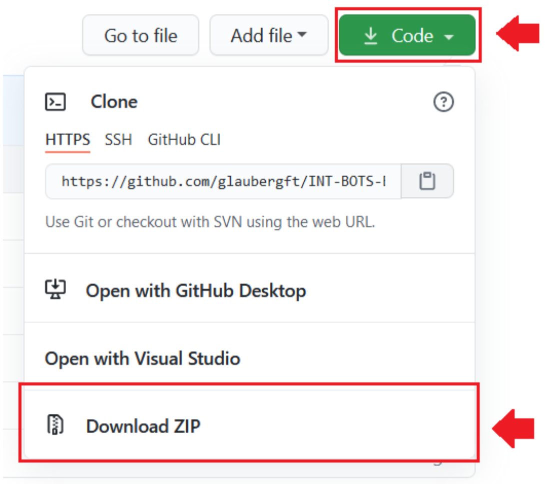
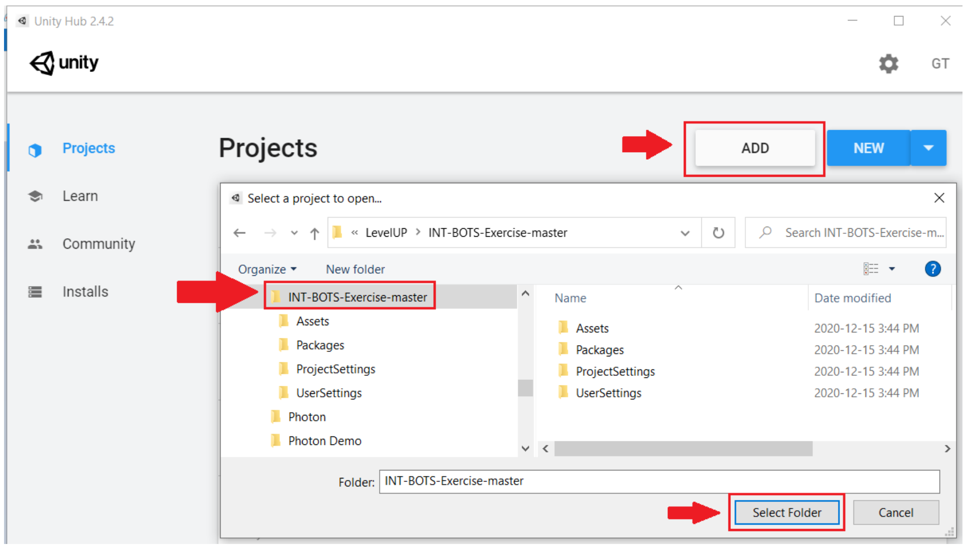
INT Bots – Multiplayer Exercise Guide

1 – Open your web browser, navigate to https://github.com/glaubergft/INT-BOTS-Exercise

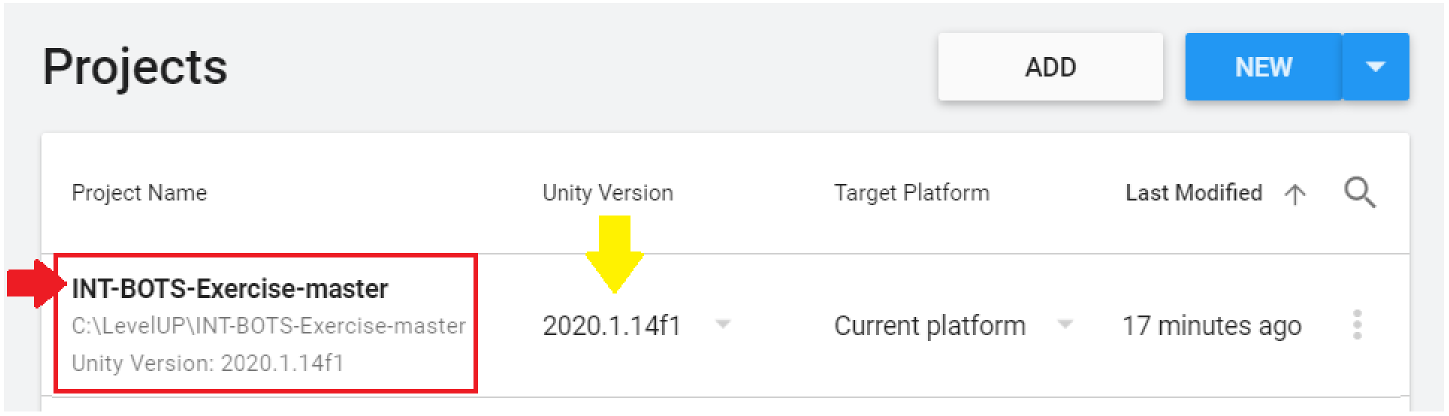
2 – Clone the repository to your machine or download the zip file.



3 – If you downloaded the zip file, extract the folder. Open Unity Hub and add this project folder. Make sure you see the folder Assets, Packages, ProjectSettings, and UserSettings. Click "Select Folder".


Unity Hub 2.4.2
unity
GT
Projects
Learn
Community
Installs
Projects
ADD
NEW
Select a project to open...
LevelUP > INT-BOTS-Exercise-master
Search INT-BOTS-Exercise-m...
Organize
New folder
INT-BOTS-Exercise-master
Assets
Packages
ProjectSettings
UserSettings
Photon
Photon Demo
Name
Date modified
Assets 2020-12-15 3:44 PM
Packages 2020-12-15 3:44 PM
ProjectSettings 2020-12-15 3:44 PM
UserSettings 2020-12-15 3:44 PM
Folder: INT-BOTS-Exercise-master
Select Folder
Cancel

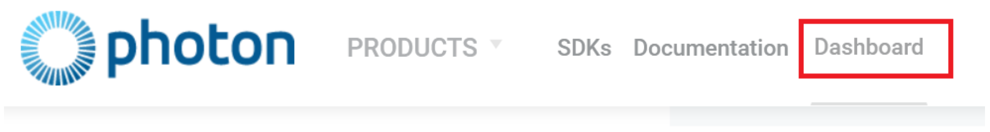
4 – Notice this project was originally made using Unity Version 2020.1.14f1. If you are using a higher version, you will be asked to upgrade this project. There's no guarantee it will work.

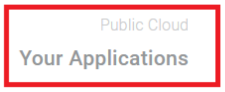


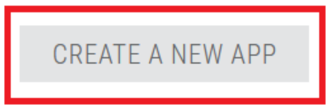
Project Name	Unity Version	Target Platform	Last Modified
INT-BOTS-Exercise-master C:\LevelUP\INT-BOTS-Exercise-master Unity Version: 2020.1.14f1	2020.1.14f1	Current platform	17 minutes ago

5 – While the project is loading, open your web browser, navigate to https://www.photonengine.com/ and Sign In.

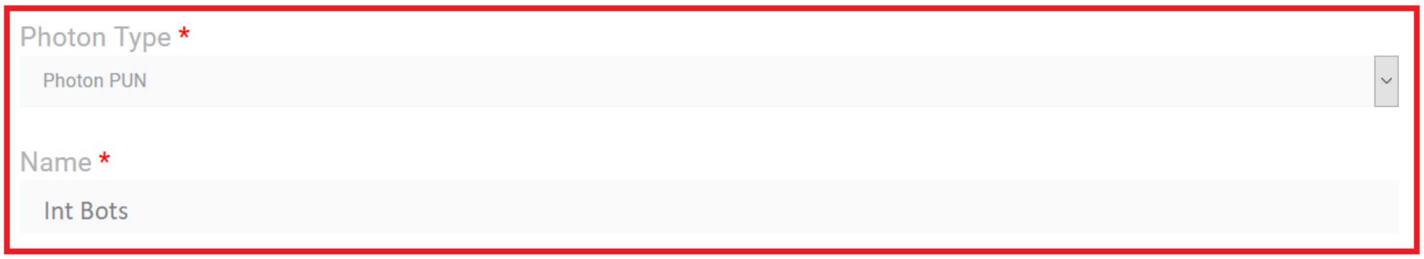
6 – Go to "Dashboard" => "Your Applications" => "CREATE A NEW APP"







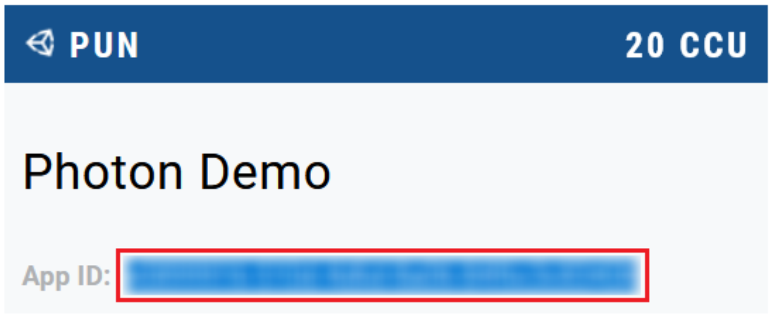
7 – For the "Photon Type" field, select "Photon PUN". Also give a name for your application. Finally click "CREATE".



Photon Type *
Photon PUN

Name *
Int Bots

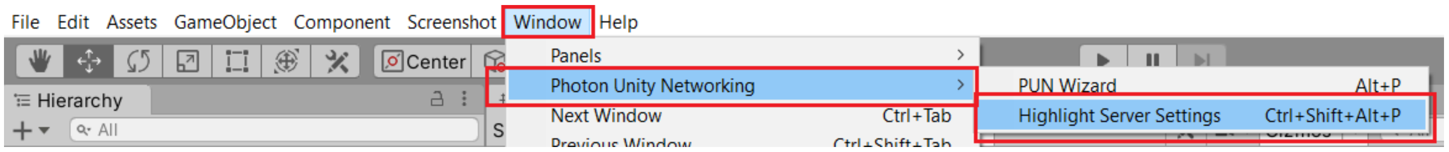
8 –Copy the entire App ID value:



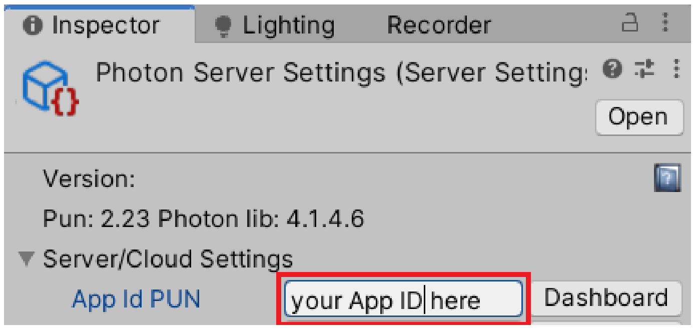
Photon Demo

App ID: [redacted]

9 – Back to Unity editor, click on the menu “Window” => “Photon Unity Networking” => “Highlight Server Settings”.



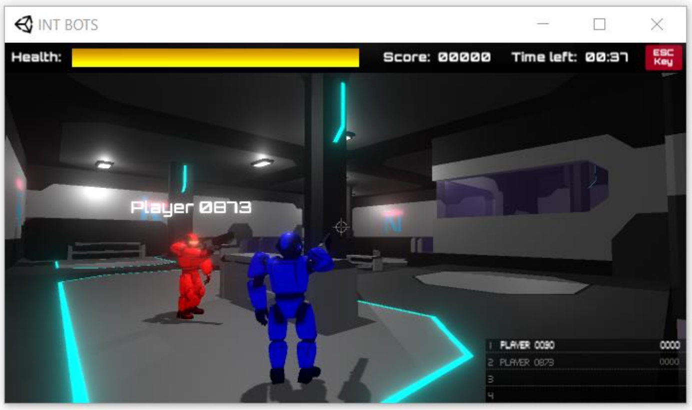
10 – Past the App ID value:

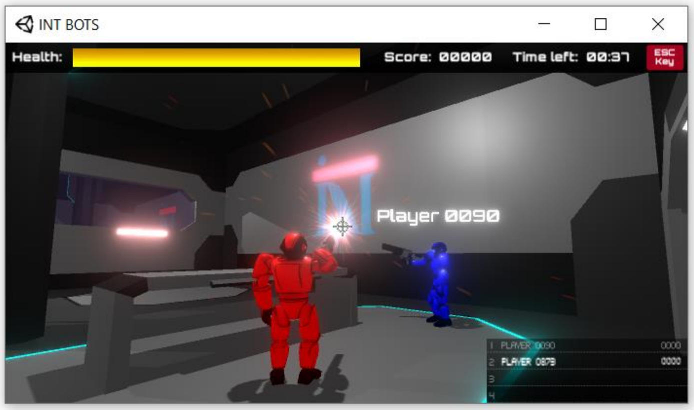


Version:
Pun: 2.23 Photon lib: 4.1.4.6
Server/Cloud Settings
App Id PUN: your App ID here

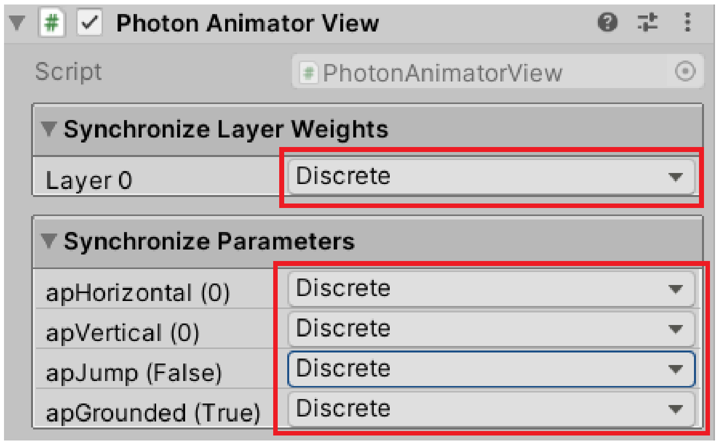
11 – In the main menu, click “File” => “Build Settings”. Click “Add open Scenes”. Make sure the current selected platform is “PC, Mac & Linux Standalone”. Check “**Development Build**” option, then click Build. Create a folder called “PCBUILD” and press “Select Folder”.

12 – Once this build process is over, open at least 2 instances of the generated executable. Test the game and notice the player animation is not working across the other game instance(s). Also, inside your own instance when you shot, the other players are shooting. Finally, your projectiles don’t show up on the other instance. Close these executables.





13 – To fix the animation bug, in the Inspector window, select the Player prefab found in the “Resources\ MultiplayerPrefabs” folder. Add the Photon Animator View component. Set the Layer 0 weight to “**Discrete**”. Set all parameters (apHorizontal, apVertical, apJump, apGrounded) to “**Discrete**”.


Photon Animator View

Script: PhotonAnimatorView

Synchronize Layer Weights

Layer 0	Discrete

Synchronize Parameters

apHorizontal (0)	Discrete
apVertical (0)	Discrete
apJump (False)	Discrete
apGrounded (True)	Discrete

14 – To fix the shooting bug, first open the script found in Scripts\Character\CharacterGun.cs then replace this code:

```
internal void Shoot()
{
    GameObject instance = Instantiate(projectile, launcher.position, launcher.rotation);
    instance.transform.position = launcher.position;
    instance.transform.forward = characterCamera.forward;
}
```

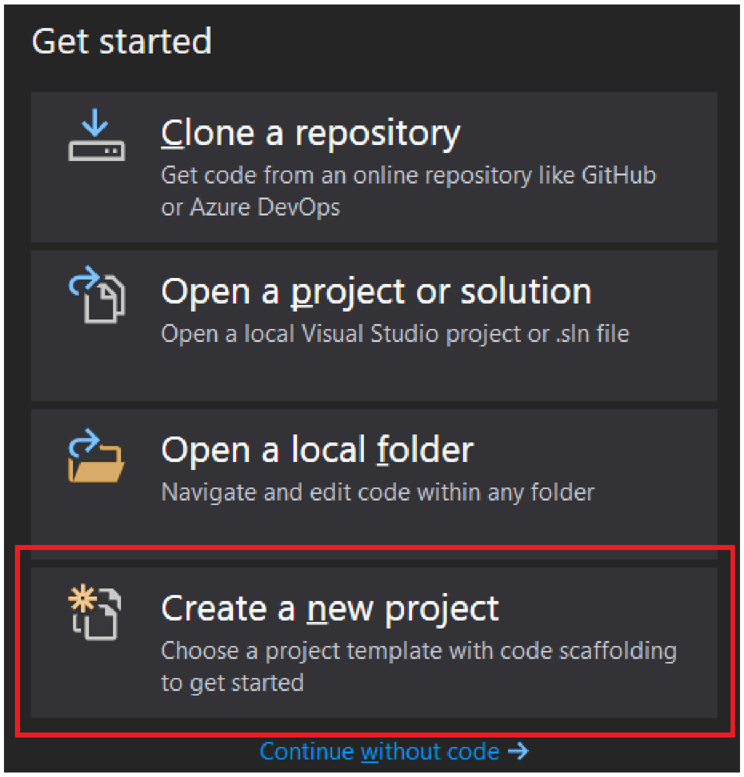
...with this:

```
internal void Shoot()
{
    string projectileId = Guid.NewGuid().ToString();
    if (PhotonNetwork.InRoom && view.IsMine)
    {
        view.RPC("ExecuteShoot", RpcTarget.All, projectileId, view.Owner.ActorNumber);
    }
    else if (!PhotonNetwork.IsConnected)
    {
        ExecuteShoot(projectileId, 0);
    }

}


[PunRPC]
private void ExecuteShoot(string projectileId, int actorNumber)
{
    GameObject instance = Instantiate(projectile, launcher.position, launcher.rotation);
    instance.transform.position = launcher.position;
    instance.transform.forward = characterCamera.forward;
    instance.GetComponent<Projectile>().ProjectileId = projectileId;
    instance.GetComponent<Projectile>().ActorNumber = actorNumber;
}
```

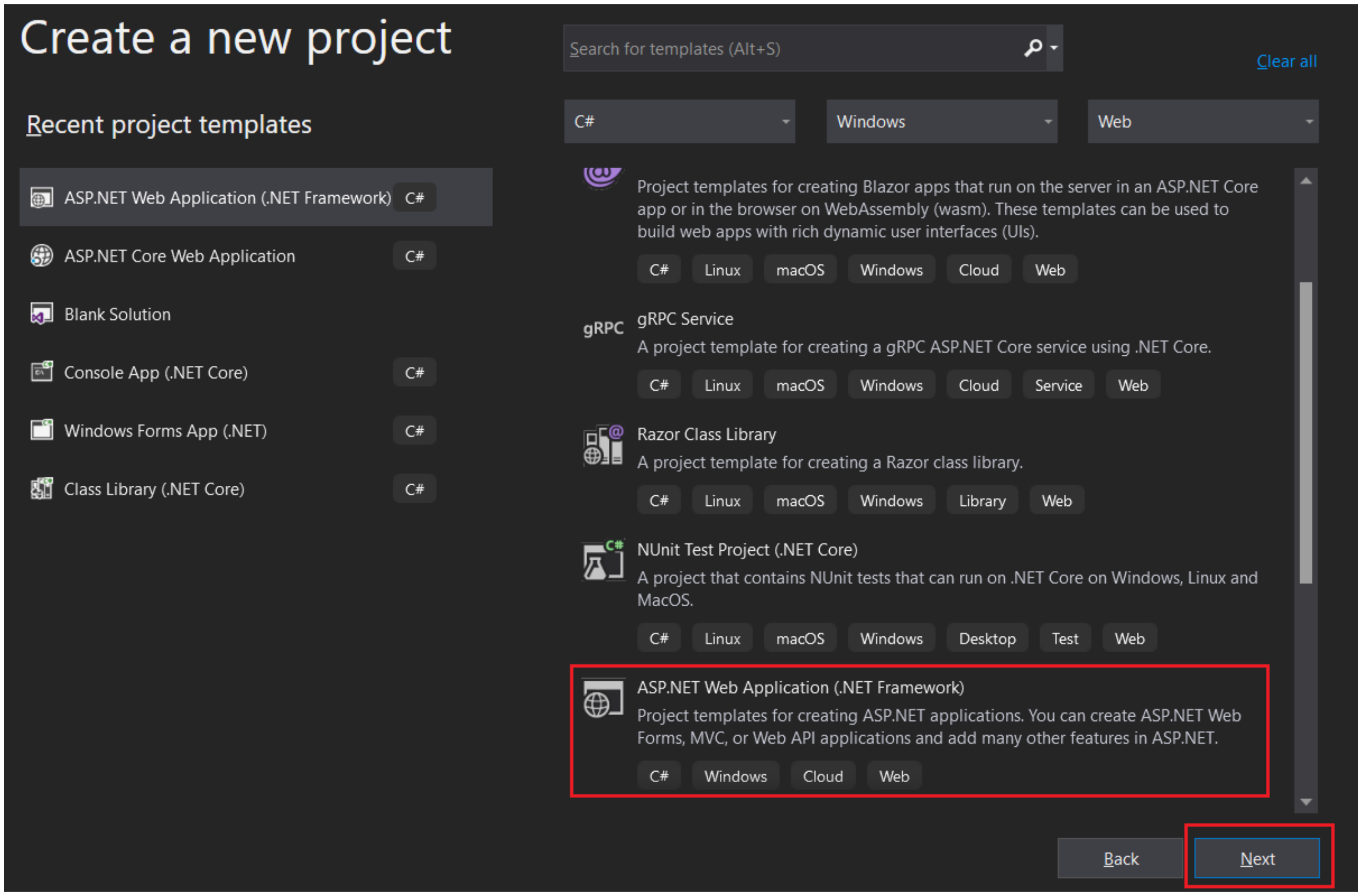
15 – Generate a new build and check if both issues were fixed by test the game again.

INT Bots – WebGL Export and Azure Publishing Guide

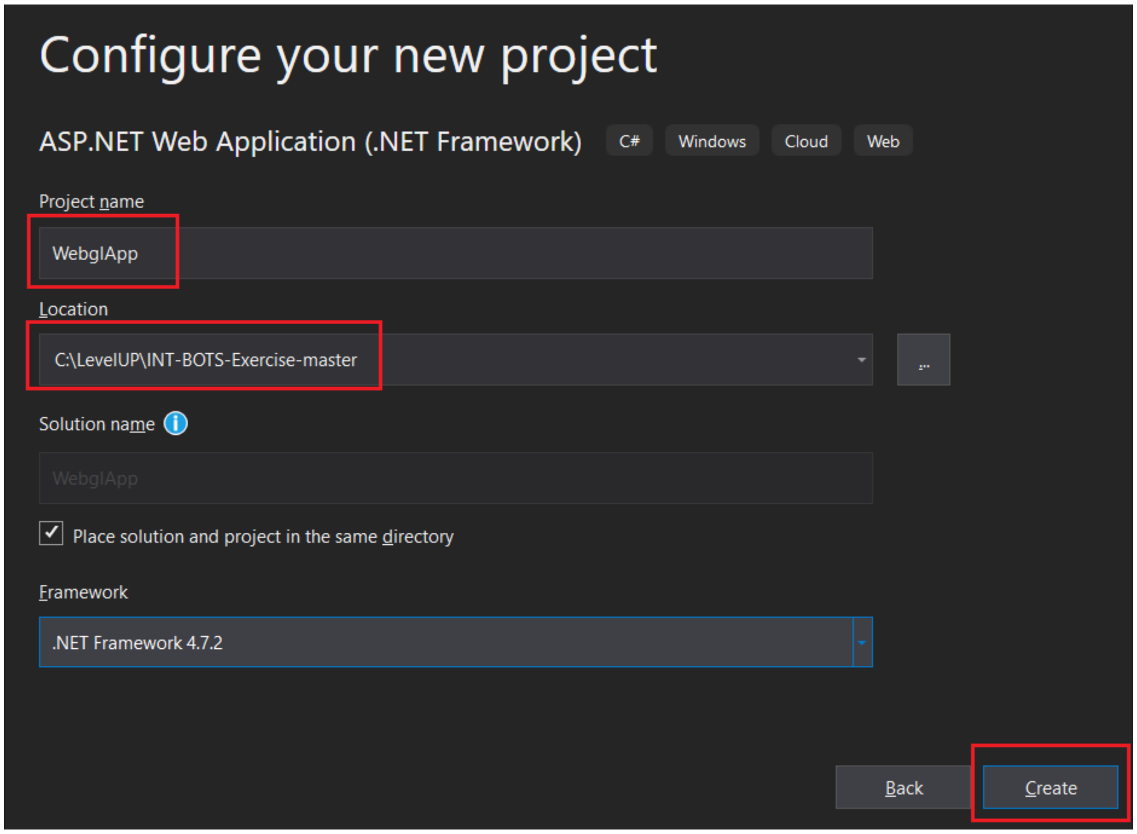
1 – Open Visual Studio 2019, click "Create a new project"


Get started
Clone a repository
Get code from an online repository like GitHub or Azure DevOps
Open a project or solution
Open a local Visual Studio project or .sln file
Open a local folder
Navigate and edit code within any folder
Create a new project
Choose a project template with code scaffolding to get started
Continue without code →

2 – On the Create a new project window, select "ASP.NET Web Application (.NET Framework) and click


Create a new project
Search for templates (Alt+S)
Clear all
C#
Windows
Web
Recent project templates
ASP.NET Web Application (.NET Framework) C#
ASP.NET Core Web Application C#
Blank Solution
Console App (.NET Core) C#
Windows Forms App (.NET) C#
Class Library (.NET Core) C#
Project templates for creating Blazor apps that run on the server in an ASP.NET Core app or in the browser on WebAssembly (wasm). These templates can be used to build web apps with rich dynamic user interfaces (UIs).
C# Linux macOS Windows Cloud Web
gRPC Service
A project template for creating a gRPC ASP.NET Core service using .NET Core.
C# Linux macOS Windows Cloud Service Web
Razor Class Library
A project template for creating a Razor class library.
C# Linux macOS Windows Library Web
NUnit Test Project (.NET Core)
A project that contains NUnit tests that can run on .NET Core on Windows, Linux and MacOS.
C# Linux macOS Windows Desktop Test Web
ASP.NET Web Application (.NET Framework)
Project templates for creating ASP.NET applications. You can create ASP.NET Web Forms, MVC, or Web API applications and add many other features in ASP.NET.
C# Windows Cloud Web
Back
Next

3 – Give the name " WebglApp" for the project and set the location as same as your Unity project. Then click “Create”.



Configure your new project

ASP.NET Web Application (.NET Framework)

Project name: WebglApp

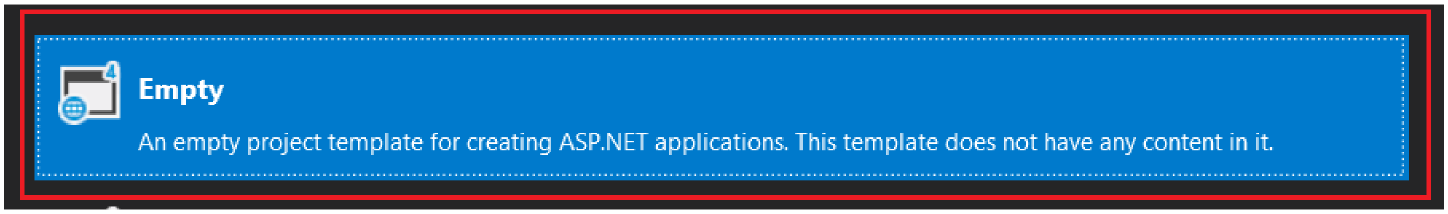
Location: C:\LevelUP\INT-BOTS-Exercise-master

Solution name: WebglApp

✓ Place solution and project in the same directory

Framework: .NET Framework 4.7.2

4 – Select the “Empty” template, leave the remaining options unchanged and then click “Create”.



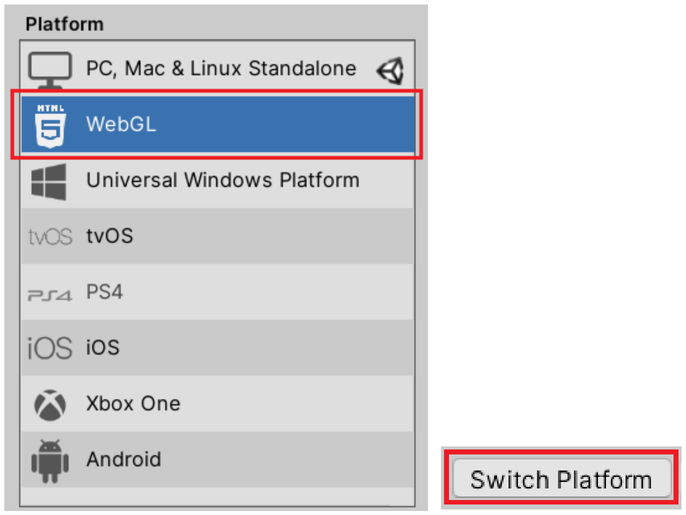
Empty

An empty project template for creating ASP.NET applications. This template does not have any content in it.

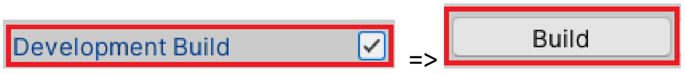
5 – Open the Web.config file. Delete all lines and paste the following content and save the file:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
 The following server configuration can be used for uncompressed WebGL builds.
 This configuration file should be uploaded to the server as "<Application Folder>/Build/web.config"
 This configuration has been tested with Unity 2020.1 builds, hosted on IIS 7.5, IIS 8.5, and IIS 10.0.
 SOURCE:
 https://docs.unity3d.com/Manual/webgl-deploying.html
-->
<configuration>
  <system.webServer>
    <!--
     IIS does not provide default handlers for .data and .wasm files (and in some cases .json files),
     therefore these files won’t be served unless their mimeType is explicitly specified.
   -->
    <staticContent>
      <!--
       NOTE: IIS will throw an exception if a mimeType is specified multiple times for the same
extension.
       To avoid possible conflicts with configurations that are already on the server, you should
remove the mimeType for the corresponding extension using the <remove> element,
       before adding mimeType using the <mimeMap> element.
     -->
      <remove fileExtension=".data" />
      <mimeMap fileExtension=".data" mimeType="application/octet-stream" />
      <remove fileExtension=".wasm" />
      <mimeMap fileExtension=".wasm" mimeType="application/wasm" />
      <remove fileExtension=".symbols.json" />
      <mimeMap fileExtension=".symbols.json" mimeType="application/octet-stream" />
    </staticContent>
  </system.webServer>
</configuration>
```

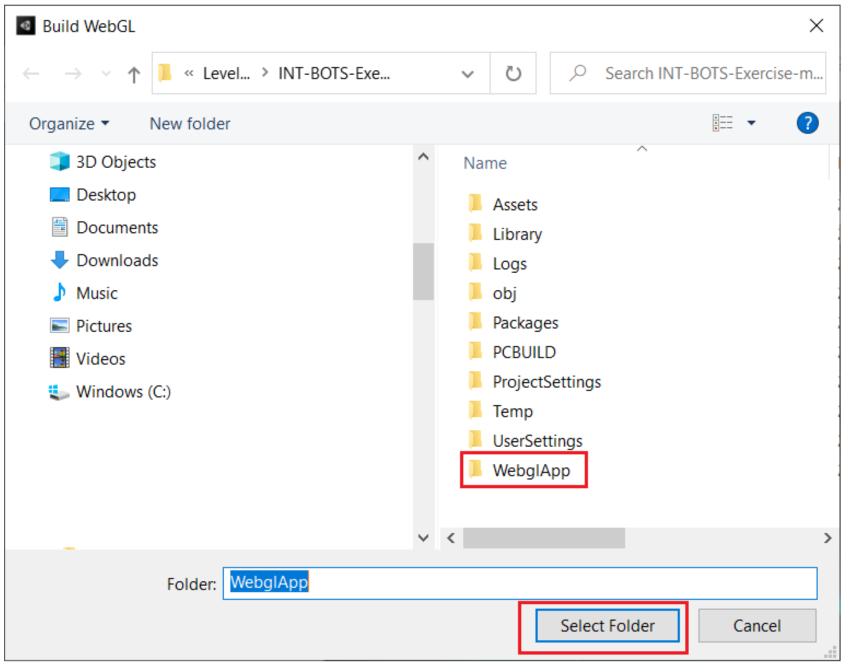
6 –Go back to INT Bots project in Unity. Go to the menu “File” => “Build Setting”. Select the “WebGL” platform and then click Switch Platform.



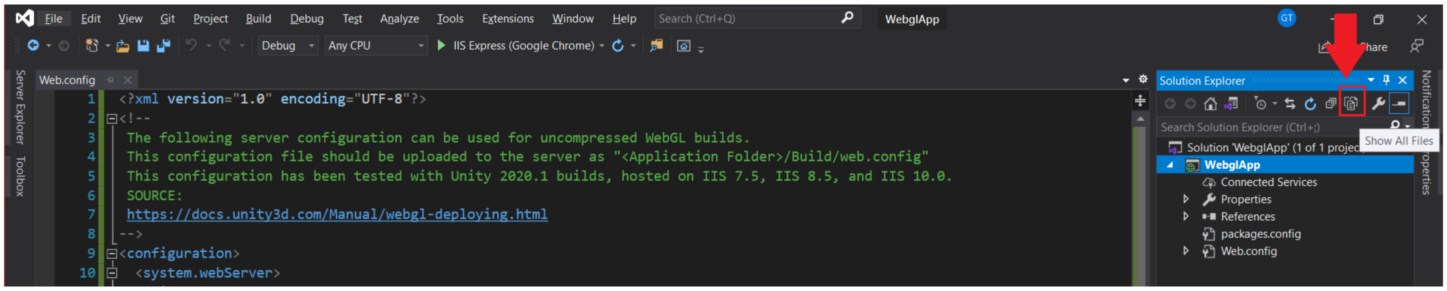
7 – After Unity finishes switching to WebGL, check “Development Build” option so we can test the multiplayer feature with Unity Editor. Finally click on the “Build” button.



8 – Select the “WebglApp” folder and click “Select Folder”

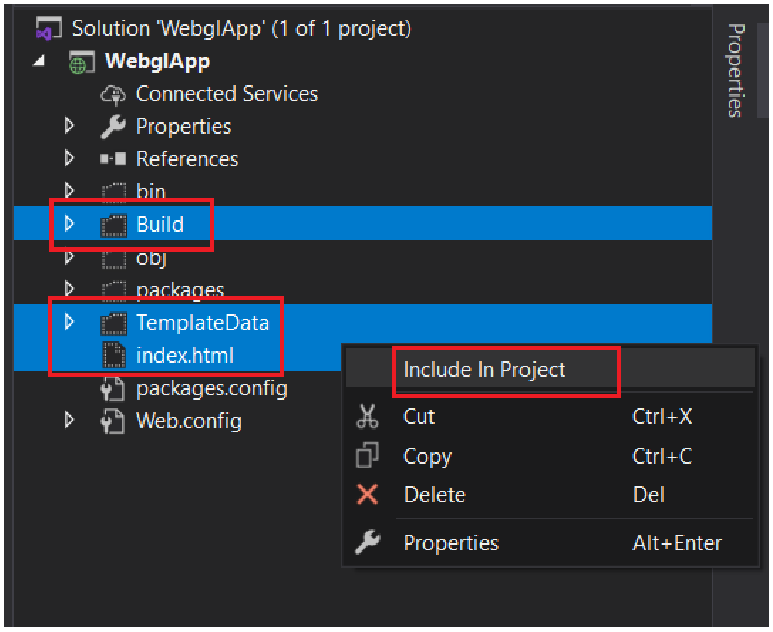


9 – Switch back to the ASP.NET Visual Studio application. In the Solution Explorer window, click “Show All Files” button.

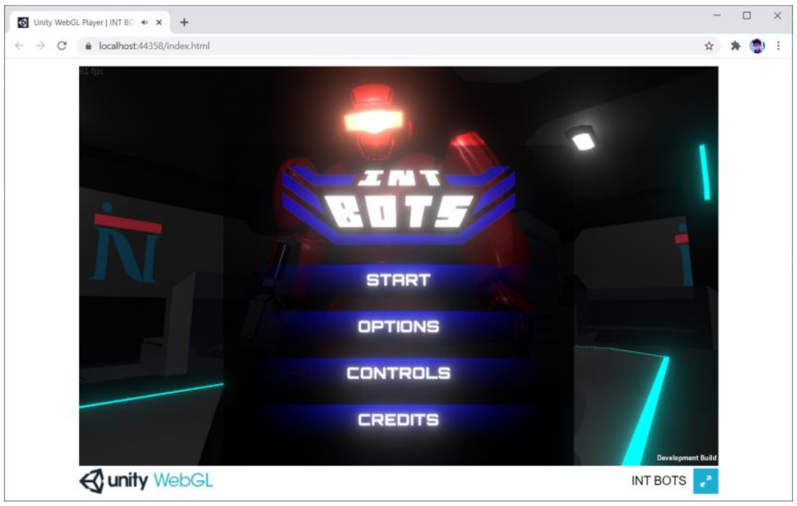

File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Search (Ctrl+Q)
WebglApp
Debug
Any CPU
IIS Express (Google Chrome)
Web.config
Solution Explorer
Search Solution Explorer (Ctrl+;)
Show All Files
Solution 'WebglApp' (1 of 1 projec
WebglApp
Connected Services
Properties
References
packages.config
Web.config

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
  The following server configuration can be used for uncompressed WebGL builds.
  This configuration file should be uploaded to the server as "<Application Folder>/Build/web.config"
  This configuration has been tested with Unity 2020.1 builds, hosted on IIS 7.5, IIS 8.5, and IIS 10.0.
  SOURCE:
  https://docs.unity3d.com/Manual/webgl-deploying.html
-->
<configuration>
  <system.webServer>
```

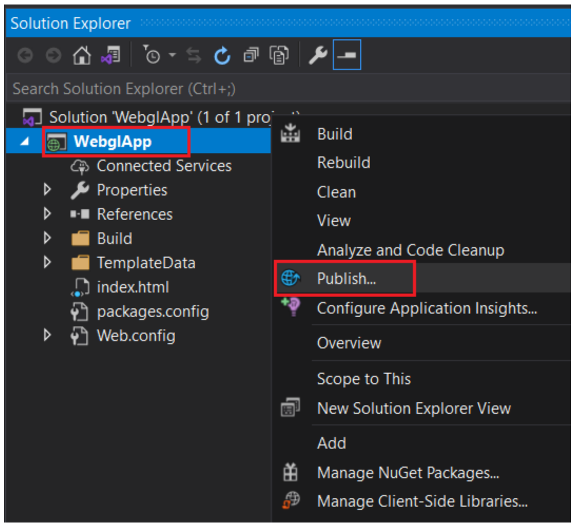
10 – Hold Ctrl key and select “Build”, “TemplateData”, and index.html. Right click and select “Include In Project”.



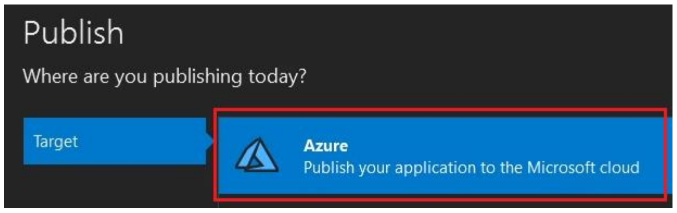
11 – Right click index.html => “View in Browser”. It should open the game in your browser by running a local webserver.



12 – Go back to Visual Studio, right click the WebglApp project, then click “Publish”.

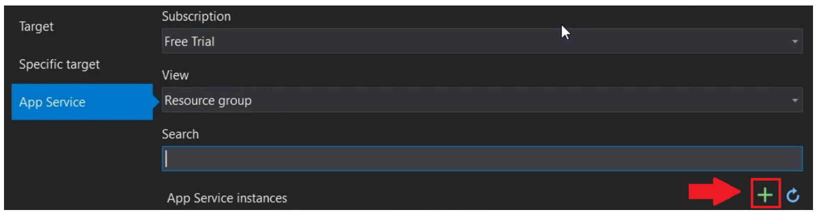


13 – For the target select “Azure” and click “Next”

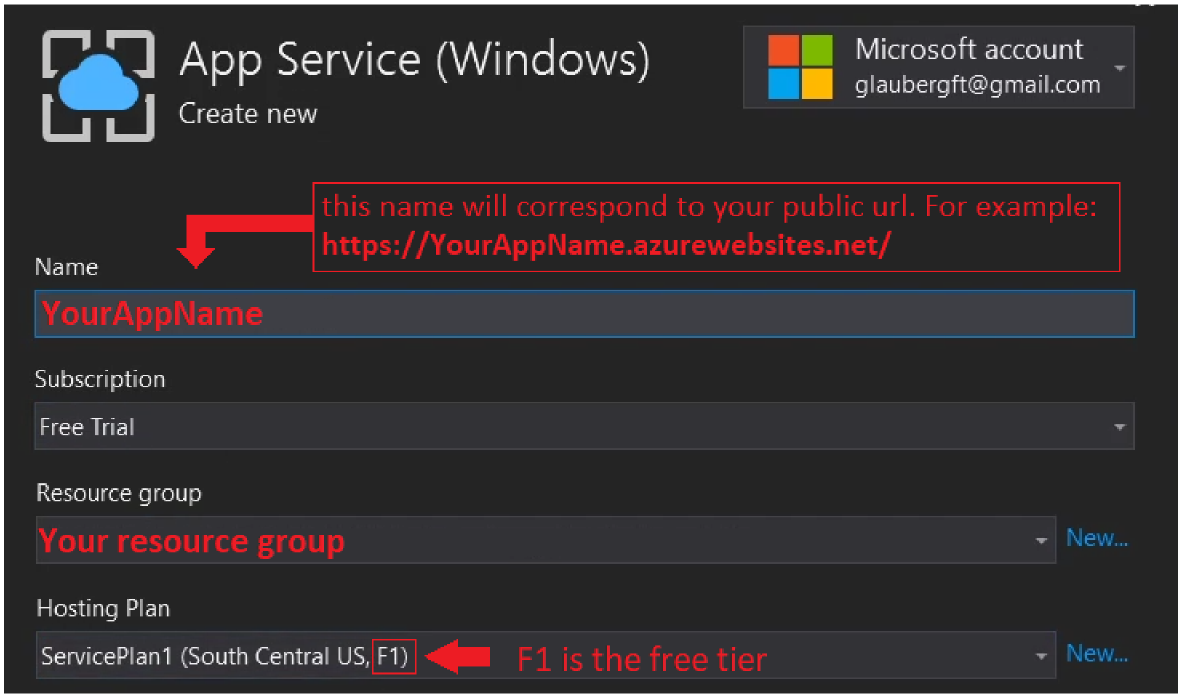


Publish
Where are you publishing today?
Target
Azure
Publish your application to the Microsoft cloud

14 – Create/Select a resource group, then create an App Service instance by click the “plus” button.



15 – Inform a name for your app. It will reflect on the public URL. Select your resource group and don't forget to select a **F1** hosting plan. F1 is the free tier. Finally, click "Finish".



App Service (Windows)
Create new

Microsoft account
glaubergft@gmail.com

this name will correspond to your public url. For example: https://YourAppName.azurewebsites.net/

Name: YourAppName

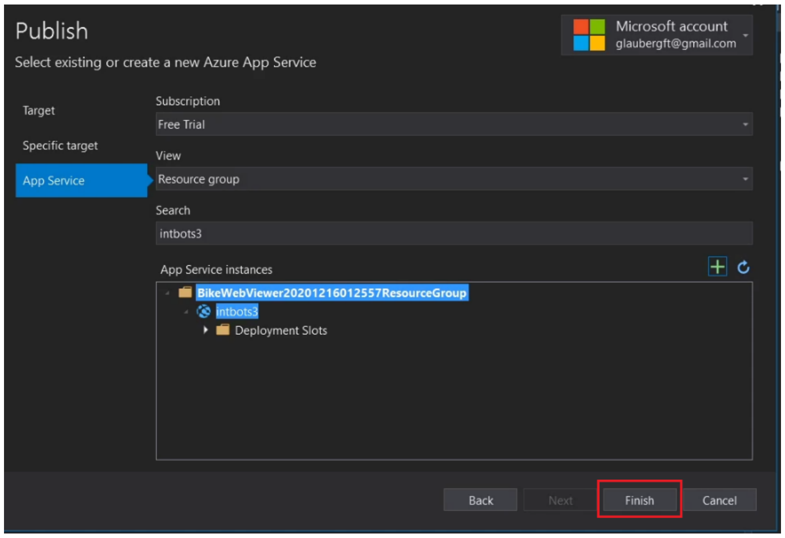
Subscription: Free Trial

Resource group: Your resource group

Hosting Plan: ServicePlan1 (South Central US, F1)

F1 is the free tier

16 – Once the app service creation process has finished, back to the Publish window, click "Finish".



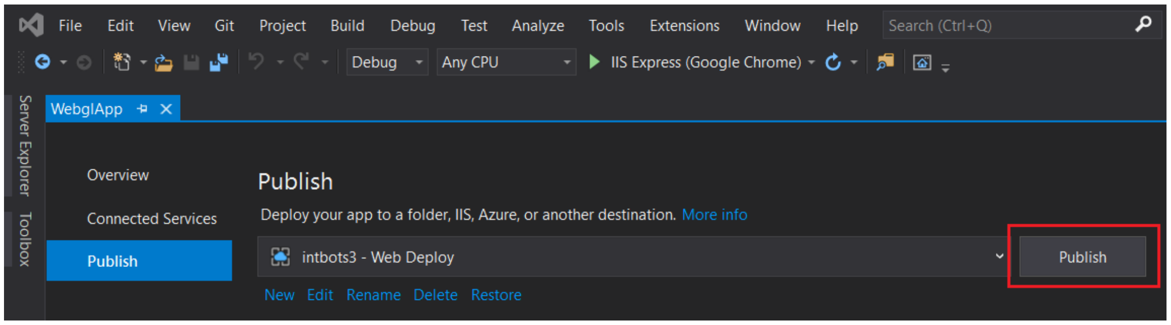
Publish
Select existing or create a new Azure App Service

Microsoft account
glaubergft@gmail.com

Subscription: Free Trial
View: Resource group
Search: intbots3

App Service instances
BikeWebViewer20201216012557ResourceGroup
intbots3
Deployment Slots

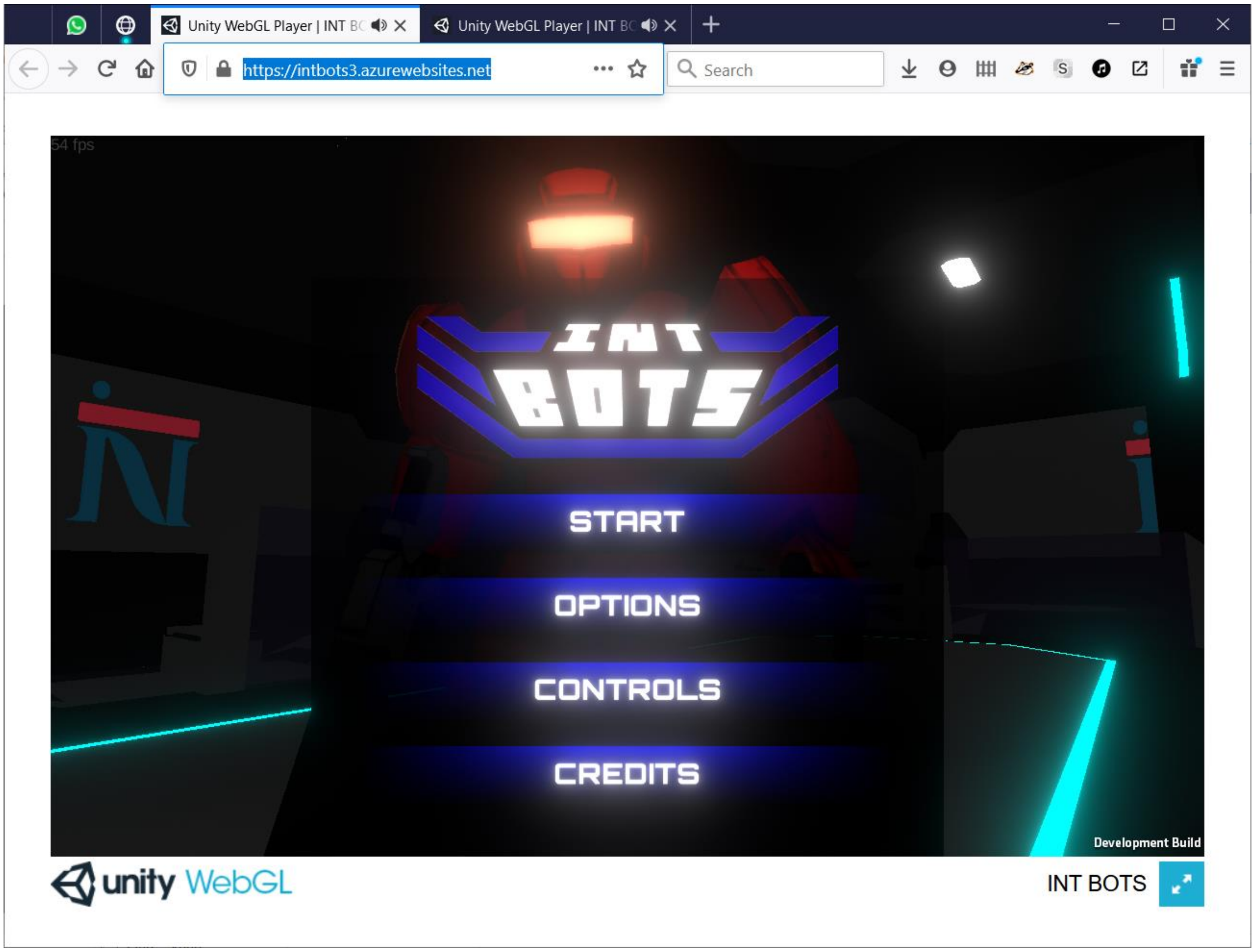
17 – Click Publish.



Publish
Deploy your app to a folder, IIS, Azure, or another destination. More info

intbots3 - Web Deploy

18 – It will take a little while to finish this process. Your game has been successfully hosted on Azure and will show up on your browser.


Unity WebGL Player | INT BC
Unity WebGL Player | INT BC
https://intbots3.azurewebsites.net
Search
54 fps
INT BOTS
START
OPTIONS
CONTROLS
CREDITS
Development Build
unity WebGL
INT BOTS