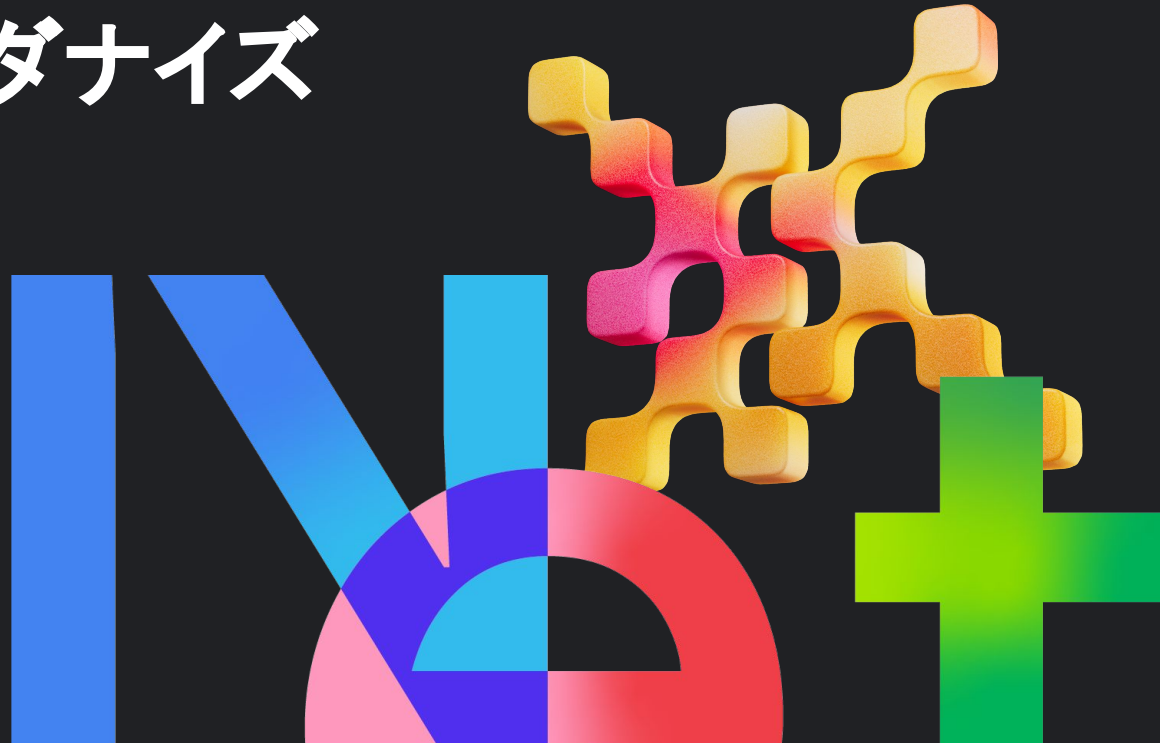
大規模レガシーを刷新！
手動運用からの脱却と
CI / CD によるモダナイズ



Google Cloud Next Tokyo

片切 亮

株式会社マクロミル

Tech PM



MACROMILL
a Macromill Group company

Proprietary

アジェンダ

- プロジェクト アプローチと
  グランド デザイン策定

  Tech PM Katagiri



- Google Cloud 上での実装と最適化の
  技術詳細

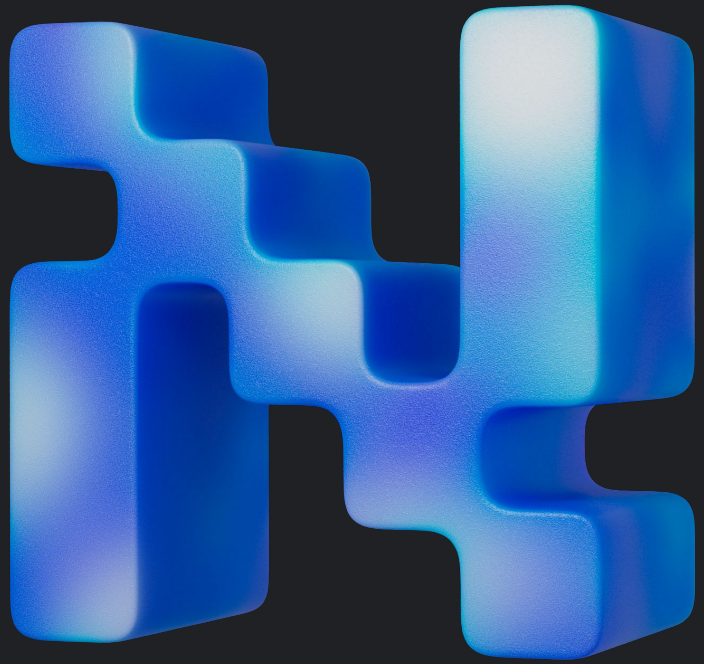
  Tech Lead Allan

Proprietary

01

はじめに

マクロミル サービスとプロジェクト背景



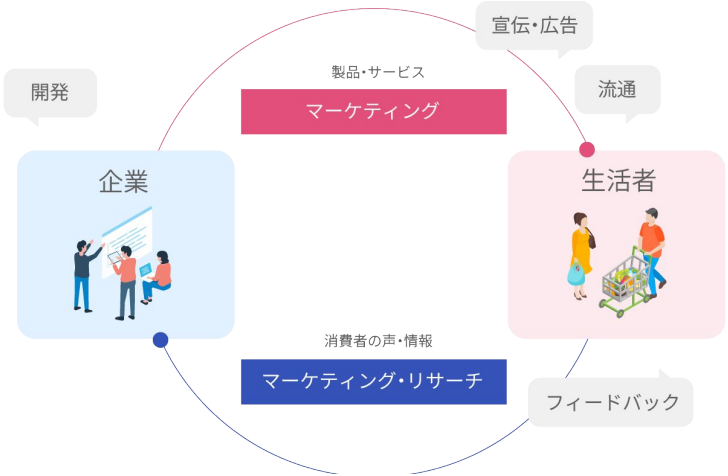
Proprietary

マクロミルとは

マクロミルは日本最大級のオンライン　リサーチ会社です。

インターネット上で多くの方にアンケートに答えて頂き、その結果を企業様の商品開発やマーケティングに活用頂く。

- **国内モニター数　3,600 万人**
- **アンケート配信から回答収集・集計・納品までリサーチのプロセスをオンラインで提供**
- **企業のマーケティング　リサーチ・商品開発・社会調査等を支える**





Proprietary

マクロミルのリサーチを支え続けた　20 年

運用

手動リリース、手動スケール

Web Server

Virtual Machine **ステートフル・サーバ台数固定**
40 以上の連携システム/1500 TPS 以上のトラフィック

Batch Server

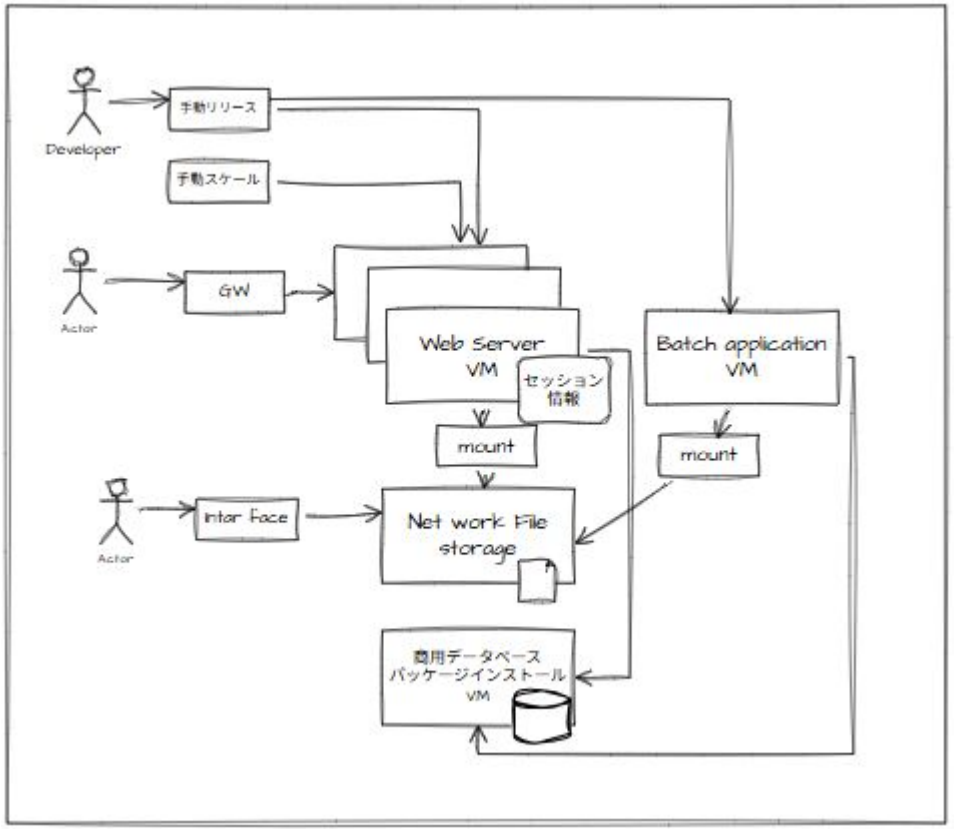
Virtual Machine 上で 約 100 のバッチ処理

File Management

Network Storage

Database

Virtual Machine ＋パッケージ インストール＋
有償ライセンス



システムが抱えていた成長の限界



スケーラビリティ

- サーバースケール困難なシステム構成



手動作業の限界

- 手動デプロイ
- 手動スケール
- 手動テスト
- サービス停止を伴うリリースと夜間体制の構築



コスト

- 有償ライセンス
- スケール困難な固定費
- 運用コスト
  - パッチ適用
  - データ クリーニング



- ❑ **リリース時間、リリースサイクルの長期化**
- ❑ **運用コスト、ライセンス費用の増加**

Proprietary

インフラの刷新とモダナイズへ

1 年間という期間限定の集中プロジェクトとして、基幹システムの基盤を刷新する

運用コスト削減と

モダナイズ & 無停止リリースの実現を目指す

Proprietary

02

プロジェクト キックオフ

1 年は無理 – そんな空気の中で動き出した



Proprietary

キックオフまでの道のり



最初はバラバラ

1 年は無理 ー
そんな空気の中で
動き出した



たたき台作成

課題・ボトルネック

マイグレーション目的
を整理



まる一日

ディスカッション

Mgr、Tech Lead、
既存開発メンバー、
SRE、全員で！



Tech Lead による

Dockerize デモ

All hands (IT 部門全
体会議) でのデモ

いけるかも
と思った瞬間



キックオフへ

覚悟が決まった日

「本当にやるのか」— そんな気持ちが少しずつ「やれる」に変わっていった

Proprietary

そしてキックオフへ

- 課題と移行戦略
- このプロジェクトの目指すゴール
- プロジェクト アプローチ
- タイムライン

Proprietary

プロジェクト移行戦略

スケーラビリティ課題

Google Kubernetes Engine (GKE)＋Valkey オートスケール

セッションを Memorystore for Valkey へ保存しステートレス化

手動作業の限界

GitHub Actions

CI/CD による自動デプロイ

無停止リリースの実現

コスト

AlloyDB

ライセンス費用削減
運用最適化・スケーラビリティ向上

Proprietary

プロジェクトで目指すゴール



システムをモダナイズすることで、迅速なリリース　/無停止リリースできる環境を構築する

CI/CD によるデプロイ自動化とオートスケール



利用ユーザ・プロダクトへの透過性維持と透過的なリリースを実行

利用者影響ゼロを目指す
透過的な切り替え

Proprietary

プロジェクト アプローチ

Transparency

- 20 名以上の大規模プロジェクト
  - 3 つのスクラムチーム
- Sprint Review を月 2 回実施
  - CI/CD デモ、アーキテクチャレビュー
- ステークホルダー・関連システムへ進捗を常に見える状態へ

建設的なディスカッション

- 保守的にならない、否定から入らない
- Tech Acceleration Program にて建設的なディスカッションでアーキテクチャ ポリシーを策定した

リスペクト

- 既存システムへのリスペクト
- 既存システム×新しい技術スタック、双方の信頼関係と歩み寄り

Proprietary

タイムライン

2025 年 7 月

キックオフ・移行戦略

7/22 - 23

Tech Acceleration Program

2025 年 12 月

開発 / UT 完了

2026 年 1 月

GCP テスト環境構築完了

2026 年 6 月

結合/性能/障害テスト完了

2026 年 7 月

リリース

Proprietary

03

グランド デザイン策定

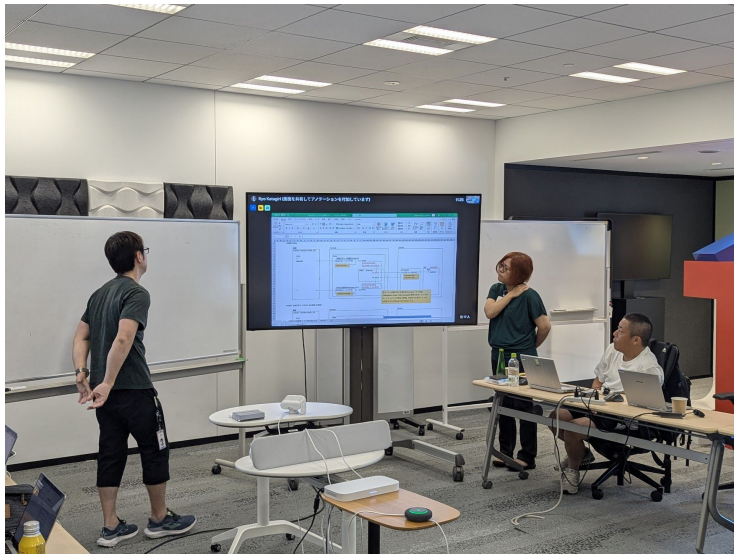
Tech Acceleration Program



Proprietary

Tech Acceleration Program でアーキテクチャのグランドデザインを描く

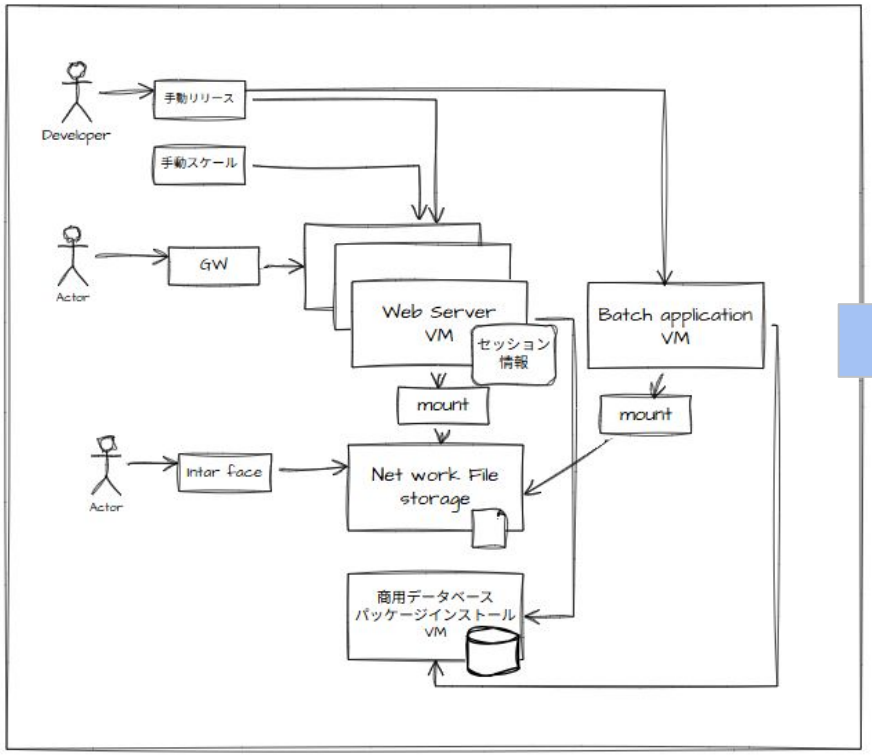
- 既存システムの課題と私たちが何をしたいかを共有した。準備も大切。
- 2 日間の建設的なディスカッション！

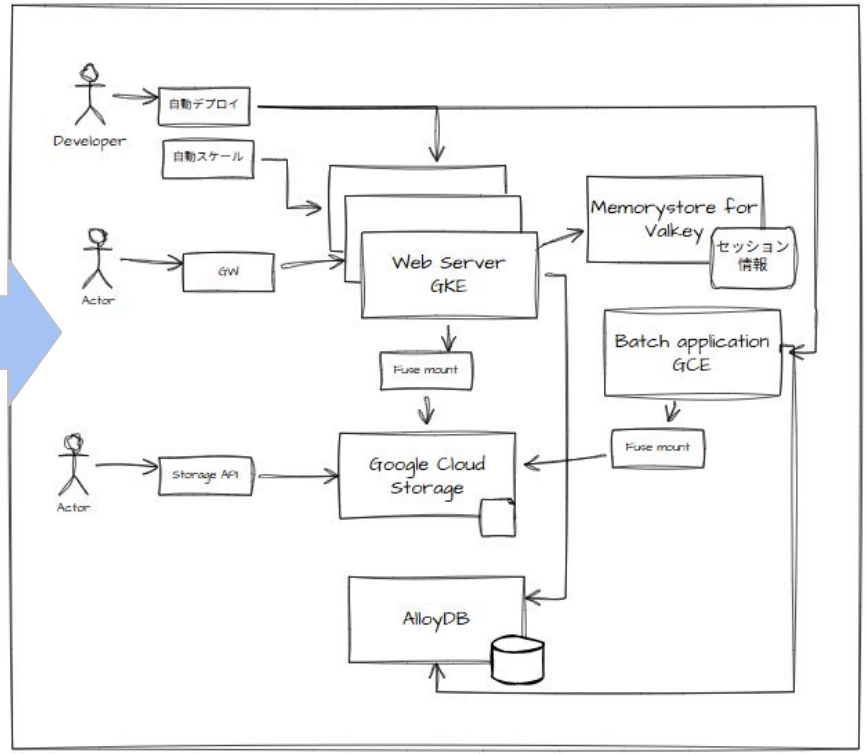




Proprietary

Tech Acceleration Program でアーキテクチャのグランドデザインを描く





Tech Acceleration Program でアーキテクチャのグランドデザインを描く



Proprietary

Tech Acceleration Program でアーキテクチャのグランドデザインを描く

アプリ・SRE メンバーの方向性が合意できた

みんなが同じ方向を向いた、これぞチーム！

Proprietary

アプリケーション改修・タスクのポイント

PostgreSQL（AlloyDB）対応コーディング　Gemini / Claude 活用

DAO・ストアドプロシージャを、PostgreSQL 互換へ書き換え
（Web アプリケーション/ バッチ アプリケーションからのDB 接続処理3,400 本）

セッション情報のステートレス化対応

セッション情報をWeb サーバ上のメモリからMemorystore for Valkey へ追い出す

データ マイグレーション

DB のデータをPostgreSQL (AlloyDB) へ完全移行
ファイル ストレージをCloud Storage へ移行

CI/CD パイプラインの構築

自動デプロイと無停止リリース

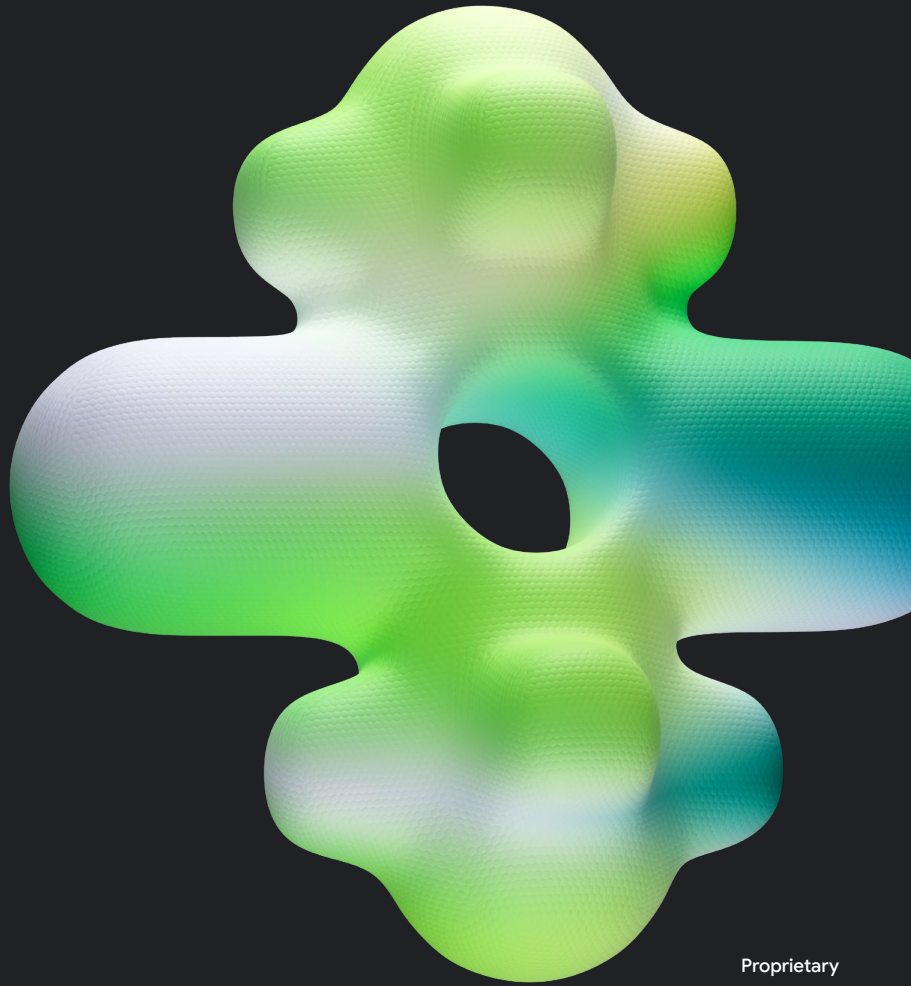
マイグレーションにフォーカス。追加機能改修、ロジック変更は行わない

Proprietary

04

モダナイゼーションによる改善結果

改善結果と振り返り



Proprietary

システムが抱えていた成長の限界
→ モダナイゼーションによって改善！



スケーラビリティ課題



GKE＋Valkey オートスケール

手動増設作業 6 時間→ 0 分
※ピーク時の手動 VM 増設から HPA が CPU/ メモリ利用率をもとに自動で POD を増減



手動作業の限界

Github Actions

CI/CDによる自動デプロイ

夜間サービス停止 6 時間→0 分
※アプリケーション レイヤの変更は無停止リリースが可能に



コスト

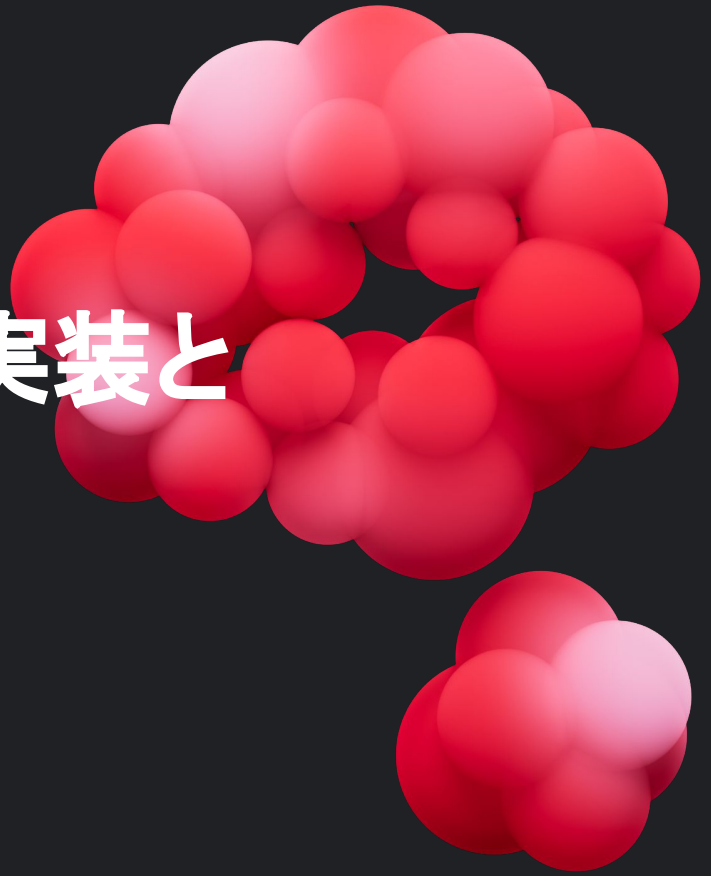


AlloyDB

運用コスト→ 50% 以上削減
※有償ライセンス費用の解消と、手動サーバ メンテナンス工数を削減

Proprietary

Google Cloud 上での実装と最適化の技術詳細



Proprietary



Allan Jone

株式会社マクロミル

Tech Lead

MACROMILL
a Macromill Group company

Proprietary

アジェンダ

05 **全体アーキテクチャ**
全体像・インフラプラットフォーム

06 **デプロイ自動化**
GitOps と Blue / Green リリース

07 **課題と成果**
乗り越えた壁と、得られたもの

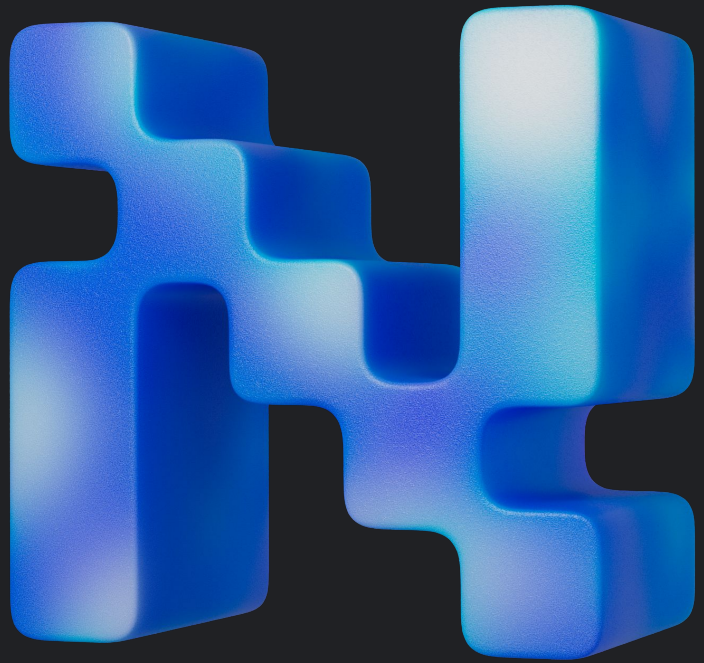


Proprietary

05

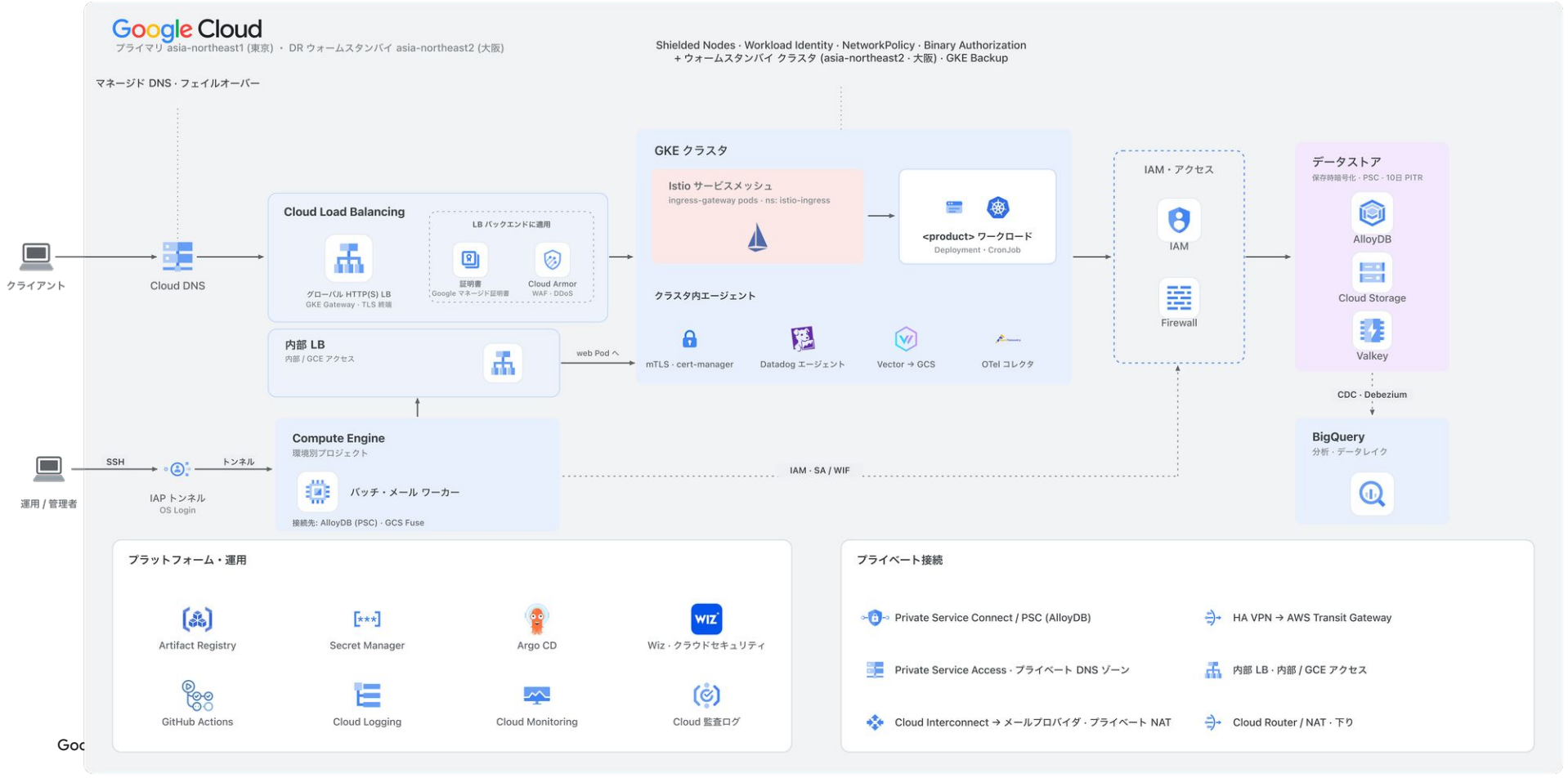
全体アーキテクチャ

全体像・インフラ プラットフォーム



Proprietary

アーキテクチャ全体像


Google Cloud
プライマリ asia-northeast1 (東京) ・ DR ウォームスタンバイ asia-northeast2 (大阪)
Shielded Nodes · Workload Identity · NetworkPolicy · Binary Authorization
+ ウォームスタンバイ クラスタ (asia-northeast2 · 大阪) · GKE Backup
マネージド DNS · フェイルオーバー
クライアント
Cloud DNS
Cloud Load Balancing
LB バックエンドに適用
グローバル HTTP(S) LB
GKE Gateway · TLS 終端
証明書
Google マネージド証明書
Cloud Armor
WAF · DDoS
内部 LB
内部 / GCE アクセス
web Pod へ
GKE クラスタ
Istio サービスメッシュ
ingress-gateway pods · ns: istio-ingress
<product> ワークロード
Deployment · CronJob
クラスタ内エージェント
mTLS · cert-manager
Datadog エージェント
Vector → GCS
OTel コレクタ
IAM・アクセス
IAM
Firewall
データストア
保存時暗号化 · PSC · 10日 PITR
AlloyDB
Cloud Storage
Valkey
CDC · Debezium
BigQuery
分析 · データレイク
運用 / 管理者
SSH
トンネル
IAP トンネル
OS Login
Compute Engine
環境別プロジェクト
バッチ・メール ワーカー
接続先: AlloyDB (PSC) · GCS Fuse
IAM · SA / WIF
プラットフォーム・運用
Artifact Registry
Secret Manager
Argo CD
Wiz · クラウドセキュリティ
GitHub Actions
Cloud Logging
Cloud Monitoring
Cloud 監査ログ
プライベート接続
Private Service Connect / PSC (AlloyDB)
HA VPN → AWS Transit Gateway
Private Service Access · プライベート DNS ゾーン
内部 LB · 内部 / GCE アクセス
Cloud Interconnect → メールプロバイダ · プライベート NAT
Cloud Router / NAT · 下り

インフラ プラットフォーム：IaC モノレポ

複数プロダクトを集約

プロダクト × 環境（dev / stg / prd / ops）で一元管理

再利用モジュール

共通 Terraform モジュールで Cloud Storage・Valkey・VM を標準化

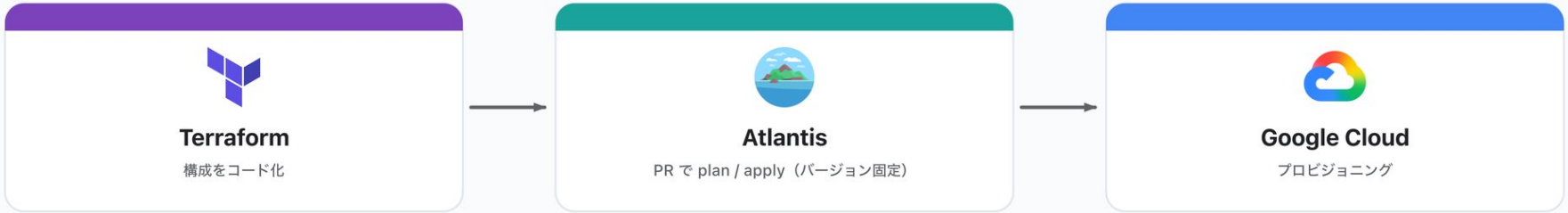
Atlantis GitOps

PR で plan / apply、並列実行、バージョン固定で再現性を確保

中央ガバナンス

組織・IAM・課金予算・タグを一元管理

インフラ（IaC）トラック：Terraform + Atlantis



Batch は Lift & Shift



無理にコンテナ化しない割り切り

- バッチ処理は Compute Engine へそのまま移設
- GCS バケットを gcsfuse でマウントし、ファイル連携を維持
- アクセス制御は OS Login ＋ IAM（YAML で宣言的に管理）
- Lift & Shift でも CI/CD を適用：GitHub Actions ＋ Ansible で自動ビルド・デプロイ
- ファイル I/O が重いバッチは fuse パス向けにチューニングして性能を確保

「すべてをモダナイズしない」勇気。期日内の移行を最優先に判断。

Proprietary

ファイル連携：gcsfuse で Cloud Storage を POSIX マウント



GKE アプリ（CSI ドライバー）

- GCS バケットを CSI ドライバーで Pod にマウント
- キャッシュやファイルモードはワークロードに合わせて調整
- Istio サイドカーと共存して動作
- 静的コンテンツの反映速度に合わせてキャッシュを調整
- レガシー FTP を内部 Pod 化し、gcsfuse 連携で提供



Batch VM（ネイティブ gcsfuse）

- Compute Engine に gcsfuse を導入し、起動時に常設マウント
- マウント先は旧基盤のパス構成を踏襲
- 既存のファイル I/O をコード変更なしで継続

アプリを書き換えず、Cloud Storage を「ファイル共有」として扱えるように。

Proprietary

06

デプロイ自動化

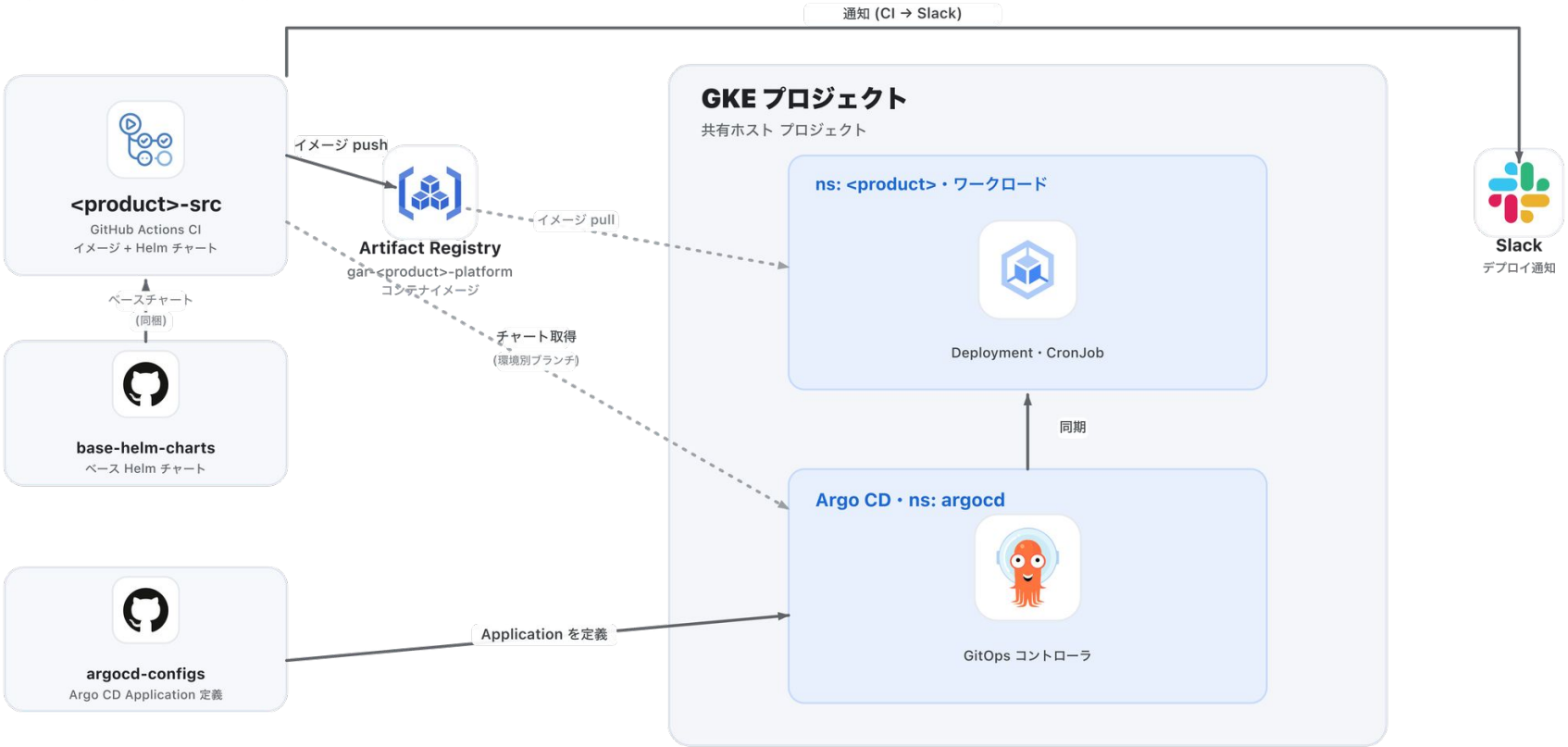
GitOps と Blue / Green リリース



Proprietary

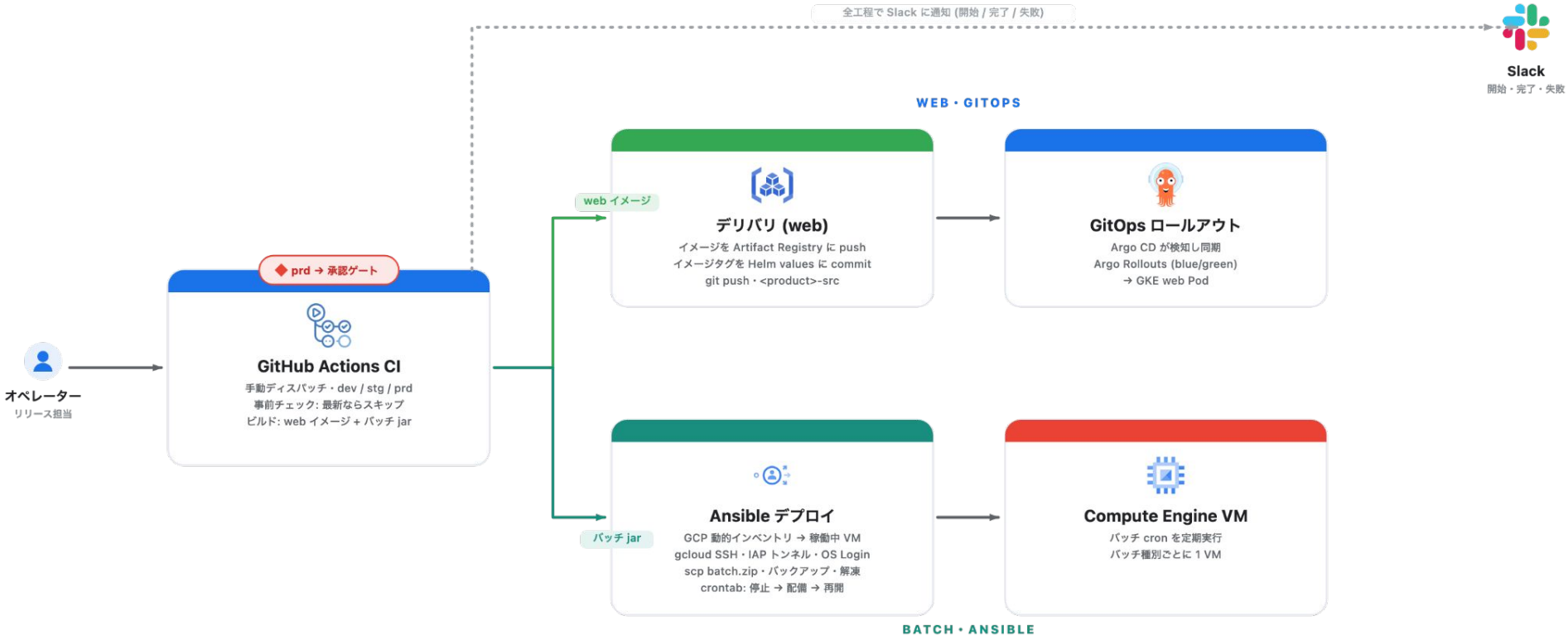
GitOps デプロイフロー

Argo CD・git-path・dev・stg・prd

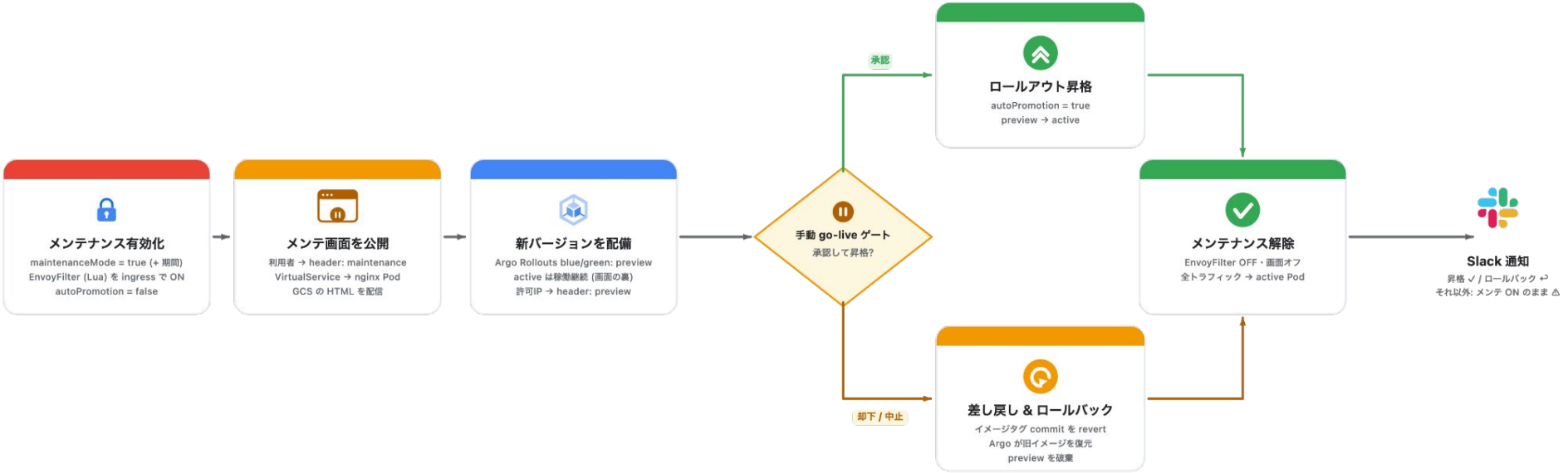


Proprietary

ビルド & デプロイ


全工程で Slack に通知 (開始 / 完了 / 失敗)
Slack
開始・完了・失敗
WEB・GITOPS
web イメージ
デリバリ (web)
イメージを Artifact Registry に push
イメージタグを Helm values に commit
git push・<product>-src
GitOps ロールアウト
Argo CD が検知し同期
Argo Rollouts (blue/green)
→ GKE web Pod
prd → 承認ゲート
GitHub Actions CI
手動ディスパッチ・dev / stg / prd
事前チェック: 最新ならスキップ
ビルド: web イメージ + バッチ jar
オペレーター
リリース担当
バッチ jar
Ansible デプロイ
GCP 動的インベントリ → 稼働中 VM
gcloud SSH・IAP トンネル・OS Login
scp batch.zip・バックアップ・解凍
crontab: 停止 → 配備 → 再開
Compute Engine VM
バッチ cron を定期実行
バッチ種別ごとに 1 VM
BATCH・ANSIBLE
任意の手動ゲート — 必要時に実行、GitHub Environment 承認で一時停止
DDL / DML
ストアドプロシージャ → AlloyDB
メール / 静的 → Cloud Storage
メンテナンスモード (blue/green)

メンテナンス モード



メンテナンスウィンドウ ー トラフィックルーティング

利用者 → メンテ画面 · 許可IP → preview (新バージョン)

内部の仕組み ー Istio

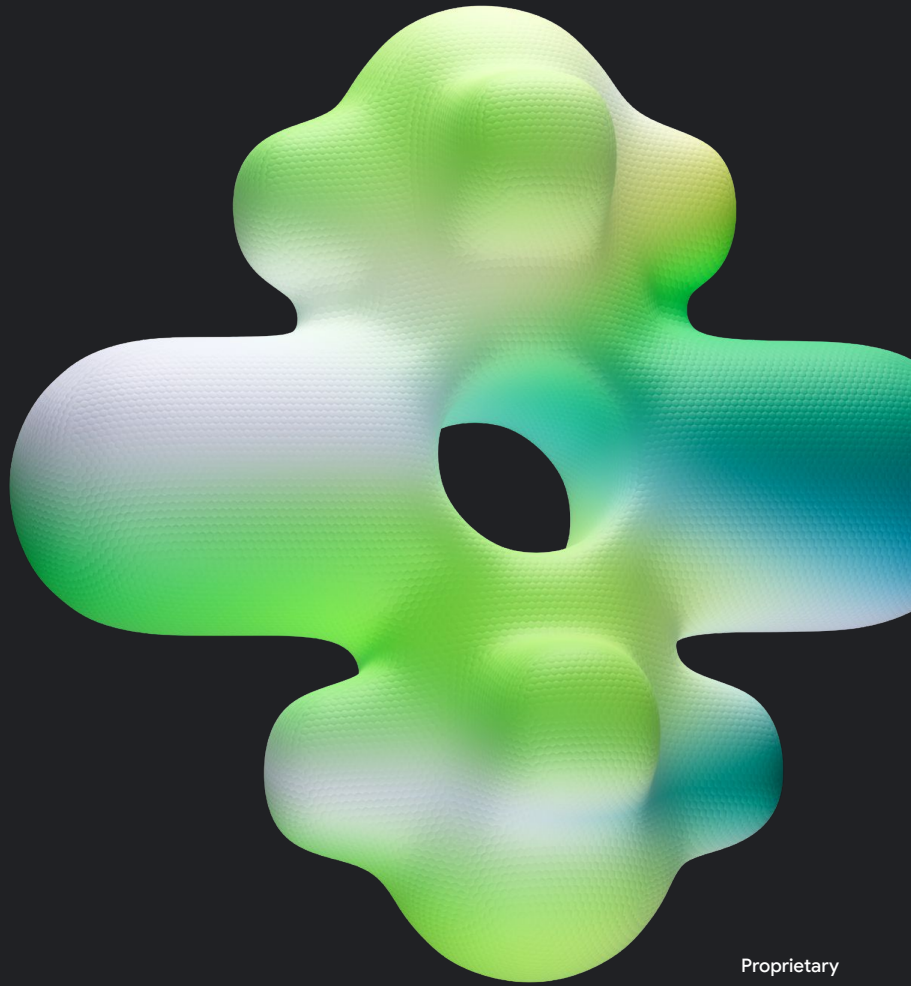
- Istio ingress ゲートウェイの EnvoyFilter (Lua) がクライアント IP を allowedIpRanges と照合 → ヘッダ x-rollout-mode を付与
- VirtualService がヘッダで振り分け: maintenance → nginx Pod (GCS HTML) · preview → 新 Pod
- 内部LB セレクタ → role: preview により Compute Engine バッチ VM も新バージョンに到達

Proprietary

07

課題と成果

乗り越えた壁と、得られたもの



Proprietary

構築の壁 ― 直面した課題と乗り越え方



レガシー RDB からの移行

型・SQL 方言の差異を吸収し、AlloyDB（PostgreSQL）へ移植



繁忙期ピーク 1,500 TPS の性能担保

3 月ピーク相当の負荷を目標に、反復的なチューニングで到達



約 4TB のファイル移行（旧クラウド → Cloud Storage）

Storage Transfer Service で一括 / 差分同期、rsync でマルチバイトファイル名を安全に移行



既存クラウド ↔ Google Cloud のハイブリッド運用

既存クラウドで稼働するサービスは VPN 経由で AlloyDB に接続



インフラを IaC でコード化

Terraform / Atlantis で構成をコード化し、再現性を確保（ 99.9% IaC）



デプロイ起因のダウンタイムをゼロに

Blue / Green リリースで無停止デプロイを実現

うまくいった話だけでなく、超えた壁こそが学び。

Proprietary

まとめ

1 **適材適所** モダナイズ（Web）と Lift & Shift（Batch）を使い分けた

2 **自動化** GitOps で再現性と安全性を獲得し、Toil を削減

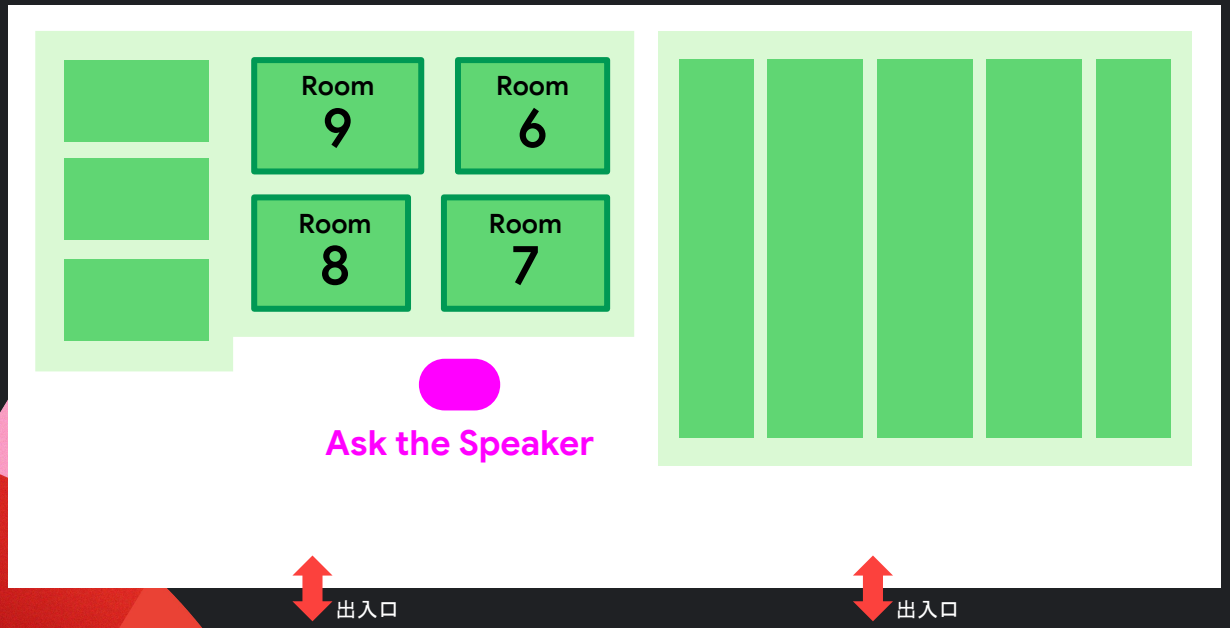
3 **リスペクト** 既存システムと新しい技術スタック、双方への歩み寄り

止めない移行と自動化で、レガシー刷新を現実的なスピードで前へ。

Proprietary

Ask the Speaker にぜひお越しください（会場は 4F）

セッションに関する質問にスピーカーが直接お答えします！



Thank you



Proprietary

Thank you



Google Cloud Next Tokyo