[AWS Black Belt Online Seminar]
AWS App Mesh

サービスカットシリーズ

Solutions Architect, Containers
Masatoshi Hayashi
2020/07/21

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

過去資料
https://amzn.to/JPArchive

自己紹介

林　政利

- 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 EKS
    - AWS Certificate Manager

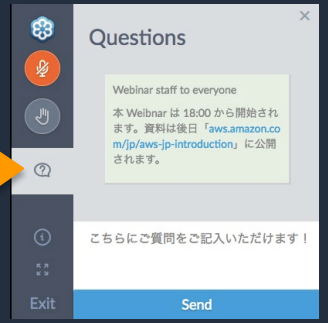
AWS Black Belt Online Seminar とは

「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7月21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
  - サービスメッシュとはどのようなものか
  - AWS App Meshとはどのようなサービスか
  - AWS App Meshの使いどころや具体的な機能
- 対象者
  - マイクロサービスの導入を検討していて、サービスメッシュに興味がある方
  - App Meshを聞いたことがあるが、実際にどのように役立てることができるか調査中の方

本日のアジェンダ

- サービスメッシュとは何か
- AWS App Mesh
    - 概要
    - 利用方法
    - 機能と活用例
    - ロードマップ
    - 価格体系

- サービスメッシュとは何か
- AWS App Mesh
    - 概要
    - 利用方法
    - 機能と活用例
    - ロードマップ
    - 価格体系

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

サービスメッシュがない場合

アプリケーションレベルの通信制御を、アプリケーション自身に組み込む

- 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ある場合

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

サービスメッシュを実現するAWS App Mesh

AWS App Mesh

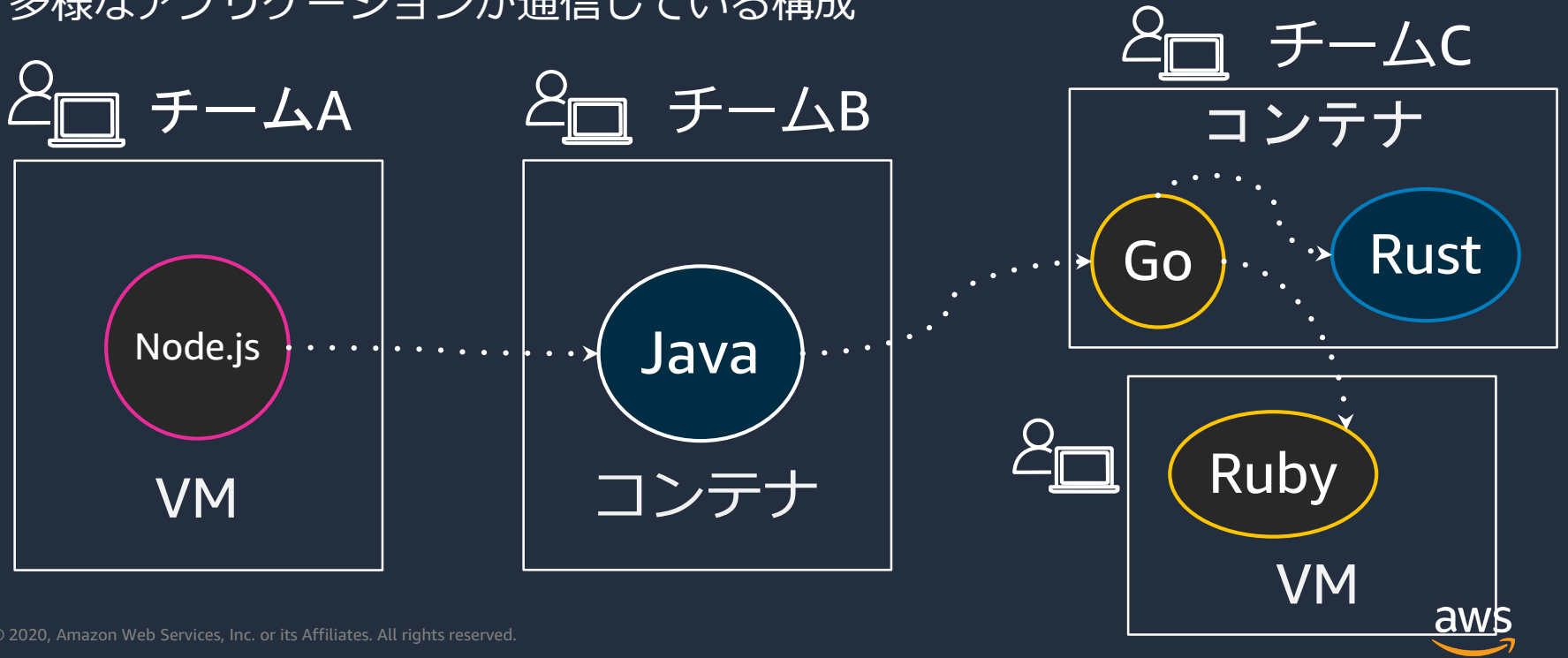
- HTTP通信のリトライやタイムアウト
- 通信のトレーシングやログ、メトリクスの取得
- TLSを使用した暗号化通信

サービスメッシュが求められるようになった背景

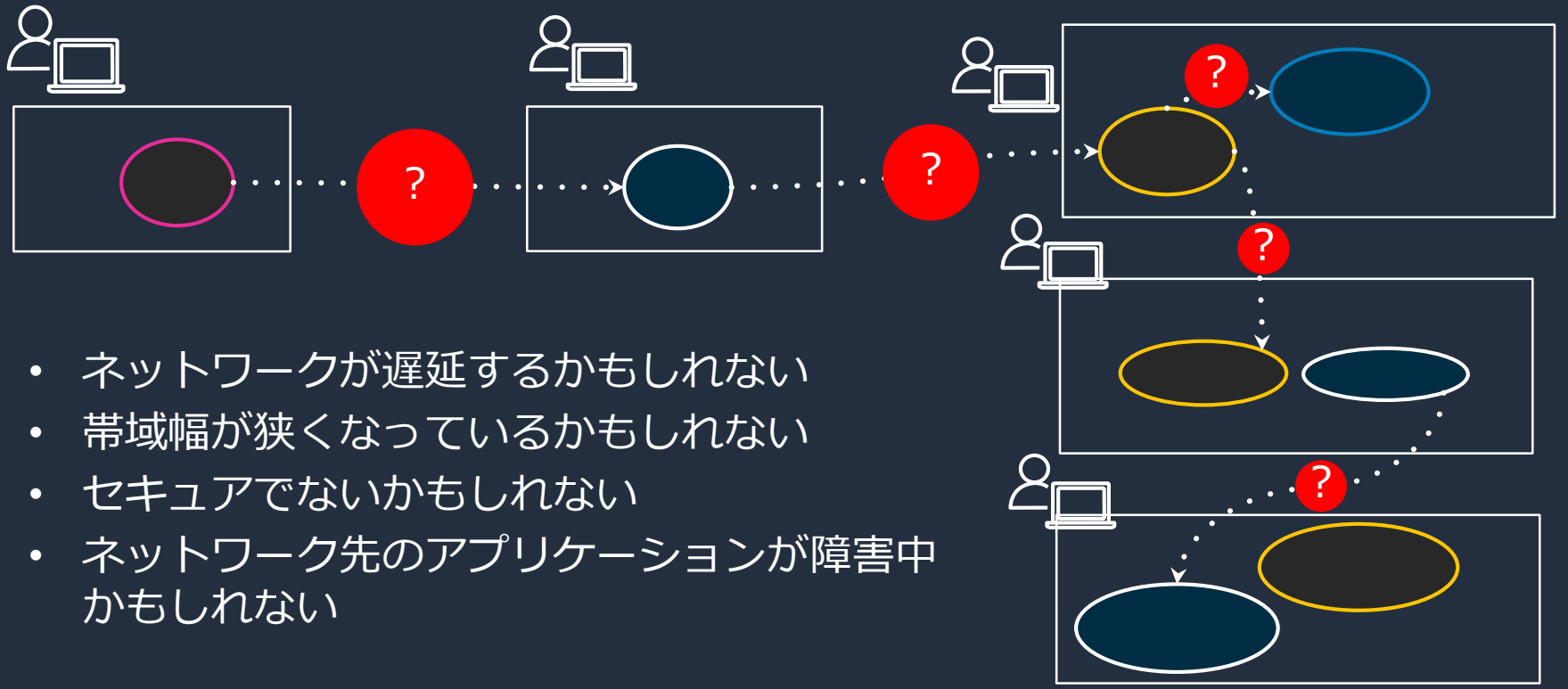
現代のシステムは、複数の言語、アーキテクチャ、アプリケーションで構成
クラウドにより多様な環境が簡単に作成できるようになった
マイクロサービスアーキテクチャの採用

多様なアプリケーションが通信しながらシステムを構成

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技術を選択できる
多様なアプリケーションが通信している構成

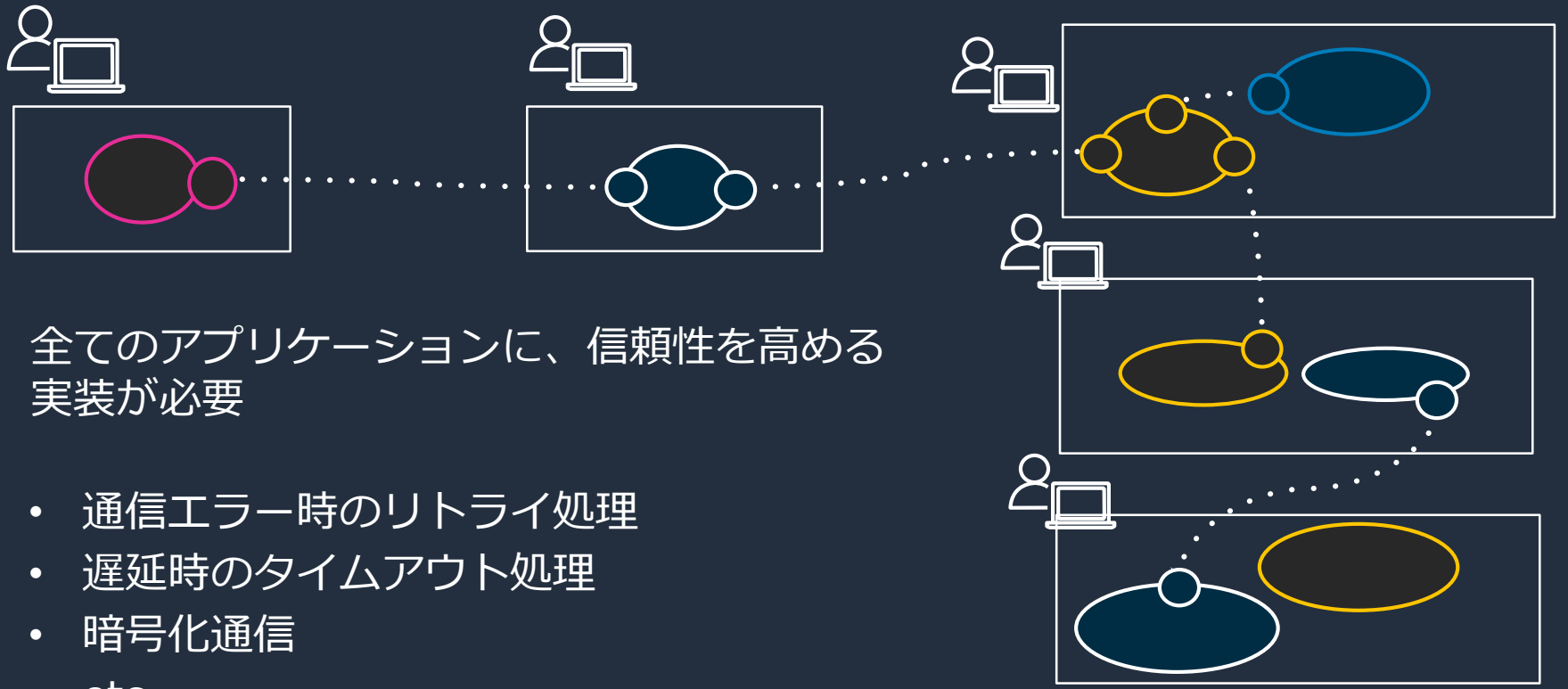


ネットワークは信頼できない



- ネットワークが遅延するかもしれない
- 帯域幅が狭くなっているかもしれない
- セキュアでないかもしれない
- ネットワーク先のアプリケーションが障害中かもしれない

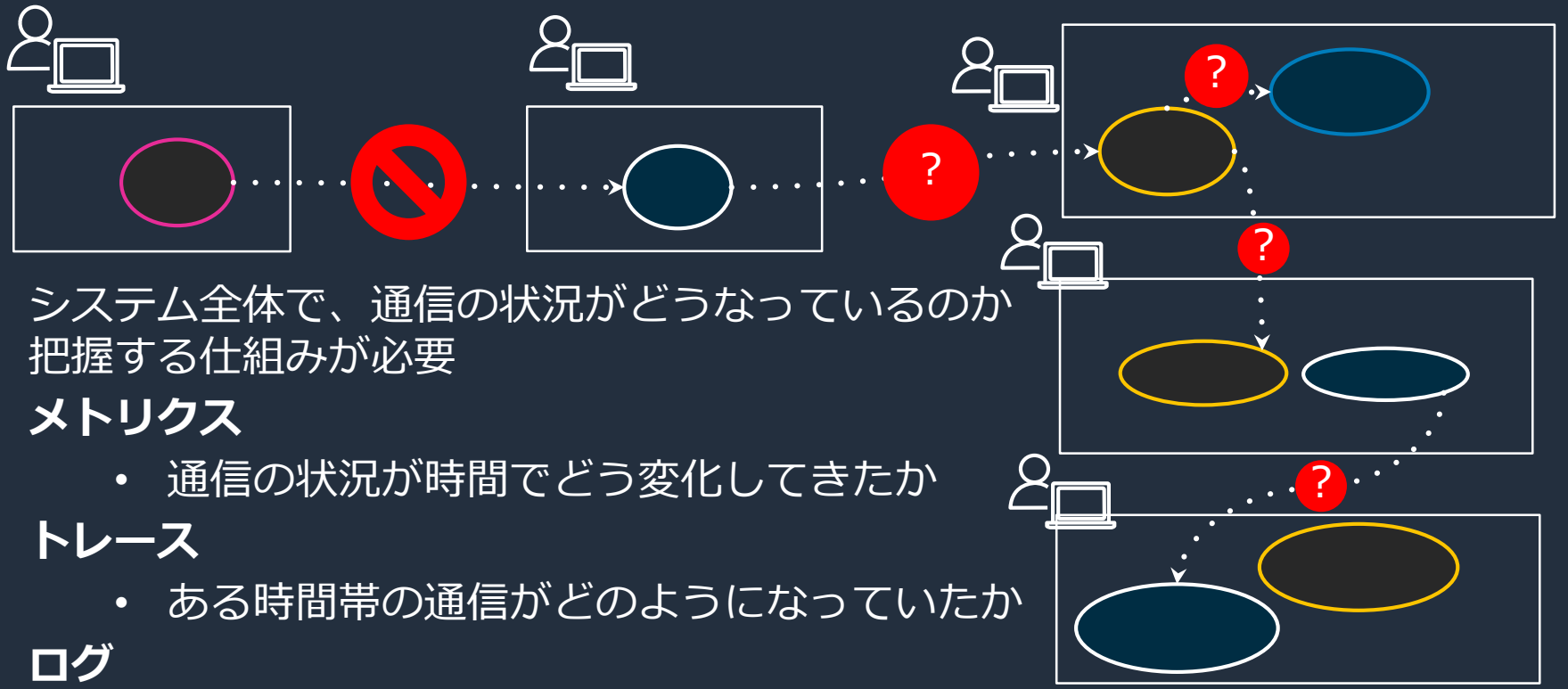
信頼性を高めるためのアプリケーション実装



全てのアプリケーションに、信頼性を高める実装が必要

- 通信エラー時のリトライ処理
- 遅延時のタイムアウト処理
- 暗号化通信
- etc.

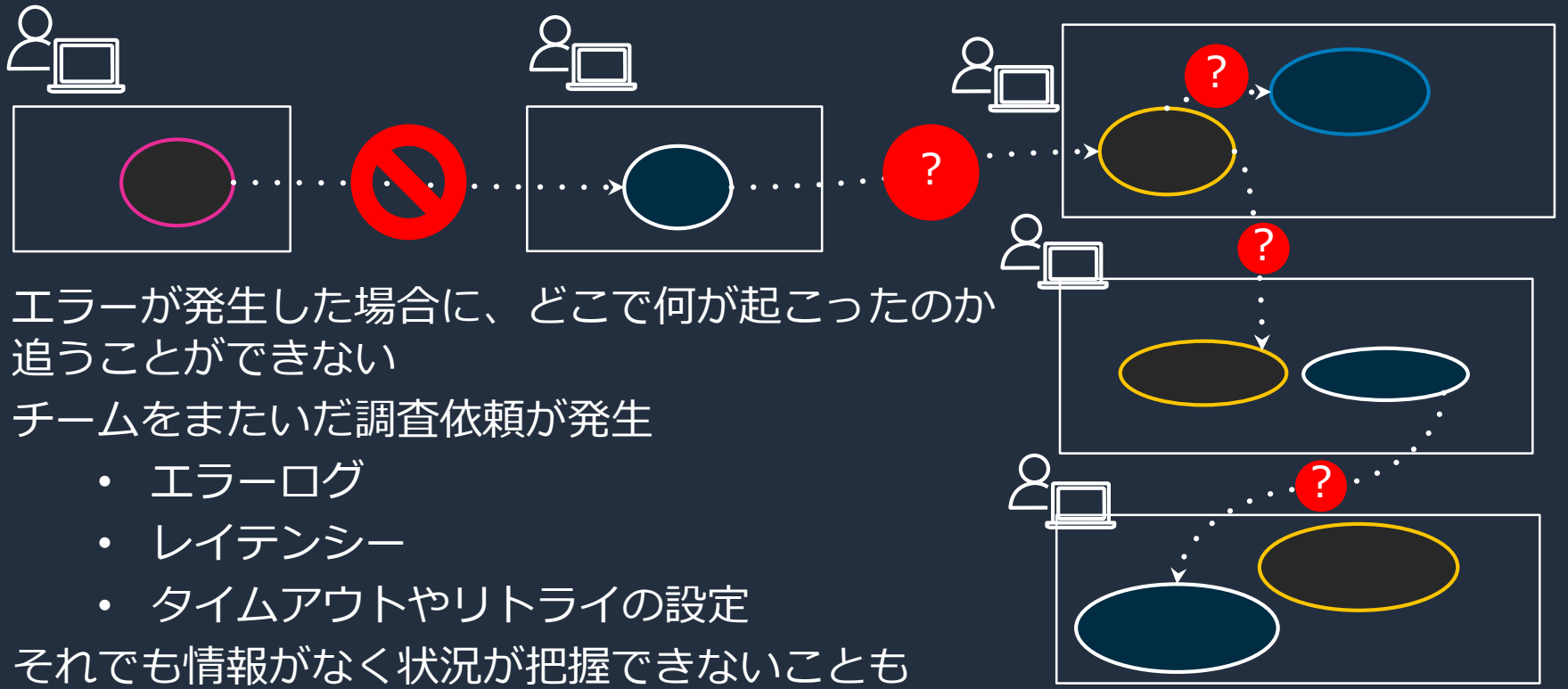
通信の可観測性(Observability)も必要



- システム全体で、通信の状況がどうなっているのか把握する仕組みが必要
- **メトリクス**
  - 通信の状況が時間でどう変化してきたか
- **トレース**
  - ある時間帯の通信がどのようになっていたか
- **ログ**
  - 通信に関する詳細な情報

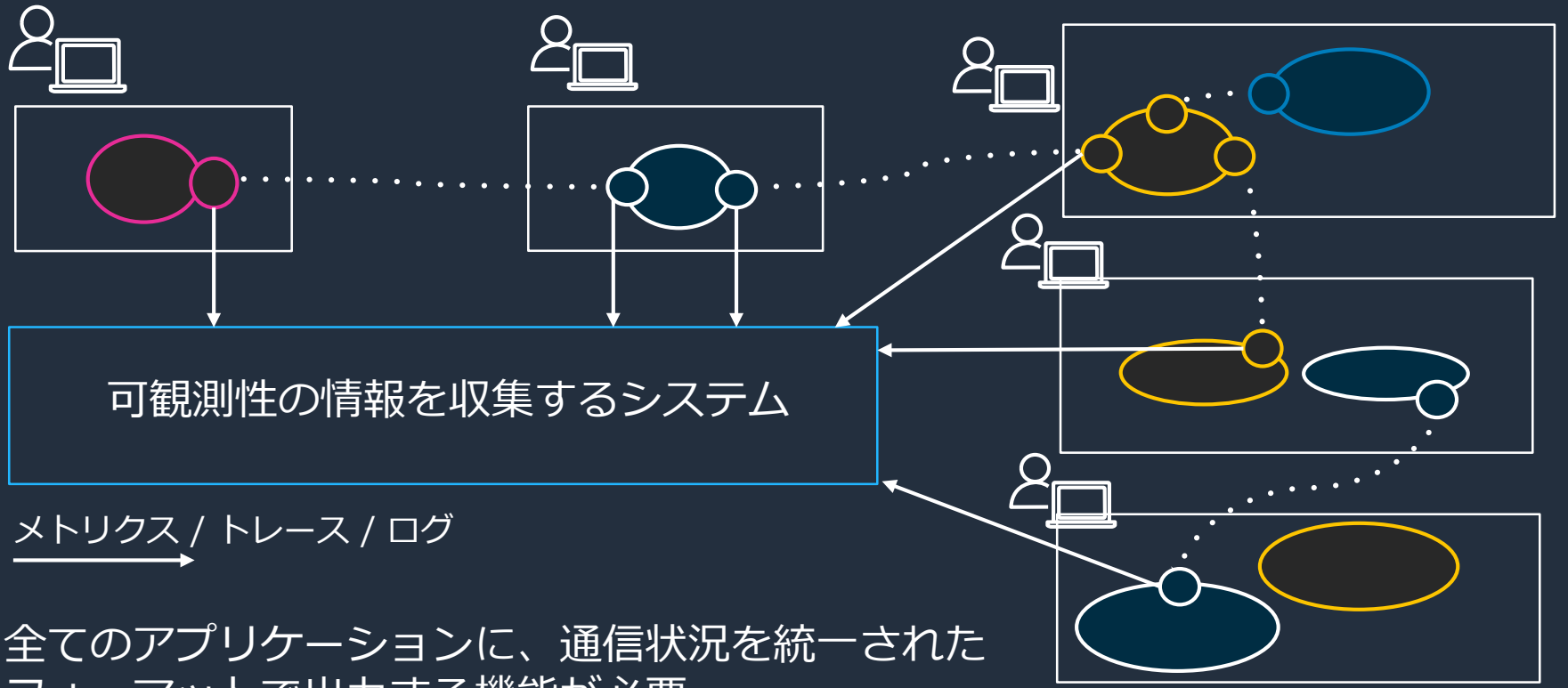
aws

通信の可観測性がないと…



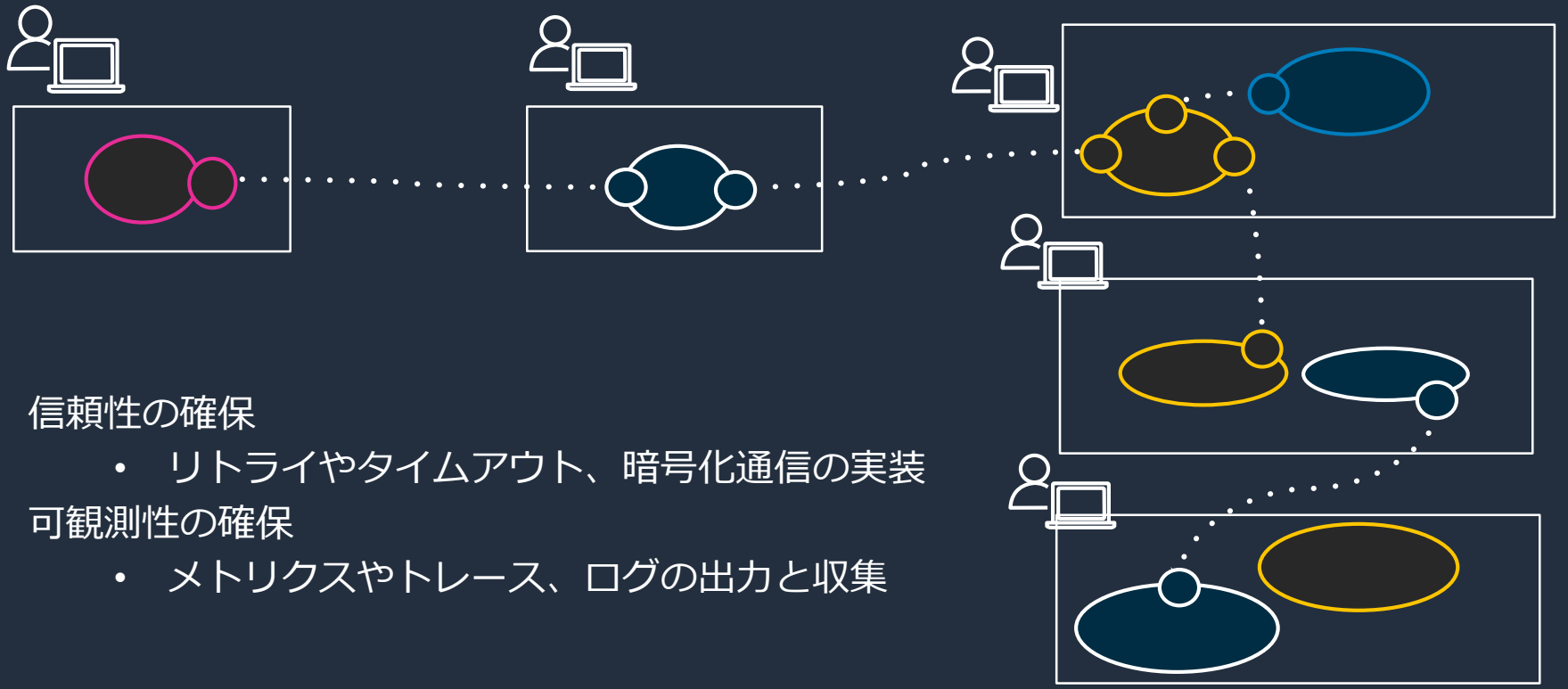
- エラーが発生した場合に、どこで何が起こったのか追うことができない
- チームをまたいだ調査依頼が発生
    - エラーログ
    - レイテンシー
    - タイムアウトやリトライの設定
- それでも情報がなく状況が把握できないことも

通信の可観測性を確保する実装



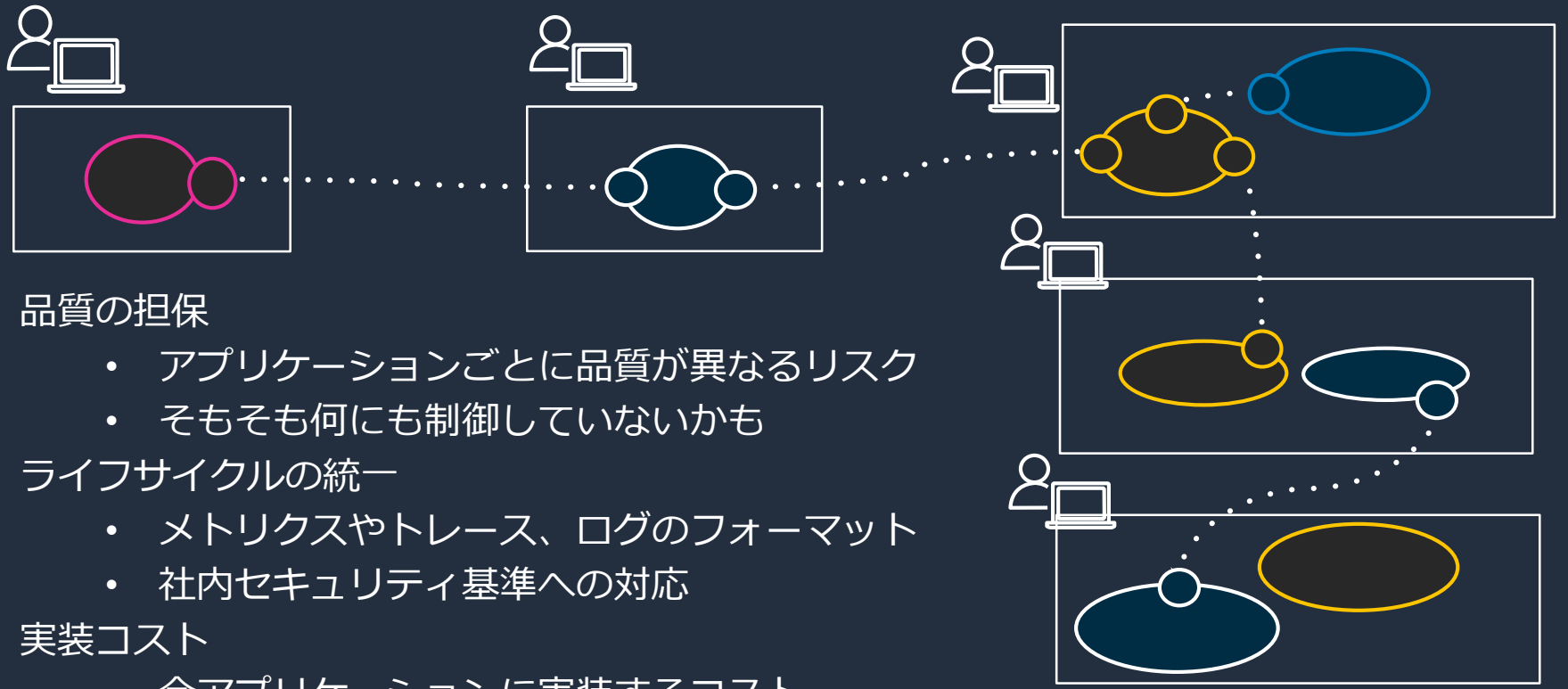
全てのアプリケーションに、通信状況を統一されたフォーマットで出力する機能が必要

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



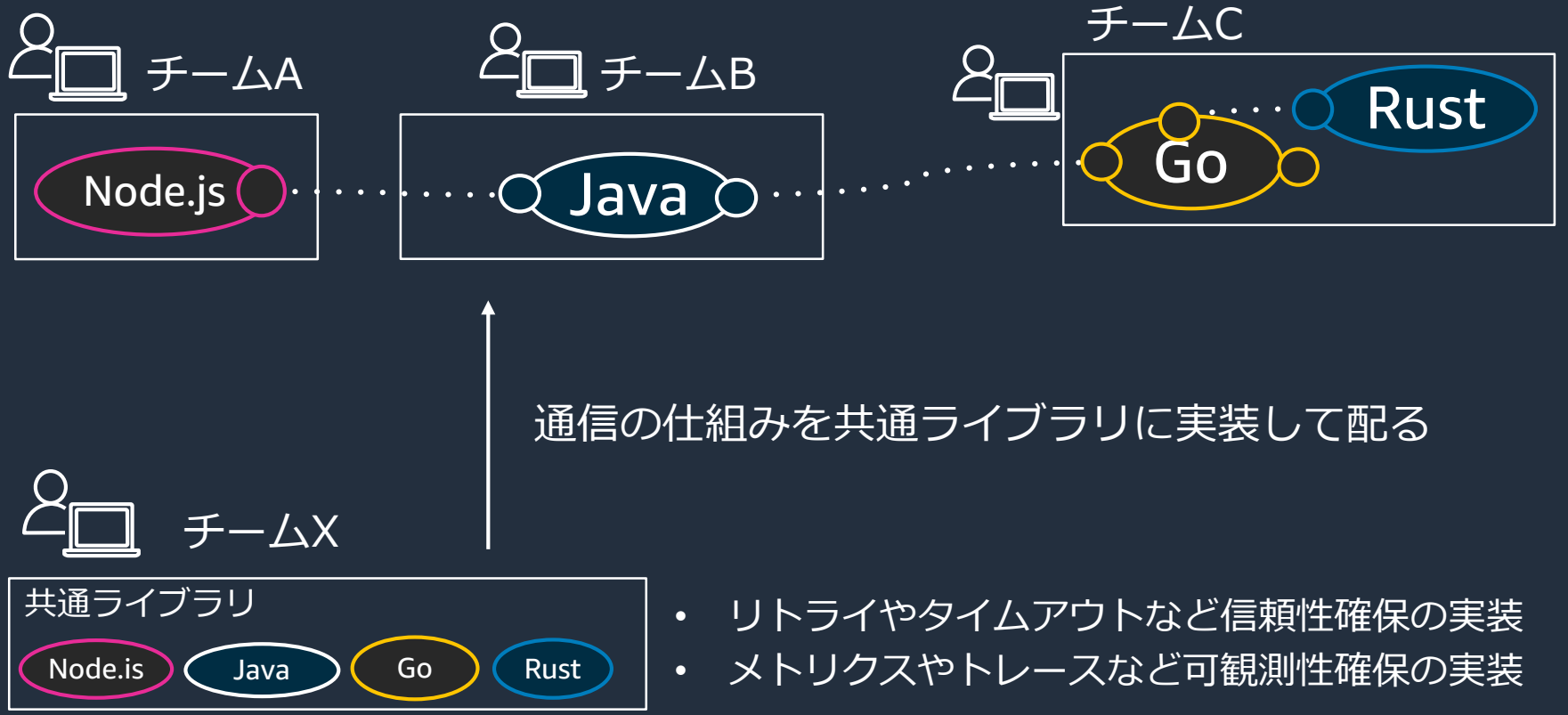
- 信頼性の確保
  - リトライやタイムアウト、暗号化通信の実装
- 可観測性の確保
  - メトリクスやトレース、ログの出力と収集

アプリケーションレベルの通信制御を共通化する必要性



- 品質の担保
  - アプリケーションごとに品質が異なるリスク
  - そもそも何にも制御していないかも
- ライフサイクルの統一
  - メトリクスやトレース、ログのフォーマット
  - 社内セキュリティ基準への対応
- 実装コスト
  - 全アプリケーションに実装するコスト

ライブラリによる通信制御の共通化



- リトライやタイムアウトなど信頼性確保の実装
- メトリクスやトレースなど可観測性確保の実装

チームX

共通ライブラリ

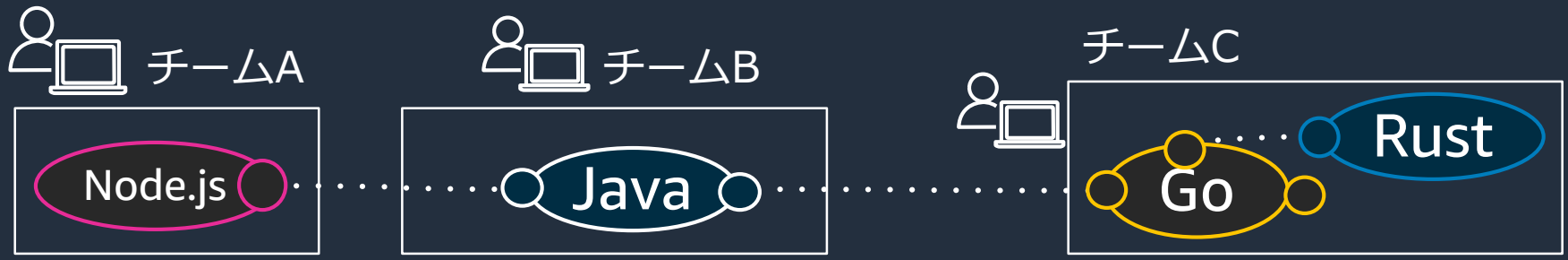
Node.is　Java　Go　Rust

通信の仕組みを共通ライブラリに実装して配る？？

- リトライやタイムアウトなど信頼性確保の実装
- メトリクスやトレースなど可観測性確保の実装

aws

アプリケーション開発チームでの共通ライブラリの課題



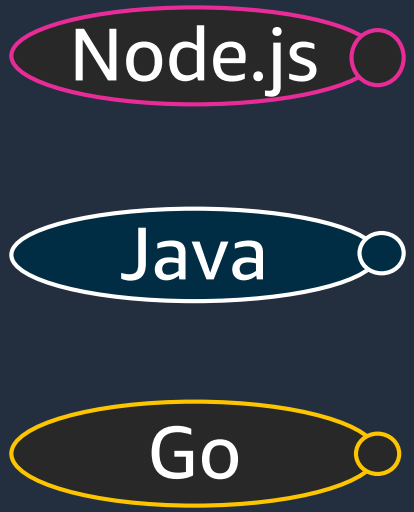
- 使用している言語のライブラリが無い
- 使用している言語のライブラリが保守されない
- 共通ライブラリを入れると依存関係が衝突する
- VMだと動かない、コンテナだと動かない、 XX環境だと動かない

共通ライブラリを開発する側の課題

チームX

- 多数の言語やSDKに習熟する必要があって負担が大きい
- 言語ごと、チームごとに配布方法を考える必要があって負担が大きい
- システムの規模、アプリケーション数に応じて開発リソースがスケールしない

結局、何が課題の原因なのか？



- 通信の処理と、アプリケーションの密結合
- システムロジックと業務ロジックは運用もライフサイクルも違う

通信の処理は共通化が求められるが、
アプリケーションコードの技術に合わせて
実装しなければならないジレンマ

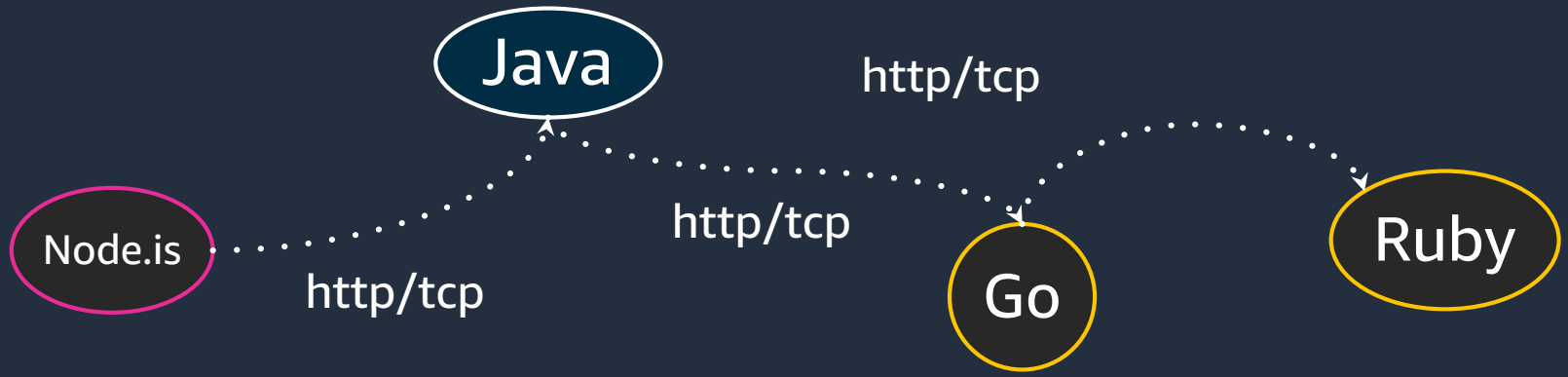
通信の処理を別のプロセスに切りはなすというアイデア

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

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



aws

サービスメッシュの仕組み

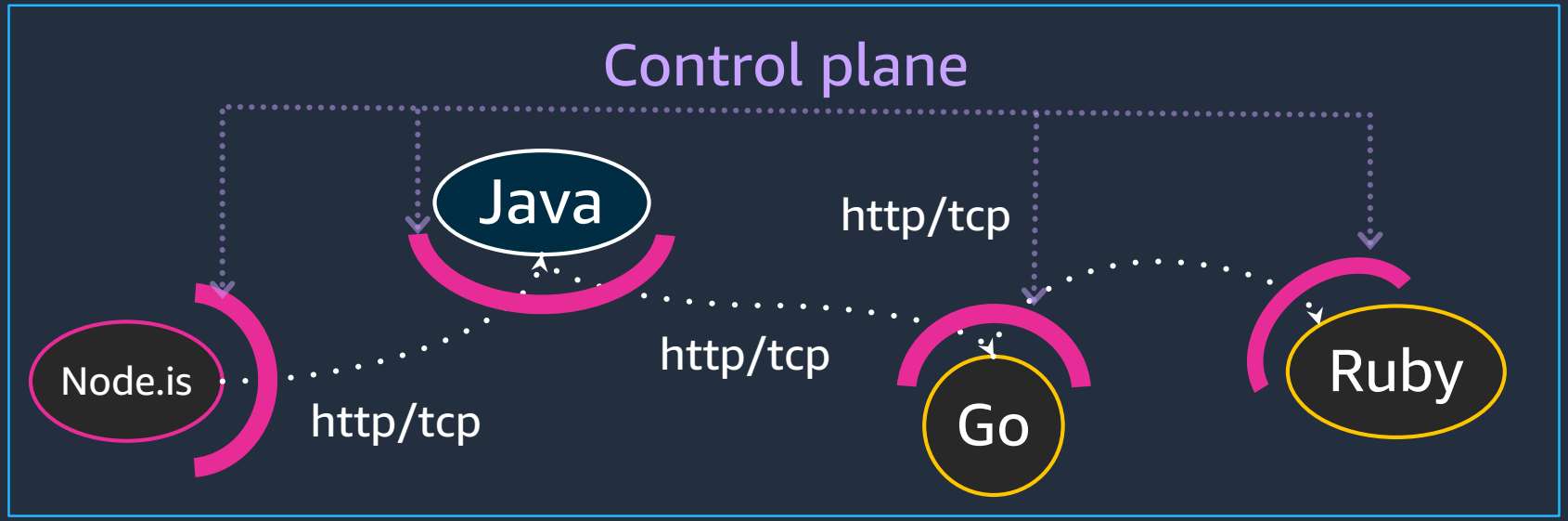
アプリケーションレベルの通信を制御するプロキシを配置する

aws

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、基盤で実施できるようにする

サービスメッシュ基盤

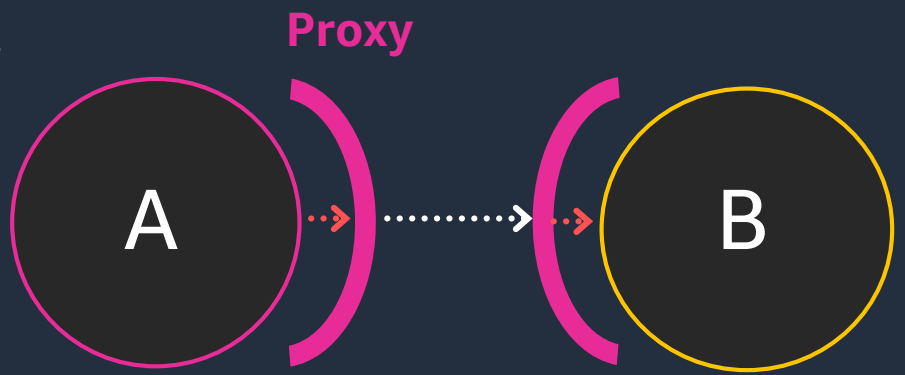


aws

サービスメッシュを実現するために必要な機能

Proxyに、通信制御に関する実装が必要

- 信頼性
  - リトライ
  - タイムアウト
  - 暗号化通信、etc.
- 可観測性
  - メトリクスやトレース、ログの出力



Envoy Proxy

サービスプロキシを開発するOSSプロジェクト

活発なコミュニティと多くのインテグレーション

CNCFのGraduated Project

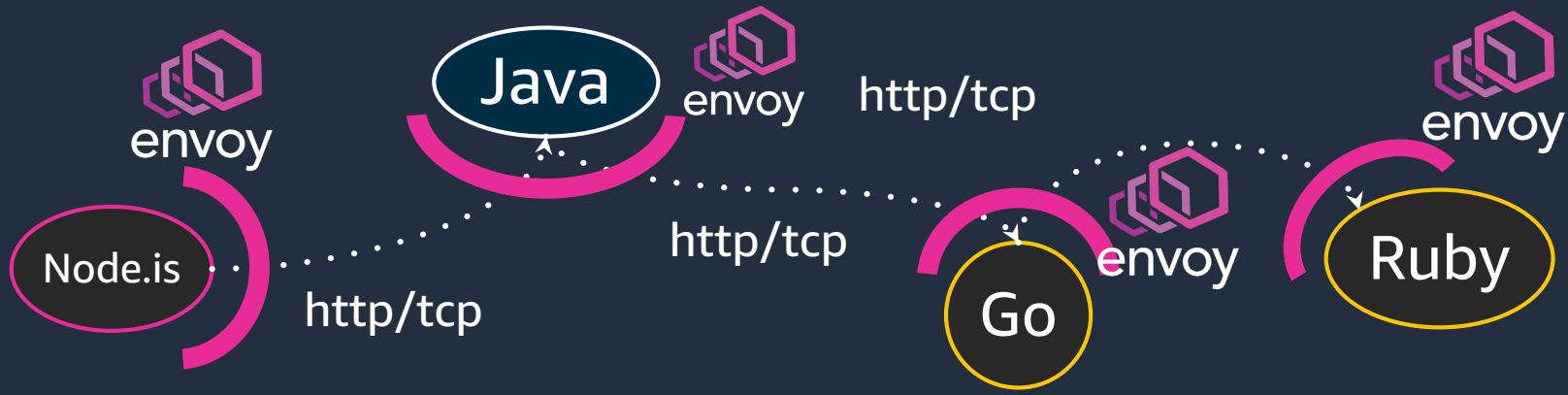
プロダクション環境での利用事例が多い

Lyft社が2016年に開始

aws

Envoyでサービスメッシュを構築する

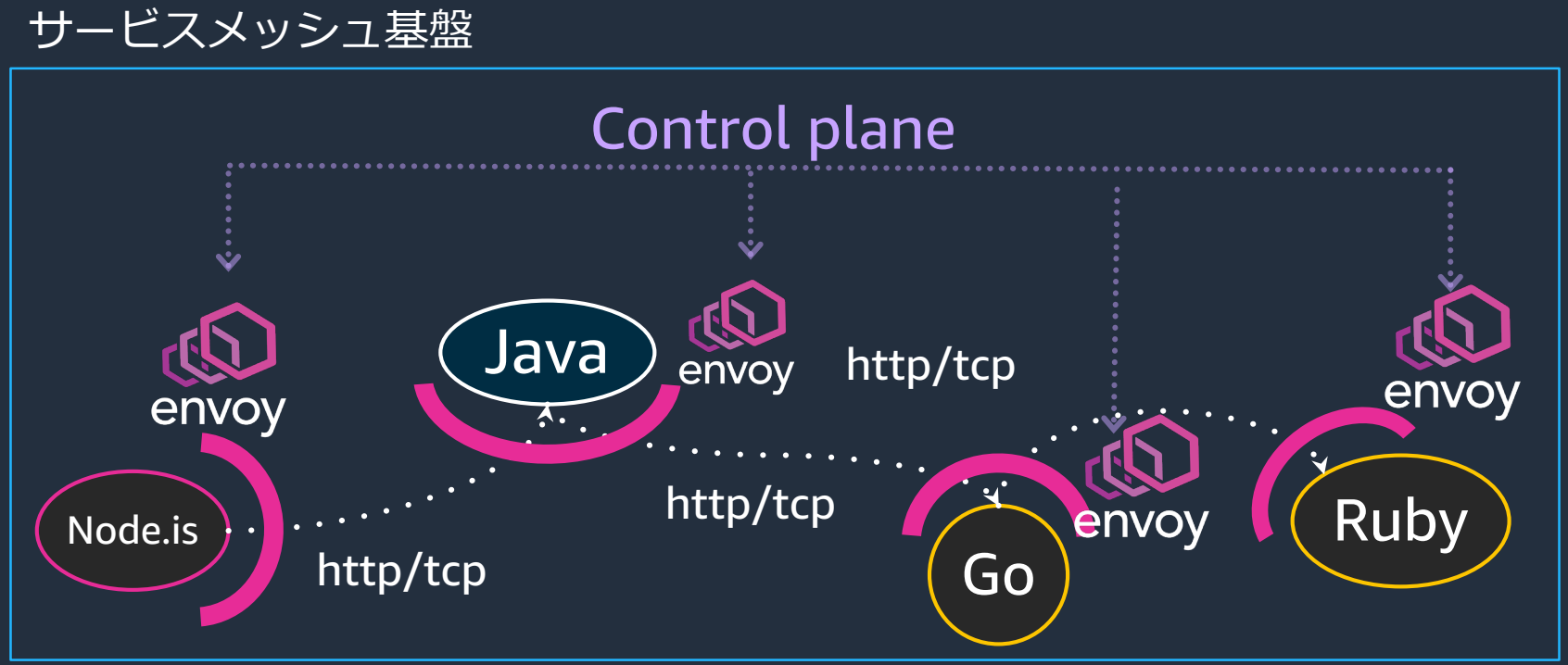
アプリケーションレベルの通信制御を行うプロキシとして、Envoyを利用できる



aws

Envoyでサービスメッシュを構築する

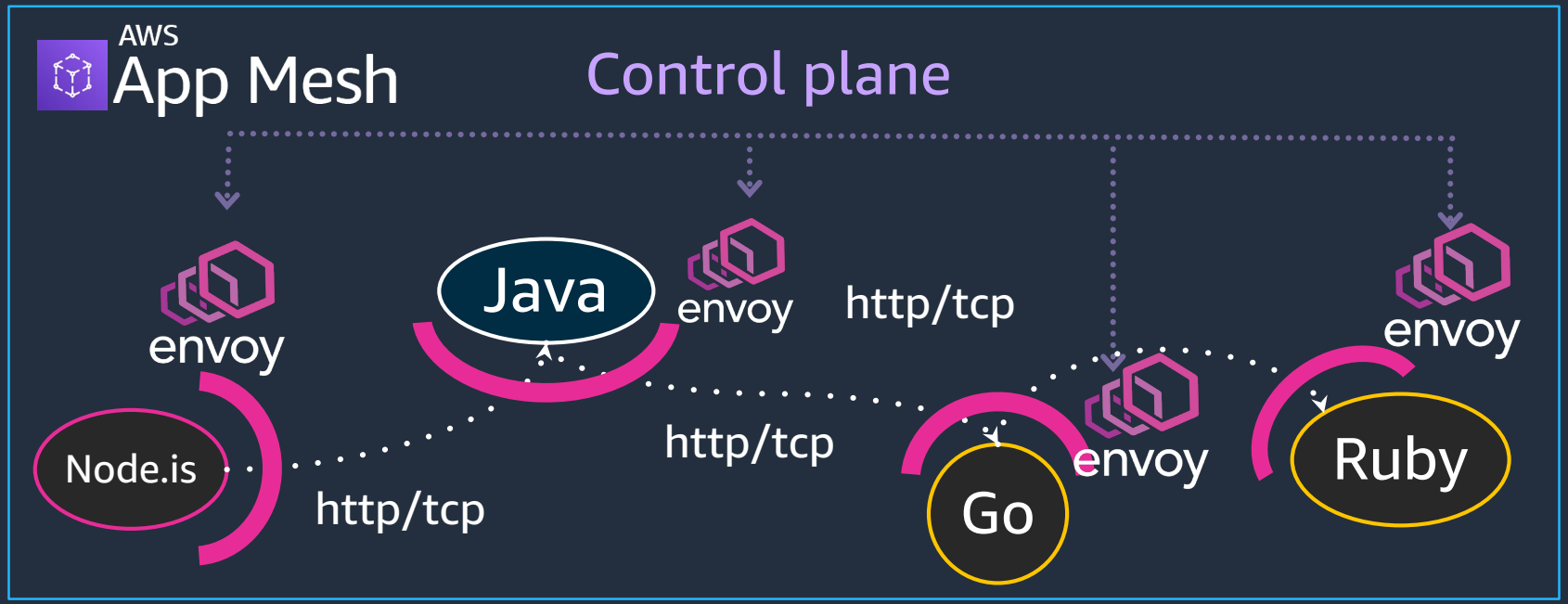
Envoyを管理するコントロールプレーンを導入



aws

AWS App Mesh

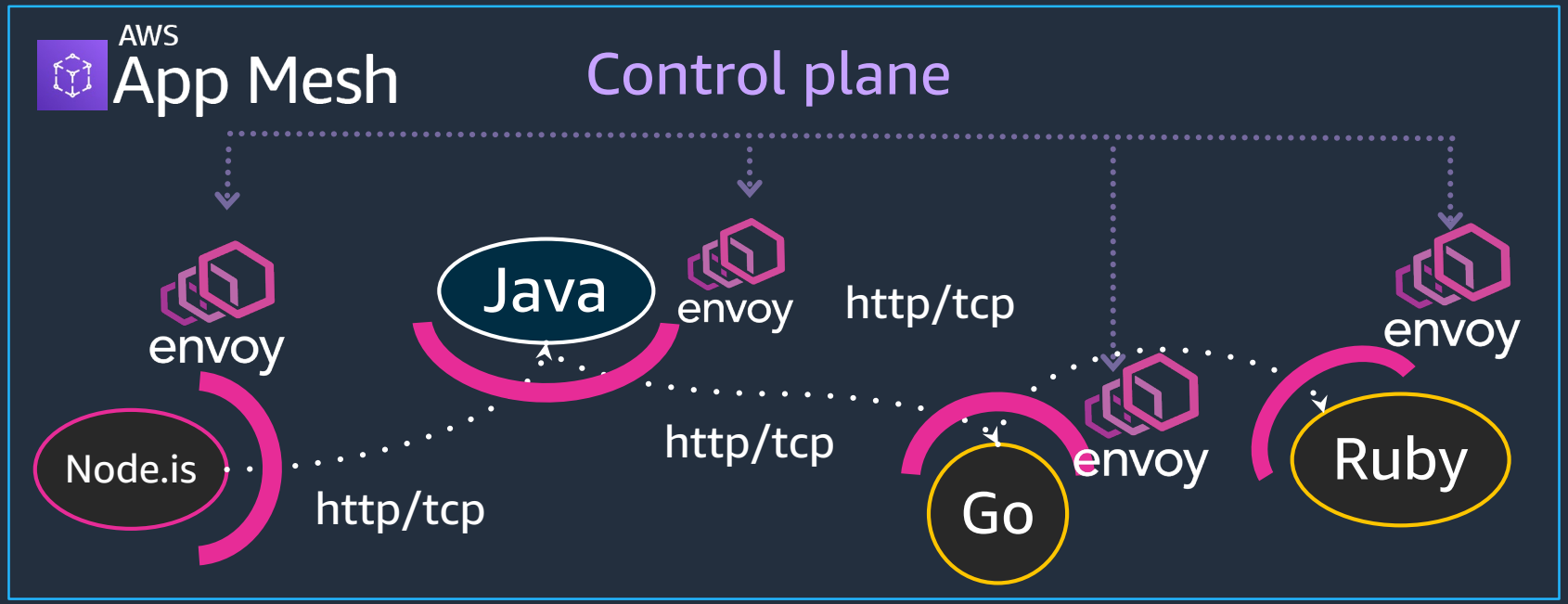
Envoyを管理するコントロールプレーンを提供し、サービスメッシュを実現するマネージドサービス



- サービスメッシュとは何か
- AWS App Mesh
  - 概要
  - 利用方法
  - 機能と活用例
  - ロードマップ
  - 価格体系

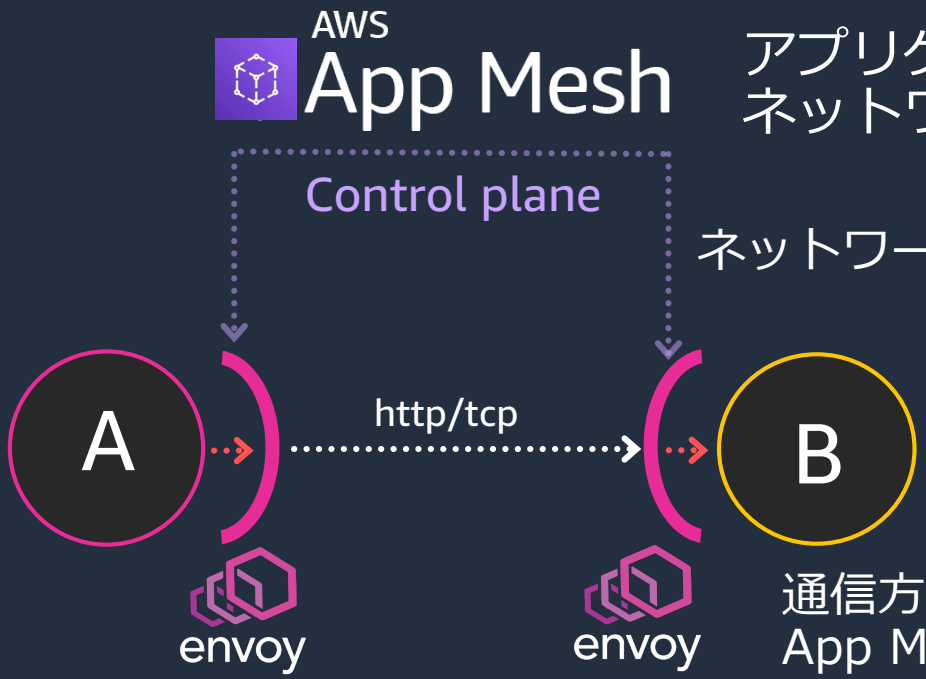
AWS App Meshとは？

サービスメッシュを提供し、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ではなく、App Meshで設定、制御できるようにするマネージドサービス



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

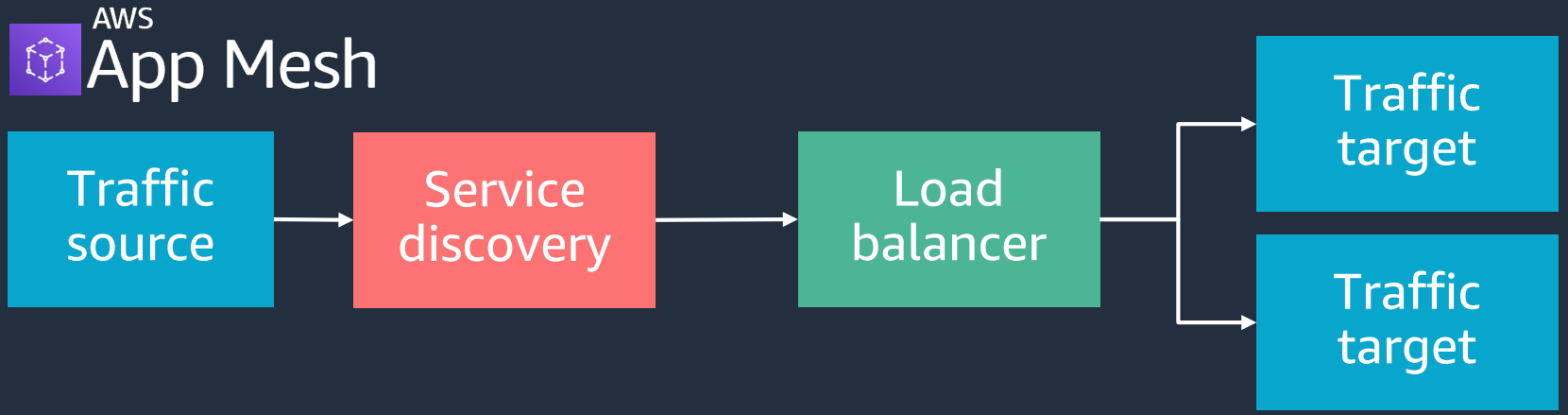


アプリケーション間がどのように通信するか
ネットワークモデルを定義できる

ネットワークモデルをEnvoyの設定に変換して配信

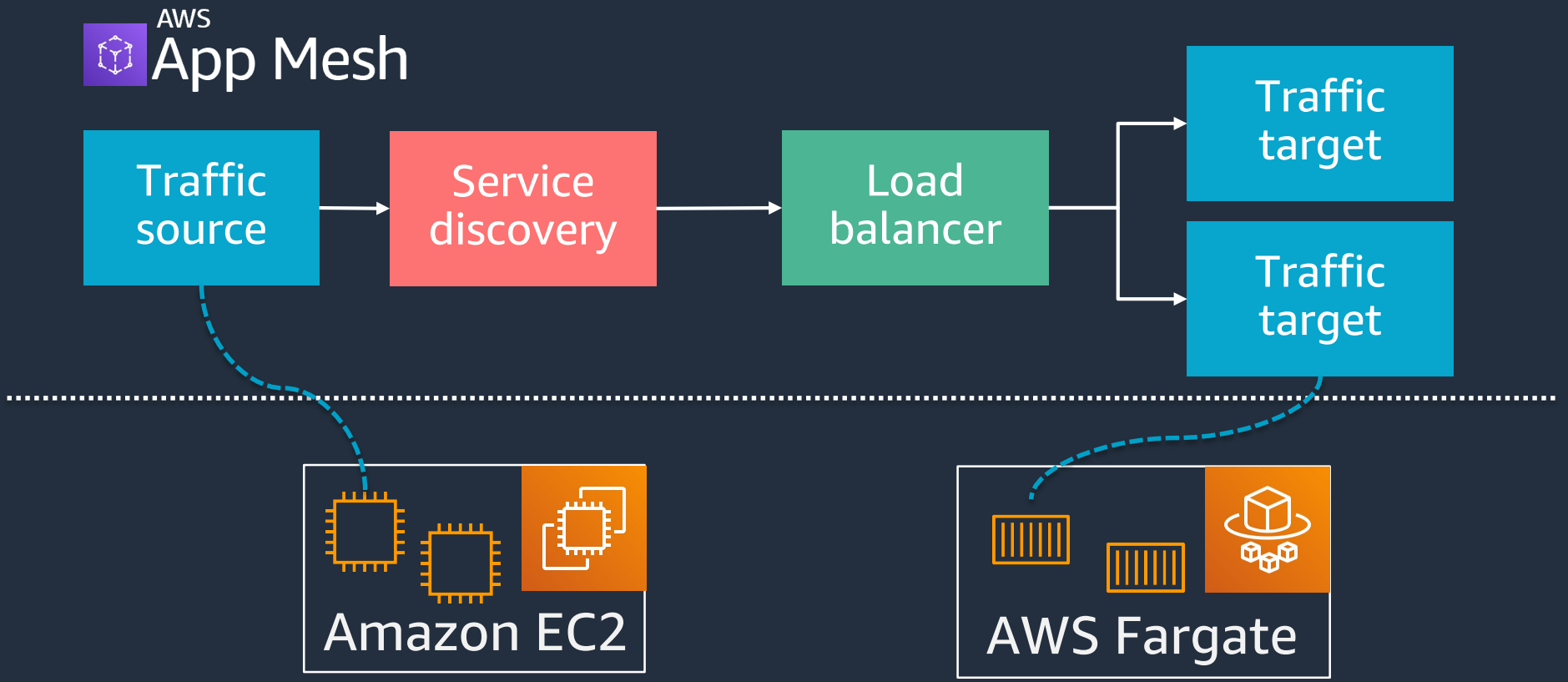
通信方法の詳細は、アプリケーションではなく
App Meshのレイヤーで制御される

App Meshのネットワークモデル



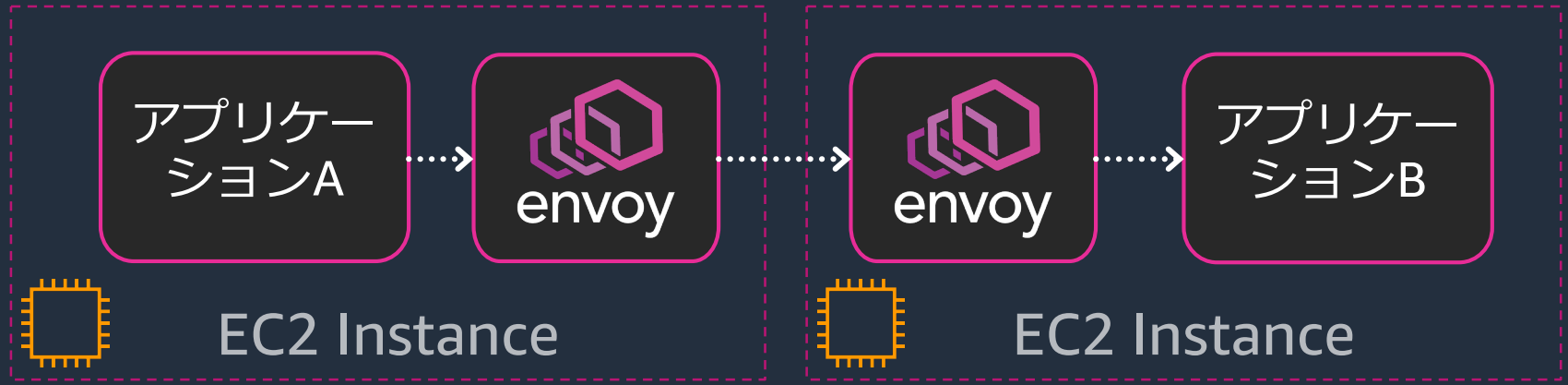
1. 通信元のアプリケーションは、通信先をサービスディスカバリーで見つけてトラフィックを投げる
2. トラフィックがロードバランサーに到達する
3. ロードバランサーは複数の通信先にトラフィックを振り分ける

App Meshのモデルと実際のアプリケーションを紐付け

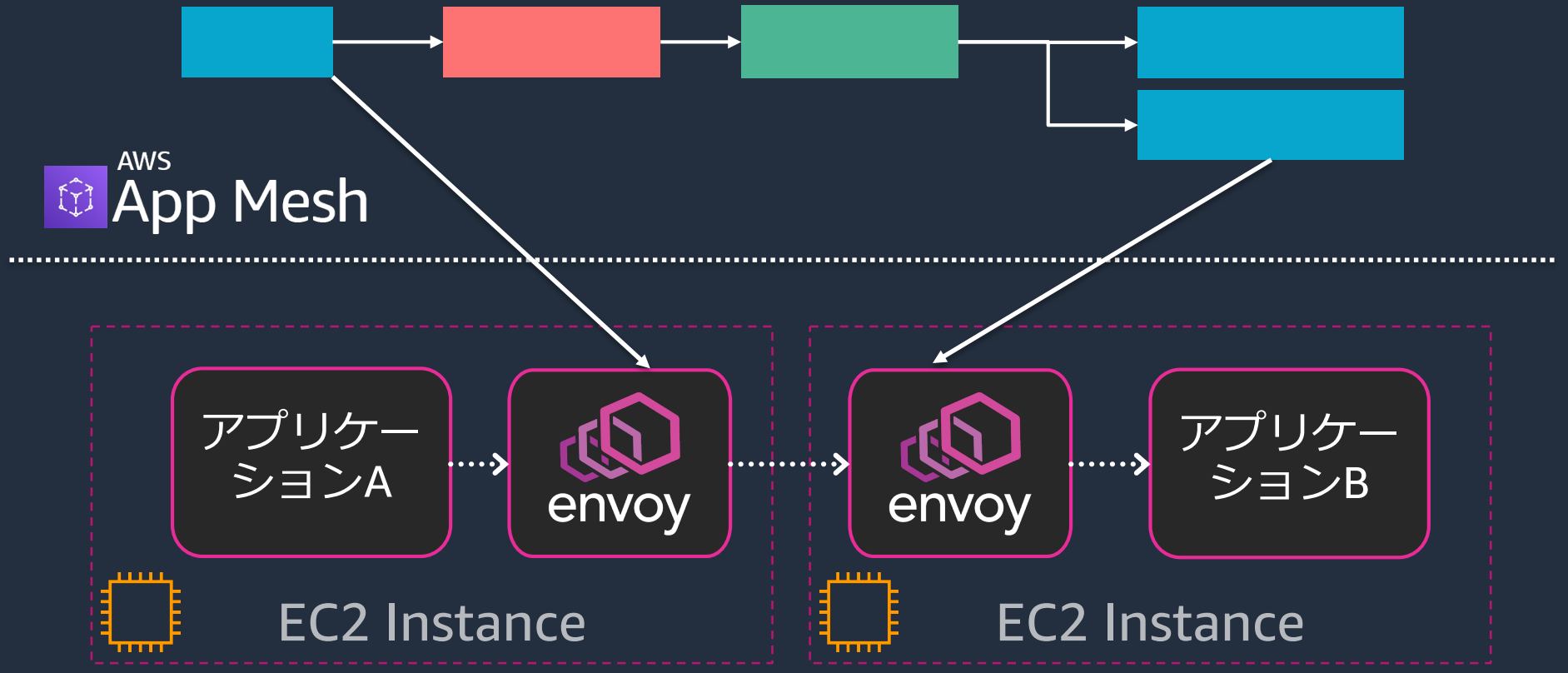


App Meshの動作イメージ / 前提

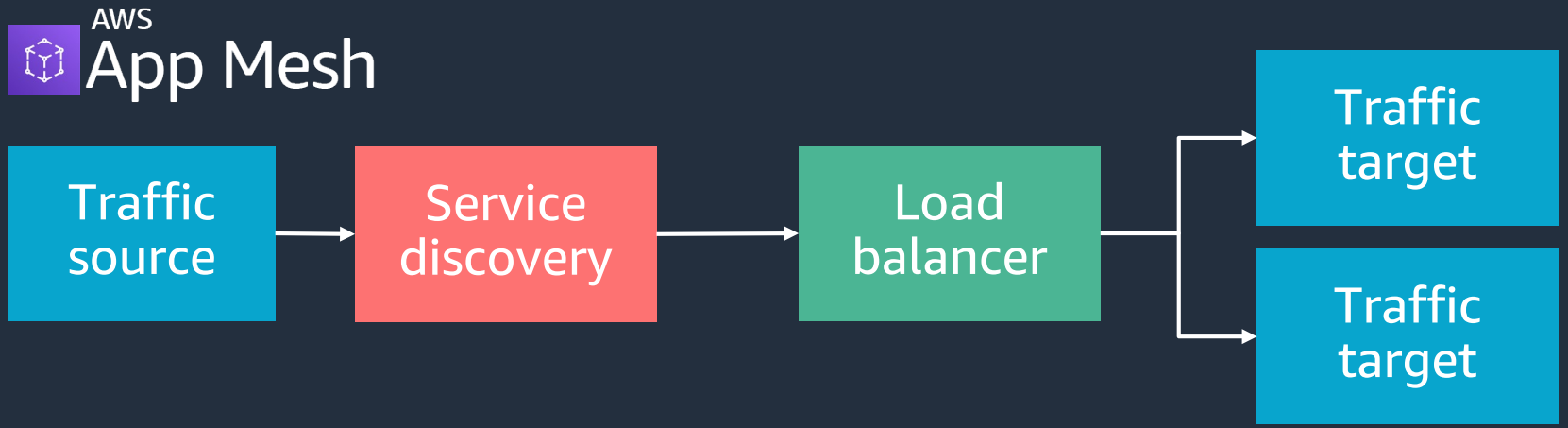
App Meshの動作イメージ / Envoy Proxyの導入



App Meshの動作イメージ / App Meshのモデルを適用

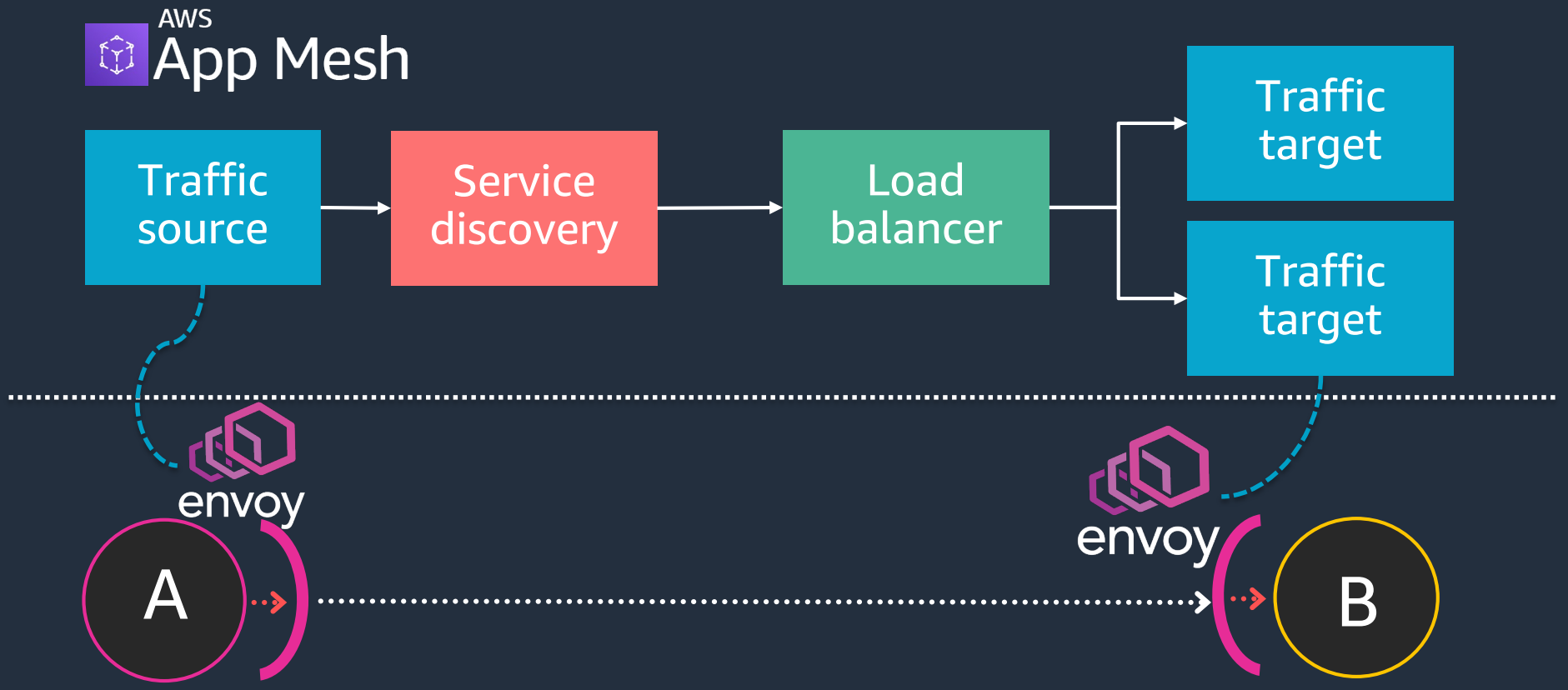


App Meshのモデル上で設定できる通信制御の例

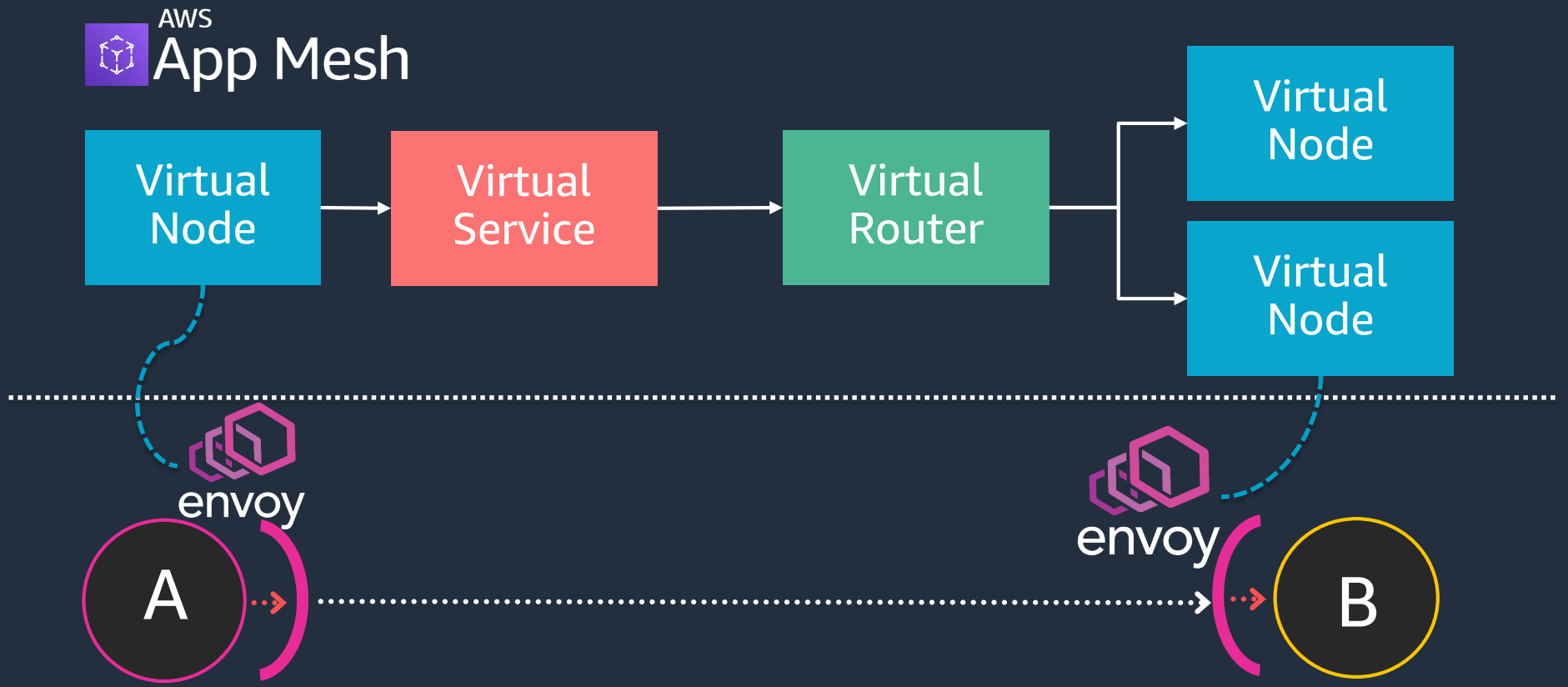


- リトライやタイムアウトを設定
- 証明書を利用したアプリケーション間の暗号化通信
- 比重を設定してトラフィックを振り分け => Canary
- etc.

App Meshにおけるトップレベルの概念



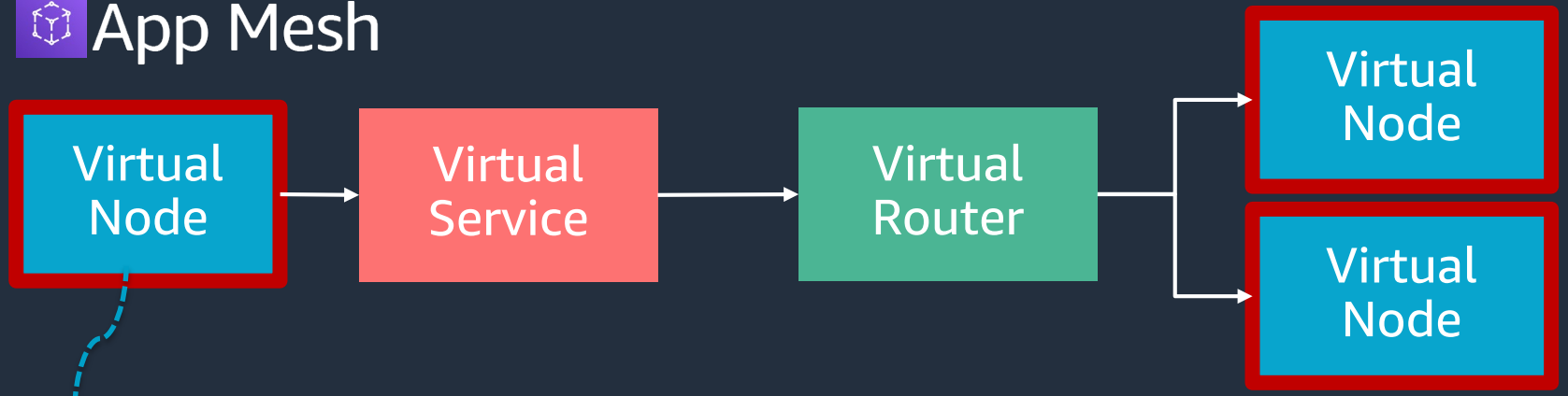
App Meshにおけるトップレベルの概念



aws

Virtual Node

AWS App Mesh



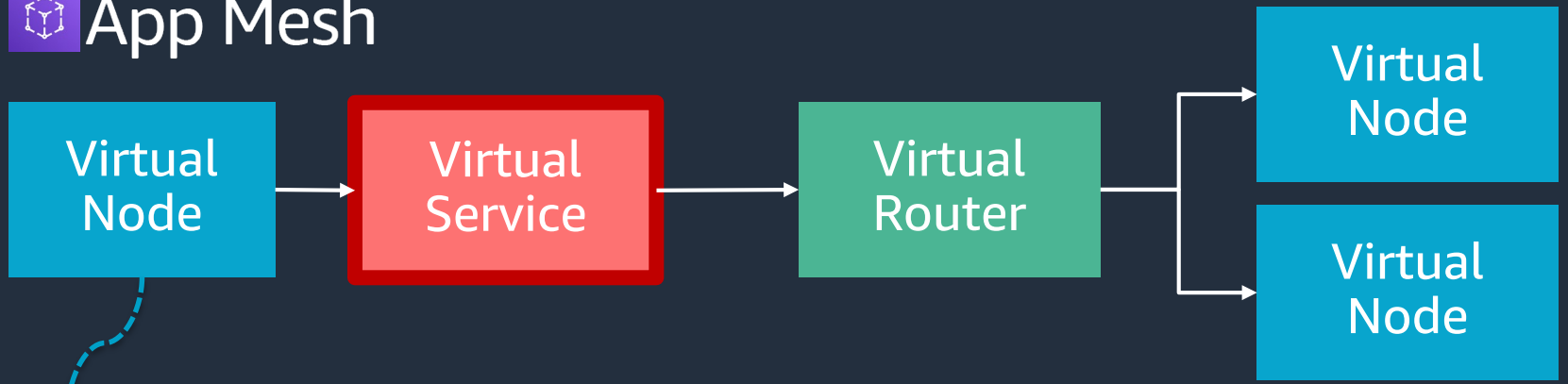
アプリケーションへの論理的なポインター

Envoyの環境変数でVirtual Node名を設定できる

aws

Virtual Service

AWS App Mesh

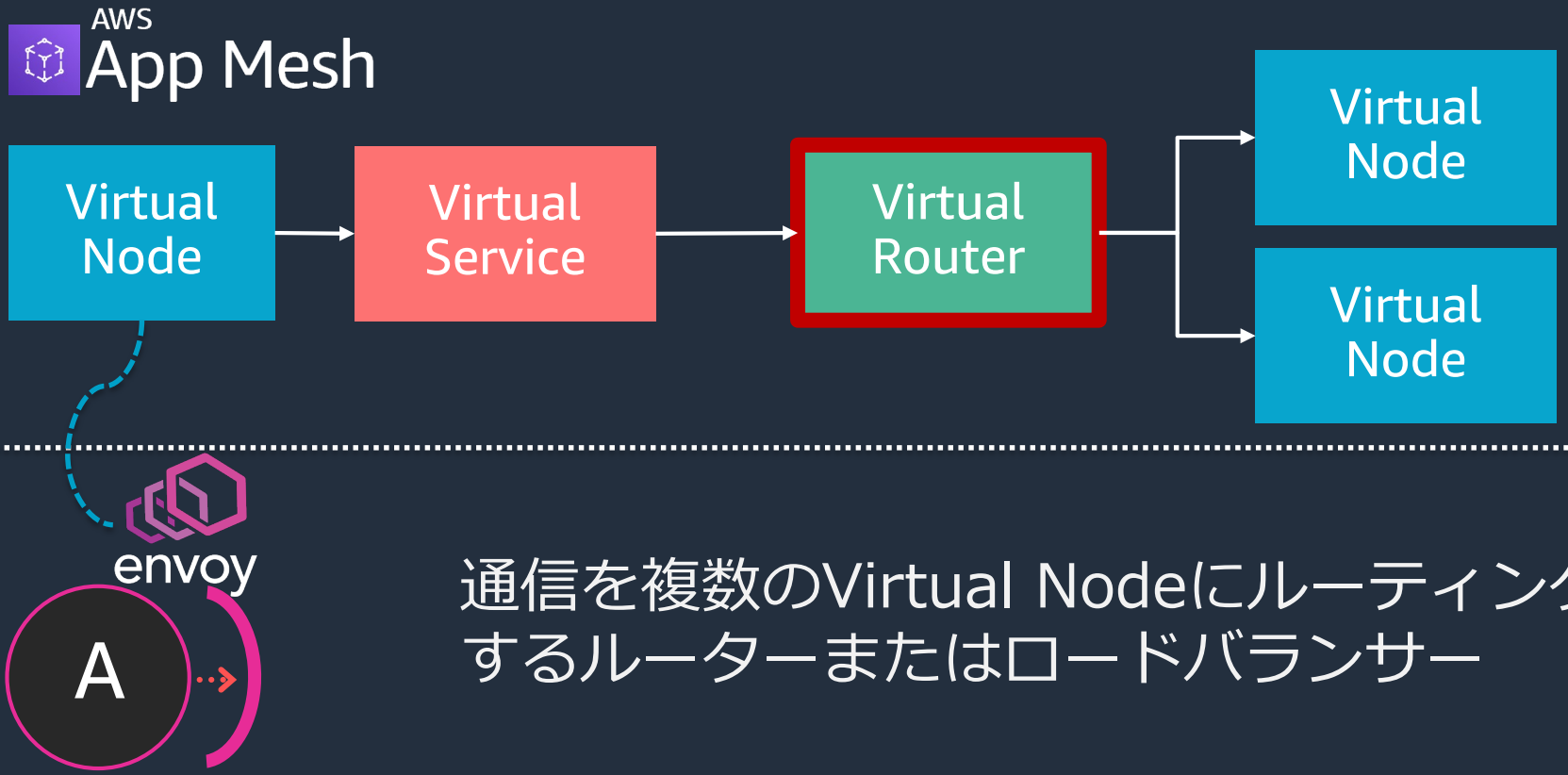


アプリケーションの通信先を表す

サービスディスカバリー上のサービスで、
ここにリクエストを投げると後続のVirtual Nodeに
ルーティングされる

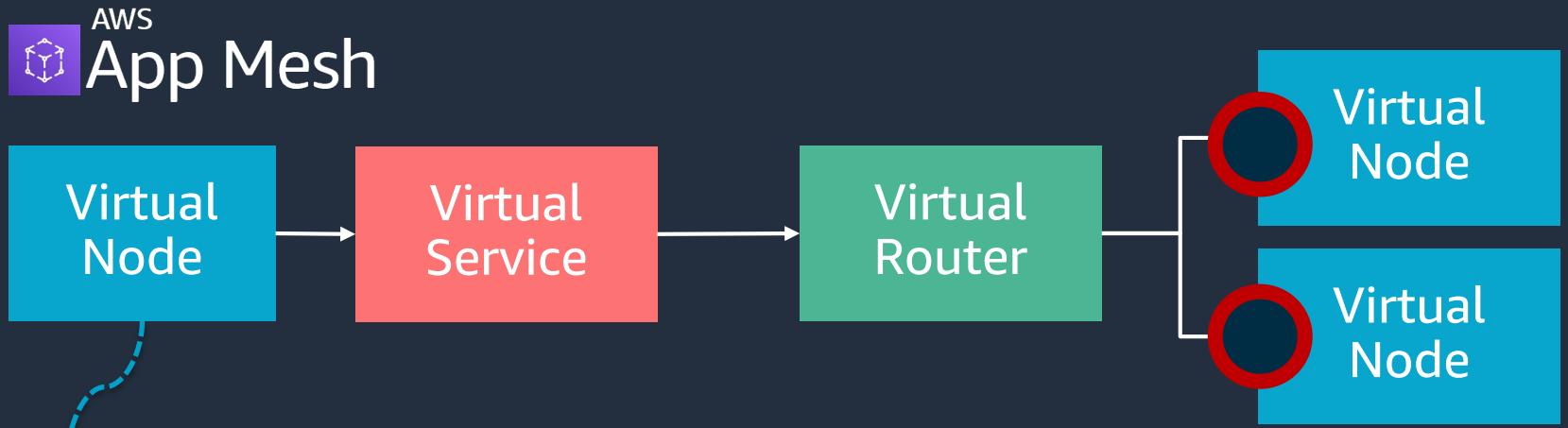
aws

Virtual Router



通信を複数のVirtual Nodeにルーティングするルーターまたはロードバランサー

Listener

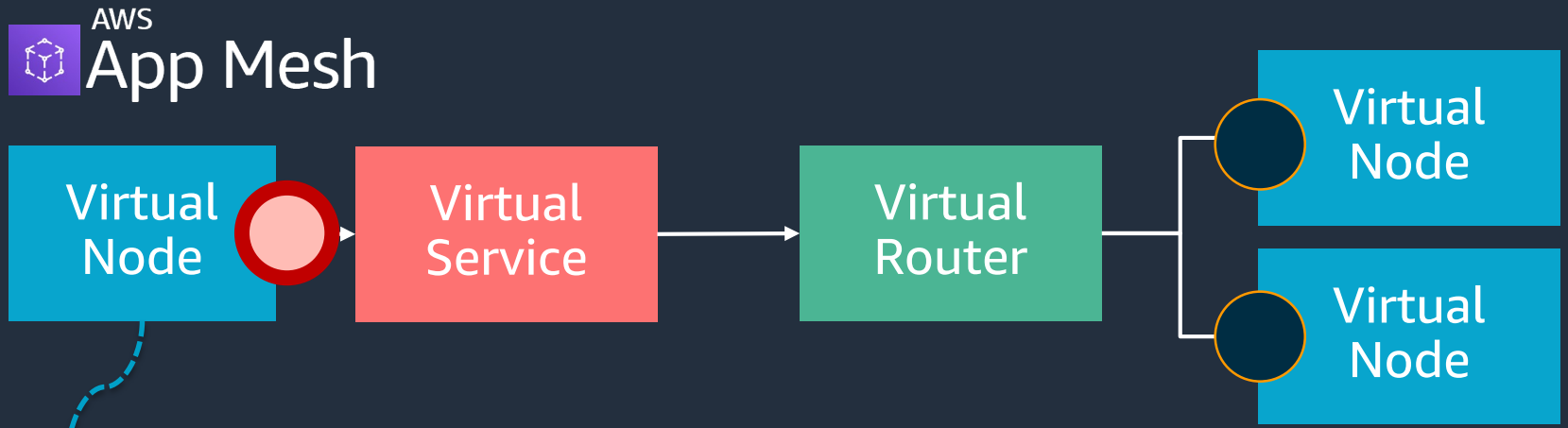


Virtual Nodeで、どのようにトラフィックを受け付けるか

- プロトコル(http、http2、grpc、tcp)
- ポート番号
- 内部的にはEnvoyがこの設定を取得してリクエストを受け付けるプロトコルやポートを設定

aws

Backend

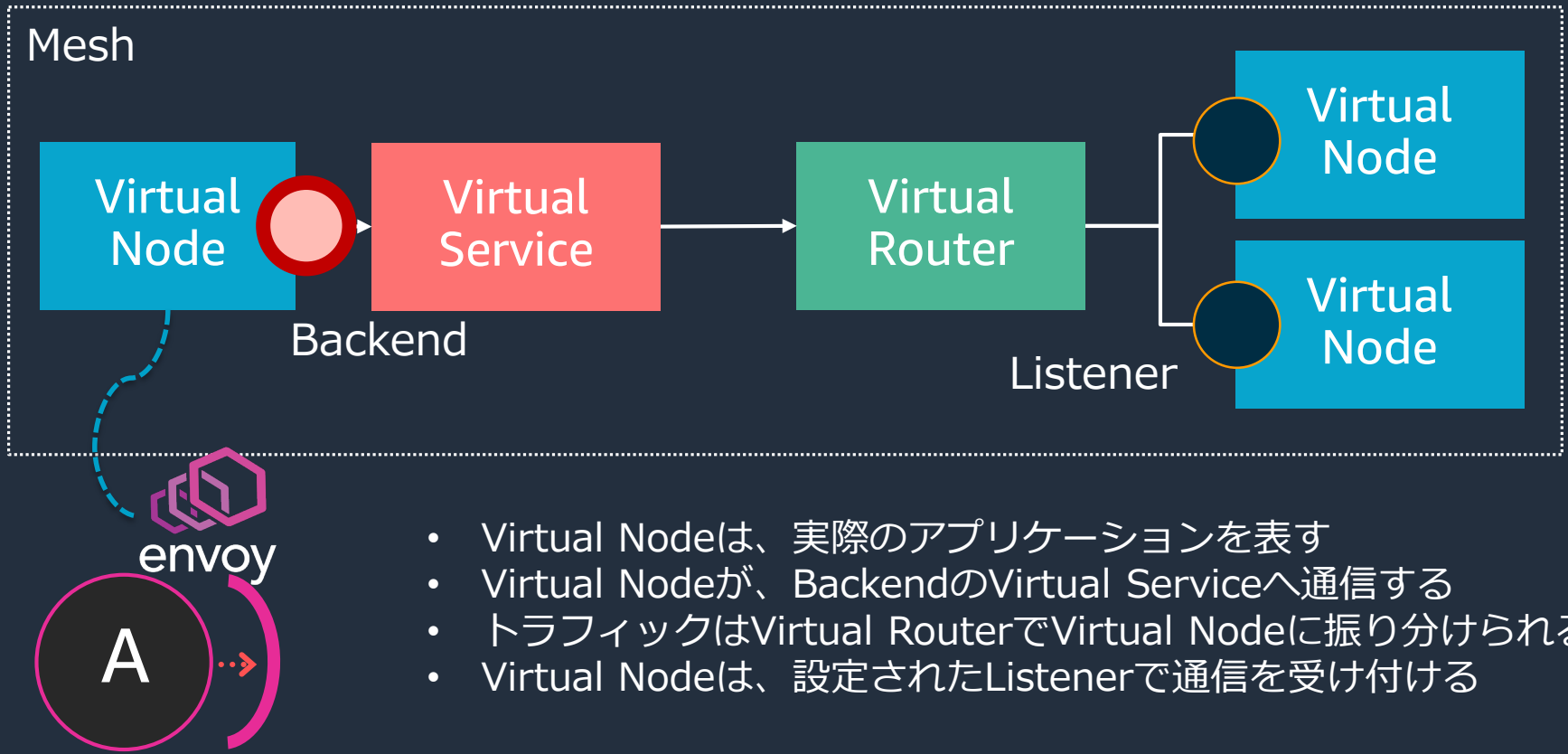


送信先として想定されるVirtual Service

- デフォルトではBackendとして設定されていない宛先には通信できない(※)
- Backendでない宛先に送信できるよう設定で変更できる

※ AWS APIを呼び出す目的で*.amazonaws.comにはアクセス可能

Meshの構築



- Virtual Nodeは、実際のアプリケーションを表す
- Virtual Nodeが、BackendのVirtual Serviceへ通信する
- トラフィックはVirtual RouterでVirtual Nodeに振り分けられる
- Virtual Nodeは、設定されたListenerで通信を受け付ける

aws

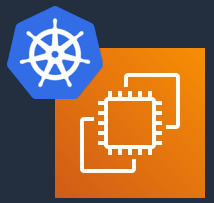
- サービスメッシュとは何か
- AWS App Mesh
    - 概要
    - 利用方法
    - 機能と活用例
    - ロードマップ
    - 価格体系

利用できるインフラストラクチャー

Amazon ECS

Amazon EC2

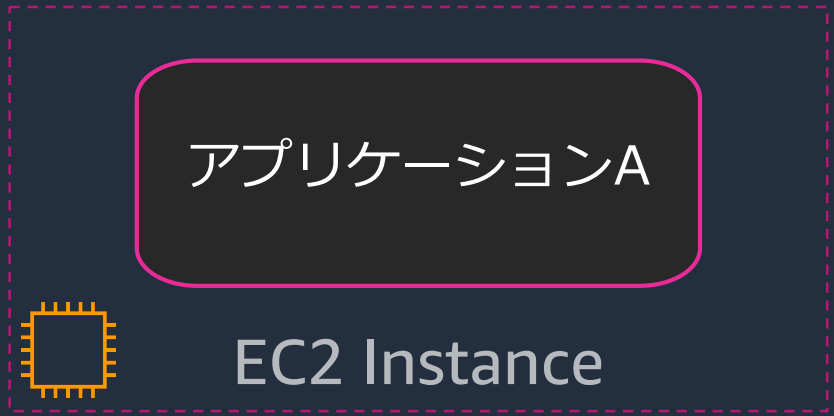
AWS Fargate



Kubernetes on EC2

Amazon EKS

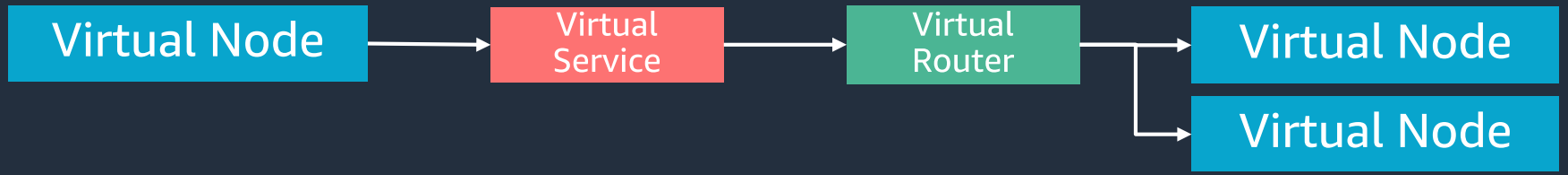
Amazon EC2での利用イメージ

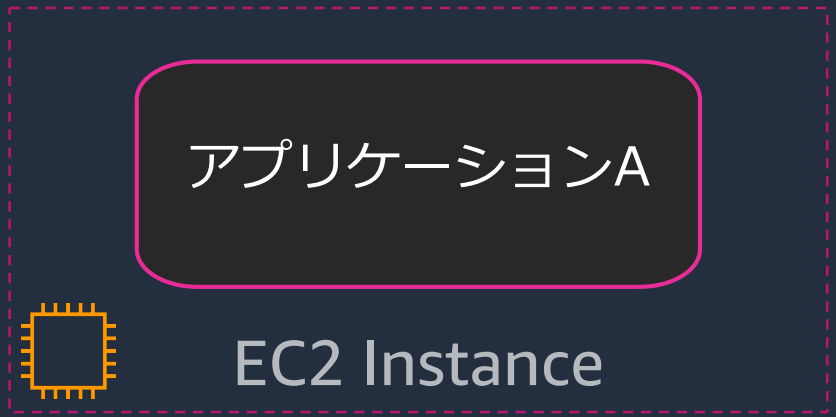


Amazon EC2での利用イメージ

Management ConsoleやCLIでMeshを作成する

AWS App Mesh



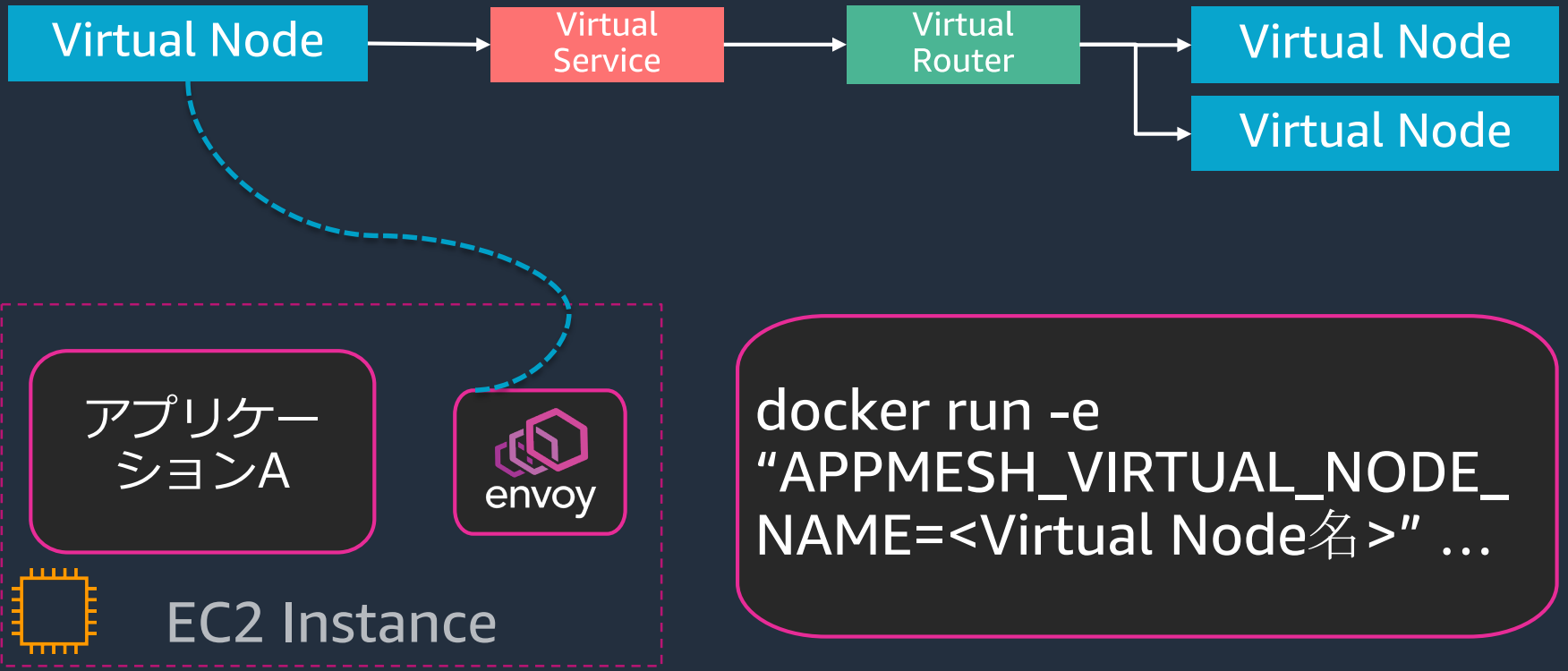


aws

Amazon EC2での利用イメージ

Envoyを導入

AWS App Mesh

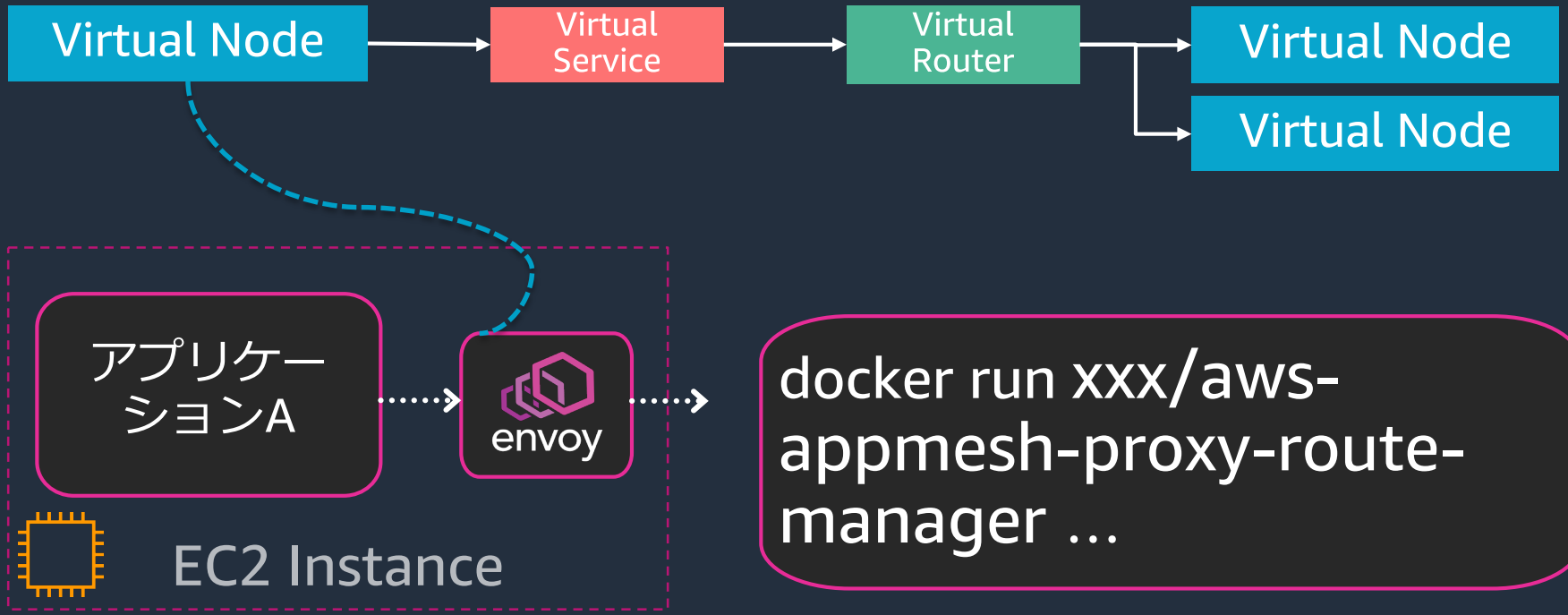


aws

Amazon EC2での利用イメージ

ネットワーク管理イメージを導入

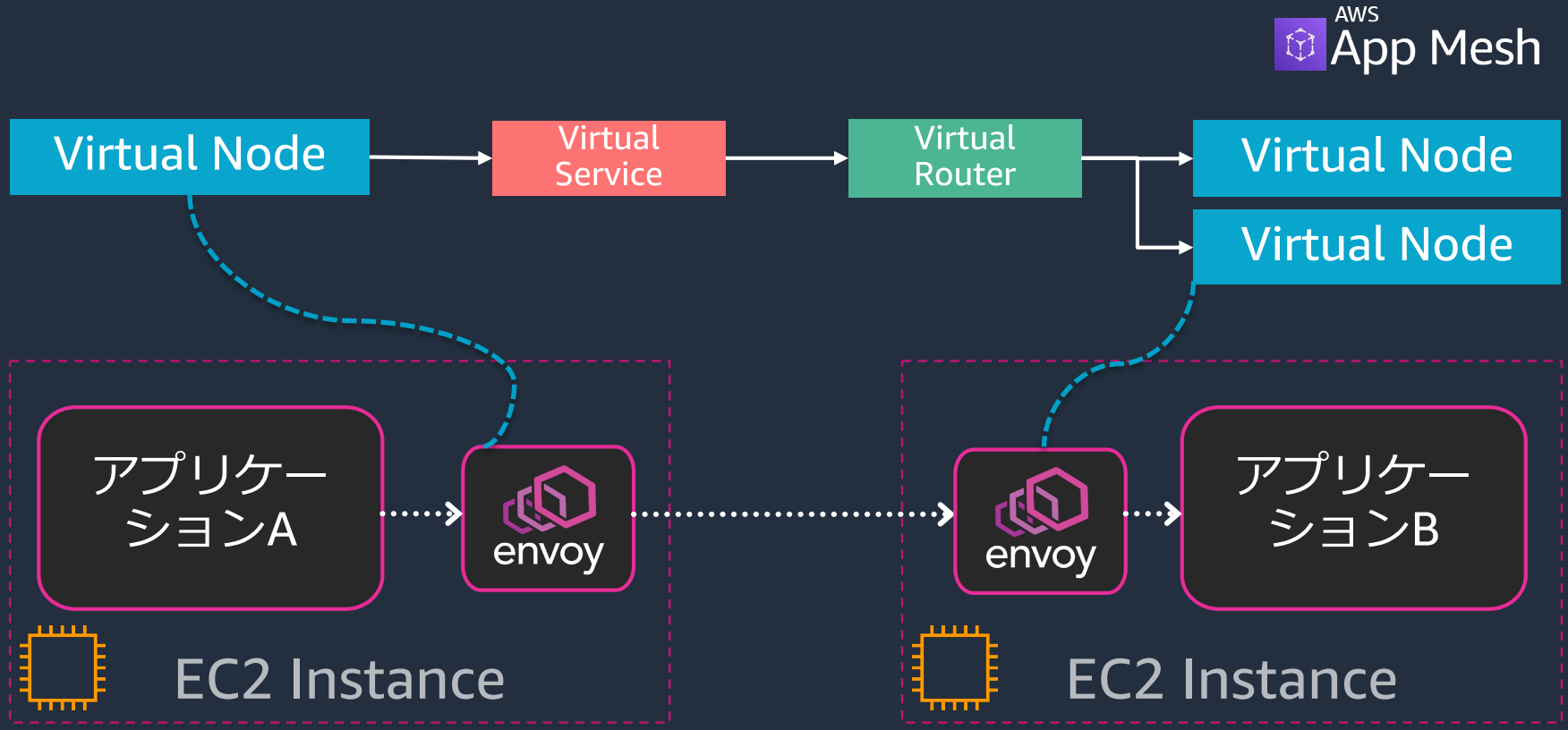
AWS App Mesh



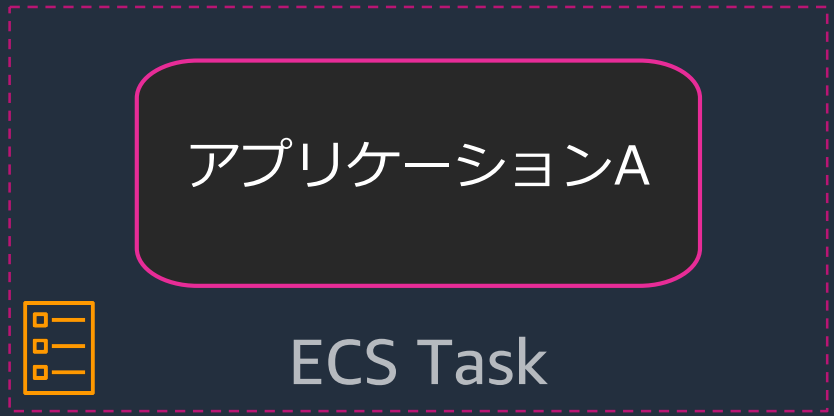
※ iptablesのルールを管理して通信をenvoy経由にする

aws

Amazon EC2での利用イメージ



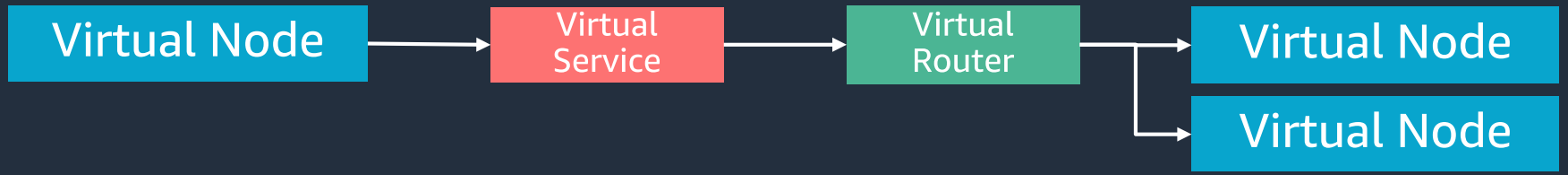
Amazon ECS / AWS Fargateでの利用イメージ

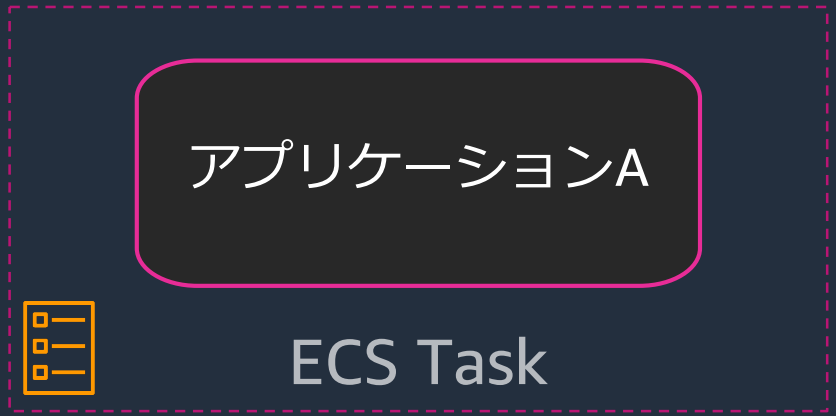


Amazon ECS / AWS Fargateでの利用イメージ

Management ConsoleやCLIでMeshを作成する

AWS App Mesh



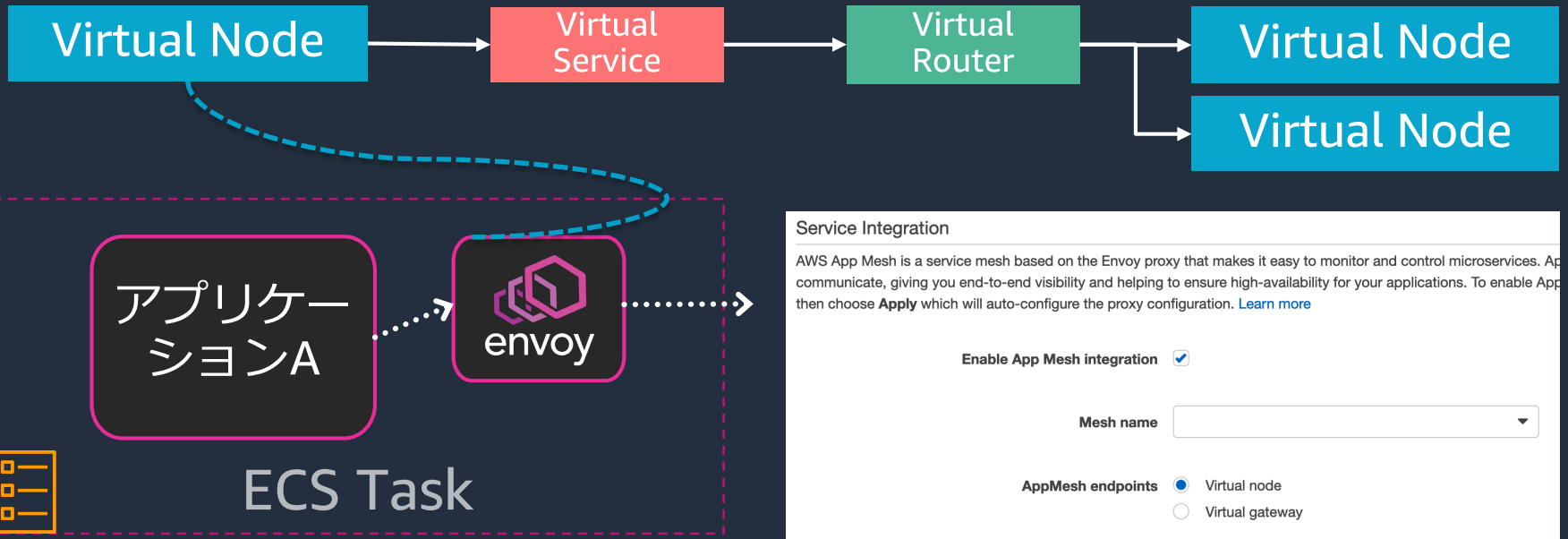


aws

Amazon ECS / AWS Fargateでの利用イメージ

Management ConsoleやCLIでMesh統合を有効にする

AWS App Mesh



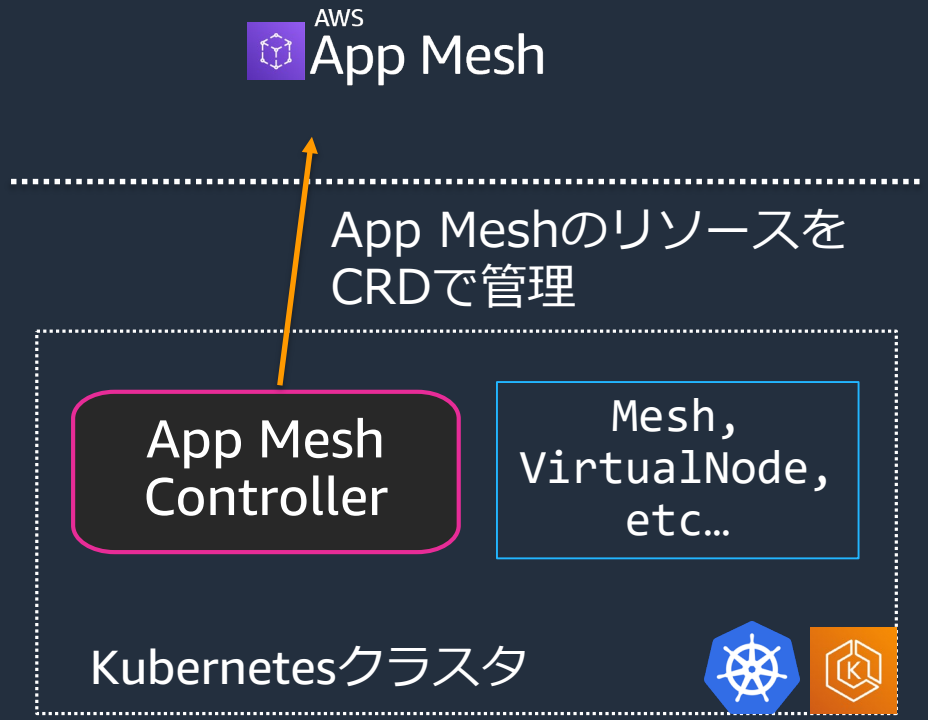
ECSタスク定義が更新され、Envoyが導入される

aws

Amazon EKS / Fargate for EKS / Kubernetes on EC2での利用イメージ

AWS App Mesh Controller For K8s

- 2020年6月にGA
- 主な機能
  - App Meshのリソースを Kubernetes上で管理
  - PodをApp Meshを利用するよう更新
- GA前は、上記二つがそれぞれ別のコントローラーで提供
  - App Mesh Controller
  - App Mesh Inject



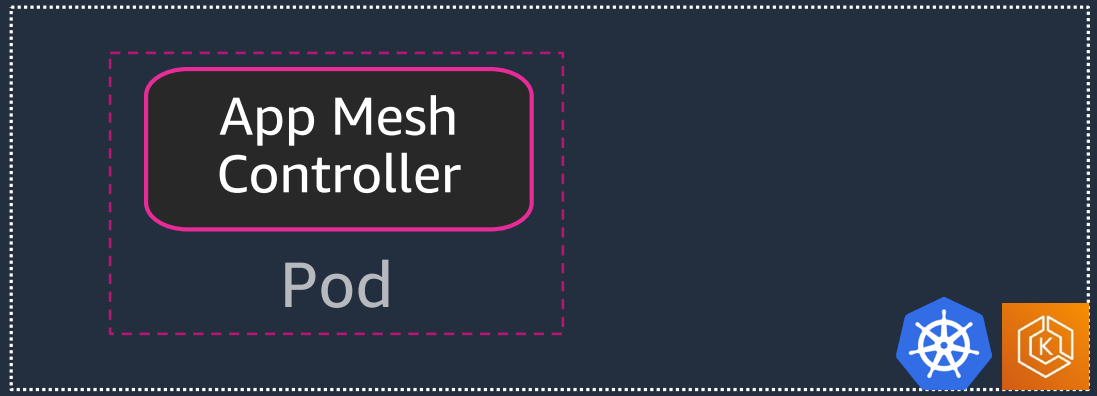
https://github.com/aws/aws-app-mesh-controller-for-k8s

Amazon EKS / Fargate for EKS / Kubernetes on EC2での利用イメージ

AWS App Mesh Controller For K8sを導入

```
$ helm repo add eks https://aws.github.io/eks-charts
$ kubectl apply -k github.com/aws/eks-charts/stable/appmesh-
controller//crds?ref=master

$ kubectl create ns appmesh-system
$ helm upgrade -i appmesh-controller eks/appmesh-controller -n appmesh-system
```



aws

Amazon EKS / Fargate for EKS / Kubernetes on EC2での利用イメージ

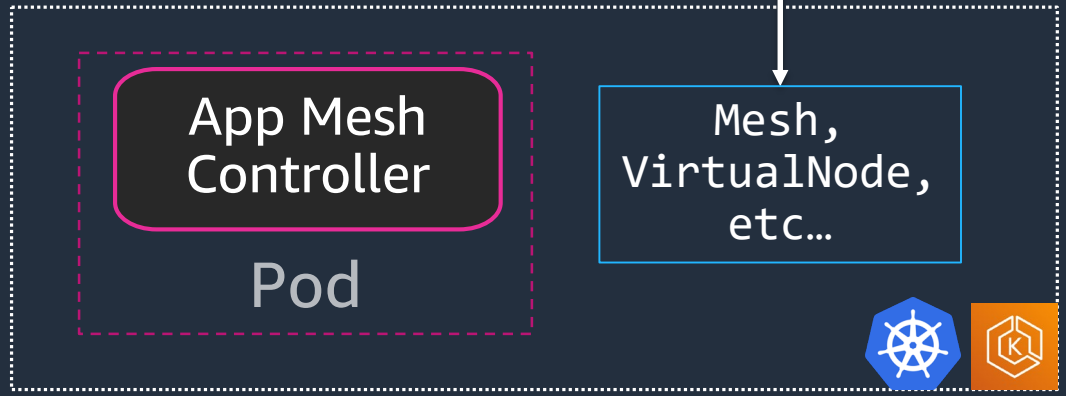
App Mesh用のCustom Resourceをインストールする

```
apiVersion: appmesh.k8s.aws/v1beta2
kind: VirtualNode
metadata:
  name: my-service-a
```

kubectl apply -f vn-a.yaml

App Mesh ControllerのCRD

- Mesh
- VirtualNode
- VirtualService
- VirtualRouter

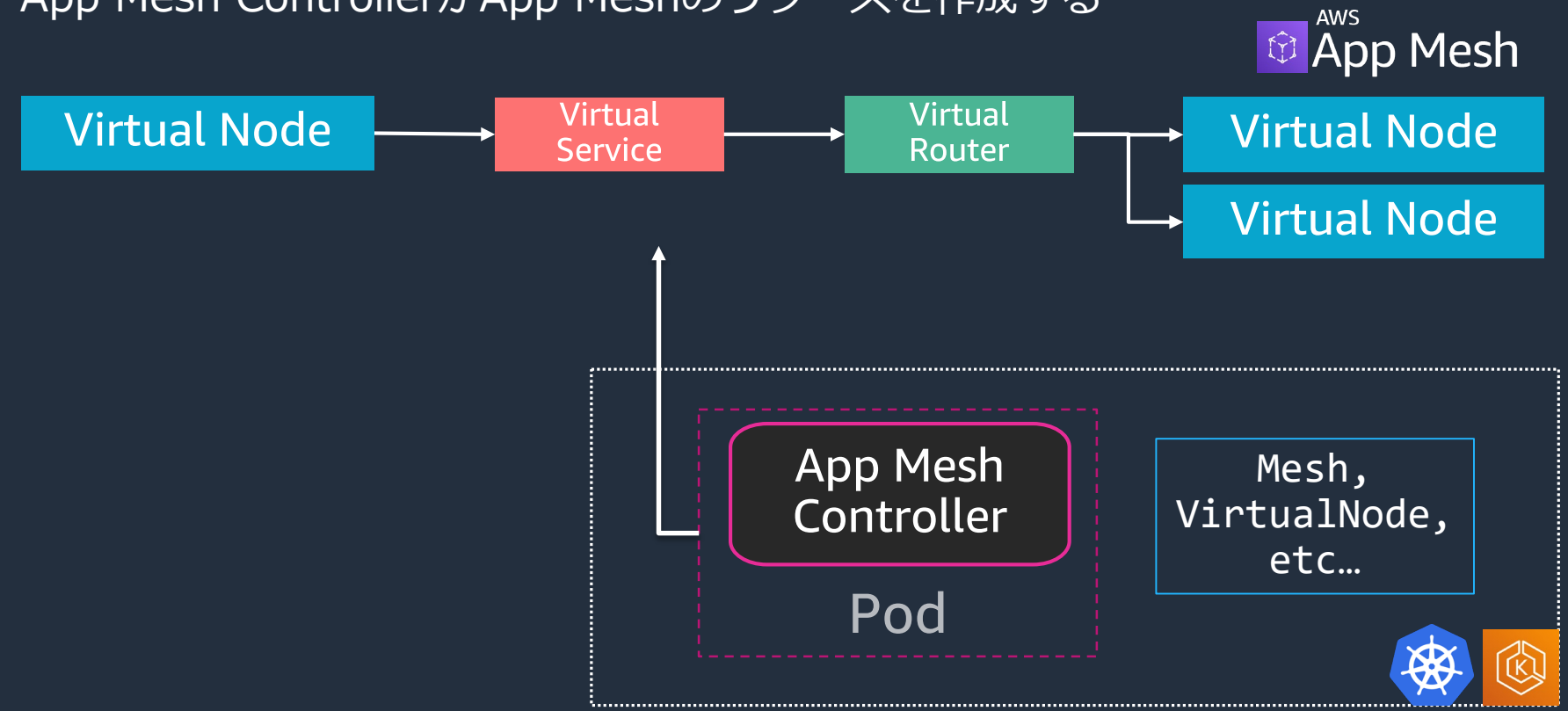


https://github.com/aws/aws-app-mesh-controller-for-k8s/blob/master/docs/reference/api_spec.md

aws

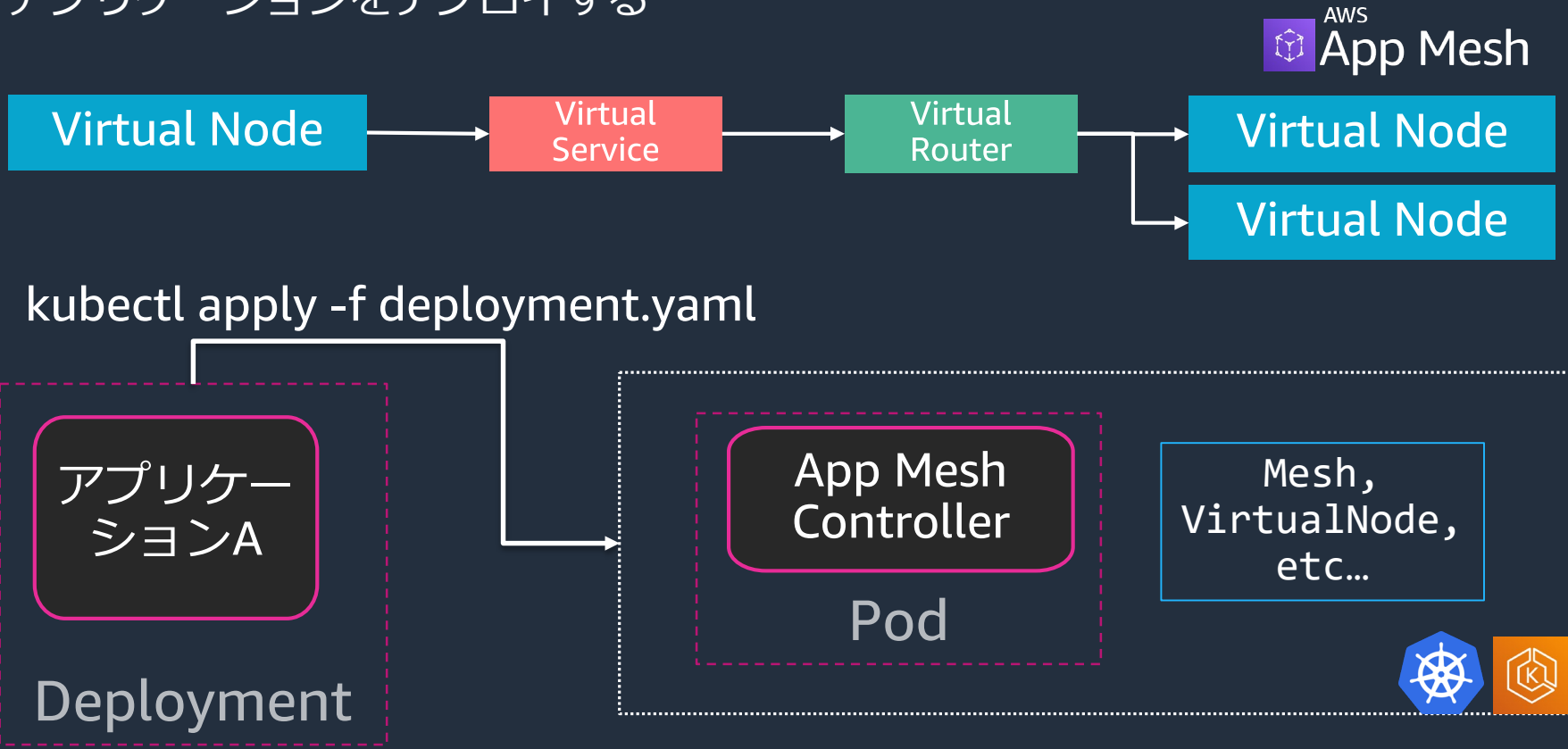
Amazon EKS / Fargate for EKS / Kubernetes on EC2での利用イメージ

App Mesh ControllerがApp Meshのリソースを作成する



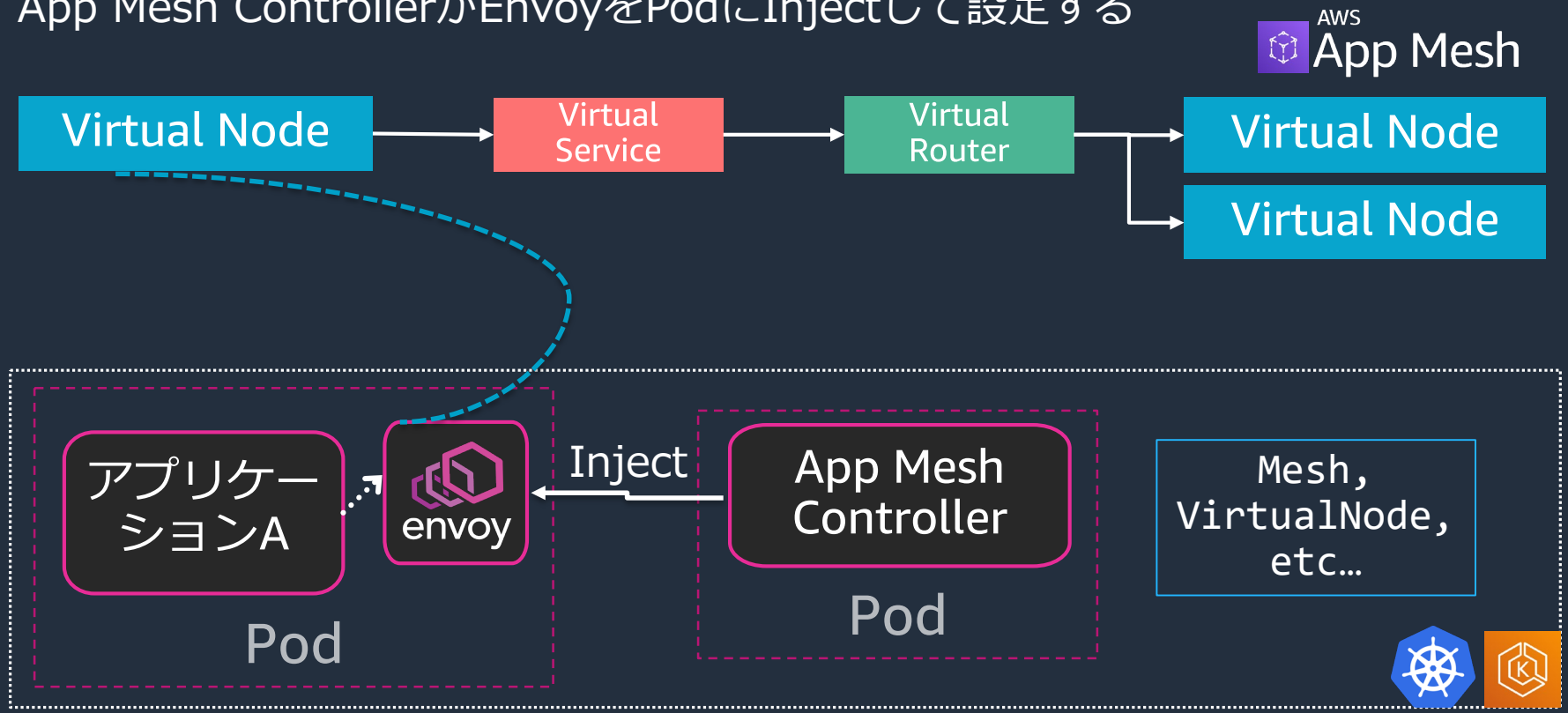
Amazon EKS / Fargate for EKS / Kubernetes on EC2での利用イメージ

アプリケーションをデプロイする

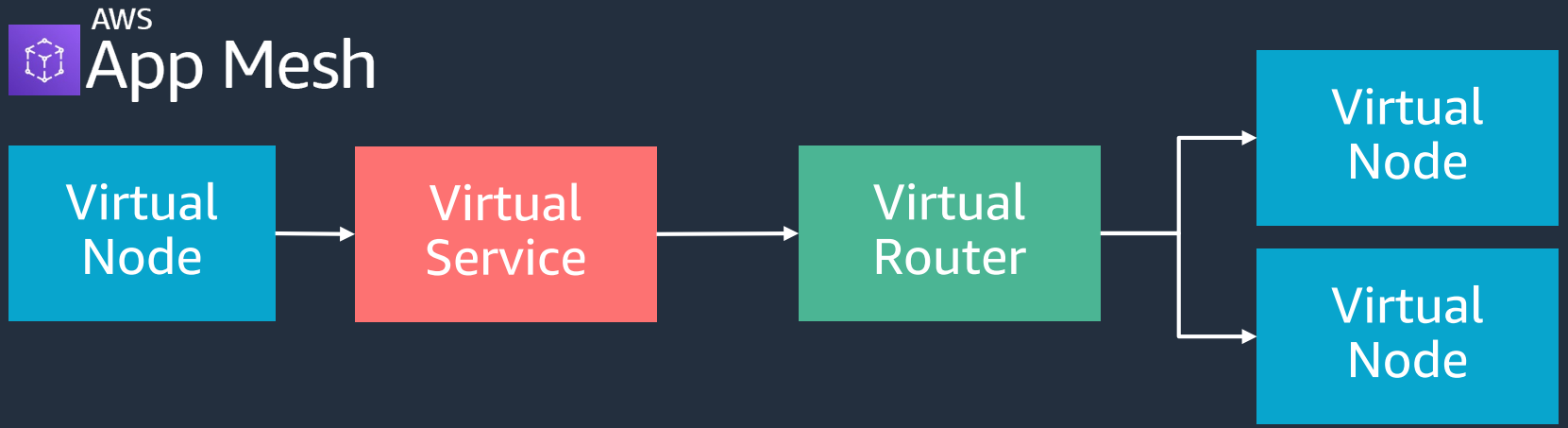


Amazon EKS / Fargate for EKS / Kubernetes on EC2での利用イメージ

App Mesh ControllerがEnvoyをPodにInjectして設定する



様々なインフラストラクチャーを一つのMeshで管理できる



Amazon EC2

Amazon ECS

Amazon EKS

AWS Fargate

Kubernetes on EC2

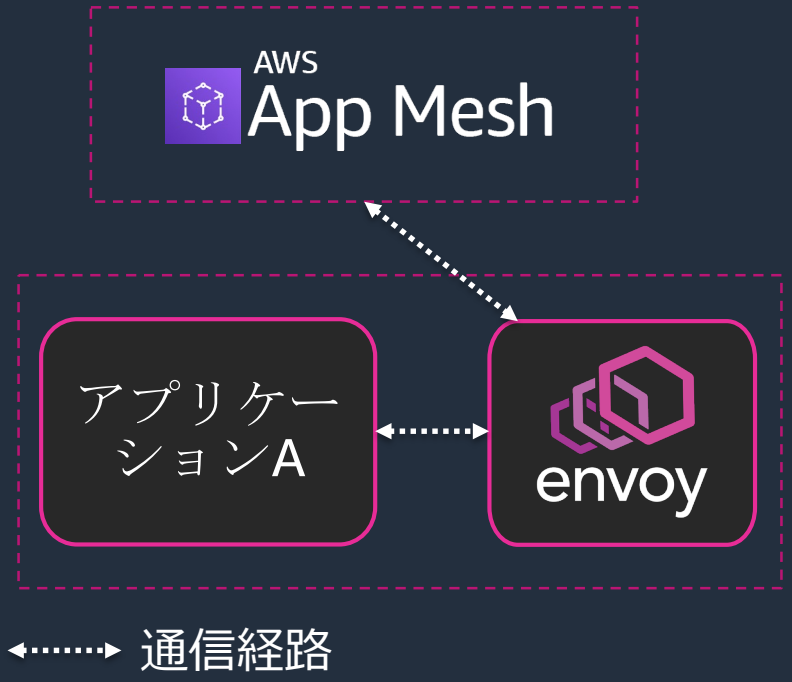
- サービスメッシュとは何か
- AWS App Mesh
  - 概要
  - 利用方法
  - 機能と活用例
  - ロードマップ
  - 価格体系

機能と活用方法

- AWS App Meshのセキュリティ機能
- AWS App Meshで利用できる機能
    - 通信可観測性を確保する
    - カナリアリリース

App Mesh Security of the Cloud

App Meshの構成の内、AWSが管理するところのセキュリティ

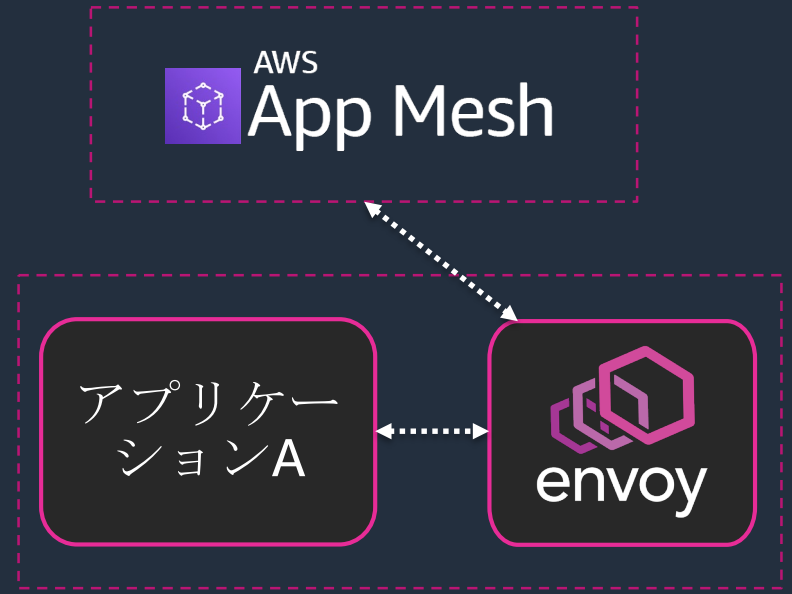


- App Meshの設定データの暗号化（配布時、保存時）
- Control Planeの可用性確保(Multi-AZ構成)
- Control Planeの自動バックアップ
- AWS PrivateLinkによるAWSネットワークに閉じたEnvoy-Control Plane間通信
- Envoyコンテナイメージの脆弱性スキャンとパッチ適用

https://docs.aws.amazon.com/app-mesh/latest/userguide/security.html

App Mesh Security in the Cloud

App Meshでお客様が管理できるセキュリティ



- App MeshのAPIに対するIAM権限
- セキュリティパッチが適用されたEnvoyコンテナイメージのデプロイ
- AWS CloudTrailでApp Mesh操作の証跡を監査

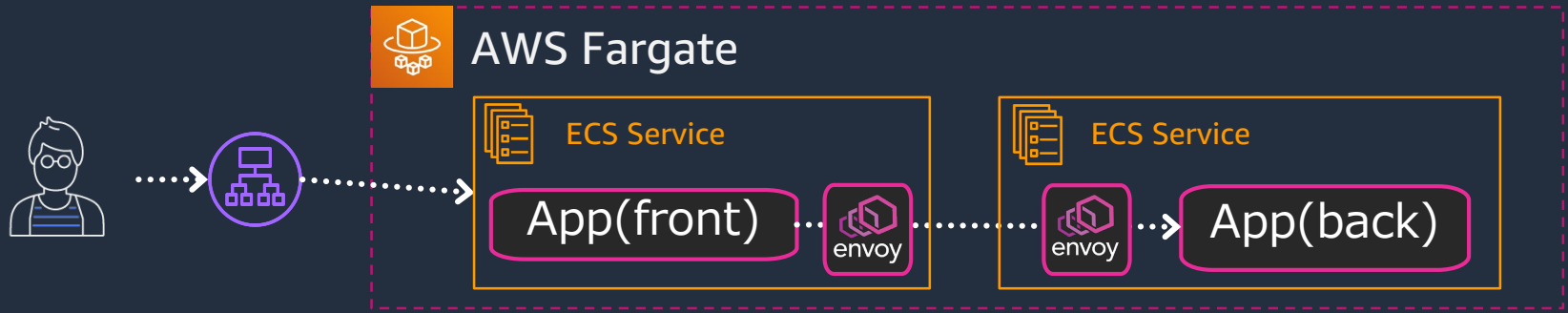
https://docs.aws.amazon.com/app-mesh/latest/userguide/security.html

App Meshで利用できる機能

クライアントサイドのロードバランシング
HTTPヘッダーベースのルーティング
Pathベースのルーティング
HTTP, HTTP2, gRPC, TCPのサポート
リトライポリシー
End-to-endのTLS暗号化
CloudWatch Logs and Metrics
AWS X-Ray Tracing
Envoyがサポートしている Metrics (StatsD, Prometheus)
EnvoyがサポートしているTracing(Zipkin, Jaeger)
Egress Traffic Policies
AWS Cloud Map Service Discovery
クロスアカウントのサポート
Kubernetes Controller
タイムアウトのサポート
Ingress Gateway

活用例: コンテナアプリケーションの通信可観測性

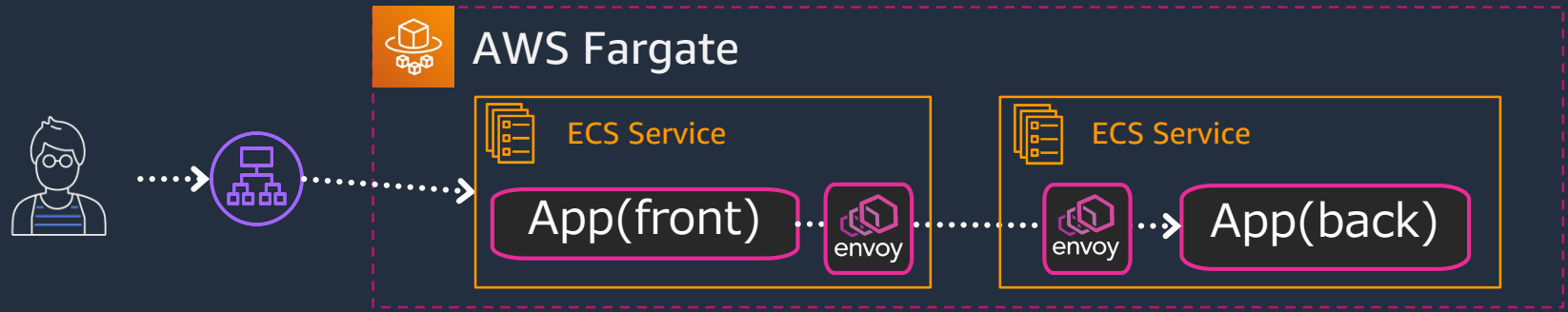
システム全体で、通信の状況がどうなっているのか把握する



- **メトリクス**
  - そのシステムで通信の状況が時間でどう変化してきたか
- **トレース**
  - システム全体のある時間帯の通信がどのようになっていたか
- **ログ**
  - あるコンポーネントの通信に関する詳細な情報

活用例: コンテナアプリケーションの通信可観測性

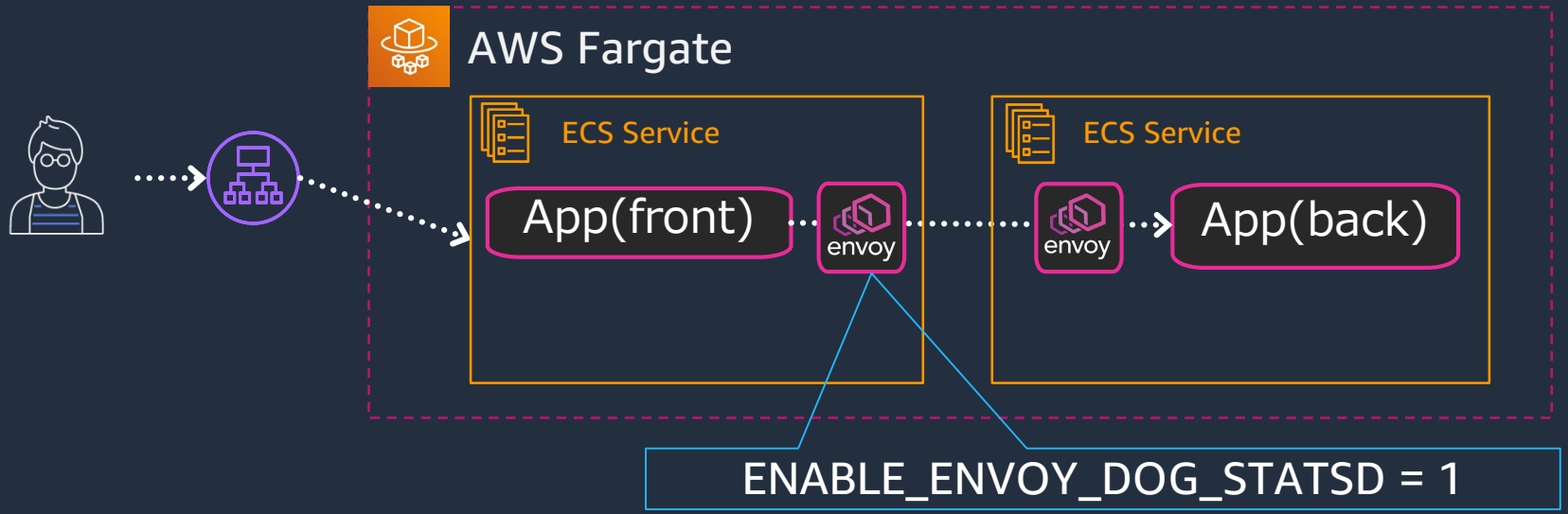
可観測性のためにApp Meshと連携できるAWSのサービス



- **メトリクス**
  - Amazon CloudWatch
- **トレース**
  - AWS X-Ray
- **ログ**
  - Amazon CloudWatch Logs

活用例: コンテナアプリケーションの通信可観測性

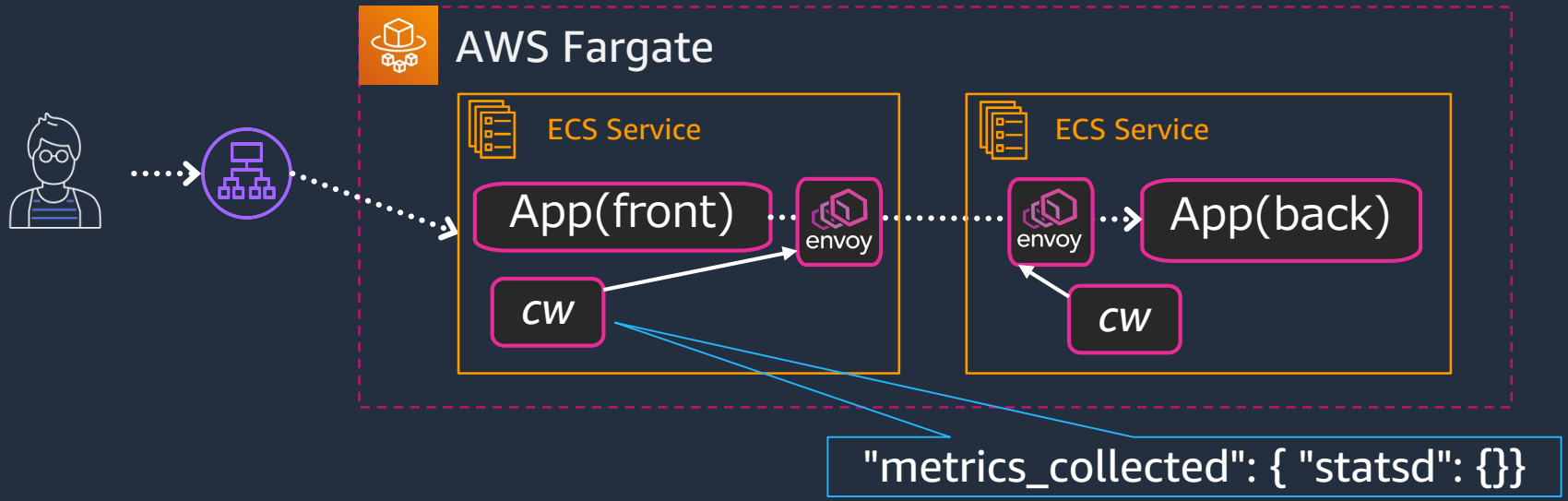
CloudWatchと連携して通信のメトリクスを取得する



Envoy Proxyコンテナに環境変数を設定し、StatsD形式のメトリクスを8125ポートで公開するよう設定する

活用例: コンテナアプリケーションの通信可観測性

CloudWatchと連携して通信のメトリクスを取得する

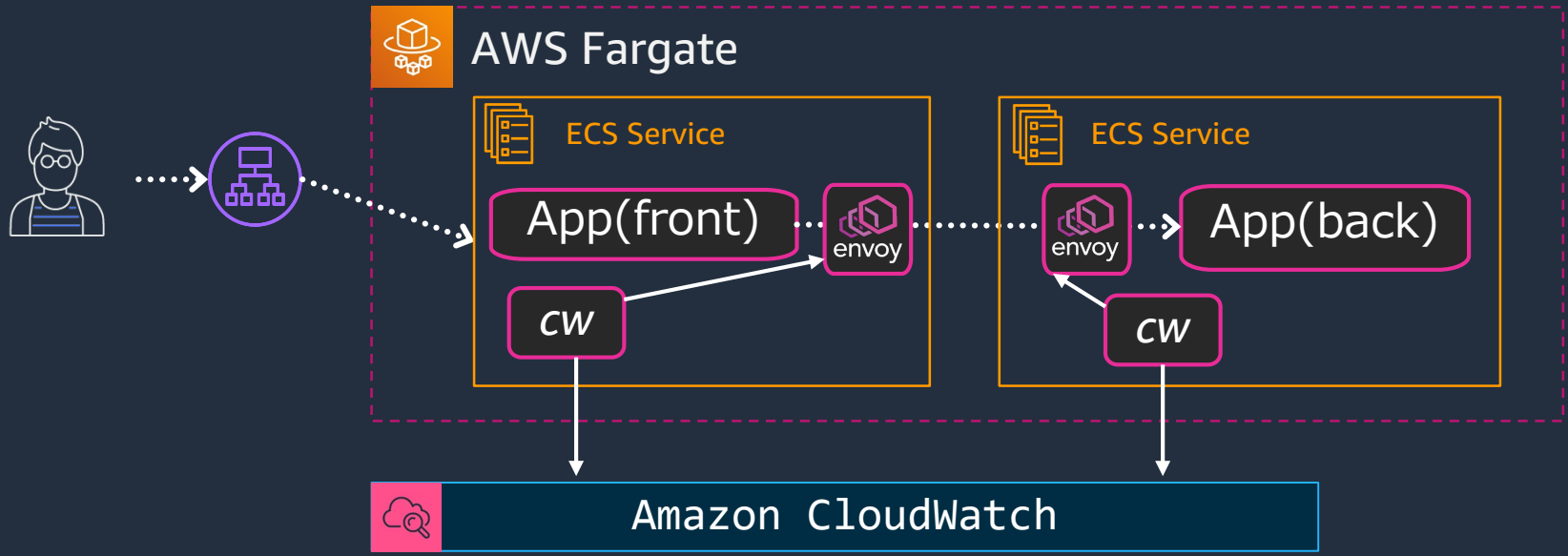


CloudWatch Agentをサイドカーとしてデプロイし、8125ポートで公開されているメトリクスを取得するよう設定する

※ CloudWatch Agentのコンテナイメージは、以下を参考にビルドする
https://github.com/aws-samples/aws-app-mesh-cloudwatch-agent/blob/master/Dockerfile

活用例: コンテナアプリケーションの通信可観測性

CloudWatchと連携して通信のメトリクスを取得する



CloudWatch Agentが、Envoyが公開するStatsDメトリクスをCloudWatchに送信する

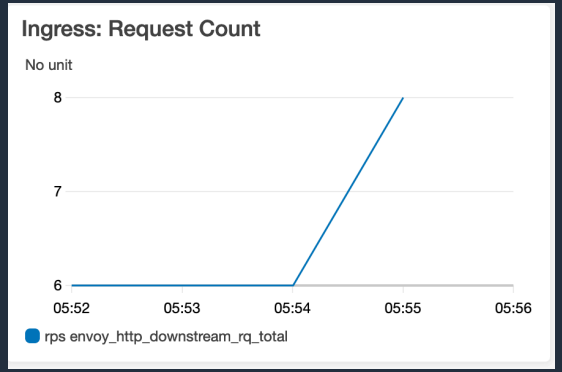
活用例: コンテナアプリケーションの通信可観測性

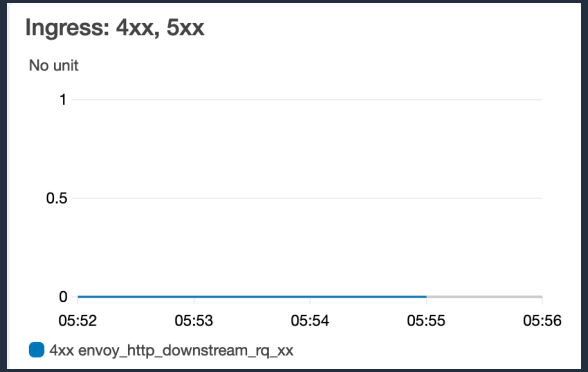
CloudWatchと連携して通信のメトリクスを取得する

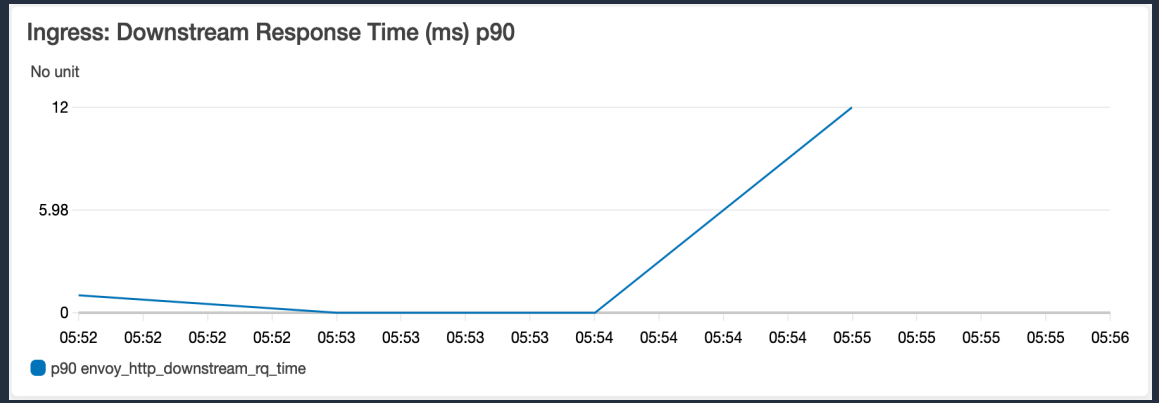
CloudWatchに通信のメトリクスが集まる

↓

- Dashboardを作成して一覧性を高める
- アラームを設定して閾値を超えたら通知する

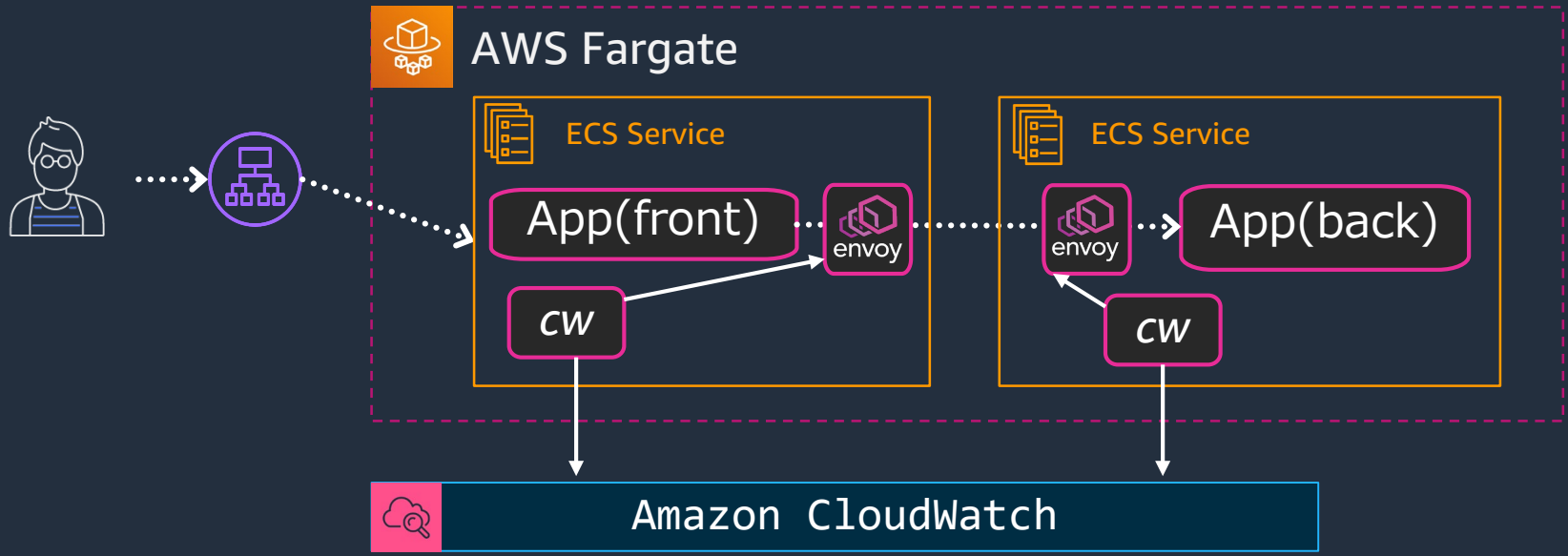






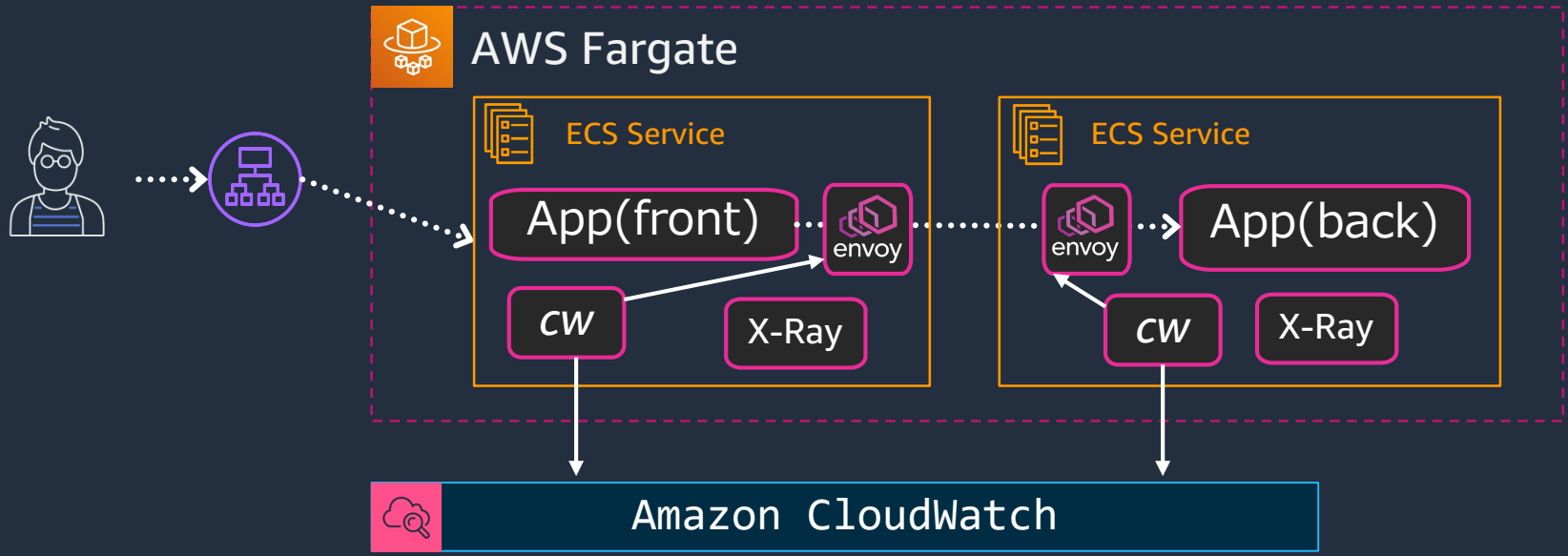
活用例: コンテナアプリケーションの通信可観測性

X-Rayで通信をトレースする



活用例: コンテナアプリケーションの通信可観測性

X-Rayで通信をトレースする

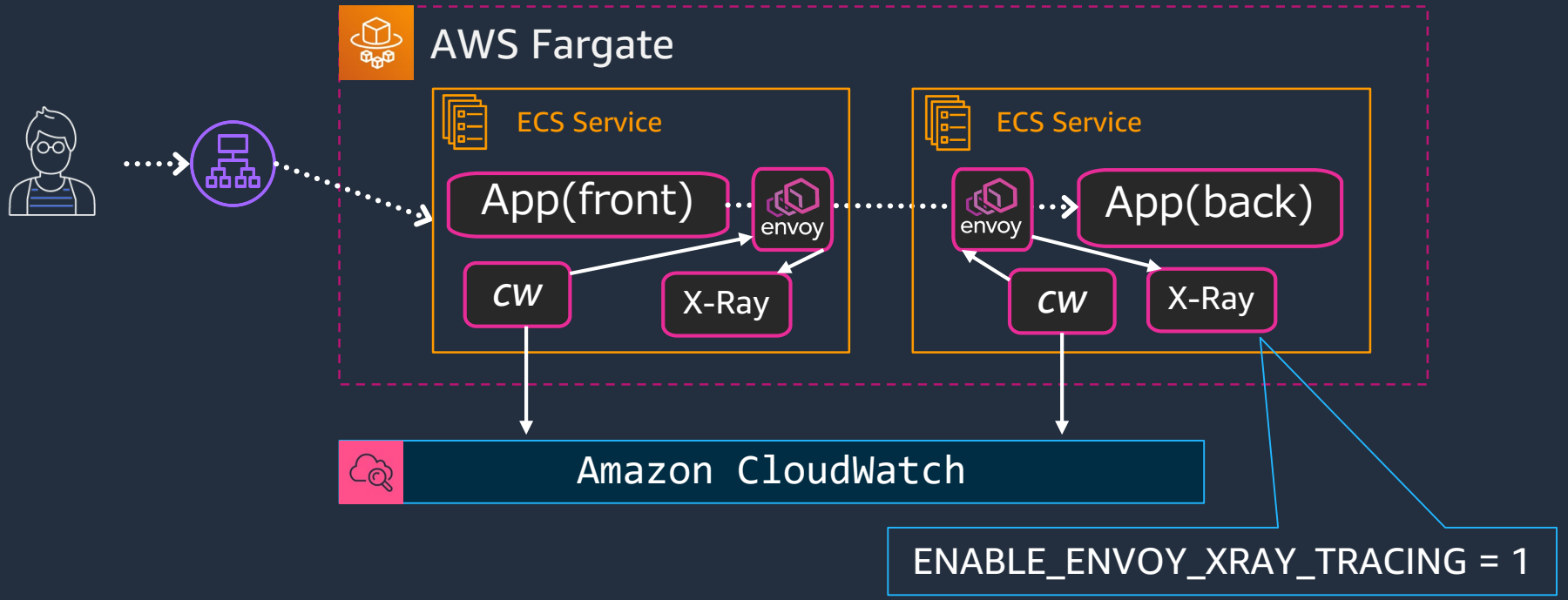


X-Rayデーモンのコンテナ(amazon/aws-xray-daemon)をサイドカーとしてデプロイ(※)

※ 詳細は「Amazon ECS で X-Ray デーモンを実行する」を参照
https://docs.aws.amazon.com/ja_jp/xray/latest/devguide/xray-daemon-ecs.html

活用例: コンテナアプリケーションの通信可観測性

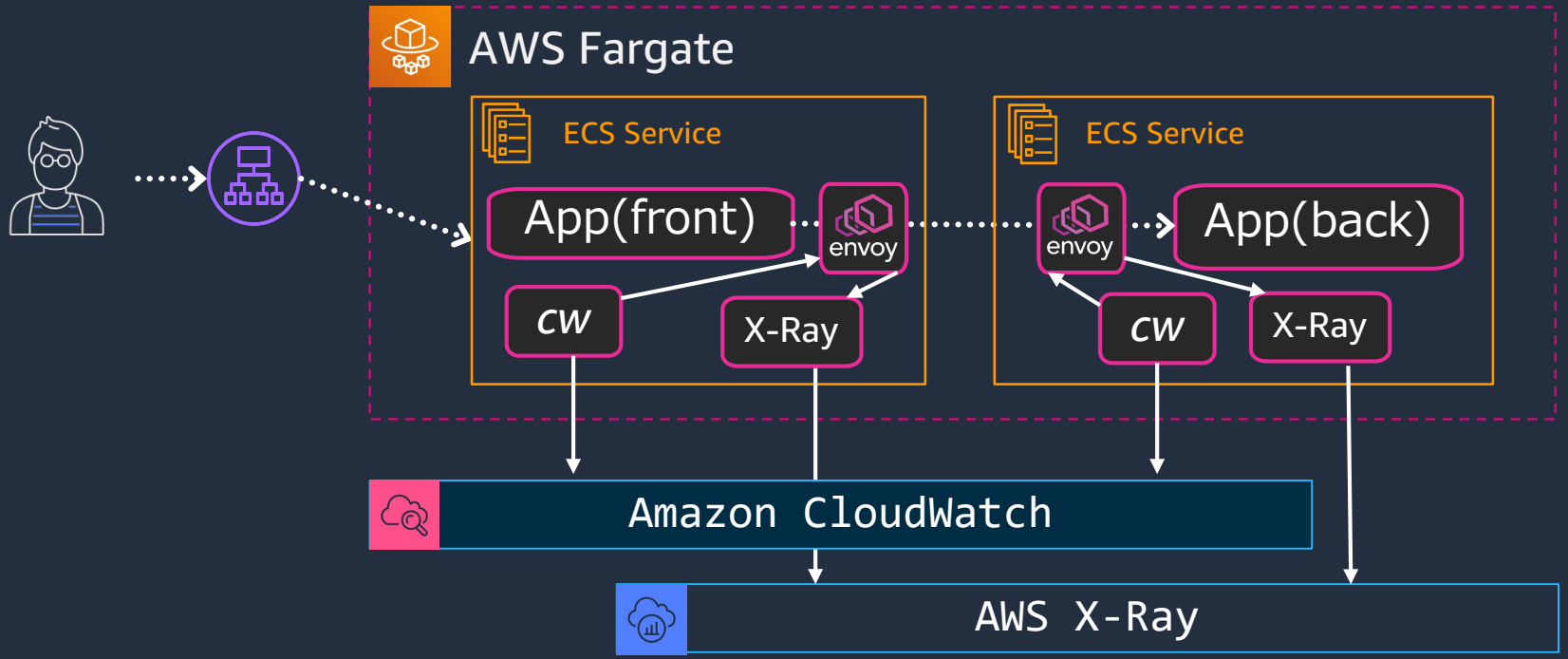
X-Rayで通信をトレースする



Envoyに環境変数を設定し、X-Rayデーモンにトレースデータを送信する

活用例: コンテナアプリケーションの通信可観測性

X-Rayで通信をトレースする

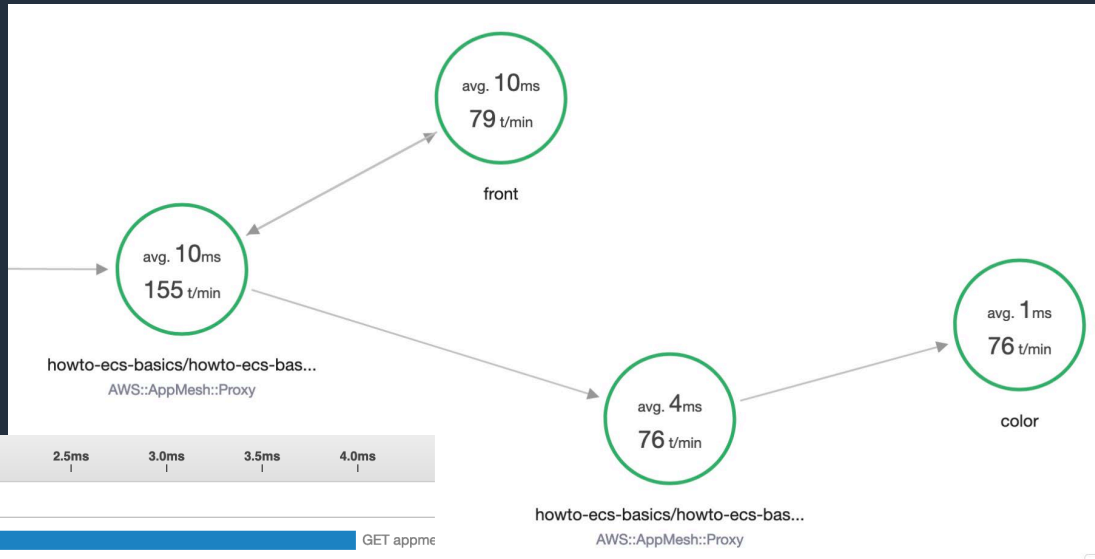


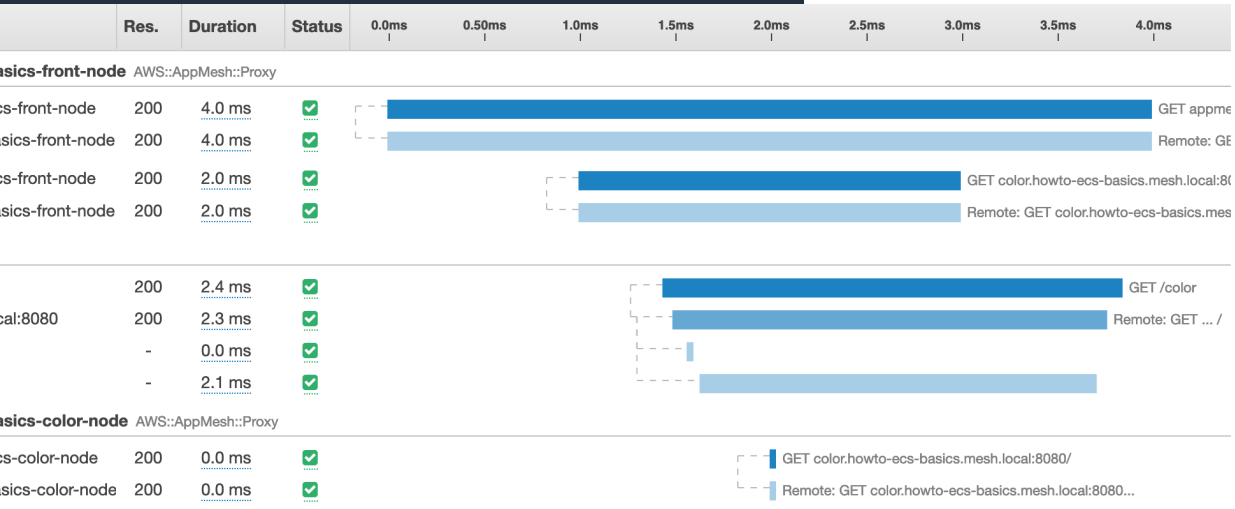
X-RayデーモンがAWS X-RayにEnvoyのトレースデータを中継する

活用例: コンテナアプリケーションの通信可観測性

X-Rayで通信をトレースする

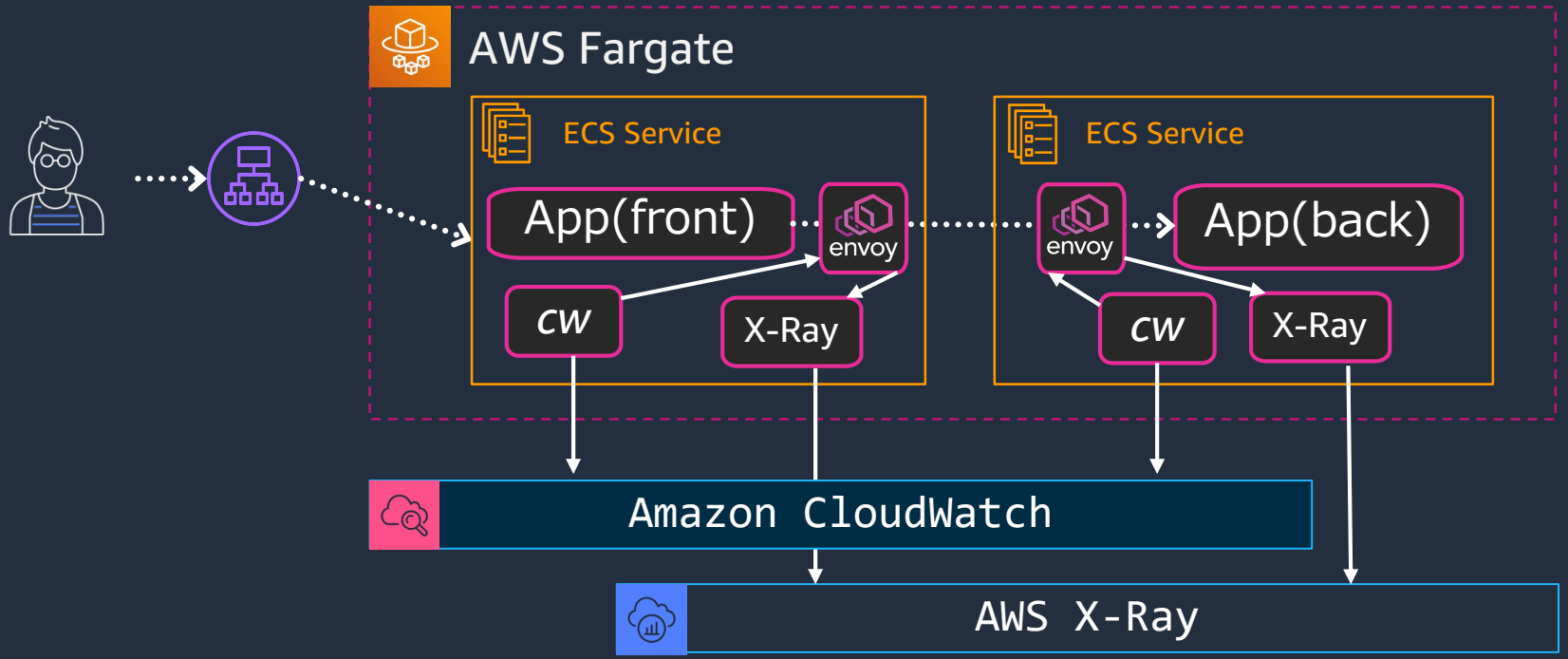
メトリクスで異常があったときに、どこに遅延やエラーがあるか確認できる





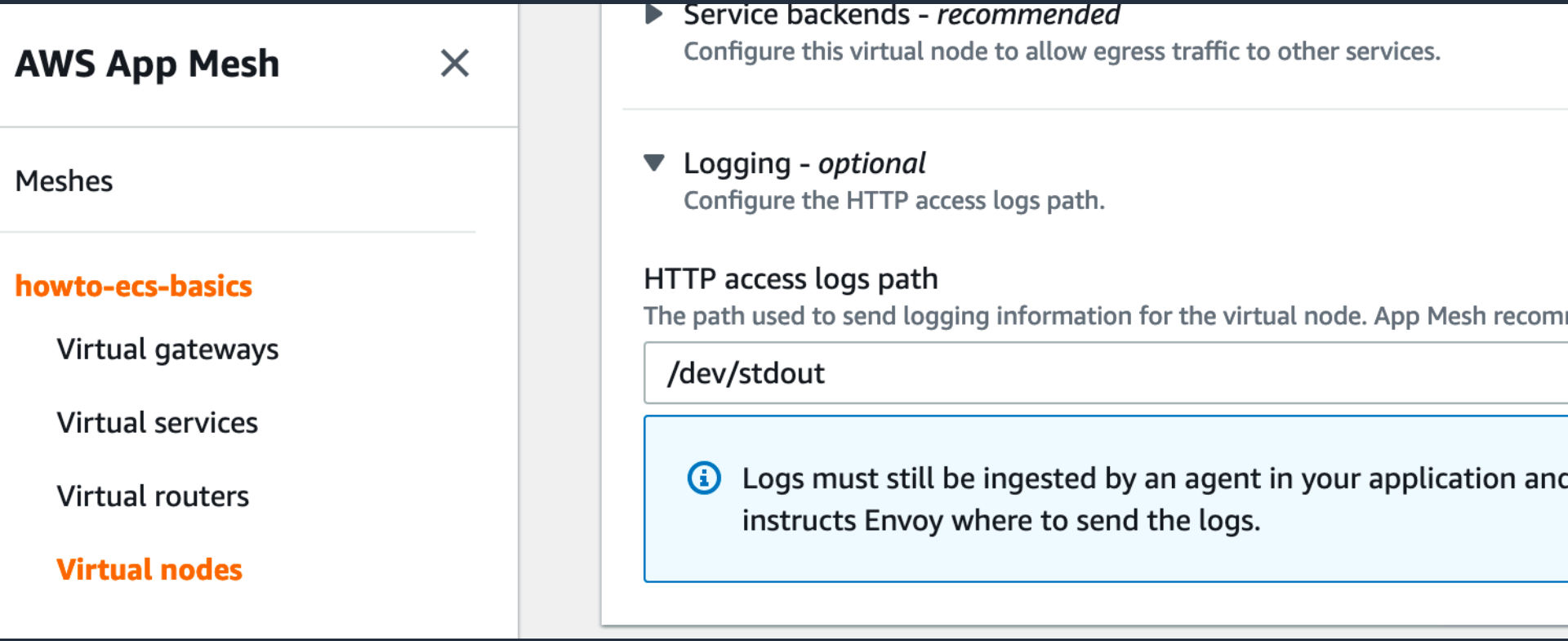
活用例: コンテナアプリケーションの通信可観測性

CloudWatch Logsでアクセスログを表示する



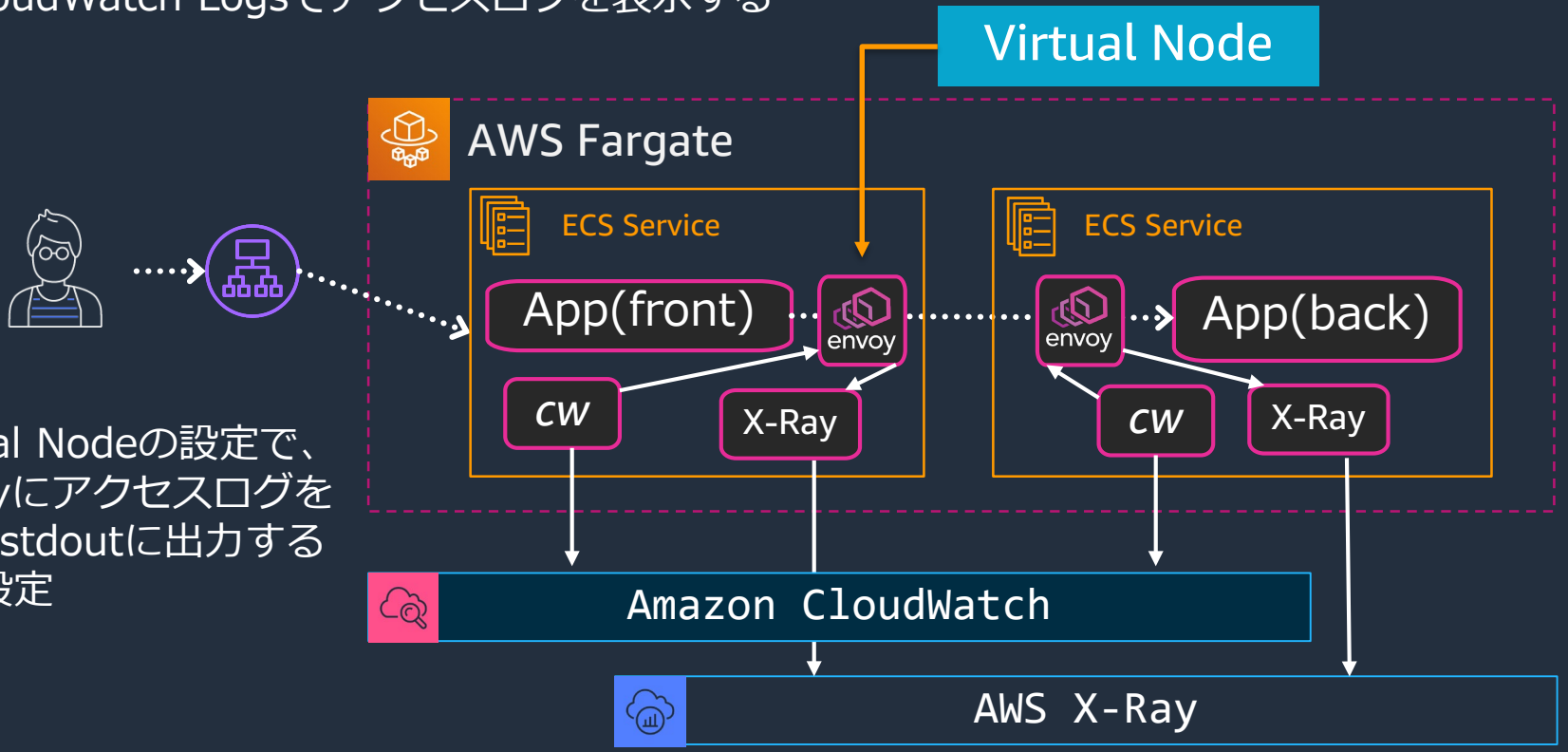
活用例: コンテナアプリケーションの通信可観測性

CloudWatch Logsでアクセスログを表示する



活用例: コンテナアプリケーションの通信可観測性

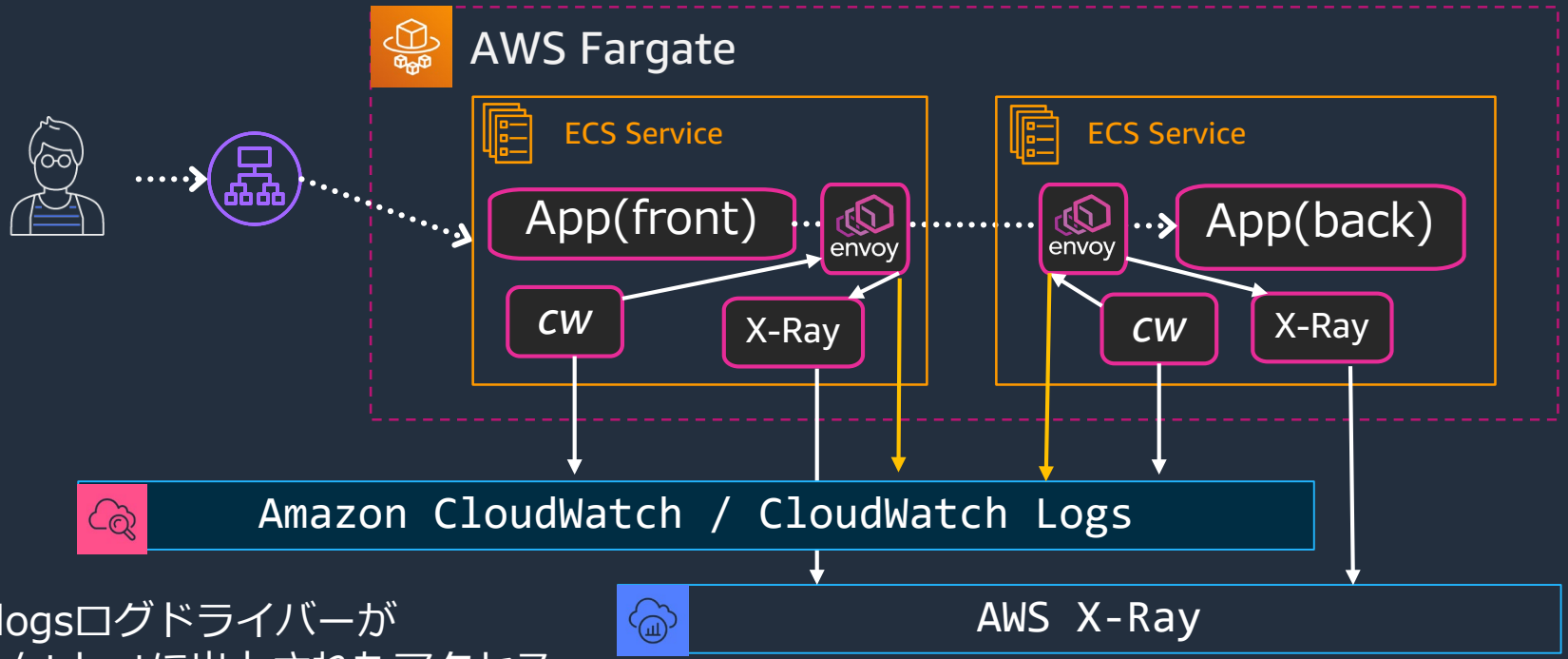
CloudWatch Logsでアクセスログを表示する



Virtual Nodeの設定で、Envoyにアクセスログを/dev/stdoutに出力するよう設定

活用例: コンテナアプリケーションの通信可観測性

CloudWatch Logsでアクセスログを表示する

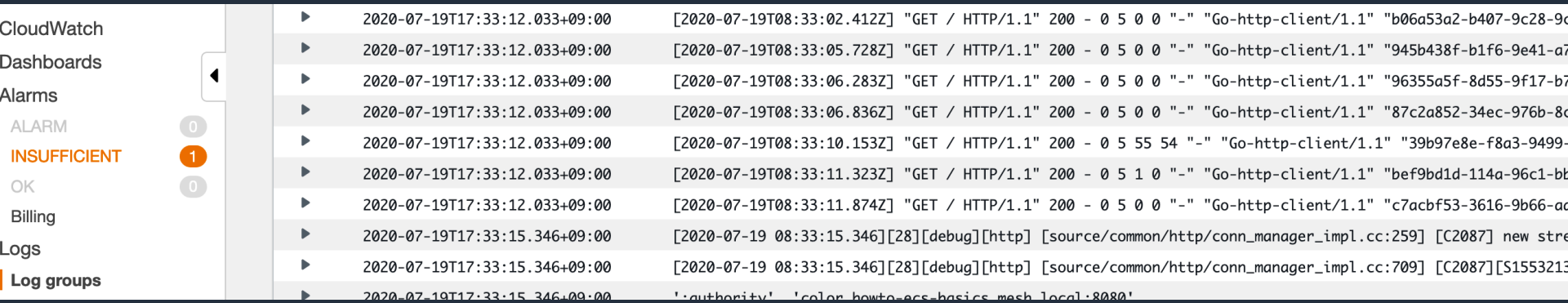


awslogsログドライバーが/dev/stdoutに出力されたアクセスログをCloudWatch Logsに送信

Envoyアクセスログの送信

活用例: コンテナアプリケーションの通信可観測性

CloudWatch Logsでアクセスログを表示する



aws

活用例: コンテナアプリケーションの通信可観測性

エラー、遅延が発生したときに状況を把握する

- **メトリクス**
  - Amazon CloudWatch
  - 異常やアラームが発生したことを把握する
- **トレース**
  - AWS X-Ray
  - アラームが発生したとき、どのような通信状況だったか把握する
- **ログ**
  - Amazon CloudWatch Logs
  - トレースで特定したコンポーネントの処理を確認する

aws

活用例: カナリアリリース

Virtual Routerによる通信先の振り分け

AWS App Mesh

AWS App Mesh

Meshes

blackbelt

Virtual gateways

Virtual services

Virtual routers

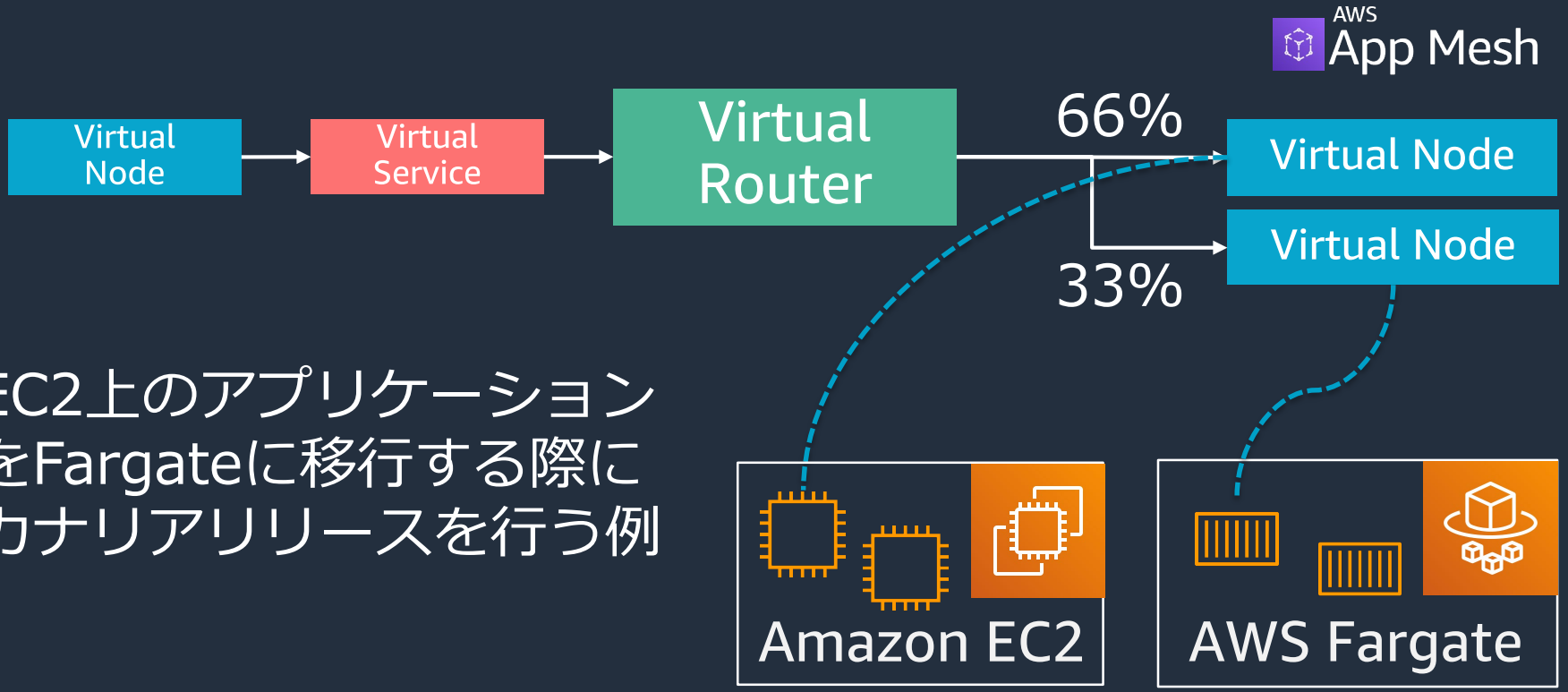
Targets

Virtual node name	Weight	Percentage
serviceb-blackbelt	2	66.7
serviceb2-blackbelt	1	33.3

Headers

aws

活用例: カナリアリリース



EC2上のアプリケーションをFargateに移行する際にカナリアリリースを行う例

- サービスメッシュとは何か
- AWS App Mesh
    - 概要
    - 利用方法
    - 機能と活用例
    - ロードマップ
    - 価格体系

AWS App Meshのロードマップ

github.com/aws/aws-app-mesh-roadmap

#33 AWS Lambdaとの連携
LambdaをMeshに追加できるようにする(EnvoyからLambdaを起動する)

#61 CloudWatchとの統合を進める
EnvoyとCloudWatchの連携を、もっとManagedなレベルで行う

#34 サービス間認証
mTLSでサービス間の認証を行う

#107 Rate Limiting
Virtual Nodeへの入出力にレートリミットを設ける

#6 Circuit Breaker
閾値を超えてエラーや遅延が発生している呼び出しを一時的に遮断する

※ 2020年7月時点のロードマップアイテムからピックアップ

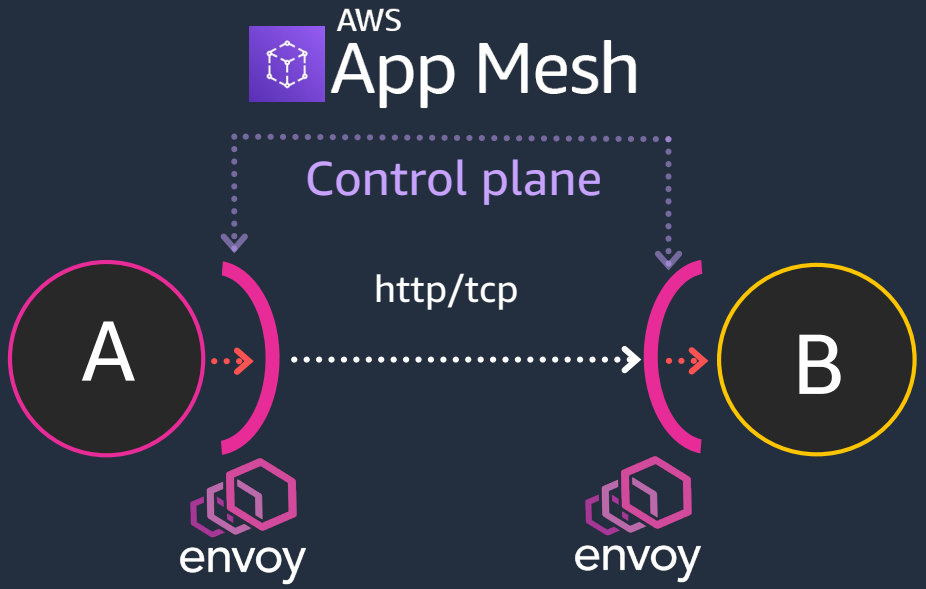
- サービスメッシュとは何か
- AWS App Mesh
  - 概要
  - 利用方法
  - 機能と活用例
  - ロードマップ
  - 価格体系

利用料金

- App Meshの利用による追加料金はない

- 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



Envoyを管理するコントロールプレーンを提供し、サービスメッシュを実現するマネージドサービス

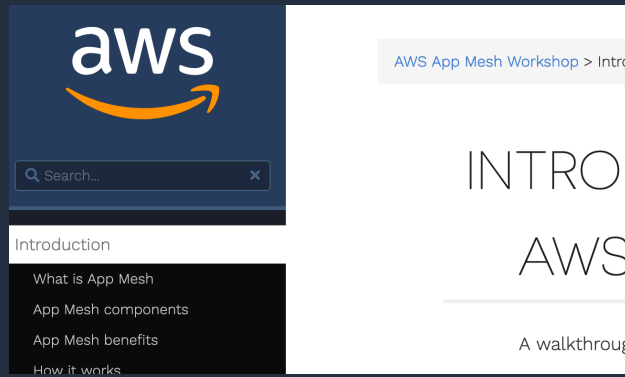
EC2, ECS, EKS, Kubernetes on EC2で利用可能

多種多様な技術で稼働する多数のアプリケーションの通信をインフラレイヤーで制御できる

手を動かしながら理解する

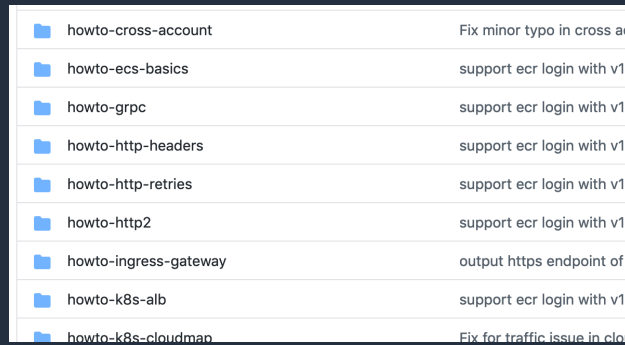
AWS App Mesh Workshop
https://www.appmeshworkshop.com/

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の利用、リトライ、Header
ベースルーティングなど、活用例を手を動かして確認



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