GKE を使ってやってみた！
少人数にて実践した
マイクロ サービス

株式会社10ANTZ

開発部 リードエンジニア
芳賀 幸一郎

会社紹介

VISION

ひらめきとモノづくりで、世界中に新たな感動を

MISSION

ファンとアーティストの“ありがとう”をつなぐ

会社紹介

モバイルゲーム事業

コンテンツ プランニング事業

オンライン コミュニケーション アプリ

Agenda

- マネージド サービスの活用
- マイクロ サービス
- Google Cloud を選択して良かったこと

マネージド サービスの活用

開発時間を捻出

- 複数のゲームタイトルを運用中
  - エンジニア リソースをそこまで割くことができない
- 多くのサービスを開発する予定がある
  - 環境構築の手間は減らしたい
- マイクロ サービスのノウハウを蓄積する時間を確保したい
  - 10ANTZ では初めてのマイクロ サービスということもあり、試行錯誤する時間が欲しい

開発時間を捻出

- Compute
  - Kubernetes Engine

- Database
  - Cloud SQL(MySQL)
  - Cloud Memorystore(Redis)
  - Cloud Firestore(native mode)



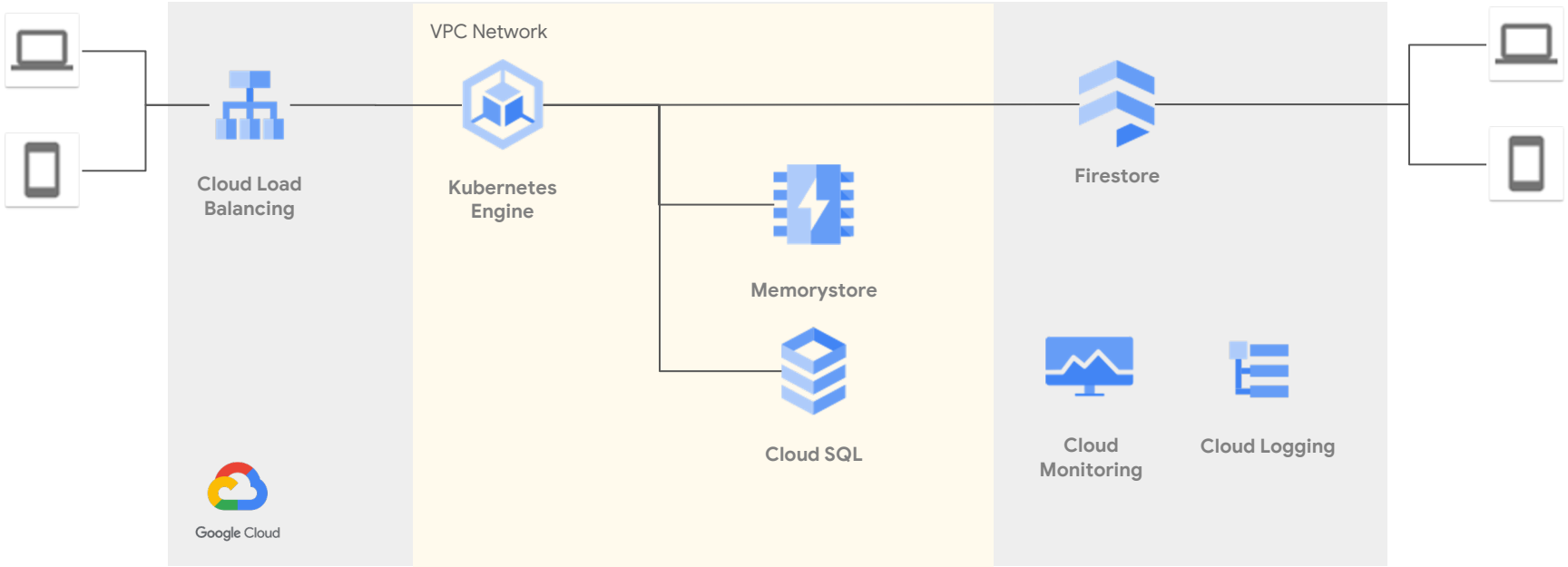
- Operations
  - Cloud Logging
  - Cloud Monitoring


IaC は Deployment Manager を採用

IaC は Deployment Manager を採用

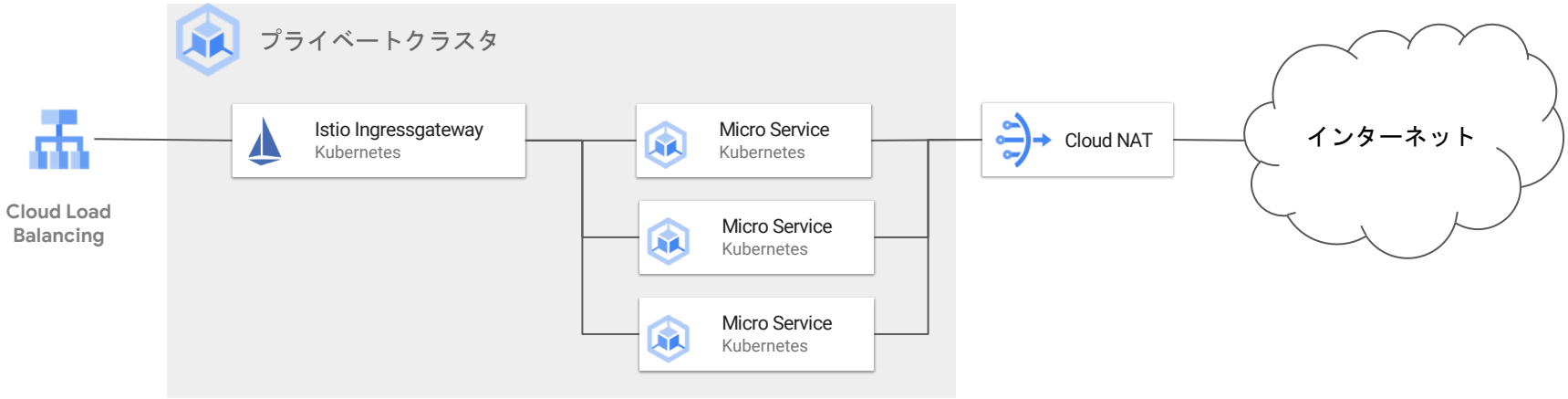
IaC は Deployment Manager を採用

サーバ構成


VPC Network
Cloud Load Balancing
Kubernetes Engine
Memorystore
Cloud SQL
Firestore
Cloud Monitoring
Cloud Logging
Google Cloud

マイクロ サービス

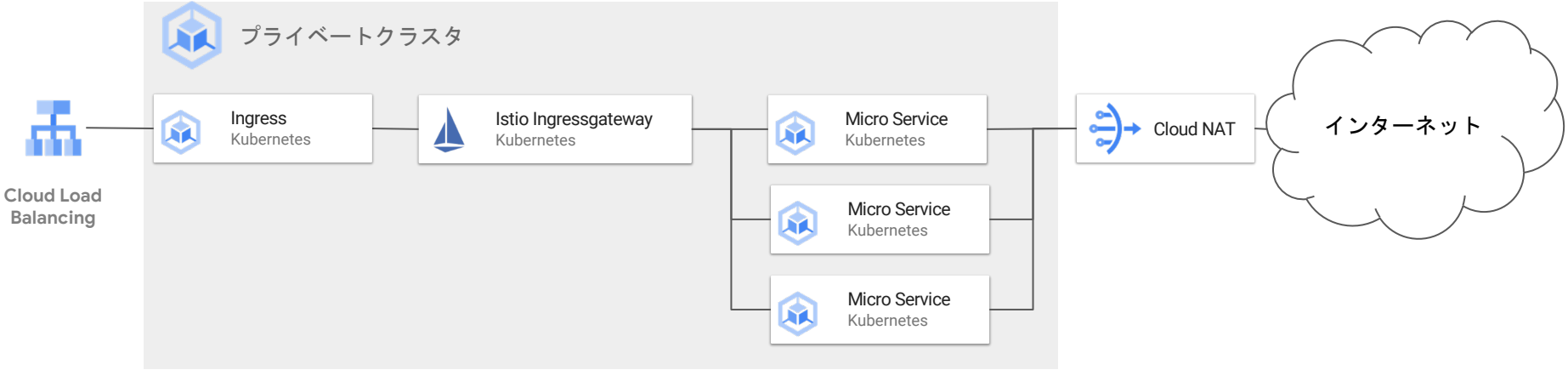
Istio を利用した Service Mesh



Istio を利用した Service Mesh


```
❯ $ kubectl get svc --selector=app=istio-ingressgateway \
···❯ $ -n istio-system \
···❯ $ -o jsonpath='{.items[0].status.loadBalancer.ingress[0].ip}'
```

L7レイヤーの外部ロードバランサーを構築する



L7レイヤーの外部ロードバランサーを構築する

- ManagedCertificate

```yaml
apiVersion: networking.gke.io/v1
kind: ManagedCertificate
metadata:
  name: certificate-name
  namespace: istio-system
spec:
  domains:
    - bar.10antz.co.jp
```

```
> $ kubectl describe managedcertificate -n istio-system
Name:         [redacted]
Namespace:    istio-system
Labels:       <none>
Annotations:  <none>
API Version:  networking.gke.io/v1
Kind:         ManagedCertificate
Metadata:
  Creation Timestamp:  2021-01-13T05:34:02Z
  Generation:          3
  Resource Version:    449947
  Self Link:           /apis/networking.gke.io/v1/namespaces/istio-system/managedcertificates/[redacted]
  UID:                 28881cf6-d83a-4907-ad9c-520c2e430d2f
Spec:
  Domains:
    [redacted]
Status:
  Certificate Name:    mcrt-323c6cc0-9cfe-4039-8189-11405a58ba5e
  Certificate Status:  Active
  Domain Status:
    Domain:     [redacted]
    Status:     Active
  Expire Time:  2021-04-12T21:43:30.000-07:00
Events:         <none>
```

L7レイヤーの外部ロードバランサーを構築する

- BackendConfig

```yaml
apiVersion: cloud.google.com/v1
kind: BackendConfig
metadata:
  name: ingress-backendconfig
  namespace: istio-system
spec:
  healthCheck:
    requestPath: /healthz/ready
    port: 15021
    type: HTTP
  securityPolicy:
    name: cloud-armor-policy-name
```

アクション	タイプ	一致	説明	優先度
許可	IP アドレス / 範囲	[redacted]	ip list of 10antz office	0
拒否（403）	IP アドレス / 範囲	*（すべての IP アドレス）	デフォルトのルール。優先度が高いとオーバーライドされます	2,147,483,647

L7レイヤーの外部ロードバランサーを構築する

- Ingress

```yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: gke-ingress-name
  namespace: istio-system
  annotations:
    networking.gke.io/managed-certificates: certificate-name
    kubernetes.io/ingress.allow-http: "false"
spec:
  backend:
    serviceName: istio-ingressgateway
    servicePort: 80
```

L7レイヤーの外部ロードバランサーを構築する


```
❯ $ kubectl get ingress \
⋯❯ $ -n istio-system \
⋯❯ $ -o jsonpath='{.items[0].status.loadBalancer.ingress[0].ip}'
```

マルチテナント構成

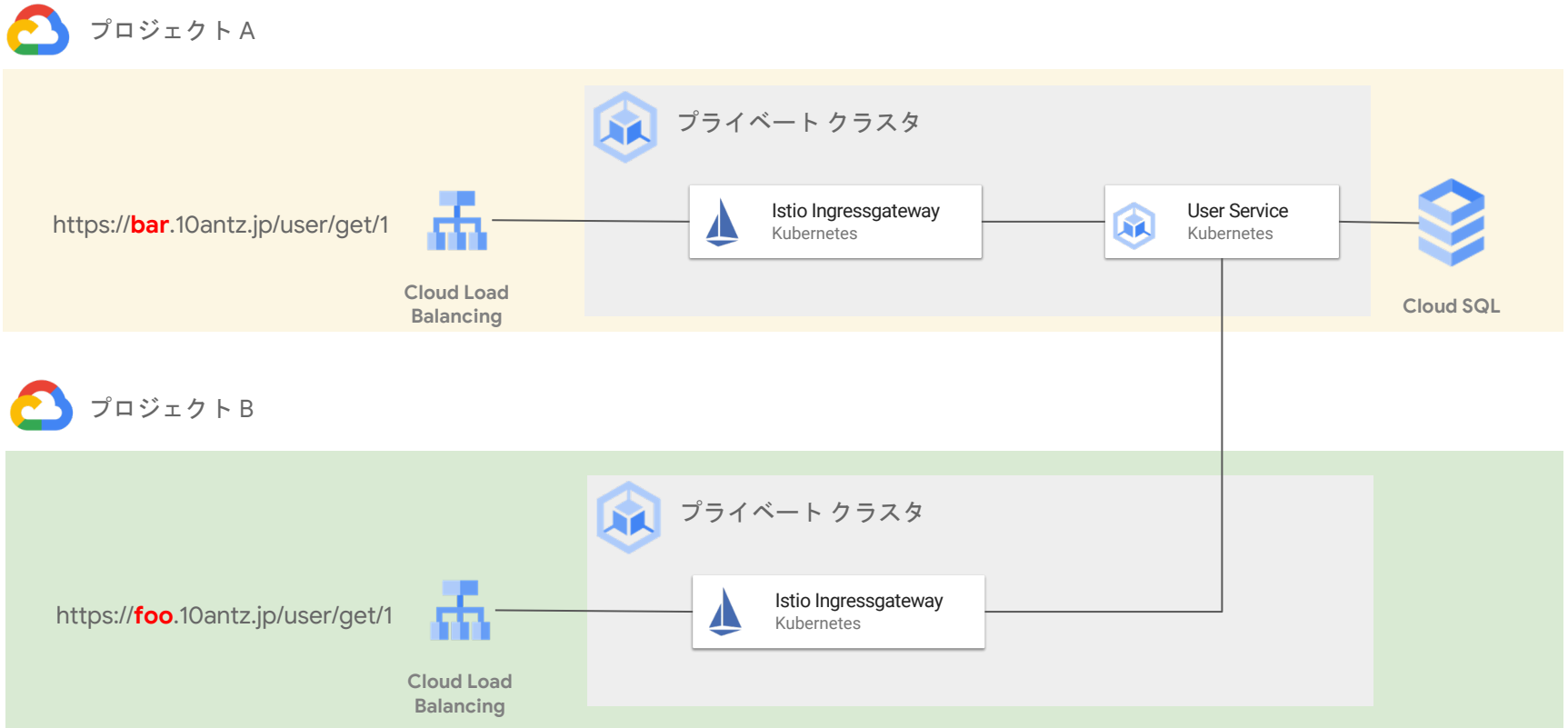
- Google Cloud プロジェクトを分けるか？
  - サーバ費用はオンラインコミュニケーションアプリとは別で計上したい
  - 一部機能はデータ含め共通化したい

結論: Google Cloud プロジェクトを分ける

マルチテナント構成

- どうやって一部機能を共通化するか
  - 同じコンテナをデプロイする？
    - Container Registry は Cloud Storage の Read 権限を持ったサービス アカウント を利用すれば問題ない
    - デプロイ先の管理コストが増える
  - Cloud SQL / Cloud Memorystore への接続は？
    - VPC ネットワークが異なる状態

マルチテナント構成（理想）



マルチクラスタ サービス メッシュ

ホーム › ソリューション

評価とクチコミ

フィードバックを送信

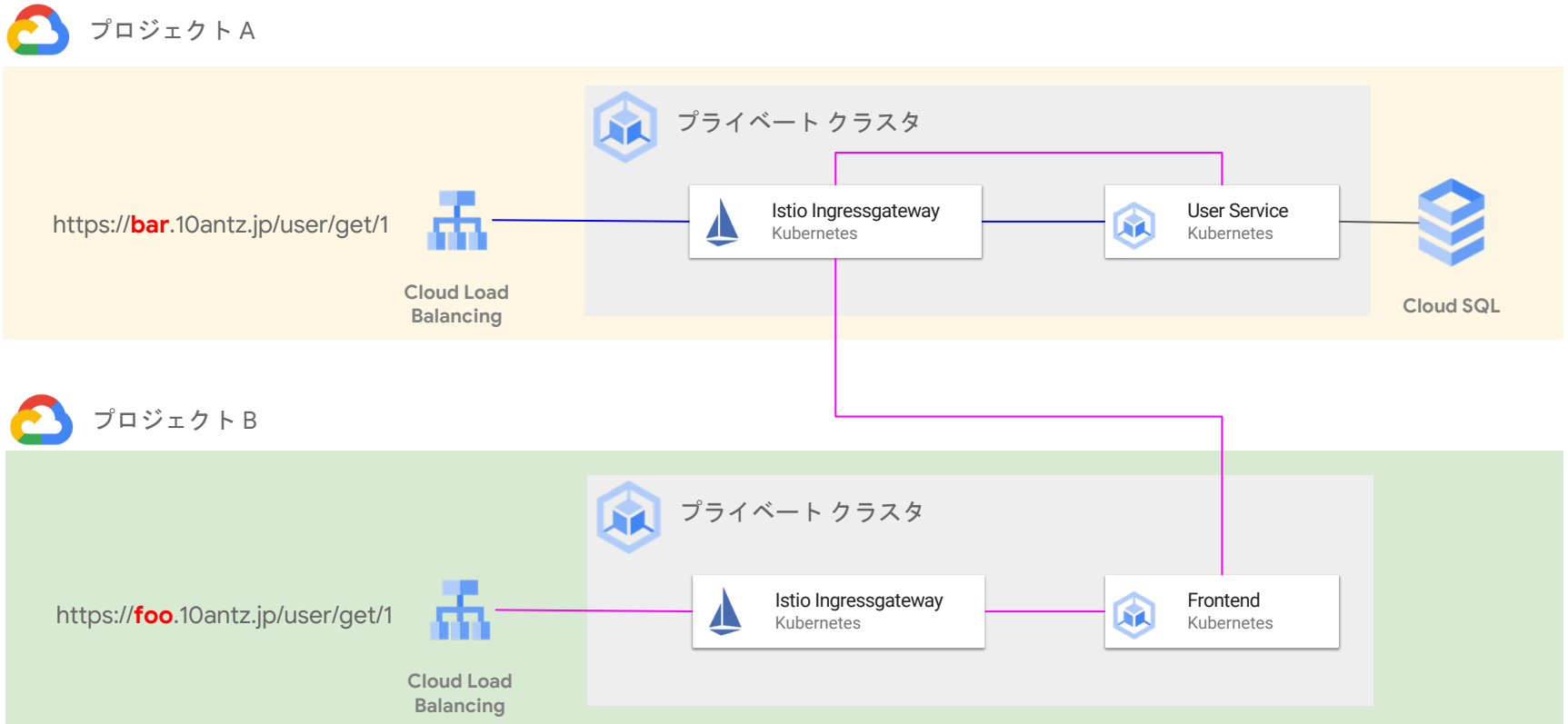
レプリケートされたコントロール プレーン アーキテクチャを使用して GKE でマルチクラスタ サービス メッシュを構築する

このチュートリアルでは、Istio マルチクラスタ サービス メッシュを使用して、複数の Kubernetes クラスタにアプリケーションをデプロイする方法について説明します。Istio マルチクラスタ サービス メッシュを使用すると、複数の Kubernetes クラスタ上で実行されているサービス間で安全に通信を行うことができます。Kubernetes クラスタはさまざまな場所で実行されています。Google Cloud で実行される Google Kubernetes Engine（GKE）クラスタや、オンプレミスのデータセンターで実行される Kubernetes クラスタなど、異なるクラウド プラットフォームで実行されている場合もあります。

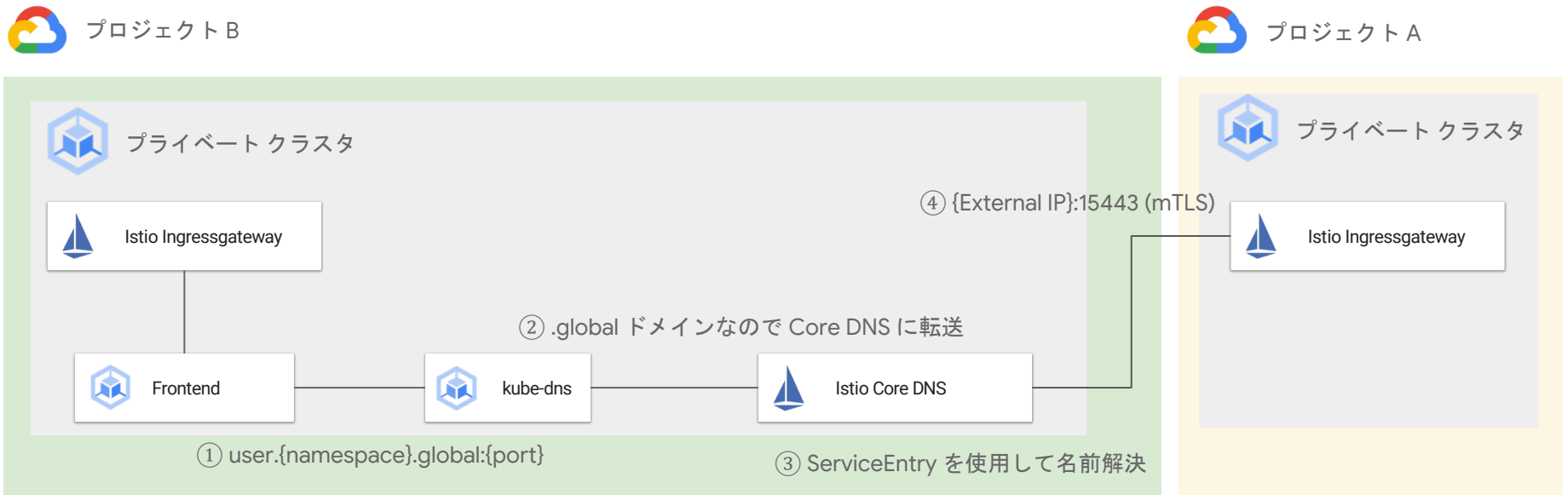
このチュートリアルでは、複数のネットワークにまたがる複数の GKE クラスタにサービス メッシュを作成する方法を説明します。このチュートリアルでは、Deployment、Service、Ingress など、Kubernetes の基本的な知識が必要になります。Istio の基礎知識は必要ありませんが、あったほうが内容を理解しやすくなります。

Istio は、Kubernetes クラスタで実行される Service の検出、動的ルーティング、安全な接続を行うためのオープンソースのサービス メッシュです。Istio は、メッシュ内でルーティング、負荷分散、スロットリング、テレメトリ、サーキット ブレーク、認証、サービス呼び出しの認可を行うためのポリシーベースのフレームワークを提供します。これにより、アプリケーションのコードをほとんど変更せずに、これらの機能を実現できます。Istio を Kubernetes クラスタに

マルチクラスタ サービス メッシュ



マルチクラスタ サービス メッシュ



マルチクラスタ サービス メッシュ

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: kube-dns
  namespace: kube-system
data:
  stubDomains: |
    {"global": ["[redacted]"]}
```

Istio CoreDNS の Cluster IP

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: ServiceEntry
metadata:
  name: user-service-entry
spec:
  hosts:
  - user.[redacted].global
  location: MESH_INTERNAL
  ports:
  - name: grpc
    number: [redacted]
    protocol: GRPC
  resolution: DNS
  addresses:
  - [redacted]
  endpoints:
  - address: [redacted]
    ports:
      grpc: 15443 # Do not change this port value
```

注意点

- Istio v1.7 系からこのソリューションが使えなくなっていた
  - https://github.com/istio/istio/issues/26990
  - Istio v1.7 で、このソリューションのテストが十分にされていなかったみたい

Google Cloud を選択して良かったこと

簡単に利用できるマネージド サービス

- マネージド サービスを利用して簡単にマイクロ サービスの環境を作れる
  - 少ない人数でもインフラ構築に費やすコストを削減することができた
  - モニタリング、ロギングが Cloud Monitoring / Cloud Logging に統合されているため開発時の不具合調査が楽だった
  - Deployment Manager を利用すれば同じ環境をすぐに作成できた

結果的に、開発者はサービス開発に時間を費やすことができた

デベロッパーに親切

- デベロッパーに寄り添ってくれる
  - Google Cloud はドキュメントが豊富
  - アーキテクチャ相談や Google Cloud の製品についての質問にすぐに回答してくれる
  - 事例やブログとあわせて、「E.G.G. Japan （ゲーム業界向け特別トレーニング）への招待」

- 積極的にセミナーや勉強会に招待してくれる
  - パートナー企業が開催してるハンズオン トレーニングを紹介してくれる
  - 今回のアーキテクチャで参考になる過去の Cloud OnAir を探してくれた

営業の方やカスタマーエンジニアの方が親身になって相談に乗ってくれる

まとめ

- 開発に専念できる環境作りを行い、時間を確保した
  - Cloud SQL / Cloud Memorystore といったマネージド サービスを利用
    - インフラ管理コストの削減
  - Cloud Monitoring / Cloud Logging の活用
    - 運用・監視ツールの導入コスト削減
  - Deployment Manager を利用した環境構築
    - インフラ環境構築の手間を削減

- GKE を利用したマイクロ サービス
  - GKE と OSS の Istio を利用するだけで、簡単にマイクロサービスを体感できる
  - GKEには Network Endpoint Group があり、External LB から直接 pod にバランシング可能

今後の展望

- Anthos Service Mesh の利用
  - 今回 Istio を利用していたが、Anthos Service Mesh に切り替えることを検討
    - コントロール プレーンが分離されて Google の管理となる
      - Istio のバージョンアップでマルチクラスタ設定が使えなくなる、といった事がなくなるので期待
    - Service Mesh の可視化

- Cloud Spanner の利用
  - 書き込み分散
  - node のスケーラビリティ

We are Hiring!!