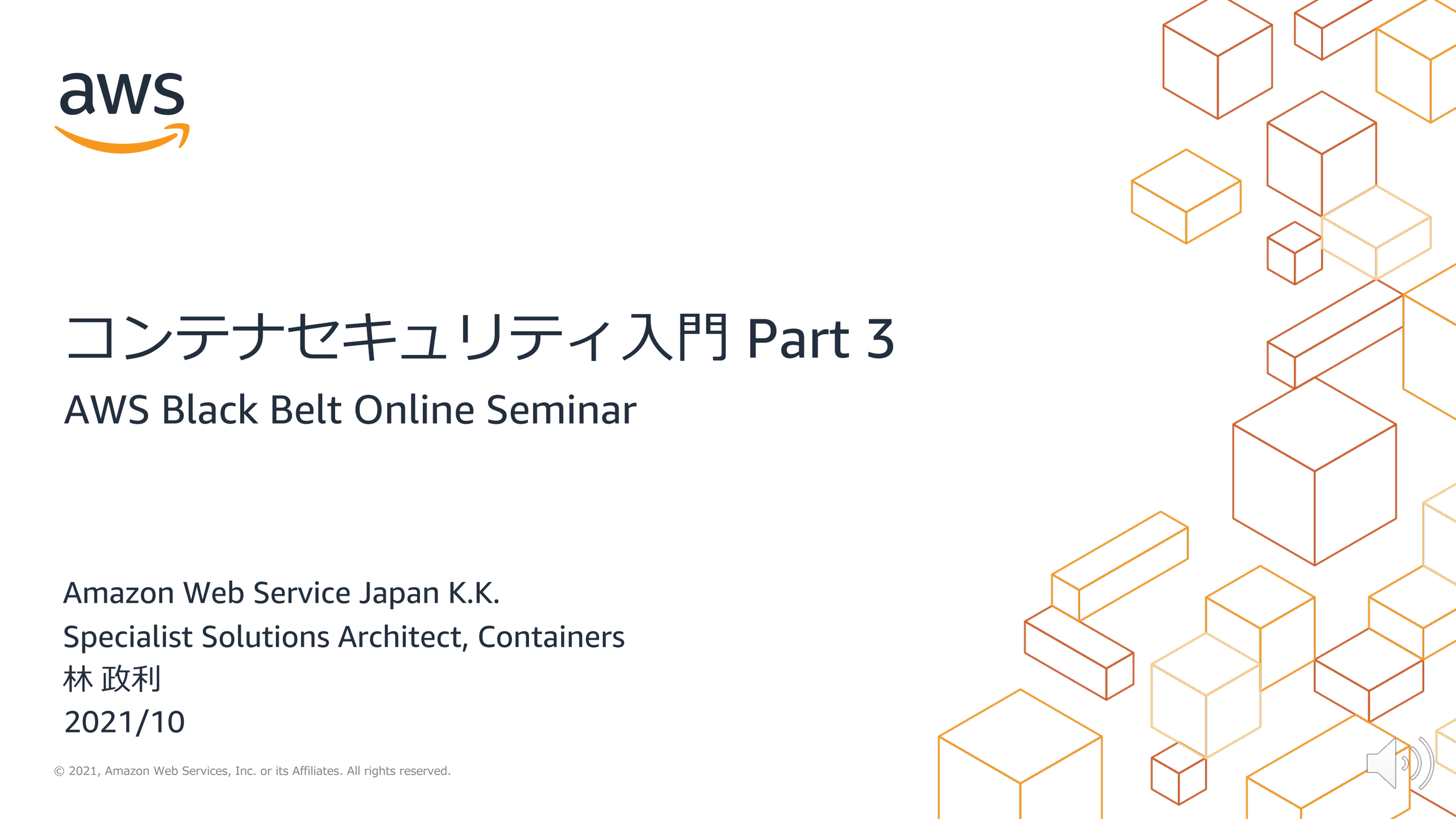
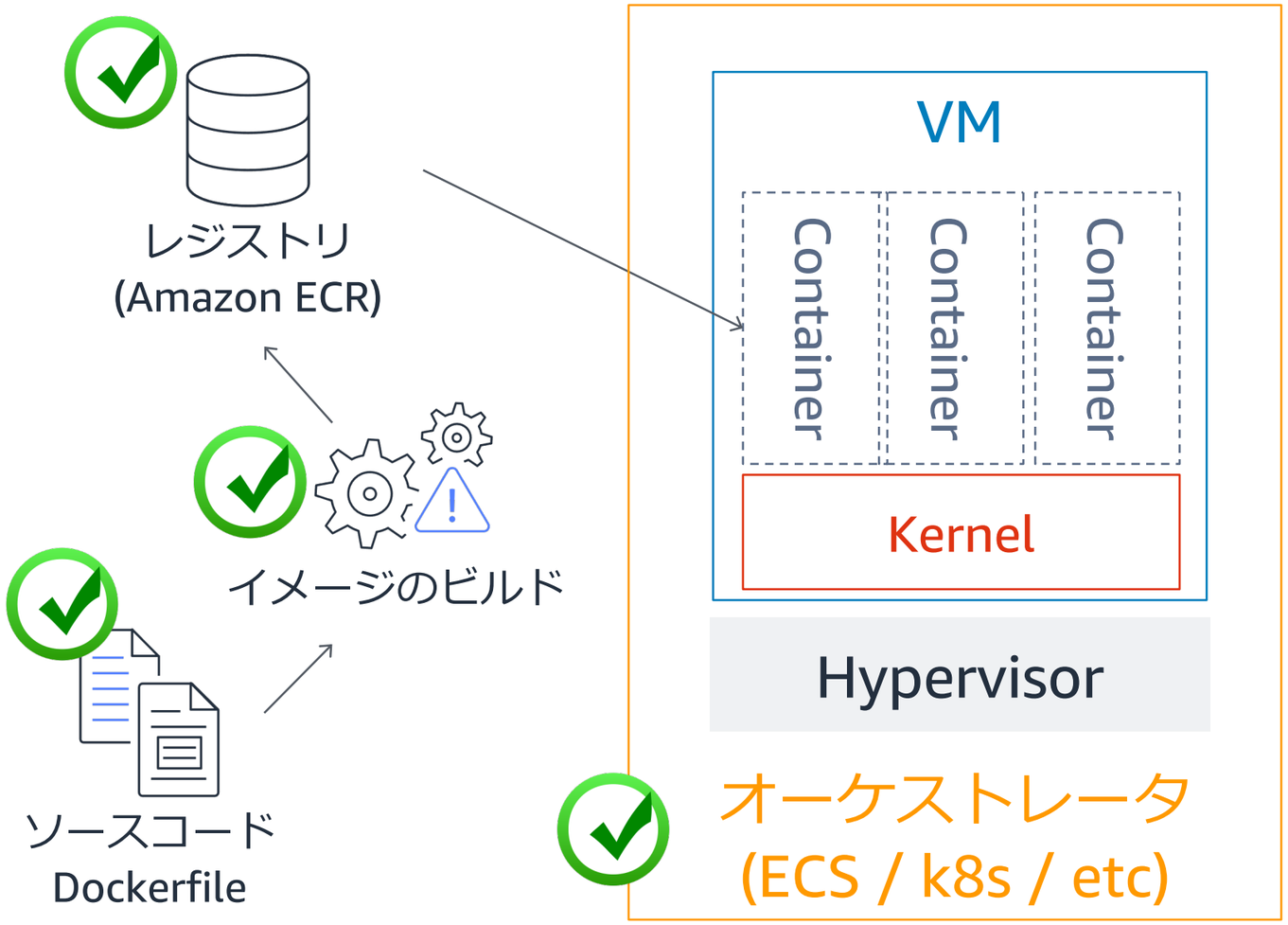


コンテナセキュリティ入門 Part 3

AWS Black Belt Online Seminar

Amazon Web Service Japan K.K.
Specialist Solutions Architect, Containers
林 政利
2021/10

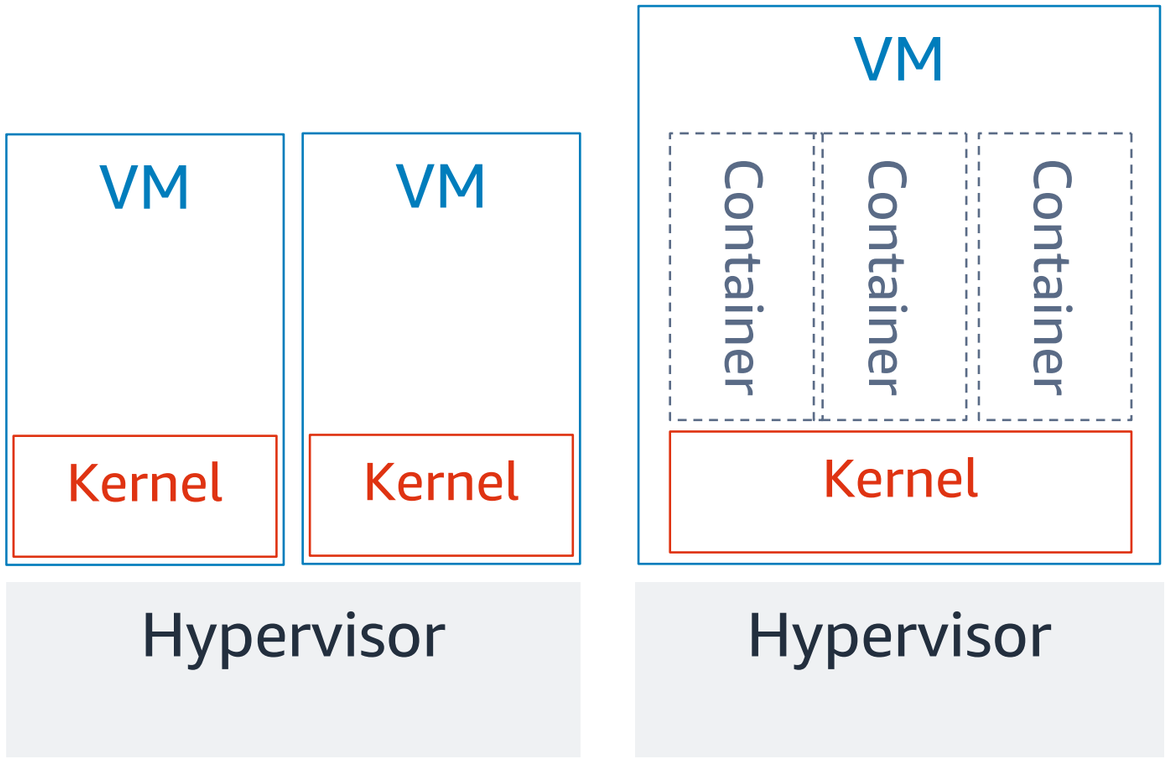
アジェンダ



- イメージの開発とビルド
- サプライチェーン
- オーケストレータの保護
- ホストのセキュリティ
- 実行時のセキュリティ

コンテナホストについて

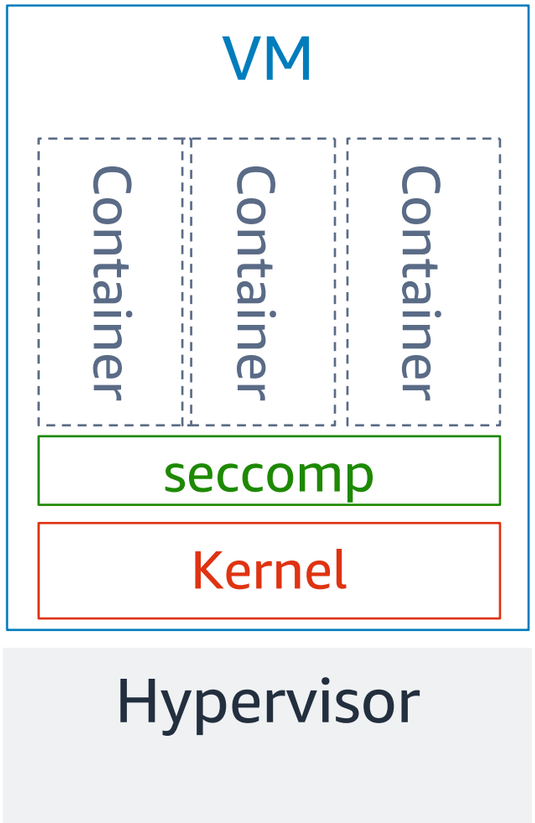
仮想マシンより低いコンテナの分離レベル



- コンテナはカーネルを共有
  - カーネルに脆弱性 -> コンテナから「エスケープ」できるリスク
- コンテナのサンドボックス機能を強化
  - コンテナやホストの危殆化対策

seccomp による分離レベルの強化

コンテナプロセスが実行できるシステムコールを制限する

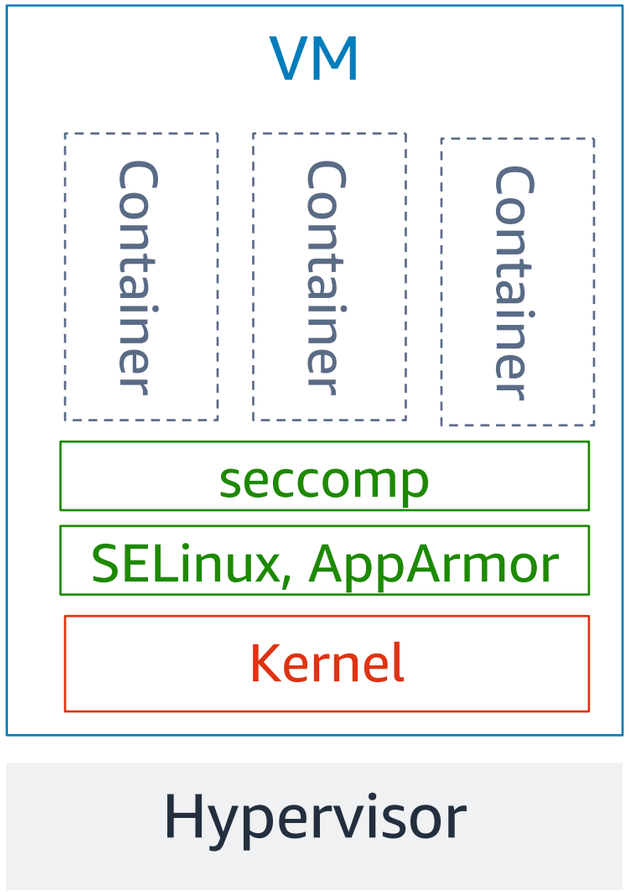


- コンテナには不要なシステムコール
  - clock_adjtime, clock_settime
  - create_module, delete_module, init_module
- Dockerデフォルトプロファイルが有用
  - 40以上のシステムコールをブロック

```
kind: Pod
spec:
 securityContext:
    seccompProfile:
      type: RuntimeDefault
  containers:
  - image: busybox
```

Linux Security Module による分離レベルの強化

SELinux, AppArmor でコンテナがアクセス出来るリソースを制限する



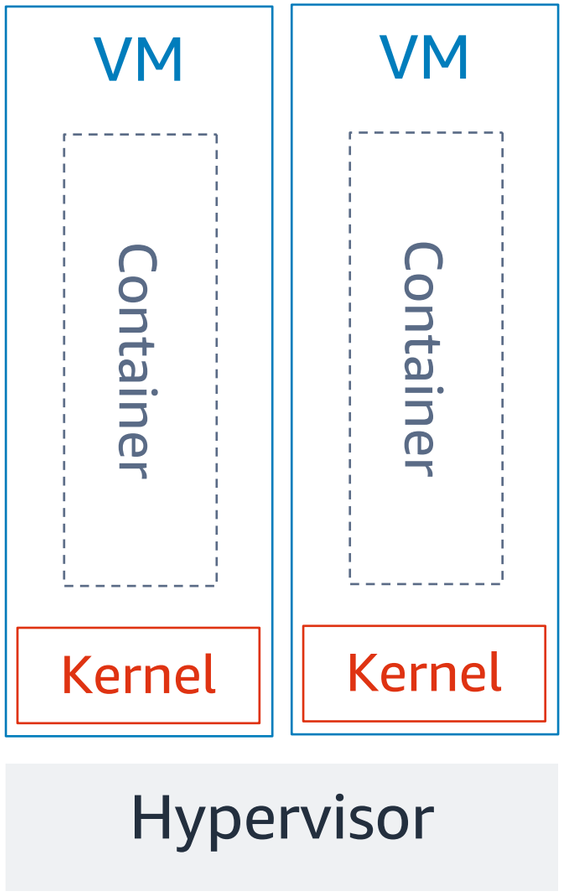
- アクセス不要なリソースへのアクセス
  - /proc, /etc など
- AppArmor の場合 Docker デフォルトプロファイルが有用
  - /proc などへのアクセスを防ぐ
- SELinux を有効化すると、デフォルトでは
  コンテナプロセスに container_t ラベルが付く

```
kind: Pod
 annotations:

container.apparmor.security.beta.kubernetes.io/<container_name>: runtime/default
spec: ...
```

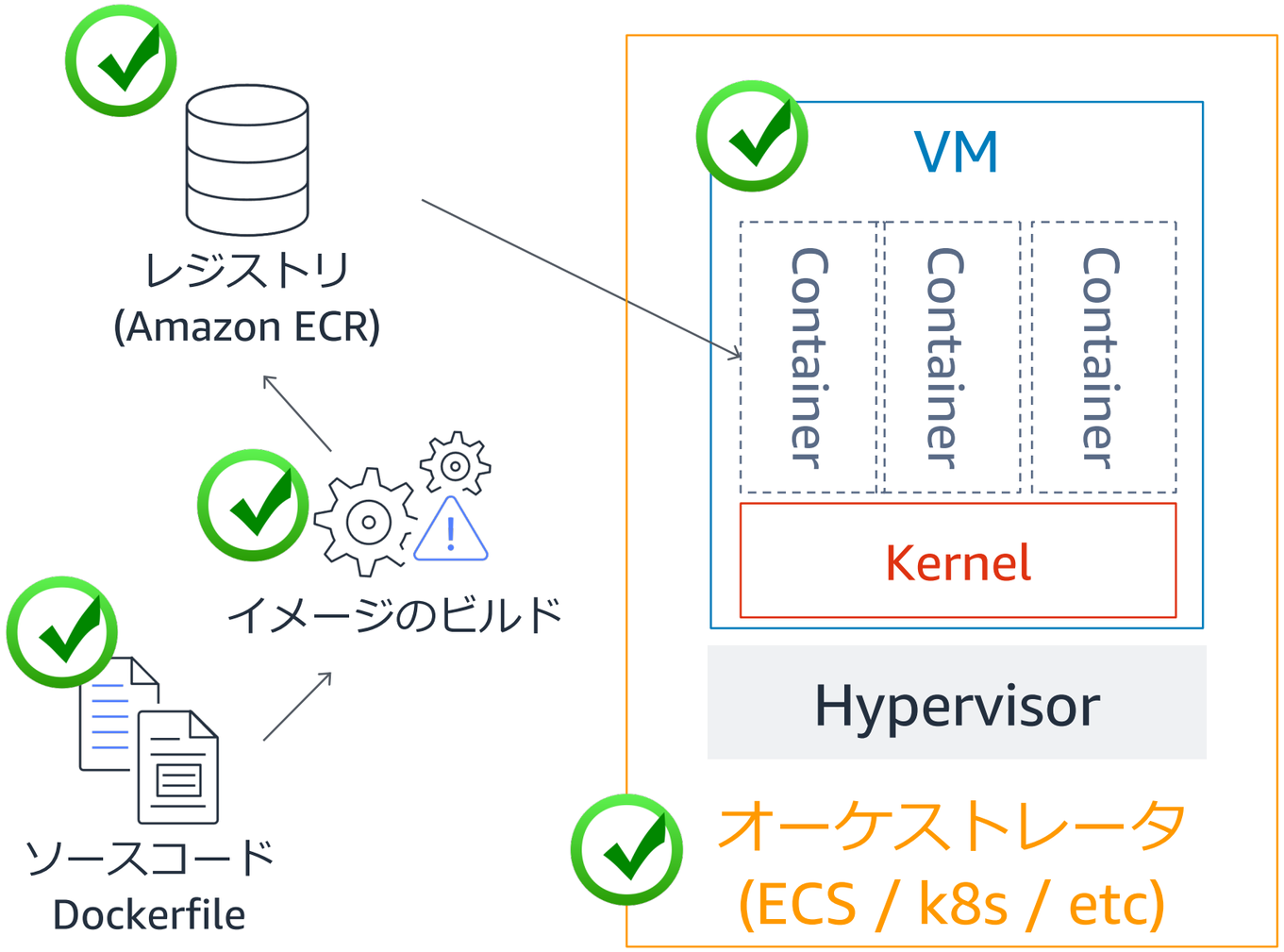
カーネル分離による分離レベルの強化

コンテナをカーネルレベルで分離する



- カーネルの共有をしないという考え方
  - Kata Container
    - コンテナ起動をプロキシして VM 内でコンテナを実行
- AWS Fargate
  - AWS で簡単にカーネルの分離を実現する

アジェンダ



- イメージの開発とビルド
- サプライチェーン
- オーケストレータの保護
- ホストのセキュリティ
- 実行時のセキュリティ

コンテナ実行中のセキュリティ

アプリケーションのデータや挙動に関するセキュリティ

ネットワークセキュリティ

コンテナが相互に、どのように通信するのかポリシーを
定義して制限する

データの保護

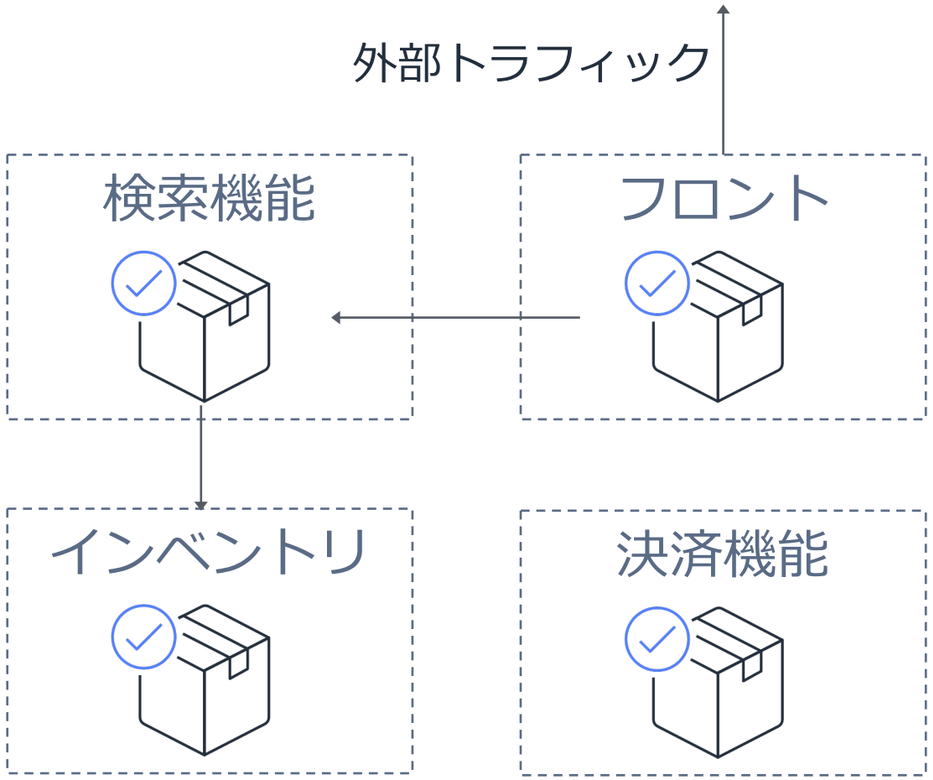
インスタンス、ストレージの暗号化、シークレットの取り扱い

ランタイムセキュリティ

コンテナアプリケーションが利用したコマンドや
ファイルアクセスなどを記録して、予期しない挙動を検知

ネットワークセキュリティ

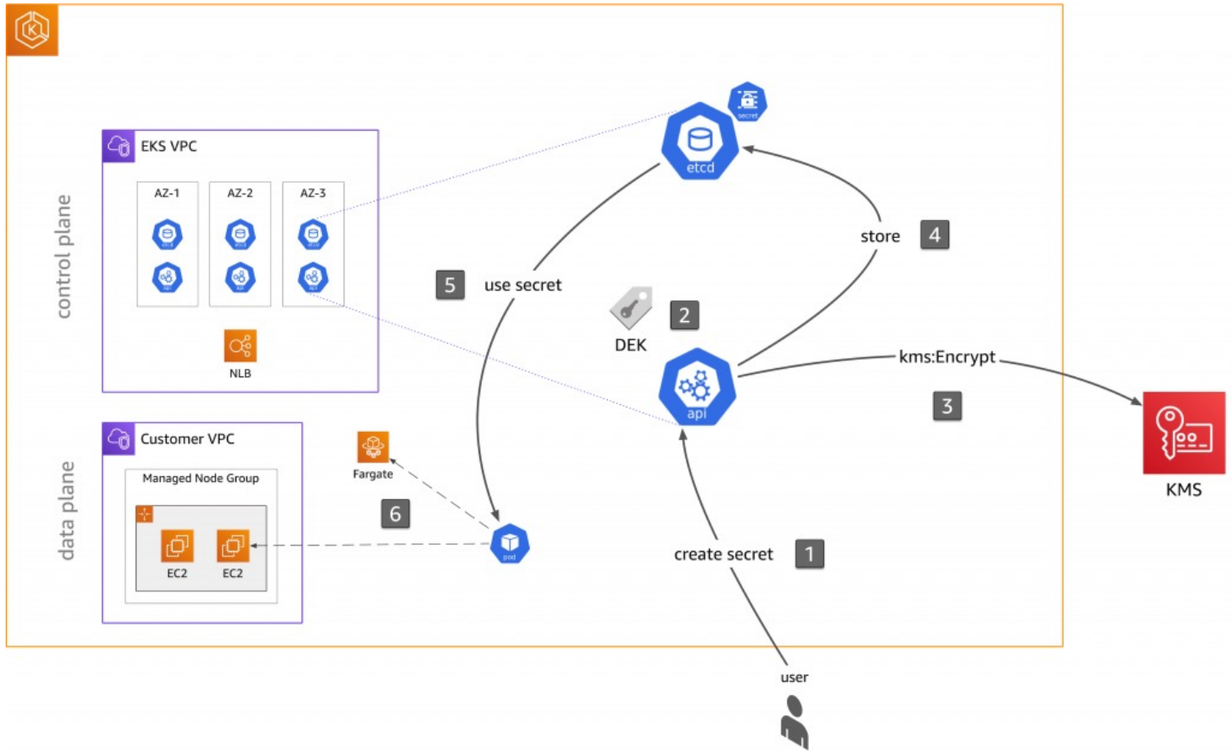
コンテナ間の通信をポリシーで定義する



- コンテナ間通信の定義をポリシーにして、ポリシー外の通信を拒否できる
  - (AWS) Security Group
    - ECS のタスク、Kubernetes のPod に割り当て
  - (Kubernetes) Network Policy
    - Calico / Cilium
  - アプリケーションレベル
    - Cilium
    - Service Mesh

データの保護

暗号化、秘匿情報の管理

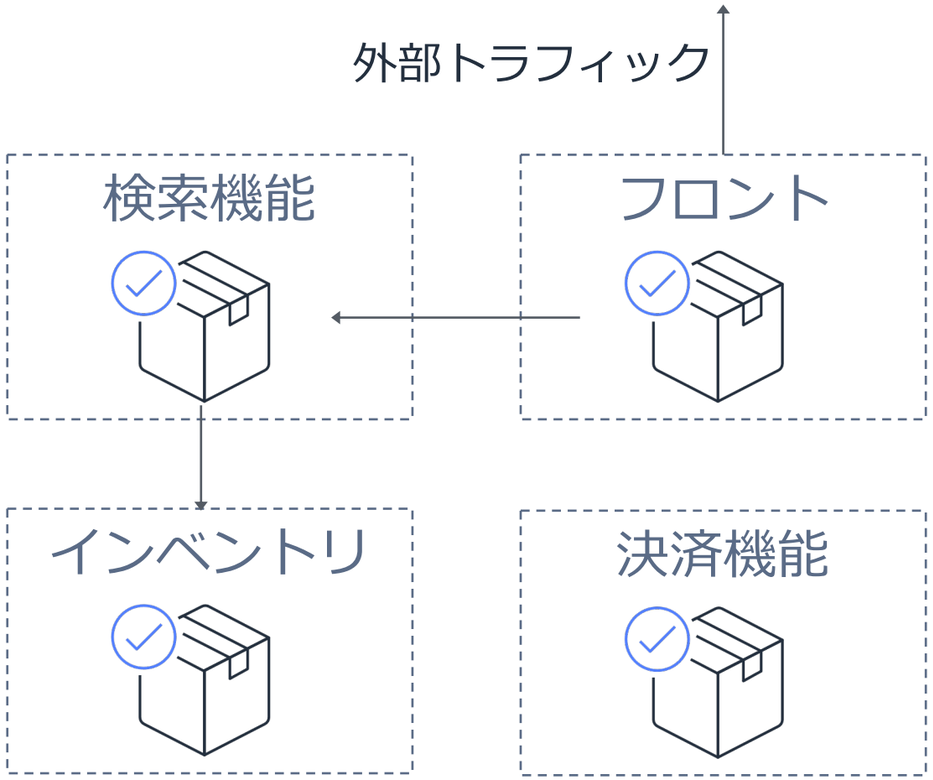


- データの暗号化
  - ノード、コンテナインスタンス
    - EC2 で暗号化を有効化する
    - Fargate (デフォルトで暗号化)
  - 外部ストレージ(転送・保存時)
    - EFS (ECS / EKS CSI Driver)
    - EBS (EKS, CSI Driver / in-tree)
- シークレット, パラメータ(転送・保存時)
  - Secret リソース (Kubernetes)
    - エンベロープ暗号化を有効にする
  - Secrets Manager, Parameter Store
  - 渡し方: ファイル / 環境変数 / API

https://aws.amazon.com/jp/blogs/news/using-eks-encryption-provider-support-for-defense-in-depth/

ランタイムセキュリティ

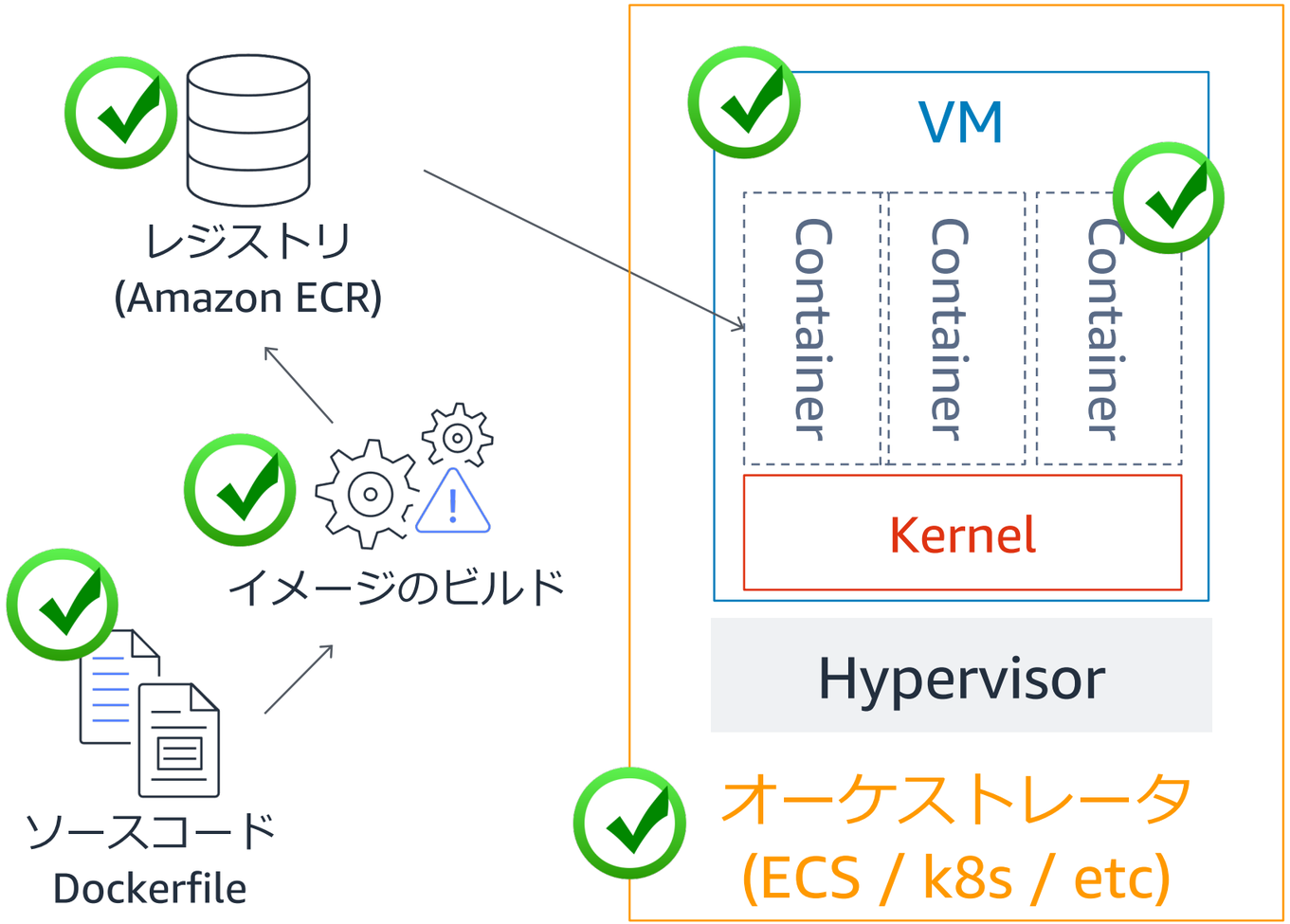
「プロファイル」を作り、コンテナの挙動がそのプロファイルに従っているか確認する



- ネットワーク
  - サービス間通信をモニタして ネットワークのポリシー を構築する
- コマンド実行
  - eBPF, ptrace を利用してコマンド実行をモニタ
- ファイルアクセス
  - eBPF, ptrace でファイルアクセスをモニタ
- 商用ツール
  - Sysdig, Aqua (ECS, EKS, Fargate)
  - Trend Micro (EKS)
- オープンソース
  - CNCF Falco (ECS, EKS, Fargate)

まとめ

フェーズごとにどんな脅威が考えられるかで整理する



- イメージの開発とビルド
- サプライチェーン
- オーケストレータの保護
- ホストのセキュリティ
- 実行時のセキュリティ

参考文献

Amazon ECS Best Practices - Security
https://docs.aws.amazon.com/AmazonECS/latest/bestpracticesguide/security.html

Amazon EKS Best Practices Guide for Security
https://aws.github.io/aws-eks-best-practices/security/docs/

Container Security (書籍)
https://www.amazon.co.jp/dp/B088B9KKGC

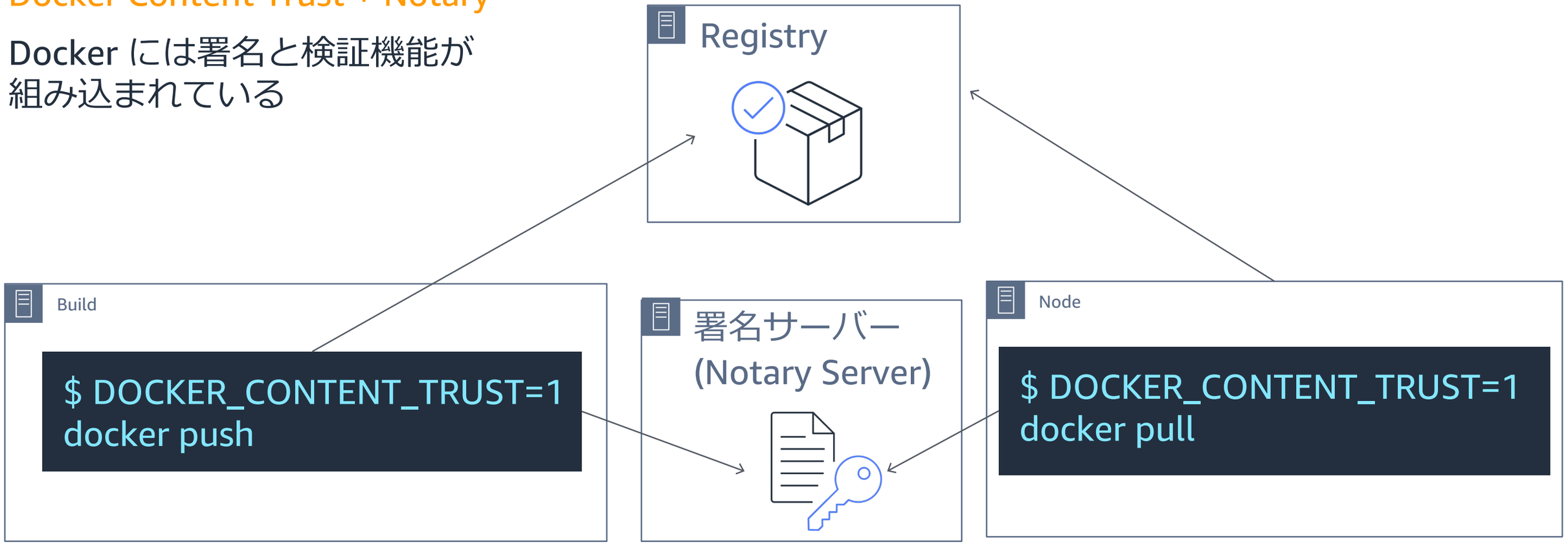
NIST SP800-900 アプリケーションコンテナセキュリティガイド
https://www.ipa.go.jp/files/000085279.pdf

ビルドしたイメージとの同一性の保証

イメージを署名して実行されているコンテナを検証する

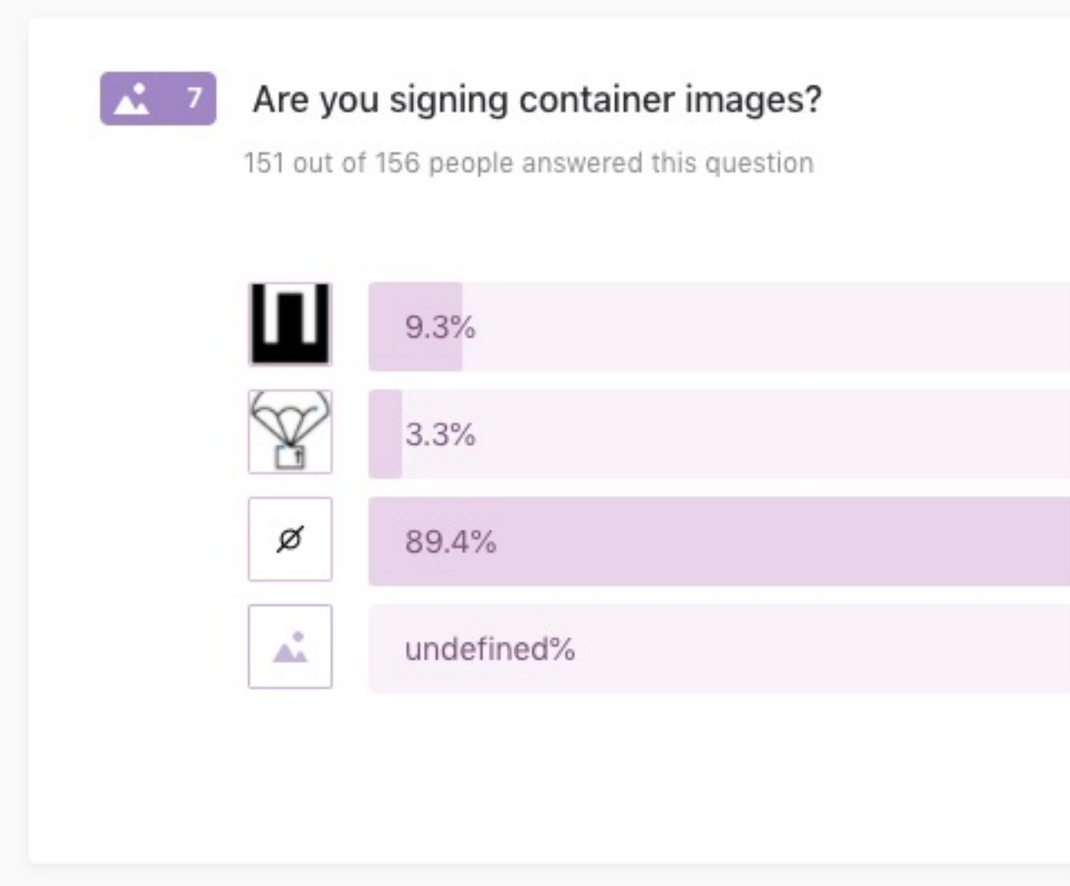
Docker Content Trust + Notary

Docker には署名と検証機能が
組み込まれている



Docker Content Trust、 Notary の課題

可搬性とユーザビリティに課題があり、署名の利用は進んでいない



オーケストレータが未対応

ECS、Kubernetes などのオーケストレータで署名を直接検証できない

Notary サーバーが必要

Notary サーバーが統合されている一部のレジストリでしか使えない

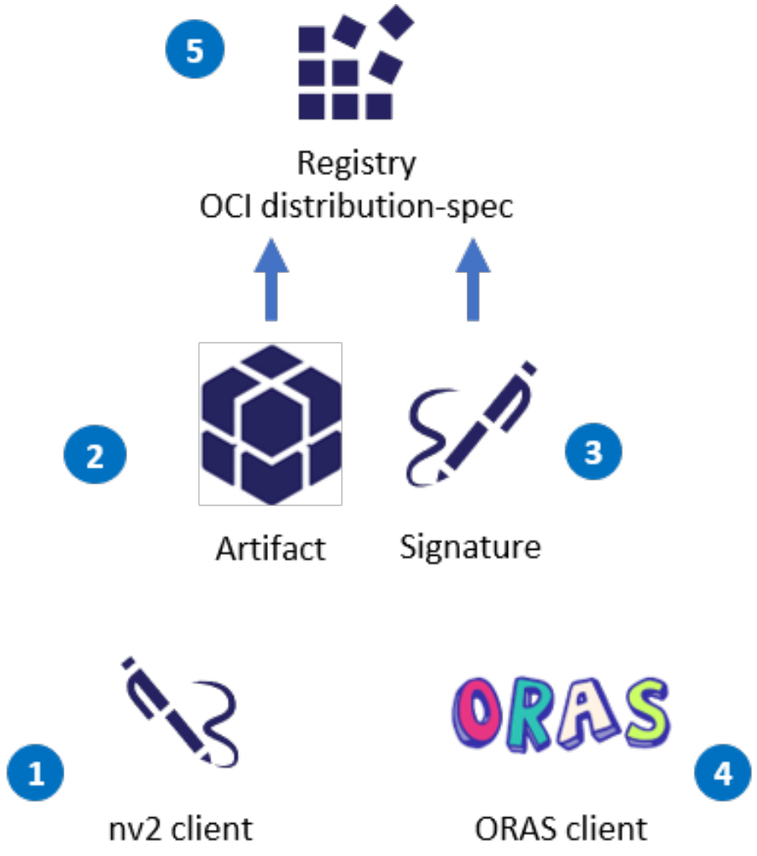
イメージの署名に可搬性がない

使える環境、レジストリが限定されているので、署名の検証ができたりできなかったりする

https://aws.amazon.com/jp/blogs/containers/results-of-the-2020-aws-container-security-survey/

署名の可搬性とユーザビリティを改善する取り組み

Notary v2 の例



レジストリの API に署名機能追加

Notary サーバーのない環境で利用できる

可搬性を意識している

ベースイメージの署名を確認してビルドし、新しく署名を登録して Push

ユーザビリティの改善

利用できる環境が標準化されることで、可能な場合はデフォルトで署名、検証する

https://github.com/notaryproject/nv2

このセッションで扱うこと・学べること

コンテナについての一般的なセキュリティの考え方

実運用でのガイダンス、指針の整理

本セッションの担当: 林　政利

Specialist Solutions Architect, Containers

好きなサービス

Amazon Elastic Kubernetes Service (Amazon EKS)
AWS Certificate Manager

SIer

フリーランス
受託開発

Web系企業
サービス開発・運用

外資ベンダー
コンテナ担当SA・
サポートエンジニア

AWS Japan
コンテナ担当SA