

# 最も効果的な エンタープライズシステム・モダナイゼーションの進め方とは

倉元 貴一

デジタルトランスフォーメーション部  
マイグレーションスペシャリスト ソリューションアーキテクト  
アマゾン ウェブ サービス ジャパン合同会社

# 自己紹介

倉元 貴一 (Kuramoto Kiichi)

所属 : アマゾン ウェブ サービス ジャパン合同会社  
デジタルトランスフォーメーション部

職種 :マイグレーションスペシャリスト  
ソリューションアーキテクト

担当 : 様々なお客様のマイグレーションや  
モダナイゼーションにおける各種サポート

経歴 : Sler にて Java アプリケーションエンジニア



# 本セッションについて

## 対象者

- 既存システムのモダナイゼーションをご検討中の方
- モダナイゼーションのためのアーキテクチャやAWSサービスに関心のある、アプリケーションアーキテクト・エンジニアの方

## ゴール

- 既存システムのモダナイゼーションの背景や取り組む観点、効果的な進め方について理解する
- モダナイゼーションに有効なAWSサービス・ツールを知っていただく

## 本セッションで話さない事

- 本セッションで登場する個別のAWSサービスの詳細

# アジェンダ

- モダナイゼーションとは何か
- モダナイゼーションを進めるポイント
- 段階的モダナイゼーションの実現方法
- まとめ

# モダナイゼーションとは何か

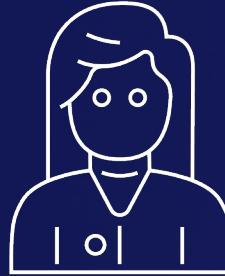
# モダナイゼーションのイメージ？



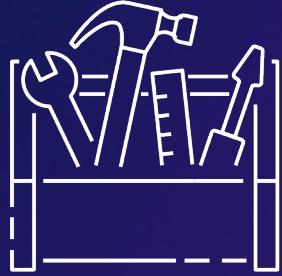
サービス



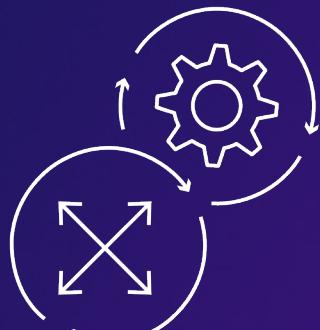
言語・アーキテクチャ



いずれも手段（How）  
何を・なぜ（What・Why）が  
必要

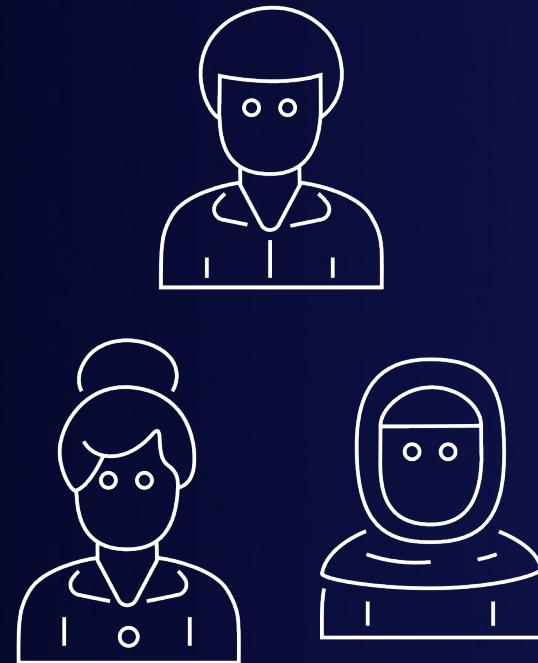


ツール



方法論

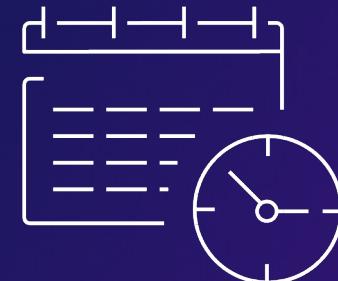
# モダナイゼーションの必要性？



ユーザーの多様化

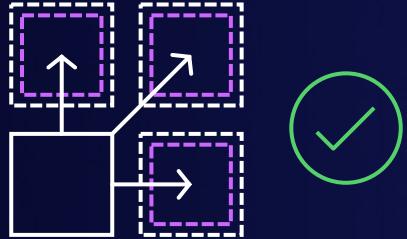


手段の多様化



加速度的に変化

# モダナイゼーションとは



IT技術、アーキテクチャ



人、プロセス、文化



組織、ガバナンス

モダナイゼーションとは、  
最新のインフラ、アーキテクチャ、組織パターンを組み合わせて  
現行システムをリファクタリングすることで、  
ビジネスの俊敏性を最大化すること。

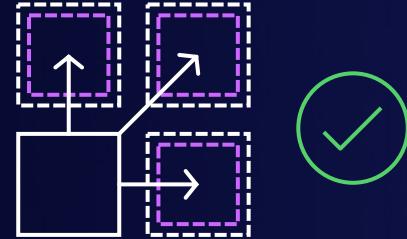
Modern : 現代式の、最新の

モダンは常に変化している

Don't Modernize,

Keep it Modern

# 常に最新の適切な手法を選択し続ける



IT技術、アーキテクチャ



コンテナ、サーバーレス  
マイクロサービス



人、プロセス、文化



CI/CD、DevOps



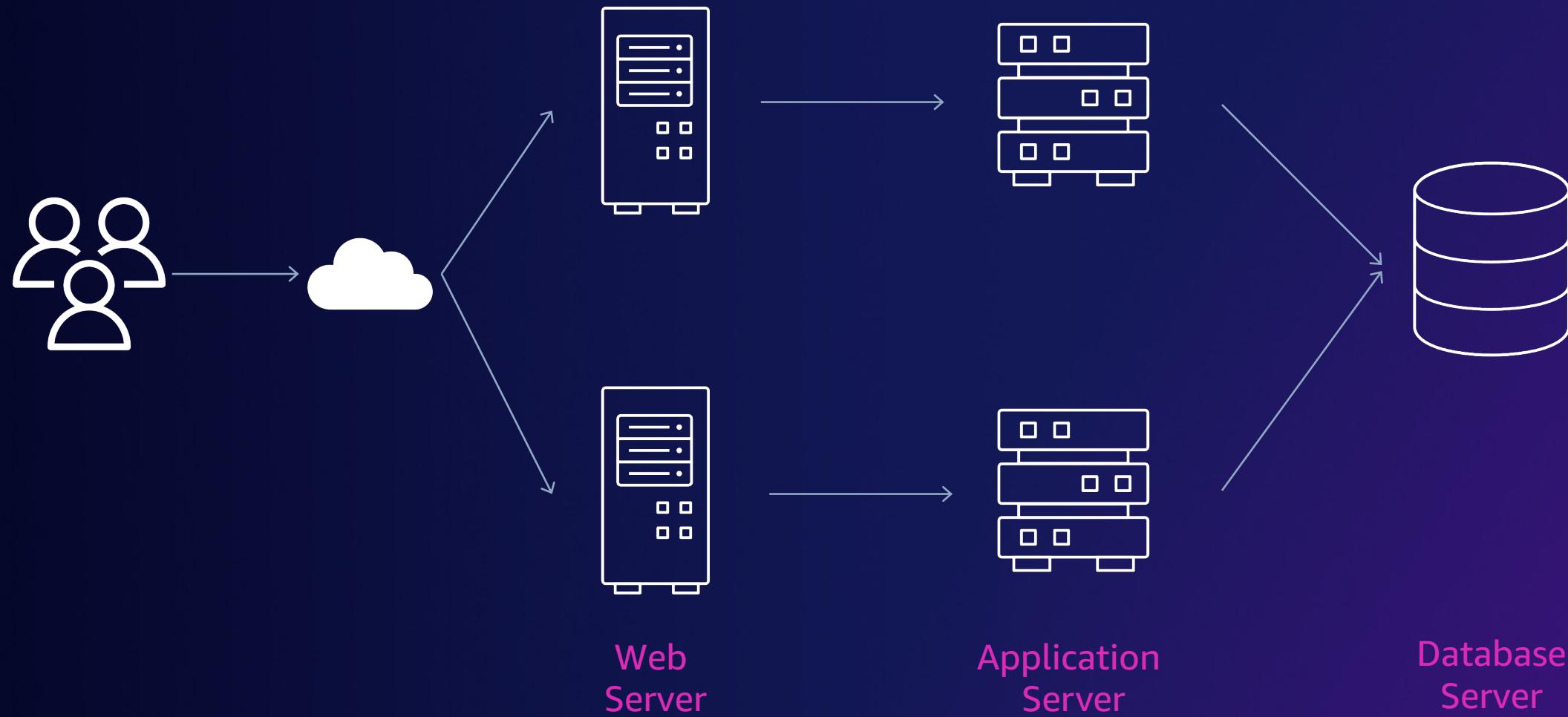
組織、ガバナンス



アジャイル、スクラム

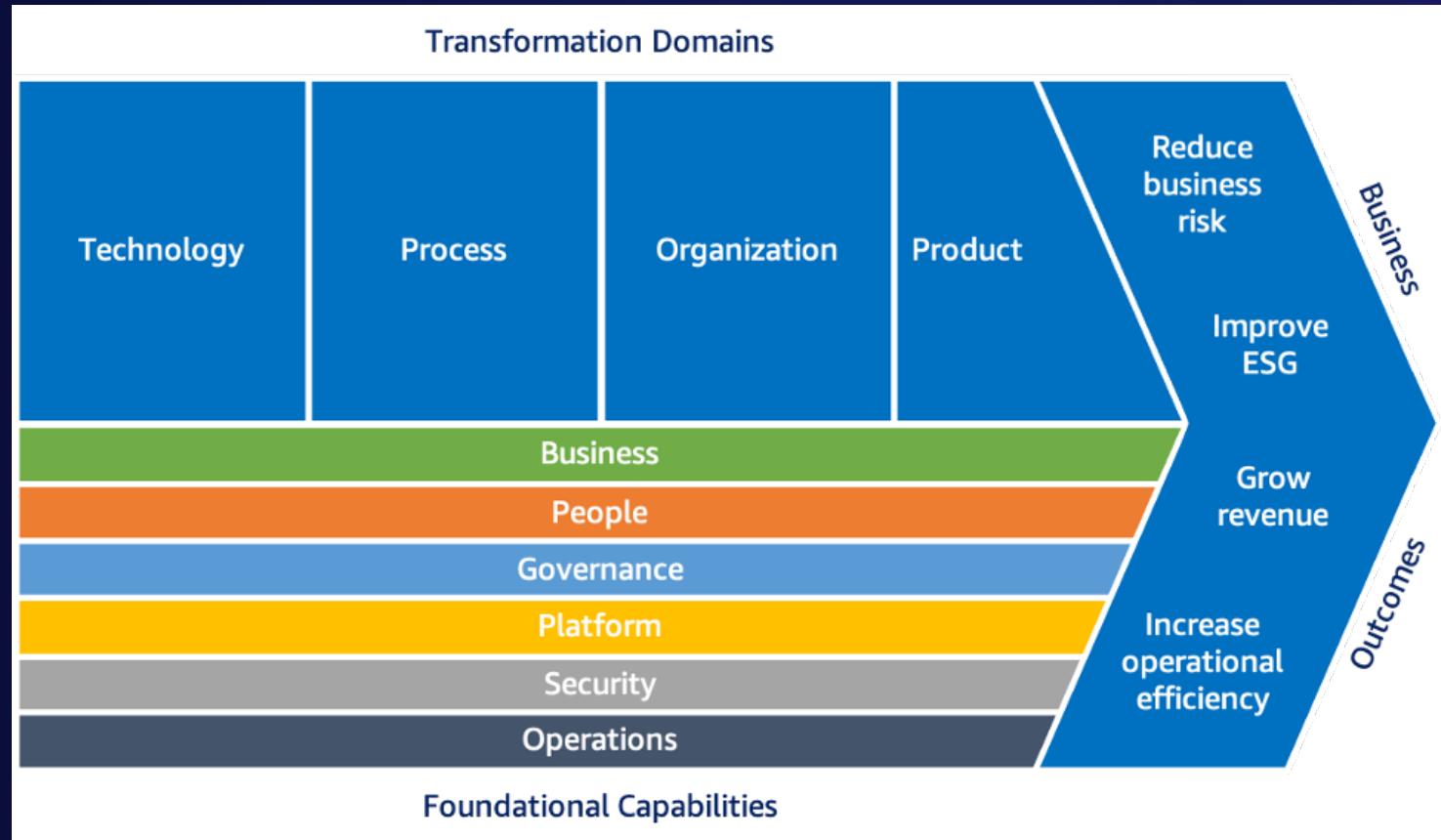
# モダナイゼーションを進める ポイント

# 従来のシステム構成



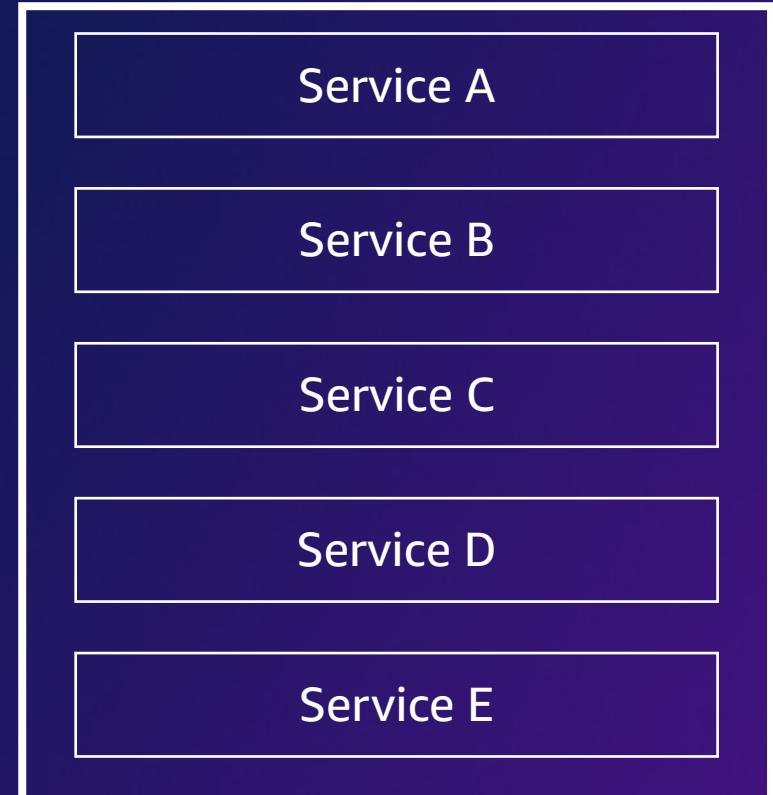
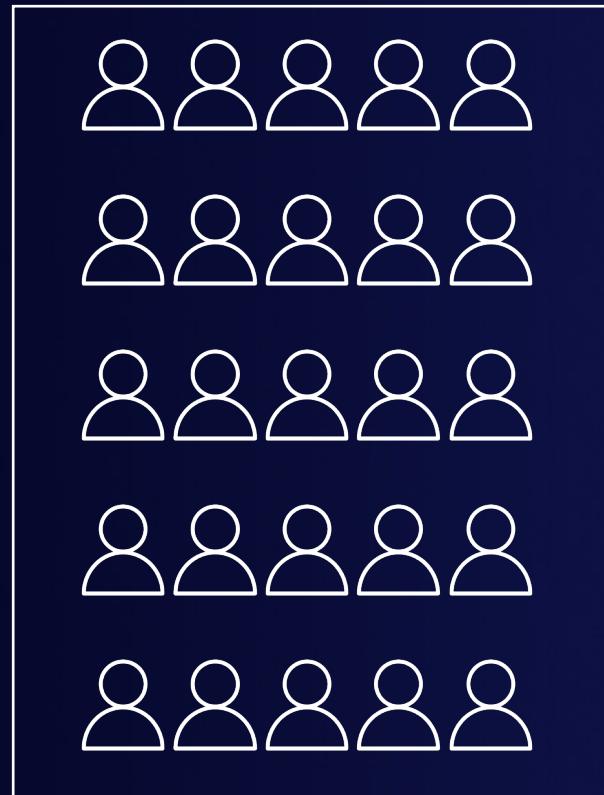
# モダナイゼーション対象の観点

利用する技術だけではなく、組織やプロセスも対象



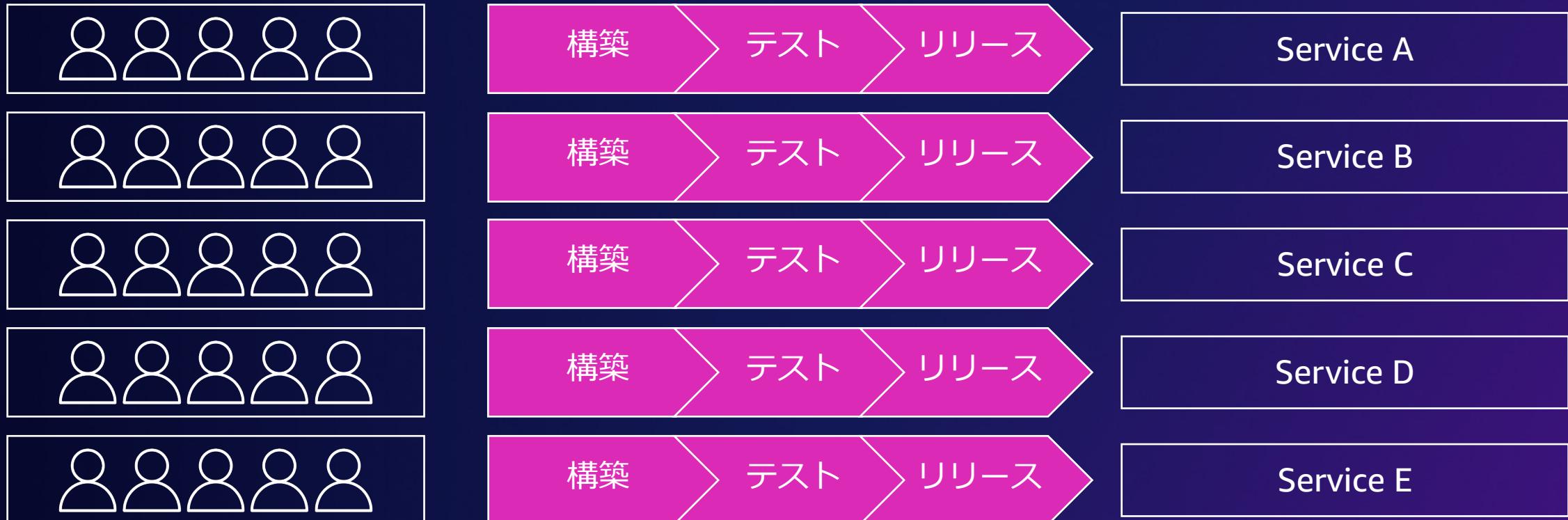
# 従来型開発の構造

数年単位の大規模プロジェクトは変化に弱い



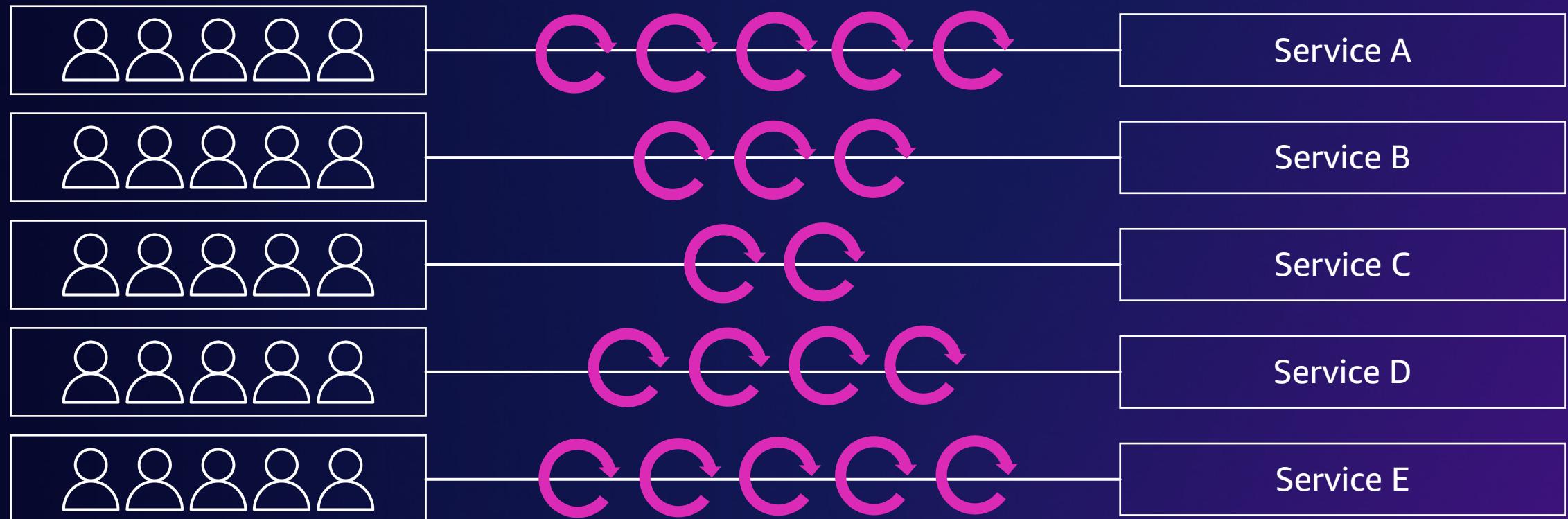
# 変化に対応しやすい組織体制

独立した開発チームごとにサービスを作る



# 変化に対応しやすい開発サイクル

独立した小さな開発サイクルを連続して回す



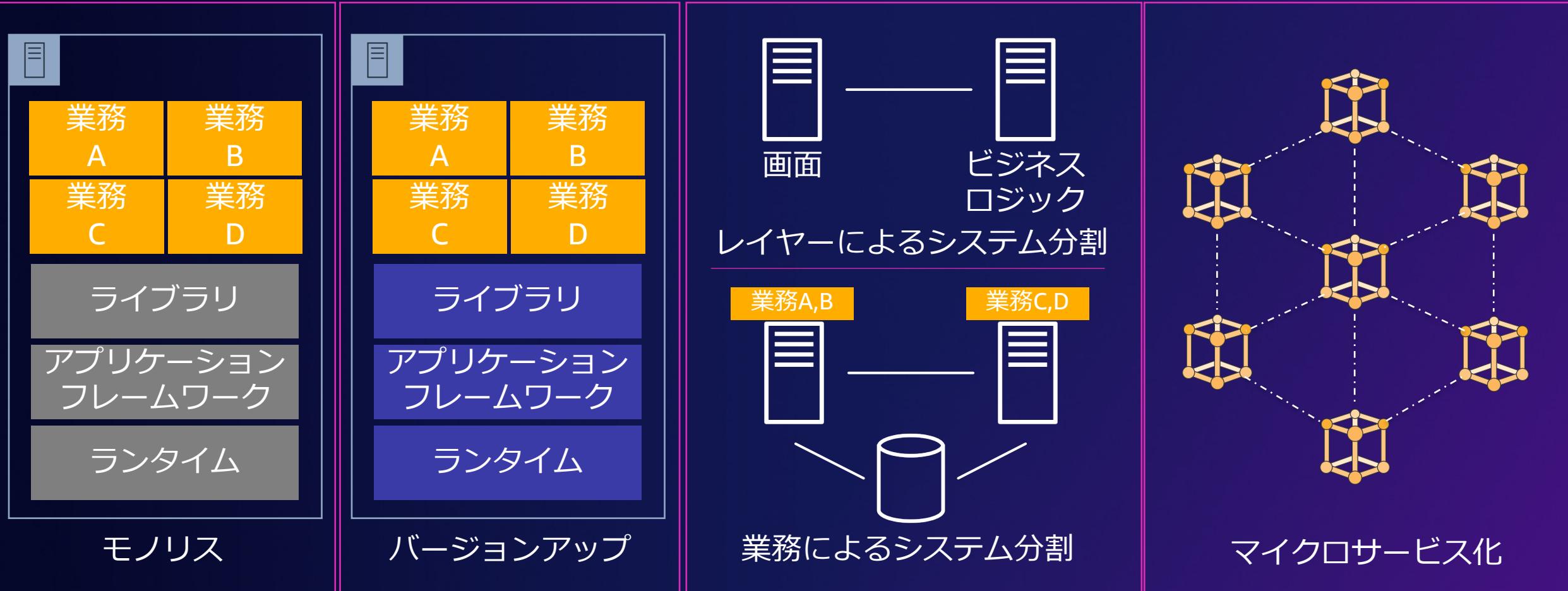
# インフラストラクチャ・モダナイゼーション

作業を可能な限りサービス側にオフロード

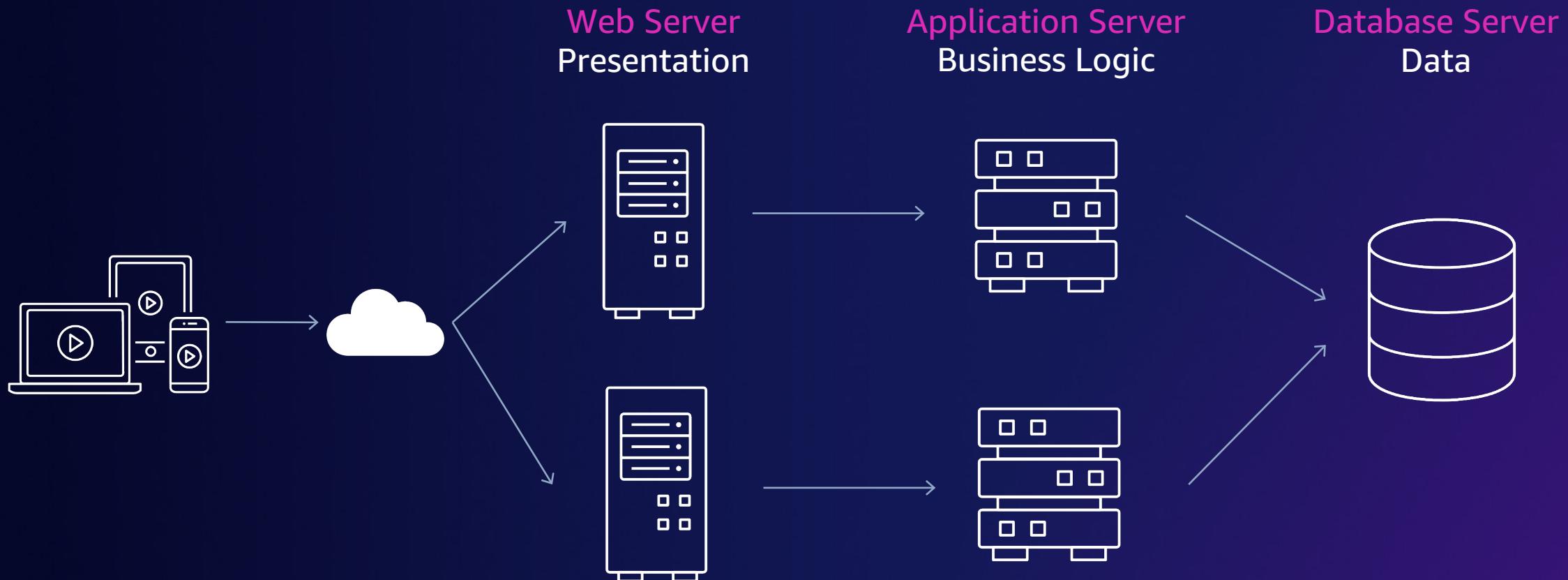


# アプリケーション・モダナイゼーション

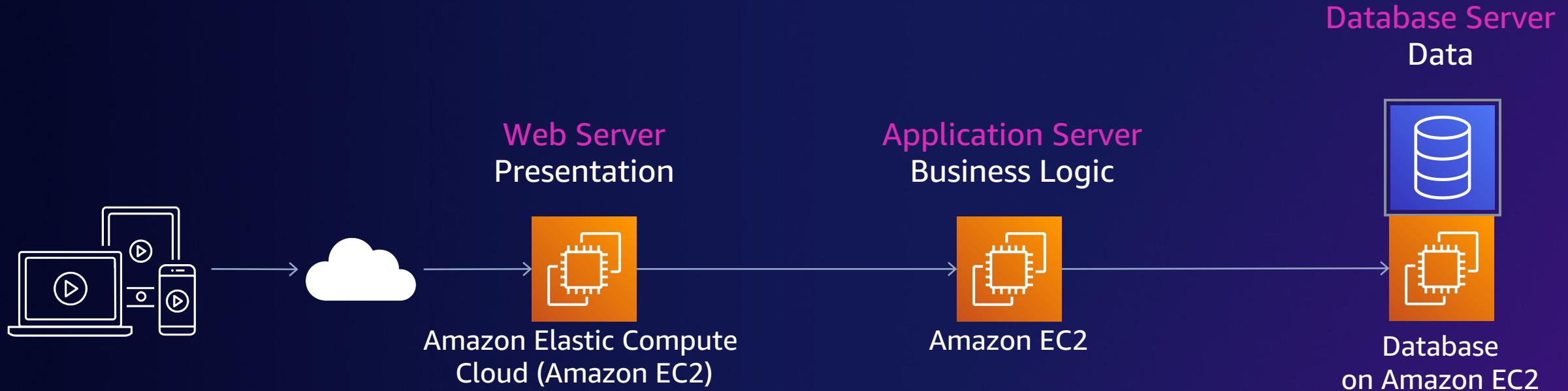
独立した開発のために分割・疎結合化



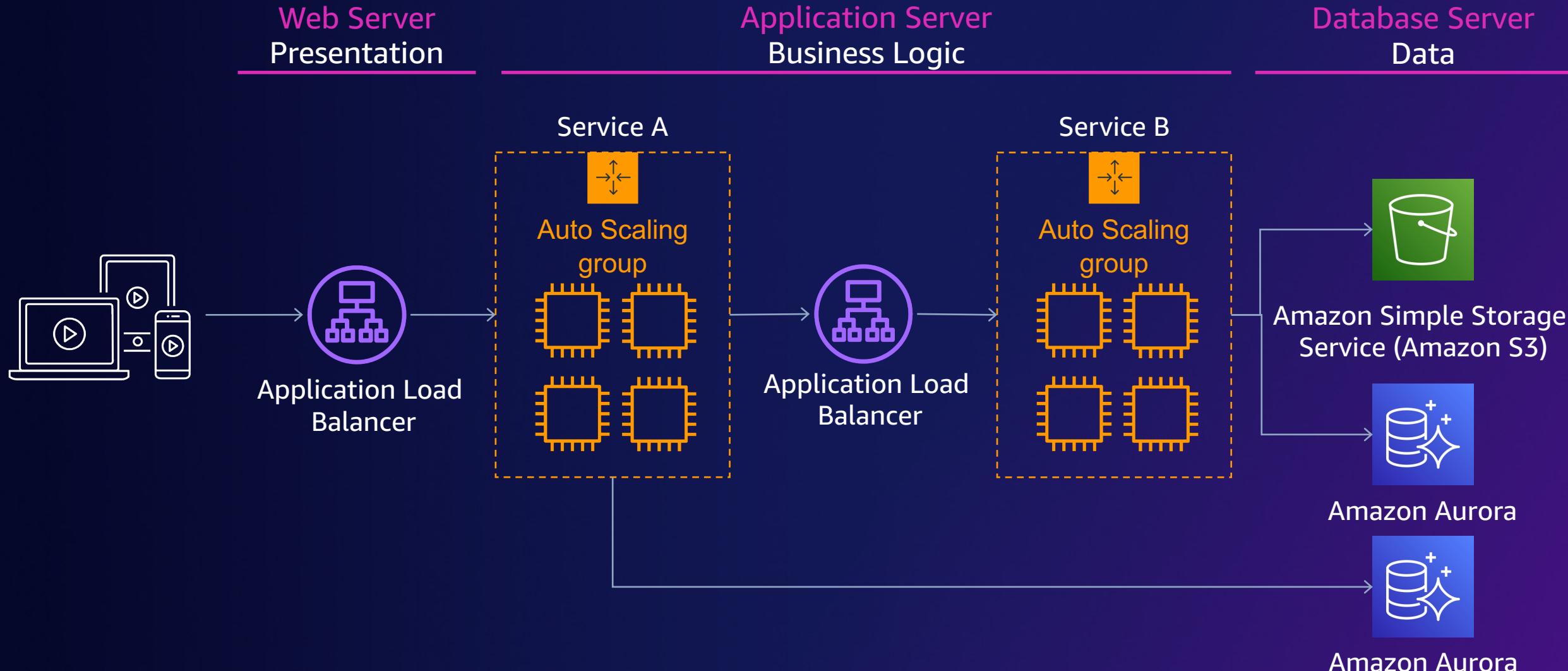
# 【再掲】従来のシステム構成



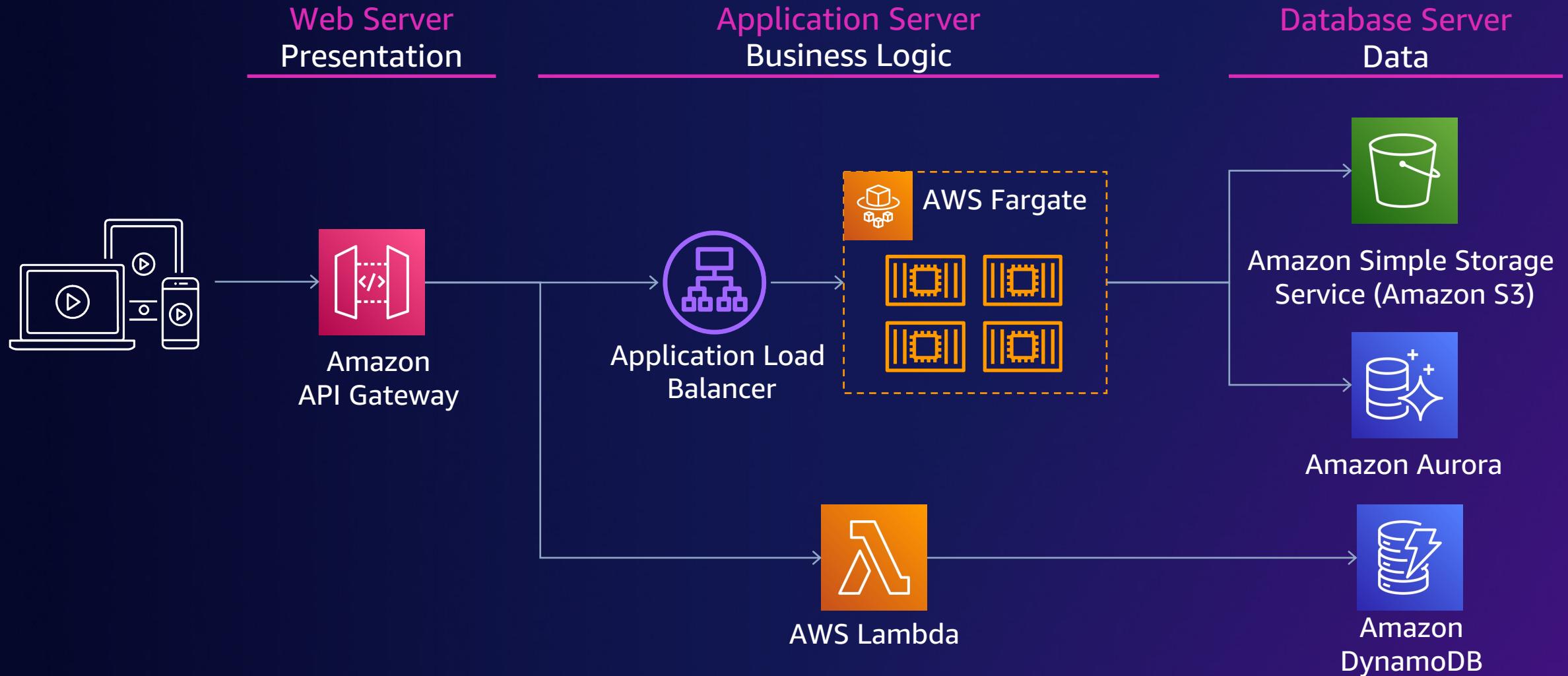
# AWS上で実現すると・・・



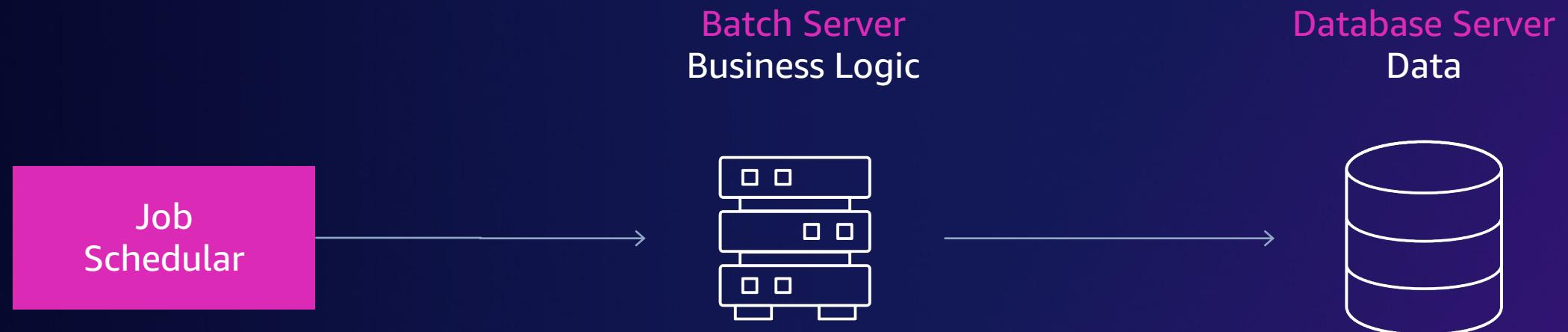
# Amazon EC2 によるモダナイズ



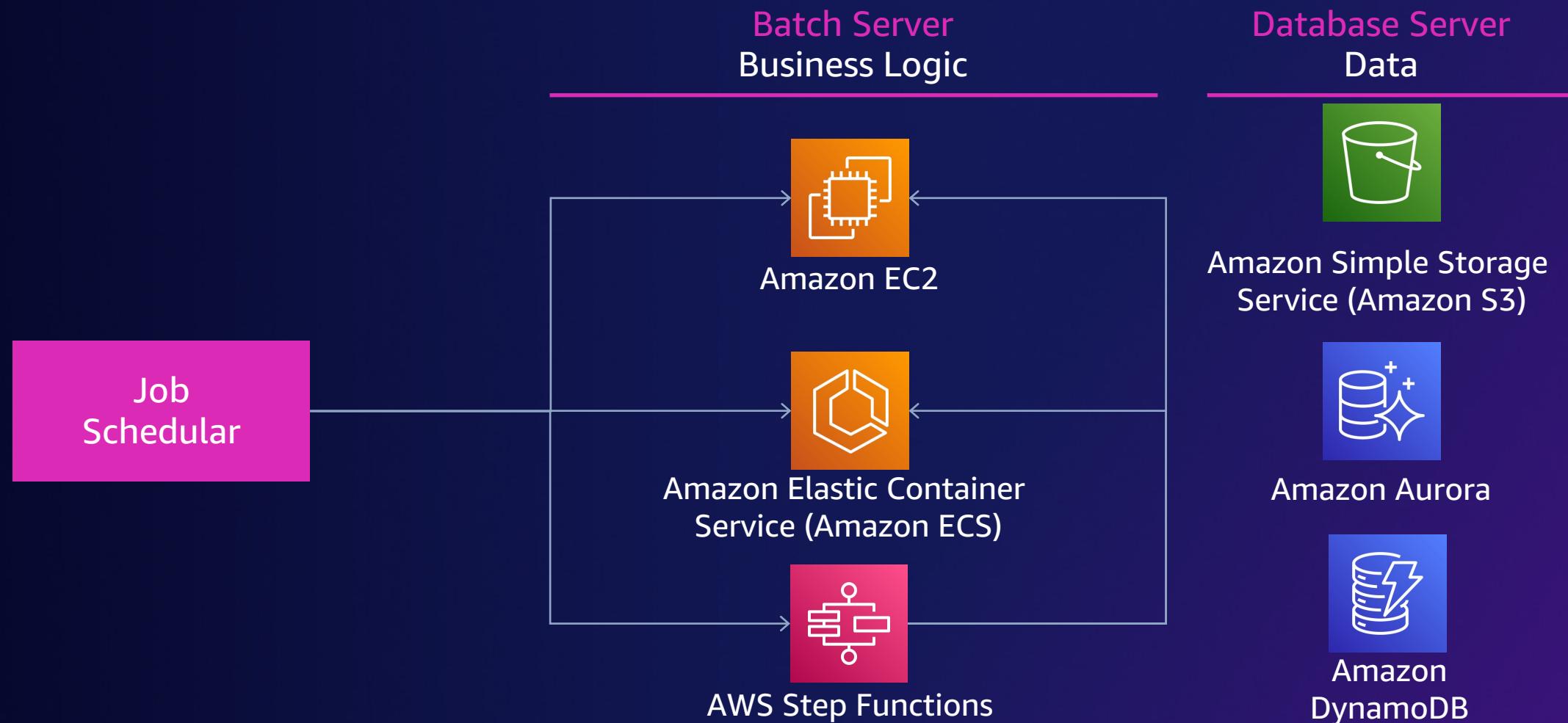
# コンテナやサーバーレスを活用したモダナイズ



# バッチアプリケーションの場合



# バッチアプリケーションもモダナイズ

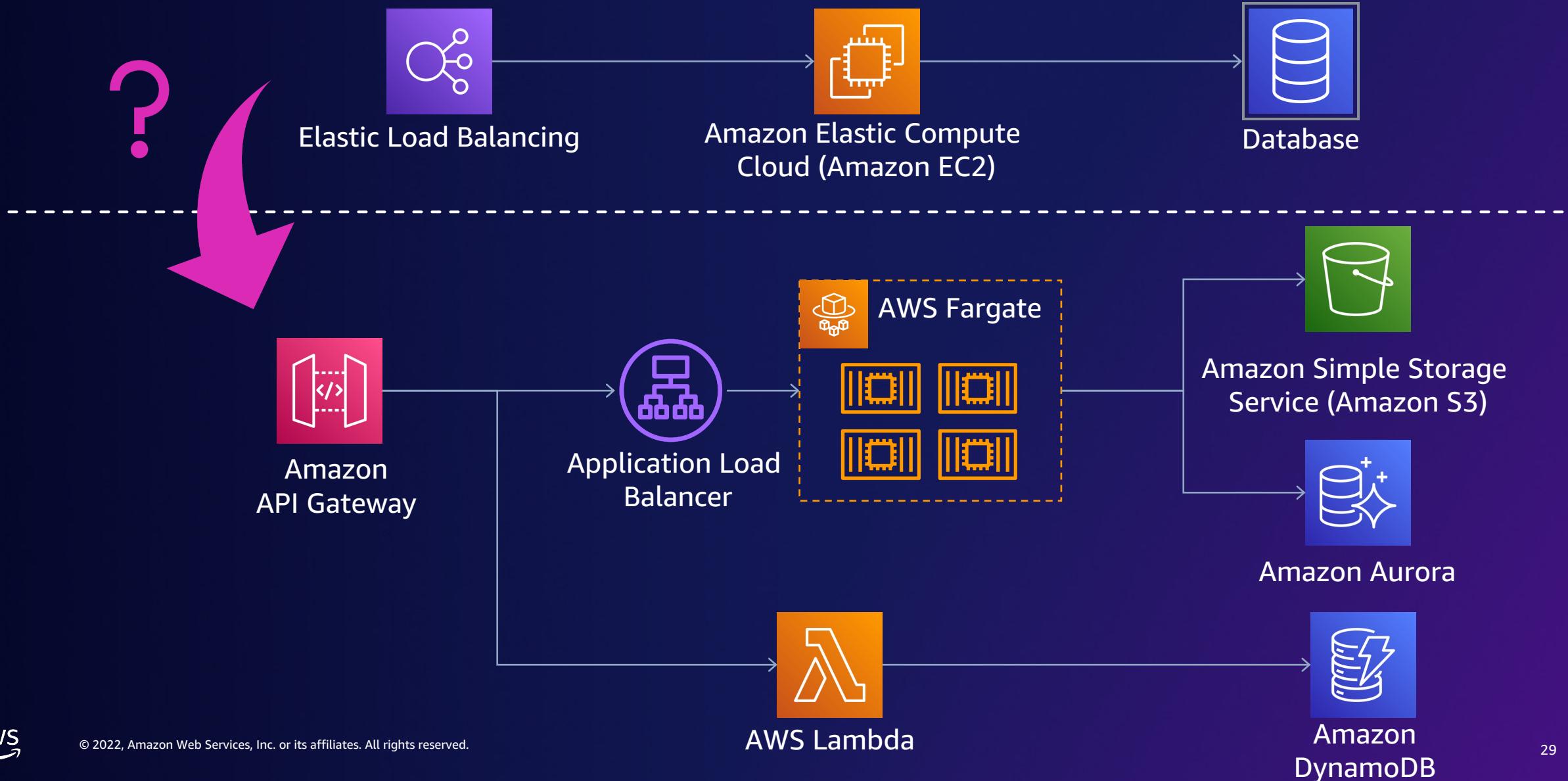


# モダナイゼーションを進めるポイントとは

- ◆ 変化に対応しやすい体制・開発サイクルを作る
- ◆ 最も注力すべき課題に着目して、FromとToを設定する

# 段階的モダナイゼーションの 実現方法

# 段階的モダナイゼーションを実現するには？



# Strangler Fig パターンの活用

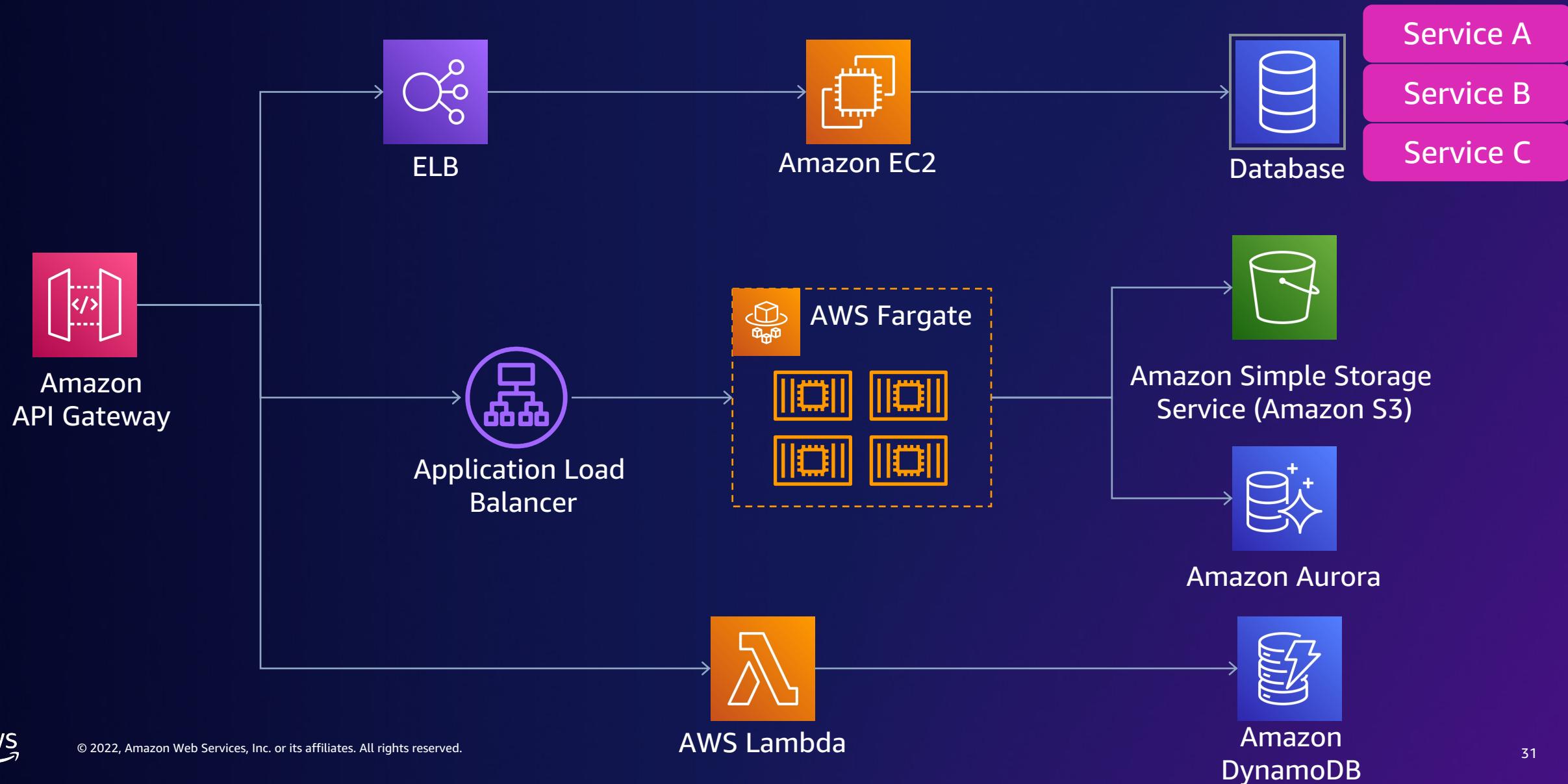


*“Gradually create a new system around the edges of the old, letting it grow slowly over several years until the old system is strangled.”*

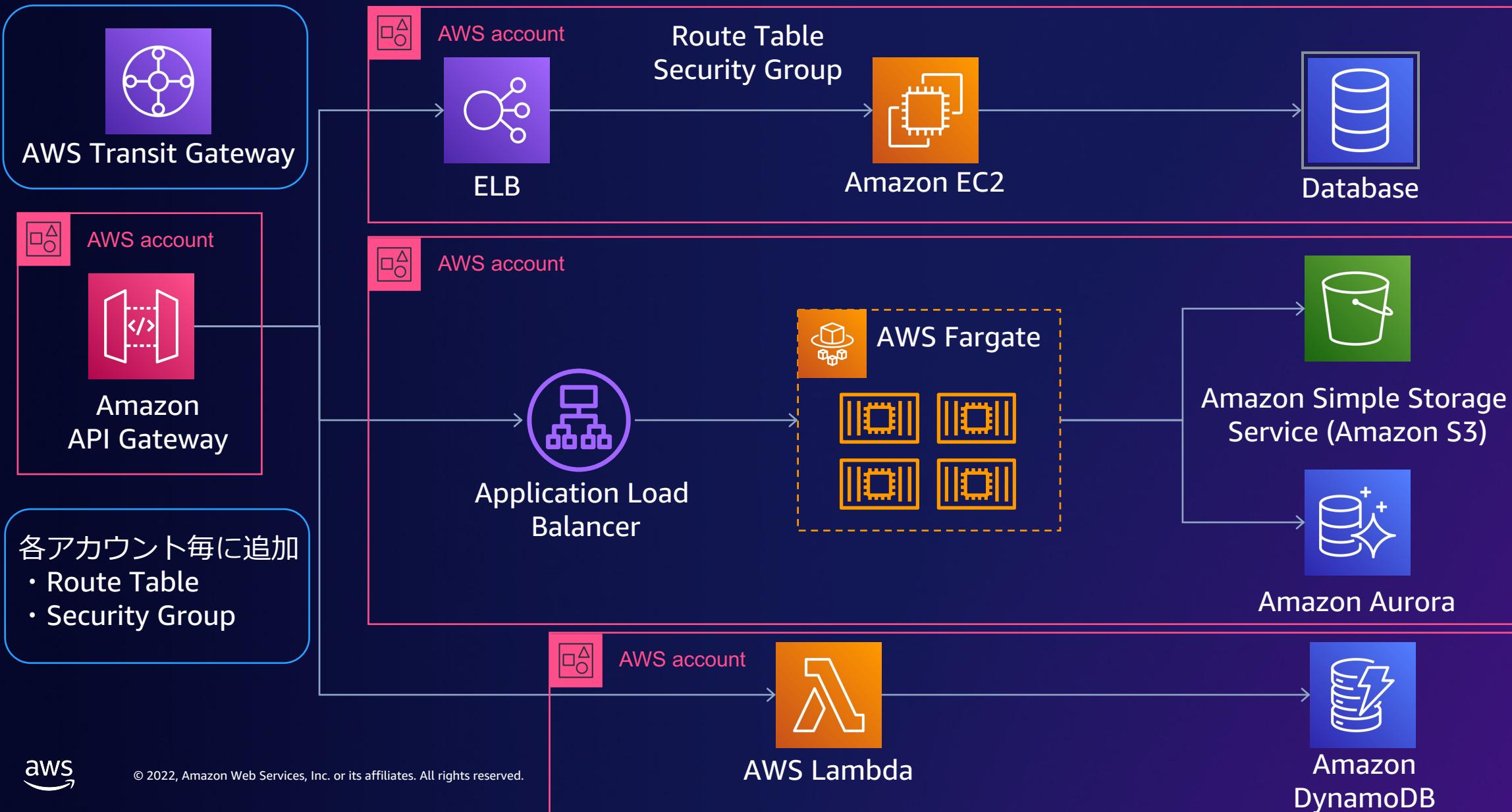
Martin Fowler, Thoughtworks

- 小さく俊敏に始められる
- すぐにモダナイゼーションの効果を得られる
- 容易に切り戻せるのでリスクを低減できる

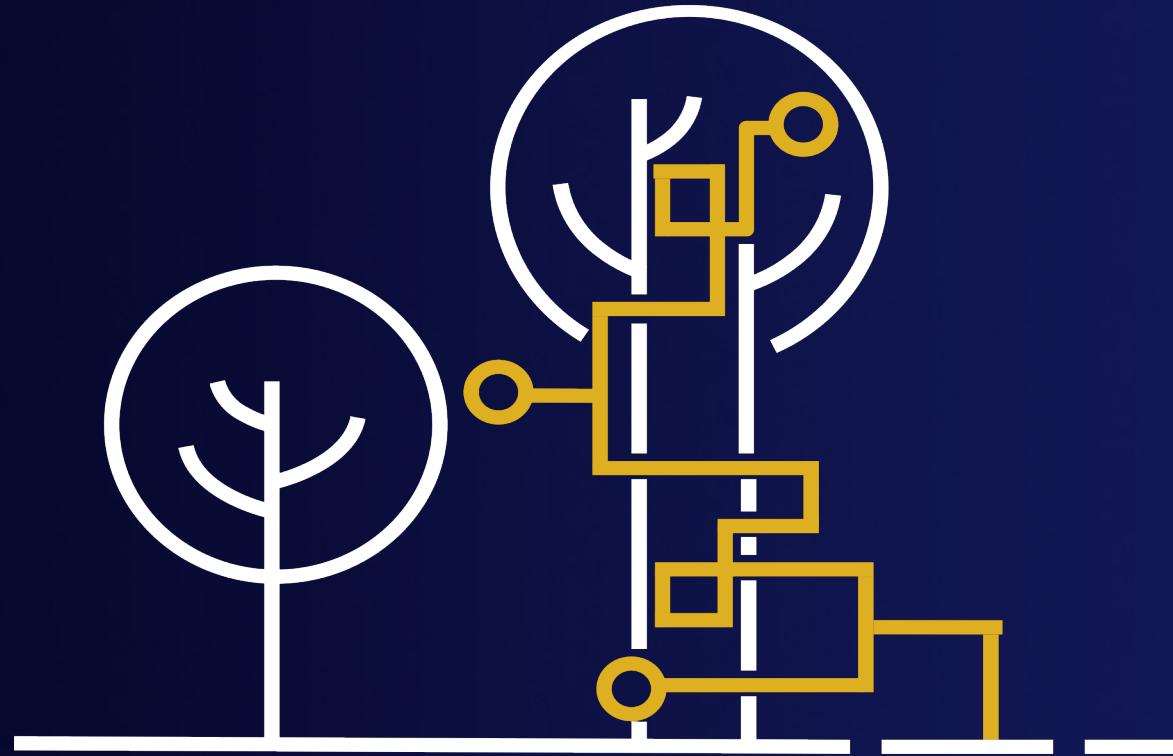
# オンラインアプリケーションへの適用



# 複数AWSアカウント構成は設定・管理が煩雑に



# AWS Migration Hub Refactor Spaces



リファクタリングを  
数ヶ月から数日に短縮

単一アプリケーションを、容易に  
レガシーアプリケーションとマイクロサービスの組み合わせへ

# AWS Migration Hub Refactor Spacesの特徴



## Migration Hub Refactor Spaces



リファクタリング環境  
のセットアップ・運用  
管理の手間を削減

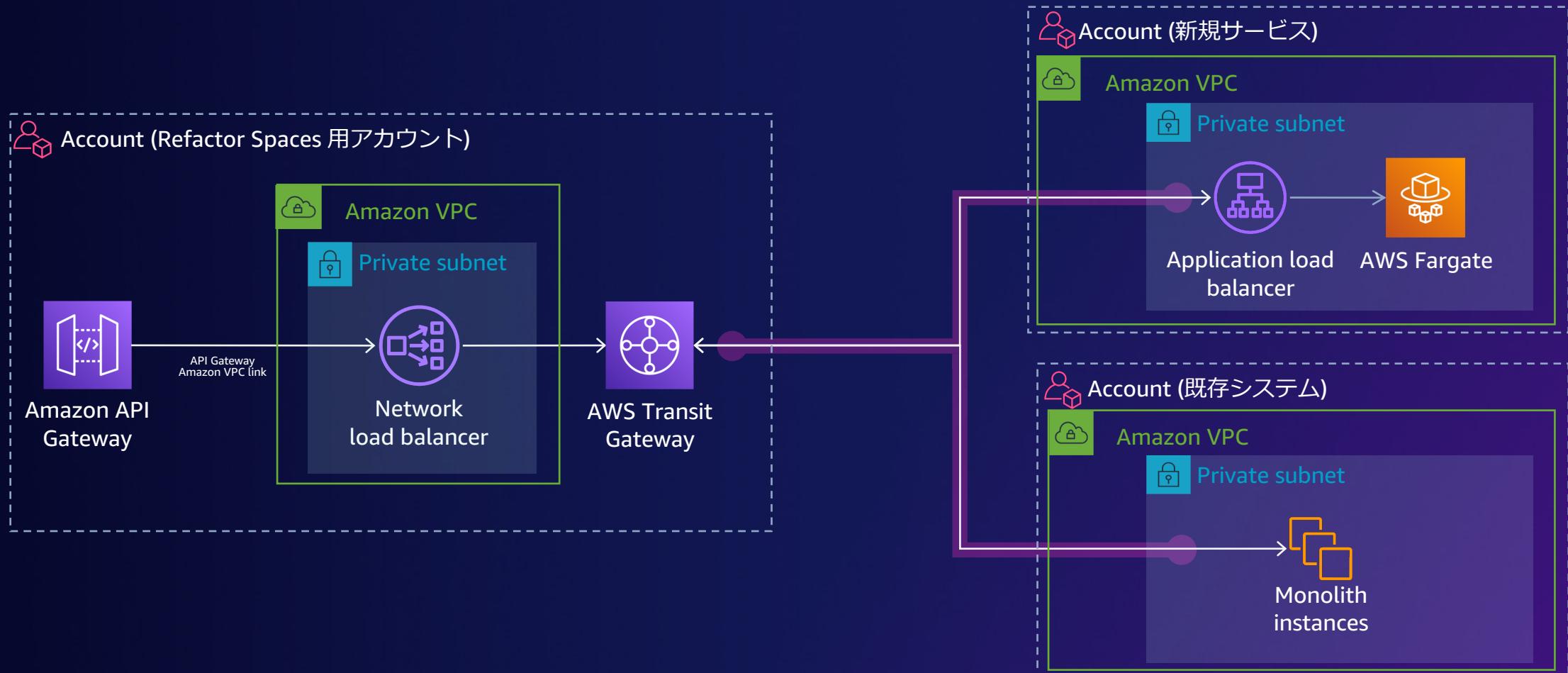
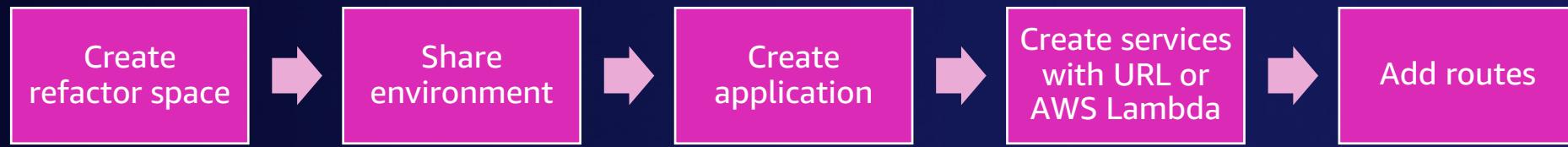


アプリケーションの呼  
び出し元に、リファク  
タリングの影響を与  
えない



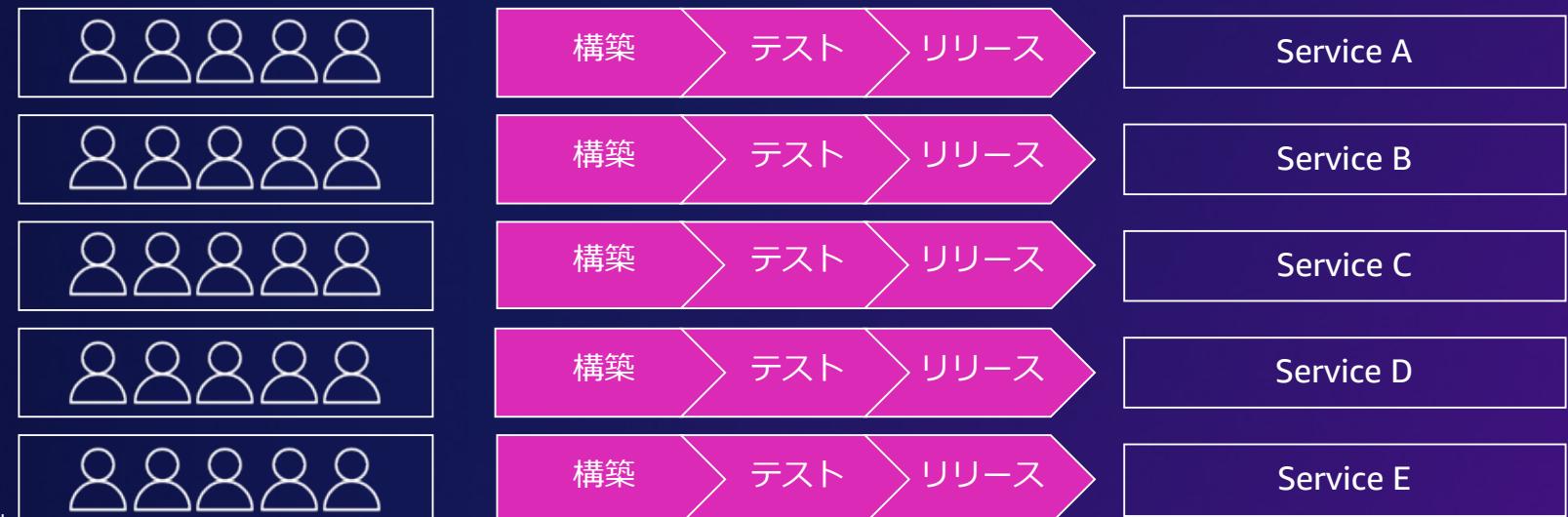
複数アカウントにまた  
がってトラフィックを  
ルーティング

# Refactor Spacesの構成例



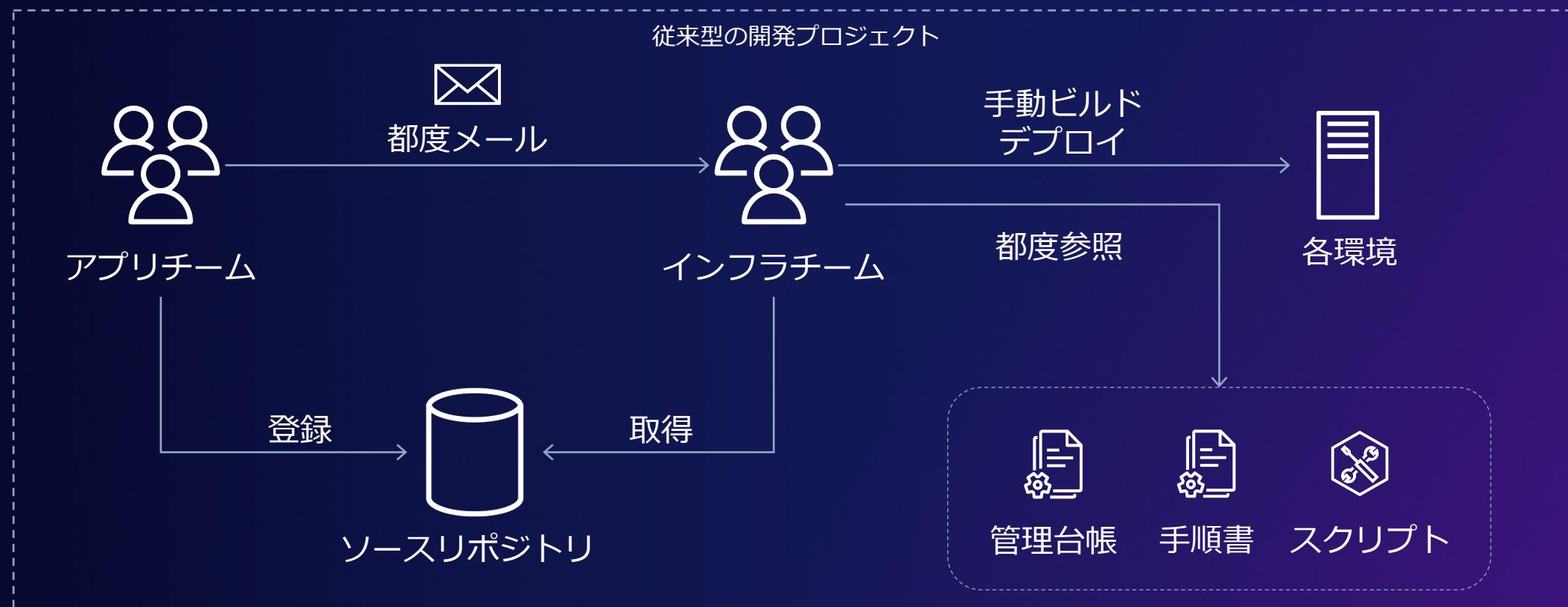
# 開発プロセスは今のまま？

既存システムのアーキテクチャを最新化したら、組織体制も解決するのか？



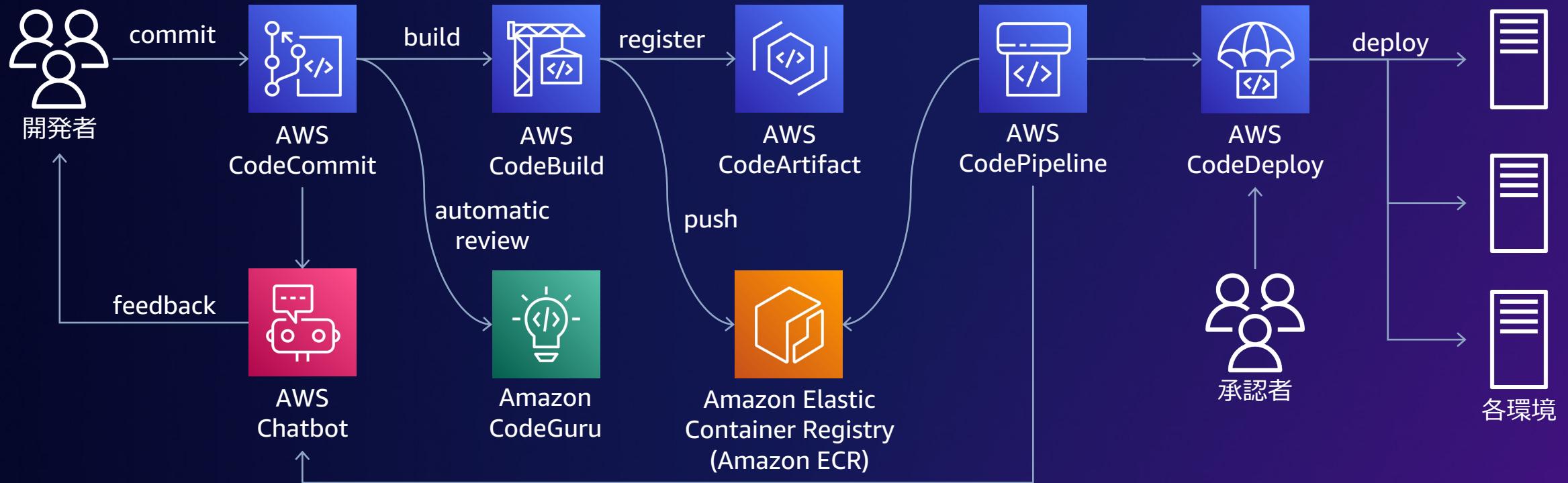
# 従来型の組織体制ではスケールしにくい

人手による作業・チームをまたがったコミュニケーションはボトルネックになりやすい



# 開発プロセスもモダナイズ

自動化・省力化によって人手をできるだけ排除し、ボトルネックになりにくい仕組みを作る



# まとめ

# 本セッションのまとめ

- ◆ モダナイゼーションはツールや方法論がポイントではなく、モダンであり続けるための組織体制や開発サイクルが重要
- ◆ あらゆる技術を導入するのではなく、現行システムにおける最も重要な改善ポイントに注力してFrom/Toを設定する
- ◆ 段階的モダナイゼーションを実現するために、AWSでは様々なサービスを提供している

Don't Modernize,

Keep it Modern

# 参考情報

- AWS クラウド導入フレームワーク (AWS CAF)  
<https://aws.amazon.com/jp/professional-services/CAF/>
- モダンアプリケーション開発  
<https://aws.amazon.com/jp/modern-apps/>
- AWS モダンアプリケーション開発 –  
AWS におけるクラウドネイティブ モダンアプリケーション開発と  
設計パターン (日本語版)  
[https://d1.awsstatic.com/whitepapers/ja\\_JP/modern-application-development-on-aws.pdf](https://d1.awsstatic.com/whitepapers/ja_JP/modern-application-development-on-aws.pdf)

# 関連セッション

AWS-03

エンタープライズシステムのモダナイゼーション  
～成功の鍵は「組織」が握っている～

AWS-05

リホストから始めるAWSへのサーバー移行

AWS-06

企業内 Windows Server アプリケーション移行手法とツール

AWS-45

サーバーレスを活用したイベント駆動アーキテクチャ

AWS-48

クラウドにリフトしたアプリケーションをコンテナ化するためのアプローチ

# Thank you!

倉元 貴一

デジタルトランスフォーメーション部  
マイグレーション スペシャリスト ソリューションアーキテクト  
アマゾン ウェブ サービス ジャパン合同会社

