AWS-50

マルチリージョンでディザスタリカバリ(DR)戦略を検討するためのポイント

大場 崇令
パートナーアライアンス統括本部 ストラテジック SI 技術部
シニアパートナーソリューションアーキテクト
アマゾン ウェブ サービス ジャパン合同会社

Who am I ?

@takaohba

- 大場 崇令（オオバ タカノリ）
  - Senior Partner Solutions Architect @Amazon Web Services Japan G.K. (Joined 2015/12)
- Background
  - AWS テクニカルトレーナー@AWSJ G.K.
  - Web サービスのインフラエンジニア
  - 国内クラウドベンダーにてテクニカルサポート
- 好きな AWS サービス
  - AWS Well-Architected Tool
  - AWS Systems Manager
  - AWS Service Catalog

Well-Architected Lead

本セッションの対象者とゴール

- 対象者：AWS で DR 戦略を検討されている全てのお客様、
  すでに AWS で複数のワークロードを運用されている方

- ゴール：AWS 上でワークロードの特性に応じた、
  ディザスタリカバリ (DR) 戦略を検討できるようになること

本日お伝えしたい内容

- 災害イベントとは
  - ワークロードまたはシステムが主要な場所でビジネス目標を達成するのを妨げるイベント
- RTO/RPO や可用性要件に基づいて、DR 戦略を検討する
  - RTO/RPO の確認
  - まずはマルチ AZ 戦略で可用性要件を満たせているか検討
  - マルチリージョン戦略は可用性要件だけでなく、ビジネスの継続性の観点も考慮して設計を検討
- AWS で検討可能な DR 戦略

バックアップと復元

- 開始が簡単
- 極めてコスト効果が高い (主にバックアップストレージ)

パイロットライト

- コアサービスのみに限定したサブサイトを運用 (DB のみ等)
- コスト効果が高い (常時稼働リソースが少ない)

ウォームスタンバイ

- 本番トラフィックをいつでも、ある程度まで処理できる
- フル DR よりも小さい IT フットプリント

マルチサイトアクティブ/アクティブ

- いつでも本番負荷を処理できる
- 障害時は自動フェイルオーバー

本日お伝えしたい内容

- 災害イベントとは
  - ワークロードまたはシステムが主要な場所でビジネス目標を達成するのを妨げるイベント
- RTO/RPO や可用性要件に基づいて、DR 戦略を検討する
  - RTO/RPO の確認
  - まずはマルチ AZ 戦略で可用性要件を満たせているか検討
  - マルチリージョン戦略は可用性要件だけでなく、ビジネスの継続性の観点も考慮して設計を検討
- AWS で検討可能な DR 戦略

バックアップと復元
- 開始が簡単
- 極めてコスト効果が高い (主にバックアップストレージ)

パイロットライト
- コアサービスのみに限定したサブサイトを運用 (DB のみ等)
- コスト効果が高い (常時稼働リソースが少ない)

ウォームスタンバイ
- 本番トラフィックをいつでも、ある程度まで処理できる
- フル DR よりも小さい IT フットプリント

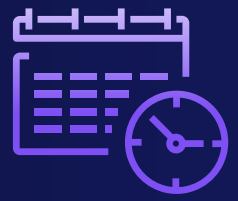
マルチサイトアクティブ/アクティブ
- いつでも本番負荷を処理できる
- 障害時は自動フェイルオーバー

企業が必要とする IT インフラ

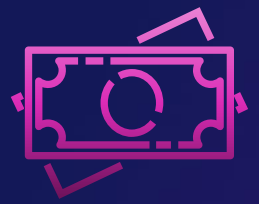
企業はこれまで以上に
強力な基盤アーキテクチャを必要としている

ハイパフォーマンス

セキュア



信頼性



コスト効率

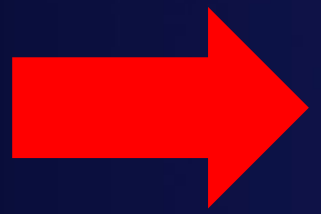
持続可能

災害イベントとは

- 自然災害：地震、洪水など
- 技術的障害：電力やネットワークの損失など
- 人的ミス：不注意、不正な変更など

災害イベントとは

- 自然災害：地震、洪水など
- 技術的障害：電力やネットワークの損失など
- 人的ミス：不注意、不正な変更など



最終的に、ワークロードまたはシステムが主要な場所でビジネス目標を達成するのを妨げるイベントは、**災害**として分類される

https://docs.aws.amazon.com/ja_jp/whitepapers/latest/disaster-recovery-workloads-on-aws/what-is-a-disaster.html

AWS のリージョン

1. 米国東部 (バージニア北部)
2. 米国西部 (北カリフォルニア)
3. 米国西部 (オレゴン)
4. 欧州 (アイルランド)
5. アジアパシフィック(東京)
6. 南米 (サンパウロ)
7. アジアパシフィック (シンガポール)
8. アジアパシフィック (シドニー)
9. GovCloud (米国西部) *1
10. 中国 (北京)*2
11. 欧州 (フランクフルト)
12. アジアパシフィック (ソウル)
13. アジアパシフィック(ムンバイ)
14. 米国東部(オハイオ)
15. カナダ(中部)
16. 欧州(ロンドン)
17. 中国(寧夏) *2
18. 欧州(パリ)
19. アジアパシフィック（大阪）
20. GovCloud (米国東部) *1
21. 欧州(ストックホルム)
22. 香港特別自治区
23. 中東(バーレーン)
24. アフリカ(ケープタウン)
25. 欧州(ミラノ)
26. アジアパシフィック(ジャカルタ)

*1 GovCloudは米国政府関係企業用です
*2 中国のリージョンはAWS アカウントとは別のアカウント作成が必要です

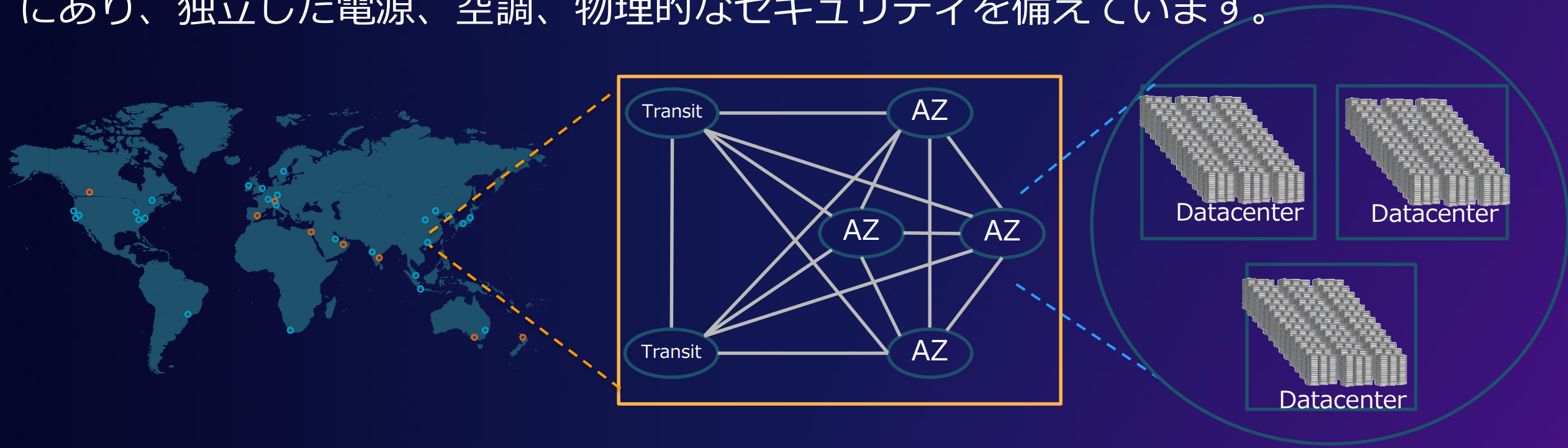
※2022/03 時点

AWS のリージョン

- AWS のリージョンは、他リージョンと完全に分離されるように設計されており、リソースは指定したリージョンに結びつけられている
- ユーザのリソース及びデータは、リージョン内にとどまる
- 複数リージョンを使う場合は、明示的にその機能を選択するか、ユーザ自身で設計・構築する必要がある

AWS のリージョンにおける アベイラビリティーゾーン (AZ)

それぞれのリージョンは、複数アベイラビリティーゾーン (AZ) で構成されています。AZ は自然災害やデータセンター単位の障害などビジネスに影響を与えるリスクを最小化するよう地理的に影響を受けない十分離れた場所 (100 km (60 マイル) 以内) にあり、独立した電源、空調、物理的なセキュリティを備えています。



本日お伝えしたい内容

- 災害イベントとは
  - ワークロードまたはシステムが主要な場所でビジネス目標を達成するのを妨げるイベント
- RTO/RPO や可用性要件に基づいて、DR 戦略を検討する
  - RTO/RPO の確認
  - まずはマルチ AZ 戦略で可用性要件を満たせているか検討
  - マルチリージョン戦略は可用性要件だけでなく、ビジネスの継続性の観点も考慮して設計を検討
- AWS で検討可能な DR 戦略
  - バックアップと復元
    - 開始が簡単
    - 極めてコスト効果が高い (主にバックアップストレージ)
  - パイロットライト
    - コアサービスのみに限定したサブサイトを運用 (DB のみ等)
    - コスト効果が高い (常時稼働リソースが少ない)
  - ウォームスタンバイ
    - 本番トラフィックをいつでも、ある程度まで処理できる
    - フル DR よりも小さい IT フットプリント
  - マルチサイトアクティブ/アクティブ
    - いつでも本番負荷を処理できる
    - 障害時は自動フェイルオーバー

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

ディザスタリカバリ (DR) 目標を定める重要性：RTO, RPO 要件の明確化

RTO と RPO は、数値が小さいほど、ダウンタイムとデータ損失が少なくなります。
ただし、RTO とRPO が低いほど、リソースへの支出と運用の複雑さという点でコストが高くなります。
したがって、ワークロードに適切な価値を提供する RTO および RPO 目標を検討する必要があります。

Recovery Point Objective (RPO)

- ディザスタリカバリの復旧時点の目標値
- バックアップ・データを
  取得するタイミングや頻度のこと
- どの程度のデータ損失が許容されるか？
- バックアップ・リストアの運用間隔や
  データレプリケーションの技術選択に影響

Recovery Time Objective (RTO)

- 該当サービス復旧にかけられる時間
- 1分、15分、1時間、4時間、1日？
- データリストアや、システム再起動等の
  技術選択に影響

災害イベントの影響範囲に応じたディザスタリカバリ (DR) 戦略の考え方

- マルチ AZ 戦略：
  停電、洪水、その他の地域的な災害などのイベントに耐える DR 戦略を設計している場合は、AWS リージョン内でマルチ AZ DR 戦略を使用することで必要な保護を検討できる

- マルチリージョン戦略：
  別々の異なる場所にある複数のデータセンターに影響を与える可能性のある十分な範囲のイベントに対するビジネス保証が提供される

可用性目標の実装例

$$\text{可用性} = \frac{\text{使用可能な時間}}{\text{全体の時間}}$$

ロードバランサー (Elastic Load Balancing(ELB))、アプリケーションサーバー (Amazon Elastic Compute Cloud(EC2))、Amazon Simple Storage Service(S3) 上の静的コンテンツ、RDBMS (Amazon Relational Database Service (RDS)) で構成される典型的なウェブアプリケーションを例に

単一リージョン

- 可用性 99% のシナリオ
  - 1 つのリージョンと 1 つのアベイラビリティーゾーン
- 可用性 99.9% のシナリオ
  - アベイラビリティーゾーンを 2 つ使用してデプロイし、アプリケーションを個別の階層に分離
- 可用性 99.99% のシナリオ
  - アベイラビリティーゾーンを 3 つ使用してデプロイ

マルチリージョン

- 可用性 99.95% で復旧時間が 5 ～ 30 分のシナリオ
- 可用性 99.999% 以上で復旧時間が 1 分未満のシナリオ

可用性目標の実装例：
https://docs.aws.amazon.com/ja_jp/wellarchitected/latest/reliability-pillar/example-implementations-for-availability-goals.html

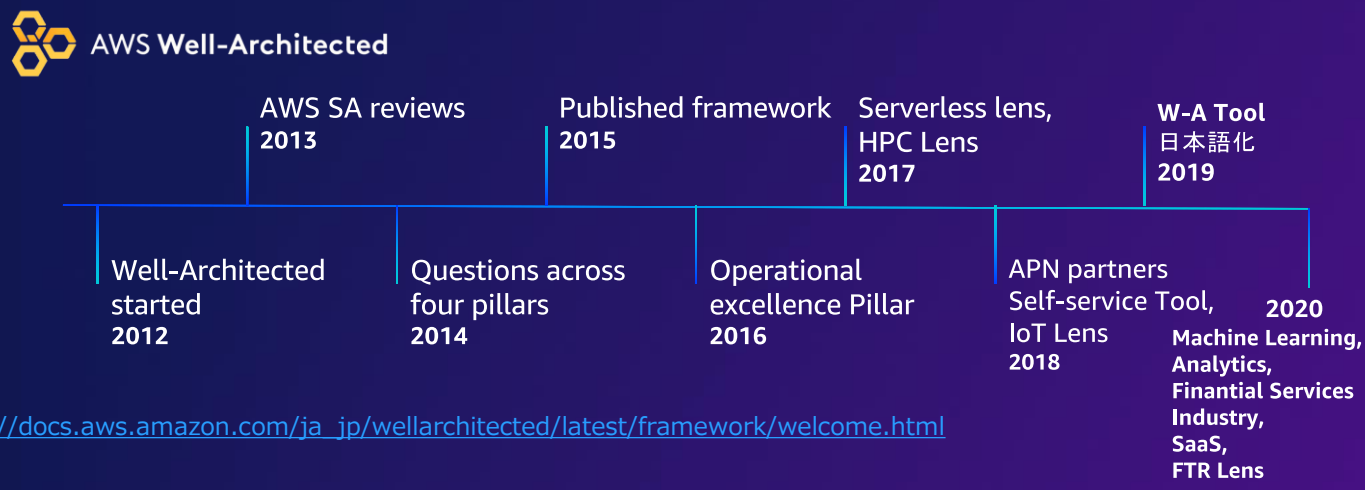
AWS Well-Architected フレームワーク

AWS Well-Architected フレームワーク (W-A) とは?

システム設計・運用の”大局的な”考え方と
ベストプラクティス集

・AWS のソリューションアーキテクト (SA)、パートナー様、お客様の 10 年以上にわたる**経験**から作り上げたもの

・AWS と**お客様**と共に、W-A も**常に進化し続ける**



https://docs.aws.amazon.com/ja_jp/wellarchitected/latest/framework/welcome.html

AWS Well-Architected フレームワークとは？

柱

設計原則
(Design principles)

質問

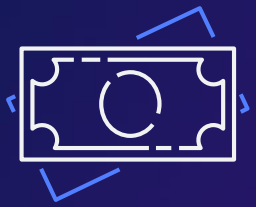
AWS Well-Architected フレームワークの 6 本の柱

運用上の優秀性

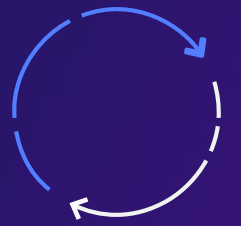
セキュリティ

信頼性

パフォーマンス効率



コスト最適化



持続可能性

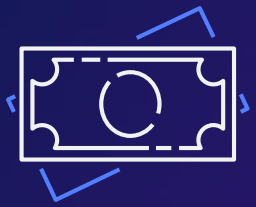
AWS Well-Architected フレームワークの 6 本の柱

運用上の優秀性

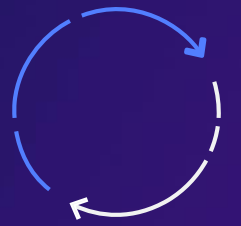
セキュリティ

信頼性

パフォーマンス効率



コスト最適化



持続可能性

信頼性の柱における設計原則

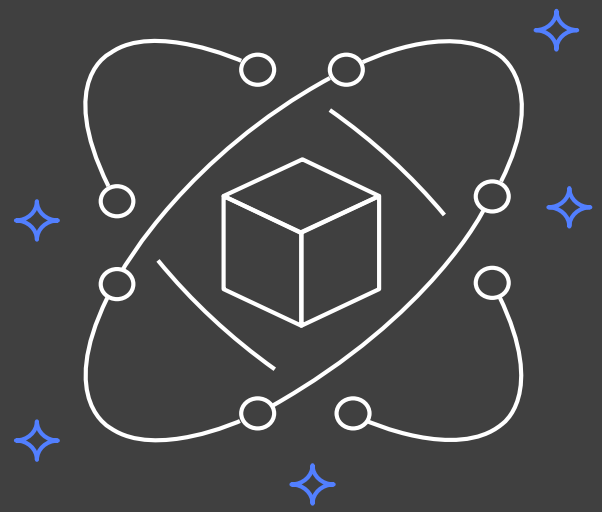
障害から自動的に復旧する

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



本日お伝えしたい内容

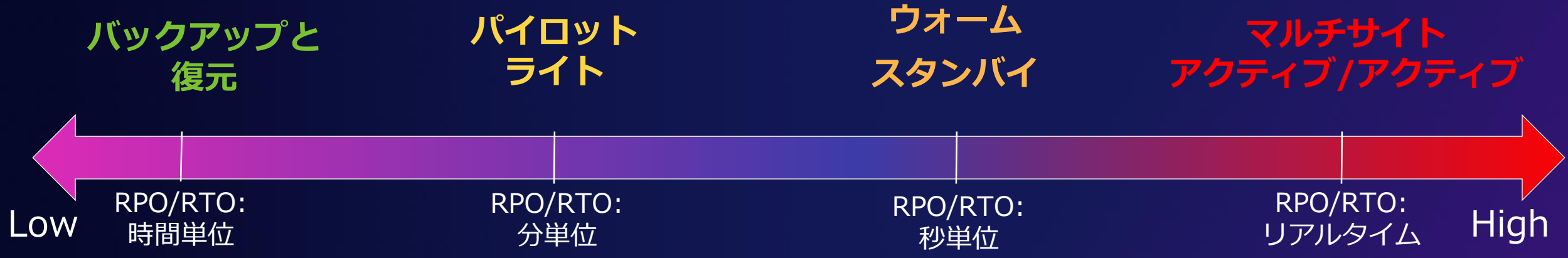
- 災害イベントとは
  - ワークロードまたはシステムが主要な場所でビジネス目標を達成するのを妨げるイベント
- RTO/RPO や可用性要件に基づいて、DR 戦略を検討する
  - RTO/RPO の確認
  - まずはマルチ AZ 戦略で可用性要件を満たせているか検討
  - マルチリージョン戦略は可用性要件だけでなく、ビジネスの継続性の観点も考慮して設計を検討

- AWS で検討可能な DR 戦略

  - バックアップと復元
    - 開始が簡単
    - 極めてコスト効果が高い (主にバックアップストレージ)
  - パイロットライト
    - コアサービスのみに限定したサブサイトを運用 (DB のみ等)
    - コスト効果が高い (常時稼働リソースが少ない)
  - ウォームスタンバイ
    - 本番トラフィックをいつでも、ある程度まで処理できる
    - フル DR よりも小さい IT フットプリント
  - マルチサイトアクティブ/アクティブ
    - いつでも本番負荷を処理できる
    - 障害時は自動フェイルオーバー

AWS におけるディザスタリカバリ (DR) 戦略

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：$$

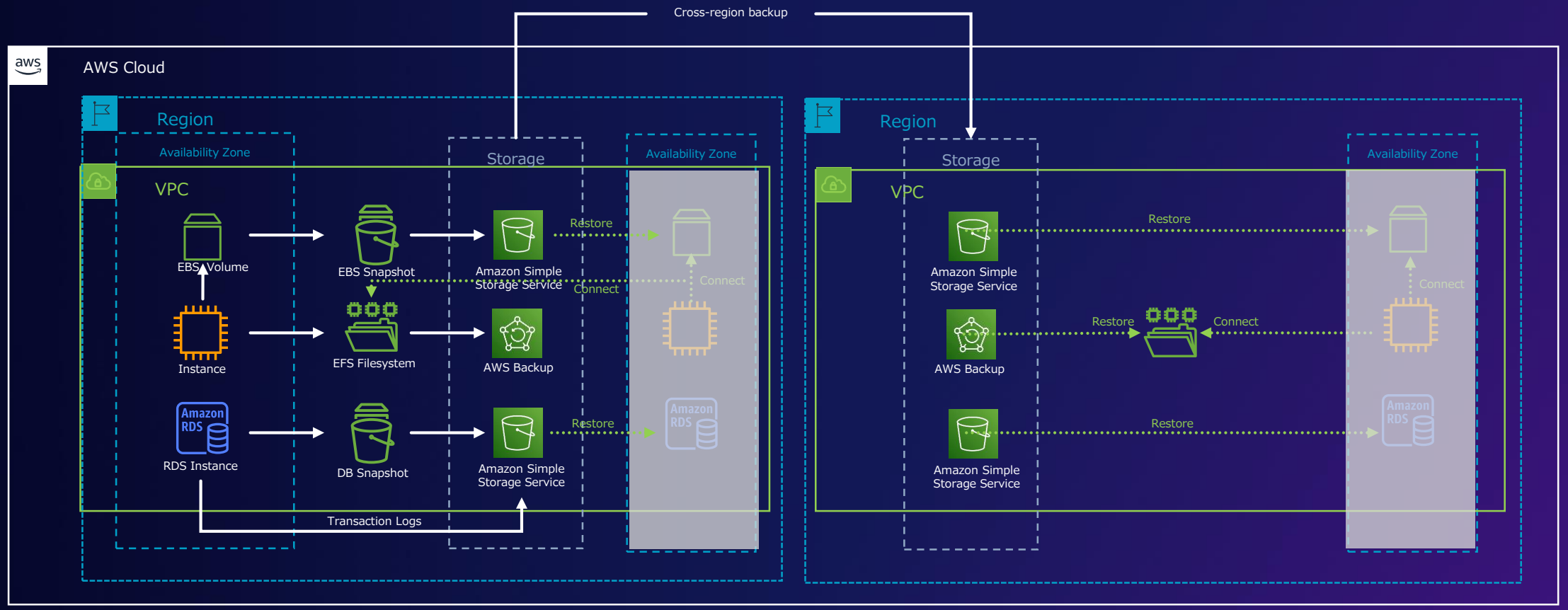
- 平常時からメイン
  サイト同等機能を
  もち規模を縮小した
  サブサイトを運用
- 障害時は自動フェイル
  オーバーし、流量を
  絞って業務を継続
- 費用：$$$

- 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

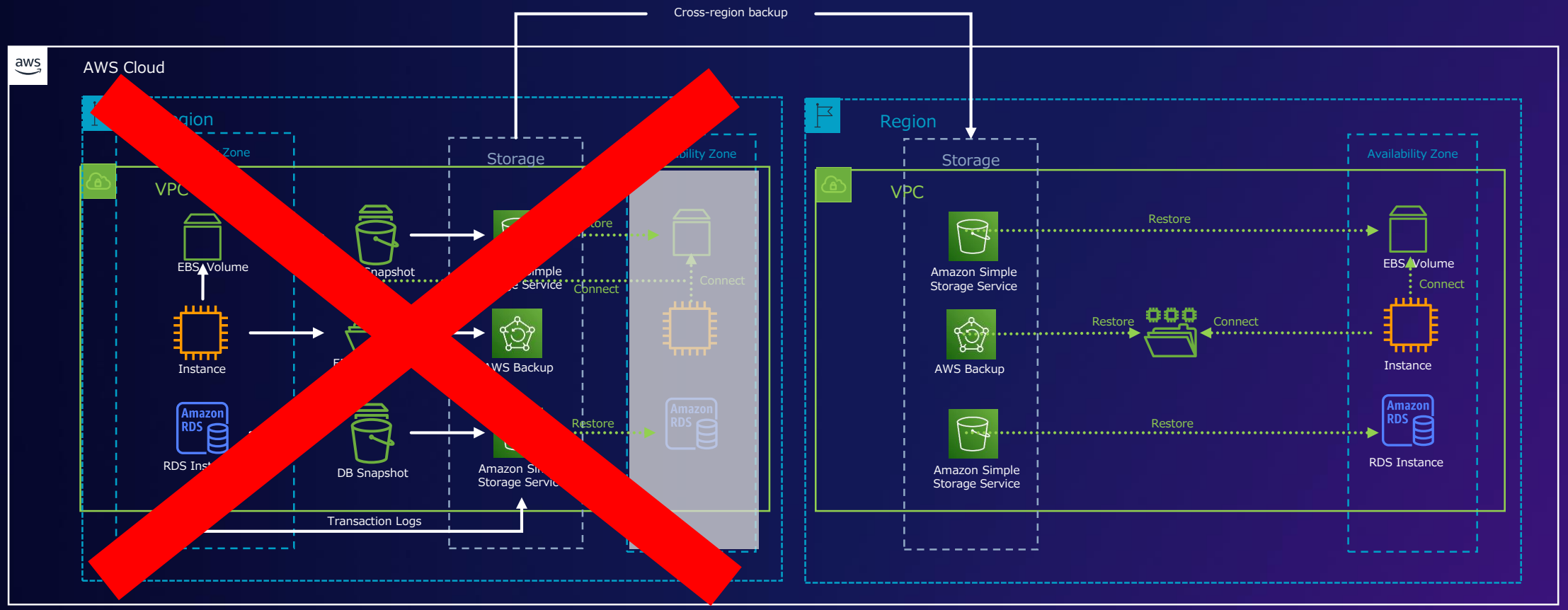
- 平常時はバックアップを取得
- RTO に応じて常時サブサイトへバックアップを転送 or 障害時に転送する
- 障害時に必要なデータをサブサイトでリストア



図：バックアップと復元戦略の例

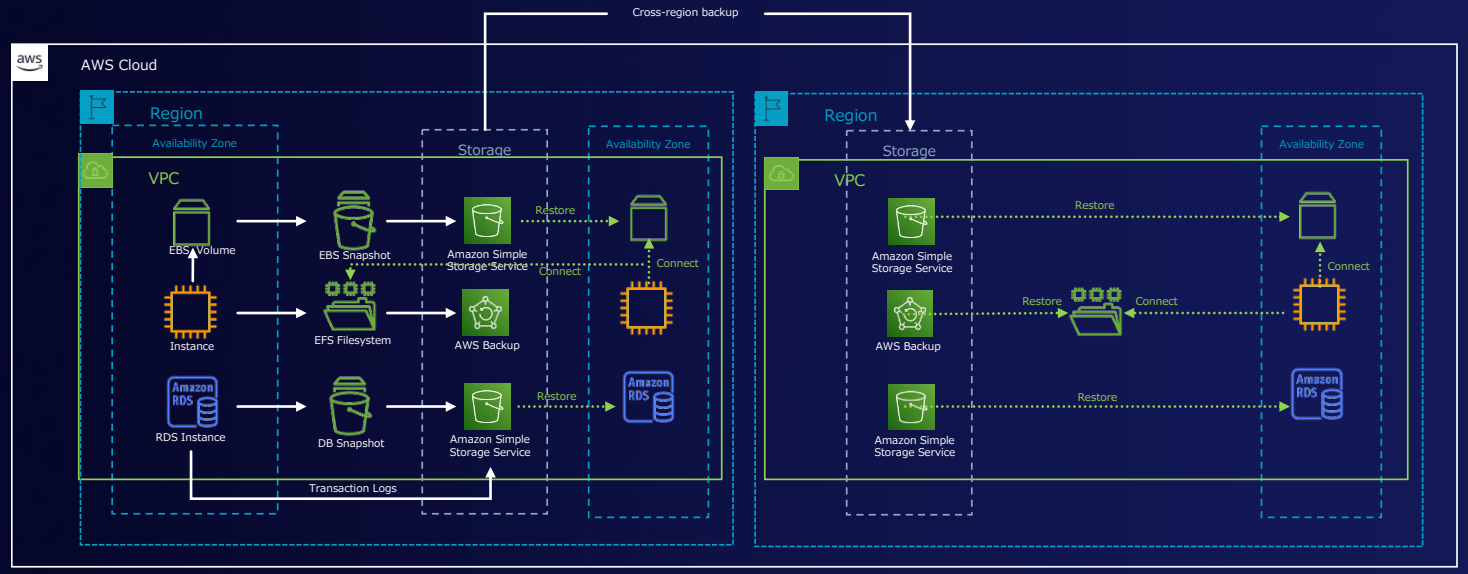
バックアップと復元

- 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

バックアップと復元：実装



準備

- 現行システムのバックアップを取得する
- バックアップを 可用性の高い データストア (Amazon S3 等) に保存する
- AWS 上で バックアップから復元する手順を記述する
  - どの AMI (Amazon マシンイメージ) を使用するかを確認し、必要に応じて独自に作成する
  - システムをバックアップから復元する方法を確認する
  - 新しいシステムに切り替える方法を確認する
  - デプロイの設定方法を確認する

災害発生時

- バックアップを Amazon S3 から取り出す
- システムをバックアップから復元し、必要なインフラストラクチャを起動する
  - Amazon EC2 インスタンスを AMI から起動 、Amazon EFS Filesystem をバックアップから復元、Amazon RDS インスタンスをスナップショットから起動
  - AWS CloudFormation を 使用してコアネットワークのデプロイを自動化
- 新しいシステムに切り替える
  - DNS レコードがサブサイトを指定するように切り替える

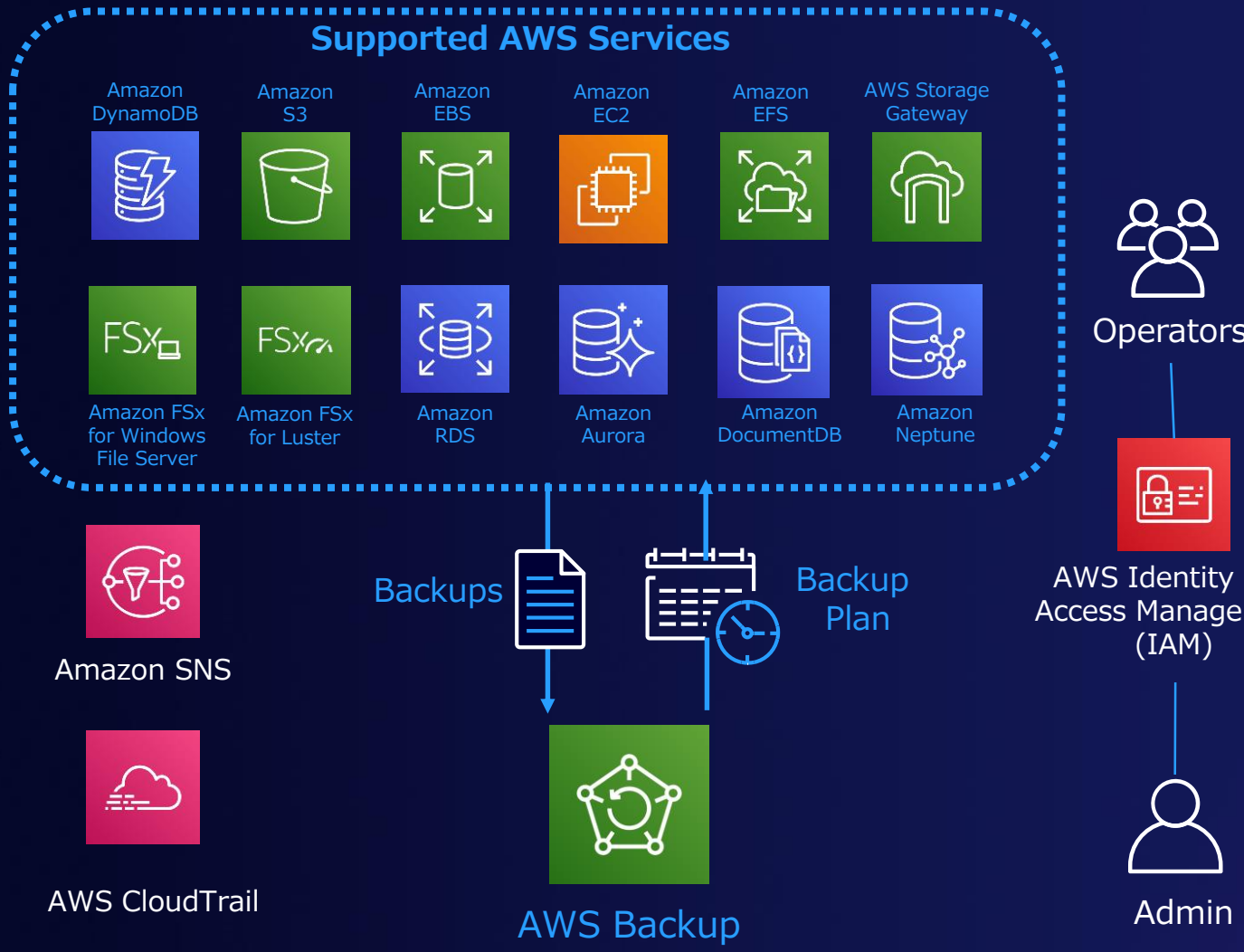
バックアップと復元：リージョン内のバックアップ

- マルチ AZ 戦略を検討
  - 99% ～ 99.99% の可用性※ は、マルチ AZ 構成で実現可能
- リージョン内でデータストアのバックアップを取得
- 対象リソースごとにバックアップと復元戦略を検討
  - バックアップ
    - AWS Backup
  - ポイントインタイムリカバリ
    - Amazon DynamoDB
    - Amazon RDS
  - バージョニング機能の有効化
    - Amazon S3

※可用性目標の実装例 (単一リージョンシナリオ)
https://docs.aws.amazon.com/ja_jp/wellarchitected/latest/reliability-pillar/single-region-scenarios.html

バックアップと復元：マネージドサービスの活用

AWS Backup



- 広範なサービスをカバーした
  マネージドバックアップサービス

- サポートされている機能：
  - バックアップスケジュールと保存管理の自動化
  - バックアップモニタリングの一元化
  - 増分バックアップ (※一部サービス除く)
  - AWS KMS 統合バックアップ暗号化
  - AWS Organizations によるアカウント間管理
  - AWS Backup Audit Manager による
    バックアップ監査とレポートの自動化
  - AWS Backup Vault Lock による
    WORM (Write-once, read-many)

- 対応リソース：
  - Amazon EC2, Windows ボリュームシャドウコピーサービス (VSS), Amazon S3, Amazon EBS, Amazon DynamoDB, Amazon RDS, Amazon Aurora, Amazon EFS, FSx for Lustre, FSx for Windows File Server, AWS Storage Gateway, Amazon DocumentDB, Amazon Neptune, VMware 仮想マシン

バックアップと復元：
別リージョンへのバックアップ

クロスリージョンバックアップのポイント

- 各種バックアップのコピー周期や

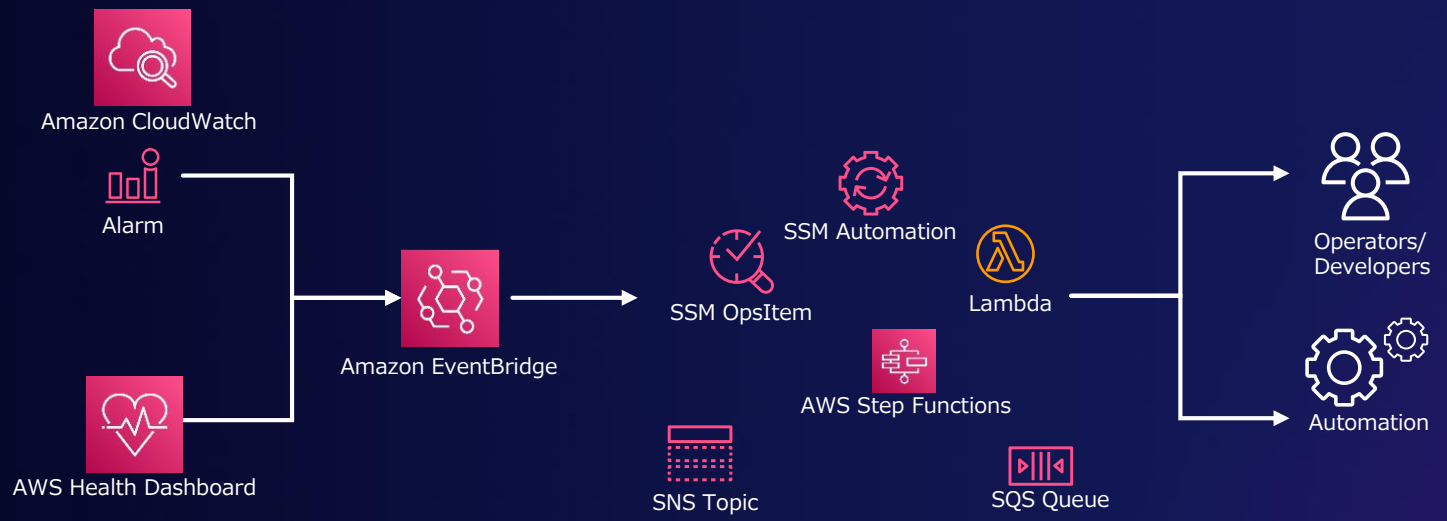
  データベースの同期方法は RPO 要件に従って定義する

- 様々な要件を満たせるクロスリージョンバックアップ機能を活用
  - AWS Backup
  - Amazon S3:
    - プレフィックス単位やタグベースでのレプリケーション、
      または、 SLA 付きのオプションなど柔軟性が高いクロスリージョンレプリケーション (CRR)
- RTO 要件を満たすプロセスの見直しやシステムデプロイのリハーサル

バックアップと復元：検出

Amazon EventBridge を活用した検出の自動化

- Amazon CloudWatch
  - Amazon CloudWatch Alarm でワークロードの動作に基づいたヘルスチェックを設定
  - Amazon CloudWatch Anomaly Detection を使い、
    サイトのエンゲージメント、注文やその他の主要パフォーマンス指標が低下したかどうかを検出
  - Amazon CloudWatch Synthetics でサービスを呼び出してレスポンスを検証するスクリプトを作成可能
- AWS Health Dashboard
  - AWS サービスの全体的ステータスの表示、お客様に影響を与える可能性のあるイベントを検出
  - Amazon EventBridge で検出したイベントを元に実行するアクションを指定



・・・ and more

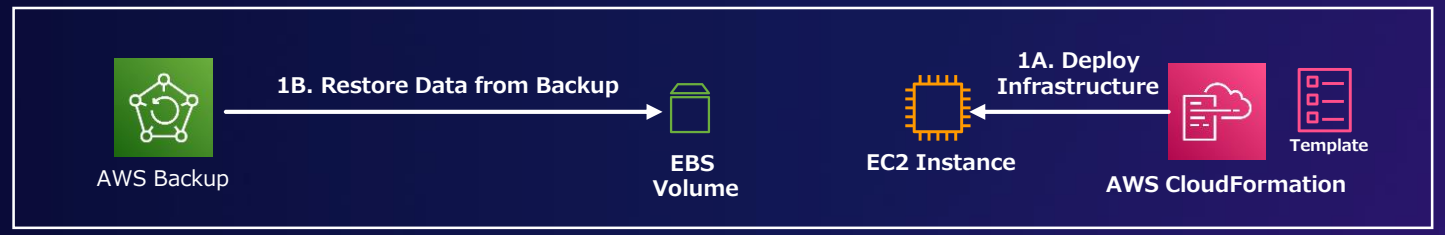
JSON

```
{
 "source": ["aws.health"],
 "detail-type": ["AWS Health Event"],
 "detail": {
  "service": ["S3"],
  "eventTypeCategory": ["issue"],
  "eventTypeCode":
["AWS_S3_INCREASED_GET_API_ERROR_RATES",
"AWS_S3_INCREASED_PUT_API_ERROR_RATES"]
 }
}
```

※Amazon S3 での PUT および GET オペレーションで
エラーレートの増加を引き起こしているイベントに
反応するように Amazon EventBridge を設定している例

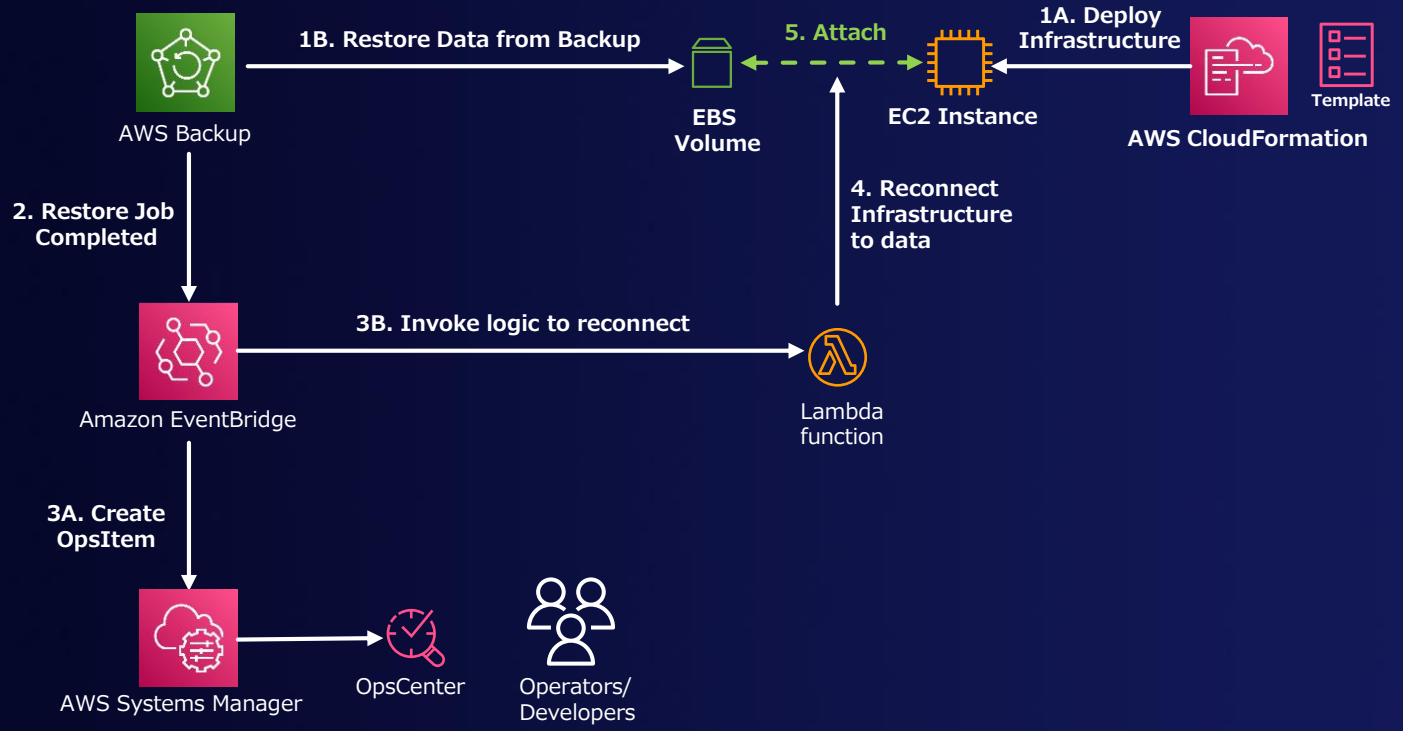
バックアップと復元：リカバリ

1. 復旧先リージョンのインフラストラクチャを
   復元を自動化するために、Infrastructure as Code (IaC) を使用
   - AWS CloudFormation 、 AWS Cloud Development Kit (AWS CDK) 等
2. Amazon マシンイメージ（AMI）を使用して、ワークロードに必要な EC2インスタンスを作成
   - 必要なオペレーティングシステムとパッケージを組み込む
3. AMI には、一貫性のあるインスタンスの
   起動に必要なコンポーネントがすべて含まれた「ゴールデン AMI」を準備しておく
4. Amazon EC2 Image Builder を使用して、ゴールデン AMI を作成し、復旧先リージョンにコピー
5. インフラストラクチャとデータを再統合
   - EBS ボリューム内のステートフルデータをリカバリ
   - ボリュームを EC2 インスタンスに再アタッチ



図：バックアップからのデータの復元と復旧先リージョンのインフラストラクチャの再構築

バックアップと復元：リカバリ



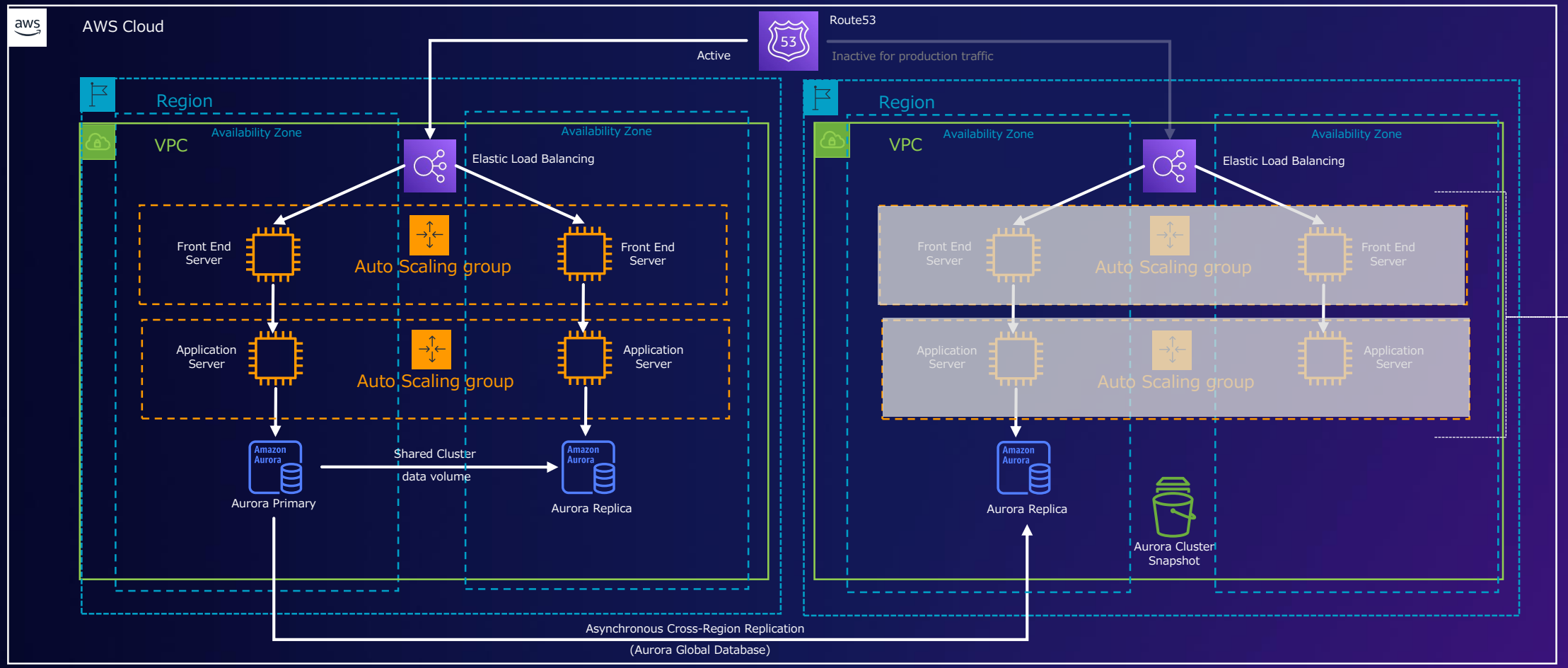
図：ワークロードの復元の一環としてインフラストラクチャとデータの統合を自動化

Amazon EventBridge を使用したサーバーレスソリューション

1. AWS CloudFormation で必要なインフラストラクチャをデプロイし、バックアップからデータを復元する
2. AWS Backup がリソースのデータ復元完了後にイベントを発行
3. Amazon EventBridge が 2 つのアクションを開始
   - 3A）記録のために AWS Systems Manager OpsCenter でアイテムを作成
   - 3B) Lambda 関数を呼び出す
4. イベントから Lambda は以下のアクションを実施
   1. 復旧先リージョンで作成された新しいリソース ID と復旧に使用されたバックアップ (復旧ポイント) ID を取得
   2. AWS Backup がリソースから作成したバックアップにタグを自動コピー
   3. AWS CloudFormation への別の API コールで、Lambda が EC2 インスタンスの ID を学習
5. Amazon EC2 の API を呼び出し、EBS ボリュームをアタッチ

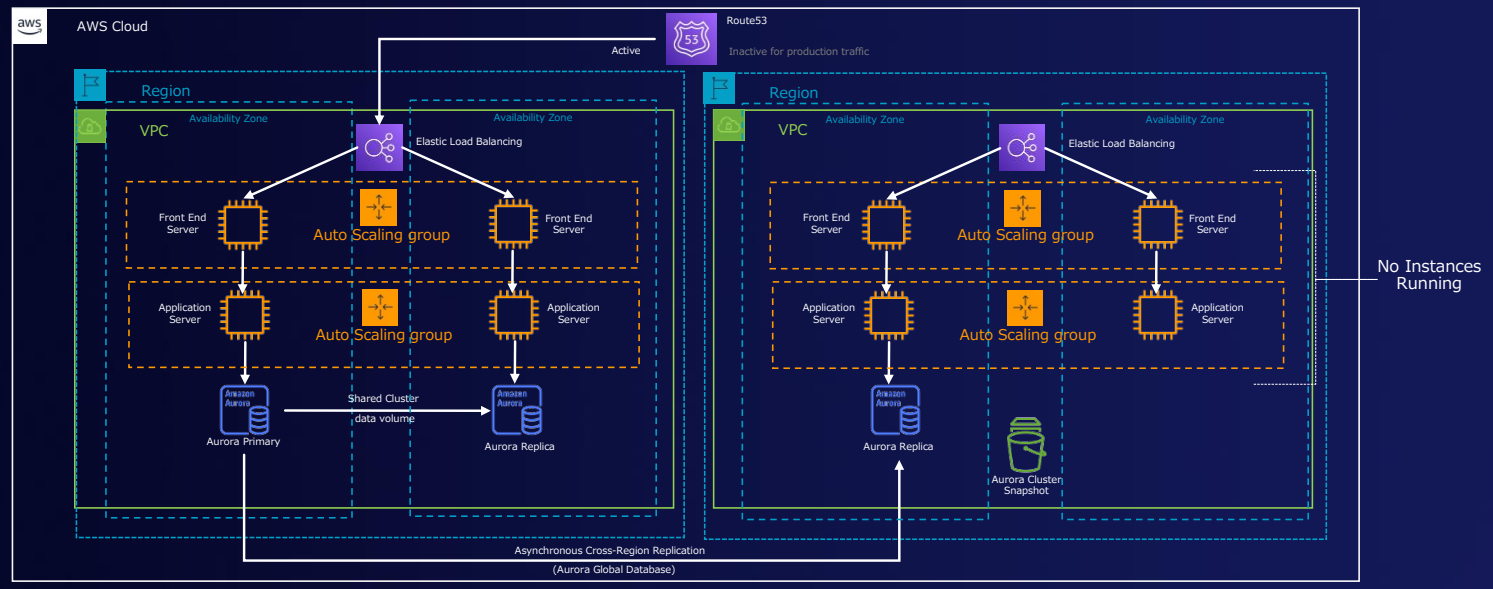
パイロットライト

- 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

パイロットライト：実装



準備

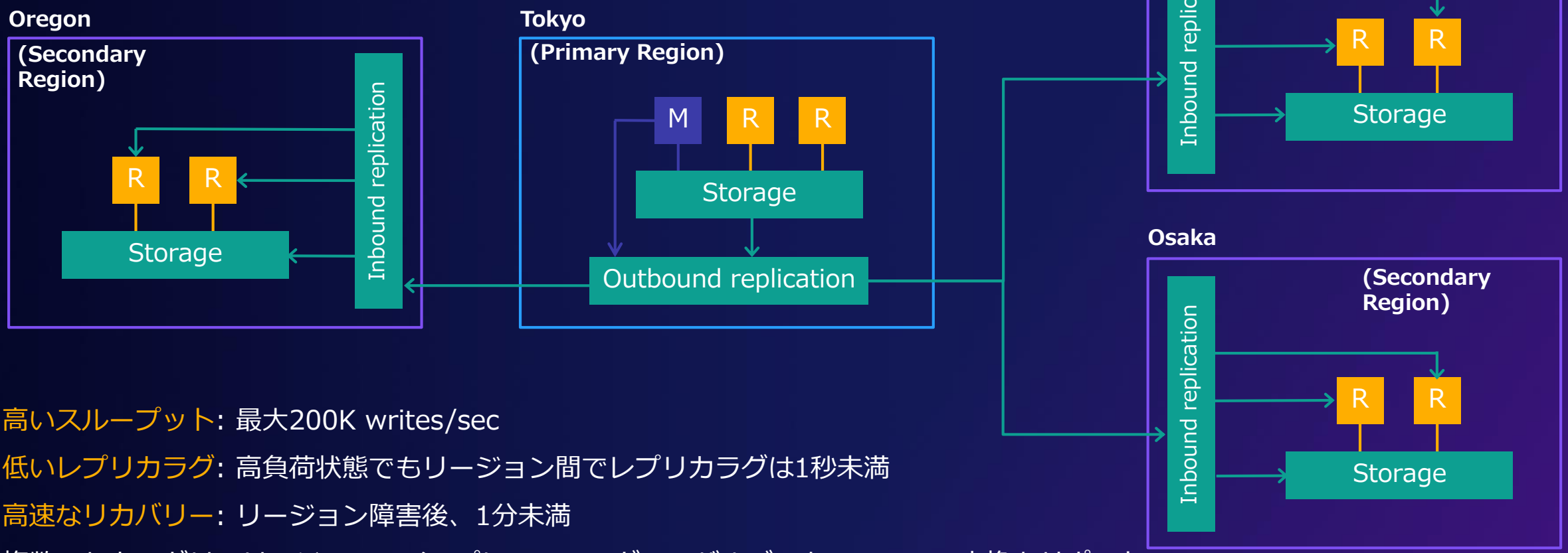
- 重要なコアデータセットを
  レプリケートまたはミラーリングする
- サブサイトでカスタムソフトウェアのパッケージ
  すべてが AWS で使用できることを確認する
- 高速復旧が必要な主要サーバーの Amazon
  マシンイメージ（AMI）を作成し、維持する
- それらのサーバーの実行とテストを定期的に
  行い、ソフトウェアの更新と構成変更を適用する
- AWS リソースの
  プロビジョニングを自動化することを検討する

災害発生時

- レプリケート済みのコアデータセットを
  中心として自動的にリソースを起動する
- その時点の本番トラフィックを処理するために
  必要に合わせてシステムをスケールする
- 新しいシステムに切り替える
  - DNS レコードがサブサイトを指定するように切り替える

パイロットライト：リカバリ

Amazon Aurora グローバルデータベース



高いスループット: 最大200K writes/sec

低いレプリカラグ: 高負荷状態でもリージョン間でレプリカラグは1秒未満

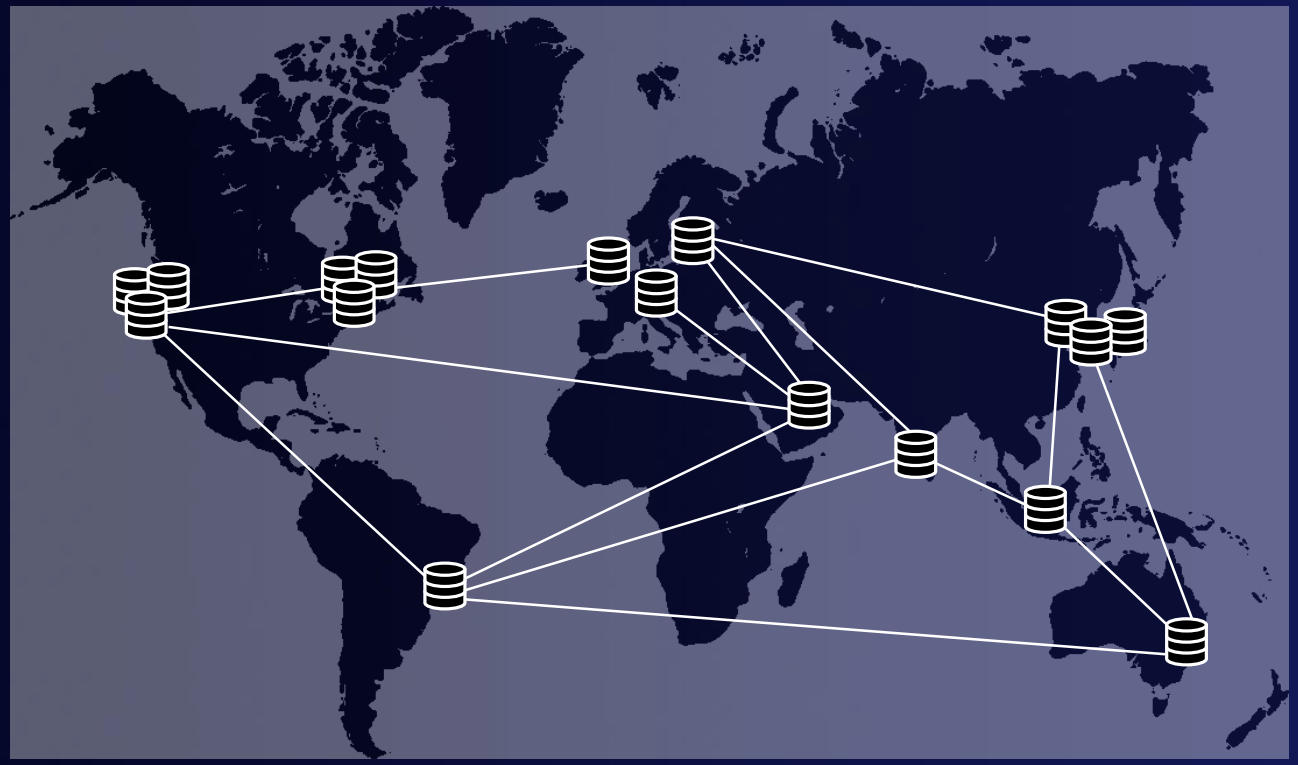
高速なリカバリー: リージョン障害後、1分未満

複数のセカンダリーリージョン、インプレースでのグローバルデータベースへの変換をサポート

詳細：https://docs.aws.amazon.com/ja_jp/AmazonRDS/latest/AuroraUserGuide/aurora-global-database.html

パイロットライト：リカバリ

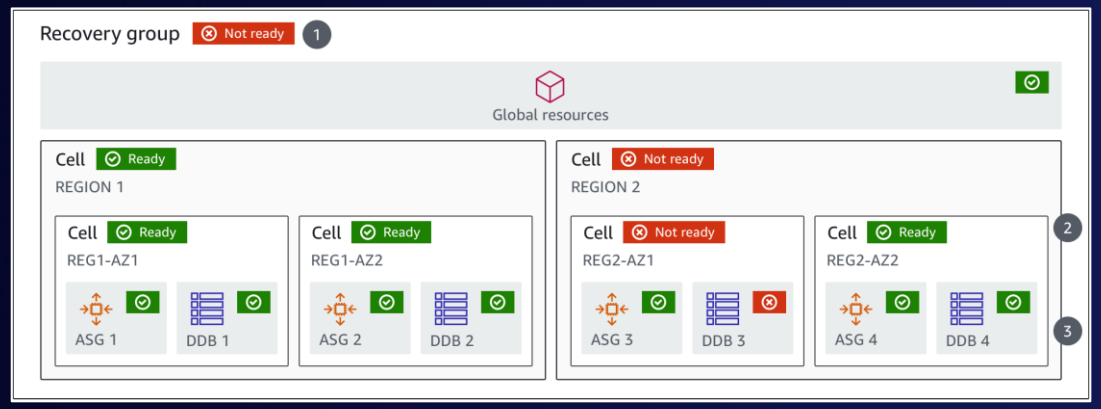
Amazon DynamoDB グローバルテーブル

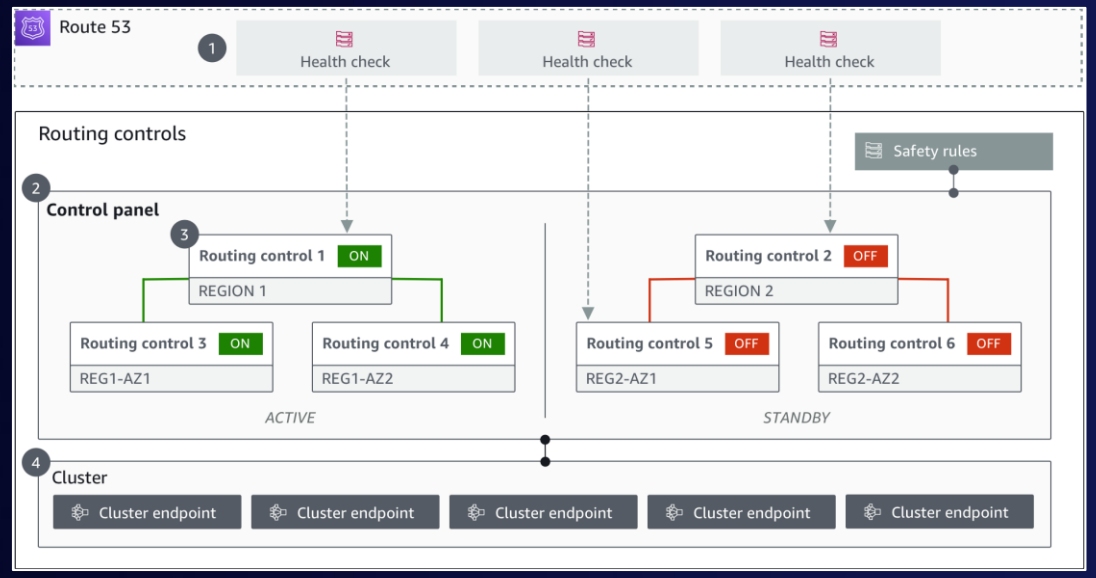


- 高性能でグローバルに分散されたアプリケーションを構築する
- ローカルテーブルへの低レイテンシーの読み込みと書き込み
- マルチリージョンの冗長性と弾力性
- 設定が簡単で、アプリケーションの書き換えは不要

パイロットライト：リカバリ

Amazon Route 53 Application Recovery Controller





Readiness Check

- アプリケーションが
  フェイルオーバーにどの程度対応できるかを確認可能
- アプリケーションのリソースが全体 (リカバリグループ) として、
  またはリージョン別、アベイラビリティーゾーン別に
  (セルとして)、どの程度正常かを確認できる
  - 例：全リージョンの
    DynamoDB テーブルを調べて、ステータスが ACTIVE であるか

Routing Control

- Amazon Route 53 ヘルスチェックと連携して、
  DNS レゾリューションを使用してトラフィックを
  アプリケーションレプリカにリダイレクト
- アプリケーションメトリクスまたは部分的な障害 (5% の
  エラー率の増加、ミリ秒単位のレイテンシーの増加など) に
  基づいて、アプリケーションスタック全体をフェイルオーバー可能

パイロットライト：リカバリ

YAML

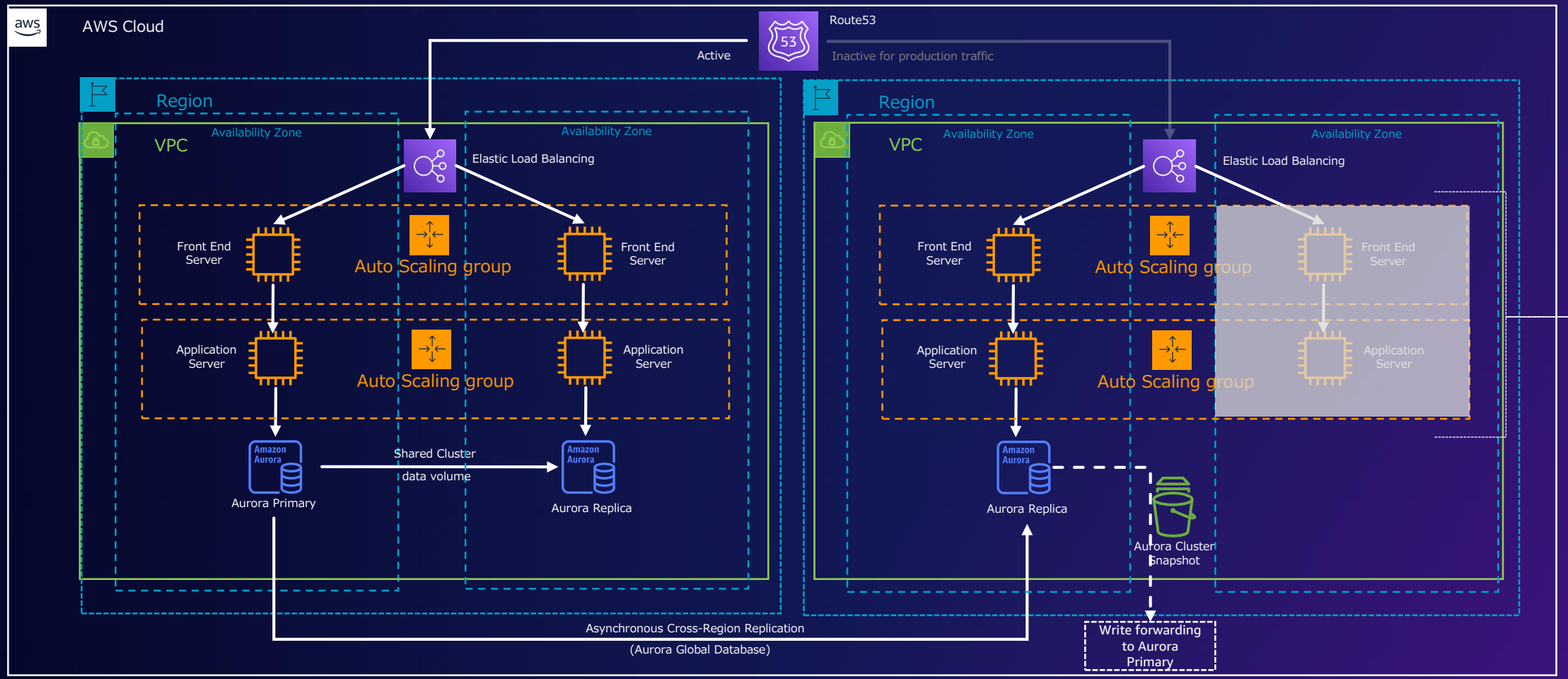
```yaml
Parameters:
  Web1AutoScaleDesired:
    Default: '3'
    Description: The desired number of web instances in auto scaling group
    # other properties omitted
  Web1AutoScaleMax:
    Default: '6'
    Description: The maximum number of web instances in auto scaling group
    # other properties omitted
  ActiveOrPassive:
    Default: 'active'
    Description: Is this the active (primary) deployment or the passive (recovery) deployment
    Type: String
    AllowedValues:
      - active
      - passive
    ConstraintDescription: enter active or passive, all lowercase

# Determine whether this is an active stack or passive stack
Conditions:
  IsActive: !Equals [!Ref "ActiveOrPassive", "active"]

Resources:
  #https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-as-group.html
  WebAppAutoScalingGroup:
    Type: 'AWS::AutoScaling::AutoScalingGroup'
    Properties:
      MinSize: !If [IsActive, !Ref Web1AutoScaleDesired, 0]
      MaxSize: !Ref Web1AutoScaleMax
      DesiredCapacity: !If [IsActive, !Ref Web1AutoScaleDesired, 0]
      # other properties omitted
```

AWS CloudFormation を活用して
AWS リソースのプロビジョニングを自動化

- AWS CloudFormation ではパラメータと条件関数を使用して、アクティブスタックまたは待機スタックの両方を 1 つのテンプレートで構築可能
- 左部コードは、可用性の高い静的 Web アプリケーションを構築するためのサンプル CloudFormation テンプレートから一部抜粋したもの
  - テンプレート全体はこちら
- パラメータ ActiveOrPassive に “active” または “passive” を指定することで、EC2 インスタンスがデプロイされるかデプロイされないかを決定できる

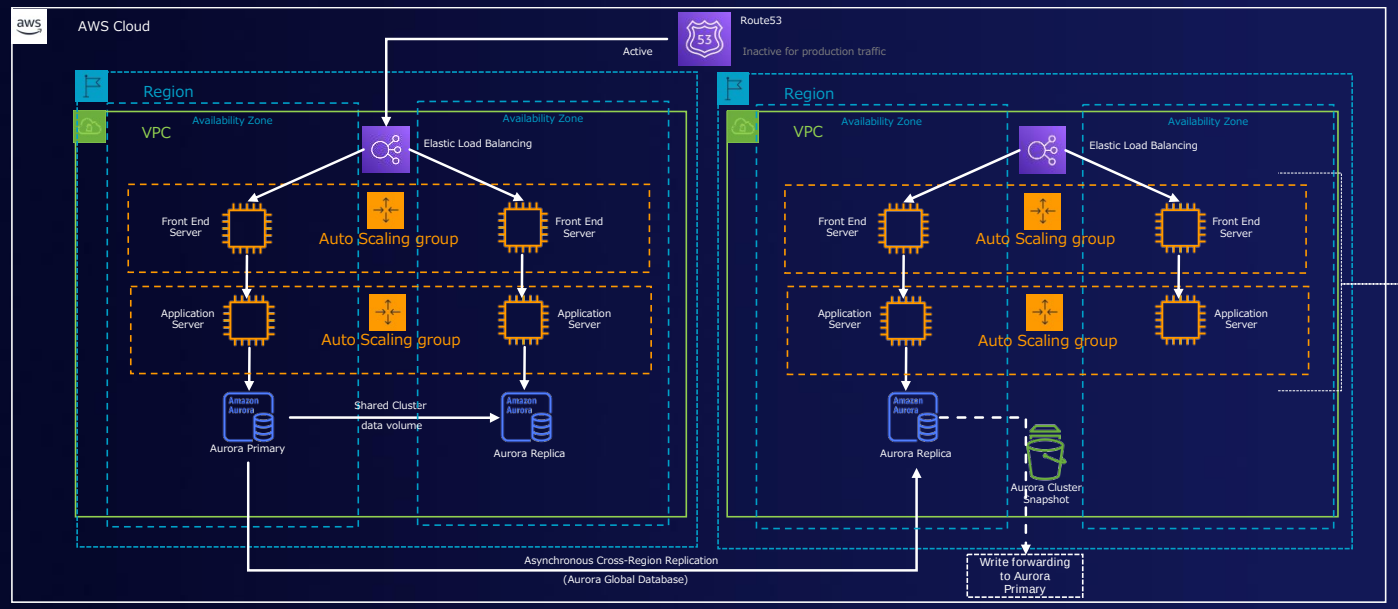
ウォームスタンバイ

- 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

ウォームスタンバイ：実装



準備

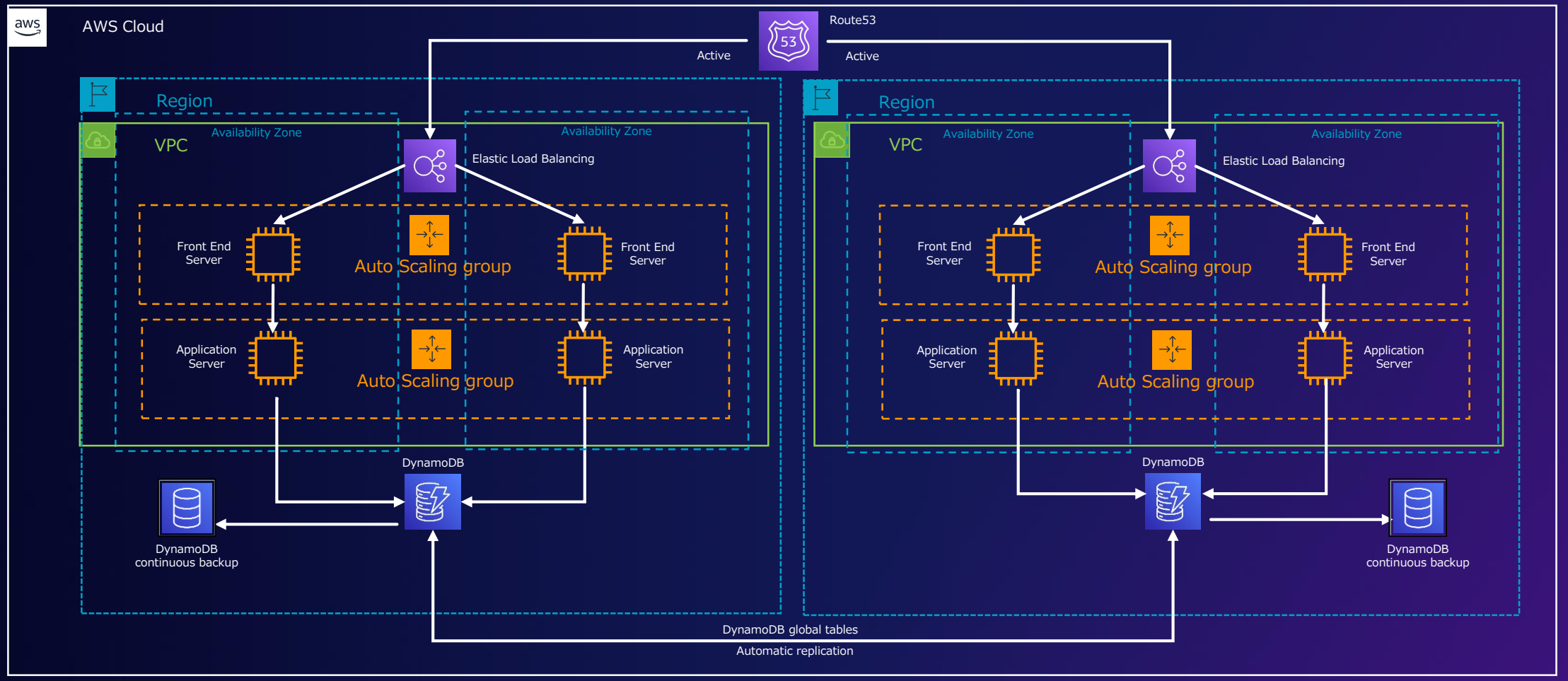
- パイロットライトと同様
- 必要なコンポーネントは
  すべて常時稼働しているが、
  本番トラフィックに合わせたスケールではない
- ベストプラクティス: 継続的なテスト
  - 本番トラフィックをサンプリングして DR サイトに "流す"

災害発生時

- 最も重要な
  本番負荷を即座にフェイルオーバーする
  - DNS レコードが AWS を指定するように切り替える
- 本番負荷をすべて処理できるよう、
  システムをさらに (自動) スケーリングする

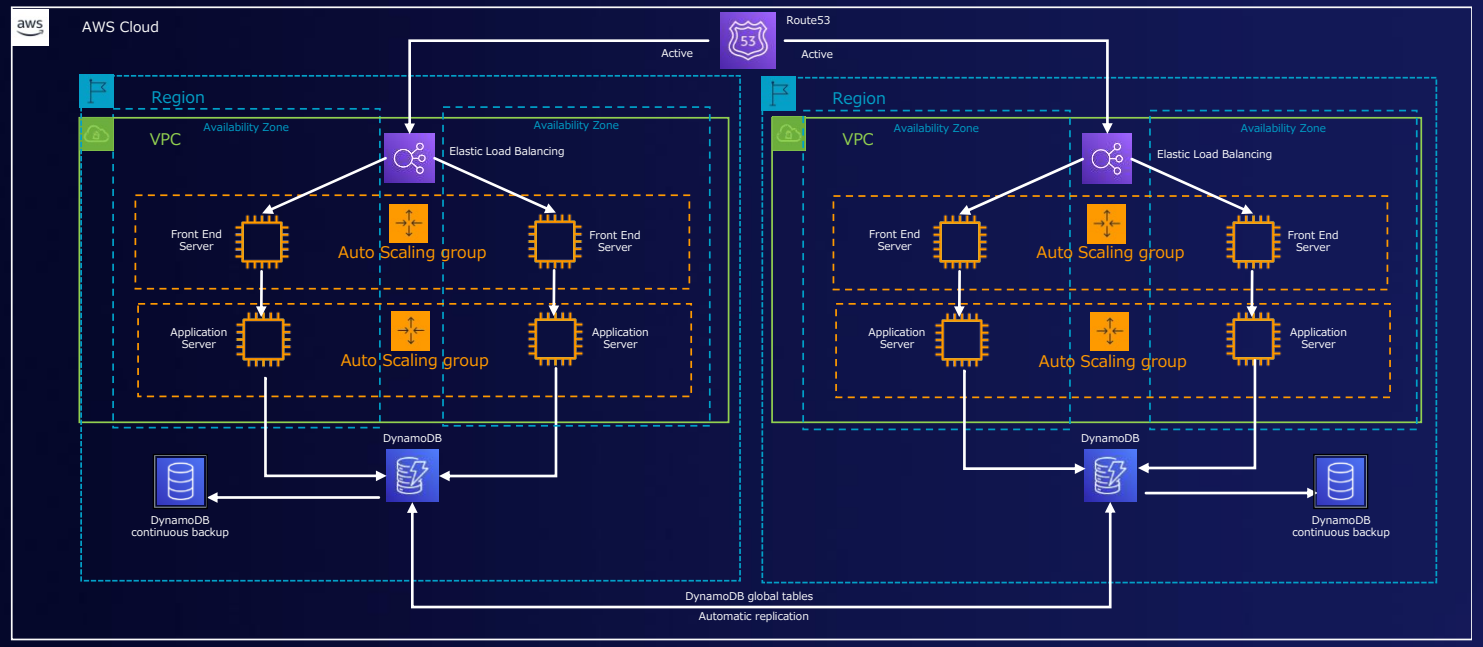
マルチサイトアクティブ/アクティブ

- 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

マルチサイトアクティブ/アクティブ：実装



準備

- ウォームスタンバイに類似
- 本番負荷に合わせて完全にスケールイン/アウト

災害発生時

- すべての本番負荷を即座にフェイルオーバーする

フェイルオーバー

- Amazon Route53 による DNS フェイルオーバー
- AWS Global Accelerator によるフェイルオーバー

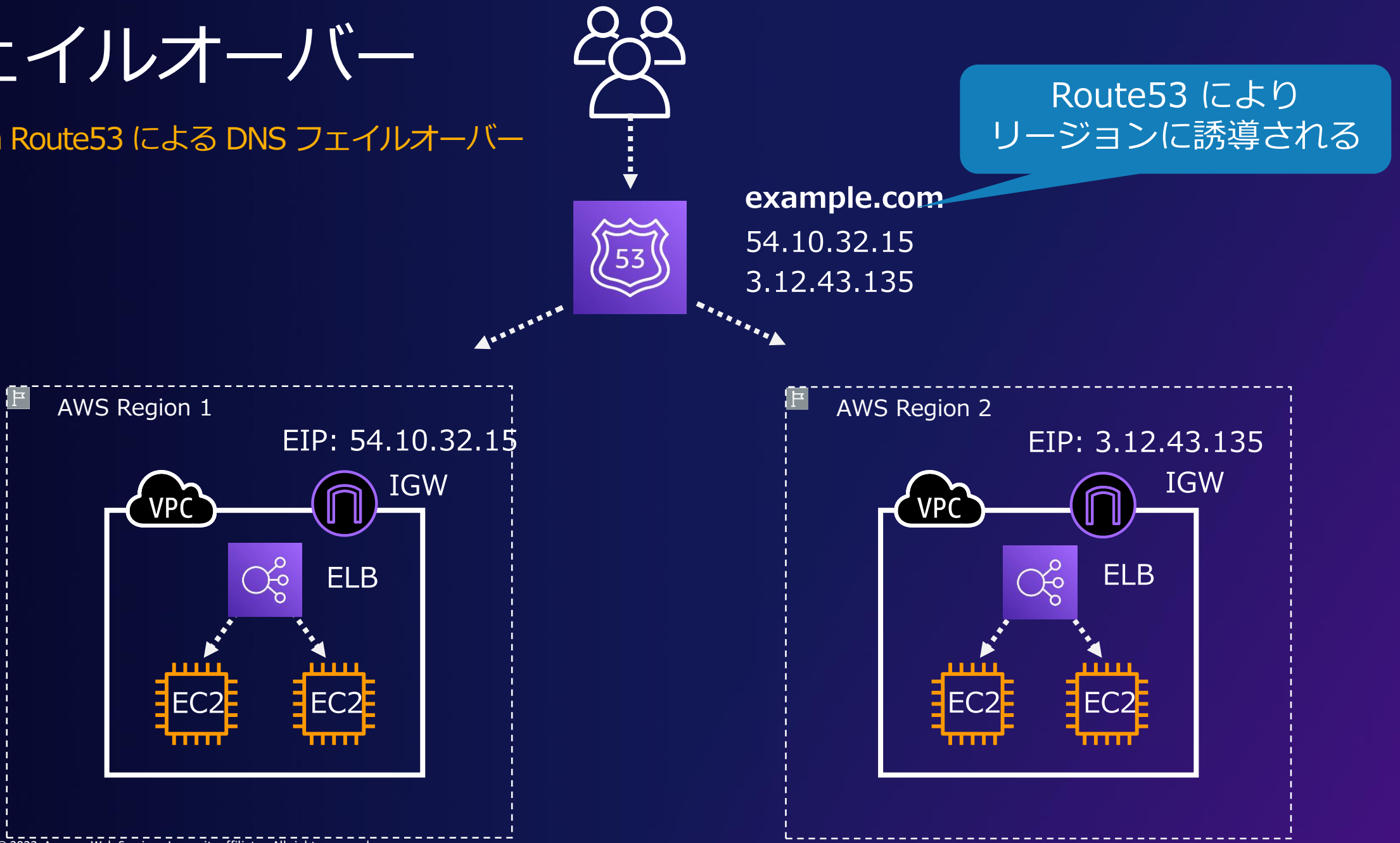
フェイルオーバー

- Amazon Route53 による DNS フェイルオーバー

- AWS Global Accelerator によるフェイルオーバー

フェイルオーバー

Amazon Route53 による DNS フェイルオーバー

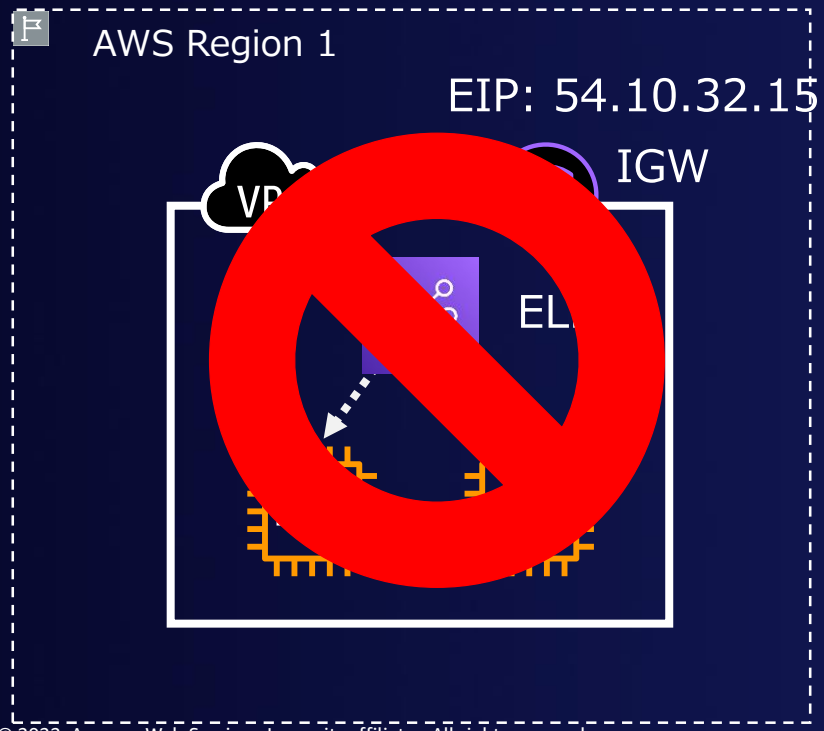


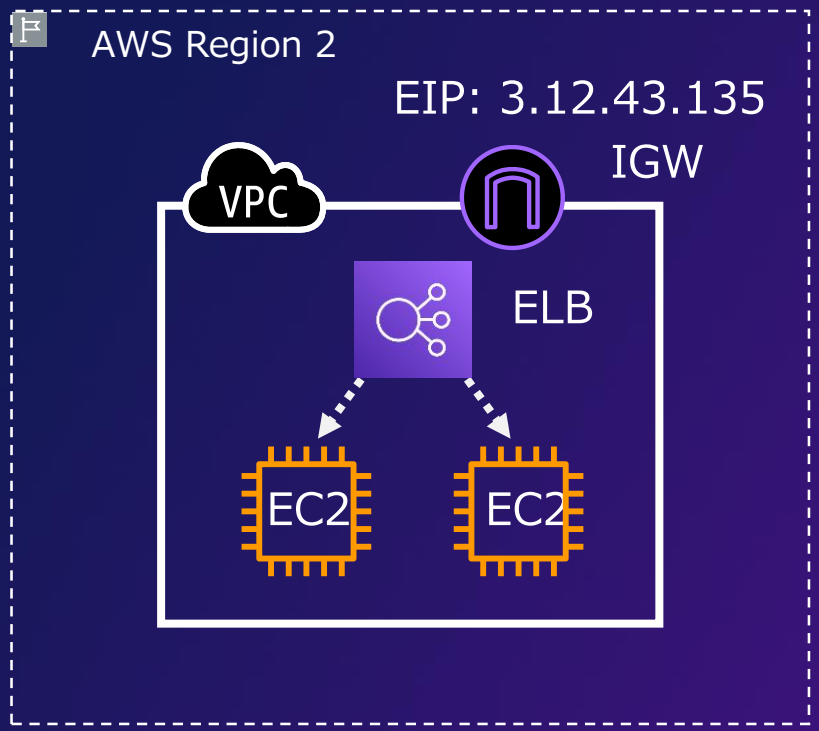
フェイルオーバー

Amazon Route53 による DNS フェイルオーバー

example.com
54.10.32.15
3.12.43.135

メインリージョン障害時は、
設定したヘルスチェックに
基づき、アクティブな
リージョンに誘導される





(参考) Amazon Route53 で設定可能なルーティングポリシー

シンプルルーティング (Simple)

- レコードセットで事前に設定された値のみに基づいて DNS クエリに応答する
- 従来の DNS と同様に、静的なマッピングによりルーティングが決定される

加重ルーティング (Weighted)

- 複数エンドポイント毎に設定された重みづけに基づいて、DNS クエリに応答する
- より重み付けの高いエンドポイントのリソースに、より多くルーティングされる

レイテンシールーティング (Latency)

- AWS リージョンとの遅延によって、DNS クエリに応答する
- リージョン間の遅延が少ない方のリソースへルーティングされる

フェイルオーバールーティング (Failover)

- ヘルスチェックの結果に基づいて、利用可能なリソースを DNS クエリに応答する
- 利用可能なリソースにのみルーティングされる

位置情報ルーティング (Geolocation)

- クライアントの位置情報に基づいて、DNS クエリに応答する
- 特定の地域・国からの DNS クエリに対して、特定のアドレスを応答する

地理的接近性ルーティング (Geoproximity)

- リソースの場所に基づいてトラフィックをルーティング
- 必要に応じてトラフィックをある場所の
  リソースから別の場所のリソースに移動する場合に使用する

ルーティングポリシーの選択 - Amazon Route 53
https://docs.aws.amazon.com/ja_jp/Route53/latest/DeveloperGuide/routing-policy.html

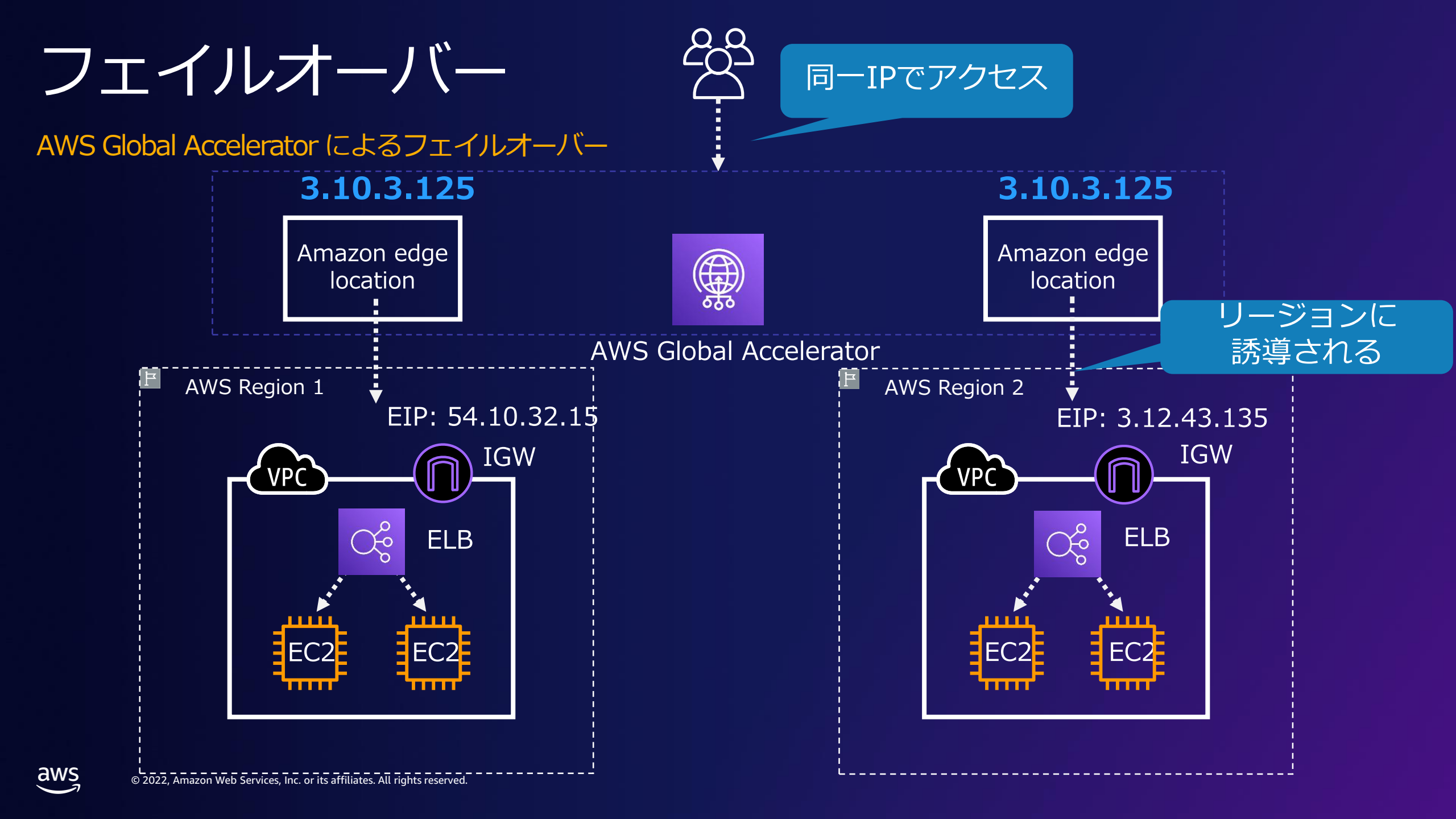
フェイルオーバー

- Amazon Route53 による DNS フェイルオーバー

- AWS Global Accelerator によるフェイルオーバー

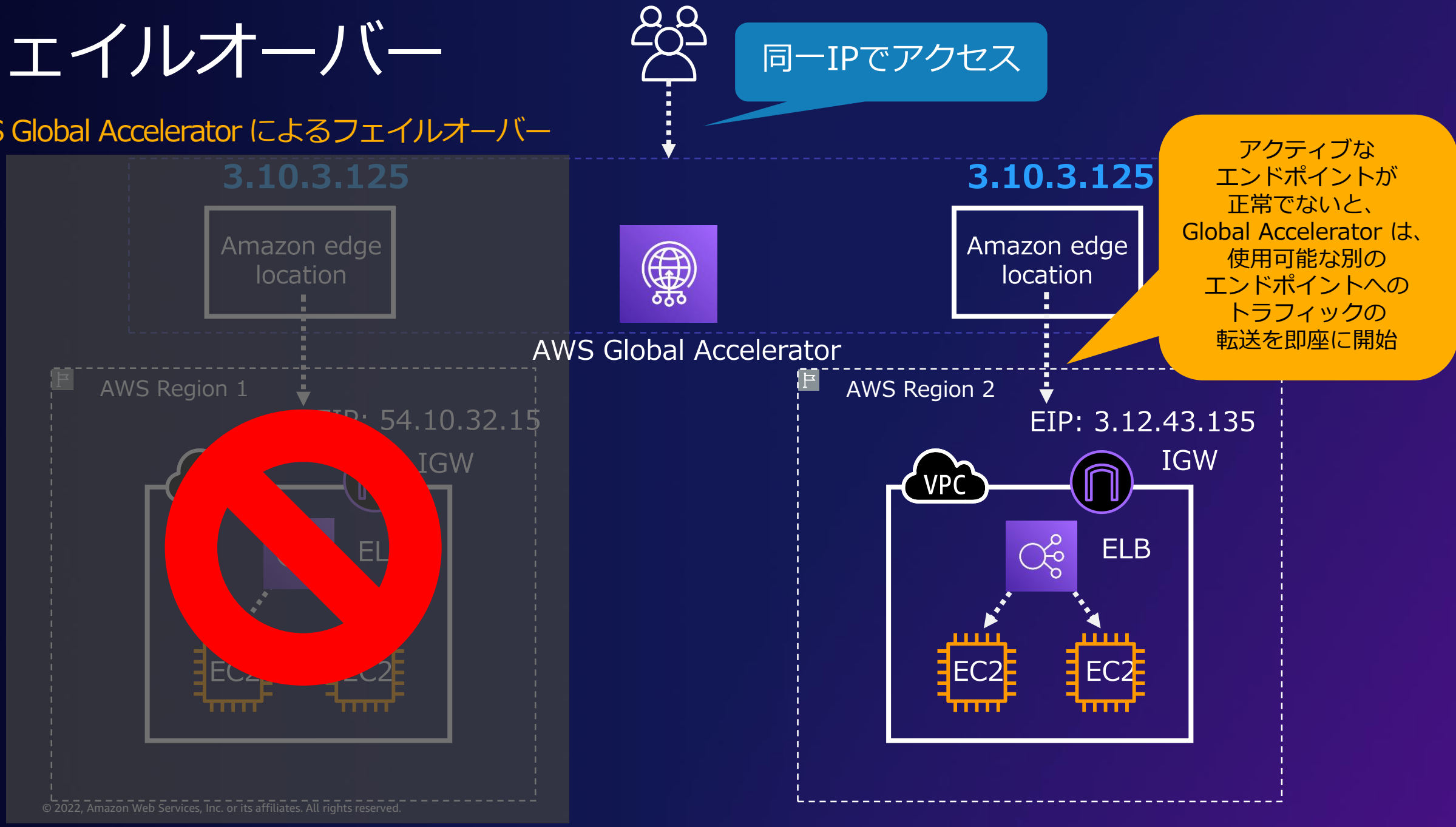
フェイルオーバー

AWS Global Accelerator によるフェイルオーバー



フェイルオーバー

AWS Global Accelerator によるフェイルオーバー



Amazon Route53 と AWS Global Accelerator 比較

Amazon Route53

- DNS ベースでの分散
- ヘルスチェックによる
  フェイルオーバー
- 様々なアルゴリズムによる
  ロケーション選択

AWS Global Accelerator

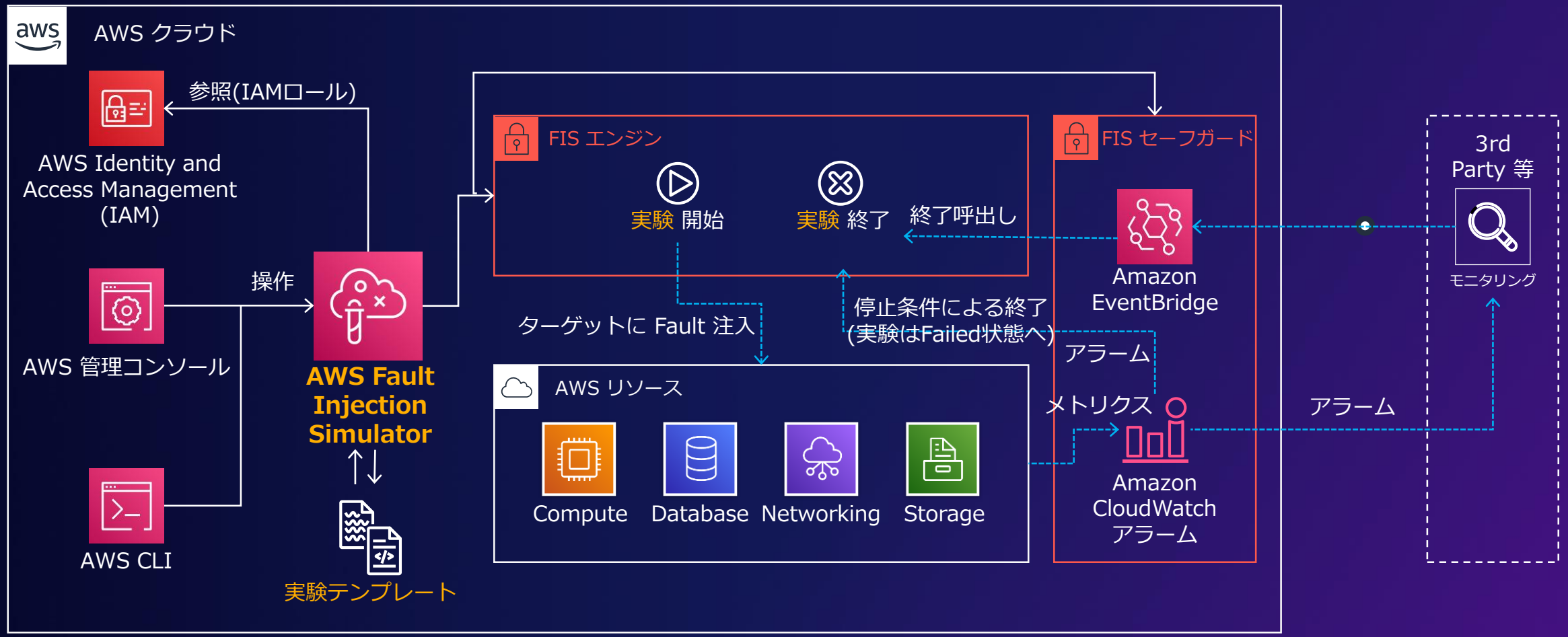
- AWS の Edge Location による
  負荷分散
- ヘルスチェックによる
  フェイルオーバー
- 固定 IP によるアクセス
- AWS バックボーン
  ネットワークによる高速化

ユースケースに合わせて選択

ディザスタリカバリ (DR) のテスト

AWS Fault Injection Simulator の使用

リソース横断での「実験 (Experiment)」開始による 障害状況 の注入と
指定条件をトリガーとした 終了 をフルマネージドサービスとして提供



準備のためのベストプラクティス

- 最初はシンプルに開始し、徐々にレベルアップする
  - AWS のバックアップを最初のステップとする
  - 継続的な努力によって RTO/RPO を徐々に改善する
- ソフトウェアライセンシングの問題があるかどうかを調べる
- DR ソリューションの練習を行う
  - 本番を想定した予行演習を実施する
  - AWS Fault Injection Simulator を利用する
  - バックアップ、スナップショット、AMI などが正常に利用できることを確認する
  - モニタリングシステムをモニタリングする

AWS の災害対策の利点

- さまざまなサービスを利用できる
- コスト対 RTO/RPO の
  トレードオフを細かくコントロールできる
- DR 計画を簡単かつ効果的にテストできる
- 世界各地の複数の場所を利用できる
- 多様なソリューション
  プロバイダー (AWS パートナー) を利用可能

アーキテクチャベストプラクティスのポートフォリオ

ACCELERATE CLOUD ADOPTION WITH CONFIDENCE

EXPERTISE

AWS Well-Architected Framework

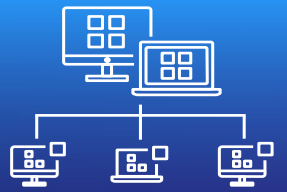
運用上の優秀性、セキュリティ、信頼性、パフォーマンス効率、コスト最適化、持続可能性

Sustainable Architecture

エネルギー削減とワークロード効率向上により、クラウドの持続可能性を強化

AWS Solutions

すぐに展開できる AWS ソリューションを使用するか、事前構成されたアーキテクチャパターンを使用して構築プロセスを開始する



Architecture Center

リファレンスアーキテクチャダイアグラム、ベストプラクティスなど、充実したコンテンツライブラリにアクセス

AWS SERVICES

AWS Well-Architected Framework と AWS Solutions を活用したアーキテクチャベストプラクティスの採用

AWS Well-Architected Tool を使用してワークロードを評価し、アーキテクチャにおける高リスクな問題を把握できる

すぐに展開できる AWS ソリューションを使って、特定の技術およびビジネス的な課題を解決する

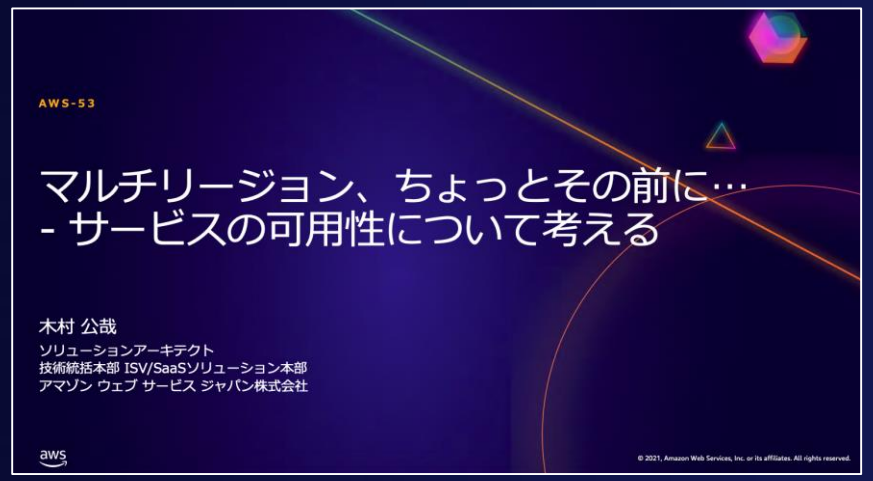
まとめ

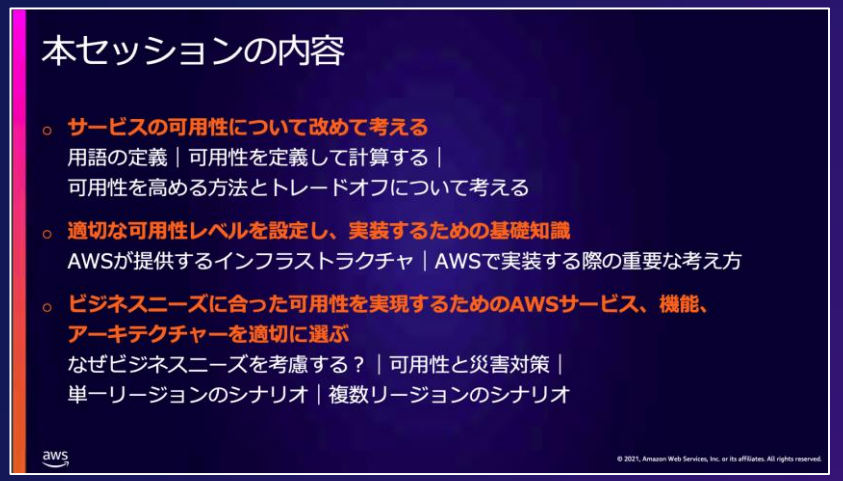
- 災害イベントとは
  - ワークロードまたはシステムが主要な場所でビジネス目標を達成するのを妨げるイベント
- RTO/RPO や可用性要件に基づいて、DR 戦略を検討する
  - RTO/RPO の確認
  - まずはマルチ AZ 戦略で可用性要件を満たせているか検討
  - マルチリージョン戦略は可用性要件だけでなく、ビジネスの継続性の観点も考慮して設計を検討
- AWS で検討可能な DR 戦略

  - バックアップと復元
    - 開始が簡単
    - 極めてコスト効果が高い (主にバックアップストレージ)
  - パイロットライト
    - コアサービスのみに限定したサブサイトを運用 (DB のみ等)
    - コスト効果が高い (常時稼働リソースが少ない)
  - ウォームスタンバイ
    - 本番トラフィックをいつでも、ある程度まで処理できる
    - フル DR よりも小さい IT フットプリント
  - マルチサイトアクティブ/アクティブ
    - いつでも本番負荷を処理できる
    - 障害時は自動フェイルオーバー

参考

- マルチ AZ 戦略にフォーカスした内容をご覧になりたい方は
  - AWS Summit Online Japan 2021
    ～ マルチリージョン、ちょっとその前に...
    サービスの可用性について考える ～
    - 資料： https://bit.ly/3I4oSzH
    - 動画： https://bit.ly/3sZ3VSv





Thank you!

Takanori Ohba (大場 崇令)

@takaohba

Resources

- AWS Well-Architected フレームワーク ホワイトペーパー
  - https://docs.aws.amazon.com/ja_jp/wellarchitected/latest/framework/welcome.html
- AWS Well-Architected フレームワーク 信頼性の柱 ホワイトペーパー
  - https://docs.aws.amazon.com/ja_jp/wellarchitected/latest/reliability-pillar/welcome.html
- Disaster Recovery of Workloads on AWS: Recovery in the Cloud ホワイトペーパー
  - https://docs.aws.amazon.com/ja_jp/whitepapers/latest/disaster-recovery-workloads-on-aws/disaster-recovery-workloads-on-aws.html
- Disaster Recovery (DR) Architecture on AWS, Part1 – 4 Blog
  - https://aws.amazon.com/jp/blogs/news/tag/disaster-recovery-series/
- AWS Well-Architected Lab
  - https://www.wellarchitectedlabs.com/