マルチテナントクラスタの
セキュリティ対策の勘所

2024年09月06日

NTTコミュニケーションズ株式会社

本日のアジェンダ

- プロダクト「**Qmonus Value Stream　**」の紹介
- オンラインセッション「**マルチテナントクラスタのセキュリティ対策の勘所**」のダイジェスト紹介

自己紹介

NTT コミュニケーションズ株式会社
イノベーションセンター
中井　悠人

略歴

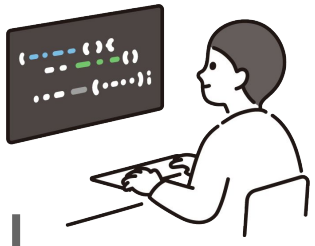
- 2014年 NTTコミュニケーションズ入社
- 2014～2017
  - クラウドのIaaS開発・運用や監視・運用システムの内製開発に従事
- 2017～現在
  - 内製フレームワークを使った社内のプロダクト開発支援の担当者（アプリ開発、インフラ構築のどちらも経験）
  - インフラ構築の自動化、プラットフォーム化のリード兼開発者
  - **（現在）プラットフォームのプロダクトマネージャー**

本日のアジェンダ

- プロダクト「Qmonus Value Stream 」の紹介
- オンラインセッション「マルチテナントクラスタのセキュリティ対策の勘所」のダイジェスト紹介

Qmonus® Value Stream

プロダクトに最適なクラウドアーキテクチャと CI/CDパイプラインを、最短10分で構築



アプリケーションに必要な要件を選択

GitHub

お客様管理のアプリの
ソースコードリポジトリ

※ GitLab / Bitbucket / Backlogも対応

Qmonus® Value Stream

要件に沿ったデリバリフローを自動生成

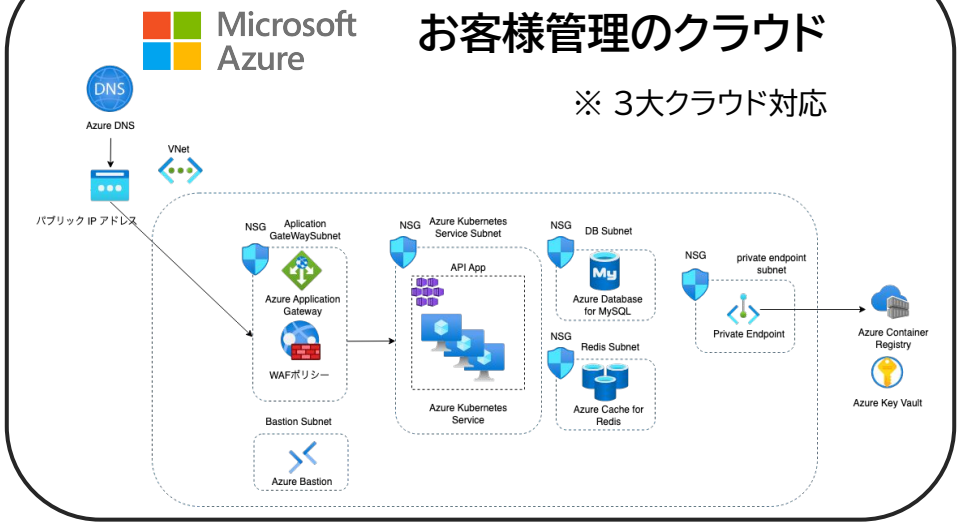
Infra Deploy → Container Image Build → Image Scan → Test → App Deploy

最適なクラウドアーキテクチャと
アプリケーションをクラウドにデプロイ

Microsoft Azure

お客様管理のクラウド

※ 3大クラウド対応



詳細はこちら（https://www.valuestream.qmonus.net/）

Qmonus Value Streamが解決する課題

課題

01 インフラストラクチャの設計、検討に時間を取られてしまう
- 適切なサービスの調査
- リソース量や料金などコストの調査
- セキュリティ要件

02 開発以外の管理業務に多く時間を取られてしまう
- インフラの管理・更新
- アプリケーションのバージョン管理
- 自動化の取り組み

03 アプリケーション基盤の更新に時間を取られてしまう
- プロダクトの成長に伴うシステム全体の冗長化など

04 上記課題の解決に時間もお金もかかる
- ベンダーやコンサルへの委託
- ベテラン社員の登用

Qmonus Value Stream でできること

01 **検証済みのインフラストラクチャ**と**ドキュメント**が設計、検討の手間を減らします

実績に基づいたクラウド構成とパイプラインをパッケージ化し、要件を選ぶだけで構築可能。選定理由などのドキュメントも完備。

02 **オールインワンのプラットフォーム**が管理業務の手間を減らします

制御・管理・実行などCI/CDやIaCのツール管理の手間から解放

03 柔軟にインフラ構成を変更できるので要件が変わってもすぐに基盤を更新できます

要件を選択し直すだけで構成変更

04 Qmouns Value Streamが今なら”**無料**”であなたの悩みを解決します

今ならフリートライアルで全機能が無料。導入支援も無料。

Qmonus Value Stream 利用イメージ

要件(基本となるアーキテクチャとオプション)を選択するだけで、適切なアプリケーション基盤とCI/CDパイプラインを構築。SaaS型プラットフォームなので実行まで可能

Google Cloud

CDNと統合されたフルマネージドな環境にSingle Page Application (SPA) を構築する

Microsoft Azure

フルマネージド環境でスケーラブルな Web API を構築する

- シンプルな Web アプリケーションをデプロイしたい
- アプリケーションの実行環境の運用コストを軽減して、アプリケーションの開発に集中したい

コンテナアプリケーションをデプロイする基盤として App Service（Web App for Containers）を使用します。またアプリケーションが利用できるデータベースとして Azure Database for MySQL を使用します。

■メインとなるサービス

App Service（Web App for Container） Azure Database for MySQL

アプリケーションの公開範囲を制限したい

コンテナの脆弱性検知をしたい

Trivyを用いてコンテナイメージのスキャンを実施します。コンテナイメージ内の脆弱性やセキュリティリスクを検出することができます。

Category

Web APIを構築したい

Provisioning Target

Microsoft Azure

Architecture

Kubernetes で大規模なコンテナオーケストレーションが可能な Web API を構築する

Option

コンテナの脆弱性検知をしたい

アプリケーションの公開範囲を制限したい

build

Pipeline: azure-api-backend-build
Deployment: azure-api-backend/azure-demo-stg

scan

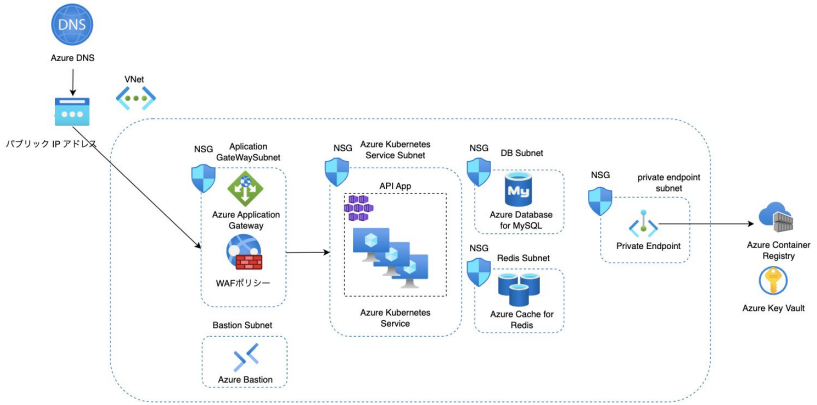
Pipeline: azure-api-backend-scan
Deployment: azure-api-backend/azure-demo-stg

deploy

Pipeline: azure-api-backend-deploy
Deployment: azure-api-backend/azure-demo-stg

Configuratio

The following is
corresponding configuration diagrams and qvsconfigs will be displayed for each.



NTTグループでの実績

35
プロジェクト

NTTコミュニケーションズをはじめ、NTTグループ各社**35PJ**にて利用

商用
環境

PoCではなく、重要なサービス基盤の**商用環境**での実績

500
回/月デプロイ

半年～1年に1度のリリースが一般的だったグループ内のサービス開発が、導入することで平均1～2ヶ月のリリース頻度に向上

フリートライアル

1 全ての機能を無料提供

お客様のチームが今回のPoCの対象とするプロダクトでDevOpsに取り組めるようになるまで無料（最大6ヶ月）

2 導入・構築の伴走支援も無料提供

私たちはNTTグループのプロダクト開発チームを支えるクラウド/DevOpsのスペシャリストチームです。私たちが**導入・構築を伴走支援します**のでご安心ください。必要に応じてお客様組織やチーム向けの勉強会を企画し実施します。

3 カスタマイズもNTT Comで対応

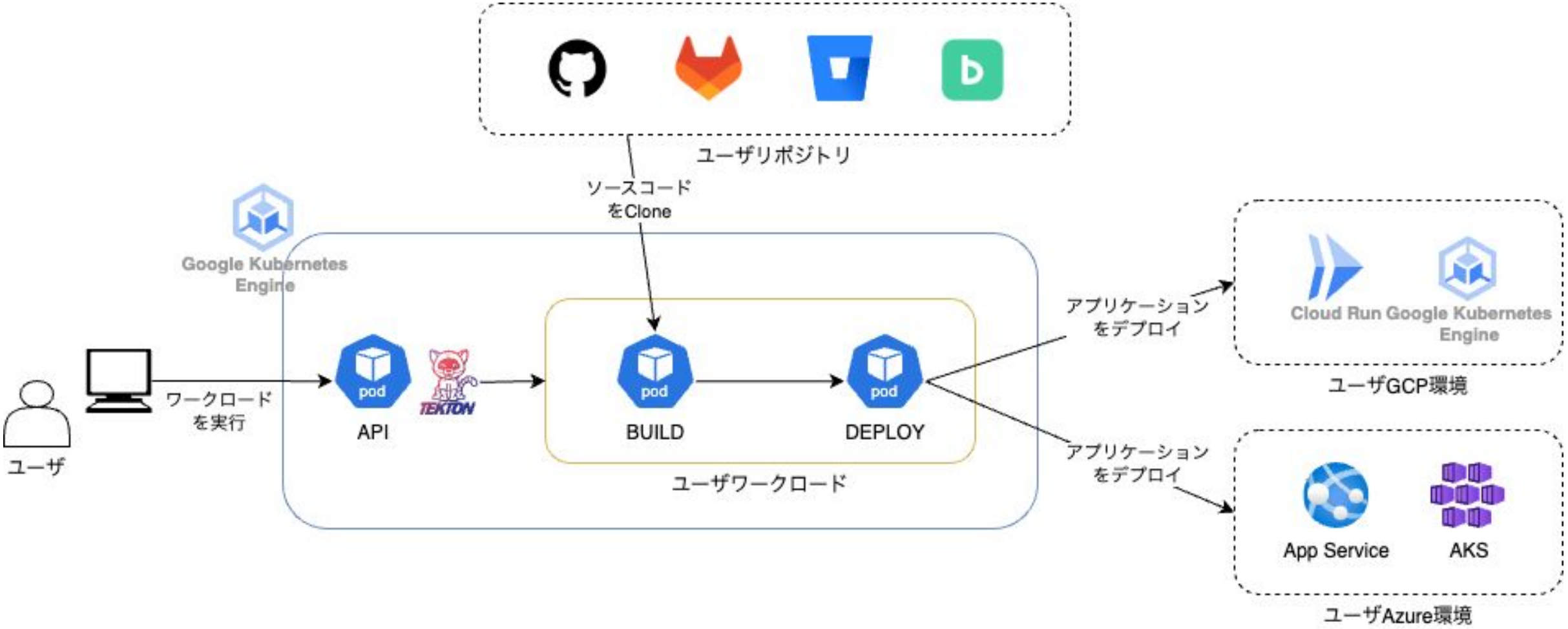
ベーシックなクラウド環境やCI/CDパイプラインは組み込み済みですが、カスタマイズにより**お客様の独自要件への対応**も可能です。カスタマイズ部分は再利用し、汎用化してプラットフォームに組み込むことで、サービス改善に繋げます。

本日のアジェンダ

- プロダクト「Qmonus Value Stream　」の紹介
- **オンラインセッション「マルチテナントクラスタのセキュリティ対策の勘所」のダイジェスト紹介**

システム構成

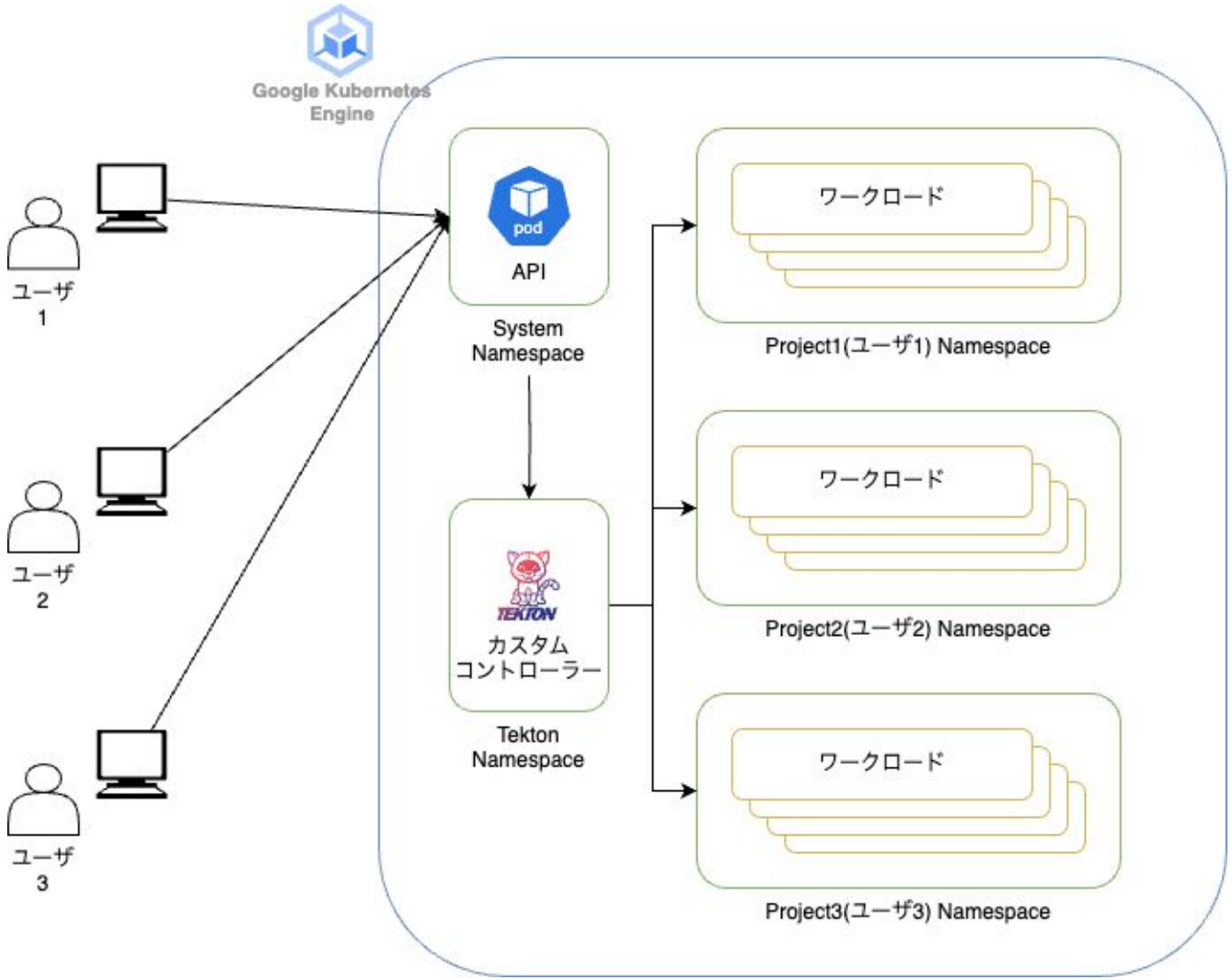
- ワークロードの実行エンジンとしてTektonを採用



ユーザのワークロードをマルチテナント環境で実行

ワークロードの実行環境

- GKE 上にワークロード実行環境を構築
- マルチテナント環境
  - 同一のクラスター上で Namespace を利用してユーザ毎のワークロードを分離
- ワークロードのカスタマイズ
  - ユーザがカスタマイズしたワークロードの実行可能



https://kubernetes.io/

https://tekton.dev/

マルチテナント環境に潜むセキュリティリスク①

ユーザが任意コンテナを起動できるリスク

例… 特権コンテナを悪用された場合

特権コンテナを利用して
ホストの root を奪取

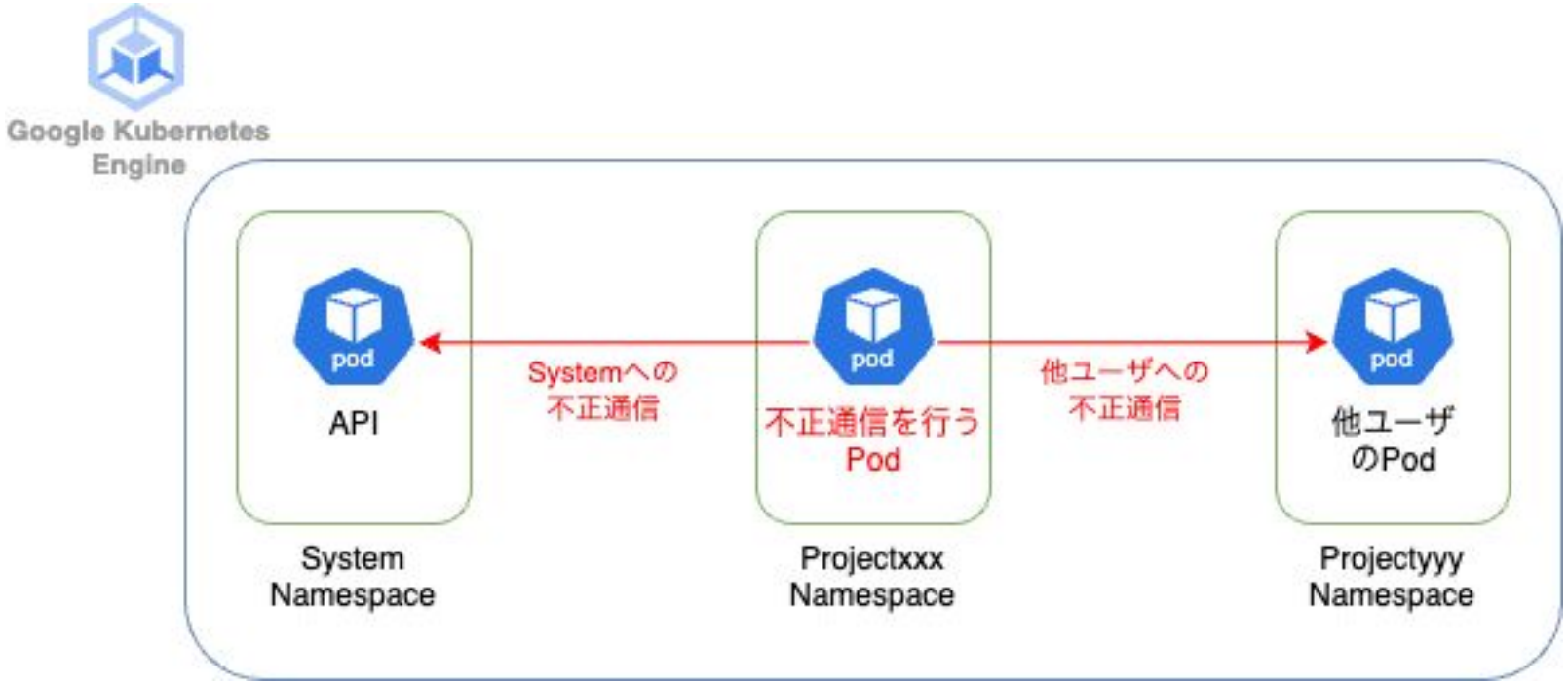
ホストの root 権限で不正操作
- 他 Namespace で不正なPodを起動
- ホスト上で不正プロセスを実行

セキュリティインシデント発生
- ユーザのシークレット情報の漏洩
- 外部へのサイバー攻撃の起点

マルチテナント環境に潜むセキュリティリスク②

Namespace 間通信のリスク

- デフォルトで Namespace 間で通信できてしまう
- 攻撃を受けるPod
  - ポートを開けているコンテナ
  - API Call を待ち受けているコンテナ



Qmonus Value Stream のセキュリティ対策

セキュリティ強化のため、任意のコンテナを起動できるリスク、Namespace 間通信のリスク対策を実施

任意のコンテナを起動できるリスク

Namespace 間通信のリスク

任意のコンテナを起動できる状態を対策するには

やらなきゃいけないこと

- 起動できるコンテナの基準の決定
- 起動NGな場合にコンテナの起動を阻止する仕組みの導入

監視

コンテナ

コンテナ

起動を阻止

コンテナの基準の検討

Pod Security Standards とは

Pod セキュリティを確保するために推奨されるPodのセキュリティ規定
以下の3つが定義されている

- Privileged
  - 制限なし
  - 可能な限り幅広いレベルの権限を提供
  - 特権コンテナの利用可能
- Baseline
  - 最小限の制限
  - デフォルトの 最小限の Pod 構成を許可
  - 特権コンテナの利用禁止
- Restricted
  - 最も厳しい制限

Pod Security Standards
を参考にコンテナの基準を決定

https://www.valuestream.gmonus.net/

コンテナの起動を阻止する仕組みの検討

方式の検討

- Pod Security Admission
  - Kubernetes の標準で利用可能
  - 定義されたセキュリティレベル（Privileged、Baseline、Restricted）で適用

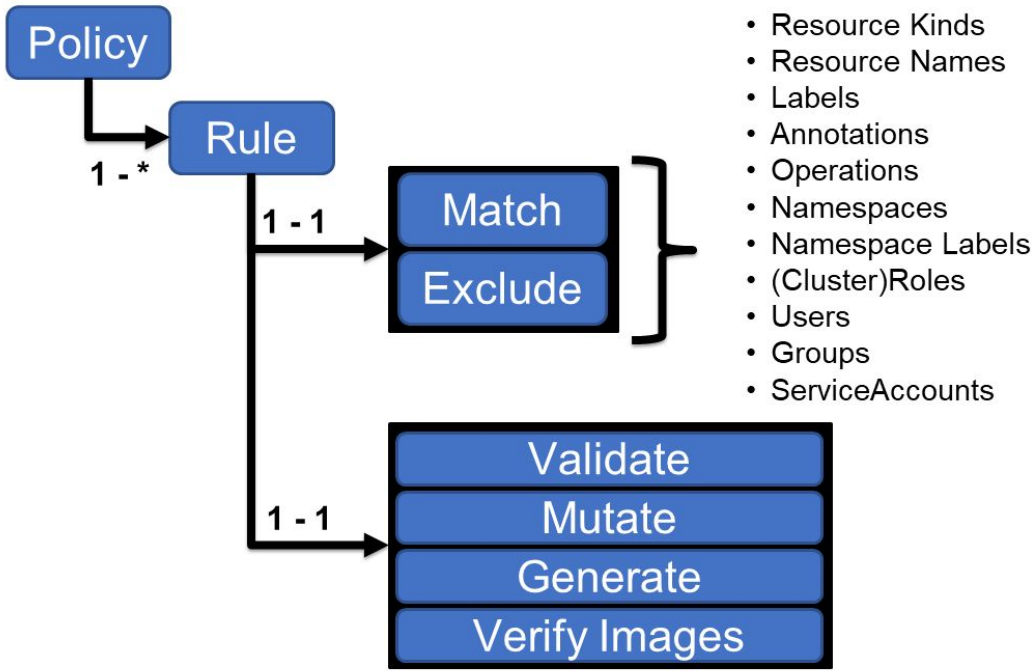
- Open Policy Agent
  - PSA と比較して柔軟な設定ができる
  - Rego 言語でポリシーを書く必要があり学習コストが高い

- Kyverno
  - PSA と比較して柔軟な設定ができる
  - Yaml 形式でポリシーを書くことができ学習コストが低い

学習コストも低く、今後より高度なセキュリティ強化が実施できそうな観点から Kyverno を採用

Kyverno とは

- Kubernetes 専用に設計されたポリシー エンジン
- YAMLでポリシーを書くことができ、ポリシーを Kubernetes リソースとして扱うことができる



- Resource Kinds
- Resource Names
- Labels
- Annotations
- Operations
- Namespaces
- Namespace Labels
- (Cluster)Roles
- Users
- Groups
- ServiceAccounts

主な機能

Validate
リソース構成がポリシーに準拠しているかどうかを確認する

Mutation
作成時にリソース構成を変更する、既存のリソース構成を変更する

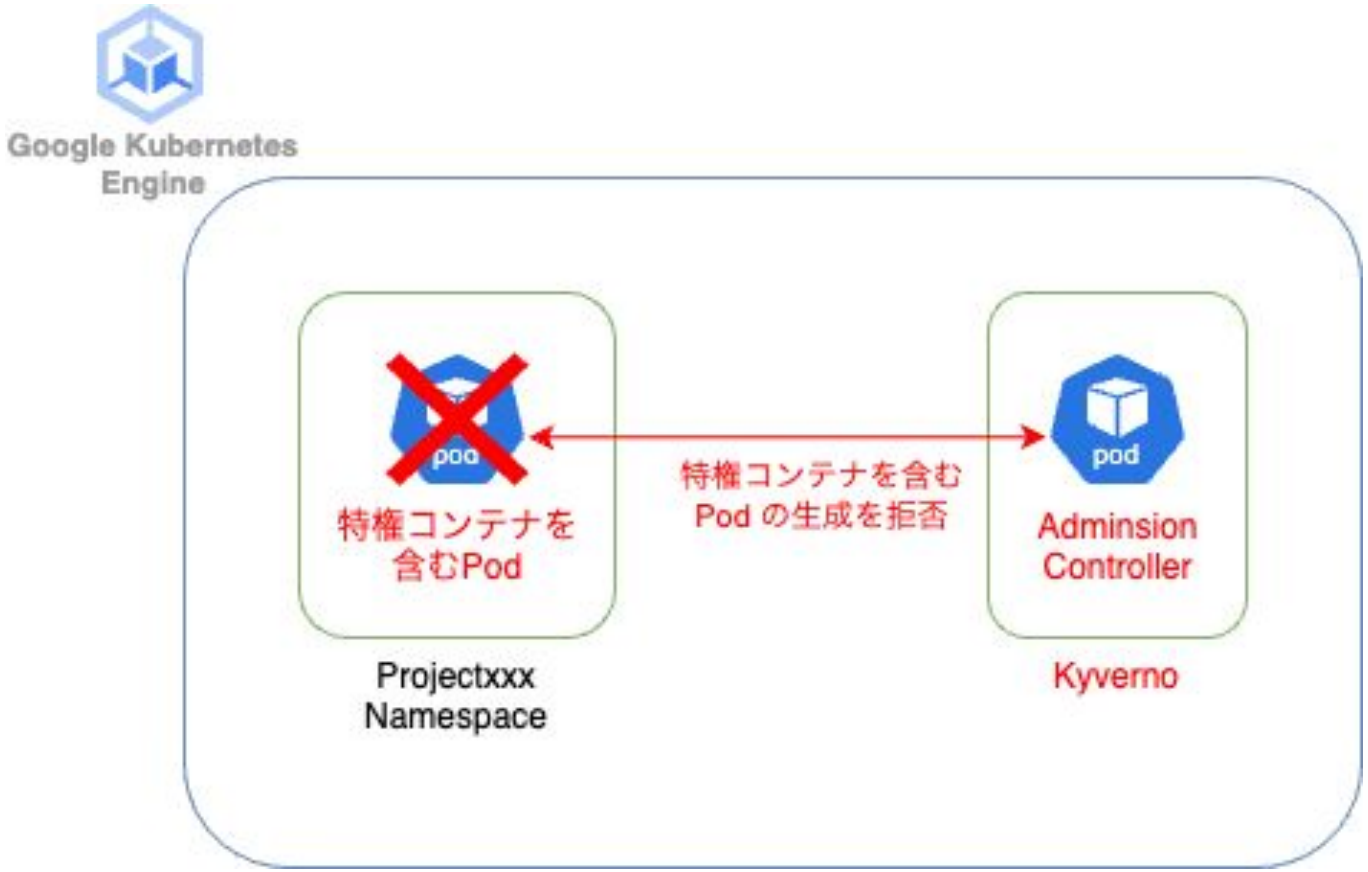
Generation
ポリシーに基づいてリソースを作成する

Verify Images
コンテナイメージの署名と証明書を確認する

https://kyverno.io/docs/kyverno-policies/

ポリシー違反した Pod の制限を実施

- 導入方法
  - Helm を利用して Kyverno をクラスターにインストール
  - ポリシーを記載した Yaml をクラスターに適用することで利用可能
- Validation ルール
  - 決定したコンテナ基準に則り Validation を行う
- モードの選択
  - Enforce モード
  - ポリシーに違反した Pod の生成不可



Kyverno による Validation イメージ

任意のコンテナを起動できるリスクに対処

- Pod Security Standards を参考にコンテナの基準を決定
- Kyverno の Validation ルールを利用して、特権コンテナ等の利用を制限

任意のコンテナを起動できるリスク

対策

Kyverno の Validation Rule を利用し、ユーザの Pod を制限

Namespace 間通信のリスク

Namespace 間通信のリスクに対処するには

Kubernetes 内のネットワーク制御は、Kubernetes 標準の
Network Policyで行うことができる

Network Policy とは

- Kubernetes 内のトラフィックフローを制御
  - TCP、UDP、SCTP プロトコルの IP アドレスまたはポートで設定
- ホワイトリスト形式で Ingress, Egress を許可する
  - except による除外設定を追加することが可能
  - ブラックリスト形式は利用できない

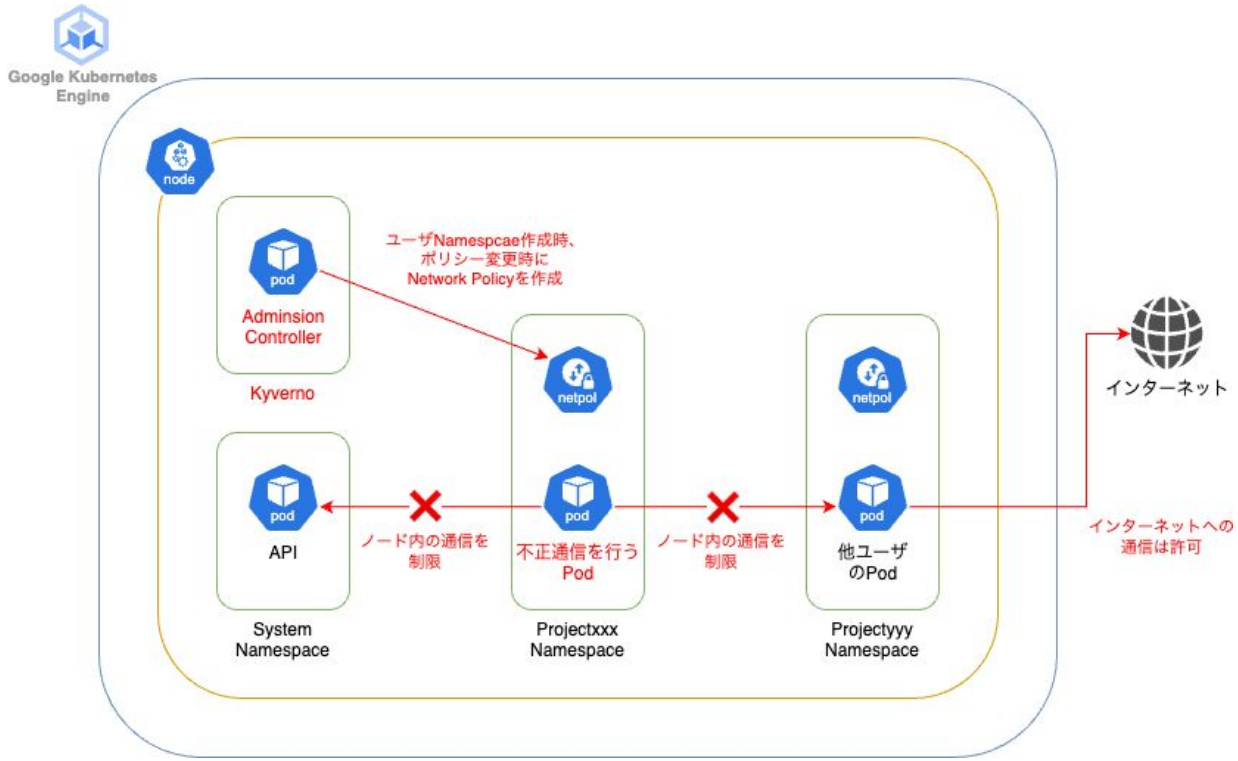
```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: test-network-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      role: db
  policyTypes:
  - Ingress
  - Egress
  ingress:
  - from:
    - ipBlock:
        cidr: 172.17.0.0/16
        except:
          - 172.17.1.0/24
    - namespaceSelector:
        matchLabels:
          project: myproject
    - podSelector:
        matchLabels:
          role: frontend
    ports:
      - protocol: TCP
        port: 6379
  egress:
  - to:
    - ipBlock:
        cidr: 10.0.0.0/24
    ports:
    - protocol: TCP
      port: 5978
```

ポリシーサンプル

https://kubernetes.io/docs/concepts/services-networking/network-policies/

Pod の通信制限を実施

- 導入方法
  - GKE 上で Calico ネットワークポリシーを有効化
  - 対象の Namespace に Network Policy を適用することで利用可能
- Network Policyのリソース管理方法
  - Kyverno Generation Rule を利用
  - ユーザ Namespace 作成/ポリシー更時に自動更新
- Pod の通信を制限
  - インターネットへの通信を許可
  - ノードのIPレンジを除外し、他 Namespace への通信を制限



Network Policy による通信制限イメージ

Namespace 間通信のリスクに対処

- Kyverno Generation Rule で Network Policy を管理
- Network Policy で Namespace 間通信のリスクに対処

任意のコンテナを起動できるリスク

対策

Kyverno の Validation Rule を利用し、ユーザの Pod を制限

Namespace 間通信のリスク

対策

Network Policy による Namespace 間通信の制限

学び ～よかったこと～

- 特定の Prefix を持つ Namespace に対するポリシー適用が簡単にできた
  - Kyverno のポリシーを適用する対象にワイルドカードが指定できる
- Kyverno の Validation Rule を利用して手軽に Pod に対する制限をかけることができた
  - Kyverno のリポジトリーに PSS（Pod Secuirty Standard）のテンプレートも用意されているため実装は簡単
  - 中身は Yaml であるため理解しやすく、カスタマイズもしやすい
- Kyverno の Generation Rule を利用することで運用負荷を軽減できた
  - ユーザが増えた場合や、ポリシーを変更した場合でも自動で適用できる構成を取ることができた

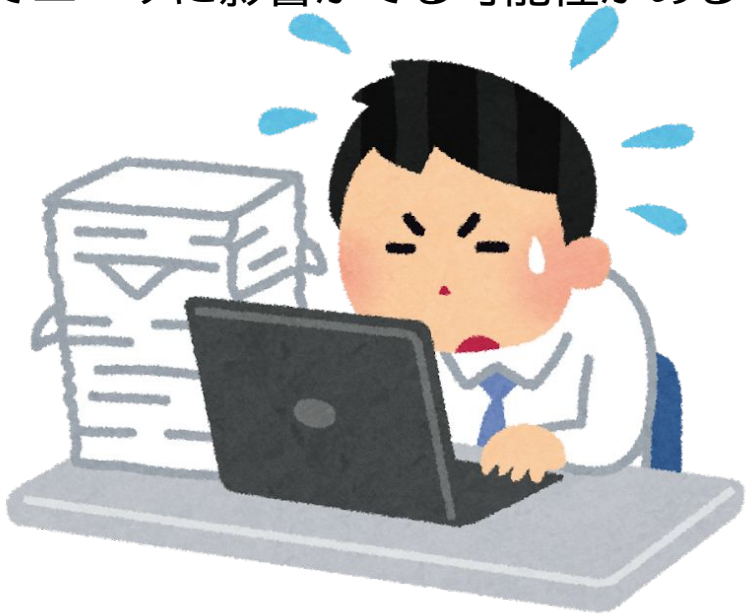
```yaml
apiVersion: kyverno.io/v1
kind: ClusterPolicy
Metadata:
 name: add-networkpolicy
 annotations:
  ...省略
spec:
 generateExistingOnPolicyUpdate: true
 background: false
 rules:
 - name: add-netpol-egress-deny
   match:
     any:
     - resources:
         kinds:
           - Namespace
         namespaces:
           - user-*
   generate:
     kind: NetworkPolicy
     apiVersion: networking.k8s.io/v1
     name: deny-all-egress
     namespace: "{{request.object.metadata.name}}"
     synchronize: true
     data:
       metadata:
         labels:
           created-by: kyverno
       spec:
         podSelector: {}
         policyTypes:
         - Egress
```

user- で始まる Namespace に対して、Network Policy を作成するサンプル

学び　～大変だったこと～

- ホワイトリスト形式の許可設定と除外設定でいかに制限をかけるかの工夫が必要
  - Network Policy はブラックリスト形式が利用できないため、ホワイトリスト形式で頑張るしかない
  - 直接他 Namespace への通信を制限することはできないため、Pod が動くネットワークアドレスを意識する必要がある

- ポリシーによる制限でユーザに影響が出ないかは地道にテストが必要
  - Validation ルールはポリシーを適用し、それぞれのテストケースを書いてスクリプトで回しテストした
  - Network Policy はきちんとテストしておかないと思わぬところでユーザに影響がでる可能性がある
    - テストをすることで kube-dns への通信が必要なことがわかった



ご清聴ありがとうございました

NTTコミュニケーションズ /
Qmonus Value Streamでブースもやっている
ので、お立ち寄りください