# Fusion BR200 - Multiplay Integration Documentation

Matchmaker

Backfill

The Fusion BR (Battle Royale) project demonstrates how to create a fully functional multiplayer game using Photon Fusion and Unity Gaming Services (UGS), including Multiplay and the Unity Matchmaker.

Before continuing, review these requirements:

- You must have a Unity ID.
- You must have a Photon account and a Photon Fusion Application Id.
- You must use Unity Editor 2021.3.5f1.

# Get started

Download the sample from the Package Manager to get started with the Fusion BR200 project. After downloading the sample project, complete the following steps:

1. Get started with UGS
2. Install the Unity Editor
3. Get started with Photon Fusion
4. Link your Photon Fusion project

**Note**: Visit Unity Dashboard Support if you need help with any Unity services. Visit Photon's Get Help page for help with Photon Fusion.

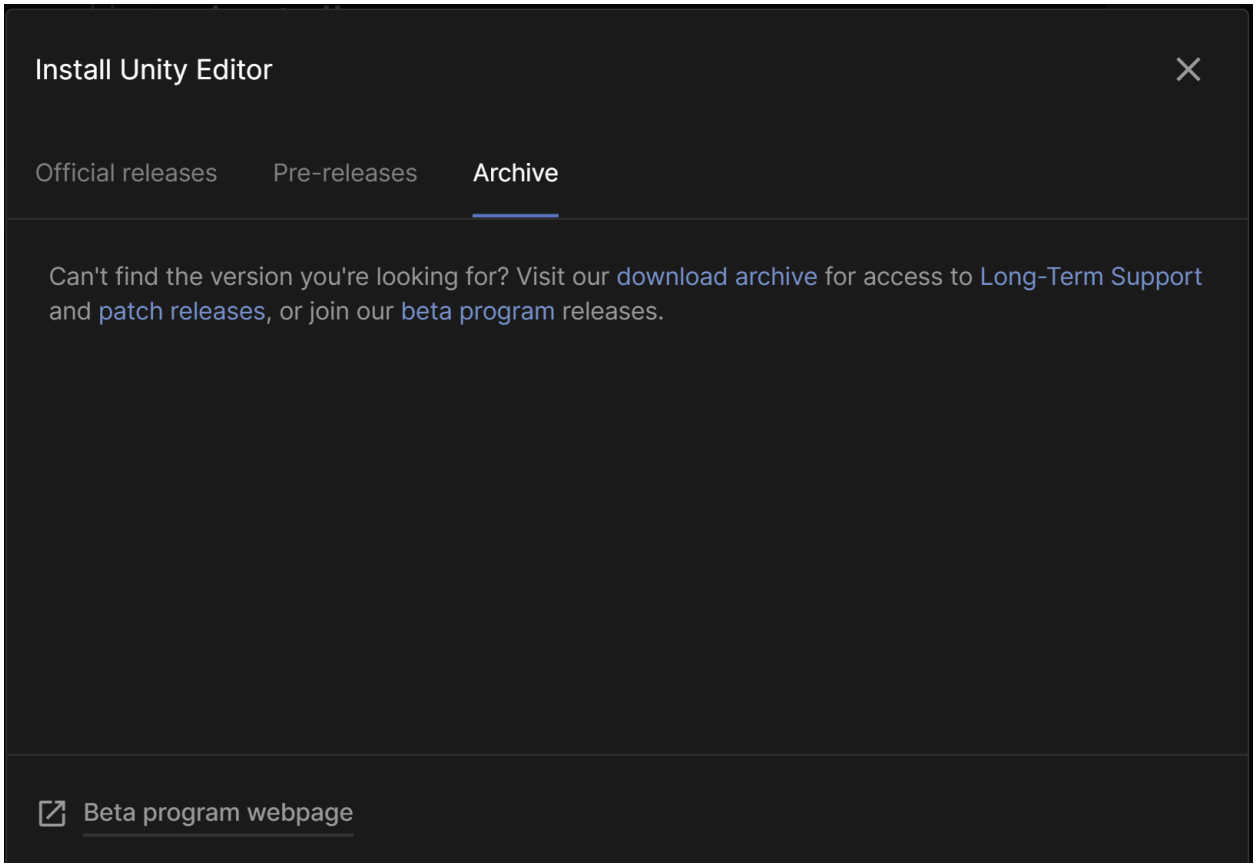
## Get started with UGS

You need a Unity account to access Multiplay and the Unity Matchmaker. If you don't already have a UGS account, see the UGS documentation to learn how to get started with UGS.

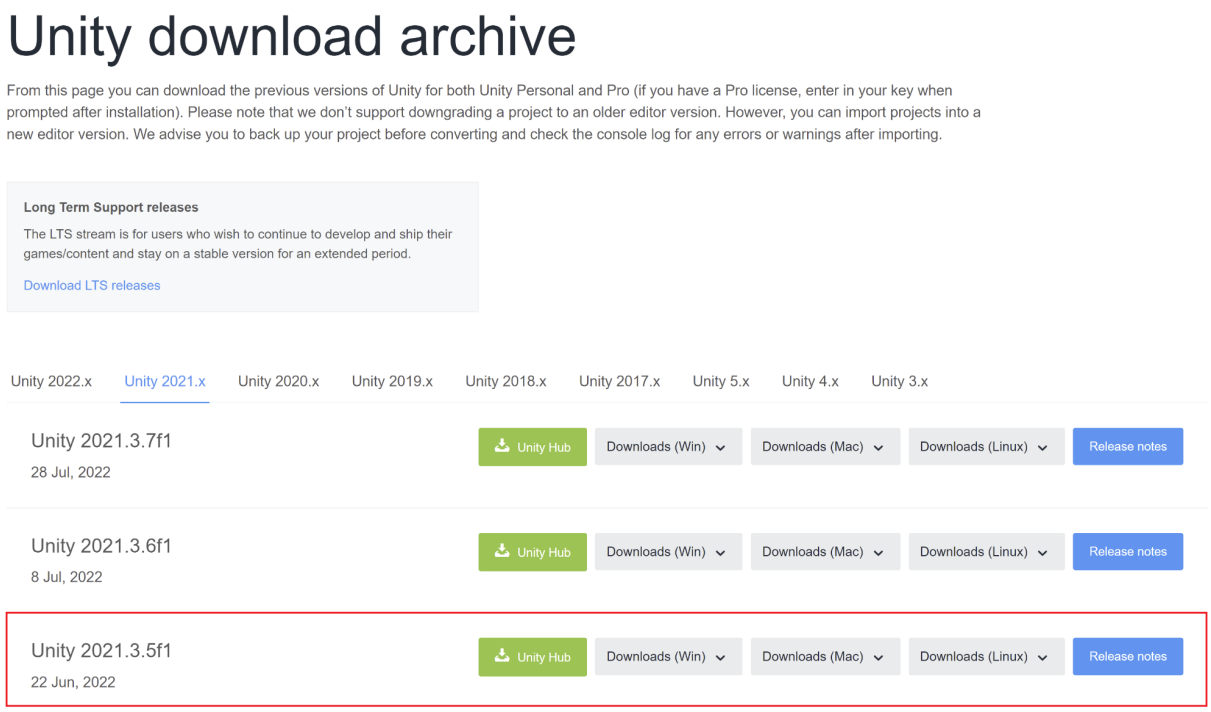
## Install the Unity Editor

To work with the Fusion BR200 project, you must use Unity Editor 2021.3.5f1. See Installing Unity to learn how to install the Unity Editor for your operating system. Use the

Archive section from the Unity Hub:

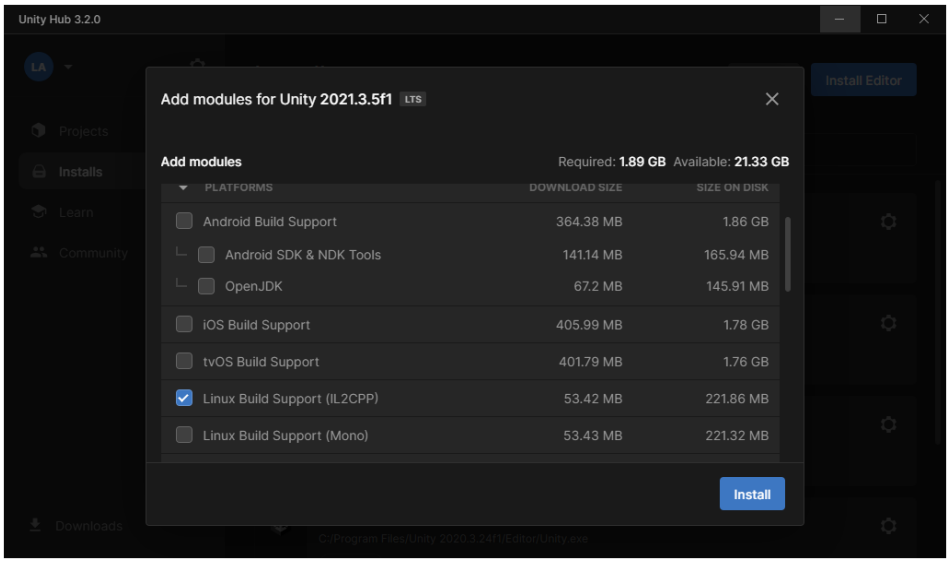


2. Select the **download archive** link to go to Unity's archive of Editor versions:




3. Select **Unity Hub**.

**Note**: When installing the Unity Editor, select **Linux Build Support IL2CPP** from the components list. Otherwise, you won't be able to build the standalone Linux binary.

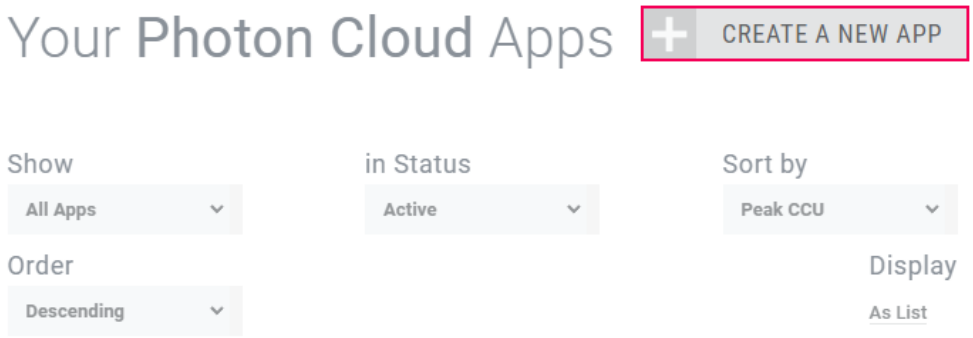


# Get started with Photon Fusion

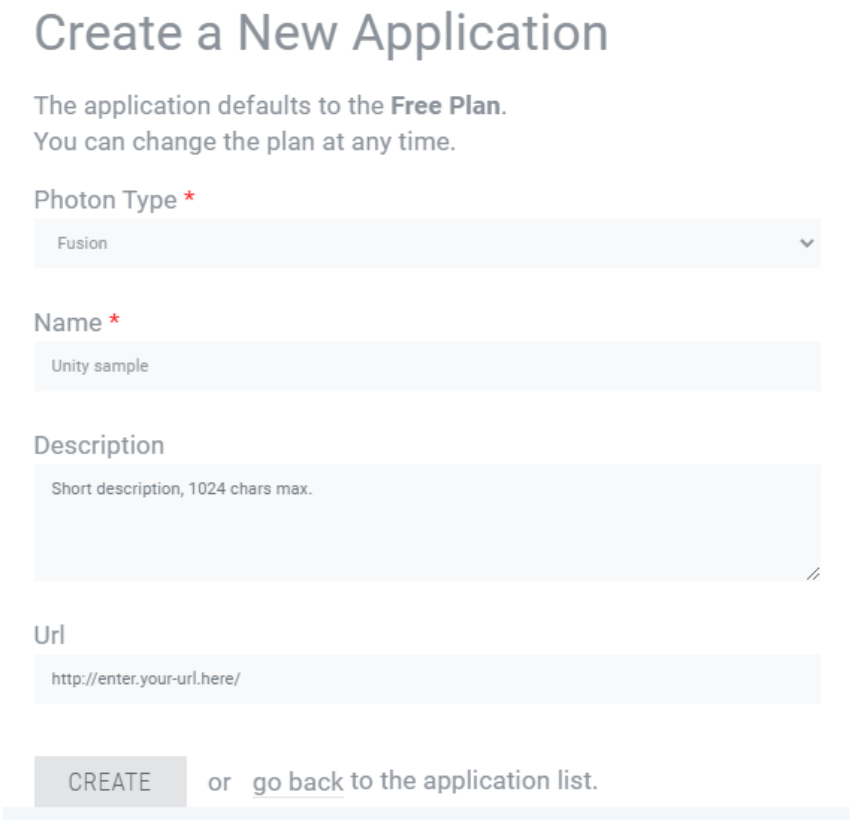
If you don't already have one, you'll need to create a Photon account to start using Photon Fusion. After you have an account, log into the Photon Dashboard and create a new Fusion application.

**Note**: See the Photon Fusion documentation if you have trouble getting started.

1. From the Photon Dashboard, select **Create a new app**.




2. Set **Photon Type** to **Fusion**.




3. Name the application.

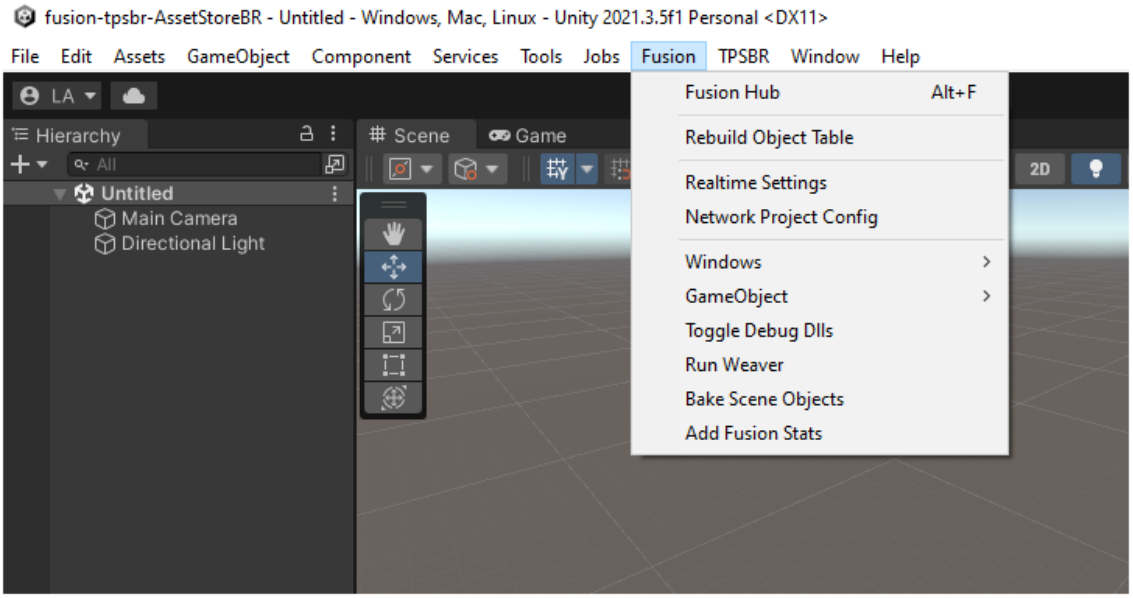
4. Optionally, provide a brief description and URL.
5. Select **Create**.

After creating the Fusion application, select it from the Photon Dashboard, then copy the **App ID**.

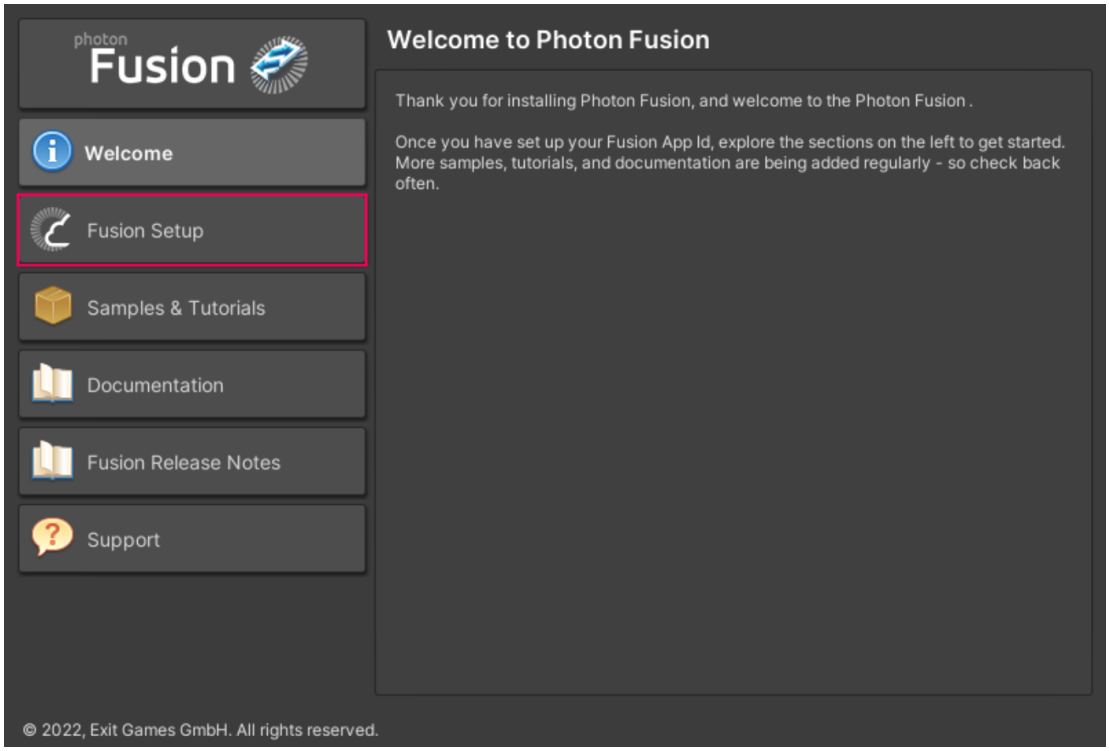

## Link the Photon Fusion project

Install the Fusion BR200 Project from the Unity Asset Store, then launch it in the Unity Editor.

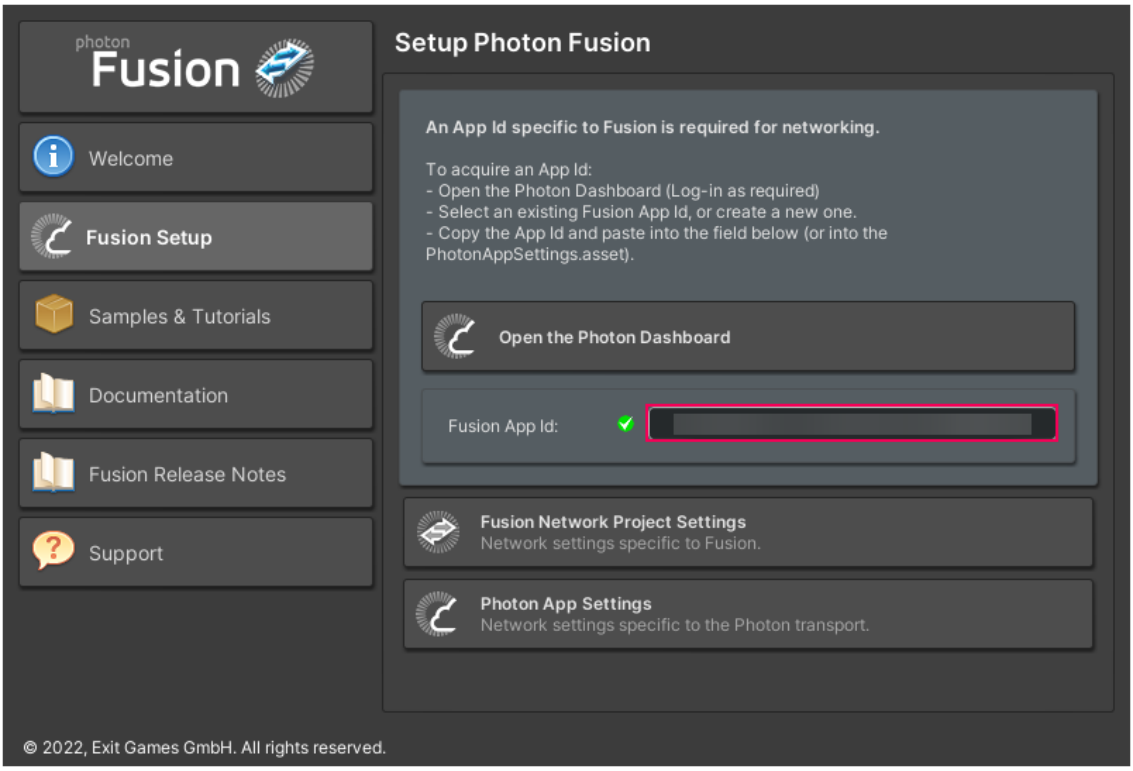
1. Launch the Fusion BR200 project in the Unity Editor.
2. Select **Fusion** > **Fusion Hub**.



3. Select **Fusion Setup**.




4. Paste the App ID you copied earlier into the **Fusion App Id** field.

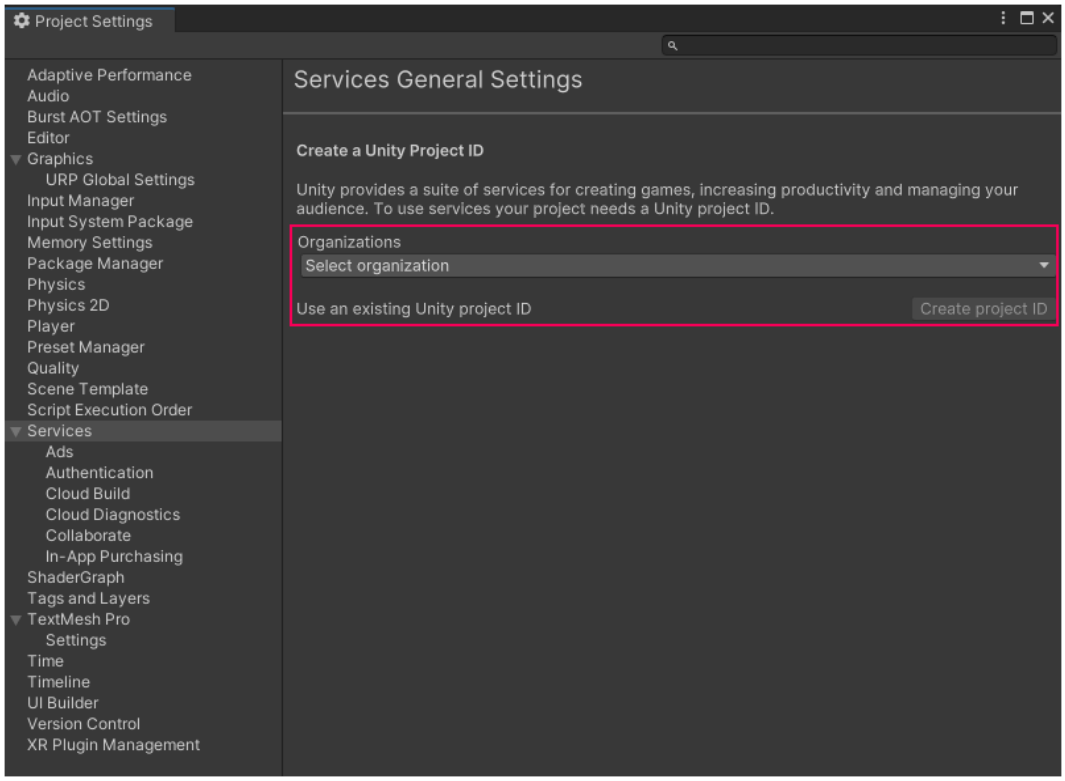


# Link your UGS project

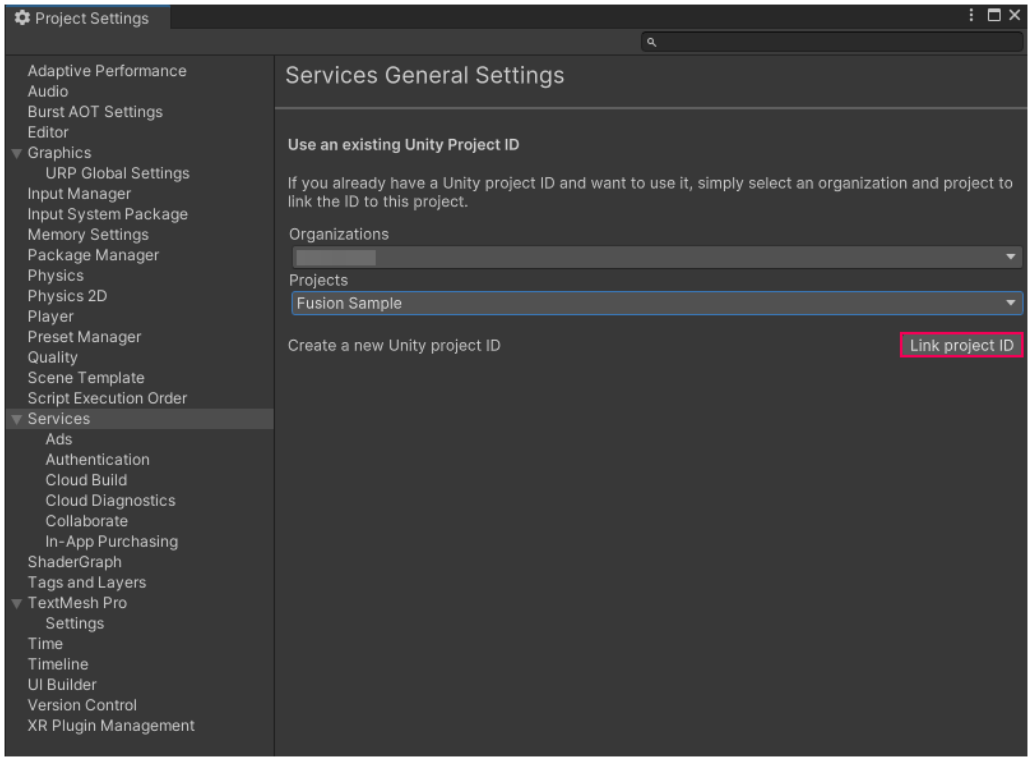
After the installation, link your UGS account and project with the Unity Editor.

1. Select **Edit** > **Project Settings** > **Services**.
2. If you already have a Unity project, select **Use an existing Unity project ID**. To create a project from the Unity Editor, select your **Organization**, then **Create project ID**.

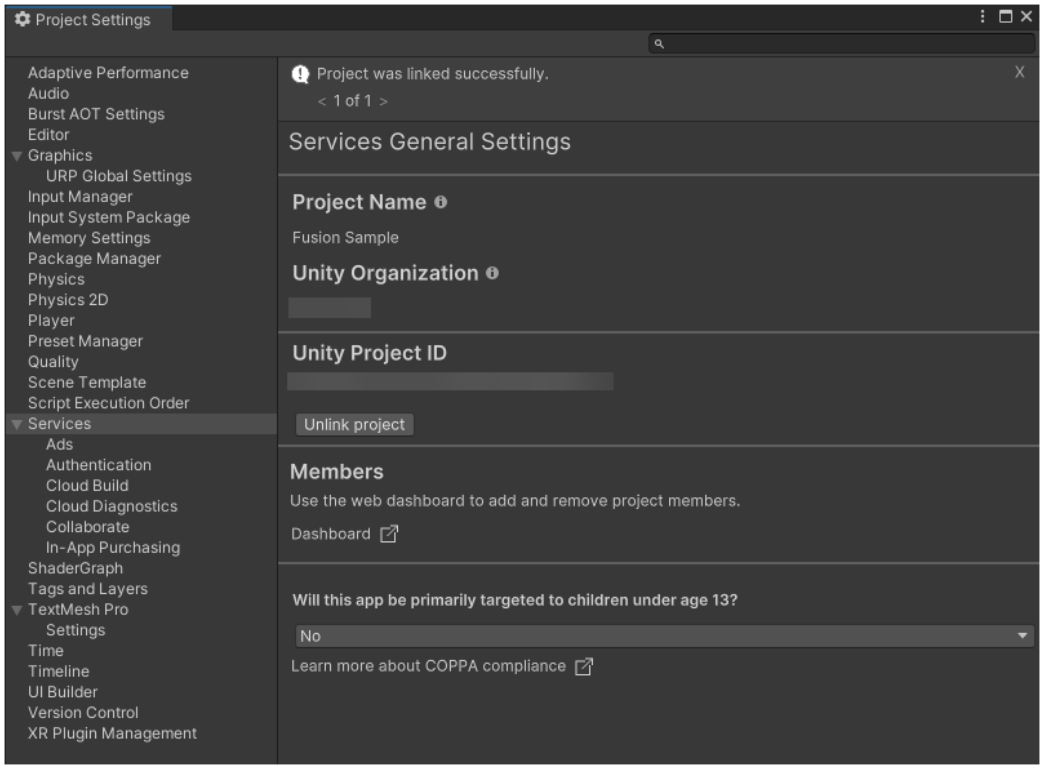
   **Note**: You can only create a project ID if you have adequate permission within the organization.



3. Select **Link project ID**.



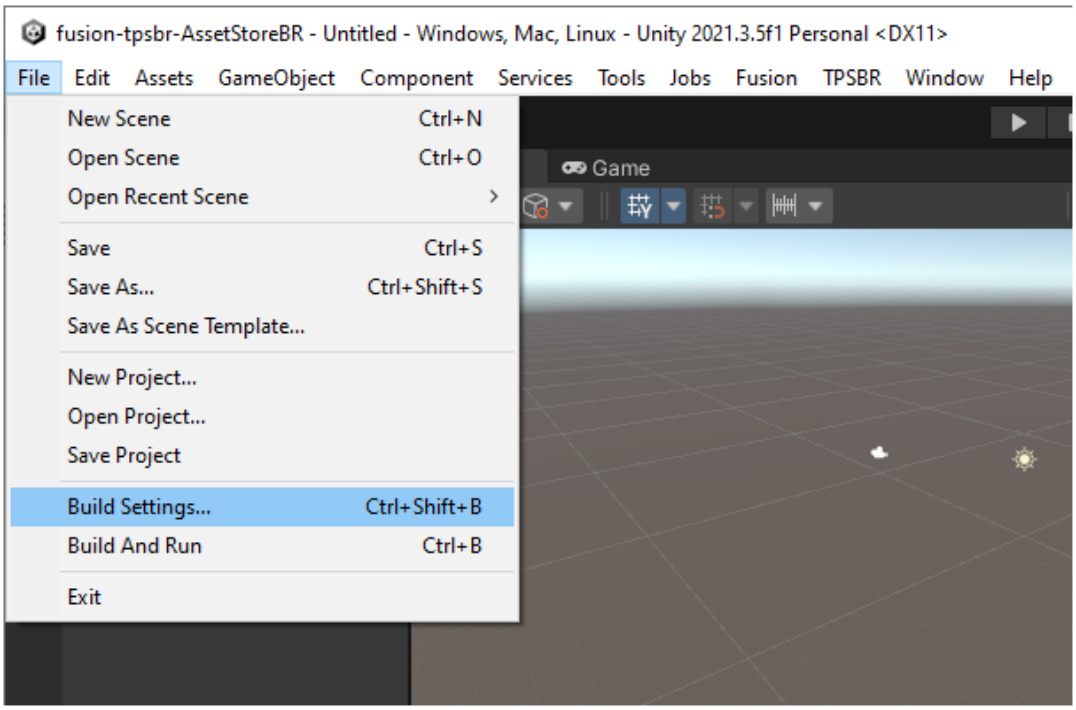

4. You should see a message stating that the project was linked successfully.



# Build the standalone server

After linking your UGS project and your Fusion App ID in the Unity Editor, you can build the standalone server binary to integrate with other Unity services.

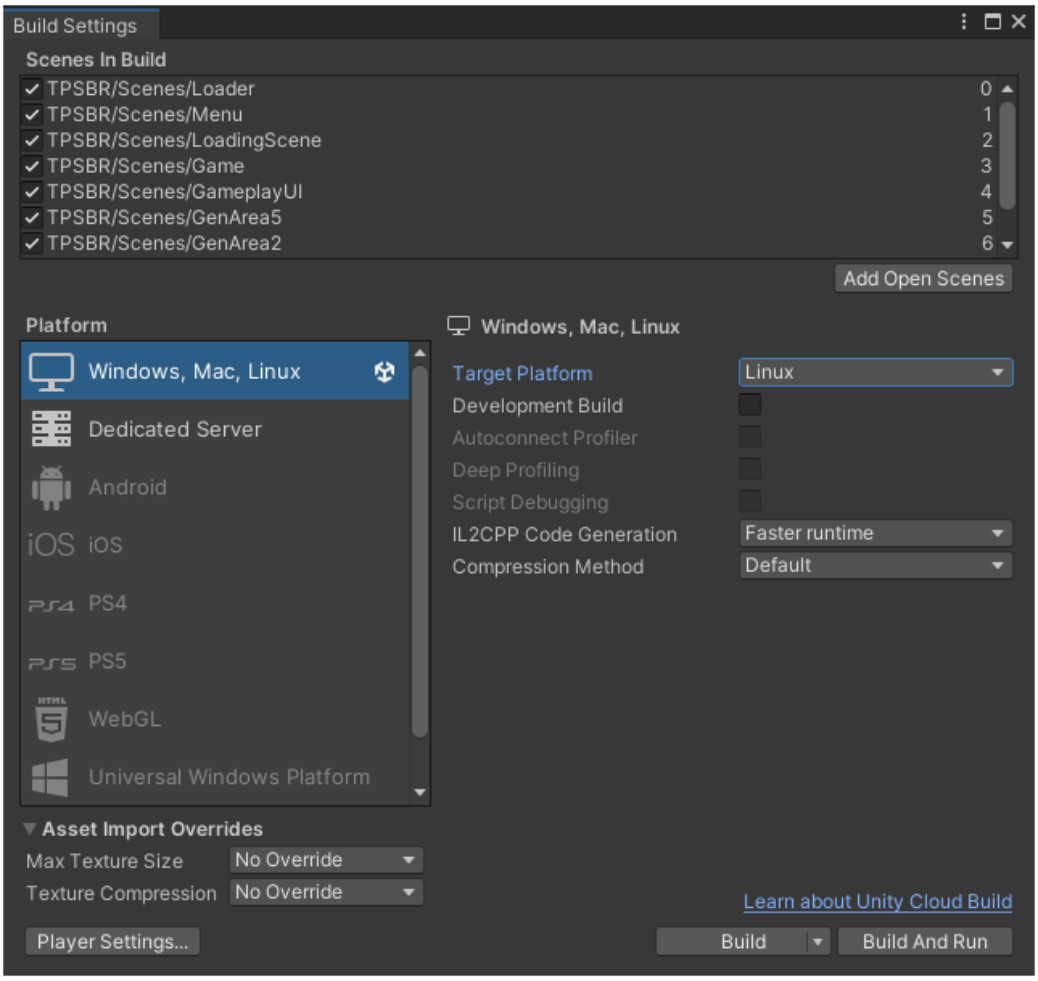
1. From the Unity Editor, go to **File** > **Build Settings...**.




2. Select **Windows, Mac, Linux** for the Platform.

   **Warning**: There are multiple reasons to target the Dedicated Server platform, such as asset stripping. However, this sample was not built specifically for targeting a Standalone Linux IL2CPP build. See Dedicated Server target for more information about Dedicated Server mode"

3. Set the **Platform** to **Linux**.




4. Select **Build**.
5. Save the build in a location that's easy to find. You'll need it when you configure Multiplay.

# Add the Multiplay Manager

The Fusion BR200 supports using Multiplay to host game servers. Follow the instructions below to add the Multiplay service to the sample project.

**Warning**: Multiplay is a pay-as-you-go service with a free tier. You must sign up for UGS services with a credit card to start using Multiplay. If you exceed the free tier usage allowance, you will be charged. See our Billing FAQ to learn more.

# Enable Multiplay

**Note**: You must be an Owner or Manager of your organization to enable Multiplay.

1. Sign in to the Unity Dashboard with your Unity account.
2. From the Unity Dashboard, go to **Multiplayer** > **Multiplay**.
3. Select **Set up Multiplay**.

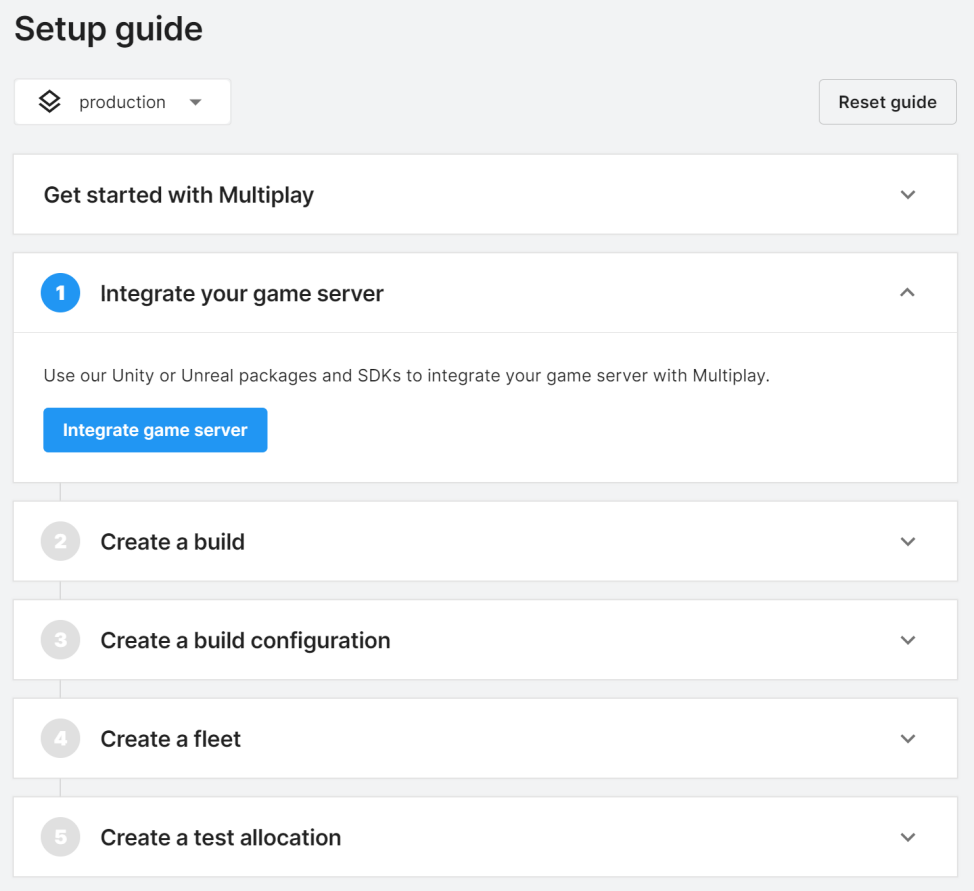
**Note**: You might need to add your credit card information before continuing. Multiplay is a pay-as-you-go service with a free usage tier. If you exceed the free usage, you will be charged. See Unity Gaming Services Pricing.

4. Wait for the Unity Dashboard to finish enabling Multiplay for your project.
5. Follow the integrated Setup Guide, starting with integrating your game server.

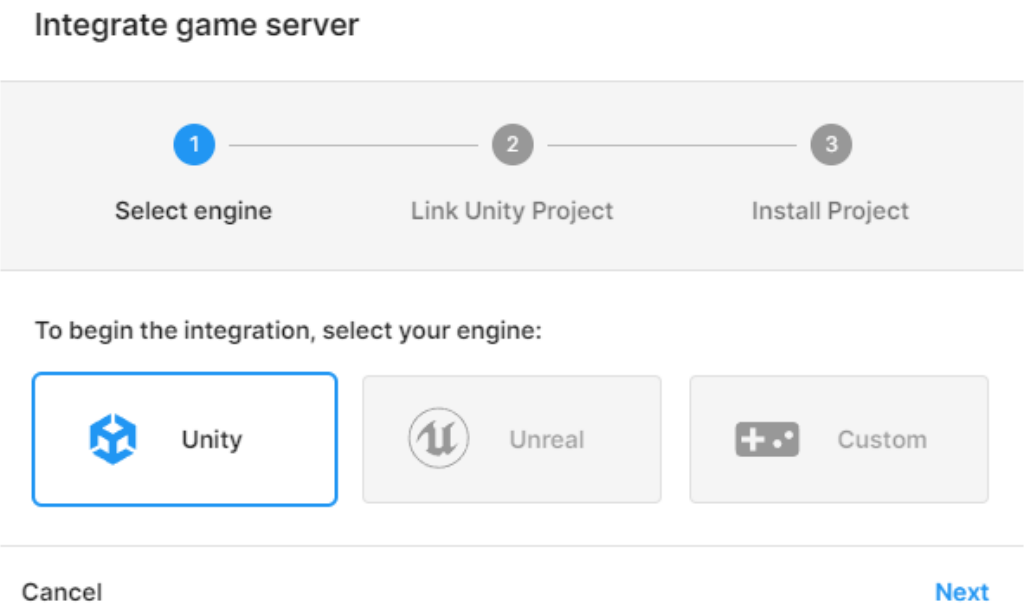
# Integrate game server

The first step is integrating Multiplay with your game through the Unity Editor. You should have completed most of this step in Link your UGS project.

1. Select **Integrate game server**.




2. Select **Unity** as the engine.



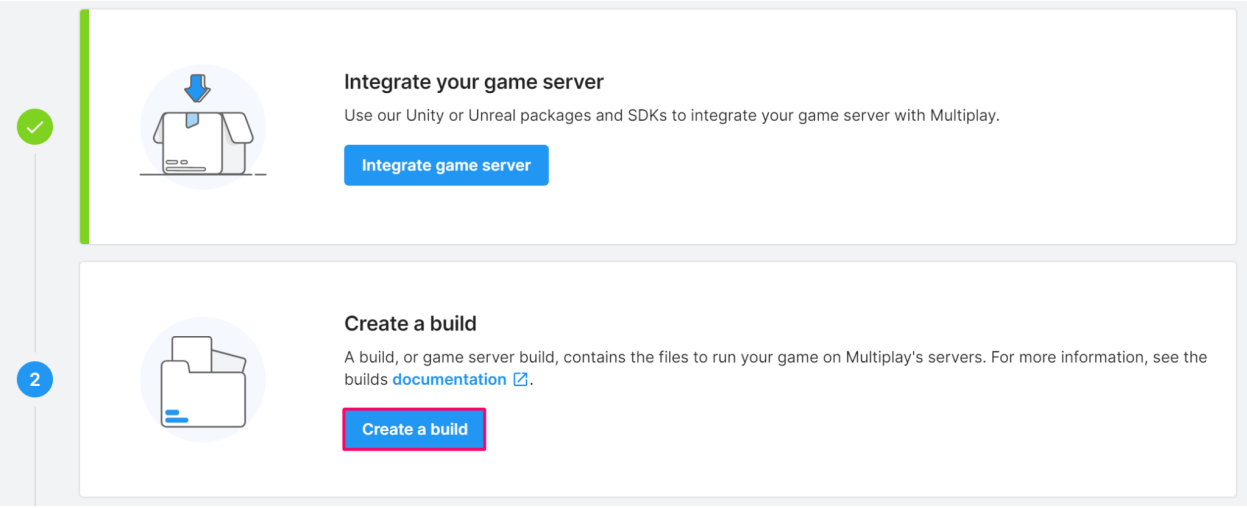
3. Select **Next** if you've already linked your Unity project with the Unity Editor.
4. Select **Finish**.

**Note**: You can skip the Install Project step because the SDK should already be installed.

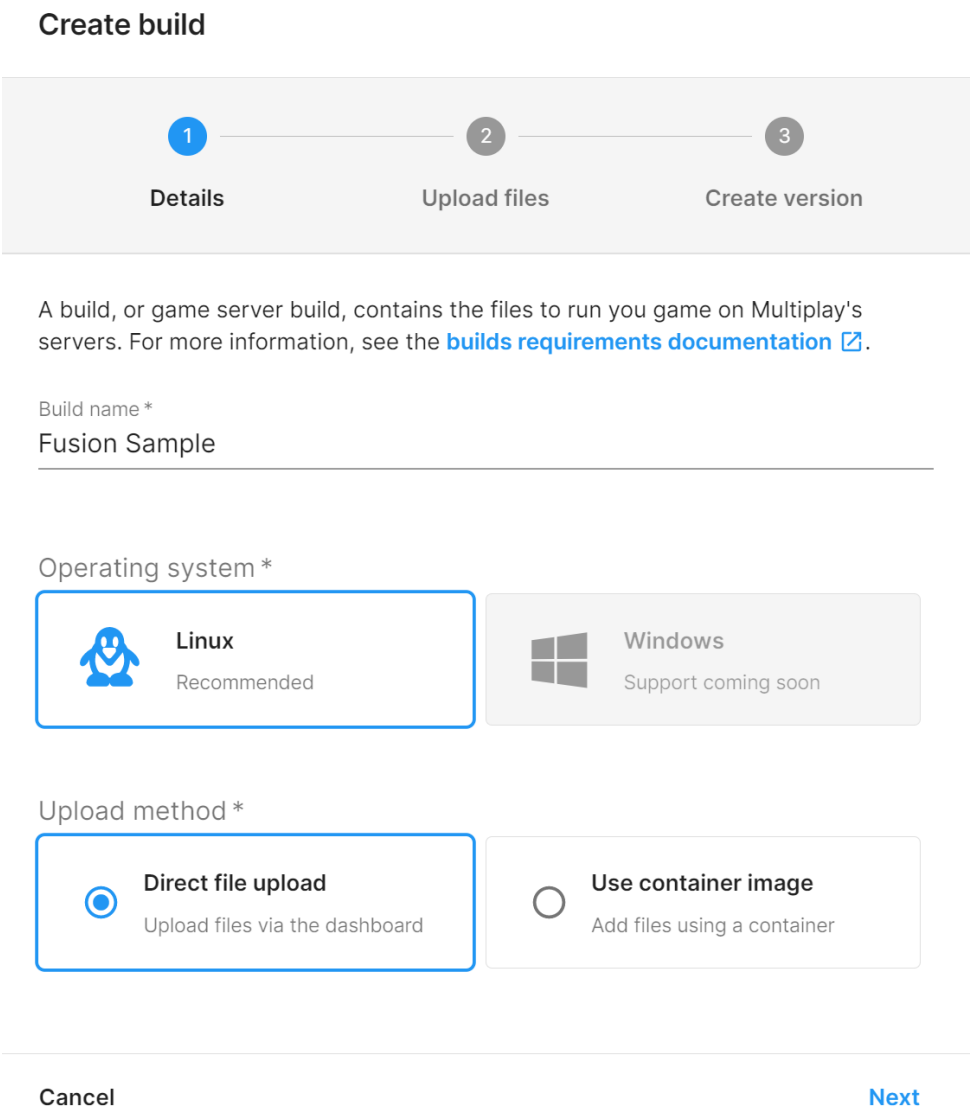
# Create a build

Create a build of your game within the Multiplay service. See the Build documentation to learn more.

1. Select **Create a build**.

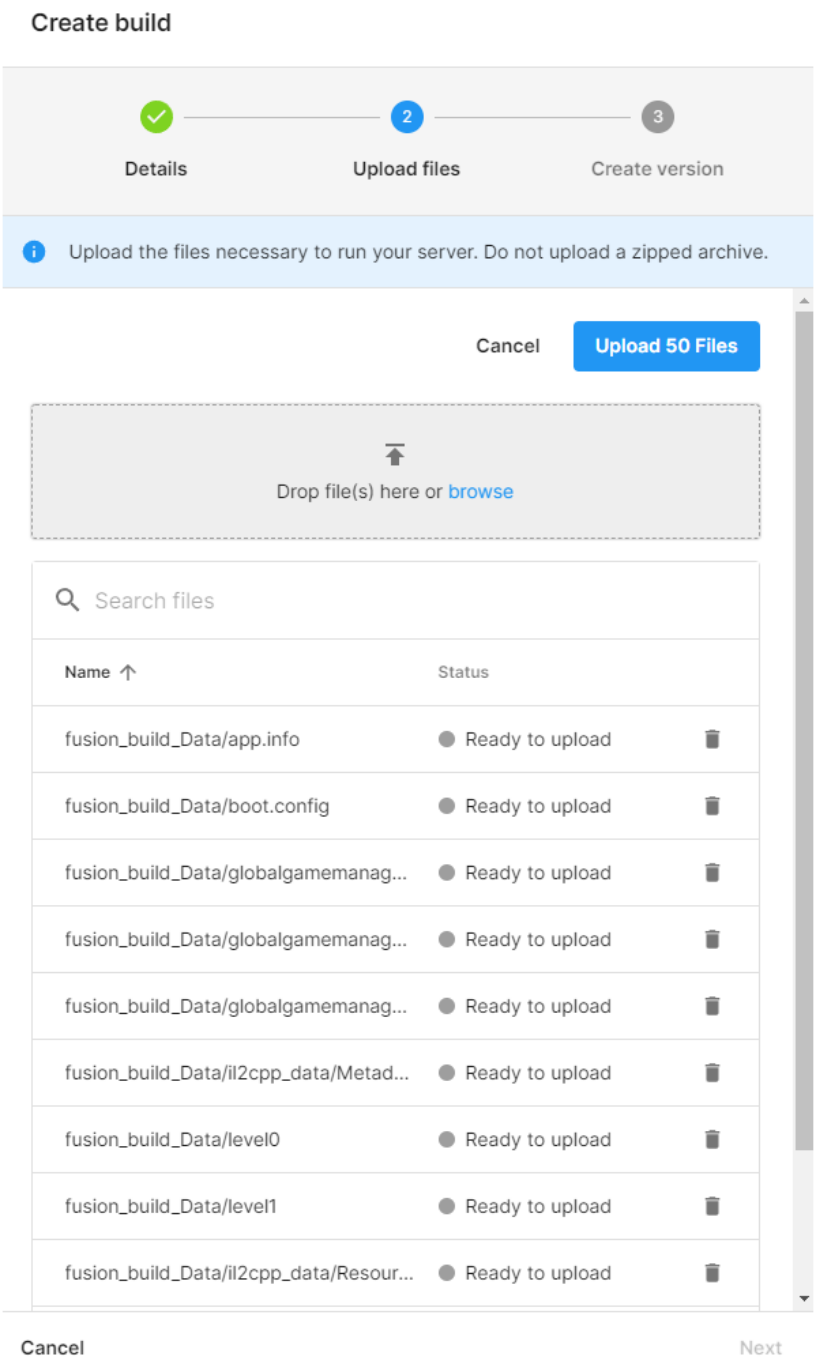



2. Give the build a name, select **Linux** as the operating system, and select **Direct file upload**.

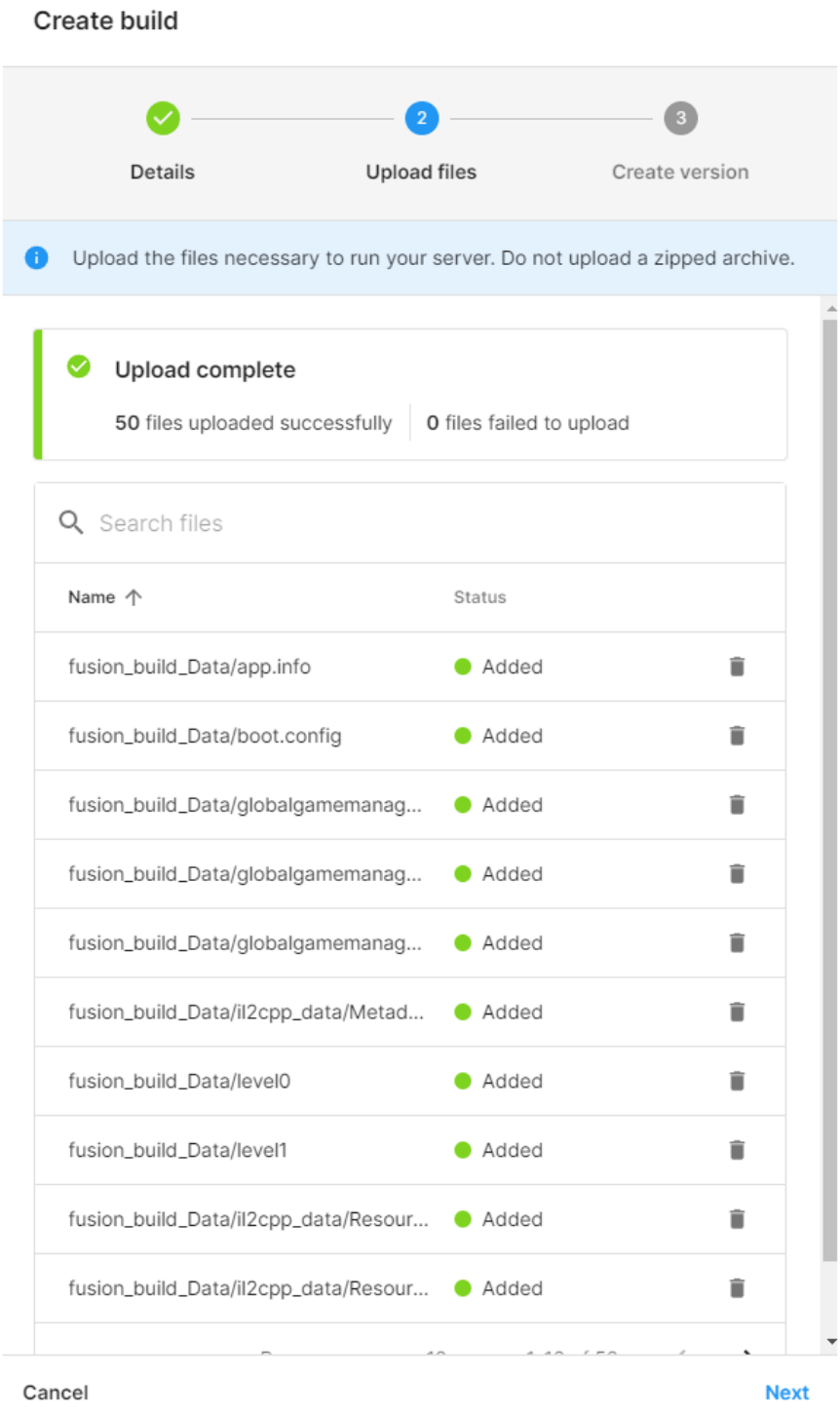



3. Select **Next**.
4. Upload the following files from the build you created in the Unity Editor using **drag-and-drop**:
    a. The .so files
    b. The .x86_64 file
    c. The `*_Data` folder

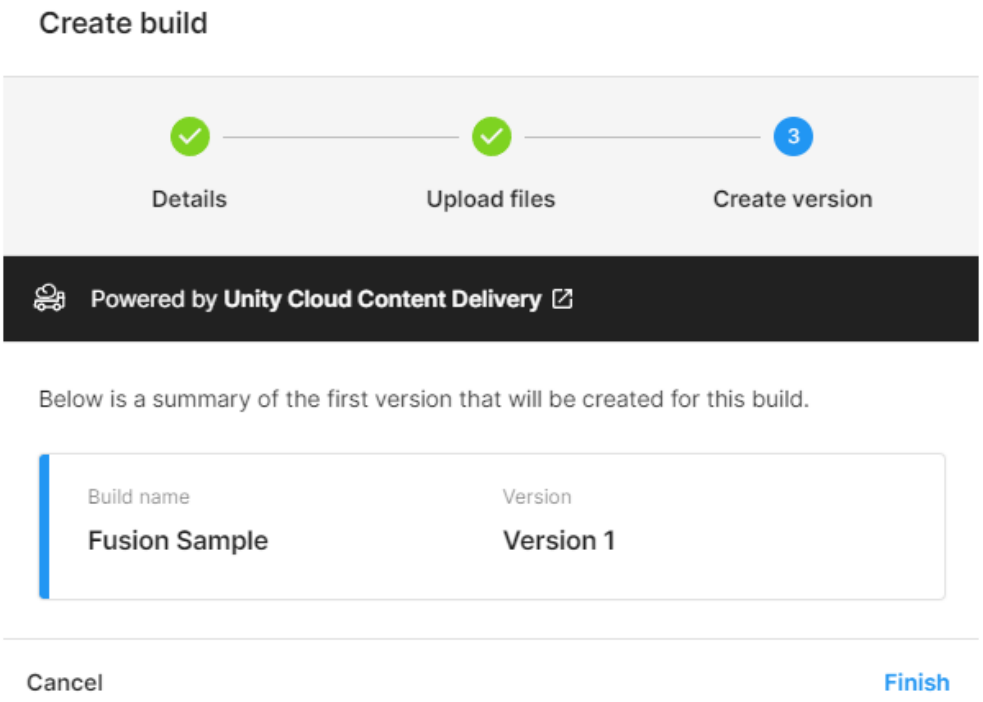
5. Select **Upload Files**.



6. Select **Next**.



7. Select **Finish** to create your first release.

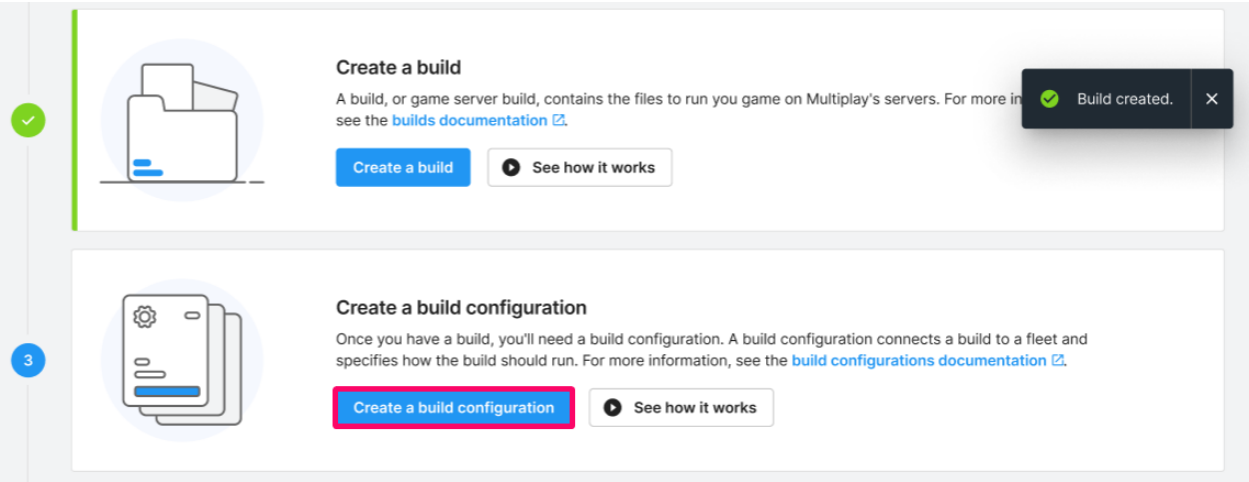



## Create a build configuration

Create a build configuration for the build you created in the previous step. See the [Build configuration documentation]() to learn more.

**Warning**: You won't be able to select the build executable for the build you created in the previous step until the files finish syncing.

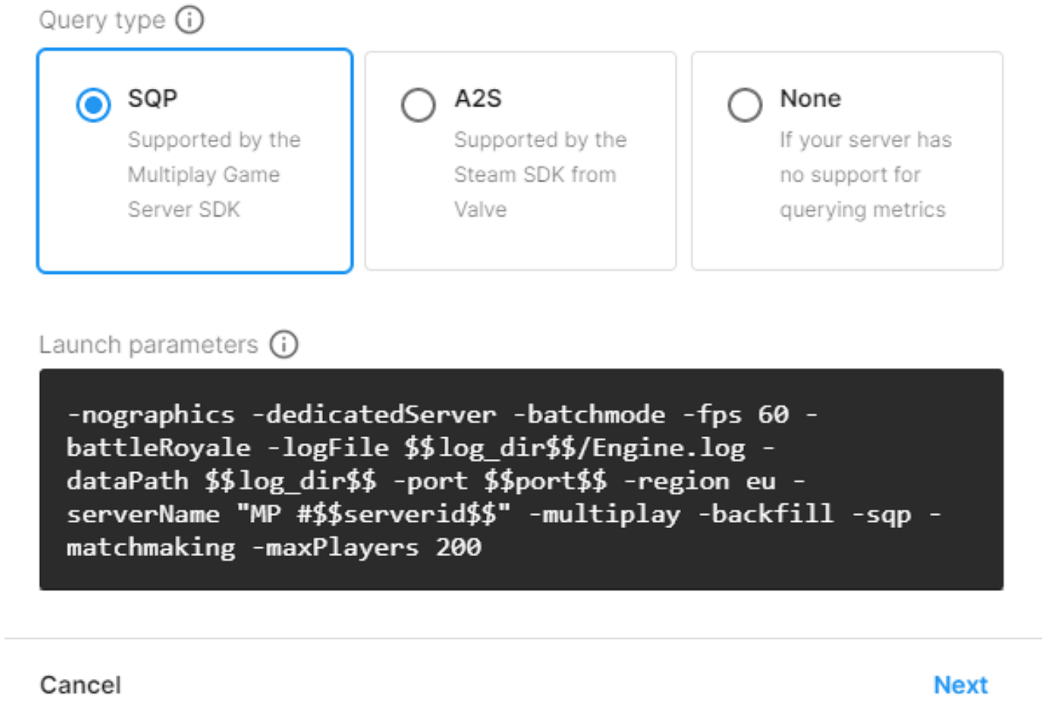
1. Select **Create a build configuration**.




2. Fill in the build configuration details.
    a. Name the build configuration.
    b. Select the build you created in the previous step.
    c. Select the build executable.
    d. Set the **Query type** to **SQP**.
    e. Enable **Custom launch parameters**, then use the following launch parameters *(**Note**: when copy pasting the lines below from this PDF, make sure to remove the line breaks so the launch parameters below stand as one line):*

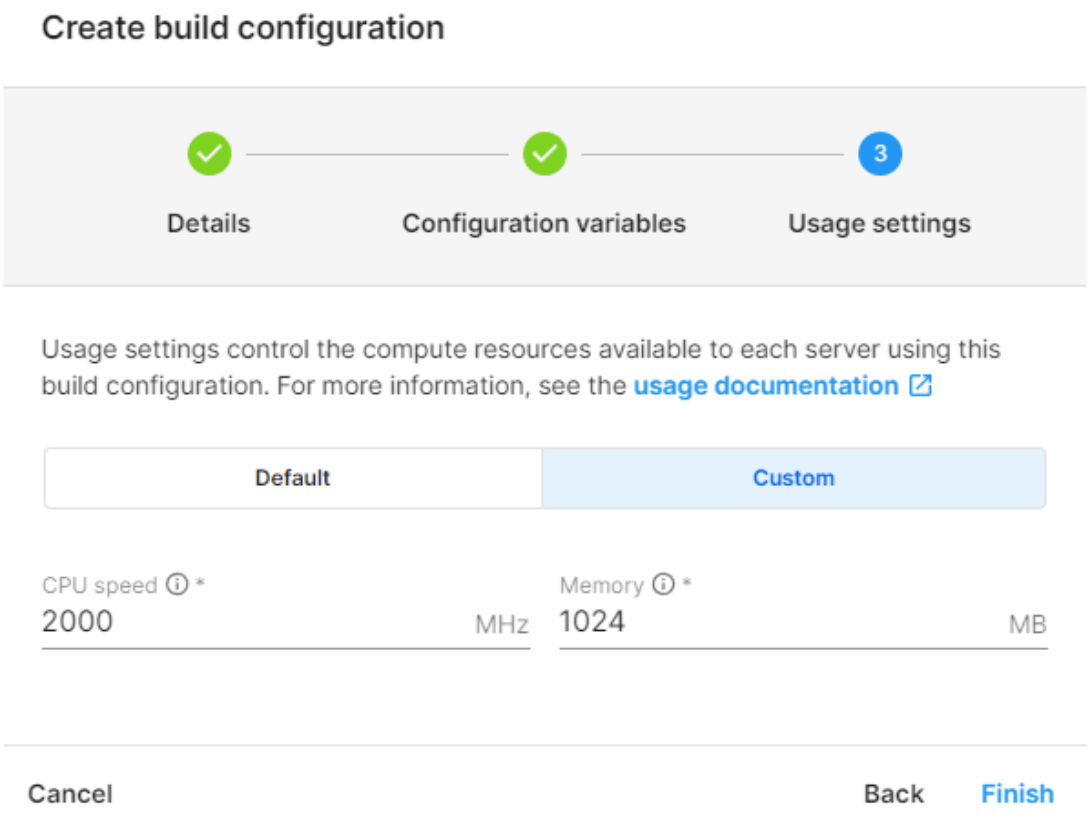
```
-nographics -dedicatedServer -batchmode -fps 60
-battleRoyale -logFile $$log_dir$$/Engine.log -dataPath
$$log_dir$$ -port $$port$$ -region eu -serverName "MP
#$$serverid$$" -multiplay -backfill -sqp -matchmaking
-maxPlayers 200
```

3. Select **Next**.




4. Select **Next** again on the Configuration variables step.
5. Define the usage settings:
    a. Select **Custom**.
    b. Set the **CPU speed** to **2000 MHz**.

c. Set the **Memory** to **1024 MB**.

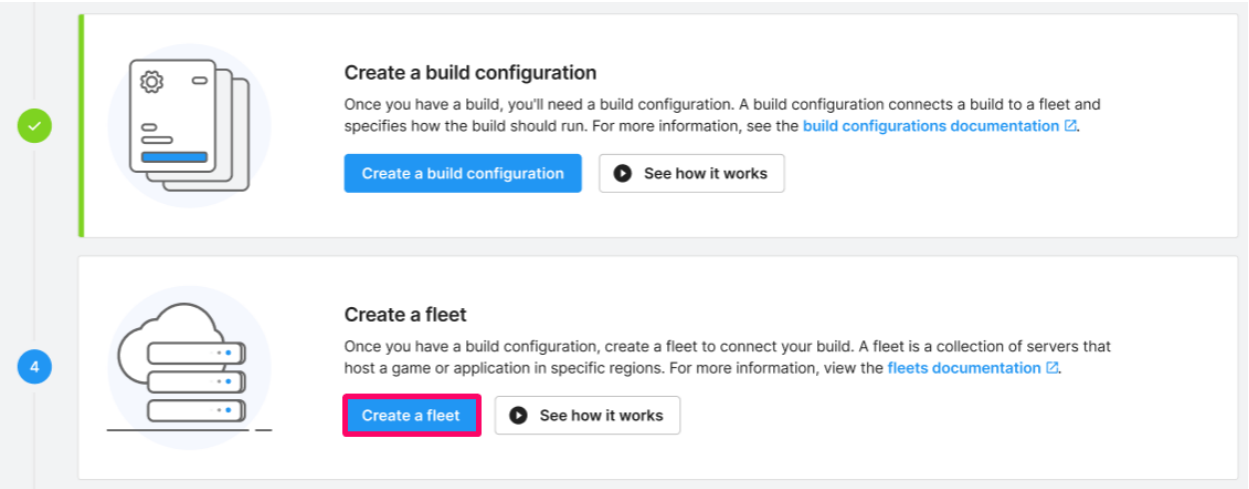



6. Select **Finish**.

## Create a fleet

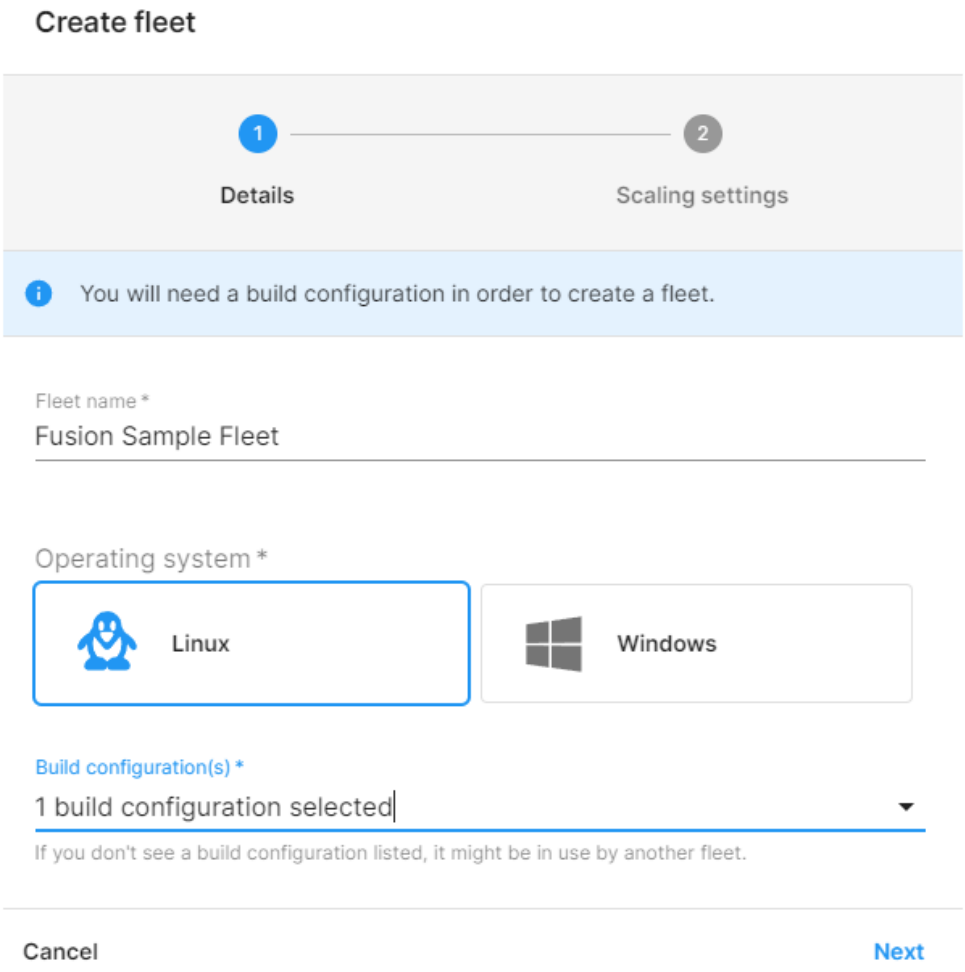
Create a fleet to host your game servers. See the Fleet documentation to learn more.

1. Select **Create a fleet**.




2. Fill in the fleet details:
    a. Name the fleet.
    b. Set the **Operating system** to **Linux**.
    c. Select the build configuration you created in the previous step.

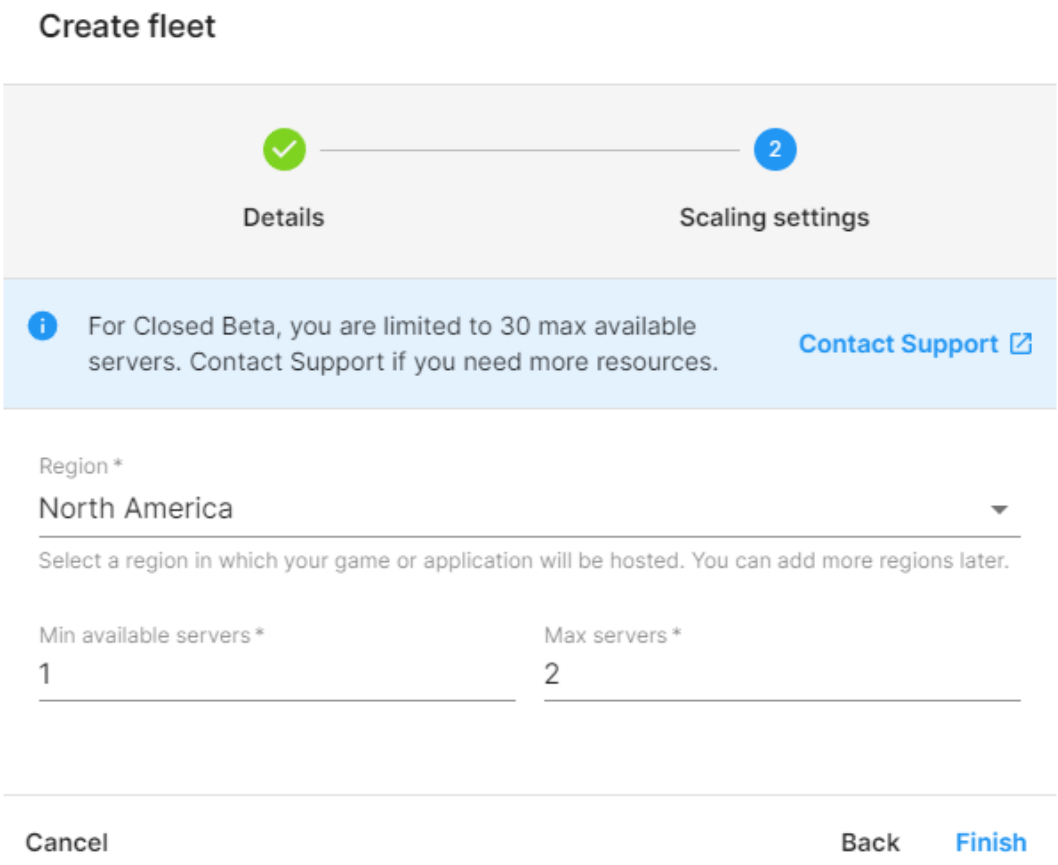
3. Select **Next**.




4. Specify the scaling settings:
    a. Select a region.
    b. Set the **Min available servers** to a value less than or equal to 1.
    c. Specify the **Max servers** to a value equal to or greater than the Min available servers value.

    **Note**: You must set **Max servers** to a value greater than 1. Otherwise, you won't be able to create a game session.

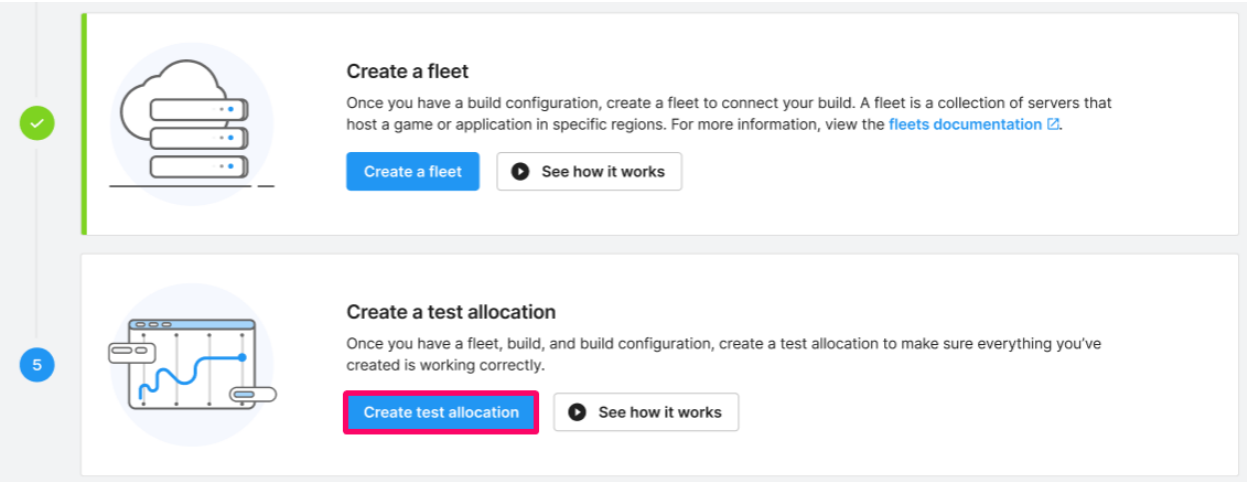
5. Select **Finish**.



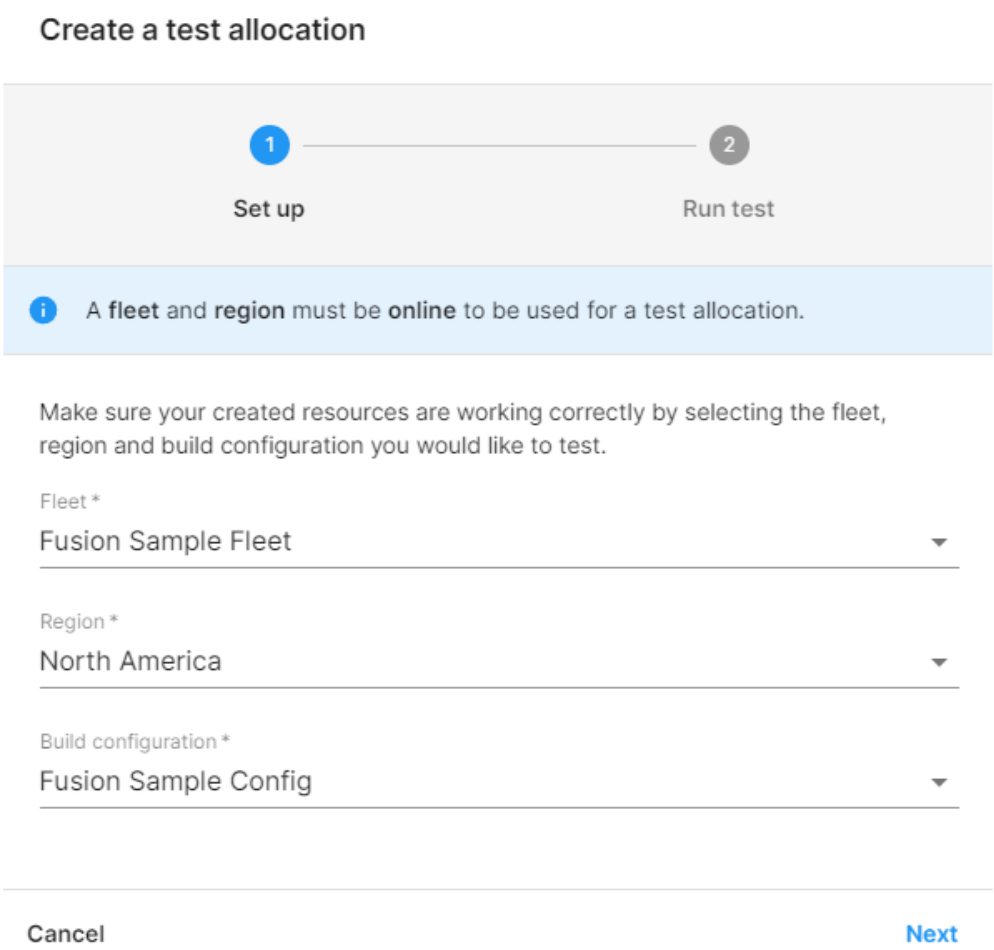

## Create a test allocation

Create a test allocation to make sure everything's working correctly. See the [Allocation documentation] for help.

1. Select **Create a test allocation**.

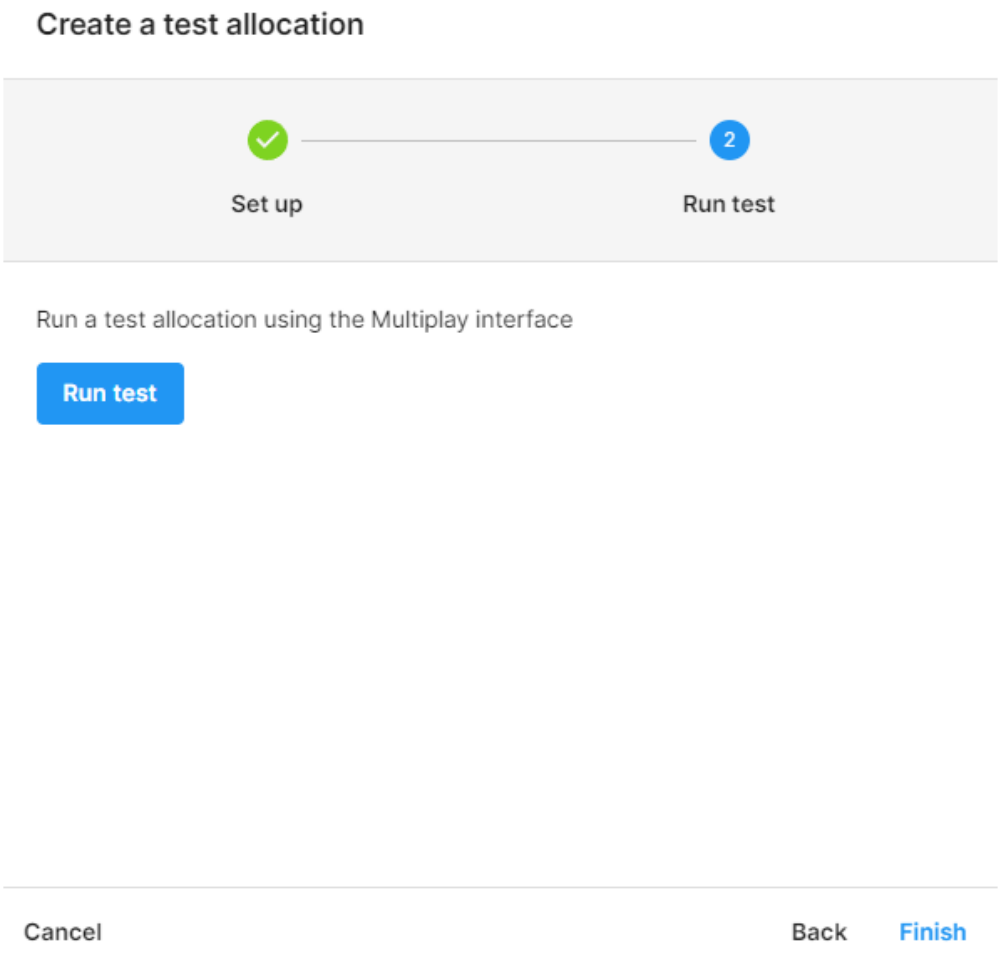



2. Select the **Fleet**, the **Region**, and the **Build configuration**.

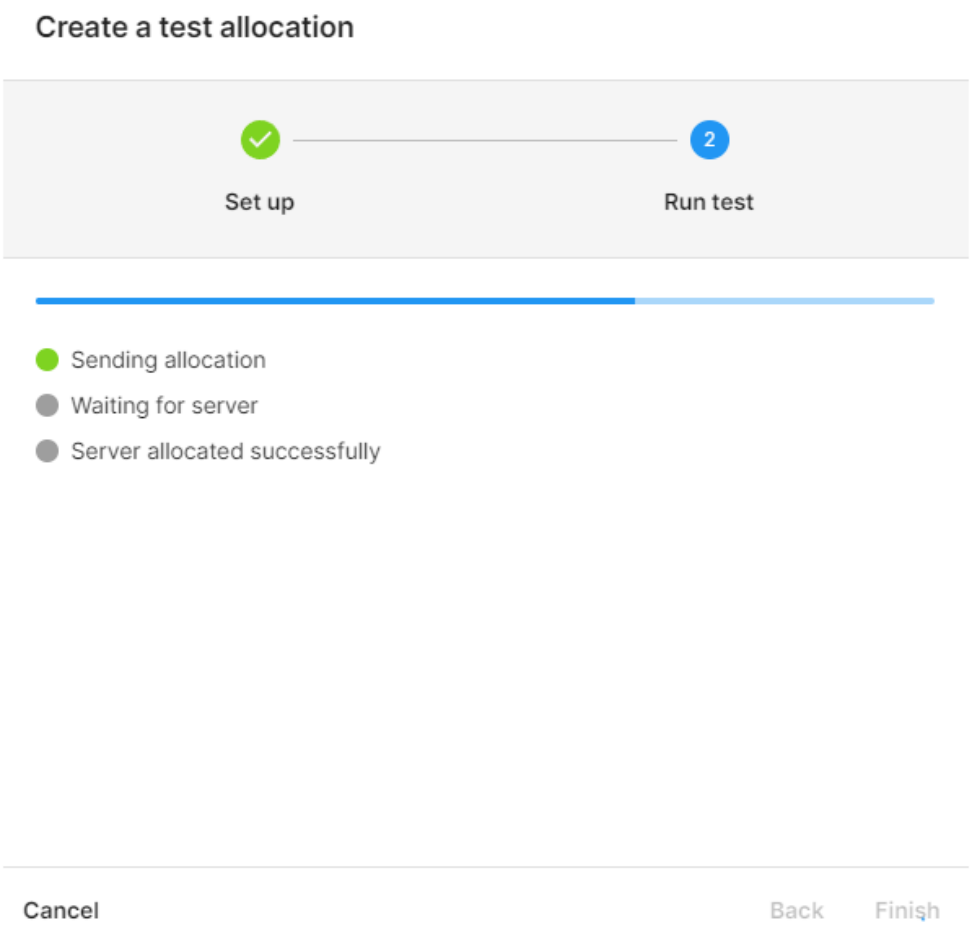



3. Select **Next**.

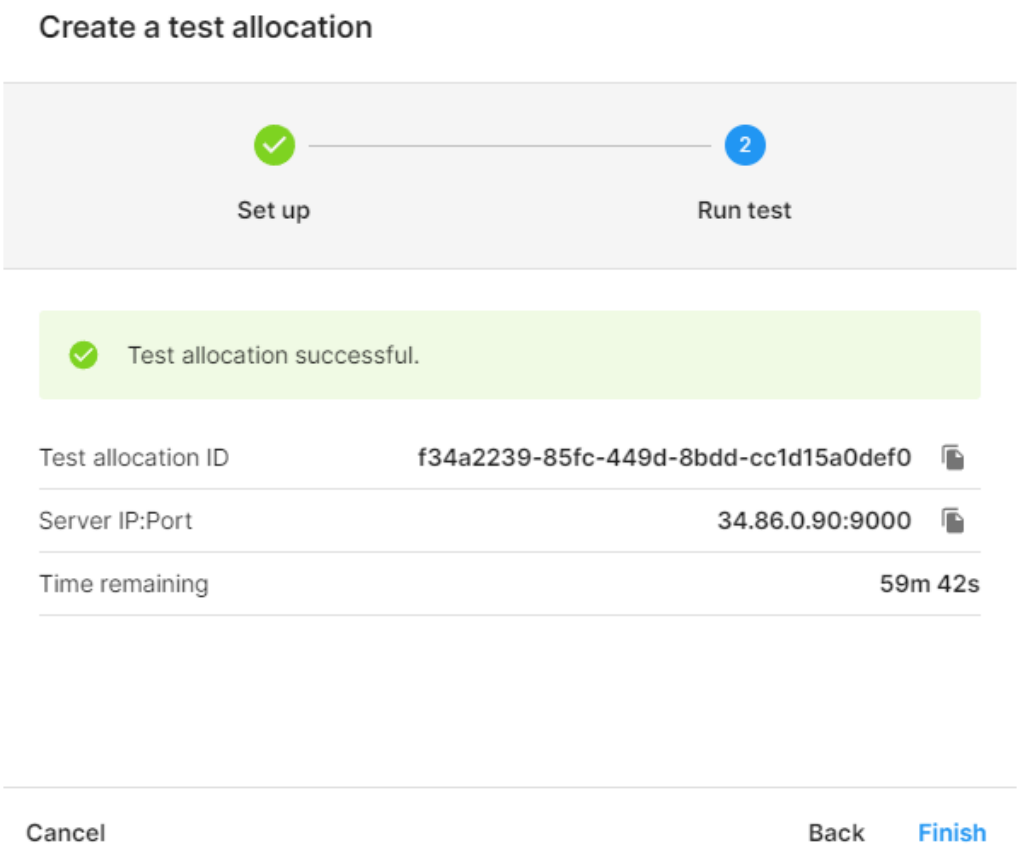
4. Select **Run test**.



5. Wait for the test to complete.



6. Select **Finish**.




Congratulations! You've successfully set up Multiplay with the Fusion BR.

# Add the Unity Matchmaker

The Fusion BR200 project supports using the Unity Matchmaker. Follow the instructions below to add the Unity Matchmaker service to the sample project.

## Enable Matchmaker

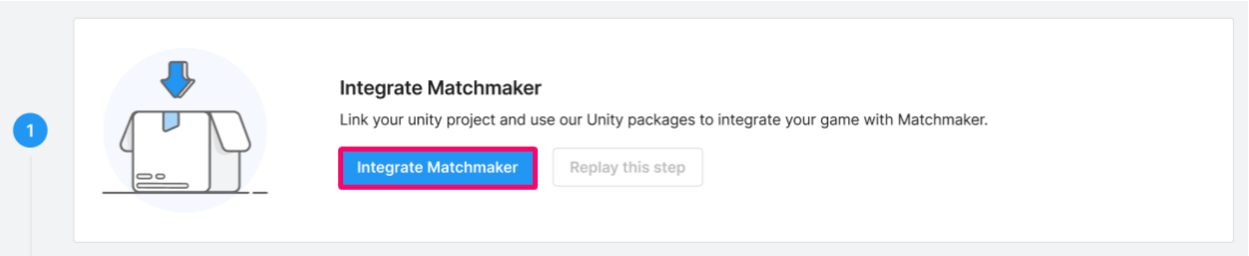
**Note**: You might need to enter payment information to continue the trial. If prompted, enter your payment information, then select **Complete onboarding**.

1. Sign in to the Unity Dashboard with your Unity account.
2. From the Unity Dashboard, go to **Multiplayer** > **Matchmaker**.
3. Select **Set up Matchmaker**.
4. Use the Setup Guide, starting with the **Integrate Matchmaker** step.

# Integrate Matchmaker

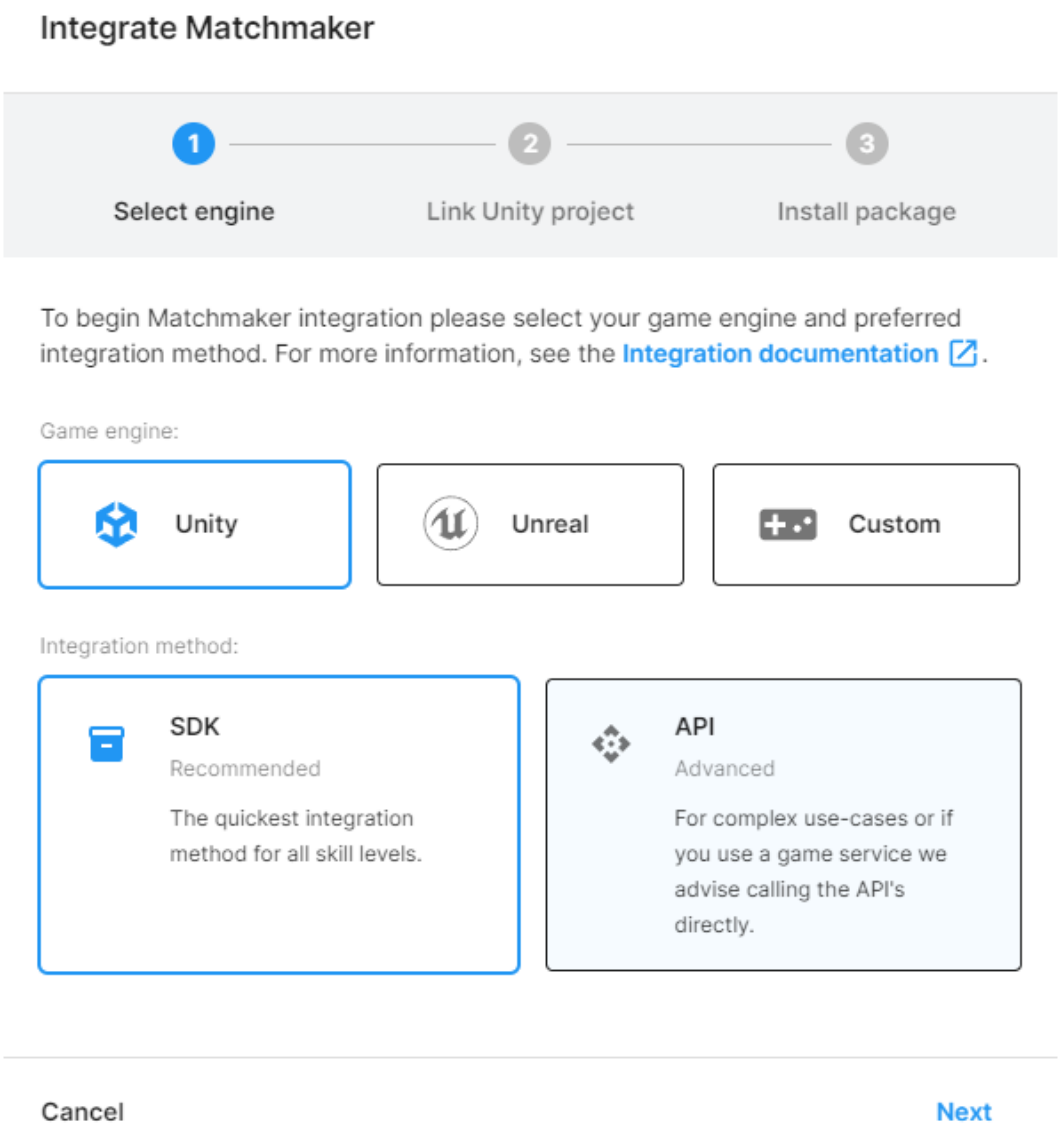
The first step is integrating Matchmaker with your game through the Unity Editor. You should have completed most of this step in Link your UGS project. See the Matchmaker integration and tools documentation for help.

1. Select **Integrate Matchmaker**.




2. Set the **Game engine** to **Unity**.

3. Set the **Integration method** to **SDK**.

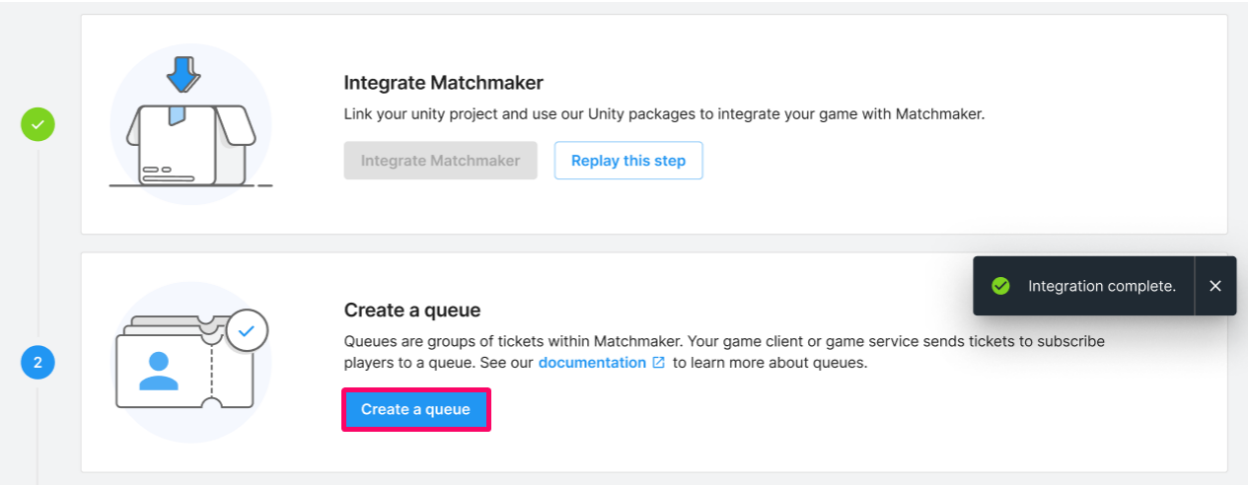



4. Select **Next**.
5. Select **Next** again for the Link Unity project step. If you haven't already linked your project, see Link your UGS project.
6. Skip the Install the Matchmaker package. The Fusion BR200 project already includes the package.
7. Select **Finish**.

# Create a queue

Create a queue for your game. See the Queues and Pools documentation for help.

1. Select **Create a queue**.




2. Name the queue "**battleRoyale**".

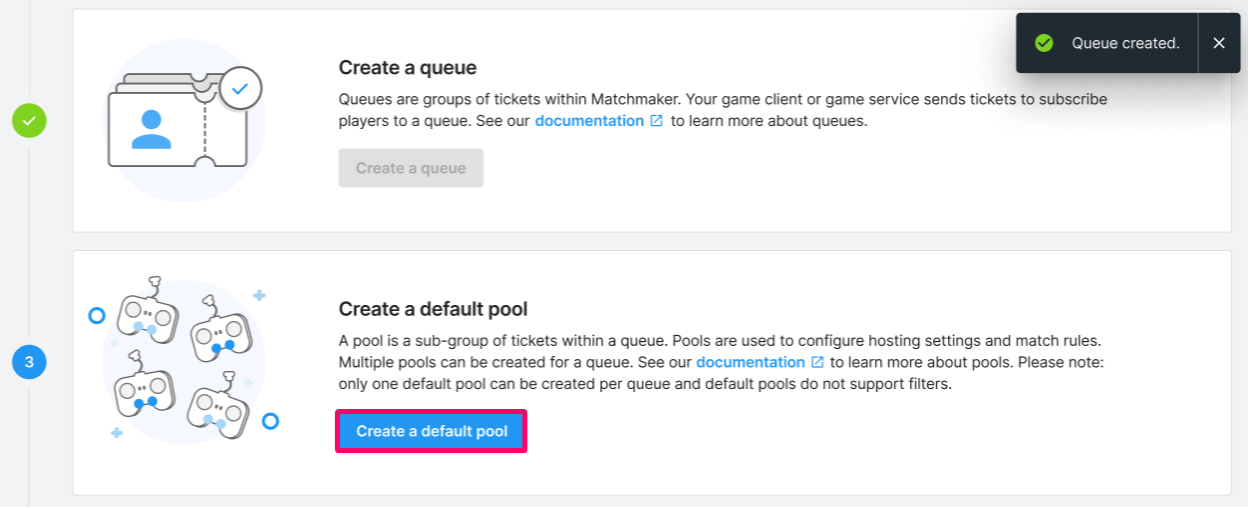
   ***Note**: The system is case sensitive, and using a queue name other than "**battleRoyale**" results in an exception.*

3. Set the **Maximum players on a ticket** to **2**.
4. Select **Create**.

# Create a default pool

Create a default pool for your game. See the [Queues and Pools documentation] for help.

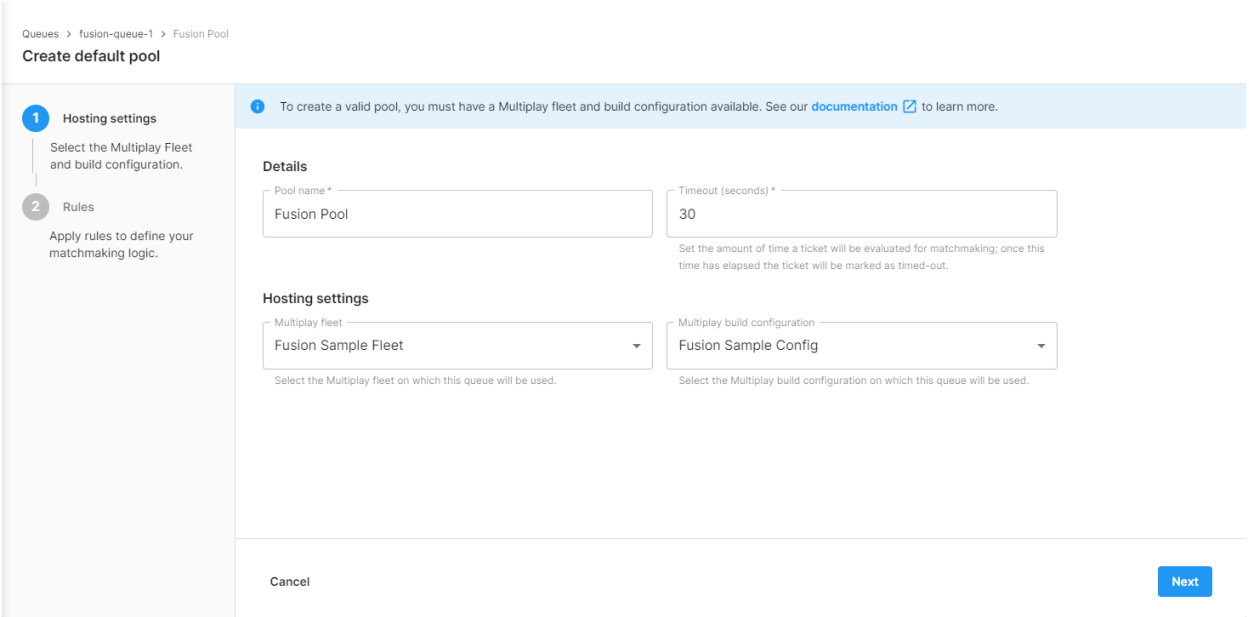
1. Select **Create a default pool**.



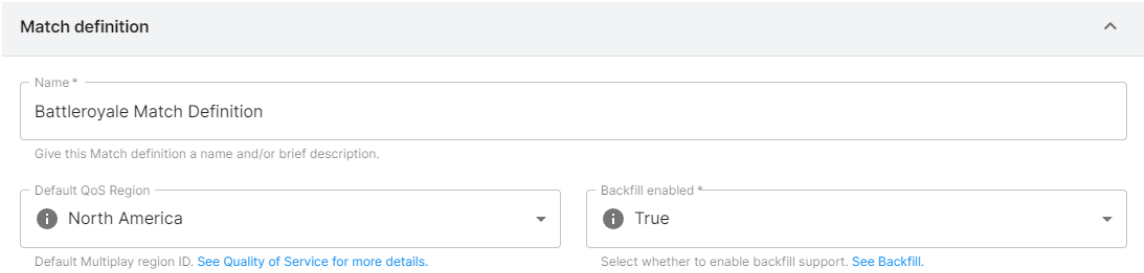

2. Fill in the **Hosting settings**:
   a. Give the pool a name.

b. Set the timeout to **30** seconds.
c. Select the Multiplay fleet you created earlier.
d. Select the Multiplay build configuration you created earlier.

3. Select **Next**.




4. Configure the **Rules**:
   a. Set the Match definition name to **Battleroyale Match Definition**.
   b. Select the **Default QoS Region**. This should be the region you selected for your fleet when you set up Multiplay.
   c. Set **Backfill enabled** to **True**.




   d. Finish configuring the remaining rule settings
      i. Set **Min teams** to **1**.
      ii. Set **Max teams** to **1**.
      iii. Set **Min players** to **1**.
      iv. Set **Max players** to **200**.

      **Note**: You must go back to Multiplay and configure the launch parameters on the build configuration to reflect the maximum number of players you set here. See Manage build configurations for

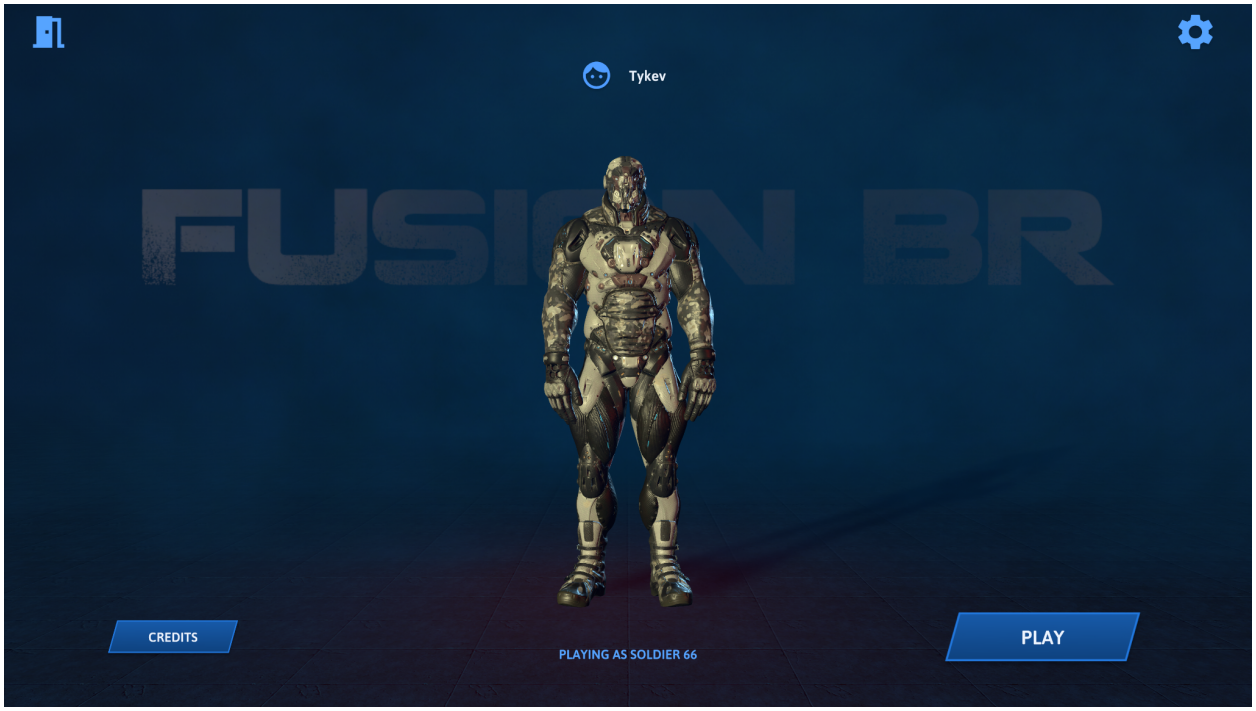
help.

5. Select **Create**.

Congratulations! You've successfully configured the Unity Matchmaker. You can go to **Multiplayer** > **Matchmaker** > **Overview** to view matchmaking traffic and match times.

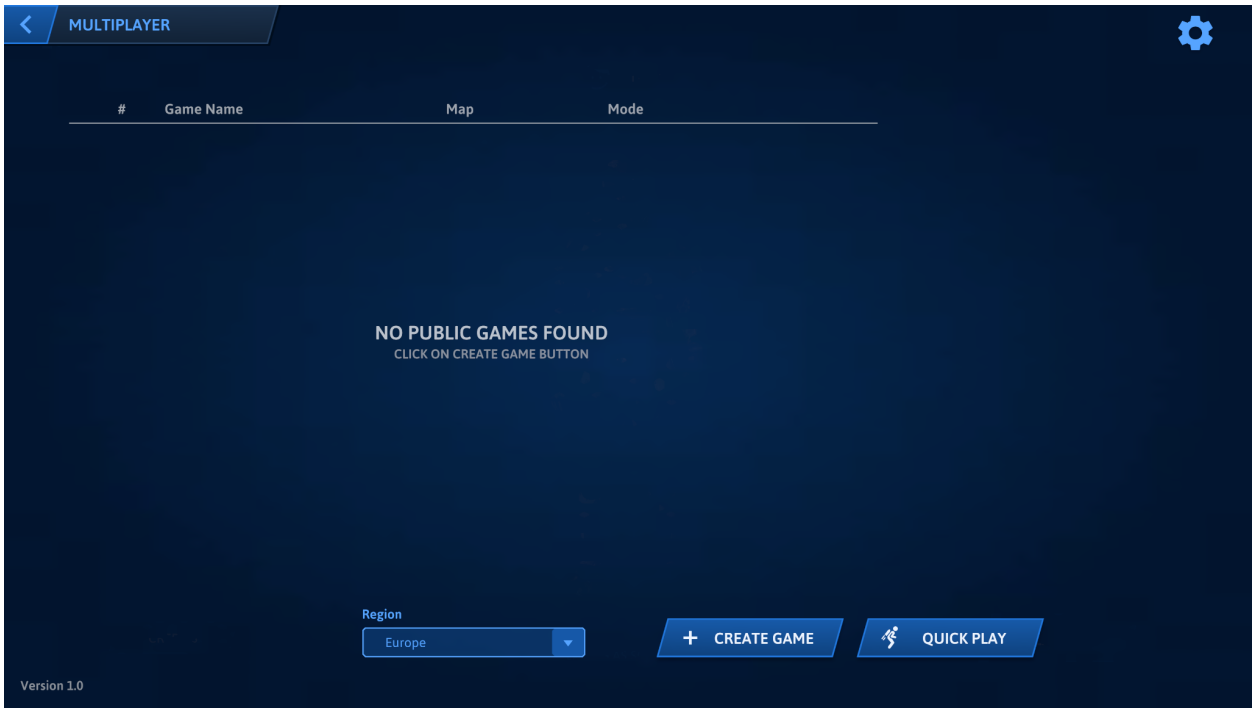
# Start the game client

You can test your game servers by launching the game client from the Unity Editor, using the `Loader.unity` scene file located in `Assets/TPSBR/Scenes`, or as a standalone build.



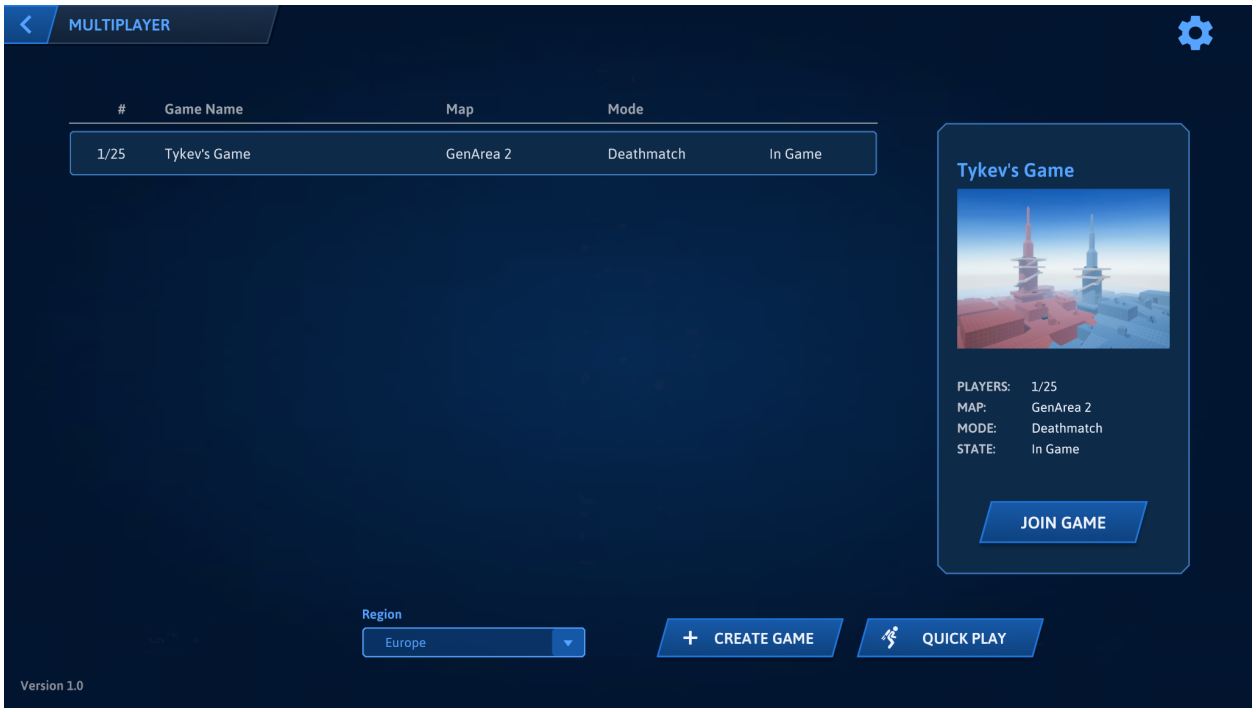

After launching, the game client shows a game session list based on the available game sessions on Multiplay servers. If this is the first time you're running the application and haven't already started any sessions on Multiplay, you won't see any game sessions available yet.

**IMPORTANT:** you will need to select Europe in your game client, due to the initial launch parameters in the Build Configuration.




**Tip**: You can go back to the Multiplay and Matchmaker dashboards to view game performance metrics.

Select **Quickplay** to enter the matchmaker and start servers on Multiplay. If there are already servers running, the game client attempts to backfill into the running game. See the Backfill documentation to learn more.




Once a connection is established and the game launches, you can play.


If you're running a standalone build, you can launch a second client to try out joining the same game.

The clients can interact with each other, including across devices. You can repeat this for up to

200 players to test feasibility, player visceral experience, and server performance scalability.


# Iterate the server build

After configuring and running the Fusion BR200, you can make changes in the Unity Editor and generate a new standalone build to test your changes.

However, before testing your changes live, you must create a new release for your build on Multiplay.

1. Log in to the Unity Dashboard.
2. Go to **Multiplayer** > **Multiplay** > **Builds**.
3. Select the build you created in the Create a build step.
4. Select **Files**, then upload the new files from the generated build.
5. Wait for the new version to sync.

Once synced, you can test the updated build live on Multiplay's servers.

# Cookbook

To add Multiplay hosting, you'll need to extend your game host lifecycle in several places.

# Multiplay Manager

The `MultiplayManager` class is an entry point for creating game sessions in response to allocations. Game servers must stay warm or sit idle in a starting state to scale rapidly. This way, the game server is ready to accept players when an allocation comes. The `StandaloneManager` starts the `MultiplayManager` if the `Loader` detects the game is running in batch mode.

**Note**: Batch mode refers to the `-batchmode` parameter passed to the build executable through the build configuration.

`MultiplayManager.cs` shows how to:

- Enable SQP. SQP is the query protocol Multiplay uses to poll for server status, player count, and other game details
- Respond to allocation events.
- Fetch matchmaking results, such as pending player connections.
- Start a Fusion session via matchmaking.

# Matchmaker

Not to be confused with Fusion's matchmaking, the Unity Matchmaker is a powerful service-side player grouping and server orchestration system.

`Matchmaker.cs` shows how to:

- Work with the basic lifecycle of a Matchmaking ticket.
- Process ticket assignments.
- Connect to the Multiplay service through Photon Cloud.

# Backfill

Backfill enables you to place new players into existing matches based on matchmaking criteria and game session vacancies. When enabled on a matchmaker pool, the Matchmaker service creates backfill tickets automatically.

The game server has two primary responsibilities:

1. Approve new players matched with the ongoing backfill ticket.
2. Update the backfill ticket if players join from outside the matchmaker or drop out of the game.

`Backfill.cs` shows how to:

- Perform backfilling based on the roster of the game
- Update backfilling when a player joins from outside matchmaking

- Enable and Disable backfilling through game-mode logic