AWS ホワイトペーパー

ゲーム業界レンズ

ゲーム業界レンズ: AWS ホワイトペーパー

Table of Contents

ゲーム業界レンズ – AWS ウェルアーキテクトフレームワーク

公開日: 2025 年 12 月 9 日 (ドキュメントの改訂)

AWS Well-Architected フレームワークは、クラウドアーキテクトがアプリケーションとワークロードのための安全で高性能、耐障害性、効率的なインフラストラクチャを構築するのに役立ちます。Well-Architected は、運用上の優秀性、セキュリティ、信頼性、パフォーマンス効率、コスト最適化、持続可能性の 6 つの柱に基づいて、顧客と AWS パートナーがアーキテクチャを評価し、リスクを修復し、ビジネス価値をもたらす設計を実装するための一貫したアプローチを提供します。

このレンズでは、 でゲームワークロードを設計、デプロイ、設計する方法に焦点を当てます AWS クラウド。Well-Architected フレームワークの適用に役立つコンポーネントを定義し、一般的なワークロードシナリオを検討し、設計原則の概要を説明します。AWS Well-Architected Framework ホワイトペーパーのベストプラクティスと質問を考慮して、アーキテクチャの設計を開始することをお勧めします。このドキュメントでは、ゲーム業界のお客様に補足的なベストプラクティスを提供します。

このレンズは、世界中のゲーム業界のデベロッパーやパブリッシャーと連携した経験に基づいて、クラウドでゲームを構築および運用する独自の特性に対処することを目的としたベストプラクティスを指定します。グローバルプレイヤーの需要の変動に合わせてコストが最適化され、スケーラブルになるように、環境を設計および運用する方法に関するガイダンスを提供します。このレンズは、ゲームインフラストラクチャを保護し、パフォーマンスを調整してポジティブなプレイヤーエクスペリエンスを提供するためのガイダンスも提供します。

このドキュメントは、最高技術責任者 (CTOs)、ゲームスタジオのテクニカルディレクター、アーキテクト、デベロッパー、運用チームメンバーなどのテクノロジー担当者を対象としています。このドキュメントを読むと、ゲームのアーキテクチャを設計する際に使用する AWS ベストプラクティスと戦略を理解できます。

レンズの可用性

カスタムレンズは、 が提供するベストプラクティスガイダンスを拡張します AWS Well-Architected Tool。 AWS WA Tool では、独自のカスタムレンズを作成したり、他のユーザーが作成した共有レンズを使用したりできます。

ゲームワークロードの確認を開始するには、AWS パブリック Well-Architected カスタムレンズ GitHub リポジトリ AWS Well-Architected Tool から Games Industry Lens をダウンロードして にインポートします。

設計原則

AWS Well-Architected フレームワークは、ゲームワークロードのクラウドでの優れた設計を容易にするために、次の一般的な設計原則を特定します。

- プレイヤーの動作と使用パターンを理解してゲームを進化させ、プレイヤーを保護する: ゲームを継続的に改善し、プレイヤーエクスペリエンスを効果的に管理するには、プレイヤーがゲーム自体と他のプレイヤーとやり取りする方法を可視化することが重要です。これにより、ゲームを改善し、コストを管理し、プレイヤーエクスペリエンスにリスクをもたらす不正な使用アクティビティをモニタリングして対応する方法を理解できます。
- ゲームオペレーションを簡素化し、開発速度を向上させるテクノロジーを使用する: スピードを向上させ、プレイヤーに新機能と改善を提供する運用オーバーヘッドを削減できるテクノロジーの採用を優先します。ゲームはヒット駆動型であり、プレイヤーが考慮すべき選択肢が多数あるため、ゲームを成功させるには、迅速に動き、変化に適応することが重要です。独自のソフトウェアを運用することに慣れているのか、、 AWS パートナー AWS、またはその両方から同等のマネージドサービスを採用したいのかを検討してください。
- アーキテクチャを最適化して実際のプレイヤーエクスペリエンスを反映するメトリクスを改善する: 時間の経過とともにアーキテクチャを適応および進化させる際には、これらの改善と変更がプレイヤーエクスペリエンスにどのように影響するかを検討してください。ゲームワークロードは、ゲームプレイへの広範な中断をブロックする障害の影響に耐え、最小限に抑えることができる必要があります。障害の影響を軽減し、プレイヤーに影響を与える問題を特定するために、互いに大きく依存していないゲーム機能とシステムを切り離す必要があります。
- ピークプレイヤーの同時実行数に合わせてインフラストラクチャを設計し、必要に応じて動的にスケールする: インフラストラクチャはプレイヤーの需要に合わせてスケールするように設計する必要があります。プレイヤーセッションの同時実行数やログイン数などのメトリクスを使用して、システムが過負荷になる前に事前にスケーリングできます。CPU やメモリの消費量などのリアクティブシステム使用率メトリクスを使用して、システムが過負荷になった後にスケールできます。インフラストラクチャを動的にスケーリングすることで、ゲームの運用コストを削減できます。
- ゲームオペレーションを改善するランブックを実装する: 定期的なゲームオペレーションタスクを一貫して管理するには、運用ランブックを使用する必要があります。ランブックは、プレイヤーレポートの調査と対応、新シーズンのローンチやゲームコンテンツリリースなどの予想される大規模なイベントに備えるためのインフラストラクチャのスケーリング前アクティビティの管理、一般的なゲームメンテナンスアクティビティへの対応など、一般的なゲームオペレーションワークフローのために存在する必要があります。

シナリオ

このセクションでは、ゲームアーキテクチャで一般的ないくつかのシナリオについて説明します。各シナリオには、設計を推進する一般的な特性とリファレンスアーキテクチャ図の例が含まれています。

シナリオ

- リアルタイムの同期ゲームプレイのためのゲームホスティング
- ゲームバックエンド
- サーバーレスベースのゲームバックエンドアーキテクチャ
- クラウドゲーム開発 (CGD)
- Game analytics pipeline (ゲーム分析パイプライン)

リアルタイムの同期ゲームプレイのためのゲームホスティング

リアルタイム同期ゲームプレイでは、2 人以上のプレイヤーがゲームに同時に参加してやり取りできます。ここでは、接続されたプレイヤー間でゲームプレイの状態が共有され、可能な限りリアルタイムエクスペリエンスに近い時間を作成できます。同期ゲームの例としては、ファーストパーソンシューティングゲーム、超マルチプレイヤーオンラインゲーム (MMOG)、スポーツおよびアクションゲーム、またはプレイエクスペリエンスをほぼリアルタイムで共有するために 2 人以上のプレイヤーを接続する必要があるオンラインゲームなどがあります。

リアルタイム同期ゲームプレイアーキテクチャの特徴は次のとおりです。

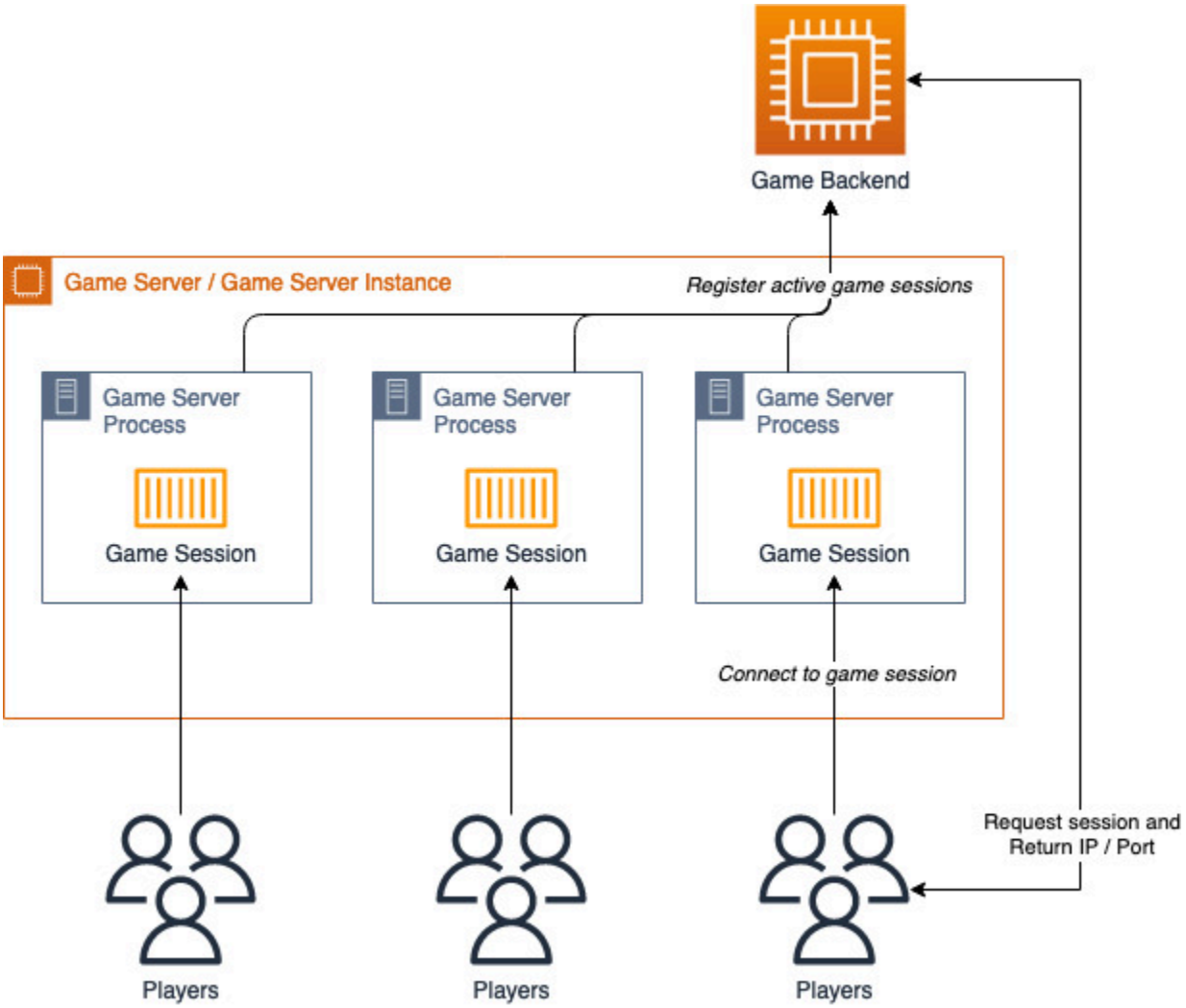
- 一部のゲームは、専用サーバーで実行されるゲームサーバープロセスを通じてゲームセッションとしてホストされる場合があります。一部のゲームでは、より軽量な Session Traversal Utilities for NAT (STUN) または NAT (TURN) サーバーの周りのリレーを使用したトラバーサルを使用する P2P-likeアーキテクチャを使用する場合があります。関連するサーバーの種類にかかわらず、ゲームサーバーは複数のデータセンターで AWS リージョン グローバルにホストされます。
- ゲームクライアントは、ゲームバックエンドシステムでホストされている一元的なマッチメーキングサービスからマッチングをリクエストするか、利用可能なゲームサーバーの事前定義されたリストからマッチングを選択することで、ゲームセッションに参加できます。ゲームクライアントには、接続先の IP アドレスとポートが用意されています。
- 多くの同期ゲームは、ファーストパーソンシューティングゲームや超マルチプレイヤーオンラインゲームなど、レイテンシーに敏感です。これには、遅延効果を最小限に抑えるために巻き戻しや時

間拡張などのアルゴリズムが含まれる可能性がありますが、レイテンシーの高い状況でプレイヤーに発生する可能性のあるラグエクスペリエンスを減らすために慎重に測定および最適化される事前定義されたレイテンシー耐性がある場合もあります。このレイテンシー情報は、ゲームクライアントを計測して使用可能なゲームサーバーに ping AWS リージョン を送信し、レイテンシー、ネットワークジッター、ゲームプレイエクスペリエンスのその他の重要なメトリクスなどのメトリクスをキャプチャすることで決定されます。これらのメトリクスは、ライブオペレーションストリームがゲームの状態をモニタリングできるように、ゲームバックエンドシステムの中央メトリクス収集サービスに送信されます。マッチメーキングプロセス中、ゲームクライアントはマッチングをリクエストするときに現在のレイテンシーデータをリクエストパラメータの 1 つとして提供し、マッチメーキングサービスはプレイヤーをホストするゲームサーバーを選択するときにそのレイテンシーデータを変数の 1 つとして使用できます。

- 通常、ゲームプレイはさまざまなプロトコル (マッチメーキング、認証、HTTPS を使用した他のクライアント/サーバートラフィックによる UDP ベースの高速メッセージングを使用するゲームサーバーなど) で行われます。
- ゲームサーバーはアルゴリズムと設計を採用して、変換ストリーミング、デルタ、データ圧縮などのクライアントとサーバーのトラフィックを最小限に抑えます。
- ゲームサーバーは悪意のあるアクティビティの頻繁なターゲットであり、 のような DDoS 保護ソリューションで保護する必要があります AWS Shield Advanced。

ゲームサーバープロセス

次の図は、一般的なゲームサーバーアーキテクチャを示しています。ゲームサーバーインスタンスと、ゲームセッションをホストするゲームサーバープロセスとの間の論理的な関係について説明します。



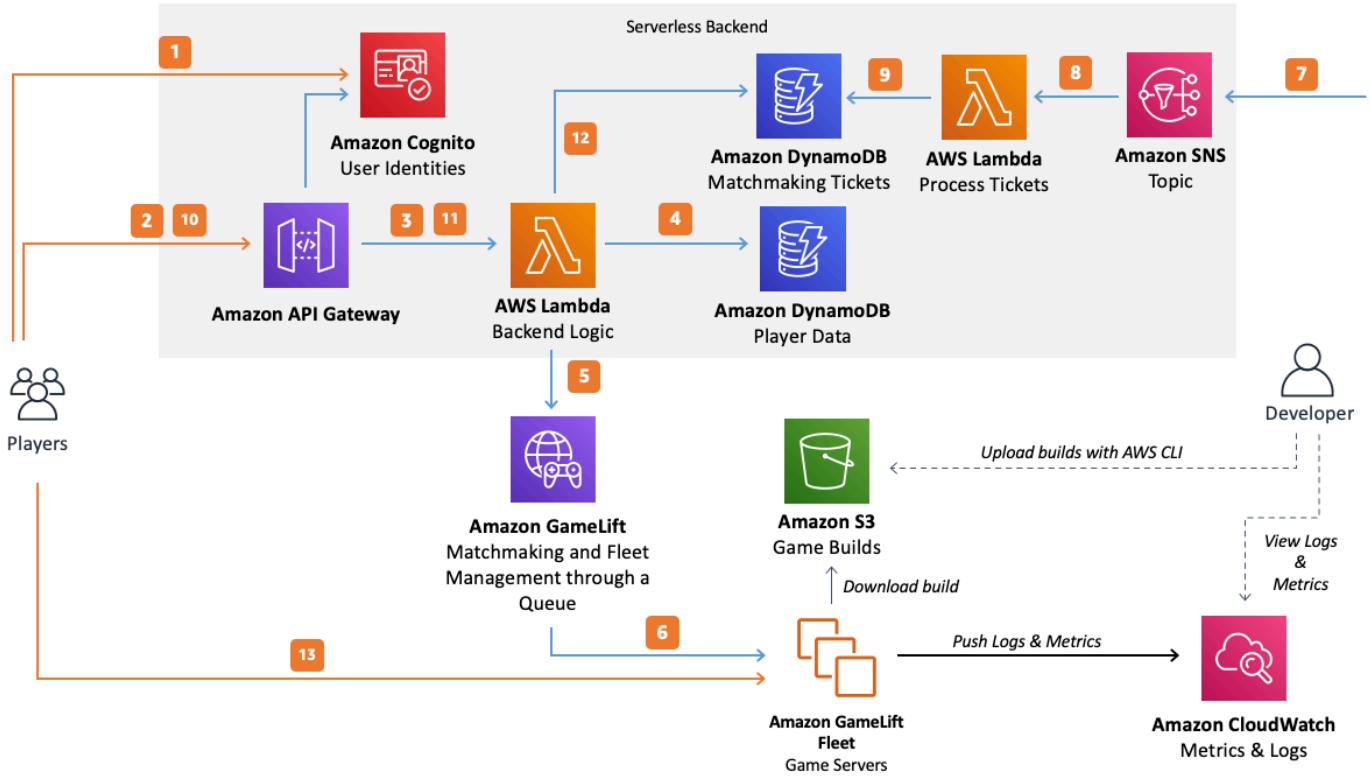
論理ゲームサーバーアーキテクチャ

- Amazon EC2 インスタンスはゲームサーバーとして使用され、ゲームサーバーインスタンスとも呼ばれます。ゲームサーバーは 1 つ以上のゲームサーバープロセスをホストし、それぞれがゲームサーバービルドのコピーを実行しています。通常、複数のゲームサーバープロセスがゲームサーバーインスタンスで実行され、コンピューティングリソースを効率的に活用してコストを削減します。ゲームセッションがアクティブでプレイヤーセッションをホストする準備ができると、そのステータスはゲームバックエンド (通常はマッチメーキングサービス) で更新され、プレイヤーをホストするために使用できるようになります。
- ゲームバックエンドは、プレイヤーがプレイに接続できるように、ゲームセッションをホストしている IP アドレスとサーバーポートをプレイヤーに提供できます。

サーバーレスバックエンドを使用したセッションベースのゲームサーバーホスティング

ゲーム用のアーキテクチャを開発するときは、必要な機能と機能、および所有する準備が整っている運用管理オーバーヘッドのレベルを考慮してください。運用の容易さと柔軟性のバランスを最適化するために、クラウドプロバイダーのマネージドサービスを使用してゲームを構築できます。マネージドサービスを使用すると、独自のカスタムゲーム機能の開発とカスタマイズを制御できると同時に、インフラストラクチャのデプロイと管理の負担を軽減できます。

セッションベースのマルチプレイヤーゲームをホストするには、ゲームサーバープロセスをホストするサーバーインフラストラクチャと、マッチメーキングとセッション管理のためのスケーラブルなバックエンドが必要です。次のリファレンスアーキテクチャは、Amazon GameLift マネージドホスティングとサーバーレスバックエンドを使用してセッションベースのゲームを管理する方法を示しています。



セッションベースのゲーム用の Amazon GameLift マネージドホスティング

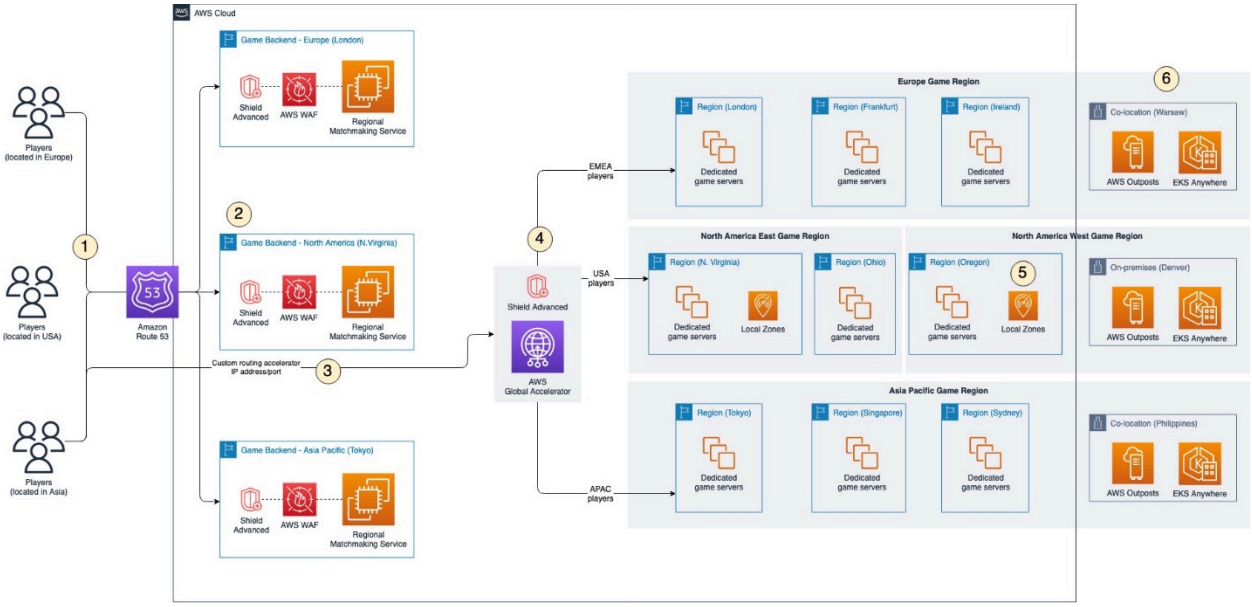
この図は、GameLift マネージドゲームホスティングで実行されているゲームにプレイヤーを参加させるプロセスを示しています。以下のステップには次のものが含まれます。

1. ゲームクライアントは、Amazon Cognito ID プールから Amazon Cognito ID をリクエストします。これは、オプションで外部 ID プロバイダーに接続できます。
2. ゲームクライアントは一時的なアクセス認証情報を受け取り、Amazon API Gateway を介してゲームセッションをリクエストします。 Amazon Cognito
3. API Gateway は AWS Lambda 関数を呼び出します。
4. Lambda 関数は、Amazon DynamoDB テーブルからプレイヤーデータをリクエストします。Amazon Cognito ID は、認証された ID がリクエストコンテキストデータで提供されるため、正しいプレイヤーデータを安全にリクエストするために使用されます。
5. 追加情報 (プレイヤースキルレベルなど) に正しいプレイヤーデータを使用すると、Lambda 関数は GameLift FlexMatch マッチメーキングを通じてマッチングをリクエストします。FlexMatch マッチメイキング設定は、JSON ベースの設定ドキュメントで定義できます。ゲームクライアントは、さまざまなリージョンのサーバーエンドポイントに ping を送信することでレイテンシーメトリクスを生成でき、レイテンシーデータを使用してレイテンシーベースのマッチメーキングをサポートできます。
6. FlexMatch は、適切なレイテンシーを持つ適切なプレイヤーグループを リージョンにマッチさせた後、GameLift キューを通じてゲームセッションの配置を要求します。キューには、1 つ以上の登録済みリージョンロケーションを持つフリートが含まれます。
7. セッションがフリートのいずれかのロケーションに配置されると、Amazon SNS トピックにイベント通知が送信されます。
8. Lambda 関数は Amazon SNS イベントを受け取り、それを処理します。
9. Amazon SNS メッセージが MatchmakingSucceeded イベントの場合、Lambda 関数はサーバーポートと IP アドレスを使用して結果を DynamoDB に書き込みます。time-to-live (TTL) 値を使用して、不要になったマッチメーキングチケットが DynamoDB から削除されるようにします。
10. ゲーム クライアントは API Gateway に対して署名付きリクエストを行い、特定の間隔でマッチメイキングチケットのステータスをチェックします。
11. API Gateway は、マッチメイキングチケットのステータスをチェックする Lambda 関数を呼び出します。
12. Lambda 関数は DynamoDB をチェックして、チケットが成功したかどうかを確認します。成功すると、Lambda 関数は IP アドレス、ポートおよびプレーヤーセッション ID をクライアントに送信します。チケットが失敗した場合、Lambda 関数はマッチングの準備が整っていないことを示すレスポンスを送信します。
13. ゲームクライアントは、バックエンドから提供されたポートと IP アドレスを使用して、TCP または UDP を使用してゲームサーバーに接続します。プレイヤーセッション ID をゲームサーバーに送信し、ゲームサーバーは Amazon GameLift Server SDK を使用して検証します。

または、前述のアーキテクチャを変更して、Amazon GameLift で API Gateway WebSockets を使用することもできます。このアプローチでは、ゲームクライアントとゲームバックエンドサービス間の通信は、WebSocket ベースの実装を使用して行われます。この実装を使用すると、ゲームバックエンドの Lambda 関数がポーリングモデルを実装するのではなく、WebSocket 経由でゲームクライアントへのサーバー側のメッセージを開始できます。

低レイテンシーゲーム用のマルチリージョンおよびハイブリッドアーキテクチャ

このセクションでは、低レイテンシーゲーム用のマルチリージョンおよびハイブリッドアーキテクチャについて説明します。



グローバルにデプロイされたネットワークアクセラレーションとゲームサーバーによるレイテンシーの削減

1. グローバルに利用可能なゲームのプレイヤーは、どこからでも発信できます。プレイヤーがゲームセッションまたは試合をリクエストすると、ゲームクライアントは Amazon Route 53 に登録されているゲームバックエンドサービスにリクエストを送信します。Route 53 レイテンシーベースのルーティングを使用して、プレイヤーを利用可能な最も近いゲームバックエンドにルーティングできます。
2. ゲームバックエンドは、プレイヤー母集団に最も AWS リージョン 近い複数の にデプロイされます。各ゲームバックエンドには、ゲームリージョン間でゲームセッションを検索するリージョン別マッチメーキングサービスが含まれています。プレイヤーのマッチメーキングリクエストは、近くにあるリージョンのマッチメーキングサービスによって処理されますが、必要に応じ

て、マッチメーキングサービスはプレイヤーを別のゲームリージョンのゲームセッションにルーティングできます。このアクションにより、耐障害性とパフォーマンスが向上します。さらに、各ゲームバックエンドサービスはAWS WAF と を使用して AWS Shield Advanced 、レイヤー 7 ウェブフィルタリングとボット制御、およびディストリビューションサービス拒否 (DDoS) 攻撃に対する保護を提供します。サーバーレス、コンテナ、EC2 インスタンス、独自のデータセンターでのゲームバックエンドサービスのホスティングなど、ゲームバックエンドサービスの構築方法には多くのオプションがあります。

3. ネットワークのレイテンシーとジッターを減らすことでプレイヤーのエクスペリエンスを向上させるために、 AWS Global Accelerator を使用してカスタムルーティングアクセラレーターがデプロイされます。Global Accelerator は、 AWS グローバルネットワークを使用してゲームクライアントからゲームサーバーへのトラフィックのルーティングを自動的に最適化します。Global Accelerator リスナーポートをゲームサーバーの EC2 インスタンスポートにマッピングするようにカスタムルーティングアクセラレーターを設定します。 EC2 ゲームクライアントは、プロキシとして機能する Global Accelerator IP とポートに接続し、プレイヤーをゲームセッションをホストしている正しいゲームサーバー IP とポートに決定的にルーティングします。
4. ゲームには、北米やアジアパシフィックなど、地理的に互いに近いゲームサーバーホスティングロケーションのコレクションを表すプレイヤーフレンドリーな論理ゲームリージョンが含まれています。レイテンシーを短縮し、地理的カバレッジを向上させるために、さまざまなゲームサーバーホスティングソリューションを組み合わせてプレイヤーエクスペリエンスを向上させることができます。これらのロケーションは完全に機能し、キャパシティフットプリントが最も大きいため、 AWS リージョン 可能な限り の使用を優先します。
5. AWS Local Zones を使用して、既存のホスティング施設がない、または AWS リージョン が利用できない、サービスが不十分なプレイヤーの地理的な場所でゲームサーバーをホストします。
6. 既存のオンプレミスデータセンターとコロケーションプロバイダー AWS Outposts にデプロイして、フルマネージドラックとラックマウントサーバーを使用して、各デプロイロケーションにシームレスなコントロールプレーンと管理エクスペリエンスを作成します。 AWS Outposts は、オンプレミス環境に既存のサーバー容量がない場合にも役立ちます。ただし、独自の既存のサーバーインフラストラクチャで実行されているソフトウェアを使用したハイブリッド実装が必要な場合は、Amazon EKS Anywhere を使用する必要があります。これにより、Amazon EKS に接続して独自のインフラストラクチャで Kubernetes クラスターを作成して実行できます。これにより、オンプレミスの Kubernetes クラスターの一貫したコンソールビューが提供されます。

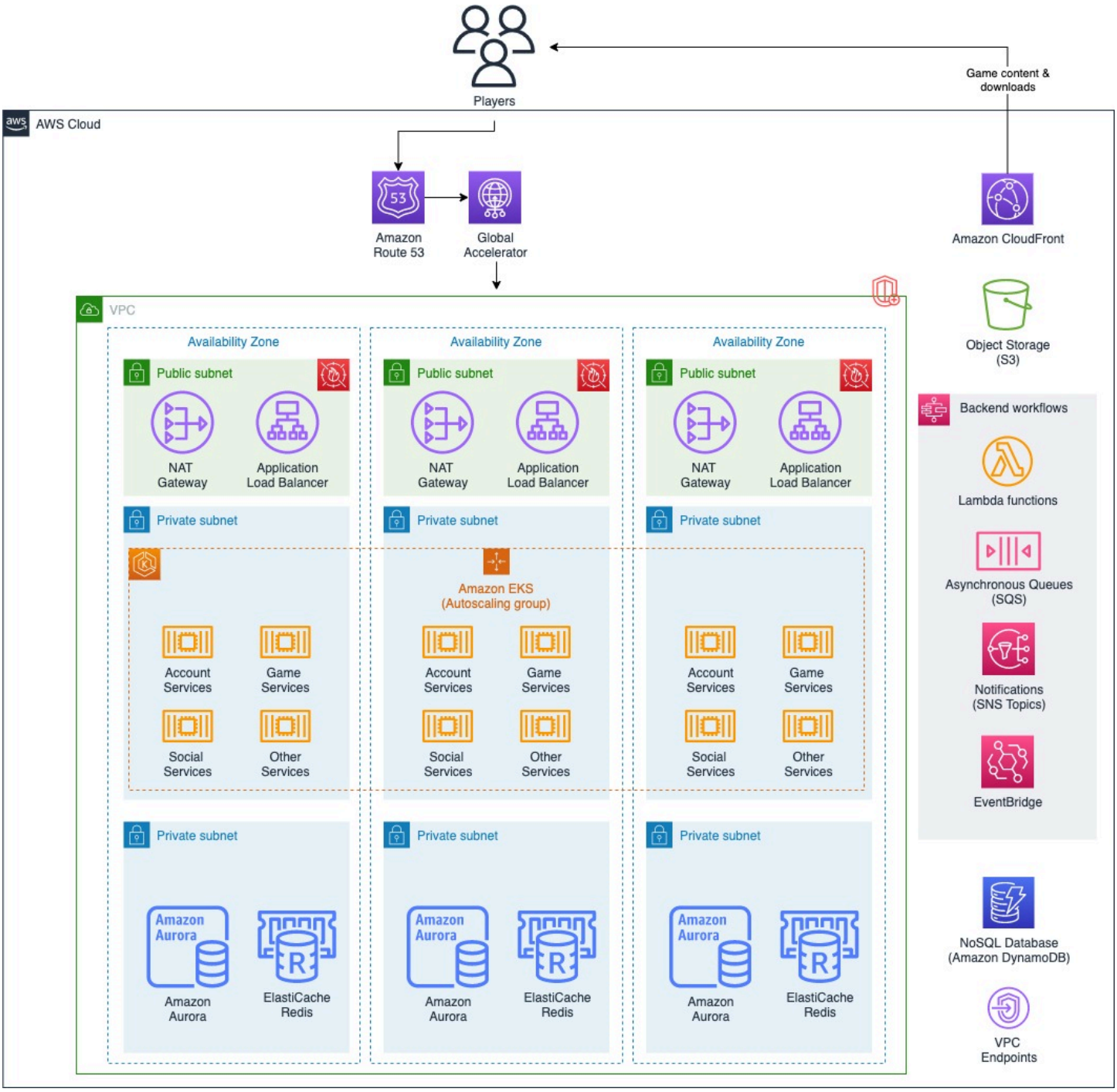
ゲームバックエンド

ゲームバックエンドは、ゲームとプレイヤーの状態を管理し、ソーシャルおよびシステムレベルの機能をゲームエクスペリエンスをサポートするゲームに統合するために使用されます。プレイヤープロファイル管理、アイテムとインベントリのストレージ、統計とリーダーボードは、ゲームバックエンドでホストされるサービスの例です。

ゲームバックエンドは通常、HTTPS を使用してクライアントがアクセスする REST APIs として構築されます。ただし、ゲーム内チャットやプレゼンスのクライアント通知などのユースケースに双方向チャネルを提供する WebSocketsなど、他のアプローチも一般的です。ゲームバックエンドは、インスタンス、コンテナ、サーバーレスアーキテクチャなど、さまざまなデプロイアーキテクチャを使用してデプロイできます。

コンテナベースのゲームバックエンドアーキテクチャ

このセクションでは、コンテナベースのゲームバックエンドアーキテクチャの概要を説明します。



コンテナを使用したゲームバックエンドのホスティング

- プレイヤーはゲームクライアントソフトウェアを使用してゲームにアクセスします。このソフトウェアは、ゲームシステム、デジタルストアフロント、または Amazon CloudFront などのコンテンツ配信ネットワーク (CDN) からの直接ダウンロードを通じて配布できます。CDNs はエッジロケーションでキャッシュを提供し、コンテンツをダウンロードするユーザーのパフォーマンスを向

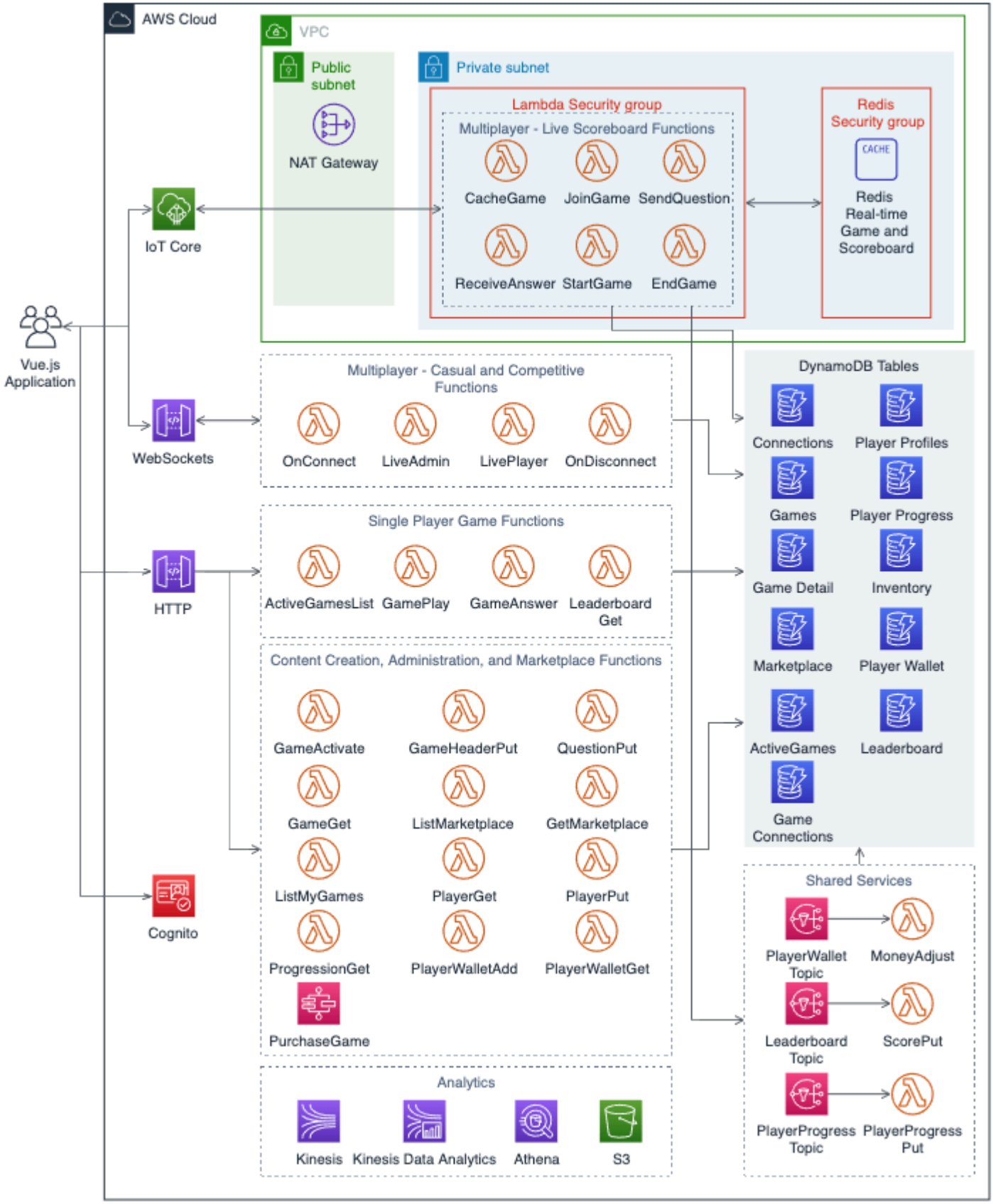
上させます。例えば、CloudFront を使用してゲームクライアントソフトウェアをプレイヤーだけでなく、ゲームアセットやその他のコンテンツに配信できます。

- AWS Global Accelerator は、マルチリージョンおよびフェイルオーバーの目的で、プレイヤーゲームクライアントからロードバランサーにトラフィックをルーティングしたり、リージョン間でトラフィックをルーティングしたりするためのトラフィックアクセラレーションとカスタマイズ可能なコントロールを提供します。ゲームバックエンド REST APIs のカスタムドメイン名は、Global Accelerator エンドポイントにトラフィックをルーティングするように Amazon Route 53 で設定されます。図に示されていないように、AWS Shield Advanced はアクセラレーターとゲームバックエンドに追加の DDoS 緩和を提供できます。
- NAT Gateway と Application Load Balancer は、リージョンで高可用性を提供するために、ゲームバックエンドで使用される各アベイラビリティーゾーンのパブリックサブネットにデプロイされます。AWS WAF はApplication Load Balancer にデプロイされ、レイヤー 7 ウェブトラフィックフィルタリングを提供します。
- ゲームバックエンドは、回復性のためにアベイラビリティーゾーンにまたがるプライベートサブネットの Amazon EKS クラスターにデプロイされた個々のコンテナベースのマイクロサービスのコレクションとしてホストされます。自動スケーリングは、通常プレイヤーの需要と相関するリソース使用率に基づいて、サービスとクラスターノードの容量を動的に調整します。Cluster Autoscaler はクラスター内のノード数を自動的に調整しますが、Horizontal Pod Autoscaler はクラスターにデプロイされたポッドを自動的にスケーリングします。
- ゲームとプレイヤーのデータはバックエンドデータベースに保存され、キャッシュはアベイラビリティーゾーン間でプライベートサブネットにデプロイされ、プライマリノードとレプリカノード間でレプリケーションされます。Amazon Aurora は、プレイヤープロファイル、使用権限、ゲーム内購入などのユースケースによく使用される選択肢であり、クエリ要件がより複雑になり、MySQL と PostgreSQL のリレーショナルデータモデリング機能の利点が得られる可能性があります。Amazon ElastiCache は、高パフォーマンスのリーダーボード、pub/sub メッセージングの構築、データベースのレイテンシーと負荷を減らすために頻繁にアクセスされるデータのキャッシュに役立ちます。Amazon DynamoDB はフルマネージド型の NoSQL データストアであり、プレイヤーとゲームの状態データ、セッションデータ、インベントリとアイテムストアなどのユースケース、または最小限のオーバーヘッドでグローバルデータベースを必要とするユースケースで、予測不可能なアクセスパターンと実質無制限のスループットにスケールする機能に最適です。
- リーダーボードの更新や友人リクエストの送信など、バックグラウンドで実行できる作業を実行するには、非同期処理ワークフローを使用する必要があります。このタイプの作業を Amazon SQS キューにプッシュするようにゲームバックエンドを設定するか、Amazon SNS トピックを使用して並列処理のために多くのコンシューマーアプリケーションキューに作業を分散することを検討してください。Lambda AWS関数 を使用してイベント駆動型の方法で処理を実行し、

コンピューティングインフラストラクチャのコストと管理オーバーヘッドを削減します。存続期間が長いワークフローや、複数のステップによるタスクの調整が必要なワークフローの場合は、 を使用してワークフロー全体をオーケストレーションすることを検討してくださいAWS Step Functions。Amazon EventBridge を使用して、 AWS サービスおよびカスタムアプリケーションイベントに応答する関数を開始できます。

サーバーレスベースのゲームバックエンドアーキテクチャ

多くのゲーム開発者はインフラストラクチャを管理したくないため、代わりにソフトウェアに集中できるテクノロジーを使用してゲームを構築することを好みます。このシナリオでは、サーバーレスアーキテクチャが推奨されます。これは、機能をより迅速に構築およびリリースし、運用オーバーヘッドを減らすことができるためです。サーバーレスアーキテクチャは、サーバーをセットアップ、管理、スケーリングすることなく、需要に応じて動的にスケールできるクラウドサービスを使用して設計されています。次のリファレンスアーキテクチャは、サーバーレスアーキテクチャを使用してゲームを構築する方法を示しています。


AWS Cloud
VPC
Public subnet
NAT Gateway
Private subnet
Lambda Security group
Multiplayer - Live Scoreboard Functions
CacheGame
JoinGame
SendQuestion
ReceiveAnswer
StartGame
EndGame
Redis Security group
CACHE
Redis Real-time Game and Scoreboard
IoT Core
Vue.js Application
WebSockets
HTTP
Cognito
Multiplayer - Casual and Competitive Functions
OnConnect
LiveAdmin
LivePlayer
OnDisconnect
Single Player Game Functions
ActiveGamesList
GamePlay
GameAnswer
Leaderboard Get
Content Creation, Administration, and Marketplace Functions
GameActivate
GameHeaderPut
QuestionPut
GameGet
ListMarketplace
GetMarketplace
ListMyGames
PlayerGet
PlayerPut
ProgressionGet
PlayerWalletAdd
PlayerWalletGet
PurchaseGame
Analytics
Kinesis
Kinesis Data Analytics
Athena
S3
DynamoDB Tables
Connections
Player Profiles
Games
Player Progress
Game Detail
Inventory
Marketplace
Player Wallet
ActiveGames
Leaderboard
Game Connections
Shared Services
PlayerWallet Topic
MoneyAdjust
Leaderboard Topic
ScorePut
PlayerProgress Topic
PlayerProgress Put

サーバーレスベースのゲームバックエンドリファレンスアーキテクチャ

このリファレンスアーキテクチャは、シングルプレイヤーとマルチプレイヤーの機能を提供するウェブベースのトリビアゲームを示しています。

- プレイヤー認証: プレイヤーは Amazon Cognito を使用して認証します。Amazon Cognito は、プレイヤー ID 管理用のユーザーディレクトリを使用した安全な認証を提供します。
- サーバーレス関数としてのゲームロジック: ゲーム機能およびバックエンドビジネスロジックは、イベントに応答して開始される関数として実行され、 AWS Lambda 関数の実行時にのみ料金を支払うため、コストが削減されます。Lambda では、選択したプログラミング言語を使用して、各ゲーム機能を個別のマイクロサービスとして柔軟に記述できます。たとえば、C# を使用して Unity ゲームを構築した経験がある場合は .NET Lambda 関数を開発するか、JavaScript でウェブベースのゲームのフロントエンドとバックエンドをプログラムする場合は Node.js Lambda 関数を開発するかを選択できます。
- ゲームおよびプレイヤーデータ用の NoSQL データストア: DynamoDB を使用してプレイヤーおよびゲームデータを保存します。これは、マイクロサービスから大量のデータを保存するために構築されているためです。このアーキテクチャで示されているように、ゲーム機能のデータストレージのニーズごとに個別のデータストアを使用するのがベストプラクティスです。これにより、機能を個別にモニタリングおよび管理しやすくなります。これにより、チーム内で機能またはサービスの所有権が変更された場合にも、分離の境界が作成されます。このリファレンスアーキテクチャでは、DynamoDB テーブルを使用して、接続状態、ゲームの詳細、プレイヤーの進行状況、リーダーボード情報などのデータを保存します。
- シングルプレイヤーゲームプレイ: シングルプレイヤー機能を使用すると、プレイヤーはゲームの選択やプレイ、リーダーボードの表示などのアクションを実行できます。これらの機能は、適切な Lambda 関数を呼び出して DynamoDB テーブルでデータを取得して設定する Amazon API Gateway HTTP API でホストされる RESTful バックエンドサービスとして実装されます。ゲームプレイが完了すると、バックエンドは Amazon SNS トピックにも通知を送信し、Lambda 関数を非同期的に開始してプレイヤーの進行状況と統計を保存します。
- マルチプレイヤーゲームプレイ: マルチプレイヤーゲーム機能では、プレイヤーはpoint-to-point通信のためにゲームとやり取りでき、他の接続されたプレイヤーから更新をブロードキャストおよび受信できる必要があります。WebSockets の実装は、トリビアなどの軽量ゲームでのpoint-to-point通信に適しています。プレイヤーは Amazon API Gateway WebSockets への WebSockets 接続を確立できます。これにより、接続を管理し、プレイヤーに対して送受信するメッセージがある場合にのみ Lambda 関数を呼び出します。プレイヤー間で one-to-manyの通信が必要なユースケースの場合、 は MQTT 経由で WebSockets を使用したメッセージングのサポート AWS IoT Core を提供します。これにより、クライアントはトピックをサブスクライブし、受信したメッセージに対応

できます。このアーキテクチャでは、MQTT 経由の WebSockets は、ゲーム内のライブ更新のブロードキャストや、接続されたプレイヤーへの質問などのユースケースをサポートするために使用されます。代わりに AWS IoT、メッセージ配信に Redis Pub/Sub を選択するか、メッセージ保持が必要な場合は Redis Streams を選択できます。

- VPC 対応 Lambda 関数を使用してプライベートサブネット内のリソースにアクセスする: Amazon ElastiCache などの VPC のプライベートサブネット内のリソースにアクセスするように VPC 対応 Lambda 関数を設定します。これは、ライブリーダーボードなどの低レイテンシーデータセットのクエリ時間を短縮するために使用されます。

詳細については、「 でのカスタムゲームバックエンドホスティングのガイダンス AWS」を参照してください。

クラウドゲーム開発 (CGD)

クラウドゲーム開発 (CGD) とは、ゲーム開発ライフサイクルがゲームを構築、テスト、開発するために必要なインフラストラクチャとツールを指します。ゲーム開発はユーザー間の共同作業であり、インフラストラクチャ要件は開発の段階を通じて頻繁に変化します。

多くのゲーム開発者は、グローバルに分散されたリモート開発チームを採用しており、このタイプの開発をサポートするテクノロジーが必要です。ゲーム開発者は、これらの環境の全部または一部をでホスト AWS し、 AWS リージョン のグローバル可用性を使用してリソースをユーザーの近くに配置し、必要に応じてコンピューティングとストレージをスケーリングすることで開発環境をより費用対効果の高い方法で管理できます。

環境はゲーム開発者のニーズに応じて異なる場合がありますが、通常、アーティスト、デザイナー、エンジニア、QA テスター、請負業者、その他の担当者が作業を実行するための開発者ワークステーションが含まれます。これらの環境には、通常、ユーザーが変更をチェックインするためのソースコードリポジトリで構成されるビルドファームと、開発されたアーティファクトを構築、パッケージング、テストするための CI/CD インフラストラクチャも含まれます。

これらのゲーム本番稼働用アーキテクチャには、次の特性があります。

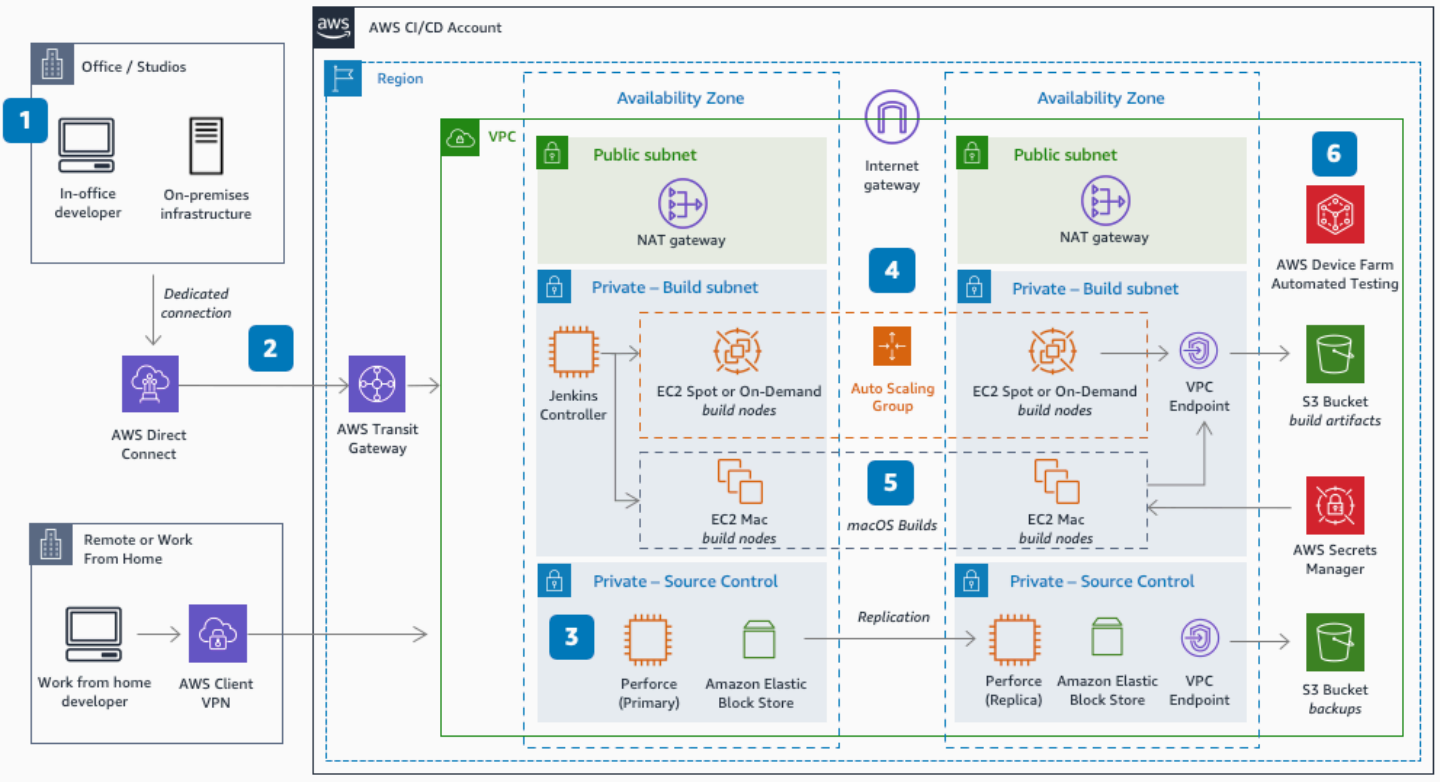
- ユーザーは、ウェブブラウザまたは Amazon DCV などのローカルデスクトップクライアントを介して仮想ワークステーションにアクセスできる必要があります。これにより、オフィスや開発スタジオのマシンで作業している場合と同じソフトウェアやツールにアクセスするための低レイテンシーのストリーミングセッションが提供されます。これらの仮想ワークステーション、通常はクラウドベースのサーバーは、ユーザーが LAN または WAN 経由で完全にクラウド環境でプロジェク

トを共同作業および作業できるようにする必要があります。ユーザーがマシンをアクティブに使用していない場合は、Amazon Amazon Elastic File System EFS FSx などのソースコントロールリポジトリやファイルシステムなど、耐久性のあるクラウドストレージに作業をバックアップし、コストを削減するためにマシンをシャットダウンする必要があります。

- Perforce などのソースコントロールリポジトリは、アベイラビリティーゾーン間、または Amazon S3 などのクラウドストレージにバックアップが保存されているオンプレミス環境間のレプリケーションを使用して、高可用性で設計する必要があります。たとえば、クラウドベースの Perforce サーバーには、あるアベイラビリティーゾーンでホストされているプライマリコミットサーバーと、同じリージョン内の別のアベイラビリティーゾーンでホストされているスタンバイサーバーへのレプリケーションを含める必要があります。
- ゲーム開発ビルドファームリソースは、コンピューティングリソースが必要に応じてプロビジョニングされるように自動スケーリングで設計する必要があります。また、EC2 スポットインスタンスを使用して、ビルドに必要なサーバー数をスケールアウトするときに発生するコストを削減する必要があります。

クラウドゲーム開発: CI/CD

CI/CD インフラストラクチャは、反復時間を改善し、信頼性を構築し、効率的なデプロイを行い、開発とリリースプロセスをより適切に制御して、高品質のゲームエクスペリエンスをプレイヤーに提供するために、チームサイズに関係なくゲームを開発するときに重要です。ゲーム開発 CI/CD パイプラインは通常、高可用性のソースコントロールサーバーとストレージ、ビルドを実行するコンピューティングリソース、および自動テストを実行するソフトウェアと、開発マシンからの適切なネットワーク接続で構成されます。次のリファレンスアーキテクチャは、ゲームビルドをリモートまたはオンプレミスのゲーム開発環境から にオフロード AWS クラウド して、開発者が新しいビルドファームを移行または構築するのを支援する方法を示しています。



ゲームビルドをクラウドにオフロードする

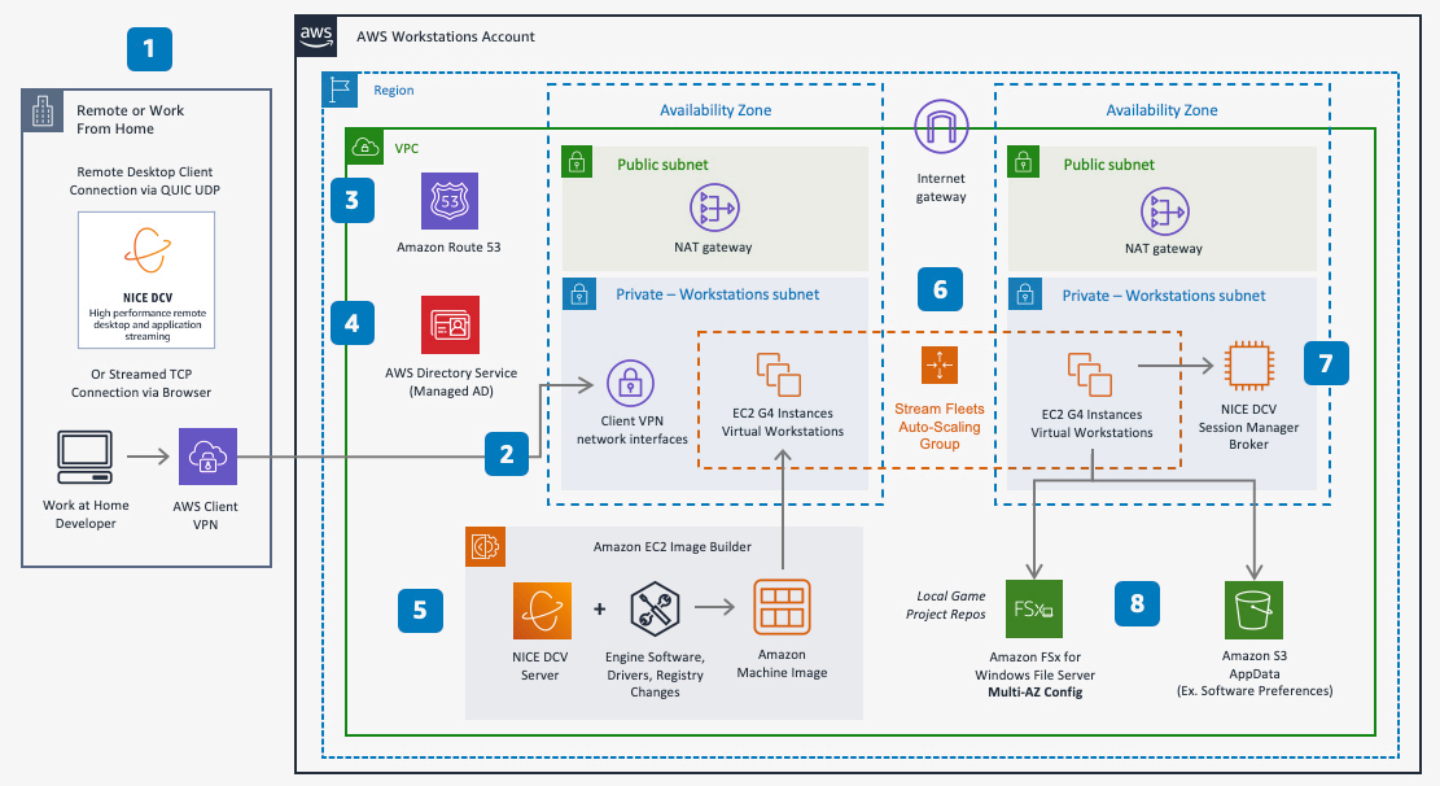
1. AWS Direct Connect は、オフィス内の開発者 AWS 向けに、低レイテンシーのプライベート専用接続を提供します。リモート開発者は、 などのゼロトラストテクノロジーや AWS Verified Access、 AWS クライアント VPN などの仮想プライベートネットワーク (VPN) を使用します。
2. AWS Transit Gateway はVPCs とオンプレミスネットワーク間の接続のためのネットワーク管理を簡素化します。
3. Perforce は、Amazon EBS ストレージにバックアップされたソースとバージョン管理 (CI) を管理し、すばやくアクセスされ、永続的なデータを提供します。Perforce Helix Core は、 で利用できますAWS Marketplace。
4. 開発者がブランチに関連付けられた Perforce に変更をプッシュすると、Jenkins でビルド (CD) が開始されます。Perforce は Jenkins への JSON ペイロードの POST を開始します。Jenkins コントローラーは、エンジンヘッドレス CLI コマンドを呼び出して、エフェメラルな Docker ノード (Amazon EC2 スポットインスタンスや Amazon EC2 オンデマンドインスタンスなど) 間でビルドプロセスを実行し、並列化します。開発者は、ロードバランサーの背後にある各アベイラビリティーゾーンに 1 つずつ、2 つの Jenkins コントローラーを使用することで可用性を高めることができます。一部のゲームエンジンでは、開発者は、同時ビルドが実行されるたびにビルドコンテキストのライセンスを提供するために、追加のサブネットに設定された追加のライセンスインフラストラクチャが必要になる場合があります。

5. iOS ビルドの Xcode 部分は Amazon EC2 Mac インスタンスにオフロードされて .IPA ファイルに署名、ビルド、エクスポートされ、プロセスを分割してビルド時間を短縮します。 はプロビジョニングプロファイル、プライベートキー、証明書AWS Secrets Manager を保持します。
6. ビルドアーティファクトは Amazon S3 に配信され、成功または失敗の通知が送信されます。 はモバイルデバイスの自動テスト AWS Device Farm を有効にします。

クラウドゲーム開発: ワークステーション

ゲーム開発には、多くの場合、さまざまな場所からリモートで作業する分散型チームが必要です。そのためには、共有インフラストラクチャへのアクセスと、地理的に分散した共同開発をサポートする能力が必要です。この傾向は、work-for-hire スタジオやリモートワークの使用など、より分散したゲーム開発プロセスに向けられており、生産性、専門知識の共有、アジャイル開発プラクティスを促進するために、堅牢なテクノロジーとワークフローを実装する必要があります。

次のリファレンスアーキテクチャは、Amazon DCV プロトコルを使用してリモートゲーム開発ワークステーションをホストするために を使用する AWS 方法を示しています。



Amazon DCV を使用してどこからでもゲーム開発をストリーミングする

1. Amazon DCV は、4K, 60-FPSのストリーミングをサポートするストリーミングプロトコルです。ブラウザを使用する開発者は TCP 接続を介して接続しますが、デスクトップクライアントはポート 8443 経由で QUIC UDP を使用してパフォーマンスを向上させることができます。
2. デベロッパーは、 AWS クライアント VPN を使用して、ソースネットワークアドレス変換 (SNAT) を使用してワークステーションサブネット内のネットワークインターフェイスに安全に接続します。
3. Amazon Route 53 は、VPC 内のリソースのプライベート DNS と、インバウンドおよびアウトバウンドの DNS 転送を提供します。
4. Directory Service は、個々のユーザーにマッピングされたローカルゲームプロジェクトストレージを有効にするマネージド Microsoft Active Directory を提供します。
5. ワークステーションは、Image Builder で構築された Amazon マシンイメージ (AMI) を使用して作成されます。イメージには、Amazon DCV サーバー、開発者ソフトウェア、レジストリの変更、NVIDIA ゲームドライバーや周辺機器ドライバーなどのドライバーが含まれます。 には、ワークステーションに使用される一般的な AMIs AWS Marketplace が含まれています。
6. ワークステーションのフリートは、GPUs を提供するグラフィックス Amazon EC2 インスタンスタイプを使用し、EC2 Auto Scaling グループを使用してスケーリングされます。
7. Amazon DCV セッションマネージャーブローカーを使用すると、Amazon DCV セッションを管理できます。
8. プロジェクトのローカルファイルストレージは、Amazon FSx for Windows File Server でホストされます。開発者は、ローカルストレージからソースコントロールにプッシュすることで、別の CI/CD パイプラインにコミットします。

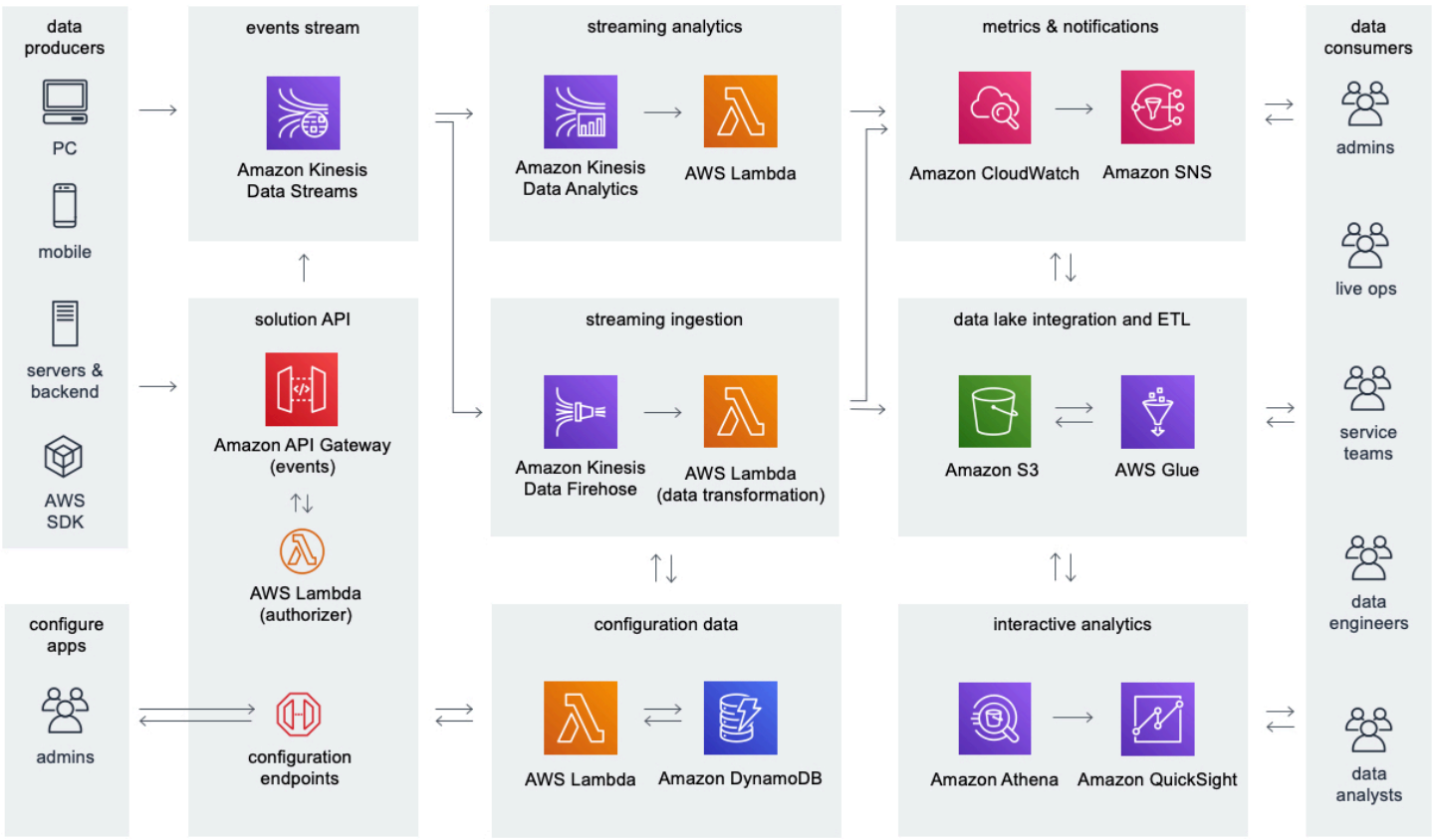
Game analytics pipeline (ゲーム分析パイプライン)

ゲームデベロッパーは、プレイヤーの行動をよりよく理解し、プレイヤーベースを維持および拡大するためのゲームプレイエクスペリエンスを改善できる方法を探し始めています。ゲーム分析は、ゲームおよび関連サービスから生成されたデータを理解して分析するために必要な技術インフラストラクチャとプロセスを表します。これには通常、Game Analytics Pipeline ソリューションの実装など、このend-to-endのプロセスをサポートできる分析パイプラインアーキテクチャを使用する必要があります。 https://aws.amazon.com/solutions/implementations/game-analytics-pipeline/

ゲーム分析アーキテクチャには次の特性があります。

- データソースは JSON などの一般的な形式でデータを送信し、通常はゲームサーバーやゲームバックエンドサービス、PC、モバイルデバイス、ゲームコンソールなどのゲームクライアントが含まれます。

- ゲーム分析パイプラインは、未加工データを取り込んで保存し、使用可能な出力形式に処理するワークフロー全体を自動化し、エンドユーザーや分析アプリケーションなどのデータコンシューマーが効率的かつ費用対効果の高い方法で分析できるようにします。
- ゲーム分析パイプラインは、ゲームの成長に合わせてスケールするために、大量のリアルタイムデータの取り込みと処理をサポートします。
- リアルタイムレポートとバッチレポートの両方のユースケースをサポートします。たとえば、リアルタイムダッシュボードとアラートは通常、ライブ運用チームがゲームインフラストラクチャとプレイヤーの動作をモニタリングして問題を検出するために使用されます。データアナリストチームは通常、時間の経過に伴う傾向を把握するために、必要に応じてバッチレポートを使用します。



ゲームプレイテレメトリ用のサーバーレスゲーム分析パイプライン

ゲームデータは、ゲームクライアント、ゲームサーバー、その他のアプリケーションから取り込まれます。ストリーミングデータは、データレイク統合とインタラクティブ分析のために Amazon S3 に取り込まれます。ストリーミング分析はリアルタイムイベントを処理し、メトリクスを生成します。データコンシューマーは、Amazon CloudWatch のメトリクスデータと Amazon S3 の raw イベントを分析します。

- ソリューション API と設定データ: Amazon API Gateway を使用して、ゲーム分析パイプラインを管理し、Lambda 関数を使用して設定データを Amazon DynamoDB に保存するための REST API を提供します。この API または管理用のカスタムコマンドラインインターフェイスの上に内部ポータルを構築できます。REST API は、データソースからゲームプレイデータを取り込み、テレメトリデータを Amazon Kinesis Data Streams に転送してリアルタイムの処理とストレージへの取り込みを行うためのサーバー認証も提供します。
- イベントストリーム: Amazon Kinesis Data Streams はゲームからストリーミングデータをキャプチャし、Amazon Data Firehose と Amazon Managed Service for Apache Flink によるリアルタイムのデータ処理を可能にします。
- ストリーミング分析: Managed Service for Apache Flink は、Kinesis Data Streams からのストリーミングイベントデータを分析し、Lambda 関数を使用して CloudWatch に発行されるカスタムメトリクスとアラートを生成できます。
- メトリクスと通知: Amazon CloudWatch を使用して、ソリューションのメトリクス、ログ、アラームをモニタリングします。Amazon SNS を使用して、オンコールエンジニアやその他のデータコンシューマーに通知を送信します。
- ストリーミング取り込み: Firehose を使用して Kinesis Data Streams からストリーミングデータを消費し、Amazon S3 のデータレイクに配信して、長期保存、変換、他のデータとの統合を行います。
- データレイク統合と ETL: ETL AWS Glue 処理ワークフローに を使用し、 でメタデータを整理します。これにより AWS AWS Glue Data Catalog、柔軟な分析ツールと統合するためのデータレイクの基礎が提供されます。
- インタラクティブ分析: エンドユーザーは Amazon Athena を使用して Amazon S3 に保存されているデータセットに対してアドホックなインタラクティブクエリを実行でき、Quick Suite を使用してダッシュボードを構築できます。

を使用してアカウントにデプロイできる分析パイプラインの自動リファレンス実装については、Game Analytics Pipeline を参照してください AWS CloudFormation。

定義

AWS Well-Architected フレームワークは、運用上の優秀性、セキュリティ、信頼性、パフォーマンス効率、コスト最適化、持続可能性の 6 つの柱に基づいています。 AWS には、ゲームワークロードに合わせてstate-of-the-artアーキテクチャを設計できる複数のコアコンポーネントが用意されています。このセクションでは、キー定義の概要を示します。

このホワイトペーパーでは、ゲームアーキテクチャには、ゲームの構築と運用に必要なバックエンドの技術インフラストラクチャが含まれています。一部のゲームには、ソーシャル、マルチプレイヤー、またはその他のオンライン機能がなく、このホワイトペーパーで説明されているバックエンド技術インフラストラクチャの特定の側面を使用する必要がない場合があります。ゲームアーキテクチャをサポートするために頻繁にデプロイされるさまざまなタイプのワークロードの詳細については、「シナリオ」を参照してください。

AWS クラウド インフラストラクチャは、リージョンとアベイラビリティーゾーンを中心に構築されています。

- リージョンは、複数のアベイラビリティーゾーンがある世界の物理的な場所です。
- アベイラビリティーゾーンは 1 つ以上の個別のデータセンターで構成され、それぞれが冗長な電源、ネットワーク、および接続を備え、別々の施設に収容されています。

ゲームの特性によっては、プレイヤーのパフォーマンスを向上させるなどの理由で、ゲームアーキテクチャの特定のコンポーネントを複数のリージョンにデプロイしたり、プレイヤーのロケーションに応じてカスタマイズされたエクスペリエンスを提供したりできます。

ゲームにはさまざまなタイプがあり、ゲームのサポートに必要なバックエンドの技術インフラストラクチャは、開発中のゲームのタイプによって異なります。例えば、一般的なタイプのゲームには、ファーストパーソンシューティングゲーム (FPS)、ロールプレイングゲーム (RPG)、超マルチプレイヤーオンラインゲーム (MMOG)、バトルロワイヤル (BR)、スポーツゲーム、パズルゲームなどがあります。また、ターンベースや同時プレイなど、ゲームのアーキテクチャに影響を与えるさまざまなゲームインタラクションモードがあり、パフォーマンス特性も異なります。

ゲームは、デスクトップ、ウェブ、モバイル、コンソール、拡張現実 (AR)、バーチャルリアリティ (VR)、ゲームストリーミングソリューションなどの新しいインタラクションモードを含む 1 つ以上のゲームシステムでプレイするように開発されています。ゲームは通常、クロスシステムゲームプレイをサポートしています。つまり、プレイヤーはゲームの進行状況を保存し、他のシステムでゲームプレイを再開したり、他のシステムでプレイヤーとゲームプレイセッションを開始したりできます。

ビデオゲームの収益化により、ゲームパブリッシャーは、広告、デジタルおよび小売ベースのゲーム購入、マイクロトランザクションと呼ばれるダウンロード可能コンテンツ (DLC) のゲーム内購入、ゲームをプレイするために必要な有料サブスクリプションなどのさまざまな戦略を使用して収益を生成できます。ゲーム業界で最も一般的な主要業績評価指標 (KPIs) には、次のようなものがあります。

- デイリーアクティブユーザー (DAU)
- 毎月のアクティブユーザー数 (MAU)
- 同時ユーザー (CCU)
- セッション期間
- インストールあたりのコスト (CPI)
- プレイヤーのライフタイム値 (LTV)
- ユーザーあたりの平均収益 (ARPU)

ゲームシステム

ビデオゲームは、クライアント入力コントロール、グラフィックス、クライアントソフトウェア (ゲームクライアントと呼ばれる)、ハードウェア、および場合によってはゲームプレイをサポートするシステム排他的な機能を提供するゲームシステムでプレイするように開発されています。

ゲームシステムは通常、次のカテゴリに分けられます。

- コンソール: Sony PlayStation、Microsoft Xbox、Nintendo Switch などの一般的な例など、ゲームをプレイするために設計された専用のエンターテインメントシステム。コンソールは、ゲームシステムプロバイダーによって製造されたコンソールハードウェアに物理またはデジタルで分散されたゲームコンテンツをインストールすることで、ゲームをプレイする機能を提供します。この定義では、コンソールは手持ちまたは静止していて、ホームエンターテインメントのシナリオで使用することを意図しています。
- パーソナルコンピュータ (PC): プレイヤーがカスタマイズできるクライアントマシンにインストールされたコンピュータソフトウェアを使用して再生されるゲーム。このため、PC ゲームは柔軟性と制御性を備えているため、プレイヤーの間で人気があります。
- ウェブ: ウェブブラウザを使用して再生するように設計されたゲーム。通常は、オペレーティングシステムに関係なくプレイヤーがゲームにアクセスできるようにするという利点があります。

- モバイル: 携帯電話、通常はスマートフォンオペレーティングシステムで再生するように開発されたゲーム。モバイルゲームは通常、デジタルアプリストアからダウンロードされ、電話にインストールされます。

前述のシステムに加えて、まだ比較的新しく成長しており、より主要なシステムと比較して市場シェアがはるかに小さい初期システムもあります。このカテゴリのゲームシステムの例には、AR、VR、ゲームストリーミングなどがあります。これはクラウドゲームとも呼ばれます。

ゲームストリーミングでは、クラウドでゲームプレイをレンダリングし、シンクライアント、通常はブラウザにストリーミングします。ゲームストリーミングを使用すると、プレイヤーは完全にリモートでホストされているゲームを、通常はゲームストリーミングサービスプロバイダーによってクラウドでプレイできます。ゲームストリーミングでは、プレイヤーは、クラウドゲームサービスプロバイダー (ゲームシステム) が提供するブラウザまたはシンクライアントを介してクラウドベースのゲームに接続します。

ゲームサーバー

ゲームサーバーは、ゲームのコンピューティングインフラストラクチャの最も重要な側面の 1 つです。専用ゲームサーバーとも呼ばれるゲームサーバーは、マルチプレイヤーゲームの開発時や、ゲームプレイイベントのサーバーによる権威ある処理が必要な場合に使用されます。ゲームサーバーはゲームアーキテクチャの中心にあり、コアロジックが実行される場所として機能します。これには、プレイヤーとゲームの状態の管理、接続されたゲームクライアントとゲームサーバー間のインタラクションの管理が含まれます。ゲームサーバーは通常、ゲームアーキテクチャで最もパフォーマンスの影響を受けやすい要素の 1 つです。ゲームサーバーは、プレイヤーのゲームクライアントからの入力を処理し、接続された他のプレイヤーにリアルタイムで適切に配布する責任があるためです。パフォーマンスの低いゲームサーバーは、ゲームエクスペリエンスの全体的なパフォーマンスに影響します。したがって、ゲームサーバーのパフォーマンスを最適化し、特にゲームの起動時またはゲームプレイのピーク時に十分な容量を提供する必要があります。

このドキュメントでは、ゲームサーバーまたはゲームサーバーインスタンスとは、1 つ以上のゲームサーバープロセスをホストする仮想マシンなどのコンピューティングを指します。ゲームサーバープロセスは、ゲームセッションをホストするゲームサーバービルドの単一のインスタンスを表します。これは、プレイヤーセッションを介してプレイヤーが接続できる実行中のゲームのインスタンスです。このため、ゲームセッションとそれをホストしているゲームサーバープロセスの間に暗黙的な 1 対 1 の関係があるため、ゲームサーバープロセスまたはゲームセッションを互換的に参照することがよくあります。では AWS、コンピューティングがゲームサーバーをホストするための複数のオプ

ションがあり、リソースの伸縮自在なプロビジョニングを通じてスケーラブルなクラウドベースの容量にアクセスできます。

Amazon EC2 は、インスタンスと呼ばれるクラウドベースの仮想サーバーを提供し、Linux および Windows の複数のバージョンをサポートします。インスタンスを作成し、別のサーバーや仮想マシンのように直接管理できます。通常、効率を向上させ、コストを削減するために、複数のゲームサーバープロセスをインスタンスにデプロイします。Amazon EC2 は、コンピューティングインフラストラクチャを最も制御したい場合は、ゲームサーバーに適しています。

Amazon GameLift は、クラウドでの専用ゲームサーバーホスティング用のフルマネージドソリューションと、GameLift FlexMatch によるマッチメーキングなどの追加機能を提供します。GameLift は Amazon EC2 上に抽象化レイヤーを提供し、ゲームサーバー管理を簡単にし、ほとんどの で利用できる AWS リージョン ため、プレイヤーに近いゲームサーバーをホストしてレイテンシーを減らし、高可用性を実現し、スポットインスタンスを使用してコストを大幅に削減できます。GameLift は既存のゲームバックエンドに統合できますが、独自のゲームサーバー管理およびマッチメーキングソリューションを開発したくなく、ゲームの成長に合わせて拡張 AWS できる によって管理されるソリューションを希望するゲーム開発者にとって特に便利です。

Amazon Elastic Container Service (Amazon ECS) は、Docker ベースのコンテナの実行に使用できるフルマネージド型のコンテナオーケストレーションサービスです。Amazon Elastic Kubernetes Service (Amazon EKS) を使用して、Kubernetes を使用して構築された Docker ベースのコンテナを実行することもできます。Amazon ECS や Amazon EKS が提供するコンテナテクノロジーを使用すると、多くのゲームサーバープロセスやその他のゲームアプリケーションインスタンスを EC2 インスタンスに効率的にパッキングすることで、コンピューティング使用率を向上させることができます。

コンテナを使用すると、開発者が開発中にローカルマシンで使用しているのと同じ Docker イメージ操作ランタイムを使用してアプリケーションをホストすることで、開発者の生産性を向上させることもできます。コンテナを実行するためのサーバーレスコンピューティングソリューションであり AWS Fargate、Amazon EKS と Amazon ECS の両方と互換性がある を使用すると、運用オーバーヘッドをさらに削減できます。Fargate は、コンテナが実行される基盤となるインスタンスを運用する責任を負うことなく、コンテナでゲームサーバーを実行するユースケースに最適です。

AWS Outposts を使用してデータセンターまたはオンプレミス施設で AWS インフラストラクチャとサービスを実行できます。これにより、ゲームをオンプレミス環境で実行し AWS 、同じインフラストラクチャを使用してハイブリッドクラウド導入戦略をサポートできます。 AWS ローカルゾーンはの拡張機能として機能し AWS リージョン 、ゲームサーバーやその他のレイテンシーの影響を受けやすいワークロードをプレイヤーや開発チームにより厳密に実行できます。さらに、ゲームサーバー

のグローバルネットワークレイテンシーを減らすために、 AWS Global Accelerator を使用してゲームサーバーへのプレイヤートラフィックのパフォーマンスを向上させることができます。

AWS Lambda はサーバーをプロビジョニングまたは管理せずにコードを実行するサーバーレスコンピューティングサービスであり、ターンベースのゲームや軽量コンピューティング要件、小さなコードベース、ステートレスマイクロサービスアーキテクチャを使用してゲームプレイ機能を設計できるゲームサーバーなどの非同期ゲームサーバーのユースケースに役立ちます。Lambda 関数は、長時間実行されるゲームサーバープロセスを実行するのではなく、リクエストごとにイベント駆動型で実行されることに注意してください。Lambda は、基盤となるアプリケーションが開発者がコードのホストから選択できるため、このホワイトペーパーのオプションを最もランタイムに抽象化できます。

ゲームサーバーホスティングのアプローチを選択するときは、運用オーバーヘッド、レガシーコードベース、パフォーマンス要件、スケールなど、さまざまな要件を考慮してください。EC2 インスタンスとコンテナは、従来のコードベースに適したオプションです。クラウドへの移行に必要な変更は最小限で済み、EC2 インスタンスを使用してコンピューティングインスタンスのリソースを専有できます。一方、コンテナを使用すると、管理と高い使用率の実現が容易になります。サーバーレス関数は最高レベルの抽象化を提供します。これを使用して、イベントに応答してのみ実行されるコードを定義できるため、コストを削減できます。

ゲームクライアント

ゲームクライアントは、プレイヤーがゲームのプレイに使用するソフトウェアとハードウェアデバイスを表します。ゲームクライアントは、プレイヤーの入力を処理のためにサーバーに送信されるメッセージに変換するためのソフトウェアを提供し、サーバーからの受信レスポンスを処理し、グラフィックなどの出力をプレイヤーにレンダリングします。リアルタイムのネットワークマルチプレイヤーゲームでは、ゲームクライアントは通常、ゲームプレイセッション中にゲームサーバーへの永続的なネットワーク接続を維持し、ネットワークレイテンシーを減らし、処理時間を最小限に抑えます。ただし、ゲームクライアントは REST を使用してゲームサーバーまたはバックエンドサービスとやり取りすることもできます。

メッセージング

通常、ゲームのメッセージには主に 3 つのカテゴリがあります。

- ゲームの招待やプッシュ通知など、特定のユーザーまたはユーザーのグループを対象としたプレイヤーエンゲージメントメッセージング
- ゲーム内チャットなどのプレイヤー間のグループメッセージング

- 2 つ以上のアプリケーションを統合するために使用される JSON メッセージなどのService-to-serviceメッセージング

これらのタイプのメッセージを送受信するための一般的な戦略は、パブリッシャーサブスクライブと非同期処理アーキテクチャパターンを使用することです。 は、ゲームでのメッセージングの実装に役立ついくつかのサービス AWS を提供します。

- Amazon Simple Notification Service (SNS): パブ/サブアーキテクチャパターンを使用してパブリッシャーとサブスクライバーの間でメッセージを配信するためのマネージドサービス。パブリッシャーは、API を使用して Amazon SNS にメッセージを送信します。Amazon SNS は、サブスクライブしているアプリケーションにメッセージを非同期的に配信し、最も広く使用されているプッシュ通知サービスをサポートするモバイルクライアントまたはデスクトップに直接プッシュ通知を配信できます。Amazon SNS は、クライアントへのプッシュ通知やservice-to-serviceメッセージングのユースケースに使用できます。
- Amazon Simple Queue Service (SQS): それぞれで使用されるプログラミング言語に関係なく、ゲームサーバーとゲームを簡単に統合できるフルマネージドキューサービス。多くのゲームタスクは、データベース内のリーダーボードやプレイタイム値の更新など、バックグラウンドで分離して処理できます。このアプローチは、ゲームのさまざまな部分を切り離し、バックエンド処理からプレイヤー向け機能を独立してスケーリングする効果的な方法です。
- Amazon Managed Streaming for Apache Kafka (MSK): 一般的なオープンソースソリューションである Apache Kafka を使用してデータストリーミングとプロデューサーまたはコンシューマーアプリケーションの構築を簡素化するフルマネージドサービス。Kafka は通常、リアルタイムストリーミングデータの取り込みと処理に使用され、service-to-serviceメッセージングに使用できます。
- Amazon ElastiCache (Redis OSS): チャットルームアプリケーションや高性能service-to-serviceメッセージングの開発に一般的に使用される Redis の一般的なパブ/サブ機能のサポートを含む、フルマネージドのインメモリデータストアを提供します。Redis はリストやセットなどの豊富なデータ型もサポートしているため、デベロッパーは Redis を使用して高性能キューイングを実行できます。
- Amazon Pinpoint: E メール、SMS、音声、プッシュ通知を通じてユーザーエンゲージメントメッセージングを提供します。例えば、Amazon Pinpoint を使用してユーザーエンゲージメントメッセージをプレイヤーに配信し、ゲームに招待したり、多要素認証トークンのサポート、注文確認、パスワードリセット E メールなどのトランザクションユースケースに使用できます。

ライブゲームオペレーション (Live Ops)

ライブゲームオペレーション (Live Ops) は、ゲームをライブサービスとして扱い、起動したゲームに新機能、更新、プロモーション、ゲーム内イベント、改善を継続的に提供して、プレイヤーコミュニティのエクスペリエンスを向上させるゲーム管理とオペレーションのスタイルです。

従来、ゲームは サービスではなく製品として配信され、新しいコンテンツや機能は、起動された製品ではなく後続のリリースや続編に頻繁に組み込まれていました。ゲーム管理に対する Live Ops アプローチを使用すると、ゲーム運用チームは、実験、プロモーション、ゲーム内イベント、イノベーションを通じてゲームを起動し、エンゲージメントの高いプレイヤーコミュニティを維持し、プレイヤーを楽しませることができます。

このアプローチには、新しいプレイヤーエンゲージメント戦略を開拓し、定期的な収益ストリームを提供するという利点がありますが、運用上の専門知識がさらに必要です。例えば、Live Ops 戦略を成功させるには、開発者がクラウドサービスと統合するか、独自のバックエンド技術インフラストラクチャを運用する必要があります。また、プレイヤーエクスペリエンスに悪影響を及ぼす可能性のあるゲーム内またはプレイヤーコミュニティ内で発生する問題を特定して対応するための効果的な方法も必要です。

オペレーショナルエクセレンス

運用上の優秀性の柱は、クラウドベースのゲームを大規模にデプロイして運用するためのベストプラクティスに焦点を当てています。運用上の優秀性に焦点を当てて、ポジティブなプレイヤーエクスペリエンスを維持し、そのエクスペリエンスに影響を与える問題に備え、そこから回復するための予防措置を講じることが重要です。

焦点

- 設計原則
- ライブオペレーション
- アカウント構造
- ゲームデプロイ
- ヘルスモニタリング
- 負荷テスト
- 継続的最適化
- リソース

設計原則

Well-Architected Framework ホワイトペーパーの設計原則に加えて、以下の設計原則はゲームの構築と運用における運用上の優秀性を実現するのに役立ちます。

- ゲーム運用チームの測定可能で達成可能な目標を定義し、必要に応じて適応する: ゲームのヒット主導型の性質上、ゲームの起動時にゲームをプレイするプレイヤーの数や、進行中のゲーム運用に対するプレイヤーの期待値を事前に判断することは困難です。利害関係者と野心的だが達成可能な目標を設定し、ゲームが射影を超えた場合にスケールアップし、ゲーム開発チームがプレイヤーエクスペリエンスを最適化している間にスケールダウンできるアプローチを設計することが重要です。これらの要件を満たすために事前に十分な準備とテストを行い、ビジネス関係者と技術関係者をゲームを運用するための目標に合わせます。ターゲットを定義すると、ゲームチームはゲームバックエンドインフラストラクチャの計画、設計、プロビジョニング、テスト、デプロイ、運用中にコストとパフォーマンスの適切なバランスを実現できます。
- 運用ランブックを使用して、ゲームのローンチや特別なイベントに関連するスケーリングアクティビティを計画する: ゲーム運用チームは、ビジネスステークホルダーと連携して、イベントの予想されるピークプレイヤーの同時実行数の予測をモデル化し、事前にインフラストラクチャ容量を

事前スケーリングするためのプロアクティブな計画を実行する必要があります。イベント中のプレイヤートラフィックは変動するため、事前の計画とスケーリング前のアクティビティによって既存の自動スケーリングシステムが強化され、イベント中の成功の可能性が向上し、プレイヤーにポジティブなエクスペリエンスを提供するのに十分なリソースがあることを確認する必要があります。パフォーマンスエンジニアリングプラクティスを実装して、リソースのベースラインとシステムの容量に関するデータ駆動型の理解を開発します。これにより、スケーリング前のアクティビティと自動スケーリング設定のガイドに役立ちます。プロセスの一貫性を確保するために運用ランブックを作成します。この高度な計画とプレイヤーの需要への対応性は、ライブサービスゲームにとって特に重要です。ライブサービスゲームでは、アクティブでエンゲージメントの高いプレイヤーベースを長期間にわたってサポートするために、信頼性の高いパフォーマンスとインフラストラクチャを維持する必要があります。

- プレイヤーサポートリクエストを受信、調査、応答するための運用モデルを確立します。起動後、ゲームに関する苦情と問題のレポートを監視します。コミュニティフォーラム、ソーシャルメディア、E メール、チケットシステム、コールセンター、自動チャットボットソリューションなど、プレイヤーの問題を適切に解決するために、安全で効果的な方法でプレイヤーとやり取りするための適切なシステムを実装します。これは、プレイヤーベースとの継続的なコミュニケーション、変化するニーズに対応するためのプレイヤーフィードバックへの応答性、および長期間にわたるエンゲージメントのあるコミュニティの維持を必要とするライブサービスゲームにとって特に重要です。

ライブオペレーション

GAMEOPS01: ゲームのライブオペレーション (Live Ops) 戦略をどのように定義しますか?

ビジネスステークホルダーと相談して、定義された目標とパフォーマンスメトリクスに基づいてゲームのライブオペレーション (Live Ops) 戦略を策定します。

ベストプラクティス

- GAMEOPS01-BP01 ゲーム目標とビジネスパフォーマンスメトリクスを使用してライブ運用戦略を開発する

GAMEOPS01-BP01 ゲーム目標とビジネスパフォーマンスメトリクスを使用してライブ運用戦略を開発する

ゲームプロデューサーやパブリッシングパートナーなどのビジネスステークホルダーに相談して、ゲームの目的とパフォーマンスメトリクスを決定します。これにより、メンテナンスウィンドウ、ソフトウェアとインフラストラクチャの更新スケジュール、システムの信頼性と回復性の目標の定義など、ゲームの管理方法の計画を立てることができます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

これらのメトリクスは、ゲームのヘルスをモニタリングし、直接的なゲームフィードバックを収集し、合理化され自動化されたリリースプロセスを構築するために、ゲームのライフサイクルのどの段階でライブオペレーションストリーム (Live Ops) を組み込むべきかを判断するのに役立ちます。たとえば、新しいゲームは、アクティブなプレイヤー数、収益、または別のメトリクスセットによって測定される特定のスケールが達成されるまで待ってから、専用のライブオペレーションチームを設定する場合があります。確立されたゲーム開発スタジオには、他のゲームのライブオペレーション経験が既にある可能性があるため、新しいゲームをオンボードするだけで済みます。

実装手順

- ゲームインフラストラクチャが効果的にサポートできるプレイヤー同時実行 (CCU) と、毎日および毎月のアクティブユーザー (DAU および MAU) のターゲット、インフラストラクチャの予算、財務目標、およびプレイヤーのエンゲージメントを高めるためのコンテンツや機能のリリース頻度などのその他のパフォーマンス目標を定義できます。これらの目標とメトリクスは、効率的な運用に必要なゲーム設計、リリース管理、オブザーバビリティ、サポートに関する決定につながっています。
- ゲームには、リリース中のダウンタイムなしで、毎月少なくとも 1 回新しいコンテンツ更新をリリースする目的がある場合があります。この情報は、リリースデプロイ戦略を定義し、1 か月間の他の時間帯にダウンタイムが必要になる可能性のある必要なメンテナンスのスケジュールを調整し、可用性 SLA に貢献するのに役立ちます。

アカウント構造

GAMEOPS02: ゲーム環境をホスト AWS アカウント するための をどのように構築しますか?

マルチアカウント戦略を実装して、さまざまなゲーム環境を分離し、セキュリティ、運用効率、スケーラビリティを強化します。 AWS Organizations を使用して、アカウントの階層を管理し、アカウントにガードレールを適用し、デプロイされたリソースにタグポリシーとタグ付けを適用します。

ベストプラクティス

- GAMEOPS02-BP01 マルチアカウント戦略を採用して、さまざまなゲームやアプリケーションを自分のアカウントに分離する
- GAMEOPS02-BP02 リソースタグ付けを使用してインフラストラクチャリソースを整理する

GAMEOPS02-BP01 マルチアカウント戦略を採用して、さまざまなゲームやアプリケーションを自分のアカウントに分離する

各環境のセキュリティ、分離、運用上のニーズに合わせてインフラストラクチャのデプロイをガイドするアカウント構造を設計します。アクセスを制限し、必要な AWS サービスのみの使用を許可することで環境を分離することは不可欠です。本番環境はロックダウンされ、開発環境とテスト環境は実験を許可する寛容です。各環境の主要なサブシステムと、複数の環境で使用され、独自の AWS アカウント 環境でホストおよび管理される一般的なサービスをさらに分離することを強くお勧めします。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

さまざまな環境 (開発、テスト、ステージング、本番稼働、共有サービスなど) を個々の に分離 AWS することで、 にマルチアカウント戦略を採用し AWS アカウント、インシデントの範囲を縮小します。オペレーションをさらに簡素化し、アカウントレベルおよび組織単位レベル (OU レベル) ポリシーを選択的に定義して適用 AWS アカウント するには、 の階層を一元管理 AWS Organizations することを検討してください。開発および本番ワークフローのニーズに合わせた適切な OU と AWS アカウント 構造を設計することで、コストを最適化し、スケーラビリティを向上させることができます。

- マルチアカウント戦略を採用する: 環境を分離してインシデントの半径を減らし、運用を簡素化します。
- 使用 AWS Organizations: アカウントを階層的に管理し、ポリシーを適用し、一元化されたガバナンスを有効にします。

- スケーラビリティの計画: きめ細かなアカウント構造を設計し、将来の成長のためのコスト削減策を実装します。

実装手順

にデプロイされるゲームシステムは、論理的に整理された複数のアカウントを使用して適切な分離を提供する AWS 必要があります。これにより、ゲームインフラストラクチャのスケールに応じて問題の発生範囲が軽減され、操作が簡素化されます。ゲームインフラストラクチャをホスト AWS アカウント する は通常、次の論理環境にグループ化されます。

- ゲーム開発環境は、開発者がゲーム用のソフトウェアとシステムを開発するために使用します。
- テスト環境または品質保証 (QA) 環境は、統合テスト、手動 QA、およびその他の自動テストを実行するために使用されます。
- ステージング環境または本番稼働前環境は、完成したソフトウェアのホスティングに使用されるため、本番稼働を開始する前に負荷テストと煙テストを実施できます。
- ライブ環境または本番環境は、ライブソフトウェアとインフラストラクチャをホストし、プレイヤーからの本番トラフィックを処理するために使用されます。
- 共有サービスまたはツール環境は、多くの異なるチームが使用する一般的なシステム、ソフトウェア、ツールへのアクセスを提供します。たとえば、中央のセルフホスト型ソースコントロールリポジトリとゲームビルドファームが共有サービスアカウントでホストされている場合があります。
- セキュリティ環境は、クラウドセキュリティに重点を置いたチームが使用する一元化されたログとセキュリティテクノロジーを統合するために使用されます。

のゲームインフラストラクチャでは AWS、ゲーム環境 (開発、テスト、ステージング、本番稼働) ごとに個別のアカウントを作成し、セキュリティ、ログ記録、中央共有サービス用のアカウントを作成することをお勧めします。

通常、限られた数のインフラストラクチャリソース、通常は数百台以下のサーバーを管理する小規模なゲーム開発スタジオでは、これらの環境 AWS アカウント (1 つの本稼働アカウント、1 つの開発アカウント、1 つのステージングアカウントなど) ごとに 1 つの を作成できます。ただし、ゲームインフラストラクチャやチームサイズが時間の経過とともに増加するにつれて、この簡略化されたモデルはうまくスケーリングされない可能性があります。

これらの環境を設定するときは、多くの AWS サービスが特定のリージョン内のアカウント全体のリソースと API レベルの Service Quotas を共有することを検討してください。これは、アカウントを論理的に整理する方法を決定する際に考慮する必要があります。 では、アカウントにデプロイさ

れたサービスを消費するためのコスト AWS アカウント のみが発生します。したがって、これにより、特にゲームが大きくなり、より多くのデベロッパーがリソースを構築および管理するためのアクセスを必要とするにつれて、リソースの競合とサービスクォータを効果的に減らすことができます。

通常、数百人の開発者がリソースにアクセスする数千台のサーバーを運用する大規模なゲーム開発スタジオでの経験に基づいて、ゲームをサポートする個々のアプリケーションが独自の開発、テスト、ステージング、本番稼働アカウントを持つ、よりきめ細かなアカウント構造を設計することをお勧めします。ライブシステムの計画と移行が複雑なため、ゲームの開始後に AWS マルチアカウント戦略を再設計するのは難しく、時間がかかるため、適切なマルチアカウント構造を決定する際には、将来のスケーリングのニーズを考慮してください。

AWS Organizations を使用して階層とグループ化を設定し AWS アカウント、組織単位 (OUs) を定義して、サービスコントロールポリシー (SCPs) を通じて共通の OU レベルのポリシーを適用できます。 は、リソースの拡大とスケーリングに合わせて環境を AWS Organizations 一元的に管理および管理します。プログラムで新しいアカウントを作成し、リソースを割り当て、アカウントをグループ化してワークフローを整理し、ガバナンスのためにポリシーをアカウントまたはグループに適用し、アカウントに 1 つの支払い方法を使用することで請求を簡素化できます。さらに、Organizations は他の サービスと統合されているため、組織内のアカウント間での一元的な設定、セキュリティメカニズム、監査要件、リソース共有を定義できます。

AWS Control Tower では、ランディングゾーンと呼ばれる安全なマルチアカウント環境を簡単にセットアップして管理できます。Control Tower は を使用してランディングゾーンを作成し AWS Organizations、継続的なアカウント管理とガバナンス、および何千人もの顧客とのクラウドへの移行 AWS経験に基づく実装のベストプラクティスを提供します。AWS Config、AWS Trusted Advisor、 AWS Security Hub CSPMは、アカウントの衛生状況を集約または一元的に把握できるサービスです。

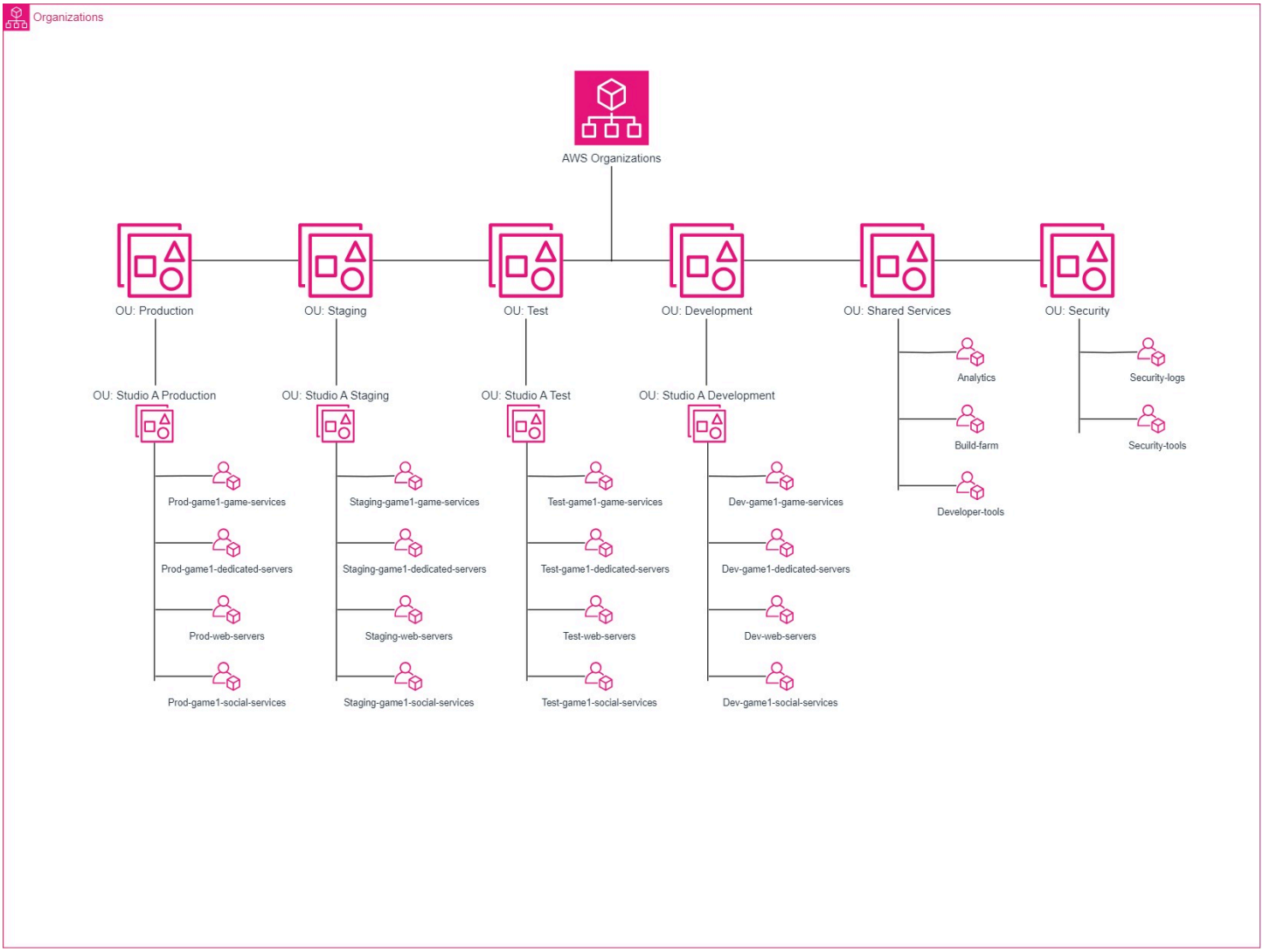
この分離は、各ゲーム環境にカスタムまたは個々のアクセス許可とガードレールを設定するのに役立ちます。本番稼働用アカウントには必要なガードレール、アクセス制限、モニタリングとアラート、セキュリティツールが必要ですが、非本番稼働用アカウントには同じレベルのガードレールとアクセス許可を必要としない場合があります。非本番環境を自動化して、数時間後にリソースをシャットダウンし、コストを削減できます。この詳細レベルでアカウントを分離すると、ゲームをサポートする各環境のインフラストラクチャコストを簡単にモニタリングできます。

以下は、 AWS Organizations と組織単位 (OUs) を使用して、別々の環境とスタジオ AWS アカウント に論理的にグループ化するゲーム会社のマルチアカウント構造の例です。この例では、OUs を使用して、環境に基づいてアカウントをグループ化し、次に環境を運用するスタジオに基づいてアカウントをグループ化します。これは、ネストされた階層を作成して、環境内の独自のアカウント (OUs

と表記) に個別のアプリケーションとゲームをデプロイできるようにする方法を示しています。これは、複数のゲームを開発して運用する場合に便利です。この柱のリソースセクションで提供されているドキュメントとホワイトペーパーを参照して、マルチアカウント戦略を整理するために検討できる追加の戦略を確認してください。

上記の説明に基づいて、以下のサンプル図は、4 つのステージ (開発、テスト、ステージング、本番稼働) で構成される開発パイプラインを持つゲームスタジオ (組織) を想定しています。特定のゲーム (game1) では、各環境 (OU) にゲームサービス、専用ゲームサーバー、ソーシャルサービス、ウェブサーバー AWS アカウント 用の個別の があります。各 で実行されるリソース AWS アカウント は、それぞれのサブシステムに関連しています。通常、この種の開発パイプラインを使用する個々のゲームは、この構造または同様の構造をレプリケートします AWS アカウント。

これらのゲーム中心の環境 OUs に加えて、共有サービス OU とセキュリティ OU もあります。これらの OUs は、個々のゲームではなく、組織全体にする必要があります。これにより、ゲームは、この例のように、開発ツールとデータと分析の共有サービスを消費します。次に、アプリケーションログとシステムログをセキュリティ OU のログ用に AWS アカウント セットアップされた に送信します。



ゲーム環境のアカウント構造の例

GAMEOPS02-BP02 リソースタグ付けを使用してインフラストラクチャリソースを整理する

でインフラストラクチャリソースを効果的に管理および追跡するには AWS、適切なリソースのタグ付けとグループ化を使用して、各リソースの所有者、プロジェクト、アプリケーション、コストセンター、およびその他のデータを特定します。タグ付けされたリソースは、運用サポートに役立つリソースグループを使用してグループ化できます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

タグ付けポリシーを定義します。一般的な戦略には、チーム名や個人名、ゲーム、アプリケーション、プロジェクトの名前、スタジオ名、環境 (開発や本番稼働など) 、リソースのロール (データベースサーバー、ウェブサーバー、専用ゲームサーバー、アプリケーションサーバー、キャッシュサーバーなど) など、リソース所有者を特定するためのリソースタグが含まれます。ビジネスや IT のニーズを支援するために、他のタグを追加できます。 AWS Configは、リソースの作成時と更新時にタグ付けポリシーを適用することもできます。タグとリソースグループは、 AWS マネジメントコンソール、、 AWS CLIおよび API オペレーションから入手できます。

実装手順

- リソースにタグを付けて、所有者、プロジェクト、アプリケーション、コストセンター、その他の関連データを特定します。
- 所有者、プロジェクト、スタジオ、環境、リソースロールのタグを含むタグ付けポリシーを実装します。
- AWS Config を使用してタグ付けポリシーを適用し、 AWS マネジメントコンソール、CLI、API を使用してタグを管理します。

ゲームデプロイ

GAMEOPS03: ゲームデプロイはどのように管理しますか?

再利用可能なコンポーネントを徹底的に検証し、定期的なパフォーマンスエンジニアリングを実施し、開発ライフサイクル全体で定期的な負荷テストを実装することで、ゲームのデプロイを管理します。

ベストプラクティス

- GAMEOPS03-BP01 ゲームで再利用する前に、既存のコアゲームシステムとインフラストラクチャを検証してテストする
- GAMEOPS03-BP02 各リリースの前にパフォーマンスエンジニアリングを実施する (または少なくともメジャーリリースの場合)
- GAMEOPS03-BP03 負荷テストを早期かつ頻繁に行う
- GAMEOPS03-BP04 プレイヤーへの影響を最小限に抑えるデプロイ戦略を採用する

- GAMEOPS03-BP05 ピーク要件をサポートするために必要なプリスケールインフラストラクチャ

GAMEOPS03-BP01 ゲームで再利用する前に、既存のコアゲームシステムとインフラストラクチャを検証してテストする

組織は、開発時間とコストを削減するために、以前のゲームの既存のコンポーネントとソースコードを再利用する傾向があります。これらのレガシーコンポーネントとコードは、徹底的なレビューを受けたり、詳細な統合テストを受けたりせず、過去のパフォーマンスに依存する可能性があります。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

再利用は生産性の向上に役立ちますが、過去のパフォーマンスや安定性の問題が新しいプロジェクトに再導入されるリスクもあります。したがって、以前のゲームの既存のコンポーネントとソースコードを再利用する場合は、堅牢なテストを実装する必要があります。

実装手順

- 再利用されたコードとコンポーネントを特定する: 以前のゲームで再利用されているソースコード、ライブラリ、コンポーネントをカタログ化します。アクティブに維持されているコードと非推奨のコードを明確に区別する
- 元の動作と既知の問題を文書化する: 元のパフォーマンス特性、機能制限、および再利用されたコンポーネントに関連する既知のバグや本番稼働用インシデントを記録します。
- 詳細なコードレビューを実行する: 再利用されたコンポーネント、特に過去に問題があったコンポーネントや文書化が不十分なコンポーネントの詳細な技術レビューを実施します。
- 高リスクのレガシーコンポーネントを置き換えるまたはリファクタリングする: 本番環境の回避策に頼るのではなく、問題履歴があるレガシーコンポーネントの置き換えまたは更新を優先します。
- 統合と互換性のテストを実施する: 新しいゲームのシステムのコンテキスト内で再利用されたコンポーネントを検証します。新しいモジュール、ツール、APIs と適切にやり取りしていることを確認します。

GAMEOPS03-BP02 各リリースの前にパフォーマンスエンジニアリングを実施する (または少なくともメジャーリリースの場合)

パフォーマンスエンジニアリングは、アプリケーションのパフォーマンスをさらに向上させる最適化の機会を発見するために、アプリケーションの複数の主要な運用メトリクスをモニタリングするプロ

セスです。これは、テストから始まり、コード、その依存関係、関連プロセス、ホストオペレーティングシステム、基盤となるインフラストラクチャを最適化する反復プロセスです。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

アプリケーションのパフォーマンスをより詳細に分析するには、アプリケーションのパフォーマンスモニタリング (APM) またはデバッグツールをアプリケーションコードに統合します。これにより、アプリケーションのフロー全体で異常の動作を追跡することで、問題を分離し、トラブルシューティング時間を短縮できます。APM ツールでは、パフォーマンスの遅いメソッドや外部オペレーションを特定することもできます。

AWS X-Ray は、パフォーマンスのボトルネックの特定、本番稼働エラーの分析とデバッグなど、開発者のパフォーマンスエンジニアリングアクティビティを支援します。X-Ray を使用して、アプリケーションとその基盤となるサービスのパフォーマンスを理解し、パフォーマンスの問題やエラーの根本原因を特定してトラブルシューティングできます。アプリケーションとそのインフラストラクチャが合成プレイヤートラフィックで徐々にロードされる負荷テストの多数のラウンドを通じて、さまざまなシステムのボトルネック、アプリケーションエラー、例外、OS の問題、およびその他の問題が他の QA テスト中には検出されなかった可能性があります。

ゲームのローンチ、コンテンツリリース、プロモーション、ゲーム内の主要なイベントなどの重要なイベントには、Countdown を使用しますAWS 。Countdown は、運用準備状況の検証、潜在的なリスクの軽減、容量のニーズの計画を行うために、ゲームの専門家によって構築されたプレイブックに基づいて実装ガイダンスを提供します。AWS Countdown にはプレミアムサポートオプションもあり、サポートを強化し、エンジニアなどのオプションでインフラストラクチャを最適化できます。

実装手順

- パフォーマンスエンジニアリングでは、主要な運用メトリクスを評価してモニタリングし、アプリケーションのコード、プロセス、オペレーティングシステム、インフラストラクチャが期待どおりに機能していることを確認します。本番稼働前レビューは、さまざまなレベルのシミュレートされた使用状況でベースラインパフォーマンスを定義するのにも役立ちます。
- sar、top、vmstat、sysstat、netstat、Performance Monitor などのシステムツールを使用して、使用率、サービス、I/O、プロセスなどの主要なメトリクスを検出して追跡します。
- などの APM ツールを使用してアプリケーションのパフォーマンスと動作を追跡 AWS X-Ray し、問題の分離、ボトルネックの特定、本番稼働エラーのデバッグを行います。

- ゲームのローンチなどの重要なイベントについては、Countdown (IEM) に AWS サブスクライブして、アーキテクチャと運用のガイダンス、オンデマンドの運用サポート、リスクの特定、緩和策の計画を行います。

GAMEOPS03-BP03 負荷テストを早期かつ頻繁に行う

負荷テストは、システム上の実際のトラフィックをシミュレートして信頼性とパフォーマンスを評価するプロセスです。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

負荷テストは、リソースのパフォーマンスベースラインを開発し、システムの容量を理解する上で重要な要素です。これにより、財務予測、アーキテクチャ設計、リソース割り当て、自動スケーリング設定、起動後のスケーリング前アクティビティをガイドできます。その他の利点には次が含まれます。

- 最適化インフラストラクチャ: リソースが過剰またはプロビジョニング不足である可能性があります。必要なリソースを理解することで、コストを削減し、管理するインフラストラクチャを減らすことができます。
- スケーラビリティの準備状況: 特定のメカニズムと機能により、ユーザーはすぐにゲームに参加できます。いつどのようにスケールするかを知ることは、需要の増加を適切に満たすこととプレイヤーを失うことの違いになる可能性があります。負荷テスト結果を使用して、さまざまなスケーリングレベルでシステムしきい値、アラートポイント、重要なアラートポイントを含むランブックを準備します。
- 高品質のコード: サービス間の過剰なクロストーク、バッチ処理されていないデータベース呼び出し、非効率的なアルゴリズム、メモリリーク、サービス低下の問題などの問題は、大規模な識別が簡単な場合があります。
- 動作の検証: さまざまな種類の障害をテストに挿入すると、システムの予想される動作を検証したり、修正が必要な エラー処理の問題を発見したりできます。

開発者は開発プロセス全体で複数の時点で負荷テストを実行するのが理想的です。それぞれがさまざまな利点をもたらす可能性があるためです。早い段階では、アーキテクチャ上の決定とリファクタリング作業を導き、変更をより安価で簡単に行うことができます。各スプリントまたはイテレーションの最後に、最新の機能を使用してアプリケーションのパフォーマンスを検証します。

本稼働環境にデプロイする前に、予想される実際の使用パターンをシミュレートした大規模な負荷テストにより、本稼働ワークロードを処理するシステムの能力が確認されます。デプロイ後、定期的な負荷テストはシステムのパフォーマンスをモニタリングし、時間の経過とともに発生する可能性のある変更やボトルネックを特定します。

プレイヤートラフィックをシミュレートするには、ゲームクライアントフローをエミュレートし、ゲームバックエンドとトランザクションして実際のプレイヤーの動作をシミュレートする軽量クライアントまたはボットが必要です。このデータは、通常、ゲームプレイログと人間主導の QA テストによって生成されたデータ、および実際のプレイヤーがゲームの早期アクセスビルドをプレイするように招待される実際の限定スケールアルファまたはベータテストによってキャプチャされます。

将来の潜在的な障害のトラブルシューティングを支援し、将来の負荷テストと比較できるパフォーマンスメトリクスを保持するために、システムの動作を運用ランブックに記録することが重要です。また、ボットが識別に失敗し、メトリクスが反映されない問題を検出する可能性があるため、ロードテスト中に人間の QA 担当者がゲームをテストすることをお勧めします。

AWS Fault Injection Service は、アプリケーションのパフォーマンス、オブザーバビリティ、回復性を向上させるために簡単にフォールトインジェクション実験を実行するためのフルマネージドサービスです。フォールトインジェクション実験は、カオスエンジニアリングで使用されます。これは、CPU やメモリ消費量の急増などの破壊的なイベントを作成し、システムの反応を監視し、改善を実装することで、テスト環境や本番環境でアプリケーションにストレスを与える方法です。フォールトインジェクション実験は、チームが分散システムで見つけるのが難しい隠れたバグを発見し、死角をモニタリングし、パフォーマンスのボトルネックを発見するために必要な実際の条件を作成するのに役立ちます。

実装手順

- Guidance for Kubernetes-Bases ゲームロードテストを使用して、分散負荷テスト環境を設定します。
- 提供されたデプロイファイルを使用して、EKS クラスター内で Locust コントロールポッドとワーカーポッドをカスタマイズしてデプロイし、スケーラブルで管理可能なロード生成を可能にします。
- 負荷テスト中のシステム動作とメトリクスを運用ランブックに記録して、今後のトラブルシューティングを支援し、パフォーマンスベースラインを確立します。
- フォールトインジェクション実験を使用して、実際の中断をシミュレートし、システムのパフォーマンス、オブザーバビリティ、回復力の隠れた問題を発見します。

GAMEOPS03-BP04 プレイヤーへの影響を最小限に抑えるデプロイ戦略を採用する

ゲームソフトウェアとインフラストラクチャのデプロイ戦略を組み込み、プレイヤーをゲームから遠ざけるダウンタイムを最小限に抑えます。特定のタイプの更新では、ゲームクライアントに新しい更新をインストールする必要がある場合がありますが、デプロイ中のダウンタイムを最小限に抑えるか、回避するようにゲームを設計します。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

ゲームデプロイ戦略を開発する際に考慮すべき最も重要なステップの 1 つは、ゲームインフラストラクチャの管理方法を決定することです。AWS CloudFormation や Hashicorp の Terraform などの Infrastructure as Code (IaC) ツールを使用してゲームインフラストラクチャを管理し、環境の準備中のヒューマンエラーを減らします。インフラストラクチャテンプレートは自動パイプラインにデプロイしてテストできるため、さまざまなゲーム環境の設定に一貫性が生まれます。

ゲームに使用できるデプロイ戦略がいくつかあります。

ローリング置換

デプロイのローリング置換の主な目的は、ゲームをシャットダウンしたり、プレイヤーに影響を与えたりせずにリリースを実行することです。実行するアップグレードまたは変更は下位互換性があり、システムの以前のバージョンの近くで動作することが重要です。

このデプロイでは、サーバーインスタンスは、更新されたバージョンを実行しているインスタンスによって段階的に置き換えられます (置換またはロールアウト)。このローリング置換は、いくつかの異なる方法で実行できます。たとえば、専用ゲームサーバーのフリートにローリング更新を実装するには、一般的なアプローチとして、デプロイされた新しいゲームサーバービルドバージョンを含む EC2 インスタンスの新しい Auto Scaling グループを作成し、この新しいサーバーフリートでホストされているゲームセッションにプレイヤーを徐々にルーティングします。新しいゲームサーバービルドを使用するための前提条件として必要な関連するゲームクライアント更新がある場合は、検証チェックを含めて、この新しいゲームクライアント更新がインストールされているプレイヤーのみがこれらのゲームセッションにルーティングされていることを確認する必要があります。

古いゲームサーバービルドバージョンを含むサーバーフリート (EC2 Auto Scaling グループなど) は、通常、ゲームオペレーションチームがこのプロセスを自動化できるようにする個別サーバーメトリクスを設定することで、アクティブなプレイヤーセッションを正常にドレインした後にのみサービ

スから削除されます。または、インフラストラクチャの量とローリングデプロイの実行時間を短縮するために、既存の本番稼働用インスタンスがサービスから削除され、新しいゲームサーバービルドで更新され、本番稼働用フリートに戻される代替アプローチを実行できます。このアプローチにより、必要なインフラストラクチャの量が減りますが、サーバーの交換に伴ってプレイヤーが利用できるライブゲームサーバーの数が減るため、リスクも高くなります。

このモデルは、ゲームプレイをホストしないデータベース、キャッシュ、アプリケーションサーバーなどのバックエンドサービスへのローリングデプロイを実行するためにも使用できます。これらのサービスが複数のクラスター化されたインスタンスで高可用性の方法でデプロイされている限り、これらのサービスへのデプロイの複雑さは、専用のゲームサーバーへのデプロイよりも小さくする必要があります。

ブルー/グリーンデプロイ

ゲームでのブルー/グリーンデプロイの主な目的は、ダウンタイムを最小限に抑え、問題が特定された場合に以前のデプロイへの安全なロールバックを可能にすることです。ゲームバックエンドの 2 つのバージョンに互換性があり、プレイヤーに同時にサービスを提供できるデプロイに適しています。

ブルー/グリーンデプロイ戦略では、2 つの同一の環境 (ブルーとグリーン) が設定されます。既存のゲームバージョンには青のラベルが付けられ、デプロイターゲットである新しいゲームバージョンには緑のラベルが付けられます。グリーン環境を移行する準備ができたら、フェイルバックが必要な場合に備えて古い環境 (ブルー) を使用できるようにしながら、トラフィックをグリーン環境に移行するようにルーティングレイヤーを設定できます。このシナリオでは、ルーティング更新でマッチメーキングサービスを更新して新しいフリートへのゲームセッションの送信を開始するように設定する必要がある場合があります。ゲームバックエンドサービスの場合、サービスの Amazon Route 53 の DNS レコードを更新したり、アプリケーションロードバランサーの重みをシフトして新しいターゲットグループにトラフィックを送信したりする場合があります。

Blue/Green デプロイ戦略の欠点の 1 つは、デプロイの実行に必要な追加のインフラストラクチャによるスタンバイ環境の固有のコストです。この追加のインフラストラクチャコストを削減するオプションは、新しいゲームソフトウェアが本番環境に既にデプロイされているのと同じサーバーにデプロイされる Blue/Green デプロイのバリアントを採用することを検討することです。このシナリオでは、新しいグリーンサーバープロセスを既存のブルーサーバープロセスとともに新しいソフトウェアで開始できます。カットオーバーは、個別の物理インフラストラクチャ間ではなくサーバープロセス間で行われます。このアプローチでは、新しいサーバーがクラウドで起動されるのを待つ必要がなくなるため、大量のインフラストラクチャにまたがるゲームのデプロイを高速化することもできます。このデプロイアプローチのベストプラクティスについては、「 でのブルー/グリーンデプロイ」を参照してください AWS。

Canary デプロイ

Canary デプロイは、ゲームの早期アルファビルドまたはベータビルドをリリースしたり、新しいゲームモード、マップ、チャレンジなどのゲーム機能を実稼働中の制限付きまたは小さなプレイヤーのセットにリリースしたりするために戦略を適用できるため、ゲーム開発者にとって便利です。このようなデプロイは Canary と呼ばれます。リリースには追加の追跡とレポートが含まれている可能性があるため、実際のプレイヤーがそのゲームや機能をプレイすると、ゲームプレイテレメトリが収集され、異常や問題が分析されます。

新機能の場合、プレイヤーはこれについて一貫して通知されないため、ゲームテレメトリはプレイヤーに問題が発生しているかどうかを判断するために使用される主要なソースであり、リリースをロールバックする必要があります。同時に、重大な問題が特定されない場合、この機能をさらに多くのプレイヤーにロールアウトして追加データを取得できます。プレイヤーに通知された場合は、そのエクスペリエンスに関する定期的なフィードバックを提供するように求められます。このようなテストアクティビティは、ライブ運用チームが調整するのが理想的です。

戦略として、Canary デプロイを標準リリースに使用して、プレイヤーが徐々に新機能を利用できるようにすることもできます。標準の Blue/Green 環境よりも潜在的な利点は、フルスケールの 2 番目の環境を必要としないことです。新しいスケールダウン環境の容量によって、新機能にオンボードするプレイヤーの数が決まります。プレイヤーを追加する前に、容量を適切にスケーリングする必要があります。このカスタマイズされたブルー/グリーン手法は、標準のブルー/グリーンよりもコストが比較的低いと予想される場合でも、Canary デプロイのローリング置換手法よりも高いコストが発生すると推定されます。

本番環境で 1 つの Canary のみを実行し、データとフィードバックに集中します。複数の Canary がデプロイされている場合、本番環境の問題のトラブルシューティングと分離が複雑になり、収集されるデータセットとフィードバックの品質が低下します。

Canary のバリエーションは、1 つ以上の実験 (通常は UI テスト) がターゲットを絞ったデプロイで実行され、ゲームバックエンドサーバーの 1 つのセットが機能の 1 つのバージョンを提供し、別の同じサイズのセットが同じ機能の別のバージョンを提供する場合です。これに対して追加のインフラストラクチャや特別なインフラストラクチャは作成されず、バックエンドサーバーの選択したポケットのみがこれらの更新を受け取ります。実験の結果は、プレイヤーが同じ機能の各バージョンにどのように反応するかを観察し、全体的な好き嫌いのコンセンサスがあるかどうかを判断し、そのユーザビリティや機能で特定された問題があるかどうかを観察することです。このような戦略的実験は A/B テストとも呼ばれ、全体的なプロセスは A/B テストと呼ばれます。これらの実験が完了すると、テストに使用されるサーバーのゲームバックエンドシステムの最新バージョンに戻す前に、必要なテストデータが収集されます。

従来のデプロイ

従来のデプロイスタイルでは、スケジュールされたメンテナンスウィンドウ中にゲームがシャットダウンされ、ゲームバックエンド内のサーバーインスタンスが最新のコードビルドで更新される前に、接続されたプレイヤーがドロップまたはドレインされます。このデプロイは、実行されるたびにプレイヤーに影響し、スケジュールの前にプレイヤーに通知する必要があります。その結果、このモデルはプレイヤーに最も大きな影響を与えるため、可能な限り避ける必要があります。

ゲームの更新がデプロイされると、ゲームを再開するのを待っているプレイヤーにゲームを開く前に、ゲームの煙をテストできます。これにより、プレイヤーが短時間でログインしてプレイしようとすると、トラフィックが急増する可能性があります。したがって、このようなトラフィックの急増を処理するようにゲームが設計されていない場合は、プレイヤーをバッチでゲームに戻すことを徐々に許可できます。

または、トラフィックのオープンスパイクを維持するためにインフラストラクチャを過剰にプロビジョニングすることを選択できます。ゲームトラフィックが解決したら、リソースをスケールダウンできます。必要に応じて、プレイヤー数が最低のオフピーク時間にこのタイプのデプロイを実行します。頻繁にスケジュールされたメンテナンスと拡張メンテナンスには、本質的にプレイヤーの減少や収益の損失のリスクがあります。また、プレイヤーは新しいリリース後に変更されることを想定しており、ダウンタイムが続くとゲームへの信頼が失われる可能性があります。

実装手順

- ダウンタイムを最小限に抑える: ダウンタイムを減らし、プレイヤーをゲームに参加させるデプロイ戦略を実装します。
- Infrastructure as Code (IaC): AWS CloudFormation や Terraform などのツールを使用してゲームインフラストラクチャを管理し、ヒューマンエラーを減らします。
- デプロイ戦略: ローリング置換、ブルー/グリーン、カナリアデプロイの 1 つまたは組み合わせを使用して、スムーズな更新を提供し、プレイヤーへの影響を軽減します。

GAMEOPS03-BP05 ピーク要件をサポートするために必要なプリスケールインフラストラクチャ

大規模なゲームイベントの前にインフラストラクチャをスケールして、プレイヤーの需要の急激な増加に対応できるようにします。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

新しいゲームのローンチに加えて、ライブゲームは通常、プレイヤーのエンゲージメントを維持および改善する方法の例として、ゲーム内イベント、プロモーション、新しいコンテンツ、およびシーズンリリースを実行します。このようなアクティビティでは、イベントまたはプロモーションの期間中、大量のプレイヤートラフィックが発生します。ビジネスは、イベントの意図した目標を達成または上回ることを期待しており、ゲームインフラストラクチャはそれを通じてそれらを維持およびサポートする必要があります。

大規模なイベント中に発生すると予想されるプレイヤーの負荷をサポートできるように、インフラストラクチャを事前に準備します。準備のために、ゲーム運用チームはセールスおよびマーケティングのステークホルダーと協力して、過去のプレイヤーの同時実行数、エンゲージメントメトリクス、販売データを調べて、今後のイベントで生成される予測需要を見積もる必要があります。イベントが新しいゲーム起動の場合、ゲーム運用チームはこれらの利害関係者と協力して、予想される規模に関する現実的な予測を特定する必要があります。ゲームの成功を予測することは難しいかもしれませんが、インフラストラクチャをスケーリングしてテストしてそれらの目標をサポートするために、成功への期待を全員が理解することが重要です。

多くのゲームは段階的に起動することを選択します。まず、少数のプレイヤーにゲームを開いてから、完全なパブリック起動の前に、各ステージでプレイヤーを有機的にスケーリングします。ソフト起動期間中は、パブリック起動の予測を改良しながら、問題をモニタリング、識別、追跡、解決します。

インフラストラクチャ要件を適切に見積もるには、ゲームの起動前に、本番稼働環境または本番稼働用のステージング環境で実行されているゲームバックエンドに対して実行される負荷テストとパフォーマンステストを通じてデータを収集します。これらのテストを複数回実行してゲームのさまざまな条件をシミュレートし、バックエンドがほとんどの条件下で負荷に耐えることができることを検証する必要があります。

これを実現するために、開発者はゲーム内のさまざまなワークフローを横断し、さまざまな条件をエミュレートするゲームプレイボットを記述できます。これらのテストでは、各レイヤーとコンポーネントがテストされ、詳細が記録されるように、ゲームバックエンドのさまざまなシステムレイヤーを検査する必要があります。これらのテストから収集されたデータを使用して、ゲーム起動のプランをプロビジョニングします。

単一障害点 (SPOF) は、アプリケーションの可用性と耐障害性を高めることで、可能な限り特定して削除する必要があります。負荷テストを使用して、さまざまなアップストリームレイヤーとダウンストリームレイヤーで障害をエミュレートし、ゲームやその他のコンポーネントの動作を検証することで、SPOFs を特定します。

ゲームの起動、ゲーム内イベント、またはプロモーションの準備に必要な推定インフラストラクチャとともに、オンデマンドで自動的にスケーリングするようにシステムを設定します。ゲームバックエンドが大量のプレイヤートラフィックを維持するためにスケーリングできるように、スケーリングイベントしきい値を定義、設定、モニタリングします。可変トラフィックの場合、スケールアウトするのに十分な時間がないため、事前プロビジョニングが最適です。手動スケーリングは、ゲームの初回起動時に必要になる場合があり、自動システムがリソースをスケールできるよりも速く予想以上の需要を引き出します。

では AWS、組織はゲームバックエンドで使用するサービスに対してより高い Service Quotas をリクエストする必要があります。Service Quotas は、意図せず立ち上がったり、意図したよりも多くのインフラストラクチャをスケーリングしたりするのを防ぐために、アカウント用に設定されています。アカウントで実行されているゲームが、そのリージョンで設定されたサービスクォータの上限に達すると、サービスはプロビジョニングされたクォータとバーストプロビジョニングを超えてリクエストを調整します。スロットリングにより、意図しないエラーや予期しないエラーが発生し、プレイヤーエクスペリエンスが損なわれる可能性があります。スロットリングを避けるために、本番稼働中のゲームで使用されるサービスのサービスクォータのしきい値を監視、追跡、および定期的に確認します。使用量が許容可能なサービスクォータのしきい値を超えると、コンソールサポートセンターからサポートケースを引き上げるか、影響を受けるアカウントにログインした後、またはサポート API を使用して、クォータの引き上げをリクエストできます。

ゲームのローンチ、コンテンツリリース、プロモーション、主要なゲーム内イベントなどの重要なイベントには、AWS Countdown を使用します。Countdown は、運用の準備状況を提供し、潜在的なリスクを軽減し、容量のニーズを計画するために Games の専門家によって構築されたプレイブックに基づく実装ガイダンスを提供します。AWS Countdown には、インフラストラクチャを最適化するためのサポートとエンジニアなどのオプションを強化するプレミアムサポートオプションもあります。

Amazon GameLift でホストされているゲームを起動する場合は、起動前のチェックリストを確認して準備します。

実装手順

- インフラストラクチャをスケールアップする: プレイヤーの需要の急激な増加に対応するため、大規模なゲームイベントに備えてインフラストラクチャを事前に準備します。
- 需要を見積もる: セールスやマーケティングと連携して、過去のプレイヤーデータとリアルな予測を使用して予測需要を見積もります。
- 負荷テストと SPOF 削除: 負荷テストを複数回実行して、バックエンド容量を検証し、単一障害点を特定し、自動スケーリングを適切に設定します。

ヘルスモニタリング

GAMEOPS04: ゲームの状態をどのようにモニタリングしますか?

包括的な計測を実装してゲームの状態をモニタリングし、クライアント側のアクティビティログ記録、バックエンドサービスのモニタリング、エラーレポートなど、プレイヤーに影響を与える問題を検出して追跡します。Amazon CloudWatch や AWS X-Ray、サードパーティーソリューションなどの AWS ツールを組み合わせて、問題をすばやく特定して解決できるようにします。

ベストプラクティス

- GAMESOPS04-BP01 ゲームを計測して、プレイヤーに影響を与える問題を検出してモニタリングする

GAMESOPS04-BP01 ゲームを計測して、プレイヤーに影響を与える問題を検出してモニタリングする

ソーシャルメディアやプレイヤーの問題レポートへの対応に加えて、プレイヤーに影響を与える問題を検出して調査するためのモニタリングソリューションを使用してゲームを計測します。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

ゲームのすべての問題を特定できるテストの量はありません。ゲームは通常、ゲームの次のリリースで徐々に修正される予定の既知の問題で起動されます。既知の再現可能な問題は、簡単に対処して修正できます。このような問題の特定を支援するために、ゲームクライアントは、バックエンドチームがクライアント側の問題を特定するのを支援するために、プレイヤーアクティビティの追跡、アプリのログ記録、レポートをさまざまな戦略的場所に実装する必要があります。このような問題を早期に見つけることができると、ゲーム開発者は問題が広範囲になる前にトラブルシューティングして修正できます。追跡コードによってレポートされるデータとログには、個人を特定できる情報 (PII) を含めることはできず、デバッグに役立つゲーム固有のメタデータのみを含める必要があります。

ゲームのクラッシュやバグなどの問題を検出して対応するためのオブザーバビリティソリューションを実装します。Amazon CloudWatch Synthetics を使用して、プレイヤー向けのバックエンドゲームサービスの状態をモニタリングできる Canary を作成できます。を使用してバックエンドサービ

スを計測AWS X-Rayして、分散サービス全体のリクエストを追跡し、カスタムログとメトリクスを Amazon CloudWatch に送信できます。

Backtrace.io や Sentry などのサードパーティーソリューションは、ゲームのエラーレポートによく使用されるソリューションです。New Relic、Splunk、Datadog、Honeycomb.io などのパートナーからのアプリケーションパフォーマンスモニタリング (APM) ソリューションも人気があります。

ゲームのライブオペレーションチームとコミュニティマネージャーは、さまざまなソーシャルネットワークとチャネルをモニタリングして、公式のサポートチャネルに加えて、プレイヤーのフィードバック、苦情、バグレポートを確認する必要があります。ゲーム固有の苦情をすべて確認して再現しようとするか、レビューのために QA チームに送信します。再現可能な場合は、より大きなプレイヤーベースに影響を与える前に、問題をゲーム開発者にエスカレーションしてトラブルシューティングと修正を行います。

実装手順

- モニタリングソリューションを実装する: モニタリングツールを使用してプレイヤーに影響を与える問題を検出し、迅速に対応します。
- プレイヤーのアクティビティとログを追跡する: ゲームクライアントを計測して、プレイヤーのアクティビティをログに記録して問題を報告し、個人を特定できる情報 (PII) が含まれていないことを確認します。
- サードパーティーおよび AWS ツールを使用する: CloudWatch、X-Ray、サードパーティーソリューションなどのツールを使用してエラーレポートやパフォーマンスモニタリングを行い、ソーシャルメディアをモニタリングしてプレイヤーのフィードバックやバグレポートを確認します。

負荷テスト

GAMEOPS05: ゲームをロードテストするときに考慮すべきことは何ですか?

ゲームを負荷テストするときは、システムのパフォーマンスとスケーラビリティを効果的に評価するために、適切なテストステージ、負荷生成アーキテクチャ、テストフレームワークを検討してください。タイミング (初期開発、スプリント、本番稼働前、またはデプロイ後) 、インフラストラクチャ (EC2、EKS、Fargate、または Lambda)、テストツール (JMeter、Locust、Grafana K6、または Gatling) の適切な組み合わせを選択して、ゲーム固有の特性と開発目標に合わせます。

ベストプラクティス

GAMEOPS05-BP01 目標を達成するための適切なステージ、アーキテクチャ、負荷テストフレームワークを選択する

ゲームの負荷テストのアプローチは、実行される開発プロセスの段階、負荷生成システム自体のアーキテクチャ、負荷テストフレームワークの選択など、多くの要因によって大きく異なる場合があります。初期フェーズ、反復スプリント中、本番デプロイ前、デプロイ後のいずれであっても、いつ実施されるかがテスト作業の目標と焦点を形成します。負荷生成インフラストラクチャの設計には独自の長所と短所があり、負荷テストフレームワークの選択は、テストプロセスで使用できる機能、使いやすさ、統合に大きな影響を与えます。これらの要素を慎重に調整することで、開発チームは負荷テストのアプローチをゲーム固有の特性に合わせて調整し、最も価値のあるパフォーマンスインサイトを引き出し、プレイヤーにスムーズなエクスペリエンスを提供できます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

さまざまな開発段階での負荷テスト

開発フェーズの早い段階で探索的負荷テストを実施することで、基盤となるシステムアーキテクチャを検証できます。これにより、デベロッパーは、広範な実装作業を行う前に、ゲームのインフラストラクチャ、データベース設計、ネットワークトポロジーについて十分な情報に基づいた意思決定を行うことができます。負荷テストはリスクを特定し、パフォーマンスベースラインを作成するため、開発ライフサイクルの後半でコストのかかる再作業や技術的負債を最小限に抑えることができます。また、チーム間でゲームのパフォーマンス要件の理解を共有し、コラボレーションと意思決定を向上させることもできます。最終的に、初期フェーズの負荷テストは、高性能でスケーラブル、回復力のあるゲームの強力な基盤を構築し、プレイヤーエクスペリエンス全体を向上させるのに役立ちます。

負荷テストでは、各スプリントまたはイテレーションの終了時に、最新のサイクルで導入された新機能、バグ修正、その他の変更のパフォーマンスへの影響を評価できます。このターゲットを絞ったアプローチにより、開発チームは最新の更新によってもたらされるリグレッションやパフォーマンスの低下をすばやく特定できるため、パイプラインをさらに進める前にこれらの問題に対処し、一貫したレベルの品質とパフォーマンスを維持できます。

堅牢な負荷テストは、本番環境にデプロイする前に、予想される実際のトラフィックと負荷条件を処理するシステムの能力を検証するのに役立ちます。本番インフラストラクチャ内のスケーラビリティ

のボトルネックやリソースの制約を発見し、ゲームのパフォーマンスを最適化する機会を提供し、初日からスムーズで応答性の高いユーザーエクスペリエンスを生み出すことができます。起動前の負荷テストから得られたインサイトは、起動日のリスクを軽減し、継続的なキャパシティプランニングに役立つため、ゲームの長期的な持続可能性とスケーラビリティの基盤となります。

すでに本番稼働しているゲームの負荷テストにより、チームはゲームのパフォーマンスをモニタリングし、時間の経過とともに発生する可能性のあるパフォーマンスの低下を特定できます。これにより、プレイヤーエクスペリエンスに影響を与え、ユーザーの保持に悪影響を及ぼす前に、プロアクティブに問題に対処できます。さらに、本番環境での負荷テストでは、実装されたパフォーマンス最適化作業またはインフラストラクチャスケーリングの有効性を検証します。このプロセスは、ゲームが進化して成熟しても、プレイヤーに高品質で応答性の高いスケーラブルなゲームエクスペリエンスを提供します。

ロード生成アーキテクチャ

ゲーム負荷テスト用の負荷生成アーキテクチャの設計にはさまざまな形式があり、それぞれに独自の利点と考慮事項があります。

最も基本的なレベルでは、セルフマネージド Amazon EC2 インスタンスをプロビジョニングして、ロードジェネレーターとして機能するように設定できます。コントロールノードとワーカーノードのアプローチでは、複数のロード生成インスタンスをセットアップできます。各インスタンスは独自のテストスクリプトを実行し、全体で 1 つのコントロールインスタンスによって管理されます。アーキテクチャは、追加のワーカーノードをスピンアップすることで複雑さを増すことなくスケールアップしてより多くの負荷を生成できますが、この実践的なアプローチでは、チームは基盤となるインフラストラクチャのプロビジョニング、設定、管理を処理する必要があります。

よりスケーラブルでオーケストレーションされたアプローチでは、Amazon EKS Kubernetes クラスターを使用して、コンテナベースのロードエージェントのフリートに負荷テストワークロードを管理および分散できます。Kubernetes 自動スケーリング機能を使用して、負荷生成ポッドのスケーリングを処理できます。一方、チーム自身がポッドをホストするクラスター内の基盤となる EC2 インスタンスを設定および管理します。

または、 のサーバーレスな性質により、必要なスケーラビリティと柔軟性を維持しながらインフラストラクチャ管理を抽象化することで、負荷テストのセットアップを高速化および簡素化AWS Fargateできます。オンプレミスの負荷生成 Kubernetes クラスターが既に存在するが、追加の容量が必要なハイブリッドソリューションの場合、EKS Anywhere は両方のクラスターを から 1 つとして管理できます AWS マネジメントコンソール。

また、要件と目標に応じて AWS Lambda関数を使用することもできます。Lambda 関数は、追加のリソースをプロビジョニングして管理することなく、比較的簡単にセットアップしてスケーリング

できます。また、他の AWS サービスとの深い統合により、より複雑で動的なテストシナリオを作成することもできます。ただし、Lambda 関数には同時関数とランタイム (15 分) に制限があるため、達成できる負荷テストのスケールと長さが制限される可能性があります。コールドスタートレイテンシーは結果の精度にも影響し、Lambda のリソース制限は要求の厳しい負荷テストワークロードには適していない可能性があります。

構築済みのソリューションの使用を希望するスタジオは、分散負荷テスト AWSを使用できます。このソリューションでは、 上の Amazon ECS AWS Fargate を使用して、数万人の接続ユーザーのシミュレーションを実行できるコンテナをデプロイします。これを使用すると、 を使用して IAC 方式で負荷テストインフラストラクチャをすばやく開始できます AWS CloudFormation。

負荷テストフレームワーク

2 つの負荷テストフレームワークが同じように構築されることはありません。テスト作成用の直感的なグラフィカルインターフェイスを持つものもあれば、コマンドラインベースのものもあります。1 つのツールは柔軟でパフォーマンスが高いものの、設定と管理に時間と労力が必要であり、もう 1 つのツールはサーバーレスですが、作成して実行できるテストには制限があります。大規模なコミュニティと多くのチュートリアルを好む人もいれば、現場では証明されていない人もいます。本番環境でバトルテストされているが、コミュニティのサポートやドキュメントがない人もいます。自分とチームに適したバランスを取るフレームワークを選択します。いくつかの一般的なオプションは次のとおりです。

- Apache JMeter: 堅牢な機能セットと使いやすさにより、人気のある Java ベースのオープンソースの負荷テストフレームワーク。複雑なユーザーシナリオ、サポートされているさまざまなプロトコル、包括的なレポート、実績をシミュレートする機能により、JMeter は負荷テストのための信頼性の高い選択肢となります。
- Locust: イベント駆動型アーキテクチャ上に構築された最新の分散負荷テストフレームワークは、リソース効率に優れたパフォーマンスを実現します。テストは Python で記述されるため、何千もの強力なサードパーティーライブラリを活用しながら、読みやすくて読みやすいまま、柔軟なテストシナリオが可能になります。
- Grafana K6: 使いやすさと高度な機能を組み合わせた強力な負荷テストフレームワーク。分散負荷生成、柔軟なスクリプティング、データの視覚化のための Grafana とのシームレスな統合をサポートする Grafana K6 は魅力的な選択肢です。
- ガトリング: パフォーマンスとスケーラビリティで知られるオープンソースの負荷テストフレームワーク。Scala ベースのドメイン固有の言語 (DSL) により、開発者は簡潔で保守可能な負荷テストスクリプトを作成できます。また、堅牢なレポートおよび分析機能により、テスト対象のシステムの詳細なインサイトが得られます。

実装手順

- 負荷テスト段階: さまざまな開発段階 (初期開発、スプリント、本番稼働前、デプロイ後) で負荷テストを実施して、システムのパフォーマンスを検証し、問題を特定します。
- 負荷生成アーキテクチャ: スケーラビリティのニーズ、管理設定、および特定のテスト要件に基づいて、適切な負荷生成アーキテクチャ (EC2、EKS、Fargate、または Lambda) を選択します。
- 負荷テストフレームワーク: チームのニーズに応じて、使いやすさ、パフォーマンス、柔軟性、コミュニティサポートのバランスがとれた負荷テストフレームワーク (JMeter、Locust、Grafana K6、Gatling など) を選択します。

継続的最適化

GAMEOPS06: 時間の経過とともにゲームを最適化する方法を教えてください。

主要なメトリクスとテレメトリデータをモニタリングして、プレイヤーの傾向、システムパフォーマンス、改善すべき分野を特定することで、ゲームを経時的に最適化します。これらのインサイトに基づいてゲーム設計、インフラストラクチャ、負荷テストのアプローチを継続的に更新し、新しいテクノロジーやフレームワークに適応して、ゲームの進化に合わせて最適なパフォーマンスとプレイヤーエクスペリエンスを提供します。

ベストプラクティス

- GAMEOPS06-BP01 主要なゲームメトリクスをモニタリングしてプレイヤーの傾向とパターンを特定し、その情報を使用してゲームを改善する
- GAMEOPS06-BP02 ゲームの変更に応じて負荷テストアプローチを更新して適応させる

GAMEOPS06-BP01 主要なゲームメトリクスをモニタリングしてプレイヤーの傾向とパターンを特定し、その情報を使用してゲームを改善する

ゲームクライアントシステムの使用状況、アプリの使用状況、例外、クラッシュデータに加えて、ゲームバックエンドシステムに送信されるゲームテレメトリデータをキャプチャします。このデータは、プレイヤーがゲームのさまざまな機能とやり取りする方法を理解できるように、プレイヤーのアクティビティを表す必要があります。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

ゲームクライアントは、その実装に応じて、ゲーム世界の事前定義されたゲーム機能または場所でテレメトリデータを収集できます。データはバックエンド取り込みサービスに送信され、処理されます。バックエンドサービスにアクセスできない場合、クライアントはバックエンドサービスが再び利用可能になるまでローカルデバイスにデータをローカルに保存できます。ゲームデザイナーはこのテレメトリデータを使用して、プレイヤーがゲームをどのようにプレイしているか、ゲームに異常があるかどうかを確認します。

たとえば、プレイヤーの動きやマップ内のアイテムとのやり取りをテレメトリデータから抽出し、プレイヤーが設定した期間にゲーム内のアクティビティのヒートマップとしてプロットできます。このようなデータは、ゲームデザイナーが武器の力、ゲーム内のキャラクターの力、マップの複雑さなど、ゲーム内のさまざまな要素のバランスを取る必要性を特定するのに役立ちます。生のテレメトリデータは一般的に保存され、アナリストが視覚化できる分析を抽出するために処理されます。

Game AnalyticsPipeline ソリューションの実装は、ゲーム開発者がスケーラブルなサーバーレスデータパイプラインを起動して、ゲームやサービスから生成されたテレメトリデータを取り込み、保存、分析するのに役立ちます。このソリューションは、データのストリーミング取り込みをサポートしているため、ユーザーは数分以内にゲームやその他のアプリケーションからインサイトを得ることができます。

カスタムゲームテレメトリデータの取り込み、ストレージ、処理、分析のために、 はビッグデータ処理と分析のための多くの特殊なサービス AWS も提供しています。

実装手順

- ゲームテレメトリデータをキャプチャする: プレイヤーのアクティビティ、システム使用状況、例外、クラッシュに関するデータを収集して、プレイヤーのインタラクションを理解し、問題を特定します。
- テレメトリコレクションを実装する: 事前定義されたゲーム機能または場所を使用してテレメトリデータを収集し、バックエンドサービスに送信し、バックエンドに到達できない場合はローカルに保存します。
- AWS 分析ソリューションを使用する: Game Analytics Pipeline などの AWS サービスを使用して、スケーラブルなデータの取り込み、ストレージ、分析、および特殊なビッグデータ処理および分析サービスを行います。

GAMEOPS06-BP02 ゲームの変更に応じて負荷テストアプローチを更新して適応させる

負荷テストアプローチの最適化は、ゲーム開発サイクルに合わせて進化すべき継続的なプロセスです。ゲームの複雑さ、ユーザーベース、機能セットが増大するにつれて、ロードテスト戦略は実際の条件を正確にシミュレートし、実用的なインサイトを提供することを確認するように適応する必要があります。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

以下の点を考慮してください。

欠落または古いテストシナリオ

開発プロセス中にゲームに新機能が追加されたら、新しい負荷テストシナリオを作成して実行し、新機能のパフォーマンスとスケーラビリティを検証します。同様に、パフォーマンスを向上させたり、プレイヤーのフィードバックに対処したり、新しい設計目標に合わせるために機能や機能がリファクタリングされることが多く、変更に合わせてテストシナリオを継続的に更新し、システムの状態を真にテストして反映する必要があります。

新しい負荷テストフレームワーク

開発者は、さまざまな理由で負荷テストフレームワークを変更する必要がある場合があります。

- 初期フレームワークでは、ユーザーの負荷を適切にシミュレートできなくなったり、システムのパフォーマンスに必要なレベルのインサイトを提供できなくなったりする可能性があります。
- 新しいゲーム機能では、新しいプロトコル、APIs、または統合ポイントの負荷テストのサポートが必要になる場合があります。
- 開発者は、負荷テストプロセスに慣れるにつれて、より高度な機能が必要になる場合があります。
- チームの技術的専門知識、プログラミング言語、または既存のツールチェーンにより適したフレームワークを優先する

開発者は、時間の経過とともに慎重に評価して適応することで、ロードテストプロセスをゲームの要件の変化に合わせて調整し、ユーザーエクスペリエンス全体を最適化して改善するために必要なインサイトを継続的に提供できます。

コストの最適化

マネージド AWS サービスを使用する容易さと利便性は、特に開発の初期段階で非常に有益です。これらのサービスは基盤となるインフラストラクチャ管理を抽象化するため、チームはソリューションを迅速にセットアップし、負荷テストシナリオの作成と結果の分析のみに集中できます。ただし、マネージドサービスを使用すると、インフラストラクチャのプロビジョニング、設定、保守、高可用性、スケーリング、モニタリング機能の提供など、付加的な価値と利便性により、コストが高くなることがよくあります。

チームが成熟し、負荷テストプロセスに慣れ、自信が増すにつれて、インフラストラクチャを自己管理することで、追加の最適化とコスト削減を実現できる時期が来るかもしれません。この実践的なアプローチでは運用オーバーヘッドが増加しますが、コンピューティングリソース、設定、スケーリング動作、リソース使用率を直接制御することで、微調整とコスト削減のための新しい機会が生まれます。たとえば、チームが AWS Fargate サーバーレスアーキテクチャで負荷テストジャーニーを開始し、後で Amazon EKS クラスター内の基盤となるノードを自己管理することに移行するのが理にかなっている場合があります。

実装手順

- テストシナリオの更新: ロードテストシナリオを継続的に作成および更新して、新機能とリファクタリングされた機能を検証し、ゲームの現在の状態を反映していることを確認します。
- 負荷テストフレームワークを評価する: 必要に応じて新しいフレームワークに適応し、ユーザーの負荷をシミュレートし、新しいプロトコルをサポートし、チームの専門知識とツールチェーンに合わせます。
- コストの最適化: 簡単で便利なマネージド AWS サービスから始め、チームが負荷テストプロセスに慣れるにつれて、コスト削減のための自己管理インフラストラクチャを検討してください。

リソース

オペレーショナルエクセレンスに関連するベストプラクティスの詳細については、以下のリソースを参照してください。

ドキュメントとブログ

- Game Tech のアーキテクチャのベストプラクティス
- pt.1 での Game Studio AWS の管理
- pt での Game Studio AWS の管理 2
- pt での Game Studio AWS の管理 3

- ベストプラクティス AWS 環境の確立
- Control Tower ランディングゾーンのマルチアカウント戦略
- ゲーム分析パイプライン
- Game Analytics Pipeline でゲームデータインサイトを最大化する
- Player Insights 用の AWS Glue と Amazon Redshift Spectrum の活用
- で CI/CD パイプラインをセットアップする方法
- AWS ビルドパイプラインで良いジョブゲームが 43% 高速化する方法
- Unity モバイルアプリ用のビルドパイプラインの実装
- その他の関連する CI/CD ブログ
- Game-Server CD Pipeline ブログで Game DevOps を簡単に
- Harmony Games が完全にカスタムのゲームバックエンドをデプロイする (AWS Cloud Development Kit (AWS CDK)AWS CDK)
- GameLift 起動の準備
- Amazon Gamelift Anywhere によるハイブリッドゲームサーバーホスティング
- Amazon Gamelift Anywhere と Amazon Gamelift エージェントを使用してゲームサーバー開発を高速化する
- Amazon Gamelift を使用して、プレイヤーごとに 1 USD 未満の Unreal Engine ゲームをホストする方法
- でスケーラブルなクロスプラットフォームゲームバックエンドを構築するための新しいソリューションガイダンス AWS
- で 100 万人の同時ユーザーに Pragma バックエンドゲームエンジンをロードテストする AWS
- Code Wizards が で 200 万人の同時プレイヤーに勲章をテストする方法 AWS
- 組織単位のベストプラクティス
- AWS X-Ray
- AWS カウントダウン
- AWS for Games Solutions Hub

パートナーソリューション

- New Relic
- Splunk APM
- Backtrace.io

- セントリー
- Datadog APM
- Honeycomb.io

ホワイトペーパー

- 複数のアカウントを使用した環境の整理
- でのスケーラブルなゲーム開発パターンの概要 AWS

動画

- YouTube シリーズ: でゲームを構築する AWS
- AWS ゲーム用: Boss LEVEL ポッドキャスト
- Re:Invent 2023: 最初の 1,000 万人 AWS のユーザーのスケーリング
- Re:Invent 2022: Riot Games が で毎日 20TB の分析を処理する方法 AWS
- Re:Invent 2022: AWS と Riot Games がガバナンスレポートエンジンを構築した方法
- Re:Invent 2023: での分散設計パターンの実装 AWS
- Re:Invent 2023: Mortal Kombat 1 を使用してマルチプレイヤーゲームを数百万にスケールする
- Re:Invent 2022: Netflix でのカオスエンジニアリングの進化
- Re:Invent 2023: クラウドガバナンスのベストプラクティス
- Re:Invent 2023: でマルチリージョンアーキテクチャを作成するためのベストプラクティス AWS

トレーニング資料

- カリキュラム – ゲームパート 1 AWS の の開始方法

セキュリティ

セキュリティの柱には、リスク評価と緩和を通じてビジネス価値を提供しながら、情報、システム、アセットを保護する能力が含まれます。グローバルな可視性と多数のプレイヤーのために、ゲームはエクスプロイター、ハッカー、およびシステムを悪用して悪用する方法を探している他の人々にとって望ましいターゲットです。これにより、強力なセキュリティ基盤が設定されていない場合、プレイヤーエクスペリエンスが失望し、ゲーム開発者のコストが増加することがよくあります。

責任共有モデルで説明されているように、セキュリティのどの側面がお客様の責任 AWS であり、どの側面がお客様の責任であるかを理解し、強力なセキュリティ体制を維持する準備を整えることが重要です。この柱は、クラウドでゲームを開発および運用する際に考慮すべきベストプラクティスのクラウドセキュリティガイダンスを提供します。

システムを設計する前に、アクセスコントロールを含む一連のセキュリティのベストプラクティスを確立する必要があります。さらに、データ保護を通じてデータの機密性と完全性を維持しながら、セキュリティインシデントを特定し、システムやサービスを保護できる必要があります。セキュリティインシデントに対応するための、明確に定義された経験豊富なプロセスを利用できます。これらのツールと手法は、財務損失の防止や規制上の義務の遵守などのビジネス目標をサポートするため、重要です。

お客様事例

AnyCompany Games は、セキュリティ体制を改善している架空のゲームスタジオです。セキュリティは、直接アプリケーションの説明がある場合に簡単に理解できます。AnyCompany Games は、このセクションで、柱で説明されているセキュリティのベストプラクティスをコンテキスト化するために使用します。

焦点

- 設計原則
- セキュリティ基盤
- 継続的なセキュリティ
- ID とアクセス管理
- アクセスコントロール
- 検出
- インフラストラクチャの保護
- インシデントへの対応

- アプリケーションのセキュリティ
- セキュリティの自動化
- 脅威モデリング
- リソース

設計原則

Well-Architected Framework ホワイトペーパーのセキュリティの柱の設計原則に加えて、次の設計原則はクラウド内のゲームワークロードのセキュリティを強化できます。

- プレイヤーの使用状況のモニタリングとモデレート: 使用状況データをキャプチャして分析し、プレイヤーがゲームやソーシャル機能とやり取りする方法を理解します。このデータを分析することで、プレイヤーのエクスペリエンスを低下させる可能性のある虐待的および不適切な動作を検出して対応できます。

セキュリティ基盤

GAMESEC01: ゲーム開発のセキュリティの基礎をどのように実装しますか?

ゲームスタジオには、開発環境とライブプレイヤーサービスの両方を保護する独自のセキュリティアプローチが必要です。ゲームスタジオの堅牢な AWS セキュリティ戦略には、マルチアカウント構造、強力な認証、IAM ポリシーを使用した明確な認可戦略の 3 つの相互接続されたコンポーネントが必要です。マルチアカウント AWS 構造により、スタジオはさまざまなゲームプロジェクト、開発ステージ、ツール環境を分離できます。これにより、スタジオは特定の環境やサービスへのアクセスなどをより詳細に制御できます。強力な認証を有効にすると、チームメンバーは、ソースコード、ゲームビルド、独自のツールを厳密に制御しながら、スタジオ内またはリモートで作業する場合でも、開発リソースに安全にアクセスできます。また、Studio には、IAM アクセス許可とロールを持つ最小特権の原則を使用してアクセス許可を付与するための明確な認可戦略が必要です。IAM ロールを使用して、開発チームに低レベルの AWS サービスへのアクセスを許可し、アーティストやデザイナーを特定のアセット管理やビルドシステムに制限するなど、さまざまな開発チームロール間にアクセス許可を割り当てます。この特殊なアプローチにより、ゲームスタジオが知的財産を保護し、効率的な開発ワークフローを維持し、チームを安全にスケーリングできると同時に、デベロッパーがプロジェクトを迅速に反復するための適切なアクセス権を付与できることを検証します。

ベストプラクティス

- GAMESEC01-BP01 アカウントのルートユーザーではなくロールとフェデレーティッドアクセスを使用して、 AWS 環境でアクションを実行する
- GAMESEC01-BP02 AWS Control Tower を使用して でマルチアカウント環境をすばやくセットアップする AWS
- GAMESEC01-BP03 特定の職務機能に合わせた最小特権ロールポリシーを使用する
- GAMESEC01-BP04 ロールとフェデレーティッドアクセスポリシーをアカウントレベルのアクセスポリシーと一緒に使用して、 AWS リソースへのアクセスを許可する
- GAMESEC01-BP05 中央 ID プロバイダーを使用する

GAMESEC01-BP01 アカウントのルートユーザーではなくロールとフェデレーティッドアクセスを使用して、 AWS 環境でアクションを実行する

を初めて作成するときは AWS アカウント、ルートユーザーと呼ばれる ID から始めます。これは、アカウントに関連付けられた E メールアドレスとパスワードを使用してアクセスされます。ルートユーザーは、そのアカウント内の AWS サービスとリソースへの完全なアクセス権を持ちます。ほとんどの場合、day-to-dayタスクにルートユーザーを使用しないでください。ルートレベルのアクセスが必要な場合は、それが絶対に必要であることを確認し、その使用を追跡するために追加の口グ記録とガードレールが設定されていることを確認します。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

AWS Organizations 設定では、各アカウントにはまだ独自のルートユーザーがありますが、day-to-dayアクセスを管理する必要があります。ゲームのライフサイクルステージとチームに合わせたロールベースのアクセスを作成します。たとえば、ライブオペレーションチームにはゲーム内イベントを管理するためのアクセス許可が必要で、デベロッパーには更新をプッシュするためのアクセス許可が必要になる場合があります。サードパーティーのサービスやパートナーと連携する場合は、フェデレーティッドアクセスを使用して、機密性の高いインフラストラクチャを公開することなく、安全なコラボレーションを可能にします。このアプローチでは、各ユーザーまたはパートナーが、ゲームのインフラストラクチャとプレイヤーデータのセキュリティを維持しながら、必要なアクセスのみを持っていることを確認します。

お客様事例

AnyCompany Games は、新しいゲームの開発時にロールベースのアクセスコントロールを実装しました。多様な開発チームに特定の IAM ロールを使用することで、共有認証情報の使用を回避できます。この設定により、開発チームはコアゲームシステムのロールを引き受けることができますが、コンテンツチームのロールはアセット管理サービスにのみアクセスできます。

実装手順

- 絶対に必要な場合を除き、アカウントのセットアップ後にルートユーザーを使用しないでください。アカウントを作成し、ルートユーザーを保護し、すぐに必要な管理 IAM ロールを作成し、そのロールをフェデレーティッドユーザーに割り当てます。
- ルートユーザーのみが使用できる限られた数のタスクを実行する必要がある場合にのみ、ルートユーザーを使用します。これらのタスクの例としては、ルートユーザーの E メールアドレスの変更や AWS サポートプランの変更などがあります。

GAMESEC01-BP02 AWS Control Tower を使用して でマルチアカウント環境をすばやくセットアップする AWS

単一のアカウント AWS で の使用を開始すると、ゲーム開発プロセスが進むにつれて、ゲームスタジオが成長する可能性があります。たとえば、単一の を使用すると AWS アカウント、サービスの制限に達し始めたり、さまざまなプロジェクトやワークロードのコストがより複雑になったりする可能性があります。ゲームタイトルや環境ごとに異なるアカウントを作成すると、チームは新機能を試し、サービスの制限を回避し、セキュリティ体制とコンプライアンスを維持できます。マルチアカウント戦略を に実装することで AWS、複数のアカウントにサービス制限を分散し、 AWS コストに関するインサイトを得ることができます。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

複数の を使用すると AWS アカウント 、自動的に混乱し、時間がかかるという一般的な誤解があります。むしろ、複数のアカウントのガバナンスを容易にするように設計された AWS サービスを使用すると、ゲームスタジオがアカウントの管理に費やす時間を短縮できます。

は、マルチアカウント AWS 環境を安全にプロビジョニングするためのサービス AWS Control Tower です。Control Tower は、新しい AWS 環境を構築する場合、ジャーニーを開始する場合 AWS、またはまったく初めて使用する場合に推奨されます AWS。短いセットアッププロセスでは、Service Catalog や AWS IAM Identity Center など AWS Organizations、アカウントの管理やユーザーアクセス AWS に関連する他の サービスと統合できます。

お客様事例

AnyCompany Games は当初 1 つの から運用され AWS アカウント、重要なベータテスト中にゲームの開発チームの 1 つが EC2 サービスの制限に達したときに複数の障害に遭遇しました。同時に、別のゲームの開発チームは、自動テストパイプラインのリソース割り当てに苦労しました。この状況は、AnyCompany Games がプロジェクト間でコストを正確に分離できず、各ゲームの開発に予算を計上することが困難になるという重大なポイントに達しました。

次に AnyCompany Games は、 を使用してマルチアカウント戦略を実装しました AWS Control Tower。開発環境、QA 環境、本番環境を区別して、ゲームプロジェクトごとに個別のアカウントを作成しました。このアカウントレベルの分離により、各プロジェクトのデータとアセットが分離されるため、あるゲームに取り組むチームは別のゲームからリソースにアクセスしたり変更したりすることはできません。を通じて AWS Organizations、各ゲームのインフラストラクチャコストを明確に示す一元的な請求構造を確立し、組織全体のアクセスポリシーを作成しました。

実装手順

- AWS Control Tower を使用して、自動マルチアカウント環境を設定します。
- 環境 (開発、QA、本番稼働など) に基づいてアカウントを整理します。
- IAM Identity Center AWS と Service Catalog を使用して、ユーザーのアクセス許可を一元化し、アカウント間のリソースプロビジョニングを合理化します。

GAMESEC01-BP03 特定の職務機能に合わせた最小特権ロールポリシーを使用する

IAM ポリシーの設定は、強力なセキュリティ基盤を確立するために不可欠です。IAM ポリシーでアクセス許可を設定するときは、タスクの実行に必要なアクセス許可のみを付与します。これを行うには、特定の条件下で特定のリソースに対して実行できるアクションを定義します。これは、最小特権アクセス許可とも呼ばれています。例えば、QA チームはテスト環境でモノを変更するためのアクセスが必要ですが、本番環境を変更することはできません。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

ワークロードやユースケースに必要なアクセス許可を調べながら、 マネージドポリシーなどの広範なアクセス許可から始めることができます。ユースケースが成熟してきたら、最小特権になるように付与する権限を減らしていくことができます。

実装手順

- ユーザーとアプリケーションの IAM ロールを作成するための最小特権のアクセス許可の実践に従います。
- AWSマネージドポリシーを使用して、チームまたはアプリケーションがタスクを実行するために必要な特定のアクセス許可を特定しながら、広範なアクセスをすばやく提供します。
- Studio は、IAM アクセスアナライザーポリシーの生成を使用して、IAM エントリで使用されるアクションとサービスを識別する CloudTrail イベントに基づいてカスタム IAM ポリシーを生成することもできます。
- IAM ポリシーを定期的に確認し、過度に寛容なポリシーを編集します。

GAMESEC01-BP04 ロールとフェデレーティッドアクセスポリシーをアカウントレベルのアクセスポリシーと一緒に使用して、 AWS リソースへのアクセスを許可する

新しい AWS ユーザーは、他のユーザーにアクセスを許可する場合にのみ IAM ポリシーを使用することが多いです。ただし、 を使用している場合は AWS Organizations、サービスコントロールポリシーと IAM ポリシーを使用して、スタジオチームメンバーと請負業者に必要なレベルのアクセスを許可する方法を検討してください。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

IAM ポリシーを作成して、 が動作する サービスまたは API アクションへのアクセス AWS を許可または拒否できます AWS Identity and Access Management。ユーザー、グループ、ロールなどの IAM ID にのみ適用できます。例えば、IAM ポリシーを使用して、ユーザーに Amazon S3 への読み取り専用アクセスを提供することができます。

サービスコントロールポリシー (SCPs) は のガードレールです AWS アカウント。SCP はアクセス許可を付与せず、個々のメンバーアカウントの AWS サービスに対するアクションを制限するために使用されます。たとえば、SCP は が特定のリージョンにアクセス AWS アカウント することを拒否できます。

アクションを実行すると、関連する IAM ポリシーが SCPs。前の例に続いて、 は EC2 インスタンスの実行を試行するロールであり、IAM はそれらが許可されていることを示し (ec2:RunInstances に対

して「許可」）、SCPs「us-east-1」は許可されますが、「us-west-1」は SCP によって拒否されます）。

IAM ポリシーと SCPsレイヤーすることで、 AWS リソースにアクセスするすべてのユーザーに、必要な適切なアクセス許可のみが付与されることを確認できます。これは、 AWS アカウント リソースと リソースが複数のリージョンにまたがっているかどうかを考慮することが特に重要ですが、ゲームスタジオ内のすべてのユーザーがそれらすべてにアクセスする必要があるわけではありません。

IAM ポリシーを調整して、ゲーム設定の更新、プレイヤーデータの管理、プロモーションイベントの設定、ユーザー生成コンテンツのモデレーションなど、特定のチームに特定のアクセス許可を付与できます。一方、SCPs、ゲームオペレーションに不可欠な組織全体のコントロールを適用できます。これには、ゲームが動作する承認済みリージョンのみへのデプロイの制限、機密プレイヤーデータストアへの不正アクセスの防止、コンプライアンス要件の適用、開発アカウント間のサービス使用量の制限によるコストの制御などが含まれます。

実装手順

- IAM ポリシーを使用して、個々のユーザー、グループ、またはロールのアクセス許可を管理します。
- でサービスコントロールポリシー (SCPs) を使用して AWS Organizations 、アカウントレベルのアクセス許可を適用します。
- IAM ポリシーと SCPs を組み合わせて、特定のユーザーとアカウントに必要なアクセスのみを付与します。

リソース

- IAM AWS のポリシーとアクセス許可
- サービスコントロールポリシー
- ジョブ機能のAWS 管理ポリシー

GAMESEC01-BP05 中央 ID プロバイダーを使用する

中央 ID プロバイダーは、ユーザーの認証情報、ID、アクセス許可、認証を保存および管理するための単一のソースとして機能します。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

中央 ID プロバイダーを使用して、ユーザー認証プロセスを合理化し、一貫したセキュリティポリシーを適用し、 AWS アカウント および アプリケーション全体でユーザー管理を簡素化します。一元化されたアプローチにより、ユーザー ID と認証情報を個別に管理する必要がなくなるため、不整合、重複、その他のセキュリティ脆弱性のリスクが軽減されます。ユーザー ID と認証を 1 か所に統合すると、 AWS 環境全体の可視性、制御、監査性が向上します。

お客様事例

AnyCompany Games は、急速に拡大 AWS するインフラストラクチャ全体で開発者のアクセスを管理する上で大きな課題に直面しました。開発チームは、3 つの主要なタイトルで 50 人から 200 人に成長しました。当初、各プロジェクトチームは独自の AWS アクセス認証情報を管理していたため、セキュリティプラクティスに一貫性がなく、新しい開発者のオンボーディングが遅れ、セキュリティインシデントが時折発生していました。

スタジオは、IAM Identity Center AWS を中央 ID プロバイダーとして実装し、ユーザー管理を 1 つのシステムに統合しました。既存の社内ディレクトリに接続し、開発者が同じ会社の認証情報 AWS を使用してアクセスできるようにします。開発者は、単一の既存の企業ログインを使用して、作業を完了するために必要な AWS アクセス権を取得できるようになりました。

実装手順

- IAM Identity Center AWS を中央 ID プロバイダーとして使用することを検討してください。これにより、 全体で一貫したアクセス管理が提供され AWS アカウント、従業員にシングルサインオン認証が提供され、 AWS アプリケーションへのユーザーアクセス監査が簡素化されます。IAM Identity Center は、サポートされている ID プロバイダーの既存の企業 ID とも接続します。

継続的なセキュリティ

GAMESEC02: 継続的なセキュリティのベストプラクティスをどのように達成、維持、モニタリングしますか?

セキュリティのベストプラクティスに従うことは、さまざまな業界、特にゲーム業界にとって最優先事項です。ゲーム業界は、プレイヤーの信頼を育み、維持し、高い評価を得ることに依存しています。また、軽微なセキュリティ問題でも、その信頼がすぐに損なわれる可能性があります。

さらに、ゲーム業界のグローバルな性質上、ゲームが提供されているリージョン全体でのデータ保護、コンシューマープライバシー、セキュリティに適用されるさまざまな業界の規制や基準への準拠が必要です。公平で安全なゲームプレイは、堅牢なセキュリティ対策の重要性を強調するもう 1 つの重要な側面です。不正行為、ハッキング、その他の形式のゲーム搾取は、正当なプレイヤーのゲームエクスペリエンスを中断する可能性があります。これにより、強力なセキュリティコントロールがゲームプレイの整合性を維持し、参加者のレベルのプレイフィールドを育成するために不可欠です。

ベストプラクティス

- GAMESEC02-BP01 標準セキュリティプラクティス用のテンプレートをデプロイする準備が整った を使用する
- GAMESEC02-BP02 セキュリティイベントが発生したときに自動修復手法を使用する

GAMESEC02-BP01 標準セキュリティプラクティス用のテンプレートをデプロイする準備が整った を使用する

Ready-to-deployテンプレートを使用すると、クラウド内のセキュリティ体制をプロアクティブかつアジャイルに評価できます。事前設定されたテンプレートは、クラウドセキュリティを評価し、必要な変更を迅速に実装します。テンプレートには、さまざまなテクノロジーと広く受け入れられているセキュリティフレームワークにわたるさまざまなベストプラクティスが含まれています。テンプレートを使用すると、特に新しいワークロード AWS アカウント をサポートするためにスケーリングして追加する可能性があるため、ゲームスタジオが一貫したインフラストラクチャ設定を維持するのに役立ちます。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

AWS サービスを使用し、ready-to-deployテンプレートを実装することで、ゲーム開発者はクラウドセキュリティ体制を積極的に評価して強化し、知的財産を保護し、プレイヤーデータを保護し、定期的なセキュリティ評価と継続的なモニタリングを通じて安全なゲーム環境を促進し、潜在的な脆弱性を迅速に特定して対処できます。

お客様事例

AnyCompany Games は、欧州業界で次のタイトルをリリースする準備をする際に、大きな課題に直面しました。既存のデータ処理プラクティスが GDPR 要件を満たしていないことに気付きました。ソリューションでは、 AWS Security Hub CSPM AWS Config と 、およびready-to-deployテンプレートを使用しました。チームは GDPR 固有のコンフォーマンスパックを に実装し AWS

Config、GDPR 標準に照らして既存のインフラストラクチャを自動的に評価しました。この最初のスキャンでは、不適切なデータ保持ポリシーや、プレイヤーデータが保存された場所に対する不適切なアクセスコントロールなど、いくつかの重要なギャップが明らかになりました。テンプレートの事前定義されたルールを使用して、AnyCompany Games は必要な変更を迅速に実装しました。さらに、テンプレートによって提供される継続的な自動コンプライアンスチェックにより、小規模なチームはゲームの更新と拡張を継続しても、GDPR コンプライアンスを簡単に維持できるようになりました。

実装手順

- のマネージドルールやコンフォーマンスパック、 の AWS Config 標準など、標準のセキュリティプラクティスには テンプレートを使用します AWS Security Hub CSPM。
- Security Hub CSPM 標準の詳細を確認して、どの標準がゲームスタジオのセキュリティニーズに最も合っているかを判断します。

GAMESEC02-BP02 セキュリティイベントが発生したときに自動修復手法を使用する

ゲーム開発者は、自動修復手法を使用して、ゲームインフラストラクチャをプロアクティブに保護および維持し、セキュリティインシデントによる潜在的な影響を最小限に抑えることができます。セキュリティの問題が検出された場合は、ランブックを使用して状況への対応をガイドします。これらのレスポンスを可能な限り自動化して、問題をより迅速に修正し、影響を軽減します。これにより、ゲームのダウンタイムや中断の可能性が減り、プレイヤーのエクスペリエンスが向上します。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

セキュリティ問題に対処する準備をすることで、プレイヤーのエクスペリエンスを保護するだけでなく、さまざまなコンプライアンスや規制の基準を満たすことができます。さらに、自動セキュリティレスポンスを使用すると、ワークロードが拡大するにつれてセキュリティオペレーションがスケールされます。 は、これらのインシデントへの対応を特定して自動化するのに役立つサービス AWS を提供します。

お客様事例

AnyCompany Games は、定期的なアセットパイプラインの更新中に、今後のゲームのリリースされていないキャラクターモデルとテクスチャを含む S3 バケットが誤って公開されたときに、重大なセ

キュリティインシデントに直面しました。自動セキュリティシステムは、バケットのアクセス許可の変更を、変更後数分以内に検出しました。システムはすぐに修復ランブックを実行しました。バケットをプライベートステータスに戻す、公開ウィンドウ中のアクセス試行の口グ記録、セキュリティチームへの通知、アクセス許可の変更に関する詳細な CloudTrail 口グの作成です。

実装手順

- での自動化されたセキュリティ対応 AWSソリューションを使用して、 のセキュリティイベントに応じて自動的に実行されるアクションを定義する自動化ランブックを実装します AWS Security Hub CSPM。

リソース

- AWS for Games ブログ — での Game Studio の管理 AWS: パート 1
- AWS for Games ブログ — AWS パート 2 での Game Studio の管理
- 既存の組織単位を に登録する AWS Control Tower
- AWS アカウント ルートユーザー
- ルートユーザーの認証情報を必要とするタスク
- での自動化されたセキュリティ対応 AWS

ID とアクセス管理

GAMESEC03: プレイヤーの ID とアクセスの管理はどのように行いますか?

ゲームを開発するときは、プレイヤーにゲームおよび関連サービスへのアクセスを提供する方法を決定する必要があります。次のセクションでは、プレイヤー認証、認可、多要素認証などの設計上の考慮事項について説明します。

ベストプラクティス

- GAMESEC03-BP01 ゲームの環境とリソースへのプレイヤーアクセスを特定して制御するアプローチを決定する
- GAMESEC03-BP02 ゲームバックエンドサービスに送信されるリクエストを認証する
- GAMESEC03-BP03 ゲームバックエンドサービスを使用して、マルチプレイヤーゲームに参加するプレイヤーリクエストを検証する

- GAMESEC03-BP04 強力なパスワードを要求してプレイヤーユーザーアカウントに厳格なセキュリティポリシーを適用する
- GAMESEC03-BP05 プレイヤーが自分のアカウントで多要素認証 (MFA) を設定するオプションを提供します

GAMESEC03-BP01 ゲームの環境とリソースへのプレイヤーアクセスを特定して制御するアプローチを決定する

この決定は、プレイヤーの取得と収益化戦略、プレイヤーエクスペリエンス、およびゲームパブリッシングパートナーによって提供される可能性のある既存の機能などのその他の要因の影響を受けます。例えば、ゲームでは購入が必要になり、プレイヤーは実際の支払い方法をアカウントに関連付けるユーザープロファイルを作成する必要があります。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

または、ゲームをプレイする前にユーザーアカウントを作成する必要がなくなり、プレイヤーが初めてゲームを試す可能性が高まることで、初めてのプレイヤーエクスペリエンスへの参入障壁を減らすこともできます。通常、ゲームはプレイヤーアイデンティティとアクセス管理アプローチの 1 つ以上の組み合わせをゲームに実装します。

認証されていないアクセスまたは匿名アクセス

このアクセスレベルは、プレイヤーが新しいユーザーアカウントを作成したり、ソーシャルネットワークやゲームシステム上のアイデンティティにリンクしたりする必要がない場合に役立ちます。これは、プレイヤーがゲームをプレイする最も簡単で迅速な方法であり、ゲーム開発者が最初のエクスペリエンスのためにエントリの障壁を減らしたいモバイルゲームで特に便利です。

このアクセスシナリオでは、ゲームのインストールから使用状況を特定する場合、一意の識別子を生成してプレイヤーのデバイスに保存するようにゲームクライアントをプログラムできます。この一意の識別子は、デバイス上のゲームセッション間でプレイヤーを識別し、時間の経過に伴う使用状況の分析レポートを許可するために使用されます。後で、プレイヤーがアカウントを作成することを選択した場合、新しいユーザーアカウントを以前に生成された一意の識別子に関連付けることができます。これにより、新しいプレイヤー ID が、統計やゲームアチーブメントなどの過去の使用状況にリンクされます。

プレイヤーが最終的にアカウントを作成してリンクしない場合、プレイヤーがゲームの操作に使用するデバイスは一意に識別できますが、プレイヤーに関する回復可能な情報は収集および保存されませ

ん。したがって、プレイヤーがデバイスを破損または紛失した場合、デバイスに関連付けられた以前の保存データも失われ、回復できない可能性があります。

ユーザー名とパスワードによる認証

ゲームでは、プレイヤーがゲームのバックエンドに保存されているユーザー名とパスワードを使用して独自のユーザーアカウントを作成できる場合があります。これは、ゲーム開発者が、開発者が統合できる既存のプレイヤーアカウントシステムを既に持っているゲームパブリッシャーとコラボレーションしている場合に発生する可能性があります。または、独自のゲームを発行するデベロッパーは、発行するゲーム間でアクセスするための単一のユーザーアカウントをプレイヤーが作成できるようにすることで、プレイヤーエクスペリエンスを簡素化したい場合があります。

サードパーティーのソーシャルネットワークやゲームシステムとの認証とアカウントリンク

オンラインゲームやソーシャル機能を備えたゲームでは、プレイヤーエクスペリエンスを簡素化するためにサードパーティーの ID プロバイダーフェデレーションを提供することが一般的です。認証のためにユーザー名とパスワードの組み合わせを作成するようにプレイヤーに依頼する代わりに、ID フェデレーションを使用して、プレイヤーがソーシャルネットワークやゲームシステムでサードパーティーアカウントを使用して認証できるようにすることができます。このログインプロセスにより、プレイヤーのサインインと登録のエクスペリエンスが簡素化されます。また、必須のアカウント作成に代わる便利な方法と、プレイヤーがゲームにアクセスするためのスムーズな方法も提供します。

ゲーム開発者にとって、フェデレーティッドログインプロセスは、合理化されたプレイヤー検証ワークフローを提供できます。また、パーソナライゼーションに使用されるプレイヤーデータをより信頼性の高い方法で管理することもできます。これは、プレイヤーがサードパーティーの ID プロバイダーに既に提供している可能性がある特定のデータを提供するようプレイヤーに依頼する必要がないためです。さらに、これらのシステムは、プレイヤーを友人にリンクする機能など、追加のソーシャル機能との統合を提供します。

実装手順

- 認証されていないアクセスまたは匿名アクセスを使用して、使用状況を追跡する一意のデバイス識別子を生成し、後でアカウントリンクを有効にすることで、初めてのプレイヤーの障壁を軽減します。
- 既存のプレイヤーアカウントシステムを使用するか、ゲーム間で統一されたエクスペリエンスを作成しながら、専用ユーザーアカウントのユーザー名とパスワード認証を実装します。
- フェデレーション認証用のサードパーティー ID プロバイダーを統合し、ログインプロセスを簡素化し、ソーシャル機能とパーソナライゼーションデータへのアクセスを可能にします。

GAMESEC03-BP02 ゲームバックエンドサービスに送信されるリクエストを認証する

ゲームバックエンドサービスに送信されるリクエストを認証すると、不要なリクエストの成功がブロックされる可能性があります。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

プレイヤーがログインするための認証サービスを提供する必要があります。これにより、プレイヤーが正常に認証されると、JSON ウェブトークン (JWT) などの安全な有効期間の短いトークンがゲームクライアントに返されます。

これらのトークンには、プレイヤー属性やその他の関連するメタデータを含むクレームアサーションを含めることができます。この関連メタデータは、ゲームクライアントからゲームバックエンドに送信される後続のリクエストで使用して、リクエストを認証し、認証されたプレイヤーのコンテキストで承認できます。

継続的な改善とメンテナンスを必要とする独自のプレイヤー認証システムを設計および構築することも、Amazon Cognito が提供するスケーラブルで安全なユーザーサインアップ、サインイン、アクセスコントロール機能を使用することもできます。

Amazon Cognito ユーザープールには、認証と認可のためのユーザーディレクトリが含まれています。ユーザープールは、サインアップ、サインイン、パスワードリセットワークフローのためにゲームに統合できる APIs を提供します。これは、サードパーティーの ID プロバイダーと統合できます。Application Load Balancer と Amazon API Gateway はどちらも Cognito との統合を提供し、これらのサービスでホストされているカスタムゲームバックエンドに送信されるリクエストのユーザー認証を統合します。

ゲームが匿名アクセスをサポートし、プレイヤーを認証できない場合は、クライアント認証アプローチを使用して、ゲームバックエンドと統合するときにより安全なエクスペリエンスを提供できます。ゲームクライアントが AWS サービスを直接使用する場合、これらのサービスへのリクエストは認証情報を使用して署名する必要があります。認証されていないユーザーにゲームクライアントに認証情報を提供するには、 SDK を使用して AWS Amazon Cognito ID プールから有効期間の短い認証情報を取得し、 AWS サービスへのリクエストの署名に使用できます。これらの認証情報はゲームクライアントから更新できます。

ゲームクライアントから AWS SDK と直接統合するだけでなく、カスタム認可をサポートする Amazon API Gateway などのサービスを使用して、独自のゲームバックエンドを構築することもで

きます。独自のゲームバックエンドサービスを設計することで、カスタムサーバー側のロジックを使用してリクエストを権威を持って制御できます。

Amazon GameLift を使用してホストされるゲームのバックエンドサービスを構築する方法の詳細については、「ゲームクライアントサービスの設計」を参照してください。

お客様事例

AnyCompany Games は、マネージド認証および認可アプローチを採用することで、次のタイトルのセキュリティを強化しました。カスタムユーザー名とパスワードシステムを維持する代わりに、Amazon Cognito ユーザープールを使用してプレイヤーのサインアップとサインインを処理し、ID プールを使用して、アカウントを作成する前にトレーニングモードを試すプレイヤーの匿名アクセスをサポートしました。また、ゲーム内にカスタム認可ロジックを実装して、Cognito で定義された管理者ロールを認識し、それらのユーザーにゲーム内の特別な管理機能へのアクセスを許可しました。

実装手順

- Amazon Cognito ユーザープールを使用してJWTs などの安全なトークンによる認証を管理し、サインアップ、サインイン、パスワードリセットなどの機能を有効にします。
- 匿名ユーザーが AWS サービスと安全にやり取りできるように、Amazon Cognito ID プールから存続期間の短い認証情報を取得します。
- カスタムサーバー側の認証ロジックに Amazon API Gateway を使用してカスタムゲームバックエンドを実装します。

GAMESEC03-BP03 ゲームバックエンドサービスを使用して、マルチプレイヤーゲームに参加するプレイヤーリクエストを検証する

通常、マルチプレイヤーゲームでは、プレイヤーは利用可能なセッションのリストからオプションを直接選択してゲームセッションに参加するか、マッチングを見つけるリクエストを送信します。後者のアプローチでは、ゲーム開発者が対象となるゲームセッションを見つけ、接続情報 (通常は IP アドレスとポート番号) をプレイヤーのゲームクライアントに提供する責任があります。実装は、開発するゲームのジャンルによって異なる場合がありますが、ゲームに参加するプレイヤーのリクエストをサーバー側で検証することはセキュリティのベストプラクティスです。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

たとえば、セッションベースのマルチプレイヤーゲームでは、プレイヤーからのゲームセッションへの参加リクエストは、サーバーへの接続を承認する前に、ゲームサーバーソフトウェアとゲームバックエンドマッチメーキングサービスで検証する必要があります。プレイヤーがゲームセッションへの参加をリクエストすると、ゲームサーバーは、プレイヤーセッション ID や、ゲームバックエンドマッチメーキングサービスによってゲームクライアントに以前に提供されたサーバー生成チケットなどの一意の識別子のリクエストをチェックする必要があります。

ゲームサーバーへの接続を開始すると、サーバー側のソフトウェアはこの情報を使用して、プレイヤーの接続リクエストが有効であることをマッチメーキングサービスで検証し、プレイヤーが以前に別のプレイヤーのゲームセッションで予約されていたスポットに参加していないことを確認できます。

Amazon GameLift でホストされているゲームについては、このタイプのサーバー側の検証を実装する方法の例については、「Amazon GameLift サーバーとのゲームクライアント/サーバーインタラクション」を参照してください。

お客様事例

AnyCompany Games の最初のベータローンチ中に、プレイヤーがゲームサーバーに直接接続してマッチメーキングシステムをバイパスしていることを発見し、深刻な競争整合性の問題を引き起こしました。ランクの高いプレイヤーは、サーバー IP アドレスを友人と共有できることに気付いたとき、スキルベースのマッチメーキングシステムの回避を開始しました。その結果、経験豊富なプレイヤーは初心者試合に参加し、新しいプレイヤーに苛立たしい経験をします。AnyCompany Games は、マッチメーキングリクエストごとに一意のセッションチケットを生成するサーバー側の検証システムを実装することで応答しました。システムでは、プレイヤー IDs とマッチメーキングリクエストチケットの両方が必要で、バックエンドマッチメーキングサービスに対して検証済みの接続試行が必要でした。

実装手順

- プレイヤーセッション IDsやサーバー生成チケットなどの一意の識別子を使用して、プレイヤー参加リクエストをサーバー側で検証します。
- 不正アクセスをブロックするために、マッチメーキングサービスとの接続リクエストの有効性を確認します。
- 検証プロセス中に、ゲームセッションの予約済みスポットが権限のないプレイヤーによってアクセスされていないことを確認します。

GAMESEC03-BP04 強力なパスワードを要求してプレイヤーユーザーアカウントに厳格なセキュリティポリシーを適用する

ゲームでプレイヤーがパスワードを使用してユーザーアカウントを作成できるようにする場合は、強力なポリシーを遵守するためにプレイヤーのパスワードを要求する必要があります。例えば、Amazon Cognito ユーザープールを使用すると、ユーザーアカウントのパスワード要件を定義できます。強力なパスワードポリシーを確立することで、ソーシャルエンジニアリングやブルートフォース攻撃によってプレイヤーのアカウントが乗っ取られるのを防ぐことができます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

お客様事例

AnyCompany Games は、パスワードポリシーが弱いために人気のあるタイトルでアカウント乗っ取りの波が発生したときに、危機に直面しました。「password123」などのシンプルなパスワードを使用していたプレイヤーは、自動ブルートフォース攻撃の犠牲者になり、アイテムが失われ、ゲーム内の通貨が侵害されました。これに対処するために、AnyCompany Games はログインシステムを改訂し、パスワードを以前に使用しないことを義務付けました。これには、少なくとも 1 つの大文字、1 つの数字、1 つの特殊文字、15 文字以上の長さを含める必要があります。

実装手順

- セキュリティを強化するために、プレイヤーアカウントに強力なパスワードポリシーが必要です。
- Amazon Cognito ユーザープールを使用して、パスワード要件を定義して適用します。

GAMESEC03-BP05 プレイヤーが自分のアカウントで多要素認証 (MFA) を設定するオプションを提供します

プレイヤーアカウントは、特にゲーム内の通貨と購入をサポートするゲームでは、悪意のあるアクターのアセットになる可能性があります。プレイヤーアカウントのハッキングやソーシャルエンジニアリング攻撃が蔓延しているため、多要素認証 (MFA) を設定してアカウントのセキュリティを強化するオプションをプレイヤーに提供します。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

プレイヤーが MFA を使用してアカウントにアクセスしようとすると、E メールアドレス、電話番号、または専用多要素認証モバイルアプリに一時コードが送信されます。正常に認証するには、プレイヤーは制限された期間内にログインシステムにコードを入力する必要があります。

MFA は、新しい位置情報からの認証を試みているアカウント、プレイヤーのサポートによって潜在的な悪意のあるアクティビティのフラグが付けられたアカウント、および長期間ゲームにログインしていないアカウントを保護するためにも使用できます。

例えば、Amazon Cognito ユーザープールは、ユーザーディレクトリで多要素認証を設定できます。

実装手順

- 多要素認証 (MFA) を有効にして、プレイヤーアカウントのセキュリティを強化します。
- E メール、電話、または MFA アプリを介して送信された一時コードを使用して、アカウントアクセスを検証します。
- 新しい位置情報、フラグ付きアカウント、または長い非アクティブ状態のアカウントに MFA を適用します。

アクセスコントロール

GAMESEC04: ゲームコンテンツへの不正アクセスをブロックするにはどうすればよいですか?

最新のゲームには、プレイヤーエンゲージメントやゲーム収益化の重要な側面であるダウンロード可能コンテンツ (DLC) など、大量のコンテンツが含まれています。プレイヤーは、新しいキャラクター、レベル、チャレンジの継続的なストリームを期待し、ゲーム開発者はプレイヤーを保持するための新しいコンテンツに対する継続的な需要に対応する必要があります。コンテンツの種類とサイズは、ゲームのタイプとゲームが再生されるデバイスによって大きく異なる場合があります。ゲームのシステムに関係なく、ゲームのコンテンツを不正アクセスから保護します。

ベストプラクティス

- GAMESEC04-BP01 ダウンロード可能なコンテンツへのアクセスを承認されたクライアントとユーザーに制限する
- GAMESEC04-BP02 オリジンアクセスを承認されたコンテンツ配信ネットワーク (CDNs

- GAMESEC04-BP03 不正アクセスを制限するための地理的制限を実装する
- GAMESEC04-BP04 デジタル著作権管理 (DRM) ソリューションを使用してコンテンツへのアクセスを制限する

GAMESEC04-BP01 ダウンロード可能なコンテンツへのアクセスを承認されたクライアントとユーザーに制限する

承認されたアプリケーションとクライアントによるゲームコンテンツへのアクセスを制限します。ダウンロード可能なゲームコンテンツを保存するための費用対効果が高くスケーラブルなオリジンとして Amazon S3 を使用することと、プレイヤーにグローバルにパフォーマンスの高いコンテンツ配信を提供するために Amazon CloudFront を使用することを検討してください。どちらのサービスも、認証されたユーザーへのアクセスを制限するなど、保存されているデータへのアクセスを制限する組み込みメカニズムを提供します。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

Amazon S3 に保存されているコンテンツへのアクセスの許可

S3 に保存されているコンテンツへのアクセスを許可する必要がある場合は、いくつかの考慮事項があります。デフォルトでは、S3 バケットを AWS アカウント 作成した のみが、そのバケットに保存されているオブジェクトにアクセスできます。内部アプリケーションへのアクセスを許可し、Amazon S3 バケットに保存されているコンテンツを管理するには、 AWS Identity and Access Management (IAM) を使用して適切なアクセスを提供するポリシーを作成します。

IAM ロールは、Amazon EC2 などの サービスでホストされているフェデレーティッドユーザー、システム AWS Lambda、またはアプリケーション、および Amazon EKS および Amazon ECS でホストされているコンテナベースのアプリケーションに関連付けることができます。例えば、 AWS SDK または を使用して AWS CLI 、S3 バケット内のゲームコンテンツアセットを公開および管理できます。このユースケースをサポートするには、適切なアクセス権を持つ IAM ロールを作成してゲームコンテンツを S3 バケットに読み書きし、ソフトウェアとスクリプトをホストする EC2 インスタンスに関連付けることができます。

リソースベースのポリシーは、バケットと特定のオブジェクトに対して定義できます。S3 バケットポリシーは S3 バケットに関連付けられており、バケットおよびバケット内のオブジェクトへのアクセスを制限したり、他のアカウントから Amazon S3 リソースへのアクセスを許可したりできます。例えば、複数のチームまたは個別のゲーム開発スタジオが同じゲームコンテンツに取り

組み、Amazon S3 で一元的にホストされたコンテンツへの同じアクセスを必要とするシナリオでは、S3 バケットポリシーを使用して、S3 リソースへのクロスアカウントアクセスのアクセス許可を定義できます。各アプリケーションまたはアプリケーションセットに固有の名前とアクセス許可を持つアクセスポイントを作成することで、共有データへのデータアクセスの管理を簡素化できる S3 アクセスポイントの使用を検討してください。Amazon S3 ドキュメントには、Amazon S3 でのアクセスコントロールに関する追加のベストプラクティスが含まれています。

Granting short-term access to your content

アクセスが特定の期間限定のみ必要な場合は、コンテンツへの短期アクセスを許可する一時的な URLsを生成します。Amazon S3 では、署名付き URLs の生成がサポートされています。これにより、オブジェクト所有者はバケットポリシーを更新せずに Amazon S3 内のオブジェクトに時間制限付きアクセスを許可できます。これにより、アクセスが付与されているエンドユーザーまたはアプリケーションは、アカウントまたは IAM アクセス許可を持つ必要はなく、代わりに署名付き URL を使用してコンテンツにアクセスします。

これは、権限のあるプレイヤーにダウンロード可能なコンテンツへのアクセス権を付与したり、期間限定のゲームコンテンツへの一時的なアクセスを提供したりするなど、さまざまなゲームのユースケースで一般的に使用されるベストプラクティスです。署名付き URLs を使用して、S3 バケットにコンテンツをアップロードするための一時的なアクセス許可を付与することもできます。例えば、署名付き URL を使用して、サポートチームがプレイヤーサポートケースのトラブルシューティングを支援するためのクライアントログをアップロードするためのアクセスをプレイヤーに提供することを検討できます。

コンテンツ配信ネットワークを使用してコンテンツへのアクセスを提供する

アプリケーション、ゲーム開発者、アーティスト、その他の担当者は、開発および管理の目的で S3 バケット内のコンテンツに直接アクセスする必要がある場合がありますが、コンテンツ配信ネットワークを使用して、インターネット経由でプレイヤーや他のユーザーが公開しているコンテンツへのアクセスを提供します。このアプローチは、頻繁にアクセスされるコンテンツをキャッシュすることで、ダウンロードパフォーマンスを向上させ、コストを削減します。Amazon CloudFront は、Amazon S3 などのゲームのダウンロードオリジンの負荷を軽減しながら、キャッシュしてプレイヤーの近くに配信することで、コンテンツをグローバルに配信できます。

S3 バケットから直接パブリックコンテンツを提供するのではなく、このコンテンツをプライベートに保ち、CloudFront を使用してパブリックに配信することをお勧めします。CloudFront は、署名付き URLs または署名付き Cookie を使用して、プライベートコンテンツ (有料プレイヤーのみの新しいゲームダウンロードなど) へのアクセスをプレイヤーに要求するように設定できます。その後、アプリケーションを開発して、署名付き URLs を作成して認証されたユーザーに配布するか、認証さ

れたユーザーに署名付き Cookie を設定する set-cookie ヘッダーを送信できます。ファイルへのアクセスを制御するために署名付き URLs または署名付き Cookie を作成する場合、終了日時を指定できます。その後、URL と Cookie は無効になります。

オプションで、コンテンツへのアクセスに使用できるコンピュータの IP アドレスまたはアドレス範囲を指定することもできます。これは、特定のゲーム開発スタジオパートナーまたは請負業者ネットワークへのアクセスを制限する場合に便利です。複数の制限付きファイルへのアクセスを許可する場合、または現在の URLs を変更しない場合は、署名付き Cookie を使用します。個々のファイルへのアクセスを制限する場合や、ユーザーが Cookie をサポートしていないクライアントを使用している場合は、署名付き URLs を使用します。署名付き URL は署名付き Cookie よりも優先されます。

実装手順

- IAM ロールとバケットポリシーを使用して、内部アプリケーション、チーム、またはクロスアカウントシナリオの S3 バケットへの適切なアクセスを許可します。
- S3 オブジェクトへの短期アクセスを許可するための署名付き URLs を生成します。ダウンロード可能なコンテンツやクライアントログなどの一時アップロードに適しています。
- 署名付き URLsまたは Cookie で Amazon CloudFront を使用して、認証されたユーザーにプライベートコンテンツをより安全に提供

GAMESEC04-BP02 オリジンアクセスを承認されたコンテンツ配信ネットワーク (CDNs

ユーザーがコンテンツ配信ネットワークを回避して、Amazon S3 バケットなどのオリジンからコンテンツに直接アクセスすることをブロックします。オリジンへのアクセスを承認された CDNs のみに制限することが重要です。これにより、オリジンからコンテンツを不必要に提供するデータ転送コストを削減できます。また、同じエントリポイントを介してオリジンコンテンツへのパブリックアクセスをフローすることで、セキュリティ体制を強化します。これにより、 AWS WAF レイヤー 7 フィルタリング、セキュリティ関連の HTTP リクエストパラメータのインジェクションと検査、分散サービス拒否 (DDoS) 保護などのエッジセキュリティコントロールをデプロイできます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

Amazon S3 オリジンにこれらのコントロールを実装するには、Amazon CloudFront オリジンアクセスアイデンティティ (OAI) を使用して、S3 オブジェクトへのリクエストが CloudFront ディストリビューションから発信されていることを確認します。CloudFront ディストリビューション AWS

WAF に関連付けて、レイヤー 7 フィルタリングを提供します。ただし、追加の CDNs からコンテンツを提供する場合は、1 つ以上のカスタム HTTP ヘッダーをオリジンリクエストに挿入するように CDN を設定できます。オリジンリクエストは、 によって検査 AWS WAF され、受信トラフィックが承認された CDN プロバイダーから発信されたことを確認できます。

このアプローチは、オリジンが Application Load Balancer (ALB) の背後でホストされているときに、ユーザーが CDN プロバイダーを回避できないようにするためにも役立ちます。ALBs は Layer-7 保護のために に関連付ける AWS WAF ことができます。ALB によって検査されるカスタム HTTP ヘッダーを挿入して、ロードバランサーへの受信トラフィックを処理して検査 AWS WAF するように を設定できます AWS WAF。

お客様事例

AnyCompany Games は、ゲームアセット、ダウンロード可能なコンテンツ、パッチファイルを不正な直接アクセスから保護するのに役立つオリジンアクセス制限を実装しています。これにより、プレイヤーは適切な認証なしでセキュリティチェックをバイパスしたり、プレミアムコンテンツを取得したりできます。このアプローチにより、一元的なポイントを通じてコンテンツアクセスパターンをモニタリングできるため、協調攻撃や不正なコンテンツ再配布の存在を示す可能性のある疑わしいダウンロード動作を簡単に特定できます。

実装手順

- Amazon CloudFront オリジンアクセスアイデンティティ (OAI) を使用して S3 オブジェクトへの直接アクセスを制限する
- CloudFront または ALB AWS WAF に関連付けてレイヤー 7 フィルタリングを提供し、DDoS 攻撃や悪意のあるリクエストから保護します。
- Cloudfront でカスタム HTTP ヘッダーを設定して、受信トラフィックが承認されたソースから発信されていることを確認します。

GAMESEC04-BP03 不正アクセスを制限するための地理的制限を実装する

プレイヤーがコンテンツをリクエストすると、Amazon CloudFront はプレイヤーの場所に関係なく、最も近いエッジロケーションからリクエストされたコンテンツを提供します。ただし、世界中の特定の地域のユーザーがコンテンツにアクセスする方法を制限する必要があるシナリオもあります。たとえば、ローリングゲームのデプロイ戦略では、country-by-country段階的にコンテンツをリリースしたり、国固有のアクセスコントロールに従う必要があります。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

地理的ブロックとも呼ばれる地理的制限を使用して、特定の地理的場所のプレイヤーが CloudFront ディストリビューションを介して配信するコンテンツにアクセスするのをブロックできます。この機能を使用すると、ディストリビューションに関連付けられているファイルへのアクセスを制限し、国レベルでアクセスを制限できます。または、サードパーティーの位置情報サービスを使用して、ディストリビューションに関連付けられているファイルのサブセットへのアクセスを制限したり、国レベルよりも細かくアクセスを制限したりできます。

CloudFront の地理的制限を使用すると、承認された国の許可リストに登録されている国のいずれかにいる場合のみ、プレイヤーがコンテンツにアクセスすることを許可できます。禁止されている国の拒否リストに登録されている国のいずれかにいる場合、プレイヤーがコンテンツにアクセスするのをブロックすることもできます。ブロックされた地理的場所からリクエストを受信した場合、CloudFront は 403 Forbidden HTTP ステータスコードをプレイヤーに返します。これは機密性の高いコンテンツには適しており、PII または機密性の高いゲームアーティファクトのスタンドアロン保護として使用しないでください。

実装手順

- CloudFront の地理的制限を使用して、国レベルの許可または拒否リストに基づいてコンテンツアクセスを許可または拒否します。
- ブロックされた地理的場所から発信されたリクエストに対して 403 Forbidden HTTP ステータスコードを返します。
- 機密コンテンツまたは PII を保護するために地理的制限のみに頼らないようにする

GAMESEC04-BP04 デジタル著作権管理 (DRM) ソリューションを使用してコンテンツへのアクセスを制限する

デジタル著作権管理 (DRM) ソリューションなどの強力な暗号化ツールを使用して、ゲームコンテンツへのアクセスを制限することを検討してください。このタイプのソリューションは、プライベートコンテンツを暗号化し、復号キーを認可されたプレイヤーに配布するために使用できます。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

DRM ソリューションは、プレイヤーがゲームコンテンツを早期にダウンロードできるようにしたいが、事前に決められた時間までコンテンツにアクセスまたは再生できないようにする場合に推奨さ

れます。例えば、これはプレイヤーがゲームの事前注文を許可され、暗号化されたファイルのダウンロードを早期に自動的に開始するようにゲームクライアントを設定できる状況で一般的です。この戦略では、ゲームが正式にリリースされたら、ゲームがダウンロードされ、プレイされる準備ができていることを確認します。ゲームがリリースされると、プレイヤーのゲームクライアントは DRM バックエンドソリューションに復号キーをリクエストして、以前にダウンロードしたファイルを復号してゲームのプレイを開始できます。

DRM システムは、承認されたプレイヤーによってダウンロードおよびインストールされたゲームの不正な再配布と操作をブロックするためにも使用されます。DRM システムでは、暗号化キーを交換し、プレイヤーに復号キーの取得を許可するために、オリジンとの統合が必要です。商用 DRM プロバイダーは、さまざまなデバイスの機能とサポートを備えた幅広いソリューションを提供します。

実装手順

- DRM ソリューションを使用してプライベートゲームコンテンツを暗号化し、復号キーを承認されたプレイヤーに配布します。
- 事前注文されたゲームの暗号化ファイルの事前ダウンロードを有効にし、リリース時に復号キーを使用してアクセスをロック解除します。
- DRM システムをオリジンと統合して暗号化キーを管理し、コンテンツの不正な再配布や操作をブロックします。

検出

GAMESEC05: ゲーム内のプレイヤーの使用状況をどのようにモニタリングおよび分析しますか?

プレイヤーの使用状況のモニタリングと分析は、ゲームスタジオにとって不可欠です。ゲームの整合性とプレイヤーの安全性を損なう可能性のあるセキュリティの脅威、不正行為、その他の不正行為を検出できるためです。異常な進行率、異常なゲーム内トランザクション、疑わしい通信動作などのパターンを追跡することで、潜在的なチーター、不正なアカウント、調整された脅威をプレイヤーエクスペリエンスに大きな影響を与える前に特定できます。

ベストプラクティス

- GAMESEC05-BP01 プレイヤーの動作をモニタリングするための包括的なデータ収集戦略を実装する
- GAMESEC05-BP02 プレイヤー使用状況ログを収集、保存、分析して不適切な動作を検出する

GAMESEC05-BP01 プレイヤーの動作をモニタリングするための包括的なデータ収集戦略を実装する

ポジティブなプレイヤーエクスペリエンスを維持するには、包括的なデータ収集と分析戦略を実装します。関連するデータをキャプチャ、保存、分析することで、プレイヤーがゲームの機能や相互にやり取りする方法に関するインサイトが得られます。このデータ駆動型アプローチは、意思決定の指針となり、プレイヤーのエンゲージメントとリテンションを強化し、収益化戦略を最適化し、最終的にプレイヤーエクスペリエンス全体を向上させることができます。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

ゲームプレイセッション、進行状況、アチーブメント、購入、ゲーム要素とのやり取り、ソーシャルアクティビティなど、関連するプレイヤーアクションをキャプチャしてログに記録するデータ収集システムを実装します。サーバー負荷、ネットワークトラフィック、エラーログなどのサーバー側のデータを収集して、技術的なパフォーマンスをモニタリングし、潜在的な問題を特定します。アンケート、フォーラム、サポートチケット、ソーシャルメディアチャネルを通じてプレイヤーのフィードバックを収集し、プレイヤーの経験と好みを理解します。

ゲームデータを保存するときは、一元化されたデータウェアハウスまたはデータレイクを確立して、収集されたデータを保存および整理し、データのクリーニング、変換、集約のためのパイプラインを実装して、効率的な分析のためにデータを準備します。

データを保存したら、それを分析して、プレイヤーの保持と解約、収益化戦略、データ視覚化ツールによる機能の使用などのインサイトを取得します。

実装手順

- プレイヤーアクション、サーバー側のメトリクス、フィードバックを取得してログに記録し、インタラクションと技術パフォーマンスをモニタリングします。
- Amazon Redshift や S3 データレイクなどの一元化されたデータウェアハウスを使用して、分析のためにゲームデータを保存、クリーンアップ、変換、整理します。
- Amazon Quicksight などの視覚化ツールを使用して収集されたデータを分析し、プレイヤーの保持、収益化、機能の使用状況に関するインサイトを取得します。

GAMESEC05-BP02 プレイヤー使用状況ログを収集、保存、分析して不適切な動作を検出する

ゲームを計測してログを収集し、プレイヤーがゲームの機能をどのように使用し、他のプレイヤーとどのようにやり取りするかを理解します。その後、プレイヤーのエクスペリエンスを低下させる可能性のある不正なアクティビティをブロックできます。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

Amazon Amazon CloudWatch Logs や Amazon OpenSearch Service などのログ記録ソリューションを使用するか、Datadog、Sumo Logic、New Relic、Honeycomb.io、Splunk などの パートナーからのソリューションを使用して、構造化ログイベントを AWS Game Analytics Pipeline に送信します。これらのプレイヤー使用状況ログを構造化して、プレイヤーによる特定のアクションを調査する必要があるタイミングを検出できるようにします。

データを取得したら、不適切な使用動作を検出するためのツールの実装を検討してください。たとえば、ゲームにゲーム内プレイヤーメッセージング、音声チャット、オンラインフォーラムなどのソーシャル機能がある場合は、モデレーション目的で分析できる形式でこれらのプレイヤーエンゲージメントのログを保存します。

録音を Amazon S3 にエクスポートするようにゲームの音声チャット機能を設定し、Amazon Transcribe を使用して音声をテキスト形式に変換し、処理のために保存できます。または、ゲームバックエンドの音声チャットサービスを Transcribe API と直接統合してストリーミング音声をリアルタイムで文字起こしすることで、リアルタイムのストリーミング文字起こしを実行することもできます。モデレーションチームはコンテンツを手動で確認でき、コンテンツが標準形式になったら、 AWS AI/ML サービスを使用してモデレーションを自動的に実行することもできます。Amazon Comprehend を使用して自然言語処理 (NLP) を実行し、非構造化テキストから情報を発見できます。これにより、会話を関連するトピックに分類して整理し、冒涜的な行為などの不適切な行動を特定できます。

実装手順

- プレイヤーの使用状況ログを収集、保存、分析します。
- 人工知能と機械学習 AWS のサービスを使用して、プレイヤーの使用状況ログをより効率的に確認してインサイトを得ることができます。

インフラストラクチャの保護

ゲームワークロードに適用されるセキュリティのためのインフラストラクチャ保護のベストプラクティスについては、 Well-Architected フレームワークホワイトペーパーを参照してください。

GAMESEC06: インフラストラクチャの脅威をモニタリングして対応するにはどうすればよいですか?

インフラストラクチャの脅威のモニタリングと対応は、ゲームスタジオにとって重要です。インフラストラクチャは、何百万人もの同時プレイヤーをサポートし、リアルマネートランザクションを処理し、貴重なプレイヤーデータと独自のゲームコンテンツを保存するバックボーンであるためです。ゲームスタジオでは、プレイヤーエクスペリエンスとビジネスオペレーションの両方の整合性を維持できるモニタリングシステムとインシデント対応プロセスを実装することが不可欠です。

ベストプラクティス

- GAMESEC06-BP01 インフラストラクチャへの脅威を検出して対応するためのツールを使用する
- GAMESEC06-BP02 人工知能と機械学習ツールを使用してインフラストラクチャ保護戦略の側面を自動化する
- GAMESEC06-BP03 システムレベルのログからのインサイトを使用してインフラストラクチャ保護戦略を継続的に改善する

GAMESEC06-BP01 インフラストラクチャへの脅威を検出して対応するためのツールを使用する

AWS 環境内の悪意のあるアクティビティや不正な動作を継続的にモニタリングするには、Amazon GuardDuty の使用を検討してください。GuardDuty は、環境内のアカウント動作、ネットワークアクティビティ、データアクセスパターンをモニタリングすることで脅威を特定します。CloudTrail イベントログ、Amazon VPC フローログ、DNS ログなど、複数のデータソースのイベントを分析し、潜在的な脅威がないか調べます。Amazon CloudWatch Events および Lambda と統合することで、GuardDuty アラートを関連するセキュリティチームに自動的に転送して詳細な分析を行うことができます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

AWS Security Hub CSPM は、 のセキュリティ状態を包括的に把握 AWS し、セキュリティ業界標準とベストプラクティスに照らして環境をチェックします。Security Hub CSPM は AWS アカウント、、 サービス、およびサポートされているサードパーティーパートナー製品全体からセキュリティデータを収集し、セキュリティの傾向を分析し、最も優先度の高いセキュリティ問題を特定します。Amazon GuardDuty と Security Hub CSPM の統合により、GuardDuty から Security Hub CSPM に検出結果を送信できます。Security Hub CSPM は、これらの検出結果をセキュリティ体制の分析に含めることができます。

悪意ある攻撃者がボットを採用してアカウントを乗っ取り、ゲームで不正行為を行うことは一般的です。WAF Bot Control は、過剰なリソースを消費したり、メトリクスを歪めたり、ダウンタイムを引き起こしたり、その他の望ましくないアクティビティを実行したりする可能性のある一般的および広範なボットトラフィックを可視化し、制御します。

ランサムウェアは、システムやデータセットへの不正アクセスを取得し、そのデータを暗号化して正当なプレイヤーによるアクセスをブロックするように設計された悪意のあるコードです。ランサムウェアがシステムからプレイヤーをロックアウトし、機密データを暗号化した後、サイバー犯罪者はデータのロックを解除するための復号キーを提供する前に身代金を要求します。組織は悪意のあるイベントによって完全にシャットダウンされ、多大なコストが発生し、ビジネスの生産性が失われる可能性があります。インシデントの発生前、発生中、発生後にランサムウェアと戦う能力を強化するために適用できるベストプラクティスについては、「ランサムウェアから AWS クラウド 環境を保護する」を参照してください。

ゲームでは、Amazon Amazon Connect Amazon Lexを使用するチャットボットなどのコールセンターを通じてプレイヤーサポートエージェントに連絡できます。Amazon Connect は、ライブ会話と録音会話のモニタリングをサポートします。 https://docs.aws.amazon.com/connect/latest/adminguide/monitoring-amazon-connect.htmlAmazon Lex で構築されたプレイヤーとプレイヤーサポートチャットボット間のインタラクションを分析するには、これらのインタラクションの会話ログを Amazon CloudWatch Logs に保存できます。このログは、前述のように Amazon S3 にエクスポートして分析できます。

最後に、インフラストラクチャ保護戦略の一環として侵入テスト演習を実施します。これらの評価を社内で実行する場合でも、 AWS パートナーを通じて実行する場合でも、AWS ペネトレーションテストに関するカスタマーサポートポリシーに従ってください。

実装手順

- Amazon GuardDuty を使用して、アカウントの動作、ネットワークアクティビティ、データアクセスパターンの脅威をモニタリングし、Security Hub CSPM と統合してセキュリティビューを統一します。
- AWS WAF Bot Control を実装して、リソースやプレイヤーのエクスペリエンスを損なう可能性のあるボットトラフィックを検出して軽減します。
- セキュリティ体制を評価および強化するために、 AWS カスタマーサポートポリシーに従って、定期的に侵入テストの演習を実施します。

GAMESEC06-BP02 人工知能と機械学習ツールを使用してインフラストラクチャ保護戦略の側面を自動化する

Amazon Lookout for Metrics は、機械学習を使用して、ビジネスデータと運用データの異常を自動的に検出して診断し、ビジネスにとって最も重要なメトリクスをより迅速かつ正確にモニタリングします。また、このサービスでは、収益の急激な減少、ログイン、トランザクション、保持など、異常の根本原因を簡単に診断できます。ゲームデベロッパーが ML エクスペリエンスをセットアップする必要はなく、Amazon S3、Amazon CloudWatch、Amazon RDS、Amazon Redshift、および多くの SaaS アプリケーションなどの一般的なデータソースに接続できます。例えば、Amazon Lookout for Metrics を Game Analytics Pipeline やその他のデータソースと統合して、動作の分析を開始して異常を検出できます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

または、Amazon SageMaker AI を使用してカスタム機械学習モデルを構築、トレーニング、ホストし、コンテンツのモデレーション、毒性検出、不正検出などのユースケースに対処することもできます。

お客様事例

AnyCompany Games は Amazon Lookout for Metrics を使用して、サーバーのパフォーマンス、プレイヤーのログイン試行、または不正な攻撃者からの脅威を示す可能性のあるトランザクションボリュームの異常なパターンを自動的に検出します。さらに、Amazon SageMaker AI を使用して、ネットワークトラフィックパターンとプレイヤーの動作を継続的に分析するカスタム機械学習モデルを開発し、仮想経済を悪用しようとしているボットネットワークなど、調整された脅威を特定するのに役立ちます。

この自動化されたアプローチにより、セキュリティチームは、何千ものメトリクスを手動でモニタリングするのではなく、真の脅威の調査と対応に集中できます。同時に、新しい脅威パターンがゲームの可用性やプレイヤーの安全性に大きな影響を与える前に検出され、対処されるようにします。

実装手順

- Amazon Lookout for Metrics を使用して、主要なビジネスデータと運用データの異常を自動的に検出して診断するのに役立ちます。
- Amazon Lookout for Metrics を Game Analytics Pipeline、Amazon S3、CloudWatch などのデータソースと統合して、収益、ログイン、保持などのメトリクスをモニタリングします。
- Amazon SageMaker AI を使用して、不正検出、不正防止、コンテンツモデレーションなどの高度なユースケース向けにカスタム機械学習モデルを構築、トレーニング、ホストします。

GAMESEC06-BP03 システムレベルのログからのインサイトを使用してインフラストラクチャ保護戦略を継続的に改善する

S3 サーバーアクセスログ、CloudFront アクセスログ、ALB アクセスログなどの関連サービスからシステムレベルのログをキャプチャして保存します。 CloudFront これらのログは アカウントの S3 バケットに保存でき、ゲーム内からのプレイヤー使用状況情報を、IP アドレス、リクエストヘッダー、ゲームバックエンド内で設定した可能性のある関連するリクエスト操作やフィルタリングなどの接続の詳細を含むシステムレベルの情報に関連付けるのに役立ちます。これらのログは、前述のものと同じログ記録ソリューションに送信でき、ログを Amazon Athena で SQL クエリを使用して分析できます。 Amazon S3

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

Access Analyzer for S3 は、バケットアクセスポリシーをモニタリングし、ポリシーが Amazon S3 リソースへの意図したアクセスのみを提供する機能です。Access Analyzer for S3 はバケットアクセスポリシーを評価し、意図しない可能性のあるバケットを検出して迅速に修復できるようにします。

実装手順

- 脅威検出とインシデント対応に AWS サービスを使用して、インフラストラクチャ保護戦略の側面を自動化します。
- 人工知能と機械学習のためのシステムレベルのログと AWS サービスを通じて、インフラストラクチャ保護に関するインサイトを取得します。

データ保護

ゲームを開発して設計するときは、スタジオが収集しているデータの種類と、その保護方法を検討してください。セキュリティのこの側面で調べるトピックは次のとおりです。

- データを識別および分類するために選択した方法
- 保管中のデータを保護する方法
- 転送中のデータを保護する方法

Games Lens に固有のデータ保護のベストプラクティスはありません。セキュリティのためのデータ保護のベストプラクティスについては、 Well-Architected フレームワークホワイトペーパーを参照してください。

インシデントへの対応

GAMESEC07: プレイヤーの不正行為や虐待行為に対応するためのポリシーをどのように定義して適用していますか?

プレイヤーの不正行為や虐待的な行動は、プレイヤーのエクスペリエンスに大きな影響を与える可能性があります。不正なプレイヤー動作は正当なプレイヤーを遠ざける可能性があり、プレイヤーの定着率の低下、ゲーム内購入による収益の減少、ゲームの評価と将来の売上を損なう可能性のある否定的なレビューにつながります。

プレイヤー間でポジティブなアクションを促進するポリシーを定義し、これらのポリシーをどのように適用するかを決定します。

ベストプラクティス

- GAMESEC07-BP01 不正行為者や不正行為に対処するインシデント対応計画を実装する
- GAMESEC07-BP02 不正なアクターに関連付けられているアカウントを禁止する

GAMESEC07-BP01 不正行為者や不正行為に対処するインシデント対応計画を実装する

ゲームの不正なアクターや虐待的な行動に対応するためのアクションプランを作成します。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

プレイヤーを一時的に停止または永続的に禁止するタイミングや、一時的に停止するプレイヤーの認証情報を無効にする期間などの要因を考慮してください。

お客様事例

AnyCompany Games は、不適切なチャットメッセージなどの軽微な違反により 24 時間の自動的なアカウント停止が発生する階層型インシデント対応システムを作成します。一方、不正行為やハラスメントなどのより重大な違反により、人間のモデレーターによるレビューが必須の 7 日間の即時停止がトリガーされます。

さらに、AnyCompany Games は、繰り返し違反者が徐々に長い停止に直面するエスカレーション手順を確立します。ID 検証要件を通じてセキュリティを維持しながら、誤ってフラグ付けされたプレイヤーが自動アクションに反対できるようにする要請プロセスを作成します。

GAMESEC07-BP02 不正なアクターに関連付けられているアカウントを禁止する

ゲームの虐待的な動作を緩和しないままにすると、他のユーザーのゲームエクスペリエンスに悪影響を及ぼし続ける可能性があるため、できるだけ早く軽減する必要があります。サービス条件に違反していると確認された不正行為者に禁止やその他の形式の制限を課すプロセスを実装します。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

通常、これらのタイプの制限を課す状況を決定するためのルールと評価プロセスは、組織内のプレイヤーコミュニティチームや信頼安全チームなどの担当者によって決定されます。不正なアクターにフラグを付けたら、事前に定義されたワークフローを実行して、特定されたプレイヤーに対してアクションを実行します。

たとえば、 関数AWS Step Functionsと AWS Lambda関数を使用して、プレイヤーアカウントのバッチを入力として受け入れる自動ワークフローを実行できます。次に、ワークフローは Bans という Amazon DynamoDB テーブルのエントリを更新します。これには、プレイヤーアカウント、禁止理由、期間に関する詳細を含めることができます。

ゲームとアカウント管理システムの設計と、悪意のあるアクターから発生する不正使用の種類に応じて、アカウント管理システムとは別の記録禁止システムを維持します。プレイヤーのアカウントをア

カウント管理システムからオフにしたくない場合があります。代わりに、プレイヤーのゲームプレイ機能をオフにします。これは、プレイヤーのアカウント認証情報を使用して、異なる利用規約またはポリシーで複数のゲームにアクセスする場合に便利です。

実装手順

- 不正行為者からの虐待的な行動に対応するためのポリシーを定義して適用します。
- AWS サービスを使用して、悪意のある攻撃者への応答を自動化します。

リソース

- AWS セキュリティインシデント対応テクニカルガイド
- AWS Machine Learningブログ: Amazon Rekognition Face Liveness を使用して実際のユーザーとライブユーザーを検出し、悪意のあるアクターを抑止する
- AWS ゲームのソリューション: コミュニティヘルス

アプリケーションのセキュリティ

GAMESEC08: CI/CD パイプラインを保護する方法

ゲーム開発 CI/CD パイプラインは通常、高可用性のソースコントロールサーバーとストレージ、ビルドを実行するコンピューティングリソース、および自動テストを実行するソフトウェアと、開発マシンからの適切なネットワーク接続で構成されます。CI/CD パイプラインを保護することは、機密情報を保護し、コードの整合性を維持し、信頼できるリリースを維持する上で重要です。ガバナンスとガードレールを埋め込むと、適切なセキュリティプラクティスを維持しながら、開発者の俊敏性が可能になります。

ゲームは多くの場合、支払い処理を処理し、個人情報を保存し、実質的な価値のある仮想経済を維持するため、開発プロセスのセキュリティ違反は、多額の財務損失、規制上の罰則、プレイヤーの信頼の喪失につながる可能性があります。

保護を統合することで、組織はソフトウェア配信プロセスの可視性と制御を維持し、迅速なインシデント対応を可能にし、安全なコーディングプラクティスの文化を促進します。

ベストプラクティス

GAMESEC08-BP01 CI/CD パイプラインのすべての段階でセキュリティを適用する

アクセスコントロール、職務の分離、監査証跡などのガードレールは、不正アクセスや悪意のあるアクティビティから保護します。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

人材、プロセス、テクノロジーもパイプラインを保護する必要があります。コードに最も近いユーザーは、安全なコーディングプラクティスを確立し、それらに従っていることを確認する必要があります。プロセスを継続的に反復して、パイプライン全体のセキュリティレベルに一貫性があることを確認します。最後に、ベストプラクティスとプロセスがバイパスされていないことを確認するテクノロジーを実装します。

お客様事例

AnyCompany Games は、ロールベースのアクセスコントロールを実装します。このコントロールでは、シニアデベロッパーのみがアンチチートシステムコードの変更を承認できますが、プレイヤーの支払いデータを処理するコンポーネントのセキュリティチームメンバーからの必須コードレビューが必要です。

CI/CD パイプラインは脅威モデルの検証チェックを自動的に実行し、プレイヤー取引市場などの新機能が、アイテムの重複の悪用や不正な取引の試みなど、以前に特定された攻撃ベクトルに対してテストされていることを確認します。

実装手順

- 最小特権の原則に基づいてユーザーにアクセス許可を付与します。
- を使用して AWS CloudTrail 、パイプラインで使用されるサービス全体で行われた API コールを監査します。
- 事前コミットフックを使用して、コードが一般的なプラクティスと会社ポリシーに従っていることを確認します。

セキュリティの自動化

GAMESEC09: CI/CD パイプライン内のセキュリティを自動化するにはどうすればよいですか?

CI/CD パイプライン内にセキュリティ対策を統合して、開発ライフサイクル全体で堅牢なセキュリティ体制を維持します。このプロセスには、パイプラインのセキュリティを確保することと同じ利点が多数あります。安全な CI/CD パイプラインを使用すると、ゲーム開発タイムラインが遅れる可能性のあるセキュリティイベントが発生する可能性が低くなります。

CI/CD パイプラインでセキュリティを提供するには、開発サイクルのすべての段階でセキュリティのベストプラクティスとツールを実装する必要があります。パイプラインにセキュリティを導入することで、セキュリティレビューの時間を短縮することもできます。

CI/CD パイプライン内のセキュリティを自動化することは、コードが変更されるたびにセキュリティコントロールが一貫して実装され、テストされていることを確認するために特に重要です。適切なツールと自動化を実装することで、安全で安全なゲームを提供できます。

ベストプラクティス

- GAMESEC09-BP01 ツールとオートメーションを統合してセキュリティレビューの平均時間を短縮する

GAMESEC09-BP01 ツールとオートメーションを統合してセキュリティレビューの平均時間を短縮する

セキュリティの脆弱性を特定するために、組織は静的アプリケーションセキュリティテスト (SAST) や動的アプリケーションセキュリティテスト (DAST) など、さまざまなツールやサービスを使用できます。SAST は、ソースコードを確認し、セキュリティの脆弱性を判断する方法です。DAST は、ソースコードを見ずにアプリケーションをテストするコードをテストするブラックボックス方法です。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

組織が使用できるもう 1 つのツールは、サードパーティーまたはオープンソースの依存関係のセキュリティを評価するソフトウェア構成分析 (SCA) です。より手動のアプローチでは、安全なコードレビューをパイプライン全体に実装できます。

お客様事例

AnyCompany Games は SAST ツールを使用して、開発プロセス中に潜在的なセキュリティ上の欠陥に自動的にフラグを付けます。また、DAST ツールを使用して、実行中のゲームビルドに対する脅威をシミュレートし、セキュリティコントロールが意図したとおりに機能していることを検証します。さらに、AnyCompany Games は、依存関係スキャンツールを開発プロセスに統合して、サードパーティーのライブラリやゲームエンジンの既知の脆弱性を自動的に特定します。

実装手順

- Amazon CodeGuru を SAST ツールとして使用します。
- OWASP 依存関係チェック、SonarQube、OWASPZap などのオープンソースツールを使用します。

リソース

- 開発者向けのセキュリティ

脅威モデリング

GAMESEC10: 脅威モデリングを組織のアプリケーション開発ライフサイクルにどのように統合しますか?

脅威モデリングは、アプリケーションに対する潜在的な脅威を特定して優先順位付けし、その軽減に使用できるソリューションを決定するプロセスです。このプラクティスは、ゲームが、機密性の高いユーザーデータとリアルマネートランザクションを処理する複雑な接続システムに進化するにつれて、ますます重要になっています。

ゲームのセキュリティをサポートするための継続的な演習として脅威モデリングを統合します。初期設計フェーズだけでなく、ゲームが成長し進化し続けるにつれて統合します。

ベストプラクティス

- GAMESEC10-BP01 アプリケーション開発ライフサイクル全体で脅威モデリングの演習をいつ、どのように完了するかを決定する

GAMESEC10-BP01 アプリケーション開発ライフサイクル全体で脅威モデリングの演習をいつ、どのように完了するかを決定する

脅威モデリングにアプローチする最適な方法は 1 つもありません。これを行うタイミングと方法の詳細は、ゲームスタジオの固有のニーズによって異なります。たとえば、スタジオのサイズによっては、脅威モデリングプロセスの 1 つ以上の側面に関与しているチームメンバーがいる場合があります。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

AWS セキュリティブログには、脅威モデリングの戦略を策定する際に留意すべき考慮事項の概要が記載されています。以下に例を示します。

- 脅威モデリングに関与すべきチームメンバーとペルソナ
- 使用する適切なワークフローツールを決定する方法
- 脅威モデリングのさまざまな側面の所有権を判断する方法
- ワークロード設計で使用するセキュリティコントロールを特定して評価する方法

お客様事例

AnyCompany Games は、プレイヤーデータ、ゲームコードやアルゴリズム、ゲーム内通貨、ユーザー生成コンテンツ、未公開コンテンツや独自のエンジンなどの知的財産などの貴重なアセットをカタログ化することから始まります。不正利益を求める不正行為者、個人データや財務データを盗もうとする不正行為者、ゲームプレイを中断しようとする悪意のあるユーザーなど、さまざまなタイプの潜在的な不正行為者を考慮します。

開発プロセス全体を通して、AnyCompany Games は脅威モデルを使用して安全なコーディングプラクティスをガイドし、テスト戦略に影響を与えてリスクの高い分野に焦点を当てます。ゲームのローンチ前に、包括的な脅威モデリングレビューを実施して、予想されるプレイヤーの負荷と不正アクセスの試みの準備状況を評価し、インシデント対応手順に備えます。

実装手順

- CI/CD パイプラインのすべての段階でガードレールを実装します。
- 自動化とツールを使用して、アプリケーションセキュリティレビューの効率を向上させます。
- 脅威モデリングをアプリケーションのセキュリティを向上させるプロセスとして使用します。

リソース

- AWS セキュリティブログ: 脅威モデリングにアプローチする方法
- NIST: データ中心のシステム脅威モデリングガイド
- ビルダーに適した脅威モデリング — AWS スキルビルダー仮想セルフペーストレーニング
- ビルダーの脅威モデリング – AWS ワークショップ

リソース

セキュリティに関連するベストプラクティスの詳細については、以下のリソースを参照してください。

関連ドキュメント:

- Amazon Cognito の一般的なシナリオ
- 署名URLs の使用
- チャネルフローを使用して、Amazon Chime SDK メッセージングのメッセージから冒涜的なコンテンツや機密コンテンツを削除する
- Amazon GameLift のセキュリティ
- Amazon CloudFront を使用したコンテンツ配信の保護
- セキュリティ対応ガイド
- AWS DDoS レジリエンシーのベストプラクティス
- ランサムウェアから AWS クラウド 環境を保護する

関連するパートナーソリューション:

- Datadog
- Sumo Logic

- Splunk
- Honeycomb.io
- New Relic
- AWS Marketplace - DRM ソリューション

関連するトレーニング資料:

- Amazon Cognito の開始方法
- セキュリティセルフペーストレーニング

信頼性

信頼性の柱には、インフラストラクチャやサービスの中断から回復し、需要を満たすために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システムの機能が含まれます。

焦点

- 設計原則
- 基礎
- ワークロードアーキテクチャ
- 変更管理
- 障害管理
- リソース

設計原則

AWS Well-Architected Framework ホワイトペーパーの設計原則に加えて、ゲームワークロードのクラウドの信頼性を高める設計原則を次に示します。

- ビジネスプロジェクションを満たすために必要なピークプレイヤーの同時実行数とシステムスケーラビリティターゲットのベースラインを確立する: ゲームを起動する前に、ライブゲームオペレーション中に、ピーク時に予想される同時プレイヤー数の見積もりを作成し、これらのプロジェクションを満たすためのシステムスケーラビリティのターゲット目標を確立します。これにより、ゲームの信頼性のベースラインを作成できます。スケーリングシステムがアクティブなプレイヤーセッションを適切に管理していることを検証することで、可用性に影響を与えることなく、需要の変化に自動的に対応できるようにスケーリングポリシーを定義します。
- 信頼性とプレイヤーエクスペリエンスへの影響を測定する: ゲームの状態を表す主要業績評価指標 (KPIs) を定義します。インフラストラクチャやゲーム機能の変更が信頼性に与える影響をモニタリングします。

基礎

信頼性を実現するには、システムには、需要や要件の変化を処理するメカニズムを備えた、十分に計画された基盤とモニタリングが整っている必要があります。システムは、障害を検出し、自動的に修復するように設計する必要があります。

Games Lens に固有の基盤のベストプラクティスはありません。ゲームワークロードに適用される信頼性の基礎のベストプラクティスについては、「Well-Architected Framework」ホワイトペーパーを参照してください。

ワークロードアーキテクチャ

GAMEREL01: ゲームアーキテクチャはクラウドの耐障害性を活用していますか?

AWS インフラストラクチャは、リージョンとアベイラビリティーゾーンを中心に構築されています。 は、低レイテンシー、高スループット、高度に冗長なネットワークを使用して接続される、物理的に分離および分離された複数のアベイラビリティーゾーン AWS リージョン を提供します。これらのコンストラクトを使用して、信頼性目標に焦点を当てたワークロードを設計できます。

ベストプラクティス

- GAMEREL01-BP01 ゲームインフラストラクチャを複数のアベイラビリティーゾーンとリージョンに分散して回復性を向上させる

GAMEREL01-BP01 ゲームインフラストラクチャを複数のアベイラビリティーゾーンとリージョンに分散して回復性を向上させる

ローカライズされたインフラストラクチャの障害がプレイヤーに与える影響を最小限に抑えるには、プレイヤーの需要を満たすのに十分な容量を確保しながら、予期しない障害に耐えられるように、インフラストラクチャのデプロイを十分な独立した場所に均一に分散する必要があります。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

ゲームインフラストラクチャをデプロイするときは、プレイヤーエクスペリエンスを中断することなく 1 つ以上のアベイラビリティーゾーンの中断に耐えられるように、リージョン内の複数のアベ

イラビリティーゾーンに容量を均一に分散することをお勧めします。ウェブアプリケーションなどのゲームバックエンドサービスは、複数のアベイラビリティーゾーン間で負荷分散するか、 AWS Lambda や Amazon API Gateway などのマネージドサービスを使用して構築する必要があります。このサービスは、リージョンの高可用性を設計で提供します。同様に、キャッシュ、データベース、メッセージキュー、ストレージソリューションなどの状態を維持するコンポーネントは、Amazon S3、DynamoDB、Amazon SQS などのサービスで設計によって提供され、他のサービスで設定できる複数のアベイラビリティーゾーンにわたってデータの永続的な永続性を提供するように設計する必要があります。

回復性のためにゲームサーバーホスティングアーキテクチャを設計する場合は、ゲームサーバーのフリートを 内のアベイラビリティーゾーン全体に均一にデプロイ AWS リージョン して、リージョン内の利用可能なコンピューティングキャパシティーへのアクセスを最大化し、アベイラビリティーゾーンの障害の影響範囲を減らします。例えば、アベイラビリティーゾーンを使用するように Amazon EC2 Auto Scaling を設定できます。EC2 インスタンスが異常になった場合、EC2 Auto Scaling はインスタンスを置き換えたり、1 つ以上のアベイラビリティーゾーンが使用できなくなった場合に他のアベイラビリティーゾーンでインスタンスを起動したりできます。

認証などの重要なインフラストラクチャでは、複数のアベイラビリティーゾーンで実行されている有効なインスタンスの最小数をプロビジョニングし、いずれかのアベイラビリティーゾーンに障害が発生した場合は、自動スケーリングを使用して負荷の増加や耐障害性を処理します。

ゲームインフラストラクチャを複数のリージョンにデプロイして、可用性を最大化します。Aurora グローバルデータベースや冗長インフラストラクチャなどのクロスリージョンディザスタリカバリ機能は、セカンダリリージョンにデプロイされた単純な DNS 変更でアクティブになり、プライマリリージョンに障害が発生した場合にサービスの継続性を提供することができます。これは、高可用性を実現するためにゲームバックエンドサービスにお勧めしますが、この推奨事項はゲームサーバーにとって特に重要です。

例えば、マルチプレイヤーゲームでは、ゲームサーバーがプレイヤーのゲームセッションをホストするために使用されるため、ゲームサーバーのインフラストラクチャ容量が他のサービスの容量ニーズを上回る可能性があります。多くのゲームは、プレイヤーを論理ゲームリージョン (米国西部や東部など) にシャードすることを選択します。プレイヤーエクスペリエンスを簡素化し、グローバルインフラストラクチャを使用してゲームをホストすることを容易にするために、ゲームサーバーを物理的にホストしている基盤となるクラウドプロバイダーリージョンまたはデータセンターの場所から、プレイヤー向けゲームリージョンの名前を、そのプレイヤーゲームリージョンをサポートするゲームサーバーインスタンスをホストしているローカルゾーンまたは独自のデータセンターなどの他のインフラストラクチャと切り離すことを検討してください。

マッチメーキングサービスを設計するときは、リージョン間で個別のソフトウェアデプロイを使用してマルチリージョンアーキテクチャをデプロイします。マッチメーキングサービスのデプロイをゲームサーバーインスタンスをホストするフリートから切り離して、マッチメーキングサービスのどのリージョンデプロイがマッチメーキングリクエストを処理したかに関係なく、リージョンのゲームサーバーにプレイヤーをルーティングできるようにします。

マッチメーキング実装でロジックを設計して、レイテンシーやその他のルールを満たすゲームサーバーリージョンを優先します。フリートの容量が少ない場合や、他のリージョンのインフラストラクチャの中断が発生した場合は、プレイヤーを他のリージョンにルーティングできます。

実装手順

- ゲームインフラストラクチャを複数のアベイラビリティーゾーンに均等に分散して、高可用性と耐障害性を実現します。
- Amazon S3 AWS Lambda、DynamoDB、SQS などのマネージド型 を使用してゲームバックエンドサービスとステートフルコンポーネントをデプロイするか、カスタムソリューションの負荷分散と耐久性を設定します。 Amazon S3
- 基になる物理的な場所から切り離された Aurora グローバルデータベースや論理プレイヤー向けリージョンなどのディザスタリカバリソリューションを使用して、重要なゲームサービスとサーバーにマルチリージョンデプロイを実装します。

リソース

- 静的安定性
- Amazon GameLift ゲームセッションキューのベストプラクティス
- Amazon GameLift マルチリージョンフリート
- Aurora Global Database for Disaster Recovery

変更管理

GAMEREL02: 需要の変化に対応するためにステートフルゲームをスケールする方法を教えてください。

プレイヤーの需要が時間の経過とともに変動するにつれて、ゲームインフラストラクチャはこれらの変化する要件に対応するために適応的にスケールできる必要があります。ゲームの人気を事前に予測することは困難ですが、プレイヤー人口の変動に対応するためにインフラストラクチャ容量の追加と削除を可能にするアーキテクチャアプローチを設計してください。

ベストプラクティス

- GAMEREL02-BP01 アクティブなプレイヤーゲームセッションの状態を組み込んだスケーリング戦略を実装する
- GAMEREL02-BP02 ゲームでの複数の EC2 インスタンスタイプの使用をサポート

GAMEREL02-BP01 アクティブなプレイヤーゲームセッションの状態を組み込んだスケーリング戦略を実装する

アクティブに接続されたプレイヤーセッションのステートフルな性質を組み込み、ゲームプレイを中断することなくスケーリングアクティビティを適切に処理する方法で、ゲームインフラストラクチャを自動的にスケーリングするソリューションを実装します。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

クラウドでゲームを開発する利点の 1 つは、需要を満たすために必要に応じてサーバーインフラストラクチャを自動的にスケーリングすることで達成できる伸縮性です。ステートレスまたは非同期のゲームとバックエンドサービスは、Amazon EC2 Auto Scaling ポリシー、EKS 自動スケーリング、またはスケーラブルなウェブアプリケーションに一般的に採用されている同様の手法を使用して動的にスケーリングできますが、ゲーム開発者は通常、アクティブなプレイヤーセッションの中断をブロックするために、ステートフルまたは同期のゲームをスケーリングするためのよりカスタマイズされたアプローチを必要とします。

実装手順

- ステートフルゲームの場合、プレイヤーセッションの状態と利用可能なゲームサーバー容量をモニタリングするために使用できるカスタムメトリクスを生成し、カスタムメトリクスとして Amazon CloudWatch に報告できます。CloudWatch Synthetics などのアプリケーションモニタリングで機能を演習し、ゲームでヘルスモニタリングが単純に検出できない機能の障害がないか確認します。
- カスタムメトリクスを使用して、ゲームサーバースケーリングソフトウェアを実装します。たとえば、 AWS Lambda 関数を使用するサーバーレスアプリケーションとして AWS Fargate、または AWS SDK を使用して API コールを行い、ゲームサーバービルドをホストする EC2 Auto Scaling

グループの最小、最大、および希望する容量設定を更新することで、専用のゲームサーバーインスタンスのフリートを管理します。

- Amazon GameLift を使用してゲームサーバーをホストし、out-of-the-boxゲームサーバーの自動スケーリング機能を使用して、このスケーリングプロセスを管理します。

Amazon GameLift の自動スケーリング機能はアクティブなプレイヤーセッションを認識し、プレイヤーをアクティブにホストしているゲームサーバーインスタンスの終了またはスケールインをブロックするように設定できます。詳細については、「Amazon CloudWatch で Amazon GameLift サーバーをモニタリングする」を参照してください。

GAMEREL02-BP02 ゲームでの複数の EC2 インスタンスタイプの使用をサポート

EC2 インスタンスを使用してゲームをホストする場合、または の EC2 インスタンスでホストされているコンテナを使用する場合は AWS アカウント、ホスティング戦略で複数のインスタンスタイプを使用します。複数のインスタンスタイプを使用すると、ゲームのスケーリング時に使用できるコンピューティングオプションの数を増やして、プレイヤーの増加をサポートするサーバーを追加できるため、希望するインスタンスタイプが使用できない場合の信頼性が向上します。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

スポットインスタンスを使用してゲームをホストする場合は、お客様の需要に応じてスポットインスタンスの可用性が変動するため、複数のインスタンスタイプを使用します。

コストとパフォーマンスの要件を満たすために複数のインスタンスタイプでゲームをテストし、インスタンスタイプの優先順位ランキングを決定します。Amazon EC2 Auto Scaling は、複数のインスタンスタイプとサイズの使用と、コンピュートオプションの優先順位付けを実装できるように設定内の各インスタンスタイプへの重みの割り当てをサポートしています。

Amazon GameLift マネージドホスティングを使用してゲームをホストする場合は、ゲームに必要なインスタンスのタイプと、ゲームサーバープロセスの実行方法 (ランタイム設定を使用) を決定します。フリートのリソースを選択するときは、ゲームオペレーティングシステム、インスタンスタイプ (コンピューティングハードウェア) 、オンデマンドインスタンス、スポットインスタンス、またはその両方を使用するかどうかなど、いくつかの要因を考慮してください。Amazon GameLift のホスティングコストは、主に使用するインスタンスのタイプによって異なります。詳細については、「マネージドフリートのコンピューティングリソースを選択する」を参照してください。

実装手順

- 複数の EC2 インスタンスタイプを使用して、EC2 またはコンテナでゲームをホストする際の信頼性とスケーリングオプションを向上させます。
- コストとパフォーマンスを最適化するために、優先順位付けされたインスタンスタイプと重みで Amazon EC2 Auto Scaling フリートまたは GameLift フリートを設定します。
- さまざまなインスタンスタイプでゲームをテストして、パフォーマンスが要件を満たしていることを確認し、それに応じてホスティング戦略を調整します。

障害管理

GAMEREL03: インフラストラクチャの中断時にゲームの状態を維持するにはどうすればよいですか?

ゲームインフラストラクチャが時間の経過とともにさまざまな運用イベントを経験するにつれて、ゲームのアーキテクチャは、プレイヤーエクスペリエンスの継続性を維持し、インフラストラクチャイベント中にゲームの状態を維持するように設計されている必要があります。これらのイベントを処理するには、モニタリング、正常なシャットダウン、状態永続化メカニズムを実装して、プレイヤーのスムーズなゲームプレイエクスペリエンスを検証します。

ベストプラクティス

- GAMEREL03-BP01 ゲームサーバーの中断を監視し、データを使用してホスティングアーキテクチャを改善し、信頼性目標を達成する
- GAMEREL03-BP02 ゲーム機能の疎結合を実装して、プレイヤーエクスペリエンスへの影響を最小限に抑えながら障害を処理する
- GAMEREL03-BP03 インフラストラクチャイベントを経時的にモニタリングして、プレイヤーの動作への影響を測定する

GAMEREL03-BP01 ゲームサーバーの中断を監視し、データを使用してホスティングアーキテクチャを改善し、信頼性目標を達成する

ゲームサーバーメトリクスと、ロード時のレイテンシーの増加などのパフォーマンスの障害や低下がプレイヤーの動作に与える影響を経時的にモニタリングし、ゲームの信頼性要件を満たすようにゲー

ムサーバーホスティング戦略を調整できるようにします。パフォーマンスが低下するゲームサーバーインフラストラクチャは、プレイヤーに影響を与えている場合は直ちにサービスから削除するか、サーバーでホストされているアクティブなプレイヤーセッションがない場合にプロアクティブに交換する必要があります。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

ゲームが REST APIs としてホストされているシナリオでは、システムの信頼性を従来のウェブアプリケーションアーキテクチャのように管理できます。このアーキテクチャでは、サーバー障害のリスクを軽減するために、トラフィックを複数のサーバー間で分散的に負荷分散できます。

リアルタイムの同期ゲームプレイの場合、ゲームセッションは通常、仮想マシンまたはゲームサーバーインスタンスで実行されているゲームサーバープロセスでホストされます。これは、ゲームプレイ状態をパフォーマンスの高い方法で維持し、接続されたゲームクライアントにレプリケートする必要があるためです。この実装は、プレイヤーのエクスペリエンスが、ゲームセッションをホストするゲームサーバープロセスのパフォーマンスと信頼性に密接に結合されていることを意味します。このタイプのアーキテクチャにより、ゲームサーバーの信頼性の管理は従来のアプローチよりも複雑になります。

ゲームサーバーの障害の影響を軽減するには、Amazon Amazon ElastiCache (Redis OSS) や Amazon MemoryDB などの高可用性キャッシュまたはデータベースに対してプレイヤーのゲーム状態の非同期更新を継続的に実行するようにゲームを設定します。サーバーに障害が発生した場合、プレイヤーの最後に保存されたゲーム状態を外部データストアから取得し、そのセッションを新しいゲームサーバーインスタンスに復元できます。

ただし、このアプローチは、この外部状態を管理するためのコストと複雑さを追加し、状態の変化が非常に頻繁で、パフォーマンスの高いインメモリキャッシュデータストアを導入してもセッションの復元にはあまりにも大きなレプリケーションラグが発生するような大規模なペースの速いゲームや競合ゲームには適していない可能性があります。この種のゲームの場合、最適な方法は、サーバーが失われたのを受け入れてプレイヤーをゲームロビーに送り返して別のセッションを見つけることです。そうしないと、自動的に別のゲームセッションにリダイレクトできます。

サーバーの中断の原因に関する有用なログデータをキャプチャして、後で問題を調査できるようにします。Amazon GameLift は、フリートの問題をデバッグするためのガイダンスを提供し、Amazon GameLift フリートインスタンスにリモートアクセスする機能を提供します。

実装手順

- ゲームサーバーメトリクスのパフォーマンス低下をモニタリングし、必要に応じてパフォーマンスが低下しているサーバーを削除または置き換えて信頼性を維持します。
- Amazon ElastiCache または MemoryDB を非同期ゲーム状態の更新に使用して、可能であればサーバー障害後のセッション復旧を有効にします。
- フリートのモニタリングとリモートアクセスに Amazon GameLift などのツールを活用して、調査とデバッグのためにサーバーの中断に関する詳細なログデータをキャプチャします。

GAMEREL03-BP02 ゲーム機能の疎結合を実装して、プレイヤーエクスペリエンスへの影響を最小限に抑えながら障害を処理する

コンポーネントのデカップリングとは、サーバーコンポーネントを可能な限り独立して動作できるように設計するという概念を指します。ゲームの一部の側面は、プレイヤーに良いゲーム内エクスペリエンスを提供するために、データをできるだけ最新の状態にする必要があるため、切り離すのは困難です。ただし、多くのコンポーネントとゲームタスクは切り離すことができます。例えば、リーダーボードと統計サービスはゲームプレイエクスペリエンスにとって重要ではなく、これらのサービスの読み取りと書き込みはゲームから非同期的に実行できます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

問題が検出された場合は、自動または管理者が無効にできるゲームの機能の正常な低下を実装し、障害を適切に処理できるように機能に依存するアップストリームサービスを設定します。たとえば、特定のプレイヤーデータがゲームクライアント内で適切にロードされない場合は、このデータがゲームプレイエクスペリエンスにとって重要であるかどうかを検討する必要があります。そうでない場合は、プレイヤーのエクスペリエンスを中断せずにこの障害を適切に処理するようにゲームクライアントを設定します。プレイヤーが画面を再確認したときに、後でこのデータの取得を再試行することもできます。

タイムアウト、再試行、バックオフなどのロジックを使用して、エラーや障害を処理します。タイムアウトにより、システムが過度に長時間ハングしなくなります。再試行により、一時的なエラーとランダムなエラーの高可用性を実現できます。

重要なコンポーネントに疎結合できる重要でないコンポーネントを定義します。疎結合を使用すると、あるコンポーネントの障害が他のコンポーネントにカスケードされないため、システムの耐障

害性が向上します。ゲーム機能にゲームサーバーまたはバックエンドへのステートフル接続が必要ない場合は、ステートレスプロトコルを実装して動的にスケーリングし、一時的な障害から回復する必要があります。HTTP/JSON API を使用して、ステートレスプロトコルと疎結合できる重要でないコンポーネントを開発します。ゲームクライアントからのネットワーク呼び出しを非同期およびノンブロッキングに実装して、パフォーマンスの低いゲーム機能やその他の依存サービスによるプレイヤーへの影響を最小限に抑えます。

疎結合によって回復性をさらに向上させるには、キューイング、ストリーミング、トピックベースのシステムなどのメッセージングサービスを、非同期で処理できるコンポーネント間で使用します。このモデルは、即時の応答を必要としないインタラクションや、リクエストが登録されたことの確認で十分であるインタラクションに適しています。このソリューションには、イベントを生成する 1 つのコンポーネントと、イベントを消費する別のコンポーネントが含まれます。2 つのコンポーネントは、point-to-pointの直接的なやり取りではなく、耐久性のあるストレージやキューイングレイヤーなどの中間コンポーネントを通じて統合されます。これにより、処理が失敗したときにメッセージを保存することで、システムの信頼性を向上させることもできます。

注文や配信メカニズムなど、さまざまなメッセージングサービスの特性が異なるため、適切なメッセージングメカニズムを調査して選択します。選択したメッセージシステムが少なくとも 1 回メッセージを配信するように、べき等なオペレーションを設計します。例えば、ゲームがプレイヤーのプレイタイム、統計、またはその他の関連データを追跡する必要がある一般的なゲームユースケースを考えてみましょう。これにより、プレイヤーの同時実行のピーク時に高い書き込みスループットのユースケースが発生する可能性があります。

信頼性の高いアーキテクチャを実装するには、プレイヤーが認識するread-after-write整合性がユースケースで必要かどうかを検討します。通常、このようなシナリオは非同期処理に適しており、リクエストが Amazon SQS などのスケーラブルで耐久性の高いメッセージキューに取り込まれ、Lambda 関数などのコンシューマーサービスを使用してバックエンドデータベースにバッチで挿入できる書き込みキューイングパターンを実装することで実現できます。このアプローチは、プレイヤーのゲームクライアント、バックエンドのウェブサーバーとアプリケーションサーバー、内部データベースシステムなど、複数の分散コンポーネント間の同期通信よりも信頼性が高くなります。また、書き込みキューからのコンシューマー処理を使用して必要に応じてこの取り込み速度を遅くできるため、バックエンドデータベースをピーク書き込みスループットに合わせてスケーリングする必要がないため、コストを削減できます。

実装手順

- リーダーボードや統計サービスなどの重要でないコンポーネントを重要なゲームプレイ機能から切り離して、非同期オペレーションを可能にし、回復性を強化します。

- タイムアウト、再試行、バックオフのロジックを使用して、重要ではない機能の正常なパフォーマンス低下を実装し、ゲームクライアントがプレイヤーエクスペリエンスを中断することなく障害を処理することを確認します。
- Amazon SQS などのメッセージングシステムを使用してコンポーネント間の非同期通信を行い、高スループットのユースケースをスケーラブル、耐久性、信頼性の高い方法で処理できます。

リソース

- マイクロサービスアーキテクチャを使用して、スケーラビリティと信頼性に優れたワークロードを構築する
- AWS サーバーレスサービスを使用したマイクロサービスの統合
- マイクロサービスの非同期メッセージングについて
- でのスケーラブルなゲーム開発パターンの概要 AWS
- グレースフルデグレードの実装

GAMEREL03-BP03 インフラストラクチャイベントを経時的にモニタリングして、プレイヤーの動作への影響を測定する

ゲームサーバープロセス、ゲームサーバーインスタンスメトリクス、ゲームエクスペリエンスメトリクスをモニタリングして、問題の根本原因を特定します。CPU とメモリのモニタリングに加えて、EC2 インスタンスのネットワーク制限に関連するネットワークメトリクスのモニタリングを設定して、帯域幅の超過、1 packets-per-second数、サーバーリソースのプロビジョニング不足を示す可能性のあるその他のネットワークレベルの問題などの問題を警告することもできます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

CloudWatch Synthetics を使用して、ログインできない、または他のサービスが問題に影響を与えるなど、プレイヤーエクスペリエンスの重要なパスアプリケーション機能を確認します。Amazon GameLift を使用してホストされているゲームサーバーの場合は、次のようなメトリクスのモニタリングを検討してください。

- GameServerInterruptions と InstanceInterruptions は、スポットインスタンスの可用性の制限が、スポットを使用してデプロイされたゲームサーバーにどのように影響しているかを理解するのに役立ちます。

- ServerProcessAbnormalTerminations。ゲームサーバープロセスの異常な終了を検出するために使用できます。

ゲームサーバーの信頼性の履歴メトリクスデータを維持することをお勧めします。この履歴データをレポート目的に使用し、他のデータセットと結合して潜在的な傾向を明らかにし、ゲームサーバーの問題に起因する可能性のあるプレイヤーの経時的な行動への影響を評価します。

Amazon CloudWatch はメトリクスを無期限に保持せず、メトリクスのストレージ解像度は時間の経過とともに向上するため、これらのメトリクスを Amazon S3 などの費用対効果の高い長期ストレージにエクスポートすることを検討してください。メトリクスを CloudWatch リージョンから独自の S3 バケットに自動的に配信するように CloudWatch Metric Streams を設定できます。このバケットは、S3 Intelligent-Tiering などのストレージ階層に長期保存し、最終的に Amazon Glacier を使用してアーカイブできます。 CloudWatch S3 メトリクスを Amazon S3 に配置することで、Amazon Athena でのインタラクティブなクエリのために、データレイク内の他のデータセットと簡単に結合できます。

実装手順

- クリティカルパス機能チェックに Amazon CloudWatch と CloudWatch Synthetics を使用して、帯域幅やpacket-per-secondの制限などのゲームサーバー、インスタンス、ネットワークメトリクスをモニタリングします。
- GameServerInterruptions や ServerProcessAbnormalTerminations などの GameLift 固有のメトリクスを追跡して、スポットインスタンスの可用性の影響を評価し、異常なサーバー終了を検出します。 ServerProcessAbnormalTerminations
- CloudWatch メトリクスを Amazon S3 にエクスポートして長期保存し、S3 Intelligent-Tiering や Glacier などの費用対効果の高い階層を使用し、Amazon Athena などのツールを使用して傾向を分析します。

リソース

- Amazon EC2 インスタンスレベルのネットワークパフォーマンスメトリクスが新しいインサイトを明らかにする
- CloudWatch メトリクスストリーム – パートナーとアプリにリアルタイムで AWS メトリクスを送信する

リソース

信頼性に関連するベストプラクティスの詳細については、以下のリソースを参照してください。

関連ドキュメント:

- での継続的インテグレーションと継続的デリバリーの実践 AWS
- 非同期ジョブキューの自動スケーリング
- WorkloadService アーキテクチャの設計
- ジッターによるタイムアウト、再試行、バックオフ
- Well-Architected フレームワーク - 信頼性の柱
- 信頼性の高いスケーラビリティのためのアーキテクチャ
- Amazon Builder のライブラリ
- マルチプレイヤーゲームの大規模スケールリアルタイムメッセージング
- でのスケーラブルなゲーム開発パターンの概要 AWS
- でコンテナ化されたマイクロサービスを実行する AWS
- クラウドでのウェブアプリケーションのホスティング
- スケーラブルで安全なマルチ VPC ネットワークインフラストラクチャの構築

関連動画:

- re:Invent 2020: Ubisoft: でのマルチプラットフォームマルチプレイヤーゲームの構築 AWS
- re:Invent 2018: Supercell- Scaling Mobile Games
- re:Invent 2019: CAPCOM がコンテナ、データ、ML を使用して楽しいゲームを構築する方法
- re:Invent 2018: 可用性を維持しながらリオットゲームでプレイヤーアカウントをグローバル化する
- re:Invent 2020: GameLoft - ダウンタイムゼロのデータレイク移行の詳細

関連するトレーニング:

- ゲームサーバーでの Amazon GameLiftFleetIQ の使用
- Amazon EC2 によるゲームサーバーのホスティング

パフォーマンス効率

パフォーマンス効率の柱は、要件を満たすためのコンピューティングリソースの効率的な使用と、需要の変化やテクノロジーの進化に応じてその効率を維持することに焦点を当てています。

データ駆動型のアプローチで、高性能アーキテクチャを選択します。高レベルの設計からリソースタイプの選択と設定まで、アーキテクチャに関する包括的なデータを収集します。進化し続けるサービスやソリューションを活用するために、アーキテクチャの選択を周期的に検討してください。メトリクスは、期待されるパフォーマンスからの逸脱を理解するのに役立つため、アクションを実行できます。データ駆動型アプローチは、パフォーマンスの向上、コストの削減、デベロッパーエクスペリエンスの向上のために、アーキテクチャとのトレードオフに役立ちます。

焦点

- 設計原則
- アーキテクチャの選択
- リージョンの選択
- 反復開発
- コンピューティングとハードウェア
- コンピューティングの選択
- データ管理
- ネットワークとコンテンツ配信
- プロセスと文化
- リソース

設計原則

AWS Well-Architected Framework ホワイトペーパーの設計原則に加えて、以下の設計原則はゲームのパフォーマンス効率を達成できます。

- ゲームのパフォーマンスをend-to-endで測定する: プレイヤーの視点から認識されるため、パフォーマンスを測定することが重要です。つまり、ゲームクライアント、ゲームインフラストラクチャ、およびプレイヤーをインフラストラクチャに接続するインターネット接続のパフォーマンスを測定する必要があります。これにより、アーキテクチャ内でパフォーマンスを向上させることができる場所を把握できます。

- アーキテクチャを最適化して実際のプレイヤーエクスペリエンスを反映するメトリクスを改善する: 時間の経過とともにアーキテクチャを適応および進化させる際には、これらの改善と変更がプレイヤーエクスペリエンスにどのように影響するかを検討してください。ゲームワークロードは、ゲームプレイへの広範な中断をブロックする障害の影響に耐え、最小限に抑えることができる必要があります。相互に重要に依存しないゲーム機能とシステムは、障害の爆発半径を減らし、プレイヤーに影響を与える問題を特定するために、切り離す必要があります。
- ゲームオペレーションを簡素化し、開発速度を向上させるテクノロジーを使用する: 開発者の効率を向上させるテクノロジーの採用を優先します。開発の本番稼働前の段階での運用上のオーバーヘッドは、ゲームプレイの改善を妨げる可能性があります。 AWS または AWS パートナーのマネージドサービスを活用することで、エンジニアリングを減らすことができます。これにより、ゲーム開発者はコアゲームループとプレイヤーエクスペリエンスに集中できます。アーキテクチャとパフォーマンスの要件はゲーム開発ライフサイクル全体で変化し、進化する可能性があります。テクノロジーのトレードオフは各フェーズで検討する必要があります。
- ピークプレイヤーの同時実行数に合わせてインフラストラクチャを設計し、必要に応じて動的にスケールする: インフラストラクチャはプレイヤーの需要に合わせてスケールするように設計する必要があります。プレイヤーセッションの同時実行数やログイン数などのメトリクスを使用して、システムが過負荷になる前に事前にスケーリングできます。CPU やメモリの消費量などのリアクティブシステム使用率メトリクスを使用して、システムが過負荷になった後にスケールできます。インフラストラクチャを動的にスケーリングすることで、ゲームの運用コストを削減できます。

アーキテクチャの選択

GAMEPERF01: ゲームサーバーに適したホスティングオプションはどのように選択しますか?

ゲームサーバーに適したホスティングオプションの選択は、ゲームサーバーのパフォーマンスの基礎となります。EC2 インスタンス、コンテナソリューション、またはフルマネージドサービスの使用を決定することは、本番稼働用に設計する際に行う最初の決定の 1 つです。各ホスティングオプションには、パフォーマンスのチューニング、スケーリング、オペレーション、統合に関するさまざまな機能と考慮事項があります。

ベストプラクティス

- GAMEPERF01-BP01 ゲームサーバーのリソース要件とスケーラビリティのニーズを評価する
- GAMEPERF01-BP02 ゲームサーバーをスケーリングするための運用オーバーヘッドを検討する

- GAMEPERF01-BP03 他の AWS サービス、開発環境、ターゲット CPU アーキテクチャ、および機能との統合を評価する

GAMEPERF01-BP01 ゲームサーバーのリソース要件とスケーラビリティのニーズを評価する

スケーラビリティのニーズに照らしてサーバー要件を評価し、要件を満たし、最適なパフォーマンスを提供するホスティングオプションを選択していることを確認します。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

ゲームサーバーに適したホスティングオプションを選択するときは、次の要素を考慮してください。

ゲームサーバーリソースの要件

ゲームサーバープロセスの CPU、メモリ、ネットワーク、ストレージの要件を評価して、ゲームが消費する内容を決定します。ネットワークを見落としないでください。各フレームでは、プレイヤーアクションを受信し、ゲームの状態を更新してプレイヤーに送り返すために CPU サイクルが必要です。パケット処理をオフロードすると、コアゲーム機能の CPU を解放できます。ネットワークはスムーズで応答性の高いゲームプレイの基盤であるため、プロセスの早い段階でテストすると、ゲームのベースラインパフォーマンスプロファイルが定義されます。

ファーストパーソンシューティングゲームは 1 秒あたりに高いアクションがあり、CPU はコンピューティング最適化 C ファミリーインスタンスを優先するネットワークにすばやく移行する必要があります。一方、1 ターンあたりにより多くの CPU サイクル処理を消費できるターンベースの戦略ゲームでは、R ファミリーインスタンスのメモリを増やして、プレイヤーに送り返す前にサーバーにゲームの状態をローカルに保存および更新する必要があります。使用率飽和とエラー (USE) メソッドなどのデータ駆動型アプローチを使用して、十分な情報に基づいたアーキテクチャの選択を行います。

スケーラビリティと伸縮性

パフォーマンスを損なうことなく、各ホスティングオプションがプレイヤーの需要に合わせてどれだけ迅速かつスムーズにスケーリングできるかを評価します。ピーク時にスムーズなゲームエクスペリエンスを維持するために、ゲームのワークロードに必要な自動化と柔軟性のレベルを検討してください。ゲームサーバーは、同じインスタンスにゲームサーバープロセスを追加することで使用率を高めることで迅速にスケールできます。この場合、ゲームバックエンドは、アクティブなユーザー数の増加とプレイ中のゲームに基づいてスケールが遅くなる可能性があります。プレイヤーがゲームに参加

するための待機時間を最小限に抑えながら、コストを最小限に抑えるために、フリートは需要に応じてスケールする必要があります。Amazon EC2 スポットインスタンスアドバイザーを確認して、ゲームサーバーフリートのコスト効率の高い利用可能な容量に関するインサイトを取得します。

実装手順

- FPS ゲームの高ネットワークスループットやターンベースの戦略ゲームのメモリ最適化など、ゲーム固有のパフォーマンスニーズを考慮して、CPU、メモリ、ネットワーク、ストレージのゲームサーバーリソース要件を評価して、適切なインスタンスタイプを選択します。
- USE メソッドなどのフレームワークを使用してパフォーマンスデータを分析することで、コンテナ、インスタンス、ベアメタル、マネージドサービスなどのさまざまなホスティングオプションを比較します。これらのインサイトを使用して、システムアーキテクチャに関するより良い意思決定を行います。
- スケーラビリティと伸縮性のためにフリートを設計し、EC2 スポットインスタンスアドバイザーなどのツールを活用してコストを最適化しながら、ピーク時にプレイヤーの需要を満たすための迅速なスケーリングを促進します。

GAMEPERF01-BP02 ゲームサーバーをスケーリングするための運用オーバーヘッドを検討する

各ホスティングオプションに関連する管理オーバーヘッドと運用オーバーヘッドを考慮してください。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

運用オーバーヘッド

EC2 またはコンテナ上のセルフホスト型ソリューションは、より多くの制御を提供できますが、より多くの管理も必要になります。ECS や EKS などのコンテナオーケストレーターは、ネットワークの複雑さとメンテナンスオーケストレーションのオーバーヘッドを増やしながら、コンテナ化されたサーバーの起動時間を短縮できます。

例えば、EKS マネージドノードグループはゲームサーバーのプロビジョニングとライフサイクル管理を自動化できますが、ノードの終了時にポッドの中断予算は考慮しません。ゲームを安全に完了するために 15 分の終了期間よりも長いゲームが必要な場合は、ライフサイクルフックを作成するか、カスタムコントローラーでセルフマネージドノードを検討してゲームの中断をブロックする必要があります。

Amazon Game Lift などのマネージドサービスは、運用上のオーバーヘッドのほとんどを処理する可能性がありますが、低レベルのネットワークとセキュリティ設定の特別な要件に対する可視性と制御の量を減らすことができます。ゲームサーバーソリューションの選択は、ゲームサーバーのパフォーマンスとスケーリング動作を調整するためのカスタマイズ、制御、責任のレベル間のトレードオフです。

実装手順

- ホスティングオプションの運用オーバーヘッドを評価し、EC2、ECS、EKS などのセルフホスト型ソリューションと Amazon Game Lift などのマネージドサービスの間で制御と管理の労力のバランスを取ります。
- 自動化には EKS マネージド型ノードグループを使用しますが、ゲームサーバーでデフォルトよりも長い終了期間が必要な場合は、ライフサイクルフックまたはカスタムコントローラーを実装します。
- ゲームサーバーソリューションを選択するときは、カスタマイズ、可視性、運用責任のトレードオフを考慮します。

GAMEPERF01-BP03 他の AWS サービス、開発環境、ターゲット CPU アーキテクチャ、および機能との統合を評価する

各ホスティングオプションが、データベース、分析、コンテンツ配信サービスなど、ゲームが依存する他の AWS サービスとどの程度統合されているかを評価します。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

他の AWS サービスとの統合

サービス間のシームレスな統合は、パフォーマンスモニタリングの向上や、ゲームコンポーネント、ゲームサーバー、ゲームバックエンドサービス、オブザーバビリティソリューション間の効率的な安全なデータ配信などの運用上の利点を提供します。

たとえば、ライブゲームのトラフィックシフトの調整は複雑になる場合があります。Amazon Route 53 は DNS レコードを最新の状態に保つため、調整されたトラフィックシフトが簡素化されます。AWS Global Accelerator トラフィックダイヤルを使用すると、トラフィックの一部を別のリージョンに送信し、メンテナンス中にゲームを実行し続けることができます。

開発環境とツール

各アーキテクチャオプションでサポートされている開発ツール、フレームワーク、環境を検討してください。選択したオプションがゲーム開発ソリューションとプログラミング言語と一致していることを確認します。これは、ゲームサーバーのパフォーマンスを最適化および維持するチームの能力に影響を与える可能性があるためです。モバイル、コンソール、PC 間でゲームを配信すると、ツールとテストの複雑さが増します。クロスシステムサポートは、一元化されたサービスがタイトル全体で開発のベストプラクティスを標準化できるマルチゲームスタジオにとって特に重要です。

ターゲット CPU アーキテクチャと機能

ゲームエンジンとゲームサーバープロセスのパフォーマンスプロファイルと利用可能な ARM サポートのレベルを考慮してください。ARM ベースの Graviton プロセッサまたは x86 ベースの AMD64 プロセッサの価格パフォーマンスの向上からメリットが得られるかどうかを評価します。AES-NI 暗号化、AVX、Turbo Boost などの Intel 機能を使用する必要がありますか? Dedicated Host タイプを確認して、シングルソケットインスタンスファミリーとマルチソケットインスタンスファミリーを識別します。マルチソケットインスタンスファミリーを使用する場合は、ゲームサーバープロセスで NUMA ピン留めと L3 キャッシュ共有を検討してください。C ステートと P ステートの設定を使用して、周波数クロックを調整し、スリープレベルを減らすことで、ゲームに最適なパフォーマンスを実現します。

実装手順

- ACM などのサービスとシームレスに統合 AWS できるホスティングオプションを選択すると AWS Secrets Manager、パフォーマンスモニタリングの合理化、データ配信の保護、手動による運用タスクの削減に役立ちます。
- ホスティングオプションと開発環境、フレームワーク、プログラミング言語との互換性を検証して、サーバーのパフォーマンスを効果的に最適化および維持します。
- CPU アーキテクチャの要件を評価し、料金パフォーマンスに Graviton、AES-NI、AVX、Turbo Boost などの特定の機能に x86 を活用し、NUMA ピン留めと C ステート/P ステートチューニングでサーバーパフォーマンスを最適化します。

リージョンの選択

GAMEPERF02: ゲームインフラストラクチャをホストする地理的リージョンを確認するにはどうすればよいですか?

ゲームインフラストラクチャに最適な場所を選択すると、プレイヤーとバックエンドのネットワークパフォーマンスを向上させることができます。プレイヤーベースがどこから接続しているか、コミュニティやサーバーがどのように構築されているかを考慮することは、地理的リージョンでの長期的な成長と持続可能性にとって重要です。分離されたゲームサーバーインフラストラクチャとバックエンドサービスをデプロイすると、複数のリージョン、ローカルゾーン、アウトポストを使用してゲームをホストすることで、全体的な運用効率が向上し、耐障害性が向上します。

ベストプラクティス

- GAMEPERF02-BP01 プレイヤーの近くにあるホームリージョンを選択する
- GAMEPERF02-BP02 レイテンシーの影響を受けやすいゲームインフラストラクチャをプレイヤーの近くに配置してパフォーマンスを向上させることをサポートするアプローチを設計する

GAMEPERF02-BP01 プレイヤーの近くにあるホームリージョンを選択する

初回のゲームローンチでは、プレイヤーがゲームを利用できると予想されるチームや、ローンチ前のマーケティングや広告活動に重点を置くチームなど、ビジネスステークホルダーとの議論に基づいてインフラストラクチャをデプロイする場所を決定する必要があります。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

ビジネスのステークホルダーには、プレイヤーの受信と実行可能性をよりよく理解するために、需要を刺激するメカニズムも必要です。例えば、これらのチームには、ゲームの事前注文、マーケティングイベントやキャンペーン、起動前にプレイヤーが関心を登録するためのパブリック E メールリストなどのメカニズム、およびゲームの起動時にプレイヤーが最も多い場所を判断するための関連するシグナルを確立するためのその他のアプローチがあります。ゲームでは、プレイテストとソフト起動フェーズを含むリージョンのロールアウト戦略を使用して、リージョンのプレイヤーの需要を判断することもできます。

プレイヤーベースとデベロッパーの近くにあり、ゲームをホストするために必要な AWS サービスと機能がある[ホームリージョンを選択します]。ホーム RSegion は、ゲームバックエンドサービスを実行する場所であり、ゲームサーバーを実行する場合もあります。サポートされているサービス、エッジロケーションへの接続、フェイルオーバーリージョンへの近接性、アベイラビリティーゾーンの数に基づいてホームリージョンを評価します。ローカルゾーンを使用している場合は、親リージョンが別の地域にある場合があることを考慮してください。例: サンティアゴ、チリローカルゾーン us-

east-1-scl-1a には、地理的にサンパウロ sa-east-1 に近いにもかかわらず、親リージョンとしてバージニア北部 us-east-1 があります。

実装手順

- 事前注文、マーケティングキャンペーン、インタレスト登録などの発売前アクティビティからのプレイヤーの需要シグナルに基づいて、デプロイリージョンを特定します。
- プライマリプレイヤーベースとデベロッパーに近いホームリージョンを選択し、必要な AWS サービス、エッジロケーション、フェイルオーバーリージョンをサポートしていることを確認します。
- 親リージョンがローカルゾーンの場所と地理的に異なる場合があるため、ローカルゾーンを慎重に評価してください。

GAMEPERF02-BP02 レイテンシーの影響を受けやすいゲームインフラストラクチャをプレイヤーの近くに配置してパフォーマンスを向上させることをサポートするアプローチを設計する

ゲームサーバーなどのレイテンシーの影響を受けやすいインフラストラクチャを個別に配置することで、長いネットワークルートの影響を最小限に抑えることができます。反復可能なデプロイにより、プレイヤーにとってパフォーマンスの高い複数の場所を簡単に維持できます。Ping はゲーム UI に表示される一般的なメトリクスであり、低い ping は差別化機能になる可能性があります。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

最初にゲームを起動するときに、プレイヤーベースに関する十分な情報がまだない場合があり、ゲームのプレイに最も関心のあるプレイヤーに最も近いインフラストラクチャをデプロイする最適な場所を十分に把握できます。これは一般的な課題であり、ホスティング配置戦略を迅速に調整して、必要なサーバーをプレイヤーの近くにデプロイできるアーキテクチャを設計することで、このシナリオに備える必要があります。ゲーム開発者は、ゲームインフラストラクチャのデプロイを定期的な起動後分析として定期的に評価し、反復的なアプローチで時間の経過とともに改善に段階的に投資するのが一般的です。

ベストプラクティスは、VPC、サブネット設定 VPCs 、重要なゲームサービスの起動に必要な依存関係などのインフラストラクチャの設定に、 AWS CloudFormation や Hashicorp の Terraform などの infrastructure-as-code テンプレートを使用することです。これにより、これらのテンプレートを参照し、必要に応じてすばやくカスタマイズして、プレイヤーをサポートするために追加のインフラストラクチャが必要な場所にデプロイできます。

また、将来の拡張を可能にするために、現在のデプロイ戦略をどのように進化させるかを理解しておく必要があります。IaC テンプレートは繰り返し可能ですが、ネットワーク計画に代わるものではありません。IPAM は VPCsを管理します。サブネットのサイズ設定、アベイラビリティーゾーンの選択、IP インベントリとクロスアカウントアベイラビリティーゾーンの調整。ネットワークは考慮すべき重要であり、変更されるとプレイヤーにとって破壊的になる可能性があります。複数の地理的ロケーションにデプロイされたゲームサーバーはゲームバックエンドに接続されます。これは、プライベート接続をサポートするために追加の設定が必要になる可能性がある単一または複数のホームリージョンでホストされることがより一般的です。これらの考慮事項は、ゲームの要件の進化やプレイヤーの要件の変化に応じてゲームホスティング戦略を変更できるように、時間の経過とともに継続的に評価する必要があります。

ゲームに使用するゲームホスティングロケーションの数を決定するときは、次の要素を考慮してください。

- プレイヤーエクスペリエンスの質の向上: ゲームホスティングロケーションを追加することで、プレイヤーエクスペリエンスをどの程度改善できますか? そうすることで達成できる増分パフォーマンスの向上は何ですか? このパフォーマンス向上をどのように測定しますか?
- 優先するプレイヤーの人数: ゲームホスティングロケーションを追加すると、何人のプレイヤーのエクスペリエンスを向上させることができますか? どのプレイヤー母集団または地理的場所を優先しますか?
- 変更によるダウンストリームへの影響: ゲームホスティング戦略を変更すると、プレイヤーのマッチメーキング待機時間にどのように影響しますか? プレイヤープール内のマッチサイズ、スキルバランス、またはプレイヤー数は、ゲームホスティングロケーション戦略を変更できますか? より多くのロケーションをサポートすると、プレイヤープールがフラグメント化され、コストと複雑さが増加する可能性があります。

これらの各考慮事項は、ゲームホスティングの場所を追加または削除する際に評価する必要があります。例えば、パフォーマンスの低いゲームプレイエクスペリエンスを持つ地理的な場所のプレイヤー、または最も声の高いパブリックフィードバックを表現するプレイヤーのエクスペリエンスの改善を優先できます。また、プレイヤーの収益化を優先順位に含めることもできます。例えば、ゲームの収益源が大きくなる地理的な場所のプレイヤーのエクスペリエンスの向上や、パフォーマンスの向上を導入した場合に増分収益を生み出す可能性に焦点を当てます。

でのインフラストラクチャのホスティングに加えて AWS リージョン、 の拡張機能である Local Zones を使用して AWS リージョン、ゲームサーバーやプレイヤーに近い音声チャットサーバーなどのレイテンシーの影響を受けやすいアプリケーションをホストできます。ゲーム開発チームのエクスペリエンスを向上させるために、Local Zones でゲーム開発インフラストラクチャを実行することも

できます。例えば、Local Zones を使用して、ゲーム開発者の近くでセルフマネージドソースコントロールサーバーのレプリカをホストするなどのユースケースに対処したり、Amazon EC2 インスタンス、EBS ボリューム、および Amazon FSx ファイルシステムを使用して、オンプレミスでインフラストラクチャをホストすることなく、開発スタジオの近くの 1 つ以上の Local Zones にデプロイされたゲーム開発仮想ワークステーションとコンテンツストレージをユーザーに提供したりできます。

Outposts は、リージョンまたはローカルゾーンが同じ地理的エリアで利用できない場合に適しています。ゲームサーバーがシステムの信頼性をバックエンドにできるように、データセンターから への接続を考慮 AWS する必要があります。 AWS Outposts および Outpost Server は、同じサービスと API を使用してデータセンター AWS で実行するように専用に構築されており、ゲームを実行する場所を問わず一貫したデプロイモデルを作成できます。 APIs 複数のラックを論理 Outpost に結合し、インフラストラクチャを共有できます AWS アカウント。ハードウェアライフサイクルは によって管理 AWS され、リードタイムは最短で 3 か月です。

コンテナを使用してゲームを構築し、独自のオンプレミスインフラストラクチャにデプロイできるオープンソースソフトウェアを使用してハイブリッドデプロイアーキテクチャを柔軟に採用したい場合は、 または Local Zones の代わりに ECS Anywhere AWS Outposts または EKS Anywhere を使用できます。Amazon GameLift でホストする場合、Amazon GameLift Anywhere を使用してローカルハードウェアでサーバービルドを実行できます。これにより、開発プロセスが高速化され、ローカルゾーンを使用したり、フリートの一部として独自のメタルを登録したりできます。

実装手順

- AWS CloudFormation や Terraform などの infrastructure-as-code ツールを反復可能なデプロイに使用して、プレイヤーのニーズに基づいてゲームホスティングロケーションをすばやくカスタマイズおよびスケーリングできます。
- ゲームホスティングロケーションを追加または削除するときに、プレイヤーエクスペリエンスの改善、プレイヤー母集団の優先順位、マッチメーキング時間などのダウンストリームへの影響を評価します。
- AWS Local Zones、Outposts、または ECS Anywhere、EKS Anywhere、GameLift Anywhere などのハイブリッドオプションを使用して、レイテンシーの影響を受けやすいインフラストラクチャを最適化し、多様なデプロイニーズに対応します。

反復開発

GAMEPERF03: Amazon GameLift を使用して反復開発のパフォーマンス効率を高めるにはどうすればよいですか?

Amazon GameLift は、ローカルテスト環境でゲームを開発およびパフォーマンステストするための end-to-endのワークフローを提供します。

ベストプラクティス

- GAMEPERF03-BP01 Amazon GameLift Anywhere と GameLift テストツールキットを使用する
- GAMEPERF03-BP02 ゲームサーバーのパフォーマンスとスケーラビリティをテストする
- GAMEPERF03-BP03 GameLift コンテナのリソース使用率を最適化する

GAMEPERF03-BP01 Amazon GameLift Anywhere と GameLift テストツールキットを使用する

反復的な開発プロセスを通じてパフォーマンス効率を向上させるには、Amazon GameLift Anywhere と Amazon GameLift テストツールキットを使用して包括的なテスト環境を確立します。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

このアプローチにより、迅速な反復、効率的なデータ収集、詳細なパフォーマンス分析が可能になります。主な手順は次のとおりです。

テスト環境を作成する

Amazon GameLift Anywhere を使用して、ローカルまたはクラウドベースのテスト環境を設定します。この設定により、各ゲームサーバービルドの反復をマネージドフリートにアップロードする必要がなくなり、アクティベーション時間が短縮されます。

Amazon GameLift テストツールキットを統合する

Amazon GameLift テストツールキットを開発ワークフローに組み込みます。このツールキットには、Amazon GameLift インフラストラクチャを視覚化し、仮想プレイヤーを起動し、FlexMatch シミュレーターを使用して FlexMatch ルールセットを反復処理するためのスクリプト、ツール、ライ

ブラリが用意されています。Amazon GameLift リソースの統合と管理を簡素化することで、一般的なタスクを自動化し、パフォーマンス分析に必要なデータを収集できます。

迅速なビルドとテストサイクル

新しいビルドでテストフリートをすばやく更新し、開始して、テストを開始します。これにより、build-test-repeatサイクルが高速化され、開発者はマルチプレイヤーインタラクションなど、ゲームのプレイヤーエクスペリエンスのさまざまな側面を検証できます。

包括的なテスト

Amazon GameLift サーバー SDK、バックエンドサービスインタラクション、マッチメーキング設定、その他の GameLift ホスティング機能とのゲームサーバー統合をテストします。GameLift テストツールキットを使用してテストを自動化し、詳細なパフォーマンスメトリクスを収集し、ゲームコンポーネントがシームレスに連携することを確認します。

パフォーマンスデータを分析する

GameLift Testing Toolkit によって収集されたデータを使用して、パフォーマンスのボトルネックを分析し、ゲームサーバーを最適化します。このツールキットは、主要なメトリクスの追跡、問題の特定、データ駆動型の意思決定を行い、パフォーマンス効率を向上させるのに役立ちます。

Amazon GameLift Anywhere と GameLift Testing Toolkit を反復開発プロセスに組み込むことで、迅速なテスト、包括的な統合チェック、詳細なパフォーマンス分析を通じてパフォーマンス効率を大幅に向上させることができます。

実装手順

- Amazon GameLift Anywhere を使用してテスト環境を作成し、ゲームサーバービルドのアクティベーション時間を短縮し、迅速な反復を可能にします。
- Amazon GameLift テストツールキットを統合して、テストタスクの自動化、プレイヤーのシミュレート、開発中の FlexMatch 設定の検証を行います。
- GameLift Testing Toolkit を使用してパフォーマンスデータを収集および分析し、ボトルネックを特定し、ゲームサーバーを最適化し、パフォーマンス効率を向上させます。

GAMEPERF03-BP02 ゲームサーバーのパフォーマンスとスケーラビリティをテストする

ゲームサーバーのパフォーマンスとスケーラビリティをテストするには、Amazon GameLift 機能と GameLift テストツールキットを使用して堅牢なテストフレームワークを実装します。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

主なプラクティスは次のとおりです。

反復テスト

Amazon GameLift Anywhere フリートを使用して、ゲームコンポーネントを繰り返し構築およびテストできるクラウドベースのホスト環境を作成します。この環境は実際のホスティング条件を反映し、現実的なパフォーマンスとスケーラビリティのテストを可能にする必要があります。

ゲームサーバー統合テスト

AWS CLI または GameLift Testing Toolkit を使用して、新しいゲームセッションの開始やゲームセッションイベントの追跡など、ゲームサーバーと Amazon GameLift サーバー SDK の統合をテストします。これにより、ゲームサーバーが GameLift 環境内で正しく機能することを確認します。

GameLift Testing Toolkit を使用してテストを自動化し、詳細なパフォーマンスメトリクスを収集します。ツールキットを使用すると、GameLift インフラストラクチャを視覚化し、負荷テスト用の仮想プレイヤーを起動し、FlexMatch シミュレーターを使用して FlexMatch ルールセットを反復処理できます。これは、サーバーインフラストラクチャをストレステストするために多数の同時ゲームセッションを作成してプレイヤーセッションをシミュレートする ECS Fargate タスクのスケーリングに特に便利です。

スケーラビリティテスト

ゲームセッションキュー設計、マルチロケーションフリート、スポットフリートとオンデマンドフリート、および複数のインスタンスタイプを試してください。ゲームセッションの配置オプション、レイテンシーポリシー、フリートの優先順位付け設定をテストします。プレイヤーの需要を満たすように容量スケーリングを設定し、システムがさまざまな条件下で予想される負荷を処理できることを検証します。

実装手順

- Amazon GameLift Anywhere を使用して、反復パフォーマンスとスケーラビリティのテストのための現実的なテスト環境を設定します。
- GameLift サーバー SDK とのゲームサーバー統合をテストし、GameLift 環境内での正しいセッション管理とイベント追跡を容易にします。

- GameLift テストツールキットを使用してスケーラビリティテストを実行し、プレイヤーの負荷をシミュレートし、セッションキューをテストし、フリートスケーリング、レイテンシーポリシー、優先順位付け設定を検証します。

GAMEPERF03-BP03 GameLift コンテナのリソース使用率を最適化する

GameLift コンテナのリソース使用率を最適化するには、コンテナフリートを効果的に設計し、正確なリソース制限を設定します。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

主なガイドラインは次のとおりです。

- コンテナグループ設計: ソフトウェアをコンテナグループに整理します。プライマリコンテナは、ゲームサーバーアプリケーションと Amazon GameLift エージェントをバンドルする必要があります。追加のソフトウェアにはサイドカーコンテナを使用して依存関係を管理し、メモリと CPU 使用率のコンテナ固有の制限を設定します。
- リソース制限を設定する: コンテナグループごとに、必要なメモリと CPU リソースを決定します。個々のコンテナにオプションの制限を設定して、予約済みリソースがあることを確認しますが、追加のリソースが利用可能な場合は、これらの制限を超えることもあります。これにより、リソースの競合やコンテナの潜在的な障害を防ぐことができます。
- デーモンコンテナグループ: プライマリコンテナグループでスケーリングする必要のないバックグラウンドプロセスまたはモニタリングプロセスにデーモンコンテナグループを使用することを検討してください。これにより、重要なバックグラウンドタスクがプライマリゲームサーバープロセスに影響を与えずに効率的に処理されることを確認します。

実装手順

- ゲームサーバーと GameLift エージェント用のプライマリコンテナと、依存関係を管理するためのサイドカーを使用して、特定のメモリと CPU 制限を持つコンテナグループを設計します。
- 各コンテナグループのリソース制限を設定して、競合を回避するために制御されたリソースの使用を許可しながら、必要なリソースを予約します。
- デーモンコンテナグループをバックグラウンドタスクまたはモニタリングタスクに使用して、プライマリゲームサーバープロセスに影響を与えずに効率的に動作することを確認します。

コンピューティングとハードウェア

GAMEPERF04: ゲームセッションが同じゲームサーバーインスタンスで実行されているプレイヤーに影響を与えるのをブロックするにはどうすればよいですか?

ゲームサーバーの実行後 AWS、リソース使用率、基盤となるコンピューティング、飽和度に関係なく、高品質のプレイヤーエクスペリエンスを提供するために、ゲームサーバーのパフォーマンスをモニタリングする必要があります。

ベストプラクティス

- GAMEPERF04-BP01 ゲームサーバープロセスをモニタリングして問題を検出する
- GAMEPERF04-BP02 シミュレートされた実際のゲームプレイシナリオでゲームサーバーのパフォーマンスをテストする

GAMEPERF04-BP01 ゲームサーバープロセスをモニタリングして問題を検出する

インスタンスごとに複数のゲームサーバープロセスを実行して、ゲームサーバーインスタンスのリソースを効率的に活用できます。その場合、ゲームセッションをホストする個々のゲームサーバープロセスが、同じインスタンスでホストされている他のゲームセッションに悪影響を及ぼさないようにアーキテクチャを設計します。メトリクスを使用して、ゲーム配置とゲームモードタイプがゲームサーバーインスタンスのパフォーマンスにどのように影響するかを理解します。ゲームサーバーインスタンスがホットスポッティングされないように、低負荷 (ロビー、ショップ、またはシングルプレイヤーチュートリアル) と高負荷 (ランク付け、マルチプレイヤー、またはハイスキルゲームプレイ) のプロセスを組み合わせて組み込みます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

ping 時間とジッター、フレームドロップ、API 応答時間、エラー、ゲームループの正常な完了に関するテレメトリを収集して、クライアント側とサーバー側のメトリクスを通じてプレイヤーエクスペリエンスをモニタリングします。これらのイベントのタイムスタンプをプレイヤーサポートの問題やサーバーログと関連付けて、パフォーマンスのボトルネックを特定しま

す。Dtrace、ftrace、uperf、eBPF などのツールは、システムパフォーマンスの詳細な調査と分析に使用できます。

ゲームサーバーインスタンスで使用できる制限されたリソースのモニタリングを実装して、個々のゲームサーバープロセスが事前に定義されたリソース予算のしきい値を超えたときにアラートを生成できるようにします。しきい値を超えると、ゲームサーバーエンジニアがこの動作を調査できるように、関連するシステムとゲームサーバーのログ記録ソリューションなどの耐久性のあるストレージにログアウトするようにゲームサーバーソフトウェアを設定できます。さらに、ゲームサーバーインスタンスは、インスタンスで実行されている各ゲームサーバープロセスのメトリクスをレポートするように設定する必要があります。これにより、ゲームサーバーインスタンスの全体的なメトリクスに加えて、これらの個々のゲームサーバープロセスをモニタリングできます。

例えば、GameLift はゲームセッションをモニタリングするためのメトリクスを提供します。このメトリクスは、ゲームサーバーインスタンスで設定できる Amazon CloudWatch エージェントを使用して収集されたカスタムゲーム固有のメトリクスとログで拡張できます。メトリクスは CloudWatch で表示することも、シングルサインオンと統合されている Amazon Managed Grafana などの他のツールにエクスポートすることもできます。これにより、 マネジメントコンソールにアクセスできないユーザーがメトリクスに簡単にアクセスできます。Amazon GameLift を使用してログとメトリクスを管理するための以下のベストプラクティスを参照してください。これにより、個々のゲームセッションログの表示もサポートされます。

実装手順

- インスタンスごとに複数のゲームサーバープロセスを実行し、低負荷ゲームモードと高負荷ゲームモードを組み合わせて、ホットスポッティングを回避し、バランスの取れたリソース使用率を検証します。
- ping、ジッター、フレームドロップ、API 応答時間などのクライアント側およびサーバー側のメトリクスをモニタリングし、これらをプレイヤーによって報告されたサーバーログや問題と関連付けてボトルネックを特定します。
- 各ゲームサーバープロセスのリソースモニタリングを設定し、しきい値違反のアラートを生成し、CloudWatch や Amazon Managed Grafana などのツールを使用して、分析のために耐久性のあるストレージにログを保存します。

GAMEPERF04-BP02 シミュレートされた実際のゲームプレイシナリオでゲームサーバーのパフォーマンスをテストする

パフォーマンステストを実施し、さまざまなゲームプレイシナリオを評価して、ゲームサーバープロセスが EC2 インスタンスメモリ、CPU、ネットワーク帯域幅などの固定リソースの使用率を適切に処理しているかどうかを判断します。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

プレイヤーの一般的なゲームプレイパスと動作をミラーリングできるボットを使用してシミュレートされたゲームプレイテストを作成すると、ゲームサーバープロセスがさまざまな使用シナリオでこれをどのように処理するかを判断できます。たとえば、ゲームクライアントシミュレーションやゲームクライアントビルドを実行してゲームプレイシナリオを生成するようにカスタマイズできる分散負荷テスト AWSなどのソリューションを に実装できます。内部プレイテストを実行し、QA チームを使用してゲームのさまざまな機能をストレステストすることで、ゲームが最適に動作するように設計されているという確信を深めることができます。 AWS Device Farmを使用して、iOS、Android、ブラウザゲームのモバイルテストを複数のデバイスタイプで実行できます。

実装手順

- 一般的なプレイヤー動作をシミュレートするボットを使用してパフォーマンステストを実行し、さまざまなシナリオでゲームサーバーのリソース使用率を評価します。
- での分散負荷テストなどのソリューションを使用して AWS 、ストレステストのゲームプレイシナリオをカスタマイズおよびシミュレートします。
- 内部プレイテストを実行し、さまざまなデバイスでモバイルおよびブラウザのゲームテスト AWS Device Farm に などのツールを使用します。

コンピューティングの選択

GAMEPERF05: ゲームに適したコンピューティングソリューションをどのように選択しますか?

コンピューティングパフォーマンスは、インスタンスサイズとファミリーによって異なります。個別のキャパシティプールからの複数のコンピューティングオプションを使用することが有益です。パ

フォーマンスを優先するが、容量不足エラーを回避するのに十分な多様性を含むフリート構成戦略を策定します。

ベストプラクティス

- GAMEPERF05-BP01 複数のコンピューティングタイプでゲームのパフォーマンスをベンチマークする
- GAMEPERF05-BP02 non-latency-sensitiveコンピューティングタスクを非同期ワークフローに移動する

GAMEPERF05-BP01 複数のコンピューティングタイプでゲームのパフォーマンスをベンチマークする

ゲームサーバーのワークロードでは、ゲームサーバーをホストするための最適なコンピューティングソリューションを特定するための単一のアプローチはありません。ゲームサーバーをベンチマークするための一般的な戦略は、コンピューティングに最適化された EC2 の「c」インスタンスから始めることです。このインスタンスファミリーは、計算負荷の高いワークロードに高いパフォーマンスを提供するためです。または、ゲームで特定の機能を実装するために大量のメモリが必要な場合は、メモリ最適化インスタンスが最も適している場合があります。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

ワークロードで重要なネットワークリソースを使用している場合は、ネットワーク最適化のインスタンスを実装することを検討してください。これは通常、インスタンス名に「n」と表示されているため、クレジットが枯渇するとパフォーマンスが低下するため、バースト可能なインスタンスタイプ「n」は避けてください。ゲームはレイテンシーやドロップされたパケットの影響を受けやすいため、ゲームサーバーのネットワークパフォーマンスを向上させるために EC2 拡張ネットワーキングを使用することをお勧めします。拡張ネットワーキングでは、単一ルート I/O 仮想化 (SR-IOV) を使用して、サポートされているインスタンスタイプ で高性能ネットワーキング機能を提供します。SR-IOV は、従来の仮想化ネットワークインターフェイスと比較し、I/O パフォーマンスが高く、CPU 利用率が低いデバイス仮想化の手法です。拡張ネットワーキングは、高い帯域幅、1 秒あたりのパケット (PPS) の高いパフォーマンス、常に低いインスタンス間レイテンシーを実現します。Elastic Network Adapter による拡張ネットワーキングは、最新の EC2 インスタンスタイプで利用でき、新しいインスタンスのパフォーマンス強化と Nitro ハイパーバイザーの改善からメリットを得るために定期的に更新することが重要です。 AWS

ゲームが複数の EC2 インスタンスタイプで同様に動作する場合は、複数のインスタンスタイプを使用してゲームサーバーをホストすることを検討する必要があります。パフォーマンスの傾向を特定するために十分な本番稼働用ゲームセッションをホストした後、時間の経過とともにパフォーマンスをモニタリングし、さらに最適化を実行します。リソースの割り当てが異なる新機能をゲームに追加すると、コンピューティング要件が変わる場合があることに注意してください。複数のインスタンスタイプを使用するように EC2 Auto Scaling グループを設定することも、個別の Auto Scaling グループを使用して、個別のインスタンスタイプを実行するゲームサーバーインスタンスをホストすることもできます。これにより、メトリクスの相関関係と集計を簡単に管理できます。

インテルベースのインスタンス、AMD ベースのインスタンス、ARM ベースの Graviton インスタンスなど、さまざまなタイプのプロセッサでゲームがどのように動作するかを評価します。Unreal Engine 5.1.1 以降では、Graviton 用のゲームサーバーをコンパイルでき、ゲームの料金パフォーマンスを向上させることができます。各ファミリー内のさまざまなサイズでスイープテストと飽和テストを実行して、使用率とパフォーマンスが一貫しているスイートスポットを判断します。

また、コンテナと Lambda 関数を使用してホストされている場合のゲームのパフォーマンスへの影響をベンチマークする必要があります。非同期ゲームやゲームバックエンドサービスなど、存続期間の長いゲームサーバープロセスを必要としないユースケースでは、Lambda でサーバーレスアーキテクチャを使用することを検討してください。これにより、ゲーム運用チームの管理と運用を簡素化できるだけでなく、ゲームをグローバルに多くの にすばやくデプロイできます AWS リージョン。サーバーレスのベストプラクティスについては、「サーバーレスアプリケーションレンズ - Well-Architected フレームワーク」を参照してください。

実装手順

- CPU 負荷の高いワークロードではコンピューティング最適化の「c」インスタンス、メモリ負荷の高いタスクではメモリ最適化インスタンス、ネットワークスループットが高い場合はネットワーク最適化の「n」インスタンスでゲームサーバーをベンチマークします。
- サポートされているインスタンスで Elastic Network Adapter (ENA) による拡張ネットワーキングを使用して、ネットワークパフォーマンスの向上、レイテンシーの短縮、パケット処理速度の向上を実現します。
- 複数のインスタンスタイプ、プロセッサ (Intel、AMD、Graviton)、コンテナまたは Lambda ホスティングオプションを評価してテストし、ゲーム機能の進化に合わせてコンピューティングソリューションを調整します。

詳細については、「グローバルゲームサーバーに適したコンピューティング戦略を選択する」を参照してください。

GAMEPERF05-BP02 non-latency-sensitiveコンピューティングタスクを非同期ワークフローに移動する

ゲームのパフォーマンスを最適化するときは、クライアントとゲームバックエンド間のインタラクションの一部のみを同期的に実行する必要があることに注意してください。プレイヤーエクスペリエンスの観点から各機能を検討し、特定のインタラクションでブロックされリソースを大量に消費する同期通信が必要かどうか、またはそれらの機能を非同期的に実装できるかどうかを決定する必要があります。ネットワーク呼び出しを実装するときは、非同期のノンブロッキングアプローチを使用します。さらに、ゲームバックエンドは、タスクをキューにオフロードし、可能な場合はクライアントへの高速応答を優先することで、効率的な方法で作業を実行するように設定する必要があります。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

たとえば、プレイヤーセッションの終了時にリーダーボードを更新すると、クライアントがリーダーボードの更新が完了するまで待つ必要がないように、非同期で実装できます。代わりに、ゲームクライアントにこの非同期を実装し、これらのタイプのオペレーションを Amazon SQS などのキューにプッシュするようにバックエンドサービスを設計することを検討してください。このアーキテクチャでは、リクエストを受け入れるようにバックエンドを設定し、非同期処理のためにメッセージを永続的に保存し、クライアントにすみやかに応答するのに役立つ SQS にキューに入れます。リーダーボードの更新が完了すると、バックエンドはゲームクライアントに更新を送信して、プレイヤーのリーダーボードの表示を更新できます。

または、プレイヤーはゲームのリーダーボード画面にアクセスして最新のデータを取得するだけで、バックエンドにウェブリクエストを発行してキャッシュから最新のデータを取得できます。

実装手順

- クライアントとバックエンドのやり取りに同期通信が必要かどうかを判断します。可能な場合は非同期のノンブロッキングアプローチを実装して、リソースの使用を最適化します。
- Amazon SQS を使用して、リーダーボードの更新などの重要でないタスクをオフロードします。
- クライアントが最新のリーダーボードデータをオンデマンドで取得したり、バックグラウンド更新で取得したりするなど、更新されたデータを非同期的に取得できるようにします。

リソース

- マイクロサービスの非同期メッセージングについて

- Lambda - サービス統合と非同期処理の使用

データ管理

GAMEPERF06: ゲームサーバーログを効率的に管理および分析し、最適なパフォーマンスを得るためにさまざまなタイプのゲームデータを保存するにはどうすればよいですか?

ゲームには、プレイヤーデータ、ゲームログ、サーバーログを含めることができます。これらのログは、可能な限り互いに切り離す必要があります。ログの取り込みとライフサイクル管理を一元化することで、ゲームチームとサーバーで何が起こっているかを把握できます。

ベストプラクティス

- GAMEPERF06-BP01 ログの収集とストレージを一元化する
- GAMEPERF06-BP02 アクセスパターンに基づいてゲームデータを分類して保存する
- GAMEPERF06-BP03 効率的なログフォーマットとバッチ処理を有効にする
- GAMEPERF06-BP04 ログのローテーションと保持ポリシーを実装する
- GAMEPERF06-BP05 モニタリングツールと視覚化ツールを使用する

GAMEPERF06-BP01 ログの収集とストレージを一元化する

一元化されたログ収集およびストレージソリューションを実装して、ゲームサーバーインスタンスと GameLift からログを収集します。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

Amazon CloudWatch Logs などのサービスを使用して、ゲームサーバーと GameLift インスタンスからログデータを収集、モニタリング、保存します。CloudWatch Logs は、ログ管理のためのスケーラブルでフルマネージド型のソリューションを提供し、ゲームサーバーのパフォーマンスに影響を与えることなくログデータの効率的な保存と取得を容易にします。CloudWatch Logs エージェントを実行している場合は、ゲームサーバーへの影響を最小限に抑えるために、バッチサイズ、バッファ期間など、さまざまなインストールタイプと設定オプションを検討してください。ゲームサーバー

インスタンスのエフェメラルを考慮し、可能な限りローカライズされたログ記録への依存を減らします。ログ記録のベストプラクティスを実装するための一元化されたポリシーを確立します。

実装手順

- Amazon CloudWatch Logs を使用して、ゲームサーバーインスタンスと GameLift からログデータを収集、モニタリング、保存し、一元化されたスケーラブルなログ管理を容易にします。

GAMEPERF06-BP02 アクセスパターンに基づいてゲームデータを分類して保存する

ゲームデータを、アクセスパターンとストレージ要件に基づいて異なるタイプに分類します。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

一般的なカテゴリには、プレイヤーデータ、ゲームセーブ、永続ワールドストレージ、分析データなどがあります。

実装手順

各データ型に適切なストレージソリューションを使用して、パフォーマンスとコスト効率を最適化します。

- プレイヤーデータ: 高速でスケーラブルな NoSQL データベースである Amazon DynamoDB を使用して、プレイヤープロファイル、設定、進行状況データを保存します。DynamoDB の低レイテンシーアクセスと自動スケーリング機能により、プレイヤーデータの効率的な取得と更新が可能になります。
- ゲームセーブ: Amazon S3 を使用してゲームセーブとチェックポイントを保存します。S3 は、大量のゲームセーブデータを保存するための高い耐久性とスケーラビリティを提供します。ゲームセーブのアップロードとダウンロードを高速化するには、S3 Transfer Acceleration または Amazon CloudFront の使用を検討してください。
- 永続ワールドストレージ: 永続ワールド状態または共有ゲームデータを持つゲームの場合は、Amazon DynamoDB、Amazon ElastiCache、または Amazon MemoryDB の使用を検討してください。ElastiCache と MemoryDB はインメモリキーバリューストアを提供し、DynamoDB は SSD ベースの NoSQL データベースです。これらのサービスは、保存されたデータへの高速アクセスを提供し、ゲームサーバープロセスがゲーム状態を保存するのにかかる時間を短縮し、プロセス全体のパフォーマンスを向上させます。

- 分析データ: Amazon Managed Streaming for Apache Kafka または Kinesis Data Streams を使用して、ゲームデータプロデューサーからデータストリームを取り込みます。Amazon Managed Service for Apache Flink は、リアルタイムの変換と分析に使用し、処理とバックエンドのデータレイク、ウェアハウス、分析サービスへの配信のために Amazon Data Firehose に送信できます。の Game Analytics Pipeline のガイダンス AWSは、サービスがどのように連携してほぼリアルタイムのバッチ分析を提供するかを示しています。

GAMEPERF06-BP03 効率的なログフォーマットとバッチ処理を有効にする

JSON などの解析可能な形式で構造化されたログを生成するようにゲームサーバープロセスを設定します。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

ログバッチ処理技術を実装して、ゲームサーバーから集中型ログストレージへのログデータ転送の頻度を最小限に抑えます。ログのバッチ処理により、ネットワークオーバーヘッドが軽減され、ゲームサーバーのパフォーマンスが向上します。詳細またはデバッグレベルのログは、可能な限り避けるべきパフォーマンスとコストのペナルティが発生する可能性があるため、デフォルトではなく例外として使用します。

GAMEPERF06-BP04 ログのローテーションと保持ポリシーを実装する

ログローテーションと保持ポリシーを確立して、ログデータの増加を管理し、ストレージ使用率を最適化します。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

サイズまたは時間間隔に基づいてログを自動的にローテーションするようにゲームサーバーを設定します。Amazon CloudWatch Logs でログ保持ポリシーを定義して、アクティブな分析やトラブルシューティングに不要になった古いログデータを自動的にアーカイブまたは削除します。

GAMEPERF06-BP05 モニタリングツールと視覚化ツールを使用する

モニタリングおよび視覚化ツールを使用して、ゲームサーバーのパフォーマンスに関するインサイトを取得し、最適化の機会を特定します。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

Amazon CloudWatch を使用して主要なメトリクスをモニタリングし、プロアクティブ通知のアラームを設定します。Amazon Managed Service for Prometheus や Amazon Managed Grafana などのツールを使用して、ゲームサーバーやインフラストラクチャからメトリクスを収集、クエリ、視覚化します。パフォーマンスを追跡し、ボトルネックを特定し、データ駆動型の最適化を行うための有益なダッシュボードを作成します。

ネットワークとコンテンツ配信

GAMEPERF07: パフォーマンスを最適化するためにマッチメーキングサービスを設計するにはどうすればよいですか?

プレイヤースキル、インターネットサービスプロバイダー (ISP) の品質、プレイヤー母集団の分布は、パフォーマンスチューニングのディメンションとして考慮することが重要です。ゲームセッションは、プレイフィールドを水平化し、公平なゲームをホストするように戦略的に配置されたサーバーに配置することができます。

ベストプラクティス

- GAMEPERF07-BP01 ゲームのネットワークレイテンシーのしきい値を定義する
- GAMEPERF07-BP02 ゲームプレイモードとゲームホスティングリージョンごとに個別のマッチメーキングサービスを実行する
- GAMEPERF07-BP03 マッチメーキングパフォーマンスを定期的にモニタリングする
- GAMEPERF07-BP04 ネットワークパフォーマンスを定期的にモニタリングする
- GAMEPERF07-BP05 ネットワークアクセラレーションテクノロジーを使用してインターネット全体のパフォーマンスを向上させる

GAMEPERF07-BP01 ゲームのネットワークレイテンシーのしきい値を定義する

マルチプレイヤーゲームを開発するときは、ゲームインフラストラクチャがプレイヤーに不要なレイテンシーを引き起こさないことを確認します。ゲームがネットワークレイテンシーに敏感である場合

は、マッチメーキングロジックでレイテンシーのしきい値を設定して、近くのゲームサーバーロケーションでホストされている、または理想的なプレイヤーエクスペリエンスの目標を満たすゲームサーバーセッションにプレイヤーを配置 AWS リージョン することを優先する必要があります。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

多くのレイテンシーの影響を受けやすいゲームでは、ゲームクライアントを計測してゲームの各インフラストラクチャロケーションに ping を送信し、ネットワークレイテンシー、ジッター、パケット損失などのパフォーマンスデータを収集し、このデータをメトリクスコレクションのバックエンドに報告して分析できるようにするのが一般的です。プレイヤーをゲームセッションにマッチングする場合、ゲームクライアントが認識するネットワークレイテンシーをマッチメーキングサービスロジックで使用される入力の 1 つとしてゲームサーバーインフラストラクチャに組み込むようにゲームを設定できます。

GAMEPERF07-BP02 ゲームプレイモードとゲームホスティングリージョンごとに個別のマッチメーキングサービスを実行する

ゲームがプレイヤーが選択できる複数のゲームプレイモードを提供している場合は、それぞれのマッチメーキングシステムを分離して、固有の要件に基づいて各ゲームプレイモードのパフォーマンスを個別に調整し、リソースの競合を減らすことができるようにする必要があります。各ゲームプレイモードには、許容可能なレイテンシー、マッチサイズ、その他のゲーム固有のマッチメーキングロジックをカスタマイズするための固有の要件がある可能性があります。また、さまざまなタイプのプレイヤーを引き付ける可能性もあります。各ゲームモードのマッチメーキングサービスを個別のソフトウェアデプロイとして実行し、ゲームモードのパフォーマンステストと運用を個別に実行できるようにします。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

例えば、ゲームモードごとに個別の Lambda 関数として実行したり、個別のコンテナベースのサービスデプロイとして運用したりできます。

マッチメーキングサービスをゲームサーバーの場所に近い複数のリージョンにデプロイします。プレイヤートラフィックには多くのルートがかかるため、低レイテンシーのゲームセッション配置の効率を向上させるには、マッチメーキングサービスが複数の ISPs にわたってup-to-dateレイテンシープロファイルを維持することが重要です。GameLift FlexMatch は、マッチメーカーのリージョンを

選択するための追加のガイダンスを提供し、マッチメーカーをマルチリージョンゲームセッションキューと統合する機能が含まれています。

GAMEPERF07-BP03 マッチメーキングパフォーマンスを定期的にモニタリングする

プレイヤーのゲームのパフォーマンスを最適化する最も顕著な方法の 1 つは、ゲームセッションに入る前に待つ必要がある時間を短縮することです。待機時間が長くなるとプレイヤーの関心が失われ、減少する可能性があるため、マッチメーキングソリューションを設計する際にはこれを考慮することが重要です。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

ゲームのマッチメーキング設定を設計する場合は、マッチを形成するために適用される条件を決定するルールを作成します。これらのルールがシステムのパフォーマンスに与える影響、特にプレイヤーの待機時間を考慮する必要があります。新しいマッチメーキング条件やフィルターの追加など、マッチメーキング実装に変更をデプロイする前に、事前にこれをテストするか、この変更を Canary または A/B テストとして少数のサンプルのプレイヤーに徐々にリリースしてパフォーマンスメトリクスを収集することを検討してください。

マッチメーキングサービスを設定して詳細なログを生成し、各マッチメーキングリクエストに適用された条件またはルールを理解します。これにより、レビューが容易になり、必要に応じてマッチメーキングの実装が調整されます。

例えば、Amazon GameLift FlexMatch は、完全マネージド型マッチメーキングサービスを提供します。このサービスは、独自のゲームサーバーホスティングでスタンドアロンサービスとして使用することも、Amazon GameLift でホストされているゲームサーバーでも使用できます。FlexMatch は Amazon EventBridge にイベント通知を生成できます。FlexMatch イベント通知の設定」を参照してください。Amazon Simple Notification Service (Amazon SNS) を使用して JSON 形式でマッチメーキングデータを受信すると、この情報を自動的に処理して分析用に保存し、マッチメーキングのパフォーマンスを向上させることができます。

マッチメーキングサービスがプレイヤーに適したゲームセッションを見つけるのにかかる時間を追跡するメトリクスを設定します。マッチメーキング期間メトリクスを定期的に確認し、これらの時間をプレイヤーの動作やコミュニティの感情と関連付けます。このデータを使用して、マッチメーキングルール設定に含めることができるマッチメーキングタイムアウトに適したしきい値を作成します。

例えば、Amazon GameLift FlexMatch は、マッチメーキングリクエストのタイムアウトを定義し、時間の経過とともに要件を緩和できるマッチメーキングルールを作成するためのサポートを提供します。この機能を使用すると、マッチを簡単に調整してマッチを作成し、マッチを見つけるのが難しい場合にプレイヤーをゲームセッションに入れることができるマッチメーキングを作成できます。

GAMEPERF07-BP04 ネットワークパフォーマンスを定期的にモニタリングする

競合ゲームでは、一貫したプレイヤーエクスペリエンスを持つことが重要です。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

より大きなプレイヤーベースで確実に 50 ミリ秒のゲームは、1 人のプレイヤーが 10 ミリ秒の ping を持ち、もう 1 人のプレイヤーが 70 ミリ秒の ping を持つマッチングよりも公平で面白いです。ISP ルーティングの変更はプレイヤー母集団の一部に影響を与える可能性があるため、マッチメーキングシステムは適応する必要があります。Amazon CloudWatch Network Monitoring は、ゲームまたはプレイヤーのインターネットプロバイダーに問題があるかどうかを判断するのに役立ちます。

実装手順

- Amazon Cloudwatch Network Monitoring を使用して、ネットワークパフォーマンスを追跡し、ルーティングの問題を特定します。
- VPC フローログを使用して、異常なトラフィックパターンやドロップされたパケットを識別します。これは、ネットワークの輻輳、ISP の問題、プレイヤーのレイテンシーに影響する設定ミスを示している可能性があります。

GAMEPERF07-BP05 ネットワークアクセラレーションテクノロジーを使用してインターネット全体のパフォーマンスを向上させる

レイテンシーの影響を受けやすいゲームインフラストラクチャをプレイヤーの近くに配置するだけでなく、ゲームのネットワークパフォーマンスを最適化することでプレイヤーのエクスペリエンスを向上させることもできます。 AWS は、BGP プロトコルを使用してインターネットルーティングに影響を与え、インターネットサービスプロバイダーからの境界ネットワークへの最速パスを使用します。独自のネットワークを運用し、ルーティング動作と BGP アドバタイズをより詳細に制御およびオブザーバビリティする必要がある場合は、プライベートピアリングまたは Direct Connect を使用して、インターネットから実行中のゲームにトラフィックをルーティングできます AWS。

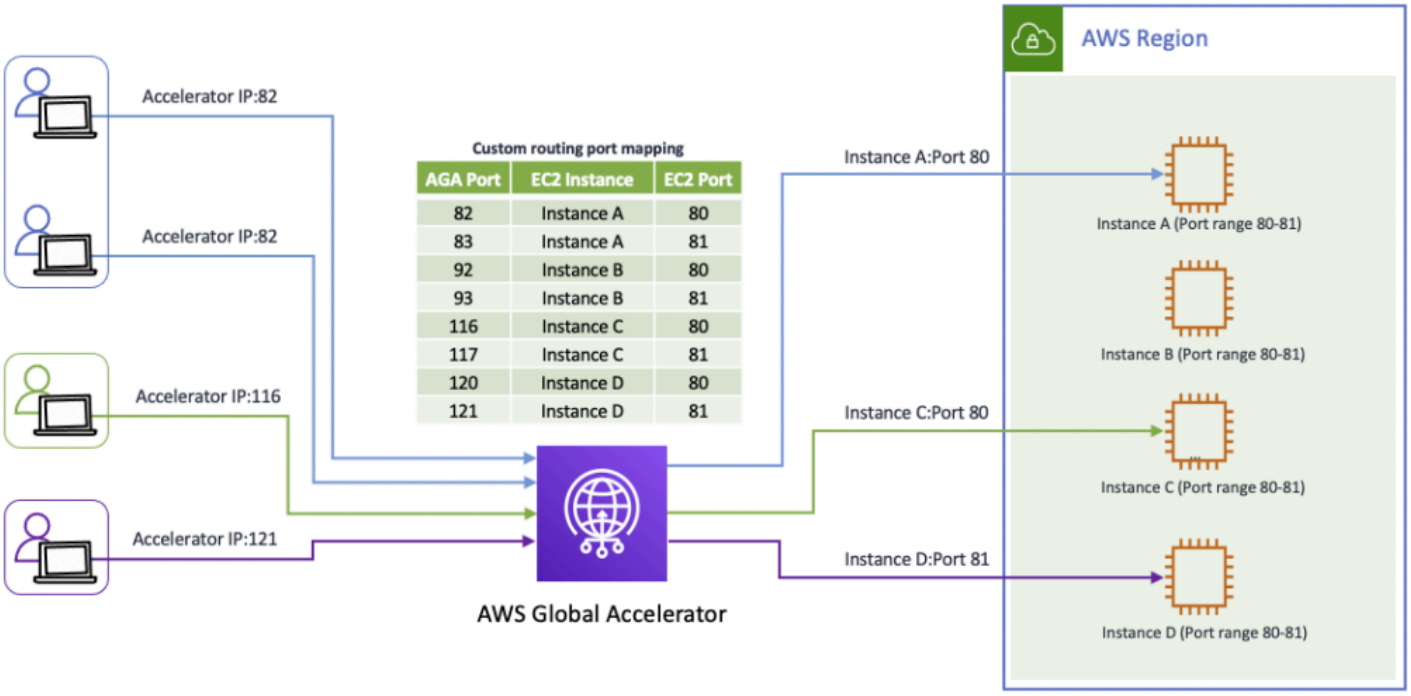
このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

インターネットのパフォーマンスと応答性の向上をサポートするために、次のリファレンスアーキテクチャを検討してください。

Global Accelerator を使用したゲームのネットワークパフォーマンスの向上

ネットワークルーティングに対するフルマネージドソリューションとして、AWS Global Accelerator は AWS グローバルネットワークを使用してアプリケーションのネットワークパフォーマンスを向上させます。グローバルネットワークは、ゲームサーバーへの高速フェイルオーバーを提供しながら、ゲームプレイトラフィック、音声チャット、リアルタイムメッセージングトラフィック、およびその他のレイテンシーの影響を受けやすいアプリケーションを加速するために使用できます。Global Accelerator カスタムルーティングアクセラレーターをマッチメーキングサービスと統合して、静的なエニーキャスト IP アドレスとポートを使用して、複数のプレイヤーを同じゲームセッションに決定的にルーティングできます。



ゲーム開発チームは世界中に分散され、共有コンテンツまたはアセットへのパフォーマンスの高いアクセスが必要になる場合があります。Amazon S3 バケットに保存されている共有コンテンツのパフォーマンスを向上させるために、S3 クロスリージョンレプリケーションを使用してリージョン間でのデータの双方向レプリケーションを設定することで、ユーザーはより近いバケットからデータにアクセスできます。このアクセスパターンを簡素化するには、Global Accelerator を使用してグロー

バルネットワーク経由で S3 へのリクエストを高速化する S3 マルチリージョンアクセスポイントを使用します。 S3

詳細については、AWS 「Global Accelerator と Amazon GameLift FleetIQ を活用してプレイヤーエクスペリエンスを向上させる」を参照してください。

実装手順

- AWS Global Accelerator を使用すると、ゲームサーバーへの高速フェイルオーバーを容易にしながら、ゲームプレイトラフィック、音声チャット、リアルタイムメッセージングのネットワークパフォーマンスを向上させることができます。
- マッチメーキングサービスと統合するように Global Accelerator カスタムルーティングアクセラレーターを設定し、静的エニーキャスト IPs を使用してプレイヤーをゲームセッションに決定的にルーティングできるようにします。
- S3 クロスリージョンレプリケーションを有効にして、リージョン間で共有コンテンツを分散ゲーム開発チームにレプリケートします。
- S3 マルチリージョンアクセスポイントを使用して、グローバルに分散されたユーザーの AWS グローバルネットワーク経由の S3 データアクセスを高速化します。

プロセスと文化

GAMEPERF08: ゲームのパフォーマンスバーをプレイヤーやデベロッパーの期待に合わせるにはどうすればよいですか?

プレイヤーと開発者を知ることは、パフォーマンス効率を向上させる上で最も重要な側面の 1 つです。運用上のオーバーヘッドを抑えてパフォーマンスの高いゲームを配信することは、プレイヤーやデベロッパーに自分の経験を気にし、ゲームやスタジオを差別化できることを示す最善の方法の 1 つです。

ベストプラクティス

- GAMEPERF08-BP01 プロセスにプレイヤーを通知して含める
- GAMEPERF08-BP02 ソリューションの選択をエンジニアリングチームのスキルと専門知識に合わせる

GAMEPERF08-BP01 プロセスにプレイヤーを通知して含める

レイテンシー、1 秒あたりのフレーム数、ドロップされたパケット数などのゲームメトリクスに表示するオプションを指定します。ステータスページなどのプレイヤー向け通信を通じて、インフラストラクチャの問題とメンテナンスのダウンタイムを表面化します。開発ブログを含むプレイヤーコミュニケーションで新しいゲームロケーションを祝い、期待されるプレイヤーエクスペリエンスの改善に対する期待を設定します。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

プレイヤーを含める

関連するファイルを収集し、ゲームクライアントからプレイヤーサポートチケットにアタッチする簡単な診断送信プロセスを提供します。プレイヤーが互いに助け合い、ゲームエクスペリエンスの向上に参加できるサポートフォーラムを有効にする

トレードオフとプレイヤーの期待を考慮する

コスト効率のためにバックエンドシステムを移動することは、プレイヤーにとっては目立たないかもしれませんが、ゲームサーバーを移動すると ping 時間が変わる可能性があります。ゲームホスティングロケーションの拡張と縮小について、プレイヤーに一貫性と公平性を持たせます。

プレイヤーコミュニティと地域には、ゲームの期待に影響を与える可能性のある独自の特性があります。たとえば、韓国には地球上で最速のインターネットがあり、ゲームプレイへの期待は 1 桁のレイテンシーであり、競争の激しいプレイを促進します。モバイルデバイスでのカジュアルなゲームプレイは、コンソールや PC セッションプレイとは異なるパフォーマンスプロファイルを作成し、使用パターンを作成します。

ログインとロビーはエクスペリエンスの一部であり、サーバーがメンテナンスのためにオフラインであっても応答性を実感する必要があります。レイドナイトプランニングやロビーでのハングアウトはプレイヤーエクスペリエンスの一部であり、パフォーマンス効率のために重点分野を選択するときは考慮することが重要です。プレイヤーはゲームクライアントを数か月間開いたままにし、時折ログインしてパッチノートを読むことがあります。Live Ops ゲームは、エンジニアリングプロセスと文化の一環として、プレイヤーエクスペリエンス全体を念頭に置く必要があります。

実装手順

- レイテンシー、FPS、パケット損失などのゲーム内メトリクスを提供し、ステータスページやプレイヤー向け更新を通じてインフラストラクチャの問題やメンテナンススケジュールを伝えます。

- ゲームクライアントに診断ダンプと送信機能を実装し、コミュニティ主導のトラブルシューティングと改善を促進するためのサポートフォーラムを作成します。
- 競合するリージョンの低レイテンシーや、偶然および長時間のセッションプレイヤーの応答的なログイン/ロビーエクスペリエンスなど、プレイヤーコミュニティの期待に合わせてパフォーマンスの最適化を調整します。
- アクティブなゲームプレイからアイドル状態のクライアント動作まで、プレイヤーエクスペリエンス全体を考慮して Live Ops ワークフローを設計し、シームレスなエンゲージメントを促進します。

GAMEPERF08-BP02 ソリューションの選択をエンジニアリングチームのスキルと専門知識に合わせる

ホスティングオプションを選択する際に、ゲームサーバーのパフォーマンスを管理および最適化するためのチームのスキルと専門知識を評価します。EC2 やコンテナなどのセルフホスト型ソリューションには、インフラストラクチャ管理、パフォーマンス調整、スケーリングに関するより多くの知識が必要です。チームがこれらのスキルを欠いている場合、GameLift などのマネージドサービスの方が適している場合があります。これは、複雑さの多くを抽象化し、チームがゲーム固有の最適化に集中できるようにするためです。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

これらの要因を評価し、さまざまなホスティングオプションでパフォーマンステストを実施することで、パフォーマンス効率を最適化しながら、ゲーム固有の要件を満たす最適なソリューションを選択できます。

リソース

パフォーマンス効率に関連するベストプラクティスについて説明します。

関連ドキュメント:

- AWS アーキテクチャセンター
- パフォーマンス効率の柱 – AWS Well-Architected フレームワーク
- オンプレミスストレージパターンと AWS ストレージサービスの比較
- EC2 インスタンスのインスタンスストア一時ブロックストレージ

- CloudWatch エージェントを使用してメトリクス、ログ、トレースを収集する
- CloudWatch エージェント
- EC2 インスタンスで拡張ネットワーキングを有効にして設定するにはどうすればよいですか?
- AWS Global Accelerator と Amazon GameLift FleetIQ を活用してプレイヤーエクスペリエンスを向上させる
- Riot Games テクノロジーブログ: Valorant のスケーラビリティと負荷テスト
- ハイブリッド AWS ソリューションを使用したハイパースケールのオンラインゲーム
- 使用率の飽和とエラー (USE) メソッド
- Amazon EC2 スポットインスタンス
- Amazon ElastiCache による大規模なパフォーマンス
- Redis を使用したデータベースキャッシュ戦略
- Amazon Virtual Private Cloud の接続オプション
- ベストプラクティスの設計パターン: Amazon S3 のパフォーマンスの最適化

関連するベンチマーク:

- Amazon EBS ボリュームのベンチマーク
- Amazon EC@ インスタンスのネットワーク帯域幅

関連ツール:

- Unreal Engine: コンテンツのテストと最適化
- Unity プロファイラー
- 3D エンジン (O3DE) クォータシステムを開く
- Amazon GameLift サーバーのモニタリング
- Amazon GameLift テストツールキット

関連動画:

- AWS re:Invent 2019: [REPEAT 2] Amazon EC2 基盤 (CMP211-R2)
- AWS re:Invent 2019: Powering next-gen Amazon EC2: Deep dive into the Nitro system (CMP303-R2)
- Amazon GameLift FleetIQ の開始方法 – AWS オンライン Tech Talks

- AWS を使用したゲームの改善に関する暴動ゲームの Zach 分割
- AWS re:Invent 2023 - AWS Graviton: AWS ワークロードに最適な価格パフォーマンス (CMP334)

関連するトレーニング:

- でのゲームサーバーのホスティング AWS
- ゲームサーバーでの Amazon GameLiftFleetIQ の使用
- ゲーム AWS の の開始方法 – パート I
- でのゲームサーバーのホスティング AWS
- AWS re:Invent 2023 – AWS Graviton: AWS ワークロードに最適な価格パフォーマンス (CMP334)

コスト最適化

コスト最適化の柱には、ライフサイクル全体にわたるシステムの改良と改善の継続的なプロセスが含まれます。このプロセスは、最初の概念実証の初期設計から本番ワークロードの継続的な運用まで多岐にわたります。このホワイトペーパーのプラクティスの概要を採用することで、希望するビジネス成果を最低価格で達成するコスト対応システムを構築して運用できます。これらのコスト最適化プラクティスを実装することで、ビジネスはクラウド投資の価値を最大化できます。

ゲームは、プレイヤーの注意力とプレイタイムのために競争する必要があるユニークなクリエイティブプロジェクトです。起動前は、ゲーム開発者がゲームの人気や存続期間を明確に理解していないことがよくあります。ゲームの収益化戦略、ビジネスの優先順位、ライフサイクルステージに応じて、デベロッパーはコスト最適化の決定を評価する際にトレードオフを行う必要があります。

例えば、期待の厳しい新しいゲームの発売前段階では、通常、speed-to-market、機能開発、パフォーマンスに重点が置かれています。優先順位は、ピーク時のプレイヤー需要に合わせてインフラストラクチャがスケーリングできることを確認することです。逆に、ゲームが成功しなかったり、開発が遅くなったりすると、既存のプレイヤーのためにゲームを運用し続けるために、可能な限りコストを削減することに焦点が移る可能性があります。

多くのゲーム開発者も複数のゲームを同時に運用するため、追加の考慮事項が必要です。インフラストラクチャ、ソフトウェア、スタッフなどのリソースは複数のライブゲーム間で共有できるため、あるゲームからの損失を別のゲームからの利益で相殺できます。このシナリオでは、コスト最適化に重点を置くことで、ゲームポートフォリオ全体の財務を改善できます。

ゲームのユニークなビジネスモデル、スケール、予測不能性を考慮すると、以下の重要な質問がコスト最適化の決定に役立ちます。

- プレイヤー、システム、ゲーム機能あたりのインフラストラクチャコストを測定するにはどうすればよいですか?
- ゲームの現在のライフサイクルステージにおけるコスト最適化とプレイヤーエクスペリエンスの適切なバランスは何ですか?
- AWS リソースに適した料金モデルを使用して投資収益率を最大化するにはどうすればよいですか?

これらのベストプラクティスを適用し、適切な質問をすることで、ゲーム開発者がコストを最小限に抑えながらビジネス成果を達成するコスト対応システムを構築して運用するのに役立ちます。

焦点

設計原則

Well-Architected フレームワークのコスト最適化の柱の設計原則に加えて、次の設計原則は、クラウドでゲームワークロードを実行するコストを最適化します。

- プレイヤー、システム、ゲーム機能あたりのインフラストラクチャコストを測定する: ゲームシステム全体の特定のプレイヤーエクスペリエンスと機能に必要なインフラストラクチャコストを理解して追跡します。これにより、コスト最適化が必要なアーキテクチャの領域を特定できます。
- コスト最適化とプレイヤーエクスペリエンスのトレードオフを評価する: ゲームがどのステージにあるかを評価して、プレイヤーエクスペリエンスまたはコスト最適化の適切な焦点を決定します。通常、ゲームがクリティカルな質量に達し、プレイヤーの人口が安定したら、運用コストの最適化に集中します。重要なのは、優れたプレイヤーエクスペリエンスを提供することと、最も費用対効果の高い方法でゲームインフラストラクチャを実行することのバランスを取ることです。これらの設計原則を実装することで、ゲーム投資に対するリターンを最大化できます。

クラウド財務管理を実践する

Games Lens に固有の Cloud Financial Management のベストプラクティスはありません。クラウド財務管理のガイダンスについては、「Well-Architected フレームワークのコスト最適化の柱」を参照してください。

経費支出と使用量の認識

GAMECOST01: ゲーム環境のコストはどのように測定しますか?

プレイヤー、ゲーム機能、環境あたりのコストを理解して、時間の経過とともにプレイヤーの数が変化し、機能が追加され、向上するにつれて支出を管理および予測できるようにします。さまざまなゲーム環境のコストを管理するには、次のベストプラクティスを検討してください。

ベストプラクティス

- GAMECOST01-BP01 プレイヤー、ゲーム機能、環境あたりのコストの属性を実装する
- GAMECOST01-BP02 最適化の機会を見つける

GAMECOST01-BP01 プレイヤー、ゲーム機能、環境あたりのコストの属性を実装する

ゲームサーバーのコスト属性は通常、ゲームバックエンドサービスよりも簡単に実行できます。これは、ゲームサーバーは通常、インスタンスの実行コスト全体で償却できるインスタンスごとに特定の数の同時プレイヤーをホストできるように最適化されているためです。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

ゲームバックエンドサービスでは、ゲームのコンポーネントを個別の論理リソースまたは物理リソースとして管理できる個別の機能に分離して、コストを簡単に分析することをお勧めします。

例えば、ゲームバックエンドサービスをホストするために単一のモノリシックアプリケーションを実装するのは簡単ですが、このパターンでは、リソースのコンピューティング、ネットワーク、ストレージのコストが機能間で共有されるため、機能を追加するにつれて、プレイヤーとゲーム機能の合計コストを経時的に導き出すのが難しくなります。Amazon Amazon API Gateway やコンピューティング、メッセージング AWS Lambda AWS Fargate には Amazon Amazon SQS や Amazon Amazon SNSには Amazon S3、データベースストレージには Amazon DynamoDB などのサービスを使用して、ゲームバックエンドサービスにサーバーレスアーキテクチャを採用することを検討してください。これらのサービスは、コストを詳細に視覚化できるように、使用量ベースで主にリクエスト量に基づく料金を提供する製品のほんの一例です。Lambda 関数、Fargate サービス、DynamoDB テーブル、S3 バケットなどの個々のリソースをコスト配分タグに関連付けることで、これらのサービスのコストをゲーム機能名で帰属させ、各サービスのコストを簡単に把握できます。

また、異なる環境のコストを属性化できるように、各ゲーム開発環境を個別に管理することをお勧めします。通常、ゲーム開発者は、このゲーム業界レンズの運用の柱で説明されているように、開発、テスト、ステージング、本番環境用に個別の環境を管理します。通常、環境ごとにスケーラビリティ、パフォーマンス、使用要件が異なり、個別のチームによって管理される場合があります。コス

トを制御するには、これらの環境を整理して、各環境のコストを適切にモニタリングして属性付けできるようにします。

詳細については、次のドキュメントを参照してください。

- スケーリングするサーバーレスマルチプレイヤーゲームの構築
- WebSockets ベースのバックエンドを備えたスタンドアロンゲームセッションサーバー
- サーバーレスバックエンドを備えたスタンドアロンゲームセッションサーバー

実装手順

- Amazon API Gateway や などのサーバーレスまたはコンテナ化されたアーキテクチャを使用して AWS Lambda、ゲームバックエンドサービスを個別の機能にデカップリング AWS Fargate し、機能ごとにきめ細かなコスト属性を有効にします。
- コスト配分タグを個々のリソース (Lambda 関数、DynamoDB テーブル、S3 バケットなど) に適用して、コストを特定のゲーム機能に関連付けることで、コスト分析を改善します。
- 開発、テスト、ステージング、本番環境用に個別の環境を管理し、スケーラビリティと使用要件に合わせてコストを個別に整理およびモニタリングします。

GAMECOST01-BP02 最適化の機会を見つける

ゲーム開発者やパブリッシャーは、 AWS FinOps プラクティスを使用してクラウドコストを最適化し、クラウド支出をよりよく可視化できます。これにより、ゲームプロデューサーは、プレイヤーのインフラストラクチャを維持するために必要な平均コストを、ゲームによって提供される財務結果に合わせることができます。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

AWS は、クラウドサービスの支出を管理および最適化するための Cloud Financial Management 向けのすぐに使用できるソリューションガイダンスを提供します。この機能には、詳細な可視性とコストと使用状況の分析が含まれており、支出ダッシュボード、最適化、支出制限、チャージバック、異常検出と対応などのトピックの意思決定をサポートします。Cloud Financial Management のソリューションガイダンスには、予算と予測機能が含まれており、ワークロード用に定義されたコスト最適化アーキテクチャが提供されるため、チームに関連する適切な料金モデルを選択し、リソースコストを属性化できます。これにより、環境とリソース全体で追跡、通知、コスト最適化の手法がアク

ティブ化されます。経費情報を一元管理し、ターゲットを絞った可視性と意思決定をサポートするために、必要に応じて重要な利害関係者にアクセスを許可できます。

もう 1 つの主要な FinOps ツールは Cost Optimization Hub です。Cost Optimization Hub は、 AWS アカウント と 全体のコスト最適化のレコメンデーションと機会の一元的なビューを提供するため AWS リージョン、 AWS 支出を最大限に活用できます。Cost Optimization Hub を使用して、 AWS アカウント と 全体の AWS コスト最適化レコメンデーションを特定、フィルタリング、集計できます AWS リージョン。リソースの適切なサイジング、アイドル状態のリソースの削除、Savings Plans、リザーブドインスタンスに関する推奨事項を作成します。単一のダッシュボードを使用すると、コスト最適化の機会を特定するために複数の AWS 製品にアクセスする必要がなくなります。

ゲームチームが共有 AWS アカウント myApplications in AWS マネジメントコンソール Home を使用している場合、 を使用して個々のワークロードのアプリケーションリソースコストを表示できます。この詳細なビューでは、ゲームインフラストラクチャ内の特定のコスト傾向を特定できるため、リソースの割り当てと最適化について情報に基づいた意思決定を行うことができます。

さらに、AWS データエクスポートを使用して請求データとコスト管理データを定期的に確認することで、隠れたコスト削減の機会を発見できます。この詳細なレポートでは、クラウド支出を包括的に把握できるため、リソースの過剰消費や使用率の低い領域を特定し、より費用対効果の高いサービスや料金モデルを活用する機会を特定できます。

FinOps の原則を採用し、 が提供するツールを活用することで AWS、ゲーム開発者やパブリッシャーはクラウドリソースを最も効率的に活用し、最終的に収益を向上させ、さらなるゲーム開発とイノベーションのための資金を解放できます。

実装手順

- AWS クラウド 財務管理ツールを使用して、きめ細かな可視性、支出ダッシュボード、異常検出、コスト配分を実現し、クラウド支出を効果的に最適化および追跡できます。
- Cost Optimization Hub を使用して、 AWS アカウント および リージョン間で適切なサイズ設定、Savings Plans、リザーブドインスタンスのレコメンデーションを一元化します。
- Data Exports と MyApplication on を使用して AWS 請求データを定期的に確認し AWS 、ワークロード固有のコストの分析、コスト削減の機会の発見、リソース割り当ての最適化に役立てます。

費用対効果の高いリソース

GAMECOST02: ゲームサーバーに適したコンピューティングソリューションをどのように選択していますか?

ゲームワークロードの最もユニークな側面の 1 つは、他のタイプのワークロードと比較して、ゲームサーバーです。プレイヤーはゲームクライアントから接続してゲームセッションをプレイするため、ゲームサーバーはプレイヤーエクスペリエンスにとって重要です。

ゲームサーバーは、マルチプレイヤーゲームを運用するためのコストの最大の推進要因の 1 つでもあります。したがって、ゲームサーバーのコンピューティングインフラストラクチャをどのように活用するかを最適化してコストを削減することが重要です。

ベストプラクティス

- GAMECOST02-BP01 インターネット経由のデータ転送コストを最適化する
- GAMECOST02-BP02 各ゲームサーバーインスタンスでホストされているゲームセッションの数を最適化してコストを最適化する
- GAMECOST02-BP03 コストを削減するために適切なコンピューティング料金オプションを選択する

GAMECOST02-BP01 インターネット経由のデータ転送コストを最適化する

AWS 主に AWS リソースからインターネットへのアウトバウンド (エグレス) データ転送に課金されますが、ゲーム企業は AWS Direct Connect または AWS Gateway ロードバランサーを介したデータ転送に関連するコストが高くなり、インバウンド (イングレス) データとアウトバウンドデータの両方に課金される可能性があります。ゲームの AWS バックエンドからプレイヤーにデータを転送する全体的なコストを削減するソリューションを実装します。 AWS リソースからのエグレス料金を最小限に抑え、 AWS 接続サービスを通じてイングレス料金とエグレス料金を管理するためのオプションの評価に焦点を当てます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

Amazon CloudFront を使用して、コンテンツ配信と一般向けウェブアプリケーションのコストを削減します。

クラウドに保存されているゲームコンテンツとアセットは通常、Amazon S3 に保存され、S3 から直接、または Amazon S3 からコンテンツを取得し、クライアントに配信する Amazon EC2 Amazon S3でホストされているウェブサーバーからゲームクライアントに配信されます。 Amazon EC2 コンテンツダウンロードのデータ転送コストを削減するには、クラウドストレージの前に Amazon CloudFront を使用してユーザーにコンテンツを配信することを検討してください。

CloudFront を使用すると、CloudFront points-of-presenceからコンテンツを配信するコストがリージョンから直接配信するよりも安く、CloudFront は Amazon EC2 や Amazon S3 などの AWSベースオリジンのオリジン取得料金を請求しないため、データ転送コストを削減できます。コンテンツが静的で頻繁に変更されない場合は、CloudFront を使用してエンドユーザーに近いデータをキャッシュできるため、コストをさらに削減できます。

また、CloudFront は、キャッシュが使用されていない場合でも、前面のパブリックウェブアプリケーションとサービスのコスト効率を向上させます。これは、 AWS ネットワーク経由でトラフィックをルーティングすることで、サーバーとクライアント間のデータ転送コストを削減できるためです。

Amazon CloudWatch を使用して、Amazon CloudFront の使用状況をモニタリングできます。複数のコンテンツ配信ネットワーク (CDNs) を使用するユースケースの場合、Amazon CloudFront Origin Shield は追加のキャッシュレイヤーを提供して、異なるプロバイダーからのオリジンリクエストを統合して削減できます。

ゲームネットワークトラフィックを理解するために、VPC フローログと Amazon CloudWatch Internet Monitor を有効にして、end-to-endで可視化できます。このアプローチにより、データ転送コストが高い原因を特定し、アーキテクチャの変更を実行してデータ転送コストを最適化できます。

実装手順

- Amazon S3 または EC2 ベースのコンテンツオリジンの前に Amazon CloudFront を使用すると、CloudFront points-of-presenceからの低コスト配信を活用し、オリジン取得料金を削除することで、データ転送コストを削減できます。 EC2-based
- VPC フローログと Amazon CloudWatch Internet Monitor を有効にして、ネットワークトラフィックを分析し、アーキテクチャの変更を特定してデータ転送コストを最適化します。
- CloudFront Origin Shield を実装して、コスト効率を高めるために複数の CDNsのオリジンリクエストを統合し、削減します。

コンテンツ配信のベストプラクティスの詳細については、「ゲームのコンテンツ配信」ホワイトペーパーを参照してください。

GAMECOST02-BP02 各ゲームサーバーインスタンスでホストされているゲームセッションの数を最適化してコストを最適化する

サーバーインスタンスごとにホストされるゲームセッションの数を最適化して、コンピューティング使用率を向上させ、コンピューティングインフラストラクチャコストを削減します。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

コストを最適化するために、ゲーム開発者は同じ物理サーバーまたは仮想サーバーでホストされているゲームセッションの数を最大化する必要があります。これは、ゲームサーバーのパッキング密度とも呼ばれます。これは、インスタンスで同時にホストできるゲームサーバープロセスの数を増やすことで実現されます。

通常、1 つのゲームサーバープロセスで EC2 インスタンスで使用できるリソース全体を使用する必要はありません。これはゲームのコンピューティングコストを削減するための最も重要な方法の 1 つであり、別々のポートで EC2 インスタンスで複数のサーバープロセスをスポーンおよび管理できるソフトウェアを使用する必要があります。

例えば、Amazon GameLift にはインスタンスあたりのゲームサーバープロセスの最大数に対するクォータがあり、ホスティングコストを削減できるように活用する必要があります。詳細については、インスタンスあたりの最大ゲームサーバープロセスの現在のクォータの詳細については、「Amazon GameLift Servers エンドポイントとクォータ」を参照してください。

EC2 インスタンスなどの仮想マシンにゲームサーバープロセスをデプロイする代わりに、ゲーム開発者はコンテナオーケストレーションソリューションを使用してコンテナベースのアプリケーションとしてゲームサーバーを実行することが人気となっています。ゲーム開発者は、Amazon Elastic Container Service (Amazon ECS) または Amazon EKS での Agones と Open Match を使用したゲームサーバーホスティングのガイダンスを使用できます。もう 1 つのオプションは、ECS と EKS の両方で動作するサーバーレスコンピューティングエンジンである Game Server Hosting on AWS Fargate です。これにより、基盤となるインフラストラクチャを管理することなくゲームに集中できます。

コンテナソリューションは、指定したリソース要件やその他のプレイスメントロジックに基づいて、クラスター内の使用可能なコンテナインスタンスを自動的に検索してゲームサーバーコンテナをホス

トできるジョブスケジューリング機能を提供します。ただし、アクティブなプレイヤーセッションを中断しないようにスケーリングとプレイヤー配置の動作を管理する方法を考慮することが重要です。

実装手順

- 個別のポートとプロセス管理ソフトウェアを使用して EC2 インスタンスごとに複数のゲームサーバープロセスを実行することで、パッキング密度を向上させます。
- Amazon GameLift または ECS、EKS、 などのコンテナソリューション AWS Fargate を使用して、ゲームサーバープロセスを効率的に管理し、インフラストラクチャコストを削減します。
- リソース使用率を継続的にモニタリングして、プレイヤーのエクスペリエンスを損なうことなくパッキング密度を改良し、コスト効率を維持します。

GAMECOST02-BP03 コストを削減するために適切なコンピューティング料金オプションを選択する

ゲームサーバーソフトウェアのパフォーマンステストをさまざまなインスタンスタイプとコンピューティングオプションで実行し、どのオプションがゲームに最も費用対効果が高いかを判断します。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

ワークロードに適した EC2 インスタンスタイプを効率的に活用することに加えて、コスト最適化の目標に最適なコンピューティング料金オプションを検討してください。オンデマンドインスタンス、スポットインスタンス、リザーブドインスタンス、Savings Plans など、複数の料金オプションを使用できます。

Savings Plans (SPs) は、使用量のコミットメントを行うことでコンピューティングに割引を提供し、1 年または 3 年間の予想される使用量を予測できないシナリオに最適です。リザーブドインスタンスなどの割引を提供し、リージョン、インスタンスファミリー、オペレーティングシステム、テナンシーにこれらの割引を柔軟に適用できます。また、カジュアルなゲームのゲームサーバーホスティングオプションとして使用できるユーザーや AWS Fargate、ゲームサーバーを必要としないターンベースのゲームの優れたオプションとして使用される AWS Lambda ユーザーにも適用できます。詳細については、「スケールするサーバーレスマルチプレイヤーゲームの構築」を参照してください。

Savings Plans は、ゲームの起動中に導入され、ゲームが対象者にリリースされたときに EC2 インスタンスの支出に寄与しているゲームサーバーのワークロードのコストを削減します。Savings Plans は、ゲーム運用チームがゲームが本番環境で長時間実行された後にプレイヤートラフィックをよりよく理解している場合に、起動後に導入することもできます。

Savings Plans はリージョンの柔軟性を提供するため、地理的にまたがって予測不可能な使用状況でゲームサーバーの支出を最適化するのに特に理想的です。

たとえば、毎日のプレイヤー使用パターンでプレイヤーベースをサポートするために少なくとも 20 台のサーバーが必要で、定期的に最大 40 台のサーバーが必要な場合は、Savings Plan コミットメントを購入して 20 台のサーバーのベースラインをカバーすることを検討してください。これは、その使用需要が予測可能で一貫性があり、購入した使用コミットメントの最大使用率になるためです。

Savings Plans の使用率を最大化し、オンデマンドやスポットインスタンスなど、予測不可能なゲームサーバーの使用量の急増に対する柔軟性を高める他の購入オプションで拡張して、最適なコスト削減を実現します。

スポットインスタンスは、最大のコンピューティング割引を提供し、使用量のコミットメントを必要とせず、予測不可能なワークロードタイプや急増するワークロードタイプの柔軟性を提供するため、ゲームサーバーの実行に最適です。ただし、スポットインスタンスは中断される可能性があるため、ゲームセッション時間が短いゲームサーバーワークロードや、中断の許容度が高い状況に最適です。

EC2 スポットインスタンスで Amazon EKS で Kubernetes を使用してゲームサーバーを実行するためのガイダンスの詳細については、「Aurora Serverless を使用して EC2 スポットで超マルチプレイヤーゲームを実行する方法」を参照してください。

Amazon EC2 スポットインスタンスを使用して、中断の可能性が最も低いプールを決定し、オンデマンド料金と比較して最大限の節約を実現します。

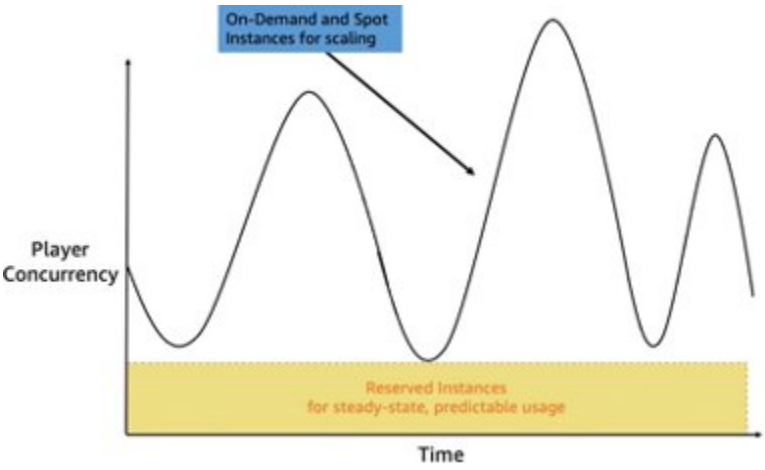
スポットを使用する場合は、 の複数の EC2 インスタンスタイプとアベイラビリティーゾーンでゲームサーバーワークロードを実行して、容量の使用量 AWS リージョン を多様化し、中断リスクを軽減することもお勧めします。

スポットインスタンスをオンデマンドインスタンスと組み合わせて使用して、アクティブなゲームセッションへの潜在的な中断の影響を最小限に抑え、容量最適化配分戦略を使用して中断のリスクをさらに軽減することを検討してください。

その他のベストプラクティスについては、Amazon EC2 スポットのベストプラクティス」を参照してください。リスクのあるスポットインスタンスを置き換えるための Auto Scaling でのキャパシティの再調整を使用すると、スポットインスタンスの中断リスクが高い場合に、プロアクティブにモニタリングしてキャパシティを追加できます。

Amazon GameLift FleetIQ はスポットインスタンスと統合して、中断のリスクを減らしながら、低コストのスポットインスタンスの使用を最適化します。GameLift を使用してゲームをホストしている場合は、GameLift ドキュメントでコンピューティングリソースを選択してください。詳細については、「マネージドフリートのコンピューティングリソースを選択する」を参照してください。

次の図は、ゲームサーバーワークロードに複数のコンピューティング料金オプションを使用する例を示しています。



複数の EC2 料金オプションを使用したゲームサーバーのホスティング

この図では、プレイヤーの同時実行数は時間の経過とともに変動するため、使用率の管理やコストの最適化が困難になります。この変動に対処するには、EC2 の Savings Plans for EC2 を使用して最小使用要件のニーズを満たすと同時に、EC2 オンデマンドインスタンスと EC2 スポットインスタンスを使用してプレイヤーの需要のニーズを満たす、さまざまなコンピューティング料金オプションを組み合わせて採用することを検討してください。 EC2

実装手順

- Savings Plans を使用して、予測可能なベースライン使用量を実現し、スポットインスタンスとオンデマンドインスタンスを組み合わせて、使用量の急増時の柔軟性とコスト最適化を実現します。
- セッション時間が短いか、中断耐性が高いゲームサーバーにはスポットインスタンスを使用し、インスタンスタイプとアベイラビリティーゾーンを分散してリスクを最小限に抑えます。
- EC2 スポットインスタンスアドバイザー、キャパシティの再調整、GameLift FleetIQ などのツールを実装して、スポットインスタンスの使用を最適化し、中断をプロアクティブに管理します。

データ転送コスト

GAMECOST03: ゲームインフラストラクチャのデータ転送コストをどのように最適化していますか?

ゲームは、プレイヤーのゲームクライアントデバイスとゲームインフラストラクチャ間でインターネット経由で大量のデータを転送し、ゲームプレイエクスペリエンスを提供するだけでなく、ゲームインフラストラクチャのコンポーネント間でもデータを転送できます。

例えば、データ転送は、プレイヤーがゲームコンテンツの更新をゲームクライアントにダウンロードし、ゲームの進行状況をクラウドに保存し、友人とリアルタイムのマルチプレイヤーゲームセッションに参加し、ゲームインフラストラクチャがリージョンとアベイラビリティーゾーン間でデータを転送したときに発生します。ゲームワークロードでデータ転送が発生する場所を理解し、アーキテクチャの選択を最適化して、このデータ転送コストを削減することが重要です。

ゲームワークロードのデータ転送コストを最適化するには、次のベストプラクティスを検討してください。

ベストプラクティス

- GAMECOST03-BP01 ユーザー生成コンテンツに適したタイプのストレージを選択してコストを削減する
- GAMECOST03-BP02 ゲームバックエンドのデータベースを最適化する

GAMECOST03-BP01 ユーザー生成コンテンツに適したタイプのストレージを選択してコストを削減する

ゲームで生成および保存される各タイプのデータには、ワークロードに適したストレージソリューションを決定する際に考慮すべき固有の特性があります。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

Amazon S3 Object Lifecycle Management を使用して、最も費用対効果の高いストレージクラスにオブジェクトデータを保存します。Amazon S3 には複数のストレージクラスとオブジェクトライフサイクル管理が用意されているため、シンプルできめ細かなポリシーを簡単に設定して、ストレージ層間でデータを自動的に移行してコストを削減できます。デフォルトでは S3 標準ストレージクラスにデータを保存するのではなく、階層間でデータを自動的に経時的に移行するようにライフサイクル設定を設定するか、S3 Intelligent-Tiering ストレージクラスを使用してアクセスパターンを不明または変更することを検討してください。

または、S3 Intelligent-Tiering は、階層間でデータを費用対効果の高い方法で自動的に移行でき、ライフサイクルポリシーを手動で設定しなくてもコストを最適化できるため、デフォルトのストレージクラスとして推奨され、現在は小規模オブジェクトや存続期間の短いオブジェクトに最適です。詳細

については、Amazon S3 Intelligent-Tiering – Improved Cost Optimizations for Short-Lived and Small Objects」を参照してください。

Amazon S3 の一般的なユースケースには、ゲームアセットのストレージ、静的コンテンツ、ゲームログ、データレイクストレージ、バックアップなどがあります。開発中に共有ファイルシステムをワークステーションにアタッチするなど、ファイルシステムが必要なユースケースでは、Amazon Elastic File System (Amazon EFS) の使用を検討してください。Amazon Elastic File System EFSは、さまざまなストレージクラスを提供し、インフラストラクチャを管理することなくファイルを追加および削除すると自動的に拡張および縮小します。

Amazon S3 One Zone-IA は、必要に応じて再作成できるゲーム内セッション、マッチメーキング、またはその他のエフェメラル情報に関連する一時的なデータに最適なストレージオプションです。このタイプのゲームデータでは、複数のアベイラビリティーゾーン (AZs) 間で冗長性は必要ありません。この低コストのストレージクラスは、分析やデバッグに使用されるプレイヤーアクション、ゲームイベント、その他のテレメトリデータのレコードに適しています。

このようなゲームデータに S3 Express One Zone を使用する主なコスト最適化の利点は、標準の S3 ストレージクラスと比較して大幅なコスト削減と、ストレージコストの最大 20% の削減です。これは、ミッションクリティカルなアプリケーションデータと同じレベルの耐久性と可用性を必要としない大量のデータを持つゲームに特に有利です。S3 One Zone を活用することで、ゲーム開発者やパブリッシャーはプレイヤーエクスペリエンス全体を損なうことなくクラウドストレージコストを最適化できます。

実装手順

- ストレージクラス間でデータを移行するように Amazon S3 ライフサイクルポリシーを設定するか、アクセスパターンの変化に伴う自動コスト最適化のデフォルトとして S3 Intelligent-Tiering を使用します。
- テレメトリやマッチメーキングレコードなどの一時的なゲームセッションデータに S3 1 ゾーン低頻度アクセスを使用すると、十分な可用性を維持しながらストレージコストを最大 20% 削減できます。
- 開発中の共有ファイルシステムのニーズについては、Amazon EFS を使用して、伸縮自在な容量と複数のストレージクラスでストレージ管理を簡素化します。

GAMECOST03-BP02 ゲームバックエンドのデータベースを最適化する

ゲームは、プレイヤープロファイルやインベントリからゲーム内のマイクロトランザクションや進行状況メトリクスまで、さまざまな重要なデータを保存するためにデータベースに大きく依存していま

す。データベースは、プレイヤーグループ、パーティーの作成と維持、モデレーションポリシーの適用など、ゲームの社会的側面を管理する上でも重要な役割を果たします。ゲームのプレイヤーベースが大きくなるにつれて、データや使用状況の需要の増加に対応するために、関連するデータベースコストは必ず増加します。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

Amazon Aurora で実行されているゲームバックエンドには、いくつかのコスト最適化戦略を使用できます。主な推奨事項の 1 つは、使用パターンに基づいてリードレプリカを自動スケーリングし、トラフィックの変動を処理するためにレプリカの数を動的にスケールアップまたはスケールダウンすることです。つまり、本当に必要なリソースに対して料金を支払うことになります。もう 1 つの最適化戦略は、ゲーム分析に使用されるリードレプリカを Amazon S3 への DB スナップショットのエクスポートに置き換えることです。これは、S3 ストレージサービスがプロビジョニングされた Aurora データベースインスタンスよりも一般的に手頃な価格であるためです。詳細については、「Amazon RDS の Amazon S3 への DB スナップショットデータのエクスポート」を参照してください。

Amazon Aurora のリザーブド DB インスタンスをコアデータベースインスタンスに使用し、Aurora Serverless 設定に移行すると、リソース使用率をより柔軟にきめ細かく制御できるため、大幅な長期コスト削減にもつながります。

同様に、Amazon DynamoDB を使用するゲームバックエンドの場合、DynamoDB オンデマンドキャパシティモードを採用すると、特に新しいワークロードや予測不可能なワークロードでは、過剰プロビジョニングを必要とせずに消費したリソースに対してのみ料金を支払うことができるため、効果的な選択肢になります。ゲームトラフィックパターンが時間の経過とともに安定し、予測可能になるにつれて、DynamoDB プロビジョンドキャパシティモードに移行できます。これにより、キャパシティプランニングを改善することでコスト削減を実現できます。DynamoDB テーブルで自動スケーリングを有効にすることは、もう 1 つの重要な最適化です。これにより、サービスはトラフィックの変動に基づいてプロビジョニングされた容量を動的に調整できます。起動前に開発環境でゲームのデータ構造をテストして、不要なローカルセカンダリインデックス (LSIs) とグローバルセカンダリインデックス (GSIs) を見つけて削除します。これにより、ゲームデータストレージとオペレーションの大幅なコスト削減につながる可能性があります。ゲームバックエンドコードから非効率的なスキャンオペレーションを削除して、よりターゲットを絞ったクエリを優先し、Amazon DynamoDB リザーブドキャパシティを購入し、AWS Lambda トリガーで DynamoDB Streams を活用してゲームバックエンドイベントを処理することで、DynamoDB コストをさらに最適化できます。詳細については、DynamoDB でのデータのクエリとスキャンのベストプラクティス」を参照してください。

Amazon Aurora と DynamoDB の両方にこれらのコスト最適化戦略を実装することで、ゲーム開発者とパブリッシャーはゲームバックエンドデータベースの支出を大幅に削減できます。

実装手順

- Aurora リードレプリカの自動スケーリングと DB スナップショットの Amazon S3 へのエクスポートを使用して、変動するトラフィックと分析のニーズをコスト効率良く処理します。
- DynamoDB のコストを最適化するには、新しいワークロードのオンデマンドキャパシティから始め、予測可能なトラフィックの自動スケーリングを使用してプロビジョニングされたキャパシティに移行し、未使用の LSIs と GSIs を削除します。
- ターゲットクエリを優先して非効率的なスキャン操作を避け、リザーブドインスタンスまたはリザーブドキャパシティーを使用し、イベント処理 AWS Lambda に DynamoDB Streams を と共に使用します。

需要と供給リソースの管理

Games Lens に固有の需要と供給リソースのベストプラクティスはありません。

需要の管理とリソースの提供の詳細については、「Cost Optimization Pillar - AWS Well-Architected Framework」を参照してください。

時間の経過に伴う最適化

Games Lens に固有の経時的な最適化のベストプラクティスはありません。

時間の経過に伴うコストの最適化の詳細については、「Cost Optimization Pillar - AWS Well-Architected Framework」を参照してください。

リソース

コスト最適化のベストプラクティスの詳細については、以下のリソースを参照してください。

関連ドキュメント:

- Amazon VPC の NAT ゲートウェイのデータ転送料金を減らすにはどうすればよいですか?
- Agones 用の Amazon GameLift FleetIQ アダプターの紹介
- Amazon VPC の NAT ゲートウェイトラフィックの上位寄与要因を見つけるにはどうすればよいですか?

- グローバルゲームサーバーに適したコンピューティング戦略を選択する
- AWS Well-Architected ラボ – コスト効率の高いリソース
- Amazon VPC CNI プラグインがノードあたりのポッド数の制限を増やす
- コスト最適化のためのアーキテクチャのベストプラクティス
- Amazon SageMaker AI RL と Amazon EKS を使用したプレイヤーの待機時間の短縮とコンピューティング割り当ての適切なサイジング
- AWS Compute Optimizer
- Electronic Arts は、Amazon S3 Intelligent-Tiering と Amazon Glacier を使用してストレージコストとオペレーションを最適化します
- Windows ワークロードを Linux に移行することで、分かりにくいライセンスプラクティスをエスケープする
- 一般的なアーキテクチャでのデータ転送コストの概要
- AWS と Kubecost が共同で EKS 顧客にコストモニタリングを提供
- 基盤の構築: コスト最適化のための環境のセットアップ
- Amazon EC2 スポットインスタンスの概要

持続可能性

持続可能性の柱は、 AWS ワークロードの持続可能性目標を達成するのに役立つ設計原則、運用ガイダンス、ベストプラクティス、改善計画を提供します。

実装に関する実装ガイダンスについては、持続可能性の柱 - AWS Well-Architected Framework ホワイトペーパーを参照してください。

焦点

- 設計原則
- リージョンの選択
- 需要に合わせた調整
- ソフトウェアとアーキテクチャ
- データ管理
- ハードウェアとサービス
- リソース

設計原則

ゲーム業界の持続可能性は、世界中のさまざまなリージョンでさまざまな割合で進化しています。北米の大規模な地域における持続可能な電力と、欧州と英国における持続可能性の義務により、アーキテクトは近い将来、よりグリーンなワークロードを達成するためのいくつかのアプローチを持っています。Well-Architected の持続可能性の柱を使用して、さまざまなワークロードに対してこれらの対策を達成できます。

レンズのこのセクションでは、ゲームワークロードに使用できるいくつかのベストプラクティスについて説明します。

- ゲームのユーザーデータに適したストレージタイプを選択します。
- データを重複排除し、不要なデータをワークロードから削除するデータライフサイクルポリシーに注意してください。
- コンピューティングリソースのデプロイを適切なサイズにすることを選択します。
- 短いプロセスとトランザクションプロセスにはサーバーレスを使用します。

リージョンの選択

Games Lens に固有のリージョン選択のベストプラクティスはありません。詳細については、「持続可能性の柱 - AWS Well-Architected フレームワーク」を参照してください。

需要に合わせた調整

Games Lens に固有のベストプラクティスを要求する整合性はありません。詳細については、「持続可能性の柱 - AWS Well-Architected フレームワーク」を参照してください。

ソフトウェアとアーキテクチャ

Games Lens に固有のソフトウェアとアーキテクチャのベストプラクティスはありません。詳細については、「持続可能性の柱 - AWS Well-Architected フレームワーク」を参照してください。

データ管理

GAMESUS01: ゲームシステムのユーザーデータとゲームデータはどのように管理しますか?

重要な履歴データのみを保持することで、データストレージと関連性を最適化するデータライフサイクル戦略を策定します。

ベストプラクティス

- GAMESUS01-BP01 ユーザーコンテンツ、サブスクライバー情報、ゲーム内購入に合わせたパターンに適合するストレージテクノロジーを使用する
- GAMESUS01-BP02 ライフサイクルポリシーまたは TTL の有効期限を使用して、不要なゲームユーザーデータ、ログファイル、または廃止されたアセットを削除する

GAMESUS01-BP01 ユーザーコンテンツ、サブスクライバー情報、ゲーム内購入に合わせたパターンに適合するストレージテクノロジーを使用する

データをタイプ、保持の必要性、アクセス頻度で分類する必要があります。これにより、ゲームまたはバックエンドサービスが生成する無数のデータ型に最も最適化されたストレージソリューションを選択できます。急速に変化するデータは、キー値またはインメモリデータベースサービスに保存する

必要があります。トランザクションデータはリレーショナルデータベースサービスに保存する必要があります。大きなファイル、ゲームアセット、またはユーザーが生成したコンテンツは、オブジェクトストレージサービスに保存する必要があります。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

ゲームは、アクセス頻度、レイテンシー、コストに最適化されたストレージソリューションを必要とするさまざまなデータ型を生成および消費します。保存されたデータは、タグを使用して分類し、削除できるデータや長期保存が必要なデータを区別する必要があります。

以下のサービスは、さまざまな Games ユースケースに適しています。

Amazon Aurora (MySQL および PostgreSQL と互換性あり) は、高可用性、低レイテンシー、自動スケーリングを提供するため、プレイヤーアカウントの管理と認証、ゲーム内経済、リーダーボードとプレイヤーランキング、ゲーム状態の永続性、イベントとキャンペーンの管理、マルチリージョンと高可用性のデプロイなど、大量のトランザクションデータを処理するのに最適です。

Amazon DynamoDB は、低レイテンシー、高スループット、シームレスなスケーラビリティで知られるフルマネージド NoSQL データベースです。これにより、リアルタイムのプレイヤーデータ、セッション管理、インベントリ、ゲーム内経済、リアルタイムのマルチプレイヤーゲーム状態、マッチメーキング、イベントログ記録、世界中の視聴者のスケーリングの処理に最適です。

Amazon DocumentDB (MongoDB 互換) は、スケーラブルで低レイテンシーのドキュメント指向データベースサービスを提供します。インベントリシステム、プレイヤープロファイルとカスタマイズの 、ゲームワールドと手続き的に生成されたコンテンツ、ソーシャルとプレイヤーのインタラクション、分析と動作の追跡、ゲーム内のメタデータと設定など、柔軟で半構造化されたデータの保存に最適です。

Amazon ElastiCache は Redis または Memcached によるインメモリキャッシュをサポートしており、迅速なデータアクセスと応答時間の短縮を実現します。これは、スムーズなユーザーエクスペリエンスのために速度とパフォーマンスが不可欠なリアルタイムのマルチプレイヤーゲームに不可欠です。ElastiCache は、リアルタイムのリーダーボード、セッション管理、ゲームメタデータのキャッシュ、ゲーム内のチャットとメッセージング、マッチメーキング、リアルタイムの分析とテレメトリ、トラフィックの多いイベントのスケーリングにゲームで使用されます。

Amazon Simple Storage Service (S3) を使用して、ゲームアセット、ビデオ、写真、テキストログファイルなどのオブジェクトを保存できます。S3 は、業界をリードするスケーラビリティ、データ可用性、セキュリティ、パフォーマンスを提供するオブジェクトストレージサービスです。

頻繁かつ低頻度のデータアクセスをサポートする複数のストレージクラスと、費用対効果の高いアーカイブストレージを提供している場合。開発中に頻繁にアクセスされるデータの場合、スタジオは低レイテンシーと高スループットのパフォーマンスのためにオブジェクトを S3 Standard に保存する必要があります。ホットからコールド、またはその逆に頻繁に送信されるデータの場合、スタジオは S3 Intelligent-Tiering を調査する必要があります。Intelligent-Tiering は、データのアクセスパターンをモニタリングし、データを最も費用対効果の高いアクセス階層に自動的に移動します。

高スループット、低レイテンシーを必要とし、単一のアベイラビリティーゾーンに住んでいるスタジオでは、S3 Express One Zone を使用します。これにより、データが 1 つの AZ にレプリケートされ、S3 標準と比較してデータアクセス速度が向上します。履歴データのディープアーカイブのニーズに対して、Amazon は Amazon Glacier も提供します。Amazon Glacier ストレージクラスは、データアーカイブ専用であり、クラウド内で高いパフォーマンス、取得の柔軟性、低コストのアーカイブストレージを提供します。

Amazon Elastic Block Store を使用して、ゲームサーバーまたはアセットリポジトリが機能するために必要なゲームサーバーのバイナリ、実行可能ファイル、設定を保存できます。EC2 インスタンスにアタッチされていない未使用のボリュームをスナップショットして削除する必要があります。これにより、不要なサービスやハードウェアの使用を減らしながら発生するストレージ料金が軽減されます。

実装手順

- ゲームデータをタイプ、保持ニーズ、アクセス頻度で分類し、データをタグ付けして短期ストレージ要件と長期ストレージ要件を区別します。
- トランザクションデータには Amazon Aurora、リアルタイムプレイヤーデータには DynamoDB、半構造化データには DocumentDB、タイムクリティカルなゲーム情報の低レイテンシーキャッシュには ElastiCache を使用します。
- ゲームアセット、ログ、ユーザー生成コンテンツを Amazon S3 に保存し、アクセスパターンとアーカイブのニーズに基づいて適切なストレージクラス (Intelligent-Tiering、1 ゾーン、Glacier など) を選択し、通常のスナップショット管理でゲームサーバーのバイナリと設定に EBS を使用します。

GAMESUS01-BP02 ライフサイクルポリシーまたは TTL の有効期限を使用して、不要なゲームユーザーデータ、ログファイル、または廃止されたアセットを削除する

タグとデータ型を使用してライフサイクルポリシーまたは TTL を作成し、データをアーカイブストレージに移動したり、サービスから完全に削除したりできます。これには、一時的な設定、期限切れのアーカイブ済みコンテンツ、不要になった履歴ログが含まれる場合があります。ほとんどの サービスはタグ付けをサポートしています。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

S3 に保存されているデータの場合、ライフサイクルポリシーを使用して、データを低頻度のアクセス階層とアーカイブ階層のストレージに移動できます。S3 ライフサイクル設定では、1 つのストレージクラスから別のストレージクラスにオブジェクトを移行し、ストレージのコストを節約できるルールを定義できます。オブジェクトのアクセスパターンが不明、またはアクセスパターンが時間の経過とともに変化している場合、コストを自動的に削減するためにオブジェクトを S3 Intelligent-Tiering ストレージクラスに移行できます。

Amazon S3 は、以下の図のようにストレージクラス間の移行のためのウォーターフォールモデルをサポートします。

移行アクションを S3 ライフサイクル設定に追加して、有効期間終了時にオブジェクトを削除するように Amazon S3 に指示できます。オブジェクトがライフサイクル設定に基づいて有効期間が終了すると、Amazon S3 はバケットの S3 バージョニング状態に基づいて有効期限アクションを実行します。

- バージョニングされていないバケット: Amazon S3 はオブジェクトを削除のためにキューに入れ、非同期的に削除して、オブジェクトを完全に削除します。
- バージョニングが有効なバケット: 現在のオブジェクトバージョンが削除マーカーでない場合、Amazon S3 は一意のバージョン ID を持つ削除マーカーを追加します。これにより、最新のバージョンが最新以外のバージョンとなり、削除マーカーが最新バージョンになります。
- バージョニングが停止されたバケット: Amazon S3 は、バージョン ID として null を持つ削除マーカーを作成します。この削除マーカーは、オブジェクトのバージョンをバージョン階層の null バージョン ID に置き換え、オブジェクトを効果的に削除します。
- ライフサイクル設定をバケットに追加すると、設定ルールは既存のオブジェクトとそれ以降に追加されるオブジェクトの両方に適用されます。例えば、有効期限アクションを使用してライフサイク

ル設定ルールを今日追加すると、特定のプレフィックスを持つオブジェクトが作成から 30 日後に期限切れになります。Amazon S3 は、30 日以上経過し、指定されたプレフィックスを持つ既存のオブジェクトの削除をキューに入れます。

DynamoDB の Time to Live (TTL) は、不要になった項目を削除するためのコスト効率に優れた方法です。TTL では、項目がいつ不要になるかを示す有効期限タイムスタンプを項目ごとに定義できます。DynamoDB は、書き込みスループットを消費することなく、有効期限が切れてから数日以内に期限切れの項目を自動的に削除します。

- TTL を使用するには、まずテーブルで TTL を有効にし、次に TTL の有効期限タイムスタンプを格納する特定の属性を定義します。タイムスタンプは UNIX エポック時間形式で秒単位で保存する必要があります。項目が作成または更新されるたびに、有効期限を計算して TTL 属性に保存できます。
- 有効期限切れの TTL 属性を持つ項目は、通常は有効期限が切れてから数日以内にシステムによって削除される場合があります。削除待ちの期限切れ項目は、TTL 属性の変更や削除を含め、引き続き更新できます。期限切れの項目を更新する際には、その項目がその後削除されないように条件式を使用することをおすすめします。フィルター式を使用して、期限切れの項目を Scan 結果と Query 結果から削除します。
- 削除済みの項目は、通常の削除操作で削除された項目と同様に機能します。削除されると、項目はユーザー削除ではなくサービス削除として DynamoDB Streams に入り、他の削除オペレーションと同様にローカルセカンダリインデックスとグローバルセカンダリインデックスから削除されます。

ElastiCache for Redis では、TTLs またはキャッシュされたキーの有効期限を使用して、キャッシュされたデータの鮮度を制御できます。設定時間が経過すると、キーはキャッシュから削除され、更新されたデータに到達しながらオリジンデータストアにアクセスする必要があります。

- 2 つの原則によって、適用する適切な TTLs と実装するキャッシュパターンのタイプが決まります。まず、基盤となるデータの変化率を理解することが重要です。次に、更新されたデータではなく、古いデータがアプリケーションに返されるリスクを評価することが重要です。
- 頻繁に変化する動的データでは、プライマリデータベースの変更率でデータの有効期限が切れる低い TTLs を適用できます。これにより、データベースリクエストをオフロードするためのバッファを提供しながら、古いデータを返すリスクが軽減されます。
- また、長時間ではなく数分または数秒間のみデータをキャッシュしている場合でも、キャッシュされたキーに TTLs を適切に適用すると、パフォーマンスが向上し、ゲームの全体的なプレイヤーエクスペリエンスが向上します。

実装手順

- Amazon S3 ライフサイクルポリシーを使用して、オブジェクトを低頻度のアクセス階層またはアーカイブ階層に移行し、ライフサイクルルールに基づいて不要なオブジェクトを削除する有効期限アクションを設定します。
- DynamoDB テーブルで有効期限 (TTL) を有効にして、書き込みスループットを消費せずに期限切れの項目を自動的に削除し、Unix エポック時間で期限切れタイムスタンプを定義します。
- 古いデータのデータ変更率とリスク許容度に基づいて ElastiCache キーに適切な TTLs を設定し、キャッシュされたデータの鮮度を促進し、プレイヤーエクスペリエンスを向上させます。

ハードウェアとサービス

GAMESUS02: ゲームバックエンドでのコンピューティングリソースの使用を管理するにはどうすればよいですか?

Studio は、さまざまなコンピューティングタイプ、マネージドサービス、削減計画を組み合わせて使用を最適化するコンピューティング戦略を開発する必要があります。また、不要なリソースの数を減らすために、ゲームサーバーとバックエンドサービスをコンピューティングインスタンスにまとめる方法を最適化する必要があります。

ベストプラクティス

- GAMESUS02-BP01 適切なコンピューティングワークロードにマネージドサービスを選択する
- GAMESUS02-BP02 コンピューティングのサイズを適正化し、必要な場合にのみ GPU パフォーマンスをデプロイする

GAMESUS02-BP01 適切なコンピューティングワークロードにマネージドサービスを選択する

ゲームバックエンドサービスを設計して、イベント駆動型または高度に可変なトラフィックワークロードにマネージドサービスを使用します。マネージドサービスは、マルチテナントのコントロールプレーンにより、インフラストラクチャの管理を に移行 AWS し、環境への影響を複数のユーザーに分散します。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

AWS AWS Fargate (コンテナ) AWS Lambdaや Amazon Gamelift (ゲームサーバーオーケストレーション) などの サービスは、基盤となるインフラストラクチャを管理することなく、コード、コンテナを実行したり、ゲームサーバーをオーケストレーションしたりできます。これらのサービスはプレイヤーの需要に基づいて自動的にスケーリングされ、消費したリソースに対してのみ課金されます。基盤となるインフラストラクチャはユーザーに代わって管理されるため、ゲームとバックエンドサービスの要件のみに集中できます。

AWS Lambda を使用すると、サーバーをプロビジョニングまたは管理することなくコードを実行できます。Lambda は、高可用性コンピューティングインフラストラクチャでコードを実行し、サーバーとオペレーティングシステムのメンテナンス、容量のプロビジョニングと自動スケーリング、ログ記録など、コンピューティングリソースの管理を実行します。Lambda では、Lambda がサポートする言語ランタイムのいずれかでコードを指定する必要があります。Lambda は、ゲームイベント、プレイヤー認証、ゲーム内購入処理、マッチメーキングリクエストの処理に役立ちます。Lambda はイベント数に基づいて自動的にスケーリングされ、トラフィックの予期しない急増を処理できます。

AWS Fargate は、Amazon Elastic Container Service (ECS) と Amazon Elastic Kubernetes Service (EKS) の両方で動作するコンテナ用のサーバーレスコンピューティングエンジンです。 AWS Fargate は、サーバーのプロビジョニングと管理の必要性を軽減することで、アプリケーションの構築に専念することを容易にします。また、アプリケーションごとにリソースを指定して料金を支払うことができ、設計上、アプリケーションの分離によってセキュリティが向上します。Fargate は、プレイヤープロファイル、状態管理、マッチメーキングを処理するバックエンドサービスに最適です。

Amazon GameLift は、セッションベースのマルチプレイヤーゲーム専用のゲームサーバーをデプロイ、運用、スケーリングするためのマネージドサービスです。最初のゲームサーバーをわずか数分でクラウドにデプロイできるため、ソフトウェアの初期開発にかかるエンジニアリング時間が最大数千時間短縮され、デベロッパーが設計からマルチプレイヤー機能を切り捨てることがよくある技術的なリスクが軽減されます。

実装手順

- ゲームイベント、プレイヤー認証、ゲーム内購入、マッチメーキングリクエストの処理などのイベント駆動型のワークロード AWS Lambda に を使用し、自動スケーリングとサーバーレス管理を活用します。
- プレイヤープロファイル、状態管理、マッチメーキングなどのバックエンドサービスに ECS または EKS AWS Fargate を使用してデプロイし、サーバー管理を削除してアプリケーションの分離を改善します。

- Amazon GameLift を使用して、セッションベースのマルチプレイヤーゲーム専用のゲームサーバーをデプロイおよびスケーリングし、開発時間と運用の複雑さを軽減します。

GAMESUS02-BP02 コンピューティングのサイズを適正化し、必要な場合にのみ GPU パフォーマンスをデプロイする

ゲームサーバーとバックエンドを設計して、コンピューティングリソースを効率的に活用します。コンピューティングを過剰にプロビジョニングすると、不要なコストが発生し、アイドル状態または使用率の低いリソースの量を最小限に抑えることができます。GPU インスタンスは、Unreal での HLOD 再構築や、ゲームサーバーが設計上それらを必要とする場合など、特定の開発作業をサポートするために使用する必要があります。これにより、ワークロードの環境への影響とコストが大幅に削減されます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

複数の EC2 インスタンスタイプと必要最小限のインスタンスを使用するように、ゲームサーバーとバックエンドサービスを最適化する必要があります。これにより、開発中またはゲームの起動時にニーズを満たすために使用できるインスタンスの数が増えます。また、インスタンスタイプをデプロイする特定のワークロードと一致させる必要があります。Compute Optimized インスタンスは、ゲームサーバーやマッチメーキングなどのバックエンドサービスなど、幅広いユースケースをサポートします。メモリ最適化インスタンスは、メモリ内の大規模なデータセットを処理するワークロードに高速なパフォーマンスを提供するように設計されています。高性能要件には GPU インスタンスを必要に応じて使用しますが、一般的なコンピューティングタスクには使用しません。可能であれば、サービスまたはゲームサーバーを AWS Graviton インスタンスで ARM で実行するように設計します。Graviton は、 で使用できるエネルギー効率の高いインスタンスタイプで最もパフォーマンスの高いインスタンスです AWS。また、x86 インスタンスタイプと比較してパフォーマンスとコストが向上します。

AWS Compute Optimizer を使用して、Amazon Elastic Compute Cloud (EC2) インスタンスタイプ、Amazon Elastic Block Store (EBS) ボリューム設定、 での Amazon Elastic Container Service (ECS) サービスのタスクサイズ AWS Fargate、商用ソフトウェアライセンス、 AWS Lambda 関数メモリサイズ、Amazon Relational Database Service (RDS) DB インスタンスクラスなどの最適な AWS リソース設定を識別し、機械学習を使用して履歴使用率メトリクスを分析します。Compute Optimizer は、一連の APIs とコンソールエクスペリエンスを提供し、ワークロードに最適な AWS リソースを推奨することでコストを削減し、 AWS ワークロードのパフォーマンスを向上させます。

実装手順

- ゲームサーバーには Compute Optimized インスタンス、大規模なデータセットには Memory Optimized インスタンス、HLOD 再構築や GPU 依存ゲームサーバーなどのタスクにのみ GPU インスタンスを使用して、コンピューティングリソースを特定のワークロードに一致させます。
- x86 インスタンスと比較して、エネルギー効率、パフォーマンス、コスト削減のために可能な限り AWS Graviton インスタンスをデプロイすることで、コンピューティング使用率を最適化します。
- AWS Compute Optimizer を使用して履歴使用率を分析し、EC2、 AWS ECS、Amazon RDS ワークロードの最も効率的な設定を推奨して AWS Lambda、コストを削減し、パフォーマンスを向上させます。

リソース

持続可能性に関連するベストプラクティスの詳細については、以下のリソースを参照してください。

- UN: ゲーム業界が地球への脅威に注目
- 中: ゲーム開発における環境持続可能性: よりグリーンな未来のために責任を持ってプレイする

主要な AWS サービス

- Amazon Aurora
- Amazon DynamoDB
- Amazon DocumentDB (MongoDB 互換性)
- Amazon ElastiCache
- Amazon S3
- Amazon S3 ストレージクラス
- Amazon S3 Intelligent-Tiering ストレージクラス
- Amazon S3 Express One Zone ストレージクラス
- Amazon Elastic Block Store
- AWS Lambda
- Amazon Elastic Container Service
- Amazon Elastic Kubernetes Service
- Amazon GameLift

- AWS Compute Optimizer

結論

ゲームは、世界中のプレイヤーにエンターテインメント体験を提供するように設計されており、使用特性は通常予測不可能で変動します。Games Industry Lens は、通常ゲームアーキテクチャを構成する一般的なタイプのシナリオについて説明し、クラウドでゲームを構築して運用するときに考慮すべき一連の質問とベストプラクティスを提供します。このフレームワークをゲームアーキテクチャに適用することで、信頼性が高く、安全で、効率的で、費用対効果の高いゲームをクラウドで構築できます。

寄稿者

このドキュメントの寄稿者は次のとおりです。

- Amazon Web Services、シニアソリューションアーキテクト、Adam Hatfield
- Amazon Web Services、テクニカルアカウントマネージャー、Brady Webb
- Bruce Ross – アマゾン ウェブ サービス、シニアソリューションアーキテクト、 Well-Architected レンズリーダー
- Amazon Web Services、Associate Solutions Architect、Caleb Cecil
- Carlos Perez、Cloud Optimization Success SA、Amazon Web Services
- Amazon Web Services、ESL テクニカルアカウントマネージャー、Chase Herrington。
- Amazon Web Services、シニアソリューションアーキテクト、Chris Blackwell
- Amazon Web Services、シニアテクニカルアカウントマネージャー、Corey Ouderkirk
- Derek Bottlevicencio、Prototyping SA、Amazon Web Services
- Amazon Web Services、シニアソリューションアーキテクト、Erik Ynigo Becerril
- Grzegorz Ochmanski、Amazon Web Services、シニアソリューションアーキテクト
- Amazon Web Services、シニアテクニカルアカウントマネージャー、Hadrian Baron
- Amazon Web Services、シニアカスタマーソリューションマネージャー、Ian Armbruster
- Amazon Web Services、シニアテクニカルアカウントマネージャー、Jed Obray
- アマゾン ウェブ サービス、シニアテクニカルアカウントマネージャー、Khurram Khokhar
- Kyle Somers、シニアマネージャー、ソリューションアーキテクチャ、Amazon Web Services
- Amazon Web Services、 Well-Architected、シニアテクニカルライター、Madhuri Srinivasan
- Matthew Wygant、シニア TPM ガイダンス、 Well-Architected、Amazon Web Services
- Nataliya Godunok、Cloud Optimization Success SA、Amazon Web Services
- Amazon Web Services、プリンシパルソリューションアーキテクト、Nirav Doshi
- Amazon Web Services、ソリューションアーキテクト、Olivia Liddell
- Amazon Web Services、プリンシパルテクニカルアカウントマネージャー、Randy James
- Amazon Web Services、Game Solutions Architect、Reou Ando
- Amazon Web Services、シニアテクニカルアカウントマネージャー、Richard Raseley
- Amazon Web Services、Senior Solutions Architect、Sam Patzer
- Amazon Web Services、シニアソリューションアーキテクト、Scott Selinger

- Amazon Web Services、シニアソリューションアーキテクト、Sean Allen
- Amazon Web Services、シニアソリューションアーキテクト、Serge Poueme
- Well-Architected、Amazon Web Services、シニアテクニカルライター、Stewart Matzek
- Trenton Potgieter、シニアソリューションアーキテクト AI/ML/Analytics、Amazon Web Services

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
新しいレンズバージョン	レンズ全体が新しいベストプラクティスガイダンスで更新されました。	2025 年 12 月 9 日
初版発行	ホワイトペーパーの初回発行。	2021 年 11 月 19 日

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。