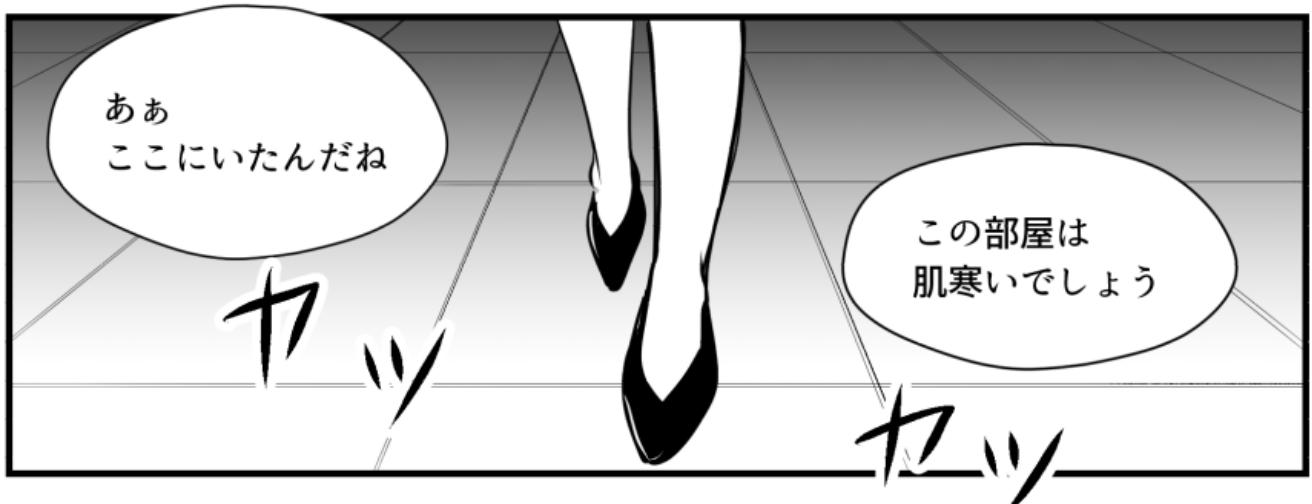
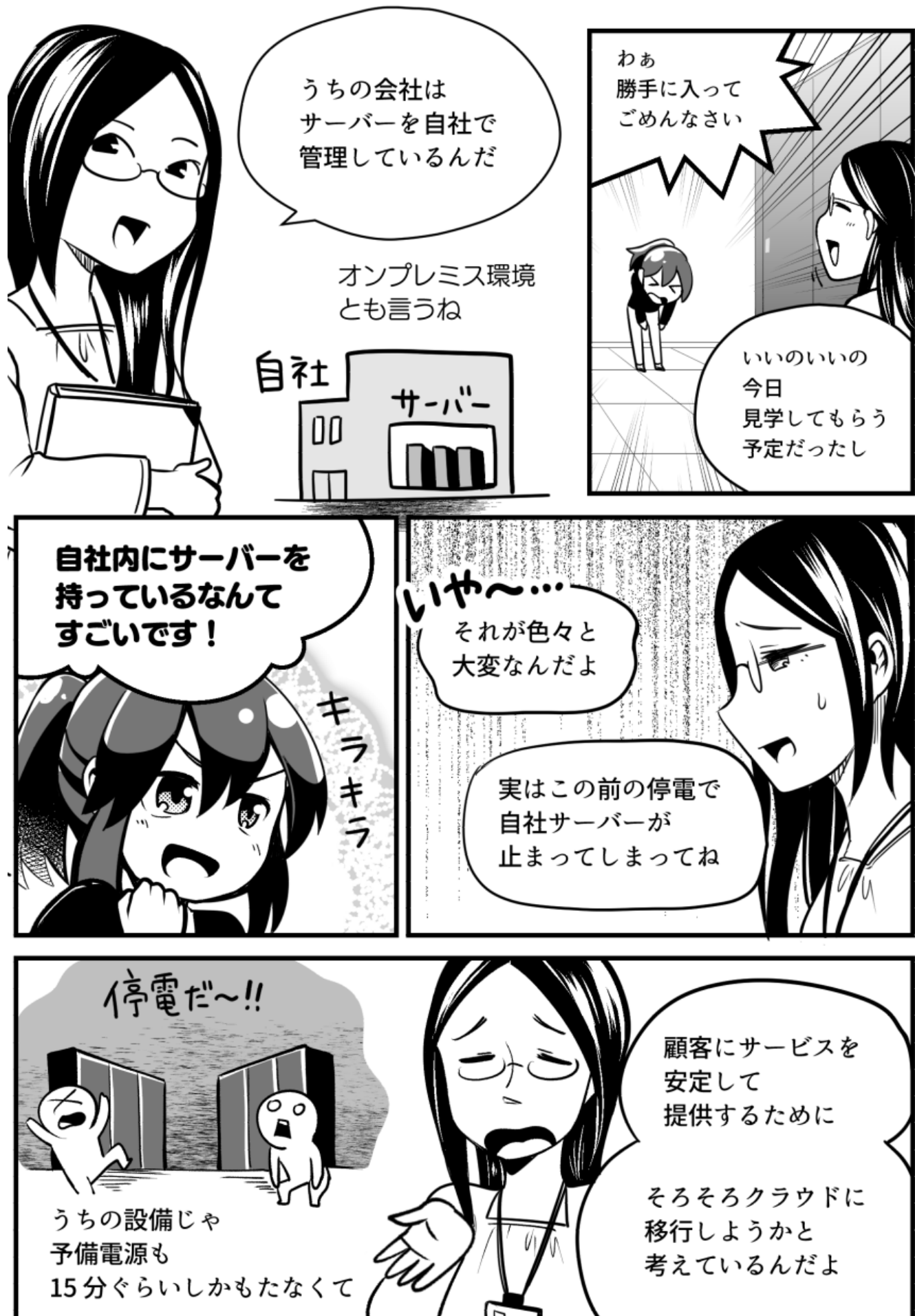


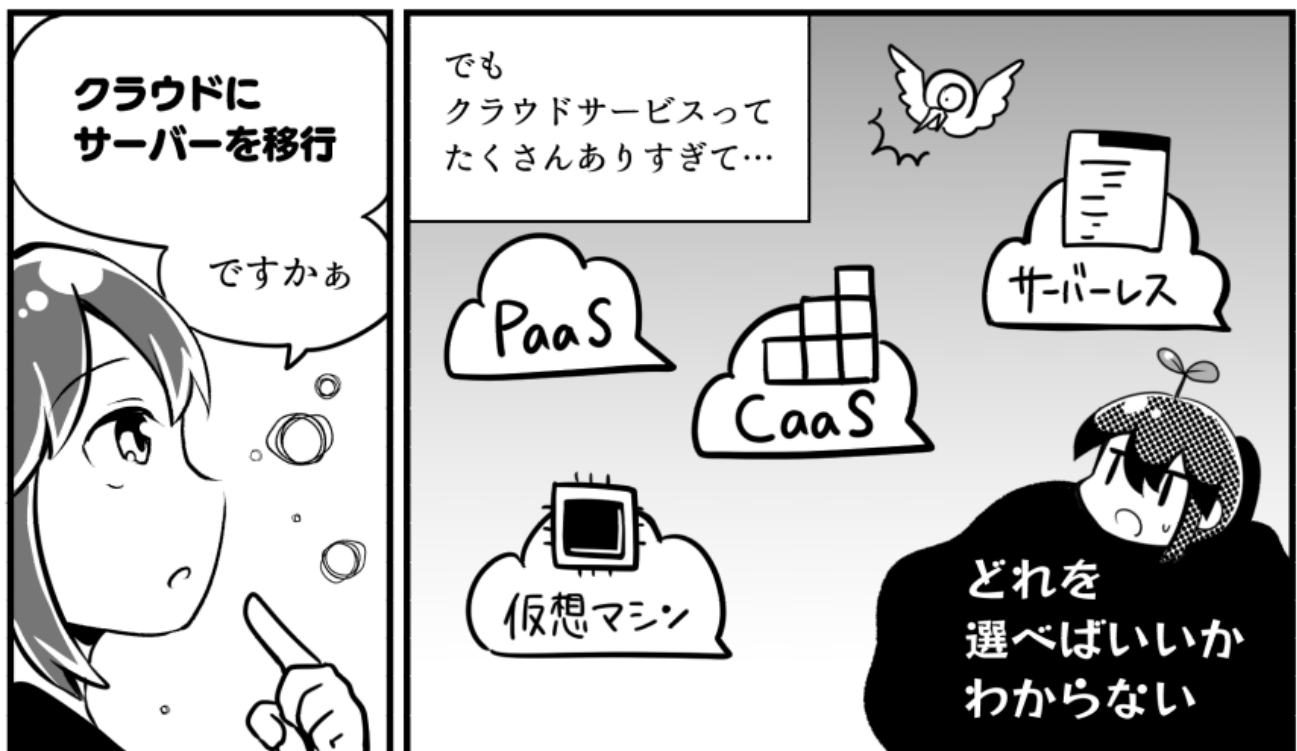
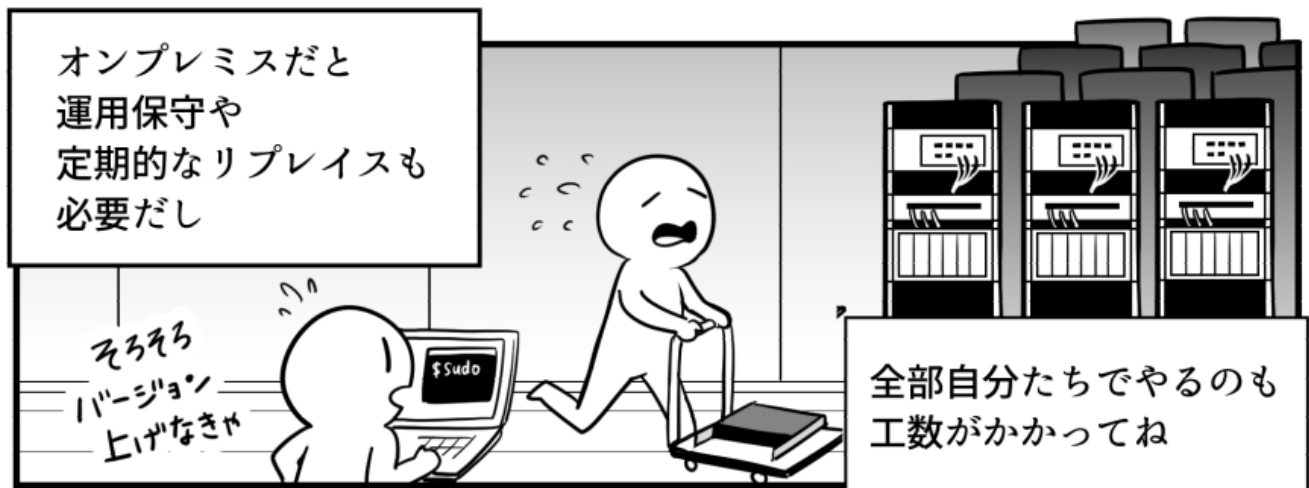
マンガでわかる Google Cloud

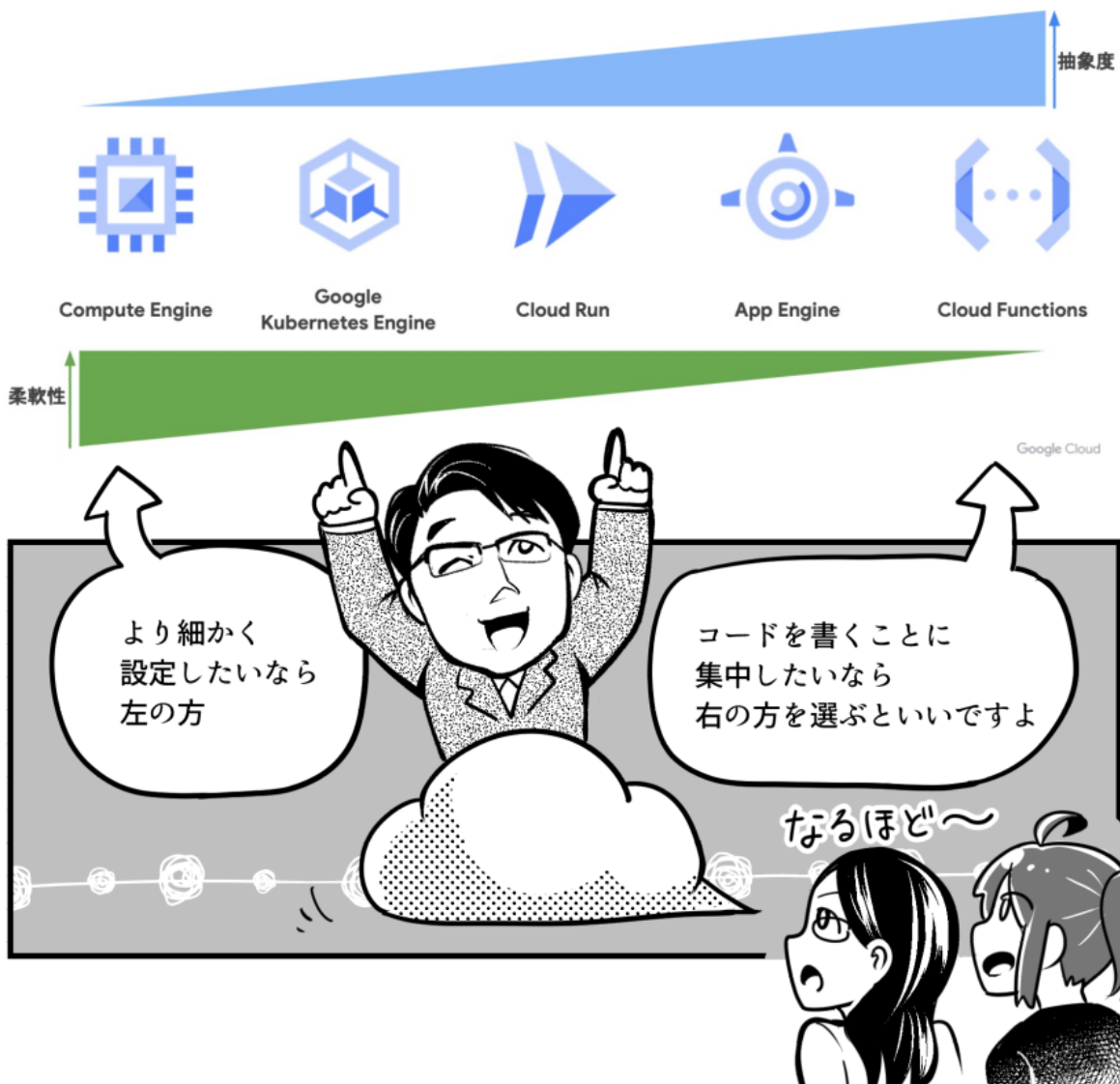
アプリケーション基盤編











概要

Google Cloud では仮想マシンからサーバーレスまで、アプリケーションを動かすための選択肢が複数用意されています。そして最適なものは利用者のユースケースにより変わり、これだけを使っておけば良いというものではありません。今回はそれぞれのプロダクトの特長、適するユースケース、また実際のシステムを想定しどのように考えてそれぞれのプロダクトを選択するかを、2022 年現在の最新情報を交えてご紹介します。

解説者



Google Cloud ソリューションズ アーキテクト 長谷部 光治

ゲーム会社でインフラ エンジニアとしてキャリアをスタート。その後、ユーザー企業でプライベート クラウド開発、パブリック クラウド導入をリード。外資系ソフトウェア ベンダーでは、クラウドのプリセールスからデリバリまで幅広く担当し、2018 年 9 月に Google Cloud Japan に参画。現在は Solutions Architect として、「Hybrid」「Retail」を軸に、お客様のクラウド利用促進に従事しています。

Compute プロダクトごとの役割を知ろう



Google Cloud は初心者だから、どのプロダクトがどの役割を果たしているかわからないのですが……？

実は、それぞれの実現モデルにあわせて、わかりやすく分かれているんですよ。

Google の Compute プロダクトごとの役割は次のとおりです。

- Compute Engine: 仮想マシン
- Google Kubernetes Engine: Container as a Service (CaaS)
- Cloud Run: サーバーレス
- App Engine: Platform as a Service (PaaS)
- Cloud Functions: サーバーレス (FaaS)

※ [CNCFWG Serverless Whitepaper v1.0](#) による分類とのマッピング



おお〜っ！

これならわかりやすい！

クラウド サービスは提供する企業によってプロダクト名が異なるから、こういうふうにそれぞれの役割がはっきりしていると助かるなあ。

では、これらについてひとつひとつ説明していきますね。
どのプロダクトが自社に合っているか、考えながら見ていきましょう。



Compute Engine

仮想マシン

- 柔軟なスペックの選択が可能
 - 高速な vCPU : ~ 112, メモリ : ~ 12 TB
 - ローカル SSD、GPU の利用
- 複数の OS を選択可能
 - 各種 Linux ディストリビューションから Windows Server まで



CentOS

ubuntu



debian

まずは Compute Engine から説明していきましょう。
これは言わずもがな、仮想マシンを提供するものです。



5 つの Compute プロダクトの中では、1 番柔軟性が高いものですね！

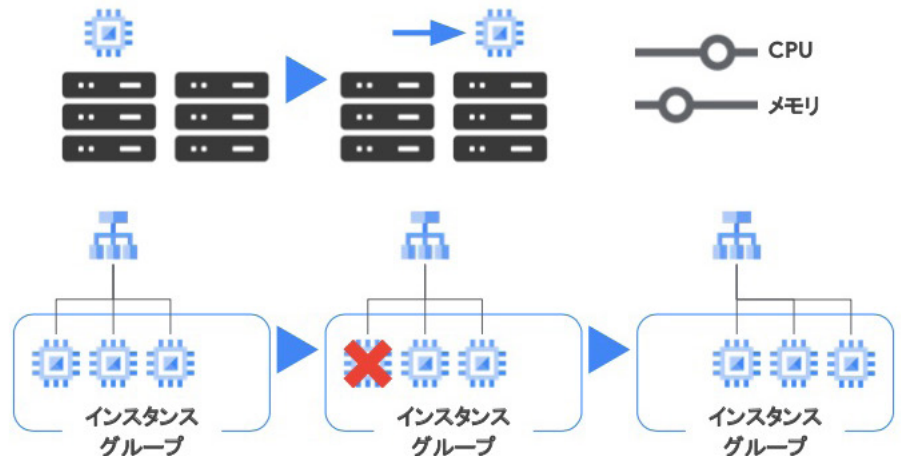
はい、そのとおりです。
処理性能が必要でしたら 100 vCPU を超えるものを、またメモリもテラバイト級のものを 1 台に積むことができます。
物理的な面では、ローカルな SSD や GPU も使えます。
OS も、各種 Linux ディストリビューションから Windows Server まで選べます。



Google Cloud

特長

- ライブ マイグレーション
- カスタム マシンタイプ
(CPU, メモリの柔軟な選択)
- インスタンス グループ
- 秒単位での課金



Google Cloud の Compute Engine の特徴は、ライブ マイグレーションです！



ライブ マイグレーション……？

- live は「生きている」という意味
- migration は「移行」という意味

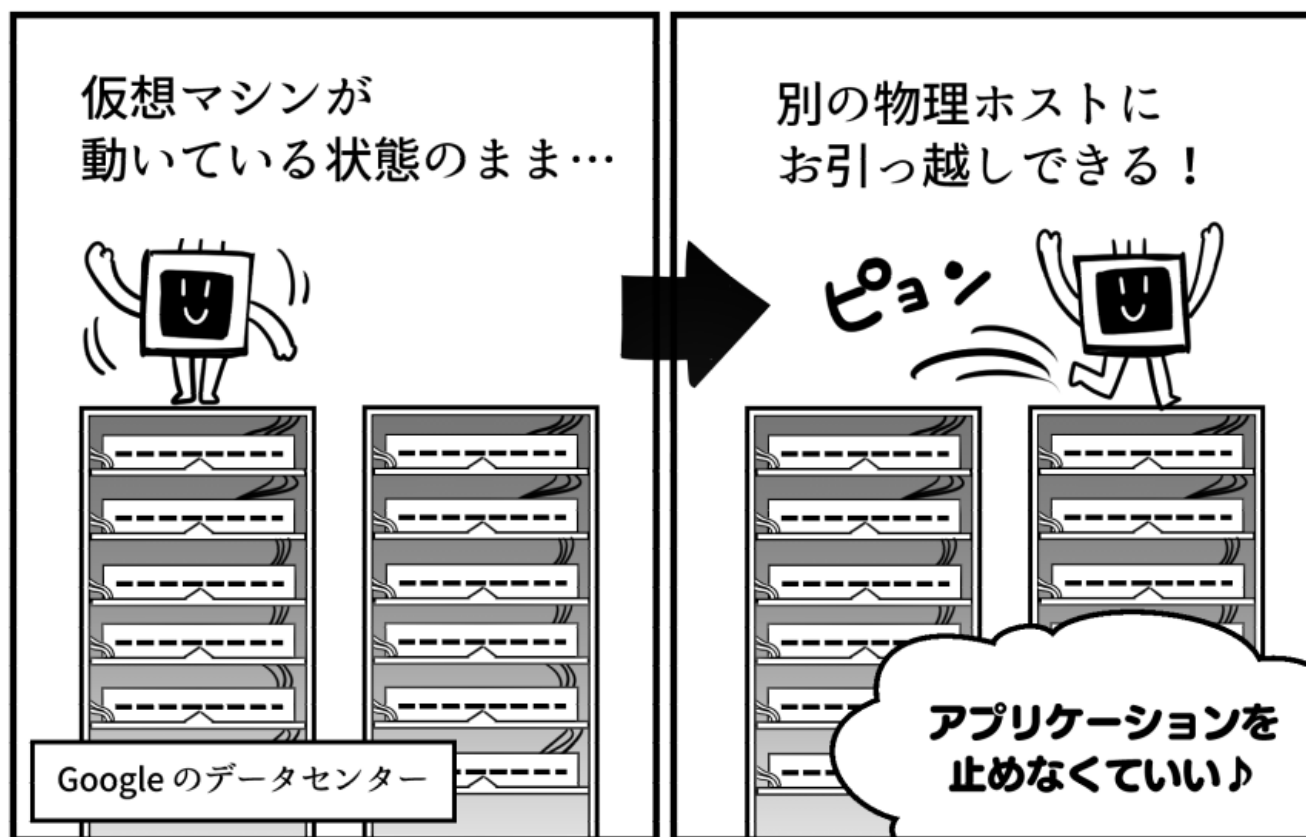
つまり「生きたまま移行する」ということですか？

そのとおりです！

実際は、Google のデータセンターの物理マシンの上で仮想マシンが動いているわけですが、その仮想マシンが動いている状態のまま別の物理ホスト上にお引っ越しできるというワケです。



ライブ マイグレーション



へえ～！ アプリケーションをストップさせずに、別のマシンへ引っ越せるんだ！
でもそれって、どういうときに使うの？

いい質問ですね！

たとえば、物理ホストに重大な脆弱性が判明したとします。
その場合、通常だとお客様に「仮想マシンを再起動してください」というお願いをするケースが多いと思います（お客様による再起動を受けて、脆弱性に対処済みの物理ホストに移動します）。
しかし、Google Cloud では、その必要はありません。
Google Cloud 側で、脆弱性の問題を含んだ物理ホストから、対処済みの物理ホスト上に、お客様の手をわずらわせることなく、稼働中の仮想マシンをそのまま移動できるんです。



それはありがたいですね！

次に特徴的なのは、カスタム マシンタイプです。
通常、仮想マシンを選ぶときって、CPU のスペックやメモリの量がある程度、決まりきっていることが多いですね。



たしかに。プルダウン型の選択肢になっていて、インスタンス タイプをそこから選ぶってことが多いかな。

Google Cloud では、CPU やメモリを自分で自由に組み合わせることができるんです！
たとえば「CPU 4 コア、メモリは少し多めにする」といったように、自分で細かくカスタマイズができるというワケです。

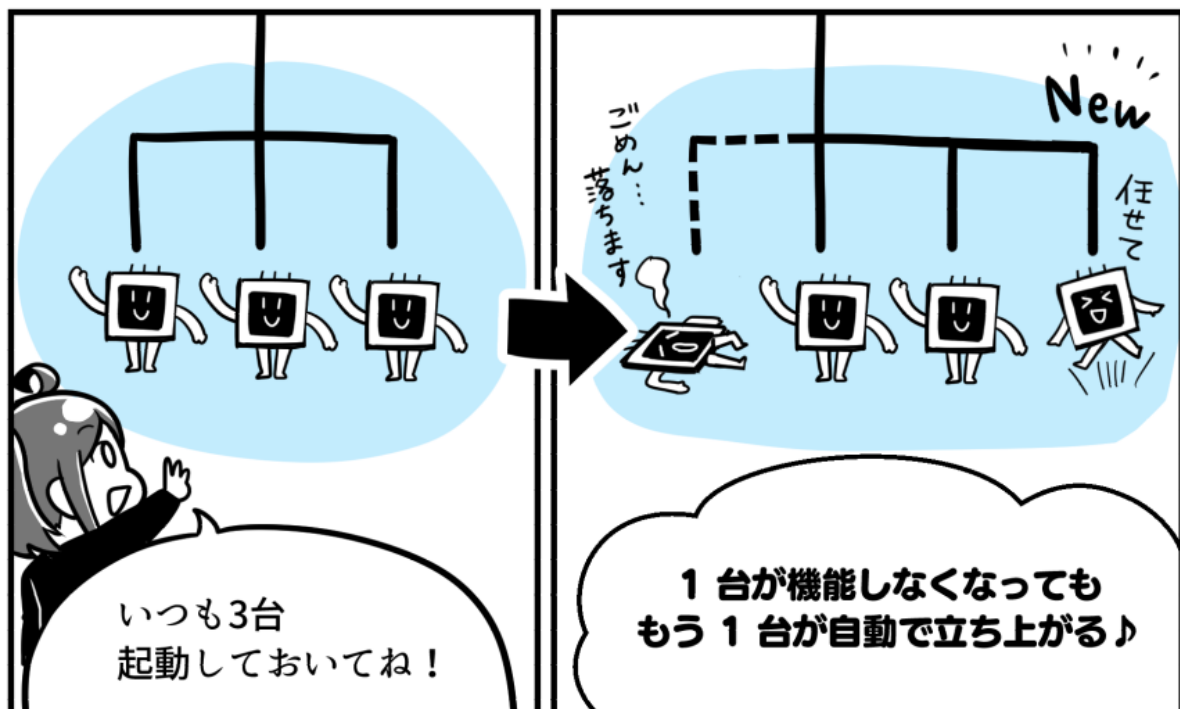


おお〜っ！
それなら、自社で使っているオンプレミスのサーバーのスペックにできるだけ近づけられそう！

インスタンス グループについても説明しますね。
インスタンス グループで「仮想マシンが 3 台稼働しておく」と設定しておく、3 台あるうち、1 台がおかしくなってしまって落ちてしまっても、もう 1 台が自動的に立ち上がります。



インスタンス グループ





人間がわざわざ再起動しなくてもいいんだね！

実際に、オンプレミス環境を 4 か月で Google Cloud に移行した事例があります。

参考にしてみてくださいね！



お客様事例

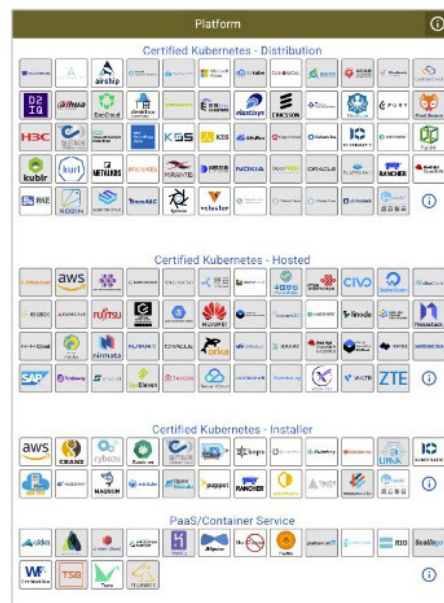
- [フォスター電機株式会社：SAP ERP を 4 か月で Google Cloud に移行、業務アプリのレスポンスとランニングコスト大幅改善](#)



Google Kubernetes Engine (GKE)

フルマネージドな Kubernetes サービス

- ユースケースに合わせた **2 つの運用モード**
 - スタンダード: ノード設定の **柔軟性**
 - Autopilot: **完全マネージド**な利用体験
- Kubernetes の **エコシステム**を活用



<https://landscape.cncf.io/>

Google Cloud

では次に、Google Kubernetes Engine (GKE) について知しましょう。

GKE は、フルマネージドな Kubernetes サービスです。
ユースケースに合わせた 2 つの運用モード（スタンダード / Autopilot）があり、Kubernetes のエコシステムを活用しています。





フルマネージドな Kubernetes サービス……？
横文字だらけ！
どういう意味ですか？

まず、マネージド サービスというのは、日本語に直すとそのまま「管理サービス」ですね。お客様に代わってサーバーを管理代行するアウトソーシング サービスのことです。



フルマネージド サービスは、マネージド サービスではカバーしきれない箇所も手厚くサポートします。たとえば GKE では、Google のサイト信頼性エンジニア(SRE)がクラスタを完全に管理します。これにより、クラスタの可用性が確保され、常に最新の状態に保たれるのです！



すごい！ Google のエンジニアが管理してくれるんですか！？
それなら Kubernetes を使いこなす方に集中できますね。



ところで Kubernetes って最近よく聞くけど、具体的にはどういうことができるんだろう？



あはは……。実は私もよく知らないんですよ。
今さら聞けないっていうか。

では Kubernetes 自体についてもおさらいしておきましょう。



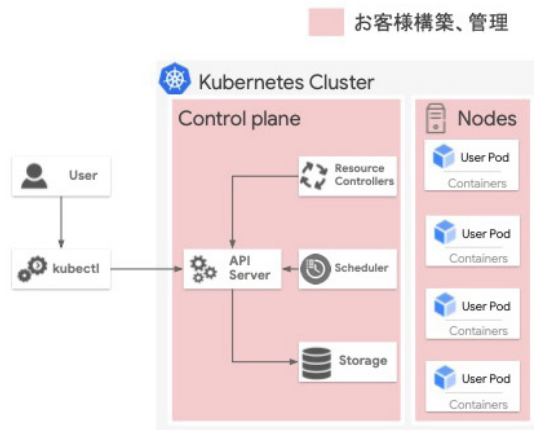


Kubernetes とは？

プロダクショングレードの

コンテナオーケストレーションシステム

- コンテナ オーケストレーション
ツールの **デファクト スタンダード**
- オープンソース
- スケジューリング、自動ヒーリング、
自動スケーリング
- 高い **拡張性**



Google Cloud

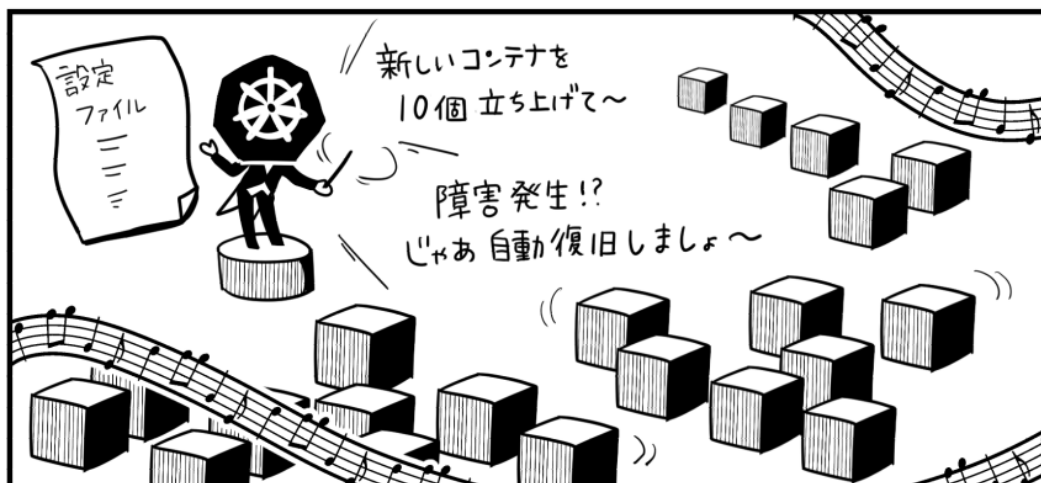
Kubernetes は、プロダクション グレードなコンテナ オーケストレーション システムです。



コンテナ技術が浸透して、本番環境で利用することも増えてきました。コンテナってリソース使用量が小さくて、ひとつひとつが軽いんですね。ひとつのアプリケーションにおいて、多数のコンテナが同時に動くというのが当たり前になっています。

よって、大量のコンテナを、オーケストレーション（設定、管理、調整の自動化）する必要が出てくるワケです。

コンテナ オーケストレーション システム



オープンソースのオーケストレーション ツールとして、
Kubernetes はデファクト スタンド（事実上の標準）ですね。

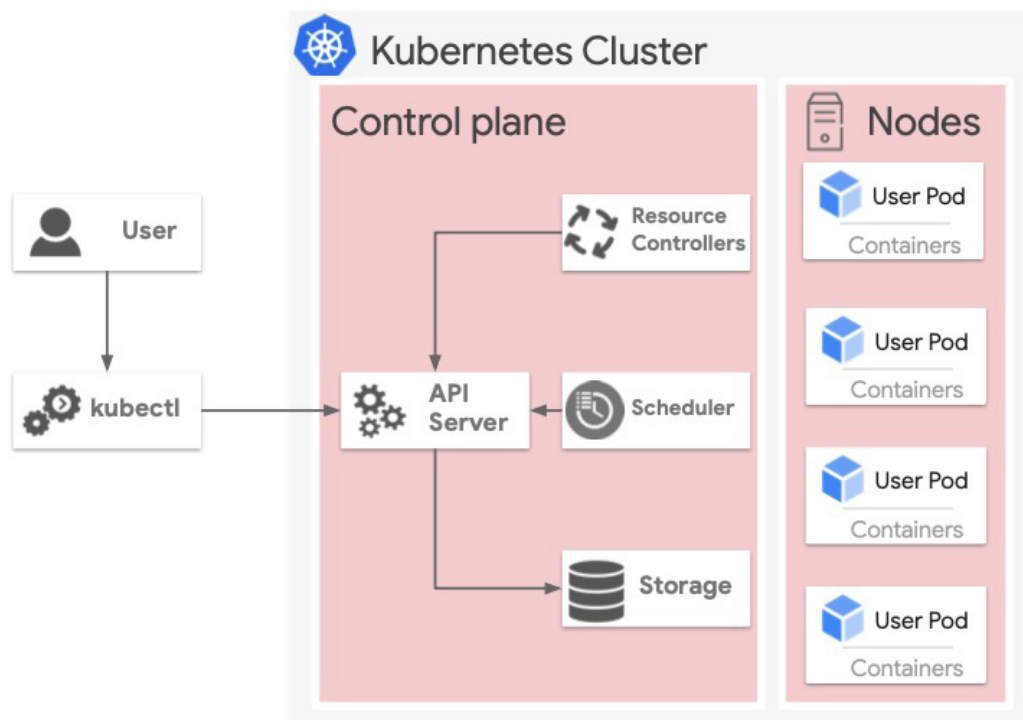


逆に、Kubernetes 以外のオーケストレーション ツールを知らなかった
です。

Kubernetes のコンポーネントの図を見てみましょう。
コンポーネントは大きく分けて Control plane と Nodes で構成され
ます。



お客様構築、管理



Control plane は、各種ノードやシステム自体をコントロールする部
分です。

Nodes というのは、お客様のコンテナが動く部分です。
オンプレミスの場合、赤く示した部分（Control plane と Nodes）を
皆さんが管理、構築していくことになります。



Google Cloud

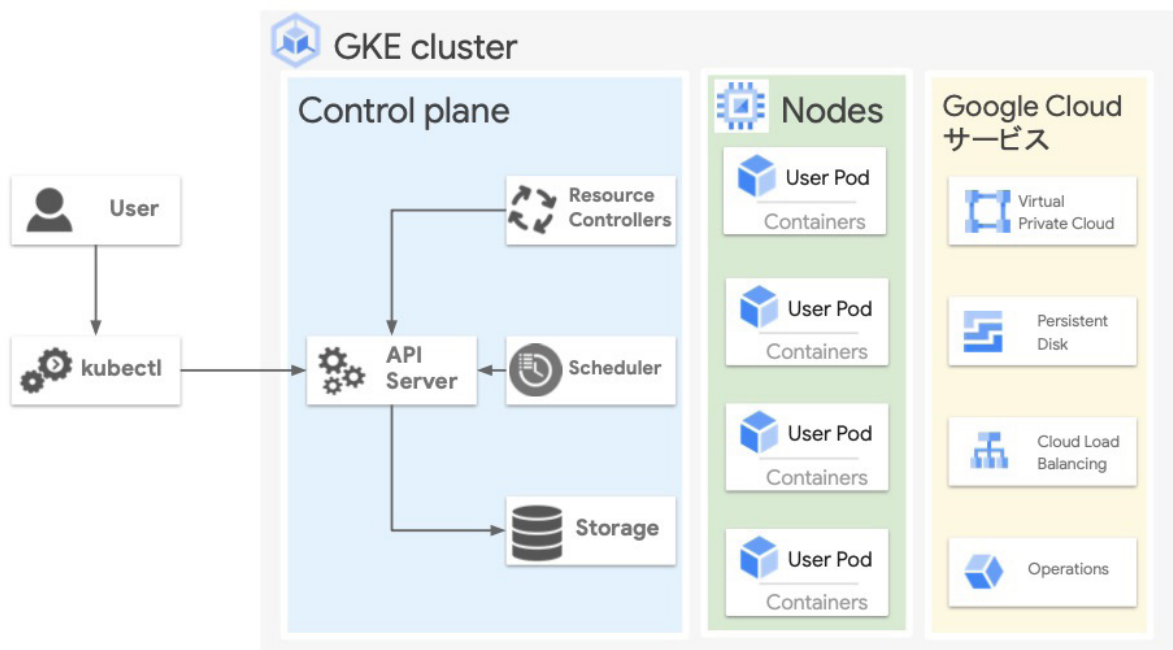


赤い部分って、全部じゃないですか〜！
これを全部まるっと管理するのは自信がないなあ。

そこで、GKE です！
これは GKE のスタンダード モードの図です。



-  GKE が構築、Google、お客様で管理
-  Google 管理
-  Google Cloud サービス



GKE スタンダード モードでは
Control plane は、Google が管理します。
Nodes は、GKE が構築し、お客様と Google が一緒に管理します。

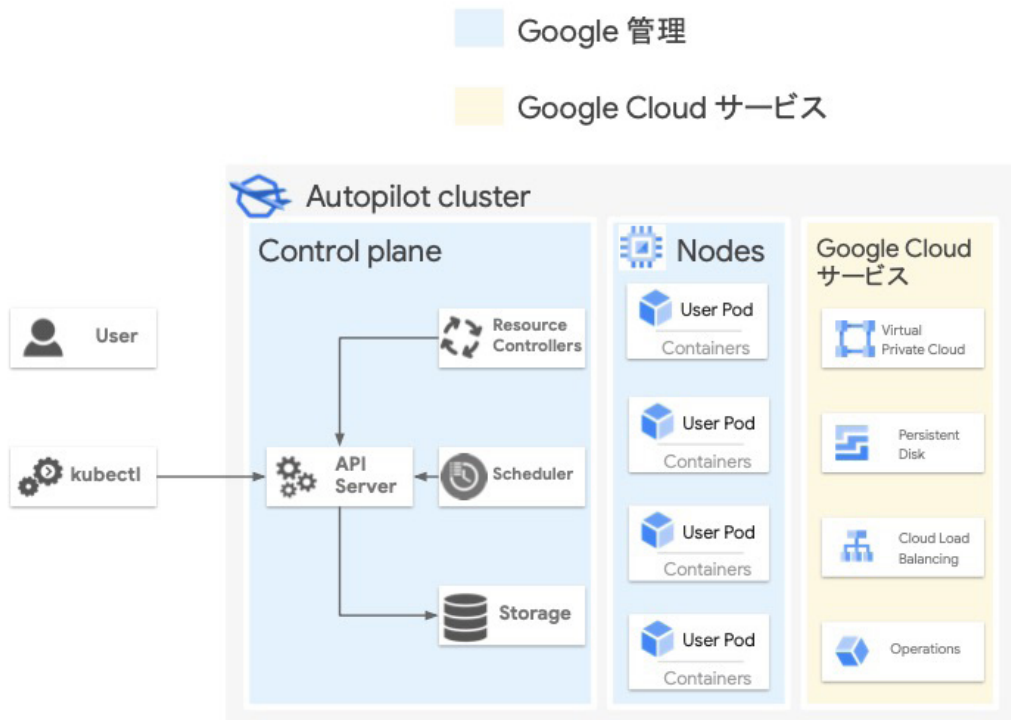


さらに、GKE スタンダード モードの特徴として

- ・セキュリティとコンプライアンスは HIPAA と PCI DSS に準拠
- ・自動でスケーリング、アップグレード、ノードを修復
- ・ハイブリッドとマルチクラウドをサポート

といった点が挙げられます。

さて、次は GKE の Autopilot モードを見ていきましょう。
Autopilot モードは Control plane に加え
Nodes も Google が管理するというのがポイントです。



すごい！ スタンダード モードでは、Nodes の管理は自分が必要があったけど、Autopilot モードは、Nodes の管理も Google がやってくれるんだ！

Autopilot モード



Autopilot モードは、スタンダード モードから、さらに進化し

- 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、ネットワーク設定など)
- Workload (Pod) ドリブンな世界へ
(Pod 単位での課金、Pod への SLA)

といった特徴があります。



よりワークロードに集中する形で Kubernetes が使えるんだね。
GKE はどんな場面で使われているの？

マイクロサービス アーキテクチャにおける大規模ウェブ アプリケーションや、
データ処理基盤として使われていることが多いですね。
活用事例がありますので、ぜひ参考にしてください。



■スタンダード: おすすめユースケース

- ノードの設定変更が必要なゲーム バックエンド
- GPU、TPU を使う ML パイプライン

お客様事例

- [株式会社スクウェア・エニックス / 株式会社コロプラ:『ドラゴンクエストウォーク』に Cloud Spanner、GKE 活用](#)

■ Autopilot: おすすめユースケース

- スケーラブルな負荷テストツール用環境
- 必要リソースに変動の少ない Static なシステム

お客様事例

- [株式会社ブレイド: GKE Autopilot で Ops レスな リアルタイム ML を実現](#)

Cloud Run

コンテナ + サーバーレス

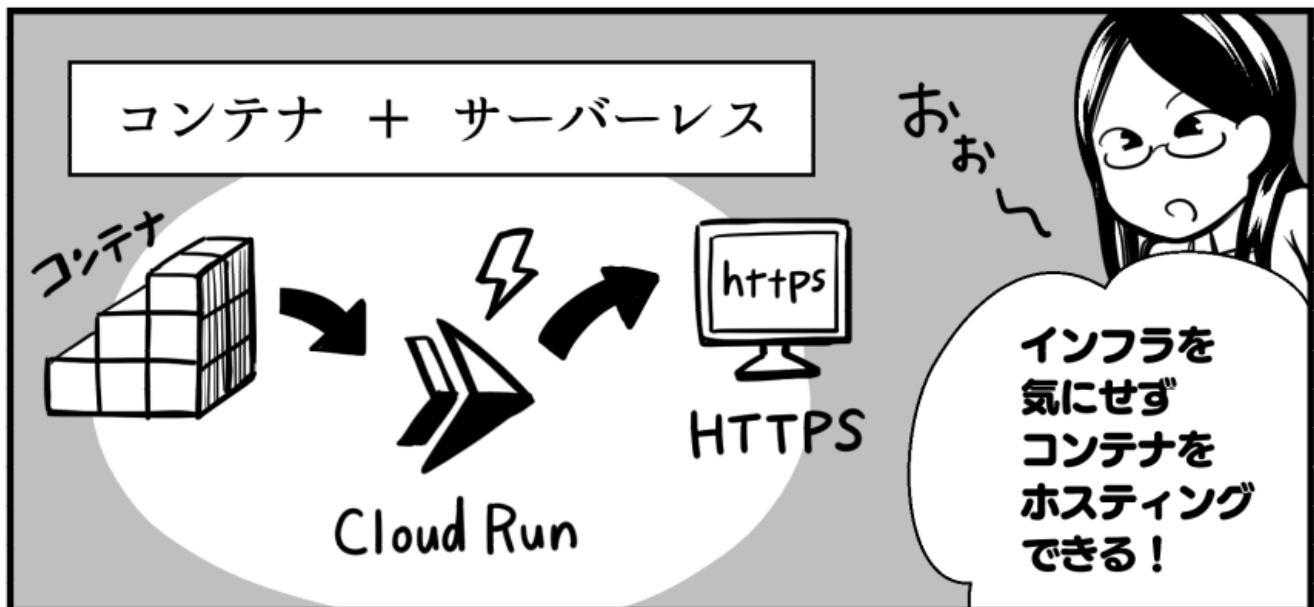
- 100 ミリ秒単位での課金
- オープンソースの **Knative** と **API 互換** になっていることによる移植の容易性
- ユースケースに合わせた **2つの選択肢**
 - Cloud Run: フルマネージド
 - Cloud Run for Anthos: 柔軟性、ハイブリッドクラウド



※ ここからの Cloud Run については、フルマネージド版をベースに説明を進めます

Google Cloud

Cloud Run の特徴



さて、次は Cloud Run についてです。
Cloud Run は、コンテナとサーバーレスを組み合わせたようなプロダクトです。





コンテナでサーバーレスを行えるってこと？ そんなことができるんだ！？

API に関しては、オープンソースの Knative と互換性があります。皆さんが自社で動かしているコンテナ アプリケーションを、Cloud Run に持ってきて動かすことができますし、その逆も然りです。



Cloud Run は、ユースケースに合わせて

- Cloud Run: フルマネージド
- Cloud Run for Anthos

という 2 つの選択肢があるのですが、ここからはフルマネージド版をベースに説明を進めます。



Cloud Run (フルマネージド)

特長

- 0 ~ N へトラフィックに応じて
高速にスケーリング
- Eventarc と連携し**イベント駆動**で
処理を実行
- HTTP/2, WebSocket, gRPC への対応

第 2 世代の実行環境 Preview

New

- CPU, ネットワークパフォーマンス向上
- すべてのシステムコールなど Linux との**完全互換性**
- **ネットワーク ファイルシステム** のサポート



Eventarc



Cloud Run の特徴は、高速なスケーリングです。トラフィックに応じて、自動でコンテナの数を調整してくれます。



0 個から N 個（開発者が決めた数）まで、コンテナを減らしたり増やしたりしてくれるんだ！

そうです。

トラフィックの場合に応じて、ゼロスケール（コンテナの数をゼロにすること）も可能なんです。

イベント駆動で処理を行うことができるのも、コンテナ + サーバーレスと呼ばれる所以です。



なるほど、たしかにリクエストが来たときだけ起動して処理するのは、FaaS みたいですね！

現在プレビュー版で、Cloud Run の第 2 世代が出ています。

Cloud Run 第 2 世代の特徴

- CPU、ネットワーク パフォーマンス向上
- すべてのシステムコールなど Linux との完全互換性
- ネットワーク ファイル システムのサポート



このように、もともとの Cloud Run ではできなかったことができるようになっています。



■ Cloud Run（フルマネージド）：おすすめユースケース

- ステートレスなモバイル バックエンド
- 軽量なデータ変換処理
- イベント駆動処理

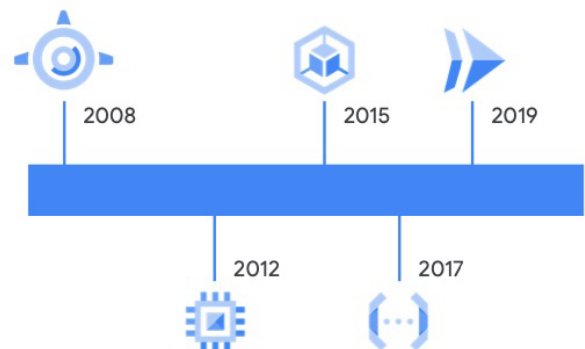
お客様事例

- [セイコーソリューションズ株式会社：クラウド辞書サービス『GIGANTES』を Cloud Run で構築](#)

App Engine

アプリケーション開発に集中するための
マネージドなサーバーレスプラットフォーム

- インフラ管理の必要なし
- 2008 年にプレビュー版がリリースされから
提供を続けてきた **信頼性、エコシステムの大きさ**
- 用途に合わせた App Engine 環境の提供



さて、折り返し地点を過ぎましたよ！
4 回目、App Engine のご紹介です。

App Engine は、アプリケーション開発に集中するためのマネージド
なサーバーレス プラットフォーム（PaaS）です。

右の年表を見てください。

実は、5 つの Compute プロダクトのうち、App Engine は最も歴史が
長くて信頼性のあるサービスなんです。



わあ、本当だ！ 仮想マシンが出る前の、2008 年からあるんだ！

App Engine のランタイム（アプリケーションの実行環境）はいろいろ
ありますが、今「App Engine」というと、次の図の真ん中にある第
2 世代を指すことが多いですね。



App Engine のランタイム

環境	スタンダード 第 1 世代	スタンダード 第 2 世代	フレキシブル
開発言語	Python 2.7 Java 8 PHP 5.5 Go 1.11	Python 3.7,3.8,3.9 Java 11 Node.js 10,12,14,16 PHP 7.2,7.3,7.4 Ruby 2.5,2.6,2.7 Go 1.12+	Node.js, Ruby, Java, Python, Go, PHP, .NET カスタムコンテナ イメージ
実行環境	サンドボックス		仮想マシン上の Docker コンテナ
Google Cloud 機能の利用	GAE 専用の API Google Cloud の API	Google Cloud の API を利用 ※	
サードパーティ バイナリのインストール	不可	可	

※ [Python 3](#), [Java 11](#), [Go 1.12+](#) では第 1 世代からのマイグレーション手段の提供を目的とし、一部の GAE 専用の API が利用できるようになっています

Google Cloud

開発言語に注目してみましょう。

ご覧のとおり、第 1 世代は古めの言語、第 2 世代は新しめの言語がメインです。

ですので、今から App Engine を使って何かアプリケーションを作るのであれば、スタンダードの第 2 世代がおすすめです。

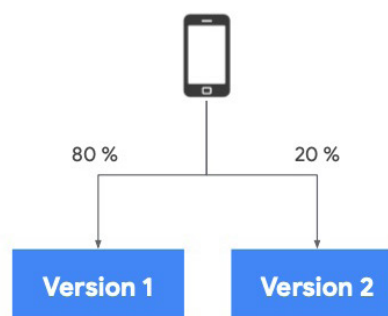
もともと App Engine を使っているお客様は、そのまま第 1 世代を使っているという具合です。



App Engine (スタンダード環境 第 2 世代)

特長

- トラフィック分割
- アプリケーション バージョニング
- トラフィックに合わせた
高速なスケーリング



App Engine も、トラフィックに合わせた高速なスケーリングが可能です。

また、右の図のように、アプリケーションのバージョン 1 とバージョン 2 を用意し、トラフィックの 80% をバージョン 1 へ、残りをバージョン 2 へ、といったことも可能ですよ。



■ App Engine: おすすめユースケース

- スパイク特性のあるウェブ アプリケーション
- 静的、動的コンテンツを利用するウェブ アプリケーション
- モバイル バックエンド

お客様事例

- [NTTコミュニケーションズ株式会社：サーバーレスソリューションで新サービスを少人数、低コストにアジャイル開発](#)



静的コンテンツを持つこともできるんですね！

私たちの会社で運用しているウェブ アプリケーションは、静的コンテンツと動的コンテンツ、両方があるので助かります。

Cloud Functions

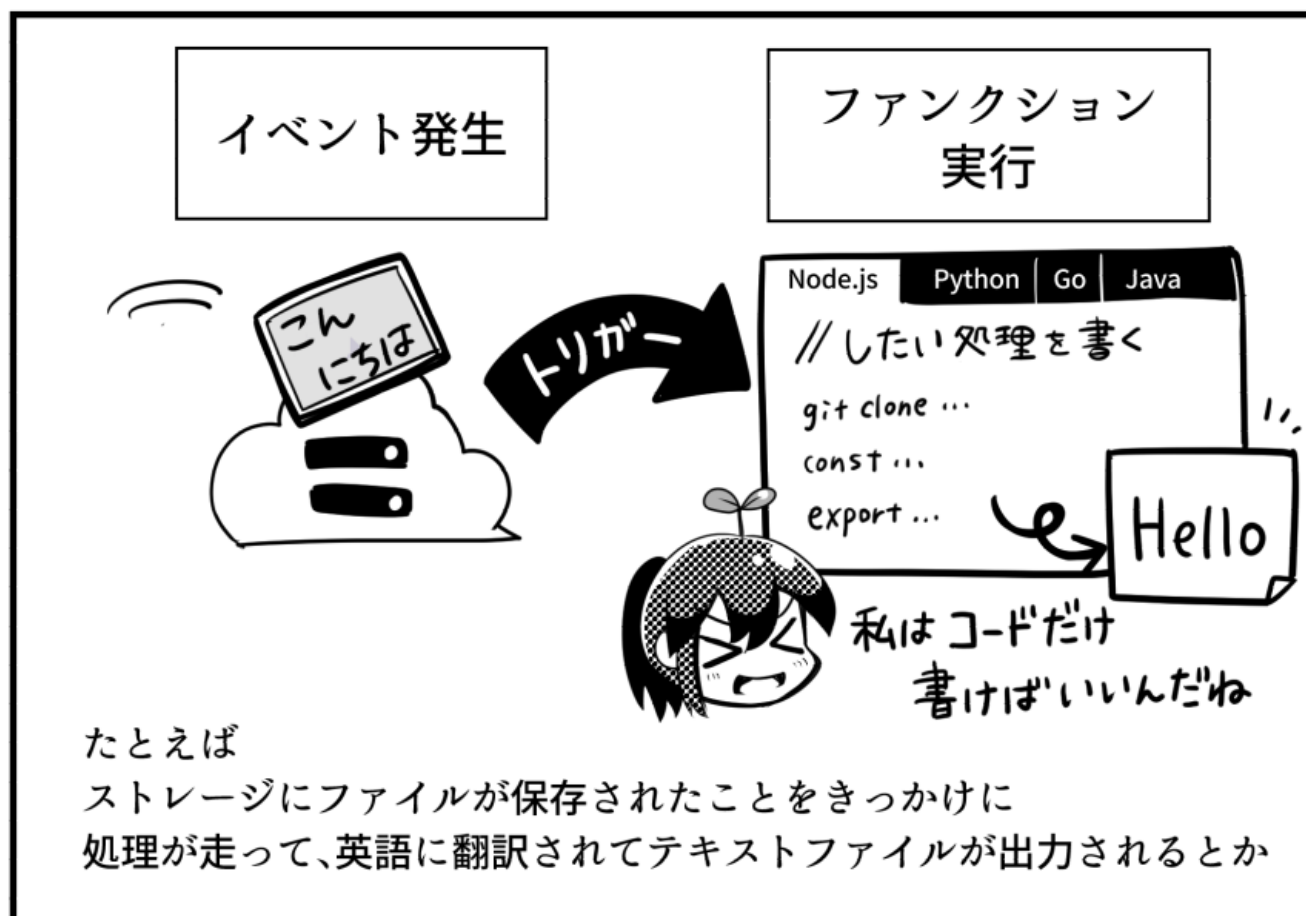
Cloud Functions は、サーバーレスな FaaS (Functions as a Service) です。

様々な Google Cloud のイベントと連携することができます。

たとえば「ストレージになんらかのファイルが置かれた」ということをトリガーにして、ファンクションを呼び出して処理を実行する、といったことが可能です。



Cloud Functions の活用例



Cloud Functions

特長

- オープンソースの FaaS フレームワークを利用し、ログインを回避
- Cloud Trace、Cloud デバッガといった Google Cloud の開発ツールとネイティブに連携

Cloud Functions 第2世代 Preview

New

- リクエスト処理時間が **最長 60 分** に
- 最大 16 GB メモリ、4 vCPU に
- 最小インスタンス設定、トラフィック分割が可能に

アプリケーション コード

Functions Framework



Cloud Functions の特徴として、オープンソースな FaaS フレームワークを利用し、ロックインを回避することが可能です。また、CloudTrace、Cloud デバッガといった Google Cloud の開発ツールと最初から連携しています。



Cloud Functions 第2世代は、簡単にいうと、よりパワフルになりました。もともと Cloud Functions というと、軽い処理をメインに扱うことが多かったのですが、もう少しやれることが増えていますし、ユースケースも広がっています。

■ Cloud Functions: おすすめユースケース

- ・サードパーティ サービス、API との統合
- ・リアルタイムでのファイル処理

お客様事例

- [ヤフー株式会社: Tech Acceleration Program で KPI に即したアーキテクチャを短期間で開発し AI を駆使した広告審査を実現](#)

ユースケースの整理

では、以上を踏まえて 5 つの Compute プロダクトのユースケースを整理してみましょう。



		Compute Engine	GKE	Cloud Run	App Engine	Cloud Functions
サードパーティアプリ		◎	○	△		
物理、仮想マシン移行		◎	○	△		
Web、モバイル バックエンド	ステートレス	△	○	◎	◎	○
	ステートフル	○	◎	○		
内部 API	ステートレス	△	◎	◎	○	△
	ステートフル	○	◎	○		

注: マークが無いところが実現不可能というわけではありません
運用負荷なども勘案し、あくまで **まず◎から検討してみる** という基準とお考えください

- GKE か、サーバーレスプロダクトを選ぶかが**大きな分岐点**となる

		Compute Engine	GKE	Cloud Run	App Engine	Cloud Functions
バッチ	処理時間長、またはリソースが必要	◎	◎	○		○
	軽い	△	◎	○		○
ワークフロー			△	○		◎
Google Cloud のイベントをソースに起動			△	○		◎

左の欄のようなことをやりたいのであれば、二重丸のついているところから検討していくといいですよ。



印がないところは絶対使えないの？

そういうわけではないので、安心して下さいね。運用負荷なども考えて、あくまで基準として考えてください。



運用していく上での考慮点



Google Cloud の 5 つの Compute プロダクトについて理解が深まりました！



でも、私たちの会社で運用していけるかなあ。費用面も気になるし……。

ズバリ、そこが気になりますよね。
ここからは、運用していく上で考えるべき点について解説します。
まずはチーム構成についてです。



チーム構成

CCoE のような、サービスを
横断で管理するチームがいる場合

- そのチームに GKE の管理を任せることで
 - マルチテナント構成でリソースの有効利用
 - 知見の集約、他チームへの展開
 - 何より Kubernetes のエコシステムの活用



たとえば、CCoE のような、クラウド化を推進する専門チームがいる場合は、GKE の最新情報のキャッチアップ、ベスト プラクティス、ガイドラインの策定を任せることで大変な部分を低減できると思います。

そうすることで知見の集約ができますし、得た知見を他のチームへも活用できます。



もうひとつ、Compute プロダクトの統一を考えてみる手もあります。アプリケーションごとの特性に合ったプロダクトを選ぶことも重要ですがワークロードを一覧で眺めてみて「Cloud Run で全部できるんじゃないか」という場合には、Cloud Run に統一してみるといった具合です。

なるほど！

手当たり次第に使うのではなくて、あえて1つのプロダクトにした方が、学習コストも減らせますね！

運用の手間もありますし、プロダクトごとのキャッチアップも大変ですし……。

そのとおり！

コストの観点は忘れてはいけませんね。

コストは利用料金のことだけではなく、潜在的なコストも含めて考えることが重要です。



■利用料金

- ・プロダクトごとに異なる、コストがかかるポイント
- ・リソースの考え方
- ・価格付けから読み取れる適切な使い方

■潜在的なコスト

- ・Day 2 operations で発生するコスト
- ・プロダクトの学習コスト
- ・将来、他のテクノロジーに移行するコスト
- ・エキスパートを採用するコスト

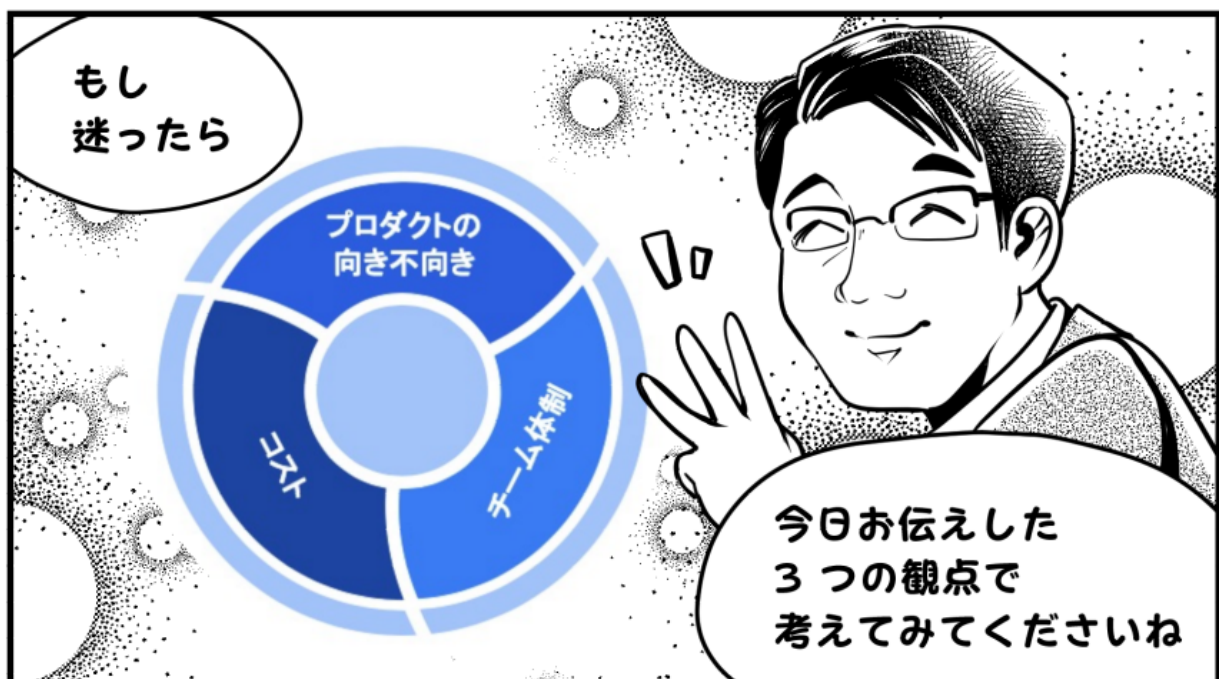


利用料金だけでなく、運用しはじめてから（Day 2 operations）のコストも考えなくちゃいけないんだね！

Day 2 operations で発生するコストも考えよう！



まとめ



Google Cloud 概要 / アジェンダ スピーカー 公開資料 FAQ セッション動画を観る

開催中のセッション 2021年開催 (アーカイブ)

CLOUD ONAIR ソリューション チャンネル

App Modernization OnAir

モダンなアプリケーション開発について紹介します。

今日は 2022 年の最新情報を受けた Google Cloud プログラムの特長とユースケースをご紹介します。ソースコードからコンテナをデプロイするまでの流れ、DevOps の進め方、GKE でマイクロサービスの構築をする基盤から高度なトラフィックルーティングと SLO 運用、Config Controller で安心安全な Google Cloud 環境を作り上げる手法についてご紹介します。

※ 株式会社ならびに報道関係者からの申し込みはお断りさせていただきます。
 ※ お客様のご情報は Google Cloud 及びパートナーから、セミナー等の案内をお送りするために使用させていただきます。
 ※ ご記入いただいたご登録情報は、イベントの協賛・協力企業に提供する場合があります。お客様のご登録情報は、協賛・協力企業よりご案内などもご提供するために共有されます。個人情報は各社の個人情報保護ポリシーに準って適切に扱われます。

#gc_appmodern

セッション動画を観る

★もっと知りたいあなたへ

Google Cloud の知見を Cloud OnAir で配信しているのでいつでも視聴しにきてくださいね

過去の放送もバックナンバーからご覧いただけますよ！

Google Cloud

アプリケーションはどこで動かすべきか - 2022 春

Google Cloud
ソリューションアーキテクト
長谷部 光治

それではまた！

ビュッ

ありがとう長谷部さん！

みんなのクラウドの疑問を解決するため

長谷部さんは今日も行く……！

Links

▼ Google Cloud を用いたモダンなアプリケーション開発が動画でわかる！

「Cloud OnAir」放送中（バックナンバーも視聴可能）

- [App Modernization OnAir](#)

▼本記事のもとになったセッションはこちら

- [アプリケーションはどこで動かすべきか 2022 春 / 長谷部 光治](#)

漫画制作



湊川 あい

IT 漫画家。マンガと図解で、技術をわかりやすく伝えることが好き。著書『わかばちゃんと学ぶ Git 使い方入門』『わかばちゃんと学ぶ Google アナリティクス』『わかばちゃんと学ぶ Web サイト制作の基本』『運用ちゃんと学ぶ システム運用の基本』が発売中のほか、マンガでわかる Git、マンガでわかる Docker、マンガでわかる Ruby といった分野横断的なコンテンツを展開している。

- [Amazon 著者ページ](#)
- [湊川あい Twitter @lminatoll](#)

製品、サービスに関するお問い合わせ



goo.gl/CCZL78

Google Cloud の詳細については、上記 URL もしくは QR コードからアクセスしていただくか、同ページ「お問い合わせ」よりお問い合わせください。

© Copyright 2022 Google

Google は、Google LLC の商標です。その他すべての社名および製品名は、それぞれ該当する企業の商標である可能性があります。