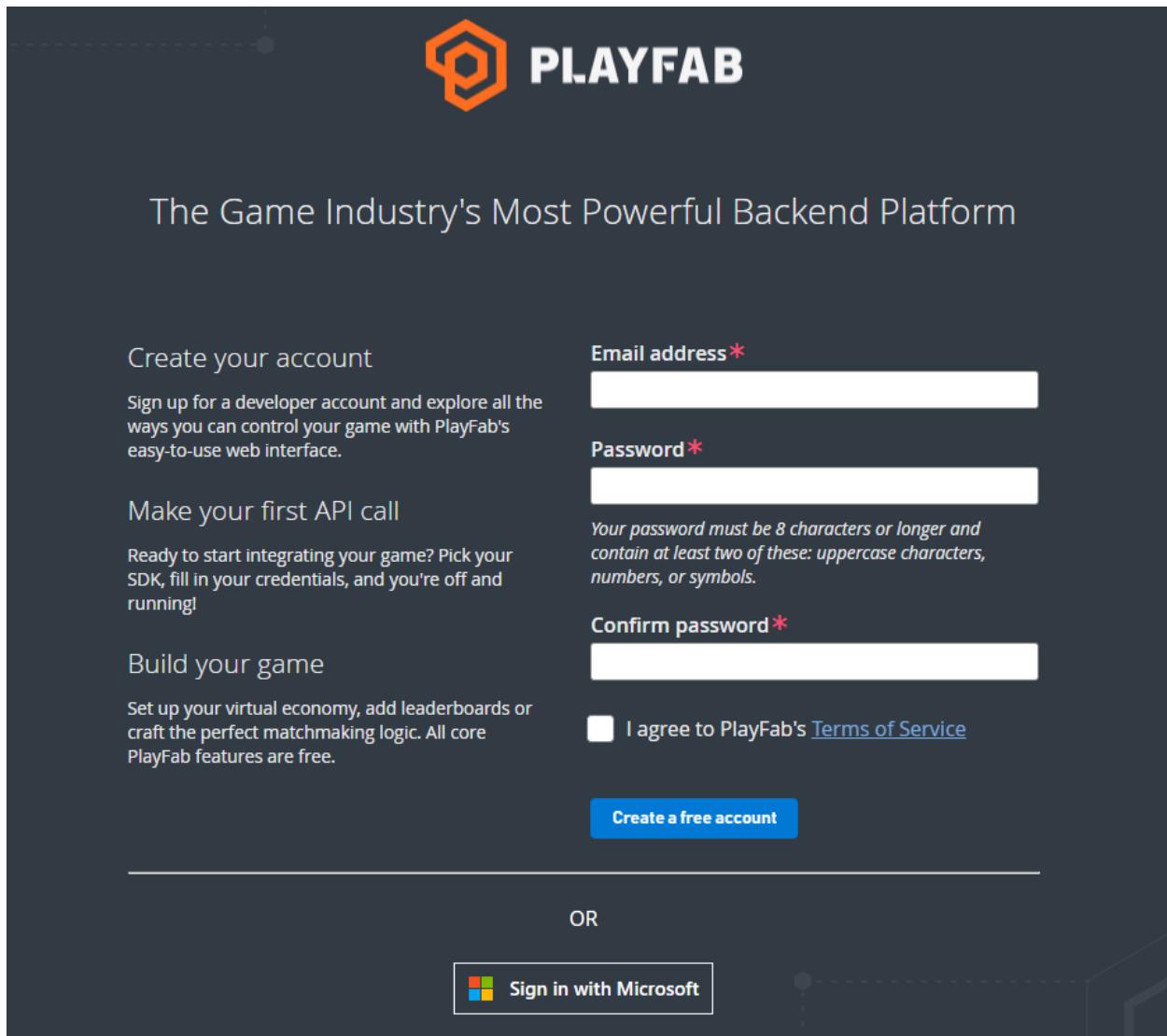


Install plugin (for version 0.4.0 or higher)

1. Playfab

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



The screenshot shows the Playfab developer account creation page. At the top is the Playfab logo (an orange hexagon icon) and the word "PLAYFAB". Below it is the tagline "The Game Industry's Most Powerful Backend Platform". On the left, there are three sections: "Create your account", "Make your first API call", and "Build your game". Each section has a brief description and a link. On the right, there are fields for "Email address*", "Password*", and "Confirm password*". A note below the password field specifies password requirements: "Your password must be 8 characters or longer and contain at least two of these: uppercase characters, numbers, or symbols." There is also a checkbox for agreeing to the Terms of Service. At the bottom is a blue "Create a free account" button.

Create your account

Sign up for a developer account and explore all the ways you can control your game with PlayFab's easy-to-use web interface.

Make your first API call

Ready to start integrating your game? Pick your SDK, fill in your credentials, and you're off and running!

Build your game

Set up your virtual economy, add leaderboards or craft the perfect matchmaking logic. All core PlayFab features are free.

Email address*

Password*

Your password must be 8 characters or longer and contain at least two of these: uppercase characters, numbers, or symbols.

Confirm password*

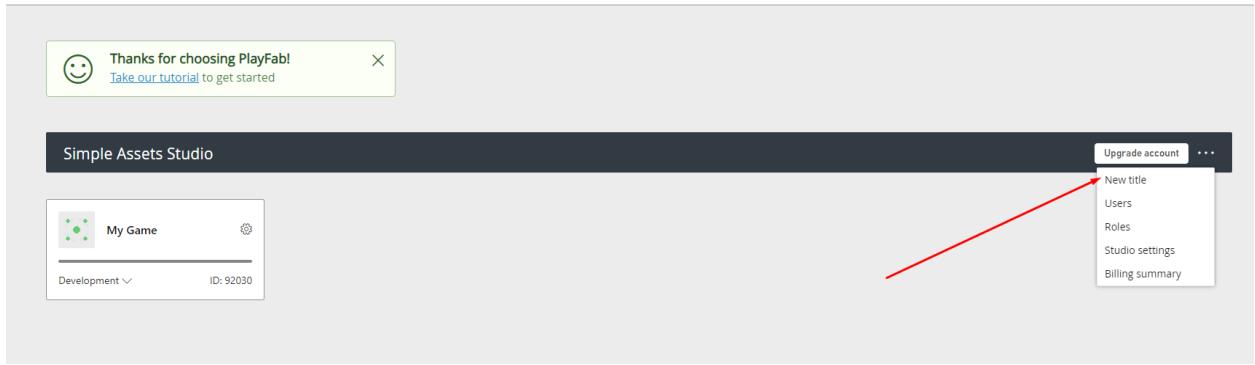
I agree to PlayFab's [Terms of Service](#)

[Create a free account](#)

OR

 Sign in with Microsoft

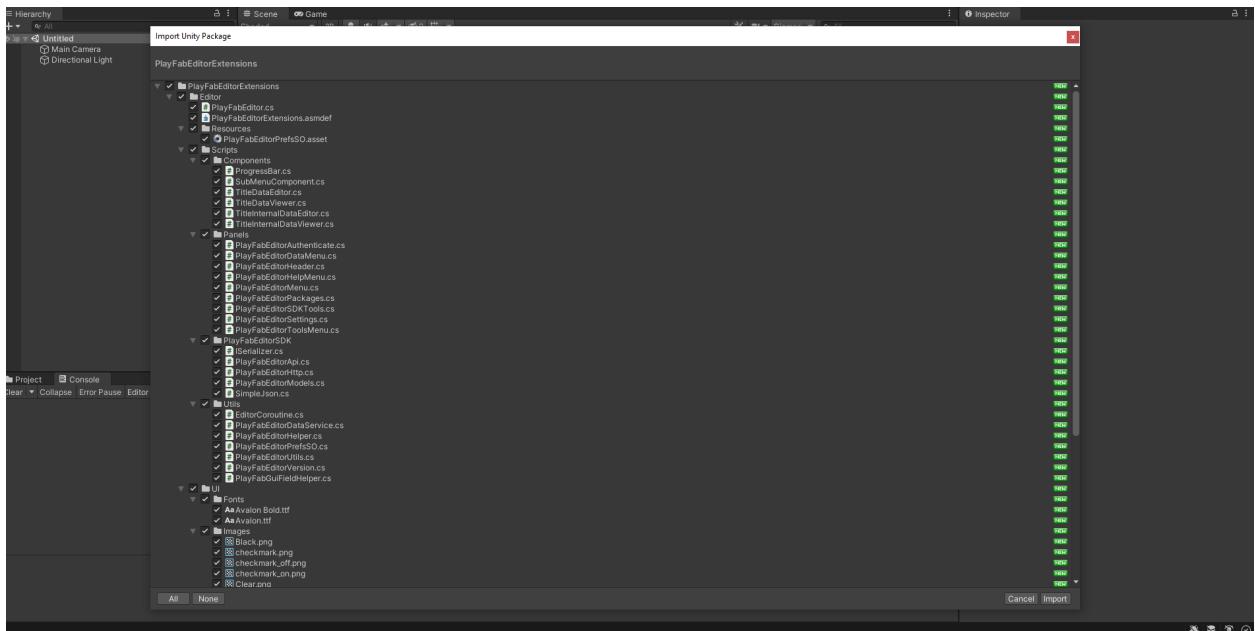
Create new title on Playfab.



It's all about Playfab.

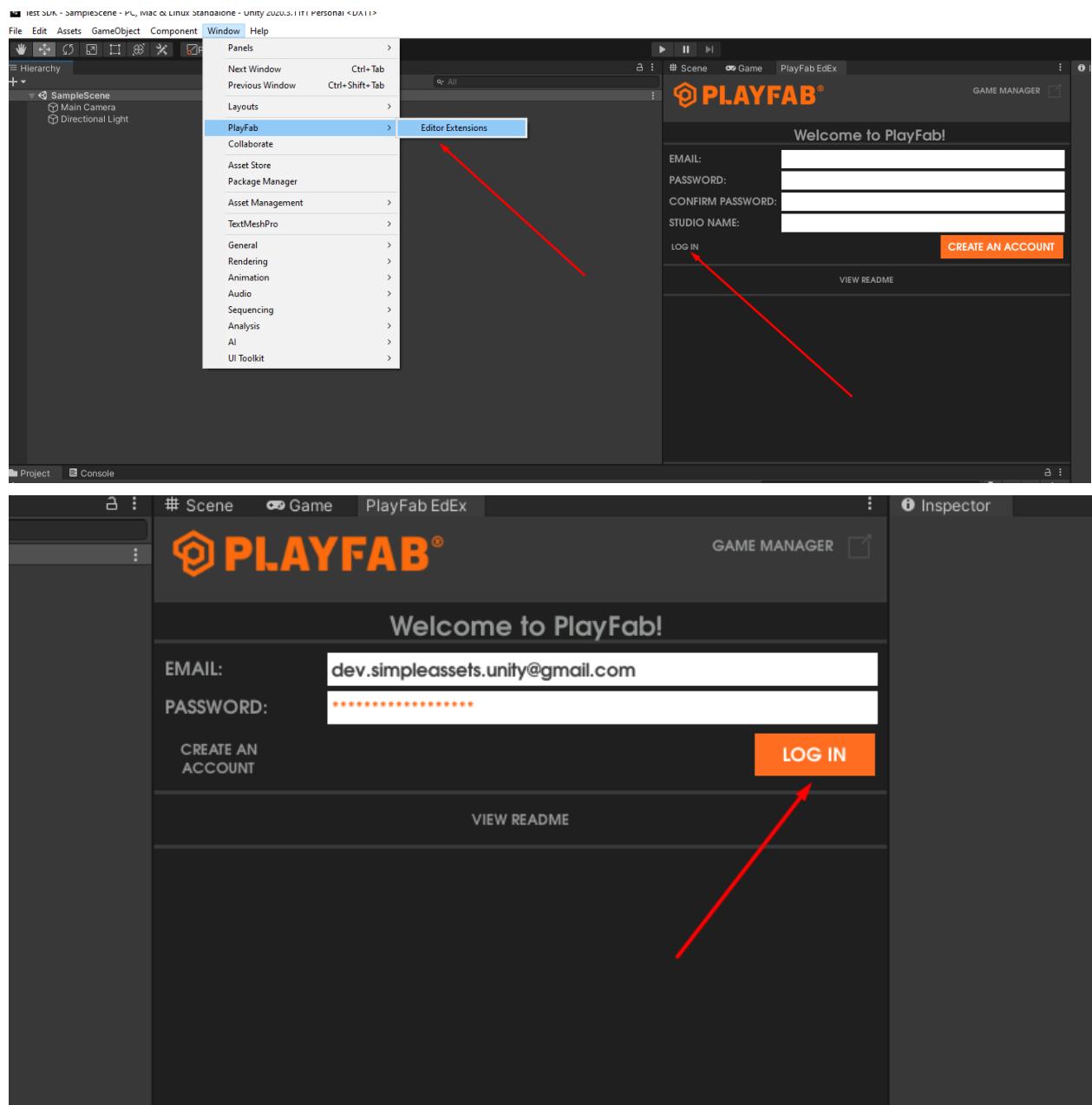
2. Import CBS Plugin.

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

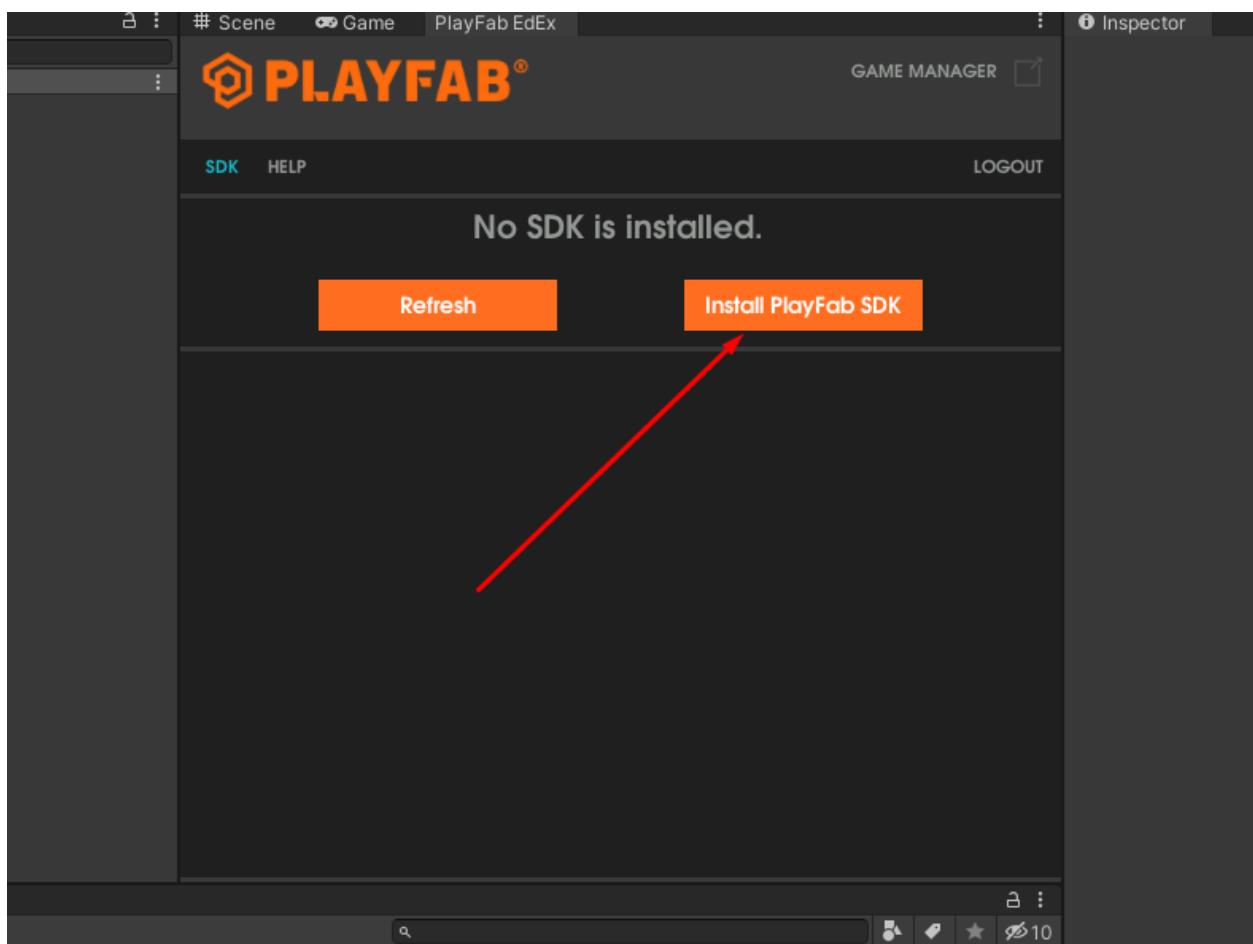


Open Playfab editor and login with Playfab account.

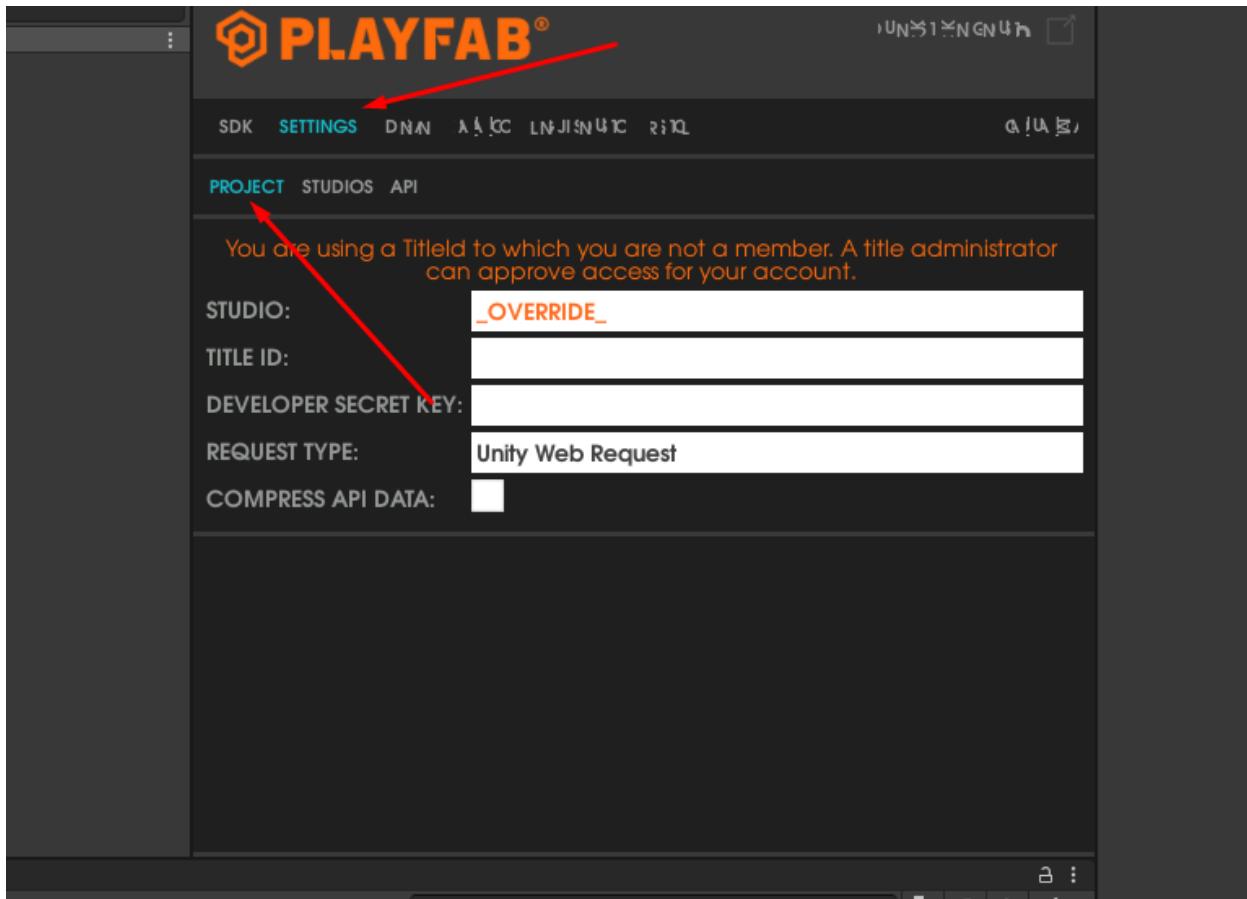
WARNING. If the Playfab context menu does not appear - just restart Unity.



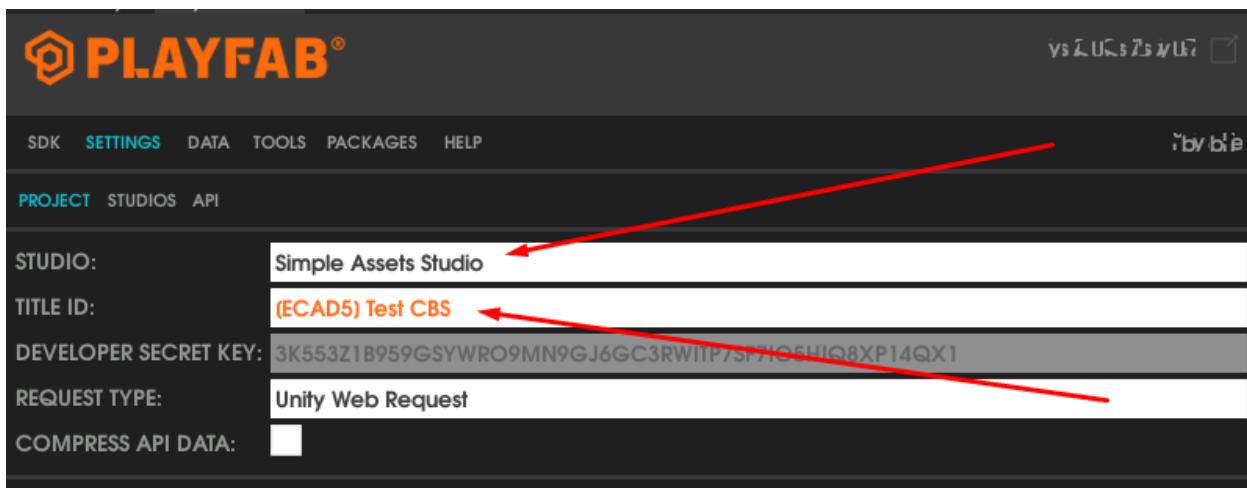
Install Playfab SDK.



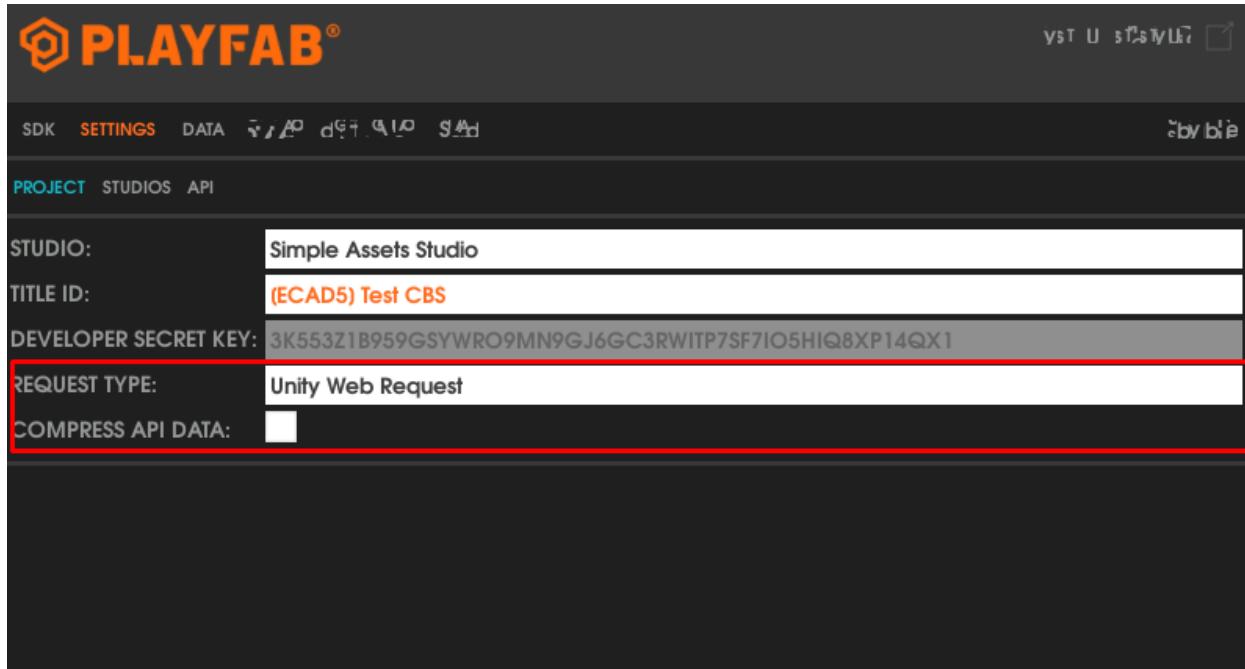
Navigate to SETTING->PROJECT



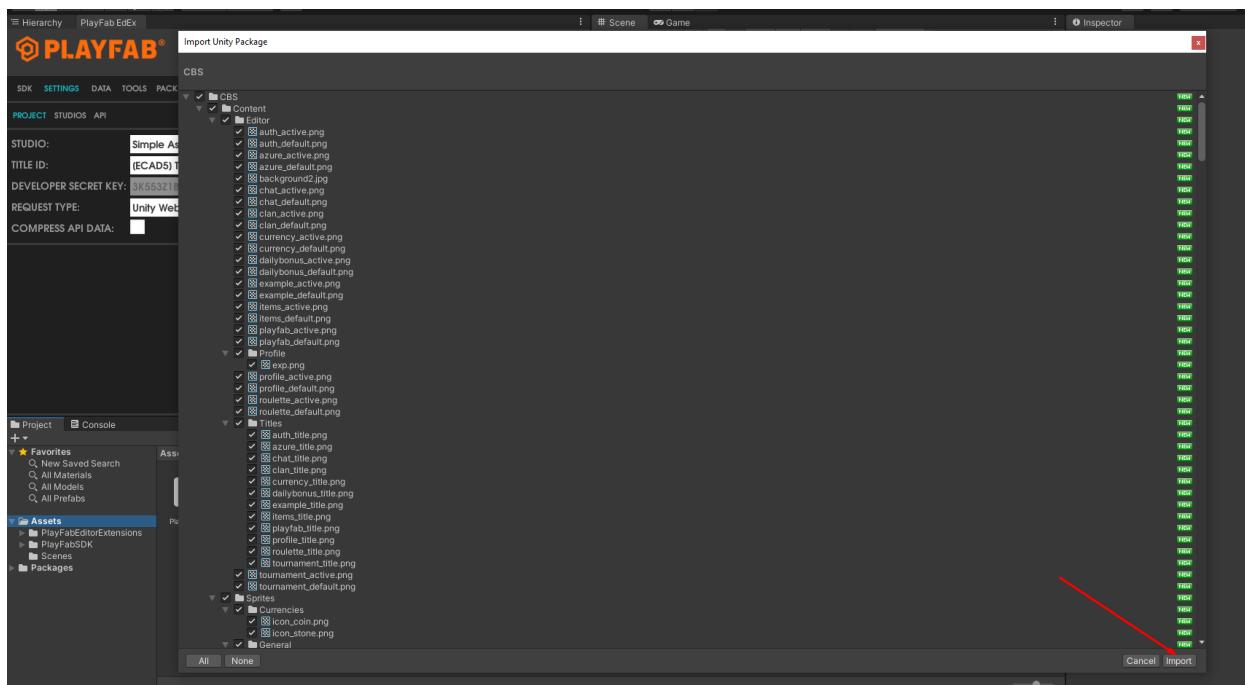
Select the Studio and Title ID you created earlier



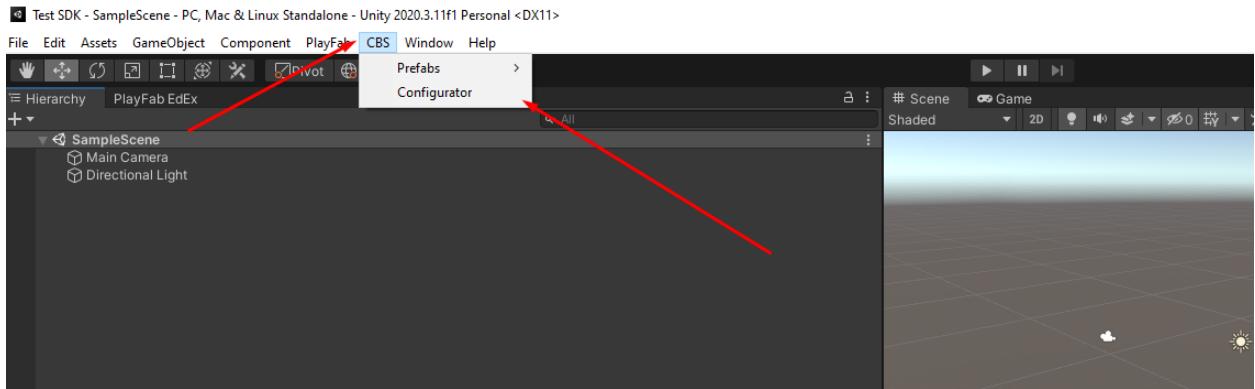
Select Request Type - Unity Web Request and don't Compress Api Data to make the solution as cross-platform as possible



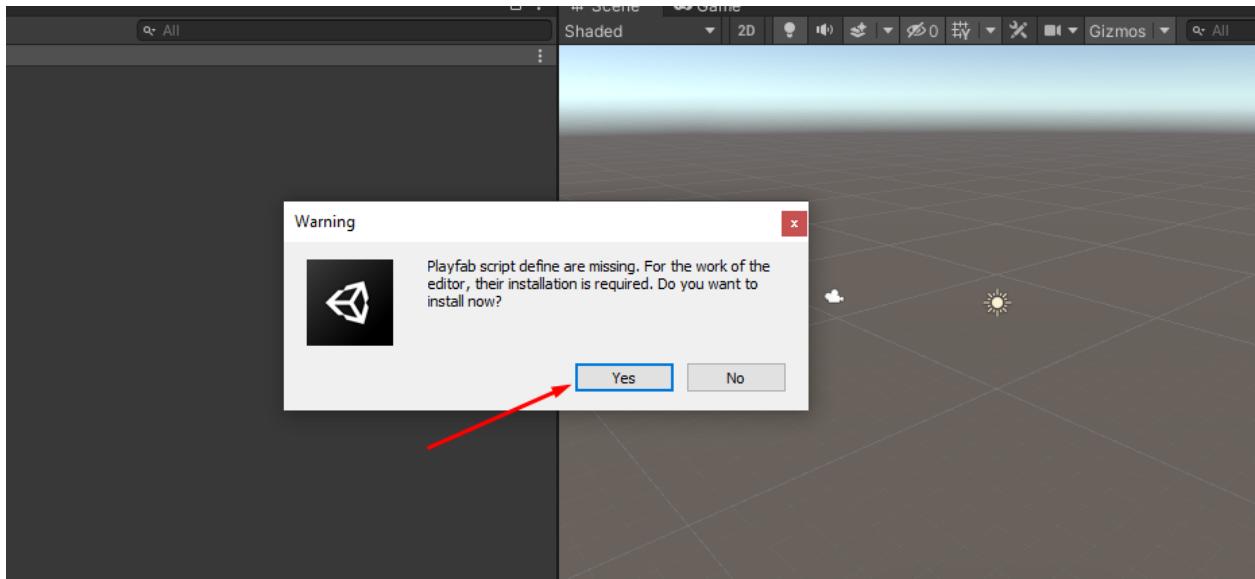
Import CBS plugin from asset-store



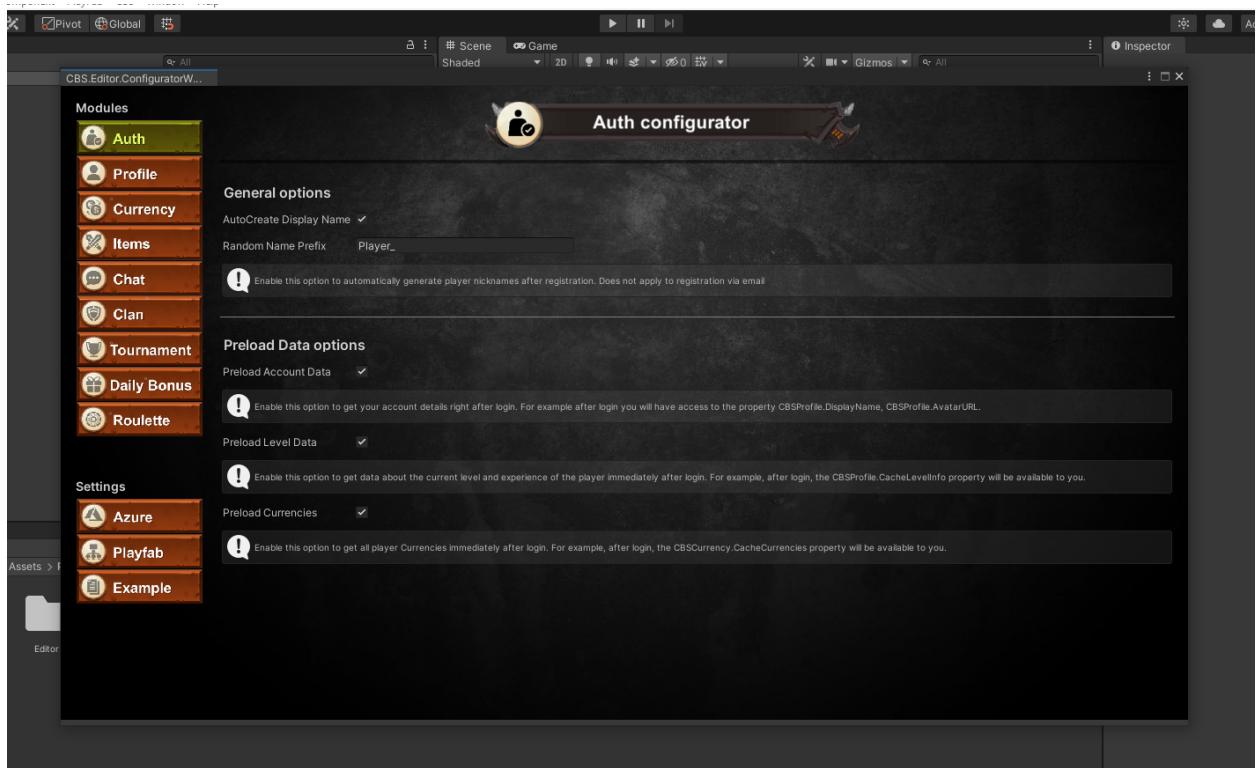
Open CBS->Configurator



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**

The screenshot shows the Azure portal homepage. On the left, a sidebar menu is open, listing various services under 'FAVORITES' such as All resources, Resource groups, App Services, Function App, SQL databases, Azure Cosmos DB, Virtual machines, Load balancers, Storage accounts, Virtual networks, Azure Active Directory, Monitor, Advisor, Security Center, Cost Management + Billing, Help + support, and another Cost Management + Billing entry. A red arrow points from the 'Storage accounts' link towards the main content area. The main content area features a 'Welcome to Azure!' message and three promotional cards: 'Start with an Azure free trial' (with a 'Start' button), 'Manage Azure Active Directory' (with a 'View' button), and 'Access student benefits' (with a 'Explore' button). Each card includes a small icon and a brief description.

Click "Start"

Fill in all the information provided and confirm it

This screenshot shows the same Azure portal homepage as the first one, but without the sidebar menu visible. It displays the 'Welcome to Azure!' message and the three promotional cards: 'Start with an Azure free trial', 'Manage Azure Active Directory', and 'Access student benefits'. A red arrow points from the bottom left towards the 'Start with an Azure free trial' card, specifically pointing at the 'Start' button.

After that navigate to storage account again

The screenshot shows the Azure portal home page. On the left, a red arrow points from the 'Create a resource' button in the sidebar to the 'Storage accounts' icon in the 'Azure services' section. Another red arrow points from the 'Storage accounts' link in the sidebar to the 'Storage accounts' section in the center. The center area displays the 'Storage accounts' section with a 'Create a resource' button, a list of existing storage accounts, and links to 'Microsoft Learn', 'Azure Monitor', and 'Recent Azure Quickstart'.

Create new storage account.

The screenshot shows the 'Storage accounts' blade in the Azure portal. A red arrow points from the 'Create storage account' button at the bottom of the blade to the 'Create storage account' button in the 'Storage accounts' section of the main page.

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

Basics Advanced Networking Data protection Tags Review + create

Notes: The cost of your storage account depends on the usage and the options you choose below. [Learn more about Azure storage accounts](#)

Project details

Select the subscription in which to create the new storage account. Choose a new or existing resource group to organize and manage your storage account together with other resources.

Subscription *

Resource group * [Create new](#)

Instance details

If you need to create a legacy storage account type, please click [here](#).

Storage account name ⓘ * cbsteststorage

Region ⓘ *

Performance ⓘ * Standard: Recommended for most scenarios (general-purpose v2 account)
 Premium: Recommended for scenarios that require low latency.

Redundancy ⓘ * Make read access to data available in the event of regional unavailability.

[Review + create](#)

< Previous

Next : Advanced >

After success deployment - click "Go to resource"

Microsoft Azure

Search resources, services, and docs (G+)

Home > cbsteststorage_1627910284535 | Overview

Deployment

Search (Ctrl+ /) Delete Cancel Redeploy Refresh

Overview Inputs Outputs Template

We'd love your feedback! →

✓ Your deployment is complete

Deployment name: cbsteststorage_1627910284535
Subscription: Azure subscription 1
Resource group: CBS_Group

Start time: 8/2/2021, 4:18:13 PM
Correlation ID: c363c025-383d-408f-8d6f-1095e776940a

Deployment details (Download)

Next steps

Go to resource



Navigate to "Shared access signature"

cbsteststorage | Shared access signature ...

Storage account

Search (Ctrl+ /)

Overview

Activity log

Tags

Diagnose and solve problems

Access Control (IAM)

Data migration

Events

Storage Explorer (preview)

Data storage

Containers

File shares

Queues

Tables

Security + networking

Networking

Azure CDN

Access keys

Shared access signature

Encryption

Security

Data management

Allow all resource types

A shared access signature (SAS) is a URI that grants restricted access rights to Azure Storage resources. You can provide a shared access signature URI to these clients, you grant them access to a resource for a specified period of time.

An account-level SAS can delegate access to multiple storage services (i.e. blob, file, queue, table). Note that stored access policies are not supported at the account level.

[Learn more](#)

Allowed services ⓘ

Blob File Queue Table

Allowed resource types ⓘ

Service Container Object

Allowed permissions ⓘ

Read Write Delete List Add Create Update Process

Blob versioning permissions ⓘ

Enables deletion of versions

Allowed blob index permissions ⓘ

Read/Write Filter

Start and expiry date/time ⓘ

Start

End

(UTC+02:00) Helsinki, Kyiv, Riga, Sofia, Tallinn, Vilnius

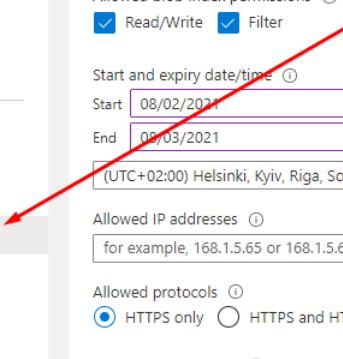
Allowed IP addresses ⓘ

for example, 168.1.5.65 or 168.1.5.65-168.1.5.70

Allowed protocols ⓘ

HTTPS only HTTPS and HTTP

Preferred routine tier ⓘ



cbsteststorage | Shared access signature

Storage account

Search (Ctrl+ /)

Overview

Activity log

Tags

Diagnose and solve problems

Access Control (IAM)

Data migration

Events

Storage Explorer (preview)

Data storage

Containers

File shares

Queues

Tables

A shared access signature (SAS) is a URI that grants restricted access rights to Azure Storage resources. You can provide a SAS to specific clients or services. By distributing a shared access signature URI to these clients, you grant them access to a resource for a specified period of time.

An account-level SAS can delegate access to multiple storage services (i.e. blob, file, queue, table). Note that stored access policies do not support delegation of access to blob, file, queue, or table services.

Learn more

Allowed services ⓘ

Blob File Queue Table

Allowed resource types ⓘ

Service Container Object

Allowed permissions ⓘ

Read Write Delete List Add Create Update Process

Blob versioning permissions ⓘ

Enables deletion of versions

Allowed blob index permissions ⓘ

Read/Write Filter

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

The screenshot shows the 'Allowed resource types' section with 'Blob', 'File', 'Queue', and 'Table' checked. Under 'Allowed permissions', 'Service', 'Container', and 'Object' are checked. In the 'Blob versioning permissions' section, 'Enables deletion of versions' is checked. The 'Allowed blob index permissions' section has 'Read/Write' checked. The 'Start and expiry date/time' section shows 'Start' as 08/02/2021 at 4:39:12 PM and 'End' as 08/03/2071 at 12:39:12 A. The 'Allowed IP addresses' section includes a placeholder 'for example, 168.1.5.65 or 168.1.5.65-168.1.5.70'. The 'Allowed protocols' section has 'HTTPS only' selected. The 'Preferred routing tier' section has 'Basic (default)' selected. A note states 'Some routing options are disabled because the endpoints are not published.' The 'Signing key' dropdown is set to 'key1'. A red arrow points from the text 'Go to Unity open CBS configurator and navigate to "Azure" tab' to the 'Generate SAS and connection string' button.

blob file queue table

Allowed resource types ⓘ

Service Container Object

Allowed permissions ⓘ

Read Write Delete List Add Create Update Process

Blob versioning permissions ⓘ

Enables deletion of versions

Allowed blob index permissions ⓘ

Read/Write Filter

Start and expiry date/time ⓘ

Start 4:39:12 PM

End 12:39:12 A

(UTC+02:00) Helsinki, Kyiv, Riga, Sofia, Tallinn, Vilnius

Allowed IP addresses ⓘ

for example, 168.1.5.65 or 168.1.5.65-168.1.5.70

Allowed protocols ⓘ

HTTPS only HTTPS and HTTP

Preferred routing tier ⓘ

Basic (default) Microsoft network routing Internet routing

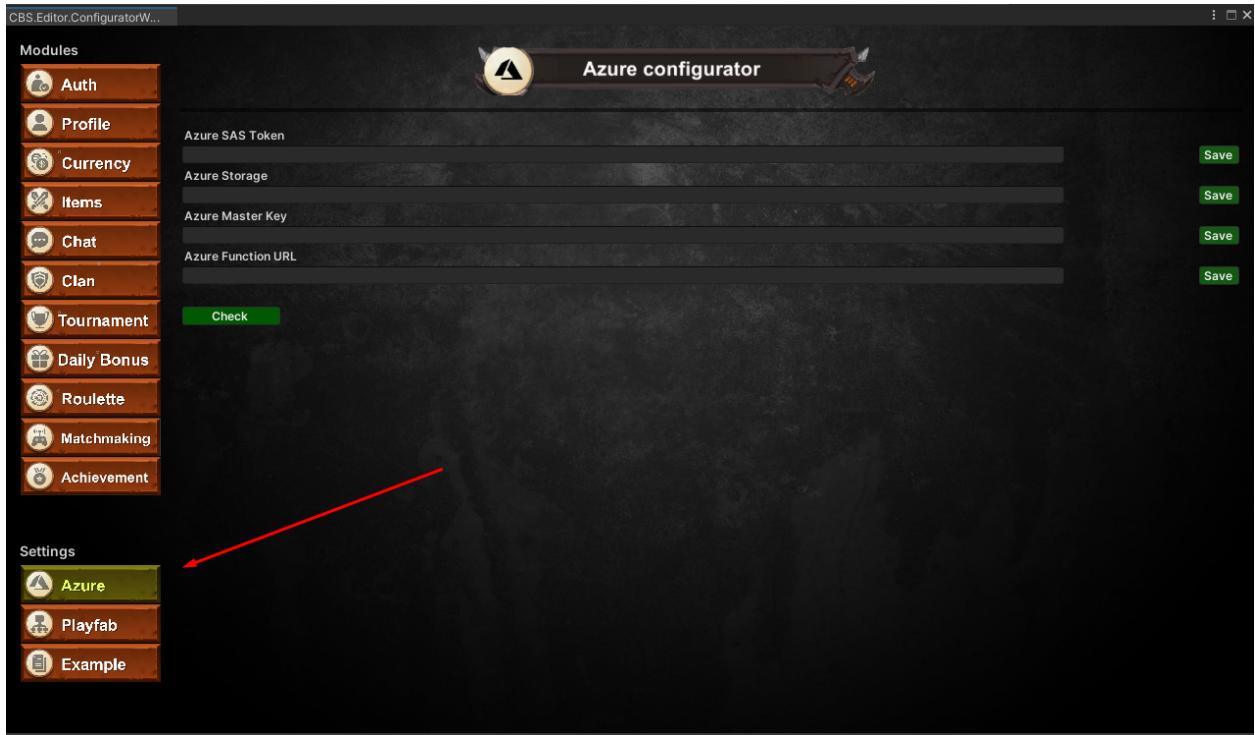
Some routing options are disabled because the endpoints are not published.

Signing key ⓘ

key1

Generate SAS and connection string

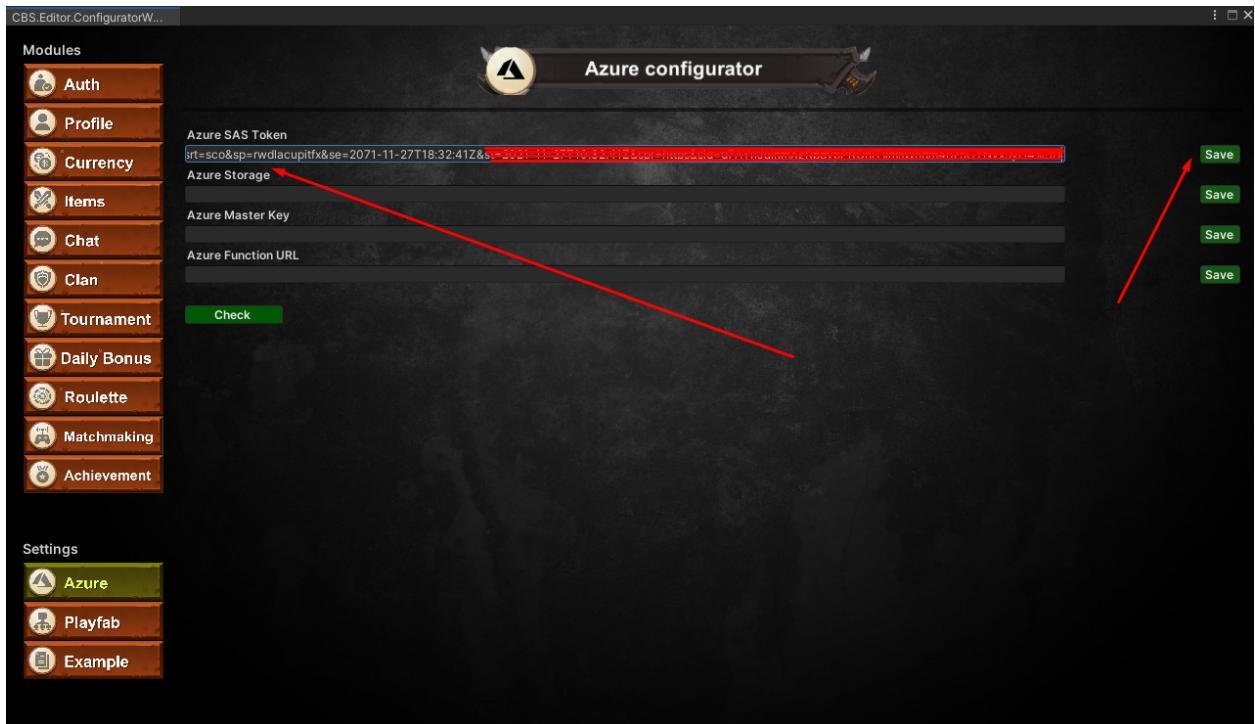
Go to Unity open CBS configurator and navigate to "Azure" tab



Copy SAS token

This screenshot shows the 'Generate SAS and connection string' dialog. It includes fields for 'Connection string', 'SAS token' (which is highlighted with a red box), 'Blob service SAS URL', 'File service SAS URL', 'Queue service SAS URL', and 'Table service SAS URL'. Each field contains a long SAS token string.

And paste into Azure SAS Token field. Click save.



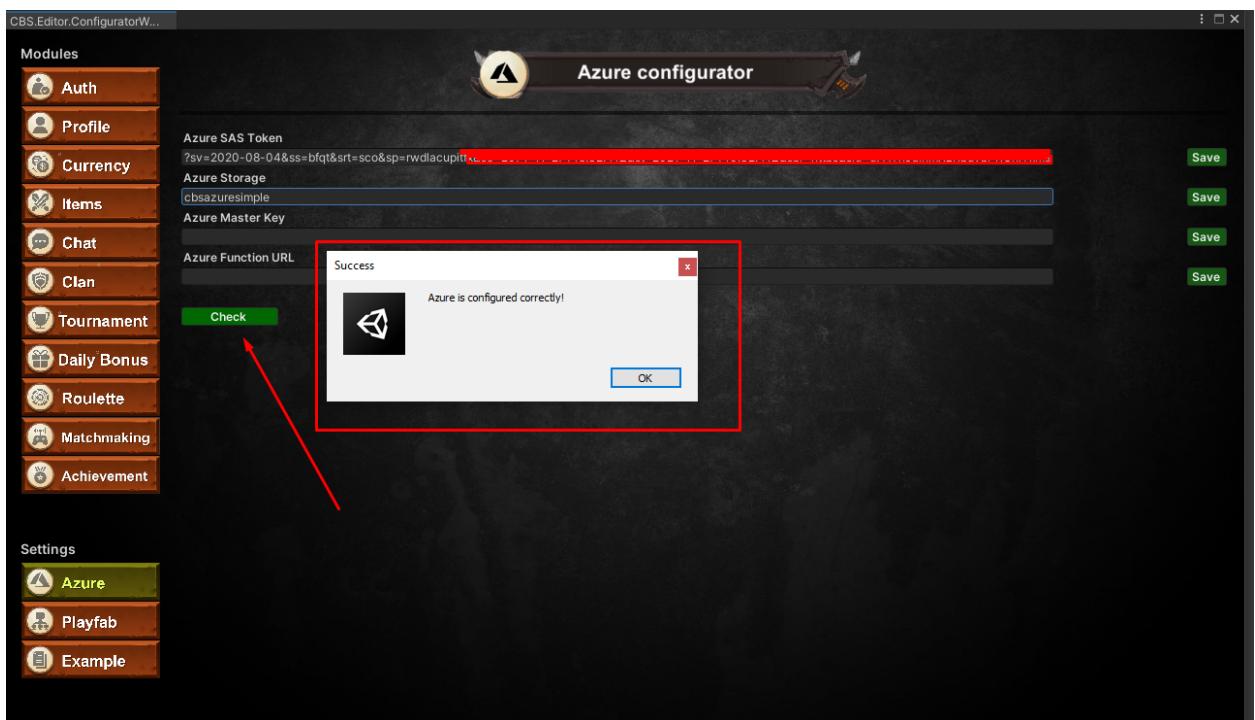
Copy azure storage account name

A screenshot of the Azure Storage Accounts blade. At the top, the URL shows 'Storage accounts > cbsazuresimple'. On the left, there's a navigation menu with 'Storage accounts', 'Create', 'Manage view', 'Filter for any field...', 'Name' (with 'cbsazuresimple' selected), 'Containers', 'File shares', 'Queues', 'Tables', 'Networking', 'Azure CDN', 'Access keys', 'Shared access signature' (which is highlighted with a red arrow), 'Encryption', and 'Security'. The main content area is titled 'cbsazuresimple | Shared access signature'. It includes a search bar, a 'Events' section, a 'Storage browser (preview)' section, and detailed configuration for a shared access signature. The configuration includes sections for 'Allowed services', 'Allowed resource types', 'Allowed permissions', 'Blob versioning permissions', 'Allowed blob index permissions', and 'Start and expiry date/time'. A red arrow also points to the search bar.

And paste into Azure Storage field. Click save.

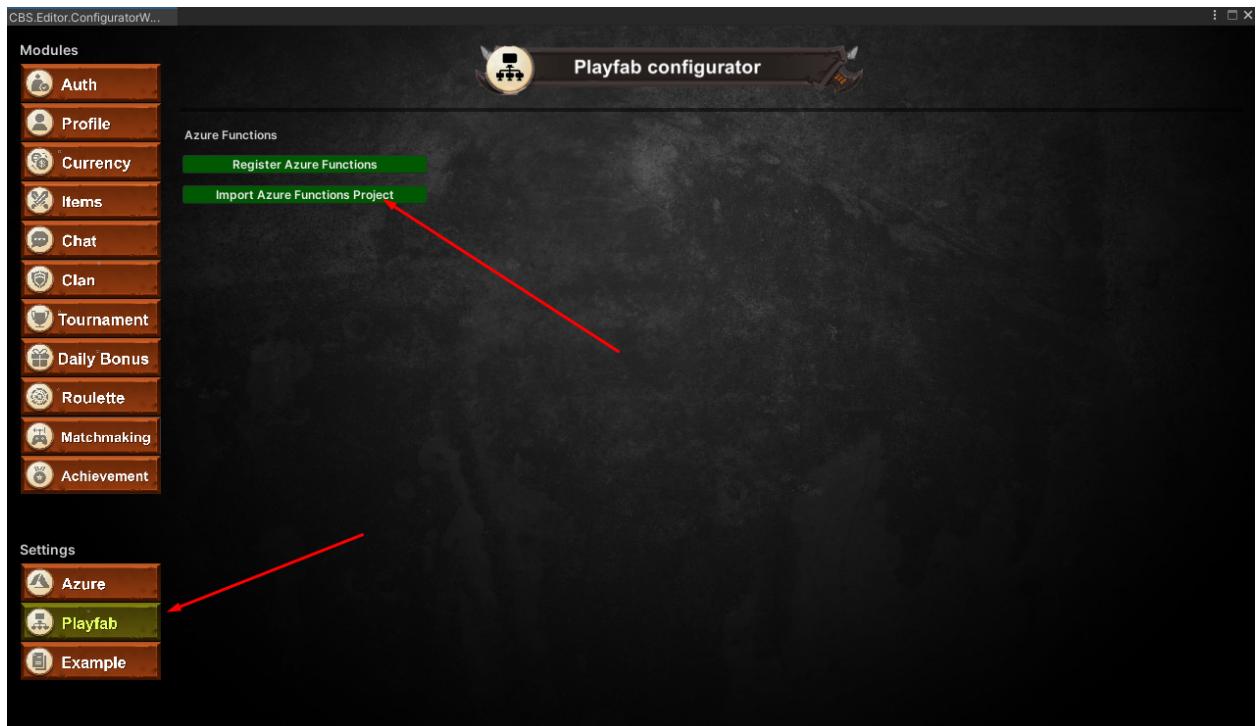


Click "Check" and you will receive a message about successful installation

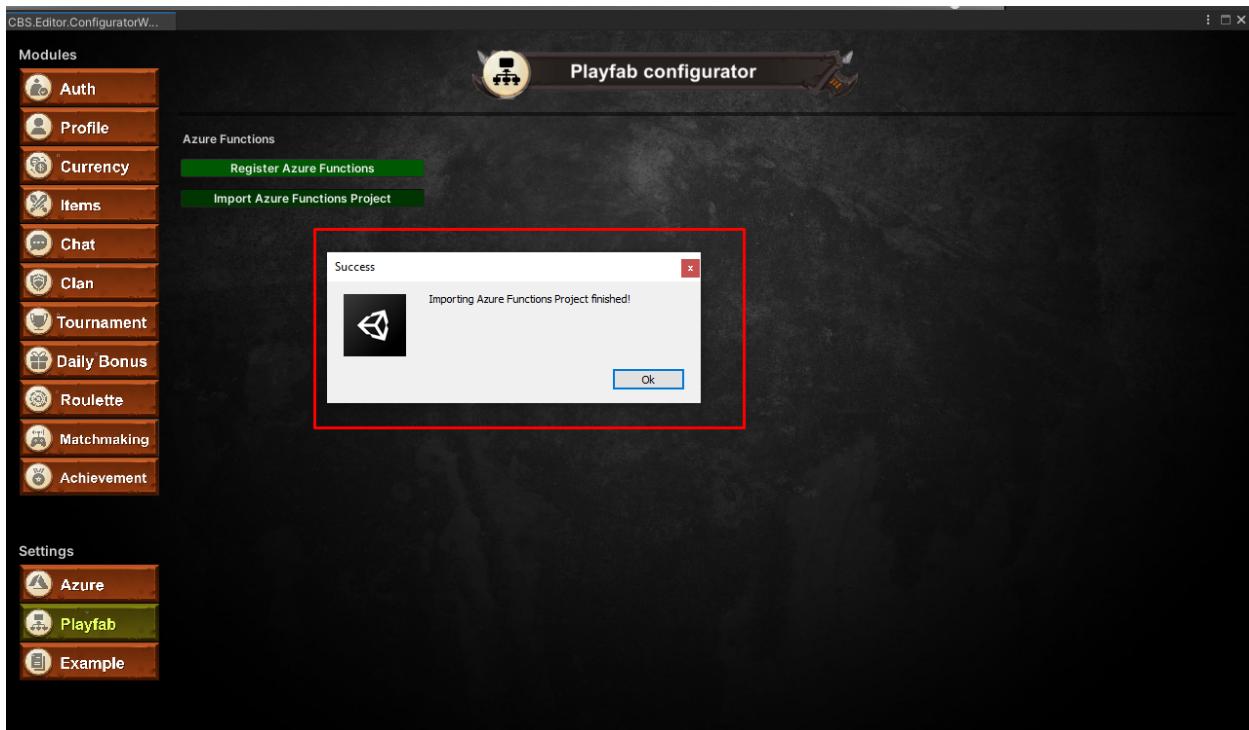


4. Upload Azure functions.

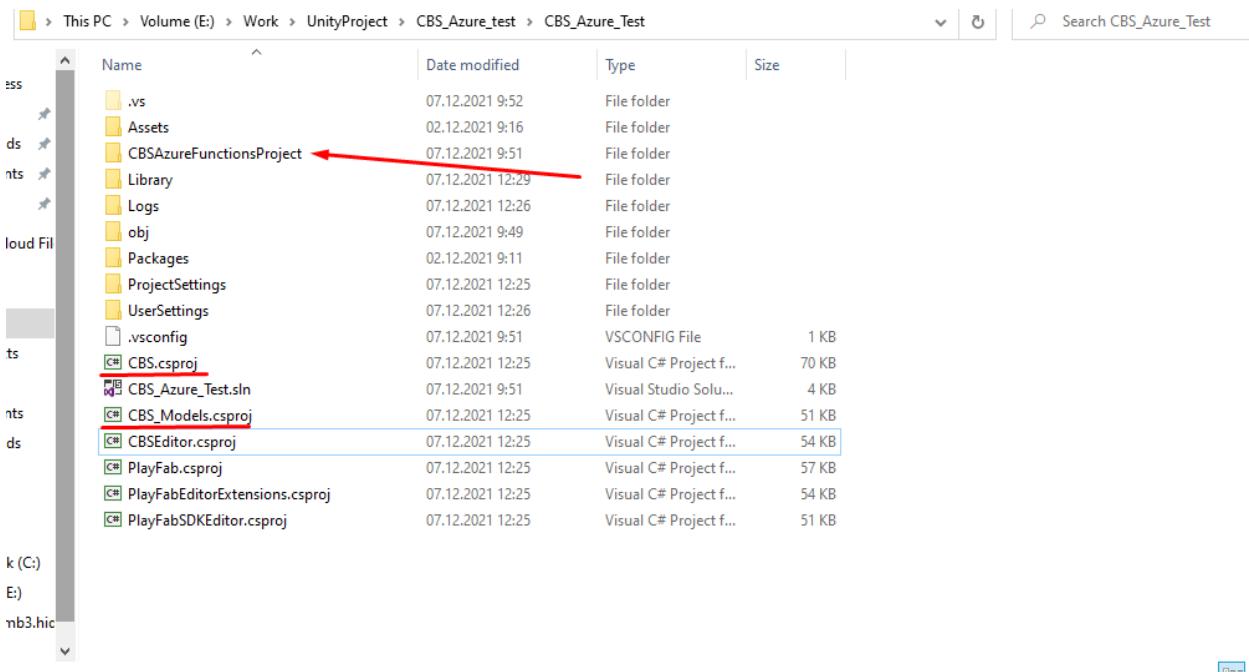
Navigate to "Playfab" tab and click "Import Azure Functions Project".



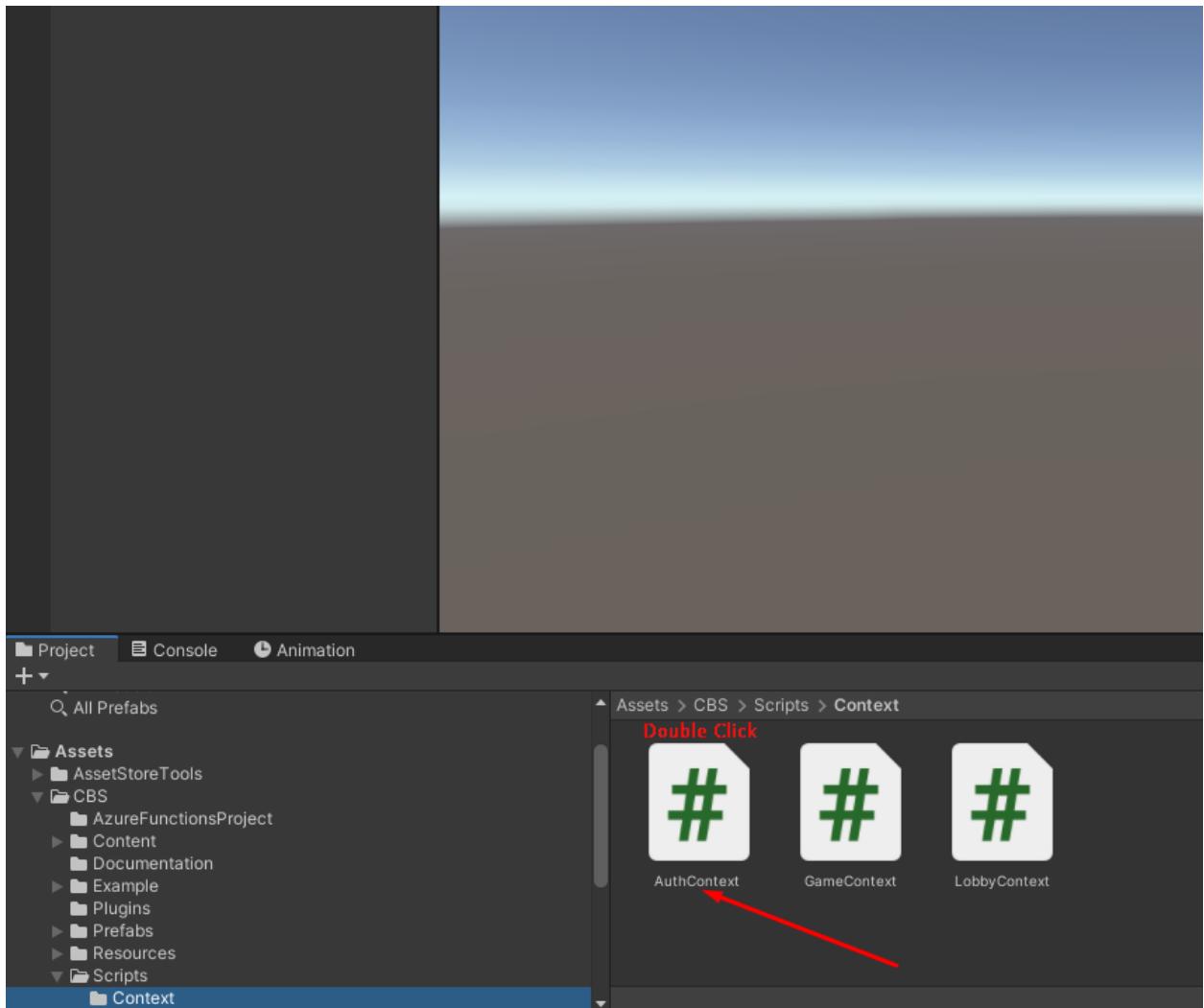
You will receive a message about successful import. This way you will notice that you will have Visual Studio open. It is necessary that VS generated automatically .csproj files for us



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>

② Not sure what to download? [See recommended downloads for the latest version of .NET](#).

6.0.0

[Release notes](#) | Latest release date 2021-11-08

Build apps - SDK

SDK 6.0.100

OS	Installers	Binaries
Linux	Package manager instructions	Arm32 Arm32 Alpine Arm64 Arm64 Alpine x64 x64 Alpine
macOS	Arm64 x64	Arm64 x64
Windows	Arm64 x64 x86	Arm64 x64 x86
All	dotnet-install script	

Visual Studio support
[Visual Studio 2022 \(v17.0\)](#)
[Visual Studio 2022 for Mac \(v17.0\)](#)

Included in
[Visual Studio 17.0](#)

Included runtimes
.NET Runtime 6.0.0
ASP.NET Core Runtime 6.0.0
.NET Desktop Runtime 6.0.0

Language support
C# 10.0
C++

Run apps - Runtime

ASP.NET Core Runtime 6.0.0

The ASP.NET Core Runtime enables you to run existing web/server applications. **On Windows** we recommend installing the [Hosting Bundle](#), which includes the .NET Runtime and IIS support.

IIS runtime support (ASP.NET Core Module v2)
16.0.21299.0

OS	Installers	Binaries
Linux	Package manager instructions	Arm32 Arm32 Alpine Arm64 Arm64 Alpine x64 x64 Alpine
macOS		Arm64 x64
Windows	Hosting Bundle x64 x86	Arm64 x64 x86

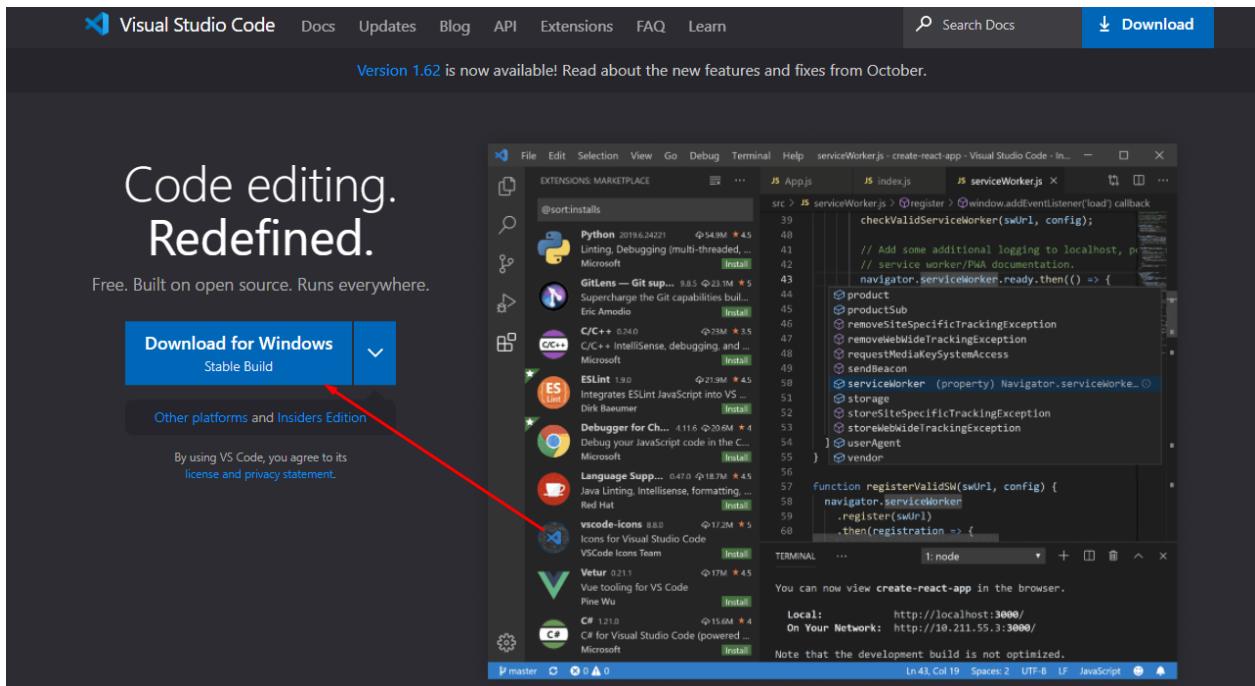
.NET Desktop Runtime 6.0.0

The .NET Desktop Runtime enables you to run existing Windows desktop applications. **This release includes the .NET Runtime: you don't need to install it separately.**

OS	Installers	Binaries
Windows		Arm64 x64 x86

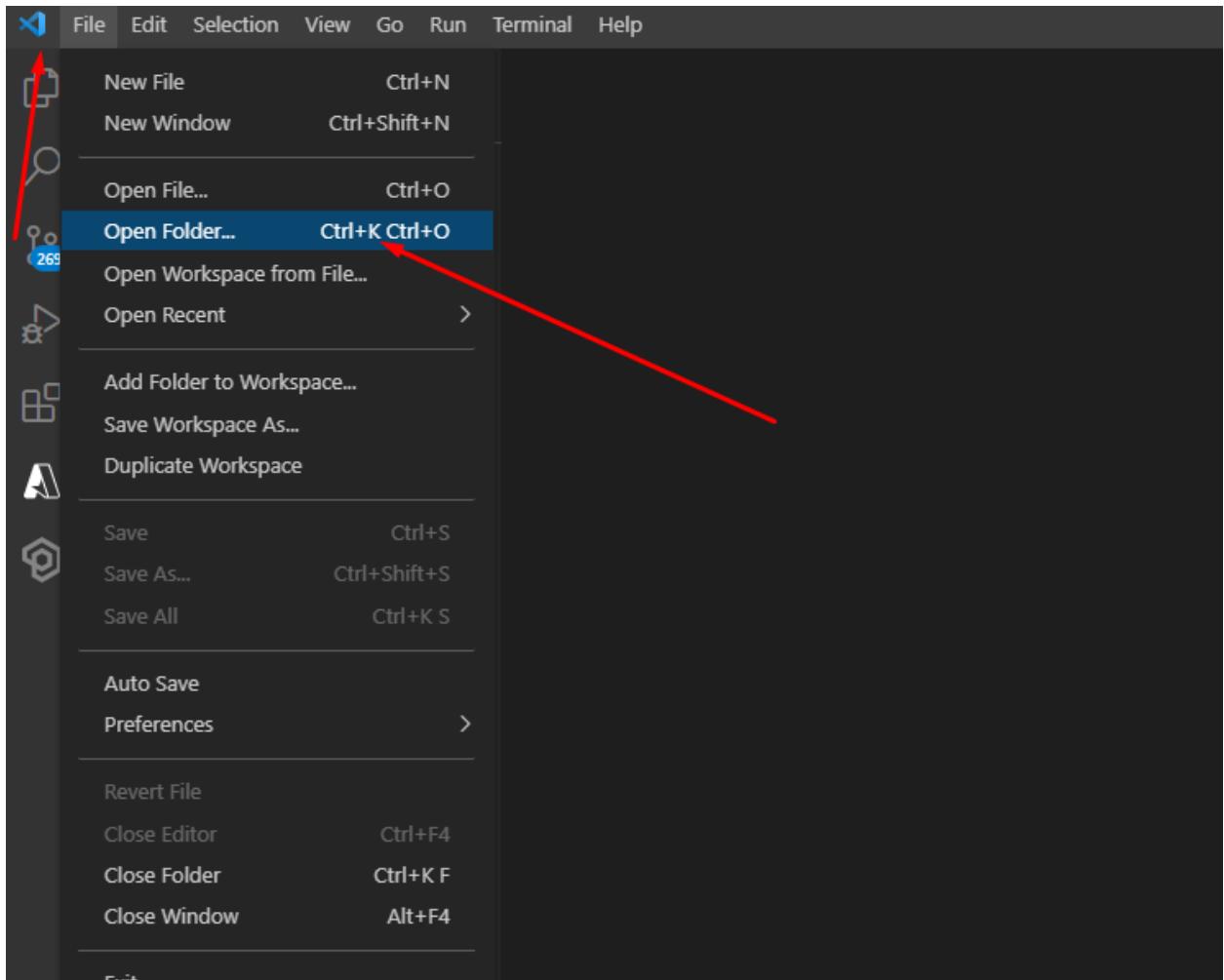
Next, install Visual Studio Code. You can download it here

<https://code.visualstudio.com/>

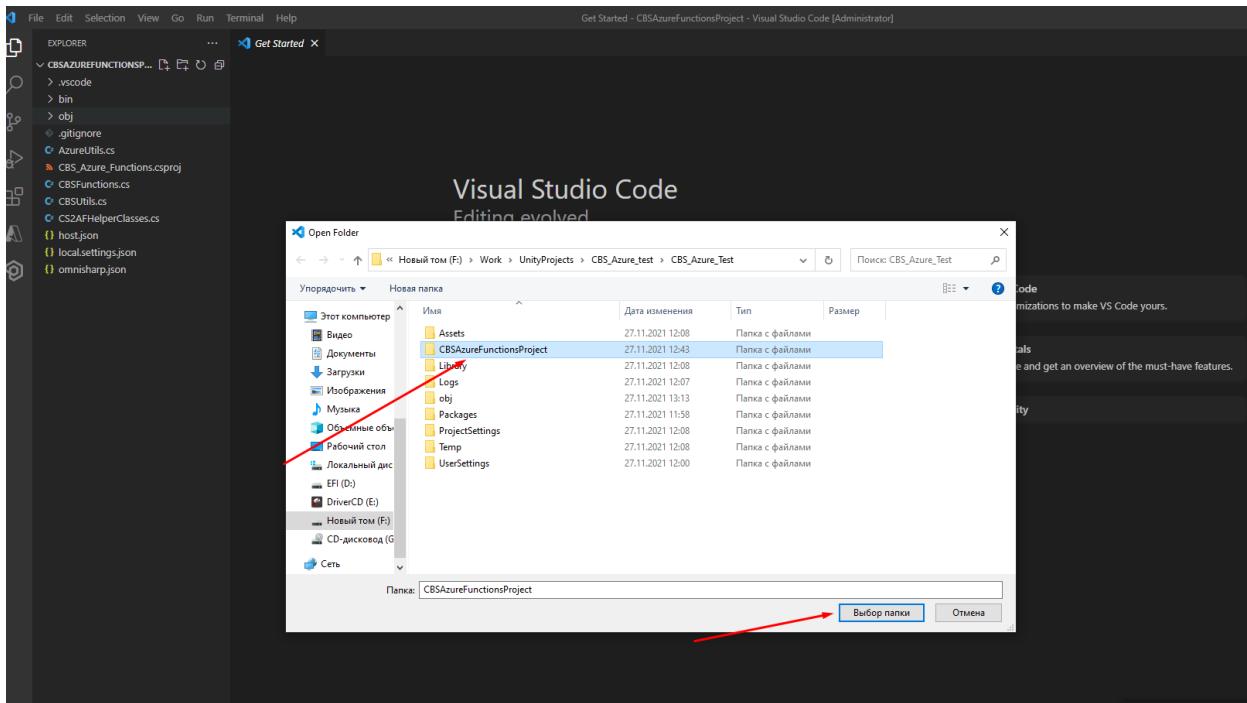


The screenshot shows the Visual Studio Code website. At the top, there's a navigation bar with links for Docs, Updates, Blog, API, Extensions, FAQ, Learn, and a Download button. A message at the top says "Version 1.62 is now available! Read about the new features and fixes from October." Below this, there's a large banner with the text "Code editing. Redefined." and "Free. Built on open source. Runs everywhere." A prominent blue button labeled "Download for Windows" with "Stable Build" underneath it is highlighted with a red arrow. To the right of the banner, there's a code editor window showing some JavaScript code related to service workers. Below the code editor, there's a terminal window with the command "node" and some output. At the bottom, there's a note about the development build not being optimized.

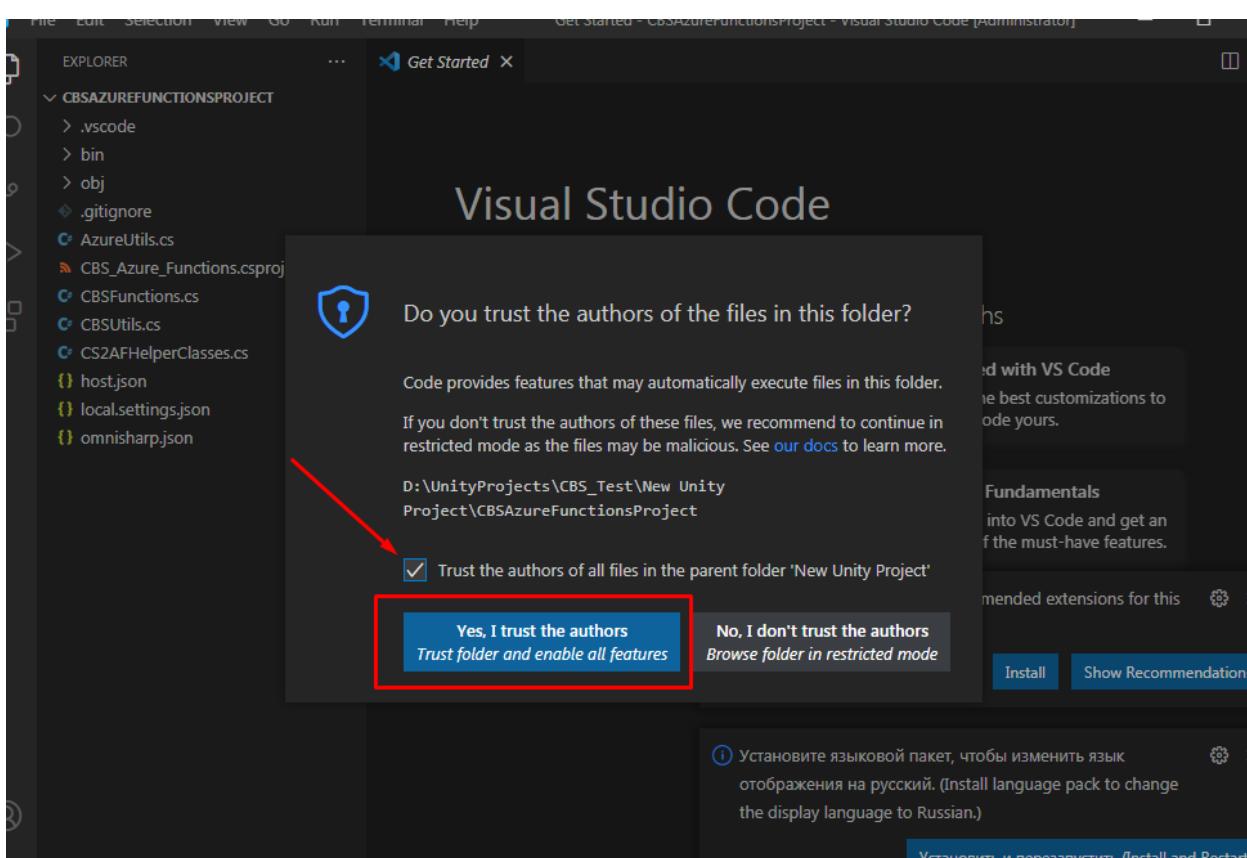
Open VS Code and click "File - Open Folder"



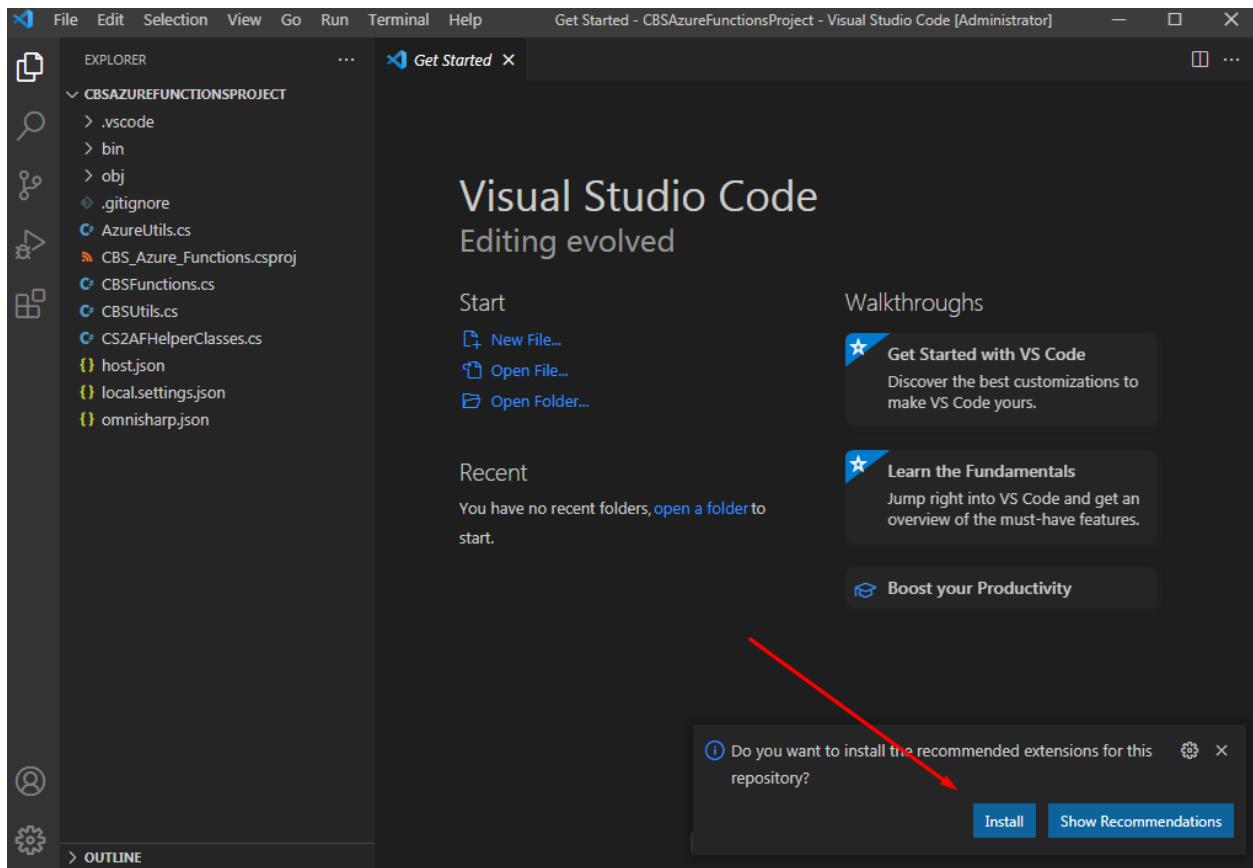
Select the folder of the previously loaded project "CBSAzureFunctionsProject"



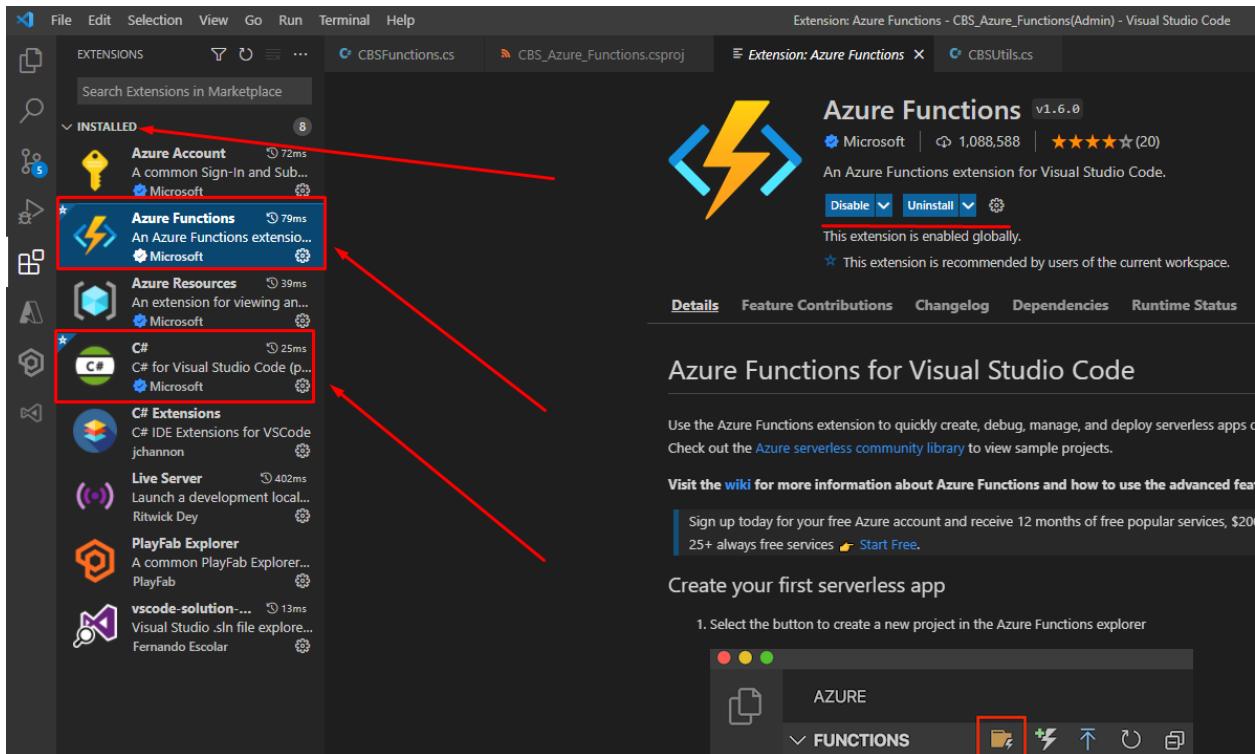
You may have a dialog box that says "**Do you trust the author of the files of this folder?**" Apply check box "**Trust the authors**" and click "**Yes I trust the authors**"



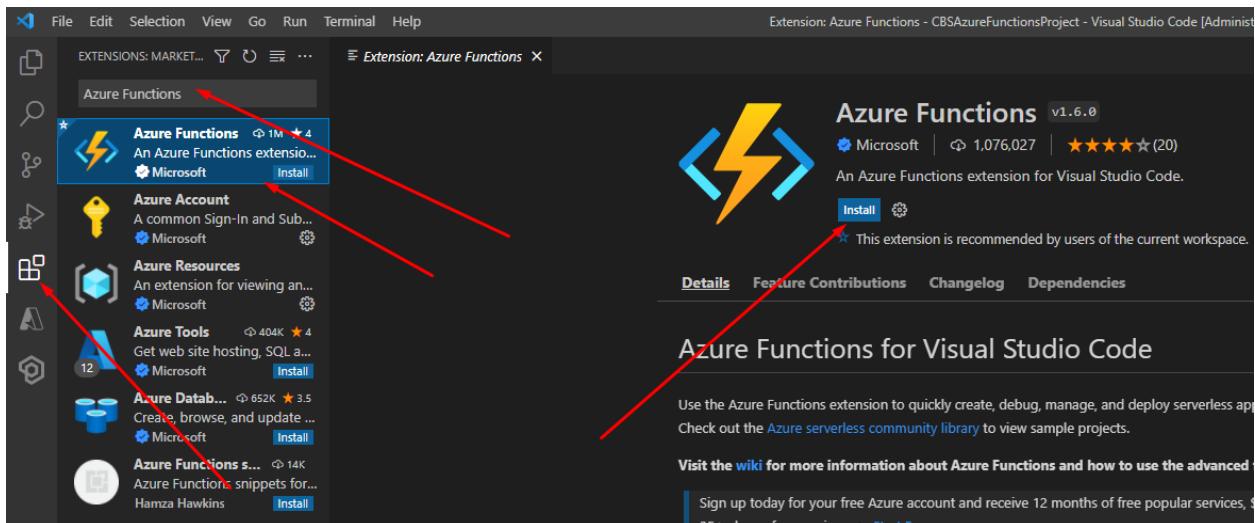
You may have a dialog box that says "**Do you want to install the recommended extensions for this repository?**" Click **Install**



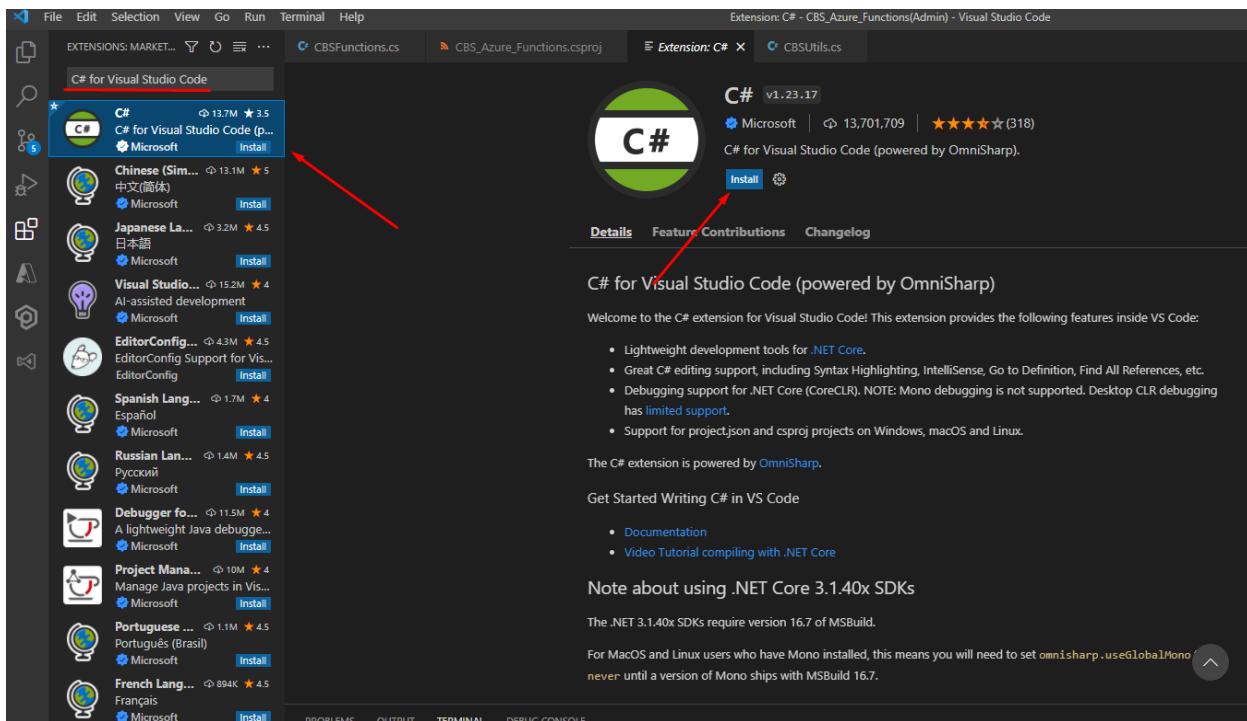
Make sure you have extensions like "**Azure Functions**" and "**C # for Visual Studio Code**" installed.



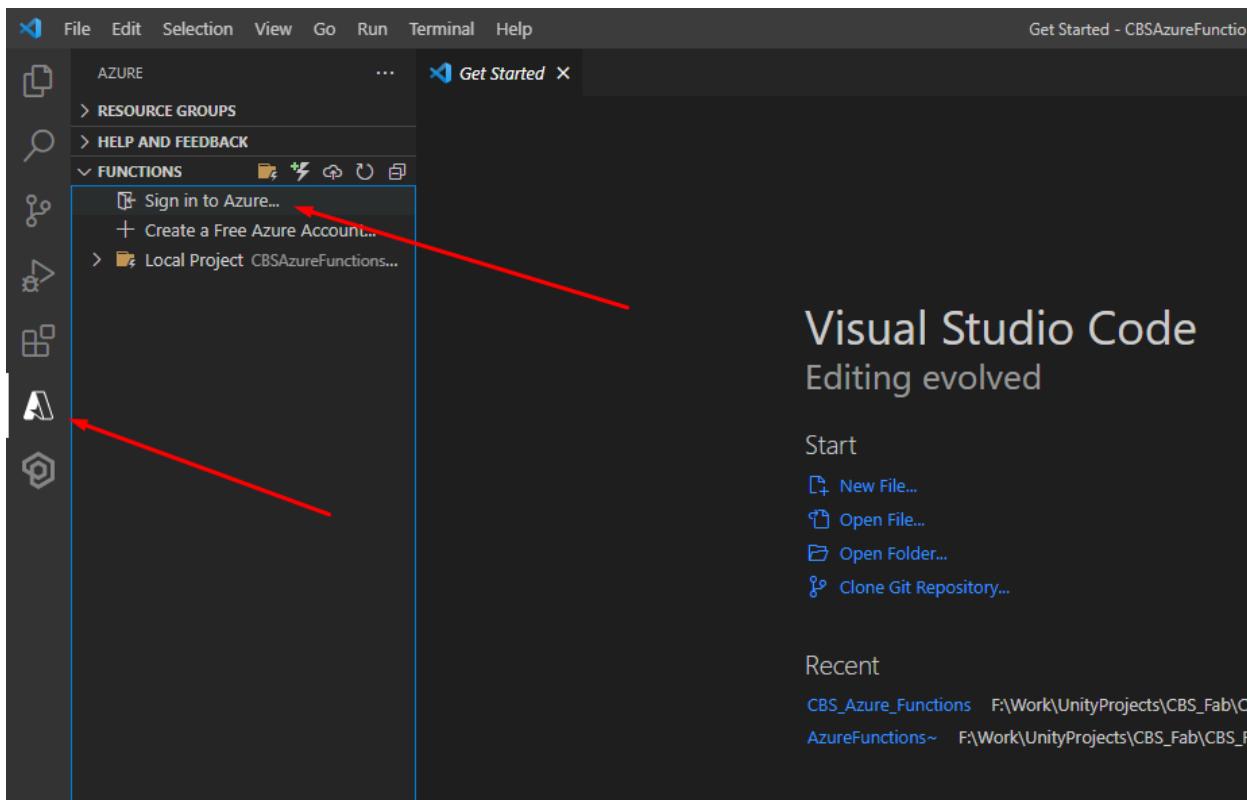
If these extensions are not installed, when you need to install them manually. Navigate to "Extensions" tab. Search "Azure Functions" extension and click "Install"



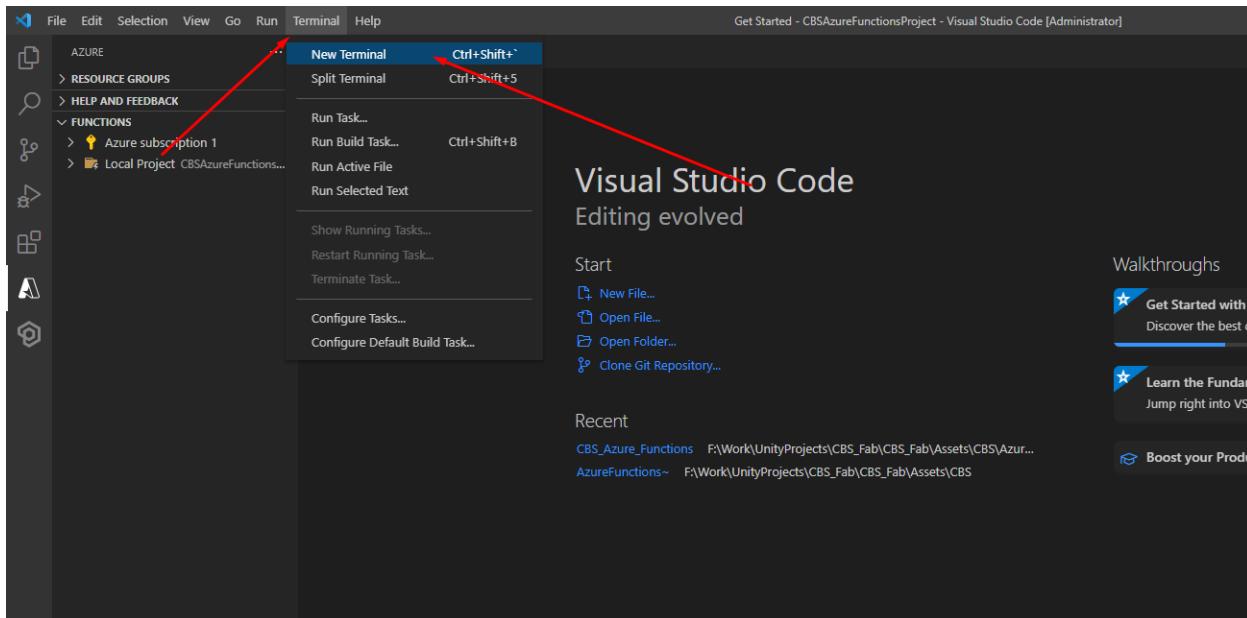
Do the same with **C# for Visual Studio Code** if it not exist.



Navigate to Azure tab and sign in using login password



After successful login - click Terminal -> New Terminal

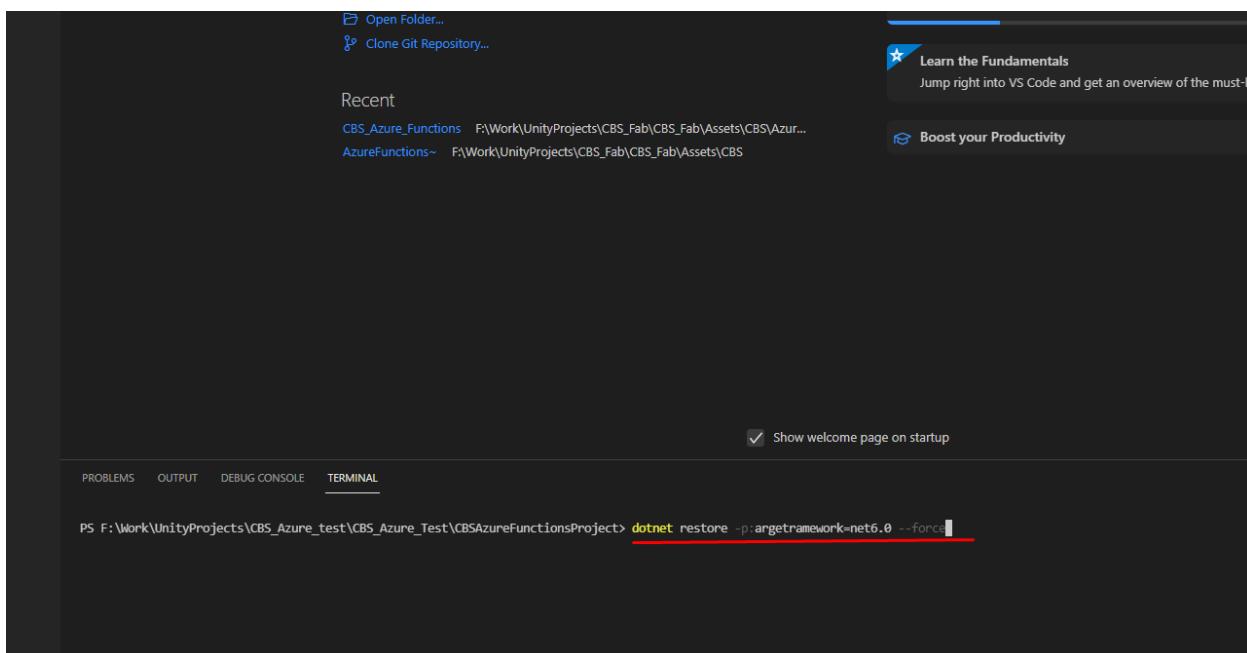


In a terminal window enter the command **dotnet restore -p: TargetFramework = net6.0 --force** and press Enter

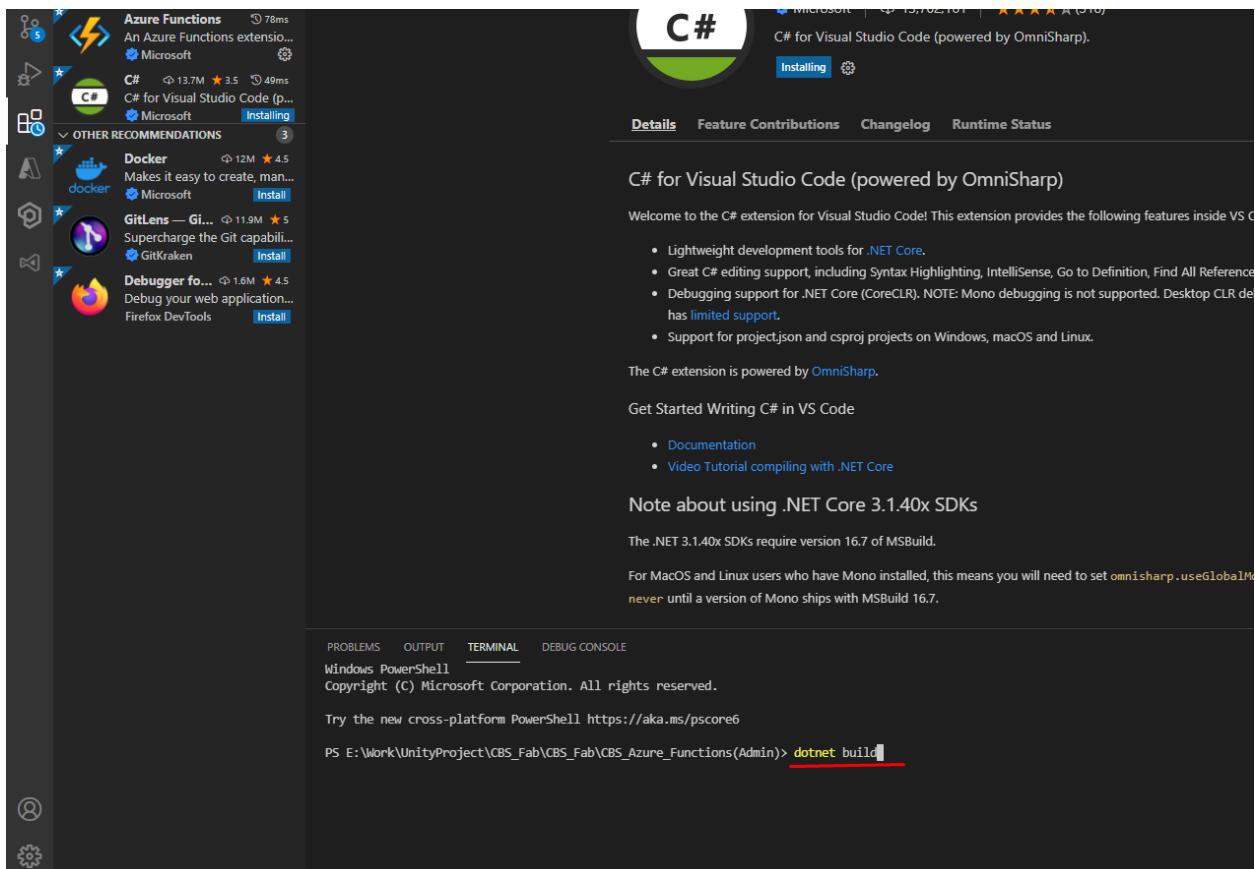
Warning. If you get an error like "**dotnet: The term 'dotnet' is not recognized as the name of a cmdlet, function, script file, or operable program**" most likely dotnet was not automatically added to the system's Environment Variables.

To fix this, watch the video on how to add dotnet to Environment Variables
https://youtu.be/jSvL_Rnm_L8

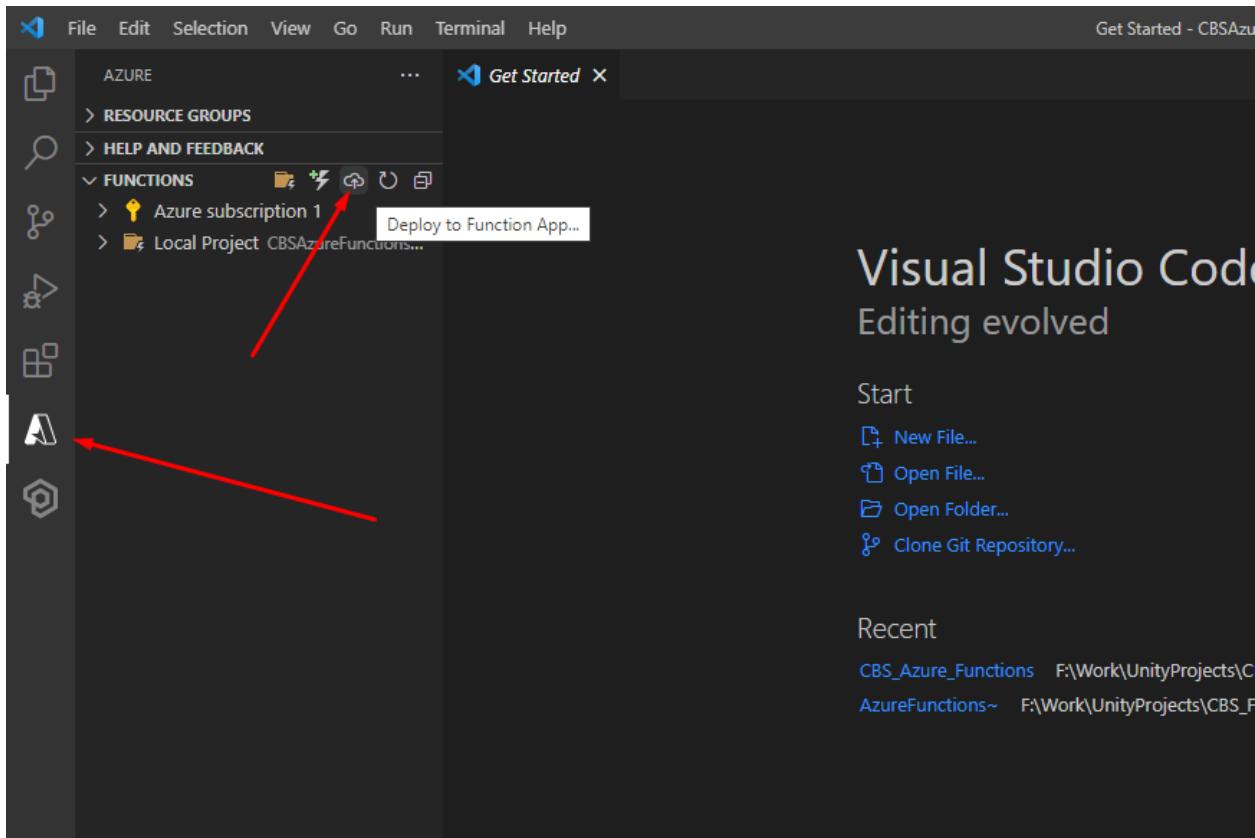
After that restart VS Code and repeat the step again



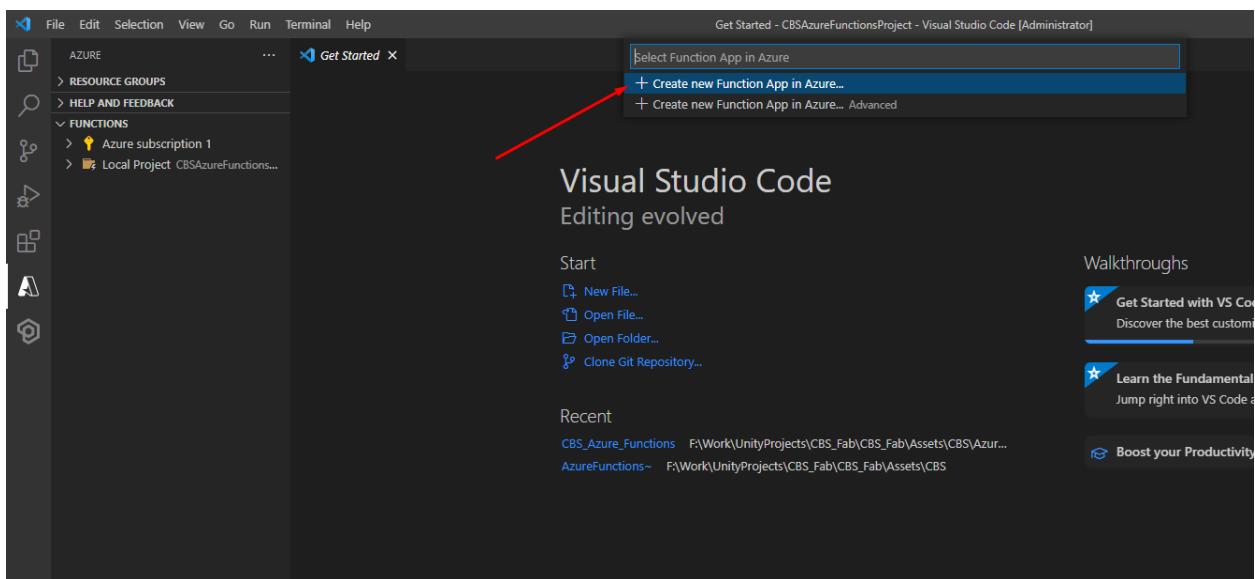
Also run **dotnet build** command



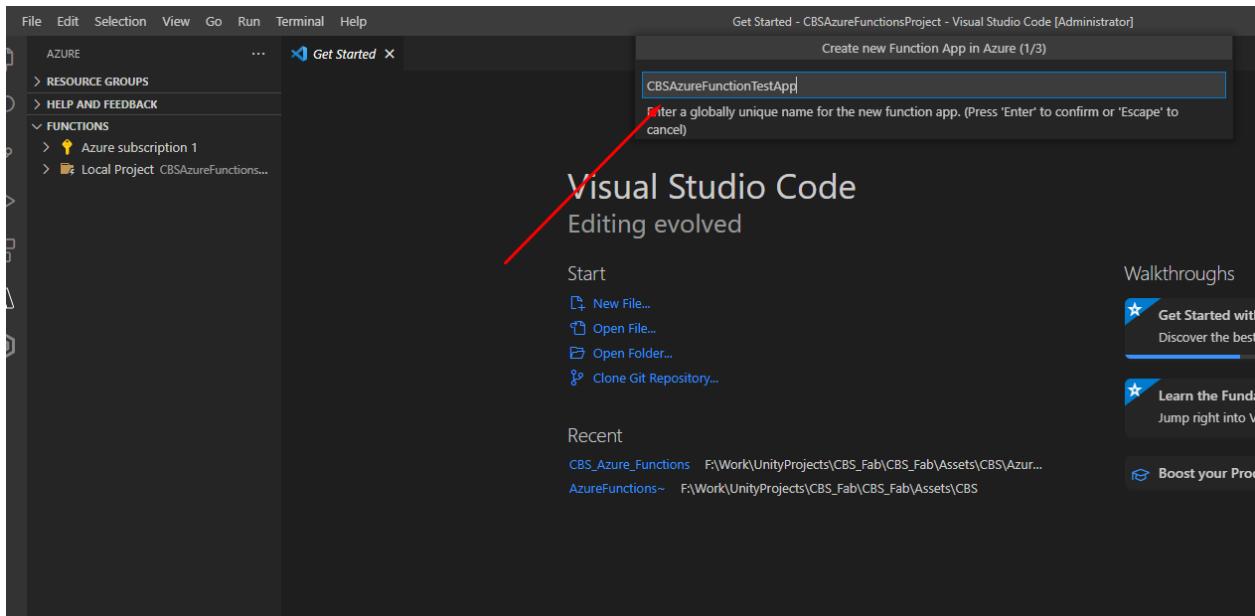
After succes build - let's upload the functions to Azure. In the "Azure" tab - click "Deploy To Function App"



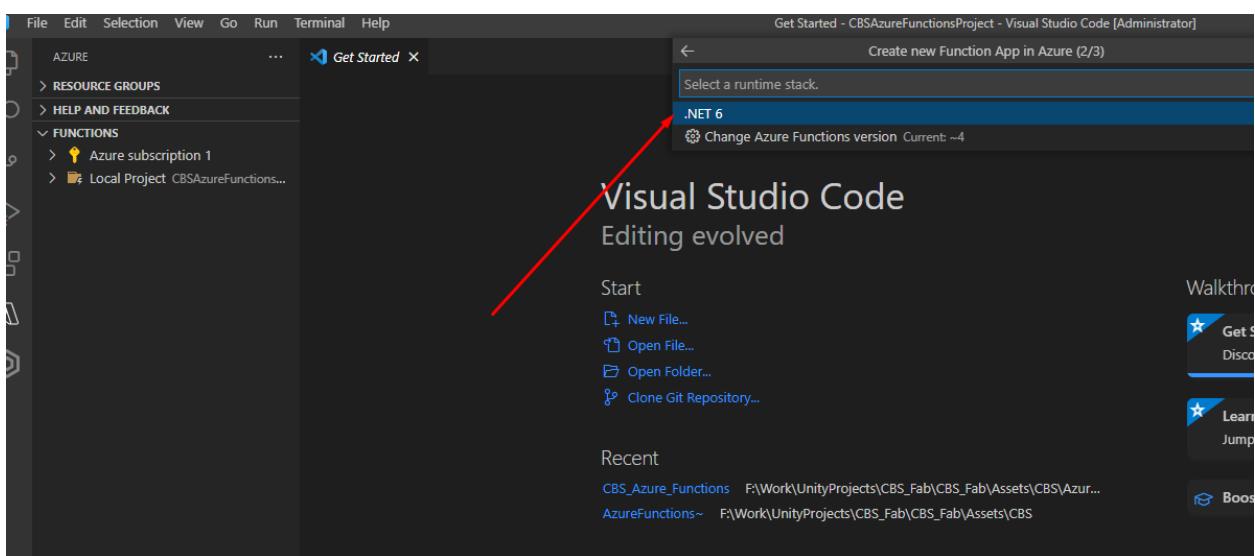
Click "Create new Function App in Azure"



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

The screenshot shows the Visual Studio Code interface. At the top, there's a dark header bar with some icons and text. Below it is a sidebar titled "Walkthroughs" containing three items: "Get Started with VS Code", "Learn the Fundamentals", and "Boost your Productivity". The main area is a terminal window with the following content:

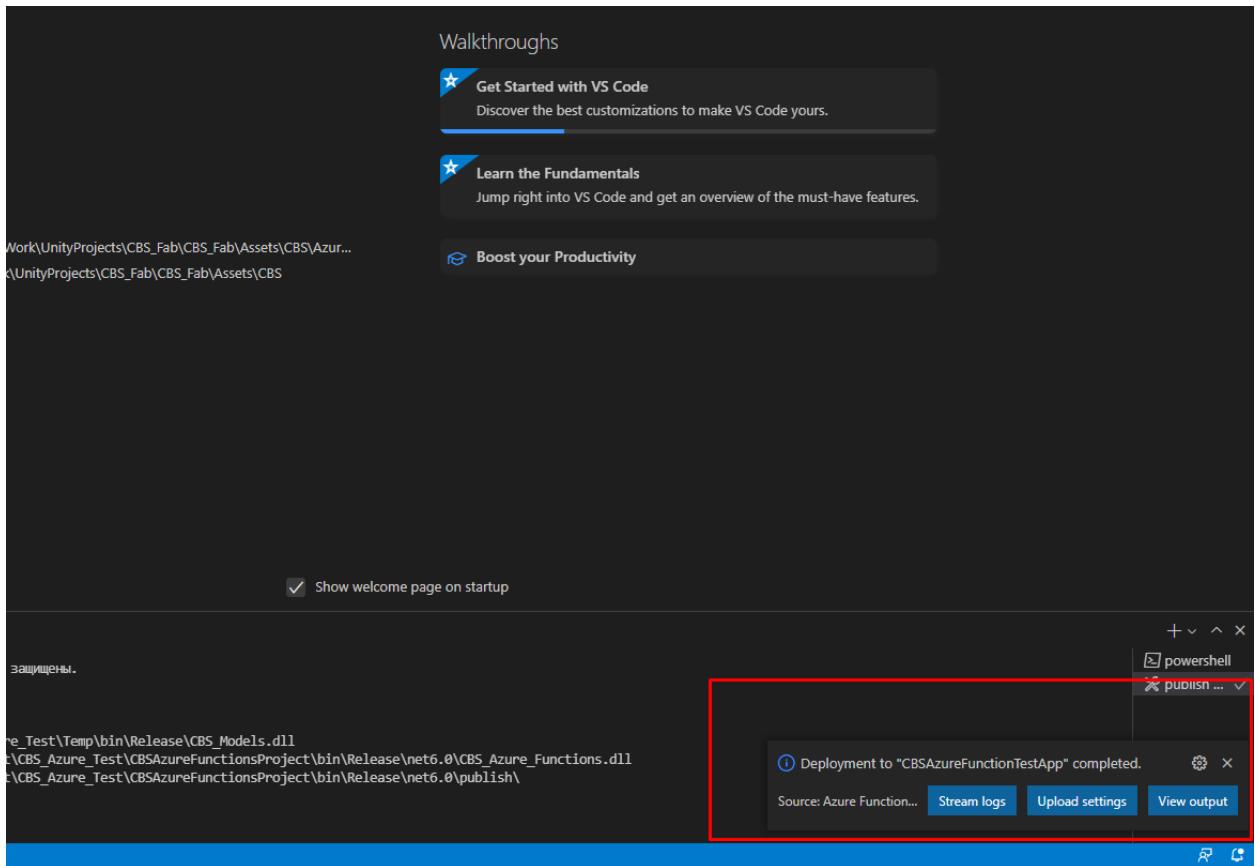
```
> dotnet restore -p:targetframework=net6.0 --force
cba\Assets\CBS_Azure_Functions.csproj (aa 905 ms).
> [redacted]
```

A red box highlights the last line of the terminal output, which reads:

Creating storage account "cbsazurefunctiontestapp" in
location "West Europe" with sku "Standard_LRS" ... (4/6)

Source: Azure Functions (Extension)

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

Microsoft Azure

cbsazuresimple | Events

Storage account

Search (Ctrl+ /)

Event Subscription Refresh

Overview Activity log Tags Diagnose and solve problems Access Control (IAM) Data migration Events Storage browser (preview)

Build reactive, event-driven apps with a fully managed event routing service that is built into Azure. Event Grid helps you build automation into your cloud infrastructure, create serverless apps, and integrate across services and clouds. [Learn more](#)

Logic Apps Azure Function Azure Data Explorer More Options

Trigger serverless Azure Functions. [Learn more](#)

Trigger Azure Functions

React to events from your storage account and execute code using Azure Functions. [Learn more](#)

Create

Click on the function app and copy the function url

Microsoft Azure

Home > cbsazuresimple >

CBSAzureFunctionTestApp

Function Apps

Search All subscriptions Function Apps CBSAzureFunctionTestApp...

The classic Azure Functions management experience is deprecated and will be taken offline in October 2020. If you are experiencing issues with the new experience, please click here to file an issue.

Overview Platform features

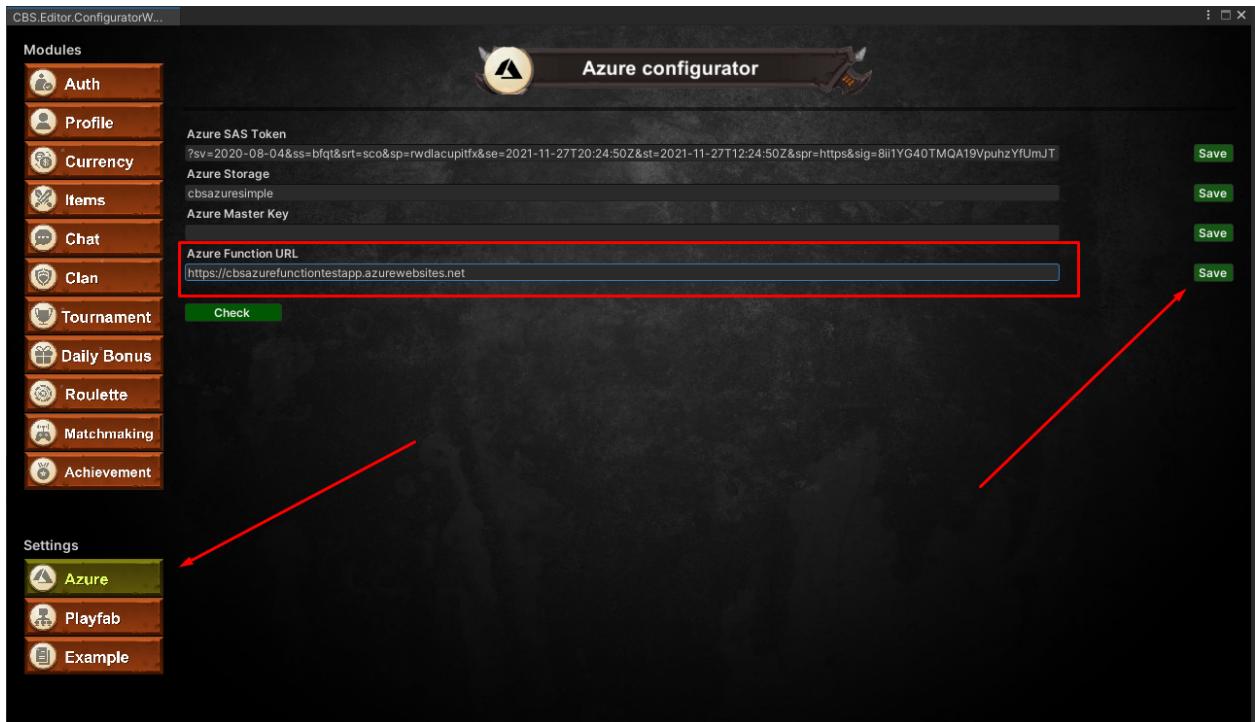
Stop Swap Restart Get publish profile Reset publish profile Download app content Delete

Status Running	Subscription Azure subscription 1	Resource group cbsazurefunctiontestapp	URL https://cbsazurefunctiontestapp.azurewebsites.net
	Subscription ID 32fe042d-2a97-4c1e-8b1d-1b0e8bd64235	Location West Europe	App Service plan / pricing tier ASP-CBSAzureFunctionTestApp-f590 (Consumption)

Configured features

- Function app settings
- Configuration
- Application Insights

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"

The screenshot shows the Azure Portal's 'Function Apps' blade. On the left, there's a tree view of function apps under 'CBSAzureFunctionTestApp'. One node, 'Functions (Read Only)', is expanded, showing several function names like 'AcceptClanInvite', 'AcceptFriend', etc. In the main pane, the 'Overview' tab is selected for the 'CBSAzureFunctionTestApp'. It shows details such as 'Status: Running', 'Subscription: Azure subscription 1', 'Resource group: cbsazurefunctiontestapp', 'URL: https://cbsazurefunctiontestapp.azurewebsites.net', 'Subscription ID: 32fe042d-2a97-4c1e-8b1d-1b0e8bd64235', 'Location: West Europe', and 'App Service plan / pricing tier: ASP-CBSAzureFunctionTestApp-f590 (Consumption)'. Below this, there's a section titled 'Configured features' with links for 'Function app settings' (which is highlighted with a red arrow), 'Configuration', and 'Application Insights'.

Copy master key

Manage application settings

Runtime version

Runtime version loaded from Application Settings: ~4

Unsupported Runtime Version
Your custom runtime version is not supported. As a result 4.0.1.16815 runtime is being used.

Cannot Upgrade with Existing Functions
Major version upgrades can introduce breaking changes to languages and bindings. When upgrading major versions of the runtime, consider creating a new function app and migrate your functions to this new app.

~1 ~2 ~3

Function app edit mode

Change the edit mode of your function app

! Your app is currently in read only mode because you are running from a package file. To make any changes update the content in your zip file and WEBSITE_RUN_FROM_PACKAGE app setting.

Read/Write Read Only

Host Keys (All functions)

NAME	VALUE	ACTIONS
_master	Click to show	<input type="button" value="Copy"/> <input type="button" value="Renew"/>
default	Click to show	<input type="button" value="Copy"/> <input type="button" value="Renew"/> <input type="button" value="Revoke"/>

Add new host key

host.json

```

1 {
2   "version": "2.0",
3   "logging": {
4     "applicationInsights": {
5       "samplingSettings": {}
6     }
7   }
8 }
```

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

CBS Editor.ConfiguratorW...

Azure configurator

Modules

- Auth
- Profile
- Currency
- Items
- Chat
- Clan
- Tournament
- Daily Bonus
- Roulette
- Matchmaking
- Achievement

Azure SAS Token
?sv=2020-08-04&ss=bfqt&srt=sc0&sp=rwdiacupitfx&se=2021-11-27T20:24:50Z&st=2021-11-27T12:24:50Z&spr=https&sig=8ii1YG40TMQA19VpuhzYfUmJT

Azure Storage
cbsazuresimple

Azure Master Key
v07ngjik5Svn0dBChZjXK/achGXyJX9RchFYYHh2hNege/LR7BWQQ==

Azure Function URL
<https://cbsazurefunctiontestapp.azurewebsites.net>

Check

Settings

- Azure**
- Playfab
- Example

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

The screenshot shows the 'API Features' tab of the PlayFab API Features page. A red arrow points from the 'Edit title info' link in the left sidebar to the 'Title ID' field, which contains '79CD8'. Another red arrow points from the 'Secret Keys' tab to the 'Policy JSON' section on the right, where the following JSON code is displayed:

```

1  [ {
2    "Action": "*",
3    "Effect": "Allow"
4    "Resource": "*-*"
5    "Principal": "*"
6    "ChildOf": {
7      "EntityType": "EntityId"
8      "EntityId": "*"
9    }
10  },
11  {
12    "Comment": "The c full access",
13    "Condition": {
14      "CallingEntity": "*"
15    }
16  },
17  {
18    "Action": "*",
19    "Effect": "Allow"
20    "Resource": "pfrr"
21    "Principal": "*"
}

```

Go back to the Azure Portal and click "Configuration"

The screenshot shows the 'CBSAzureFunctionTestApp' function app in the Azure portal. A red arrow points from the 'Functions (Read Only)' list to the 'Configuration' tab in the 'Configured features' section. The 'Configuration' tab is highlighted.

Click "New Application Setting"

Microsoft Azure

Search resources, services, and docs (G+)

Home > Storage accounts > cbsazuresimple > CBSAzureFunctionTestApp > Configuration ...

[Application settings](#) [Function runtime settings](#) [General settings](#)

Application settings

Application settings are encrypted at rest and transmitted over an encrypted channel. You can choose to display them in plain text in your browser by using the controls below. Application Settings are exposed as environment variables for access by your application at runtime

+ New application setting [Show values](#) [Advanced edit](#)

Name	Value	Source	Deployment slot setting	Delete
APPINSIGHTS_INSTRUMENTATIONKEY	(Hidden value. Click to show value)	App Service Config		
AzureWebJobsStorage	(Hidden value. Click to show value)	App Service Config		
FUNCTIONS_EXTENSION_VERSION	(Hidden value. Click to show value)	App Service Config		
FUNCTIONS_WORKER_RUNTIME	(Hidden value. Click to show value)	App Service Config		
WEBSITE_CONTENTAZUREFILECONNECTIONSTRING	(Hidden value. Click to show value)	App Service Config		
WEBSITE_CONTENTSHARE	(Hidden value. Click to show value)	App Service Config		
WEBSITE_RUN_FROM_PACKAGE	(Hidden value. Click to show value)	App Service Config		

Connection strings

Connection strings are encrypted at rest and transmitted over an encrypted channel. Connection strings should only be used with a function app if you are using entity framework. For other scenarios use App Settings. [Learn more](#)

+ New connection string [Show values](#) [Advanced edit](#)

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

Microsoft Azure

Search resources, services, and docs (G+)

Home > Storage accounts > cbsazuresimple > CBSAzureFunctionTestApp > Configuration ...

[Application settings](#)

Application settings are encrypted at rest and transmitted over an encrypted channel. You can choose to display them in plain text in your browser by using the controls below. Application Settings are exposed as environment variables for access by your application at runtime

+ New application setting [Show values](#) [Advanced edit](#)

Add/edit application setting

Name:

Value:

Deployment slot setting

OK **Cancel**

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

My Game

Development 1/100K

General API Features Secret Keys Email Preferences Push Notifications Limits Client Profile Options OpenID Connect Data Collection Economy

Secret Keys

Secret keys are used to authenticate all calls to the PlayFab [server APIs](#) or [admin APIs](#). Never distribute a secret key to an untrusted source. Generate a new key for each application or game server so you can revoke them if needed.

Delete

Status	Name	Secret key	Expiration date (UTC)
Active	Default Key	NFABGZQBYZHEE9W9UA68I6YEZR1RBBG1A7S...	

New secret key

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

Microsoft Azure

Search resources, services, and docs (G/)

Home > Storage accounts > cbsazuresimple > CBSAzureFunctionTestApp >

Configuration

Refresh Save Discard Leave Feedback

Application settings are encrypted at rest and transmitted over an encrypted channel. You can choose to display them in plain text in your browser by using the controls below

+ New application setting Show values Advanced edit

Filter application settings

Name	Value
APPINSIGHTS_INSTRUMENTATIONKEY	(Hidden value. Click to show value)
AzureWebJobsStorage	(Hidden value. Click to show value)
FUNCTIONS_EXTENSION_VERSION	(Hidden value. Click to show value)
FUNCTIONS_WORKER_RUNTIME	(Hidden value. Click to show value)
PLAYFAB_TITLE_ID	(Hidden value. Click to show value)
WEBSITE_CONTENTAZUREFILECONNECTIONSTRING	(Hidden value. Click to show value)
WEBSITE_CONTENTSHARE	(Hidden value. Click to show value)
WEBSITE_RUN_FROM_PACKAGE	(Hidden value. Click to show value)

Connection strings

Connection strings are encrypted at rest and transmitted over an encrypted channel. Connection strings should only be used with a function app if you are using entity framework

+ New connection string Show values Advanced edit

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

Microsoft Azure

Home > Storage accounts > cbsazuresimple > CBSAzureFunctionTestApp >

Configuration ...

Application settings

Application settings are encrypted at rest and transmitted over an SSL connection.

+ New application setting

Filter application settings

Name
Value

Deployment slot setting

Name
APPINSIGHTS_INSTRUMENTATIONKEY
AzureWebJobsStorage
FUNCTIONS_EXTENSION_VERSION
FUNCTIONS_WORKER_RUNTIME
PLAYFAB_TITLE_ID
WEBSITE_CONTENTAZUREFILECONNECTIONSTRING
WEBSITE_CONTENTSHARE
WEBSITE_RUN_FROM_PACKAGE

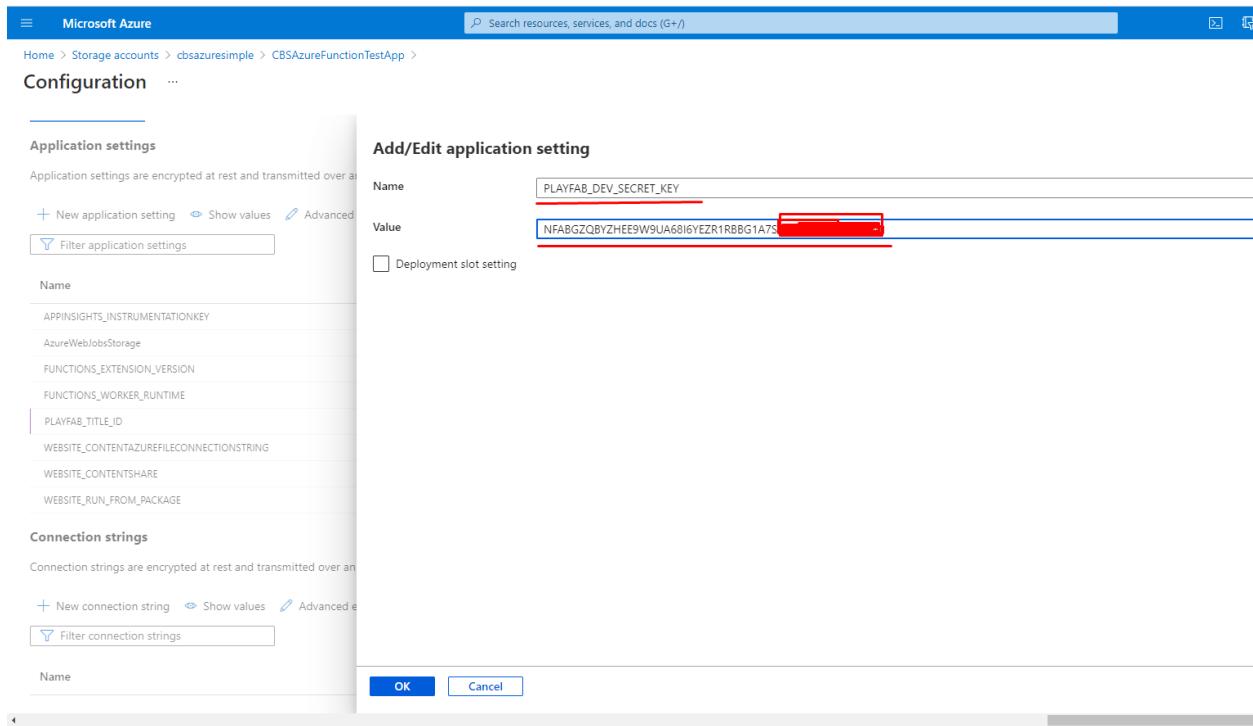
Connection strings

Connection strings are encrypted at rest and transmitted over an SSL connection.

+ New connection string

Filter connection strings

Name



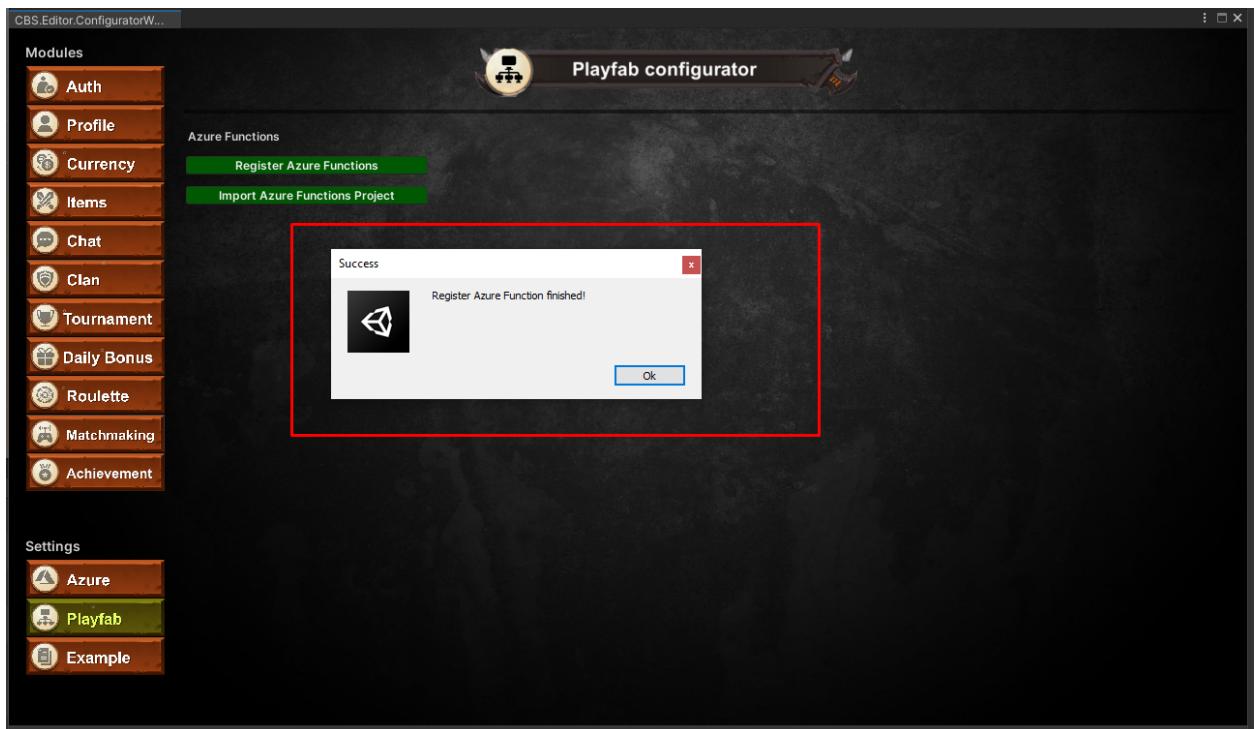
Finally click **Save**



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



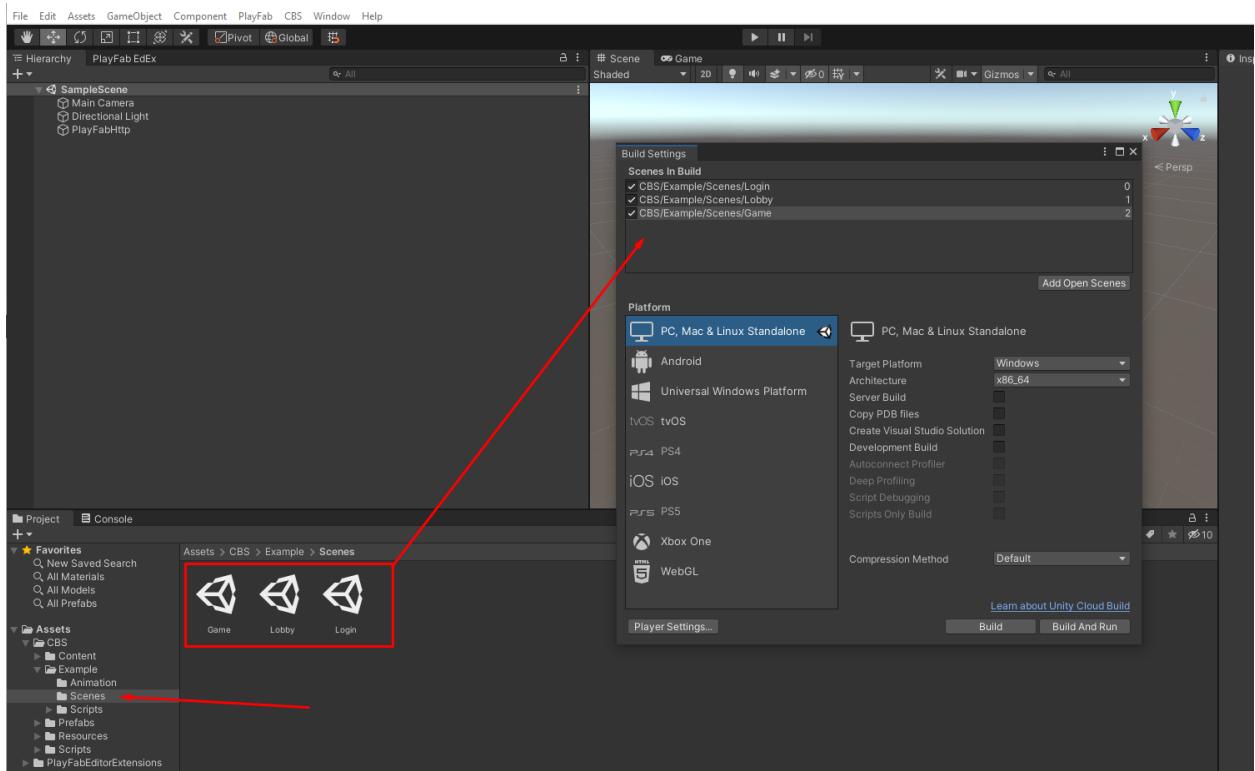
Finally, you will receive a message about the successful registration of functions on the Playfab.



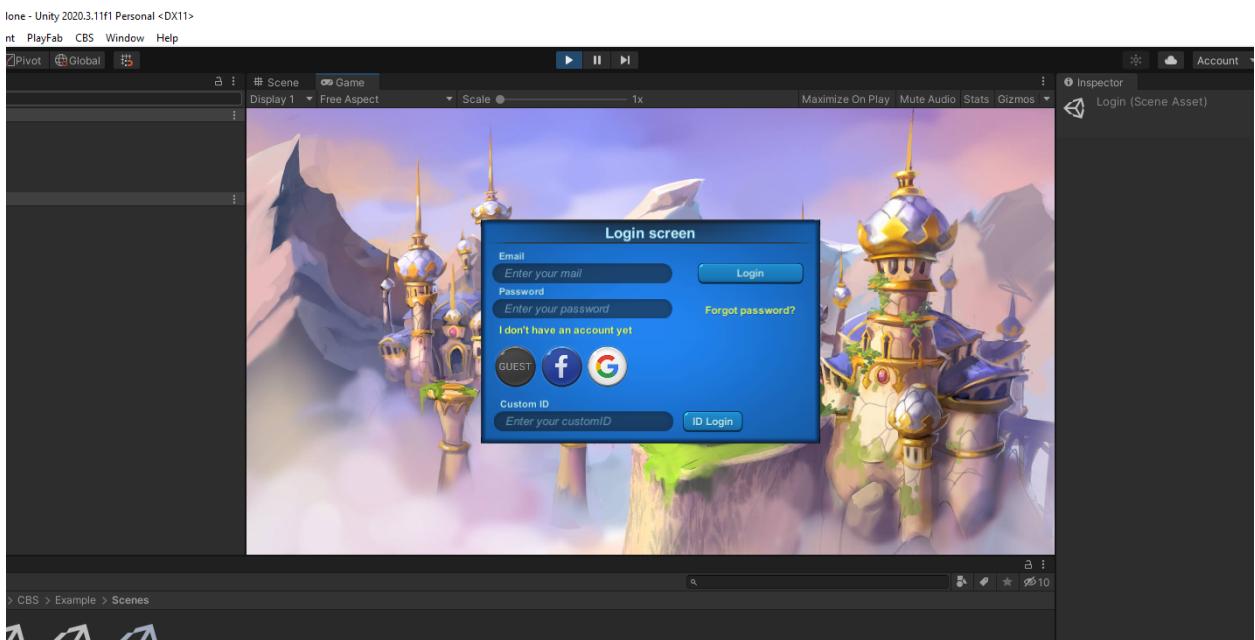
Congratulations, you have successfully finished installing the plugin!

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>