

VISUAL STUDIO CODE LAB

LAB PREREQUISITES

If you want to do this lab using GitHub CodeSpaces:

- Have a GitHub Alias
- Create your own GitHub repository
- Download the samples solution and the PCF control from the lab GitHub repository
- Access to CodeSpaces
- Access to a Power platform environment (new environment is preferred)
 - The type of environment does not matter (It can be Trial, Time bound Trial, Sandbox, and Production)
 - Import the Collaboration.zip file into your repository

If you want to do this lab without GitHub CodeSpaces

- Have a GitHub Alias
- Create your own GitHub repository
- Access to a Power platform environment (new environment is preferred)
 - The type of environment does not matter (It can be Trial, Time bound Trial, Sandbox, and Production)
 - Import the Collaboration.zip file into your repository
- Have npm and node installed on your machine

INSTALLATION

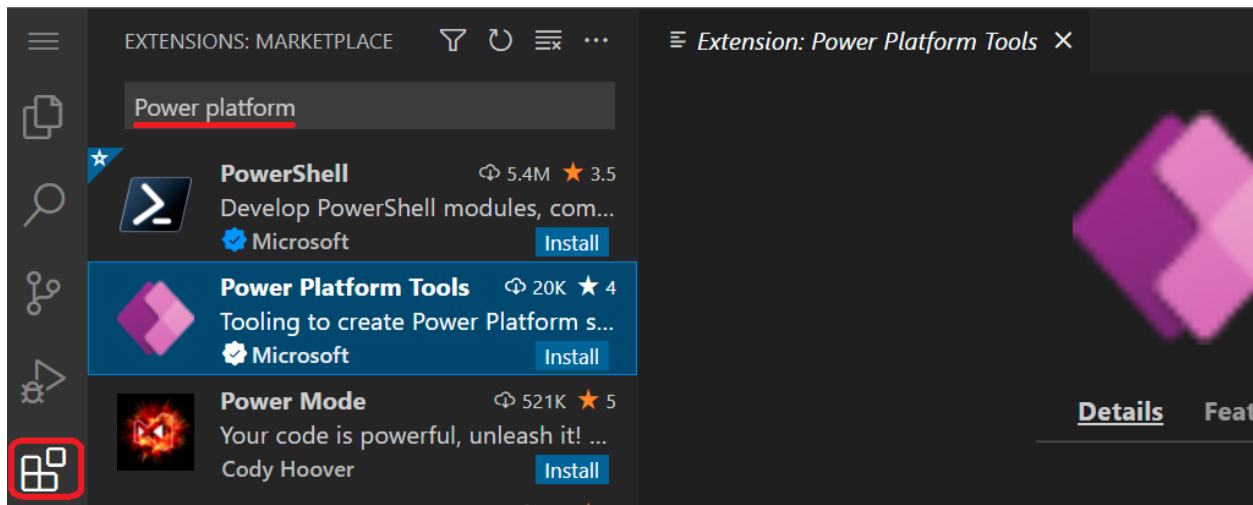
The installation of VS code extension can happen in 3 ways:

- 1) Install from within the VS Code Extension
- 2) Install from VS Code Market Place
- 3) Sideload the VSIX

INSTALLATION FROM WITHIN VS CODE

Launch Visual Studio Code or Run GitHub CodeSpace (if you have access to Codespaces)

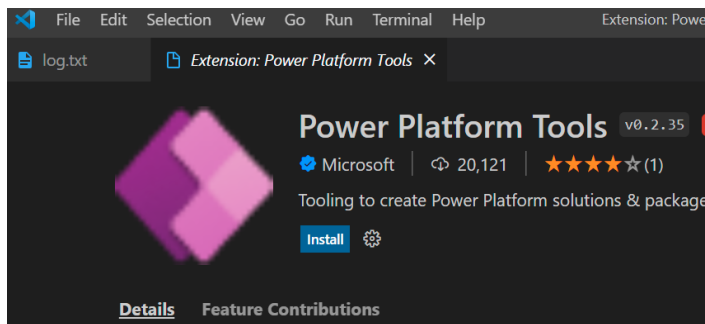
Then click on the extension icon on the Activity bar and search for Power Platform on the Extensions side bar.



If you are installing in GitHub CodeSpace, you will see the following option for installation



or you will see the following in your native VS Code instance

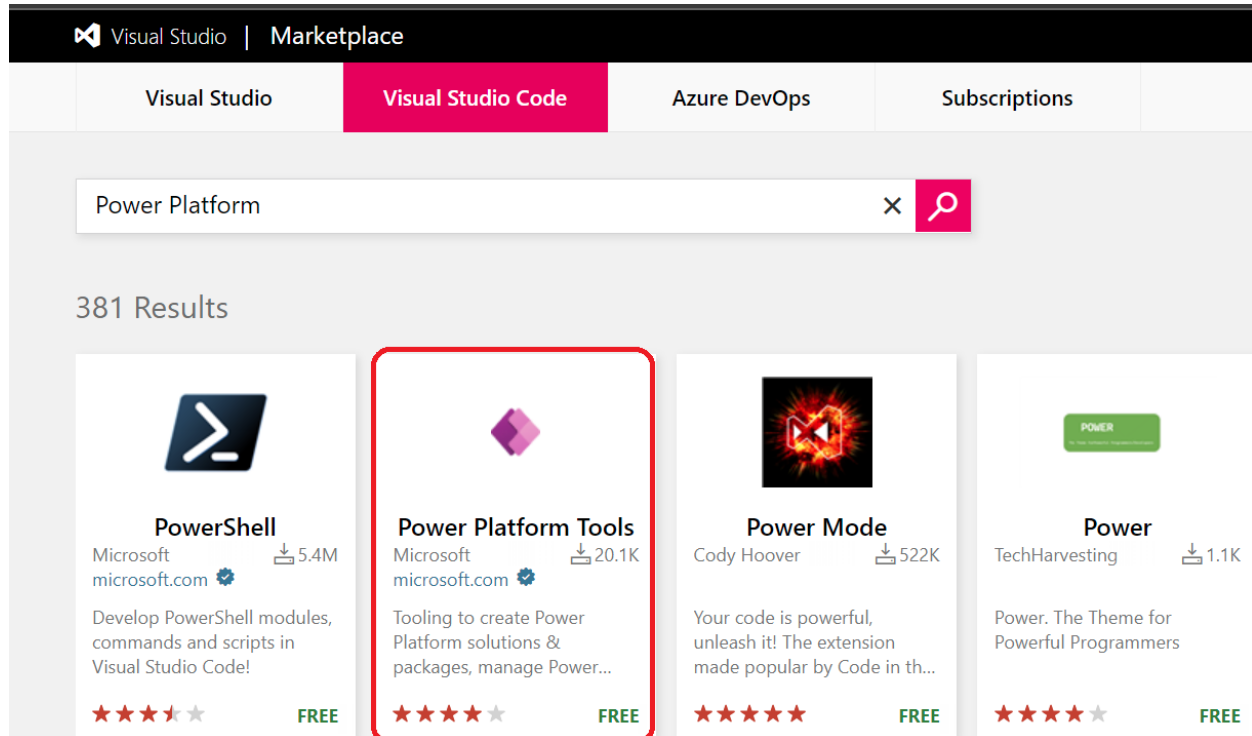


And the installation should be completed.

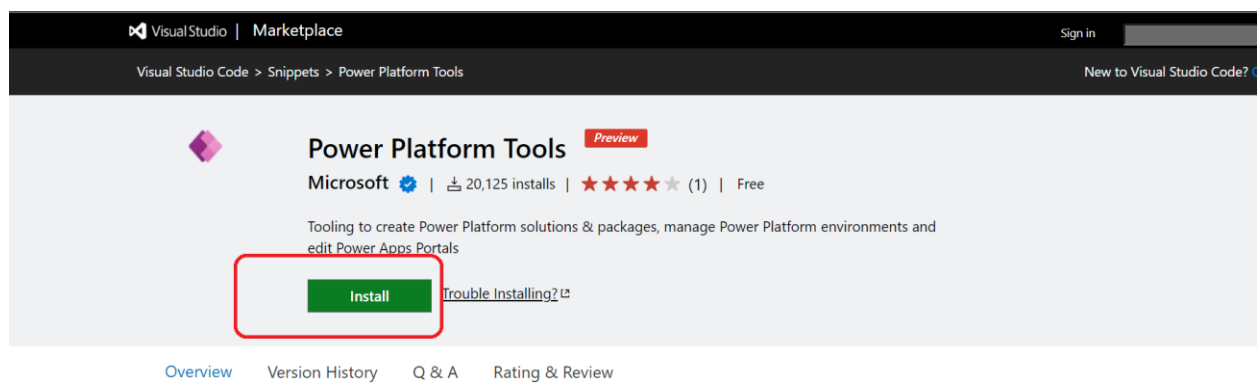
INSTALLING FROM VISUAL STUDIO MARKETPLACE

Go to the following URL: <https://marketplace.visualstudio.com/>

Select Visual Studio code and Search for Power Platform and Select the extension

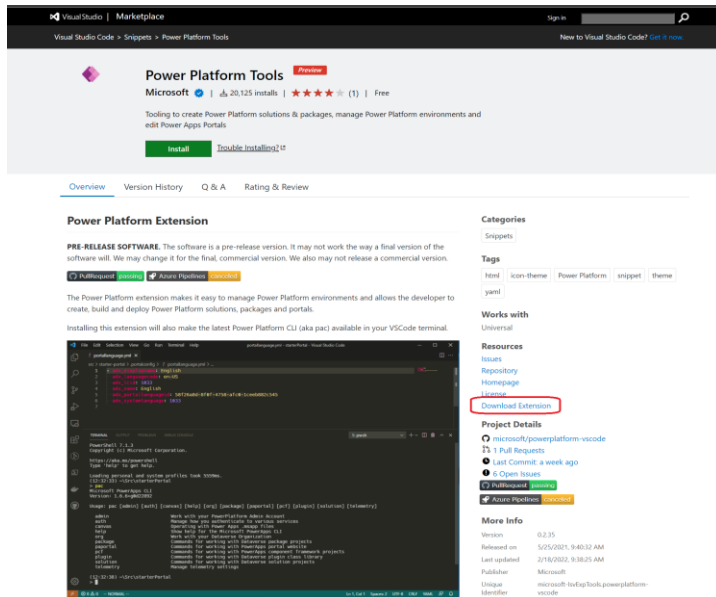


This will take you to the extensions page and from there you can launch your installation in your local instance of VS Code

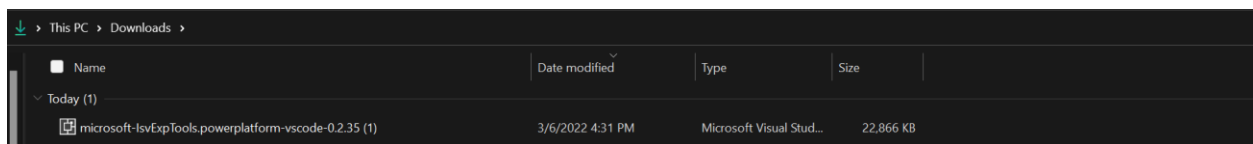


SIDE LOADING AND INSTALLING THE EXTENSION

In some organizations, running an installation process directly off the web is prohibited. In most cases the organization downloads the installation in a secure location, tests it to make sure it does not violate company policy, and then makes it available to the rest of the organization. To do this, from the marketplace site for the extension, as mentioned in the prior step, instead of pressing the install button, you will select the Download extension link from the page.

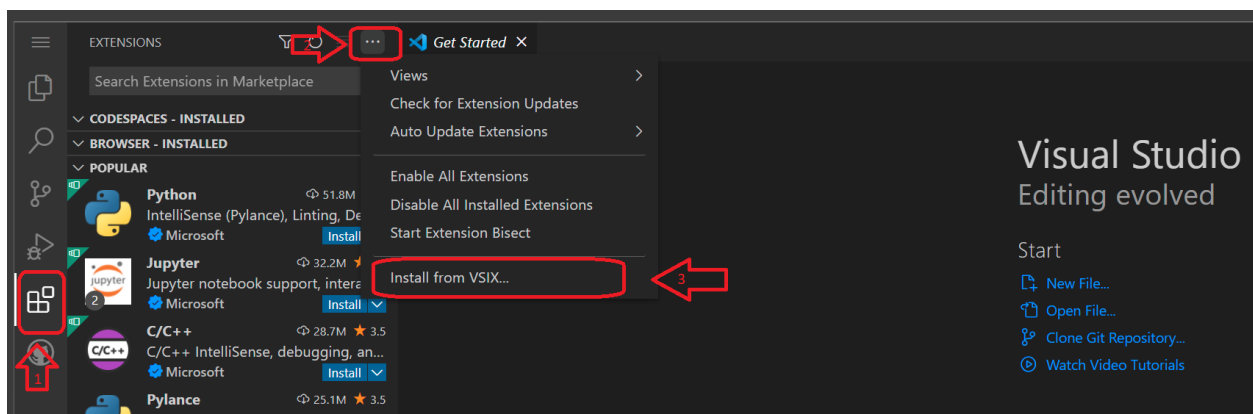


This will download a VSIX file, on your workstation, on PC it downloads to the Download folder.

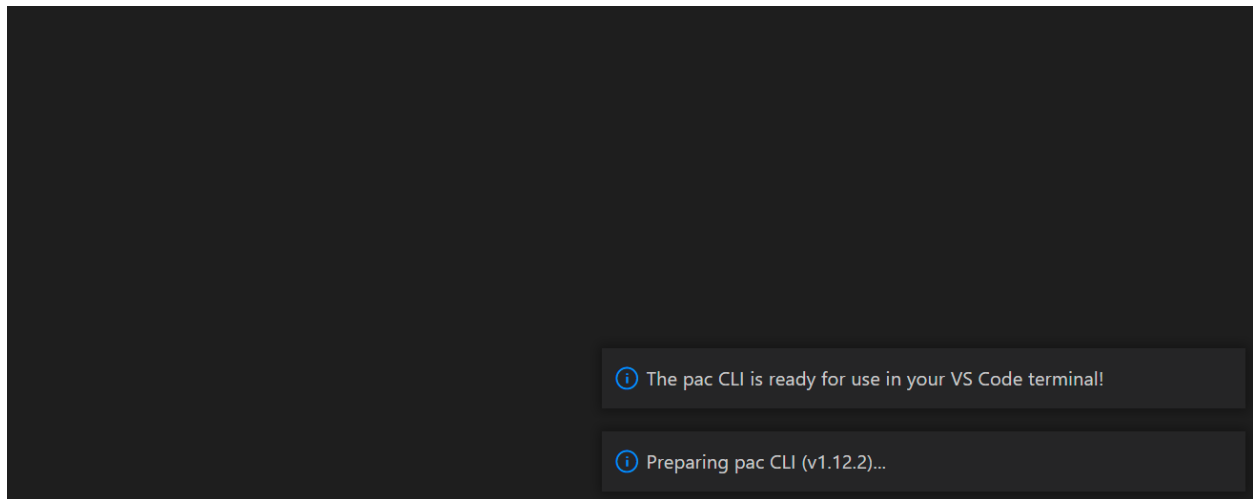


Once that is downloaded, launch Visual Studio Code on your workstation

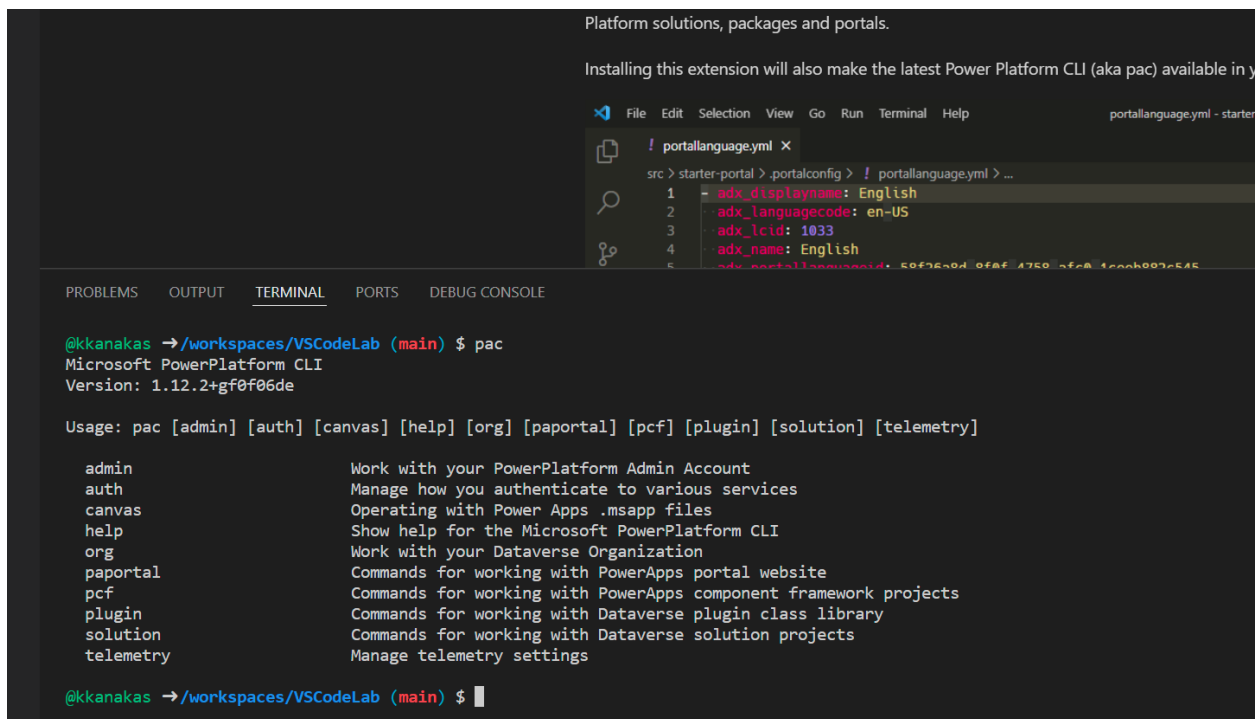
1. Select the Extensions icon from the Activity bar
2. Then click on the ellipsis icon on the Extensions side bar
3. Select install from VSIX



This will now install the Power Platform extension in your Visual Studio Code or GitHub Extension. On the bottom right-hand corner, you will see the following message.



Once the installation is done, launch the terminal interface from the Visual Studio Code interface



and you will see the following icon show-up on the activity panel



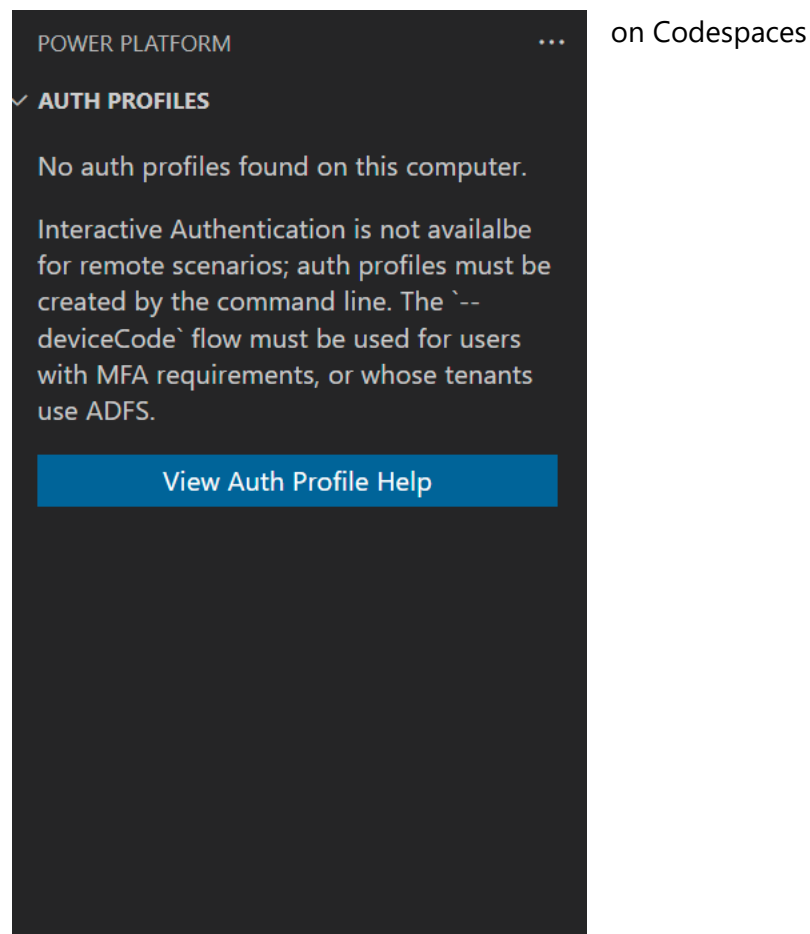
AUTHENTICATING TO POWER PLATFORM AND LISTING ENVIRONMENTS AND SOLUTIONS

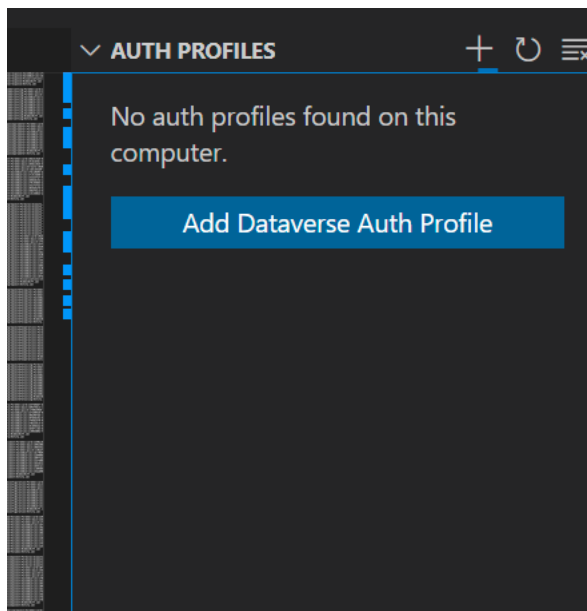
When you click on the Power platform icon on the activity bar, within the side bar, you will 2 panels, they are called

- Authentication Profiles Panel
- Environments and Solution Panel

The Authentication panel will only list the dataverse environments to which you are connected and based on that connection, it will populate the environment and solutions to which you have access to.

If you don't have any authentication profiles listed, then you will see the following:





on your workstation, if you click on the add Dataverse Auth Profile. It will launch the terminal window and show how to create the authentication profile using Power Platform CLI. The command in this case **"pac auth create"**

pac auth create allows to create two kinds of connection profiles

- 1) Administrator – used to create, list, backup, and remove environments
- 2) Dataverse – Authenticated to the relevant environment to execute solution operations.

The Visual studio code extension, we focus on the latter.

```
(c) Microsoft Corporation. All rights reserved.

C:\Users\kartikka>pac auth help
Microsoft PowerPlatform CLI
Version: 1.12.2+gf0f06de

Help:
Manage how you authenticate to various services

Commands:
Usage: pac auth [create] [list] [select] [delete] [name] [clear]

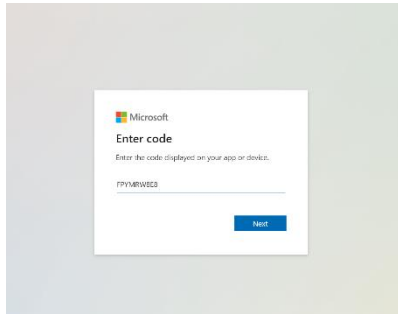
create      Create and store authentication profiles on this computer
list        List the authentication profiles stored on this computer
select      Select which authentication profile should be active
delete      Delete a particular authentication profile
name        Name or rename an existing authentication profile
clear       Clear all authentication profiles stored on this computer
```

The command to create an authentication profile is **"pac auth create"** this command by default creates Dataverse authentication profile, unless you specify **"admin"** as part of the **--kind or -k** parameter.

You can specify username and password for the connection with the **--username** or **--password**, but these are optional. If you don't specify these parameters, the AAD authentication window pops up. You can also authenticate using Service Principal credentials, as in using your client id and secret. You also have the option to use the Device Code authentication flow, if you don't want to show your username and password, when you are doing this lab. In this example we are using the device code authentication

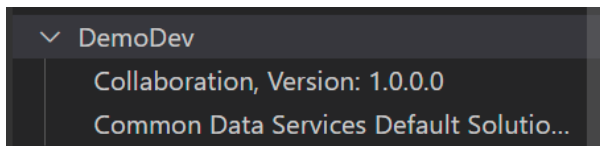
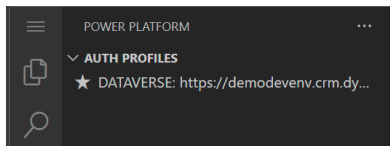
```
C:\Users\kartikka>pac auth create --url https://demodevenv.crm.dynamics.com -dc
To sign in, use a web browser to open the page https://microsoft.com/devicelogin and enter the code FPYMRW8E8 to authenticate.
```

launch a browser session on your workstation, and go to the URL <https://microsoft.com/devicelogin> and provide the device code.



you then proceed to provide the name of the account you want to authenticate with Azure AD credentials and then auth profile token gets updated and the Auth profile side bar shows the authenticated profile with a star against it.

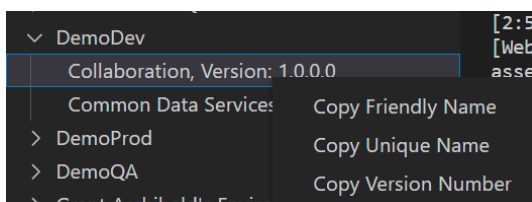
```
@kkanakas →/workspaces/VSCodeLab (main) $ pac auth create --url https://demodev.crm.dynamics.com -dc
To sign in, use a web browser to open the page https://microsoft.com/devicelogin and enter the code RDPX3RHBZ to authenticate.
Authentication successfully created.
Validating connection...
Connected to...DemoDev
* DATAVERSE https://demodev.crm.dynamics.com/ : user [redacted] Public
@kkanakas →/workspaces/VSCodeLab (main) $
```



In the environments and solution section you will see the solutions that are available in the environment to which you are authenticated too.

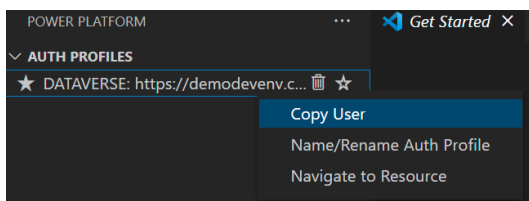
Now you have authenticated Power Platform your Visual Studio Code interface.

When you right-click on the solution you have the following options to select from:



- Copying the friendly name of the solution
- Copying the Unique Name(this is the important one, for command operations)
- Copy the version numbers of the solution

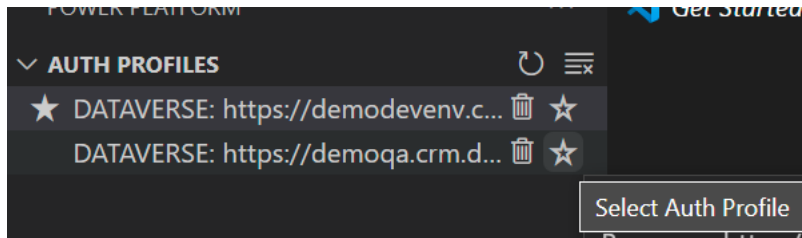
and you right-click on your auth profile you get the following options:



- Copying username used to authenticate to Power Platform
- Name/Rename the Auth profile to a friendly one
- Open a new tab and navigate to the environment url

Now, if you want to delete the profile, you can press the trash icon, and that will delete your authentication profile.

If you have multiple auth profiles, then you can select the star icon against the profile that you want to change to, and that will be the profile that will be in effect.

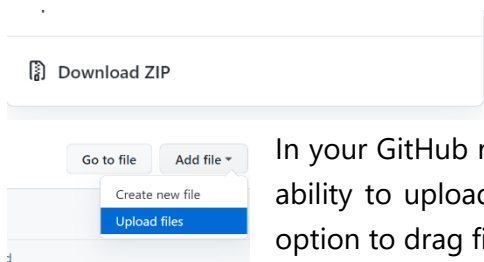


This makes it easier to navigate across different auth profiles. To create a second auth profile, just repeat the same steps that you did to create the first one.

WORKING WITH THE POWER PLATFORM TOOLS EXTENSION FOR VISUAL STUDIO CODE

Now that we have installed the extension, authenticated to the environment, we are going to work with solutions using the Visual Studio Code user interface.

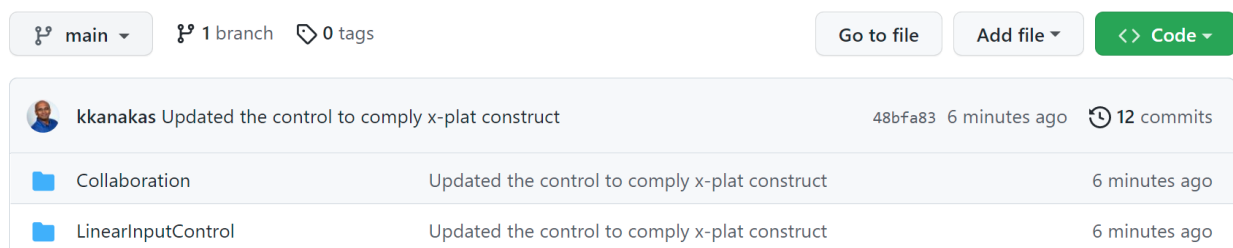
Download the zip files that you need for this project from the [GitHub repository](#).



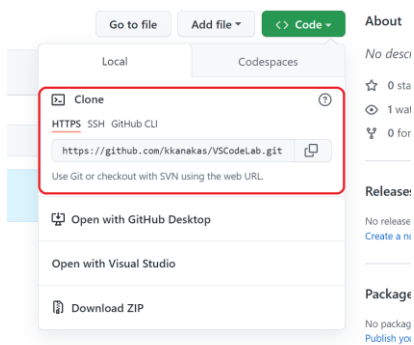
This zip file contains a solution and a PCF control that we are going to use for this project.

In your GitHub repository upload the extracted contents of the zip file. The ability to upload files is under the add files button. This will give you the option to drag files from your workstation and commit them to GitHub.

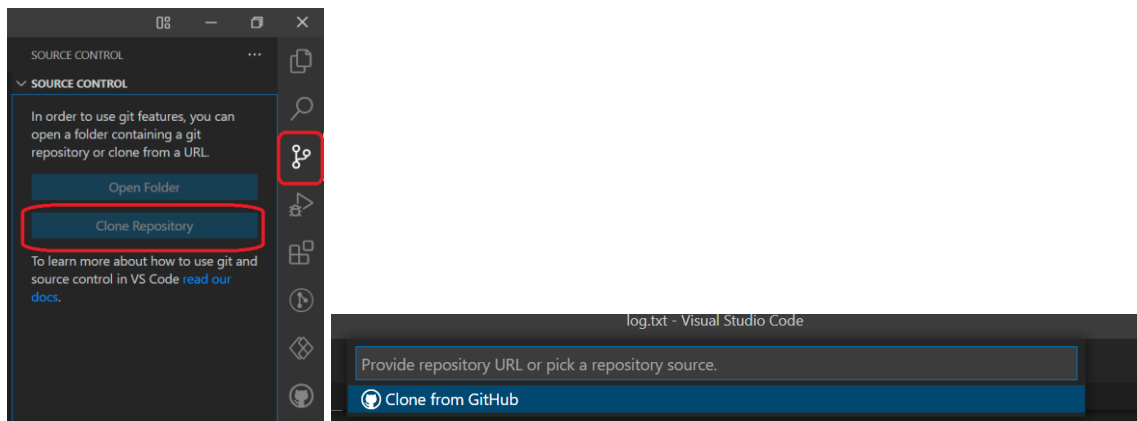
In your personal GitHub Repository should look like the following:



If you are using Visual Studio Code locally on your workstation, then proceed to clone your repository on to your workstation. Under the code button select clone and copy the link to your repository



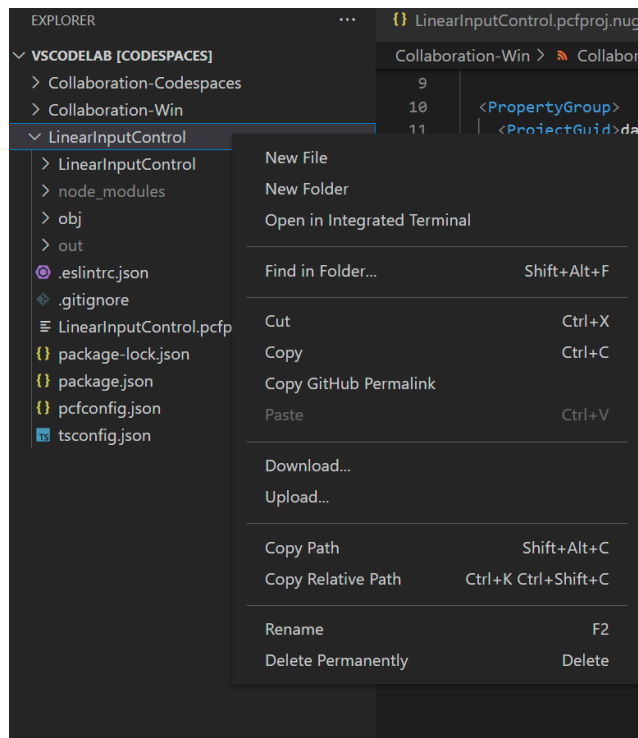
Then on Visual Studio code, select git from the activity bar and then press the clone repository button in the side bar, then paste the URL in the command palette. If you have not authenticated to GitHub, then Visual Studio Code will prompt for you for your GitHub credentials and after providing that you will be able to clone your repository



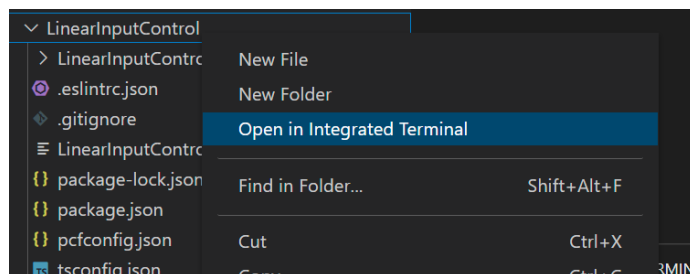
Now that your repository has been successfully cloned. Let us start working with the solutions and controls. **Note: if you are using GitHub codespaces you do not have to authenticate to the GitHub as you are already authenticated. All you must do is launch CodeSpaces from the code button.**

WORKING WITH CONTROLS AND SOLUTIONS

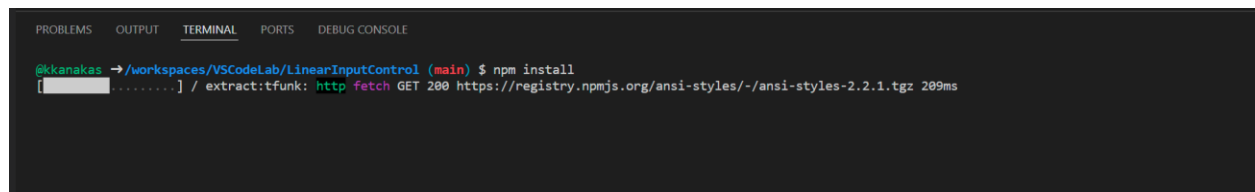
Now that we are in our Visual Studio code environment, Select the explorer icon from the activity bar, and your side bar should look like the following, with the folders completely expanded:



then right click on the linear control folder and select open in integrated terminal



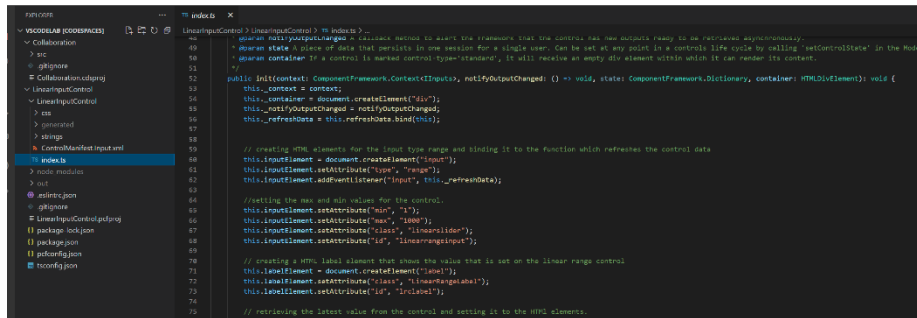
In the terminal window, type the following:



This will install the appropriate node libraries needed for your PCF control. If there are things that need to be fixed, please run **"npm audit fix"** to fix any vulnerabilities.

Once you have done those operations, now let us look at the control, and then proceed to build it.

Select the explorer icon and expand the LinearInputControl folder which contained within the LinearInputControl folder



The screenshot shows the VS Code interface. On the left, the Explorer view is open, showing the project structure. The 'LinearInputControl' folder is expanded, and the 'index.ts' file is selected. On the right, the Source Control view shows the 'index.ts' file with the following code:

```
// index.ts
import { ComponentFrameworkContext, ComponentFrameworkContextInputs, ComponentFrameworkContextOutputs, ComponentFrameworkContextOutputs } from '@microsoft/component-framework';
import { ComponentFrameworkContextInputs } from '@microsoft/component-framework';
import { ComponentFrameworkContextOutputs } from '@microsoft/component-framework';

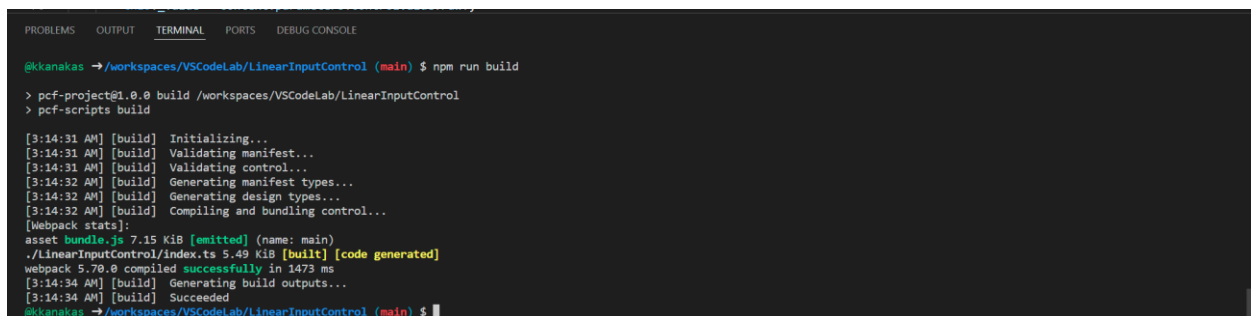
// Context Inputs
// Context Outputs
// Context Inputs
// Context Outputs

// Context Inputs
// Context Outputs
// Context Inputs
// Context Outputs

// Context Inputs
// Context Outputs
// Context Inputs
// Context Outputs
```

Select the index.ts file and you can see that this is a slider control. Now when we proceed to build it, it will take these typescript files and create the appropriate javascript

for the web control. So now, in the terminal window type **npm run build** and you should see the following:



The screenshot shows the terminal window with the following output:

```
@kkanakas → /workspaces/VSCoDeLab/LinearInputControl (main) $ npm run build
> pcfr-project@1.0.0 build /workspaces/VSCoDeLab/LinearInputControl
> pcfr-scripts build

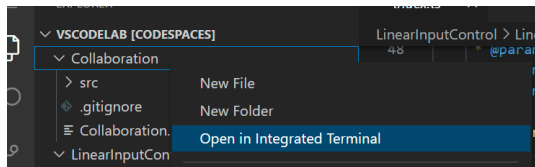
[3:14:31 AM] [build] Initializing...
[3:14:31 AM] [build] Validating manifest...
[3:14:31 AM] [build] Validating control...
[3:14:32 AM] [build] Generating manifest types...
[3:14:32 AM] [build] Generating design types...
[3:14:32 AM] [build] Compiling and bundling control...
[Webpack stats]:
asset bundle.js 7.15 KiB [emitted] (name: main)
./LinearInputControl/index.ts 5.49 KiB [built] [code generated]
webpack 5.70.0 compiled successfully in 1473 ms
[3:14:34 AM] [build] Generating build outputs...
[3:14:34 AM] [build] Succeeded
@kkanakas → /workspaces/VSCoDeLab/LinearInputControl (main) $
```

And then we are going to test it, by using the command **npm run start**. This launch this web control in a browser under component framework testing environment. You slide the control back and forth and see the values change in the Data Input section.



Now, we know our control works properly, let go ahead add it to our solution.

On the Explorer window, select the collaboration folder and right-click and select open in integrated terminal



In the integrated terminal, type the “**pac solution add-reference --path <path to the pcf control>**”

```
@kkanakas → /workspaces/VSCoDeLab/Collaboration (main X) $ pac solution add-reference --path ../LinearInputControl
Project reference successfully added to Dataverse solution project.
```

this will add the pcf control to the solution

Now proceed to build the solution by typing “**dotnet build**”, this will start building the zip file needed to import into your Power Platform environment.

```
PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE

@kkanakas → /workspaces/VSCoDeLab/Collaboration (main X) $ pac solution add-reference --path ../LinearInputControl
Project reference successfully added to Dataverse solution project.
@kkanakas → /workspaces/VSCoDeLab/Collaboration (main X) $ dotnet build
Microsoft (R) Build Engine version 17.0.0-0cdeb0dd4 for .NET
Copyright (C) Microsoft Corporation. All rights reserved.

Determining projects to restore...
  Restored /workspaces/VSCoDeLab/Collaboration/Collaboration.cdsproj (in 533 ms).
  Restored /workspaces/VSCoDeLab/LinearInputControl/LinearInputControl.pcfproj (in 533 ms).

> pcf-project@1.0.0 build
> pcf-scripts build "--noColor" "--buildMode" "development" "--outDir" "/workspaces/VSCoDeLab/LinearInputControl/out/controls/" "--buildSource" "MSBuild"

[11:41:09 PM] [build] Initializing...
[11:41:09 PM] [build] Validating manifest...
[11:41:09 PM] [build] Validating control...
[11:41:10 PM] [build] Generating manifest types...
[11:41:10 PM] [build] Generating design types...
[11:41:10 PM] [build] Compiling and bundling control...
[webpack stats]:
asset bundle.js 7.15 KiB [emitted] (name: main)
./LinearInputControl/index.ts 5.49 KiB [build] [code generated]
webpack 5.70.0 compiled successfully in 1615 ms
[11:41:12 PM] [build] Generating build outputs...
[11:41:12 PM] [build] Succeeded
Running Solution Packager to build package type: Unmanaged bin/Debug/Collaboration.zip

Packing /workspaces/VSCoDeLab/Collaboration/obj/Debug/Metadata to bin/Debug/Collaboration.zip

Processing Component: Entities
Processing Component: Roles
Processing Component: Workflows
Processing Component: FieldSecurityProfiles
Processing Component: Templates
Processing Component: EntityMaps
Processing Component: EntityRelationships
Processing Component: OrganizationSettings
Processing Component: OptionSets
Processing Component: CustomControls
  - SampleNamespace.LinearInputControl
Processing Component: EntityDataProviders
Processing Component: CanvasApps
  - collab_connectionsapp_06f79
Processing Component: Connectors
  - collab_assemblyresult
Following root components are not defined in customizations:
Type='Connector', Id (or schema name)='237bb2c4-f777-47d3-ad5e-4844e434c0d3'.

Unmanaged Pack complete.

Solution: bin/Debug/Collaboration.zip generated.
Solution Package Type: Unmanaged generated.
Completed intermediate files clean up.

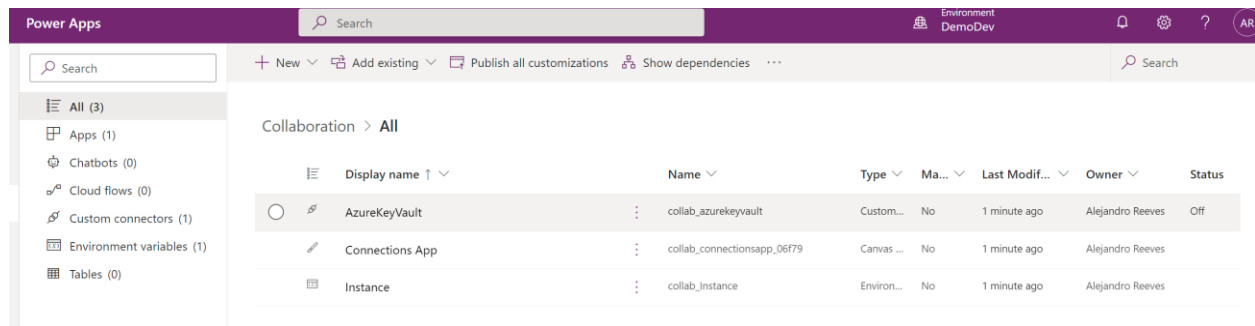
Build succeeded.
0 Warning(s)
0 Error(s)

Time Elapsed 00:00:07.43
```

Once the project is built navigate to the bin\Debug folder under the Collaboration folder and run the solution import command back into our dev environment.

```
@kkanakas → .../VSCoDeLab/Collaboration-Codespaces/bin/Debug (main X) $ pac solution import --path ./Collaboration.zip
Connected to...DemoDev
Solution Importing...
Solution Imported successfully. Import id: 270c2c33-1c3c-43a8-8b0e-048d36c9179f
@kkanakas → .../VSCoDeLab/Collaboration-Codespaces/bin/Debug (main X) $
```

And you confirm if your solution updated in your environment using the Power Platform Web UI



The screenshot shows the Power Apps web interface. The top navigation bar includes the 'Power Apps' logo, a search bar, and the environment name 'DemoDev'. The left sidebar contains a search bar and a list of categories: 'All (3)', 'Apps (1)', 'Chatbots (0)', 'Cloud flows (0)', 'Custom connectors (1)', 'Environment variables (1)', and 'Tables (0)'. The main content area is titled 'Collaboration > All' and displays a table of components.

	Display name ↑	Name	Type	Ma...	Last Modif...	Owner	Status
	AzureKeyVault	collab_azurekeyvault	Custom...	No	1 minute ago	Alejandro Reeves	Off
	Connections App	collab_connectionsapp_06f79	Canvas ...	No	1 minute ago	Alejandro Reeves	
	Instance	collab_instance	Environ...	No	1 minute ago	Alejandro Reeves	

You have now successfully used the Power Platform Visual Studio Code Extension.