Unreal Engine 開発者向けガイド

AWS GameKit

AWS GameKit: Unreal Engine 開発者向けガイド

Table of Contents

現在、Unreal Engine ソフトウェア用のコンテンツを表示しています。AWS GameKit のドキュメントをすべて見る

AWS GameKit とは

ゲームエンジンから AWS を利用したクラウド機能を構築できます。

AWS GameKit は、高品質のクラウドベースの機能を自社製品に組み込みたいと考えている開発者向けのオープンソース SDK です。クラウド機能はゲームにいくつかの大きなメリットをもたらします。例えば、セキュリティの強化、スケーラビリティ、コスト削減、そしてプレイ場所やプレイ方法の柔軟性の向上といったプレイヤー体験の改善などです。

AWS GameKit の目標は、主要な課題を取り除きながら、クラウド機能の力を利用できるようにすることです。AWS GameKit は、AWS やクラウドアーキテクチャ設計に関する深い知識はないが、プロジェクトに合わせてカスタマイズできる適切に構成された機能を求めている開発者向けに設計されました。AWS GameKit では、インフラストラクチャを構築するための完全なクラウドアーキテクチャテンプレートとツールを入手できます。プロジェクトに必要な機能を選択し、2~3 つのステップでクラウドバックエンドをセットアップして、クライアントアプリに機能を追加します。自分のペースで作業して、顧客に合わせてクラウドバックエンドを拡張し、カスタマイズすることができます。

AWS GameKit は、次のゲーム関連機能のソリューションを提供します。

- ID と認証 — ゲームの安全な登録と強固な ID 管理によりプレイヤーを保護します。プレイヤーのログインを検証してプレイヤーセッションへのアクセスを管理し、AWS GameKit クラウド機能の認証を使用します。
- アチーブメント — 評価を得たり、報酬を獲得したり、ゲームイベントを開始したりするためにプレイヤーが達成する目標を作成します。プレイヤーのアチーブメントをクラウドで管理し、長期目標に向けた進行状況を追跡します。
- ゲームの状態のクラウド保存 — クラウド内のゲームのセーブデータを同期して、プレイヤーが別の場所や別のデバイスからプレイを再開したり、必要に応じてゲームの進行状況を回復したりできるようにします。
- ユーザーゲームプレイデータ — インベントリ、統計、クロスプレイの永続性など、各プレイヤーのゲームプレイデータを管理し、プレイヤーがゲームにログインしたときはいつでもどこでも利用できるようにします。

AWS GameKit の利点

ゲームプロジェクト用のクラウドベースのバックエンドインフラストラクチャの構築を任されている開発者やアーキテクトは、以下の利点を活用できます。

- ゲームエンジンからクラウドバックエンドを構築して管理できる。合理化されたワークフローを備えた AWS GameKit for Unreal Engine を使用して、ゲームプロジェクトの AWS クラウドバックエンドを作成および管理します。ツールを使用して機能を組み込みます。複数の環境でバックエンドインフラストラクチャをセットアップし、それぞれを個別に管理します。
- 専門的に設計されたクラウドアーキテクチャから始められる。AWS GameKit の各クラウド機能の AWS ソリューションは、クラウドアーキテクチャの専門家が設計しており、安全で高性能、かつ回復力のある効率的なソリューションを実現する AWS Well-Architected フレームワークに基づいています。このソリューションには、ゲーム開発のベストプラクティスと顧客からのフィードバックが組み込まれています。
- 進めながら学べる。AWS GameKit には、バックエンド用のカスタマイズ可能なソリューションテンプレートと API が用意されています。つまり、本番環境に対応したバックエンドですぐに開始することができます。このときから、テンプレートの変更、代替 AWS 機能およびサービスの試行、プロジェクト用のカスタムクラウドインフラストラクチャの構築において幅広い柔軟性が得られます。
- 機能設計とラピッドプロトタイピングを統合できる。クラウドバックエンドが整ったら、事前設定された UI コンポーネント、サンプルコード、サンプルゲームを使用して、反復的な設計と開発を行って機能を統合します。
- AWS GameKit SDK とツールをカスタマイズできる。AWS GameKit SDK コンポーネントはソースが入手可能な状態で提供されるため、開発プロセスに合わせてツールを変更または構築できます。既存の AWS GameKit for Unreal Engine プラグインプラグインをカスタマイズするか、他のゲームエンジン用のバージョンを作成します。C++ API のコア機能を変更または拡張します。

AWS GameKit の開始方法

AWS GameKit を使用するのは初めてですか? ここから始めることをお勧めします。

- 自社のプロジェクト用に AWS GameKit をセットアップする
  - AWS GameKit for Unreal Engine をダウンロードする
  - AWS GameKit をインストールする
  - AWS GameKit の AWS アカウントとユーザーを設定する
- AWS GameKit UI について知る
  - Unreal Editor で AWS GameKit を試してみる
- 最新のニュースやリリースを入手する
  - AWS GameKit フォーラム。開発者コミュニティで質問やコメントを共有できます。

- AWS Game Tech ブログ。AWS Game Tech のすべてのサービスについて、新機能を学び、開発者向けのヒントを入手することができます。
- AWS GameKit のリリース。バージョン更新と既知の問題を把握できます。

関連サービス

AWS GameKit は、完全にカスタマイズ可能な AWS クラウドベースの機能をゲームに組み込み、それらの機能をカスタマイズしていく能力を保持するための確かな選択肢です。以下のゲーム関連の AWS サービスも検討してください。

- Amazon GameSparks – Amazon GameSparks は、ゲーム開発者にマルチサービスのバックエンドを提供するフルマネージド AWS サービスです。
- Amazon GameLift - GameLift は、ゲームサーバーをデプロイ、操作、スケーリングするフルマネージドサービスを含む、クラウドでのセッションベースのマルチプレイヤーゲームサーバーをホストするためのソリューションを提供します。
- Open 3D Engine (O3DE) – O3DE は、ゲーム、シミュレーション、およびマルチメディアの作成者向けのオープンソースの 3D 開発エンジンです。これはモジュール式で、クロスプラットフォームです。
- Amazon Nimble Studio – Amazon Nimble Studio は、ビジュアルエフェクト、アニメーション、インタラクティブコンテンツの各チームがスケーラブルなプライベートクラウドサービス内でコンテンツを安全に作成することを可能にする仮想スタジオです。

AWS 関連トピック

AWS Well-Architected

AWS GameKit ソリューションは、AWS Well-Architected とその 6 つの柱となるフレームワーク、すなわち運用上の優秀性、セキュリティ、信頼性、パフォーマンス効率、コスト最適化、持続可能性に基づいています。適切に設計された AWS ソリューションがどのように構築されるかについて詳しく知りたい場合は、ベストプラクティス、設計原則、業界固有のホワイトペーパーなど、このサイトのリソースを使用してください。

AWS for Games

マルチプレイヤーサービス向けの Amazon GameLift や、ゲーム分析と AI に特化したソリューションなど、ゲーム開発向けの他の AWS サービスやソリューションの詳細をご確認ください。

AWS GameKit の仕組み

概要

このトピックでは、Unreal Engine プロジェクトで AWS GameKit を使用する方法を知りたいゲーム開発者向けに技術的概要を示します。この概要では、AWS GameKit のコアコンポーネントについて説明し、開発者がそれらを操作してバックエンドのゲーム機能を構築および管理する方法について説明します。

AWS GameKit がお客様のゲームとどのように連携するか

AWS GameKit には、クラウドベースのゲーム機能をプロジェクトに組み込む作業を簡素化するツールが用意されています。AWS GameKit は、適切に設計されたゲームバックエンドを AWS クラウド上で設定してデプロイするのに役立ち、それをゲームフロントエンドに接続するための API も提供します。

次の図に示すように、AWS GameKit はゲームエンジンおよび開発環境で使用します。AWS GameKit は、クラウドバックエンドを構築および保守するための機能をゲームエンジンに追加します。

AWS GameKit では、ゲームに追加したいクラウドベースの機能に対応する、すぐに使えるバックエンドソリューションをデプロイできます。これらのソリューションには、バックエンドの実行とゲームクライアントとの通信に必要なアーキテクチャ設計と設定がすべて含まれています。ゲーム機能を選択し、オプションのカスタマイズをいくつか設定して、デプロイを開始します。10～30 分で、機能のバックエンドが稼働し、ゲームクライアントと通信できる状態になります。

AWS GameKit を使用してゲームバックエンドをデプロイすると、ゲームプロジェクトは、ゲームバックエンドのエンドポイントで自動的に設定されます。機能要素をゲームクライアントに追加し、AWS GameKit の API コールを使用してゲームバックエンドに接続します。エンジンでゲームをプレイテストし、バックエンドのライブコールを行うことができます。ゲームを構築して配信用にパッケージ化すると、ゲームは AWS 上のゲームのバックエンドリソースに自動的に接続されます。

AWS GameKit の機能を使用してゲームをプレイするユーザーは、プレイする前にゲームを登録してログインします。この時点からプレイヤーがログアウトするまで、AWS GameKit API のすべてのゲームクライアント呼び出しは、ログインしているプレイヤーに代わって行われます。例えば、ゲームクライアントが、クラウドに保存されているプレイヤーのゲームプレイデータを更新する API リクエストを送信すると、そのリクエストはプレイヤーの登録済み ID を自動的に参照します。

AWS GameKit のコンポーネント

AWS GameKit はゲーム開発者向けのオープンソースソリューションです。含まれているコンポーネントは次のとおりです。

- AWS GameKit プラグイン — ゲームエンジン用の AWS GameKit バージョンを使用して、各ゲーム機能のゲームバックエンドサービスを設定、デプロイ、管理します。バックエンドが用意できたら、AWS GameKit API を使用してバックエンドをゲームフロントエンドの機能に接続できます。バックエンドサービスは、ゲームエンジンから管理できます。また、AWS ツールを使用してモニタリングや変更を行うこともできます。ゲームエンジンのサポートの詳細については、「AWS GameKit のコンポーネント」を参照してください。
- AWS バックエンドサービスアーキテクチャソリューション — 各ゲーム機能ですぐに構築できるソリューションでは、AWS CloudFormation テンプレートを使用して AWS リソースをデプロイし、それらを AWS Lambda 駆動型ロジックで接続します。AWS GameKit で機能を設定してデプロイすると、これらのテンプレートを使用してゲームのバックエンドが構築されます。

  各ゲーム機能ソリューションで以下のようなタスクが処理されます。

  - AWS インフラストラクチャ上でバックエンドサービスを実行するためのリソースをプロビジョニングします。
  - バックエンドサービスのデプロイおよび更新アクティビティを管理します。
  - 認証と承認、セキュリティ、トラフィック管理、モニタリングなどの API 通信を処理します。
  - データ管理やゲームロジックなど、ゲーム機能のアクティビティを制御する Lambda 関数を提供します。
  - ゲーム機能のアクティビティをモニタリングし、ログを生成し、メトリクスの追跡を可能にします。Amazon CloudWatch を使用してダッシュボードを設定し、ゲーム機能のバックエンドの運用メトリクスを追跡します。
- AWS GameKit API — この API は、デプロイされたバックエンドサービスにゲームのフロントエンドを接続する機能固有のオペレーションを提供します。AWS GameKit API は、AWS コアサービス機能をゲーム固有のワークフローの API リクエストにまとめます。例えば、`UpdateAchievement()` オペレーションには、プレイヤーのリクエストを認証したり、アチーブメントの要件に基づいてロジックを実行したり、プレイヤーに保存されているアチーブメントステータスを更新したりするための AWS サービスの直接呼び出しが含まれます。

  AWS GameKit の機能が含まれるゲームは、ゲームバックエンドのエンドポイントで自動的に設定されます。さらに、すべての API コールは、ログインしているプレイヤーの固有のプレイヤー ID を参照します。

- サンプルコードおよびアセット — サンプルは、ゲーム機能をゲームに統合するのに役立つ出発点となります。各ゲーム機能には、図解やラピッドプロトタイピングに役立つサンプル素材一式が含まれています。
- AWS GameKit プラグインのソースコード — このソースコードは「入手可能なソース」ベースで GitHub からダウンロードできます。ソースコードを使用して、カスタマイズしたり、まだサポートされていないゲームエンジンの新しいバージョンを作成したりします。

AWS GameKit を使用して、AWS アカウントを通じて所有および管理するゲームバックエンド用のセルフマネージド AWS サービスをセットアップします。AWS GameKit を使用すると、しっかりと構築された基本的な、ゲーム用のバックエンドアーキテクチャを導入できます。ゲームの進化に合わせて、AWS ツールを使用してゲームバックエンドをカスタマイズしたり拡張したりできる柔軟性が非常に高くなります。

AWS GameKit の特徴

AWS GameKit は、ゲーム開発に対する以下のアプローチをサポートしています。

すべてゲームエンジン内で作業できる

AWS GameKit を使用すると、クラウドベースのゲーム機能に関するほぼすべての作業をゲームエンジンから行うことができます。

ゲームバックエンドを設定するには、AWS GameKit を使用して各ゲーム機能の AWS リソースを設定、デプロイ、管理します。また、ゲームエンジンでバックエンドサービスの進行状況を追跡したり、運用メトリクスにアクセスしたりすることもできます。

ゲームクライアントをバックエンドに接続するには、AWS GameKit API を使用してバックエンドサービスにリクエストを行います。API にアクセスして、ゲームエンジンでサンプルを実行できます。ゲーム機能のバックエンドを設定したら、ゲームエンジンでゲームをテストし、ログメッセージを使用して機能の問題をデバッグできます。

複数の環境で作業できる

ゲーム開発チームは、複数の環境を使用して、プロジェクト開発のさまざまなステージを同時に管理できます。ゲームエンジンで作業するときに AWS GameKit 環境を切り替えることで、チームは環境ごとに AWS リソースの別個のスタックを設定してデプロイすることができます。ゲームプロジェクトは、現在アクティブな環境のバックエンドエンドポイントに接続するように自動的に設定されます。

チームは、開発用、QA 用、本番用に事前定義された環境を使用できます。また、カスタム環境を作成することもできます。

環境を切り替えると、次のような作用が生じます。

- 各ゲーム機能の設定 UI が更新されて、アクティブな環境でデプロイされた AWS リソースの設定オプションとデプロイステータスが反映されます。
- アチーブメントの定義については、クラウドの同期によって、アクティブな環境のデータストアからプルされます。
- ダッシュボードリンクは、アクティブな環境の機能ダッシュボードを指し示します。
- AWS リソースを作成、再デプロイ、または削除するための UI 要素は、アクティブな環境のリソースにのみ影響を与えます。
- ゲームプロジェクト設定用の AWS GameKit は、アクティブな環境のバックエンドエンドポイントで更新されます。ゲームエンジンでゲームをプレイすると、あるいはゲームを構築またはパッケージ化すると、ゲームプロジェクトが環境のゲームバックエンドに自動的に接続されます。

ゲーム機能の AWS リソースをデプロイすると、リソース名は、使用中の AWS GameKit 環境を参照します。この命名規則は、AWS ツールを使用して、環境に基づいてリソースとコストを追跡するのに役立ちます。

チームは、必要に応じて、環境ごとに AWS ユーザーアクセスを管理できます。例えば、チームは本番環境のリソースへの書き込みアクセスを制限する場合があります。

バックエンドサービスを地理的に配置する

ゲームバックエンドの AWS リソースをデプロイするときは、リソースを物理的にデプロイする場所を選択します。利用可能な AWS リージョンから選択できます。各リージョンは、コンピューティングハードウェアの地理的位置を表します。AWS GameKit をサポートするすべての AWS リージョンのリストについては、「AWS GameKit がサポートされている AWS リージョン」を参照してください。

各 AWS GameKit 環境には AWS リージョンがあり、当該環境用に作成されたすべてのリソースは、そのリージョンにデプロイされます。バックエンドサービスを複数のリージョンに設定するには、複数の環境を作成します。

リージョンを選択するときは、次の点を考慮します。

- プレイヤーベースから地理的に近い位置にサービスをデプロイすることで、レイテンシーの問題の影響を減らすことができます。

- 停止や速度低下はまれですが、リソースが豊富で使用量の少ないリージョンを使用することで、それらが発生する可能性をさらに最小限に抑えることができます。
- コストや、サポートされる AWS サービスは、リージョンによって異なります。
- 環境のリージョン設定を変更するには、その環境にデプロイされているすべての AWS リソースを削除して置き換える必要があります。

チームで開発する

複数のメンバーがいる開発チームは、それぞれのバージョン管理システムで AWS GameKit を使用できます。AWS GameKit は、追跡と共有を簡単なものにするために、ファイルをゲームプロジェクトフォルダに保存します。

チームの AWS GameKit ユーザーは、AWS GameKit リソースに対する権限がある AWS アカウントアクセス権を持っている必要があります。管理者は、チームメンバーの AWS ユーザーを管理し、ユーザーグループを設定して適切な権限レベルを管理することができます。

AWS GameKit による開発ワークフロー

概要

このトピックでは、ゲーム開発者が AWS GameKit を使用してクラウドベースの機能をゲームプロジェクトに追加する際に想定される手順の概要を説明します。

次の手順で、AWS GameKit のゲーム機能を追加するための一般的な統合プロセスについて説明します。これらの手順は、ゲームエンジンで実行することで完了します。

1. 開発者 (または AWS アカウント管理者) は、AWS アカウントを作成するか、既存のアカウントを指定して、AWS 上のゲームバックエンドを管理します。AWS GameKit プラグインを使用するチームメンバーには、AWS GameKit アクセス権限とセキュリティ認証情報を持つ AWS ユーザー ID が必要です。
2. 開発者は、ゲームプロジェクト用の AWS GameKit プラグインをインストールします。
3. 開発者は、ゲームエンジンでゲームプロジェクトを開いた状態で、プロジェクト用の AWS GameKit 設定を作成します。このステップで、開発者はゲームプロジェクトのエイリアスを作成し、作業する環境を選択します。AWS ユーザー認証情報を送信すると、ゲームプロジェクトが AWS アカウントにリンクされます。それに応じて、AWS GameKit はデフォルトの設定ファイルを生成し、ゲームプロジェクトフォルダに追加します。

4. 開発者は、ID と認証のゲーム機能のバックエンドを設定してデプロイします。
   a. 開発者は、ゲーム機能の設定オプションを設定します。
   b. 開発者は、[作成] を選択し、ID と認証のバックエンドに設定されたとおりに AWS リソースをデプロイします。AWS GameKit は、すべての機能で使用するためのコアリソースもいくつかデプロイします。ID と認証のリソーススタックには、プレイヤー登録用の Amazon Cognito ユーザープールが含まれます。開発者は、プラグインの UI とゲームエンジンの出力ログで AWS リソース作成の進行状況を追跡します。

      > **Note**
      >
      > リソースをデプロイすると、AWS アカウントに使用料が発生し始める可能性があります。詳細については、「AWS GameKit 利用時の AWS の料金」を参照してください。

   c. 必要に応じて、開発者は、機能の構成設定を編集してバックエンドを再デプロイしたり、バックエンドを削除して最初からやり直したりすることができます。
5. 開発者は、フロントエンドのゲームコードにゲーム機能を追加します。機能の統合作業はさまざまですが、一般的には、新しい UI 要素の作成、ゲームロジックの追加、AWS GameKit API を使用したゲームバックエンドの呼び出しが含まれます。ID と認証のために、開発者は UI ワークフローを作成し、ユーザーがプレイヤー ID を登録してゲームにログイン/ゲームからログアウトできるようにします。これにはパスワードリセットオプションが含まれる場合もあります。

   開発者は、次のプラグイン機能を使用して統合に役立てることができます。
   - プラグインに付属するサンプルアセットを使用します。これには、サンプルの UI 要素、API コールを示すコードスニペット、完全に動作するゲームサンプルが含まれます。
   - ゲームエンジンで ID と認証の機能をテストします。エディタで実行されるゲームは、ゲームバックエンドに接続するように自動的に設定されます。機能に、デプロイされたゲームバックエンドが存在する場合、ゲームはバックエンドに API コールを送信してレスポンスを受け取ることができます。
6. 開発者は、ID のゲーム機能を導入したら、AWS GameKit の追加機能を統合します。この作業では、ID と認証と同様のプロセスに従います。つまり、ゲーム機能のバックエンドを作成し、フロントエンドに機能を追加して、接続をテストします。
7. 開発者は、AWS GameKit のコンポーネントが含まれるようにゲームをパッケージ化します。
8. ゲーム開発が進むにつれて、開発者は、プラグインを使用して、QA 用、本番用、その他のステージ用の新しい環境を作成できるようになります。開発者は、各環境で、ゲームバックエンド用の AWS リソースのスタックを個別に作成します。これにより、開発者は、別々の環境で個別に機能開発とバックエンド設定を管理することができます。

AWS GameKit 利用時の AWS の料金

概要

このトピックは、ゲーム開発者が AWS GameKit でゲームバックエンドを作成する際に AWS アカウント でどのような料金が発生するかを理解するのに役立ちます。このトピックでは、お客様のコスト管理を支援するために AWS が提供しているツールについて説明します。

AWS では AWS GameKit ツールを無料で提供しています。AWS GameKit でゲームバックエンドを構築する場合は、ゲームバックエンドを実行するために作成して使用する AWS 製品に対してのみ料金が発生します。標準の AWS サービス料金は、公開されているとおりに適用されます。

ゲーム開発中は AWS 無料利用枠を利用できる場合があります。AWS のお客様は、無料利用枠の特典を利用して、さまざまな AWS 製品を無料で実際に体験することができます。AWS GameKit で作成したすべての AWS リソースは、AWS 無料利用枠の一部として利用できます。無料利用枠の特典には、期間制限や使用量制限がある場合があります。AWS 無料利用枠の詳細については、以下を参照してください。

- 「Free Tier benefits by service」
- 「AWS 無料利用枠を開始しましょう」(AWS ホワイトペーパー)

AWS アカウント 所有者は、AWS リソースを追加、変更、削除することで支出を完全にコントロールできます。コスト管理に役立つさまざまな AWS ツールも利用できます。AWS アカウント では、AWS リソースをデプロイするまで料金は発生しません。AWS GameKit プラグインは、いつ料金が発生し始める可能性があるかをユーザーに通知します。デプロイされた AWS リソースを削除すれば、料金の発生を止めることができます。

AWSコストの見積もり

AWS GameKit 利用時のコストは、構築するゲームバックエンドとゲームでの使用方法によって異なります。バックエンドを実行するための総コストは、バックエンドの各 AWS サービスのコストの合計と等しくなります。

各 AWS サービスのコスト基準はそれぞれ異なります。時間ベースの料金を請求するサービスもあれば、使用量に基づいて請求するサービスもあります。例えば、Amazon Cognito では月間アクティブユーザー数に基づいて請求がなされますが、AWS Lambda ではリクエスト数と使用されたコンピューティング時間に基づいて請求がなされます。

ゲームバックエンドには次の 2 つの AWS リソースセットが存在します。

- コアリソース — このサービスはバックエンドインフラストラクチャの管理に役立ちます。AWS GameKit は、ID と認証の機能のバックエンドを初めて作成するときにコアリソースをデプロイします。ID と認証の機能のバックエンドは、常に最初に作成されるゲーム機能です。
- 機能リソース — このサービスは、各 AWS GameKit 機能のソリューションアーキテクチャを構成します。機能固有のリソースとコスト情報の詳細については、各ゲーム機能のガイドセクションを参照してください。

AWS GameKit コアサービスには以下が含まれます。

- Amazon API Gateway — AWS GameKit は、このサービスを使用して、ゲームのバックエンドサービスの API コールを管理します。API Gateway は、トラフィック、承認とアクセス制御、スロットリング、モニタリングを管理します。API Gateway の料金モデルは API コールの量に依存しています。12 か月間の無料利用枠の特典には、このサービスが含まれます。詳細については、「Amazon API Gateway の料金」を参照してください。
- AWS Lambda — AWS GameKit は、このサービスを使用して、リソースの更新、デプロイ、コンピューティングサービスなどのアクティビティを管理するカスタムコードを実行します。Lambda の料金モデルは、リクエストの数と、必要なコンピューティング時間に依存しています。12 か月間の無料利用枠の特典には、このサービスが含まれます。詳細については、「AWS Lambda の料金」を参照してください。
- Amazon Simple Storage Service (Amazon S3) — AWS GameKit は、AWS リソースの作成および更新時にこのサービスを使用します。Amazon S3 の料金モデルは、使用されるストレージ容量と、PUT リクエストおよび GET リクエストの数に依存しています。12 か月間の無料利用枠の特典には、このサービスが含まれます。詳細については、「Amazon S3 の料金」を参照してください。
- AWS CloudFormation — AWS GameKit は、このサービスを使用して、ゲームバックエンドのクラウドインフラストラクチャリソースをモデル化し、プロビジョニングします。このサービスは、標準の無料利用枠の上限までは無料です。詳細については、「AWS CloudFormation の料金」を参照してください。
- AWS Identity and Access Management (IAM) — AWS GameKit は、このサービスを使用して、ゲームの AWS リソースへのアクセスを制御します。IAM の使用料は発生しません。
- Amazon CloudWatch — AWS GameKit は、このサービスを使用して、各ゲーム機能のバックエンドの運用メトリクスモニタリングツールでダッシュボードを管理します。CloudWatch の料金モデルは、使用量と毎月のダッシュボード料金に依存しています。12 か月間の無料利用枠の特典には、このサービスが含まれます。詳細については、「Amazon CloudWatch の料金」を参照してください。

> **Note**
>
> CloudWatch の無料利用枠では、AWS アカウント ごとに最大 3 つのダッシュボードを使用できます。利用可能なすべての AWS GameKit 機能のバックエンドを作成し、それぞれのダッシュボードをアクティブ化すると、AWS アカウント の無料利用枠の特典を超えることになります。

- Amazon Cognito — AWS GameKit は、このサービスを使用して、プレイヤーの ID と認証の認証情報を管理します。Amazon Cognito の料金モデルは、ユーザー (プレイヤー) プール内の月間アクティブユーザー数 (MAU) に依存しています。12 か月間の無料利用枠の特典には、このサービスが含まれます。詳細については、「Amazon Cognito の料金」を参照してください。

AWS のコスト管理

使用状況とコストを追跡するには、AWS Management Console の AWS Billing and Cost Management ダッシュボードを使用します。AWS GameKit を通じてデプロイされるすべての AWS リソースは、リソースとコストの追跡に役立つ命名規則を使用しています。リソース名は文字列 `gamekit_` で始まり、次の要素が含まれています。

- 環境コード (「dev」など)
- AWS リージョンコード (「us-west-2」など)
- AWS GameKit のゲームタイトル/エイリアス
- ゲーム機能名 (該当する場合) (「UserGameplayData」など)

予期せぬコストや過剰なコストを避けるため、ベストプラクティスとして、AWS Budgets を使用してアラートで予算の上限を設定します。コスト管理の詳細については、10 分間のチュートリアル「AWS コストを管理する」を参照してください。

AWS GameKit のコンポーネント

AWS GameKit には、次のコンポーネントが含まれています。詳細なバージョン情報については、「AWS GameKit リリース」を参照してください。

AWS GameKit for Unity ソフトウェア

Unity ゲームプロジェクトに追加すると、AWS GameKit パッケージは、Unity Editor から直接プロジェクトのクラウドバックエンドを構築するための UI と機能を追加します。パッケージのダウンロードには、AWS GameKit C# API for Unity と C# のサンプルコードが含まれています。開発者は、Apache 2.0 ライセンスで「ソースが入手可能」として提供されているパッケージソースコードをカスタマイズできます。

AWS GameKit for Unity をダウンロードする、または GitHub からソースコードを入手する

Unreal Engine 用 AWS GameKit プラグイン

このプラグインは、AWS GameKit の機能とアセットを Unreal Editor に追加します。このプラグインを使用すると、AWS クラウド上にゲームバックエンドを構築し、Unreal ゲームプロジェクトにクラウドベースの機能を追加することができます。このプラグインのダウンロードには、AWS GameKit C++ API for Unreal、C++ サンプルコード、Unreal ブループリント、サンプルゲーム要素が含まれています。開発者は、Apache 2.0 ライセンスで「ソースが入手可能」として提供されているプラグインソースコードにアクセスして、プラグインをカスタマイズできます。

最新の Unreal Engine 用 AWS GameKit プラグインをダウンロードする

GitHub で Unreal Engine 用 AWS GameKit プラグインのソースコードを入手する

AWS GameKit Core C++ SDK

この SDK には AWS GameKit のコアゲーム機能が含まれており、エンジン固有の API の基礎となります。開発者は、Apache 2.0 ライセンスで「ソースが入手可能」として提供されている SDK ソースコードにアクセスして、あらゆるゲームエンジン用のカスタム API およびプラグインを構築できます。

GitHub で AWS GameKit C++ SDK ソースを入手する

その他のリソース

- AWS GameKit ドキュメント
- AWS GameKit リリース (現在および以前のバージョンへのリンク、リリースノート、既知の問題を含む)
- AWS GameKit フォーラム
- AWS Game Tech ブログ

AWS GameKit のセットアップ

AWS GameKit を使用する準備ができたら、以下のセットアップタスクから始めます。完了すると、ゲームエンジンに AWS GameKit がインストールされ、ゲームプロジェクト用のクラウドバックエンドの構築を開始できます。

- AWS GameKit プラグインを入手してインストールします。
- AWS アカウント をセットアップします。
- プラグインで使用する AWS GameKit の許可とセキュリティ認証情報を使用して AWS アカウントユーザーを作成します。

> **Note**
>
> セットアップタスクによっては、AWS Management Consoleを使用する必要がある場合があります。セットアップが完了すると、必要に応じてゲームエンジンからゲームのバックエンド全体に対して作業ができるようになります。

トピック

Unreal Engine での AWS GameKit プラグインのインストール

概要

Unreal Engine ゲームプロジェクトで使用するための AWS GameKit をダウンロードしてインストールします。このトピックでは、ゲーム開発者向けに、Unreal Editor で AWS GameKit プラグインをセットアップするための詳細な手順を示します。

プラグインの要件

AWS GameKit プラグインを使用するには、以下の条件があります。

- プラグインのバージョンと互換性のある Unreal Engine のバージョン (「AWS GameKit のバージョン情報」を参照)。
- C++ Unreal ゲームプロジェクト。ソースコードのないブループリントのみのプロジェクトは、このプラグインと互換性がありません。

AWS GameKit プラグインのソースコードはカスタマイズ可能です。コードを変更してカスタムプラグインを生成するには、C++ ゲームプロジェクトを扱うためのコード工ディタが必要です。例えば、Visual Studio には以下のツールが必要です。

- 以下のツールがインストールされた Visual Studio 2019。
  - [Workloads] (ワークロード) タブで、以下を行います。
    - C++ によるデスクトップ開発
    - 以下のオプションを備えた、C++ によるゲーム開発:
      - C++ プロファイリングツール
      - C++ AddressSanitizer (オプション)
      - Windows 10 SDK (10.0.18362 以降)
      - Unreal Engine インストーラ
  - [個別のコンポーネント] タブ: .NET Framework 4.8 SDK

AWS GameKit のダウンロードに含まれるもの

ダウンロードには以下が含まれます。

- Unreal Engine のプラグインバイナリ。
- 各ゲーム機能に対応する機能を備えた AWS GameKit C++ ライブラリ。
- 各ゲーム機能のブループリントコードと UI サンプル。
- 各ゲーム機能の API コールを備えた C++ コード例。
- AWS GameKit の許可とセキュリティ認証情報を持つ AWS ユーザーをセットアップするための自動スクリプト。
- AWS GameKit がゲームのバックエンドの作成に使用する、デフォルト設定ファイル。

プラグインをインストールする

AWS GameKit パッケージをダウンロードし、C++ Unreal ゲームプロジェクト用のプラグインをインストールします。

プラグインをインストールするには:

1. ゲームエンジン用の AWS GameKit を入手します。Unreal Engine 用 AWS GameKit GitHub リポジトリ (aws/aws-gamekit-unreal) から `.zip` ファイルをダウンロードします。
2. **`.zip`** ファイルを解凍します。

   a. AWS GameKit で使用する C++ Unreal ゲームプロジェクトのディレクトリパスを見つけます。ディレクトリフォルダ `Plugins/Marketplace` を開きます。例: `... > Unreal Projects > MagicChickenGame >Plugins > Marketplace`。このフォルダが存在しない場合は、作成します。

   b. AWS GameKit プラグインの zip ファイルの内容を抽出し、ファイルをゲームプロジェクトフォルダに配置します。抽出されたファイルは、ルートに AWS GameKit プラグイン記述子ファイルの `AwsGameKit.uplugin` がある、`AwsGameKit` という名前のフォルダにまとめられています。Unreal Engine はこのファイルをプラグインとして認識します。

   > **Note**
   >
   > プラグインをインストールして任意の Unreal プロジェクトで使用するには、Unreal Engine のインストール先のディレクトリパスにあるフォルダ `Plugins/Marketplace` にファイルを配置します。プラグインを両方の場所にインストールしようとしないでください。エラーが発生します。

3. AWS GameKit を使用してプロジェクトを再構築します。

   a. ゲームプロジェクトのルートフォルダに移動し、ソリューション (`*.sln`) ファイルを探します。存在しない場合は、`.uproject` ファイルを見つけてプロジェクトファイルを生成します。

   b. ソリューションファイルを開き、プロジェクトを構築または再構築します。

4. ゲームプロジェクトのプラグインを有効にします。

   a. Unreal Editor でゲームプロジェクトを開きます。メインメニューで [編集]、[プラグイン] を開き、AWS GameKit プラグインを検索します。

b. [有効] ボックスをオンにして、ゲームプロジェクトのプラグインを有効にします。このアクションにより、エディタを再起動するよう求めるプロンプトが生成されます。エディタに [プロジェクトは期限切れです。更新しますか?] というプロンプトが表示されたら、[更新] を選択します。

c. このゲームプロジェクトを使用してエディタを再起動します。プロジェクトを構築または再構築するよう求めるエラーメッセージが表示された場合は、ステップ 3 を繰り返します。

5. プラグインがインストールされていることを確認します。コンテンツブラウザで AWS GameKit コンテンツを探します。コンテンツが表示されない場合は、[表示オプション] 設定で [プラグインコンテンツを表示] オプションが選択されていることを確認します。

6. プロジェクトの **`.Build.cs`** ファイルを更新します。

   a. *`[project name]`*`.Build.cs` ファイルを見つけて IDE で開きます。例: `...Unreal Projects\MagicChickenGame\MagicChickenGame.Build.cs`。

   b. `PublicDependencyModuleNames` に次の文字列を追加します:「AwsGameKitCore」と「AwsGameKitRuntime」。

      例:

```
PublicDependencyModuleNames.AddRange(new string[] { "Core", "CoreUObject",
 "Engine", "InputCore", "AwsGameKitCore", "AwsGameKitRuntime" });
PrivateDependencyModuleNames.AddRange(new string[] { "AwsGameKitCore",
 "AwsGameKitRuntime" });
```

AWS GameKit の AWS アカウントのセットアップ

概要

このトピックでは、AWS GameKit アクティビティ用の AWS アカウントとユーザーをセットアップする方法について説明します。この情報は、管理者と、AWS ユーザーアカウントを管理するその他のユーザーを対象としています。

AWS GameKit を使用してゲームプロジェクトにクラウドベースの機能を組み込むための最初の手順として、プロジェクトで使用する AWS アカウントを作成します。このアカウントを使用して、ゲームのバックエンドのクラウドリソースを管理します (コストの追跡やユーザーアクセスの制御など)。

ゲームプロジェクト用の AWS アカウントをセットアップするには:

- AWS アカウントを取得します。既存のアカウントを使用するか、新しいアカウントを作成できます。
- AWS アカウントでユーザーをセットアップします。AWS GameKit を使用するためにアクセスを拡張し、各ユーザーのセキュリティ認証情報を生成します。

> **Note**
>
> アカウントで新しい AWS IAM アイデンティティセンターを使用してユーザーを管理する場合、AWS GameKit プラグインで異常な動作やメッセージングが発生する可能性があります。このセキュリティ認証情報では短期間のアクセスが提供されるため、ユーザーは今後の使用のために保存できず、頻繁に再生成する必要があります。AWS GameKit に付属する自動スクリプトを使用してユーザーを作成すると、ユーザーは長期的なアクセスキーを取得します。

AWS アカウントへのサインアップ

AWS アカウントを持っていない場合や、プロジェクト用に別のアカウントをセットアップする場合は、以下の手順を実行してアカウントを作成します。

AWS にサインアップしてアカウントを作成するには

1. https://portal.aws.amazon.com/billing/signup を開きます。
2. オンラインの手順に従います。サインアップ手順の一環として、通話呼び出しを受け取り、電話のキーパッドを用いて検証コードを入力するように求められます。

   AWS アカウントにサインアップすると、AWS アカウントのルートユーザーが作成されます。ルートユーザーには、アカウントのすべての AWS サービスとリソースへのアクセス権があります。セキュリティのベストプラクティスとして、管理ユーザーに管理アクセスを割り当て、ルートユーザーのみを使用してルートユーザーアクセスが必要なタスクを実行してください。

サインアップ処理が完了すると、AWS からユーザーに確認メールが送信されます。https://aws.amazon.com/ の [アカウント] をクリックして、いつでもアカウントの現在のアクティビティを表示し、アカウントを管理することができます。

サインアッププロセスに関する詳細なガイダンスとヒントについては、「新しい AWS アカウントアカウントを作成してアクティブ化する方法を教えてください」を参照してください。

管理者のセットアップ

AWS アカウントにサインアップした後、日常的なタスクにルートユーザーを使用しないように、管理ユーザーを作成します。

AWS アカウントのルートユーザーの保護

1. [ルートユーザー] を選択し、AWS アカウントの E メールアドレスを入力することで、アカウント所有者として AWS マネジメントコンソールにサインインします。次のページでパスワードを入力します。

   ルートユーザーを使用してログインする方法については、「AWS サインインユーザーガイド」の「ルートユーザーとしてサインインする」を参照してください。
2. ルートユーザーの多要素認証 (MFA) を有効にします。

   手順については、「IAM ユーザーガイド」の「AWS アカウントのルートユーザーの仮想 MFA デバイスを有効にする (コンソール)」を参照してください。

管理ユーザーを作成する

日常的な管理タスクでは、AWS Identity and Access Management (IAM) サービスの管理ユーザーにアクセスを割り当てます。

1. ルートユーザー認証情報 (AWS アカウントの E メールアドレスとパスワード) で AWS マネジメントコンソールにサインインします。
2. [コンソールホーム] のページで、IAM サービスを選択します。
3. ナビゲーションペインで [ユーザー]、[ユーザーを追加] の順に選択します。
4. [ステップ 1: ユーザーの詳細を指定] で、以下のように設定します。

   - 新しいユーザーの [ユーザー名] を入力します。これは、AWS のサインイン名です。
   - [AWS マネジメントコンソール (オプション) へのユーザーアクセスを提供] を選択します。これにより、新しいユーザーの AWS Management Consoleサインイン認証情報が生成されます。[IAM ユーザーを作成する] オプションを選択します。
   - [コンソールパスワード] オプションを選択します。

   [Next] (次へ) をクリックします。

5. [ステップ 2: アクセス許可を設定] で、[ポリシーを直接アタッチする] オプションを選択し、リストから [AdministratorAccess] ポリシーを選択します。[Next] (次へ) をクリックします。
6. [ステップ 3: 確認と作成] で、設定を確認し、[ユーザーの作成] を選択します。
7. [ステップ 4: パスワードを取得] では、新しいユーザーによるサインインに関する情報がコンソールに表示されます。この時点で、新しい管理ユーザーはコンソールのサインイン認証情報を持っていますが、AWS GameKit プラグインを使用するために必要なセキュリティ認証情報をまだ持っていません。[ユーザーの表示] ボタンを選択します。
8. ユーザー詳細ページで、[セキュリティ認証情報] タブを開き、[アクセスキー] セクションに移動します。[Create access key] (アクセスキーの作成) を選択します。
9. [主要なベストプラクティスと代替案にアクセスする] で [ローカルコード] オプションを選択して、推奨事項を確認し、[次へ] を選択してアクセスキーを作成します。手順に従って新しいアクセスキーをダウンロードし、保存します。AWS GameKit を使用するには、このキーが必要です。

AWS GameKit アクセス権を持つユーザーのセットアップ

すべての AWS GameKit ユーザーは、ゲームプロジェクトの AWS リソースをデプロイ、更新、または削除する前に、AWS アクセス権を持つ必要があります。AWS アカウント管理者は、ユーザーを作成し、ユーザーアクセスを管理し、ユーザーごとに固有のセキュリティ認証情報を生成します。

AWS GameKit には、AWS GameKit アクセス権を持つ AWS ユーザーをセットアップするための自動スクリプトが用意されています。管理者はこのスクリプトを使用して、新しいユーザーを作成したり、既存のユーザーにアクセス権を拡大したりできます。このスクリプトは、長期的なアクセスキーを持つ IAM ユーザーを作成します。別の方法として、管理者が AWS Identity and Access Management (IAM) サービスで直接作業することもできます。IAM での作業のガイダンスについては、「AWS 関連トピック」を参照してください。

AWS GameKit スクリプトを使用して AWS ユーザーを作成または更新するには

1. AWS GameKit のインストールファイルから、Python スクリプト `create_IAM_user.py` を見つけます。

```
[AWS GameKit install location]\AwsGameKit\Resources\cloudResources\policies
\create_IAM_user.py
```

このスクリプトを使用するには、ユーザーを追加または更新する AWS アカウントの管理者権限が必要です。その他の要件については、同じディレクトリにある `requirements.txt` ファイルを参照してください。

2. ユーザーを作成または更新するには、以下の引数を指定してスクリプトを実行します。

```
python create_IAM_user.py [AWS USERNAME] [AWS ACCESS KEY] [AWS SECRET KEY]
```

- `AWS USERNAME` は、追加または更新するユーザー名です。
- `AWS ACCESS KEY` と `AWS SECRET KEY` は、AWS アカウントの管理者認証情報です。

リクエストが成功すると、スクリプトは以下のアクションを実行します。

- リクエストされたユーザー名が AWS アカウント内に既に存在するかどうかを確認します。存在しない場合、スクリプトはアカウント内に新しい IAM ユーザーを作成します。
- 「GameKitDevGroup」という名前のユーザーグループが AWS アカウント内に存在するかどうかを確認します。存在しない場合、スクリプトは新しい「GameKitDevGroup」ユーザーグループを作成し、`GameKitDeveloperPolicy` アクセス許可ポリシーをアタッチします。このポリシーは AWS GameKit ダウンロードパッケージ (`GameKitDeveloperPolicy_Template.json`) にも含まれています。
- リクエストされたユーザーを `GameKitDevGroup` ユーザーグループに追加します。
- 新しく作成されたユーザーの場合、そのユーザーの長期的なセキュリティ認証情報を生成し、`...\policies` ディレクトリの `[username]_credentials.txt` に保存します。

AWS 関連トピック

- 「新しい AWS アカウントを作成してアクティブ化する方法を教えてください」、「AWS ナレッジセンター」
- 「 Organizing Your AWS Environment Using Multiple Accounts」、「AWS ホワイトペーパー」
- 「AWS アクセスキーを管理するためのベストプラクティス」、「AWS 全般のリファレンス」
- 「IAM ユーザーのアクセス許可の変更」、「IAM ユーザーガイド」
- 「Where are configuration settings/credential information stored?」、「AWS Command Line Interfaceユーザーガイド」

IAM を使用した AWS GameKit ユーザーアクセスのセットアップ

自動スクリプトを使用して、AWS GameKit にアクセスできる AWS ユーザーをセットアップすることができます。別の方法として、以下の手順に従って、これらのタスクを AWS Management Consoleで直接完了することもできます。

- AWS GameKit のアクセス権を持つユーザーのアクセス許可ポリシーを作成します。
- このアクセス許可ポリシーを、新規または既存のユーザーグループにアタッチします。
- IAM ユーザーを作成してユーザーグループに追加します。

AWS GameKit のアクセス許可ポリシーを作成するには:

1. AWS Management Console にサインインして、IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. IAM コンソールで、[アクセス管理] > [ポリシー] ページを開いて、AWS アカウント用の AWS GameKit アクセス許可ポリシーを作成または更新します。[Create policy] (ポリシーの作成) ボタンを選択します。
3. [JSON] タブを選択して エディタを開きます。既存の内容を、AWS GameKit ユーザー用のアクセス許可の構文に置き換えます。置き換えるポリシー構文を生成するには:

    a. AWS GameKit のポリシーテンプレートは、AWS GameKit ダウンロードパッケージに含まれていて、次の場所にあります: `...AwsGameKit\Resources\cloudResources\policies\GameKitDeveloperPolicy_Template.json`。使用している AWS GameKit プラグインのバージョン用のポリシーテンプレートを必ず使用してください。

    b. AWS アカウントを使用してテンプレート構文をカスタマイズします。`<YOUR_ACCOUNT_ID>` という文字列を、AWS アカウント ID (9 桁のアカウント番号) に置き換えます。AWS コンソール上部のアカウントユーザー名の下にある、アカウント ID を探します。

    これらの文字列を手動で置き換えるか、提供されている python スクリプト (`generate_policy_instance.py`) を使用して、カスタマイズした構文で新しい JSON ファイルを作成することができます。このスクリプトは、ポリシーテンプレートと同じディレクトリにある AWS GameKit ダウンロードパッケージに含まれています。AWS アカウント ID を使用してスクリプトを呼び出します。

```
python generate_policy_instance.py [AWS ACCOUNT ID]
```

スクリプトは、新しいポリシーファイルを同じディレクトリに保存します。AWS アカウント ID を含むファイル名 (`GameKitDeveloperPolicy-123456789` など) を探します。

4. IAM コンソールで、新しいポリシー構文を入力したら、[ポリシーを確認] ページが表示されるまで [次へ] を選択します。[Create policy] (ポリシーの作成) ボタンを選択します。

ユーザーグループにアクセス許可ポリシーを追加するには:

1. ナビゲーションペインで、[ユーザーグループ] を選択します。既存のユーザーグループのページを開くか、[グループを追加] を選択して新しいユーザーグループを作成します。
2. 新しいグループを作成する場合は、グループ名を入力します。
3. [許可ポリシーをアタッチ] で、カスタマー管理ポリシーの「GameKitDeveloperPolicy」を選択します。
4. ワークフローを完了して、ユーザーグループを作成または更新します。

AWS GameKit アクセス権を持つユーザーをセットアップするには

1. ナビゲーションペインで [ユーザー]、[ユーザーを追加] の順に選択します。
2. [ステップ 1: ユーザーの詳細を指定] で、以下のように設定します。

   - 新しいユーザーの [ユーザー名] を入力します。これは、AWS のサインイン名です。
   - [AWS マネジメントコンソール (オプション) へのユーザーアクセスを提供] を選択します。これにより、新しいユーザーの AWS Management Consoleサインイン認証情報が生成されます。[IAM ユーザーを作成する] オプションを選択します。
   - [コンソールパスワード] オプションを選択します。

   [Next] (次へ) をクリックします。
3. [ステップ 2: アクセス許可を設定] で、[ユーザーをグループに追加] オプションを選択します。リストから AWS GameKit のアクセス許可を持つユーザーグループを選択します。[Next] (次へ) をクリックします。
4. [ステップ 3: 確認と作成] で、設定を確認し、[ユーザーの作成] を選択します。
5. [ステップ 4: パスワードを取得] では、新しいユーザーによるサインインに関する情報がコンソールに表示されます。この時点で、新しいユーザーはコンソールのサインイン認証情報を持っていますが、AWS GameKit プラグインを使用するためのセキュリティ認証情報をまだ持っていません。[ユーザーの表示] ボタンを選択します。

6. ユーザー詳細ページで、[セキュリティ認証情報] タブを開き、[アクセスキー] セクションに移動します。[Create access key] (アクセスキーの作成) を選択します。
7. [主要なベストプラクティスと代替案にアクセスする] で [ローカルコード] オプションを選択して、推奨事項を確認し、[次へ] を選択してアクセスキーを作成します。手順に従って新しいアクセスキーをダウンロードし、保存します。AWS GameKit を使用するには、このキーが必要です。

アチーブメントのアクセス許可の管理

アチーブメントのゲーム機能で作業をする場合は、アチーブメント定義で作業をするために追加のアクセス許可が必要になる場合があります。デフォルトの `GameKitDeveloperPolicy` アクセス許可ポリシーでは、開発環境で作業する場合のみ、アチーブメント定義をクラウドに同期できます。

アチーブメント定義で作業をするには、`AwsGameKitAchievementAdmin` API を直接呼び出す必要があります。AWS GameKit は IAM ロールを使用して `AchievementAdmin` アクセス許可を管理します。これにより、ゲームを保護するための制御とセキュリティが強化されます。IAM ロールには、(1) 誰がそのロールを引き受けることができるか、(2) どのリソースをコントロールできるかを指定します。

ユーザーのアクセス許可を変更するには、AWS アカウントの管理者アクセスが必要です。ベストプラクティスとして、ユーザーグループにアクセス許可を割り当て、適切なアクセス許可を持つユーザーグループにユーザーを追加することで、ユーザーのアクセス許可を管理します。

`AchievementAdmin` アクセス許可の編集のオプション:

開発環境でのユーザーアクセスを削除するには

ユーザーグループのアクセス許可ポリシーから、以下のセクションを削除します。

```
      {
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": "arn:aws:iam::[YOUR_ACCOUNT_ID]:role/
gamekit_dev_*_AchievementsAdminInvokeRole"
        }
```

他の環境でのユーザーアクセスを追加するには

以下の手順に従います。

1. _AchievementsAdminInvokeRole で、ロールの信頼関係を編集して、特定のユーザーグループ ID を追加します。詳細な手順については、「ロールの信頼ポリシーの変更」を参照してください。
2. このロールを引き受けるアクセス許可を持つ IAM ユーザーグループを作成します。

```
      {
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": "arn:aws:iam::[YOUR_ACCOUNT_ID]:role/gamekit_[game
title]_AchievementsAdminInvokeRole"
        }
```

3. IAM ユーザーにアクセス権を付与するには、そのユーザーをこの新しいユーザーグループに追加します。

AWS アカウント管理者向けのヒント

複数のチームメンバーによる AWS GameKit プロジェクトの AWS アカウントを管理する場合は、以下のベストプラクティスを検討します。

- AWS GameKit のユーザーをセットアップするときは、ユーザーに対してプログラムによるアクセスとコンソールによるアクセスの両方を有効にすることを検討します。AWS リソースの表示、問題のトラブルシューティング、およびその他の理由で、ユーザーが AWS Management Consoleの使用を希望する可能性があります。
- IAM ユーザーグループを使用して、チームメンバーのアクセス許可レベルを管理します。ユーザーグループでは、個々のユーザーではなく、グループのアクセス許可ポリシーを設定します。アクセスレベルの異なる個別のユーザーグループを作成して、適切なアクセス許可を持つグループに個々のユーザーを追加することができます。
- デフォルトの AWS GameKit アクセス許可ポリシーテンプレートには、AWS GameKit ソリューションが使用するすべての AWS のサービスとリソースへの包括的なアクセス権が含まれています。カスタムポリシーを作成して適用することで、ユーザーアクセスを調整することができます。例えば、ゲーム機能の AWS リソースを表示することはできて、デプロイや削除はできないように、ユーザーのアクセス権を変更することができます。AWS GameKit ソリューションには、複雑な AWS サービスのコレクションが関係することに留意してください。アクセス許可の変更が、AWS GameKit の機能を扱うユーザーの能力にどのように影響するかは、必ずしも容易にわかるとは限りません。

- 環境ステージごとにアクセスレベルを管理することを検討します。AWS GameKit ユーザーは、作業する環境ごとにアカウント認証情報を提供する必要があります。例えば、開発ステージではすべてのアクションを許可し、本番ではアクセスを制限することができます。

AWS のセキュリティ認証情報の取得

概要

AWS GameKit ユーザーがプラグインを使用するには、AWS アカウントとセキュリティ認証情報が必要です。このトピックは、プラグインユーザーまたは AWS アカウント管理者が、AWS ユーザーの認証情報を取得するために役立ちます。

AWS GameKit ユーザーは、自分の AWS ユーザーのセキュリティ認証情報を使用してプラグインにサインインする必要があります。この認証情報で、ゲームのクラウドバックエンドをプラグインから直接作成および管理できるように、AWS へのプログラムによるアクセスを承認します。

既存の AWS ユーザーのセキュリティ認証情報を取得するには、以下の手順を使用します。AWS GameKit アクセス権を持つ新しいユーザーを作成するには、「AWS GameKit の AWS アカウントのセットアップ」を参照してください。

セキュリティ認証情報の取得

AWS ユーザーは、セキュリティ認証情報をローカルに保存する必要があります。以下の場所で既存のセキュリティ認証情報を探します。

- `create_IAM_user.py` スクリプト (AWS GameKit プラグインのダウンロードに含まれています) で作成された AWS ユーザーの場合、スクリプトはユーザーのセキュリティ認証情報を生成し、`[username]_credentials.txt` ファイルに保存します。スクリプトを実行した場合、このファイルがローカルマシンの `...\policies\` ディレクトリに保存されます。
- AWS Management Consoleからセキュリティ認証情報を生成する場合、認証情報を *<user name>*`_accessKeys.csv` という名前のローカルファイルにダウンロードできます。
- AWS GameKit プラグインやその他の AWS のプログラムツールで認証情報を使用している場合、その認証情報がホームディレクトリ (例: `C:\Users\`**`<user ID>`**`\.aws\credentials`) に保存されていることがあります。

新しいセキュリティ認証情報の生成

有効なセキュリティ認証情報がない場合や、既存の認証情報を紛失した場合は、以下の手順に従って AWS ユーザー用の新しい認証情報を作成します。

> **Note**
>
> `create_IAM_user` スクリプト (AWS GameKit プラグインのダウンロードに含まれています) で作成された AWS ユーザーの場合は、AWS Identity and Access Management (IAM) ユーザータイプ用の手順を使用します。このようなユーザーには、短期的または長期的なアクセスキーを取得できます。

AWS Management Console の外部で AWS を操作するには、ユーザーはプログラムによるアクセスが必要です。プログラムによるアクセスを許可する方法は、AWS にアクセスしているユーザーのタイプによって異なります。

ユーザーにプログラムによるアクセス権を付与するには、以下のいずれかのオプションを選択します。

どのユーザーがプログラムによるアクセスを必要としますか?	To	By
ワークフォース ID (IAM Identity Center で管理されているユーザー)	一時的な認証情報を使用して、AWS CLI、AWS SDK、または AWS API へのプログラムによるリクエストに署名します。	使用するインターフェイス用の手順に従ってください。 • AWS CLI については、「AWS Command Line Interface ユーザーガイド」の「AWS IAM Identity Center を使用するための AWS CLI の設定」を参照してください。 • AWS SDK、ツール、および AWS API については、「AWS SDK とツールリファレンスガイド」

どのユーザーがプログラムによるアクセスを必要としますか?	To	By
		の「IAM Identity Center 認証」を参照してください。
IAM	一時的な認証情報を使用して、AWS CLI、AWS SDK、または AWS API へのプログラムによるリクエストに署名します。	「IAM ユーザーガイド」の「AWS リソースでの一時的な認証情報の使用」の指示に従ってください。
IAM	(非推奨) 長期的な認証情報を使用して、AWS CLI、AWS SDK、AWS API へのプログラムによるリクエストに署名します。	使用するインターフェイス用の手順に従ってください。 • AWS CLI については、「AWS Command Line Interface ユーザーガイド」の「IAM ユーザー認証情報を使用した認証」を参照してください。 • AWS SDK とツールについては、「AWS SDK とツールリファレンスガイド」の「長期認証情報を使用して認証する」を参照してください。 • AWS API については、「IAM ユーザーガイド」の「IAM ユーザーのアクセスキーの管理」を参照してください。

AWS GameKit プラグインによる認証情報の保護

AWS アカウントの認証情報を AWS GameKit プラグインに入力しても安全です。AWS GameKit が認証情報をゲームプロジェクトや AWS GameKit の設定ファイルに保存することはありません。認証情報はゲームの配布物には含まれません。

AWS GameKit プラグインは、ゲームプロジェクトのセットアップ中に AWS 認証情報を要求します。デフォルトでは、認証情報はローカルにキャッシュされるため、再入力する必要はありません。環境を切り替えるときなど、いつでも別の認証情報を入力できます。

AWS GameKit で認証情報を保護するためのヒント:

- 公式ソース以外から AWS GameKit プラグインをダウンロードしないでください。
- テストコードでの便宜上であっても、ゲームコードには認証情報を入力しないでください。
- 共有されるファイルにローカルで認証情報を保存することは避けてください。AWS GameKit プラグインオプションの [認証情報の保存] を使用すると、認証情報がホームディレクトリ (`~/.aws/credentials`) に保存されます。この標準の場所は、他の AWS ツールでも使用されます。

AWS 関連トピック

- 「Understanding and Getting your AWS credentials, Programmatic access」、「AWS 全般のリファレンス」
- 「AWS アクセスキーを管理するためのベストプラクティス」、「AWS 全般のリファレンス」
- 「Where are configuration settings/credential information stored?」、「AWS Command Line Interfaceユーザーガイド」

AWS GameKit の開始方法

このセクションのコンテンツを使用して、AWS GameKit for Unreal について確認し、クラウドベースのゲーム機能を Unreal プロジェクトに統合する方法について説明します。

トピック

Unreal Editor で AWS GameKit を試してみる

概要

この概要では、ゲーム開発者向けに Unreal Editor 用の AWS GameKit プラグインのツールおよび機能について簡単に説明します。

AWS GameKit プラグインを有効にした状態で Unreal プロジェクトで作業する場合、AWS GameKit コンポーネントを使用して次のことができます。

- the section called “クラウドプロジェクトを管理する” — Unreal プロジェクト用に AWS GameKit 設定をセットアップし、AWS アカウントにリンクして、ステージング環境を管理します。
- the section called “クラウド機能のバックエンドサービスをデプロイする” — 各ゲーム機能のバックエンドを設定してデプロイします。
- the section called “AWS GameKit の機能をフロントエンドに組み込む” — サンプル素材を使用して、ゲーム内のクラウド機能を試したり、ラピッドプロトタイピングを行ったりできます。

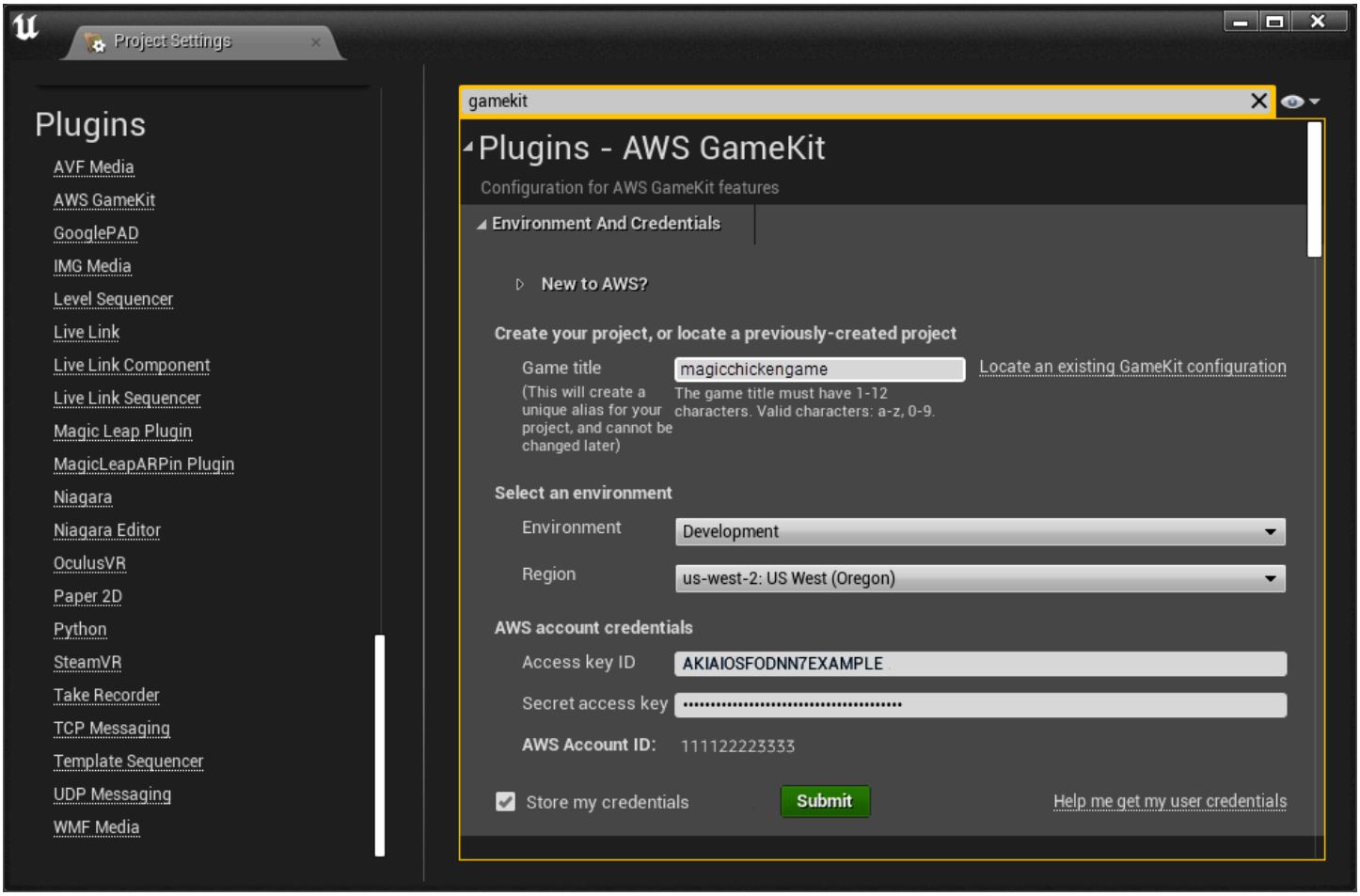
クラウドプロジェクトを管理する

各 Unreal プロジェクトには、Unreal プロジェクトの AWS 上のバックエンドサービスを管理するためのコンパニオン AWS GameKit クラウドプロジェクトが必要です。クラウドプロジェクトには、各クラウド機能の設定とコードが含まれます。AWS GameKit は、これらの機能の AWS リソースをデプロイしたり操作したりするときに、この情報を使用します。

クラウドプロジェクト設定を検索する

1. Unreal Editor ツールバーで、[編集] > [プロジェクト設定] を開き、[AwsGameKit] プラグインセクションに移動します。

2. 次のスクリーンショットに示すように、[環境と認証情報] セクションを展開します。



クラウドプロジェクト設定を使用する

> **Note**
>
> 新しい AWS IAM Identity Center で作成されたユーザーの AWS 認証情報を使用すると、異常な動作やメッセージングが発生することがあります。これらの認証情報は短期間のアクセスのみを提供します。つまり、定期的に新しい認証情報を生成してプラグインに入力する必要があります。AWS GameKit ダウンロードパッケージに付属する自動スクリプトを使用してユーザーを作成した場合、それらのユーザーの AWS 認証情報は長期アクセスキーとなります。

- クラウドプロジェクトを設定します。クラウドプロジェクトを設定する場合は、次の情報を指定します。

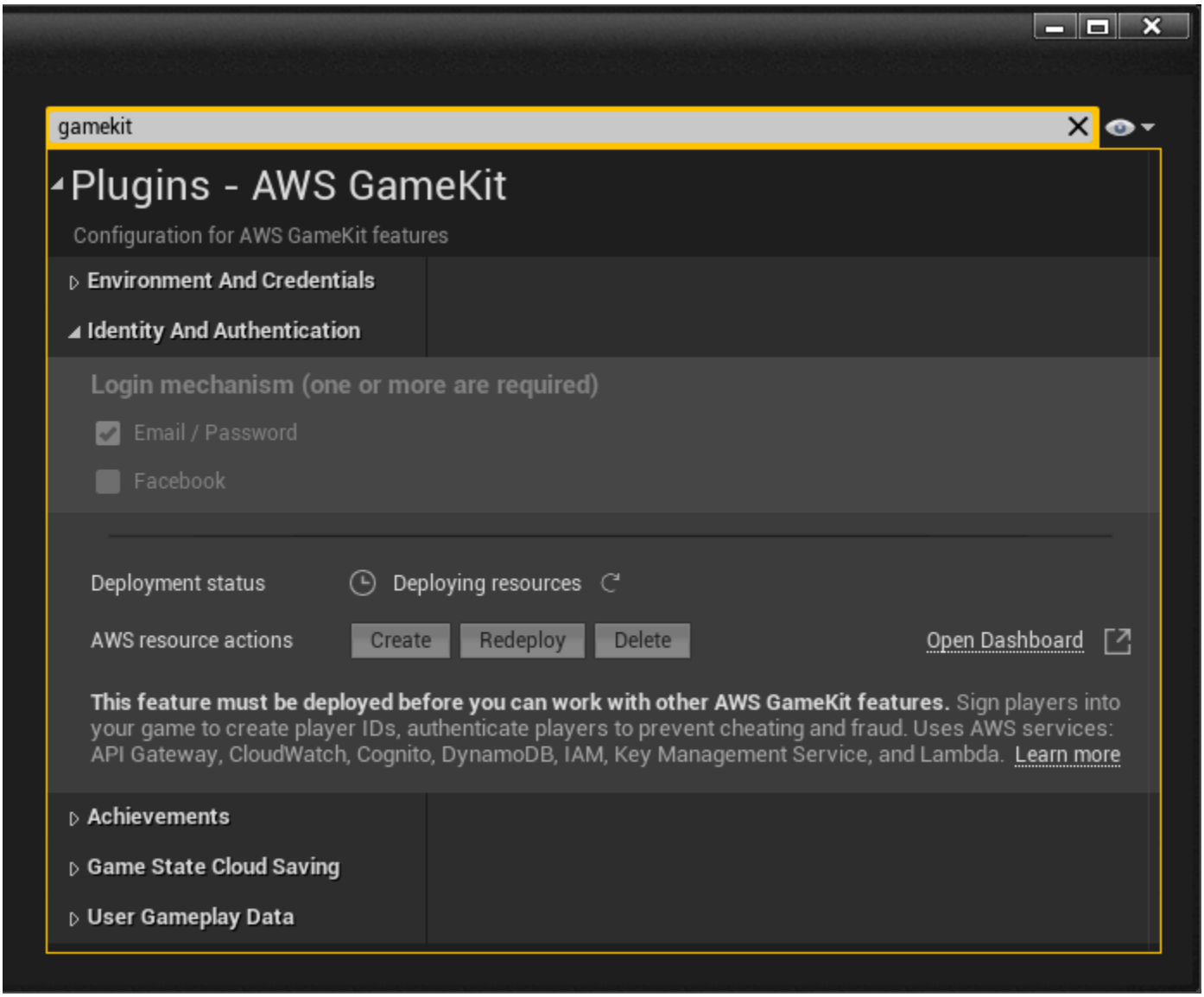
  - ゲームタイトルは、クラウドプロジェクトの固有のエイリアスです。AWS GameKit は、Unreal プロジェクトフォルダ内のこの名前のフォルダにクラウドプロジェクト設定およびコードを格納し、プロジェクトにデプロイされたすべての AWS リソースでこのゲームタイトルを参照します。
  - 開発者は、複数の環境により、開発用、テスト用、本番用など、プロジェクトのバックエンドの複数のレプリカを並行して管理できます。ユーザーは、作業するアクティブな環境を設定します。プロジェクトチームは通常、デフォルトの [開発] 環境から始めます。
  - リージョンは、AWS GameKit が環境の AWS リソースをデプロイする物理的な場所を指定します。各環境には 1 つの指定リージョンがあります。
  - AWS アカウント認証情報は、プロジェクトに使用する AWS アカウントを識別し、アカウントユーザーを検証します。アカウントが有効で、ユーザーが AWS GameKit のアクセス権を持っている場合は、プロジェクトのバックエンドの AWS リソースをデプロイまたは更新することができます。
- 既存のプロジェクトを使用します。ゲームタイトルを指定し、作業する環境を選択して、AWS セキュリティ認証情報を送信します。
- 環境を切り替えます。一度にアクティブにできるクラウドプロジェクト環境は 1 つだけです。クラウド機能およびデプロイされたバックエンドリソースに対して行うすべてのアクションは、アクティブな環境にのみ影響します。このページで別の環境に切り替えることができます。セキュリティ認証情報の再送信が必要な場合もあります。
- アカウント認証情報を変更します。AWS GameKit は、現在のデバイスで前回のセッションのセキュリティ認証情報を自動的に再利用します。認証情報が変更された場合は、クラウドプロジェクト設定を使用して新しい認証情報を送信してください。
- デプロイリージョンを変更します。アクティブな環境の AWS リソースをデプロイする場所は変更した方がいいかもしれません。環境に AWS リソースがデプロイされている場合、デプロイリージョンは変更できません。

クラウド機能のバックエンドサービスをデプロイする

ユーザーは、機能設定で、AWS GameKit の各機能のクラウド接続バックエンドを構築できます。ユーザーは、作業する環境を選択し、認証情報を送信した後で、機能を設定し、その機能のバックエンドを実行する AWS リソースを作成または更新することができます。AWS GameKit は、各環境の機能設定とデプロイステータスをクラウドプロジェクトで管理します。

ゲーム機能の設定を検索する

1. Unreal Editor ツールバーで、[編集] > [プロジェクト設定] を開き、[AwsGameKit] プラグインセクションに移動します。
2. 各ゲーム機能のセクションを展開します。例えば、次のスクリーンショットは、ID と認証の機能の設定を示しています。



機能設定を使用する

- デプロイされた機能のステータスを表示します。各機能セクションには、機能の現在のデプロイステータスが表示されます。構成設定には、デプロイされた機能のバックエンドが反映されます。機能がまだデプロイされていない場合は、デフォルト値が表示されます。

- 機能設定を行います。各機能には、機能のバックエンドをデプロイする前にカスタマイズできる設定があります。例えば、ID と認証では、登録済みのユーザー名とパスワード、または Facebook アカウント、あるいはその両方を使用してプレイヤーがログインできるように選択することができます。
- AWS リソースをデプロイ、更新、または削除します。機能設定を選択したら、その機能の新しい AWS リソースを作成できます。既にデプロイされている場合は更新することができます。デプロイされたリソースを削除することで、機能を削除することもできます。
- 機能ダッシュボードにアクセスします。AWS GameKit の各機能のメトリクスダッシュボードをアクティブ化できます。各ダッシュボードには、その機能が使用する AWS サービスとリソースの運用メトリクスが含まれています。例えば、バックエンドの API コールをモニタリングできます。ダッシュボードは Amazon CloudWatch サービスを使用します。

AWS GameKit の機能をフロントエンドに組み込む

AWS GameKit プラグインは、Unreal プロジェクトのフロントエンドを AWS 上のバックエンドサービスに接続するためのツールを提供します。これには、各機能のバックエンドサービスと接続するための AWS GameKit API や、フロントエンドの API コールとコアワークフローを説明するサンプルアセットが含まれます。

AWS で機能のバックエンドサービスをデプロイすると、現在アクティブな AWS GameKit 環境のバックエンドエンドポイントで Unreal プロジェクトが自動的に設定されます。プロジェクトのバックエンドと通信するサンプルアセットを使用して、ライブ API コールを行い、実行レベルを作成することができます。

AWS GameKit のアセットとツールを検索する

Unreal Editor で [コンテンツブラウザ] を開き、以下のフォルダを探します。

- `AwsGameKit Content` には、各ゲーム機能のサンプルのブループリント、UI 要素、ウィジェットブループリントが含まれています。ユーザーゲームプレイデータ機能の完全なゲームサンプルもあります。
- `AwsGameKit C++ Classes` には、C++ コードを使用してゲーム機能をゲームに統合するためのサンプルの C++ コードとリソースが含まれています。
  - `AwsGameKitEditor` パブリックフォルダには、各ゲーム機能のサンプルのコードファイルが含まれています。このコードには、ゲーム機能のすべての API オペレーションの関数呼び出しが含まれています。

- `AWSGameKitRuntime` パブリックフォルダには、サンプルアセットをサポートする関数ライブラリとユーティリティが含まれています。

プロジェクトのバックエンドの呼び出しをテストする

AWS GameKit の機能のいずれかのバックエンドをデプロイしている場合は、Unreal Editor でそのバックエンドを直接呼び出すことができます。

1. [コンテンツブラウザ] で `AwsGameKitEditor` パブリックフォルダを開き、サンプルコードファイルを選択します。
2. アセットをプロジェクトのレベルに追加し、ビューポートでオブジェクトを選択します。
3. [詳細] パネルを開きます。カスタムの AWS GameKit [詳細] UI には、サンプルコードファイル内のすべての API コールが表示されます。
4. プレイヤーアカウントでログインします。まだアカウントを設定していない場合は、ID と認証のサンプルファイルを使用してアカウントを登録し、検証します。
5. アチーブメントのサンプルコードを使用するときは、提供されている関数を使って、サンプルのアチーブメント定義セットをアチーブメントのバックエンドに保存します。このデータを使用してアチーブメント機能を試して、必要に応じて削除することができます。

AWS GameKit 機能のゲームへの統合

このトピックでは、AWS GameKit を使用して AWS GameKit のクラウドベースのゲーム機能を構築する場合の実装手順を概説します。各ステップには詳細なドキュメントへのリンクがあります。

1. ゲームエンジン用の AWS GameKit を入手します。
   - 開発環境用のプラグインをダウンロードします。
   - AWS GameKit プラグインをインストールします。詳細については、「Unreal Engine での AWS GameKit プラグインのインストール」を参照してください。
2. AWS アカウントを取得し、ゲームプロジェクトのユーザーアカウントを設定します。
   - AWS アカウント を作成するか、既存のアカウントを使用します。このアカウントを使用して、すべての AWS GameKit のリソースとサービスを管理します。詳細については、「AWS アカウントへのサインアップ」を参照してください。
   - AWS GameKit プラグインを使用するすべてのユーザーに AWS ユーザーをセットアップします。このステップには、AWS GameKit のアクセス許可を追加することも含まれます。各ユーザーには、プラグインで使用するセキュリティ認証情報と、(オプションで) AWS Management

Console を使用するためのサインイン認証情報が必要です。詳細については、「AWS GameKit アクセス権を持つユーザーのセットアップ」を参照してください。

3. AWS GameKit クラウドプロジェクトをセットアップします。

   - ゲームエンジンの AWS GameKit UI を使用して、ゲームのタイトル/エイリアス、作業環境、バックエンドサービスの場所などのクラウドプロジェクトの設定を定義します。このステップでは、AWS GameKit が AWS でのバックエンドの構築に使用する設定とテンプレートのセットを作成します。詳細については、「 ゲーム用の AWS GameKit プラグインのセットアップ」を参照してください。
   - ユーザーセキュリティ認証情報を入力して、プロジェクトを AWS アカウントにリンクします。

4. ID と認証のゲーム機能を設定し、AWS リソースをデプロイします。

   その他の AWS GameKit 機能にはすべて ID と認証が必要なので、他の機能を設定する前に、この機能のバックエンドを作成する必要があります。詳細については、 プロジェクトに ID と認証を追加する を参照してください。

   - AWS GameKit プラグインのプロジェクト設定を使用して、ゲームの ID と認証機能を設定します。
   - 認証バックエンド用の AWS リソースを作成します。デプロイプロセスを追跡し、AWS GameKit UI からリソースを再デプロイまたは削除できます。

     > **Note**
     >
     > アカウントが AWS 無料利用枠の特典を受ける資格があるかどうかによっては、AWS リソースのデプロイ時に AWS の料金が発生し始める場合があります。

   - オプションで、新しくデプロイした ID と認証のバックエンドの運用メトリクスダッシュボードを有効にします。詳細については、「 ゲーム機能のダッシュボードを使用する」を参照してください。

5. ID と認証のワークフローをゲームのフロントエンドに追加します。

   ゲームのフロントエンドにワークフローを作成して、プレイヤーがゲームに登録したり、ログインやログアウトを行ったり、その他の ID 関連のタスクを行えるようにします。ID と認証のバックエンドが用意できたら、AWS GameKit のサンプル資料を使用してワークフローのプロトタイプを作成し、テストできます。詳細については、「 ID と認証のサンプルを使用する」を参照してください。

6. AWS GameKit の追加機能用にゲームのバックエンドを設定してデプロイし、ゲーム機能を接続します。
   - アチーブメント
   - ユーザーゲームプレイデータ
   - ゲーム状態のクラウド保存

AWS GameKit UI での作業

このセクションの内容では、AWS GameKit プラグインインターフェイスを使用して、AWS クラウドサービスに裏づけられたクラウドベースのゲーム機能を構築する方法について詳しく説明します。

トピック

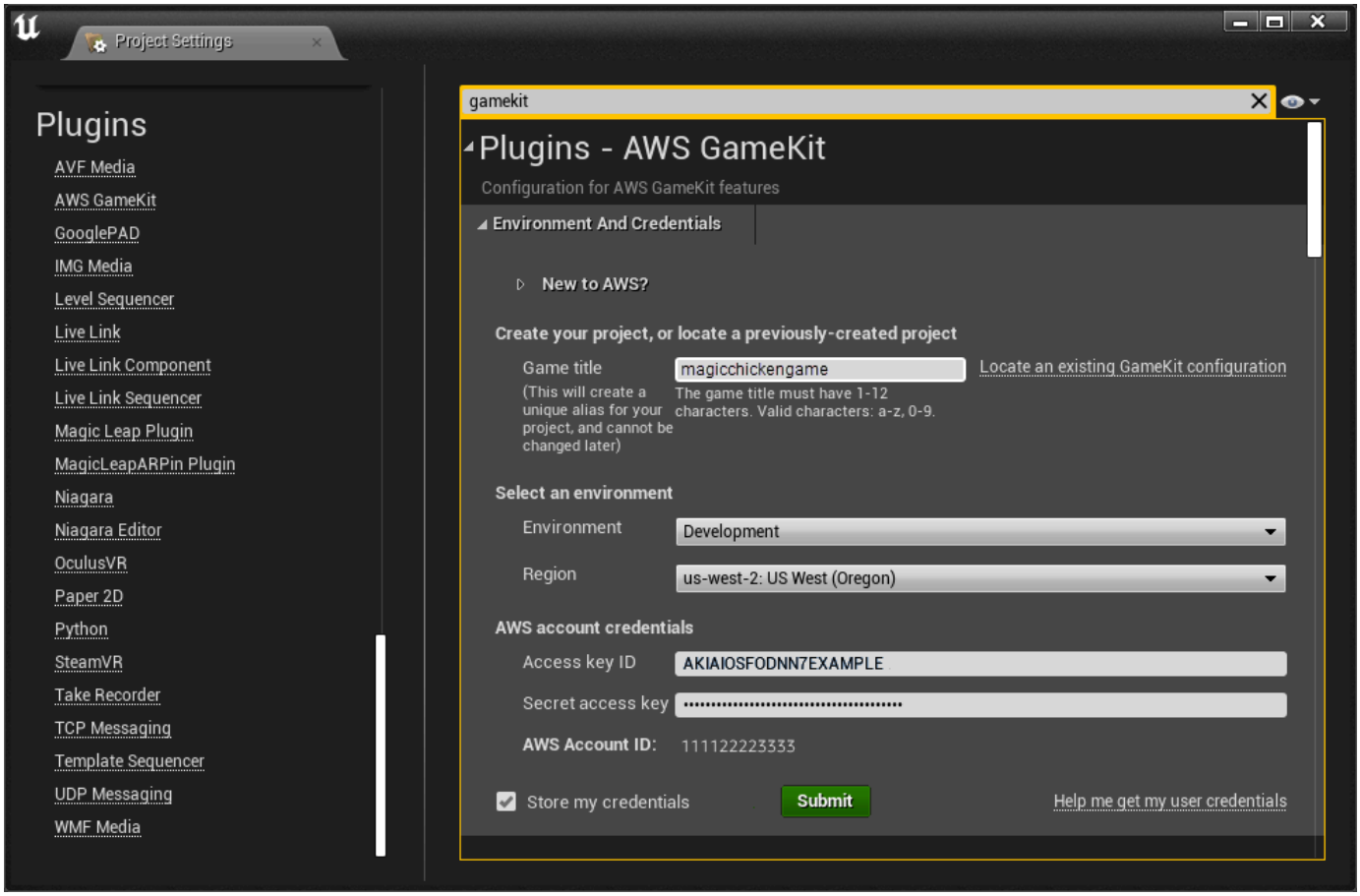
ゲーム用の AWS GameKit プラグインのセットアップ

AWS GameKit プラグインを使用してゲームの AWS クラウドベースのサービスを構築して維持する前に、AWS GameKit 設定と AWS アカウントを使用してゲームプロジェクトをセットアップする必要があります。

> **Note**
>
> ゲーム用の AWS GameKit 設定を既に作成している場合、2 つ目の同時の設定を作成することはできません。

AWS GameKit 設定をセットアップするには

1. Unreal Editor ツールバーで、[編集] > [プロジェクト設定] を開き、[AwsGameKit] プラグインセクションに移動します。
2. [環境と認証情報] を展開します。



3. 次のいずれかを実行します。

   - 新しい設定を作成するには、新しい [ゲームタイトル] を入力します。ゲームタイトルの文字列は、すべて小文字の英数字でなければなりません。スペースや特殊文字を含めることはできません。

     AWS GameKit は、ゲームプロジェクトのコンポーネント名にこのゲームタイトルを使用します。これには、ゲームプロジェクトファイルと共にローカルに保存される設定フォルダや、このゲームプロジェクトにデプロイするすべての AWS リソースが含まれます。

     > **Warning**
     >
     > 次の文字列はゲームのバックエンドサービスを作成する際に問題の原因となる可能性があるため、ゲームタイトルには使用しないでください。
     >
     > - `aws`
     > - `amazon`

ゲームタイトルを送信してゲーム機能での作業を開始した後は、ゲームタイトルを変更することはできません。後からゲームタイトルを変更する必要が生じた場合は、すべてのバックエンドサービスを含むゲームの AWS GameKit 設定全体を削除し、新しい設定を開始する必要があります。

- 以前に作成した設定を使用するには、[既存の GameKit 設定を探す] を選択します。次に、ゲームプロジェクトファイルを参照し、AWS GameKit 設定フォルダを検索します。

以降のセッションでは、最後に使用されたゲームタイトルが AWS GameKit プラグインにより自動的に入力されます。

4. 作業する [環境] を選択します。デフォルト環境 ([開発]、[テスト]、または [本番]) を選択するか、カスタム環境を作成することができます。

   以降のセッションでは、最後に使用した環境が AWS GameKit プラグインにより自動的に選択されます。環境はいつでも切り替えることができます。

   ゲームプロジェクトの AWS GameKit 設定では、環境ごとに個別の情報セットが維持されます。作業する環境を選択すると、選択した環境のゲーム機能設定と AWS デプロイが AWS GameKit プラグインによりロードされます。

5. 選択した環境で、ゲームのバックエンドのリソースを配置する AWS [リージョン] を選択します。各環境で 1 つのリージョンにのみリソースをデプロイできます。

   その環境に最も適したリージョンを選択します。例えば、開発環境のリソースは地理的に開発チームに近いリージョンに配置し、本番のリソースはプレイヤーの近くに配置することができます。追加するゲーム機能のリージョンサポートを確認するには、「AWS GameKit がサポートされている AWS リージョン」を参照してください。

   以降のセッションでは、選択した環境のリージョンが AWS GameKit プラグインにより自動的に使用されます。

6. [AWS アカウント認証情報] で、2 つの部分に分かれた AWS Identity and Access Management (IAM) ユーザーアクセスキーを入力します。このキーには、[アクセスキー ID] と [シークレットアクセスキー] が含まれています。これらの認証情報により、AWS アカウント内の固有の IAM ユーザーが一意に識別されます。詳細については、「AWS のセキュリティ認証情報の取得」を参照してください。

> **Important**
>
> IAM ユーザーがゲームのバックエンドの AWS リソースで作業をするには、アクセス許可が必要です。許可がない場合は、AWS GameKit プラグインでユーザー認証情報を正常に送信できますが、ゲーム機能の AWS リソースを作成、再デプロイ、または削除することはできません。詳細については、「AWS GameKit の AWS アカウントのセットアップ」を参照してください。

AWS Command Line Interface (AWS CLI) を既にインストールしていて、同じ認証情報で設定している場合、その認証情報が AWS GameKit プラグインにより自動的に認識されて使用されます。

7. オプションで、[認証情報の保存] を選択して現在のデバイスに AWS アカウントの認証情報を保存し、以降のセッションで使用できるようにします。認証情報を保存しない場合は、プラグインの各セッションの開始時や、環境を切り替えるたびに、認証情報を再入力する必要があります。AWS GameKit プラグインのセキュリティに関する質問については、「AWS GameKit プラグインによる認証情報の保護」を参照してください。

   AWS GameKit プラグインは、認証情報をホームディレクトリのテキストファイル (`C:\Users\<user ID>\.aws\credentials`) にローカルに保存します。保存された認証情報の各セットは、ゲームタイトルと環境に関連付けられます。これにより、環境ごとに異なる認証情報を使用できるため、ゲームの AWS リソースへのアクセスを厳密に制御するチームに役立ちます。

8. [送信] を選択して新しい AWS GameKit 設定を作成し、アクティブな環境を設定します。

   以降は、このゲームプロジェクトを Unreal Editor で開くと、最後に使用したゲームタイトル、環境、リージョン、認証情報 (保存している場合) が AWS GameKit プラグインにより自動的に選択されます。環境または認証情報は、AWS GameKit のプロジェクト設定 ([環境と認証情報]) でいつでも切り替えることができます。

ゲームプロジェクトからの AWS GameKit の削除

概要

このトピックの手順を使用して、ゲームプロジェクトから一部またはすべての AWS GameKit ゲーム機能を削除します。使用しなくなった機能に対して引き続き料金が発生しないようにするには、デプロイされたすべての AWS クラウドリソースを削除する必要があります。このトピックは、ゲームプ

ロジェクトから不要なすべての AWS GameKit コンポーネントが削除されたことを確認するゲーム開発者を対象としています。

以下のプロセスのいずれかを使用して、ゲームプロジェクトから AWS GameKit コンポーネントを削除することができます。

- プロジェクトから個々のゲーム機能を削除します。
- プロジェクトからすべての AWS GameKit コンポーネントを削除します。

個々のゲーム機能の削除

既存のゲーム機能の設定を新しい設定に置き換える場合、デプロイした AWS リソースを削除する必要はありません。代わりに、AWS GameKit プラグイン UI を使用して設定を変更してから、デプロイされた AWS リソースを更新します。

ゲームからゲーム機能を削除する場合は、以下のタスクを行います。

- ゲームのフロントエンドで、ゲームのバックエンドに依存するすべての機能を削除します。このような機能は、バックエンドサービスを実行する AWS リソースを削除するとすぐに動作しなくなるためです。
- 削除するゲーム機能に関連するデータをバックアップすることを検討します。これには、アチーブメント定義などの設定データや、ゲームで生成されたプレイヤーデータが含まれる場合があります。
- 以下の手順で説明されているように、ゲーム機能のバックエンドサービスを実行する AWS リソースを削除します。

> **Note**
>
> 他の AWS GameKit ゲーム機能を使用している場合は、ID と認証の機能は削除できません。他のすべての機能が、プレイヤー認証プロセスを使用します。

Unreal Engine

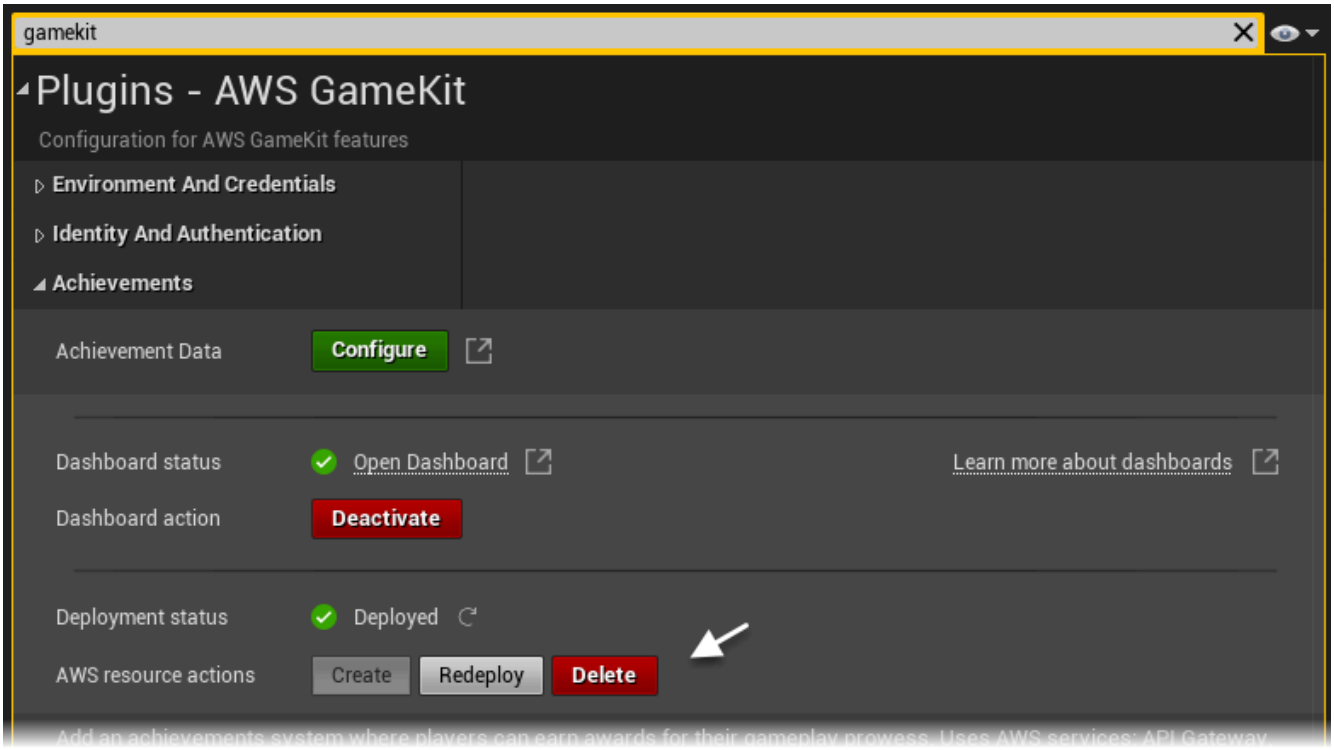
AWS GameKit ゲーム機能を削除するには

1. Unreal Editor ツールバーで、[編集] > [プロジェクト設定] を開き、[AwsGameKit] プラグインセクションに移動します。

2. [環境と認証情報] セクションで、ゲーム機能を削除する環境と AWS リージョンを選択します。適切な AWS 認証情報を入力し、[送信] を選択します。

> **Note**
>
> ゲーム機能を削除する環境とリージョンの組み合わせごとに、この手順を実行する必要があります。

3. 削除するゲーム機能のプロジェクト設定セクションを展開し、デプロイのステータスを確認します。ゲーム機能に AWS リソースがデプロイされている場合は、AWS リソースアクションの [削除] を選択します。このアクションは、ゲーム機能のバックエンドサービスとダッシュボードを提供するすべての AWS リソースを削除します。一部のログは保持される場合があります。



ゲーム機能の設定を削除または変更したり、ゲームプロジェクトと共にローカルに保存されている AWS GameKit ファイルを削除したりする必要はありません。

すべての AWS GameKit プラグインコンポーネントの削除

AWS GameKit を最初からやり直すには、以下の手順に従って既存の設定を削除します。その後、新しいデフォルトの AWS GameKit 設定で最初からやり直すことができます。

ゲームプロジェクトから AWS GameKit プラグインを削除するには、以下のタスクを行います。

- ゲームのフロントエンドで、ゲームのバックエンドに依存するすべての機能を削除します。このような機能は、バックエンドサービスを実行する AWS リソースを削除するとすぐに動作しなくなるためです。
- 現在ゲームで使用されているゲーム機能に関連するデータをバックアップすることを検討します。これには、アチーブメント定義などの設定データや、ゲームで生成されたプレイヤーデータが含まれる場合があります。
- 以下の手順で説明されているように、すべてのゲーム機能の現在デプロイされている AWS リソースを削除します。

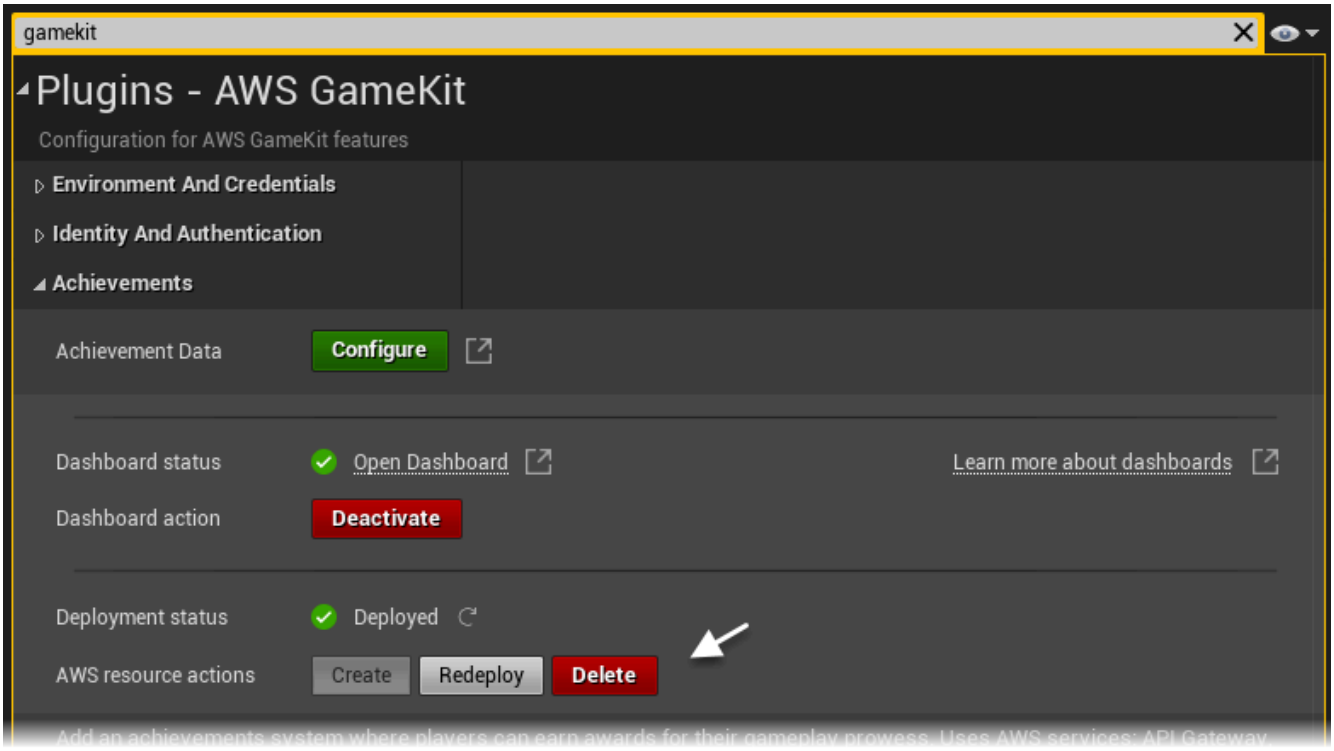
Unreal Engine

すべてのプラグインコンポーネントを削除するには

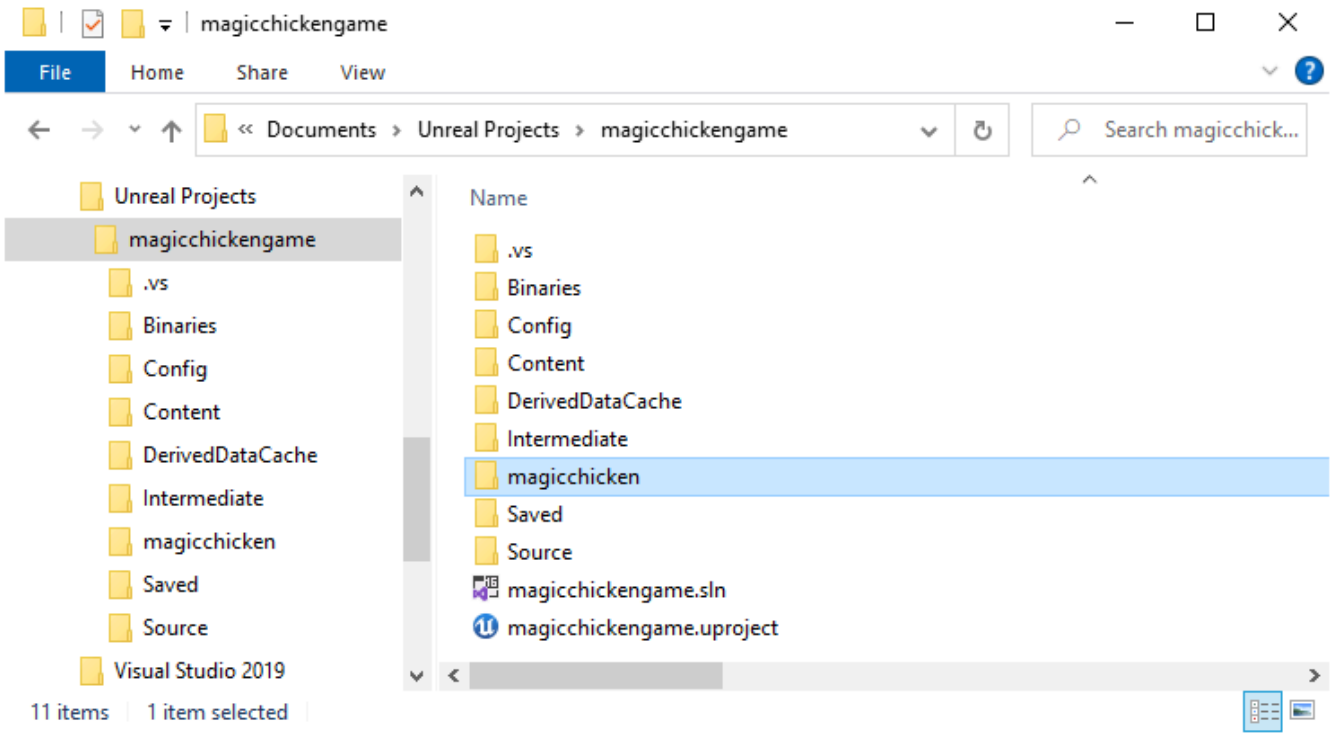
1. Unreal Editor ツールバーで、[編集] > [プロジェクト設定] を開き、[AwsGameKit] プラグインセクションに移動します。
2. [環境と認証情報] セクションで、1 つ以上のゲーム機能に AWS リソースが現在デプロイされている環境と AWS リージョンを選択します。適切な AWS 認証情報を入力し、[送信] を選択します。

   > **Note**
   >
   > AWS リソースをデプロイしたすべての環境とリージョンの組み合わせについて、この手順に従う必要があります。

3. AWS GameKit の各ゲーム機能について、プロジェクト設定セクションを展開し、デプロイのステータスを確認します。ゲーム機能に AWS リソースがデプロイされている場合は、AWS リソースアクションの [削除] を選択します。このアクションは、ゲーム機能のバックエンドサービスとダッシュボードを提供するすべての AWS リソースを削除します。一部のログは保持される場合があります。



4. AWS GameKit ゲーム機能の AWS リソースをデプロイしたすべての環境とリージョンの組み合わせについて、このプロセスを繰り返します。
5. ファイルブラウザで、ゲームプロジェクトファイルが含まれているディレクトリを開きます。AWS GameKit のファイルは、AWS GameKit のゲームタイトルが付けられた名前のフォルダにあります。次のスクリーンショットは、プロジェクト名が「magicchickengame」で、ゲームタイトルが「magicchicken」のプロジェクトのフォルダを示しています。このディレクトリを削除します。この手順により、AWS GameKit プラグインが各ゲーム機能のバックエンドサービスを管理するために使用するローカルにキャッシュされた設定やその他のサポートファイルがすべて削除されます。



6. Unreal Editor で、[編集] > [プラグイン] > [AwsGameKit] に移動し、ゲームプロジェクトのプラグインを無効にします。この手順により、ゲームプロジェクトのリリースパッケージに AWS GameKit コンポーネントが含まれなくなります。

ゲームプロジェクトの AWS GameKit プラグインは、いつでも再度有効にすることができます。その時点で、新しいゲームタイトルを入力し、ゲームの新しいデフォルト設定を作成するよう求められます。

AWS GameKit プラグインの問題のトラブルシューティング

このトピックでは、AWS GameKit プラグインの一般的な使用上の問題を解決するために役立つガイダンスを提供します。

[Unreal] AWS GameKit プラグインを有効にした後にゲームプロジェクトを開くことができない

問題: Unreal Editor で AWS GameKit プラグインを有効にして、プロンプトに従ってプログラムを再起動すると、Unreal Editor の再起動が試みられますが、次のメッセージが表示されて失敗します。

```
You need to rebuild your Unreal project in Visual Studio before using AWS GAMEKIT.
```

原因: この問題は、ゲームコードがまだ構築されていない C++ ゲームプロジェクトの AWS GameKit プラグインを有効にした場合に発生します。例えば、新しいゲームプロジェクトで AWS GameKit プラグインを有効にすると、この問題が発生する可能性があります。

解決方法: ゲームプロジェクトソリューションを手動で構築する必要があります。

1. Unreal ゲームプロジェクトのプロジェクトファイルが含まれているフォルダを開き、プロジェクトのソリューションファイル (`.sln`) を探します。
2. IDE でファイルを開き、ゲームプロジェクトコードを構築します。Visual Studio を使用している場合は、メニューツールバーから [ビルド] > [ソリューションのビルド] を選択します。
3. ソリューションが正常に構築された後は、Unreal Editor でゲームプロジェクトを開き、AWS GameKit プラグイン要素にフルアクセスすることができます。

[Unreal] デプロイが完了しない

問題: ゲーム機能の AWS リソースをデプロイすると、アクションが完了するまでに長い時間がかかります。

原因: AWS リソースをデプロイすると、背後で行われるセットアップアクティビティが完了するまでに時間がかかることがあります。各機能の平均デプロイ時間は以下のとおりです。

- ID と認証: 5 ~ 10 分
- アチーブメント: 10 ~ 30 分
- ユーザーのゲームプレイデータ: 5 ~ 10 分
- ゲームの状態のクラウド保存: 5 ~ 10 分

解決方法: デプロイアクティビティの進行状況を以下の方法で追跡できます。

- Unreal Editor の出力ログを使用して、詳細なイベントメッセージを表示します。ログを開くには、[ウィンドウ] > [開発者用ツール] > [出力ログ] に移動します。
- ゲーム機能のカスタムダッシュボードに移動します。機能のダッシュボードは、デプロイプロセスの開始時付近に生成されます。ダッシュボードを開くには、Unreal Editor で [編集] > [プロジェクト設定] > [プラグイン] > [AwsGameKit] に移動します。
- AWS Management Consoleで進行中のイベントを確認します。コンソールで AWS CloudFormation サービスに移動し、デプロイされているゲーム機能のスタックを開きます。

ゲーム機能のダッシュボードを使用する

概要

このトピックは、Unreal Engineプロジェクトのバックエンドアクティビティをモニタリングしたいと考えているゲーム開発者を対象としています。AWS GameKit の各機能について、カスタム Amazon CloudWatch メトリクスダッシュボードをアクティブ化できます。これらのダッシュボードを使用して、AWS GameKit を通じてセットアップされたバックエンド AWS サービスをモニタリングします。

AWS GameKit では、ゲームのバックエンド用に CloudWatch ダッシュボードをアクティブ化できます。ダッシュボードには、各ゲーム機能のバックエンドサービスを実行するデプロイ済み AWS リソースのメトリクスとアラームが含まれます。これらのダッシュボードは、各ゲーム機能全体の運用アクティビティに関するインサイトを提供し、通常、ゲームのテスト中、またはゲームにアクティブなプレイヤーベースがいる本番環境で使用されます。

ダッシュボードのメトリクスには、各 AWS サービスのコスト計算に使用されるメトリクスが含まれます。これらのメトリクスにより、ゲームのバックエンドのアクティビティがコストにどのように影響しているかを明確にすることができます。

CloudWatch ダッシュボードの詳細については、「AWS 関連トピック」を参照してください。

ダッシュボードをアクティブ化または非アクティブ化する

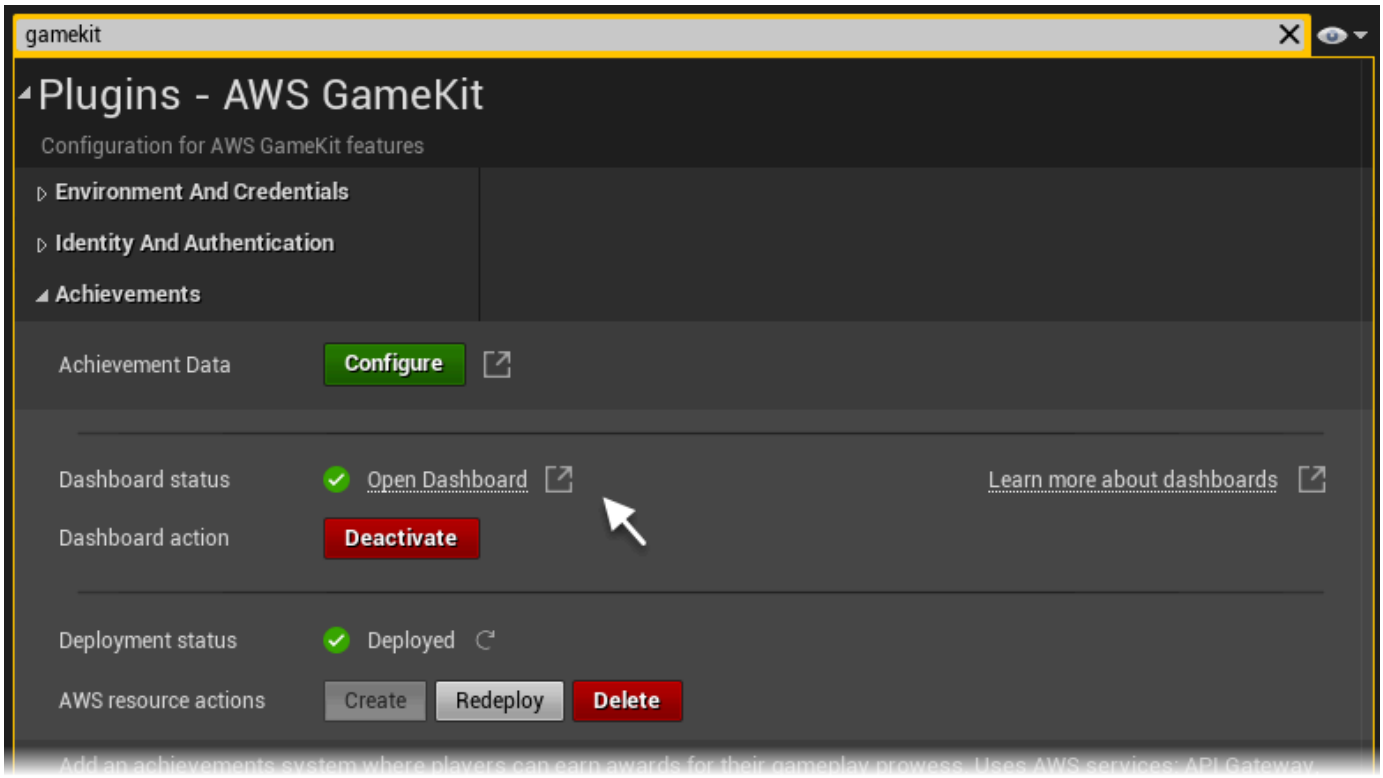
AWS GameKit ダッシュボードは、ゲーム機能のデプロイ済み AWS リソースのメトリクスを追跡し、ゲーム機能設定の一部としてダッシュボードのステータスを管理します。ゲーム機能を複数のステージング環境 (QA や本番など) にデプロイする場合は、各環境で機能のダッシュボードをアクティブ化または非アクティブ化します。機能のダッシュボードステータスはいつでも設定できます。

> **Note**
>
> CloudWatch の AWS 無料利用枠には、最大 3 つのダッシュボードが含まれます。複数のダッシュボードをアクティブ化すると、このクォータを超えて AWS アカウント で料金が発生する可能性があります。

ダッシュボードのアクティベーションステータスを設定するには

AWS GameKit 設定でダッシュボードのアクティベーションステータスを設定します。

1. Unreal Editor ツールバーで、[編集] > [プロジェクト設定] を開き、[AwsGameKit] プラグインセクションに移動します。[環境と認証情報] で、作業する環境を選択します。



2. ダッシュボードをアクティブ化または非アクティブ化するゲーム機能を開きます。各ゲーム機能セクションには、現在のダッシュボードステータスが表示されます。前のスクリーンショットでは、アチーブメント機能ダッシュボードがアクティブなダッシュボードと共にデプロイされています。

3. [ダッシュボードのアクション] では、[非アクティブ化] または [アクティブ化] を選択して、機能の現在のダッシュボードステータスを変更します。まだデプロイされていない機能のダッシュボードをアクティブ化した場合は、デプロイ中にダッシュボードが作成されます。機能が既にデプロイされている場合、AWS GameKit は機能のダッシュボードを作成または削除します。この更新はすぐに行われるため、機能を再デプロイする必要はありません。

   デフォルトのダッシュボードステータスは、アクティブな環境によって異なります。ダッシュボードは、[テスト] 環境と [本番稼働用] 環境では自動的にアクティブ化されます。

   変更は [デプロイのステータス] フィールドで追跡できます。

ダッシュボードを開く

デプロイされた機能のアクティブ化されたダッシュボードは、ゲームエンジンまたは CloudWatch の AWS Management Console からアクセスできます。

ゲームエンジンからダッシュボードを開くには

AWS GameKit 設定でゲーム機能のダッシュボードを開きます。

1. Unreal Editor ツールバーで、[編集] > [プロジェクト設定] を開き、[AwsGameKit] プラグインセクションに移動します。[環境と認証情報] で、作業する環境を選択します。
2. ダッシュボードを開きたい機能を開きます。機能にアクティブなダッシュボードがある場合は、[ダッシュボードのステータス] フィールドにチェックマークが表示されます。そうでない場合は、ダッシュボードをアクティブ化し、AWS GameKit によって作成されるまで待ってください。作成には数分ほどかかります。
3. [ダッシュボードを開く] を選択します。ダッシュボードは、CloudWatch の AWS Management Console のブラウザウィンドウ内で開きます。

ダッシュボードのコンテンツを表示する

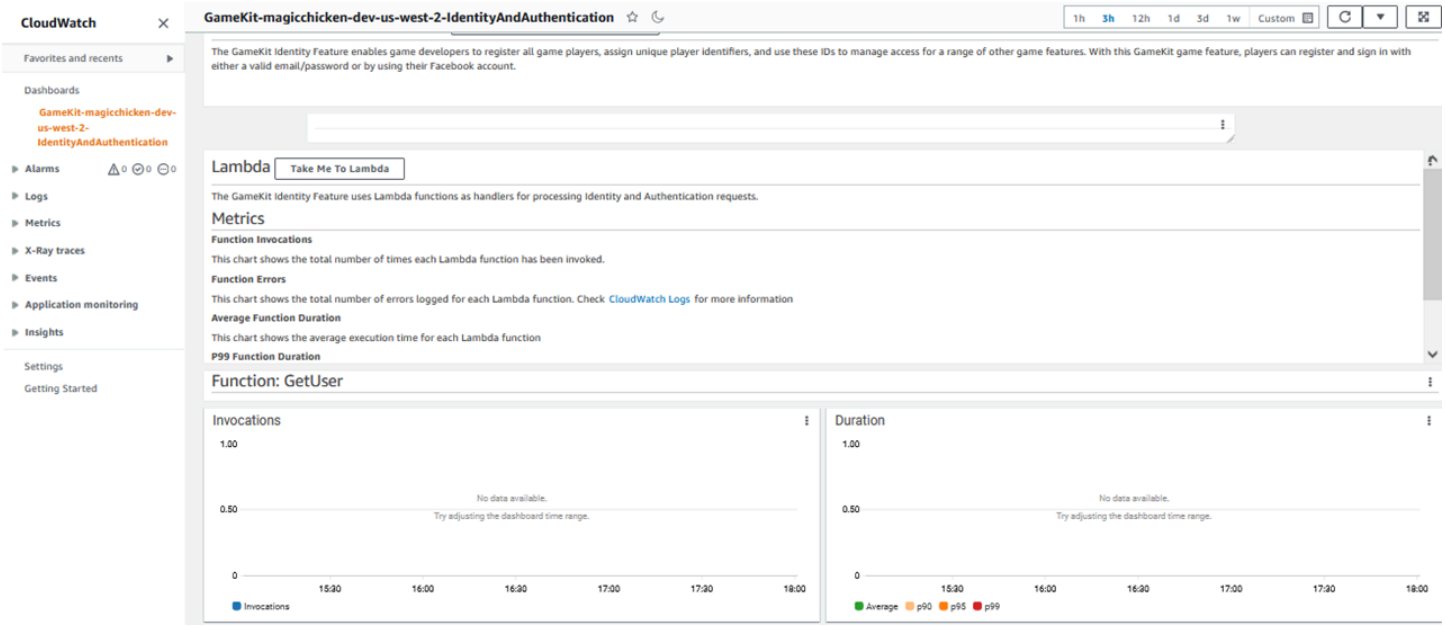
AWS GameKit の各機能には独自のダッシュボードがあります。ダッシュボード名には、AWS GameKit を通じてデプロイされる AWS リソースと同じ命名規則を使用します。例えば、`GameKit-magicchicken-dev-us-west-2-IdentityAndAuthentication` という名前のダッシュボードは、似たようなプレフィックスの付いた AWS リソースを追跡します。この名前は、ゲームプロジェクトの AWS GameKit 設定の次の情報を参照しています。

- エイリアス/ゲームタイトル (「magicchicken」に対応)
- デプロイ環境 (「dev」に対応)
- デプロイ AWS リージョン (「us-west-2」に対応)
- ゲーム機能 (「IdentityAndAuthentication」に対応)

次のスクリーンショットに示すように、ダッシュボードは CloudWatch の AWS Management Console で表示されます。ダッシュボードのデザインと形式は、AWS GameKit のバージョンによって異なります。

CloudWatch ダッシュボードツールを使用して AWS GameKit ダッシュボードをカスタマイズできます。これらの変更は、その特定のカスタムダッシュボードに適用され、そのダッシュボードを非アク

ティブ化すると失われます。永続的またはグローバルな変更を AWS GameKit ダッシュボードに適用する場合は、ゲームの AWS GameKit のベーステンプレートを更新します。



主なダッシュボードメトリクス

各ゲーム機能は、バックエンドのさまざまな AWS サービスコレクションを使用するため、各機能のダッシュボードでは、それぞれのメトリクスセットを追跡することになります。ただし、すべての AWS GameKit ダッシュボードには、以下のコアサービスからの情報が含まれています。

AWS Lambda

AWS GameKit のすべてのゲーム機能は、Lambda 関数を使用して、機能関連の API リクエストを処理します。各リクエストタイプの Lambda メトリクスには、呼び出しの数と、リクエストの処理に使用された平均コンピューティング時間が含まれます。特に注意すべきメトリクスには以下が含まれます。

- レイテンシー (P99、P95、P90)。これらのグラフは、バックエンドでのリクエストまたは操作が完了するまでにかかる時間を示しています。レイテンシーを評価する際、特定の「良い」値や「悪い」値はありません。代わりに、レスポンスタイムの低下につながる可能性のある急激な上昇を監視します。これを緩和するには、Lambda 関数の数を調整することを検討してください (「launch readiness guide」の「AWS Lambda の設定を更新する」を参照)。
- 同時実行数。これらのグラフでは、ゲームバックエンドが同時に実行する関数の数を追跡します。特に、同時実行数が一定である場合にレイテンシーが増加するかどうかを確認します。このシナリオは、同時実行の数だけでは負荷を処理するには不十分であることを示しています。これを緩

和するには、同時実行の許容数を増やすことを検討してください (「launch readiness guide」の「AWS Lambda の設定を更新する」を参照)。

- 関数エラー。これらのグラフは、API ごとのエラー数を示しています。これらのエラーの原因を理解するには、Amazon CloudWatch Logs を調べ、必要に応じて条件を緩和します。

Amazon API Gateway

ゲームクライアントから AWS GameKit API へのすべてのリクエストは、API Gateway を経由してバックエンドの Lambda 関数に渡されます。API Gateway のメトリクスには、一定期間に受信したリクエストの数と平均応答時間が含まれます。特に注意すべきメトリクスには以下が含まれます。

- レイテンシー (P99、P95、P90)。これらのグラフは、バックエンドでのリクエストまたは操作が完了するまでにかかる時間を示しています。レイテンシーを評価する際、特定の「良い」値や「悪い」値はありません。代わりに、レスポンスタイムの低下につながる可能性のある急激な上昇を監視します。このシナリオを緩和するには、基盤となるサービス (通常は Lambda 関数と DynamoDB) のレイテンシーグラフを調べ、各サービスの推奨事項に基づいて緩和します。
- 4XX および 5XX エラー。これらのグラフは、API Gateway リソースごとのエラー数を示しています。これらのエラーの原因を理解するには、Amazon CloudWatch Logs を調べ、必要に応じて条件を緩和します。

Amazon Cognito

AWS GameKit の ID と認証の機能を使用するゲームの場合、Amazon Cognito は、ユーザー名/パスワードを使用するか、Facebook などのサードパーティの ID プロバイダーを使用するかを問わず、すべてのプレイヤーの登録とログインを管理します。Amazon Cognito のメトリクスは、AWS GameKit API リクエストに関連するすべての認証イベントとセキュリティイベントを追跡します。特に注意すべきメトリクスには以下が含まれます。

- セキュリティ。このグラフは、発生したセキュリティ関連イベントの数を示しています。イベントの数が多い場合は、高度なセキュリティ機能を有効にすることを検討してください (「launch readiness guide」の「Amazon Cognito の設定を更新する」を参照)。

Amazon DynamoDB

DynamoDB テーブルには、プレイヤー ID、アチーブメント定義、ユーザーゲームプレイデータ、ゲームセーブ、プレイヤーアチーブメントステータスなど、AWS GameKit ゲーム機能に関するさまざまな情報が格納されます。DynamoDB のメトリクスには、消費キャパシティの割合、読み取り/書

き込みアクティビティ、リクエストレイテンシー、および個々のゲーム機能に関連するその他のメトリクスが含まれます。特に注意すべきメトリクスには以下が含まれます。

- スロットル。これらのグラフでは、スロットリングされたリクエストとイベントの数を追跡します。
- テーブルリクエストのレイテンシー。これらは、DynamoDB でのリクエストまたは操作が完了するまでにかかる時間を示しています。レイテンシーを評価する際、特定の「良い」値や「悪い」値はありません。代わりに、レスポンスタイムの低下につながる可能性のある急激な上昇を監視します。

スロットリングやレイテンシーの問題を軽減するには、DynamoDB 自動スケーリング機能を有効にすることを検討してください (「launch readiness guide」の「Amazon DynamoDB の設定を更新する」を参照)。

AWS 関連トピック

- 「Amazon CloudWatch ユーザーガイド」:「Amazon CloudWatch ダッシュボードの使用」

AWS GameKit 機能: ID と認証

概要

ID と認証のゲーム機能を使用すると、プレイヤーのサインインワークフローを設定し、各プレイヤーに一意の ID を割り当てることができます。プレイヤー ID は、ゲームクライアントとバックエンドサービス間の通信など、検証と認証を必要とするさまざまなシナリオをサポートします。他の AWS GameKit 機能を使用するには、ゲームの ID と認証の機能をデプロイする必要があります。それらはすべて、プレイヤーの認証機能に依存しています。主な対象者: ID と認証のゲーム機能のユースケースをより深く理解したいと思っているゲームの所有者や開発者。

AWS GameKit の ID と認証の機能には、一意のプレイヤー ID を作成および保存するためのツールが用意されています。AWS GameKit では、プレイヤー ID は、プレイヤーアクセスの管理、ゲームクライアントとバックエンドサービス間の通信の認証、および本人確認と認証を必要とするその他のシナリオに使用されます。

プレイヤーの ID と認証のための AWS GameKit ソリューションは、新規プレイヤー登録、プレイヤーサインイン、およびアカウント回復のワークフローをサポートします。E メール検証などの安全機能により、E メールやソーシャルメディアに基づいて新規プレイヤーを登録できます。

プレイヤー ID の想定される用途:

- プレイヤーが複数のデバイスでゲームにサインインし、中断したところからゲームを再開できるようにします。ユーザーゲームプレイデータ、アチーブメント、ゲームの状態のクラウド保存などの AWS GameKit クラウドベース機能は、プレイヤー固有の情報を追跡し、プレイヤーが使用しているどのようなデバイスにもその情報を配信します。この機能は、プレイヤーがいつでも、どのデバイスからでもゲームプレイを再開したいと考える長時間プレイするゲーム (ワールドビルダーやストーリー主導のゲームなど) で特に役立ちます。
- 認証済みのプレイヤー ID を必要とするリモートゲームサービス (マッチメイキング、マルチプレイヤーゲームサーバー、ソーシャルネットワークなど) とやり取りできます。データの検証や保存など、クラウドベースの他のゲーム機能を組み込みます。
- ゲーム管理ツールと連携して、ゲーム分析、ゲームプレイのカスタマイズ、ライブオペレーション、実験など、プレイヤーの行動に関するデータを収集して使用できます。

> **Note**
>
> その他の AWS GameKit 機能を使用するには、ID と認証の機能をゲームプロジェクトに追加する必要があります。それらはすべて、プレイヤーデータを取得または入力するためのクライアントリクエストの認証に依存しています。

トピック

ID と認証の仕組み

概要

ID と認証のゲーム機能を使用すると、ゲームにプレイヤーのサインインワークフローを構築する、一意のプレイヤー ID を生成し、それを使用して、ゲームプレイ中にプレイヤーの ID を検証する、プレイヤー固有のデータへのアクセスを管理する、プレイヤー認証を必要とするその他の機能を追加するといったことが可能になります。また、プレイヤーが E メールまたは有効な Facebook アカウントで登録できるようになります。主な対象者: ID と認証のゲーム機能が提供するものと、それをゲームに組み込むために必要な作業について大まかに理解したいと思っているゲームの所有者や開発者。

ID と認証のゲーム機能の主なメカニズムは、一意のプレイヤー ID です。プレイヤーは、ゲームに登録することで、検証済み ID を確立し、ゲームで使用するための一意のプレイヤー ID を取得します。プレイヤーがゲームにサインインすると、AWS GameKit はこの ID を使用して、プレイヤーが使用しているゲーム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を認証します。

AWS GameKit のすべての機能は、ID と認証を利用して、ゲームクライアントからのプレイヤー固有のリクエストが認証されていることを検証します。これらのゲーム機能は、ゲームプレイデータ、アチーブメント、ゲーム状態の保存など、プレイヤー固有のデータをプレイヤー ID 別にクラウドに保存します。

AWS GameKit での ID と認証では、シンプルで安全なワークフローが使用されます。プレイヤーの登録と一意のプレイヤー ID の確立には、次の方法のいずれかまたは両方を実装できます。

- プレイヤーは E メールアドレスとパスワードを指定できます。サインインのワークフローには E メール検証が含まれており、プレイヤーは登録を完了するためにこれに返答する必要があります。また、パスワード回復ワークフローも含まれています。
- プレイヤーは Facebook をサードパーティの ID プロバイダーとして使用してサインインを行うことができます。このオプションの場合、ゲームに Facebook ログインを設定する必要があります。このサインインワークフローでは、プレイヤーは Facebook にリダイレクトされ、そこで認証情報を入力します。Facebook が認証を処理し、その結果をゲームに伝えます。

ゲームに両方のサインイン方法を含めることを選択し、プレイヤーが両方のタイプの認証情報を入力した場合は、ID と認証によって両方のログインを同じプレイヤー ID に関連付けることができます。

このゲーム機能により、認証情報の漏洩チェックやアカウント乗っ取り防止などのセキュリティ機能が組み込まれます。

AWS GameKit での ID と認証では、認証チャレンジ、カスタム検証 E メール、ユーザーディレクトリ管理などの機能の特別サポートは提供しません。ただし、AWS ツールを使用して AWS リソースを手動でカスタマイズすることで、これらの機能やその他の機能を追加することができます。AWS バックエンドソリューションの詳細については、「ID と認証のソリューションアーキテクチャ」を参照してください。

ID と認証のワークフロー

プレイヤー登録のために、AWS GameKit は次のシナリオをサポートしています。

- プレイヤーは E メール/パスワードでサインインします。このシナリオでは、AWS GameKit が自動的に検証ワークフローをトリガーし、検証コードが記載された E メールをその E メールアドレスに送信します。登録を完了するには、プレイヤーが検証コードを取得してゲーム UI に入力する必要があります。登録に成功すると、新しい一意のプレイヤー ID が作成され、そのサインイン情報が暗号化されて保存されます。
- プレイヤーは Facebook アカウントでサインインします。このシナリオでは、AWS GameKit は、ユーザーに Facebook のウェブページに移動してログインするように求めるワークフローをトリガーします。ログインに成功し、プレイヤーが初めてゲームにサインインしたことをゲームの ID と認証のバックエンドが検出すると、新しい一意のプレイヤー ID が作成されます。

プレイヤーの登録とサインインのワークフローは次のとおりです。

1. ゲームクライアントで、プレイヤーが E メールアドレスを入力するか、[...でサインイン] ボタンを選択して 使用する外部 ID プロバイダーを選択すると、新しいゲームアカウントを作成するオプションが表示されます。
2. プレイヤーが E メールアドレスとパスワードを入力した場合:
    a. AWS GameKit は、セッションベースの確認コードを含む検証 E メールを、指定された E メールアドレスに送信します。
    b. ゲームクライアントで、プレイヤーは確認コードを入力し、ゲームに対して登録の確認を求めます。
    c. 確認が成功すると、AWS GameKit は新しいプレイヤーレコードを作成し、ID トークンを返します。このトークンを使用して、プレイヤーのゲームクライアントとゲームの ID と認証のバックエンド間の通信が承認されます。
3. プレイヤーが Facebook でサインインすることを選択した場合:
    a. [Facebook でサインイン] ボタンをクリックすると、ゲームの ID と認証のバックエンドから Facebook のフェデレーションログイン URL をリクエストするようにゲームがトリガーされます。URL には Facebook のゲームのアカウント ID が含まれます。
    b. ゲームクライアントは Facebook のログイン URL をブラウザで開き、プレイヤーは Facebook アカウントにログインします。
    c. Facebook はログインステータスを返します。成功すると、AWS GameKit は新しいプレイヤーレコードを作成し、ID トークンを返します。このトークンを使用して、プレイヤーのゲームクライアントとゲームの ID と認証のバックエンド間の通信が承認されます。
4. ゲームクライアントで、プレイヤーは既存のゲームアカウントを使用してゲームにサインインします。
5. ゲームでプレイヤーのサインインが試行されます。サインインの試行が有効な場合、AWS GameKit はセッションベースのトークンで応答します。

トピック

- ID と認証のソリューションアーキテクチャ
- ID と認証の設定オプション
- ID と認証の呼び出し可能なアクション

ID と認証のソリューションアーキテクチャ

このトピックでは、AWS GameKit の ID と認証の機能をサポートするクラウドベースのバックエンドサービスを提供する AWS ソリューションについて詳しく説明します。AWS GameKit を使用してアチーブメント機能をゲームに組み込んで保守を行う前に、この情報を習得しておく必要はありませんが、ゲームのバックエンドにデプロイされる AWS サービスとリソースをより深く理解するのに役に立ちます。バックエンドのコンポーネントはいつでも AWS で直接確認でき、モニタリングや分析などの他の AWS サービスと併用することもできます。ゲームのバックエンドサービスを AWS GameKit で利用できるサービスを超えてカスタマイズまたは拡張したい場合は、ソリューションの各コンポーネントの役割を理解する必要があります。

ID と認証のバックエンドアーキテクチャでは、ゲームクライアントからの API リクエストを認証するための次の呼び出しフローが実装されています。

1. ゲームクライアントは ID と認証の API オペレーションを呼び出します。このオペレーションにより、AWS GameKit は Amazon API Gateway エンドポイントにリクエストを送信するよう求められます。
2. Amazon Cognito はゲームクライアントのアクセストークンを検証します (存在する場合)。トークンがないか無効な場合、クライアントはサインインページにリダイレクトされます。
3. ゲームクライアントはプレイヤーのサインイン認証情報 (ユーザー名/パスワードまたは Facebook サインイン) で認証し、Amazon Cognito ID トークンを受け取ります。
4. ゲームクライアントは Amazon Cognito ID トークンを使用して API リクエストを繰り返します。
5. ゲームクライアントのリクエストは、現在検証されている Amazon Cognito ID トークンと共に、関連する Lambda 関数にパススルーされます。

ID と認証のサービス

で説明されているように、すべての AWS GameKit AWS ソリューションはコアサービスセットに依存しています[コアサービス]。

ID と認証のアクティビティの管理には次のサービスが使用されます。

Amazon Cognito

AWS GameKit は、プレイヤーの ID と認証の認証情報を管理するための Amazon Cognito ユーザープールを作成します。ユーザープールは、E メール/パスワード、またはさまざまな外部 ID プロバイダー (Facebook など) を受け入れるように設定できます。Amazon Cognito は、サインイン検証およびパスワード回復ワークフローを管理します。

AWS Lambda

AWS GameKit は、Lambda 関数を使用して、プレイヤーが正常に登録されたときに Amazon DynamoDB テーブルに ID 情報を保存するプロセスを管理します。

Amazon DynamoDB

AWS GameKit は、プレイヤーの ID 情報を追跡するための DynamoDB テーブルを作成します。例えば、プレイヤーのユーザー名は、E メールアドレスと、外部 ID プロバイダーのアカウントの両方にリンクできます。

ID と認証のデータ暗号化

プレイヤーデータは転送中および保管時に暗号化されます。

転送中、AWS GameKit は AWS でのゲームのフロントエンドおよびバックエンドコンポーネント間の通信に Transport Layer Security (TLS) 1.2 以降を使用します。AWS GameKit のすべてのゲーム機能は、Amazon API Gateway サービスを使用して API コールを受け入れ、処理します。「API Gateway 開発者ガイド」の「Data protection in transit」を参照してください。

保管中のプレイヤー ID データは、ID と認証のゲーム機能が使用する AWS サービスによって暗号化されます。これらのサービスは業界標準に準拠しています。これらのサービスが保管中のデータ暗号化をどのように処理するかについては、以下を参照してください。

- 「Amazon Cognito デベロッパーガイド」の「Amazon Cognito でのデータ保護」
- 「Amazon DynamoDB 開発者ガイド」、「保管時の DynamoDB 暗号化」

ID と認証の設定オプション

ゲームの ID と認証の機能を設定する際、次の特性をカスタマイズできます。これらのカスタマイズは、この機能のバックエンドコンポーネントの構築方法に影響します。

- プレイヤーが E メールとパスワードでサインオンすることを有効または無効にします。
- プレイヤーが Facebook アカウントでサインオンすることを有効または無効にします。このオプションを有効にするには、Facebook で開発者のアカウントを設定し、Facebook アプリ ID を取得する必要があります。

ID と認証の呼び出し可能なアクション

AWS GameKit API は、ID と認証のゲーム機能に対して次のアクションを提供します。ID と認証のバックエンドの AWS リソースをデプロイすると、ゲームフロントエンドは、これらの呼び出しを使用してバックエンドと通信できます。

- `Register()` プレイヤーのユーザー名、E メールアドレス、パスワードを取り込んで新しいプレイヤー ID を作成します。登録後、プレイヤーはこの情報を使用してゲームにログインできます。
- `ConfirmRegistration` 登録確認コードを検証します。E メールアドレスを使用して新しいプレイヤー ID を登録すると、プレイヤーは、確認コードが記載された検証 E メールを受信します。この確認コードをゲーム UI に入力する必要があります。このアクションは入力されたコードを検証します。
- `ResendConfirmationCode()` プレイヤーは E メールアドレスを検証するための新しい確認コードをリクエストできます。確認コードの有効期間は短くなっています。
- `Login()` 登録したユーザー名とパスワードでプレイヤーをゲームにサインインさせます。
- `GetFederatedLoginUrl()` プレイヤーが自分のアカウントにログインできる、Facebook などの外部 ID プロバイダーの URL を取得します。このアクションがプレイヤーに対して初めて呼び出された場合、AWS GameKit はそのプレイヤーを登録し、新しいプレイヤー ID を生成します。
- `GetUser()` 現在ログインしているプレイヤーのユーザー情報を取得します。これにはプレイヤーの一意の ID が含まれます。
- `Logout()` プレイヤーをゲームからサインアウトさせます。
- `ForgotPassword()` E メール/パスワードで登録したプレイヤーは、このアクションによりパスワード回復ワークフローがトリガーされます。
- `ConfirmForgotPassword()` プレイヤーによるパスワードの変更を許可する確認コードを検証します。これはパスワード回復ワークフローの一部です。

ID と認証の予想コスト

次の表は、このゲーム機能のコストが発生する可能性のある ID と認証のための AWS サービスのセットの概要を示しています。この条件は、各サービスの無料利用枠の特典の範囲内で許容される上限を示しています。無料利用枠の特典の中には、期間限定で利用できるものもあれば、使用制限までは常に無料のものもあることに注意してください。

サービス	条件	無料利用枠のコスト/月
Amazon S3	Less than 320 deployments per day	0
Amazon CloudFormation	Less than 1,000 deployments per month	0
Amazon Cognito	Less than 50,000 monthly active users (MAU) per month	0
Amazon API Gateway	Less than 1M calls per month (combined registration, login, confirmation, password reset, etc.)	0
AWS Lambda (x86 architecture)	Less than 1M requests per month and 400,000 GB-seconds of compute time per month (combined registration, login, confirmation, password reset, etc.)	0
Amazon DynamoDB	Under 25 requests per second (registrations, confirmation, logins, etc.) $0.00065 WCU per hour and $0.00013 RCU per hour after estimated: 5.81/ month	0
AWS Key Management Service (KMS)	1 日あたり 222 件の Facebook ログイン (1 か月あたり 20,000 件未満のリクエスト)	0

プロジェクトに ID と認証を追加する

概要

完全なクラウドベースの ID と認証のシステムを構築し、Unreal Engine プロジェクトに統合する方法について説明します。このトピックでは、開発者がバックエンドサービスを構築し、AWS GameKit API を使用してフロントエンドコードを AWS のバックエンドに接続する方法について説明します。

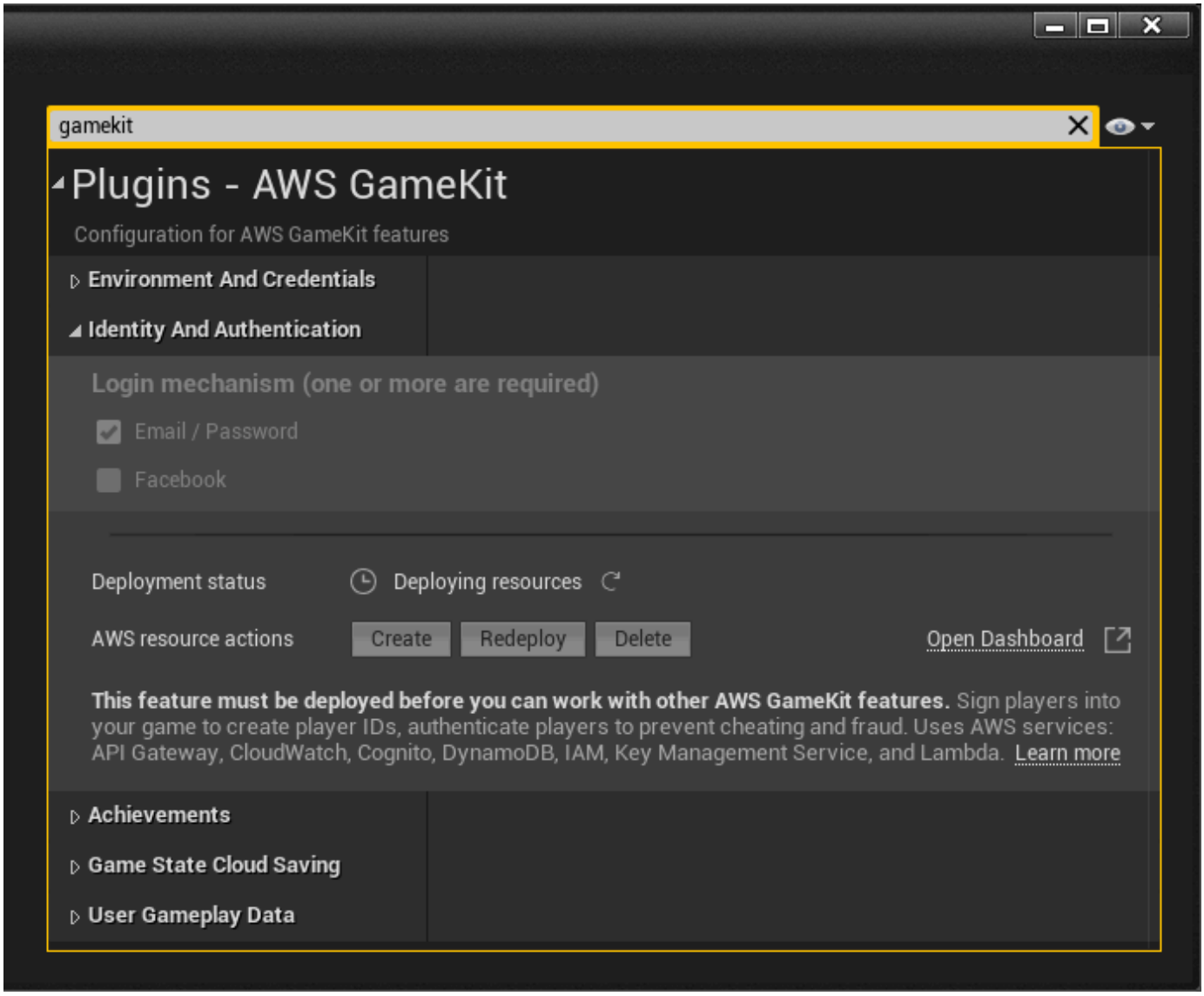
ゲームの ID と認証の機能を構築する準備ができたら、以下の基本ステップを実行します。プロジェクトに AWS GameKit をまだインストールしていない場合は、「Unreal Engine での AWS GameKit プラグインのインストール」および「ゲーム用の AWS GameKit プラグインのセットアップ」を参照してください。

ステップ 1。ID と認証のゲーム機能を設定する。

1. Unreal Editor ツールバーで、[編集] > [プロジェクト設定] を開き、[AwsGameKit] プラグインセクションに移動します。作業する環境を選択し、必要に応じて有効な AWS 認証情報を入力します。詳細については、「ゲーム用の AWS GameKit プラグインのセットアップ」を参照してください。
2. [ID と認証] セクションを展開します。次の設定オプションを指定します。

   - [ログインメカニズム]。プレイヤーに提供するログインメカニズムを選択します。デフォルトは [E メール/パスワード] です。少なくとも 1 つのオプションを選択する必要があります。Facebook でのプレイヤーログインをサポートする場合は、Facebook アプリ ID とゲームのシークレットキーを入力します。Facebook アプリ ID の取得方法の詳細については、「Facebook 開発者ポータル」を参照してください。

   機能の設定に加えた変更はすべて、現在のセッション中にローカルにキャッシュされます。



ステップ 2。ID と認証のバックエンドの AWS リソースをデプロイする。

1. [ID と認証] の AWS GameKit 設定を開いたまま、デプロイのコントロールまでスクロールします。AWS リソースアクション [作成] を選択します。このアクションにより、AWS GameKit は、このゲーム機能のバックエンドサービスを実行するための完全な AWS ソリューションをデプロイするように指示されます。デプロイ時に、AWS GameKit は AWS に接続し、アクティブな環境用に管理されている設定テンプレートを使用して、そのソリューションのすべての AWS リソースを作成します。
2. ID と認証のリソースのデプロイは、通常 5 分で完了します。この間、機能のデプロイステータスは [リソースのデプロイ中] と表示されます。デプロイステータスの進行状況を追跡できます。

- Unreal Editor で出力ログウィンドウを開き、デプロイ全体のステータスメッセージ、イベント、エラーを監視します。
- AWS Management Console で AWS CloudFormation サービスを開きます。[スタック] ビューでは、ゲームプロジェクトの Identity スタックがデプロイされている様子を見ることができます。

デプロイが完了すると、機能のデプロイステータスは [デプロイ済み] と表示されます。これで ID と認証のゲームバックエンドの準備が整いました。AWS GameKit API を使用して呼び出すことができます。

> **Note**
>
> 注: この時点から、このゲーム機能のコストが発生し始める可能性があります。まだ AWS 無料利用枠に入っている場合は、無料利用枠の制限を超えた場合にのみ費用が発生します。

ステップ 3。プレイヤー ID ワークフローをゲームに追加する。

UI 要素を作成し、次のワークフローのコードを追加します。説明についてはプラグインの ID の例を参照し、C++ コードサンプルを使用して API コールを試してみます。

ID と認証のワークフローには以下が含まれます。

- E メールで新規プレイヤーを登録する
  - `Register()`
  - `ConfirmRegistration()`
  - `ResendConfirmationCode()`
- 既存の E メールアカウントでプレイヤーにサインインする
  - `Login()`
  - `GetUser()` および `GetResponseBody()`
  - `Logout()`
- Facebook で新規または既存のプレイヤーにサインインする
  - `GetFederatedLoginUrl()`

  - `PollAndRetrieveFederatedTokensAsync()`、または `PollAndRetrieveFederatedTokensBlueprintAsync()`
  - `GetFederatedAccessToken()`
  - `Logout()`
- 既存のアカウントのパスワードをリセットする
  - `ForgotPassword()`
  - `ConfirmForgotPassword()`

ID と認証のサンプルを使用する

AWS GameKit プラグインには、ID と認証のゲーム機能のサンプルアセットが含まれています。

Unreal Engine

Unreal Engine 用 AWS GameKit プラグインには、C++ コード、ブループリント、UI コンポーネントを含むサンプルが用意されています。サンプルファイルには Unreal Editor のコンテンツブラウザからアクセスできます。

- C++ サンプルの操作
- ブループリントと UI サンプルの操作

C++ サンプルの操作

Unreal Editor のコンテンツブラウザで、以下の場所にあるサンプルアセットを探します。

```
AwsGameKit C++ Classes > AwsGameKitEditor > Public > Identity >
 AwsGameKitIdentityExamples
```

このアセットは `.cpp` ファイルです。このファイルは、`...\AwsGameKit\Source\AwsGameKitEditor\Private\Identity\AwsGameKitIdentityExamples.cpp` にある AWS GameKit プラグインファイルにもあります。

このサンプルファイルには、ID と認証 API の各アクションを呼び出す方法を示すサンプルコードが含まれています。このファイルには、実行可能なコードの基本セットと詳細なコメントが含まれています。

このサンプルは 2 つの方法で操作できます。1 つは IDE でコードを表示する方法、もう 1 つは Unreal Editor で API コールを試してみることです。

サンプルコードを表示または編集するには:

- AwsGameKitIdentityExamples アセットをダブルクリックして IDE でファイルを開きます。このファイルにアクセスする前に AWS 認証情報を入力したり、AWS リソースをデプロイしたりする必要はありませんが、デプロイされたリソースがないと API コールは動作しません。

  サンプルコードには、AWS リソースがデプロイされていることを確認する標準チェックが含まれています。

  最初のステップとして、サンプルコードは `UAwsGameKitIdentityCallableWrapper` のインスタンスを作成して初期化することに注意してください。これは API コールを行う前に行う必要があります。

Unreal Editor でサンプルを試すには:

ID と認証用の AWS リソースをデプロイし、AWS GameKit プラグインのプロジェクト設定で有効な AWS 認証情報を送信する必要があります (「ゲーム用の AWS GameKit プラグインのセットアップ」を参照)。

1. AwsGameKitIdentityExamples アセットをビューポートのレベルにドラッグします。アセットはどのレベルに追加してもかまいません。これはアセットの設定を操作できるようにする単なるメカニズムです。
2. Editor の [詳細] ペインでは、ID と認証の API コールはすべて、API のリクエスト値とレスポンス値を使用して実行できます。
3. API コールを行うには、入力値を指定して [呼び出し] ボタンをクリックします。レスポンスは「戻り値」フィールドに表示されます。
4. 次の呼び出しシーケンスを実行して、標準の ID シナリオをシミュレートしてみます。

   - 新しいプレイヤーを E メールで登録する: プレイヤーを登録する、E メールを確認する
   - 既存のアカウントでプレイヤーにサインインする: ログインする、ユーザーを取得する、ログアウトする
   - Facebook で新規または既存のプレイヤーにサインインする: Facebook のログインを開く、ユーザーを取得する、ログアウトする

- 既存アカウントのパスワードを回復する: パスワードを忘れた場合、パスワードを忘れた場合の確認

ブループリントと UI サンプルの操作

Unreal Editor のコンテンツブラウザで、以下の場所にあるサンプルアセットを探します。

```
AwsGameKit Content > Identity >
```

ここでは、以下の 2 つのサンプルアセットを示します。

- BP_AwsGameKitIdentityExamples

  このアセットは、サインイン機能をゲームコードに追加する方法を示す基本的なブループリントです。

  アクション:
  - ブループリントを開くには、アセットをダブルクリックします。
- BP_AwsGameKitIdentityExamplesUI

  このアセットには、一般的なサインインワークフローシナリオの詳細なブループリントとサンプル UI オブジェクトが含まれています。

  アクション:
  - ブループリントを開くには、アセットをダブルクリックします。

  サンプルを実行するには、[再生] をクリックします。

AWS GameKit 機能: ユーザーゲームプレイデータ

AWS GameKit のゲーム機能であるユーザーゲームプレイデータは、プレイヤーのゲーム関連データをクラウドに保存するためのツールを提供します。この機能により、ゲームプレイデータはゲームセッションをまたがって保持され、複数のゲームクライアントやデバイス間で同期できます。

このゲーム機能は、ゲームプレイ中にその場でデータ同期を処理するように設計されています。プレイヤーのプロフィールやアカウント情報など、プレイヤーに関する他のデータ用には設計されていません。ユーザーゲームプレイデータには、プレイヤーのスコア、インベントリ、またはゲームに必要なその他のプレイヤー関連データなどの情報が含まれる場合があります。プロフィールやアカウント情報など、ゲーム以外のプレイヤーデータは含まれません。

このゲーム機能の潜在的な用途には以下が含まれます。

- ゲームのインベントリ
- ゲームエコノミー/通貨
- プレイヤーの統計
- プレイヤーのゲーム選択 (キャラクターやゲームの進行に影響するゲーム内の意思決定など)
- レベルアクティビティ (訪問した場所など)

トピック

ユーザーゲームプレイデータの仕組み

概要

このトピックでは、ゲーム機能の概要を説明し、その機能をゲームに統合する方法の一般的なシナリオについて紹介します。

このゲーム機能には、非常に柔軟なデータスキーマが用意されています。データをどのように整理するかは自分で決めます。簡単に言うと、ゲームプレイデータはバンドルアイテムと呼ばれるキーと値のペアのセットとして保存されます。バンドルとは、関連するバンドルアイテムをグループ化するために使用できるコンストラクトです。

基本的なスキーマ構造は次のようになります。

- バンドル名。バンドルは、関連するバンドルアイテムのコレクションを作成できるコンストラクトです。例えば、プレイヤーの武器インベントリのデータをバンドルにまとめることができます。
  - バンドルアイテム。データ識別子 (キー) と保存された値からなる単独のデータ。どのデータ項目を保存する必要があるかを決めるのはユーザーです。例えば、武器のインベントリでは、少なくとも (1) その武器が入手可能かどうか、(2) 弾薬の量、(3) 改造、(4) ショートカットキーが割り当てられているかどうかを追跡する必要があるでしょう。これらの情報の一つひとつがバンドルアイテムとして保存されます。
    - バンドルアイテムキー。一意のアイテム名または ID。
    - バンドルアイテム値。値。

例:

```
"playerid_bundle":
 "d99a7965f7b86d299c272b183c3de54c4180c63d3caf8de2edd37a6eec0ea200_BEST_TIME",
"bundle_item_key": "TRACK_1",
"bundle_item_value": "3:36.113"
```

各プレイヤーが保管できるバンドルの数に制限はなく、バンドルに含まれるアイテムの数にも制限はありません。データストレージの使用量に応じた支払いとなるため、ユーザーの判断に委ねられます。

このゲーム機能は、ゲームデータの頻繁な読み取りまたは書き込みをサポートするように設計されています。開発者はデータ項目をリアルタイムで簡単に設定および取得できます。

このゲーム機能には、接続が失われた場合でもデータの整合性を維持するための保護機能が追加されています。配信できないすべてのリクエストがメッセージキューに入れられます。接続できない場合、追加、更新、削除のリクエストはすべて自動的にメッセージキューに入れられます。接続が回復すると、リクエストはキューに入れられた順序 (古い順) で処理されます。リクエストがキューに入っている間に同じバンドルアイテムに対して複数の更新リクエストが発生した場合、新しいリクエストが自動的に古いリクエストに置き換えられるため、バンドルアイテムに対して処理される書き込みリクエストは 1 つだけです。GET リクエストはキューに入れられないことに注意してください。

ユーザーゲームプレイデータのソリューションアーキテクチャ

このトピックでは、AWS GameKit のユーザーゲームプレイデータ機能をサポートするため、クラウドベースのバックエンドサービスを提供する AWS ソリューションについて詳しく説明します。AWS GameKit を使用してアチーブメント機能をゲームに組み込んで保守を行う前に、この情報を習得しておく必要はありませんが、ゲームのバックエンドにデプロイされる AWS サービスとリソースをより深く理解するのに役に立ちます。バックエンドのコンポーネントはいつでも AWS で直接確認でき、モニタリングや分析などの他の AWS サービスと併用することもできます。ゲームのバックエンドサービスを AWS GameKit で利用できるサービスを超えてカスタマイズまたは拡張したい場合は、ソリューションの各コンポーネントの役割を理解する必要があります。

ワークフローのシーケンスは次のとおりです。

1. ゲームクライアントはユーザーゲームプレイデータ API オペレーションを呼び出し、これにより AWS GameKit は API Gateway エンドポイントにリクエストを送信するよう促されます。Amazon Cognito は、「 ID と認証のソリューションアーキテクチャ」で説明されているように、ゲームクライアントのアクセストークンを検証します。リクエストにプレイヤーのアチーブメントデータが含まれる場合、Amazon Cognito オーソライザーは、アクセストークンがユーザープールで定義されているプレイヤーに有効であることを確認します。
2. 認証が成功すると、ゲームクライアントのリクエストは関連する Lambda 関数に渡されます。
3. Lambda 関数は DynamoDB とやり取りして、リクエストに応じてバンドルまたはバンドルアイテム上のデータを保存または取得します。DynamoDB テーブルは 2 つあり、1 つはバンドルを追跡するためのもので、もう 1 つはバンドルアイテムを追跡するためのものです。

ユーザーゲームプレイデータのサービス

で説明されているように、すべての AWS GameKit AWS ソリューションはコアサービスセットに依存しています[コアサービス]。

これらのサービスはユーザーゲームプレイデータのアクティビティの管理に使用されます。

AWS Lambda

AWS GameKit は Lambda 関数を使用してゲームプレイデータの保存と取得を管理します。

Amazon DynamoDB

AWS GameKit は 2 種類の DynamoDB テーブルを作成します。1 つはバンドル名を保存するためのもので、もう 1 つはバンドルアイテムと値をプレイヤーとバンドル名ごとに格納するためのものです。

ユーザーゲームプレイデータの暗号化

プレイヤーデータは転送中および保管時に暗号化されます。

転送中、AWS GameKit は AWS でのゲームのフロントエンドおよびバックエンドコンポーネント間の通信に Transport Layer Security (TLS) 1.2 以降を使用します。AWS GameKit のすべてのゲーム機能は、Amazon API Gateway サービスを使用して API コールを受け入れ、処理します。「API Gateway 開発者ガイド」の「Data protection in transit」を参照してください。

保管中のデータは、ユーザーゲームプレイデータのゲーム機能が使用する AWS サービスによって暗号化されます。これらのサービスは業界標準に準拠しています。これらのサービスが保管中のデータ暗号化をどのように処理するかについては、以下を参照してください。

- 「Amazon DynamoDB 開発者ガイド」、「保管時の DynamoDB 暗号化」

ユーザーゲームプレイデータの呼び出し可能なアクション

AWS GameKit API は、ユーザーゲームプレイデータのゲーム機能用に以下のアクションを提供します。ユーザーゲームプレイデータのバックエンド用に AWS リソースをデプロイすると、ゲームのフロントエンドはこれらの呼び出しを使用してバックエンドと通信できます。

- `AddUserGameplayData()`: この呼び出しを使用してバンドルを作成または更新します。バンドル名のみで呼び出すことも、1 つ以上のバンドルアイテムを含めることもできます。バンドルアイテムキーがクラウドに既に存在する場合、値は更新されます。
- `GetAllUserGameplayData()`: プレイヤー ID に保存されているすべてのデータバンドルを取得します。バンドルアイテムデータは、バンドル名別に整理されたデータと共に返されます。
- `GetUserGameplayDataBundle()`: 指定したバンドル内のプレイヤーのバンドルアイテムを取得します。データは、キーと値のペアのマップとして返されます。
- `GetUserGameplayDataBundleItem()`: バンドル名とバンドルアイテムキーが指定されている場合、プレイヤーの 1 つのバンドルアイテムを取得します。データは、単一のキーと値のペアとして返されます。
- `UpdateUserGameplayDataBundleItem()`: バンドル内の単一のバンドルアイテムを追加または更新します。

- `DeleteUserGameplayDataBundleItem()`: 指定したバンドル内の単一のバンドルアイテムを削除します。
- `DeleteUserGameplayDataBundle()`: 指定したバンドル (それに含まれるすべてのバンドルアイテムを含む) を削除します。
- `DeleteAllUserGameplayData()`: プレイヤーのすべてのバンドルとバンドルアイテムを削除します。

ユーザーゲームプレイデータをゲームに追加

Unreal Engine プロジェクトにユーザーゲームプレイデータのゲーム機能を追加する基本的な手順を以下に説明します。ゲームエンジン用の AWS GameKit をまだ設定していない場合は、「Unreal Engine での AWS GameKit プラグインのインストール」を参照してください。

ユーザーゲームプレイデータ機能の構築

ステップ 1。ユーザーゲームプレイデータのバックエンドに AWS ソリューションをデプロイします。

1. Unreal Editor ツールバーで、[編集] > [プロジェクト設定] を開き、[AwsGameKit] プラグインセクションに移動します。作業する環境を選択し、必要に応じて有効な AWS 認証情報を入力します。詳細については、「ゲーム用の AWS GameKit プラグインのセットアップ」を参照してください。
2. [ユーザーゲームプレイデータ] セクションを開きます。このゲーム機能には特別な設定はありません。
3. AWS リソースアクション [作成] を選択します。このアクションにより、AWS GameKit は、このゲーム機能のバックエンドサービスを実行するための完全な AWS ソリューションをデプロイするよう促されます。デプロイ時に、AWS GameKit は最初にカスタム設定を含む AWS ソリューションのテンプレートを生成し、次に AWS に接続してテンプレートで定義されているソリューションの AWS リソースを作成します。AWS リソースは、アクティブな環境用に選択された AWS リージョンにデプロイされます。
4. ユーザーゲームプレイデータのバックエンド用にリソースをデプロイするには、通常 5 分かかります。デプロイステータスの進行状況を次のように追跡できます。
    - [機能] タブを下にスクロールして [ログ] ウィンドウを表示します。ログには、デプロイ全体のステータスメッセージ、イベント、エラーが表示されます。

- AWS マネジメントコンソールで AWS CloudFormation サービスを開きます。[スタック] ビューでは、ゲームプロジェクトの UserGameplayData スタックがデプロイされている様子を見ることができます。
-

デプロイが完了すると、ユーザーゲームプレイデータのゲーム機能のゲームバックエンドが完成します。AWS GameKit API を使用して呼び出すことができます。

> **Note**
>
> 注: AWS 無料利用枠の対象かどうかによっては、この時点からコストが発生し始める可能性があります。

ステップ 3。ゲームプレイデータ機能をゲームに追加します。

1. ユーザーゲームプレイデータ API を呼び出す前に、`GdkUserGameplayDataCreate` を呼び出して `Initialize()` を呼び出します。
2. ゲームプレイデータをクラウドバックエンドに保存し、ゲームクライアントとの同期を維持するコードを追加します。その一環として、バンドルを使用して必要に応じてデータ項目をコレクションにグループ化する、ゲーム用のゲームプレイデータスキーマを開発します。バンドルの作成、バンドルアイテムの追加、アイテム値の更新を行う場所とタイミングを決定します。
3. 次のようなワークフローを追加します。
   - データ項目のコレクションを含むバンドルを作成する
     - `AddUserGameplayData()`
   - バンドルアイテムを新しい値に更新する
     - `UpdateUserGameplayDataBundleItem()`
   - バンドルされたデータアイテムのコレクションを取得する
     - `GetUserGameplayDataBundle()`
   - 1 つのバンドルアイテム値を取得する
     - `GetUserGameplayDataBundleItem()`
   - バンドルを削除する
     - `DeleteUserGameplayDataBundle()`

統合のヒント

ゲームプレイデータ機能をゲームに追加する際には、以下の問題を考慮してください。

- 追加、更新、削除の API コールは非同期的に行うことを検討します。GET 呼び出しは同期的に行うことができます。
- ゲームクライアントでは、改ざんを防ぐためにデータ検証とサニティチェックを追加できます。

ユーザーゲームプレイデータのサンプルの操作

AWS GameKit プラグインには、ユーザーゲームプレイデータのゲーム機能用のサンプルアセットが含まれています。

Unreal Engine

Unreal Engine 用 AWS GameKit プラグインには、C++ コード、ブループリント、UI コンポーネントを含むサンプルが用意されています。サンプルファイルには Unreal Editor のコンテンツブラウザからアクセスできます。

- C++ サンプルの操作
- ブループリントと UI サンプルの操作

C++ サンプルの操作

Unreal Editor のコンテンツブラウザで、以下の場所にあるサンプルアセットを探します。

```
AwsGameKit C++ Classes > AwsGameKitEditor > Public > UserGamePlayData >
 AwsGameKitUserGamePlayDataExamples
```

このアセットは .cpp ファイルです。このファイルは、`...\AwsGameKit\Source\AwsGameKitEditor\Private\UserGamePlayData\AwsGameKitUserGamePlayDataExamples.cpp` にある AWS GameKit プラグインファイルにもあります。

このサンプルファイルには、ユーザーゲームプレイデータ API アクションをそれぞれ呼び出す方法を示すサンプルコードが含まれています。このファイルには、実行可能なコードの基本セットと詳細なコメントが含まれています。

このサンプルは 2 つの方法で操作できます。1 つは IDE でコードを表示する方法、もう 1 つは Unreal Editor で API コールを試してみることです。

サンプルコードを表示または編集するには:

- AwsGameKitUserGamePlayDataExamples アセットをダブルクリックして IDE でファイルを開きます。このファイルにアクセスする前に AWS 認証情報を入力したり、AWS リソースをデプロイしたりする必要はありませんが、デプロイされたリソースがないと API コールは動作しません。

  サンプルコードには、AWS リソースがデプロイされていることを確認する標準チェックが含まれています。

  最初のステップとして、サンプルコードは `UAwsGameKitUserGamePlayDataCallableWrapper` のインスタンスを作成して初期化することに注意してください。これは API コールを行う前に行う必要があります。

Unreal Editor でサンプルを試すには:

ID と認証用の AWS リソースをデプロイし、AWS GameKit プラグインのプロジェクト設定で有効な AWS 認証情報を送信する必要があります (「ゲーム用の AWS GameKit プラグインのセットアップ」を参照)。

1. AwsGameKitUserGamePlayDataExamples アセットをビューポートのレベルにドラッグします。アセットはどのレベルに追加してもかまいません。これはアセットの設定を操作できるようにする単なるメカニズムです。
2. Editor の [詳細] ペインで、ユーザーゲームプレイデータ API コールを API リクエスト値とレスポンス値と共に使用できます。この UI を使用して行われた呼び出しは、AWS にデプロイされたゲームのバックエンドに接続されます。
3. API コールを行うには、入力値を指定して [呼び出し] ボタンをクリックします。レスポンスは「戻り値」フィールドに表示されます。
4. 以下の呼び出しシーケンスを実行して、標準のゲームプレイデータシナリオをシミュレートしてみてください。

   - 1 つ以上のバンドルアイテムを含むバンドルの作成: ユーザーゲームプレイデータを追加
   - 既存のバンドルアイテムの値を更新: ユーザーゲームプレイデータのバンドルアイテムを更新
   - バンドル内のすべてのアイテム値を取得: ユーザーゲームプレイデータのバンドルを取得

- 単独のバンドルアイテム値を取得: ユーザーゲームプレイデータのバンドルアイテムを取得
- バンドルを削除: ユーザーゲームプレイデータのバンドルを削除

ブループリントと UI サンプルの操作

Unreal Editor のコンテンツブラウザで、以下の場所にあるサンプルアセットを探します。

```
AwsGameKit Content > UserGameplayData >
```

ここでは、以下の 2 つのサンプルアセットを示します。

- BP_AwsGameKitUserGameplayDataExamples

  このアセットは、ゲームプレイデータの保存と取得機能をゲームコードに追加する方法を示す基本的なブループリントです。

  アクション:
  - ブループリントを開くには、アセットをダブルクリックします。
- BP_AwsGameKitUserGameplayDataExamplesUI

  このアセットには、一般的なワークフローシナリオの詳細なブループリントとサンプル UI オブジェクトが含まれています。

  アクション:
  - ブループリントを開くには、アセットをダブルクリックします。

  サンプルを実行するには、[再生] をクリックします。

AWS GameKit 機能: ゲーム状態のクラウド保存

AWS GameKit のゲーム状態のクラウド保存機能を使用すると、プレイヤーはゲームの状態を保存してクラウドに保管し、ローカルのゲームクライアントとの同期を維持できます。ゲームの保存は、特にストーリーが非常に長いゲームでは、プレイヤーにとって非常に価値の高いゲーム機能です。クラウド保存と同期は、その他にもプレイヤーに次のようなメリットをもたらします。

- 複数のゲームプラットフォームをサポートするゲームでは、プレイヤーは複数のデバイスでクロスプレイできる真のポータビリティを実現できます。
- ローカルデバイスで障害が発生しても、プレイヤーは最小限の損失でゲームの進行状況を回復できます。
- アンインストールされたゲームであっても、プレイヤーは過去にプレイしたゲームのセーブデータに引き続きアクセスできます。

このゲーム機能を使うと、プレイヤーがセーブポイントを選択できる明示的なセーブに加えて、自動セーブシステムも実装できます。

トピック

ゲーム状態のクラウド保存の仕組み

このゲーム機能は主に 2 つのタスクを処理します。1 つはプレイヤーのゲームセーブファイルをクラウドに保存すること、もう 1 つは各ゲームセーブファイルのローカルバージョンとクラウドバージョンを同期することです。同期することで、プレイヤーが複数のデバイスで同じゲームをプレイしていたとしても、常に最新バージョンのセーブデータでプレイできます。

ゲームセーブファイルをクラウドに保存する

プレイヤーのゲームの状態はファイルとして保存され、クラウド内のファイルストレージにアップロードできます。この機能では、ゲームセーブファイルはスロットとして管理されます。各スロットには、セーブ名、最終更新日スタンプなどのメタデータと、オプションで JSON 形式の開発者が定義したメタデータの文字列がアタッチされています。

ゲームセーブファイルの同期

同期により、プレイヤーがゲームをプレイするために使用しているデバイスの数を問わず、ゲームセーブファイルの更新が正しい順序で行われるため、プレイヤーの体験がシームレスになります。

この機能は、次に示す 5 つの潜在的な状態と、それらを同期させるために実行すべきアクションを追跡します。

- ローカルのみ: ゲームセーブファイルはローカルデバイスにのみ存在します。アクション: ローカルのゲームセーブファイルをクラウドにアップロードします。
- クラウドのみ: ゲームセーブファイルはクラウドにのみ存在します。この状態は、プレイヤーが新しいデバイスでゲームを再開しているときに発生することがあります。アクション: クラウドのゲームセーブファイルをローカルデバイスにダウンロードします。
- ローカル/クラウド同期: ゲームセーブファイルはローカルとクラウドの両方に存在し、最終更新日時のタイムスタンプは一致しています。対処は必要ありません。
- ローカルが最新: ゲームセーブファイルはローカルとクラウドの両方に存在しますが、ローカルバージョンの更新日の方が最新です。この状態は、プレイヤーがローカルデバイスでゲームをプレイしていて、ゲームの進行状況を保存したい場合に発生します。アクション: 最新のローカルバージョンのゲームセーブファイルをスロットにアップロードします。
- クラウドが最新: ゲームセーブファイルはローカルとクラウドの両方に存在しますが、クラウドバージョンの更新日の方が最新です。この状態は、プレイヤーが最近別のデバイスでゲームをプレイし、ゲームの進行状況をそこに保存し、現在のデバイスでゲームを再開したい場合に発生する可能性があります。アクション: 最新のクラウドバージョンのゲームセーブファイルをスロットにダウンロードします。

ゲーム状態のクラウド保存のワークフロー

ゲーム状態のクラウド保存システムの仕組みを、プレイヤーの視点から大まかに説明します。

プレイヤーとしてゲームの状態を保存:

1. ゲーム中、プレイヤーはゲームを保存するためのアクションを実行します。
2. イベントに応じて、ゲームクライアントは API コールを行い、現在のゲーム状態ファイルをクラウドに保存します。
3. ゲームのバックエンドは、リクエストを受け取ると、ゲームセーブファイル、およびタイムスタンプを含む対応するメタデータを保存します。
4. プレイヤーがゲームを再開すると、同期ステータスが自動的にリクエストされます。AWS GameKit はゲーム保存タイムスタンプを比較し、ローカルバージョンとクラウドバージョンのどちらが最新かを判断します。

ゲーム状態のクラウド保存のソリューションアーキテクチャ

このトピックでは、AWS GameKit のゲーム状態のクラウド保存機能をサポートするため、クラウドベースのバックエンドサービスを提供する AWS ソリューションについて詳しく説明します。AWS GameKit を使用してアチーブメント機能をゲームに組み込んで保守を行う前に、この情報を習得しておく必要はありませんが、ゲームのバックエンドにデプロイされる AWS サービスとリソースをより深く理解するのに役に立ちます。バックエンドのコンポーネントはいつでも AWS で直接確認でき、モニタリングや分析などの他の AWS サービスと併用することもできます。ゲームのバックエンドサービスを AWS GameKit で利用できるサービスを超えてカスタマイズまたは拡張したい場合は、ソリューションの各コンポーネントの役割を理解する必要があります。

このソリューションでは、Lambda 関数を使用して、セーブファイルのローカルバージョンとクラウドバージョンのメタデータを比較します。ファイルを同期するために、この比較により次の 2 つのコールフローのいずれかが促されます。

クラウド保存状態をダウンロード:

1. AWS GameKit は、ローカル保存状態がクラウド保存状態よりも古いことを識別し、現在のスロットに対して署名付き URL (GET) の生成を要求します。
2. Lambda 関数は、`BucketName`、`PlayerID`、`SlotName` の組み合わせから URL を作成し、AWS GameKit がセーブデータを取得するために使用できる署名付き URL を生成します。
3. AWS GameKit は、HTTP GET リクエストを介して指定された URL を使用して S3 からセーブファイルを取得します。

ローカル保存状態をクラウドにアップロード:

1. AWS GameKit は、クラウド保存状態がローカル保存状態よりも古いことを識別し、現在のセーブスロットに対して署名付き URL (PUT) の生成を要求します。
2. AWS GameKit は HTTP PUT リクエストを介してセーブデータを S3 にプッシュします。
3. ファイルがアップロードされると、Lambda 関数がインスタンス化され、そのオブジェクトのメタデータが渡されます。
4. Lambda 関数は、`PlayerID` と `SlotName` の組み合わせを使用して S3 に保存されているオブジェクトを検索し、すべての属性を更新します。

ゲーム状態のクラウド保存サービス

で説明されているように、すべての AWS GameKit AWS ソリューションはコアサービスセットに依存していますコアサービス。

以下のサービスは、特にゲーム状態のクラウド保存アクティビティの管理に使用されます。

Amazon Simple Storage Service (Amazon S3)

AWS GameKit では、Amazon S3 バケットを使用してゲームのセーブファイルを保存します。Amazon S3 は耐久性に優れたオブジェクトストレージ機能を提供します。

Amazon DynamoDB

AWS GameKit では、DynamoDB テーブルを使用してゲームのセーブファイルのメタデータを保存します。DynamoDB を使用してこのタイプのデータを保存することで、ゲームクライアントからの頻繁な読み取り/書き込みリクエストに対応します。

Lambda

AWS GameKit は、Lambda 関数を使用して、ゲームのセーブファイルやメタデータを保存し、同期状態を分析するタスクを管理します。

ゲーム状態のクラウド保存データの暗号化

プレイヤーデータは転送中および保管時に暗号化されます。

転送中、AWS GameKit は AWS でのゲームのフロントエンドおよびバックエンドコンポーネント間の通信に Transport Layer Security (TLS) 1.2 以降を使用します。AWS GameKit のすべてのゲーム機能は、Amazon API Gateway サービスを使用して API コールを受け入れ、処理します。「API Gateway 開発者ガイド」の「Data protection in transit」を参照してください。

保管中のプレイヤー ID データは、アチーブメントのゲーム機能が使用する AWS サービスによって暗号化されます。これらのサービスは業界標準に準拠しています。これらのサービスが保管時のデータ暗号化を処理する方法の詳細については、以下を参照してください。[質問: S3 では、管理キー (SSE-S3) と KMS を使用したサーバー側の暗号化が他にもあります。当社のソリューションはこれらの暗号化をカバーしていますか？ もしそうなら、ここで言及すべきかもしれません。]

- 「Amazon DynamoDB 開発者ガイド」、「保管時の DynamoDB 暗号化」
- 「Amazon S3 ユーザーガイド」、「暗号化を使用したデータの保護」

ゲーム状態のクラウド保存の設定オプション

ゲームにゲーム状態のクラウド保存機能を設定する際に、以下の特性をカスタマイズできます。これらのカスタマイズは、この機能のバックエンドコンポーネントの構築方法に影響します。

- スロット制限: この値セットは、プレイヤーがゲームで使用できるクラウドゲームセーブファイルの数を指定します。このオプションにはハード制限はありません。ゲーム内で許可されるクラウドゲームセーブ数は、このゲーム機能のコストに影響を与え、コストは使用されるストレージ容量によって異なります。(必須)

ゲーム状態のクラウド保存の呼び出し可能なアクション

AWS GameKit API は、ゲーム状態のクラウド保存のゲーム機能用に以下の主要なアクションを提供します。ゲーム状態のクラウド保存用に AWS リソースをデプロイすると、ゲームのフロントエンドはこれらの呼び出しを使用してバックエンドのリソースと通信できます。

これらのアクションはそれぞれ、同期的にも非同期的にも呼び出すことができます。各アクションの詳細については、「AWS GameKit Core C++ API Reference」を参照してください。

- `GetAllSlotSyncStatuses()` は、ローカルまたはクラウドに存在するすべてのゲームセーブスロットの現在の同期ステータスを判断します。これは、ローカルバージョンとクラウドバージョンの両方の最終更新日を比較することによって行われます。同期ステータスは、(1) ゲームのセーブスロットが同期中であること、または (2) バージョンを同期するための推奨アクションを示します。
- `GetSlotSyncStatus` は、指定したゲームセーブスロットの現在の同期ステータスを判断します。これは、ローカルバージョンとクラウドバージョンの両方の最終更新日を比較することによって行われます。同期ステータスは、(1) ゲームのセーブスロットが同期中であること、または (2) バージョンを同期するための推奨アクションを示します。

- `SaveSlot` はローカルのゲームセーブファイルをクラウドにアップロードします。このアクションは、新しいゲームセーブスロットを作成したり、既存のスロットを更新したりするために使用されます。このアクションにより、スロットのメタデータがクラウドだけでなくローカルにも保存されます。
- `LoadSlot` はクラウドに保存されたゲームセーブファイルをローカルデバイスにダウンロードし、ローカルスロットのメタデータを更新します。このアクションは、既存のローカルスロットをクラウドバージョンと同期させるか、存在しない場合は新しいローカルスロットを作成するために使用されます。
- `DeleteSlot` は、クラウドに保存されたゲームセーブファイルとすべてのスロットメタデータをローカルとクラウドの両方で削除します。ローカルバージョンのゲームセーブファイルは削除されません。

ゲーム状態のクラウド保存をゲームに追加

Unreal Engine プロジェクトにゲーム状態のクラウド保存のゲーム機能を追加する基本的な手順を以下に説明します。ゲームエンジン用の AWS GameKit をまだ設定していない場合は、「Unreal Engine での AWS GameKit プラグインのインストール」を参照してください。

ステップ 1。ゲーム状態のクラウド保存のゲーム機能をプロジェクトに合わせて設定します。

1. Unreal Editor ツールバーで、[編集] > [プロジェクト設定] を開き、[AwsGameKit] プラグインセクションに移動します。作業する環境を選択し、必要に応じて有効な AWS 認証情報を入力します。詳細については、「ゲーム用の AWS GameKit プラグインのセットアップ」を参照してください。
2. [ゲーム状態のクラウド保存] セクションを開きます。次の設定オプションを指定します。
   - 最大セーブスロット。各プレイヤーに許可するゲームセーブスロットの最大数を指定します。

ステップ 2。ゲーム状態のクラウド保存のバックエンド用に AWS リソースをデプロイします。

1. AWS GameKit プロジェクト設定の [ゲーム状態のクラウド保存] で、AWS リソースアクション [作成] を選択します。このアクションにより、AWS GameKit は、このゲーム機能のバックエンドサービスを実行するための完全な AWS ソリューションをデプロイするよう促されます。デプロイ時に、AWS GameKit は最初にカスタム設定を含む AWS ソリューションのテンプレートを生成し、次に AWS に接続してテンプレートで定義されているソリューションの AWS リソースを作成します。AWS リソースは、アクティブな環境用に選択された AWS リージョンにデプロイされます。

2. ゲーム状態のクラウド保存のバックエンド用にリソースをデプロイするには、通常 5 分かかります。デプロイステータスの進行状況を次のように追跡できます。

   - Unreal Editor の AWS GameKit プロジェクト設定で、デプロイステータスを更新するか、このゲーム機能のカスタムダッシュボードを開きます。これらのダッシュボードはデプロイプロセスの開始時に生成されます。
   - Unreal Editor の [Window] > [開発者ツール] > [出力ログ] で出力ログを開き、デプロイ全体のステータスメッセージ、イベント、エラーを監視します。
   - AWS マネジメントコンソールで AWS CloudFormation サービスを開きます。[スタック] ビューでは、ゲームプロジェクトのゲーム状態のクラウド保存スタックがデプロイされている様子を見ることができます。

   デプロイが完了すると、ゲーム状態のクラウド保存のゲーム機能のゲームバックエンドが完成します。AWS GameKit API を使用して呼び出すことができます。

   > **Note**
   >
   > 注: AWS 無料利用枠の対象かどうかによっては、この時点からこのゲーム機能のコストが発生し始める可能性があります。

ステップ 3。ゲーム状態のクラウド保存機能をゲームに追加します。

UI 要素を作成し、ワークフローのコードを追加して、ゲームのニーズに応じてゲーム状態を保存および取得します。説明については、プラグインのゲーム状態のクラウド保存のサンプルを参照してください。

ワークフローには以下が含まれる場合があります。

- ゲームセーブファイルをクラウドにアップロード: ファイルを保存。
- 全スロットの同期ステータスを確認: すべての同期ステータスを取得。
- クラウドからゲームセーブファイルをダウンロード: ファイルをロード。
- クラウド内のゲームセーブファイルの削除: スロットを削除。

ゲーム状態のクラウド保存のサンプルの操作

AWS GameKit プラグインには、ゲーム状態のクラウド保存のゲーム機能用のサンプルアセットが含まれています。

Unreal Engine

Unreal Engine 用 AWS GameKit プラグインには、C++ コード、ブループリント、UI コンポーネントを含むサンプルが用意されています。サンプルファイルには Unreal Editor のコンテンツブラウザからアクセスできます。

- C++ サンプルの操作
- ブループリントと UI サンプルの操作

C++ サンプルの操作

Unreal Editor のコンテンツブラウザで、以下の場所にあるサンプルアセットを探します。

```
AwsGameKit C++ Classes > AwsGameKitEditor > Public > GameStateCloudSaving >
 AwsGameKitGameStateCloudSavingExamples
```

このアセットは .cpp ファイルです。このファイルは、`...\AwsGameKit\Source\AwsGameKitEditor\Private\GameStateCloudSaving\AwsGameKitGameStateCloudSavingExamples.cpp` にある AWS GameKit プラグインファイルにもあります。

このサンプルファイルには、ゲーム状態のクラウド保存 API アクションをそれぞれ呼び出す方法を示すサンプルコードが含まれています。このファイルには、実行可能なコードの基本セットと詳細なコメントが含まれています。

このサンプルは 2 つの方法で操作できます。1 つは IDE でコードを表示する方法、もう 1 つは Unreal Editor で API コールを試してみることです。

サンプルコードを表示または編集するには:

- AwsGameKitGameStateCloudSavingExamples アセットをダブルクリックして IDE でファイルを開きます。このファイルにアクセスする前に AWS 認証情報を入力したり、AWS リソー

スをデプロイしたりする必要はありませんが、デプロイされたリソースがないと API コールは動作しません。

サンプルコードには、AWS リソースがデプロイされていることを確認する標準チェックが含まれています。

最初のステップとして、サンプルコードは `UAwsGameKitGameStateCloudSavingCallableWrapper` のインスタンスを作成して初期化することに注意してください。これは API コールを行う前に行う必要があります。

Unreal Editor でサンプルを試すには:

ID と認証用の AWS リソースをデプロイし、AWS GameKit プラグインのプロジェクト設定で有効な AWS 認証情報を送信する必要があります (「ゲーム用の AWS GameKit プラグインのセットアップ」を参照)。

1. AwsGameKitGameStateCloudSavingExamples アセットをビューポートのレベルにドラッグします。アセットはどのレベルに追加してもかまいません。これはゲーム機能の設定を操作できるようにする単なるメカニズムです。
2. Editor の [詳細] ペインで、すべてのゲーム状態のクラウド保存 API コールを API リクエスト値とレスポンス値と共に使用できます。
3. API コールを行うには、入力値を指定して呼び出しボタンをクリックします。レスポンスは「戻り値」フィールドに表示されます。
4. 以下の呼び出しシーケンスを実行して、標準のゲーム状態のクラウド保存シナリオをシミュレートしてみてください。

   - ゲームセーブファイルをクラウドにアップロード: ファイルを保存。
   - 全スロットの同期ステータスを確認: すべての同期ステータスを取得。
   - クラウドからゲームセーブファイルをダウンロード: ファイルをロード。
   - クラウド内のゲームセーブファイルの削除: スロットを削除。

ブループリントと UI サンプルの操作

Unreal Editor のコンテンツブラウザで、以下の場所にあるサンプルアセットを探します。

```
AwsGameKit Content > GameStateCloudSaving >
```

ここでは、以下の 2 つのサンプルアセットを示します。

- BP_AwsGameKitGameStateCloudSavingExamples

  このアセットは、ゲーム状態関連の機能をゲームコードに追加する方法を示す基本的なブループリントです。

  アクション:

  - ブループリントを開くには、アセットをダブルクリックします。
- BP_AwsGameKitGameStateCloudSavingExampleUI

  このアセットには、アチーブメント情報とプレイヤーのステータスを表示するためのブループリントとサンプル UI オブジェクトが含まれています。

  アクション:

  - ブループリントを開くには、アセットをダブルクリックします。

  サンプルを実行するには、[再生] をクリックします。

AWS GameKit 機能: アチーブメント

AWS GameKit のアチーブメント機能には、ゲームのアチーブメントシステムを管理するためのツールが用意されています。アチーブメントはプレイヤーにゲームの特定の側面を探求するよう招待するもので、プレイヤーのエンゲージメントを高めることが実証されています。アチーブメントでは、特定のプレイヤーのアクションをマークしたり、複数のステップからなる目標に向けたプレイヤーの進捗状況を追跡したりできます。アチーブメントは自慢する権利を提供する場合もあれば、より具体的な報酬をもたらす場合もあります。

アチーブメントのゲームでの一般的な使用方法は次のとおりです。

- ゲームのマイルストーンをマークし、プレイヤーの進行状況を追跡します。「ミュータントモンキーを 100 匹捕獲する」といったマイルストーンをゲームの重要な目標とすることができるかもしれません。また、プログレッシブアチーブメント (「おめでとう、セルリアンメイジのステータスを獲得しました!」) は、ゲームを進めていく中でのプレイヤーの継続的な達成感を高めます。
- 明確で達成可能な目標を設定してプレイヤーの関心を維持し、プレイヤーが迷ったり、退屈したり、イライラしたりしないようにしましょう。このタイプのアチーブメントは、非線形な物語やオープンワールドで非常に役立ちます。「森を探検せよ」や「ジンジャーキャットと話せ」などのアチーブメントは、プレイヤーを有意義なゲームプレイに導き、障害物を突破するのに役立つ手がかりとなります。
- ゲーム世界における目標、アクション、その他の概念の意味と価値をプレイヤーが理解できるようにします。「最初の 10 回のトレードを完了して交渉者バッジを獲得」や「手作りのツール 5 個または発見したツール 10 個で自転車を手に入れる」などのアチーブメントは、ゲーム内で何が重要かをプレイヤーに伝える貴重な手がかりとなります。
- プレイヤーに新しいスキルの習得とテクニックのアップグレードを促します。「6 分以内に戦いに勝利する」などのアチーブメントは、新しいスキルや力を試したり、ゲームの後半で必要となるような方法でテクニックを磨くようプレイヤーを促します。
- 獲得が難しいアチーブメントをオプションで用意して、プレイヤーに挑戦してもらいましょう。「スカベンジャーハントのサイドミッションをクリア」や「誰も殺すな」などのチャレンジは、プレイヤーに貴重なオプション体験を与えるだけでなく、ゲームのプレイ方法を根本的に変える可能性さえあります。このようなアチーブメントは、ゲームのリプレイバリューを高めることができます。

> **Note**
>
> この機能を追加するには、AWS GameKit の ID および認証機能も必要です。この機能では、ID 管理を利用して各プレイヤーのアチーブメントステータスを管理します。

トピック

アチーブメントの仕組み

デプロイ後の AWS GameKit アチーブメントのバックエンドには、主に 3 つの機能があります。

- ゲームのすべてのアチーブメントの定義を格納します。定義には、アチーブメント名、獲得方法、オプションの報酬、アイコンやメッセージなどのプレイヤー向けの情報が含まれます。AWS GameKit ツールを使用してアチーブメントの定義を作成し、それをアチーブメントのバックエンドと同期します。アチーブメントがバックエンドに保存されると、プレイヤーはアチーブメントの獲得を開始できます。アチーブメントの定義はいつでも追加または更新できます。ゲームクライアントは AWS GameKit API を使用してアチーブメントの定義のリストをリクエストできます。
- 各プレイヤーのアチーブメントステータスを保存します。プレイヤーがゲームを操作してアチーブメントに向けて前進すると、ゲームクライアントはバックエンドに AWS GameKit API リクエストを送信してプレイヤーのステータスを更新します。ゲームクライアントは、獲得した報酬や複数ステップの報酬に向けた進捗状況など、プレイヤーのアチーブメントステータスをリクエストすることもできます。
- ロジックを実行して、プレイヤーがアチーブメントを獲得したかどうかを評価します。バックエンドは、アチーブメントの定義とプレイヤーのステータスを使用して、Lambda 関数を利用してプレイヤーがアチーブメントの要件を満たしているかどうかを判断します。獲得ステータスは、プレイヤーのステータスを追跡するだけでなく、ゲームクライアントからのアチーブメント情報のリクエストに対するバックエンドの対応方法にも影響します。

アチーブメントのタイプ

AWS GameKit のアチーブメントのゲーム機能は、以下のアチーブメントタイプをサポートしています。

シングルステップアチーブメント

ステートレスアチーブメントとも呼ばれるこのタイプは、プレイヤーが特定の宝物アイテムを見つけたり、パズルをアンロックしたりするなど、ゲームプレイ中の特定のイベントに応答して獲得されます。ゲームクライアントは、アチーブメントのバックエンドに更新リクエストを送信してイベントに応答します。ステートレスアチーブメントのプレイヤーステータスは、ロック済み (未獲得) またはアンロック (獲得済み) のどちらかです。

マルチステップアチーブメント

ステートフルアチーブメントとも呼ばれるこのタイプは、ゲームプレイ中の一連のイベントに応じて獲得されます。ステートフルアチーブメントの定義では、アチーブメントを獲得するのに必要なステップ数を指定します。イベントが発生すると、ゲームクライアントは番号付きの更新リクエストをアチーブメントのバックエンドに送信し、バックエンドサービスはそれに応答して、ステップ要件が達成されプレイヤーがアチーブメントを獲得するまで、その番号をプレイヤーの現在の進行状況に追加します。例えば、「バナナを 1000 本食べる」というアチーブメントのステップ値は 1000 です。プレイヤーがバナナを食べるたびに、ゲームクライアントはバックエンドに「1」という値を通知し、バックエンドはプレイヤーのステータスを 1 単位ずつ、1000 に達するまで増加させます。このアチーブメントタイプでは、プレイヤーの進行状況をどのように増やしたいかを柔軟に設計できます。10 個ずつ増えるバナナバンチや、100 個ずつ増えるメガバンチを追加できます。ステートフルアチーブメントのプレイヤーステータスは、「ロック済みで未開始」、「ロック済みで進行中」、「アンロック済み」です。

シークレットアチーブメント

このタイプは、アチーブメントを定義したいが、プレイヤーが獲得するまで公開したくない場合に便利です。例えば、「ナチスのスパイが脱走する前に摘発せよ」などのシークレットアチーブメントがあれば、その秘密の計画を明かさずにプレイヤーに報酬を与えることができます。ゲームクライアントからのアチーブメント情報のリクエストに応答する場合、プレイヤーがそのアチーブメントを獲得するまで、アチーブメントのバックエンドにはシークレットアチーブメントは含まれません。アチーブメントの「シークレット」フラグはいつでも追加または削除できます。シークレットアチーブメントはステートレスとすることもステートフルとすることもできます。

隠しアチーブメント

このタイプでは、特定のアチーブメントをゲームから隠すことができます。これは、特別なイベントなど、今後のリリースに向けてアチーブメントをステージングする場合に便利です。ゲームクライアントからのリクエストに応答する場合、アチーブメントのバックエンドは隠しアチーブメントに関する情報を返さず、プレイヤーのステータスを追跡しません。アチーブメントの「隠し」フラグはいつでも追加または削除できます。隠しアチーブメントはステートレスとすることもステートフルとすることもできます。

アチーブメントのワークフロー

アチーブメントシステムの仕組みを、プレイヤー/ゲームクライアントの視点から大まかに説明します。

1. プレイヤーはゲームで何らかの操作を行ってイベントを発生させます。イベントに応じて、ゲームクライアントは AWS GameKit API オペレーションを呼び出し、アチーブメントに向けたプレイヤーの進捗状況を報告します。この呼び出しには、プレイヤー ID、アチーブメントの名前、およびプレイヤーが何をしたか (バナナを 10 本食べる、ドラゴンを倒すなど) を表す数値などの情報が含まれます。
2. アチーブメントのバックエンドがリクエストを受け取ると、指定されたアチーブメントに対するプレイヤーのステータスを更新します。次に、組み込みロジックを実行して、プレイヤーがアチーブメントを獲得したかどうかを判断します。これには、(アチーブメントの定義で指定されている) 目標をプレイヤーの現在の進行状況と比較することが含まれます。プレイヤーがゴールに到達すると、アチーブメントがアンロックされます。
3. ゲームクライアントは AWS GameKit API オペレーションを呼び出して、1 つのアチーブメントまたはすべてのアチーブメントに対するプレイヤーのステータスを取得し、更新された情報をプレイヤーに表示できます。

アチーブメントのソリューションアーキテクチャ

このトピックでは、AWS GameKit のアチーブメント機能をサポートするため、クラウドベースのバックエンドサービスを提供する AWS ソリューションについて詳しく説明します。AWS GameKit を使用してアチーブメント機能をゲームに組み込んで保守を行う前に、この情報を習得しておく必要はありませんが、ゲームのバックエンドにデプロイされる AWS サービスとリソースをより深く理解するのに役に立ちます。バックエンドのコンポーネントはいつでも AWS で直接確認でき、モニタリングや分析などの他の AWS サービスと併用することもできます。ゲームのバックエンドサービスを

AWS GameKit で利用できるサービスを超えてカスタマイズまたは拡張したい場合は、ソリューションの各コンポーネントの役割を理解する必要があります。

アチーブメントのバックエンドアーキテクチャは、次の 2 つの呼び出しフローを管理します。

- アチーブメントの定義を管理する呼び出しフロー。このフローは AWS GameKit プラグインで使用され、プレイヤーがゲームで獲得できるアチーブメントのセットを設定します。
- プレイヤーに関連するアチーブメントアクションとステータスを管理する呼び出しフロー。

どちらのフローでもワークフローの順序は同様です。

1. ゲームクライアントはアチーブメント API オペレーションを呼び出し、これにより AWS GameKit は API Gateway エンドポイントにリクエストを送信するよう促されます。Amazon Cognito は、「ID と認証のソリューションアーキテクチャ」で説明されているように、ゲームクライアントのアクセストークンを検証します。リクエストにプレイヤーのアチーブメントデータが含まれる場合、Amazon Cognito オーソライザーは、アクセストークンがユーザープールで定義されているプレイヤーに有効であることを確認します。
2. 認証が成功すると、ゲームクライアントのリクエストは関連する Lambda 関数に渡されます。
3. Lambda 関数は DynamoDB と対話し、リクエストに従ってデータを保存または取得します。アチーブメントの定義とプレイヤー関連データは、2 つの別々の DynamoDB テーブルに保存されます。1 つはアチーブメントの定義用、もう 1 つはプレイヤー関連のアクション用です。

アチーブメントサービス

で説明されているように、すべての AWS GameKit AWS ソリューションはコアサービスセットに依存していますコアサービス。

以下のサービスは、特にアチーブメントアクティビティの管理に使用されます。

Amazon DynamoDB

AWS GameKit は DynamoDB テーブルを使用してゲームのアチーブメントの定義を保存し、各プレイヤーのアチーブメントステータスを追跡します。DynamoDB を使用してこのタイプのデータを保存することで、ゲームクライアントからの頻繁な読み取り/書き込みリクエストに対応します。

AWS Lambda

AWS GameKit は Lambda 関数を使用して、DynamoDB テーブルでのアチーブメントデータの保存および取得プロセスを管理します。別の Lambda 関数がロジックを実行して、プレイヤーがアチーブメントをいつ正常に獲得したかを判断します。

Amazon Simple Storage Service

AWS GameKit では、Amazon S3 バケットを使用してアチーブメントのイメージファイルを保存します。Amazon S3 は耐久性に優れたオブジェクトストレージ機能を提供します。

Amazon CloudFront

AWS GameKit は CloudFront を使用して、ゲームに表示するアチーブメントのイメージファイルを公開します。CloudFront は、コンテンツをダウンロードする際のレイテンシーを最小限に抑えるため、プレイヤーに地理的に近い場所でコンテンツをキャッシュできるようにするコンテンツ配信システムです。

アチーブメントデータの暗号化

プレイヤーデータは転送中および保管時に暗号化されます。

転送中、AWS GameKit は AWS でのゲームのフロントエンドおよびバックエンドコンポーネント間の通信に Transport Layer Security (TLS) 1.2 以降を使用します。AWS GameKit のすべてのゲーム機能は、Amazon API Gateway サービスを使用して API コールを受け入れ、処理します。「API Gateway 開発者ガイド」の「Data protection in transit」を参照してください。

保管中のプレイヤー ID データは、アチーブメントのゲーム機能が使用する AWS サービスによって暗号化されます。これらのサービスは業界標準に準拠しています。これらのサービスが保管中のデータ暗号化をどのように処理するかについては、以下を参照してください。

- 「Amazon DynamoDB 開発者ガイド」、「保管時の DynamoDB 暗号化」
- 「Amazon S3 ユーザーガイド」、「暗号化を使用したデータの保護」
- 「Amazon CloudFront 開発者ガイド」、「Amazon CloudFront におけるデータ保護」

アチーブメントの設定オプション

アチーブメントのゲーム機能の設定には、ゲームのアチーブメント定義をコンパイルすることが含まれます。アチーブメントを定義する際には、以下の特性を指定できます。

- アチーブメントの一意の名前。(必須)
- プレイヤー向けのアチーブメント名。

- プレイヤー向けのアチーブメントの説明。ロック状態とアンロック状態には別々の説明文字列を指定できます。ゲームクライアントがプレイヤーのアチーブメント情報をリクエストすると、AWS GameKit はプレイヤーがアチーブメントを獲得したかどうかに基づいて文字列を返します。
- プレイヤー向けの画像またはアイコンの URL。ロック状態とアンロック状態には別々の URL を指定できます。ゲームクライアントがプレイヤーのアチーブメント情報をリクエストすると、AWS GameKit はプレイヤーがアチーブメントを獲得したかどうかに基づいて URL を返します。
- 最大値。これは、プレイヤーがアチーブメントを獲得するために到達しなければならない数値です。アチーブメントを獲得するのに単一のイベントが必要なステートレスなアチーブメントでは、この値を 1 に設定できます。アチーブメントまでの進行状況を追跡するステートフルアチーブメントの場合、この値には 1000 (「バナナを 1000 本食べた」などのアチーブメントの場合) などの必要な数を設定します。(必須)
- 付与されたポイント数。アチーブメントを獲得したときにアチーブメントの報酬としてゲームポイントが付与される場合、この値はプレイヤーに与えられるポイント数を示します。
- `Is_stateful` フラグはアチーブメントがステートフル (true) かステートレス (false) かを示します。このフラグは、バックエンドサービスがプレイヤーの現在のステータスを評価して、そのプレイヤーがアチーブメントを獲得したかどうかを判断する方法を決定します。
- `Is_secret` フラグは、ゲームクライアントがアチーブメント情報をいつ取得できるかを示します。このフラグを true に設定すると、プレイヤーがそのアチーブメントを獲得した場合にのみ、アチーブメント情報とプレイヤーのステータスが返されます。
- `Is_hidden` フラグは、そのアチーブメントをゲームで使用できるタイミングを示します。このフラグが true に設定されている場合、このアチーブメントに関する情報は返されず、プレイヤーのステータスは更新できません。
- 並び順番号。このオプションの値は、要求されたときにアチーブメント情報が返される順序を示します。これを使用して、アチーブメントをゲームクライアントのディスプレイにどのように表示するかを決定できます。

アチーブメントの呼び出し可能なアクション

AWS GameKit API は、アチーブメントのゲーム機能用に以下のアクションを提供します。アチーブメント用に AWS リソースをデプロイすると、ゲームのフロントエンドはこれらの呼び出しを使用してバックエンドのリソースと通信できます。

- `UpdateAchievement()` アチーブメントに関するプレイヤーの現在のステータスを更新し、そのプレイヤーがアチーブメントを獲得するための要件を満たしているかどうかを確認し、満たしている場合はアチーブメントを「獲得済み」に設定します。

- `GetAchievement` 表示可能なすべてのアチーブメントについて、プレイヤーの現在のステータスを含む情報を取得します。アチーブメントに非表示のフラグが立てられている場合や、シークレットとしてフラグが立てられていて、プレイヤーがまだアチーブメントを獲得していない場合、アチーブメントは表示できません。
- `GetAchievement` アチーブメント ID で指定された 1 つのアチーブメントについて、プレイヤーの現在のステータスを含む情報を取得します。この呼び出しにより、表示可能なすべてのアチーブメントが返されます。非表示のアチーブメントや獲得していないシークレットアチーブメントは返されません。

プロジェクトへのアチーブメントの追加

概要

完全なクラウドベースのアチーブメントシステムを構築し、Unreal Engineプロジェクトに統合する方法を学びましょう。このトピックでは、開発者がバックエンドサービスを構築し、AWS GameKit API を使用してフロントエンドコードを AWS のバックエンドに接続する方法について説明します。

ゲームのアチーブメント機能を構築する準備ができたら、以下の基本ステップに従ってください。プロジェクトに AWS GameKit をまだインストールしていない場合は、「Unreal Engine での AWS GameKit プラグインのインストール」および「ゲーム用の AWS GameKit プラグインのセットアップ」を参照してください。

ステップ 1。ゲームプロジェクトにアチーブメントのゲーム機能を設定します。

1. Unreal Editor ツールバーで、[編集] > [プロジェクト設定] を開き、[AwsGameKit] プラグインセクションに移動します。作業する環境を選択し、必要に応じて有効な AWS 認証情報を入力します。詳細については、「ゲーム用の AWS GameKit プラグインのセットアップ」を参照してください。
2. [アチーブメント] セクションを展開します。このゲーム機能を使うには、ゲームに含めたいアチーブメントを定義する必要があります。[設定] ボタンを選択し、アチーブメントの定義を追加または更新します。すべてのアチーブメントデータは、AWS と同期するまでローカルにキャッシュされます。また、更新をローカルに保存して後のセッションで使用することもできます。

   AWS GameKit プラグインでアチーブメントの定義を管理する方法はいくつかあります。

   - プラグイン UI を使用してアチーブメントの定義を追加します。エントリは自動的にローカルにキャッシュされます。

- [テンプレートを取得] を選択して、アチーブメントの JSON テンプレートをダウンロードします。テンプレートを直接編集し、[ローカルファイルからインポート] を選択してプラグインにアップロードできます。
- [クラウドからデータを取得] を選択して、クラウドに保存されたアチーブメントの定義を取得します。

いずれの方法でも、アチーブメントの定義が更新されてローカルに保存されます。アチーブメントをゲームで利用する準備ができたら、この機能用に新しい AWS リソースをデプロイするか、アチーブメントのバックエンドが既にデプロイされている場合は、[データをクラウドに保存] を選択してローカルアップデートをクラウドベースのバージョンと同期します。

ステップ 2。アチーブメントのバックエンドに AWS リソースをデプロイします。

1. [アチーブメント] の AWS GameKit 設定を開いたまま、デプロイのコントロールまでスクロールします。AWS リソースアクション [作成] を選択します。このアクションにより、AWS GameKit はこの機能のバックエンドサービスの完全な AWS ソリューションをデプロイするよう指示されます。デプロイ時に、AWS GameKit は AWS に接続し、アクティブな環境の設定テンプレートを使用して、ソリューション用のすべての AWS リソースを作成します。
2. ID と認証のためのリソースのデプロイは、完了するまでに通常は 30 分かかります。この間、機能のデプロイステータスは [リソースのデプロイ中] と表示されます。デプロイステータスの進行状況を追跡できます。

   - Unreal Editor で出力ログウィンドウを開き、デプロイ全体のステータスメッセージ、イベント、エラーを監視します。
   - AWS Management Console で AWS CloudFormation サービスを開きます。[スタック] ビューでは、ゲームプロジェクトの Identity スタックがデプロイされている様子を見ることができます。

   デプロイが完了すると、機能のデプロイステータスは [デプロイ済み] と表示されます。アチーブメント用のゲームのバックエンドが整いました。AWS GameKit API を使用して呼び出すことができます。

> **Note**
>
> 注: この時点から、このゲーム機能のコストが発生し始める可能性があります。まだ AWS 無料利用枠に入っている場合は、無料利用枠の制限を超えた場合にのみ費用が発生します。

3. この機能用のクラウドバックエンドをデプロイする場合、アチーブメントの定義を AWS のバックエンドサービスと同期します。新しいアチーブメントデータをクラウドにアップロードするには、[データを保存] を選択します。保存したアチーブメントデータをクラウドからローカルマシンにダウンロードするには、[最新バージョンを取得] を選択します。

> **アチーブメントの定義へのアクセス許可とバージョン管理**
>
> デフォルトでは、AWS GameKit ユーザーには、開発環境でのみアチーブメントの定義をクラウドに保存するためのアクセス許可が付与されます。これらのユーザー権限を変更する方法については、「アチーブメントのアクセス許可の管理」を参照してください。
>
> AWS GameKit では、アチーブメントの定義の更新に関するバージョン管理は行いません。チーム開発の場合は、定義の更新を独自のバージョン管理システムで管理し、アチーブメント定義をクラウドバックエンドにアップロードするユーザー権限を制限することをお勧めします。

ステップ 3。アチーブメント機能をゲームに追加します。

ゲームで必要な場合、UI 要素を作成してアチーブメント関連のワークフローのコードを追加します。説明については、プラグインのアチーブメントのサンプルを参照してください。ワークフローには以下が含まれる場合があります。

- すべてのアチーブメントの情報とプレイヤーのステータスを取得
  - `GetAchievements()`
- 単一のアチーブメントの情報とプレイヤーのステータスを取得
  - `GetAchievement()`
- プレイヤーのアチーブメントステータスの更新を促すトリガーをゲームイベントに追加
  - `UpdateAchievement()`

アチーブメントのサンプルを操作する

AWS GameKit プラグインには、アチーブメントのゲーム機能用のサンプルアセットが含まれています。

Unreal Engine

Unreal Engine 用 AWS GameKit プラグインには、C++ コード、ブループリント、UI コンポーネントを含むサンプルが用意されています。サンプルファイルには Unreal Editor のコンテンツブラウザからアクセスできます。

- C++ サンプルの操作
- ブループリントと UI サンプルの操作

C++ サンプルを使って作業してください。

Unreal Editor のコンテンツブラウザで、以下の場所にあるサンプルアセットを探します。

```
AwsGameKit C++ Classes > AwsGameKitEditor > Public > Achievements >
 AwsGameKitAchievementsExamples
```

このアセットは .cpp ファイルです。このファイルは、`...\AwsGameKit\Source\AwsGameKitEditor\Private\Achievements\AwsGameKitAchievementsExamples.cpp` にある AWS GameKit プラグインファイルにもあります。

このサンプルファイルには、ID と認証 API の各アクションを呼び出す方法を示すサンプルコードが含まれています。このファイルには、実行可能なコードの基本セットと詳細なコメントが含まれています。

このサンプルは 2 つの方法で操作できます。1 つは IDE でコードを表示する方法、もう 1 つは Unreal Editor で API 呼び出しを試してみることです。

サンプルコードを表示または編集するには:

- [AwsGameKitAchievementsExamples] アセットをダブルクリックして IDE でファイルを開きます。このファイルにアクセスする前に AWS 認証情報を入力したり、AWS リソースをデプ

ロイしたりする必要はありませんが、デプロイされたリソースがないと API 呼び出しは動作しません。

サンプルコードには、AWS リソースがデプロイされていることを確認する標準チェックが含まれています。

最初のステップとして、サンプルコードは `UAwsGameKitAchievementsCallableWrapper` のインスタンスを作成して初期化することに注意してください。これは API 呼び出しを行う前に行う必要があります。

Unreal Editor でサンプルを試すには:

ID と認証用の AWS リソースをデプロイし、AWS GameKit プラグインのプロジェクト設定で有効な AWS 認証情報を送信する必要があります (「ゲーム用の AWS GameKit プラグインのセットアップ」を参照)。

1. [AwsGameKitAchievementsExamples] アセットをビューポートのレベルにドラッグします。アセットはどのレベルに追加してもかまいません。これはアセットの設定を操作できるようにする単なるメカニズムです。
2. Editor の [詳細] ペインで、すべてのアチーブメント API 呼び出しを API リクエスト値とレスポンス値と共に使用できます。
3. API 呼び出しを行うには、入力値を指定して [呼び出し] をクリックします。レスポンスは [戻り値] フィールドに表示されます。
4. 以下の呼び出しシーケンスを実行して、標準のアチーブメントシナリオをシミュレートしてみてください。

   - すべての表示可能なアチーブメント情報とプレイヤーのステータスを取得: アチーブメントを取得。
   - 単一のアチーブメントの情報とプレイヤーのステータスを取得: アチーブメントを取得。
   - ステートレスアチーブメントのプレイヤーステータスを更新し、獲得ステータスを確認: アチーブメントを更新、アチーブメントを取得。ステートレスアチーブメントの場合は、現在の値を「1」に更新し、獲得ステータスが「true」になっていることを確認します。

ブループリントと UI サンプルを使って作業してください。

Unreal Editor のコンテンツブラウザで、以下の場所にあるサンプルアセットを探します。

```
AwsGameKit Content > Achievements >
```

ここでは、以下の 2 つのサンプルアセットを示します。

- BP_AwsGameKitAchievementsExamples

  このアセットは、アチーブメント関連の機能をゲームコードに追加する方法を示す基本的なブループリントです。

  アクション:

  - ブループリントを開くには、アセットをダブルクリックします。
- BP_AwsGameKitAchievementsExamplesUI

  このアセットには、アチーブメント情報とプレイヤーのステータスを表示するためのブループリントとサンプル UI オブジェクトが含まれています。

  アクション:

  - ブループリントを開くには、アセットをダブルクリックします。

  サンプルを実行するには、[再生] をクリックします。

AWS GameKit 機能を使用してゲームを起動する

ゲームプロジェクトをディストリビューション用にパッケージ化する場合は、AWS 上でゲームバックエンドサービスと連携するようにゲームを正しくセットアップするために、いくつかの追加手順を実行する必要があります。これらのステップは、AWS GameKit API を呼び出すすべてのプロジェクトに必要です。

トピック

- AWS GameKit 機能を使用してゲームプロジェクトをパッケージ化する
- モバイル向けにゲームを最適化する
- AWS GameKit バックエンドを本番環境での運用に向けて準備する

AWS GameKit 機能を使用してゲームプロジェクトをパッケージ化する

ゲームプロジェクトをディストリビューション用にパッケージ化する場合は、AWS 上でゲームバックエンドサービスと連携するようにゲームを正しくセットアップするために、いくつかの追加手順を実行する必要があります。これらのステップは、AWS GameKit API を呼び出すすべてのプロジェクトに必要です。

トピック

- Windows 用または macOS 用ゲームをパッケージ化する
- iOS 用ゲームをパッケージ化する
- Android 用ゲームをパッケージ化する

Windows 用または macOS 用ゲームをパッケージ化する

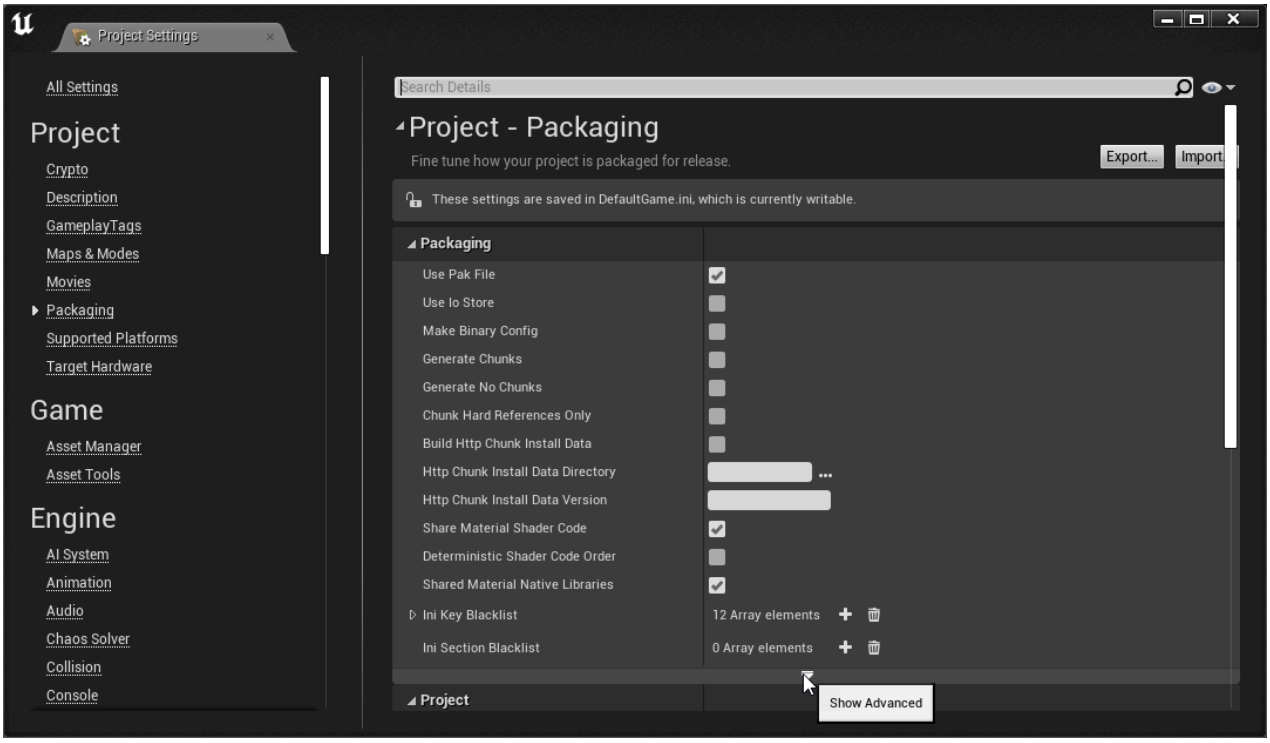
Windows または macOS ディストリビューション用ゲームをパッケージ化する場合は、次のタスクを完了させます。

Unreal Engine

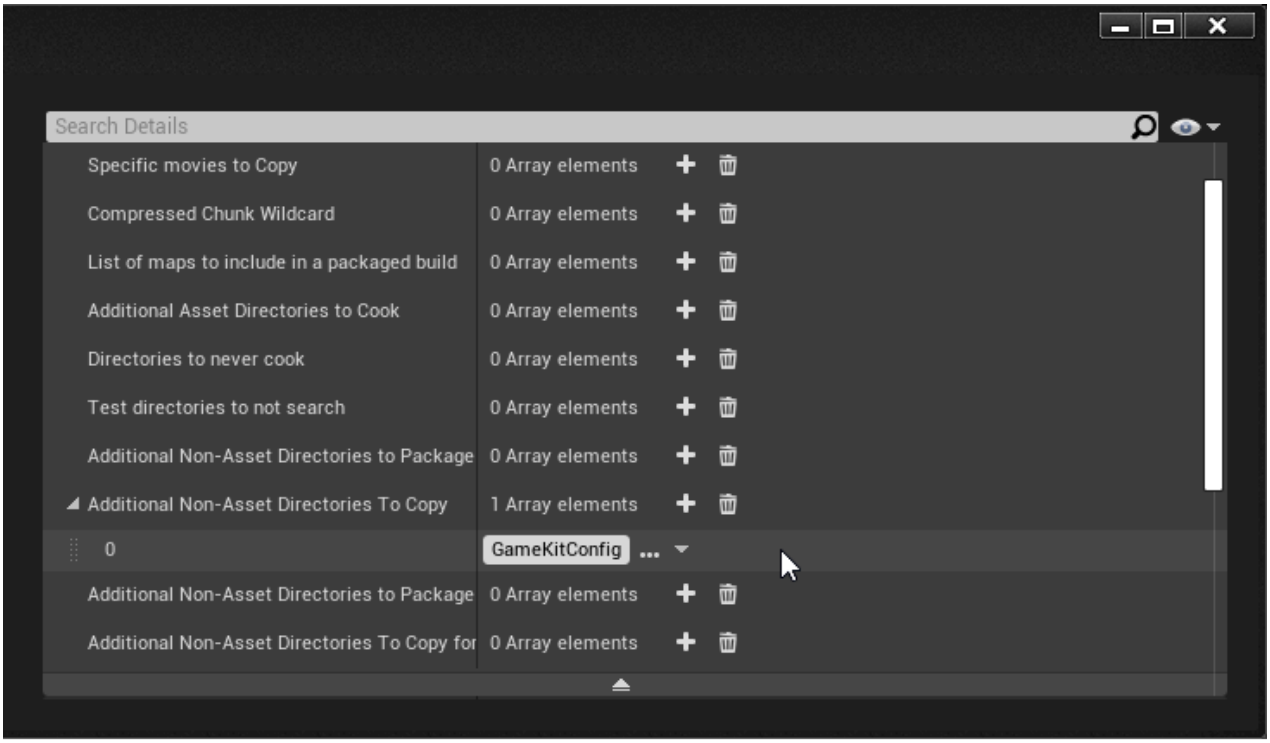
Unreal Engine のゲームプロジェクトについては、Unreal Engine プロジェクトのパッケージ化手順に従い、以下の変更を加えてください。

1. パッケージ化セットアッププロセスを開始する前に、現在の AWS GameKit プラグイン設定を確認します。Unreal のパッケージ化プロセスでは、現在アクティブな AWS GameKit 設定のいずれかを使用します。Unreal Editor ツールバーで、[編集] > [プロジェクト設定] を開き、[AwsGameKit] プラグインセクションに移動します。[環境と認証情報] セクションで、ゲームタイトル、環境、リージョンの選択内容を確認します。
2. Unreal のパッケージ化手順「Setting a Game Default Map」を完了させます。
3. パッケージ化設定で AWS GameKit 設定を追加します。

   a. Unreal Editor ツールバーで、[編集]、[プロジェクト設定]、[パッケージ化] の順に選択し、図のように [パッケージ化] の詳細設定を展開します。



   b. [コピーする追加の非アセットディレクトリ] で、配列要素を追加し、値 **`GameKitConfig`** を入力します。



4. ゲームを出荷用にパッケージ化する準備ができたら、Unreal 自動化ツールの BuildCookRun コマンドを使用します。Unreal Editor を終了し、コマンドプロンプトを開いて、次のコマンドを実行します (実際のゲームプロジェクトに合わせて修正します)。Unreal Editor でゲームパッケージを作成する場合、AWS GameKit プラグインは、Shipping ビルド設定の代わりにゲームの Binaries\ ディレクトリにある .dll ファイルを使用します。

   Windows 用ゲームを (Windows デバイスから) パッケージ化するには:

```
"C:\Program Files\Epic Games\UE_4.27\Engine\Binaries\DotNET\AutomationTool.exe"
 -ScriptsForProject="[PATH_TO_GAME].uproject" BuildCookRun -nocompileeditor
 -installed -nop4 -project="[PATH_TO_GAME].uproject" -cook -stage -archive
 -archivedirectory=[DESTINATION_DIRECTORY_ROOT] -package -ue4exe="C:
\Program Files\Epic Games\UE_4.27\Engine\Binaries\Win64\UE4Editor-Cmd.exe"
 -ddc=InstalledDerivedDataBackendGraph -pak -prereqs -nodebuginfo -
targetplatform=Win64 -build -target=[UNREAL_PROJECT_NAME] -clientconfig=Shipping
 -utf8output
```

   - *[PATH_TO_GAME]* — ゲームプロジェクトファイルへのディレクトリパス (例: C:/Unreal Projects/MagicChickenGame/MagicChickenGame)。
   - *[DESTINATION_DIRECTORY_ROOT]* — 完成したパッケージ製品のターゲット場所 (例: C:\Archive)。

- *[UNREAL_PROJECT_NAME]* — Unreal ゲームプロジェクトに関連付けられた名前 (例: MagicChickenGame)。

macOS 用ゲームを (macOS デバイスから) パッケージ化するには:

```
/Users/Shared/Epic\ Games/UE_4.27/Engine/Build/BatchFiles/RunUAT.sh -
ScriptsForProject="[PATH_TO_GAME].uproject" BuildCookRun -nocompileeditor -
installed -nop4 -project="[PATH_TO_GAME].uproject" -cook -stage -archive -
archivedirectory=[DESTINATION_DIRECTORY_ROOT] -package -ue4exe="/Users/Shared/
Epic Games/UE_4.27/Engine/Binaries/Mac/UE4Editor.app/Contents/MacOS/UE4Editor"
 -compressed -ddc=InstalledDerivedDataBackendGraph -pak -prereqs -nodebuginfo -
targetplatform=Mac -build -target=[UNREAL_PROJECT_NAME] -clientconfig=Shipping -
utf8output
```

- *[PATH_TO_GAME]* — ゲームプロジェクトファイルへのディレクトリパス (例: /Users/amansa/Documents/Unreal\ Projects/MagicChickenGame/MagicChickenGame)。
- *[DESTINATION_DIRECTORY_ROOT]* — 完成したパッケージ製品のターゲット場所 (例: /Users/amansa/Documents/Unreal\ Projects/Archive)。
- *[UNREAL_PROJECT_NAME]* — Unreal ゲームプロジェクトに関連付けられた名前 (例: MagicChickenGame)。

iOS 用ゲームをパッケージ化する

iOS 用ゲームをパッケージ化する場合は、次のタスクを完了させます。

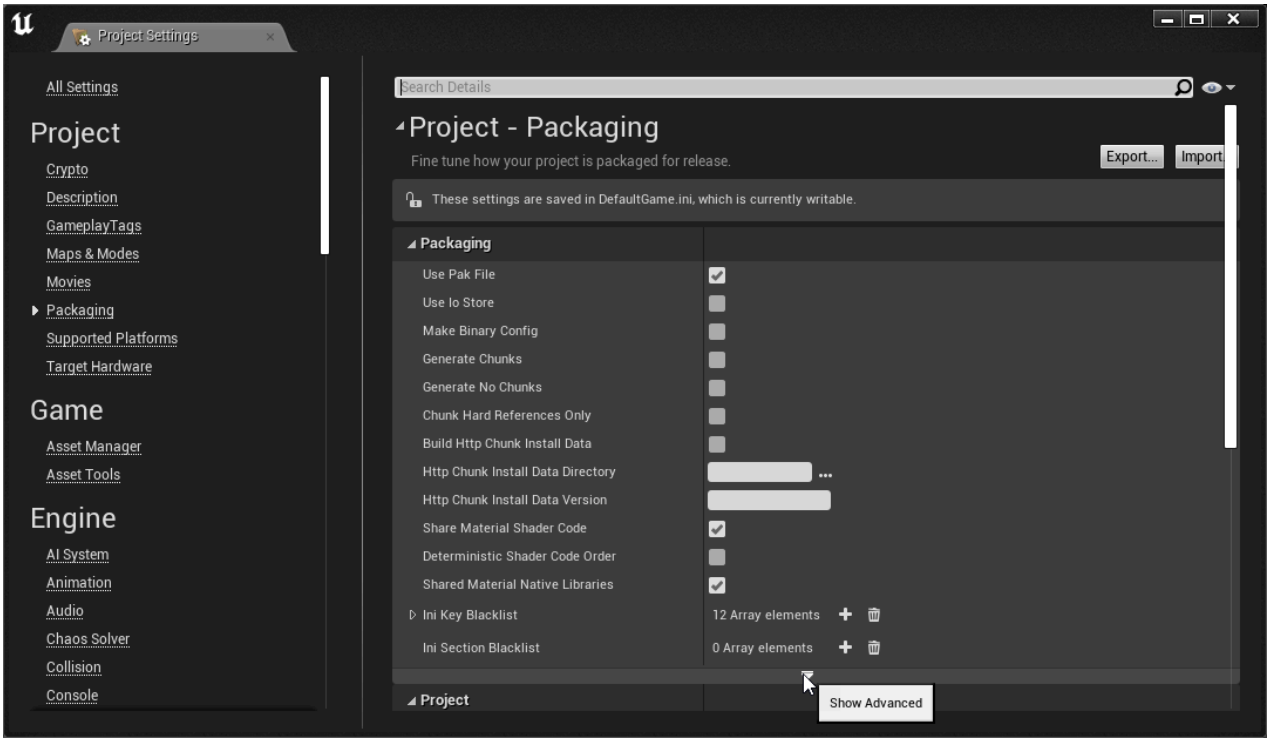
Unreal Engine

Unreal Engine のゲームプロジェクトについては、Unreal Engine プロジェクトのパッケージ化手順に従い、以下の変更を加えてください。

1. パッケージ化セットアッププロセスを開始する前に、現在の AWS GameKit プラグイン設定を確認します。Unreal のパッケージ化プロセスでは、現在アクティブな AWS GameKit 設定のいずれかを使用します。Unreal Editor ツールバーで、[編集] > [プロジェクト設定] を開き、[AwsGameKit] プラグインセクションに移動します。[環境と認証情報] セクションで、ゲームタイトル、環境、リージョンの選択内容を確認します。
2. Unreal のパッケージ化手順「Setting a Game Default Map」を完了させます。

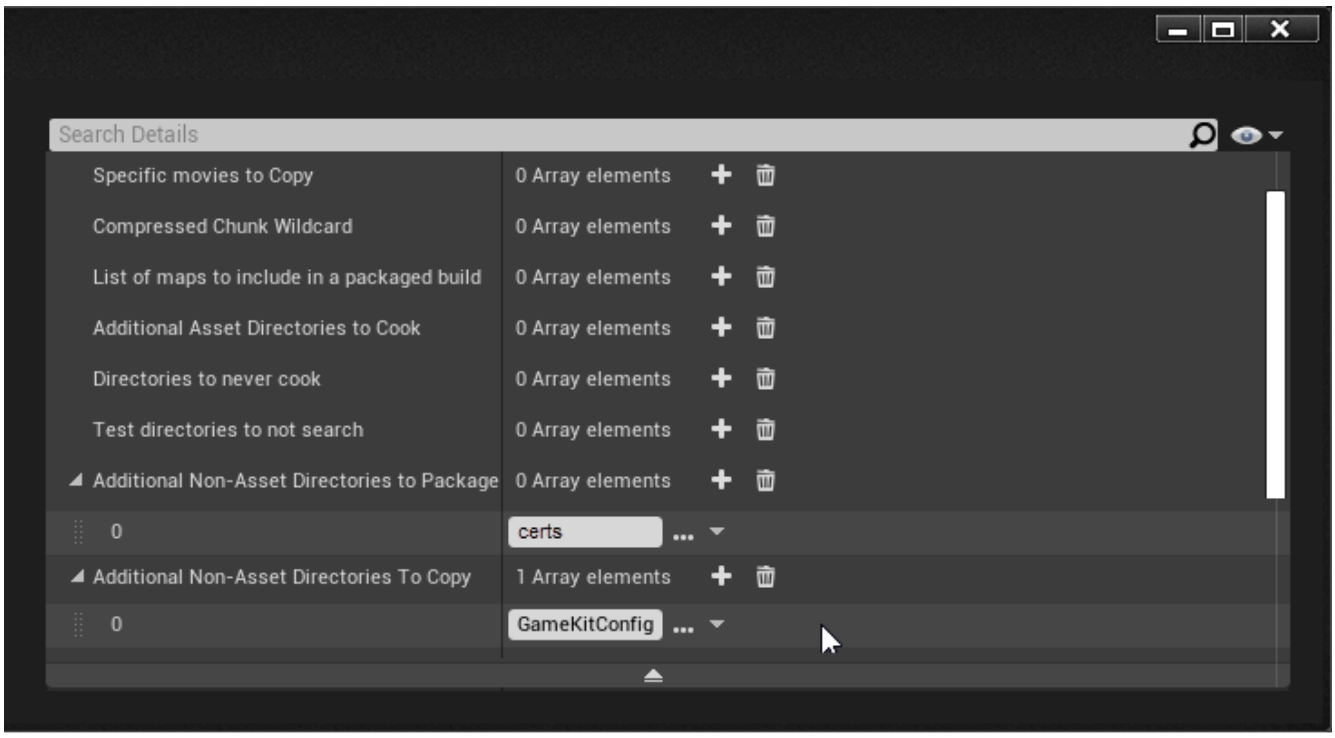
3. パッケージ化設定で AWS GameKit 設定を追加します。

   a. Unreal Editor ツールバーで、[編集]、[プロジェクト設定]、[パッケージ化] の順に選択し、図のように [パッケージ化] の詳細設定を展開します。



   b. [パッケージ化する追加の非アセットディレクトリ] で、配列要素を追加し、ディレクトリ **`certs`** を入力します。このディレクトリは、`Content` の下のゲームプロジェクトファイルにあります。

   c. [コピーする追加の非アセットディレクトリ] で、配列要素を追加し、ディレクトリ **`GameKitConfig`** を入力します。このディレクトリは、`Content` の下のゲームプロジェクトファイルにあります。



4. ゲームプロジェクトファイルの Source ディレクトリにある Target.cs ファイルを開きます。ターゲットコンストラクタに次の行を追加します。

```
if (Target.Platform == UnrealTargetPlatform.IOS){
    bOverrideBuildEnvironment = true;
    GlobalDefinitions.Add("FORCE_ANSI_ALLOCATOR=1");}
```

この更新により、iOS 用のビルド時に FMallocAnsi の使用が強制されます。UnrealBuildTool のターゲットファイルの詳細については、Unreal Engine のドキュメントのトピック「Targets」を参照してください。

5. ゲームを出荷用にパッケージ化する準備ができたら、Unreal のパッケージ化手順「Creating Packages」に進み、ゲームを iOS 用にパッケージ化します。
6. iOS デバイスを接続し、[起動] を選択します。

Android 用ゲームをパッケージ化する

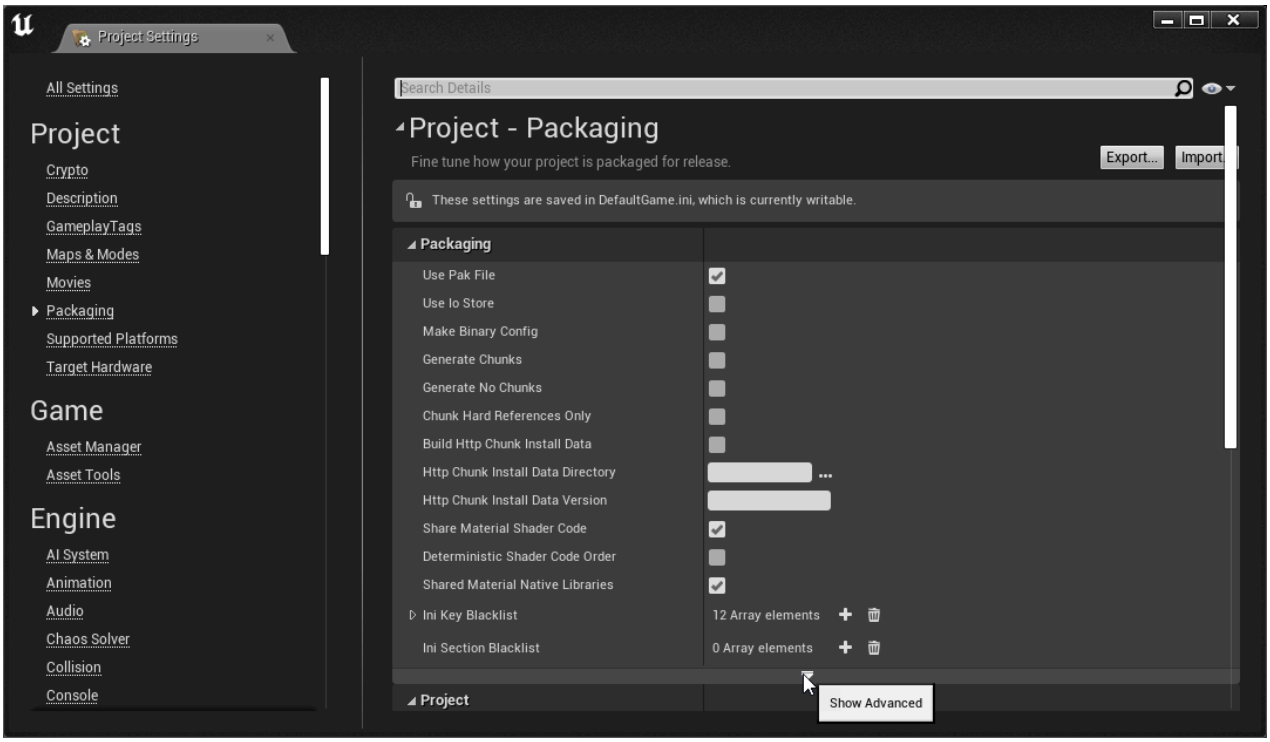
Android 用ゲームをパッケージ化する場合は、次のタスクを完了させます。

Unreal Engine

Unreal Engine のゲームプロジェクトについては、Unreal Engine プロジェクトのパッケージ化手順に従い、以下の変更を加えてください。

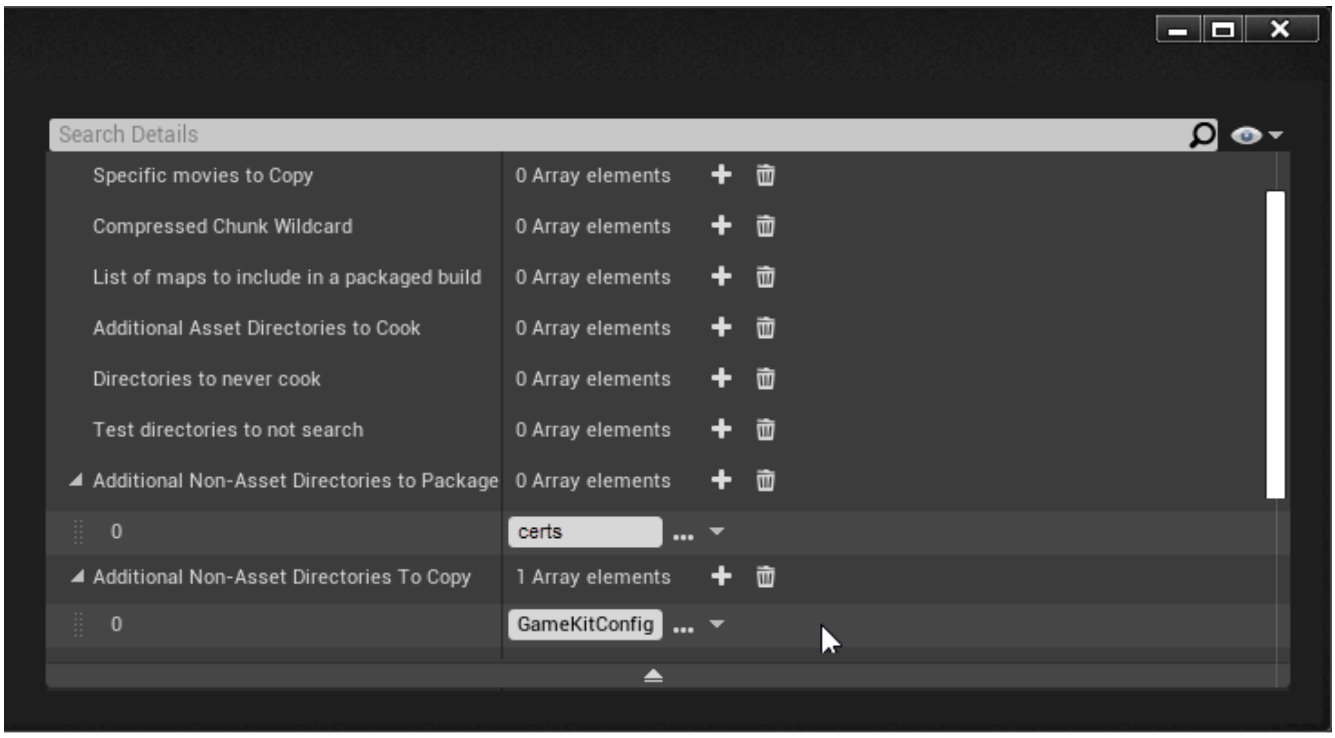
1. パッケージ化セットアッププロセスを開始する前に、現在の AWS GameKit プラグイン設定を確認します。Unreal のパッケージ化プロセスでは、現在アクティブな AWS GameKit 設定のいずれかを使用します。Unreal Editor ツールバーで、[編集] > [プロジェクト設定] を開き、[AwsGameKit] プラグインセクションに移動します。[環境と認証情報] セクションで、ゲームタイトル、環境、リージョンの選択内容を確認します。
2. Unreal のパッケージ化手順「Setting a Game Default Map」を完了させます。
3. パッケージ化設定で AWS GameKit 設定を追加します。

    a. Unreal Editor ツールバーで、[編集]、[プロジェクト設定]、[パッケージ化] の順に選択し、図のように [パッケージ化] の詳細設定を展開します。

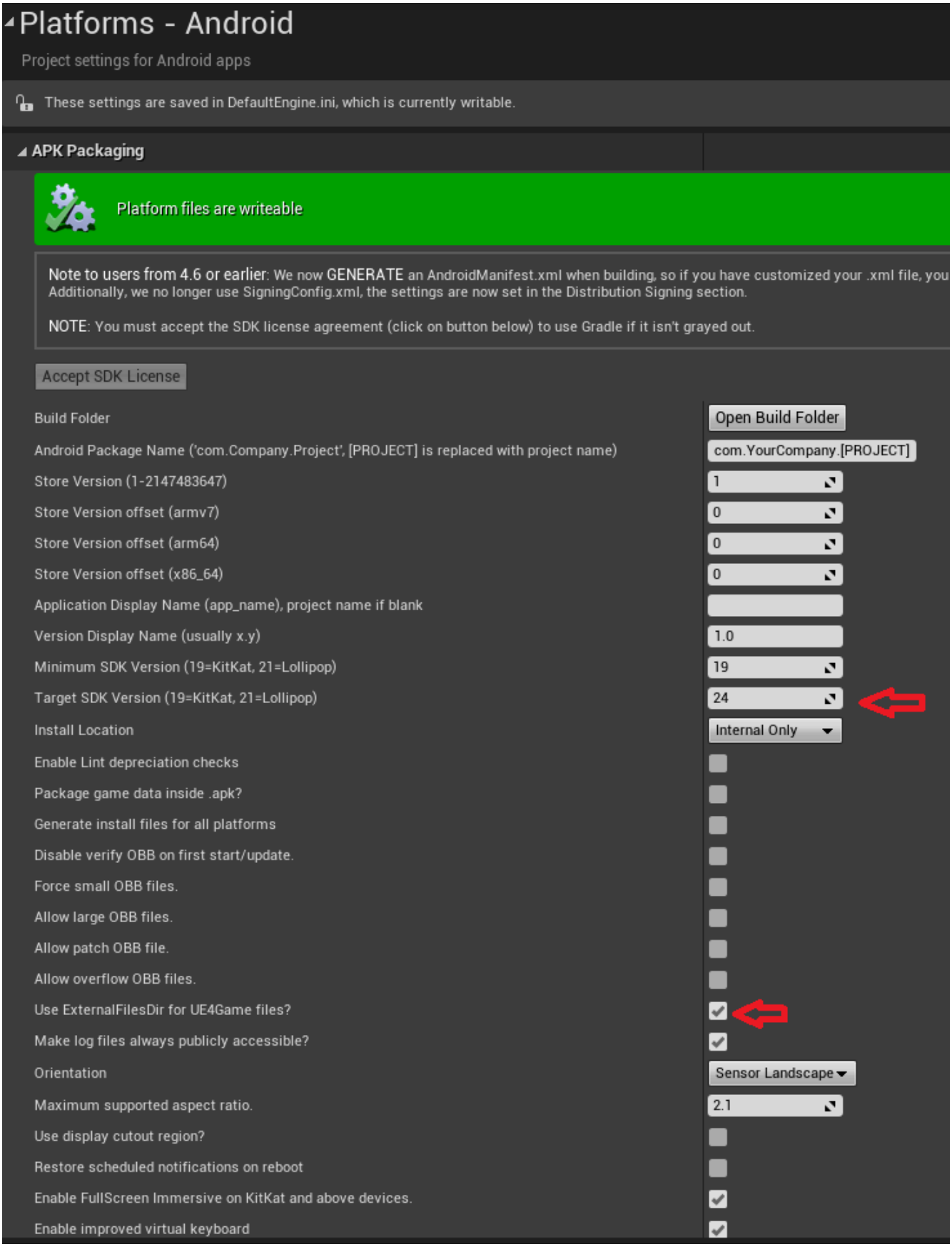


    b. [パッケージ化する追加の非アセットディレクトリ] で、配列要素を追加し、値 **`certs`** を入力します。

    c. [コピーする追加の非アセットディレクトリ] で、配列要素を追加し、値 **`GameKitConfig`** を入力します。



4. Android 用のゲームプロジェクトを設定します。Unreal Editor で、[編集]、[プロジェクト設定]、[プラットフォーム]、[Android] の順に選択して開きます。プロジェクトがまだ設定されていない場合は、[今すぐ設定する] を選択します。
5. [APK パッケージ化] セクションで、次の値を設定します。

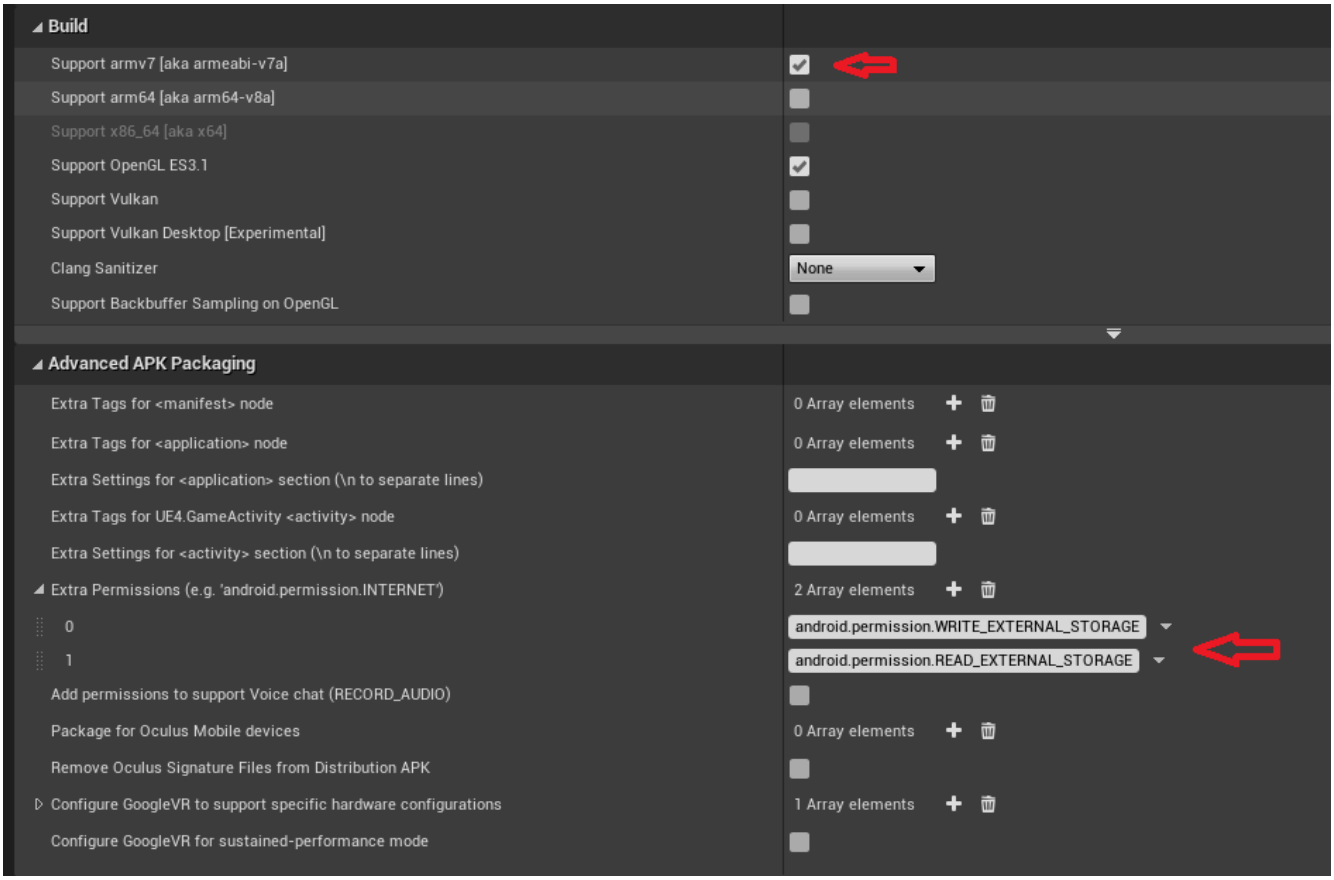

Platforms - Android
Project settings for Android apps
These settings are saved in DefaultEngine.ini, which is currently writable.
APK Packaging
Platform files are writeable
Note to users from 4.6 or earlier: We now GENERATE an AndroidManifest.xml when building, so if you have customized your .xml file, you
Additionally, we no longer use SigningConfig.xml, the settings are now set in the Distribution Signing section.
NOTE: You must accept the SDK license agreement (click on button below) to use Gradle if it isn't grayed out.
Accept SDK License
Build Folder
Open Build Folder
Android Package Name ('com.Company.Project', [PROJECT] is replaced with project name)
com.YourCompany.[PROJECT]
Store Version (1-2147483647)
1
Store Version offset (armv7)
0
Store Version offset (arm64)
0
Store Version offset (x86_64)
0
Application Display Name (app_name), project name if blank
Version Display Name (usually x.y)
1.0
Minimum SDK Version (19=KitKat, 21=Lollipop)
19
Target SDK Version (19=KitKat, 21=Lollipop)
24
Install Location
Internal Only
Enable Lint depreciation checks
Package game data inside .apk?
Generate install files for all platforms
Disable verify OBB on first start/update.
Force small OBB files.
Allow large OBB files.
Allow patch OBB file.
Allow overflow OBB files.
Use ExternalFilesDir for UE4Game files?
Make log files always publicly accessible?
Orientation
Sensor Landscape
Maximum supported aspect ratio.
2.1
Use display cutout region?
Restore scheduled notifications on reboot
Enable FullScreen Immersive on KitKat and above devices.
Enable improved virtual keyboard

a. [ターゲット SDK バージョン] を 24 に設定します。

> **Note**
>
> AWS GameKit はこの SDK バージョンでテストされています。ビルドするデバイスによっては別のバージョンを使用する必要がある場合があります。

b. [UE4Game ファイルには ExternalFilesDir を使用する] オプションを選択します。

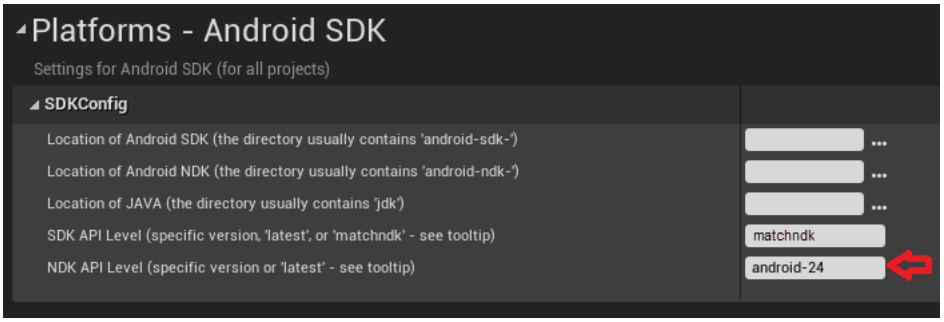
6. [ビルド] セクションで、[armv7 をサポートする] オプションを選択します。



7. [高度な APK パッケージ化] セクションの [追加の権限] で、次の値を持つ配列要素を追加します。

   - **`android.permission.WRITE_EXTERNAL_STORAGE`**
   - **`android.permission.READ_EXTERNAL_STORAGE`**

8. ゲームプロジェクトの [Android SDK] を指定します。Unreal Editor で、[編集]、[プロジェクト設定]、[プラットフォーム]、[Android SDK] の順に選択して開きます。



a. インストールに合わせて、次のディレクトリの場所を設定します。

- Android SDK の場所
- Android NDK の場所
- Java の場所

b. [SDK API レベル] を値 **`matchndk`** に設定します。

c. [NDK API レベル] を値 **`android-24`** に設定します。

9. ゲームを出荷用にパッケージ化する準備ができたら、Unreal のパッケージ化手順「Creating Packages」に進み、選択した Android プラットフォーム用にゲームをパッケージ化します。
10. Android デバイスを接続し、[起動] を選択します。

パッケージ化に関するトラブルシューティング

スクリプト不明エラー

スクリプトが見つからないというエラーメッセージが表示される場合は、ビルドツールのバージョン 30.0.3 を使用しているか確認してください。

API レベルエラー

次のエラーが表示される場合は、Android Studio に Android API レベル 29 をインストールして、もう一度パッケージ化してください。

```
UATHelper: Packaging (Android (ASTC)): …\GameActivity.java:3217: error: cannot find
 symbol
UATHelper: Packaging (Android (ASTC)):
 powerManager.addThermalStatusListener(getMainExecutor(), new
 PowerManager.OnThermalStatusChangedListener()
```

モバイル向けにゲームを最適化する

ゲームをモバイルアプリとして配信する予定の場合は、このような最適化を検討することをお勧めします。ゲームプロジェクトをモバイル向けにパッケージ化する前に、以下の更新を行います。

シャットダウン動作を設定する

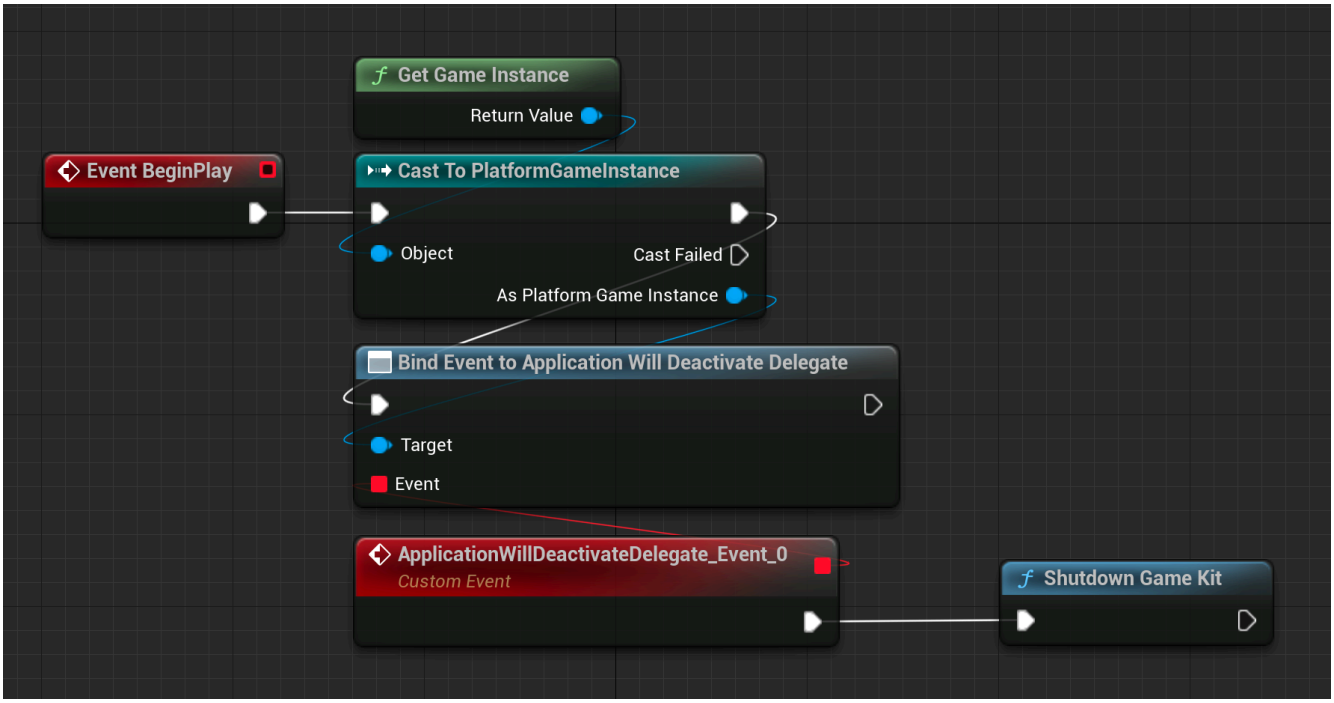
アプリがバックグラウンドで実行されているときはいつでも AWS GameKit コンポーネントをシャットダウンできます。

Unreal Engine

電話やテキストメッセージなど、優先度の高いアクティビティが原因でゲームが非アクティブ化されたときに AWS GameKit アクティビティをシャットダウンするステップをコードに追加します。

ブループリントビジュアルコードを使用するゲームの場合:

1. Unreal Editor で、[編集]、[プロジェクト設定]、[マップとモード] の順に選択して開きます。[ゲームインスタンスクラス] を **`PlatformGameInstance`** に設定します。
2. 次のブループリント例に示すように、AWS GameKit のシャットダウンで終わるロジックを追加します。



C++ コードを使用するゲームの場合:

- ゲームを終了する前に、次のコードを追加して呼び出します。

```
FAwsGameKitCoreModule* coreModule =
 FModuleManager::GetModulePtr<FAwsGameKitCoreModule>("AwsGameKitCore");
FAwsGameKitRuntimeModule* runtimeModule =
 FModuleManager::GetModulePtr<FAwsGameKitRuntimeModule>("AwsGameKitRuntime");
runtimeModule->ShutdownModule();
coreModule->ShutdownModule();
```

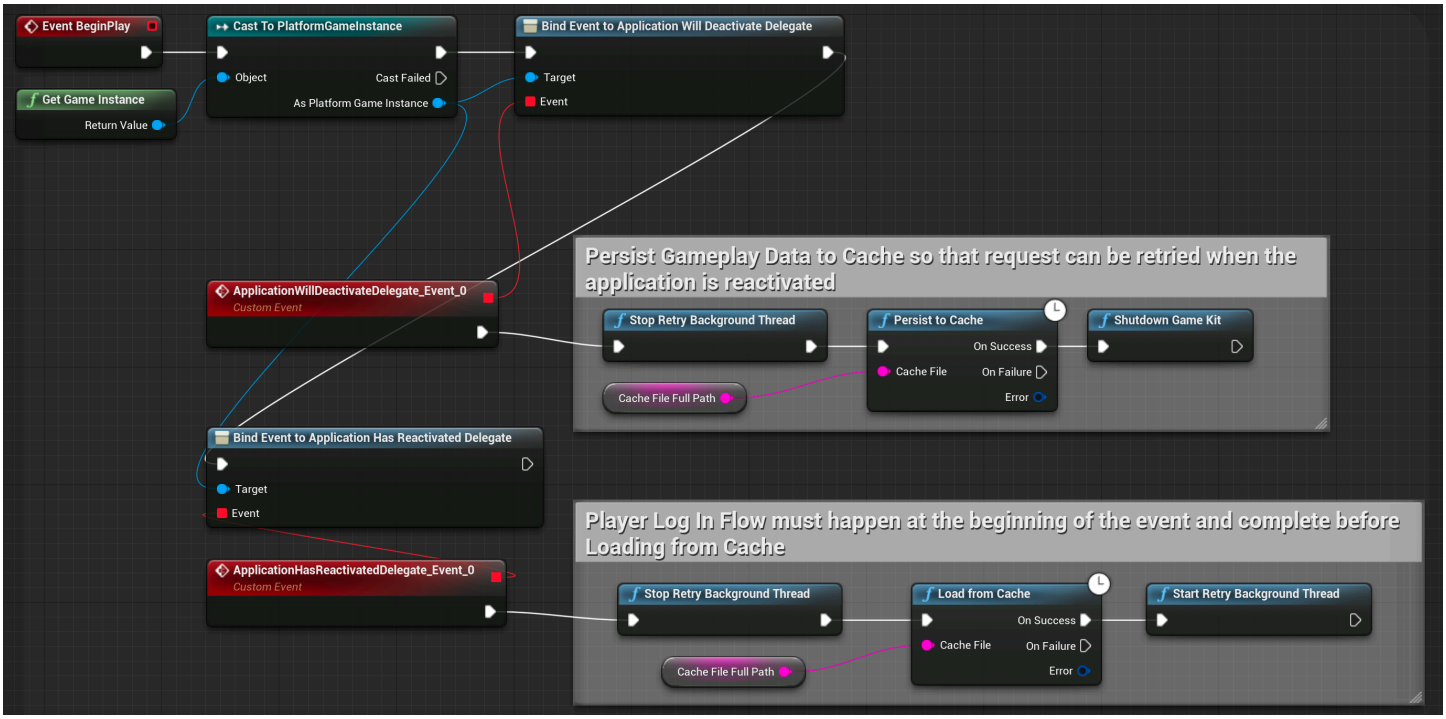
> **Note**
>
> アプリをバックグラウンドに移動したときに iOS 上のゲームを終了させたい場合は、iOS プラットフォームのプロジェクト設定で、次のキーを [追加 Plist データ] に追加します。
>
> ```
> <key>UIApplicationExitsOnSuspend</key><true/>
> ```

ユーザーゲームプレイデータのキャッシュを維持する

ゲームがユーザーゲームプレイデータ機能を使用しており、モバイルデバイスでの配信を目的としている場合は、ゲームに次の変更を加える必要があります。これらの変更により、AWS GameKit がシャットダウンされた場合でも、ユーザーゲームプレイデータ機能の内部リクエストキャッシュが確実に維持されます。キャッシュには、シャットダウン前にクラウドに同期されていなかったユーザーゲームプレイデータの更新が保存され、アプリの再起動時にそれらの更新が行われるようになります。

次の変更を加えます。

- `ApplicationWillDeactivate` デリゲートを使用してキャッシュを維持します。
- `ApplicationWillReactivate` デリゲートを使用して、プレイヤーがログインした後にキャッシュをロードします。



AWS GameKit バックエンドを本番環境での運用に向けて準備する

リリースに向けてプロジェクトを準備する際は、このガイドを参考にして、本番環境レベルの負荷に対応できるように AWS GameKit バックエンドを準備します。

各ゲーム機能の AWS GameKit ソリューションでは、プロジェクトの開発ステージとテストステージに最適な AWS サービスとデフォルト設定値を使用しています。本番環境の準備をするときには、バックエンドサービスを微調整することをお勧めします。特に、本番環境で製品がサポートされるようにキャパシティを調整し、ライブプレイヤーをサポートするサービス機能 (モニタリングなど) を追加することを検討してください。このような変更の中には、追加費用がかかるものもあります。

このトピックでは、AWS GameKit バックエンドのコア AWS サービスを最適化するための推奨事項と手順について説明します。

機能の使用パターンを分析する

開発ステージとテストステージで、AWS GameKit の各機能の使用パターンと負荷パターンを分析することを強くお勧めします。AWS GameKit GitHub リポジトリには、使用状況テストと負荷テストを自動化するためのスターター Python スクリプトが含まれています。バックエンドテストの要件はゲームごとに異なります。要件は、プロジェクトに含まれている AWS GameKit の機能と想定される使用パターンに基づきます。例えば、プロジェクトで多数のアチーブメントが使用されていて、ユー

ザーゲームプレイデータは少量しか保存されない場合は、アチーブメントのバックエンド API に負荷をかけることに重点を置いた方がいいでしょう。

テスト戦略を実践する際は、各機能のカスタム AWS GameKit ダッシュボードで収集された使用状況データを取得します。

モニタリングダッシュボードをセットアップする

テスト環境と本番環境でモニタリングダッシュボードをアクティブ化します。モニタリングダッシュボードは、AWS クラウド上のゲームのバックエンドについてデータ駆動型の意思決定を行う上で不可欠です。使用状況の経時的な変化を追跡し、システムヘルスとコスト効率を維持するための調整を行うことができます。

AWS GameKit には、クラウド機能ごとに詳細なカスタム Amazon CloudWatch ダッシュボードが付属しています。各機能のダッシュボードをアクティブ化または非アクティブ化したり、ゲームエンジンで AWS GameKit 設定から直接アクセスしたりすることができます。ダッシュボードの使用方法の詳細については、「ゲーム機能のダッシュボードを使用する」を参照してください。使用パターンをテストまたは分析するときは、以下のメトリクスに特に注意してください。

AWS Lambda

- レイテンシー (P99、P95、P90)
- 同時実行数
- 関数エラー

Amazon API Gateway

- レイテンシー (P99、P95、P90)
- 4xx および 5xx エラー

Amazon DynamoDB

- スロットル
- テーブルリクエストのレイテンシー

Amazon Cognito

- セキュリティ

ゲームにとって重要なメトリクスについてはアラームを作成することもお勧めします。CloudWatch では、メトリクスがしきい値を超えたときに通知するようにアラームを作成できます。アラームの設定の詳細については、「Amazon CloudWatch ユーザーガイド」のトピック「Amazon CloudWatch でのアラームの使用」を参照してください。

AWS CloudFormation テンプレートを変更する

ゲームで使用される各 AWS GameKit 機能で、その機能の AWS CloudFormation テンプレートを本番稼働前に次のように更新します。本番環境 (本番環境またはカスタム環境) として使用する予定の AWS GameKit 環境にリソースをデプロイする前に、これらの提示された変更を実行しておくことをお勧めします。

- `IsProduction` 条件を追加する
- AWS Lambda の設定を更新する
- Amazon Cognito の設定を更新する
- Amazon DynamoDB の設定を更新する
- カスタムオーソライザーをセットアップする

AWS CloudFormation テンプレートを見つけるには:

本番環境にリソースをまだデプロイしていない場合:

AWS CloudFormation ベーステンプレートに変更を加えます。そして、本番環境で新しい AWS リソースを設定して作成すると、AWS GameKit は、お客様のゲーム用の更新されたベーステンプレートを自動的に使用するようになります。AWS GameKit ベーステンプレートを見つけるには:

- Unreal のインストール場所の AWS GameKit プラグイン:

```
[install location]\Plugins\AwsGameKit\Resources\cloudResources\cloudformation\
```

- Unity プロジェクトファイル:

```
[Unity project]Packages\com.amazonaws.gamekit\Editor]CloudResources\.BaseFiles
```

本番環境で既に AWS リソースを作成している場合:

プロジェクト固有の AWS CloudFormation テンプレートが既に存在しています。それらの既存のテンプレートを更新し、更新した各機能を再デプロイします。プロジェクト固有のテンプレートを見つけるには:

- Unreal ゲームプロジェクトファイル:

```
[Unreal project]\[GameKit game title]\[environment]\cloudformation\
```

- Unity プロジェクトファイル:

```
[Unity project]\Packages\com.amazonaws.gamekit\Editor]CloudResources
\InstanceFiles[GameKit project alias ]\[environment]\[region]\
```

AWS CloudFormation の問題のトラブルシューティングを行うには:

AWS GameKit 機能をデプロイする際に AWS CloudFormation テンプレートに関連する問題が発生した場合は、「AWS CloudFormation ユーザーガイド」のトピック「CloudFormation のトラブルシューティング」を参照して、一般的な問題の解決方法を確認してください。

`IsProduction` 条件を追加する

各テンプレートで `IsProduction` 条件を追加します。

- テンプレートで、次の例に示すように `Parameters` セクションを探し、その下に `Conditions` セクションを追加します。そして、他の本番環境固有のテンプレートの更新のために、`IsProduction` 条件を含めます。

```
```
Parameters:
  ...
Conditions:
   # This condition will toggle certain settings on/off for Production
  IsProduction: !Equals [{ Ref: GameKitEnv }, 'prd']
```
```

AWS Lambda の設定を更新する

すべての AWS GameKit 機能で AWS Lambda が使用されます。使用パターンはそれぞれ異なります。各テンプレートの Lambda の設定に対して次の更新を行います。Lambda の AWS

CloudFormation 構文の詳細については、「AWS CloudFormation ユーザーガイド」のトピック「AWS::Lambda::Function」と「AWS::Lambda::Version」を参照してください。

- `MemorySize` 設定を更新する — 使用パターンによっては、メモリをデフォルトの 128 MB から増やすことを検討してください。例:

```
MyLambdaFunction:
  Type: 'AWS::Lambda::Function'
  Properties:
    # Conditionally set the memory size to 256 for Production and 128 elsewhere
    !If
      - IsProduction
      - MemorySize: 256
      - MemorySize: 128
```

- `ProvisionedConcurrency` 設定を追加する — この設定は、リクエストされた数の実行環境を初期化して、関数の呼び出しに応答する準備を整えます。プロビジョニングされた同時実行を設定すると、AWSに料金が請求されます。プロビジョニングされた同時実行を設定するには、Lambda 関数バージョン (Lambda 関数のバージョン管理されたコピー) を作成し、`ProvisionedConcurrencyConfig` セクションを追加して、`ProvisionedConcurrentExecutions` を本番環境の負荷を処理できる値に設定します。詳細については、「AWS Lambda デベロッパーガイド」のトピック「Lambda 関数のバージョン」を参照してください。例:

```
 MyVersionedLambdaFunction:
   # This configuration creates a Lambda Version with Provisioned Concurrency for
Production
   Type: 'AWS::Lambda::Version'
   DependsOn: MyLambdaFunction
   Properties:
     FunctionName: !Ref MyLambdaFunctionName
     Description: My Lambda Function With Provisioned Concurrency
     ProvisionedConcurrencyConfig:
       !If
         - IsProduction
         - ProvisionedConcurrentExecutions: 4
         - Ref: AWS::NoValue
```

バージョン管理された関数を指すように、必ず関数の参照をすべて更新します。これを行うには、`${MyLambdaFunction.Arn}` の行を `${MyLambdaFunction.Arn}:${MyVersionedLambdaFunction.Version}` に変更します。

Amazon Cognito の設定を更新する

AWS GameKit 機能の ID と認証は、ユーザー登録とログインのワークフローを処理する Amazon Cognito に依存しています。本番環境では次の更新を検討してください。Lambda の AWS CloudFormation 構文の詳細については、「AWS CloudFormation ユーザーガイド」のトピック「AWS::Cognito::UserPool」を参照してください。

- AdvancedSecurityMode 機能を有効にする — この機能により、高度なセキュリティリスク検出が可能になります。この機能を使用するには、ユーザープール設定で ID と認証の AWS CloudFormation テンプレートを次のように変更します。

```
GameKitUserPool:
  Type: 'AWS::Cognito::UserPool'
  Properties:
  UserPoolName: !Ref CognitoUserPoolName
  Schema:
  ...
  Policies:
  ...
  AutoVerifiedAttributes:
  ...
  AliasAttributes:
  ...
  LambdaConfig:
  ...
  UserPoolAddOns:
    # Set AdvancedSecurityMode to AUDIT
    !If
    - IsProduction
    - AdvancedSecurityMode: AUDIT
    - Ref: AWS::NoValue
```

Amazon DynamoDB の設定を更新する

いくつかの AWS GameKit 機能で DynamoDB が使用されます。使用パターンはそれぞれ異なります。各テンプレートの DynamoDB の設定に対して次の更新を行います。DynamoDB の AWS CloudFormation 構文の詳細については、「AWS CloudFormation ユーザーガイド」のトピック「AWS::DynamoDB::Table」と「AWS::ApplicationAutoScaling::ScalableTarget」を参照してください。

- 自動スケーリングを有効にする — 読み取り/書き込みキャパシティユニットに対して DynamoDB 自動スケーリング機能を使用します。この機能を使用すると、製品で、使用状況メトリクスに基づいてキャパシティを調整し、本番環境の負荷の増加を必要に応じて処理することができます。DynamoDB の自動スケーリングの詳細については、「Amazon DynamoDB デベロッパーガイド」のトピック「DynamoDB Auto Scaling によるスループットキャパシティの自動管理」を参照してください。

  自動スケーリングを使用するには、これを必要とする DynamoDB テーブルごとに次の新しいセクションを作成します。予想される負荷に適したユニットとキャパシティを指定します。

  - `MyTableReadCapacityScalableTarget`
  - `MyTableReadScalingPolicy`
  - `MyTableWriteCapacityScalableTarget`
  - `MyTableWriteScalingPolicy`

  例:

```
MyTable:
    Type: 'AWS::DynamoDB::Table'
    Properties:
      ...
      BillingMode: !If [ IsProduction, PROVISIONED, PAY_PER_REQUEST ]
      ProvisionedThroughput:
        !If
        - IsProduction
        - ReadCapacityUnits: 20
          WriteCapacityUnits: 20
        - Ref: AWS::NoValue
      TableName: !Ref MyTableName
  MyTableReadCapacityScalableTarget:
    Type: "AWS::ApplicationAutoScaling::ScalableTarget"
    DependsOn: MyTable
    Condition: IsProduction
```

```
    Properties:
      MaxCapacity: 200
      MinCapacity: 20
      ResourceId: !Sub table/${MyTableName}
      RoleARN: !Sub arn:aws:iam::${AWS::AccountId}:role/aws-
service-role/dynamodb.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_DynamoDBTable
      ScalableDimension: "dynamodb:table:ReadCapacityUnits"
      ServiceNamespace: dynamodb
  MyTableReadScalingPolicy:
    Type: "AWS::ApplicationAutoScaling::ScalingPolicy"
    DependsOn: MyTable
    Condition: IsProduction
    Properties:
      PolicyName: ReadAutoScalingPolicy
      PolicyType: TargetTrackingScaling
      ScalingTargetId:
        Ref: MyTableReadCapacityScalableTarget
      TargetTrackingScalingPolicyConfiguration:
        TargetValue: 70
        ScaleInCooldown: 60
        ScaleOutCooldown: 60
        PredefinedMetricSpecification:
          PredefinedMetricType: DynamoDBReadCapacityUtilization
  MyTableWriteCapacityScalableTarget:
    Type: "AWS::ApplicationAutoScaling::ScalableTarget"
    DependsOn: MyTable
    Condition: IsProduction
    Properties:
      MaxCapacity: 200
      MinCapacity: 20
      ResourceId: !Sub table/${MyTableName}
      RoleARN: !Sub arn:aws:iam::${AWS::AccountId}:role/aws-
service-role/dynamodb.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_DynamoDBTable
      ScalableDimension: "dynamodb:table:WriteCapacityUnits"
      ServiceNamespace: dynamodb
  MyTableWriteScalingPolicy:
    Type: "AWS::ApplicationAutoScaling::ScalingPolicy"
    DependsOn: MyTable
    Condition: IsProduction
    Properties:
      PolicyName: WriteAutoScalingPolicy
      PolicyType: TargetTrackingScaling
```

```
      ScalingTargetId:
        Ref: MyTablWriteCapacityScalableTarget
      TargetTrackingScalingPolicyConfiguration:
        TargetValue: 70
        ScaleInCooldown: 60
        ScaleOutCooldown: 60
        PredefinedMetricSpecification:
          PredefinedMetricType: DynamoDBWriteCapacityUtilization
```

別の例については、ユーザーゲームプレイデータの AWS GameKit ベーステンプレートを参照してください。この機能では、本番環境の DynamoDB 自動スケーリングがデフォルトで有効になっています。

- ワークロードが大きく、消費される読み取りユニット数が多いゲームの場合は、DynamoDB Accelerator の使用を検討してください。この機能の詳細については、「Amazon DynamoDB Accelerator (DAX)」を参照してください。

カスタムオーソライザーをセットアップする

プレイヤーのログインと認証に別のサービス (カスタムまたはサードパーティ) を使用している場合は、カスタムオーソライザーを使用するように AWS GameKit バックエンドを設定します。ID と認証の機能の AWS CloudFormation パラメータファイル (*[GameKit cloud templates]*\identity\parameters.yml) を変更します。

- `UseThirdPartyIdentityProvider` を `TRUE` に設定します。
- `JwksThirdPartyUri` に値を指定します。

ゲームで使用されるすべての AWS GameKit 機能の `parameters.yml` ファイルでこれらの変更を行います。ID と認証の機能の AWS リソースを本番環境にデプロイする前に、これらの変更を行う必要があります。

サービスクォータを増やす

ゲームの予想される使用負荷に応じて、次のサービスクォータの引き上げをリクエストすることを検討してください。ほとんどの AWS サービスにはクォータがあり、使用負荷が高いとゲームのパフォーマンスに影響する可能性があります。

- Amazon API Gateway — (1 リージョン、1 AWS アカウントあたりの)「1 秒あたりのリクエスト数」を増やします。API Gateway アカウントレベルの制限の詳細と引き上げのリクエストについ

ては、「API Gateway デベロッパーガイド」のトピック「Amazon API Gateway のクォータと重要な注意点」を参照してください。

- AWS Lambda —「同時実行」を増やします。詳細および引き上げのリクエストについては、「Lambda クォータ」を参照してください。
- AWS Key Management Service —「1 秒あたりのリクエスト数」を増やします。詳細および引き上げのリクエストについては、「AWS KMS クォータのリクエスト」を参照してください。

プレイヤー登録 E メールをカスタマイズする

ゲームが Amazon Cognito プールと、ID と認証の機能を使用している場合、Amazon Cognito は、新規プレイヤーがゲームに登録したとき、そのプレイヤーに自動的に検証 E メールを送信します。この E メールのデフォルトテキストをカスタマイズするオプションがあります。

プレイヤーの登録検証 E メールをカスタマイズするには:

1. Amazon Cognito の AWS Management Console を開き、[ユーザープールの管理] オプションを選択します。
2. ゲームの本番環境バージョンのユーザープールの名前を選択します。GameKit では、ユーザープール名は `gamekit_[environment]_[game title]_UserPool` のパターンに従います。例: `gamekit_prod_magicchicken_UserPool`。
3. ユーザープール設定が表示された状態で、左側のナビゲーションで [メッセージのカスタマイズ] を選択します。
4. [E メール検証メッセージをカスタマイズしますか?] というタイトルのセクションに移動します。
5. このセクションでは、プレイヤーに検証コード値を提供するように Amazon Cognito に指示するコードオプションを選択し、カスタム E メールの件名とメッセージを入力します。カスタムメッセージでは、必ず検証コードのプレースホルダーを適切に配置してください。最大長などの詳細については、「Amazon Cognito デベロッパーガイド」のトピック「E メール検証メッセージのカスタマイズ」を参照してください。
6. 完了したら、[変更の保存] を選択します。

オプションサービスを追加する

プロジェクトで次のオプションサービスを活用することを検討してください。これらのサービスは AWS GameKit 機能のベーステンプレートには含まれていませんが、いつでも追加できます。

Amazon Simple Email Service

Amazon Simple Email Service (Amazon SES) を使用して、Amazon Cognito で使用されているデフォルトアドレスの代わりにカスタム E メールアドレスからプレイヤー登録検証 E メールを送信します。詳細については、次のトピックを参照してください。

- 「Amazon SES E メールを代理送信するための Amazon Cognito の承認」(「Amazon Cognito デベロッパーガイド」)
- 「Amazon Simple Email Service デベロッパーガイド」

AWS ウェブアプリケーションファイアウォール

AWS ウェブアプリケーションファイアウォール (AWS WAF) は、可用性に影響を与えたり、セキュリティを侵害したり、リソースを過剰に消費させたりする一般的なウェブベースの攻撃からの保護に役立ちます。本番環境のデプロイでは、AWS WAF を有効にしておくことをお勧めします。本番環境に他の機能をデプロイする前に、以下の手順を実行してください。

AWS WAF を追加するには:

1. AWS GameKit ユーザーの IAM アクセス許可を更新します。AWS WAF を使用する AWS リソースをデプロイするユーザーには AWS WAF アクセス許可が必要です。次の構文を使用して新しいアクセス許可ポリシーを作成し、その新しいポリシーを IAM ユーザーグループにアタッチします。AWS GameKit のアクセス許可ポリシーの作成の詳細については、「AWS GameKit アクセス権を持つユーザーのセットアップ」を参照してください。

   この構文は、次のルールを使用して AWS WAF WebACL を作成します。

   - コアルールセット (CRS) (「AWS WAF デベロッパーガイド」の「ベースラインルールグループ」を参照)
   - SQL インジェクションルールセット (「AWS WAF デベロッパーガイド」の「ユースケース固有のルールグループ」を参照)
   - IP 評価ルールセット (「AWS WAF デベロッパーガイド」の「IP 評価ルールグループ」を参照)

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Action": [
                "apigateway:SetWebACL",
                "wafv2:AssociateWebACL",
                "wafv2:CreateIPSet",
                "wafv2:CreateRegexPatternSet",
                "wafv2:CreateRuleGroup",
                "wafv2:CreateWebACL",
                "wafv2:DeleteIPSet",
                "wafv2:DeleteLoggingConfiguration",
                "wafv2:DeleteRegexPatternSet",
                "wafv2:DeleteRuleGroup",
                "wafv2:DeleteWebACL",
                "wafv2:DisassociateWebACL",
                "wafv2:GetWebACL",
                "wafv2:GetWebACLForResource",
                "wafv2:ListTagsForResource"
            ],
            "Resource": "*"
        }
    ]
}
```

2. ゲームのメイン AWS CloudFormation テンプレートを更新します。AWS WAF の AWS CloudFormation 構文の詳細については、「AWS CloudFormation ユーザーガイド」のトピック「AWS::WAFv2::WebACL」と「AWS::WAFv2::WebACLAssociation」を参照してください。

   AWS CloudFormation テンプレートを見つけるには、「AWS CloudFormation テンプレートを変更する」を参照してください。メインテンプレートは *[GameKit cloud templates]*\main\cloudformation.yml です。

   Resources セクションに次の構文を追加します。

```
MainWAFWebAcl:
    Type: AWS::WAFv2::WebACL
    Properties:
        Name: !Sub 'gamekit_${GameKitEnv}_${GameKitGameName}_waf_webacl'
        Description: !Sub 'GameKit ${GameKitEnv} Main stack WebACL for
 ${GameKitGameName}'
        Scope: REGIONAL
        DefaultAction:
          Allow: {}
        VisibilityConfig:
```

```
          SampledRequestsEnabled: true
          CloudWatchMetricsEnabled: true
          MetricName: !Sub gamekit_${GameKitEnv}_${GameKitGameName}_WAF_WebACL
        Rules:
          - Name: AWS-Common-Rule
            Priority: 1
            OverrideAction:
              Count: {}
            Statement:
              ManagedRuleGroupStatement:
                VendorName: AWS
                Name: AWSManagedRulesCommonRuleSet
            VisibilityConfig:
              SampledRequestsEnabled: true
              CloudWatchMetricsEnabled: true
              MetricName: !Sub gamekit_${GameKitEnv}_
${GameKitGameName}_AWS_Common_Rule
          - Name: AWS-SQLInjection-Rule
            Priority: 2
            OverrideAction:
              Count: {}
            Statement:
              ManagedRuleGroupStatement:
                VendorName: AWS
                Name: AWSManagedRulesSQLiRuleSet
            VisibilityConfig:
              SampledRequestsEnabled: true
              CloudWatchMetricsEnabled: true
              MetricName: !Sub gamekit_${GameKitEnv}_
${GameKitGameName}_AWS_SQLInjection_Rule
          - Name: AWS-IPReputation-Rule
            Priority: 3
            OverrideAction:
              Count: {}
            Statement:
              ManagedRuleGroupStatement:
                VendorName: AWS
                Name: AWSManagedRulesAmazonIpReputationList
            VisibilityConfig:
              SampledRequestsEnabled: true
              CloudWatchMetricsEnabled: true
              MetricName: !Sub gamekit_${GameKitEnv}_
${GameKitGameName}_AWS_IPReputation_Rule
        Capacity: 1500
```

```
  MainWebAclRestAssociation:
    Type: AWS::WAFv2::WebACLAssociation
    Properties:
      ResourceArn: !Sub
        - 'arn:aws:apigateway:${AWS::Region}::/restapis/${RestApi}/stages/${Stage}'
        - Stage: !Ref MainDeploymentStage
      WebACLArn: !GetAtt MainWAFWebAcl.Arn
```

3. AWS GameKit 機能をデプロイまたは再デプロイします。このアクションにより、最新の AWS WAF の変更を含むメインスタックが自動的に再デプロイされます。

AWS Shield

AWS Shield Standard は、ウェブサイトやアプリケーションを標的として最も頻繁に発生するネットワークおよび転送レイヤーの DDoS 攻撃に対して防御します。この保護はデフォルトでオンになっています。AWS Shield Advanced は、Amazon Elastic Compute Cloud (Amazon EC2)、Elastic Load Balancing (ELB)、Amazon CloudFront、Global Accelerator、Amazon Route 53 で実行されるインターネット向けアプリケーションをさらに保護する有料サービスです。費用の詳細については、「AWS Shield 料金」を参照してください。

AWS Shield Advanced をオンにするには、次の 2 つの方法があります。

- AWS Shield の AWS Management Console を使用してカバレッジを設定します。
- 「AWS CloudFormation ユーザーガイド」のトピック「AWS::FMS::Policy」で説明されているように AWS CloudFormation テンプレートを更新します (例を参照)。

AWS GameKit クライアント API の使用状況を調整する

ユーザーゲームプレイデータ機能では、多数 (数百ほど) のバンドルまたはアイテムで AWS GameKit API を呼び出すと、HTTP リクエストがタイムアウトする可能性があります。タイムアウトのリスクを軽減する方法として以下が挙げられます。

- データを小さなバッチにして API コールを行う。
- アカウント作成時にユーザーのバンドルを設定する。
- `FUserGameplayDataClientSettings` で `ClientTimeoutSeconds` の値を増やす。

AWS リソースの使用

AWS GameKit プラグインを使用してゲーム機能のバックエンドサービスをホストする AWS リソースをデプロイすると、AWS GameKit は AWS CloudFormation サービスを使用してこれらのリソースの更新を作成および管理します。

AWS GameKit はゲーム機能の設定を取得し、この情報を使用してゲーム機能のバックエンドソリューション用の AWS CloudFormation テンプレートを構築します。AWS CloudFormation テンプレートはソリューション内の AWS リソースのセットを記述し、AWS CloudFormation はこのテンプレートを使用して AWS GameKit のゲーム機能をサポートするリソーススタックをデプロイします。

AWS Management Console や AWS Command Line Interface (AWS CLI) などの AWS ツールを使用して、AWS GameKit で作成されたテンプレートとリソーススタックを直接表示または変更することができます。

トピック

AWS テンプレートとリソースの表示

AWS Management Console を使用して、AWS GameKit がゲーム機能用に生成する AWS CloudFormation テンプレートとリソーススタックを表示できます。

AWS リソーススタックを表示する

1. AWS CloudFormation コンソールを開きます。コンソールにサインインするときは、AWS GameKit プラグインで使用したものと同じ AWS ユーザーを使用してリソースをデプロイします。AWS ユーザーには、ゲームエンジンに必要なセキュリティ認証情報とは異なるサインイン認証情報が必要です。
2. [スタック] ページを開きます。AWS CloudFormation コンソールでこのページがすぐに開かない場合は、左側のナビゲーションペインに移動して [スタック] を選択します。[スタック] ページには、AWS アカウントが所有するすべてのリソーススタックが表示されます。
3. AWS CloudFormation コンソールで、AWS リージョンコントロールを使用して AWS GameKit リソースがデプロイされるリージョンを選択します。

4. 一覧表示されたスタックから、表示するスタックを選択します。
5. 「Identity」など、機能の名前で終わるスタック名を開きます。[スタックの詳細] ページには、スタックに関する一般的な情報が含まれます。現在の AWS CloudFormation テンプレート設定にアクセスしたり、スタック内の AWS リソースのリストを表示したり、スタックリソースとテンプレートに関連するアクティビティを追跡するイベントログを開いたりできます。

AWS GameKit 機能の AWS リソースを表示するには

- AWS CloudFormation コンソールの [スタック] ページのリソースリストには、現在スタック内にあるデプロイされたすべての AWS リソースの物理 ID リンクが含まれています。これらのリンクを使用してリソースのサービスコンソールを開き、そのリソースに関する詳細情報を取得します。

  例えば、ID /認証機能スタックでは、ユーザープールリソースの物理 ID リンクをたどることができます。このリンクをクリックすると Amazon Cognito サービスコンソールが開き、ユーザープールリソースをより詳細に調べることができます。

AWS テンプレートとリソースの更新

概要

AWS マネジメントコンソールなどの AWS ツールを使用すると、AWS GameKit にデプロイされた AWS リソースに直接アクセスできます。これらのリソースや設定テンプレートを直接更新する場合、AWS GameKit のゲーム機能にエラーが生じる可能性があることに注意してください。このトピックは、AWS ツールを使ってゲームのバックエンドコンポーネントについてより深く理解し、更新によってゲーム機能にどのような影響が及ぶかを理解したい開発者を対象としています。

AWS GameKit プラグインのみで作業する場合、AWS 上の AWS GameKit で作成したテンプレートやリソースは、常に機能設定と一致したままとなります。プラグインの外部で AWS CloudFormation のテンプレートやリソースに変更を加えると、整合性が取れなくなり、競合や予期しない動作が発生するリスクがあります。このような理由から、プラグインを引き続き使用する予定がある場合は、AWS GameKit で作成したテンプレートとリソースを更新しないことをお勧めします。

AWS CloudFormation テンプレートを直接更新する場合は、その方法が更新の処理方法にどのように影響するかに注意してください。

1. プラグインを使用して機能設定を更新し、再デプロイします。再デプロイ時に、AWS GameKit はまず既存の CloudFormation スタックテンプレートを変更し、次にスタックリソースを更新または

置き換えます。機能のテンプレートとリソースは、最新の AWS GameKit 機能設定にリセットされます。

注: 機能設定を更新したが再デプロイしない場合、更新はローカルにのみ保存されます。AWS テンプレートやリソースには影響しません。

AWS GameKit は、AWS CloudFormation テンプレートのバージョンを (`*.yml` ファイルとして) ローカルのゲームプロジェクトファイルと共に保存します。プラグイン UI で作業する代わりに、これらのファイルを編集することもできます。AWS GameKit はこのファイルを使用してプラグイン UI の設定を入力します。テンプレートはゲームプロジェクトのディレクトリ: `...\Unreal Projects\<game project >\<AWS GameKit game title>\cloudformation\` に保存されます。

2. CloudFormation テンプレートまたは AWS リソースを直接更新します。CloudFormation を使用してスタックを変更する場合、テンプレートを更新し、そのテンプレートを使用してスタックリソースを更新します。これらの変更はいずれも AWS GameKit 機能設定には保存されません。後で AWS GameKit プラグインを使用してリソースを再デプロイすると、これらの変更は失われます。
3. AWS を使用して AWS のライブリソースを直接更新します (非推奨)。関連する AWS サービスのリソースを直接操作して、AWS リソースを表示したり変更したりすることもできます。例えば、Amazon Cognito コンソールで ID ユーザープールに直接アクセスできます。これらの変更はいずれも AWS AWS CloudFormation スタックテンプレートや AWS GameKit 機能設定には反映されません。これらの変更は、次にスタックが更新されたとき、または機能リソースが AWS GameKit プラグインに再デプロイされたときに失われます。

AWS GameKit リファレンス

このセクションには、AWS GameKit で使用する参考資料のコレクションが含まれています。

トピック

AWS のサービスとリソース

このリファレンスページには、AWS GameKit の各機能のバックエンドソリューションを構成する AWS のサービスとリソースが一覧表示されています。

コアサービス

AWS GameKit のコア機能には、以下のサービスとリソースが使用されます。

- AWS CloudFormation
- Amazon API Gateway
- AWS Identity and Access Management
- AWS Identity and Access Management (IAM)
- Amazon CloudWatch Logs
- Amazon CloudWatch (ダッシュボード用)
- Amazon Simple Storage Service (Amazon S3)
- Amazon Cognito (プレイヤー認証用)

ID と認証のサービス

ID と認証機能には、以下のサービスとリソースが使用されます。ID と認証ソリューションのアーキテクチャの詳細については、以下を参照してください。

- すべてのコアサービス。
- Amazon Cognito リソースには、サードパーティの ID プロバイダー用のユーザープールとアイデンティティプールが含まれます。

- Amazon DynamoDB リソースには、プレイヤーのログイン方法を追跡するためのテーブルが含まれます。
- AWS Key Management Service (AWS KMS).
- AWS Secrets Manager。サードパーティの ID プロバイダー用のアプリ認証情報を保存するために使用されます。

アチーブメントサービス

アチーブメント機能には、以下のサービスとリソースが使用されます。アチーブメントソリューションアーキテクチャの詳細については、以下を参照してください。

- すべてのコアサービス。
- Amazon DynamoDB リソースには 2 つのテーブルが含まれます。1 つはゲームのアチーブメント定義を保存するテーブルで、もう 1 つはプレイヤーのアチーブメントステータスを追跡するためのテーブルです。
- Amazon S3。リソースには、アチーブメントアイコンイメージファイルを保存する S3 バケットが含まれます。
- Amazon CloudFront リソースには、アイコンイメージをゲームクライアントに配信するための CloudFront ディストリビューションが含まれます。

ユーザーゲームプレイデータのサービス

ユーザーゲームプレイデータ機能には、以下のサービスとリソースが使用されます。ユーザーゲームプレイデータソリューションアーキテクチャの詳細については、以下を参照してください。

- すべてのコアサービス。
- Amazon DynamoDB リソースには 2 つのテーブルが含まれます。1 つはゲームのアチーブメント定義を保存するテーブルで、もう 1 つはプレイヤーのアチーブメントステータスを追跡するためのテーブルです。

ゲーム状態のクラウド保存サービス

ゲームの状態のクラウド保存機能には、以下のサービスとリソースが使用されます。ゲームの状態のクラウド保存ソリューションアーキテクチャの詳細については、以下を参照してください。

- すべてのコアサービス。

- Amazon Simple Storage Service (Amazon S3) AWS GameKit では、Amazon S3 バケットを使用してゲームのセーブファイルを保存します。
- Amazon DynamoDB AWS GameKit では、DynamoDB テーブルを使用してゲームのセーブファイルのメタデータを保存します。DynamoDB を使用してこのタイプのデータを保存することで、ゲームクライアントからの頻繁な読み取り/書き込みリクエストに対応します。
- AWS Lambda。AWS GameKit は、Lambda 関数を使用して、ゲームのセーブファイルやメタデータを保存し、同期状態を分析するタスクを管理します。

AWS GameKit がサポートされている AWS リージョン

このリファレンスページには、すべての AWS リージョンでの現在の AWS GameKit の可用性のステータスが一覧表示されています。リージョンでは、AWS GameKit ゲーム機能のバックエンドに必要なすべての AWS のサービスとリソースがサポートされている必要があります。

AWS GameKit プラグインでは、作業する環境を選択します。環境には AWS リージョンの選択が含まれ、これにより、その環境のすべてのゲームのバックエンドサービスとリソースがデプロイされる場所が決定されます。

AWS GameKit リソースをサポートしていない AWS リージョンにバックエンドをデプロイすると、[CreateStack Failed: Template format error: Unrecognized resource types] などのメッセージでデプロイが失敗します。

利用可能なリージョン

AWS GameKit 対応ゲームプロジェクトで使用する AWS リージョンを選択するには、次の表を使用します。

リージョン	機能: ID と認証	機能: アチーブメント	機能: ユーザーゲームプレイデータ	機能: ゲームの状態のクラウド保存
us-east-1: 米国東部 (バージニア北部)	使用可能	使用可能	使用可能	使用可能
us-east-2: 米国東部 (オハイオ)	使用可能	使用可能	使用可能	使用可能
us-west-1: 米国西部 (北カリフォルニア)	使用可能	使用可能	使用可能	使用可能

リージョン	機能: ID と認証	機能: アチーブメント	機能: ユーザーゲームプレイデータ	機能: ゲームの状態のクラウド保存
us-west-2: 米国西部 (オレゴン)	使用可能	使用可能	使用可能	使用可能
af-south-1: アフリカ (ケープタウン)	####			
ap-east-1: アジアパシフィック (香港)	####			
ap-south-1: アジアパシフィック (ムンバイ)	使用可能	使用可能	使用可能	使用可能
ap-northeast-3: アジアパシフィック (大阪)	####			
ap-northeast-2: アジアパシフィック (ソウル)	使用可能	使用可能	使用可能	使用可能
ap-southeast-1: アジアパシフィック (シンガポール)	使用可能	使用可能	使用可能	使用可能
ap-southeast-2: アジアパシフィック (シドニー)	使用可能	使用可能	使用可能	使用可能
ap-northeast-1: アジアパシフィック (東京)	使用可能	使用可能	使用可能	使用可能
ca-central-1: カナダ (中部)	使用可能	使用可能	使用可能	使用可能
eu-central-1: 欧州 (フランクフルト)	使用可能	使用可能	使用可能	使用可能
eu-west-1: 欧州 (アイルランド)	使用可能	使用可能	使用可能	使用可能
eu-west-2: 欧州 (ロンドン)	使用可能	使用可能	使用可能	使用可能

リージョン	機能: ID と認証	機能: アチーブメント	機能: ユーザーゲームプレイデータ	機能: ゲームの状態のクラウド保存
eu-south-1: 欧州 (ミラノ)	####			
eu-west-3: 欧州 (パリ)	使用可能	使用可能	使用可能	使用可能
eu-north-1: 欧州 (ストックホルム)	使用可能	使用可能	使用可能	使用可能
me-south-1: 中東 (バーレーン)**	使用可能 (有効になっている場合)	使用可能 (有効になっている場合)	使用可能 (有効になっている場合)	使用可能 (有効になっている場合)
sa-east-1: 南米 (サンパウロ)	使用可能	使用可能	使用可能	使用可能
中国 (北京および寧夏) リージョン	####			

**

この AWS リージョンは、AWS アカウントで自動的には有効になりません。ゲームのバックエンドをこのリージョンにデプロイするには、このリージョンを有効にするためのアクションを実行する必要があります。手順については、「Enabling an AWS Region」を参照してください。

AWS GameKit のデプロイ状態

このページでは、AWS GameKit プラグインを使用して作成された AWS ソリューションで使用されるデプロイ状態について説明します。

Unreal Editor で各ゲーム機能の AWS ソリューションの現在のステータスを確認できます。[プロジェクト設定] ウィンドウを開き、AWS GameKit プラグインページに移動します。アクティブな環境が選択され、有効な認証情報が入力されていることを確認します。

デプロイ状態には、以下の 2 種類があります。

- 定常状態とは、完了したデプロイアクションの最終状態を表します。デプロイアクションはもう進行中ではありません。
- 過渡的な状態では、現在進行中のデプロイアクションに関する詳細が提供されます。

> **Note**
>
> デプロイ状態では、AWS GameKit のローカルにキャッシュされた設定に関する情報は提供されません。ローカル設定が、デプロイされたリソースの設定と同期しているか、異なっているかは示されません。

定常状態

ステータス	説明
[アンデプロイ済み]	現在、ゲーム機能用の AWS リソースはデプロイされていません。このステータスは、リソースがまだデプロイされていないこと、または既存のリソースが正常に削除されたことを示している場合があります。
Deployed	現在、ゲーム機能用の使用可能な AWS リソースがデプロイされています。このステータスは、直近のデプロイまたは更新アクションが成功したことを示しています。
エラー	ゲーム機能用に AWS リソースがデプロイされましたが、リソースがエラー状態にあり、使用できません。このステータスは、直近のデプロイ、更新、または削除アクションが失敗し、システムが以前の使用可能な状態に完全にロールバックできなかったことを示しています。
[ロールバックが完了しました]	直近のデプロイ関連のアクションが失敗し、システムが以前の使用可能な状態に正常にロールバックされました。

過渡的な状態

これらの状態は、デプロイ関連のアクティビティが進行中であることを示しています (5 ~ 15 分以上かかる場合があります)。発生順に以下に示します。

- [テンプレートの生成中]
- [ダッシュボードのアップロード中]
- [レイヤーのアップロード中]
- [関数のアップロード中]
- [リソースのデプロイ中] または [リソースの削除中]
- [ステータスの取得中]
- [実行中] (この段階では AWS リソースは作成中です)

AWS GameKit の概念と用語

AWS GameKit に関連する用語と概念を以下に示します。ゲーム開発業界で一般的に使用される用語については、その用語が AWS GameKit でどのように使用されているかを以下の定義によって説明します。

AWS GameKit の用語

構成

AWS GameKit がクラウドベースのゲーム機能のバックエンドリソースをデプロイおよび追跡するために使用するマテリアルのコレクション。各機能バックエンドのソリューションアーキテクチャ、機能のセットアップ時に選択する構成オプション、ゲームプロジェクト固有のその他の詳細を定義するテンプレートが含まれています。複数の環境で AWS GameKit を使用してリソースをデプロイする場合、構成には環境ごとの個別の設定が含まれます。構成は、ゲームプロジェクトファイルと共に、ゲームタイトルにちなんだ名前のフォルダにローカルに保存されます。

環境

AWS GameKit プラグインで使用され、ゲームプロジェクトの構成設定とデプロイされた AWS リソースのコレクションを指定します。ゲームプロジェクトに指定できる構成は 1 つだけですが、それぞれの構成で、開発環境、テスト環境、本番環境など、複数の環境の設定とデプロイを追跡できます。AWS リソースのデプロイを含む、プラグイン内のすべてのアクティビティは、現在選択されている環境に基づいて実行されます。

ゲーム状態のクラウド保存

AWS クラウド上のバックエンドサービスを使用してゲームセーブファイルを保存する AWS GameKit ゲーム機能。この機能は、双方向のファイル同期を使用して、ローカルに保存されたゲームセーブファイルとクラウドに保存されたゲームセーブファイルが最新バージョンに更新されるようにします。ゲームの状態のクラウド保存により、プレイヤーは、セーブファイルを手動で追跡しなくても、複数のデバイスでゲームをプレイすることができます。

ゲームタイトル

ゲームプロジェクトの AWS GameKit 構成のユーザー定義の一意の識別子。ゲームタイトルの割り当ては、AWS GameKit プラグインを使用するゲームプロジェクトをセットアップする際に行う最初のステップの 1 つです。AWS GameKit を通じてデプロイするすべての AWS リソースの名前には、ゲームタイトルの文字列が含まれています。

プレイヤーデータ

ユーザー名、E メール、ソーシャルメディア ID など、個人を特定できる情報が含まれることがある、プレイヤー固有の情報。また、サブスクリプションデータ、ゲーム内購入、友達/ブロックリスト、デモグラフィックなどのアカウント情報が含まれる場合もあります。プレイヤーデータには個人識別情報が含まれているため、この種のデータの処理ではセキュリティが強化されています。

ステートフルアチーブメント

マルチステップとも呼ばれるこのタイプのアチーブメントでは、獲得するまでに複数のアクションが必要になります。ステートフルアチーブメントを定義するときは、アチーブメントを獲得するまでにかかるステップ数を指定します。AWS GameKit アチーブメントバックエンドは、アチーブメント獲得に向けたプレイヤーの継続的な進行状況を追跡し、アチーブメントを獲得するまでプレイヤーのステータスを徐々に更新していきます。

ステートレスアチーブメント

シングルステップとも呼ばれるこのタイプのアチーブメントは、1 回のアクションで獲得できます。ステートレスアチーブメントを定義するときは、ステップ数を「1」と指定します。AWS GameKit アチーブメントバックエンドは、プレイヤーのアチーブメントのステータスをロック済み (未獲得) またはロック解除 (獲得済み) として追跡します。

ユーザーゲームプレイデータ

AWS クラウド上のバックエンドサービスを使用してゲームプレイ中に生成されたデータを保存する AWS GameKit ゲーム機能。ゲームプレイデータの例としては、プレイヤーのスコア、インベントリ、プレイヤーの進行状況などがあります。また、ゲーム分析や、ゲーム内の意思決定や行動などのテレメトリも含まれる場合があります。ゲームプレイデータをクラウドに保存することで、プレイヤーはデバイスを切り替えたり、ゲームプレイを続行したり、データがプレイヤーに追従するようにしたりできます。ユーザーゲームプレイデータは、アカウント情報などのプレイヤーデータとは異なります。ユーザーゲームプレイデータには、個人識別情報は含まれません。

ゲーム開発用語

ゲームバックエンド

ゲーム機能はサポートするが、ゲームアプリケーションとは別のサービスおよびリソースを指します。通常、バックエンドサービスはリモートサーバー上で実行され、ゲームフロントエンド

は API を使用して通信します。クラウドベースのバックエンドは、仮想リソース上で実行されるサービスです。AWS GameKit は、その優れたパフォーマンス、セキュリティ、信頼性を活用して、AWS でクラウドベースのゲームバックエンドサービスを構築するためのツールを提供します。ゲームバックエンドサービスの一般的な用途には、永続的なデータストレージ、ID 検証、通知の他、リーダーボードやマッチメイキングなどのマルチプレイヤー機能のカスタムロジックなどがあります。

ゲームフロントエンド、ゲームクライアント

ビジュアルプレゼンテーションやユーザーインターフェイスなど、ユーザーが操作するゲームの部分を指します。多くのゲーム機能では、フロントエンドとバックエンドのコンポーネントが連携しています。例えば、ゲームセーブ機能では、フロントエンドは、プレイヤーがゲームの進行状況を保存できるメニューまたはメッセージとして存在する可能性があります。このプレイヤーアクションは、ゲームの状態をキャプチャし、「保存」API リクエストを適切なバックエンドサービスに送信して、成功または失敗を処理するようにフロントエンドに指示します。アチーブメントなどの他のゲーム機能は、ゲームプレイアクティビティを通じてフロントエンドでアクティブ化されます。

ゲームエンジン用語

以下のリンクを使用して、AWS GameKit でサポートされているゲームエンジンに関する用語やドキュメントにアクセスしてください。

- 「Unreal Engine 4 の用語」

ブループリント

Unreal Engine のビジュアルコーディングシステム。「ブループリントビジュアルスクリプティング」の詳細については、Unreal Engine のドキュメントを参照してください。

AWS 用語

リソーススタック

単一のユニットとして管理できる AWS リソースのコレクション。AWS GameKit は、ゲーム機能のバックエンドのすべての AWS リソースを単一の AWS CloudFormation スタックで管理します。プラグインを使用してゲーム機能のリソースをデプロイすると、AWS GameKit がスタックを作成します。AWS GameKit で作成された各スタックは、ゲームプロファイル名、環境、AWS

リージョン、およびゲーム機能に固有のものです。リソーススタックは、AWS CloudFormation の AWS マネジメントコンソールで表示できます。

リソーススタックテンプレート

スタックに作成される AWS リソースのコレクションをモデル化する構成設定。AWS CloudFormation テンプレートは、JSON または YAML でフォーマットされたテキストファイルです。AWS GameKit は、AWS CloudFormation サービスを使用し、テンプレートを使ってゲーム機能の AWS リソースを指定された AWS リージョンにデプロイします。テンプレートは、AWS CloudFormation の AWS マネジメントコンソールで表示できます。

ユーザープール

ゲームに登録していて、認証に使用できる固有のプレイヤー ID を持っているプレイヤーのディレクトリ。AWS GameKit で作成されたゲームの場合、ID と認証のバックエンドは、Amazon Cognito を使用してユーザープールを管理します。ユーザープールのプレイヤーは、E メールとパスワードを使用して直接ゲームにサインインするか、サードパーティを通じてゲームにサインインできます。

AWS GameKit リリース

リリースや既知の問題などの、AWS GameKit コンポーネントの包括的なバージョン情報については、「AWS GameKit リリース」を参照してください。