[AWS Black Belt Online Seminar]

AWS App Mesh Deep Dive

サービスカットシリーズ

Solutions Architect, Containers
Kyosuke Ochimizu
2020/10/14

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

過去資料
https://amzn.to/JPArchive

自己紹介

落水 恭介 (Ochimizu Kyosule)

- Specialist Solutions Architect, Containers
    - AWS のコンテナ関連サービスを担当

- 好きなサービス
    - Amazon Elastic Container Service (Amazon ECS)
    - Amazon Elastic Kubernetes Service (Amazon EKS)

aws

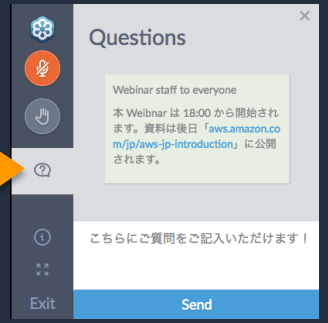
AWS Black Belt Online Seminar とは

「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 10 月 14 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。

- 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.

本セミナーの概要

- 本セミナーで学習できること
  - 異なるプラットフォームで稼働するサービスの
    サービス間通信に AWS App Mesh を導入するメリット
  - Amazon ECS および Amazon EKS における
    AWS App Mesh の具体的な利用方法
- 対象者
  - AWS App Mesh の導入を検討しているアーキテクト、
    技術者の方

本日のアジェンダ

- サービスメッシュとは？
- AWS App Mesh の概要
- Example for AWS App Mesh
- AWS App Mesh の利用料金

本日のアジェンダ

- サービスメッシュとは？
- AWS App Mesh の概要
- Example for AWS App Mesh
- AWS App Mesh の利用料金

サービスメッシュとは

アプリケーションレベルの通信を、アプリケーション自身が制御するのではなく
インフラストラクチャーで制御できるようにする技術

アプリケーションが行う通信制御

- HTTP 通信のリトライやタイムアウト
- 通信のトレーシングやログ、メトリクスの取得
- TLS を使用した暗号化通信

サービスメッシュとは

アプリケーションレベルの通信制御を、サービスメッシュの基盤で行うので、
アプリケーションに組み込む必要がなくなる

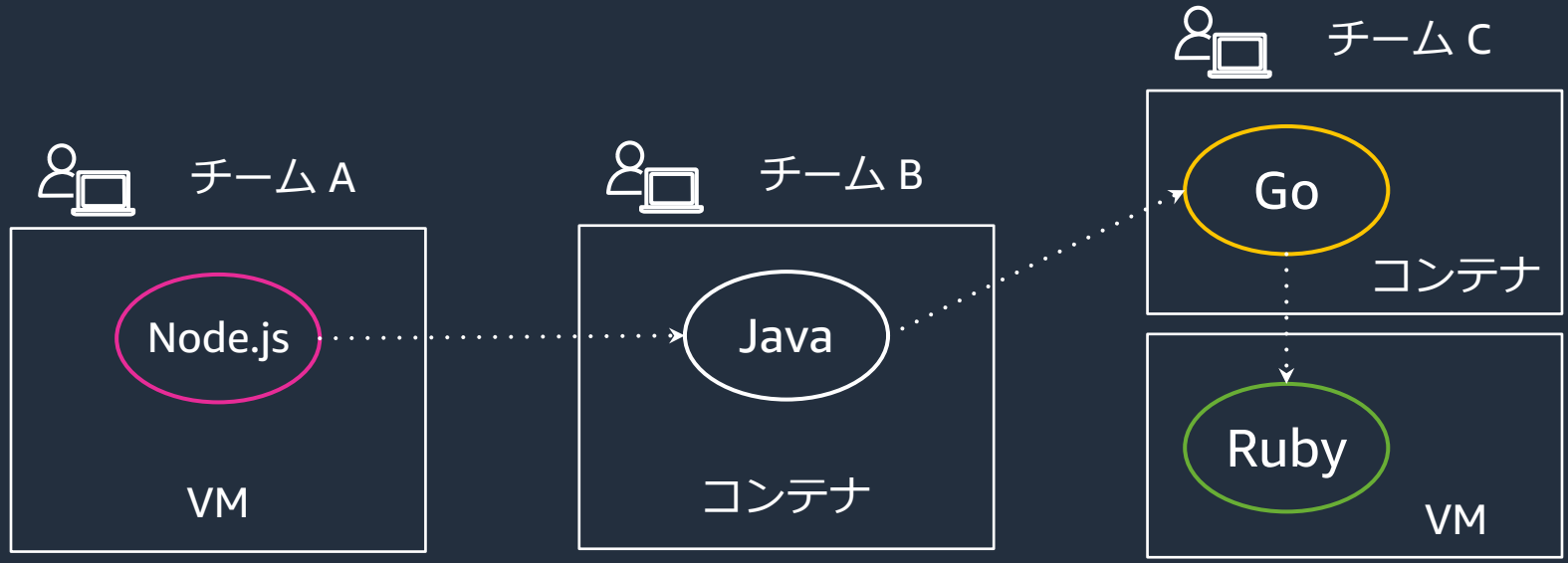
- HTTP 通信のリトライやタイムアウト
- 通信のトレーシングやログ、メトリクスの取得
- TLS を使用した暗号化通信

サービスメッシュが求められるようになった背景

- 現代のシステムは 複数の言語、アーキテクチャ、アプリケーション で構成
- クラウドにより多様な環境が簡単に作成できるように
- マイクロサービスアーキテクチャの採用

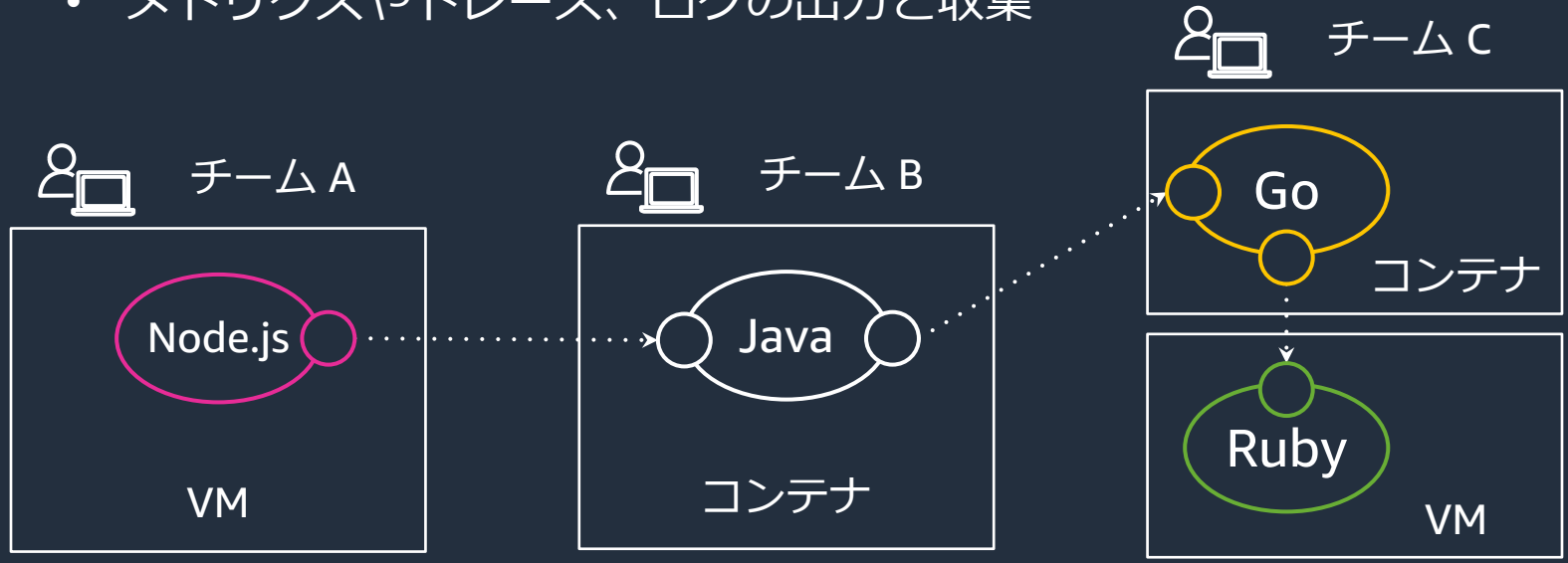
例: 多様なアプリケーションが通信を行うシステム

- チームごとに最適な技術を選択してアプリケーションを動かす
- プログラミング言語だけでなく、VM やコンテナなど様々なコンピューティングプラットフォームを選択できる

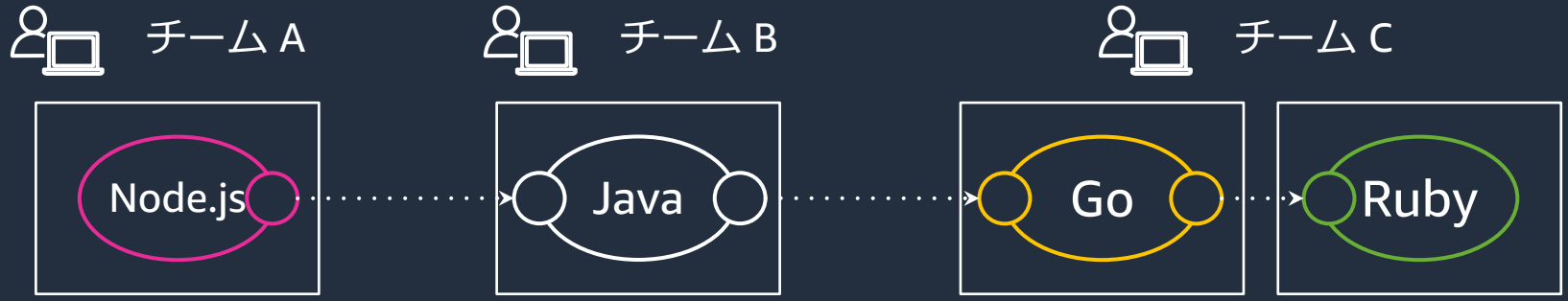


全てのアプリケーションに、同じような通信の仕組みが必要

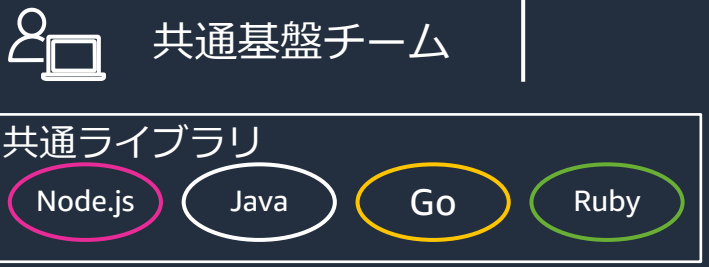
- 信頼性の確保
    - リトライやタイムアウト、暗号化通信の実装
- 可観測性の確保
    - メトリクスやトレース、ログの出力と収集



ライブラリによる通信制御の共通化

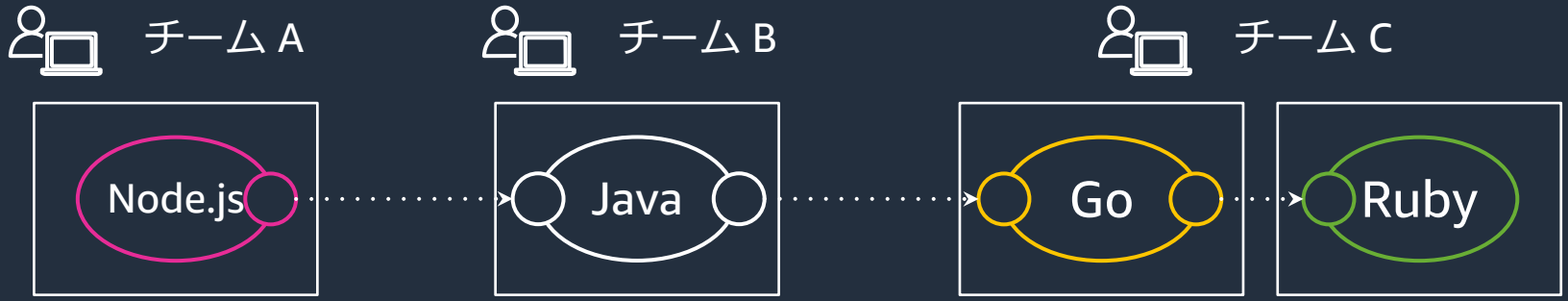


通信の仕組みを共通ライブラリとして実装、配布



- リトライやタイムアウト、暗号化通信の実装
- メトリクスやトレース、ログの出力と収集

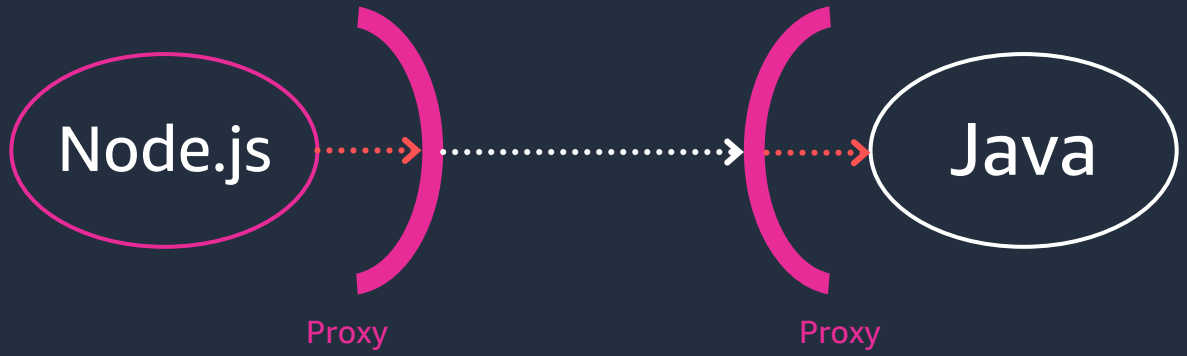
共通ライブラリの導入における課題



- 共通ライブラリを入れると依存関係が衝突する
- 言語ごとにライブラリを実装する必要があり負担が大きい

アプリケーション開発者・共通基盤チーム共に運用負荷が増大する

アプリケーションから通信処理を分離



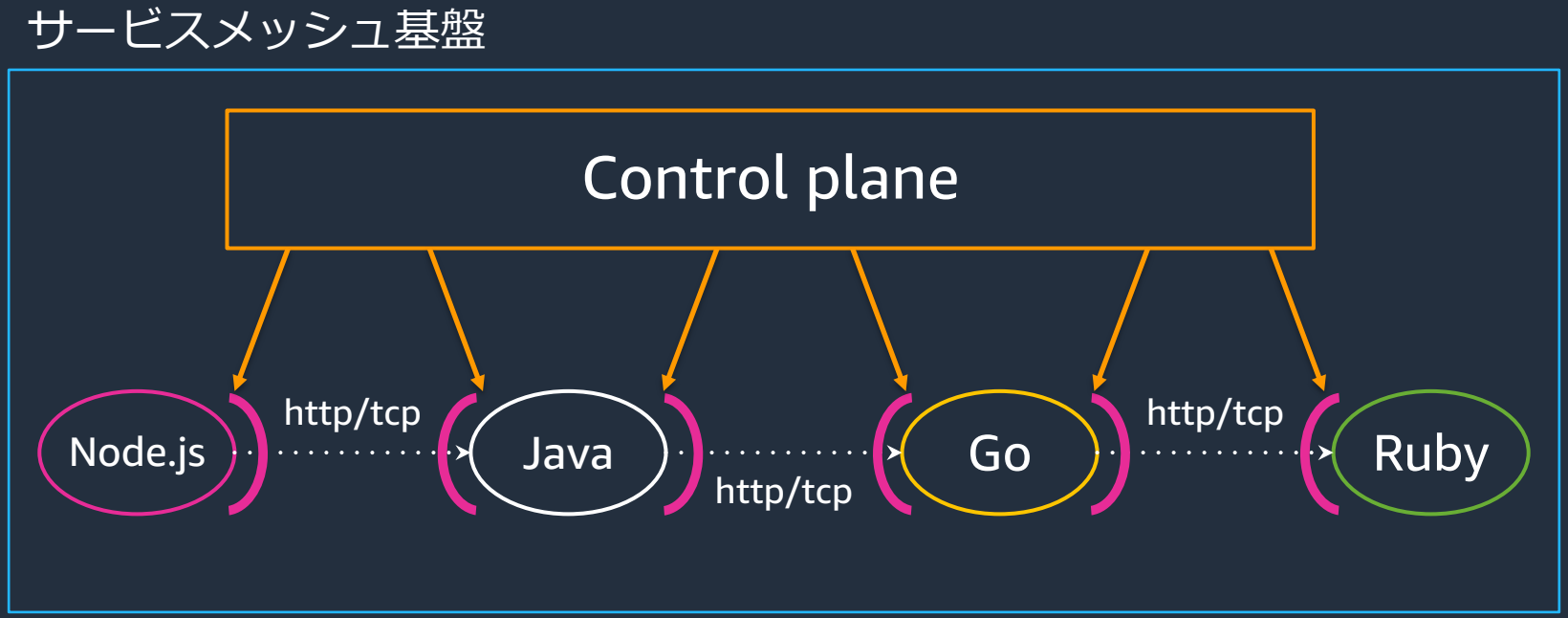
Proxy

アプリケーション間の通信をプロキシして、通信制御を行うプロセス

サービスメッシュの仕組み

プロキシをコントロールプレーンで管理

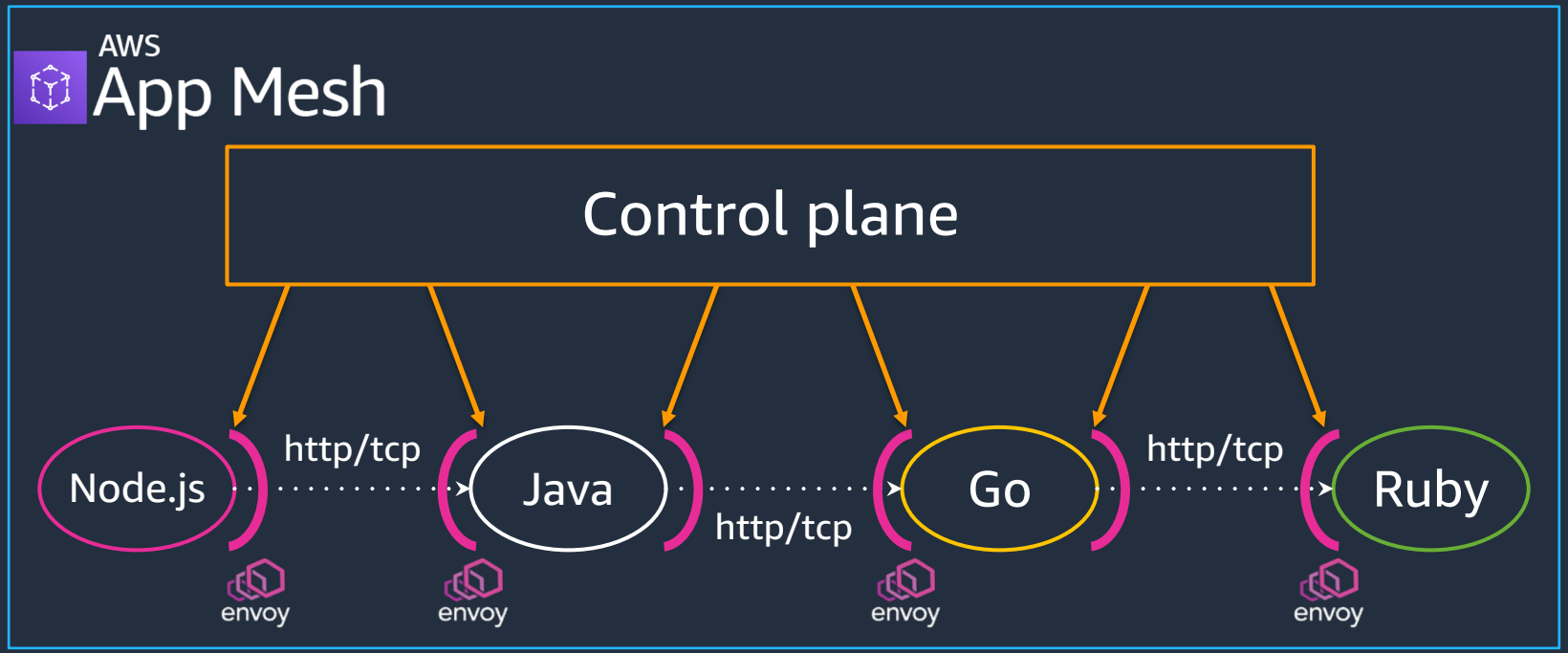
=> アプリケーションレベルの通信制御をインフラストラクチャ側で管理可能に



aws

AWS App Mesh とは

Envoy を管理するコントロールプレーンを提供するサービスメッシュ

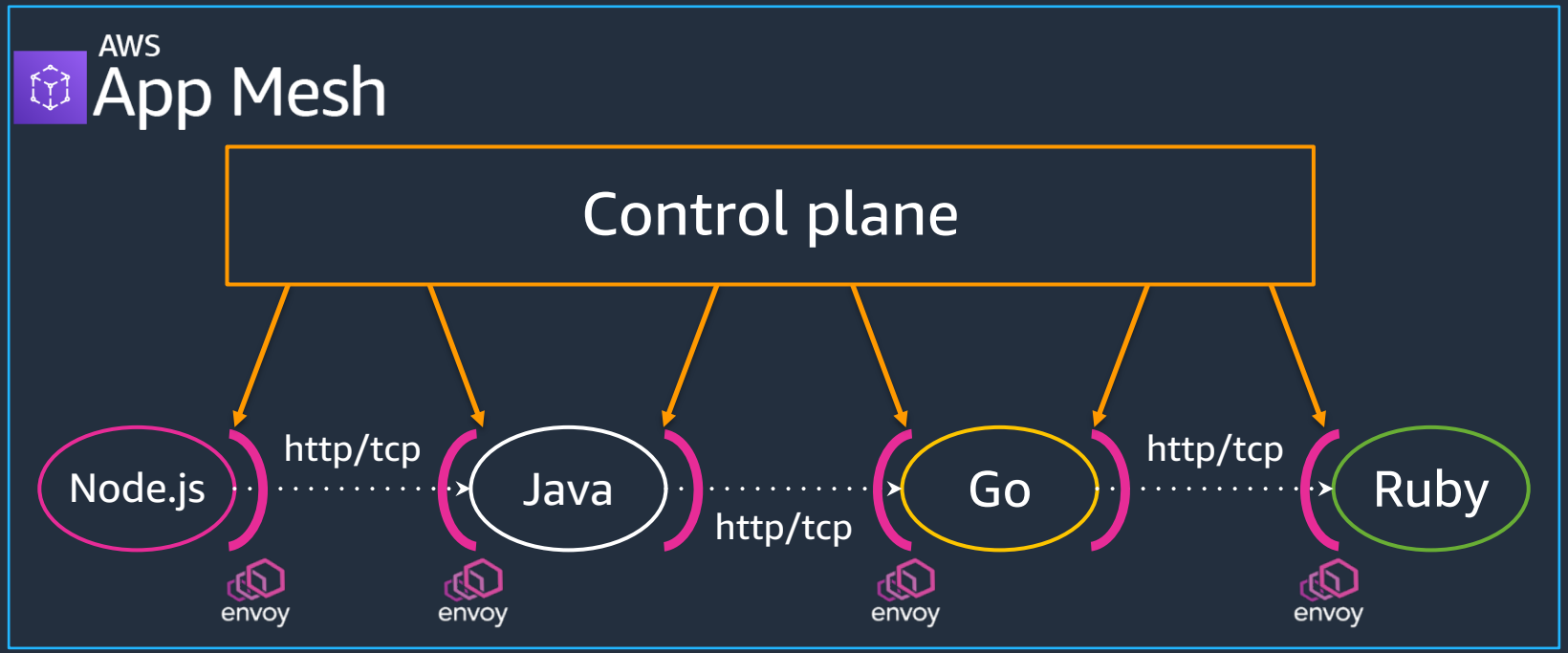


本日のアジェンダ

- サービスメッシュとは？
- AWS App Mesh の概要
- Example for AWS App Mesh
- AWS App Mesh の利用料金

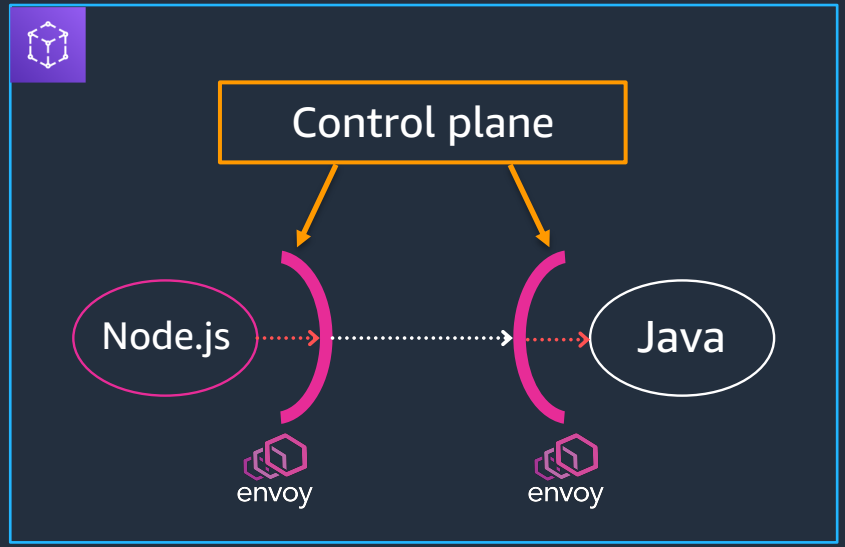
[再掲] AWS App Mesh とは

Envoy を管理するコントロールプレーンを提供するサービスメッシュ



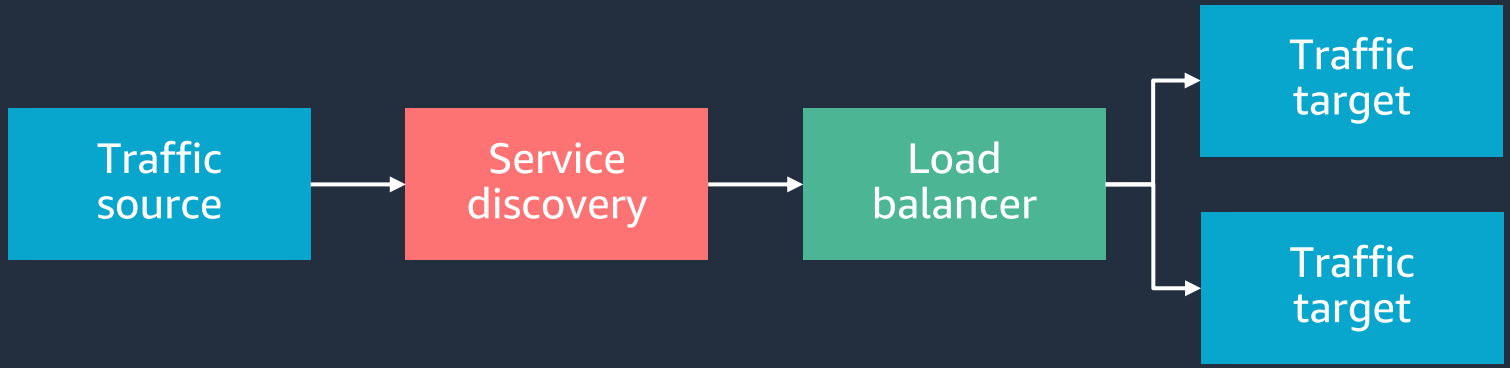
AWS App Mesh

サービスメッシュを管理するコントロールプレーンを提供する



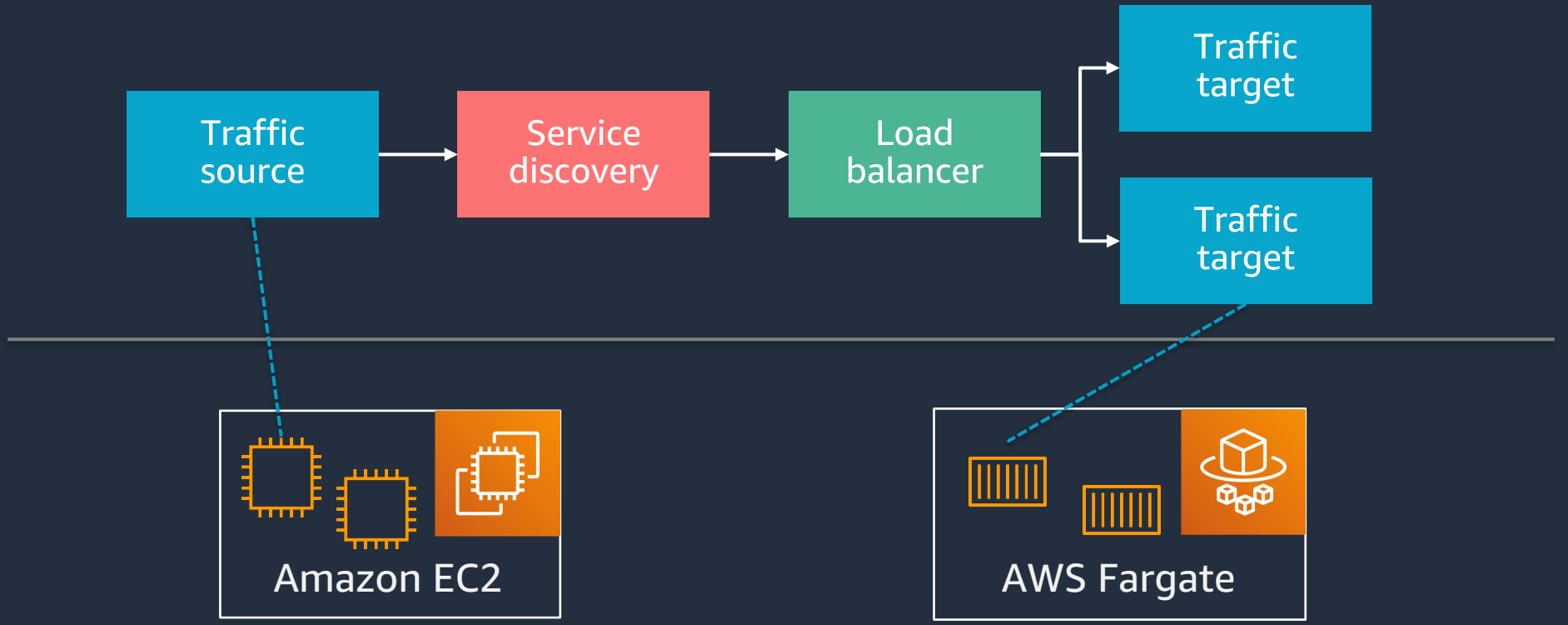
- アプリケーションレベルの通信をネットワークモデルとして定義
- ネットワークモデルを Envoy の設定に変換して配布

AWS App Mesh のネットワークモデル



1. 通信元のアプリケーションは、通信先をサービスディスカバリーで発見してリクエストを送信
2. リクエストがロードバランサーに到達
3. ロードバランサーが、複数の通信先にリクエストを振り分け

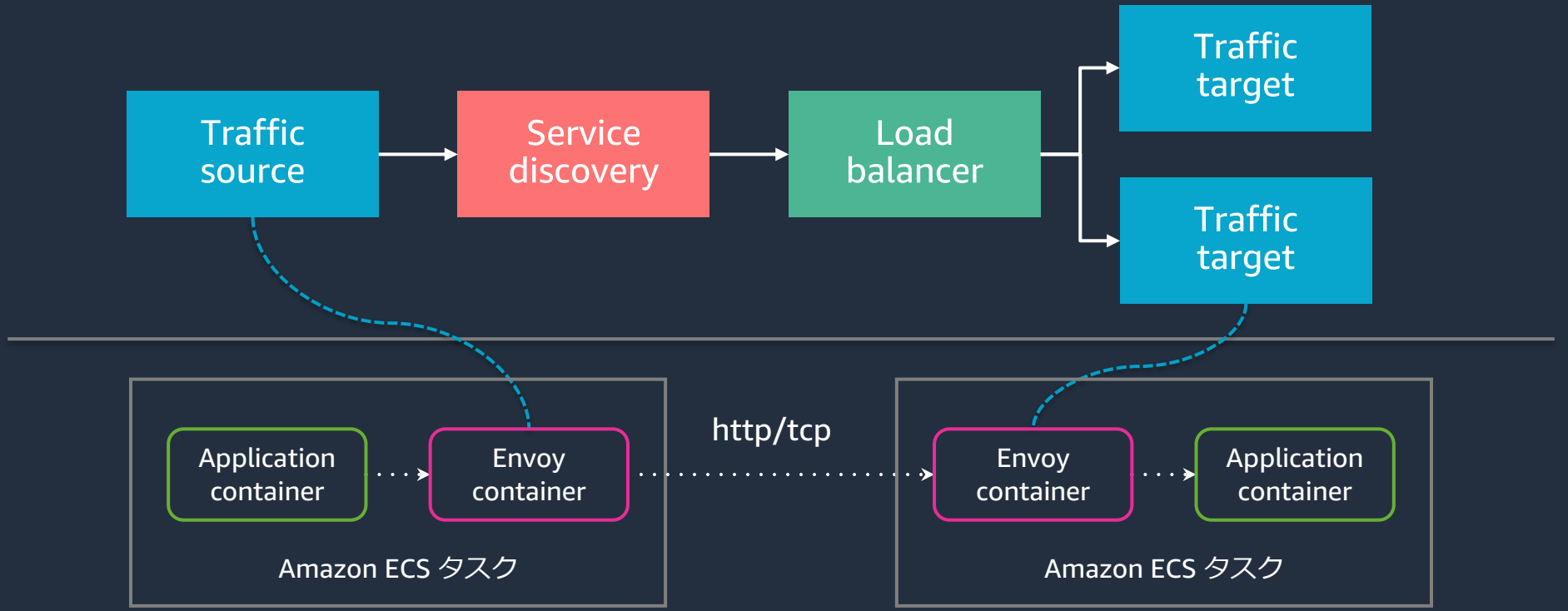
ネットワークモデルとアプリケーションの関係



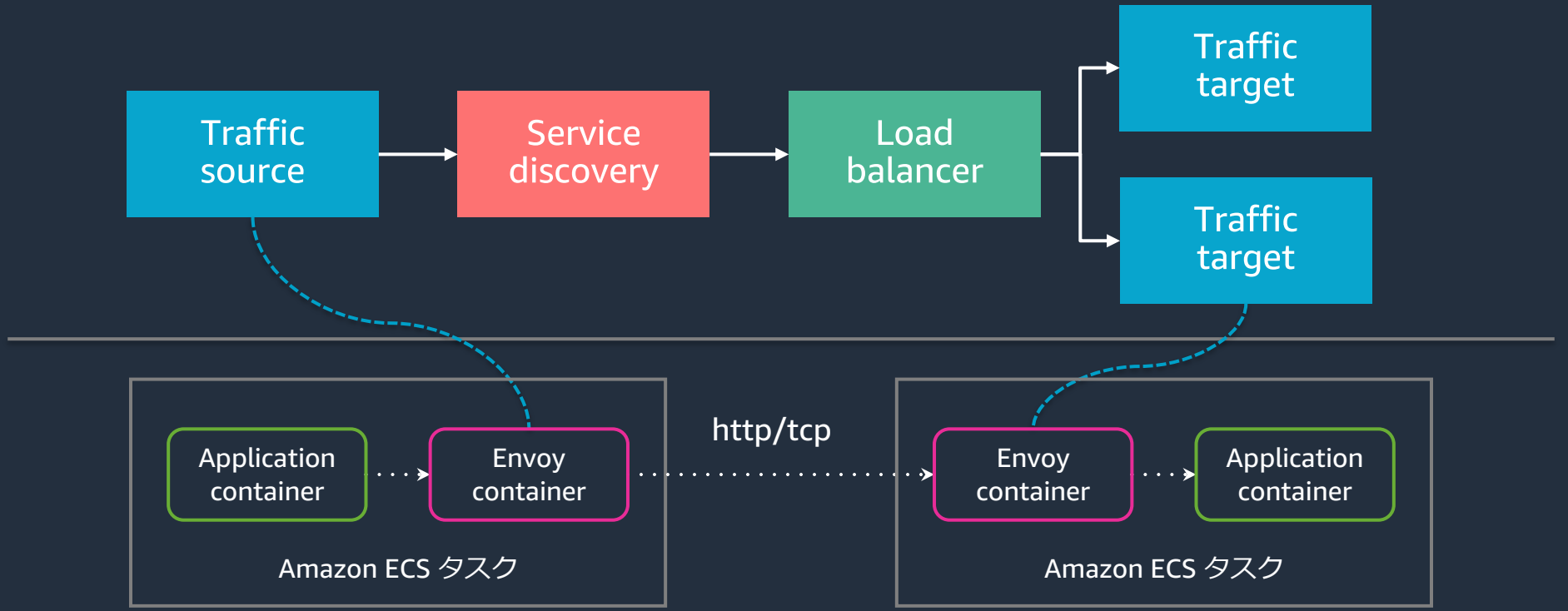
AWS App Mesh の動作イメージ

AWS App Mesh の動作イメージ: Envoy proxy の導入

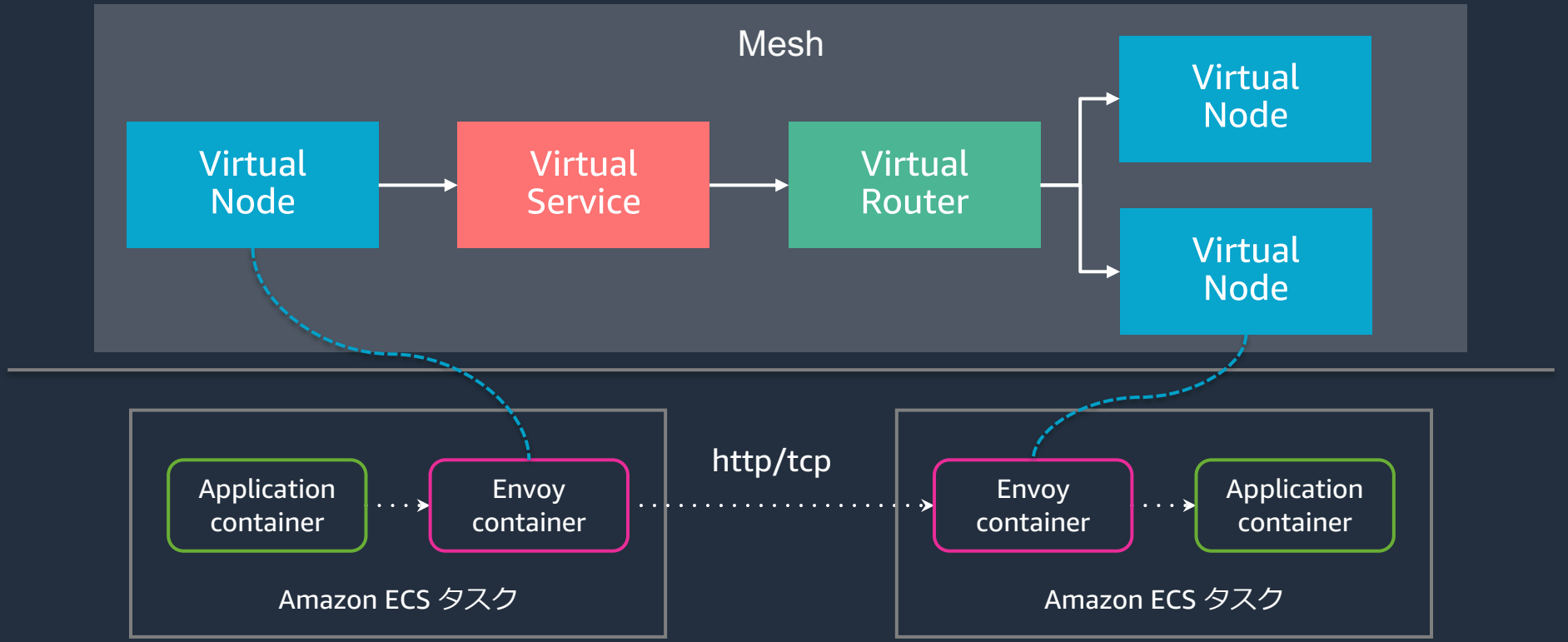
AWS App Mesh の動作イメージ: ネットワークモデルとの関係



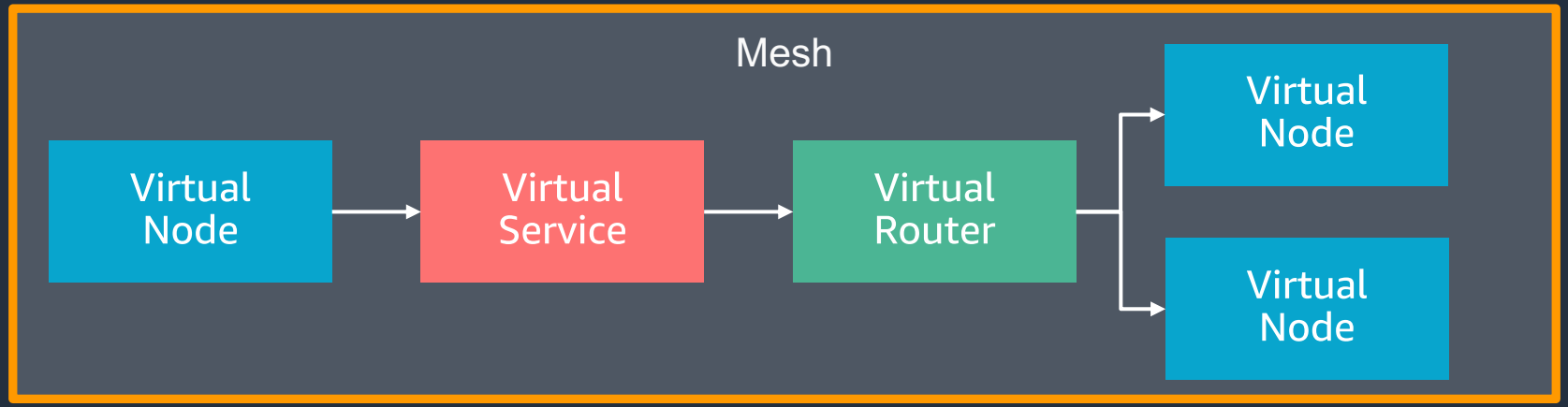
AWS App Mesh の動作イメージ: トップレベルの概念



AWS App Mesh の動作イメージ: トップレベルの概念



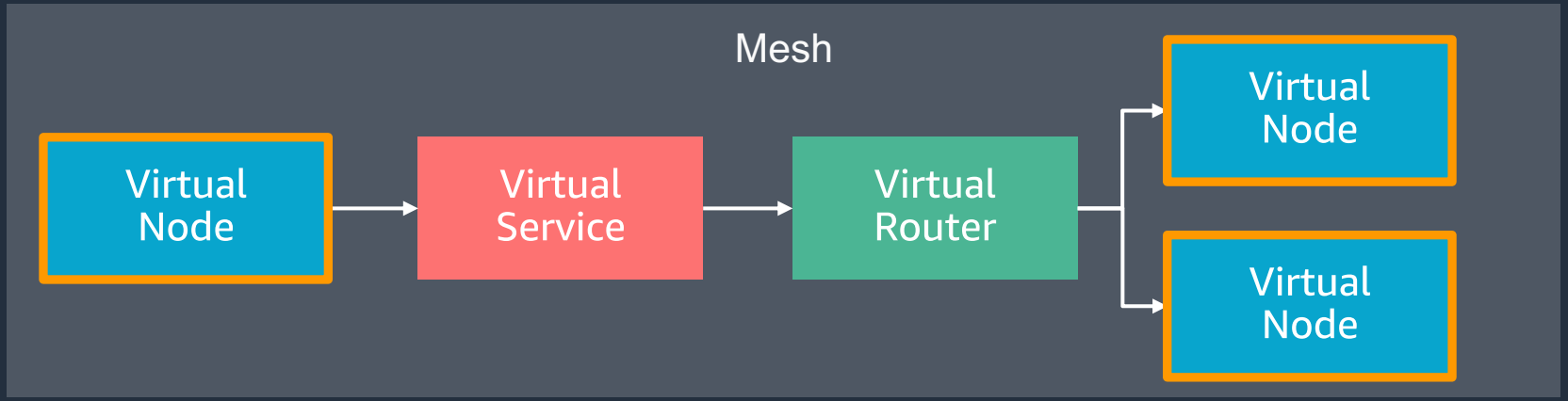
トップレベルの概念: Mesh



- サービス間の通信制御を行う論理的な境界
- Mesh の中に Virtual Node や Virtual Service などを組み合わせて
  ネットワークモデルを構築する

aws

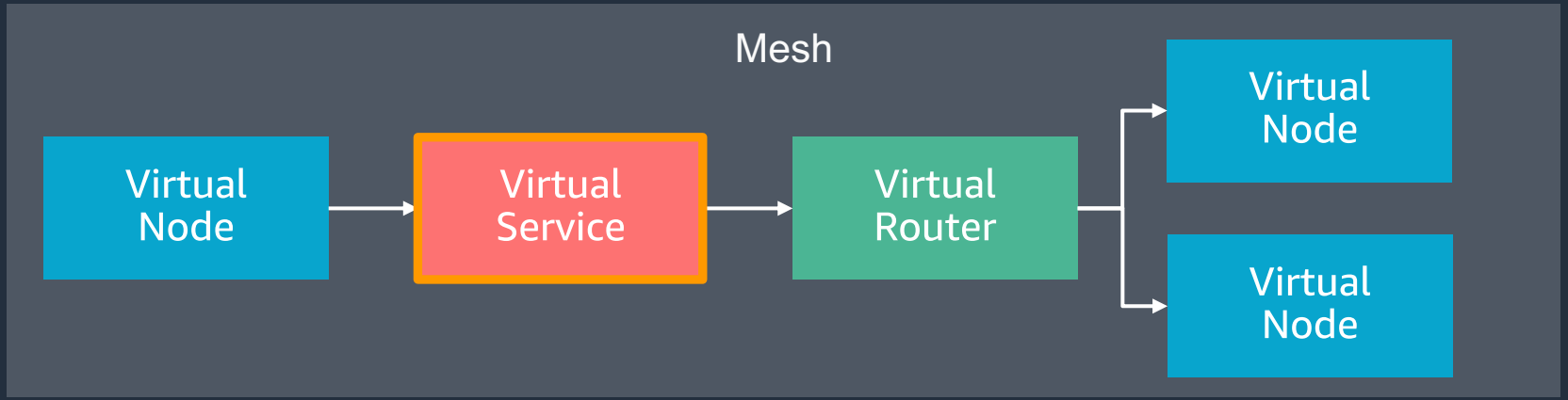
トップレベルの概念: Virtual Node



- 実際のアプリケーションへの論理的なポインタ
  - 例) ECS サービス、Kubernetes デプロイメント
- Envoy の実行時パラメータ (環境変数など) で Virtual Node 名を設定し、アプリケーション (Envoy) と Virtual Node を関連づける

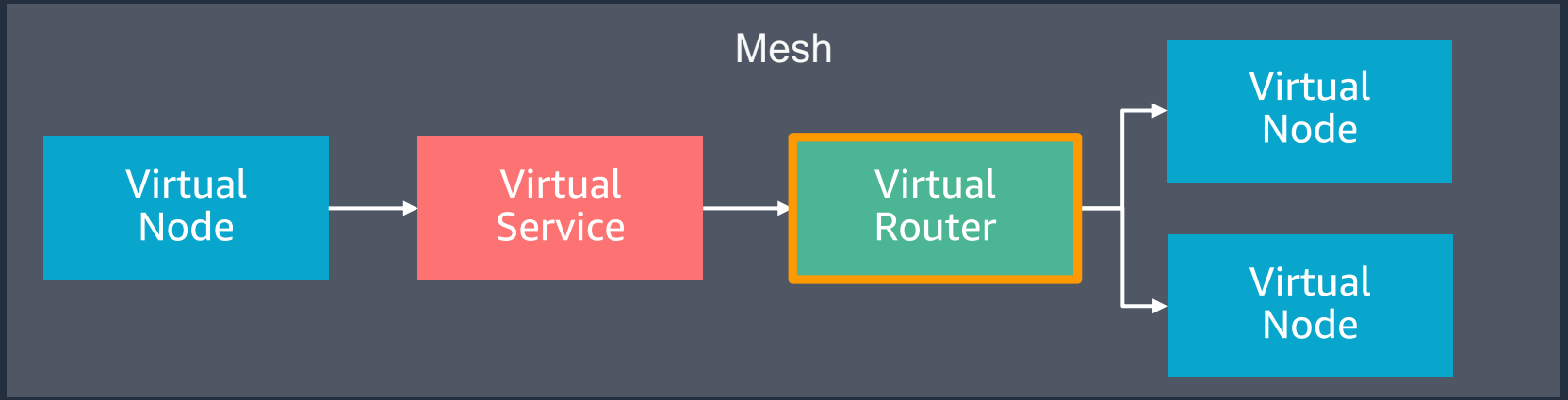
aws

トップレベルの概念: Virtual Service



- アプリケーションの通信先を表す
- リクエストは Virtual Router または Virtual Node にルーティングされ、実際のアプリケーションに到達する

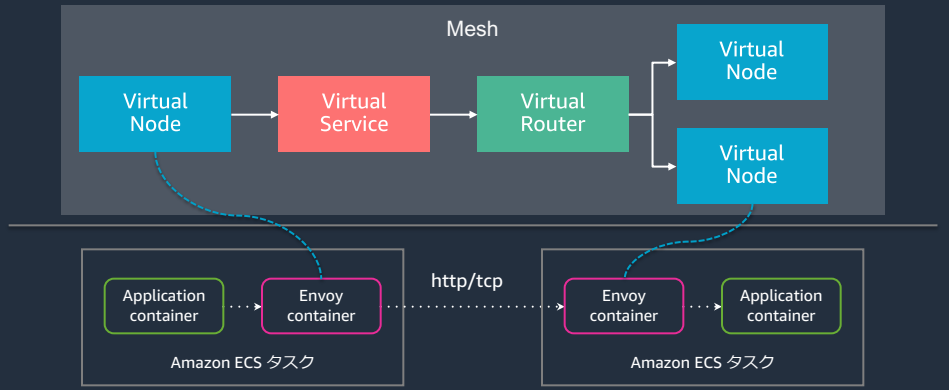
トップレベルの概念: Virtual Router



- リクエストのルーティングを管理するロードバランサー
- 単一または複数の Virtual Node にルーティングされる

トップレベルの概念まとめ

- Mesh
  - サービスメッシュの論理的な境界
- Virtual Node
  - アプリケーションのポインタ
- Virtual Service
  - アプリケーションの通信先
- Virtual Router
  - ルーティング管理



関連資料

※AWS App Mesh の基本的な機能についての情報が必要の方は、下記の AWS Black Belt Online Seminar の資料をご参照ください。

[AWS Black Belt Online Seminar]
AWS App Mesh

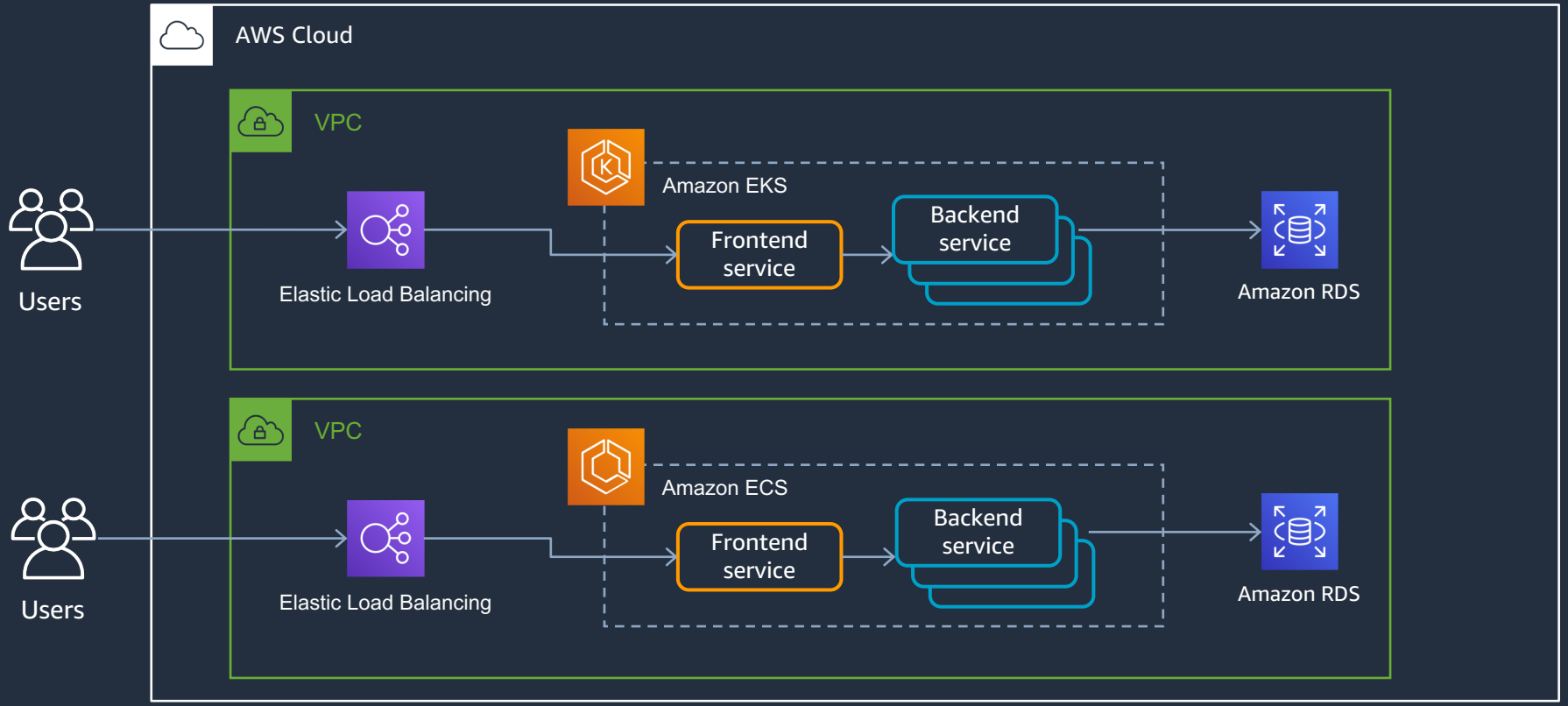
動画:
https://youtu.be/IC7sxVh7HNM
資料:
https://d1.awsstatic.com/webinars/jp/pdf/services/20200721_BlackBelt_AWS_App_Mesh.pdf

Speaker 林 政利

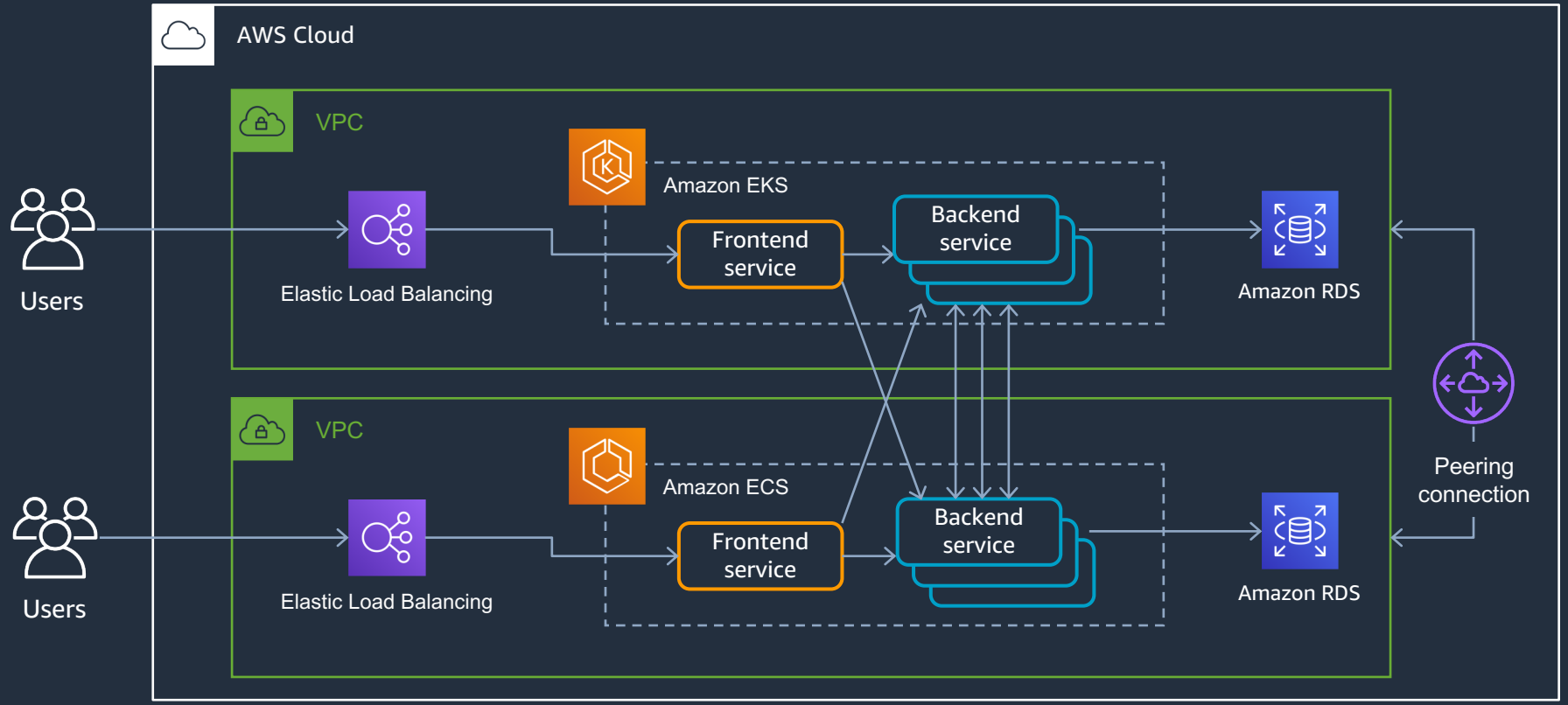
本日のアジェンダ

- サービスメッシュとは？
- AWS App Mesh の概要
- Example for AWS App Mesh
- AWS App Mesh の利用料金

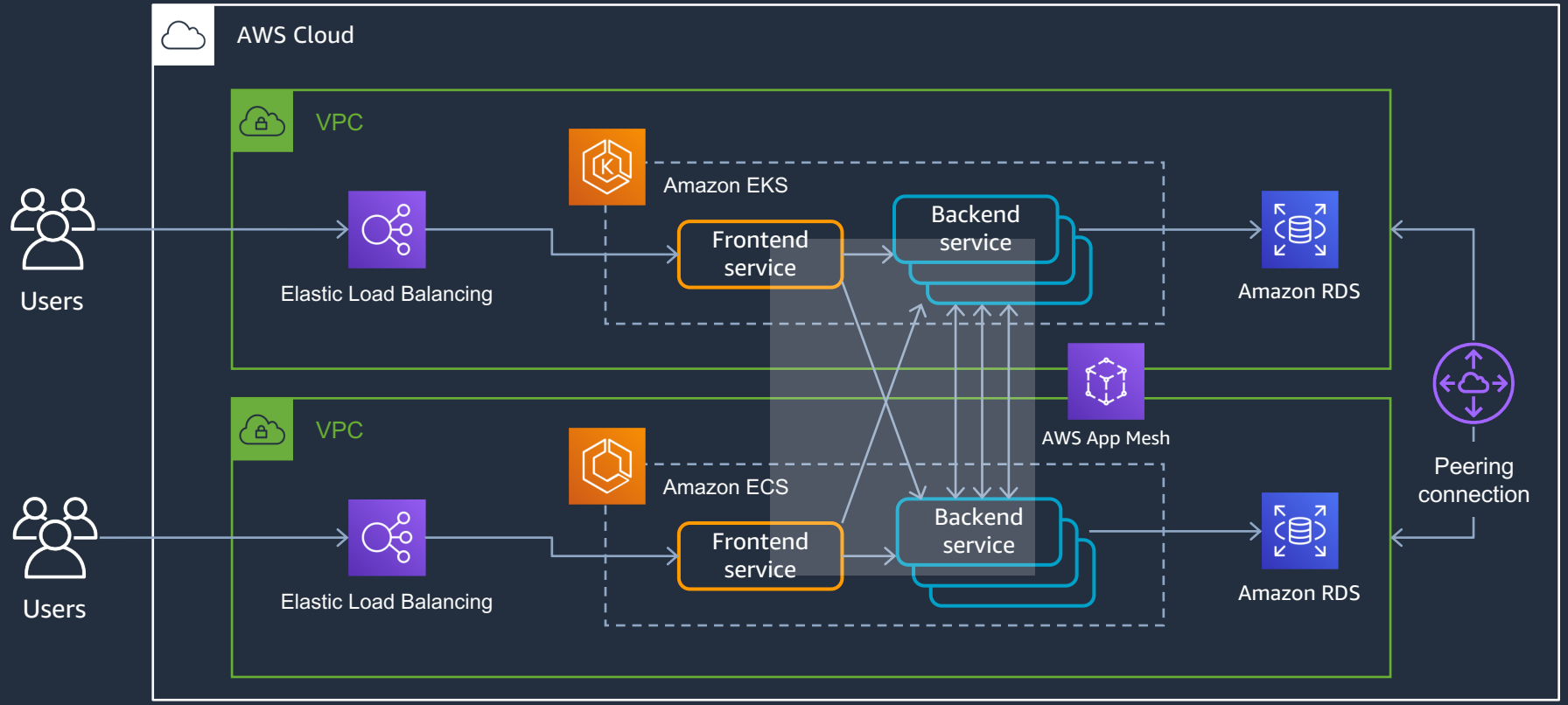
例: Amazon ECS と Amazon EKS で稼働するサービス間通信



例: Amazon ECS と Amazon EKS で稼働するサービス間通信



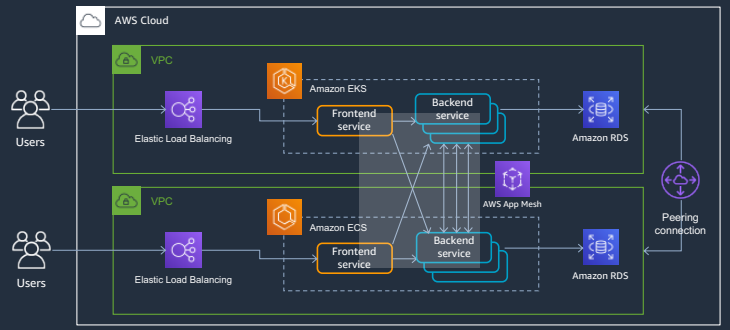
例: Amazon ECS と Amazon EKS で稼働するサービス間通信



例: Amazon ECS と Amazon EKS で稼働するサービス間通信

想定シナリオ

1. Amazon ECS と Amazon EKS それぞれで
   サービスが稼働している
2. ビジネス上の理由で、Amazon ECS で稼働する
   サービスと Amazon EKS で稼働するサービス
   の相互通信が必要に
3. [課題] アプリケーションコードの変更や、
   対向サービスとの結合度が強くなってしまう



AWS App Mesh を導入することで、上記の課題の解決を図る

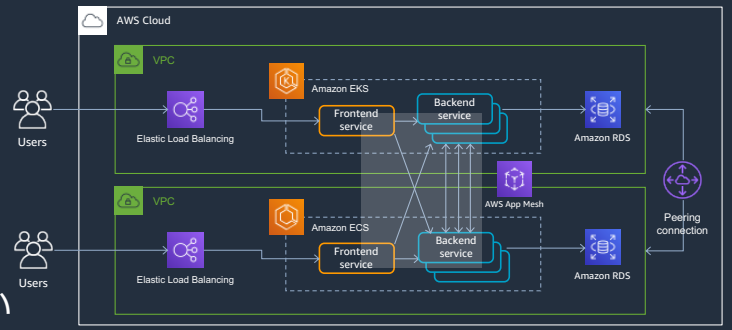
AWS App Mesh の導入後のサービス間通信

アプリケーションコードの変更：

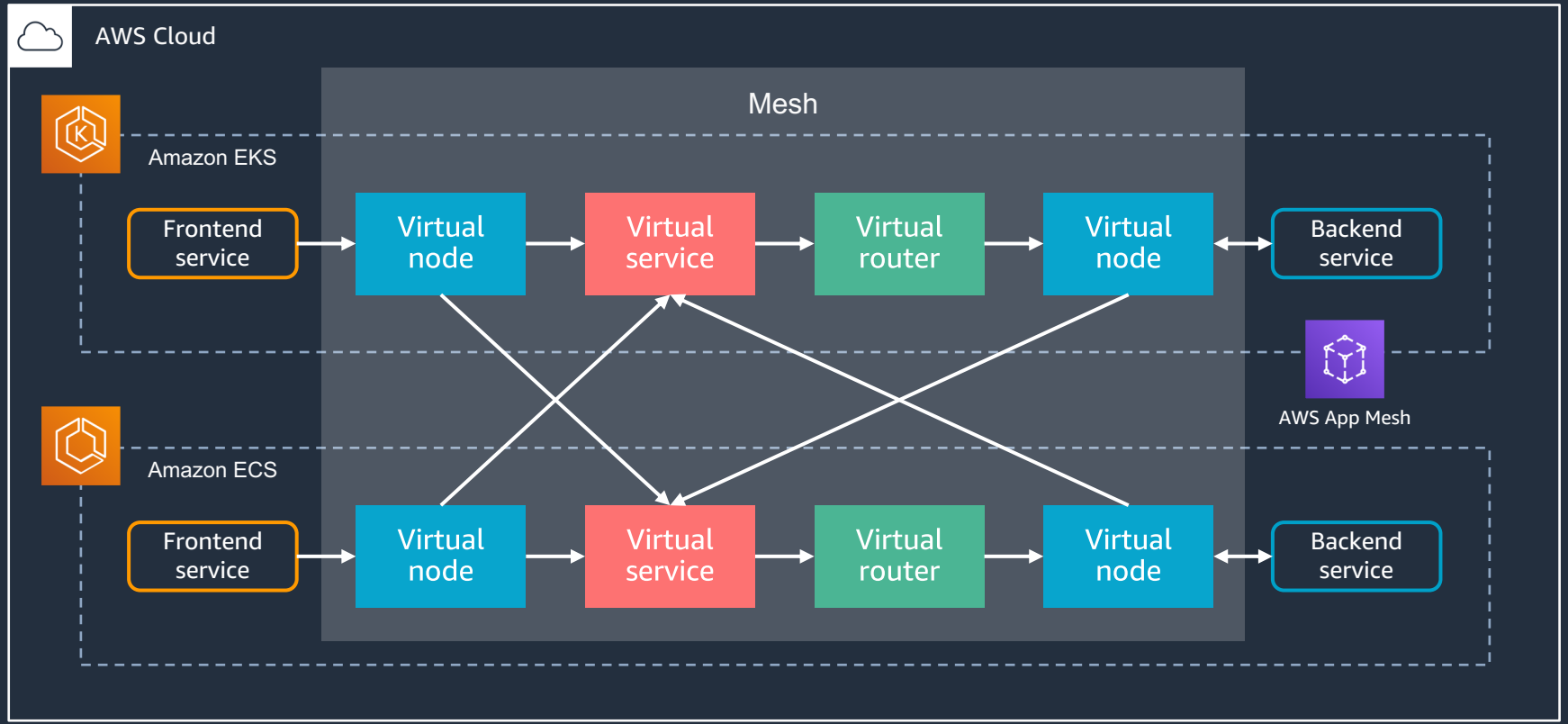
- リトライやタイムアウトの処理を Envoy proxy 側で行うため、アプリケーション側の変更は最小限に

対向サービスとの結合度：

- 実際のルーティング設定は AWS App Mesh 側で行うため、対向サービスの状態や構成変更に依存しない



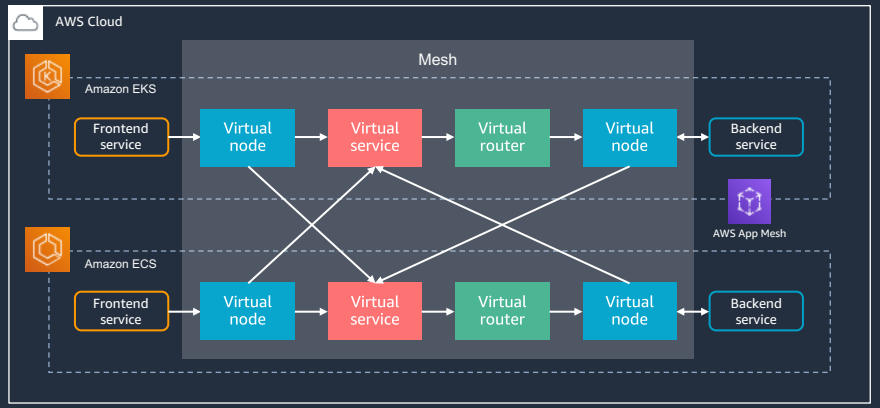
例: AWS App Mesh 導入後の構成



AWS App Mesh の導入手順

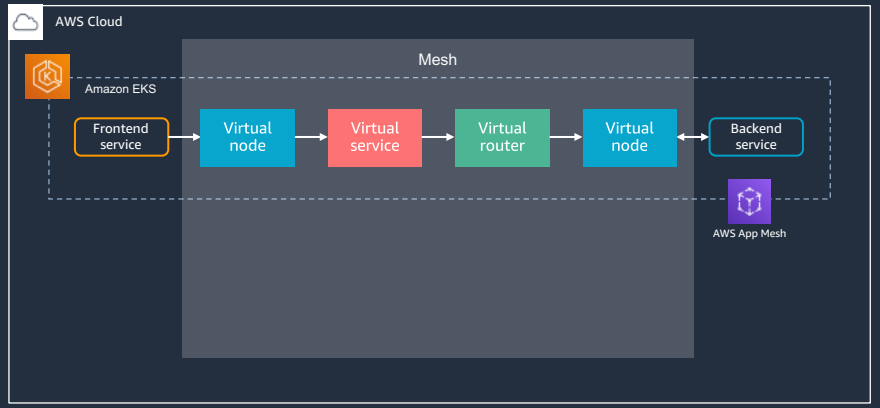
今回は以下の手順で AWS App Mesh を導入する

1. Amazon EKS での設定
   - ➢ AWS App Mesh Controller For K8s による設定
2. Amazon ECS での設定
   - ➢ マネジメントコンソールによる設定
3. Amazon EKS での設定
   - ➢ Virtual node の更新



AWS App Mesh の導入手順

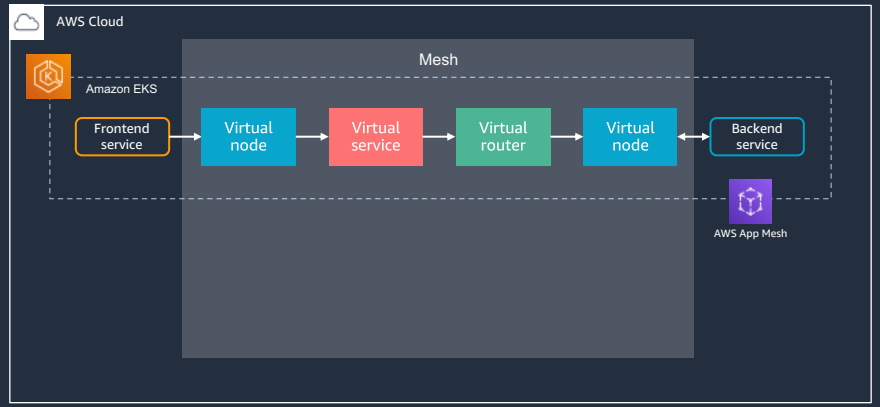
1. Amazon EKS での設定
   - ➢ AWS App Mesh Controller For K8s による設定
2. Amazon ECS での設定
   - ➢ マネジメントコンソールによる設定
3. Amazon EKS での設定
   - ➢ Virtual node の更新



AWS App Mesh の導入: Amazon EKS での設定

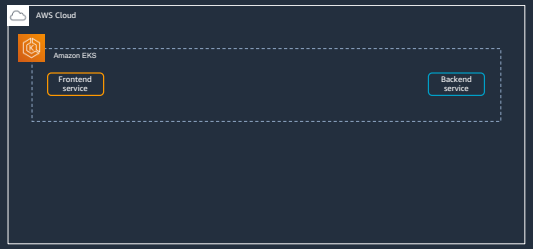
Amazon EKS での導入手順はこちら

1. AWS App Mesh Controller For K8s のインストール
2. Namespace の変更
3. Mesh の作成
4. AWS Cloud Map 名前空間の作成
5. Virtual node の作成
6. Virtual router の作成
7. Virtual service の作成
8. Virtual node の更新
9. Deployment の更新



- https://docs.aws.amazon.com/eks/latest/userguide/mesh-k8s-integration.html
- https://aws.github.io/aws-app-mesh-controller-for-k8s/guide/installation/

AWS App Mesh の導入: Amazon EKS での設定

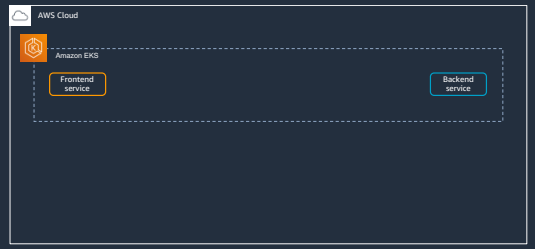


1. AWS App Mesh Controller For K8s のインストール

前提条件:

- Kubernetes クラスターのバージョンが 1.13 以上
- AWS App Mesh および AWS Cloud Map を操作する権限が付与されている
  - 管理ポリシー AWSAppMeshFullAccess / AWSCloudMapFullAccess
  - ノード IAM ロール、もしくは IAM roles for service accounts で設定

AWS App Mesh の導入: Amazon EKS での設定



1. AWS App Mesh Controller For K8s のインストール

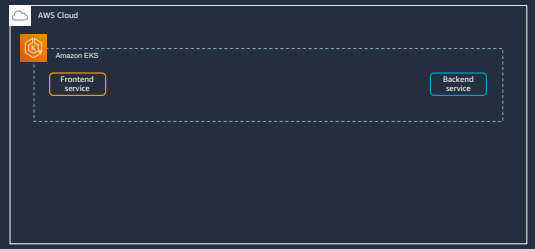
```
# eks-charts リポジトリを Helm に追加
$ helm repo add eks https://aws.github.io/eks-charts

# App Mesh 用の custom resource definitions (CRD) を追加
$ kubectl apply -k github.com/aws/eks-charts/stable/appmesh-controller//crds?ref=master

# App Mesh Controller を実行する Namespace を追加
$ kubectl create ns appmesh-system

# App Mesh Controller をデプロイ
$ helm upgrade -i appmesh-controller eks/appmesh-controller --namespace appmesh-system
```

AWS App Mesh の導入: Amazon EKS での設定



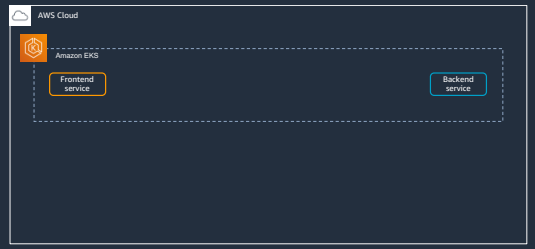
2. Namespace の変更

Mesh の指定、および Sidecar Injector の有効化を行う

例) Namespace *my-apps*

```
apiVersion: v1
kind: Namespace
metadata:
  name: my-apps
  labels:
    mesh: my-mesh
    appmesh.k8s.aws/sidecarInjectorWebhook: enabled
```

AWS App Mesh の導入: Amazon EKS での設定



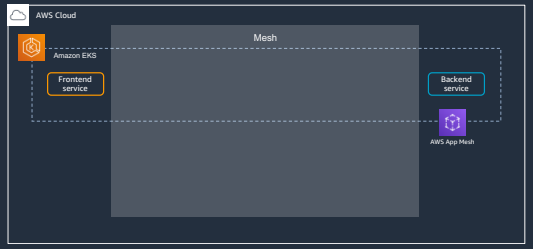
[補足] Sidecar Injector の設定について

Namespace の Label / Pod の Annotation で Inject Mode を制御 #1

- enabled mode
  - Sidecar が追加される
  - Virtual Node / Virtual Gateway の設定がされていない場合、Pod の起動に失敗する
- disabled mode
  - Sidecar の追加処理がスキップされる
- unspecified mode
  - Virtual Node / Virtual Gateway の設定がされている場合、Sidecar が追加される

#1: *appmesh.k8s.aws/sidecarInjectorWebhook*

AWS App Mesh の導入: Amazon EKS での設定



3. Mesh の作成

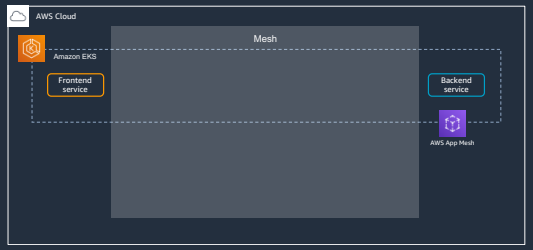
spec.namespaceSelector:

- Mesh のメンバーとなるリソースが所属する Namespace

例) Mesh *my-mesh*

```
apiVersion: appmesh.k8s.aws/v1beta2
kind: Mesh
metadata:
  name: my-mesh
spec:
  namespaceSelector:
    matchLabels:
      mesh: my-mesh
```

AWS App Mesh の導入: Amazon EKS での設定

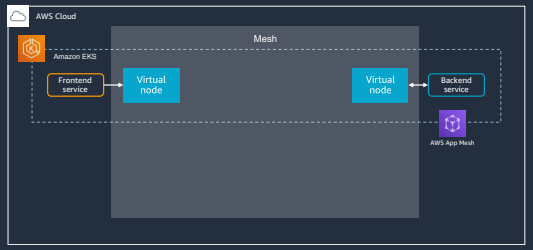


4. AWS Cloud Map 名前空間の作成

Virtual node のサービスディスカバリに AWS Cloud Map を使用

```
$ aws servicediscovery create-private-dns-namespace \
--name my-mesh.svc.cluster.local \
--vpc {{VPC_ID}}
```

AWS App Mesh の導入: Amazon EKS での設定



5. Virtual node の作成

spec.listeners:

- Virtual node のリクエスト受信設定

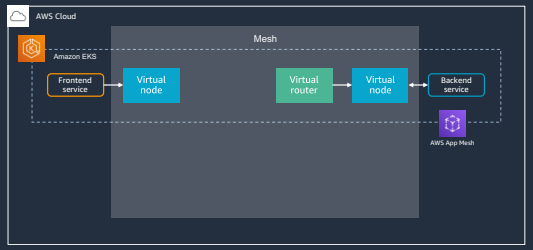
spec.serviceDiscovery:

- Virtual node と関連づけられている、実際のアプリケーションのサービスディスカバリ情報

```
apiVersion: appmesh.k8s.aws/v1beta2
kind: VirtualNode
metadata:
  name: eks-frontend
  namespace: my-apps
spec:
  podSelector:
    matchLabels:
      app: eks-frontend
  listeners:
    - portMapping:
        port: 80
        protocol: http
serviceDiscovery:
    awsCloudMap:
      namespaceName: my-mesh.svc.cluster.local
      serviceName: eks-frontend
```

例) Virtual node *eks-frontend*

AWS App Mesh の導入: Amazon EKS での設定



5. Virtual router の作成

spec.listeners:

- Virtual router のリクエスト受信設定

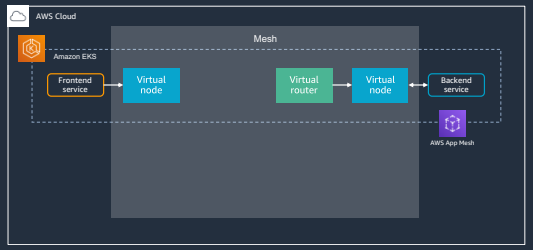
spec.routes:

- ルーティング情報

```
apiVersion: appmesh.k8s.aws/v1beta2
kind: VirtualRouter
metadata:
  namespace: my-apps
  name: eks-backend-virtual-router
spec:
  listeners:
    - portMapping:
        port: 80
        protocol: http
  routes:
    - name: eks-backend-route
      httpRoute:
        match:
          prefix: /
        action:
          weightedTargets:
            - virtualNodeRef:
                name: eks-backend
              weight: 1
```

例) Virtual router *eks-backend-virtual-router*

AWS App Mesh の導入: Amazon EKS での設定



5. Virtual router の作成

spec.routes[].httpRoute:

- HTTP ルーティング情報
  - http2Route や grpcRoute も設定可能

spec.routes[].httpRoute.match:

- route で処理をするリクエストの設定

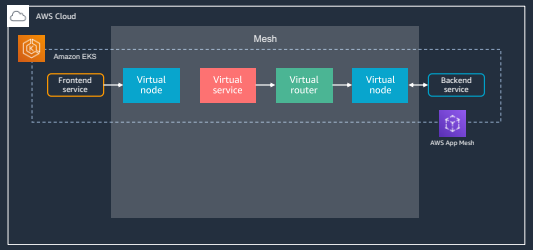
spec.routes[].httpRoute.action:

- マッチしたリクエストの転送先

```
apiVersion: appmesh.k8s.aws/v1beta2
kind: VirtualRouter
metadata:
  namespace: my-apps
  name: eks-backend-virtual-router
spec:
  listeners:
    - portMapping:
        port: 80
        protocol: http
  routes:
    - name: eks-backend-route
      httpRoute:
        match:
          prefix: /
        action:
          weightedTargets:
            - virtualNodeRef:
                name: eks-backend
              weight: 1
```

例) Virtual router *eks-backend-virtual-router*

AWS App Mesh の導入: Amazon EKS での設定



6. Virtual service の作成

spec.awsName:

- Virtual service のリソース名

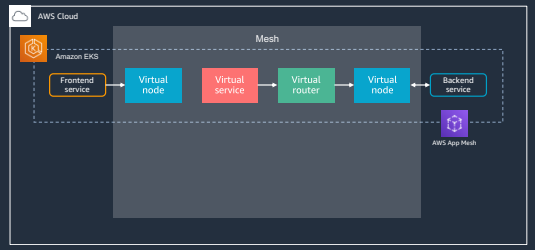
spec.provider:

- Virtual service のリクエスト転送先
  - Virtual router / Virtual node を指定

例) Virtual service *eks-backend*

```
apiVersion: appmesh.k8s.aws/v1beta2
kind: VirtualService
metadata:
  name: eks-backend
  namespace: my-apps
spec:
  awsName: eks-backend.my-apps.svc.cluster.local
  provider:
    virtualRouter:
      virtualRouterRef:
        name: eks-backend-virtual-router
```

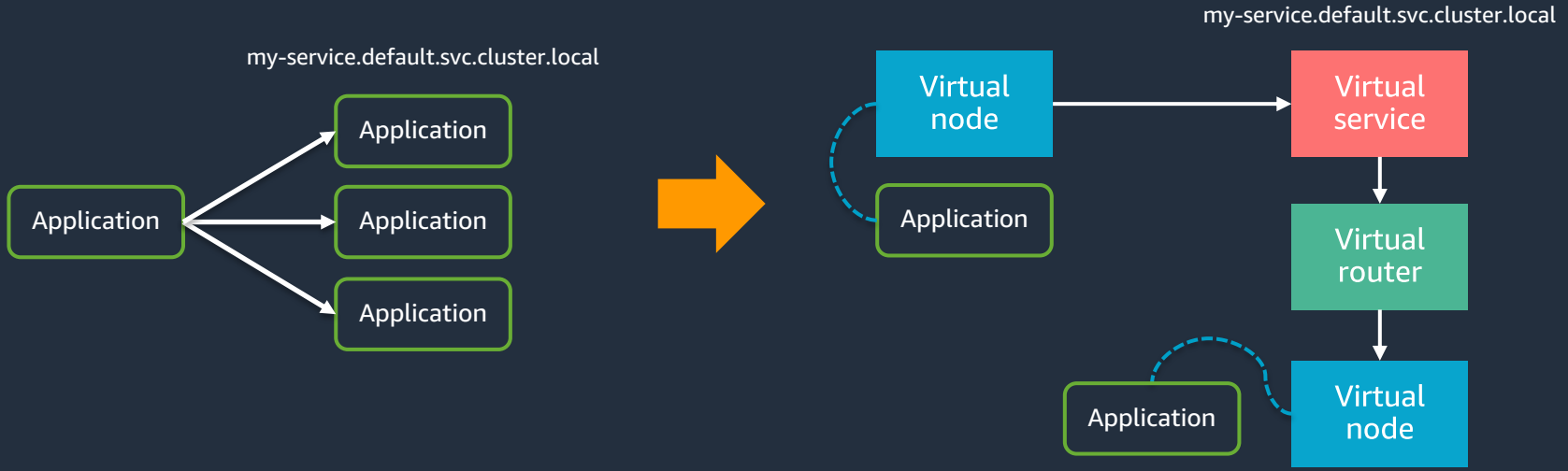
AWS App Mesh の導入: Amazon EKS での設定



[補足] Virtual service のリソース名について

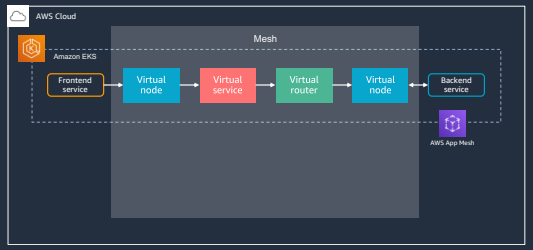
Virtual service name と実際のサービスディスカバリ名を揃えることを推奨 #1

- 既存のコードを変更する必要がないため



#1: https://docs.aws.amazon.com/app-mesh/latest/userguide/virtual_services.html#create-virtual-service

AWS App Mesh の導入: Amazon EKS での設定



7. Virtual node の更新

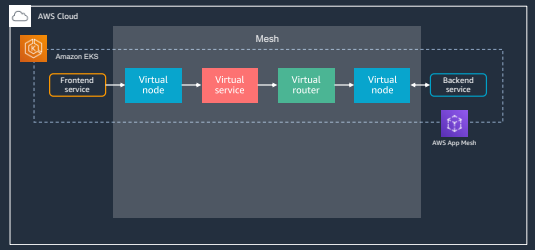
spec.backends:

- Virtual node がリクエストを送信する Virtual service
- 先ほど作成した VirtualService custom resource のリソース名を指定

```
apiVersion: appmesh.k8s.aws/v1beta2
kind: VirtualNode
metadata:
  name: eks-frontend
  namespace: my-apps
spec:
  podSelector:
    matchLabels:
      app: eks-frontend
  listeners:
    - portMapping:
        port: 80
        protocol: http
  backends:
    - virtualService:
        virtualServiceRef:
          name: eks-backend
  serviceDiscovery:
    awsCloudMap:
      namespaceName: my-mesh.svc.cluster.local
      serviceName: eks-frontend
```

例) Virtual node *eks-frontend*

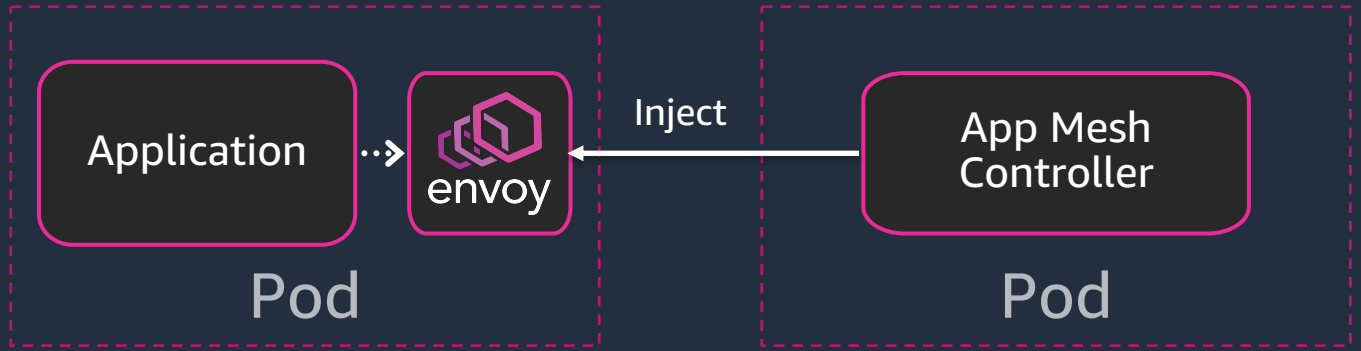
AWS App Mesh の導入: Amazon EKS での設定



8. Deployment の更新

Envoy proxy を Pod へ追加するためにローリングアップデートを行う

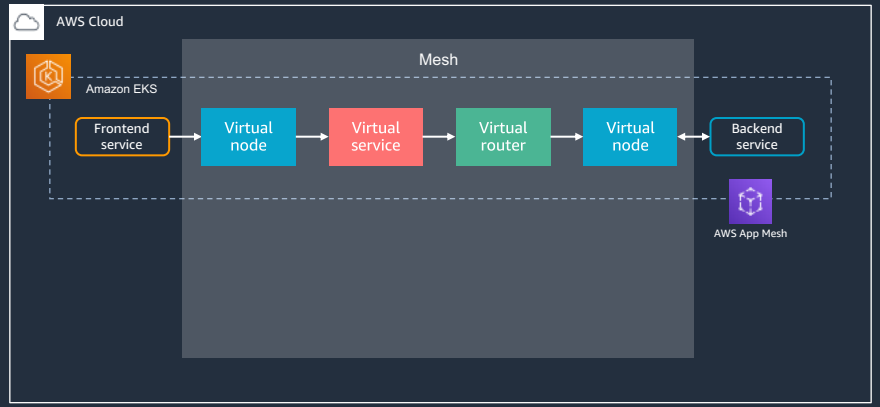
```
$ kubectl rollout restart deployment/XXX
...
```



AWS App Mesh の導入手順: Amazon EKS での設定

[再掲] 実施した手順はこちら

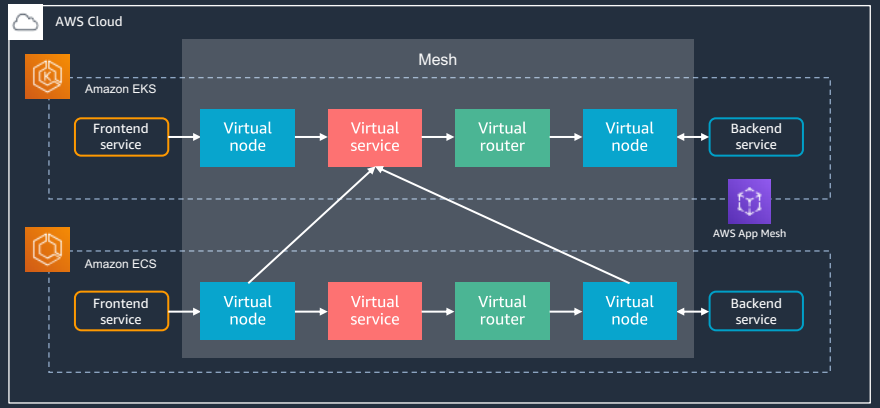
1. AWS App Mesh Controller For K8s のインストール
2. Namespace の変更
3. Mesh の作成
4. AWS Cloud Map 名前空間の作成
5. Virtual node の作成
6. Virtual router の作成
7. Virtual service の作成
8. Virtual node の更新
9. Deployment の更新



- https://docs.aws.amazon.com/eks/latest/userguide/mesh-k8s-integration.html
- https://aws.github.io/aws-app-mesh-controller-for-k8s/guide/installation/

AWS App Mesh の導入手順

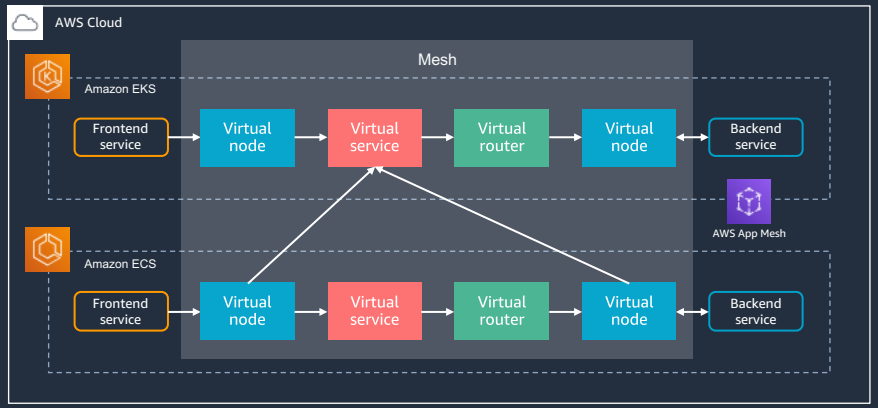
1. Amazon EKS での設定
   - ➢ AWS App Mesh Controller For K8s による設定
2. Amazon ECS での設定
   - ➢ マネジメントコンソールによる設定
3. Amazon EKS での設定
   - ➢ Virtual node の更新



AWS App Mesh の導入: Amazon ECS での設定

Amazon ECS での手順はこちら

1. Virtual node の作成
2. Virtual router の作成
3. Route の作成
4. Virtual service の作成
5. Virtual node の更新
6. タスク定義の更新
7. サービスの更新

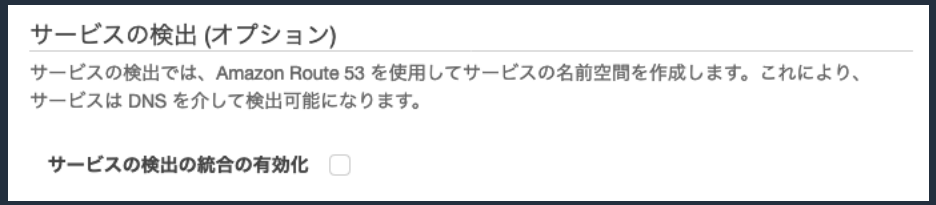


https://docs.aws.amazon.com/AmazonECS/latest/developerguide/appmesh-getting-started.html

AWS App Mesh の導入: Amazon ECS での設定

[推奨] ECS サービスでサービス検出の統合が有効になっていること

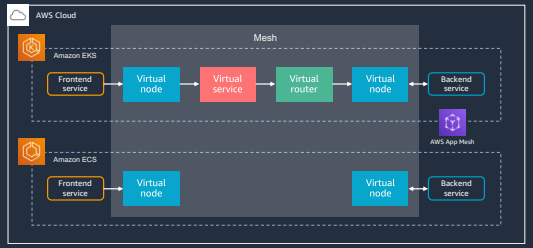
- Virtual node の背後にいる実際のアプリケーションを検出するため
  - ➢ 自前の仕組みを用意することも可能だが、運用面からサービス検出の利用を推奨



- [注意] ECS サービスの作成後、後からサービス検出の設定を変更することはできない #1

#1: https://docs.aws.amazon.com/AmazonECS/latest/developerguide/appmesh-getting-started.html

AWS App Mesh の導入: Amazon ECS での設定

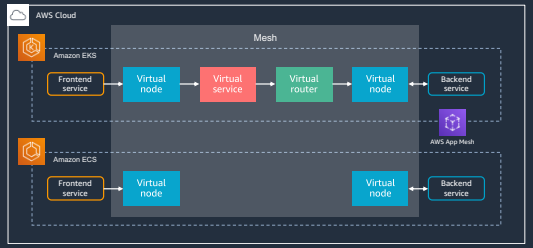


1. Virtual node の作成

サービスの検出方法:

- ECS サービスで設定したサービス検出の AWS Cloud Map 情報

AWS App Mesh の導入: Amazon ECS での設定

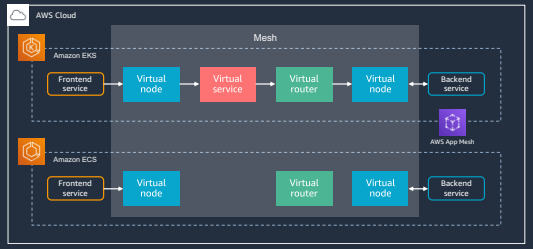


1. Virtual node の作成

リスナー設定:

- Virtual node のリクエスト受信設定

AWS App Mesh の導入: Amazon ECS での設定

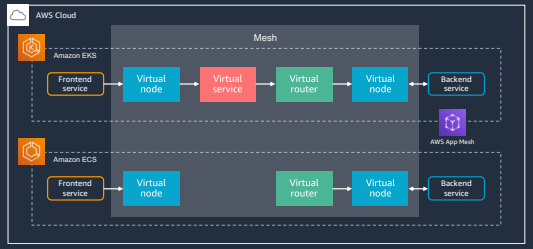


2. Virtual router の作成

リスナー設定:

- Virtual router のリクエスト受信設定

AWS App Mesh の導入: Amazon ECS での設定



3. Route の作成

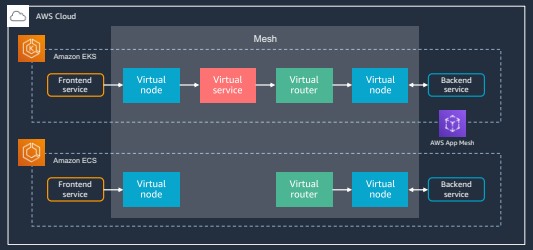
ルートタイプ:

- Route で利用するプロトコル

ターゲット:

- リクエストの転送先

AWS App Mesh の導入: Amazon ECS での設定



[補足] Amazon EKS の手順における Route の作成について

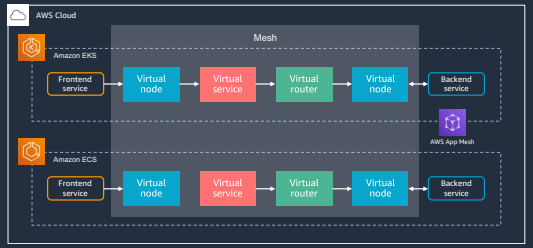
VirtualRouter custom resource にて Route リソースの作成も
同時に行われる

- Virtual router の定義
- Route の定義

```
apiVersion: appmesh.k8s.aws/v1beta2
kind: VirtualRouter
metadata:
  namespace: my-apps
  name: eks-backend-virtual-router
spec:
  listeners:
    - portMapping:
        port: 80
        protocol: http
  routes:
    - name: eks-backend-route
      httpRoute:
        match:
          prefix: /
        action:
          weightedTargets:
            - virtualNodeRef:
                name: eks-backend
              weight: 1
```

例) Virtual router *eks-backend-virtual-router*

AWS App Mesh の導入: Amazon ECS での設定



4. Virtual service の作成

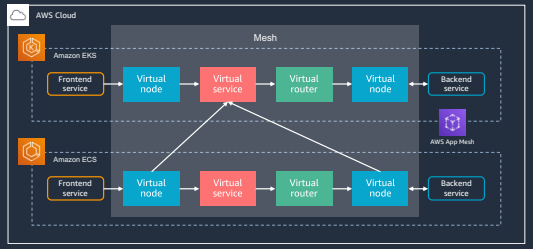
仮想サービス名:

- 実際のサービスディスカバリ名

プロバイダ:

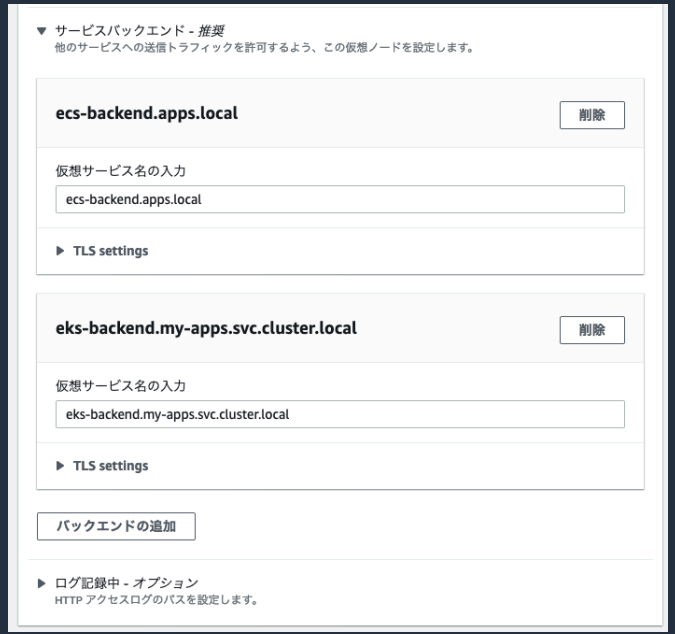
- リクエストの転送先

AWS App Mesh の導入: Amazon ECS での設定

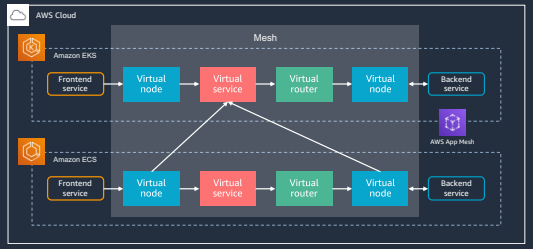


5. Virtual node の更新

- 先ほど作成した Virtual service をバックエンドに追加
- EKS 側の Virtual service も追加



AWS App Mesh の導入: Amazon ECS での設定

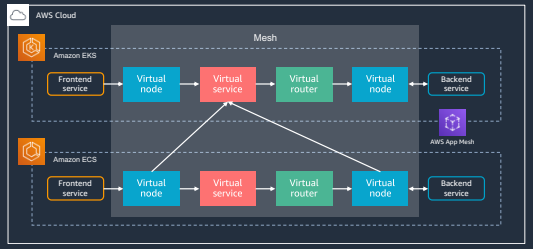


6. タスク定義の更新

- 「App Mesh 統合の有効化」をチェック
- 作成した AWS App Mesh リソースを関連づける
- AWS App Mesh が提供する Envoy イメージを設定 #1

#1: https://docs.aws.amazon.com/app-mesh/latest/userguide/envoy.html

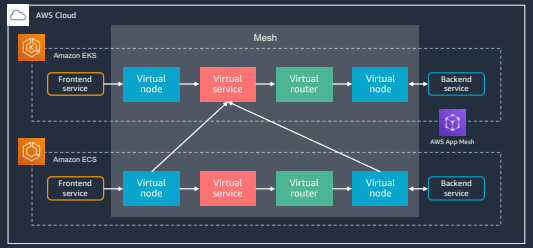
AWS App Mesh の導入: Amazon ECS での設定



6. タスク定義の更新

- マネジメントコンソールにて「サービス統合」を適用することで、「プロキシ設定」は自動で設定される

AWS App Mesh の導入: Amazon ECS での設定



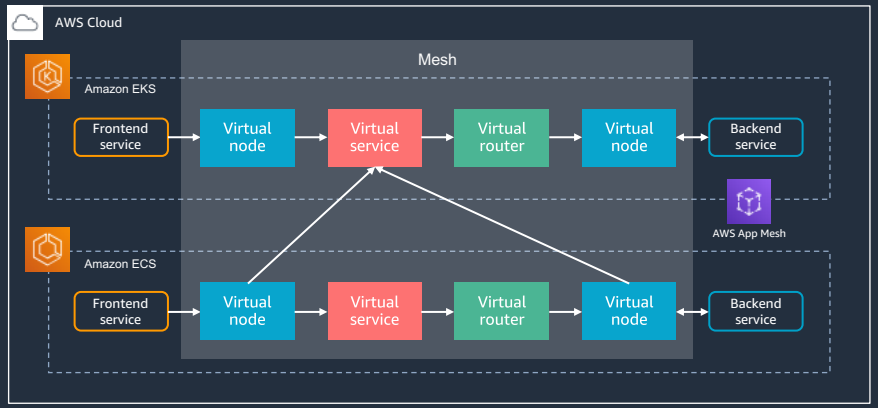
7. サービスの更新

作成したタスク定義のリビジョンを利用してサービスを更新

AWS App Mesh の導入: Amazon ECS での設定

[再掲] 実施した手順はこちら

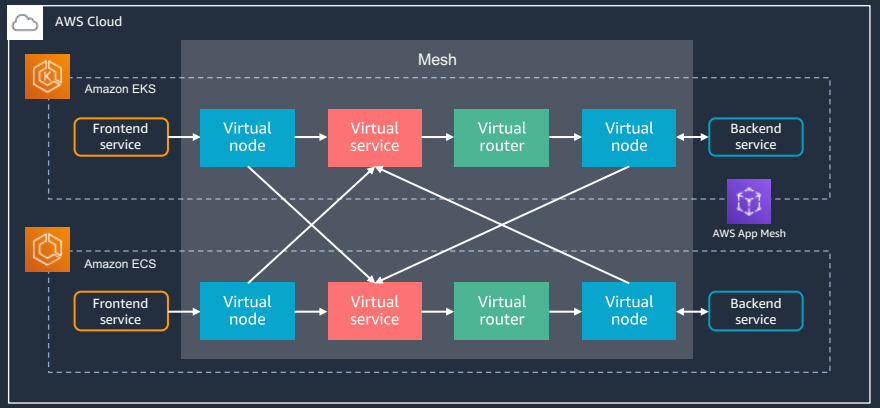
1. Virtual node の作成
2. Virtual router の作成
3. Route の作成
4. Virtual service の作成
5. Virtual node の更新
6. タスク定義の更新
7. サービスの更新



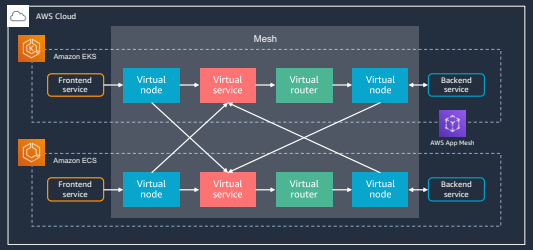
https://docs.aws.amazon.com/AmazonECS/latest/developerguide/appmesh-getting-started.html

AWS App Mesh の導入手順

1. Amazon EKS での設定
   - ➢ AWS App Mesh Controller For K8s による設定
2. Amazon ECS での設定
   - ➢ マネジメントコンソールによる設定
3. Amazon EKS での設定
   - ➢ Virtual node の更新



AWS App Mesh の導入: Amazon EKS での設定



[Amazon EKS での手順] Virtual node の更新

- ECS 側の Virtual service を backends に追加

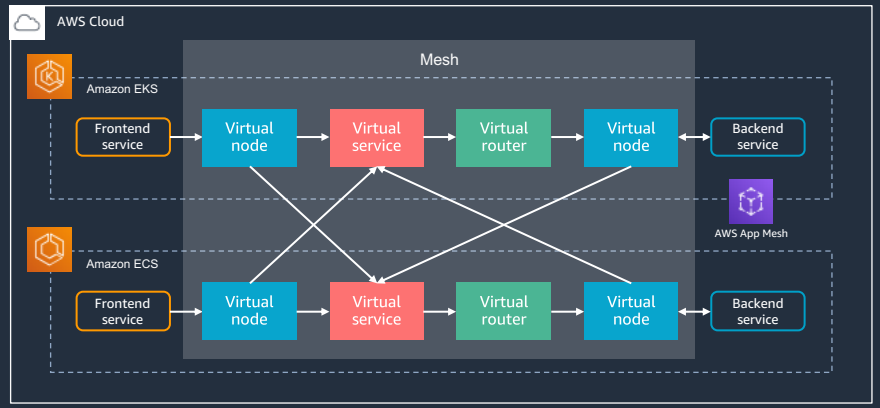
- [注意] ECS 側の Virtual service は ARN で指定する必要がある
  - AWS CLI で取得

```
(中略)
backends:
  - virtualService:
      virtualServiceRef:
        name: eks-backend
  - virtualService:
      virtualServiceARN:
        <ecs-backend-virtual-service-ARN>
```

```
$ aws appmesh describe-virtual-service --mesh-name my-mesh --virtual-service-name ecs-backend.apps.local --query 'virtualService.metadata.arn'
```

[再掲] AWS App Mesh の導入手順

1. Amazon EKS での設定
   - ➢ AWS App Mesh Controller For K8s による設定
2. Amazon ECS での設定
   - ➢ マネジメントコンソールによる設定
3. Amazon EKS での設定
   - ➢ Virtual node の更新



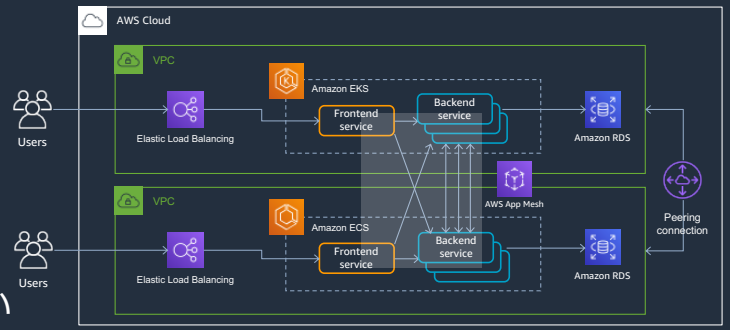
[再掲] AWS App Mesh の導入後のサービス間通信

アプリケーションコードの変更：

- リトライやタイムアウトの処理を Envoy proxy 側で行うため、アプリケーション側の変更は最小限に

対向サービスとの結合度：

- 実際のルーティング設定は AWS App Mesh 側で行うため、対向サービスの状態や構成変更に依存しない



アプリケーションが通信要件や他サービスから分離された状態を実現

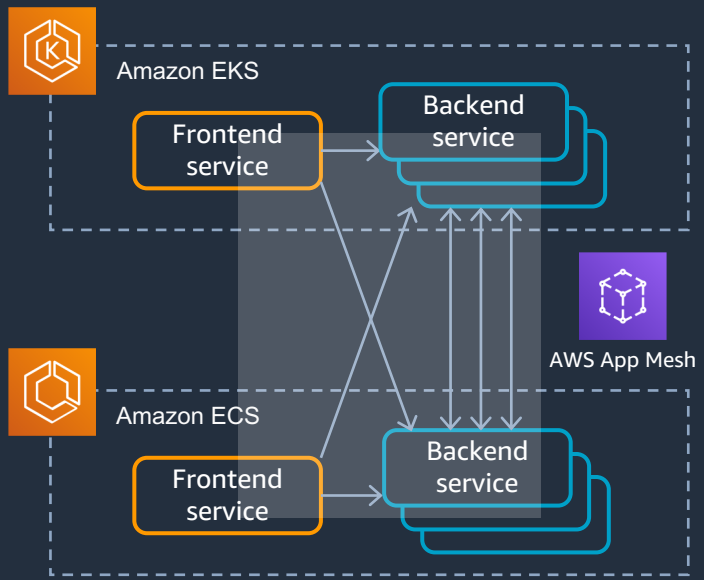
本日のアジェンダ

- サービスメッシュとは？
- AWS App Mesh の概要
- Example for AWS App Mesh
- AWS App Mesh の利用料金

利用料金

- AWS App Mesh は追加料金なしで使用可能
- Envoy Proxy を稼働するためのリソースに対して料金が発生
- AWS X-Ray、 Amazon CloudWatch Logs など連携先のサービスの料金が発生

まとめ



AWS App Mesh の導入：

- サービス間の通信に、一貫したネットワーク制御と可視性を提供
- アプリケーションが通信要件や他サービスから分離された状態を実現

AWS App Mesh 関連情報: 手を動かしながら理解する

AWS App Mesh Workshop
https://www.appmeshworkshop.com/

AWS App Mesh の機能と利用方法を理解するためのワークショップ

aws / aws-app-mesh-examples
https://github.com/aws/aws-app-mesh-examples

クロスアカウント、gRPC、リトライ、Header ベースルーティングなど、
活用例を確認可能なコードサンプル

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。

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