レジリエンス概要

レジリエンシーにおけるAWSの強み

リファレンスアーキテクチャ

ミッションクリティカルワークロード実装のベストプラクティス

High Availability

設計および運用メカニズムによる一般的な障害に対する耐性

コア サービス、可用性の目標を達成するための設計

Disaster recovery

まれではあるが影響の大きい障害について、目標内に運用に戻る

バックアップ、データ管理、**RTO/RPO**の管理

Resilience の継続性

CI/CD、**Code** の改善、運用テスト、**Observability** / モニタリング、カオスエンジニアリング

1つのリージョンに複数DC、東京、大阪の2つのリージョンを日本で展開

AWS グローバルインフラストラクチャ

レジリエンシーにフォーカスしたアセスメントとコンサルティングサービス

システムの安定稼働を支えるプレミアムサポート

レジリエンシーを高度化するAWSサービス群

- **高可用性/DR機能組み込み済みのサービス群**
- **システム全体に渡る可観測性の提供**
- **健全性担保のための障害シミュレーション**
- **RPO/RTOをシステム単位で管理可能**

レジリエンス (回復力) とは？

高可用性

設計および運用メカニズムによる一般的な障害への耐性

可用性の目標を満たすためのコアサービスの設計目標

ディザスタリカバリ

大規模な障害が発生した場合に目標の時間内に復旧

バックアップとリカバリ、データの保護、マネージド RPO/RTO

継続的なレジリエンス

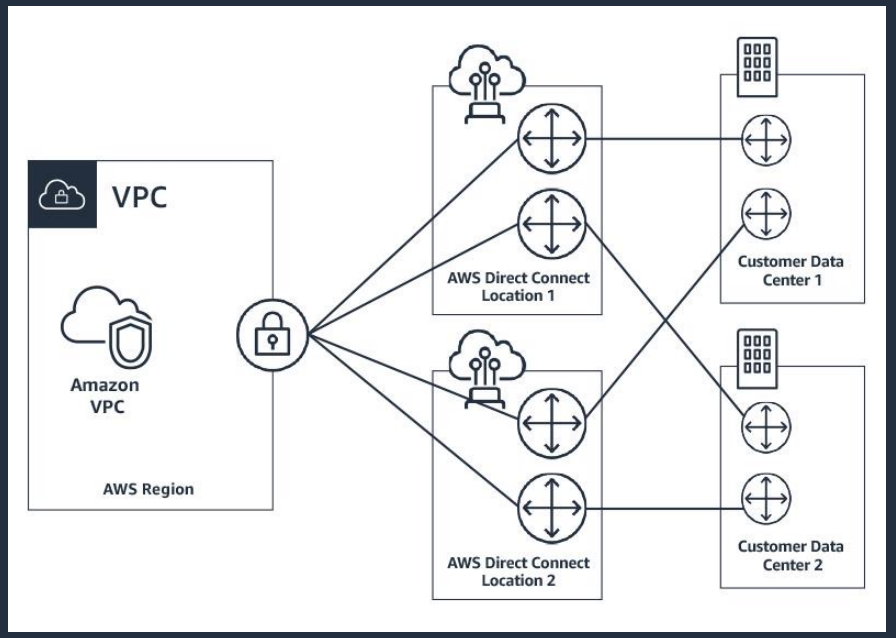
デプロイ前のテストからカオスエンジニアリングパターンへの移行

フォールトインジェクション API、カオスエンジニアリング、Game Day

システム全体に渡るオブザーバビリティ（可観測性）

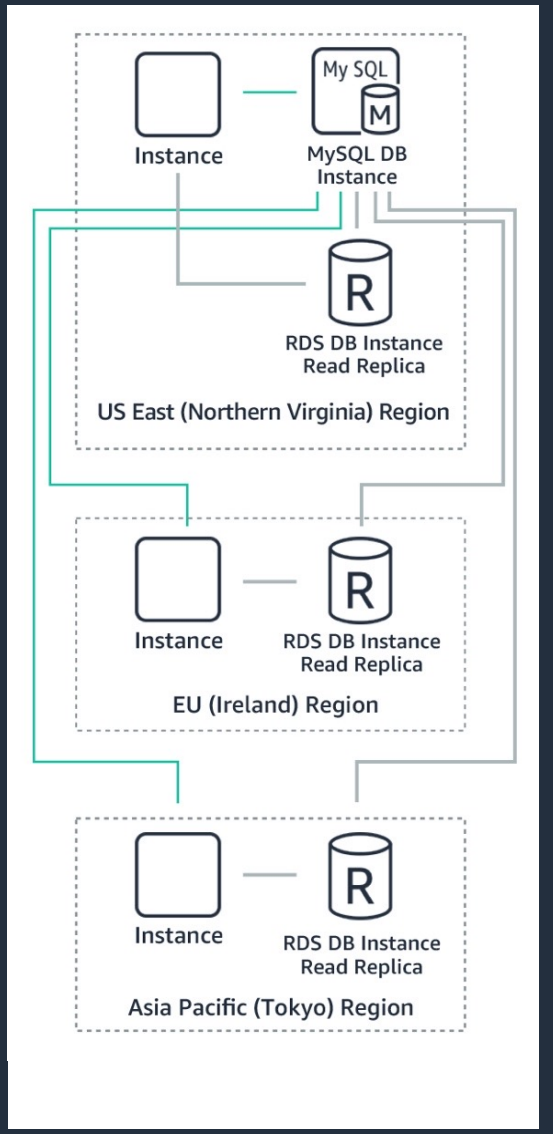
レジリエンス (回復力) のあるネットワーク

- ネットワークは重要基盤
- アプリケーションをサポートするネットワークは、適切な冗長性を備え、常に利用可能で、シームレスにルーティングする必要がある
- AWS 利用するリージョンに対して、エンドユーザー、運用担当を含めた各拠点からの経路冗長性、帯域を確保する
- AWS のサービス
  - Amazon EC2 ネットワーキング
  - Amazon Virtual Private Cloud (VPC)、VPC ピアリング、VPC シェアリング
  - AWS Direct Connect
  - 外部、内部、オンプレミスにルーティングするための AWS のゲートウェイ (VPN、Transit)
  - DNS (Route 53)
  - Elastic Load Balancer (ELB)



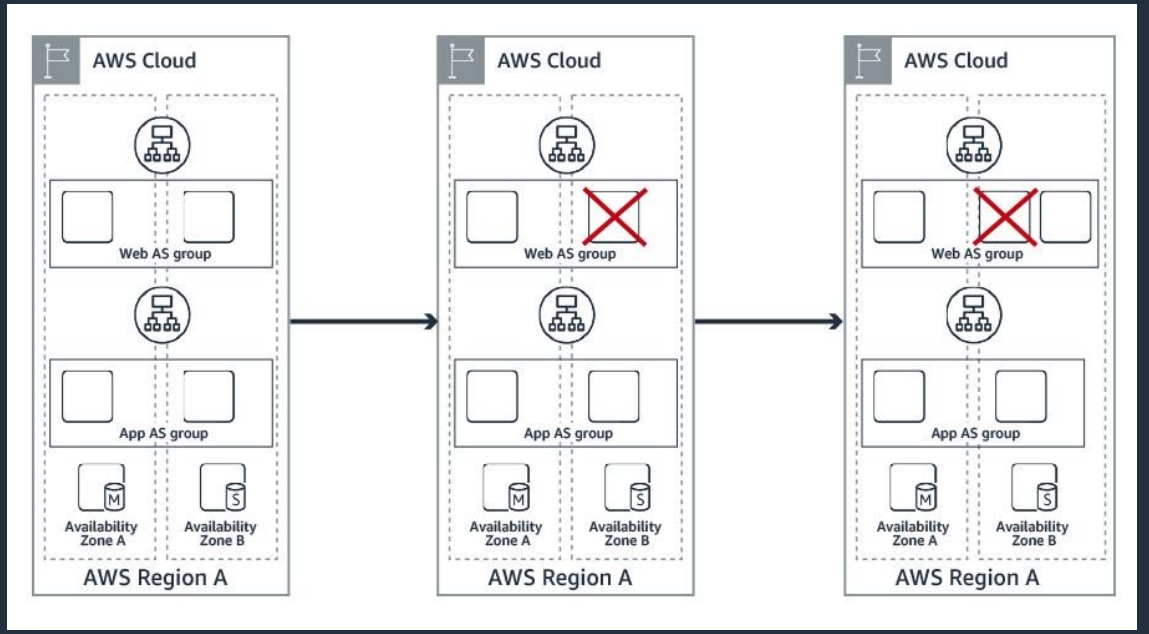
レジリエンス (回復力) のあるデータ

- データのレジリエンスに対して徹底的に検討する。
- さまざまな形式：ファイルシステム、ブロックストレージ、データベース、インメモリキャッシュ
- 想定する問題に対して、目標とするRPOを満たす設計となっているか？
- 結果整合性が設計にどのような影響を与えるか？
- AWS のサービス:
  - Amazon S3 クロスリージョンレプリケーション
  - クロスリージョンスナップショット (Amazon EBS ボリューム)
  - Amazon RDS クロスリージョンレプリカ
  - AWS のストレージゲートウェイとファイルゲートウェイ
  - Amazon FSx (for Windows File Server と for Lustre)



自己修復アプリケーション

- レジリエンスの高いアプリケーションは自己修復できる必要があります
- 手段
  - マイクロサービスのアーキテクチャを活用
  - 相互依存関係を切り離し、疎結合化
  - アプリのコンポーネントからステートを削除
- AWS のサービス:
  - Elastic/Application Load Balancer
  - Amazon CloudWatch
  - AWS X-Ray
  - AWS Lambda



責任共有モデル と インフラストラクチャ

レジリエンスにおける責任共有モデル

お客様

Resilience ‘in’ the cloud

に対する責任を持つ

- 重要インフラストラクチャの継続的なテスト
- ワークロードのアーキテクチャ
- 変更管理と オペレーショナルレジリエンス
- オブザーバビリティと 障害管理
- ネットワーキング、クォータ、制約

AWS

Resilience ‘of’ the cloud

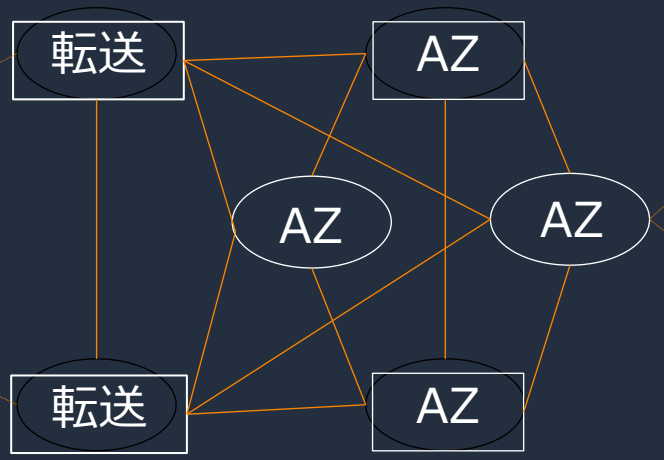
に対する責任を持つ

- ハードウェアとサービス
- コンピュート
- ストレージ
- データベース
- ネットワーク
- AWS グローバルインフラストラクチャ
- リージョン
- アベイラビリティゾーン
- エッジロケーション

AWSはレジリエンスを備えたリージョンをグローバルに展開

マルチ AZ とマルチデータセンターの設計で、一貫して構築されたグローバルフットプリント

AWS リージョン



AWS アベイラビリティーゾーン (AZ)

リージョンは世界各国にある物理的な場所であり、その中に複数のアベイラビリティーゾーンがあります。

アベイラビリティーゾーンは、1 つまたは複数の独立したデータセンターで構成されています。各データセンターには冗長電源、ネットワーク機器、回線を備えており、独立したファシリティに格納されています。

リージョン & アベイラビリティーゾーン数

発表済みのリージョン
バーレーン、香港特別行政区、スウェーデン

AWSの提供サービスと障害分離境界

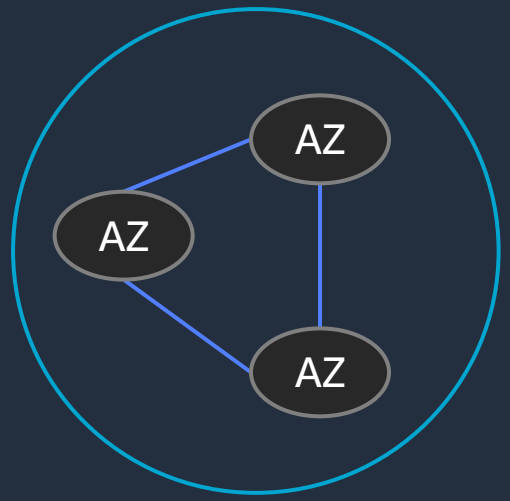
各サービスにおいてコントロールプレーンとデータプレーンを分離

Zonal services

Amazon RDS　Amazon EC2　Amazon EBS

Control plane: Regional
Data plane: Zonal

Regional services



Amazon S3　Amazon SQS　Amazon DynamoDB

Control plane: Regional
Data plane: Regional

Global services

IAM　Amazon CloudFront　Amazon Route 53　AWS Global Accelerator

Control plane: Single Region
Data plane: Global

コントロールプレーンとデータプレーン

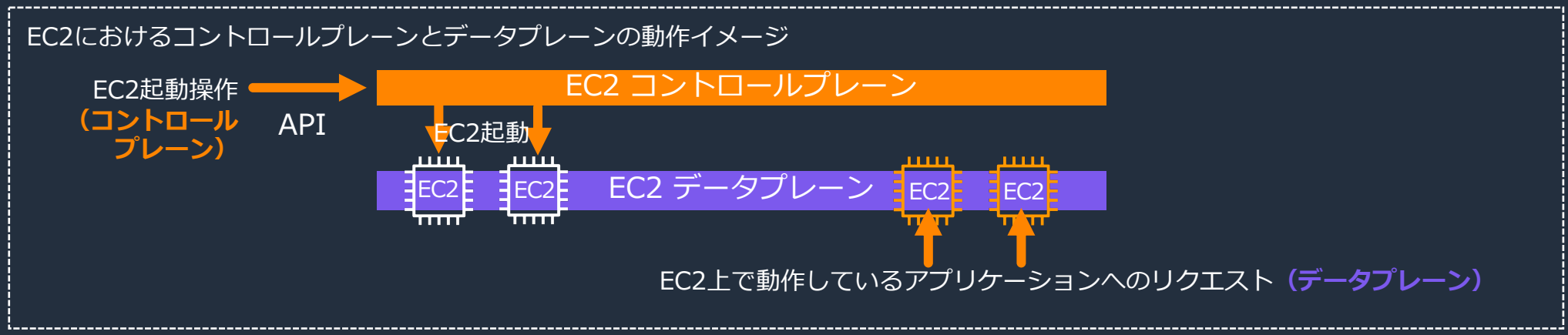
コントロールプレーン

AWSリソースの更新（複雑な処理）
- 例）EC2インスタンスの作成/起動
- 例）Route53のリソース作成/更新/削除

互いに分離

データプレーン

AWSリソースのコア機能の提供（シンプルな処理）
- 例）EC2インスタンスでプログラムが動作
- 例）Route53のDNSクエリ応答



コントロールプレーンの障害時でも、データプレーンは安定して動作するよう設計

障害回復においてコントロールプレーンに依存しない設計が重要

Amazon Builders' Library アベイラビリティーゾーンを使用した静的安定性
https://aws.amazon.com/jp/builders-library/static-stability-using-availability-zones/

オブザーバビリティ（可観測性）とは

- システムの外部出力をもとにシステムの内部状態を判断できる状態
- モニタリングではシステムが機能しているかどうかを把握できる一方、オブザーバビリティによりなぜ機能していないかを理解することができる

視認性

迅速なトラブルシューティング

顧客体験

オブザーバビリティの 3つの柱

メトリクス

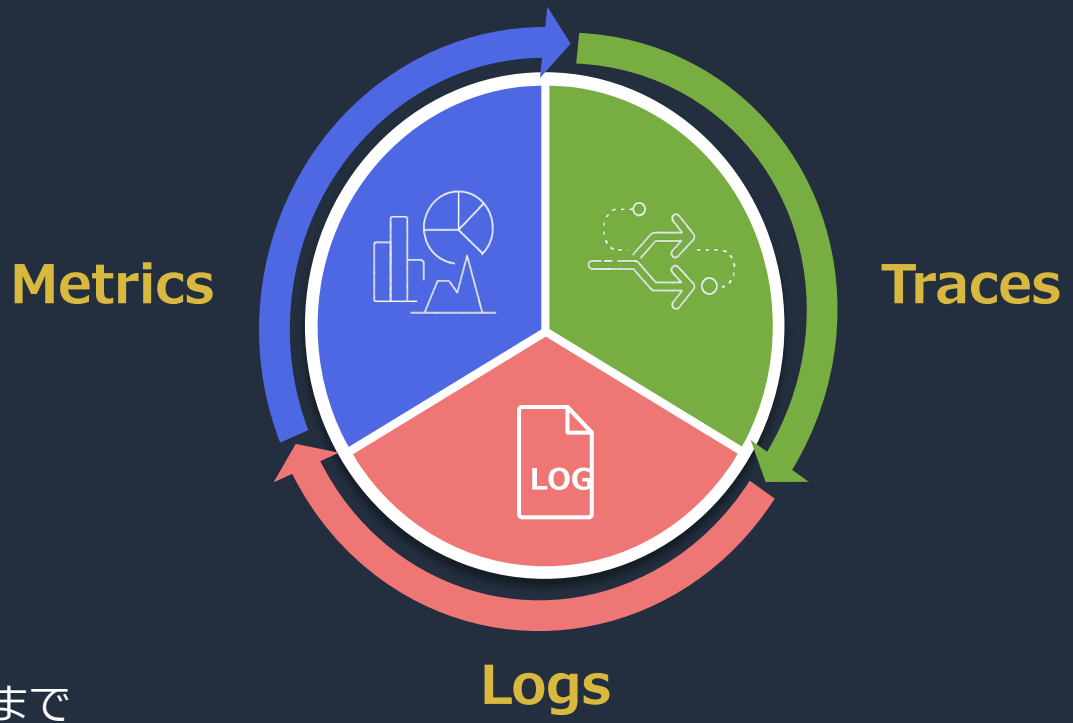
- ある時点のシステム状態を表現する 数値情報
- 例) CPU 使用率、リクエストレート、など

ログ

- システム内で発生した イベント情報
- 例) アクセスログ、 エラー情報、など

トレース

- 1 つのトランザクションにおける フロー情報
- 例) HTTP リクエストの受け取りからレスポンスまで



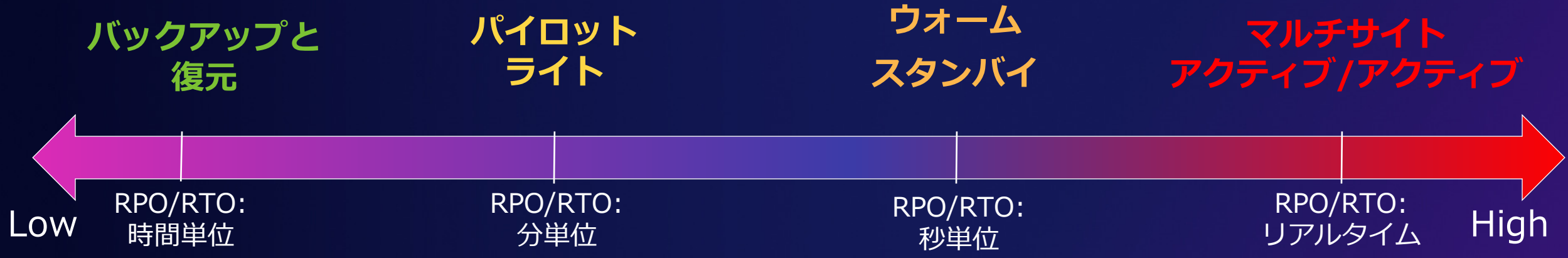
アーキテクチャ

ディザスタリカバリ (DR) 目標

災害イベントによってワークロードがダウンする可能性があるため、
DR の目的は、ワークロードを元の状態に戻すか、ダウンタイムを完全に回避することです。次の目標値を使います。

- RTO (目標復旧時間) :サービスの中断から
  サービスの復旧までの最大許容遅延時間
- RPO (目標復旧時点) :最後のデータ復旧ポイントからの最大許容時間

AWS におけるディザスタリカバリ (DR) 戦略



- 平常時は
  バックアップを取得
- RTO に応じて常時
  サブサイトへバック
  アップを転送する or
  障害時に転送する
- 障害時に必要な
  データをサブサイトで
  リストア
- 費用：$

- 平常時は
  コアサービスのみに限定
  したサブサイトを運用
  (例 DB のみ常時運用
  etc.)
- 障害時に、必要に応じて
  その他サービスを起動
- 費用：$$

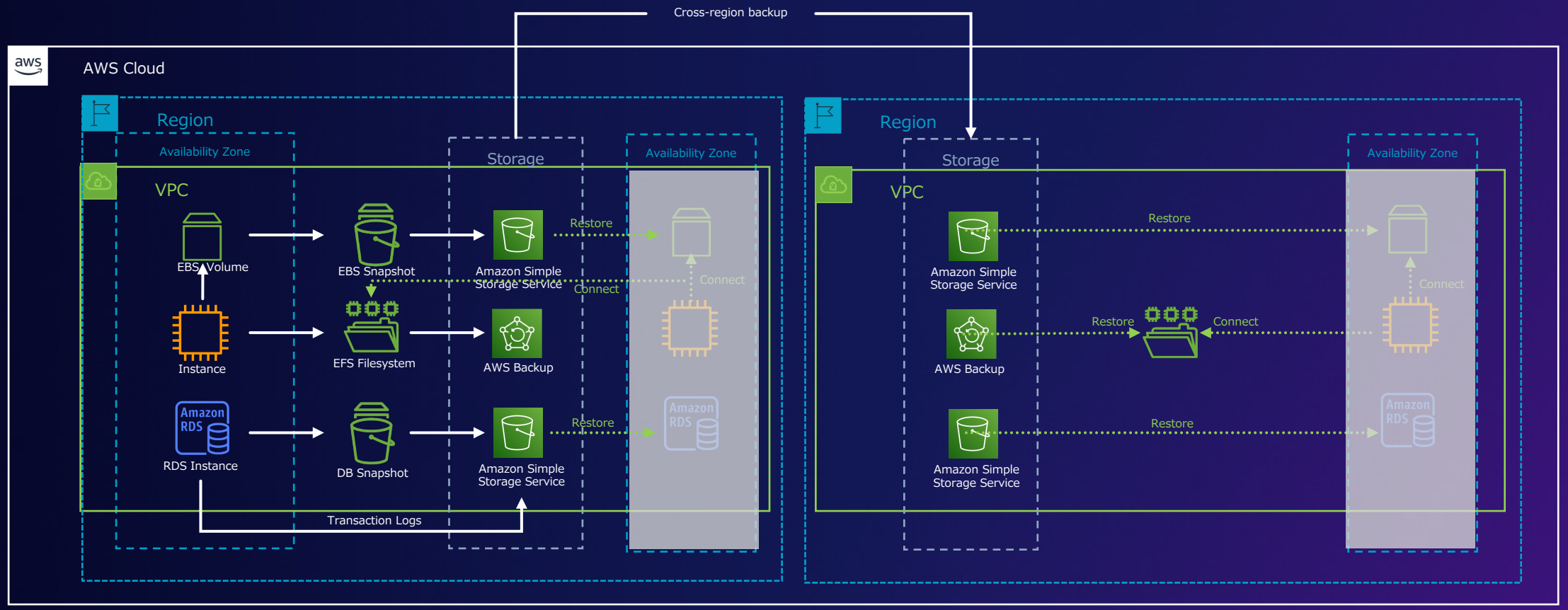
- 平常時からメイン
  サイト同等機能を
  もち規模を縮小した
  サブサイトを運用
- 障害時は自動フェイル
  オーバーし、流量を
  絞って業務を継続
- 費用：$$$

- 平常時からメイン
  サイト同等構成の
  サブサイトを運用
- 障害時は
  自動フェイルオーバー
- 費用: $$$$

参考①：https://docs.aws.amazon.com/ja_jp/wellarchitected/latest/reliability-pillar/plan-for-disaster-recovery-dr.html
参考②：https://docs.aws.amazon.com/ja_jp/whitepapers/latest/disaster-recovery-workloads-on-aws/disaster-recovery-options-in-the-cloud.html

バックアップと復元

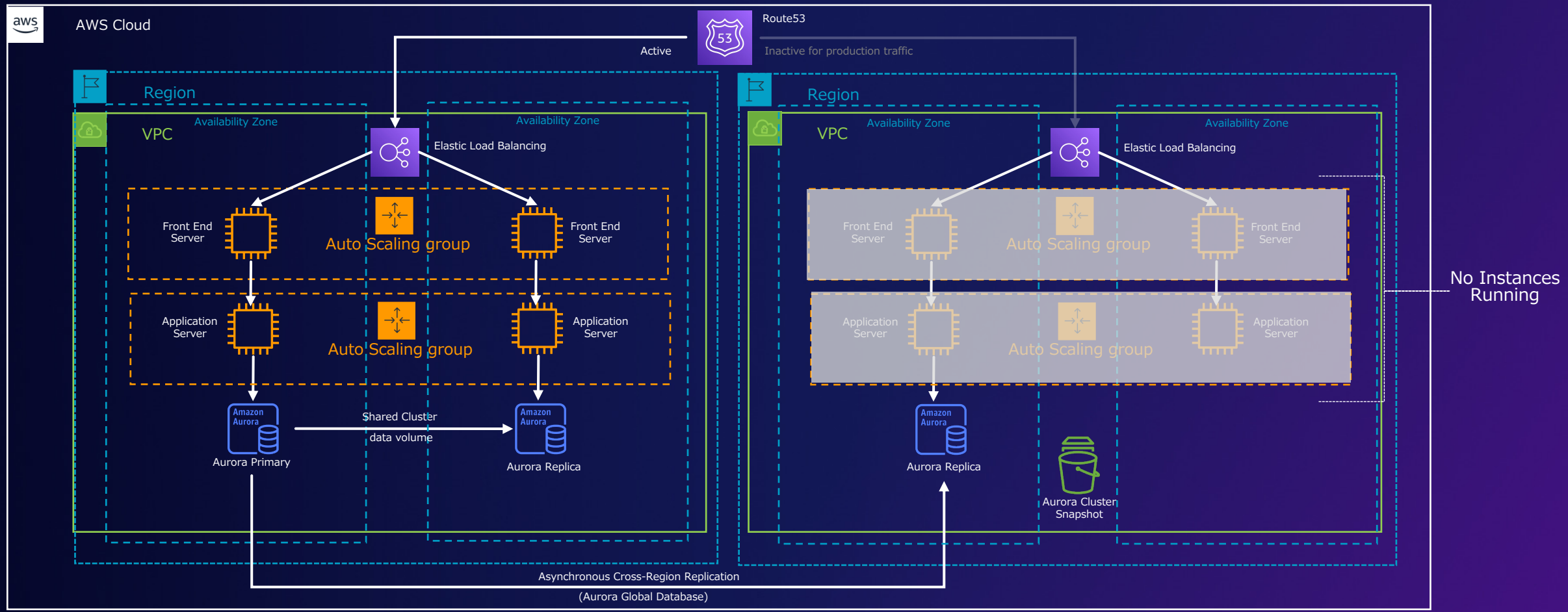
- 平常時はバックアップを取得
- RTO に応じて常時サブサイトへバックアップを転送 or 障害時に転送する
- 障害時に必要なデータをサブサイトでリストア



図：バックアップと復元戦略の例

パイロットライト

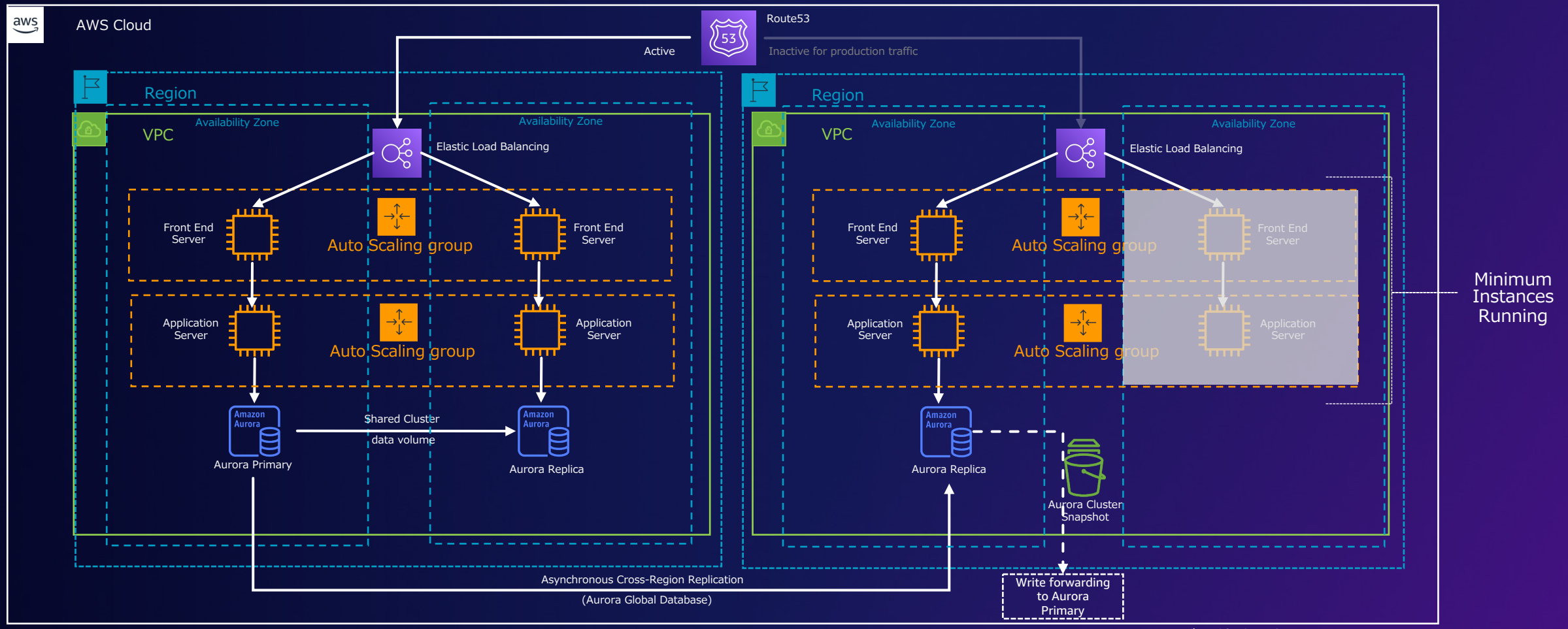
- 平常時はコアサービスのみに限定したサブサイトを運用 (例 DB のみ常時運用 etc.)
- 障害時に、必要に応じてその他サービスを起動



図：パイロットライト戦略の例

ウォームスタンバイ

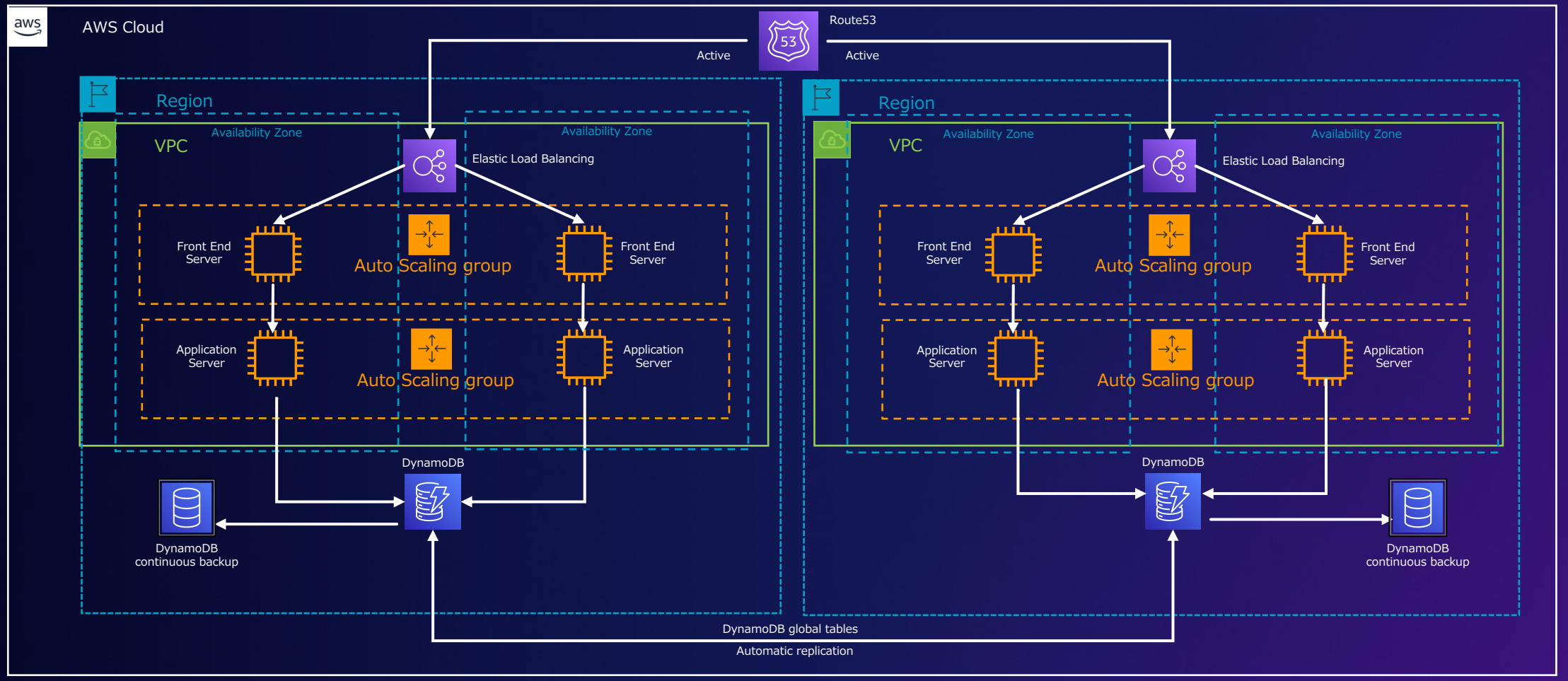
- 平常時からメイン サイト同等機能を もち規模を縮小した サブサイトを運用
- 障害時は自動フェイルオーバーし、流量を絞って業務を継続



図：ウォームスタンバイ戦略の例

マルチサイトアクティブ/アクティブ

- 平常時からメイン サイト同等構成の サブサイトを運用
- 障害時は 自動フェイルオーバー



図：マルチサイトアクティブ/アクティブ戦略の例

W-A

信頼性の柱における設計原則

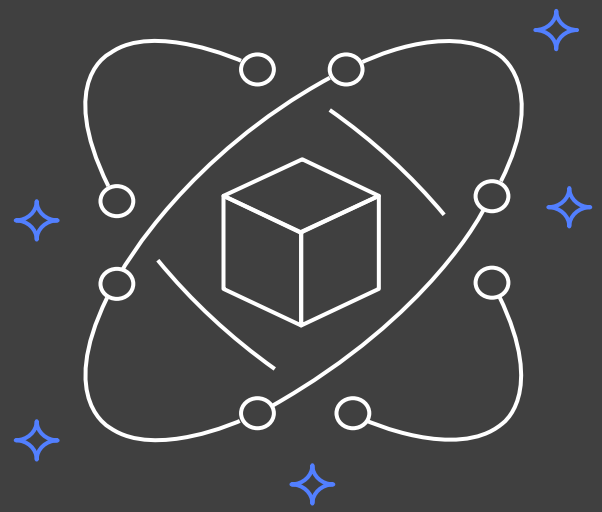
障害から自動的に復旧する

復旧手順をテストする

水平方向にスケールしてワークロード全体の可用性を高める

キャパシティを推測することをやめる

オートメーションで変更を管理する



AWS Well-Architected フレームワークの 6 本の柱

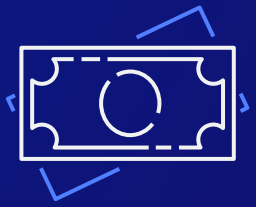
AWS Well-Architected フレームワーク全体の設計原則

運用上の優秀性

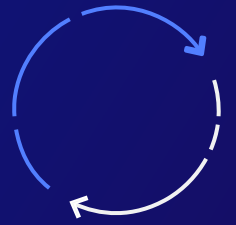
セキュリティ

信頼性

パフォーマンス効率

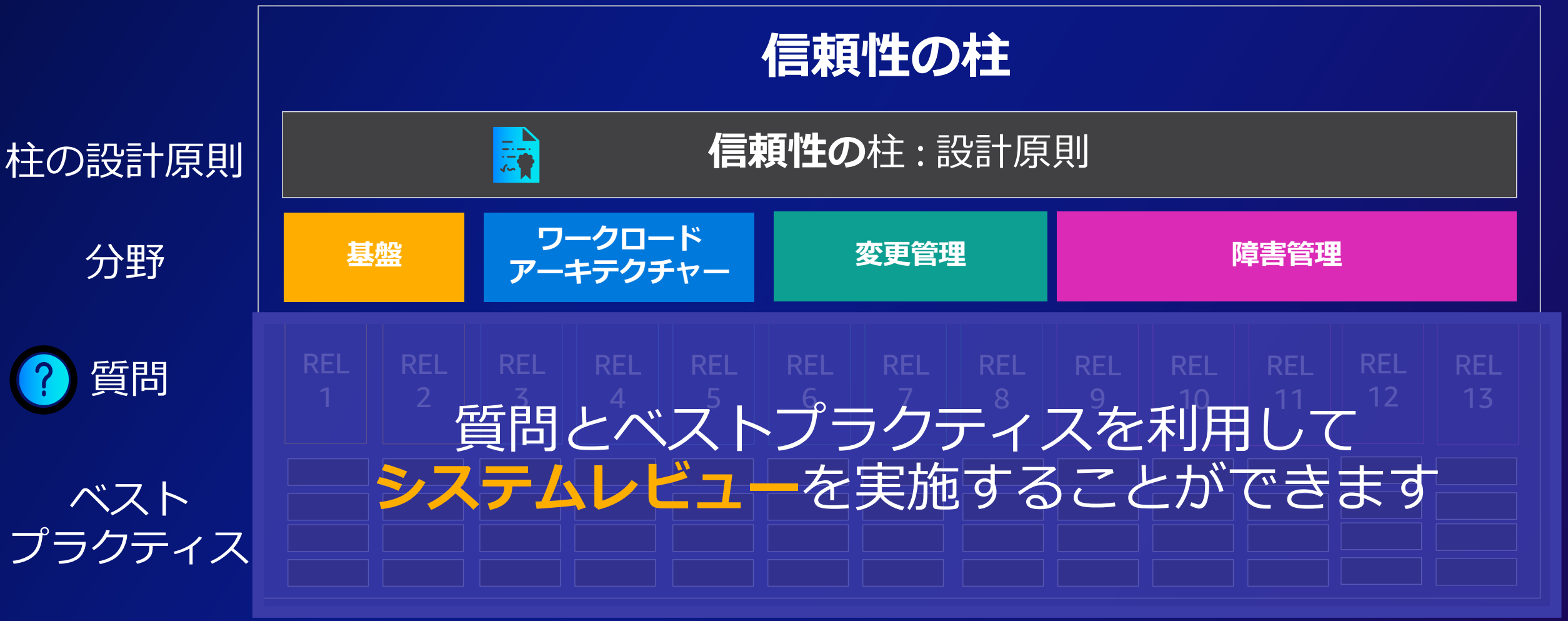


コスト最適化



持続可能性

AWS Well-Architected Framework の構造



信頼性における質問

分野	質問のトピック
基盤	REL 1 サービスクォータと制約はどのように管理しますか?
	REL 2 ネットワークトポロジをどのように計画しますか?
ワークロードアーキテクチャー	REL 3 ワークロードサービスアーキテクチャをどのように設計すればよいですか?
	REL 4 障害を防ぐために、分散システムでの操作をどのように設計すればよいですか?
	REL 5 障害を緩和または耐えるために、分散システムの操作をどのように設計しますか?
変更管理	REL 6 ワークロードリソースをモニタリングするにはどうすればよいですか?
	REL 7 需要の変化に適応するようにワークロードを設計するには、どうすればよいですか?
	REL 8 変更はどのように実装するのですか?
障害管理	REL 9 データはどのようにバックアップするのですか?
	REL 10 ワークロードを保護するために、障害分離をどのように使用すればよいですか?
	REL 11 コンポーネントの障害に耐えるようにワークロードを設計するにはどうすればよいですか?
	REL 12 信頼性はどのようにテストすればよいですか?
	REL 13 災害対策 (DR) はどのように計画するのですか?