

# Fusion BR200 - Multiplay Integration Documentation

## [Get started](#)

[Get started with UGS](#)

[Install the Unity Editor](#)

[Get started with Photon Fusion](#)

[Link the Photon Fusion project](#)

[Link your UGS project](#)

[Build the standalone server](#)

## [Add the Multiplay Manager](#)

[Enable Multiplay](#)

[Integrate game server](#)

[Create a build](#)

[Create a build configuration](#)

[Create a fleet](#)

[Create a test allocation](#)

## [Add the Unity Matchmaker](#)

[Enable Matchmaker](#)

[Integrate Matchmaker](#)

[Create a queue](#)

[Create a default pool](#)

## [Start the game client](#)

## [Iterate the server build](#)

## [Cookbook](#)

[Multiplay Manager](#)



[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:

### Install Unity Editor ✕

[Official releases](#)   [Pre-releases](#)   [Archive](#)

Can't find the version you're looking for? Visit our [download archive](#) for access to [Long-Term Support](#) and [patch releases](#), or join our [beta program](#) releases.

[Beta program webpage](#)

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

## Unity download archive

From this page you can download the previous versions of Unity for both Unity Personal and Pro (if you have a Pro license, enter in your key when prompted after installation). Please note that we don't support downgrading a project to an older editor version. However, you can import projects into a new editor version. We advise you to back up your project before converting and check the console log for any errors or warnings after importing.

### Long Term Support releases

The LTS stream is for users who wish to continue to develop and ship their games/content and stay on a stable version for an extended period.

[Download LTS releases](#)

Unity 2022.x [Unity 2021.x](#) Unity 2020.x Unity 2019.x Unity 2018.x Unity 2017.x Unity 5.x Unity 4.x Unity 3.x

Unity 2021.3.7f1  
28 Jul, 2022



Downloads (Win) ▾

Downloads (Mac) ▾

Downloads (Linux) ▾

Release notes

Unity 2021.3.6f1  
8 Jul, 2022



Downloads (Win) ▾

Downloads (Mac) ▾

Downloads (Linux) ▾

Release notes

Unity 2021.3.5f1  
22 Jun, 2022



Downloads (Win) ▾

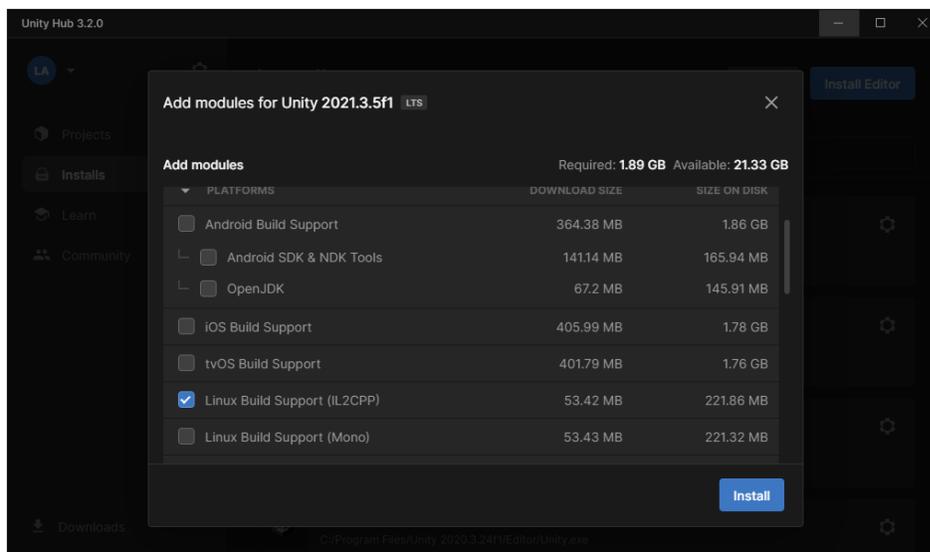
Downloads (Mac) ▾

Downloads (Linux) ▾

Release notes

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.

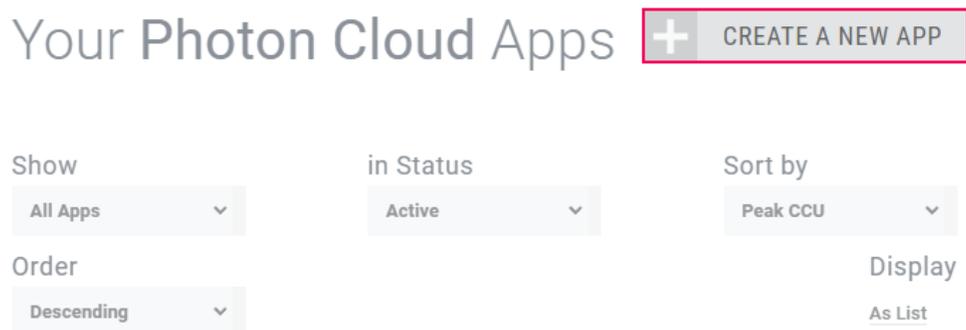


## Get started with Photon Fusion

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**.

### Create a New Application

The application defaults to the **Free Plan**.  
You can change the plan at any time.

Photon Type \*

Fusion

Name \*

Unity sample

Description

Short description, 1024 chars max.

Url

http://enter.your-url.here/

CREATE

or [go back](#) to the application list.

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**.

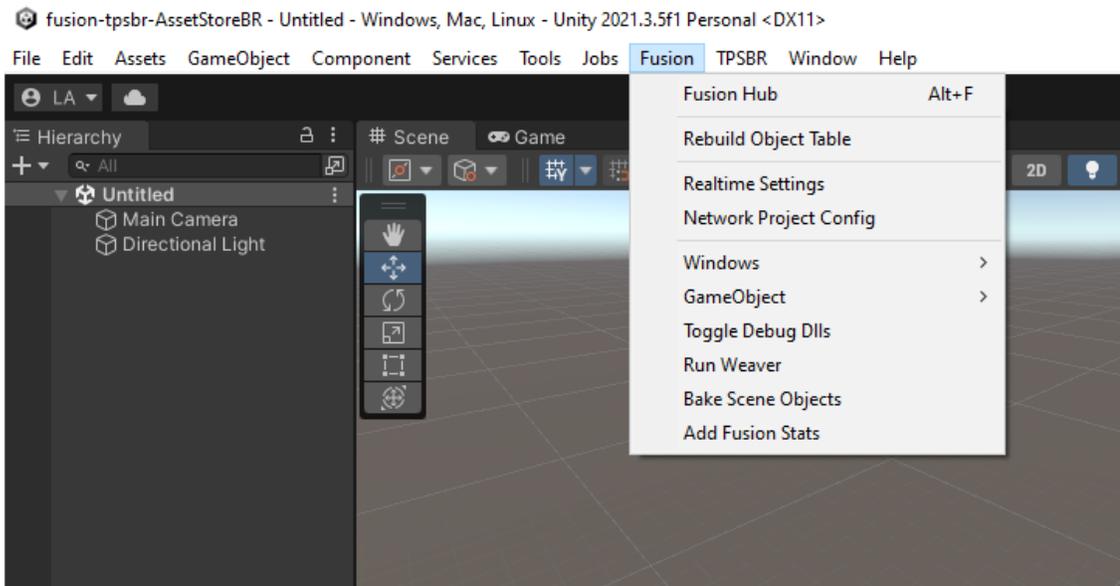
## Manage Unity sample

App ID:

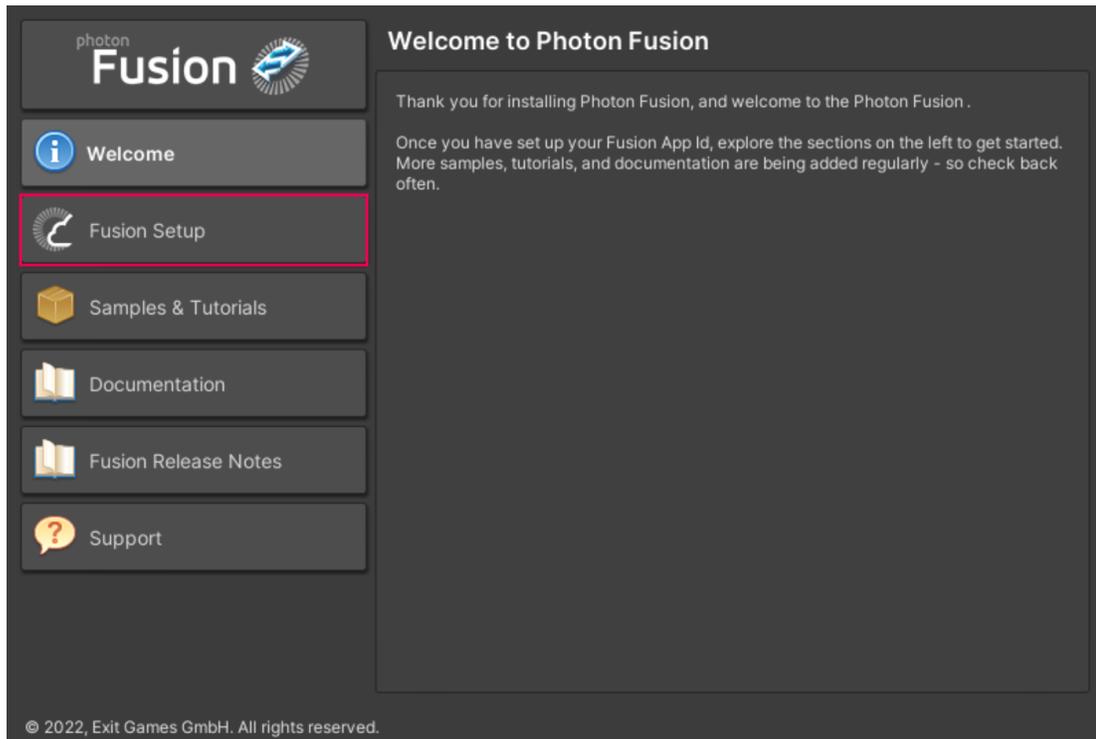
## Link the Photon Fusion project

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

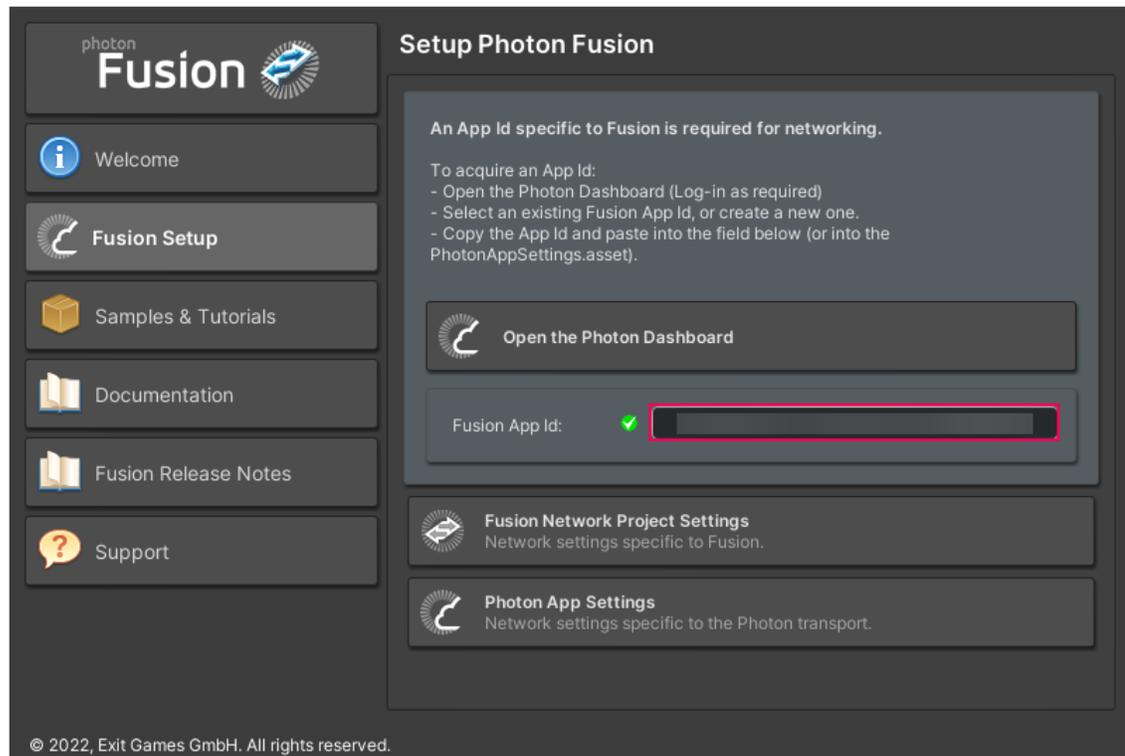
1. Launch the Fusion BR200 project in the Unity Editor.
2. Select **Fusion > Fusion Hub**.



3. Select **Fusion Setup**.



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

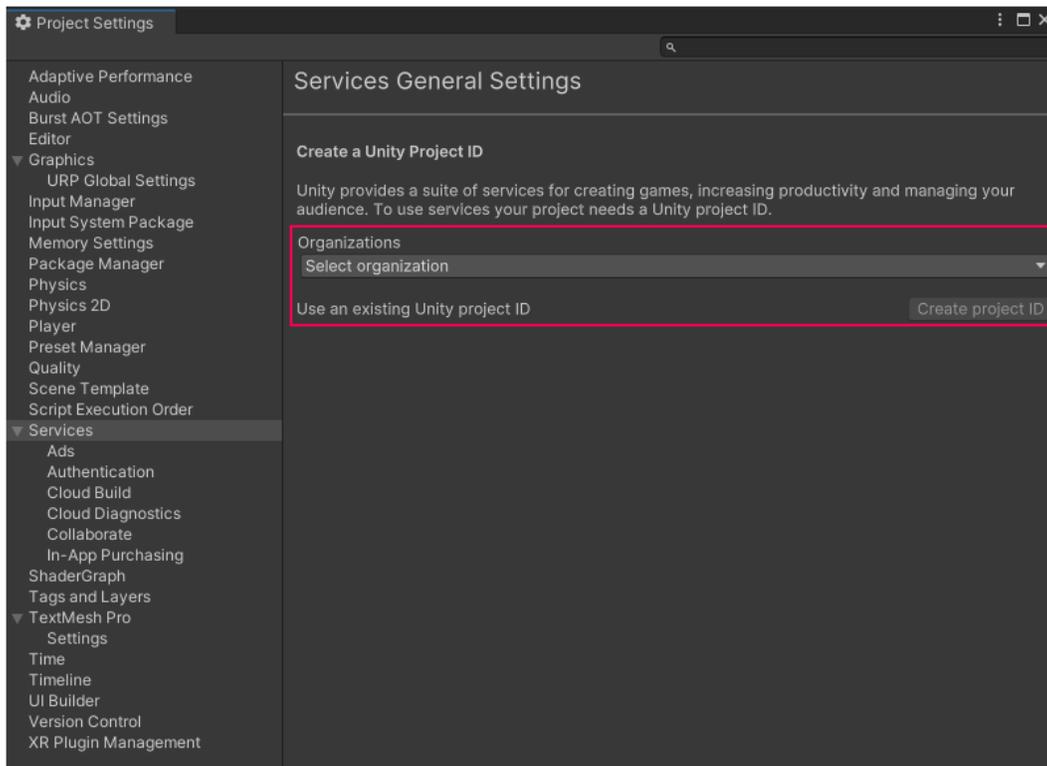


## Link your UGS project

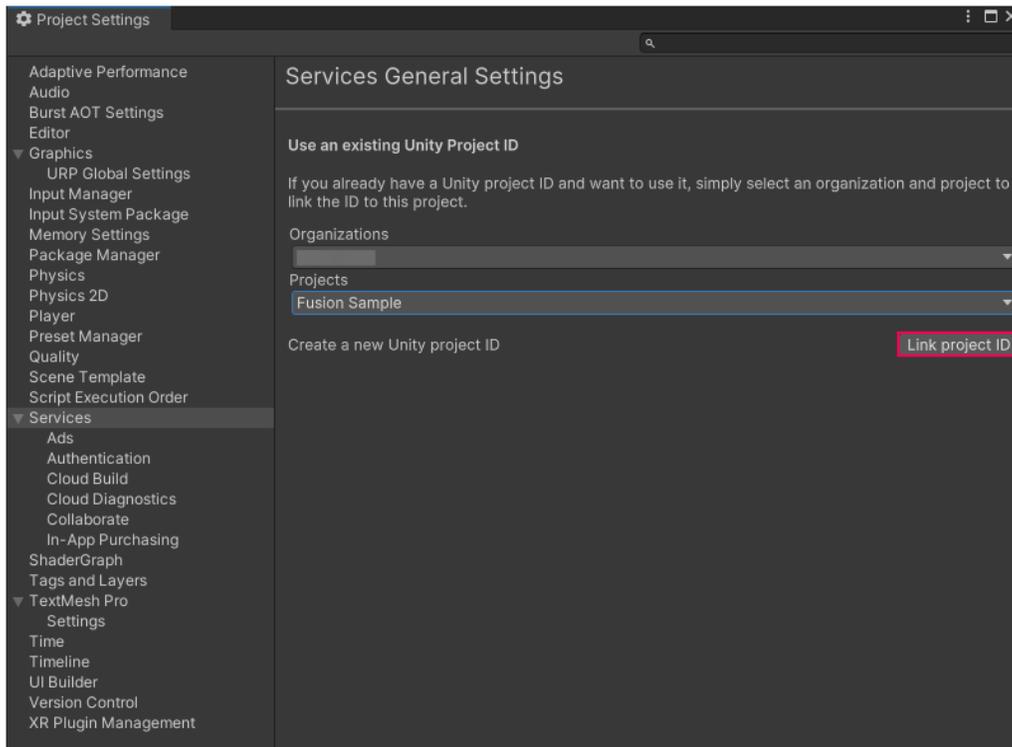
After the installation, link your UGS account and project with the Unity Editor.

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**.

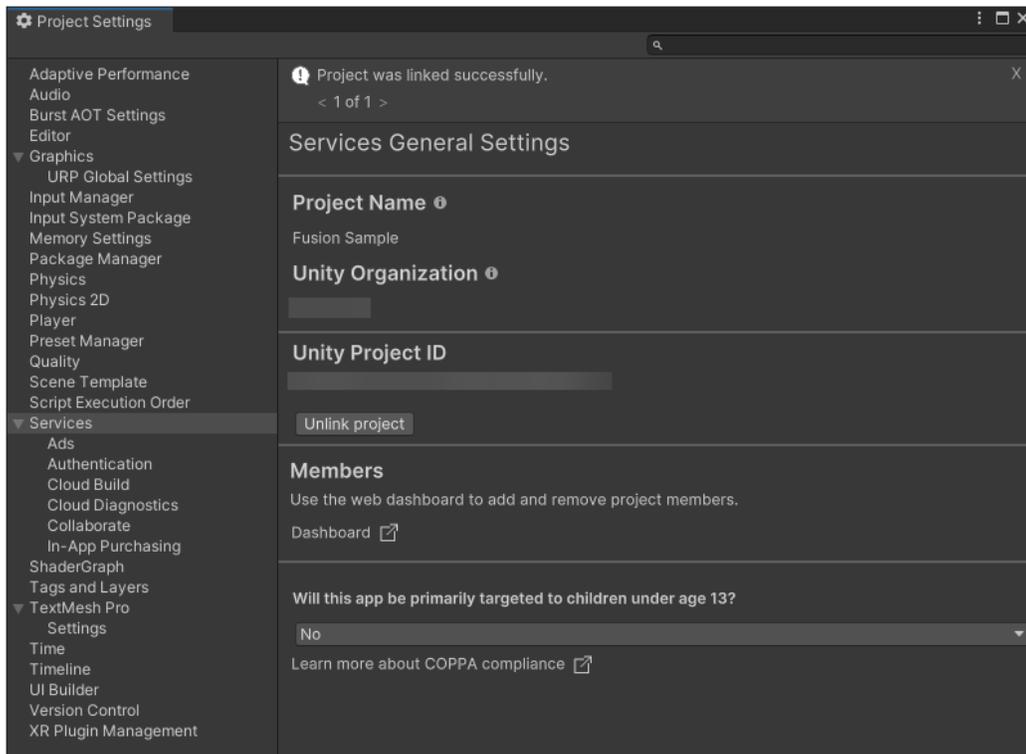
**Note:** You can only create a project ID if you have adequate permission within the organization.



### 3. Select **Link project ID**.



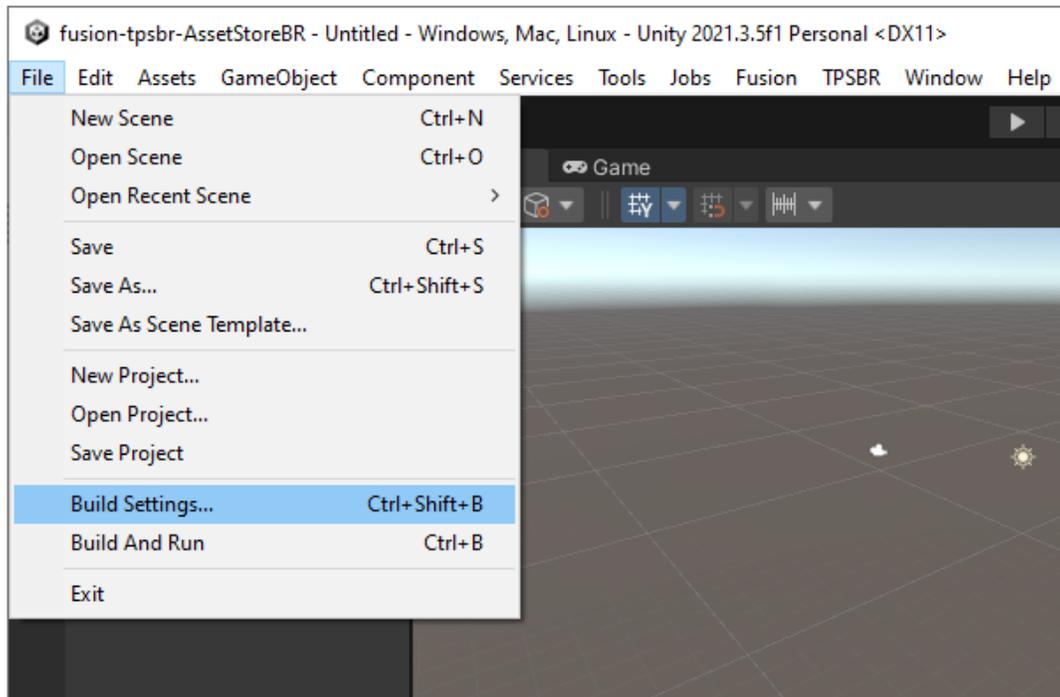
### 4. You should see a message stating that the project was linked successfully.



## 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.

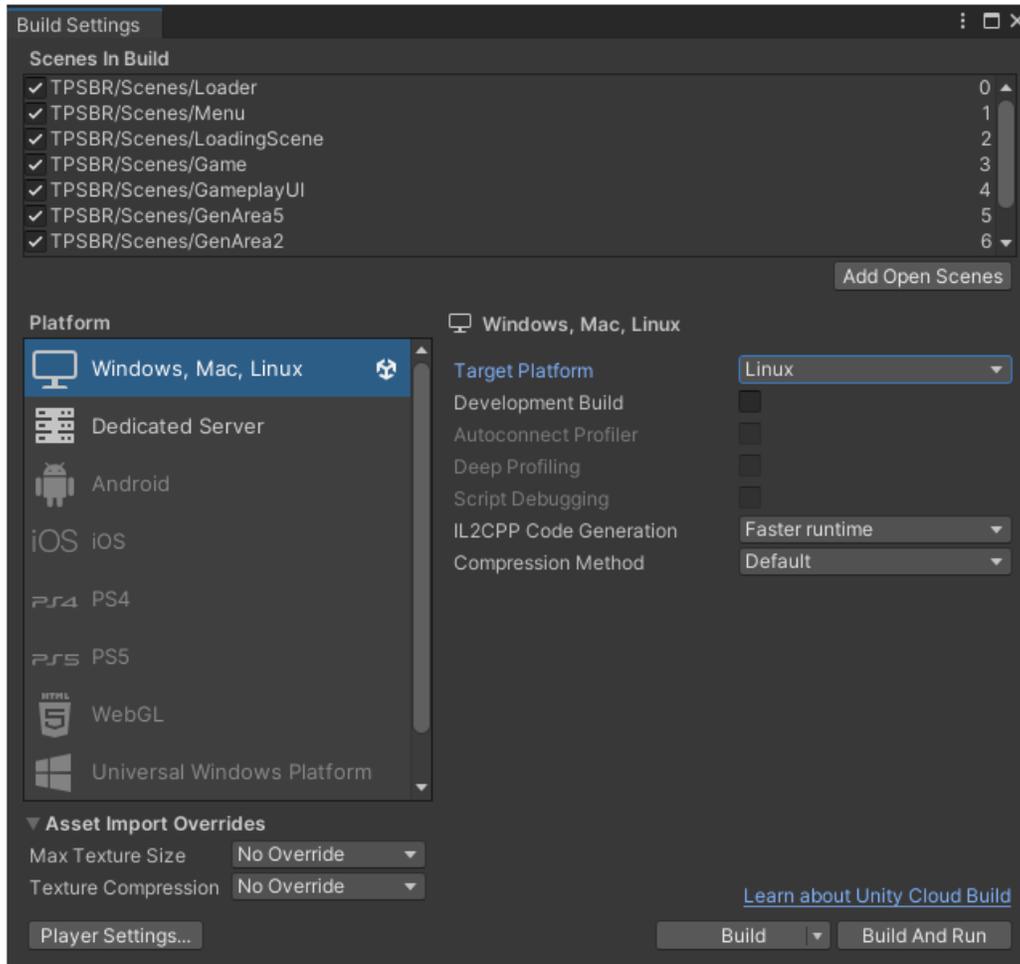
1. From the Unity Editor, go to **File > Build Settings....**



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 **Multplayer** > **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**.

### Setup guide

production ▾ Reset guide

Get started with Multiplay ▾

**1** Integrate your game server ^

Use our Unity or Unreal packages and SDKs to integrate your game server with Multiplay.

[Integrate game server](#)

2 Create a build ▾

3 Create a build configuration ▾

4 Create a fleet ▾

5 Create a test allocation ▾

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

### Integrate game server

1 — 2 — 3

Select engine      Link Unity Project      Install Project

To begin the integration, select your engine:

 Unity       Unreal       Custom

Cancel Next

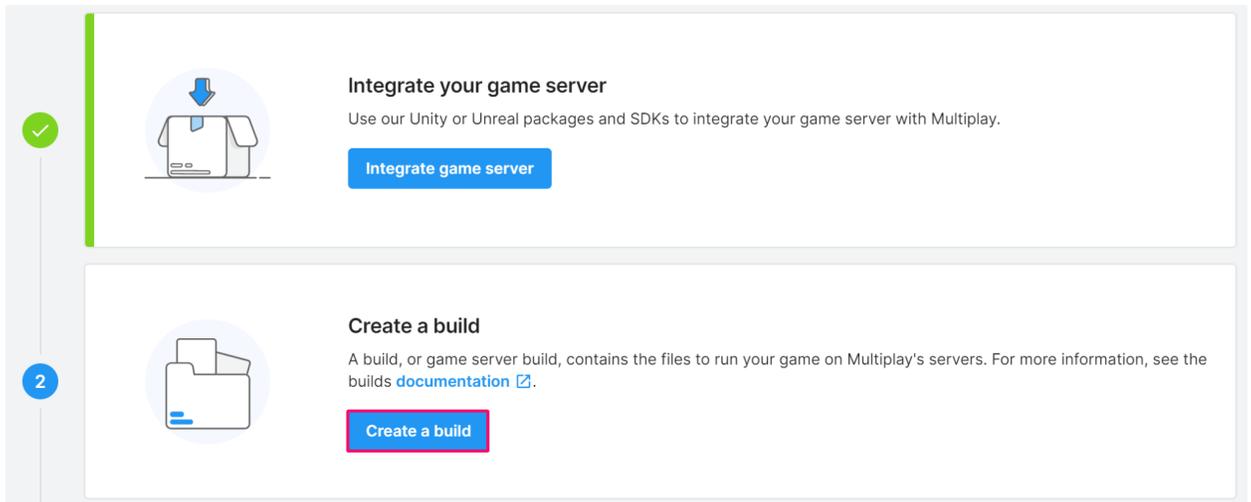
3. Select **Next** if you've already linked your Unity project with the Unity Editor.
4. Select **Finish**.

**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**.

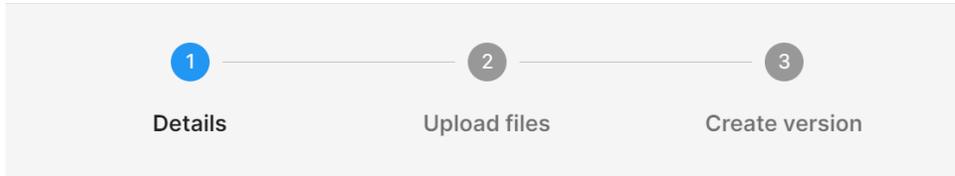


The screenshot shows a two-step process in the Multiplay service interface. A vertical progress bar on the left indicates the current step. The first step, 'Integrate your game server', is marked with a green checkmark and a blue bar. The second step, 'Create a build', is marked with a blue circle containing the number '2' and a red bar. Each step includes an icon, a title, a description, and a button.

Step	Icon	Title	Description	Action
1		<b>Integrate your game server</b>	Use our Unity or Unreal packages and SDKs to integrate your game server with Multiplay.	<a href="#">Integrate game server</a>
2		<b>Create a build</b>	A build, or game server build, contains the files to run your game on Multiplay's servers. For more information, see the builds <a href="#">documentation</a> .	<a href="#">Create a build</a>

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

## Create build



A build, or game server build, contains the files to run your game on Multiplay's servers. For more information, see the [builds requirements documentation](#).

Build name \*

Fusion Sample

Operating system \*

 <b>Linux</b> Recommended	 <b>Windows</b> Support coming soon
---	---

Upload method \*

<input checked="" type="radio"/> <b>Direct file upload</b> Upload files via the dashboard	<input type="radio"/> <b>Use container image</b> Add files using a container
--	---

Cancel

**Next**

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**.

**Create build**

Progress: 1. Details (✓) — 2. Upload files (●) — 3. Create version (●)

**Upload files**

**Upload the files necessary to run your server. Do not upload a zipped archive.**

[Cancel](#) [Upload 50 Files](#)

Drop file(s) here or [browse](#)

Search files

Name ↑	Status	
fusion_build_Data/app.info	● Ready to upload	🗑
fusion_build_Data/boot.config	● Ready to upload	🗑
fusion_build_Data/globalgamemanag...	● Ready to upload	🗑
fusion_build_Data/globalgamemanag...	● Ready to upload	🗑
fusion_build_Data/globalgamemanag...	● Ready to upload	🗑
fusion_build_Data/il2cpp_data/Metad...	● Ready to upload	🗑
fusion_build_Data/level0	● Ready to upload	🗑
fusion_build_Data/level1	● Ready to upload	🗑
fusion_build_Data/il2cpp_data/Resour...	● Ready to upload	🗑

[Cancel](#) [Next](#)

## 6. Select **Next**.

### Create build

✓ — 2 — 3  
Details Upload files Create version

**i** Upload the files necessary to run your server. Do not upload a zipped archive.

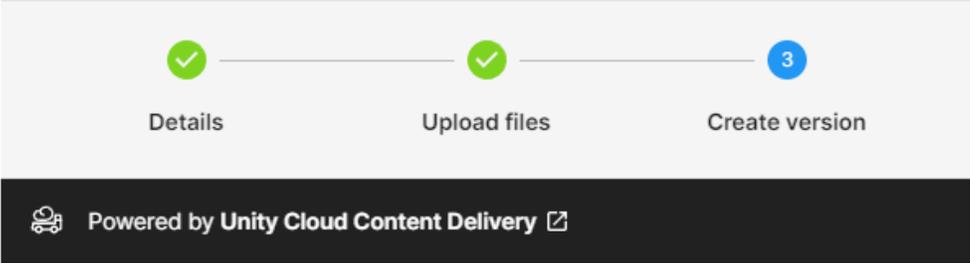
✓ **Upload complete**  
50 files uploaded successfully | 0 files failed to upload

Name ↑	Status	
fusion_build_Data/app.info	● Added	🗑
fusion_build_Data/boot.config	● Added	🗑
fusion_build_Data/globalgamemanag...	● Added	🗑
fusion_build_Data/globalgamemanag...	● Added	🗑
fusion_build_Data/globalgamemanag...	● Added	🗑
fusion_build_Data/il2cpp_data/Metad...	● Added	🗑
fusion_build_Data/level0	● Added	🗑
fusion_build_Data/level1	● Added	🗑
fusion_build_Data/il2cpp_data/Resour...	● Added	🗑
fusion_build_Data/il2cpp_data/Resour...	● Added	🗑

Cancel Next

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

### Create build



Below is a summary of the first version that will be created for this build.

Build name	Version
Fusion Sample	Version 1

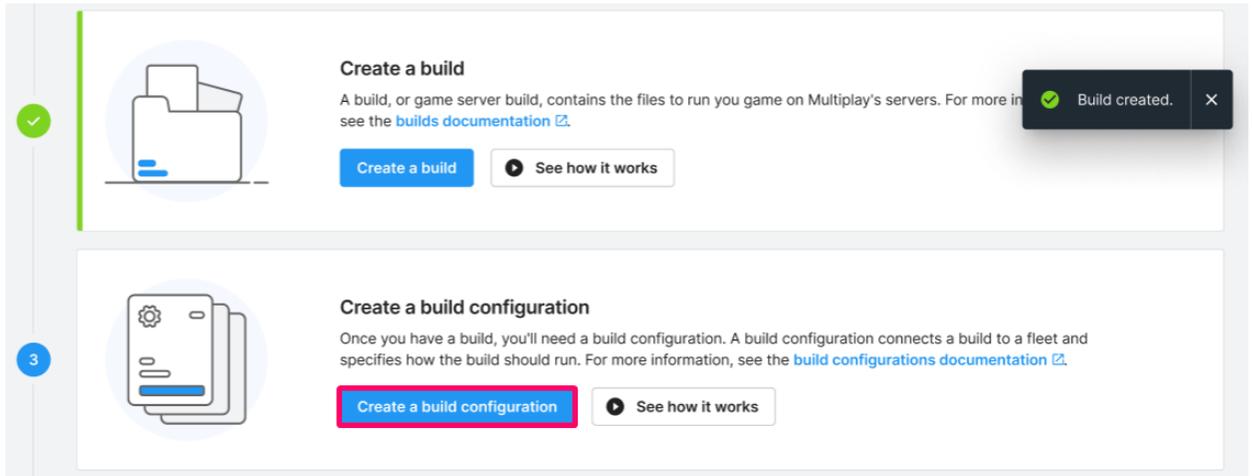
Cancel Finish

## 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**.



The screenshot shows a two-step process in the Unity console. The first step, 'Create a build', is marked with a green checkmark and a folder icon. The second step, 'Create a build configuration', is marked with a blue circle containing the number '3' and a document icon. The 'Create a build configuration' button is highlighted with a pink border. A notification 'Build created.' is visible in the top right corner.

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 **SQL**.
- 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**.

Query type ⓘ

<input checked="" type="radio"/> <b>SQP</b> Supported by the Multiplay Game Server SDK	<input type="radio"/> <b>A2S</b> Supported by the Steam SDK from Valve	<input type="radio"/> <b>None</b> If your server has no support for querying metrics
---	---	---

Launch parameters ⓘ

```
-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
```

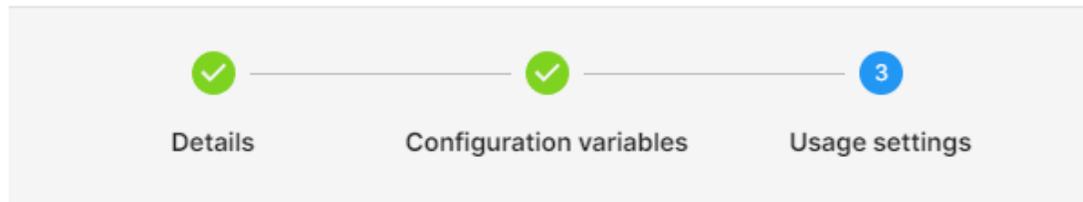
Cancel

**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**.

### Create build configuration



Usage settings control the compute resources available to each server using this build configuration. For more information, see the [usage documentation](#) 

Default       Custom

CPU speed ⓘ *	2000	MHz	Memory ⓘ *	1024	MB
---------------	------	-----	------------	------	----

Cancel

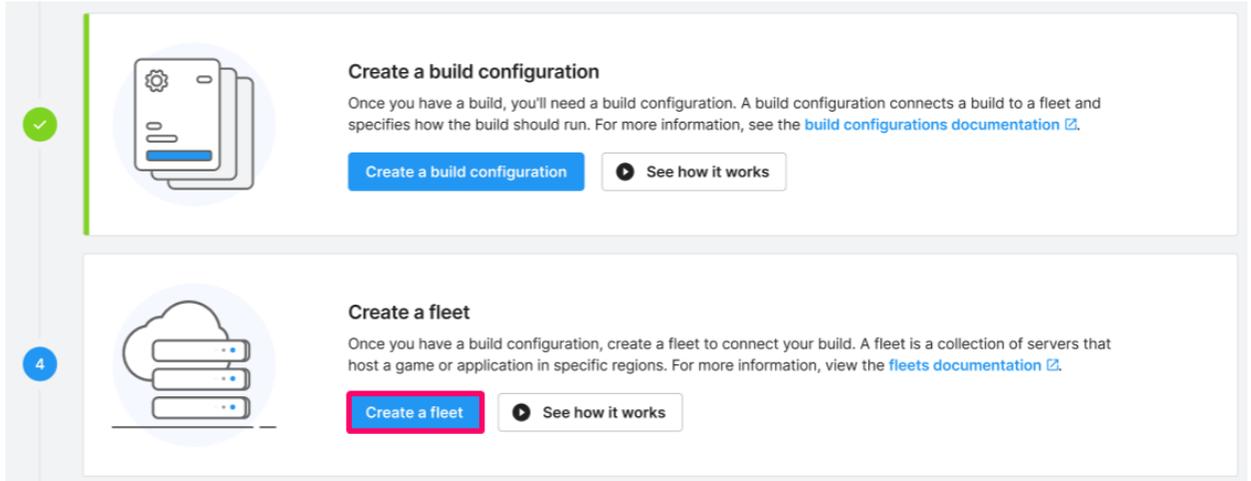
Back      **Finish**

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**.



The screenshot shows a vertical list of two steps. The first step, 'Create a build configuration', is marked with a green checkmark. The second step, 'Create a fleet', is marked with a blue circle containing the number 4. Each step includes an icon, a title, a descriptive paragraph, and two buttons: 'Create a build configuration' and 'See how it works'.

**Create a build configuration**  
Once you have a build, you'll need a build configuration. A build configuration connects a build to a fleet and specifies how the build should run. For more information, see the [build configurations documentation](#).

**Create a fleet**  
Once you have a build configuration, create a fleet to connect your build. A fleet is a collection of servers that host a game or application in specific regions. For more information, view the [fleets documentation](#).

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**.

### Create fleet

1 ————— 2  
**Details**                      **Scaling settings**

**i** You will need a build configuration in order to create a fleet.

Fleet name \*  
Fusion Sample Fleet

Operating system \*

 **Linux**

 **Windows**

Build configuration(s) \*  
1 build configuration selected

If you don't see a build configuration listed, it might be in use by another fleet.

Cancel**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 fleet

Progress indicator: Details (1) — Scaling settings (2)

**i** For Closed Beta, you are limited to 30 max available servers. Contact Support if you need more resources. [Contact Support](#) 

Region\*  
North America 

Select a region in which your game or application will be hosted. You can add more regions later.

Min available servers*	Max servers*
<input type="text" value="1"/>	<input type="text" value="2"/>

Cancel Back **Finish**

## Create a test allocation

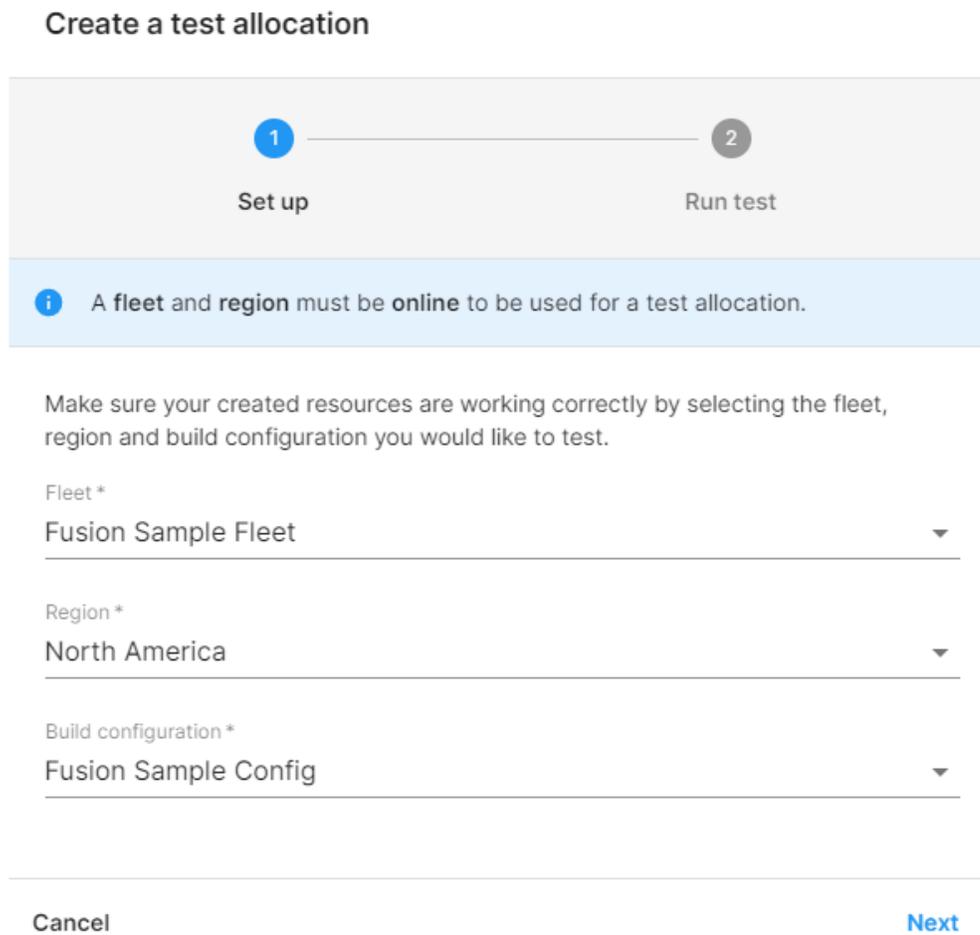
Create a test allocation to make sure everything's working correctly. See the [Allocation documentation](#) for help.

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

fleets documentation.' and two buttons: 'Create a fleet' and 'See how it works'. The bottom section is 'Create a test allocation', which is marked as the current step with a blue circle containing the number '5'. It includes a graph icon, a description: 'Once you have a fleet, build, and build configuration, create a test allocation to make sure everything you've created is working correctly.', and two buttons: 'Create test allocation' (highlighted with a red border) and 'See how it works'." data-bbox="176 132 945 358"/&gt;

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

### Create a test allocation



1 Set up ————— 2 Run test

**i** A fleet and region must be online to be used for a test allocation.

Make sure your created resources are working correctly by selecting the fleet, region and build configuration you would like to test.

Fleet\*  
Fusion Sample Fleet

Region\*  
North America

Build configuration\*  
Fusion Sample Config

Cancel Next

3. Select **Next**.

4. Select **Run test**.

Create a test allocation



Run a test allocation using the Multiplay interface

**Run test**

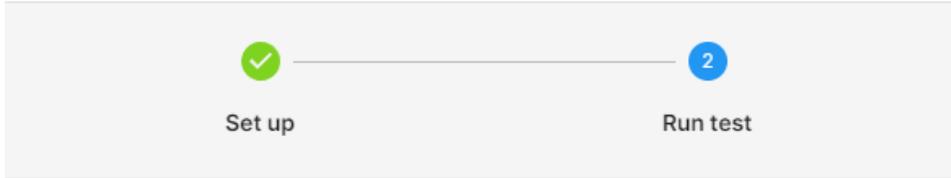
---

Cancel

Back **Finish**

5. Wait for the test to complete.

### Create a test allocation



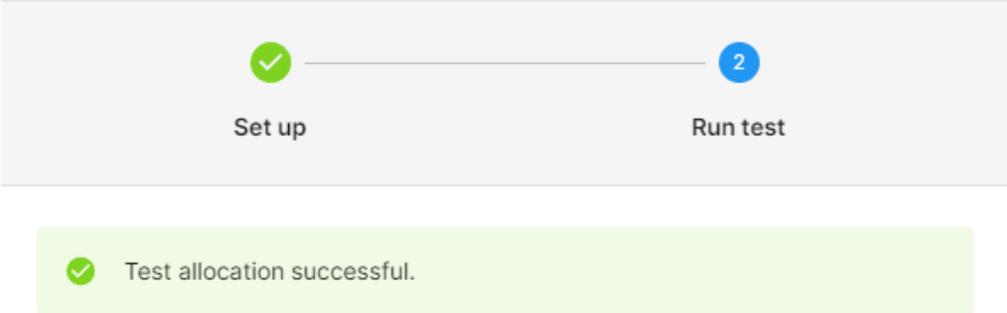
- Sending allocation
- Waiting for server
- Server allocated successfully

Cancel

Back Finish

6. Select **Finish**.

**Create a test allocation**



Test allocation ID	f34a2239-85fc-449d-8bdd-cc1d15a0def0	
Server IP:Port	34.86.0.90:9000	
Time remaining	59m 42s	

[Cancel](#) [Back](#) [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**.

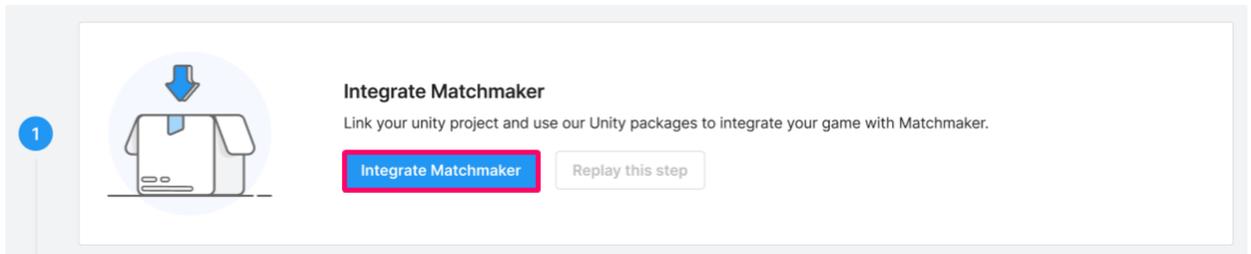
1. Sign in to the Unity Dashboard with your Unity account.
2. From the Unity Dashboard, go to **Multiplayer > Matchmaker**.
3. Select **Set up Matchmaker**.
4. Use the Setup Guide, starting with the **Integrate Matchmaker** step.

## 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**.

### Integrate Matchmaker



To begin Matchmaker integration please select your game engine and preferred integration method. For more information, see the [Integration documentation](#) .

Game engine:

 <b>Unity</b>	 <b>Unreal</b>	 <b>Custom</b>
--	---	---

Integration method:

 <b>SDK</b> Recommended The quickest integration method for all skill levels.	 <b>API</b> Advanced For complex use-cases or if you use a game service we advise calling the API's directly.
--	--

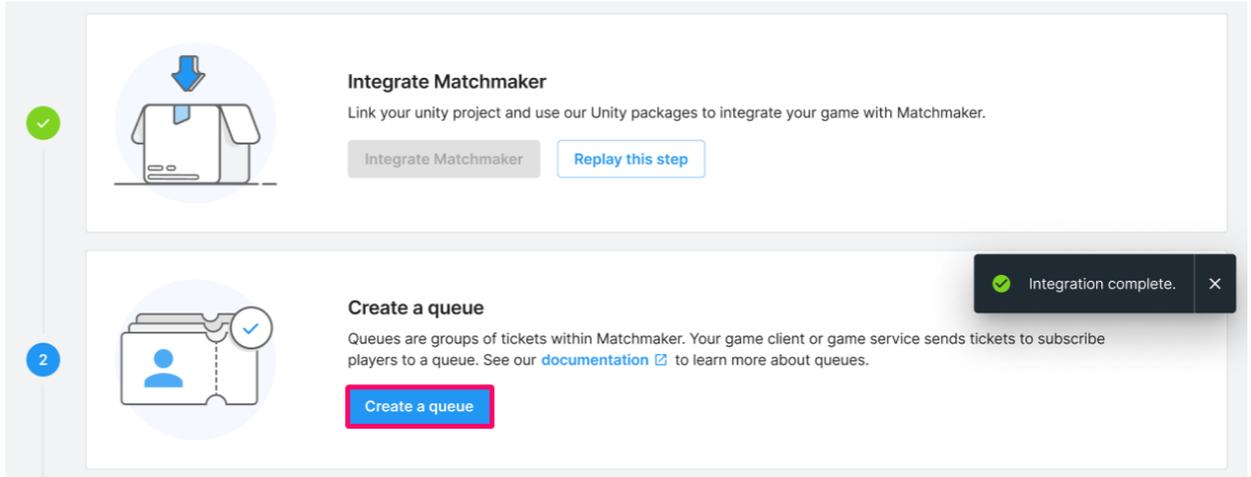
Cancel Next

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**.



The screenshot shows a two-step process. Step 1, 'Integrate Matchmaker', is marked as complete with a green checkmark. Step 2, 'Create a queue', is the current step, marked with a blue '2'. The 'Create a queue' button is highlighted with a red border. A notification in the top right corner says 'Integration complete.' with a green checkmark and a close button.

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

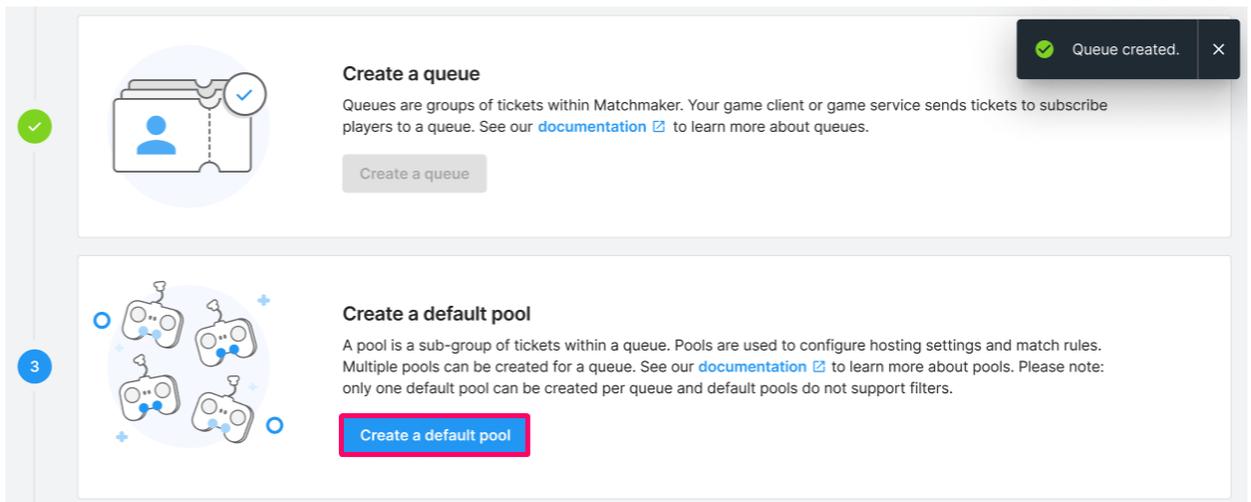
**Note:** The system is case sensitive, and using a queue name other than "**battleRoyale**" results in an exception.

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.

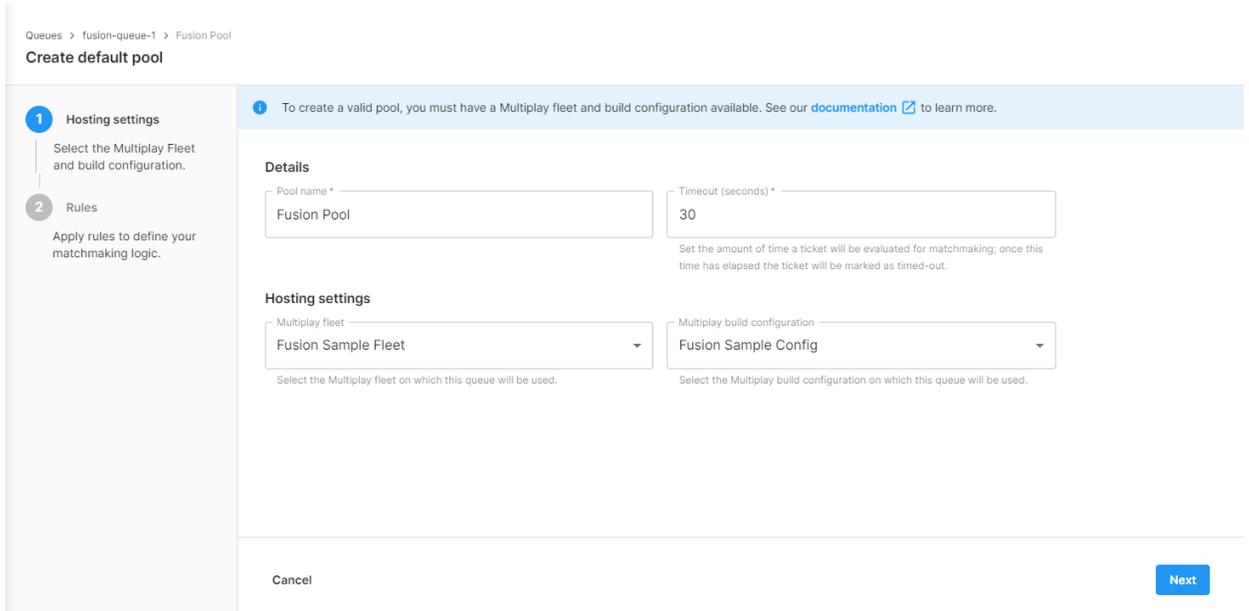
1. Select **Create a default pool**.



The screenshot shows the 'Create a default pool' step. The 'Create a queue' step above it is marked as complete with a green checkmark. The 'Create a default pool' button is highlighted with a red border. A notification in the top right corner says 'Queue created.' with a green checkmark and a close button.

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**.



Queues > fusion-queue-1 > Fusion Pool

### Create default pool

**1** Hosting settings  
Select the Multiplay Fleet and build configuration.

**2** Rules  
Apply rules to define your matchmaking logic.

To create a valid pool, you must have a Multiplay fleet and build configuration available. See our [documentation](#) to learn more.

**Details**

Pool name \*  
Fusion Pool

Timeout (seconds) \*  
30

Set the amount of time a ticket will be evaluated for matchmaking; once this time has elapsed the ticket will be marked as timed-out.

**Hosting settings**

Multiplay fleet  
Fusion Sample Fleet

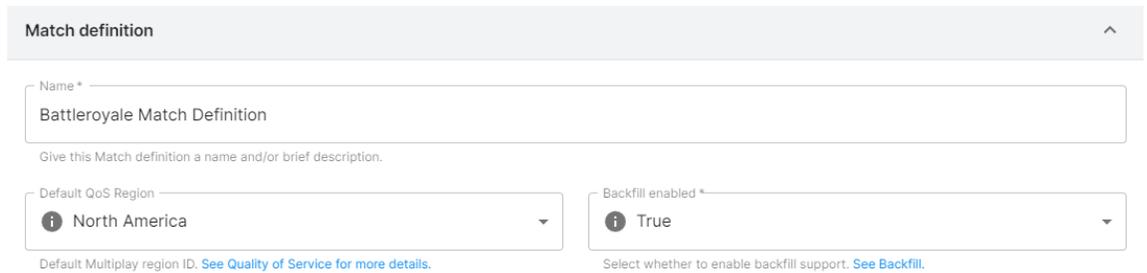
Multiplay build configuration  
Fusion Sample Config

Select the Multiplay fleet on which this queue will be used.

Select the Multiplay build configuration on which this queue will be used.

Cancel 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**.



Match definition

Name \*  
Battleroyale Match Definition

Give this Match definition a name and/or brief description.

Default QoS Region  
North America

Backfill enabled \*  
True

Default Multiplay region ID. [See Quality of Service for more details.](#)

Select whether to enable backfill support. [See Backfill.](#)

- 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

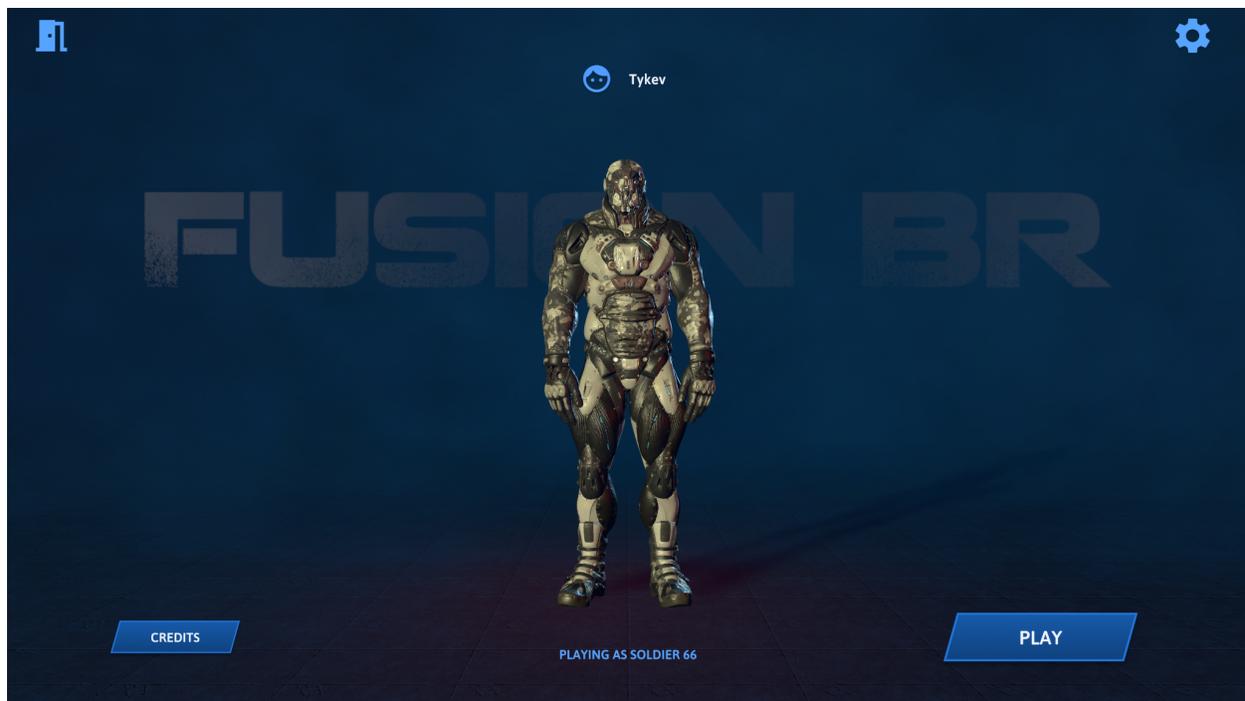
help.

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.

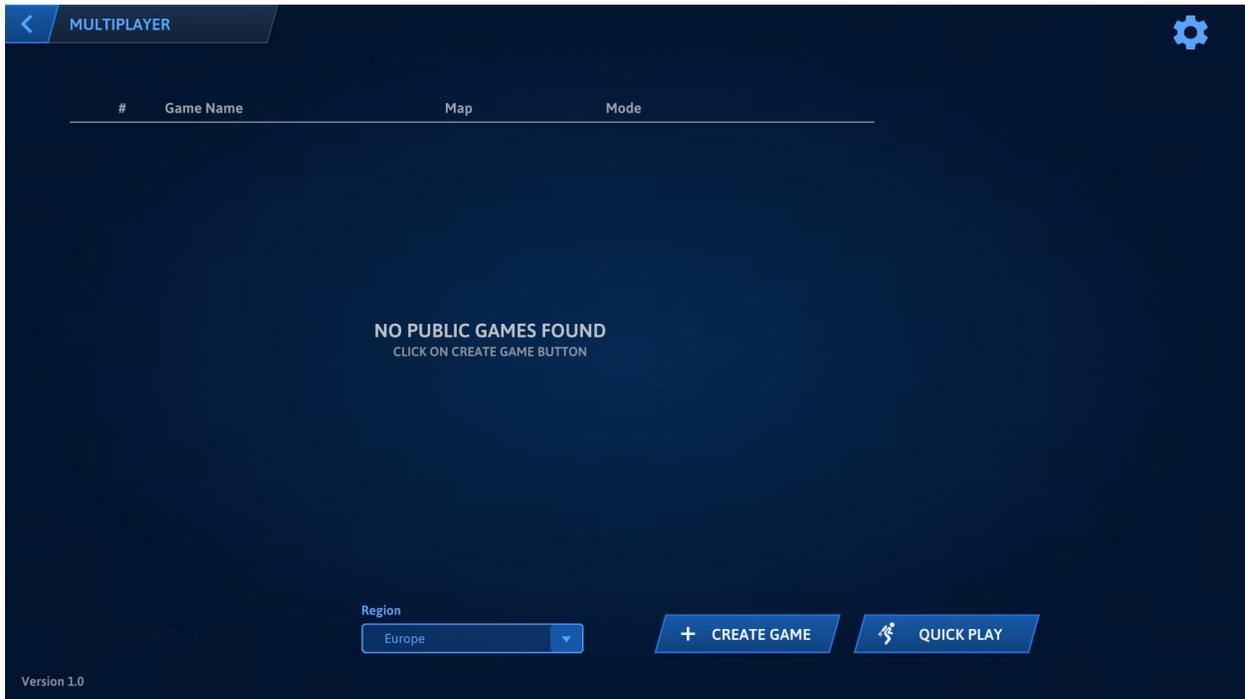
## Start the game client

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.



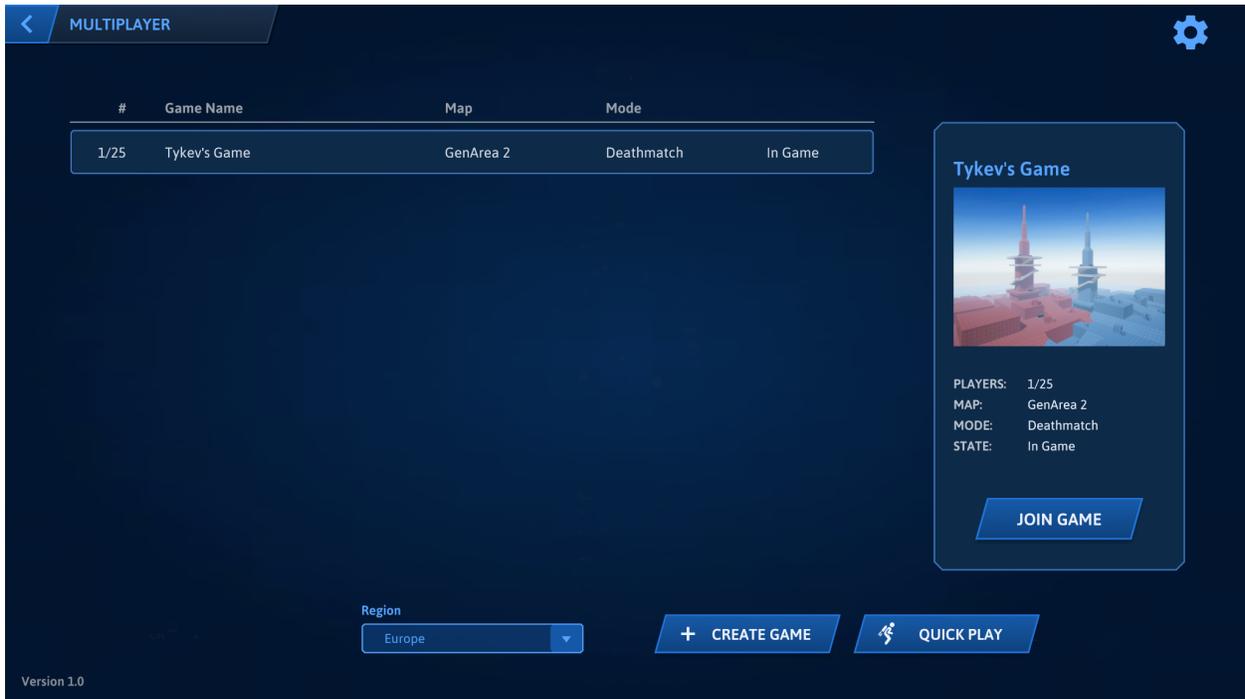
After launching, the game client shows a game session list based on the available game sessions on Multiplay servers. If this is the first time you're running the application and haven't already started any sessions on Multiplay, you won't see any game sessions available yet.

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

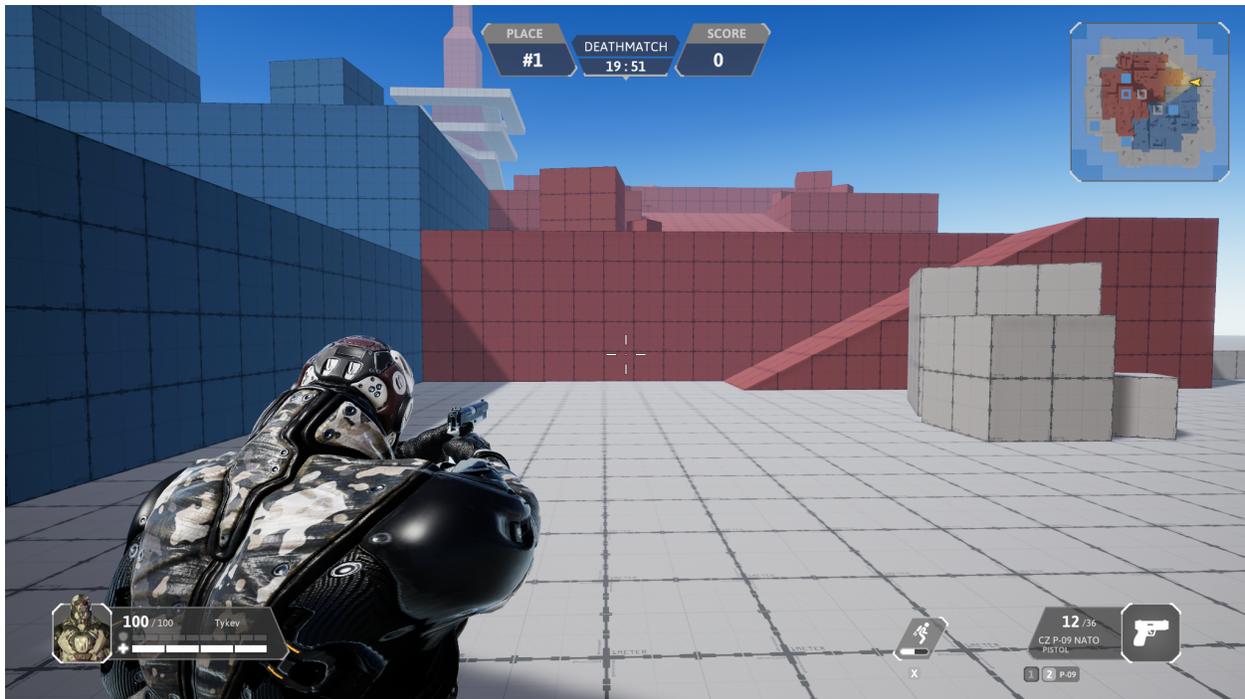


**Tip:** You can go back to the Multiplay and Matchmaker dashboards to view game performance metrics.

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