UNITY WEBGL MULTIPLAYER ROOMS TEMPLATE DOCUMENTATION

============= HOW TO SETUP =============

It is important to understand that we have to first have a specific setup so that this project can work. Consider this project split into 3 different sections.

1 – The Unity project itself and the scenes with all the prefabs and scripts. This includes other assets such as music and sprites. Also, related to this section we will have a Github repository and a WebGL Template with a JavaScript client side code to receive broadcasts from the server.

2 – The server-side JavaScript with all the server functions that will manage all players online and manage their rooms. Also, this section includes installing and running Nodejs. This also will have a Github repository to deploy into your WebService of choice. (AWS, Azure, Heroku)

3 – This project can work in localhost and can support as many players locally. But for the best experience, we can use a web service to host the game so that players from around the world can play together. There are many free services to host your game and this project includes a step by step on how to set it up on Heroku.

FIRST STEP

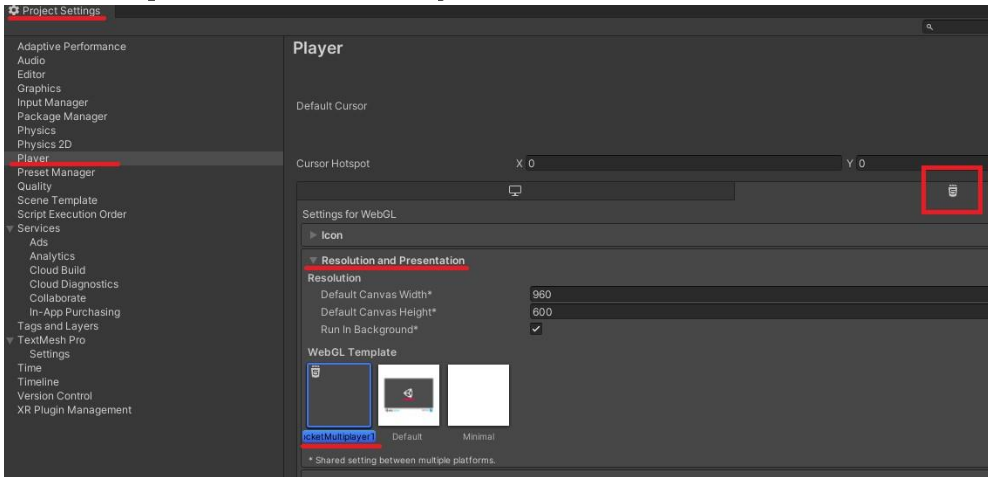
In any WebGL Unity project, import the WebGLMultiplayerRoomsTemplate from the Asset Store.

Then go to this link: https://danielboldrin.azurewebsites.net/Assets
And download the <u>WebGL Multiplayer Rooms Template Extra Content</u>

This **zip**/**rar**/**7z** file will include the server and the WebGLTemplates to be added into the assets folder of your Unity project. So…

Extract the extra content pack, move both folders “Server” and “WebGLTemplates” to the folder “Assets” of your Unity project. It **HAS** to be in your assets folder, otherwise Unity won’t load the templates.

You can check if Unity imported everything correctly by going to Project Settings -> Player Settings -> Resolution and Presentation.
Check if you can see an extra option in your WebGL Templates. Make sure you are in WebGL platform. See image below:



Default Cursor
Cursor Hotspot X 0 Y 0
Resolution
Default Canvas Width* 960
Default Canvas Height* 600
Run In Background* ✓
WebGL Template
cketMultiplayer1 Default Minimal
* Shared setting between multiple platforms.

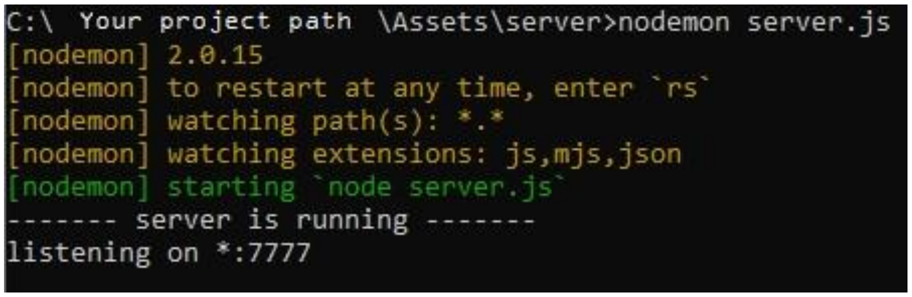
SECOND STEP

You are ready to test the game locally and see if it works. If you want to deploy it into a web service, you can move to the next step.

Download Node.JS -> Official website: https://nodejs.org/en/download/
You don't need to install any node packages because the extra content already contains the required ones in it.

Then, go to your CMD and use the command:
cd **Your Unity Project Folder Path**/Assets/Server/
after, use this command:
nodemon server.js
or, if you have problems using nodemon, try just:
server.js

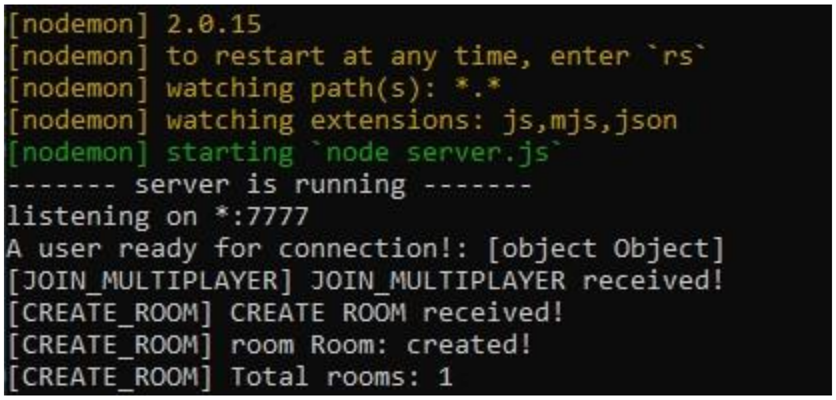
This will start the server in your machine.



```
C:\  Your project path  \Assets\server>nodemon server.js
[nodemon] 2.0.15
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
------- server is running -------
listening on *:7777
```

To test if the connections will work, you can go to Unity and select build and choose the folder public inside of server. **(Make sure the main menu and multiplayer scenes are added to the build, in this order)**

Once the game loads check your server running and you will see a log notifying that a player is ready for a connection.
On the Unity game that will start in your browser, you can join the multiplayer and create rooms.
To test multiple players, you can open different tabs in your browser and visit the same link: http://localhost:7777/ (Or any other port you set)



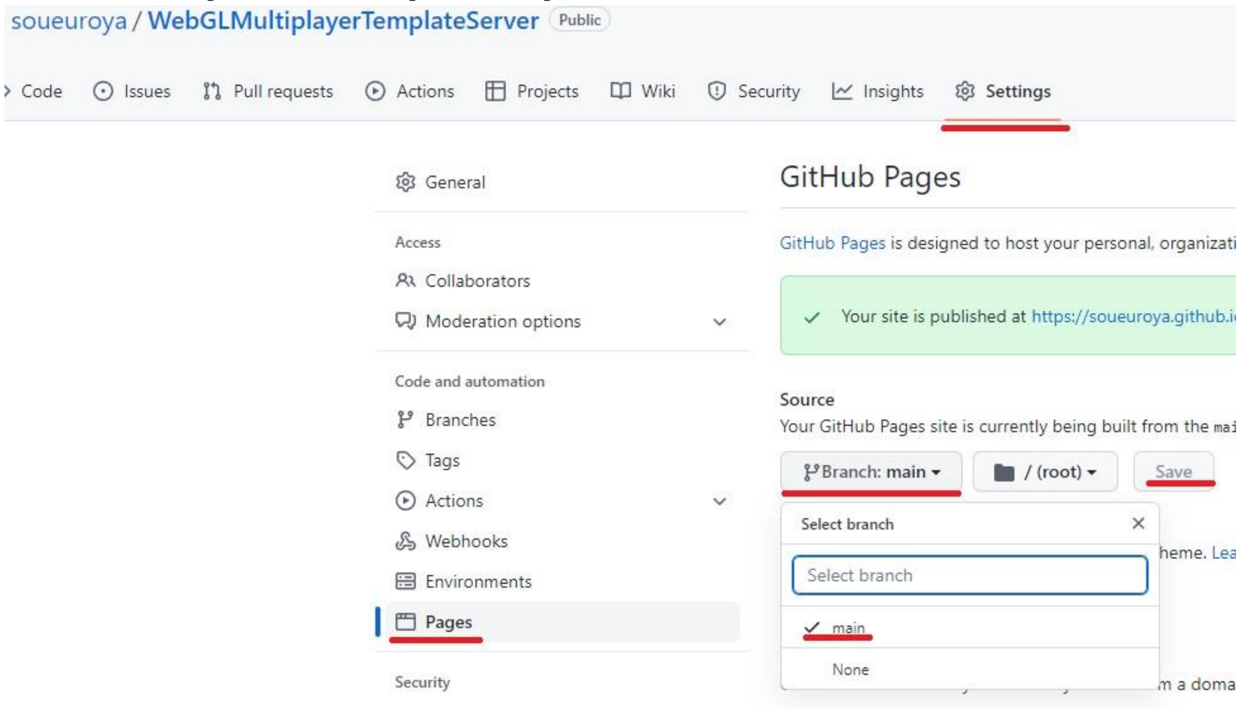
```
[nodemon] 2.0.15
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
------- server is running -------
listening on *:7777
A user ready for connection!: [object Object]
[JOIN_MULTIPLAYER] JOIN_MULTIPLAYER received!
[CREATE_ROOM] CREATE ROOM received!
[CREATE_ROOM] room Room: created!
[CREATE_ROOM] Total rooms: 1
```

The server has some default logs included so you can check in the console. But you can also check the console and logs from the Unity game in each browser! Just simply access the development settings in the page with the game running and the console should also print useful information.

THIRD STEP

To test your project in a web service you just need to create a Github repository with your WebGL server. Here is my repository as an example:
https://github.com/soueuroya/WebGLMultiplayerTemplateServer

Make sure to go to Settings -> Pages -> select the main branch and save.



GitHub Pages is designed to host your personal, organizati

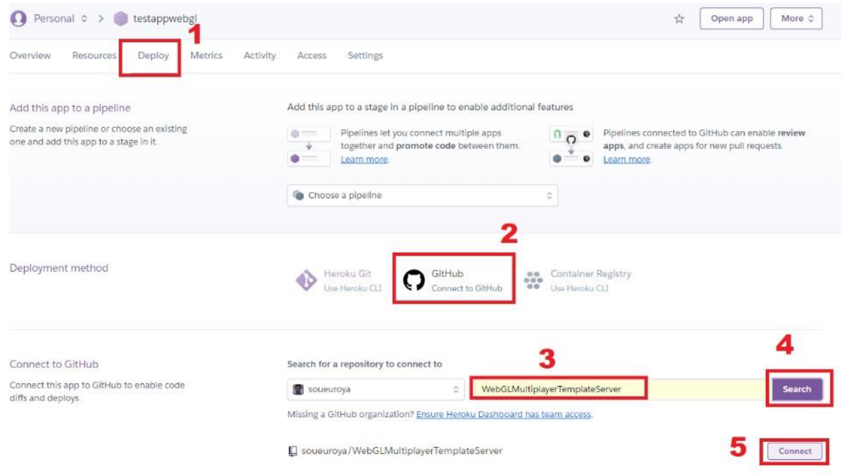
Your site is published at https://soueuroya.github.i

Source
Your GitHub Pages site is currently being built from the mai

After pushing your changes to Github, you need to setup your web service to run the server.

I’ll add how to add this into different web services but for now here is how to set it up with **Heroku**:

Create a free account at: https://dashboard.heroku.com/
After all security checks and confirming your email, select **Create New App** – Add a name and select create app.



Add this app to a pipeline

Create a new pipeline or choose an existing one and add this app to a stage in it.

Add this app to a stage in a pipeline to enable additional features

Pipelines let you connect multiple apps together and **promote code** between them. Learn more.

Pipelines connected to GitHub can enable **review apps**, and create apps for new pull requests. Learn more.

Deployment method

Connect to GitHub

Connect this app to GitHub to enable code diffs and deploys.

Search for a repository to connect to

Missing a GitHub organization? Ensure Heroku Dashboard has team access.

soueuroya/WebGLMultiplayerTemplateServer

Follow the steps above and click in Deploy Branch to update your server app. You will be able to visit the website and the game will be connected to the server, so all player can play together! That’s it!

============= END OF HOW TO SETUP =============

============= CODE HIGHLIGHTS =============

The most important script is the MultiplayerManager.cs which handles receiving and sending sockets to the server.

The server listens for all players and sends a message to the client. Then the client calls the C# script in the game with the necessary data.

This code prioritizes performance and game smoothness by only sending sockets on velocity change and preserving speed unless it detects changes.

This code also optimizes external calls by caching frequently used variables such as Transform and Rigidbody.