

Fusion BR200 - Unity Game Server Hosting Integration Documentation



The **BR200** project demonstrates how to create a fully functional multiplayer game using **Photon Fusion** and **Unity Gaming Services** (UGS), including [Unity Game Server Hosting](#) 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 2 Application Id.
- You must use Unity Editor 2022.3.20f1.

Table of contents

Get started.....	3
Get started with Unity Gaming Services.....	3
Install the Unity Editor.....	3
Get started with Photon Fusion.....	5
Link the Photon Fusion project.....	7
Link your Unity Gaming Services project.....	8
Build the standalone server.....	10
Configure the Unity Game Server Hosting.....	11
Enable Game Server Hosting.....	11
Integrate your game server.....	11
Create a build.....	13
Create a build configuration.....	16
Create a fleet.....	18
Create a test allocation.....	20
Configure the Unity Matchmaker.....	23
Enable Matchmaker.....	23
Integrate Matchmaker.....	23
Create a queue.....	24
Create a default pool.....	25
Start the game client.....	29
Iterate the server build.....	31
Implementation details.....	32
MultiplayManager.....	32
Matchmaker.....	32
Backfill.....	32

Get started

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

1. [Get started with Unity Gaming Services](#)
2. [Install the Unity Editor](#)
3. [Get started with Photon Fusion](#)
4. [Link your Photon Fusion project](#)

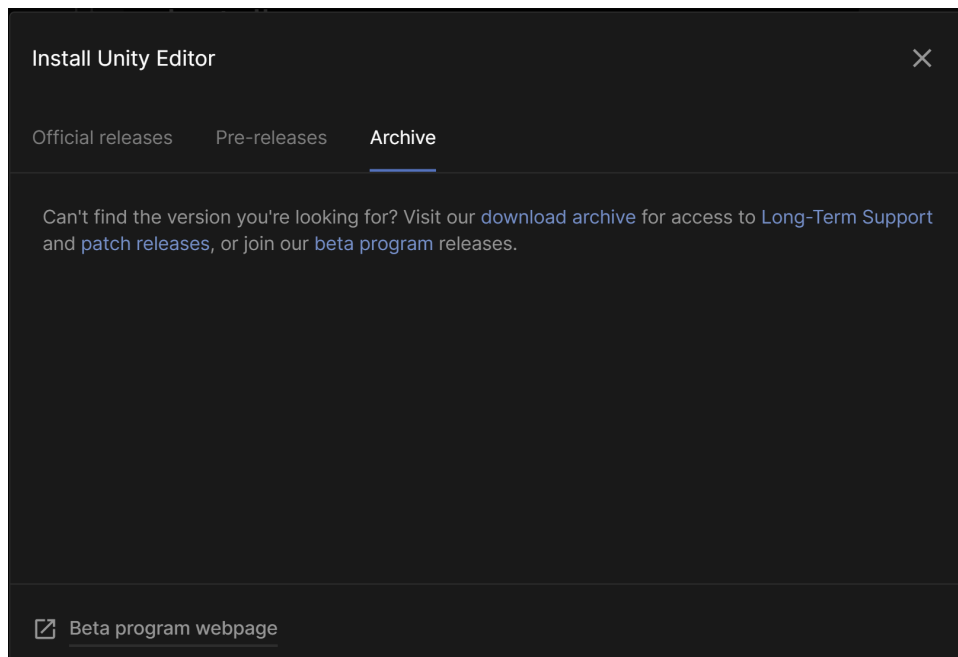
Note: Visit [Unity Dashboard Support](#) if you need help with any Unity services. Visit [Photon Contact](#) page for help with Photon Fusion.

Get started with Unity Gaming Services

You need a [Unity account](#) to access Unity Game Server Hosting and the Unity Matchmaker. If you don't already have a Unity Gaming Services (UGS) account, see the [UGS Documentation](#) and learn how to [get started with UGS](#).

Install the Unity Editor

To work with the BR200 project, you must use [Unity Editor 2022.3.20f1](#). 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:

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.

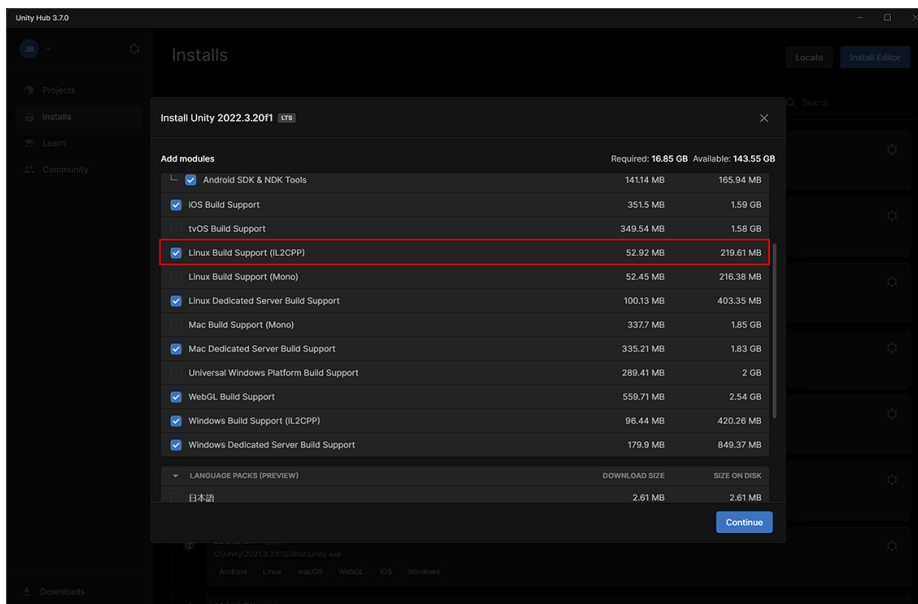
[Download LTS Releases](#)

Unity 2023.X **Unity 2022.X** Unity 2021.X Unity 2020.X Unity 2019.X Unity 2018.X Unity 2017.X Unity 5.X

Unity 2022.3.20 February 14, 2024	Unity Hub	Downloads (Win) ▾	Downloads (Mac) ▾	Downloads (Linux) ▾	Release Notes
Unity 2022.3.19 January 31, 2024	Unity Hub	Downloads (Win) ▾	Downloads (Mac) ▾	Downloads (Linux) ▾	Release Notes
Unity 2022.3.18 January 20, 2024	Unity Hub	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.



Unity Hub 3.2.0

Installs

Install Unity 2022.3.20f1 LTS

Required: 16.85 GB Available: 143.55 GB

Module	Download Size	Size on Disk
<input checked="" type="checkbox"/> Android SDK & NDK Tools	141.14 MB	165.94 MB
<input checked="" type="checkbox"/> iOS Build Support	351.5 MB	1.59 GB
<input type="checkbox"/> tvOS Build Support	349.54 MB	1.58 GB
<input checked="" type="checkbox"/> Linux Build Support (IL2CPP)	52.92 MB	219.61 MB
<input type="checkbox"/> Linux Build Support (Mono)	52.45 MB	216.38 MB
<input checked="" type="checkbox"/> Linux Dedicated Server Build Support	100.13 MB	403.35 MB
<input type="checkbox"/> Mac Build Support (Mono)	337.7 MB	1.85 GB
<input checked="" type="checkbox"/> Mac Dedicated Server Build Support	335.21 MB	1.83 GB
<input type="checkbox"/> Universal Windows Platform Build Support	289.41 MB	2 GB
<input checked="" type="checkbox"/> WebGL Build Support	559.71 MB	2.54 GB
<input checked="" type="checkbox"/> Windows Build Support (IL2CPP)	96.44 MB	420.26 MB
<input checked="" type="checkbox"/> Windows Dedicated Server Build Support	179.9 MB	849.37 MB
LANGUAGE PACKS (PREVIEW)		
<input type="checkbox"/> 日本語	2.61 MB	2.61 MB

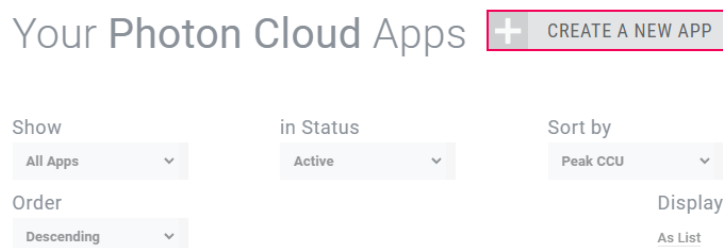
[Continue](#)

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.

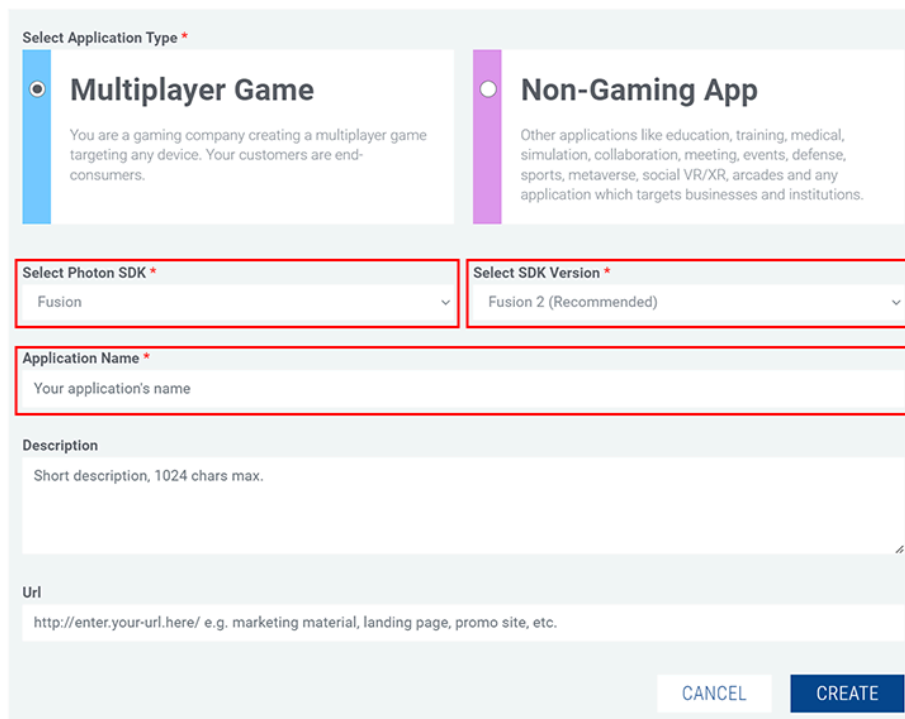
1. From the Photon Dashboard, select **Create a new app**.



2. Set **Photon SDK** to **Fusion** and make sure **SDK Version** is set to **Fusion 2**.

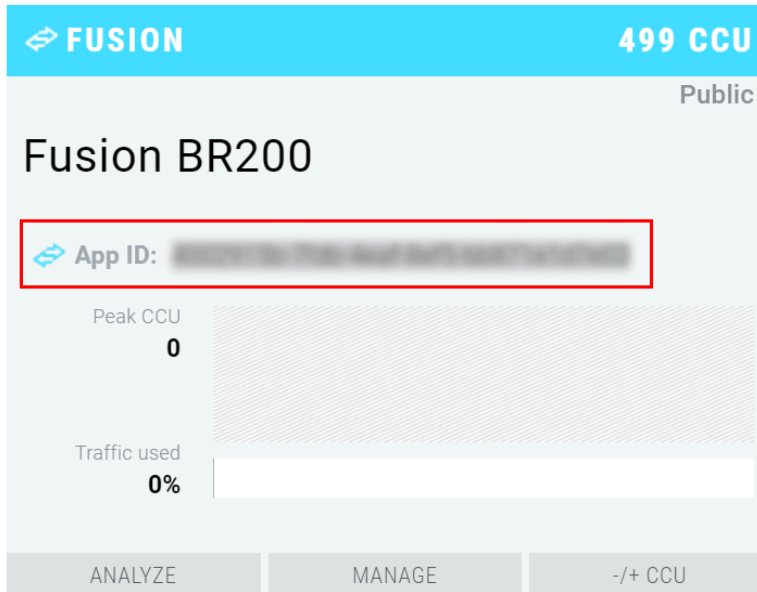
Create New Application

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



3. Name the application.

4. Optionally, provide a brief description and URL.
5. Select **Create**.
6. After creating the Fusion application, select it from the Photon Dashboard, then copy the **App ID**.

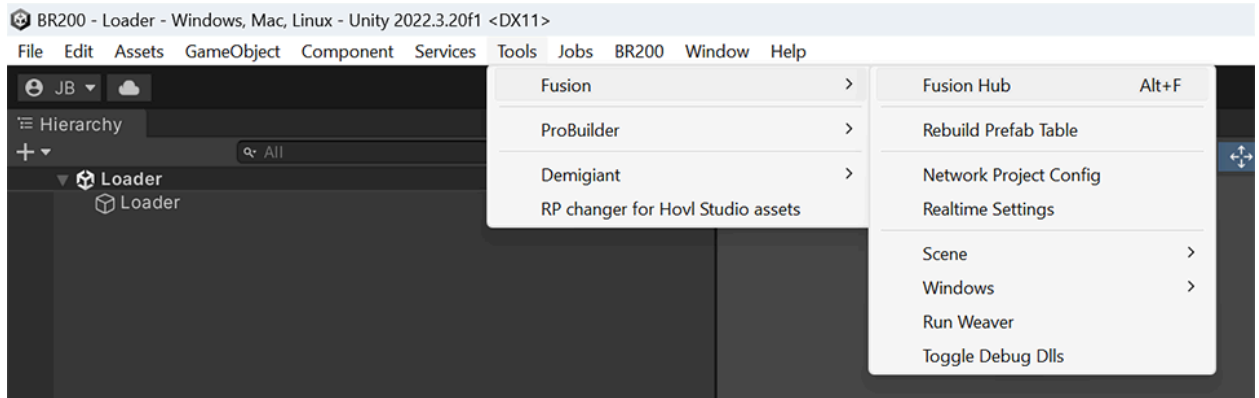


The screenshot shows the Photon Fusion application dashboard for an application named "Fusion BR200". At the top, a blue header bar displays "FUSION" on the left and "499 CCU" on the right. Below the header, the application name "Fusion BR200" is shown, with "Public" status indicated to its right. A red rectangular box highlights the "App ID" field, which contains a long alphanumeric string. Below the App ID, there are two performance metrics: "Peak CCU" with a value of "0" and "Traffic used" with a value of "0%". A large, light gray area with diagonal hatching is positioned to the right of these metrics. At the bottom of the dashboard, there are three buttons: "ANALYZE", "MANAGE", and "-/+ CCU".

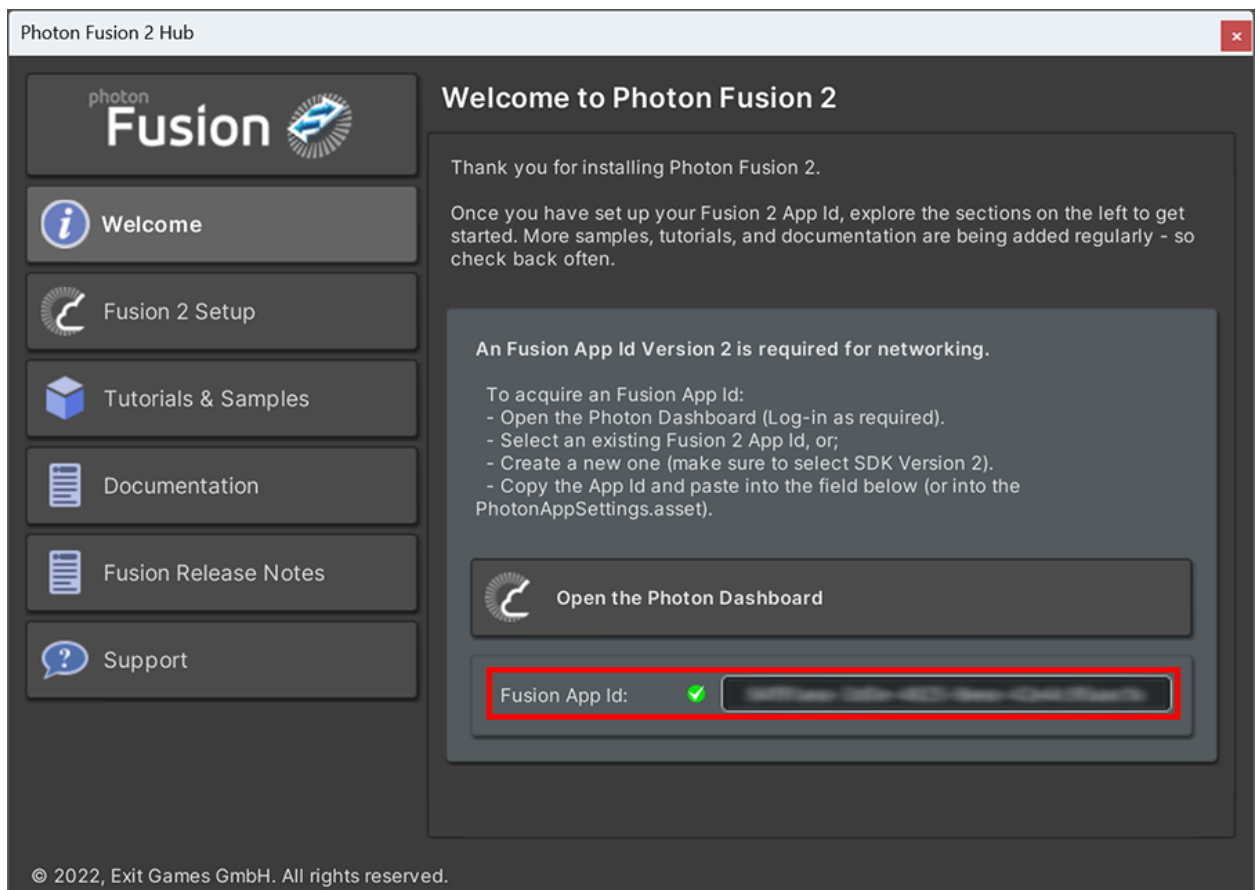
Link the Photon Fusion project

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

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



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

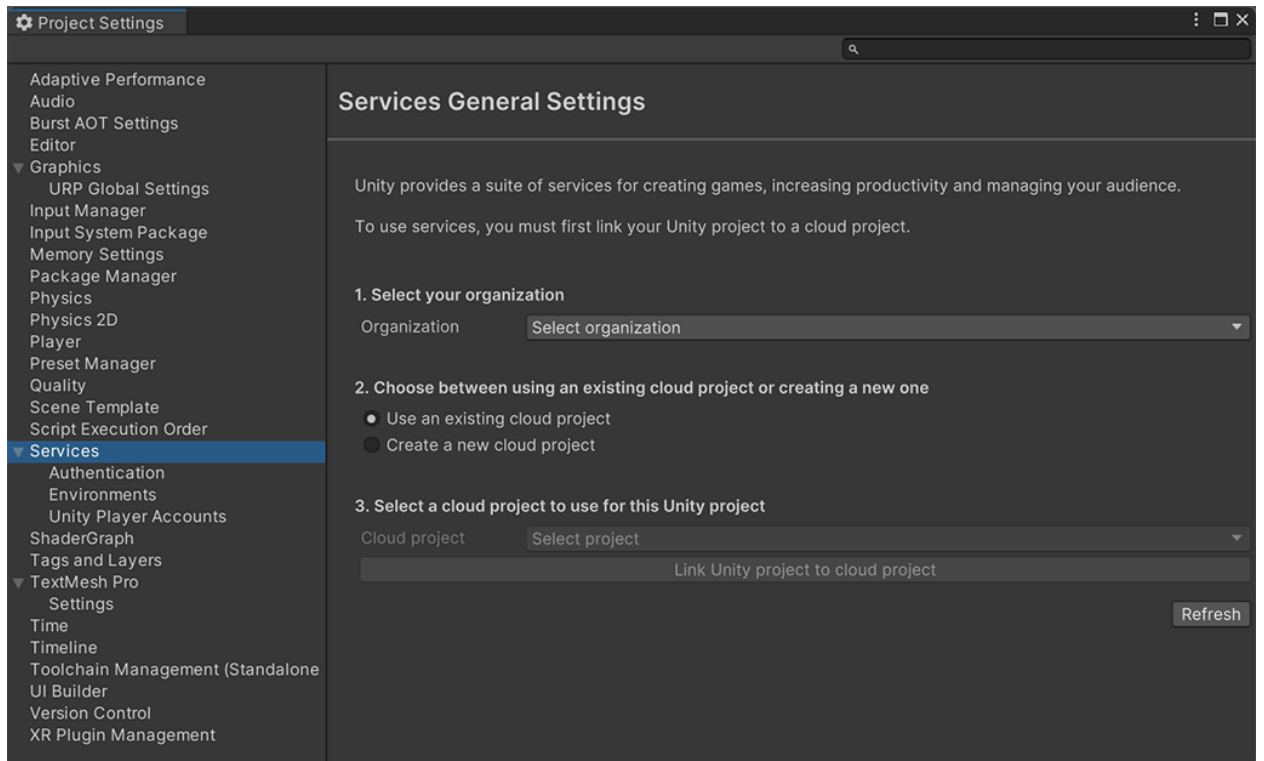


Link your Unity Gaming Services project

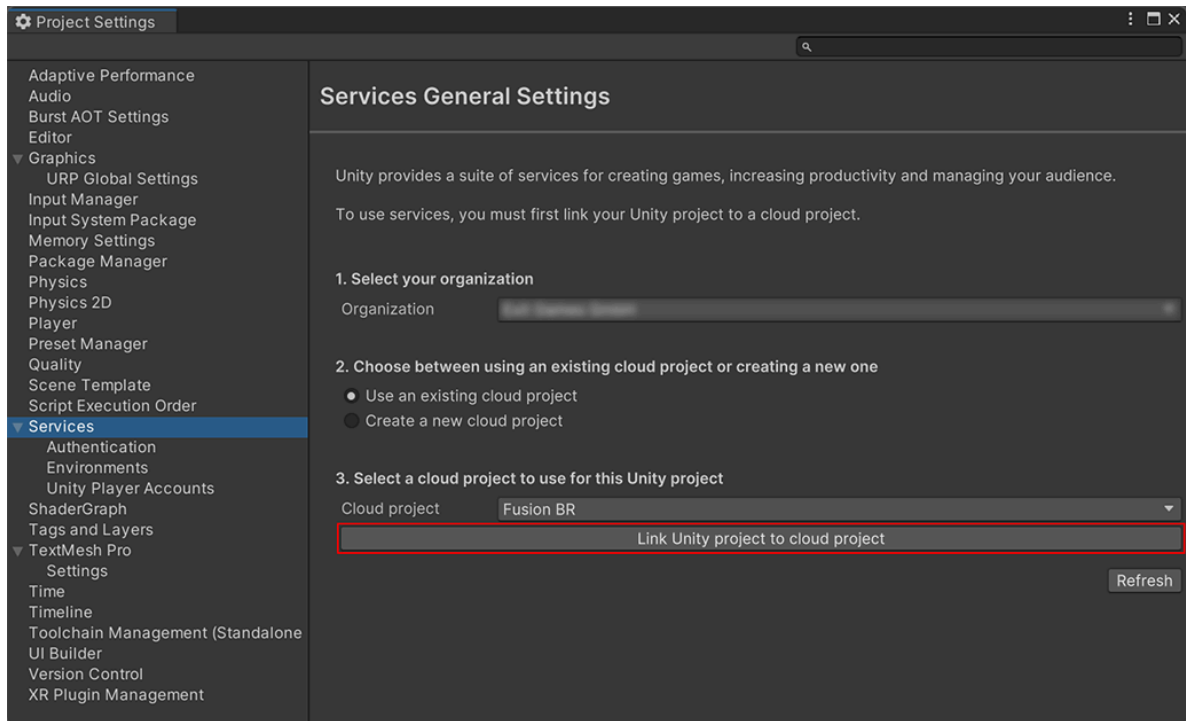
After the installation, link your UGS account and project with the Unity Editor.

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

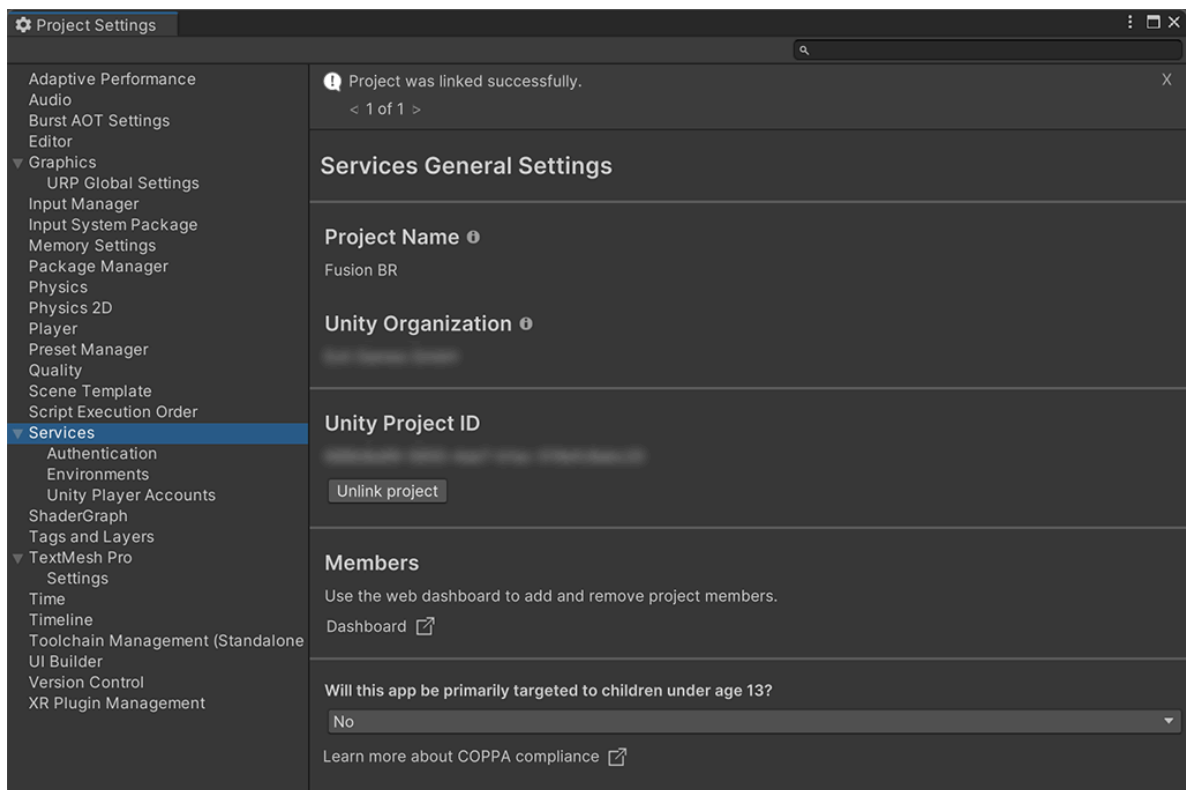
Note: You can only create a new cloud project if you have adequate permission within the organization.



4. Select cloud project and **Link Unity project to cloud project**.



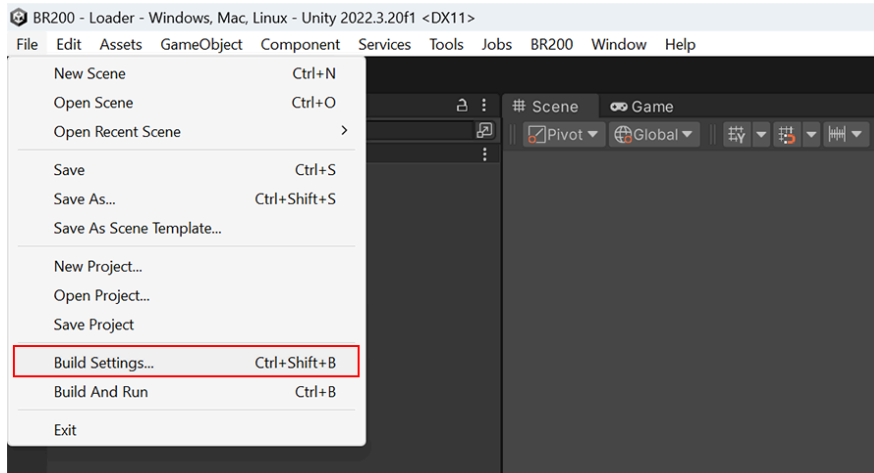
5. You should see a message stating that the project was linked successfully.



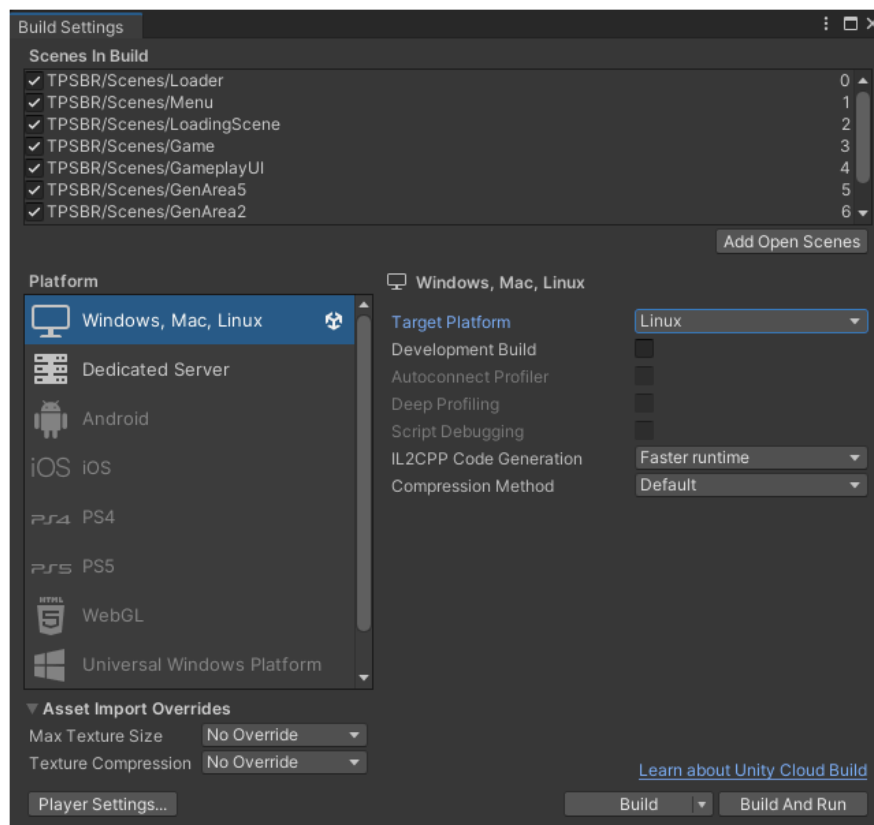
Build the standalone server

After linking your UGS project and your Fusion 2 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.
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 Game Server Hosting](#).

Note: There are multiple reasons to target the Dedicated Server platform, such as asset stripping. This platform is supported in BR200. See [Dedicated Server target](#) for more information about Dedicated Server platform.

Configure the Unity Game Server Hosting

The BR200 supports Unity Game Server Hosting to host game servers. Follow the instructions below to add the Game Server Hosting service to the sample project.

Warning: Game Server Hosting 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 Game Server Hosting. If you exceed the [free tier usage allowance](#), you will be charged. See our [Billing FAQ](#) to learn more.

Enable Game Server Hosting

Note: You must be an Owner or Manager of your organization to enable Game Server Hosting.

1. Sign in to the [Unity Cloud Dashboard](#) with your Unity account.
2. From the Unity Cloud, go to **Products > Game Server Hosting**.
3. Set up Game Server Hosting.

Note: You might need to add your credit card information before continuing. Game Server Hosting 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 Cloud to finish enabling Game Server Hosting for your project.
5. Follow the integrated Setup Guide, starting with integrating your game server.

Integrate your game server

The first step is integrating Game Server Hosting with your game through the Unity Editor. You should have completed most of this step in [Link your UGS project](#).

1. Select **Integrate your game server**.

Setup guide

production Reset guide

Get started with Game Server Hosting
Welcome! Use this setup guide to start hosting your game with Game Server Hosting. Each step in the guide is accompanied by a detailed walkthrough. Let's get started!

[Learn about Game Server Hosting](#) [Game Server Hosting documentation](#)

1 Integrate your game server
Use our Unity or Unreal packages and SDKs to integrate your game server with Game Server Hosting.
[Integrate game server](#)

2 Create a build
A build contains the files to run your game servers. For more information, see the builds [documentation](#).
[Create a build](#)




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.

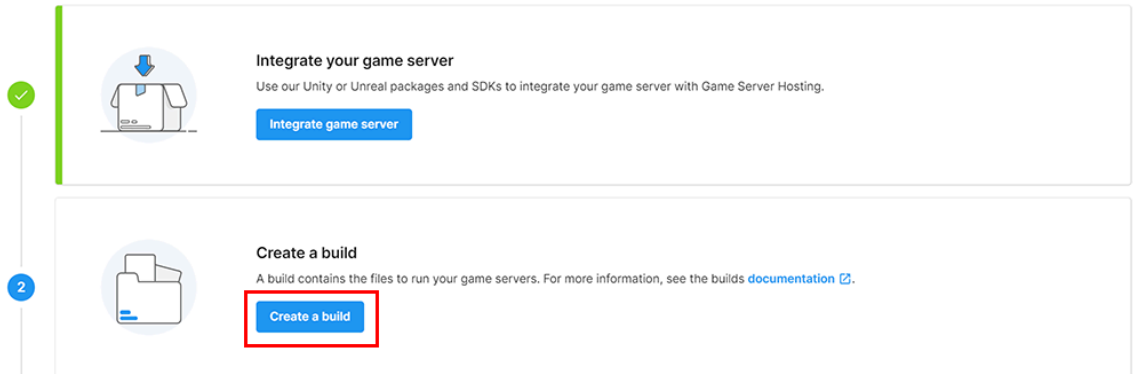
4. Select **Finish**.

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

Create a build

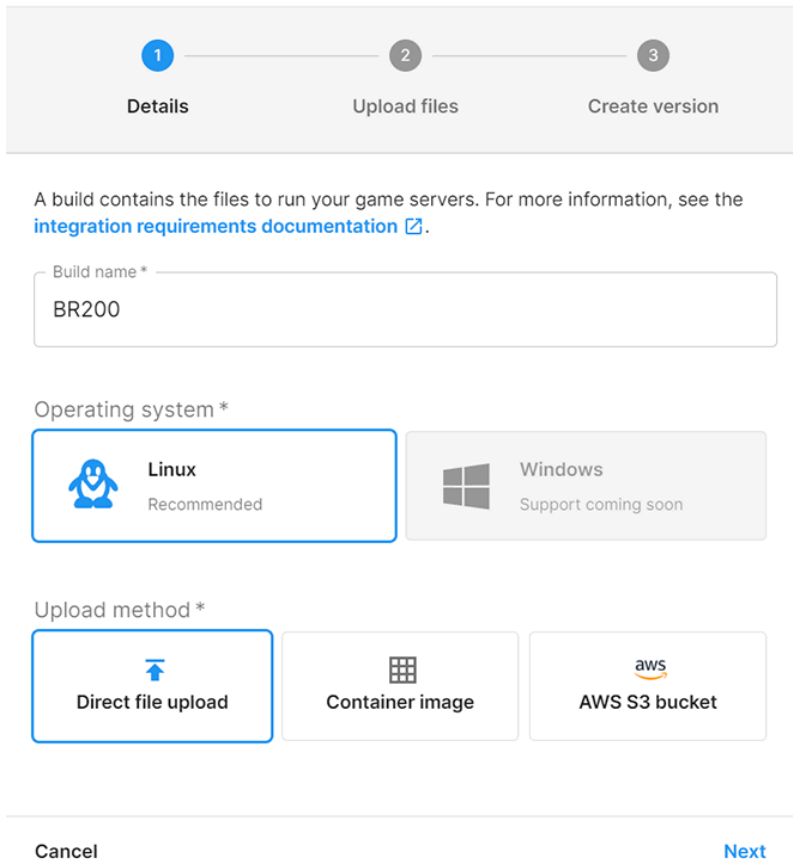
Create a build of your game within the Game Server Hosting service. See the [Builds documentation](#) to learn more.

1. Select **Create a build**.



The screenshot shows a two-step process. Step 1, 'Integrate your game server', is marked with a green checkmark and a '1' in a circle. It includes a laptop icon, a description, and an 'Integrate game server' button. Step 2, 'Create a build', is marked with a blue '2' in a circle and is highlighted with a red box. It includes a folder icon, a description, and a 'Create a build' button.

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



The form shows a progress bar with three steps: 1. Details (active), 2. Upload files, and 3. Create version. Below the progress bar, there is a text input for 'Build name *' containing 'BR200'. Under 'Operating system *', 'Linux' is selected and highlighted with a blue border, with 'Windows' as an alternative. Under 'Upload method *', 'Direct file upload' is selected and highlighted with a blue border, with 'Container image' and 'AWS S3 bucket' as alternatives. At the bottom, there are 'Cancel' and 'Next' buttons, with 'Next' being highlighted in blue.

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**.


✓
Details

2
Upload files

3
Create version

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

CancelUpload 53 Files


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

🔍 Search files

Name ↑	Status	
BR200_Data/app.info	● Ready to upload	🗑️
BR200_Data/boot.config	● Ready to upload	🗑️
BR200.x86_64	● Ready to upload	🗑️
GameAssembly.so	● Ready to upload	🗑️

⏪⏩

Rows per page: 10 ▾ 1-10 of 53 ⏪ ⏩

CancelNext

6. Select **Next**.

Progress bar: 1 (Details) — 2 (Upload files) — 3 (Create version)

Upload complete
53 files uploaded successfully | 0 files failed to upload
527.86 MB of 527.86 MB uploaded

Search files

Name ↑	Status	
BR200_Data/app.info	● Added	🗑️
BR200_Data/boot.config	● Added	🗑️
BR200_Data/globalgamemangers	● Added	🗑️

Rows per page: 10 ▾ 1-10 of 53 < >

Cancel Next

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

Progress bar: 1 (Details) — 2 (Upload files) — 3 (Create version)

Powered by [Unity Cloud Content Delivery](#)

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

Version name

Leave empty if you would like this update to automatically be given a version name

Build name	Version
BR200	v1709286012185

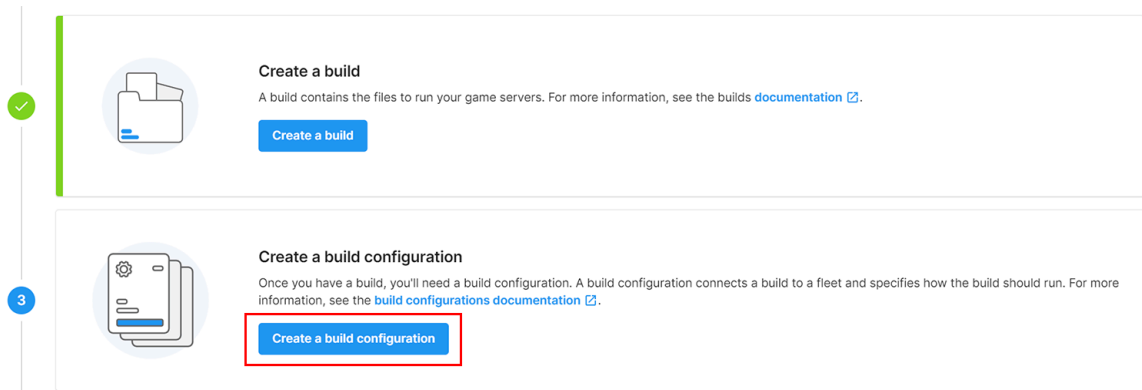
Cancel Finish

Create a build configuration

Create a build configuration for the build you created in the previous step. See the [Build configurations 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**.

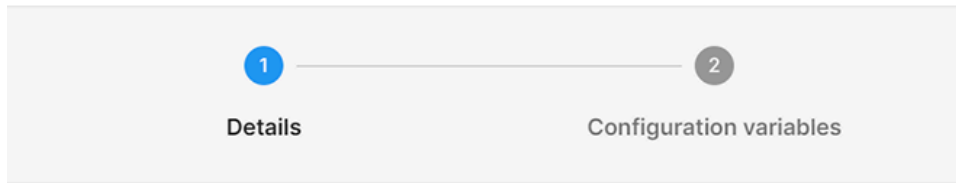



2. Fill in the build configuration details.

- Name the build configuration.
- Select the build you created in the previous step.
- Select the build executable.
- Set the **Query type** to **SQP**.
- Enable **Custom launch parameters**, then use the following 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 -snp -matchmaking -maxPlayers 200
```


3. Select **Next**.



A build configuration connects a build to a fleet and specifies how the build should run. For more information, see our [documentation](#) .

Build configuration name *

Build *

Select a build to assign to this build configuration.

Game server executable


Define the game server executable within your build by typing its name.

Query type 

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

Server readiness 

Enable server readiness for this build configuration.

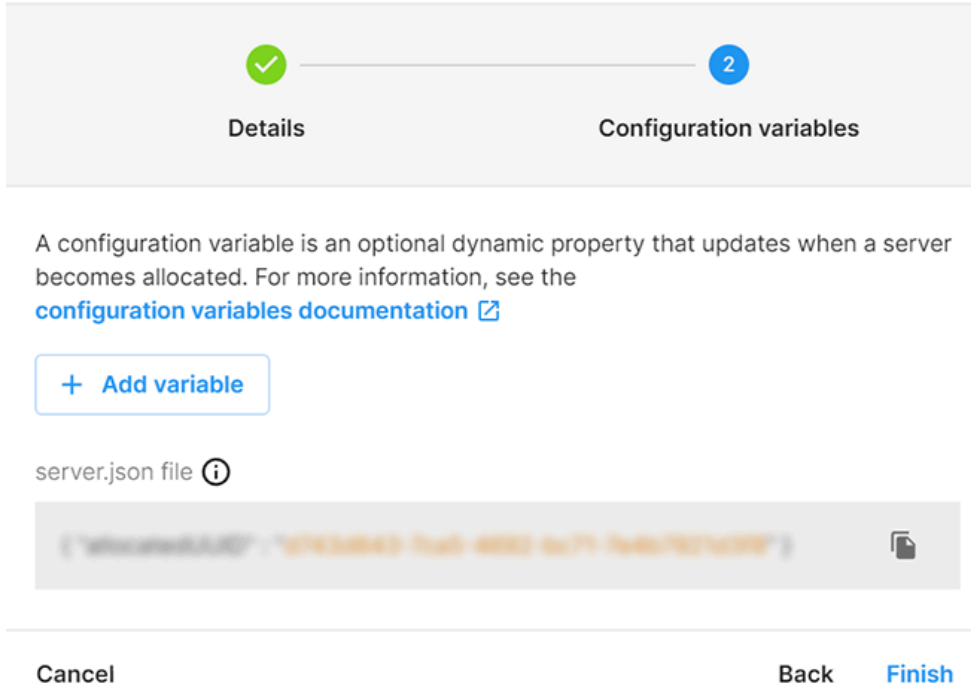
Launch parameters 


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

Cancel


[Next](#)

4. Select **Finish**.



A configuration variable is an optional dynamic property that updates when a server becomes allocated. For more information, see the [configuration variables documentation](#) 

[+ Add variable](#)

server.json file 

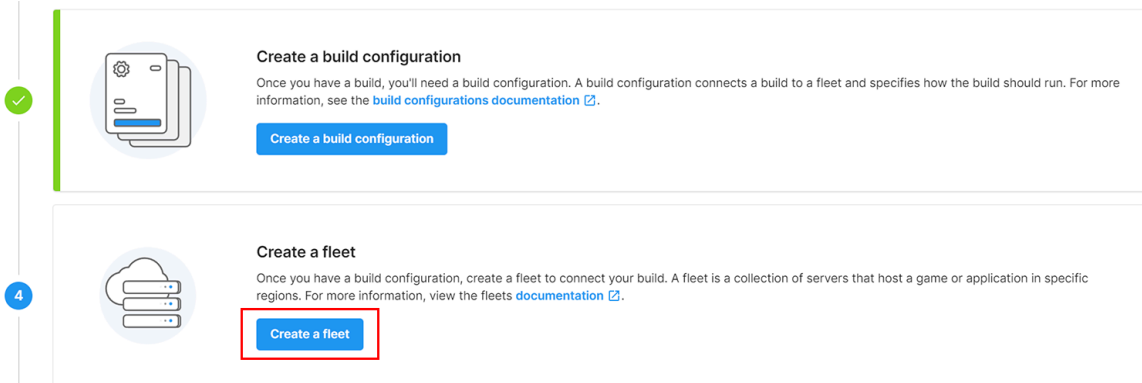
```
{ "serverName": "1234567890", "serverAddress": "1234567890", "serverPort": "1234567890" }
```


Cancel Back Finish

Create a fleet


Create a fleet to host your game servers. See the [Fleets documentation](#) to learn more.

1. Select **Create a fleet**.



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 build configuration](#)

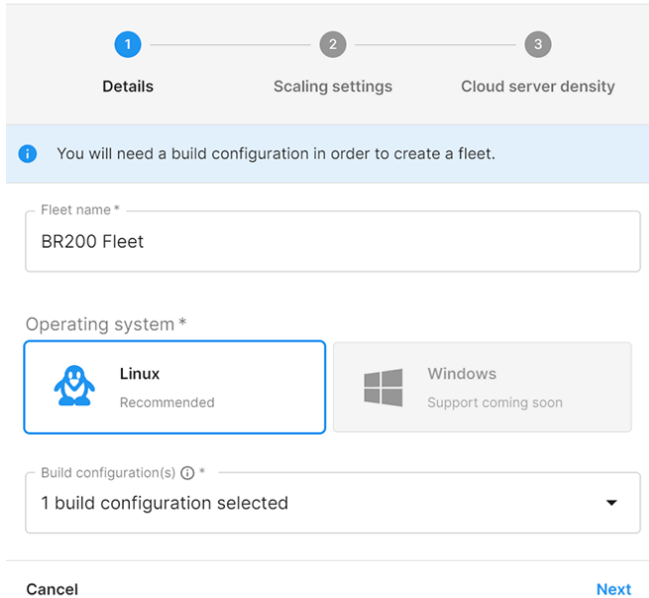
Create a fleet
Once you have a build configuration, create a fleet to connect your build. A fleet is a collection of servers that host a game or application in specific regions. For more information, view the [fleets documentation](#) .

[Create a fleet](#)

2. Fill in the fleet details:

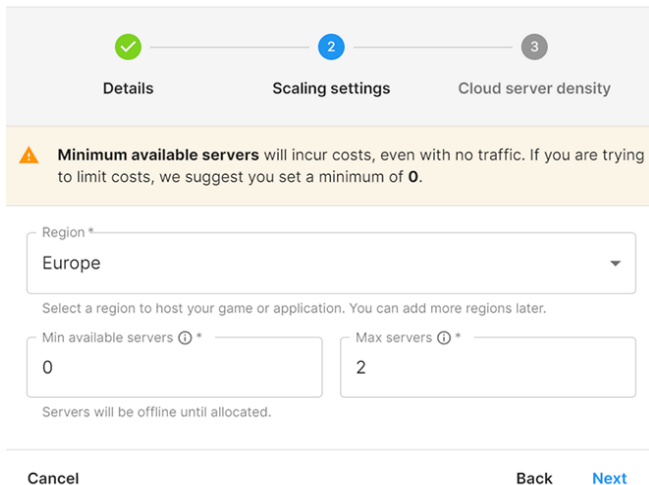
- a. Name the fleet.
- b. Set the **Operating system** to **Linux**.
- c. Select the build configuration you created in the previous step.

3. Select **Next**.



4. Specify the scaling settings:

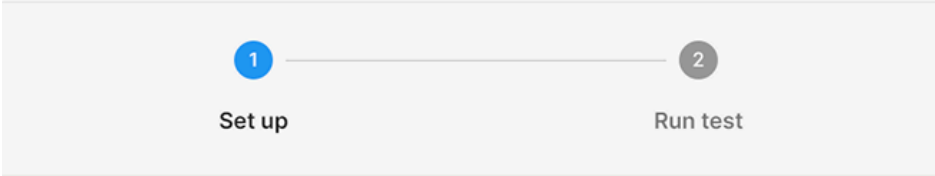
- a. Select a region.
- b. Set the **Min available servers** to a value less than or equal to 1.
- c. Specify the **Max servers** to a value equal to or greater than the Min available servers value



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


Warning: Any number of available servers incurs costs, even without traffic. If you are in development or trying to limit costs in a low traffic environment, set the **Min available servers** value to 0.

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



1 ————— **2**
Set up 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. For more information, view the [allocations documentation](#) .

Fleet *
BR200 Fleet ▼

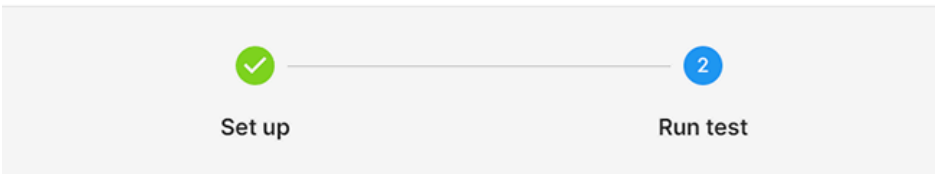
Region *
Europe ▼

Build configuration *
BR200 ▼

Cancel

[Next](#)

3. Select **Next**.
4. Select **Run test**.



✓ ————— **2**
Set up Run test

Run a test allocation using the Game Server Hosting interface

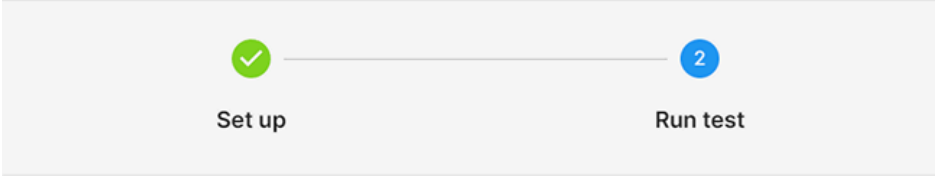
Run test

Cancel

[Back](#)

[Finish](#)

5. Wait for the test to complete.



i An allocation might take some time if a new server is being created.

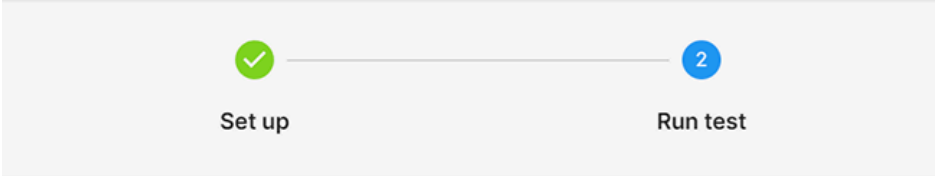
- Sending allocation
- Waiting for server

Server allocated successfully


Cancel


Back Finish

6. Select **Finish**.



✓ Test allocation successful.

Test allocation ID XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX 

Server IP:Port **i** XX.XXX.XX.XXX:XXXX 

Time remaining **i** 59m 51s

Cancel

Back **Finish**

Congratulations! You've successfully set up Game Server Hosting with the BR200.

Configure the Unity Matchmaker

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

Enable Matchmaker

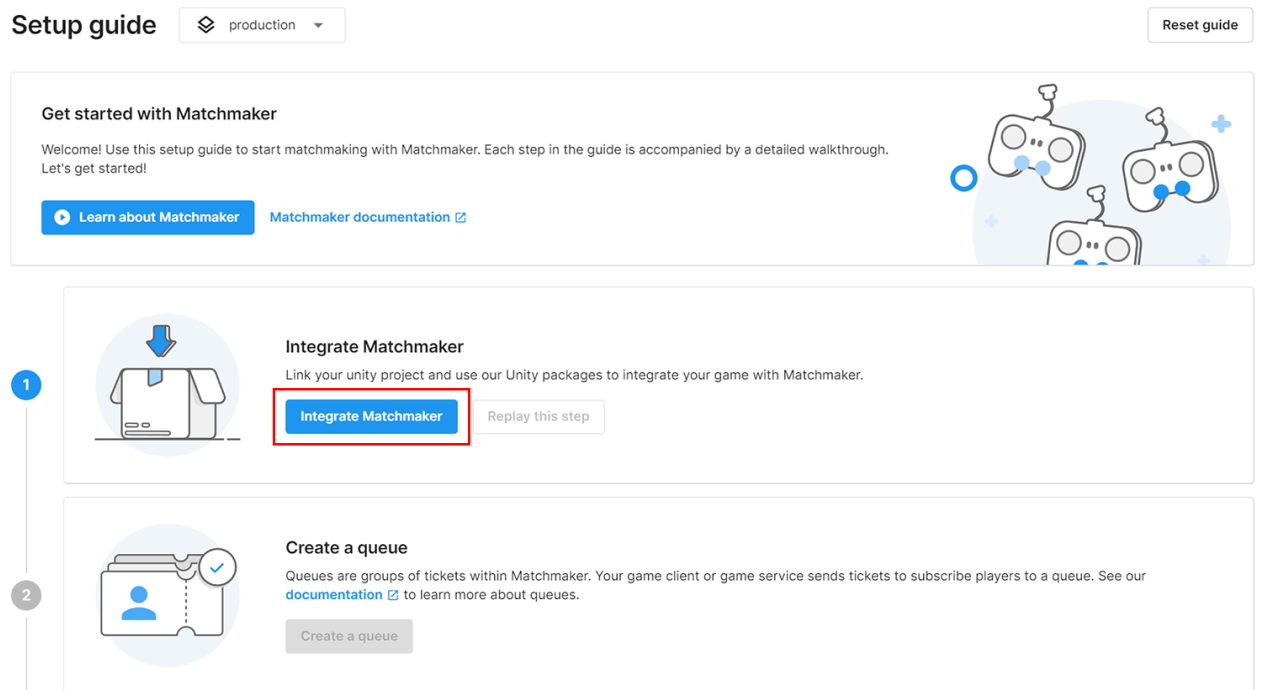
Note: You might need to enter payment information to continue the trial. If prompted, enter your payment information, then select **Complete onboarding**.

1. Sign in to the [Unity Cloud Dashboard](#) with your Unity account.
2. From the Unity Cloud, go to **Products > Matchmaker**.
3. 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 documentation](#) for help.

1. Select **Integrate Matchmaker**.



Setup guide production Reset guide

Get started with Matchmaker

Welcome! Use this setup guide to start matchmaking with Matchmaker. Each step in the guide is accompanied by a detailed walkthrough. Let's get started!

[Learn about Matchmaker](#) [Matchmaker documentation](#)

1 Integrate Matchmaker

Link your unity project and use our Unity packages to integrate your game with Matchmaker.

[Integrate Matchmaker](#) Replay this step


2 Create a queue

Queues are groups of tickets within Matchmaker. Your game client or game service sends tickets to subscribe players to a queue. See our [documentation](#) to learn more about queues.

[Create a queue](#)

2. Set the **Game engine** to **Unity**.
3. Set the **Integration method** to **SDK**.

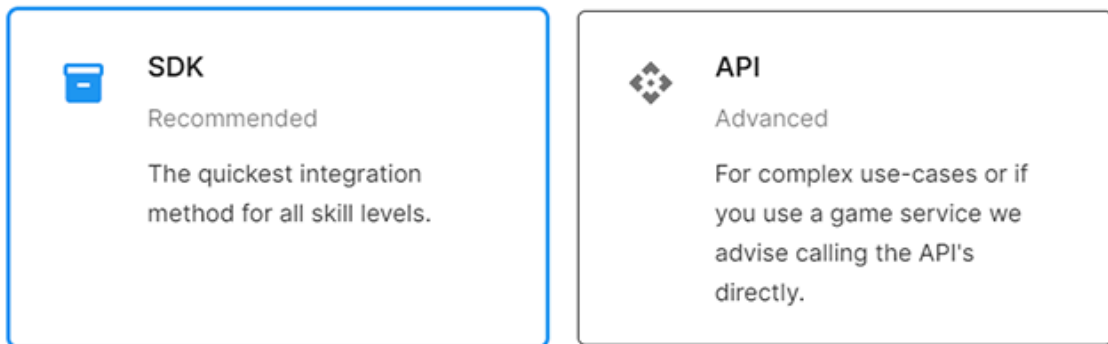


To begin Matchmaker integration please select your game engine and preferred integration method. For more information, see the [Integration documentation](#) .

Game engine:



Integration method:



Cancel

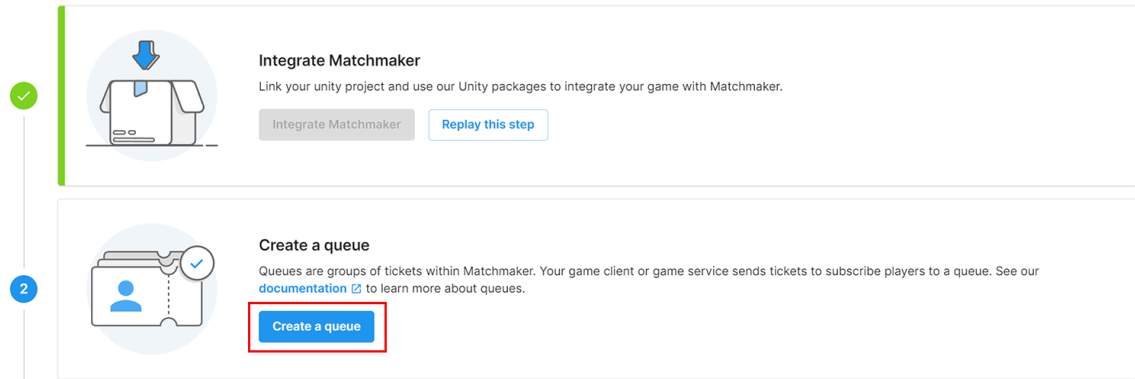
Next

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 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 in step 2 is highlighted with a red border.

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

Warning: Using a queue name other than "battleRoyale" results in an exception.

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

Queues are groups of tickets within Matchmaker. For more information on queues, view our [documentation](#).

Queue name *

battleRoyale

Maximum players on a ticket *

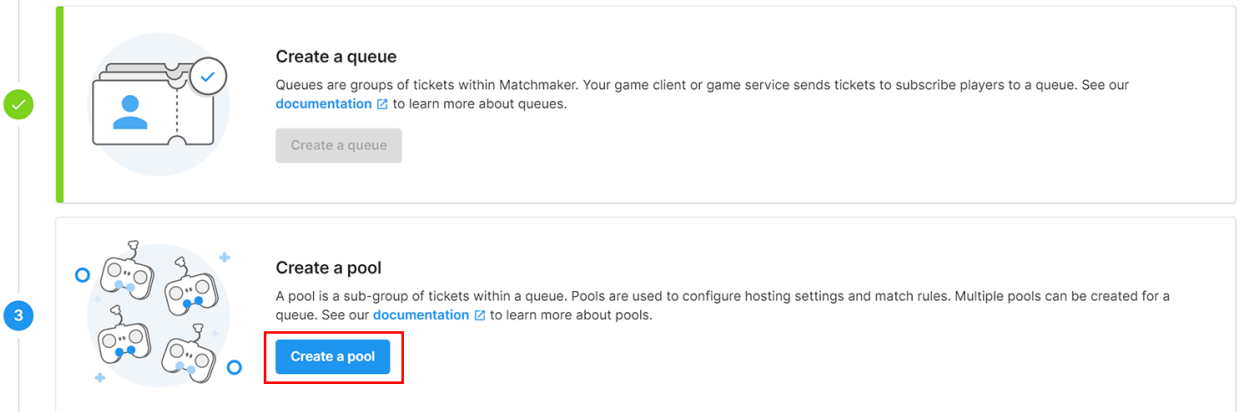
2

Cancel **Create**

Create a default pool

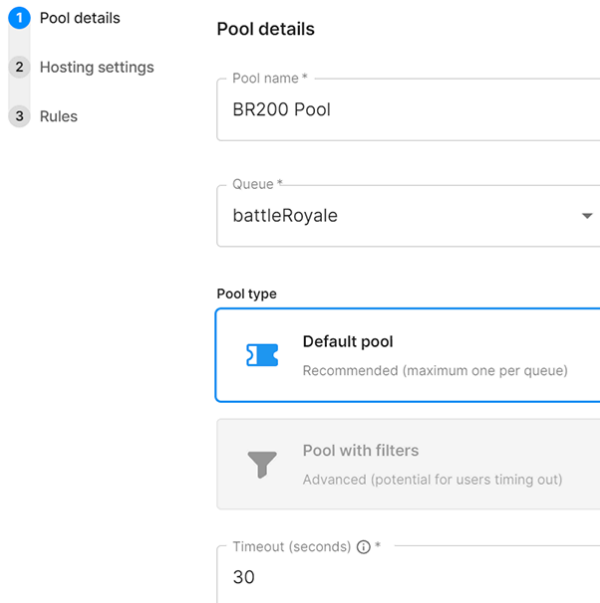
Create a default pool for your game. See the [Queues and Pools documentation](#) for help.

1. Select **Create a pool**.



The screenshot shows two steps in a vertical list. Step 1, 'Create a queue', is marked with a green checkmark and a grey button labeled 'Create a queue'. Step 2, 'Create a pool', is marked with a blue circle containing the number '3' and a blue button labeled 'Create a pool' which is highlighted with a red box. The text for 'Create a pool' reads: 'A pool is a sub-group of tickets within a queue. Pools are used to configure hosting settings and match rules. Multiple pools can be created for a queue. See our [documentation](#) to learn more about pools.'

2. Fill in the **Pool details**:
 - a. Give the pool a name.
 - b. Select the queue you created in previous step
 - c. Set the timeout to **30** seconds.
3. Select **Next**.

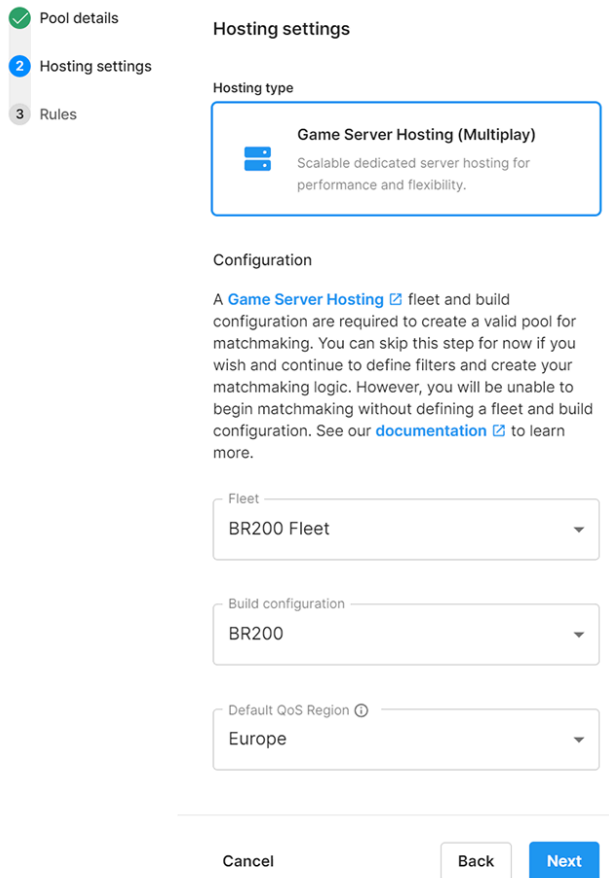


The screenshot shows the 'Pool details' form. On the left, a vertical list of steps is shown: '1 Pool details' (highlighted), '2 Hosting settings', and '3 Rules'. The form fields are: 'Pool name *' with the value 'BR200 Pool'; 'Queue *' with a dropdown menu showing 'battleRoyale'; 'Pool type' with two options: 'Default pool' (Recommended (maximum one per queue)) and 'Pool with filters' (Advanced (potential for users timing out)); and 'Timeout (seconds) ⓘ *' with the value '30'.

Cancel

Next

4. Fill in the **Hosting settings**:
 - a. Select the [fleet you created earlier](#).
 - b. Select the [build configuration you created earlier](#).
 - c. Select the **Default QoS Region**. This should be the region [you selected for your fleet when you set up Game Server Hosting](#).
5. Select **Next**.



The screenshot shows the 'Hosting settings' configuration page in the Unity console. On the left, a progress indicator shows three steps: 'Pool details' (completed), 'Hosting settings' (current step), and 'Rules' (pending). The main content area is titled 'Hosting settings' and includes a 'Hosting type' section with a highlighted 'Game Server Hosting (Multiplay)' option, described as 'Scalable dedicated server hosting for performance and flexibility.' Below this is a 'Configuration' section with a warning message: 'A Game Server Hosting fleet and build configuration are required to create a valid pool for matchmaking. You can skip this step for now if you wish and continue to define filters and create your matchmaking logic. However, you will be unable to begin matchmaking without defining a fleet and build configuration. See our documentation to learn more.' Three dropdown menus are visible: 'Fleet' set to 'BR200 Fleet', 'Build configuration' set to 'BR200', and 'Default QoS Region' set to 'Europe'. At the bottom, there are 'Cancel', 'Back', and 'Next' buttons.

6. Configure the **Rules**:
 - a. Set the Match definition name to **Battleroyale Match Definition**.
 - b. Set **Backfill enabled** to **True**.
 - c. 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: If you set a value different than 200 you must go back to Game Server Hosting and configure the app launch parameters on the build configuration to reflect the maximum number of players you set here.

Match definition

Name *

Backfill enabled * True

Give this Match definition a name and/or brief description. Select whether to enable backfill support. [See Backfill.](#)

Team definitions

Create teams and define rules to govern how tickets/players are assigned to those teams, see our [documentation](#) to learn more.

Default team (required)

Team name *

Give this team a name.

Team count

Define the minimum and maximum number of replicas of this team and add relaxations if required, see our [documentation](#) to learn more.

Team count min * Team count max *

Minimum number of replicas of the team. Maximum number of replicas of the team.

[+ Add range relaxation](#)

Player count

Define the minimum and maximum number of players in this team and add relaxations if required, see our [documentation](#) to learn more.

Player count min * Player count max *

Minimum number of players in the team. Maximum number of players in the team.

[+ Add range relaxation](#)

7. Select **Create**.

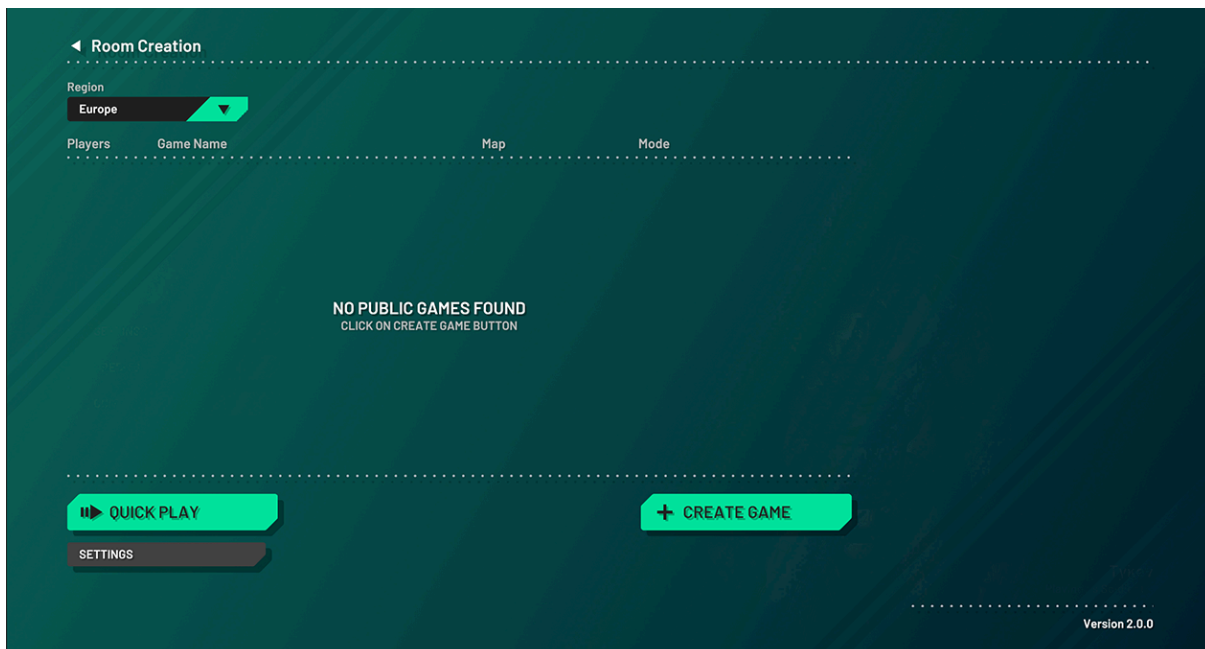
Congratulations! You've successfully configured the Unity Matchmaker. You can go to **Products > Matchmaker > Overview** to view matchmaking traffic and match times.

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.

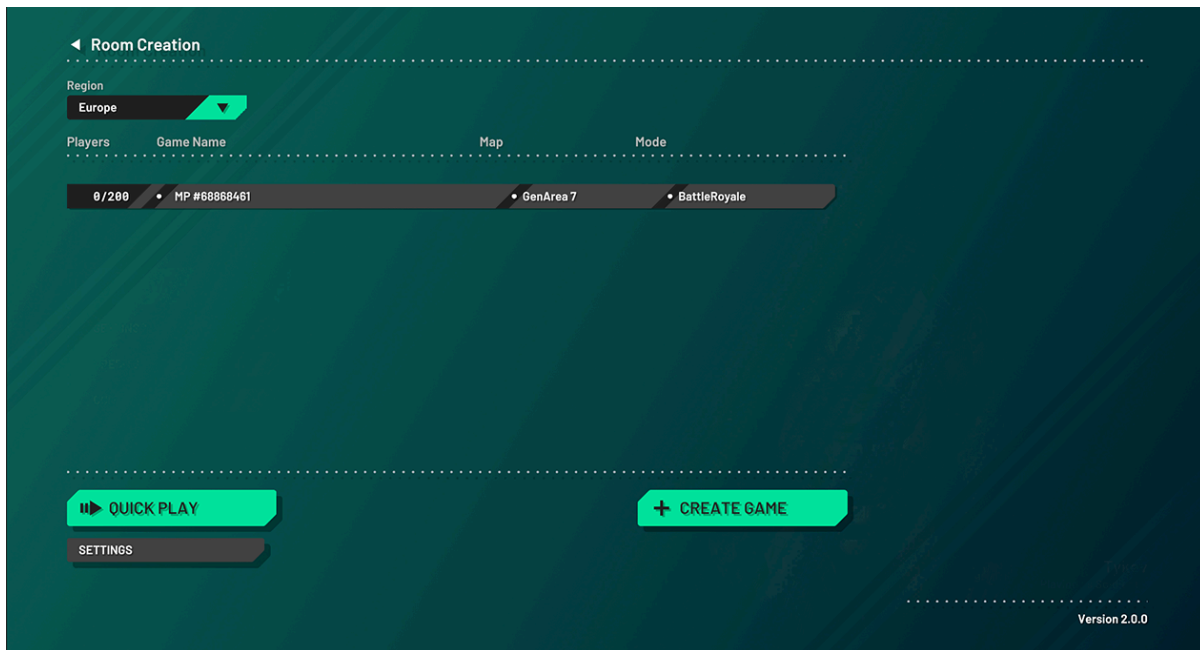


After launching, the game client shows a game session list based on the available game sessions on Unity Game Server Hosting. If this is the first time you're running the application and haven't already started any sessions, you won't see any game sessions available yet.

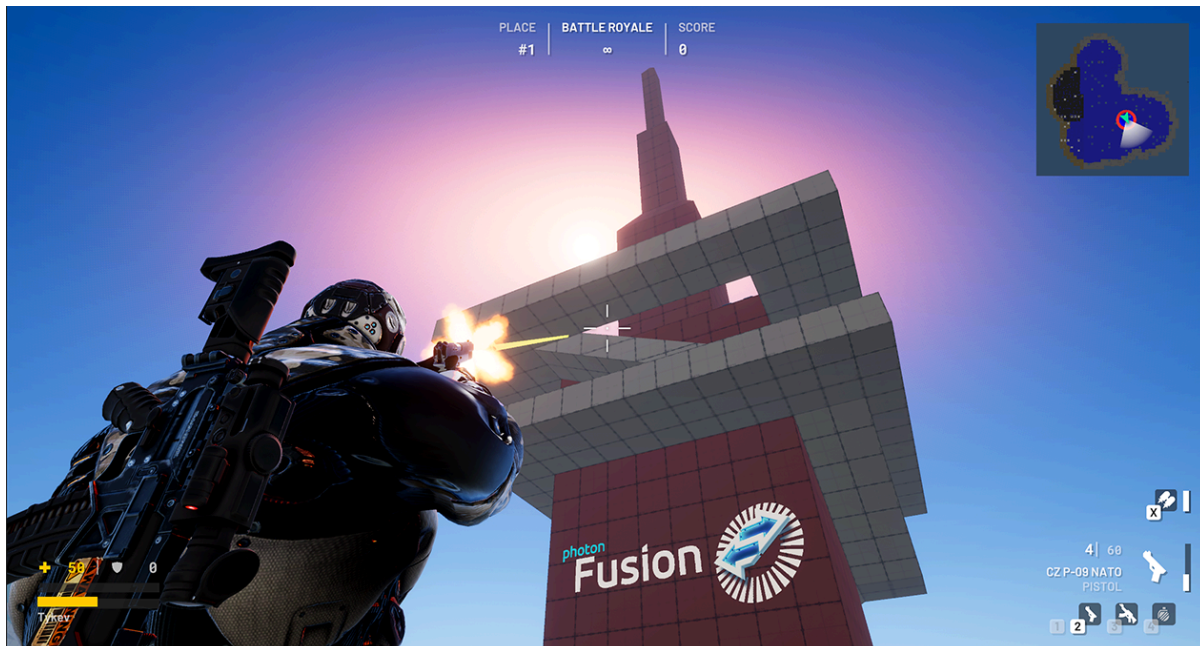


Tip: You can go back to the Game Server Hosting and Matchmaker dashboards to view game performance metrics.

Select **Quickplay** to enter the matchmaker and start servers on Game Server Hosting. 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 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 Game Server Hosting.

1. Log in to the [Unity Cloud Dashboard](#).
2. Go to **Products > Game Server Hosting > Builds**.
3. Select the build you created in the [Create a build](#) step.
4. Select **Files**.
5. Select **Update files**, then upload the new files from the generated build.
6. Wait for the new version to sync.

Once synced, you can test the updated build live on Unity Cloud servers.

Implementation details

To add Game Server Hosting, you'll need to extend your game host lifecycle in several places.

MultiplayManager

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 configurations](#).

`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 Game Server Hosting 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