AWS ホワイトペーパー

AWS マルチリージョンの基礎

AWS マルチリージョンの基礎: AWS ホワイトペーパー

Table of Contents

AWS マルチリージョンの基礎

発行日: 2022 年 12 月 20 日 (ドキュメントの改訂)

要約

この高度な 300 レベルの文書は、クラウドアーキテクトおよびマルチリージョンアーキテクチャを使用してワークロードの耐障害性を向上させることに関心を持ち、AWS でワークロードを構築する上級管理者を対象としています。この文書は、AWS インフラストラクチャとサービスに関する基本的な知識を前提としています。マルチリージョンの一般的なユースケースについて概説し、設計、開発、デプロイに関するマルチリージョンの基本的な概念と影響について共有し、マルチリージョンアーキテクチャがワークロードに適しているかどうかを判断するのに役立つ規範的ガイダンスを提供します。

Well-Architected の実現状況の確認

AWS Well-Architected フレームワークは、クラウド内でのシステム構築に伴う意思決定の長所と短所を理解するのに役立ちます。このフレームワークの 6 つの柱により、信頼性、安全性、効率、費用対効果、持続可能性の高いシステム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を確認できます。AWS Management Console で無料で提供されている AWS Well-Architected Tool 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評価できます。

クラウドアーキテクチャに関する専門的なガイダンスやベストプラクティス (リファレンスアーキテクチャのデプロイ、図、ホワイトペーパー) については、AWS アーキテクチャセンターを参照してください。

序章

各 AWS リージョン は、1 つの地理的エリア内における、複数の独立した、物理的にも分離されたアベイラビリティーゾーンで構成されています。各リージョンのソフトウェアサービス間の論理的な分離は、厳密に管理されています。この意図的な設計により、あるリージョンのインフラストラクチャまたはサービスの障害によって、別のリージョンで相関する障害が発生することはありません。

AWS のお客様のほとんどは、複数のアベイラビリティーゾーン (AZ) またはリージョンの AWS のサービスを使用して、単一リージョンにおけるワークロードの耐障害性の目標を達成できます。ただし、3 つの理由からマルチリージョンアーキテクチャを検討するお客様もいます。

- 単一のリージョンでは満たせないと思われる最も高い階層のワークロードに対して、高可用性とオペレーションの継続性という要件があります。
- 特定の法管轄区域内でワークロードを運用することを要求するデータ主権要件 (現地の法律、規制、コンプライアンスの遵守など) を満たしている必要があります。
- エンドユーザーに最も近い場所でワークロードを実行することにより、ワークロードのパフォーマンスとカスタマーエクスペリエンスを向上させる必要があります。

この文書は、高可用性とオペレーションの継続性の要件に焦点を当てており、ワークロードにマルチリージョンアーキテクチャを採用する際の考慮事項をナビゲートするのに役立ちます。マルチリージョンのワークロードの設計、開発、デプロイに適用される基本的な概念と、マルチリージョンアーキテクチャが特定のワークロードに適しているかどうかを判断するのに役立つ規範的なフレームワークについて説明します。マルチリージョンアーキテクチャは困難で、正しく実行しないとワークロードの全体的な可用性が減少する可能性があるため、マルチリージョンアーキテクチャがワークロードにとって正しい選択であることを確認する必要があります。

単一リージョンでの耐障害性を実現するエンジニアリングとオペレーション

マルチリージョンの概念に取り掛かる前に、まず、単一のリージョンでワークロードが既に可能な限り回復性を備えていることを確認します。これを実現するには、AWS Well-Architected フレームワークの信頼性の柱とオペレーショナルエクセレンスの柱に照らしてワークロードを評価し、推奨されるベストプラクティスを採用するために必要な変更を加えます。AWS Well-Architected フレームワークには、次の概念が含まれます。

- ドメイン境界に基づくワークロードのセグメント化
- 明確に定義されたサービス契約
- 依存関係の管理と結合
- 障害、再試行、バックオフ戦略の処理
- 冪等性オペレーションおよびステートフルトランザクションとステートレストランザクションの比較
- オペレーショナルレディネスと変更管理
- ワークロードの状態の理解
- イベントへの対応

単一リージョンでの耐障害性をさらに高めるには、「グレー障害に対処するための高度なマルチ AZ 耐障害性パターン」で説明されている概念を確認して適用してください。この文書では、各アベイラビリティーゾーンでレプリカを使用して障害を封じ込める際のベストプラクティスを説明し、AWS Well Architected で導入されたマルチ AZ の概念について詳しく説明します。1 つのリージョンで最高の耐障害性を実現するための推奨コンセプトとベストプラクティスを完全に適用したら、特定のワークロードをマルチリージョンアーキテクチャの基本と照らし合わせて評価し、マルチリージョンアプローチを使用してワークロードの耐障害性を高めることができるかどうかを判断できます。

マルチリージョンの基本 1: 要件の理解

前述のように、高可用性とオペレーションの継続性が、マルチリージョンアーキテクチャを追求する一般的な理由です。可用性メトリクスは、定義された期間内でワークロードが使用可能な時間の割合を測定します。一方、オペレーション継続性メトリクスは、大規模で通常は長期間にわたるイベントの回復率を測定します。

可用性の測定はほぼ継続的なプロセスです。具体的な測定値やメトリクスはさまざまですが、通常は目標の可用性 (99.99% の可用性など) を中心にまとめられます。可用性の目標は、常に同じではありません。可用性の目標は、すべてのワークロードに単一の目標を適用するのではなく、重要でないコンポーネントと重要なコンポーネントを分離してワークロードレベルで確立する必要があります。

オペレーションを継続するためには、通常、以下のポイントインタイム測定が使用されます。

- 目標復旧時間 (RTO) – 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。この値によって、サービスが停止される許容時間が決まります。
- 目標復旧時点 (RPO) — RPO は、最後に実行されたデータ復旧時点からの最大許容時間です。これにより、最新の復旧時点からサービス中断までの間のデータ損失がどの程度まで許容されるかが決まります。

可用性目標の設定と同様に、RTO と RPO もワークロードレベルで定義する必要があります。より積極的なオペレーションの継続性や高可用性要件を実現するには、投資を増やす必要があります。とはいえ、すべてのアプリケーションが同じレベルの耐障害性を要求したり、必要とするわけではありません。階層化メカニズムを作成すると、ビジネスオーナーと IT オーナーが連携して、ビジネスへの影響度に基づいて最も要求の厳しいアプリケーションを特定し、それに応じて階層化するためのフレームワークを確立するのに役立ちます。階層化の例は次の表のとおりです。

表 1 — SLA の耐障害性階層化の例

可用性サービスレベルアグリーメント (SLA)	耐障害性階層	許容できるダウンタイム/年
99.99%	プラチナ	52.60 分
99.90%	ゴールド	8.77 時間
99.5%	シルバー	1.83 日間

表 2 — RTO と RPO の耐障害性階層化の例

階層	最大 RTO	最大 RPO	条件	コスト
プラチナ	15 分	5 分	ミッションクリティカルなワークロード	$$$
ゴールド	15 分 ~ 6 時間	2 時間	重要だがミッションクリティカルではないワークロード	$$
シルバー	6 時間 ~ 数日間	24 時間	クリティカルでないワークロード	$

耐障害性を考慮したワークロードを設計する際には、高可用性とオペレーションの継続性の関係を理解する必要があります。例えば、99.99% の可用性を必要とするワークロードの場合、年間 53 分を超えるダウンタイムは許容されません。障害を検出するまでに少なくとも 5 分かかり、オペレーターが関与して復旧手順を決定し、これらの手順を実行するまでにさらに 10 分かかる場合があります。1 つの問題から回復するのに 30 ~ 45 分かかることも珍しくありません。この場合、相関関係のある影響を排除する独立したインスタンスを提供するマルチリージョン戦略を採用することで、初期障害を個別に優先順位付けしながら、一定の時間内にフェイルオーバーすることで、オペレーションを継続できます。このような場合には、適切な RTO と RPO を定義する必要があります。

極めて高い可用性ニーズ (99.99% 以上の可用性など) が必要なミッションクリティカルなワークロードや、別のリージョンへのフェイルオーバーによってのみ満たされる厳しいオペレーションの継続性要件には、マルチリージョンのアプローチが適している場合があります。ただし、これらの要件は通常、リカバリ時間が分単位または時間単位で制限されている企業のワークロードポートフォリオのごく一部にのみ適用されます。アプリケーションに数分または数時間の復旧時間が必要でない限り、影響を受けるリージョン内でアプリケーションの障害が修正されるのを待つ方が適切な方法であり、通常は下位階層のワークロードに合わせて調整されます。

マルチリージョンのアーキテクチャを実装する前に、業務上の意思決定者と技術チームは、オペレーションコストやインフラストラクチャのコスト要因など、コスト面での影響について意見を一致させる必要があります。一般的なマルチリージョンアーキテクチャでは、単一リージョンのアプローチ

に比べて 2 倍のコストがかかる可能性があります。ビジネス継続のためのマルチリージョンパターンには、ホットスタンバイ、ウォームスタンバイ、パイロットライトでの実行など、いくつかあります。復旧目標を達成するリスクが最も低いパターンは、ホットスタンバイの実行で、ワークロードのコストが 2 倍になります。

主なガイダンス

- RTO や RPO などのオペレーションの可用性と継続性の目標は、ワークロードごとに設定し、ビジネスと IT の利害関係者と調整する必要があります。
- 可用性とオペレーションの継続性に関する目標は、ほとんどの場合 1 つのリージョン内で達成できます。単一のリージョンでは達成できない目標については、コスト、複雑さ、利点間のトレードオフを明確に把握した上で、マルチリージョンを検討する必要があります。

マルチリージョンの基本 2: データについて

データ管理は、マルチリージョンアーキテクチャを使用する重要な問題です。リージョン間の地理的な距離によって課されるレイテンシーは避けることができません。これは、リージョン間でデータをレプリケートするのにかかる時間として現れます。可用性、データ整合性、およびマルチリージョンアーキテクチャを使用するワークロードに桁違いに高いレイテンシーを導入することとの間でのトレードオフが必要になります。非同期または同期レプリケーションのどちらを使用する場合も、レプリケーションテクノロジーにより課せられる動作の変化を処理できるようにアプリケーションを変更する必要があります。データ整合性とレイテンシーに関する課題により、単一リージョン向けに設計された既存のアプリケーションをマルチリージョンにすることは非常に困難です。特定のワークロードのデータ整合性要件とデータアクセスパターンを理解することは、トレードオフを比較検討するのに重要です。

2a: データ整合性要件について

CAP 定理は、データ整合性、可用性、ネットワークパーティションの間のトレードオフ (1 つのワークロードで同時に満たせるのは2つだけです) を推論するためのリファレンスを提供します。定義上、マルチリージョンはリージョン間のネットワークパーティションを含むため、可用性と整合性のどちらかを選択する必要があります。

リージョン間のデータの可用性を選択した場合、リージョン間でコミットされたデータの非同期レプリケーションに依存するため、トランザクションの書き込み中に大きなレイテンシーが発生することはなく、レプリケーションが完了するまでリージョン間の整合性が低下します。非同期レプリケーションでは、プライマリリージョンで障害が発生すると、プライマリリージョンからの書き込みがレプリケーションを保留にする可能性が高くなります。その結果、レプリケーションが再開されるまで最新のデータが使用できなくなり、停止したリージョンからレプリケートされなかった未完了のトランザクションを処理するための調整プロセスが必要になります。

非同期レプリケーションが有利なワークロードの場合、リージョン間での非同期レプリケーションを提供する Amazon Aurora や Amazon DynamoDB などのサービスを使用できます。Amazon Aurora Global Database と Amazon DynamoDB のグローバルテーブルの両方に、レプリケーションラグのモニタリングに役立つデフォルトの Amazon CloudWatch メトリクスがあります。

ワークロードはデータの非同期レプリケーションを受け入れることができ、イベントを再生することにより状態を再構築できるため、イベント駆動型アーキテクチャを活用するようにワークロードを組み込むことは、マルチリージョン戦略にとってメリットになります。ストリーミングサービスおよびメッセージングサービスはメッセージペイロードデータを単一のリージョンにバッファするため、

リージョンのフェイルオーバー/フェイルバックプロセスには、クライアントの入力データフローをリダイレクトするメカニズムと、機能停止が発生したリージョンに保存されている未処理や未配信のペイロードを調整するメカニズムが含まれている必要があります。

整合性を選択した場合、トランザクションの書き込み中にデータが同期的にレプリケートされるため、大きなレイテンシーが発生します。複数のリージョンに同期的に書き込みを行う場合、すべてのリージョンで書き込みが成功しなかった場合、トランザクションがコミットされず、再試行が必要になるため、可用性が潜在的に低下します。データをすべてのリージョンに同期的に書き込もうとする再試行は、試行のたびにレイテンシーを犠牲にして行われます。ある時点で、再試行回数を使い切ったときに、トランザクションを完全に失敗させて可用性を低下させるか、使用可能なリージョンにのみトランザクションをコミットして不整合を生じさせるかの決定をする必要があります。Paxos のようなクォーラムフォーミングテクノロジーは、データを同期的にレプリケートしてコミットするのに役立ちますが、開発者は多くの投資を必要とします。

強固な整合性要件を満たすため、書き込みに複数のリージョン間での同期レプリケーションが含まれる場合、書き込みレイテンシーは桁違いに大きくなります。通常、書き込みレイテンシーが高い場合、大幅な変更を加えずに遡及的にアプリケーションに適合させることはできません。理想的には、アプリケーションを最初に設計するときに考慮する必要があります。同期レプリケーションが優先されるマルチリージョンのワークロードの場合、AWS パートナーソリューションが役に立ちます。

2b: データアクセスパターンについて

ワークロードのデータアクセスパターンは、読み取り集約型か書き込み集約型のいずれかの種類に分類されます。特定のワークロードにおけるこの特性を理解することが、適切なマルチリージョンアーキテクチャの選択のガイドになります。

完全な読み取り専用の静的コンテンツなど、読み取り集約型のワークロードの場合、アクティブ/アクティブのマルチリージョンアーキテクチャの実現は非常に複雑というわけではありません。コンテンツ配信ネットワーク (CDN) を使用してエッジで静的コンテンツを提供すると、エンドユーザーに最も近いコンテンツをキャッシュすることにより可用性を確保できます。Amazon CloudFront 内のオリジンフェイルオーバーなどの機能セットを使用して、これを実現できます。もう 1 つのオプションは、ステートレスコンピューティングを複数のリージョンにデプロイし、DNS を使用してユーザーを最も近いリージョンにルーティングしてコンテンツを読み込ませることです。これを実現するには、位置情報ルーティングポリシーを使用した Route 53 を使用できます。

書き込みより読み取りの割合が大きい読み取り集約型のワークロードの場合、ローカル読み取り、グローバル書き込み戦略を使用できます。これにより、すべての書き込みを特定のリージョンのデータベースに送り、他のすべてのリージョンには非同期でデータをレプリケートします。すべてのリージョンで読み取りを実行でき、これが達成されます。このアプローチでは、リージョン間の書き込

みレプリケーションのレイテンシーが増加するため、ローカルでの読み取りが古くなることがあるため、結果整合性を受け入れるワークロードが必要です。

Aurora グローバルデータベースは、すべての読み取りトラフィックをローカルでのみ処理できるスタンバイリージョンのリードレプリカと、書き込みを処理する特定のリージョンの単一のプライマリデータストアのプロビジョニングに役立ちます。データはプライマリデータベースからスタンバイデータベース (リードレプリカ) に非同期でレプリケートされ、スタンバイリージョンに操作をフェイルオーバーする必要がある場合はスタンバイデータベースをプライマリに昇格できます。ワークロードが非リレーショナルデータモデルにより適している場合、このアプローチでも同様に DynamoDB を使用できます。ワークロードは結果整合性を受け入れることが必要であり、最初からそのように設計されていなかった場合は、書き直す必要が生じる可能性があります。

書き込み集約型のワークロードの場合、プライマリリージョンを選択し、スタンバイリージョンへのフェイルオーバー機能をワークロードに組み込む必要があります。アクティブ/アクティブアプローチと比較し、プライマリ/スタンバイアプローチはそれほど複雑ではありません。これは、アクティブ/アクティブアーキテクチャでは、リージョンへのインテリジェントなルーティングを処理し、セッションアフィニティを確立し、冪等性トランザクションを確保し、潜在的な競合を処理するようにワークロードを書き直す必要があるためです。

耐障害性のためにマルチリージョンに注目しているワークロードのほとんどは、アクティブ/アクティブアプローチを必要としません。シャーディング戦略を使用し、クライアントベース全体で障害の影響範囲を制限することにより、耐障害性を高めることができます。クライアントベースを効果的にシャードできる場合、各シャードに異なるプライマリリージョンを選択できます。例えば、リージョンをセルとして扱って、クライアントの半分をリージョン 1 に、残り半分をリージョン 2 に合わせるようにクライアントをシャードできる場合、マルチリージョンセルによるアプローチ作成でき、その結果、ワークロードへの影響範囲が小さくなります。

シャーディングアプローチをプライマリ/スタンバイアプローチと組み合わせて、シャードにフェイルオーバー機能を提供することができます。テスト済みのフェイルオーバープロセスをワークロードに組み込む必要があり、またデータ調整のプロセスも組み込んで、フェイルオーバーの後のデータストアのトランザクション整合性を確保する必要があります。これらの詳細については、この文書の後半で説明します。

主なガイダンス

- 障害が発生すると、レプリケーション待ちの書き込みがスタンバイリージョンにコミットされない可能性が高くなります。レプリケーションが再開されるまで、データは使用できません (非同期レプリケーションの場合)。

- フェイルオーバーの一環として、非同期レプリケーションを使用するデータストアのトランザクションの一貫した状態を維持するためにデータ調整プロセスが必要になります。
- 強固な整合性が必要な場合は、同期的にレプリケートするデータストアに必要なレイテンシーを許容するようにワークロードを変更する必要があります。

マルチリージョンの基本 3: ワークロードの依存関係を理解する

特定のワークロードには、使用される AWS のサービス、内部依存関係、サードパーティーの依存関係、ネットワークの依存関係、証明書、キー、シークレット、パラメータなど、リージョン内に複数の依存関係が存在する場合があります。障害発生時にワークロードを確実に稼働させるには、プライマリリージョンとスタンバイリージョンの間に依存関係がなく、それぞれが独立して動作できる必要があります。そのためには、ワークロードのすべての依存関係を精査して、各リージョン内で利用可能であることを確認する必要があります。これが必要なのは、プライマリリージョンで障害が発生してもスタンバイリージョンには影響を与えないようにするためです。さらに、依存関係の機能が低下したり、完全に利用できなくなったりした場合に、ワークロードがどのように動作するかについての知識が不可欠です。そうすることで、これを適切に処理するソリューションを設計できます。

3a: AWS のサービス

マルチリージョンアーキテクチャを設計する場合、使用される特定の AWS のサービスを理解する必要があります。1 つ目の側面は、サービスがマルチリージョンを有効にするためにどのような機能を備えているか、そしてマルチリージョンの目標を達成するためにソリューションを設計する必要があるかどうかを理解することです。例えば、Amazon Aurora と Amazon DynamoDB には、スタンバイリージョンにデータを非同期でレプリケートする機能があります。AWS のサービスの依存関係はすべて、ワークロードが実行されるすべてのリージョンで利用できる必要があります。使用するサービスが該当するリージョンで利用できることを確認するには、「AWS リージョン サービスリスト」を確認してください。

3b: 内部およびサードパーティーの依存関係

ワークロードに内部依存関係がある場合は、そのワークロードが動作するリージョンから利用できることを確認してください。例えば、ワークロードが多数のマイクロサービスで構成されている場合は、ビジネス機能を構成するすべてのマイクロサービスに精通している必要があります。そこから、ワークロードが動作する各リージョンに、それらのマイクロサービスがすべてデプロイされていることを確認します。

ワークロード内のマイクロサービス間のクロスリージョンコールは推奨されず、リージョンの分離を維持する必要があります。これは、クロスリージョンの依存関係を作成すると、相関する障害の発生リスクが高まり、ワークロードを独立したリージョンで実装することで得られるメリットが損なわれるためです。オンプレミスの依存関係もワークロードの一部になる可能性があるため、プライマリ

リージョンが変更された場合にこれらの統合の特性がどのように変化するかを理解することが不可欠です。例えば、スタンバイリージョンがオンプレミス環境から離れた場所にある場合、レイテンシーの増加は悪影響を及ぼします。

Software as a Service (SaaS) ソリューション、Software Development Kit (SDK)、およびその他のサードパーティー製品の依存関係を理解し、これらの依存性が低下したり利用できなくなったりするシナリオを実行できれば、さまざまな障害モードでシステムチェーンがどのように動作し、機能するかについて、より多くの洞察を得ることができます。これらの依存関係は、AWS Secrets Manager やサードパーティーのボールトソリューション (Hashicorp など) を使用してシークレットを外部で管理する方法から、フェデレーテッドログイン用の IAM Identity Center に依存関係を持つ認証システムまで、アプリケーションコード内に存在する可能性があります。

依存関係に冗長性を持たせることで、耐障害性を高めることができます。また、SaaS ソリューションやサードパーティーの依存関係が、ワークロードと同じプライマリ AWS リージョン を使用している可能性もあります。その場合は、ベンダーと協力して、ベンダーの耐障害性がワークロードの要件に合っているかどうかを確認する必要があります。

さらに、サードパーティー製アプリケーションなど、ワークロードとその依存関係との間で共有される運命にも注意してください。フェイルオーバー後にセカンダリリージョンで (またはセカンダリリージョンから) 依存関係が利用できない場合、ワークロードは完全には回復しない可能性があります。

3c: フェイルオーバーメカニズム

ドメインネームシステム (DNS) は、トラフィックをプライマリリージョンからスタンバイリージョンに移すフェイルオーバーメカニズムとして一般的に使用されています。フェイルオーバーメカニズムが取るすべての依存関係を慎重に確認し、精査してください。例えば、ワークロードが Amazon Route 53 を使用している場合、コントロールプレーンが US-East-1 でホストされているということは、その特定のリージョンのコントロールプレーンに依存することになります。プライマリリージョンが US-East-1 の場合も、これをフェイルオーバーメカニズムの一部として使用することはお勧めしません。別のフェイルオーバーメカニズムを使用している場合は、そのメカニズムが想定どおりに動作しないシナリオを十分に理解する必要があります。この理解が確立されたら、不測の事態に備えるか、必要に応じて新しいメカニズムを開発します。フェイルオーバーを成功させるために使用できるアプローチについては、「Creating Disaster Recovery Mechanisms Using Amazon Route 53」を参照してください。

内部依存関係のセクションで説明したように、ビジネス機能の一部であるマイクロサービスはすべて、ワークロードがデプロイされる各リージョンで利用できる必要があります。フェイルオーバー

戦略の一環として、ビジネス機能をまとめてフェイルオーバーし、クロスリージョンコールの可能性を排除する必要があります。また、マイクロサービスが独立してフェイルオーバーすると、マイクロサービスがクロスリージョンコールを行う可能性があるという望ましくない動作が発生する可能性があり、これによりレイテンシーが発生し、クライアントがタイムアウトした場合にワークロードが使用できなくなる可能性があります。

3d: 設定の依存関係

証明書、キー、シークレット、パラメータは、マルチリージョンの設計時に必要な依存関係分析の一部です。可能な限り、これらのコンポーネントを各リージョン内でローカライズして、これらの依存関係についてリージョン間で運命を共有しないようにするのが最善です。証明書について、期限切れとなる証明書 (事前に通知するように設定されている) が複数のリージョンに影響を与えるというシナリオを避けるため、証明書の有効期限は、証明書ごとに、可能であればリージョンごとに異なる必要があります。

暗号化キーとシークレットもリージョン固有のものでなければなりません。そうすれば、キーやシークレットのローテーションで エラーが発生しても、影響は特定のリージョンに限定されます。

最後に、ワークロードが特定のリージョンで取得できるように、すべてのワークロードパラメータをローカルに保存する必要があります。

主なガイダンス

- マルチリージョンアーキテクチャは、リージョン間の物理的および論理的な分離からメリットを得られます。アプリケーションレイヤーでクロスリージョンの依存関係を導入すると、このメリットは損なわれます。このような依存関係は避けてください。
- フェイルオーバー制御は、プライマリリージョンに依存することなく機能する必要があります。
- クロスリージョンコールのレイテンシーや依存性が増す可能性を排除するには、フェイルオーバーをビジネス機能に合わせて調整する必要があります。

マルチリージョンの基本 4: 運用準備について

マルチリージョンのワークロードの運用は、マルチリージョン特有の運用の課題を伴う複雑なタスクです。これらには、AWS アカウント 管理、デプロイプロセスの変更、マルチリージョンオブザーバビリティ戦略の作成、フェイルオーバーおよびフェイルバックのランブックの作成とテスト、そしてコストの管理が含まれます。運用準備状況のレビュー (ORR) は、単一のリージョンで実行されているか、複数のリージョン間で実行されているかにかかわらず、チームが本番稼働用のワークロードを準備するのに役立ちます。

4a: AWS アカウント 管理

AWS リージョン 間でワークロードをデプロイするには、アカウント内のすべての AWS Service Quotas がリージョン間で同等であることを確認します。最初に、アーキテクチャの一部であるすべての AWS のサービスについて調べ、スタンバイリージョンの使用プランを確認してから、現在の使用状況と比較します。場合によっては、これまでスタンバイリージョンを使用したことがないときに、デフォルトの Service Quotas を参照して開始点を理解できます。次に、使用するすべてのサービス間で、Service Quotas コンソール (ログインが必要です) または API を使用してクォータの引き上げをリクエストします。

AWS Identity and Access Management (IAM) ロールを各リージョンで設定し、オペレーター、オートメーションツール、および AWS サービスがスタンバイリージョン内のリソースに対して適切なアクセス許可を持つようにする必要があります。ロールをリージョン別に分離することで、マルチリージョンアーキテクチャのために求められているリージョン別の分離を達成しています。スタンバイリージョンでの本番稼働の前に、これらのアクセス許可が実施されていることを確認してください。

4b: デプロイのプラクティス

マルチリージョン機能を使用すると、複数のリージョンへのワークロードのデプロイが複雑になる可能性があります。AWS CloudFormation は単一または複数のリージョンへのインフラストラクチャのデプロイに役立ち、必要に応じてカスタマイズできます。AWS CodePipeline は、パイプラインが存在するリージョンとは異なるリージョンへのデプロイを可能にするクロスリージョンアクションを備えた、ほぼ継続的なインテグレーション/継続的デリバリー (CI/CD) のパイプラインを提供するのに役立ちます。これをブルー/グリーンなどの堅牢なデプロイ戦略と組み合わせると、最小限の場合、ダウンタイムがゼロのデプロイが可能になります。

ただし、アプリケーションやデータの状態が永続的なストアに外部化されていない場合、ステートフル機能のデプロイはより複雑になる可能性があります。このような場合は、ニーズに合わせてデプロ

イプロセスを慎重にカスタマイズしてください。同時に複数のリージョンにデプロイするのではなく、一度に 1 つのリージョンにデプロイするように、デプロイパイプラインとプロセスを設計してください。これにより、リージョン間で相関する障害が発生する可能性が低くなります。Amazon がソフトウェアのデプロイを自動化するために使用する方法については、Builder Library の記事、「安全なハンズオフデプロイメントの自動化」を参照してください。

4c: オブザーバビリティ

マルチリージョン向けに設計する場合は、各リージョンのすべてのコンポーネントの正常性をどのようにモニタリングして、リージョンの正常性の全体図を取得するか考慮してください。これには、単一リージョンのワークロードでは考慮されないレプリケーションラグのメトリクスのモニタリングが含まれることがあります。

マルチリージョンアーキテクチャを構築する場合、スタンバイリージョンのワークロードのパフォーマンスのモニタリングも考慮します。これには、スタンバイリージョンからヘルスチェックや canary (合成テスト) を実行して、プライマリの正常性を外部から確認できるようにすることが含まれます。さらに、Amazon CloudWatch Internet Monitor を使用して、エンドユーザーの観点から外部ネットワークの状態とワークロードのパフォーマンスを理解できます。同様に、スタンバイリージョンをモニタリングするには、プライマリリージョンでも同じオブザーバビリティを実施している必要があります。これらの canary はカスタマーエクスペリエンスメトリクスをモニタリングして、ワークロード全体の正常性を取得する必要があります。これが必要なのは、プライマリリージョンに問題がある場合、プライマリリージョンでのオブザーバビリティに障害が発生し、ワークロードの正常性を評価する機能に影響することがあるためです。

その場合、そのリージョンを外部から観察することでインサイトが得られます。これらのメトリクスは、各リージョンで利用可能なダッシュボードと、各リージョンで作成されたアラームにロールアップする必要があります。Amazon CloudWatch はリージョン別のサービスであるため、両方のリージョンに配置することが必要です。このモニタリングデータは、プライマリリージョンからスタンバイリージョンへのフェイルオーバーの呼び出しに使用されます。

4d: プロセス、手順、およびテスト

「いつフェイルオーバーを実行するか」という質問に答える最適なタイミングは、必要になるよりずっと前です。ユーザー、プロセス、およびテクノロジーを含むビジネス継続性計画はすべて問題が発生する前に定義し、定期的にテストする必要があります。リカバリの意思決定のフレームワークを決定しておきます。慣れたリカバリプロセスがあり、リカバリの時間をよく理解していれば、フェイルオーバーによって RTO ターゲットを満たすリカバリプロセスを開始するタイミングを選択でき

ます。その時点は、プライマリリージョンのアプリケーションの問題が特定された直後のこともあれば、リージョンのアプリケーション内のリカバリオプションが使い果たされ、RTO を満たすためにフェイルオーバーを開始する必要がある場合のこともあります。

フェイルオーバーアクション自体は 100% 自動化されていますが、フェイルオーバーを有効にする決定は人間 (通常は組織内で事前に決められた少数の個人) が決定する必要があります。また、フェイルオーバーを決定する基準を明確に定義し、組織全体で理解している必要があります。これらのプロセスは AWS System Manager ランブックを使用して定義および完了できます。これにより、完全なエンドツーエンドの自動化が可能になり、テストおよびフェイルオーバー中に実行されるプロセスの整合性が確保されます。

これらのランブックをプライマリリージョンとスタンバイリージョンで使用できるようにし、フェイルオーバーまたはフェイルバックプロセスを開始できるようにする必要があります。この自動化が実施されたら、定期テストの頻度を定義し、従う必要があります。これにより、実際にイベントが発生したときに、組織が信頼し、明確に定義され、練習されたプロセスに基づいてレスポンスが実行されることが確保されます。また、データ調整プロセスの許容値が確立されていることに注意することも重要です。確立された RPO/RTO 要件が提案されたプロセスと一致していることを確認してください。

4e: コストと複雑さ

マルチリージョンアーキテクチャのコストに与える影響は、インフラストラクチャの使用、運用のオーバーヘッド、およびリソース時間の上昇によって駆動されます。前述のように、スタンバイリージョンのインフラストラクチャコストは、事前プロビジョニング時のプライマリリージョンのインフラストラクチャコストと同じであるため、コストは 2 倍になります。日常業務には十分であり、需要の急増も許容できる十分なバッファ容量を予約するように容量をプロビジョニングし、各リージョンに同じ制限を設定します。

さらに、アクティブ-アクティブアーキテクチャを採用している場合、マルチリージョンアーキテクチャを正常に実行するためにアプリケーションレベルの変更が必要になることがあります。これは設計と運用に時間とリソースを大量に消費する可能性があります。少なくとも、組織は各リージョンの技術的およびビジネス上の依存関係を理解し、フェイルオーバーとフェイルバックのプロセスを設計することに時間を費やす必要があります。

また、チームは通常のフェイルオーバーとフェイルバックの演習を行い、イベント中に使用されるランブックに慣れる必要があります。これらの演習は、複数のリージョンの投資から期待される成果を得るためにはきわめて重要ですが、機会損失を表し、他のアクティビティから時間とリソースを取り上げることになります。

主なガイダンス

- AWS Service Quotas を確認し、ワークロードが動作するすべてのリージョンで同等にする必要があります。
- デプロイプロセスでは、同時に複数のリージョンをターゲットにするのではなく、一度に 1 つのリージョンをターゲットにする必要があります。
- レプリケーションラグなど、追加のメトリクスをモニタリングする必要があります。これはマルチリージョンのシナリオ特有のものです。
- プライマリリージョンを超えてワークロードのモニタリングを拡張します。カスタマーエクスペリエンスメトリクスは、リージョンごとにモニタリングし、ワークロードが実行されている各リージョンの外から計測する必要があります。
- フェイルオーバーとフェイルバックは定期的にテストする必要があります。テストとライブイベントの両方で使用されるフェイルオーバーとフェイルバックのプロセスを単一のランブックで実装します。テスト用ランブックとライブイベント用のランブックに違いはありません。

結論

このホワイトペーパーでは、マルチリージョンの一般的な使用例、マルチリージョンアーキテクチャの実装方法の基礎、およびこのアプローチの影響について説明しました。これらの基本はあらゆるワークロードに適用でき、マルチリージョンアーキテクチャが特定のビジネスにとって適切なアプローチであるかどうかを判断する際のフレームワークとして使用できます。

寄稿者

本ドキュメントの寄稿者は次のとおりです。

技術的な寄稿者:

- AWS マルチリージョンチーム、プリンシパルソリューションアーキテクト、John Formento, Jr.

編集寄稿者:

- プロダクトマーケティング担当シニアマネージャー、Lisi Lewis

詳細情報

詳細については、次を参照してください。

- マルチ AZ の高度なレジリエンスパターン (AWS ホワイトペーパー)
- 信頼性の柱 – AWS Well-Architected フレームワーク
- 可用性およびその他:AWS の分散システムの回復力の理解と向上 (AWS ホワイトペーパー)
- AWS 障害分離境界 (AWS ホワイトペーパー)

ドキュメントの改訂

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
ドキュメントの公開	初版	2022 年 12 月 20 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ます。本書は、(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ます。本書は、AWS とお客様との間で締結されるいかなる契約の一部でもなく、その内容を修正するものでもありません。

AWS 用語集

AWS の最新の用語については、「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。