Guide for Unreal Engine Developers

AWS GameKit

AWS GameKit: Guide for Unreal Engine Developers

Table of Contents

You are currently viewing content for use with Unreal Engine software. See all AWS GameKit documentation

What is AWS GameKit?

Build AWS-powered cloud features from your game engine.

AWS GameKit is an open-source SDK for developers who want to build high quality, cloud-based features into their products. Cloud features bring some significant benefits to a game, including greater security, scalability, cost reductions, and player experience improvements such as greater flexibility in where and how people play.

Our goal with AWS GameKit is to give you the power of cloud features while removing the major challenges. We designed AWS GameKit for developers who don't have deep knowledge of AWS or cloud architecture design, but still want well-architected features that they can customize for their projects. With AWS GameKit, you get complete cloud architecture templates and the tools to build your infrastructure. Choose the features you want for your project, set up a cloud backend in two or three steps, and add feature functionality to your client apps. Work at your own pace to extend and customize the cloud backend for your customers.

AWS GameKit provides solutions for the following game-related features.

- **Identity and authentication –** Protect players with secure registration and robust identity management for your game. Verify player logins to manage access to player sessions, and use authentication for AWS GameKit cloud features.
- **Achievements –** Create goals that players can achieve to earn recognition, win rewards, or initiate game events. Manage players' achievements in the cloud and track their progress toward long-term goals.
- **Game state cloud saving –** Synchronize game saves in the cloud so that players can resume their play from different locations and devices, or recover game progress as needed.
- **User gameplay data –** Maintain gameplay data for each player, such as inventory, statistics, and cross-play persistence, and make it available to players wherever and whenever they log in to the game.

AWS GameKit benefits

Developers and architects tasked with building cloud-based backend infrastructure for their game projects can take advantage of these benefits:

- **Build and maintain a cloud backend from your game engine.** Use AWS GameKit for Unreal Engine with its streamlined workflows to create and manage an AWS cloud backend for your game project. Use tools to incorporate feature functionality. Set up backend infrastructure in multiple environments and maintain each independently.
- **Start with expertly designed cloud architectures.** The AWS solution for each AWS GameKit cloud feature is designed by cloud architecture experts and based on the AWS Well-Architected Framework for secure, high-performing, resilient, and efficient solutions. The solutions incorporate game development best practices and customer feedback.
- **Learn as you go.** AWS GameKit provides customizable solution templates and APIs for your backend. This means you get started fast with a production-ready backend, and then . From this point, you have wide flexibility to modify the templates, experiment with alternative AWS features and services, and build a custom cloud infrastructure for your projects.
- **Integrate feature design with rapid prototyping.** With a cloud backend in place, use pre-configured UI components, example code, and example games to integrate the features using iterative design and development.
- **Customize the AWS GameKit SDK and tools.** AWS GameKit SDK components are available on a "source- available basis, so you can modify or build tools to fit your development process. Customize the existing AWS GameKit for Unreal Engine plugin plugin or create versions for other game engines. Modify or extend the core C++ API functionality.

Get started with AWS GameKit

New to AWS GameKit? We recommend that you start here:

- **Set up AWS GameKit** for your projects
  - Download AWS GameKit for Unreal Engine
  - Install the AWS GameKit
  - Set up an AWS account and users for AWS GameKit
- **Learn about the AWS GameKit UI**
  - Explore AWS GameKit in the Unreal Editor
- **Keep up with the latest news and releases**
  - AWS GameKit forum. Share questions and comments with the developer community.
  - AWS Game Tech blog. Learn about new features and get developer tips for all AWS Game Tech offerings.

- AWS GameKit releases. Monitor version updates and known issues.

Related services

AWS GameKit is a solid choice for building fully customizable AWS Cloud-based features into your games while retaining the ability to customize those features going forward. Also consider these game-related AWS offerings:

- **Amazon GameSparks –** Amazon GameSparks is a fully managed AWS service that provides a multi-service backend for game developers.
- **Amazon GameLift –** GameLift provides solutions for hosting session-based multiplayer game servers in the cloud, including a fully managed service for deploying, operating, and scaling game servers.
- **Open 3D Engine (O3DE) –** O3DE is an open-source 3D development engine for game, simulation, and multimedia creators. It is modular and cross-platform.
- **Amazon Nimble Studio –** Amazon Nimble Studio is a virtual studio that empowers visual effects, animation, and interactive content teams to create content securely within a scalable, private cloud service.

Related AWS topics

AWS Well-Architected

AWS GameKit solutions are based on AWS Well-Architected and its six-pillared framework: operational excellence, security, reliability, performance efficiency, cost optimization, and sustainability. To learn more about how well-architected AWS solutions are built, use the resources on this site, which include best practices, design principles, and industry-specific whitepapers.

AWS for Games

Learn more about other AWS services and solutions for game development, including Amazon GameLift for multiplayer services and specialized solutions for game analytics and AI.

How AWS GameKit works

Summary

This topic is a high-level technical overview of AWS GameKit for game developers who want to know how to use it in their Unreal Engine projects. This overview describes the core AWS GameKit components and how developers interact with them to build and manage backend game features.

How AWS GameKit works with your game

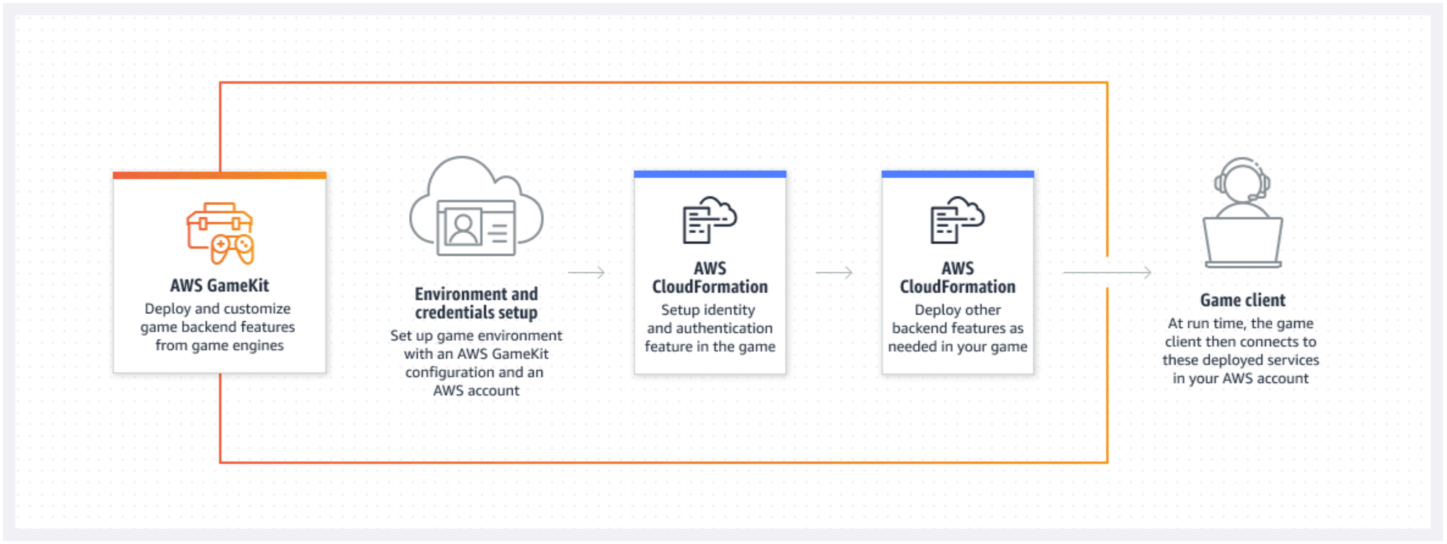
AWS GameKit provides tools to simplifies the work of building cloud-based game features into your project. AWS GameKit helps you configure and deploy a well-designed game backend on AWS Cloud and provides APIs to connect it to your game frontend.

As shown in the following diagram, use AWS GameKit with your game engine and development environment. AWS GameKit adds functionality to your game engine to build and maintain a cloud backend.

With AWS GameKit, deploy ready-to use backend solutions for cloud-based features that you want to add to your game. These solutions contain all the architecture design and configuration settings needed to run the backend and communicate with a game client. Choose a game feature, set some optional customizations, and initiate deployment. In 10 to 30 minutes, your feature backend is up and running and ready to communicate with your game client.

When you deploy the game backend using AWS GameKit, the game project is automatically configured with endpoints for the game backend. You add feature elements into your game client and use AWS GameKit API calls to connect to the game backend. You can play test your game in the engine, making live calls to the backend. When you build and package your game for distribution, the game is automatically connected to the game's backend resources on AWS.

When someone plays your game with AWS GameKit features, they register and log into the game before playing. From this point until the player logs out, all game client calls to the AWS GameKit API are made on behalf of the logged-in player. For example, when a game client sends an API request to update the player's cloud-stored gameplay data, the request automatically references the player's registered ID.


AWS GameKit
Deploy and customize game backend features from game engines
Environment and credentials setup
Set up game environment with an AWS GameKit configuration and an AWS account
AWS CloudFormation
Setup identity and authentication feature in the game
AWS CloudFormation
Deploy other backend features as needed in your game
Game client
At run time, the game client then connects to these deployed services in your AWS account

AWS GameKit components

AWS GameKit is an open-source solution for game developers. It includes the following components:

- **AWS GameKit plugin** – Use the AWS GameKit version for your game engine to configure, deploy, and manage game backend services for each game feature. With the backend in place, you can connect it to feature functionality in your game frontend using the AWS GameKit API. You can manage the backend services from your game engine, or you can use AWS tools to monitor and modify them. For information on game engine support, see AWS GameKit components.
- **AWS backend service architecture solutions** – Ready-to-build solutions for each game feature use AWS CloudFormation templates to deploy AWS resources and connect them with AWS Lambda-driven logic. When you configure and deploy a feature with AWS GameKit, it uses these templates to build the backend for your game.

  Each game feature solution handles these tasks and others:
  - Provisions resources to run the backend services on AWS infrastructure.
  - Manages deployment and update activity for the backend services.
  - Handles API communication, including authentication and authorization, security, traffic management, and monitoring.
  - Provides Lambda functions that drive activity for the game feature, including data management and game logic.

  - Monitors game feature activity, generates logs, and enables metrics tracking. Set up dashboards through Amazon CloudWatch to track operational metrics for the game feature's backend.
- **AWS GameKit API –** The API offers feature-specific operations that connect your game frontend to the deployed backend services. The AWS GameKit API wraps core AWS service functionality into API requests for game-specific workflows. For example, the `UpdateAchievement()` operation includes direct calls to AWS services to authenticate a player request, run logic on the achievement's requirements, and update a player's stored achievement status.

  Games with AWS GameKit features are automatically configured with the endpoint for the game backend. In addition, all API calls reference the unique player ID for a logged-in player.
- **Example code and assets –** Examples provide a starting point to help you integrate game feature functionality into your game. Each game feature includes a set of example materials for illustration or to support rapid prototyping.
- **AWS GameKit plugin source code –** You can download this source code from GitHub on a "source available" basis. Use the source code to customize or to create a new version for game engines that aren't yet supported.

Use AWS GameKit to set up self-managed AWS services for your game backend, which you own and manage through your AWS account. With AWS GameKit you can put in place a basic, well-built backend architecture for your game. You have full flexibility to use AWS tools to customize and extend the game backend as your game evolves.

AWS GameKit characteristics

AWS GameKit supports these approaches to game development.

Work entirely in the game engine

With AWS GameKit, you can do virtually all work on cloud-based game features from your game engine.

To set up a game backend, use AWS GameKit to configure, deploy, and manage AWS resources for each game feature. You can also track ongoing status and access operational metrics for the backend services in your game engine.

To connect a game client to the backend, use the AWS GameKit API to make requests to the backend services. You can access the API and run samples in the game engine. After setting up the backend for a game feature, you can test the game in the game engine and debug feature issues using log messaging.

Work in multiple environments

Game development teams can use environments to manage different stages of project development concurrently. By switching the AWS GameKit environment when working in the game engine, teams can configure and deploy separate stacks of AWS resources for each environment. The game project is automatically configured to connect to the backend endpoint for whichever environment is currently active.

Teams can use pre-defined environments for development, QA, and production. They also have the option to create custom environments.

Switching environments has the following effects:

- The Settings UI for each game feature updated to reflect the configuration options and deployment status for the AWS resources deployed in the active environment.
- For achievement definitions, cloud sync pulls from the data store for the active environment.
- Dashboard links point to the feature dashboards for the active environment.
- UI elements to create, redeploy, or delete AWS resources impact resources in the active environment only.
- The AWS GameKit for the game project configuration is updated with the backend endpoints for the active environment. The game project automatically connects to the environment's game backend when playing a game in the game engine and when building or packaging the game.

When you deploy AWS resources for a game feature, the resource names reference the AWS GameKit environment in use. This naming convention helps you use AWS tools to track resources and costs based on the environment.

Teams can optionally manage AWS user access for each environment. For example, a team might restrict write access for resources in a production environment.

Position backend services geographically

When deploying AWS resources your game backend, you choose where to physically deploy them. You can choose from available AWS Regions, each of which represents a geographical location of computing hardware. For a complete list of AWS Regions that support AWS GameKit, see AWS GameKit supported AWS Regions.

Each AWS GameKit environment has an AWS Region, and all resources created for that environment are deployed to that Region. To set up backend services in more than one Region, you can create multiple environments.

When choosing a region, consider the following:

- You can decrease the effect of latency issues by deploying services geographically near your player base.
- Although outages and slowdowns are rare, you can further minimize the possibility by using regions with plenty of resources and lower usage.
- Regions vary in cost and in the AWS services that they support.
- Changing an environment's region setting requires that you delete and replace all AWS resources deployed in the environment.

Develop in teams

Development teams with multiple members can use AWS GameKit with their version control systems. AWS GameKit stores its files in the game project folder to simplify tracking and sharing.

AWS GameKit users on the team must have AWS account access with permissions for AWS GameKit resources. Administrators can manage AWS users for team members and set up user groups to manage appropriate permission levels.

Development workflow with AWS GameKit

Summary

This topic outlines the steps that game developers can expect to work on when using AWS GameKit to add a cloud-based feature to a game project.

The following steps describe the typical integration process for adding an AWS GameKit game feature. You complete these steps by working in your game engine.

1. Developer (or AWS account administrator) creates an AWS account or designates an existing account to manage the game backend on AWS. Team members who work with the AWS GameKit plugin need an AWS user ID with AWS GameKit access permissions and security credentials.

2. Developer installs the AWS GameKit plugin for their game project.
3. With the game project open in the game engine, the developer creates an AWS GameKit configuration for the project. In this step, the developer creates a game project alias and chooses an environment to work in. They submit their AWS user credentials, which links the game project to an AWS account. In response, AWS GameKit generates default configuration files and adds them to the game project folder.
4. Developer configures and deploys a backend for the identity and authentication game feature.
   a. Developer sets configuration options for the game feature.
   b. Developer chooses **Create** to deploy AWS resources as configured for the identity and authentication backend. AWS GameKit also deploys some core resources for use with all features. The resource stack for identity and authentication includes an Amazon Cognito user pool for player registrations. The developer tracks the progress of AWS resource creation in the plugin UI and in the game engine output log.

      Note

      After resource deployment, the AWS account can begin incurring usage charges. For more information, see AWS pricing with AWS GameKit.
   c. As necessary, the developer can edit the feature's configuration settings and redeploy the backend, or delete it and start over.
5. Developer adds game feature functionality to frontend game code. Feature integration work varies but generally involves creating new UI elements, adding game logic, and making calls to the game backend using the AWS GameKit API. For identity and authentication, the developer creates UI workflows so that users can register a player ID, log in, and log out of the game. It might also include a password reset option.

   The developer can use these plugin features to help with integration:
   - Work with example assets that come with the plugin. These include example UI elements, code snippets showing the API calls, and complete working game samples.
   - Test the identity and authentication feature in the game engine. A game running in the editor is automatically configured to connect to the game backend. If a feature has a deployed game backend, the game can send API calls to the backend and receive responses.
6. With the identity game feature in place, the developer integrates additional AWS GameKit features. This work follows a similar process as with identity and authentication: create a game feature backend, add feature functionality to the frontend, and test the connection.

7. Developer packages the game to include AWS GameKit components.
8. As game development progresses, the developer can use the plugin to create new environments for QA, production, and other stages. In each environment, the developer creates a separate stack of AWS resources for the game backend. In this way the developer can manage feature development and backend configuration independently in the different environments.

AWS pricing with AWS GameKit

Summary

This topic helps game developers understand how their AWS account incurs charges when creating a game backend with AWS GameKit. This topic discusses tools that AWS provides to help customers manage their costs.

AWS offers AWS GameKit tools without charge. When you build a game backend with AWS GameKit, you pay only for the AWS products that you create and use to run the game backend. Standard AWS service pricing applies as published.

During game development, you may be able to take advantage of the AWS Free Tier. AWS customers can use Free Tier benefits to get real-world experience with a wide range of AWS products at no cost. All AWS resources that you create with AWS GameKit are available as part of the AWS Free Tier. Free Tier benefits might be time-limited or usage-limited. For more information about the AWS Free Tier, see the following:

- Free Tier benefits by service
- Get started with the AWS Free Tier (AWS Whitepaper)

As an AWS account owner, you have full control over spending by adding, modifying or removing, AWS resources. You can also use various AWS tools to help manage your costs. Your AWS account doesn't incur charges until you deploy AWS resources. The AWS GameKit plugin notifies users when charges might start to incur. You can delete deployed AWS resources to stop incurring charges.

Estimating AWS costs

With AWS GameKit, your costs depend on the game backend you build and how your game uses it. The total cost of running the backend equals the sum of the costs for each AWS service in the backend.

The cost basis for each AWS service varies. Some services charge a time-based fee, while others charge based on usage. For example, Amazon Cognito charges by monthly active users, while AWS Lambda charges based on the number of requests and compute time used.

There are two sets of AWS resources in a game backend:

- Core resources – These services help manage the backend infrastructure. AWS GameKit deploys core resources when you first create a backend for the identity and authentication feature, which is always the first game feature created.
- Feature resources – These services make up the solution architecture for each AWS GameKit feature. For more details on feature-specific resources and cost information, see the guide sections for each game feature.

AWS GameKit core services include:

- **Amazon API Gateway** – AWS GameKit uses this service to manage API calls to a game's backend services. API Gateway manages traffic, authorization & access control, throttling, and monitoring. The API Gateway pricing model relies on the volume of API calls. The 12-month Free Tier benefit includes this service. For more information, see Amazon API Gateway pricing.
- **AWS Lambda** – AWS GameKit uses this service to run custom code that manages activities such as resource updates, deployments, and compute services. The Lambda pricing model relies on the number of requests and the compute time required. The 12-month Free Tier benefit includes this service. For more information, see AWS Lambda Pricing.
- **Amazon Simple Storage Service (Amazon S3)** – AWS GameKit uses this service when creating and updating AWS resources. The Amazon S3 pricing model relies on the amount of storage space used and the number of put and get requests. The 12-month Free Tier benefit includes this service. For more information, see Amazon S3 pricing.
- **AWS CloudFormation** – AWS GameKit uses this service to model and provision cloud infrastructure resources for the game backend. This service is free up to the standard Free Tier limits. For more information, see AWS CloudFormation Pricing.
- **AWS Identity and Access Management (IAM)** – AWS GameKit uses this service to control access to AWS resources for a game. There is no charge for using IAM.
- **Amazon CloudWatch** – AWS GameKit uses this service to maintain a dashboard with operational metrics monitoring tools for each game feature's backend. The CloudWatch pricing model relies on usage and a monthly dashboard fee. The 12-month Free Tier benefit includes this service. For more information, see Amazon CloudWatch pricing.

Note

The CloudWatch Free Tier allows up to three dashboards per AWS account. If you create a backend for every available AWS GameKit feature and activate a dashboard for each, your AWS account will exceed the Free Tier benefits.

- **Amazon Cognito** – AWS GameKit uses this service to manage player identities and authentication credentials. The Amazon Cognito pricing model relies on the monthly active users (MAUs) in a user (player) pool. The 12-month Free Tier benefit includes this service. For more information, see Amazon Cognito Pricing.

Managing AWS costs

To track your usage and costs, use the AWS Billing and Cost Management dashboard in the AWS Management Console. All AWS resources deployed through AWS GameKit use a naming convention to help you track resources and costs. Resource names start with the string `gamekit_` and include the following elements:

- Environment code (such as "dev")
- AWS Region code (such as "us-west-2")
- AWS GameKit game title/alias
- Game feature name, if relevant (such as "UserGameplayData")

To help avoid unexpected or excessive costs, as a best practice use AWS Budgets to set budget limits with alerts. For more information about managing costs, see the 10-minute tutorial Control your AWS costs.

AWS GameKit components

AWS GameKit contains the following components. You can see detailed version information in AWS GameKit Releases.

AWS GameKit for Unity software

When added to your Unity game project, the AWS GameKit package adds UI and functionality for building a cloud backend for the project directly from the Unity Editor. The package download includes the AWS GameKit C# API for Unity and C# sample code. Developers can customize the package source code, offered as "source available" under an Apache 2.0 license.

Download AWS GameKit for Unity or get source code from GitHub

AWS GameKit plugin for Unreal Engine

This plugin adds AWS GameKit functionality and assets to your Unreal Editor. With the plugin, you can build a game backend on AWS Cloud and add cloud-based features to your Unreal game projects. The plugin download includes the AWS GameKit C++ API for Unreal, C++ sample code, Unreal blueprints, and example game elements. Developers can access the plugin source code, offered as "source available" under an Apache 2.0 license, to customize the plugin.

Download the latest AWS GameKit plugin for Unreal Engine

Get source code for the AWS GameKit plugin for Unreal Engine on GitHub

AWS GameKit Core C++ SDK

The SDK contains the core AWS GameKit game feature functionality and forms the basis for engine-specific APIs. Developers can access the SDK source code, offered as "source available" under an Apache 2.0 license, to build custom APIs and plugins for any game engine.

Get AWS GameKit C++ SDK source on GitHub

Additional resources

- AWS GameKit documentation
- AWS GameKit releases, including links to current and previous versions, release notes, known issues
- AWS GameKit forum
- AWS Game Tech blog

Setting up for AWS GameKit

If you're ready to get hands on AWS GameKit, start with these set-up tasks. When done, you have AWS GameKit installed with your game engine and you can start building a cloud backend for your game project.

- Get and install the AWS GameKit plugin.
- Set up an AWS account.
- Create an AWS account user, with AWS GameKit permissions and security credentials for use with the plugin.

Note

You might have to use the AWS Management Console for some set-up tasks. Once you've completed set-up, you can work on your game backend entirely from your game engine if desired.

Topics

Install the AWS GameKit plugin with Unreal Engine

Summary

Download and install AWS GameKit for use with your Unreal Engine game project. This topic gives game developers step-by-step instructions for setting up the AWS GameKit plugin in the Unreal Editor.

Plugin requirements

To use the AWS GameKit plugin as provided:

- Unreal Engine version compatible with the plugin version (see AWS GameKit version information).
- A C++ Unreal game project. Blueprint-only projects, which have no source code, aren't compatible with the plugin.

The AWS GameKit plugin source code is available for customization. To modify the code and generate a custom plugin, you need a code editor to work with C++ game projects. For example, for Visual Studio you need these tools:

- Visual Studio 2019 with the following tools installed:
  - On the **Workloads** tab:
    - Desktop Development with C++
    - Game Development with C++, with these options:
      - C++ profiling tools
      - C++ AddressSanitizer (optional)
      - Windows 10 SDK (10.0.18362 or newer)
      - Unreal Engine installer
  - On the **Individual components** tab: .NET Framework 4.8 SDK

What's in the AWS GameKit download

The download includes the following:

- Plugin binaries for Unreal Engine.
- AWS GameKit C++ libraries with functionality for each game feature.
- Blueprint code and UI samples for each game feature.
- C++ example code with API calls for each game feature.
- Automated scripts for setting up AWS users with permissions for AWS GameKit and security credentials.
- Default configuration files, which AWS GameKit uses to create your game backend.

Install the plugin

Download the AWS GameKit package and install the plugin for a C++ Unreal game project.

To install the plugin:

1. **Get AWS GameKit for your game engine.** Download the `.zip` file from the AWS GameKit for Unreal Engine GitHub repo: aws/aws-gamekit-unreal.
2. **Unpack the `.zip` file.**
   a. Find the directory path for the C++ Unreal game project that you want to use AWS GameKit with. Open the directory folder `Plugins/Marketplace`. For example: `... > Unreal Projects > MagicChickenGame >Plugins > Marketplace`. If this folder doesn't exist, create it.
   b. Extract the AWS GameKit plugin zip file contents and place the files into the game project folder. The extracted files are organized in a folder called `AwsGameKit`, with the AWS GameKit plugin descriptor file, `AwsGameKit.uplugin`, at the root. Unreal Engine recognizes this file as a plugin.

   > **Note**
   >
   > To install the plugin for use with any Unreal project, place the files in the directory path for your Unreal Engine installation, in the folder `Plugins/Marketplace`. Don't try to install the plugin in both locations, as this results in errors.
3. **Rebuild the project with AWS GameKit.**
   a. Go to the game project's root folder and look for a solution (`*.sln`) file. If none exists, find the `.uproject` file and generate project files.
   b. Open the solution file and build or rebuild the project.
4. **Enable the plugin for the game project.**
   a. Open the game project in the Unreal Editor. In the main menu, open **Edit, Plugins** and search for the AWS GameKit plugin.
   b. Select the **Enabled** box to turn on the plugin for the game project. This action generates a prompt to restart the Editor. If you get an Editor prompt "Project is out of date. Would you like to update it?" choose **Update**.
   c. Restart the Editor with the game project. If you get an error message that prompts you to build or rebuild your project, repeat step 3.

5. **Verify that the plugin is installed.** Look in the Content Browser for the AWS GameKit content. If you don't see the content, make sure that your **View Options** setting has the **Show Plugin Content**option selected.
6. **Update your project's `.Build.cs` file.**

   a. Locate the *`[project name]`*`.Build.cs` file and open it in your IDE. For example: `...Unreal Projects\MagicChickenGame\MagicChickenGame.Build.cs`.

   b. Add the following strings to `PublicDependencyModuleNames`: "AwsGameKitCore" and "AwsGameKitRuntime".

      For example:

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

Set up AWS account for AWS GameKit

Summary

This topic provides instructions for how to set up an AWS account and users for AWS GameKit activities. This information is for administrators and others who manage AWS user accounts.

As a first step to building cloud-based features into your game projects with AWS GameKit, create an AWS account for use with the project. You use this account to manage cloud resources for your game backend, including tracking costs and controlling user access.

To set up an AWS account for a game project:

- Get an AWS account. You can use an existing account or create a new one.
- Set up AWS users on the account. Extend access to use AWS GameKit and generate security credentials for each user.

Note

If your account uses the new AWS IAM Identity Center to manage users, users might experience unusual behavior and messaging in the AWS GameKit plugin . Their security credentials provide short-term access, users can't store them for future use and have to regenerate them often. If you create users with the automated script supplied with AWS GameKit, users get long-term access keys.

Sign up for an AWS account

If you don't have an AWS account, or if you want to set up a separate account for a project, complete the following steps to create one.

To sign up for an AWS account

1. Open https://portal.aws.amazon.com/billing/signup
2. Follow the online instructions. Part of the sign-up procedure involves receiving a phone call and entering a verification code on the phone keypad.

   When you sign up for an AWS account, an *AWS account root user* is created. The root user has access to all AWS services and resources in the account. As a security best practice, assign administrative access to an administrative user, and only use the root user to perform tasks that require root user access.

AWS sends you a confirmation email after the sign-up process is complete. At any time, you can view your current account activity and manage your account by going to https://aws.amazon.com/ and choosing **My Account**.

Get more detailed guidance and tips on the sign-up process in How do I create and activate a new AWS account?.

Set up an administrator

After you sign up for an AWS account, create an administrative user so that you don't use the root user for everyday tasks.

Secure your AWS account root user

1. Sign in to the AWS Management Console as the account owner by choosing **Root user** and entering your AWS account email address. On the next page, enter your password.

For help signing in using root user, see Signing in as the root user in the *AWS Sign-In User Guide*.

2. Turn on multi-factor authentication (MFA) for your root user.

   For instructions, see Enable a virtual MFA device for your AWS account root user (console) in the *IAM User Guide*.

Create an administrative user

For your daily administrative tasks, assign access to an administrative user in the AWS Identity and Access Management (IAM) service.

1. Sign in to the AWS Management Console with your root user credentials (AWS account email address and password.
2. On the **Console Home** page, select the IAM service.
3. In the navigation pane, select **Users** and then select **Add users**.
4. In **Step 1: Specify user details**, set the following:

   - Enter a **User name** for the new user. This is their sign-in name for AWS.
   - Select **Provide user access to the – AWS Management Console *optional*** This produces AWS Management Console sign-in credentials for the new user. Choose the option **I want to create an IAM user**.
   - Choose a **Console password** option.

   Choose **Next**.
5. In **Step 2: Set permissions**, choose the option **Attach policies directly** and select the **AdministratorAccess** policy from the list. Choose **Next**.
6. In **Step 3: Review and create**, check your settings and then choose **Create user**.
7. In **Step 4: Retrieve password**, the Console displays information on signing in with the new user. At this point, the new administrative user has sign-in credentials for the console but doesn't yet have the security credentials needed to use the AWS GameKit plugin . Choose the **View user** button.
8. On the user detail page, open the **Security Credentials** tab and go to the **Access Keys** section. Choose **Create access key**.

9. Under **Access key best practices & alternatives**, choose the **Local code** option, acknowledge the recommendations, and choose **Next** to create the access key. Follow the instructions to download and store the new access key. You need this key to use AWS GameKit.

Set up a user with AWS GameKit access

All AWS GameKit users must have AWS access before they can deploy, update, or delete AWS resources for their game project. An AWS account administrator creates users, manages user access, and generates unique security credentials for each user.

AWS GameKit provides an automated script for setting up AWS users with AWS GameKit access. Administrators can use the script to create new users or extent access to existing users. This script creates IAM users with long-term access keys. As an alternative, administrators can work directly in the AWS Identity and Access Management (IAM) service. For guidance on working with IAM, see Related AWS topics.

To create or update AWS users using the AWS GameKit script

1. Find the Python script `create_IAM_user.py` in your AWS GameKit install files.

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

   To use this script, you need administrative rights to the AWS account that you are adding or updating users for. For additional requirements, see the `requirements.txt` file located in the same directory.

2. Run the script with the following arguments to create or update a user:

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

   - `AWS USERNAME` is the user name that you want to add or update.
   - `AWS ACCESS KEY` and `AWS SECRET KEY` are your administrator credentials for the AWS account.

   On a successful request, the script takes the following actions:

- Checks to see if the requested user name already exists in the AWS account. If it doesn't exist, the script creates a new IAM user in the account.
- Checks for a user group in the AWS account with the name "GameKitDevGroup". If none exists, the script creates a new "GameKitDevGroup" user group and attaches the `GameKitDeveloperPolicy` permissions policy. This policy is also included in the AWS GameKit download package (`GameKitDeveloperPolicy_Template.json`).
- Adds the requested user to the `GameKitDevGroup` user group.
- For a newly created user, generates long-term security credentials for the user and saves them to `[username]_credentials.txt` in the `...\policies` directory.

Related AWS topics

- How do I create and activate a new AWS account?, *AWS Knowledge Center*
- Organizing Your AWS Environment Using Multiple Accounts, *AWS Whitepapers*
- Best Practices for Managing AWS Access Keys, *AWS General Reference*
- Changing permissions for an IAM user, *IAM User Guide*
- Where are configuration settings/credential information stored?, *AWS Command Line Interface User Guide*

Use IAM to set up AWS GameKit user access

You can use an automated script to set up AWS users with access to AWS GameKit. As an alternative, use the following instructions to complete these tasks directly in the AWS Management Console:

- Create a user permission policy with AWS GameKit access.
- Attach the permission policy to a new or existing user group
- Create an IAM user and add to the user group.

To create a permissions policy for AWS GameKit:

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.

2. In the IAM console, open the **Access Management > Policies** page to create or update a AWS GameKit permissions policy for your AWS account. Choose the **Create Policy** button.
3. Choose the **JSON** tab to open the editor. Replace existing content with permissions syntax for AWS GameKit users. To generate the replacement policy syntax:

   a. A policy template for AWS GameKit, is included in the AWS GameKit download package at: `...AwsGameKit\Resources\cloudResources\policies\GameKitDeveloperPolicy_Template.json`. Be sure to use the policy template for the AWS GameKit plugin version you're using.
   b. Customize the template syntax with your AWS account. Replace the strings `<YOUR_ACCOUNT_ID>` with your AWS account ID (9-digit account number). Look for your account ID at the top of the AWS Console under your account user name.

      You can replace these strings manually or use the provided python script, `generate_policy_instance.py`, to create a new JSON file with the customized syntax. This script is included in the AWS GameKit download package in the same directory as the policy template. Call the script with your AWS account ID:

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

      The script saves the new policy file in the same directory. Look for a file name with your AWS account ID, such as `GameKitDeveloperPolicy-123456789`.
4. In the IAM console, after you've entered the new policy syntax, choose **Next** until you reach the **Review policy** page. Choose the **Create policy** button.

To add the permissions policy to a user group:

1. In the navigation pane, select **User groups**. Open the page for an existing user group, or choose **Add group** to create a new one.
2. If you're creating a new group, enter a group name.
3. In **Attach permissions policies**, select the customer-managed policy "GameKitDeveloperPolicy".
4. Complete the workflow to create or update the user group.

To set up a user with AWS GameKit access

1. In the navigation pane, select **Users** and then select **Add users**.
2. In **Step 1: Specify user details** set the following:

   - Enter a **User name** for the new user. This is their sign-in name for AWS.
   - Select **Provide user access to the – AWS Management Console *optional*** This produces AWS Management Console sign-in credentials for the new user. Choose the option **I want to create an IAM user**.
   - Choose a **Console password** option.

   Choose **Next**.
3. In **Step 2: Set permissions**, choose the option **Add user to group**. Select a user group with AWS GameKit permissions from the list. Choose **Next**.
4. In **Step 3: Review and create**, check your settings and then choose **Create user**.
5. In **Step 4: Retrieve password**, the Console displays information on signing in with the new user. At this point, the new user has sign-in credentials for the console but doesn't yet have security credentials, which they to use the AWS GameKit plugin . Choose the **View user** button.
6. On the user detail page, open the **Security Credentials** tab and go to the **Access Keys** section. Choose **Create access key**.
7. Under **Access key best practices & alternatives**, choose the **Local code** option, acknowledge the recommendations, and choose **Next** to create the access key. Follow the instructions to download and store the new access key. You need this key to use AWS GameKit.

Manage permissions for achievements

When working with the achievements game feature, users might need additional access permissions to work with achievement definitions. The default `GameKitDeveloperPolicy` permissions policy allows users to sync achievement definitions to the cloud when working in the Development environment only.

Working with achievement definitions involves direct calls to the `AwsGameKitAchievementAdmin` API. AWS GameKit manages `AchievementAdmin` permissions with IAM roles, which offer additional controls and security to protect your game. An IAM role specifies two things: (1) who can assume the role, and (2) which resources they can control.

You need AWS account admin access to change user permissions. As a best practice, assign permissions to user groups and manage user permissions by adding users to user groups with the appropriate permissions.

Options for editing `AchievementAdmin` permissions:

To remove user access in the Development environment

Remove the following section from the user group permissions policy.

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

To add user access in other environments

Follow these steps:

1. In `_AchievementsAdminInvokeRole`, edit the role's trust relationship to add specific user group IDs. For detailed instructions, see [Modifying a role trust policy](#).
2. Create an IAM user group with permissions to assume this role.

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

3. To give IAM users access, add them to the new user group.

Tips for AWS account administrators

If you manage an AWS account for an AWS GameKit project with multiple team members, consider these best practices.

- When setting up users for AWS GameKit, consider enabling both programmatic and console access for users. People might want to use the AWS Management Console to view AWS resources, troubleshoot issues, and other reasons.
- Use IAM user groups to manage permissions levels for team members. With user groups, you set the permissions policies for the group instead of for individual users. You can create separate user groups with different levels of access, and add individual users to groups with the appropriate permissions.
- The default AWS GameKit permissions policy template includes comprehensive access to all AWS services and resources that AWS GameKit solutions use. You can choose to adjust user access by creating and applying custom policies. For example, you might change a user's access to view but not deploy or delete AWS resources for a game feature. Keep in mind that AWS GameKit solutions involve complex collections of AWS services. It's not always apparent how changes to access permissions might affect a user's ability to work with AWS GameKit features.
- Consider managing access levels for each environment stage. AWS GameKit users must provide account credentials for each environment they work in. For example, you might choose to allow all actions in the Development stage but restrict access in Production.

Get your AWS security credentials

Summary

AWS GameKit users must have an AWS account and security credentials to use the plugin. This topic helps plugin users or AWS account administrators get credentials for an AWS user.

AWS GameKit users must sign in to the plugin with their AWS user security credentials. These credentials authorize a user's programmatic access to AWS so that they can create and manage their game's cloud backend directly from the plugin.

Use the following procedures to get security credentials for an existing AWS user. To create new users with AWS GameKit access, see Set up AWS account for AWS GameKit.

Retrieve security credentials

AWS users must store security credentials locally. Look for your existing security credentials in the following locations:

- For AWS users created with the `create_IAM_user.py` script (included in the AWS GameKit plugin download), the script generates security credentials for the user and saves them to [username]_credentials.txt file. If you ran the script, the file is saved to your local machine in the directory ...\policies\.

- When generating security credentials through the AWS Management Console, you can download the credentials to a local file named *<user name>*_accessKeys.csv.
- If you've used your credentials with the AWS GameKit plugin or other AWS programmatic tools, they may be saved to your home directory (for example: C:\Users\**<user ID>**\.aws \credentials).

Generate new security credentials

If you don’t have valid security credentials, or you've lost your existing credentials, follow these instructions to create new ones for your AWS user.

Note

For AWS users created with the create_IAM_user script (included in the AWS GameKit plugin download), use the instructions for the AWS Identity and Access Management (IAM) user type. You can get short-term or long-term access keys for these users.

Users need programmatic access if they want to interact with AWS outside of the AWS Management Console. The way to grant programmatic access depends on the type of user that's accessing AWS.

To grant users programmatic access, choose one of the following options.

Which user needs programmatic access?	To	By
Workforce identity (Users managed in IAM Identity Center)	Use temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions for the interface that you want to use. • For the AWS CLI, see Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS*

Which user needs programmatic access?	To	By
		Command Line Interface User Guide. • For AWS SDKs, tools, and AWS APIs, see IAM Identity Center authentication in the *AWS SDKs and Tools Reference Guide*.
IAM	Use temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions in Using temporary credentials with AWS resources in the *IAM User Guide*.
IAM	(Not recommended) Use long-term credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions for the interface that you want to use. • For the AWS CLI, see Authenticating using IAM user credentials in the *AWS Command Line Interface User Guide*. • For AWS SDKs and tools, see Authenticate using long-term credentials in the *AWS SDKs and Tools Reference Guide*. • For AWS APIs, see Managing access keys for IAM users in the *IAM User Guide*.

Securing credentials with the AWS GameKit plugin

It's safe to enter your AWS account credentials into the AWS GameKit plugin. AWS GameKit never stores your credentials with your game project or in the AWS GameKit configuration files. Credentials are never included in game distributables.

The AWS GameKit plugin asks for your AWS credentials during set up for your game project. By default, your credentials are cached locally so you don't need to re-enter them. You can enter different credentials at any time, such as when switching environments.

Tips for protecting your credentials with AWS GameKit:

- Don't download the AWS GameKit plugin from anywhere other than an official source.
- Don't enter credentials in your game code, even in test code for convenience.
- Avoid storing credentials locally in files that are shared. The AWS GameKit plugin option "Store my credentials" saves your credentials to your home directory (`~/.aws/credentials`). This standard location is used by other AWS tools.

Related AWS topics

- Understanding and Getting your AWS credentials, Programmatic access, *AWS General Reference*
- Best practices for managing AWS access keys, *AWS General Reference*
- Where are configuration settings/credential information stored?, *AWS Command Line Interface User Guide*

Getting started with AWS GameKit

Use the content in this section to explore AWS GameKit for Unreal and how to start integrating cloud-based game features to your Unreal projects.

Topics

Explore AWS GameKit in the Unreal Editor

Summary

This overview gives game developers a brief survey of the AWS GameKit plugin tools and functionality for the Unreal Editor.

When working on an Unreal project with the AWS GameKit plugin enabled, you can use AWS GameKit components to:

- the section called “Manage your cloud project” – Set up an AWS GameKit configuration for your Unreal project, link to an AWS account, and manage staging environments.
- the section called “Deploy backend services for your cloud features” – Configure and deploy a backend for each game feature.
- the section called “Build AWS GameKit features into your frontend” – Use example materials to experiment with cloud feature functionality in your game or for rapid prototyping.

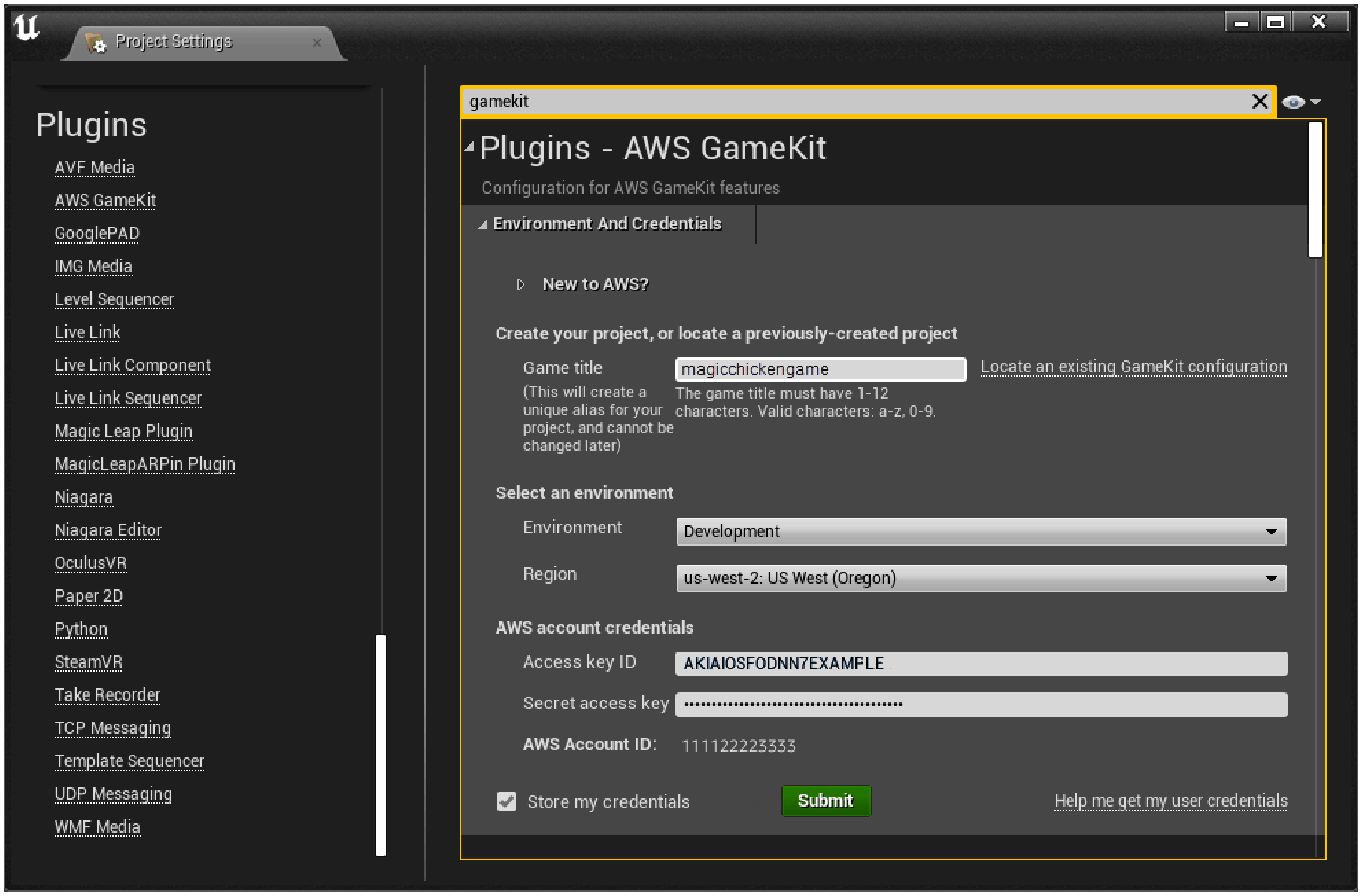
Manage your cloud project

Each Unreal project must have a companion AWS GameKit cloud project to manage the Unreal project's backend services on AWS. The cloud project contains your configurations and code for each cloud fetaure. AWS GameKit uses this information when deploying and interacting with AWS resources for these features.

Find cloud project settings

1. In the Unreal Editor toolbar, open **Edit, Project Settings** and go to the **AwsGameKit** plugin section.

2. Expand the **Environment and Credentials** section, as shown in the following screenshot.



gamekit

Plugins - AWS GameKit

Configuration for AWS GameKit features

Environment And Credentials

New to AWS?

Create your project, or locate a previously-created project

Game title: magicchickengame
(This will create a unique alias for your project, and cannot be changed later)
The game title must have 1-12 characters. Valid characters: a-z, 0-9.

Select an environment

Environment: Development

Region: us-west-2: US West (Oregon)

AWS account credentials

Access key ID: AKIAIOSFODNN7EXAMPLE

Secret access key: ••

AWS Account ID: 111122223333

✓ Store my credentials

Work with cloud project settings

Note

If you use AWS credentials for users created with the new AWS IAM Identity Center, you might experience unusual behavior and messaging. These credentials provide short-term access only, which means you must regularly generate new credentials and enter them into the plugin, If you created users with the automated script supplied with the AWS GameKit download package, the AWS credentials for these users are long-term access keys.

- Set up a cloud project. When setting up a cloud project, you provide the following information:

    - Game title is a unique alias for the cloud project. AWS GameKit stores cloud project configurations and code in a folder with this name in the Unreal project folder, and references the game title in all AWS resources deployed for the project.
    - Environments let developers manage multiple replicas of project's backend in parallel, such as for development, testing, and production. Users set the active environment they want to work in. Project teams commonly start with the default **Development** environment.
    - Region specifies the physical location where AWS GameKit deploys the AWS resources for an environment. Each environment has one designated region.
    - AWS account credentials identify which AWS account to use for the project and validate an account user. If the account is valid and the user has access rights for AWS GameKit, then they can deploy or update AWS resources for the project's backend.
- Work with an existing project. Specify the game title, choose an environment to work in, and submit your AWS security credentials.
- Switch environments. Only one cloud project environment can be active at a time. All actions you take on cloud features and deployed backend resources impact the active environment only. You can switch to a different environment on this page. You might also have to resubmit security credentials.
- Change account credentials. AWS GameKit automatically reuses security credentials from the previous session on the current device. If your credentials have changed, use cloud project settings to submit new ones.
- Change deployment region. You might want to change the location where AWS resources are deployed for the active environment. You can't change the deployment region if the environment has AWS resources deployed.

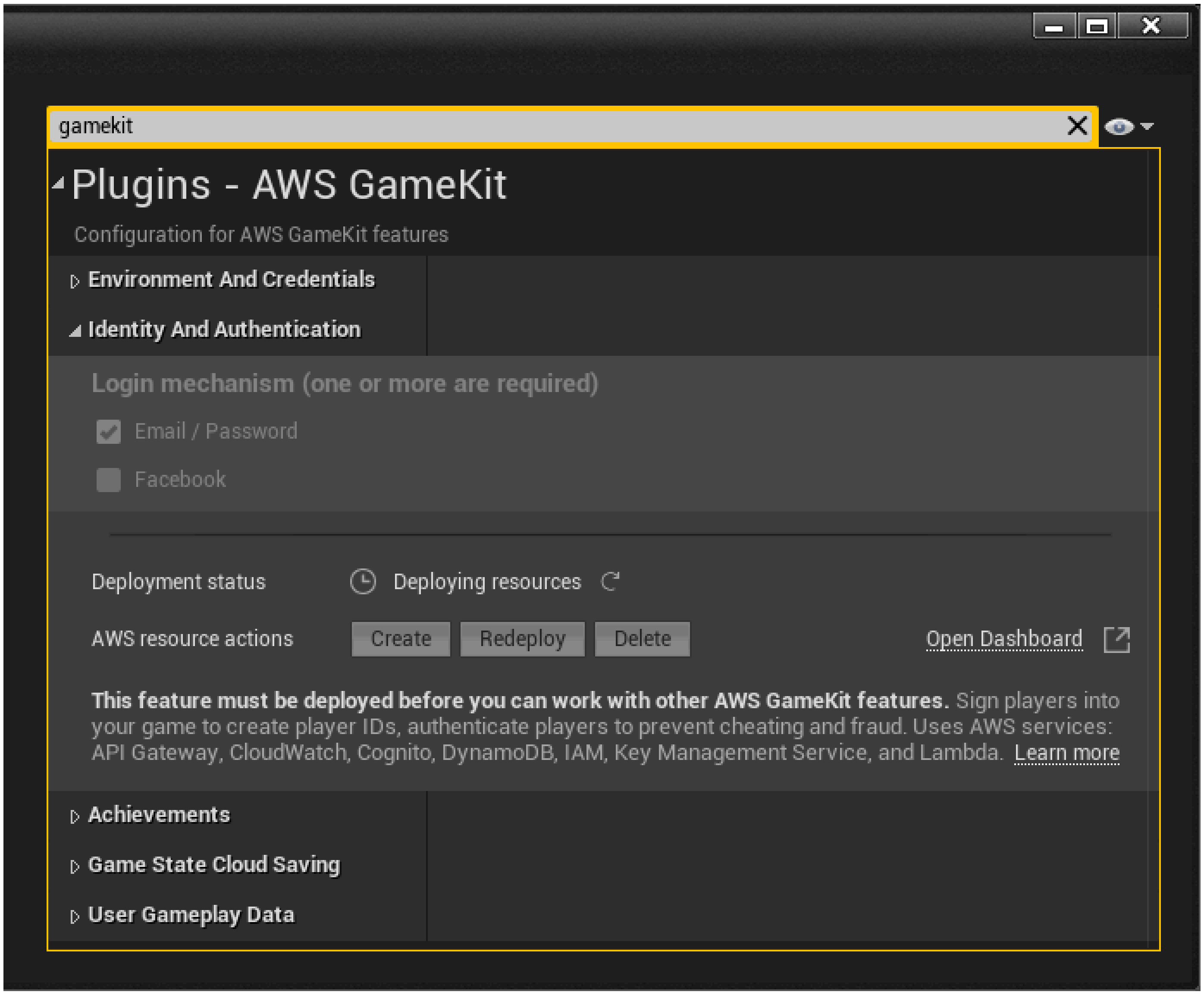
Deploy backend services for your cloud features

In feature settings, users can build a cloud-connected backend for each AWS GameKit feature. After selecting an environment to work in and submitting credentials, users can configure a feature and then create or update AWS resources to run the feature's backend. AWS GameKit maintains each environment's feature configuration and deployment status in the cloud project.

Find game feature settings

1. In the Unreal Editor toolbar, open **Edit, Project Settings** and go to the **AwsGameKit** plugin section.

2. Expand the section for each game feature. For example, the following screenshot displays the settings for the identity and authentication feature.



This feature must be deployed before you can work with other AWS GameKit features. Sign players into your game to create player IDs, authenticate players to prevent cheating and fraud. Uses AWS services: API Gateway, CloudWatch, Cognito, DynamoDB, IAM, Key Management Service, and Lambda. Learn more

Work with feature settings

- View status of deployed features. Each feature section displays the feature's current deployment status. Configuration settings reflect the feature backend as deployed. If the feature isn't deployed yet, the default values are shown.
- Configure feature settings. Each feature has settings that you can customize before deploying the feature backend. For example, with identity and authentication, you can choose to let players log in with a registered user name and password, a Facebook account, or either.

- Deploy, update, or delete AWS resources. After choosing feature settings, you can create new AWS resources for the feature or update them if they're already deployed. You can also remove the feature by deleting the deployed resources.
- Access feature dashboards. You can activate a metrics dashboard for each AWS GameKit feature. Each dashboard contains operational metrics for the AWS services and resources used by the feature. For example, you can monitor API calls to the backend. Dashboards use the Amazon CloudWatch service.

Build AWS GameKit features into your frontend

The AWS GameKit plugin provides tools for connecting your Unreal project frontend to your backend services on AWS. These include the AWS GameKit API for connecting with the backend services for each feature and example assets to illustrate API calls and core workflows for the frontend.

After you deploy a feature's backend services on AWS, your Unreal project is automatically configured with the backend endpoints for the current active AWS GameKit environment. You can make live API calls and run levels with example assets that communicate with your project backend.

Find AWS GameKit assets and tools

In the Unreal Editor, open the **Content Browser** and locate the following folders:

- `AwsGameKit Content` contains example blueprints, UI elements, and widget blueprints for each game feature. There is also a complete game example for the user gameplay data feature
- `AwsGameKit C++ Classes` contains example C++ code and resources for integrating game feature functionality into a game using C++ code.
  - The `AwsGameKitEditor` public folder contains a code example file for each game feature. This code includes function calls for all API operations for the game feature.
  - The `AWSGameKitRuntime` public folder contains function libraries and utilities that support the example assets.

Test calls to your project backend

If you've deployed a backend for any of the AWS GameKit features, you can make direct calls to it in the Unreal Editor:

1. In the Content Browser, open the `AwsGameKitEditor` public folder and choose a code example file.
2. Add the asset to a level in your project, and select the object in the viewport.
3. Open the **Details** panel. The custom AWS GameKit **Details** UI displays all the API calls in the code example file.
4. Log in with a player account. If you haven't set one up yet, use the identity and authentication example file to register and verify an account.
5. When working with the achievements code examples, use the provided function to save a set of sample achievement definitions to your achievements backend. You can use this data to experiment with the achievements functionality and delete it as needed.

Integrate AWS GameKit features into your game

This topic outlines an implementation path to follow when using AWS GameKit to build cloud-based game features with AWS GameKit. Each step provides a link to detailed documentation.

1. **Get the AWS GameKit for your game engine.**
   - Download the plugin for your development environment.
   - Install the AWS GameKit plugin. For more information, see Install the AWS GameKit plugin with Unreal Engine.
2. **Get an AWS account and set up user accounts for your game project.**
   - Create an AWS account or use an existing account. You manage all your AWS GameKit resources and services through this account. For more information, see Sign up for an AWS account.
   - Set up an AWS user for every person who works with the AWS GameKit plugin. This step includes adding access permissions for AWS GameKit. Each user must have security credentials for use with the plugin and (optionally) sign-in credentials to use the AWS Management Console. For more information, see Set up a user with AWS GameKit access.
3. **Set up an AWS GameKit cloud project.**
   - Use the AWS GameKit UI in your game engine to define cloud project settings, including a game title/alias, a working environment, and a location for the backend services. This step creates a set of configurations and templates that AWS GameKit uses to build your backend on AWS. For more information, see Set up the AWS GameKit plugin for your game.
   - Link the project to an AWS account by providing your user security credentials.

4. **Configure the identity and authentication game feature and deploy AWS resources.**

   All other AWS GameKit features require identity and authentication, so you must create the backend for this feature before setting up other features. For more information , see Add identity and authentication to your project.

   - Use the AWS GameKit plugin project settings to configure the identity and authentication feature for your game.
   - Create AWS resources for the authentication backend. You can track the deployment process, redeploy or delete resources from the AWS GameKit UI.

     > **Note**
     >
     > You might begin incurring AWS charges when you deploy AWS resource, depending on whether your account is eligible for AWS Free Tier benefits.

   - Optionally, activate an operational metrics dashboard for your newly deployed identity and authentication backend. For more information, see Work with game feature dashboards.

5. **Add identity and authentication workflows to your game frontend.**

   Create workflows in your game frontend to let players register with your game, log in and out, and other identity-related tasks. With the identity and authentication backend in place, you can use the AWS GameKit sample materials to prototype and test the workflows. For more information, see Work with the identity and authentication examples.

6. **Configure and deploy game backends for additional AWS GameKit features and connect game functionality.**
   - Achievements
   - User gameplay data
   - Game state cloud saving

Working in the AWS GameKit UI

The content in this section provide detailed instructions on how to use the AWS GameKit plugin interface to build cloud-based game features backed with AWS Cloud services.

Topics

Set up the AWS GameKit plugin for your game

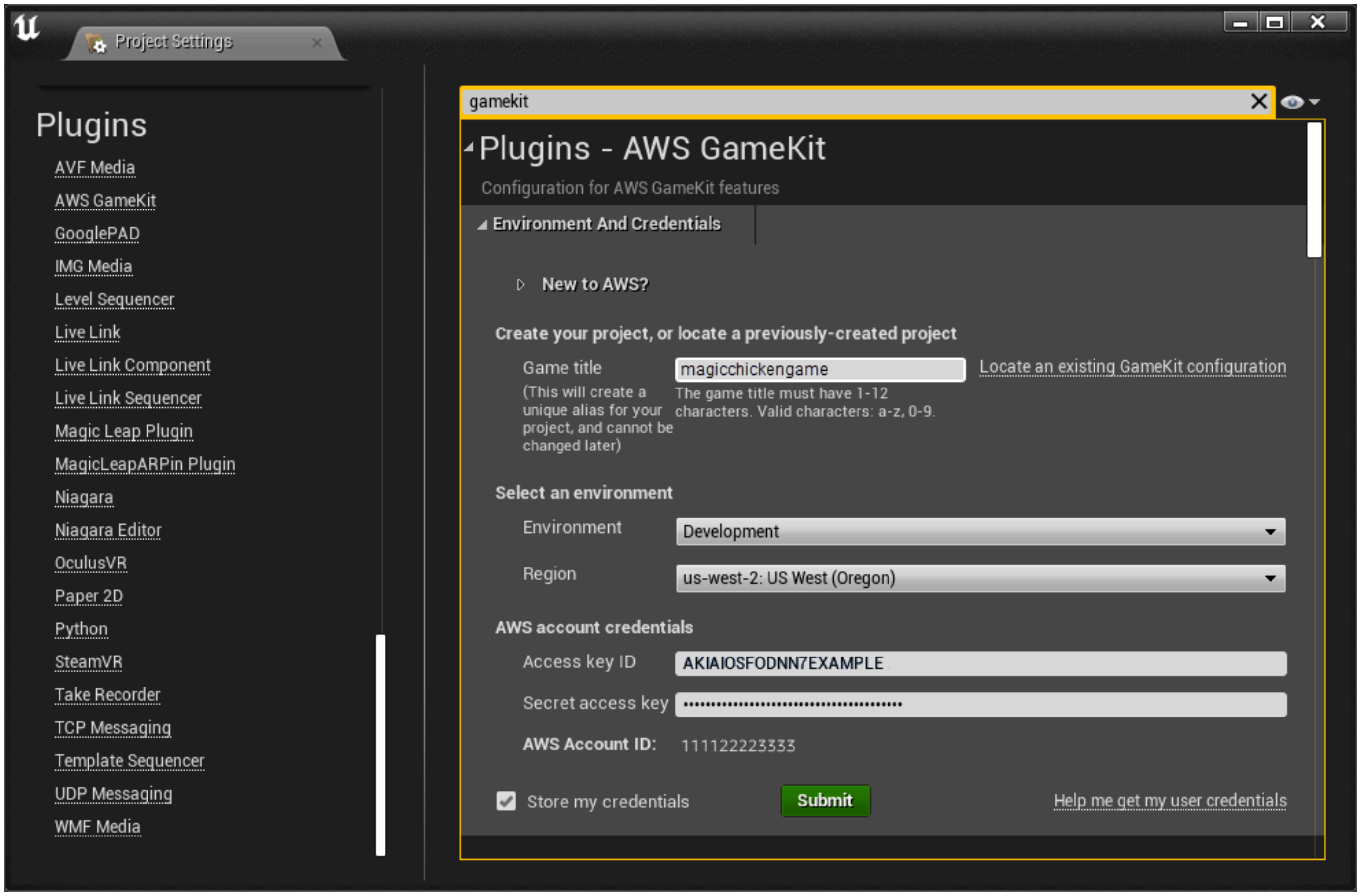
Before you can use the AWS GameKit plugin to build and maintain your game's AWS Cloud-based services, you must set up your game project with an AWS GameKit configuration and an AWS account.

Note

If you've already created an AWS GameKit configuration for your game, you cannot create a second, concurrent configuration.

To set up your AWS GameKit configuration

1. In the Unreal Editor toolbar, open **Edit, Project Settings** and go to the **AwsGameKit** plugin section.
2. Expand **Environment and Credentials**.



Configuration for AWS GameKit features

Create your project, or locate a previously-created project

Game title: magicchickengame

(This will create a unique alias for your project, and cannot be changed later)

The game title must have 1-12 characters. Valid characters: a-z, 0-9.

Select an environment

Environment: Development

Region: us-west-2: US West (Oregon)

AWS account credentials

Access key ID: AKIAIOSFODNN7EXAMPLE

Secret access key: ••

AWS Account ID: 111122223333

✓ Store my credentials

3. Do either of the following:

   - To create a new configuration, enter a new **Game title**. The game title string must be all lowercase alphanumeric characters. It cannot contain spaces or special characters.

     AWS GameKit uses the game title in the names of components for your game project. This includes a configuration folder, which is stored locally with your game project files, and every AWS resource that you deploy for this game project.

     Warning

     Avoid using the following strings in your game title, as they can cause issues when creating the backend services for your game:

     - `aws`
     - `amazon`

You cannot change the game title after you've submitted it and begun to work with game features. If you need to change a game title later, you must delete the entire AWS GameKit configuration for your game, including all backend services, and start a new configuration.

- To use a previously created configuration, choose **Locate an existing GameKit configuration**. Then browse to your game project files and search for the AWS GameKit configuration folder.

In future sessions, the AWS GameKit plugin automatically enters the last used game title.

4. Choose an **Environment** to work in. You can select a default environment (**Development**, **Testing**, or **Production**) or create a custom environment.

   In future sessions, the AWS GameKit plugin automatically selects the last used environment. You can switch environments at any time.

   The AWS GameKit configuration for your game project maintains separate sets of information for each environment. When you select an environment to work in, the AWS GameKit plugin loads the game feature settings and AWS deployments for the selected environment.

5. For the selected environment, choose an AWS **Region** where you want to place resources for your game backend. Each environment can deploy resources to only one Region.

   Choose a region that makes the most sense for the environment. For example, you might place Development environment resources to be geographically near the development team, but place Production resources close to your players. To verify region support for the game features you want to add, see AWS GameKit supported AWS Regions.

   In future sessions, the AWS GameKit plugin automatically uses the Region for whichever environment you select.

6. Under **AWS account credentials**, enter your two-part AWS Identity and Access Management (IAM) user access key. This key includes an **Access key ID** and a **Secret access key**. These credentials uniquely identify your unique IAM user in an AWS account. For details, see Get your AWS security credentials.

> **Important**
>
> Your IAM user must have access permissions in order to work with the AWS resources for your game backend. If you don't have these permissions, you can successfully submit your user credentials in the AWS GameKit plugin, but you won't be able to create, redeploy, or delete AWS resources for the game features. For details, see Set up AWS account for AWS GameKit.

If you already have the AWS Command Line Interface (AWS CLI) installed and configured with the same credentials, the AWS GameKit plugin automatically recognizes those credentials and uses them.

7. Optionally, select **Store my credentials** to save your AWS account credentials on the current device for use in future sessions. If you choose not to save your credentials, you must re-enter them at the start of each session in the plugin and whenever you switch environments. For questions about security with the AWS GameKit plugin, see Securing credentials with the AWS GameKit plugin.

   The AWS GameKit plugin stores credentials locally in a text file in your home directory `C:\Users\<user ID>\.aws\credentials`. Each set of saved credentials is associated with a game title and environment. This means that you can use different credentials with each environment, which is useful for teams that tightly control access to a game's AWS resources.

8. Choose **Submit** to create a new AWS GameKit configuration and set the active environment.

   In the future, when you open this game project in the Unreal Editor, the AWS GameKit plugin automatically selects the last used game title, environment, region, and credentials (if you stored them). You can switch environments or credentials at any time in the project settings for AWS GameKit, **Environment and Credentials**.

Remove AWS GameKit from a game project

Summary

Use the procedures in this topic to remove some or all AWS GameKit game features from your game projects. To ensure that you don't continue incurring charges for features that you're no longer using, *you need to remove all deployed AWS cloud resources. This topic is for game developers who want to ensure that all unneeded AWS GameKit components are removed or deleted from a game project.*

You can remove AWS GameKit components from your game project using one of the following processes:

- Remove individual game features from your project.
- Remove all AWS GameKit components from your project.

Removing individual game features

If you want to replace an existing game feature configuration with a new configuration, you do not need to delete your deployed AWS resources. Instead, use the AWS GameKit plugin UI to revise your configuration settings and then update your deployed AWS resources.

If you decide to remove a game feature from your game, complete the following tasks:

- In your game frontend, remove all functionality that relies on the game backend, as it will stop working as soon as you delete the AWS resources that run the backend services.
- Consider backing up data related to the game feature that is being removed. This might include configuration data, such as achievement definitions, as well as game-generated player data.
- Delete the AWS resources that run the backend services for the game feature, as described in the following steps.

Note

You cannot remove the identity and authentication feature if you are using any other AWS GameKit game features. All other features use the player authentication processes.

Unreal Engine

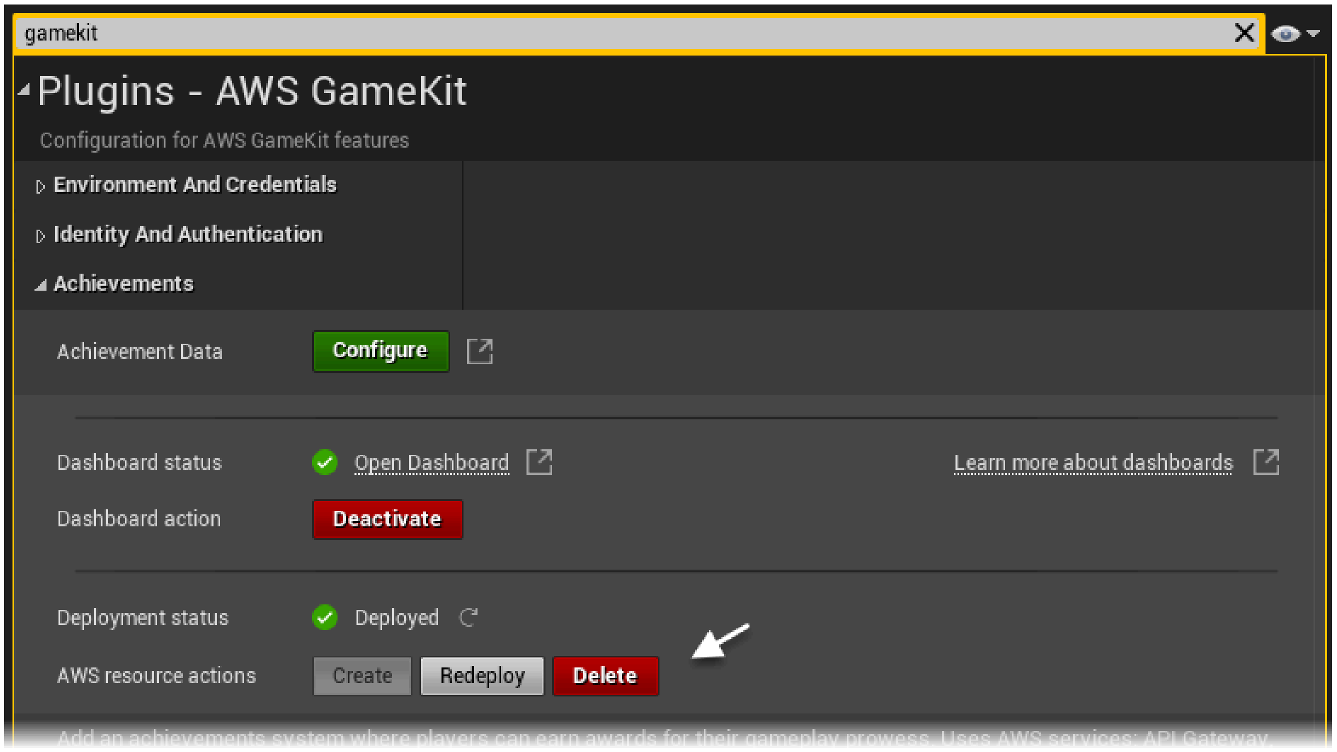
To remove a AWS GameKit game feature

1. In the Unreal Editor toolbar, open **Edit, Project Settings** and go to the **AwsGameKit** plugin section.

2. In the **Environment and Credentials** section, select the environment and AWS Region that you want to remove a game feature for. Enter the appropriate AWS credentials and choose **Submit**.

    > **Note**
    >
    > You must complete this procedure for each environment/Region combination where you want to remove the game feature.

3. Expand the project settings section for the game feature that you want to remove and check the deployment status. If AWS resources are deployed for the game feature, choose the AWS resource action **Delete**. This action deletes all of the AWS resources that provide backend services and dashboards for the game feature. Some logs may be retained.



gamekit

Plugins - AWS GameKit

Configuration for AWS GameKit features

Environment And Credentials

Identity And Authentication

Achievements

Achievement Data: Configure

Dashboard status: Open Dashboard — Learn more about dashboards

Dashboard action: Deactivate

Deployment status: Deployed

AWS resource actions: Create, Redeploy, Delete

Add an achievements system where players can earn awards for their gameplay prowess. Uses AWS services: API Gateway

You do not need to delete or change the game feature's configuration settings or delete AWS GameKit files that are stored locally with your game project.

Removing all AWS GameKit plugin components

If you want to start fresh with AWS GameKit, follow the steps below to remove the existing configuration. You can then start over with a new default AWS GameKit configuration.

To remove the AWS GameKit plugin from your game project, complete the following tasks:

- In your game frontend, remove all functionality that relies on the game backend, as it will stop working as soon as you delete the AWS resources that run the backend services.
- Consider backing up data related to the game features that are currently being used in your game. This might include configuration data, such as achievement definitions, as well as game-generated player data.
- Delete the AWS resources that are currently deployed for all game features, as described in the following steps.

Unreal Engine

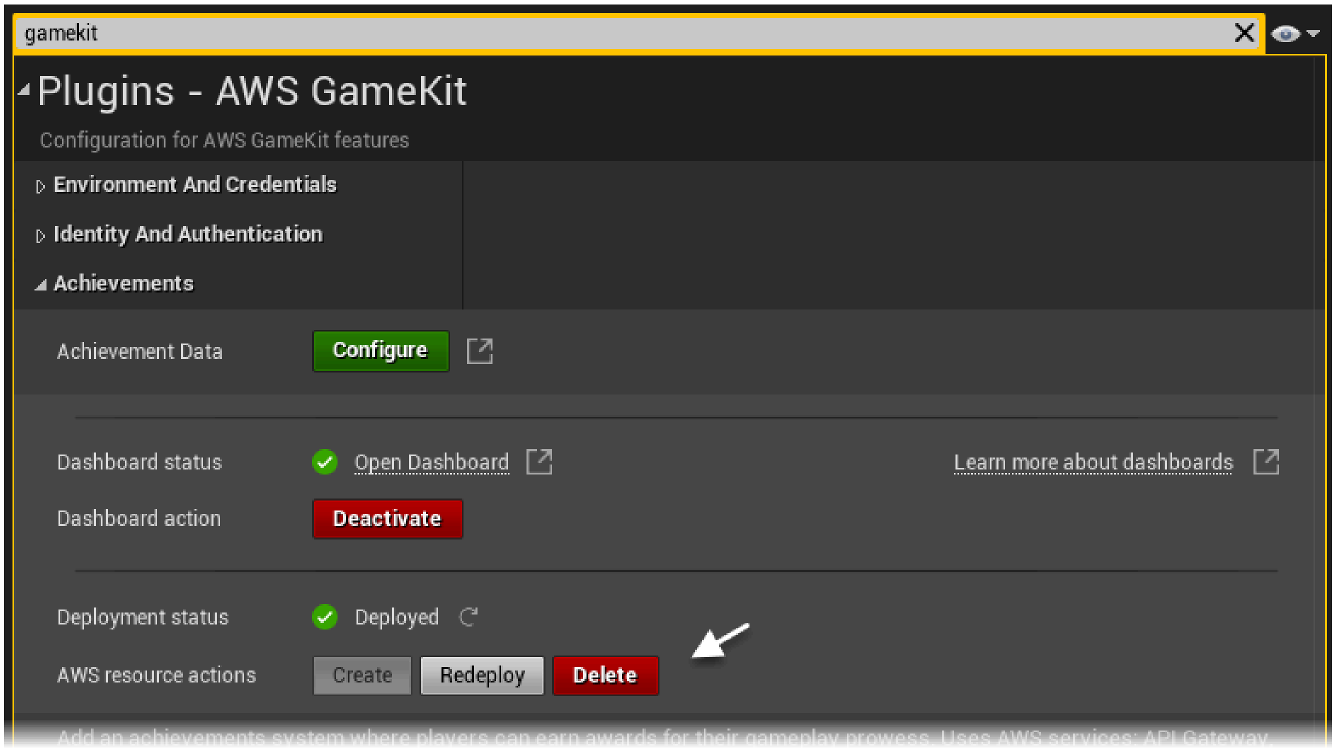
To remove all plugin components

1. In the Unreal Editor toolbar, open **Edit, Project Settings** and go to the **AwsGameKit** plugin section.
2. In the **Environment and Credentials** section, select an environment and AWS Region where AWS resources are currently deployed for one or more game features. Enter the appropriate AWS credentials and choose **Submit**.

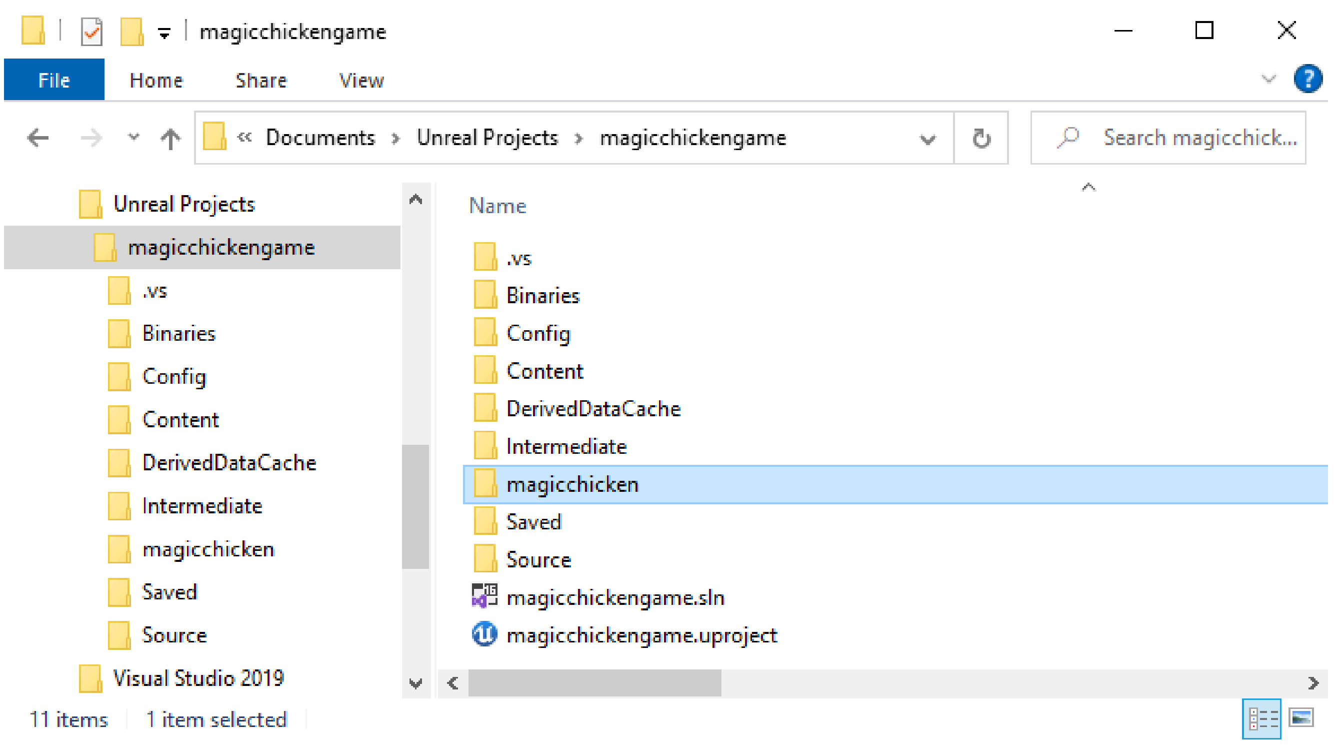
   Note

   You must follow this procedure for every environment/Region combination where you've deployed AWS resources.

3. For each AWS GameKit game feature, expand the project settings section and check the deployment status. If AWS resources are deployed for the game feature, choose the AWS resource action **Delete**. This action deletes all of the AWS resources that provide backend services and dashboards for the game feature. Some logs may be retained.


gamekit
Plugins - AWS GameKit
Configuration for AWS GameKit features
Environment And Credentials
Identity And Authentication
Achievements
Achievement Data
Configure
Dashboard status
Open Dashboard
Learn more about dashboards
Dashboard action
Deactivate
Deployment status
Deployed
AWS resource actions
Create
Redeploy
Delete
Add an achievements system where players can earn awards for their gameplay prowess. Uses AWS services: API Gateway

4. Repeat this process with every environment/Region combinations where you've deployed AWS resources for a AWS GameKit game feature.
5. In a file browser, open the directory containing for your game project files. Files for AWS GameKit are located in folder that is named with the your AWS GameKit game title. The following screenshot illustrates the folder for a project names "magicchickengame", with a game title of "magicchicken". Delete this directory. This step removes all of the locally cached configuration settings and other support files that the AWS GameKit plugin uses to manage the backend services for each game feature.



6. In the Unreal Editor, go to **Edit > Plugins > AwsGameKit** and disable the plugin for your game project. This step ensures that no AWS GameKit components are included in your game project's release packages.

You can decide at any time to re-enable the AWS GameKit plugin for your game project. At that point, you'll be prompted enter a new game title and create a new default configuration for your game.

Troubleshoot AWS GameKit plugin issues

This topic provides guidance to help you resolve common usage issues with the AWS GameKit plugin.

[Unreal] Can't open game project after enabling the AWS GameKit plugin

Problem: When you enable the AWS GameKit plugin in Unreal Editor and restart the program as prompted, Unreal Editor attempts to restart but fails with the following message:

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

Cause: This issue occurs when you enable the AWS GameKit plugin for a C++ game project where the game code has not yet been built. For example, if you enable the AWS GameKit plugin for a brand new game project, you are likely to encounter this issue.

Resolution: You need to manually build your game project solution:

1. Open the folder that contains the project files for your Unreal game project and locate the project's solution file (`.sln`).
2. Open the file in your IDE and build your game project code. If you're using Visual Studio, choose **Build > Build Solution** from the menu toolbar.
3. After the solution builds successfully, you can open the game project in the Unreal Editor, with full access to the AWS GameKit plugin elements.

[Unreal] Deployment is not completing

Problem: When deploying AWS resources for a game feature, the action is taking a long time to complete.

Cause: When you deploy AWS resources, it can take some time to complete the set-up activity that happens behind the scenes. Average deployment times for each feature are:

- Identity and authentication: 5-10 minutes
- Achievements: 10-30 minutes
- User gameplay data: 5-10 minutes
- Game state cloud saving: 5-10 minutes

Resolution: You can track the ongoing progress of deployment activity in the following ways:

- Use the Unreal Editor output log to view detailed event messaging. To open the log, go to **Window > Developer Tools > Output Log**.
- Go to the custom dashboard for the game feature. A feature's dashboard is generated near the beginning of the deployment process. To open the dashboard, in the Unreal Editor, go to **Edit > Project Settings > Plugins > AwsGameKit**.
- Watch ongoing events in the AWS Management Console. In the console, go to the AWS CloudFormation service and open the stack for the game feature being deployed.

Work with game feature dashboards

Summary

This topic is for game developers who want to monitor backend activity for their Unreal Engine projects. For each AWS GameKit feature, you can activate a custom Amazon CloudWatch metrics dashboard. Use these dashboards to monitor your backend AWS services set up through AWS GameKit.

In AWS GameKit, you can activate CloudWatch dashboards for your game's backend. Dashboards include metrics and alarms for the deployed AWS resources that run backend services for each game feature. These dashboards provide insight on operational activity across each game feature, and are typically used during game testing or in production when the game has an active player base.

Dashboards metrics include those used in the cost calculations for each AWS service. These metrics can provide clarity on how your game's backend activity is impacting cost.

For more information about CloudWatch dashboards, see Related AWS topics.

Activating or deactivating a dashboard

AWS GameKit dashboards track the metrics for a game feature's deployed AWS resources, and you manage dashboard status as part of the game feature configuration. If you deploy a game feature to more than one staging environment (such as QA and Production), you activate or deactivate the feature's dashboard in each environment. You can set the dashboard status for a feature at any time.

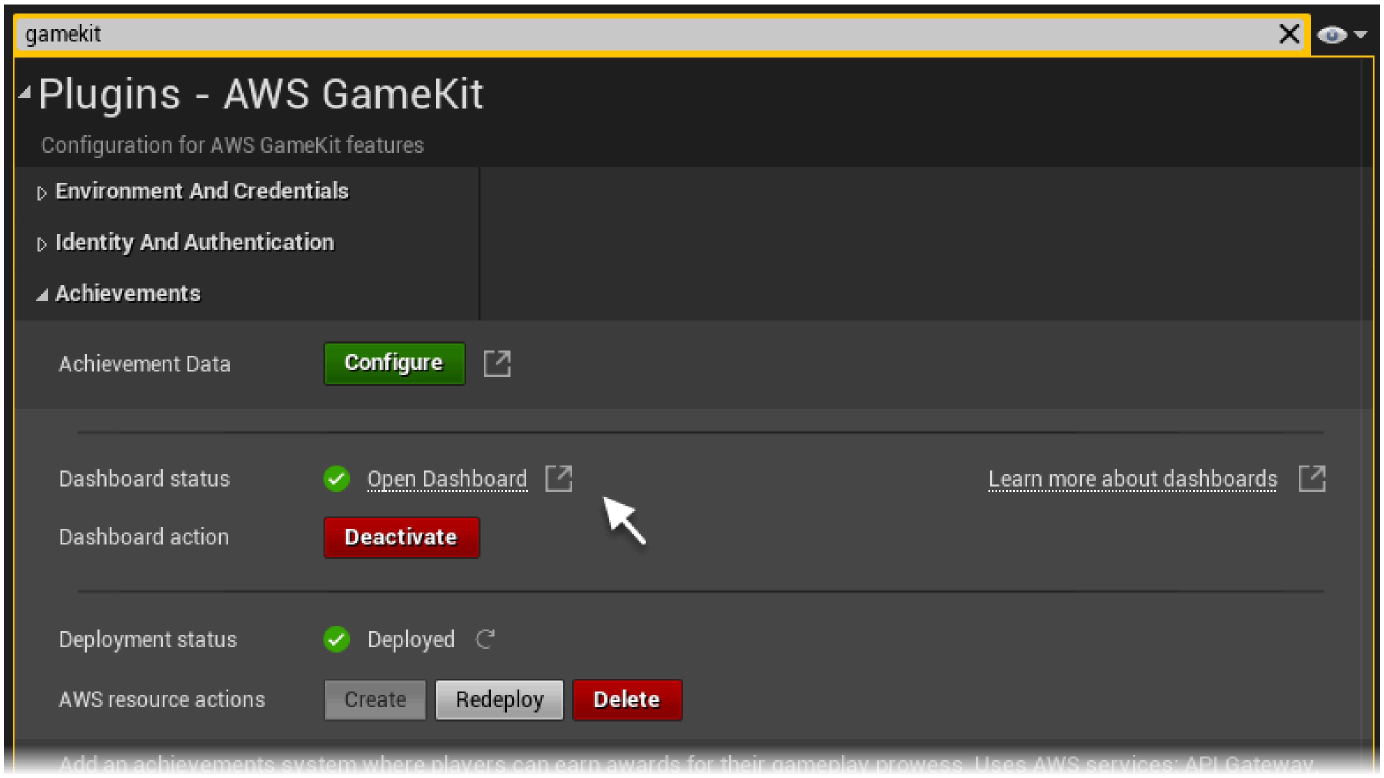
Note

The AWS Free Tier for CloudWatch includes up to three dashboards. If you activate multiple dashboards, you can exceed this quota and your AWS account can incur charges.

To set activation status for a dashboard

Set dashboard activation status in the AWS GameKit settings.

1. In the Unreal Editor toolbar, open **Edit, Project Settings** and go to the **AwsGameKit** plugin section. Select the environment you want to work with in **Environment & Credentials**.



gamekit

Plugins - AWS GameKit

Configuration for AWS GameKit features

Environment And Credentials

Identity And Authentication

Achievements

Achievement Data	Configure
Dashboard status	✓ Open Dashboard
Dashboard action	Deactivate
Deployment status	✓ Deployed
AWS resource actions	Create Redeploy Delete

Learn more about dashboards

2. Open the game feature you want to activate or deactivate a dashboard for. Each game feature section displays the current dashboard status. In the preceding screenshot, the achievements feature dashboard has been deployed with an active dashboard.
3. For **Dashboard action**, choose **Deactivate** or **Activate** to change the feature's current dashboard status. If you activate the dashboard for a feature that isn't deployed yet, a dashboard is created during deployment. If the feature is already deployed, AWS GameKit creates or deletes the feature's dashboard. This update is made immediately; you don't have to redeploy the feature.

   The default dashboard status depending on which environment is active. Dashboards are automatically activated in the **Testing** and **Production** environments.

   You can track the change in the **Deployment status** field.

Opening a dashboard

You can access an activated dashboard for a deployed feature from your game engine or in the AWS Management Console for CloudWatch.

To open a dashboard from the game engine

Open game feature dashboards in the AWS GameKit settings.

1. In the Unreal Editor toolbar, open **Edit, Project Settings** and go to the **AwsGameKit** plugin section. Select the environment you want to work with in **Environment & Credentials**.
2. Open the feature you want to open a dashboard for. If the feature has an active dashboard, the **Dashboard status** field contains a check mark. If not, activate the dashboard and wait for AWS GameKit to create it, which takes up to a few minutes.
3. Choose **Open Dashboard**. The dashboard opens in a browser window in the AWS Management Console for CloudWatch.

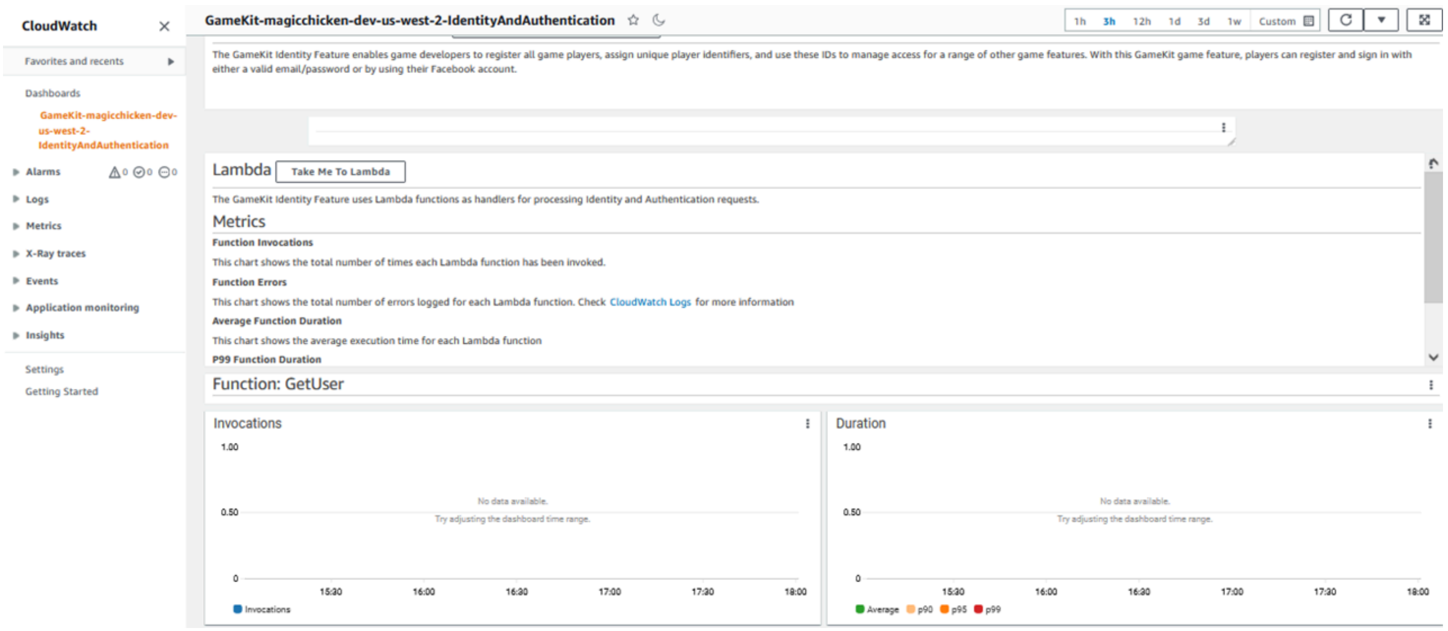
Viewing dashboard content

Each AWS GameKit feature has its own dashboard. Dashboard names use the same naming convention as the AWS resources that are deployed through AWS GameKit. For example, a dashboard named `GameKit-magicchicken-dev-us-west-2-IdentityAndAuthentication` tracks AWS resources with a similar prefix. The name references the following information from your game project's AWS GameKit configuration:

- Alias/game title (as in "magicchicken").
- Deployment environment (as in "dev").
- Deployment AWS Region (as in "us-west-2").
- Game feature (as in "IdentityAndAuthentication").

The dashboard is viewed in the AWS Management Console for CloudWatch, as shown in the following screenshot. Dashboard design and format varies by AWS GameKit version.

You can customize an AWS GameKit dashboard using the CloudWatch dashboard tools. These changes apply to this specific custom dashboard, and will be lost if this dashboard is deactivated. If you want to apply permanent or global changes to AWS GameKit dashboards, update your game's base templates for AWS GameKit.



The GameKit Identity Feature enables game developers to register all game players, assign unique player identifiers, and use these IDs to manage access for a range of other game features. With this GameKit game feature, players can register and sign in with either a valid email/password or by using their Facebook account.

The GameKit Identity Feature uses Lambda functions as handlers for processing Identity and Authentication requests.

Function Invocations

This chart shows the total number of times each Lambda function has been invoked.

Function Errors

This chart shows the total number of errors logged for each Lambda function. Check CloudWatch Logs for more information

Average Function Duration

This chart shows the average execution time for each Lambda function

P99 Function Duration

Key dashboard metrics

Because each game feature uses a different collection of AWS services for its backend, the dashboard for each feature tracks a different set of metrics. However, all AWS GameKit dashboards include information from these core services.

AWS Lambda

All AWS GameKit game features use Lambda functions to handle processing for feature-related API requests. The Lambda metrics for each request type include the number of invocations and the average amount of compute time used to process the requests. Metrics to pay particular attention to include:

- **Latency (P99, P95 and P90).** These charts show how long it takes for requests or operations to complete on the backend. When evaluating latency, there are no specific “good” or “bad” values; instead, watch for sudden increases that can result in a degradation in response time. To mitigate, consider adjusting the number of Lambda functions (see Update AWS Lambda settings in the launch readiness guide).
- **Concurrent executions.** These charts track the number of functions the game backend executes at any single moment. In particular, look for an increase in latency with a constant number of concurrent executions. This scenario indicates that the number of concurrent executions isn't enough to handle the load. To mitigate, consider increasing the allowed number of concurrent executions (see Update AWS Lambda settings in the launch readiness guide).

- **Function errors.** These charts show the number of errors per API. To understand the cause of these errors, inspect the Amazon CloudWatch logs and mitigate as needed.

Amazon API Gateway

All requests to the AWS GameKit API from game clients pass through API Gateway to backend Lambda functions. Metrics for API Gateway include the number of requests received over time and the average response time. Metrics to pay particular attention to include:

- **Latency (P99, P95 and P90).** These charts show how long it takes it takes for requests or operations to complete on the backend. When evaluating latency, there are no specific “good” or “bad” values; instead, watch for sudden increases, which can result in a degradation in response time. To mitigate this scenario, inspect the latency charts for the underlying services (usually Lambda functions and DynamoDB) and mitigate based on the recommendations for each service.
- **4XX and 5XX errors.** These charts show the number of errors per API Gateway resource. To understand the cause of these errors, inspect the Amazon CloudWatch logs and mitigate as needed.

Amazon Cognito

For games that use the AWS GameKit identity and authentication feature, Amazon Cognito manages all player registrations and logins, whether through username/password or through third-party identity providers such as Facebook. Metrics for Amazon Cognito track all authentication and security events involving AWS GameKit API requests. Metrics to pay particular attention to include:

- **Security.** This chart shows the number of security-related events that occur. If you experience a high number of events, consider enabling advanced security features (see Update Amazon Cognito settings in the launch readiness guide).

Amazon DynamoDB

DynamoDB tables store a variety of information for AWS GameKit game features, including player IDs, achievement definitions, user gameplay data, game saves, and player achievement status. Metrics for DynamoDB include percentage capacity consumed, read/write activity, request latency, and other metrics relevant to individual game features. Metrics to pay particular attention to include:

- **Throttle.** These charts track the number of throttled requests and events.
- **Table request latency.** These show how long it takes for requests or operations to complete on DynamoDB . When evaluating latency, there are no specific “good” or “bad” values; instead, watch for sudden increases, which can result in a degradation in response time.

To mitigate throttling or latency issues, consider enabling the DynamoDB auto scaling feature (see Update Amazon DynamoDB settings in the launch readiness guide).

Related AWS topics

- *Amazon CloudWatch User Guide*: Using Amazon CloudWatch dashboards

AWS GameKit feature: Identity and authentication

Summary

With the identity and authentication game feature, you can set up player sign-in workflows and assign each player a unique ID. Player IDs support a range of scenarios that require verification and authentication, including communication between game clients and backend services. You must deploy the identity and authentication feature for your game to use any other AWS GameKit feature, which all rely on the ability to authenticate players. Primary audience: game owners and developers who want to better understand the use cases for identity and authentication game feature.

The AWS GameKit feature identity and authentication provides tools for creating and storing unique player identifiers. With AWS GameKit, Player IDs are used to manage player access, authenticate communications between game clients and backend services, and other scenarios that require identity verification and authentication.

The AWS GameKit solution for player identity and authentication supports workflows for new player registration, player sign-in, and account recovery. New players can be registered based on email and/or social media, with secure features such as email verification.

Potential uses for player identity:

- Enable players to sign in to a game on multiple devices and pick up the game where they left off. AWS GameKit cloud-based features such as user gameplay data, achievements and game state cloud saving track player-specific information and deliver it to whatever device the player is using. This feature is particularly useful with long playing games like world builders and story-driven games, where players want to resume gameplay at any time and from any device.
- Interact with remote game services, such as matchmaking, multiplayer game servers, and social networks, that require authenticated player IDs. Incorporate other cloud-based game features, such as for data validation and storage.
- Work with game management tools to collect and use data on player behavior, such as for game analytics, gameplay customization, live operations and experimentation.

Note

The identity and authentication feature must be added to your game project in order to use any other AWS GameKit feature, all of which rely on authentication of client requests to get or put player data.

Topics

How identity and authentication works

Summary

Use the identity and authentication game feature to build player sign-in workflows into your game, generate unique player IDs and use them to verify a player's identity during gameplay, manage access to player-specific data, and add other functionality that requires player authentication. Allow players to register with an email or a valid Facebook account. Primary audience: game owners and developers who want a high-level understanding of what the identity and authentication game feature delivers and the work required to build it into their games.

The primary mechanism of the identity and authentication game feature is the unique player identity. By registering with your game, players establish a verified identity and get a unique player ID for use with your game. When players sign in to your game, AWS GameKit uses this ID to authenticate the interactions between the game client that the player is using and your game backend.

Every AWS GameKit feature relies on identity and authentication to verify that player-specific requests coming from a game client are authenticated. These game features store player-specific data in the cloud by player ID, including gameplay data, achievements, and game state saves.

Identity and authentication with AWS GameKit uses simple, secure workflows. You can implement either or both of the following methods for registering players and establishing unique player IDs:

- Players can provide an email address and password. The sign-in workflow includes email verification, which the player must answer to complete registration. It also includes a password recovery workflow.
- Players can sign in using Facebook as a third-party identity provider. With this option, you must set up Facebook Login for your game. The sign-in workflow redirects players to Facebook where they enter their credentials. Facebook handles the authentication and then communicates the results back to your game.

If you choose to include both sign-in methods in your game, and players provide both types of credentials, the identity and authentication can connect both logins with the same player ID.

This game feature incorporates security features such as checks for compromised credentials and account takeover protection.

Identity and authentication with AWS GameKit does not offer special support for features such as authentication challenges, custom verification emails, or user directory management. However, you can add these features and more by manually customizing your AWS resources using AWS tools. Learn more about the AWS backend solution in Identity and authentication solution architecture.

Identity and authentication workflows

For player registration, AWS GameKit supports the following scenarios:

- Player signs in with an email/password. In this scenario, AWS GameKit automatically triggers a verification workflow, causing an email to be sent to the email address with a verification code. The player must get the verification code and enter it into a game UI to complete registration. On successful registration, a new unique player ID is created and their sign-in information is encrypted and stored.
- Player signs in with their Facebook account. In this scenario, AWS GameKit triggers a workflow that prompts the user to go to a Facebook web page to log in. If it is successful and the game's identity and authentication backend detects that this is the first time the player has signed in to the game, a new unique player ID is created.

The player registration and sign-in workflow is as follows:

1. In the game client, the player is presented with the option to create a new game account by either providing an email address or choosing a “Sign in with...” button and selecting an external identity provider to use.

2. If the player enters an email address and password:

    a. AWS GameKit sends a verification email to the provided email address with a session-based confirmation code.
    b. In the game client, the player enters the confirmation code and prompts the game to confirm the registration.
    c. If confirmation succeeds, AWS GameKit creates a new player record and returns an identity token, which can be used to authorize communication between the player's game client and the game's identity and authentication backend.

3. If the player opts to sign in with Facebook:

    a. The “Sign in with Facebook” button triggers the game to request a federated login URL for Facebook from the game's identity and authentication backend. The URL includes the game's account ID with Facebook.
    b. The game client opens the Facebook login URL in a browser, and the player logs in to their Facebook account.
    c. Facebook returns a login status. If successful, AWS GameKit creates a new player record and returns an identity token, which can be used to authorize communication between the player's game client and the game's identity and authentication backend.

4. In the game client, the player signs in to the game using their existing game account.
5. The game attempts to sign in the player. If the sign-in attempt is valid, AWS GameKit responds with a session-based token.

Topics

Identity and authentication solution architecture

This topic offers a detailed description of the AWS solution that provides cloud-based backend services to support the AWS GameKit identity and authentication feature. You don't have to master this information before using AWS GameKit to build the feature into your game and maintaining it. However, it is useful in gaining a deeper understanding of the AWS services and resources that are deployed for your game backend. You always have the option to view the backend components directly in AWS and use them with other AWS services, such as for monitoring or analytics. If you want to further customize or extend your game's backend services beyond what is available through AWS GameKit, you need to understand the role of each component in the solution.

The identity and authentication backend architecture implements the following call flow to authenticate an API request from a game client:

1. A game client calls an identity and authentication API operation, which prompts AWS GameKit to send a request to the Amazon API Gateway endpoint.
2. Amazon Cognito verifies the game client’s access token, if present. If the token is absent or invalid, the client is redirected to the sign-in page.
3. Game client authenticates with the player's sign-in credentials (username/password or Facebook sign-in) and receives a Amazon Cognito ID token.
4. Game client repeats the API request with the Amazon Cognito ID token.
5. Game client request is passed through to the relevant Lambda function, along with the now-validated Amazon Cognito ID token.

Identity and authentication services

All AWS GameKit solutions rely on a core set of AWS services, as described in Core services.

The following services are used to manage identity and authentication activity:

Amazon Cognito

AWS GameKit creates a Amazon Cognito user pool to manage player identities and authentication credentials. The user pool can be configured to accept an email/password or a variety of external identity providers, including Facebook. Amazon Cognito manages the sign-in verification and password recovery workflows.

AWS Lambda

AWS GameKit uses a Lambda function to manage the process of storing identity information in an Amazon DynamoDB table when a player successfully registers.

Amazon DynamoDB

AWS GameKit creates a DynamoDB table to track player identity information. For example, a player's username can be linked to both an email address and their account with an external identity provider.

Identity and authentication data encryption

Player data is encrypted both in transit and at rest.

In transit, AWS GameKit uses transport layer security (TLS) 1.2 or later for communication between a game frontend and backend components on AWS. All AWS GameKit game features use the Amazon API Gateway service to accept and process API calls. Learn more in the API Gateway Developer Guide, Data protection in transit.

At rest, player identity data is encrypted by the AWS services that the identity and authentication game feature uses. These services comply with industry standards. Learn more about how these services handle data encryption at rest:

- *Amazon Cognito Developer Guide*, Data protection in Amazon Cognito
- *Amazon DynamoDB Developer Guide*, DynamoDB encryption at rest

Identity and authentication configuration options

When configuring the identity and authentication feature for your game, you can customize the following characteristics. These customizations affect how the backend components for this feature are constructed.

- Enable/disable players to sign on with email and password.
- Enable/disable players to sign on with their Facebook account. To enable this option, you must set up a developer account with Facebook and get a Facebook App ID.

Identity and authentication callable actions

The AWS GameKit API provides the following actions for the identity and authentication game feature. After deploying AWS resources for the identity and authentication backend, your game frontend can use these calls to communicate with the backend.

- `Register()` Takes in a player's username, email address, and password to create a new player ID. After registering, players can use this information to log in to the game.
- `ConfirmRegistration` Verifies the registration confirmation code. When registering a new player ID with an email address, the player receives a verification email with a confirmation code, which they must enter into a game UI. This action verifies the entered code.

- `ResendConfirmationCode()` Allows player to request a fresh confirmation code to verify their email address. Confirmation codes are short-lived.
- `Login()` Signs the player into the game with their registered username and password.
- `GetFederatedLoginUrl()` Retrieves the URL of an external identity provider, such as Facebook, where the player can log in to their account. When this action is called for a player the first time, AWS GameKit registers the player and generates a new player ID.
- `GetUser()` Retrieves user information for a currently logged in player. This includes the players unique ID.
- `Logout()` Signs the player out of the game.
- `ForgotPassword()` For players who registered with an email/password, this action triggers a password recovery workflow.
- `ConfirmForgotPassword()` Verifies a confirmation code that allows a player to change their password. This is part of the password recovery workflow.

Identity and authentication estimate costs

The following table outlines the set of AWS services for the identity and authentication that may generate costs for this game feature. The conditions indicate the upper limits allowed within the Free Tier benefits for each service. Please note that some free-tier benefits are available for a limited time, while others are always free up to the usage limit.

Service	Condition	Free Tier Cost / Month
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,	0

Service	Condition	Free Tier Cost / Month
	confirmation, password reset, etc.)	
AWS Lambda (x86 architect ure)	Less than 1M requests per month and 400,000 GB-seconds of compute time per month (combined registrat ion, 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)	222 Facebook logins per day (less than 20,000 requests per month)	0

Add identity and authentication to your project

Summary

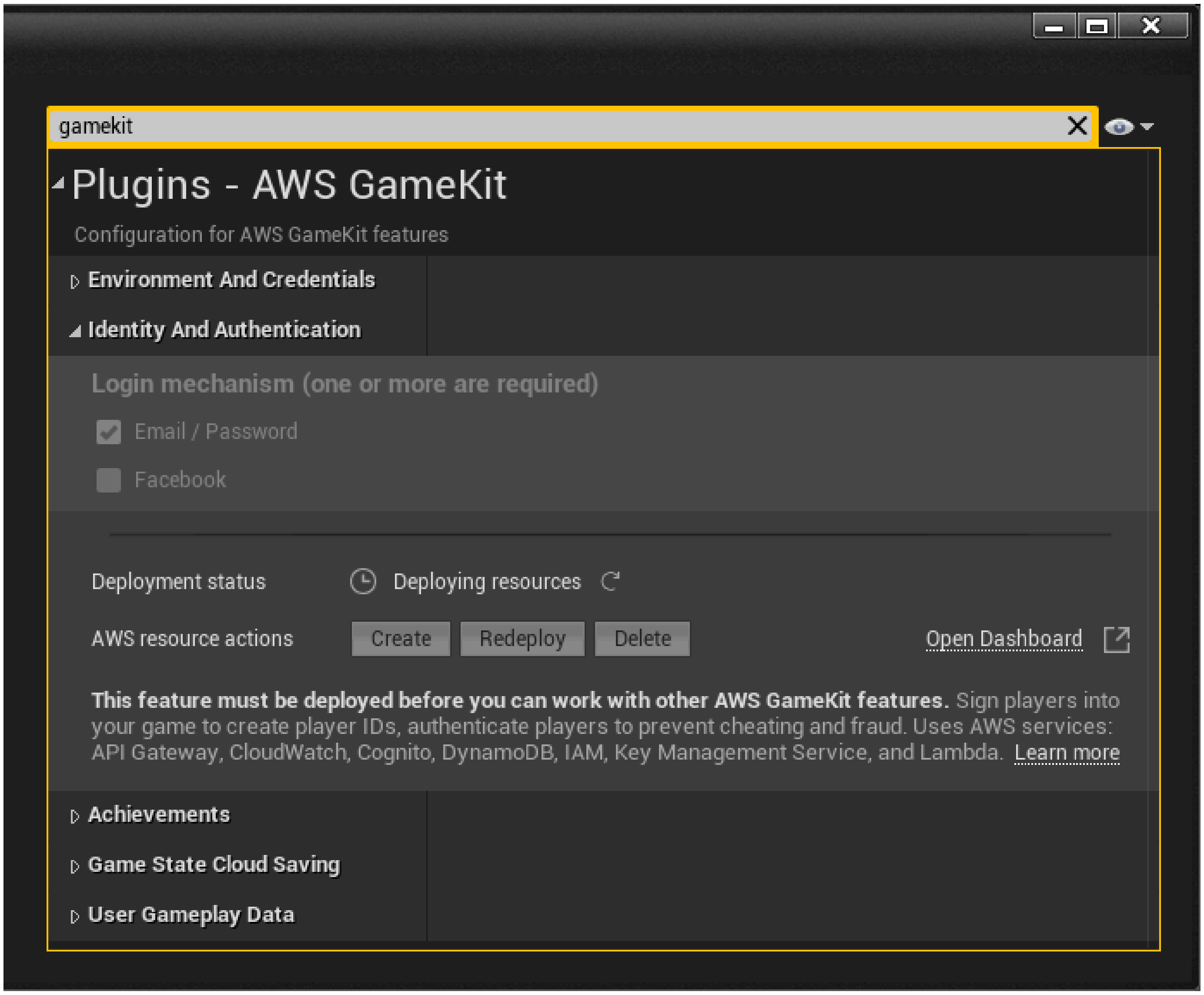
Learn how to build a complete cloud-based identity and authentication system and integrate it into a Unreal Engine project. This topic guides developers through building backend services and then using the AWS GameKit API to connect frontend code to the backend on AWS.

When you're ready to build out the identity and authentication feature for your game, follow these basic steps. If you don't yet have AWS GameKit installed for your project, see Install the AWS GameKit plugin with Unreal Engine and Set up the AWS GameKit plugin for your game.

Step 1. Configure the identity and authentication game feature.

1. In the Unreal Editor toolbar, open **Edit, Project Settings** and go to the **AwsGameKit** plugin section. Select the environment you want to work in and enter valid AWS credentials as needed. For more details, see Set up the AWS GameKit plugin for your game.
2. Expand the **Identity and authentication** section. Specify the following configuration options:
   - Login mechanisms. Choose which login mechanisms that you want to offer to your players. The default selection is Email/Password, and you must have at least one option selected. If you choose to support player login with Facebook, enter the Facebook app ID and secret key for your game. Learn more about getting a Facebook app ID on the Facebook Developers portal.

   All changes you make to feature settings are cached locally during the current session.



Login mechanism (one or more are required)

✓ Email / Password

Facebook

Deployment status: Deploying resources

AWS resource actions

This feature must be deployed before you can work with other AWS GameKit features. Sign players into your game to create player IDs, authenticate players to prevent cheating and fraud. Uses AWS services: API Gateway, CloudWatch, Cognito, DynamoDB, IAM, Key Management Service, and Lambda.

Step 2. Deploy AWS resources for your identity and authentication backend.

1. While still in the AWS GameKit settings for **Identity and Authentication**, scroll down to the deployment controls. Choose the AWS resource action **Create**. This action prompts AWS GameKit to deploy the complete AWS solution to run the backend services for this game feature. When deploying, AWS GameKit connects to AWS to create all the AWS resources for the solution, using the configuration template maintained for the active environment.
2. Deploying resources for identity and authentication typically takes 5 minutes to complete. During this time, the feature's deployment status reads **Deploying resources**. You can track the progress of your deployment status:

   - In the Unreal Editor, open the output log window to monitor status messages, events, and errors throughout the deployment.
   - In the AWS Management Console, open the AWS CloudFormation service. In the **Stacks** view you can watch as the Identity stack for your game project is deployed.

   When deployment is complete, the feature's deployment status reads **Deployed**. Your game backend for identity and authentication is in place. You can make calls to it using the AWS GameKit API.

   Note

   NOTE: From this point on, you might begin incurring costs for this game feature. If you're still in the AWS Free Tier window, you will only incur costs if you exceed free tier limits.

Step 3. Add player identity workflows to your game.

Create UI elements and add code for the following workflows. See the plugin's Identity examples for illustration and experiment with API calls using the C++ code examples.

Identity and authentication workflows include:

- Register a new player with email
  - `Register()`
  - `ConfirmRegistration()`

- `ResendConfirmationCode()`
- Sign in a player with an existing email account
  - `Login()`
  - `GetUser()` and `GetResponseBody()`
  - `Logout()`
- Sign in a new or existing player with Facebook
  - `GetFederatedLoginUrl()`
  - `PollAndRetrieveFederatedTokensAsync()` or `PollAndRetrieveFederatedTokensBlueprintAsync()`
  - `GetFederatedAccessToken()`
  - `Logout()`
- Reset the password for an existing account
  - `ForgotPassword()`
  - `ConfirmForgotPassword()`

Work with the identity and authentication examples

The AWS GameKit plugin includes example assets for the identity and authentication game feature.

Unreal Engine

The AWS GameKit plugin for Unreal Engine provides examples with C++ code, blueprints, and UI components. You can access the example files in the Unreal Editor content browser.

- Work with the C++ examples
- Work with the Blueprint and UI examples

Work with the C++ examples

In the Unreal Editor content browser, find the example asset at the following location:

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

This asset is a `.cpp` file. You can also find this file in the AWS GameKit plugin files, located at `...\AwsGameKit\Source\AwsGameKitEditor\Private\Identity\AwsGameKitIdentityExamples.cpp`.

This example file contains sample code that illustrates how to call each of the identity and authentication API actions. This file contains the basic set of runnable code and includes detailed comments.

You can work with this example in two ways: view the code in your IDE, or experiment with the API calls in Unreal Editor.

To view or edit the example code:

- Double-click the **AwsGameKitIdentityExamples** asset to open the file in your IDE. You do not need to enter AWS credentials or deploy AWS resources before accessing this file, however none of the API calls can work without deployed resources.

  The example code includes a standard check to verify that AWS resources are deployed.

  Note that, as a first step, the example code creates an instance of the `UAwsGameKitIdentityCallableWrapper` and initializes it. This must be done before making any API calls.

To experiment with the example in Unreal Editor:

You must have AWS resources deployed for identity and authentication and you must submit your valid AWS credentials in the AWS GameKit plugin project settings (see Set up the AWS GameKit plugin for your game).

1. Drag the **AwsGameKitIdentityExamples** asset into level in the view port. It doesn't matter what level you add the asset to, this is just a mechanism that enables you to work with the asset settings.
2. In the Editor's **Details** pane, all of the identity and authentication API calls are available with API request and response values.
3. To make an API call, enter some input values and click the "Call" button. The response is displayed in the **Return Value** field.
4. Try running the following call sequences to simulate standard identity scenarios:

   - Register a new player with email: Register Player, Confirm Email

- Sign in a player with an existing account: Login, Get User, Logout
- Sign in a new or existing player with Facebook: Open Facebook Login, Get User, Logout
- Recover a password for an existing account: Forgot Password, Confirm Forgot Password

Work with the Blueprint and UI examples

In the Unreal Editor **Content Browser**, find the example assets at the following location:

```
AwsGameKit Content > Identity >
```

There are two example assets:

- **BP_AwsGameKitIdentityExamples**

  This asset is a basic blueprint that illustrates how you might add sign-in functionality to your game code.

  Actions:
  - To open the blueprint, double-click the asset.
- **BP_AwsGameKitIdentityExamplesUI**

  This asset includes a detailed blueprint for common sign-in workflow scenarios and sample UI objects.

  Actions:
  - To open the blueprint, double-click the asset.

  To run the example, click **Play**.

AWS GameKit feature: User gameplay data

The AWS GameKit game feature user gameplay data provides tools for storing a player's game-related data in the cloud. With this feature, gameplay data can persisted across game sessions and can be synchronized across multiple game clients and devices.

This game feature is designed to handle data synchronization on the fly during gameplay. It is not designed for other player data, such as for player profiles or account information. User gameplay data might include information such as player scores, inventory, or other player-associated data that is needed in the game. It does not include non-game player data, such as profile or account information.

Potential uses for this game feature include:

- Game inventories
- Game economy/currency
- Player statistics
- Player game choices, such as a character or in-game decisions that impact game progress
- Level activity, such as visited locations

Topics

How user gameplay data works

Summary

This topic provides a high-level description of the game feature and describes common scenarios for how the feature might be integrated into a game.

This game feature offers a very flexible data schema. You decide how you want to organize data. Briefly, gameplay data is saved as a set of key:value pairs, called bundle items. A bundle is a construct that you can use to group related bundle items.

The basic schema structure looks like this:

- Bundle name. Bundle is construct that lets you create collections of related bundle items. For example, you might want to group data for the player's weapons inventory in a bundle.
  - Bundle item. A single piece of data, consisting of a data identifier (key) and saved value. It's up to you to decide what data items you need to save. For example, for a weapons inventory, you probably need to track at minimum (1) whether the weapon is available, (2) amount of ammunition, (3) modifications, and (4) if a shortcut key is assigned to it. Each of these pieces of information would be stored as a bundle item.
    - Bundle item key. A unique item name or ID.
    - Bundle item value. A value

For example:

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

There are no limits on the number of bundles each player can store, and no limit on the number of items in a bundle. Since you pay for the amount of data storage you use, this is purely up to you.

This game feature is designed to support frequent reads or writes of game data. Developers can easily set and get data items in real time.

This game feature has additional protection to maintain data integrity in case of connectivity loss. It maintains a message queue for all requests that can't be delivered. When no connection is available, all add/update/delete requests are automatically placed in the message queue. When a connection is restored, the requests are processed in the order they were queued (oldest first). If multiple update requests occur for the same bundle item while requests are being queued, newer requests automatically replace the older requests, so only one write request is processed for the bundle item. Note that get requests aren't queued.

User gameplay data solution architecture

This topic offers a detailed description of the AWS solution that provides cloud-based backend services to support the AWS GameKit user gameplay data feature. You don't have to master this information before using AWS GameKit to build the feature into your game and maintaining it. However, it is useful in gaining a deeper understanding of the AWS services and resources that are deployed for your game backend. You always have the option to view the backend components directly in AWS and use them with other AWS services, such as for monitoring or analytics. If you want to further customize or extend your game's backend services beyond what is available through AWS GameKit, you will need to understand the role of each component in the solution.

The workflow sequence is as follows:

1. A game client calls a user gameplay data API operation, which prompts AWS GameKit to send a request to the API Gateway endpoint. Amazon Cognito verifies the game client’s access token, as described in Identity and authentication solution architecture. If the request involves player achievement data, an Amazon Cognito authorizer verifies that the access token is valid for the player as defined in the user pool.
2. If authentication is successful, the game client request is passed through to the relevant Lambda function.
3. The Lambda function interacts with DynamoDB to store or retrieve data on bundles or bundle items as requested. There are two DynamoDB tables, one to track bundles, and one to track bundle items.

User gameplay data services

All AWS GameKit solutions rely on a core set of AWS services, as described in Core services.

These services are used to manage user gameplay data activity:

AWS Lambda

AWS GameKit uses Lambda functions to manage gameplay data storage and retrieval.

Amazon DynamoDB

AWS GameKit creates two types of DynamoDB tables, one to store bundle names, and a second one to store bundle items and values by player and bundle name.

User gameplay data encryption

Player data is encrypted both in transit and at rest.

In transit, AWS GameKit uses transport layer security (TLS) 1.2 or later for communication between a game frontend and backend components on AWS. All AWS GameKit game features use the Amazon API Gateway service to accept and process API calls. Learn more in the API Gateway Developer Guide, Data protection in transit.

At rest, data is encrypted by the AWS services that the user gameplay data game feature uses. These services comply with industry standards. Learn more about how these services handle data encryption at rest:

- *Amazon DynamoDB Developer Guide*, DynamoDB encryption at rest

User gameplay data callable actions

The AWS GameKit API provides the following actions for the user gameplay data game feature. After deploying AWS resources for the user gameplay data backend, your game frontend can use these calls to communicate with the backend.

- `AddUserGameplayData()`: Use this call to create or update a bundle. You can call this with a bundle name only, or include one or more bundle items. If a bundle item key already exists in the cloud, the value is updated.
- `GetAllUserGameplayData()`: Retrieves all data bundles that are saved for the player ID. Bundle item data is returned with data organized by bundle name.
- `GetUserGameplayDataBundle()`: Retrieves a player's bundle items in a specified bundle. Data is returned as a map of key:value pairs.
- `GetUserGameplayDataBundleItem()`: Retrieves a single bundle item for a player, when a bundle name and bundle item key is specified. Data is returned as a single key:value pair.
- `UpdateUserGameplayDataBundleItem()`: Adds or updates a single bundle item in a bundle.
- `DeleteUserGameplayDataBundleItem()`: Deletes a single bundle item in a specified bundle.
- `DeleteUserGameplayDataBundle()`: Deletes a specified bundle, including all of the bundle items that it contains.
- `DeleteAllUserGameplayData()`: Deletes all bundles and bundle items for a player.

Add user gameplay data to your game

The following describes the basic steps to add a user gameplay data game feature to your Unreal Engine project. If you don't yet have AWS GameKit for your game engine, see Install the AWS GameKit plugin with Unreal Engine.

Build your User Gameplay Data feature

Step 1. Deploy an AWS solution for your user gameplay data backend.

1. In the Unreal Editor toolbar, open **Edit, Project Settings** and go to the **AwsGameKit** plugin section. Select the environment you want to work in and enter valid AWS credentials as needed. For more details, see Set up the AWS GameKit plugin for your game.
2. Open the **User Gameplay Data** section. For this game feature, there are no special configuration settings.
3. Choose the AWS resource action **Create**. This action prompts AWS GameKit to deploy a complete AWS solution to run the backend services for this game feature. When deploying, AWS GameKit first generates a template for the AWS solution, including your custom configuration settings, and then connects to AWS to create the solution's AWS resources as defined in the template. The AWS resources are deployed to the AWS region as selected for the active environment.
4. Deploying resources for the user gameplay data backend typically takes 5 minutes to complete. You can track the progress of your deployment status as follows:

   - In the **Features** tab, scroll down to view the Log window. The log displays status messages, events, and errors throughout the deployment.
   - In the AWS Management console, open the AWS CloudFormation service. In the Stacks view you can watch as the UserGameplayData stack for your game project is deployed.
   -

   When deployment is complete, your game backend for the user gameplay data game feature is in place. You can make calls to it using the AWS GameKit API.

Note

NOTE: From this point on you may begin incurring cost, depending on whether you’re in the AWS Free Tier window.

Step 3. Add gameplay data functionality to your game.

1. Before calling any of the user gameplay data APIs, invoke `GdkUserGameplayDataCreate` and call `Initialize()`.
2. Add code to save gameplay data to your cloud backend and keep it synchronized with the game client. As part of this work, develop a gameplay data schema for your game, using bundles to group data items into collections as needed. Determine where and when to create bundles, add bundle items, and update item values.
3. Add workflows such as:
   - Create a bundle with a collection of data items
     - `AddUserGameplayData()`
   - Update a bundle item with a new value
     - `UpdateUserGameplayDataBundleItem()`
   - Retrieve a bundled collection of data items
     - `GetUserGameplayDataBundle()`
   - Retrieve a single bundle item value
     - `GetUserGameplayDataBundleItem()`
   - Delete a bundle
     - `DeleteUserGameplayDataBundle()`

Integration tips

Consider these issues when adding gameplay data functionality to your game.

- Consider making add, update, and delete API calls asynchronously. Get calls can be made synchronously.

- On the game client, you may want to add some data validation and sanity checks to protect against tampering.

Work with the user gameplay data examples

The AWS GameKit plugin includes example assets for the user gameplay data game feature.

Unreal Engine

The AWS GameKit plugin for Unreal Engine provides examples with C++ code, blueprints, and UI components. You can access the example files in the Unreal Editor content browser.

- Work with the C++ examples
- Work with the Blueprint and UI examples

Work with the C++ examples

In the Unreal Editor content browser, find the example asset at the following location:

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

This asset is a `.cpp` file. You can also find this file in the AWS GameKit plugin files, located at `...\AwsGameKit\Source\AwsGameKitEditor\Private\UserGamePlayData\AwsGameKitUserGamePlayDataExamples.cpp`.

This example file contains sample code that illustrates how to call each of the user gameplay data API actions. This file contains the basic set of runnable code and includes detailed comments.

You can work with this example in two ways: view the code in your IDE, or experiment with the API calls in Unreal Editor.

To view or edit the example code:

- Double-click the **AwsGameKitUserGamePlayDataExamples** asset to open the file in your IDE. You do not need to enter AWS credentials or deploy AWS resources before accessing this file, however none of the API calls can work without deployed resources.

The example code includes a standard check to verify that AWS resources are deployed.

Note that, as a first step, the example code creates an instance of the `UAwsGameKitUserGamePlayDataCallableWrapper` and initializes it. This must be done before making any API calls.

To experiment with the example in Unreal Editor:

You must have AWS resources deployed for identity and authentication and you must submit your valid AWS credentials in the AWS GameKit plugin project settings (see Set up the AWS GameKit plugin for your game).

1. Drag the **AwsGameKitUserGamePlayDataExamples** asset into level in the view port. It doesn't matter what level you add the asset to, this is just a mechanism that enables you to work with the asset settings.
2. In the Editor's **Details** pane, all of the user gameplay data API calls are available, with API request and response values. Calls made using this UI connect to your deployed game backend on AWS.
3. To make an API call, enter some input values and click the "Call" button. The response is displayed in the **Return Value** field.
4. Try running the following call sequences to simulate standard gameplay data scenarios:
    - Create a bundle with one or more bundle items: Add User Gameplay Data
    - Update the value for an existing bundle item: Update User Gameplay Data Bundle Item
    - Retrieve all item values in the bundle: Get User Gameplay Data Bundle
    - Retrieve a single bundle item value: Get User Gameplay Data Bundle Item
    - Delete the bundle: Delete User Gameplay Data Bundle

Work with the Blueprint and UI examples

In the Unreal Editor **Content Browser**, find the example assets at the following location:

```
AwsGameKit Content > UserGameplayData >
```

There are two example assets:

- **BP_AwsGameKitUserGameplayDataExamples**

  This asset is a basic blueprint that illustrates how you might add gameplay data save and retrieval functionality to your game code.

  Actions:

  - To open the blueprint, double-click the asset.
- **BP_AwsGameKitUserGameplayDataExamplesUI**

  This asset includes a detailed blueprint for common workflow scenarios and sample UI objects.

  Actions:

  - To open the blueprint, double-click the asset.

  To run the example, click **Play**.

AWS GameKit feature: Game state cloud saving

The AWS GameKit game state cloud saving feature enables players to save their game state, store it in the cloud, and keep it synchronized with their local game clients. Game saving is a highly valued game feature for players, particularly with games with very long storylines. Cloud saving and synchronization offers additional benefits to players, including:

- For games that support more than one game platform, players can get true crossplay portability on multiple devices.
- Players can recover game progress with minimal loss in the event of local device failures.
- Players still have access to past played game saves, even for games that have been uninstalled.

Use this game feature to implement an autosave system as well for explicit saves that allow players to choose their save points.

Topics

How game state cloud saving works

This game feature handles two primary tasks: first, it stores a player's game save files in the cloud, and second, it synchronizes local and cloud versions of each game save file. Synchronization ensures that the player is always playing with the latest version of a game save, even when they're playing the same game across multiple devices.

Storing game save files in the cloud

Players game states are saved as files, which can then be uploaded into file storage in the cloud. In this feature, game save files are managed as slots. Each slot has metadata attached, including a save name, a last-modified date stamp, and, optionally, a string of developer-defined metadata formatted as JSON.

Synchronizing game save files

Synchronization ensures that updates to a game save file happen in the right sequence, regardless of how many devices a player uses to play your game, making the player's experience seamless.

This feature tracks five potential states and any actions that should be taken to synchronize them:

- Local only: The game save file only exists on the local device. Action: upload the local game save file to the cloud.
- Cloud only: The game save file only exists on the cloud. This state might occur when the player is resuming their game on a new device. Action: download the cloud game save file to the local device.
- Local/Cloud in sync: The game save file exists both locally and on the cloud, and the last modified timestamps match. No action required.
- Local most recent: The game save file exists both locally and on the cloud, but the local version has a more recent modified date. This state occurs when a player has played the game on the local device and wants to save the game with their progress. Action: upload the latest local version of the game save file to the slot.
- Cloud most recent: The game save file exists both locally and on the cloud, but the cloud version has a more recent modified date. This state can occur when a player has recently played the game on another device, saved their game progress there, and now wants to resume their game on the current device. Action: download the latest cloud version of the game save file to the slot.

Game state cloud saving workflow

Here's a high level description of how the game state cloud saving system works from the player perspective.

Saving game state as a player:

1. In the game, a player takes an action to save their game.
2. In response to the event, the game client makes an API call to save the current game state file to the cloud.

3. When the game backend receives the request, it stores the game save file and corresponding metadata, including a timestamp.
4. When the player restarts the game, sync status is automatically requested. AWS GameKit compares the game save timestamps and tries to determine whether the local or cloud versions are most recent.

Game state cloud saving solution architecture

This topic offers a detailed description of the AWS solution that provides cloud-based backend services to support the AWS GameKit game state cloud saving feature. You don't have to master this information before using AWS GameKit to build the feature into your game and maintaining it. However, it is useful in gaining a deeper understanding of the AWS services and resources that are deployed for your game backend. You always have the option to view the backend components directly in AWS and use them with other AWS services, such as for monitoring or analytics. If you want to further customize or extend your game's backend services beyond what is available through AWS GameKit, you will need to understand the role of each component in the solution.

This solution uses a Lambda function to compare the metadata for the local and cloud versions of a save file. To synchronize the files, the comparison prompts one of the two call flows:

Download the cloud save state:

1. AWS GameKit identifies that the local save state is older than the cloud save state, and requests a pre-signed URL (GET) be generated for its current slot.
2. The Lambda function composes the URL from a combination of the `BucketName`, `PlayerID` and `SlotName`, then generates a pre-signed URL that AWS GameKit can use to fetch the save data.
3. AWS GameKit fetches the save file from S3 using the provided URL via a HTTP GET request

Upload the local save state to the cloud:

1. AWS GameKit identifies that the cloud save state is older than the local save state, and requests a pre-signed URL (PUT) be generated for its current save slot.
2. AWS GameKit pushes the save data to S3 via a HTTP PUT request.
3. After the file has been uploaded, a Lambda function is instantiated and is passed the metadata for that object.

4. The Lambda function looks up the object that has been stored in S3, using a combination of `PlayerID` and `SlotName`, and updates all attributes.

Game state cloud saving services

All AWS GameKit solutions rely on a core set of AWS services, as described in Core services.

The following services are used specifically to manage game state cloud saving activity:

Amazon Simple Storage Service (Amazon S3)

AWS GameKit uses an Amazon S3 bucket to store game save files. Amazon S3 provides durable object storage capabilities.

Amazon DynamoDB

AWS GameKit uses a DynamoDB table to store metadata about the game save files. Using DynamoDB to store this type of data supports frequent read and write requests from game clients.

Lambda

AWS GameKit uses Lambda functions to manage the tasks of storing game save files and metadata and analyzing synchronization states.

Game state cloud saving data encryption

Player data is encrypted both in transit and at rest.

In transit, AWS GameKit uses transport layer security (TLS) 1.2 or later for communication between a game frontend and backend components on AWS. All AWS GameKit game features use the Amazon API Gateway service to accept and process API calls. Learn more in the API Gateway Developer Guide, Data protection in transit.

At rest, player identity data is encrypted by the AWS services that the Achievement game feature uses. These services comply with industry standards. Learn more about how these services handle data encryption at rest: [Question: S3 has some additional server-side encryption available with managed keys (SSE-S3) and KMS. Does our solution cover any of this? If so we should probably mention it here.]

- *Amazon DynamoDB Developer Guide*, DynamoDB encryption at rest

- *Amazon S3 User Guide*, [Protecting data using encryption]

Game state cloud saving configuration options

When configuring the game state cloud saving feature for your game, you can customize the following characteristics. These customizations affect how the backend components for this feature are constructed.

- Slot limit: This value sets specifies the number of cloud game save files that players can use in your game. There is no hard limit on this option. The number of allowed cloud game saves in a game does impact the cost of this game feature, which is dependent on the amount of storage space used. (Required)

Game state cloud saving callable actions

The AWS GameKit API provides the following primary actions for the game state cloud saving game feature. After you deploy AWS resources for game state cloud saving, your game frontend can use these calls to communicate with the backend resources.

Each of these actions can be called either synchronously or asynchronously. Learn more about each action in the AWS GameKit Core C++ API Reference.

- `GetAllSlotSyncStatuses()` determines the current synchronization status of all game save slots that exist locally and/or in the cloud. This is done by comparing the last modified date of both local and cloud versions. Sync status indicates (1) that the game save slot is in sync, or (2) a recommended action to synchronize the versions.
- `GetSlotSyncStatus` determines the current synchronization status of a specified game save slot. This is done by comparing the last modified date of both local and cloud versions. Sync status indicates (1) that the game save slot is in sync, or (2) a recommended action to synchronize the versions.
- `SaveSlot` uploads a local game save file to the cloud. This action is used to create a new game save slot or to update an existing slot. This action saves the slot metadata locally as well as in the cloud.
- `LoadSlot` downloads a cloud-stored game save file to the local device and updates local slot metadata. This action is used to sync an existing local slot with the cloud version or, if none exists, to create a new local slot.

- `DeleteSlot` deletes the cloud-stored game save file and all slot metadata both locally and in the cloud. It does not delete the local version of the game save file.

Add game state cloud saving to your game

The following describes the basic steps to add a game state cloud saving game feature to your Unreal Engine project. If you don't yet have AWS GameKit for your game engine, see Install the AWS GameKit plugin with Unreal Engine.

Step 1. Configure the game state cloud saving game feature for your project.

1. In the Unreal Editor toolbar, open **Edit, Project Settings** and go to the **AwsGameKit** plugin section. Select the environment you want to work in and enter valid AWS credentials as needed. For more details, see Set up the AWS GameKit plugin for your game.
2. Open the **Game State Cloud Saving** section. Specify the following configuration options:
   - **Maximum save slots.** Specify the maximum number of game save slots that you want to allow for each player.

Step 2. Deploy AWS resources for your game state cloud saving backend.

1. In the AWS GameKit project settings under **Game State Cloud Saving**, choose the AWS resource action **Create**. This action prompts AWS GameKit to deploy a complete AWS solution to run the backend services for this game feature. When deploying, AWS GameKit first generates a template for the AWS solution, including your custom configuration settings, and then connects to AWS to create the solution's AWS resources as defined in the template. The AWS resources are deployed to the AWS region as selected for the active environment.
2. Deploying resources for the game state cloud saving backend typically takes 5 minutes to complete. You can track the progress of your deployment status as follows:
   - In the Unreal Editor, AWS GameKit project settings, refresh the deployment status or open the custom dashboard for this game feature. These dashboards are generated at the beginning of the deployment process.
   - In the Unreal Editor, open the output logs at **Window>Developer Tools, Output Log** to view status messages, events and errors throughout the deployment.
   - In the AWS Management console, open the AWS CloudFormation service. In the Stacks view you can watch as the game state cloud saving stacks for your game project are deployed.

When deployment is complete, your game backend for the game state cloud saving game feature is in place. You can make calls to it using the AWS GameKit API.

> **Note**
>
> NOTE: From this point on you may begin incurring costs for this game feature, depending on whether you're in the AWS Free Tier window.

Step 3. Add game state cloud saving functionality to your game.

Create UI elements and add code for the workflows to save and retrieve game states as needed for your game. See the plugin's game state cloud saving examples for illustration.

Workflows might include:

- Upload a game save file to the cloud: Save file.
- Check sync status for all slots: Get All Sync Statuses.
- Download a game save file from the cloud: Load file.
- Delete a game save file in the cloud: Delete slot.

Work with the game state cloud saving examples

The AWS GameKit plugin includes example assets for the game state cloud saving game feature.

Unreal Engine

The AWS GameKit plugin for Unreal Engine provides examples with C++ code, blueprints, and UI components. You can access the example files in the Unreal Editor content browser.

- Work with the C++ examples
- Work with the Blueprint and UI examples

Work with the C++ examples

In the Unreal Editor content browser, find the example asset at the following location:

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

This asset is a `.cpp` file. You can also find this file in the AWS GameKit plugin files, located at `...\AwsGameKit\Source\AwsGameKitEditor\Private\GameStateCloudSaving\AwsGameKitGameStateCloudSavingExamples.cpp`.

This example file contains sample code that illustrates how to call each of the game state cloud saving API actions. This file contains the basic set of runnable code and includes detailed comments.

You can work with this example in two ways: view the code in your IDE, or experiment with the API calls in Unreal Editor.

To view or edit the example code:

- Double-click the **AwsGameKitGameStateCloudSavingExamples** asset to open the file in your IDE. You do not need to enter AWS credentials or deploy AWS resources before accessing this file, however none of the API calls can work without deployed resources.

  The example code includes a standard check to verify that AWS resources are deployed.

  Note that, as a first step, the example code creates an instance of the `UAwsGameKitGameStateCloudSavingCallableWrapper` and initializes it. This must be done before making any API calls.

To experiment with the example in Unreal Editor:

You must have AWS resources deployed for identity and authentication and you must submit your valid AWS credentials in the AWS GameKit plugin project settings (see Set up the AWS GameKit plugin for your game).

1. Drag the **AwsGameKitGameStateCloudSavingExamples** asset into a level in the view port. It doesn't matter what level you add the asset to, this is just a mechanism that enables you to work with the game feature configuration.
2. In the Editor's **Details** pane, all of the game state cloud saving API calls are available with API request and response values.

3. To make an API call, enter some input values and choose the call button. The response is displayed in the **Return Value** field.
4. Try running the following call sequences to simulate standard game state cloud saving scenarios:

    - Upload a game save file to the cloud: Save file.
    - Check sync status for all slots: Get All Sync Statuses.
    - Download a game save file from the cloud: Load file.
    - Delete a game save file in the cloud: Delete slot.

Work with the Blueprint and UI examples

In the Unreal Editor **Content Browser**, find the example assets at the following location:

```
AwsGameKit Content > GameStateCloudSaving >
```

There are two example assets:

- **BP_AwsGameKitGameStateCloudSavingExamples**

  This asset is a basic blueprint that illustrates how you might add game-state-related functionality to your game code.

  Actions:
  - To open the blueprint, double-click the asset.
- **BP_AwsGameKitGameStateCloudSavingExampleUI**

  This asset includes a blueprint and sample UI objects for displaying achievement information and player status.

  Actions:
  - To open the blueprint, double-click the asset.

  To run the example, click **Play**.

AWS GameKit feature: Achievements

The AWS GameKit achievements feature offers tools to manage an achievements system for your game. Achievements are a proven way to boost player engagement; they invite players to explore certain aspecs of the game and en . Achievements might mark a specific player action or might track a player's progress toward a multi-step goal. Achievements might offer bragging rights only or they can deliver more tangible rewards.

Some common ways that achievements are used in games:

- Mark game milestones and track player progress. A milestone like "Capture 100 mutant monkeys" might represent a key game objective. Alternatively, progressive achievements ("Congrats, you've achieved Cerulean Mage status!") boost a player's sense of ongoing accomplishment as they move through the game.
- Set clear and achieveable goals to keep players engaged and avoid getting lost, bored, or frustrated. This type of achievement is very useful with non-linear stories and open worlds. Achievements like "Explore the forest" or "Talk to the ginger cat" offer cues to help guide players to meaningful gameplay and breach obstacles.
- Help players understand the meaning and value of objects, actions, and other concepts in the game world. Achievements like "Complete your first 10 trades to earn negotiator badge" or "Acquire a bicycle with 5 handcrafted tools or 10 found tools" offer valuable clues to players about what matters in the game.
- Prompt players to learn new skills and upgrade techniques. An achievement like "Win a battle in less than six minutes" might prod a player to explore new skills and powers or refine their technique in ways that they'll need later in the game.
- Challenge players with optional hard-to-earn achievements. Challenges like "Complete the scavenger hunt side mission" or "Don't kill anyone" give players valuable optional experiences and can even fundamentally change how the game is played. These types of achievements can boost a game's replay value.

Note

To add this feature, you must also have the AWS GameKit identity and authentication feature. This feature relies on identity management to manage each player's achievement status.

Topics

How achievements work

When deployed, your AWS GameKit achievements backend has three primary functions:

- It stores the definitions for all the achievements in your game. Definitions include information such as the achievement name, how it's earned, an optional reward, and player-facing details such as icons or messaging. You use the AWS GameKit tool to create achievement definitions and sync them to your achievements backend. With achievements saved to the backend, players can start to earn them. You can add or update achievement definitions at any time. Game clients can use the AWS GameKit API to request a list of achievement definitions.
- It stores each player's achievements status. When a player does something in your game that earns them progress toward an achievement, your game client sends an AWS GameKit API request to the backend to update the player's status. The game client can also request player achievement status, including earned awards and progress toward multi-step awards.
- It runs logic to evaluate if a player has earned achievements. Using the achievement definitions and player status, the backend uses a Lambda function to determine if a player has completed the achievement requirements. In addition to tracking player status, earned status can affect how the backend responds to game client requests for achievement information.

Achievement types

The AWS GameKit achievements game feature supports the following achievement types:

Single-step achievement

Also called a stateless achievement, a player earns this type in response to a single specific event in gameplay, such as when they find a certain treasure item or unlocks a puzzle. Your game client responds to the event by sending an update request to your achievements backend Player status for stateless achievements is either locked (not earned) or unlocked (earned).

Multi-step achievement

Also called a stateful achievement, a player earns this type in response to a series of events in gameplay. A stateful achievement definition specifies the number of steps that are required to earn the achievement. When an event occurs, the game client sends an update request, with a number, to the achievements backend, and the backend service responds by adding that number to the player's current progress until the step requirement is reached and the player earns the achievement. For example, the achievement "eat 1000 bananas" has a step value of 1000. Each time a player eats a banana, the game client notifies the backend with a value of "1", and the backend increments the player's status by 1 until it reaches 1000. With this achievement type, you have the flexibility to design how you want to increment player progress. You might add banana bunches that increment by 10 or mega-bunches that increment by 100. Player status for a stateful achievement is locked and not started, locked and in progress, or unlocked.

Secret achievement

This type is useful when you want to define achievements but not reveal them to players until after they've earned them. For example, with a secret achievement such as "Expose the Nazi mole before they defect", you can reward players without giving the end game away. The achievements backend does not include secret achievements when responding to a game client's request for achievement information until the player has earned them. You can add or remove an achievement’s "secret" flag at any time. Secret achievements can be stateless or stateful.

Hidden achievement

This type allows you to hide a defined achievement from your game. This is useful when you want to stage an achievement for future release, such as for a special event. When responding to requests from a game client, the achievements backend does not return information on hidden achievements and does not track player status for them. You can add or remove an achievement’s "hidden" flag at any time. Hidden achievements can be stateless or stateful.

Achievements workflow

Here's a high level description of how the achievement system works from the player/game client perspective.

1. In your game, a player does something that fires an event. In response to the event, your game client calls an AWS GameKit API operation to report player progress toward the achievement. This call contains the following information: player ID, name of the achievement, and a number value that represents what the player has done, such as eat ten bananas or slay a dragon.
2. When the achievements backend receives the request, it updates the player's status for the specified achievement. It then runs some built-in logic to determine if the player has earned the achievement. This involves comparing the goal (as specified in the achievement definition) to the player's current progress. If the player has reached the goal, the achievement is unlocked.
3. The game client can call AWS GameKit API operations to get a player's status for a single achievement or all achievements, and displays updated information to the player.

Achievements solution architecture

This topic offers a detailed description of the AWS solution that provides cloud-based backend services to support the AWS GameKit achievements feature. You don't have to master this information before using AWS GameKit to build the feature into your game and maintaining it. However, it is useful in gaining a deeper understanding of the AWS services and resources that are deployed for your game backend. You always have the option to view the backend components directly in AWS and use them with other AWS services, such as for monitoring or analytics. If you want to further customize or extend your game's backend services beyond what is available through AWS GameKit, you need to understand the role of each component in the solution.

The achievements backend architecture manages two call flows:

- Call flow to manage achievement definitions. This flow is used in the AWS GameKit plugin to configure the set of achievements that players can earn in the game.

- Call flow to manage player-related achievement actions and statuses.

The workflow sequence is similar for both flows:

1. A game client calls an achievements API operation, which prompts AWS GameKit to send a request to the API Gateway endpoint. Amazon Cognito verifies the game client's access token, as described in Identity and authentication solution architecture. If the request involves player achievement data, an Amazon Cognito authorizer verifies that the access token is valid for the player as defined in the user pool.
2. If authentication is successful, the game client request is passed through to the relevant Lambda function.
3. The Lambda function interacts with DynamoDB to store or retrieve data as requested. Achievements definitions and player-related data are stored in two separate DynamoDB tables, one for achievement definitions, and one for player-related actions.

Achievements services

All AWS GameKit solutions rely on a core set of AWS services, as described in Core services.

The following services are used specifically to manage achievements activity:

Amazon DynamoDB

AWS GameKit uses DynamoDB tables to store achievement definitions for the game and to track each player's achievement status. By using DynamoDB to store this type of data supports frequent read and write requests from game clients.

AWS Lambda

AWS GameKit uses Lambda functions to manage the process of storing and retrieving achievement data in the DynamoDB tables. Another Lambda function runs the logic to determines when a player successfully earns an achievement.

Amazon Simple Storage Service

AWS GameKit uses an Amazon S3 bucket to store achievement image files. Amazon S3 provides durable object storage capabilities.

Amazon CloudFront

AWS GameKit uses CloudFront to publish achievement image files for display in your game. CloudFront is a content delivery system that lets you cache content geographically near your players to minimize latency when downloading the content.

Achievements data encryption

Player data is encrypted both in transit and at rest.

In transit, AWS GameKit uses transport layer security (TLS) 1.2 or later for communication between a game frontend and backend components on AWS. All AWS GameKit game features use the Amazon API Gateway service to accept and process API calls. Learn more in the API Gateway Developer Guide, Data protection in transit.

At rest, player identity data is encrypted by the AWS services that the Achievement game feature uses. These services comply with industry standards. Learn more about how these services handle data encryption at rest:

- *Amazon DynamoDB Developer Guide*, DynamoDB encryption at rest
- *Amazon S3 User Guide*, Protecting data using encryption
- *Amazon CloudFront Developer Guide*, Data protection in Amazon CloudFront

Achievements configuration options

The configuration for your achievement game feature includes compiling your game's achievement definitions. When defining an achievement, you can specify the following characteristics:

- Unique name for the achievement. (Required)
- Player-facing achievement name.
- Player-facing achievement description. You can provide separate description strings for locked and unlocked states. When a game client requests achievement information for a player, AWS GameKit returns the string based on whether or not the player has earned the achievement.
- URL for a player-facing image or icon. You can provide separate URLs for locked and unlocked states. When a game client requests achievement information for a player, AWS GameKit returns the URL based on whether or not the player has earned the achievement.
- Maximum value. This is the numeric value that must be reached for a player to earn the achievement. For stateless achievements that require a single event to earn the achievement, this value might be set at 1. For stateful achievements that track progression toward the

achievement, this value sets the required number, such as 1000 (for an achievement like "Ate 1000 bananas"). (Required)

- Points awarded. If the achievement awards game points when the achievement is earned, this value indicates the number of points given to a player.
- The `Is_stateful` flag indicates if the achievement is stateful (true) or stateless (false). This flag determines how the backend service evaluates the player's current status to determine whether the player has earned the achievement.
- The `Is_secret` flag indicates when a game client can get the achievement information. When this flag is set to true, information and player status is returned only if the player has earned the achievement.
- The `Is_hidden` flag indicates when the achievement can be used in the game. When this flag is set to true, no information is returned for this achievement and player status can't be updated.
- Sort order number. This optional value indicates the order in which achievement information is returned when requested. It can be used to determine how achievements are listed in a game client display.

Achievements callable actions

The AWS GameKit API provides the following actions for the achievements game feature. After deploying AWS resources for achievements, your game frontend can use these calls to communicate with the backend resources.

- `UpdateAchievement()` Updates a player's current status for an achievement, checks to see if the player has met the requirements to earn the achievement, and, if so, sets the achievement to "earned".
- `GetAchievement` Retrieves information, including a player's current status, for all viewable achievements. Achievements are not viewable if they are flagged as hidden or if they are flagged as secret and the player hasn't yet earned them.
- `GetAchievement` Retrieves information, including a player's current status, for a single achievement, as specified by an achievement ID. This call returns all viewable achievements. It does not return hidden or unearned secret achievements.

Add achievements to your project

Summary

Learn how to build a complete cloud-based achievements system and integrate it into a Unreal Engine project. This topic guides developers through building backend services and then using the AWS GameKit API to connect frontend code to the backend on AWS.

When you're ready to build the achievements feature for your game, follow these basic steps. If you don't yet have AWS GameKit installed for your project, see Install the AWS GameKit plugin with Unreal Engine and Set up the AWS GameKit plugin for your game.

Step 1. Configure the achievements game feature for your game project.

1. In the Unreal Editor toolbar, open **Edit, Project Settings** and go to the **AwsGameKit** plugin section. Select the environment you want to work in and enter valid AWS credentials as needed. For more details, see Set up the AWS GameKit plugin for your game.
2. Expand the **Achievements** section. For this game feature you must define the achievements you want to include in your game. Choose the Configure button add or update achievement definitions. All achievement data is cached locally until you choose to sync it with AWS, and you can choose to save updates locally to use in later sessions.

   You have several options for maintaining achievement definitions in the AWS GameKit plugin:

   - Use the plugin UI to add achievement definitions. Entries are automatically cached locally.
   - Download an achievements JSON template by choosing **Get Template**. You can edit it directly and then upload to the plugin by choosing **Import from Local File**.
   - Retrieve the cloud-saved achievement definitions by choosing **Get Data from Cloud**.

   In each method, you are updating achievement definitions and saving them locally. When you're ready to make an achievement accessible to your game, either deploy new AWS resources for this feature or, if an Achievements backend has already been deployed, choose **Save Data to Cloud** to sync your local updates with the cloud-based versions.

Step 2. Deploy AWS resources for your achievements backend.

1. While still in the AWS GameKit settings for **Achievements**, scroll down to the deployment controls. Choose the AWS resource action **Create**. This action prompts AWS GameKit to deploy the complete AWS solution for this feature's backend services. When deploying, AWS GameKit connects to AWS to create all the AWS resources for the solution, using the configuration template for the active environment.

2. Deploying resources for identity and authentication typically takes 30 minutes to complete. During this time, the feature's deployment status reads **Deploying resources**. You can track the progress of your deployment status:

   - In the Unreal Editor, open the output log window to monitor status messages, events, and errors throughout the deployment.
   - In the AWS Management Console, open the AWS CloudFormation service. In the **Stacks** view you can watch as the Identity stack for your game project is deployed.

   When deployment is complete, the feature's deployment status reads **Deployed**. Your game backend for achievements is in place. You can make calls to it using the AWS GameKit API.

   Note

   NOTE: From this point on, you might begin incurring costs for this game feature. If you're still in the AWS Free Tier window, you will only incur costs if you exceed free tier limits.

3. When you've deployed the cloud backend for this feature, sync your achievements definitions with the backend services on AWS. To upload new achievement data to the cloud, choose **Save Data**. To download saved achievement data from the cloud to your local machine, choose **Get Latest**.

   Permisisons and version control for achievements definitions

   By default, AWS GameKit users are given access permissions to save achievement definitions to the cloud in the Development environment only. For instructions on how to modify these user permissions, see Manage permissions for achievements.
   AWS GameKit does not provide version control for achievement definition updates. For teams development, we recommend that you manage definition updates through your own version control system and restrict user permissions for uploading achievement definitions to your cloud backend.

Step 3. Add achievements functionality to your game.

Create UI elements and add code for the achievement-related workflows as needed for your game. See the plugin's achievements examples for illustration. Workflows might include:

- Get all achievement information and player status
  - `GetAchievements()`
- Get information and player status for a single achievement
  - `GetAchievement()`
- Add a trigger to a game event that prompts an update for a player's achievement status
  - `UpdateAchievement()`

Work with the achievements examples

The AWS GameKit plugin includes example assets for the achievements game feature.

Unreal Engine

The AWS GameKit plugin for Unreal Engine provides examples with C++ code, blueprints, and UI components. You can access the example files in the Unreal Editor content browser.

- Work with the C++ examples
- Work with the Blueprint and UI examples

Work with the C++ examples

In the Unreal Editor content browser, find the example asset at the following location:

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

This asset is a `.cpp` file. You can also find this file in the AWS GameKit plugin files, located at `...\AwsGameKit\Source\AwsGameKitEditor\Private\Achievements\AwsGameKitAchievementsExamples.cpp`.

This example file contains sample code that illustrates how to call each of the identity and authentication API actions. This file contains the basic set of runnable code and includes detailed comments.

You can work with this example in two ways: view the code in your IDE, or experiment with the API calls in Unreal Editor.

To view or edit the example code:

- Double-click the **AwsGameKitAchievementsExamples** asset to open the file in your IDE. You do not need to enter AWS credentials or deploy AWS resources before accessing this file, however none of the API calls can work without deployed resources.

  The example code includes a standard check to verify that AWS resources are deployed.

  Note that, as a first step, the example code creates an instance of the `UAwsGameKitAchievementsCallableWrapper` and initializes it. This must be done before making any API calls.

To experiment with the example in Unreal Editor:

You must have AWS resources deployed for identity and authentication and you must submit your valid AWS credentials in the AWS GameKit plugin project settings (see Set up the AWS GameKit plugin for your game).

1. Drag the **AwsGameKitAchievementsExamples** asset into level in the view port. It doesn't matter what level you add the asset to, this is just a mechanism that enables you to work with the asset settings.
2. In the Editor's **Details** pane, all of the achievements API calls are available with API request and response values.
3. To make an API call, enter some input values and click **Call**. The response is displayed in the **Return Value** field.
4. Try running the following call sequences to simulate standard achievements scenarios:

   - Retrieve all viewable achievement information and player status: Get Achievements.
   - Retrieve information and player status for a single achievement: : Get Achievement.
   - Update player status for a stateless achievement and verify earned status: Update Achievement, Get Achievement. For a stateless achievement, update the current value to "1", and then verify that the earned status has flipped to "true".

Work with the Blueprint and UI examples

In the Unreal Editor **Content Browser**, find the example assets at the following location:

```
AwsGameKit Content > Achievements >
```

There are two example assets:

- **BP_AwsGameKitAchievementsExamples**

  This asset is a basic blueprint that illustrates how you might add achievement-related functionality to your game code.

  Actions:

  - To open the blueprint, double-click the asset.
- **BP_AwsGameKitAchievementsExamplesUI**

  This asset includes a blueprint and sample UI objects for displaying achievement information and player status.

  Actions:

  - To open the blueprint, double-click the asset.

  To run the example, click **Play**.

Launch your game with AWS GameKit features

When packaging a game project for distribution, there are some additional steps you must take to ensure that the game is correctly set up to work with your game backend services on AWS. These steps are required for any project that makes calls to the AWS GameKit API.

Topics

Package a game project with AWS GameKit features

When packaging a game project for distribution, there are some additional steps you must take to ensure that the game is correctly set up to work with your game backend services on AWS. These steps are required for any project that makes calls to the AWS GameKit API.

Topics

Package a game for Windows or macOS

When packaging your game for distribution on Windows or macOS, complete the following tasks.

Unreal Engine

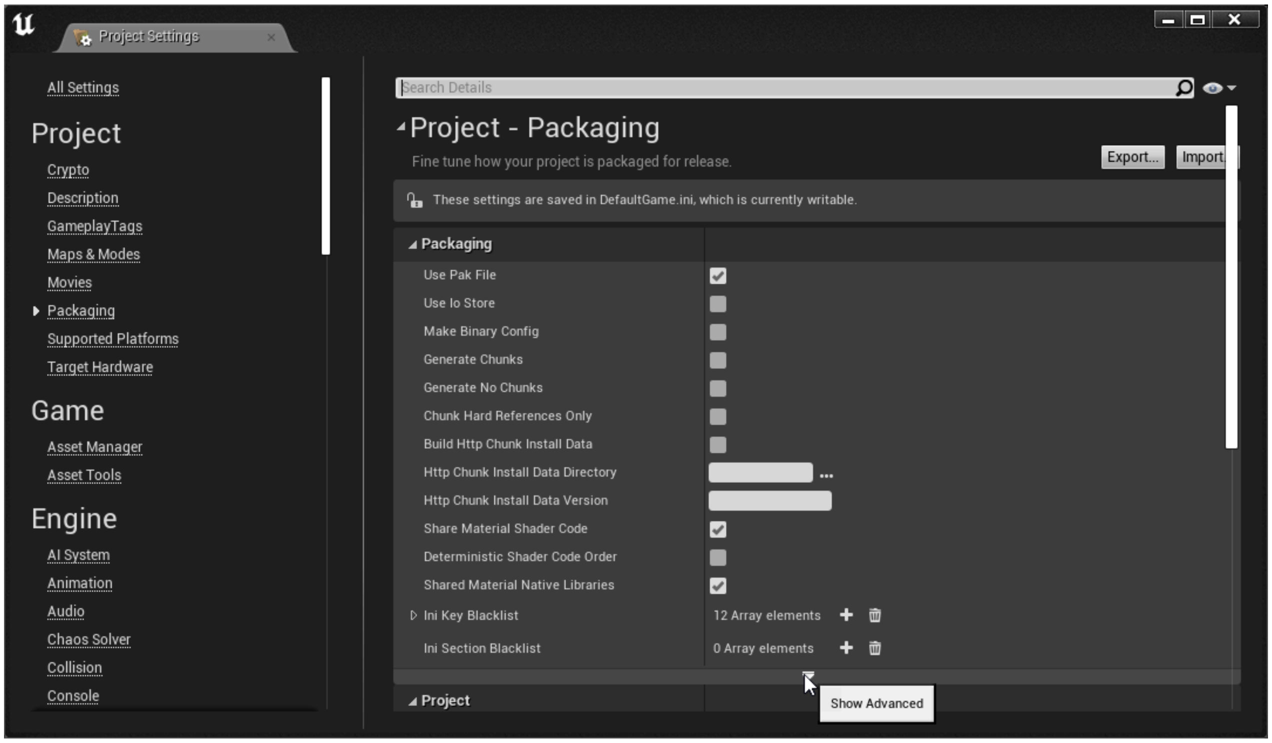
For Unreal Engine game projects, use the Unreal Engine project packaging instructions with the following modifications.

1. Before you start the packaging setup process, verify your current AWS GameKit plugin configuration. The Unreal packaging process uses whichever AWS GameKit configuration settings are currently active. In the Unreal Editor toolbar, open **Edit, Project Settings** and go to the **AwsGameKit** plugin section. In the **Environment and Credentials** section, verify the selections for game title, environment, and region.

2. Complete the Unreal packaging step "Setting a Game Default Map".

3. Add the AWS GameKit configuration in the packaging settings.

    a. In the Unreal Editor toolbar, choose **Edit**, **Project Settings**, **Packaging**, and expand the Packaging advanced settings, as shown.



Project - Packaging

Fine tune how your project is packaged for release.

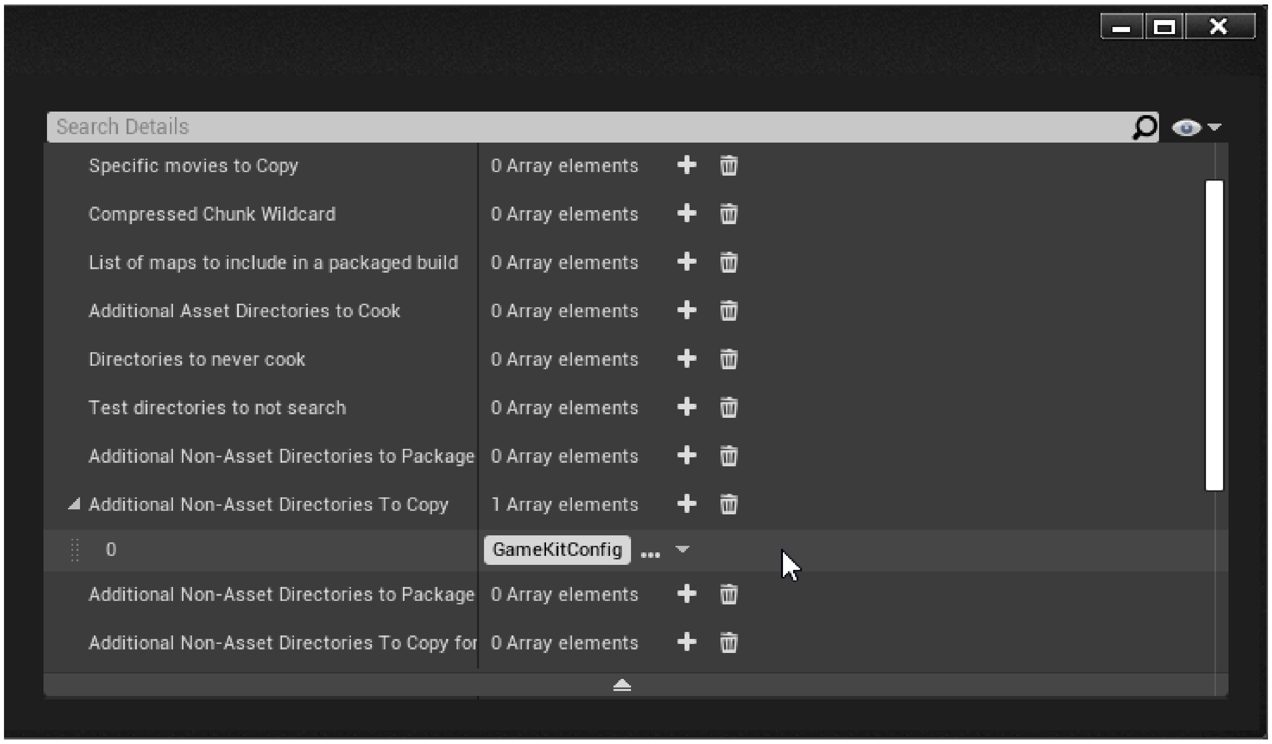
These settings are saved in DefaultGame.ini, which is currently writable.

Packaging

Use Pak File	✓
Use Io Store	
Make Binary Config	
Generate Chunks	
Generate No Chunks	
Chunk Hard References Only	
Build Http Chunk Install Data	
Http Chunk Install Data Directory	
Http Chunk Install Data Version	
Share Material Shader Code	✓
Deterministic Shader Code Order	
Shared Material Native Libraries	✓
Ini Key Blacklist	12 Array elements
Ini Section Blacklist	0 Array elements

Project

    b. In **Additional Non-Asset Directories to Copy**, add an array element and enter the value **`GameKitConfig`**.


Search Details

Specific movies to Copy	0 Array elements
Compressed Chunk Wildcard	0 Array elements
List of maps to include in a packaged build	0 Array elements
Additional Asset Directories to Cook	0 Array elements
Directories to never cook	0 Array elements
Test directories to not search	0 Array elements
Additional Non-Asset Directories to Package	0 Array elements
Additional Non-Asset Directories To Copy	1 Array elements
0	GameKitConfig
Additional Non-Asset Directories to Package	0 Array elements
Additional Non-Asset Directories To Copy for	0 Array elements

4. When you're ready to package your game for shipping, use the Unreal automation tool's `BuildCookRun` command. Exit the Unreal Editor, open a command prompt, and run the following command (modified for your game project). If you create your game packages in the Unreal Editor, the AWS GameKit plugin uses the `.dll` files in the game's `Binaries\` directory instead of the Shipping build configuration.

   To package a game for Windows (from a Windows device):

```
"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]`* – The directory path to your game project files (for example: `C:/Unreal Projects/MagicChickenGame/MagicChickenGame`).
   - *`[DESTINATION_DIRECTORY_ROOT]`* – A target location for the completed packaged product (for example: `C:\Archive`).

- *[UNREAL_PROJECT_NAME]* – The name associated with your Unreal game project (for example: `MagicChickenGame`).

To package a game for macOS (from a macOS device):

```
/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]* – The directory path to your game project files (for example: `/Users/amansa/Documents/Unreal\ Projects/MagicChickenGame/MagicChickenGame`).
- *[DESTINATION_DIRECTORY_ROOT]* – A target location for the completed packaged product (for example: `/Users/amansa/Documents/Unreal\ Projects/Archive`).
- *[UNREAL_PROJECT_NAME]* – The name associated with your Unreal game project (for example: `MagicChickenGame`).

Package a game for iOS

When packaging your game for iOS, complete the following tasks.

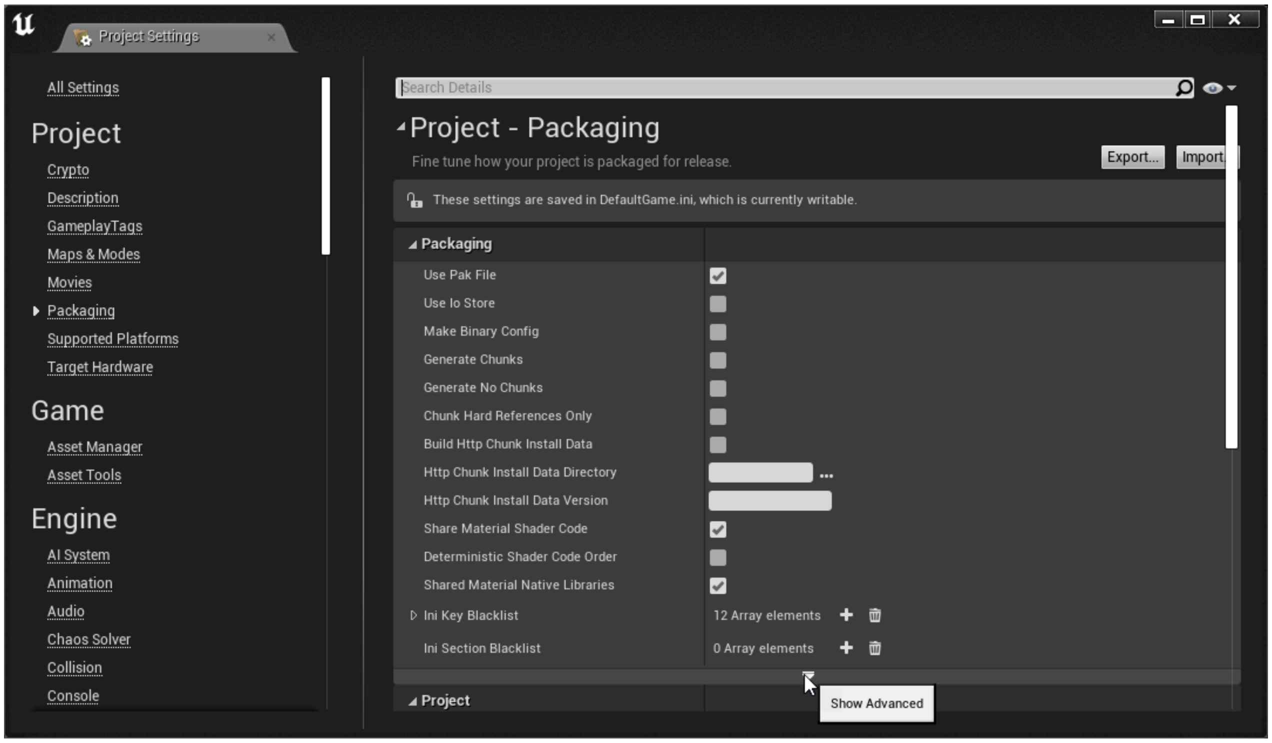
Unreal Engine

For Unreal Engine game projects, follow the Unreal Engine project packaging instructions, with the following modifications.

1. Before you start the packaging setup process, verify your current AWS GameKit plugin configuration. The Unreal packaging process will use whichever AWS GameKit configuration settings are currently active. In the Unreal Editor toolbar, open **Edit, Project Settings** and go to the **AwsGameKit** plugin section. In the **Environment and Credentials** section, verify the selections for game title, environment, and region.
2. Complete the Unreal packaging step "Setting a Game Default Map".

3. Add the AWS GameKit configuration in the packaging settings.

   a. In the Unreal Editor toolbar, choose **Edit**, **Project Settings**, **Packaging**, and expand the Packaging advanced settings, as shown.



Project - Packaging

Fine tune how your project is packaged for release.

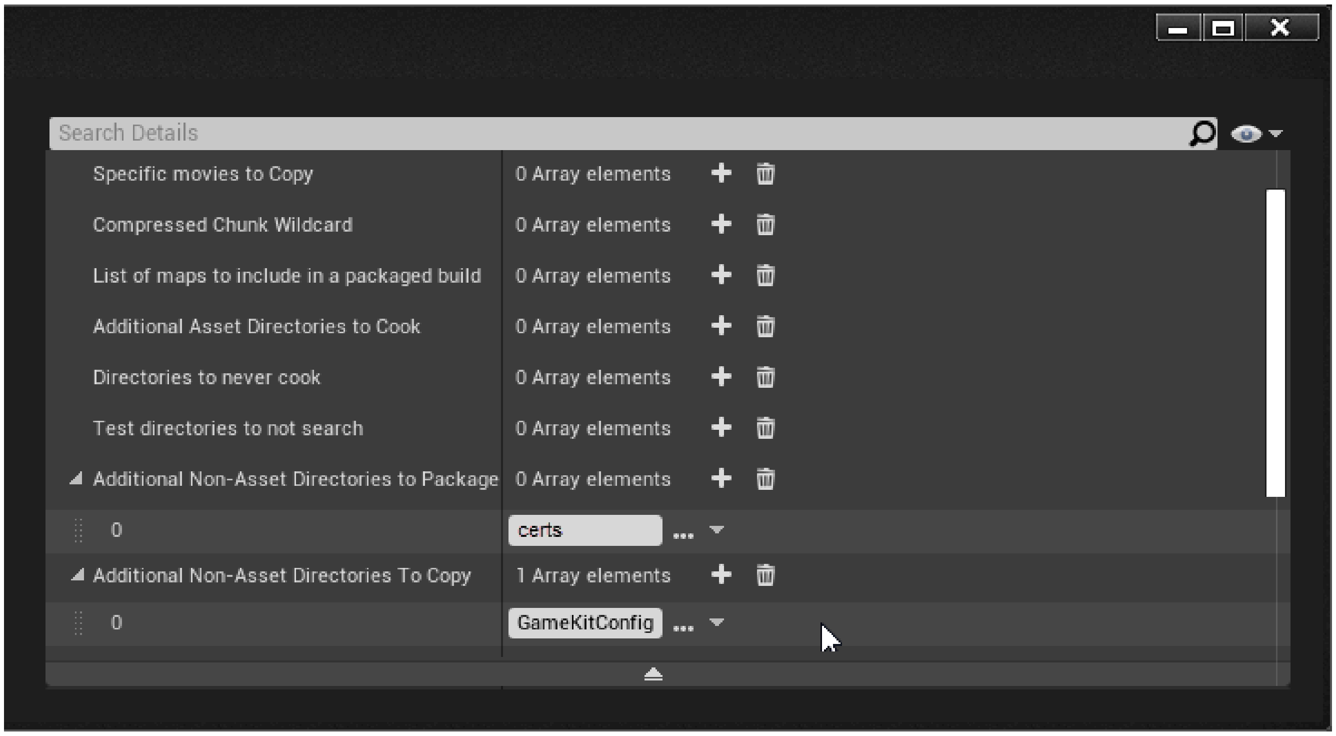
These settings are saved in DefaultGame.ini, which is currently writable.

Packaging

Use Pak File	✓
Use Io Store	
Make Binary Config	
Generate Chunks	
Generate No Chunks	
Chunk Hard References Only	
Build Http Chunk Install Data	
Http Chunk Install Data Directory	
Http Chunk Install Data Version	
Share Material Shader Code	✓
Deterministic Shader Code Order	
Shared Material Native Libraries	✓
Ini Key Blacklist	12 Array elements
Ini Section Blacklist	0 Array elements

   b. In **Additional Non-Asset Directories to Package**, add an array element and enter the directory **`certs`**. This directory is located in your game project files under `Content`.

   c. In **Additional Non-Asset Directories to Copy**, add an array element and enter the directory **`GameKitConfig`**. This directory is located in your game project files under `Content`.



Specific movies to Copy	0 Array elements
Compressed Chunk Wildcard	0 Array elements
List of maps to include in a packaged build	0 Array elements
Additional Asset Directories to Cook	0 Array elements
Directories to never cook	0 Array elements
Test directories to not search	0 Array elements
Additional Non-Asset Directories to Package	0 Array elements
0	certs
Additional Non-Asset Directories To Copy	1 Array elements
0	GameKitConfig

4. Open the `Target.cs` file, located in the `Source` directory of your game project files. Add the following lines to the target constructor.

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

   This update forces the use of `FMallocAnsi` when building for iOS. See the Unreal Engine documentation topic Targets for more information on UnrealBuildTool target files.

5. When you're ready to package your game for shipping, continue to the Unreal packaging step "Creating Packages" to package the game for iOS.

6. Connect your iOS device, and select Launch.

Package a game for Android

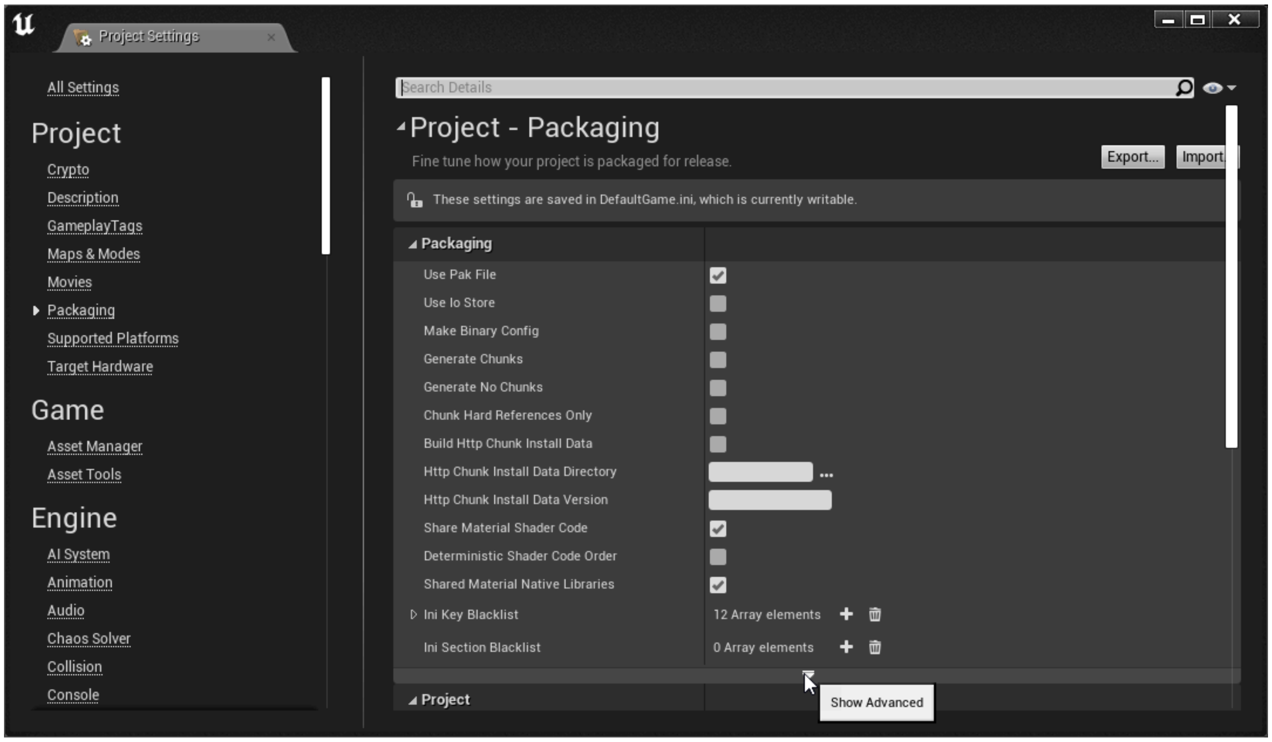
When packaging your game for Android, complete the following tasks.

Unreal Engine

For Unreal Engine game projects, follow the [Unreal Engine project packaging instructions], with the following modifications.

1. Before you start the packaging setup process, verify your current AWS GameKit plugin configuration. The Unreal packaging process will use whichever AWS GameKit configuration settings are currently active. In the Unreal Editor toolbar, open **Edit, Project Settings** and go to the **AwsGameKit** plugin section. In the **Environment and Credentials** section, verify the selections for game title, environment, and region.
2. Complete the Unreal packaging step "Setting a Game Default Map".
3. Add the AWS GameKit configuration in the packaging settings.

    a. In the Unreal Editor toolbar, choose **Edit**, **Project Settings**, **Packaging**, and expand the Packaging advanced settings, as shown.



Project - Packaging

Fine tune how your project is packaged for release.

These settings are saved in DefaultGame.ini, which is currently writable.

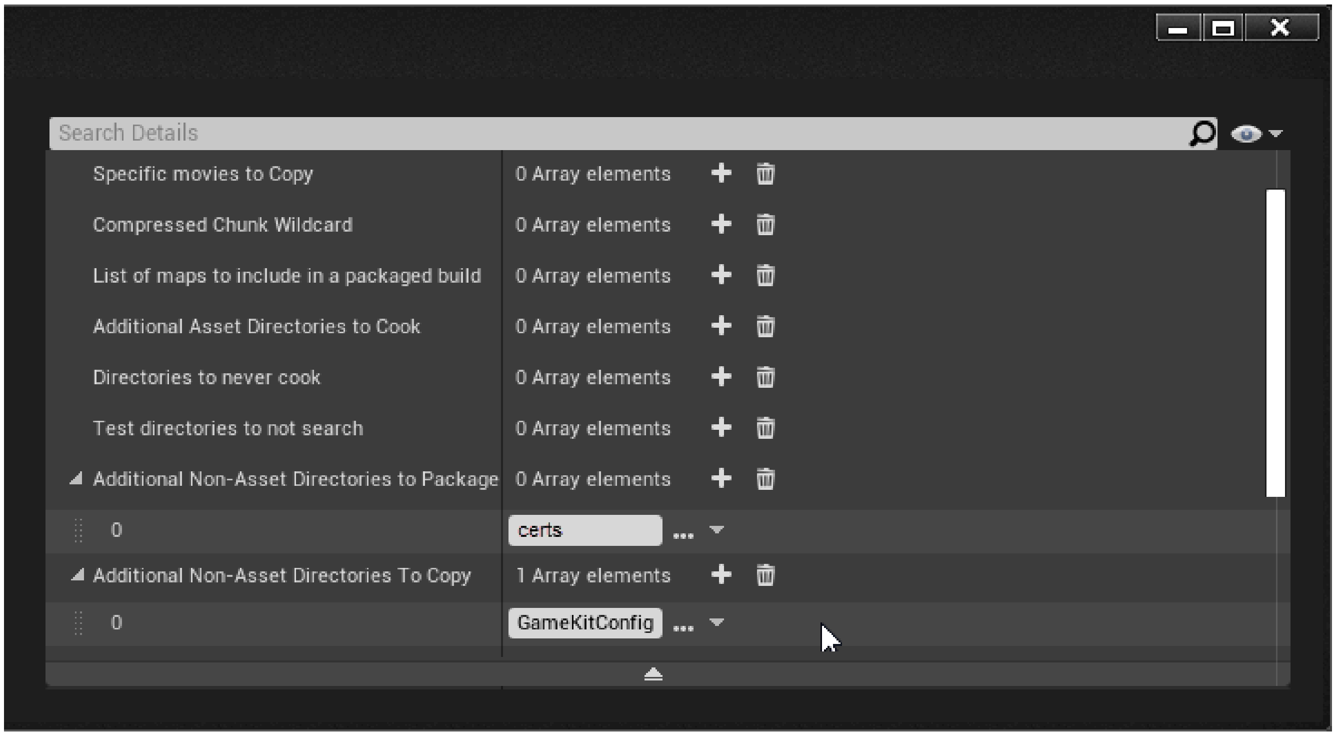
Packaging

Use Pak File	✓
Use Io Store	
Make Binary Config	
Generate Chunks	
Generate No Chunks	
Chunk Hard References Only	
Build Http Chunk Install Data	
Http Chunk Install Data Directory	
Http Chunk Install Data Version	
Share Material Shader Code	✓
Deterministic Shader Code Order	
Shared Material Native Libraries	✓
Ini Key Blacklist	12 Array elements
Ini Section Blacklist	0 Array elements

Project

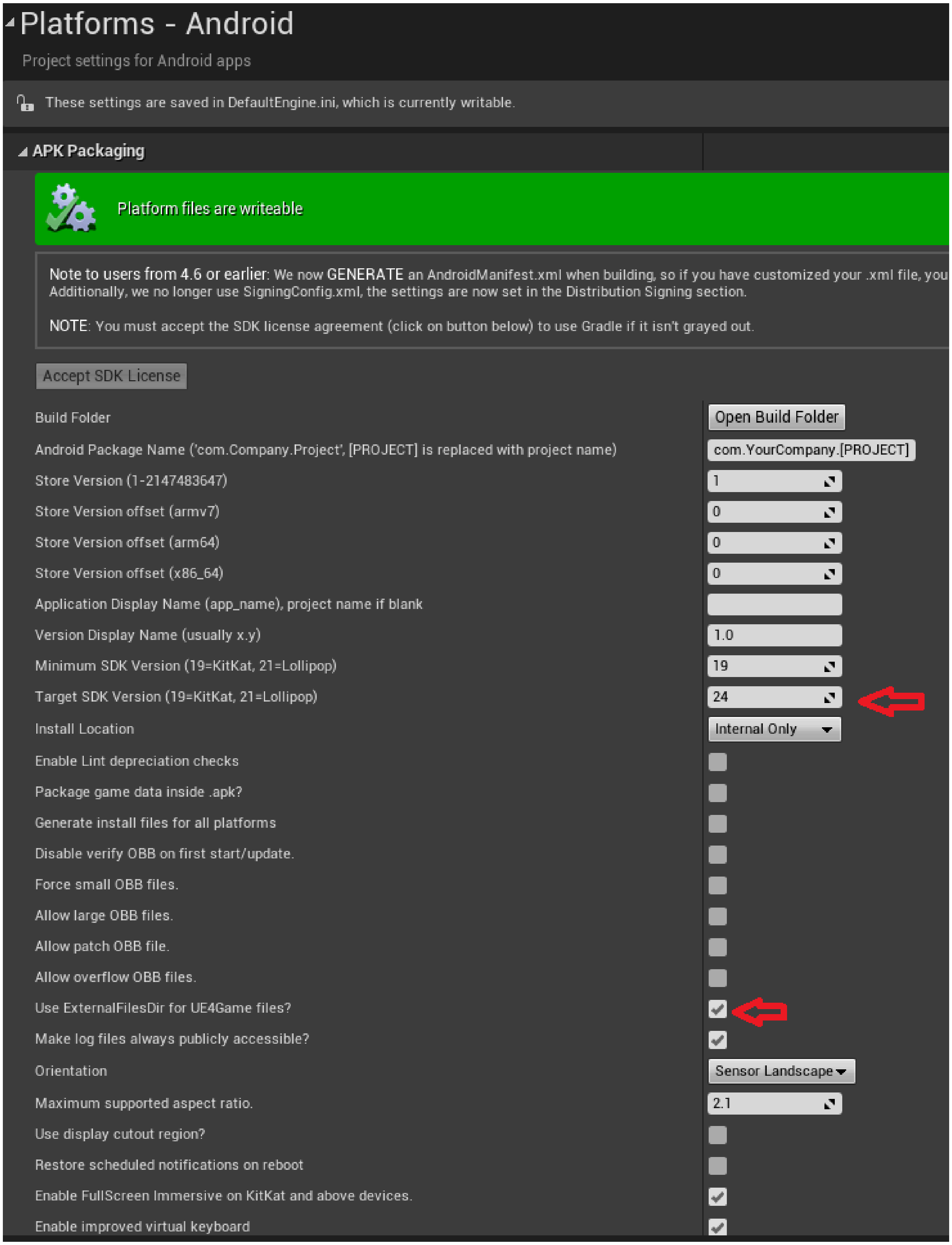
    b. In **Additional Non-Asset Directories to Package**, add an array element and enter the value **`certs`**.

    c. In **Additional Non-Asset Directories to Copy**, add an array element and enter the value **`GameKitConfig`**.



Specific movies to Copy	0 Array elements
Compressed Chunk Wildcard	0 Array elements
List of maps to include in a packaged build	0 Array elements
Additional Asset Directories to Cook	0 Array elements
Directories to never cook	0 Array elements
Test directories to not search	0 Array elements
Additional Non-Asset Directories to Package	0 Array elements
0	certs
Additional Non-Asset Directories To Copy	1 Array elements
0	GameKitConfig

4. Configure the game project for Android. In the Unreal Editor, open **Edit**, **Project Settings**, **Platforms**, **Android**. If the project hasn't yet been configured, choose **Configure Now**.
5. In the **APK Packaging** section, set the following values:



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

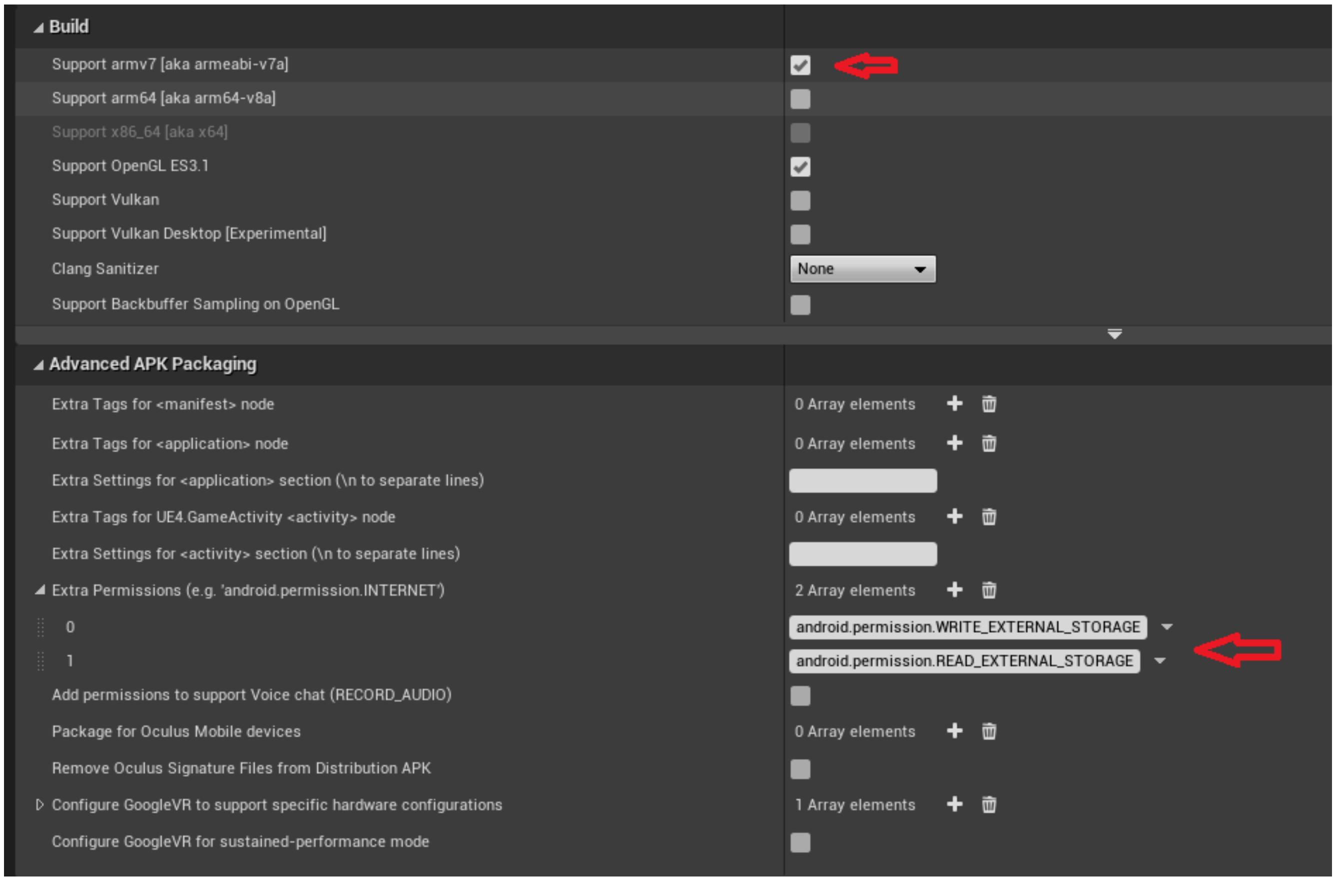
Build Folder	Open Build Folder
Android Package Name ('com.Company.Project', [PROJECT] is replaced with project name)	com.YourCompany.[PROJECT]
Store Version (1-2147483647)	1
Store Version offset (armv7)	0
Store Version offset (arm64)	0
Store Version offset (x86_64)	0
Application Display Name (app_name), 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. Set **Target SDK version** to 24.

> **Note**
>
> AWS GameKit has been tested on this SDK version. You might need to use another version depending on the device you're building for.

b. Select the option **Use ExternalFilesDir for UE4Game files**.

6. In the **Build** section, select the option **Support armv7**.



Build

Support armv7 [aka armeabi-v7a]	✓
Support arm64 [aka arm64-v8a]	
Support x86_64 [aka x64]	
Support OpenGL ES3.1	✓
Support Vulkan	
Support Vulkan Desktop [Experimental]	
Clang Sanitizer	None
Support Backbuffer Sampling on OpenGL	

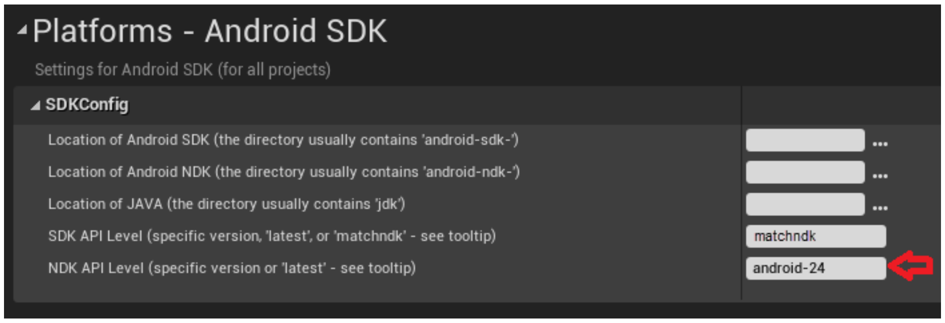
Advanced APK Packaging

Extra Tags for <manifest> node	0 Array elements
Extra Tags for <application> node	0 Array elements
Extra Settings for <application> section (\n to separate lines)	
Extra Tags for UE4.GameActivity <activity> node	0 Array elements
Extra Settings for <activity> section (\n to separate lines)	
Extra Permissions (e.g. 'android.permission.INTERNET')	2 Array elements
0	android.permission.WRITE_EXTERNAL_STORAGE
1	android.permission.READ_EXTERNAL_STORAGE
Add permissions to support Voice chat (RECORD_AUDIO)	
Package for Oculus Mobile devices	0 Array elements
Remove Oculus Signature Files from Distribution APK	
Configure GoogleVR to support specific hardware configurations	1 Array elements
Configure GoogleVR for sustained-performance mode	

7. In the **Advanced APK Packaging** section, under **Extra Permissions** add array elements with the following values:

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

8. Specify the **Android SDK** for your game project. In the Unreal Editor, open **Edit**, **Project Settings**, **Platforms**, **Android SDK**.


Platforms - Android SDK

Settings for Android SDK (for all projects)

SDKConfig

Location of Android SDK (the directory usually contains 'android-sdk-')

Location of Android NDK (the directory usually contains 'android-ndk-')

Location of JAVA (the directory usually contains 'jdk')

SDK API Level (specific version, 'latest', or 'matchndk' - see tooltip): matchndk

NDK API Level (specific version or 'latest' - see tooltip): android-24

a. Set the following directory locations to match your installation:

   - Location of Android SDK
   - Location of Android NDK
   - Location of Java

b. Set SDK API Level to the value **`matchndk`**.

c. Set NDK API Level to the value **`android-24`**.

9. When you're ready to package your game for shipping, continue on to the Unreal packaging step "Creating Packages" to package the game for the Android platform of your choice.

10. Connect your Android device, and select **Launch**.

Troubleshoot packaging issues

Missing script error

If you get an error message concerning a missing script, verify that you are using version 30.0.3 of the build tools.

API level errors

If you get the following errors, install Android API Level 29 in Android Studio and package again.

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

Optimize your game for mobile

If you're planning to distribute your game as a mobile app, we recommend that you consider these optimizations. Make these updates before you package your game project for mobile.

Set shutdown behavior

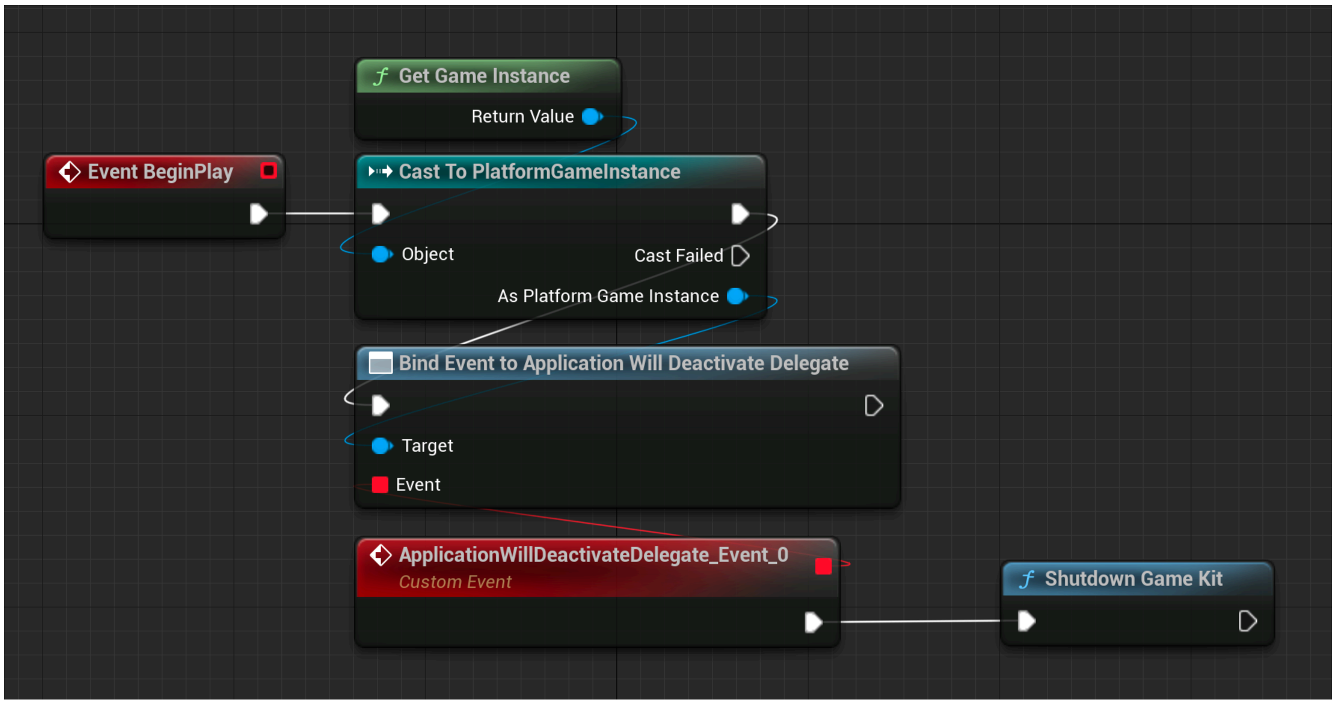
Shut down AWS GameKit components whenever your app is running in the background.

Unreal Engine

Add steps to your code to shut down AWS GameKit activity when the game is deactivated due to higher-priority activity, such as a phone call or text.

For games that use blueprint visual code:

1. In the Unreal Editor, open **Edit**, **Project Settings**, **Maps and Modes**. Set **Game Instance Class** to **`PlatformGameInstance`**.
2. Add logic ending in a AWS GameKit shutdown, as shown in the following blueprint example.



Get Game Instance
Return Value
Event BeginPlay
Cast To PlatformGameInstance
Object
Cast Failed
As Platform Game Instance
Bind Event to Application Will Deactivate Delegate
Target
Event
ApplicationWillDeactivateDelegate_Event_0
Custom Event
Shutdown Game Kit

For games that use C++ code:

- Add the following code and call it before exiting the game.

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

Note

If you want your game on iOS to exit when the app is moved to the background, in your project settings iOS platform, add the following key to **Additional Plist Data**:

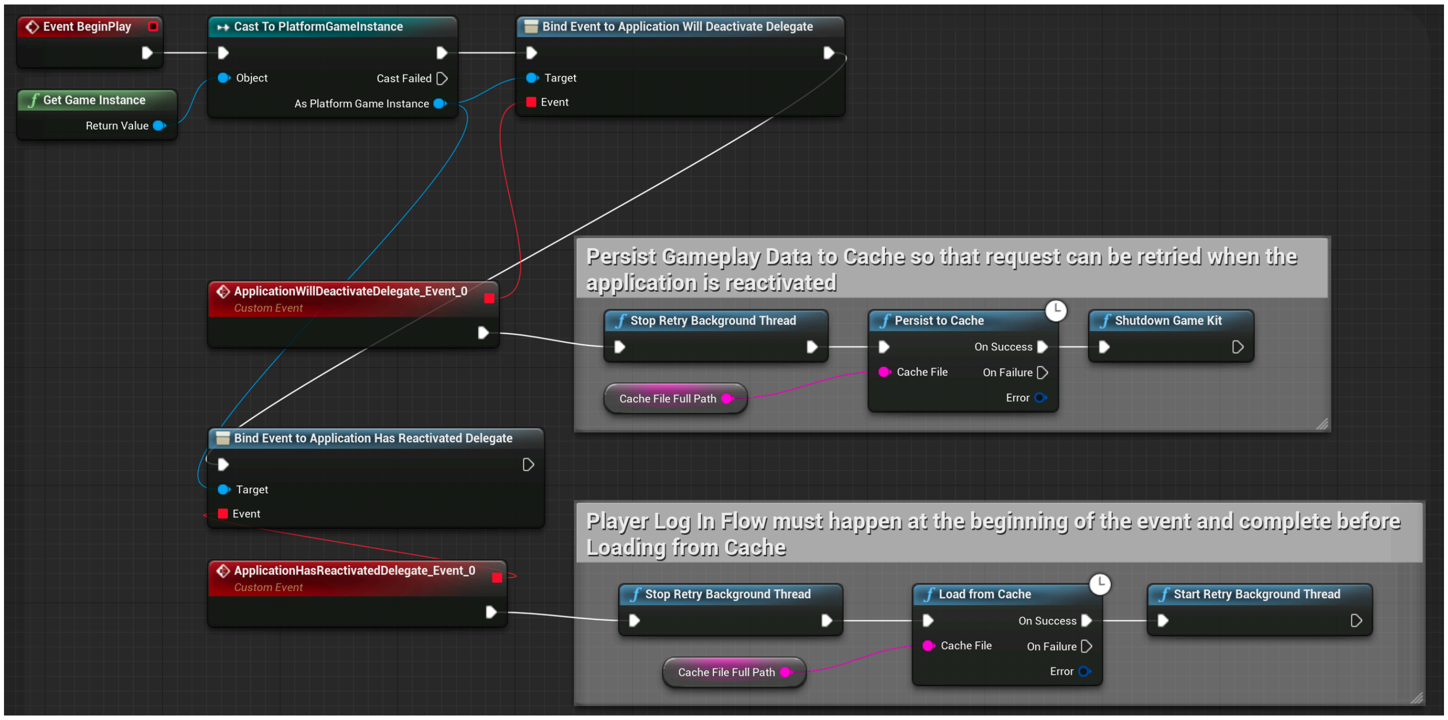
```
<key>UIApplicationExitsOnSuspend</key><true/>
```

Persist cache for user gameplay data

If your game uses the user gameplay data feature and is intended for distribution on mobile devices, you must make the following modifications to your game. These changes ensure that the internal request cache for the user gameplay data feature is persisted in the event of a AWS GameKit shutdown. The cache contains updates to user gameplay data that were not synced to the cloud before a shutdown, and ensures that these updates can be made when the app starts up again.

Make the following modifications:

- Use the `ApplicationWillDeactivate` delegate to persist the cache.
- Use the `ApplicationWillReactivate` delegate to load the cache after a player has logged in.


Event BeginPlay
Cast To PlatformGameInstance
Object
Cast Failed
As Platform Game Instance
Bind Event to Application Will Deactivate Delegate
Target
Event
Get Game Instance
Return Value
Persist Gameplay Data to Cache so that request can be retried when the application is reactivated
ApplicationWillDeactivateDelegate_Event_0
Custom Event
Stop Retry Background Thread
Persist to Cache
On Success
Cache File
On Failure
Error
Shutdown Game Kit
Cache File Full Path
Bind Event to Application Has Reactivated Delegate
Target
Event
Player Log In Flow must happen at the beginning of the event and complete before Loading from Cache
ApplicationHasReactivatedDelegate_Event_0
Custom Event
Stop Retry Background Thread
Load from Cache
On Success
Cache File
On Failure
Error
Start Retry Background Thread
Cache File Full Path

Prepare your AWS GameKit backend for production

As you prepare your project for release, use this guide to help get your AWS GameKit backend ready for production-level loads.

The AWS GameKit solutions for each game feature use a selection of AWS services and default configuration values that are best suited for project development and testing stages. When getting ready for production, its a good idea to fine tune your backend services. In particular, consider adjusting capacity to support a product in production and adding service features to support live players (such as monitoring). Some of these changes involve additional costs.

This topic provides recommendations and instructions for optimizing the core AWS services of your AWS GameKit backend.

Analyze feature usage patterns

We highly recommend that you analyze the usage and load patterns for each AWS GameKit feature in the development and testing stages. The AWS GameKit GitHub repo includes a starter python script for automating usage and load tests. Each game has different requirements for backend testing, based on the project's included AWS GameKit features and expected usage patterns. For example, if your project uses a lot of achievements but stores a small amount of user gameplay data, you probably want to focus on stressing the backend APIs for achievements.

When putting your testing strategy into practice, get the usage data collected in the custom AWS GameKit dashboards for each feature.

Set up monitoring dashboards

Activate monitoring dashboards in your test and production environments. Monitoring dashboards are critical for helping you make data-driven decisions about your game's backend on AWS Cloud. Use them to track how usage changes over time and make adjustments to maintain system health and cost efficiency.

AWS GameKit comes with detailed custom Amazon CloudWatch dashboards for each cloud feature. You can activate or deactivate each feature dashboard and access them directly from the AWS GameKit settings in your game engine. For help with using the dashboards, see Work with game feature dashboards. When testing or analyzing usage patterns, pay particular attention to these metrics:

AWS Lambda

- Latency (P99, P95 and P90)
- Concurrent executions
- Function errors

Amazon API Gateway

- Latency (P99, P95 and P90)
- 4xx and 5xx errors

Amazon DynamoDB

- Throttle
- Table request latency

Amazon Cognito

- Security

We also recommend that you create alarms for any metrics that are important to your game. You can create these alarms in CloudWatch that notify you when a metric crosses a threshold. For help setting up alarms, see the *Amazon CloudWatch User Guide* topic Using Amazon CloudWatch alarms.

Modify your AWS CloudFormation templates

For each AWS GameKit feature that your game uses, make the following pre-production updates to the feature's AWS CloudFormation template. We recommend that you make these proposed changes before you begin deploying resources to the AWS GameKit environment that you plan to use for production (either Production or a custom environment).

- Add an `IsProduction` condition
- Update AWS Lambda settings
- Update Amazon Cognito settings
- Update Amazon DynamoDB settings
- Set up a custom authorizer

To locate AWS CloudFormation templates:

If you haven't yet deployed resources in your production environment:

Make changes to the base AWS CloudFormation templates. Then, when you configure and create new AWS resources in a production environment, AWS GameKit automatically uses the updated base templates for your game. To locate the AWS GameKit base templates:

- In your AWS GameKit plugin for Unreal install location:

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

- In your Unity project files:

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

If you've already created AWS resources in your production environment:

Project-specific AWS CloudFormation templates already exist. Make updates to these existing templates and then redeploy each updated feature. To locate your project-specific templates:

- In your Unreal game project files:

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

- In your Unity project files:

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

To troubleshoot AWS CloudFormation issues:

If you have issues related to AWS CloudFormation templates when deploying your AWS GameKit features, see the *AWS CloudFormation User Guide* topic Troubleshooting CloudFormation for help with common issues.

Add an `IsProduction` condition

For each template, add an `IsProduction` condition.

- In the template, locate the `Parameters` section and add a `Conditions` section below it, as shown in the following example. Then, for other production-specific template updates, include the `IsProduction` condition.

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

Update AWS Lambda settings

All AWS GameKit features use AWS Lambda, each with a different usage pattern. Make the following updates to the Lambda configuration settings in each template. For help with AWS CloudFormation syntax for Lambda, see these *AWS CloudFormation User Guide* topics: AWS::Lambda::Function and AWS::Lambda::Version.

- Update `MemorySize` setting – Depending on your usage pattern, consider increasing memory from the default 128 MB. For example:

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

- Add `ProvisionedConcurrency` setting – This setting initializes a requested number of execution environments so that they're prepared to respond to your function's invocations. Configuring provisioned concurrency incurs charges to your AWS account. To configure provisioned concurrency, create a Lambda function version (a versioned copy of a Lambda function), add a `ProvisionedConcurrencyConfig` section, and set `ProvisionedConcurrentExecutions` to a value that can handle your production load. For details, see the *AWS Lambda Developer Guide* topic Lambda function versions. For example:

```
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
```

Make sure to update all references to the function so that they point to the versioned function. You can do this by changing the line `${MyLambdaFunction.Arn}` to `${MyLambdaFunction.Arn}:${MyVersionedLambdaFunction.Version}`.

Update Amazon Cognito settings

The AWS GameKit feature identity and authentication relies on Amazon Cognito to handle user registration and login workflows. Consider the following updates for production. For help with AWS CloudFormation syntax for Lambda, see these *AWS CloudFormation User Guide* topics: AWS::Cognito::UserPool.

- Turn on AdvancedSecurityMode feature – This feature provides advanced security risk detection. To use this feature, modify your AWS CloudFormation template for identity and authentication in the user pool settings as follows:

```
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
```

Update Amazon DynamoDB settings

Several AWS GameKit features use DynamoDB, each with a different usage pattern. Make the following updates to the DynamoDB configuration settings in each template. For help with AWS CloudFormation syntax for DynamoDB, see these *AWS CloudFormation User Guide* topics: AWS::DynamoDB::Table and AWS::ApplicationAutoScaling::ScalableTarget.

- Turn on automatic scaling – Use the DynamoDB autoscaling feature for read and write capacity units. With this feature, your product can handle increased production loads as needed by adjusting capacity based on usage metrics. For more details on DynamoDB autoscaling, see the *Amazon DynamoDB Developer Guide* topic Managing Throughput Capacity Automatically with DynamoDB Auto Scaling.

  To use autoscaling, create the following new sections for each DynamoDB table that requires it. Provide units and capacities that are appropriate for your expected load.

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

  For example:

```
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
```

For another example, see the AWS GameKit base template for user gameplay data. This feature has DynamoDB autoscaling enabled by default for the **Production** environment.

- For games with large workloads and a high number of consumed read units, consider using DynamoDB Accelerator. For more information on this feature, see Amazon DynamoDB Accelerator (DAX).

Set up a custom authorizer

If you're using a separate service (custom or third-party) for player login and authentication, set up your AWS GameKit backend to use a custom authorizer. Modify the AWS CloudFormation parameters file for the identity and authentication feature (`*[GameKit cloud templates]*\identity\parameters.yml`).

- Set `UseThirdPartyIdentityProvider` to TRUE.
- Provide a value for `JwksThirdPartyUri`.

Make these changes in the `parameters.yml` file for every AWS GameKit feature that your game uses. You must make these changes before you deploy AWS resources in your production environment for the identity and authentication feature.

Increase service quotas

Depending on your game's expected usage load, consider requesting the following service quota increases. Most AWS services have quotas, which might impact your game's performance at high usage loads.

- Amazon API Gateway – Increase "requests per second" (per AWS account per region). For details on API Gateway account-level limits and to request an increase, see this API Gateway Developer Guide topic Amazon API Gateway quotas and important notes.
- AWS Lambda – Increase "concurrent executions". For details and to request an increase, see Lambda quotas.
- AWS Key Management Service – Increase "requests-per second". For details and to request an increase, see AWS KMS request quotas.

Customize player registration email

If your game uses the identity and authentication feature with Amazon Cognito pools, Amazon Cognito automatically sends a verification email to new players when they register in your game. You have the option to customize the default text for this email.

To customize your registration verification email for players:

1. Open the AWS Management Console for Amazon Cognito, and select the option **Manage User Pools**.
2. Select the name of the user pool for the production version of your game. For GameKit, user pool names follow the pattern `gamekit_[environment]_[game title]_UserPool`. For example: `gamekit_prod_magicchicken_UserPool`.
3. With the user pool settings displayed, in the left side navigation, choose **Message customizations**.
4. Go to the section titled **Do you want to customize your email verification message?**
5. In this section, choose the code option, which directs Amazon Cognito to provide a verification code value to your players, and enter a custom email subject line and message. Be sure to position the verification code placeholder appropriately in your custom message. For more information, including maximum lengths, see the Amazon Cognito Developer Guide topic Customizing email verification messages.
6. When you're finished, select **Save changes**.

Add optional services

Consider taking advantage of the following optional services for your project. These services aren't included in the base templates for AWS GameKit features, but you can add them at any time.

Amazon Simple Email Service

Use Amazon Simple Email Service (Amazon SES) to send player registration verification emails from a custom email address instead of the default address used by Amazon Cognito. For details, see the following topics:

- Authorizing Amazon Cognito to send Amazon SES email on your behalf, *Amazon Cognito Developer Guide*

- Amazon Simple Email Service Developer Guide

AWS Web Application Firewall

AWS Web Application Firewall (AWS WAF) helps protect against common web-based exploits that affect availability, compromise security, and consume excessive resources. For production deployments, we recommend enabling AWS WAF. Follow these steps before you deploy any other features to the production environment.

To add AWS WAF:

1. Update IAM permissions for AWS GameKit users. Users who deploy AWS resources that use AWS WAF must have AWS WAF permissions. Create a new permission policy with the following syntax, and attach the new policy to IAM user groups. For details on creating permissions policies for AWS GameKit, see Set up a user with AWS GameKit access.

   This syntax creates AWS WAF WebACL with the following rules:

   - Core rule set (CRS) (see Baseline rule groups, *AWS WAF Developer Guide*
   - SQL injection rule set (see Use-case specific rule groups, *AWS WAF Developer Guide*
   - IP reputation rule set (see IP reputation rule groups, *AWS WAF Developer Guide*

   ```
   {
       "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. Update your game's main AWS CloudFormation template. For help with AWS CloudFormation syntax for AWS WAF, see these *AWS CloudFormation User Guide* topics: AWS::WAFv2::WebACL and AWS::WAFv2::WebACLAssociation.

   To locate your AWS CloudFormation templates, see Modify your AWS CloudFormation templates. The main template is `[GameKit cloud templates]\main\cloudformation.yml`.

   Add the following syntax to the `Resources` section:

```
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. Deploy or redeploy any AWS GameKit feature. This action automatically redeploys the main stack with your latest AWS WAF changes.

AWS Shield

AWS Shield Standard provides protection against the most frequently occurring network and transport layer DDoS attacks that target a web site or application. This protection is on by default. AWS Shield Advanced is a paid service that provides additional protection for internet-facing applications that run on Amazon Elastic Compute Cloud (Amazon EC2), Elastic Load Balancing (ELB), Amazon CloudFront, Global Accelerator, and Amazon Route 53. For information on costs, see AWS Shield Pricing.

To turn on AWS Shield Advanced, you have two options:

- Use the AWS Management Console for AWS Shield to configure you coverage.
- Update your AWS CloudFormation templates as described in the AWS CloudFormation User Guide topic AWS::FMS::Policy (see examples).

Adjust usage of AWS GameKit client API

For the user gameplay data feature, when calling the AWS GameKit API with large numbers of bundles or items (in the order of hundreds), there is the potential for HTTP request timeouts. Ways to mitigate timeout risks include:

- Make API calls with data in smaller batches.
- Set up bundles for users at account creation time.
- Increase the value of `ClientTimeoutSeconds` in `FUserGameplayDataClientSettings`.

Working with AWS resources

When you use the AWS GameKit plugin to deploy AWS resources to host backend services for a game feature, AWS GameKit creates and manages updates for these resources by using the AWS CloudFormation service.

AWS GameKit takes your game feature configuration settings and uses this information to build an AWS CloudFormation template for the game feature's backend solution. The AWS CloudFormation template describes the set of AWS resources in the solution, and AWS CloudFormation uses this template to deploy resource stacks to support AWS GameKit game features.

You have the option to view and/or modify AWS GameKit-created templates and resource stacks directly by using AWS tools such as the AWS Management Console or the AWS Command Line Interface (AWS CLI).

Topics

View AWS templates and resources

You can use the AWS Management Console to view the AWS CloudFormation templates and resource stacks that AWS GameKit generates for your game features.

Viewing an AWS resource stack

1. Open the AWS CloudFormation console. When you sign in to the console, use the same AWS user that you used in the AWS GameKit plugin to deploy the resources. The AWS user must have sign-in credentials, which are different from the security credentials that you need in your game engine.
2. Open the **Stacks** page. If the AWS CloudFormation console doesn't immediately open to this page, go to the left navigation pane and choose **Stacks**. The **Stacks** page displays all of the resource stacks that your AWS account owns.
3. In the AWS CloudFormation console, use the AWS Region control to select the Region where your AWS GameKit resources are deployed.

4. From the listed stacks, locate the stack you want to view.
5. Open the stack name that ends in the name of a feature, such as "Identity". The **Stack details** page contains general information about the stack. You can access the current AWS CloudFormation template configuration, view the list of AWS resources in the stack, and open an event log that tracks activity related to the stack resources and template.

To view AWS resources for a AWS GameKit feature

- In the AWS CloudFormation console, **Stack** page, the resource list includes a physical ID link for every deployed AWS resource that is currently in the stack. Use these links to open the service console for a resource and get more information about it.

  For example, in the Identity/Authentication feature stack, you can follow the physical ID link for the user pool resource. This link opens the Amazon Cognito service console, where you can explore the user pool resource in greater detail.

Update AWS templates and resources

Summary

You can access your AWS GameKit-deployed AWS resources directly using AWS tools such as the AWS Management Console. If you update these resources or their configuration templates directly, be aware that you can introduce errors into your AWS GameKit game features. This topic is for developers who want to use AWS tools to gain greater insight about their game backend components and understand how any updates might affect their game features.

If you work exclusively in the AWS GameKit plugin, your AWS GameKit-created templates and resources on AWS will always remain in alignment with your feature configurations. If you make changes to the AWS CloudFormation templates and resources outside of the plugin, they run the risk of falling out of alignment, causing conflicts and unanticipated behavior. For this reason, we recommend that you avoid updating your AWS GameKit-created templates and resources if you plan to continue using the plugin.

If you decide to update your AWS CloudFormation templates directly, be aware of how the method affects how your updates are handled.

1. Use the plugin to update feature configuration and redeploy. On redeployment, AWS GameKit first modifies the existing CloudFormation stack template and then updates or replaces the

stack resources. Templates and resources for the feature are reset to the latest AWS GameKit feature configuration.

NOTE: If you update a feature configuration but don't redeploy, the updates are saved locally only. They do not affect your AWS templates and resources.

AWS GameKit stores a version of the AWS CloudFormation templates (as `*.yml` files) with your local game project files. Instead of working in the plugin UI, you can opt to edit these files. AWS GameKit uses this file to populate the configurations in the plugin UI. The templates are stored in your game project's directory: `...\Unreal Projects\<game project >\<AWS GameKit game title>\cloudformation\`).

2. Update CloudFormation templates or AWS resources directly. When you use CloudFormation to modify a stack, you make updates to the template, which is then used to update stack resources. None of these changes are saved back to the AWS GameKit feature configuration. These changes will be lost if the AWS GameKit plugin is later used to redeploy resources.
3. Update live AWS resources directly using AWS (not recommended). You have the option of viewing and changing AWS resources by working directly with the resource in its associated AWS service. For example, you might access your Identity user pool directly in the Amazon Cognito console. None of these changes are propagated back to the AWS AWS CloudFormation stack template or the AWS GameKit feature configuration. These changes will be lost the next time the stack is updated or the feature resources are redeployed in the AWS GameKit plugin.

AWS GameKit Reference

This section contains a collection of reference material for use with AWS GameKit.

Topics

AWS services and resources

This reference page lists the AWS services and resources that make up the backend solutions for each AWS GameKit feature.

Core services

The following services and resources are used for core AWS GameKit functionality:

- AWS CloudFormation
- Amazon API Gateway
- AWS Identity and Access Management
- AWS Identity and Access Management (IAM)
- Amazon CloudWatch Logs
- Amazon CloudWatch (for dashboards)
- Amazon Simple Storage Service (Amazon S3)
- Amazon Cognito (for player authentication)

Identity and authentication services

The following services and resources are used for identity and authentication functionality. For more details on how identity and authentication solution architecture, see .

- All core services.

- Amazon Cognito. Resources include a user pool and an identity pool for third-party identity providers.
- Amazon DynamoDB. Resources include a table to track player login methods.
- AWS Key Management Service (AWS KMS).
- AWS Secrets Manager. Used to store app credentials for third-party identity providers.

Achievements services

The following services and resources are used for achievements functionality. For more details on how achievements solution architecture, see .

- All core services.
- Amazon DynamoDB. Resources include two tables, one to store a game's achievement definitions, and one to track player achievement status.
- Amazon S3. Resources include an S3 bucket to store achievement icon image files.
- Amazon CloudFront. Resources include a CloudFront distribution to deliver icon images to a game client.

User gameplay data services

The following services and resources are used for user gameplay data functionality. For more details on how user gameplay data solution architecture, see .

- All core services.
- Amazon DynamoDB. Resources include two tables, one to store a game's achievement definitions, and one to track player achievement status.

Game state cloud saving services

The following services and resources are used for game state cloud saving functionality. For more details on the game state cloud saving solution architecture, see .

- All core services.
- Amazon Simple Storage Service (Amazon S3). AWS GameKit uses an Amazon S3 bucket to store game save files.

- Amazon DynamoDB. AWS GameKit uses a DynamoDB table to store metadata about the game save files. Using DynamoDB to store this type of data supports frequent read and write requests from game clients.
- AWS Lambda. AWS GameKit uses Lambda functions to manage the tasks of storing game save files and metadata and analyzing synchronization states.

AWS GameKit supported AWS Regions

This reference page lists the current status of AWS GameKit availability in all AWS Regions. A Region must support all of the AWS services and resources that are required for a AWS GameKit game feature backend.

In the AWS GameKit plugin, you select an environment to work in. The environment includes an AWS Region selection, which determines where all of the game backend services and resources for that environment will be deployed.

If you deploy the backend to an AWS Region that doesn't support AWS GameKit resources, the deployment will fail with messaging such as CreateStack Failed: Template format error: Unrecognized resource types.

Region availability

Use the following table to select an AWS Region for use with your AWS GameKit-enabled game project.

Region	Feature: Identity and authentic ation	Feature: Achieveme nts	Feature: User gameplay data	Feature: Game state cloud saving
us-east-1: US East (N. Virginia)	**Available**	**Available**	**Available**	**Available**
us-east-2: US East (Ohio)	**Available**	**Available**	**Available**	**Available**
us-west-1: US West (N. California)	**Available**	**Available**	**Available**	**Available**

Region	Feature: Identity and authentication	Feature: Achievements	Feature: User gameplay data	Feature: Game state cloud saving
us-west-2: US West (Oregon)	**Available**	**Available**	**Available**	**Available**
af-south-1: Africa (Cape Town)	*`Not Available`*			
ap-east-1: Asia Pacific (Hong Kong)	*`Not Available`*			
ap-south-1: Asia Pacific (Mumbai)	**Available**	**Available**	**Available**	**Available**
ap-northeast-3: Asia Pacific (Osaka)	*`Not Available`*			
ap-northeast-2: Asia Pacific (Seoul)	**Available**	**Available**	**Available**	**Available**
ap-southeast-1: Asia Pacific (Singapore)	**Available**	**Available**	**Available**	**Available**
ap-southeast-2: Asia Pacific (Sydney)	**Available**	**Available**	**Available**	**Available**
ap-northeast-1: Asia Pacific (Tokyo)	**Available**	**Available**	**Available**	**Available**
ca-central-1: Canada (Central)	**Available**	**Available**	**Available**	**Available**
eu-central-1: Europe (Frankfurt)	**Available**	**Available**	**Available**	**Available**
eu-west-1: Europe (Ireland)	**Available**	**Available**	**Available**	**Available**
eu-west-2: Europe (London)	**Available**	**Available**	**Available**	**Available**

Region	Feature: Identity and authentication	Feature: Achievements	Feature: User gameplay data	Feature: Game state cloud saving
eu-south-1: Europe (Milan)	*Not Available*			
eu-west-3: Europe (Paris)	**Available**	**Available**	**Available**	**Available**
eu-north-1: Europe (Stockholm)	**Available**	**Available**	**Available**	**Available**
me-south-1: Middle East (Bahrain)**	**Available** (if enabled)	**Available** (if enabled)	**Available** (if enabled)	**Available** (if enabled)
sa-east-1: South America (Sao Paulo)	**Available**	**Available**	**Available**	**Available**
China (Beijing and Ningxia) Regions	*Not Available*			

**

This AWS Region is not automatically enabled for your AWS account. To deploy your game backend to this Region, you must take action to enable it. See Enabling an AWS Region for instructions.

AWS GameKit deployment states

This page describes the possible deployment states for an AWS solution as created through the AWS GameKit plugin.

You can check the current status of your AWS solutions for each game feature while in the Unreal Editor. Open the **Project Settings** window and go to the AWS GameKit plugin page. Verify that you have an active environment selected and valid credentials entered.

There are two types of deployment states:

- Steady states describe the end state of a completed deployment action. The deployment action is no longer in progress.
- Transient states provide detail about a deployment action that is currently in progress.

Note

Deployment states do not provide information about locally cached configuration settings for AWS GameKit. They do not indicate if local settings are in sync with or have diverged from the configuration of deployed resources.

Steady states

Status	Description
Undeployed	No AWS resources are currently deployed for the game feature. This status might indicate that no resources have been deployed yet, or that existing resources have been successfully deleted.
Deployed	Viable AWS resources are currently deployed for the game feature. This status indicates that the most recent deploy or update action was successful.
Error	AWS resources were deployed for the game feature, but the resources are in an error state and are not viable. This status indicates that the most recent deploy, update, or delete action failed, and the system was not able to completely roll back to an earlier viable state.
Rollback complete	The most recent deployment-related action failed, and the system has successfully rolled back to an earlier viable state.

Transient states

These states indicate that deployment-related activity is in progress, which can take five to fifteen minutes or more. They are listed below in the sequence they occur.

- **Generating templates**
- **Uploading dashboards**
- **Uploading layers**
- **Uploading functions**
- **Deploying resources** or **Deleting resources**
- **Retrieving status**
- **Running** (AWS resources are being created during this stage)

AWS GameKit concepts and terminology

The following terminology and concepts are relevant to AWS GameKit. For terms that are commonly used in the game development industry, the definitions describe how they apply to AWS GameKit.

AWS GameKit terms

Configuration

The collection of materials that AWS GameKit uses to deploy and keep track of the backend resources for cloud-based game features. It includes templates that define the solution architecture for each feature backend, the configuration options that you select when setting up a feature, and other details specific to your game project. If you work in multiple environments to deploy resources through AWS GameKit, your configuration contains separate settings for each environment. The configuration is stored locally with the game project files, in a folder that's named for the game title.

Environment

Used in the AWS GameKit plugin to designate a collection of configuration settings and deployed AWS resources for a game project. Although a game project can have only one configuration, each configuration can track settings and deployments for multiple environments, such as for development, testing, and production. All activity in the plugin, including deploying AWS resources, is done with respect to the currently selected environment.

Game state cloud saving

AWS GameKit game feature that uses a backend service on the AWS Cloud to store game save files. This feature employs two-way file synchronization to ensure that locally stored and cloud-stored game save files are updated to the latest version. With game state cloud saving, players can play games across multiple devices without having to manually track save files.

Game title

User-defined unique identifier for a game project's AWS GameKit configuration. Assigning a game title is one of the first steps you take when setting up a game project to use the AWS GameKit plugin. The names of all AWS resources that you deploy through AWS GameKit include the game title string.

Player data

Player-specific information that might include personal identifiable information such as user names, email, or social media IDs. It also might include account information such as subscription data, in-game purchases, friends/blocked lists, and demographics. Because player data includes personal identifying information, heightened security is used to handle this type of data.

Stateful achievement

Also called multi-step, this type of achievement requires more than one action to earn. When defining a stateful achievement, you specify the number of steps it takes to earn the achievement. The AWS GameKit achievements backend tracks a player's ongoing progress toward earning the achievement, updating the player's status incrementally until they earn the achievement.

Stateless achievement

Also called single-step, this type of achievement requires just one action to earn. When defining a stateless achievement, you specify the number of steps as "1". The AWS GameKit achievements backend tracks the player's status for the achievement as either locked (not earned) or unlocked (earned).

User gameplay data

AWS GameKit game feature that uses a backend service on the AWS Cloud to store data generated during gameplay. Examples of gameplay data include player scores, inventories, and player progression. It can also include game analytics and telemetry, such as in-game decisions and behavior. With gameplay data saved in the cloud, players can switch devices, continue gameplay, and have their data follow them. User gameplay data is distinct from players data, such as account information. User gameplay data does not include personal identifying information.

Game development terms

Game backend

Refers to services and resources that support game features but are separate from the game application. Typically, backend services run on remote servers, and game frontends use APIs to communicate with them. Cloud-based backends are services that are running on virtual resources. AWS GameKit provides tools to build cloud-based game backend services with AWS, taking advantage of its superior performance, security, and reliability. Some common uses for game backend services include persistent data storage, identity validation, notifications, and custom logic for multi-player features like leaderboards and matchmaking.

Game frontend, Game client

Refers to the part of the game that users interact with, including the visual presentation and user interface. Many game features have coordinated frontend and backend components. For example, with a game save feature, the frontend might exist as a menu or messaging that lets a player save their game progress. This player action prompts the frontend to capture the game state, send a "save" API request to the appropriate backend service, and handle success or failure. Other game features, such as achievements, are activated on the frontend through gameplay activity.

Game engine terms

Use these links to access terminology and documentation on game engines that are supported by AWS GameKit:

- Unreal Engine 4 Terminology

Blueprint

Unreal Engine's visual coding system. Learn more about Blueprint visual scripting in the Unreal Engine documentation.

AWS terms

Resource stack

A collection of AWS resources that can be managed as a single unit. AWS GameKit manages all of the AWS resources for a game feature backend in a single AWS CloudFormation stack. When you deploy resources for a game feature using the plugin, AWS GameKit creates a stack. Each AWS GameKit-created stack is specific to the game profile name, environment, AWS Region, and game feature. You can view your resource stacks with the AWS Management Console for AWS CloudFormation.

Resource stack template

Configuration settings that model the collection of AWS resources to be created in a stack. AWS CloudFormation templates are formatted text files in JSON or YAML. AWS GameKit, using the AWS CloudFormation service, uses the template to deploy AWS resources for a game feature in a specified AWS Region. You can view your templates with the AWS Management Console for AWS CloudFormation.

User pool

A directory of players who have registered in your game and have a unique player ID that can be used for authentication. For games created with AWS GameKit, the identity and authentication backend uses Amazon Cognito to manage the user pool. Players in the user pool can sign in to your game directly with an email and password, or through a third party.

AWS GameKit releases

To find comprehensive version information for AWS GameKit components, including releases and known issues, see AWS GameKit Releases.