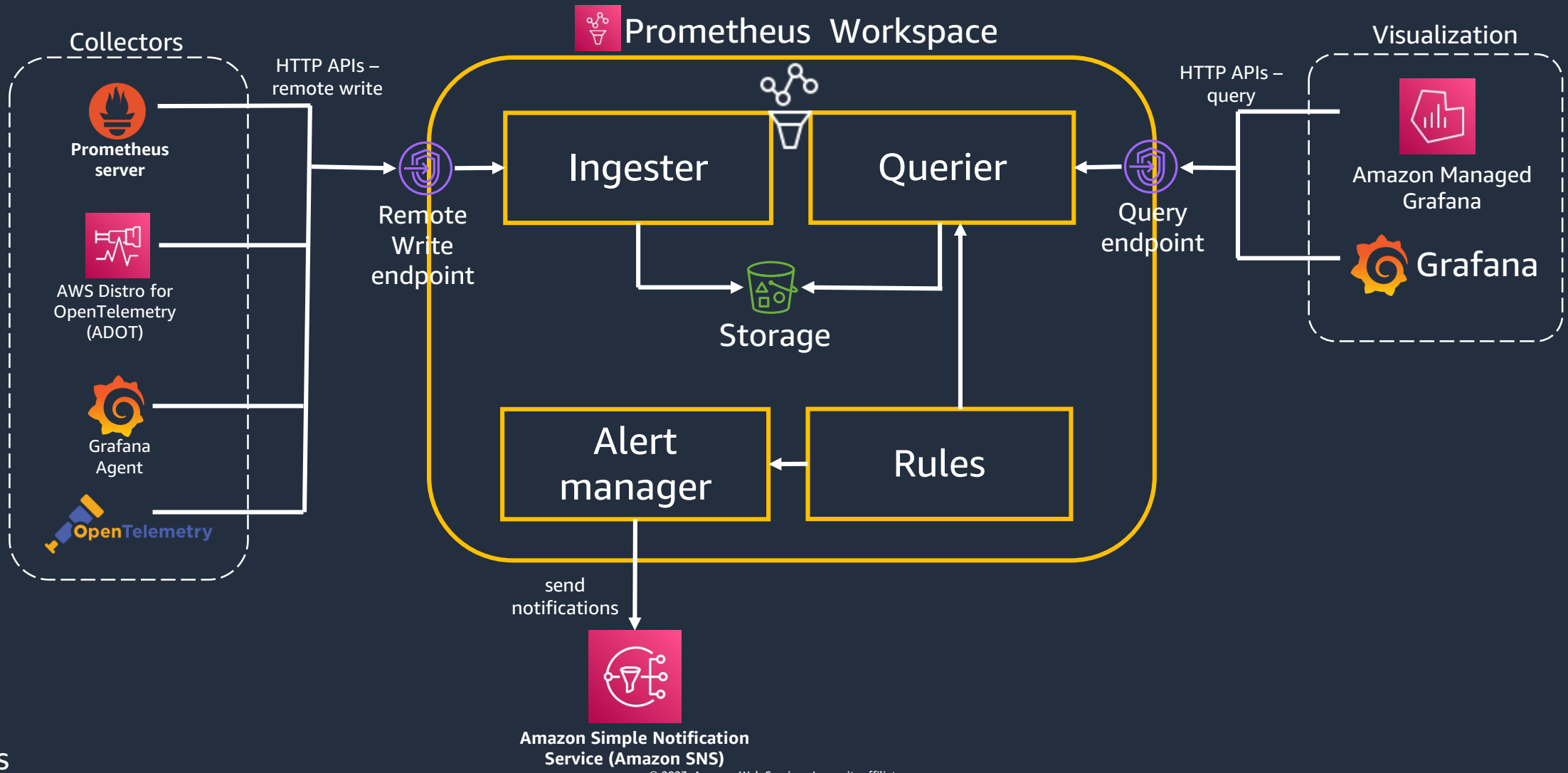
Amazon Managed Service for Prometheus とは？

Amazon Managed Service for Prometheus (AMP) とは？

フルマネージドの Prometheus 互換サービス

- 高いスケーラビリティ、マルチ AZ 配置による高可用性
- フルマネージド、サーバーレス
  - Prometheus サーバーの構成、アップグレードなどの差別化につながらない面倒な作業を削減
- Prometheus 互換
  - Prometheus と同じデータモデル、クエリ言語を使用
  - remote write によるアクセス
- IAM によるセキュアなアクセス
- メトリクスの収集量、クエリ量に応じた従量課金

AMP のアーキテクチャ

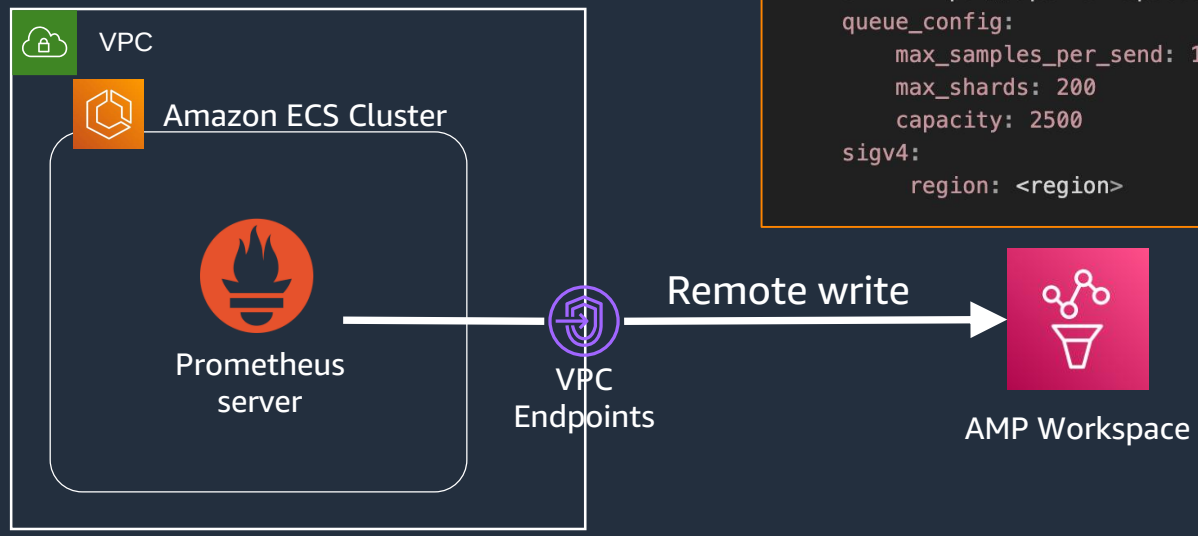


主要な機能

メトリクスの収集

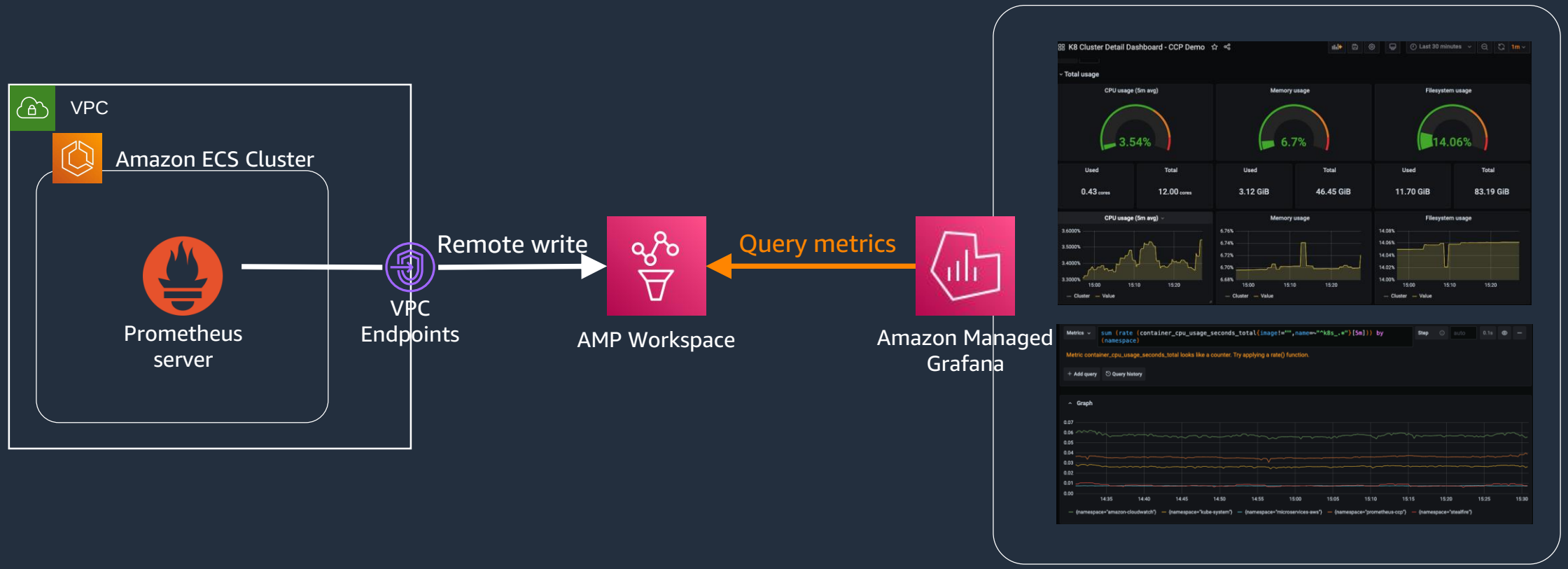
- Prometheus サーバー、OpenTelemetry、Grafana Agent から AMP の Remote Write エンドポイント経由で書き込む
  - ストレージは自動でスケール
  - パブリックエンドポイントに加え VPC Endpoint も利用可能

```
remote_write:
  - url: https://aps-workspaces.<region>.amazonaws.com/workspaces/<workspace-id>/api/v1/remote_write
    queue_config:
        max_samples_per_send: 1000
        max_shards: 200
        capacity: 2500
    sigv4:
         region: <region>
```



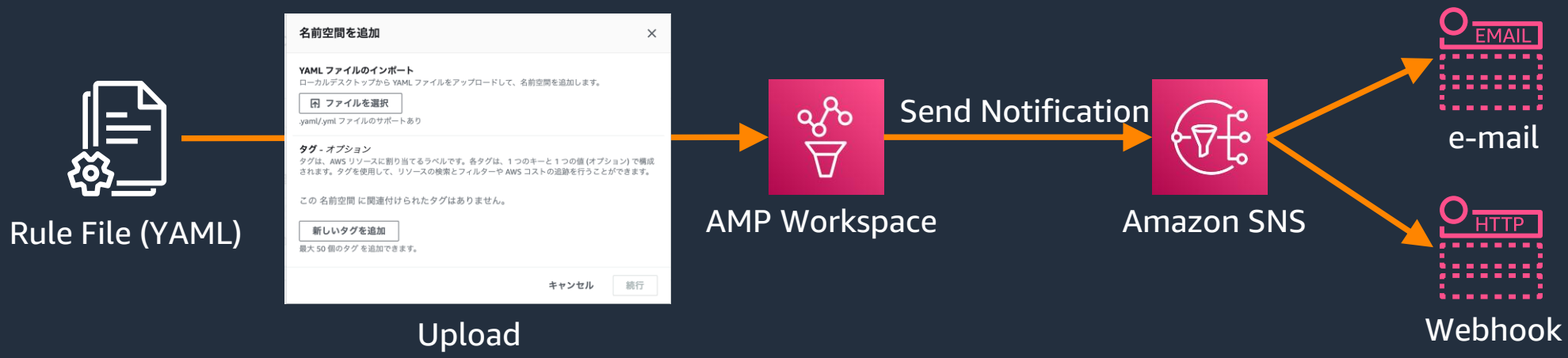
メトリクスの可視化

- Amazon Managed Grafana (AMG) やセルフホストな Grafana 等 Prometheus 互換 API をサポートするツールと連携可能
- AMP ワークスペースのクエリエンドポイントをGrafanaのデータソースに設定



アラートマネージャーとルール管理機能

- 記録ルール (Recording Rules) と アラートルール (Alerting Rules)をサポート
  - YAML 形式のルールファイルをワークスペースにアップロードして設定
  - アラートの通知先として Amazon Simple Notification Service (Amazon SNS) をサポート



アラートルールの通知

API とアクセスコントロール

- コントロールプレーン・データプレーンに対して、
  AWS Identity and Access Management (IAM) によるアクセス管理を設定

IAM actions

データプレーン

- aps:RemoteWrite
- aps:QueryMetrics
- aps:GetLabels
- aps:GetSeries
- aps:GetMetricMetadata

コントロールプレーン

- aps:CreateWorkspace
- aps:UpdateWorkspaceAlias
- aps:DeleteWorkspace
- aps:DescribeWorspaces
- aps:ListWorkspaces

HTTP APIs

Action	HTTP verb	URI
RemoteWrite	POST	/workspaces/{workspaceId}/api/v1/remote_write
QueryMetrics	GET, POST	/workspaces/{workspaceId}/api/v1/query /workspaces/{workspaceId}/api/v1/query_range
GetLabels	GET, POST	/workspaces/{workspaceId}/api/v1/labels /workspaces/{workspaceId}/api/v1/label/{name}/values (only supports GET)
GetSeries	GET, POST	/workspaces/{workspaceId}/api/v1/series
GetMetricMetadata	GET	/workspaces/{workspaceId}/api/v1/metadata

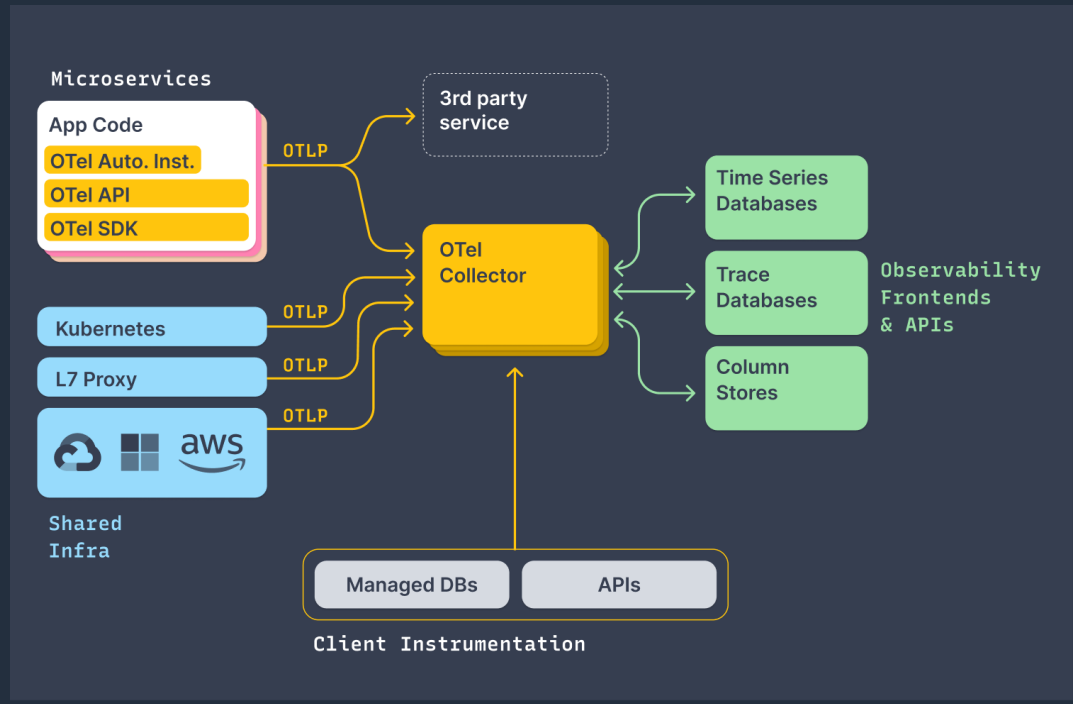
Amazon ECS や Amazon EKS との連携

ADOT Collector による AMP へのメトリクス送信 (1/2)

OpenTelemetry / ADOT とは

https://opentelemetry.io/

- OpenTelemetry とは
  - Cloud Native Computing Foundation (CNCF) でホストされるオープンソースプロジェクト
  - メトリクス、ログ、トレースなどのテレメトリーデータの計装 (インストルメンテーション) と送信の標準仕様と実装を提供
  - 具体的には以下を提供
    - Cross-language な仕様
    - Collector エージェント
    - 各言語の SDK
    - 自動計装 (オートインストルメンテーション)
- AWS Distro for OpenTelemetry (ADOT) とは
  - AWS がサポートするセキュアで Production Ready な OpenTelemetry ディストリビューション
  - AWS 上で使いやすいように機能を拡張

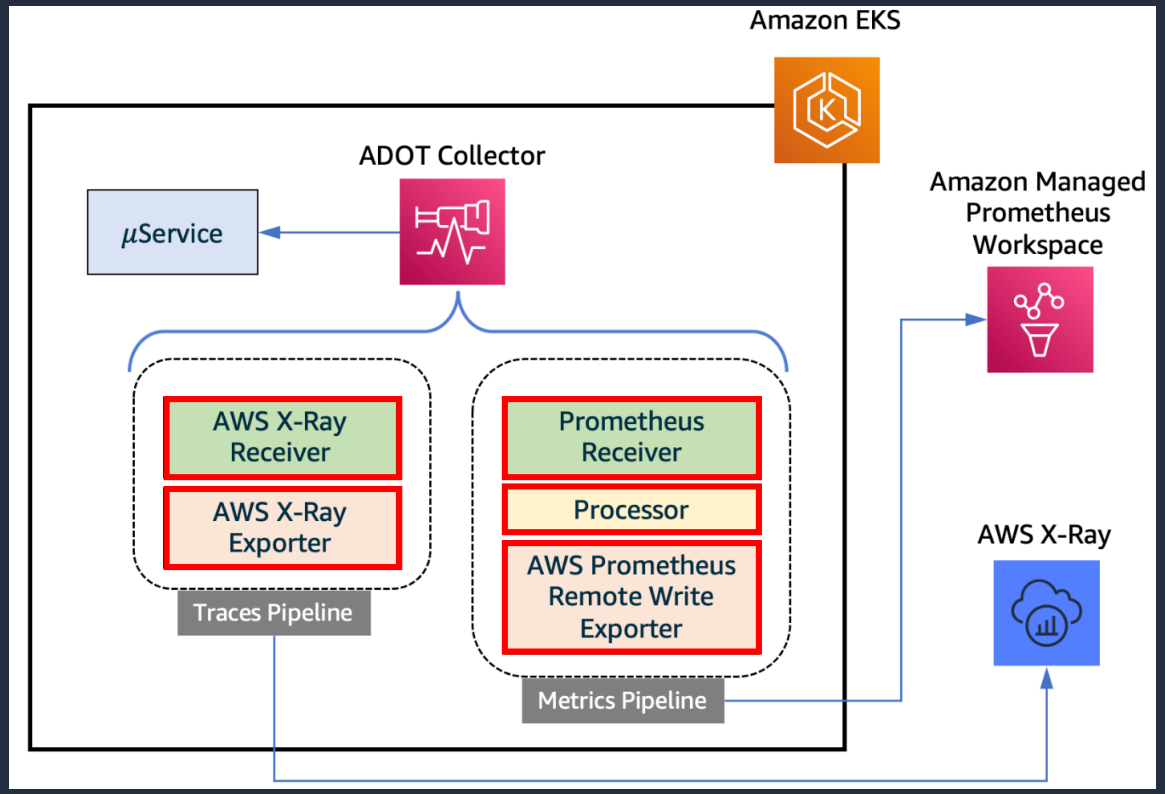


https://aws-otel.github.io/

ADOT Collector による AMP へのメトリクス送信 (2/2)

ADOT Collector とは

- テレメトリデータの収集、整形、バックエンドへの送信のためのエージェント
- Collector を分離することで、各言語 SDK が各バックエンドに向けた実装を持たなくてよくなる
- 3 つの要素でパイプラインを構成
  - Receiver
    - 特定のフォーマットでデータを受け取る
    - プル型とプッシュ型
    - E.g. ecs-metrics, Prometheus, OTLP
  - Processor
    - データのバッチ処理、フィルタリング、変換など
  - Exporter
    - テレメトリーデータをバックエンドに送信
    - E.g. AMP, EMF (CloudWatch, Container Insights)

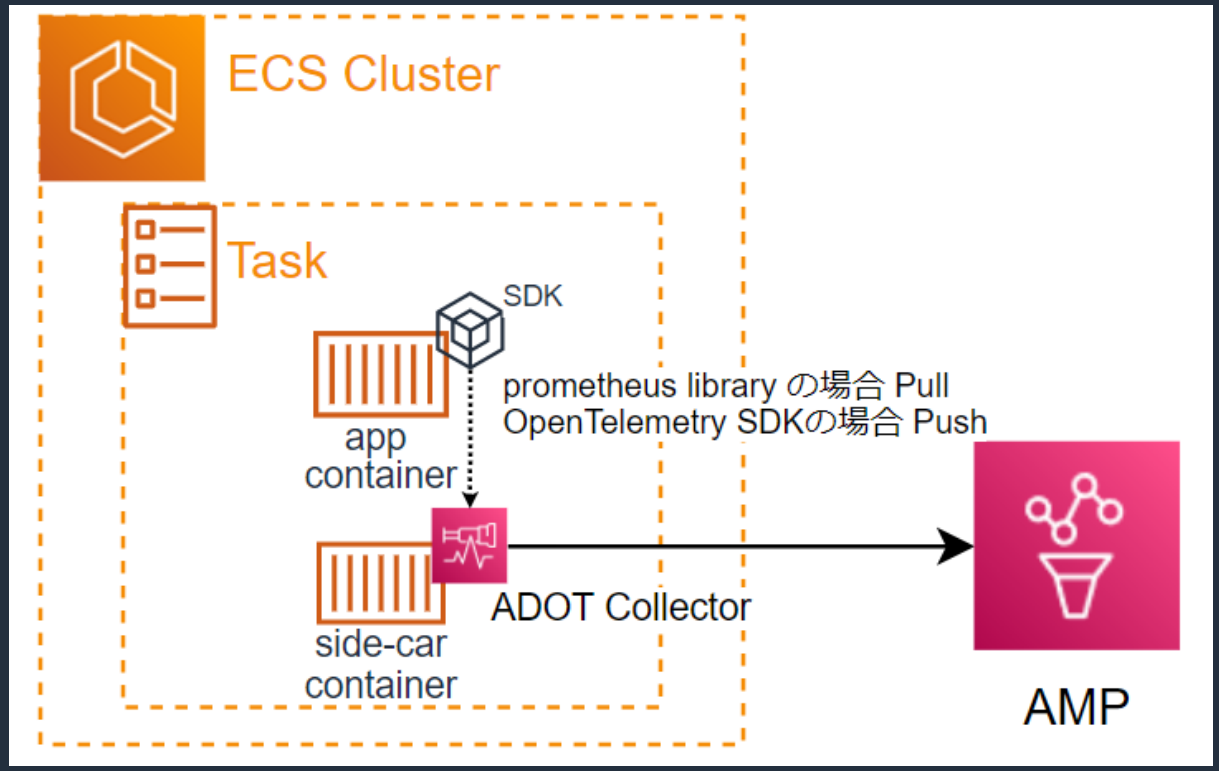



```
extensions:
  sigv4auth:
    service: "aps"
    region: "user-region"

exporters:
  prometheusremotewrite:
    endpoint: "https://aws-managed-prometheus-endpoint/v1/api/remote_write"
    auth:
      authenticator: sigv4auth
```

ECS のメトリクスを AMP に送信する (1/5)

- ECS は、タスク/コンテナレベルの CPU / Memory / Network / Storage メトリクス と カスタムメトリクス の AMP へのエクスポートをサポート
- ADOT Collector をサイドカーコンテナとしてデプロイすることで実現
  - サイドカーコンテナのデプロイ方法は2通り
    - 自分でタスク定義を記述する（参考）
    - ECS New Console から自動でインジェストする（後述）
  - ADOT の ecs-metrics Receiver により ECS のメトリクスを収集
  - カスタムメトリクスは別途公開が必要
    - Prometheus library
    - OpenTelemetry SDK

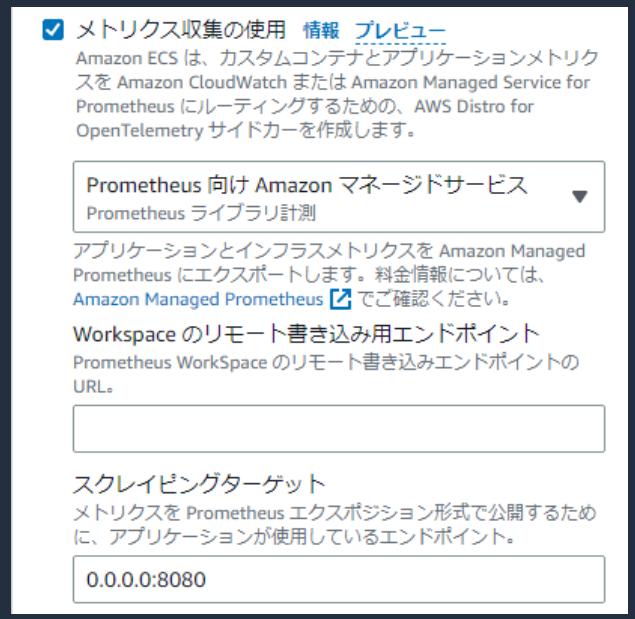


https://docs.aws.amazon.com/AmazonECS/latest/developerguide/application-metrics-prometheus.html
https://aws-otel.github.io/docs/setup/ecs

ECS のメトリクスを AMP に送信する (2/5)

ECS new console

- ECS new console では自動で AMP にメトリクスを送信する ADOT Collector のインジェストが可能
- タスク定義の作成画面から追加
  - カスタムメトリクスの収集方法を以下選択可能
    - Prometheus library
    - OpenTelemetry SDK



https://docs.aws.amazon.com/AmazonECS/latest/developerguide/create-task-definition.html#:~:text=Use%20metric%20collection

ECS のメトリクスを AMP に送信する (3/5)

ECS new console

- タスク定義に以下のように ADOT Collect のサイドカーコンテナ定義が追加される

Prometheus libraries instrumentation

```
...
{
    "name": "aws-otel-collector",
    "image": "public.ecr.aws/aws-observability/aws-otel-collector:latest",
    "cpu": 0,
    "portMappings": [],
    "essential": true,
    "command": [
        "--config=/etc/ecs/ecs-amp-prometheus.yaml"
    ],
    "environment": [
        {
            "name": "AWS_PROMETHEUS_SCRAPING_ENDPOINT",
            "value": "0.0.0.0:8080"
        },
        {
            "name": "AWS_PROMETHEUS_ENDPOINT",
            "value": "workspace-url"
        }
    ],
...

}
...
```

OpenTelemetry instrumentation

```
...
{
    "name": "aws-otel-collector",
    "image": "public.ecr.aws/aws-observability/aws-otel-collector:latest",
    "cpu": 0,
    "portMappings": [],
    "essential": true,
    "command": [
        "--config=/etc/ecs/ecs-amp.yaml"
    ],
    "environment": [
        {
            "name": "AWS_PROMETHEUS_ENDPOINT",
            "value": "worksapce-url"
        }
    ],
...
}
...
```

https://github.com/aws-observability/aws-otel-collector/tree/main/config/ecs

ECS のメトリクスを AMP に送信する (4/5)

ECS new console

- それぞれの設定で、利用される ADOT Collector の Receiver 設定が異なる
  - OTLP receiver の場合は、特定ポートに対してメトリクスを SDK から投げる仕組み（Push型）
    - ENDPOINTは OTEL_EXPORTER_OTLP_ENDPOINT 環境変数で指定する
    - https://opentelemetry.io/docs/concepts/sdk-configuration/otlp-exporter-configuration/
  - Prometheus receiver の場合は、指定したスクレイプターゲットで SDK がメトリクスを公開する設計（Pull型）

ecs-amp-prometheus.yaml

```
...
receivers:
  awsecscontainermetrics:
  prometheus:
    config:
      global:
        scrape_interval: 20s
        scrape_timeout: 10s
      scrape_configs:
        - job_name: "otel-collector"
          static_configs:
            - targets: [$AWS_PROMETHEUS_SCRAPING_ENDPOINT]
...
```

ecs-amp.yaml

```
...
receivers:
  otlp:
    protocols:
      grpc:
        endpoint: 0.0.0.0:4317
      http:
        endpoint: 0.0.0.0:4318
  awsecscontainermetrics:
...
```

https://github.com/aws-observability/aws-otel-collector/tree/main/config/ecs

ECS のメトリクスを AMP に送信する (5/5)
独自の ADOT Collector 設定の導入

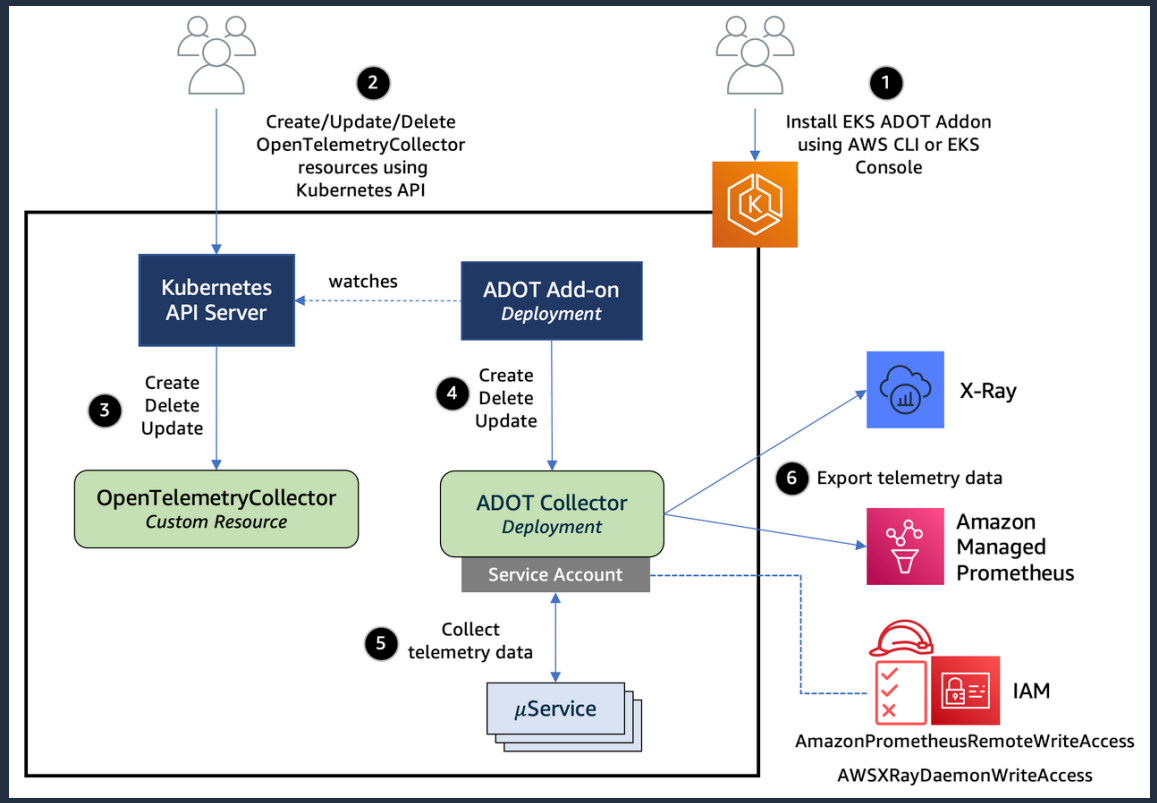
- 独自の ADOT Collector の設定を入れたい場合には SSM Parameter Store から ECS に読み込ませることが可能
  1. ECS の環境変数 AOT_CONFIG_CONTENT としてパラメータ名を事前に定義
  2. SSM Parameter Store にて、1 で指定したパラメータ名にて、ADOT Collector 設定ファイルを定義
     - type: String
     - Data type: Text

https://aws-otel.github.io/docs/setup/ecs/config-through-ssm

EKS のメトリクスを AMP に送信する (1/5)

- EKS では ADOT Operator の EKS アドオンを提供
- これを利用することで、ADOT Collector 経由での AMP へのメトリクス送信が可能

  - ADOT Operator では ADOT Collector の設定やデプロイ方法をリソース定義として記述可能
    - AMP 送信時の設定例
  - EKS アドオンを利用することで、ADOT Collector のバージョン管理を容易にできる

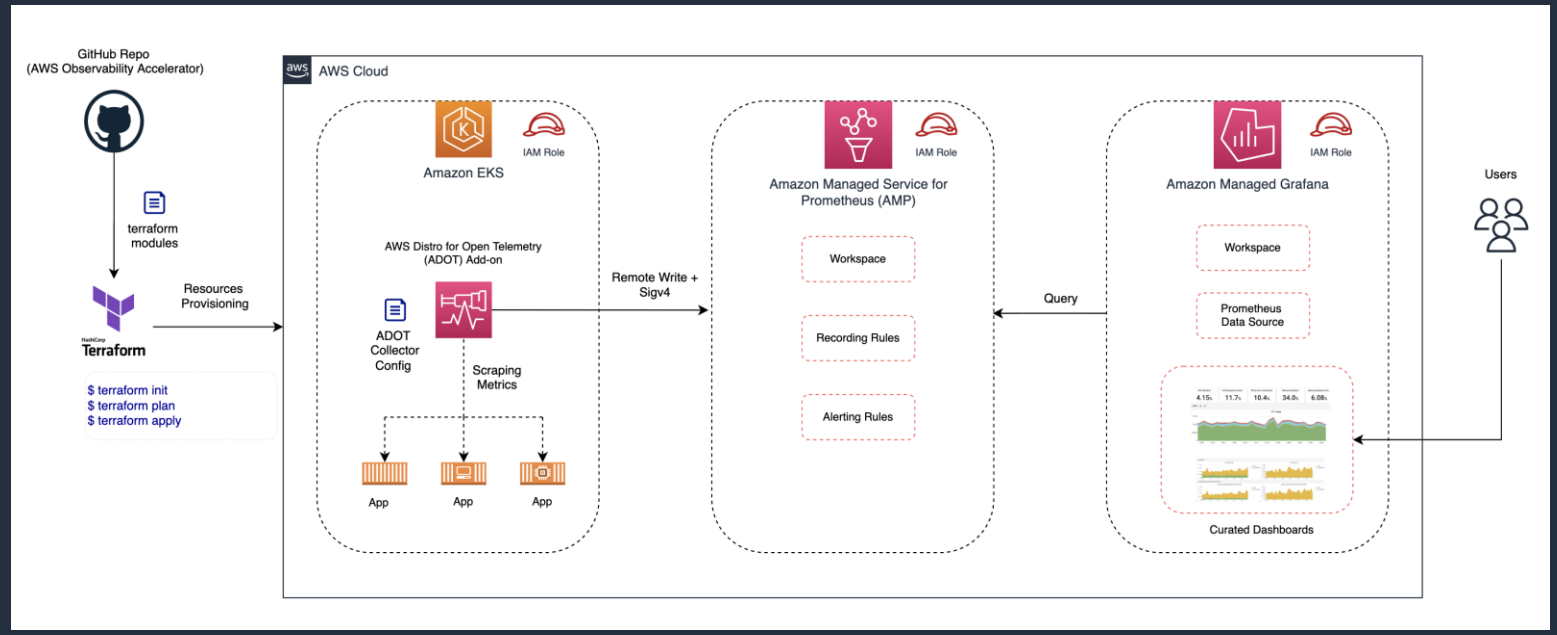


https://docs.aws.amazon.com/eks/latest/userguide/opentelemetry.html
https://aws.amazon.com/jp/blogs/news/metrics-and-traces-collection-using-amazon-eks-add-ons-for-aws-distro-for-opentelemetry/
https://github.com/aws-observability/aws-otel-community/tree/master/sample-configs/operator

EKS のメトリクスを AMP に送信する (2/5)

EKS とのセットアップを助ける AWS Observability Accelerator

- AWS Observability Accelerator により Terraform モジュールを利用して、既存の EKS クラスタに対して、以下のオブザーバビリティツールを迅速に開始可能
- 事前定義済みの設定/ダッシュボードが提供される
- デプロイされるサービス
  - AMP
  - ADOT
    - EKS アドオン or helm
  - Amazon Managed Grafana (AMG)



https://docs.aws.amazon.com/prometheus/latest/userguide/obs_accelerator.html
https://github.com/aws-observability/terraform-aws-observability-accelerator
https://aws-observability.github.io/terraform-aws-observability-accelerator/
https://aws.amazon.com/jp/blogs/news/announcing-aws-observability-accelerator-to-configure-comprehensive-observability-for-amazon-eks-jp/

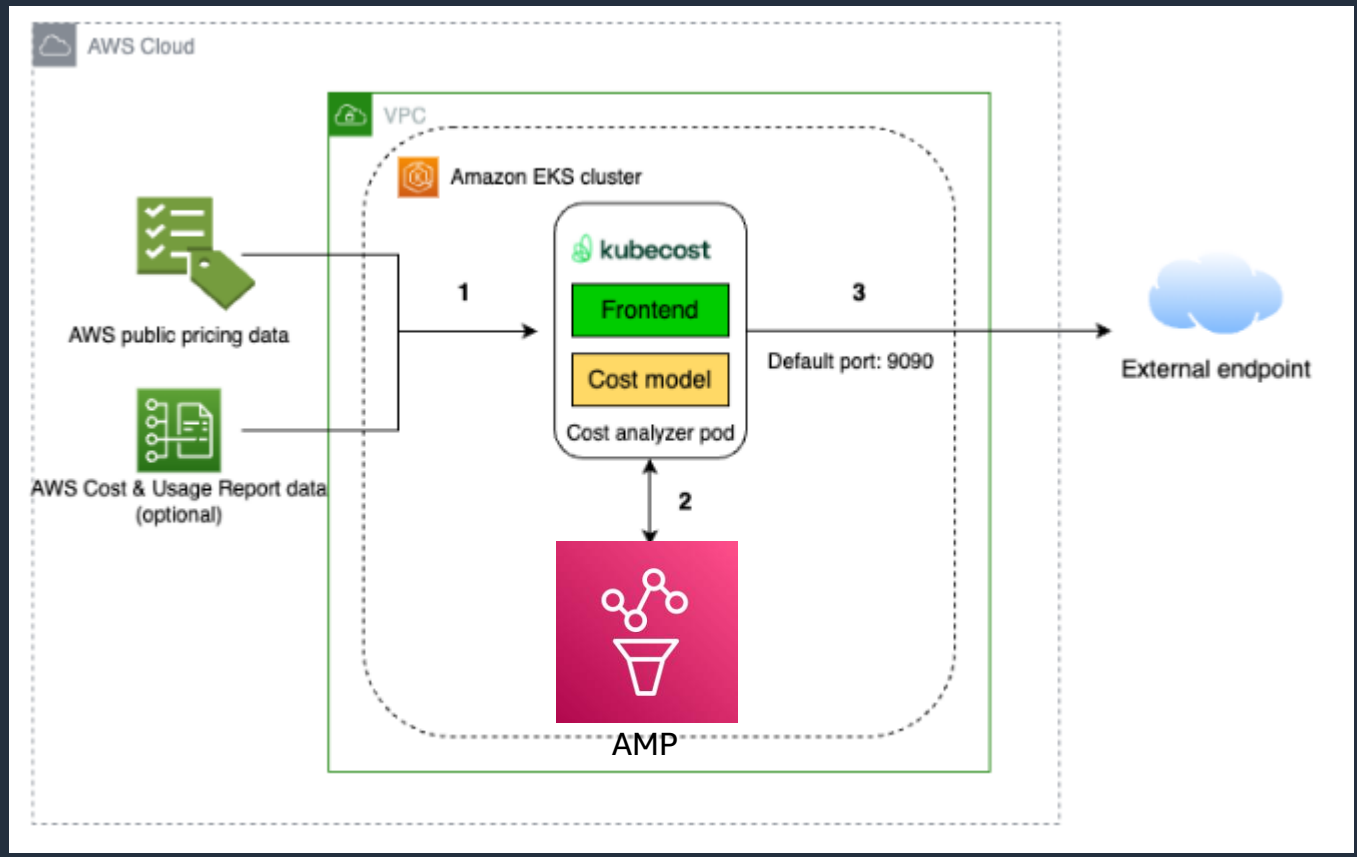
EKS のメトリクスを AMP に送信する (3/5)

AMP を利用した Kubecost による EKS コストモニタリング

- AMP は Kubecost と連携して、EKS のコストモニタリングが可能
- 手順
  - Kubecost のインストール (helm)
  - IRSA の作成
    - kubecost-cost-analyzer
    - kubecost-prometheus-server
  - AMP を利用するように Kubecost を設定 (helm パラメータの更新)
  - kubecost-prometheus-server Deployment の rollout

```
global:
  amp:
    enabled: true
    prometheusServerEndpoint:
http://localhost:8005/workspaces/$<AMP_WORKSPACE_ID>
    remoteWriteService: https://aps-workspaces.us-west-
2.amazonaws.com/workspaces/$<AMP_WORKSPACE_ID>/api/v1/remote_write
    sigv4:
      region: us-west-2
...
```

helm パラメータ



https://docs.aws.amazon.com/prometheus/latest/userguide/integrating-kubecost.html

EKS のメトリクスを AMP に送信する (4/5)

CloudWatch Container Insights との使い分け

- Prometheus が有用なケース
  - 監視したいメトリクスが Container Insights ではカバーできないケース
    - 例
      - コントロールプレーンのメトリクス
      - より詳細な cluster メトリクス（細かい Pod のステータス状況、Deployment や DaemonSet の数）
      - より詳細な node レベルのメトリクス（pod 数、ディスク系）
      - より詳細な pod（コンテナのステータス、CPUスロットル）
      - MW（Nginx、JMX ,etc）、アプリケーションからの計装（後述）やメッシュのメトリクス
  - メトリクス量が多く Container Insights の料金が高騰するケース
    - Container Insights は CloudWatch Logs で取り込んだ後に EMF にてメトリクス化しているため、取り込み料金がかかる
- Prometheusの魅力
  - Kubernetes との親和性
    - kubelet が、コンテナリソース情報を Prometheus メトリクスとして公開
    - サービスディスカバリの連携
  - エクスポーターが豊富で、監視したいメトリクスに合わせて選べる（E.g. kube-state-metrics, node-exporter）

EKS のメトリクスを AMP に送信する (5/5)

CloudWatch Container Insights のメトリクスカバー範囲

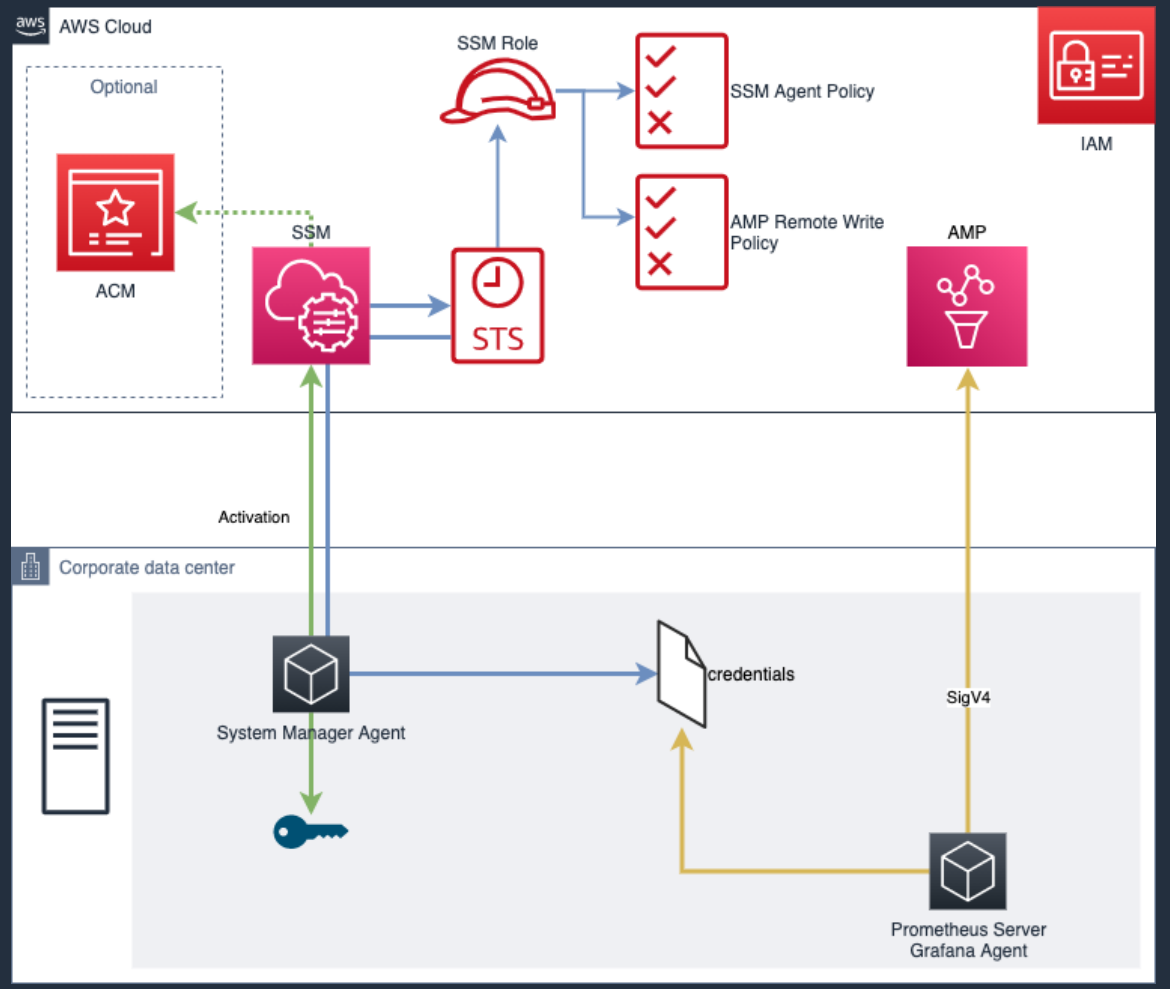
	監視対象	メトリクス例	収集方式
サービス監視 [RED]	各サービス LB, APIGW, Proxy	Throughput (rps) Error% Response Time	☐ LB, APIGW, Web server metrics (E.g. ELB, Nginx, Apache) ☐ Proxy on Service Mesh • Make it a service mesh and collect metrics for each Service with Envoy metrics ☐ OpenTelemetry or Prometheus SDK
リソース監視 [USE]	Node * Only for on EC2	[OS metrics] CPU/Memory/NW/Disk usage	☐ EC2 metrics ☐ **Container Insights** ☐ Prometheus Node Exporter
	Pod / Container	[OS metrics] CPU/Memory/NW/Disk usage	☐ **Container Insights** ☐ cAdvisor
	MW / AP	[Runtime metrics] E.g. Number of threads GC [middleware specific metrics] E.g. Concurrency	☐ Prometheus exporter ☐ OpenTelemetry or Prometheus SDK

Container Insights のカバー範囲

Tips

オンプレミスのメトリクスの AMP への収集

- AMP への remote write には IAM ロールが必要
- SSMエージェント経由でSTSによる一時クレデンシャルを使用したアクセスが可能
  - ※アクセスキーは漏洩リスクが高いため非推奨

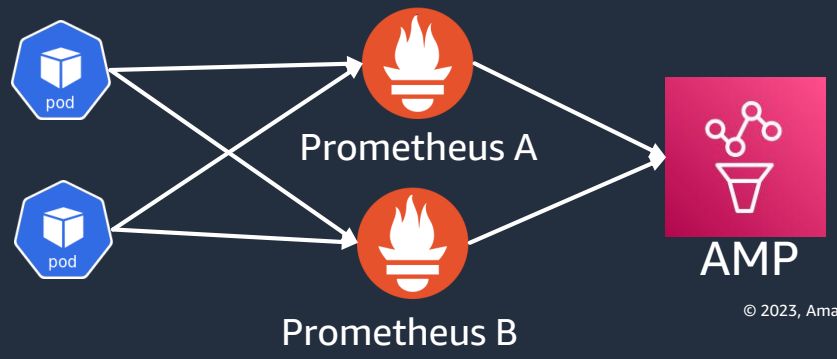


https://aws.amazon.com/jp/blogs/news/collect-on-premises-metrics-using-amazon-managed-service-for-prometheus/

冗長構成を取る Prometheus からの AMP にメトリクスを送信する

- 冗長構成を取る Prometheus からメトリクスを取り込む場合、メトリクスが重複する（タイムスタンプには誤差がある）のが課題
- メトリクスを重複して取り込まない工夫として、AMP は以下のような仕組みを提供
  - 冗長構成のクラスタの内1台をリーダーとして認識し、リーダーが停止するまではリーダーからのみメトリクスを取り込む
  - リーダーから取り込みが30秒間停止した場合、自動的に別のレプリカをリーダーとし、メトリクスを取り込みを再開する

- 必要な設定
  - 収集元の Prometheus クラスタに、ラベル付与（cluster, __replica__）
  - Prometheus Operator の場合は、 replicaExternalLabelName, externalLabels にて対応



https://docs.aws.amazon.com/prometheus/latest/userguide/AMP-ingest-dedupe.html

クオータと料金

クオータ

一部抜粋

クオータ	デフォルト値	調整可能か
アクティブなメトリクス （過去2時間以内に取り込んだメトリクス） /ワークスペース毎、メトリクス名毎	200万件	Yes ※後述の通り2億個まで緩和可能
取り込みレート	14万件/秒	Yes
メトリクスあたりのラベル数	70個	Yes
クエリあたりの上限サイズ / 日	5GB	No
クエリあたりのチャンク数	200万	No
クエリ当たりのサンプル数	500万	No
クエリの時間範囲	32日	No
メトリクス保持期間	150日	Yes
リージョンあたりの Workspace 数	25個	Yes
アクティブなアラート	1000	Yes

https://docs.aws.amazon.com/general/latest/gr/prometheus-service.html#prometheus-quotas

2022/11 Update:
AMP でワークスペースあたり 2億個 のアクティブメトリクスをサポート

- コンテナは短命で自動スケーリングが可能なため、コンテナ環境を監視するために取り込んで分析する必要のあるメトリクスは、すぐに数千万個に達してしまう
- 今回のリリースで、上限引き上げの申請後、1つのワークスペースに最大2億個のアクティブメトリクスを送信することができ、アカウントごとに多数のワークスペースを作成できるため、数十億のPrometheusメトリクスの保存と分析が可能に

Resource	Default quota	Possible error message
Active series (metrics that have reported data in the past 2 hours) per workspace	2,000,000. You can request a quota increase.	per-user series limit of 2000000 exceeded, please contact administrator to raise it
Active series per metric name	2,000,000. You can request a quota increase.	per-metric series limit of 2000000 exceeded, please contact administrator to raise it

Wat's New : https://aws.amazon.com/about-aws/whats-new/2022/11/amazon-managed-service-prometheus-200m-active-metrics-workspace/
Blog : https://aws.amazon.com/jp/blogs/mt/amazon-managed-service-for-prometheus-adds-support-for-200m-active-metrics/

料金 (2023年1月25日現在 : 東京リージョン)

メトリクスの収集(サンプル)数・ストレージ・クエリ実行時間に対する従量課金

メトリクスの収集(サンプル)数

最初の 20 億件	$0.90 / 1000万件
次の 2500 億件	$0.35 / 1000万件
2520 億件以上	$0.16 / 1000万件

ストレージ使用量・クエリ量

メトリクスのストレージ	$0.03 / GB
クエリにより処理されるサンプル数 (Query Sample Processed : QSP)	$0.1 / 100億 QSP

無料利用枠

メトリクスの収集 (サンプル) 数	4000 万件
クエリにより処理されるサンプル数 (Query Sample Processed : QSP)	200 億 QSP
ストレージ (in GB)	10 GB

料金ページ: https://aws.amazon.com/jp/prometheus/pricing/

参考 : AWS における Observability の選択肢

AMP の学習リソース

AMP の学習リソース

AMP ユーザーガイド

https://docs.aws.amazon.com/prometheus/latest/userguide/what-is-Amazon-Managed-Service-Prometheus.html

One Observability ワークショップ

https://catalog.us-east-1.prod.workshops.aws/workshops/31676d37-bbe9-4992-9cd1-ceae13c5116c/ja-JP/intro

AWS Observability Best Practice

https://aws-observability.github.io/observability-best-practices/

まとめ

まとめ

01

Amazon Managed Service for Promtheus (AMP) は、フルマネージドな Prometheus 環境

Amazon EKS, Amazon ECS などのコンテナ環境に対して大規模なメトリクス収集・モニタリングの実現を簡単に

02

セキュアでスケーラブル

セキュリティパッチの自動適用, AWS サービスとの統合によるマルチアカウント・マルチリージョン環境への容易な対応をはじめとした、エンタープライズでの利用に対応したレベルのセキュリティ

03

他の AWS サービスと容易に連携可能

Amazon SNS, AWS IAM, Amazon Managed Grafana など, AWS サービスとシームレスに連携可能な Prometheus 環境

本資料に関するお問い合わせ・ご感想

技術的な内容に関しましては、有料のAWSサポート窓口へ
お問い合わせください

https://aws.amazon.com/jp/premiumsupport/

料金面でのお問い合わせに関しましては、カスタマーサポート窓口へ
お問い合わせください（マネジメントコンソールへのログインが必要です）

https://console.aws.amazon.com/support/home#/case/create?issueType=customer-service

具体的な案件に対する構成相談は、後述する個別相談会をご活用ください

ご感想はTwitterへ！ハッシュタグは以下をご利用ください
#awsblackbelt

その他コンテンツのご紹介

ウェビナーなど、AWSのイベントスケジュールをご参照いただけます

https://aws.amazon.com/jp/events/

ハンズオンコンテンツ

https://aws.amazon.com/jp/aws-jp-introduction/aws-jp-webinar-hands-on/

AWS 個別相談会

AWSのソリューションアーキテクトと直接会話いただけます

https://pages.awscloud.com/JAPAN-event-SP-Weekly-Sales-Consulting-Seminar-2021-reg-event.html

Thank you!