



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](#)

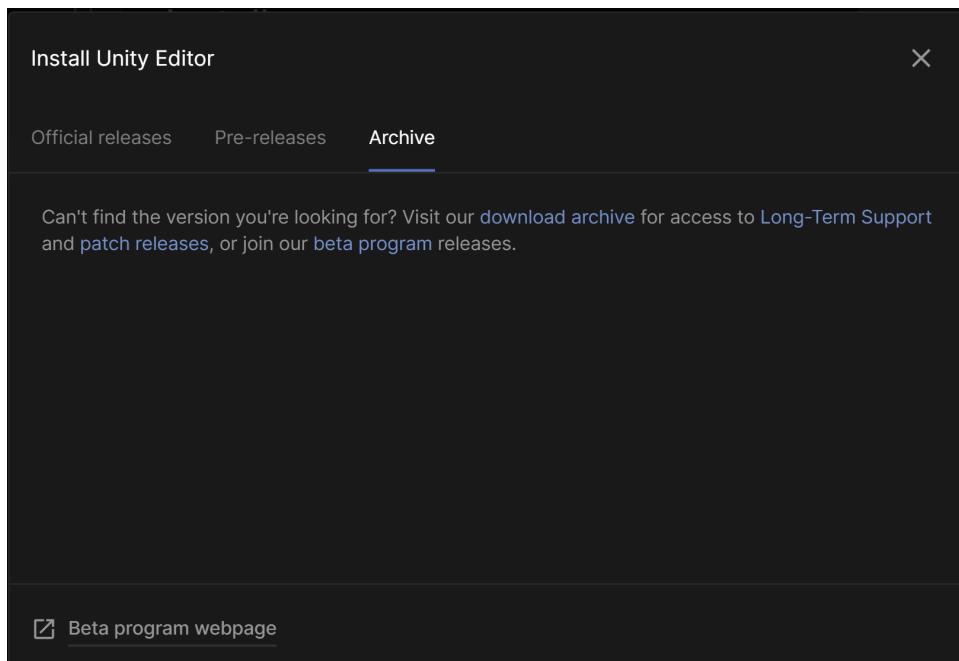
Note: Visit [Unity Dashboard Support](#) if you need help with any Unity services. Visit [Photon Contact](#) page for help with Photon Fusion.

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.

The screenshot shows the Unity download archive interface. At the top, there's a section for 'Long Term Support releases' with a button to 'Download LTS Releases'. Below this, a navigation bar includes links for Unity 2023.X, Unity 2022.X (which is highlighted in blue), Unity 2021.X, Unity 2020.X, Unity 2019.X, Unity 2018.X, Unity 2017.X, and Unity 5.X. The main content area displays three Unity versions: Unity 2022.3.20 (February 14, 2024), Unity 2022.3.19 (January 31, 2024), and Unity 2022.3.18 (January 20, 2024). Each version entry includes a 'Unity Hub' button, dropdown menus for 'Downloads (Win)', 'Downloads (Mac)', and 'Downloads (Linux)', and a 'Release Notes' button. A red box highlights the 'Release Notes' button for the Unity 2022.3.20 entry.

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.

The screenshot shows the Unity Hub application window. On the left, there are tabs for 'Projects', 'Installs' (which is selected), 'Learn', and 'Community'. The main area is titled 'Installs' and shows a dialog box for 'Install Unity 2022.3.20f1 LTS'. The dialog lists various build supports under 'Add modules': Android SDK & NDK Tools, iOS Build Support, tvOS Build Support, **Linux Build Support (IL2CPP)** (which is checked and highlighted with a red box), Linux Build Support (Mono), Linux Dedicated Server Build Support, Mac Build Support (Mono), Mac Dedicated Server Build Support, Universal Windows Platform Build Support, WebGL Build Support, Windows Build Support (IL2CPP), and Windows Dedicated Server Build Support. At the bottom of the dialog, there are buttons for 'Locate' and 'Install Editor'. Below the dialog, the Unity Hub interface shows a sidebar with 'Downloads' and a footer with the text 'C:\Unity\2021.3.20f1\Editor\Unity.exe' and platform options: Android, Linux, macOS, WebGL, iOS, Windows. The footer also shows the date '2021.3.20f1 LTS'.

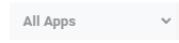
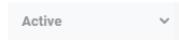
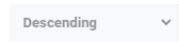
Get started with Photon Fusion

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**.

Your Photon Cloud Apps 

Show  in Status  Sort by 
Order  Display 

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

Create New Application

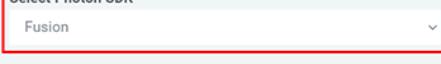
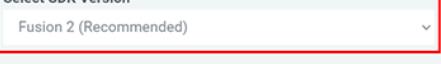
The application defaults to the **Free Plan** for development only.
You can change the plan at any time.

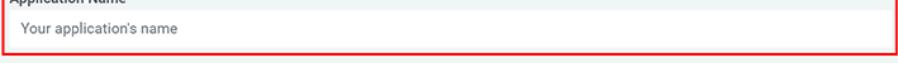
Select Application Type *

Multiplayer Game **Non-Gaming App**

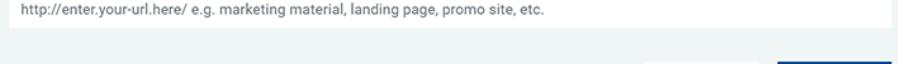
You are a gaming company creating a multiplayer game targeting any device. Your customers are end-consumers.

Other applications like education, training, medical, simulation, collaboration, meeting, events, defense, sports, metaverse, social VR/XR, arcades and any application which targets businesses and institutions.

Select Photon SDK *  Select SDK Version * 

Application Name * 

Description 

Url 

CANCEL **CREATE**

3. Name the application.



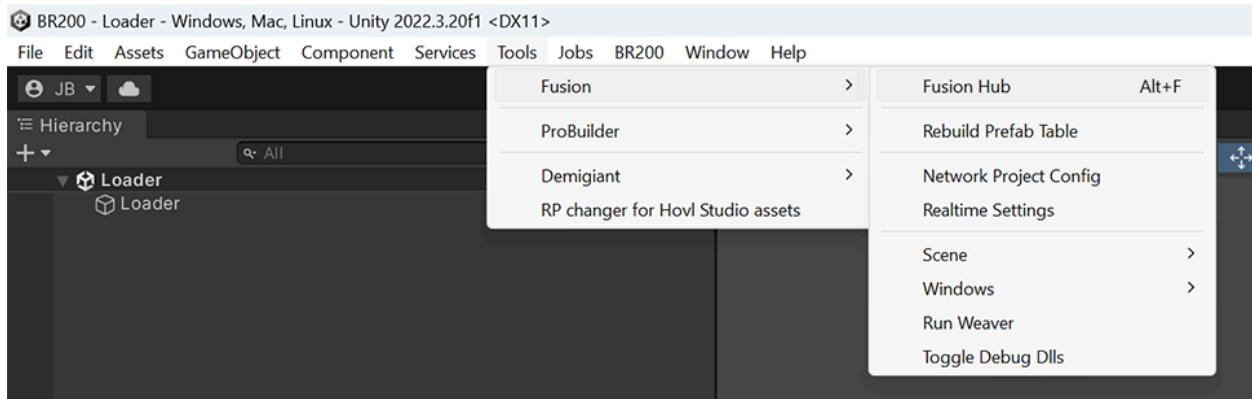
4. Optionally, provide a brief description and URL.
5. Select **Create**.
6. After creating the Fusion application, select it from the Photon Dashboard, then copy the **App ID**.

The screenshot shows the Photon Dashboard interface. At the top, there's a blue header bar with the text "FUSION" and "499 CCU". Below the header, the word "Public" is visible. The main content area displays the name "Fusion BR200". A red box highlights the "App ID" field, which contains a placeholder text starting with "appID: ". Below this, there are two performance metrics: "Peak CCU" (0) and "Traffic used" (0%). At the bottom, there are three buttons: "ANALYZE", "MANAGE", and "-/+ CCU".

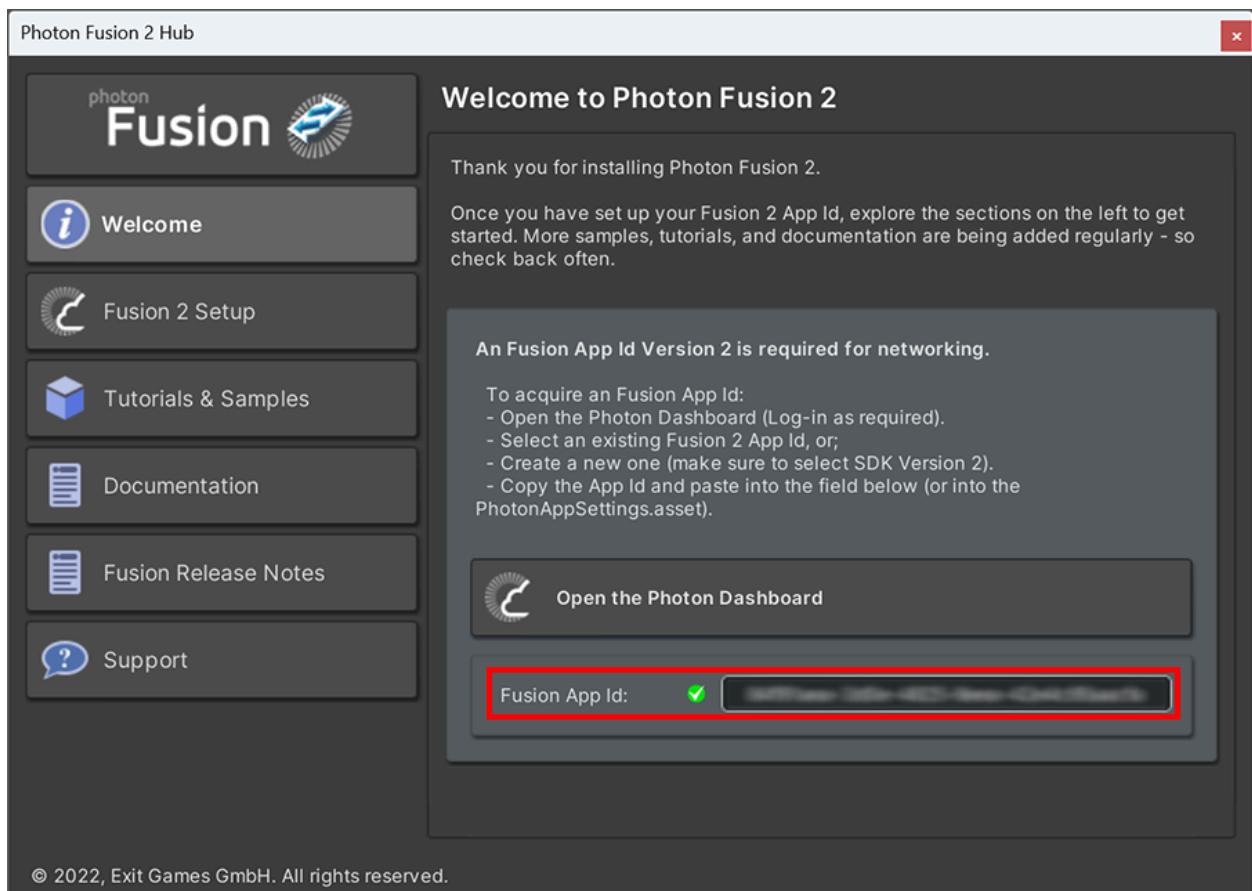
Link the Photon Fusion project

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.



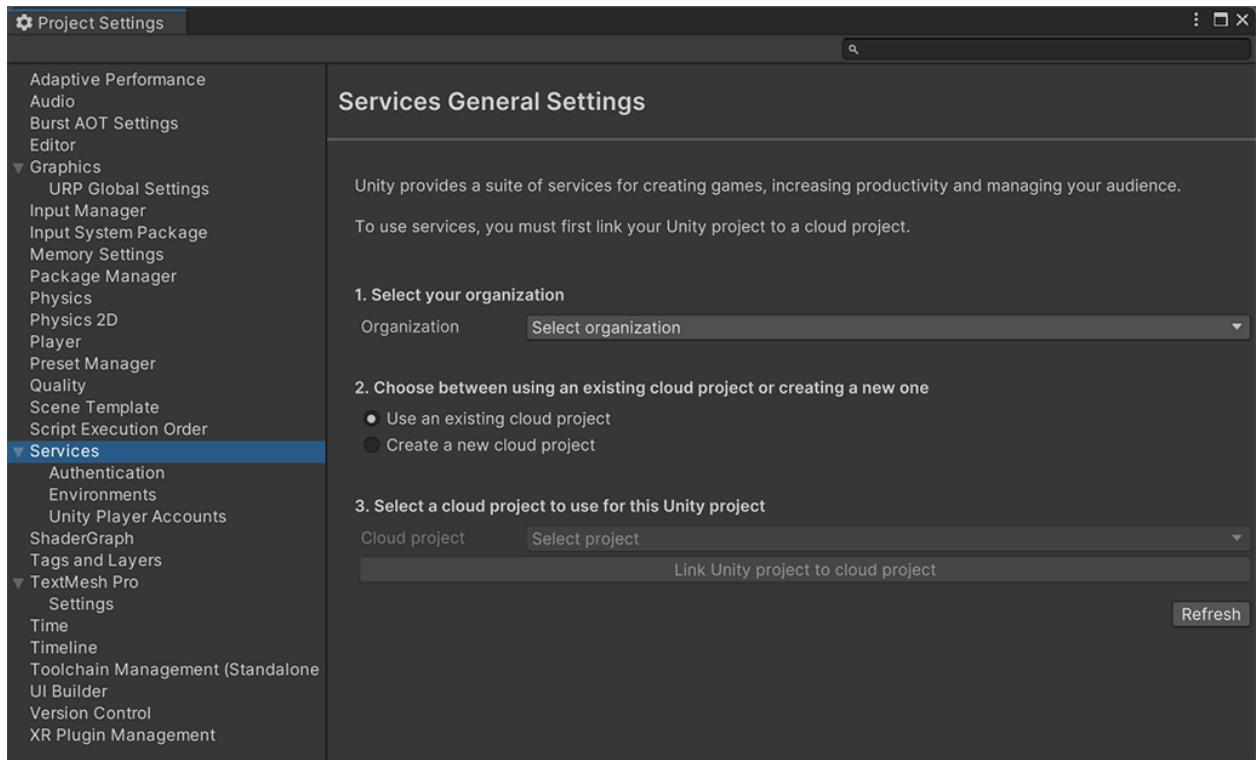


Link your Unity Gaming Services project

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

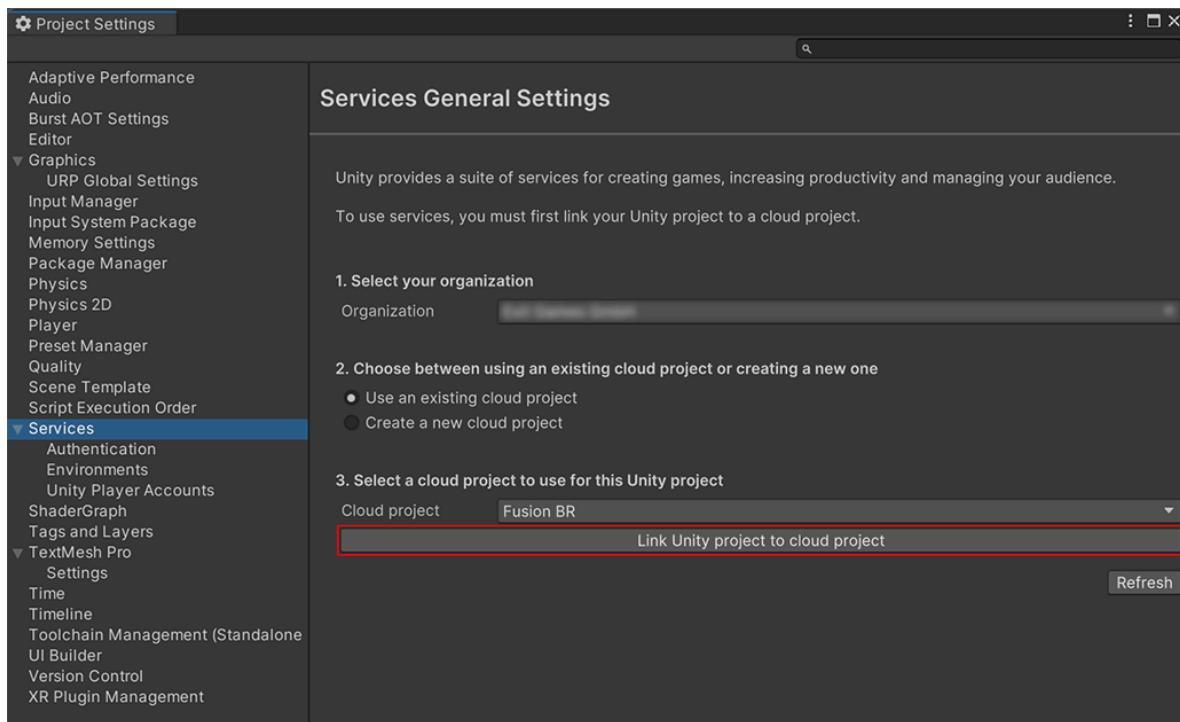
1. Select **Edit > Project Settings > Services**.
2. Select your **Organization**.
3. If you already have a Unity project, select **Use an existing cloud project**. To create a project from the Unity Editor, then **Create a new cloud project**.

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

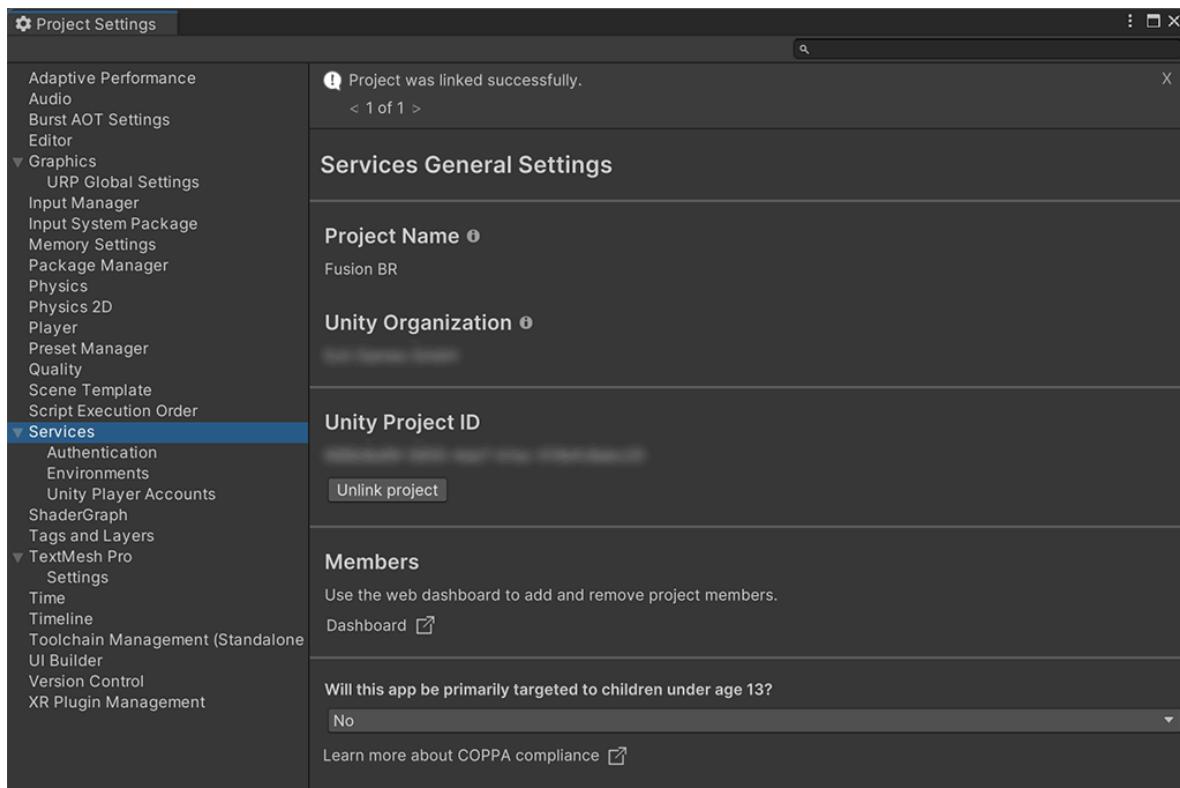




4. Select cloud project and **Link Unity project to cloud project**.



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

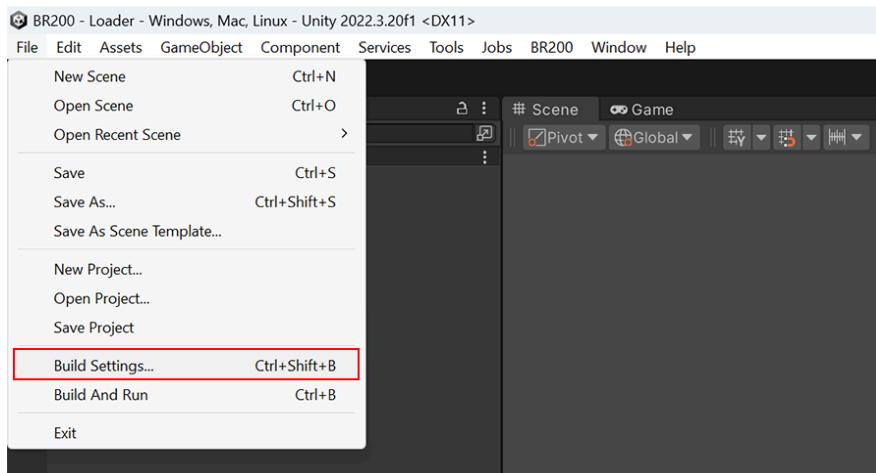




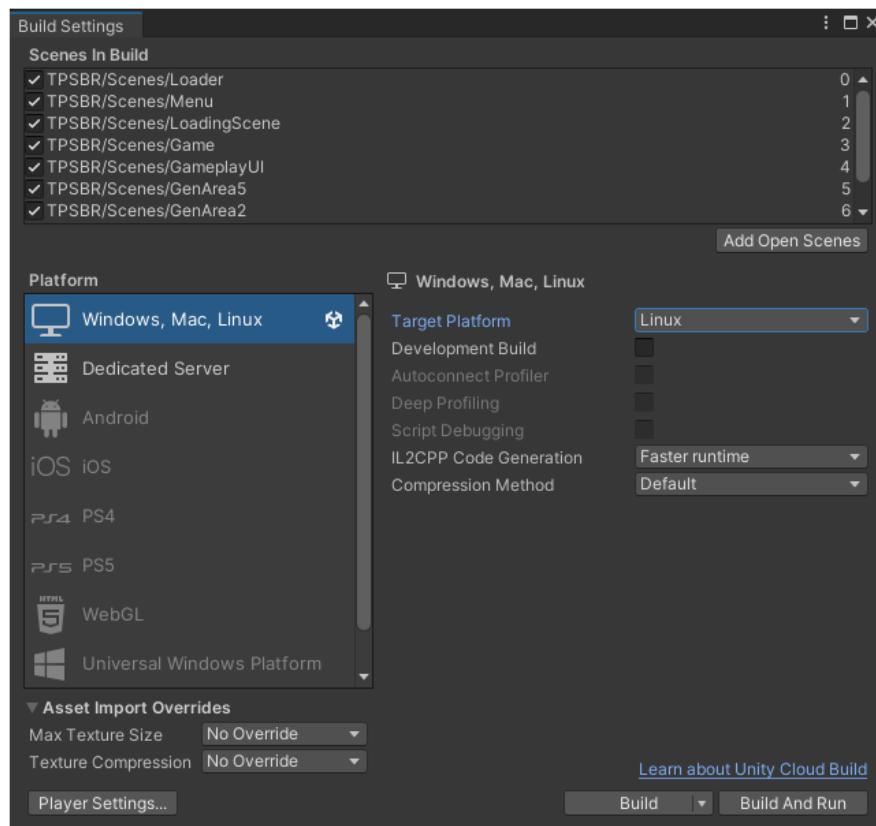
Build the standalone server

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

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

2 Create a build

Integrate game server

Create a build

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

Integrate game server

1 Select engine 2 Link Unity Project 3 Install Project

Select engine

Link Unity Project

Install Project

To begin the integration, select your engine:

Unity

Unreal

Custom

Cancel

Next

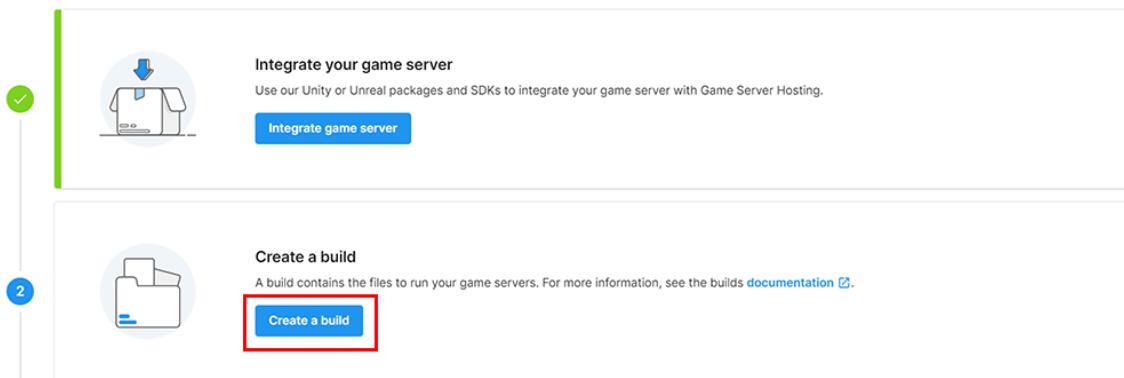
3. Select **Next** if you've already linked your Unity project with the Unity Editor.
4. Select **Finish**.

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**.



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

1 2 3

Details Upload files Create version

A build contains the files to run your game servers. For more information, see the [integration requirements documentation](#).

Build name * —
BR200

Operating system *

Linux
Recommended

Windows
Support coming soon

Upload method *

Direct file upload

Container image

AWS S3 bucket

[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**.

The screenshot shows the Unity Asset Store upload interface. At the top, there is a progress bar with three steps: 'Details' (green checkmark), 'Upload files' (blue circle with '2'), and 'Create version' (grey circle with '3'). Below the progress bar, a note says: 'Upload the files necessary to run your server. Do not upload a zipped archive.' In the center, there is a 'Cancel' button and a prominent blue 'Upload 53 Files' button with a red border. Below these buttons is a dashed box area with an upward arrow icon and the text 'Drag file(s) here or [browse](#)'. At the bottom, there is a search bar with the placeholder 'Search files'. A table lists five files with their names, status as 'Ready to upload', and a trash bin icon for deletion. The table includes columns for 'Name' and 'Status'. The files listed are: BR200_Data/app.info, BR200_Data/boot.config, BR200.x86_64, and GameAssembly.so. Below the table, there is a pagination control with 'Rows per page: 10', '1-10 of 53', and navigation arrows. At the very bottom, there are 'Cancel' and 'Next' buttons.

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



6. Select **Next**.

1 2 3

Details Upload files 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.

1 2 3

Details Upload files 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

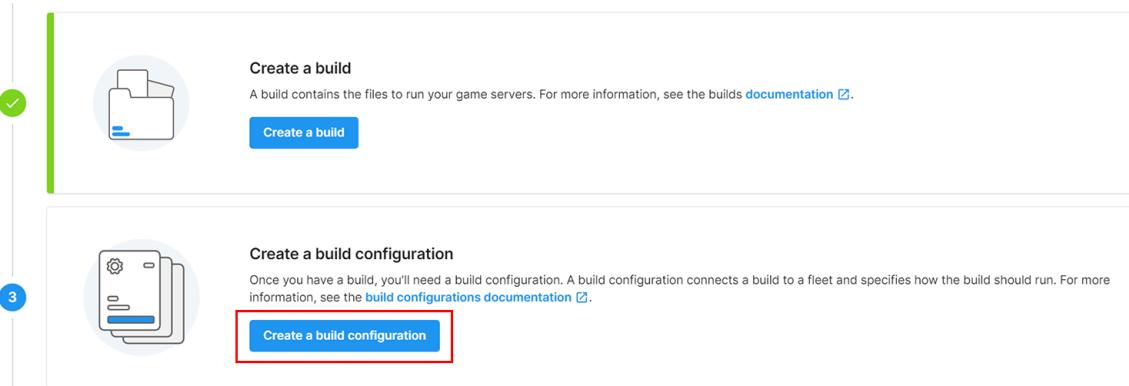
[Cancel](#) [Finish](#)

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 -sqp -matchmaking -maxPlayers 200
```



3. Select **Next**.

1 2

Details Configuration variables

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 * _____
BR200

Build * _____
BR200 (Direct File Upload) ▼

Select a build to assign to this build configuration.

Game server executable _____
BR200.x86_64

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

Query type (i)

SQP
Supported by the
Game Server
Hosting SDK

A2S
Supported by the
Steam SDK from
Valve

None
If your server has
no support for
querying metrics

Server readiness (i)

Enable server readiness for this build configuration.

Launch parameters (i)

```
-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 **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 ⓘ

```
{ "region": "us-central1", "os": "Ubuntu 18.04 LTS" }
```

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**.

1

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

4

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:

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



3. Select **Next**.

Fleet name *
BR200 Fleet

Operating system *

Linux
Recommended

Windows
Support coming soon

Build configuration(s) ⓘ *
1 build configuration selected

Cancel Next

4. Specify the scaling settings:

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

Region *
Europe

Select a region to host your game or application. You can add more regions later.

Min available servers ⓘ *
0

Max servers ⓘ *
2

Servers will be offline until allocated.

Cancel Back Next

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.



5. Set server allowances and select **Finish**.

Machine specification ⓘ GCP-N2 @ 2.8 GHz | 8 GB RAM

Servers per machine * ⓘ 2

CPU allowance per server ⓘ 50.00%

Memory allowance per server ⓘ 4 GB

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**.

5

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

Create a test allocation

Once you have a fleet, build, and build configuration, create a test allocation to ensure everything you've created is working correctly.

Create a test allocation



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.

The screenshot shows a progress bar at the top with two steps: 'Set up' (green checkmark) and 'Run test' (blue circle with the number '2'). Below the progress bar is a blue info box containing the text: 'An allocation might take some time if a new server is being created.' Underneath the progress bar is a horizontal timeline with three stages: 'Sending allocation' (green dot), 'Waiting for server' (blue dot), and 'Server allocated successfully' (green dot). At the bottom of the screen are three buttons: 'Cancel', 'Back', and 'Finish'.

6. Select **Finish**.

The screenshot shows the summary page after a successful test allocation. It features a green success message: 'Test allocation successful.' Below this are three data fields: 'Test allocation ID' (redacted), 'Server IP:Port' (redacted), and 'Time remaining' (59m 51s). At the bottom are 'Cancel', 'Back', and 'Finish' buttons.

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

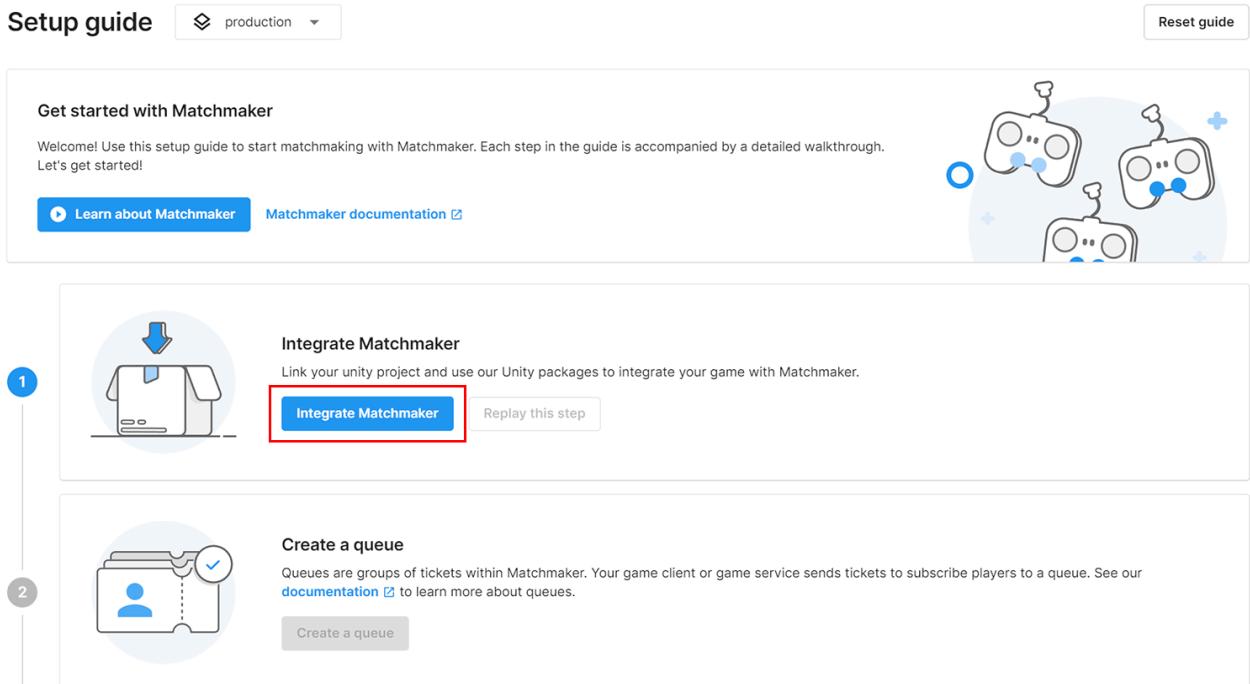
Note: You might need to enter payment information to continue the trial. If prompted, enter your payment information, then select **Complete onboarding**.

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**.



The screenshot shows the 'Setup guide' interface for Matchmaker. At the top, there's a dropdown menu set to 'production' and a 'Reset guide' button. Below that is a section titled 'Get started with Matchmaker' with a welcome message and links to 'Learn about Matchmaker' and 'Matchmaker documentation'. To the right is an illustration of three game controllers. The main area contains two steps: Step 1, 'Integrate Matchmaker', which shows a icon of a folder with a download arrow and a red box around the 'Integrate Matchmaker' button; and Step 2, 'Create a queue', which shows an icon of a folder with a checkmark. Both steps have 'Replay this step' buttons.



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

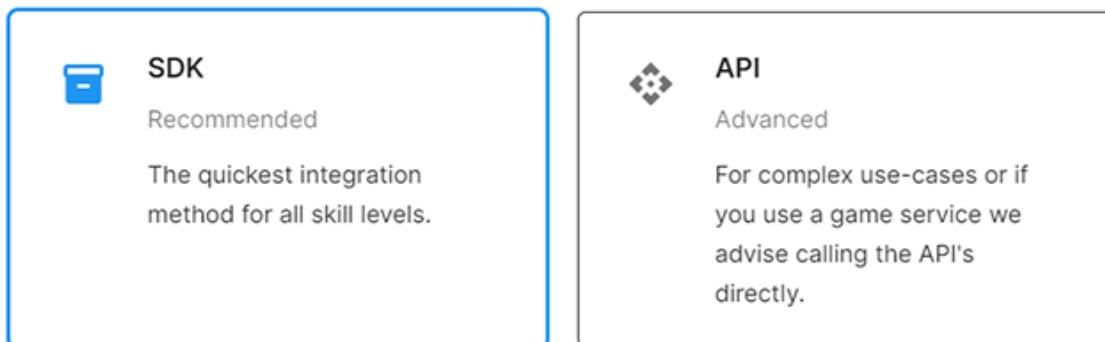


To begin Matchmaker integration please select your game engine and preferred integration method. For more information, see the [Integration documentation](#).

Game engine:



Integration method:



[Cancel](#)

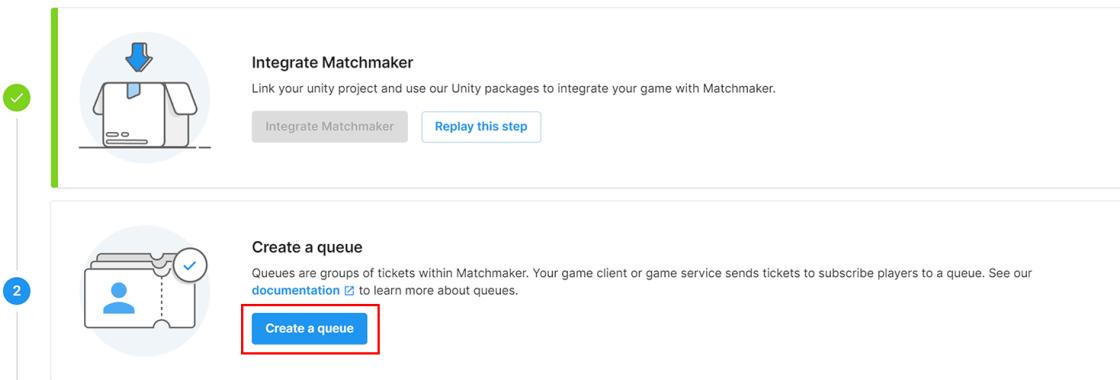
[Next](#)

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**.



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 * —

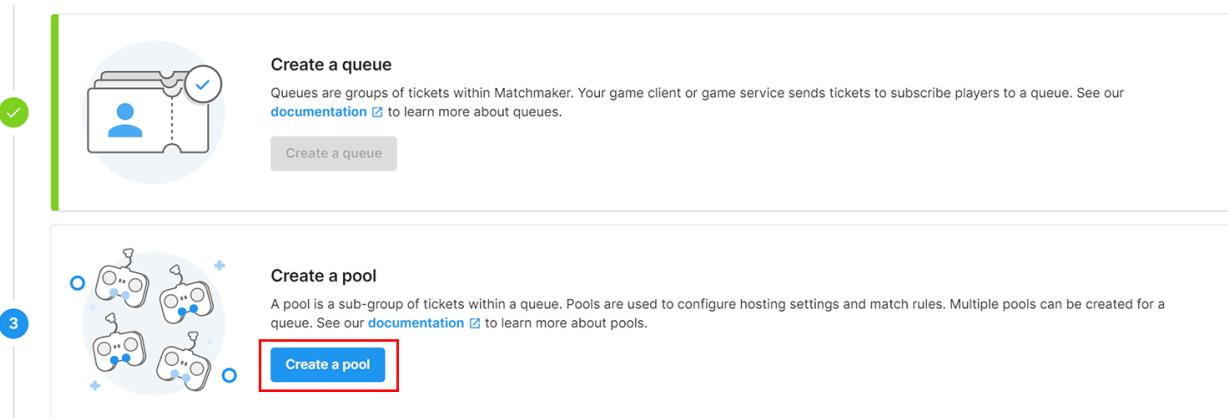
Maximum players on a ticket * —

[Cancel](#) [Create](#)

Create a default pool

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

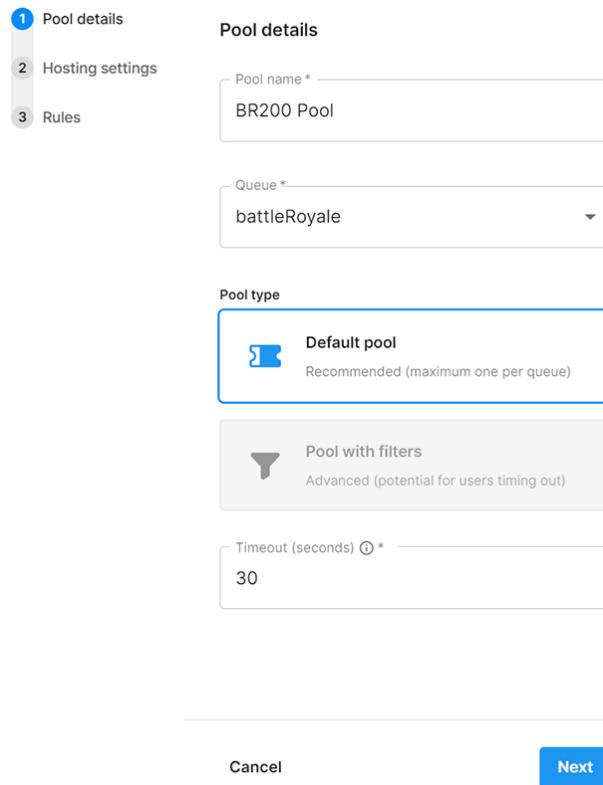
1. Select **Create a pool**.



2. Fill in the **Pool details**:

- Give the pool a name.
- Select the queue you created in previous step
- Set the timeout to **30** seconds.

3. Select **Next**.



The screenshot shows the 'Pool details' step of the wizard. The 'Pool name' field contains 'BR200 Pool'. The 'Queue' dropdown is set to 'battleRoyale'. Under 'Pool type', the 'Default pool' option is selected and highlighted with a blue box, with the note 'Recommended (maximum one per queue)'. Below it, 'Pool with filters' is listed as an alternative. The 'Timeout (seconds)' field is set to '30'. At the bottom, there are 'Cancel' and 'Next' buttons.

1 Pool details
2 Hosting settings
3 Rules

Pool details

Pool name * BR200 Pool

Queue * battleRoyale

Pool type

Default pool
Recommended (maximum one per queue)

Pool with filters
Advanced (potential for users timing out)

Timeout (seconds) ⓘ * 30

Cancel Next



4. Fill in the **Hosting settings**:

- Select the [fleet you created earlier](#).
- Select the [build configuration you created earlier](#).
- 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**.

Pool details

Hosting settings

Hosting type

Game Server Hosting (Multiplay)

Scalable dedicated server hosting for performance and flexibility.

Configuration

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.

Fleet

BR200 Fleet

Build configuration

BR200

Default QoS Region ⓘ

Europe

Cancel Back Next

6. Configure the **Rules**:

- Set the Match definition name to **Battleroyale Match Definition**.
- Set **Backfill enabled** to **True**.
- Finish configuring the remaining rule settings
 - Set **Min teams** to **1**.
 - Set **Max teams** to **1**.
 - Set **Min players** to **1**.
 - 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 *

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**.

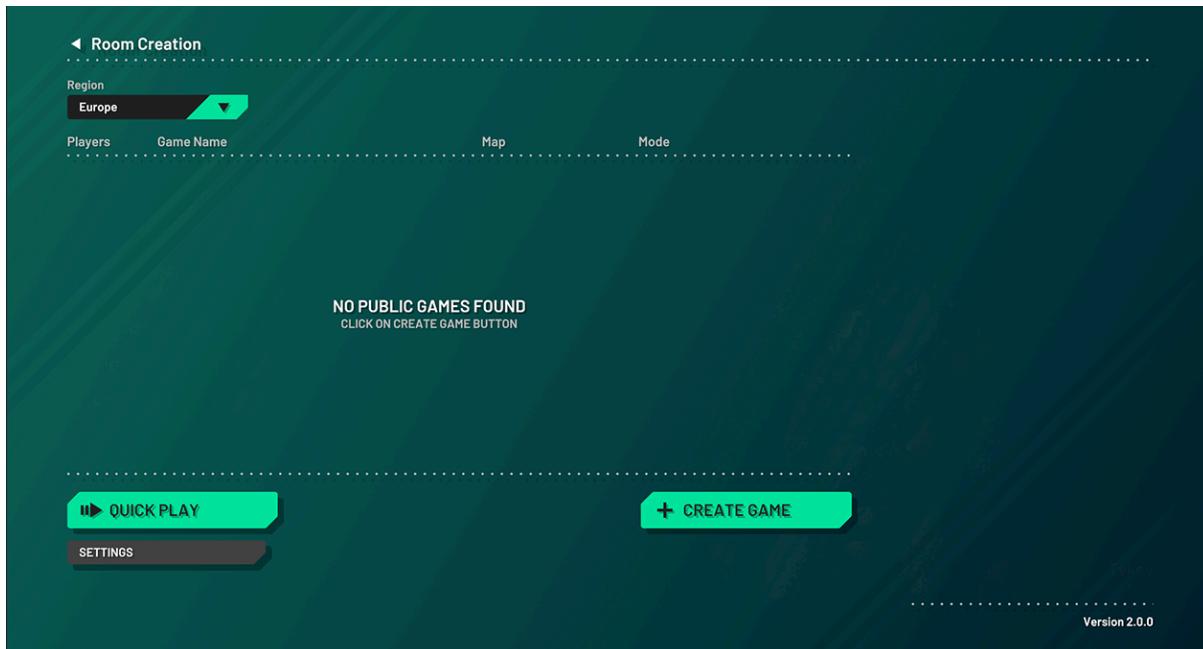
Congratulations! You've successfully configured the Unity Matchmaker. You can go to **Products > Matchmaker > Overview** to view matchmaking traffic and match times.

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.

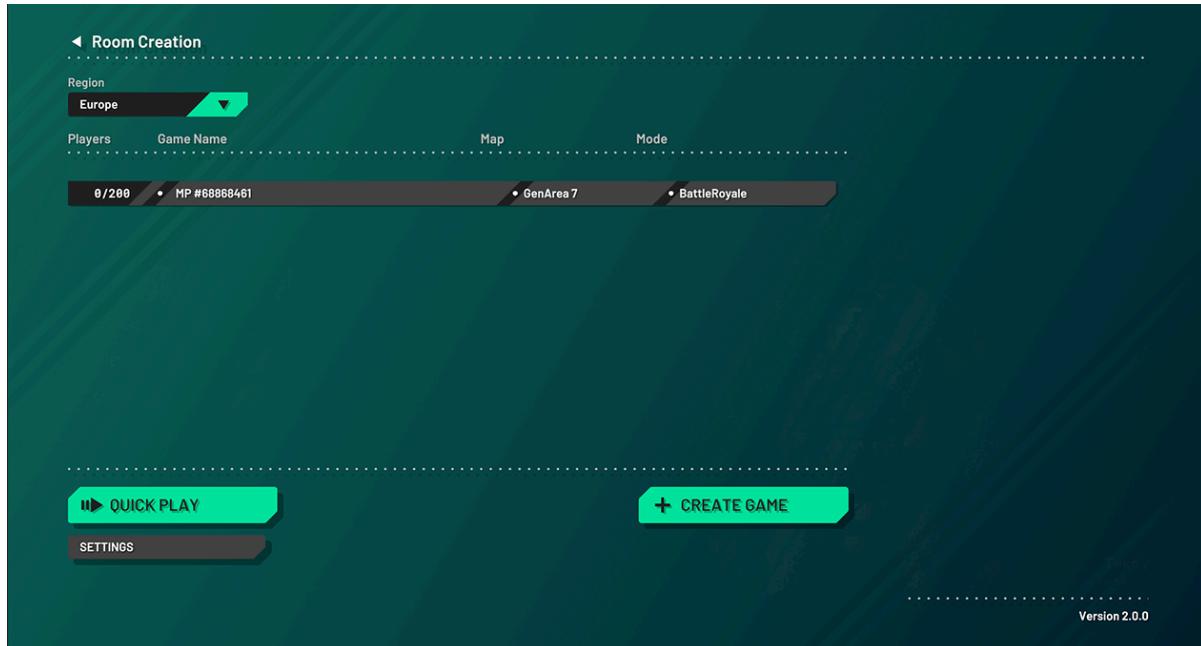


After launching, the game client shows a game session list based on the available game sessions on Unity Game Server Hosting. If this is the first time you're running the application and haven't already started any sessions, you won't see any game sessions available yet.

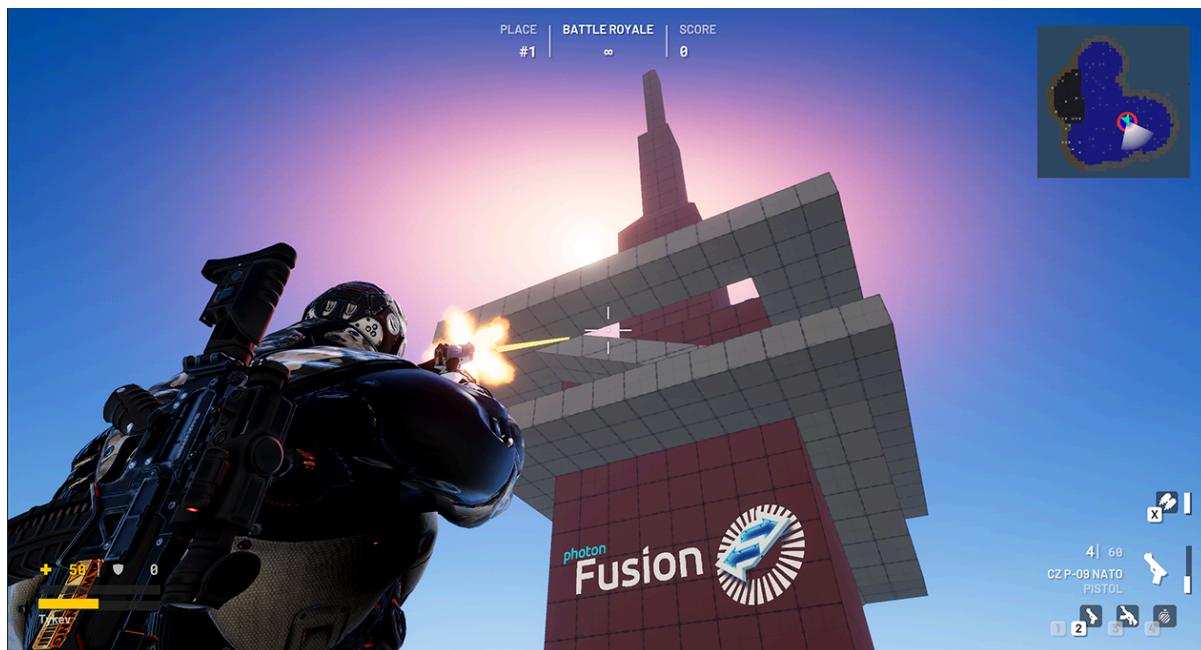


Tip: You can go back to the Game Server Hosting and Matchmaker dashboards to view game performance metrics.

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