Kubernetes on Premise から EKS on Fargateへ

株式会社ぐるなび　開発部 EngineeringS Unit10
楠本 忠嗣（Engineering Lead）
竹原 駿平（Tech Lead）

ぐるなび

アジェンダ

アジェンダ

- 自己紹介
- 概要
  - きっかけ
  - EKS on Fargateを選んだ理由
  - 非エンジニアとの対話
- Tech & Knowledge
  - Logging
  - AWS Load Balancer Controller
  - Fargate Profile
  - Fargate Pod
  - Cluster Building
- 結果

自己紹介

自己紹介（楠本）

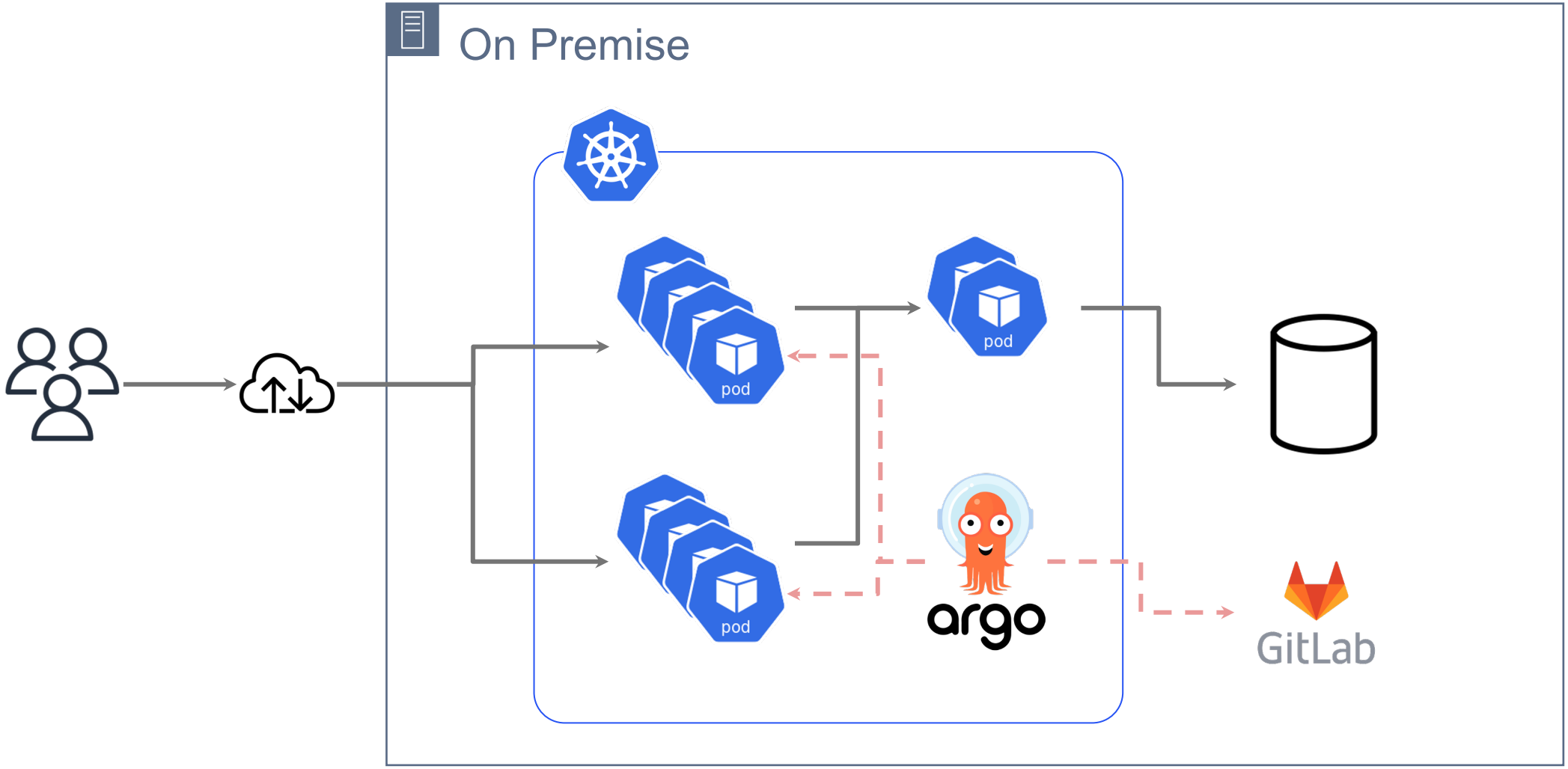
- 楠本　忠嗣
  - 開発部 EngineeringS Unit10
  - Web Application Engineer
  - Engineering Lead
- 技術
  - Nodejs, Typescript, React
  - Kubernetes
- 興味
  - Golang, GraphQL

自己紹介（竹原）

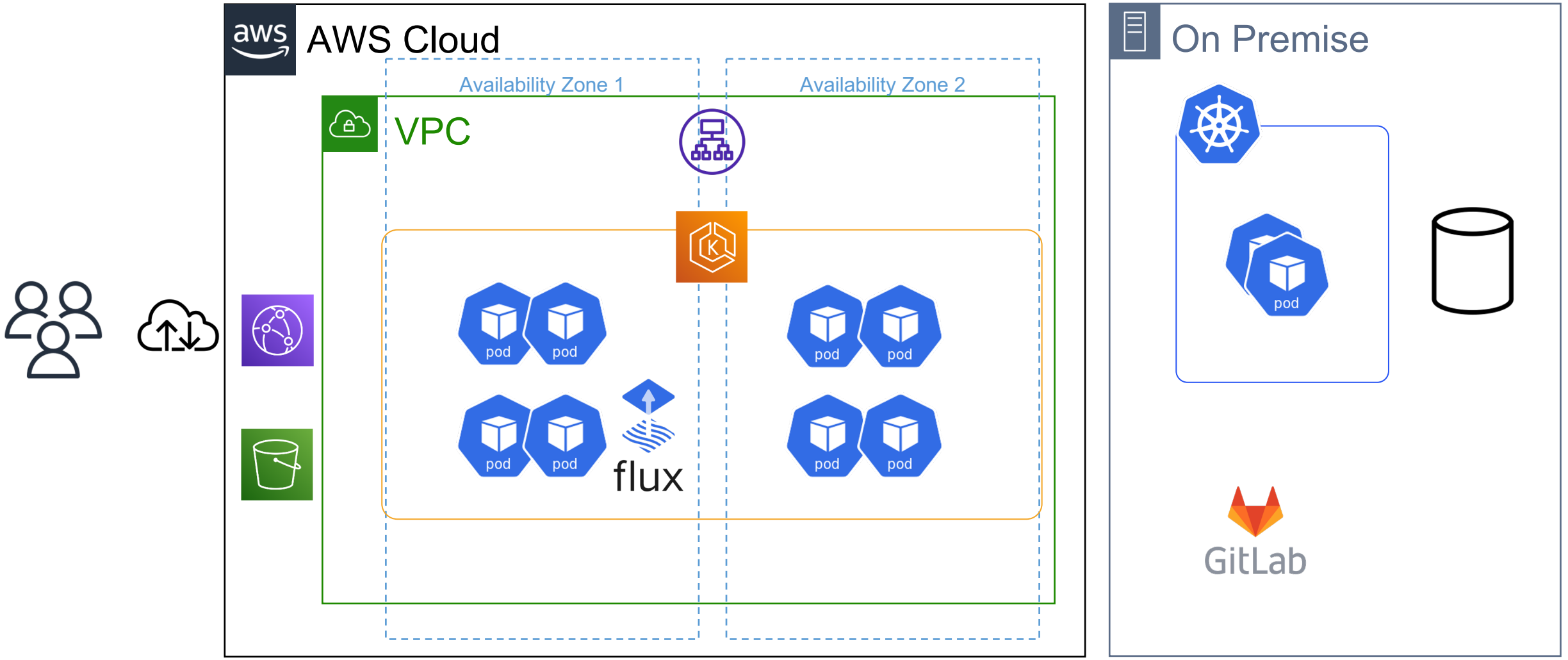
- 竹原　駿平
  - 開発部 EngineeringS Unit10
  - Web Application Engineer
  - Tech Lead
- 技術
  - Node.js, Typescript, Kubernetes
  - DevOps, Rust

概要

概要



概要


AWS Cloud
Availability Zone 1
Availability Zone 2
VPC
pod
pod
pod
pod
pod
pod
pod
pod
flux
On Premise
pod
GitLab

きっかけ

静的コンテンツの新規追加プロジェクト

静的コンテンツの新規追加プロジェクト

サービスの特徴として**海外ユーザー**がターゲット

- 低レイテンシーでコンテンツ配信したい
- AWSクラウドの利用促進

サービスの特徴として**海外ユーザー**がターゲット

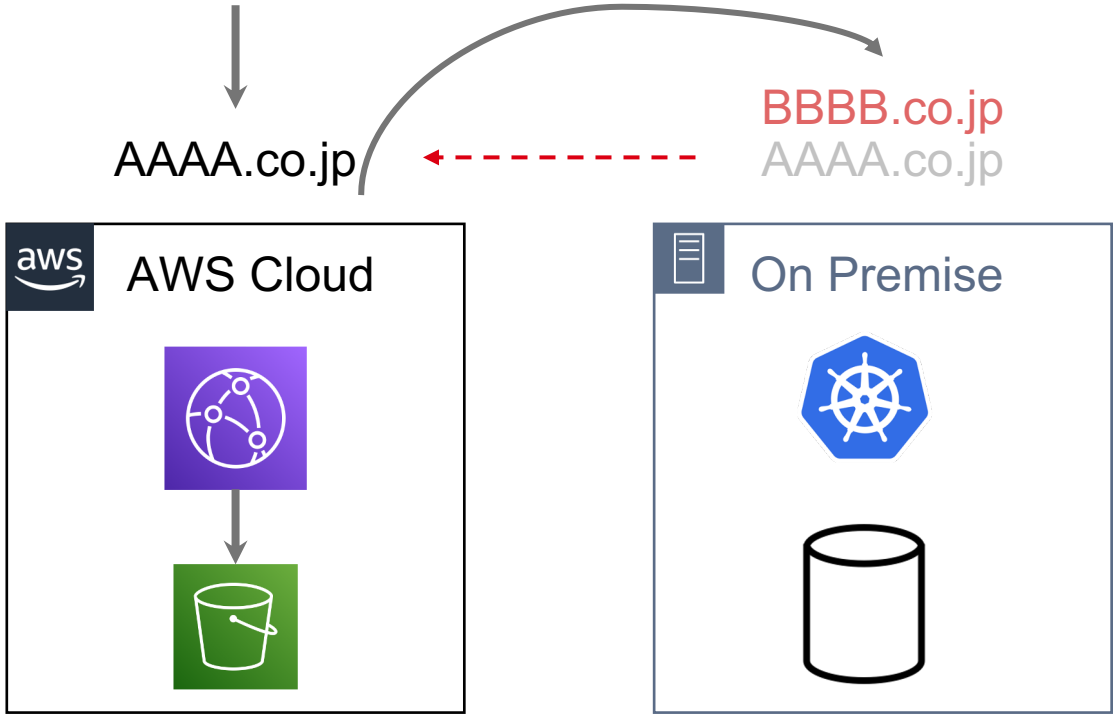
低レイテンシーでコンテンツ配信したい

AWSクラウドの利用促進

CloudFront

きっかけ

ネットワークをシンプルにしたい



コストの明確化、リソースの柔軟性

- インフラコストを明確化したい
- リソース調整コストを無くしたい
- リソース調達を柔軟にしたい

EKS on Fargateを選んだ理由

EKS on Fargateを選んだ理由

ECS or EKS

Amazon Elastic Container Service (Amazon ECS)

Amazon Elastic Kubernetes Service (Amazon EKS)

EKS on Fargateを選んだ理由

ECS or EKS

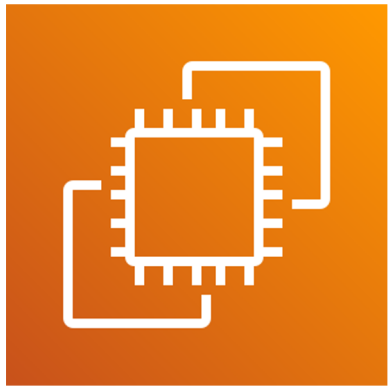
Amazon Elastic Container Service (Amazon ECS)

Amazon Elastic Kubernetes Service (Amazon EKS)

オンプレKubernetesの実績

EKS on Fargateを選んだ理由

EC2 or Fargate

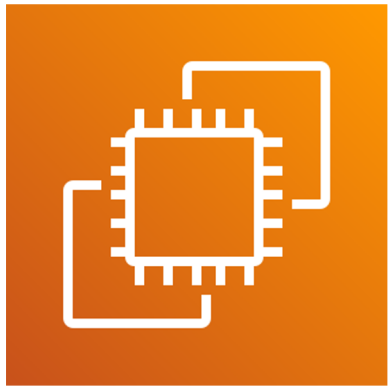


Amazon Elastic Compute Cloud (Amazon EC2)

AWS Fargate

EKS on Fargateを選んだ理由

EC2 or Fargate



Amazon Elastic Compute Cloud (Amazon EC2)

AWS Fargate

サーバ管理しない

EKS on Fargateを選んだ理由

EKS on Fargate

Amazon Elastic Kubernetes Service (Amazon EKS)

AWS Fargate

- Kubernetesの資産を活かせた
- アプリケーション開発に集中できた
- 予算内に収まった

非エンジニアとの対話

非エンジニアとの対話

コスト・予算

期間・期日

スコープ

品質

非エンジニアとの対話

コスト・予算

コストメリットを伝える

期間・期日

スコープ

品質

非エンジニアとの対話

コスト・予算

期間・期日

スコープ

期日を調整できないか相談する

品質

非エンジニアとの対話

コスト・予算

期間・期日

スコープ

品質

移行による保守性・拡張性・可用性を伝える

非エンジニアとの対話

コスト・予算

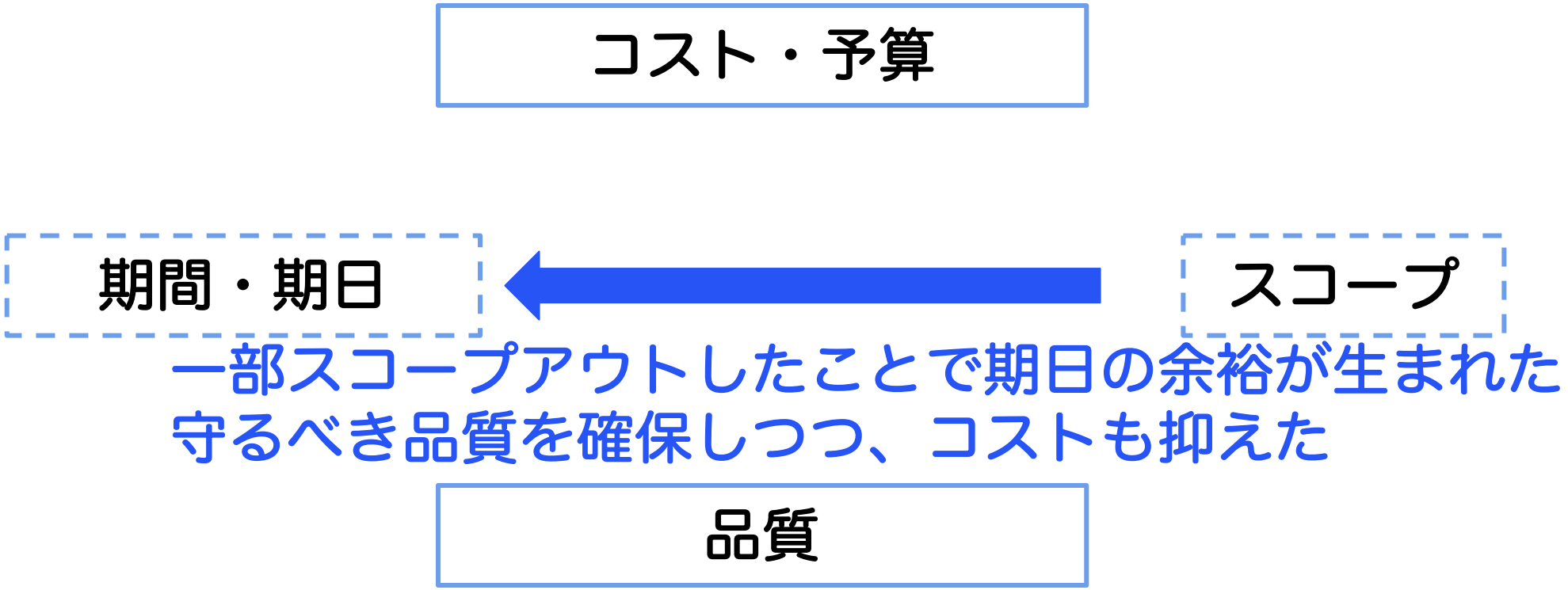
期間・期日

スコープ

スコープアウト、フェーズ分けを相談する

品質

非エンジニアとの対話



一部スコープアウトしたことで期日の余裕が生まれた
守るべき品質を確保しつつ、コストも抑えた

Logging

EKS on Fargateでのロギング

1. DeamonSetを使う方法

2. アプリケーションで独自実装する方法

3. Sidecarコンテナを使う方法

EKS on Fargateでのロギング

~~1. DeamonSetを使う方法~~
→ FargateではDeamonSetは利用できない

~~2. アプリケーションで独自実装する方法~~
→ アプリの改修が必要。OSSや3rd party製では出来ない。

3. Sidecarコンテナを使う方法
→ 以前のFargateでは実質これ１択だったが・・・

参考: https://kubernetes.io/docs/concepts/cluster-administration/logging/

Amazon Web Services ブログ

Fluent Bit for Amazon EKS on AWS Fargate をリリース

by Kazunori Harada | on 17 DEC 2020 | in Amazon CloudWatch, Amazon Elastic Kubernetes Service, Amazon Elasticsearch Service, Amazon Kinesis, Announcements, AWS Fargate, AWS Re:Invent, Containers | Permalink | Share

本投稿は、Akshay Ram, Prithvi Ramesh, Michael Hausenblas による寄稿を翻訳したものです。

Container roadmap 上の issue 701 では、 EKS on Fargate 利用時の CNCF Fluent Bit を利用したログルーターのサポートについて議論していました。このブログ記事では、EKS on Fargete利用時におけるいくつかの設定ステップによってCloudWatch へ直接ログを送信する事が出来る新しい機能とそれを利用する流れをみていきましょう。

出典: https://aws.amazon.com/jp/blogs/news/fluent-bit-for-amazon-eks-on-aws-fargate-is-here-jp/

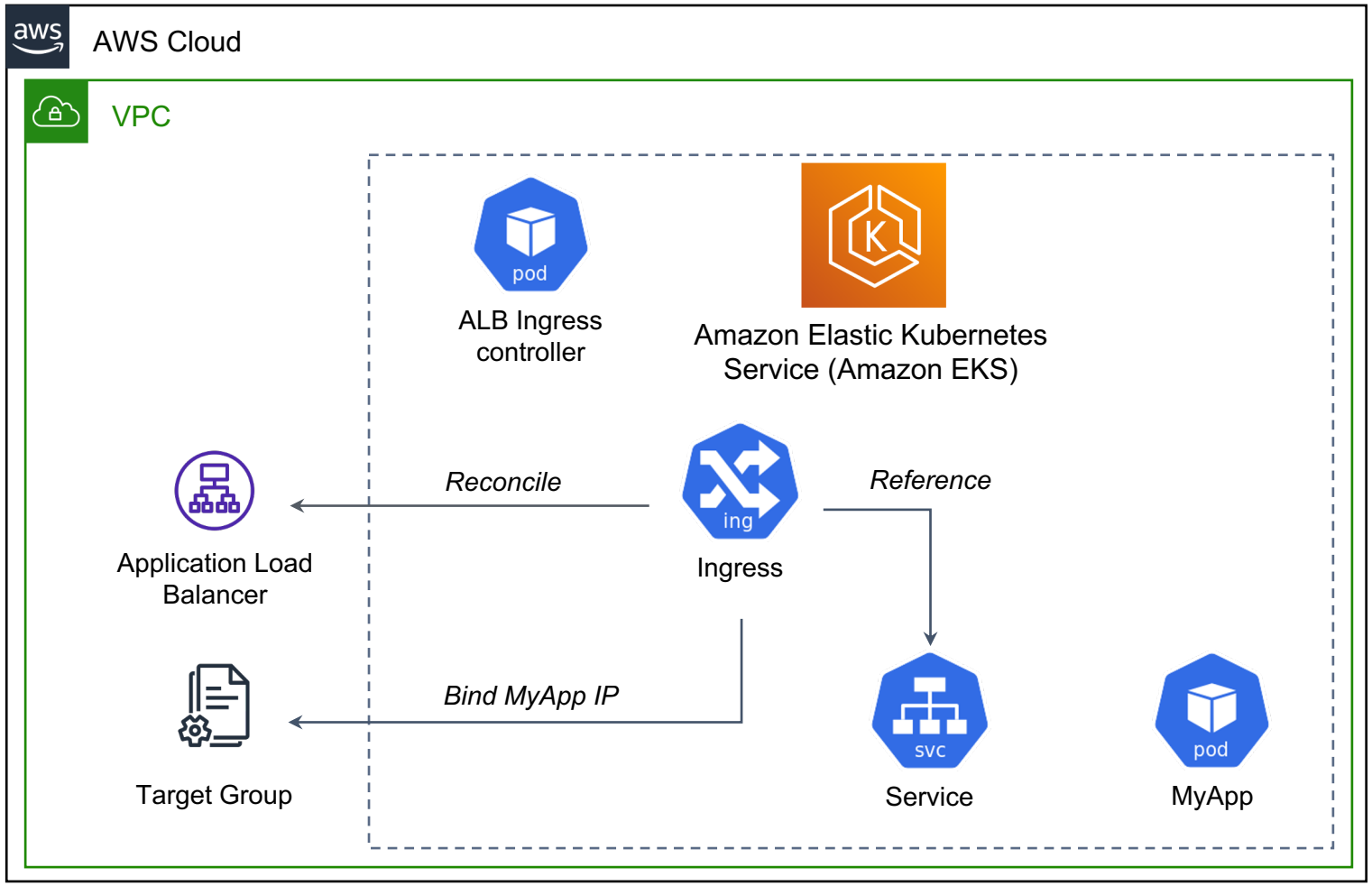
Fluent Bit for Amazon EKS on AWS Fargate

- Sidecarコンテナ不要
- ConfigMapに設定を書くだけでログ集約できるようになった

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: aws-logging
  namespace: aws-observability
data:
  output.conf: |
    [OUTPUT]
        Name cloudwatch_logs
        Match   *
        region ap-northeast-1
        log_group_name fluent-bit-cloudwatch
        log_stream_prefix from-fluent-bit-
        auto_create_group On
```

AWS Load Balancer Controller

ALB Ingress Controller (旧バージョン) for EKS on Fargate



・Ingress 毎に ALB が作成される

・Service 毎に TargetGroup が作成される

・Pod 毎に IPアドレスが Target として登録される

AWS Load Balancer Controller の紹介

投稿日: Oct 23, 2020

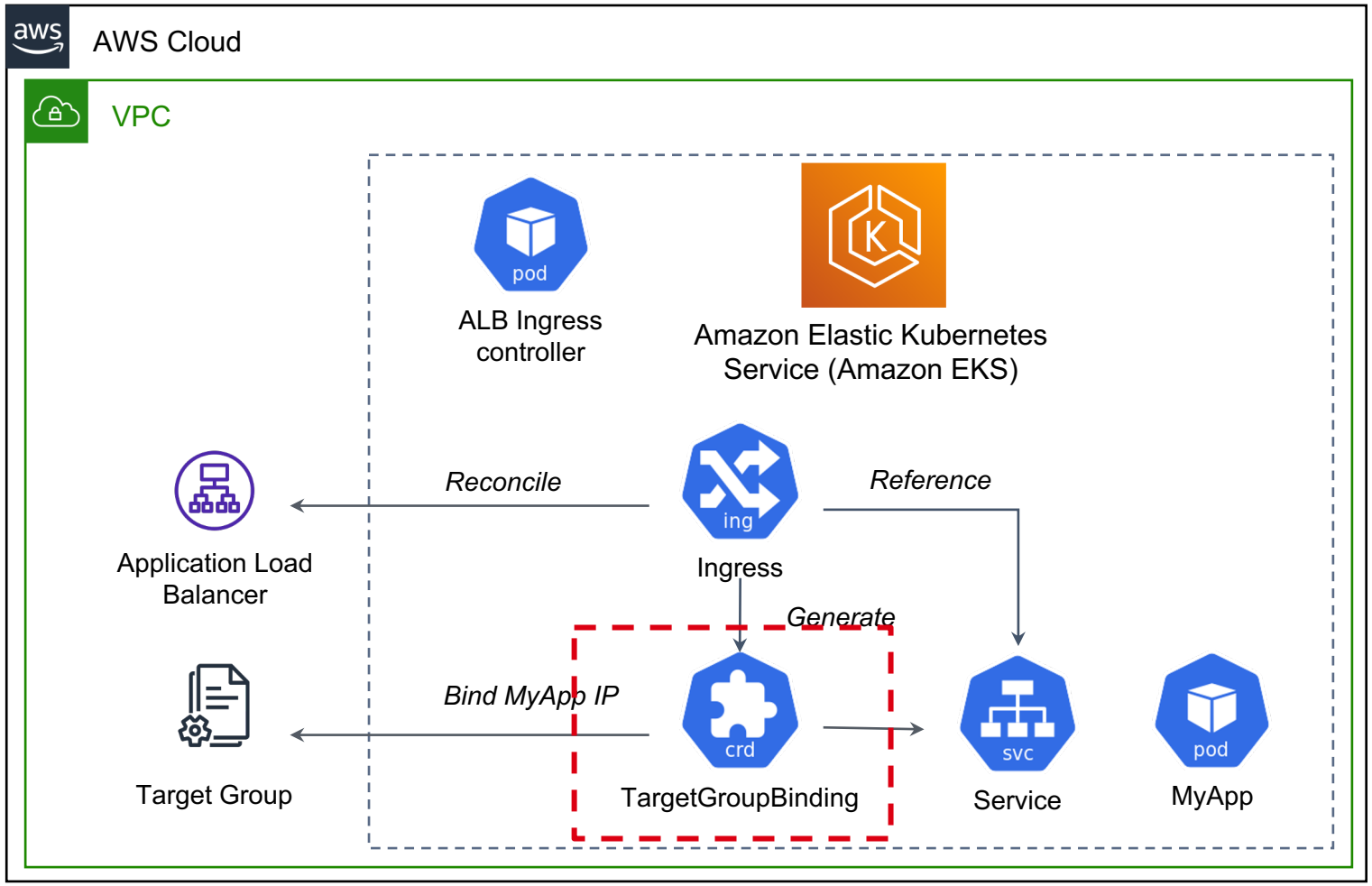
ALB Ingress Controller は AWS Load Balancer Controller になり、Application Load Balancer と Network Load Balancer のどちらもサポートしています。新しいコントローラーにより、Kubernetes クラスター内の複数のアプリケーション間で Application Load Balancer を共有したり、Network Load Balancer を使用して AWS Fargate で実行されているポッドをターゲットにすることで、操作を簡素化し、コストを節約できます。

出典: https://aws.amazon.com/jp/about-aws/whats-new/2020/10/introducing-aws-load-balancer-controller/

- Target Group Binding
- Ingress Group
- Network Load Balancer IP モード

- **Target Group Binding**
- Ingress Group
- Network Load Balancer IP モード

Target Group Binding

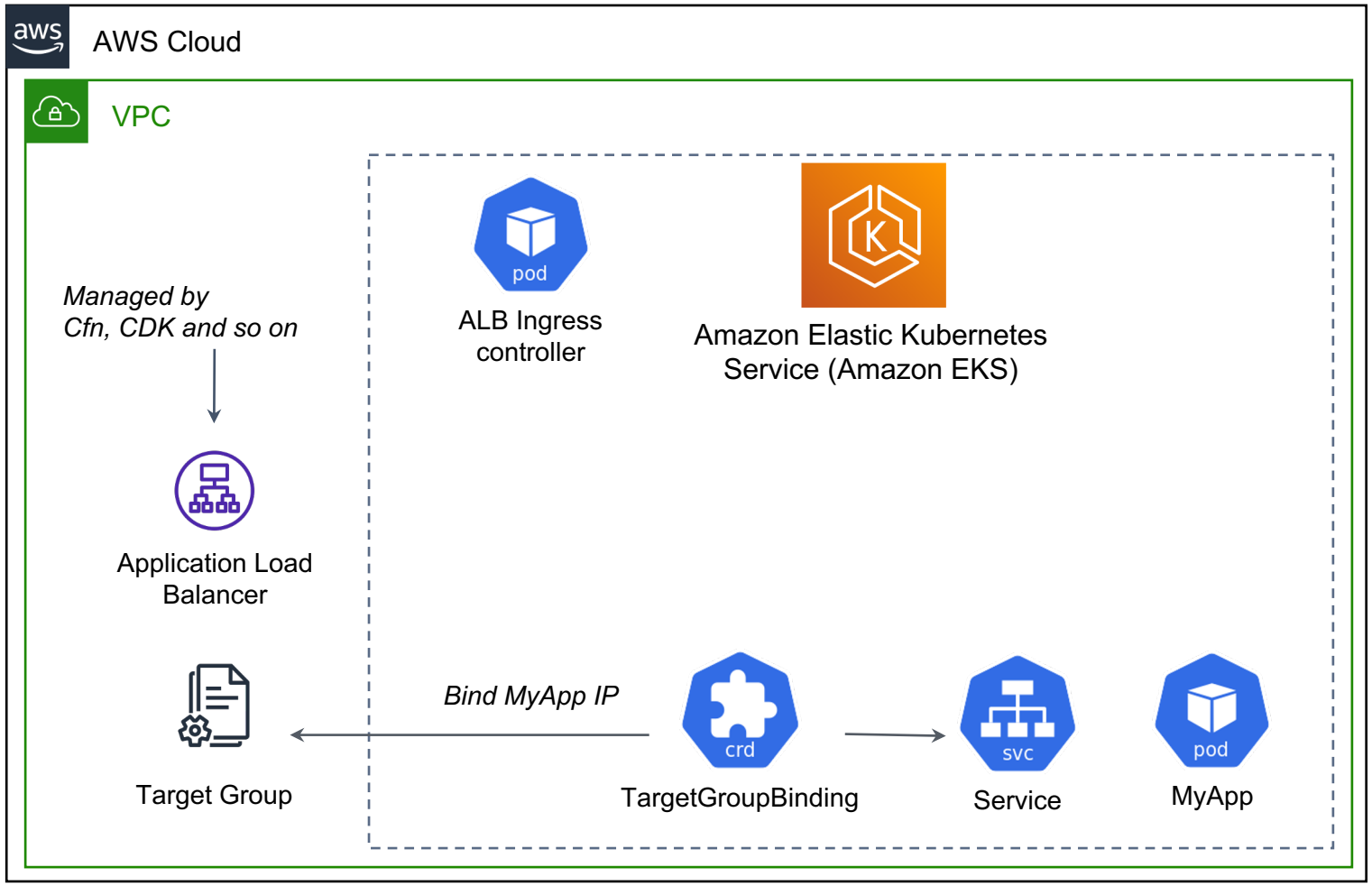


新しい CRD

Target Group Binding (TGB)

Target Groupは
TGB によって間接的に同期されるようになった

Target Group Binding



TGB だけで運用可能

ALB を別の方法で管理できるようになった

- 既存 ALB への統合
- 他サービスとの連携

Target Group Binding

```yaml
apiVersion: elbv2.k8s.aws/v1beta1
kind: TargetGroupBinding
metadata:
  name: api
spec:
  networking:
    ingress:
    - from:
      - securityGroup:
          groupID: sg-0123456789abcdef1
      ports:
      - protocol: TCP
  serviceRef:
    name: api
    port: 3001
  targetGroupARN: arn:aws:elasticloadbalancing:ap-northeast-1:012345678901:targetgroup/my-tg/0123456789abcdef
  targetType: ip
```

Service と Target Group の結びつきを記述

on Fargate 特有のハマりどころがある

- VPC ID と Region を明示的に指定しないとエラーになるので注意

```yaml
apiVersion: helm.toolkit.fluxcd.io/v2beta1
kind: HelmRelease
metadata:
  name: aws-load-balancer-controller
  namespace: kube-system
spec:
  interval: 5m
  chart:
    spec:
      chart: aws-load-balancer-controller
      version: 1.1.3
      sourceRef:
        kind: HelmRepository
        name: aws-load-balancer-controller
        namespace: flux-system
      interval: 1m
  values:
    clusterName: my-cluster
    vpcId: vpc-0123456789abcdef0
    region: ap-northeast-1
    serviceAccount:
      annotations:
        eks.amazonaws.com/role-arn: arn:aws:iam::012345678901:role/MyAWSLoadBalancerControllerRole
```

これが
on Fargate
では必須

Fargate Profile

Fargate Profile Overview

EKS のデータプレーン
- Amazon EC2
- AWS Fargate

Pod を Fargate で起動させるための設定が
Fargate Profile

```
FargateDefault:
  Type: AWS::EKS::FargateProfile
  Properties:
    ClusterName: !Ref EKS
    FargateProfileName: 'fp-default'
    PodExecutionRoleArn: !ImportValue EKSFargatePodRole
    Selectors:
      - Namespace: 'default'
      - Namespace: 'kube-system'
    Subnets:
      - !Ref SubnetPrivateAzA
      - !Ref SubnetPrivateAzC
```

```
$ kubectl get nodes
NAME                                                STATUS   ROLES    AGE    VERSION
fargate-ip-172-22-189-125.ap-northeast-1.compute.internal   Ready    <none>   11d    v1.18.9-eks-866667
fargate-ip-172-22-189-149.ap-northeast-1.compute.internal   Ready    <none>   117d   v1.18.8-eks-7c9bda
fargate-ip-172-22-189-153.ap-northeast-1.compute.internal   Ready    <none>   11d    v1.18.9-eks-866667
fargate-ip-172-22-189-182.ap-northeast-1.compute.internal   Ready    <none>   11d    v1.18.9-eks-866667
fargate-ip-172-22-189-208.ap-northeast-1.compute.internal   Ready    <none>   11d    v1.18.9-eks-866667
fargate-ip-172-22-189-222.ap-northeast-1.compute.internal   Ready    <none>   117d   v1.18.8-eks-7c9bda
fargate-ip-172-22-189-25.ap-northeast-1.compute.internal    Ready    <none>   117d   v1.18.8-eks-7c9bda
fargate-ip-172-22-189-250.ap-northeast-1.compute.internal   Ready    <none>   11d    v1.18.9-eks-866667
fargate-ip-172-22-189-37.ap-northeast-1.compute.internal    Ready    <none>   117d   v1.18.8-eks-7c9bda
fargate-ip-172-22-189-45.ap-northeast-1.compute.internal    Ready    <none>   117d   v1.18.8-eks-7c9bda
fargate-ip-172-22-189-51.ap-northeast-1.compute.internal    Ready    <none>   105d   v1.18.8-eks-7c9bda
fargate-ip-172-22-189-65.ap-northeast-1.compute.internal    Ready    <none>   11d    v1.18.9-eks-866667
fargate-ip-172-22-189-72.ap-northeast-1.compute.internal    Ready    <none>   117d   v1.18.8-eks-7c9bda
fargate-ip-172-22-189-73.ap-northeast-1.compute.internal    Ready    <none>   11d    v1.18.9-eks-866667
```

- k8s 側から見ると 1 Pod につき 1 Node できている
- この Node は Fargate によって作られたもの

Fargate Profile の注意点

・作成後、設定を変更出来ない

・namespaceにワイルドカード等は利用出来ない

・1 EKS クラスタに付き 10 Fargate Profileまで

シングルテナントでの運用が良い

Fargate Profile の管理

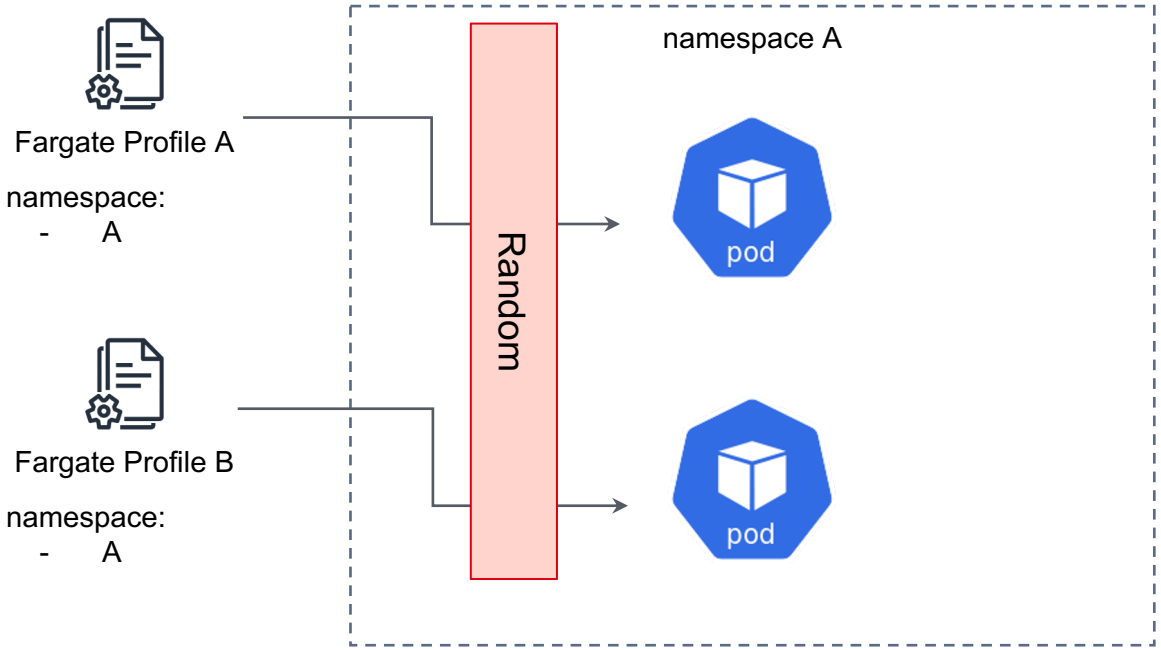
更新出来ない = シビアになる必要がある？

→ ダウンタイムなしで整理する方法がある

Fargate Profile の管理

複数の Fargate Profile が同じ namespace を指定

→ Pod 毎にランダムにどちらかが割当たる



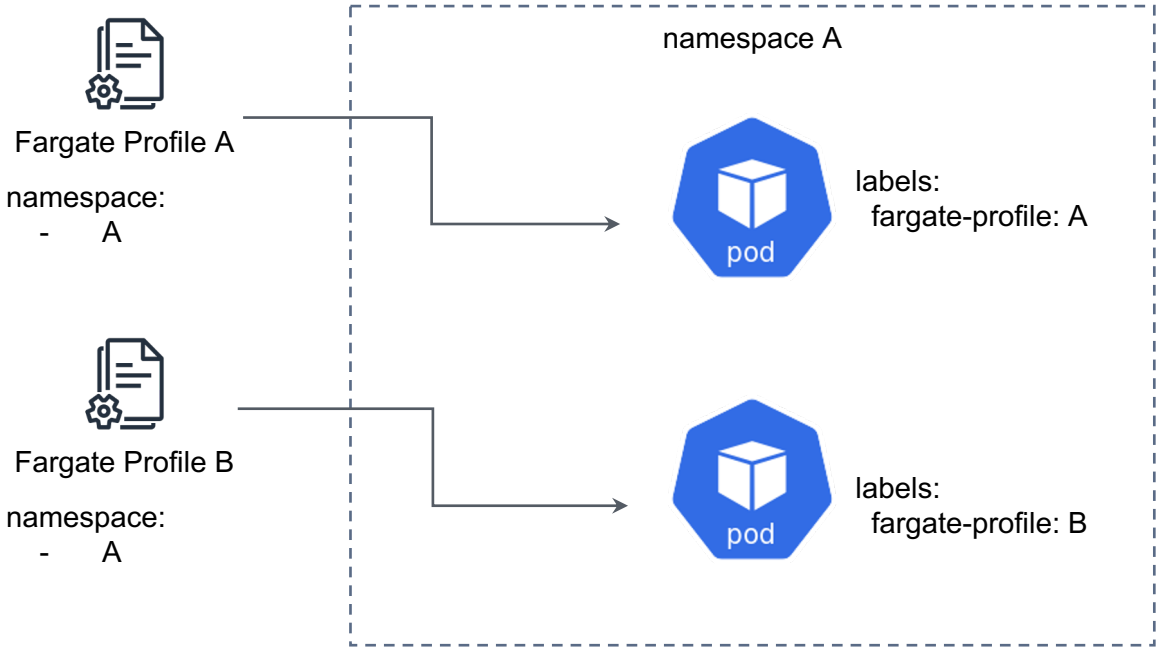
Pod に割りあたっている Fargate Profile はlabelで確認できる

```
  labels:
    eks.amazonaws.com/component: coredns
    eks.amazonaws.com/fargate-profile: fp-default
    k8s-app: kube-dns
    pod-template-hash: 74dfc597cf
```

Fargate Profile の管理

label で Fargate Profile を指定

→ 指定した Fargate Profile で起動する



これを使うとダウンタイムなしで Fargate Profile を整理できる

1. Fargate Profile を作成
2. label で新しい Fargate Profileを割り当てる
3. 古い Fargate Profileの削除

上限数まで Fargate Profile を作ってしまうと詰むので注意！

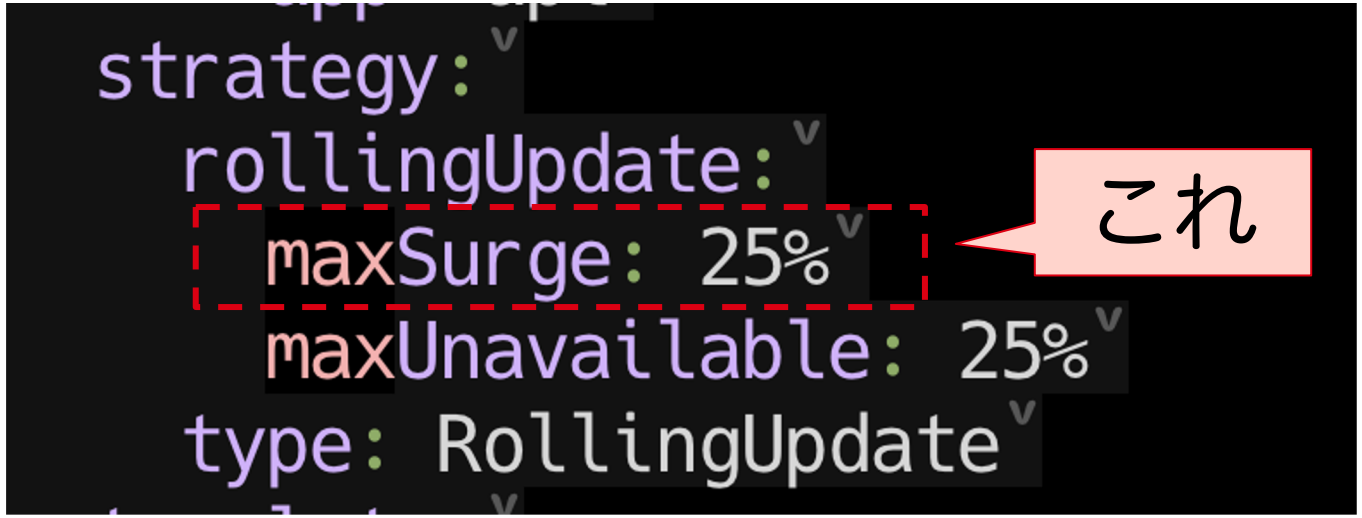
Fargate Pod

Fargate Pod

- 起動するまでの時間
- vCPU / Memory アロケーション
- 配置される AZ のコントロール

起動するまでの時間

- Fargate Pod は起動に時間がかかる (1 〜 2 分)
- Rolling Update 時に maxSurge をあげておくと良い



vCPU / Memory アロケーション

- 利用可能な組み合わせが決まっている
- 量 × 時間 で課金

vCPU 値	メモリの値
.25 vCPU	0.5 GB、1 GB、2 GB
.5 vCPU	1 GB、2 GB、3 GB、4 GB
1 vCPU	2 GB、3 GB、4 GB、5 GB、6 GB、7 GB、8 GB
2 vCPU	4 GB ～ 16 GB (1 GB のインクリメント)
4 vCPU	8 GB ～ 30 GB (1 GB のインクリメント)

出典: https://docs.aws.amazon.com/ja_jp/eks/latest/userguide/fargate-pod-configuration.html

AZ のコントロール

- 経験則では3レプリカ以上で multi AZ に分散してくれる
- コントロールしたいなら AZ 毎に Fargate Profile を分ける

```
FargateDefault:
  Type: AWS::EKS::FargateProfile
  Properties:
    ClusterName: !Ref EKS
    FargateProfileName: 'fp-default'
    PodExecutionRoleArn: !ImportValue EKSFargatePodRole
    Selectors:
      - Namespace: 'default'
      - Namespace: 'kube-system'
    Subnets:
      - !Ref SubnetPrivateAzA
      - !Ref SubnetPrivateAzC
```

https://github.com/aws/containers-roadmap/issues/824

Cluster Building

どうやって EKS クラスタを構築するか？

- Amazon EKS API
- Cloud Formation
- eksctl
- CDK

どうやって EKS クラスタを構築するか？

- Amazon EKS API
- Cloud Formation
- eksctl
- CDK

EKS クラスタ構築には複雑なステップが必要

- Fargate Profile の作成
- ConfigMap や Deployment の変更
- IAM Roles for Service Accounts のセットアップ
- ミドルウェア のインストール

現状、すべてをカバーしているのは CDK のみ

結果

移行期間

約1ヶ月（4人）

コスト変化

約 1/2

サーバ費用が明確に
リソース調達が柔軟に
リソース調整も不要に

Let’s Try Containers
And
Let’s Try EKS on Fargate

Thank you for your attention