

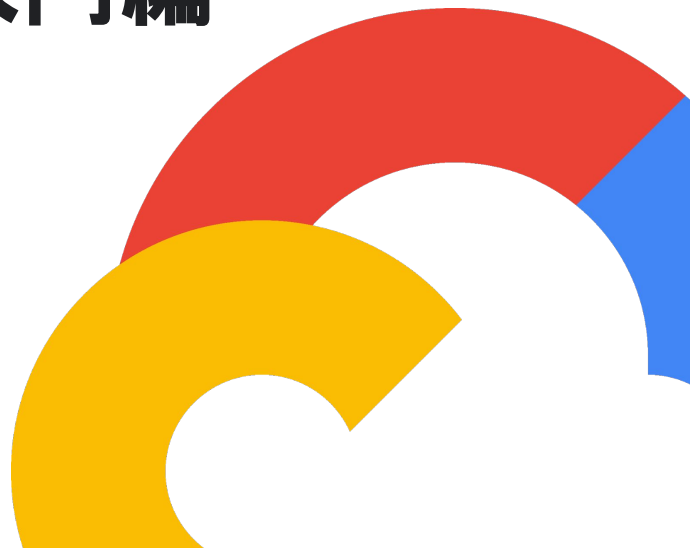
Google Cloud

実践 Google Cloud ハンズオン セミナー

Google Cloud で始める

Platform Engineering ~ 入門編 ~

2024 年 11月 7日



はじめるまえに

準備 - Qwiklabs アカウントの作成

事前に作成済の方は
実施不要です

1. <https://explore.qwiklabs.com/> にアクセスし、右上の [参加] をクリック



準備 - Qwiklabs アカウントの作成

事前に作成済の方は
実施不要です

2. 氏名・メールアドレス・会社とパスワードを入力して[アカウントを作成]をクリック

- **! 重要!!** メールアドレスは本ハンズオン登録時に入力いただいた Qwiklabs メールアドレスを入力してください

3. Qwiklabs へようこそという件名のメールが届いたら、本文中の[メールアドレスを確認]をクリック

Google Labs for Sales

アカウントを作成

Google アカウントでログイン

または

* 姓 * 名

* メール

* 会社

* パスワード * パスワードの確認

サービスについて不定期の更新情報、お知らせ、特典を受け取る

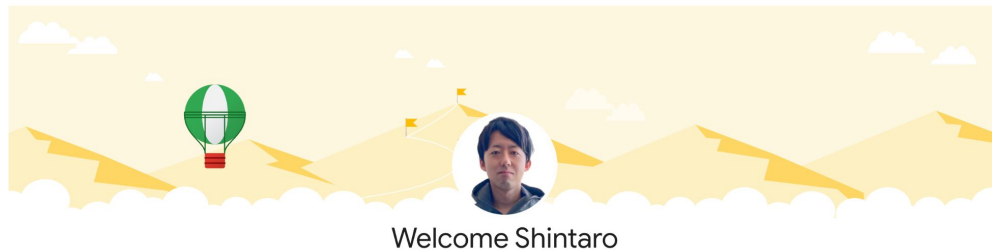
参加すると、Qwiklabs の [利用規約](#) と [プライバシー ポリシー](#) に同意したことになります。

代わりにログイン

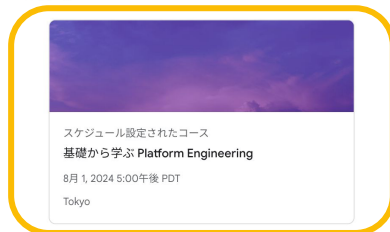
アカウントを作成

準備 - Lab の開始

4. <https://explore.qwiklabs.com/> にアクセス、ログインし、
[入門編: Google Cloud で始める Platform Engineering] をクリック

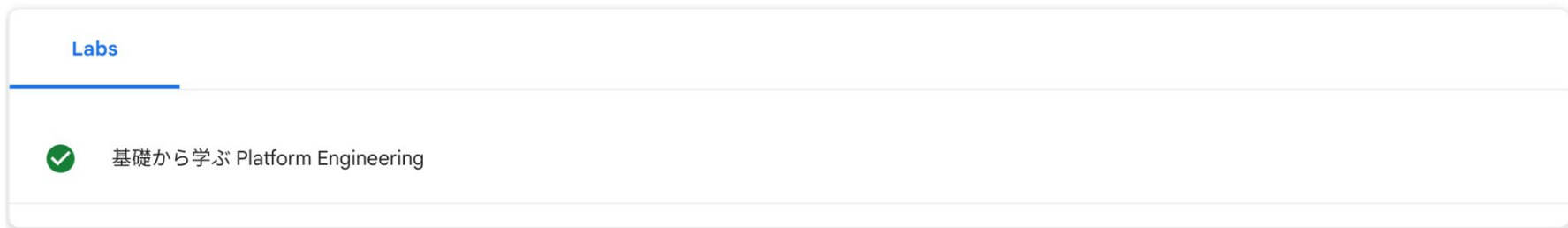


進行中



準備 - Lab の開始

5. Labs タブで [入門編: Google Cloud で始める Platform Engineering] をクリック



準備 - Lab の開始

6. [ラボを開始] をクリックし、ラボが起動

し表示が変わるのを待つ



基礎から学ぶ Platform Engineering

🕒 6時間 📺 No cost

☆☆☆☆☆ [ラボを評価する](#)

📌 This lab may incorporate AI tools to support your learning.

ラボ起動後の表示

A screenshot of the lab console interface. At the top, a red button says 'ラボを終了' (End Lab) and a timer shows '02:29:40'. Below is a warning message: '注意: Console では、ラボの手順どおりに操作してください。手順どおりに操作しないと、アカウントがブロックされる場合があります。詳細' (Note: In the Console, please operate according to the lab procedure. If you do not operate according to the procedure, your account may be blocked. Details). A button labeled 'Console を開く' (Open Console) is present. Below are fields for 'ユーザー名' (Username) with value 'student-04-4decd1522b5c', 'パスワード' (Password) with value 'Mggaqru6j0Ei', and 'プロジェクトID' (Project ID) with value 'qwiklabs-gcp-03-1d4978f'. Each field has a copy icon to its right.

※表示時間は異なる可能性があります。

準備 - Lab の開始



ブラウザのゲストモード
またはシークレットウィンドウで開
きます

4. [Console を開く] を右クリックし、[シークレットウィンドウで開く] を選択
5. ユーザー名とパスワードを入力し、[次へ] をクリック

ラボを終了 02:29:40

注意: Console では、ラボの手順どおりに操作してください。手順どおりに操作しないと、アカウントがブロックされる場合があります。 [詳細](#)

[Console を開く](#)

ユーザー名

student-04-4decd1522b5d

パスワード

Mggaqr6j0Ei

プロジェクト ID

qwiklabs-gcp-03-1d4978l

メールアドレスまたは電話番号

student-04-4decd1522b5d@qwiklabs.net

メールアドレスを忘れた場合

ご自分のパソコンでない場合は、ゲストモードを使用し
て非公開でログインしてください。
[ゲストモードの使い方の詳細](#)

アカウントを作成 [次へ](#)


パスワードを入力

Mggaqr6j0Ei

パスワードを表示する

パスワードをお忘れの場合 [次へ](#)

準備 - ハンズオンの開始

6. [ Cloud Shell をアクティブにする] をクリックし、画面下部に Cloud Shell のターミナルが開くのを待つ

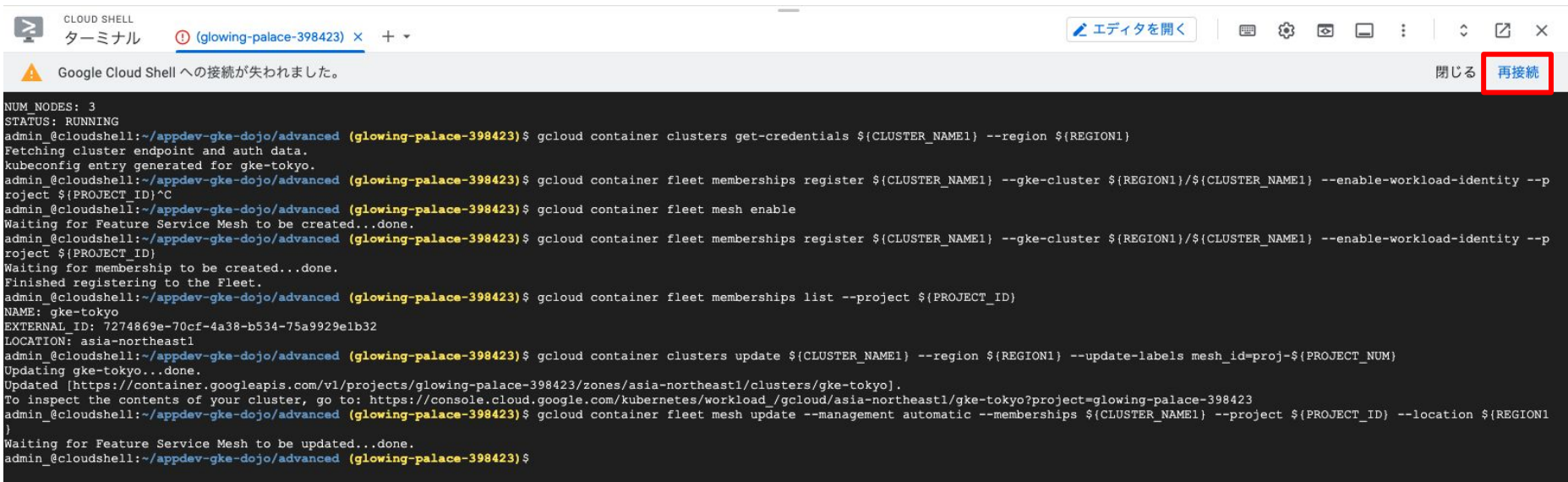


準備 - ハンズオンの開始

7. 以下のコマンドをコピーし、Cloud Shell ターミナルで実行してチュートリアルを開く

```
cd ~; git clone https://github.com/GoogleCloudPlatform/gcp-getting-started-lab-jp  
cd gcp-getting-started-lab-jp/pfe-basic-sep  
teachme tutorial.md
```

Cloud Shell への再接続



The screenshot shows a Google Cloud Shell terminal window. At the top, there is a title bar with 'CLOUD SHELL' and 'ターミナル (glowing-palace-398423)'. Below the title bar, a notification bar displays a warning icon and the text 'Google Cloud Shell への接続が失われました。' (Connection to Google Cloud Shell has been lost). To the right of this notification are two buttons: '閉じる' (Close) and '再接続' (Reconnect), with the latter button highlighted by a red rectangular box. The terminal content shows a series of commands and their outputs related to setting up a Kubernetes cluster. The commands include 'gcloud container clusters get-credentials', 'gcloud container fleet memberships register', and 'gcloud container fleet mesh update'. The terminal output shows the cluster name 'gke-tokyo' and its location 'asia-northeast1'.

```
NUM_NODES: 3
STATUS: RUNNING
admin@cloudshell:~/appdev-gke-dojo/advanced (glowing-palace-398423)$ gcloud container clusters get-credentials ${CLUSTER_NAME} --region ${REGION1}
Fetching cluster endpoint and auth data.
kubeconfig entry generated for gke-tokyo.
admin@cloudshell:~/appdev-gke-dojo/advanced (glowing-palace-398423)$ gcloud container fleet memberships register ${CLUSTER_NAME} --gke-cluster ${REGION1}/${CLUSTER_NAME} --enable-workload-identity --project ${PROJECT_ID}^C
admin@cloudshell:~/appdev-gke-dojo/advanced (glowing-palace-398423)$ gcloud container fleet mesh enable
Waiting for Feature Service Mesh to be created...done.
admin@cloudshell:~/appdev-gke-dojo/advanced (glowing-palace-398423)$ gcloud container fleet memberships register ${CLUSTER_NAME} --gke-cluster ${REGION1}/${CLUSTER_NAME} --enable-workload-identity --project ${PROJECT_ID}
Waiting for membership to be created...done.
Finished registering to the Fleet.
admin@cloudshell:~/appdev-gke-dojo/advanced (glowing-palace-398423)$ gcloud container fleet memberships list --project ${PROJECT_ID}
NAME: gke-tokyo
EXTERNAL ID: 7274869e-70cf-4a38-b534-75a9929e1b32
LOCATION: asia-northeast1
admin@cloudshell:~/appdev-gke-dojo/advanced (glowing-palace-398423)$ gcloud container clusters update ${CLUSTER_NAME} --region ${REGION1} --update-labels mesh_id=proj-${PROJECT_NUM}
Updating gke-tokyo...done.
Updated [https://container.googleapis.com/v1/projects/glowing-palace-398423/zones/asia-northeast1/clusters/gke-tokyo].
To inspect the contents of your cluster, go to: https://console.cloud.google.com/kubernetes/workload/gcloud/asia-northeast1/gke-tokyo?project=glowing-palace-398423
admin@cloudshell:~/appdev-gke-dojo/advanced (glowing-palace-398423)$ gcloud container fleet mesh update --management automatic --memberships ${CLUSTER_NAME} --project ${PROJECT_ID}
Waiting for Feature Service Mesh to be updated...done.
admin@cloudshell:~/appdev-gke-dojo/advanced (glowing-palace-398423)$
```

ハンズオンを始める

事前準備: Lab-00.Lab 向けクラスタの準備

GKE クラスタのデプロイに時間がかかるため、事前にデプロイしておきます

チュートリアル [Step 5/9](#) の「[Lab-00.Lab 向けクラスタの準備](#)」まで実施してください

チュートリアルを再度起動する

1. チュートリアル リソースがあるディレクトリに移動する

```
cd /$HOME/gcp-getting-started-lab-jp/pfe-basic-sep
```

2. チュートリアルを開く

```
teachme tutorial.md
```

3. ステップ 4/9 の“**参考: Cloud Shell の接続が途切れてしまったときは ?**”まで進み
手順 3, 4 を実行する

プラットフォーム エンジニアリングとは

プラットフォーム エンジニアリングとは

プラットフォーム エンジニアリングとは、組織において**有用な抽象化**を行い、**セルフサービス インフラストラクチャを構築するアプローチ** です

散乱したツールをまとめ、**デベロッパーの生産性を高める効果** があります。プラットフォーム エンジニアリングの狙いは、**デベロッパーが体験する日常的な困難を解消して、行きすぎた責任共有モデルが引き起こす学習の手間を抑制** することです

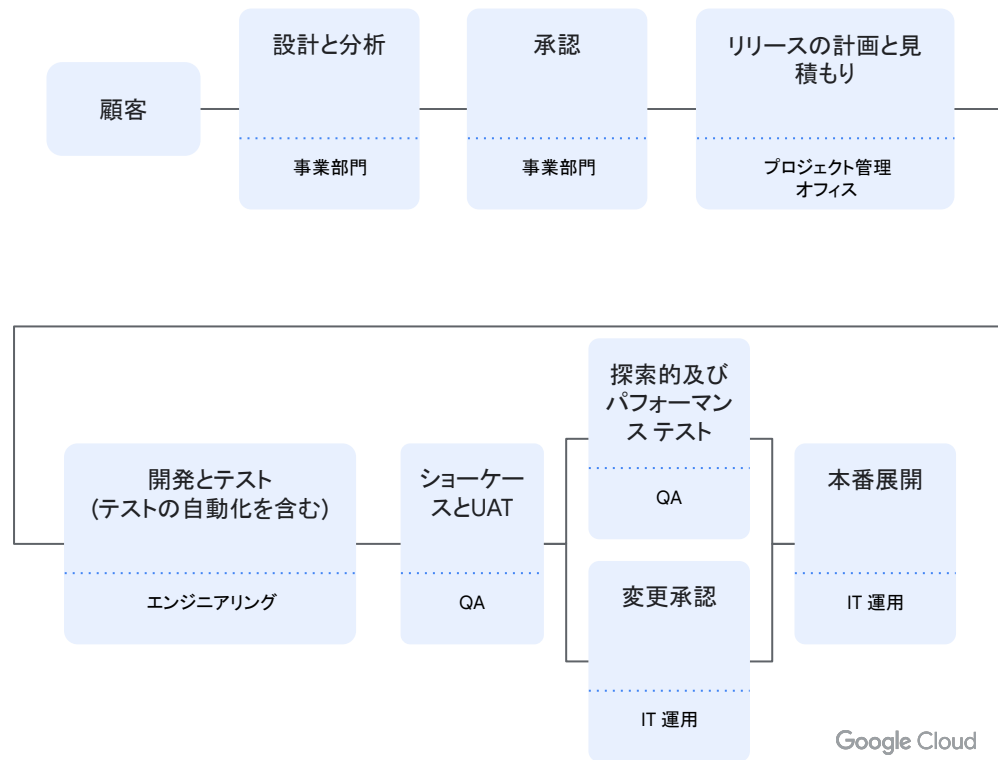
Google Cloud ブログ - 「道を照らす: プラットフォーム エンジニアリング、ゴールデンパス、セルフサービスのパワー」(2023 年 11 月 27 日) より

<https://cloud.google.com/blog/ja/products/application-development/golden-paths-for-engineering-execution-consistency>

開発者の範囲が設計から本番までの全体に

幅広い専門分野における熟練した技術
が求められるようになった

ソフトウェア開発
テストの自動化
データベース管理
サーバーとストレージのプロビジョニング
ネットワークの設計と実装
セキュリティの強化とパッチ適用
スケーリング
高可用性と災害復旧
...その他



Platform Engineering



開発チームに **舗装された道路** を提供できるようにチームを強化し、**プラットフォーム** をプロダクトとして提供することにより、ユーザーのニーズに合わせて **プラットフォーム** が進化し続けられるようにします

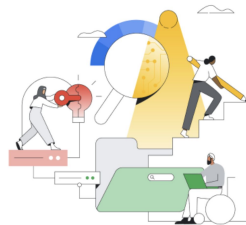
開発生産性を高めるためのプラットフォーム

プロビジョニングの自動化



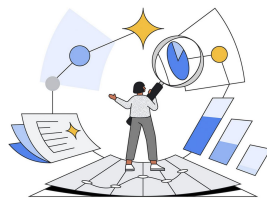
インフラを抽象化して自動化することで、プラットフォーム利用者の認知負荷を軽減

アプリケーション ライブラリ



新しいアプリケーションを素早くブートストラップするために、テンプレートのライブラリを用意

セルフサービス UX



新しいアプリや機能の導入は、ツール、ポータルUI、またはその両方を通じたセルフサービス

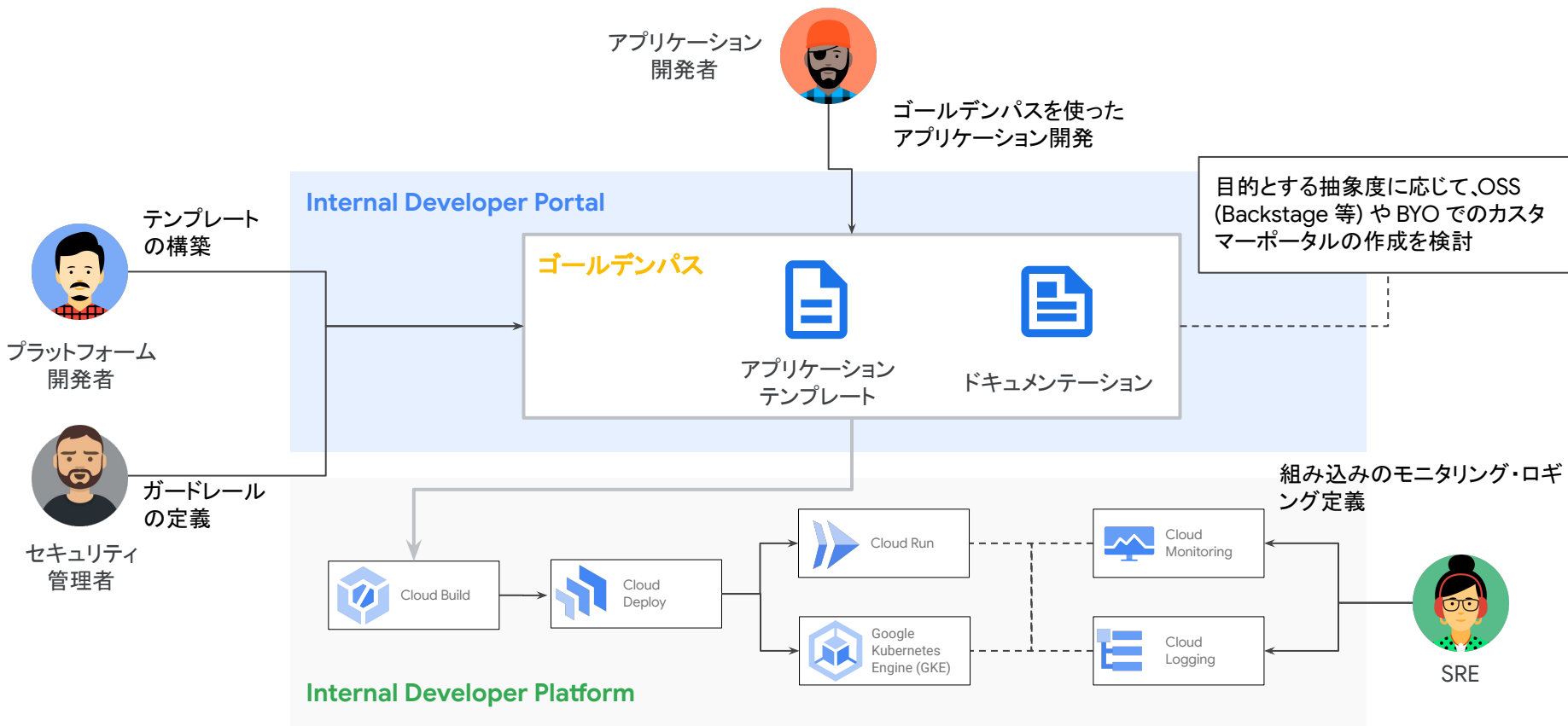
ガバナンスとガードレール



コスト管理、セキュリティ、ガバナンスがプラットフォームに標準組み込み

Google Cloud を活用した プラットフォーム エンジニアリング

Internal Developer Platform と Internal Developer Portal



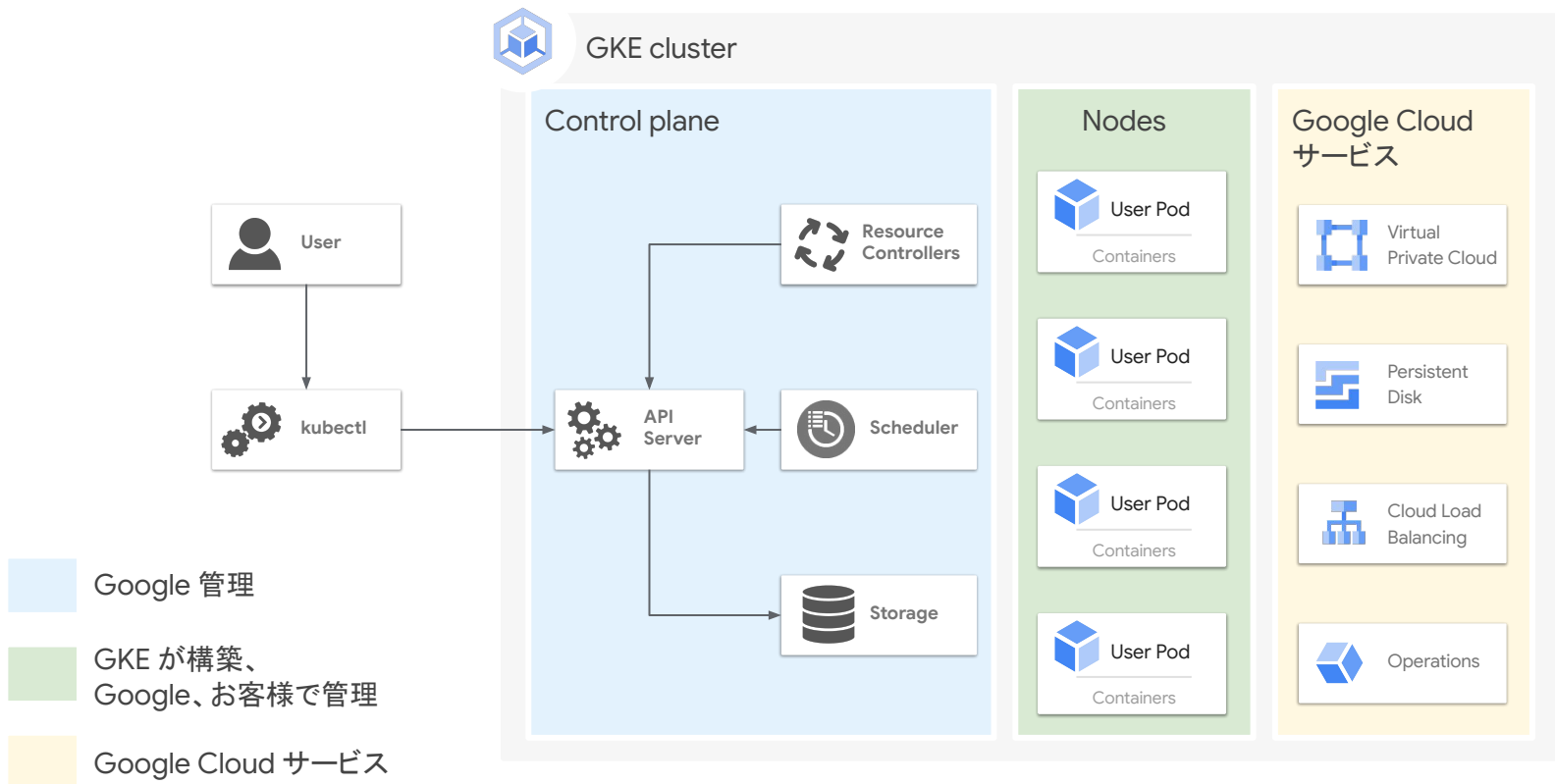
Google Kubernetes Engine (GKE)

Google が提供する**マネージド**な Kubernetes サービス

- **ワンクリック**で Kubernetes を構築
- **運用の手間を軽減**するオートヒーリング
オートアップグレード、リリースチャネル
- ログイング、モニタリングなど Google Cloud の
各種サービスとの**ネイティブな連携**
- **エンタープライズでの利用を前提**とした SLA の設定、
HIPAA、PCI-DSS といったセキュリティ基準への準拠



GKE のハイレベル アーキテクチャ



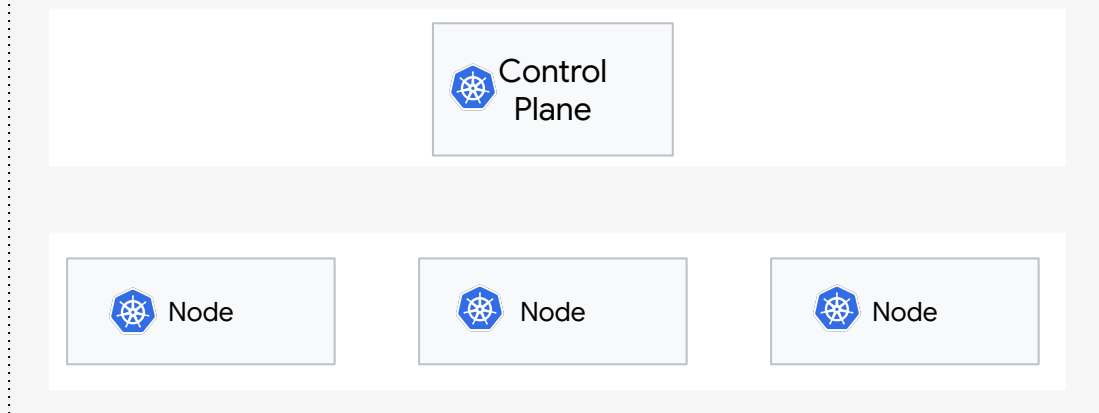
シングルゾーン クラスタ

- Control Plane と Node は同じゾーン
- **Control Plane の SLA: 99.5 %**
(Stable channel のバージョンが対象)
- アップグレード時に一時的に Control Plane の利用不可



シングルゾーンクラスタ

ゾーン A



```
$ gcloud container clusters create [CLUSTER_NAME] \  
  --zone compute-zone
```

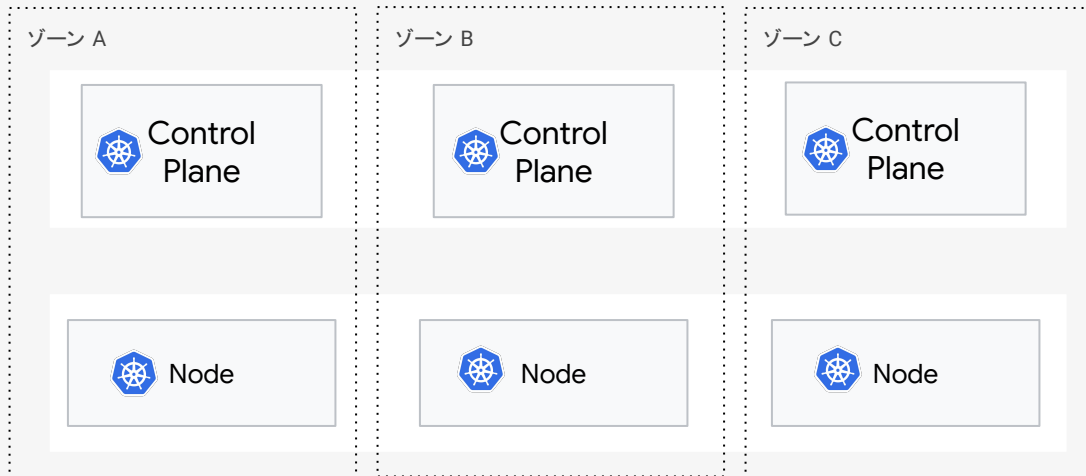

リージョン クラスタ

推奨



リージョンクラスタ

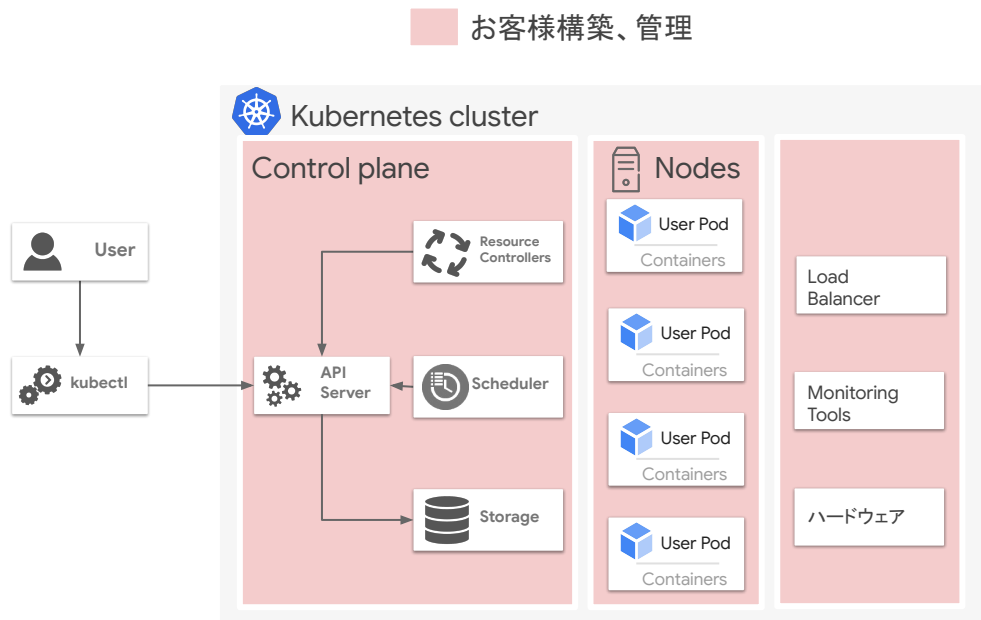
- Control Plane: 複数ゾーンで冗長化
- Node: 複数ゾーンで冗長化
- **Control Plane の SLA: 99.95 %**
(Stable channel のバージョンが対象)
- アップグレード時も
Control Plane が継続利用可能



```
$ gcloud container clusters create [CLUSTER_NAME] \  
  --region compute-region
```

自前運用の Kubernetes

- コントロールプレーンの構築 & 管理
- ワーカーノード構築 & 管理
- セキュリティ & ネットワーク各種設定
- パッチ管理 & アップグレード
- モニタリング
- スケーリングなど



GKE - スタンダード モード

- クラスターの自動アップグレード
- 4次元 自動スケーリング
- ノードの自動修復
- リージョナル クラスター

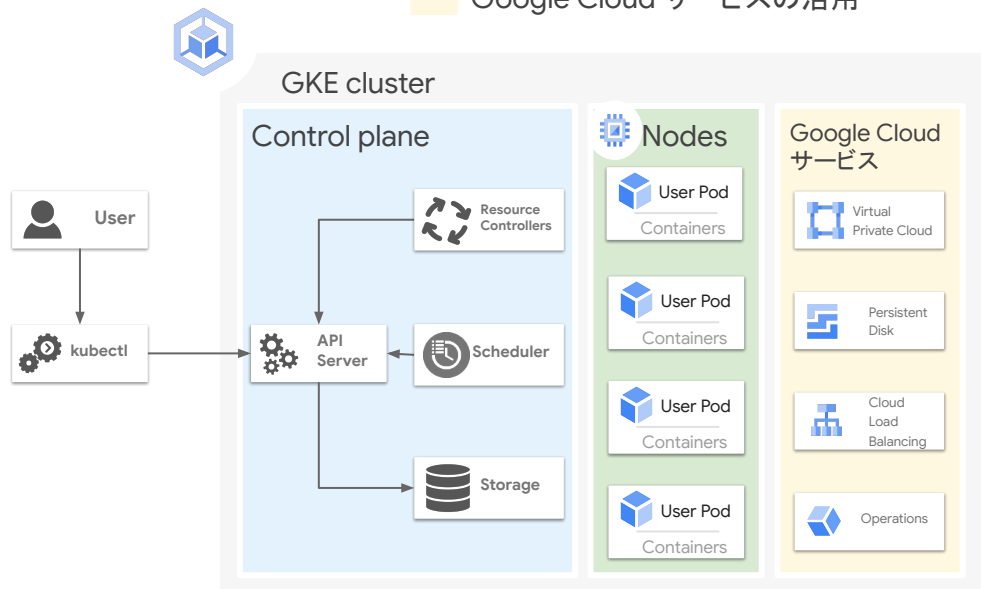
+ Multi Cluster Gateway

- 柔軟なメンテナンス 設定
- セキュリティ ベストプラクティス
 - 限定公開クラスター
 - ネットワーク ポリシー
 - Confidential GKE ノード
 - Binary Authorization
 - VPC-SC などなど

Google 管理

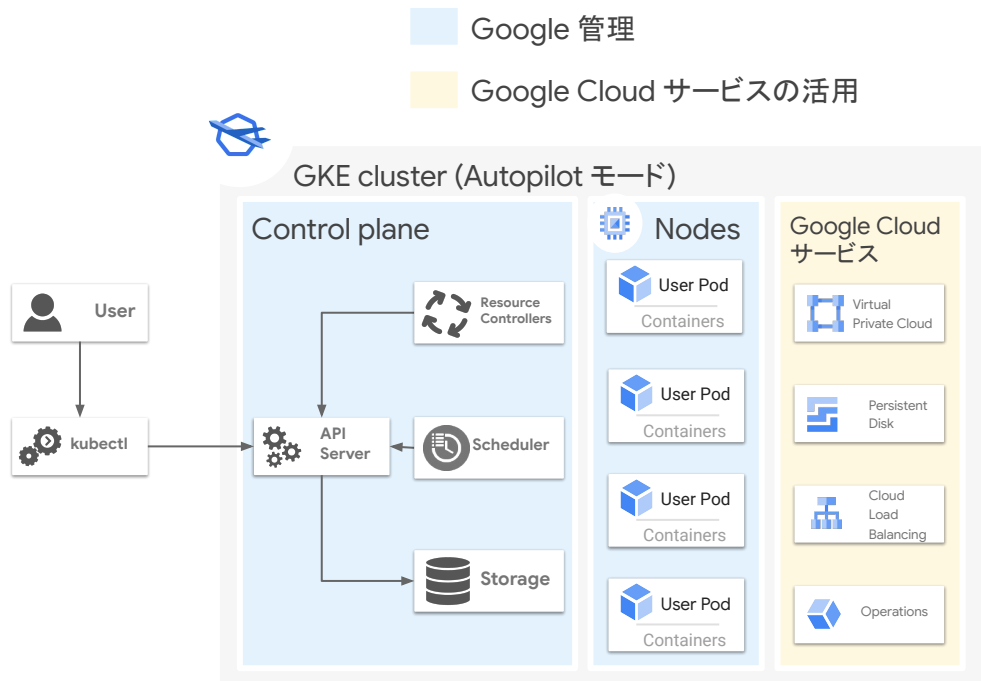
GKE が構築、Google、お客様で管理

Google Cloud サービスの活用



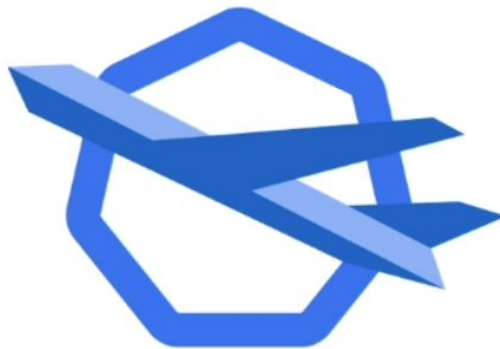
GKE - Autopilot モード

- コントロールプレーンだけでなくノードも Google Cloud が管理
- 本番ワークロードに適したベストプラクティスを初期適用
 - セキュリティ
 - ワークロード
- **ワークロード フォーカス**
Pod 単位での課金、Pod への SLA

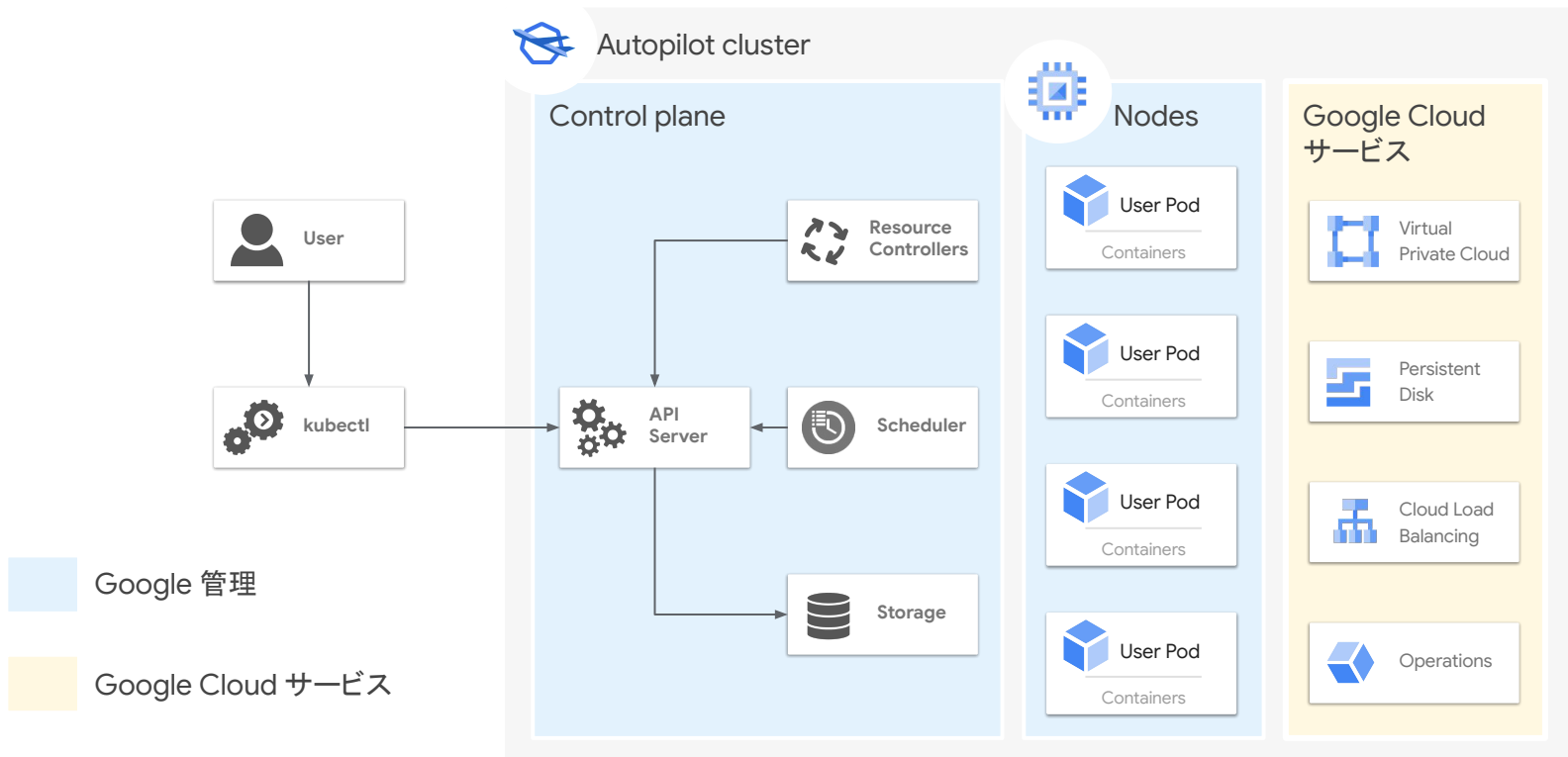


GKE - Autopilot モード

- Control plane に加え **Node も Google マネージド** に
- **本番ワークロードに適したベストプラクティス** が適用済み
 - セキュリティ
 - ワークロード
 - 各種設定
- Workload (Pod) ドリブンな世界へ
 - **Pod 単位での課金、Pod への SLA**



Autopilot のハイレベル アーキテクチャ

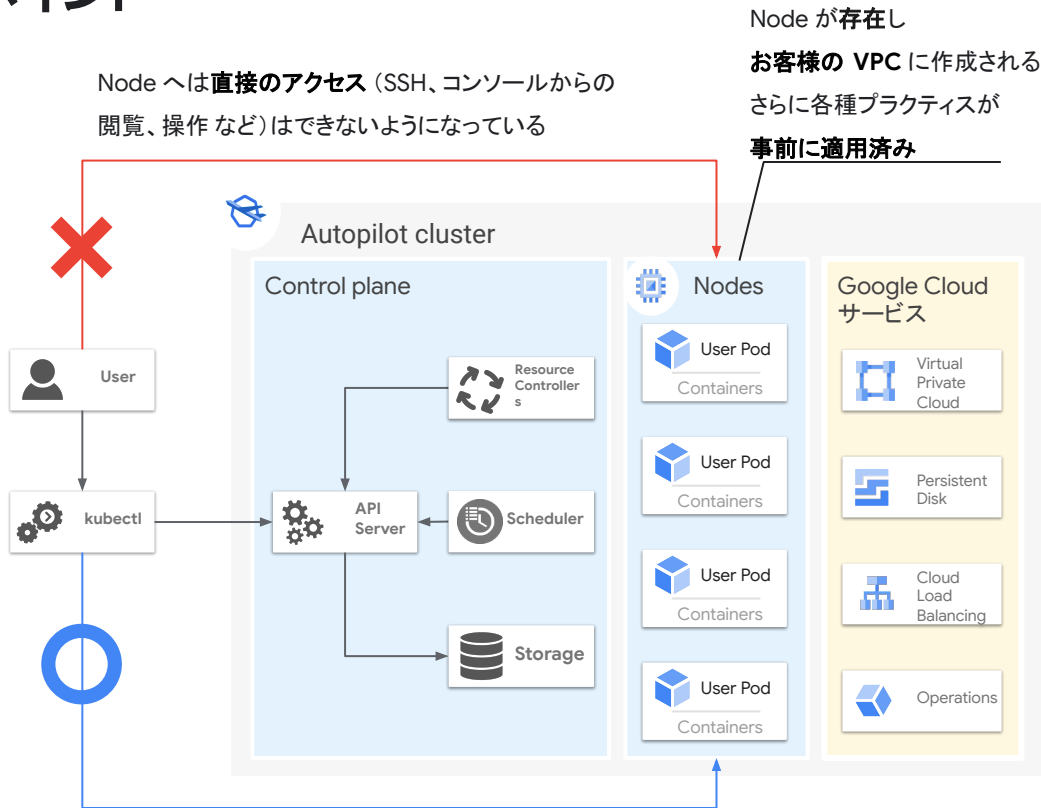


Autopilot アーキテクチャのポイント

Autopilot は GKE (K8s) の
良いところをそのまま残しつつ

- 運用負荷の低減
- ベストプラクティスの適用
- セキュリティの向上

を実現している



Node へは**直接のアクセス** (SSH、コンソールからの閲覧、操作など)はできないようになっている

Node が存在し
お客様の VPC に作成される
さらに各種プラクティスが
事前に適用済み

K8s 目線で見ると、閲覧、**必要な操作のみ**を許可している
(Pod affinity, Pod anti-affinity, DaemonSet の利用など)

Autopilot のもたらす価値

運用が
より簡単に

Node の管理を Google に任せることで
利用者は**ワークロードに集中** することが可能に

コストの
最適化

Node の課金から Pod の課金に変わること
で**利用リソースに即したコスト** へ最適化

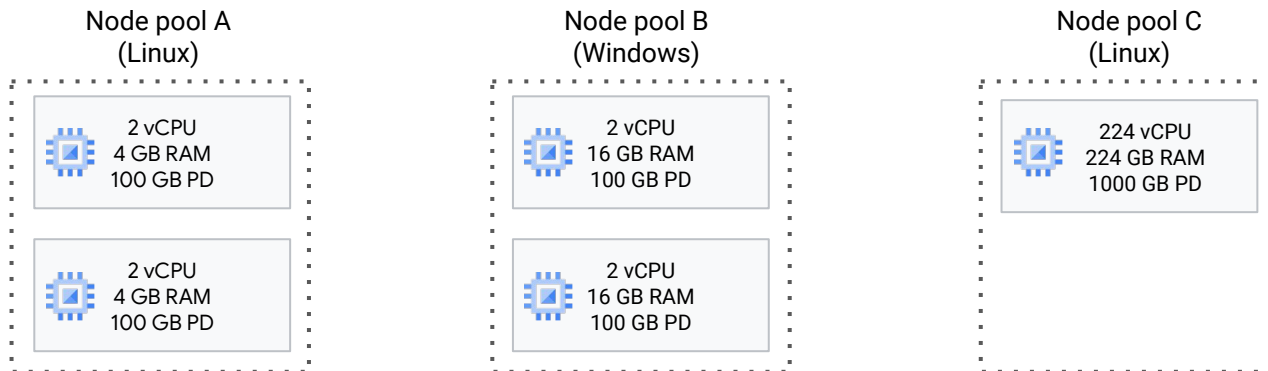
GKEは **すべてを自動化**

K8s をスケーラブルに使うために必要なものが全て**自動化**されている

- 自動修復
 - 問題があるノード を自動的に入れ替え
- 自動アップグレード
 - Kubernetes バージョンを自動的に更新
- 自動スケールアウト/スケールイン
 - 需要に応じて ノード 数を自動的に増減
- 自動プロビジョニング
 - リソースに最適な ノードプールを自動で作成・削除

Node Pool

- Node Pool は同じ設定の Node (Compute Engine インスタンス) で構成されるグループ
 - 同じマシンスペック、同じ OS、同じ GKE バージョン
- Node 数の増減や Node のアップグレードなどは Node Pool 単位で行う
- 各種自動化機能のサポート
 - **自動修復**、**自動スケール**、**自動アップグレード**



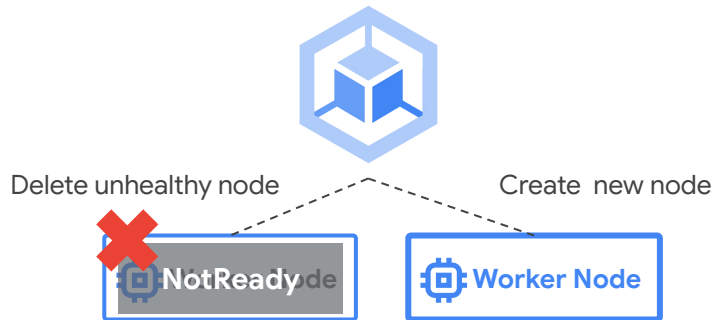
自動修復

Node が Unhealthy な状態になっていることを検知し、自動的に修復プロセスを開始する

Node が Unhealthy と判断される例:

- 約 10 分間、Node が NotReady ステータスを報告
- 約 10 分間、Node がステータスを何も報告しない
- 長期間(約 30 分間)、DiskPressure ステータスを報告、等

修復が必要な Node が検出されると、対象の Node を Drain 後に再作成される



自動アップグレード

GKE では Control Plane を Google が管理しており、自動的にパッチ適用・アップグレードされるが Node については自動もしくは手動でのアップグレードを選択可能（Autopilot は自動のみ）

1. リリースチャンネルを指定したクラスタ

- Control Plane: **自動**
- Node: **自動**
- GKE Autopilot はリリース チャンネルに登録される

2. 静的にバージョンを指定したクラスタ

- Control Plane: **自動**
- Node: **自動** or **手動**
 - Node のアップグレード タイミングを自分でコントロールしたい場合は **手動**

リリース チャンネル

バージョンングとアップグレードを行う際のベスト プラクティスを提供する仕組み

リリース チャンネルに新しいクラスタを登録すると、Google により

Control Plane と **Node** のバージョンとアップグレード サイクルが自動的に管理される

利用できる機能と更新頻度の異なる、以下 3つのチャンネルがある

- **Rapid** ... 最新のバージョンが利用可能。検証目的での利用を推奨 (SLA 対象外)
- **Regular** ... 機能の可用性とリリースの安定性のバランス
- **Stable** ... 新機能よりも安定性を優先する場合



メンテナンスの時間枠と除外

- **メンテナンスの時間枠**
自動メンテナンスを **許可する** 時間枠
 - 32 日周期で最低 48 時間必要
 - 各時間枠は 4 時間以上連続した時間
- **メンテナンスの除外**
自動メンテナンスを **禁止する** 時間枠
 - 詳細は後続資料で説明

使い方の例:

- 週末を避ける
- 大型セールやイベント期間中を避ける
- 一時的にアップグレードを延期する

メンテナンスの時間枠を有効化

週次エディタ
 カスタム エディタ

 32 日間のローリング ウィンドウ内で少なくとも 48 時間はメンテナンスが可能な状態にする必要があります。メンテナンスに 4 時間以上連続する時間を用意してください。

開始時刻 長さ

時刻は、ユーザーの地域のタイムゾーン (UTC+9) で表示されます

 曜日は常に UTC で指定します。メンテナンスの時間枠を水曜日の 02:00:00+06:00 (UTC+6) に開始する場合、これは火曜日の 20:00:00+00:00 (UTC) に相当します。開始時間には午前 2 時 (開始時間は現地時刻)、時間枠の日付には火曜日 (曜日は UTC) を選択します。 [詳細](#)

曜日

月曜日 火曜日 水曜日 木曜日

金曜日 土曜日 日曜日

メンテナンスの除外

Configure maintenance exclusions to specify when you don't want automated version upgrades of the control plane and nodes to occur. This can help prevent disruption to your workloads during specific times, such as during peak hours or outside of working hours. [Learn more](#)

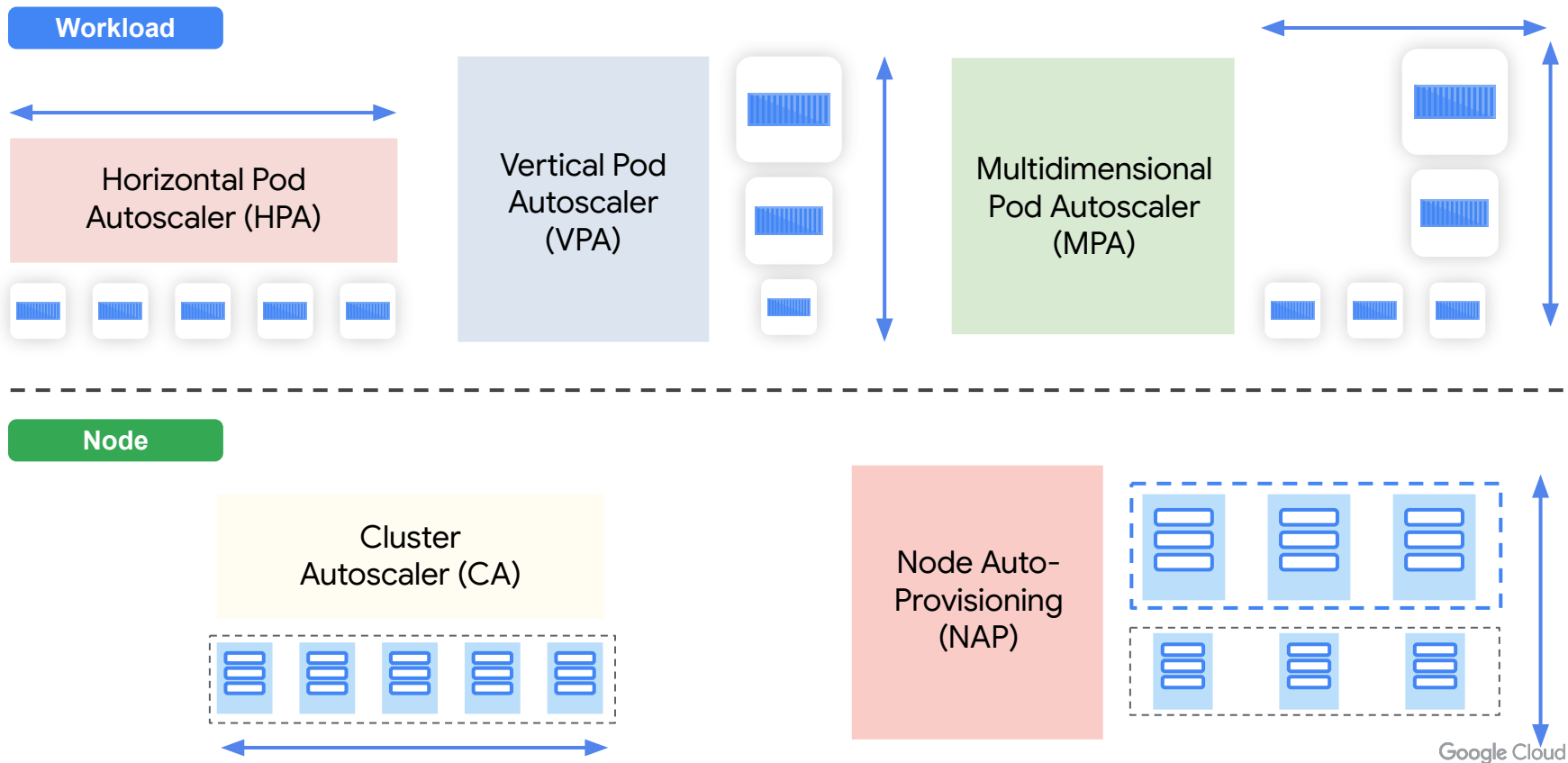
To specify times when routine, non-emergency maintenance won't happen, set maintenance exclusions on your cluster. Normally, routine Kubernetes Engine maintenance may run at any time on your cluster. [Learn more](#)

除外タイプ 1 ▼

開始時間 1* 🗓

終了時間 1* 🗓

自動スケール



HPA - Horizontal Pod Autoscaler

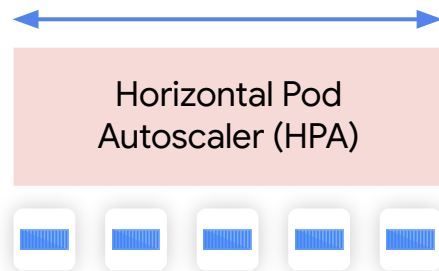
ワークロードの CPU やメモリの消費量等に応じて、自動的に Pod 数を増減させる機能

CPU / メモリ以外にも、カスタム指標や外部指標を使ったオートスケールもサポート

API version により、利用可能なメトリクスが異なる

- **autoscaling/v1**: CPU 利用率のみ
- **autoscaling/v2beta2**: カスタム メトリクスや外部メトリクスをサポート

複数のメトリクスを利用する場合、各メトリクスで算出されたレプリカのうち最大値を選択する



[ターゲットの決め方の例]

$$\text{Utilization target} = \frac{1 - \text{buffer}}{1 + \text{percent}}$$

10% CPU buffer

$$\frac{1 - 0.1}{1 + 0.3} = 0.69$$

Expect 30% traffic growth
in 2 or 3 minutes a Pod takes to start up
(consider node provisioning)

VPA - Vertical Pod Autoscaler

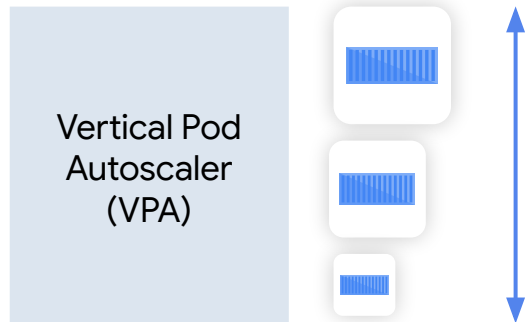
実行中のワークロードを分析し、CPU やメモリの requests / limits 値の推奨値算出やリソース値を自動更新する機能

VPA の更新モード (updateMode):

- **Off:** 推奨値を算出するのみ
- **Initial:** Pod の作成時に割り当て、既存 Pod の再起動無し
- **Auto:** 既存 Pod の再起動あり

急激なトラフィック増に対処する場合は、VPA ではなく HPA を利用することを推奨

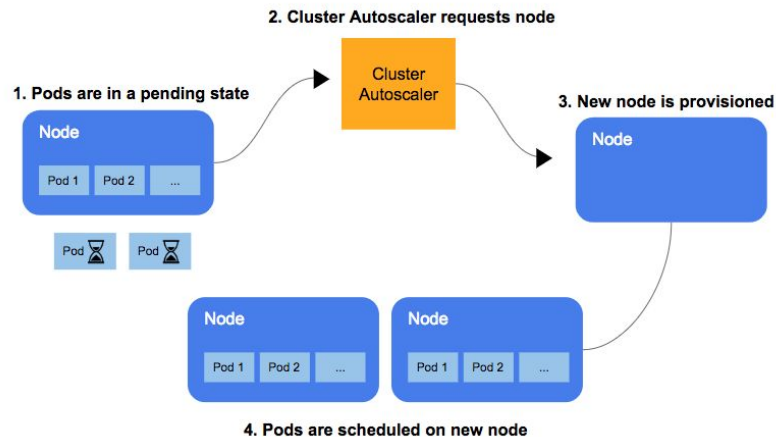
VPA を使う場合はまず **updateMode: Off** で推奨値の算出から始めてみる



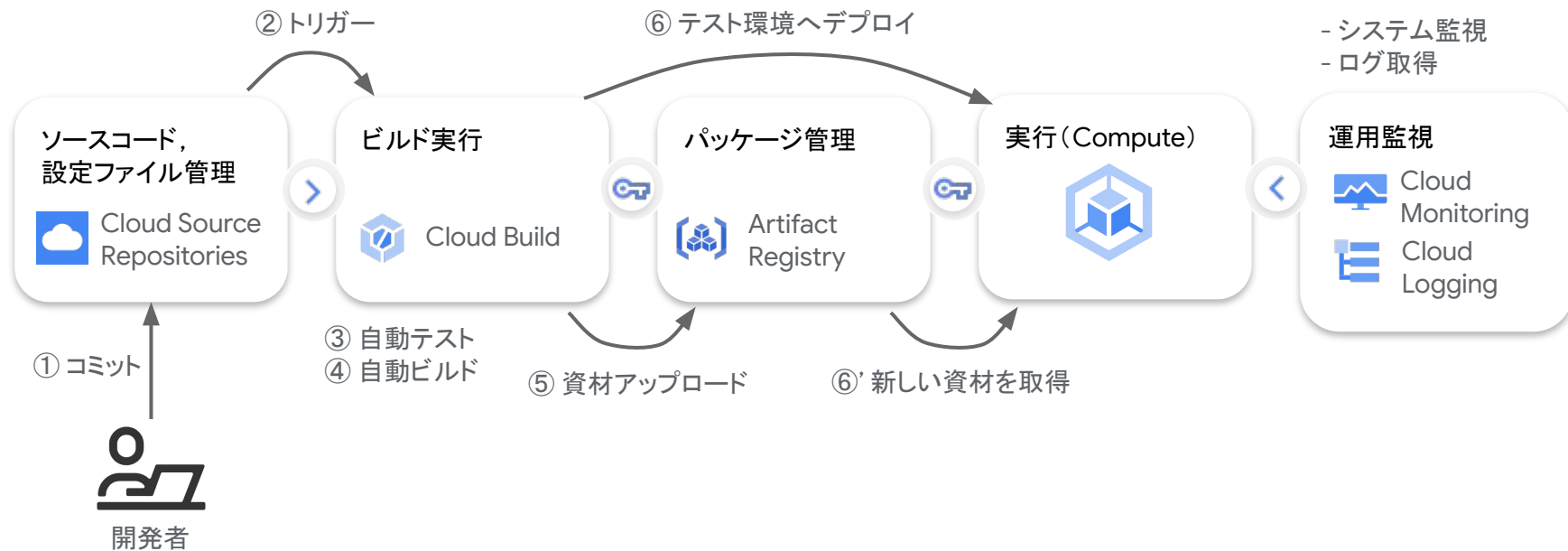
CA - Cluster Autoscaler

ワークロードの需要に応じて Node pool 内の Node 数を自動で増減させる機能

- Node pool 内の Node 数が不足し Pod のスケジューリングができない場合 Node を追加
- Node の使用率が低く、Node pool 内 Node 数を少なくしてもすべての Pod のスケジューリングが可能な場合 Node を削除



Google Cloud における CI / CD パイプラインの例



Artifact Registry

- Docker コンテナ イメージや Java, Node.js, Python などの各種パッケージを 1 か所で保管し管理できる保管場所
- Container Analysis を使用してコンテナの脆弱性をスキャンすることが可能



Cloud Build (ビルド & デプロイツール)



デベロッパー フレンドリー

CSR、GitHub または Bitbucket での変更をトリガーに

柔軟なビルドステップ

あらゆる CLI ツールをビルドステップとして
組み込むことが可能

フルマネージド CI プラットフォーム

お客様が VM を用意したりキャパシティの管理をする必要はない

Buildpacks

指定したディレクトリ以下を解析し、

Dockerfile なしに適切なコンテナへビルドします

サポート言語

- Go
- Node.js
- Python
- Java
- .NET Core
- Ruby
- PHP

```
$ ls
index.js  package.json

$ gcloud builds submit \
  --pack image=gcr.io/my-project/my-app
```

Cloud Deploy



継続的デリバリー (CD) に特化

CD のための各種機能をフルマネージドでご提供します

CI はこれまでのパイプラインで実施、本機能はデプロイのみを担当

成果物の厳密な管理

リリース コンテンツを事前にまとめ、
環境依存のない一貫性ある成果物管理

重要指標の可視化

CI / CD プロセスそのものの改善を促す指標を可視化

DORA 4 指標の“デプロイ頻度”と“デプロイ時失敗率”を組み込み

GKE Enterprise での Kubernetes マルチテナント管理

GKE Enterprise マルチテナントをよりシンプルに

マルチクラスタ管理

- フリートベースのマルチクラスタ管理
- セキュリティとガバナンス
- フリートとチーム全体のコストとパフォーマンスの可視化
- マルチクラスタネットワークング、ロードバランシング、サービスメッシング
- 自動化されたインフラストラクチャ管理のための GitOps

マルチテナントとチーム管理

- フリートベースのマルチチーム管理
- セルフサービス開発者環境
- Connect Gateway によるクラスターへのプライベートアクセス
- 各チーム向けのコストとパフォーマンスのダッシュボードと推奨事項

マネージドコンテナ プラットフォームでコスト削減

- 単一のプラットフォームでマルチクラウド、オンプレミス、およびエッジベースのワークロードを実行
- OSS ツールをマネージドサービス化 (Config Sync, OPA, Gateway Controllers, Istio)
- フリート全体を可視化する統合運用コンソール

Team Scope

チーム管理 API

アクセス、クォータ、セキュリティ ポリシーのサポート

チームのオンボーディングと可視性のための新しい UI

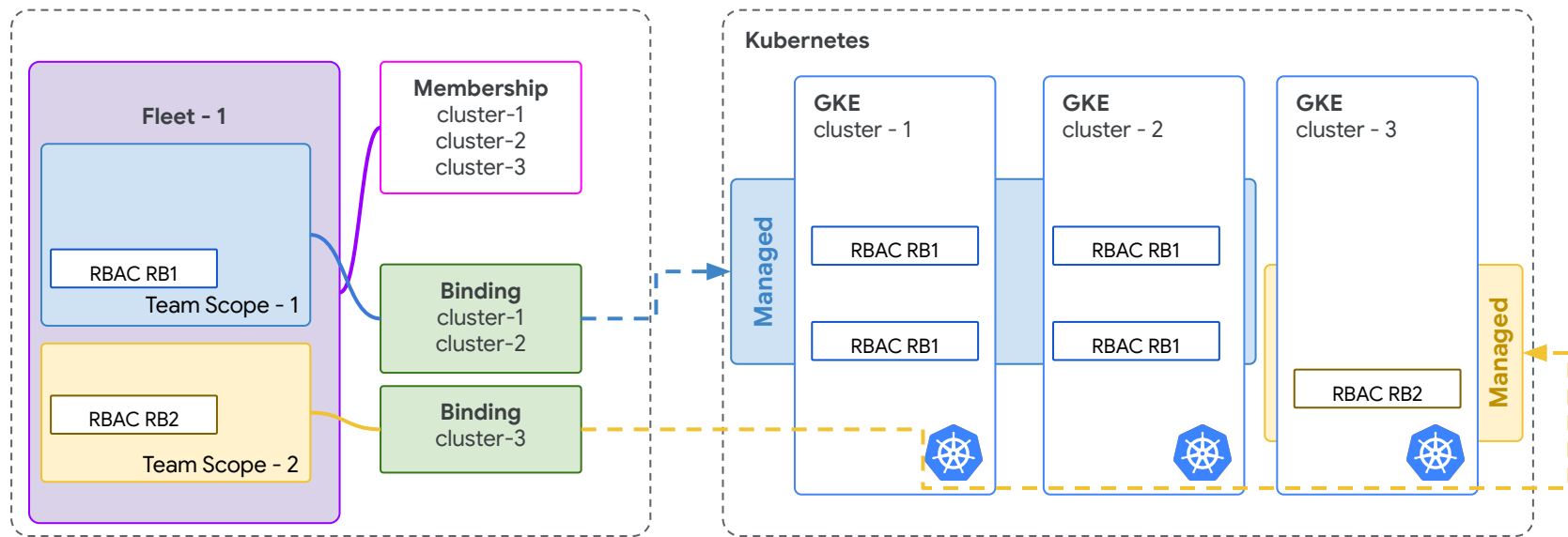
The screenshot shows the Google Cloud console interface for Fleet scopes. The page title is "Fleet scopes" with a "CREATE" button. The main heading is "Isolate, organize, and manage access for your team resources". Below this, there is a sub-heading "Provision and manage the infrastructure resources your teams need" and a section titled "Onboard your teams in minutes" with a brief description of the feature.

The screenshot shows the "Fleet scope details" page for a team named "bank-of-anthos-team-multicloud". The page is divided into several sections:

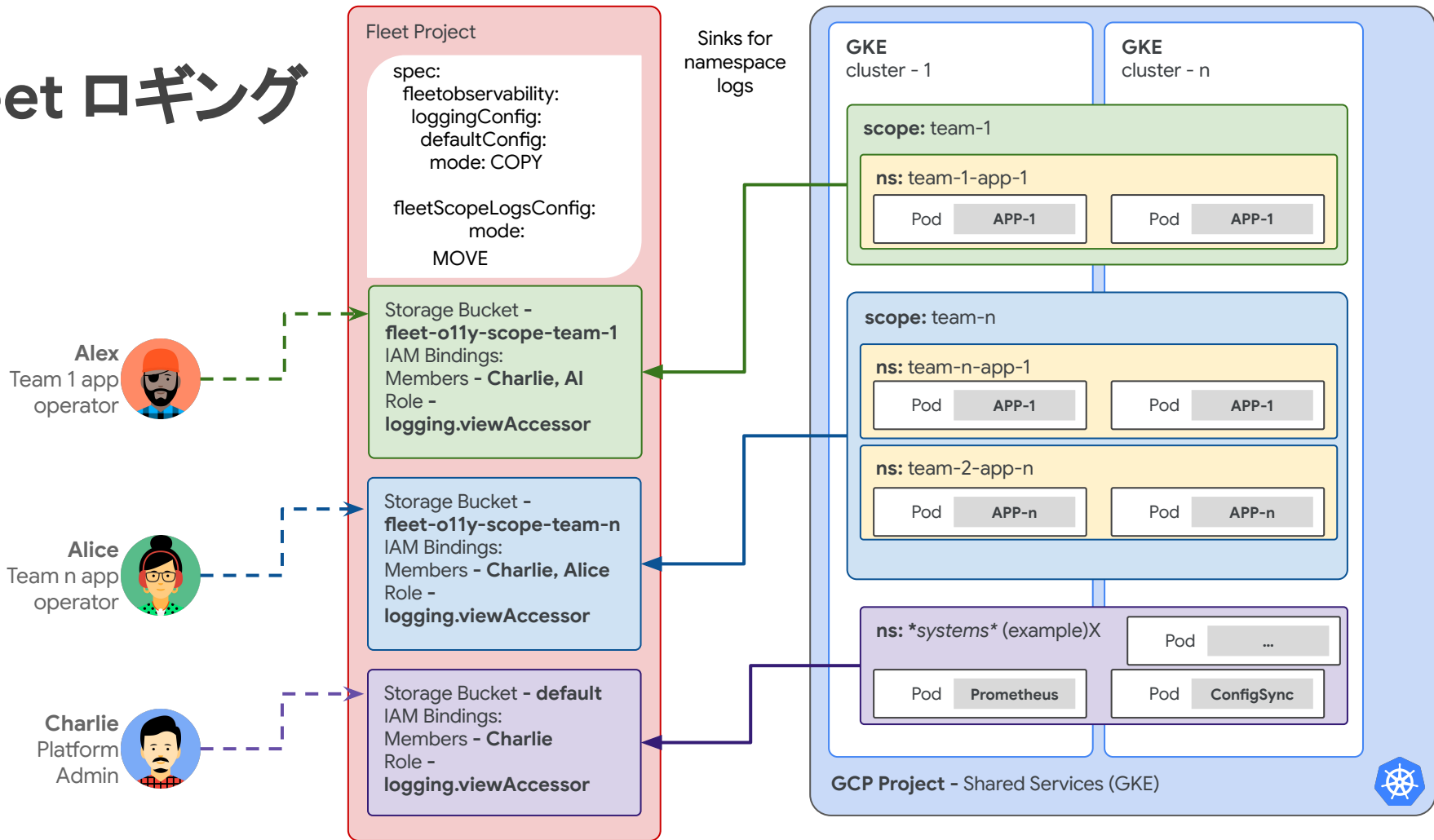
- Overview:** Shows 1 Cluster, 1 Namespace, 0 Errors, and 11.77K Restarts. Total Utilization is shown as CPU (4%), Memory (42%), and Disk (0%).
- CPU utilization by team:** A bar chart showing CPU usage over time, with a limit of 25CPUs and current usage of 0.67CPUs.
- Memory utilization by team:** A bar chart showing memory usage over time, with a limit of 14.5GB and current usage of 1.66GB.
- Disk utilization by team:** A bar chart showing disk usage over time, with a limit of 0.09GB and current usage of 0.09GB.

Team Scope

複数の GKE クラスターでマルチテナント管理



Fleet ロギング



Lab-01 GKE Enterprise によるマルチチームでの GKE の利用

チュートリアル [Step 6/9](#) の「[Lab-01 GKE Enterprise によるマルチチームでの GKE の利用](#)」まで実施してください

Cloud Workstations による オンボーディング速度の向上

Cloud Workstations

開発者のためのマネージドな開発環境



システム管理者

- 全体の開発環境を管理
- セキュリティ ポリシー
- セキュアな開発環境用イメージ

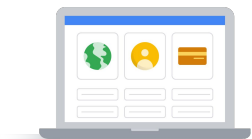


開発者

- オンデマンド
- どこからでもアクセス
- 導入、設定済みの開発ツール
- チーム間での一貫性

```
22 // Application will fail if environment variables are not set
23 if(!process.env.PORT) {
24   const errMsg = "PORT environment variable is not defined"
25   console.error(errMsg)
26   throw new Error(errMsg)
27 }
28
29
30 if(!process.env.GUESTBOOK_API_ADDR) {
31   const errMsg = "GUESTBOOK_API_ADDR environment variable is not
32   console.error(errMsg)
33   throw new Error(errMsg)
34 }
35
36 // Starts an http server on the $PORT environment variable
37 const PORT = process.env.PORT;
38 app.listen(PORT, () => {
39   console.log('App listening on port ${PORT}');
40   console.log('Press Ctrl+C to quit.');
```

開発チームに共通する IDE 関連の課題



開発環境のセットアップ

新/リモートメンバーの環境準備
開発メンバーが利用する
ハイスペックなマシンのコスト



セキュリティ対策/ 情報流出を防ぐための ガードレール

ローカルに保存されている
ソースコードの管理
開発者のローカル端末の
セキュリティ対策



開発者の生産性

プライベートな
ネットワーク環境で開発
ビルドに時間がかかる
アプリケーションが必要とする
資材の複雑化

ポイント

満たすべき最低限の機能



エンタープライズ対応

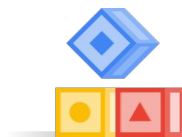
- フルマネージド
- カスタム仮想マシン
- VPC サポート
- コンプライアンスをカバー

その他の差別化ポイント



セキュリティ ガードレール

- 厳選されたアクセス ルール
- ロギング / 監査
- アップデート適用の強制
- 分離された開発環境



カスタマイズ可能

- カスタムイメージ
- 複数のエディタ対応 (IntelliJ, JupyterLab)
- ローカル / リモート IDE 連携
- サードパーティ DevOps ツールのサポート

数分で開発環境を用意

Chrome File Edit View History Bookmarks Profiles Tab Window Help

My Workstations - Cloud Worl x +

pantheon.corp.google.com/workstations/myworkstations?project=cloud-workstations-next

Google Cloud cloud-workstations-next Search Products, resources, docs (/)

Cloud Workstations My Workstations PREVIEW REFRESH

My resources ^

- My Workstations

Project resources ^

- Workstations
- Configurations
- Clusters

Cloud Workstations

Cloud Workstations provides managed, on-demand, development environments in the cloud. They can be accessed through a browser UI, a terminal/SSH, or from your local IDE through an SSH bridge. Get started by creating a workstation. [Learn more](#)

[CREATE WORKSTATION](#) [TAKE TUTORIAL](#)

Now viewing project "cloud-workstations-next" in organization "hekate.joonix.net" X

Performance issues detected! [Show debug panel](#)

一貫性のある開発環境

ワークステーション構成による共通設定

Google Cloud cloud-workstations-next

Create configuration PREVIEW

- Basic information
- Machine configuration
- Environment customization

The container image defines the initial state of a workstation, such as what binaries are pre-installed and what processes are running at startup time. We provide a set of managed images with standard toolchains and some popular code editors. You can customize this further by providing your own custom image.

Managed workstation images with preinstalled code editors

Custom container image

Container image URL SELECT

You can customize Cloud Workstations with container images hosted on any container registry. [Learn how to extend our base images.](#)

Service account

The service account that will be used on VM instances to support this config. If a custom container image is used, this service account must have permissions to pull the container image (or if this is not set, the image should be publicly accessible).

Storage settings

Files in the workstation home directory are stored in a persistent disk so that they persist between Workstation sessions. Storage settings cannot be modified after the configuration is created.

Disk type *

Disk size *

Advanced container options

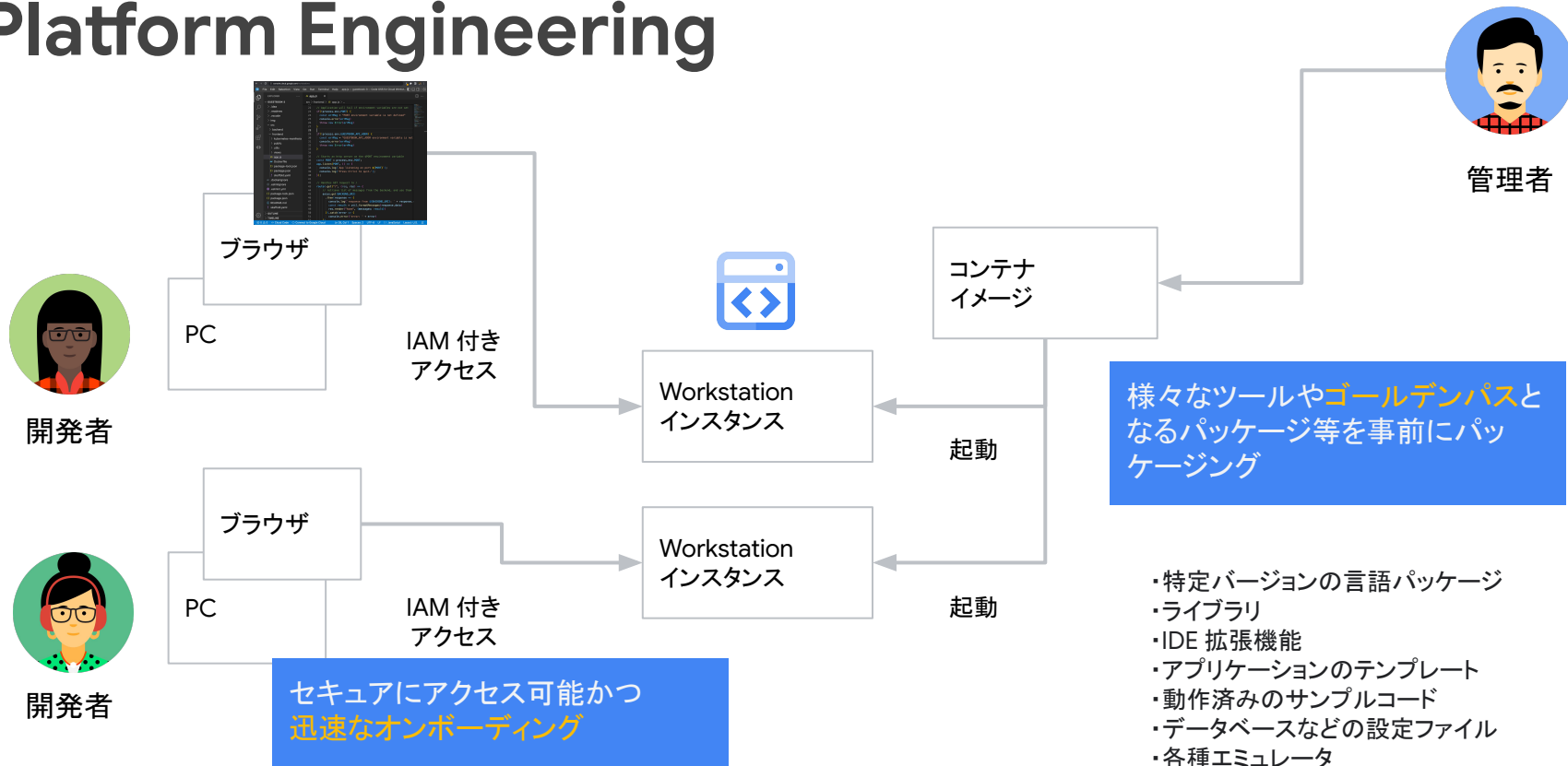
CREATE CANCEL

コンテナ設定による更なるカスタマイズ

```
Dockerfile X $ entrypoint.sh

java-env > Dockerfile
1 FROM us-central1-docker.pkg.dev/cloud-workstations-images/predefined/code-oss:latest
2
3 #Install OpenJDK 17 + tools
4 RUN sudo apt update
5 RUN sudo apt install gettext-base jq httpie -y
6 RUN sudo apt install openjdk-17-jdk -y
7
8 #Java extension pack
9 RUN wget https://open-vsx.org/api/vscjava/vscode-java-pack/0.25.0/file/vscjava.vscode-jav
10 unzip vscjava.vscode-java-pack-0.25.0.vsix "extension/*" &&\
11 mv extension /opt/code-oss/extensions/java-extension-pack
12
13 #Java debug
14 RUN wget https://open-vsx.org/api/vscjava/vscode-java-debug/0.43.0/file/vscjava.vscode-jav
15 unzip vscjava.vscode-java-debug-0.43.0.vsix "extension/*" &&\
16 mv extension /opt/code-oss/extensions/java-debug
17
18 COPY ./entrypoint.sh /
19 RUN chmod +x /entrypoint.sh
20 ENTRYPOINT ["/entrypoint.sh"]
21
22
```

Cloud Workstations を活用した Platform Engineering



プラットフォーム エンジニアリングを実践してみる

- ユーザー(開発者)中心のプラットフォーム開発
 - ユーザー中心設計の製品開発と同様にストーリー形式でバックログを管理する
 - Who / What / How と Acceptance Criteria を定義する
 - チケット・バックログ管理ツールを活用する

Lab02 に進む前に

- 背景

インタビューの結果、Java の開発者 (Taro) は Google Cloud や GKE 上での開発に慣れていないので、簡単にデプロイ可能なアプリケーションのテンプレートを必要とすることがわかりました。

- To Do

この内容をユーザーストーリーとして記述してバックログとして管理します。

- How to

チケット管理ツールや、テキストエディタなど皆さんの手元で記載して下さい。

ユーザーストーリーの記載例

ニーズ

Taro (開発者のペルソナ) は簡単にデプロイ可能なアプリケーションやマニフェストのサンプルが欲しい

目的

サンプルを利用して、高速に開発を開始するため

Acceptance Criteria

1. 簡単にアプリケーションのテンプレートが入手できること
2. 入手したアプリケーションを GKE の 開発環境 Cluster にデプロイしてみることが可能なこと

事前準備: Lab2. Cloud Workstations による開発環境と ゴールデンパスの提供

チュートリアル [Step 7/9](#) の「[Lab2. Cloud Workstations による開発環境とゴールデンパスの提供](#)」まで実施してください

Google Cloud でのオブザーバビリティ

Google Cloud Observability

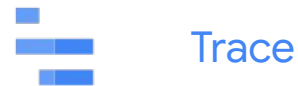
モニタリング



ロギング



アプリケーション パフォーマンス管理



Cloud Logging



- Google Cloud のすべてのログを集約する SaaS
- 収集されるログには、以下のようなものがある
 - Google Cloud の操作ログ
 - データアクセスのログ
 - サービス固有のログ
- fluentbit ベースの エージェント
 - GKE、GAE では、エージェントはすでに VM イメージに含まれている
 - 1つの手順で Compute Engine Linux VM 全体にインストールできる

Cloud Logging - コンポーネント

ログの発生元

監査ログ、GCP サービスのプラットフォーム ログ、システム ログ、アプリケーション ログなど、ログはカテゴリ別に分類されています



クラウド
サービス



OSS
コンポーネント



アプリ



ログ ルーター

すべてのログの発生元から一元管理の API にログを送信することで、任意の組み合わせの任意の宛先にログが確実に配信されます



Log Router



ログの任意の宛先

ログの一般的なアプリケーションには、コンプライアンス、アラート、アプリケーションのトラブルシューティング、およびビジネス インテリジェンスが含まれます



Cloud Logging
バケット



BigQuery



Pub/Sub

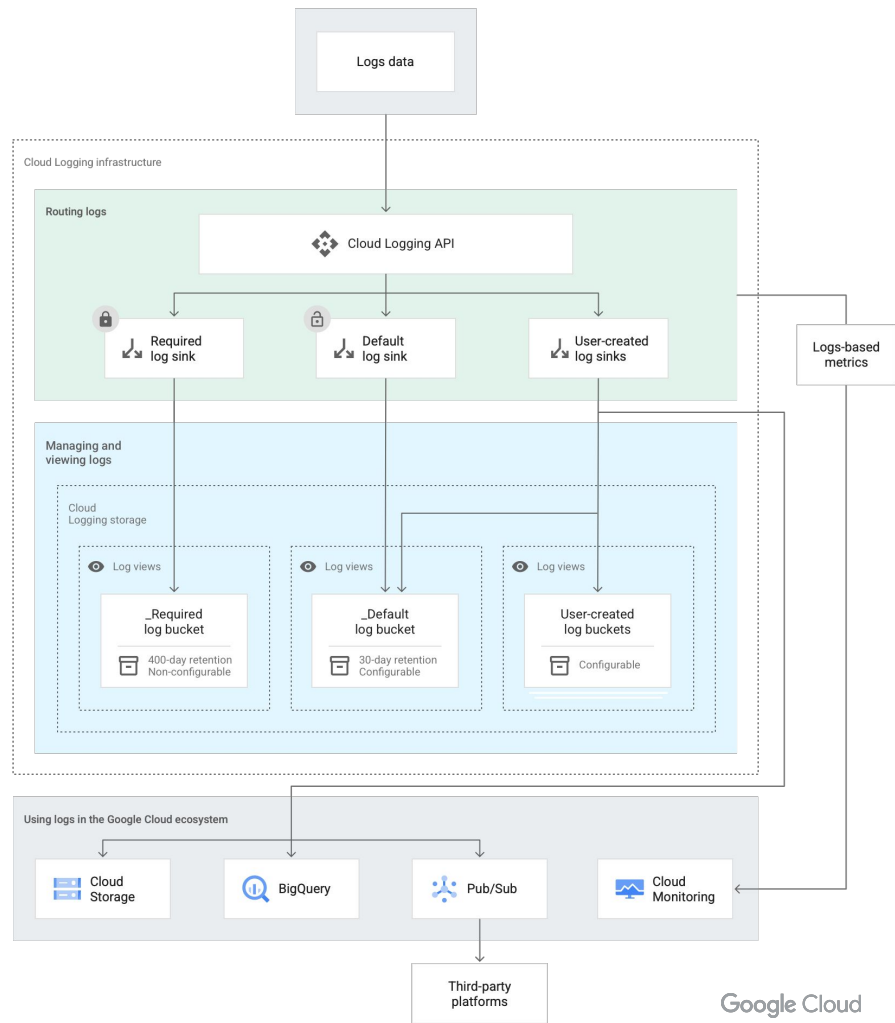


Stack



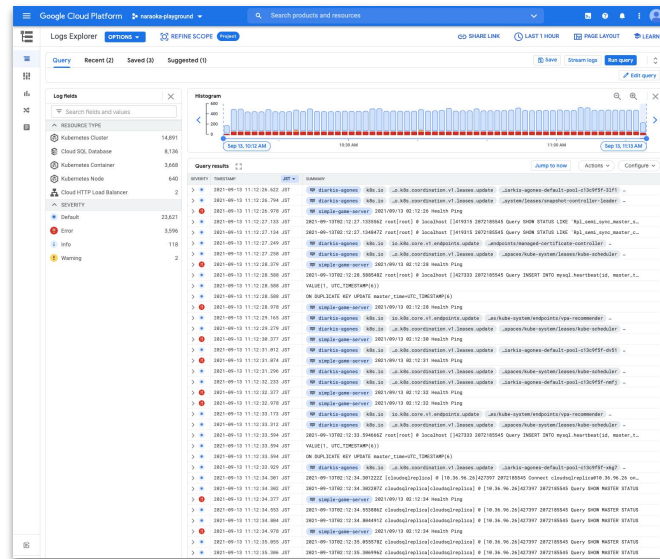
Cloud Logging - アーキテクチャ

- **_Required**
 - 管理アクティビティ ログ、システムイベント ログ、アクセス透過性ログで構成
 - 400 日間保持され、変更不可
 - 無料
- **_Default**
 - **_Required** バケットによって取り込まれないログ エントリで構成
 - 保存期間はデフォルトで 30 日間
 - 変更可能
 - 費用: 取り込み量、ストレージ
 - 無効化と変更が可能
- **User-created**
 - ユーザーの必要に応じてカスタマイズされたログ バケット



GKE のロギング

- GKE は**デフォルト**で以下のログを **Cloud Logging** に収集
 - 各コンテナの標準出力及び標準エラー出力
 - システムログ
(kubelet、docker / containerd など)
 - クラスタのイベントログ (Audit ログ含む)
- Cloud Logging に収集するログをシステムログのみにすることも可能
- 収集したログから Log-based メトリクスの作成
- 収集したログを Cloud Storage, BigQuery, Cloud Pub/Sub へエクスポート

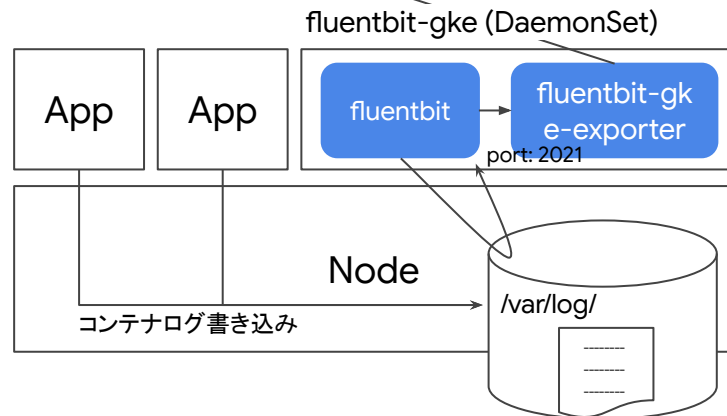
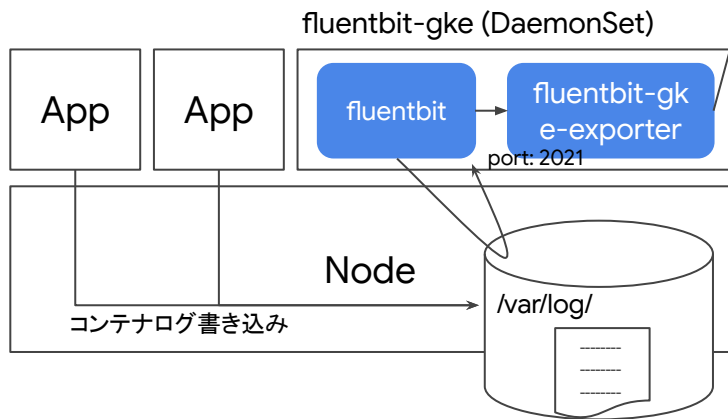


GKE のログ転送の仕組み (デフォルト構成)

コンテナのログは実行 Node の
`/var/log/containers` 配下に格納される

GKE ログエージェント ([fluent-bit](#) ベース) は
hostPath としてマウントした Node 内のログ領
域から収集したコンテナログを
fluentbit-gke-exporter 経由で Cloud Logging
に転送する

Cloud Logging



Fleet Logging

GKE のチーム管理機能の 1 つ

Namespace / チームスコープ単位の
ログバケットの作成と転送設定
を自動的に構成する

Alex
Team 1 app
運用者



Alice
Team n app
運用者



Charlie
Platform
管理者



Fleet Project

```
spec:  
  fleetobservability:  
    loggingConfig:  
      defaultConfig:  
        mode: COPY
```

```
  fleetScopeLogsConfig:  
    mode:  
      MOVE
```

Storage Bucket -
fleet-o11y-scope-team-1
IAM Bindings:
Members - **Charlie, AI**
Role -
logging.viewAccessor

Storage Bucket -
fleet-o11y-scope-team-n
IAM Bindings:
Members - **Charlie, Alice**
Role -
logging.viewAccessor

Storage Bucket - **default**
IAM Bindings:
Members - **Charlie**
Role -
logging.viewAccessor

Namespace
単位での
ログシンク

GKE
cluster - 1

GKE
cluster - n

scope: team-1

ns: team-1-app-1

Pod

APP-1

Pod

APP-1

scope: team-n

ns: team-n-app-1

Pod

APP-1

Pod

APP-1

ns: team-2-app-n

Pod

APP-n

Pod

APP-n

ns: *systems* (example)X

Pod

Prometheus

Pod

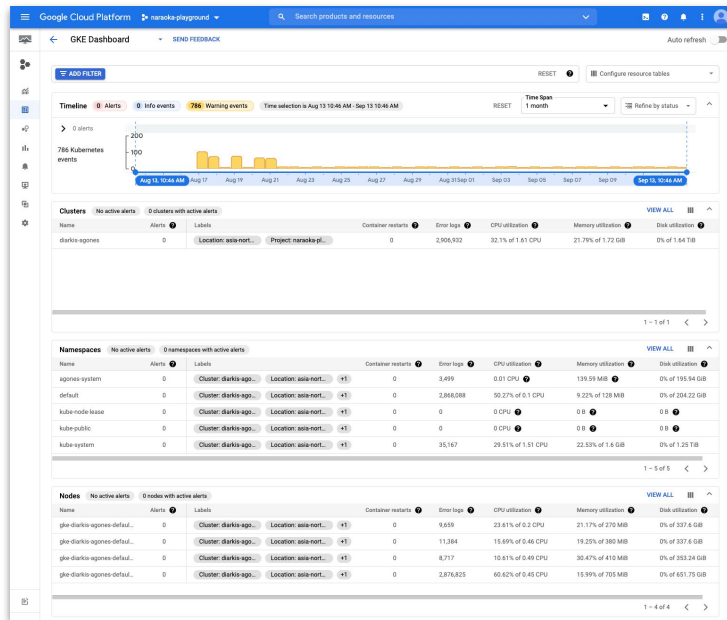
ConfigSync

Google Cloud Project - Shared Services (GKE)



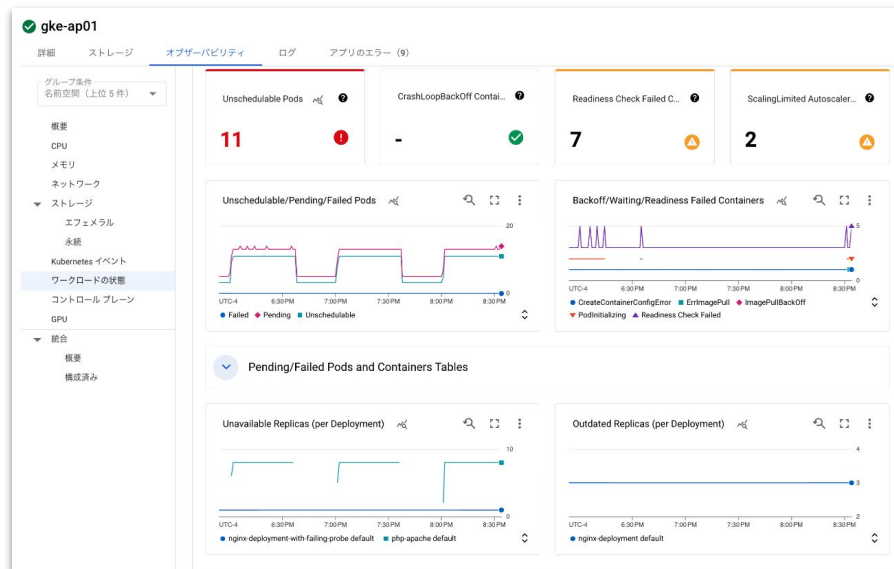
Cloud Monitoring

- GKE はデフォルトで Node / Pod のメトリクスを収集
- 収集したメトリクスは Cloud Monitoring へ送られる
 - CPU Usage
 - Memory Usage
 - Storage Usage
 - Network Usage
 - リスタートの数 (Pod, Container)
- アプリケーションのメトリクスを Managed Service for Prometheus (GMP) へ連携することも可能

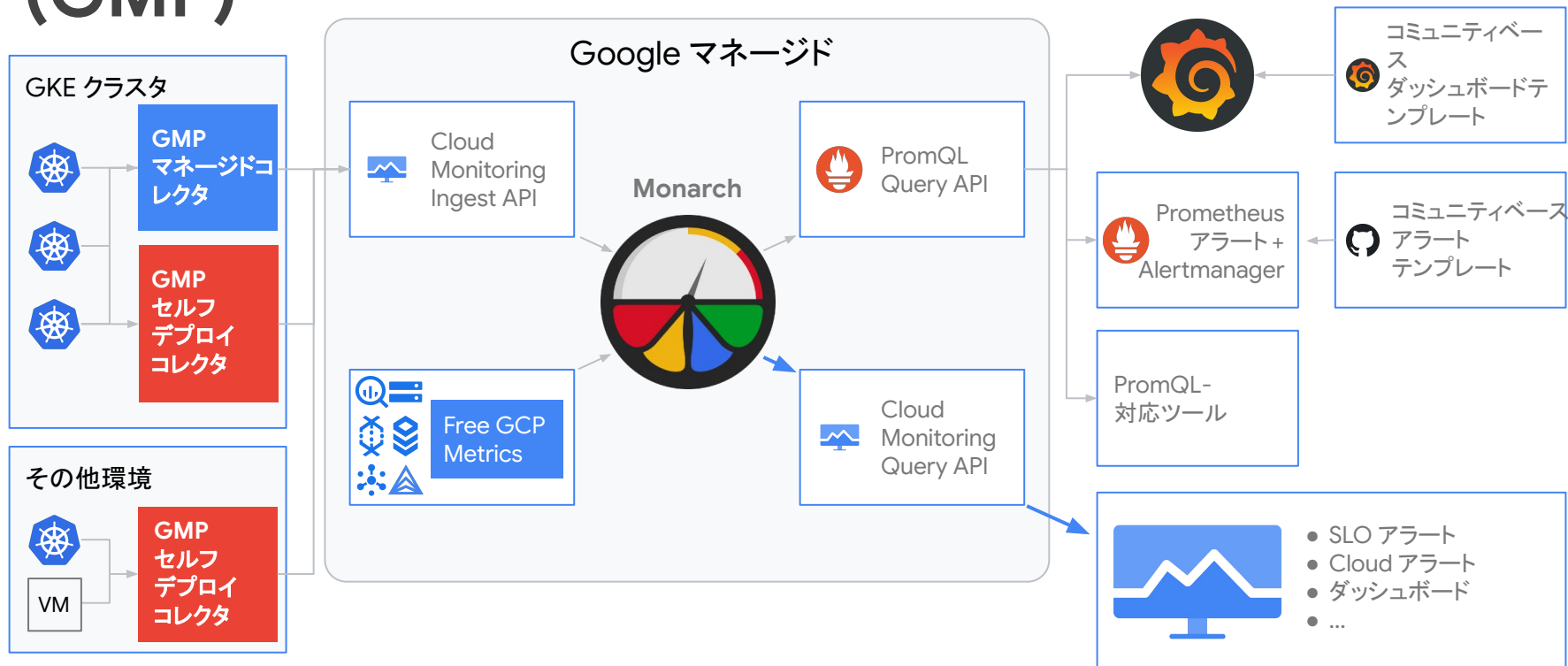


ワークロードステータス メトリクス

- Kube State Metrics の自動収集をサポート
- 以下のような不健全な状態のワークロードを可視化することができる
 - Node にスケジュールできない
 - CrashLoopBackOff 状態になっている
 - Readiness Check に失敗した
 - HPA のスケール上限に達している
- メトリクスの収集には Managed Service for Prometheus (GMP) の有効化が必要



Google Managed Service for Prometheus (GMP)



Managed Service for Prometheus (GMP) の特長

運用負荷の低減

フルマネージドなデータストア / コレクタを活用することで
運用負荷を低減

貴重なエンジニアリング リソースをコア業務に集中

複数環境の統合監視

複数クラスタ / プロジェクトを
一元的にモニタリング

オンプレミス・他社クラウド上のワークロードからもメトリクスを収集可能

既存監視ツールの有効活用

Grafana / Alertmanager 等
既存の Prometheus エコシステムを活用可能

既存の運用は変えずにストレージのみマネージドサービス側に移行

データ収集: コレクタ

- Prometheus のバイナリを置き換えてデータを収集
 - GMP にデータを送るための修正が入っている
- オープンソース
 - <https://github.com/GoogleCloudPlatform/prometheus-engine>
- 導入パターンは マネージド (GKE, GCE) or セルフ デプロイ
- データをローカルに保持しない
 - HA 設定がいないケースも
- 素の Prometheus と**並行稼働**できます



データ収集：コレクタ 導入パターンの違い

マネージド (推奨)

- GMP operator (GKE) / Ops Agent (GCE) を使い導入
- 設定、スケールなど Operator が受け持つ
- prometheus-operator から簡単に移行可能 (GKE)
- 未サポートのユースケースあり
 - [カーディナリティを減らすためのローカル集計](#) など

セルフ デプロイ

- Prometheus の [バイナリを置き換えて](#) 導入
- 今までの導入、管理方法 (prometheus-operator など) をそのまま利用可能
- スケール、シャーディングなどは自分で対応
- マネージド未対応パターンも対応可能

クエリ

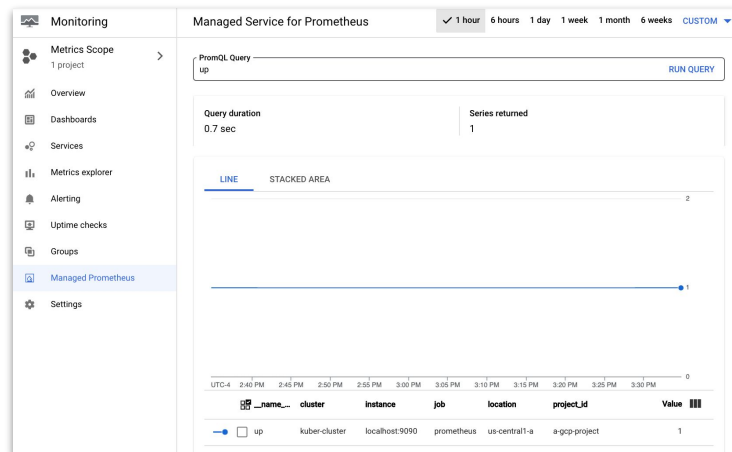
- PromQL をサポート
- 最大 1,000 の Google Cloud プロジェクトを
グローバルな 1 データソースとして 監視できます
- クエリ実行時にプロジェクトのグループに
読み取り権限を設定できます
(指標スコープを利用)

<https://cloud.google.com/monitoring/settings?hl=ja#>

concept-scope

Prometheus サービスからデータをクエリする | オペレーションスイート | Google Cloud

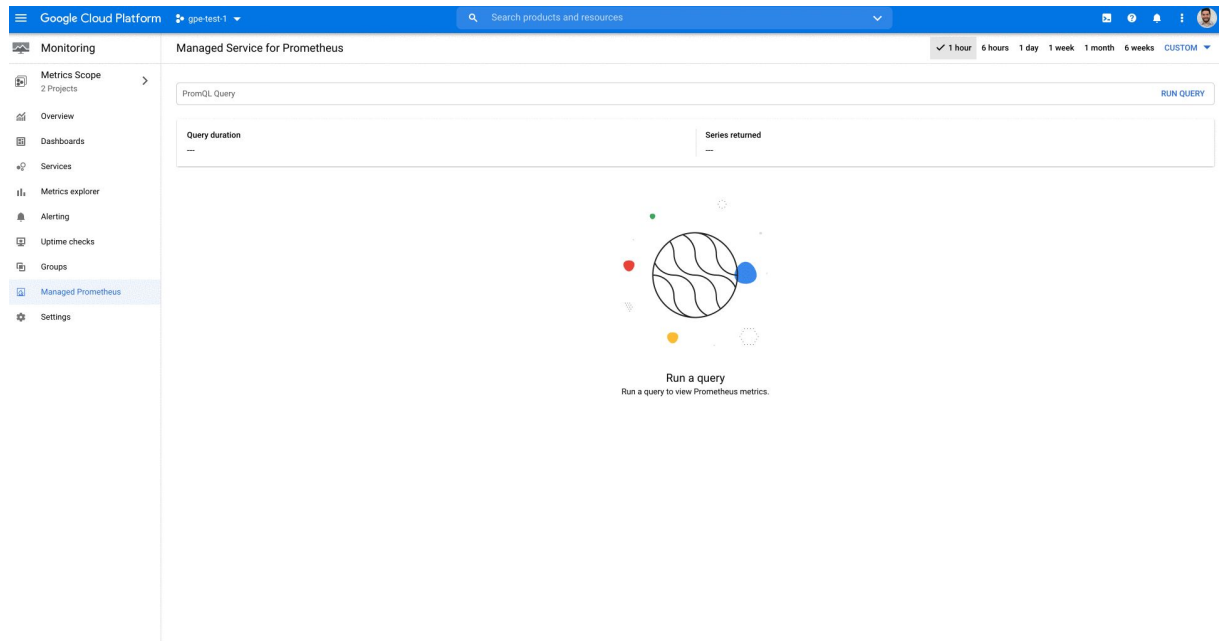
<https://cloud.google.com/stackdriver/docs/managed-prometheus/query?hl=ja>



Cloud Monitoring UI

Cloud Monitoring UI から
PromQL クエリを発行することが
可能

ダッシュボードやアラート等のコン
ポーネント管理が不要となり、
運用負荷を低減



Prometheus サービスからデータをクエリする | オペレーションスイート | Google Cloud

<https://cloud.google.com/stackdriver/docs/managed-prometheus/query?hl=ja>

既存 Prometheus 関連資産の活用

Grafana や Alertmanager など既存の Prometheus 関連資産を活用することも可能

Grafana で GMP をデータソースとして利用するために
データソース同期ツールをデプロイする



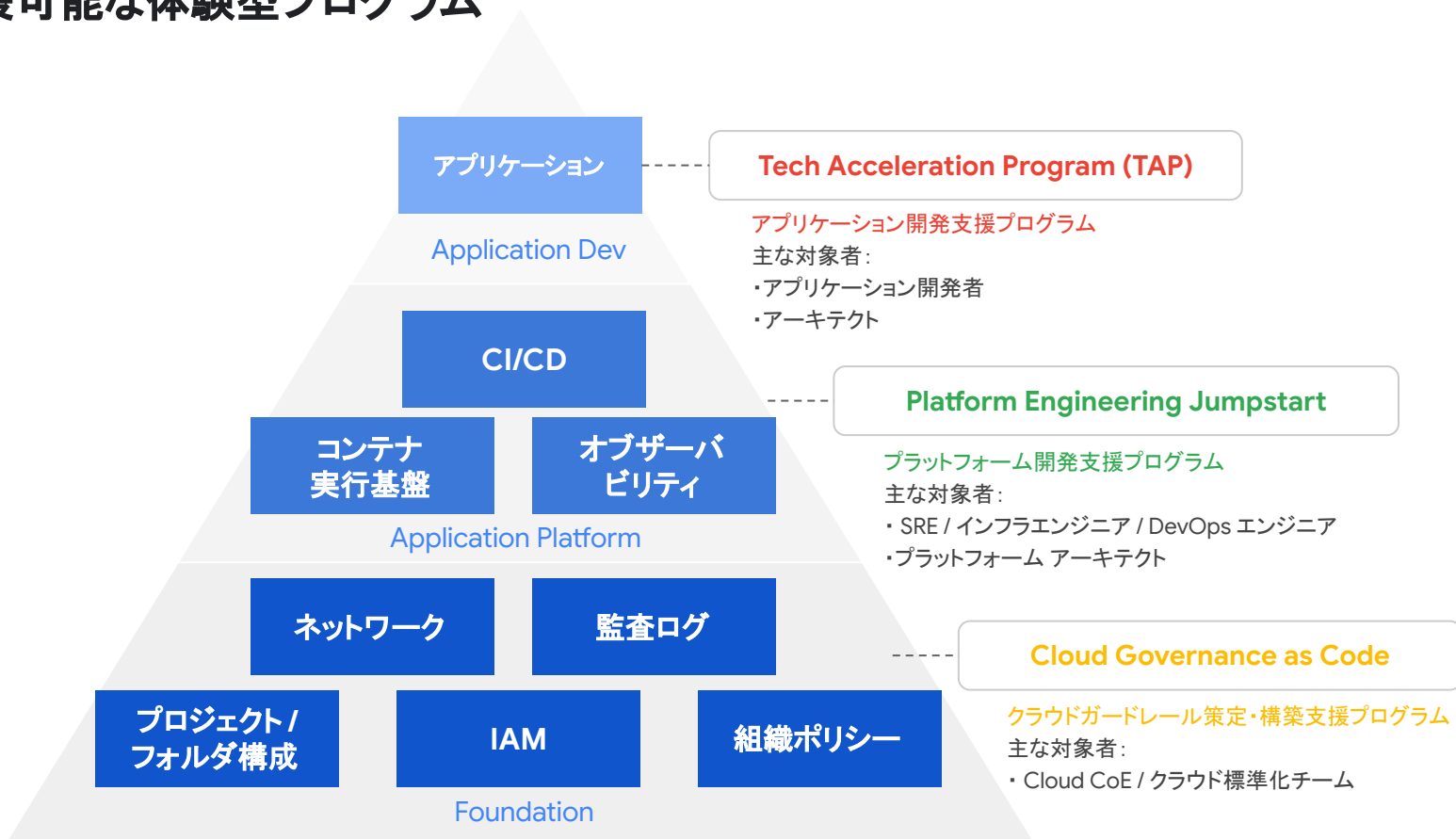
事前準備: Lab-03. Platform 管理者のためのオブザーバビリティ

チュートリアル [Step 8/9](#) の「[Lab-03. Platform 管理者のためのオブザーバビリティ](#)」まで

実施してください

体験型プログラムのご紹介

ご支援可能な体験型プログラム



Platform Engineering Jumpstart

Google Kubernetes Engine (GKE) をベースにした 開発者向けプラットフォームのプロトタイプ構築を Google Cloud のエンジニアがサポートします。

開発者向けプラットフォームを構築することによって、Developer Experience (DevEx) と開発生産性の向上が見込まれます。

本ワークショップを通じて、Kubernetes の基礎から実践的なプラットフォームの設計・構築方法について知見を得ることができます。

利用予定のプロダクト: GKE, Anthos Service Mesh, Cloud Build, Cloud Deploy, Cloud Operations 等



Typically 2-3 days

対象者、アウトプット、アジェンダ

対象者

- SRE, インフラ エンジニア, DevOps エンジニア
- プラットフォーム アーキテクト
- アプリケーション開発者

アウトプット想定

- GKE をベースとした開発者向けプラットフォームのアーキテクチャ図
- 上記を実現するシステム (プロトタイプ)

アジェンダ

- プラットフォームの目指すべき姿やスコープの確認
- Kubernetes / GKE 概要説明
- プラットフォームのアーキテクチャ議論
- プラットフォームの構築・実装 (プロトタイプ)
- 継続的な支援についての議論

実施スケジュールの例

Day 0

Planning

本ワークショップで構築するプラットフォームの目指すべき姿やスコープの確認

必要に応じて Google Cloud や Kubernetes に関する概要説明・勉強会を事前に実施

Day 1

Knowledge Transfer

プロトタイピングに向けて必要となるナレッジのトランスファー

Customer Engineer による GKE や CI/CD, Operations プロダクトの各機能の概要説明

Day 2

Design & Architecting

開発者向けプラットフォームのベースとなる GKE クラスターや CI/CD, Operations の設計・アーキテクティングを実施

プロトタイプの開発に着手できる状態を作る

Day 3

Develop Prototype

ここまでデザインしたプラットフォームを Customer Engineer が支援をしながら実装

必要となる情報、課題などをその場で解決。PoC の検証可能なプロトタイプを作り上げる

Thank you

Google Cloud

