EdgeGap FishNet Sample Guide

Description:

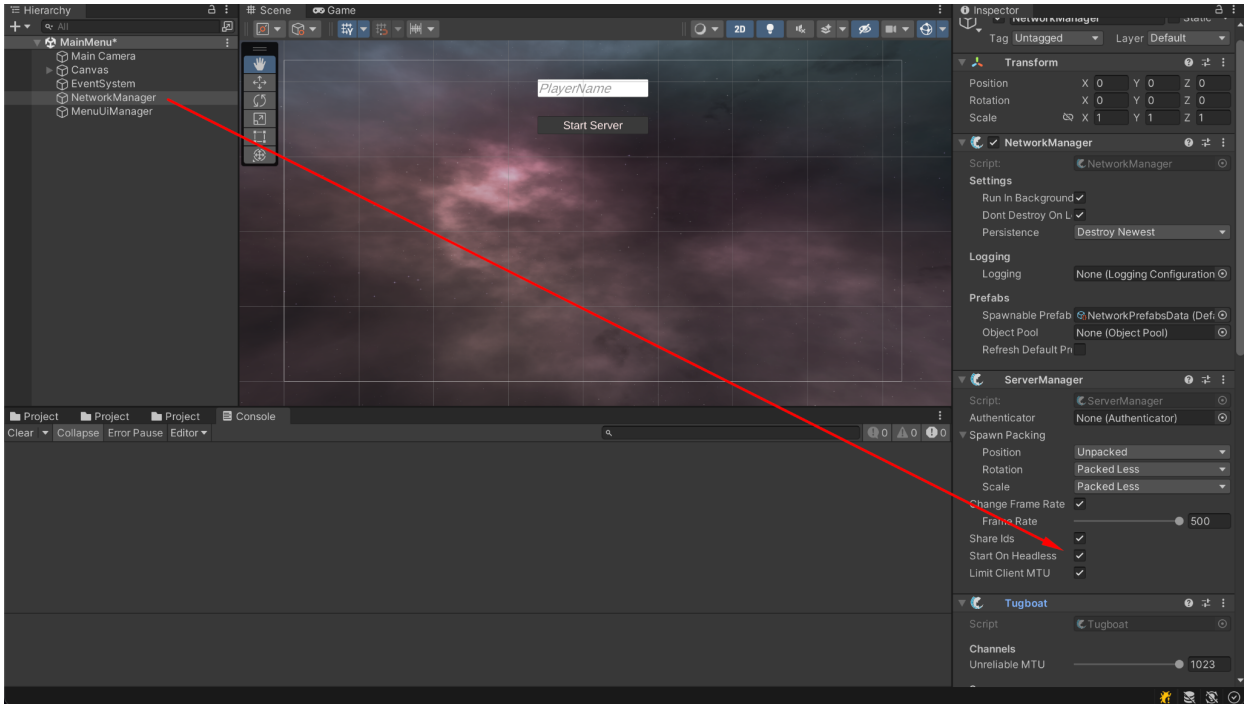
This sample is built to demonstrate EdgeGap running a fully server authoritative game made using FirstGearGames/FishNet - Unity Networking Evolved. networking framework for unity.
The core of this sample is the “MatchmakingSystem” script. This script communicates with the EdgeGap API to find live deployments of the available server for the game and if no available servers are found then it requests the EdgeGap API to deploy a new instance of the game’s server for the client based on their location. This script explains the core concepts of communicating with the EdgeGap API using the Unity HTTP Requests system and utilizing the data sent by the EdgeGap API for making decisions for the game’s matchmaking.
This game auto connects to the first instance found on the list of all the live server deployments for the game, this is to make testing easier and due to the fact that EdgeGap free tier allows only 1 live server deployment for the game at a given time. But if a user is not using free tier then this matchmaking system can be really easily extended to give players an option to choose from all the available servers to connect to.

This guide will explain step by step how to,
1- Prepare a server build
2- Containerize it and push it to a repo
3- Setup EdgeGap Application
4- Configure the client build

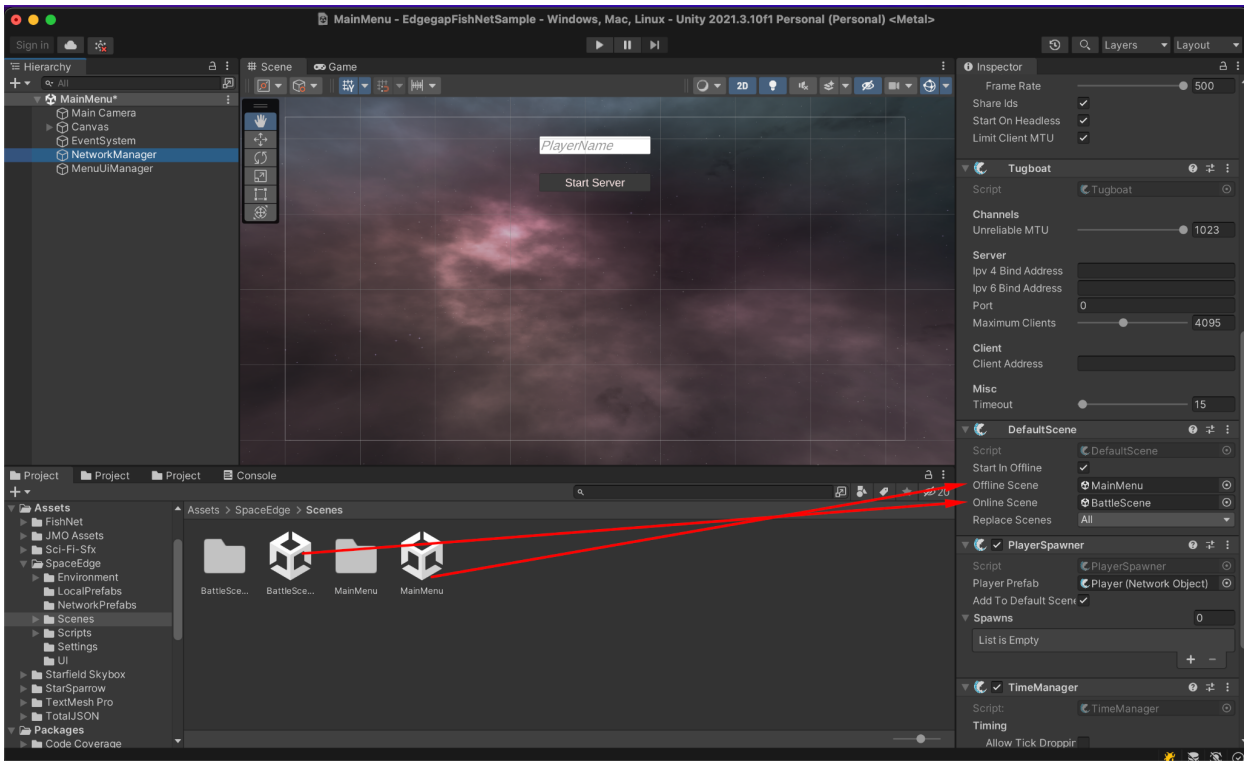
Tip: This project uses a bunch of free assets from the Unity Asset Store. At the end of this document there are links to all these assets.
This Project is tested on Unity Version 2021 LTS.

Preparing The Server Build:

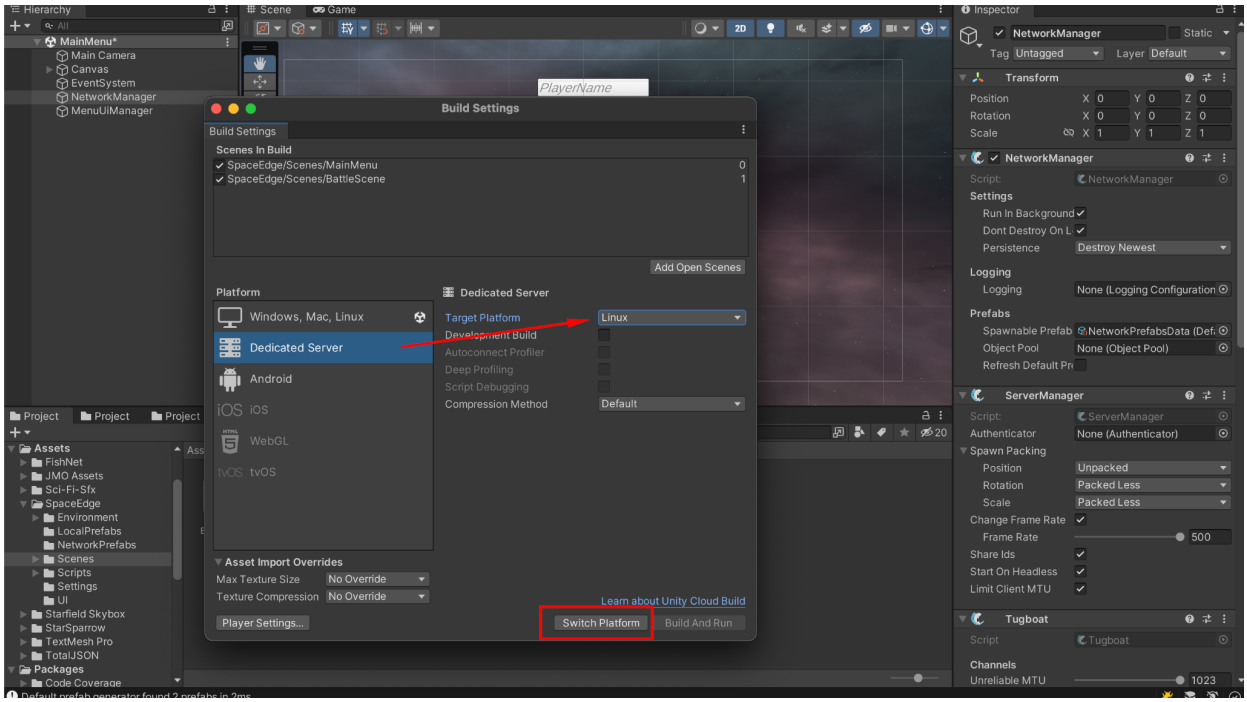
Make sure that “Start On Headless” is checked on the ServerManager component of FishNet. This ensures that the network manager auto start server when the build mode is set to “Headless”



Check that the DefaultScenes component have Offline and Online scene set up properly, so the SceneManager can auto load the proper scene based on the Network Mode (Server or Client)

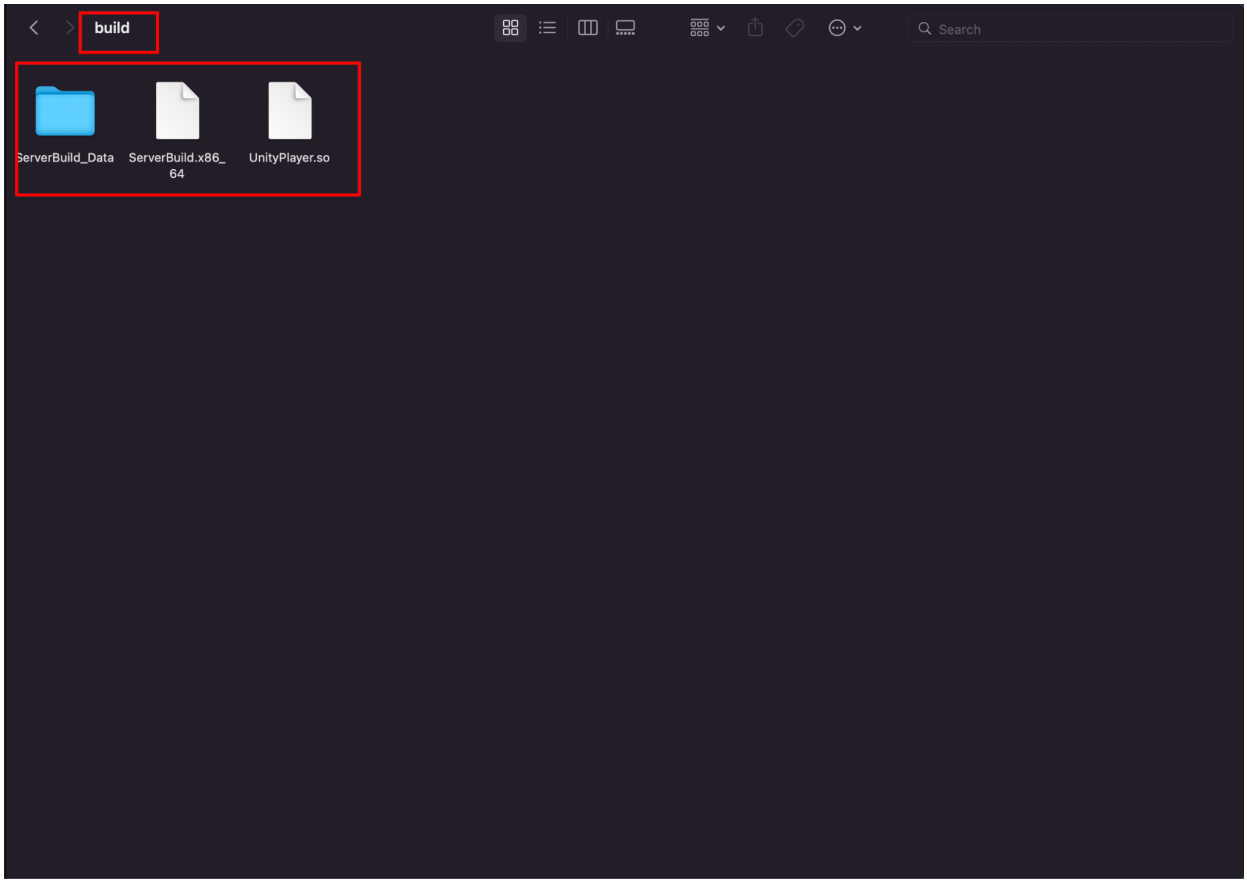


Switch the build type to “Dedicated Server” and the target platform to linux.



Tip: If you are building the game with “IL2CPP” scripting backend then you would need to install sysroot toolchain from here - [Unity IL2CPP Build Support for Linux | Sysroot Base | 2.0.3]

Click on the “Build” button, when prompted, enter the name “ServerBuild” as the build name. Once the build is complete you will have 2 (ServerBuild.x86_64 and UnityPlayer.so) and 1 folder (ServerBuild_Data). Move both the files and the folder inside a new folder and name the new folder as “build”.



Create a new folder named “DockerBuild” and move the “build” folder inside it. Create a new file inside the “DockerBuild” and name this file as “Dockerfile” and make sure the file has no extension, open the file using a text editor that allows changing Encoding Type and Line Ending of the file. BBEdit for Mac, and Notepad ++ for Windows are the best text editors for this job and they are free as well.
Once the “Dockerfile” is created, open it using the text editor and paste the below text in it-

```
FROM ubuntu
MAINTAINER Edgegap <youremail@edgegap.com>

ARG DEBIAN_FRONTEND=noninteractive
ARG DOCKER_VERSION=17.06.0-ce

RUN apt-get update && \
apt-get install -y libglu1 xvfb libxcursor1

EXPOSE 7770/udp

COPY build/ /root/build/
COPY boot.sh /boot.sh

WORKDIR /root/
ENTRYPOINT ["/bin/bash", "/boot.sh"]
```

Take a note of the line #10 “EXPOSE 7770/udp” this is the port number and protocol type we are going to configure in the EdgeGap console later on. Make sure the file Encoding is set to UTF-8 and the Line Ending is set to Line Feed (LF). Now save the file without any extension (.txt, .rtf, etc…)

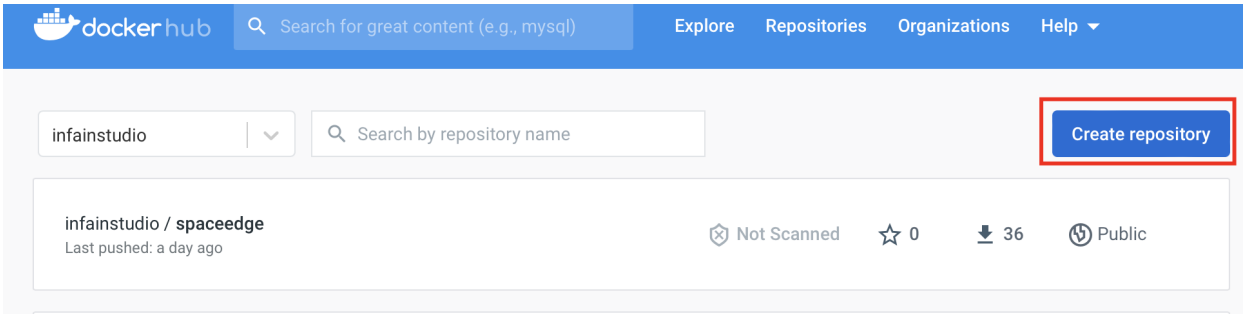
Now create one more file in the “DockerBuild” folder and name it “boot.sh”. Open using the text editor and paste the below text in it

```
#!/bin/bash
chmod +x /root/build/ServerBuild.x86_64
xvfb-run /root/build/ServerBuild.x86_64
```

Again make sure the Line Endings and Encoding is correct (LF and UTF-8 respectively). Save the file with the extension .sh.

Building The Container Image:

Create an account at Docker and then sign in to Docker Hub using your docker credentials.
Create a new repository using the new repository button on the top right hand

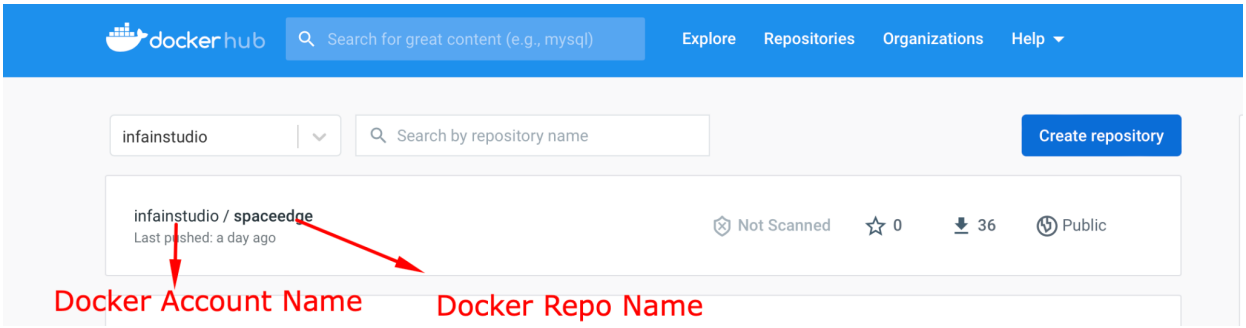


Give the repository your desired name and a description (optional) then click the “Create” button, now you have your own docker repo.
Now install the Docker Desktop and open it, login using your Docker credentials.

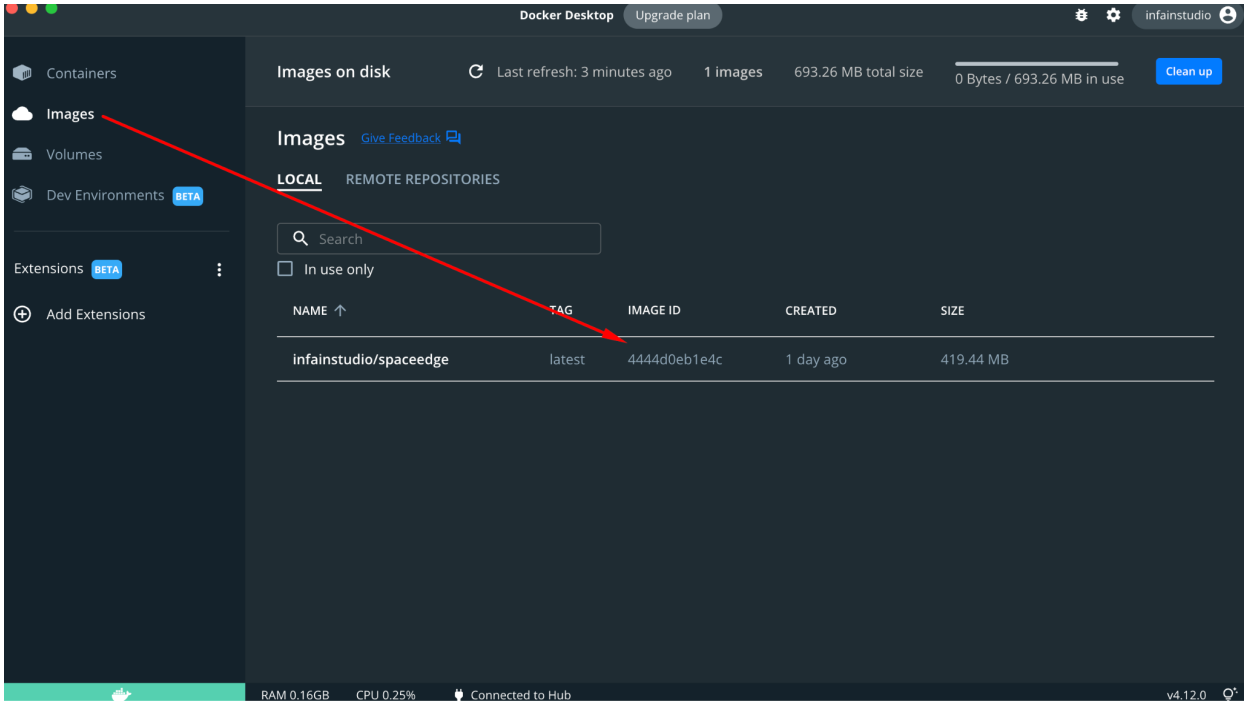
Open a new terminal window on your system and change the working directory to the “DockerBuild” folder that we created earlier.
Tip: If you don’t know how to use a terminal or what a terminal is, please refer to these guides- For Mac , For Windows , and For Linux.

Once your working directory is set to the “DockerBuild” folder, use the below command to build a new “Container Image” with your server build

docker build --platform=linux/amd64 -t [Your Docker Account Name]/[Your Docker Repo Name]



Once the build is done you should see this image popup into your Docker Desktop app’s “images” section.



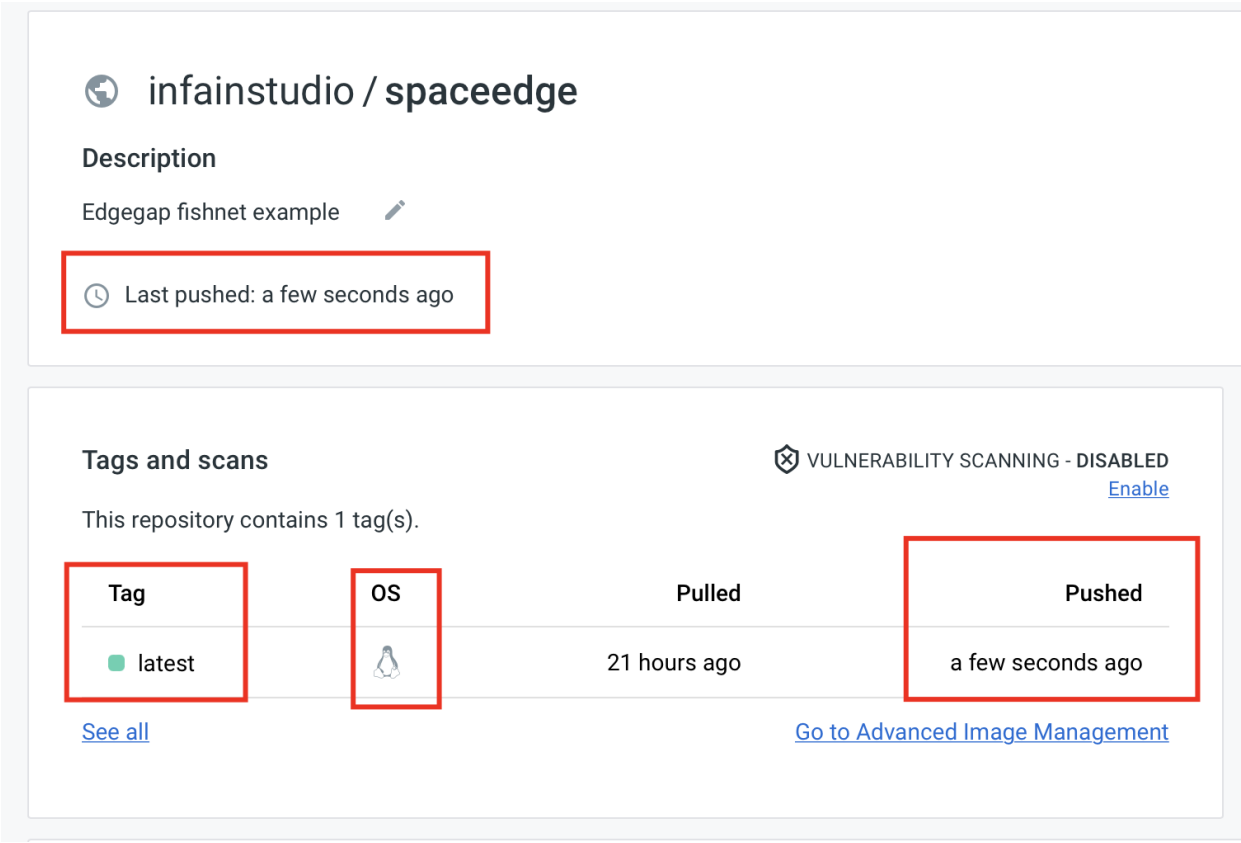
Copy the IMAGE ID from here and run this command in your terminal (Make sure the working directory is still “DockerFolder”)

docker tag [Your Image ID] [Your Docker Account Name]/[Your Docker Repo Name]:latest

Now your Docker Image is ready to be pushed to the docker hub, that can be calling this command from the terminal (Make sure the working directory is still “DockerFolder”)

docker push [Your Docker Account Name]/[Your Docker Repo Name]

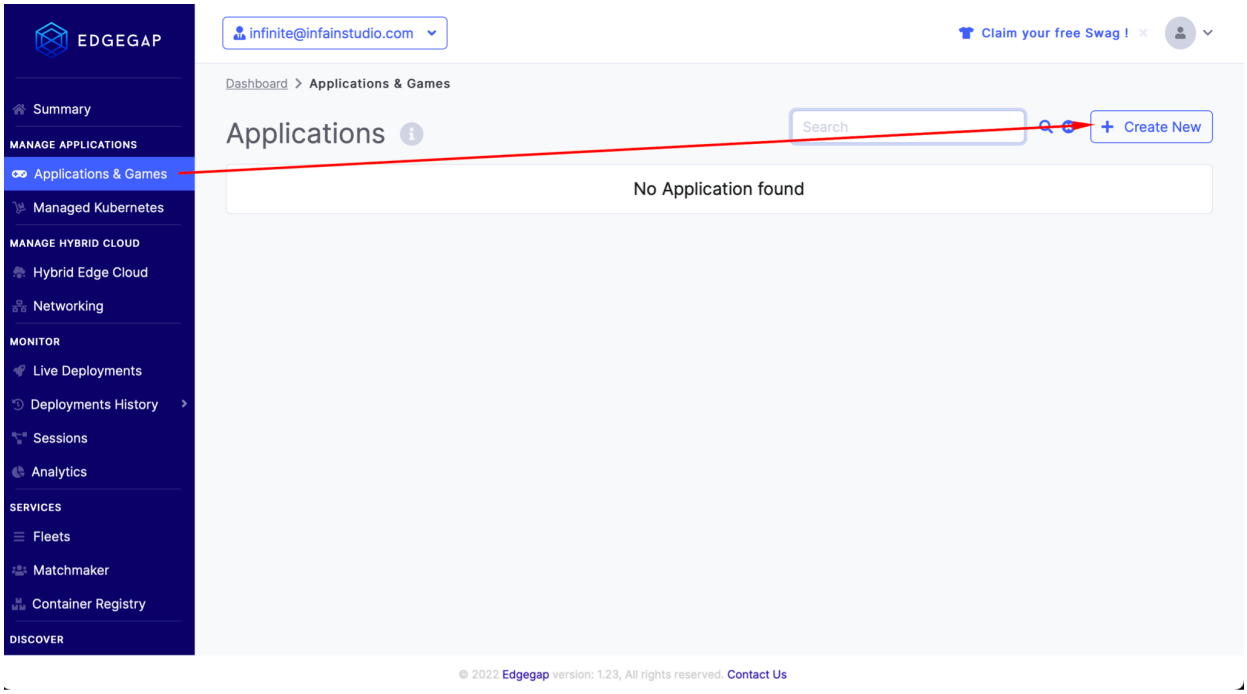
Now you should see the image pushed on your Docker Hub page with “tag” set as “latest” and OS as Linux.



Setup The EdgeGap Account:

For instructions on how to sign up for EdgeGap please refer to- [1 - Create Your Studio Profile | Edgegap's Documentation]

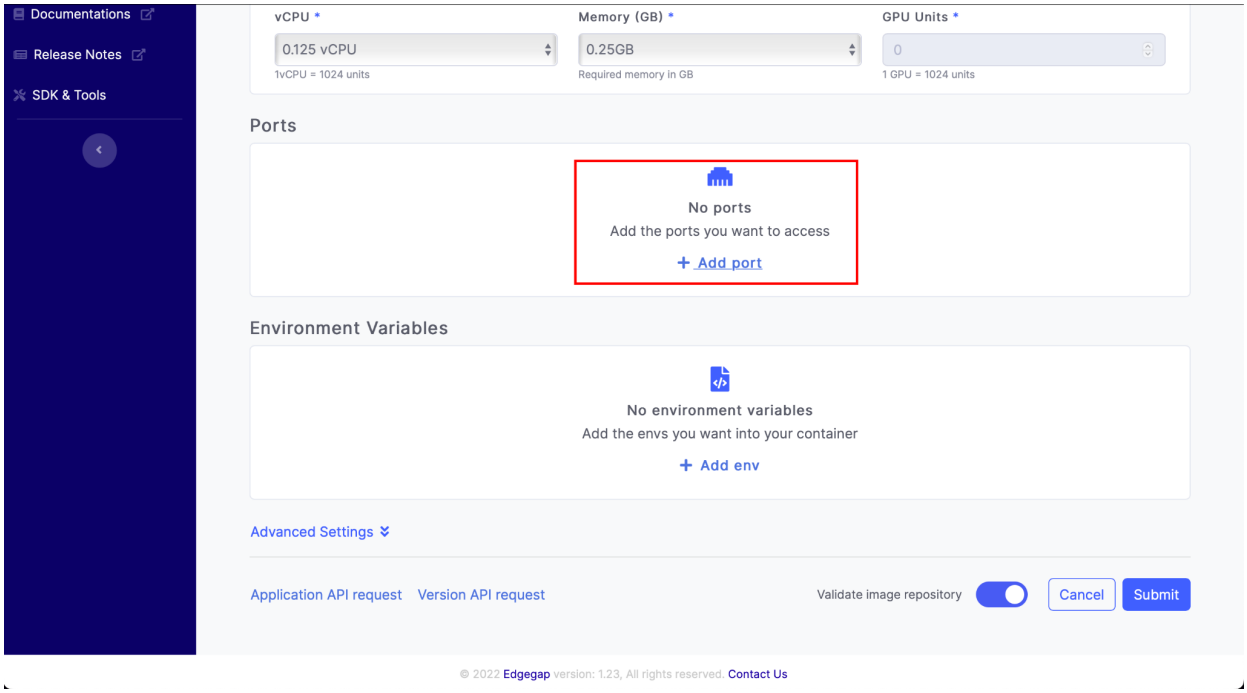
Sign in to https://console.edgegap.com/arbitrium/ and go to the “Applications And Games” tab and click one the “Create New” button on the top right corner.



Now fill in all the fields, you can use the below examples as a reference, for more details on creating an app please refer to [2 - Create your App or Game Profile | Edgegap's Documentation]

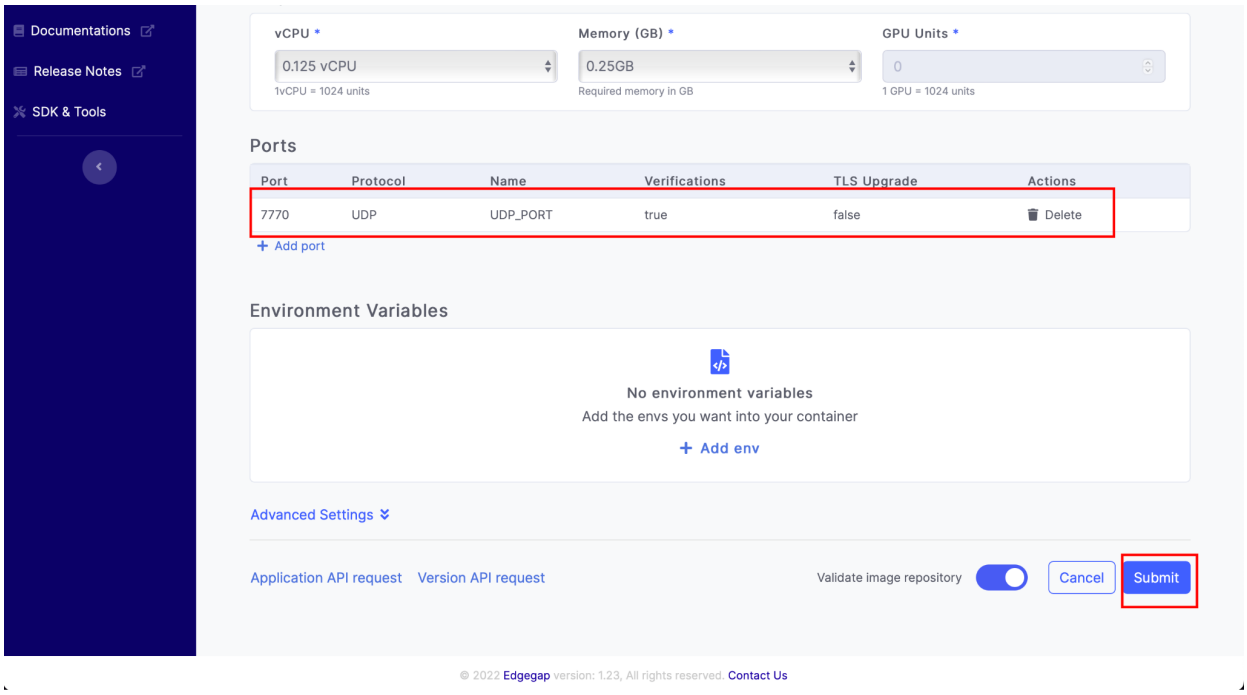
Application name: “TestApp”
Version name: “v1”
Registry: docker.io
Image repository: [Your Docker Account Name]/[Your Docker Repo Name]
Tag: latest (You can ignore the latest tag not recommended warning as it’s just a test app)
For now you can leave the Requirements to the default.
Tip: Note down the “Application name” and “Version name” as they will be used later while configuring the client build. You can always come back to the EdgeGap console and check these fields in case you forget.

Now click on the “Add port” button



Now for the port number itself, use 7770 (remember this is the port we exposed in our dockerfile using the “EXPOSE 7770/udp” command)
For the port protocol use UDP (The default FishNet transport “tugboat” uses only UDP so select only UDP and not TCP/UDP)
And for the port name use “UDP_PORT”
It’s crucial that you leave the “Verification” toggle on for our “MatchmakingSystem” to work properly.
Once all done just click the “Add” button. Should now see your port in the “Ports” list

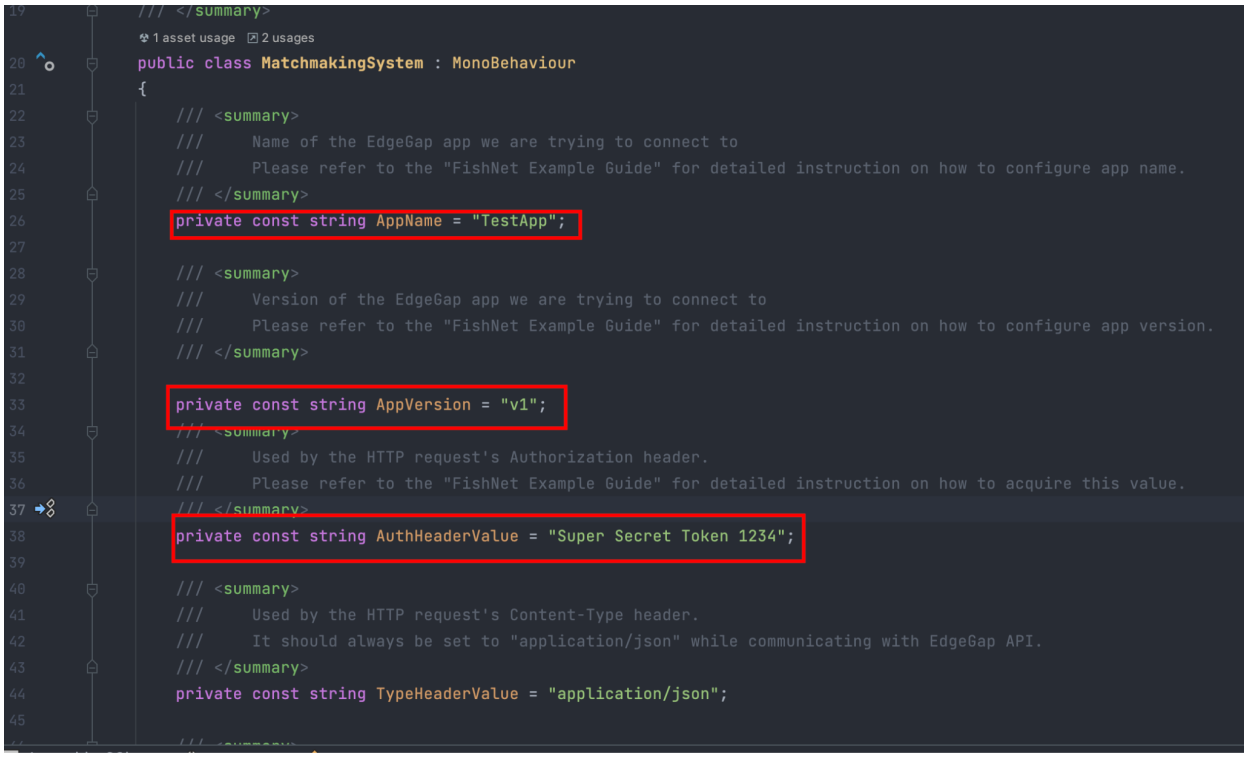
Now you can just submit your application by clicking on the “Submit” button on the bottom of the page.



Congratulations! Now your containerised app is successfully deployed on the EdgeGap.

Configure The Client Build:

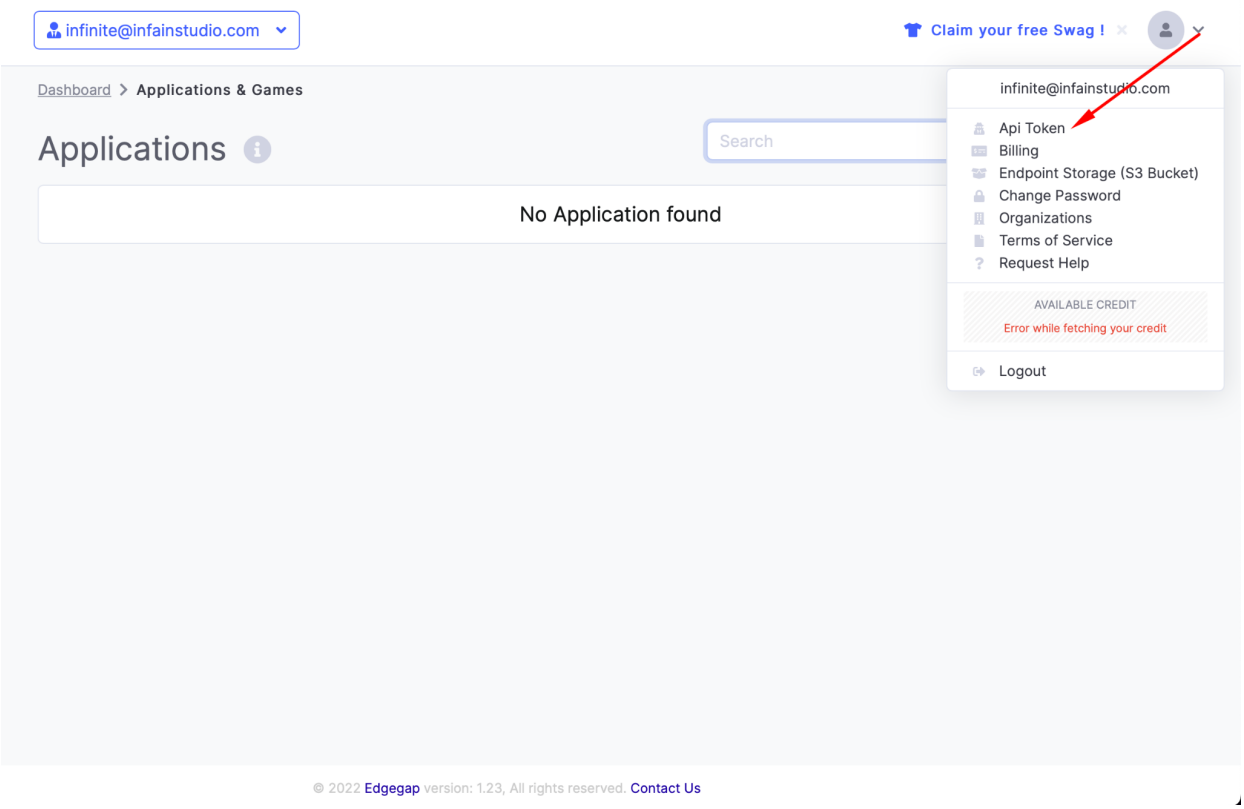
In unity project navigate to Assets>SpaceEdge>Scripts>Systems
Open the MatchmakingSystem.cs script and the following fields-



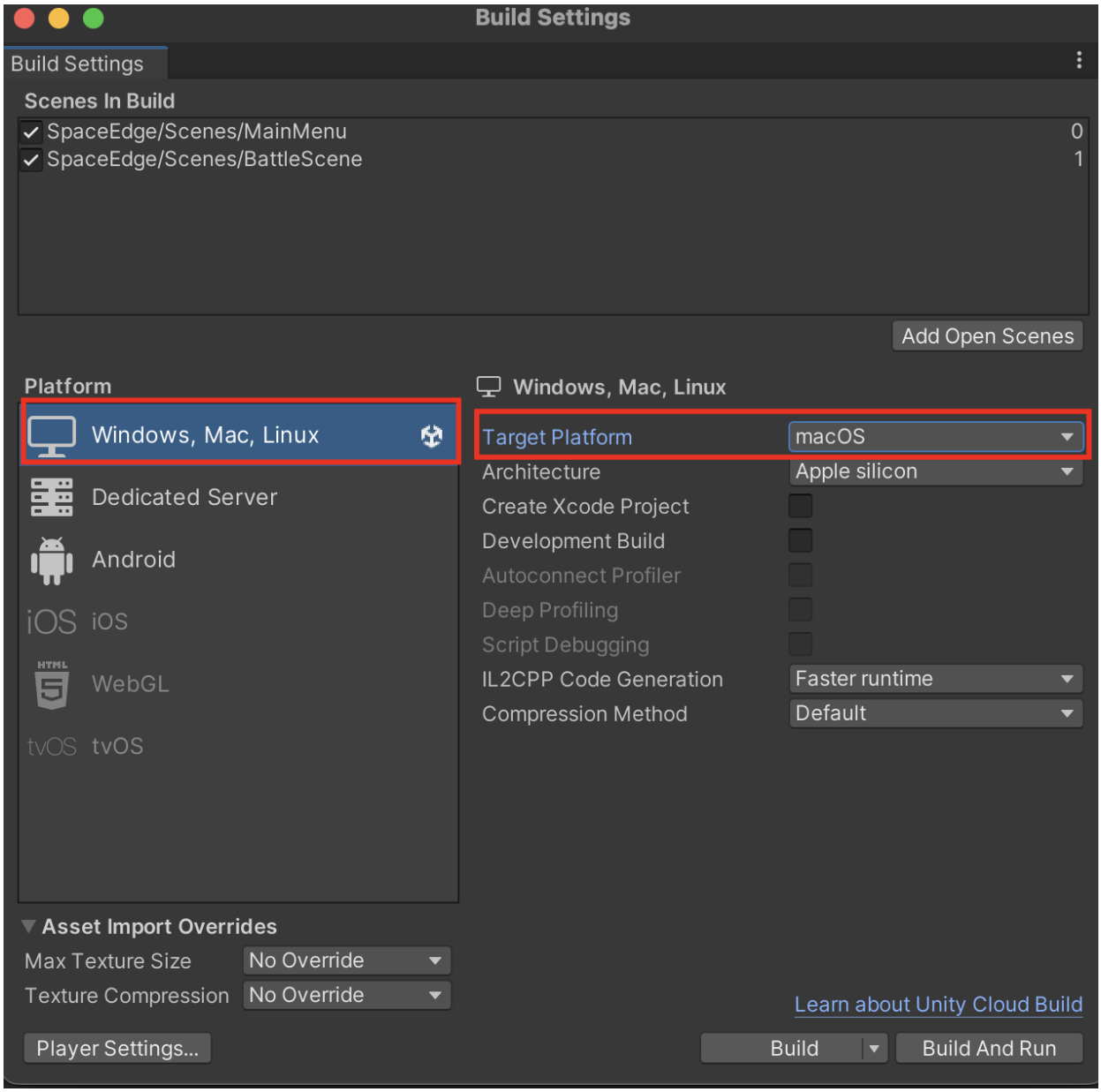
```
/// </summary>
1 asset usage  2 usages
public class MatchmakingSystem : MonoBehaviour
{
    /// <summary>
    ///     Name of the EdgeGap app we are trying to connect to
    ///     Please refer to the "FishNet Example Guide" for detailed instruction on how to configure app name.
    /// </summary>
    private const string AppName = "TestApp";

    /// <summary>
    ///     Version of the EdgeGap app we are trying to connect to
    ///     Please refer to the "FishNet Example Guide" for detailed instruction on how to configure app version.
    /// </summary>

    private const string AppVersion = "v1";
    /// <summary>
    ///     Used by the HTTP request's Authorization header.
    ///     Please refer to the "FishNet Example Guide" for detailed instruction on how to acquire this value.
    /// </summary>
    private const string AuthHeaderValue = "Super Secret Token 1234";

    /// <summary>
    ///     Used by the HTTP request's Content-Type header.
    ///     It should always be set to "application/json" while communicating with EdgeGap API.
    /// </summary>
    private const string TypeHeaderValue = "application/json";
```

AppName= “Set it to the “Application name” you used while setting up the EdgeGap application”.
AppVersion= “Set it to the “Version name” you used while setting up the EdgeGap application”.
AuthHeaderValue=”Set it to your EdgeGap Api Token that you can get from the EdgeGap console”.

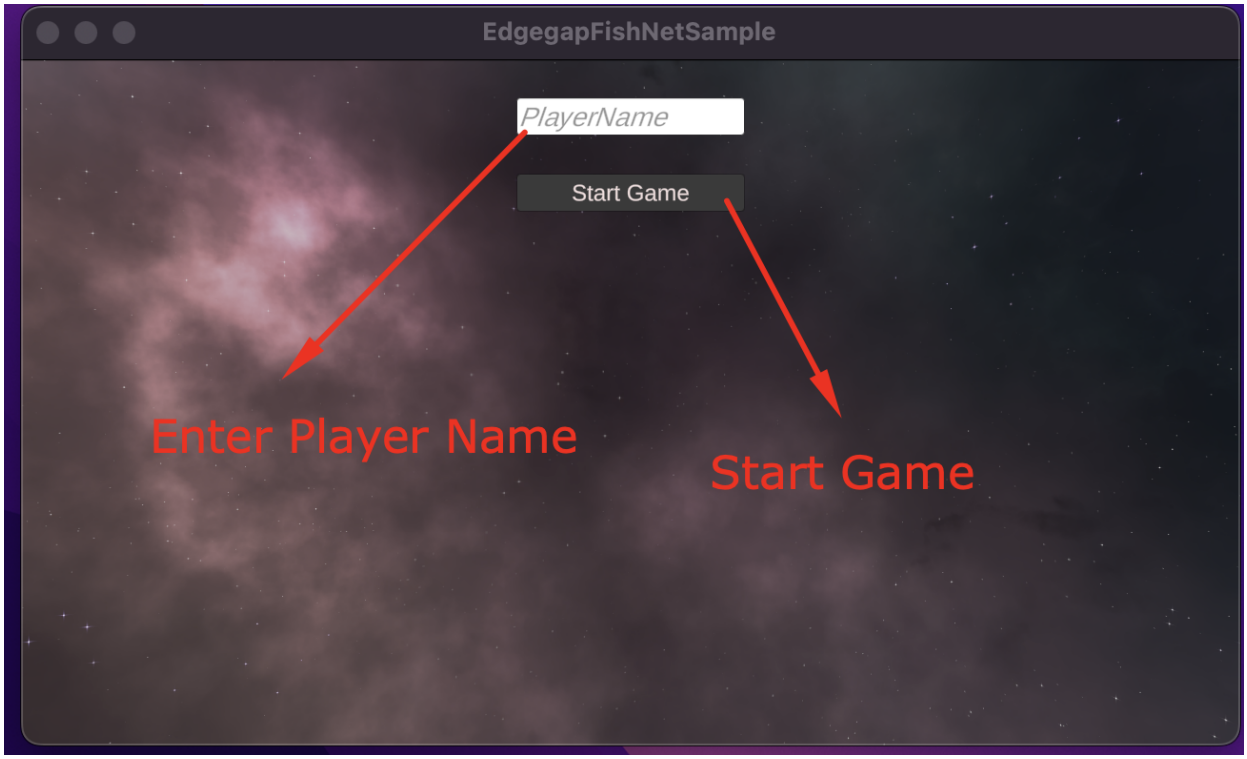
Tip: Remove the “token” word from the Api Token because we are setting this as the Authentication Header schema during the Awake() method of the MatchmakingSystem.



For the client build you can change the build type to “Windows, Mac, Linux” and build platform to any platform of your choice.



Now to test the whole system you can run the client build (or use the editor play mode). Enter your player name (optional) and click on the start game button.
The MatchmakingSystem should now search for any live deployments to connect you , and if no live deployments are found then it will request EdgeGap to deploy a new instance of your game's server and connect you to that instance once it's ready.



List Of All The Free Assets Used

- Cartoon FX Remaster Free | VFX Particles | Unity Asset Store
- Sci-Fi Sfx | Audio Sound FX | Unity Asset Store
- Starfield Skybox | 2D Sky | Unity Asset Store
- Star Sparrow Modular Spaceship | 3D Space | Unity Asset Store
- Total JSON | Input Management | Unity Asset Store