AWS Proton Hands on Workshop

説明資料

Nripendra Shrestha

2022/05/13

Amazon Web Services Japan G.K.

AWS Proton Hands-on Workshop

Sr.	Module	Level	予定
1	AWS Proton 概要及びHands on Labついて説明	200	5月13日(金)　15:00 - 16:00
2	Workshop準備 (AWS Account, Cloud9, GitHub Account)	200	（AWS側で準備）
3	AWS Proton Core Lab　（Proton, ECS)	300	5月16日(月)　10:00 - 12:00
4	Observability and Networking (AppMesh, X-ray, Prometheus)　**[Stretch Goal]**	300	5月16日(月) 資料提供・自由実施
5	予備/振り返り及びNext StepsについてDiscussion	200	5月17日(火)　14:00 - 15:00
6	Observability and Networking (Grafana) [Skip]	300	資料提供・自由実施
7	Multi-account Deployment　[Skip]	300	資料提供・自由実施
8	Using Terraform for Deployment [Skip]	300	資料提供・自由実施

AGENDA

- Hands on Labの説明
  - スケジュール
  - Labの流れ
  - Lab環境について
- 振り返り
  - AWS Protonの概要
  - 用語説明
  - Q&A
- Workshop環境へアクセス確認

PART I

Hands on Labの説明

Hands on Labの目的

- AWS Proton利用プロセス全体の体験
- AWS Protonの理解を深める
- AWS Protonについて運用・開発チーム協業イメージの確立
- AWS Protonを活用したサービス開発に自信を持っていただく

Hands on Labの流れ

準備

- AWSアカウント開設
- Cloud9の設定　（コード確認・CLIベースに操作）
- GitHubアカウント開設・コードFork・CodeStar経由で接続

CoreLabの実施

- 環境作成
- サービス作成
- MINOR機能更新：FARGATE -> FARGATE_SPOT

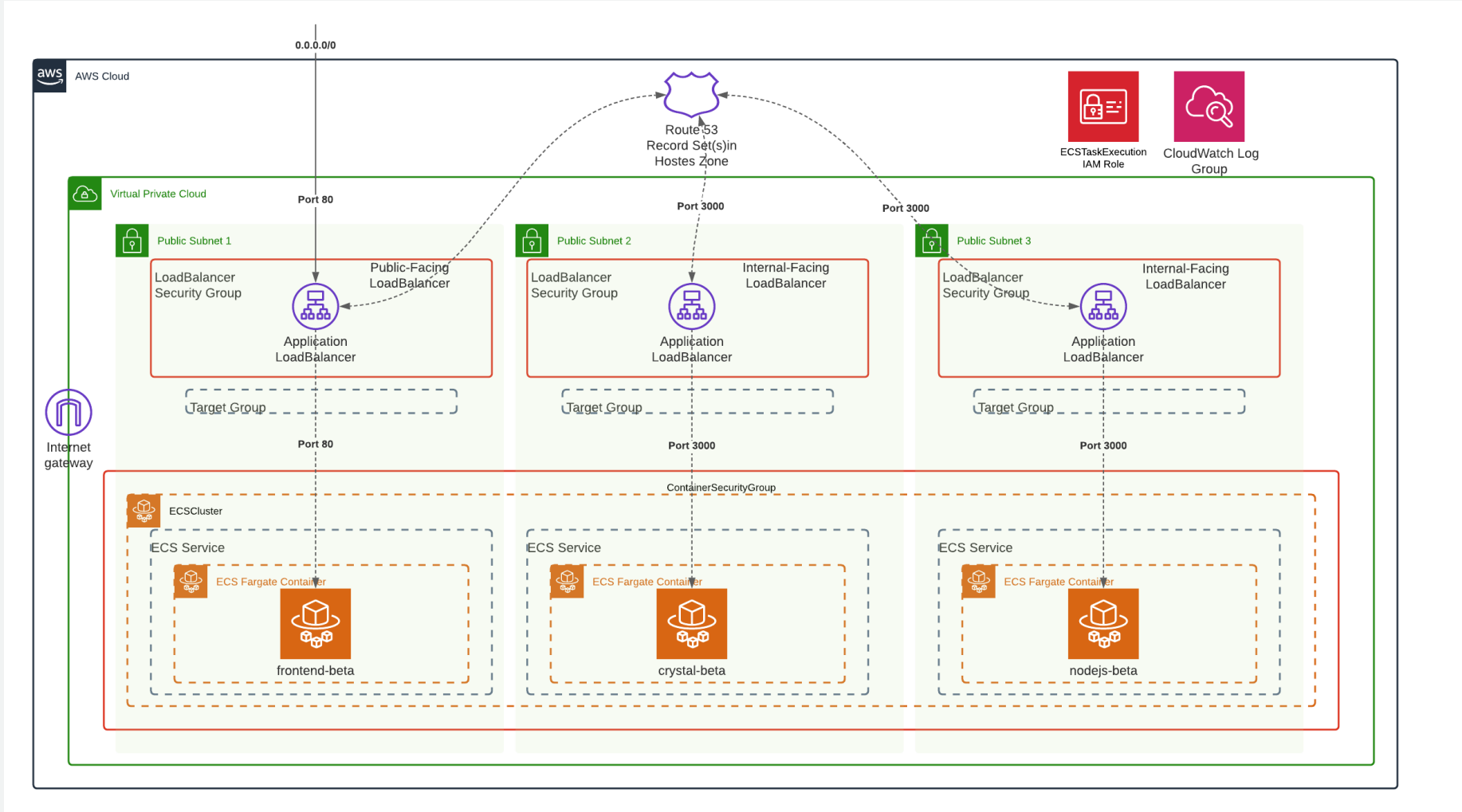
Observabilityの機能追加

- MAJOR機能更新：AppMesh及びPrometheusの追加
- サービスの更新・動作確認

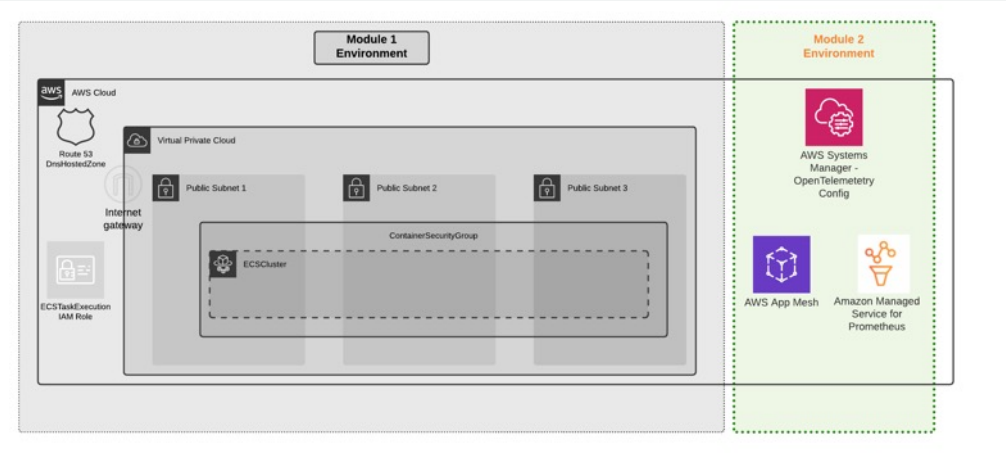
マルチアカンウトの利用

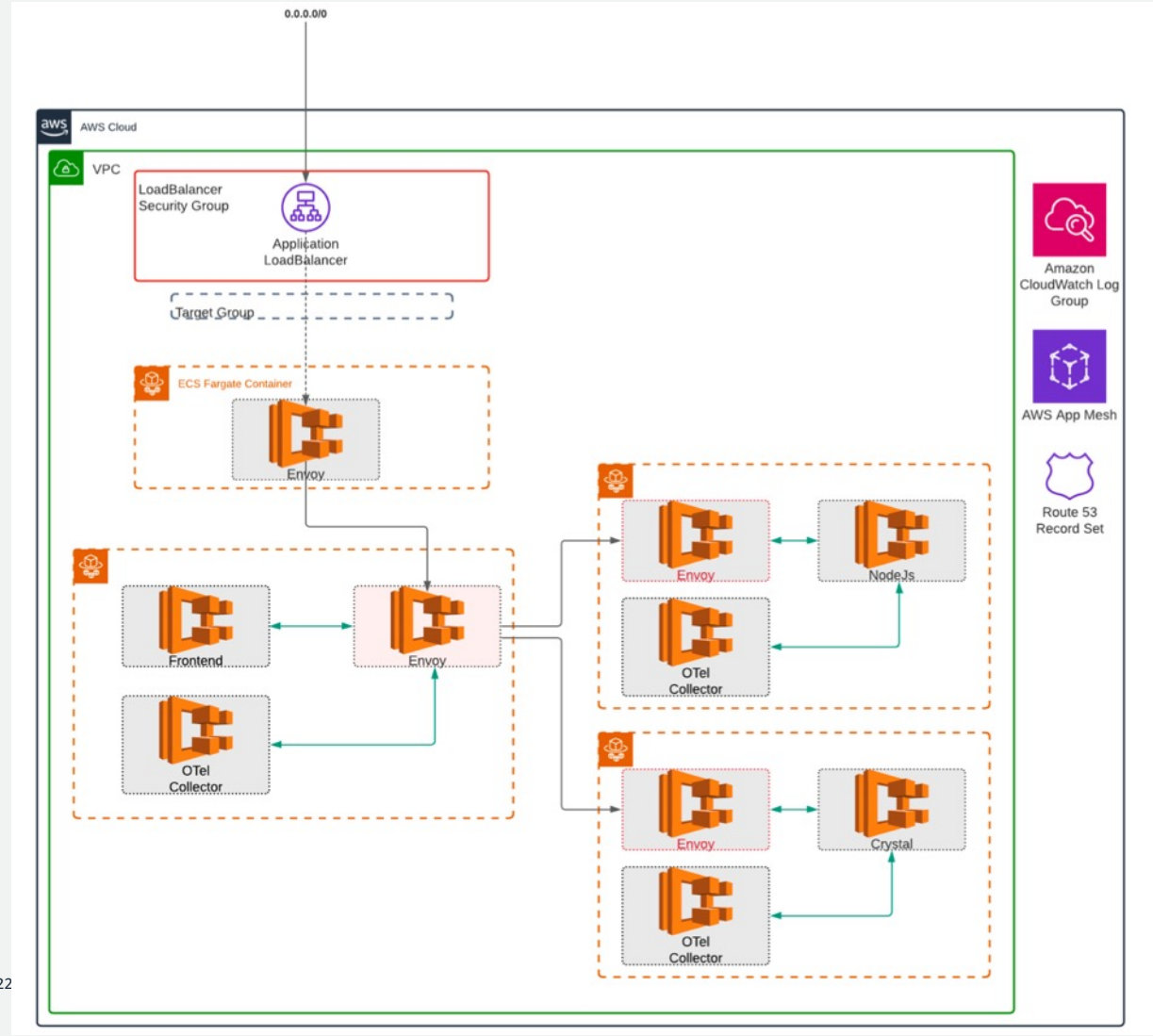
- アカウント間接続設定
- 新規環境・サービスの作成

Core Workshop(Module1) Target Architecture



Standard Observability (Module2) Target Architecture



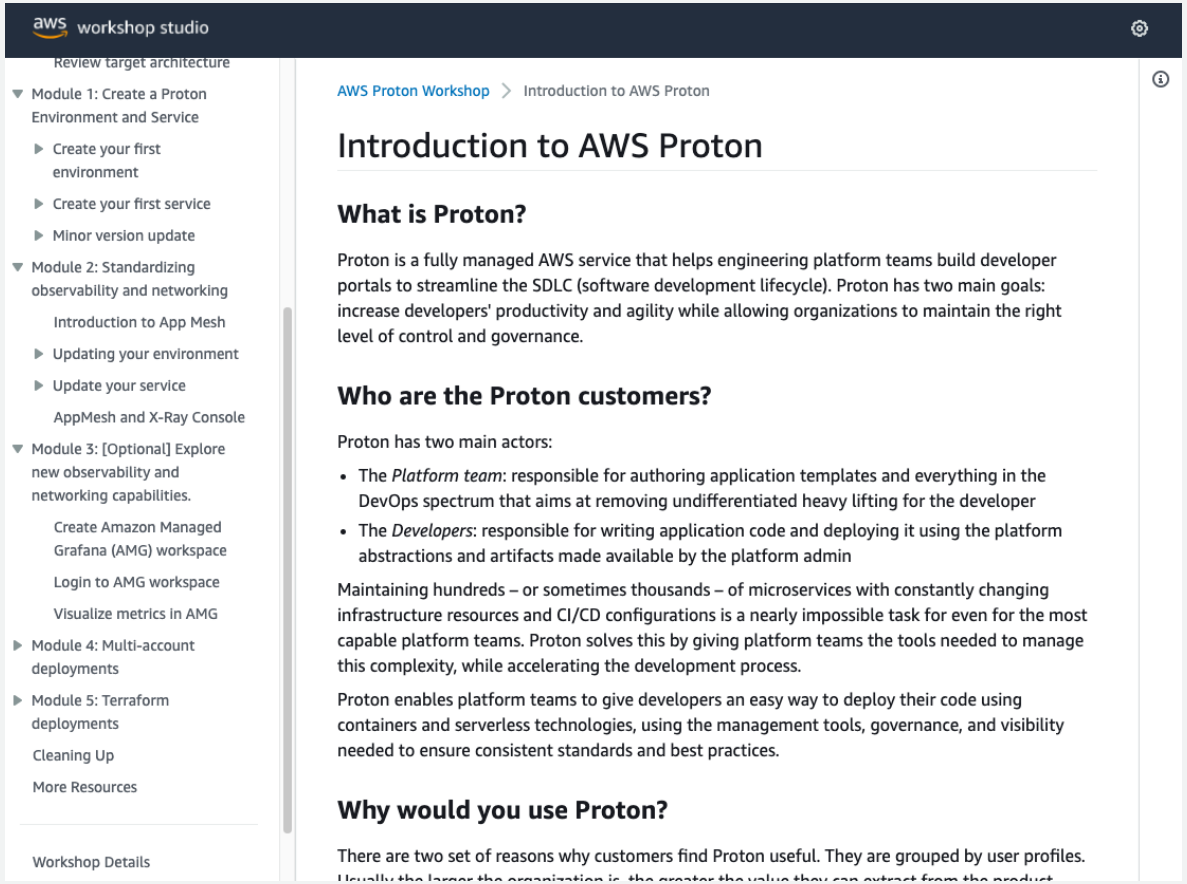


Workshop 準備について

- WebEx (AWSが準備済み）
- AWSアカウント開設（人数分）- AWS SSO (本日確認）
  - AWS管理コンソールへのアクセスできる（WTSは準備済み)
  - ChromeなどModernBrowserがインストールされている
- Cloud9の設定　（コード確認・CLIベースに操作）
- GitHubアカウント開設 （人数分）- 本日確認
  - IaCのSkeletonコード取得
  - Applicationコードの取得
  - CI/CDのトリガー
- Workshopの資料は英語・PPTで提供・コード別提供（Cloud9上)

補足

Workshopは近日公開（2022/5/13）



補足

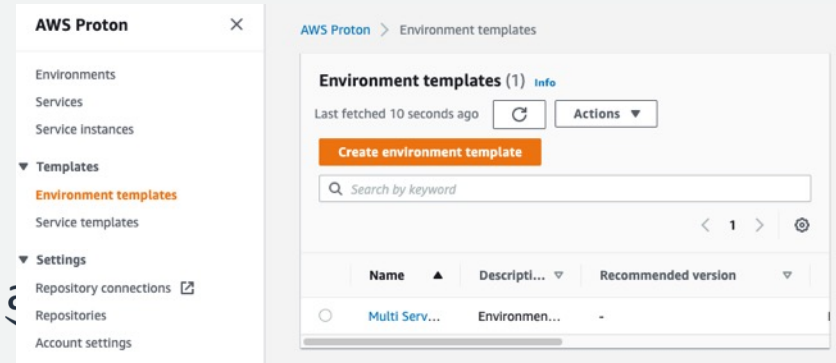
3.1.2 Environment Template, Repository Link 作成

Create Environment Template

#3.1.2.1
sh commands/createEnvTemplate

```
aws proton create-environment-template \
  --region ${AWS_REGION} \
  --name "multi-svc-env" \
  --display-name "Multi Service Environment" \
  --description "Environment with VPC and public subnets"
```

```
AWSReservedSSO_AWSAdministratorAccess_82957608b23b12c3:~/environment $ sh commands/crea
teEnvTemplate {
    "environmentTemplate": {
        "arn": "arn:aws:proton:ap-northeast-1:841600881451:environment-template/multi-s
vc-env",
        "createdAt": "2022-05-12T01:16:18.669000+00:00",
        "description": "Environment with VPC and public subnets",
        "displayName": "Multi Service Environment",
        "lastModifiedAt": "2022-05-12T01:16:18.669000+00:00",
        "name": "multi-svc-env"
    }
}
```

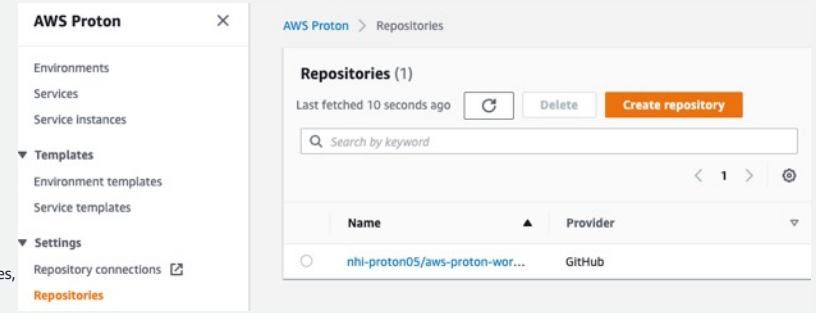


Create Proton Repository Link

#3.1.2.2
sh commands/createProtonRepositoryLink

```
CODESTAR_CONNECTION_ARN=$(aws codestar-connections list-connections \
        --provider-type GitHub \
        --region ${AWS_REGION} | \
      jq -r '.Connections[] | select( .ConnectionName=="MyConnection")
aws proton create-repository \
  --region ${AWS_REGION} \
  --connection-arn $CODESTAR_CONNECTION_ARN \
  --name "${GITHUB_USERNAME}/aws-proton-workshop-bundles" \
  --provider "GITHUB"
```

```
AWSReservedSSO_AWSAdministratorAccess_82957608b23b12c3:~/environment $ sh commands/crea
teProtonRepositoryLink{
    "repository": {
        "arn": "arn:aws:proton:ap-northeast-1:841600881451:repository/github:nhi-proton
05/aws-proton-workshop-bundles-public",
        "connectionArn": "arn:aws:codestar-connections:ap-northeast-1:841600881451:conn
ection/0aa37f19-84e8-48f0-87a4-95abbaef5b5f",
        "name": "nhi-proton05/aws-proton-workshop-bundles-public",
        "provider": "GITHUB"
    }
}
```



AWS Proton概要

コンテナを活用する上での課題

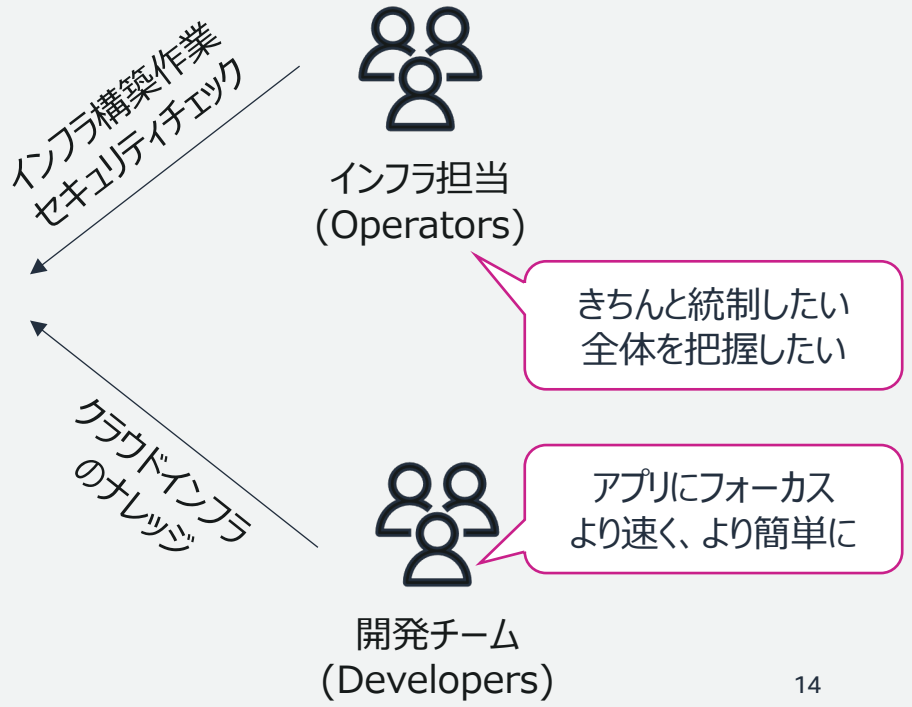
コンテナは扱いやすいが、動かすためには色々な準備が必要

フロントエンド
アプリケーション

バックエンド
アプリケーション

- ビルドパイプライン
- オートスケーリング
- ロードバランサー
- オーケストレーションツール
- ホスティング環境

アプリケーションをコンテナ上でホストするのに必要な要素



Summary

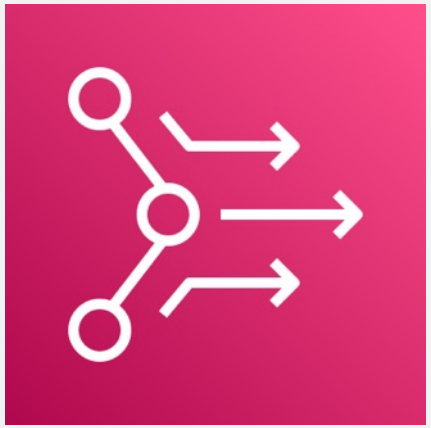
- Proton は何をするものか？
    - 基盤管理者に対してはIaC で制御可能なインフラストラクチャ標準機構と一元管理機構を提供する
    - 開発者に対してはセルフサービスでアプリ実行環境とコードのデプロイポイントを提供する
        - 開発プロセス（SDLC)にそう形で仕組み化

- Proton を使った場合の統制やオブザーバビリティ観点での制約はあるのか？
    - 基本、大きな制約は無いと考えていただいて良いが、EKSは未だ対応していない
    - Proton はアプリケーションワークロードの機能を制限しない（サービステンプレートで記述することで、チューニングの余地や自由度を保つことが可能）
    - オブザーバビリティ　機能についても同様のことが言え、お客様はモニタリングツールを変更したりする必要は無い

- Proton を使うには
    - 環境やサービスといった概念を理解した上で自社のニーズや利用形態にマップする
    - テンプレートのプロトタイピング
    - モルモットまたはトライアル実施

AWS Proton

インフラストラクチャの統制を強化しつつ開発者の生産性向上とイノベーションスピードを加速するマネージドサービス
インフラ担当者（Ops）と開発者（Dev)を結びつける機能を提供



AWS Proton

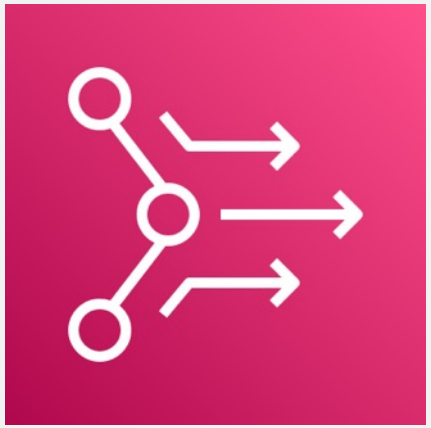
- コンテナアプリケーションまたはサーバレスアプリケーションが対象

- Proton が提供する機能
  - 開発者向けセルフサービスUI（管理コンソール）
  - 一元管理・可視性
  - 2種類のテンプレートとバージョン管理
  - 1クリック・アップグレード
  - インフラストラクチャ向けパイプライン
  - アプリケーションデプロイパイプライン（３rdパーティ統合）

- マルチアカウントサポート
- Proton自体は追加料金無しで利用可能

AWS Proton : プラットフォーム視点

インフラストラクチャの統制を強化しつつ開発者の生産性向上とイノベーションスピードを加速するマネージドサービス
インフラ担当者（Ops）と開発者（Dev)を結びつける機能を提供



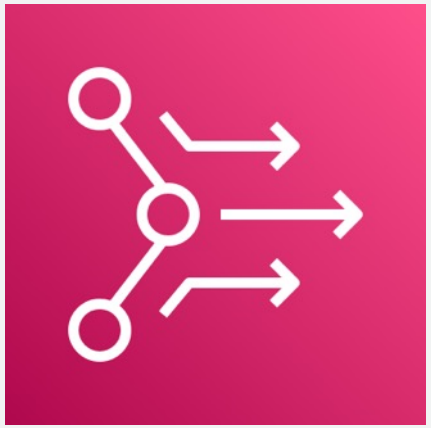
AWS Proton

- プラットフォーム（基盤管理）チームがコンテナのインフラとデプロイメントツールを定義する
- 開発者はコンテナのインフラをセルフサービスで展開でき、コードを自由にデプロイできる
- 基盤チームは必要なインフラリソースを展開し、監視ツールやCI/CDパイプラインを含むアプリケーションスタックを定義する
- CloudFormationを含む既存のIaCツール*で管理できる要素・リソースはProtonでも管理・調整可能

*Terraform ベースのテンプレートも利用可能

AWS Proton : 開発者視点

インフラストラクチャの統制を強化しつつ開発者の生産性向上とイノベーションスピードを加速するマネージドサービス
インフラ担当者（Ops）と開発者（Dev)を結びつける機能を提供



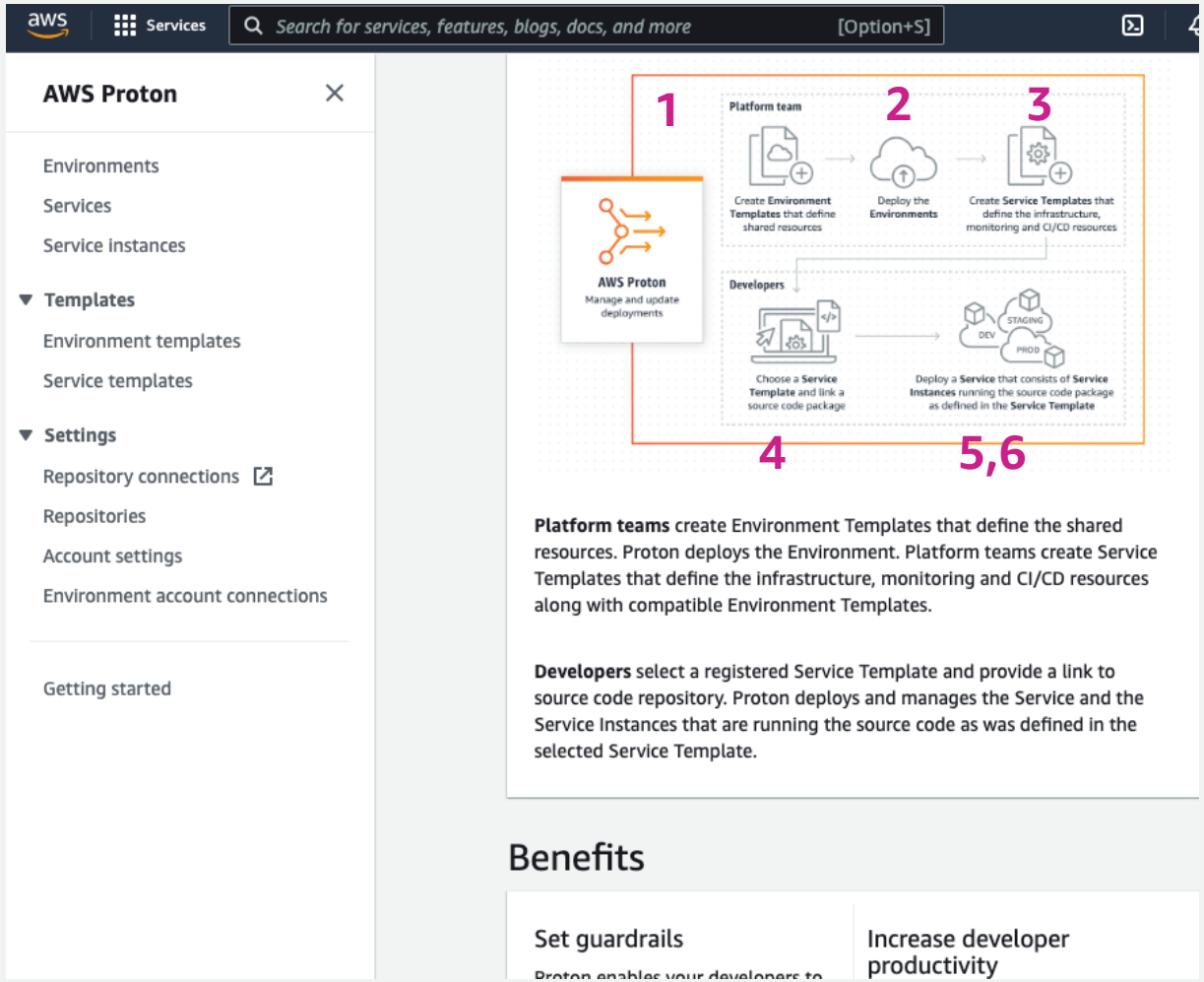
AWS Proton

- 開発者はコンテナのインフラをセルフサービスで展開でき、展開したあとはコードを自由にデプロイできる
  - 展開時に必要なパラメータ*を指定するのみ
- コードをデプロイするためのツール、パイプラインもそこに用意されるため、アプリケーションに集中できる

*例えばCPUやメモリといったコンテナのリソースなどを指定する

基本概念・用語

AWS Proton 利用イメージ



1. 環境テンプレートを作成
   環境テンプレートは共有リソースを定義する
2. 環境をデプロイ
3. サービステンプレートを作成
   サービステンプレートはインフラ、モニタリング、CI/CDリソースを含む
4. サービステンプレートを選び、ソースコードパッケージを連携
5. サービステンプレートの内容に従って開発、ステージング、本番といった環境をオンデマンド展開
6. 継続的な更新も可能

環境 テンプレート とサービス テンプレート (Bundle)

Environment Template

環境テンプレート

アプリケーションワークロードが乗る**足回りの共通リソース**を定義する

- VPC
- Subnets, Security Groups
- IAM Roles
- ECS Clusters…etc.

```
environment
├── schema
|   ├── schema.yaml
├── infrastructure
    ├── manifest.yaml
    ├── cloudformation.yaml
```

Service Template

サービステンプレート

アプリケーションワークロードそのもの（を構成する要素）を定義する

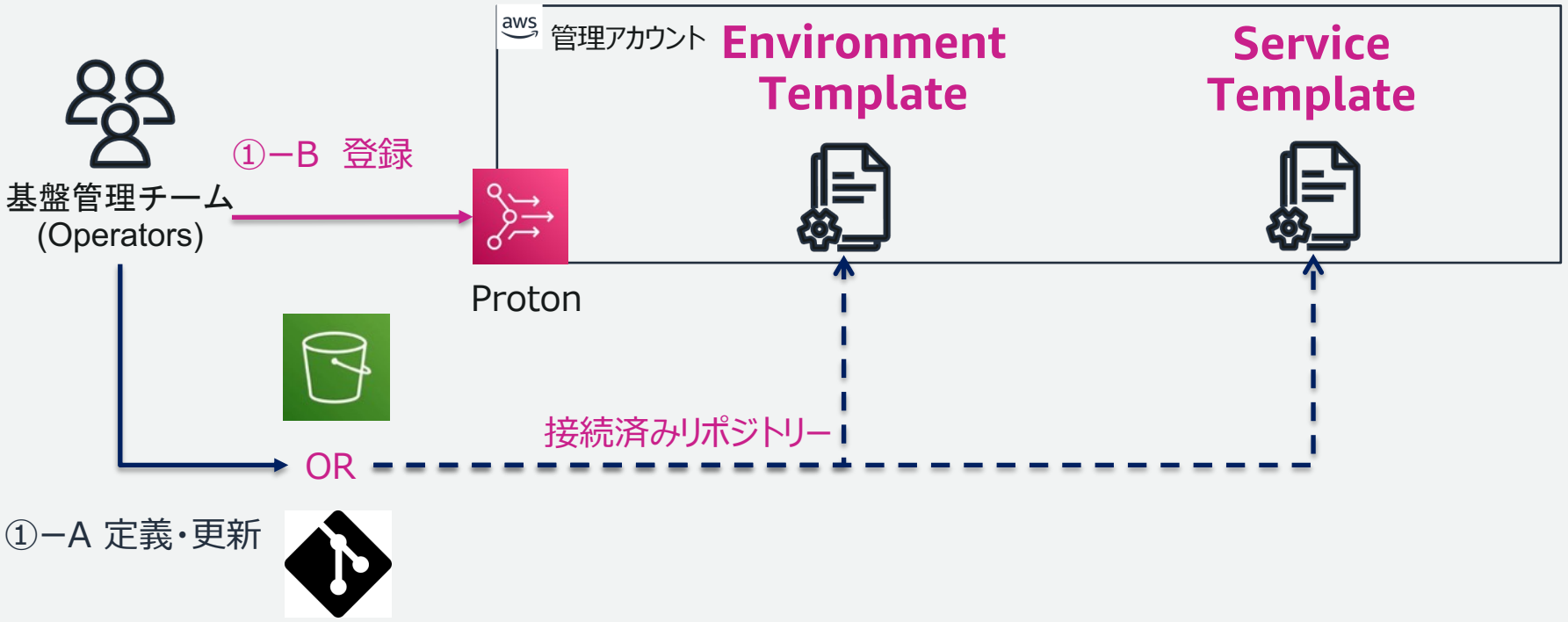
例）

- ECS Tasks/Services
- ELB
- Code Pipeline, Code Build, Code Deploy
- ECR Registry…etc.

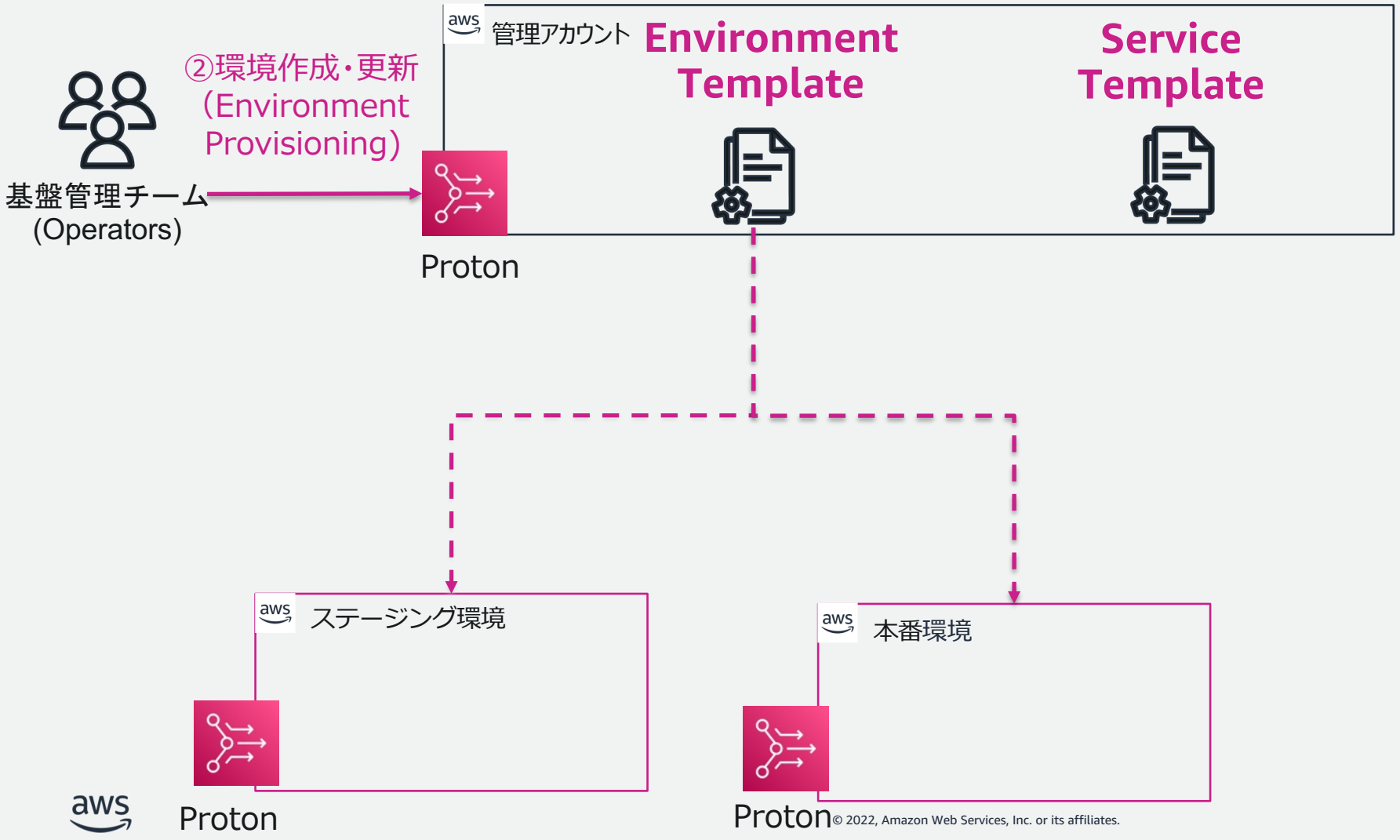
```
service
├── schema
|   ├── schema.yaml
├── instances_infrastructure
|   ├── manifest.yaml
|   ├── cloudformation.yaml
├── pipeline_infrastructure
    ├── manifest.yaml
    ├── cloudformation.yaml
```

ベストプラクティスに則したCurated Templates　が雛形として用意されている
https://github.com/aws-samples/aws-proton-sample-templates

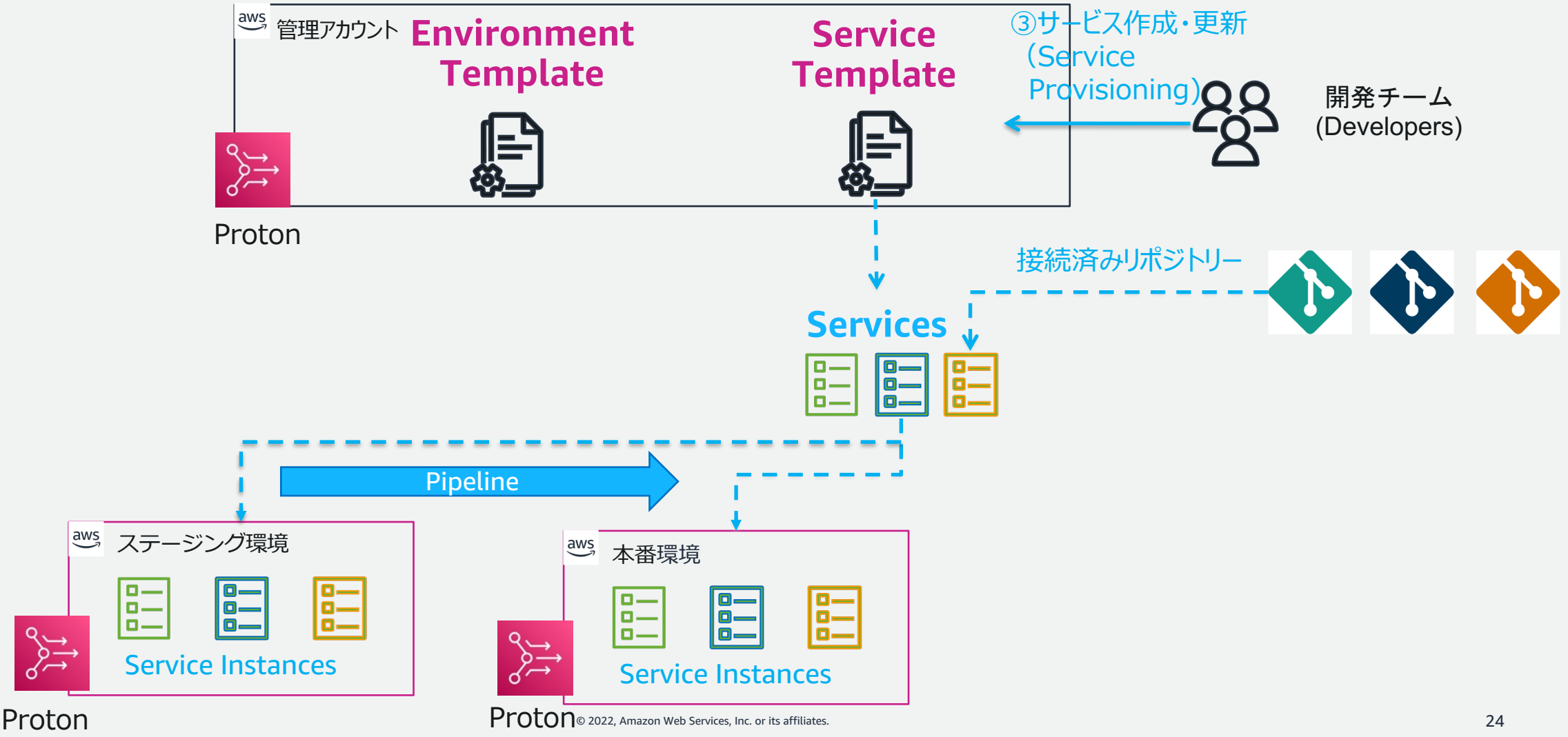
作業のイメージ



作業のイメージ



作業のイメージ

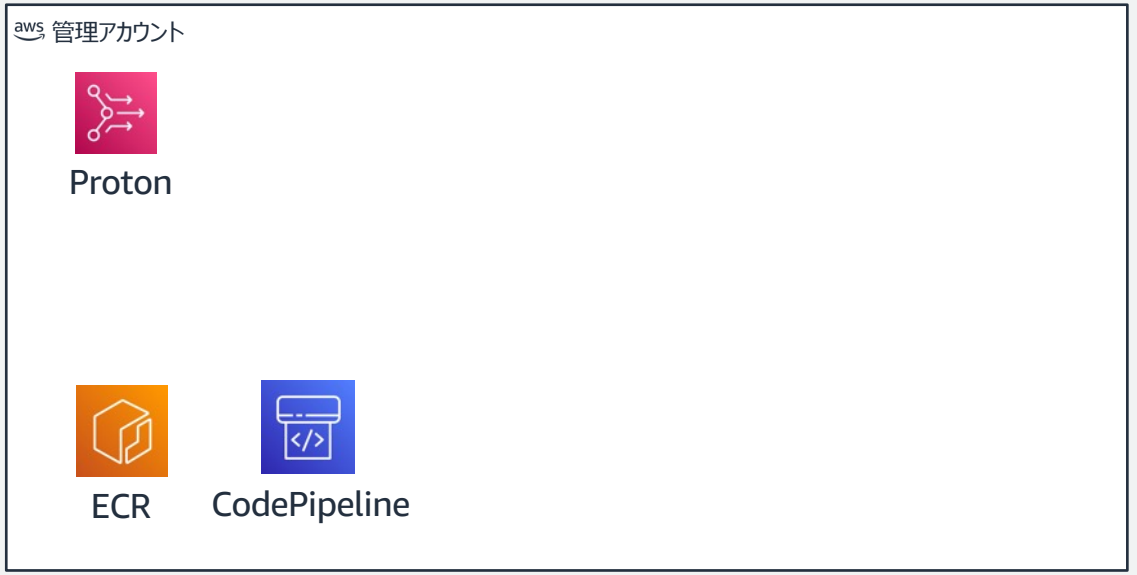


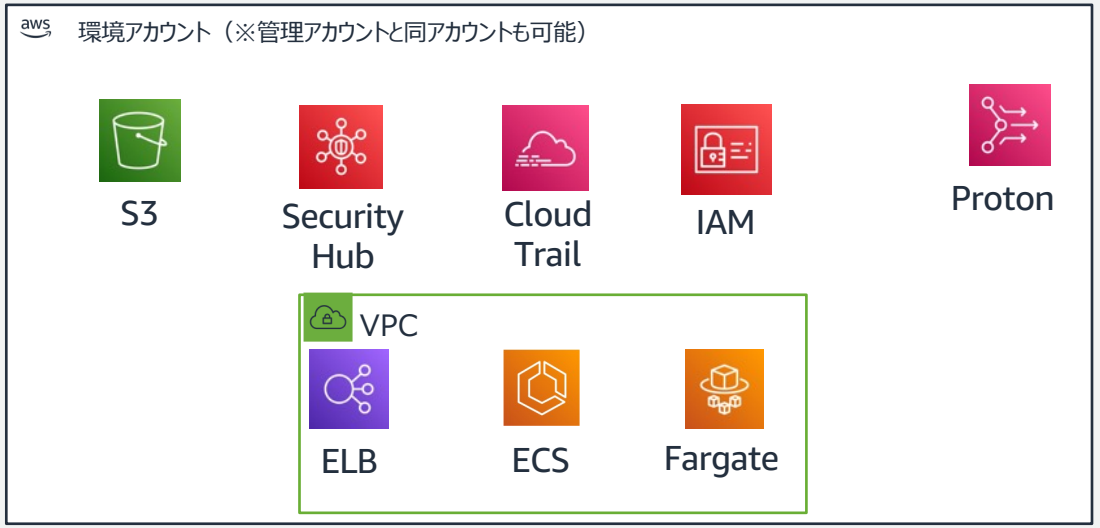
環境とサービス

Protonが扱うインフラストラクチャの構成要素は「環境」と「サービス」の２つからなる

環境として展開されるもの（例）

サービスとして展開されるもの（例）

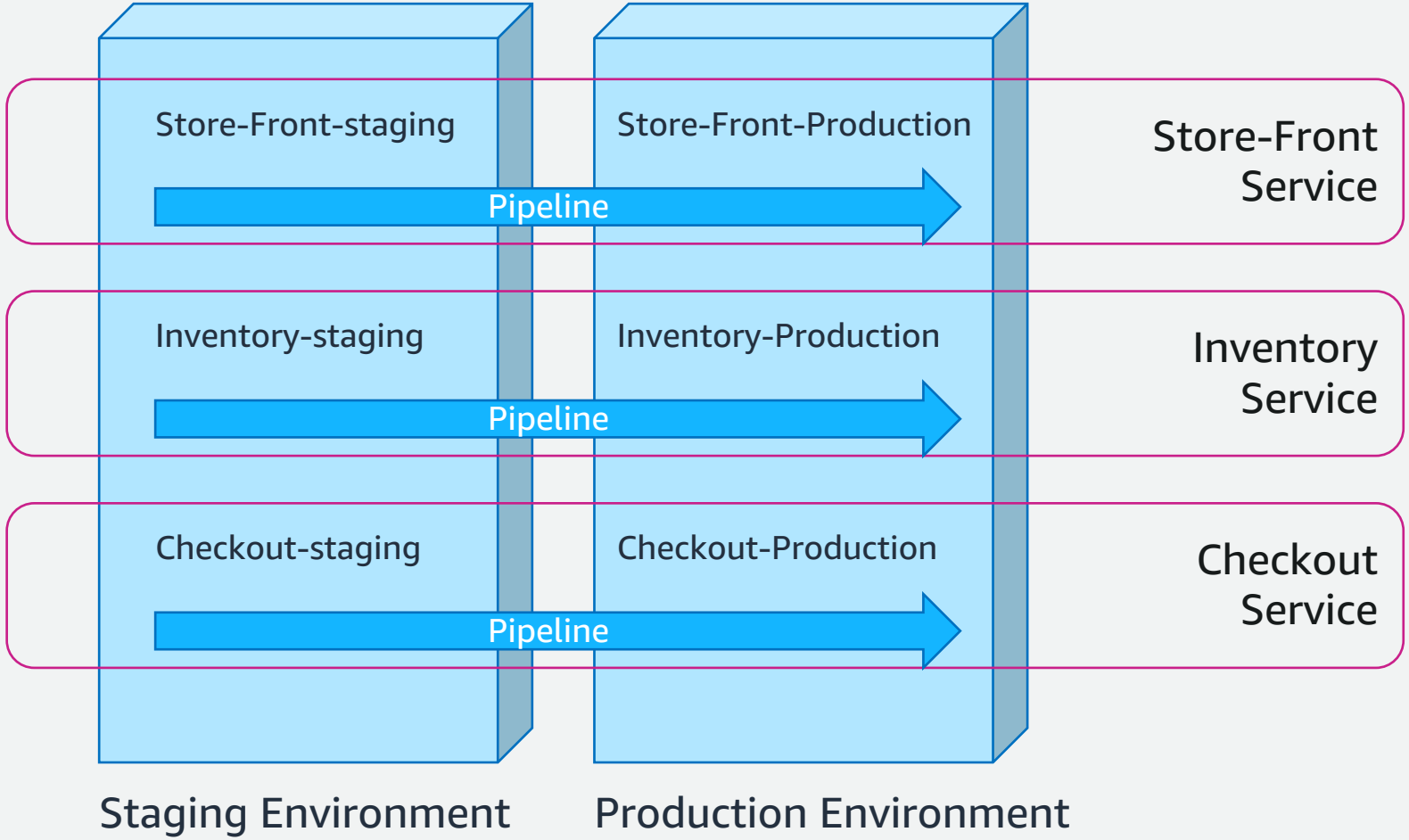




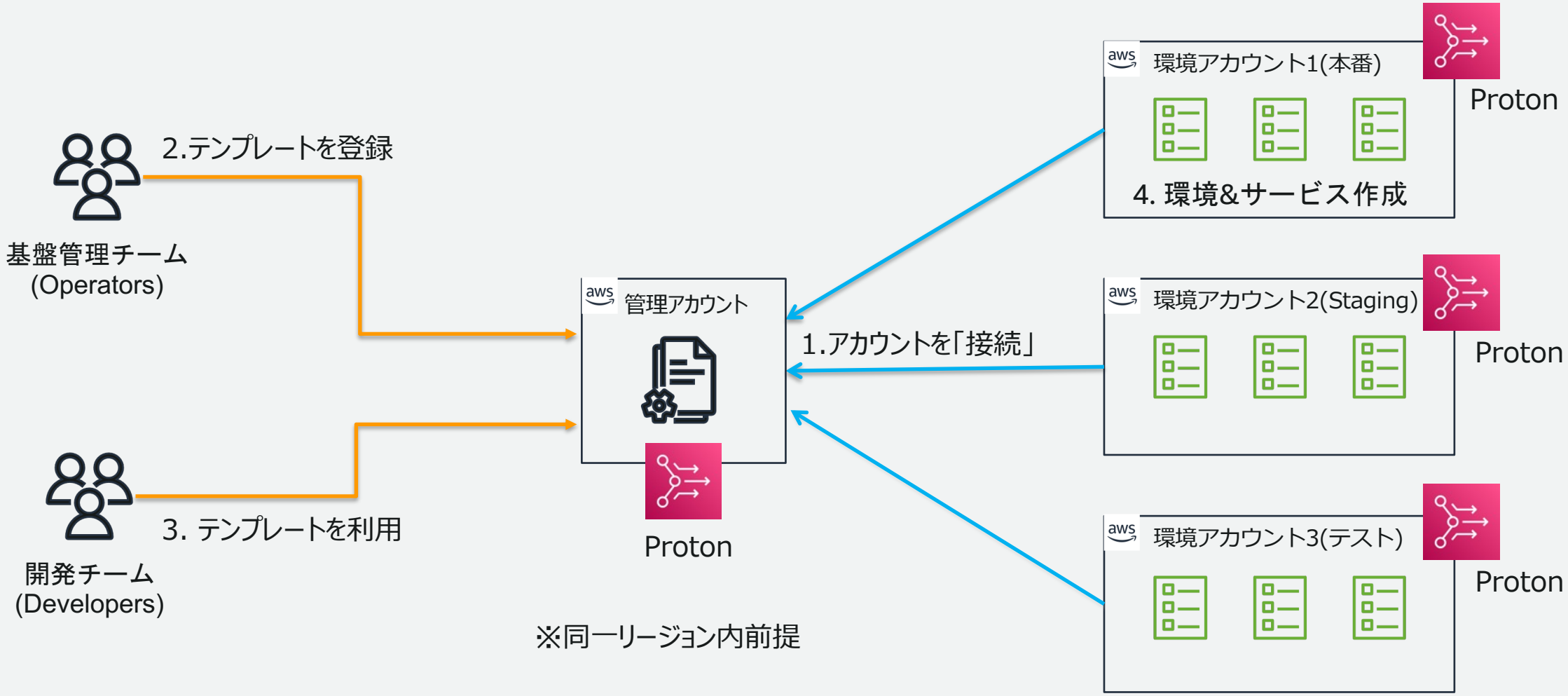
環境とサービス：アプリケーションへのマッピング例

Web Service : Container (w/pipeline)

Web Service: Serverless (w/pipeline)

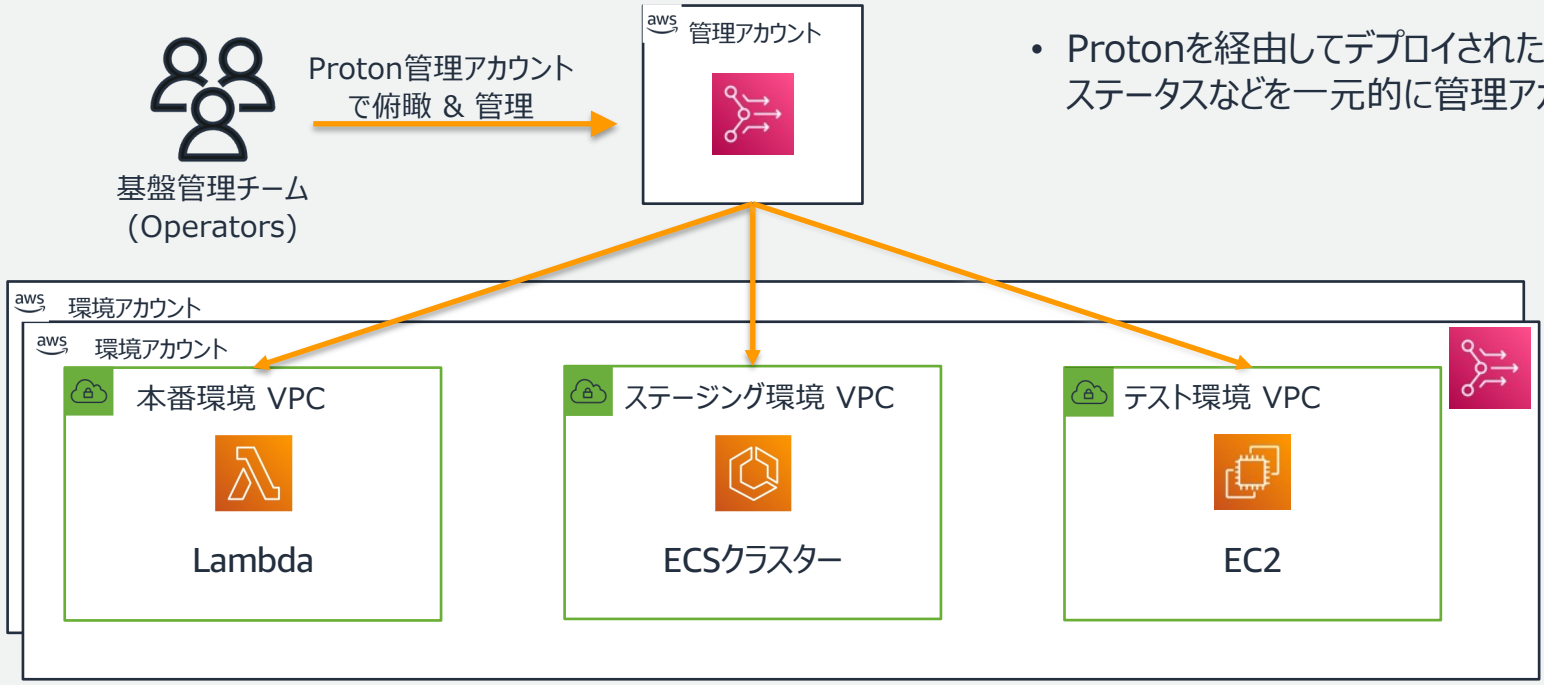


マルチアカウントの考え方



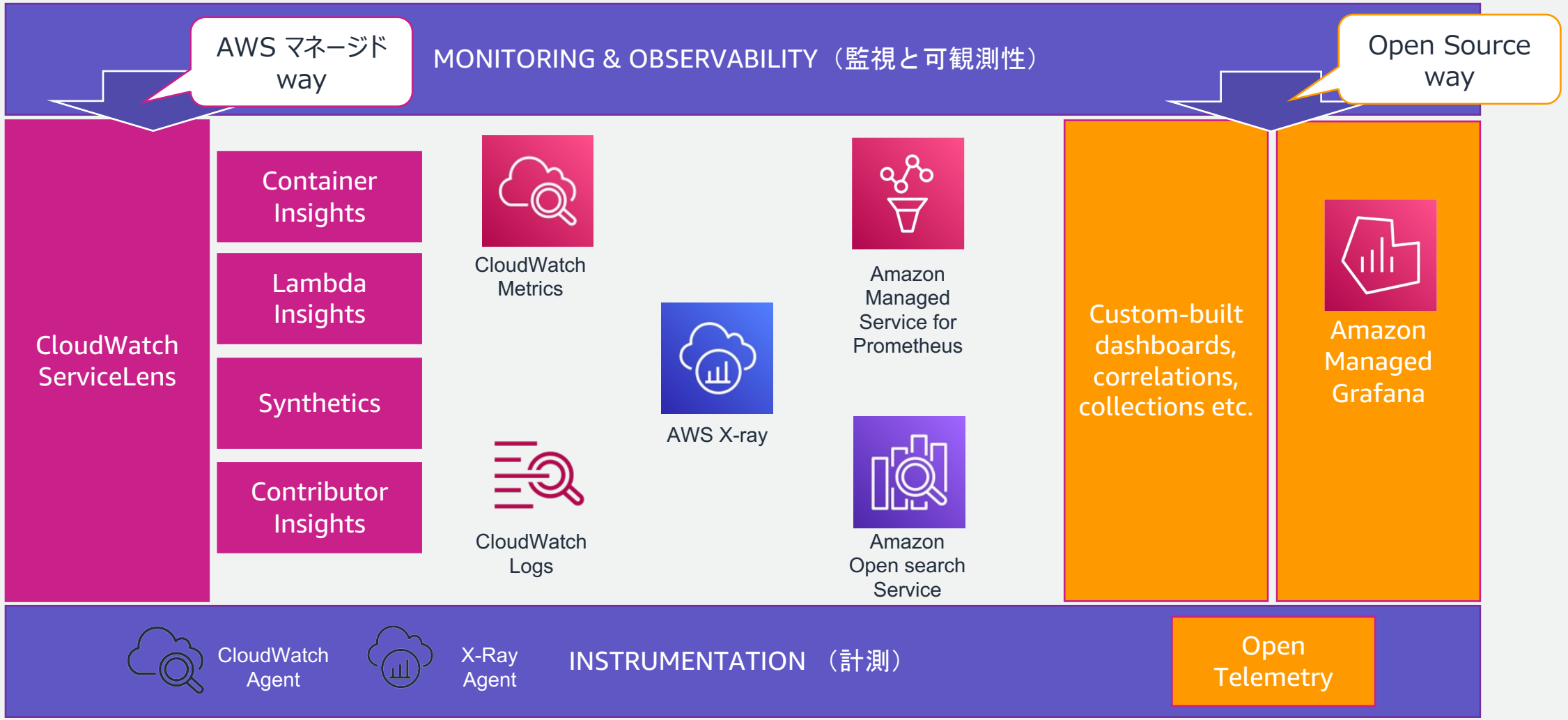
統制の考え方

開発者によってデプロイされるコンテナアプリケーションやサーバレスアプリケーションの環境を俯瞰的に把握



- Protonを経由してデプロイされたテンプレートのバージョン、ステータスなどを一元的に管理アカウントから管理できる

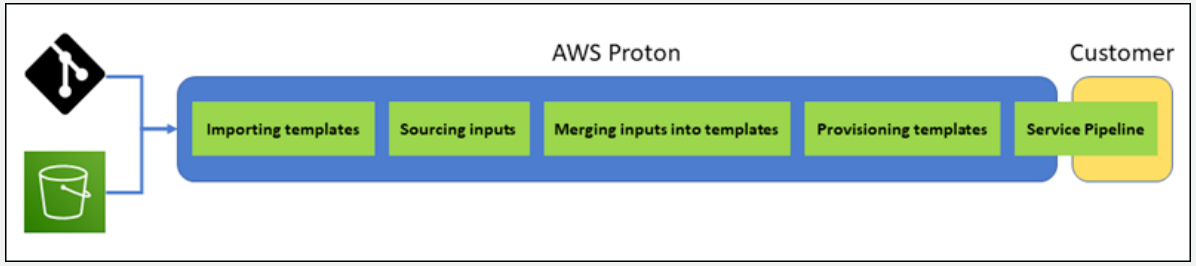
監視と可観測の考え方



Infrastructure Provisioningの考え方

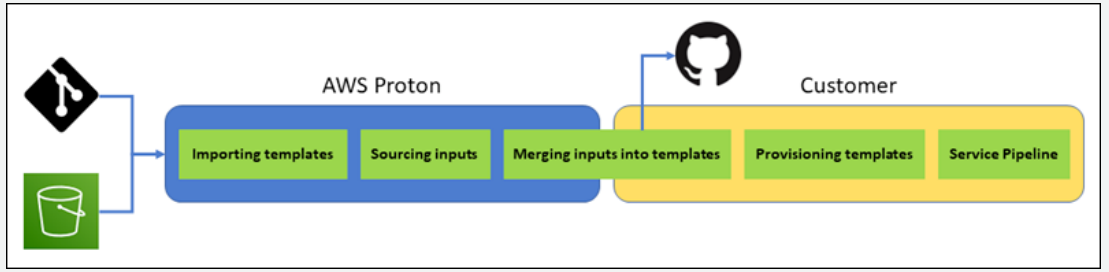
AWS-managed Provisioning using CloudFormation

- Proton 自信がprovisioning 実施



Self-managed Provisioning using Terraform

- Proton がリポシトリーに対し pull request (PR) を発行し、ユーザーの基盤デプロイ仕組みが provisioning 実施



- Provisioning方法は環境レベルで定義。Service Instanceの環境レベルの定義に従じる。
- 開発者はProvisioning方法については意識不要。Service Provisioningに影響なし。

Workshop環境へのアクセス確認

環境とアクセス方法

https://proton05.awsapps.com/start/

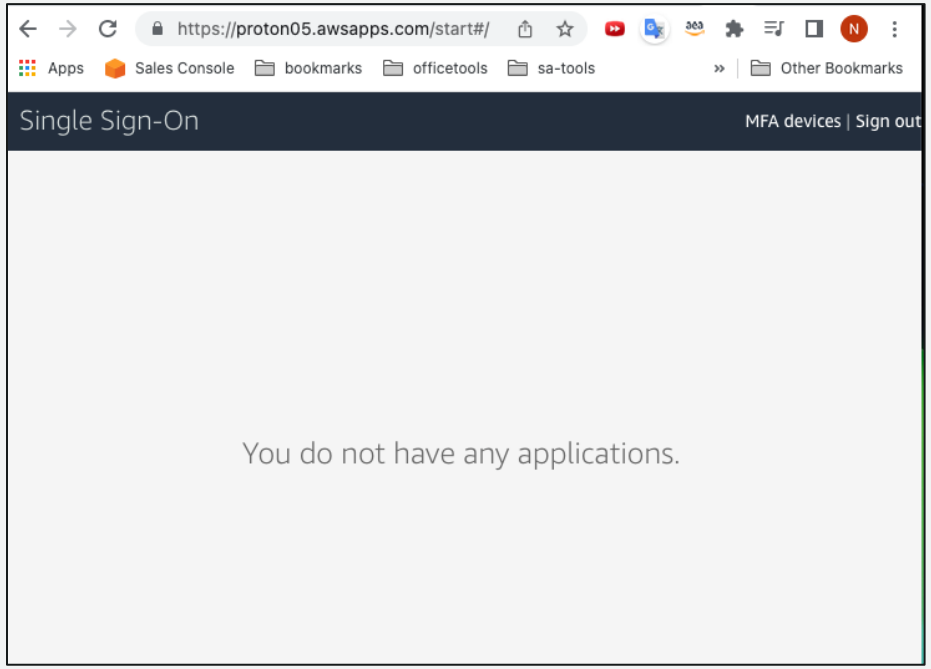
https://github.com/login

Sr.	User ID	AWS Password	GitHub Password	GitHub Passcode
1	nomura-user01			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
2	nomura-user02			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
3	nomura-user03			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
4	nomura-user04			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
5	nomura-user05			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
6	nomura-user06			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
7	nomura-user07			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

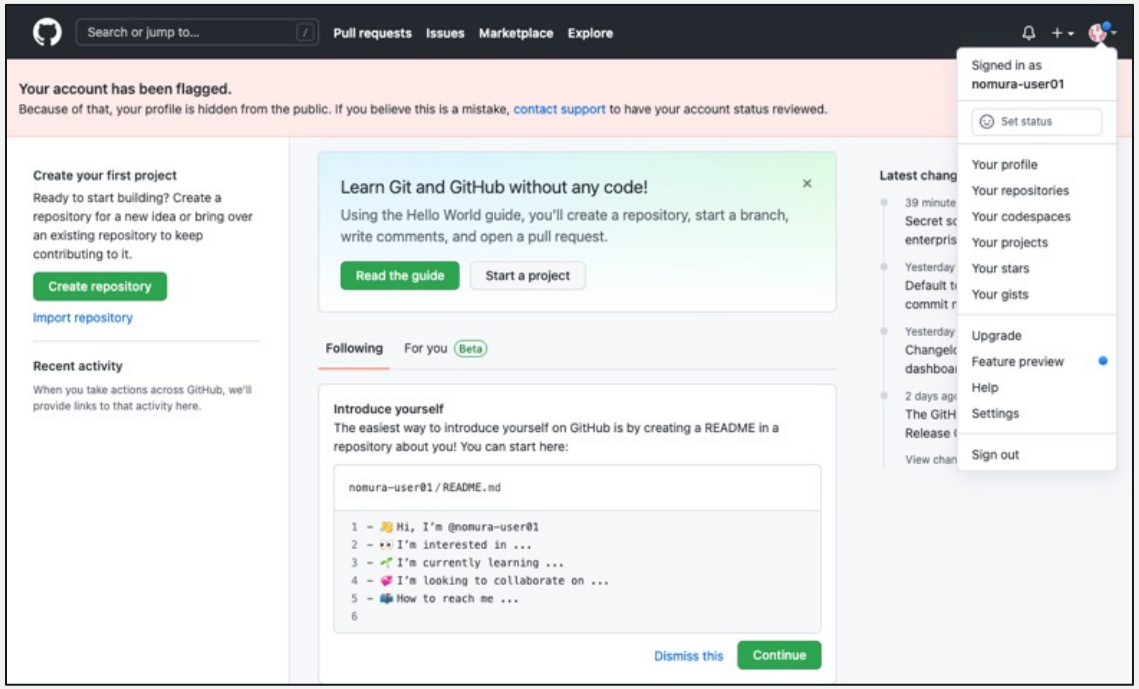
セッション中に共有

想定画面

AWS SSO



GitHub



Q&A
+
Discussion

Discussion

- 基盤 : Nomura AWS ? Cloud Native基盤 ?
- アプリケーション
- リポジトリ
- CI/CD

Thank you!

Appendices

UI ベースのDemo

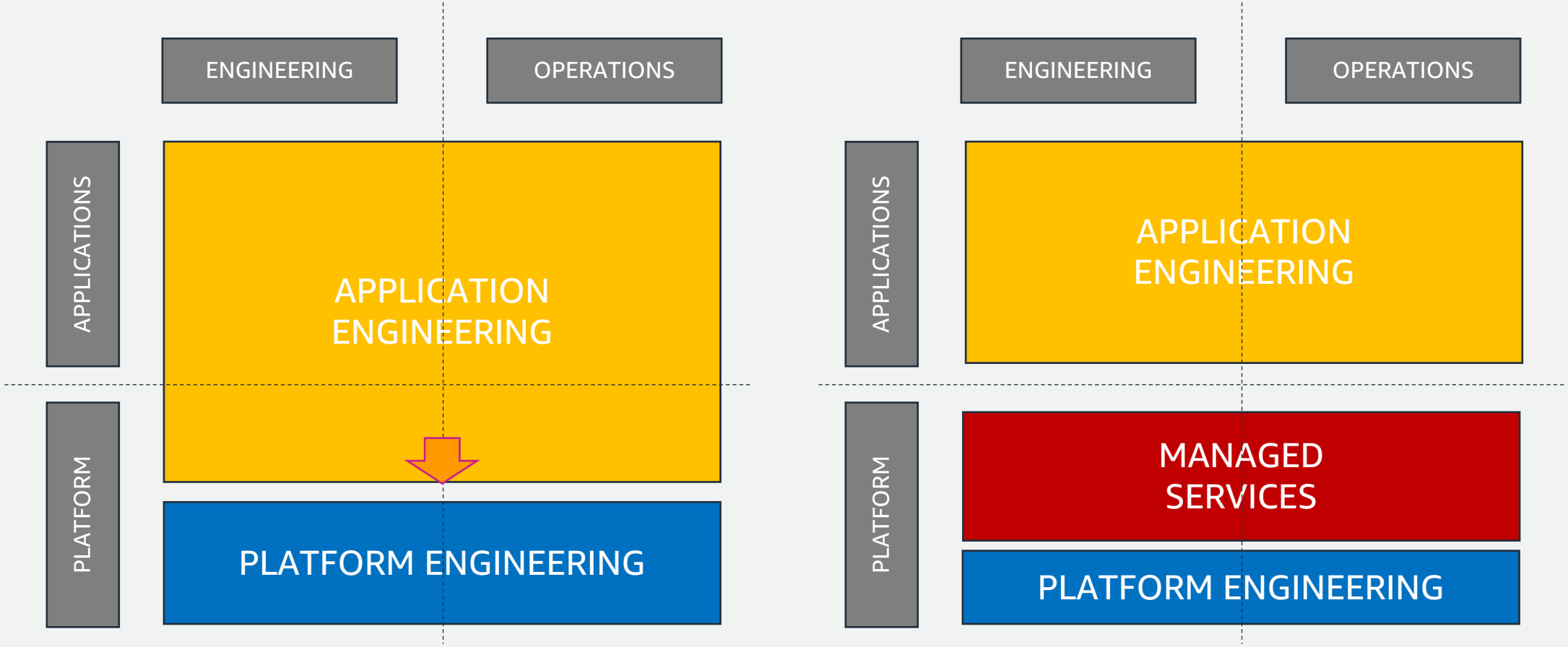
- https://www.youtube.com/watch?v=7D0k4KuHOaA

開発と運用、アプリとインフラの関係性

Well-Architected Framework「運用における優秀性の柱」にはこの観点でのベストプラクティスが含まれている

開発と運用、アプリとインフラの関係性

主な方向性、「You Build It, You Run It」を前提にすると…



AWS Proton 作業のイメージ