



AWS Well-Architected フレームワーク

でのワークロードのディザスタリカバリ AWS: クラウドでのリカバリ



でのワークロードのディザスタリカバリ AWS: クラウドでのリカバリ: AWS Well-Architected フレームワーク

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon の商標およびトレードドレスは、Amazon のものではない製品またはサービスと関連付けてはならず、また、お客様に混乱を招くような形や Amazon の信用を傷つけたり失わせたりする形で使用することはできません。Amazon が所有しない商標はすべてそれぞれの所有者に所属します。所有者は必ずしも Amazon と提携していたり、関連しているわけではありません。また、Amazon 後援を受けているとはかぎりません。

Table of Contents

要約	1
序章	2
ディザスタリカバリと可用性	2
Well-Architected の実現状況の確認	4
回復力に関する責任共有モデル	5
AWS 責任「クラウドの回復性」	5
お客様の責任「クラウドにおける回復力」	5
災害とは	7
高可用性はディザスタリカバリではありません	8
事業継続計画 (BCP)	9
ビジネスへの影響分析とリスク評価	9
復旧目標 (RTO と RPO)	10
クラウド内の災害対策は異なる	13
単一の AWS リージョン	13
複数の AWS リージョン	14
クラウド内での災害対策オプション	15
バックアップと復元	16
AWS サービス	17
パイロットライト	20
AWS サービス	21
AWS Elastic Disaster Recovery	24
ウォームスタンバイ	25
AWS サービス	26
マルチサイトアクティブ/アクティブ	26
AWS サービス	28
検出	30
ディザスタリカバリのテスト	31
結論	32
寄稿者	33
詳細情報	34
ドキュメント履歴	35
注意	36
AWS 用語集	37
.....	xxxviii

でのワークロードのディザスタリカバリ AWS: クラウドでのリカバリ

公開日: 2021 年 2 月 12 日 ([ドキュメント履歴](#))

ディザスタリカバリは、災害に備え、災害から復旧するプロセスです。ワークロードやシステムが主要なデプロイ先でビジネス目標を達成できないイベントは、災害と見なされます。このホワイトペーパーでは、にデプロイされたワークロードのディザスタリカバ리를計画およびテストするためのベストプラクティスの概要を示し AWS、リスクを軽減し、そのワークロードの目標復旧時間 (RTO) と目標復旧時点 (RPO) を満たすためのさまざまなアプローチを提供します。

このホワイトペーパーでは、でワークロードのディザスタリカバリを実装する方法について説明します AWS。 [オンプレミスワークロードのディザスタリカバリサイトとしてを使用する方法については、「オンプレミスアプリケーションの AWSディザスタリカバリ」](#)を参照してください。 AWS

序章

ワークロードは、意図した機能を正しく一貫して実行する必要があります。これを実現するには、回復性を考慮して設計する必要があります。回復力とは、ワークロードがインフラストラクチャ、サービス、またはアプリケーションの中断から回復し、需要に合わせてコンピューティングリソースを動的に取得し、設定ミスや一時的なネットワーク問題などの中断を軽減する能力です。

ディザスタリカバリ (DR) は耐障害性戦略の重要な部分であり、災害発生時のワークロードの対応 ([災害](#)はビジネスに重大な悪影響を与えるイベント) について懸念しています。このレスポンスは、[目標復旧時点 \(RPO\)](#) と呼ばれるデータの損失を回避し、[目標復旧時間 \(RTO\)](#) と呼ばれるワークロードが使用できないダウンタイムを削減するためのワークロードの戦略を指定する組織のビジネス目標に基づいている必要があります。したがって、特定の 1 回限りの災害イベントの復旧目標 ([RPO と RTO](#)) を満たすには、クラウド内のワークロードの設計に回復力を実装する必要があります。このアプローチは、[事業継続計画 \(BCP\) の一環として事業継続性](#)を維持するのに役立ちます。

このホワイトペーパーでは、ビジネスのディザスタリカバリ目標 AWS を満たすアーキテクチャを計画、設計、実装する方法に焦点を当てています。ここで共有される情報は、最高技術責任者 (CTOs)、アーキテクト、デベロッパー、運用チームメンバー、リスクの評価と軽減を担当するテクノロジー担当者を対象としています。

ディザスタリカバリと可用性

ディザスタリカバリは、耐障害性戦略のもう 1 つの重要な要素である可用性と比較できます。ディザスタリカバリは 1 回限りのイベントの目標を測定しますが、可用性目標は一定期間の平均値を測定します。

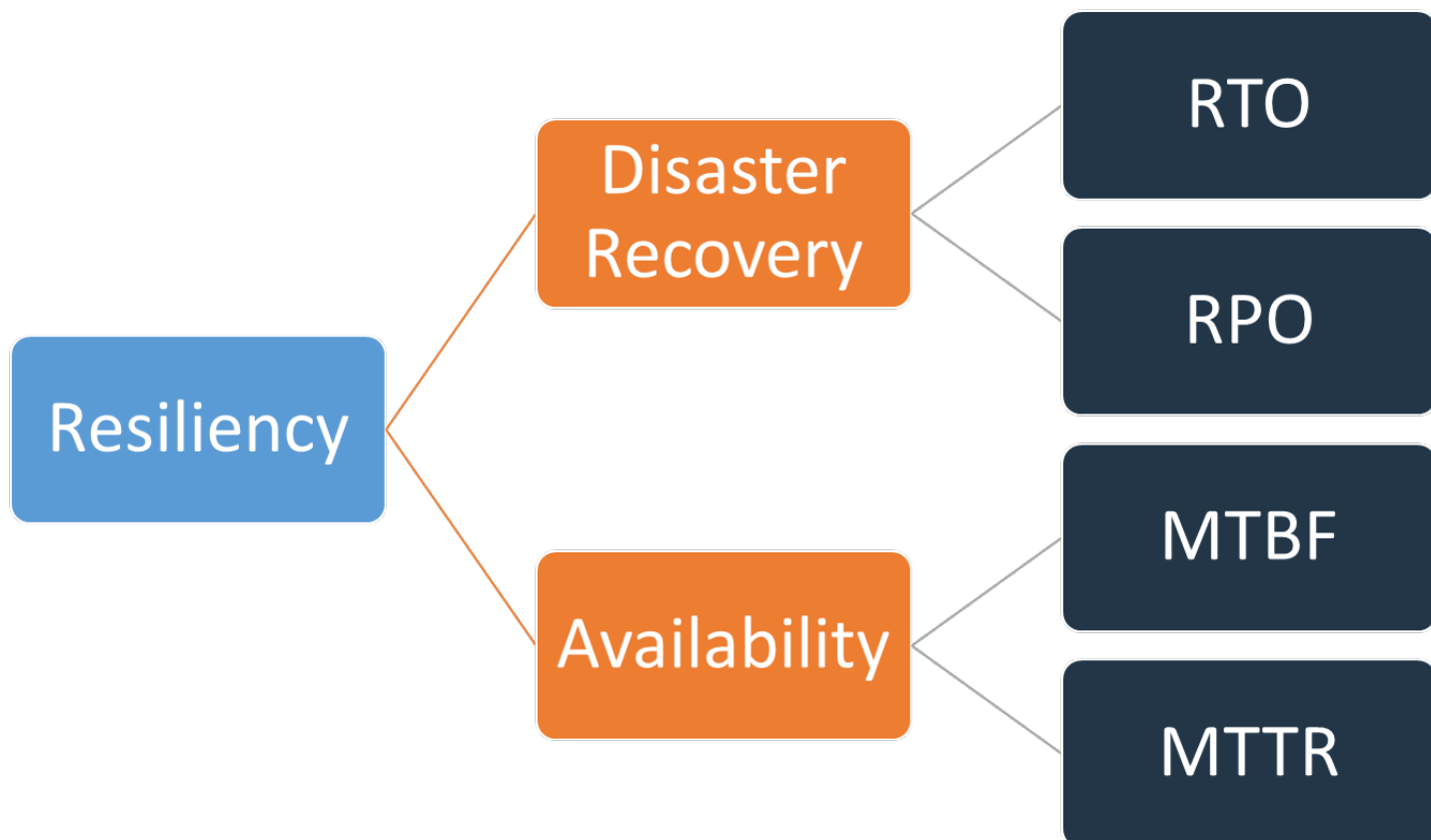


図 1 - 耐障害性の目的

可用性は、平均障害間隔 (MTBF) と平均復旧時間 (MTTR) を使用して計算されます。

$$Availability = \frac{Available\ for\ Use\ Time}{Total\ Time} = \frac{MTBF}{MTBF + MTTR}$$

このアプローチは「nines」と呼ばれ、99.9%の可用性ターゲットは「three nines」と呼ばれます。

ワークロードでは、時間ベースのアプローチを使用する代わりに、成功したリクエストと失敗したリクエストをカウントする方が簡単な場合があります。この場合、次の計算を使用できます。

$$Availability = \frac{Successful\ Responses}{Valid\ Requests}$$

ディザスタリカバリはディザスタイベントに重点を置っていますが、可用性はコンポーネントの障害、ネットワークの問題、ソフトウェアのバグ、負荷の急増など、より小規模な規模のより一般的な中断に重点を置いています。ディザスタリカバリの目的はビジネス継続性ですが、可用性の問題は、ワークロードが意図したビジネス機能を実行するために利用できる時間を最大化することです。どちらも耐障害性戦略の一部である必要があります。

Well-Architected の実現状況の確認

[AWS Well-Architected フレームワーク](#)は、クラウドでシステムを構築するときに行う決定のメリットとデメリットを理解するのに役立ちます。このフレームワークの6つの柱を使用することで、信頼性、セキュリティ、効率、コスト効果および持続可能なシステムを設計し、運用するためのアーキテクチャのベストプラクティスを学習できます。[AWS マネジメントコンソールで無料で利用できる AWS Well-Architected ツール](#)を使用すると、柱ごとに一連の質問に答えることで、これらのベストプラクティスに照らしてワークロードを確認できます。<https://console.aws.amazon.com/wellarchitected>

このホワイトペーパーで説明されている概念は、「[信頼性の柱](#)」ホワイトペーパーに含まれるベストプラクティス、特に<https://docs.aws.amazon.com/wellarchitected/latest/framework/a-failure-management.html>「ディザスタリカバリ (DR) をどのように計画するか」という質問に詳しく説明されています。このホワイトペーパーのプラクティスを実装したら、AWS Well-Architected ツールを使用してワークロードを確認 (または再確認) してください。

回復力に関する責任共有モデル

レジリエンシーは、AWS お客様とお客様との間の責任共有です。ディザスタリカバリと可用性がこの共有モデルの下でどのように機能するかを理解することが重要です。

AWS 責任「クラウドの回復性」

AWS は、AWS クラウドで提供されるすべてのサービスを実行するインフラストラクチャの回復性に責任を負います。このインフラストラクチャは、AWS クラウドサービスを実行するハードウェア、ソフトウェア、ネットワーク、および施設で構成されます。AWS は、これらの AWS クラウドサービスを利用可能にするために商業上合理的な努力を払い、サービスの可用性が [AWS サービスレベルアグリーメント \(SLAs\)](#) を満たすか超えるようにします。

[AWS グローバルクラウドインフラストラクチャ](#)は、お客様が回復力の高いワークロードアーキテクチャを構築できるように設計されています。各 AWS リージョンは完全に分離され、複数の [アベイラビリティゾーン](#) で構成されます。[アベイラビリティゾーン](#) は、インフラストラクチャの物理的に分離されたパーティションです。アベイラビリティゾーンは、ワークロードの回復力に影響を及ぼす可能性のある障害を分離し、リージョン内のその他のゾーンへの影響を回避します。ただし、同時に、AWS リージョンのすべてのゾーンは、ゾーン間で高スループットで低レイテンシーのネットワークを提供する完全冗長の専用メトロファイバーを介して、高帯域幅で低レイテンシーのネットワークで相互接続されます。ゾーン間のすべてのトラフィックは暗号化されています。ゾーン間の同期レプリケーションを実行するために、十分なネットワークパフォーマンスが提供されます。アプリケーションを AZ 間でパーティショニングすると、企業は、停電、落雷、竜巻、台風などの問題からよりよく隔離され、保護されます。

お客様の責任「クラウドにおける回復力」

お客様の責任は、選択した AWS クラウドサービスによって決まります。選択したサービスにより、お客様が回復力に関する責任の一環として実行する必要がある、設定作業の量が決まります。例えば、Amazon Elastic Compute Cloud (Amazon EC2) のようなサービスでは、必要となる回復力の設定と管理をすべてお客様が実行する必要があります。Amazon EC2 インスタンスをデプロイするお客様は、[複数のロケーション \(AWS アベイラビリティゾーンなど\) に EC2 インスタンスをデプロイ](#)し、Amazon EC2 Auto Scaling などのサービスを使用して [自己修復を実装](#)し、インスタンスにインストールされたアプリケーションの [回復力のあるワークロードアーキテクチャのベストプラクティス](#)を使用する責任があります。Amazon S3 や Amazon DynamoDB などのマネージドサービスの場合、AWS はインフラストラクチャレイヤー、オペレーティングシステム、プラットフォームを運用

し、お客様はエンドポイントにアクセスしてデータを保存および取得します。お客様は、バックアップ、バージョンニング、レプリケーション戦略など、データの回復力を管理する責任があります。

AWS リージョン内の複数のアベイラビリティゾーンにワークロードをデプロイすることは、問題を1つのアベイラビリティゾーンに分離してワークロードを保護するように設計された高可用性戦略の一部であり、他のアベイラビリティゾーンの冗長性を使用してリクエストを処理し続けます。マルチ AZ アーキテクチャは、ワークロードをより適切に分離し、停電、落雷、竜巻、地震などの問題から保護するように設計された DR 戦略の一環でもあります。DR 戦略では、複数の AWS リージョンを使用する場合があります。たとえば、アクティブ/パッシブ設定では、アクティブなリージョンがリクエストを処理できなくなった場合、ワークロードのサービスはアクティブなリージョンから DR リージョンにフェイルオーバーされます。

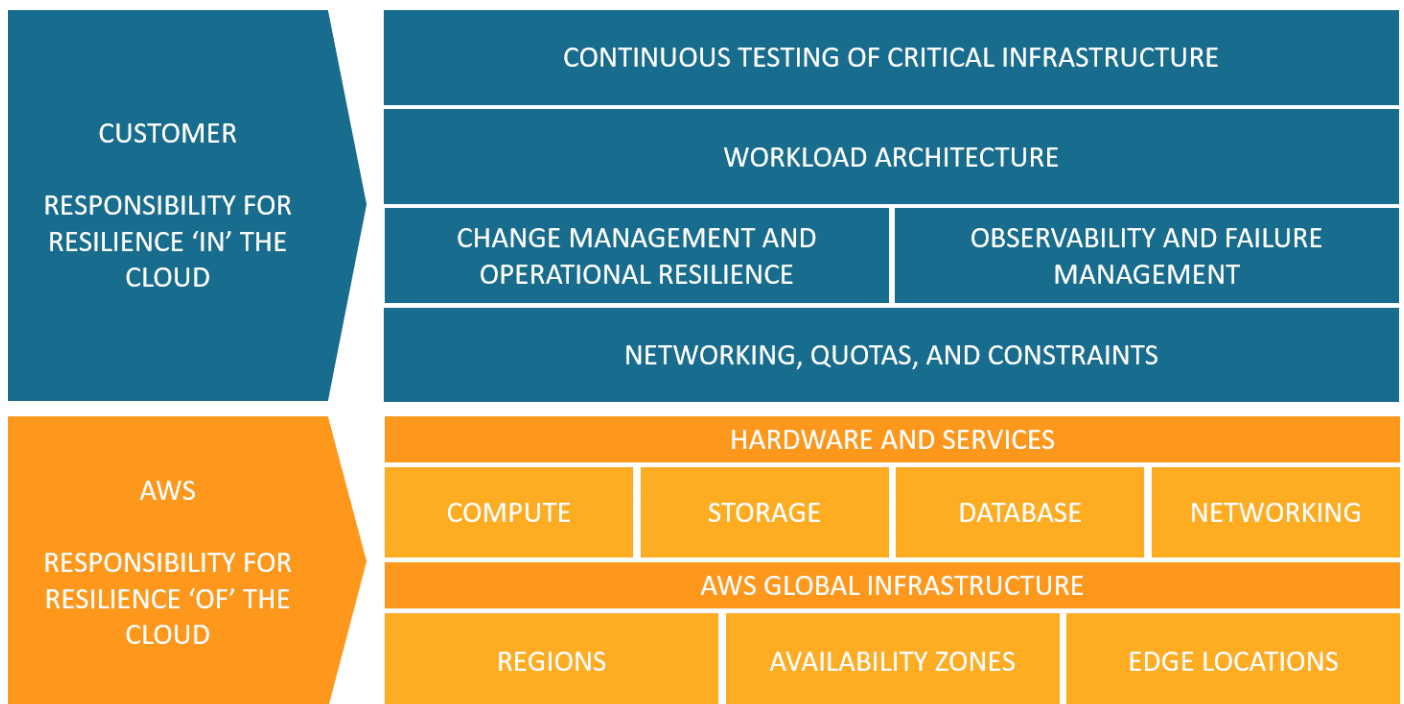


図 2 - 回復力は AWS とお客様の責任を共有します

災害とは

ディザスタリカバ리를計画するときは、次の3つの主なディザスタカテゴリの計画を評価します。

- 地震や洪水などの自然災害
- 停電やネットワーク接続などの技術的な障害
- 不注意による設定ミス、不正な/外部関係者によるアクセスや変更などの人間の行動

これらの潜在的な災害はそれぞれ、地域、地域、国全体、大陸、またはグローバルの地理的な影響も及ぼします。ディザスタリカバリ戦略を検討するときは、ディザスタの性質と地理的影響の両方が重要です。例えば、複数のアベイラビリティーゾーンに影響を与えないため、マルチ AZ 戦略を採用することで、データセンターの停止を引き起こすローカルフラッド問題を軽減できます。ただし、本番稼働用データに対する攻撃では、別の AWS リージョンのバックアップデータにフェイルオーバーするディザスタリカバリ戦略を呼び出す必要があります。

高可用性はディザスタリカバリではありません

可用性とディザスタリカバリはどちらも、障害のモニタリング、複数の場所にデプロイする、自動フェイルオーバーなど、同じベストプラクティスの一部に依存しています。ただし、可用性はワークロードのコンポーネントに重点を置いているが、ディザスタリカバリはワークロード全体の個別のコピーに重点を置いています。ディザスタリカバリにはアベイラビリティとは異なる目的があり、災害に該当する大規模なイベントの後の復旧時間を測定します。可用性の高いアーキテクチャにより、可用性に影響するイベントが発生した場合に顧客のニーズを満たすことができるため、まずワークロードが可用性の目標を満たしていることを確認する必要があります。ディザスタリカバリ戦略には、必要に応じてワークロード全体をフェイルオーバーできるように、個別のシステムを複数の場所にデプロイすることに焦点を当てた、可用性のためのアプローチとは異なるアプローチが必要です。

ディザスタリカバリ計画では、ワークロードの可用性を考慮する必要があります。これは、実行するアプローチに影響するためです。1つのアベイラビリティゾーンの1つのAmazon EC2 インスタンスで実行されるワークロードには、高可用性がありません。ローカルフラッドの問題がそのアベイラビリティゾーンに影響を与える場合、このシナリオではDR目標を達成するために別のAZへのフェイルオーバーが必要です。このシナリオを高可用性のワークロードデプロイ[マルチサイトアクティブ/アクティブ](#)と比較します。ワークロードは複数のアクティブなリージョンにデプロイされ、すべてのリージョンが本番トラフィックを処理します。この場合、万一大規模な災害によりリージョンが使用不能になった場合でも、DR戦略はすべてのトラフィックを残りのリージョンにルーティングすることで実現されます。

データへのアプローチ方法は、可用性とディザスタリカバリでも異なります。高可用性を実現するために、別のサイトに継続的にレプリケートするストレージソリューションを検討してください(マルチサイト、アクティブ/アクティブワークロードなど)。プライマリストレージデバイスでファイルを削除または破損した場合、それらの破壊的な変更はセカンダリストレージデバイスにレプリケートできます。このシナリオでは、高可用性にもかかわらず、データの削除や破損が発生した場合にフェイルオーバーする機能が侵害されます。代わりに、DR戦略の一部としてpoint-in-timeバックアップも必要です。

事業継続計画 (BCP)

ディザスタリカバリ計画は、組織の事業継続計画 (BCP) のサブセットである必要があり、スタンドアロンドキュメントであってはなりません。ワークロード以外のビジネスの要素に対する災害の影響のためにワークロードのビジネス目標を達成できない場合、ワークロードを復元するための積極的なディザスタリカバリ目標を維持する意味はありません。例えば、地震により、eCommerce アプリケーションで購入した製品の輸送が妨げられる可能性があります。効果的な DR によってワークロードが機能し続ける場合でも、BCP は輸送ニーズに対応する必要があります。DR 戦略は、ビジネス要件、優先順位、コンテキストに基づいている必要があります。

ビジネスへの影響分析とリスク評価

ビジネスへの影響分析では、ワークロードの中断によるビジネスへの影響を定量化する必要があります。ワークロードを使用できない内部および外部の顧客への影響と、がビジネスに与える影響を特定する必要があります。この分析は、ワークロードを使用可能にする必要がある速度と、許容できるデータ損失の量を決定するのに役立ちます。ただし、復旧目標は単独で行うべきではないことに注意してください。中断の可能性と復旧コストは、ワークロードにディザスタリカバリを提供することのビジネス価値を知らせるのに役立つ重要な要素です。

ビジネスへの影響は時間に依存する場合があります。これをディザスタリカバリ計画に組み込むことを検討してください。たとえば、給与システムの中断は、すべての人が支払いを受ける直前にビジネスに非常に大きな影響を与える可能性があります。すべての人が既に支払いを受けた直後には影響が少ない可能性があります。

災害の種類と地理的影響のリスク評価とワークロードの技術的実装の概要によって、災害の種類ごとに中断が発生する確率が決まります。

非常に重要なワークロードでは、ビジネスへの影響を最小限に抑えるために、データレプリケーションと継続的なバックアップを備えたインフラストラクチャを複数のリージョンにデプロイすることを検討してください。重要度の低いワークロードの場合、有効な戦略は、ディザスタリカバリをまったく実施しないことです。また、一部の災害シナリオでは、災害が発生する可能性が低いことに基いて、情報に基づいた決定としてディザスタリカバリ戦略を設定しないことも有効です。AWS リージョン内のアベイラビリティゾーンは、すでに意味のある距離で設計されており、ロケーションの慎重な計画を立てています。そのため、最も一般的な災害は、1つのゾーンにのみ影響し、他のゾーンには影響しません。したがって、AWS リージョン内のマルチ AZ アーキテクチャは、すでにリスク軽減ニーズの多くを満たしている可能性があります。

ディザスタリカバリオプションのコストは、ビジネスへの影響とリスクを考慮して、ディザスタリカバリ戦略が適切なレベルのビジネス価値を提供するように評価する必要があります。

この情報はすべて、さまざまな災害シナリオの脅威、リスク、影響、コスト、および関連する復旧オプションを文書化することができます。この情報は、各ワークロードの復旧目標を決定するために使用されます。

復旧目標 (RTO と RPO)

ディザスタリカバリ (DR) 戦略を作成する場合、組織は最も一般的に目標復旧時間 (RTO) と目標復旧時点 (RPO) を計画します。

How much data can you afford to recreate or lose?

**How quickly must you recover?
What is the cost of downtime?**

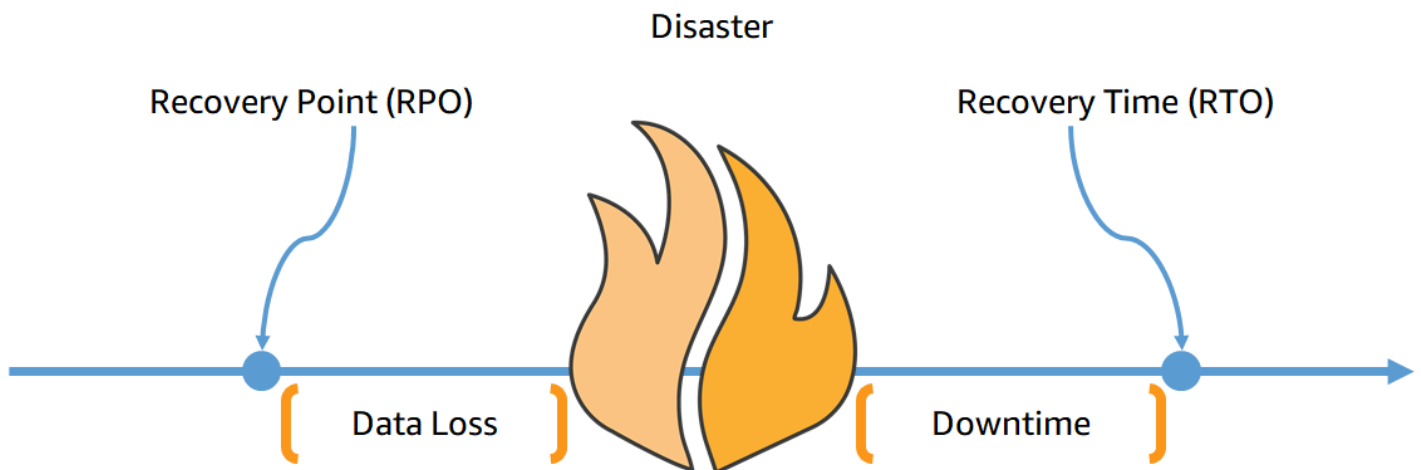


図 3 - 復旧目標

目標復旧時間 (RTO) は、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です。この目的は、サービスが利用できず、組織によって定義されている場合に許容される時間枠と見なされるものを決定します。

このホワイトペーパーでは、バックアップと復元、パイロットライト、ウォームスタンバイ、マルチサイトアクティブ/アクティブの 4 つの DR 戦略について説明しています ([「クラウドのディザスタリカバリオプション」](#)を参照)。次の図では、ビジネスは最大許容 RTO と、サービス復元戦略に支出できる上限を決定しています。ビジネス目標を考慮すると、DR 戦略のパイロットライトまたはウォームスタンバイは、RTO とコストの両方の基準を満たします。

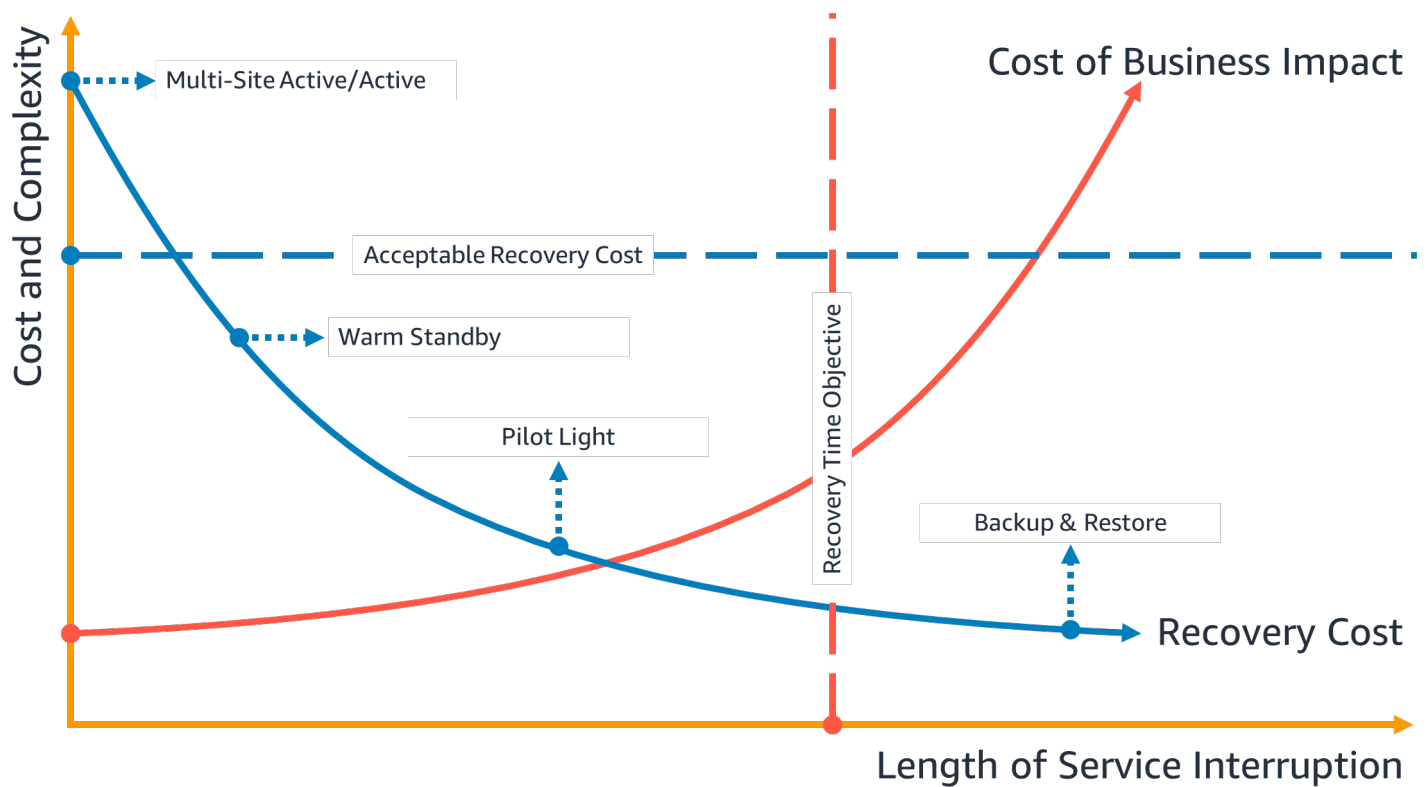


図 4 - 目標復旧時間

目標復旧時点 (RPO) は、最後のデータ復旧時点からの最大許容時間です。この目的は、最後の復旧時点からサービスの中断までの間に許容可能なデータ損失と見なされるものを決定し、組織が定義します。

次の図では、ビジネスが最大許容 RPO と、データ復旧戦略に支出できる上限を決定しています。4 つの DR 戦略のうち、パイロットライトまたはウォームスタンバイ DR 戦略は、RPO とコストの両方の基準を満たしています。

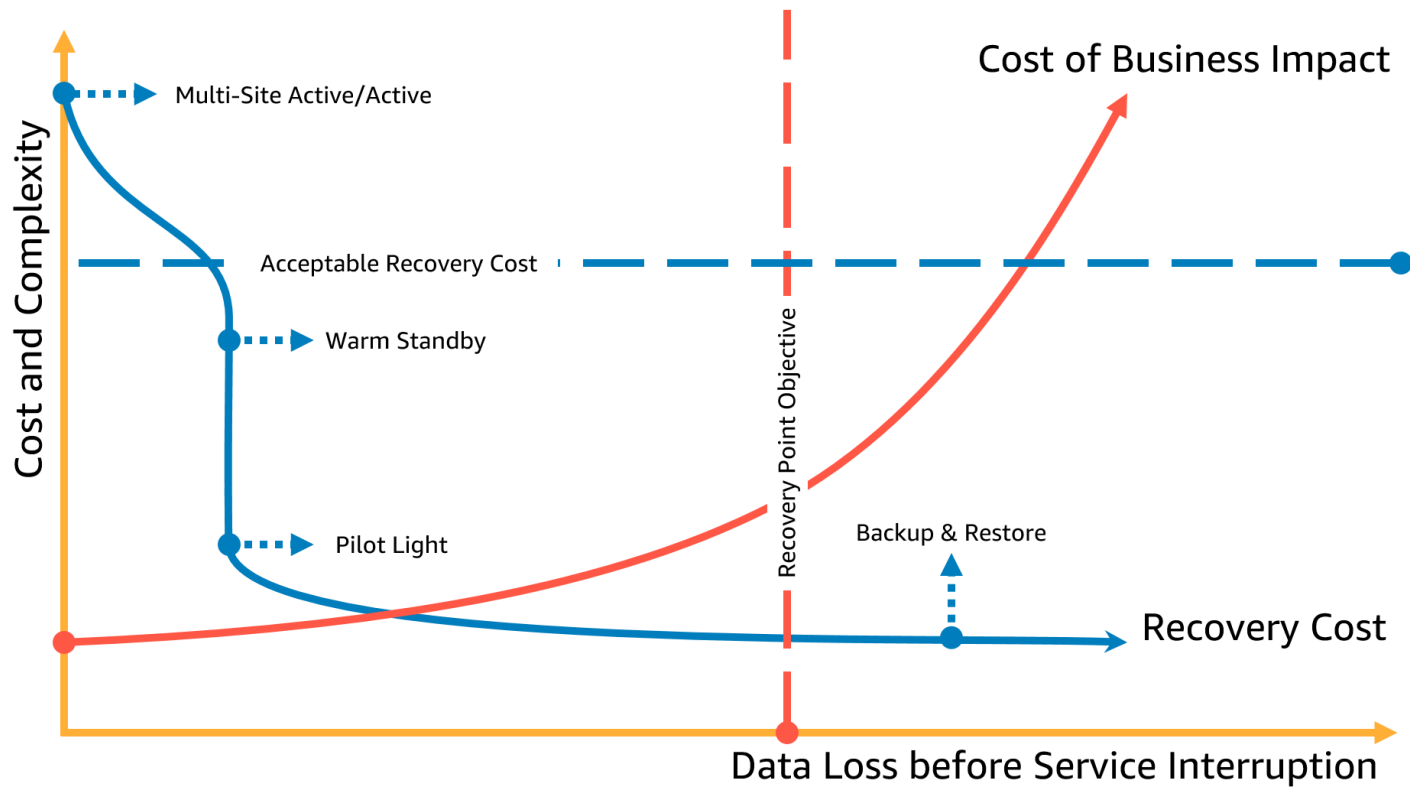


図 5 - 目標復旧時点

Note

復旧戦略のコストが障害や損失のコストよりも高い場合は、規制要件などのセカンダリドライバーがない限り、復旧オプションを設定しないでください。この評価を行うときは、さまざまなコストの復旧戦略を検討してください。

クラウド内の災害対策は異なる

災害対策戦略は、技術的イノベーションとともに進化します。オンプレミスのディザスタリカバリプランには、テープの物理的な転送や別のサイトへのデータのレプリケーションが含まれる場合があります。組織は、AWS での DR 目標を達成するために、以前のディザスタリカバリ戦略のビジネスへの影響、リスク、コストを再評価する必要があります。AWS クラウドのディザスタリカバリには、従来の環境よりも次のような利点があります。

- 複雑さを軽減して災害から迅速に復旧する
- シンプルで反復可能なテストにより、より簡単かつ頻繁にテストできます。
- 管理オーバーヘッドを減らすことで運用上の負担を軽減
- エラーの可能性を減らし、復旧時間を改善する機会

AWS では、物理バックアップデータセンターの固定資本コストをクラウド内の適切な環境の可変運用コストと交換できるため、コストを大幅に削減できます。

多くの組織では、オンプレミスのディザスタリカバリは、データセンター内のワークロードやワークロードが中断されるリスクと、バックアップまたはレプリケートされたデータをセカンダリデータセンターに復旧するリスクに基づいていました。組織がワークロードを AWS にデプロイする場合、適切に設計されたワークロードを実装し、AWS グローバルクラウドインフラストラクチャの設計に依存して、このような中断の影響を軽減できます。クラウドで信頼性、安全性、効率、費用対効果の高いワークロードを設計および運用するためのアーキテクチャのベストプラクティスの詳細については、[AWS Well-Architected Framework - Reliability Pillar ホワイトペーパー](#)を参照してください。[AWS Well-Architected Tool](#) を使用してワークロードを定期的に確認し、Well-Architected フレームワークのベストプラクティスとガイダンスに従っていることを確認します。このツールは、無料で利用できます[AWS マネジメントコンソール](#)。

ワークロードが AWS 上にある場合、データセンターの接続 (データセンターへのアクセス機能を除く)、電源、冷暖房、火災抑制、ハードウェアについて心配する必要はありません。これらはすべてお客様に代わって管理され、複数の障害分離されたアベイラビリティゾーン (それぞれが 1 つ以上の個別のデータセンターで構成されます) にアクセスできます。

単一の AWS リージョン

1 つの物理データセンターの中断または損失に基づく災害イベントの場合、高可用性ワークロードを 1 つの AWS リージョン内の複数のアベイラビリティゾーンに実装することで、自然災害や技術的

災害を軽減できます。この単一リージョン内のデータの継続的なバックアップにより、データ損失につながる可能性のあるエラーや不正なアクティビティなど、人的脅威のリスクを減らすことができます。各 AWS リージョンは複数のアベイラビリティゾーンで構成され、それぞれが他のゾーンの障害から分離されています。各アベイラビリティゾーンは、1 つ以上の個別の物理データセンターで構成されます。影響のある問題をより適切に分離し、高可用性を実現するには、同じリージョン内の複数のゾーンにワークロードを分割できます。アベイラビリティゾーンは、物理的な冗長性を考慮して設計されており、回復力を備えているため、停電、インターネットのダウンタイム、洪水、その他の自然災害が発生した場合でも、パフォーマンスを中断することなく実行できます。これを行う方法については、「AWS [グローバルクラウドインフラストラクチャ](#)」を参照してください。

単一の AWS リージョン内の複数のアベイラビリティゾーンにデプロイすることで、ワークロードを単一の (または複数の) データセンターの障害から保護できます。単一リージョンのデプロイをより確実にするために、データと設定 (インフラストラクチャ定義を含む) を別のリージョンにバックアップできます。この戦略により、ディザスタリカバリプランの範囲が縮小され、データのバックアップと復元のみが含まれます。別の AWS リージョンにバックアップしてマルチリージョンの耐障害性を活用することは、次のセクションで説明する他のマルチリージョンオプションと比較して簡単で安価です。例えば、[Amazon Simple Storage Service \(Amazon S3\)](#) にバックアップすると、データをすぐに取得できます。ただし、データの一部に対する DR 戦略で取得時間 (数分から数時間) の要件がより緩い場合、[Amazon Glacier](#) または [Amazon Glacier Deep Archive](#) を使用すると、バックアップおよびリカバリ戦略のコストが大幅に削減されます。

一部のワークロードには、規制データレジデンシー要件がある場合があります。これは、現在 AWS リージョンが 1 つしかないローカルのワークロードに適用される場合、上記のように高可用性を実現するマルチ AZ ワークロードを設計することに加えて、そのリージョン内の AZs を個別の場所として使用することもできます。これは、そのリージョン内のワークロードに適用されるデータレジデンシー要件に対処するのに役立ちます。以下のセクションで説明する DR 戦略では、複数の AWS リージョンを使用しますが、リージョンの代わりにアベイラビリティゾーンを使用して実装することもできます。

複数の AWS リージョン

複数のデータセンターを互いにかなり離れて失うリスクを含む災害が発生した場合は、AWS 内のリージョン全体に影響を与える自然災害や技術的災害を軽減するためのディザスタリカバリオプションを検討する必要があります。以下のセクションで説明するすべてのオプションは、このような災害から保護するためにマルチリージョンアーキテクチャとして実装できます。

クラウド内での災害対策オプション

AWS 内で利用できるディザスタリカバリ戦略は、低コストと複雑さの低いバックアップから、複数のアクティブなリージョンを使用したより複雑な戦略まで、4つのアプローチに広く分類できます。アクティブ/パッシブ戦略では、アクティブなサイト (AWS リージョンなど) を使用してワークロードをホストし、トラフィックを処理します。パッシブサイト (別の AWS リージョンなど) が復旧に使用されます。パッシブサイトは、フェイルオーバーイベントがトリガーされるまでトラフィックをアクティブに処理しません。

ディザスタリカバリ戦略を定期的に評価してテストし、必要に応じて確実に呼び出すことが重要です。[AWS Resilience Hub](#) を使用して、RTO と RPO の目標を達成する可能性が高いかどうかなど、AWS ワークロードの耐障害性を継続的に検証および追跡します。

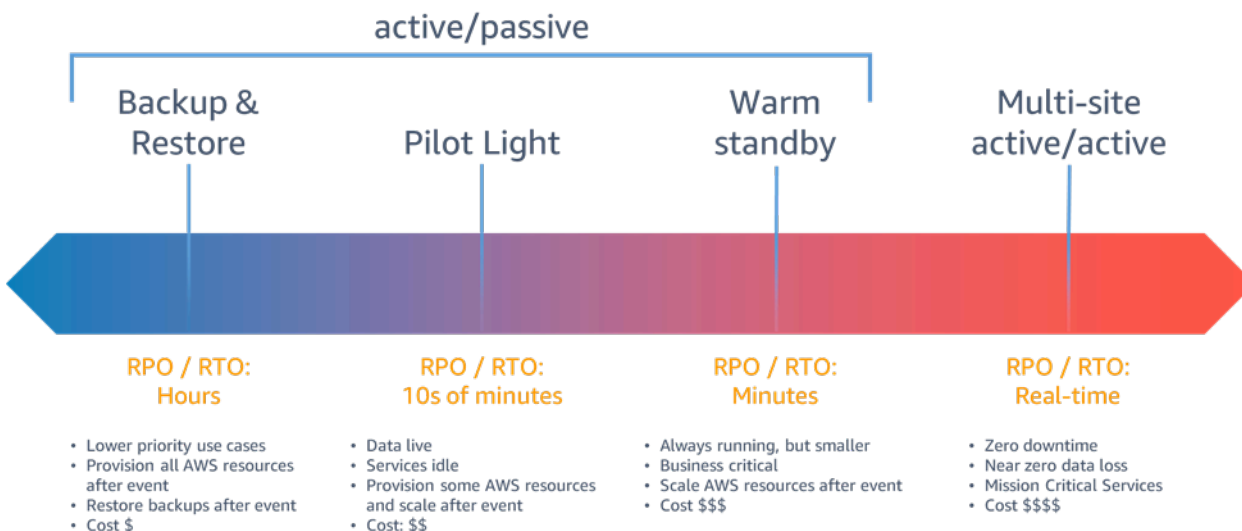


図 6 - ディザスタリカバリ戦略

適切に設計され、可用性の高いワークロードのための1つの物理データセンターの中断または損失に基づく災害イベントの場合、ディザスタリカバリへのバックアップと復元のアプローチのみが必要になる場合があります。災害の定義が、物理的なデータセンターの中断や損失を超えてリージョンの定義である場合、またはそれを必要とする規制要件の対象である場合は、パイロットライト、ウォームスタンバイ、またはマルチサイトアクティブ/アクティブを検討する必要があります。

戦略とそれを実装する AWS リソースを選択するときは、AWS 内では通常、サービスをデータプレーンとコントロールプレーンに分割することに注意してください。コントロールプレーンで環境を設定しつつ、データプレーンではリアルタイムのサービスを提供します。回復性を最大化するには、フェイルオーバーオペレーションの一部としてデータプレーンオペレーションのみを使用する必要があります。

あります。これは、データプレーンには通常、コントロールプレーンよりも高い可用性の設計目標があるためです。

バックアップと復元

バックアップと復元は、データの損失や破損を軽減するための適切なアプローチです。このアプローチは、データを他の AWS リージョンにレプリケートすることでリージョンの災害を軽減したり、単一のアベイラビリティゾーンにデプロイされたワークロードの冗長性の欠如を軽減したりするためにも使用できます。データに加えて、復旧リージョンにインフラストラクチャ、設定、アプリケーションコードを再デプロイする必要があります。エラーなしでインフラストラクチャを迅速に再デプロイできるようにするには、[AWS CloudFormation](#)やなどのサービスを使用して、Infrastructure as Code (IaC) を使用して常にデプロイする必要があります。[AWS Cloud Development Kit \(AWS CDK\)](#)。IaC を使用しない場合、復旧リージョンでワークロードを復元するのが複雑になり、復旧時間が長くなり、RTO を超える可能性があります。ユーザーデータに加えて、Amazon EC2 インスタンスの作成に使用する [Amazon マシンイメージ \(AMIs\)](#) を含むコードと設定も必ずバックアップしてください。を使用して[AWS CodePipeline](#)、アプリケーションコードと設定の再デプロイを自動化できます。

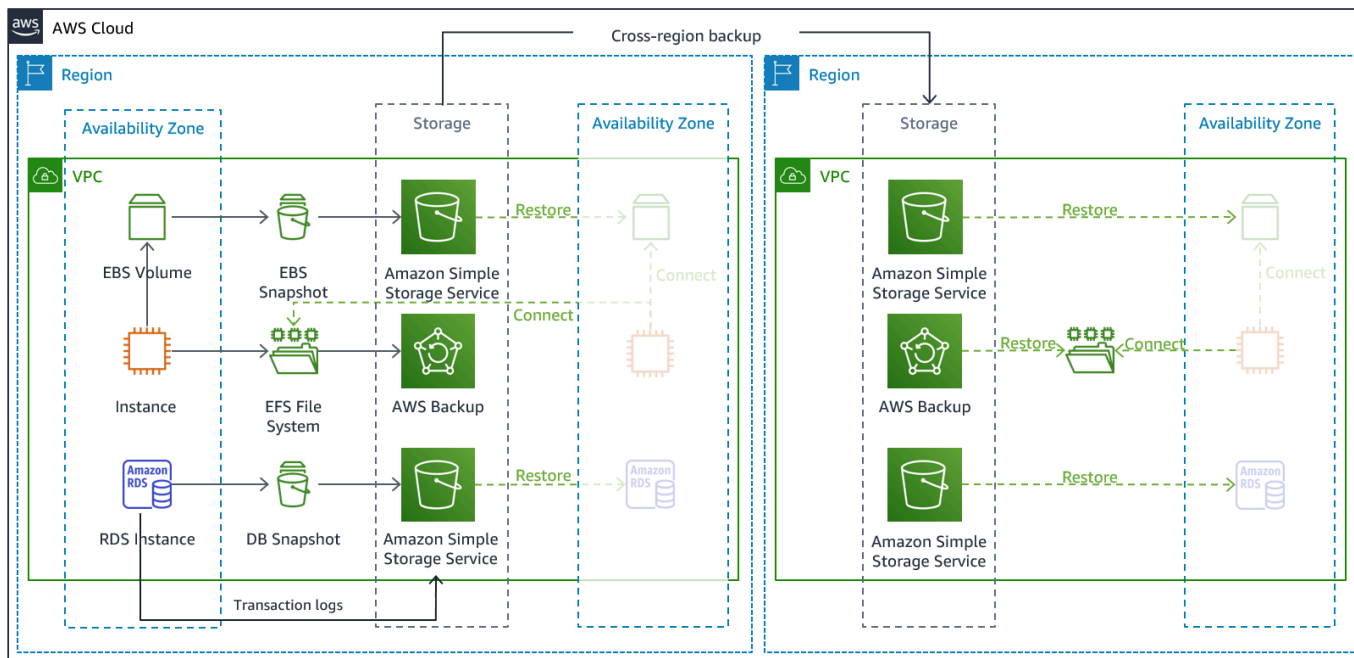


図 7 - バックアップと復元のアーキテクチャ

AWS サービス

ワークロードデータには、定期的に行われるバックアップ戦略または継続的なバックアップ戦略が必要です。バックアップを実行する頻度によって、達成可能な復旧ポイントが決まります (RPO に合わせて調整する必要があります)。バックアップは、バックアップが作成された時点で復元する方法も提供する必要があります。point-in-timeリカバリによるバックアップは、以下のサービスとリソースを通じて利用できます。

- [Amazon Elastic Block Store \(Amazon EBS\) スナップショット](#)
- [Amazon DynamoDB バックアップ](#)
- [Amazon RDS スナップショット](#)
- [Amazon Aurora DB スナップショット](#)
- [Amazon EFS バックアップ](#) (使用時 AWS Backup)
- [Amazon Redshift スナップショット](#)
- [Amazon Neptune スナップショット](#)
- [Amazon DocumentDB](#)
- [Amazon FSx for Windows File Server](#)、[Amazon FSx for Lustre](#)、[Amazon FSx for NetApp ONTAP](#)、[Amazon FSx for OpenZFS](#)

Amazon Simple Storage Service (Amazon S3) では、[Amazon S3 クロスリージョンレプリケーション \(CRR\)](#) を使用して、保存されたオブジェクトのバージョンングを提供しながら、DR リージョンの S3 バケットにオブジェクトを継続的に非同期的にコピーし、復元ポイントを選択できます。データの継続的なレプリケーションには、データのバックアップにかかる最短時間 (ゼロに近い時間) という利点がありますが、データの破損や悪意のある攻撃 (不正なデータ削除など) などの災害イベントやpoint-in-timeバックアップから保護できない場合があります。継続的レプリケーションについては、「AWS [Services for Pilot Light](#)」セクションを参照してください。

[AWS Backup](#) は、以下のサービスとリソースの AWS バックアップ機能を設定、スケジュール、モニタリングするための一元的な場所を提供します。

- [Amazon Elastic Block Store \(Amazon EBS\) ボリューム](#)
- [Amazon EC2 インスタンス](#)
- [Amazon Relational Database Service \(Amazon RDS\) データベース](#) ([Amazon Aurora](#) データベースを含む)
- [Amazon DynamoDB](#) テーブル

- [Amazon Elastic File System \(Amazon EFS\)](#) ファイルシステム
- [AWS Storage Gateway](#) ポリユーム
- [Amazon FSx for Windows File Server](#)、[Amazon FSx for Lustre](#)、[Amazon FSx for NetApp ONTAP](#)、[Amazon FSx for OpenZFS](#)

AWS Backup は、ディザスタリカバリリージョンなど、リージョン間でのバックアップのコピーをサポートしています。

Amazon S3 データの追加のディザスタリカバリ戦略として、[S3 オブジェクトのバージョニング](#)を有効にします。オブジェクトバージョニングは、アクションの前に元のバージョンを保持することで、削除または変更アクションの結果から S3 内のデータを保護します。オブジェクトのバージョニングは、ヒューマンエラータイプの災害の軽減に役立ちます。S3 レプリケーションを使用して DR リージョンにデータをバックアップしている場合、デフォルトでは、レプリケート元バケットでオブジェクトが削除されると、[Amazon S3 はレプリケート元バケットにのみ削除マーカを追加します](#)。このアプローチは、DR リージョン内のデータをソースリージョン内の悪意のある削除から保護します。

データに加えて、ワークロードを再デプロイし、目標復旧時間 (RTO) を満たすために必要な設定とインフラストラクチャもバックアップする必要があります。[AWS CloudFormation](#)は、Infrastructure as Code (IaC) を提供し、ワークロード内のすべての AWS リソースを定義して、複数の AWS アカウントと AWS リージョンに確実にデプロイおよび再デプロイできるようにします。ワークロードで使用される Amazon EC2 インスタンスを Amazon マシンイメージ (AMIs)としてバックアップできます。AMI は、インスタンスのルートボリュームと、インスタンスにアタッチされたその他の EBS ボリュームのスナップショットから作成されます。この AMI を使用して、EC2 インスタンスの復元されたバージョンを起動できます。[AMI は、リージョン内またはリージョン間でコピーできます](#)。または、[AWS Backup](#) を使用して、アカウント間および他の AWS リージョンにバックアップをコピーすることもできます。クロスアカウントバックアップ機能は、インサイダーの脅威やアカウントの侵害を含む災害イベントからの保護に役立ちます。AWS Backup または、インスタンスの個々の EBS ボリュームに加えて、EC2 バックアップ用の機能を追加し、インスタンスタイプ、設定された仮想プライベートクラウド (VPC)、セキュリティグループ、[IAM ロール](#)、モニタリング設定、タグなどのメタデータ AWS Backup も保存および追跡します。ただし、この追加メタデータは、EC2 バックアップを同じ AWS リージョンに復元する場合にのみ使用されます。

バックアップとしてディザスタリカバリリージョンに保存されているデータは、フェイルオーバー時に復元する必要があります。は復元機能 AWS Backup を提供しますが、現在、スケジュールされた復元または自動復元は有効にしていません。AWS SDK を使用して APIs を呼び出すことで、DR リージョンへの自動復元を実装できます AWS Backup。これは定期的に繰り返されるジョブとし

で設定することも、バックアップが完了するたびに復元をトリガーすることもできます。次の図は、[Amazon Simple Notification Service \(Amazon SNS\)](#) と [AWS Lambda](#) を使用した自動復元の例を示しています。バックアップからのデータ復元はコントロールプレーンオペレーションであるため、スケジュールされた定期的なデータ復元を実装することをお勧めします。災害発生時にこのオペレーションが利用できなかった場合でも、最近のバックアップから運用可能なデータストアが作成されます。

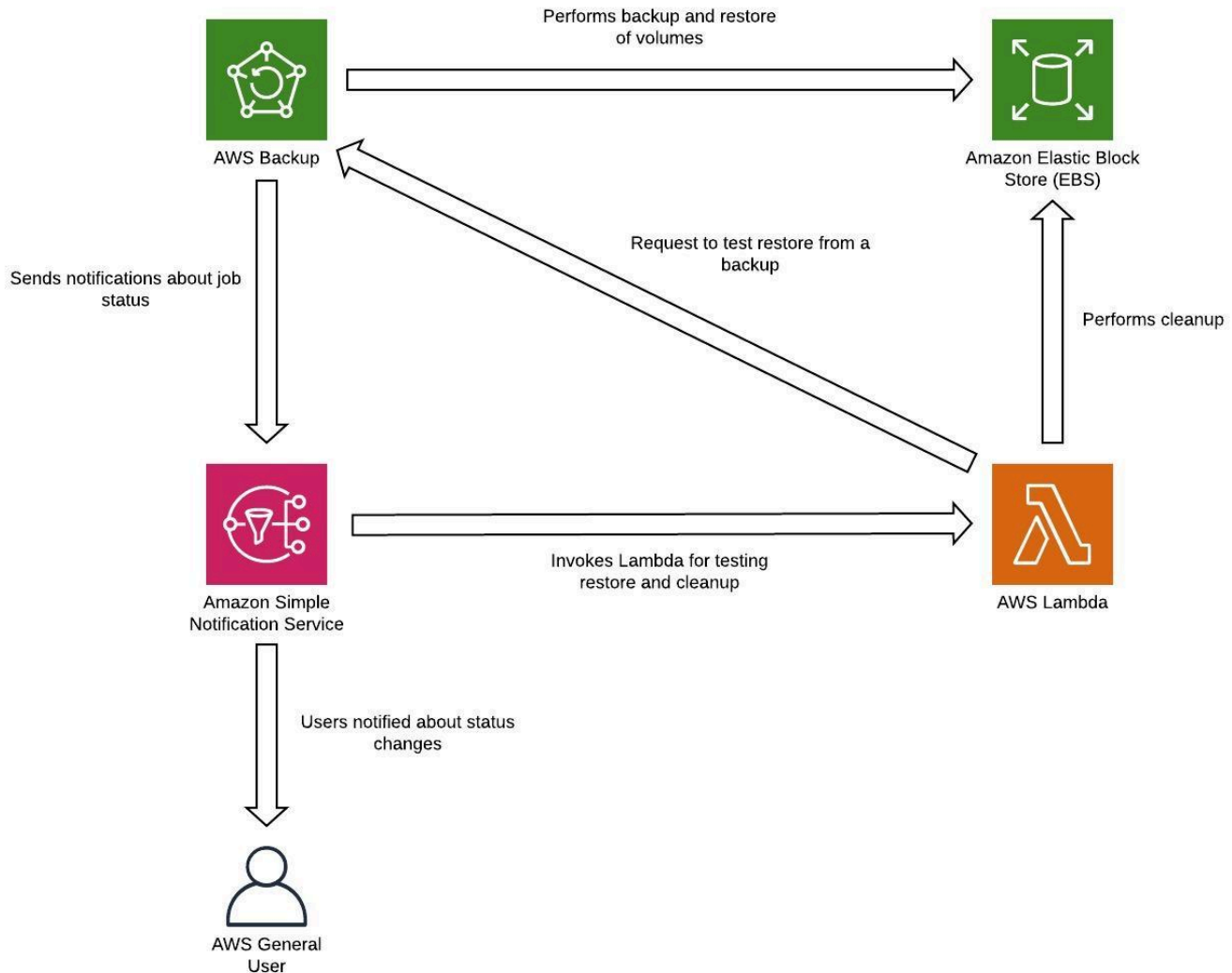


図 8 - バックアップの復元とテスト

Note

バックアップ戦略にはバックアップのテストが含まれていなければなりません。詳細については、[「ディザスタリカバリのテスト」](#) セクションを参照してください。実装の実践的なデ

モンストレーションについては、[AWS Well-Architected Lab: Testing Backup and Restore of Data](#) を参照してください。

パイロットライト

パイロットライトアプローチでは、あるリージョンから別のリージョンにデータをレプリケートし、コアワークロードインフラストラクチャのコピーをプロビジョニングします。データベースやオブジェクトストレージなど、データのレプリケーションとバックアップのサポートに必要なリソースは、常にオンです。アプリケーションサーバーなどの他の要素にはアプリケーションコードと設定がロードされますが、「スイッチオフ」され、テスト中またはディザスタリカバリフェイルオーバーが呼び出されたときにのみ使用されます。クラウドでは、不要なリソースを柔軟にプロビジョニング解除し、必要なときにプロビジョニングできます。「スイッチオフ」のベストプラクティスは、リソースをデプロイせず、必要に応じてデプロイする設定と機能（「スイッチオン」）を作成することです。バックアップと復元のアプローチとは異なり、コアインフラストラクチャは常に利用可能であり、アプリケーションサーバーをオンにしてスケールアウトすることで、フルスケールの本番稼働環境を迅速にプロビジョニングできます。

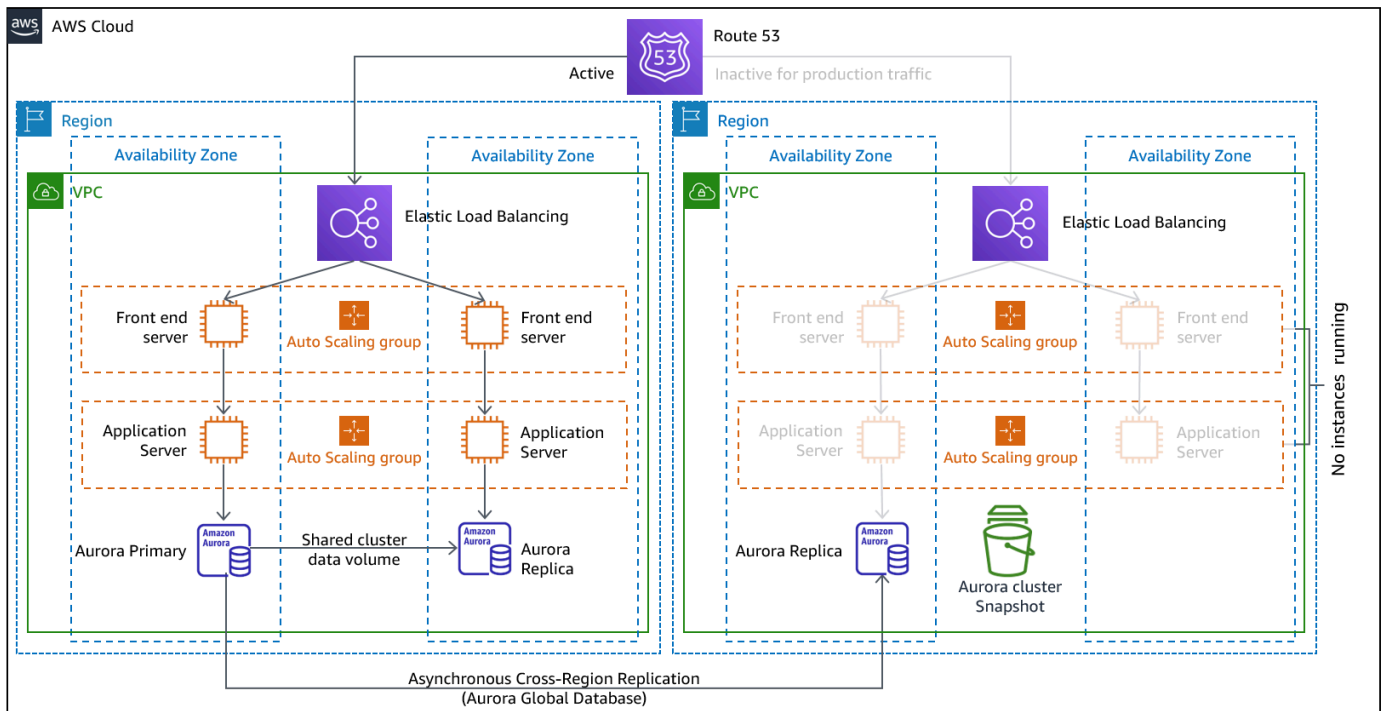


図 9 - パイロットライトアーキテクチャ

パイロットライトアプローチは、アクティブなリソースを最小限に抑えることでディザスタリカバリの継続的なコストを最小限に抑え、コアインフラストラクチャ要件がすべて整っているため、ディザスタ時の復旧を簡素化します。この復旧オプションでは、デプロイアプローチを変更する必要があります。コアインフラストラクチャの変更を各リージョンに行い、ワークロード (設定、コード) の変更を各リージョンに同時にデプロイする必要があります。このステップは、デプロイを自動化し、Infrastructure as Code (IaC) を使用して複数のアカウントとリージョンにインフラストラクチャをデプロイすることで簡素化できます (プライマリリージョンへのフルインフラストラクチャデプロイと、DR リージョンへのスケールダウン/スイッチオフインフラストラクチャデプロイ)。リージョンごとに異なるアカウントを使用して、最高レベルのリソースとセキュリティの分離を提供することをお勧めします (侵害された認証情報がディザスタリカバリプランの一部である場合も同様)。

このアプローチでは、データ災害も軽減する必要があります。連続的なデータレプリケーションは、特定のタイプの災害から保護しますが、戦略に、保存データのバージョンングまたはポイントインタイムリカバリのためのオプションが含まれていない限り、データの破損や破壊からは保護しません。ディザスタリージョンでレプリケートされたデータをバックアップして、同じリージョンに point-in-time バックアップを作成できます。

AWS サービス

Backup [and Restore](#) セクションで説明されている AWS のサービスを使用して point-in-time バックアップを作成することに加えて、パイロットライト戦略では次のサービスも検討してください。

パイロットライトの場合、DR リージョンのライブデータベースとデータストアへの継続的なデータレプリケーションは、低 RPO の最善のアプローチです (前述の point-in-time バックアップに加えて使用する場合)。AWS は、以下のサービスとリソースを使用して、データの継続的、クロスリージョン、非同期データレプリケーションを提供します。

- [Amazon Simple Storage Service \(Amazon S3\) レプリケーション](#)
- [Amazon RDS リードレプリカ](#)
- [Amazon Aurora グローバルデータベース](#)
- [Amazon DynamoDB グローバルテーブル](#)
- [Amazon DocumentDB グローバルクラスター](#)
- [Amazon ElastiCache のグローバルデータストア \(Redis OSS\)](#)

継続的レプリケーションでは、データのバージョンは DR リージョンでほぼすぐに使用できます。実際のレプリケーション時間は、[S3 オブジェクトの S3 Replication Time Control \(S3 RTC\)](#) や [Amazon](#)

[Aurora グローバルデータベースの管理機能などのサービス機能](#)を使用してモニタリングできます。

S3

ディザスタリカバリリージョンから読み取り/書き込みワークロードの実行にフェイルオーバーする場合は、RDS リードレプリカをプライマリインスタンスに昇格させる必要があります。[Aurora 以外の DB インスタンスの場合、プロセス](#)が完了するまでに数分かかり、再起動はプロセスの一部です。RDS によるクロスリージョンレプリケーション (CRR) とフェイルオーバーの場合、[Amazon Aurora グローバルデータベース](#)を使用すると、いくつかの利点があります。グローバルデータベースは、専用インフラストラクチャを使用してデータベースをアプリケーションに完全に提供し、一般的なレイテンシーが 1 秒未満 (AWS リージョン内は 100 ミリ秒未満) でセカンダリリージョンにレプリケートできます。Amazon Aurora グローバルデータベースでは、プライマリリージョンでパフォーマンスの低下や機能停止が発生した場合、いずれかのセカンダリリージョンを昇格させて、リージョンが完全に機能停止した場合でも 1 分以内に読み取り/書き込みの責任を引き受けることができます。すべてのセカンダリクラスターの RPO ラグタイムをモニタリングするように Aurora を設定して、少なくとも 1 つのセカンダリクラスターがターゲット RPO ウィンドウ内にとどまるようにすることもできます。

DR リージョンにデプロイする必要があるリソースが少ないか小さいコアワークロードインフラストラクチャのスケールダウンバージョン。を使用すると AWS CloudFormation、インフラストラクチャを定義し、AWS アカウント間および AWS リージョン間で一貫してデプロイできます。は事前定義された[擬似パラメータ](#) AWS CloudFormation を使用して、デプロイ先の AWS アカウントと AWS リージョンを識別します。したがって、[CloudFormation テンプレートに条件ロジック](#)を実装して、スケールダウンされたバージョンのインフラストラクチャのみを DR リージョンにデプロイできます。EC2 インスタンスのデプロイの場合、Amazon マシンイメージ (AMI) はハードウェア設定やインストール済みソフトウェアなどの情報を提供します。必要な AMIs を作成し、プライマリリージョンとバックアップリージョンの両方にコピーする [Image Builder](#) パイプラインを実装できます。これにより、これらのゴールデン AMIs に、災害が発生した場合にワークロードを新しいリージョンに再デプロイまたはスケールアウトするために必要なものがすべて揃うようになります。Amazon EC2 インスタンスは、スケールダウンされた設定でデプロイされます (プライマリリージョンよりも少ないインスタンス)。本番トラフィックをサポートするようにインフラストラクチャをスケールアウトするには、「[ウォームスタンバイ](#)」セクションの[Amazon EC2 Auto Scaling](#)」を参照してください。

パイロットライトなどのアクティブ/パッシブ設定の場合、すべてのトラフィックは最初にプライマリリージョンに送信され、プライマリリージョンが使用できなくなった場合はディザスタリカバリリージョンに切り替わります。このフェイルオーバー操作は、自動または手動で開始できます。ヘルスチェックまたはアラームに基づいて自動的に開始されるフェイルオーバーは、慎重に使用する必要があります。ここで説明したベストプラクティスを使用しても、復旧時間と復旧ポイントは 0 より大きくなり、可用性とデータの損失が発生します。不要な場合にフェイルオーバーした場合 (誤った

アラーム)、これらの損失が発生します。そのため、多くの場合、手動によるフェイルオーバーの開始が使用されます。この場合でも、フェイルオーバーのステップを自動化できるため、手動開始はボタンを押すようなものです。

AWS サービスを使用する際に考慮すべきトラフィック管理オプションがいくつかあります。

1 つのオプションは、[Amazon Route 53](#) を使用することです。Amazon Route 53 を使用すると、1 つ以上の AWS リージョンの複数の IP エンドポイントを Route 53 ドメイン名に関連付けることができます。次に、そのドメイン名で適切なエンドポイントにトラフィックをルーティングできます。フェイルオーバー時に、トラフィックを復旧エンドポイントに切り替え、プライマリエンドポイントから遠ざける必要があります。Amazon Route 53 ヘルスチェックは、これらのエンドポイントをモニタリングします。これらのヘルスチェックを使用すると、自動的に開始される DNS フェイルオーバーを設定して、トラフィックが正常なエンドポイントにのみ送信されるようにできます。これは、データプレーンで実行される信頼性の高いオペレーションです。手動で開始されたフェイルオーバーを使用してこれを実装するには、[Amazon Application Recovery Controller \(ARC\)](#) を使用できます。ARC では、Route 53 ヘルスチェックを作成できます。実際にはヘルスチェックは行われませんが、完全に制御できるオン/オフスイッチとして機能します。AWS CLI または AWS SDK を使用すると、この高可用性のデータプレーン API を使用してフェイルオーバーをスクリプト化できます。スクリプトは、プライマリリージョンではなく復旧リージョンにトラフィックを送信するように Route 53 に指示するこれらのスイッチ (Route 53 ヘルスチェック) を切り替えます。一部の が使用した手動で開始されたフェイルオーバーのもう 1 つのオプションは、加重ルーティングポリシーを使用し、プライマリリージョンとリカバリリージョンの重みを変更して、すべてのトラフィックがリカバリリージョンに転送されるようにすることです。ただし、これはコントロールプレーンオペレーションであるため、Amazon Application Recovery Controller (ARC) を使用したデータプレーンアプローチほどの回復力がないことに注意してください。

もう 1 つのオプションは、[AWS Global Accelerator](#) を使用することです。AnyCast IP を使用すると、1 つ以上の AWS リージョンの複数のエンドポイントを同じ静的パブリック IP アドレスに関連付けることができます。AWS Global Accelerator その後、はそのアドレスに関連付けられた適切なエンドポイントにトラフィックをルーティングします。[Global Accelerator ヘルスチェック](#) はエンドポイントをモニタリングします。これらのヘルスチェックを使用して、はアプリケーションの状態 AWS Global Accelerator をチェックし、ユーザートラフィックを正常なアプリケーションエンドポイントに自動的にルーティングします。手動で開始されたフェイルオーバーでは、トラフィックダイヤルを使用してトラフィックを受信するエンドポイントを調整できますが、これはコントロールプレーンオペレーションであることに注意してください。Global Accelerator は、広範な AWS エッジネットワークを使用してトラフィックをできるだけ早く AWS ネットワークバックボーンに配置するため、アプリケーションエンドポイントへのレイテンシーが低くなります。Global Accelerator は、DNS システム (Route 53 など) で発生する可能性のあるキャッシュの問題も回避します。

[Amazon CloudFront](#) はオリジンフェイルオーバーを提供します。プライマリエンドポイントへの特定のリクエストが失敗した場合、CloudFront はリクエストをセカンダリエンドポイントにルーティングします。前述のフェイルオーバーオペレーションとは異なり、後続のすべてのリクエストは引き続きプライマリエンドポイントに送信され、フェイルオーバーはリクエストごとに行われます。

AWS Elastic Disaster Recovery

[AWS Elastic Disaster Recovery \(DRS\)](#) は、基盤となるサーバーのブロックレベルのレプリケーション AWS を使用して、サーバーがホストするアプリケーションとサーバーがホストするデータベースを任意のソースからに継続的にレプリケートします。Elastic Disaster Recovery を使用すると、のリージョンを、オンプレミスまたは別のクラウドプロバイダーでホストされているワークロードとその環境のディザスタリカバリターゲット AWS クラウド として使用できます。EC2 で AWS ホストされているアプリケーションとデータベース (RDS ではない) のみで構成される場合、ホストされたワークロードのディザスタリカバリにも使用できます。Elastic Disaster Recovery は、パイロットライト戦略を使用して、ステージングエリアとして使用される [Amazon Virtual Private Cloud \(Amazon VPC\)](#) 内のデータのコピーと「スイッチオフ」リソースを維持します。フェイルオーバーイベントがトリガーされると、ステージングされたリソースを使用して、復旧場所として使用されるターゲット Amazon VPC にフルキャパシティデプロイが自動的に作成されます。

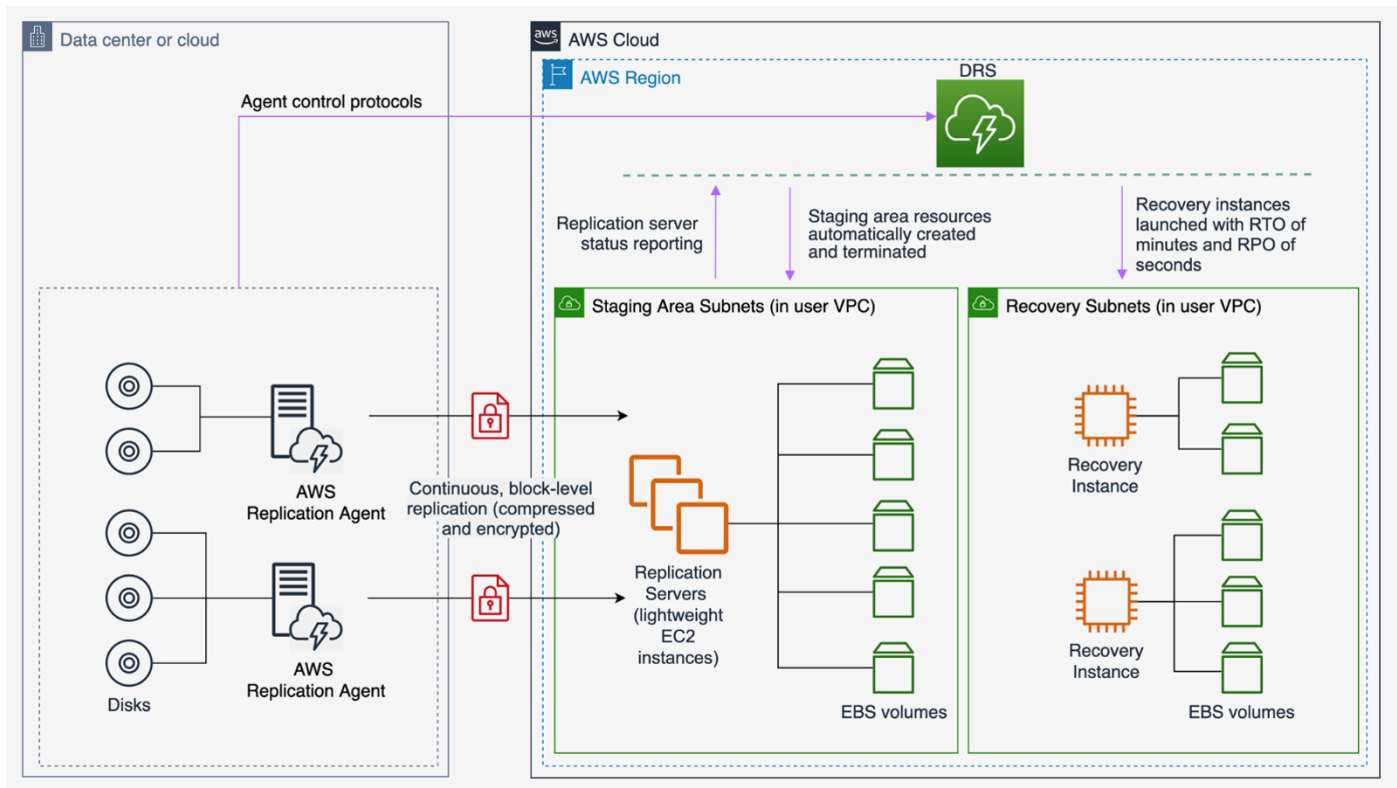


図 10 - AWS Elastic Disaster Recovery アーキテクチャ

ウォームスタンバイ

ウォームスタンバイアプローチでは、本番稼働環境の完全に機能する縮小コピーを別のリージョンに用意します。このアプローチは、パイロットライトの概念を拡張して、ワークロードが別のリージョンに常駐するため、復旧時間が短縮されます。このアプローチにより、テストをより簡単に実行したり、継続的なテストを実装したりして、災害から回復する能力の信頼性を高めることもできます。

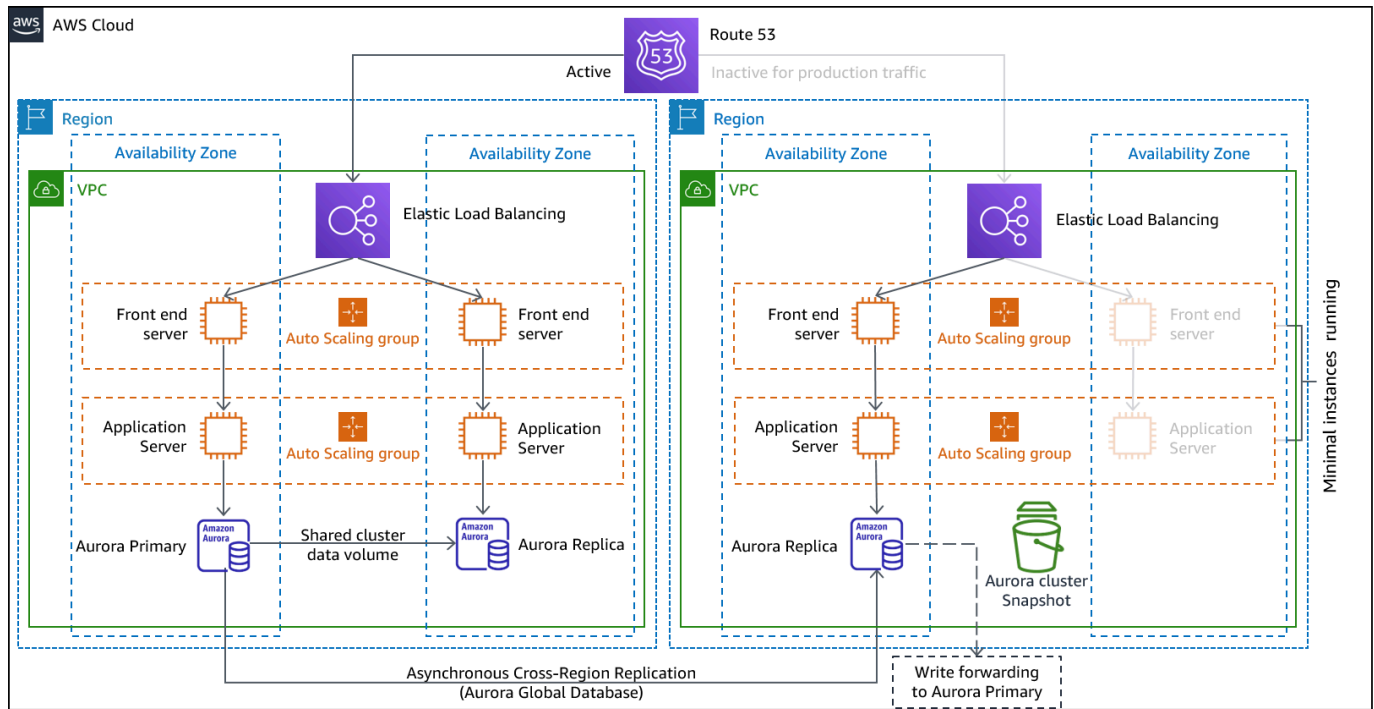


図 11 - ウォームスタンバイアーキテクチャ

注：[パイロットライト](#)と[ウォームスタンバイ](#)の違いを理解するのは難しい場合があります。どちらも、プライマリリージョンアセットのコピーを含む環境を DR リージョンに含めます。違いは、パイロットライトは追加のアクションを最初に実行しない限りリクエストを処理できないのに対し、ウォームスタンバイはトラフィックを (キャパシティーレベルを減らして) すぐに処理できることです。パイロットライトアプローチでは、サーバーを「オン」し、場合によっては追加の (非コア) インフラストラクチャをデプロイしてスケールアップする必要がありますが、ウォームスタンバイではスケールアップのみが必要です (すべてが既にデプロイされ、実行されています)。RTO と RPO のニーズを使用して、これらのアプローチから選択できます。

AWS サービス

[バックアップ](#)、[復元](#)、[パイロットライト](#)の対象となるすべての AWS サービスは、データバックアップ、データレプリケーション、アクティブ/パッシブトラフィックルーティング、EC2 インスタンスを含むインフラストラクチャのデプロイのためにウォームスタンバイでも使用されます。

[Amazon EC2 Auto Scaling](#) は、AWS リージョン内の Amazon EC2 インスタンス、Amazon ECS タスク、Amazon DynamoDB スループット、Amazon Aurora レプリカなどのリソースをスケールするために使用されます。[Amazon EC2 Auto Scaling](#) は、AWS リージョン内のアベイラビリティーゾーン間で EC2 インスタンスのデプロイをスケーリングし、そのリージョン内の回復性を提供します。Auto Scaling を使用して、パイロットライトまたはウォームスタンバイ戦略の一環として、DR リージョンをフルプロダクション機能にスケールアウトします。たとえば、EC2 の場合は、Auto Scaling グループの希望する容量設定を増やします。この設定は、を使用して手動で調整することも AWS マネジメントコンソール、AWS SDK を使用して自動的に調整することも、新しい希望する容量値を使用して AWS CloudFormation テンプレートを再デプロイすることもできます。AWS CloudFormation パラメータを使用すると、CloudFormation テンプレートの再デプロイが簡単になります。DR リージョンの[サービスクォータ](#)が、本番環境の容量へのスケーリングを制限しないように十分に高く設定されていることを確認します。

Auto Scaling はコントロールプレーンのアクティビティであるため、それに依存すると、全体的な復旧戦略の耐障害性が低下します。これはトレードオフです。リカバリリージョンがデプロイされた本番稼働負荷全体を処理できるように、十分な容量をプロビジョニングすることを選択できます。この静的に安定した設定は、ホットスタンバイと呼ばれます(次のセクションを参照)。または、プロビジョニングするリソースを減らしてコストを抑えることもできますが、Auto Scaling に依存します。一部の DR 実装では、初期トラフィックを処理するのに十分なリソースをデプロイし、低 RTO を確保してから、Auto Scaling を使用して後続のトラフィックを増やします。

マルチサイトアクティブ/アクティブ

マルチサイトアクティブ/アクティブまたはホットスタンバイアクティブ/パッシブ戦略の一環として、複数のリージョンでワークロードを同時に実行できます。マルチサイトアクティブ/アクティブは、デプロイ先のすべてのリージョンからのトラフィックを処理しますが、ホットスタンバイは 1 つのリージョンからのトラフィックのみを処理し、他のリージョン(複数可)はディザスタリカバリにのみ使用されます。マルチサイトアクティブ/アクティブアプローチを使用すると、ユーザーはデプロイされている任意のリージョンでワークロードにアクセスできます。このアプローチはディザスタリカバリに対する最も複雑でコストのかかるアプローチですが、正しいテクノロジーの選択と実装により、ほとんどのディザスタの復旧時間をほぼゼロに短縮できます(ただし、データの破損はバツ

クアックに依存する必要があり、通常はゼロ以外の復旧ポイントになる可能性があります)。ホットスタンバイは、ユーザーが単一のリージョンにのみ誘導され、DR リージョンがトラフィックを受け取らないアクティブ/パッシブ設定を使用します。ほとんどのお客様は、2 番目のリージョンで完全な環境を立ち上げる場合は、アクティブ/アクティブを使用するのが理にかなっています。または、両方のリージョンを使用してユーザートラフィックを処理したくない場合、ウォームスタンバイはより経済的で運用上それほど複雑ではないアプローチを提供します。

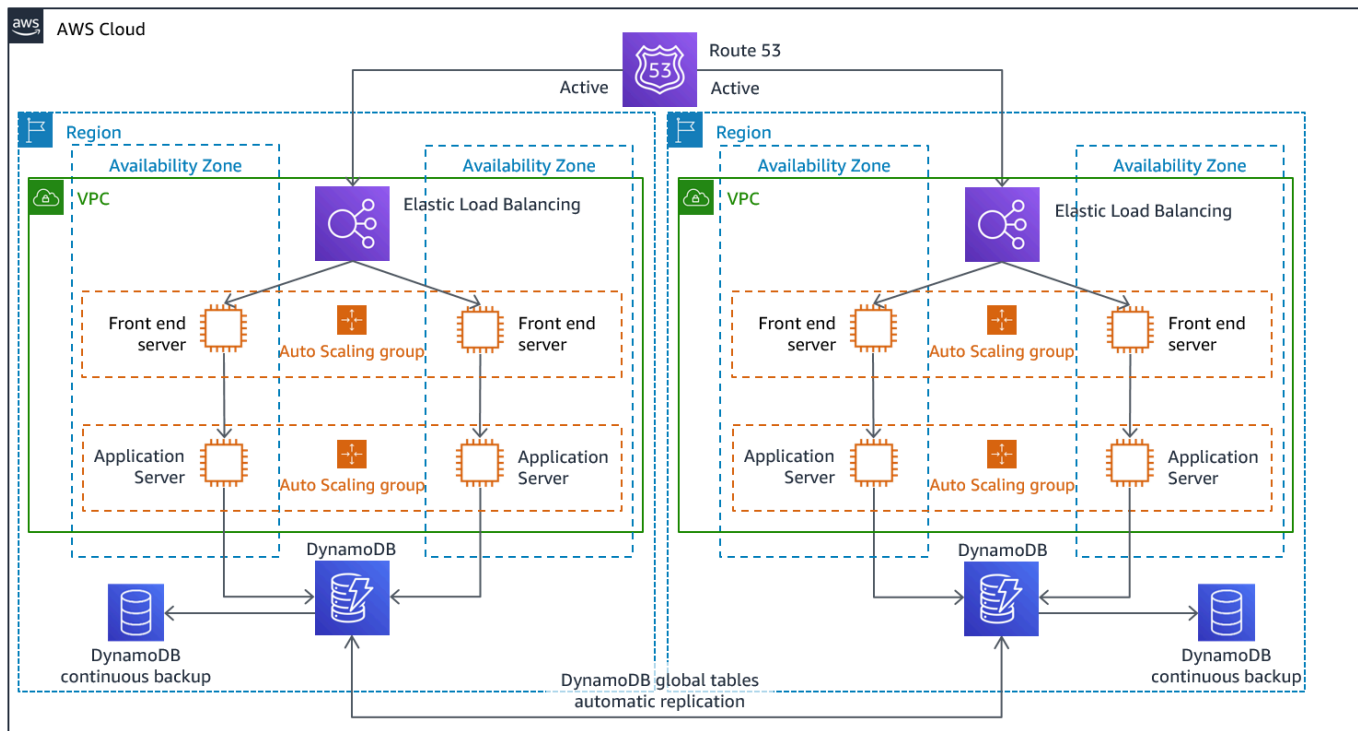


図 12 - マルチサイトアクティブ/アクティブアーキテクチャ (ホットスタンバイの場合は 1 つのアクティブパスを非アクティブに変更)

マルチサイトがアクティブ/アクティブの場合、ワークロードが複数のリージョンで実行されているため、このシナリオではフェイルオーバーのようなものではありません。この場合のディザスタリカバリテストでは、ワークロードがリージョンの損失にどのように反応するかに焦点を当てます。トラフィックは障害が発生したリージョンからルーティングされますか？ 他のリージョン (複数可) はすべてのトラフィックを処理できますか？ データ災害のテストも必要です。バックアップと復旧は依然として必要であり、定期的にテストする必要があります。また、データの破損、削除、難読化を伴うデータ災害の復旧時間は常に 0 より大きく、復旧ポイントは常に災害が発見される前のいずれかの時点になることに注意してください。マルチサイトアクティブ/アクティブ (またはホットスタンバイ) アプローチの複雑さとコストがほぼゼロの復旧時間を維持するために必要になった場合は、セキュリティを維持し、人為的災害を軽減するための人為的ミスを防ぐために、追加の努力を行う必要があります。

AWS サービス

[バックアップと復元](#)、[パイロットライト](#)、[ウォームスタンバイ](#)の対象となるすべての AWS サービスは、ここでは point-in-time データバックアップ、データレプリケーション、アクティブ/アクティブトラフィックルーティング、EC2 インスタンスを含むインフラストラクチャのデプロイとスケールアップにも使用されます。

前述のアクティブ/パッシブシナリオ (パイロットライトとウォームスタンバイ) AWS Global Accelerator では、Amazon Route 53 と の両方をアクティブなリージョンへのネットワークトラフィックのルーティングに使用できます。ここでのアクティブ/アクティブ戦略では、どちらのサービスも、どのユーザーがどのアクティブなリージョンエンドポイントに移動するかを決定するポリシーの定義を有効にします。AWS Global Accelerator では、各アプリケーションエンドポイントに転送される [トラフィックの割合を制御するようにトラフィックダイヤル](#)を設定します。Amazon Route 53 はこのパーセンテージアプローチと、地理的近接性やレイテンシーベースのポリシーなど、[利用可能な他の複数のポリシー](#)をサポートしています。[Global Accelerator は、AWS エッジサーバーの広範なネットワークを自動的に活用](#)して、トラフィックをできるだけ早く AWS ネットワークバックボーンにオンボードするため、リクエストのレイテンシーが低くなります。

この戦略による非同期データレプリケーションにより、ほぼゼロの RPO が可能になります。[Amazon Aurora グローバルデータベース](#)などの AWS サービスは、専用インフラストラクチャを使用して、データベースをアプリケーションに完全に提供し、1 秒未満の一般的なレイテンシーで最大 5 つのセカンダリリージョンにレプリケートできます。アクティブ/パッシブ戦略では、書き込みはプライマリリージョンに対してのみ行われます。アクティブ/アクティブとの違いは、アクティブな各リージョンへの書き込みとのデータ整合性の処理方法を設計することです。読み取りローカルと呼ばれる、ユーザーに最も近いリージョンから提供されるユーザー読み取りを設計するのが一般的です。書き込みには、いくつかのオプションがあります。

- 書き込みグローバル戦略は、すべての書き込みを 1 つのリージョンにルーティングします。そのリージョンに障害が発生した場合、別のリージョンが書き込みを受け入れるように昇格されます。[Aurora グローバルデータベース](#)は、リージョン間でリードレプリカとの同期をサポートしているため、書き込みグローバルに適しています。また、いずれかのセカンダリリージョンを昇格させて、読み取り/書き込みの責任を 1 分以内に引き受けることができます。Aurora は書き込み転送もサポートしています。これにより、Aurora グローバルデータベースのセカンダリクラスターは、書き込みオペレーションを実行する SQL ステートメントをプライマリクラスターに転送できます。
- 書き込みローカル戦略は、書き込みを最も近いリージョン (読み取りと同様) にルーティングします。[Amazon DynamoDB グローバルテーブル](#)では、このような戦略が可能になり、グローバルテーブルがデプロイされるすべてのリージョンからの読み取りと書き込みが可能になります。

す。Amazon DynamoDB グローバルテーブルでは、最後のライターが同時更新間の調整を獲得します。

- 書き込みパーティション戦略は、書き込みの競合を避けるために、パーティションキー (ユーザー ID など) に基づいて特定のリージョンに書き込みを割り当てます。この場合、[双方向に設定された Amazon S3 レプリケーション](#)を使用できます。現在、は 2 つのリージョン間のレプリケーションをサポートしています。このアプローチを実装するときは、バケット A と B の両方で[レプリカ変更同期](#)を有効にして、レプリケートされたオブジェクトのオブジェクトアクセスコントロールリスト (ACLs)、オブジェクトタグ、オブジェクトロックなどのレプリカメタデータの変更をレプリケートするようにしてください。アクティブなリージョンのバケット間で[削除マーカをレプリケート](#)するかどうかを設定することもできます。レプリケーションに加えて、データの破損や破壊イベントから保護するためのpoint-in-timeバックアップも戦略に含める必要があります。

AWS CloudFormation は、複数の AWS リージョンの AWS アカウント間で一貫してデプロイされたインフラストラクチャを強制する強力なツールです。[AWS CloudFormation StackSets](#) は、単一のオペレーションで複数のアカウントとリージョンにまたがる CloudFormation スタックを作成、更新、または削除できるようにすることで、この機能を拡張します。AWS CloudFormation は YAML または JSON を使用して Infrastructure as Code を定義しますが、[AWS Cloud Development Kit \(AWS CDK\)](#) では使い慣れたプログラミング言語を使用して Infrastructure as Code を定義できます。コードは CloudFormation に変換され、AWS にリソースをデプロイするために使用されます。

検出

ワークロードが提供すべきビジネス成果を達成していないことをできるだけ早く知ることが重要です。これにより、災害を迅速に宣言し、インシデントから復旧できます。積極的な復旧目標の場合、復旧目標を達成するには、この応答時間と適切な情報を組み合わせることが重要です。目標復旧時間が 1 時間の場合は、インシデントの検出、適切な担当者への通知、エスカレーションプロセスの関与、予定復旧時間に関する情報の評価 (DR 計画を実行せずに)、災害の宣言、1 時間以内の復旧を行う必要があります。

Note

RTO がリスクにさらされているにもかかわらず、利害関係者が DR を呼び出さないことになった場合は、DR 計画と目標を再評価します。DR 計画を呼び出さない決定は、計画が不十分であるか、実行に自信がないためである可能性があります。

インシデントの検出、通知、エスカレーション、発見、宣言を計画と目標に組み込み、ビジネス価値を提供する現実的で達成可能な目標を提供することが重要です。

AWS は、サービスの可用性に関する up-to-the-minute [Service Health Dashboard](#) に公開します。いつでも [こちら](#) をチェックして現在のステータス情報を取得するか、RSS フィードをサブスクライブして、個々のサービスの中断を通知します。Service Health Dashboard に表示されていないサービスのいずれかで、リアルタイムの運用上の問題が発生した場合は、[サポートリクエスト](#)を作成できます。

[AWS Health Dashboard](#) は、アカウントに影響を与える可能性のある AWS Health イベントに関する情報を提供します。情報は 2 つの方法で表示されます。ダッシュボードには、最近のイベントおよび予定されているイベントがカテゴリ別に分類されて表示されます。詳細なイベントログには、過去 90 日間のすべてのイベントが表示されます。

最も厳しい RTO 要件については、[ヘルスチェックに基づいて自動フェイルオーバーを実装できます](#)。ユーザーエクスペリエンスを表し、主要パフォーマンス指標に基づいてヘルスチェックを設計します。ディープヘルスチェックは、ワークロードの主要な機能を実行し、浅いハートビートチェックを超えています。複数のシグナルに基づいてディープヘルスチェックを使用します。このアプローチでは、誤アラームをトリガーしないように注意してください。不要な場合にフェイルオーバーすると、可用性のリスクが生じるためです。

ディザスタリカバリのテスト

ディザスタリカバリの実装をテストして実装を検証し、ワークロードの DR リージョンへのフェイルオーバーを定期的にテストして、RTO と RPO が満たされていることを確認します。

回避すべきパターンは、ほとんど実行されない復旧パスの開発です。例えば、読み取り専用のクエリに使用されるセカンダリデータストアがあるとします。データストアの書き込み時にプライマリデータストアで障害が発生した場合、セカンダリデータストアにフェイルオーバーします。もしこのフェイルオーバーを頻繁にテストしない場合、セカンダリデータストアの機能に関する前提が正しくない可能性があります。最後にテストしたときに十分だったセカンダリの容量が、このシナリオでは負荷を許容できなくなるか、セカンダリリージョンのサービスクォータが十分ではない可能性があります。

エラー復旧がうまくいくのは頻繁にテストする経路のみであることは、これまでの経験からも明らかです。これが、少数の復旧パスが最適である理由です。

復旧パターンを確立して定期的にテストできます。複雑または重要な復旧パスがある場合でも、復旧パスが機能することを検証するために、本番環境でその障害を定期的に行う必要があります。

DR リージョンで設定ドリフトを管理します。インフラストラクチャ、データ、および設定が DR リージョンで必要に応じてあることを確認します。たとえば、AMIs とサービスクォータが up-to-dateであることを確認します。

[AWS Config](#) を使用して、AWS リソース設定を継続的にモニタリングおよび記録できます。AWS Config はドリフトを検出し、[AWS Systems Manager Automation](#) をトリガーしてドリフトを修正し、アラームを発生させます。[AWS CloudFormation](#) は、デプロイしたスタックのドリフトをさらに検出できます。

結論

お客様は、クラウドでのアプリケーションの可用性について責任を負います。災害とは何かを定義し、この定義とそれがビジネス成果に与える影響を反映するディザスタリカバリ計画を立てることが重要です。影響分析とリスク評価に基づいて目標復旧時間 (RTO) と目標復旧時点 (RPO) を作成し、災害を軽減するための適切なアーキテクチャを選択します。災害の検出を可能かつタイムリーに行う — 目標がいつ危険にさらされているかを把握することが重要です。計画があることを確認し、テストして計画を検証します。信頼性の欠如やディザスタリカバリ目標の達成の失敗により、検証されていないディザスタリカバリ計画が実装されないリスク。

寄稿者

本ドキュメントの寄稿者は次のとおりです。

- Alex Livingstone、プラクティスリードクラウドオペレーション、AWS エンタープライズサポート
- アマゾン ウェブ サービス、プリンシパル信頼性ソリューションアーキテクト、Seth Eliot

詳細情報

詳細については、次を参照してください。

- [AWS アーキテクチャセンター](#)
- [信頼性の柱、AWS Well-Architected フレームワーク](#)
- [ディザスタリカバリ計画チェックリスト](#)
- [ヘルスチェックの実装](#)
- [AWS でのディザスタリカバリ \(DR\) アーキテクチャ、パート I: クラウドでの復旧戦略](#)
- [AWS でのディザスタリカバリ \(DR\) アーキテクチャ、パート II: 迅速な復旧によるバックアップと復元](#)
- [AWS でのディザスタリカバリ \(DR\) アーキテクチャ、パート III: パイロットライトとウォームスタンバイ](#)
- [AWS でのディザスタリカバリ \(DR\) アーキテクチャ、パート IV: マルチサイトアクティブ/アクティブ](#)
- [Amazon Route 53 を用いたディザスタリカバリ \(DR\) のメカニズム](#)
- [ディザスタリカバリディザスタリカバリプランの依存関係を最小化する](#)
- [AWS Well-Architected ディザスタリカバリラボのハンズ](#)
- [AWS ソリューションの実装: マルチリージョンアプリケーションアーキテクチャ](#)
- [AWS re:Invent 2018: マルチリージョンアクティブ/アクティブアプリケーションのアーキテクチャパターン \(ARC209-R2\)](#)

ドキュメント履歴

このホワイトペーパーの更新に関する通知を受け取るには、RSS フィードにサブスクライブしてください。

変更	説明	日付
マイナーな更新	全体のバグ修正と多数のマイナーな変更。	2022 年 4 月 1 日
ホワイトペーパーの更新	軽微な編集上の更新。	2022 年 3 月 21 日
ホワイトペーパーの更新	データプレーンとコントロールプレーンに関する情報を追加しました。アクティブ/パッシブフェイルオーバーの実装方法の詳細を追加しました。CloudEndure ディザスタリカバリを Elastic Disaster Recovery AWS に置き換えました。	2022 年 2 月 17 日
マイナーな更新	AWS Well-Architected Tool 情報を追加しました。	2022 年 2 月 11 日
初版発行	ホワイトペーパーの初回発行。	2021 年 2 月 12 日

注意

お客様は、本書に記載されている情報を独自に評価する責任を負うものとし、(a) 情報提供のみを目的とし、(b) AWS の現行製品と慣行について説明しており、これらは予告なしに変更されることがあり、(c) AWS およびその関連会社、サプライヤー、またはライセンサーからの契約上の義務や保証をもたらすものではありません。AWS の製品やサービスは、明示または黙示を問わず、一切の保証、表明、条件なしに「現状のまま」提供されます。お客様に対する AWS の責任は AWS 契約によって規定されています。また、本文書は、AWS とお客様との間の契約に属するものではなく、また、当該契約が本文書によって修正されることもありません。

© 2022 Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS 用語集

最新の AWS 用語については、AWS の用語集 リファレンスの[AWS 用語集](#)を参照してください。

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。