Amazon ECS で実現する プラットフォームエンジニアリング

Kenta Goto

Amazon Web Services Japan G.K.
Solutions Architect

自己紹介

- 名前
  - Kenta Goto（ごとけん）
- 所属
  - ISV/SaaS Solutions Architect
- 職歴
  - SIer → Web系（インフラエンジニア）
- 趣味
  - サウナ

@kennygt51

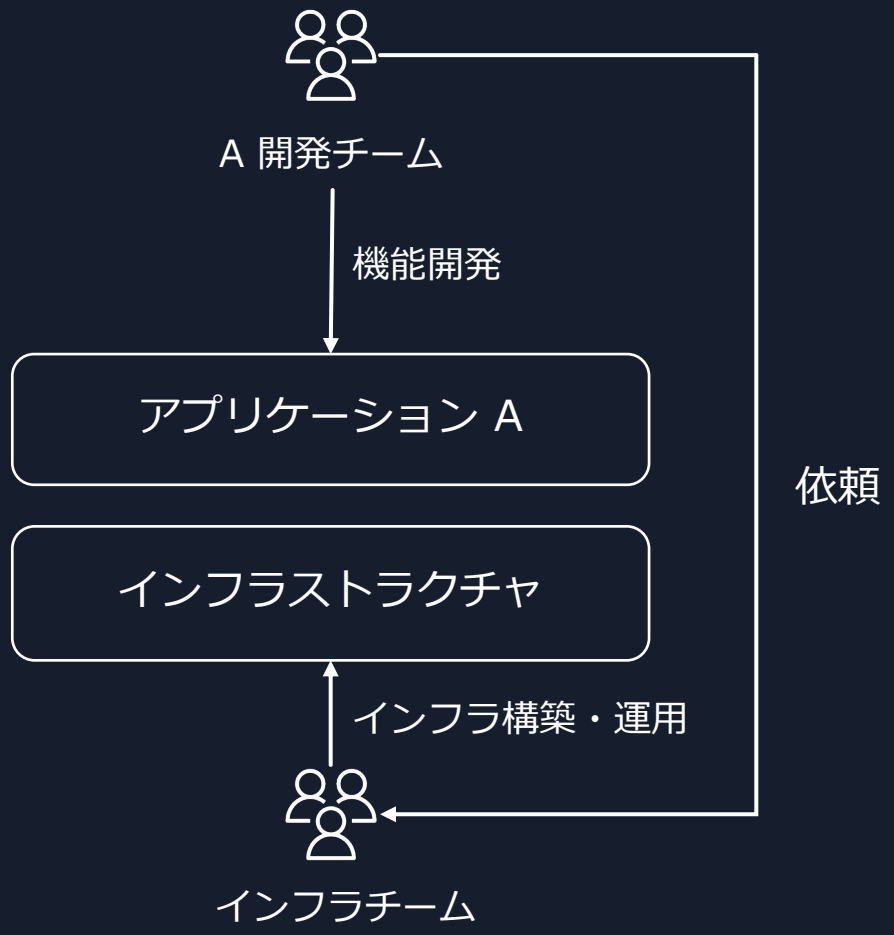
Agenda

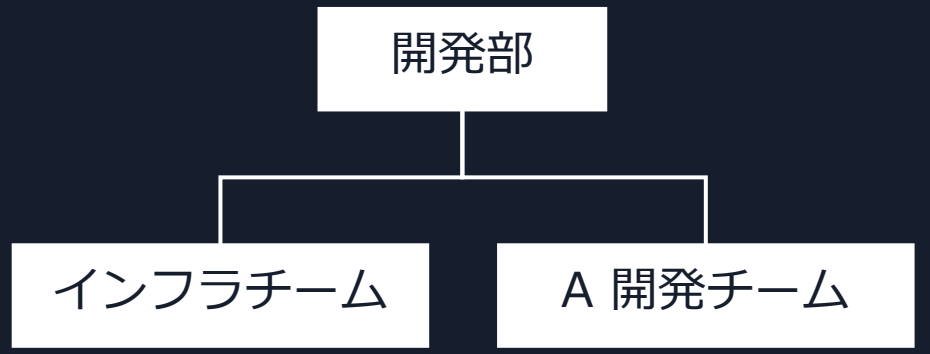
- 事業の拡大と開発組織の変遷
- プラットフォームエンジニアリングの始め方
- Amazon ECS で実現するプラットフォームエンジニアリング
- まとめ

Agenda

- 事業の拡大と開発組織の変遷
- プラットフォームエンジニアリングの始め方
- Amazon ECS で実現するプラットフォームエンジニアリング
- まとめ

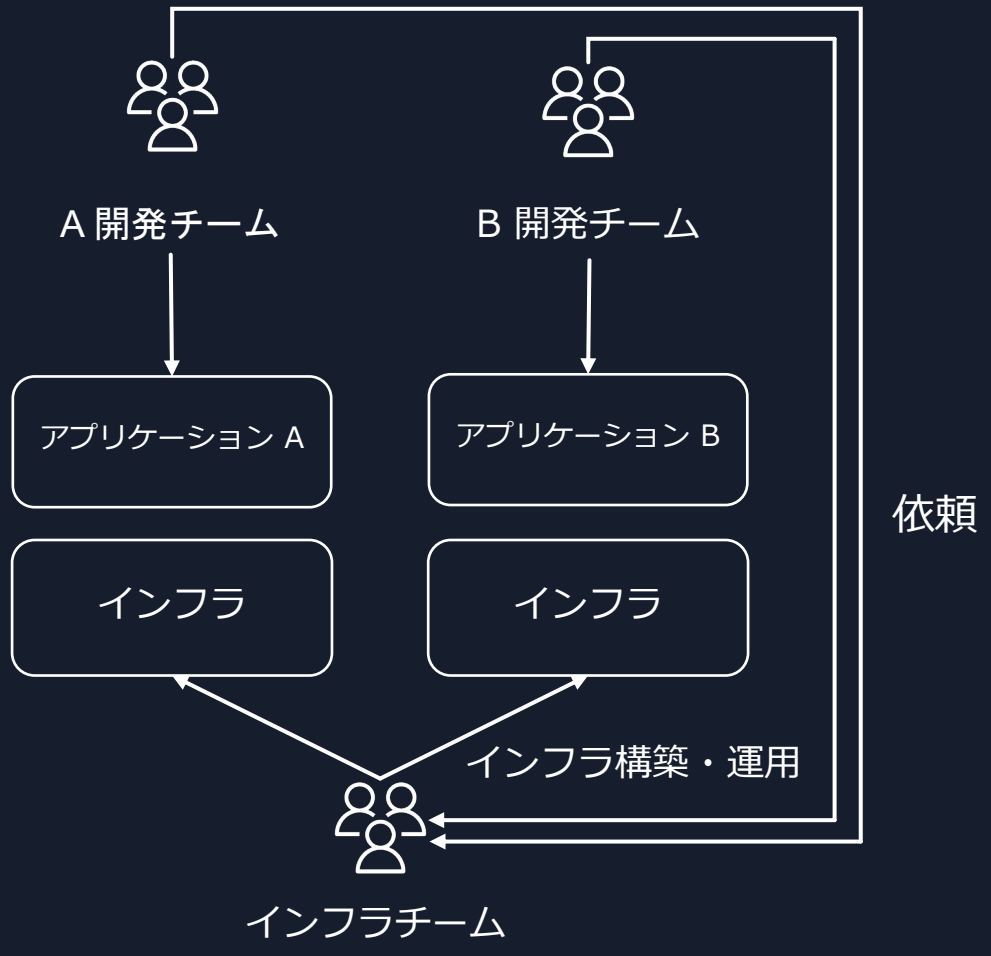
単一のプロダクトにフォーカス

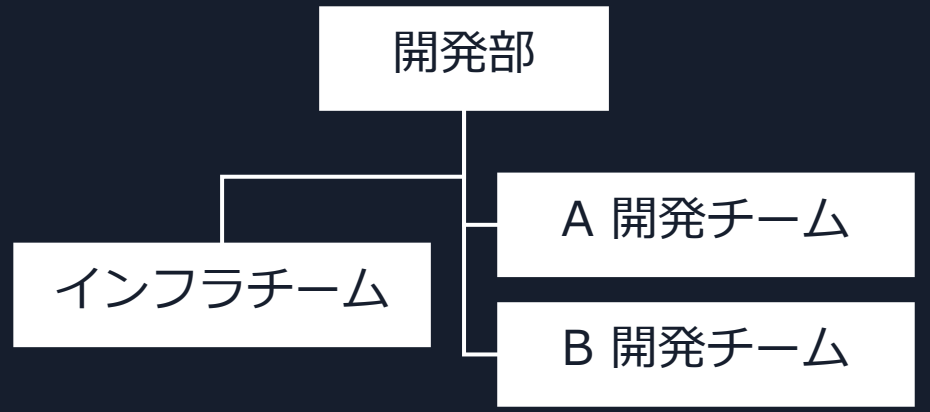




- シンプルな開発体制
- インフラの変更が必要な場合は、機能開発を担当するチームから、インフラを担当するチームへの依頼ベースでの対応

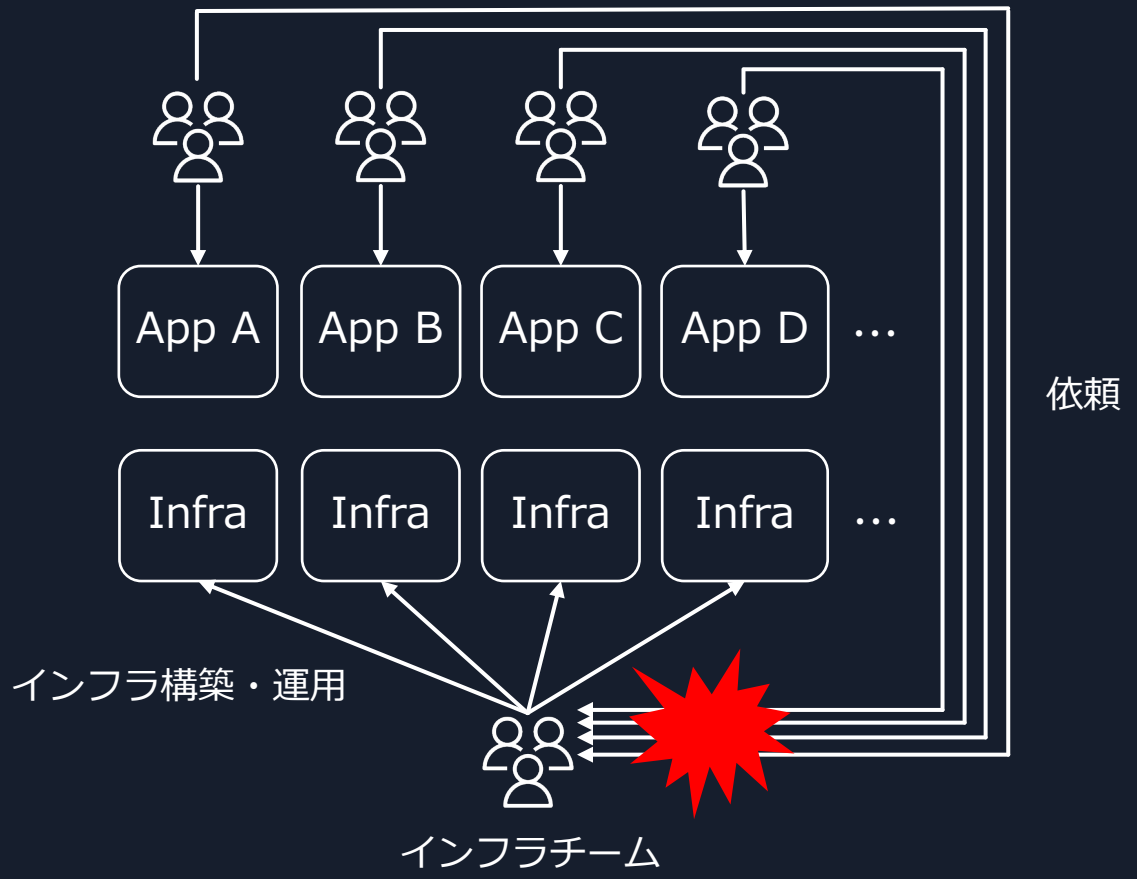
事業の拡大に伴い、プロダクト数が増加

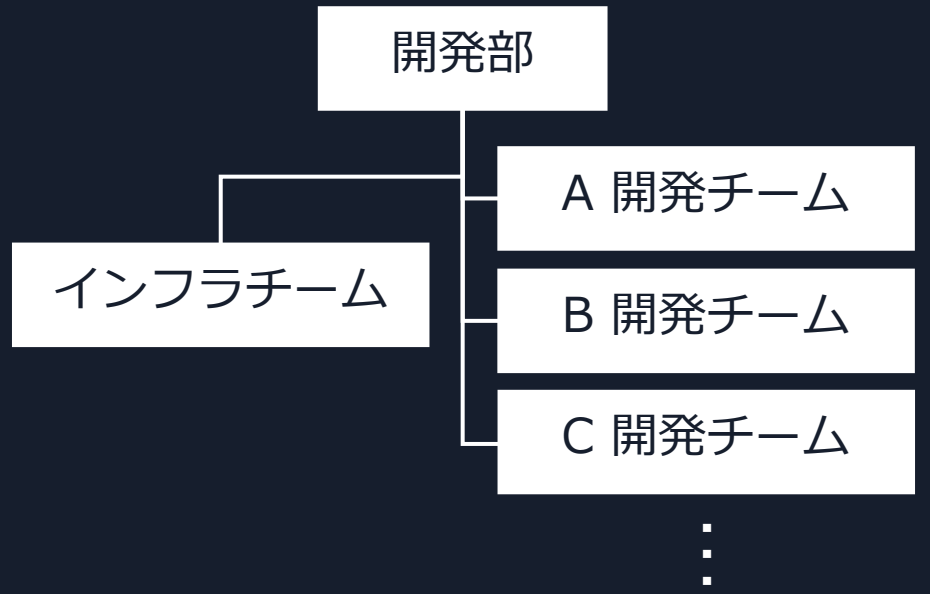




- 事業が拡大し、プロダクト数が増加すると、開発組織も拡大していく
- インフラチームへの依頼の数も増加
- プロダクト数が少なければよいが、、

インフラチームがボトルネックに





- さらなる事業の成長に伴い、プロダクトの数が増え、インフラを担当するチームへの依頼数が増加
- 作業負荷の急増によって、インフラチームがボトルネックになり、開発生産性の低下をもたらす

スケールしていく開発組織が抱える課題

- インフラチームへの負荷集中
  - 開発チームの拡大に追いつくように、インフラチームを拡大することは難しい
    - 採用の難しさにも起因
  - 結果として、インフラチームに負荷が集中しがち
  - 本来取り組むべきことにリソースを避けない状況が生まれる

- 開発チームの開発生産性の低下
  - 機能開発に必要なインフラ側の対応がスタックすることで、開発生産性の低下に繋がる
  - 結果として、ビジネス価値提供のスピードが失われる

スケールしていく開発組織が抱える課題

- インフラチームへの負荷集中
  - 開発チームの拡大に伴い、インフラチームが抱える負荷も増大
    - 採用の難しさにも起因
  - 結果として、インフラチームに負荷が集中しがち
  - 本来取り組むべきことにリソースを避けない状況が生まれる

- 開発チームの開発生産性の低下
  - 機能開発に必要な対応が遅れることで、開発生産性の低下に繋がる
  - 結果として、ビジネス価値提供のスピード感が失われる

スケールしていく開発組織が抱える課題と向き合うための

「プラットフォームエンジニアリング」

というアプローチ

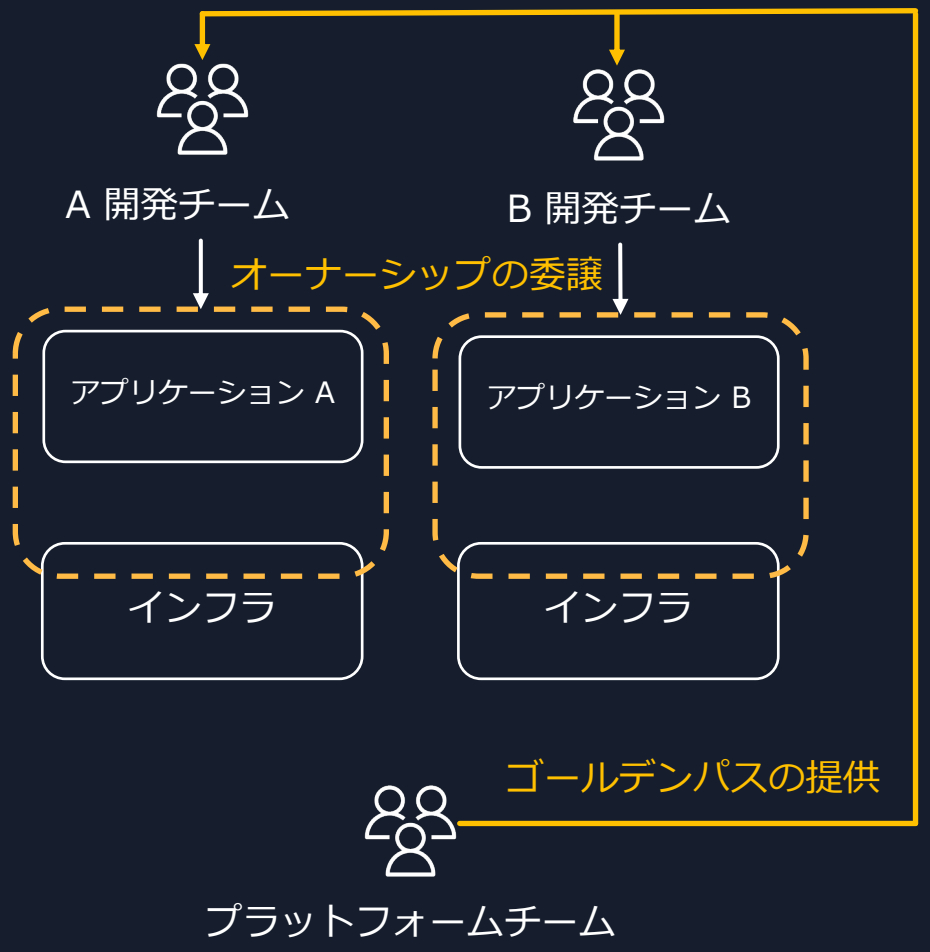
プラットフォームエンジニアリングとは

- Gartner (※1)
  - プラットフォーム・エンジニアリングは、再利用可能なツールとセルフサービス機能を実装し、インフラストラクチャ・オペレーションを自動化することで、開発者のエクスペリエンスと生産性を向上させる
- platformengineering.org (※2)
  - Platform engineering is the discipline of designing and building toolchains and workflows that enable self-service capabilities for software engineering organizations in the cloud-native era. Platform engineers provide an integrated product most often referred to as an “Internal Developer Platform” covering the operational necessities of the entire lifecycle of an application.

開発チームの開発者体験と生産性を向上させることを目的として、
プラットフォームチームとして、
セルフサービスで利用できるプロダクトを設計・構築する分野

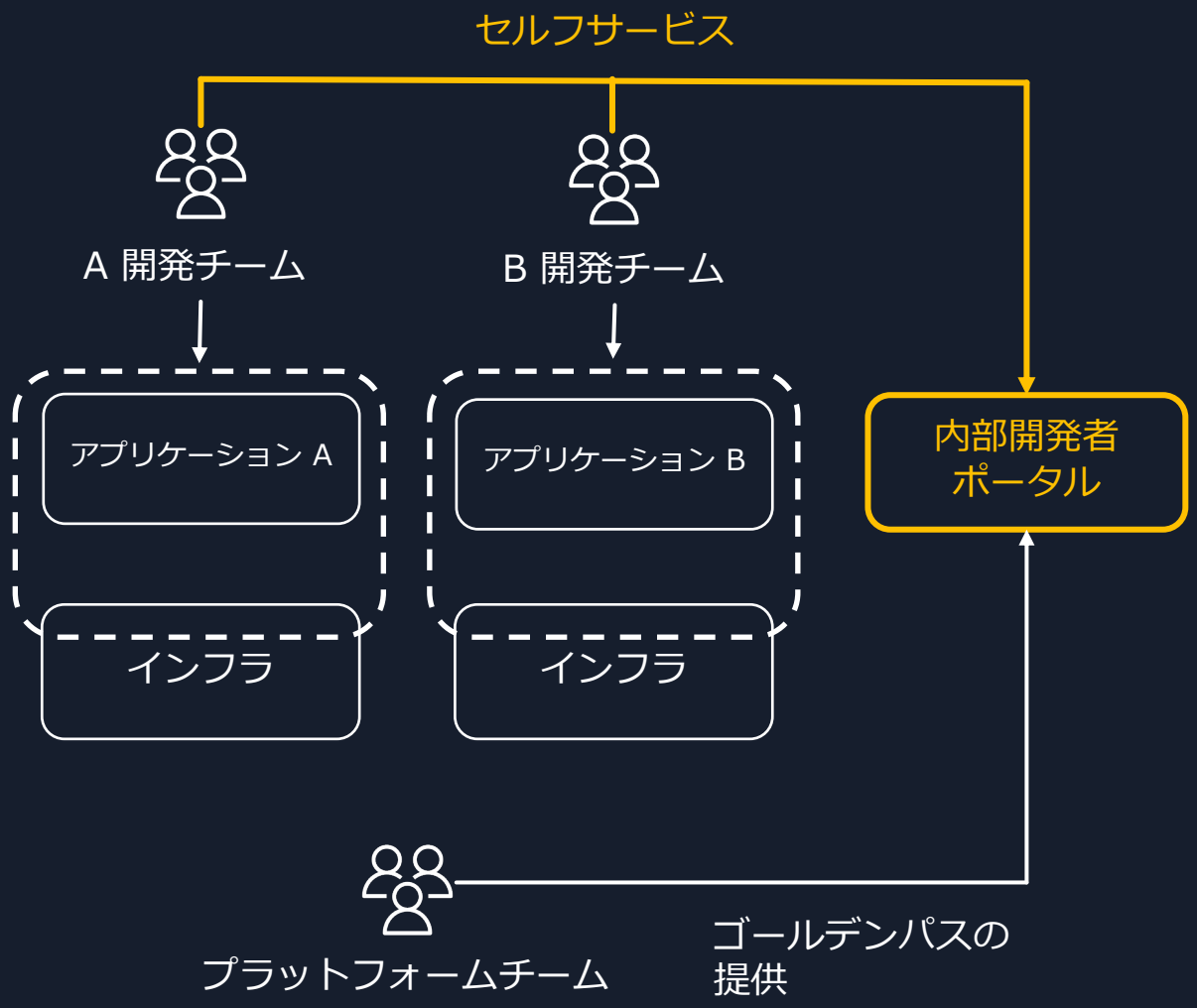
※1 https://www.gartner.co.jp/ja/articles/what-is-platform-engineering
※2 https://platformengineering.org/blog/what-is-platform-engineering

ゴールデンパス



- インフラチームから開発チームに、基盤となるインフラのオーナーシップを委譲
- 同時に、プラットフォームチームによるゴールデンパス（IaC テンプレートやオンボーディングなど）の整備によって、開発チームの負荷を軽減

セルフサービス



- 開発チームとインフラチームのやり取りを “X-as-a-Service” で実現する「内部開発者ポータル」を導入
- 開発チームが自律的に（セルフサービスで）設計・開発をおこなえるようになり、開発生産性の向上に繋がる

Agenda

- 事業の拡大と開発組織の変遷
- プラットフォームエンジニアリングの始め方
- Amazon ECS で実現するプラットフォームエンジニアリング
- まとめ

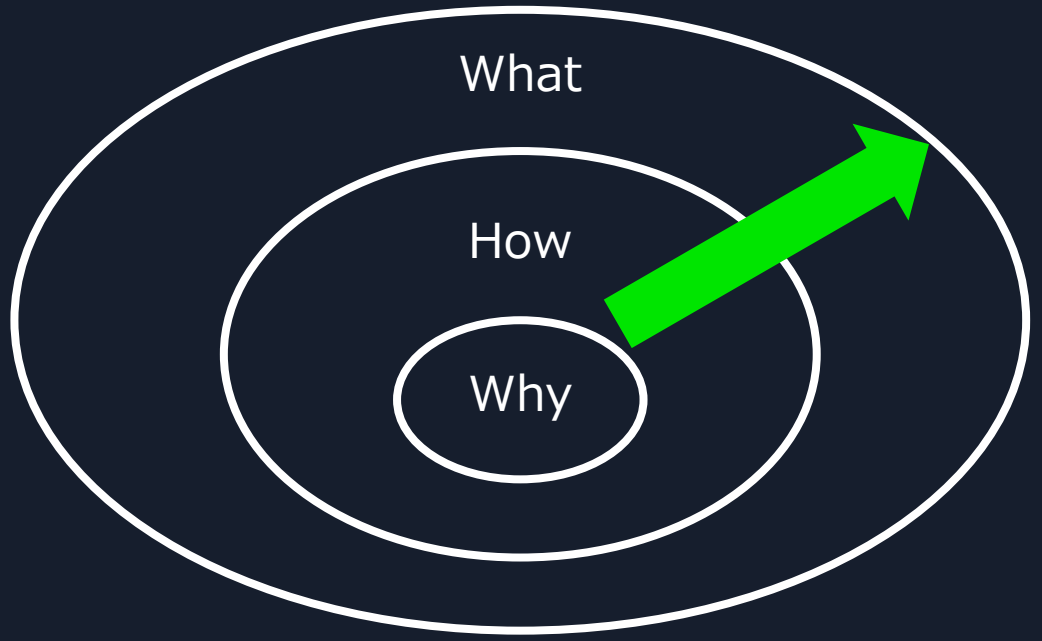
プラットフォームエンジニアリングの始め方

1. プロジェクトの目線合わせ
2. 経営層との合意
3. チームを組成
4. 最小限のプラットフォームを開発
5. 他チームに徐々に展開
6. マイグレーション

プロジェクトの目線合わせ

ゴールデンサークル理論

人を動かす偉大な企業や人物というのは「ゴールデンサークル」というシンプルなパターンに基づいて行動している



https://simonsinek.com/golden-circle/

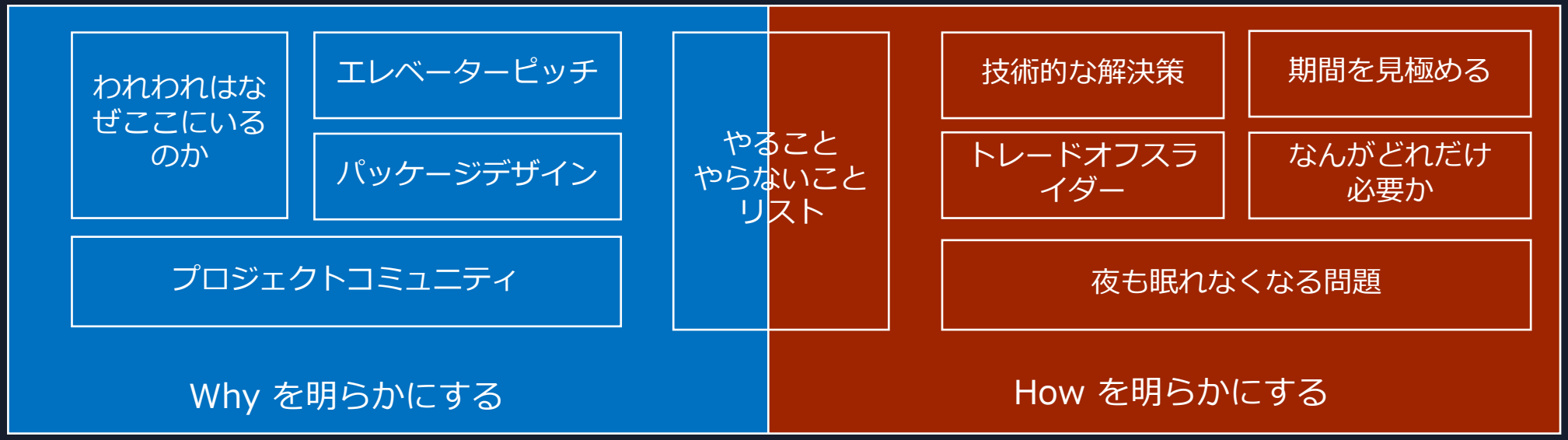
- ✓ Why：目的はなにか？
- ✓ How：どんな手段を取るか？
- ✓ What：どう行動するか？

プラットフォームの開発を始める前に、
「なぜプラットフォームを作るのか？」
を明文化する

プロジェクトの目線合わせ

インセプションデッキ

プロダクトが向かう方向を明らかにし、ステークホルダー間で共有するためのフレームワーク



プラットフォームの顧客は誰か、プラットフォームの価値は何か

経営層との合意

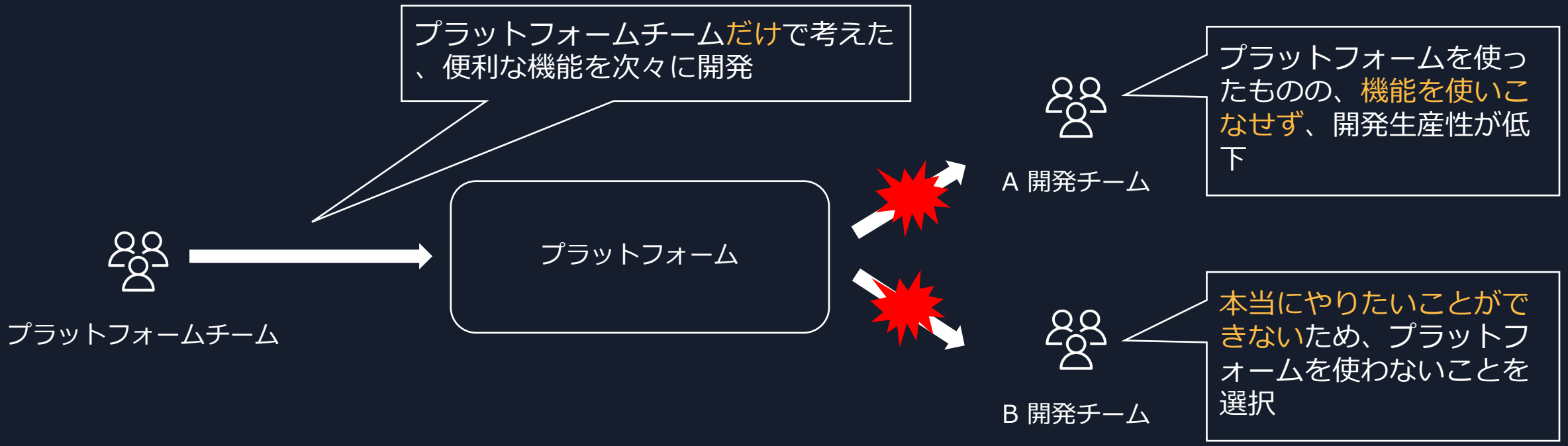
- “多くの経営者はITインフラをコストとして認識しており、プラットフォームに割り当てられるコストやリソースを抑制しようとする可能性がある”(※)
- “ビジネスへの直接的な影響と関係性を示して、プラットフォームチームがビジネスサイドへの戦略的パートナーであることを示す必要がある”(※)

- ➢ プラットフォームの開発が、自社ビジネスを伸ばしていくために必要であることを合意する
- ➢ 定量的・定性的にプラットフォームのパフォーマンスを測定する

※ https://tag-app-delivery.cncf.io/whitepapers/platforms/

チームを組成

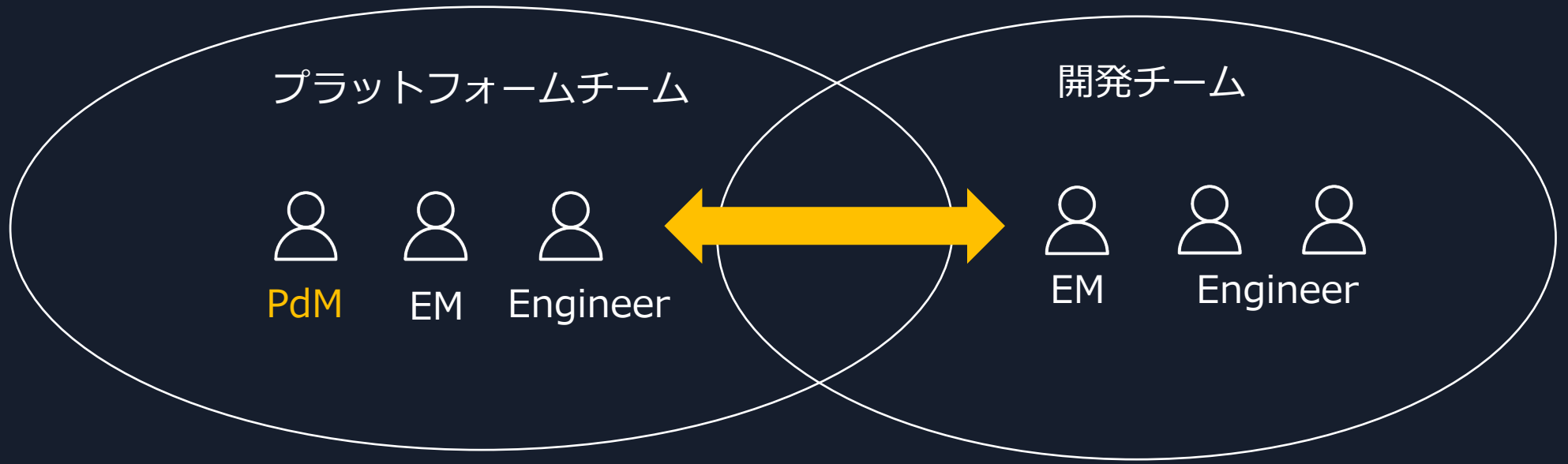
よくある失敗ケース



使われない機能が開発され、開発チームには使用されない
結果的に、使われないプラトットフォームになる

チームを組成

- プロジェクト開始時に開発チームを巻き込み、コラボレーションしながらプラットフォームの設計・構築に取り組む
- （理想的には）プラットフォームチームにプロダクトマネージャを配置する



最小限のプラットフォームを開発

TVP (Thinnest Viable Platform) を目指す (※1)

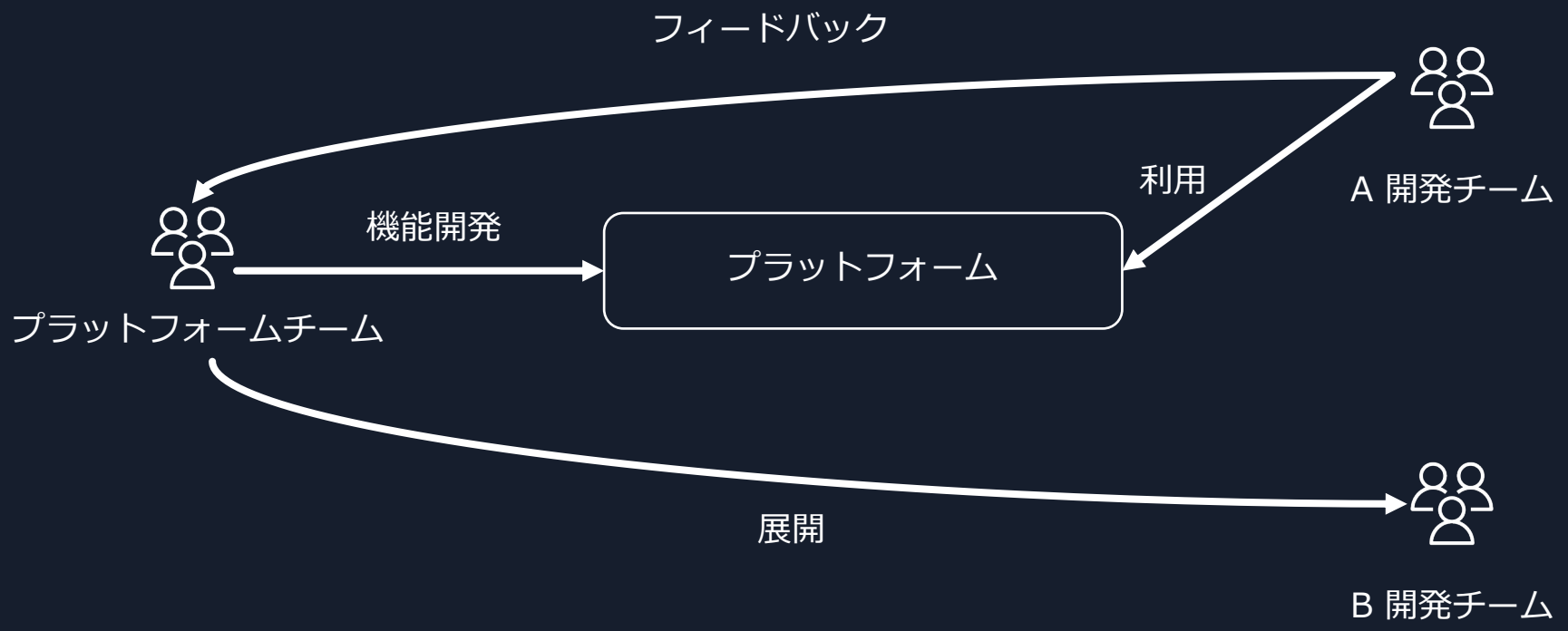
- その時「必要なもの」だけを作り、プラットフォームを小さく保つ
- 開発組織の拡大、複雑な課題の発生により「必要なもの」は変化する
- 最低限「必要なもの」を特定し、”薄い”プラットフォームを継続的に維持する
- 技術のエコシステムの進化に合わせて、プラットフォームは継続的に進化する

※1 https://teamtopologies.com/key-concepts-content/what-is-a-thinnest-viable-platform-tvp

最小限のプラットフォームを開発

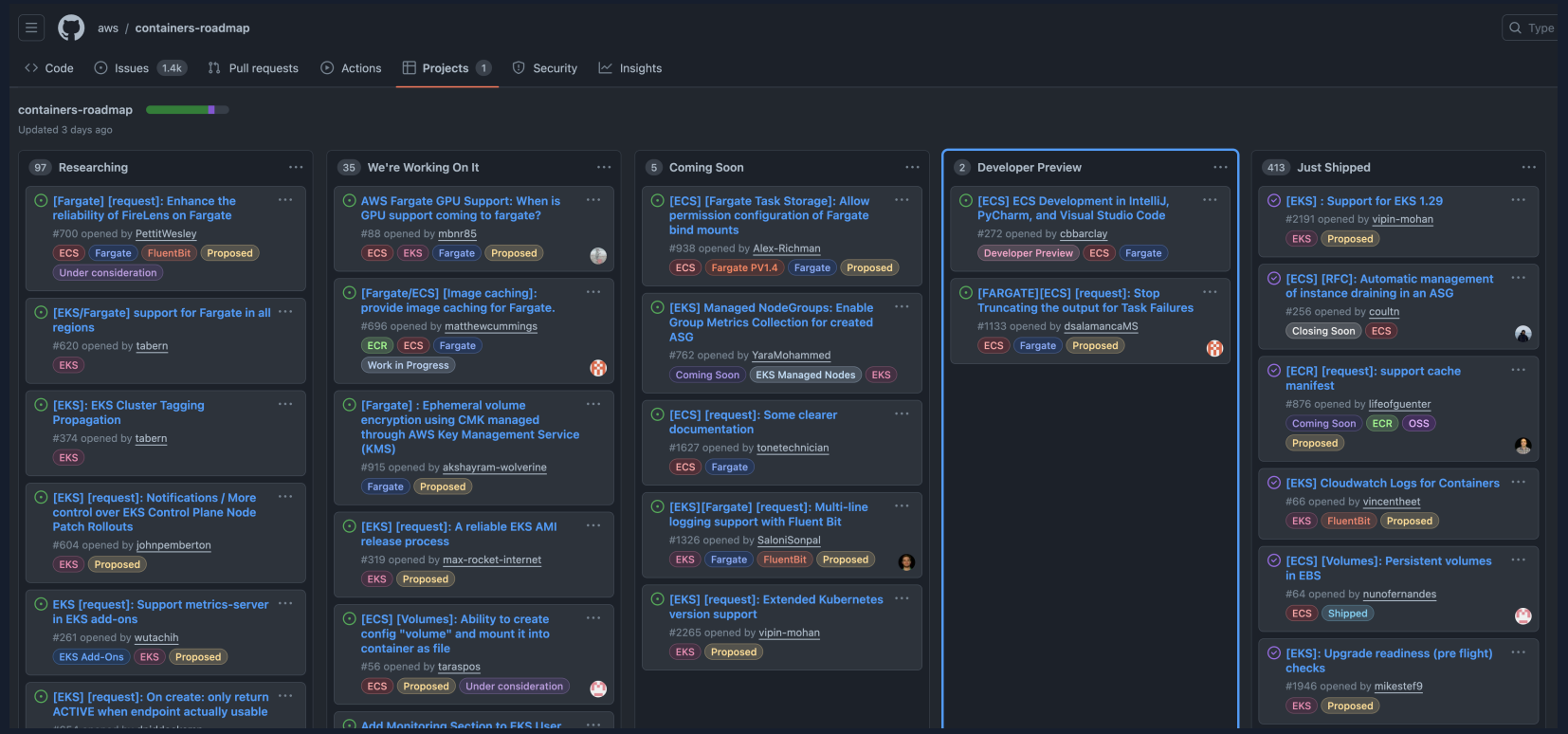
他のチームに展開

ユーザーからのフィードバックを元にプラットフォームの開発サイクルを回しながら、他のチームに展開していく



他のチームに展開

ユーザーに対して、プラットフォームのロードマップを公開



https://github.com/aws/containers-roadmap/projects/1

マイグレーション

- 現行システムをプラットフォームに移行する

- 一定のリソースを割く必要がある
  - マイグレーションの対象（スコープ）を決定
  - 中長期的な移行計画を策定

- プラットフォームチームだけで対応するのではなく、開発チームにも協力を依頼

Agenda

- 事業の拡大と開発組織の変遷
- プラットフォームエンジニアリングの始め方
- Amazon ECS で実現するプラットフォームエンジニアリング
- まとめ

Why から始める

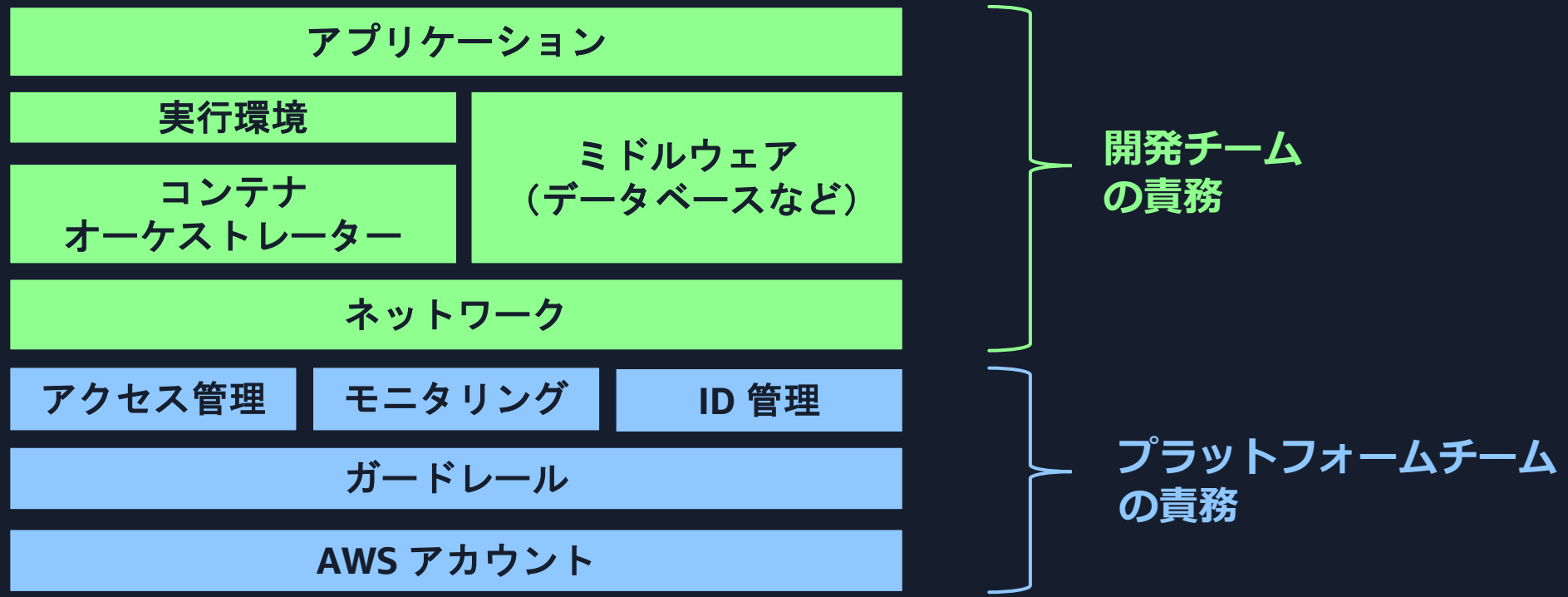
- **Why から始める**
  - プラットフォームの顧客は誰か、プラットフォームの価値は何か

- プロダクトが稼働している基盤・開発組織の特性・想定するべきプラットフォームとしてのスケーラビリティなど、様々な要素を考慮しどのようなプラットフォームを構築するのかを検討する

ゴールデンパスの実装

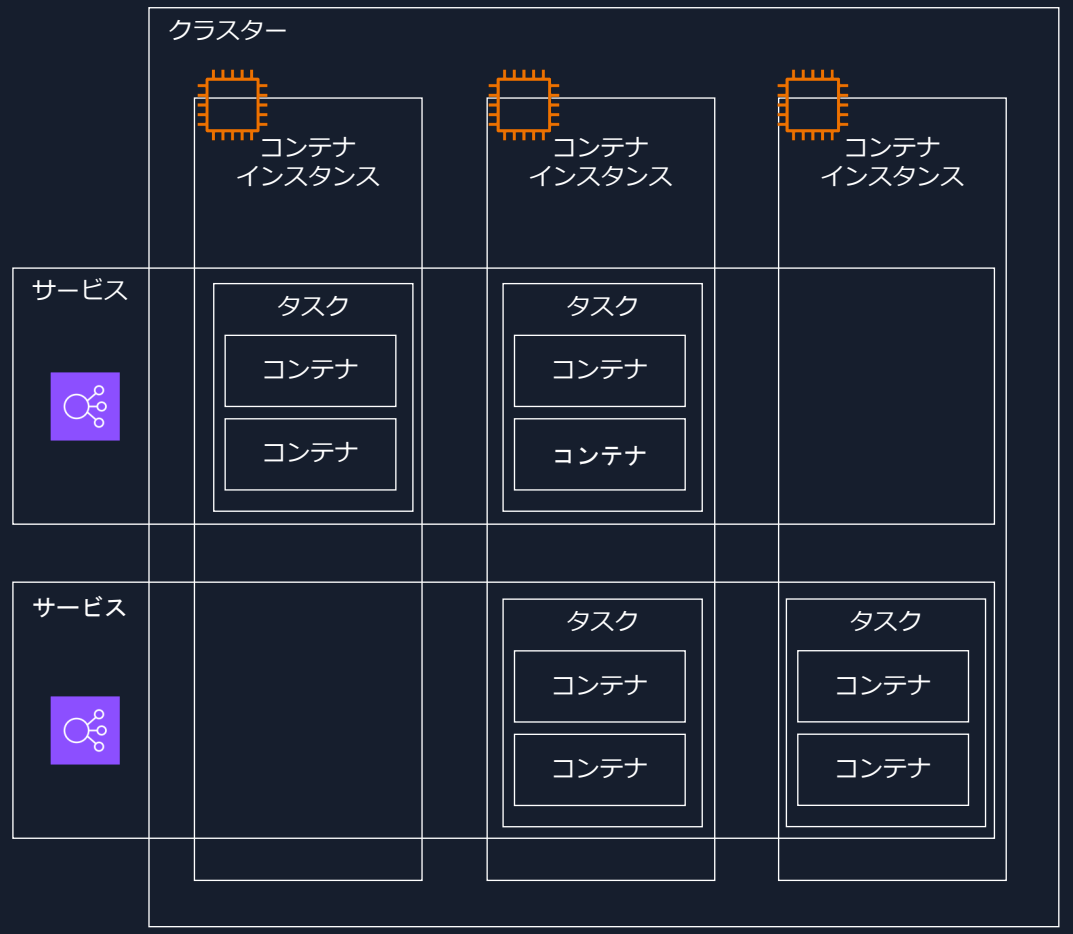
オーナーシップの定義

- 開発チームとプラットフォームチームの責務分担を定義し、必要に応じてオーナーシップを委譲
- **IaC テンプレートを実装する前に、オーナーシップを明確化しておく**



Amazon ECS を構成するコンポーネント

- 実行内容の定義
  - コンテナ　　：プログラムの実行単位
  - タスク　　：コンテナの集合体
  - タスク定義　：コンテナとタスクの定義
- 実行内容と実行リソースの紐づけ
  - サービス　　：複数タスクの稼働設定
- 実行リソース
  - インスタンス：タスク実行環境（EC2やFargate）
  - クラスター　：インスタンスの集合体



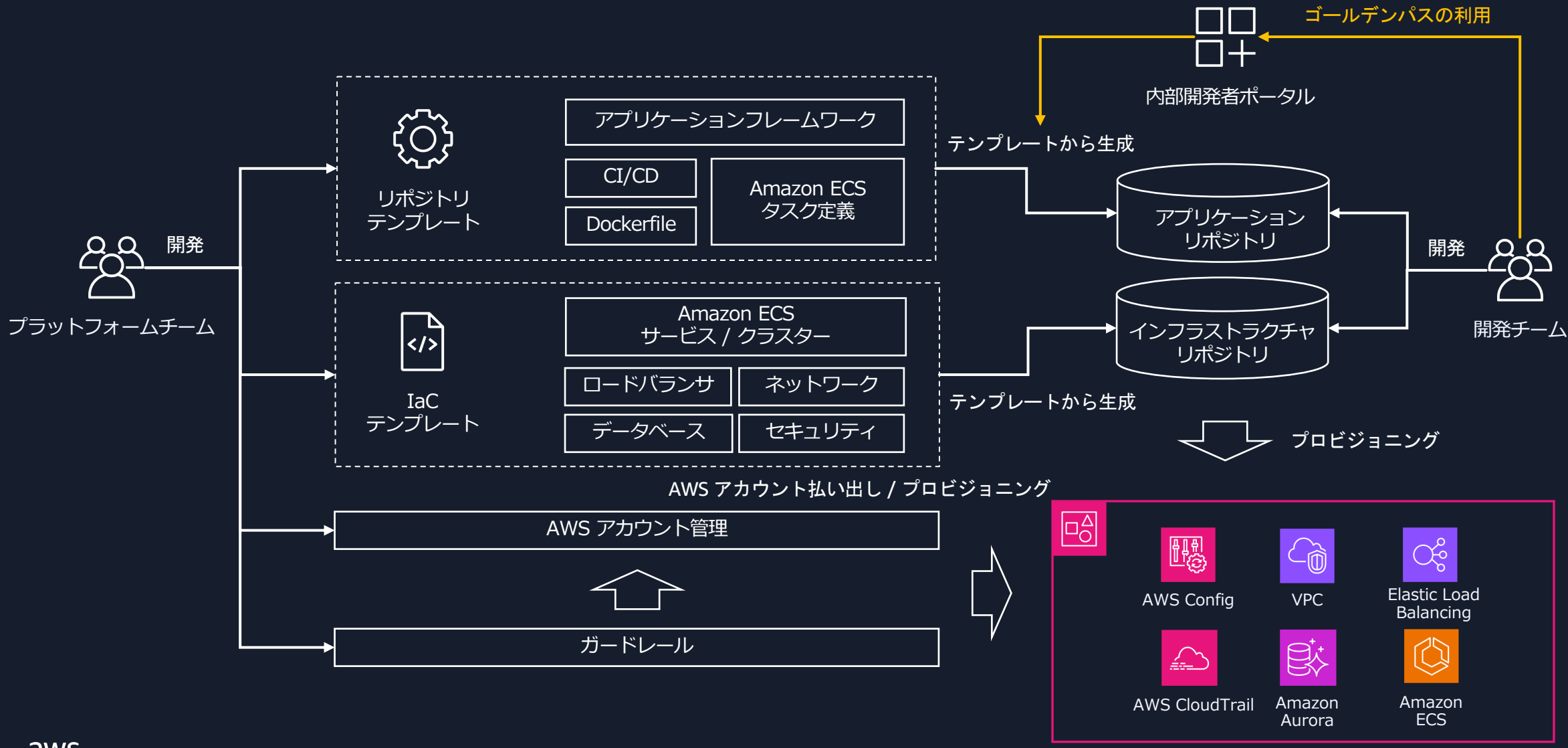
Amazon ECS を構成するコンポーネント

- 実行内容の定義
  - コンテナ　　：プログラムの実行単位
  - タスク　　：コンテナの集合体
  - タスク定義　：コンテナとタスクの定義
- 実行内容と実行リソースの紐づけ
  - サービス　　：複数タスクの稼働設定
- 実行リソース
  - インスタンス：タスク実行環境（EC2やFargate）
  - クラスター　：インスタンスの集合体

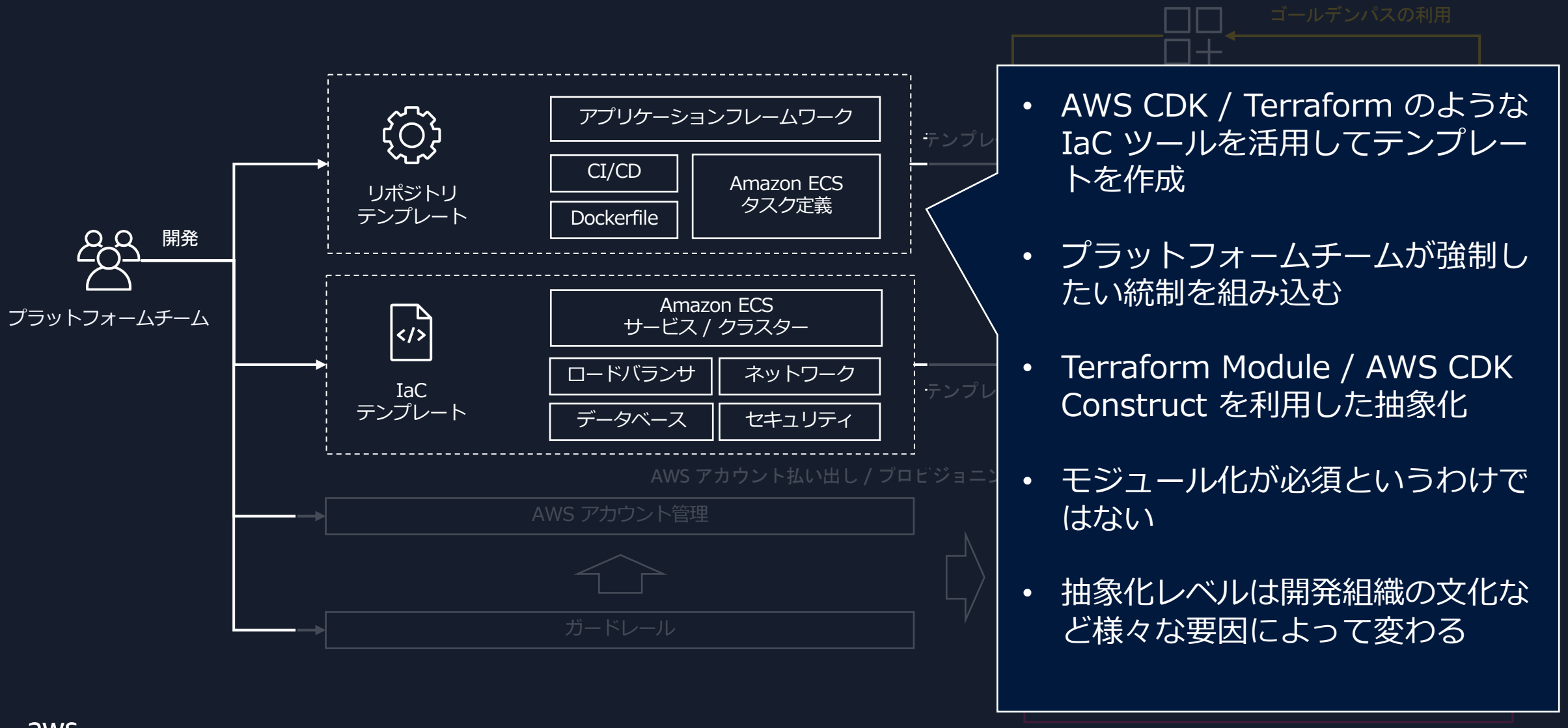
Amazon ECS を構成するコンポーネント

- 実行内容の定義
  - コンテナ　　：プログラムの実行単位
  - タスク　　：コンテナの集合体
  - タスク定義　：コンテナとタスクの定義
- 実行内容と実行リソースの紐づけ
  - サービス　　：複数タスクの稼働設定
- 実行リソース
  - インスタンス：タスク実行環境（EC2やFargate）
  - クラスター　：インスタンスの集合体

ゴールデンパスの実装



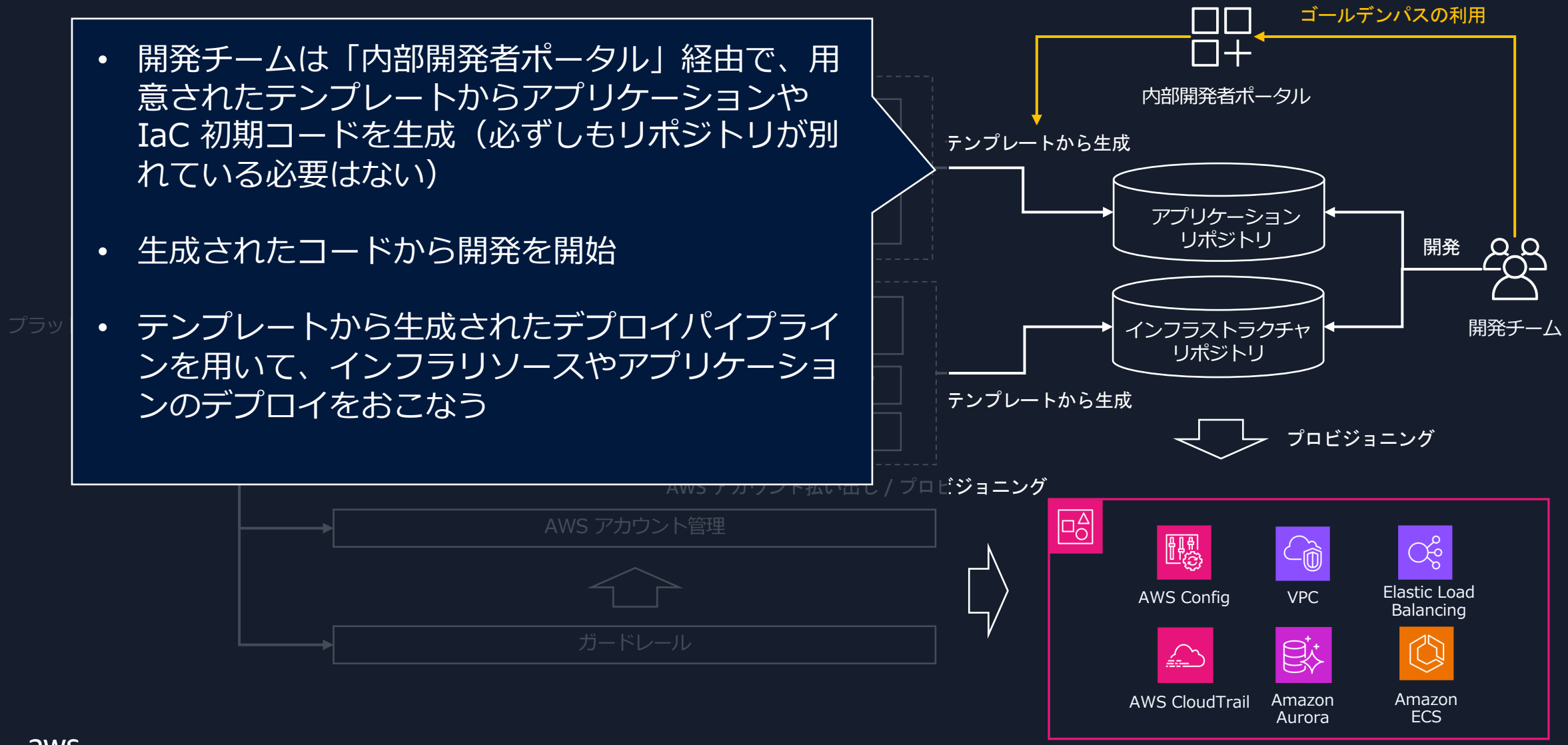
ゴールデンパスの実装



- AWS CDK / Terraform のような IaC ツールを活用してテンプレートを作成
- プラットフォームチームが強制したい統制を組み込む
- Terraform Module / AWS CDK Construct を利用した抽象化
- モジュール化が必須というわけではない
- 抽象化レベルは開発組織の文化など様々な要因によって変わる

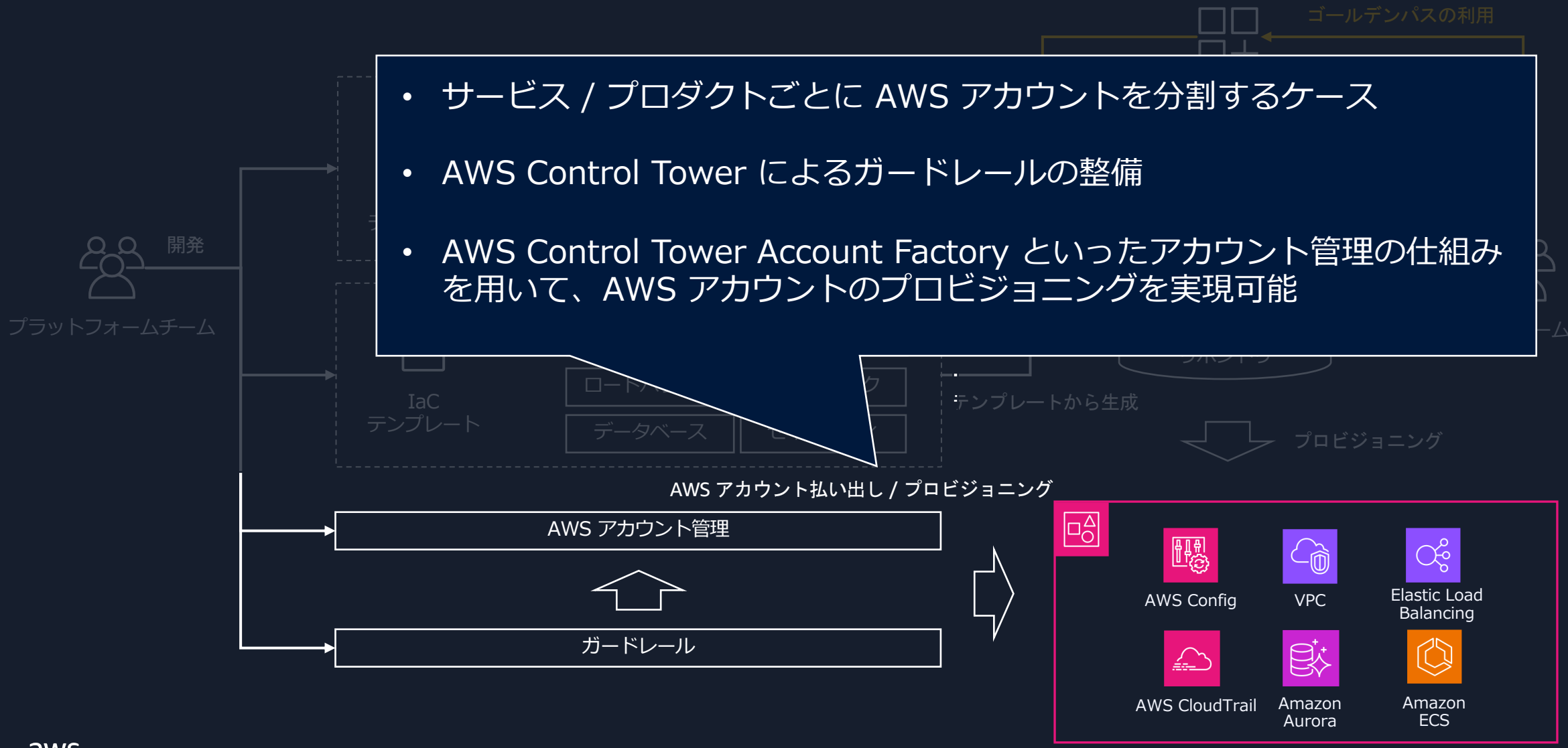
ゴールデンパスの実装

- 開発チームは「内部開発者ポータル」経由で、用意されたテンプレートからアプリケーションや IaC 初期コードを生成（必ずしもリポジトリが別れている必要はない）
- 生成されたコードから開発を開始
- テンプレートから生成されたデプロイパイプラインを用いて、インフラリソースやアプリケーションのデプロイをおこなう



ゴールデンパスの実装

- サービス / プロダクトごとに AWS アカウントを分割するケース
- AWS Control Tower によるガードレールの整備
- AWS Control Tower Account Factory といったアカウント管理の仕組みを用いて、AWS アカウントのプロビジョニングを実現可能



内部開発者ポータルの構築

内部開発者ポータルの構築

- TVP を意識した内部開発者ポータル
  - テンプレートの利用方法を記載した Wiki のページ

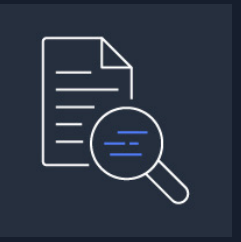
- セルフサービス化の促進、ユーザーの利用体験を高めるための選択肢
  - AWS Proton
  - AWS Service Catalog
  - Backstage

AWS Proton

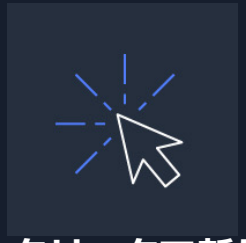
クラウドインフラストラクチャの管理を強化しつつ、開発チームの
イノベーションのペースを加速する マネージドサービス

開発者向けのセルフサービス UI(AWS管理コンソール)

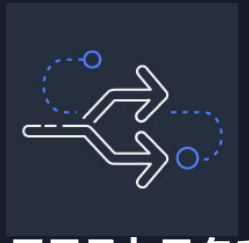
一元的な管理と可視性



テンプレートの バージョン管理



ワンクリックで新しい バージョンへの アップグレード



インフラストラクチャ パイプライン

3rdPartyとの統合 (デプロイパイプライン)

AWS Proton

開発チームは、サービステンプレートを選択し、
(テンプレートで宣言されているパラメータを入力して) デプロイする

1. サービス テンプレートを選択

例：Fargate Webサービス
Lambda関数など

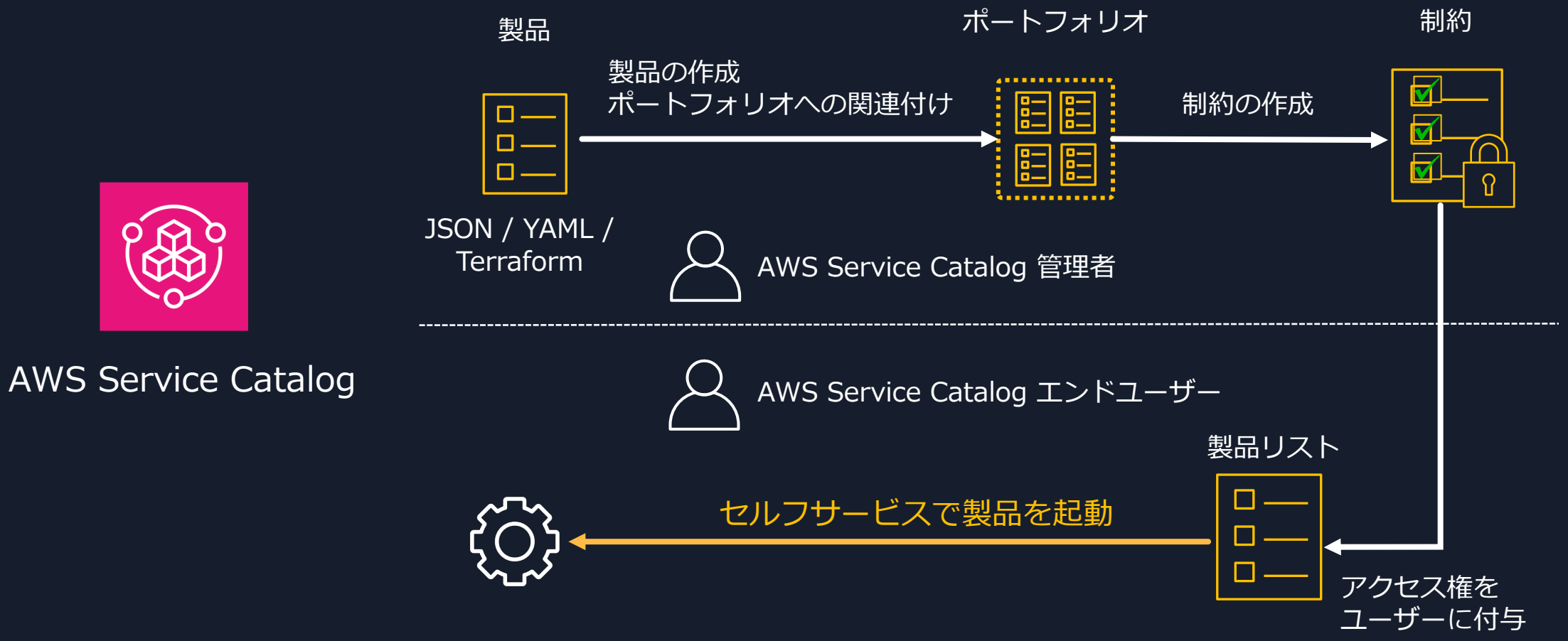
2. パラメータの入力

例：サービス名、
ドメイン名、vCPU、メモリ

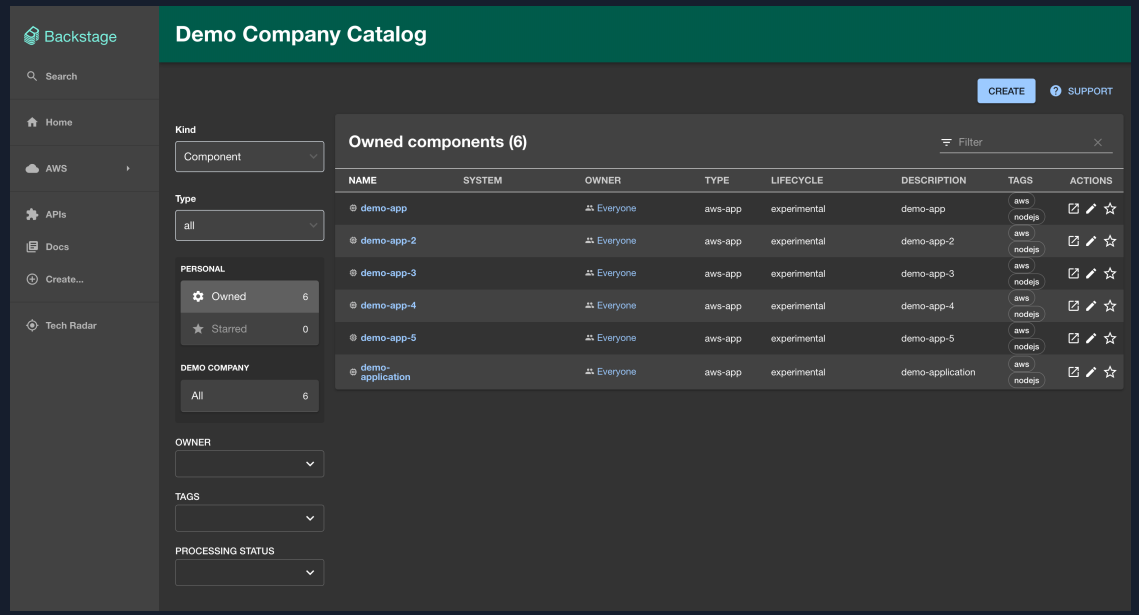
3.デプロイ

AWS Service Catalog

AWS Service Catalog による、セルフサービスでのプロビジョニング



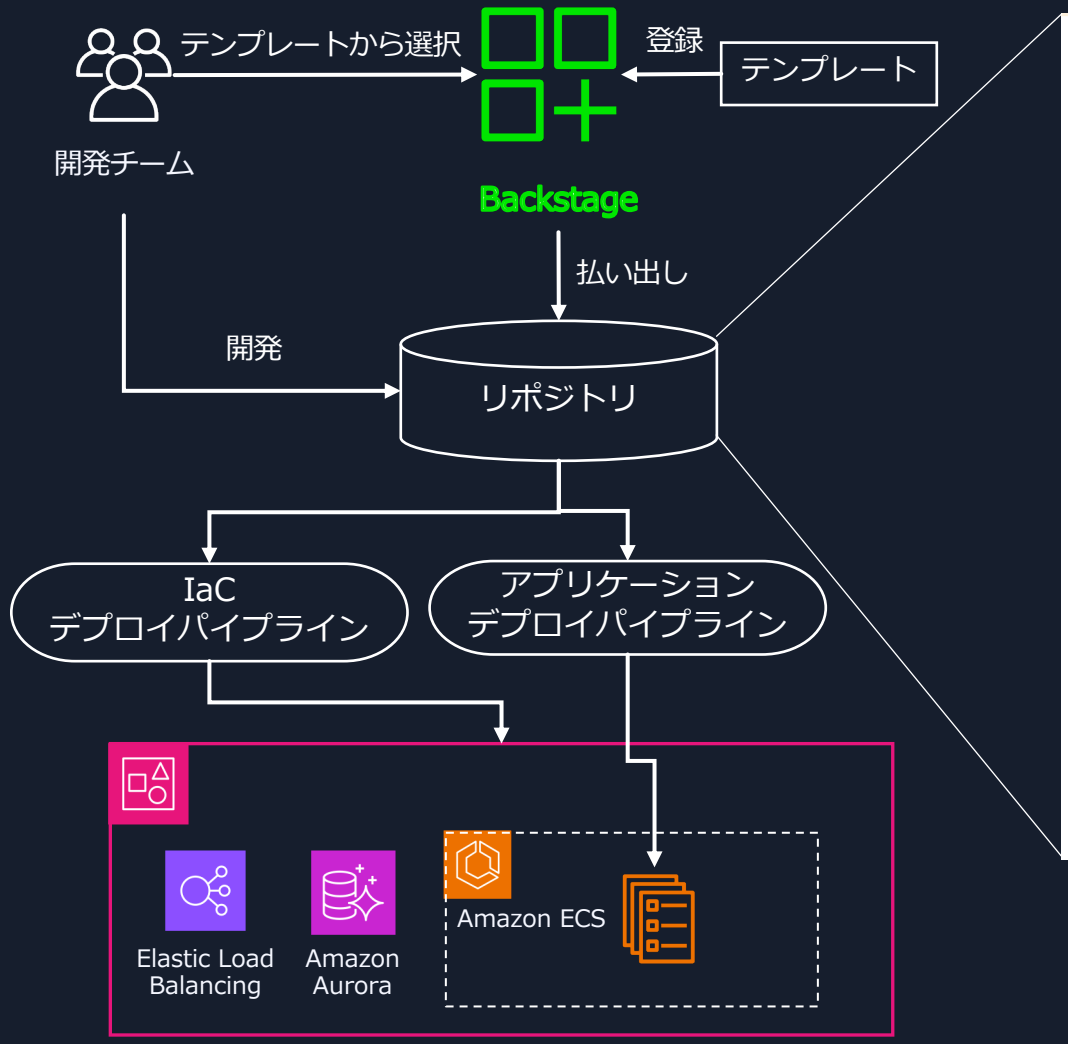
Backstage

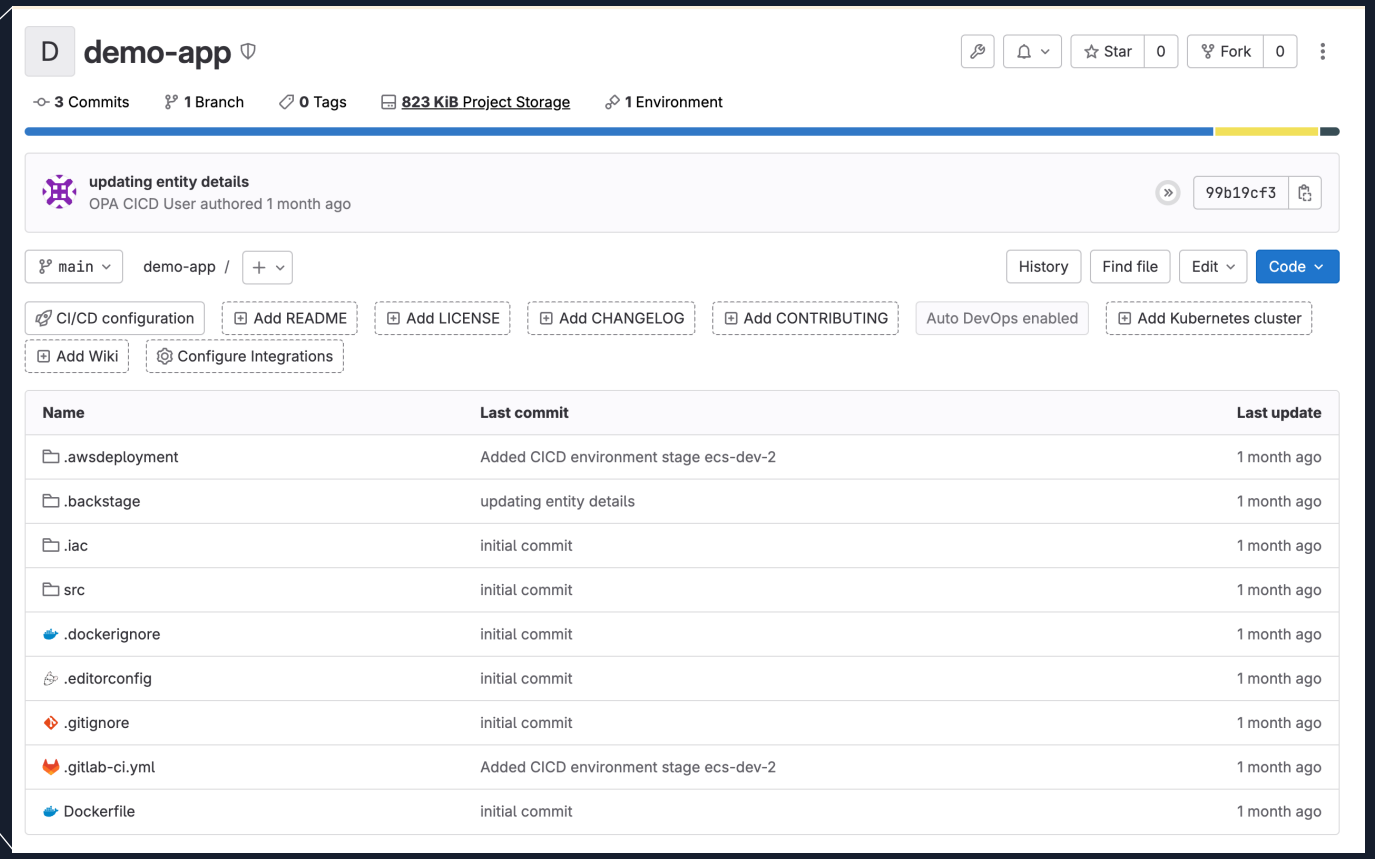


- 社内のソフトウェア資産（マイクロサービス、ライブラリ、データパイプライン、Webサイト、MLモデルなど）を「サービスカタログ」として管理することにより、開発者自身が民主的、自律的に問題へ対処できる

- 内部開発者ポータルを実現するオープンソース

Backstage によるプラットフォームの実装例





デモ

Agenda

- 事業の拡大と共に、開発組織が直面する課題
- プラットフォームエンジニアリングの始め方
- Amazon ECS で実現するプラットフォームエンジニアリング
- まとめ

まとめ

- プラットフォームエンジニアリングというアプローチは、スケールする開発組織が抱える課題の解消の一助となりうる
- プラットフォームエンジニアリングを進めるためには、プラットフォームをプロダクトと捉えることが重要
  - ユーザーからのフィードバック、TVP、責任分担の合意
- プラットフォームエンジニアリングを実現するためには、様々なツールやサービスを組み合わせて、開発組織にあった形で実装する必要がある

- **AWS のソリューションアーキテクトにご相談ください！**

Thank you!

Kenta Goto (ごとけん)

@kennygt51