超分散システムを統治せよ！
マイクロサービス開発・展開のエッセンス

日本アイ・ビー・エム株式会社
IBMクラウド事業本部
エグゼクティブITスペシャリスト

樽澤 広亨

自己紹介

IBM クラウド事業本部 アーキテクト

ISO IEC JTC1/SC38 専門委員

クラウド国際標準策定に従事

過去

米IBM ソフトウェア開発研究所

ソフトウェア製品開発エンジニア

IBM ソフトウェア・グループ

エバンジェリスト, 技術営業, etc.

このセッションについて

目的

- マイクロサービス適用時の設計のポイントを知る
- サービス・メッシュ最適化の方策を知る

内容

- ✓ マイクロサービス概要
- ✓ サービスの設計
- ✓ Istioによるサービス・メッシュ最適化
- ✓ まとめ

マイクロサービス概要

マイクロサービス概要

クラウド・ネイティブ・アプリケーション開発・運用のベスト・プラクティス

動機

- アプリケーション個別の保守を可能にする柔軟なモジュラー構造(サービス)の実現

対象分野

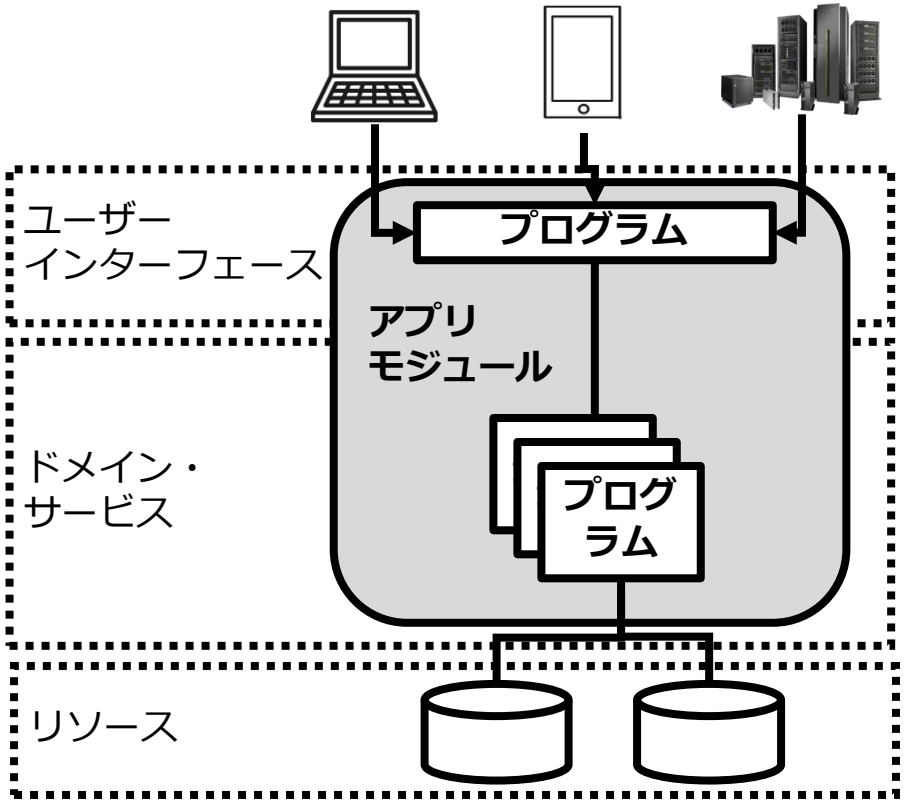
- アーキテクチャー
- 開発・運用チーム・フォーメーション
- システム・ライフサイクル

マイクロサービス・アークテクチャー

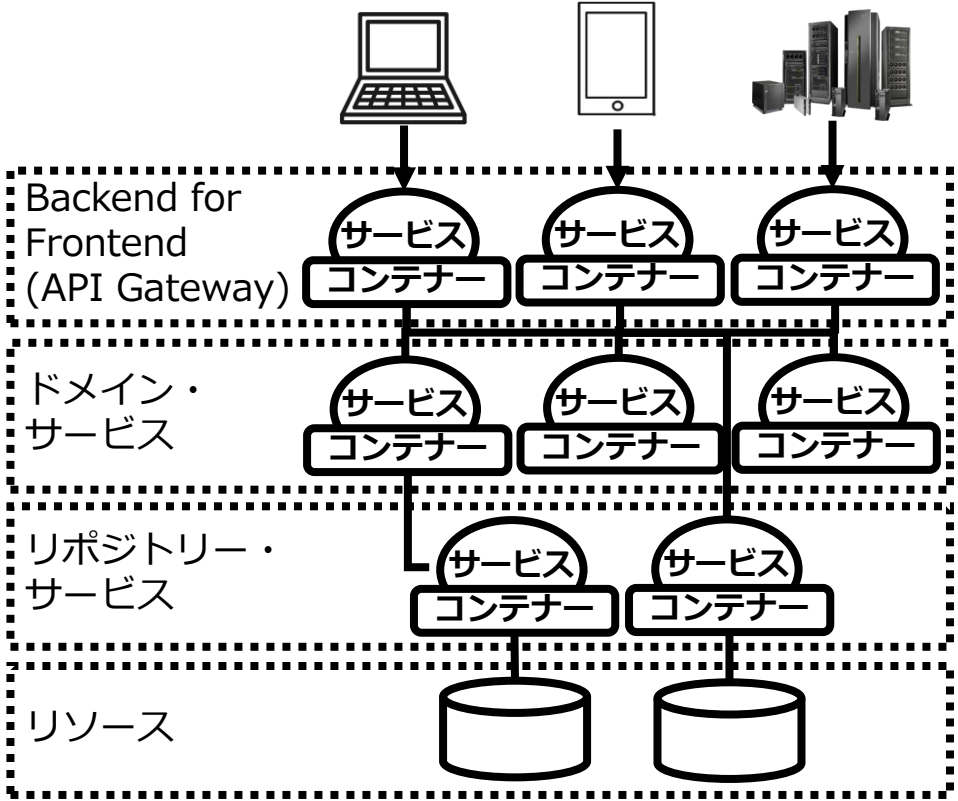
- **小さなサービスの組合せ**でアプリを構成
- 各サービスは**独立したコンテナで稼動**
- 各サービスは**軽量プロトコルで連携**
- 各サービスはそれぞれ**個別に自動デプロイ**
- 各サービスはそれぞれ**最適なプログラミング言語で実装**
- 各サービスはそれぞれ**最適なデータ・ストレージを利用**

マイクロサービス・アーキテクチャー全体像

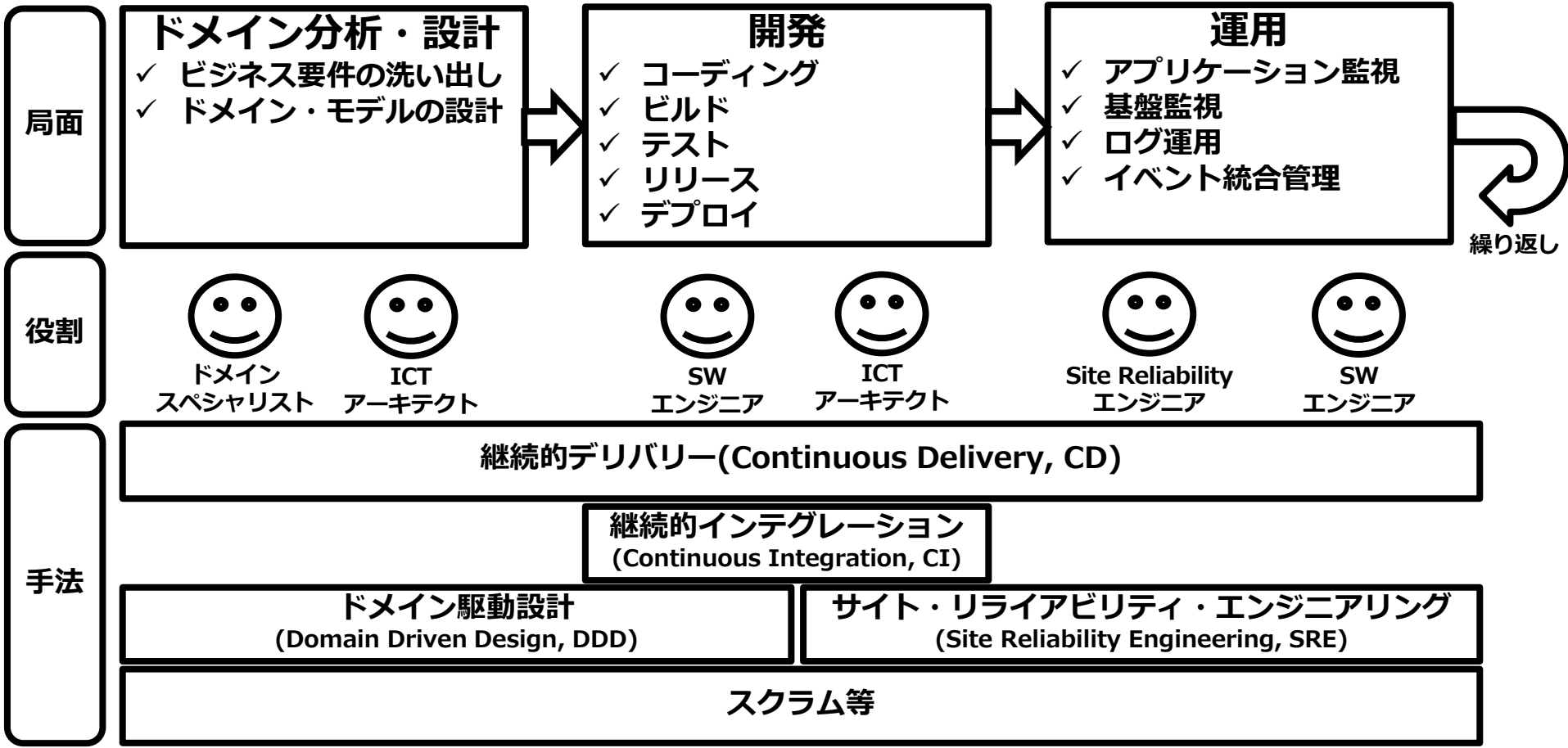
モノリス (1枚岩)アプリケーション構造



マイクロサービスに基づいたアプリケーション構造

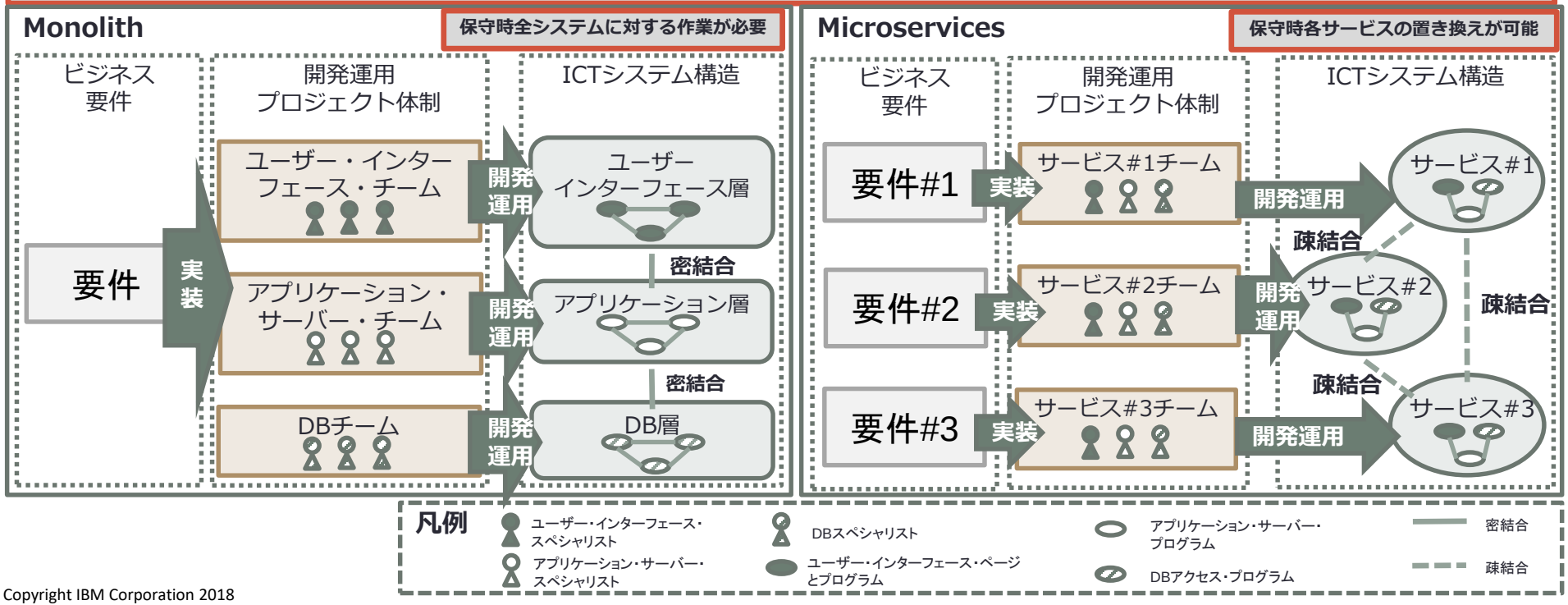


クラウド・ネイティブ・コンピューティング：開発・運用の流れと役割

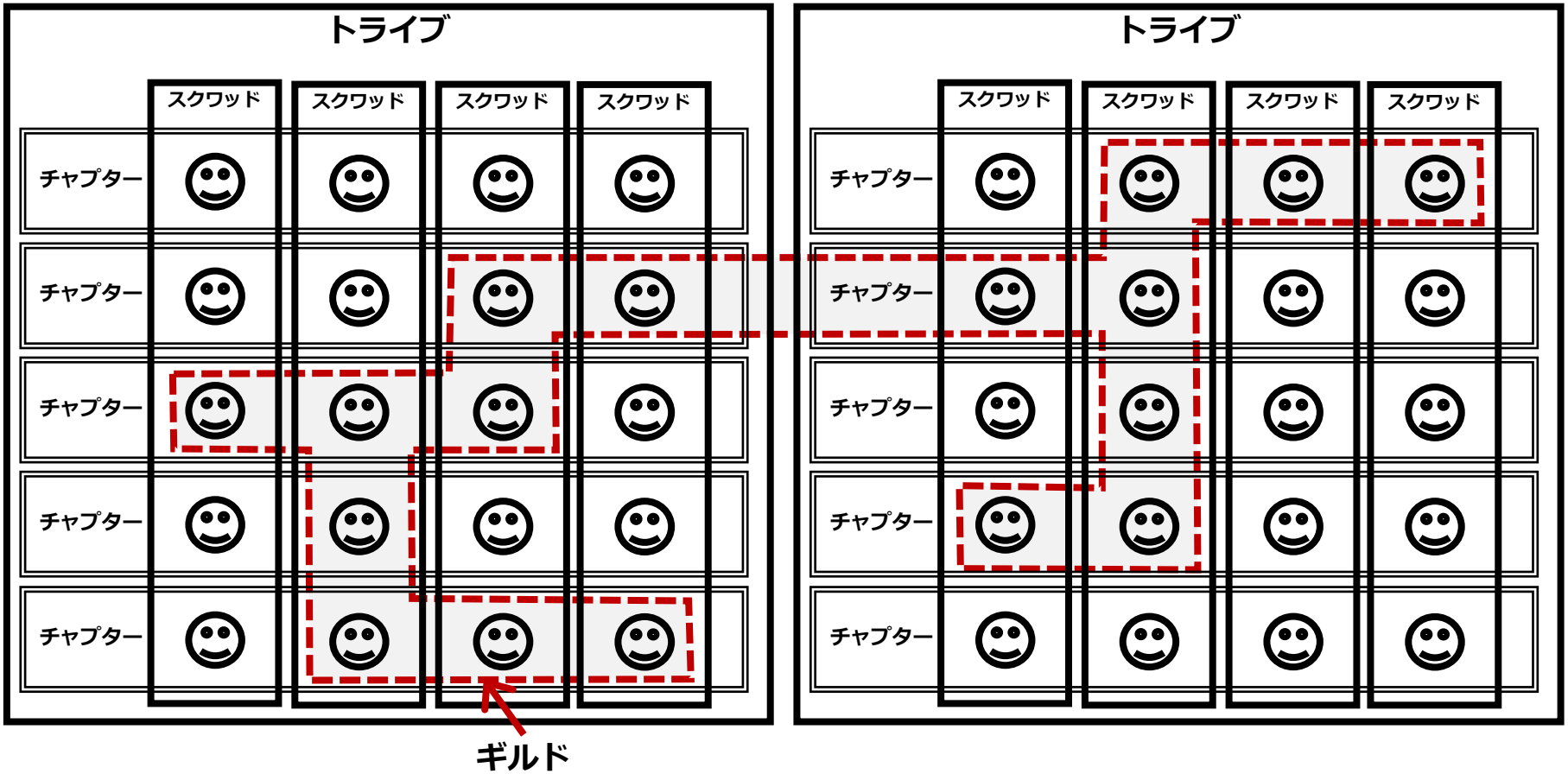


サービスと開発・運用チーム・フォーメーションの関係

- コンウェイの法則：システム構造は, プロジェクト体制を反映する
- マイクロサービス：
  - ビジネス目的に基づいてチーム編成
  - システムは、独立して置き換え可能なサービスで構成される



Spotifyのチーム編成

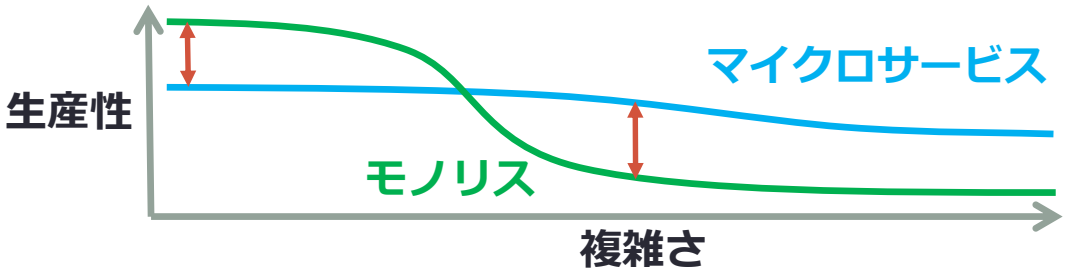


マイクロサービス、始める？始めない？

基本方針
対象システムが複雑：マイクロサービス化を検討する
対象システムが単純：マイクロサービス化には向いていない

■単純なシステムのマイクロサービス化は、高いコストに帰結
～”**MicroservicePremium**”

●自動デプロイメント, 分散システム運用監視, 結果整合性, ビジネス分析, …



マイクロサービス適用に先立つ懸案事項

マイクロサービス適用の結果…超分散システム化 !?

- 様々な懸案の解決が求められる

◆サービス設計

- ✓構造
- ✓粒度～サービス化のすすめ方
- ✓データ・モデル
- ✓トランザクション

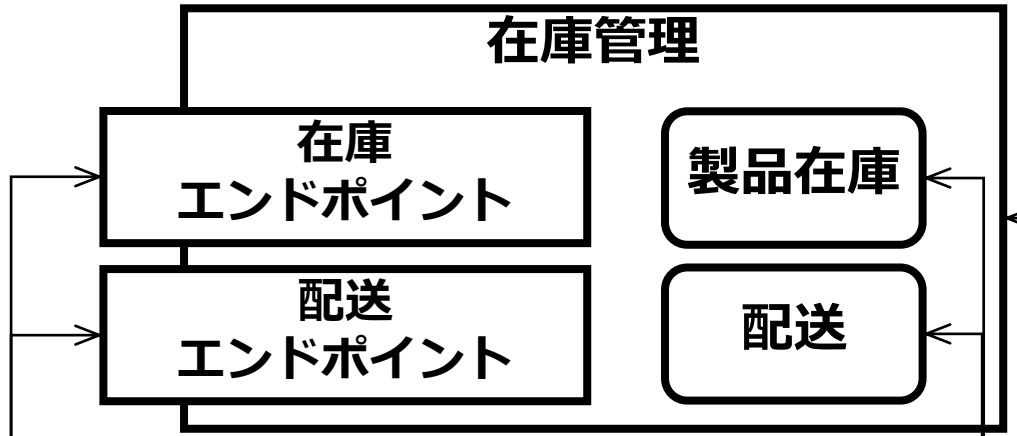
◆サービス・メッシュ最適化

- ✓ルーティング
- ✓サービスのリリース
- ✓テスト
- ✓耐障害性
- ✓テレメトリー

サービスの設計

サービスの構造

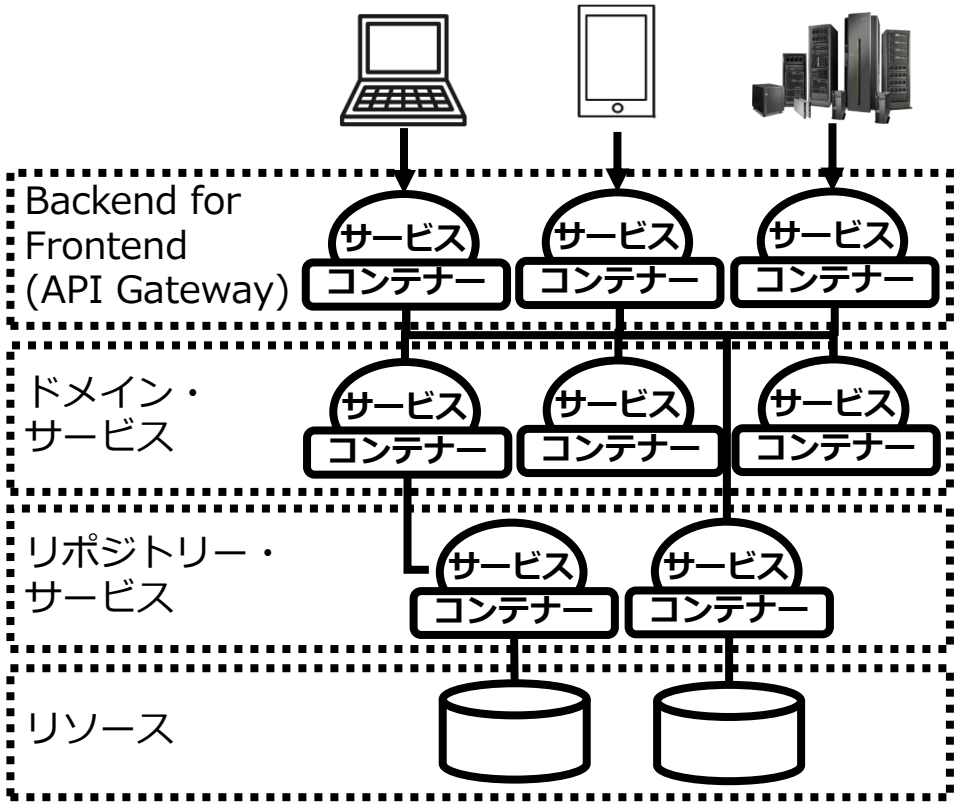
サービス
実装すべきビジネス要件
✓ インターフェースとモデルを包含する



インターフェース
プログラミング・インターフェース

モデル
ビジネス・データ

サービスのレイヤー構造例



- ●ゲートウェイ
- ●Façade

✓APIエンド・ポイント
✓データ/プロトコル変換
✓サービス・アグリゲーション

- ●ビジネス・ロジック

✓業務処理実装

- ●リソース・アクセス

✓データ, メッセージングへのアクセス

サービス化の進め方：ビジネス指向と繰り返し開発

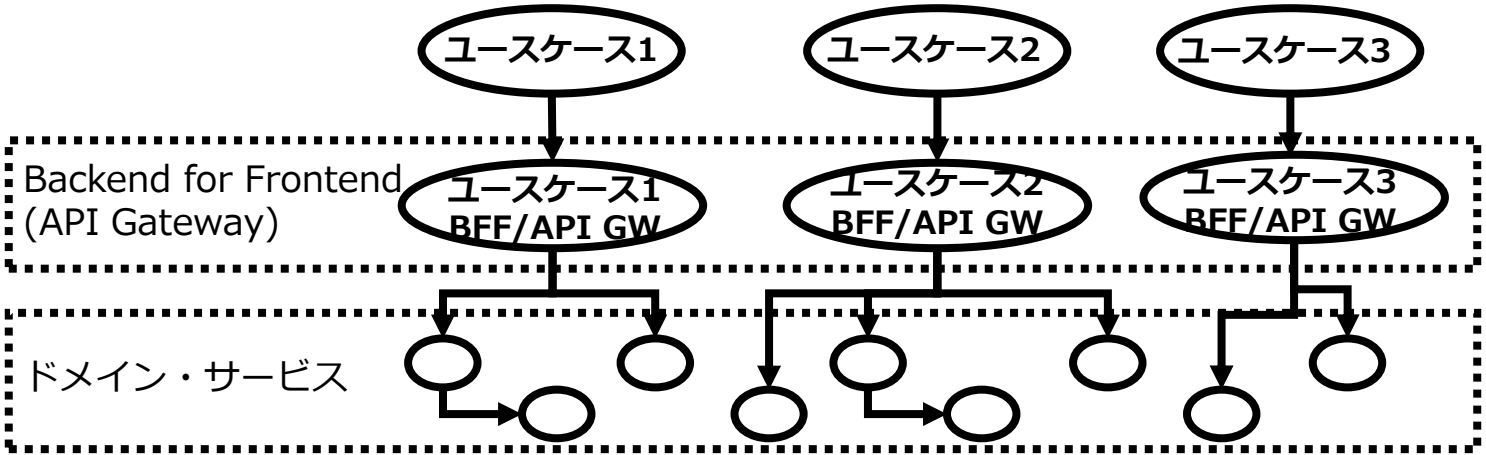
前提/文脈	● 素早いビジネス立上げ ● 柔軟なビジネス変更	● 1回の分析・設計で サービス最適化は不可能
方針	ビジネス指向	開発・運用の 繰り返し
方策	✓ 業務観点でサービスの 切り出し ✓ ドメイン駆動設計	✓ アジャイル開発
期待される効果	ビジネスのローンチ&トランスフォーメーション	サービス粒度の最適化

サービス化の進め方：BFF/API Gatewayの粒度

- Backend for Frontend (BFF)
  - ◆サービス・アグリゲーション
- API Gateway (API GW)
  - ◆APIエンドポイント
  - ◆データ＆プロトコル変換
- 2つの観点より設置単位を検討
  - ◆ユースケース
  - ◆クライアントのタイプあるいは種類

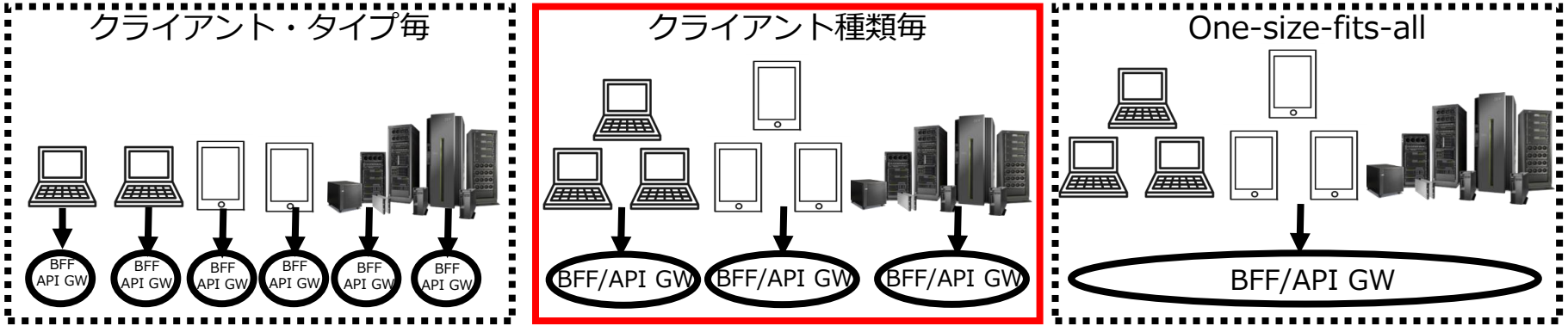
サービス化の進め方：BFF/API Gatewayの粒度

- ユースケースの観点での検討
  - ◆1ユースケース毎に、1 BFF/API GWを推奨
  - ◆複数ユースケースに1 BFF/API GWを設置した場合、粒度が大きくなり、メンテナンスのボトルネックとなる



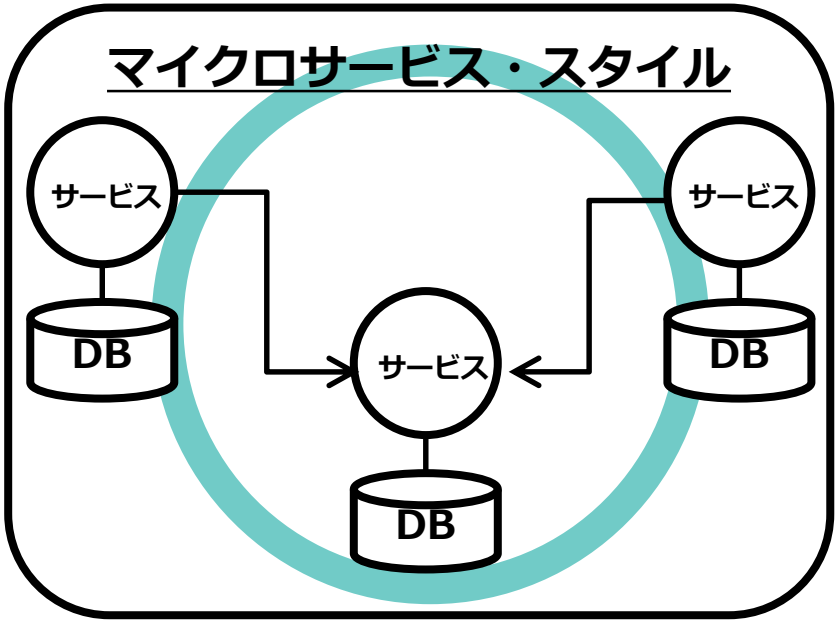
サービス化の進め方：BFF/API Gatewayの粒度

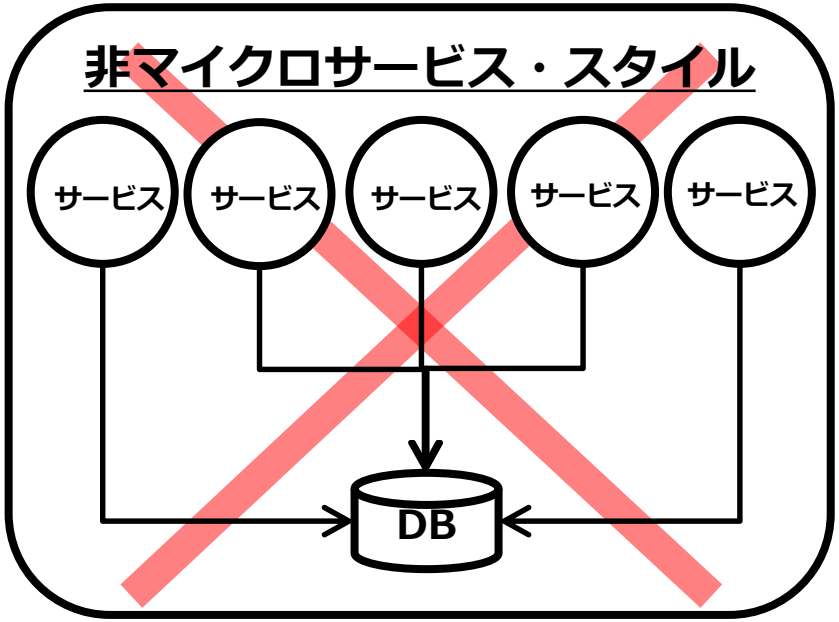
- クライアントのタイプあるいは種類の観点での検討
  - ◆原則クライアント種類毎に1 BFF/API GW設置を推奨
  - ◆クライアント・タイプ毎に1 BFF/API GW設置した場合
    - ✓BFF/API GWの種類が多くなりすぎ、メンテナンスのボトルネックになる懸念
  - ◆One-size-fits-allの場合
    - ✓BFF/API GWが大きくなりすぎ、メンテナンスのボトルネックになる懸念



データ・アクセス

サービスを介してデータベースにアクセスする



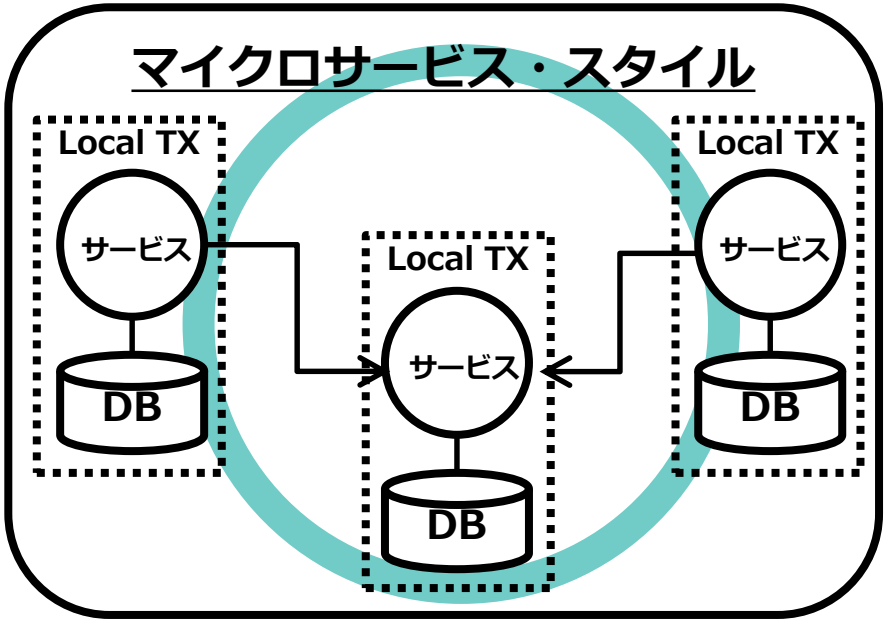


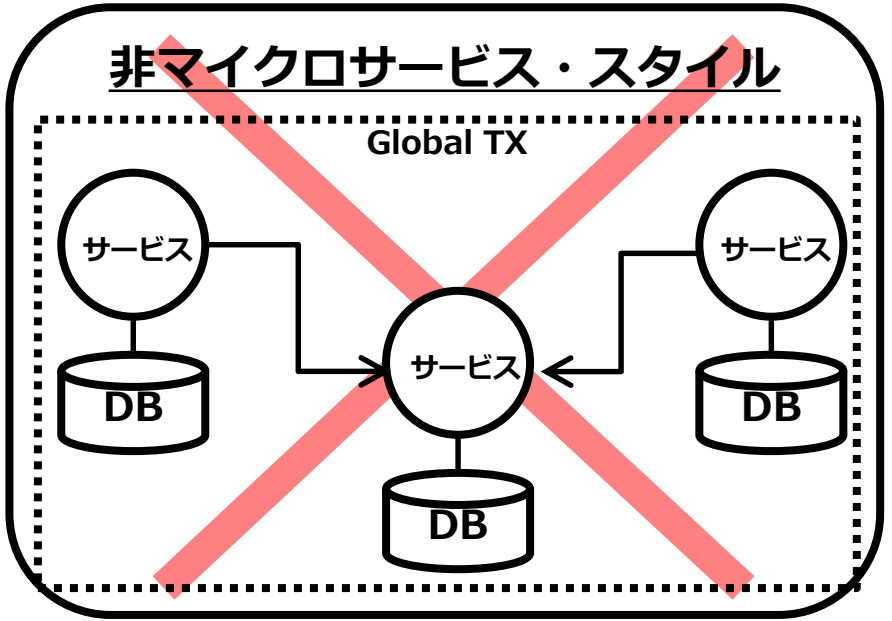
DBのメンテナンスの影響を最小化し、サービスへの影響を極力排除する

トランザクション処理

原則ローカル・トランザクション処理

Local TX：ローカル・トランザクション
Global TX：グローバル・トランザクション

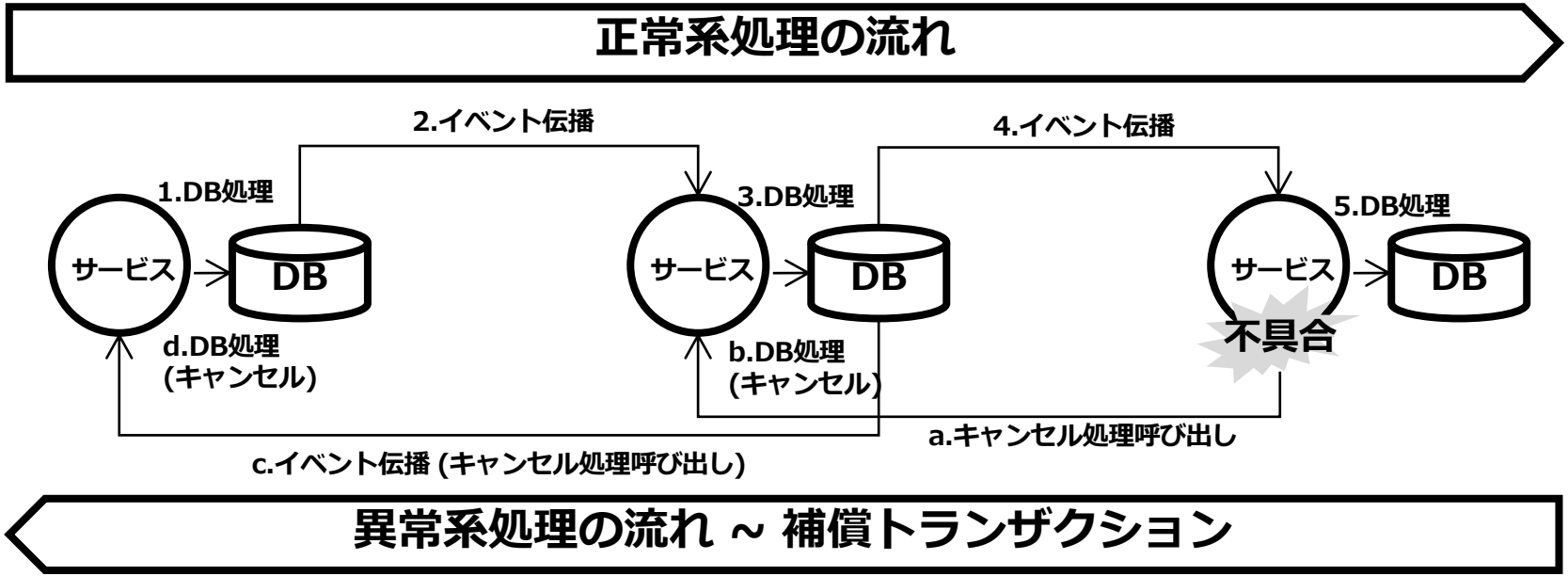




- 実装をシンプルに保ち、スピーディな障害対応に備える
- サービス間の疎結合を維持する

トランザクション処理：複数リソースの同期処理

- Saga パターン
  - ◆複数リソースの同期を取るデザイン・パターン
  - ◆ローカル・トランザクション, イベント, 補償トランザクションを活用

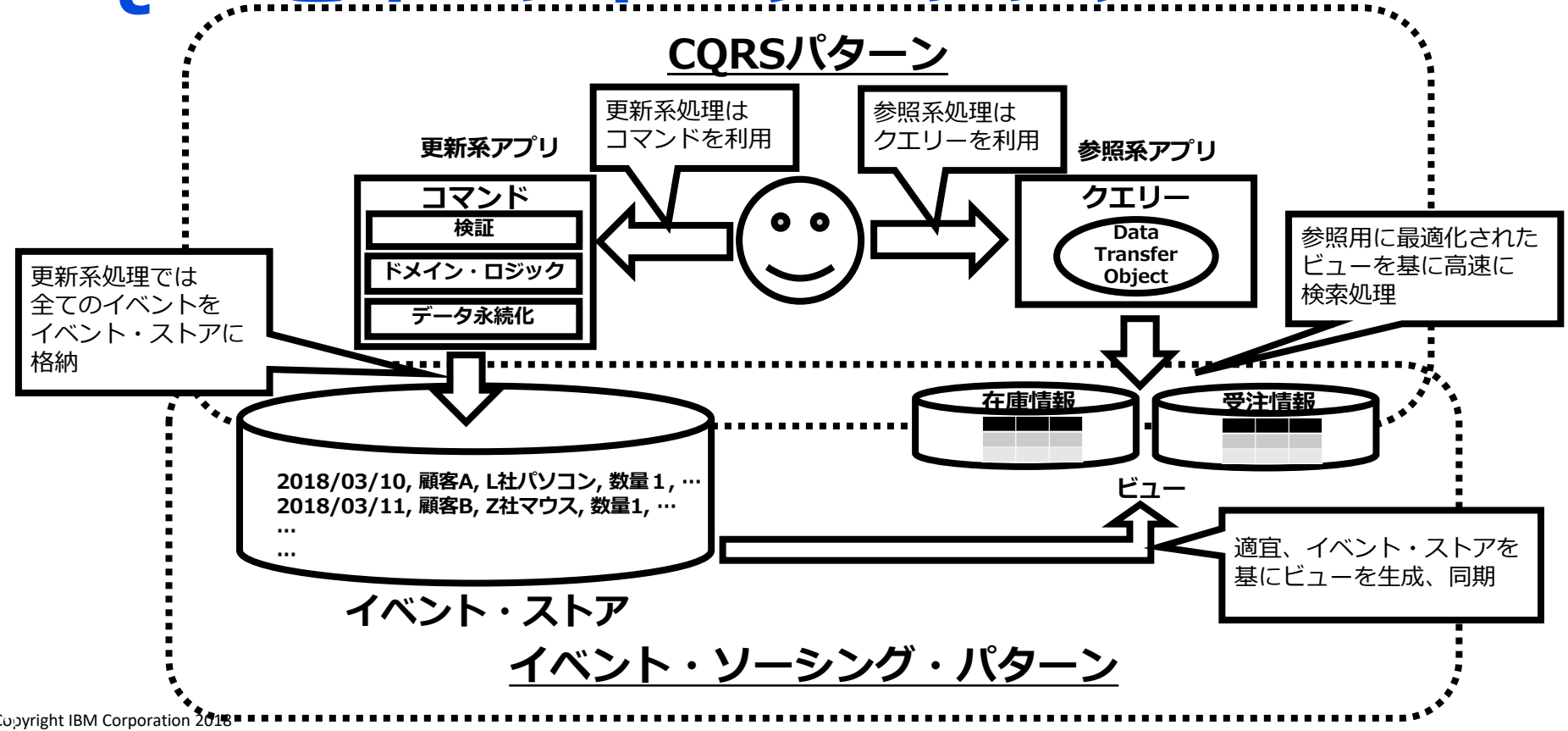


サーバーレス(FaaS)のイベント駆動処理に最適な処理パターン…

データ＆トランザクション処理：CQRSとイベント・ソーシング

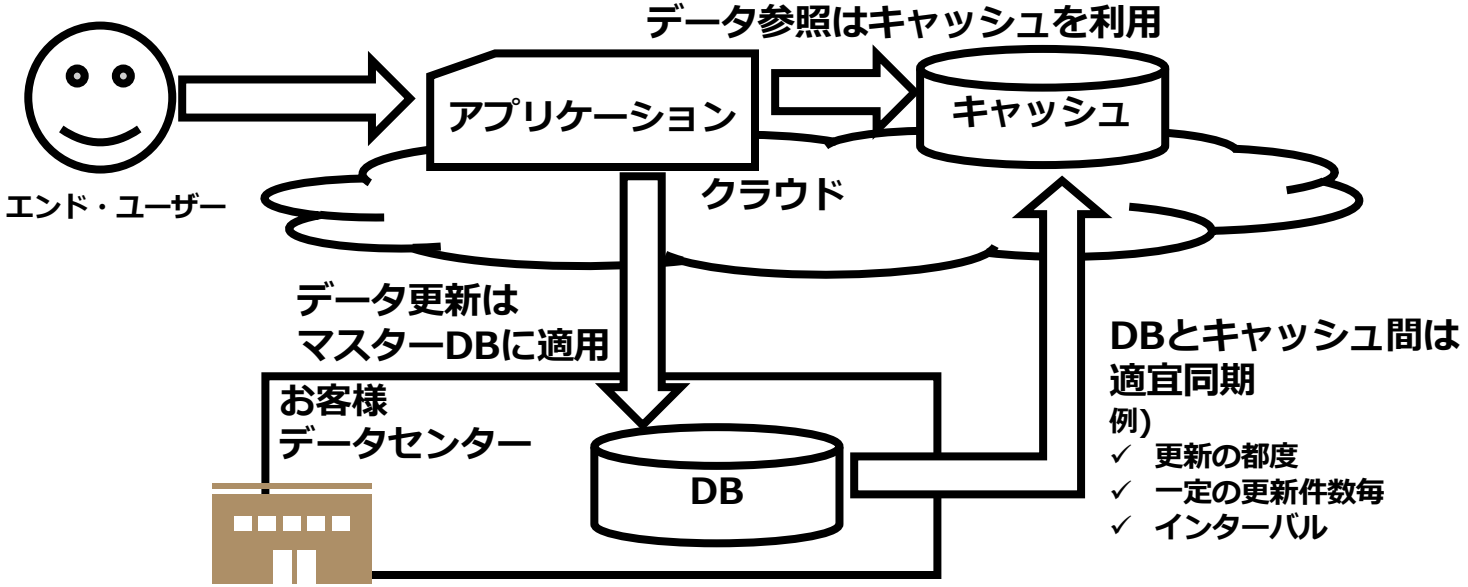
- Command Query Responsibility Segregation
  - ◆コマンド(データ更新)とクエリ(データ参照)に、異なるデータ・モデルを利用
- イベント・ソーシング
  - ◆一連の業務情報をイベント・ストアに書き込む
  - ◆適宜ビューを生成し参照処理に充てる

データ&トランザクション処理：CQRSとイベント・ソーシング



データ＆トランザクション処理：データをどこに配置？

- 参照系データと更新系データの分離の実装モデル



セキュア

- マスターDBを安全なお客様データセンターに配備

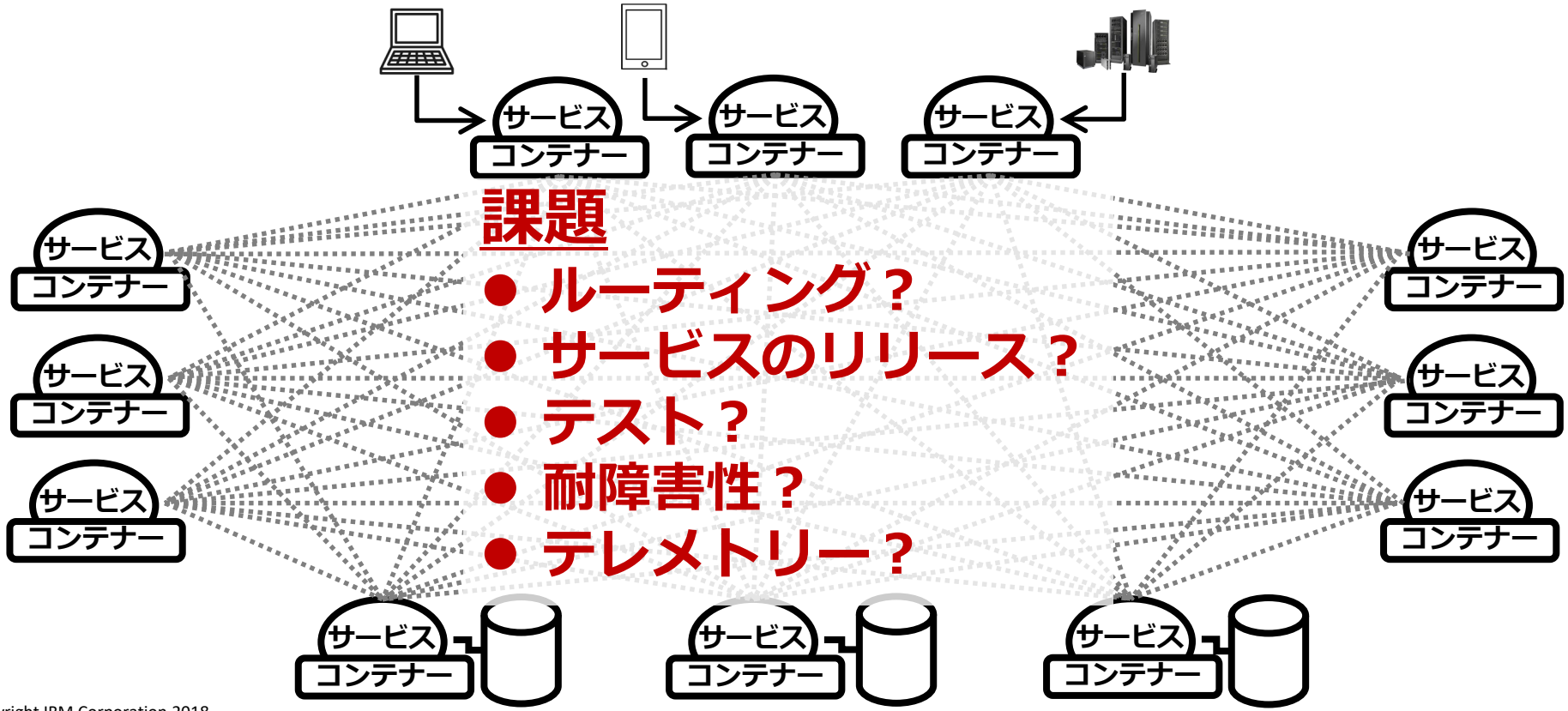
パフォーマンス最適化

- 高頻度の検索要求にはクラウド上のキャッシュで素早く処理
- クラウド-DC間のネットワークは、比較的低頻度の更新処理に限定

Istioによるサービス・メッシュ最適化

サービス・メッシュ

マイクロサービス・アーキテクチャーにおけるサービス間ネットワーク



Istio

マイクロサービスの相互接続, 管理,
セキュリティのためのオープンプラットフォーム
⇨**サービス・メッシュ最適化のソリューション**

- オープンソース・コミュニティ開発
- マルチ・プラットフォーム・サポート
- アプリケーション非依存

Istioの主要な機能

サービスの継続的なリリース
- ブルー/グリーン・デプロイメント
- カナリア・リリース

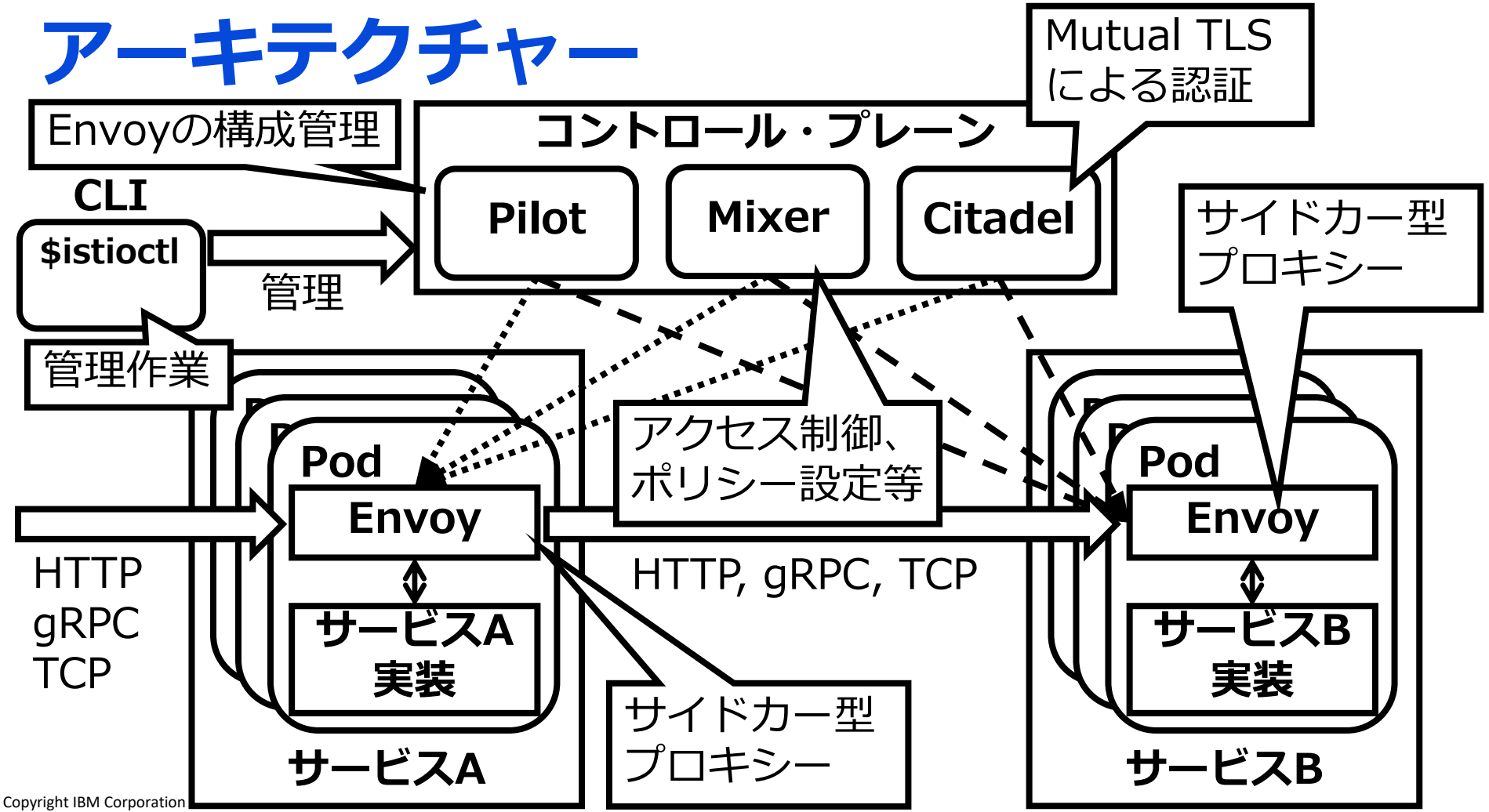
サービス間通信の認証・認可・暗号化
- 相互TLS

信頼性検証のためのEnd-to-Endテスト支援
- フォールト・インジェクション(カオス・エンジニアリング)

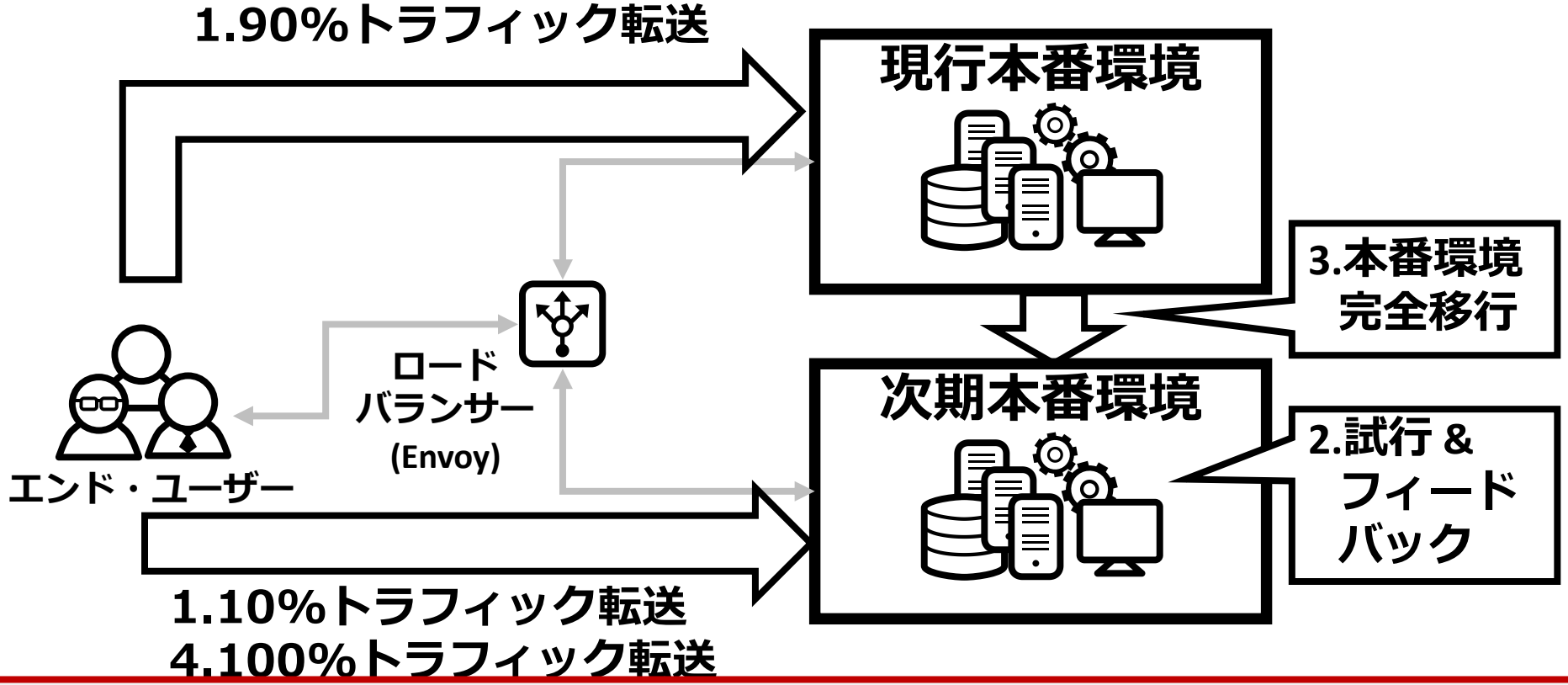
システム障害の影響最小化
- サーキット・ブレーカー

統合ダッシュボードログ, トレース, メトリクスの一覧監視
- 統合ダッシュボード

アーキテクチャー

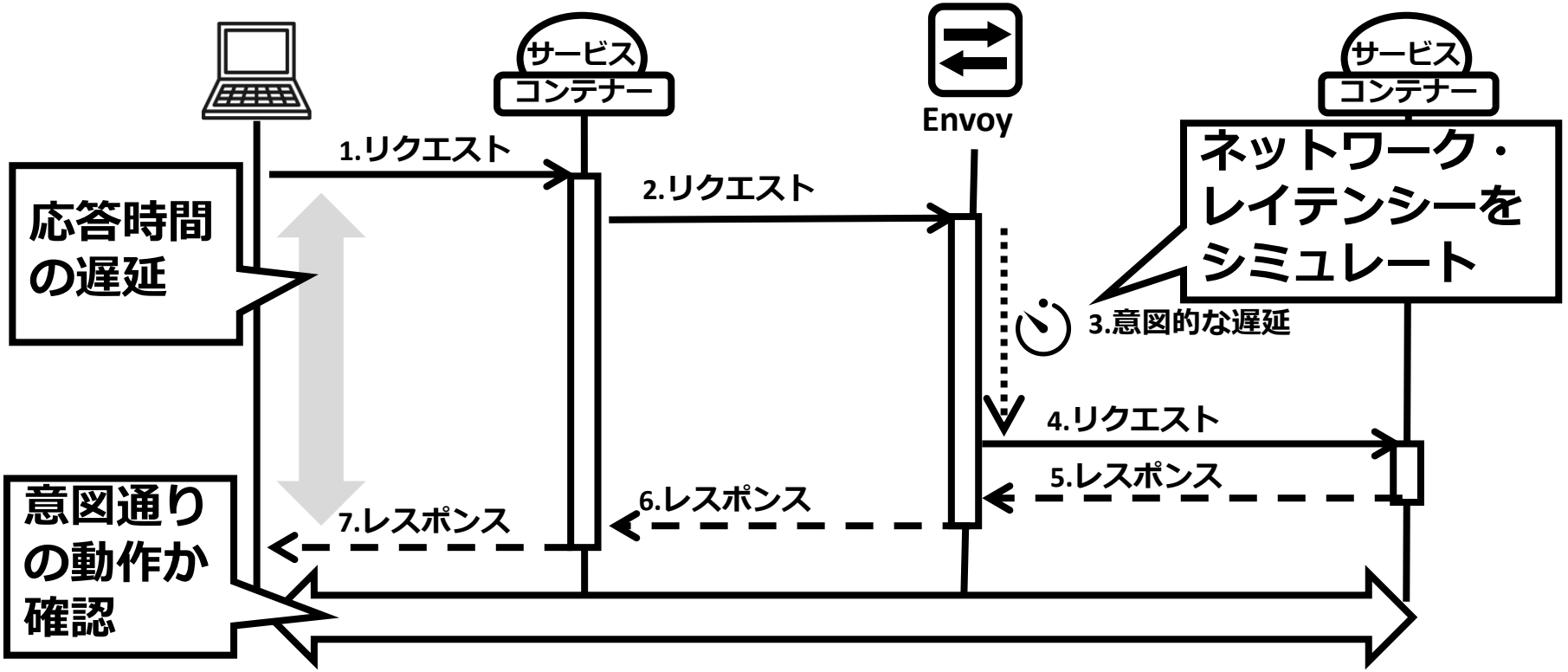


カナリア・リリース



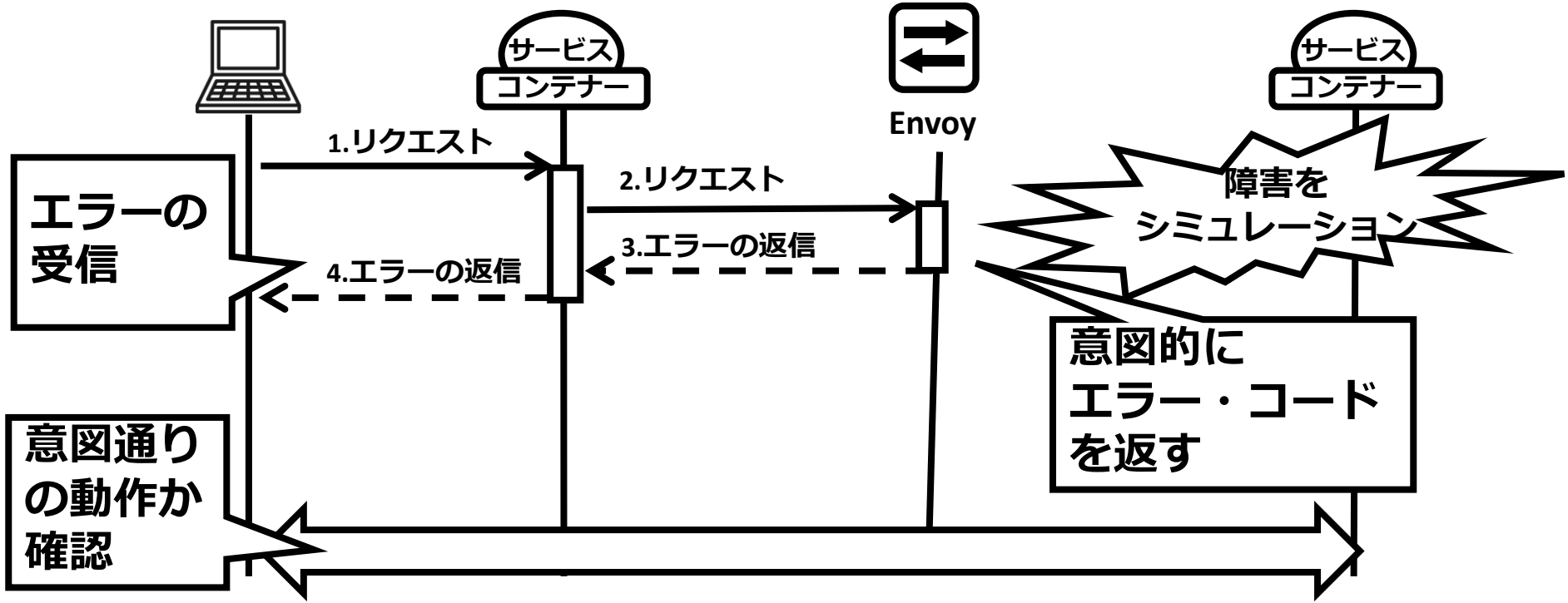
次期/代替リリースの試行とフィードバック

フォールト・インジェクション：遅延



ネットワーク遅延障害時のEnd-to-endテストを容易に実施可能

フォールト・インジェクション：エラー・コードの返信



ネットワーク・エラー発生時のEnd-to-endテストを容易に実施可能

サーキット・ブレーカー

サービス呼び出し失敗時の影響を最小化するデザイン・パターン

■**超分散システムでは, サービス呼出失敗が大規模障害に繋がる**

◆想定される影響

- ●応答時間遅延
- ●システム・リソースのブロック
- ●システム全体のダウン

■**解決策：サーキット・ブレーカー**

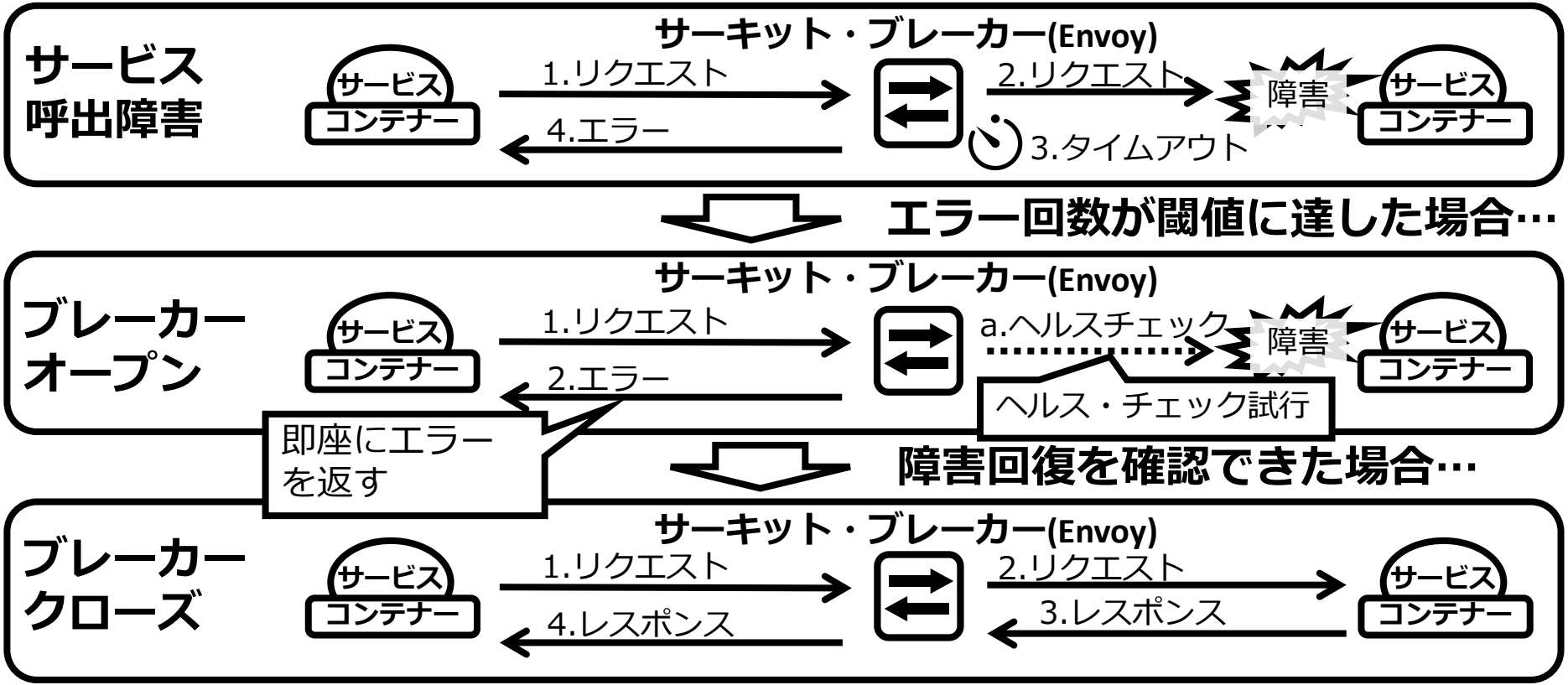
◆一定回数のサービス呼び出し失敗以降、サービス呼出元に即座にエラーを返す

◆期待される効果

- ●応答時間短縮
- ●システム・リソース・ブロックの排除

サーキット・ブレーカー

■動作フロー



まとめ

まとめ

- マイクロサービスは、複雑なシステム開発・運用に効果的
- マイクロサービス開発・運用にあたっては、サービス設計に加え, サービス・メッシュの最適化が求められる
- Istioは, サービス・メッシュ最適化のソリューション

ワークショップ、セッション、および資料は、IBMまたはセッション発表者によって準備され、それぞれ独自の見解を反映したものです。それらは情報提供の目的のみで提供されており、いかなる参加者に対しても法律的またはその他の指導や助言を意図したものではなく、またそのような結果を生むものでもありません。本講演資料に含まれている情報については、完全性と正確性を期するよう努力しましたが、「現状のまま」提供され、明示または暗示にかかわらずいかなる保証も伴わないものとします。本講演資料またはその他の資料の使用によって、あるいはその他の関連によって、いかなる損害が生じた場合も、IBMは責任を負わないものとします。 本講演資料に含まれている内容は、IBMまたはそのサプライヤーやライセンス交付者からいかなる保証または表明を引きだすことを意図したものでも、IBMソフトウェアの使用を規定する適用ライセンス契約の条項を変更することを意図したものでもなく、またそのような結果を生むものでもありません。

本講演資料でIBM製品、プログラム、またはサービスに言及していても、IBMが営業活動を行っているすべての国でそれらが使用可能であることを暗示するものではありません。本講演資料で言及している製品リリース日付や製品機能は、市場機会またはその他の要因に基づいてIBM独自の決定権をもっていつでも変更できるものとし、いかなる方法においても将来の製品または機能が使用可能になると確約することを意図したものではありません。本講演資料に含まれている内容は、参加者が開始する活動によって特定の販売、売上高の向上、またはその他の結果が生じると述べる、または暗示することを意図したものでも、またそのような結果を生むものでもありません。 パフォーマンスは、管理された環境において標準的なIBMベンチマークを使用した測定と予測に基づいています。ユーザーが経験する実際のスループットやパフォーマンスは、ユーザーのジョブ・ストリームにおけるマルチプログラミングの量、入出力構成、ストレージ構成、および処理されるワークロードなどの考慮事項を含む、数多くの要因に応じて変化します。したがって、個々のユーザーがここで述べられているものと同様の結果を得られると確約するものではありません。

記述されているすべてのお客様事例は、それらのお客様がどのようにIBM製品を使用したか、またそれらのお客様が達成した結果の実例として示されたものです。実際の環境コストおよびパフォーマンス特性は、お客様ごとに異なる場合があります。

IBM、IBM ロゴ、ibm.com、IBM Cloudは、 世界の多くの国で登録されたInternational Business Machines Corporationの商標です。他の製品名およびサービス名等は、それぞれIBMまたは各社の商標である場合があります。現時点での IBM の商標リストについては、www.ibm.com/legal/copytrade.shtmlをご覧ください。