1人から1000万人までの道のり：AWSにおけるスケールするインフラ設計とは？

石見 和也
ソリューションアーキテクト
技術統括本部 インターネットメディアソリューション部

aws SUMMIT ONLINE

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Kazuya Iwami

Internet Media Solution Architect,

Amazon Web Services Japan

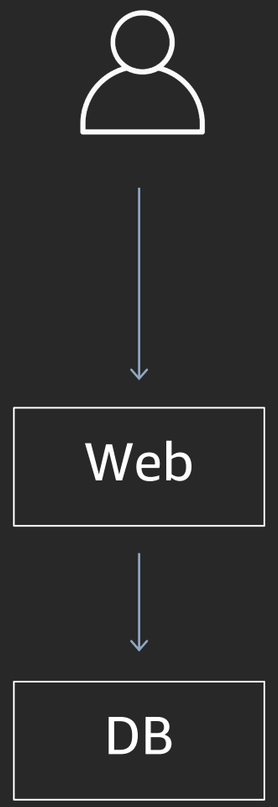
Machine Learningプロジェクトの支援も行う

最近お気に入りのAWSサービス

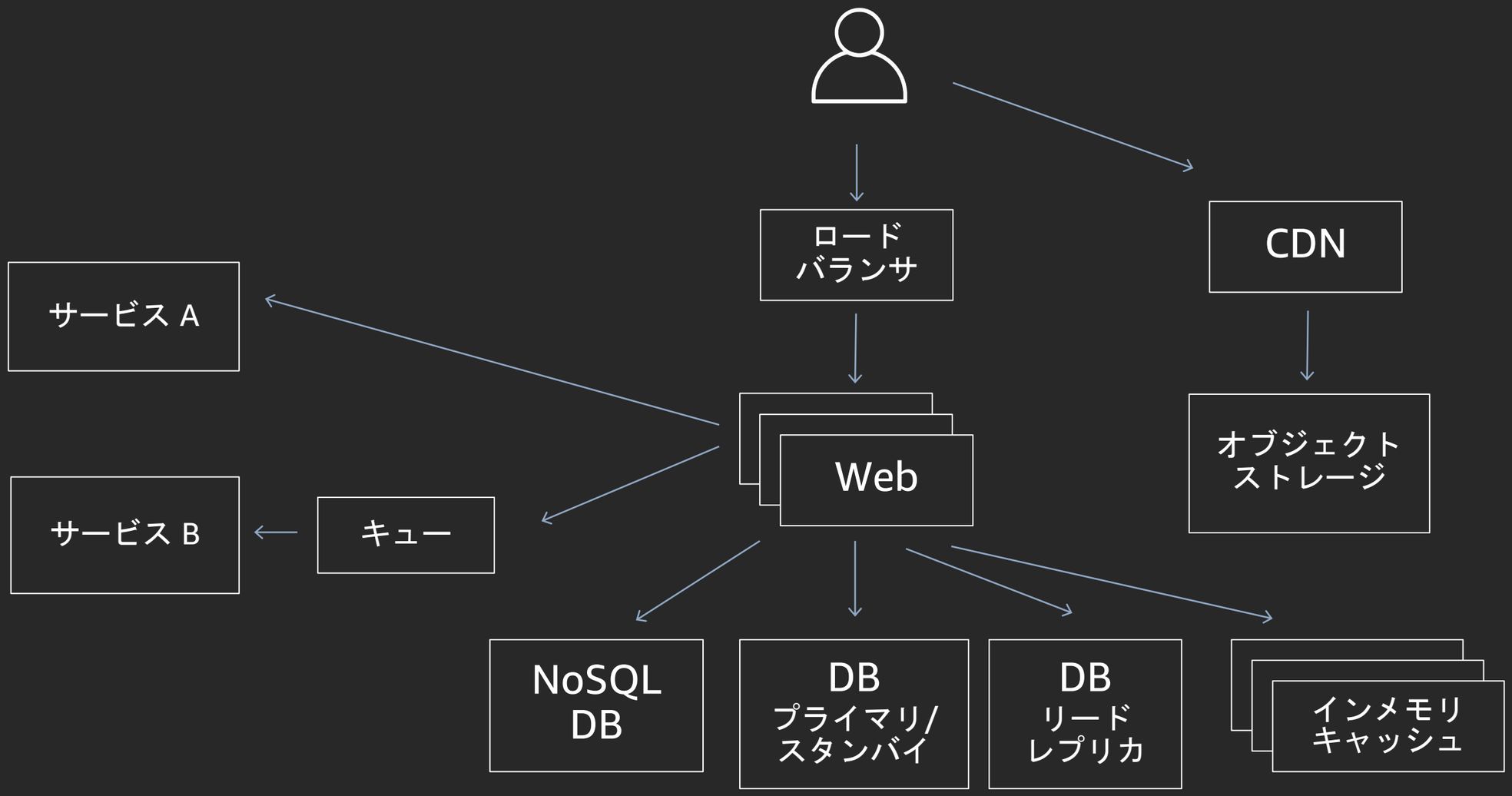
- AWS CDK
- AWS Fargate
- AWS Chime SDK

kazuya_iwami

Webアプリケーションを構築する場合、どのような構成から始めますか？



Webアプリケーションを構築する場合、どのような構成から始めますか？



フェーズや目的によって
目指すべきアーキテクチャは異なる

- 100人 | - 10,000人 | - 1,000,000人 | それ以上

本セッションのねらい

- AWSでWebサービスを構築する場合、フェーズごとに
どのAWSサービスを組み合わせて設計するとよいのかという基礎を学ぶ

- システムをスケールさせていく上でどのような課題が発生しうるかを
知り、一般的な対策を学ぶ

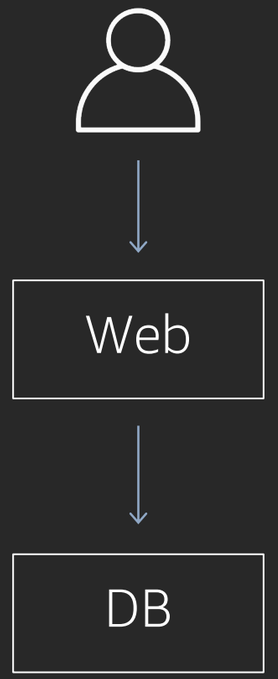
- 基礎となる仮想サーバー利用時の考え方にフォーカスします

- セキュリティに関する運用、個別のAWSサービスの詳細はお話しません

Day1

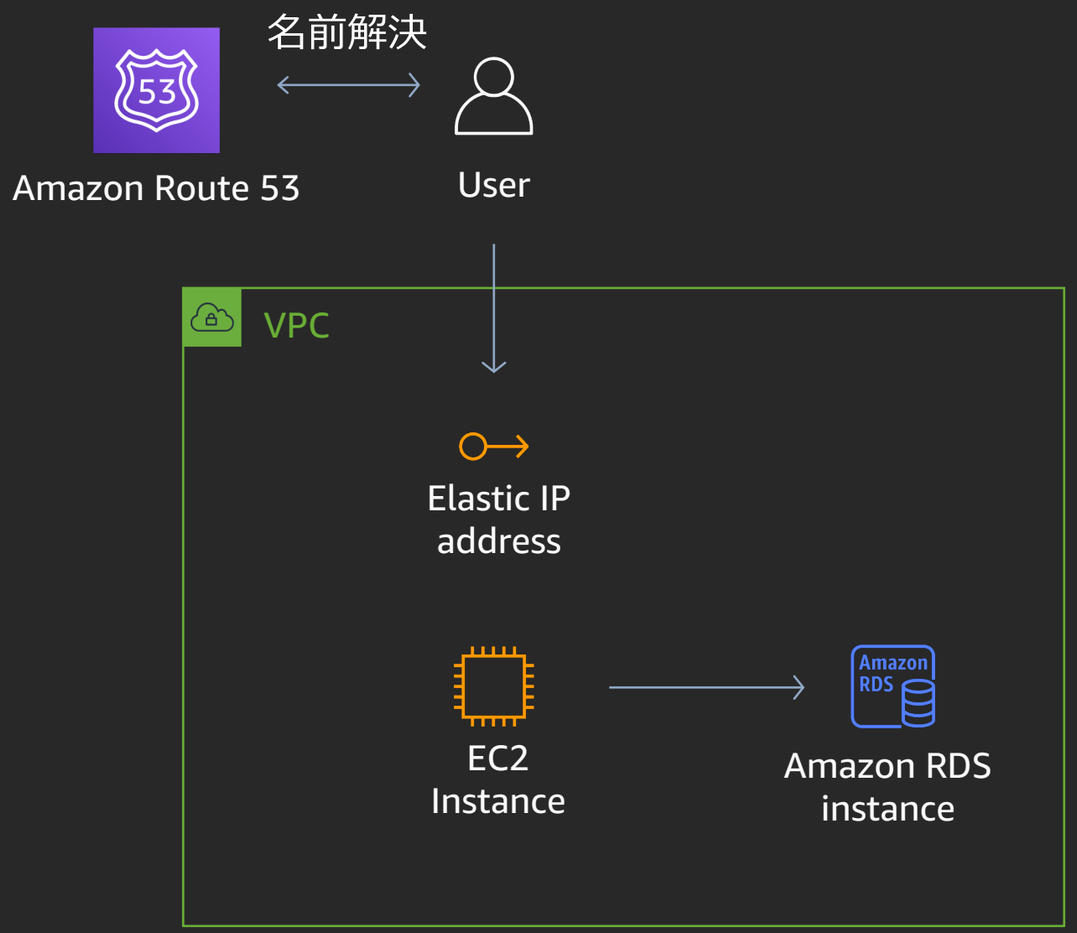
プロトタイプのフェーズ

- ユーザーは社内の数人から数十人
- シンプルで安価な構成にしたい
- 可用性や拡張性は求められていない



- 100人

AWSでの構成例



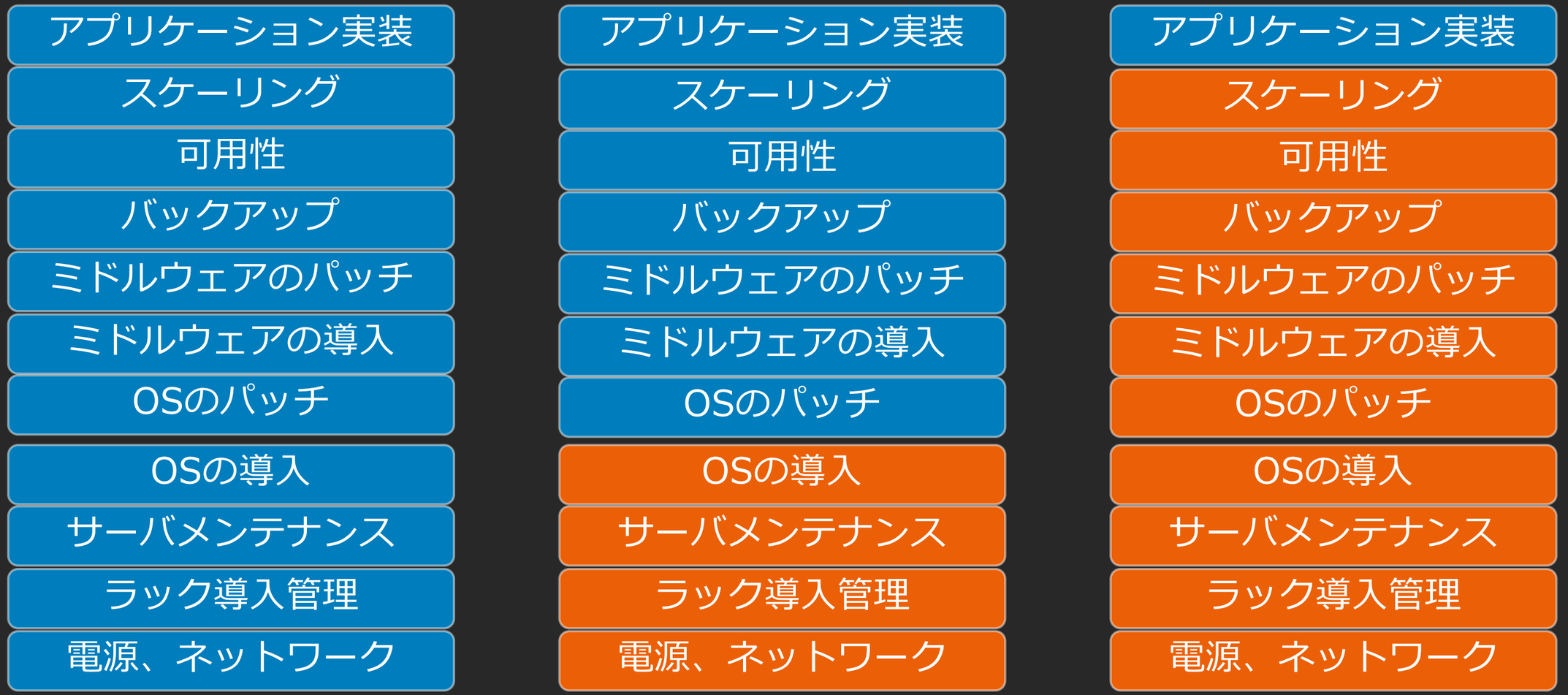
Amazon Route 53
高い可用性と信頼性を持つDNSサービス

Elastic IP Address
静的IPv4アドレス

Amazon EC2
仮想サーバーサービス

Amazon RDS
マネージドなリレーショナルデータベースサービス

データベースの選択肢

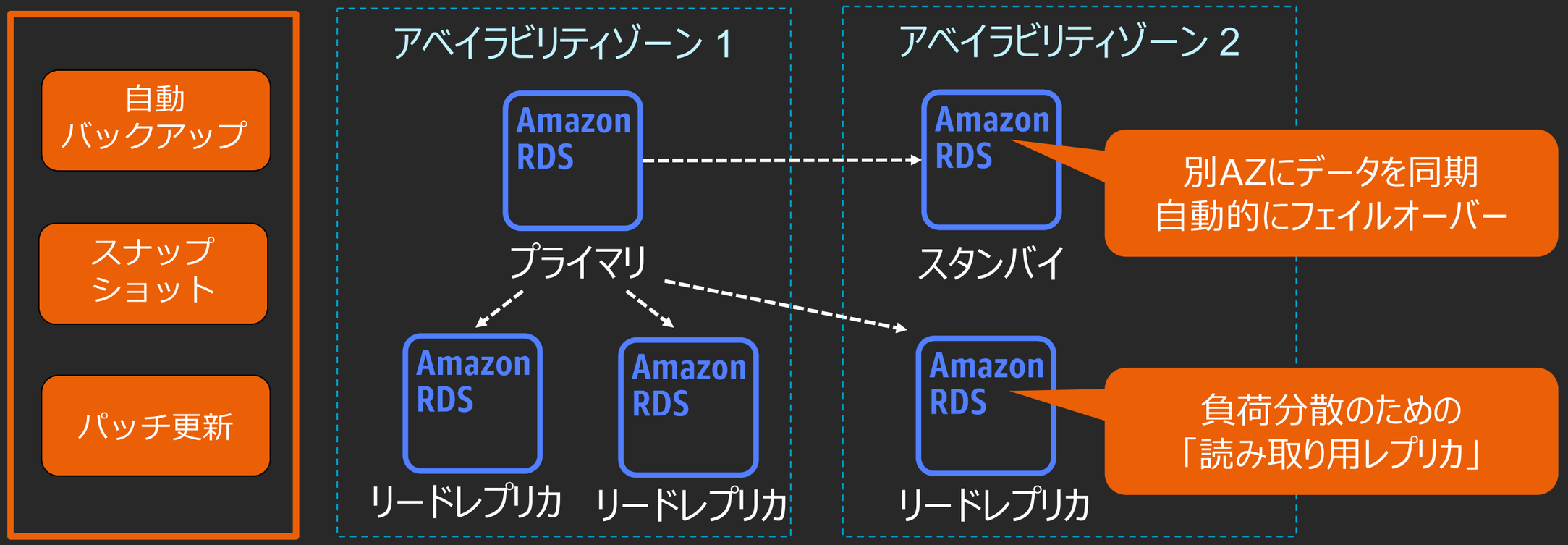


お客様がご担当する作業

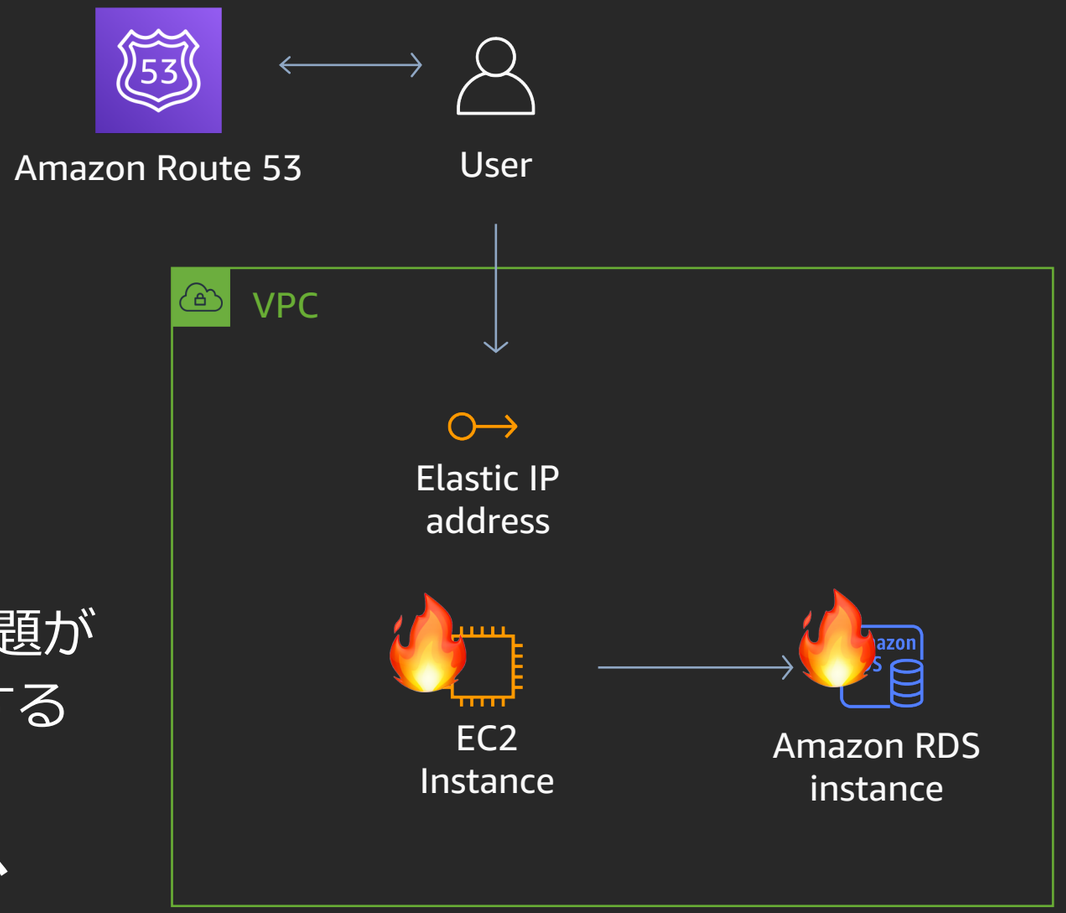
AWSが提供するマネージド機能

Amazon RDS (Relational Database Service)

- フルマネージドのRDBMSサービス
  - DBエンジンはMySQL, Oracle, SQL Server, PostgreSQL, MariaDB, Aurora から選択可能
- DBのセットアップやバックアップ、パッチ適用などをAWSが管理してくれるためメンテナンスコストを大幅に削減



現在の構成における将来的な課題



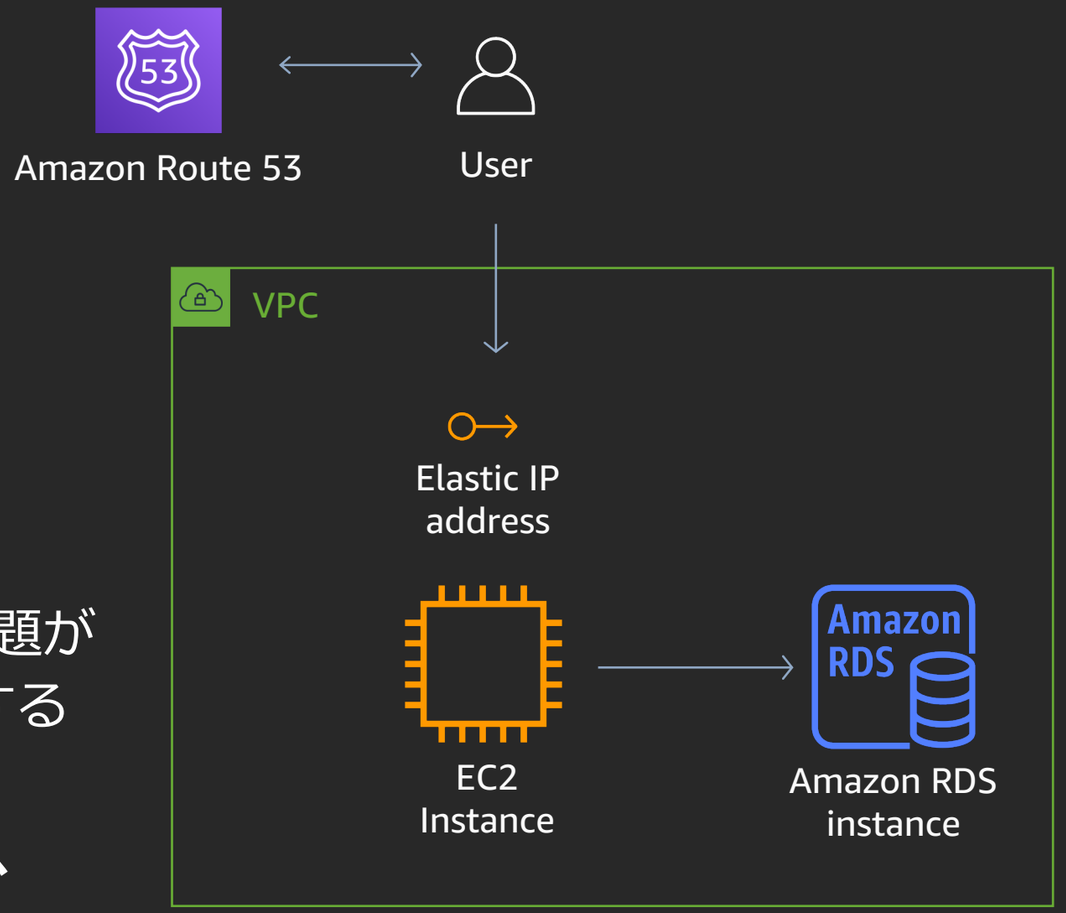
アプリ・インスタンスに問題が
生じるとサービスが停止する

II

単一障害点が複数存在し、
可用性に課題がある

「Design for Failure」という考え方が重要

現在の構成における将来的な課題



アプリ・インスタンスに問題が
生じるとサービスが停止する

II

単一障害点が複数存在し、
可用性に課題がある

ユーザーが増える度に
スケールアップする必要あり

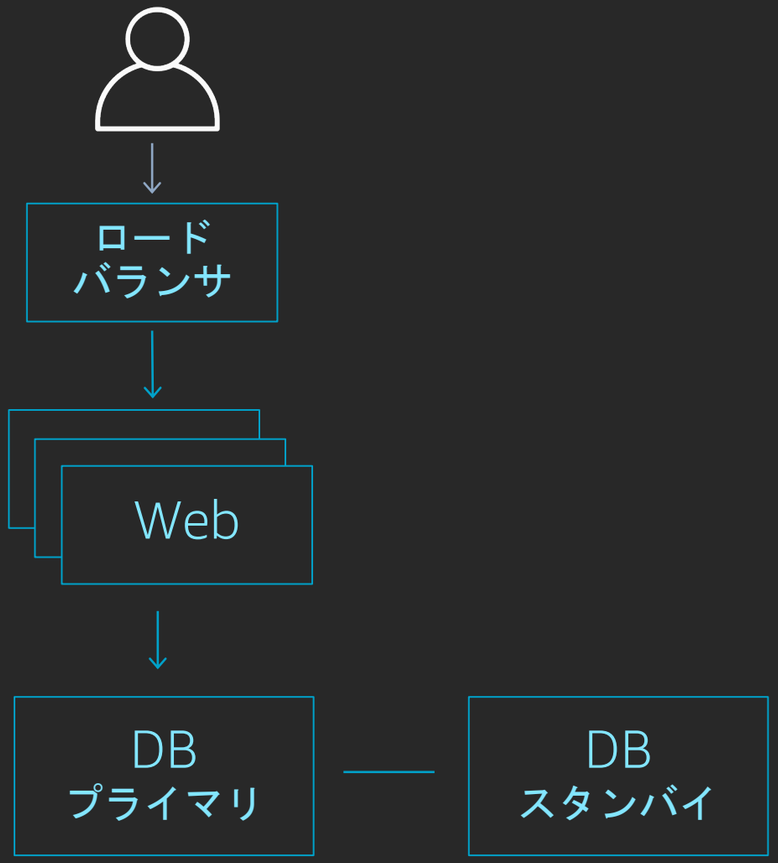
II

スケールアップには限界がある

「Design for Failure」という考え方が重要

サービスを一般公開するフェーズ

- サービスを一般向けに公開
- ユーザーは数百人から数千人
- 可用性や拡張性が求められる



- 100人
- 10,000人

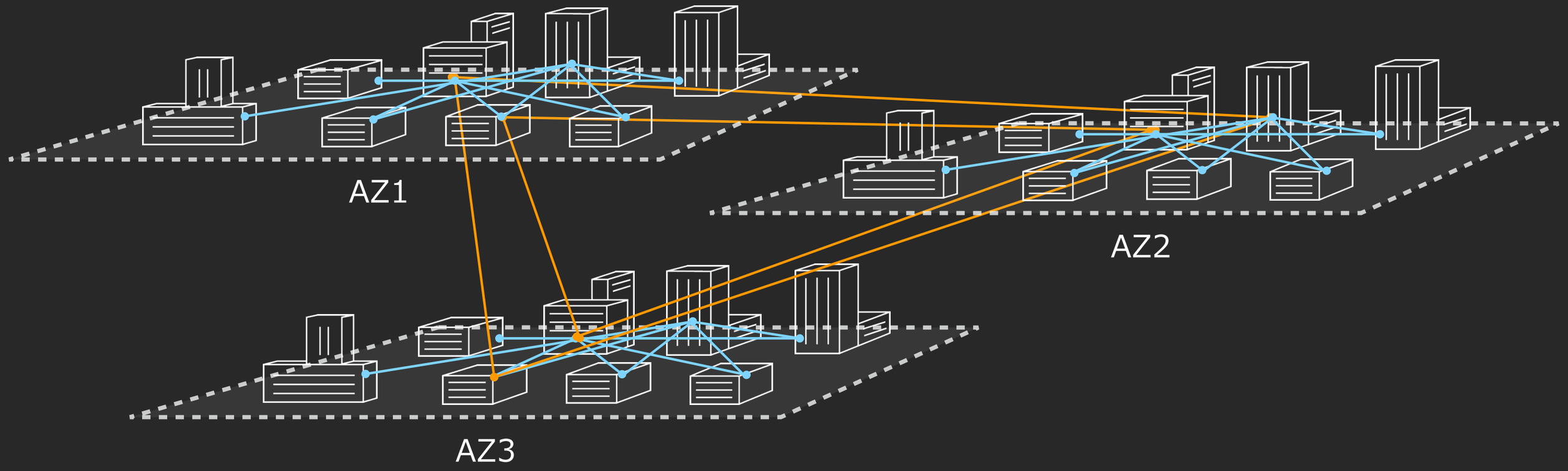
可用性向上のために

「Design for Failure」を実現するために

複数のアベイラビリティゾーンを利用した
単一障害点の無い構成を目指す

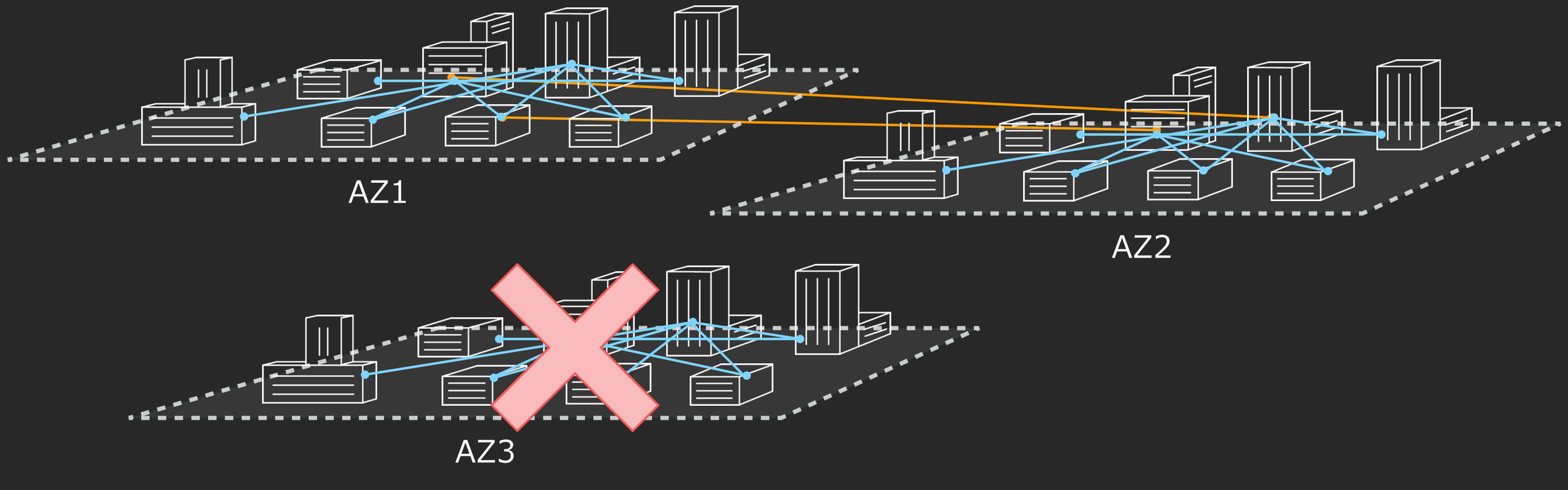
リージョン内の構成

- 複数のDCから構成されるAZ (Availability Zone)
- リージョン内に地理的に離された複数のAZを用意
- DC間は冗長化された数msecの低遅延メッシュネットワーク

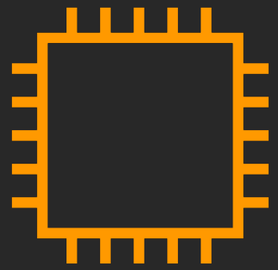


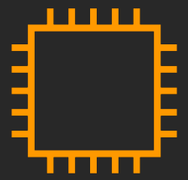
リージョン内の構成

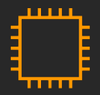
- 複数のDCから構成されるAZ (Availability Zone)
- リージョン内に地理的に離された複数のAZを用意
- DC間は冗長化された数msecの低遅延メッシュネットワーク



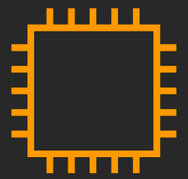
中長期的な拡張性を確保するために

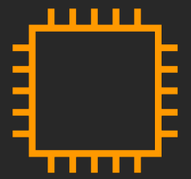


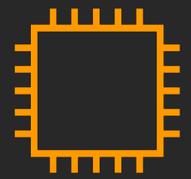




サーバー1台あたりの
リソースを強化する方法
（スケールアップ）



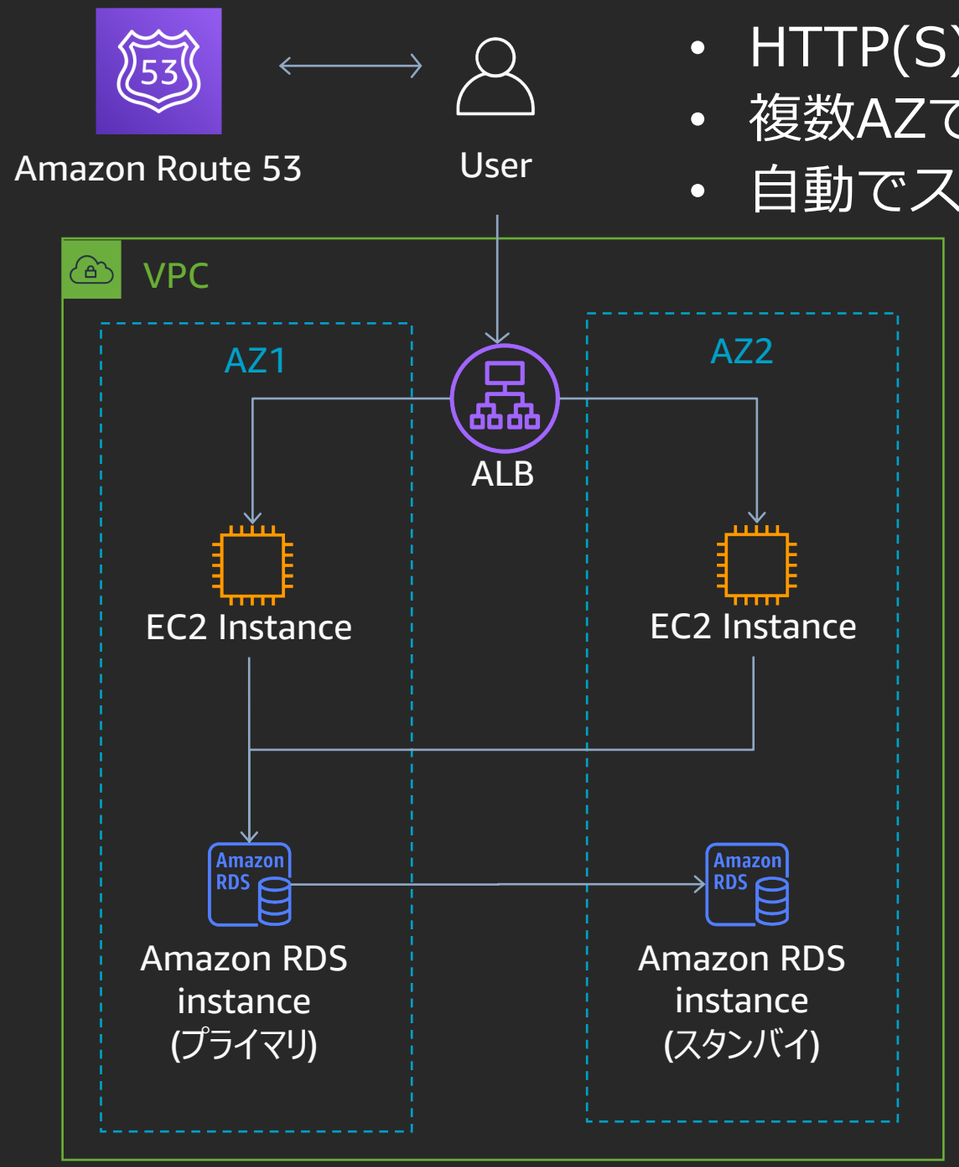




サーバーの台数を増やす方法
（スケールアウト）

Application Load Balancer (ALB)

- L7のロードバランサーサービス
- HTTP(S)のパスベースの振り分けが可能
- 複数AZで動作し、高い耐障害性を持つ
- 自動でスケーリング



複数AZに跨りEC2を配置

RDSもマルチAZ配置を設定し
異なるAZにスタンバイを配置

※ AZ = Availability Zone

- 100人 | - 10,000人

ステートレスなアプリケーションであることが大切

長時間存在するすべてのデータは、
ステートフルなバックエンドサービスに格納しなければならない

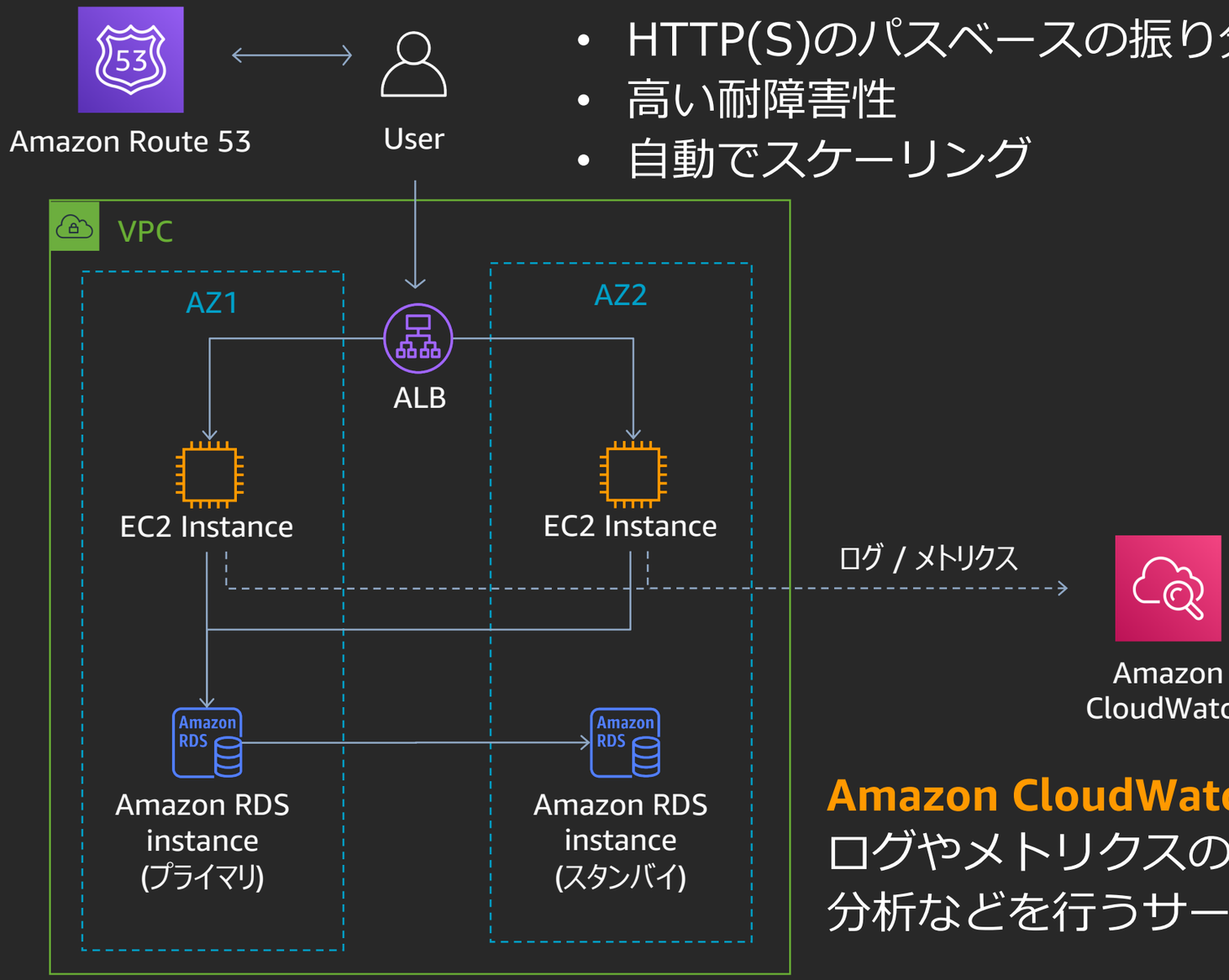
Beyond the Twelve-Factor App [1]

- セッション情報
  - Amazon RDSやAmazon DynamoDBのようなDBサービスに保存
  - もしくはAmazon ElastiCache (Memcached/Redis) のようなインメモリキャッシュに保存
- ログ
  - CloudWatch Agentを利用し、Amazon CloudWatch Logsにログを送信
  - Kinesis Agentを利用し、Amazon Kinesis Data Firehoseを経由しAmazon S3にログを送信

[1] https://www.oreilly.com/library/view/beyond-the-twelve-factor/9781492042631/

Application Load Balancer (ALB)

- L7のロードバランサーサービス
- HTTP(S)のパスベースの振り分けが可能
- 高い耐障害性
- 自動でスケーリング



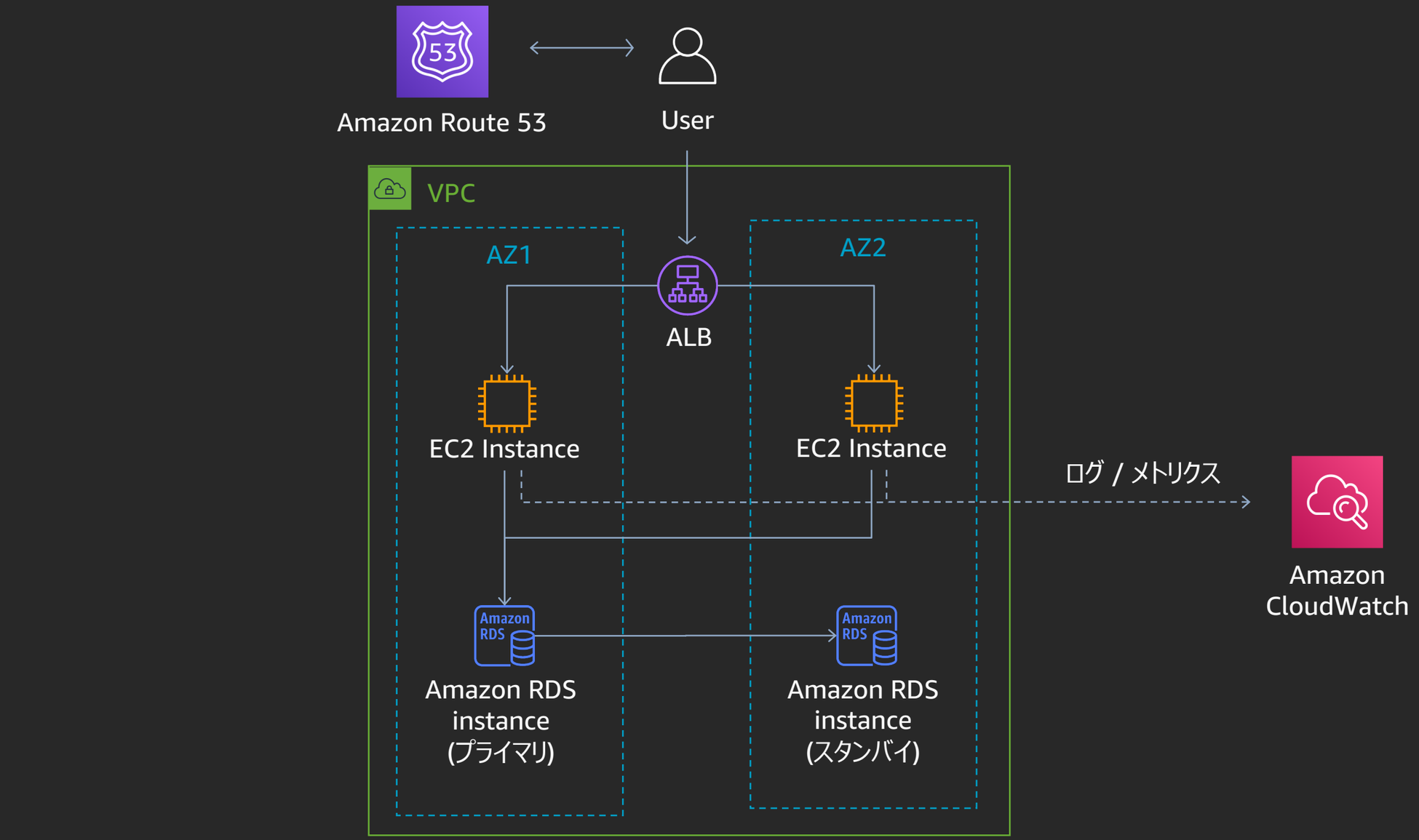
複数AZに跨りEC2を配置

RDSもマルチAZ配置を設定し
異なるAZにスタンバイを配置

Amazon CloudWatch
ログやメトリクスの収集・監視・分析などを行うサービス

- 100人 | - 10,000人

現在の構成における将来的な課題



- 100人

- 10,000人

現在の構成における将来的な課題

ユーザー増加に伴いサーバー台数が増加

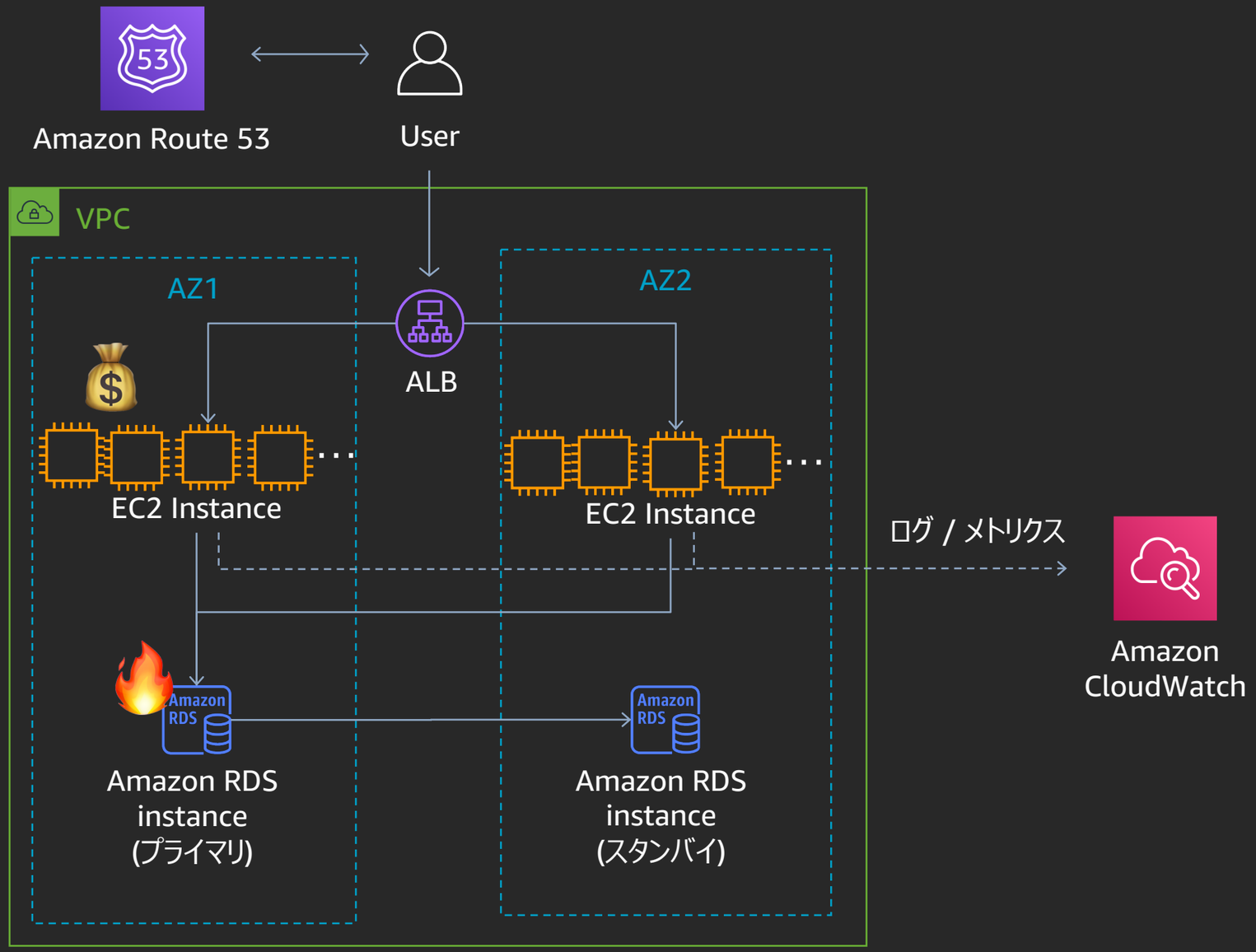
=

サーバー費用の増加
サーバー管理コストの増加

DBへのクエリ量やデータ量も増加

=

DBの読み書き負荷増加
レスポンス速度の低下

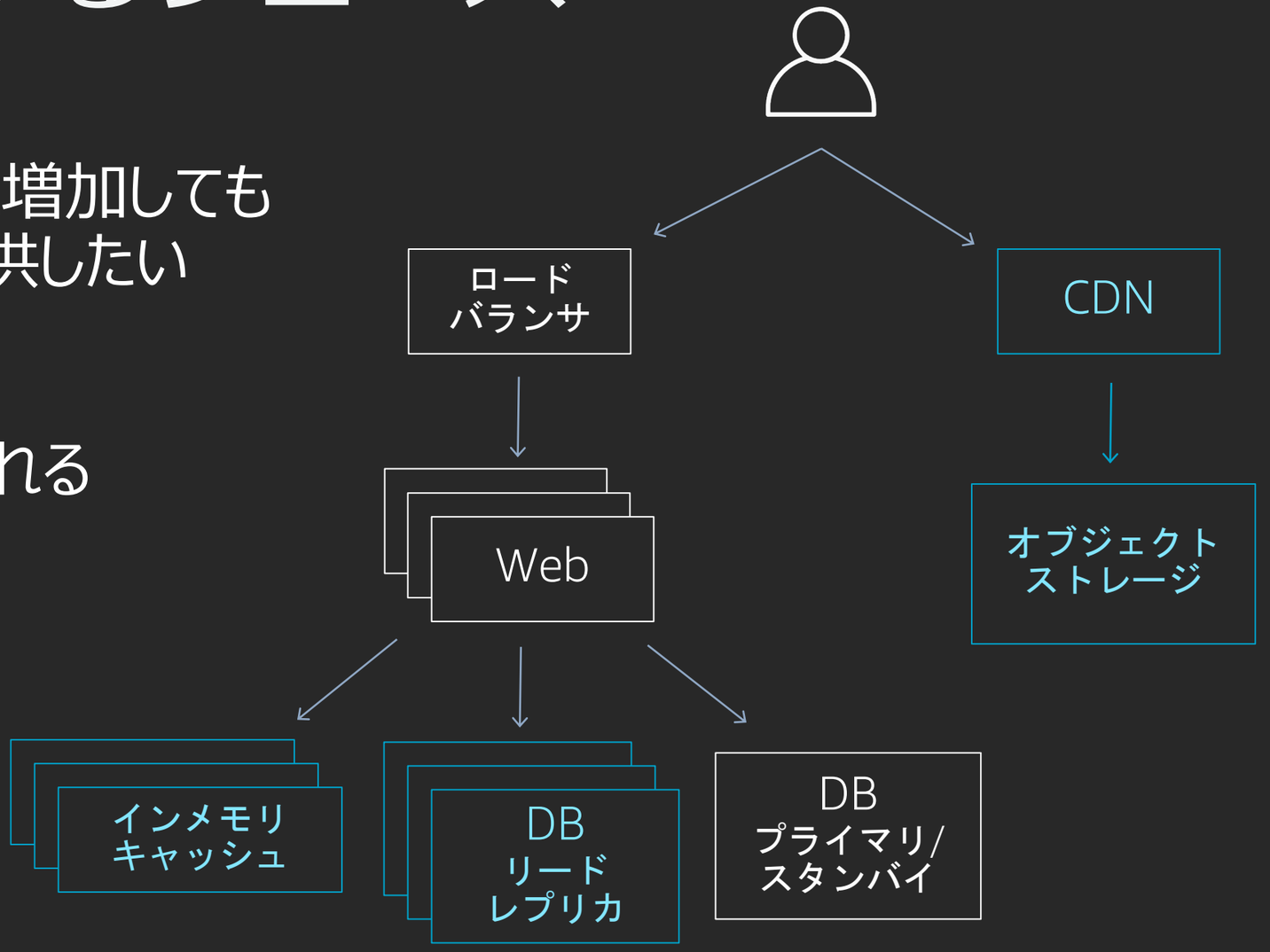


- 100人

- 10,000人

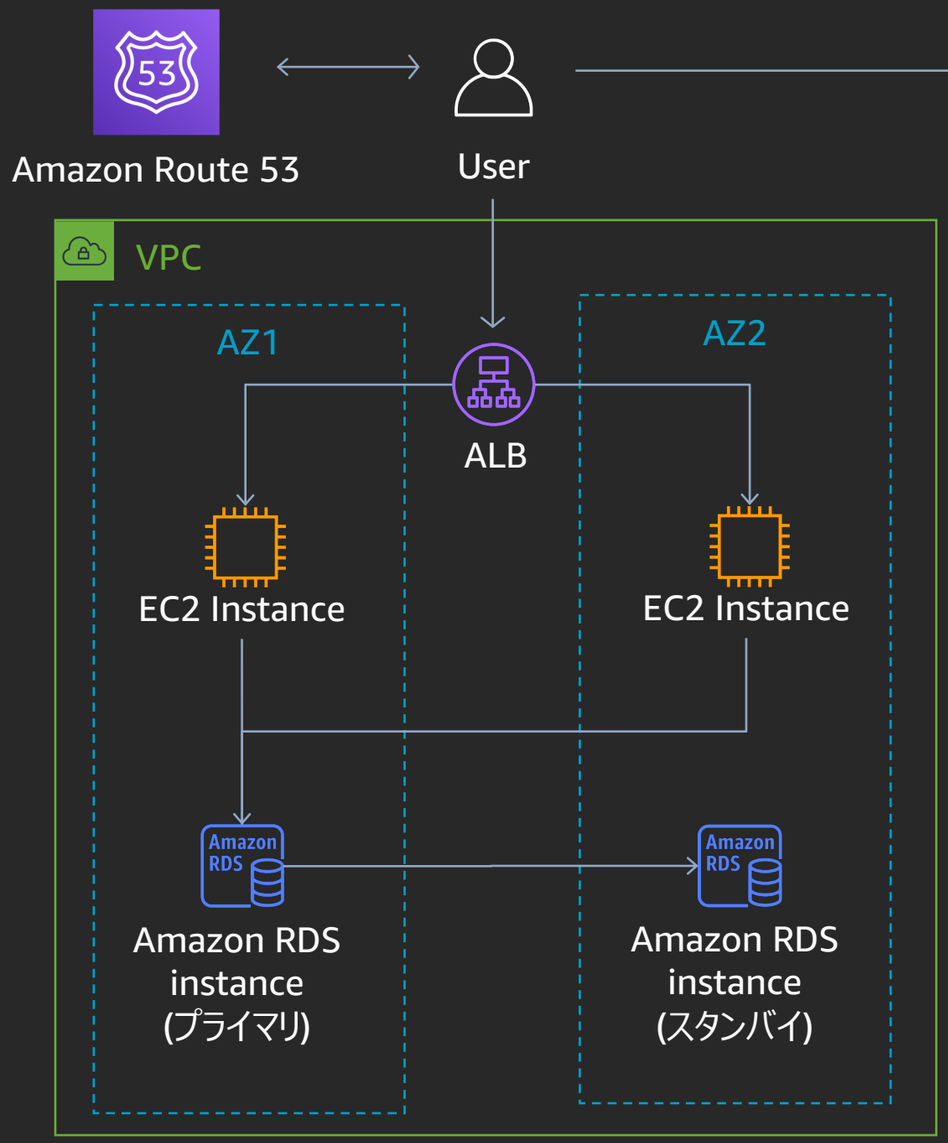
多数のユーザーが快適に利用できるよう改善するフェーズ

- ユーザーが数万人から数十万人に増加しても安定したレイテンシでサービスを提供したい
- 可用性や拡張性が求められる
- 自動化やコストの最適化が求められる



- 100人 | - 10,000人 | - 1,000,000人

静的コンテンツ配信部分をオフロード



CDN

コンテンツのキャッシュ

オブジェクトストレージ

安価で耐久性が高いストレージ

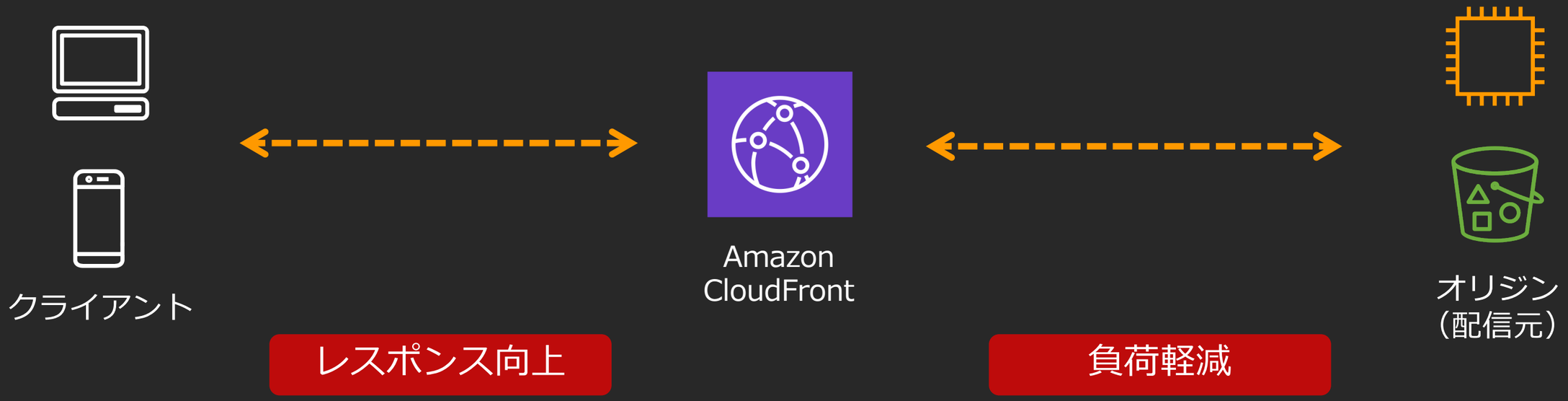
画像 / 動画 / データファイル / HTML / CSS / JavaScript

- 100人 | - 10,000人 | - 1,000,000人

Amazon CloudFront

CDN (Content Delivery Network) とは
- 大容量キャパシティを持つ地理的に分散したサーバー群(エッジ)から
  コンテンツをキャッシュしたり代理配信をするサービス

CDN導入の利点
- ユーザーを一番近いエッジロケーションに誘導することで 配信を高速化
- エッジサーバでコンテンツのキャッシングを行い オリジンの負荷をオフロード

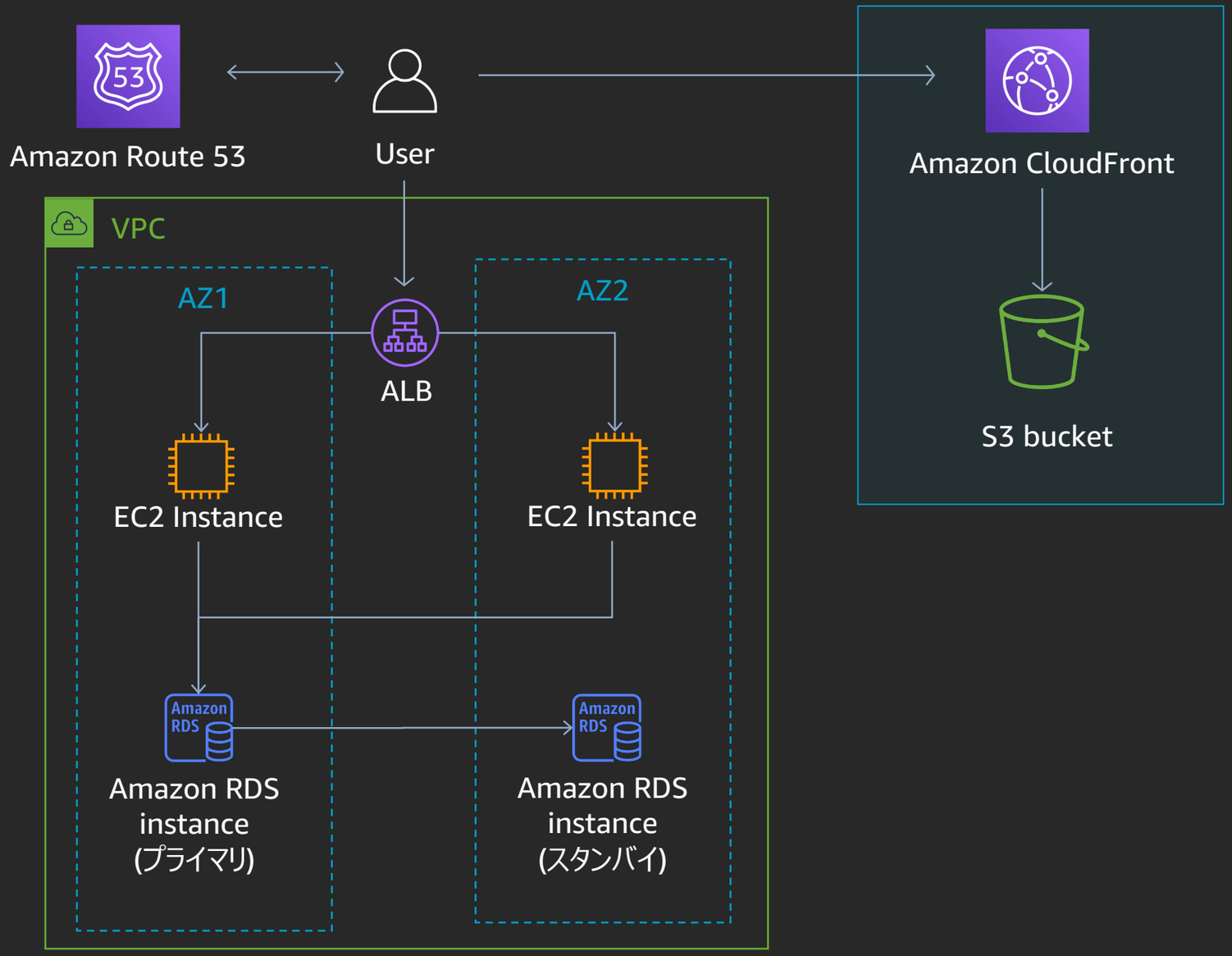


Amazon S3 (Simple Storage Service)

- データ保存・バックアップ用途に向くオブジェクトストレージ
- S3自体に静的コンテンツ配信機能あり
- 自動的に3箇所以上のAZに隔地保管
- データ耐久性は、99.999999999%
- 容量無制限、サイジング不要
- REST APIを介してデータを操作
- 1プレフィックス辺り秒間5500回以上GETリクエストが可能
- 従量課金 1GByteあたり月間 $0.0125~
- 長期アーカイブ用なら$0.00099～

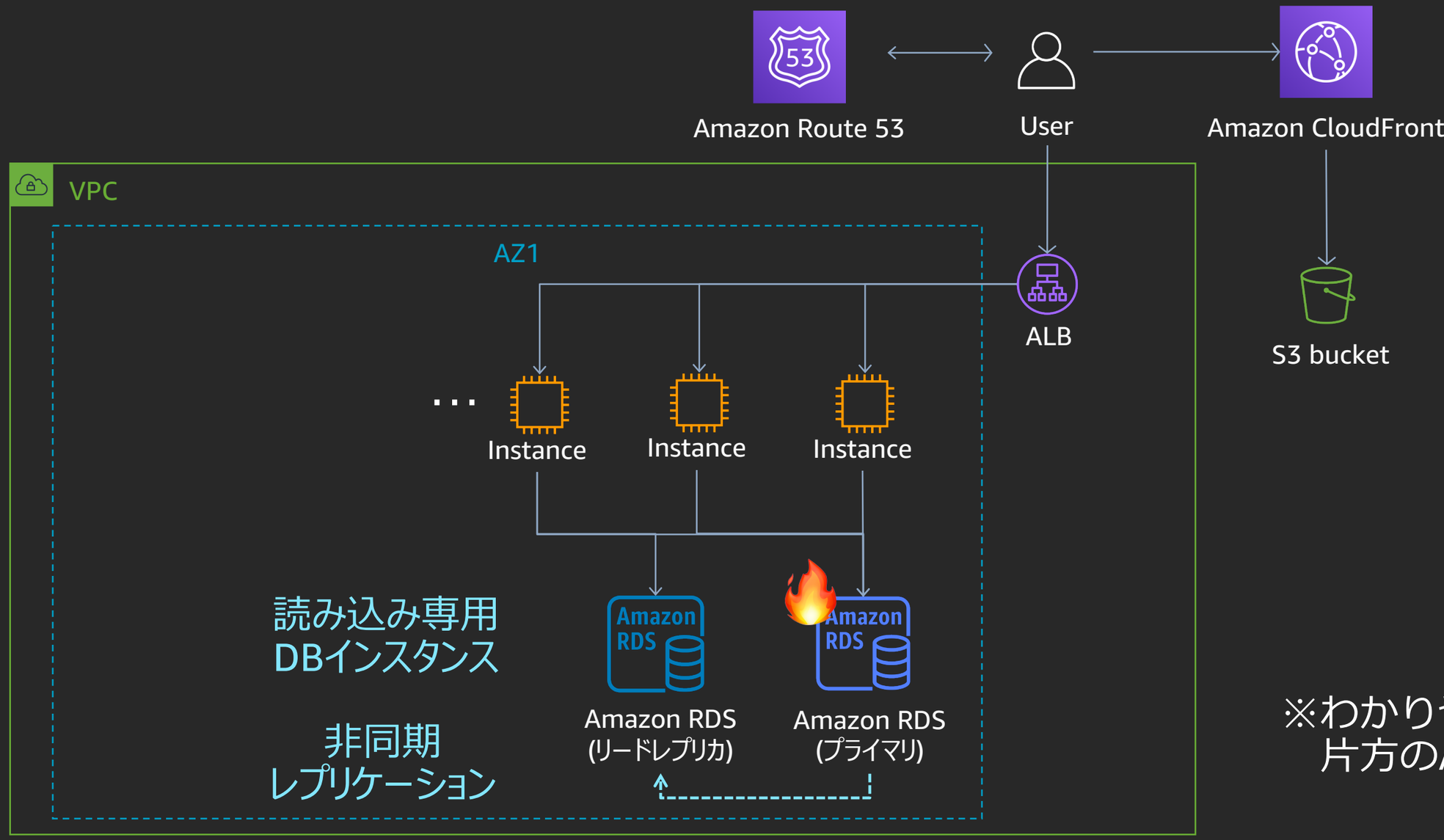
※ 2020年6月時点での米国東部の金額

静的コンテンツ配信部分をオフロード



- 100人 - 10,000人 - 1,000,000人

データベースへの読み込みクエリをオフロード



※わかりやすさのため
片方のAZのみ記載

- 100人 - 10,000人 - 1,000,000人

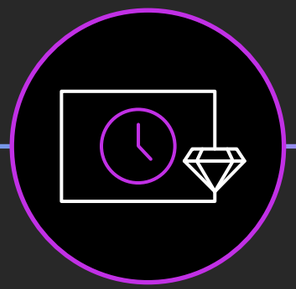
Amazon Aurora

クラウド向けに再設計された MySQL, PostgreSQL と互換性のあるRDBMS
コマーシャルデータベースの性能と可用性を1/10のコストで

優れた性能と拡張性

標準的なMySQLと比べて5倍、
標準的なPostgreSQLと比べて
3倍のスループットを実現;
リードレプリカを最大 15 個
追加してスケールアウト可能

高可用性と耐久性



耐障害性、自己修復機能を
兼ね備えたストレージ;
3 つのAZにわたり、6 個の
コピーを保持;
Amazon S3への継続的な
バックアップ

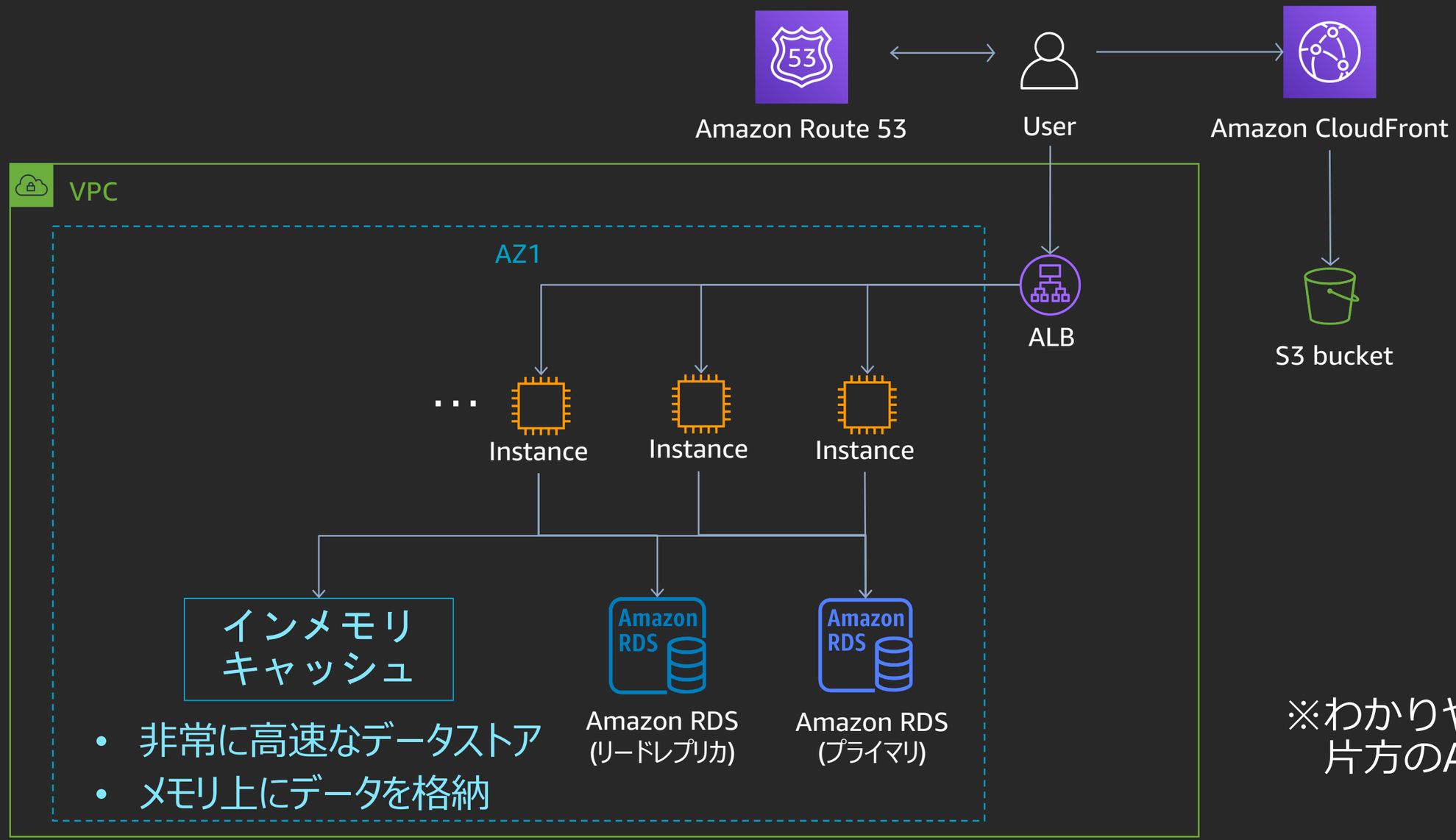
高い安全性

ネットワーク分離、
保管時/通信の暗号化

フルマネージド

ハードウェアの用意、
ソフトウェアのパッチ適用、
セットアップ、構成、
バックアップといった
管理タスクからの解放

頻繁に利用されるデータをキャッシュ



※わかりやすさのため
片方のAZのみ記載

- 100人
- 10,000人
- 1,000,000人

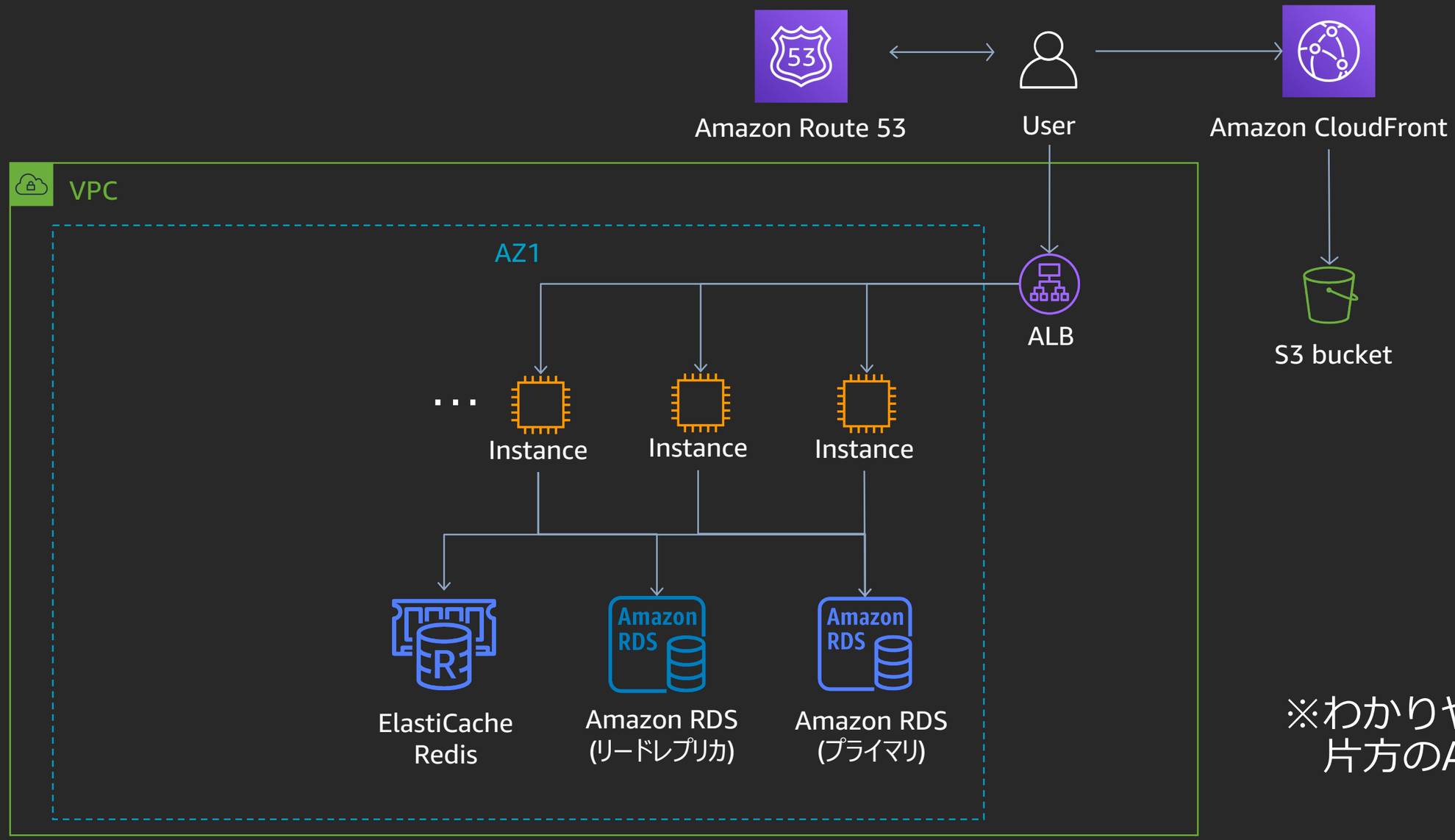
Amazon ElastiCache

- フルマネージド
  - インメモリキャッシュを数クリックで起動、実行、スケーリング可能
  - 可用性を向上させる機能
    （モニタリング、自動障害検出、復旧、拡張、パッチ適用、バックアップ）

- 非常に高いパフォーマンス
  - 非常に高速な2種類のエンジン(**Memcached, Redis**)をサポート
  - ミリ秒未満の応答時間を実現
  - AmazonによるRedisの機能拡張

- 主な用途
  - キャッシュ、セッション管理、リアルタイム集計

頻繁に利用されるデータをキャッシュ



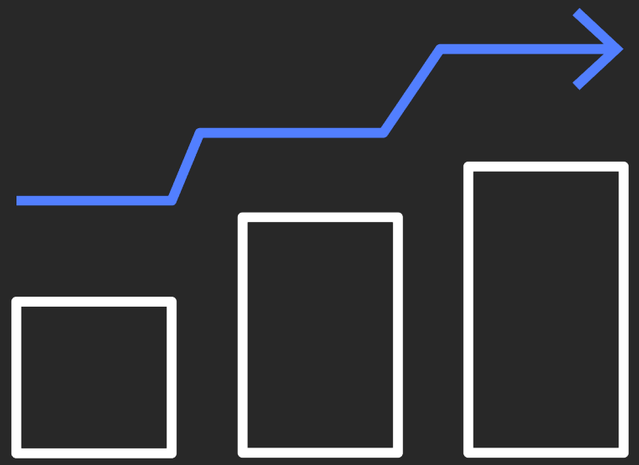
※わかりやすさのため
片方のAZのみ記載

- 100人
- 10,000人
- 1,000,000人

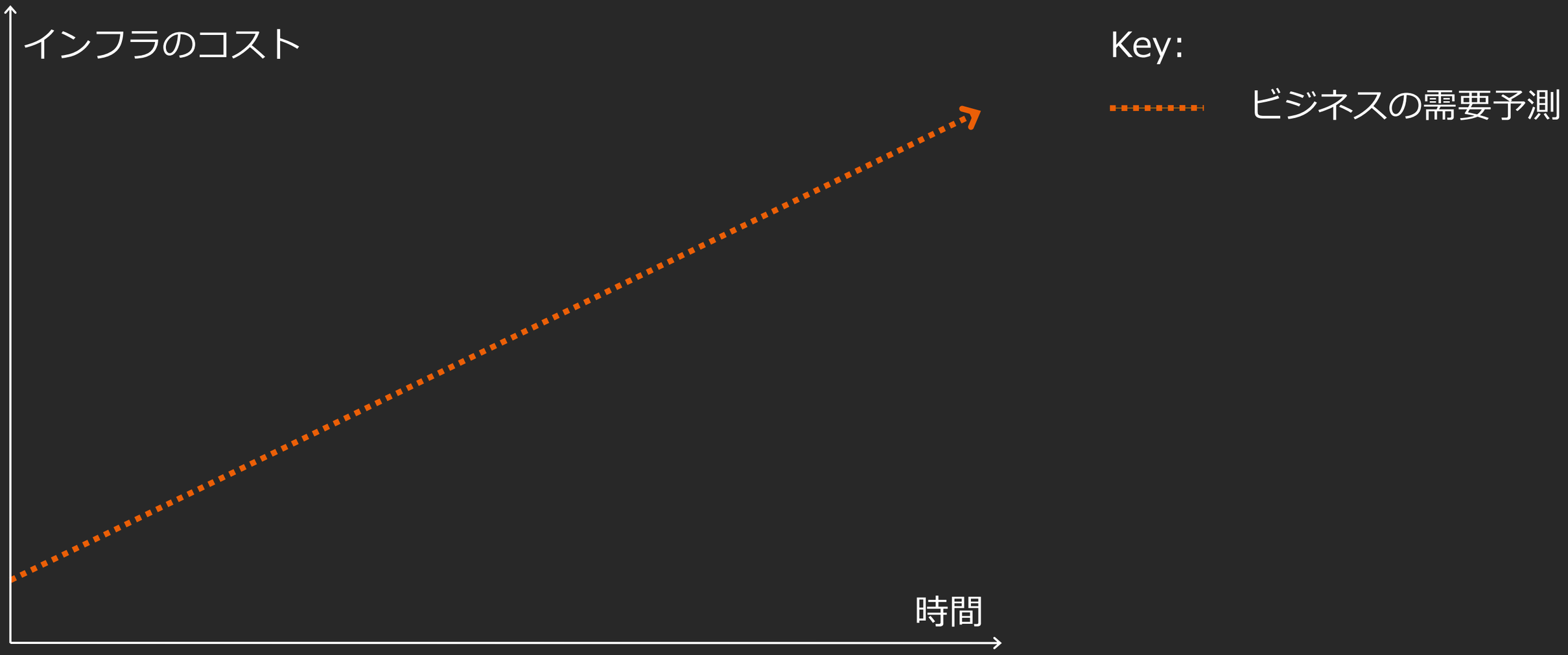
Auto Scaling

Webサーバーのコスト最適化や、複数のEC2の管理の自動化のために

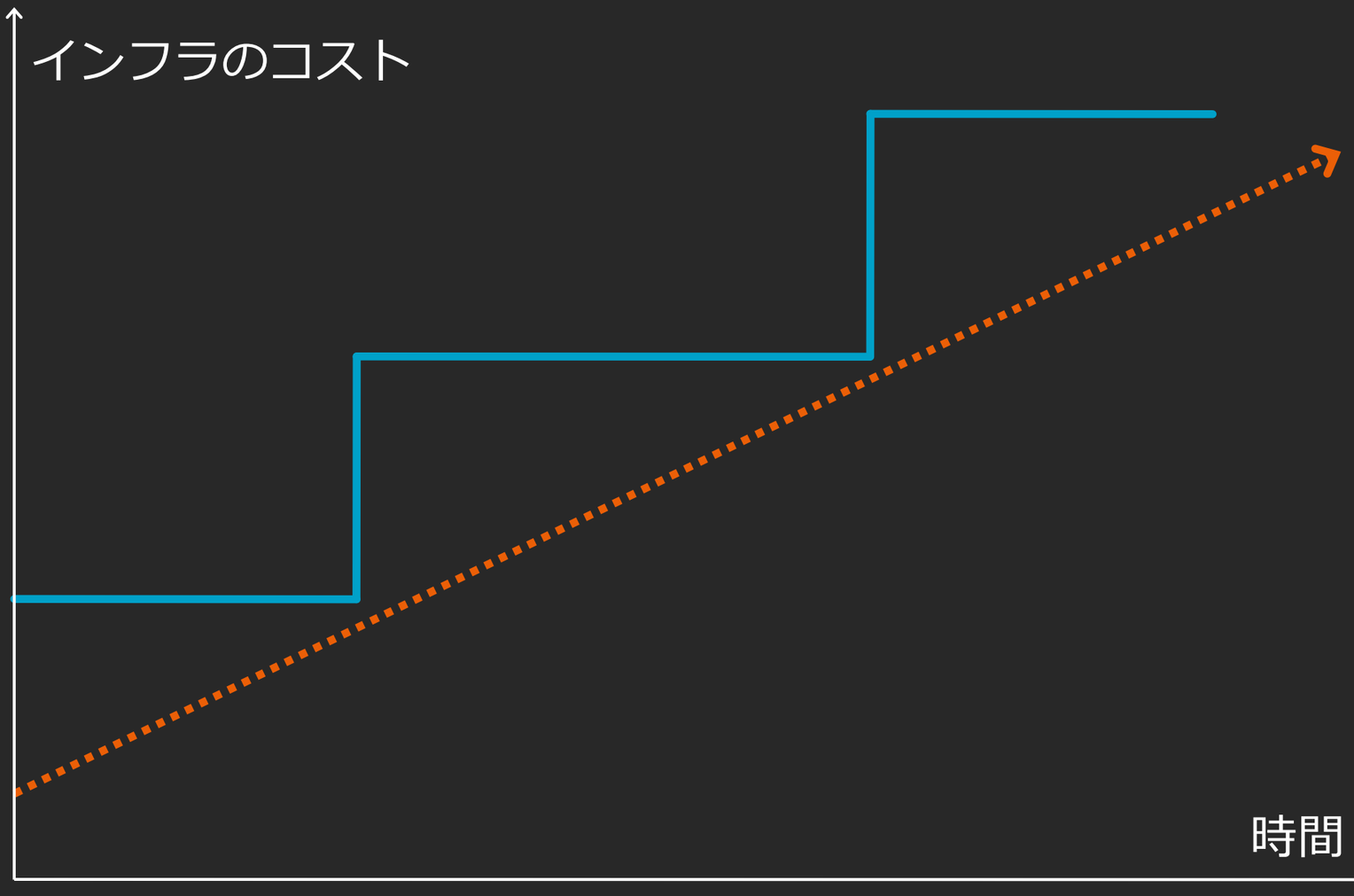
Auto Scalingの導入を検討



ビジネス需要に応じたキャパシティ準備



ビジネス需要に応じたキャパシティ準備

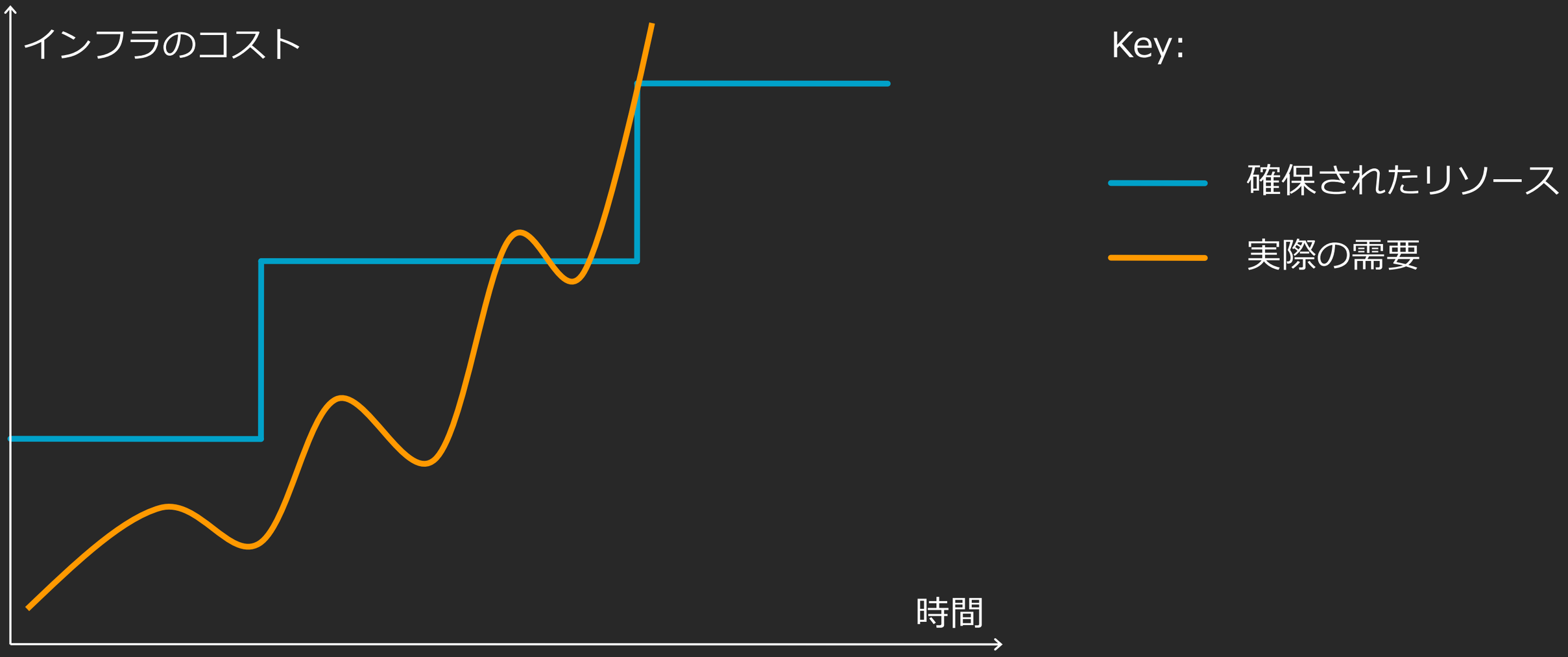


Key:

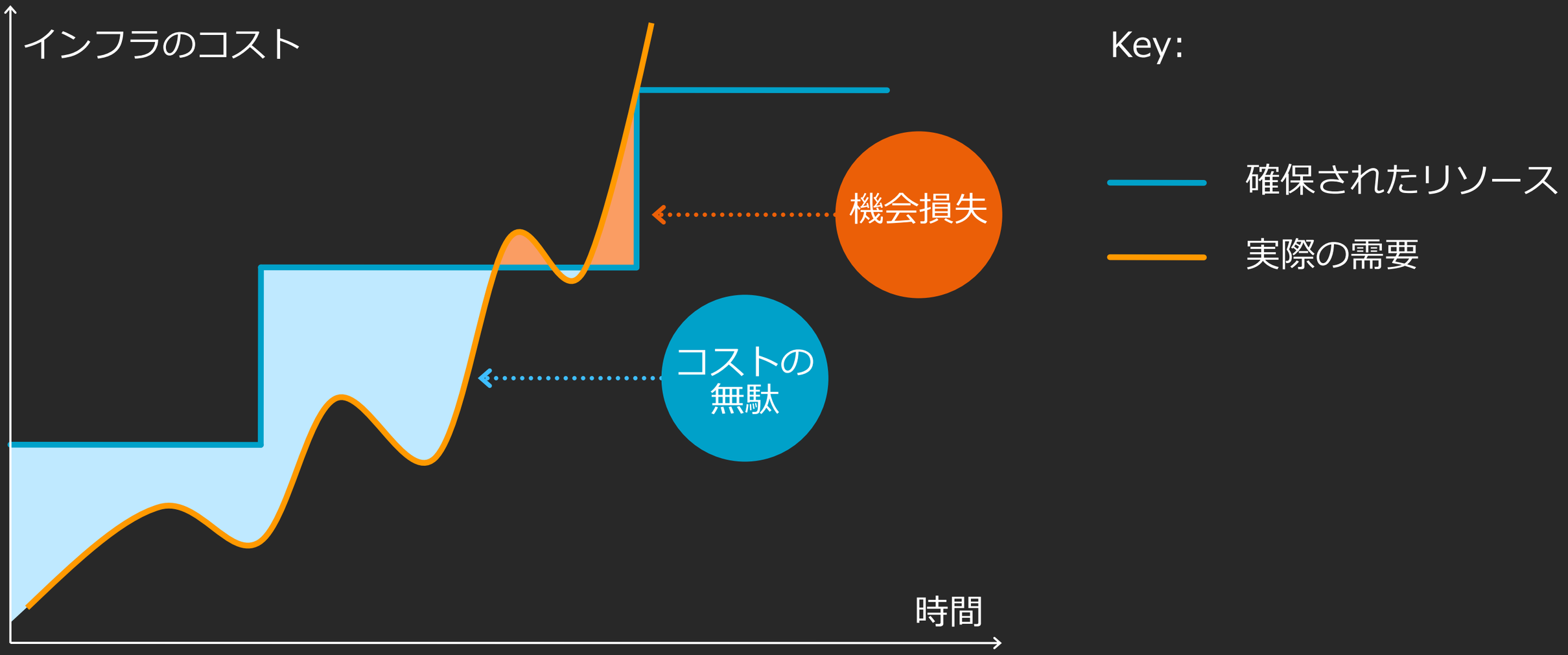
ビジネスの需要予測

確保されたリソース

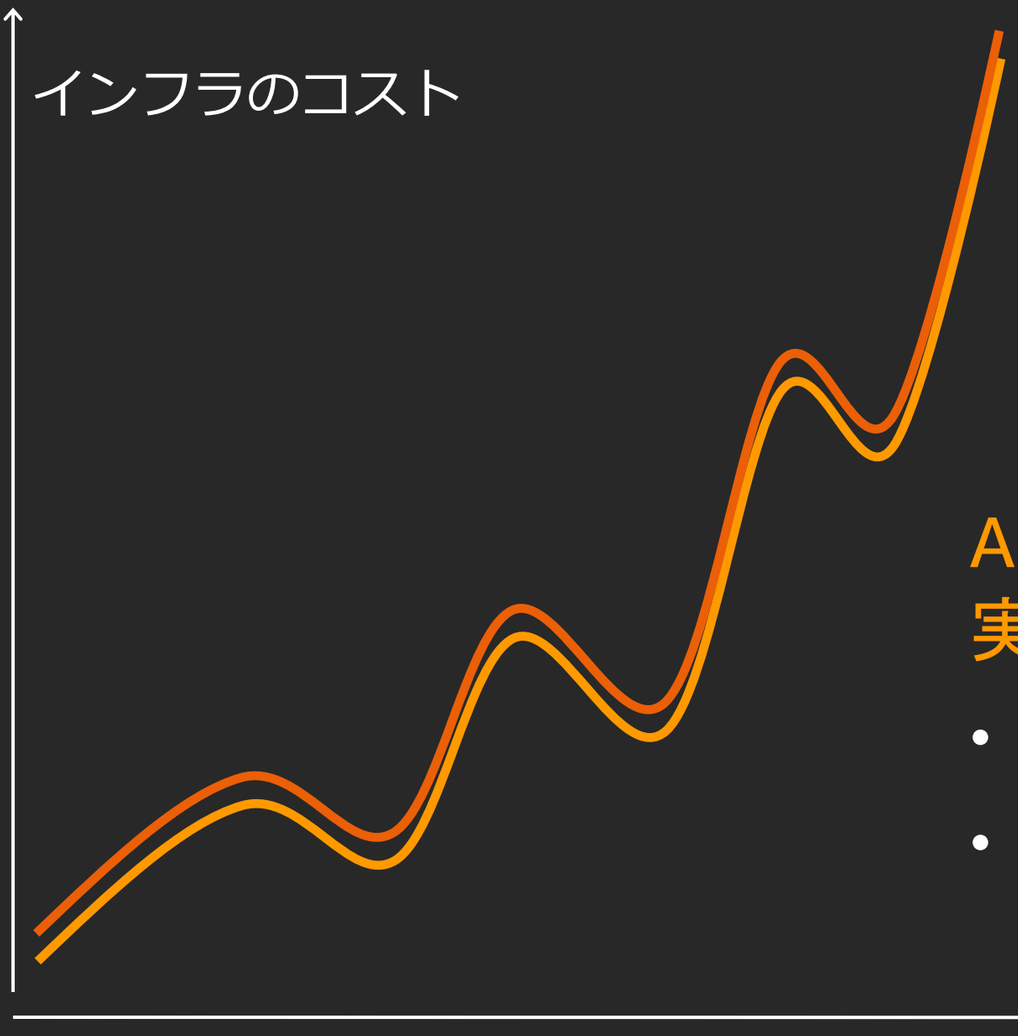
ビジネス需要に応じたキャパシティ準備



ビジネス需要に応じたキャパシティ準備



ビジネス需要に応じたキャパシティ準備



Key:

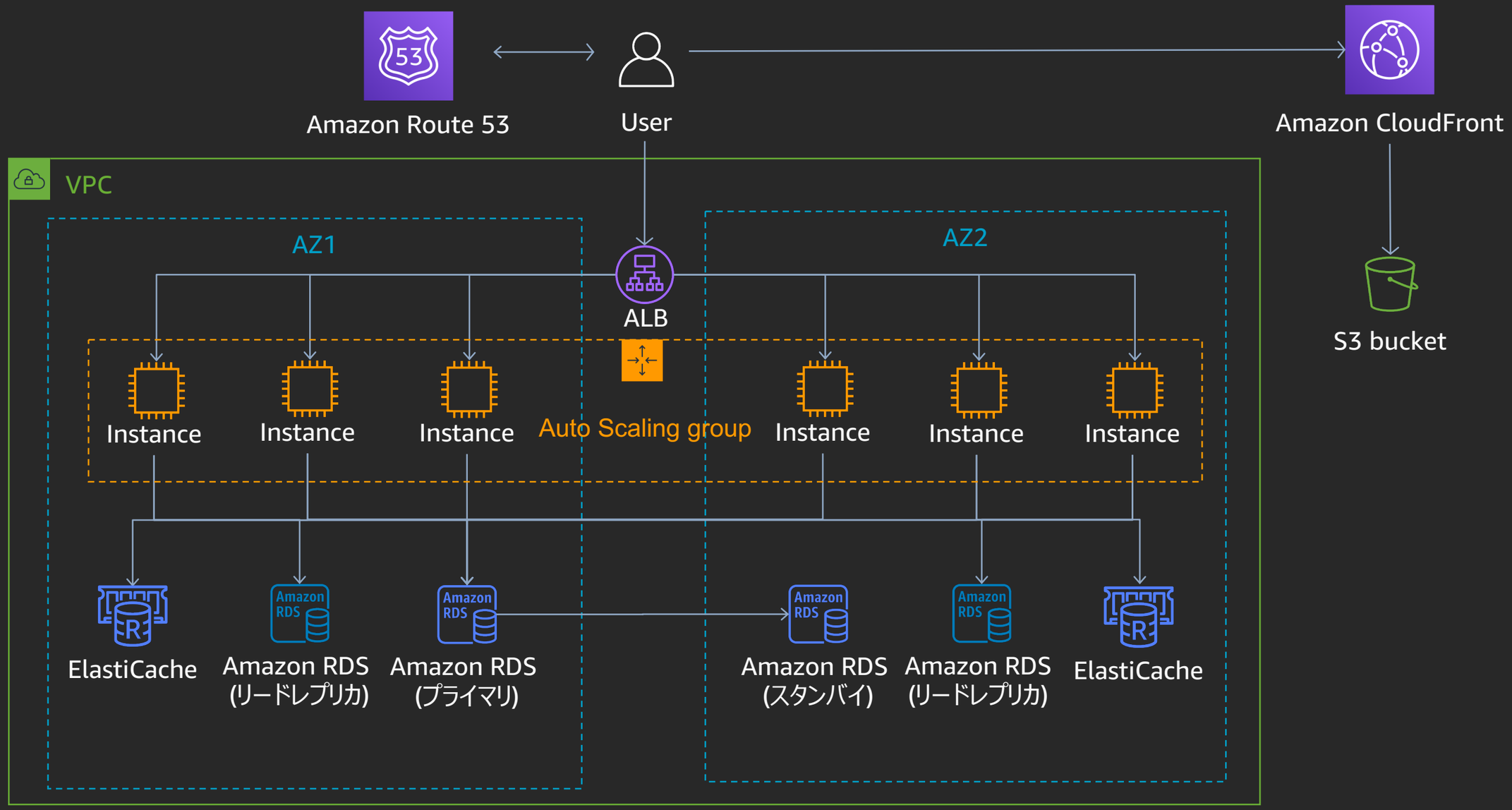
実際のAWSリソース

実際の需要

Auto Scalingサービス群により
実際の需要に合わせたオートスケールが可能

- 確保しすぎることによる無駄を極小化
- 機会損失を極小化

ここまでの振り返り



- 100人 - 10,000人 - 1,000,000人

ここまでの振り返り（補足）

- 想定される課題に応じて、各コンポーネントの導入順序は変わりうる
- 次のフェーズでのコンポーネントの導入を見越したアプリケーションの設計にできると望ましい
- 負荷テストを十分に行うことが大切
- AWSのサービス上限にも注意し、AWSサポートに上限緩和を申請

現在の構成における将来的な課題

開発人数やコードベースの拡大

=

アプリケーションの変更に時間が掛かる

次第にレスポンスが遅くなってきた

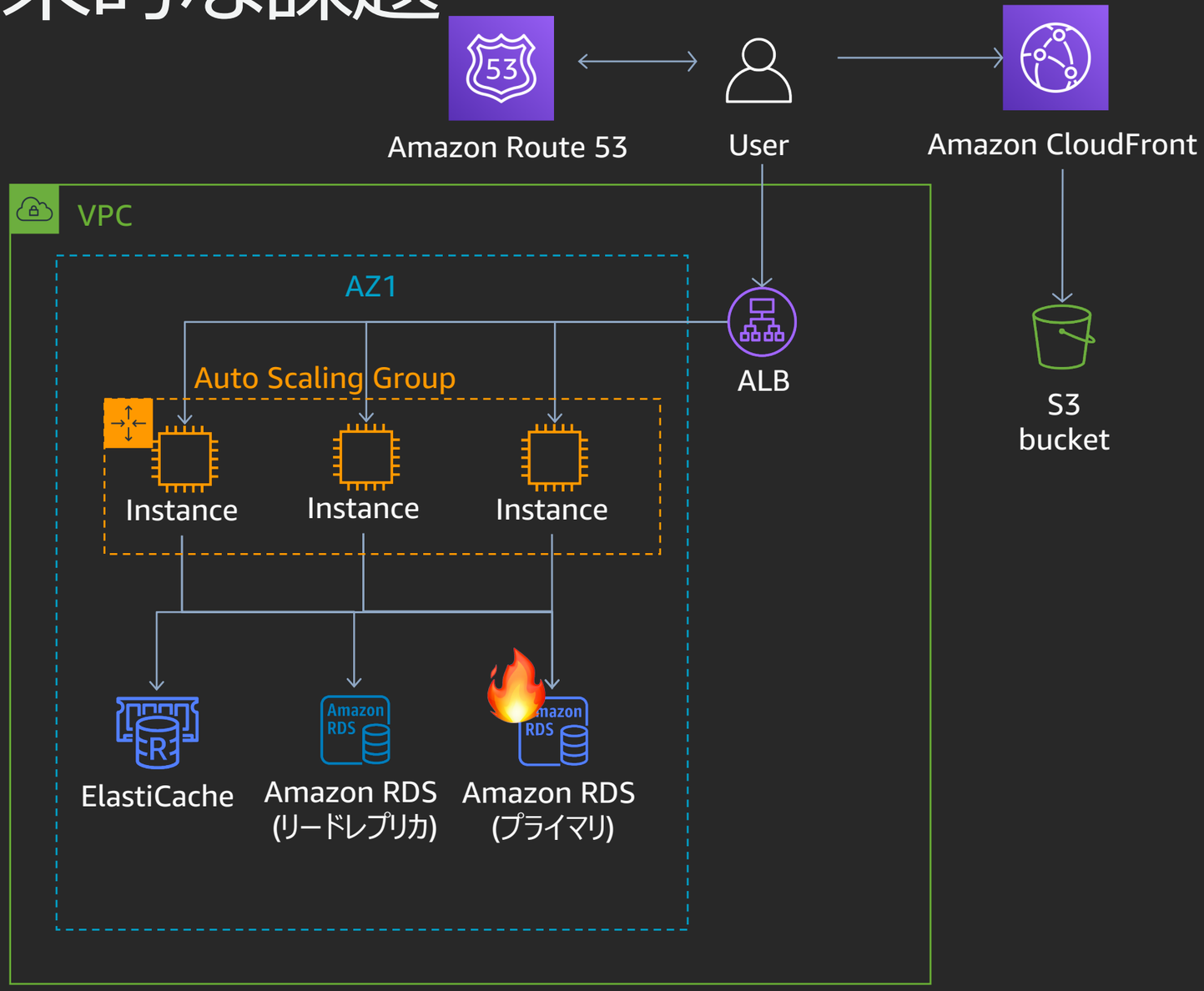
=

アプリ側の様々なボトルネック

DBの書き込み負荷増大

=

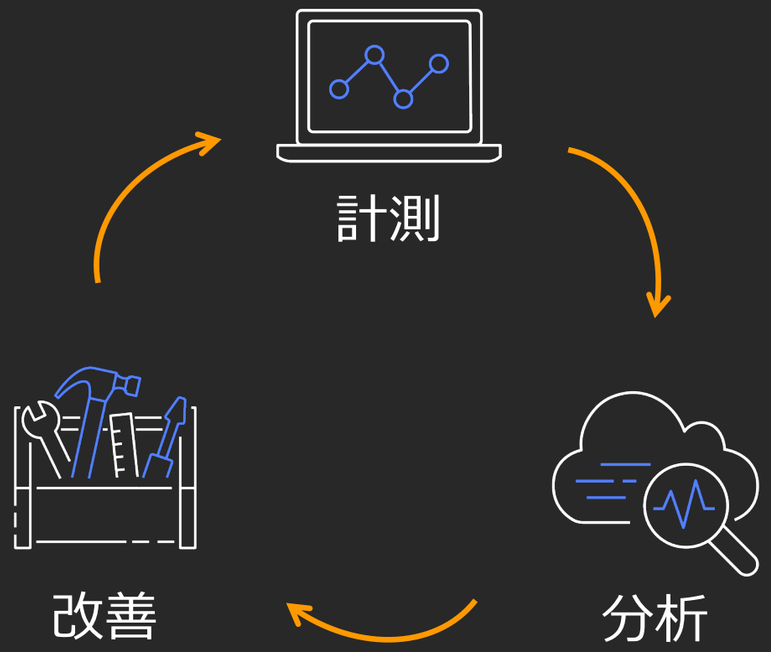
プライマリDBのスケールアップの限界



- 100人 - 10,000人 - 1,000,000人

更に意識して計測・分析・改善のサイクルを回すフェーズ

- ユーザーが数百万人以上に増加しても安定したレイテンシでサービスを提供したい
- 可用性や拡張性が強く求められる
- 自動化やコストの最適化が強く求められる
- 組織や運用もスケールする設計が求められる



- 100人 | - 10,000人 | - 1,000,000人 | それ以上

具体的な改善の例

アプリケーションパフォーマンスの改善のために

- アプリケーションのプロファイリングの実施や、メトリクス・ログを詳細に出力し分析
- アプリケーションロジックやSQLの改善
- 様々なレイヤーでのキャッシュの利用
- 非同期処理化

RDSプライマリインスタンスへの負荷集中対策のために

- DBのシャーディング
- NoSQLの利用

運用の改善のために

- 様々なレイヤーでマネージドサービスを活用し、可用性やスケーラビリティを担保
- サービスをモジュール、コンポーネント、マイクロサービスなどの単位で分割
- 運用オペレーションの自動化
- Infrastructure as CodeやCI/CDの導入

具体的な改善の例

アプリケーションパフォーマンスの改善のために

- アプリケーションのプロファイリングの実施や、メトリクス・ログを詳細に出力し分析
- アプリケーションロジックやSQLの改善
- 様々なレイヤーでのキャッシュの利用
- 非同期処理化

RDSプライマリインスタンスへの負荷集中対策のために

- DBのシャーディング
- NoSQLの利用

運用の改善のために

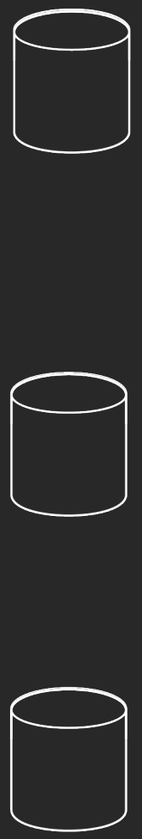
- 様々なレイヤーでマネージドサービスを活用し、可用性やスケーラビリティを担保
- サービスをモジュール、コンポーネント、マイクロサービスなどの単位で分割
- 運用オペレーションの自動化
- Infrastructure as CodeやCI/CDの導入

RDSにおけるDBの分割方法

Forum DB

User DB

Product DB



垂直分割

User	ShardID
002345	A
002346	B
002347	C
002348	B
002349	A

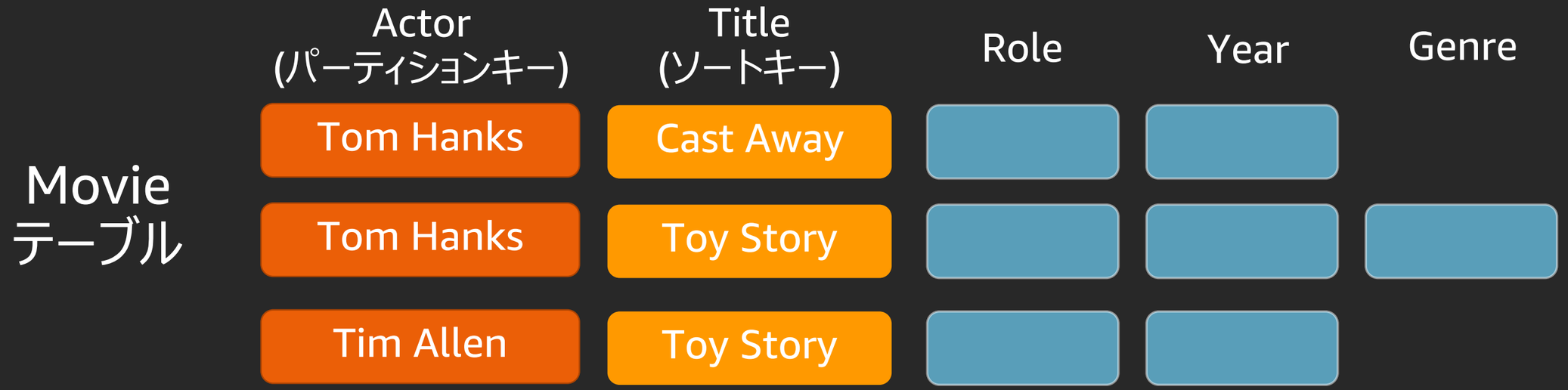
A B C

水平分割（シャーディング）

Amazon DynamoDB

- 完全マネージド型の NoSQL データベースサービス

- 3つのAZを利用した高い可用性と堅牢性
- 必要な分だけのスループットを確保可能
- パッチ適用やアップグレードなどの運用管理の必要なし
- ストレージの容量制限が無い



具体的な改善の例

アプリケーションパフォーマンスの改善のために

- アプリケーションのプロファイリングの実施や、メトリクス・ログを詳細に出力し分析
- アプリケーションロジックやSQLの改善
- 様々なレイヤーでのキャッシュの利用
- 非同期処理化

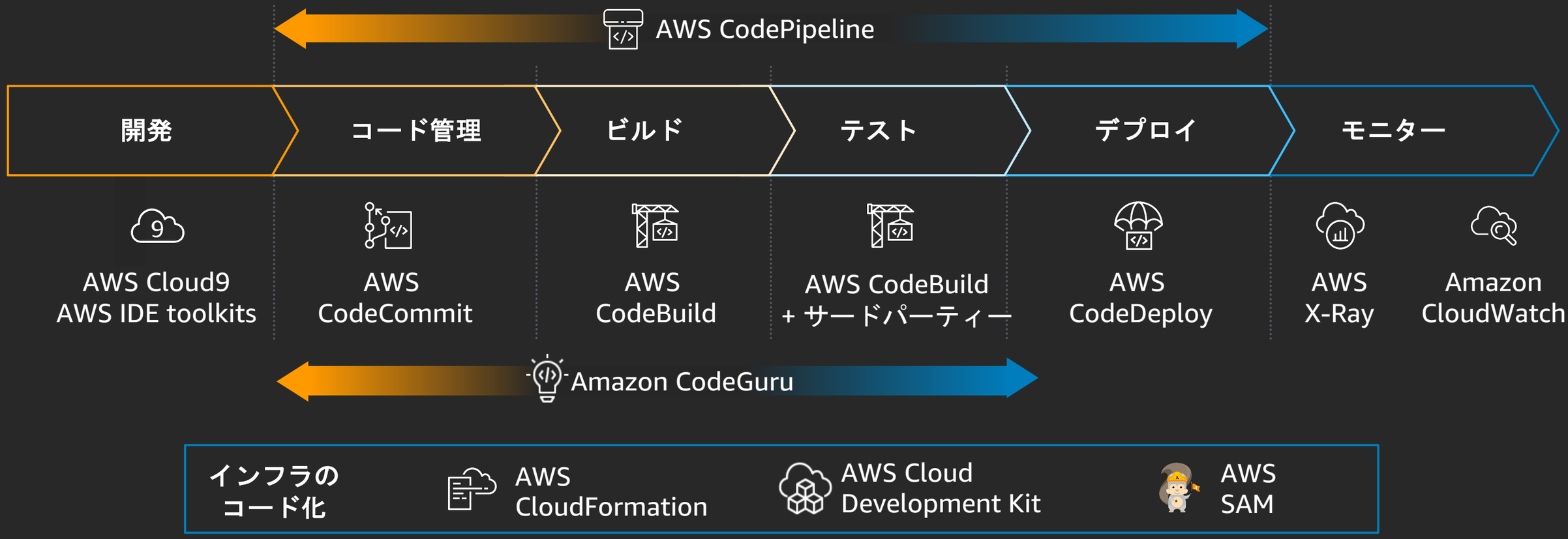
RDSプライマリインスタンスへの負荷集中対策のために

- DBのシャーディング
- NoSQLの利用

運用の改善のために

- 様々なレイヤーでマネージドサービスを活用し、可用性やスケーラビリティを担保
- サービスをモジュール、コンポーネント、マイクロサービスなどの単位で分割
- 運用オペレーションの自動化
- **Infrastructure as CodeやCI/CDの導入**

モダンなソフトウェア開発におけるCI/CD



具体的な改善の例

アプリケーションパフォーマンスの改善のために

- アプリケーションのプロファイリングの実施や、メトリクス・ログを詳細に出力し分析
- アプリケーションロジックやSQLの改善
- 様々なレイヤーでのキャッシュの利用
- 非同期処理化

RDSプライマリインスタンスへの負荷集中対策のために

- DBのシャーディング
- NoSQLの利用

運用の改善のために

- 様々なレイヤーでマネージドサービスを活用し、可用性やスケーラビリティを担保
- サービスをモジュール、コンポーネント、マイクロサービスなどの単位で分割
- 運用オペレーションの自動化
- Infrastructure as CodeやCI/CDの導入

Amazon Builder's Libraryもぜひご覧ください

ソフトウェアの配信と運用

LEVEL 400

デプロイ時におけるロールバックの安全性の確保

Author: Sandeep Pokkunuri

後方互換性、ロールバック、および状態を念頭に置いて稼働中のシステムをアップグレードし、変更によるお客様への影

ソフトウェアの配信と運用

LEVEL 400

ヘルスチェックの実装

Author: David Yanacek

フリート全体でフォールスポジティブの意図しない結果は例外として、サーバー障害を自動的に検出して最小限に抑えます。

ソフトウェアの配信と運用

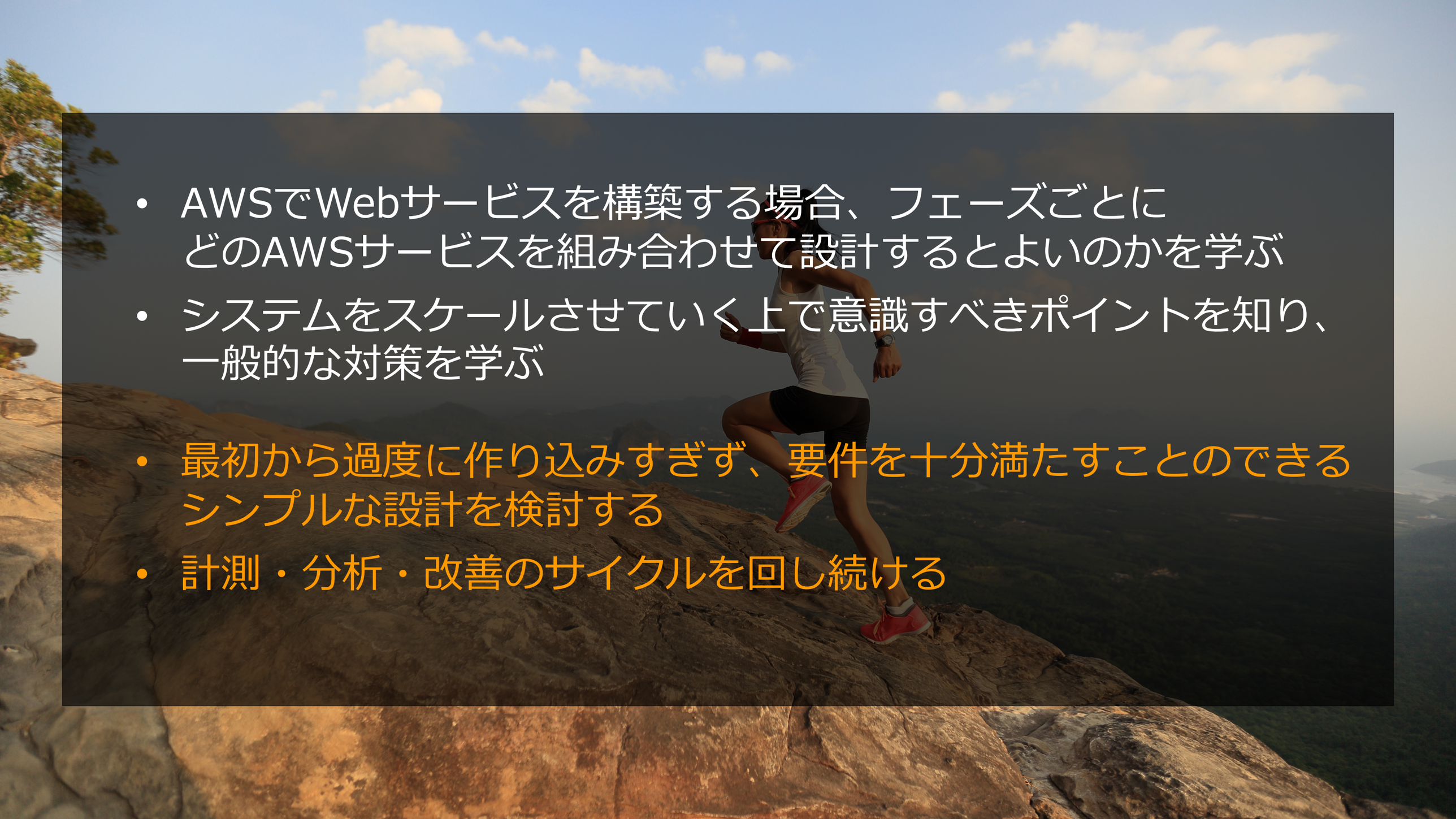
LEVEL 300

継続的デリバリーによる高速化

Author: Mark Mansour

スピードと信頼性のためのソフトウェアテストとデプロイプロセスの自動化。

https://aws.amazon.com/jp/builders-library



- AWSでWebサービスを構築する場合、フェーズごとに
  どのAWSサービスを組み合わせて設計するとよいのかを学ぶ
- システムをスケールさせていく上で意識すべきポイントを知り、
  一般的な対策を学ぶ

- 最初から過度に作り込みすぎず、要件を十分満たすことのできる
  シンプルな設計を検討する
- 計測・分析・改善のサイクルを回し続ける

関連セッションのご案内

AWS-01

今だからこそ知りたい、AWSのデータベース 2020 基礎

AWS-04

30分で理解するAWSストレージサービスの全体像

AWS-15

クラウドへのシステム移行戦略 Lift & Shift とクラウドネイティブ

AWS-21

モダンアプリケーションのためのアーキテクチャデザインパターンと実装

Thank you!

aws SUMMIT ONLINE

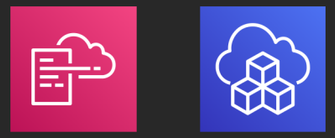
© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Appendix. 開発を支援するサービスの一覧

CI/CD tools

AWS CodeStar
AWS CodeBuild
AWS CodeCommit
AWS CodeDeploy
AWS CodePipeline

Infrastructure as code

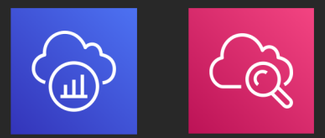


AWS CloudFormation
AWS CDK

IDE

AWS Cloud9

Monitoring & tracing



AWS X-Ray
Amazon CloudWatch

Web apps

AWS Elastic Beanstalk

IDE and DevOps toolkits

Visual Studio Code
IntelliJ
PyCharm
Visual Studio
Eclipse
VSTS

CLI and scripting tools

AWS CLI
Tools for PowerShell

Languages

Amazon Corretto

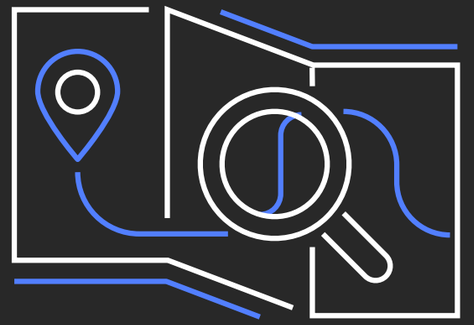
Mobile

AWS Amplify

SDKs

JavaScript
Python
PHP
.NET
Ruby
Java
Go
Node.js
C++

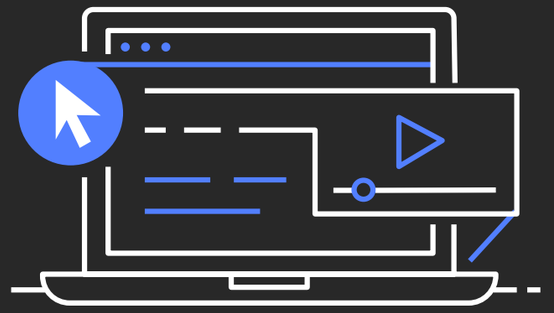
AWS トレーニングと認定



クラウド人材の育成

AWS トレーニングを活用し、ビジネスを牽引する人材の育成と組織作りを促進する

AWS トレーニング活用事例 »



自習コンテンツの活用

ウェビナーやのデジタルトレーニングを受講して、個人のスキルアップを目指す

AWS デジタルトレーニング »

AWS 認定取得を目指す

認定取得を目指して知識を底上げし、AWS の経験とスキルを証明する

AWS 認定の詳細 »

学習パスをお探しの方に

日本語版ランプアップガイドを公開しました。AWS ウェブページ、無料のデジタルトレーニング、クラスルームコース、動画、ホワイトペーパー、認定等を含んだ、9 種の役割別学習ガイドをご覧いただけます。 詳細を見る »

aws.amazon.com/training