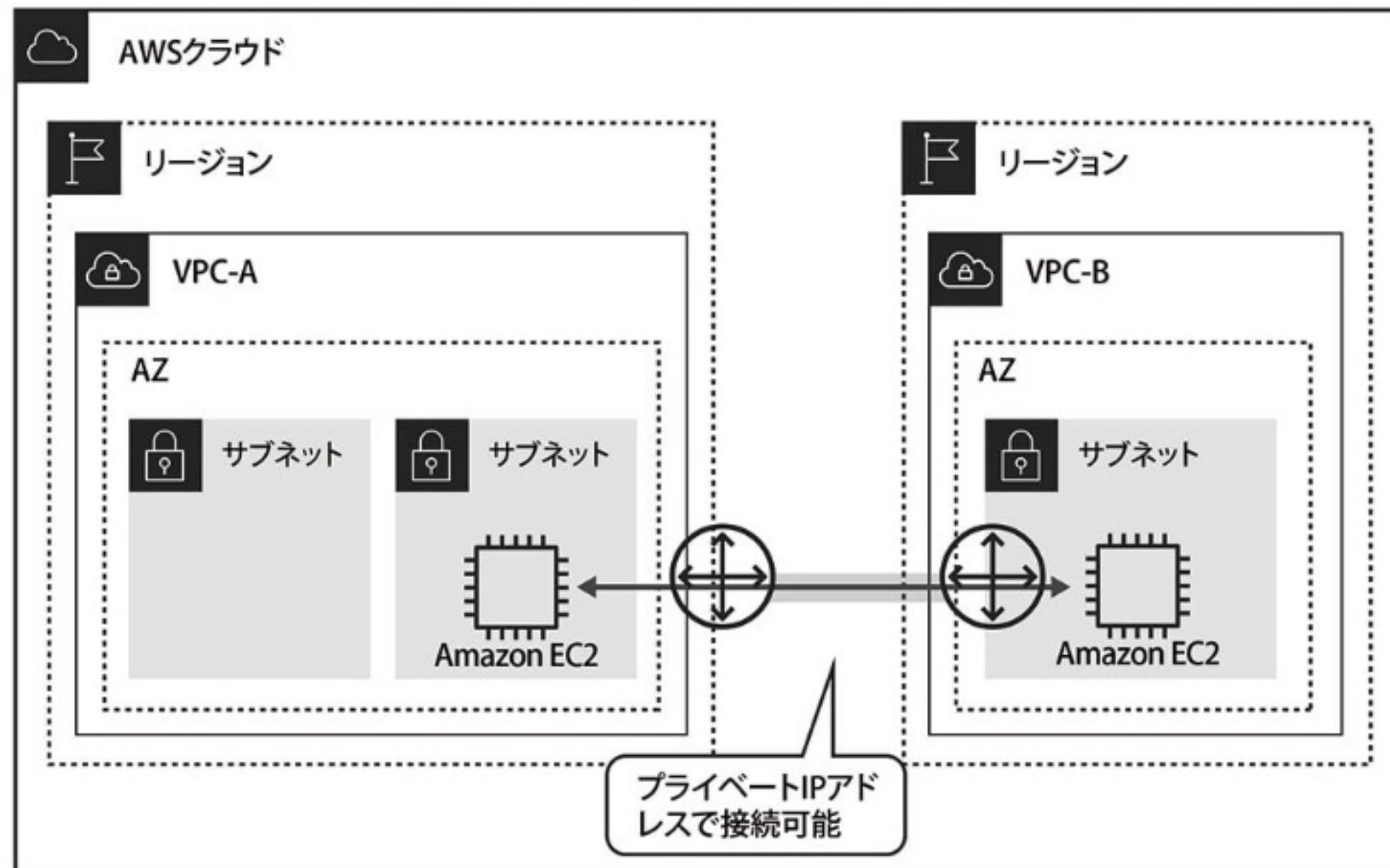


●Disaster Recovery(DR)サイトの構築

複数のリージョンを横断するDRサイトを構築する場合、それぞれのリージョンでVPCを作成することになります。メインサイトのデータをコピーしたりデータベースをレプリケーションする場合は、VPC間の接続をプライベートに行うため、VPCピアリング(26ページを参照)を利用します。

メインサイトからDRサイトへの切り替えは、Route 53のフェイルオーバールーティングなど、ユーザー側で設定する必要があります。

【VPCピアリングの接続例】



Q 演習問題

1 ある会社では、AWSとオンプレミス環境間の通信を行うためにAWS Direct Connectを導入して運用していましたが、先日、Direct Connectの障害でシステムが長時間ダウンしてしまいました。ソリューションアーキテクトは回線の冗長化を検討しており、できるだけ早くかつ低コストで導入したいと考えています。次のうちどのサービスが適切ですか。

- A AWS Direct Connect
- B AWS Site-to-Site VPN
- C Elastic Load Balancing
- D NATゲートウェイ

2 アプリケーションへのパブリックアクセスを制限しながら、最低限の管理と高可用性、高拡張性を備えたAWSサービスは、次のうちどれですか。

- A プライベートサブネットにNATゲートウェイを作成し、プライベートサブネットからインターネットゲートウェイへのルーティングを設定する
- B パブリックサブネットにNATゲートウェイを作成し、プライベートサブネットからNATゲートウェイへのルーティングを設定する
- C プライベートサブネットにNATインスタンスを作成し、プライベートサブネットからインターネットゲートウェイへのルーティングを設定する
- D パブリックサブネットにNATインスタンスを作成し、プライベートサブネットからNATインスタンスへのルーティングを設定する

A**解答**

1

B

AのDirect Connectは、高速なネットワーク帯域で安定した通信を行うことができますが、導入に時間を要し、回線コストもBのSite-to-Site VPNより高価になります。

BのSite-to-Site VPNはインターネット回線を利用しているため、Aの Direct Connectよりスループットや品質は低下しますが、導入が早く、低コストで運用できます。

CのElastic Load Balancingは、Amazon EC2や特定のIPアドレスへのトラフィックを分散するロードバランシング サービスで、AWSとオンプレミス環境間をプライベート接続するサービスではありません。

DのNATゲートウェイは、プライベートサブネットからインターネットへ接続するためのNAT機能を提供するサービスで、AWSとオンプレミス環境間をプライベート接続するサービスではありません。

したがって、**B**が正解です。

2

B

NATゲートウェイには、プライベートサブネットからインターネットへ接続するためのNAT機能を提供しているマネージドサービスで、AZ内で冗長化されています。

プライベートサブネットからインターネットへ接続するためには、インターネットへ接続できるパブリックサブネットに配置したNATゲートウェイヘルーティングする必要があります。

一方、NATインスタンスは、EC2インスタンスをNATサーバーとして利用するもので、EC2の管理や高可用性の仕組みを検討する必要があります。したがって、**B**が正解です。

3-3

コンピューティングにおける高可用性の実現

AWSの各種コンピューティングサービスでは、動作させるアプリケーションの特性に応じて適切な可用性を設計する必要があります。本節では、AWSにおけるコンピューティングサービスや複数のサービスを組み合わせた可用性の実現方法、障害発生時の挙動について説明します。

1

コンピューティングサービス

ここでは、EC2インスタンスの高可用化に貢献するコンピューティングサービスの詳細について説明します。

●Amazon Elastic Compute Cloud(EC2)

Amazon EC2インスタンスはオンプレミス環境と同様に、サーバーにクラスタリング ソフトウェアを導入して高可用性を実現することができますが、AWSでは、Elastic Load Balancing(ELB)による負荷分散、Auto Scalingによる自動スケールアウト／スケールイン、Amazon Route 53によるDNSフェイルオーバーなど、マネージドサービスと組み合わせることで運用の負荷を減らし、障害発生時にも迅速に対応できるシステム設計を行うのが一般的です。

●Auto Scaling

Auto Scalingでは、複数のアベイラビリティゾーン(AZ)を利用して、EC2インスタンスをスケールアウト／スケールインすることができます。

第1章でスケーリングプラン、起動設定、Auto Scalingグループについて説明しましたが、ここでは具体的にEC2インスタンスがどのように増減するのかを説明します。

●スケールアウト

スケールアウトでは、スケーリングプランの設定値をトリガーにして、Auto

Scalingグループの設定値に従ってEC2インスタンスの数を増加させます。

EC2インスタンスは、Auto Scalingグループで指定したサブネットで動作しますが、AZが複数ある場合はAZ間でEC2インスタンス数の均衡を保つように起動します。

たとえば、AZ-Aに3台、AZ-Bに1台の場合、均衡を保つためにAZ-BでEC2インスタンスを起動します。

● スケールイン

一方、複数のAZに分散したEC2インスタンスのスケールインは、デフォルトで次に示す順序で実行されます。

- ① EC2インスタンス数が最も多いAZ内から、ランダムにAZを選択
- ② AZのインスタンス数が同じ場合は、最も古い起動設定を使用したEC2インスタンスのあるAZを選択
- ③ 最も古い起動設定を使用したEC2インスタンスが複数存在する場合は、次に課金が発生するまでの時間が最も短いEC2インスタンスを選択
- ④ さらに、課金が発生するまでの時間が最も短いEC2インスタンスが複数存在する場合は、そのなかからランダムにEC2インスタンスを選択

スケールアウトとスケールインのいずれも、AZでEC2インスタンス数の均衡を保つように設計されているため、いずれかのAZに障害が発生しても、システムを継続することができます。

しかし、単にAuto Scalingを設定するだけでは意図したとおりに動作しないこともあるため、クールダウンやライフサイクル フックといったオプションを設定しておくとよいでしょう。

● クールダウン

クールダウンは、Auto Scalingが連続で実行されないようにAuto Scalingの待ち時間を設定する機能です。

Auto Scalingでは、新しいEC2インスタンスを起動するまでに数分の時間を要しますが、たとえば、起動中に負荷が低減せずに次のAuto Scalingが実行されると、本来想定していたよりもEC2インスタンスの数が増えてしまうことがあります。そのような状況を防ぐために、数分のクールダウンを設定し、EC2

インスタンスが完全に起動するまでは次のAuto Scalingが実行されないようにします。

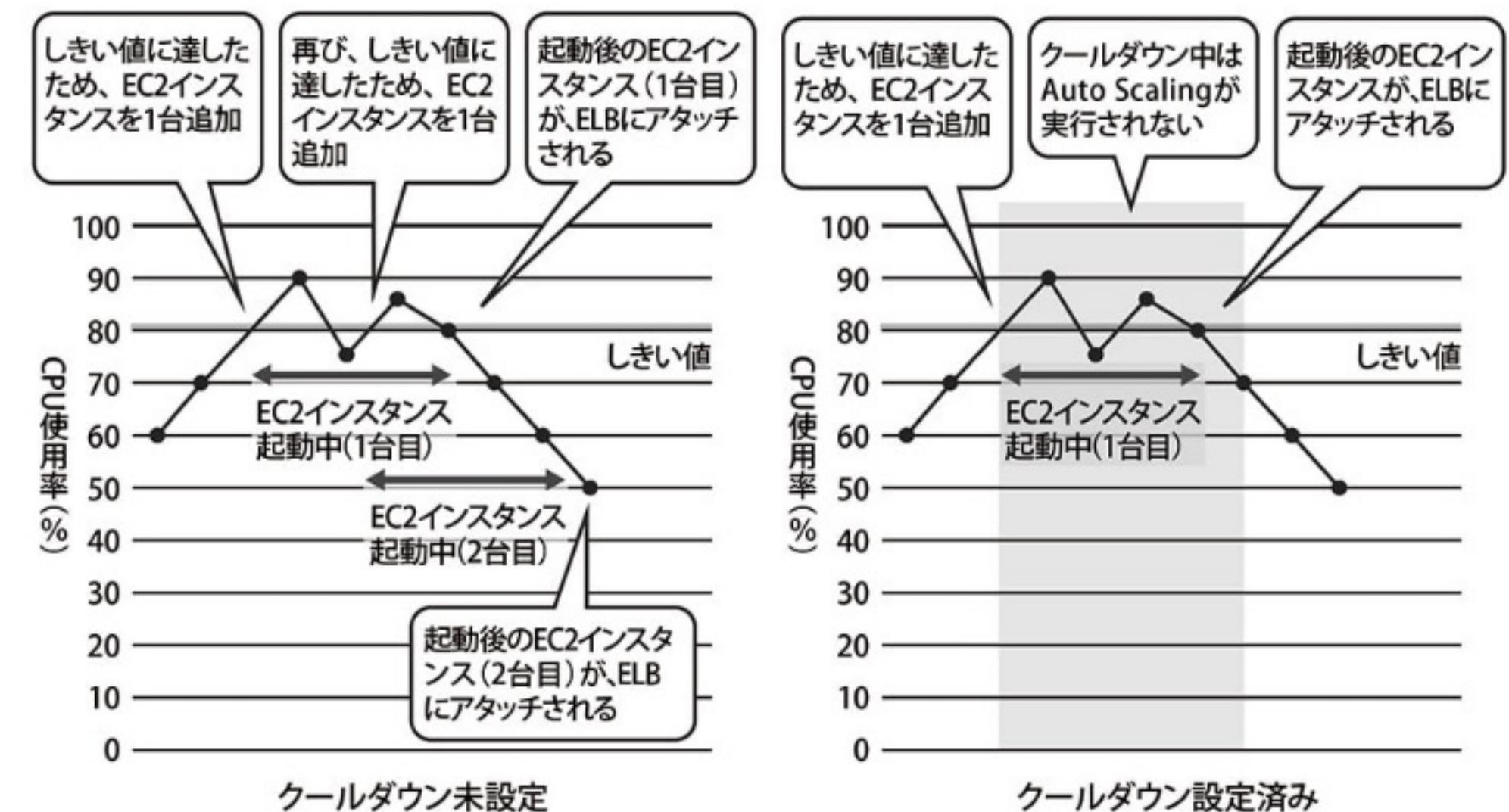
クールダウンには、Auto Scalingグループ全体へ適用するデフォルトのクールダウンと、特定のスケーリングポリシーに適用するスケーリング固有のクールダウンがあります。

● ライフサイクルフック

ライフサイクルフックでは、Auto ScalingによるEC2インスタンスの起動または終了を一時的に待機させて、指定したアクションを実行することができます。たとえば、終了時にログを外部へ出力したり、起動時に特定のデータをロードする場合などに利用します。

【クールダウンのイメージ】

CPU使用率が80%を超えたたら、Amazon EC2インスタンスを1台追加する場合



● Amazon CloudFront

Amazon CloudFrontは、エッジロケーションからコンテンツを配信するCDN(Content Delivery Network)サービスです。

CloudFrontは、高可用性、高パフォーマンス、低レイテンシーなネットワークを備えており、CloudFront自体の可用性を考慮した設計を行う必要はありません。

世界中からアクセスされる大規模なWebサーバーや、予期できないアクセス集中が発生し得るサービスを提供する際に利用されます。

詳細については、「4-2 ネットワークサービスにおけるパフォーマンス」を参照してください。

● AWS Lambda

AWS Lambdaは、サーバーを起動することなくコードが実行できるコンピューティングサービスです。

可用性やスケーリングはすべてLambdaで管理されており、実行した分だけ料金が発生します。処理時間に限りがあるため、時間を要するコードは実行できません。

詳細については、「1-4 コンピューティングサービス」を参照してください。

2

VPCリソースにおける高可用性コンピューティング

AWSでは、さまざまなサービスを組み合わせることで、より可用性の高いシステムを構築することができます。

たとえば、Auto Scalingなどを利用してEC2インスタンスを常に冗長構成にできるように準備しておけば、障害が突然発生してもシステムの稼働を維持することができます。

また、EC2インスタンスの障害以外に、システム負荷も考慮する必要があります。EC2インスタンスの負荷分散にElastic Load Balancing(ELB)を利用するとき、バックエンドインスタンスの負荷を均一に分散できるため、特定のEC2インスタンスの高負荷が原因でシステムダウンするようなケースを避けることができます。

ELBとEC2を高可用性の観点から冗長化する場合は、いずれも複数のAZにリソースを配置することが重要です。

● Elastic Load Balancing (ELB)

Elastic Load Balancing(ELB)は、EC2インスタンスや特定のIPアドレスへのトラフィックを分散するロードバランシングサービスで、EC2の可用性を高めるためにも利用されます。

また、ロードバランサー自身も冗長化しなければ单一障害点となります。ELBは自動的にスケールしており、AWS内部で冗長構成になっています。



通常、Webアクセスを行う際には、WebサーバーまたはロードバランサーのIPアドレスを指定する必要がありますが、ELBを利用する場合はエンドポイントを指定してアクセスします。

ELB自体もIPアドレスを保持していますが、不定期に変更されているため、接続の一貫性を保つためにエンドポイントによる名前解決でアクセスします。

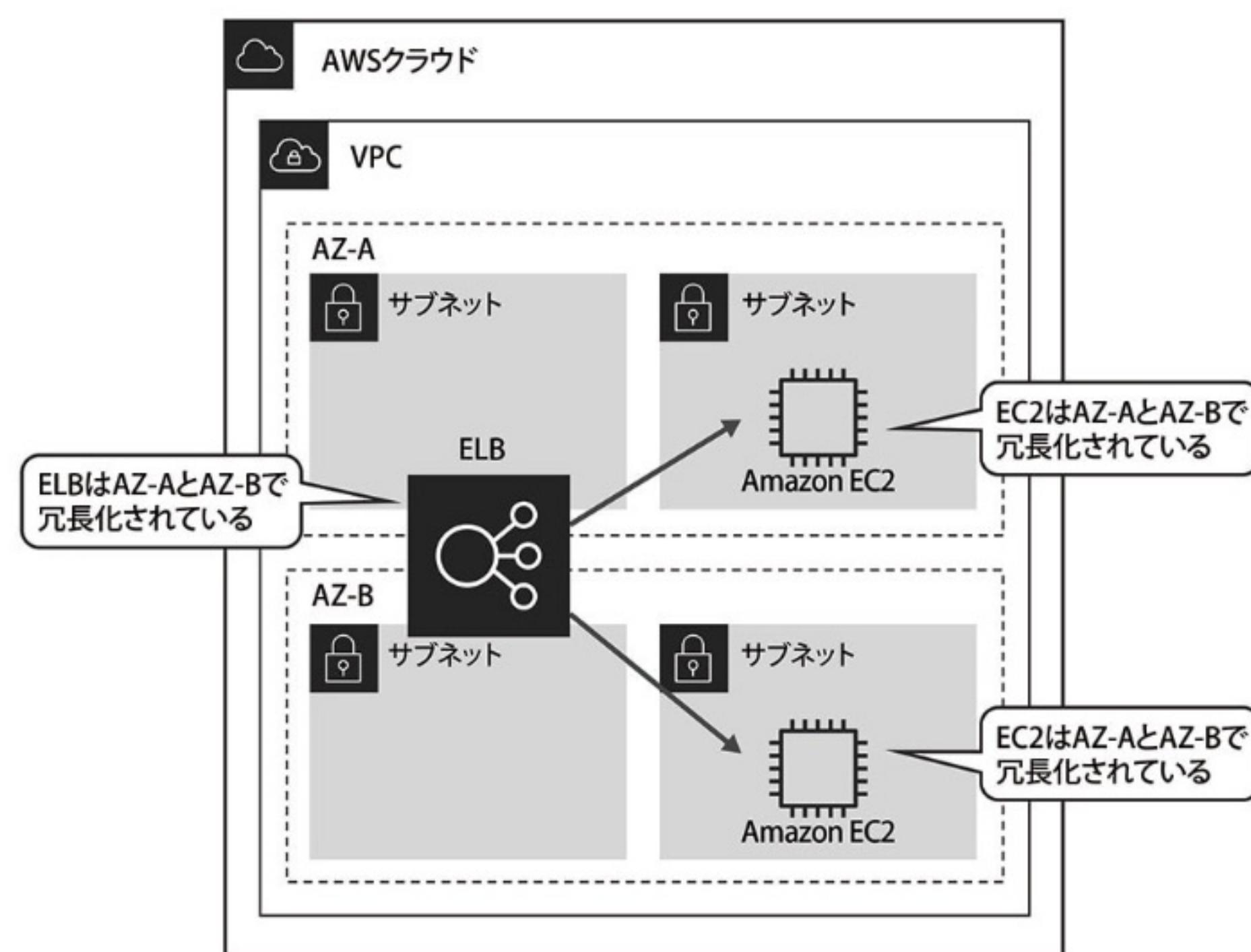
ここからは、VPCリソースにおける高可用性の実現方法を例をあげて説明します。

● 一般的なWebアプリケーションにおける構成例

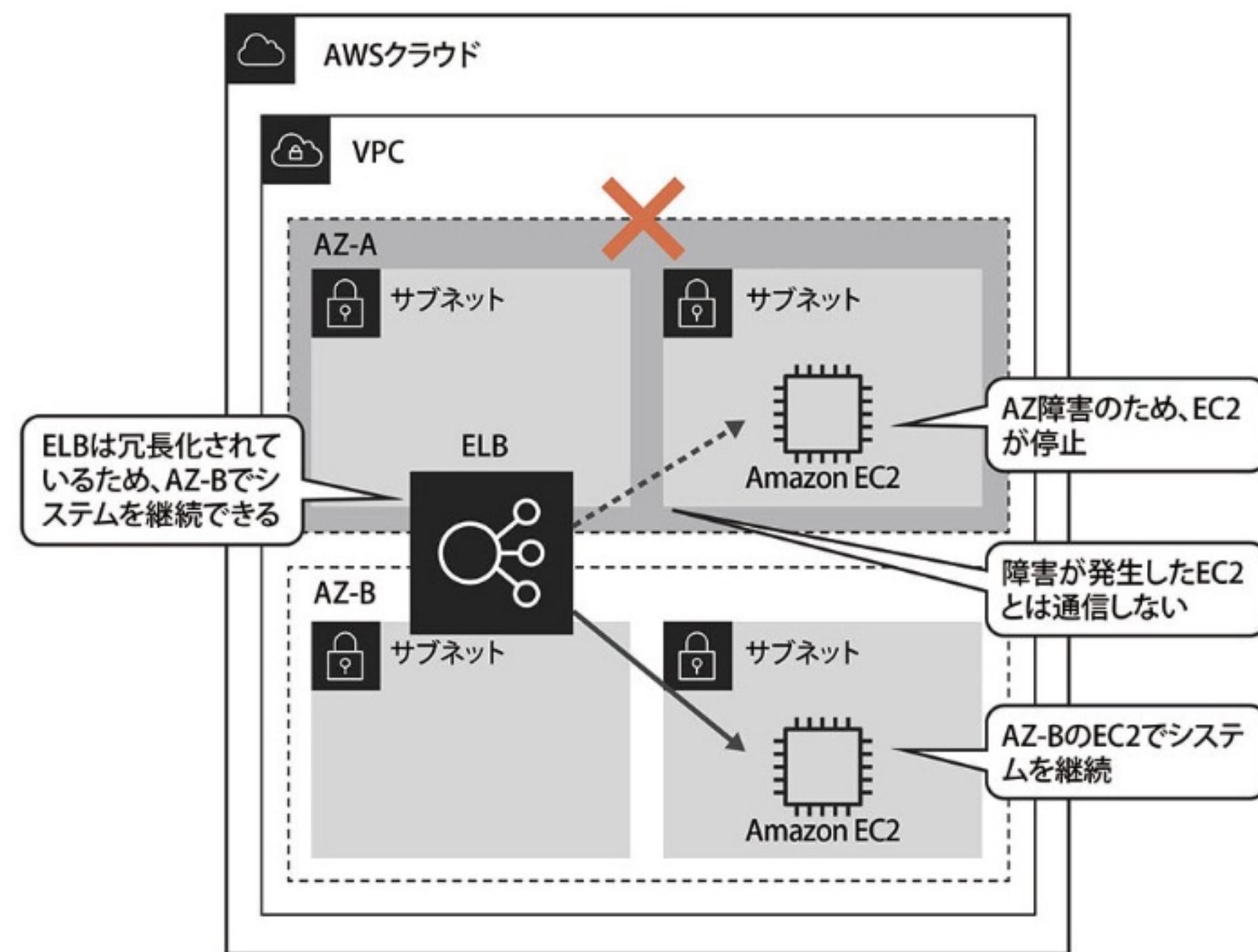
近年は、Webアプリケーションにサーバーレスアーキテクチャが採用されるケースが増えていますが、実際にはまだ、Amazon EC2のような仮想サーバーなどで、アプリケーションを実行するケースも少なくありません。

次の図に、ELBを使用して冗長化した簡単なWebアプリケーションの構成を示します。この例では、ELBおよびEC2インスタンスをマルチAZ構成にしているため、データセンターレベルでの障害発生時でも、システムの動作を継続できます。

【EC2を冗長化したWebアプリケーションの構成例】

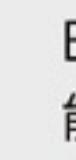
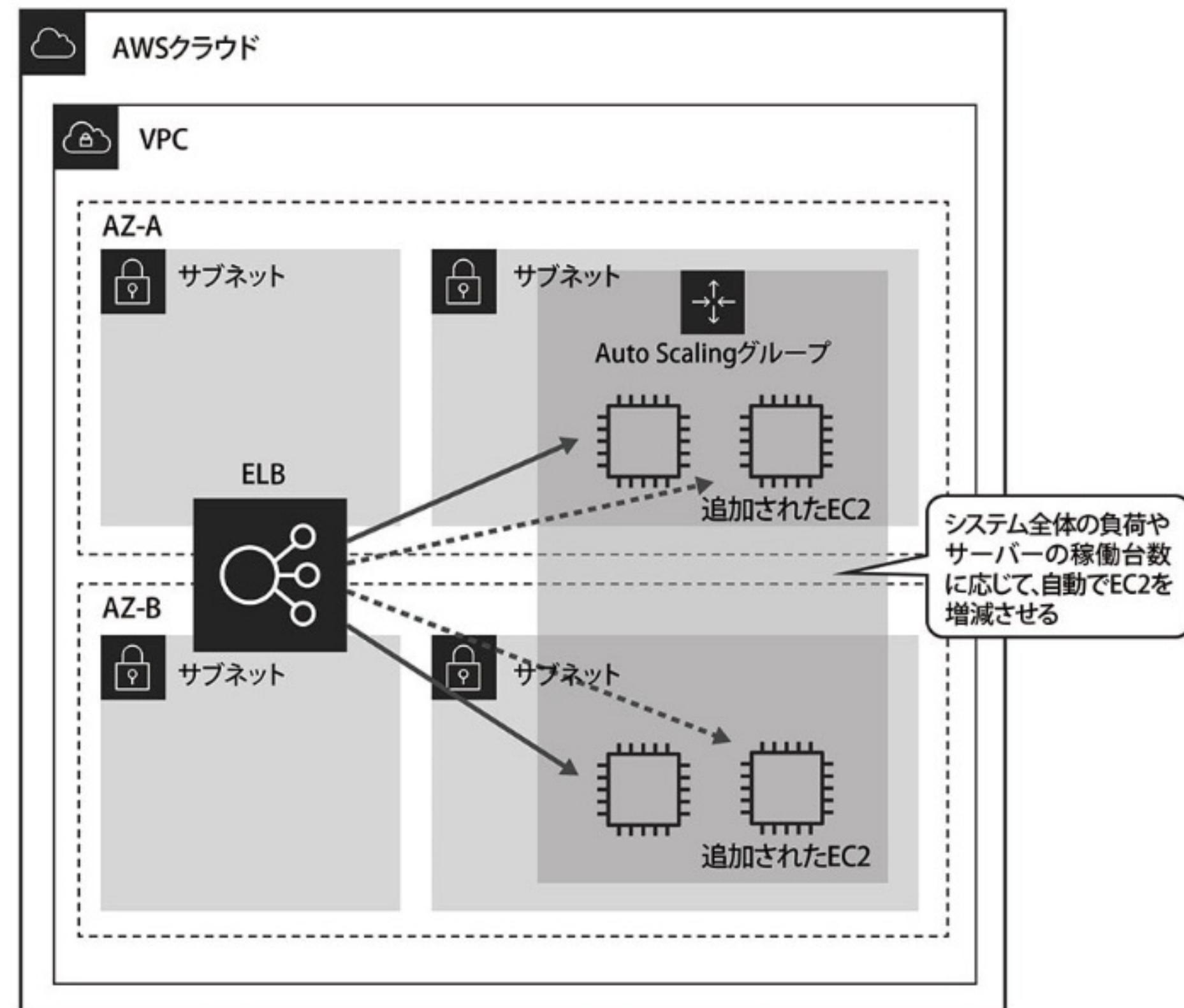


【AZ-Aが障害で利用できない場合】



以下では、**Auto Scaling**を利用してEC2インスタンスを自動で増減するWebアプリケーションの例を示します。この例では、Auto ScalingグループのEC2インスタンスの最小数を2台、最大数を4台としています。

【Auto Scalingを利用したWebアプリケーションの構成例】



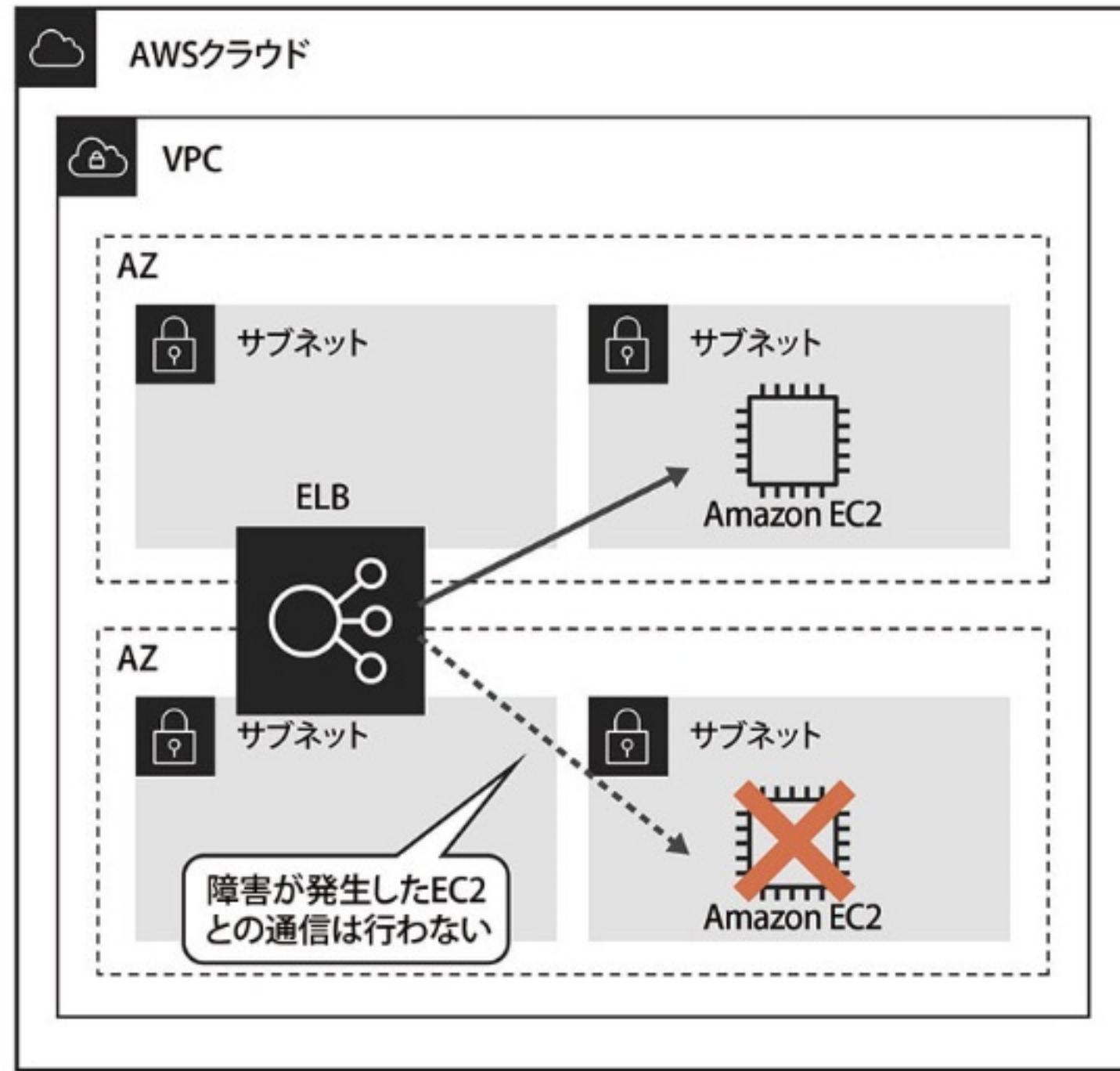
ELBやAuto Scalingを利用して自動的にスケールすることは可能ですが、短時間でアクセスが急増すると、スケールアウトが間に合わずにシステムが過負荷な状態になる可能性があります。AWS CloudFrontなどを活用してキャッシュから応答することで、予期できないアクセス集中に対応できるアーキテクチャを考えることも重要です。

●一般的なWebアプリケーションのEC2インスタンス障害発生時の挙動

Webアプリケーション構成でEC2インスタンスに障害が発生した場合、ELBからEC2インスタンスへのヘルスチェックがエラーになります。ELBは、ヘルスチェックが正常なEC2インスタンスにだけ通信を振り分けるため、ダウンし

たEC2インスタンスとの通信は行わないよう制御します。障害発生時にEC2インスタンスへの接続を行っていたユーザーは、セッションなどを失いますが、システム自体は残りのEC2インスタンスで稼働し続けます。

【WebアプリケーションにおけるECインスタンス障害発生時の挙動】



このWebアプリケーションにおける障害発生時の挙動は、セッションの状態を保持し続けるステートフルなアプリケーションが稼働している場合の例です。

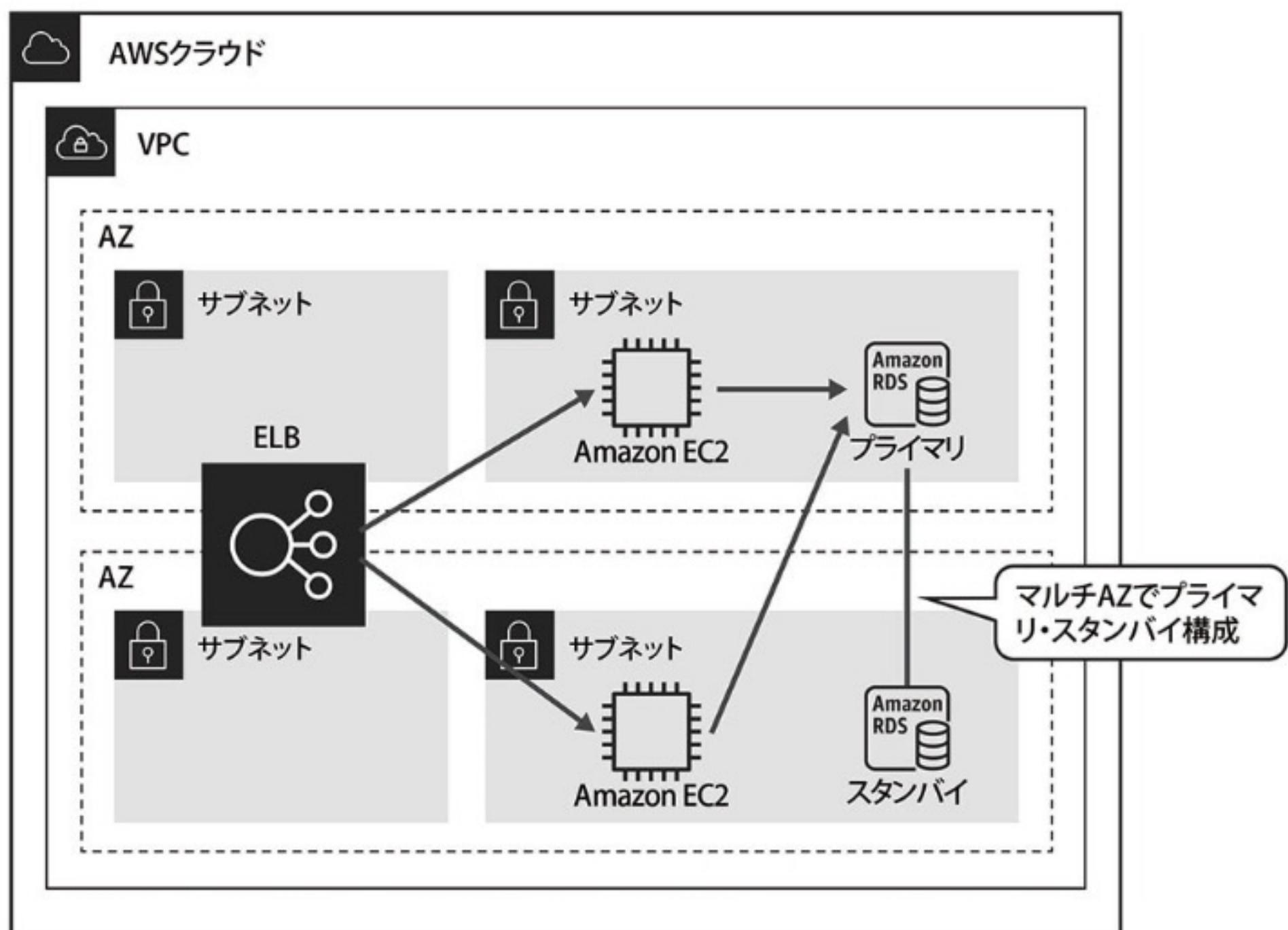
AWSのベストプラクティスとしては、セッション情報を保持しないステートレスなアプリケーションが推奨されます。

●データベースを持つWebアプリケーションにおける構成例

AWSでは、Amazon RDSというデータベースサービスが提供されています。

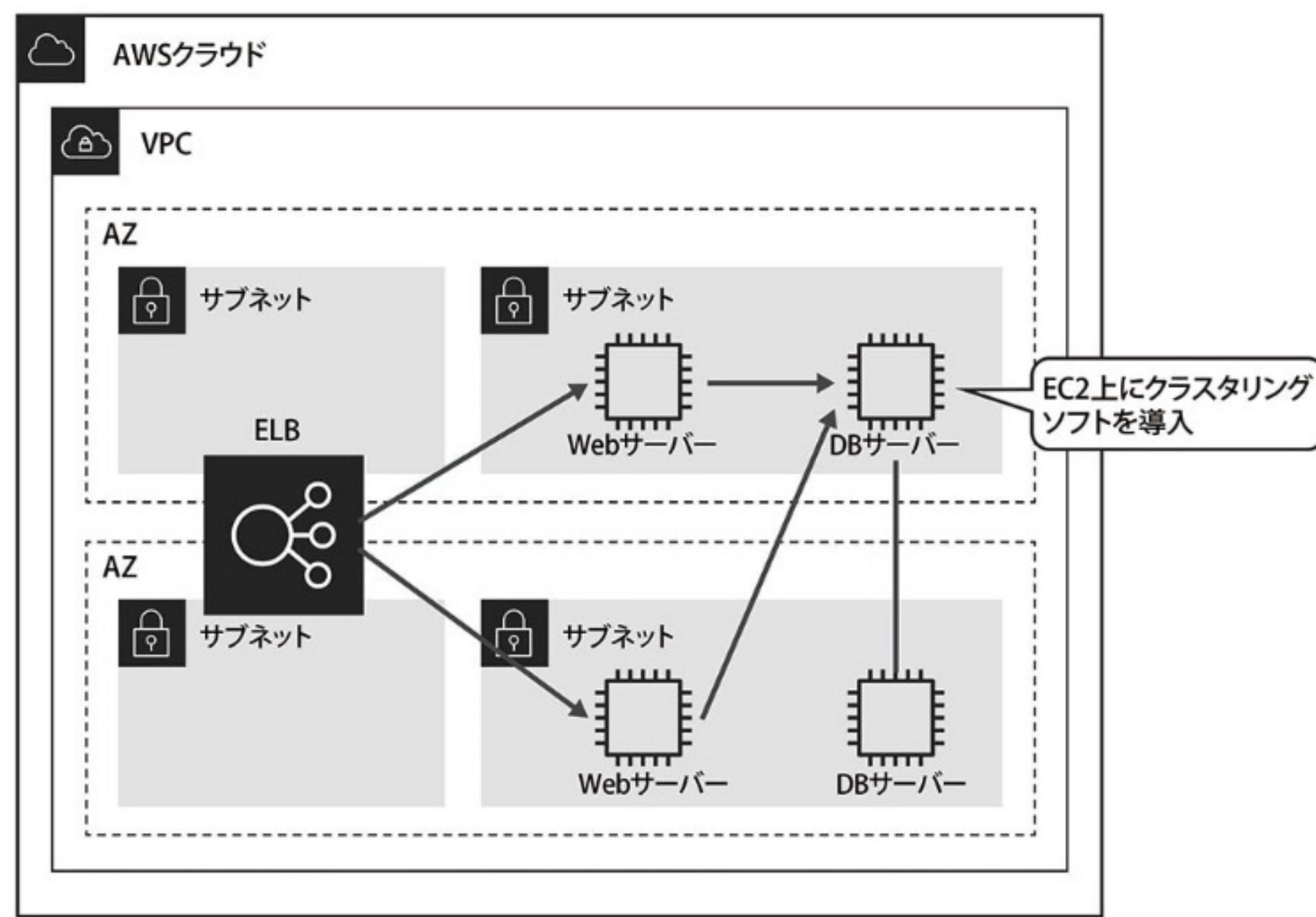
次の図は、Amazon RDSの**マルチAZ機能**によるプライマリ・スタンバイ構成の例です。RDSのプライマリ データベースに障害が発生した場合でも、すぐにスタンバイ データベースへフェイルオーバーすることでシステムの可用性を高めることができます。

【Webデータベースアプリケーションの構成例(RDSを使用)】



次の例では、EC2インスタンス上にデータベースサーバーを構築し、サードパーティ製のクラスタリングソフトウェアで冗長化しています。

【Webデータベースアプリケーションの構成例(EC2を使用)】



● RDSデータベースインスタンス障害発生時の挙動

Amazon RDSの障害は、シングルAZとマルチAZで挙動が変わり、また障害の発生パターンによって復旧方法が異なります。

● シングルAZのRDSに障害が発生した場合

RDSには、障害発生時に自動的に再起動する機能があります。そのため、ユーザー側で再起動のオペレーションは不要ですが、再起動の間はダウンタイムが発生します。

RDS内のデータについては、ユーザー側でデータを復元する必要があるため、ポイントインタイムリカバリ機能などを利用します。

● マルチAZのRDSに障害が発生した場合

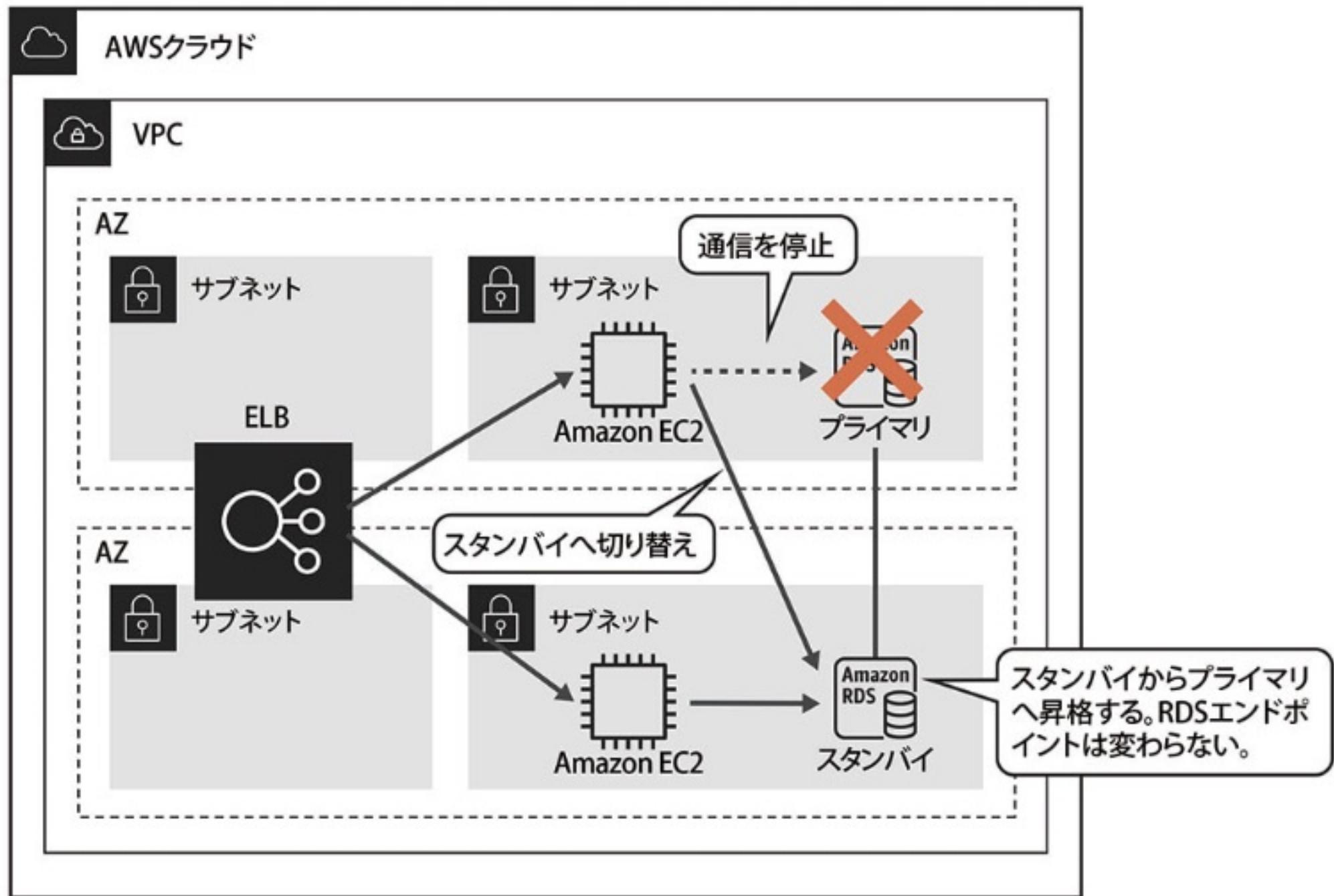
プライマリのRDSデータベースインスタンスに障害が発生した場合、自動的にスタンバイへのフェイルオーバーを行います。

内部的な挙動としては、スタンバイのRDSデータベースインスタンスをプライマリに昇格させます。その際に、プライマリで利用していたDNSレコードがスタンバイのDNSレコードへ切り替わり、RDSのエンドポイントを変更するこ

となくシームレスにスタンバイに切り替わります。この一連の流れは、RDSが自動的に行なうため、ユーザー側でのオペレーションは不要です。

データについては、プライマリとスタンバイとの間で常に同期されているため、リカバリの必要はありません。なお、フェイルオーバー中はわずかにダウンタイムが発生します。

【RDSのフェイルオーバーの動作イメージ】



● データベース内のレコードが破損した場合

RDSの障害以外に、アプリケーションのバグやオペレーションミスによるデータベース内レコードの破損も障害となります。こうしたケースでは、RDSの自動再起動やフェイルオーバーでは復旧できないため、バックアップからリストアする必要があります。

リストアする方法にはいくつかあります。次に示す方法はその一例です。

- ① 障害が発生したRDSのエンドポイント名を控えたあと、別のエンドポイント名に変更する
- ② バックアップから変更前のエンドポイント名でリストアする(必要に応じてポイントインタイムリカバリを行う)

③ エンドポイント名を変更したRDSデータベースインスタンスを削除する

これらの方法よりも早くシステムを復元するには、自動スナップショットからリストアします。この場合、RDSデータベースインスタンスの削除時に、自動バックアップの保持を行う必要があります。自動バックアップを保持しない場合は、削除時に自動スナップショットも一緒に削除されるため、リストアできなくなります。

● インメモリデータベースを持つWebアプリケーションにおける構成例

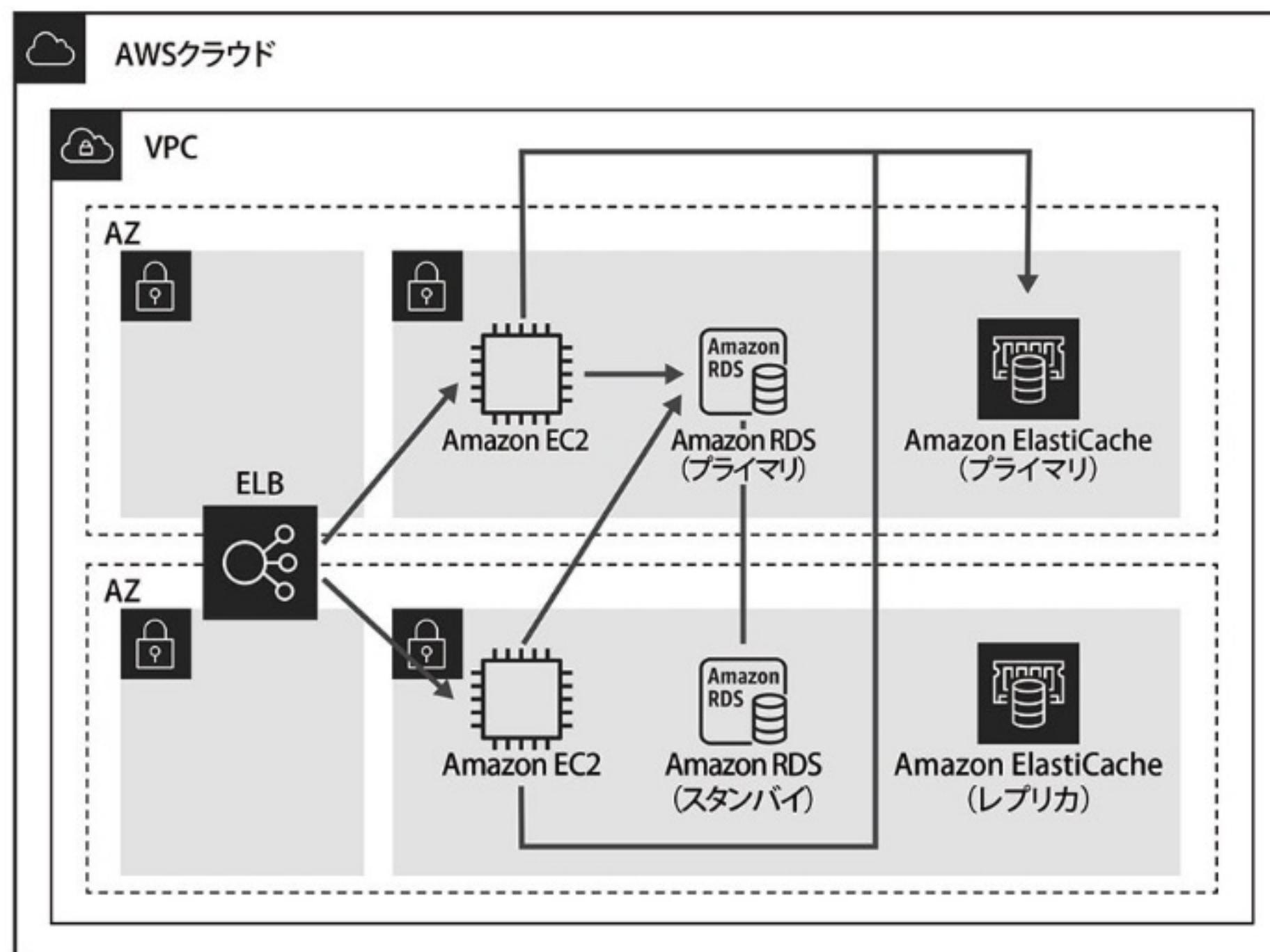
次に、RDSを利用した構成からさらに疎結合な構成を検討してみましょう。

Webのセッション情報や一時的なデータストアには、Amazon ElastiCacheやAmazon DynamoDBを利用します。

次の図で、ElastiCacheを利用した場合の構成を示します。ElastiCacheでは、インメモリデータベースとしてMemcachedとRedisが利用できます。Redisではプライマリ/レプリカ構成でマルチAZ配置が可能で、この構成により高可用性を実現しています。

次の図はRedisの使用を前提とした構成を示しています。

【ElastiCacheを含めたWebアプリケーションの構成例】

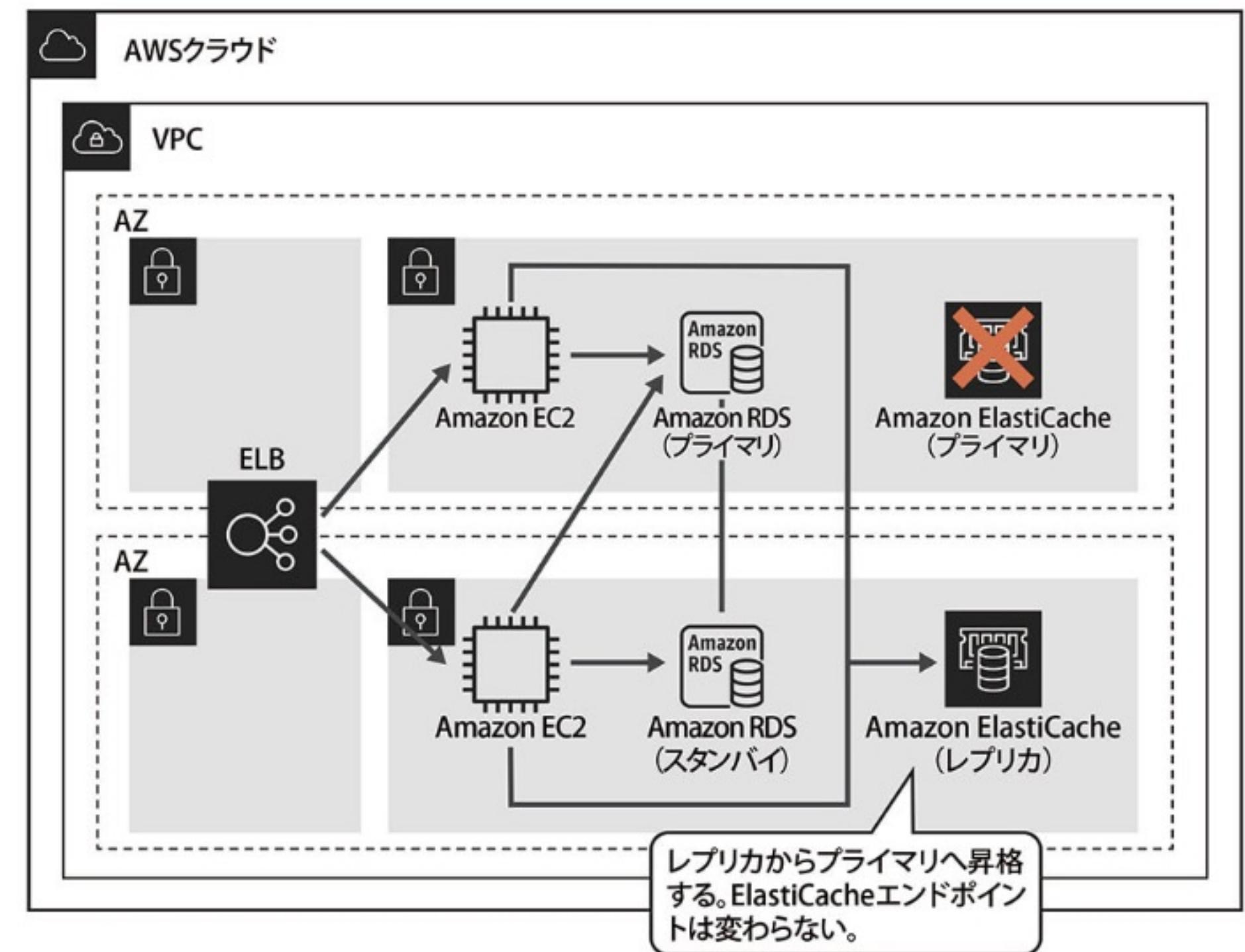


● マルチAZのElastiCacheに障害が発生した場合

プライマリのElastiCacheに障害が発生した場合は、自動的にレプリカへフェイルオーバーを行います。内部的な挙動としては、RDSと同様にレプリカのElastiCacheをプライマリに昇格させます。

なお、フェイルオーバー中はわずかにダウンタイムが発生します。

【ElastiCacheのフェイルオーバーの動作イメージ】



コンピューティングリソースの可用性を向上させるためには、複数のAZにリソースを冗長化することが重要です。Amazon EC2の場合は、ELBやAuto Scalingと組み合わせることで自動スケーリングが可能になり、Amazon RDSの場合はマルチAZ配置にすることで自動切り替えが可能になります。

● サーバーレスWebアプリケーションにおける構成例

ここまででは、コンピューティングリソースとしてAmazon EC2を前提に説明してきましたが、EC2は可用性の考慮や運用管理をユーザー側で行う必要があります。そこで、サーバーレスアーキテクチャを採用することで、運用管理が

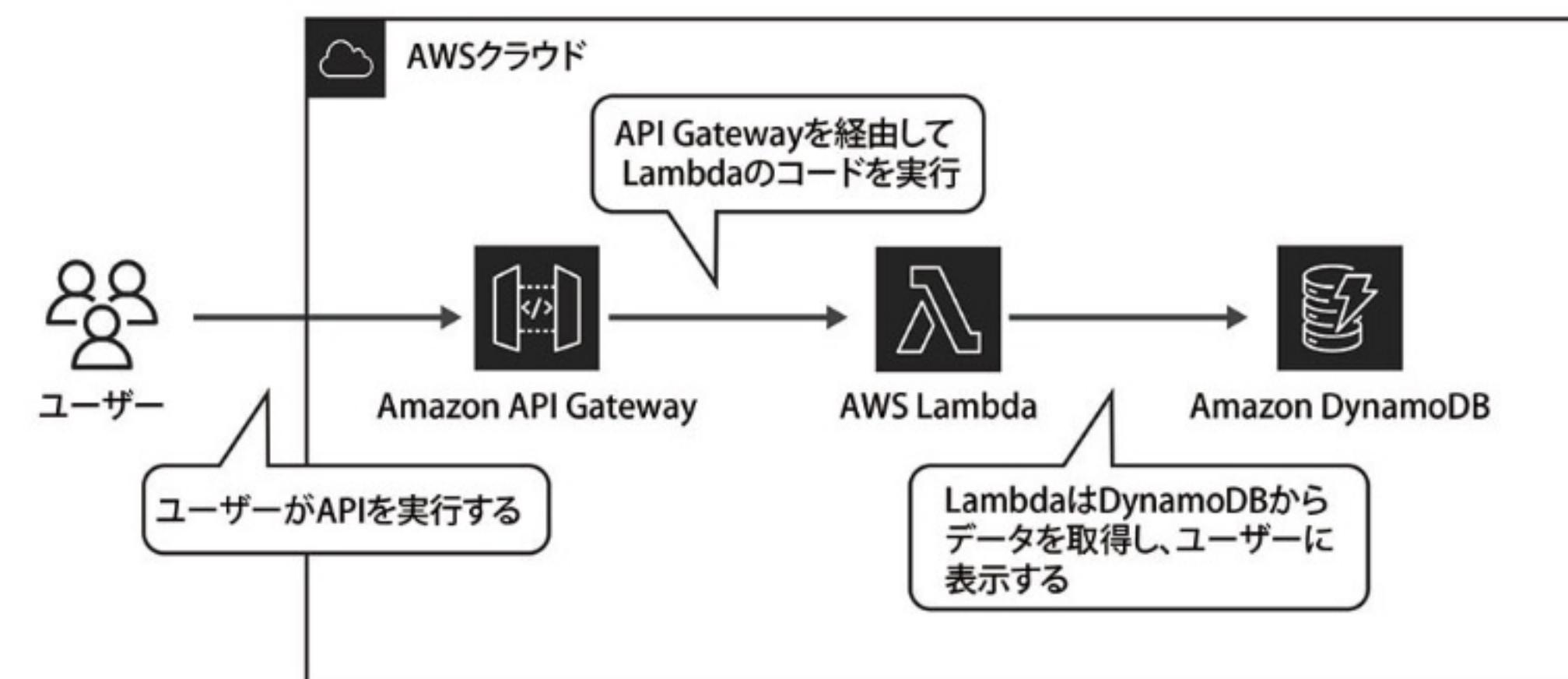
不要になり、より拡張性に優れた構成を実現できます。

ユーザーは、アプリケーションをEC2で実行するのではなく、Amazon API GatewayとAWS Lambdaを組み合わせて実行することで、EC2の可用性を考慮する必要がなくなり、利用頻度に応じたコスト削減が可能になります。

ただし、LambdaはEC2と異なり永続ストレージを持っていないため、データストアが必要な場合はAmazon DynamoDBやAmazon S3を利用したり、他のアプリケーションと連携する場合はAmazon SQSを使用する必要があります。

次の図は、DynamoDBのデータベースをユーザーに表示するWebアプリケーション構成例です。API GatewayやLambda、DynamoDBのいずれも、サーバレスアーキテクチャのサービスを利用しているため、EC2のように複数のAZで冗長化することを考慮する必要がありません。

【DynamoDBのデータベースをユーザーに表示するWebアプリケーション構成例】



3

グローバルサービスにおける高可用性コンピューティング

ここからは、Amazon Route 53やAmazon CloudFrontなどの高可用性を実現するグローバルサービスを組み合わせた高可用システムについて考えてみます。

これらのサービスは、特にグローバルに展開するシステムやDR(災害復旧)サイトをリージョン間で構成する場合に利用します。

●Route 53を利用したWebアプリケーションの構成例

Route 53のネームサーバーは、世界各地に点在するエッジロケーションに存在し、高可用な名前解決を提供します。

Route 53は、ELBやCloudFrontに対して、Zone ApexレコードをAレコードで指定することができるため、非常に効率のよいネットワーク環境を提供します。

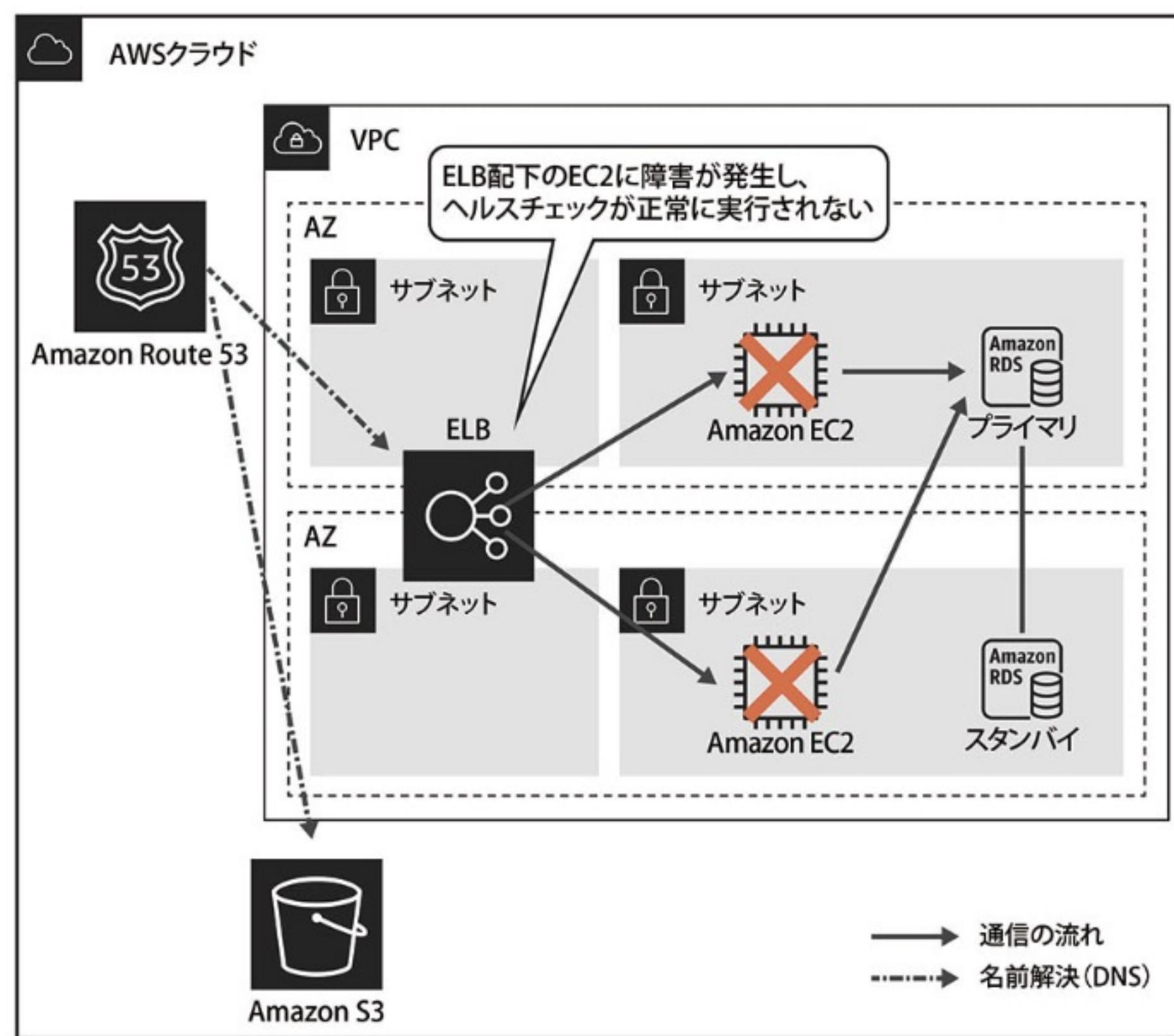
グローバルネットワークを構成するうえで、Route 53を利用するシーンは多くあります。

●DNSフェイルオーバーの構成例

まず最初に、Route 53、ELB、Amazon S3を利用した構成例を示します。ELBを含むWebアプリケーション構成では、通常の名前解決はELBに対して行いますが、EC2インスタンスの障害時には、ELBのヘルスチェックがエラーになります。このような場合、**Route 53のDNSフェイルオーバー**機能を使用することで、DNSのレコードをELBからS3へ変更し、名前解決が適切に行われるようになります。

ELBをマルチAZ配置にすることで、アベイラビリティゾーン(AZ)の障害に対する可用性が向上します。万一、リージョン障害が発生した場合は、S3にSorryページなどのソースファイルを用意しておけば、自動的にSorryページが表示されるようになります。

【DNSフェイルオーバーのイメージ】

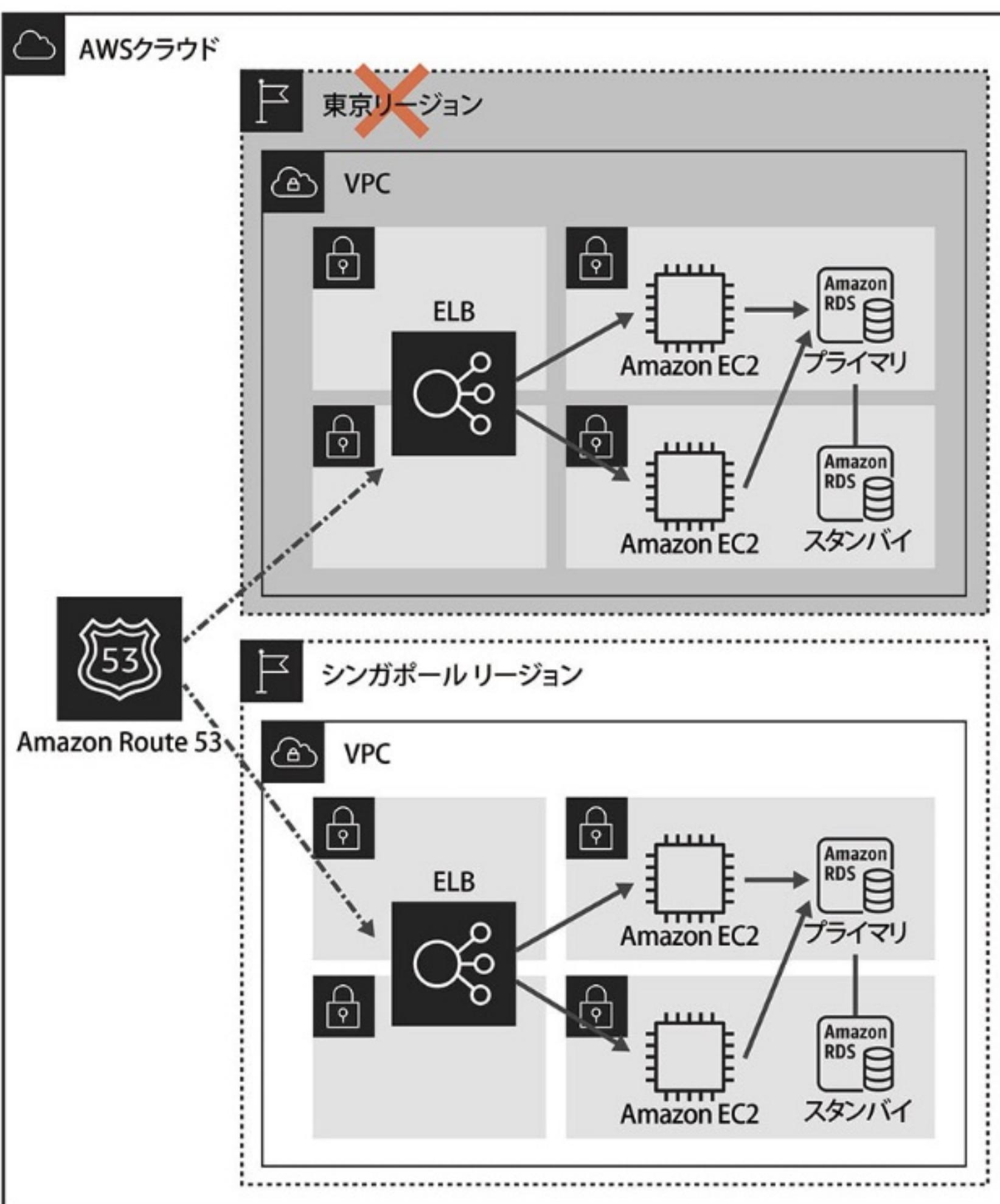


●Route 53によるマルチリージョンの構成例

Route 53のネームサーバーはエッジロケーションに存在するため、リージョンを超えた可用性を実現することもできます。

次の図は、同じVPC構成を複数のリージョンで作成したマルチリージョン構成の例です。

【Route 53を利用したマルチリージョン構成における障害発生時の動作イメージ】



図に示したマルチリージョン構成では、Route 53のDNSフェイルオーバー先を別リージョンに用意しておくことで、リージョン間フェイルオーバーが実現可能です。

これは同じ環境を2カ所に用意するため運用コストが必要になりますが、リージョンレベルの障害やDRサイトとしてシステム稼働の継続性を高めることができます。

また、可用性の観点以外にも、たとえば、Route 53のルーティングポリシーを設定することで、世界中からのアクセスを適切なサーバーに分散できるため、パフォーマンスを向上させることもできます。

なお、Route 53はグローバルリソースであるため、複数のリージョンで共有して利用することができますが、ELBはVPC内のリソースであるため、リージョンごとに用意する必要があります。

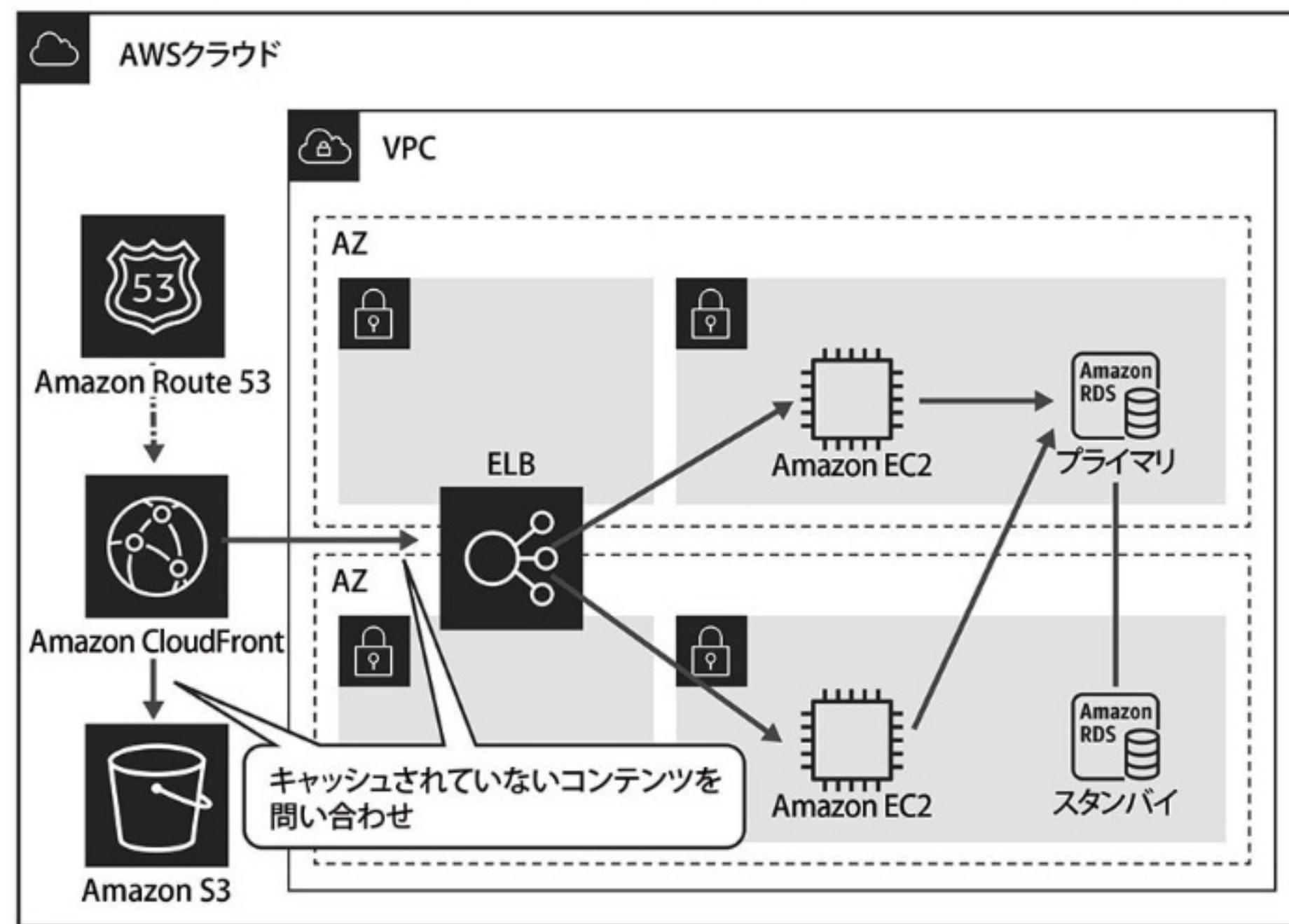
●CloudFrontを利用したWebアプリケーションの構成例

これまで、Webサービスのフロントエンドとして、ELBやAmazon EC2を使用していました。そこで、Route 53のような高可用なDNSを利用しつつ、突発的なアクセス集中によるシステムダウンや障害発生時のリカバリ作業自体を極力回避できるアーキテクチャの構成を考えてみます。ここでは、グローバルで高可用なエッジサービスであるCloudFrontを利用した構成を紹介します。

CloudFrontのエッジロケーションからエンドユーザーへWebサービスを配信することで、フロントエンドの高可用性を維持しつつ、バックエンドのELBやEC2の負荷や障害の影響を軽減することができます。この構成では、CloudFrontは次のように動作します。

- ① CloudFrontがリクエストを受け付け、CloudFrontにコンテンツがキャッシュされていればそのコンテンツを返し、なければバックエンドのオリジン(コンテンツ元のサーバーやサービス)へリクエストをルーティングする
- ② オリiginは複数登録でき、ルーティング先はパスパターンによって指定できる。たとえば、静的コンテンツ(例:jpg)ならAmazon S3、動的コンテンツ(例:mp4)ならELBにルーティングするように構成できる

【CloudFrontのイメージ】



図【CloudFrontのイメージ】で示したAmazon CloudFrontを利用した構成例では、複数のAWSサービスを組み合わせることで单一障害点を排除し、システム全体で高可用を実現しています。この構成におけるポイントは次のとおりです。

- ・Route 53のDNSによる名前解決
- ・CloudFrontのキャッシュサーバーと効率的なコンテンツ配信の仕組み
- ・ELBのマルチAZ配置
- ・EC2インスタンスおよびRDSデータベースインスタンスの冗長構成



演習問題

1 あるアプリケーションは、1台のAmazon EC2と1台のAmazon RDSで構成されています。このアプリケーションの可用性を高め、かつ停止時間を短縮する構成は、次のうちどれですか。

- A EC2を別のAZに追加で構築する
- B EC2とRDSを別のAZに追加で構築する
- C EC2を別のAZに追加で構築し、RDSをマルチAZ配置にする
- D EC2をRDSとは別のAZで再構築する

2 動的コンテンツを扱う3層のWebアプリケーションは、東京リージョンでフロントにELB、アプリケーション実行にAmazon EC2、データストアにAmazon RDSを利用しています。EC2とRDSはいずれもマルチAZ構成で動いていますが、リージョン障害を危惧した経営陣から、シンガポールリージョンにDRサイトを構築するよう指示されました。通常は東京リージョンを使用しながら、大規模障害の発生時のみシンガポールリージョンへ自動切り替えできる適切なサービスの組み合わせは、次のうちどれですか。

- A 東京リージョンのELBとシンガポールリージョンのELBに対し、Route 53の加重ルーティングを設定する
- B 東京リージョンのELBとシンガポールリージョンのELBに対し、Route 53のフェイルオーバールーティングを設定する
- C 東京リージョンとシンガポールリージョンにCloudFrontを構築し、オリジンにS3を設定する
- D 東京リージョンとシンガポールリージョンにCloudFrontを構築し、それぞれのリージョンだけがアクセスできる地域制限を設定する

3 ある会社では、EC2インスタンスで実行されるAPIベースのアプリケーションが稼働しており、そのアプリケーションはAmazon DynamoDBにデータを保存しています。ソリューションアーキテクトは、ここ最近、アプリケーションが中断することを発見したため、

アーキテクチャの変更を検討しています。アプリケーションの可用性を向上させるためには、次のうちどの構成が適切ですか。

- A アプリケーションをWebサイトホスティング機能を有効化したS3上で実行する
- B EC2インスタンスの前にAPI Gatewayを配置し、API GatewayからEC2インスタンスのアプリケーションを実行する
- C アプリケーションをAPI GatewayとLambdaに変更する
- D DynamoDBをS3に変更する



解答

1

C

Aは、EC2を別のAZに追加で構築することで、EC2を冗長化することはできますが、RDSが冗長化できていないため単一障害点となり、高可用性を実現する構成にはなりません。

Bは、EC2とRDSを別のAZに追加で構築することで、EC2とRDSを冗長化できますが、RDSが2台になり、1台に障害が発生した場合はエンドポイントの切り替え作業が発生するため、停止時間が長くなる可能性があります。

Cは、EC2を別のAZに追加で構築してRDSをマルチAZ配置にすることで、EC2とRDSを冗長化でき、RDSに障害が発生してもマルチAZにより自動で切り替えることができます。

Dは、EC2をRDSとは別のAZで再構築しても、どちらも単一障害点になることには変わりなく、高可用性を実現する構成にはなりません。したがって、Cが正解です。

2

B

Aは、Route 53の加重ルーティングを100:0のように設定することで、リージョンの接続先を指定することができますが、大規模障害の発生時には加重の変更が必要になります。

Bは、Route 53のフェイルオーバールーティングを設定することで、通常は東京リージョンに接続し、東京リージョン内のリソースのヘル

チェックが停止した場合に、自動でシンガポールリージョンへ切り替えることができます。

Cは、CloudFrontのオリジンにS3を設定しても、動的コンテンツのWebアプリケーションを扱うことはできません。

Dは、CloudFrontの地域制限を設定すると、その地域内からしかシステムにアクセスできなくなるため、日本国内からシンガポールリージョンを利用することができなくなります。

したがって、Bが正解です。

3

C

API GatewayとLambdaのサーバーレスアーキテクチャに変更することで、EC2より可用性を向上させることができます。

Aは、S3のWebサイトホスティングは静的コンテンツの提供を行っているため、APIアプリケーションの実行はできません。

Bは、API GatewayからEC2インスタンスのアプリケーションを実行したとしても、EC2で発生する可能性がある障害点を排除することはできません。

Cは、API GatewayとLambdaのサーバーレスアーキテクチャに変更することで、EC2より可用性を向上させることができます。

Dは、DynamoDBをS3に変更しても、EC2で発生する可能性がある障害点を排除することはできません。

したがって、Cが正解です。

3-4

ストレージにおける高可用性の実現

AWSでは複数のストレージサービスが提供されており、データの特性やシステムの要件に応じたストレージサービスを選択することができます。

本節では、ストレージサービスにおける可用性の実現について説明します。

1

ストレージサービス

高可用性を実現するためのストレージサービスには、次のようなサービスがあります。

●Amazon Elastic Block Store (EBS)

Amazon EBSは、永続可能なブロックストレージサービスです。

EBSはアベイラビリティゾーン(AZ)単位で作成され、AZ内で自動的に複製されるため、単一のディスク障害については考慮する必要がありません。ただし、AZ障害時にはデータが消失する可能性があるため、スナップショットを適宜バックアップとして取得するケースがよくあります。

●インスタンスストア(エフェメラルディスク)

インスタンスストアは、無料で利用できる高パフォーマンスなストレージサービスです。ただし、揮発性のディスクであるため、EC2インスタンス停止時にデータは消失してしまいます。高パフォーマンスのストレージに可用性を求める場合は、Amazon ElastiCacheを使うなど、別の手段を検討する必要があります。

●Amazon Elastic File System (EFS)

Amazon EFSは、スケーラブルな共有ストレージサービスです。

EFSは、自動で複数のAZで冗長化されるため、ユーザー側で可用性を考慮する必要はありませんが、EBSのようにスナップショットを取得することはできません。EFSのバックアップは、AWS Backupと連携して行います。

●Amazon FSx

Amazon FSxは、マネージド型のストレージサービスです。

マルチAZによる高可用性だけでなく、S3へのバックアップを行うことで、データの高耐久性も実現することができます。

●Amazon Simple Storage Service (S3)

Amazon S3は、高耐久・大容量のオブジェクトストレージサービスです。

S3は、デフォルトでは3カ所のAZで自動的に複製されるため、データの耐久性は高いものの、可用性の観点ではSLAが99.99%とされています。

●Amazon S3 Glacier

Amazon S3 Glacierはあくまでもアーカイブを目的としているため、データへアクセスするためには時間を要します。

データの取り出しには、次の図に示すオプションがあります。

【Glacierのデータ取り出しのオプション】

オプション	説明
迅速(Expedited)	1~5分でアーカイブデータの取り出しが可能 通常は「オンデマンド」と呼ばれる、成功する保証がないタイプで実行される。「プロビジョニング」タイプでは、取り出しのリクエストはすぐに処理されるが、費用が高くなる。
標準(Standard)	3~5時間でアーカイブデータの取り出しが可能
大容量(Bulk)	5~12時間でアーカイブデータの取り出しが可能

2

コンピューティングサービスにおけるストレージの選択

AWSで、より可用性の高いシステムを構築するために、さまざまなストレージサービスを利用します。

以下、コンピューティングサービスで利用するストレージについて説明します。

●EBSとインスタンスストアの違い

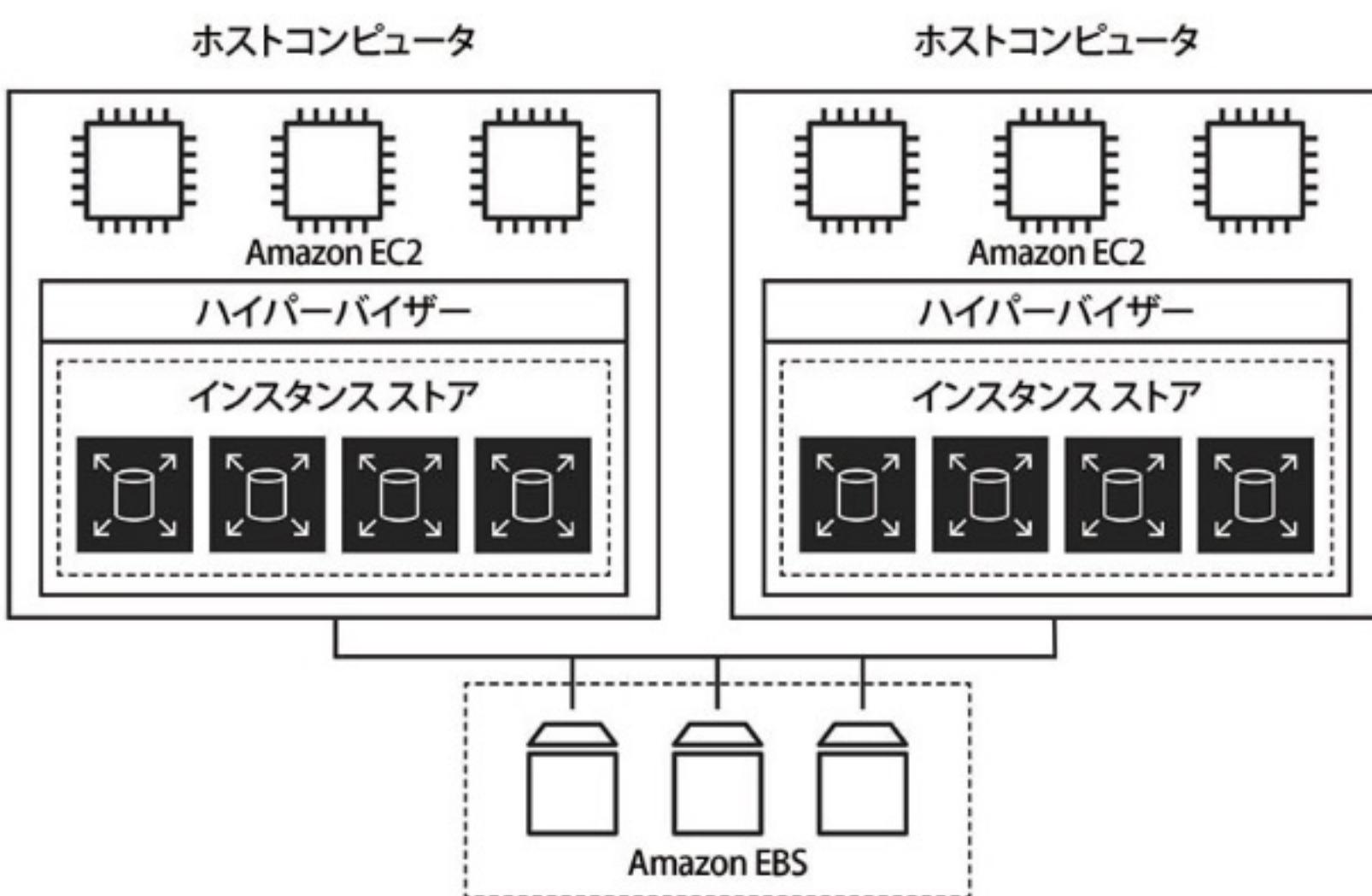
Amazon EBSとインスタンスストアはどちらもAmazon EC2で利用できますが、アプリケーションやデータ特性に応じて適切に選択する必要があります。

次の表にそれぞれの特性の比較をまとめます。

【EBSとインスタンスストアの比較】

	Amazon EBS	インスタンスストア
接続	EC2インスタンスとネットワーク接続	ホストコンピュータのストレージを利用
利用方法	ボリューム作成後にEC2にアタッチ	インスタンス起動時にEC2にアタッチ
データ永続性	永続化可能	永続化不可
耐久性	単一のAZ内で冗長化	冗長化なし
拡張性	ボリューム拡張可能	ボリューム拡張不可
パフォーマンス	高速	非常に高速
料金体系	ギガバイトあたりの従量課金	無料

【EBSとインスタンスストアのイメージ】



前表の【EBSとインスタンスストアの比較】に示したとおり、EC2インスタンスを利用する場合は、EBSを採用するケースが多くなります。

インスタンスストアはデータの永続化ができないため、可用性や耐久性が求められるストレージとしては不向きですが、ネットワーク接続型のEBSに比べて高いパフォーマンスを発揮できるため、高速処理が要求されるシステムで利用すると効果的です。

●EBSとインスタンスストアの利用手段

EBSとインスタンスストアは、どちらもOSのルート領域としても使用可能です。

EBSをOSのルート領域として利用したEC2インスタンスは**EBS-Backedインスタンス**、インスタンスストアをOSのルート領域として利用したEC2インスタンスは**Instance Store-Backedインスタンス**と呼ばれます。

このうちInstance Store-Backedインスタンスは、停止するとデータが消失してしまうため、インスタンスの停止ができません。

●S3をコンピューティングサービスとして利用

Amazon S3は非常に耐久性の高いオブジェクトストレージサービスですが、**静的コンテンツのWebサイトホスティング**という機能を使用することで、コンピューティングサービスとしても利用できます。

その利用手順も非常に簡単で、S3バケット内に必要なソースコードを配置し、静的コンテンツのWebサイトホスティングの機能を有効化するだけです。

Amazon EC2でWebサービスを提供するのに比べて、次のような利点があります。

- ・スケールする必要がなく、アクセス集中に強い
- ・運用に費やすコストを減らすことができる

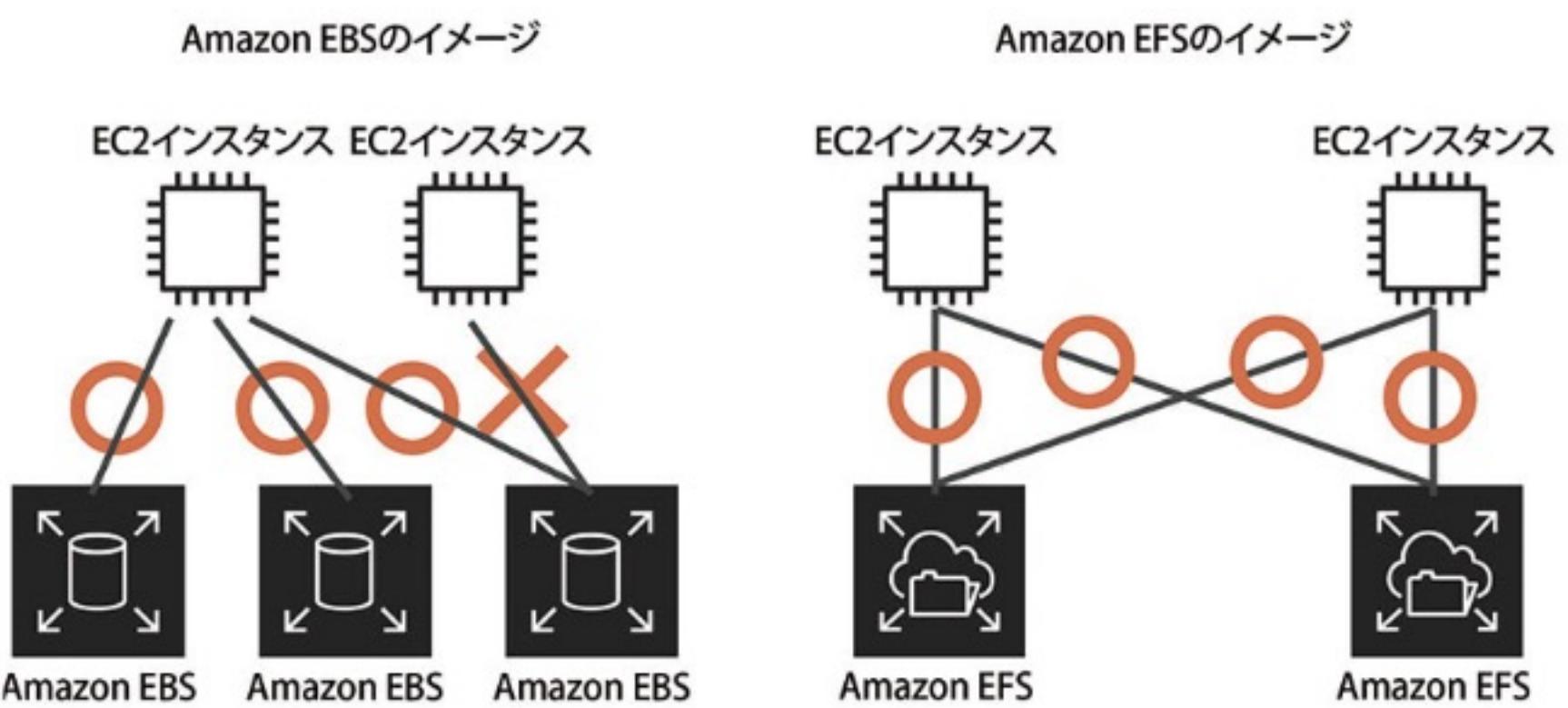
ただし、あくまでも静的なコンテンツだけに利用できるサービスであるため、動的コンテンツを含むWebサービスでは利用できません。

●EBSとEFSの違い

EC2で利用できるストレージサービスとして、Amazon EBSのほかにAmazon EFSがあります。

EBSは原則1台のEC2インスタンスと接続してそのインスタンスでしか利用できませんが、EFSは複数のEC2インスタンスと接続して利用することができます。

【EBSとEFSのイメージ】



Amazon EFSは、Amazon EBSに比べてコストが高くなるものの、可用性や拡張性に優れています。また、EFSはEBSでは不可能であった複数EC2インスタンスとの共有ディスクとして利用できるため、データベースサーバーやファイルサーバーのストレージとして利用されています。

●EBSを複数のインスタンスで共有する方法

前述したとおり、EBSは原則1台のEC2インスタンスで利用しますが、マルチアタッチを使用することで複数のEC2インスタンスと共有することができます。

ただし、次のような制約があるため、可用性が要求される共有ストレージには、複数のAZで冗長化されているEFSを利用しましょう。(2022年12月時点)

- ・同じAZ内にあるNitroベースのEC2インスタンスであること
- ・EBSのボリュームタイプがプロビジョンドIOPS(io1、io2)であること
- ・マルチアタッチできるEC2インスタンスは最大16台まで

●EFSとFSxの違い

では、共有ストレージサービスとして利用されるFSxは、同じ共有ストレージサービスであるEFSと、どのような違いがあるのでしょうか。

どちらも複数のAmazon EC2と接続できますが、EFSはNFS(Network File System)で、FSx for WindowsはSMB(Server Message Block)でアクセスします。

いずれもマルチAZ配置が可能なため、可用性に大きな違いはありません。

【EBS、EFS、FSxの比較】

	Amazon EBS	Amazon EFS	Amazon FSx for Windows
接続	EC2インスタンスとネットワーク接続	EC2インスタンスとネットワーク接続	EC2インスタンスとネットワーク接続
接続可能台数	原則1台のEC2インスタンス	複数のEC2インスタンス	複数のEC2インスタンス
利用方法	EC2にアタッチ	NFSプロトコルで接続	SMBプロトコルで接続
データ永続性	永続化可能	永続化可能	永続化可能
耐久性	単一のAZ内で冗長化	複数のAZで冗長化	複数のAZで冗長化(オプションで選択)
拡張性	手動でボリューム拡張可能	自動でボリューム拡張可能	手動でボリューム拡張可能
パフォーマンス	高速 EBS最適化、プロビジョンドIOPSによる高速化が可能	高速 最大I/Oパフォーマンスマードによる高速化が可能	高速 スループットキャパシティレベルによる高速化が可能
データ暗号化	任意で暗号化可能	任意で暗号化可能	自動で暗号化
月額ストレージ料金 [東京リージョン、2022年10月時点]	0.096USD/GB (gp2)	0.36USD/GB (標準)	0.276USD/GB (SSDマルチAZ)



Amazon EBS、Amazon EFS、Amazon FSxのそれぞれの違いを覚えておきましょう。特にEFSとFSxは利用するプロトコルで違うため、問題文ではどのプロトコルやOSで利用しているのかに注目することで、適切なストレージがわかるケースがよくあります。

3 各ストレージのバックアップ

AWSには、信頼性を向上するためのサービスが多数用意されていますが、それぞれで障害が発生する可能性は否定できません。したがって、ユーザー側でデータをバックアップする仕組みを検討する必要があります。

ここからは、各サービスにおけるバックアップ手法について説明します。

●EBSのバックアップとリストア

Amazon EBSのバックアップには、AWSで標準提供されている**スナップショット**機能を使用します。

通常、EBSはAZ内で自動的に複製されているため、单一のディスク障害が起

きてもユーザーは復旧作業を行う必要はありませんが、AZ障害時にはスナップショットから復旧する必要があります。

EBSスナップショットは、バックエンドでAmazon S3を使用して保存されているため、非常に高い耐久性を備えています。また、バックアップデータは差分で取得されるため安価に利用できます。

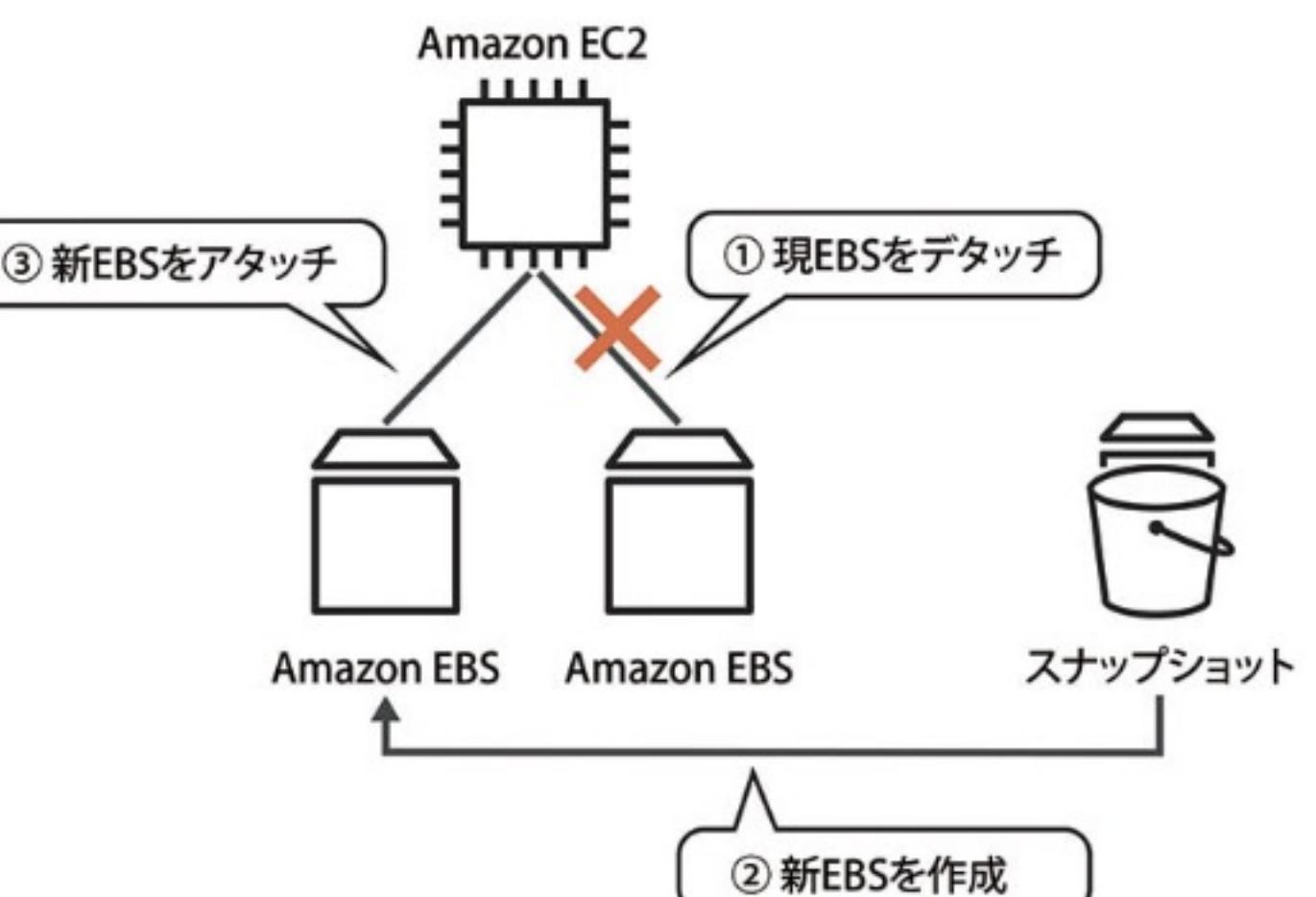
EBSスナップショットは、システムを停止せずにオンラインで取得することも可能ですが、データの整合性を求められるシステムではI/Oを停止してバックアップします。

復旧時は、スナップショットからEBSボリュームを作成することになるため、ディスク作成が完了するまでは利用することができません。

EBSスナップショットはリージョンごとに保管されていますが、EBSボリュームはリストア時にAZを指定する必要があるので、EBSボリュームを利用したいEC2インスタンスが配置されているサブネットのAZを、事前に確認しておきましょう。

また、EBSスナップショットを別リージョンのDRサイトなどで利用する場合は、リージョン間でコピーする必要があります。

【EBSの復旧イメージ】



EBSスナップショットには、取得開始時点のデータが保管されます。取得に際してEC2インスタンスを停止する必要はありませんが、取得開始から取得完了までの間に変更があった場合は、スナップショットに反映されません。

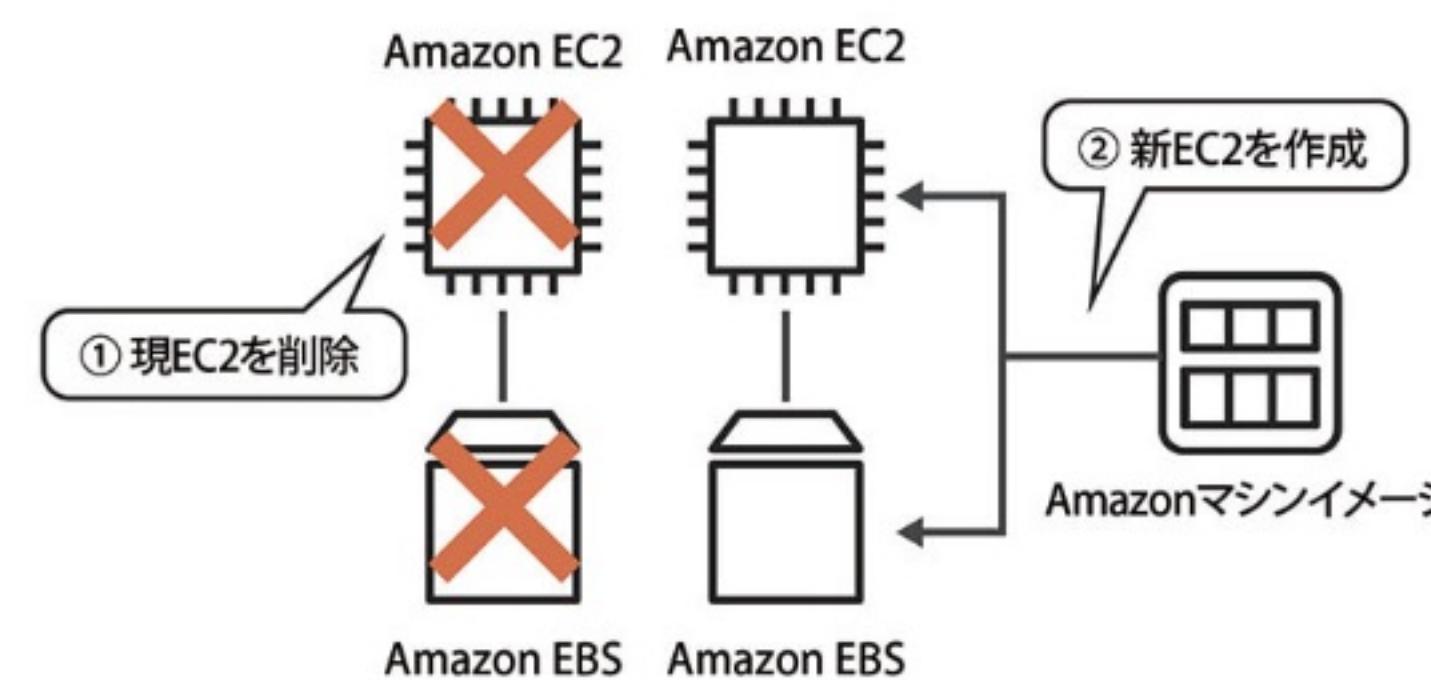
●EC2のバックアップとリストア

EC2インスタンスはEBSを使うことが多いため、EBSスナップショットを取ることでAmazon EC2のデータをバックアップできますが、ほとんどの場合はAMI(Amazon Machine Image)を使用してバックアップされています。

AMIの実体は、EBSスナップショットとEC2の構成情報を組み合わせたデータで、AMIからEC2を復元することで、マウントされたドライブの情報やアプリケーション構成などをそのまま再現できます。

AMIはリージョンごとに保管されているため、別リージョンでDRサイトを構築している場合は、リージョン間でAMIをコピーしておく必要があります。

【AMIによる復旧イメージ】



AMIは共有したり、公開することができます。AWSが提供しているAMIを使用するのが一般的ですが、AWS Marketplaceにはベンダーからも多数のAMIが提供されています。

転送しますが、帯域やデータ量がDirect Connectで許容できる場合は、Direct Connect経由で転送するケースがほとんどです。

また、DataSyncはNFSやSMBなどのファイル共有プロトコルをサポートし、S3以外の移行先にも対応していることが大きなメリットです。

●SnowballによるAWSへのデータ移行

大量データをオンプレミス環境からAWSへ移行する際に、ネットワーク経由で移行する場合は次のような課題があります。

- ・移行が完了するまでに時間を要する
- ・高額なデータ転送料が必要になる
- ・Direct Connectの場合は稼働している他システムと帯域を共有しなければならない

移行時間の制約やコスト最適化の観点で見ると、大量データの移行にはAWS Snowballが適しています。

AWSから配送されたSnowballの筐体に移行データを保存し、筐体をAWSへ返送すると、AWSが移行データをS3にコピーしてくれます。

たとえば、移行のために使えるネットワーク帯域が限られていて、かつ1カ月以内にテラバイト級のデータを移行したいケースなどに有効です。

4

AWSへのデータ移行

●DataSyncによるAWSへのデータ移行

オンプレミス環境のファイルサーバーから、Amazon EFSやAmazon S3上のファイルシステムにデータを効率的に転送したい場合、AWS DataSyncを利用することで効率的なデータ移行を行うことができます。

DataSyncは、通常インターネットまたはAWS Direct Connect経由でデータ



演習問題

1

ある企業では、Windows共有ファイルストレージを必要とする大規模なサーバーをオンプレミス環境で運用しています。この企業は、このサーバーをAWSに移行したいと考えており、さまざまなストレージオプションを検討しています。ストレージは、高い可用性を備え、Active Directoryと統合できる必要があります。これらの要件を満たすストレージは次のうちどれですか。

- A Amazon EFSストレージを構築し、認証用にActive Directoryドメインを設定する
- B ファイルサーバー用とバックアップ用の2つのEBSを構築し、Active DirectoryをインストールしたEC2インスタンスにアタッチする
- C S3バケットを作成し、Active DirectoryをインストールしたEC2インスタンスにアタッチする
- D FSx for Windows File Serverを構築し、認証用にActive Directoryドメインを設定する

2

ソリューションアーキテクトは、オンプレミス環境のアプリケーションをAWSへ移行することを検討しています。アプリケーションは、数十ギガバイトから数百テラバイトまで、さまざまな出力ファイルを生成します。また、アプリケーションデータは、標準のファイルシステム構造に保存する必要があります。ソリューションアーキテクトは、容量が自動的に拡張され、可用性が高く、運用上のオーバーヘッドが最小限で済むソリューションを望んでいます。これらの要件を満たすソリューションは、次のうちどれですか。

- A アプリケーションをAmazon ECSに移行して、ストレージにAmazon S3を使用する
- B アプリケーションをAmazon ECSに移行して、ストレージにAmazon EBSを使用する
- C マルチAZに展開したAuto ScalingグループのAmazon EC2インスタンスにアプリケーションを移行し、ストレージにAmazon EFSを使用する

- D マルチAZに展開したAuto ScalingグループのAmazon EC2インスタンスにアプリケーションを移行し、ストレージにAmazon EBSを使用する



解答

1

D

Aは、EFSは高可用性の共有ストレージとして利用できますが、標準でActive Directory認証機能を備えていません。
 Bは、EBSは同一のAZで動作するため、同じEC2インスタンスで利用すると、AZ障害発生時には利用できなくなるため、高可用ストレージとしては適切ではありません。
 Cは、S3は高可用性のストレージとして利用できますが、標準でEC2インスタンスにアタッチして運用することができません。
 Dは、FSx for Windows File ServerはマルチAZに対応しているため高い可用性で運用することができ、Active Directoryとの連携が可能です。
 したがって、Dが正解です。

2

C

Aは、S3は実質容量に制限がなく高い可用性を備えていますが、標準のファイルシステムで利用することができません。
 Bは、EBSは標準のファイルシステムを利用することができますが、自動拡張されず、AZ障害発生時には利用できなくなるため、高可用ストレージとしては適切ではありません。
 Cは、EFSはNFSを利用して利用でき、かつ容量が自動で拡張され、マルチAZで動作することができます。
 Dは、EBSは標準のファイルシステムを利用することができますが、自動拡張されず、AZ障害発生時には利用できなくなるので、高可用ストレージとしては適切ではありません。
 したがって、Cが正解です。

3-5

データベースにおける高可用性の実現

AWSでは、複数のマネージドデータベースサービスを提供しており、データの種別や用途に応じた適切なサービスを選択することができます。

本節では、データベースサービスにおける可用性の実現について説明します。

1

データベースサービスの可用性向上策

ここでは、各データベースで高可用性を実現するための方法について説明します。

●Amazon Relational Database Service (RDS)

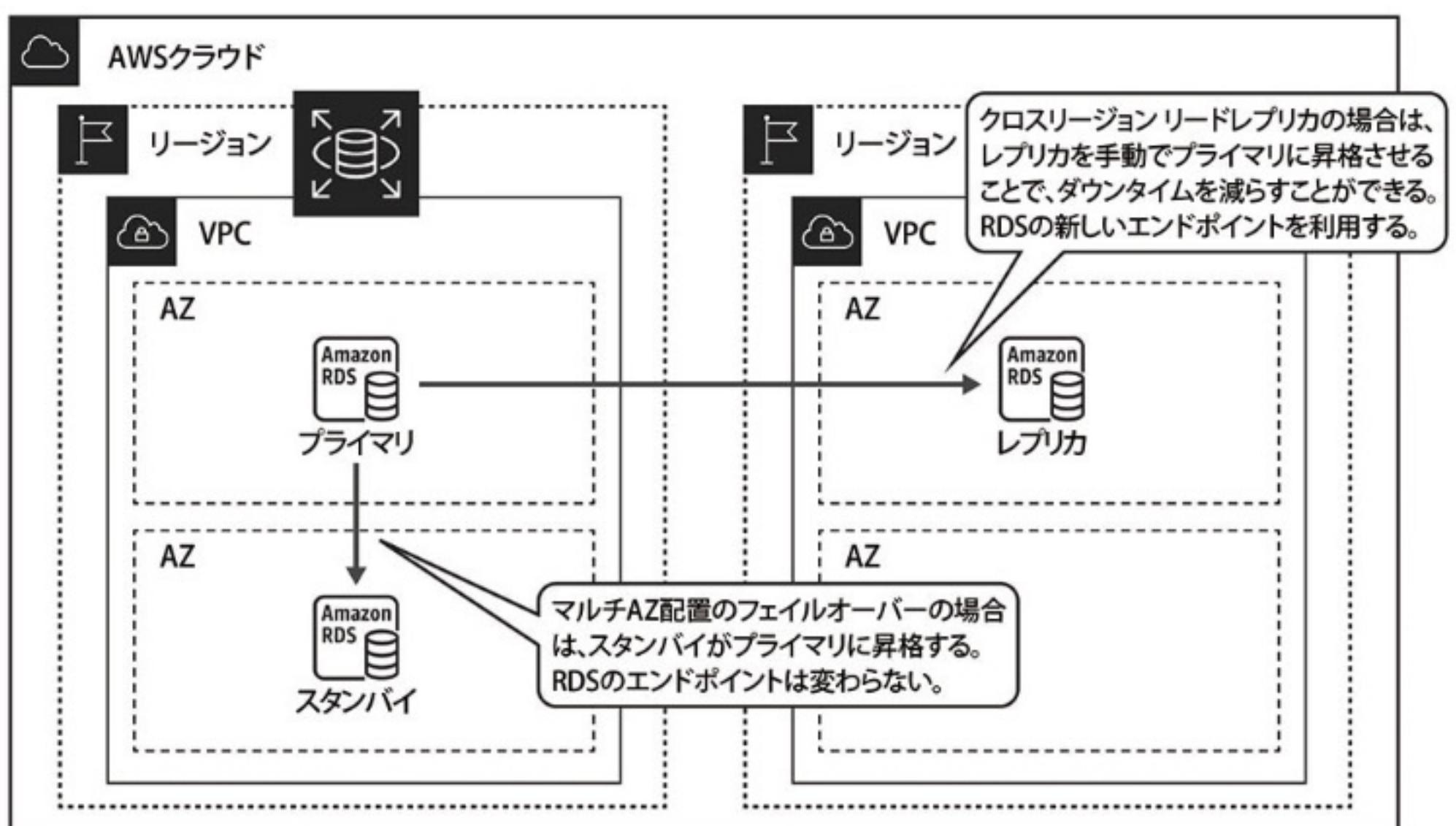
Amazon RDSは、マネージドサービス型のリレーショナルデータベースサービスです。

RDSで高可用性を実現するには、マルチAZ配置が不可欠です。プライマリのRDSデータベースインスタンスに障害が発生した場合は、自動的にスタンバイへのフェイルオーバーを行いますが、RDSのエンドポイントを変更することなくシームレスにスタンバイに切り替えることができるため、バックアップからリストアする方法と比べてダウンタイムを大幅に減らすことができます。

DRサイトを構築する場合は、マルチAZでは対応できないため、クロスリージョンリードレプリカを作成するか、またはバックアップをリージョン間転送しておく必要があります。

クロスリージョンリードレプリカを作成しておけば、レプリカをプライマリに昇格させることで、障害発生の際には迅速に切り替えることができます。

【RDSのマルチAZとクロスリージョンリードレプリカの動作イメージ】



バックアップから復元する場合は、通常のリストアを行うか、またはポイントインタイムリカバリで復元します。この場合、データベースインスタンスを最初から構築することになるため、マルチAZやクロスリージョンリードレプリカよりも復旧に時間を要しますが、コストは抑えることができます。

詳細については、「1-6 データベースサービス」を参照してください。



Amazon RDSの可用性を求められた場合は、マルチAZ配置になっていることを必ず確認しましょう。



Amazon RDSでは、マルチAZによるフェイルオーバーをより高速に行うことができるマルチAZ DB Clusterという機能が提供されるようになりました。

通常のマルチAZでは、スタンバイ側のインスタンスへの切り替え処理のためフェイルオーバー処理に時間を要していました。これに対してマルチAZ DB Clusterでは、あらかじめWriterとは別のAZにアクティブなReaderのインスタンスを配置し、フェイルオーバー時はReaderをWriterに素早く昇格させることで、フェイルオーバーの高速化を図っています。

詳細については以下のページを参照してください。

https://docs.aws.amazon.com/ja_jp/AmazonRDS/latest/UserGuide/multi-az-db-clusters-concepts.html

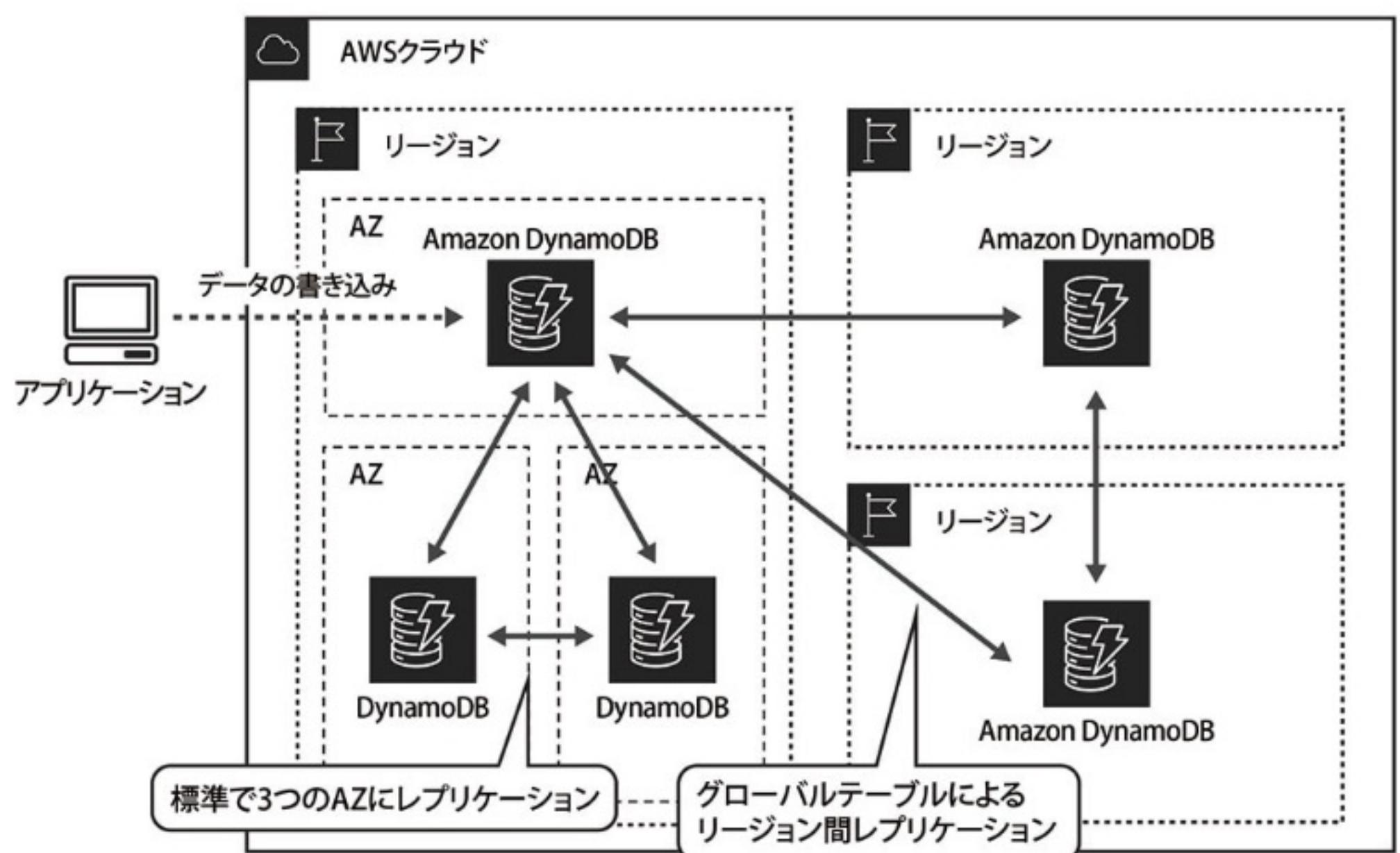
● Amazon DynamoDB

Amazon DynamoDBは、マネージド型のNoSQLデータベースサービスです。データは標準で自動的に3ヵ所のAZに保存され、ストレージも必要に応じて自動拡張するため、非常に高可用なデータベースとして利用できます。

また、DynamoDBは、グローバルテーブルを使用することで、アプリケーションの可用性と高いパフォーマンスをグローバルで確保することができます。

ある特定のリージョンで障害が発生した場合でも、別リージョンのレプリカテーブルに対してデータを読み書きできるため、他リージョンで同じテーブルを利用できます。また、障害が発生したリージョンが復旧すると、テーブル間で保留していた伝播を再開します。

【DynamoDBの冗長性イメージ】



DynamoDBでは、オンデマンドバックアップまたはポイントインタイムリカバリによるバックアップを実行することができます。

通常では、オンデマンドバックアップは別のアカウントまたはリージョンにコピーすることはできないため、そうした操作が必要な場合は、AWS Backupと連携する必要があります。

● オンデマンドバックアップ

テーブル全体をバックアップする方法です。バックアップ中のパフォーマンス低下やレプリケーション先への切り替えなどは発生しません。

● ポイントインタイムリカバリ

テーブルを自動でバックアップする方法です。過去35日の任意の地点までデータを復元することができます。



Amazon DynamoDBのグローバルテーブルを利用してリージョン間の可用性を高めることができます。グローバルアプリケーションなどではパフォーマンス向上の側面もあるので、試験問題ではどのような要件が問われているのか見極めましょう。

●Amazon ElastiCache

Amazon ElastiCacheは、マネージド型のインメモリデータベースで、メモリ上で処理を実行するため、高スループットかつ低レイテンシーな処理を実現できます。

ElastiCacheのインメモリデータベースとして、MemcachedとRedisが使用できます。Redisではプライマリ/レプリカ構成のマルチAZ配置が可能で、この構成により高可用性を実現できます。

プライマリのElastiCacheに障害が発生した場合は、自動的にレプリカにフェイルオーバーを行いますが、Amazon RDSと同様に、わずかなダウンタイムが発生します。



演習問題

1

ある企業では、MySQLデータベースをオンプレミス環境からAWSへ移行したいと考えています。最近、この企業はビジネスに大きな影響を与えるデータベースの停止を経験しました。今後は同様の障害が発生しないように、データの損失を最小限に抑え、障害発生時に自動で切り替えを実行でき、少なくとも2台以上にデータを保存できる信頼性の高いソリューションを検討しています。これらの要件を満たすソリューションは、次のうちどれですか。

- A 3つのアベイラビリティーゾーンにそれぞれRDSインスタンスを構築し、レプリケーションする
- B マルチAZ機能を有効化したRDSインスタンスを構築する
- C RDSインスタンスからEC2インスタンスへデータをレプリケーションするAWS Lambda関数を作成する
- D DynamoDBを構築し、MySQLから移行する



解答

1

B

Aは、レプリケーションされた3つのRDSインスタンスで高可用性を実現することはできますが、それぞれ異なるエンドポイントを持つため、切り替えを手動で行う必要があります。

Bは、マルチAZのRDSインスタンスでは2台以上でデータを保存でき、障害発生時には自動でフェイルオーバーします。

Cは、EC2インスタンス上でデータベースを稼働した場合は、EC2インスタンスの運用管理が必要になり、また障害発生時の切り替えは手動で行う必要があります。

Dは、DynamoDBはNoSQLのデータベースサービスであり、MySQLデータベースを扱うことはできません。

したがって、Bが正解です。

Column



AWS主催の大規模カンファレンス

AWSでは、世界中のユーザーが参加する「AWS re:Invent」と呼ばれる大規模な年次カンファレンスを、米国ラスベガスで開催しています。

re:Inventでは新サービスや新機能、事例紹介、ハンズオンなど数多くのコンテンツが用意されており、会場には世界中から約6万人以上の関係者が集まります(2019年開催時)。

筆者自身が2019年に実際に参加してみた感想ですが、単なるカンファレンス兼イベントというよりもお祭りに近い雰囲気で、参加者の圧倒的な熱量に大きな刺激を受けました。

わざわざ米国まで行くことに疑問を抱く人も居るとは思いますが、さまざまな国の人々との交流や、日本では体験できないスケールのイベントに参加でき、AWSの凄さを身に染みて感じられます。また、ブースツアーやJapan Nightなどの日本人向けイベントも用意されており、現地で日本からの参加者と交流することもできます。

re:Inventは、個人で申し込むほかに旅行会社のツアーでも参加でき、カンファレンス以外にもエンタテイメントや観光のオプショナルツアーを楽しむことができます。

一方、国内では、AWS Summitと呼ばれるカンファレンスが毎年開催されています。過去には幕張メッセやグランフロント大阪など、大規模なイベント会場で開催されていました。規模は本家のre:Inventに及びませんが、多くのセッションやブースが用意され、AWS一色のイベントを楽しむことができます。

AWS re:InventとAWS Summitのどちらにも共通していることですが、「認定者ラウンジ」というAWSの資格保有者だけが利用できる場所や施設が用意されています。このラウンジでは軽食や飲み物が提供されており、作業机やテーブルも用意されているため、会場を歩き回って疲れたらゆっくり休憩することができます。ソリューションアーキテクトに合格したら、立ち寄ってみてはいかがでしょうか。



第4章

AWSにおけるパフォーマンス

4-1 AWSにおけるパフォーマンスの考え方

4-2 ネットワークサービスにおけるパフォーマンス

4-3 コンピューティングサービスにおけるパフォーマンス

4-4 ストレージサービスにおけるパフォーマンス

4-5 データベースサービスにおけるパフォーマンス

4-1

AWSにおけるパフォーマンスの考え方

AWSでは、多くのサービスが提供される一方で、その組み合わせも多岐にわたります。それらのサービスから用途に合ったサービスや機能を選択し組み合わせることで、費用対効果に優れたシステムを構築することができます。

本節では、AWS上でパフォーマンス効率の良いシステムを構築する際の設計原則について説明します。

1

AWSにおけるパフォーマンス効率の設計原則

AWSでは、第1章でその概要を説明したサービス以外にも、数多くのサービスが提供されています。それらのサービスやサービス内で提供される機能は、市場に投入されてから今日まで、多様なユーザーの要件に応えるために開発・改善され続けています。

ユーザーは、各種のサービスや機能を適切に組み合わせることで、要件を満たすシステムを容易に構築できるだけでなく、オンプレミス環境よりも低コストで高パフォーマンスのシステムを構築することもできます。

AWSには、クラウド上でパフォーマンス効率を高めるための、次のような設計原則があります。

●最新技術の導入

従来は、新しい技術を取り入れたくてもスキル不足のために導入を諦めざるを得ないことがありました。現在では、専門知識を備えたクラウド事業者に導入やデプロイを任せることで、新技術を簡単に取り入れてクラウドをより有効に活用することができます。

●グローバルな環境

AWSクラウドでは、世界中のリージョンにシステムを構築することができます。このため、どの地域のユーザーにも、低成本で低レイテンシーのサービスを提供することができます。

●サーバーレスアーキテクチャの使用

AWSでは**サーバーレスアーキテクチャ**を利用することができます。

オンプレミスのような非クラウドの環境では、インフラストラクチャを用意し、プログラムを処理するためのサーバーを導入・構築する必要がありました。

サーバーレスアーキテクチャでは、サーバーをユーザー側で調達する必要がありません。ユーザーはコードをデプロイするだけで、それを呼び出すイベントによってプログラムを実行できます。

サーバーレスアーキテクチャを活用することにより、ユーザーは本来のアプリケーション開発に集中でき、開発コストを抑えることができます。また、サーバーの運用コストも削減できます。

●比較テストの実施

オンプレミス環境では、システム構築時にプロビジョニングを実施し、システムを設計して必要なリソースを調達してからでなければ、パフォーマンスを把握することができませんでした。

AWSでは、システム構築時にインスタンス、メモリ、ストレージなどのリソースを選択しますが、システム運用の段階でもこれらのリソースを変更できます。こうした柔軟性のある運用の特徴によって、異なるタイプのリソースを選択してパフォーマンスを比較することが容易になります。

●適切な技術の利用

AWSは、常に新しい機能やサービスをリリースし続けています。そのため、システムの要件を満たすサービスが複数存在し、選択肢が多岐にわたる場合もあります。ユーザーは、それらのなかから、実現しようとしているシステムの要件に適合する最適な技術を使用する必要があります。

次節からは、最適な技術を適切に選択するために、パフォーマンス効率の高いシステムの構築に利用できるサービスや、システム構築時の考え方について説明します。



演習問題

1 次の選択肢のうち、パフォーマンス効率の設計原則に含まれない説明はどれですか(1つ選択)。

- A 東京リージョンにあるアプリケーションをオレゴンリージョンに展開する
- B アプリケーションをServerless Frameworkを使用してデプロイする
- C AWSでの新システムの構築をクラウド事業者に依頼する
- D マネージドサービスを利用する



解答

1 D

パフォーマンス効率の設計原則の知識を問う問題です。

D以外はパフォーマンス効率の設計原則に含まれる内容です。Dはコスト最適化の設計原則に含まれる内容になります。

4-2

ネットワークサービスにおけるパフォーマンス

AWSでは、世界中に展開されたリソースを上手に活用することで、多くのユーザーに高パフォーマンスのネットワークサービスを提供することができます。

本節では、パフォーマンスの観点から見たネットワークサービスについて説明します。

1

ネットワークにおけるパフォーマンスの考え方

AWSは、全世界に設置されたデータセンターにリソースが配備されているため、グローバルでサービスを利用できることが大きな利点です。しかし、グローバルで利用できるからこそ注意すべきこともあります。それは、サービスを提供するサーバーの設置拠点とユーザーの場所が地理的に離れている場合に、レイテンシーが発生してしまうということです。たとえば、北米のリージョンからログインして東京リージョンのサービスにアクセスすると、大きなレイテンシーが発生することがあります。

また、全世界のユーザーに動画やモバイルアプリケーションなどのコンテンツを配信するサービスにおいて、ユーザーがサービス拠点と地理的に離れたリージョンにアクセスするのでは、常にレイテンシーが発生して実利用に耐えられないでしょう。

そのような問題を解決するために、AWSにはユーザーに最も近い場所からコンテンツを配信する仕組みや、コンテンツをキャッシュする仕組みが用意されています。

次項から、ネットワークレイテンシーの低減に有用なサービスについて説明します。

2

Amazon CloudFront

AWSではCloudFrontというCDNサービスが提供されています。CDN(Content Delivery Network)とは、その名のとおりコンテンツを配信するネットワークです。

CDNと呼ばれる技術が最初に登場したのは1990年代ですが、当時のCDNは、動画ファイルや画像データなどをダウンロードするだけの技術でした。しかし現在では、動画のストリーミング配信や、従来とは比較にならないほどの大容量のコンテンツを高速で配信できるなど、技術とともに機能性が格段に進化しています。

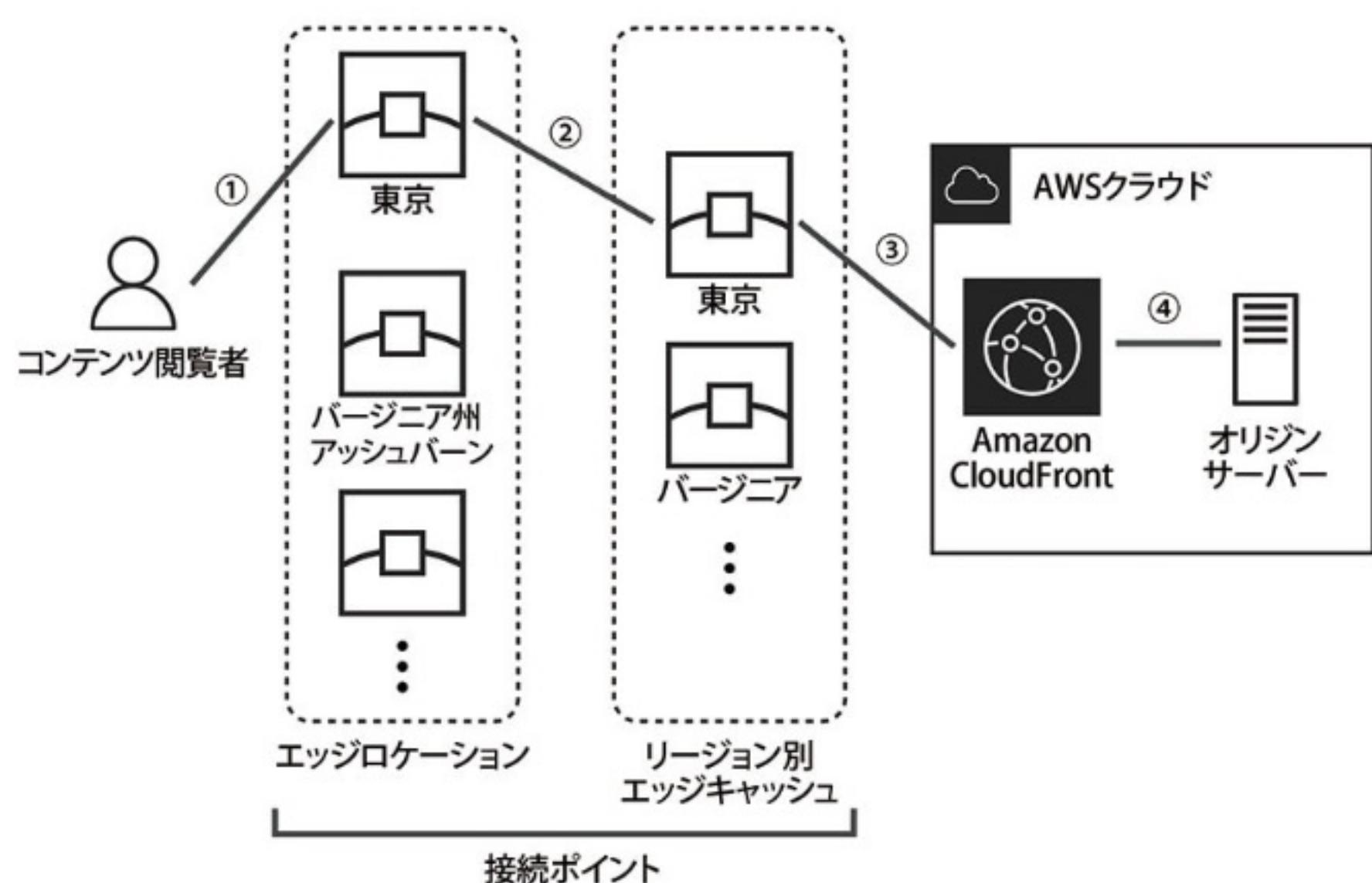
AWSのCDNサービスは配信拠点が世界中に散在しており、ユーザーがCDNにアクセスすると、グローバルに存在する拠点のうち、ユーザーがアクセスした場所に最も近い拠点からコンテンツが配信されます。

現在のAmazon CloudFrontのコンテンツ配信拠点(接続ポイント)は、世界48カ国90都市で410カ所を超えており(2022年12月時点)。接続ポイントは、400を超えるエッジロケーションと13のリージョン別エッジキャッシュで構成されています。

【CloudFrontの接続ポイント】

接続ポイント	説明
エッジロケーション	コンテンツをキャッシュして保持できるサーバー。Amazon CloudFrontの利用時に、コンテンツがキャッシュされている場合は、このサーバーから配信される。
リージョン別エッジキャッシュ	エッジロケーションよりも大容量のデータをキャッシュできるサーバー。エッジロケーションにコンテンツがキャッシュされていない場合は、オリジンサーバーの前に、このサーバーから配信される。

【CloudFrontのイメージ】



CloudFrontを使用したCDNで、ユーザー(閲覧者)がコンテンツにアクセスした場合の基本的な処理は次のとおりです。

- ① 閲覧者がコンテンツにアクセスすると、閲覧者から最も近いエッジロケーションに接続される。エッジロケーションに該当コンテンツのデータがキャッシュされている場合は、そのデータを返す
- ② エッジロケーションにデータがキャッシュされていない場合は、リージョン別エッジキャッシュに取得しにいく。リージョン別エッジキャッシュに該当コンテンツのデータがキャッシュされている場合は、そのデータを返す
- ③ リージョン別エッジキャッシュにデータがキャッシュされていない場合は、CloudFrontにデータを取得しにいく
- ④ CloudFrontは、コンテンツの元データが格納されているオリジンサーバーからデータを取得して返す



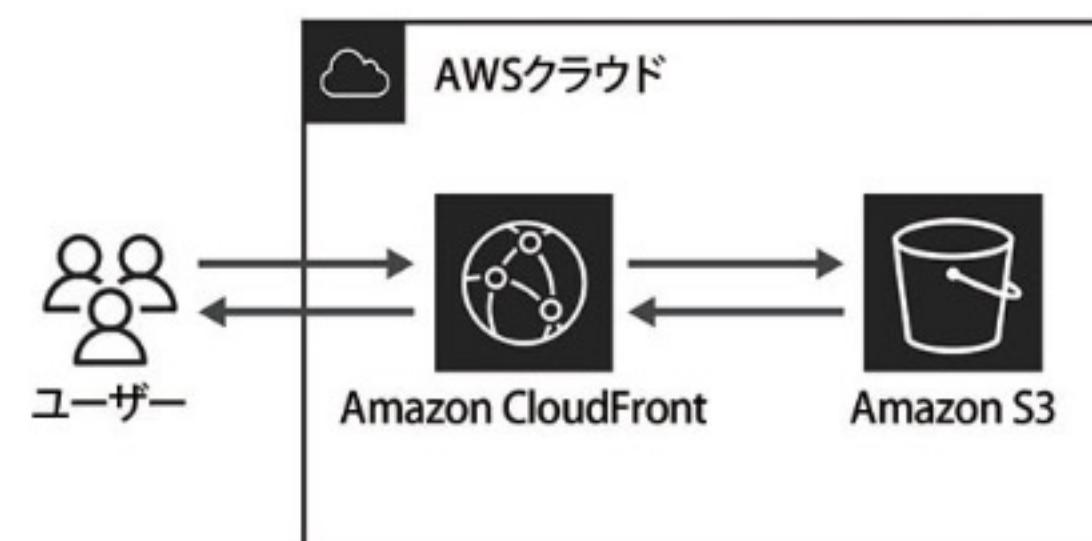
Amazon CloudFrontにはエッジロケーションとリージョン別エッジキャッシュという接続ポイントが存在しますが、CloudFrontを利用したシステム構成図やイメージ図を記載する際は、接続ポイントを記載しないことが一般的です。本書でも、これ以降の説明でCloudFrontの図を記載する場合は、接続ポイントを省略しています。

●CloudFrontのコンテンツ

CloudFrontが配信するのは、主に静的コンテンツと動的コンテンツです。HTMLや画像など、リクエストに影響されずに同じ結果になるコンテンツを静的コンテンツといいます。CloudFrontを利用して静的コンテンツを配信する場合は、一般的にAmazon S3と組み合せます。

S3に格納されたオブジェクトをCloudFront経由でユーザーに配信する場合の構成例を次の図に示します。

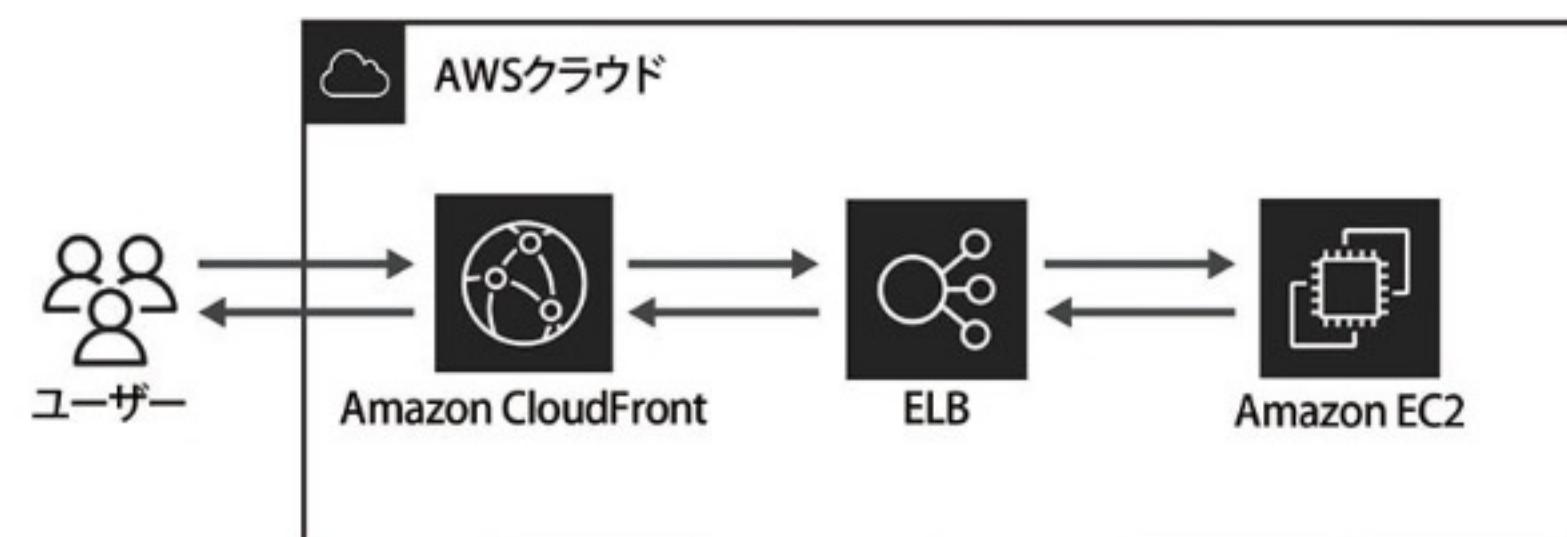
【静的コンテンツを配信するCDNの構成例】



動的コンテンツは、リクエストに応じて動的に生成されるコンテンツを指します。動的コンテンツは、主にコンピューティングサービス(Amazon EC2など)を利用して生成されるため、CloudFrontからロードバランシングサービスのElastic Load Balancing(ELB)を経由して、コンピューティングサービスにアクセスします。

EC2を利用する場合の構成例を次の図に示します。

【動的コンテンツを配信するCDNの構成例】



●CloudFrontで配信するコンテンツの保護

CloudFrontでは、配信するコンテンツを保護するための対策として、次のような方法があります。

- ・ HTTPS接続
- ・ 特定のユーザーや特定地域のユーザーだけがコンテンツを閲覧できるようにするアクセス制限
- ・ 通信中のデータを暗号化するためのフィールドレベル暗号化

フィールドレベル暗号化は、個人情報に関わるような機密性の高いデータのセキュリティを確保するため、CloudFrontに追加された機能です。

通信データのうち、特に機密性の高い一部情報(フィールド)にユーザー自身が固有の暗号化鍵を設定し、その鍵を使ってデータを暗号化します。このフィールド固有の暗号化鍵を使うと、リクエストがオリジンサーバーに転送される前に、HTTPS通信でデータがさらに暗号化され、機密データのセキュリティをより高めることができます。

●CloudFrontでのHTTPS接続

CloudFrontでカスタムドメイン名を使用してHTTPS接続を必須にする場合、設定するSSL/TLS証明書がAWS Certificate Manager(ACM)で管理されている必要があります。

ACMで管理される証明書は、ACMが提供する証明書か、またはACMにインポートされたサードパーティー製の証明書のいずれかになります。



Amazon CloudFrontにおけるHTTPS接続時の証明書管理方法について覚えておきましょう。

●S3コンテンツへのアクセスを制限した配信

Amazon S3とCloudFrontで静的コンテンツを構築する場合の最も簡単な方法は、オリジンサーバーであるS3バケットへのアクセス設定をパブリック(公開)にすることです。これにより、オリジンサーバーへの直接的なアクセスを許可することになり、S3バケットに格納されたコンテンツを配信できるようにな

ります。

しかし、プライベートコンテンツの配信など、S3バケットへの直接的なアクセスを許可したくないケースもあります。このような場合は、**オリジンアクセスアイデンティティ(OAI)**という機能を使用します。OAIを使用することで、CloudFrontのみにS3バケットへのアクセスを許可してコンテンツを取得できるようにし、ユーザーにはCloudFrontにだけアクセスを許可するといった構成になります。

03

試験対策

オリジンアクセスアイデンティティ(OAI)を使用したAmazon S3バケットへのアクセス制限方法を覚えておきましょう。

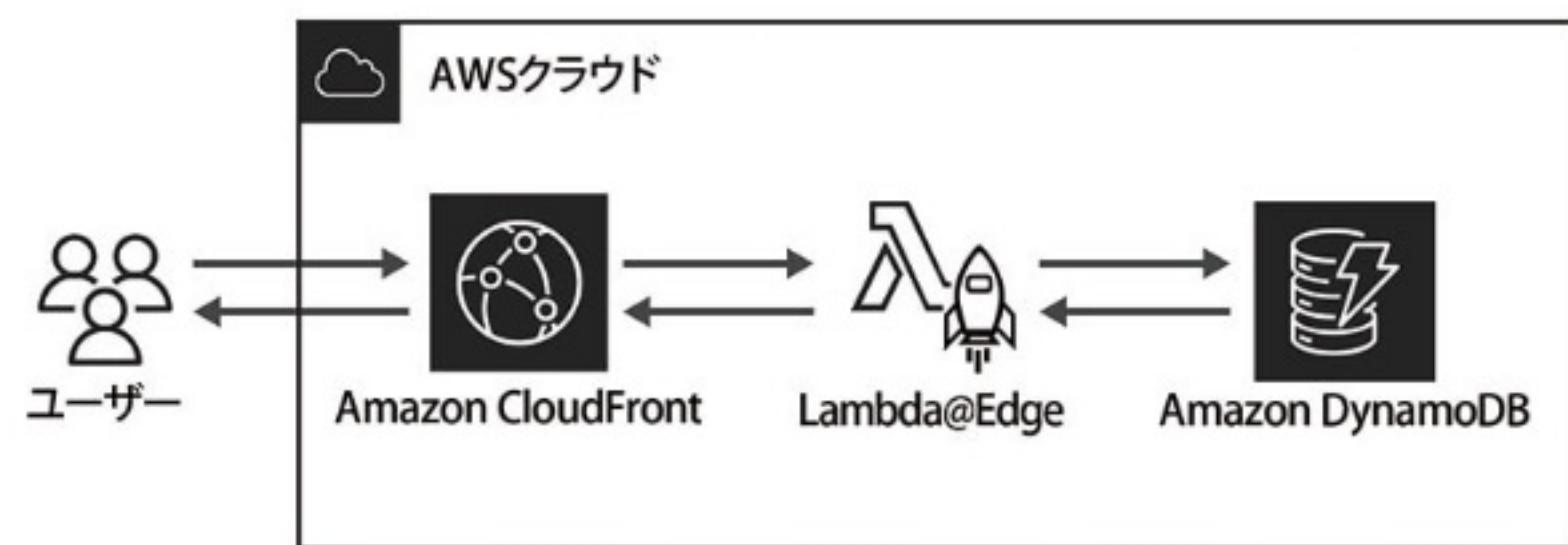
3

Lambda@Edge

Lambda@EdgeはAmazon CloudFrontの機能の1つで、**CloudFrontのエッジロケーション**上でLambdaのプログラムを実行するサービスです。

CloudFrontのCDNによって生成されたイベントをトリガーとして、対応するLambdaのプログラムが実行されます。Lambda関数はユーザーに近いロケーションで実行されるため、通常のLambdaを上回るパフォーマンスと待ち時間の短縮化を実現します。

【Lambda@Edgeを利用して動的コンテンツを配信するCDNの構成例】



4

Amazon Route 53

AWSには、**Amazon Route 53**というDNSサービスが用意されています。DNS(Domain Name System)は、ドメイン名とIPアドレスを紐付けることで名前解決を行うシステムです。

複数のリージョンにAmazon EC2などのコンピューティングサービスが配置されている前提で、パフォーマンスの向上を目的としてRoute 53を利用する場合、レイテンシーが低いリージョンからリクエストを処理する**レイテンシーベースルーティング**という機能を利用します。

たとえば、米国北部とシンガポールにロードバランサーも含めたAWSサービスが配備されており、東京にいるユーザーがWebサービスにアクセスしたときに、米国北部の方が低レイテンシーで処理できる場合は、地理的に近いシンガポールではなく、米国北部に接続されます。



演習問題

1 ある企業がコンシューマー向けのアプリケーションをAWSで開発しており、アプリケーションはAmazon EC2とパブリックサブネット上のELBで構成されています。また、アプリケーションは現在、東京リージョンのみで提供されていますが、近々複数の国での提供開始が予定されています。同社はパフォーマンスとセキュリティを低下させることなく、コストを最小に抑える構成に変更したいと考えています。この要件を満たす構成は、次のうちどれですか。

- A アプリケーションをAmazon S3に移行し、パブリック公開する
- B 東京リージョンのEC2とELBを展開先リージョンでも構築し、Amazon Route 53でルーティングする
- C EC2をAmazon ECSに移行し、Auto Scalingでスケールするように設定する
- D EC2とELBをオリジンサーバーとして、Amazon CloudFrontを使用する

2 ある企業では静的コンテンツを配信するWebアプリを運営しており、オリジンにAmazon S3が設定されたAmazon CloudFrontで複数のリージョンに配信されています。同社ではコンテンツの拡充計画があり、それに合わせて、アクセスするユーザーの地域によって表示するコンテンツを変えられるようにしたいと考えています。これをWebアプリを大きく改修することなく実現するには、次の方法のうちどれが最適ですか。

- A リージョンごとにS3と静的コンテンツを準備して、CloudFront のオリジンとして設定する
- B アクセス元の地域によって表示コンテンツを変更するように、アプリケーションを改修する
- C CloudFrontにLambda@Edgeを設定する
- D CloudFrontで地域ごとのコンテンツを表示するように、リダイレクトを設定する



解答

1 D

Amazon CloudFrontについての知識を問う問題です。「パフォーマンスとセキュリティを低下させることなく」と要件にあるため、AとCは適切ではありません。また、「コストを最小に抑えたい」とあるので、Bの構成ではコストが増加するため不正解です。それらすべての要件を満たす構成はDのため、Dが正解です。

2 C

Amazon CloudFrontのLambda@Edgeについての知識を問う問題です。アクセスするユーザーの地域によってコンテンツを変更するためには、CのLambda@Edgeが適切な選択になります。AとBはアプリケーションの改修または追加が必要になるため、不正解です。また、Dも要件を満たせないため不正解です。

4-3

コンピューティングサービスにおけるパフォーマンス

AWSでは、適切なインスタンスタイプを選択することで、要求に合ったパフォーマンスを実現することができます。

本節では、パフォーマンスの観点から見たコンピューティングサービスの利用方法について説明します。

1

プレイスメントグループ

プレイスメントグループとは、Amazon EC2インスタンスを論理的にグループ化したものです。

EC2インスタンスを同一リージョン内に複数作成した場合、異なるアベイラビリティーゾーン(AZ)にEC2インスタンスを配置することで、可用性と高速通信を同時に実現できますが、プレイスメントグループを利用すると、そのグループ化したEC2インスタンス間での通信が、通常のEC2インスタンス間通信よりもさらに高速化されます。そのため、複数のコンピューティングリソースを一体化して機能させるクラスターコンピューティングを実装するような用途に最適です。

プレイスメントグループには次のような種類があります。

● クラスタープレイスマントグループ

クラスタープレイスマントグループは、単一のAZ内のEC2インスタンスを論理的にグループ化していますが、すべて同じラックに格納されます。

同一ラックに格納することによって、このラック内に配置されたEC2インスタンス間の通信を高速に行うことができます。

したがって、プレイスメントグループは、低レイテンシーかつ高スループットなネットワークが求められるアプリケーションの用途に適しています。

● パーティションプレイスメントグループ

パーティションプレイスメントグループは、EC2インスタンスの各グループ

を「パーティション」と呼ばれる論理的なセグメントで分けたものです。パーティションの単位でラックが分かれしており、各ラックはネットワークと電源をそれぞれ独自に備えています。

そのため、1つのラックでハードウェア障害が発生した場合でも、他のハードウェアは動作を継続することができ、ハードウェア障害によるシステムの停止時間を短縮することができます。

また、パーティションプレイスメントグループは分散処理環境をデプロイできるため、EC2を利用したHadoopなどの大規模な分散ワークロード処理が実現できます。

● スプレッドプレイスメントグループ

スプレッドプレイスメントグループでは、EC2インスタンスがそれぞれ個別にネットワークと電源を備えたラックに配置されます。

このプレイスメントグループは、離れたAZにも配置することができるため、EC2インスタンスが配置されているラックの1つに障害が発生した場合でも、システム全体が障害の影響を受けるリスクを軽減できます。

3つのプレイスメントグループの違いは、次の図に示すとおりです。

【各プレイスメントグループの違い】

種類	EC2インスタンスを配置するラック	特徴
クラスタープレイスマントグループ	同じラックに配置	低レイテンシーかつ高スループットなネットワークが求められるアプリケーションに適している
パーティションプレイスメントグループ	パーティション単位で異なるラックに配置	分散処理に適している
スプレッドプレイスメントグループ	すべてのEC2インスタンスを異なるラックに配置	障害発生時の影響を低減したいシステムに適している



プレイスメントの日本語訳は「配置」です。プレイスメントグループは、Amazon EC2インスタンスを配置するグループと考えると覚えやすいでしょう。

2

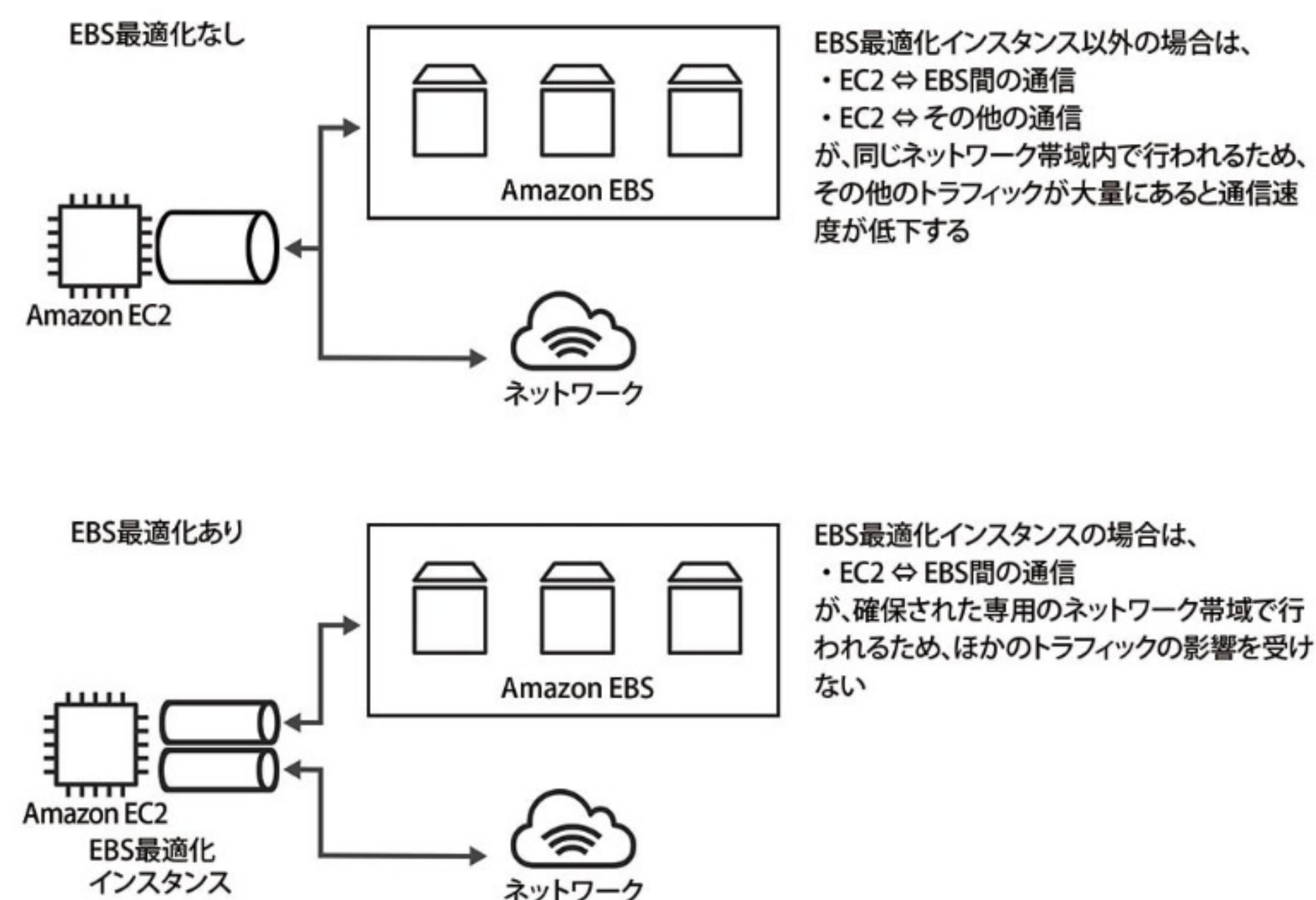
EBS最適化インスタンス

Amazon EBSのパフォーマンスは、そのボリュームタイプによって異なります。EBSのデータを利用するAmazon EC2インスタンス側で、適切なインスタンスタイプを選択することで、そのEBSのパフォーマンスを最大に発揮できるようになります。

また、パフォーマンスを発揮するには、EC2インスタンスとEBSの間の通信も考慮する必要があります。EC2とEBSはネットワーク経由で通信していますが、**EBS最適化**することで、EC2とEBS間の通信専用帯域を確保でき、安定した最適なパフォーマンスを実現できます。

EBS最適化はインスタンスタイプによって、「デフォルトで有効」「デフォルトでは無効だが手動で有効化が可能」「デフォルトでも手動でも有効化不可」の3種類に分けられます。EBSが最適化されていない場合は、ほかのネットワークとの通信が共有され通信速度が低下するので注意しましょう。

【EBS最適化インスタンスの通信イメージ】



EBS最適化インスタンスを利用する場合は、構築するインスタンスタイプがEBS最適化をサポートしているかどうかを確認しておきましょう。EBS最適化インスタンスの詳細については、以下のWebページを参照してください。

https://docs.aws.amazon.com/ja_jp/AWSEC2/latest/UserGuide/ebs-optimized.html

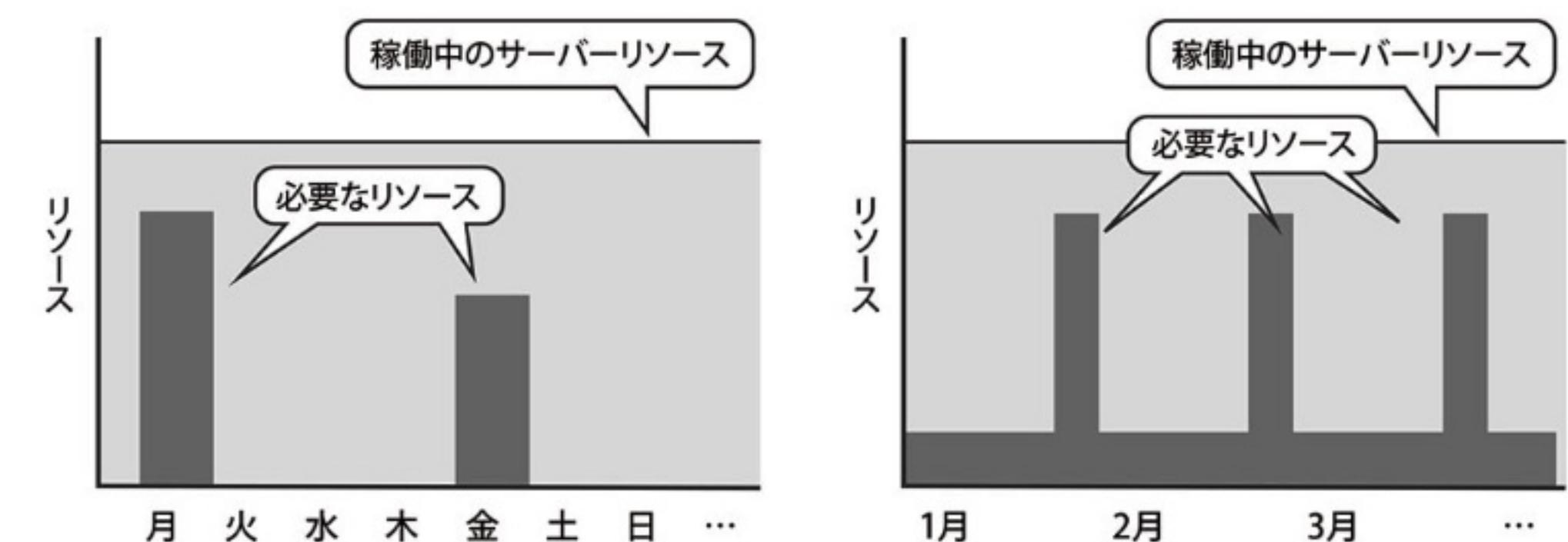
3

Auto Scalingによる最適化の実現

オンプレミス環境のシステム運用に携わるサーバー管理者のなかでも、クラウドネイティブではない管理者は、「サーバーの台数は固定で運用したい。運用中はサーバー台数をあまり増減したくない」と考えるでしょう。

従来のオンプレミス環境では、ワークロードに関係なく仮想マシンを常時起動したままシステムが運用されていました(次の図を参照)。

【従来のサーバー使用状況・サーバーリソース稼働状況のイメージ】



サーバーの使用頻度は高くなりが、常に起動したままの状態にある。ピーク時の消費リソースを予測し、その予測値以上のリソースでサーバーを稼働させている。

処理が集中する毎月末に負荷が高くなり、それ以外の期間はほとんど使用されないことを把握していても、ピーク時の消費リソースを予測し、その予測値以上のリソースでサーバーを稼働させている。

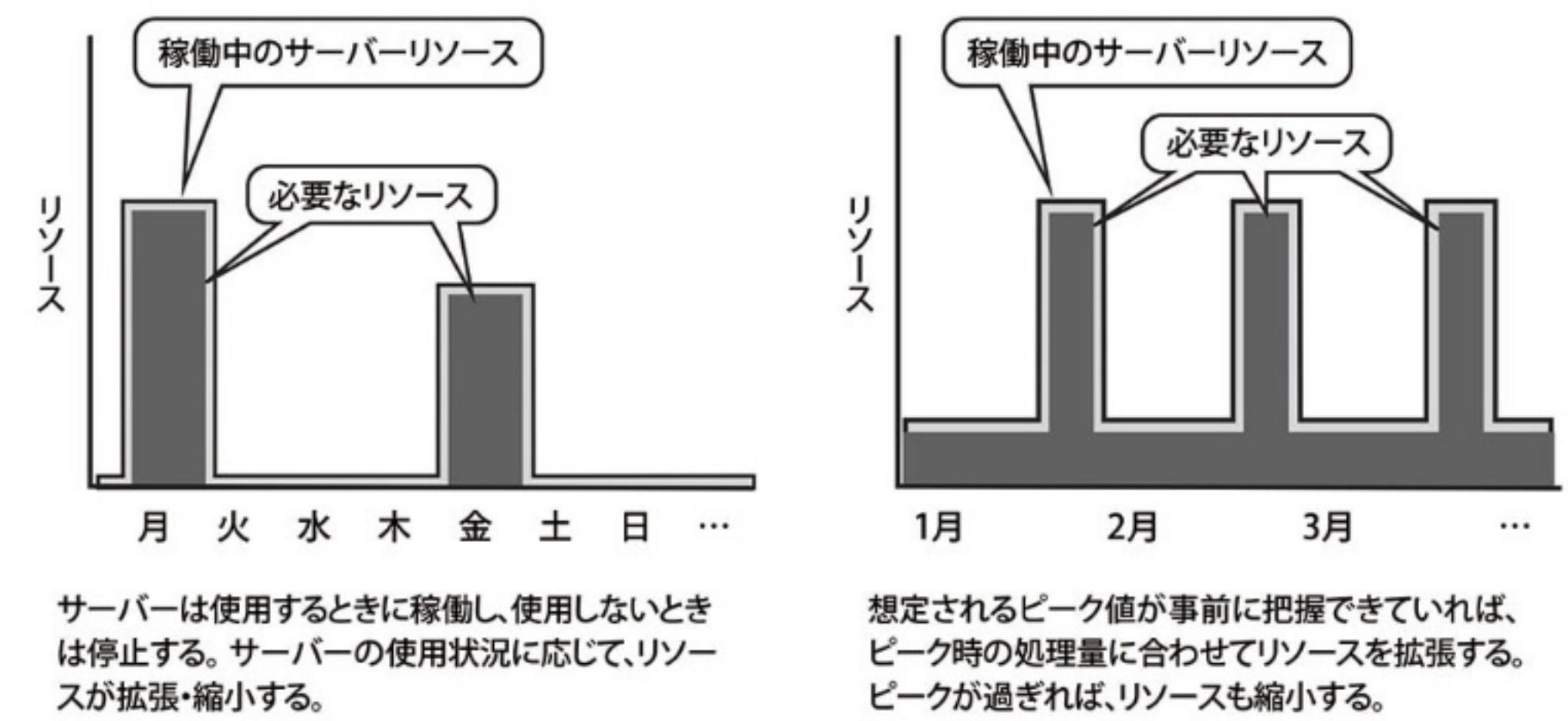
これに対してクラウド環境では、「従量課金制+オンデマンド」でリソースを利用することが基本になります。「負荷に応じて必要な分だけ仮想マシンを起動する」ことが望ましく、コスト効率よく運用するには、そうしたマインドチェンジが必要です。

AWSでは、Auto Scalingを活用することで、サーバーリソースに対する負荷の変化に適切に対応することができます。

サーバーリソースなどをモニタリングするサービスであるAmazon CloudWatchとAuto Scalingを組み合わせることで、システムのパフォーマンスを維持しつつ利用コストを抑えたシステムを構築し運用できます。

たとえば、CloudWatchの監視項目であるCPU使用率の特定メトリクスにしきい値を設定し、そのしきい値を超えたこと(アラート)をトリガーとしてAuto Scalingを起動し、スケールアウト／スケールインを実施するといった使い方です。

【Auto Scalingによるサーバーリソースの最適管理】



Amazon CloudWatchとAuto Scalingを利用すると、パフォーマンスを効率的に管理できることを覚えておきましょう。

4

Lambdaのレイテンシー対策

AWS Lambdaはアプリケーションコードをコンテナ上で実行しますが、コードのデプロイ後に初回実行されるまでは、実行環境のコンテナが起動していません。Lambda関数はコンテナ起動後に実行されるため、処理開始までにレイテンシーが発生します。

一度起動したコンテナは一定時間その状態が維持されるため、この間であれ

ばLambda関数は低レイテンシーで実行できます。一定時間が経過すると、コンテナは停止します。この状態でLambda関数を実行すると、コンテナが再起動する時間が必要になるため、再び処理を開始するまでのレイテンシーが発生します。

また、Lambda関数はVPCに配置することができますが、VPCエンドポイントでElastic Network Interface(ENI)を確立する必要があるため、VPC外にLambda関数を配置する場合と比較するとレイテンシーが大きくなります。

従来からレイテンシーの低減策として、CloudWatch Eventsなどを用いて定期的にLambda関数を実行し、ホットスタンバイの状態を維持する方法が利用されていました。しかし、この方法はスケールアップの場合や別のアベイラビリティーゾーン(AZ)で関数が実行された場合などには対応しておらず、完全な対策ではありませんでした。

現在では、Lambdaに追加された「プロビジョニングされた同時実行の設定」を利用して対応することができます。この機能では、Provisioned Concurrency(同時実行数)を設定すると、設定した数だけコンテナがウォームスタンバイの状態を維持し続けるため、低レイテンシーな処理が可能になります。

ただし、Provisioned Concurrencyの設定を有効にすると、そのウォームスタンバイの時間分だけ使用料金が課されるため、速度とコストを加味して対策を決定する必要があります。また、Lambda関数の\$LATESTバージョンでは使えないなどの制約もあるため、設定する際には注意が必要です。



AWS Lambdaのレイテンシー対策として、re:Invent 2022でSnapStart機能が発表されました。これは簡単にいえば、事前にLambda関数の実行環境をスナップショットで保持しておくことで、起動に要する時間を短縮できるという機能です。試験には出題されませんが、実際にLambdaを使用する場合は覚えておくとよいでしょう。Lambda SnapStartについては、以下のWebページを参照してください。
<https://aws.amazon.com/jp/blogs/news/new-accelerate-your-lambda-functions-with-lambda-snapstart/>



演習問題

1

ある企業では、複数のAmazon EC2上でアプリケーションが稼働しており、アプリケーションの仕様でEC2間の通信が大量に発生しています。また、アプリケーションはEC2内のファイルの読み書きを頻繁に行っています。あるときシステム担当者は、アプリケーションの処理時間がいつもより遅いことに気が付きました。原因を調べたところ、ファイルを読み書きする処理に想定よりも時間がかかっているようです。最小限の作業でこの問題を解決するためにシステム担当者が採るべき方法は、次のうちどれですか。

- A EC2間の通信量が減少するようにアプリケーションを改修する
- B EC2を追加で構築し、ELBで負荷分散する
- C EC2インスタンスのEBS最適化を有効にする
- D ファイルの格納先をAmazon EFSに変更する

2

システム担当者は、Amazon EC2インスタンスで稼働しているアプリケーションの負荷分散を行うために、Auto Scalingの導入を検討しています。その際は、CPU使用率を判断材料としてスケーリングを実施したいと考えています。EC2インスタンスのサーバーリソースをモニタリングするサービスとして、適切なサービスを1つ選びなさい。

- A Amazon CloudFront
- B Amazon EventBridge
- C AWS CloudTrail
- D Amazon CloudWatch

3

ある企業では、社内向けのアプリケーションを、Amazon EC2とAmazon RDSで実装しています。最近は、アプリケーションの利用者(ユーザー)が増加してきたため、パフォーマンスを低下させずにリソースを最適化するように構成を変更したいと考えています。具体的には、CPUとネットワークの使用率に基づいて、EC2の台数を

増減できるようにすることを希望しています。この要件を満たすために使用できるサービスは、次のうちどれですか。(2つ選択)

- A Auto Scaling
- B AWS Step Functions
- C Amazon Simple Notification Service
- D Amazon CloudWatch
- E AWS CloudFormation

A

解答

1

C

EBS最適化についての知識を問う問題です。
問題文から、EC2間で発生している大量の通信が、EC2とファイルが格納されているEBSとのネットワークに影響を及ぼしていることが原因だとわかります。「最小限の作業で」と問題文にあるため、Cが正解になります。EBS最適化を有効にすることでEBS専用のネットワーク帯域が確保されるため、EC2間通信のネットワークと切り離して影響を受けないようにすることができます。C以外の選択肢はEBSのネットワークが改善されないか、最小限の作業ではないため、不正解です。

2

D

EC2のリソースを収集してモニタリングするサービスについての知識を問う問題です。
Aはコンテンツ配信サービス、Bはアプリケーションから発生したイベントデータなどをやり取りするイベントバスサービス、CはAWSで実行されたAPIログなどを管理するイベント記録サービスであるため、不正解です。DのCloudWatchを使用すると、EC2からサーバーリソースを取得することができます。そのため、Dが正解になります。

3 A、D

クラウド特有の、負荷に応じて必要最小限のリソースを利用する構成についての知識を問う問題です。

CのAmazon SNSはフルマネージドのメッセージングサービスで、EのAWS CloudFormationはIaC(Infrastructure as Code)でのプロビジョニング自動化サービスであるため、要件を満たしません。BのAWS Step Functionsはステートフルなワークフローサービスで、サーバーの増減には対応していないため、不正解です。DのAmazon CloudWatchはEC2からCPUやネットワーク使用率の収集が可能で、AのAuto Scalingはそれらの使用率に応じてEC2の増減が可能なサービスです。したがって、**AとD**が正解になります。

4-4

ストレージサービスにおけるパフォーマンス

AWSの代表的なストレージサービスであるAmazon EBSを適切に利用することで、コストを抑えながら、要求されるパフォーマンスを実現できます。

本節では、EBSのボリュームタイプとユースケースについて説明します。

1

Amazon Elastic Block Store (EBS)

Amazon EBSは、ハイパーバイザー上のOSやアプリケーション、データを配置する場所として利用されるストレージです。EBSはAmazon EC2インスタンスにアタッチされることで、デバイスとしてインスタンスにマウントされます。また、EBSはAmazon RDSのストレージボリュームとしても利用されます。

以下に、EBSの**ボリュームタイプ**について説明します。

●EBSボリュームタイプのスループットとIOPS

Amazon EBSでは、性能やコストが異なる5種類のボリュームタイプが用意されており、より性能の高いボリュームタイプを選択することで高パフォーマンスを実現できます。

たとえば、Amazon EC2やAmazon RDSを利用し、データ格納先としてEBSを使用している場合、I/Oパフォーマンスが低いために処理が遅いようであれば、ボリュームタイプを変更することでパフォーマンスを改善できことがあります。ただし、パフォーマンスの高いボリュームタイプは応分のコストが必要になります。

HDD(Hard Disk Drive)のほうがSSD(Solid State Drive)よりも低コストで、ストレージのなかでもIOPS(I/O Operations Per Second)が低いタイプが低成本になります。

次に、各EBSボリュームタイプの特徴をまとめます。

● 汎用 SSD(General Purpose SSD : gp2 および gp3)

価格とパフォーマンスのバランスがよいボリュームタイプです。

汎用SSDではgp2とgp3の2つのボリュームタイプが用意されています。gp2とgp3とで大きく異なる点は、ボリュームあたりの最大スループットです。gp2はボリュームサイズ次第で最大250MiB/秒まで増加しますが、gp3ではプロビジョンドIOPS次第で最大1,000MiB/秒まで増加します。

ユースケースとして、仮想デスクトップ、低レイテンシーなアプリケーション、開発・テスト環境などがあげられます。

● プロビジョンド IOPS SSD (PIOPS SSD : io1 および io2)

汎用SSDでは対応できないミッションクリティカルなシステムに適した、低レイテンシーかつ高スループットなボリュームタイプです。EBS最適化インスタンスでの起動が推奨されています。io1とio2はボリュームの耐久性に違いがあり、それぞれの年間故障率(AFR : Annual Failure Rate)はio1が0.2%以下、io2は0.001%以下となっています。

ユースケースとして、高スループットが必要なアプリケーション、大規模データベース環境(RDBMSやNoSQL)があげられます。

● スループット最適化 HDD (st1)

スループットが高く低成本なHDDです。アクセス頻度が高い用途に適しています。

ユースケースとして、ビッグデータやデータウェアハウス、ログ処理があげられます。

● コールド HDD (sc1)

アクセス頻度が低い用途に適したHDDです。

アクセス頻度の低い大量データを保存するストレージに使用されます。

● マグネティック

旧世代のボリュームタイプです。アクセス頻度が低い用途に適した旧世代のHDDです。

アクセス頻度の低いワークロード向けに使用されます。

【EBSボリュームタイプのパフォーマンス特性】(2022年12月時点)

種類	ソリッドステートドライブ(SSD)				ハードディスクドライブ(HDD)	
	汎用SSD(gp2)	汎用SSD(gp3)	PIOPS SSD(io1)	PIOPS SSD(io2)	スループット最適化HDD(st1)	コールドHDD(sc1)
ボリュームサイズ	1GiB~16TiB				4GiB~16TiB	
ボリュームあたりの最大IOPS	16,000 ^{※1}				64,000 ^{※3}	500 250
ボリュームあたりの最大スループット	250MiB/秒 ^{※1}	1,000MiB/秒 ^{※2}		1,000MiB/秒 ^{※3}	500MiB/秒	250MiB/秒

※1 スループットの制限は、ボリュームサイズに応じて128 MiB/秒～250 MiB/秒の間で変動する

※2 スループットの制限は、プロビジョニングされたIOPSに応じて125 MiB/秒～1,000 MiB/秒の間で変動する

※3 通常は最大32,000 IOPSおよび500 MiB/秒を保証。AWS Nitro System上でのみ64,000 IOPS、1,000 MiB/秒となる

【旧世代EBSボリュームタイプのパフォーマンス特性】(2022年12月時点)

種類	ハードディスクドライブ(HDD)
ボリュームタイプ	マグネット
ボリュームサイズ	1GiB~1TB
ボリュームあたりの最大IOPS	40~200
ボリュームあたりの最大スループット	40~90MiB/秒



Amazon EBSの各ボリュームタイプの特徴とユースケースを押さえおきましょう。



Amazon EBSのボリュームタイプ選択時には、IOPS重視であればSSDタイプ、スループット重視で低成本で利用したい場合はHDDタイプを選択するとよいでしょう。



マグネットは、旧世代のボリュームタイプです。旧世代のボリュームより高パフォーマンスまたはパフォーマンスの安定性が必要な場合は、現行のボリュームタイプの使用を検討しましょう。