AWS Black Belt Online Seminar

Modernize Enterprise Java Application

Modernization入門シリーズ

倉元 貴一

Solutions Architect
2023/07

自己紹介

名前：倉元 貴一

所属：アマゾン ウェブ サービス ジャパン合同会社

マイグレーション&モダナイゼーション
ソリューションアーキテクト

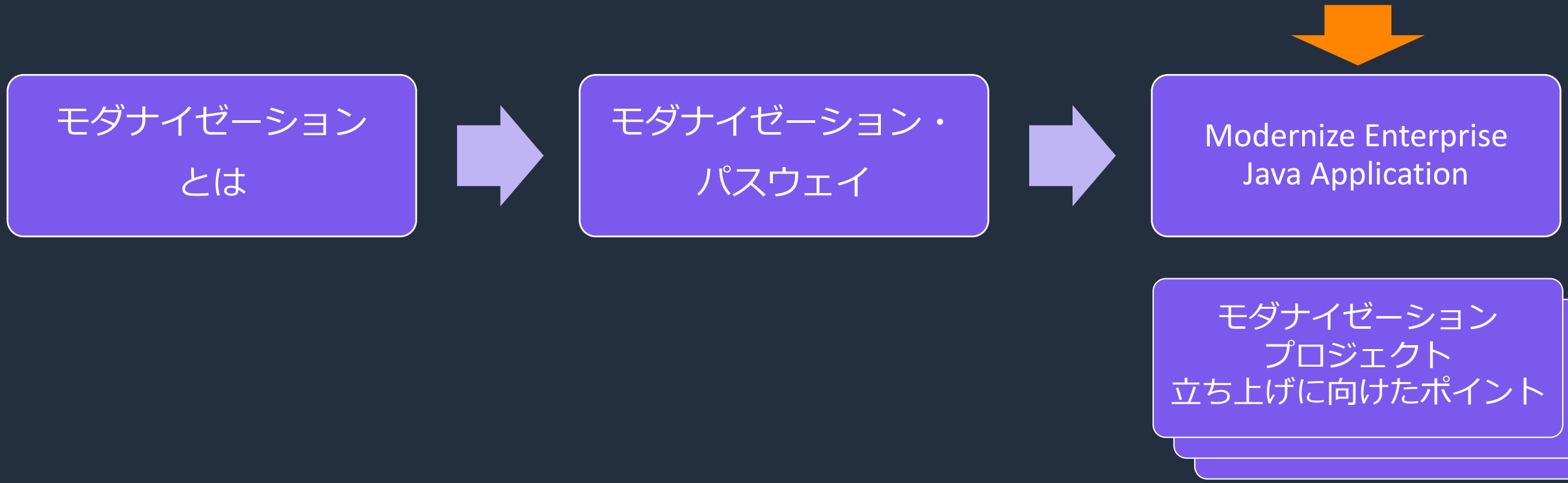
経歴：SIerにてJavaアプリケーションエンジニア

好きなAWSサービス：Amazon ECS, AWS Lambda,
AWS CodeBuild

Modernization入門シリーズ とは

モダナイゼーションについて初学者向けにステップを踏んで解説

本セッションは「モダナイゼーションとは」や
「モダナイゼーション・パスウェイ」の続編の位置付け



本セミナーの対象者 と 目的

対象者

- Javaアプリケーションのモダナイゼーションに関心のある、ITアーキテクトやソフトウェアエンジニア
- Javaアプリケーションの維持運用に課題を感じている方

目的

- Javaアプリケーションをモダナイゼーションするポイントやアプローチについて理解を深めていただく

アジェンダ

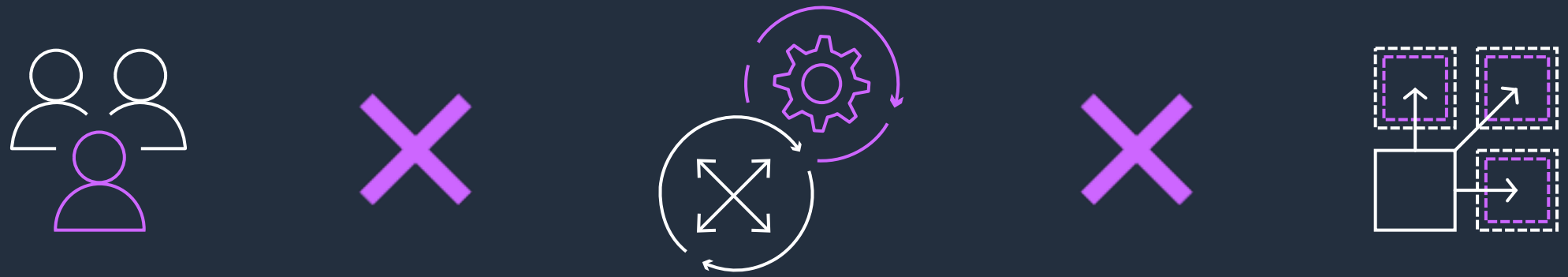
1. モダナイゼーションの目的
2. Javaアプリケーションの現状
3. AWSを活用したモダナイゼーションの実現
4. Javaアプリケーションモダナイズ例
5. Javaアプリケーション開発モダナイズ例
6. まとめ

モダナイゼーションの目的

モダナイゼーションと3つの柱

モダナイゼーションとは個別技術要素のことではなく
企業や組織が社会の変化にあわせて
素早く価値を提供し続けるために
組織やシステムを常に新しくしていくこと

人・プロセス・技術 の3つを軸に組み合わせて推進するとよい



People **Process** **Technology**

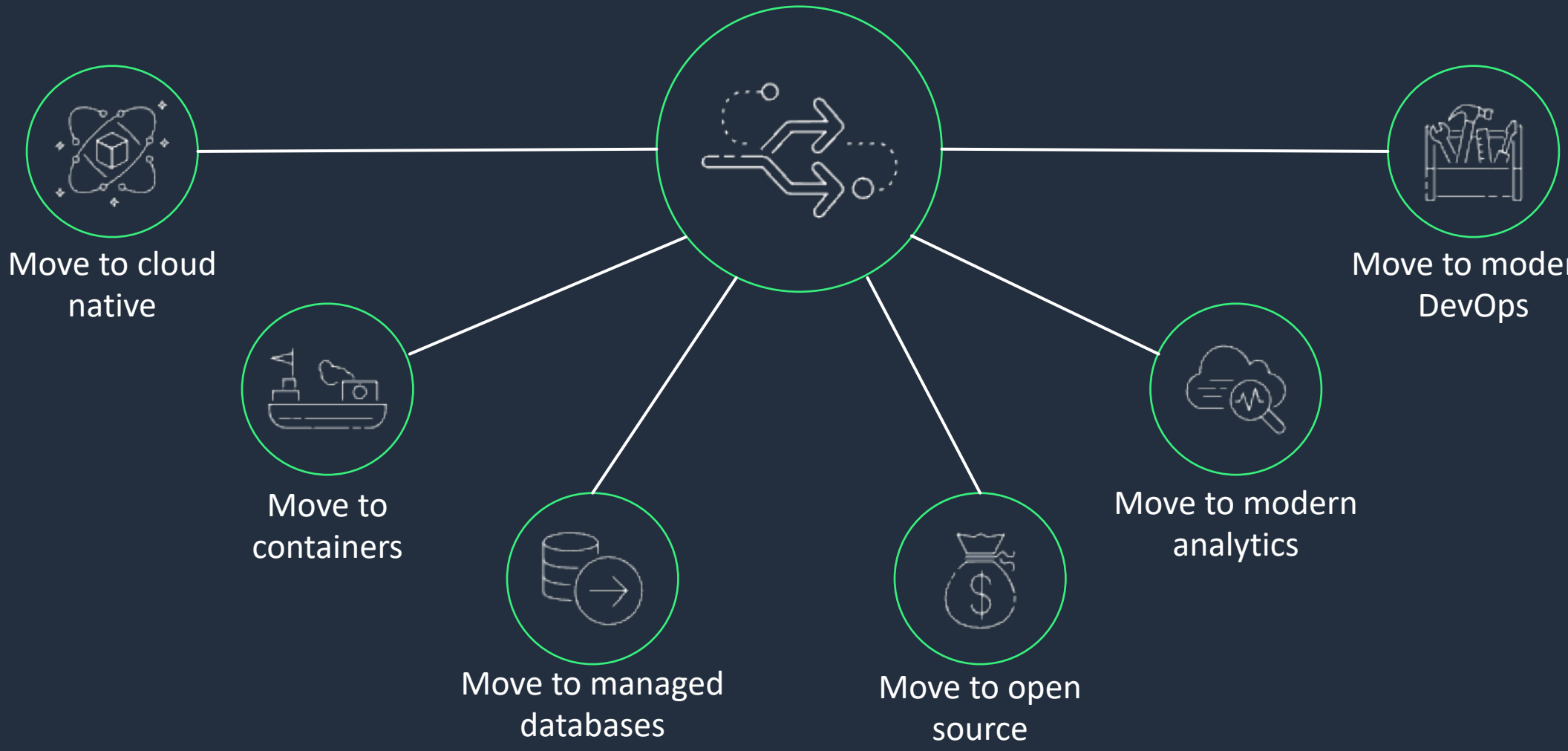
モダナイゼーションの主目的

目的によって実現手段は変わるはず

- インフラを早く調達したい
- 商用ライセンスから脱却したい
- 運用の手間をできる限り削減したい
- 新しい機能をどんどんリリースしたい

モダナイゼーション・パスウェイ ？

AWSが過去の支援を通じて見出した代表的な6つのパターン



パスウェイを要素とレベルでグループ化

1. パスウェイ上のどこからどこへ向かうかを意識する
2. モダナイゼーションの中にも基礎部分と応用的な部分がある

Javaアプリケーションの現状

このような疑問はありませんか？

他のプログラミング言語に変えないといけないのかな？

今使っている製品は変えないといけないのかな？

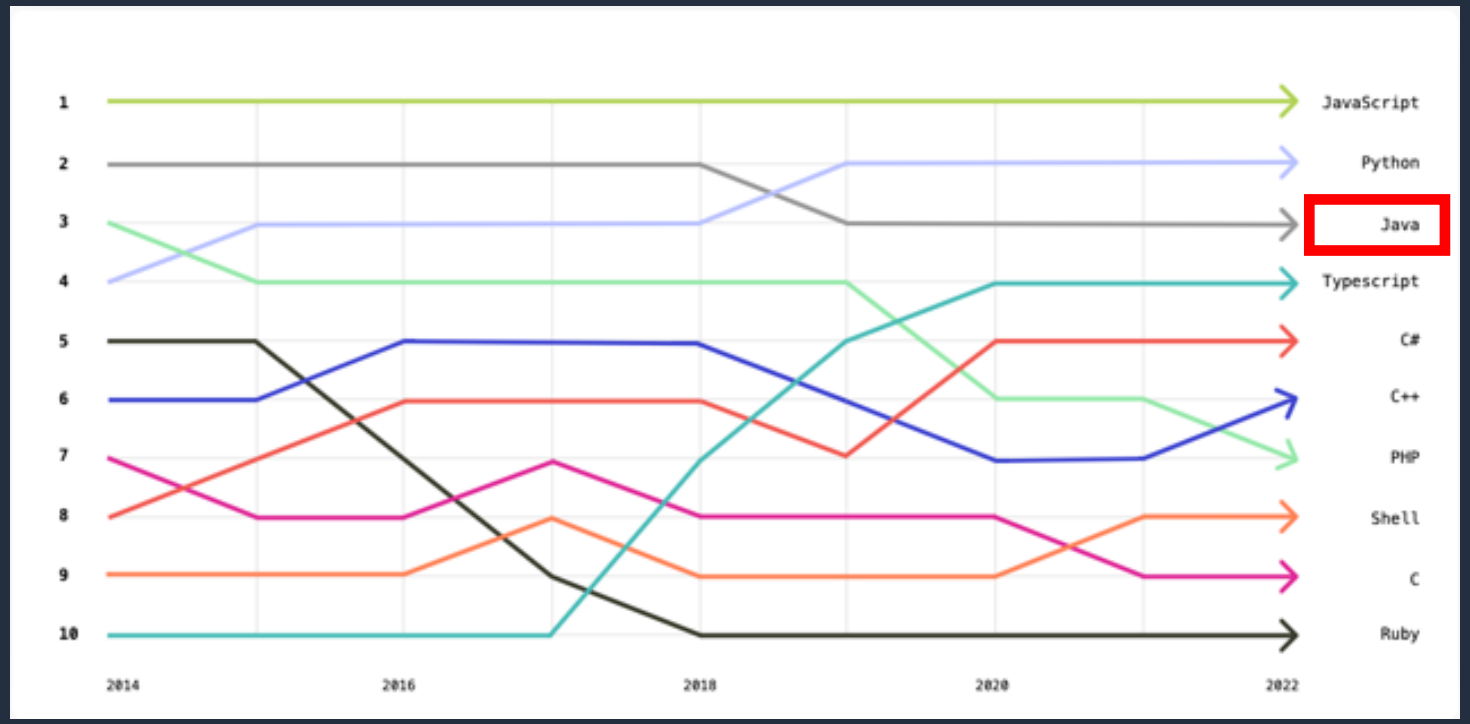
今のアーキテクチャを変えた方がいいのかな？
どう変えるとより良くなるんだろう？

Javaのシェア

“Java is the most popular programming language.”

IDC Survey Illustrates the Growing Importance of Developers to the Modern Enterprise, July 07, 2021

GitHub - Top languages used in 2022



https://octoverse.github.com/2022/top-programming-languages

Javaのメリット - 継続的なリリース

状況に応じた言語仕様拡充、リリースサイクル変更

1998-2002	J2SE 1.2-1.4 基礎機能の拡充
2004	J2SE 1.5 言語仕様の拡張(ジェネリクス, アノテーション)
2006-2014	Java SE 6-8 言語仕様の拡張(ラムダ式, Stream API)
2017-2021	Java SE 9-17 リリースサイクル変更(3年ごとのLTSリリース)
2022-2023	Java SE 18-20 リリースサイクル変更(2年ごとのLTSリリース)

Javaのメリット - 充実したエコシステム

IDE

- Eclipse
- Intellij IDEA
- Visual Studio Code

ビルド

- Maven
- Gradle

フレームワーク

- Spring Framework/SpringBoot
- Quarkus
- Micronaut

テスト

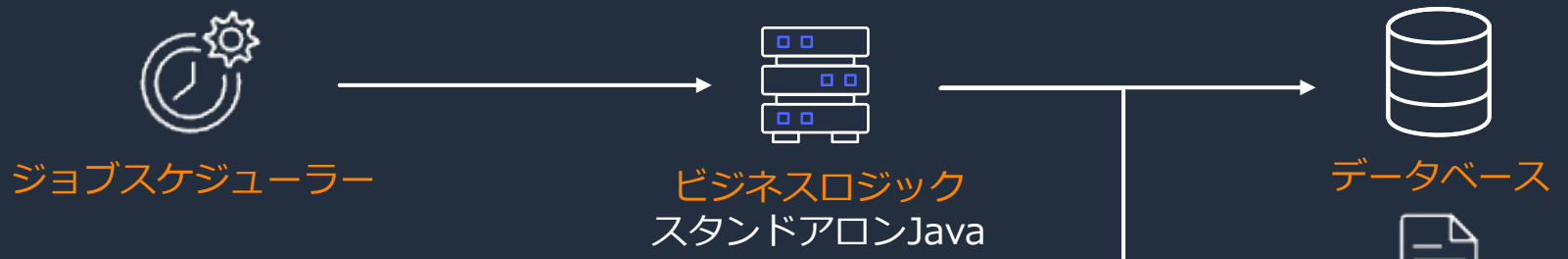
- JUnit
- Selenium

etc.

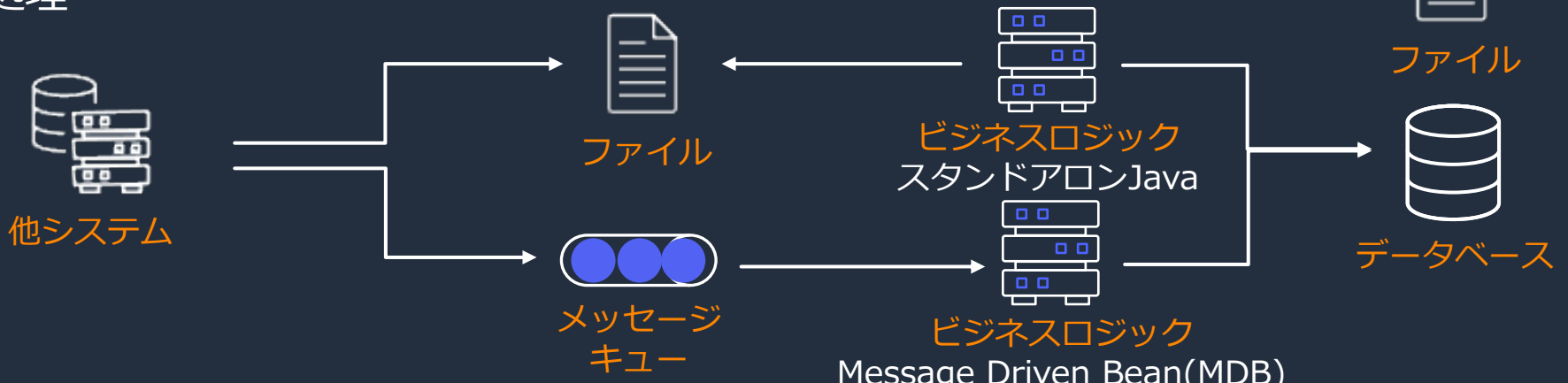
エンタープライズJavaアプリケーション アーキテクチャ例

①: オンライン処理

②:バッチ処理



③:非同期処理



AWSを活用した
モダナイゼーションの実現

Amazon Corretto

- Amazon がサポートする OpenJDK ディストリビューション
- Amazon 内部の数千のサービスで稼働
- マルチプラットフォーム: Linux, Windows, macOS, Alpine Linux, Amazon Linux 2, Docker, etc.
- 無料: 有料の追加機能や制限はなし
- 無料の長期サポート(LTS): パフォーマンス向上, セキュリティ更新
- Corretto 8, 11, 17
- オープンソース – 全てのパッチや機能強化は Open JDK へ貢献: https://github.com/corretto

サポート期限はFAQ参照
https://aws.amazon.com/jp/corretto/faqs/

モダナイゼーション・パスウェイ for Java

効果的なモダナイゼーションポイント

No	カテゴリ	AS-IS	TO-BE
1	Java, JDK	• 実行環境に制約がある • 追加でサポートコストがかかる	• 制約や追加コストを低減できる JDKディストリビューションを採用
2	ミドルウェア	• 商用製品で固められている	• OSSプロダクトを活用
3	ITアーキテクチャー	• リソースが余っている • サーバーが状態をもっている (スケールアウトしにくい) • モジュールが複雑化していて手が入れにくい	• 必要な分だけコストパフォーマンスに優れたリソースを使う • スケールしやすいアーキテクチャーにする • モジュールを整理、分離する
4	開発, 維持運用	• 手作業が多い	• 定型作業を自動化・省力化

モダナイゼーションの実現方法例

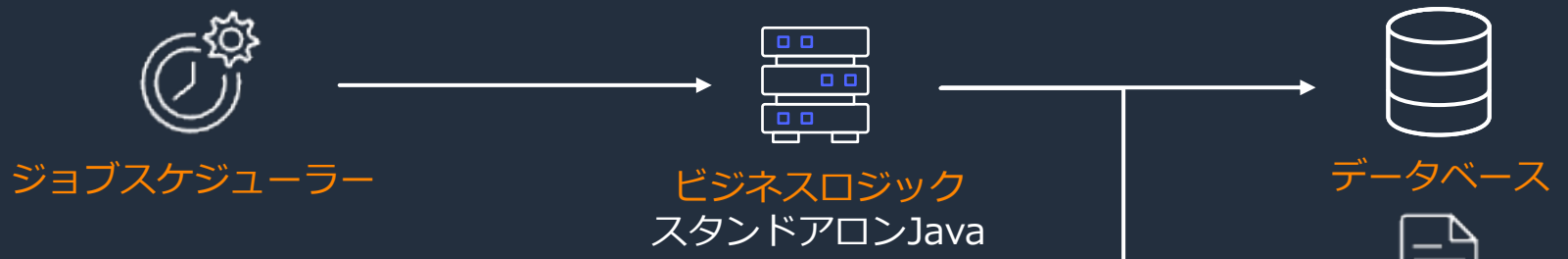
No	カテゴリ	TO-BE	実現方法例
1	Java, JDK	• 制約や追加コストを低減できるJDKディストリビューションを採用	• Amazon Corretto を採用
2	ミドルウェア	• OSSプロダクトを活用	• Spring Framework や Tomcat などのOSSプロダクトを活用
3	ITアーキテクチャー	• 必要な分だけコストパフォーマンスに優れたリソースを使う • スケールしやすいアーキテクチャーにする • モジュールを整理、分離する	• インスタンスサイズ適正化、リザーブドインスタンス、Savings Plans、Gravitonインスタンス活用 • サーバーレス(Lambda, Fargate等) の活用 • ファイルをS3、セッション情報をElastiCache、それぞれ外部に出し都度取得し、サーバーをステートレス化 • 画面とロジックの分離やモジュール分割によって複雑性を分散
4	開発, 維持運用	• 定型作業を自動化・省力化	• サーバーレス化により作業負担軽減 • Codeシリーズ(Code Buildなど)でビルドやデプロイを自動化 • CloudFormationで環境構築を自動化

Javaアプリケーション モダナイズ例

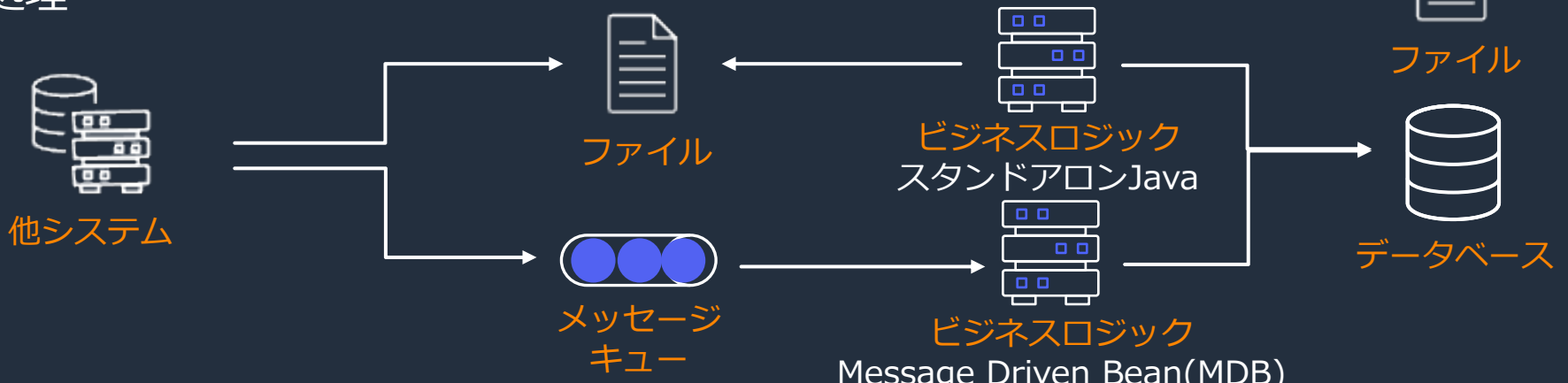
エンタープライズJavaアプリケーション アーキテクチャ例

①: オンライン処理

②:バッチ処理

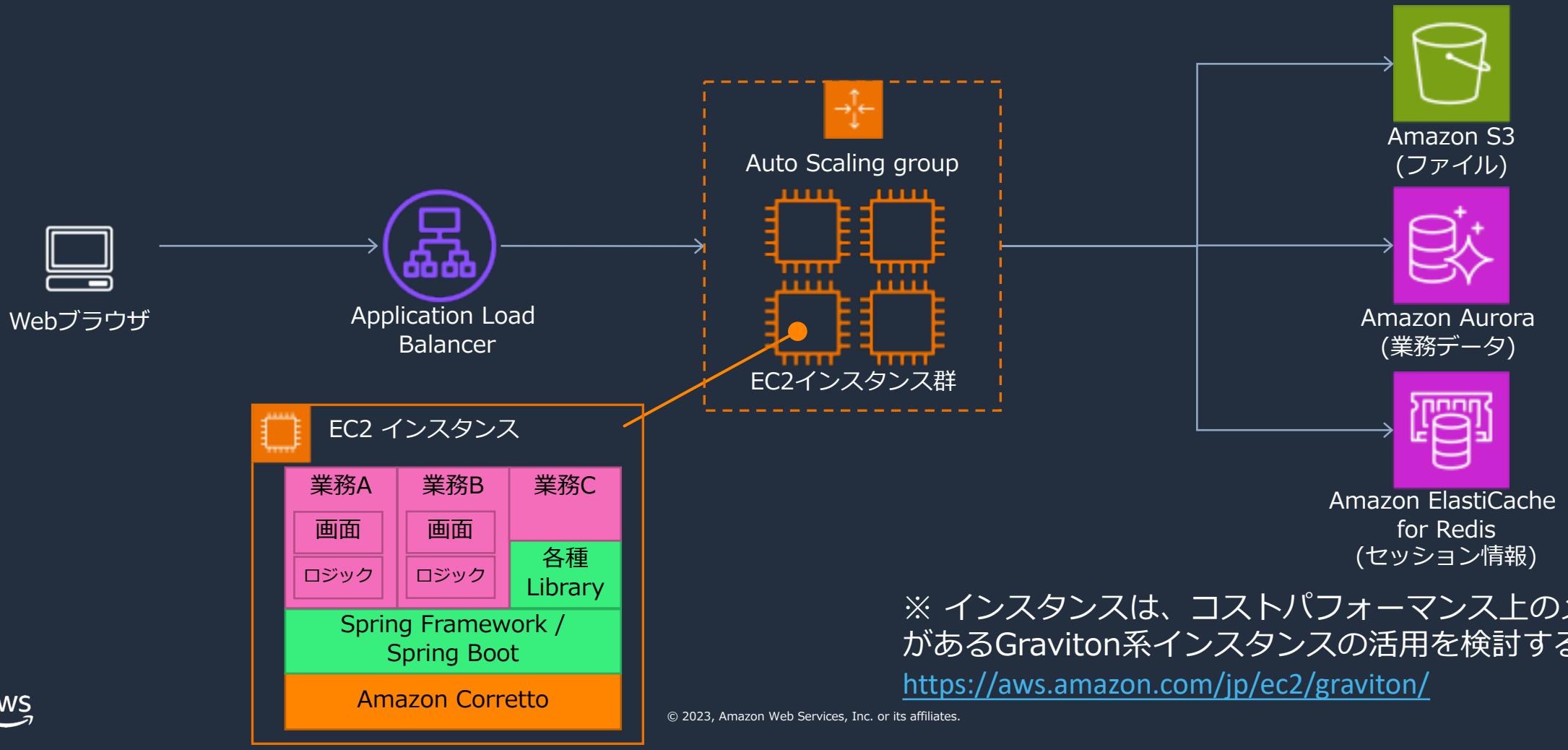


③:非同期処理



①: オンライン処理 の モダナイゼーション

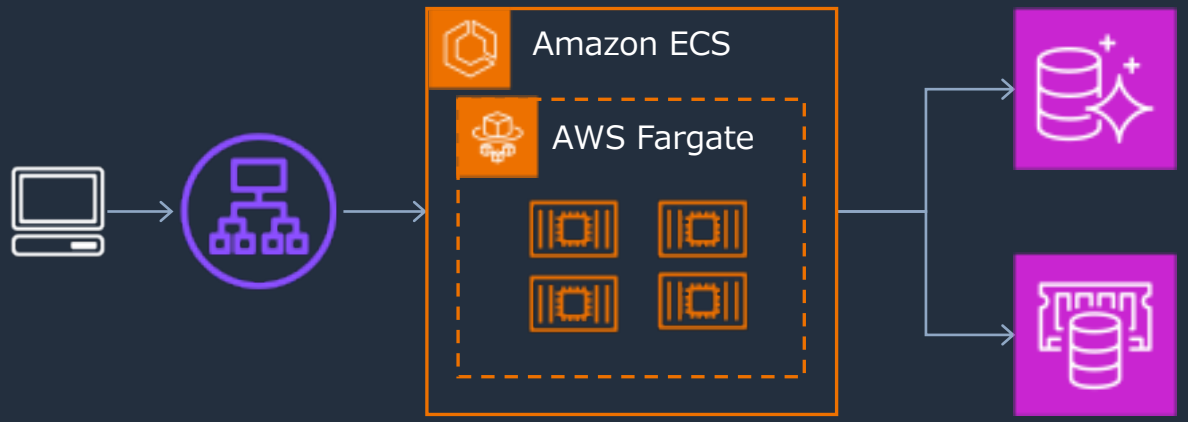
コスト効率よく、スケールアウト可能な構成に



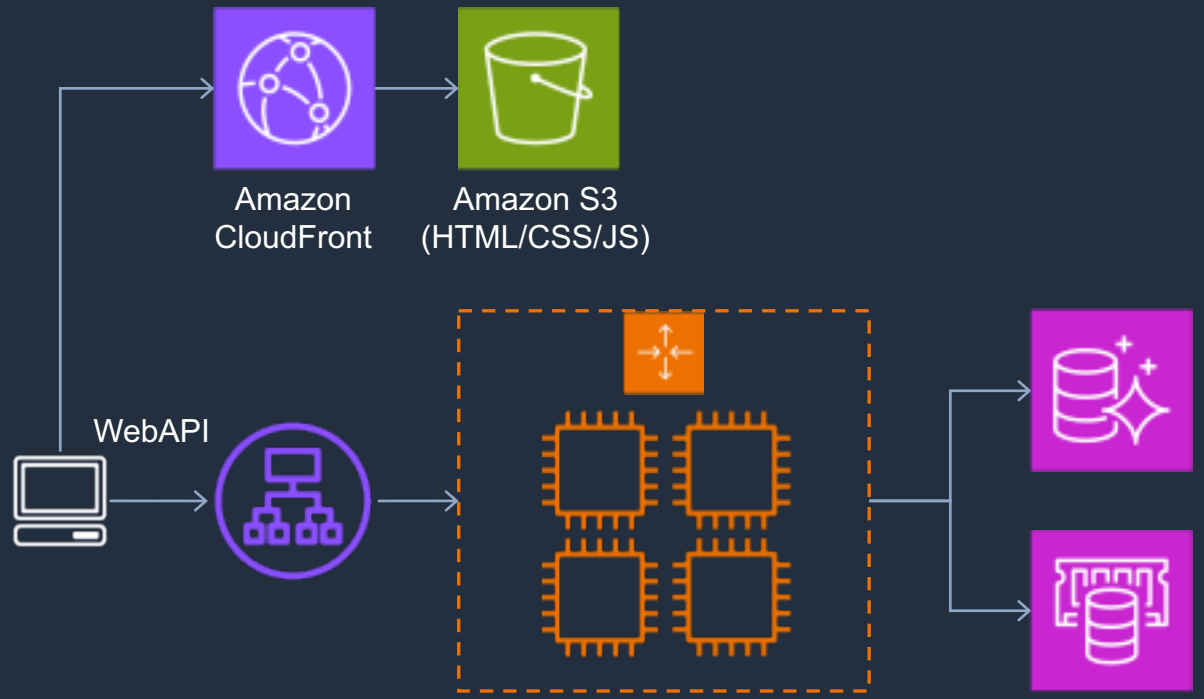
※ インスタンスは、コストパフォーマンス上のメリットがあるGraviton系インスタンスの活用を検討するとよい
https://aws.amazon.com/jp/ec2/graviton/

①: オンライン処理 の モダナイゼーション

コンテナを活用しサーバーレス化

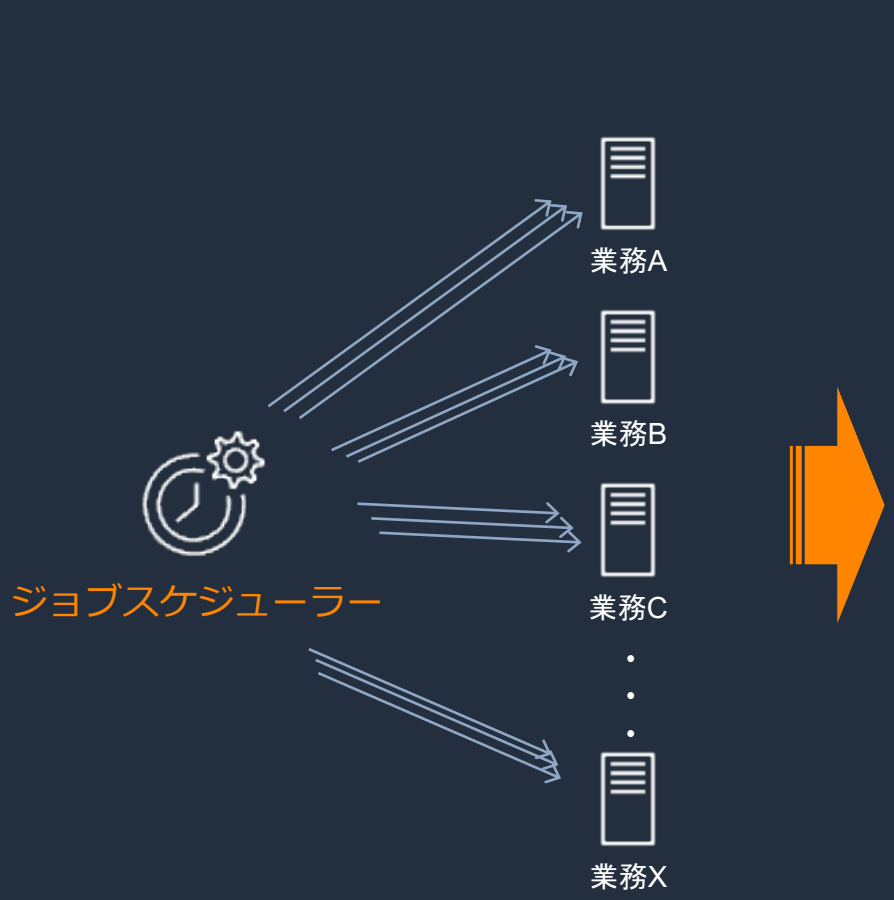


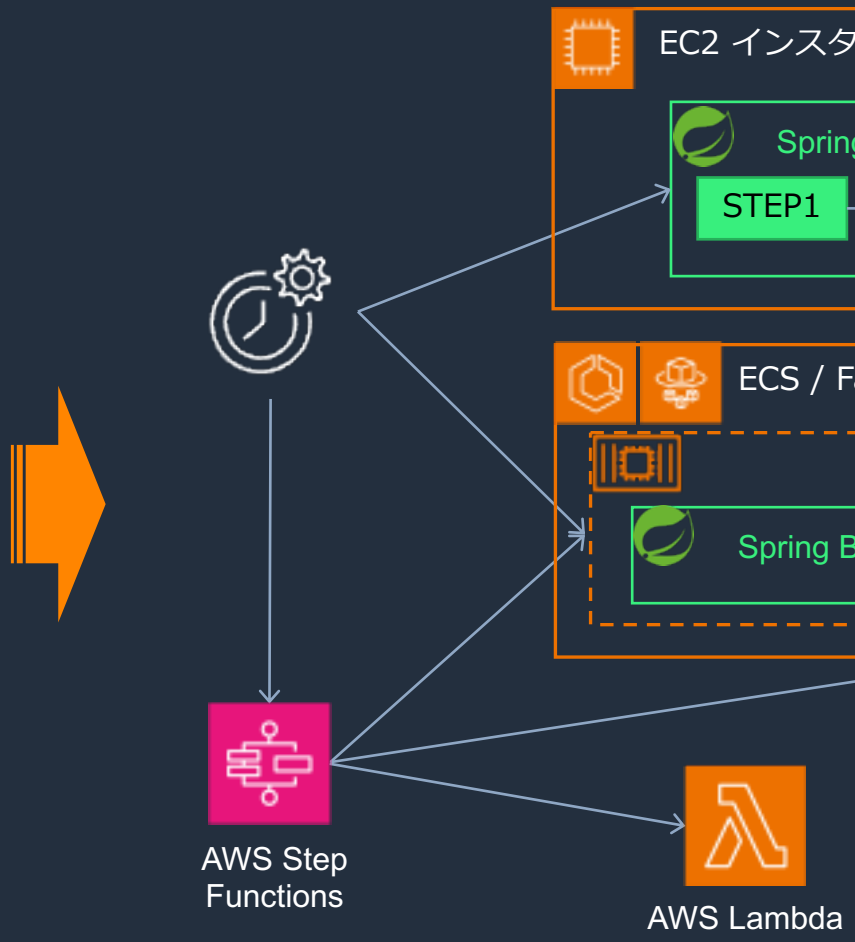
画面とロジックの分離



②: バッチ処理 の モダナイゼーション

フレームワークやサービスへ複雑性を分散



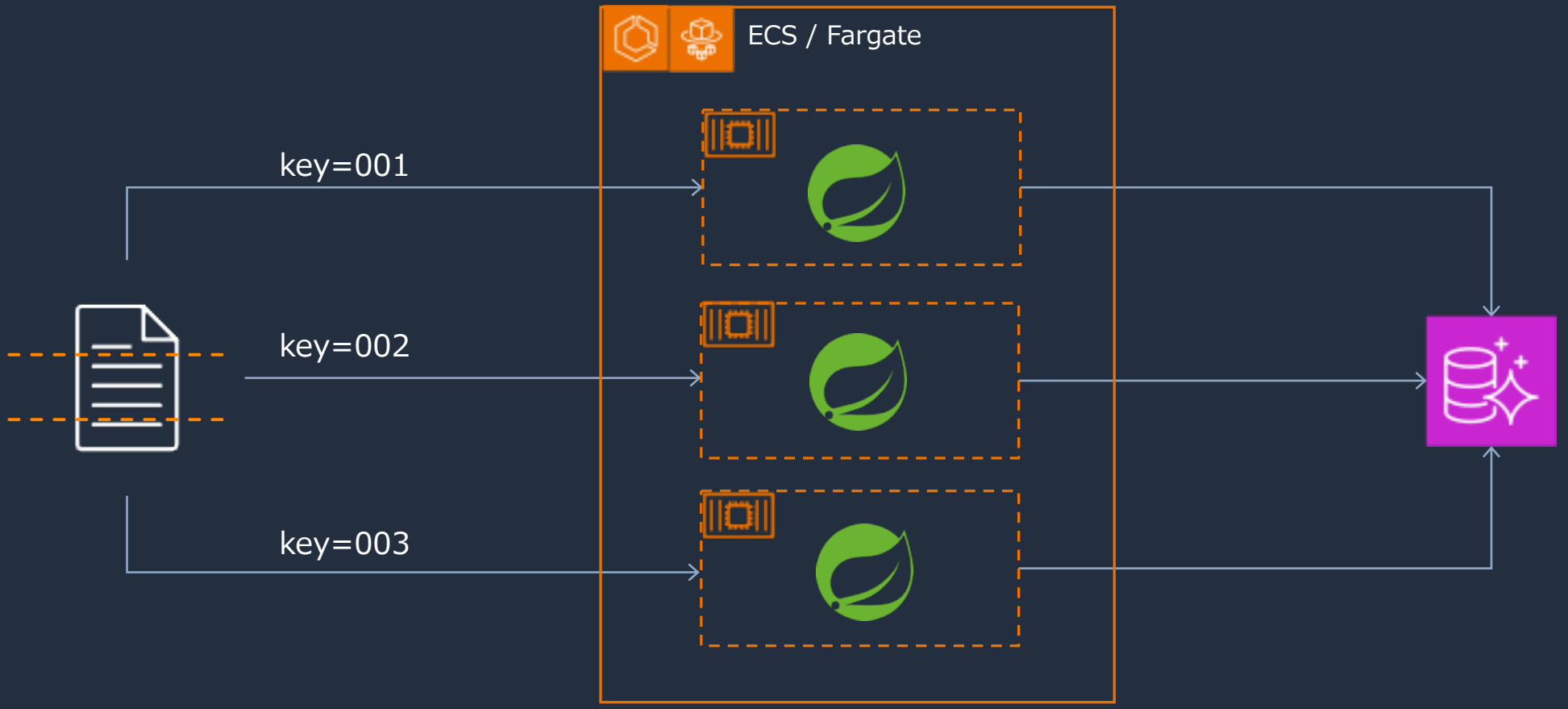


リソース ＼ 実行時間	15分以内	15分超過
10GB以内	Lambda	ECS/Fargate
16vCPU 120GB以内	ECS/Fargate	
16vCPU 120GB超過	EC2	

使い分け例

②: バッチ処理 の モダナイゼーション

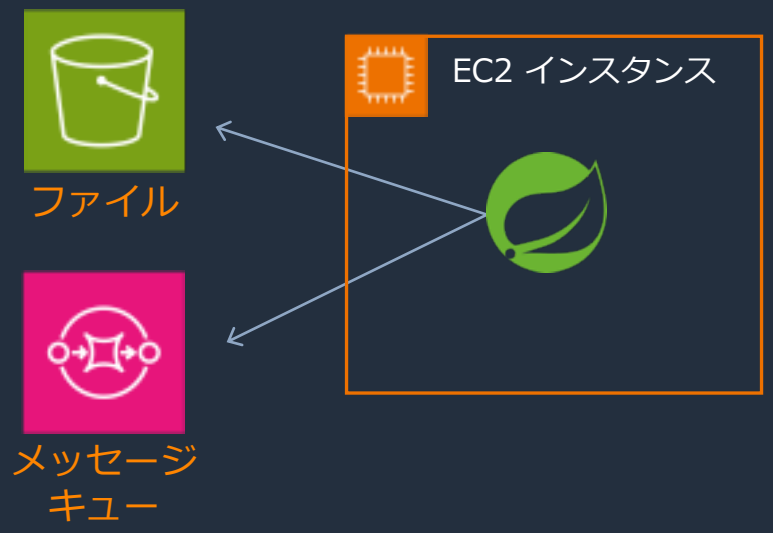
大量データ処理をスケールしやすく



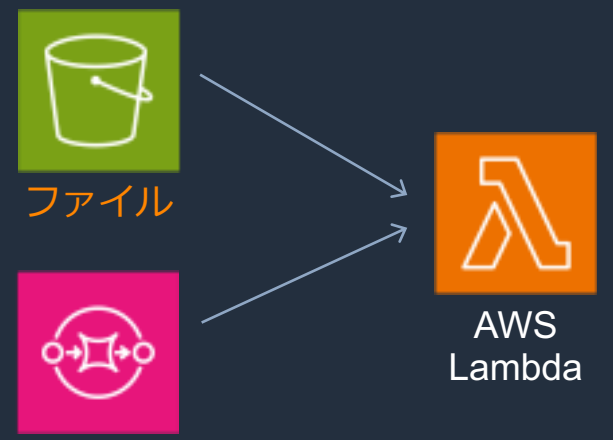
③: 非同期処理 の モダナイゼーション

OSSプロダクトやサーバーレスを活用

フレームワークが変更を監視する



変更イベント契機で処理を起動する



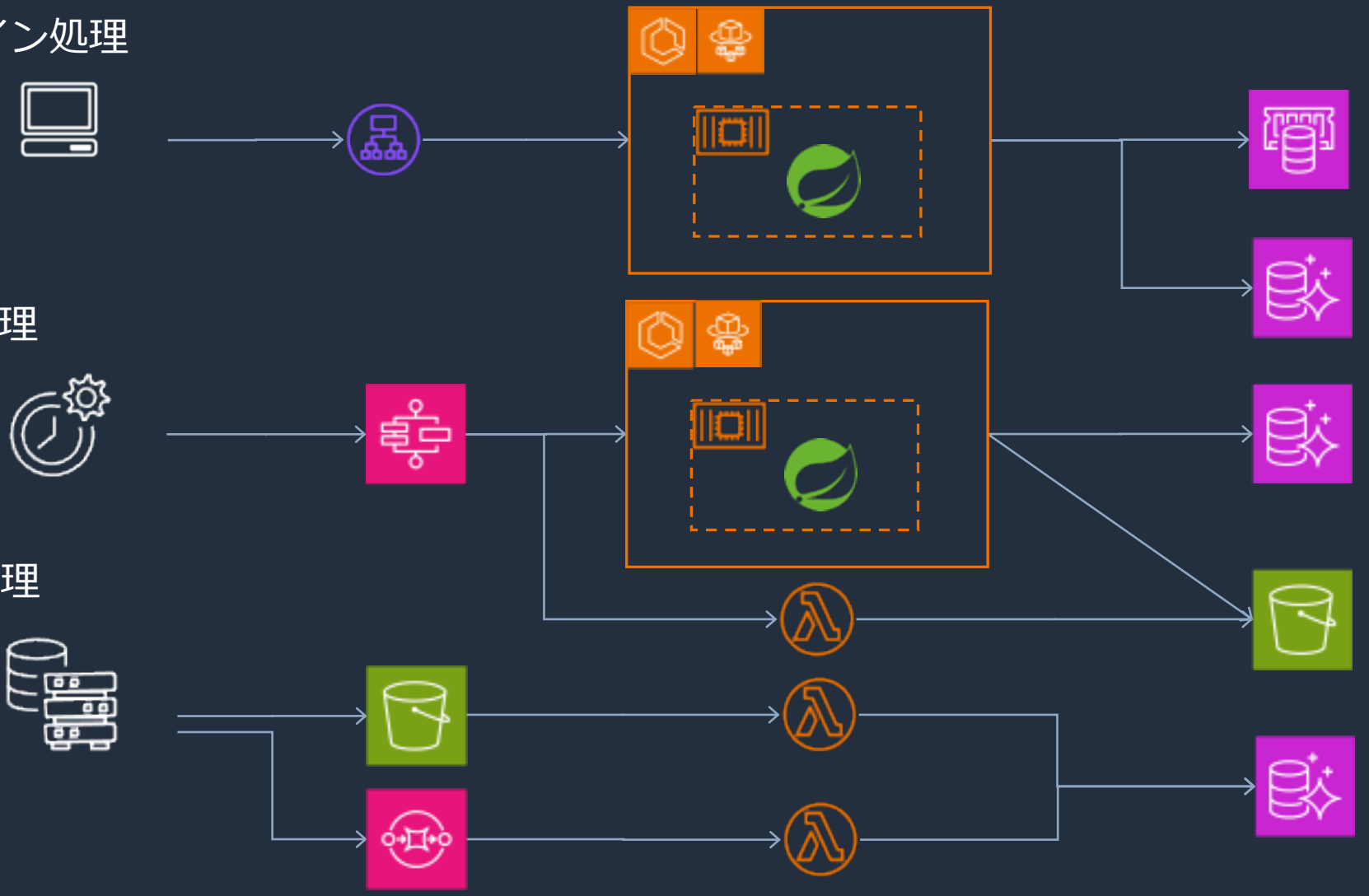
※ Lambdaのコールドスタートは
Provisioned Concurrency や
Lambda SnapStart 等で対処

モダナイズ後のアーキテクチャ例

①: オンライン処理

②:バッチ処理

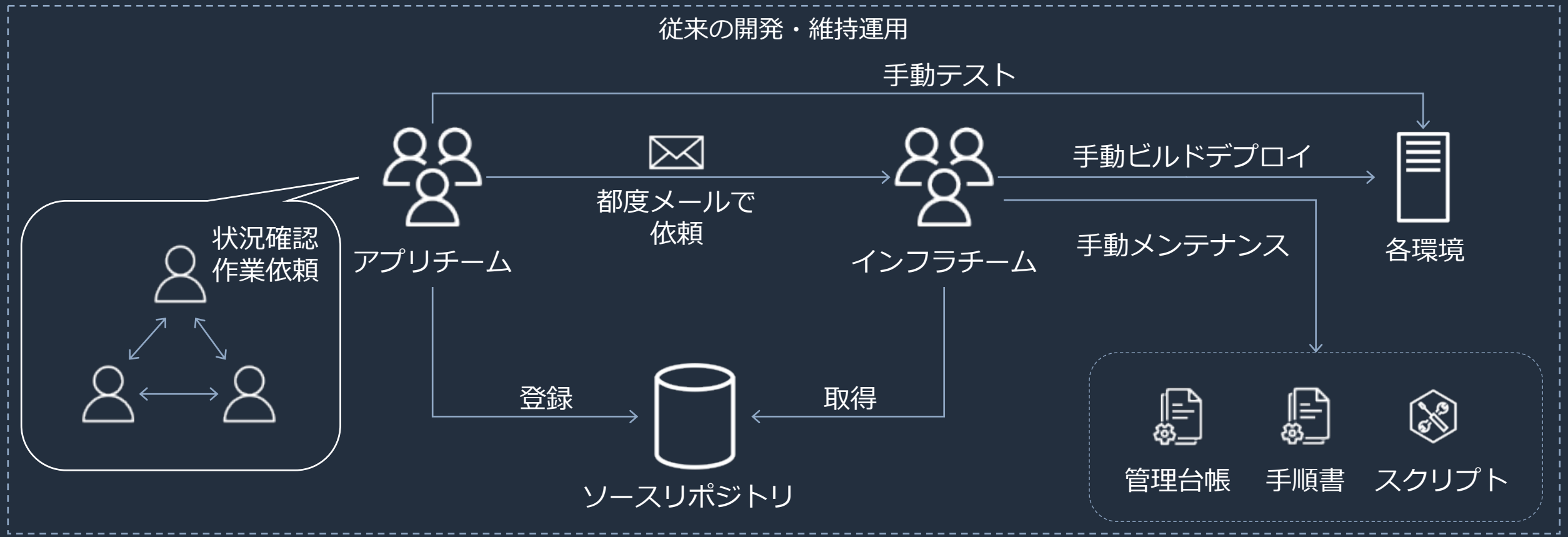
③:非同期処理



Javaアプリケーション開発
モダナイズ例

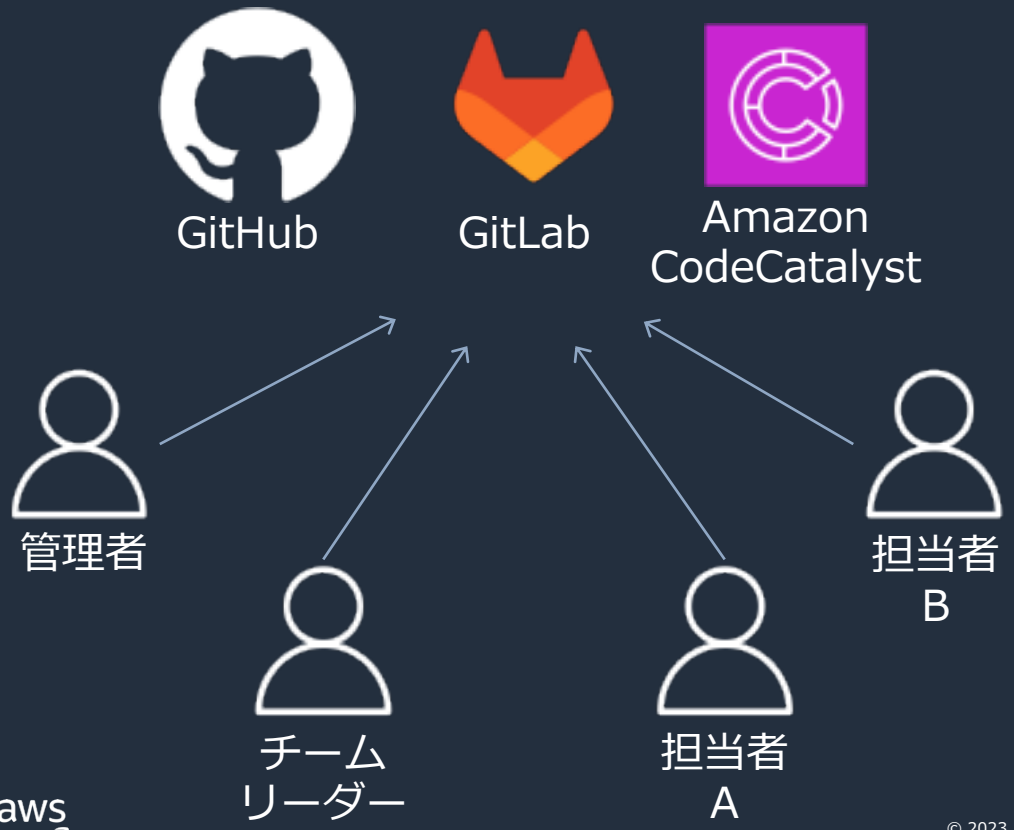
開発や維持運用を支えるには

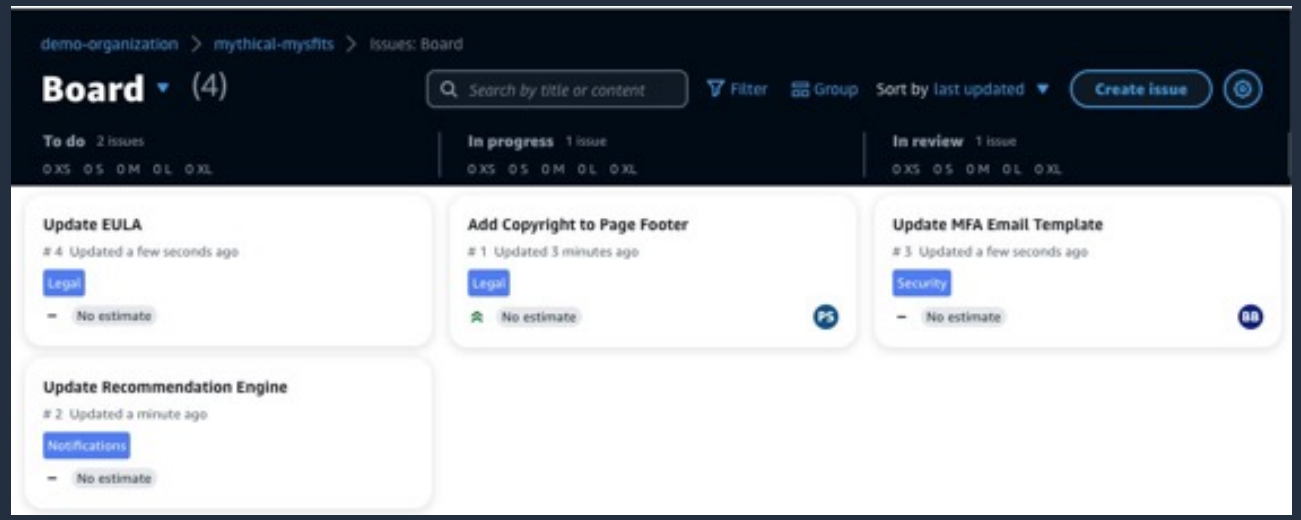
アーキテクチャーや要素技術のモダナイゼーションを支えるために、
開発や維持運用の仕組みもモダナイゼーションする必要がある



作業状況の可視化

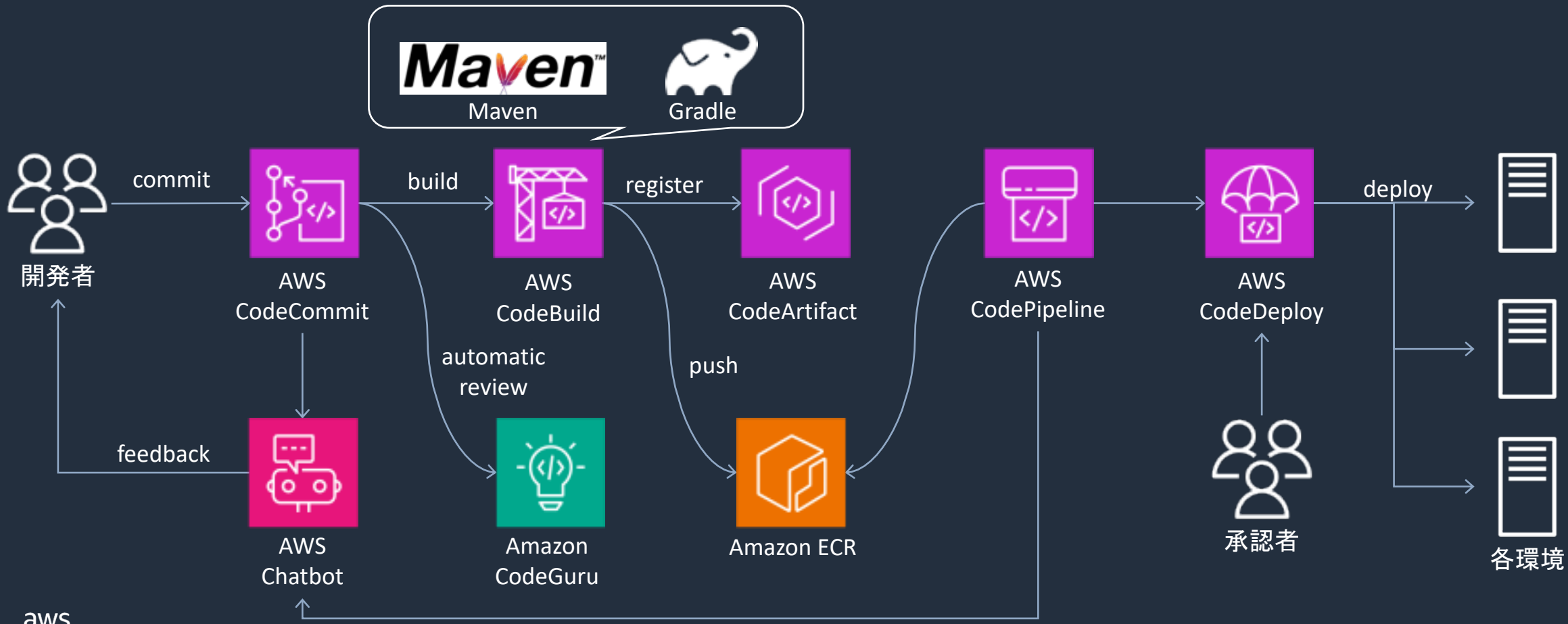
タスクや進捗の情報を開発プラットーフォーム上で可視化





開発上の定型作業を自動化

可能な限り自動化・省力化
人手を排除しボトルネックになりにくい仕組みを作る



運用の効率化

可観測性(オブザーバビリティ)を確保し運用しやすく

メトリクス

ログ

トレース

Observability

監視
SaaS

new relic.

DATADOG

AWS
サービス

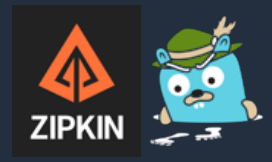
CloudWatch
Logs

CloudWatch
Metrics

X-Ray

Managed
Grafana/Prometeus

OpenSearch Service



Zipkin, Jaeger

CloudWatch
Agent

X-Ray
Agent

Instrumentation

Spring Boot
Actuator

Micrometer

AWS Distro for
OpenTelemetry

まとめ

本セッションのまとめ

- Javaアプリケーションは処理方式やシステム構成要素ごとに、モダナイゼーションできる様々なポイントがある。
- JavaのエコシステムとAWSのサービスをうまく組み合わせることで相乗効果を生み出せる。

参考資料

- AWS Black Belt

- モダナイゼーションとは
  https://www.youtube.com/watch?v=-Nu-KTsFk1A

- モダナイゼーション・パスウェイ
  https://www.youtube.com/watch?v=wyp8EOkLUpk

AWS Black Belt Online Seminar とは

- 「サービス別」「ソリューション別」「業種別」などのテーマに分け、アマゾン ウェブ サービス ジャパン合同会社が提供するオンラインセミナーシリーズです
- AWS の技術担当者が、 AWS の各サービスやソリューションについてテーマごとに動画を公開します
- 以下の URL より、過去のセミナー含めた資料などをダウンロードすることができます
  - https://aws.amazon.com/jp/aws-jp-introduction/aws-jp-webinar-service-cut/
  - https://www.youtube.com/playlist?list=PLzWGOASvSx6FIwIC2X1nObr1KcMCBBlqY

ご感想は Twitter へ！ハッシュタグは以下をご利用ください
#awsblackbelt

内容についての注意点

- 本資料では 2023 年 7 月時点のサービス内容および価格についてご説明しています。AWS のサービスは常にアップデートを続けているため、最新の情報は AWS 公式ウェブサイト (https://aws.amazon.com/) にてご確認ください
- 資料作成には十分注意しておりますが、資料内の価格と AWS 公式ウェブサイト記載の価格に相違があった場合、AWS 公式ウェブサイトの価格を優先とさせていただきます
- 価格は税抜表記となっています。日本居住者のお客様には別途消費税をご請求させていただきます
- 技術的な内容に関しましては、有料の AWS サポート窓口へお問い合わせください
- 料金面でのお問い合わせに関しましては、カスタマーサポート窓口へお問い合わせください (マネジメントコンソールへのログインが必要です)

Thank you!