\Orchestrating a brighter world NEC

～春のAWS コンテナ祭り with Amazon EKS～
EKS利用から運用まで

２０２０年3月２０日

NEC (日本電気株式会社)
宮下　昂明　(Takaaki Miyashita)

\Orchestrating a brighter world

未来に向かい、人が生きる、豊かに生きるために欠かせないもの。
それは「安全」「安心」「効率」「公平」という価値が実現された社会です。

NECは、ネットワーク技術とコンピューティング技術をあわせ持つ
類のないインテグレーターとしてリーダーシップを発揮し、
卓越した技術とさまざまな知見やアイデアを融合することで、
世界の国々や地域の人々と協奏しながら、
明るく希望に満ちた暮らしと社会を実現し、未来につなげていきます。

自己紹介

- 名前：宮下　昂明(みやした　たかあき)
- 所属：NEC (日本電気株式会社)
- 職種：インフラエンジニア７年目
  - １～５年目はDC内サーバー構築
  - ５年目以降はクラウド構築＆DC内サーバー保守…(笑)
- 保有資格
  - Solutions Architect Professional
  - Solutions Architect Associate
- 好きなAWSサービス

もちろんEKS！！！

最近嬉しかったこと

EKSクラスターの利用料金が
一気に半額になったこと！！！

単価	値下前料金（USD）	値下後料金（USD）
1時間	0.20	0.10
1日	4.80	2.40
1ヶ月	144.00	72.0

1ヶ月に72USD下がるのは本当に
現場としては嬉しいことです！！！
AWS様ありがとうございます！ (もっと安くしても…(笑))

目次

1．はじめに

はじめに

今までの私の経歴

- 入社からHW導入・OSインストール・SWインストール、各種設定など
  どっぷり、インフラ環境構築を実施してきました。
- 主にWeb向けシステムを担当して、アプリケーションサーバー、データベース
  メール中継など、様々なソフトウェアも触ってきました。

本日伝えたいこと

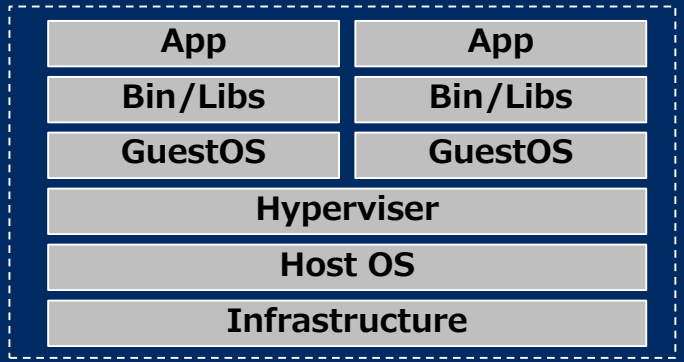
✓ **EKSを使ってどのような実装・運用に気を付けたかを**
インフラ目線で説明します！！！

2．マネージド型Kubernetesサービスの必要性

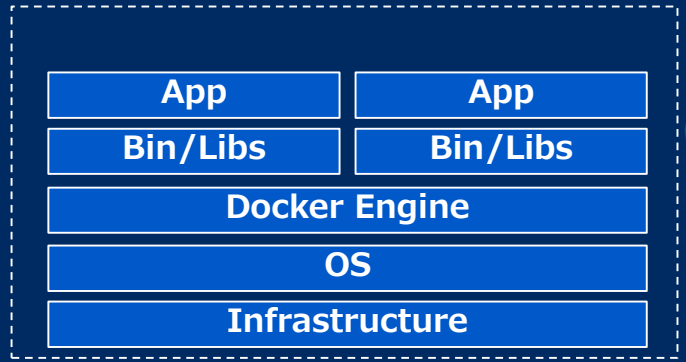
コンテナについて

OSのリソースを隔離して作成された仮想的なアプリ実行環境

- コンテナランタイムはDockerが主に利用されている。



ホストOS仮想化



コンテナ型仮想化

Dockerのメリット

- ✓ **ハードウェア・OSの共有化・可搬性が高い（Dockerfile/Docker Image）**
- ✓ **不変な環境（何度実行しても同等環境）**

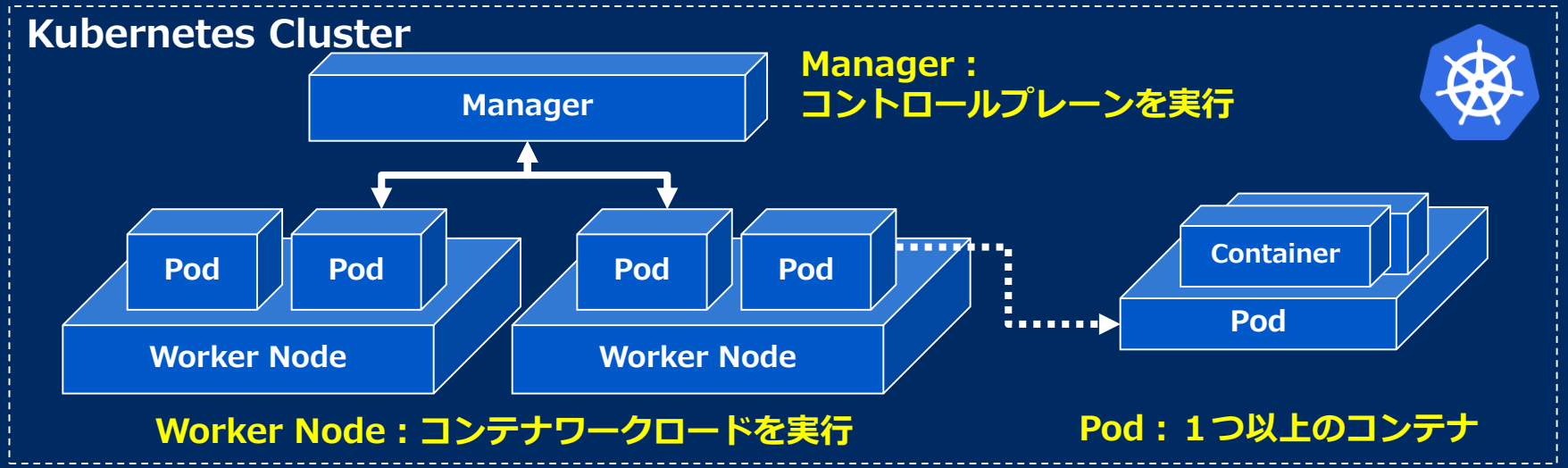
Dockerのみ運用のデメリット

- ✓ **ネットワークが煩雑となる**
- ✓ **セキュリティ、可用性、監視など様々な点を運用を考慮する必要あり**

コンテナオーケストレーションシステム

コンテナオーケストレーションでは、コンテナの実行や管理を制御する

- **Kubernetesがデファクトスタンダード**（Red Hat OpenShiftもコア部分はKubernetes）



コンテナオーケストレーションの役割

- ✓ **クラスタ管理**
- ✓ **コンテナ運用自動化/スケジューリング**
- ✓ **ロードバランシング**
- ✓ **Auto Scaling/Auto Healing**
- ✓ **コンテナの死活監視…など**

Kubernetesの課題

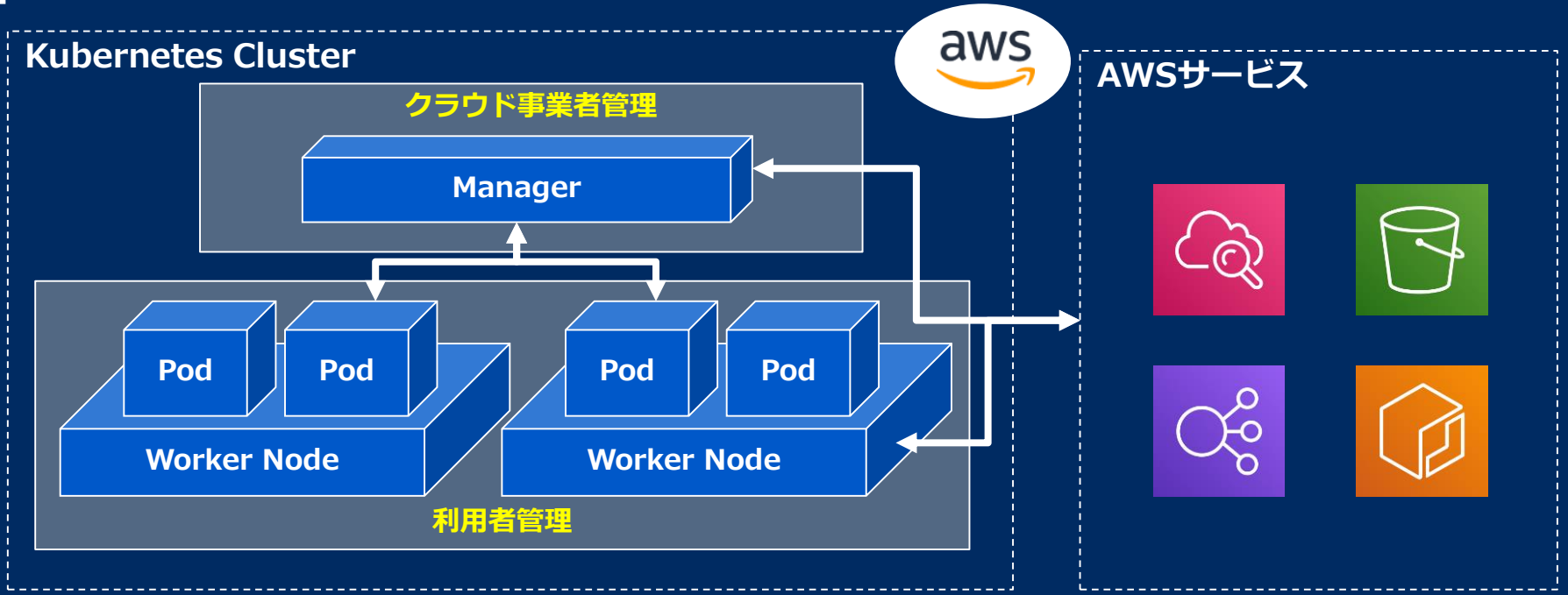
Kubernetes 自体の学習コストがそれなりに高い上に、自前で構築・運用する場合、考慮すべき事項が多数存在する。

インフラの設計(OS設定等) ★私は大好きです	Kubernetesの構築
可用性：冗長化（サーバー）	拡張性：スケーリング、 スケールアウト（サーバー）
セキュリティ （サーバー／コンテナ）	運用監視 （サーバー／コンテナ）
コンテナレジストリ	多々あるバージョンアップ運用 サポート

- ✓ 上記を考慮するのは時間もかかる！
- ✓ インフラ構築だけで、コンテナの良さが消えてしまう！

マネージド型Kubernetesサービス

- 管理コストの高い**Master についてクラウド事業者が管理**
- ユーザーのワークロードを実行する**Worker Nodeは利用者が管理**
- **様々なクラウドサービスと連携が可能**



３．私がEKSを利用するまでに至った理由

AWS・Kubernetes採用の理由

2018年頃、Webシステム構築を Kubernetesを用いる前提で検討

AWS採用の理由

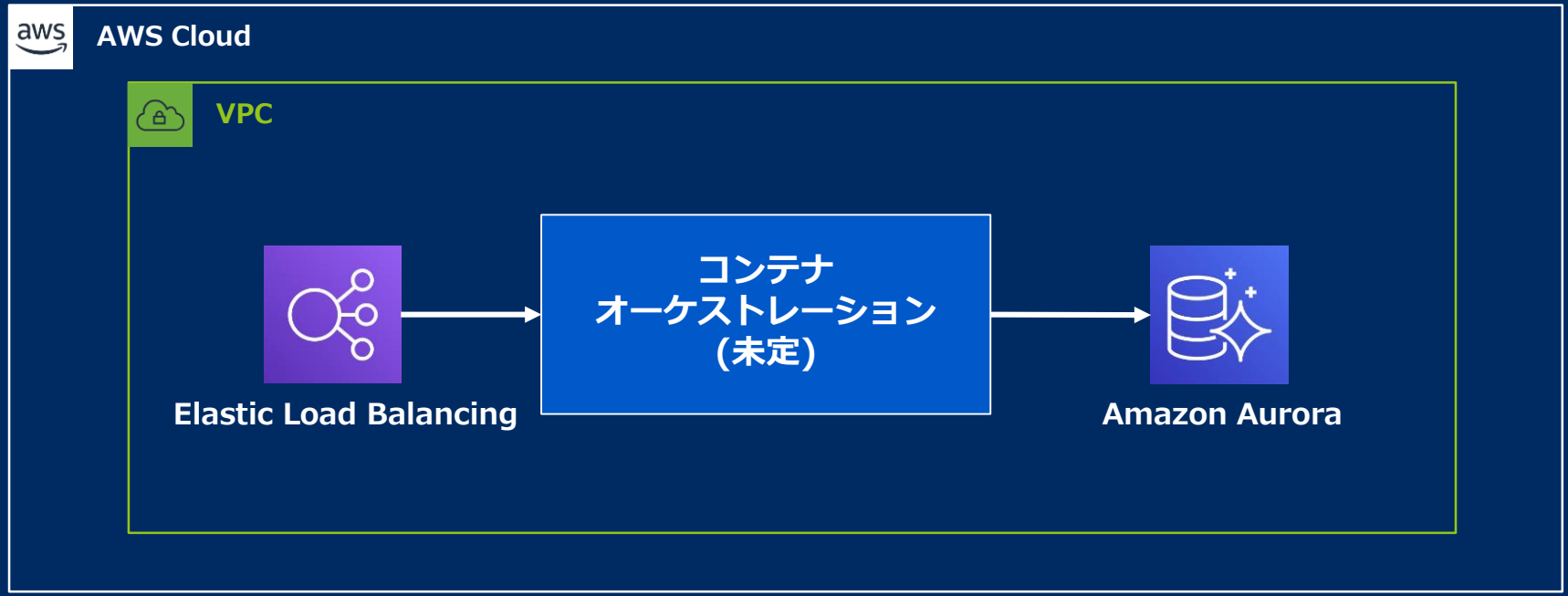
- 何より**実績**があること
- 様々なニーズに合わせたサービスを提供しており、**多様性**を担保できること

Kubernetes採用の理由

- 複数のWebサービスを提供予定で、**テナント分離したい**
- 多数の複数コンテナを扱うため、**マイクロサービス運用したい**
- **クラウド全体のコンテナ周りを一元管理したい**
- **コンテナオーケストレーションとしても可搬性を持ちたい**

コンテナオーケストレーション選定の苦悩

東京リージョンのEKSを使いたかったが、**当時EKSがまだ発表されておらず、コンテナオーケストレーションの選定がなかなか決まらず…。**



コンテナオーケストレーション選定の苦悩

Kubernetes on EC2構成案

- メリット
  - Kubernetesを利用できる
- デメリット
  - **とにかく構築が大変、マネージドプレーン部分、名前解決やストレージ、APIなど**
  - **とにかく運用が大変、ノードの増減運用やバージョンアップが多い**
  - **異常時の対応も大変、異常時はOSSコードを直接読んだり苦労することも想定された**

ECS on Fargate構成案（当時、こっちに変更しようした）

- メリット
  - とにかくシンプルでマネジメントコンソールで対応可能・AWS APIで統一
  - Dockerのセットアップ不要でCloudWatch・CloudTrailとの連携が強い
  - オートスケール・ALB対応・FargateなのでEC2を意識する必要なし
- デメリット
  - **ECS独自(AWS独自)のデプロイメント仕様**

EKS選定理由

2018年12月20日

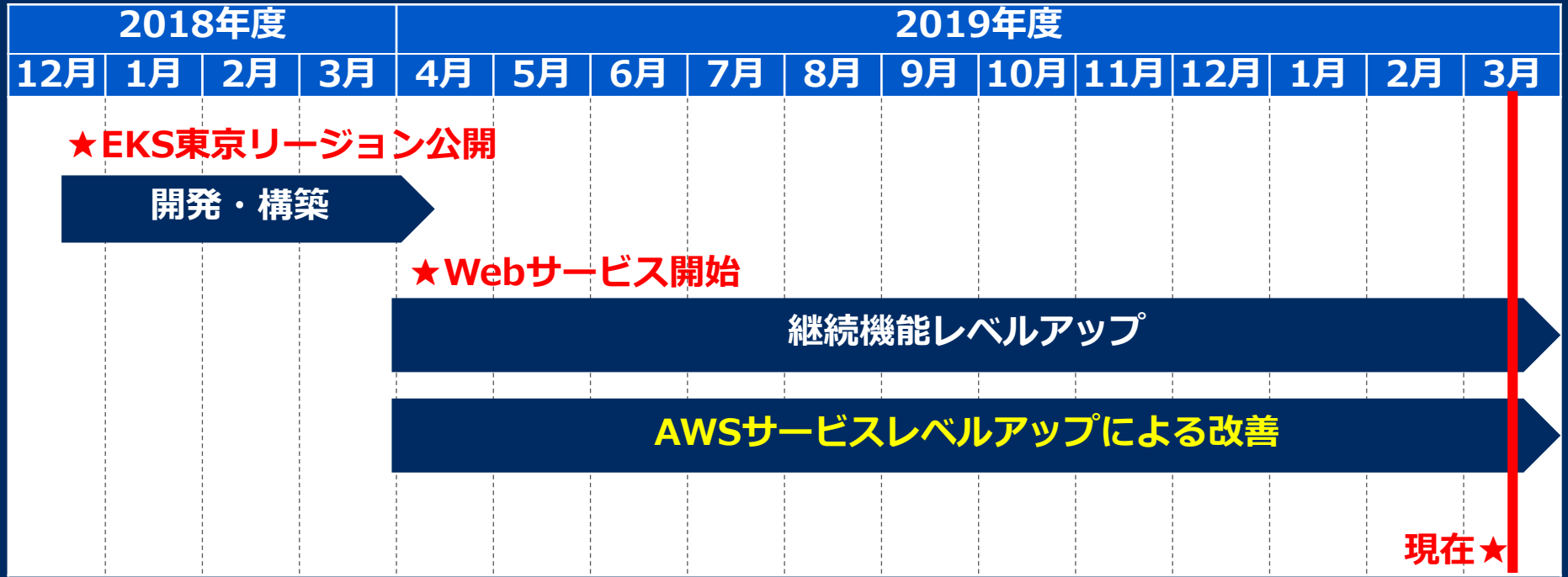
https://aws.amazon.com/jp/blogs/news/amazon-eks-tokyo-region/

東京リージョンにEKSが公開！！！
EKSを利用することで決定！！！

SINから～運用まで

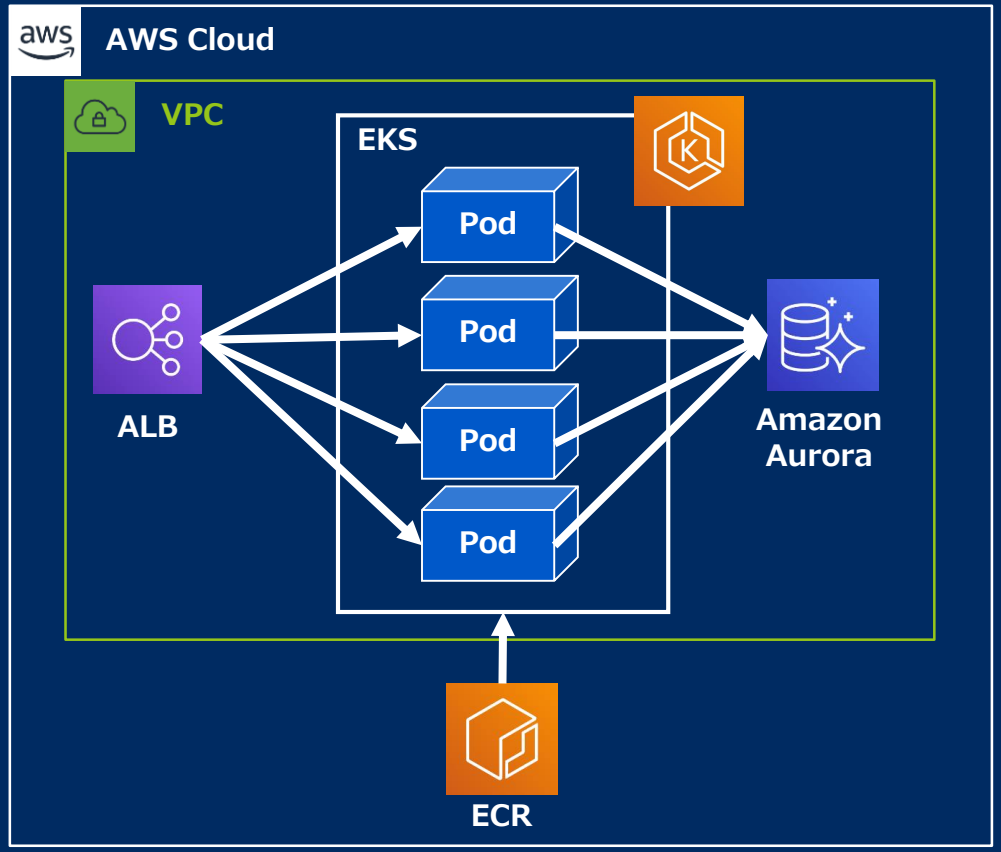
サービス開始から1年

- **約3ヶ月でサービス開始！**AWS・EKSだからスピード感もってリリースできた！
- 現在特に大きな障害はなく、安定稼働中！**AWSの機能レベルアップによる改善も実施中！**



4．EKSの実装・運用例

EKS全体構成（概要）



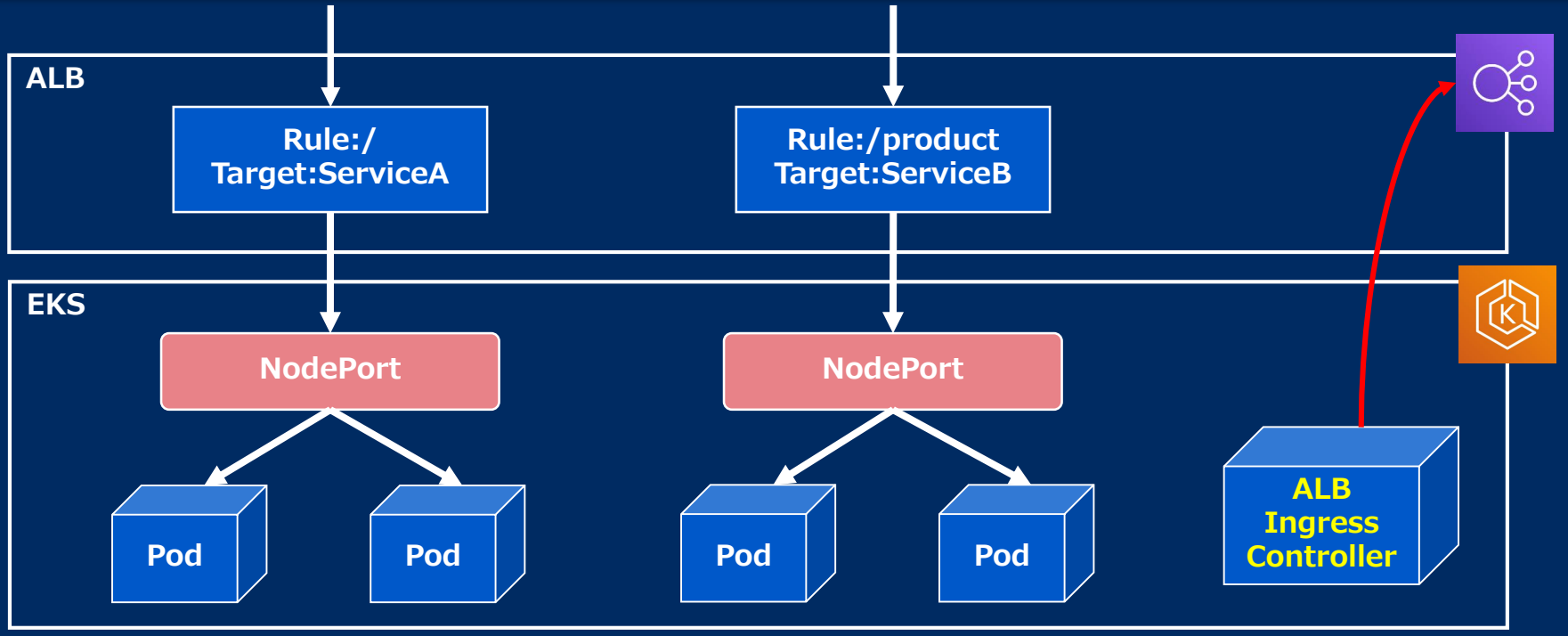
※補足：実構成ではなく、全体構成の概要図となります

全体構成

- ✓フロントにはALBを利用
- ✓データベースはAmazon Auroraを利用
- ✓APはマイクロサービスアーキテクチャを採用
- ✓PVの利用はなし
- ✓CronJobを利用し、定期処理を実行
- ✓各テナント用に名前空間を分離
- ✓コンテナレジストリはECRを利用
- ✓**AWSの実装・運用については、Well Architected レビューを実施**

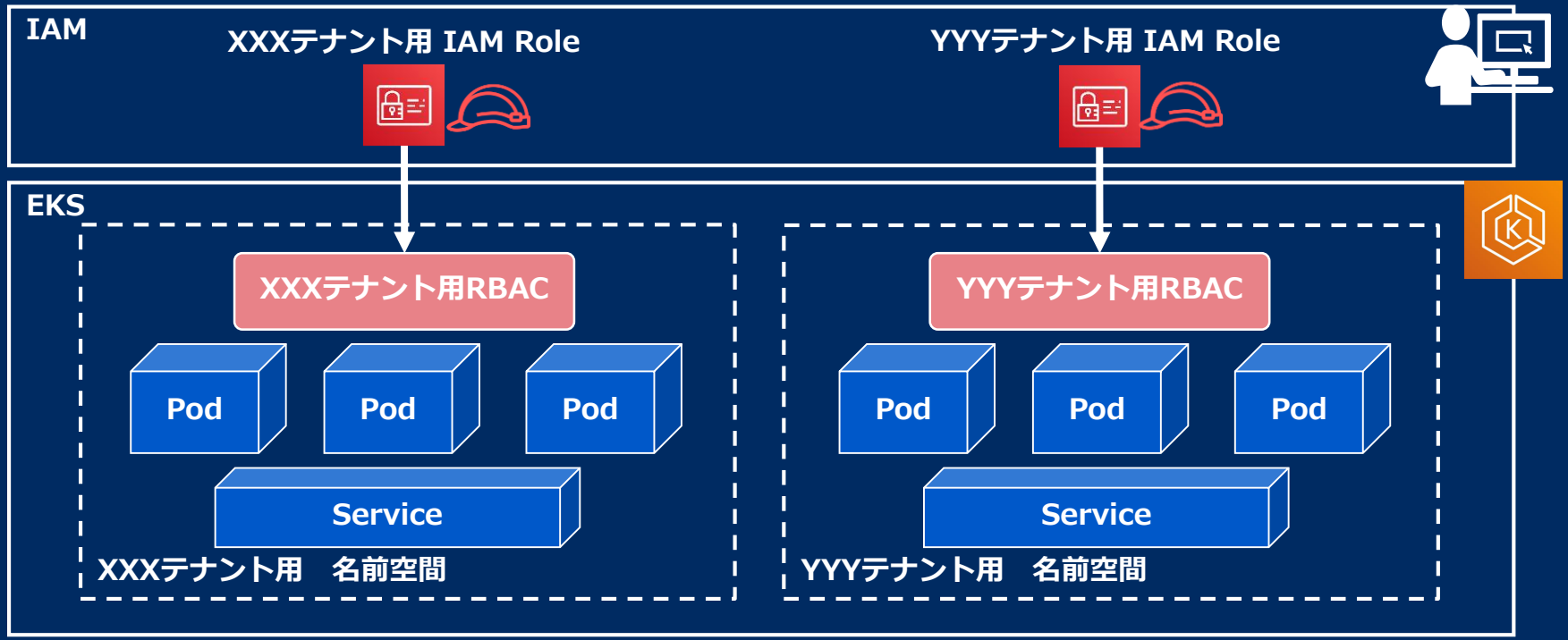
ALBの利用について

Ingress ControllerとしてALB Ingress Controllerを導入し、公開するAPIごとにRuleを設定し各Podに分散を実施



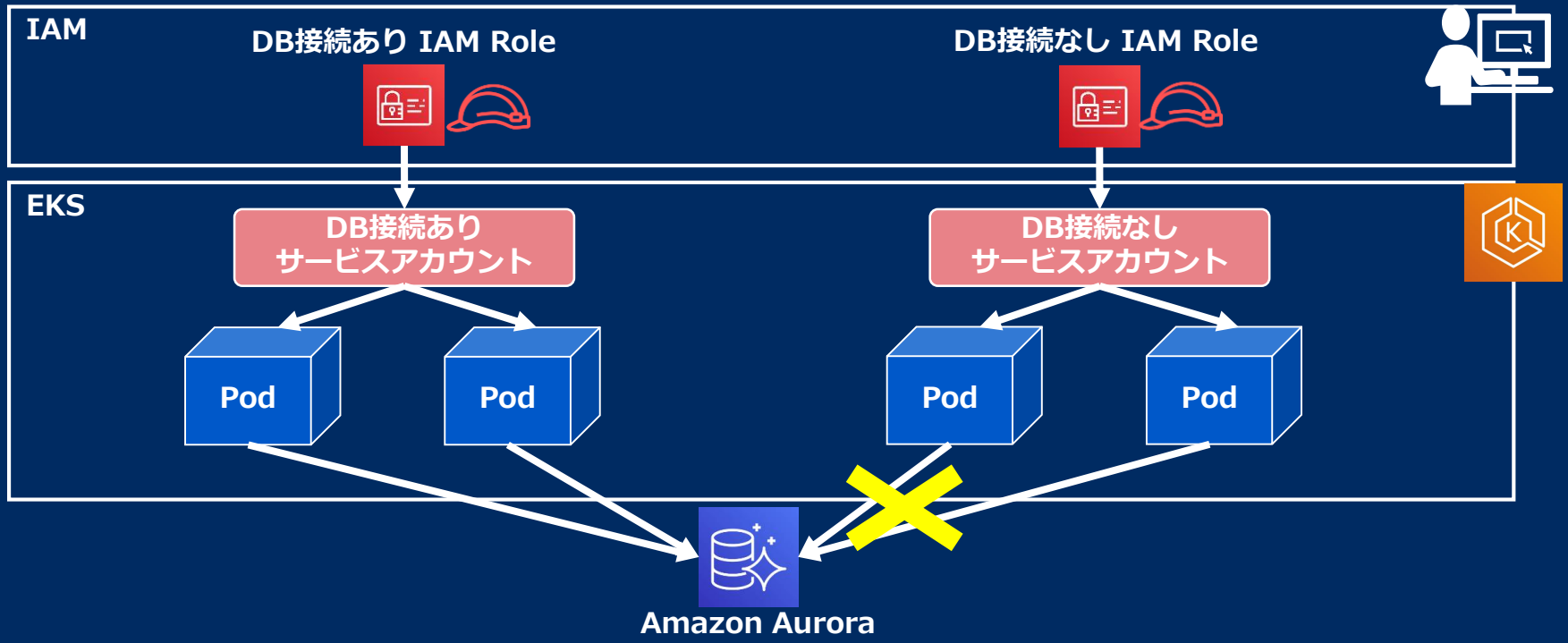
名前空間ごとの権限分離

KubernetesのRBACとIAMユーザー/ロールの紐づけを行うことで、名前空間ごとのIAMによる権限分離を実現



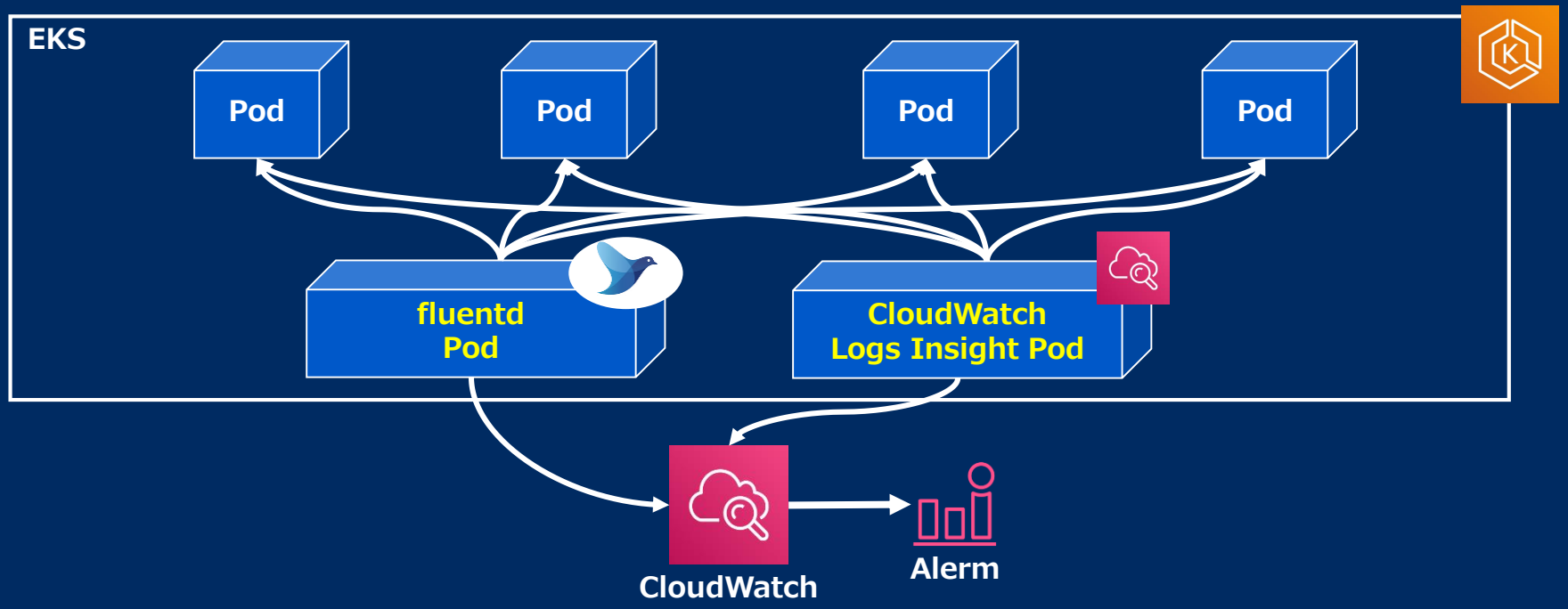
Podごとの権限分離

KubernetesのサービスアカウントとIAMのロールの紐づけを行うことで、Podごとに細かい権限分離を実現することが可能



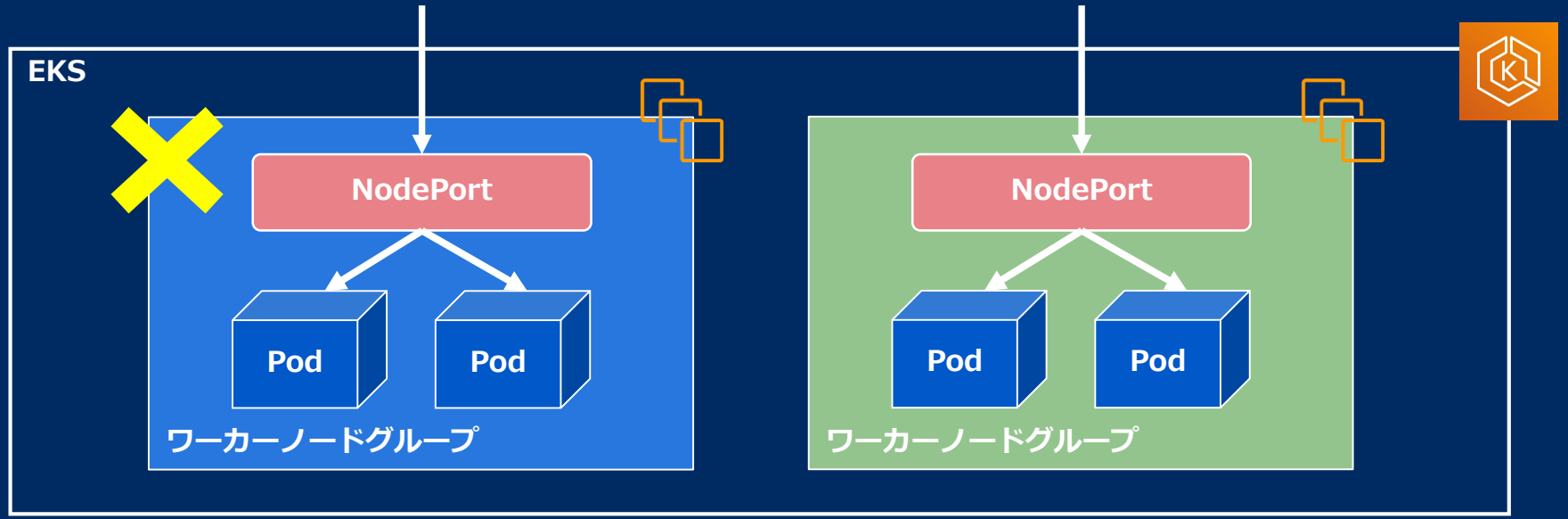
モニタリング

各コンテナログとPodログについては、fluentd・CloudWatch Logs Insightを利用し、CloudWatchに集約・監視を実施



無停止デプロイ運用

グリーン環境(新)をデプロイ後、ブルー環境(旧)をcordon、drainすることで切り離しし、ブルー環境を削除し無停止運用を実現



EKSバージョンアップ運用

- EKSのバージョンアップは以下対応が必要
  - EKSクラスタ自体のバージョンアップ
  - ワーカーノード側のKubernetesコンポーネントのバージョンアップ
  - kubectrl等のクライアント側のバージョンアップ 等
- バージョンアップ後に困ったこと
  - **例：バージョンアップ後、Pod間の通信が一部不可となる事象が発生**
    結果、Amazon VPC CNI Pluginの不具合あることが発覚
    https://github.com/aws/amazon-vpc-cni-k8s/issues/641

✓ **バージョンアップ後は検証環境など、十分な動作確認をした上で、プロダクション環境へのバージョンアップが必要！！！**

その他運用について

コンテナレジストリ運用

- ECRのPermission設定で、IAM権限ごとに操作権限を与えることが可能
- 脆弱性スキャン機能を利用し、コンテナイメージのセキュリティ対策も実施

Kubernetesの機能について

- 基本的なKubernetesの機能は利用可能だが、
  EKSでは一部デフォルトで利用できないものもあり

✓ **他サービスと組み合わせて十分なセキュリティ対策が可能！**
✓ **Kubernetesで利用したい機能がある場合は事前に確認する必要！！！**

5．まとめ

まとめ

これまで

- 細かい障害（AWS障害も含めて(笑)）もありました
- Kubernetesを利用することでリリースプロセスの短縮ができました
- 新規サービス展開が広がりました

これから

- サービス拡大とより安定したサービス提供へ
- バージョンアップ時にワーカーノードを意識したくない！！！
- 2019年12月にラスベガスで公開された**EKS on Fargate**を利用したい！！
- **更なる運用レベルアップへ！！！（AWS様、アップデート期待してます）**

ちなみに

EKS on Fargateについて

- Fargateで設定するプロファイル名とKubernetesの設定する名前空間名を一致させないとFargate上で起動できない
- コンテナログの収集はfluentdのサイドカーにする必要あり
- Podの性能情報メトリクスが現状、取得できない

EKS on Fargateは
考慮が多々必要そうなので、
現在絶賛検証中！！！

ご清聴
ありがとうございました。


Orchestrating a brighter world
NEC