[AWS BlackBelt Online Seminar]
SaaS アーキテクチャ 入門編
~マルチテナント SaaS とは~

櫻谷 広人
Amazon Web Services Japan G.K.
Partner Solutions Architect
2022/7/11

AWS 公式 Webinar
https://amzn.to/JPWebinar

過去資料
https://amzn.to/JPArchive

AWS Black Belt Online Seminar とは

- 「サービス別」「ソリューション別」「業種別」のそれぞれのテーマに分け、アマゾン ウェブ サービス ジャパン合同会社が主催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 の各サービスについてテーマごとに動画を公開します
- お好きな時間、お好きな場所でご受講いただけるオンデマンド形式です
- 動画を一時停止・スキップすることで、興味がある分野・項目だけの聴講も可能、スキマ時間の学習にもお役立ていただけます

内容についての注意点

- 本資料では2022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。
  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。

- AWS does not offer binding price quotes. AWS pricing is publicly available and is subject to change in accordance with the AWS Customer Agreement available at http://aws.amazon.com/agreement/. Any pricing information included in this document is provided only as an estimate of usage charges for AWS services based on certain information that you have provided. Monthly charges will be based on your actual use of AWS services, and may vary from the estimates provided.

自己紹介

櫻谷広人 (さくらや ひろと)

SaaS Partner Solutions Architect

アマゾン ウェブ サービス ジャパン 合同会社

❑ 経歴

主にバックエンドエンジニアとして Web サービスや
ネイティブアプリの開発を経験。
SIer ⇨ フリーランス ⇨ スタートアップ ⇨ AWS

❑ 好きな領域

サーバーレス、マイクロサービスアーキテクチャ

本ウェビナーの対象者

- SaaS ビジネスに従事されている開発者の方
- アーキテクチャ選定で迷っている方
- マルチテナント SaaS の開発・運用に課題をお持ちの方

  例：シングルテナントとマルチテナントどっちで作るのがいいんだろう... 🤔

各種 AWS サービスの基本的な説明や仕様については、
サービスカットの AWS Black Belt Online Seminar シリーズの動画をご覧ください

アジェンダ

[AWS BlackBelt Online Seminar]
SaaS アーキテクチャ 入門編
~マルチテナント SaaS とは~

[AWS BlackBelt Online Seminar]
SaaS アーキテクチャ 実践編
~マルチテナント SaaS の作り方~

アジェンダ

- SaaS の定義の確認
- SaaS アーキテクチャモデルの紹介
- アーキテクチャモデル選定のコツ
- まとめ

SaaS の定義

SaaS とは？

- **Software-as-a-Service**
- ソフトウェアの提供形態を指す用語
- 利用者側でインフラ構築・管理等が不要
- ネットワーク越しにサービスとしてソフトウェアの機能を利用できる
- 主にサブスクリプションと呼ばれる課金形態で提供されることが多い
- 事業者にとっても顧客にとってもメリットが多い提供形態

SaaS とは提供形態のことである（ビジネス的な視点）

従来のパッケージ販売

ソフトウェアパッケージを購入し、
手元の PC または会社のサーバーにインストール

SaaS モデル

ブラウザなどからネットワーク越しに利用
ソフトウェアの実体は SaaS 事業者側に存在

オフィスや自宅など場所を問わない
PC に限らずモバイル端末からも利用可能

SaaS とは提供形態のことである（ビジネス的な視点）

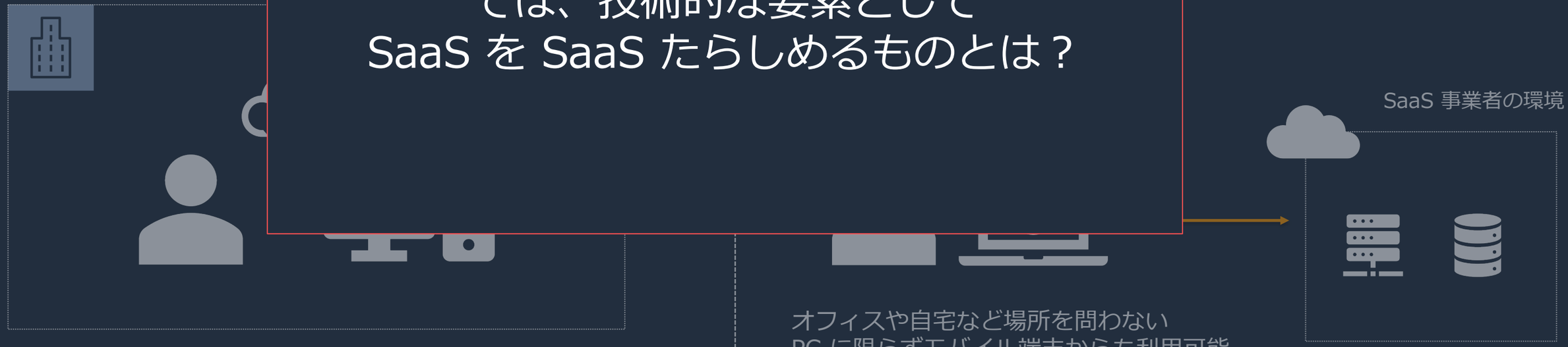
従来のパッケージ販売

SaaS モデル

ソフトウェアパッ
手元の PC または

ットワーク越しに利用
は SaaS 事業者側に存在

では、技術的な要素として
SaaS を SaaS たらしめるものとは？



オフィスや自宅など場所を問わない
PC に限らずモバイル端末からも利用可能

技術的な視点から見た SaaS の定義とは？

SaaS

共通の運用

共通の
アプリケーション

全ての顧客に対して共通のアプリケーションを運用し、
標準化された業務ナレッジをサービスとして提供すること

管理の効率化

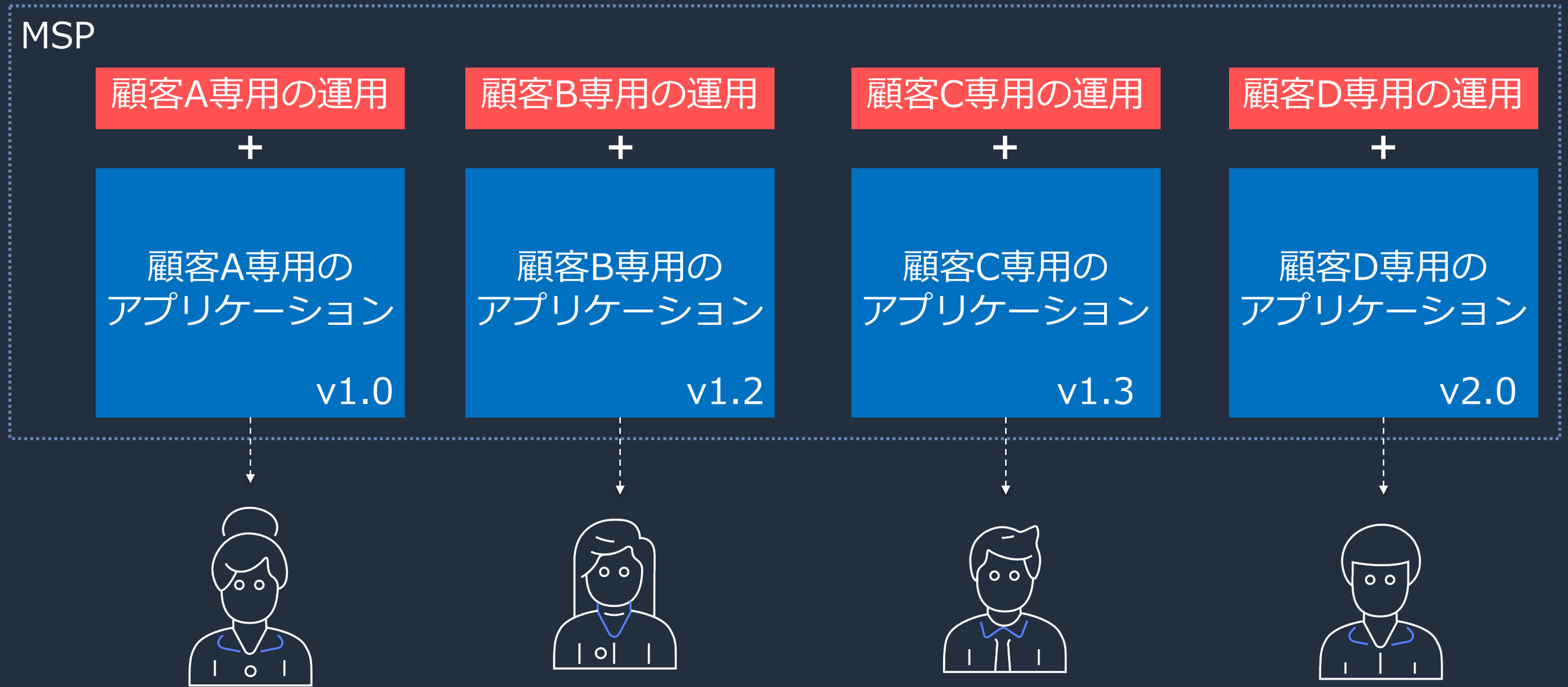
ビジネスのスケーラビリティ

スケーラブルな SaaS として成立しているかを図るための質問

- 明日の朝、急に **10,000** 社から契約の申込みがあった場合、
  全ての顧客に迅速にサービスを提供し始めることができるようになっていますか？
- 各社の要件に合わせたアプリケーションの個別カスタマイズは不要になっていますか？
- 全ての顧客の環境を運用できるエンジニアリソースはありますか？

正しく SaaS を作っていれば可能
MSP 的なモデルだと難しい

MSP (マネージドサービスプロバイダー) のモデル



個別の運用管理によってコストや複雑性は増すが、
顧客ごとの要件に柔軟に対応できる

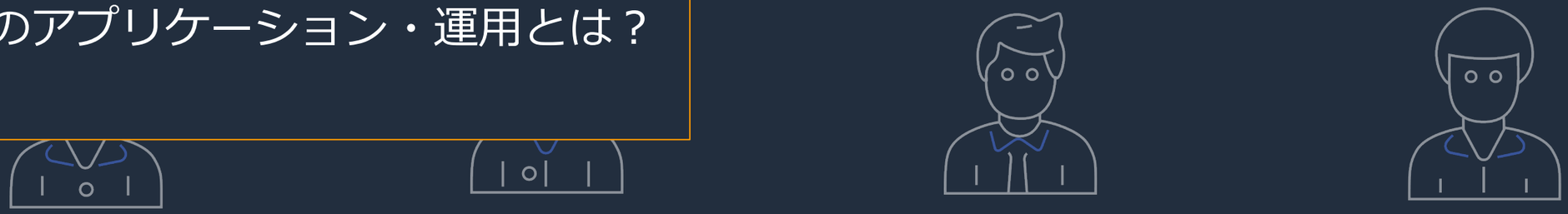
技術的な視点から見た SaaS の定義とは？

SaaS

共通の運用

共通の
アプリケーション

共通のアプリケーション・運用とは？

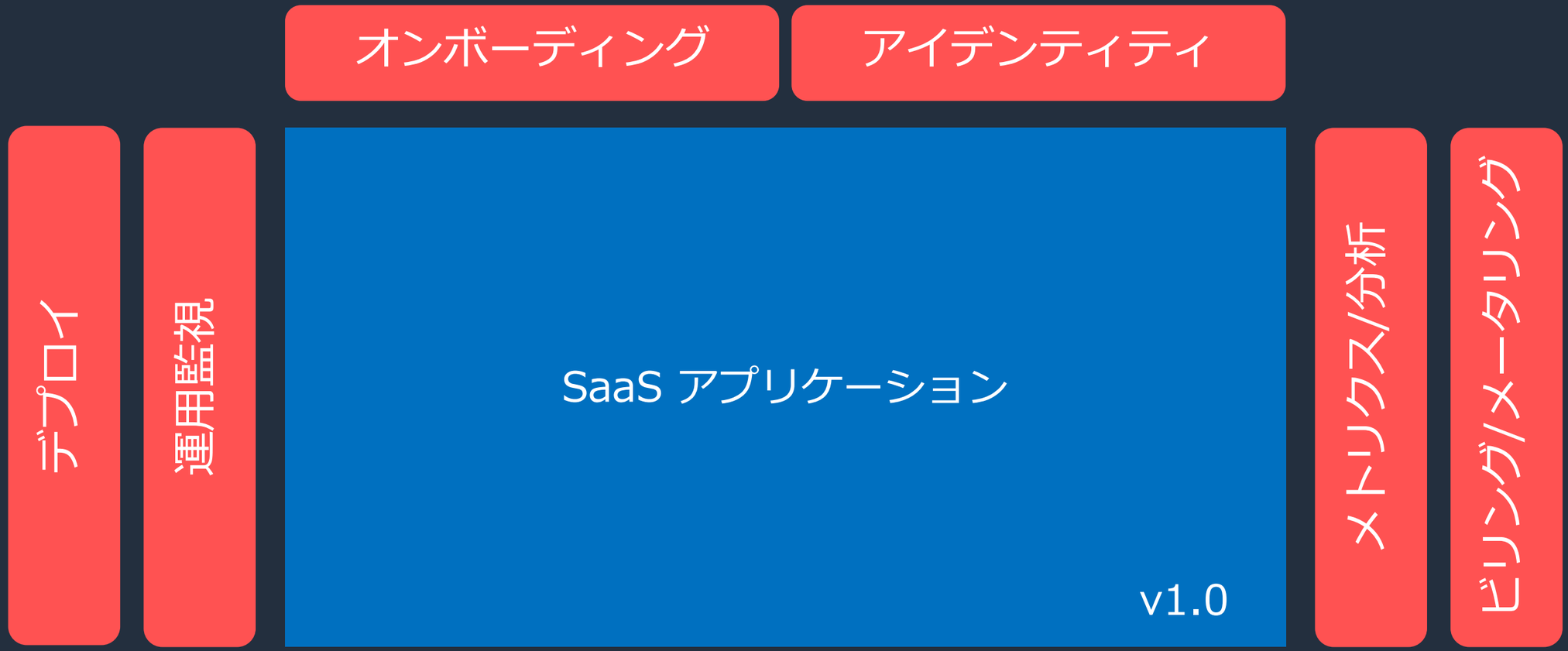


全ての顧客に対して共通のアプリケーションを運用し、
標準化された業務ナレッジをサービスとして提供すること

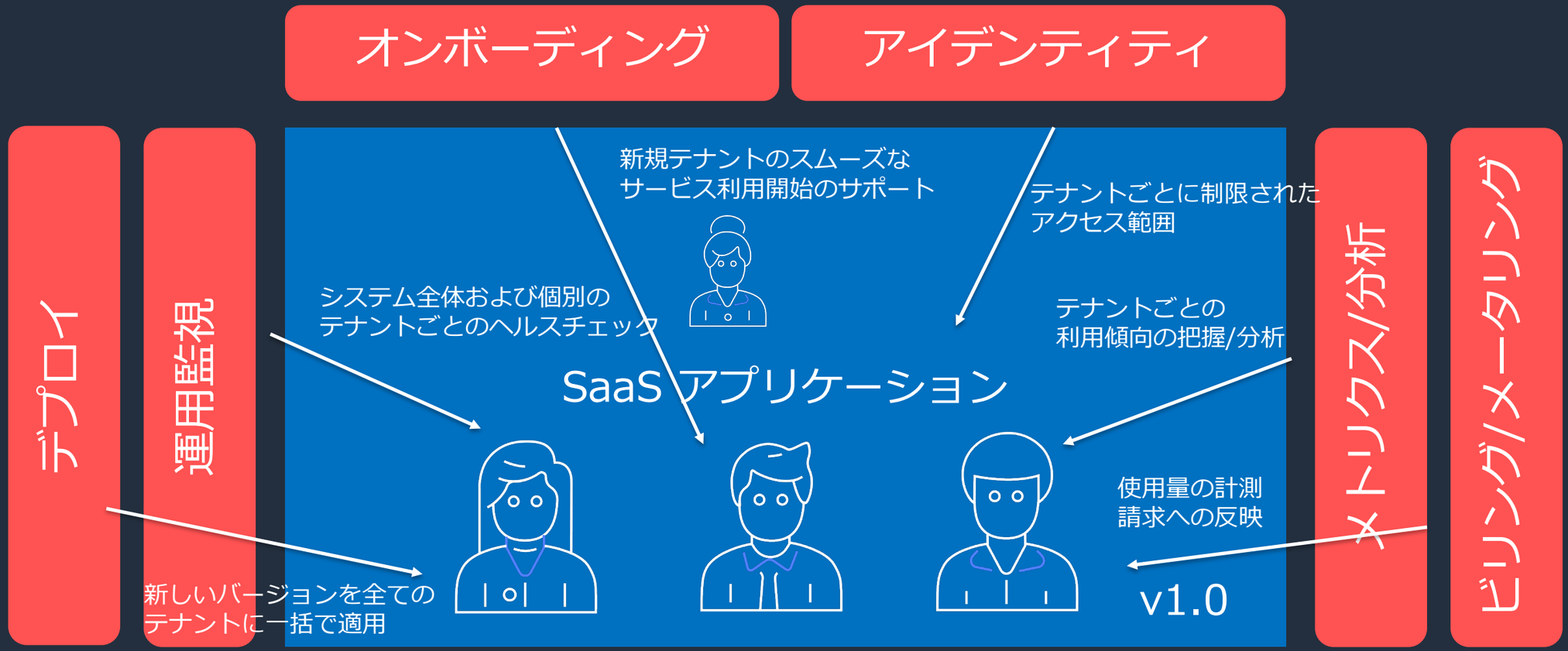
管理の効率化

ビジネスのスケーラビリティ

SaaS リファレンスアーキテクチャ



SaaS リファレンスアーキテクチャ



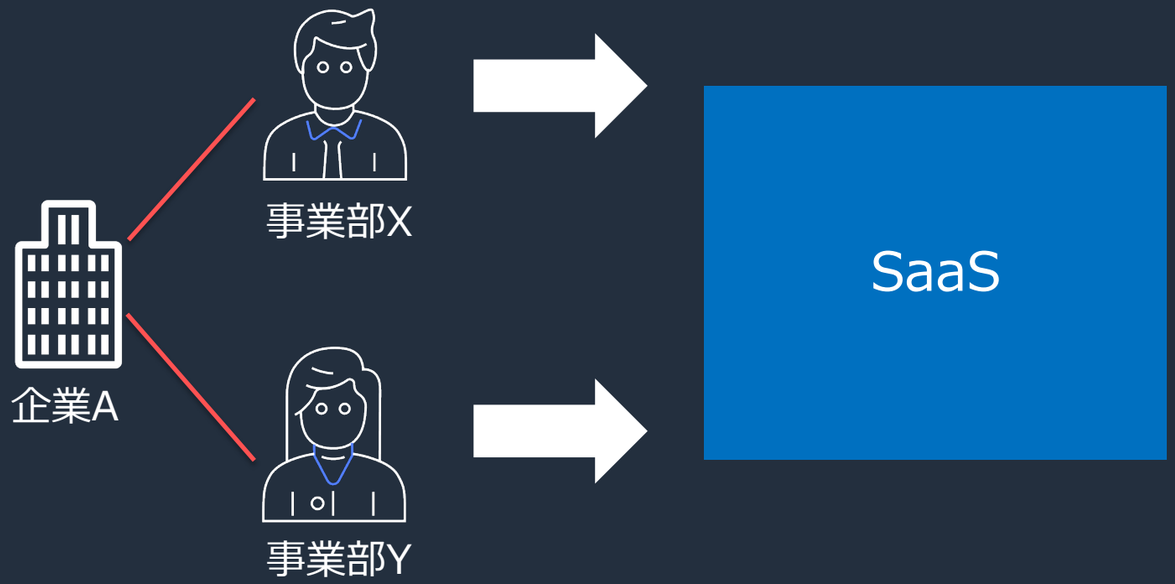
- 全てのテナントが同じ仕組みの下に管理される
- 共通＝マルチテナントではない
- シングルテナントでも共通化の思想は当てはまる

SaaS アーキテクチャモデル

テナントとは？

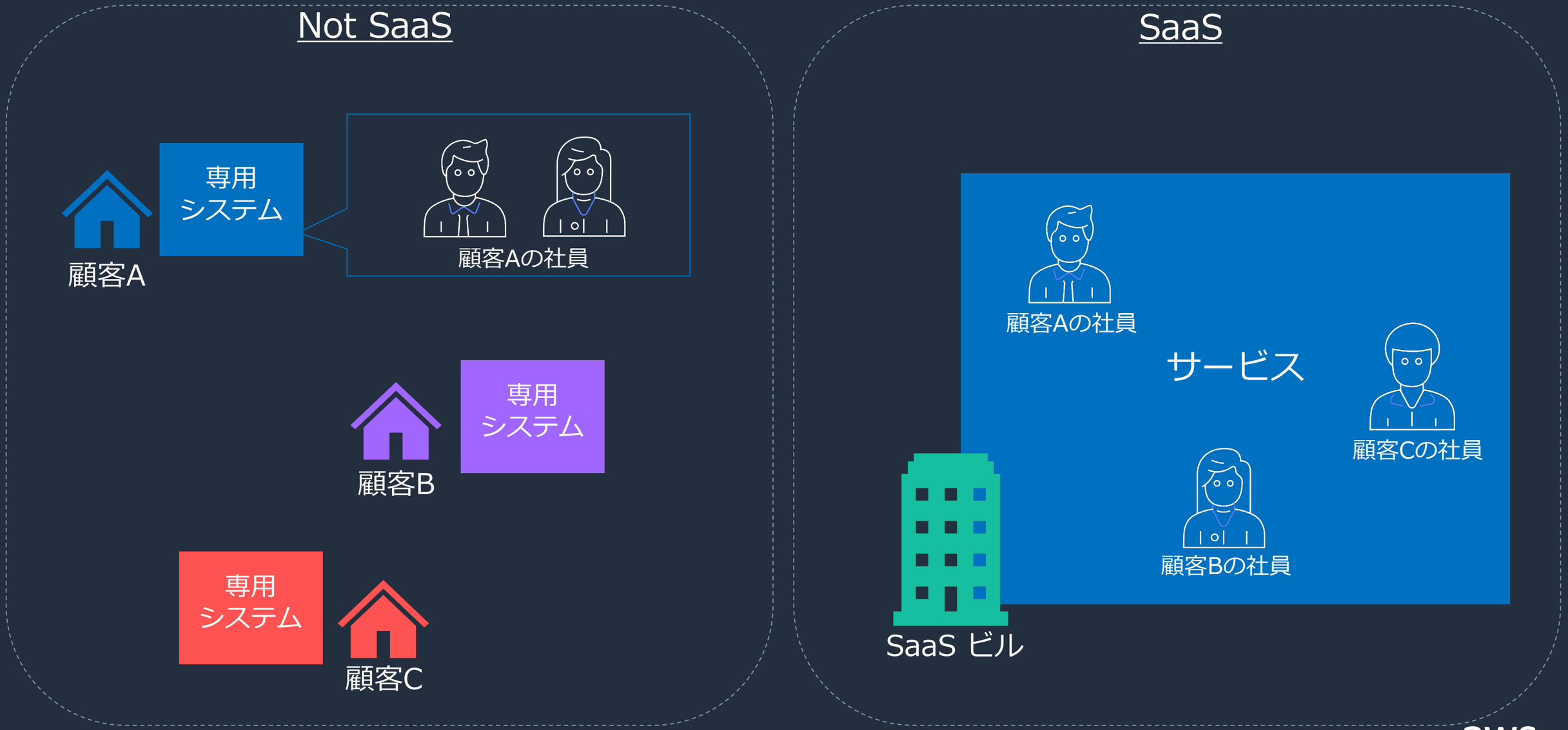
システムが取り扱う SaaS を利用する顧客の論理的な単位。実体はサービスによって様々。
1 つのテナントが企業を表す場合もあれば、事業部、ユーザーといった単位になる場合も

例：企業がテナント

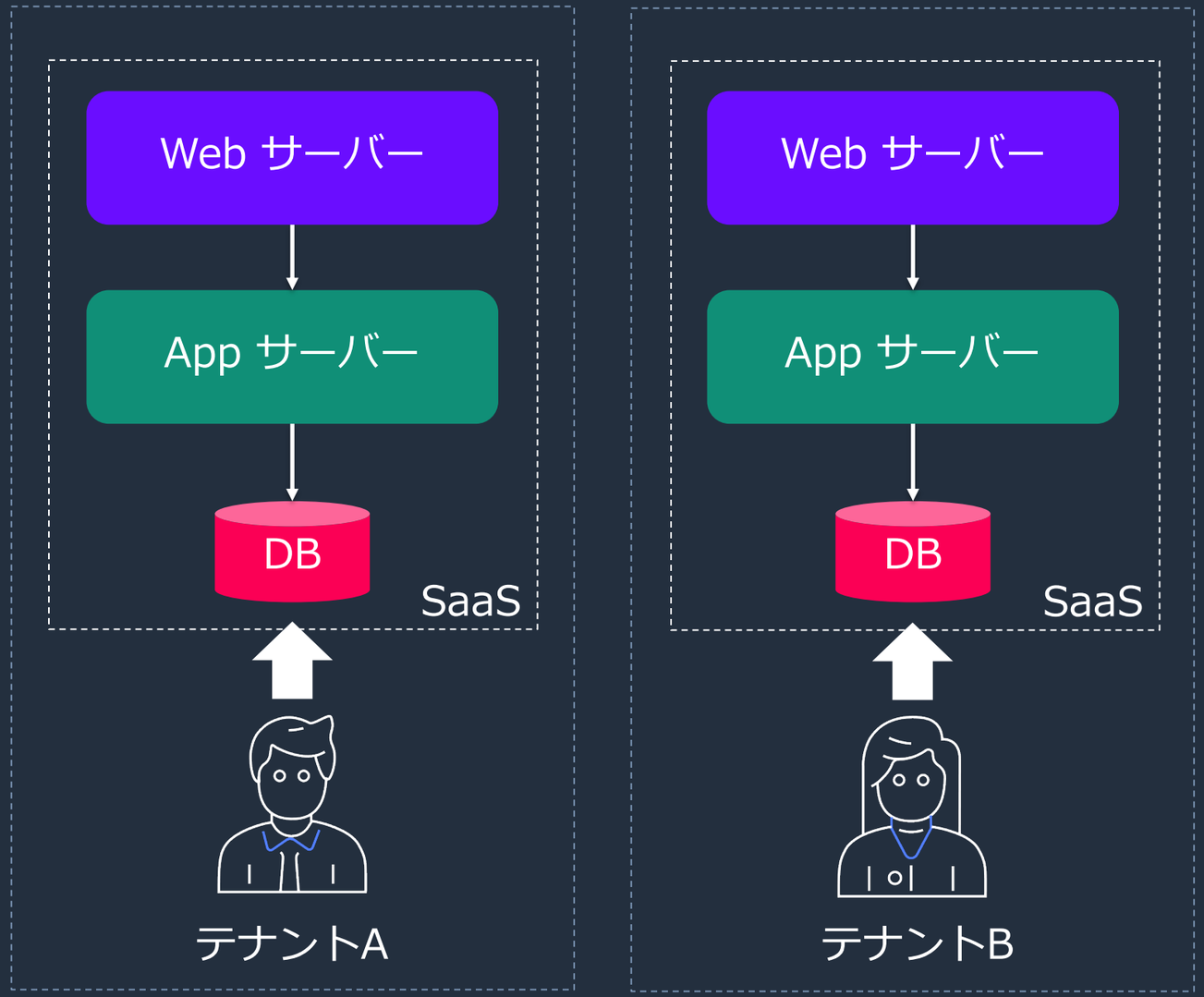


例：事業部がテナント

複数の利用者がサービスを共有しているイメージ



シングルテナントとは？



- テナントごとに専用の環境を提供
- サイロモデルとも呼ばれる

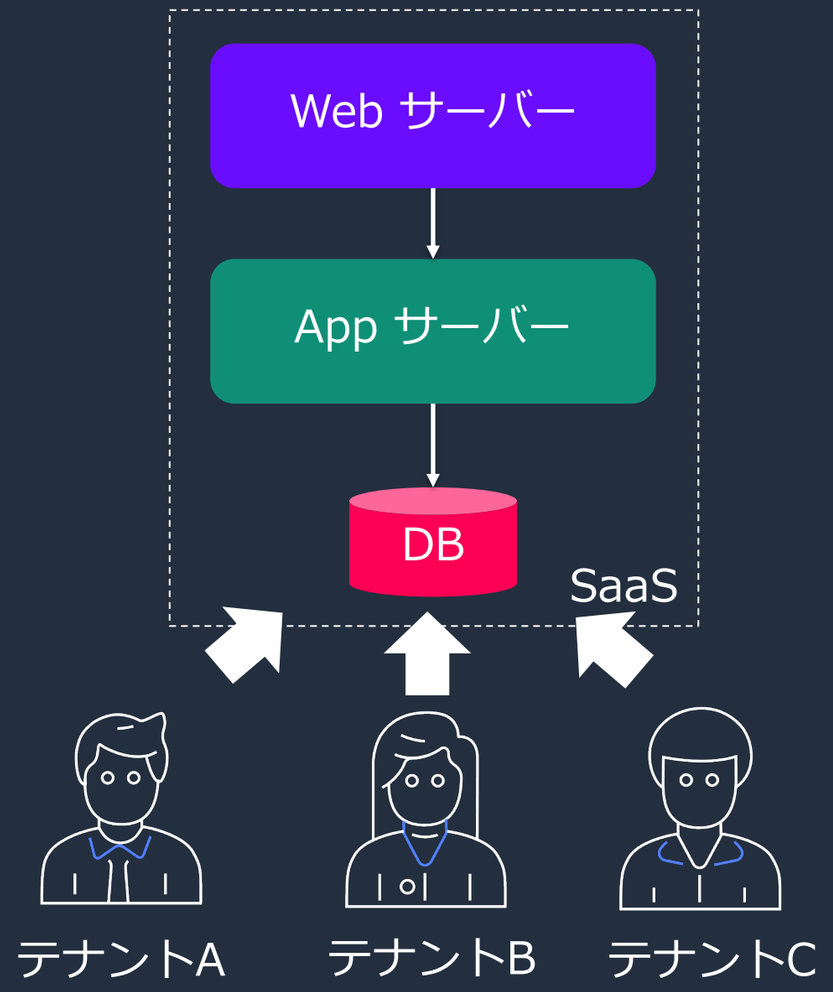
メリット

- コンプライアンス準拠
- 環境の分離
- テナントごとの可用性
- 他テナントによる影響の回避
- 個別のチューニング

デメリット

- コスト
- アジリティ
- 運用管理
- デプロイ
- 横断的なデータ分析

マルチテナントとは？



- 全てのテナントで環境を共有
- プールモデルとも呼ばれる

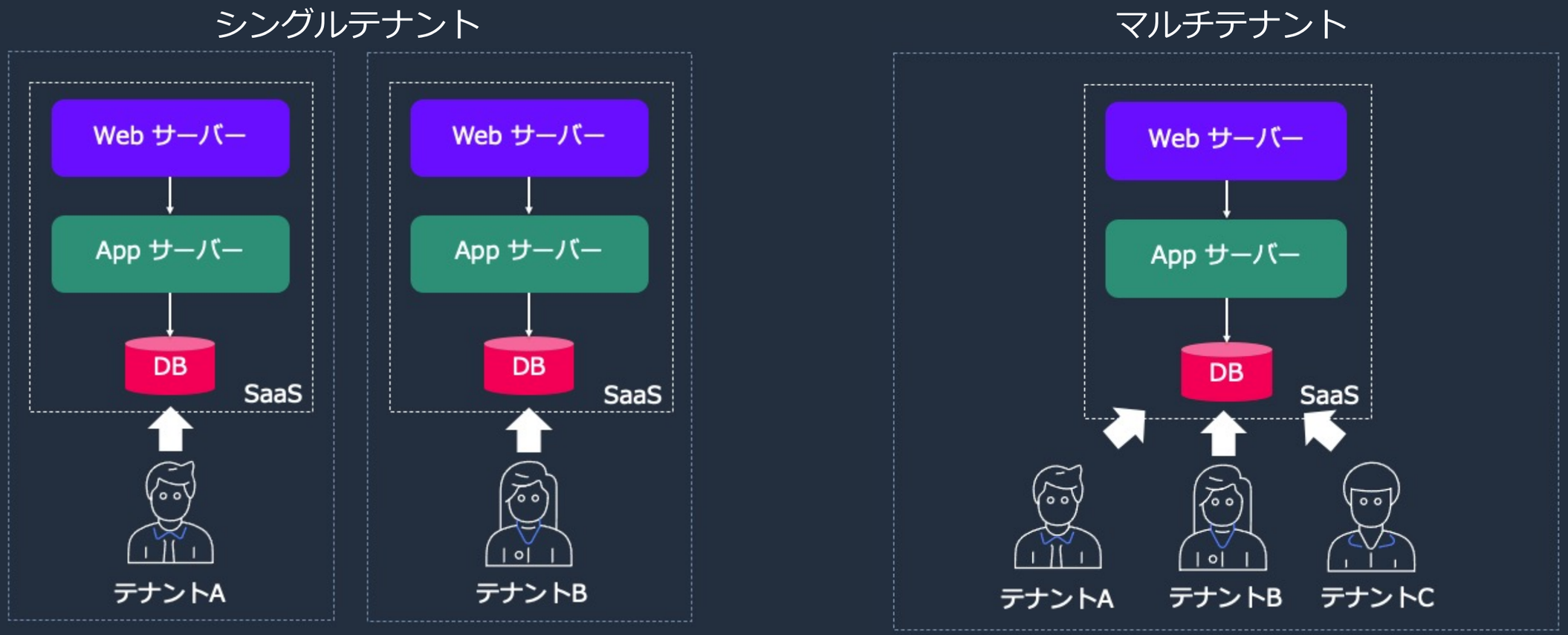
メリット

- コスト削減
- アジリティ
- 運用管理の集約
- シンプルなデプロイ
- 横断的なデータ分析

デメリット

- コンプライアンス
- ノイジーネイバー
- 単一の可用性
- 実装の複雑化

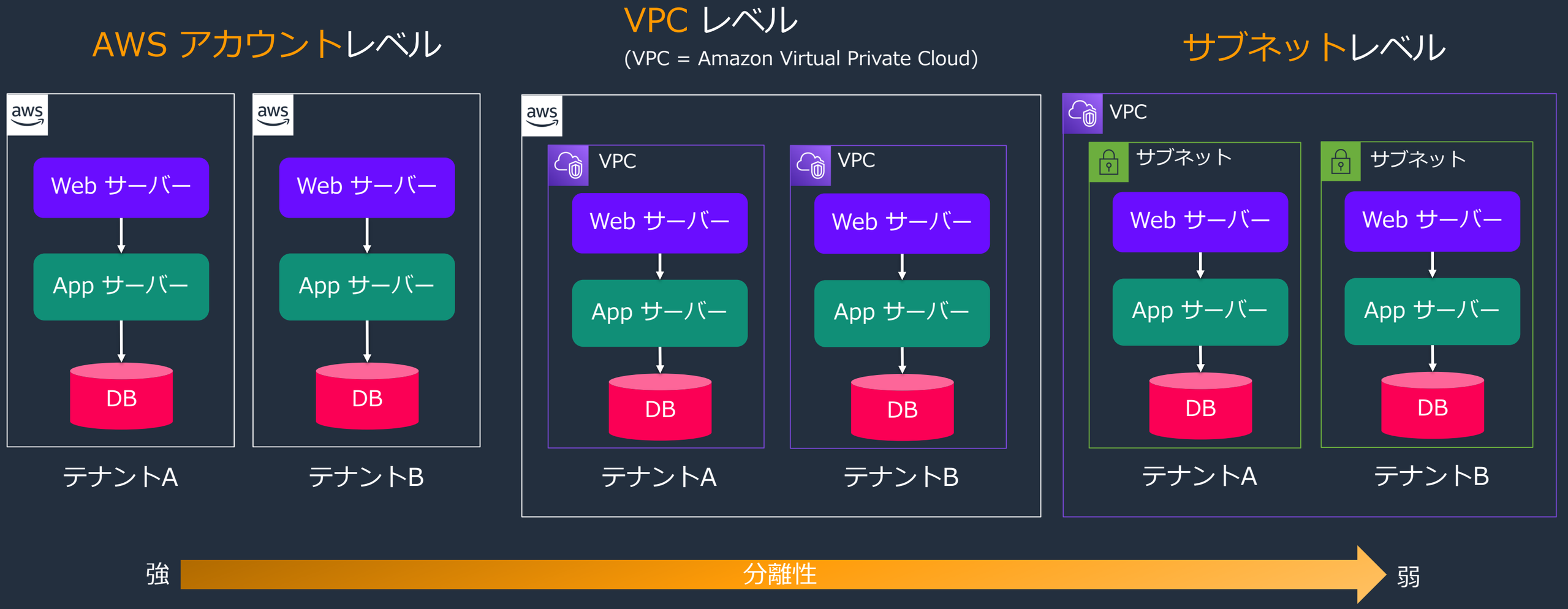
コンプライアンス観点でのメリット/デメリット



他のテナントとのリソース共有レベルはどこまで許容されますか？

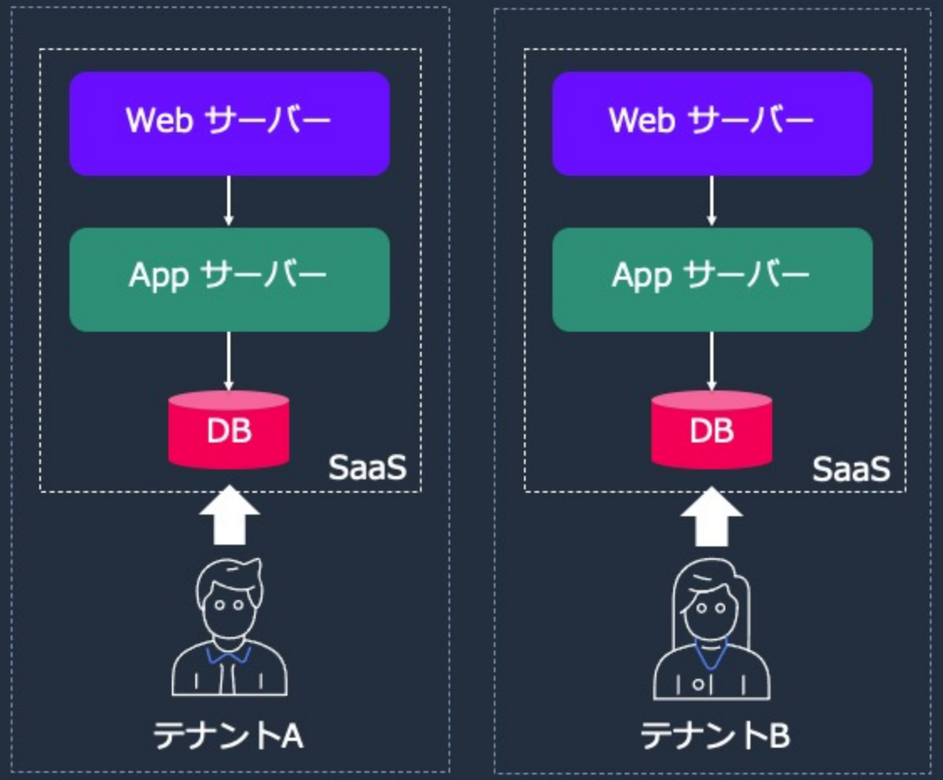
Ex. 物理/仮想レベル、データレジデンシー、ネットワーク分離、etc.

SaaS on AWS におけるサイロモデルの構築パターン



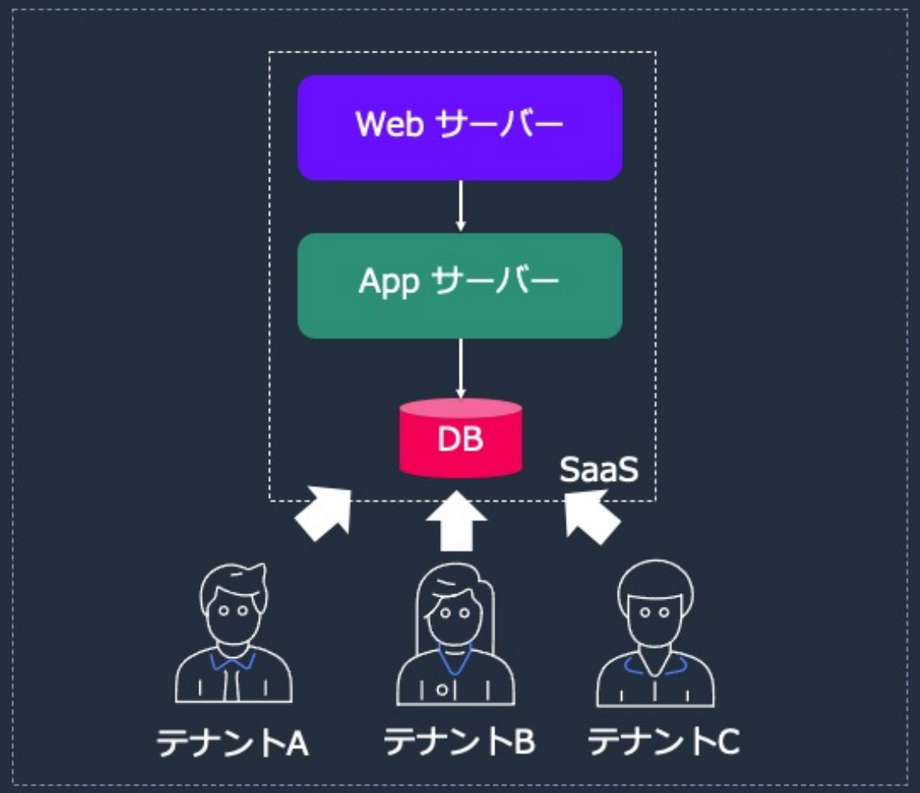
コスト観点でのメリット/デメリット

シングルテナント



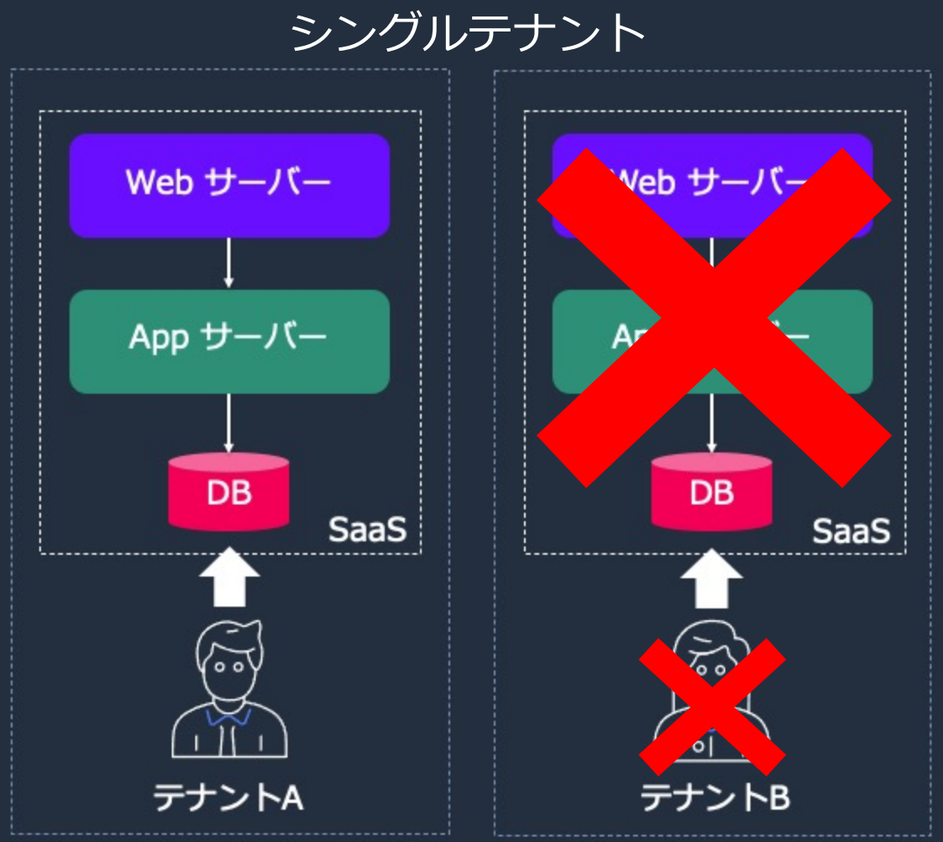
- テナント数に比例して線形に増えていく
- 運用コストも増加し、いつか限界が来る
- テナントによってリソースの使用率は様々（遊休リソースの発生）

マルチテナント



- テナントが増えても既存のリソースをスケールさせるだけ
- 中央管理による運用の効率化
- 余ったリソースは他のテナントによって使用されるため、無駄なくリソースが使える

可用性観点でのメリット/デメリット

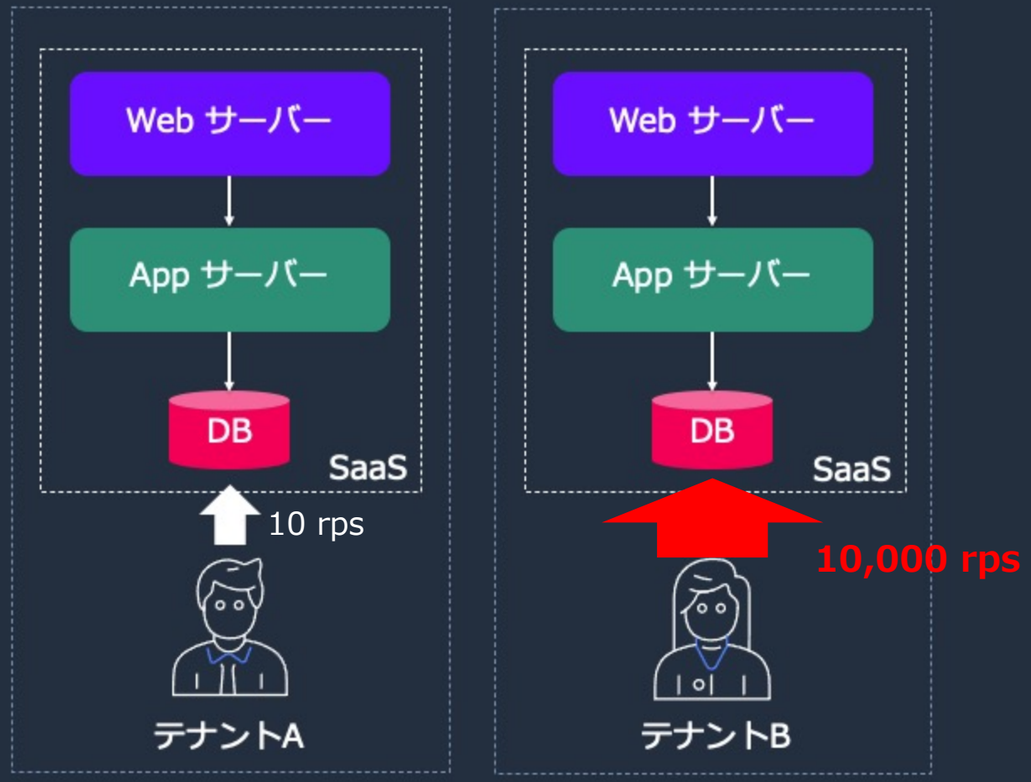


- 障害の影響範囲は限定的
- 他のテナントは問題なく利用し続けられる
- テナントごとに個別の DR 戦略を採用可能

- All-or-Nothing
- 全てのテナントで利用不可になるリスク

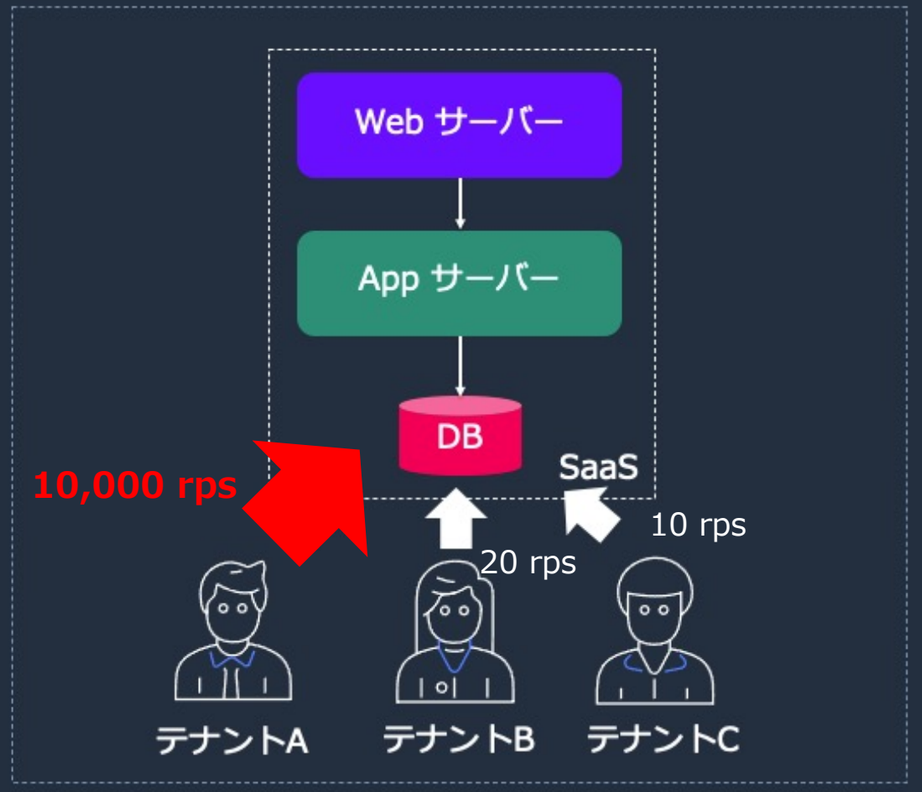
パフォーマンス観点でのメリット/デメリット

シングルテナント



- テナントごとに負荷の大きさやピークは異なる
- リソースは専用なので他テナントの影響は受けない
- テナントごとにキャパシティを調整可能

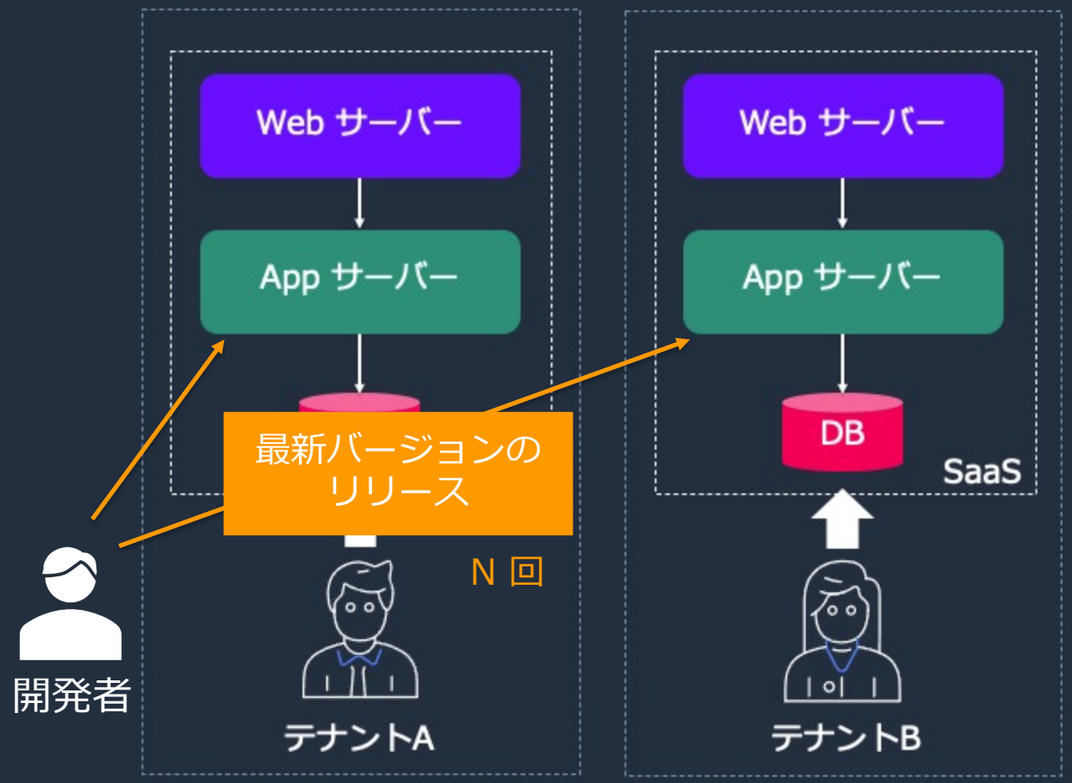
マルチテナント



- テナントごとに負荷の大きさやピークは異なる
- 一部のテナントが他テナントの分までリソースを使い尽くすリスクがある (**ノイジーネイバー問題**)
- リソース割り当て、スロットリングの仕組みが必要

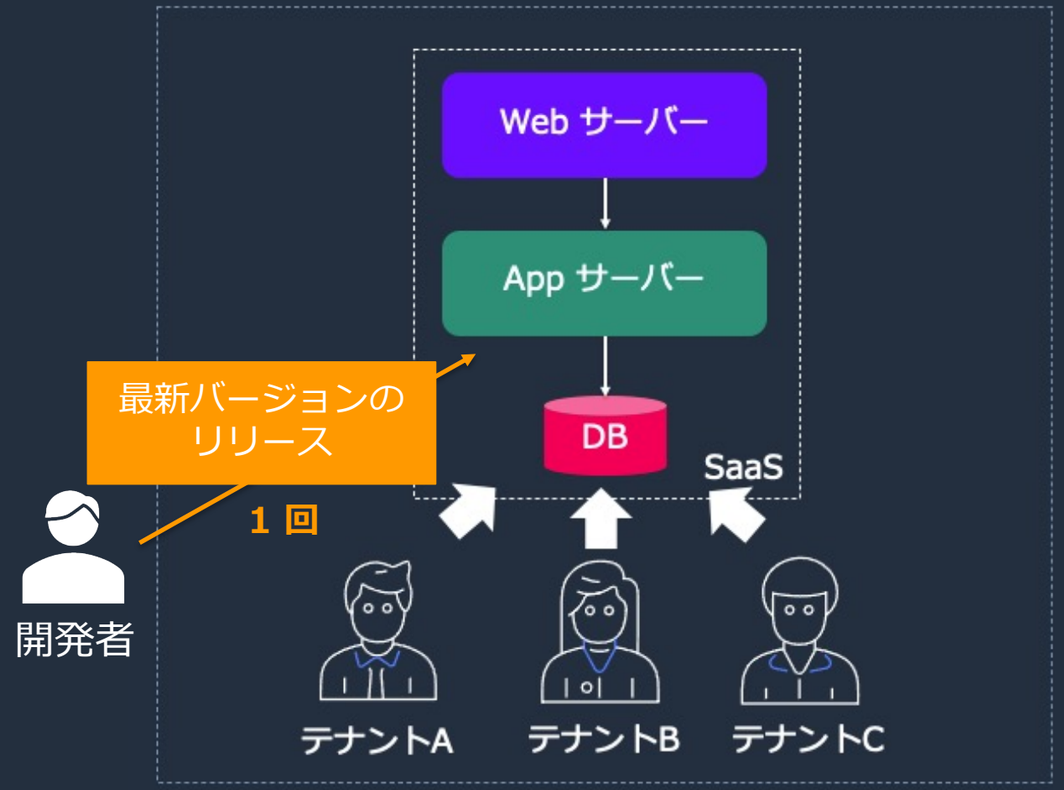
デプロイ観点でのメリット/デメリット

シングルテナント



- テナント数分デプロイが必要
- ロールアウト/ロールバックに時間がかかる
- 並列実行の作り込み

マルチテナント



- デプロイは 1 回で済む
- 迅速なリリースとロールバック
- DB のスキーマ変更なども含めると大きな利点に

アーキテクチャモデルは
どうやって選べば良い？

SaaS の初期開発における重要な課題

シングルテナントで作るべきか？

マルチテナントで作るべきか？

SaaS アーキテクチャモデルを考える上での 5 つのポイント

- 一つのモデルに固執しない。必要に応じて複数のモデル組み合わせる
- ビジネス要件を元に、トレードオフを理解した上で決める
- 事業のステージの変化に応じて、リアーキテクチャを行う
- 特にマルチテナントにおける課題をしっかりと認識する
- AWS のマネージドサービスを活用して、効率良く実装する

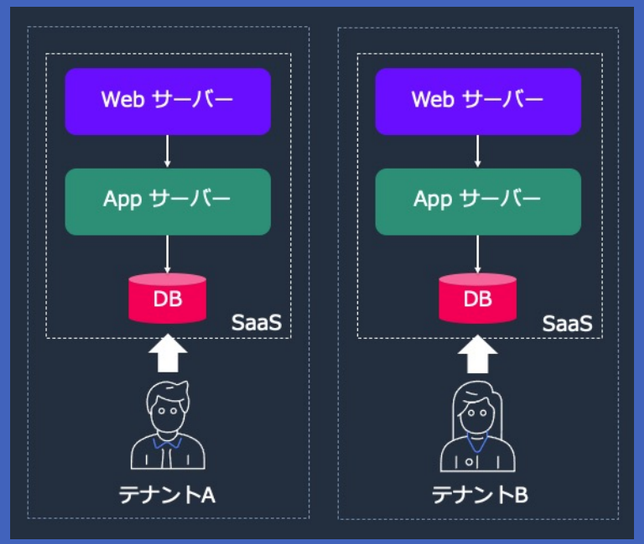
次の動画で解説

SaaS アーキテクチャモデルを考える上での 5 つのポイント

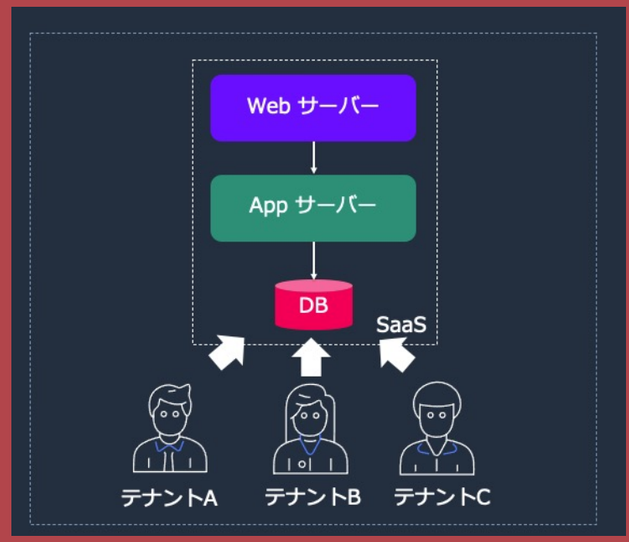
- 一つのモデルに固執しない。必要に応じて複数のモデル組み合わせる
- ビジネス要件を元に、トレードオフを理解した上で決める
- 事業のステージの変化に応じて、リアーキテクチャを行う
- 特にマルチテナントにおける課題をしっかりと認識する
- AWS のマネージドサービスを活用して、効率良く実装する

次の動画で解説

モデルは二者択一ではない

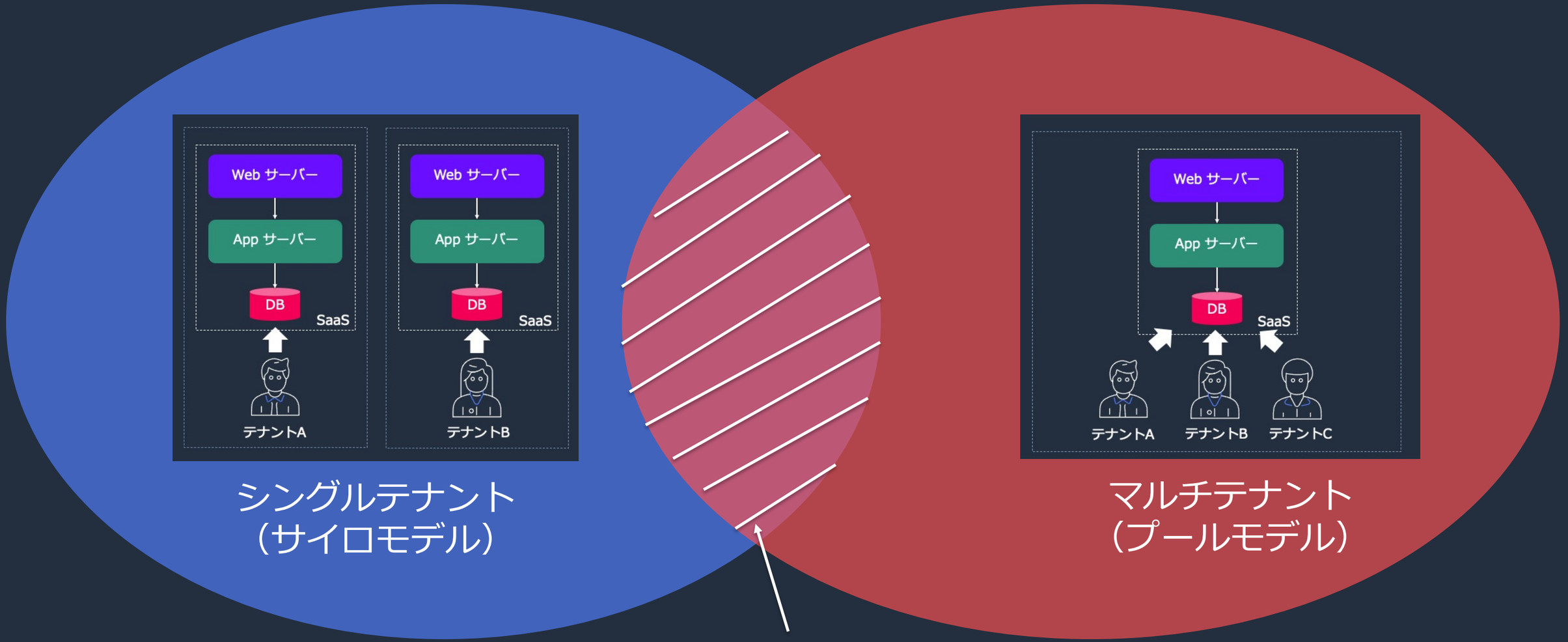


シングルテナント
（サイロモデル）

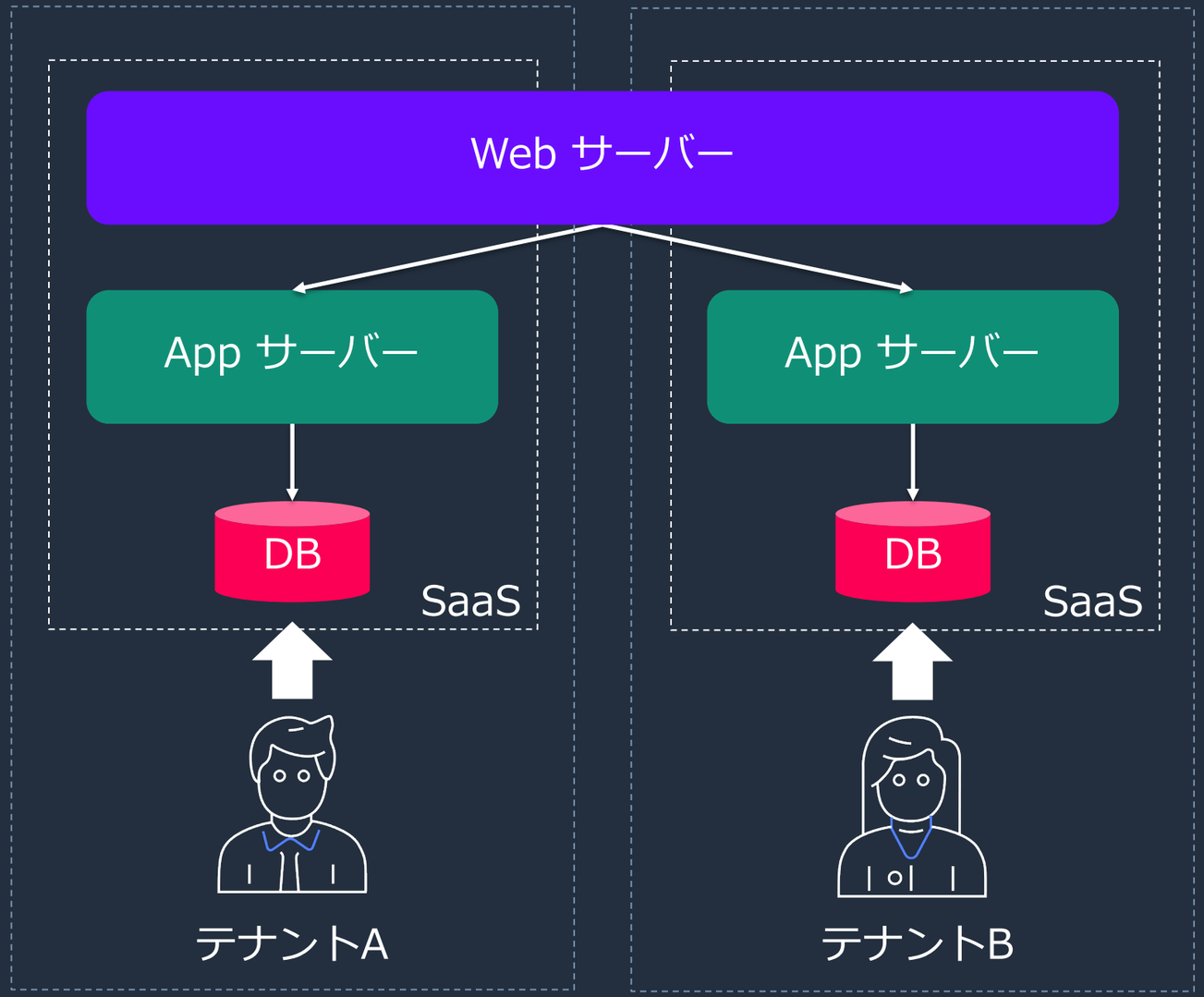


マルチテナント
（プールモデル）

モデルは二者択一ではない



ブリッジモデル（シングルテナント×マルチテナント）

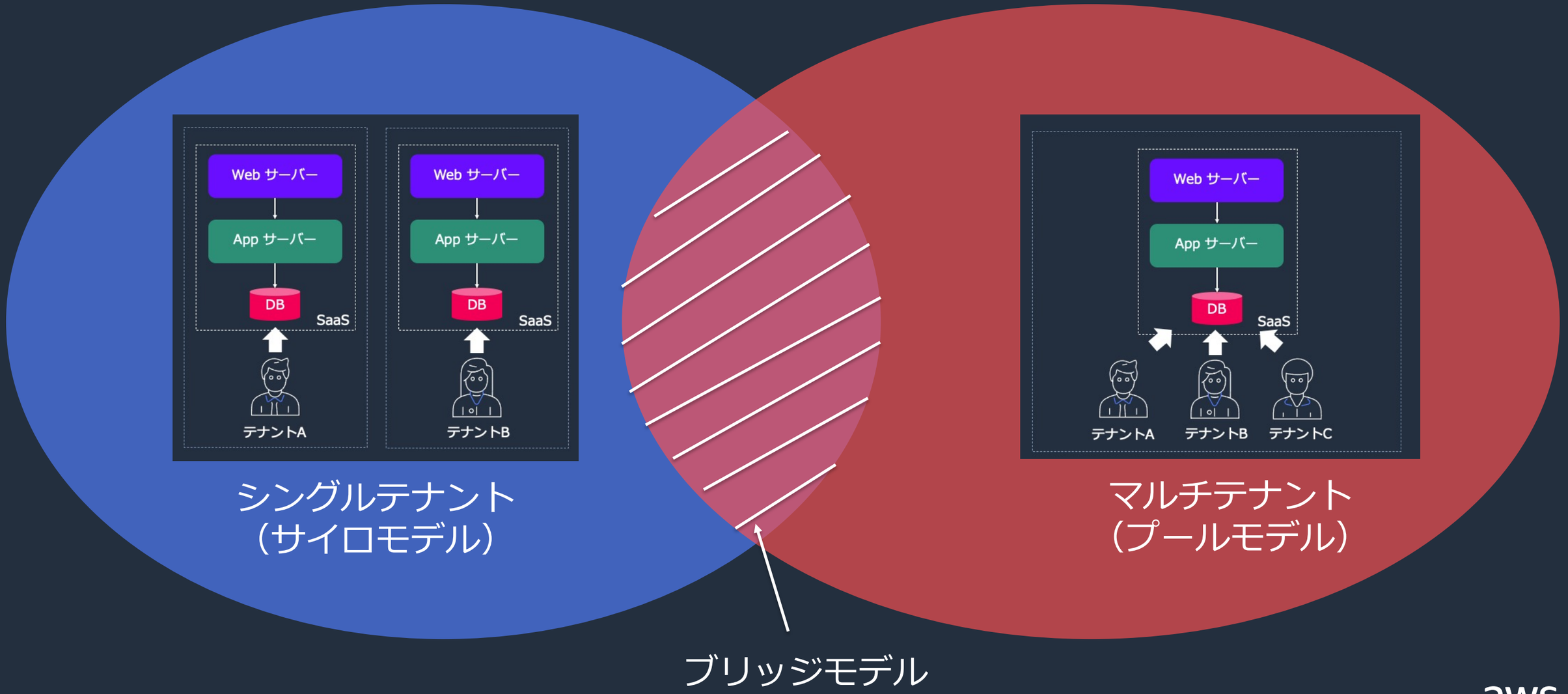


- システムを構成する各コンポーネントごとにサイロまたはプールモデルを選択して組み合わせる方式
- システム要件に合わせてサイロとプールのメリットをバランス良く享受することができるのでおすすめ

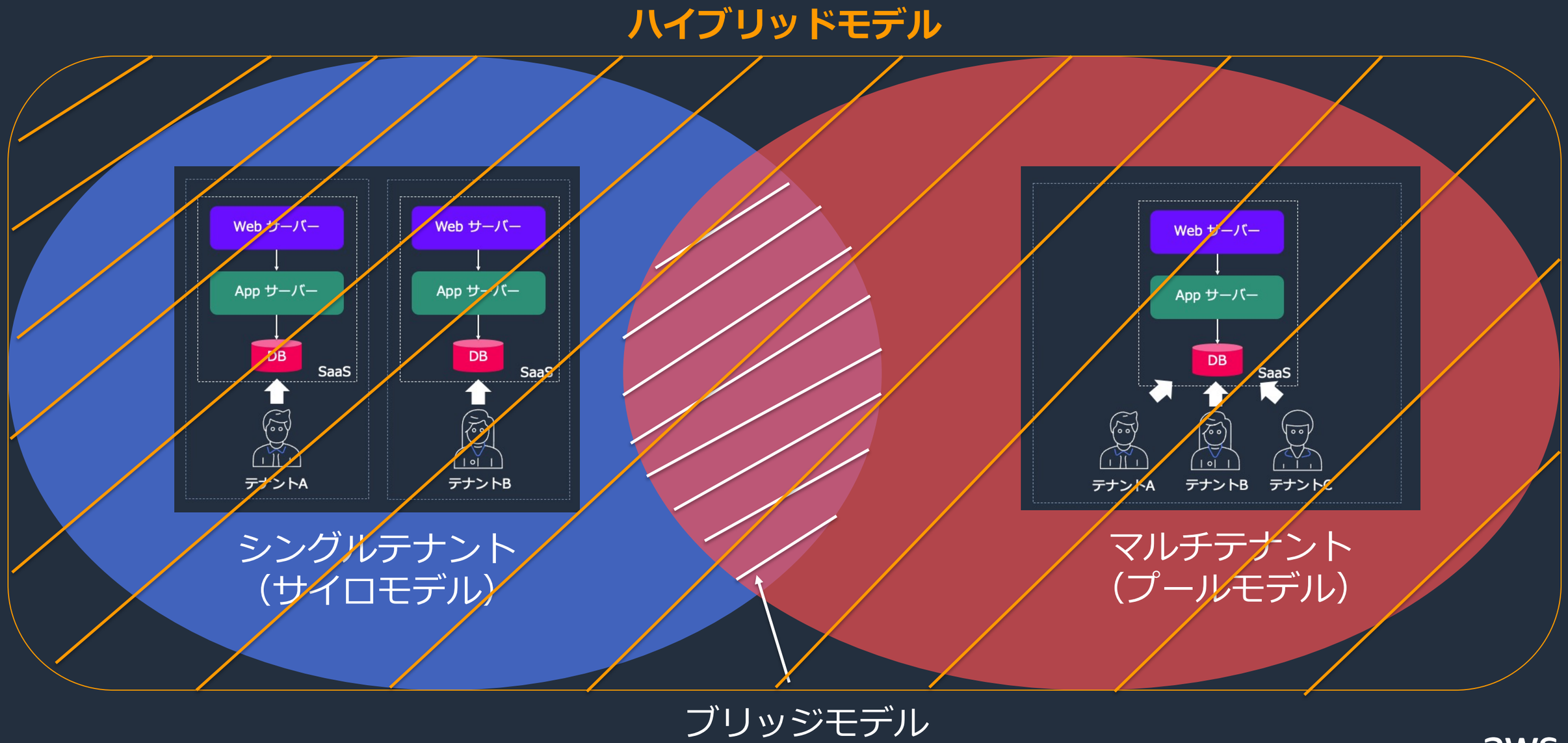
左の図は、テナントごとに分ける必要性がないWeb サーバーを共有し、その他の部分はサイロモデルで構成する例

別途ルーティングの仕組みなどを導入する必要があるが、フットプリントは小さくなりコストも削減される

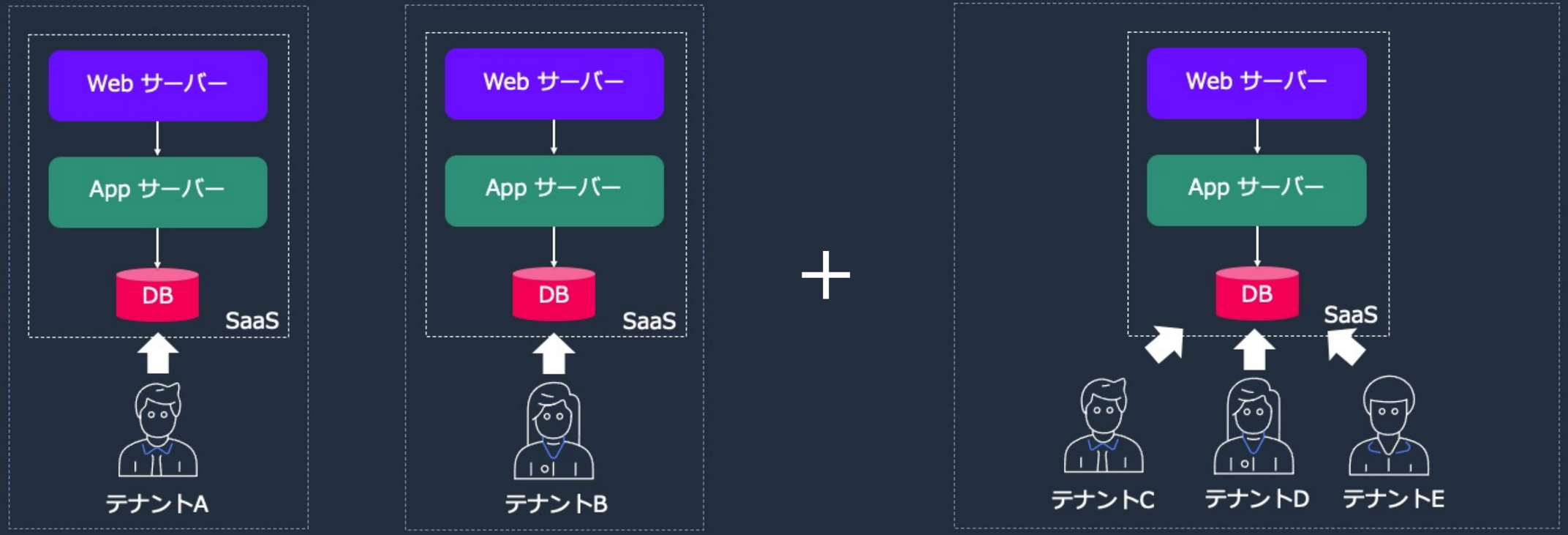
モデルは二者択一ではない



モデルは二者択一ではない



ハイブリッドモデル（シングルテナント+マルチテナント）

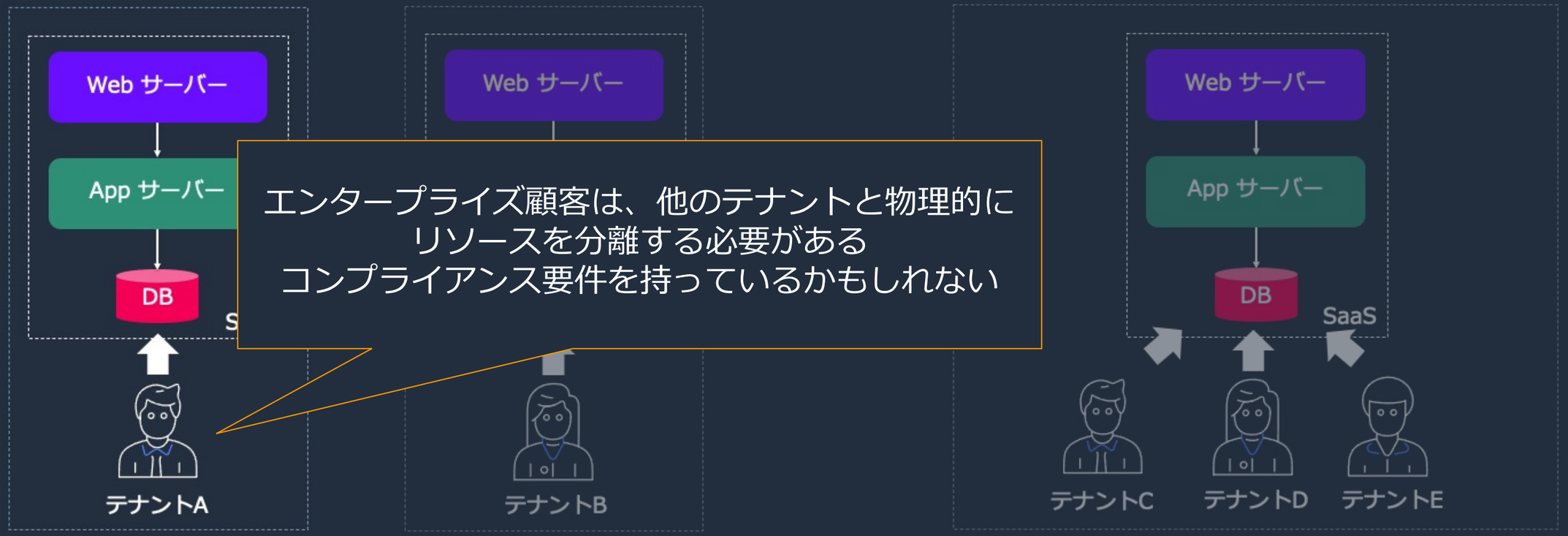


テナントA, B にはサイロモデルで専用の環境を提供

テナントC, D, E はプールモデルで環境を共有

テナントが求める様々な要件、契約プランに応じて提供する
体験を柔軟に調整することが可能

ハイブリッドモデル（シングルテナント+マルチテナント）

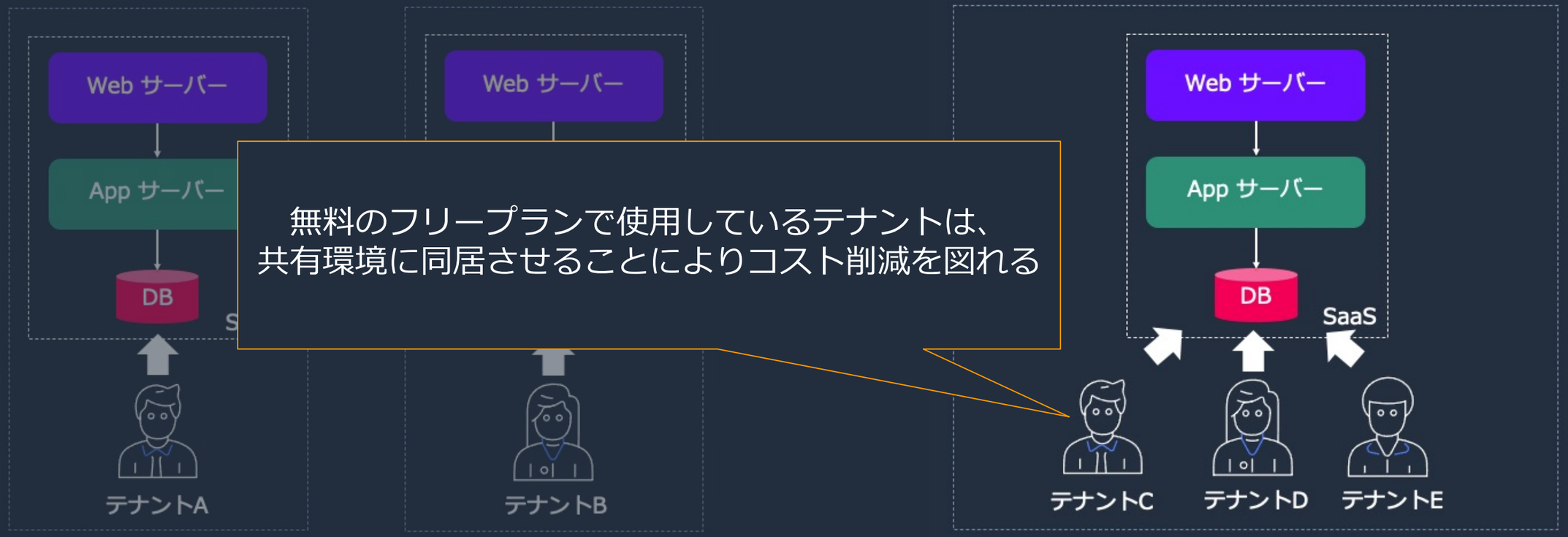


テナントA, B にはサイロモデルで専用の環境を提供

テナントC, D, E はプールモデルで環境を共有

テナントが求める様々な要件、契約プランに応じて提供する
体験を柔軟に調整することが可能

ハイブリッドモデル（シングルテナント+マルチテナント）



テナントA, B にはサイロモデルで専用の環境を提供

テナントC, D, E はプールモデルで環境を共有

テナントが求める様々な要件、契約プランに応じて提供する
体験を柔軟に調整することが可能

SaaS アーキテクチャモデルを考える上での 5 つのポイント

- 一つのモデルに固執しない。必要に応じて複数のモデル組み合わせる
- ビジネス要件を元に、トレードオフを理解した上で決める
- 事業のステージの変化に応じて、リアーキテクチャを行う
- 特にマルチテナントにおける課題をしっかりと認識する
- AWS のマネージドサービスを活用して、効率良く実装する

次の動画で解説

求める要件はテナントによって様々

テナント 1：数千名規模のエンタープライズ企業

- データは他のテナントとは物理的に分離してほしい
- プライベートなネットワークでサービスを利用したい
- 可用性は 99.999 %以上が必要
- コストは惜しまない

SaaS

- セキュリティは最低限でOK
- 可用性は 99 % 以上であればよい
- なるべくコストを抑えたい
- 一度に大量のデータをアップロードできるようにしてほしい

テナント 2：10~15 名のスタートアップ

求める要件はテナントによって様々

テナント 1：数千名規模のエンタープライズ企業

- データは他のテナントとは物理的に分離してほしい
- プライベートなネットワークでサービスを利用したい
- 可用性は 99.999 %以上が必要
- コストは惜しまない

Basic Plan

Enterprise Plan

- セキュリティは最低限でOK
- 可用性は 99 % 以上であればよい
- なるべくコストを抑えたい
- 一度に大量のデータをアップロードできるようにしてほしい

テナント 2：10~15 名のスタートアップ

要件に合わせて体験を差別化＋アーキテクチャの一致

Basic プラン 月額 980 円	Business プラン 月額 2,980 円	Enterprise プラン 月額 9,980 円
最大 5 ユーザーまで利用可能	最大 15 ユーザーまで利用可能	最大 50 ユーザーまで利用可能
SLA なし	サービス稼働率 99.9 %を保証	サービス稼働率 99.999 %を保証
最大 100 商品まで登録可能	最大 500 商品まで登録可能	商品登録数の制限なし
一部機能の制限あり	全ての機能を利用可能 ※商品のバッチ更新は一度に 50 点まで	全ての機能を利用可能 ※商品のバッチ更新は一度に 500 点まで
カスタマーサポート 3 営業日以内に返信	カスタマーサポート 24 時間以内に返信	カスタマーサポート 1 時間以内に返信

マルチテナント ＋ シングルテナント

ハイブリッドモデル

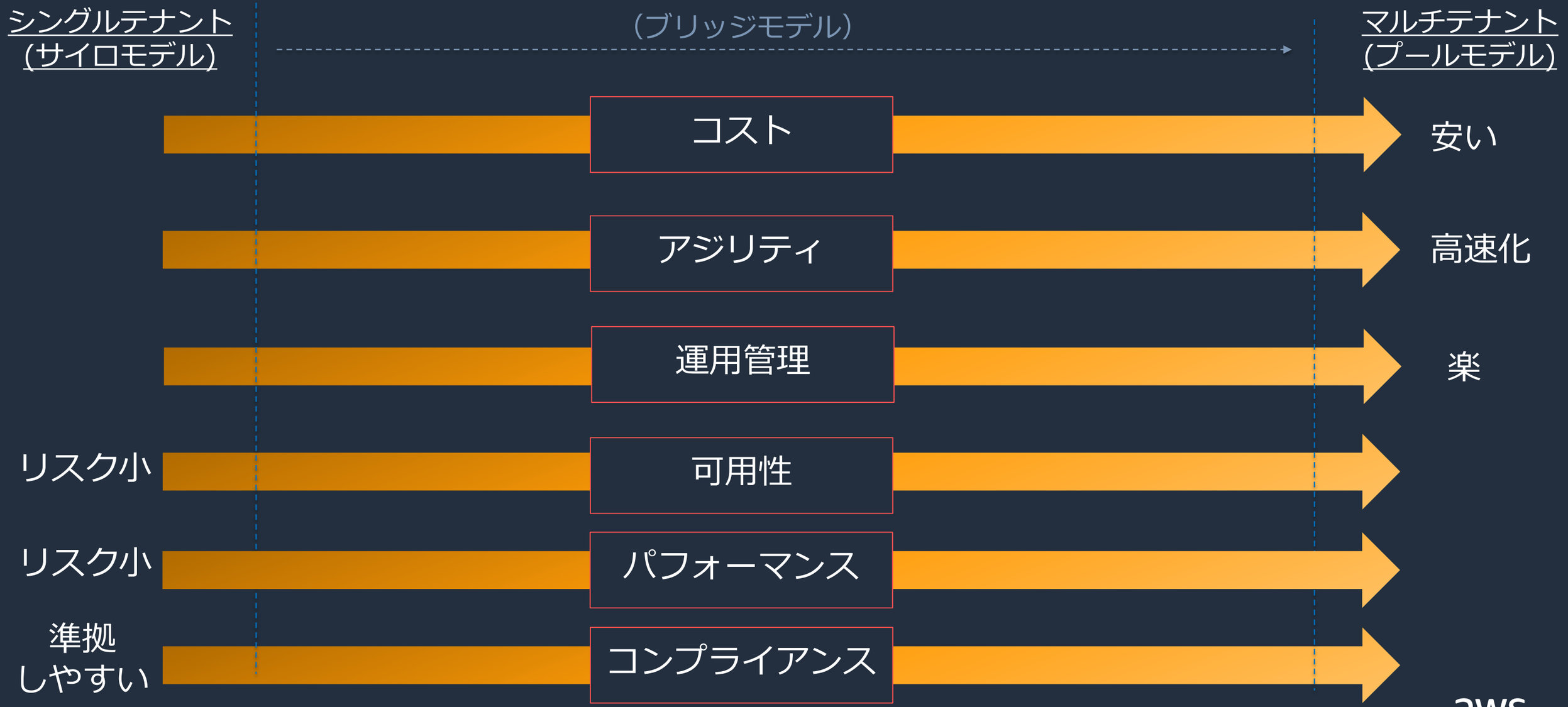
SaaS ビジネスモデルを構成する要素

How

What

誰に対してどんな体験を提供したいかが決まっていないと
アーキテクチャは決められない

トレードオフを理解する

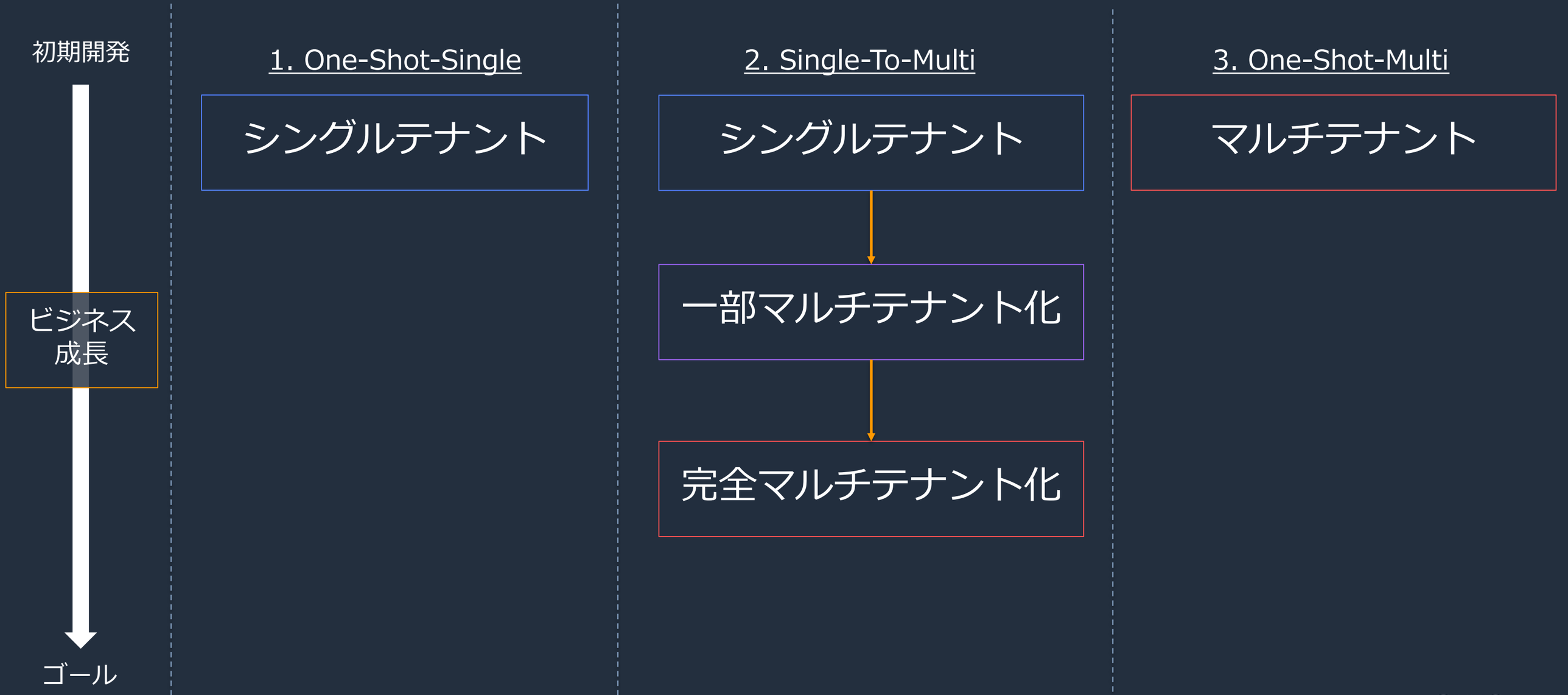


SaaS アーキテクチャモデルを考える上での 5 つのポイント

- 一つのモデルに固執しない。必要に応じて複数のモデル組み合わせる
- ビジネス要件を元に、トレードオフを理解した上で決める
- 事業のステージの変化に応じて、リアーキテクチャを行う
- 特にマルチテナントにおける課題をしっかりと認識する
- AWS のマネージドサービスを活用して、効率良く実装する

次の動画で解説

SaaS 化アプローチパターン



SaaS 化アプローチパターン

初期開発

ビジネス成長

ゴール

1. One-Shot-Single

シングルテナント

- テナント数が限られている
- 契約プランの差別化が少ない
- マルチテナント化できない

などの条件が揃えば
シングルテナントがゴールになり得る

一般的にはテナント数が増えるにつれ、
コストや運用管理のスケールに限界が来る

マルチテナント化へ

2. Single-To-Multi

シングルテナント

一部マルチテナント化

完全マルチテナント化

3. One-Shot-Multi

マルチテナント

SaaS 化アプローチパターン

初期開発

ビジネス成長

ゴール

1. One-Shot-Single

シングルテナント

- テナント数が限られている
- 契約プランの差別化が少ない
- マルチテナント化できない

などの条件が揃えば
シングルテナントがゴールになり得る

一般的にはテナント数が増えるにつれ、
コストや運用管理のスケールに限界が来る

マルチテナント化へ

2. Single-To-Multi

シングルテナント

一部マルチテナント化

完全マルチテナント化

シングルテナント→マルチテナントは
アプリケーションにも大きな変更が加わる
ので徐々に移行するのがおすすめ

あらかじめ事業計画に予算を組み込んでおく

3. One-Shot-Multi

マルチテナント

SaaS 化アプローチパターン

初期開発

ビジネス成長

ゴール

1. One-Shot-Single

シングルテナント

- テナント数が限られている
- 契約プランの差別化が少ない
- マルチテナント化できない

などの条件が揃えば
シングルテナントがゴールになり得る

一般的にはテナント数が増えるにつれ、
コストや運用管理のスケールに限界が来る

マルチテナント化へ

2. Single-To-Multi

シングルテナント

一部マルチテナント化

完全マルチテナント化

シングルテナント→マルチテナントは
アプリケーションにも大きな変更が加わる
ので徐々に移行するのがおすすめ

あらかじめ事業計画に予算を組み込んでおく

3. One-Shot-Multi

マルチテナント

最終的にマルチテナントにすること
が決まっていれば、初めから
マルチテナントで作るのも良い

ただしシングルテナントに比べて
実装が複雑なのでスタートダッシュ
は遅れる

既存の制約がない新規の SaaS 開発
ではおすすめのパターン

SaaS アーキテクチャモデルを考える上での 5 つのポイント

- 一つのモデルに固執しない。必要に応じて複数のモデル組み合わせる
- ビジネス要件を元に、トレードオフを理解した上で決める
- 事業のステージの変化に応じて、リアーキテクチャを行う
- 特にマルチテナントにおける課題をしっかりと認識し対策する
- AWS のマネージドサービスを活用して、効率良く実装する

次回の「実践編」にて詳しく見ていきます！

まとめ

- シングルテナントで作るかマルチテナントで作るかは二者択一ではない
- 顧客の要件に応じて最適なモデルを組み合わせて作ることができる
- それぞれのモデルにはメリット/デメリットがあり、銀の弾丸はない
- ビジネス要件から逆引きしてアーキテクチャを検討すること
- 事業の成長フェーズに応じてモデルの切り替えも計画すること

学習リソース

- AWS Well-Architected フレームワーク SaaS レンズ
- [AWS BlackBelt Online Seminar]
  AWS SaaS Boost で始める SaaS 開発入門
- 『SaaS のテナント分離戦略』ホワイトペーパー
- 『SaaS ストレージ戦略』ホワイトペーパー
- 『SaaS Journey Framework』ホワイトペーパー
- builders.flash SaaS 連載

ご視聴ありがとうございました