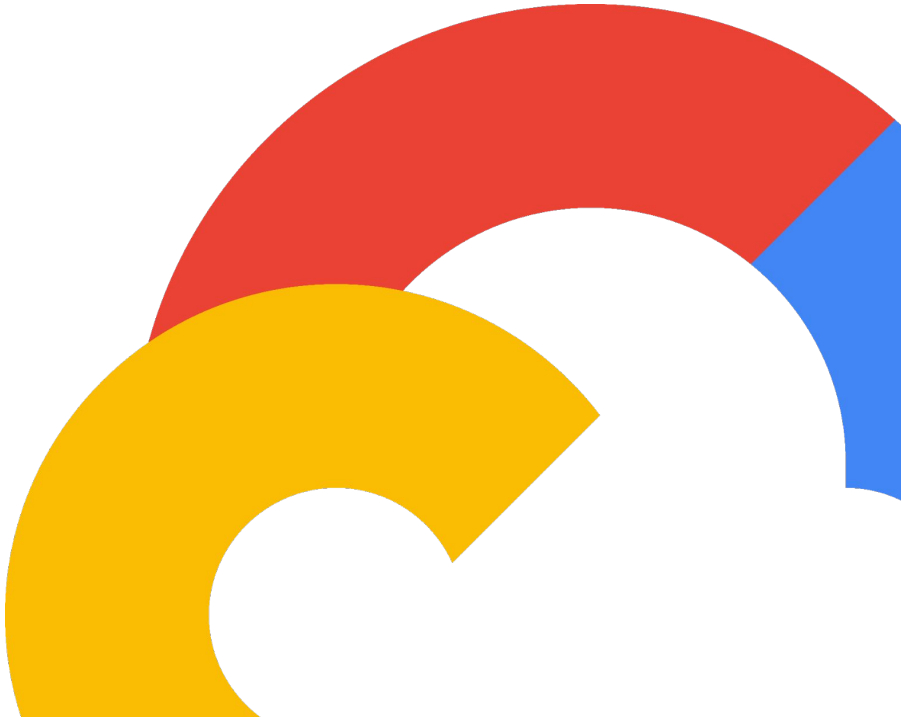
GKE 大規模活用術 おすすめ機能と 選び方

March 1, 2024

クラウドエース株式会社
/ Champion Innovator
間瀬 真

Google Cloud



Contents

01
自己紹介

自己紹介

間瀬 真
(@Makocchan_Re)

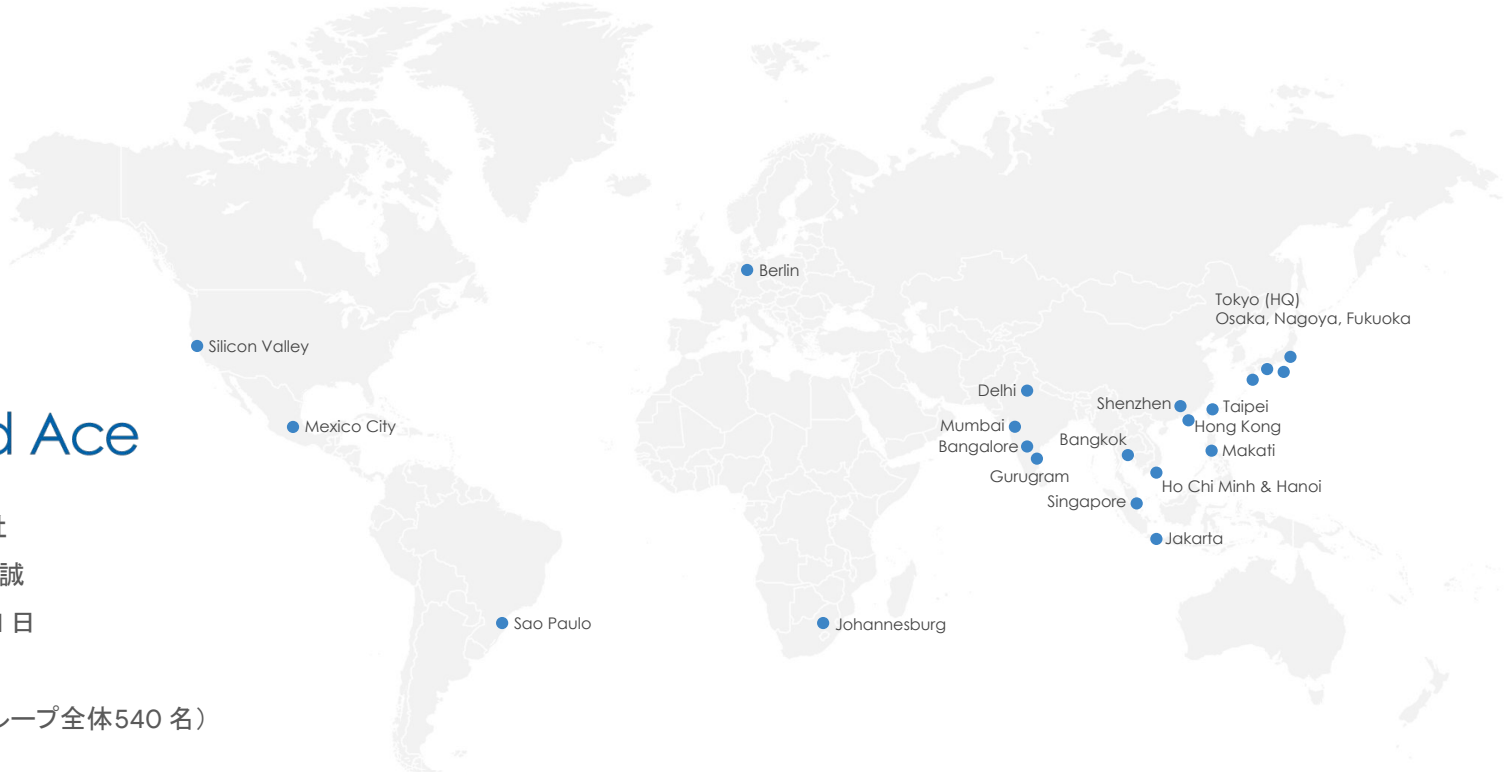
クラウドエース株式会社

技術部 開発統括部SRE 部 アシスタントマネージャー

- 業務内容
  - Google Cloud を中心としたコンサル、インテグレーションに従事
  - 主にコンテナ領域を専門としながらアプリ、インフラ、マネジメント全般を担当
- その他
  - リモートでの資格試験では自宅の浴室で受験
  - Google Cloud Champion Innovators (Modern Architecture, Serverless App Development)
  - Google Cloud Partner Top Engineer 2024

☜最近浴室受験で取得

02
会社紹介



Cloud Ace

クラウドエース株式会社

代表取締役社長　青木誠

設立　:2016 年 11 月 1 日

資本金　:1 億円

従業員数: 300 名（グループ全体540 名）

※2023 年 11 月時点

Google Cloud

4.アライアンス・マーケットプレイス オンボーディング

Google Cloud を活用したサービスの利用拡大を目的として、**パートナー製品**としての販売、Google Cloud マーケット プレイスへの登録などを支援します。

3.クラウド エンジニアの育成

現役のトップ エンジニア講師による **Google Cloud 認定トレーニング**を提供し、毎年数千人のエンジニアを育成しています。

2.Google Cloud を活用した開発の受託と伴走型支援

システムのクラウド移行やクラウド ネイティブ開発など、お客様のニーズに合わせて**コンサルティングおよび受託開発や内製化技術支援**をご提供します。

1. Google Cloud とサポートの提供

Google Cloud の請求代行と共に、特別な料金プランとテクニカル サポートを提供します。また、個別の課題に対する相談窓口として、カスタマー サクセス担当者もご支援します。

本日お話しすること

- GKE が提供する機能において、よく伺う課題より選択に迷いがちな点について解説します。
- これから大規模に GKE を利用される方向けの内容となります。

注意

- 本セッションの内容は 2024/2/16 時点のプロダクトの仕様や料金ルールに基づいています。

03
GKE の特徴と課題

Google Kubernetes Engine

- マネージドな Kubernetes サービス
- ユースケースに応じてクラスタモードの選択が可能
  - Standard
  - Autopilot

- Cloud Run では実現が困難または、複雑な構成となるユースケースにおいて選択することが多い

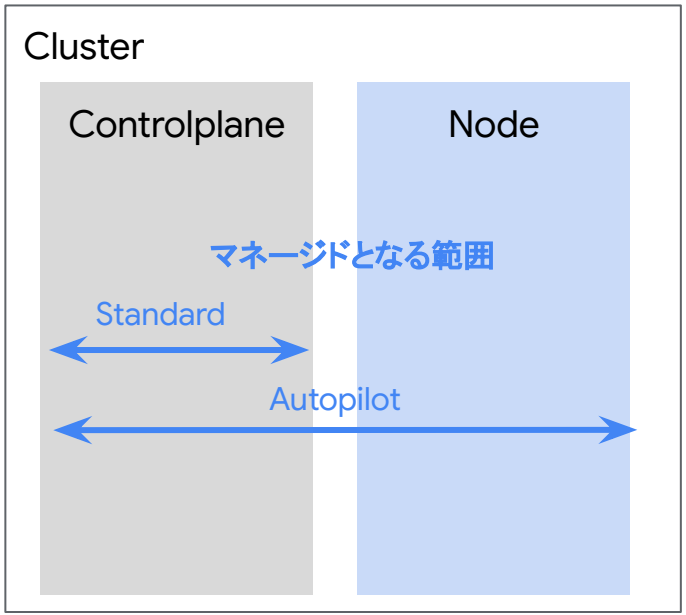
大規模での利用において課題となりやすい点と解決策候補

課題	解決策候補
クラスタの運用負荷	• GKE Autopilot • Enterprise エディション
アプリケーションをクラスタ外へ公開する方法	• Ingress • Gateway API • スタンドアロンNEG
サービス メッシュの導入要否	• Anthos Service Mesh • Dataplane V2

04
Autopilot

クラスタモードの違い

- Standard はノードをユーザで管理する。
- Autopilot はノードの管理が不要に。
  - Standard モードが抽象化されるイメージ
  - Pod に対する課金によってコスト最適化
  - 運用負荷が減る一方で制約もある

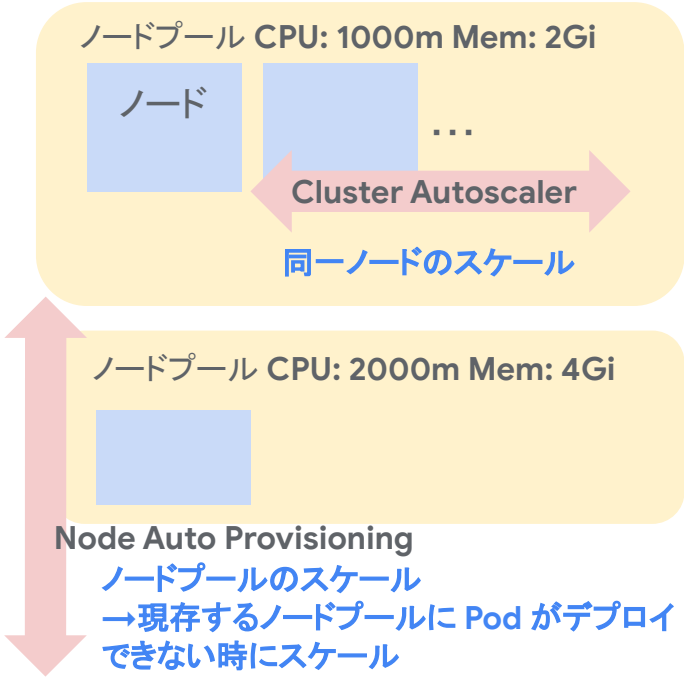


Standard vs Autopilot

比較要素	Standard	Autopilot
SLA	ゾーンクラスタ コントロール プレーン : 99.5% リージョン クラスタ コントロール プレーン :99.95% ノード: GCE の SLA が適用(99.9%~99.99%)	コントロール プレーン :99.95% 複数ゾーンにデプロイされた Pod : 99.9%
コスト	**ノードリソースに対する課金 +管理手数料**	**Podのリクエスト リソースに対する課金 +管理手数料**
ノードのスケールアウト・マシンサイズ変更	ユーザにて管理	**管理不要** - コンピューティング (汎用 / バランス等) を選択することが可能
ノード アップグレード	手動または自動実行 - サージまたは Blue/Green によるアップグレードを選択可能	**自動実行** - サージアップ グレードのみ - メンテナンス ウィンドウは設定可能
Pod ライフサイクル	ユーザにて管理	・特権モードでの実行不可 ・**QoS は Guaranteed のみ** ・ノードセレクタによるワークロードの分離や Pod エビクション時の停止までの時間延長は可能）

Autopilot におけるノードのプロビジョニング

- **Node Auto Provisioning と Cluster Autoscaler という機能が組み合わされている。**
- GKE ではノードをノードプールというグループで管理する。ノードプール内のノードは同じスペックとなる。
- Cluster Autoscaler は Pod の需要に合わせてノードプール内のノードをスケールする機能
- Node Auto Provisioning は Pod の需要に合わせてノードプール自体をスケールする機能

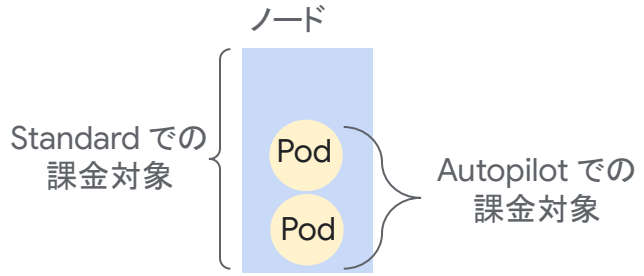


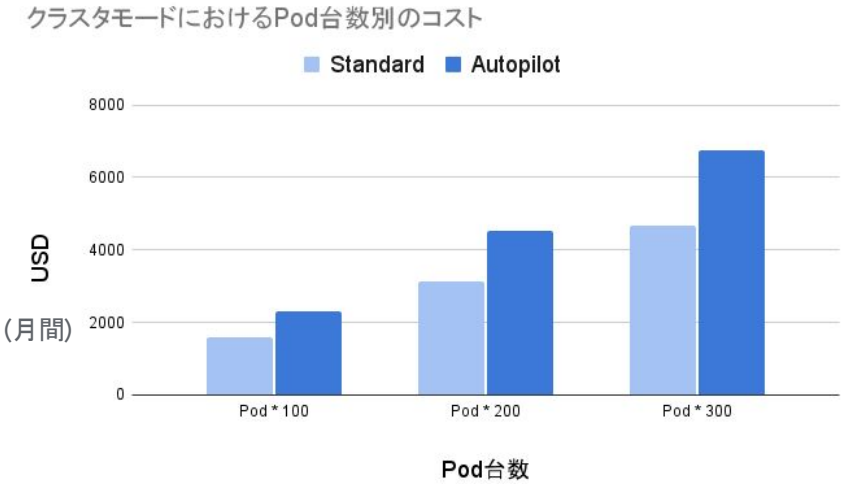
コスト

- Pod 台数別のコスト参考値
  - Pod の CPU / Mem = 500m / 512MiB
  - Standard モードでは必要なリソース量よりノード台数を試算
    ※リソースの余剰はほぼ無し
  - マシンタイプはバランス型

☞課金対象となるリソースの単価は Autopilot の方が若干高い

☞但し、Standard クラスタにおいてノードリソースに余剰がある場合は Standard クラスタの方が高くなる可能性がある。





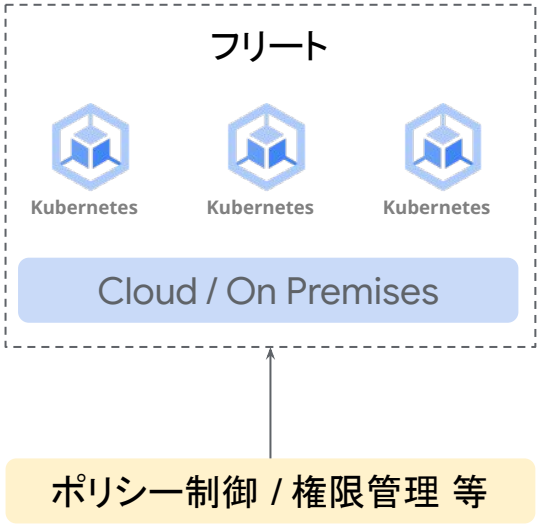
どちらを使うべきか

- **まずは Autopilot を選択できないか検討する。**
- Autopilot の制約を受け入れながら目的を達成できるか。
    - 例えば、以下のようなケースでは Standard モードを視野に
        - アップグレードを B/G や手動にて行う必要がある。
        - 大量のトラフィックを捌く必要がある且つ、トラフィックの増減が大きいワークロードで安定したレイテンシが求められる。※安定化のための改善は可能なので、検証した上で判断
- 開発環境や検証用途のクラスタのみ Autopilot を利用する選択肢もある。

05
Enterprise エディション

Enterprise エディション

- Google Cloud、他パブリック クラウドやオンプレミス上のクラスタを統合管理するためのエディション
- フリートと呼ばれる論理グループにクラスタを所属させて、グループ単位に管理することが可能に
- GitOps、ポリシー制御、クラスタに対する権限管理やセキュリティスキャンといった機能を提供



エディションはクラスタモードとは別物

- Standard モード、Autopilot モード共に Enterprise エディションが利用できる

クラスタモード/エディション	Enterprise エディション	Standard エディション
Standard モード	利用可能	利用可能
Autopilot モード	利用可能	利用可能

Enterprise エディションのユースケース

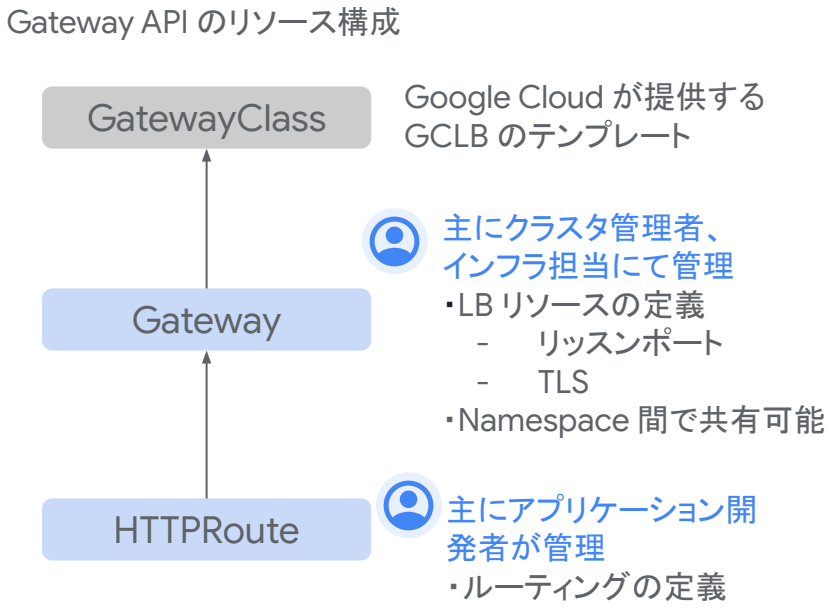
- **Google Cloud 内外でのマルチクラスタを運用する、セキュリティガバナンスをしたい**

- 上記以外で、Enterprise エディションの有効化が条件になっている機能を利用したい
  - Binary Authorization : デプロイするコンテナイメージの検証
  - Advanced Vulnerability Insights: アプリケーション パッケージの脆弱性スキャン
  - FQDN ネットワーク ポリシー : FQDN による通信の制御
  - ユーザー管理鍵によるノード間通信の暗号化
  - などなど

06
クラスタ外へアプリケーションを公開する方法

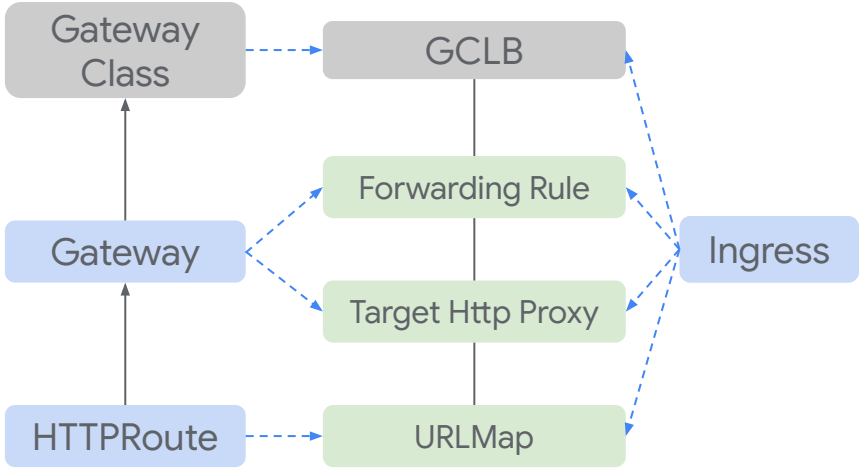
Gateway API

- GKE 上のアプリケーションをクラスタ外へ公開するための機能
- 公開に必要なGCLB リソースとアプリケーションへのルーティング ルールの
- リソースが分離されることで、チームの所掌が分離しやすくなった。
- パスや HTTP ヘッダ等によるルーティングやトラフィック分割などの機能を提供



Ingress

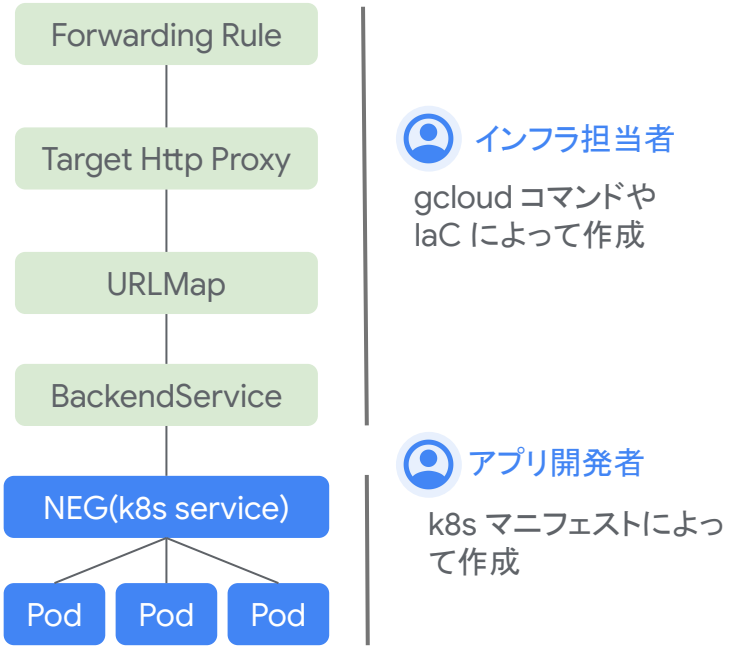
- GKE 上のアプリケーションをクラスタ外に公開するための機能でGateway APIよりも前から提供されている。
- 一つの Ingress リソースにインフラチームが管理したい GCLB リソースと、アプリチームで管理したいルーティング ルールを定義する必要があった。
- Namespace 毎に Ingress リソースが必要で共有することができなかった。



Gateway API と Ingress と GCLB リソースの対応付け

スタンドアロン NEG

- Gateway API や Ingress を使わずにクラスタ外にアプリケーションを公開する方法として、スタンドアロン NEG(Network Endpoint Group) を利用する方法がある。
- GCLB リソースを個別に作成して、GKE の Service リソースを NEG として紐付けることになる。
- 所掌の分離は可能だが、アプリケーション開発チームの関心が深そうなURLMap は分離しづらい。



Gateway API vs Ingress vs スタンドアロン NEG

比較要素	Gateway API	Ingress	スタンドアロン NEG
URI パスによるルーティング	可能	可能	可能
所掌の分離	可能	不可	一応可能
マルチクラスタへのルーティング	可能	不可	可能
カスタム リクエスト/レスポンス ヘッダ	可能	不可	可能
ヘッダ、メソッドによる高度なルーティング	可能	不可	不可
トラフィック分割	可能	不可	不可
トラフィック ベースの Pod オートスケール	可能	不可	不可

どれを使うべきか？

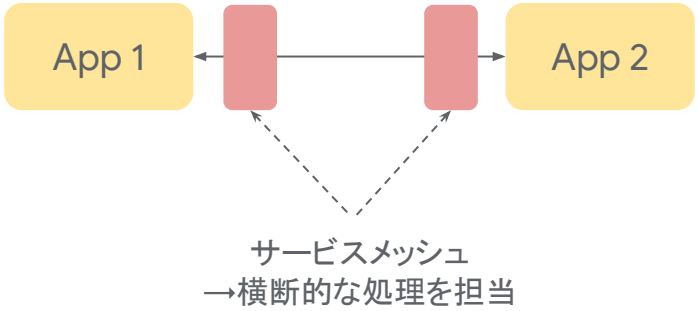
- これから GKE を利用する場合は Gateway API を推奨
- 移行を伴うが、Ingress や スタンドアロン NEG から Gateway API への変更は可能

07
サービス メッシュの導入要否

サービス メッシュの主な役割とユースケース

役割

- アプリケーション間で通信する際に、アプリケーションとは異なるレイヤで以下のような横断的な処理を行う。
  - トレース
  - ロギング
  - トラフィック分割
  - 通信の暗号化

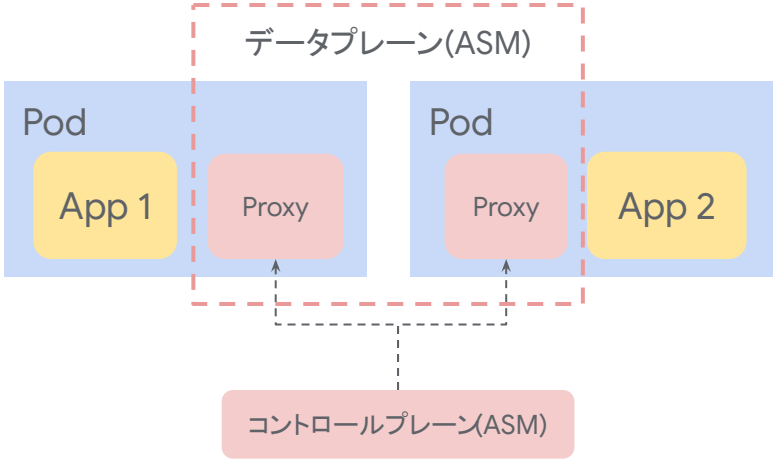


ユースケース

- サービス間の通信を可視化して依存関係を把握したい、ログを出力したい。
- マイクロ サービス間の通信制御 (許可/拒否)、通信の暗号化をしたい。
- マイクロサービス別に B/G、カナリア デプロイ、A/B テストをしたい。

Anthos Service Mesh

- Google Cloud が提供するマネージドなサービス メッシュ
- OSS の Istio がベースになっており、マネージドな範囲が異なるデプロイ オプションが 2 つある。
  - マネージド Anthos Service Mesh (ASM)
  - クラスタ内 ASM
- Istio が多くの機能を提供するため、前頁のユースケースを満たすことができる。



デプロイ オプションの大きな違いはコントロール プレーンを自分で管理するかどうか。

ASM 導入によってトレードオフとなる点

アップグレードによる運用負担

- GKE に加えて ASM のアップグレードが必要
  - マネージド ASM によって省力化は可能

Pod の複雑化

- Pod のサイドカー として Proxy コンテナが挿入されるため、Pod の構成がやや複雑になりアプリ開発チームが混乱しがち。
- 上記に伴い、Pod の起動時間が比較的長くなる。

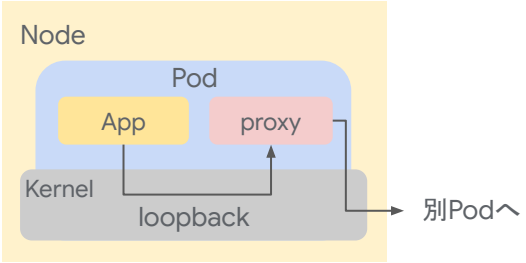
リソース増

- 1 Pod に対して 1 Proxy コンテナが追加されるため、クラスタにおいて必要なリソースが増加する。

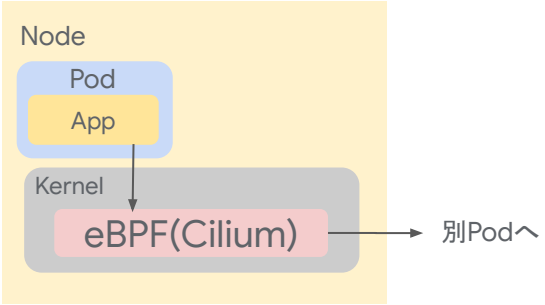
eBPF によるサイドカーレスなサービス メッシュ

- ユーザー プログラムを Linux Kernel で動作させることができる eBPF によって、プロキシ コンテナの役割を代替する。
- サイドカーが不要になることにより、Pod がシンプルになりコンテナ数も削減される。
- OSS の Cilium により提供されている。

サイドカー



サイドカーレス



GKE Dataplane V2

概要

- **Cilium をベースとした GKE のオプション機能**
- Cilium が提供する機能が徐々にサポートされている印象
- 現状、本機能に関する個別のアップグレードは不要

主な機能

Monitoring

- Hubble による サービスマップ やトラフィックのモニタリング (※1)
- Google Cloud Managed Service for Prometheus への指標公開 (※1)

Security

- ユーザー管理鍵によるノード間の通信暗号化 (※2)
- FQDN ネットワーク ポリシーによる FQDN での通信制御 (※2)

Networking

- eBPF による Pod 間通信におけるルーティング処理の効率化

(※1) 2024/2 時点では Public Preview
(※2) Enterprise エディションのみ利用可能

ASM vs GKE Dataplane V2

比較要素		ASM	GKE Dataplane V2
トラフィック管理	トラフィック分割	可能	不可
	http ヘッダ等による高度なルーティング	可能	不可
	サーキットブレーカー、リトライ、障害注入等	可能	不可
セキュリティ	通信暗号化	可能	可能(※下記 FQDN による制御機能と排他関係)
	サービス間通信の制御	可能	可能(FQDN による制御のみ)
	認証/認可	可能	不可
モニタリング	ロギング	可能	可能
	トレース	可能	不可
	サービス間通信の可視化	可能	可能
	メトリクスの公開	可能	可能(公開されるメトリクスは少ない)

どれを使うべきか？

- **現時点では排他的に選択するものではない。両方使うことも考える。(※)**
- **サービス メッシュが必要な場合は ASM を利用すべき**
  - **一方で、サービス間通信の依存関係だけ可視化できればよいのであれば Dataplane V2 のみでも充足する可能性はある。**

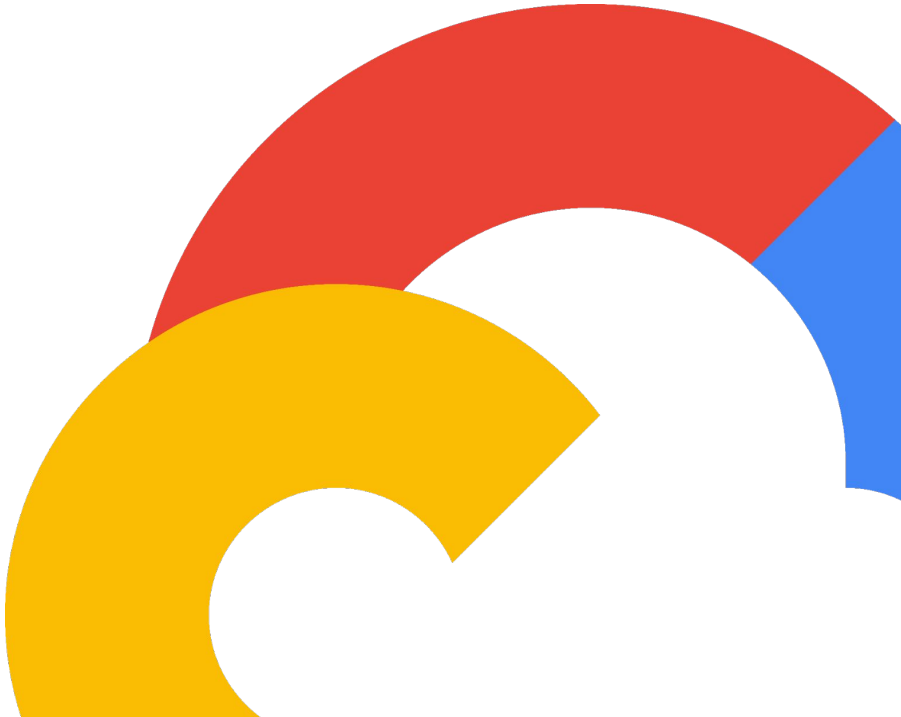
- 現状、Dataplane V2 はクラスタ新規作成時にしか有効化できないため、これから GKE を利用される方はとりあえず有効化しておくと今後の機能拡充にも備えられる。(現時点では有効化による追加コストなし)

※ 注意点として、Cilium はサイドカーレスなサービスメッシュを掲げていますが、Dataplane V2 についても同様かは特に言及されているわけではありません。

まとめ

Autopilot	● トラフィックの増減が大きくなく、安定したレイテンシを保証する必要がない。（判断する上で迷ったら検証することを推奨） ● 自動でのクラスタアップグレードを許容できる。 ● ノードのリソースやスケーリングの設計をしたくない。
Enterprise エディション	● Google Cloud 内外でのマルチクラスタを運用する、セキュリティ ガバナンスをしたい。 ● 上記以外で特定の機能（FQDN での通信制御等）を利用したい。
クラスタ外へのアプリケーション公開方法	● これから利用する場合において特に理由がなければ Gateway API を推奨
サービス メッシュ要否	● Dataplane V2 と ASM は現時点では排他的に選択しなくてよい。 ● サービス間の依存関係の可視化だけが目的なら Dataplane V2 を利用する。 ● サービス メッシュとしての多くの機能が必要なら ASM を利用する。

Thank you



Google Cloud