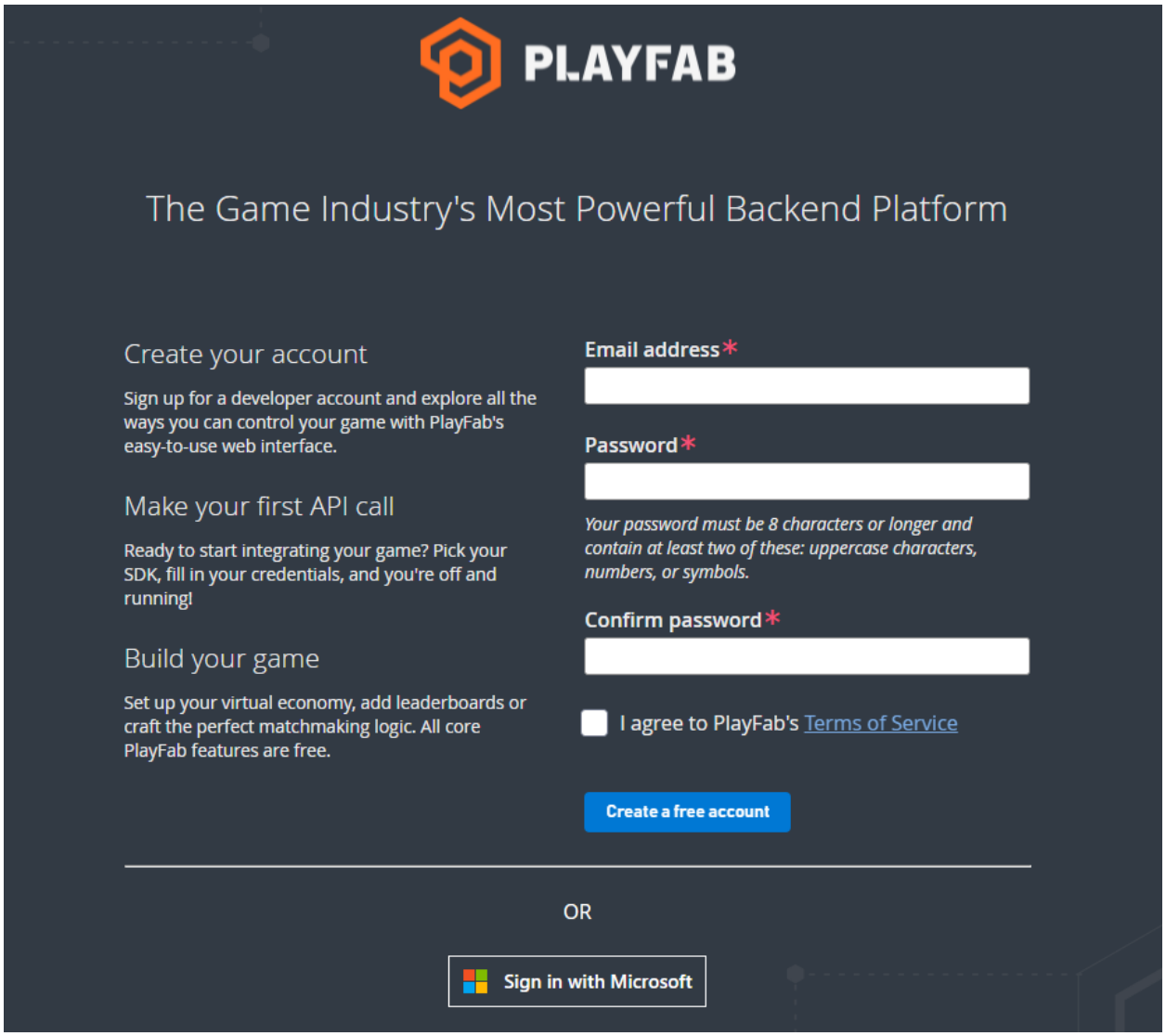
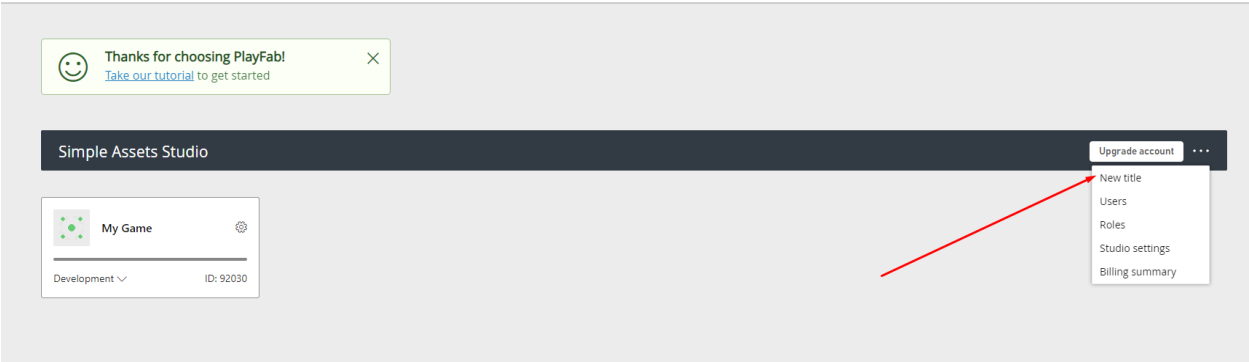
Install plugin (for version 0.4.0 or higher)
1. Playfab

First of all you need to register on Playfab. https://developer.playfab.com/



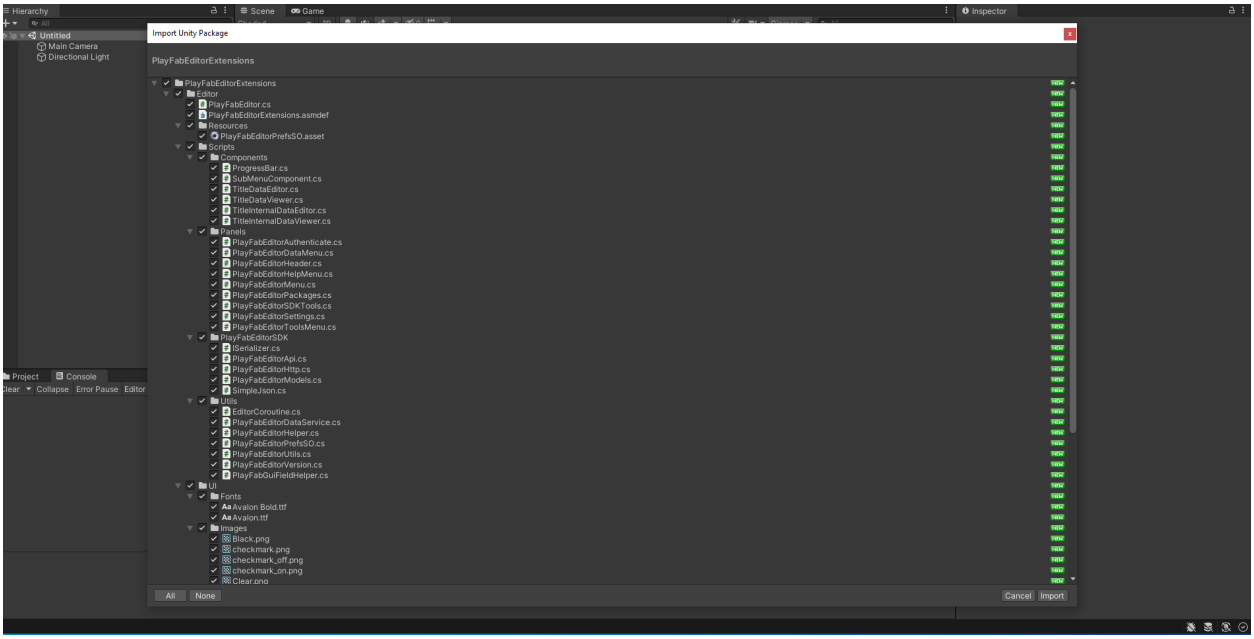
Create new title on Playfab.



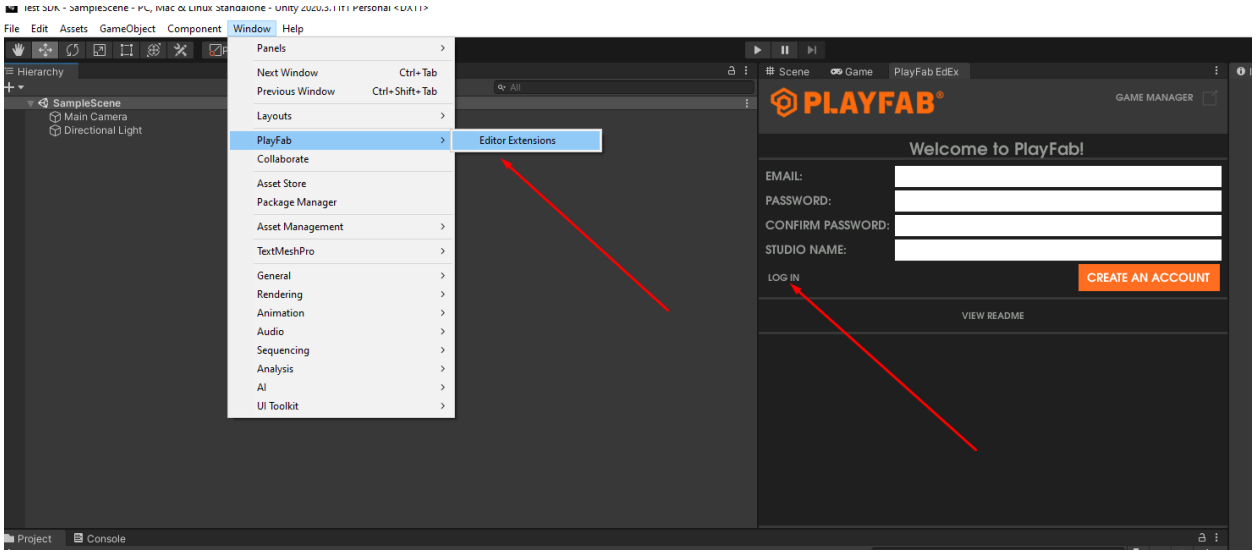
It's all about Playfab.

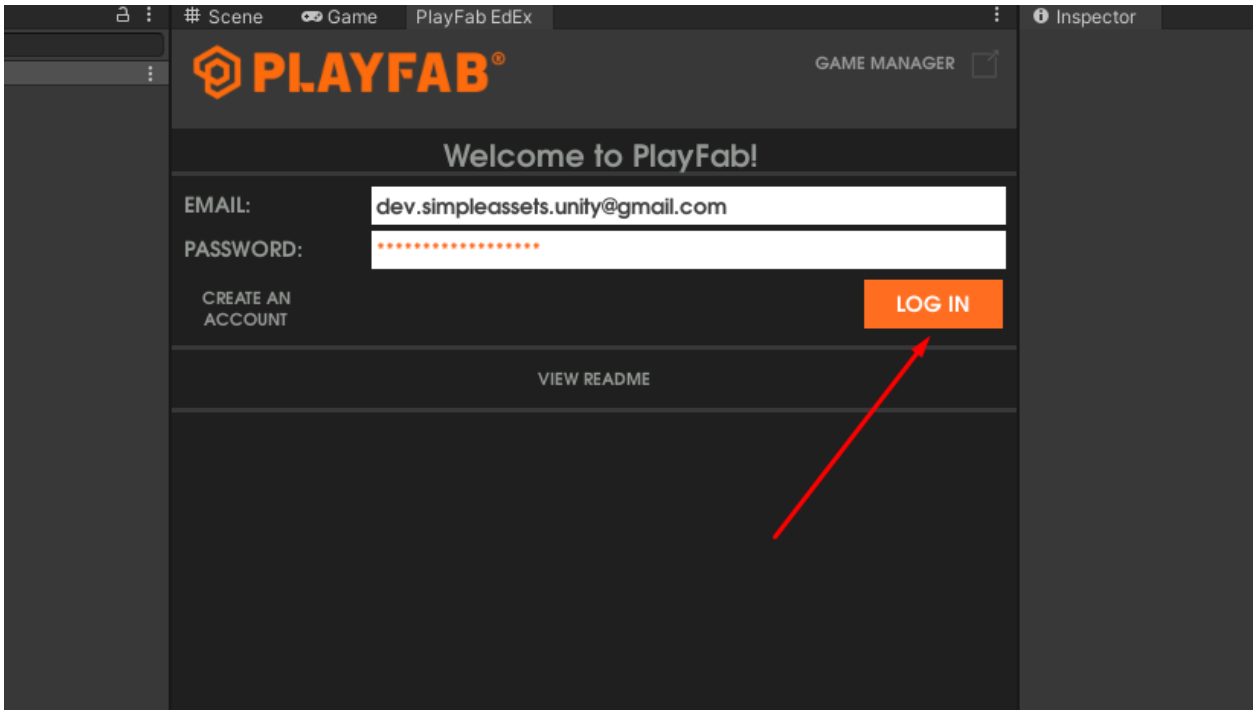
2. Import CBS Plugin.

First of all, install the Playfab Editor Extension. Download unitypackage https://aka.ms/playfabunityextension and import to Unity.

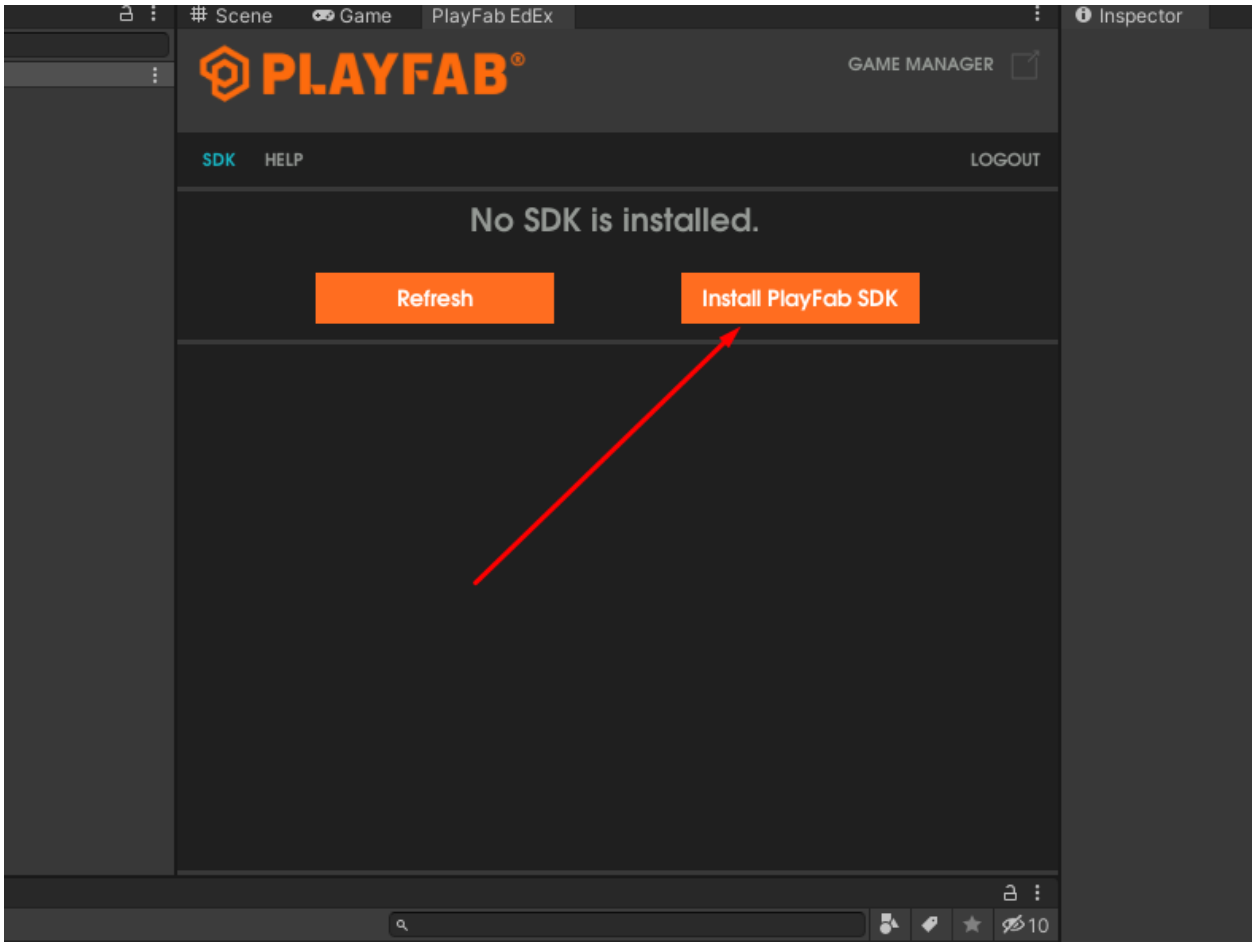


Open Playfab editor and login with Playfab account.
WARNING. If the Playfab context menu does not appear - just restart Unity.

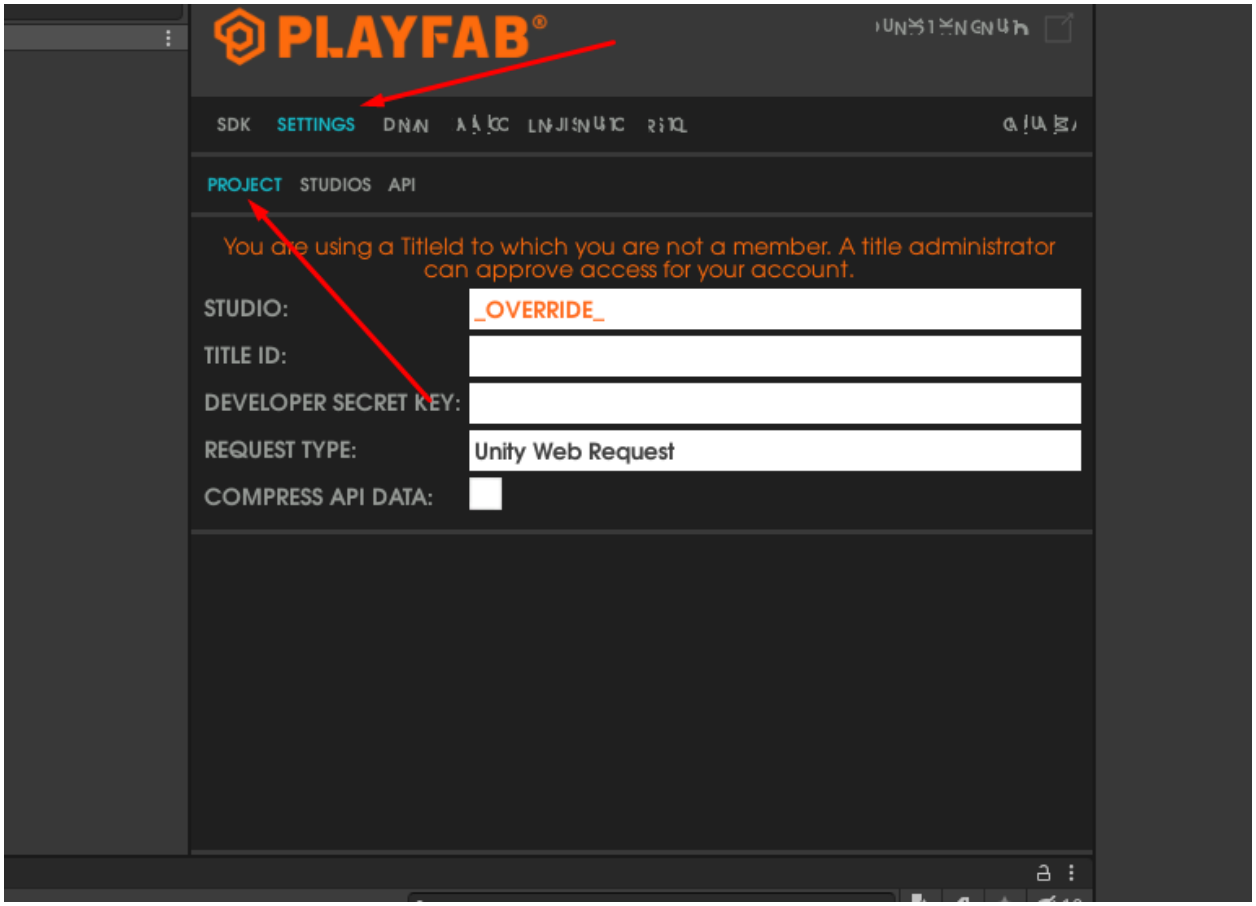




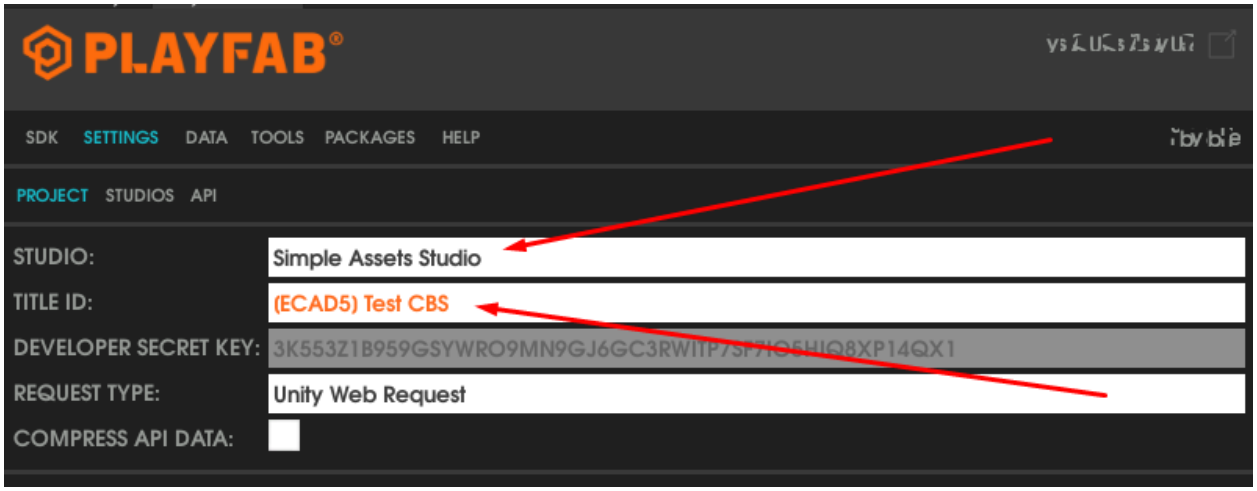
Install Playfab SDK.



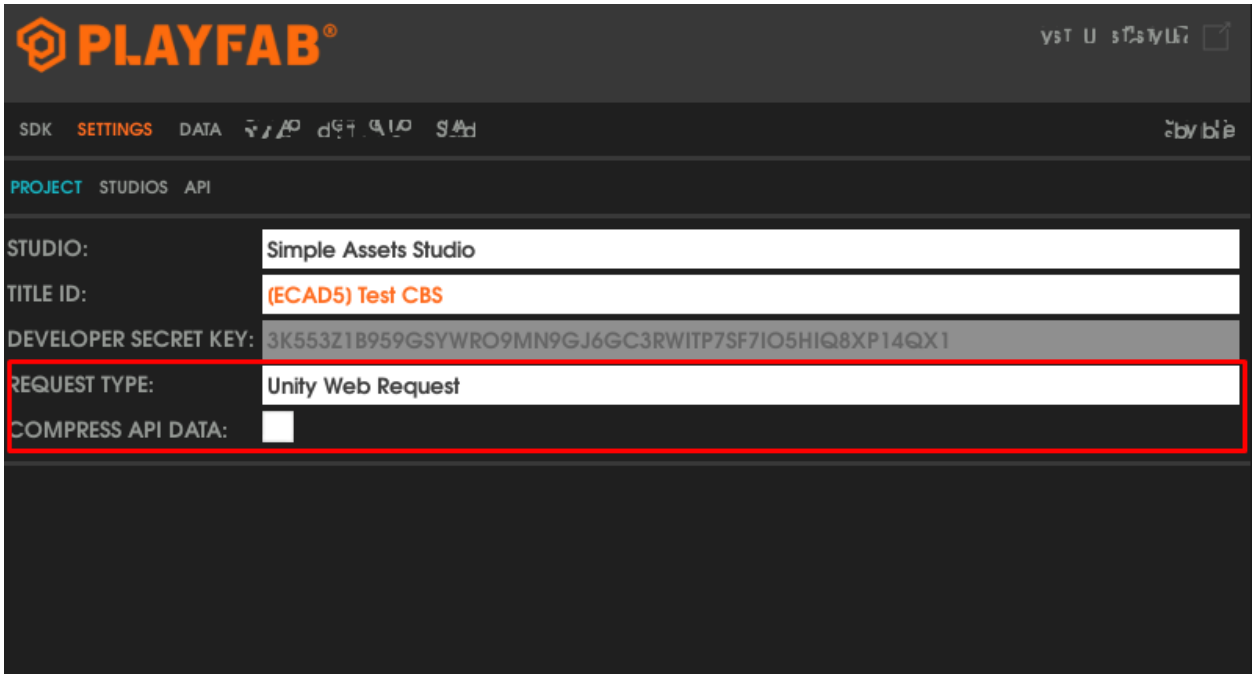
Navigate to SETTING->PROJECT



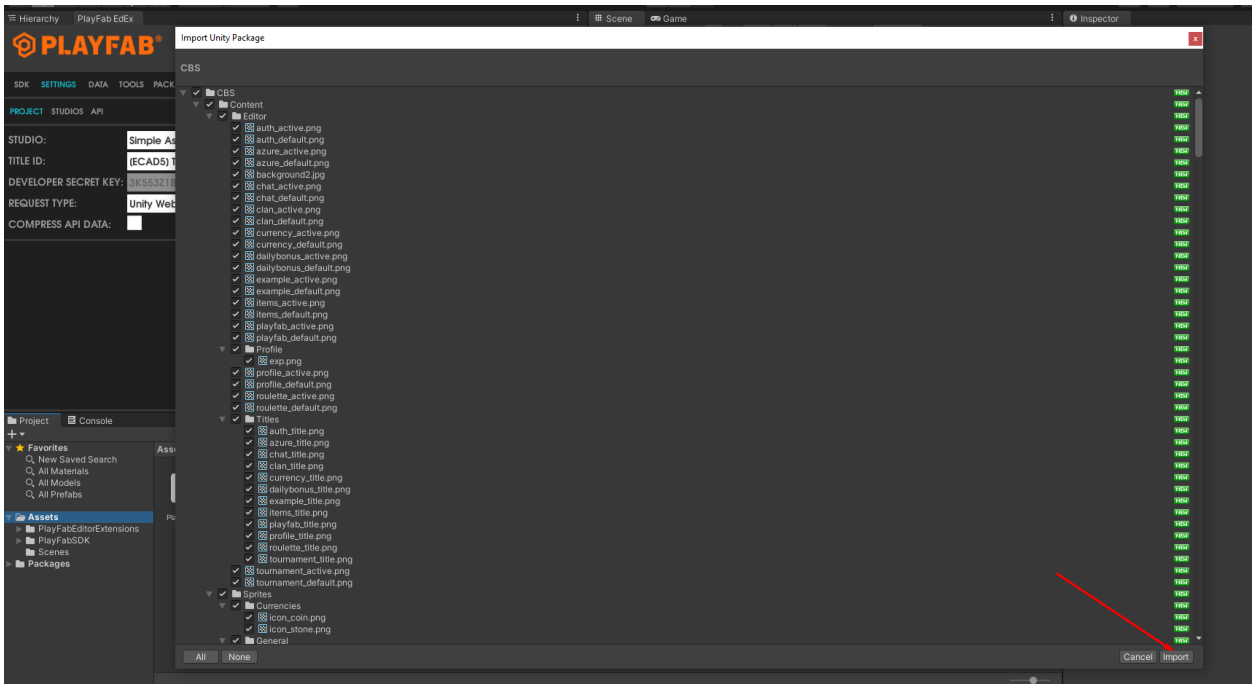
Select the Studio and Title ID you created earlier



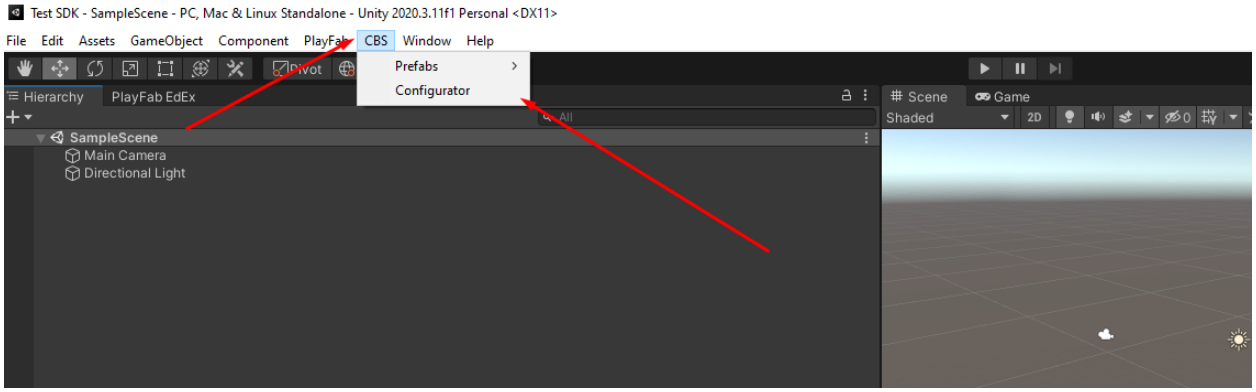
Select Request Type - Unity Web Request and don't Compress Api Data to make the solution as cross-platform as possible



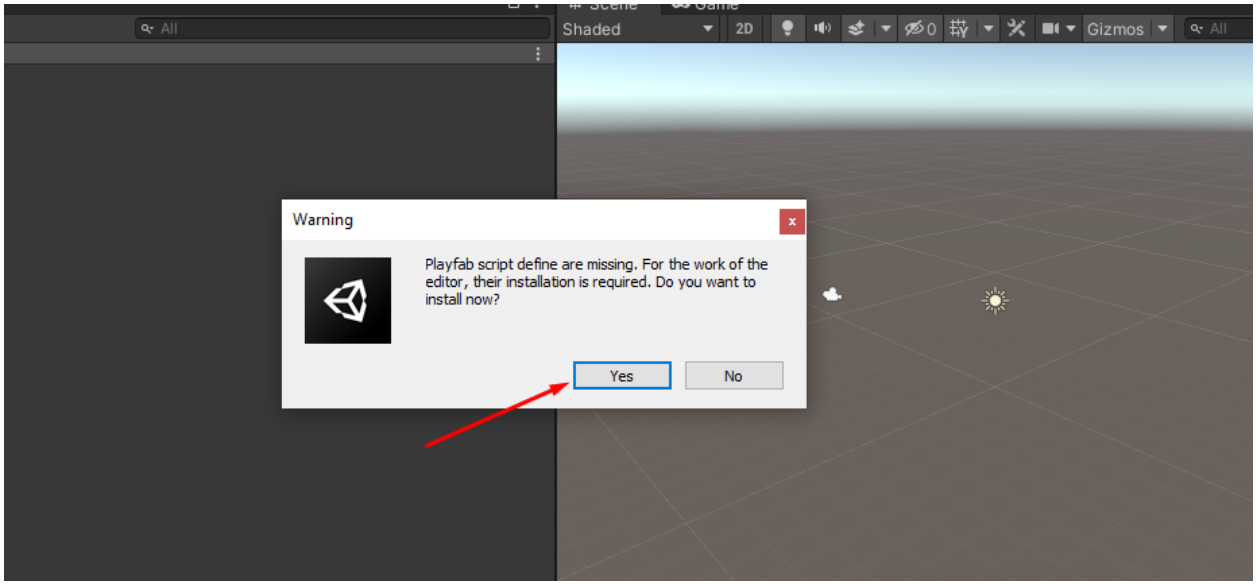
Import CBS plugin from asset-store



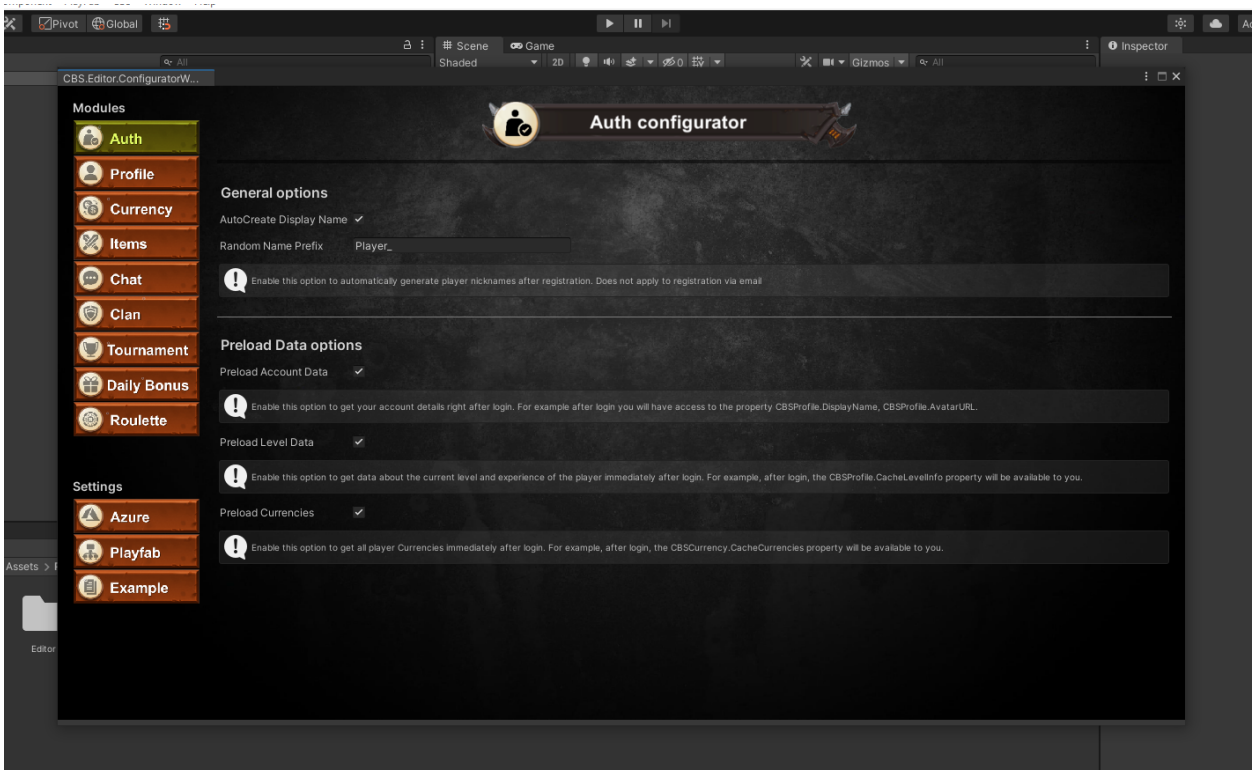
Open CBS->Configurator



The first time a Missing Script Define warning appears. Click **Yes**.



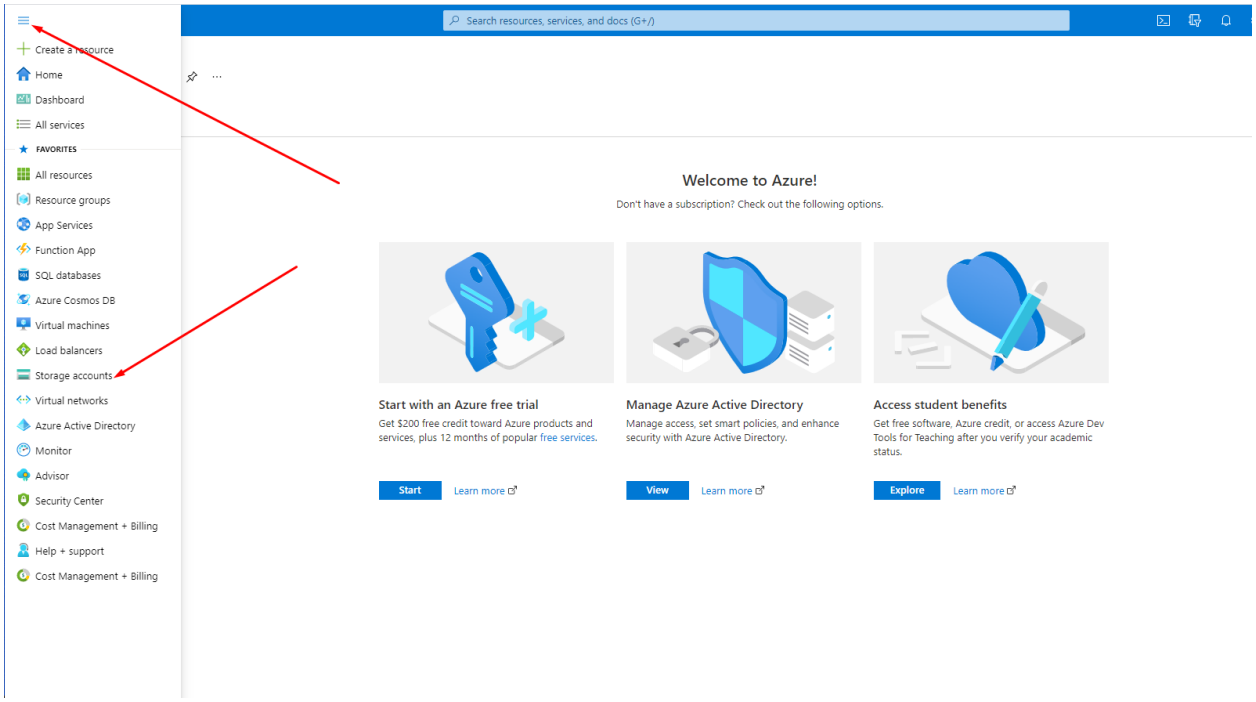
Again navigate to Open CBS->Configurator and CBS editor window will appear if everything is install correctly.



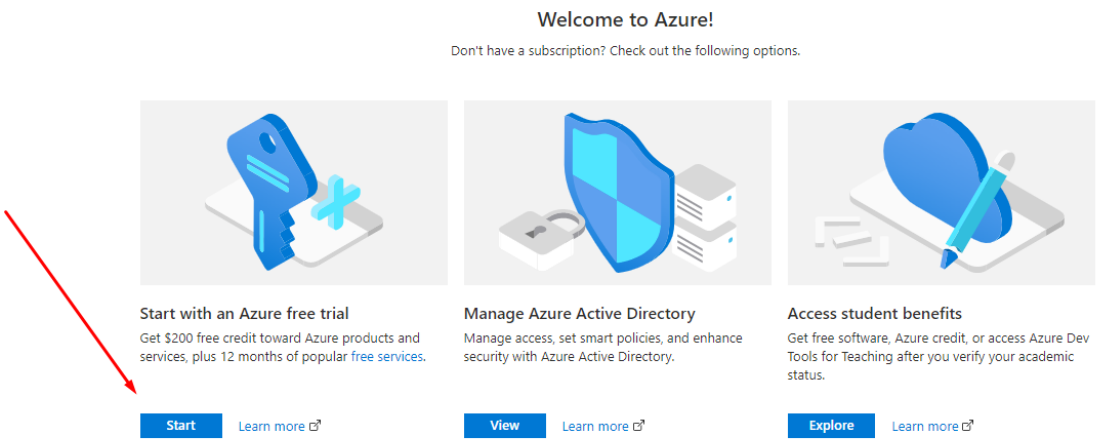
3. Azure Storage Table

First of all register on https://portal.azure.com/

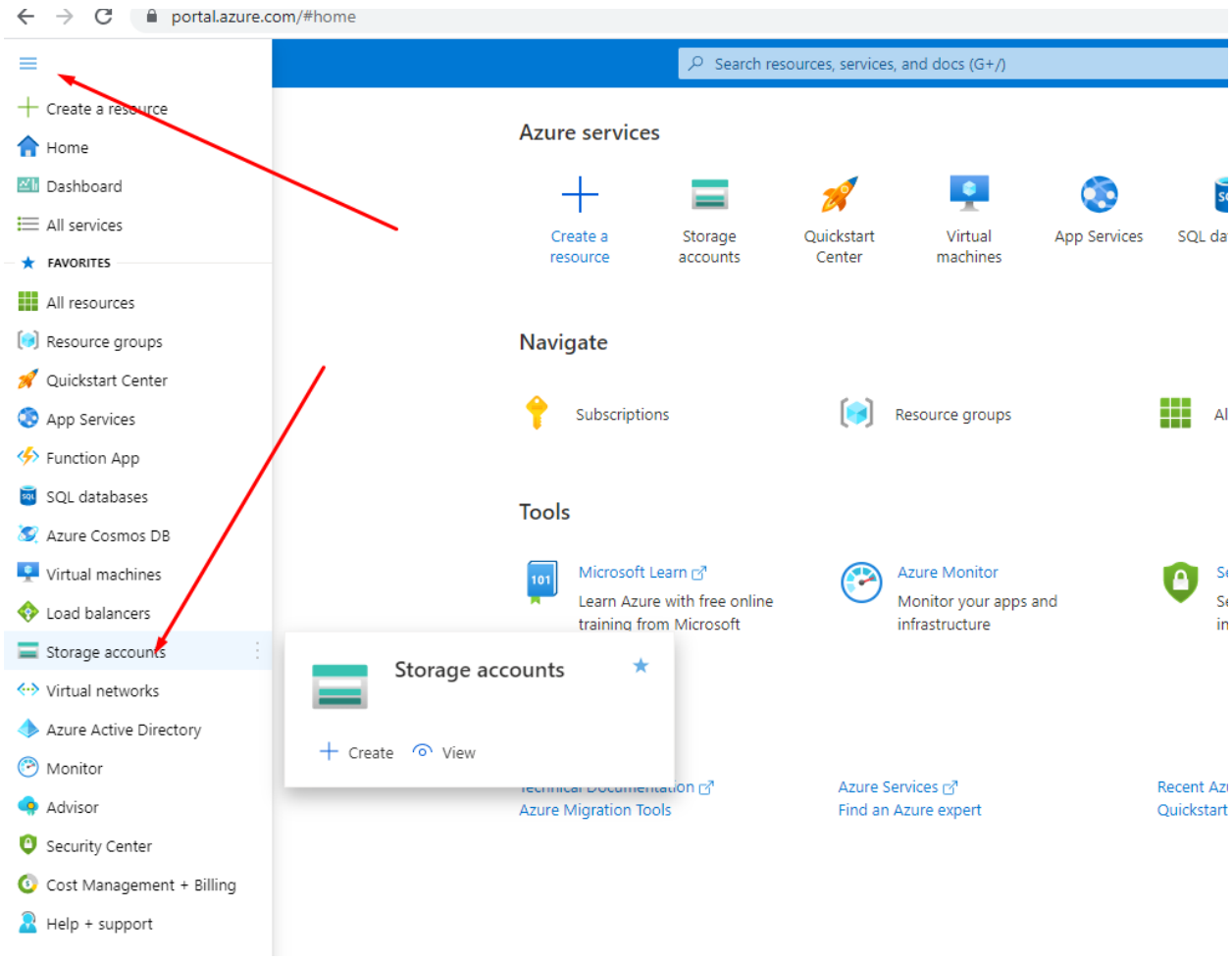
After login navigate to **storage accounts**



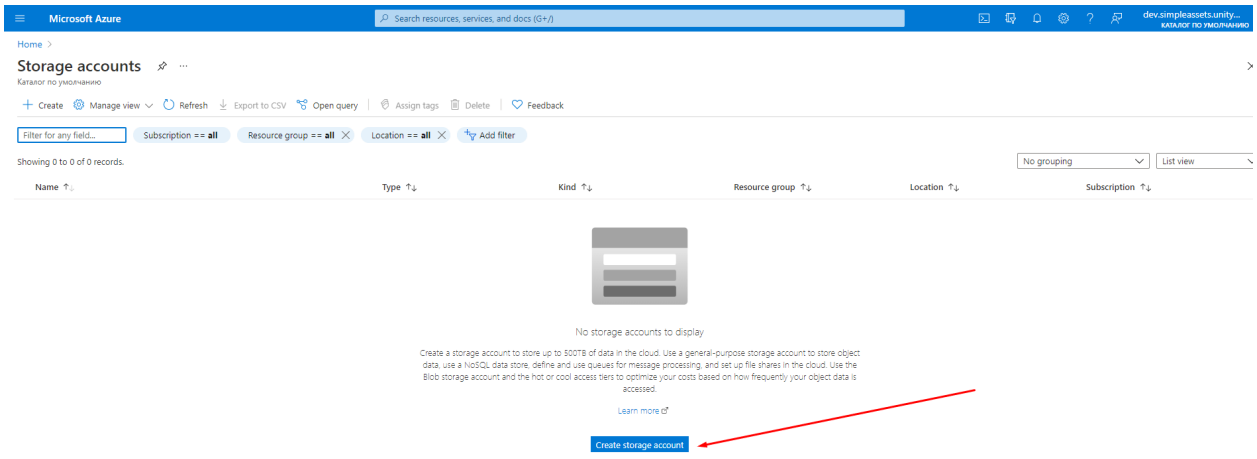
Click "Start"
Fill in all the information provided and confirm it



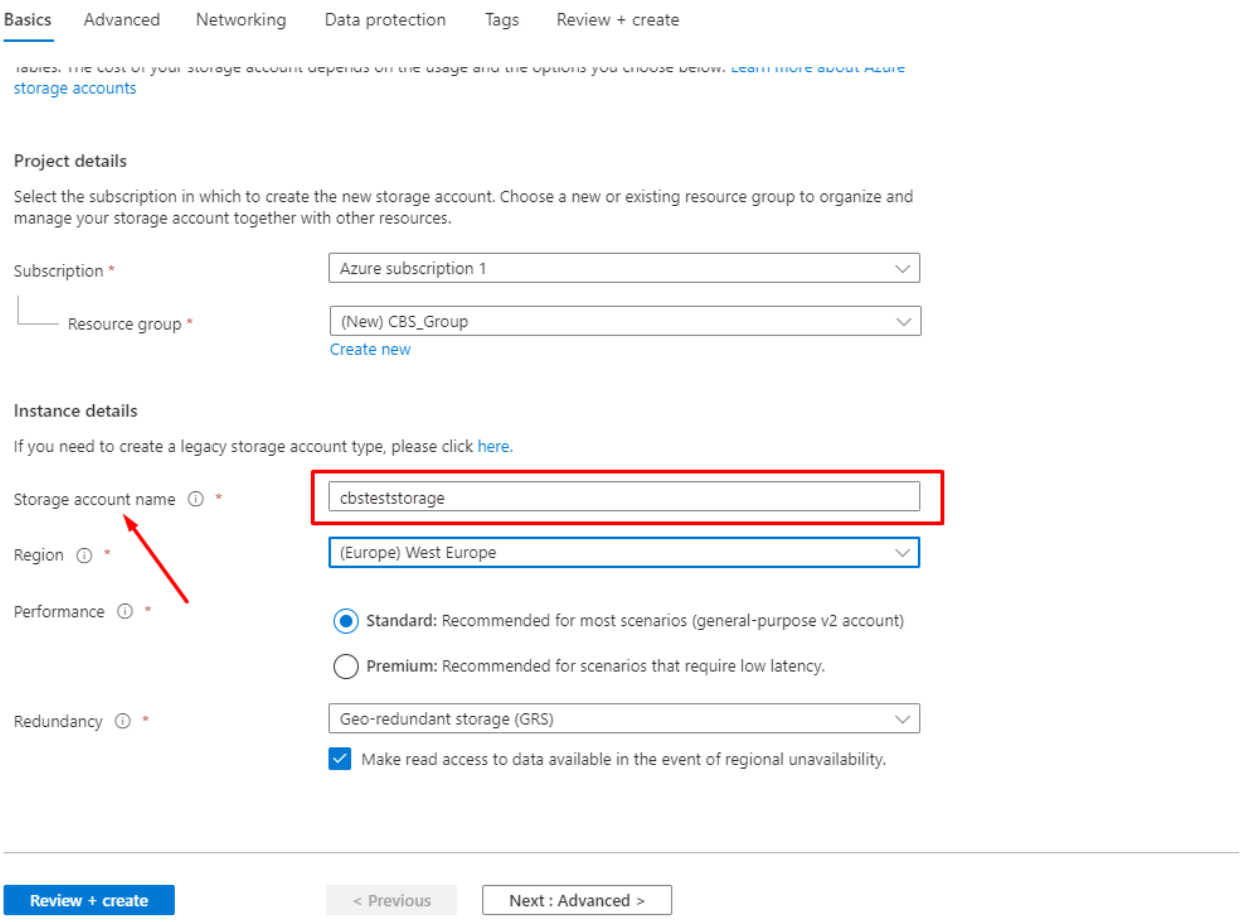
After that navigate to storage account again



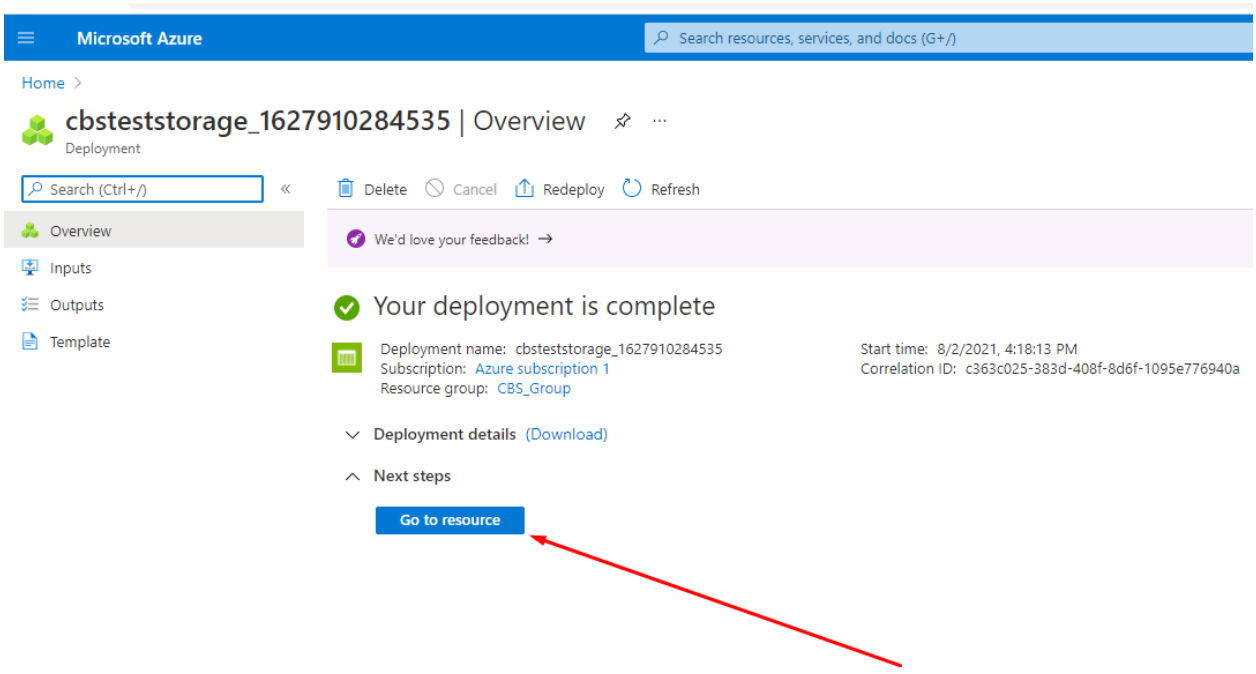
Create new storage account.



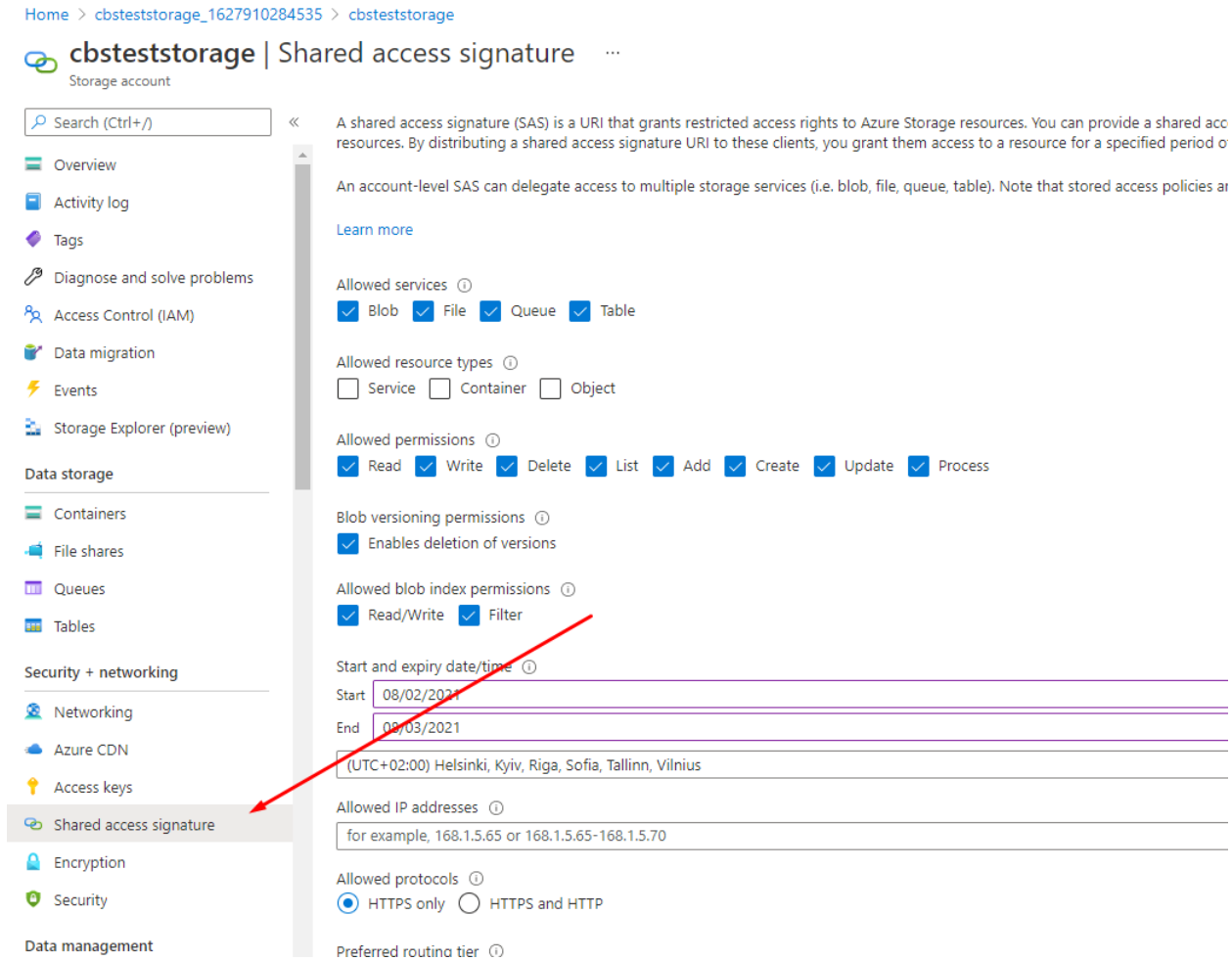
Enter Storage account name and select region and click Review+create



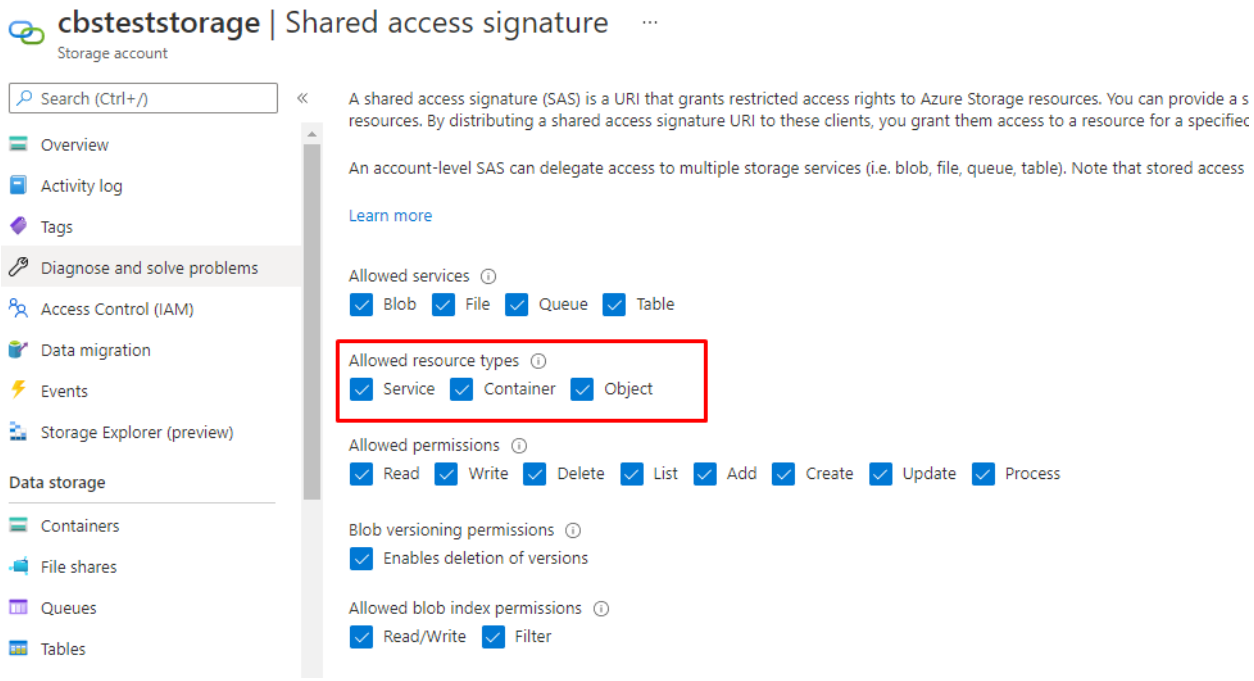
After success deployment - click "Go to resource"



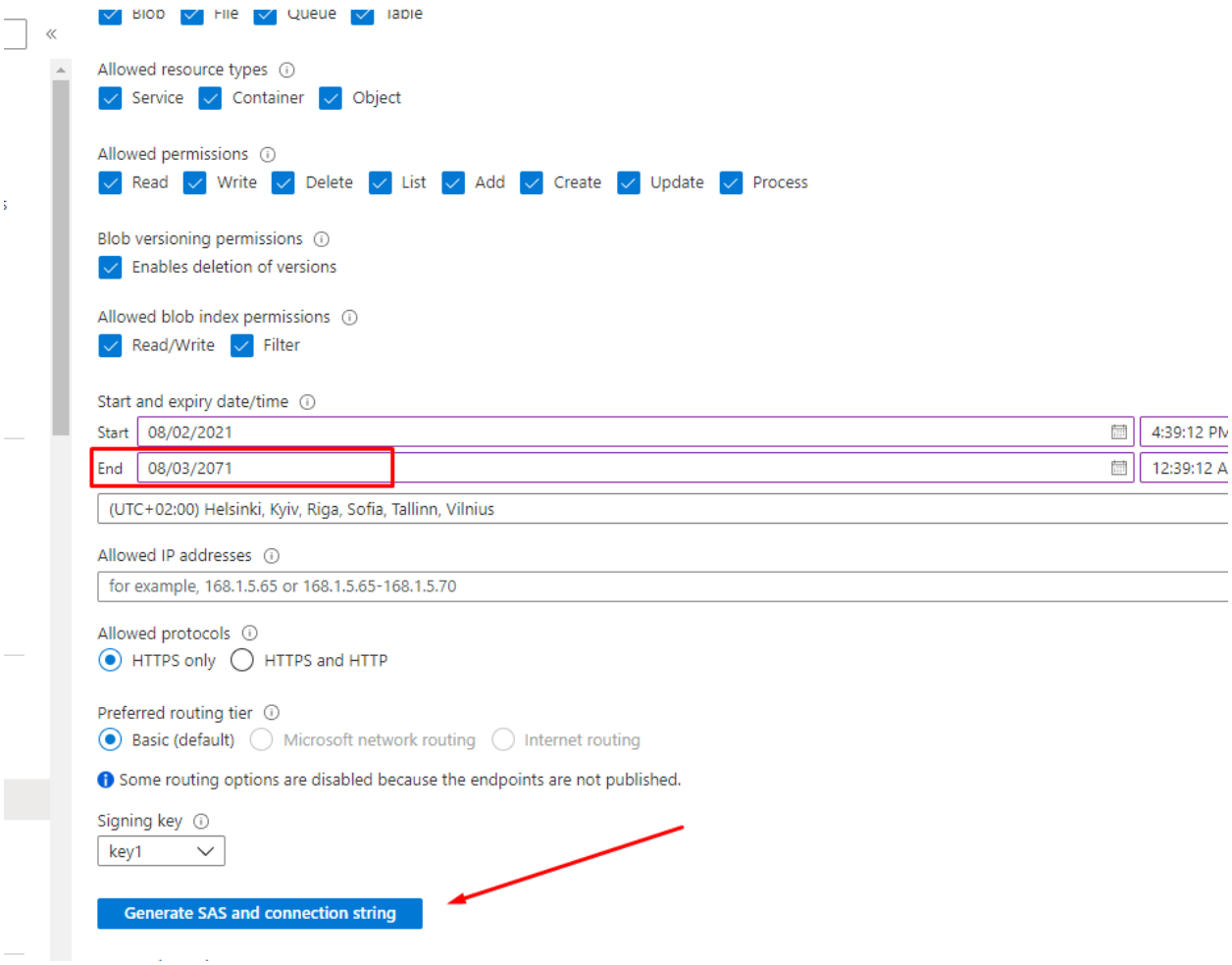
Navigate to "Shared access signature"



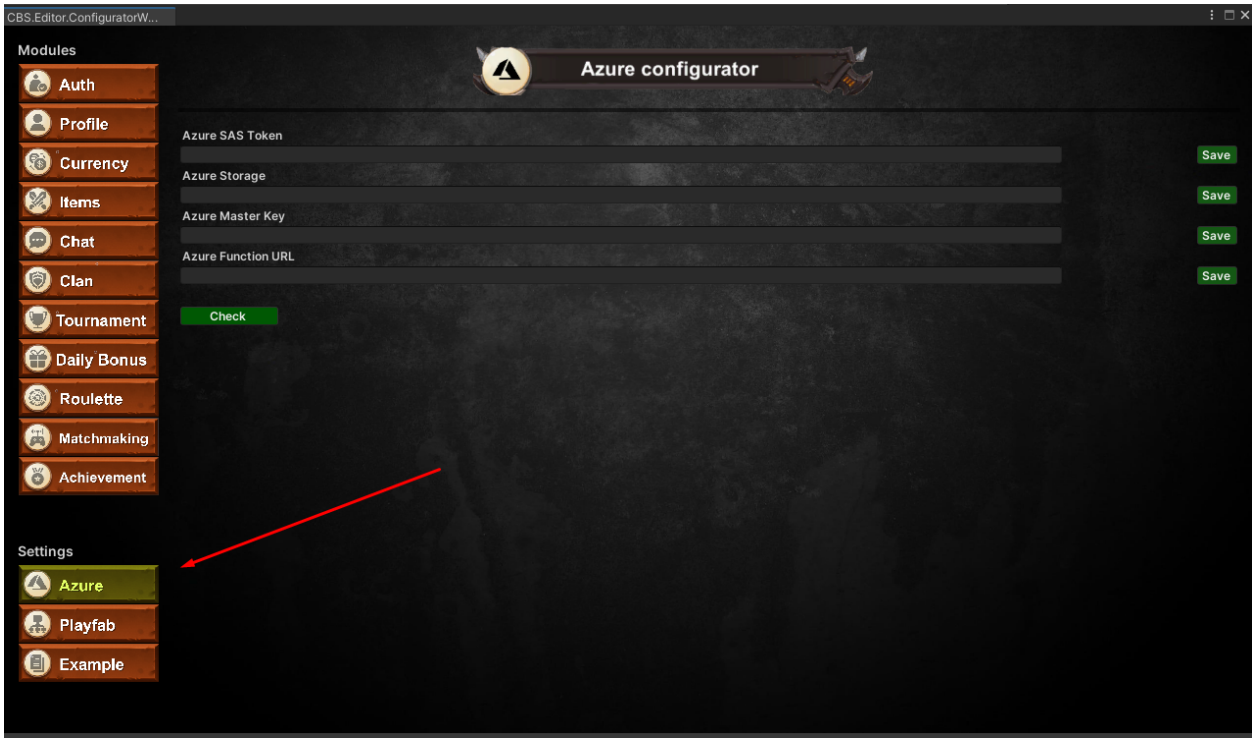
Allow all resource types



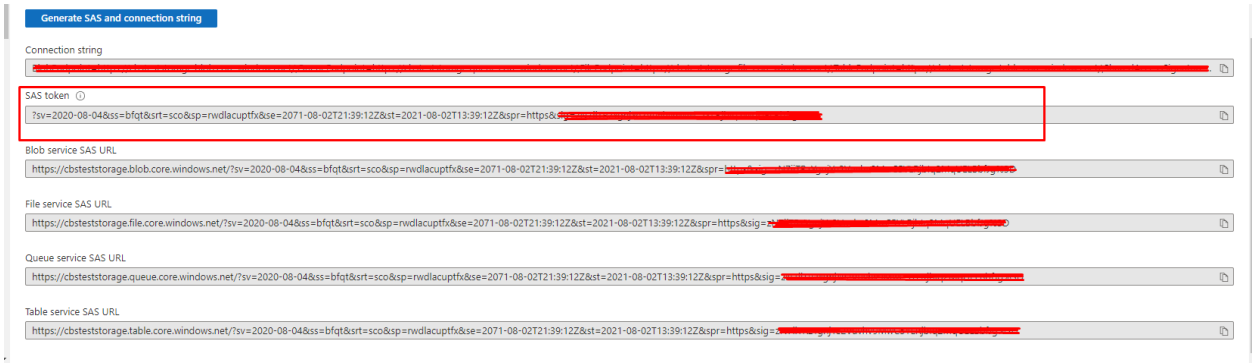
Set expiry date 50 years ahead for example. And click "Generate SAS and connection string"



Go to Unity open CBS configurator and navigate to "Azure" tab

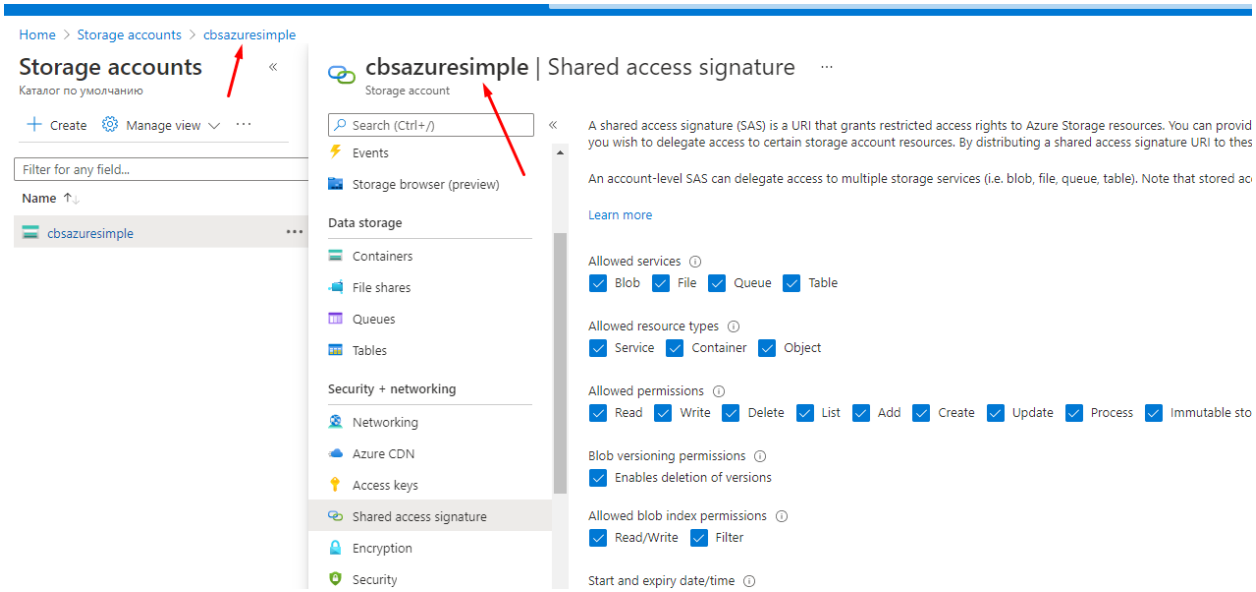


Copy SAS token

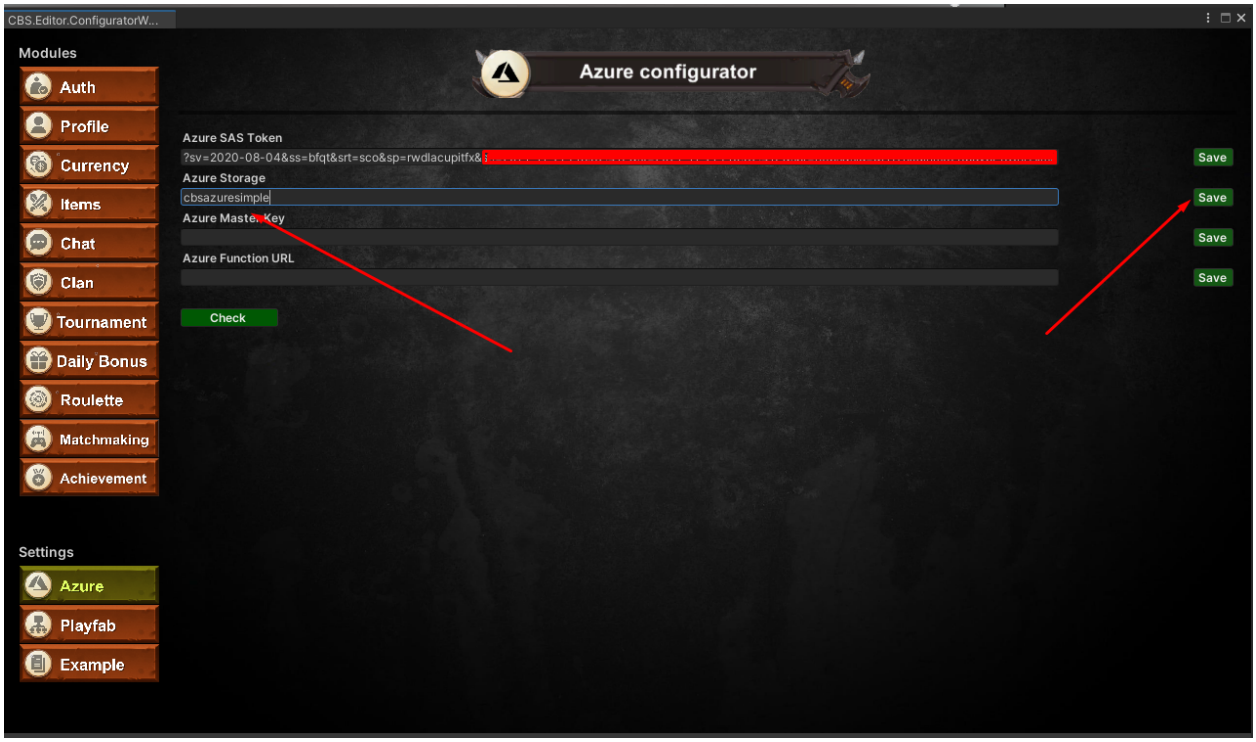


And paste into Azure SAS Token field. Click save.

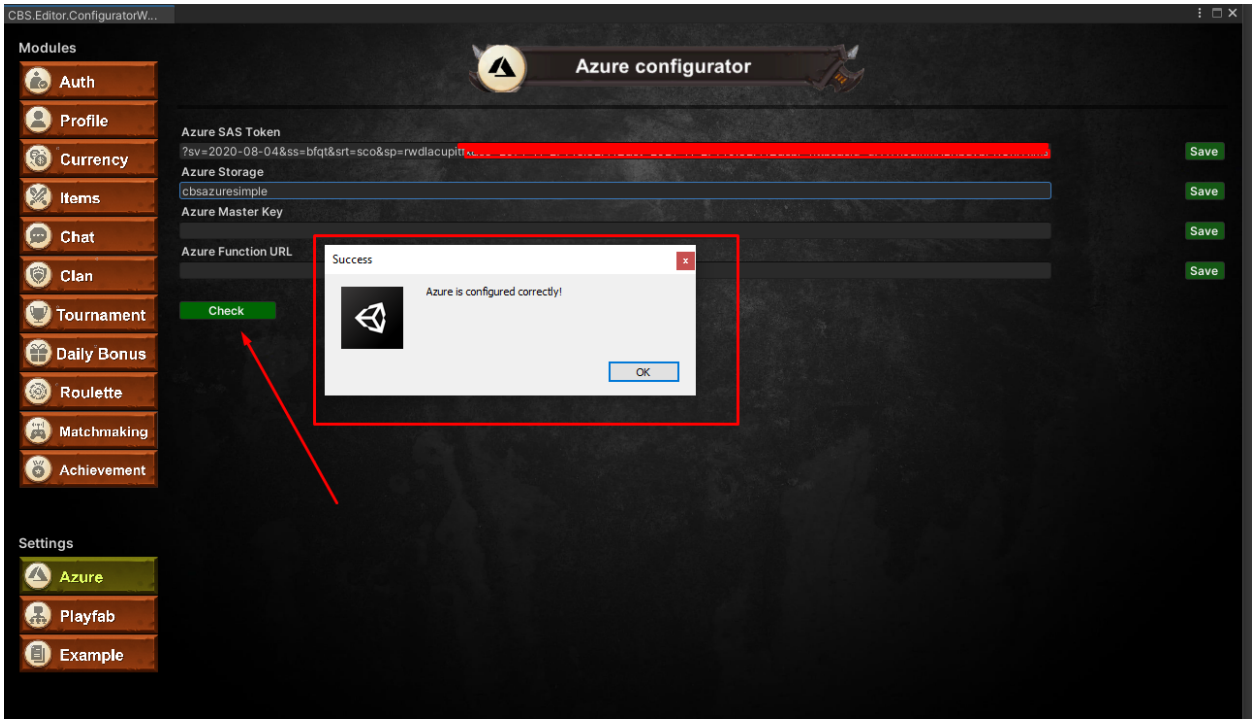
Copy azure storage account name



And paste into Azure Storage field. Click save.



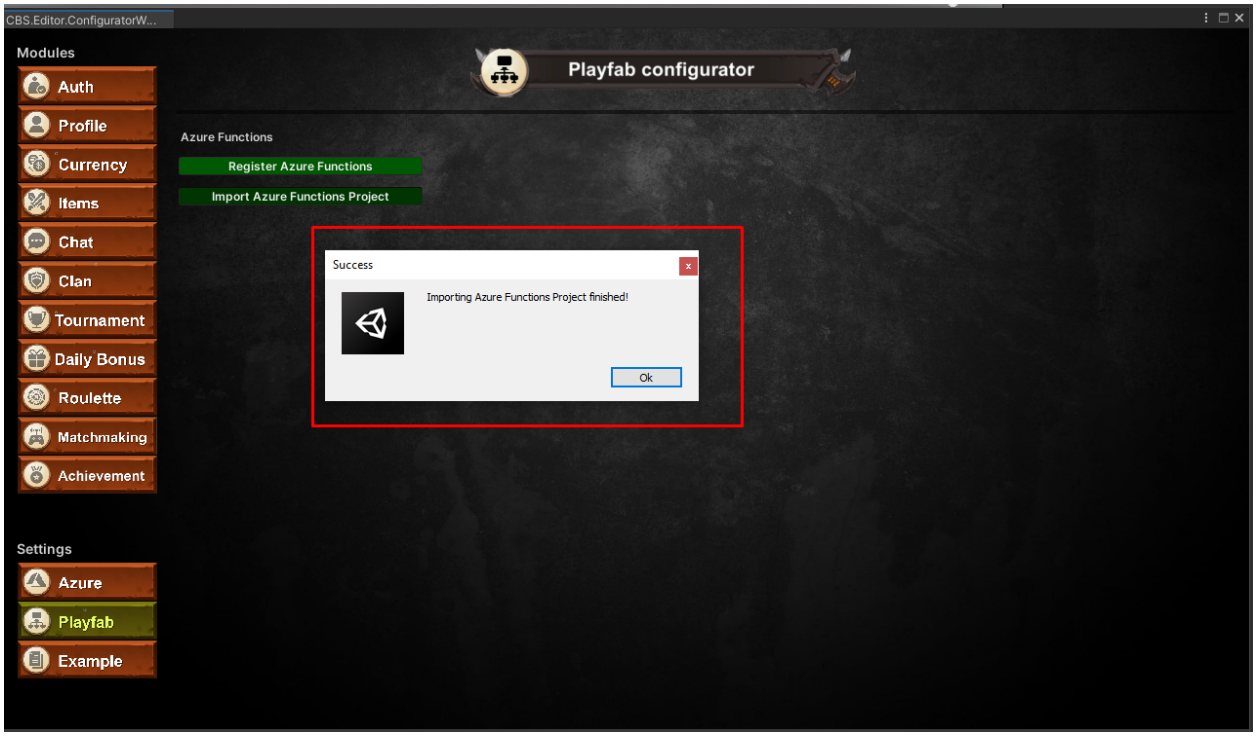
Click "Check" and you will receive a message about successful installation



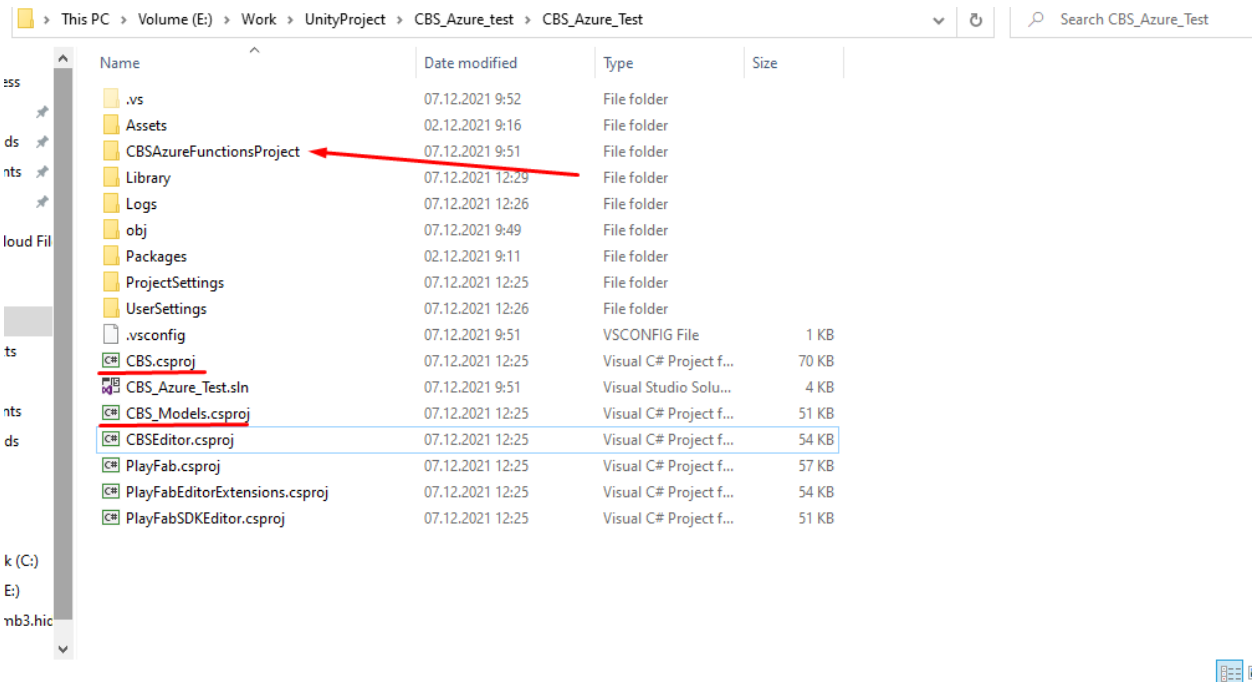
4. Upload Azure functions.

Navigate to "Playfab" tab and click "Import Azure Functions Project".

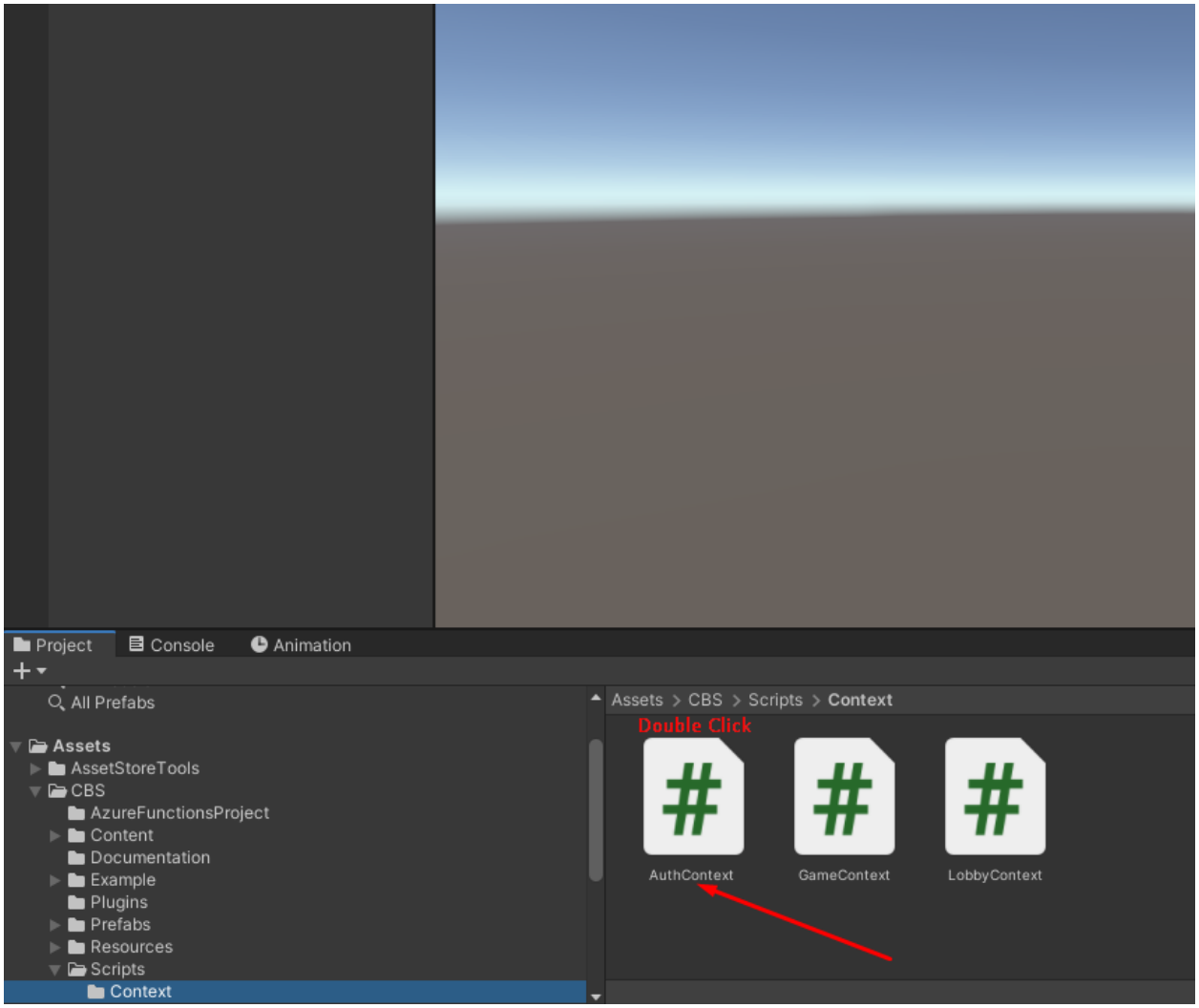
You will receive a message about successful import. This way you will notice that you will have Visual Studio open. It is necessary that VS generated automatically .csproj files for us



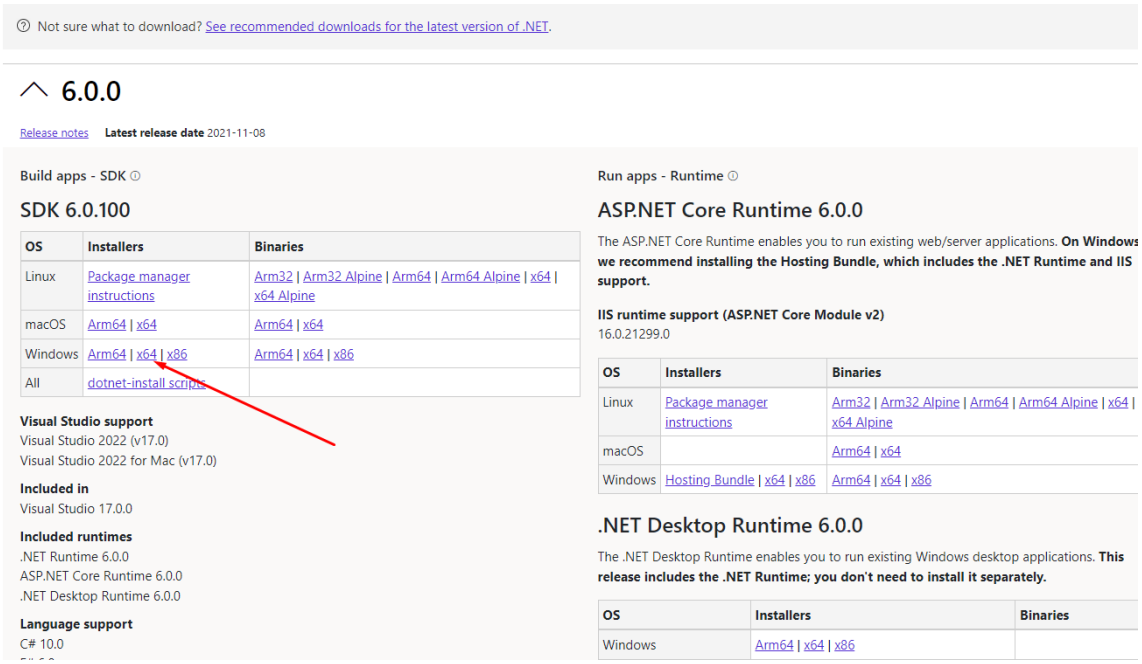
At the root of the project, you should see the **CBSAzureFunctionsProject** folder. This is a vscode server-side source project. Also make sure you have **CBS.csproj** and **CBS_Models.csproj** files



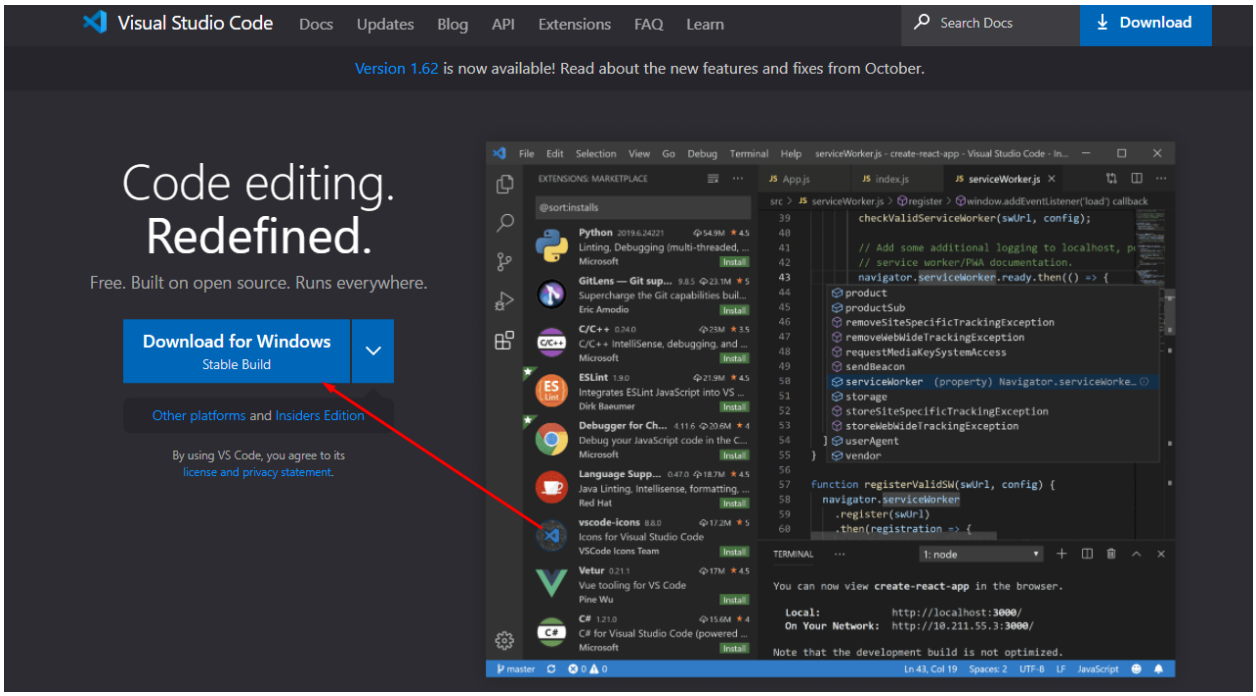
If the files **CBS.csproj** and **CBS_Models.csproj** were not generated, open any CBS script (just double-click to open any CBS script in Unity). This will cause Visual Studio to invoke and the CBS.csproj and CBS_Models.csproj files are automatically generated.



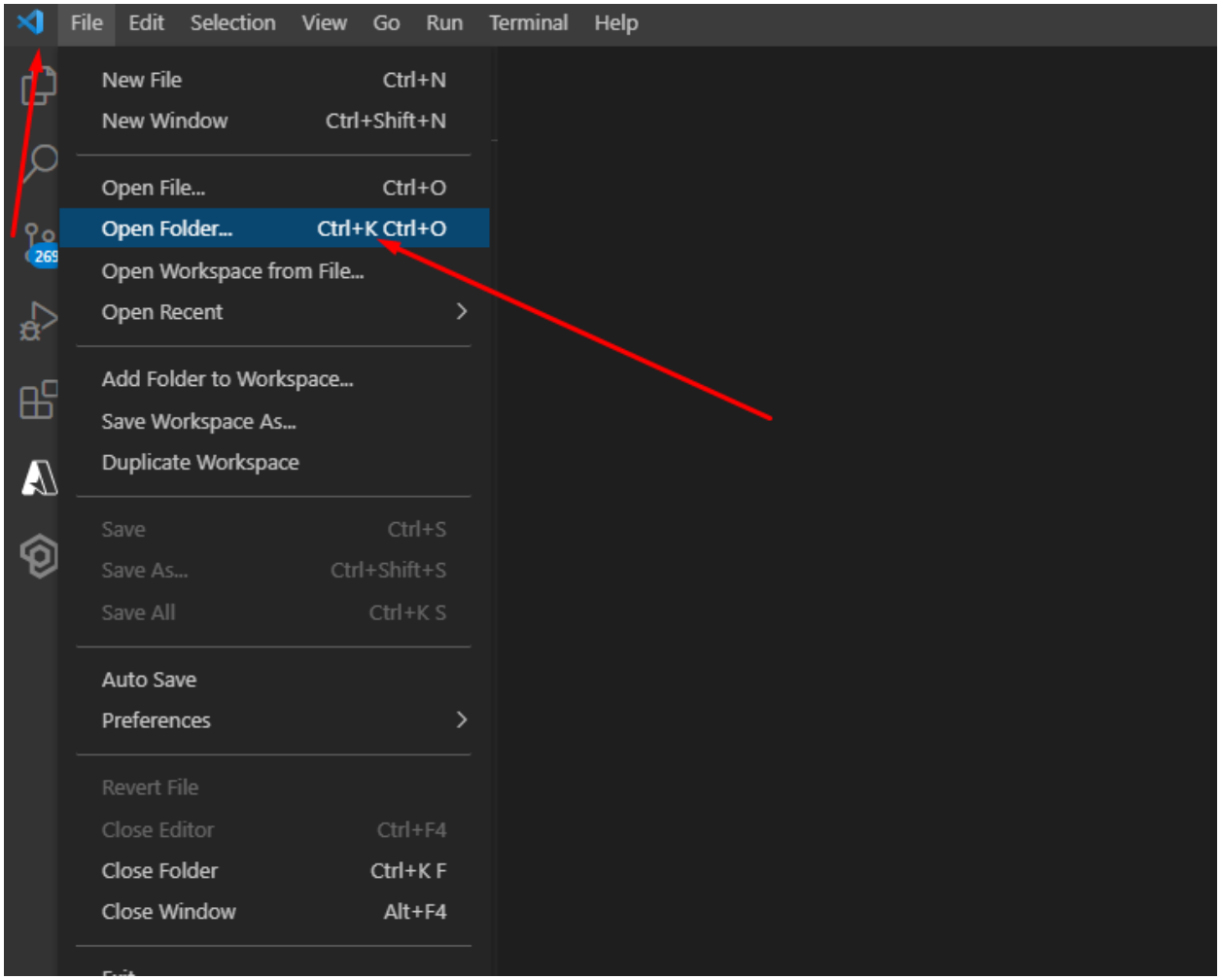
To upload server-side code to Azure, we need the **.NET 6** and **Visual Studio Code**. First, let's install .NET 6.
Download the .NET 6 **installer** and install it on your computer. Select the installer suitable for your processor architecture (x64 in my case)
https://dotnet.microsoft.com/download/dotnet/6.0



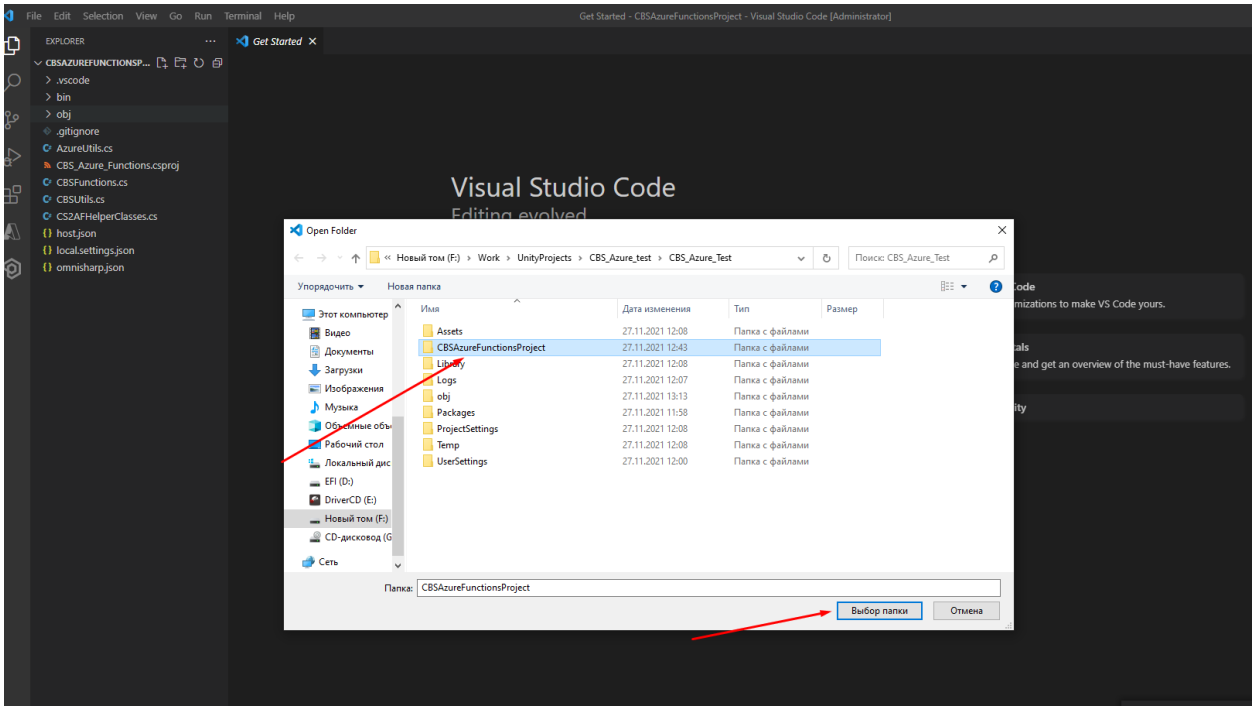
Next, install Visual Studio Code. You can download it here
https://code.visualstudio.com/



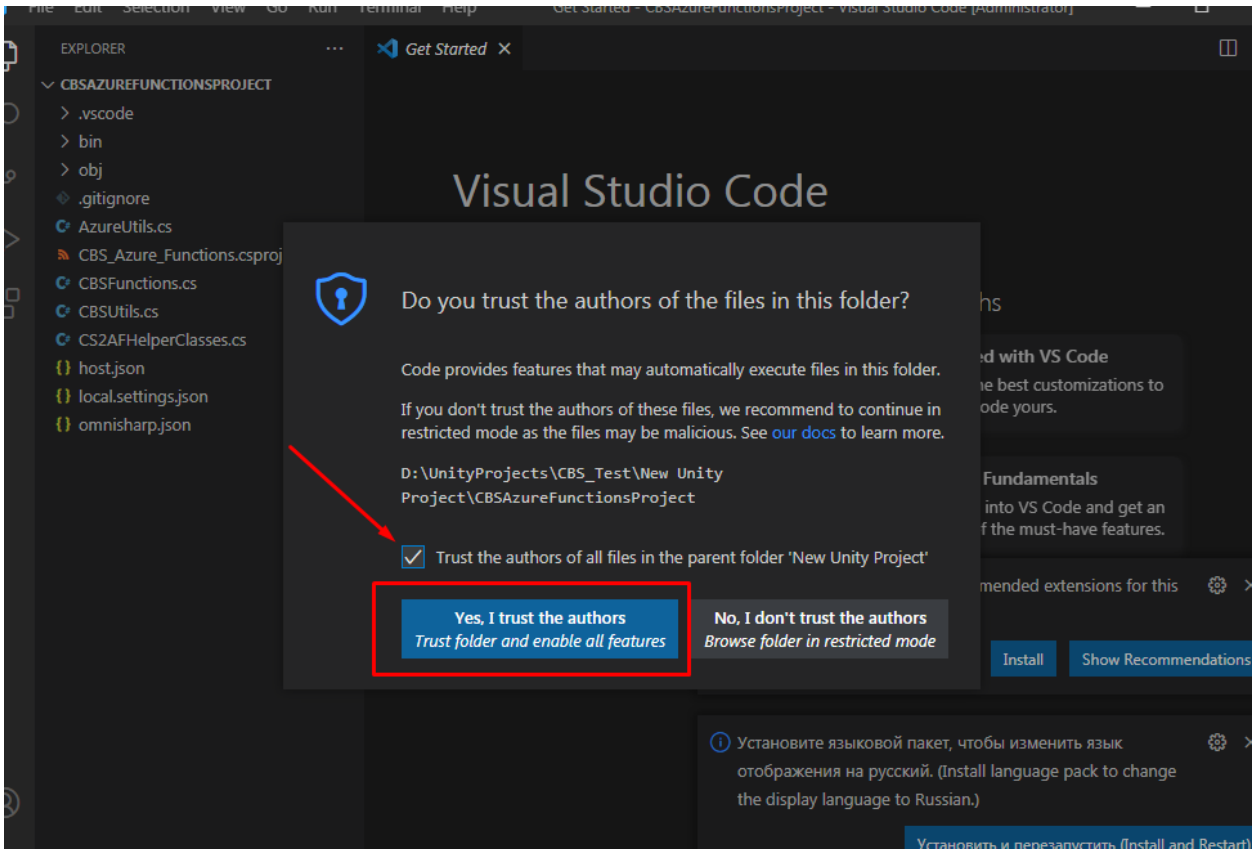
Open VS Code and click "File - Open Folder"



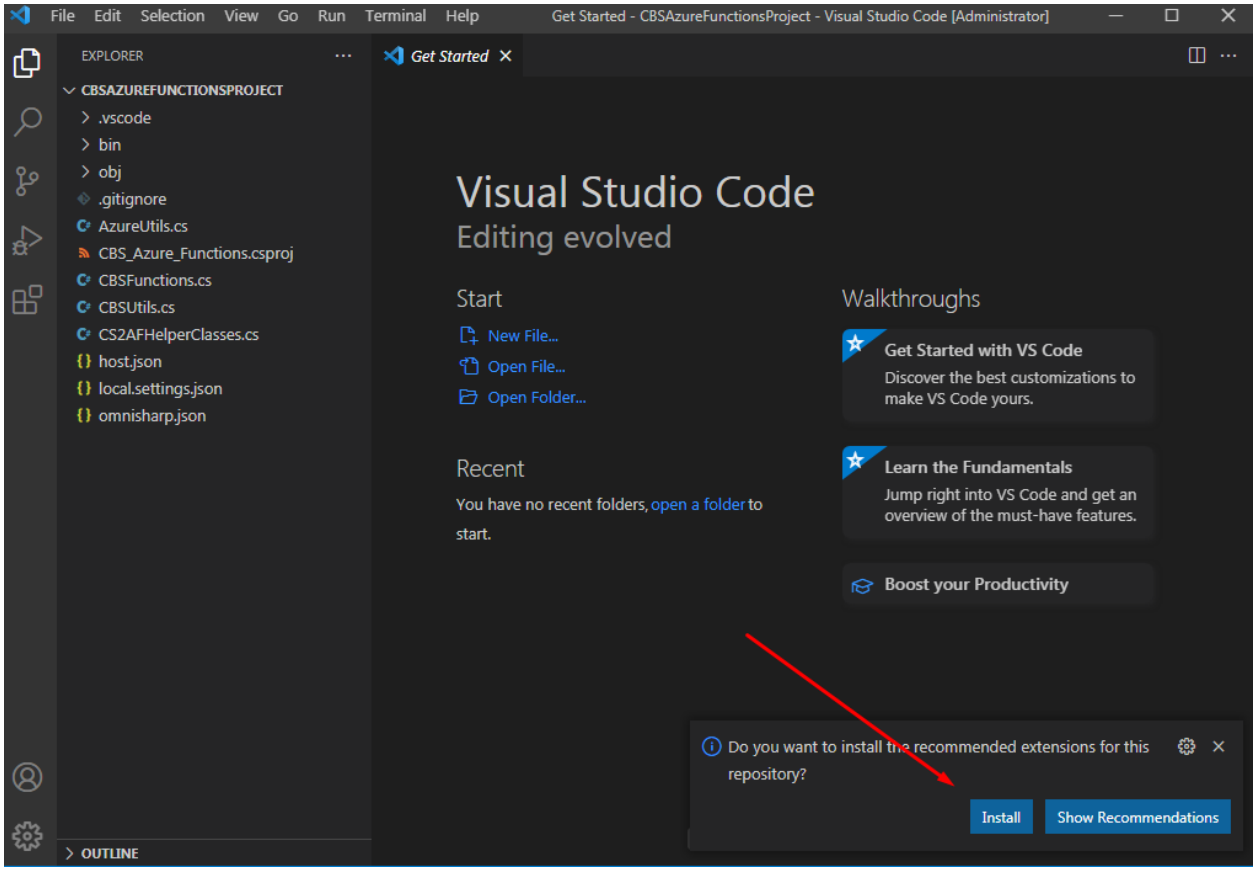
Select the folder of the previously loaded project "CBSAzureFunctionsProject"



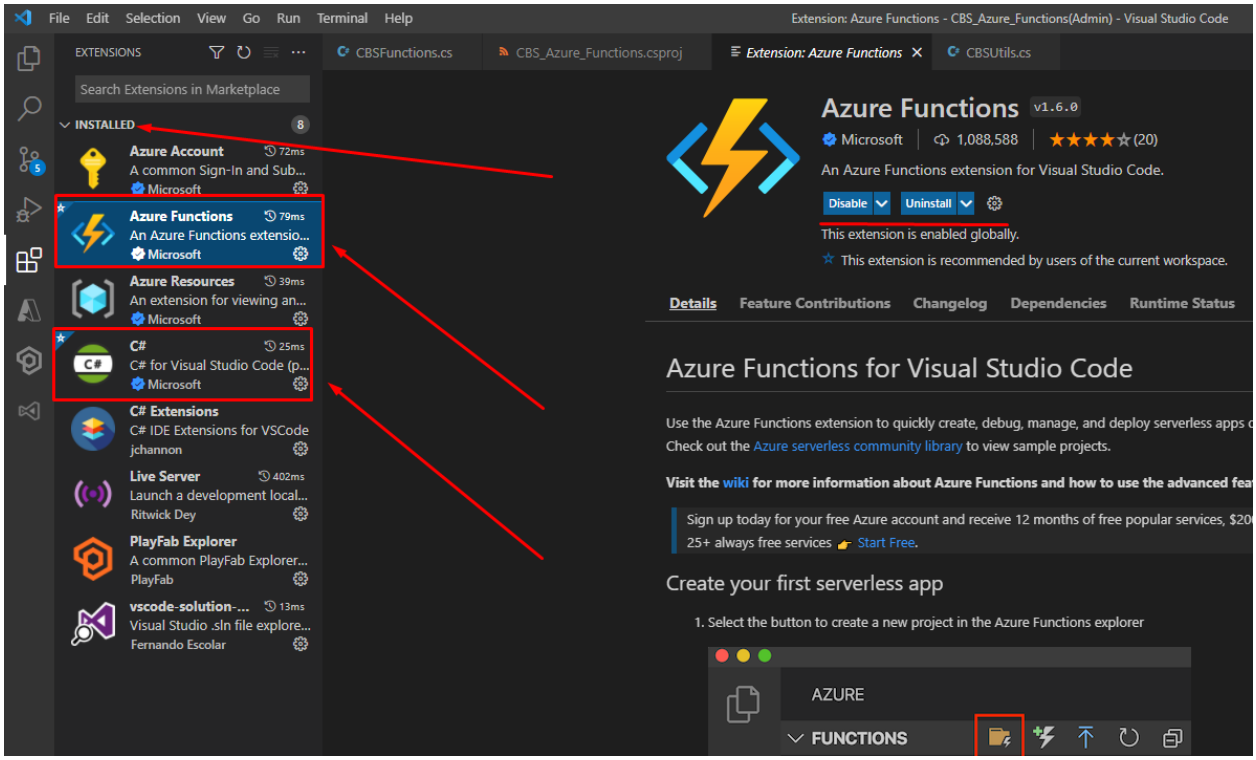
You may have a dialog box that says "**Do you trus the author of the files of this folder?**" Apply check box "**Trust the authors**" and click "**Yes I trust the authors**"



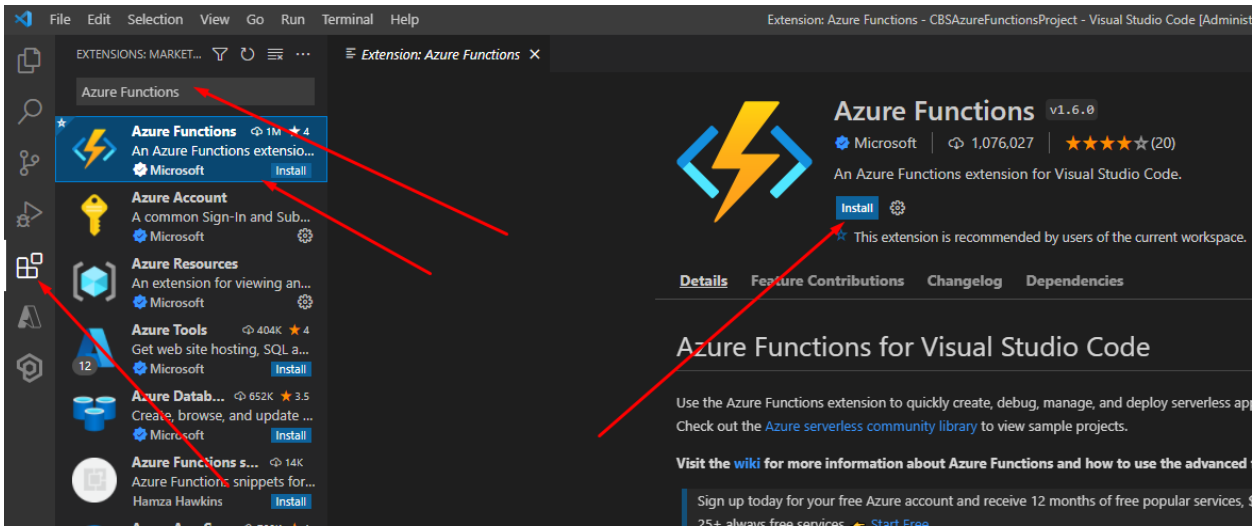
You may have a dialog box that says "**Do you want to install the recommended externsions for this repository?**" Click **Install**



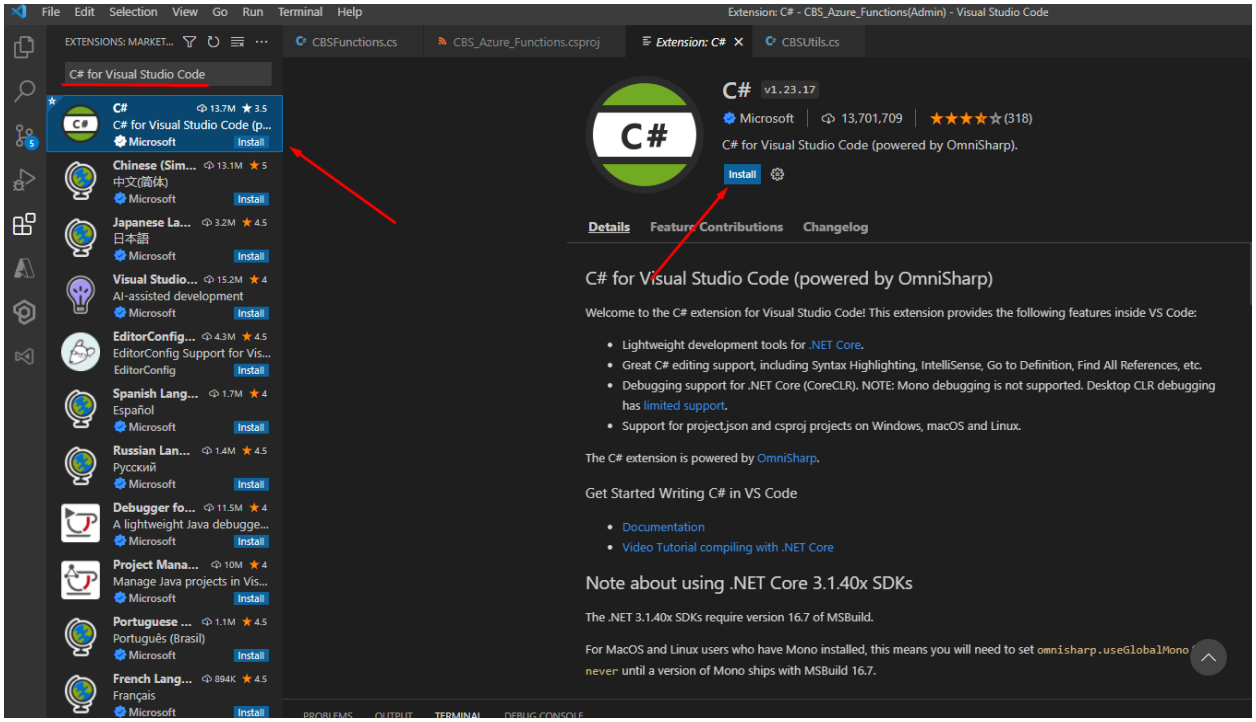
Make sure you have extensions like "**Azure Functions**" and "**C # for Visual Studio Code**" installed.



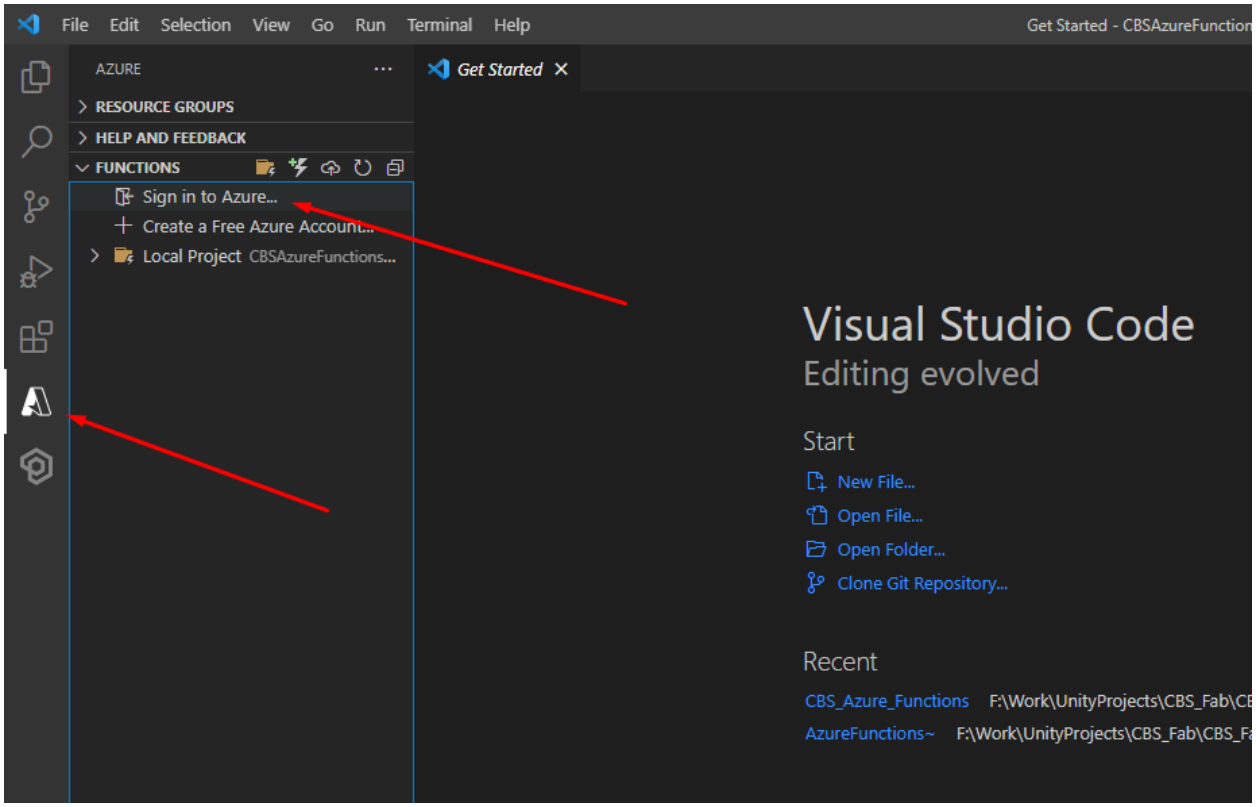
If these extensions are not installed, when you need to install them manually. Navigate to "Extensions" tab. Search "Azure Functions" extension and click "Install"



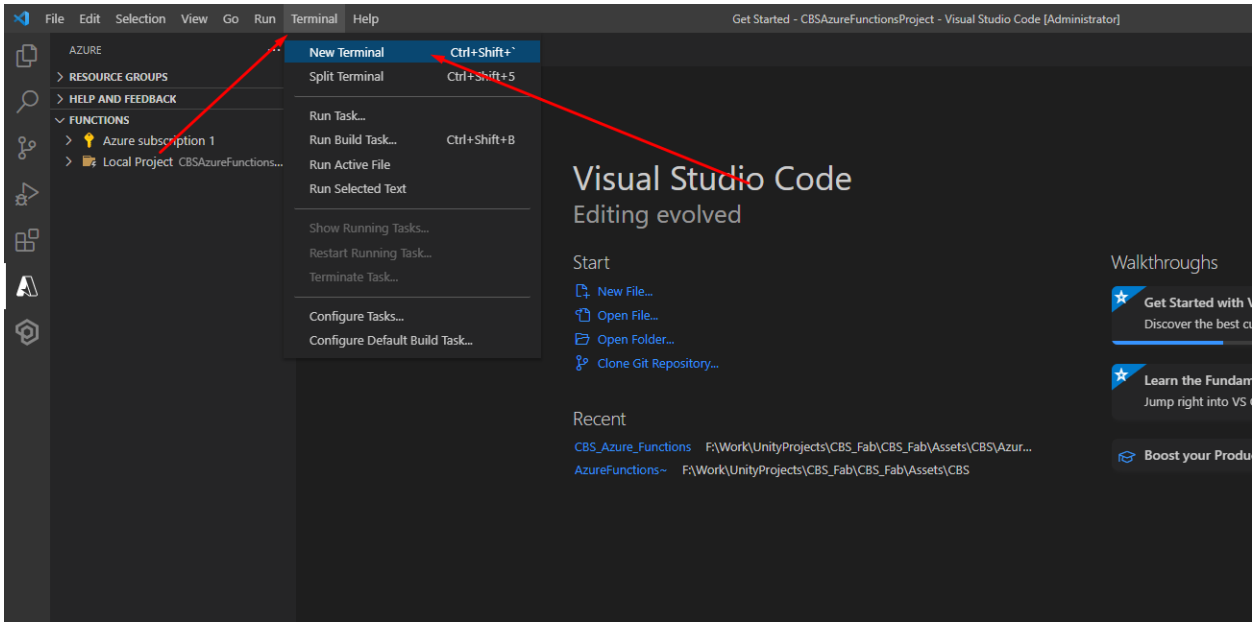
Do the same with **C# for Visual Studio Code** if it not exist.



Navigate to Azure tab and sign in using login password

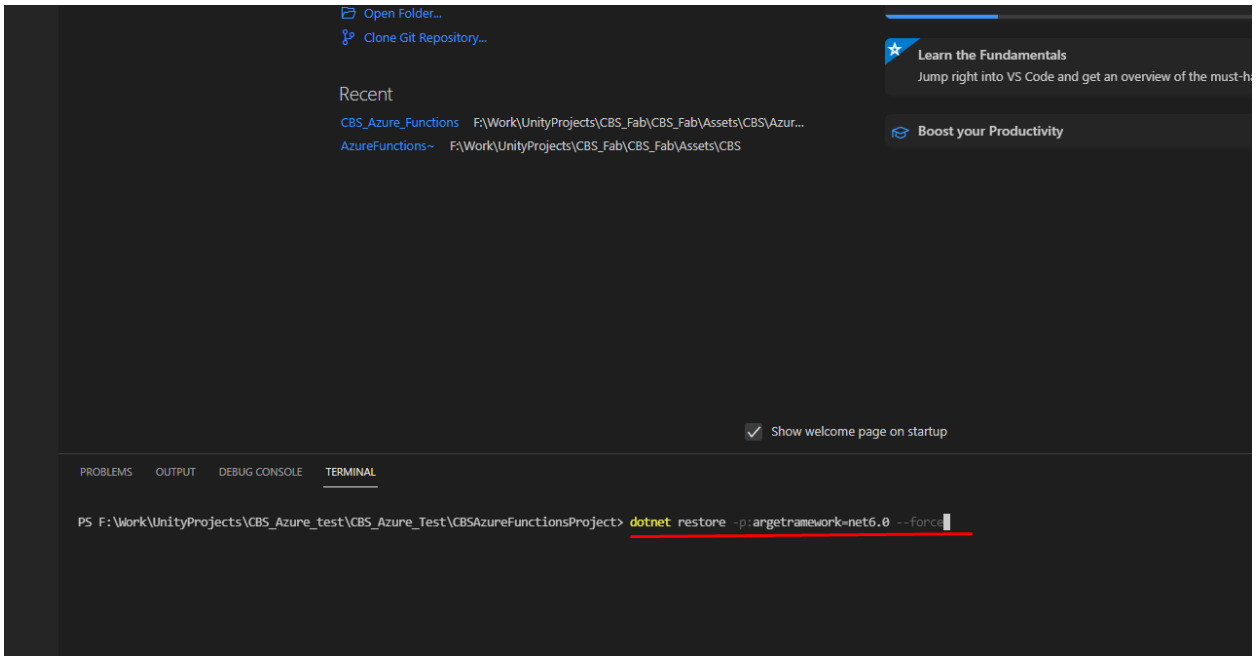


After successful login - click Terminal -> New Terminal

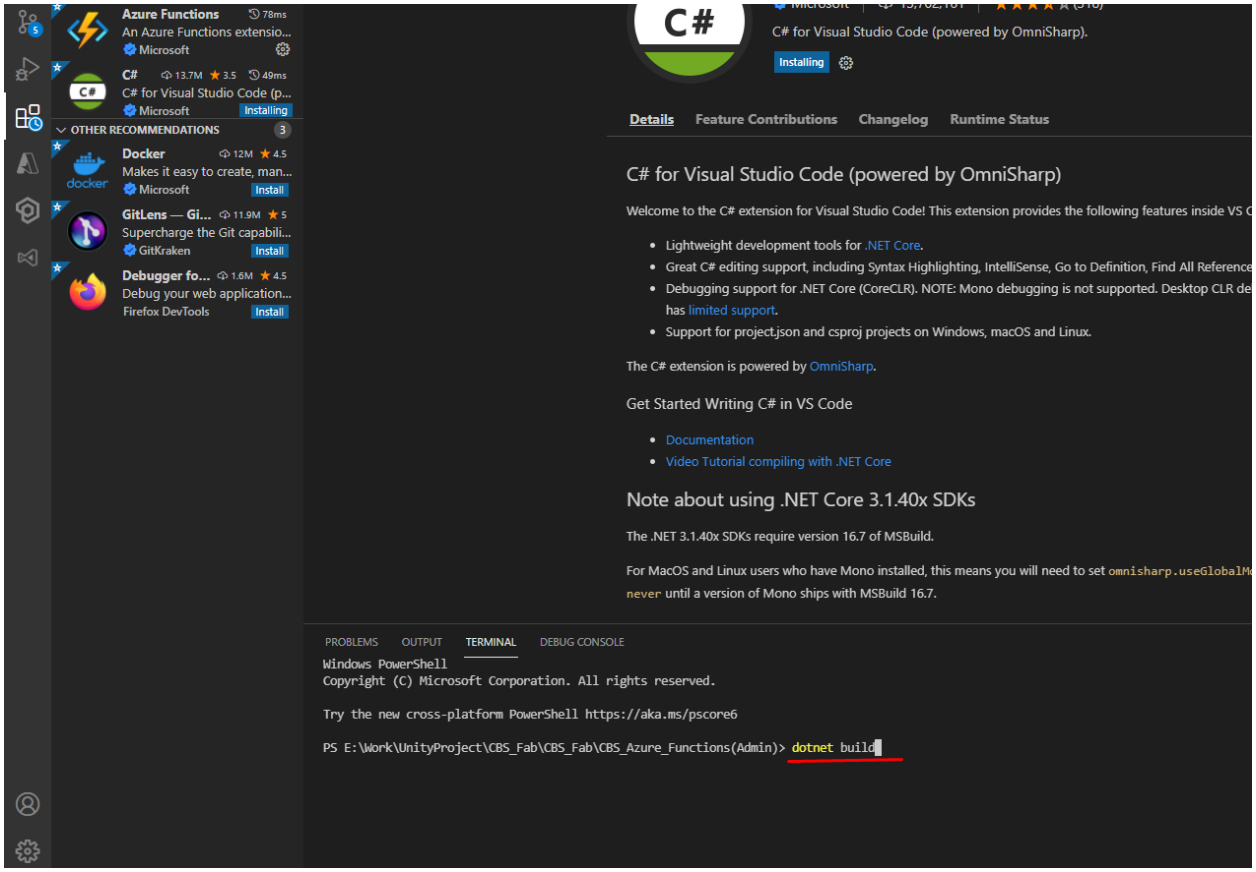


In a terminal window enter the command **dotnet restore -p: TargetFramework = net6.0 --force** and press Enter

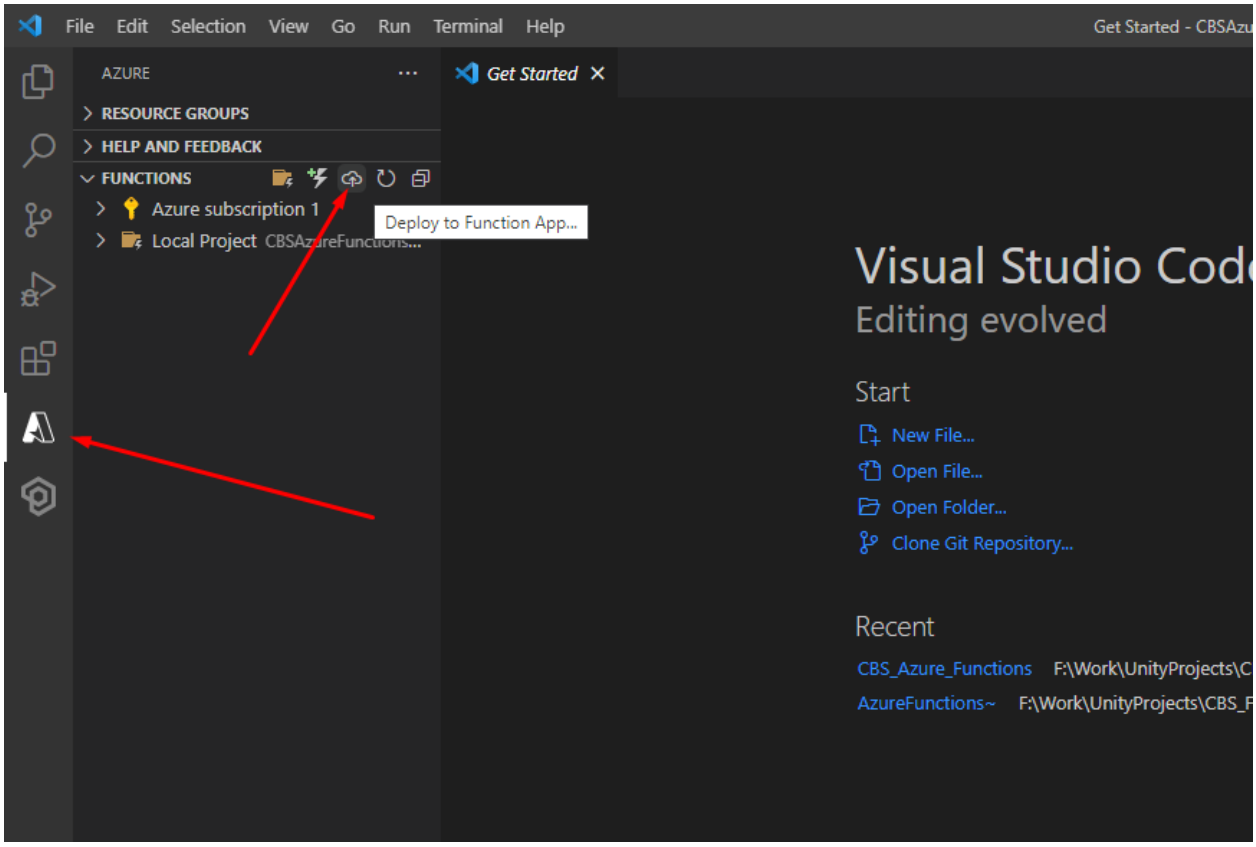
Warning. If you get an error like **"dotnet: The term 'dotnet' is not recognized as the name of a cmdlet, function, script file, or operable program"** most likely dotnet was not automatically added to the system's Environment Variables.
To fix this, watch the video on how to add dotnet to Environment Variables
https://youtu.be/jSvL_Rnm_L8
After that restart VS Code and repeat the step again



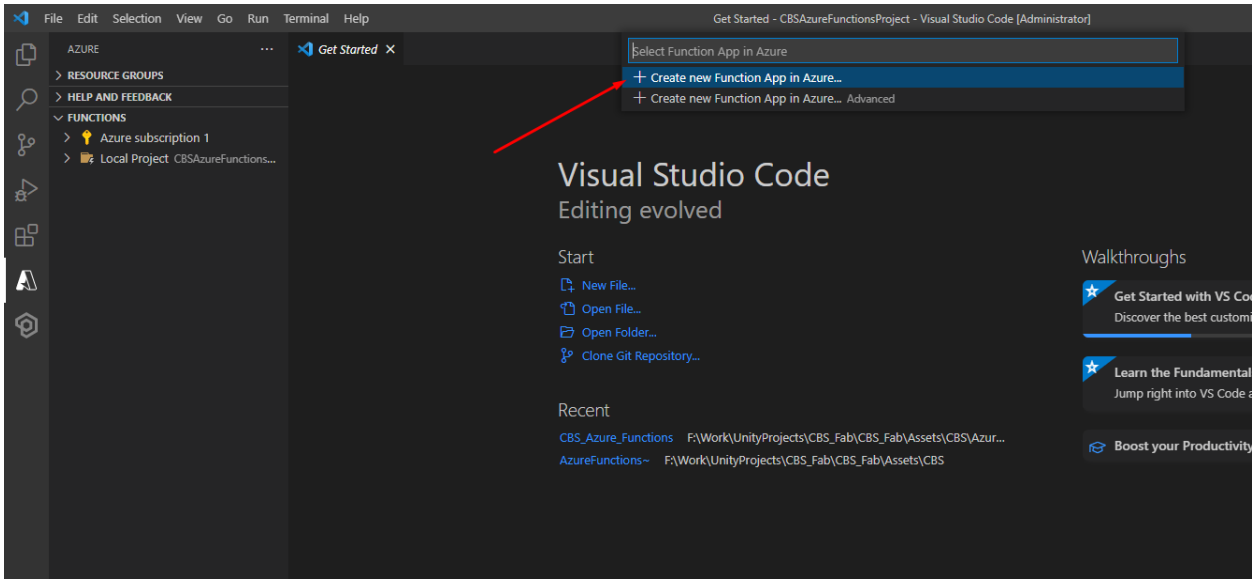
Also run **dotnet build** command



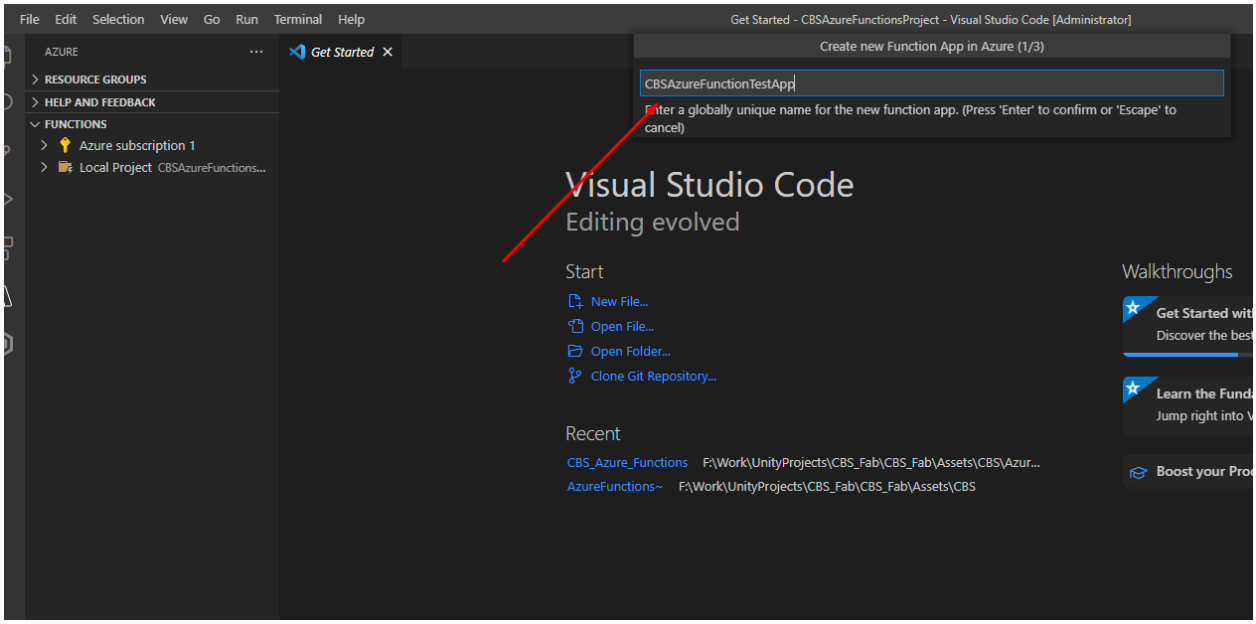
After succes build - let's upload the functions to Azure. In the "Azure" tab - click "Deploy To Function App"



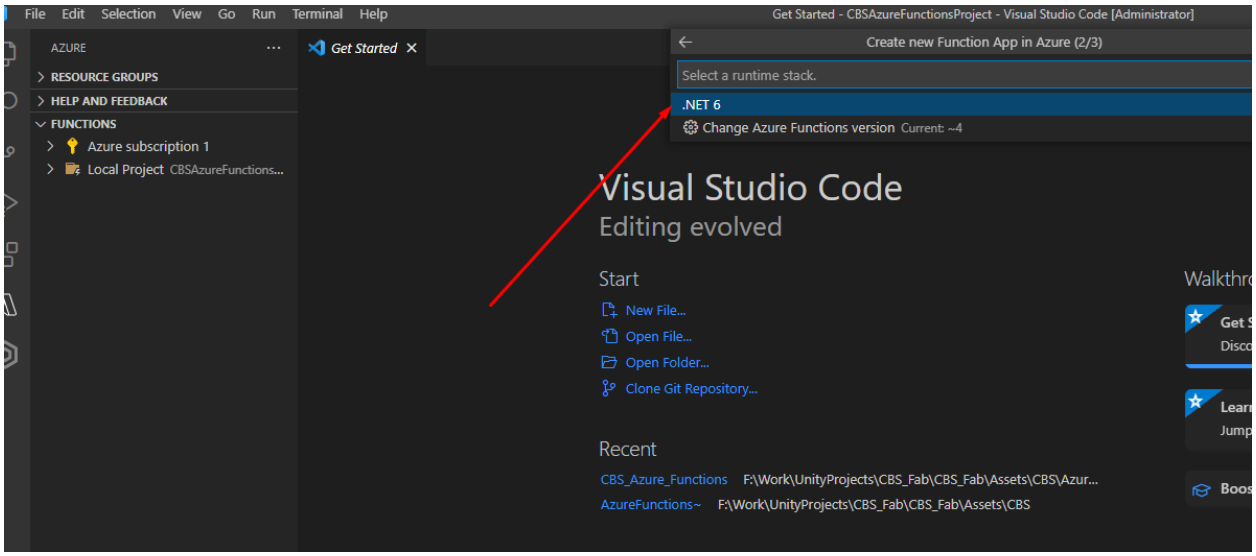
Click "Create new Function App in Azure"



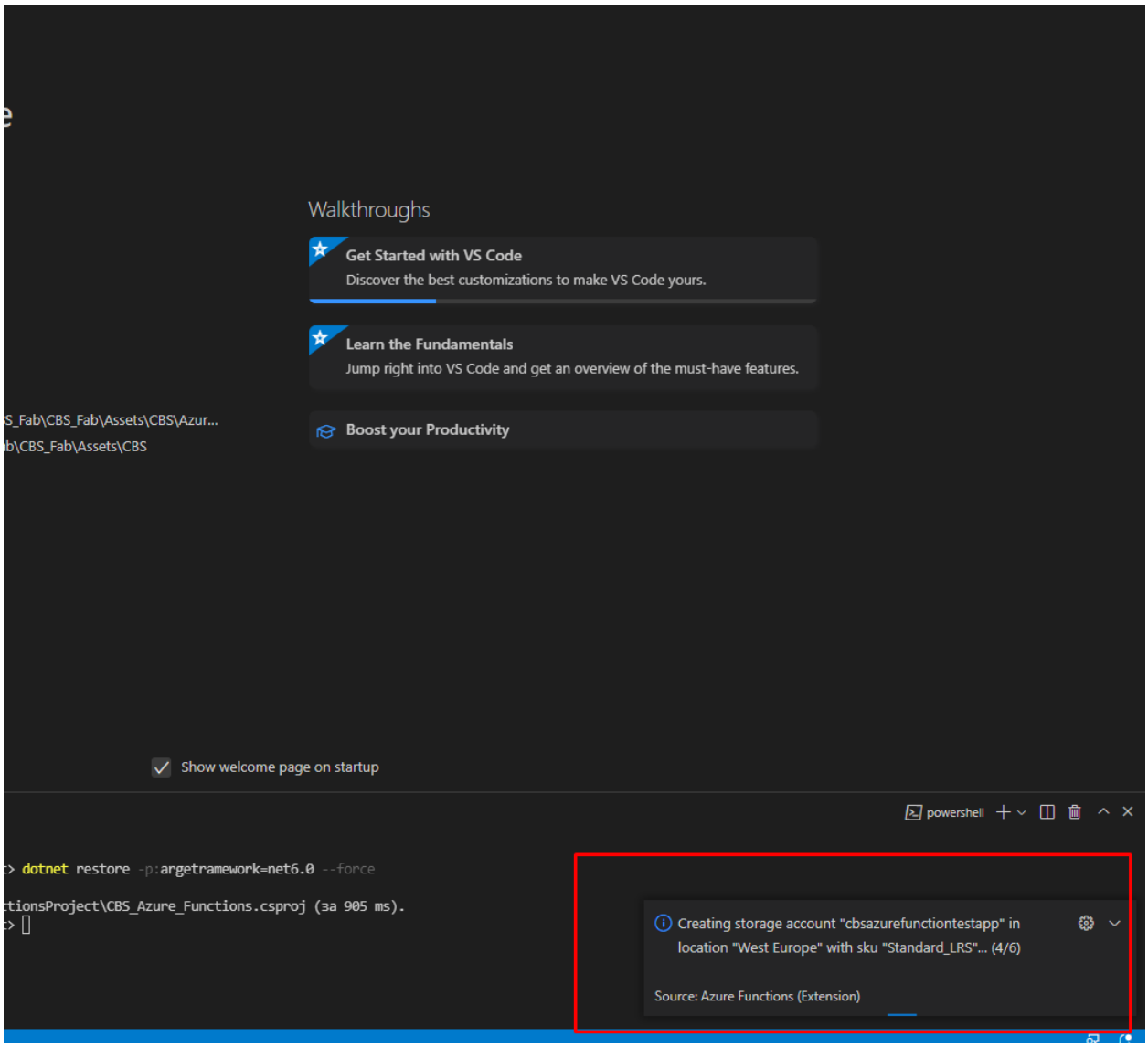
Enter a unique name for the app



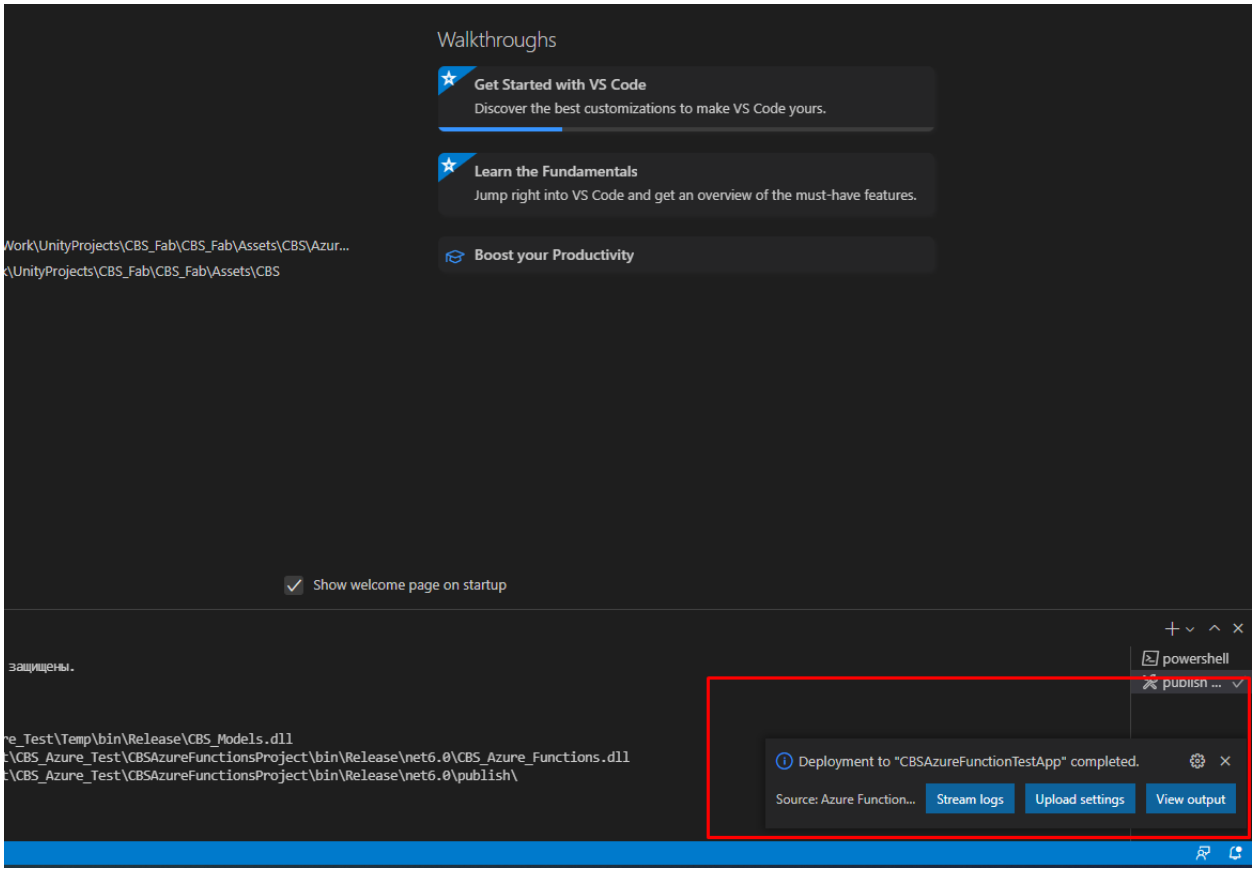
Select **.NET 6** stack



Choose a region suitable for you and wait until the end of creating the Function App

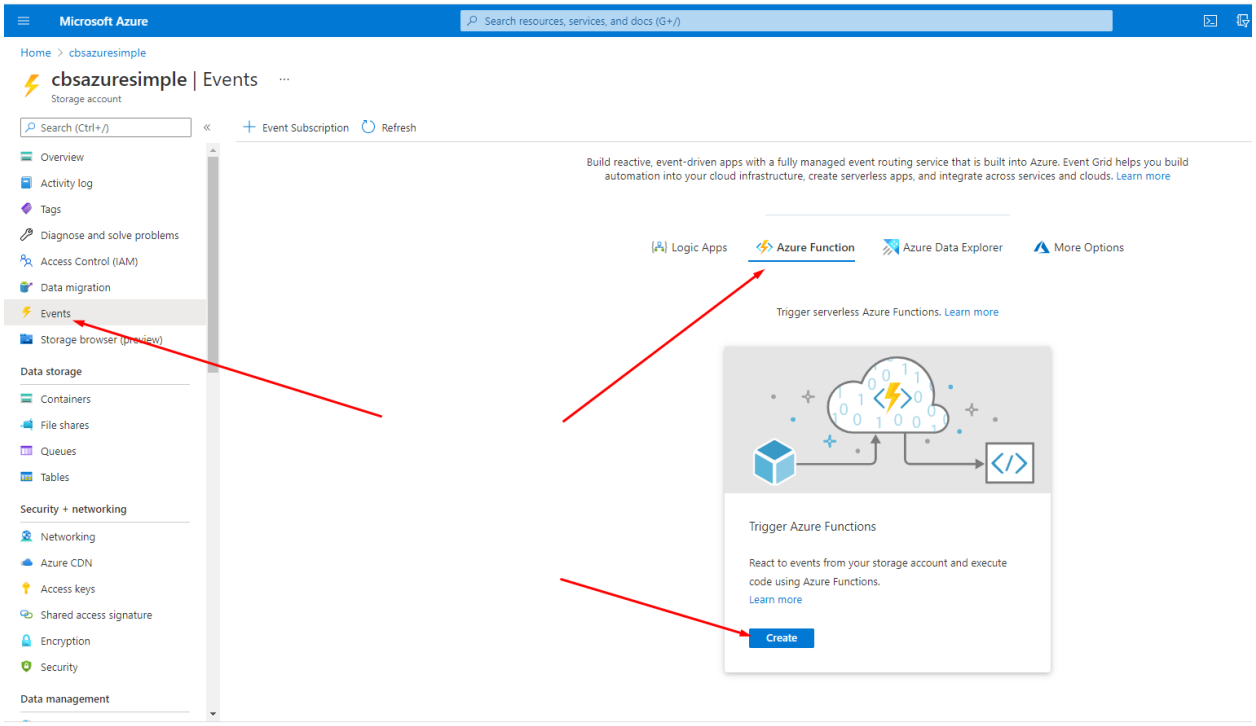


Finally you will get a message that the functions have uploaded successfully

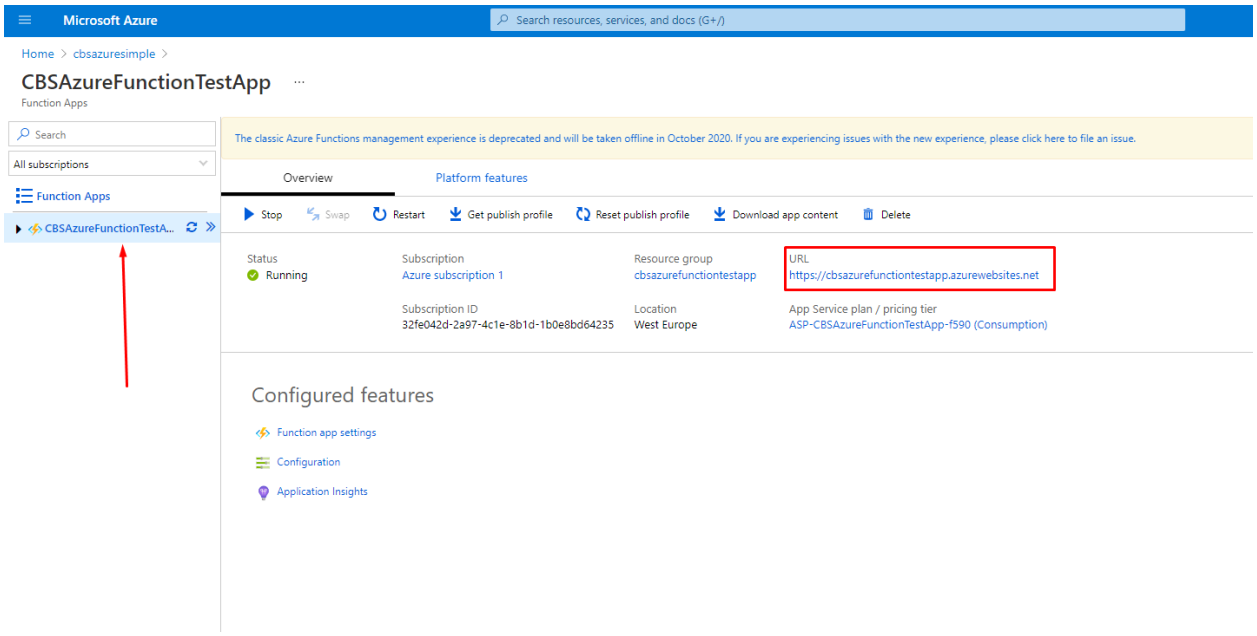


5. Link Azure and Playfab

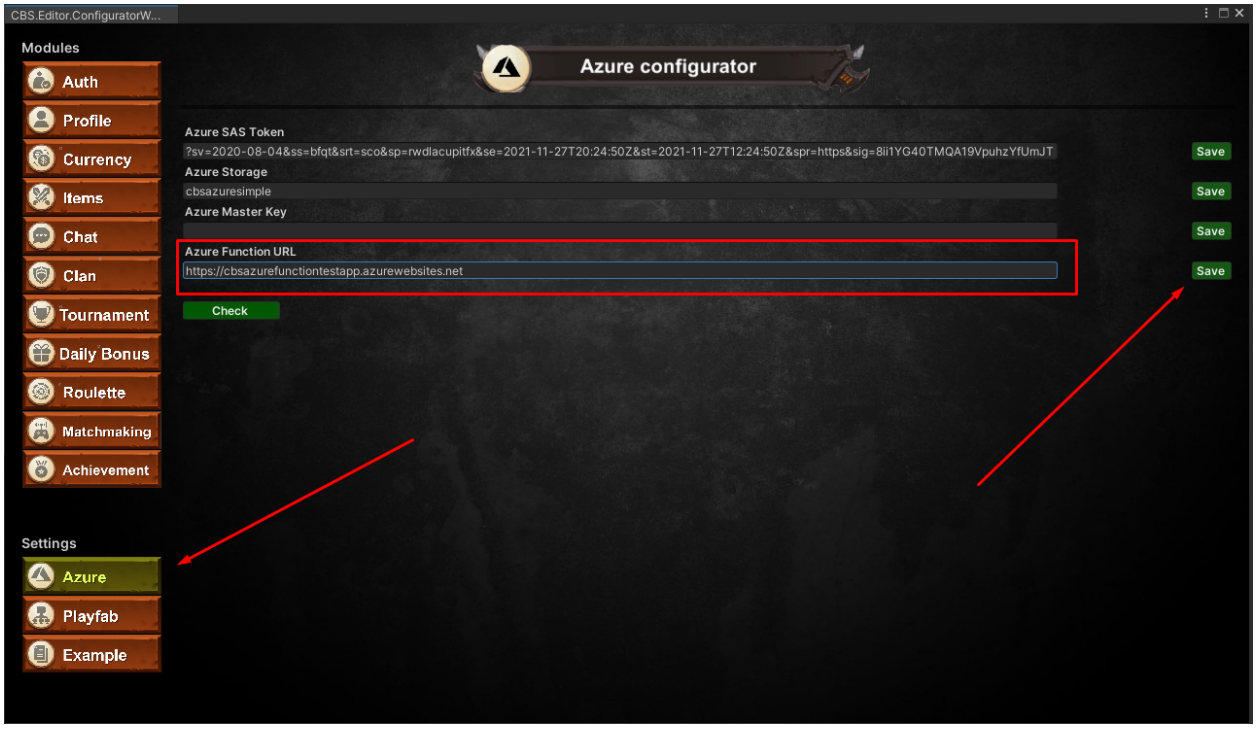
Return to the Azure Portal, go to the Events tab, select the Azure Function and click Create



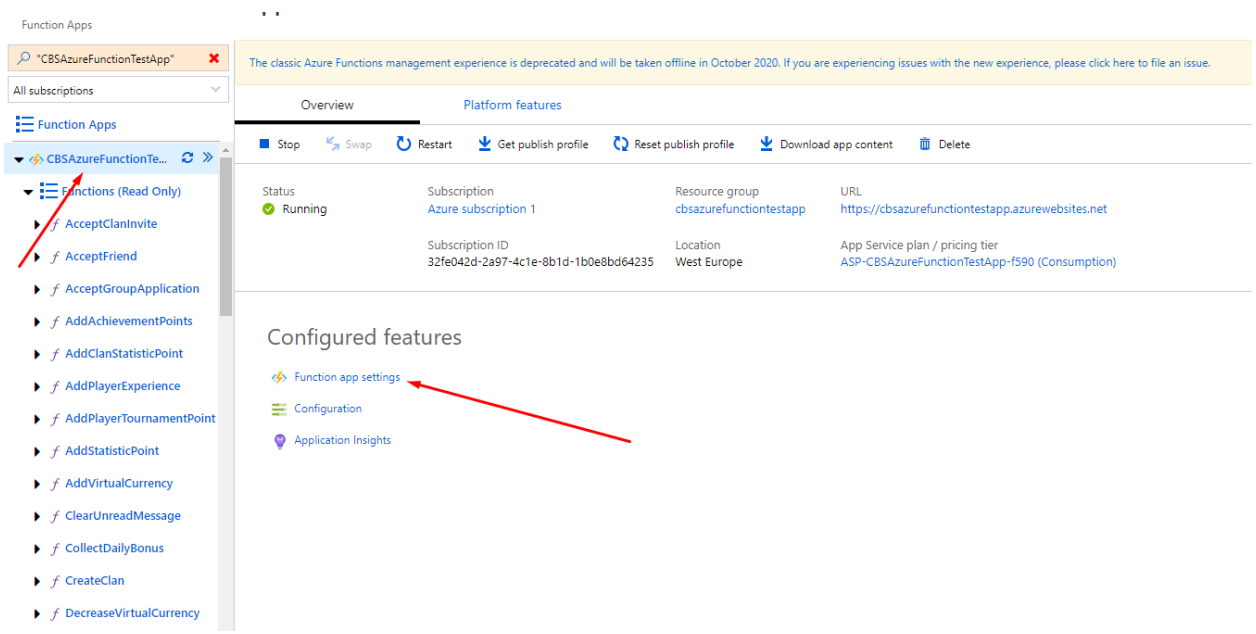
Click on the function app and copy the function url



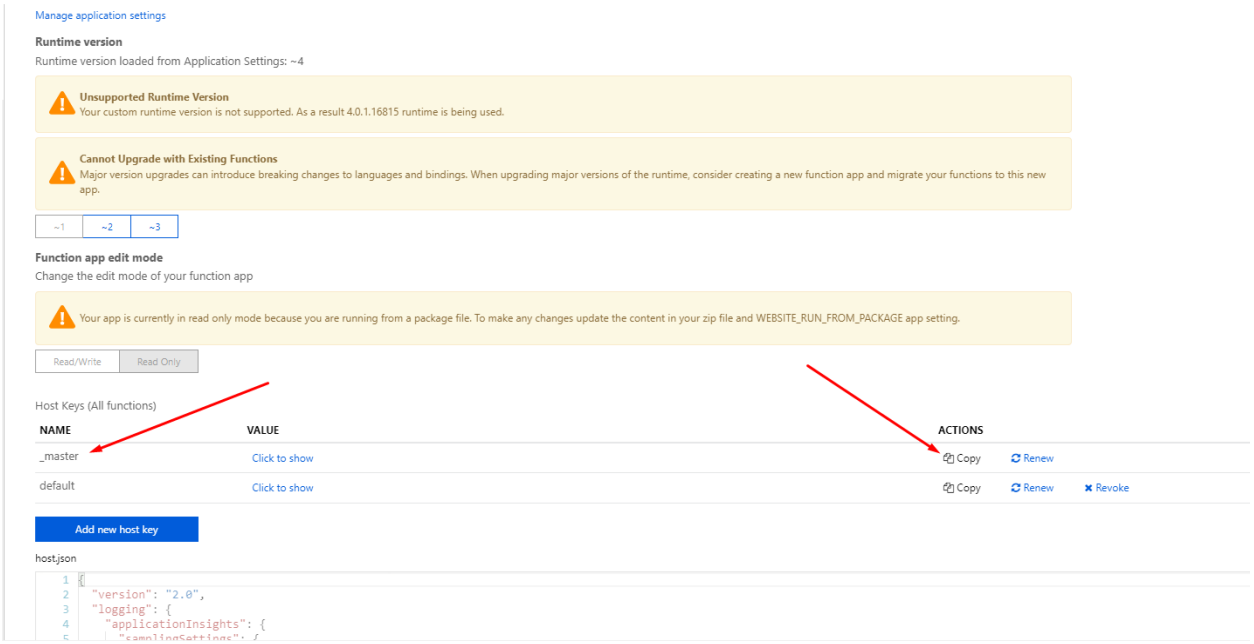
Go back to the Unity editor, go to the Azure tab and paste the copied URL into the "Azure Function URL" box. Click Save



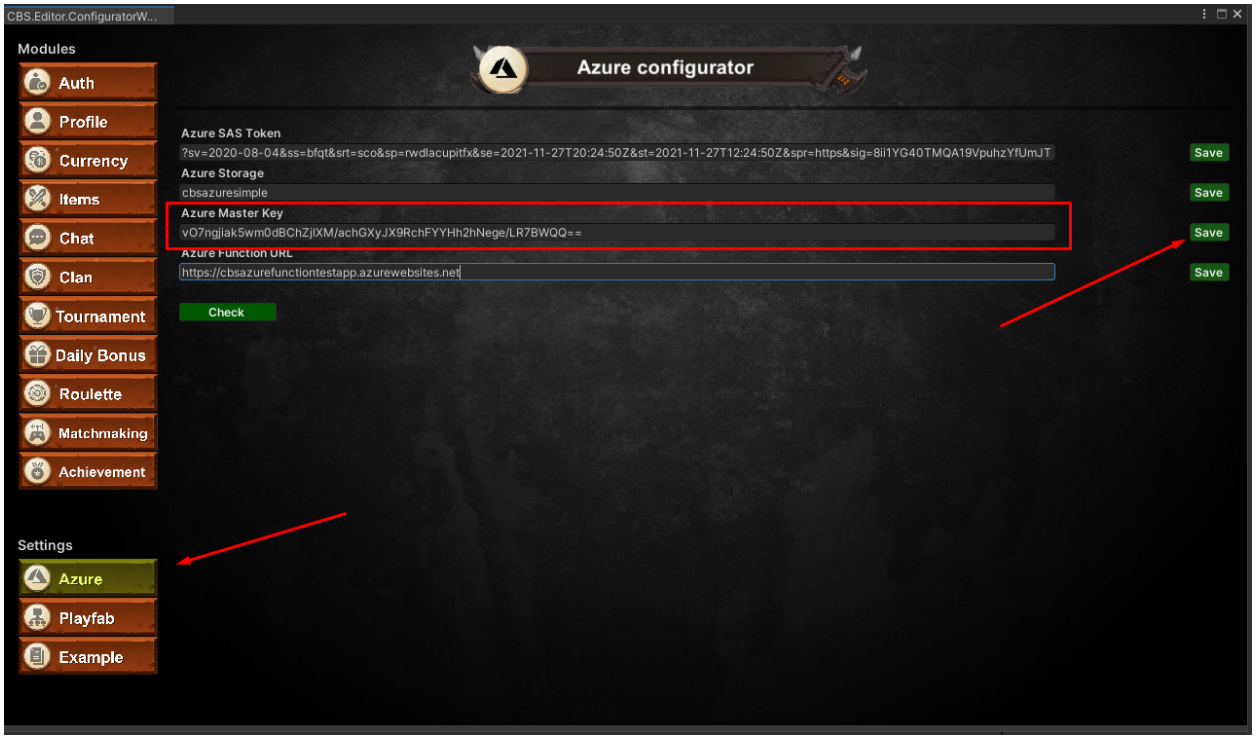
Go back to the Azure Portal and click "Function app settings"



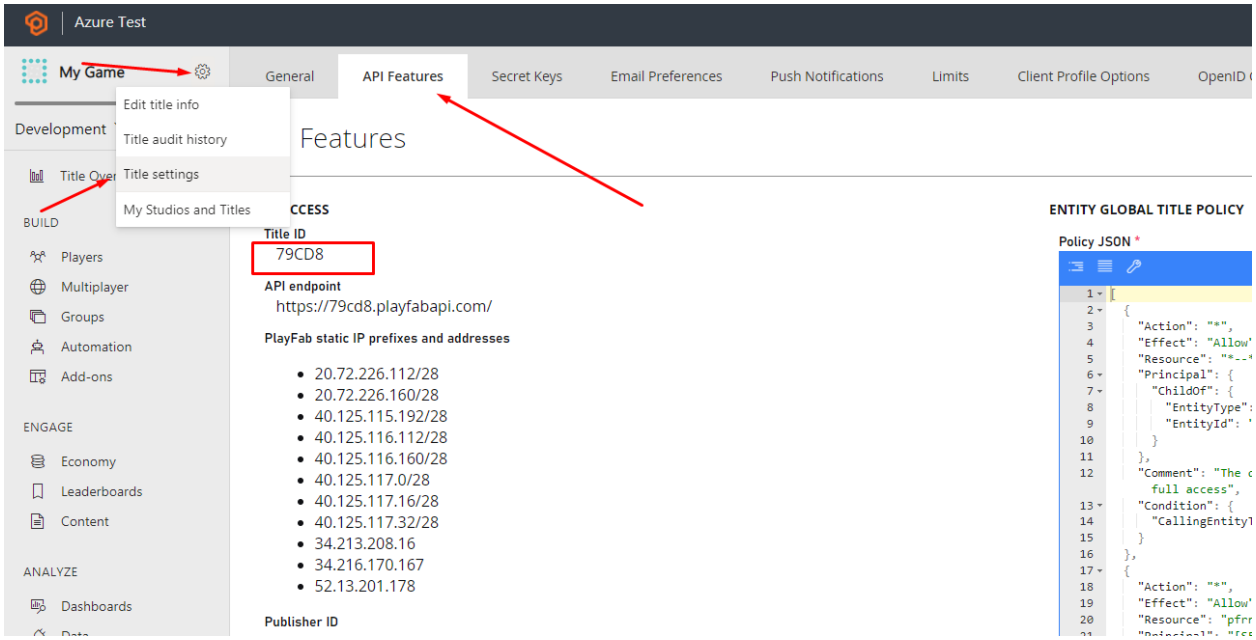
Copy master key



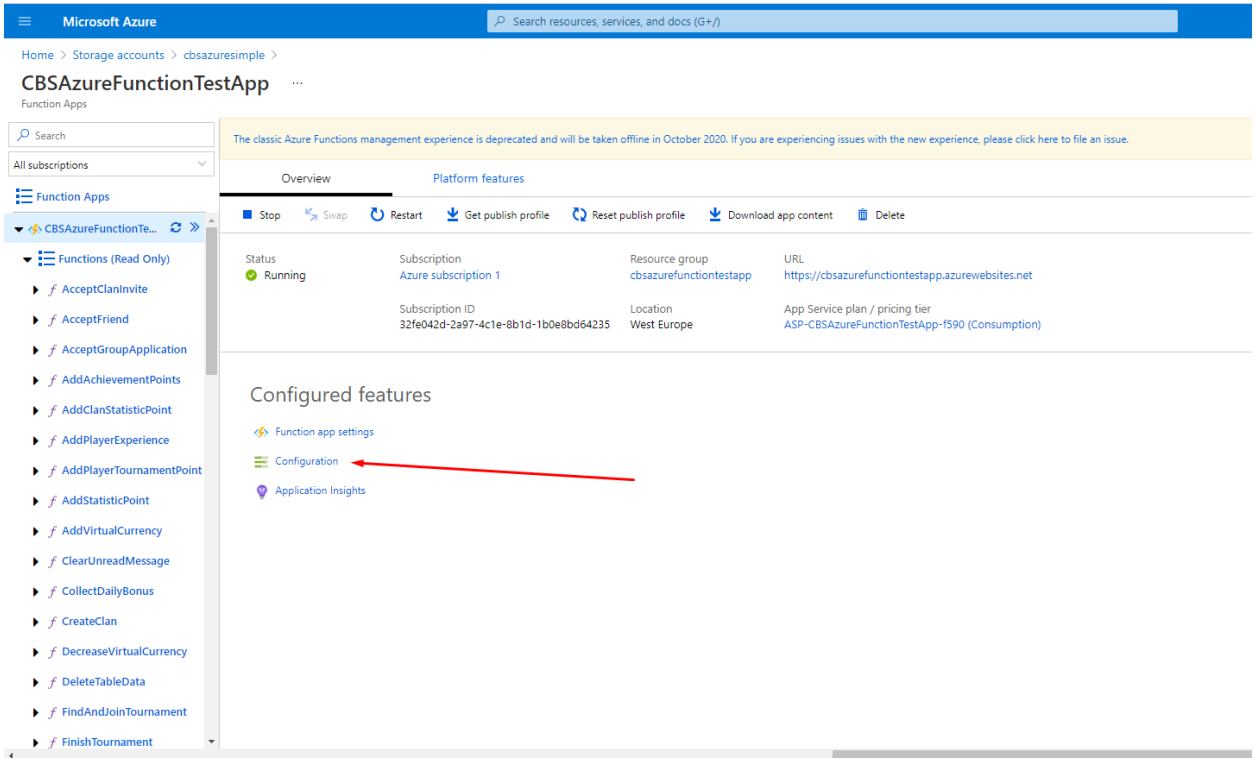
Go back to the Unity editor, go to the Azure tab and paste the copied master key into the "Azure Master Key" box. Click Save



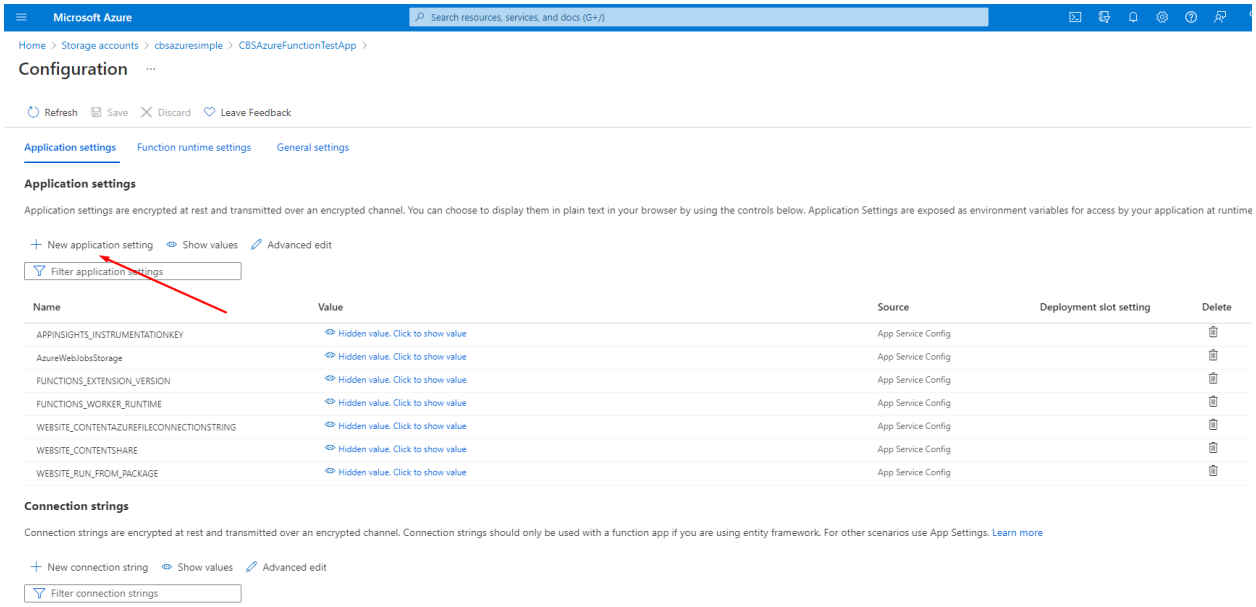
Go to the Playfab portal. Go to the "**API Features**" tab. Copy "**Title ID**"



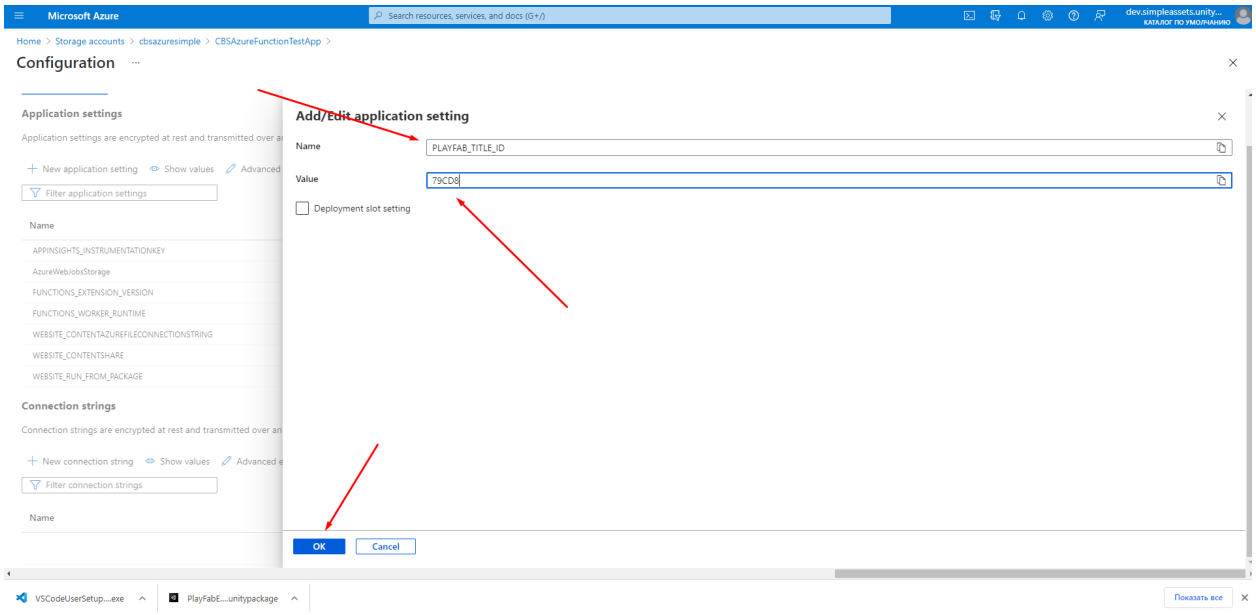
Go back to the Azure Portal and click "Configuration"



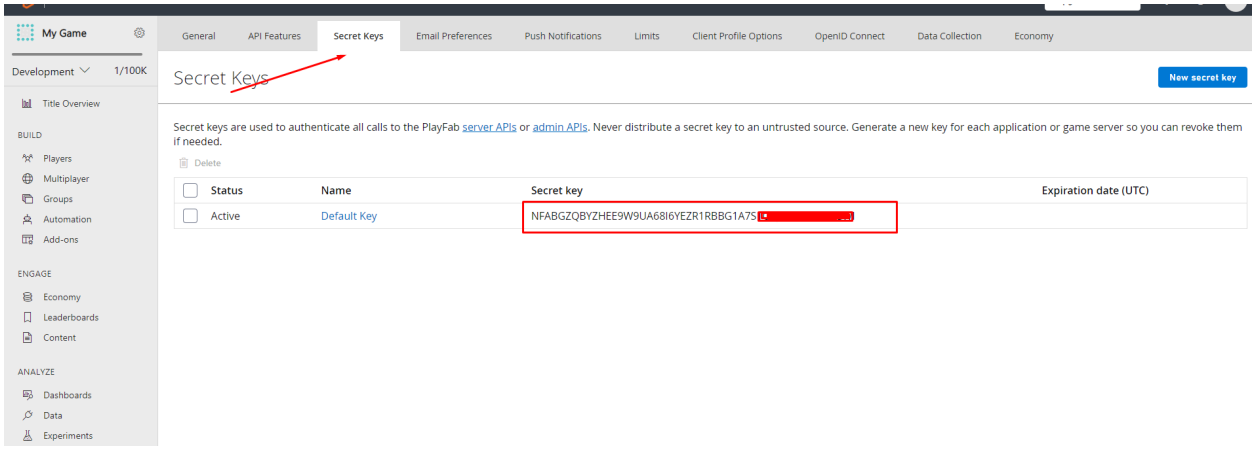
Click "New Application Setting"



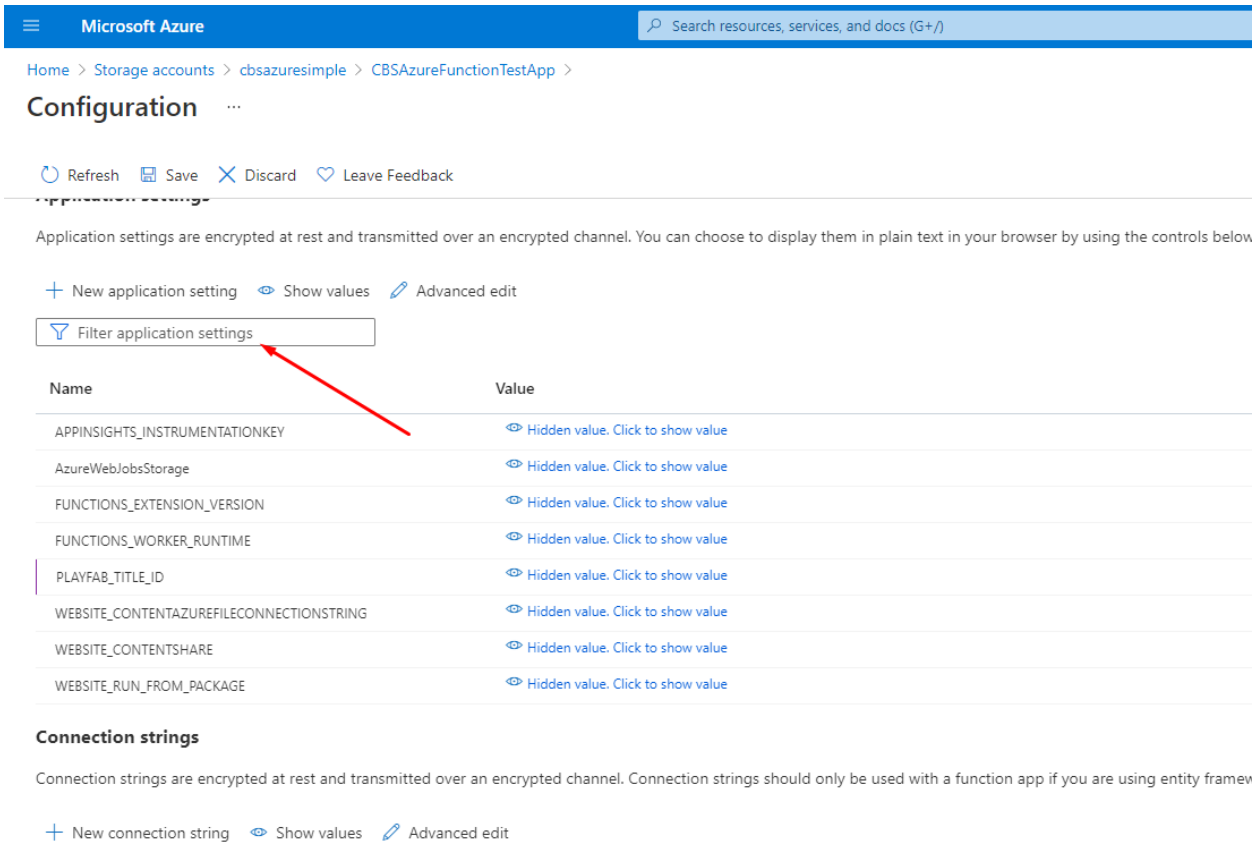
Enter **PLAYFAB_TITLE_ID** in the Name field and the **copied Title ID** in the Value field. Click OK



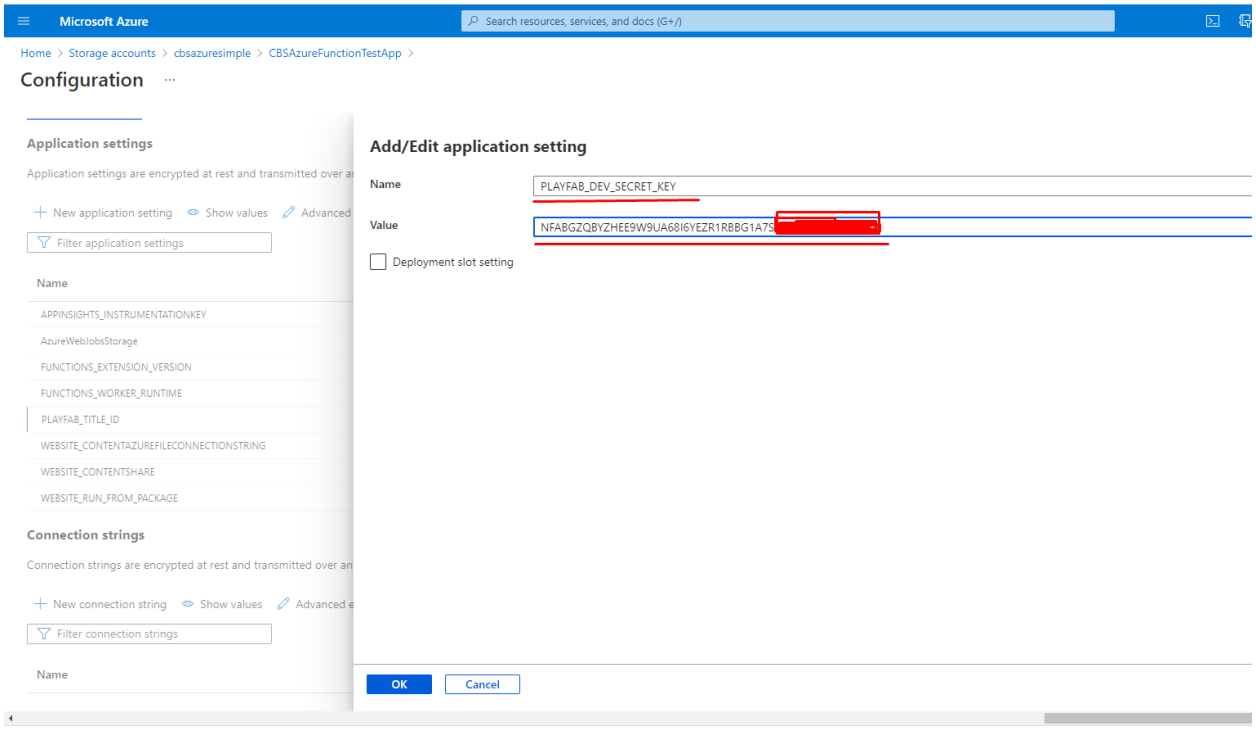
Go to the Playfab portal. Go to the "**Secret Keys**" tab. Copy your "Secret key"



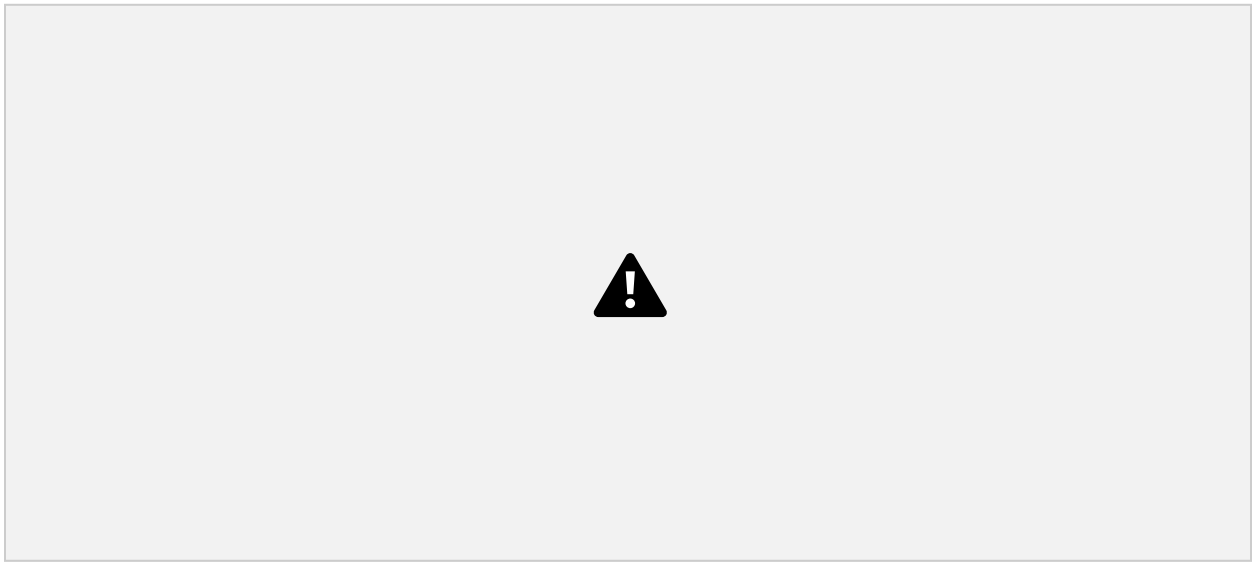
Return to Azure Portal. Again Click "New Application Setting"



Enter **PLAYFAB_DEV_SECRET_KEY** in the Name field and the **copied Secret key** in the Value field. Click OK

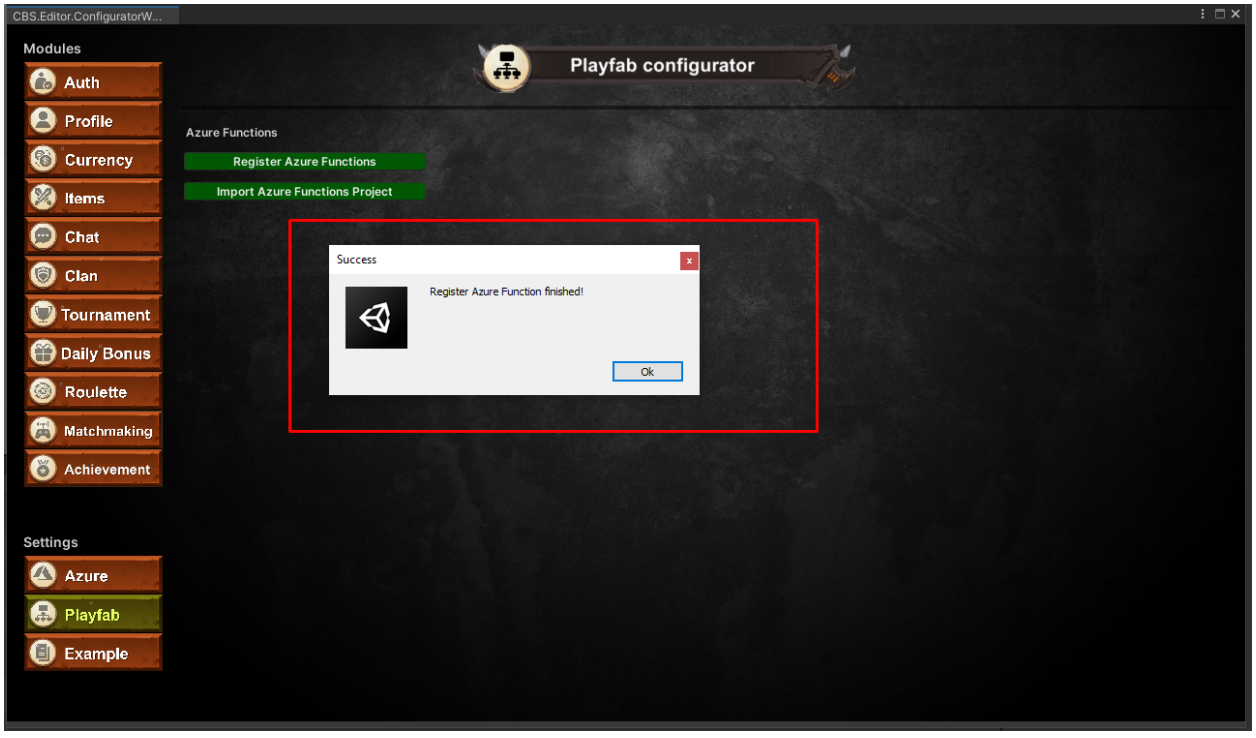


Finally click **Save**



Return to the Unity Editor, go to the "Playfab" tab and click "**Register Azure Functions**"

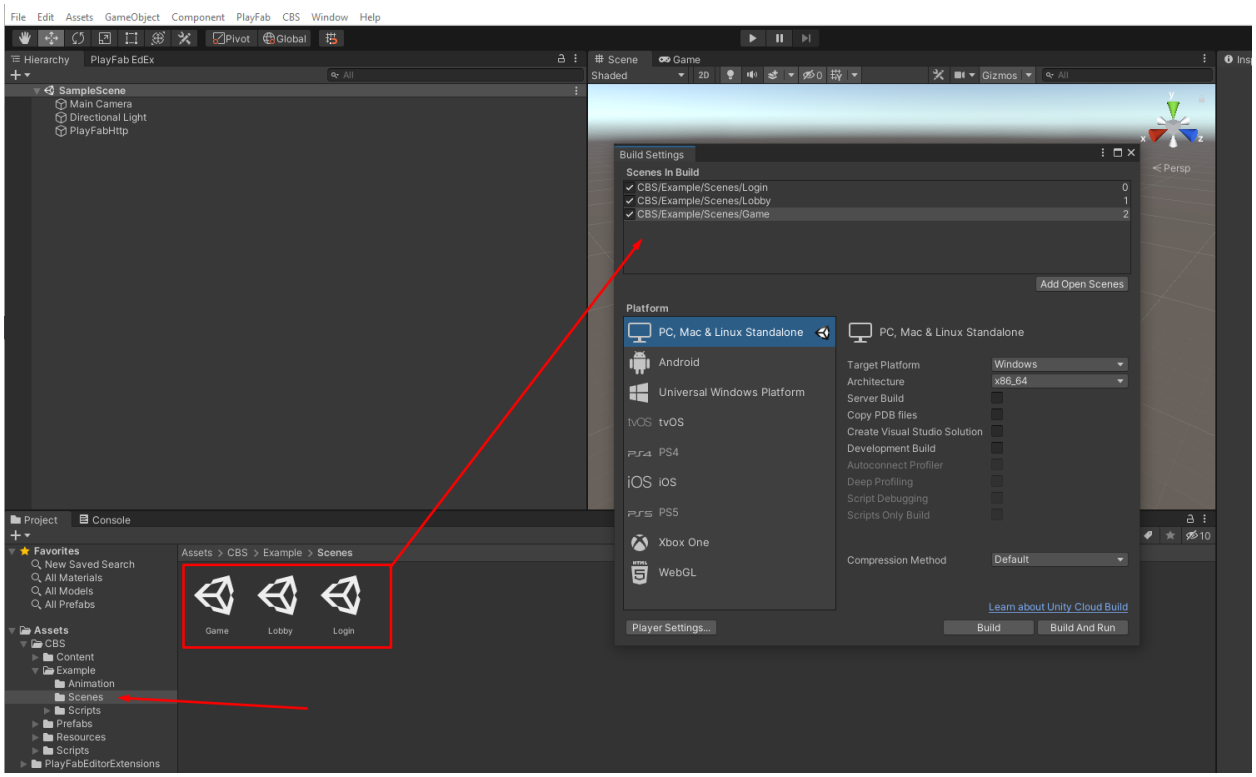
Finally, you will receive a message about the successful registration of functions on the Playfab.



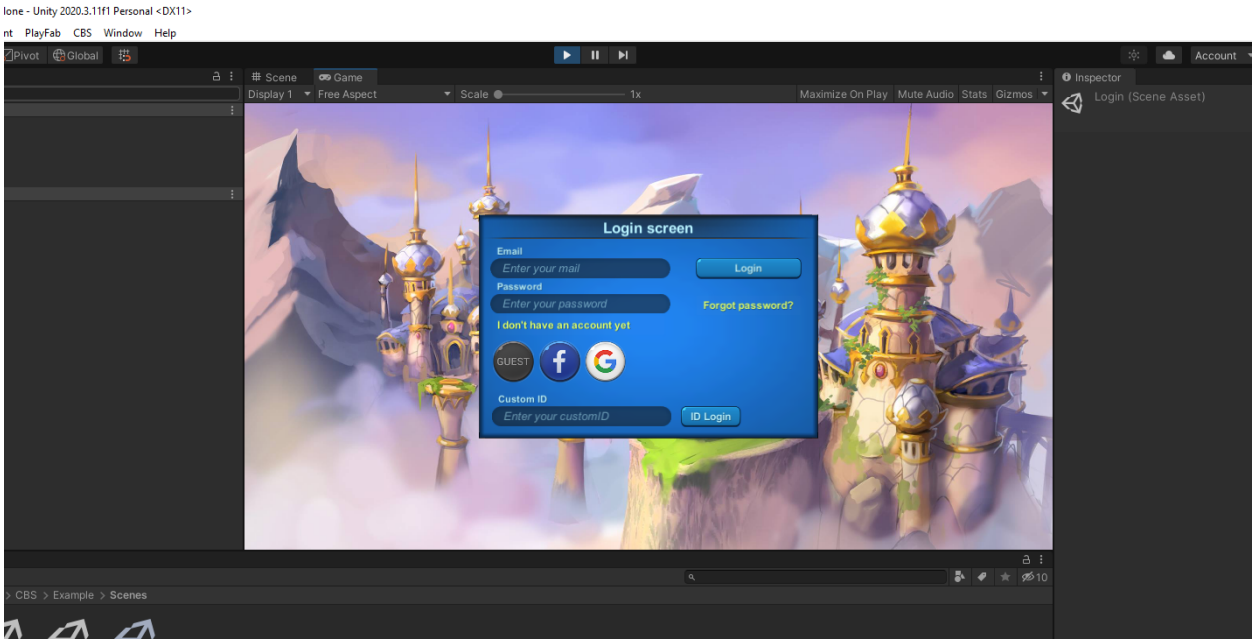
Congratulations, you have successfully finished installing the plugin!

6. Run examples

Add examples scenes to "Build Settings"



Run **Login** scene



If you have any questions - please contact us - **dev.simpleassets.unity@gmail.com**

Unity forum
https://forum.unity.com/threads/released-cbs-cross-platform-backend-solution.1176128/
Discord server
https://discord.gg/qUwPVPfxve