

# Skill up with Power Platform and GitHub

---

Lab Setup

TenantId:  <TenantID>

ApplicationId:  <ApplicationID>

Client Secret:  <Secret>

SPN:  <ServicePrincipalName>

EnvironmentId:

**NOTE:** Please make sure **Lab Setup** button has generated values prior to pressing **Create App User** button.

Create App User

## Lab Outline

---

- ☐ We change some source code .
- ☐ Push that source code control into an environment .
- ☐ Build a low code app and use the code-based control.
- ☐ Run a workflow in GitHub to promote the changes from the developer environment to your test environment and understand the difference between managed and unmanaged solutions.

## Items pre-provisioned

- Developer environment
- QA environment
- App registration in Azure AD and we provide the APP ID and Client ID and the SECRET (this is required for the workflow)

⚠ As a prerequisite, you will need a GitHub Account.

We will need to provide the ability to clone GitHub repository with all modifications for the PCF control.

### What is expected out of this lab?

The expectation of this lab is as follows:

Understand the few of the code first capabilities available in Power Platform and how can code first developers work with Power Platform using their existing tools. We will go through a set of tools that are familiar to developers and a few concepts that are unfamiliar to developers. We will also experience how to develop and deploy low code applications using a continuous integration/continuous delivery paradigm. The intent of this lab is to do everything over the web and not have install any components on the local workstation, but you will have step by step instructions on how to proceed and become productive with your existing developer skills with low code platforms.

We will learn how to deploy low code applications with code first components and how to stage them from Development, to QA to Production.

We will:

- ☐ Use codespace and build a PCF control.
- ☐ Push it into a DEV environment.
- ☐ Add the PCF control to an app.
- ☐ Export the new app with PCF control to GitHub.
- ☐ Have a GitHub workflow to deploy from Dev to Test.

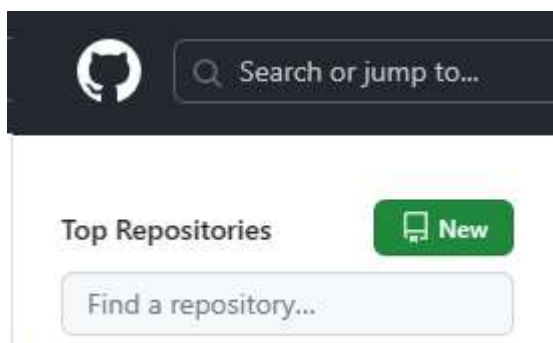
---

## Use GitHub Repository, Codespaces, and build a PCF control

We begin with GitHub once connected to your Lab machine. We will experience how to make use of GitHub codespace to create and deploy code component using Microsoft Power Platform CLI.

1. Please **login to your GitHub account**.
2. Let's start to create a clone of GitHub repository that consists of needed artifacts.

Click **New** for repository and on the next screen find a hyperlink **Importing a repository** to click on.



## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?

[Import a repository.](#)

**GitHub Repo to use:** <https://github.com/snizar007/BuildLab2023.git>

Create a repository of your choice and make a note of the repository name that will be used later. Click **Begin import**.

## Import your project to GitHub

Import all the files, including revision history, from another version control system.

Support for importing Mercurial, Subversion and Team Foundation Version Control (TFVC) repositories will end on October 17, 2023. For more details, see the [changelog](#).

Your old repository's clone URL

<https://github.com/snizar007/BuildLab2023.git>

Learn more about the types of [supported VCS](#).

### Your new repository details

Owner \*

 snizar00

Repository name \*

Build2023

✓ Build2023 is available.

- ☒  **Public**  
Anyone on the internet can see this repository. You choose who can commit.
- ☐  **Private**  
You choose who can see and commit to this repository.

 You are creating a public repository in your personal account.

[Cancel](#)

[Begin import](#)

## Preparing your new repository

There is no need to keep this window open, we'll email you when the import is done.

 snizar00/Build2023. Cancel



 Optimizing repository and pushing commits to GitHub...

## Preparing your new repository




There is no need to keep this window open, we'll email you when the import is done.


 snizar00/Build2023.



✓ Importing complete! Your new repository snizar00/Build2023. is ready.





3. Once your repository has been imported, you should see it in your list of repositories. Click on your repository name to view imported artifacts (*like below*).

 main  1 branch  0 tags Go to file Add file <> Code

 snizar007

Delete LinearInputControl/solutions/LinearInputControlSolution directory

f8a8b1c 24 minutes ago 65 commits

 .github/workflows	Delete Release-SolutionToTest.yml	11 hours ago
 LinearInputControl/controls	Delete LinearInputControl/solutions/LinearInputControlSolution directory	24 minutes ago
 out	Updates from the dev environment	12 hours ago
 README.md	Initial commit	yesterday

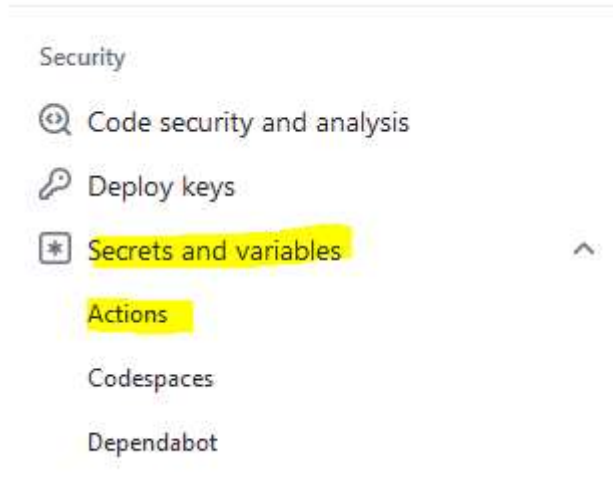
README.md

# BuildLab2023

4. Now let's setup GitHub secret that is provided as part of the lab using steps below:

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

5. Click on **Secrets and variables** > **Actions**.



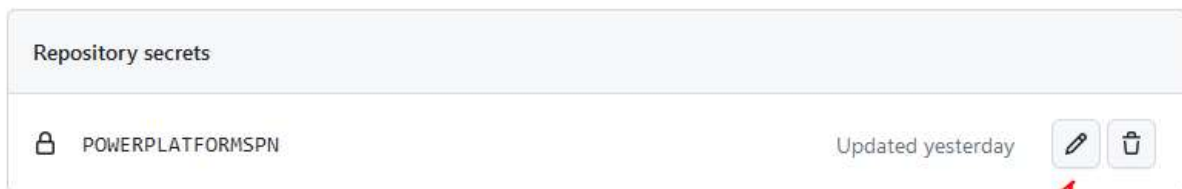
6. Click the **New repository secret** button.

Name the secret **PowerPlatformSPN** under **Repository secrets**. Type the **Client Secret Value** provided in the **Lab** > in the **Secret** area.

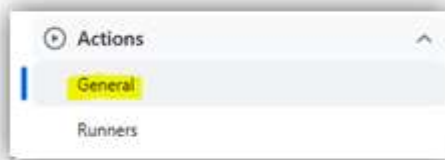
After updating the value, click **Add secret**.

A screenshot of the 'New repository secret' form. The 'Name' field is highlighted with a yellow background and contains the text 'PowerPlatformSPN'. The 'Secret' field is empty and has a blue border. A green button labeled 'New repository secret' is at the top.

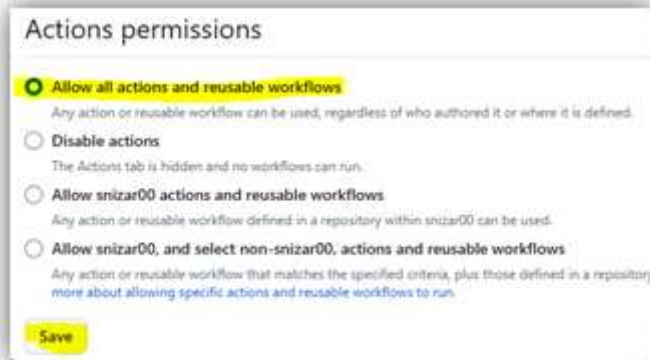
**NOTE:** If you need to update the secret value, click **edit** (below) and type the Secret value and click **Update secret**.



7. Please make sure that you have proper Actions Permissions. On **Settings** page, click **Actions** > **General**.

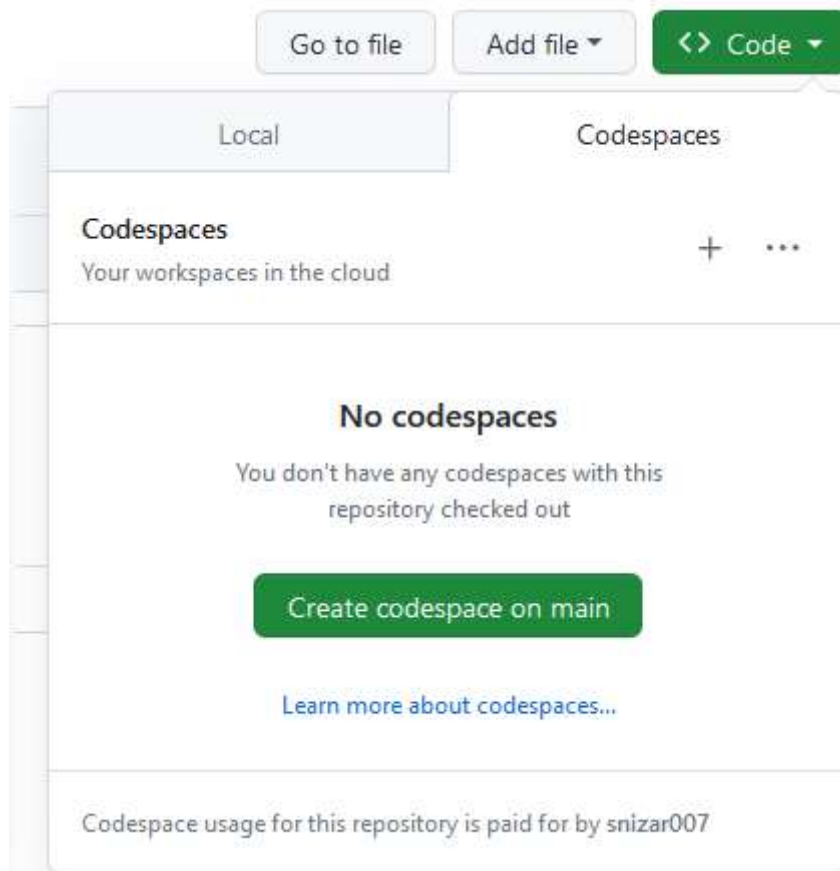


Make sure "Allow all actions and reusable workflows" is checked. If not, check it and click **Save**.

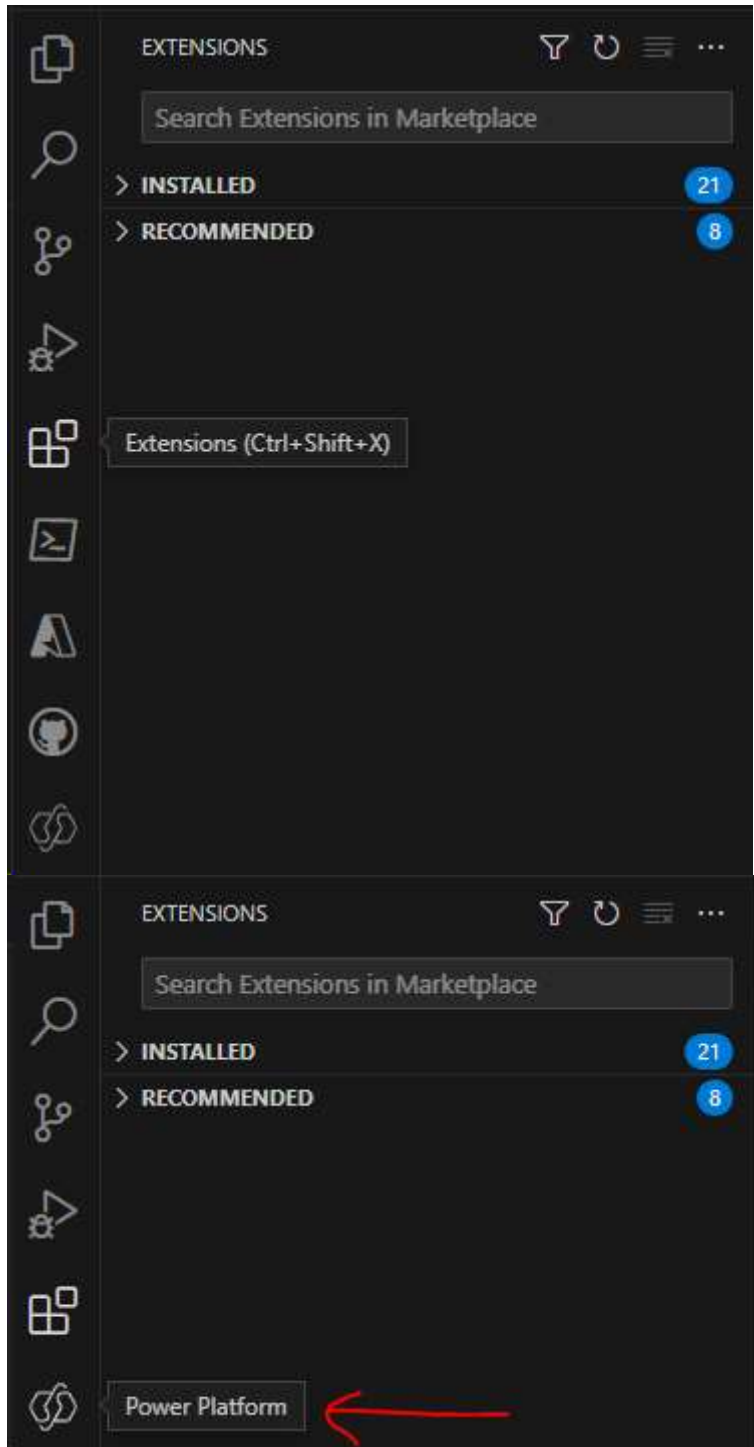


8. Next, you will create a GitHub codespace in the same REPO.

On Code menu option, click on **<> Code > Create codespace on main** under **Codespaces** tab.

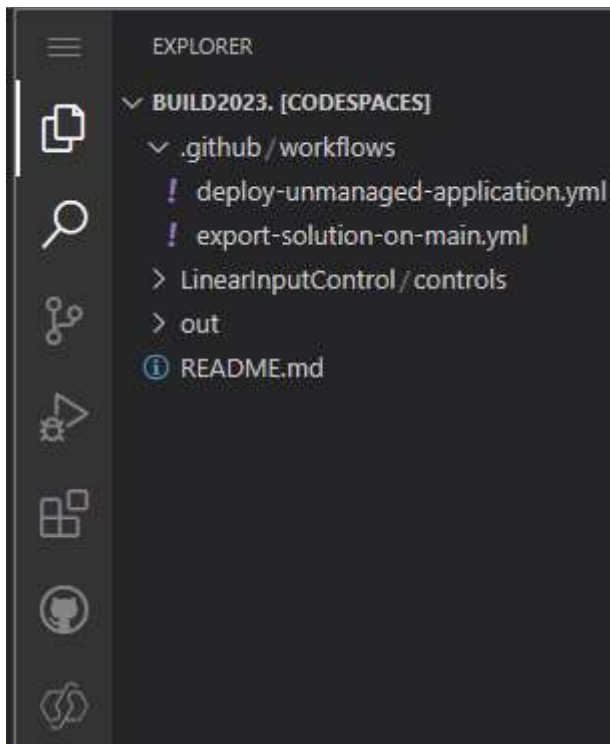


9. Once your codespace starts, let's install **Power Platform Tools** Extension. Click **Extensions >** search **Power Platform Tools >** click **Install**.



You shall see a cloned repo and Power Platform extension.





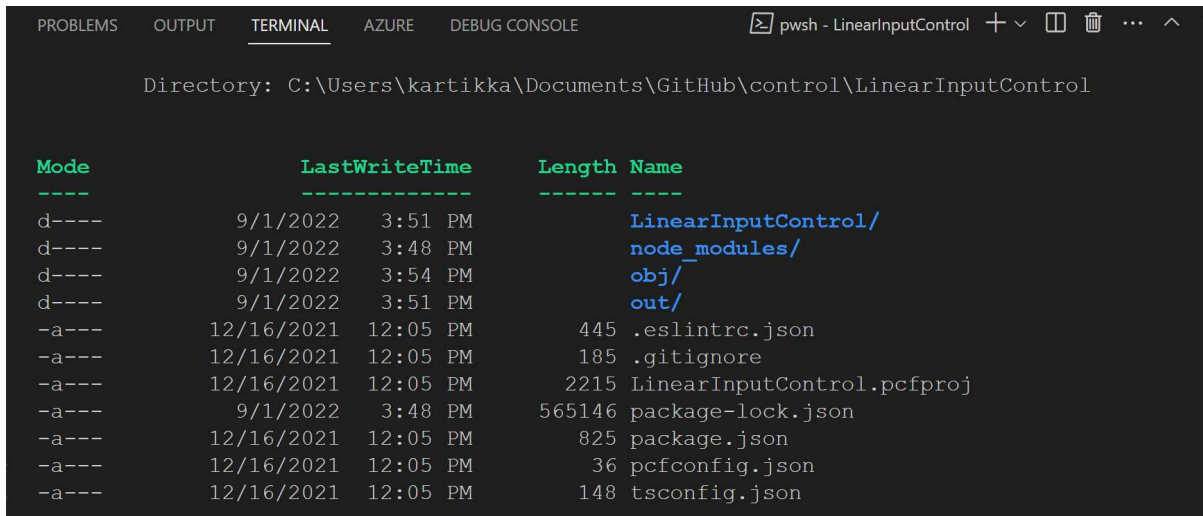
In your TERMINAL section of the screen let's review. We have a Code component folder (*LinearInputControl*) and a **README.md** file. Below image shows folders (*LinearInputControl*, and *controls* folder within *LinearInputControl* folder) you should see in your TERMINAL section.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  COMMENTS

• @snizar00 →/workspaces/BuildTest (main) $ ls -l
total 8
drwxrwxrwx+ 3 codespace root 4096 May 22 20:26 LinearInputControl
-rw-rw-rw- 1 codespace root 14 May 22 20:26 README.md
• @snizar00 →/workspaces/BuildTest (main) $ cd LinearInputControl/
○ @snizar00 →/workspaces/BuildTest/LinearInputControl (main) $
○ @snizar00 →/workspaces/BuildTest/LinearInputControl (main) $
• @snizar00 →/workspaces/BuildTest/LinearInputControl (main) $ ls -l
total 4
drwxrwxrwx+ 3 codespace root 4096 May 22 20:26 controls
• @snizar00 →/workspaces/BuildTest/LinearInputControl (main) $ cd controls/
○ @snizar00 →/workspaces/BuildTest/LinearInputControl/controls (main) $
• @snizar00 →/workspaces/BuildTest/LinearInputControl/controls (main) $ ls -l
total 340
drwxrwxrwx+ 4 codespace root 4096 May 22 20:26 LinearInputControl
-rw-rw-rw- 1 codespace root 2284 May 22 20:26 controls.pcfproj
-rw-rw-rw- 1 codespace root 325495 May 22 20:26 package-lock.json
-rw-rw-rw- 1 codespace root 947 May 22 20:26 package.json
-rw-rw-rw- 1 codespace root 34 May 22 20:26 pcfconfig.json
-rw-rw-rw- 1 codespace root 144 May 22 20:26 tsconfig.json
○ @snizar00 →/workspaces/BuildTest/LinearInputControl/controls (main) $
```

Use GitHub codespace and build a PCF control:



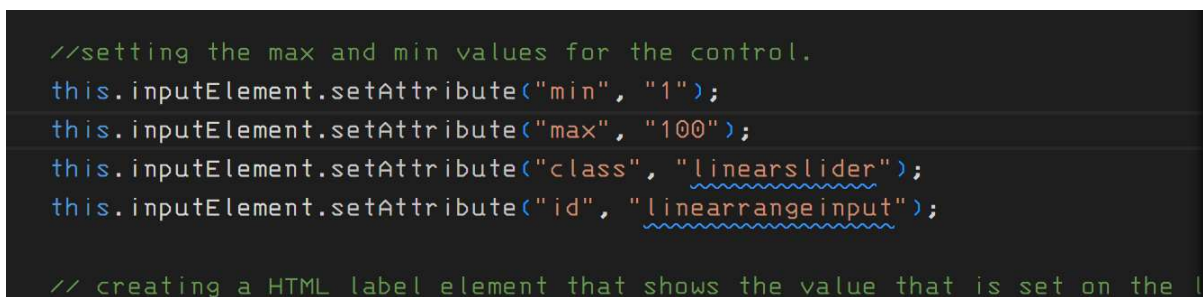


Terminal window showing the directory listing for the `LinearInputControl` folder. The directory is `C:\Users\kartikka\Documents\GitHub\control\LinearInputControl`. The listing shows the following files and folders:

Mode	LastWriteTime	Length	Name
d----	9/1/2022 3:51 PM		LinearInputControl/
d----	9/1/2022 3:48 PM		node_modules/
d----	9/1/2022 3:54 PM		obj/
d----	9/1/2022 3:51 PM		out/
-a---	12/16/2021 12:05 PM	445	.eslintrc.json
-a---	12/16/2021 12:05 PM	185	.gitignore
-a---	12/16/2021 12:05 PM	2215	LinearInputControl.pcfproj
-a---	9/1/2022 3:48 PM	565146	package-lock.json
-a---	12/16/2021 12:05 PM	825	package.json
-a---	12/16/2021 12:05 PM	36	pcfconfig.json
-a---	12/16/2021 12:05 PM	148	tsconfig.json

- Here we are going to go into the **LinearInputControl/LinearInputControl** folder (*in the left Explorer blade*) and click on the **index.ts** file. Please change the following value from a 1000 to a 100. This is to reduce the slider numbers that go upto 100.

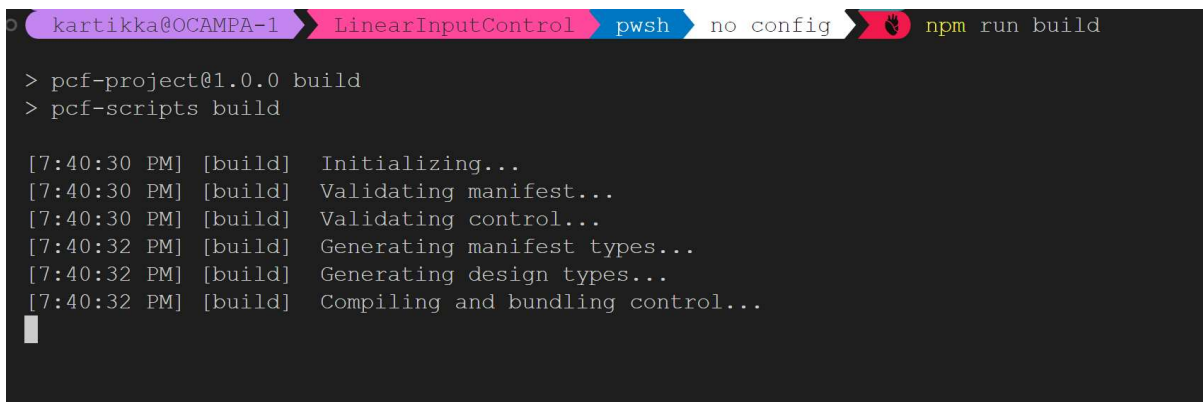
**NOTE:** Please make sure to **commit your GitHub changes**.



```
//setting the max and min values for the control.
this.inputElement.setAttribute("min", "1");
this.inputElement.setAttribute("max", "100");
this.inputElement.setAttribute("class", "linearslider");
this.inputElement.setAttribute("id", "linearrangeinput");

// creating a HTML label element that shows the value that is set on the l
```

- Now we **save**. Then we go to the terminal window and type **npm run build** (**NOTE:** we may need to run `npm install` first within `LinearInputControl/control` folder where you have `package.json` file)



Terminal window showing the build process. The prompt is `kartikka@OCAMPA-1`. The command `npm run build` is executed. The output shows the build process for `pcf-project@1.0.0` and `pcf-scripts`. The build is successful.

```
> pcf-project@1.0.0 build
> pcf-scripts build

[7:40:30 PM] [build] Initializing...
[7:40:30 PM] [build] Validating manifest...
[7:40:30 PM] [build] Validating control...
[7:40:32 PM] [build] Generating manifest types...
[7:40:32 PM] [build] Generating design types...
[7:40:32 PM] [build] Compiling and bundling control...
```

Once the build is successful you will see the following result with successful build message (below):

```

> pcf-project@1.0.0 build
> pcf-scripts build

[7:40:30 PM] [build] Initializing...
[7:40:30 PM] [build] Validating manifest...
[7:40:30 PM] [build] Validating control...
[7:40:32 PM] [build] Generating manifest types...
[7:40:32 PM] [build] Generating design types...
[7:40:32 PM] [build] Compiling and bundling control...
Browserslist: caniuse-lite is outdated. Please run:
  npx update-browserslist-db@latest
  Why you should do it regularly: https://github.com/browserslist/update-db#readme
[Webpack stats]:
asset bundle.js 7.21 KiB [emitted] (name: main)
./LinearInputControl/index.ts 5.52 KiB [built] [code generated]
webpack 5.74.0 compiled successfully in 17671 ms
[7:41:02 PM] [build] Generating build outputs...
[7:41:02 PM] [build] Succeeded

```

If the build does not run, this is so that the repository was cloned hence it does not have the dependencies to run the build. This is why we should run **npm install** (in *LinearInputControl/controls* folder).



```

@snizar00 →/workspaces/Build2023./LinearInputControl/controls (main) $ npm run build

> pcf-project@1.0.0 build
> pcf-scripts build

sh: 1: pcf-scripts: not found
@snizar00 →/workspaces/Build2023./LinearInputControl/controls (main) $ npm i -g

added 1 package in 135ms
@snizar00 →/workspaces/Build2023./LinearInputControl/controls (main) $ npm install

added 693 packages, and audited 694 packages in 13s

115 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
npm notice
npm notice New patch version of npm available! 9.6.3 -> 9.6.6
npm notice Changelog: https://github.com/npm/cli/releases/tag/v9.6.6
npm notice Run npm install -g npm@9.6.6 to update!
npm notice
@snizar00 →/workspaces/Build2023./LinearInputControl/controls (main) $

```

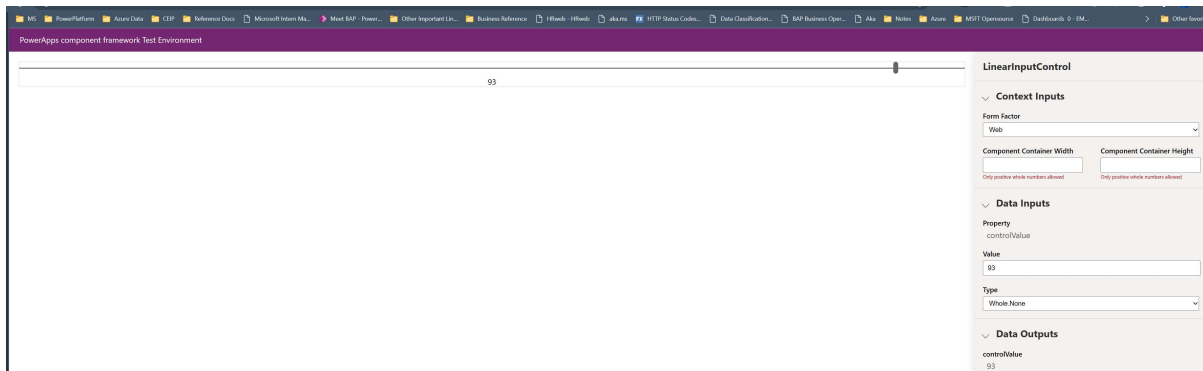
12. Now your build should succeed when you run **npm run build** as you can see a succeeded build below.

```
@snizar00 → /workspaces/Build2023./LinearInputControl/controls (main) $ npm run build

> pcf-project@1.0.0 build
> pcf-scripts build

[7:58:30 PM] [build] Initializing...
[7:58:30 PM] [build] Validating manifest...
[7:58:30 PM] [build] Validating control...
[7:58:32 PM] [build] Running ESLint...
[7:58:33 PM] [build] Generating manifest types...
DeprecationWarning: 'createInterfaceDeclaration' has been deprecated since v4.8.0. Decorators are no longer supported for this function. Callers should switch to an overload that does not accept a 'decorators' parameter.
[7:58:33 PM] [build] Generating design types...
[7:58:33 PM] [build] Compiling and bundling control...
[Webpack stats]:
asset bundle.js 7.12 KiB [emitted] (name: main)
./LinearInputControl/index.ts 5.48 KiB [built] [code generated]
webpack 5.82.0 compiled successfully in 2436 ms
[7:58:36 PM] [build] Generating build outputs...
[7:58:36 PM] [build] Succeeded
@snizar00 → /workspaces/Build2023./LinearInputControl/controls (main) $
```

13. Now let us test this component and type in **npm run start**. You will be prompted to open browser, go ahead and do that. (NOTE: *If your browser tab is taking time, click stop > refresh.*)



This will start the test harness and make sure that our changes are in. We changed slider value to 100.

Once confirmed slider, close slider browser TAB, come back to GitHub codespace and press Ctrl-C in the TERMINAL.

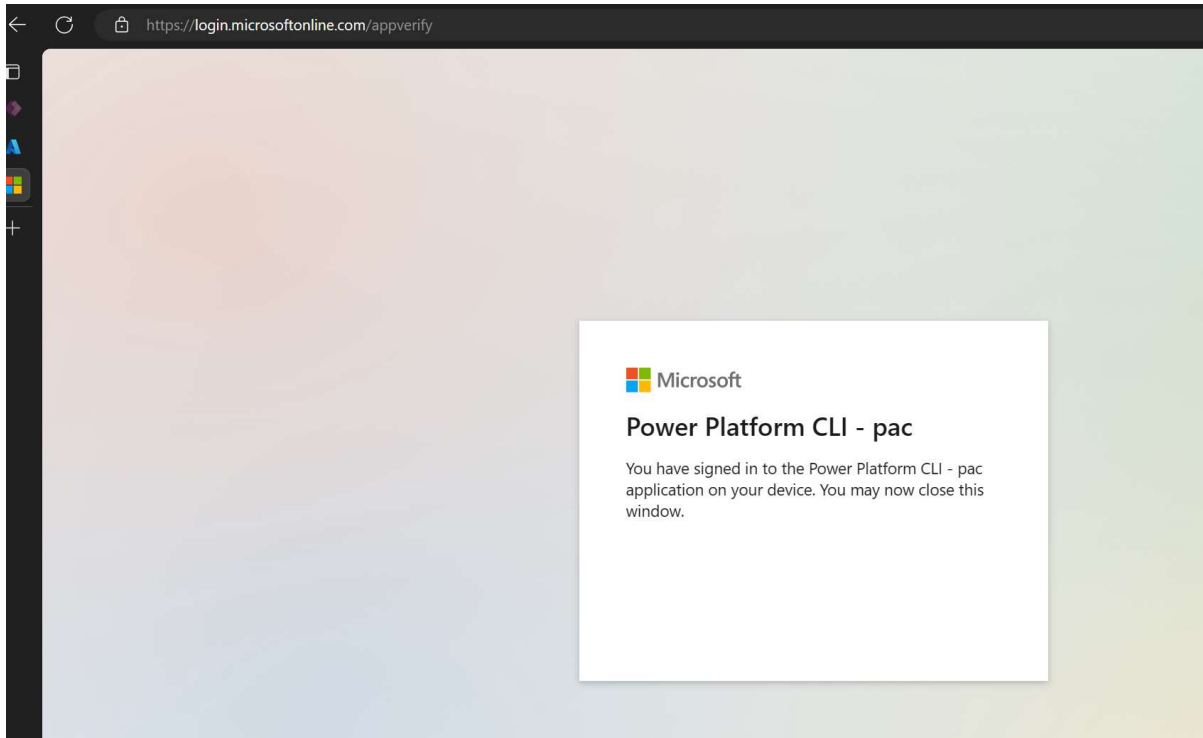
---

## Build PCF control and push it into a DEV environment

Now that we have built the control let us make it available to our Power platform environment.

1. First, we authenticate using a device code, this is the preferred method especially when using browser-based sessions. For this step, go back to your GitHub Codespaces environment. In your TERMINAL, type in command **pac auth create --devicecode** which will provide a URL with a code to be used at the provided URL to authenticate.

Enter the code, and use the Office 365 Tenant **username and password** provided in the lab resources to sign-in.

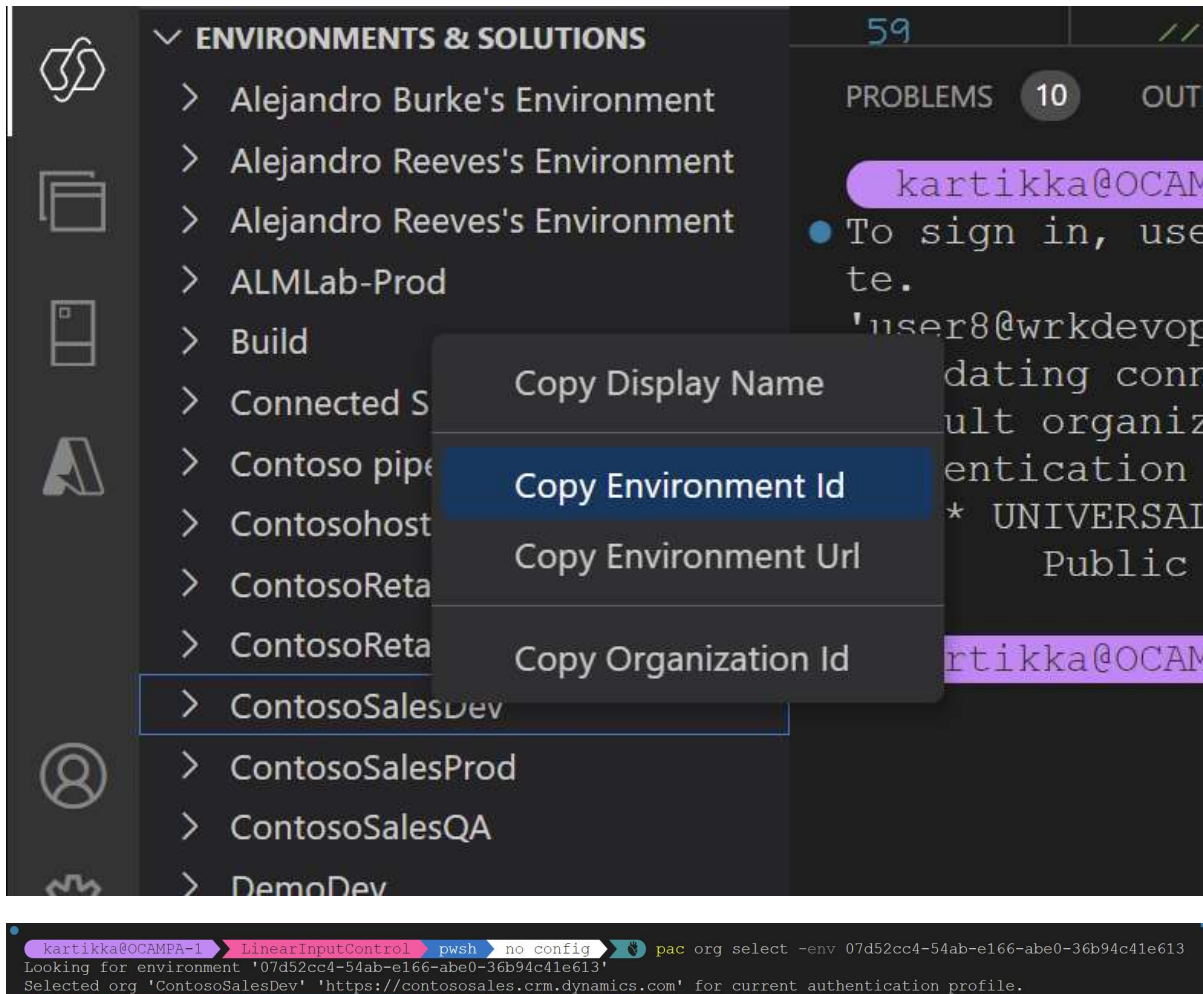


```
kartikka@OCAMPA-1 > LinearInputControl pwsh no config > pac auth create --deviceCode
To sign in, use a web browser to open the page https://microsoft.com/devicelogin and enter the code RUR8Y2PR4 to authentic
te.
'user8@wrkdevops.onmicrosoft.com' authenticated successfully.
Validating connection...
Default organization: T-Logistics-Dev
Authentication profile created
* UNIVERSAL
Public
https://orgab9e0a25.crm.dynamics.com/ : user8@wrkdevops.onmicrosoft.com
```

2. Once authenticated, now let us select our development environment, that has been created for us using the command **pac org select --environment <ENVIRONMENT\_URL -or- ENVIRONMENT\_ID>**

**NOTE:** ENVIRONMENT\_ID is provided at the top of the lab. For ENVIRONMENT\_URL, refresh your Power Platform in the left blade, you will see an authenticated profile (*top*) and list of environments (*dev and test at the botton*), **right-click** on the **DEV** environment to get the Environment URL.

Let us select the environment (*select DEV environment*) we want to push our new controller for use.



3. Now make sure that the developer environments have the code component enabled. You can run a check by using command **pac org who** to ensure you are connected to the correct environment.
4. Now open another browser tab and using Office 365 Tenant credentials provided in the Lab Resources area, go to <https://admin.powerplatform.microsoft.com> (we have pre-provisioned environments that you will see when you click Environments on left pane)

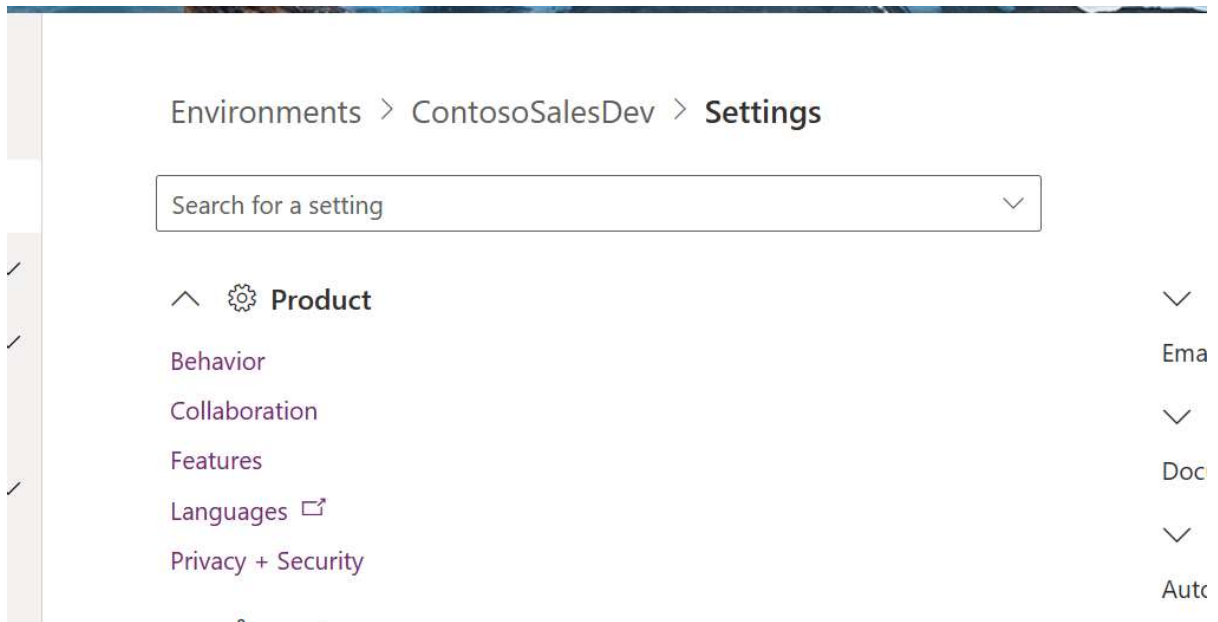
Select **Environments** on left pane.

Select **dev environment** > **Settings** -> **Product** -> **Features**

Select **test environment** > **Settings** -> **Product** -> **Features**

**NOTE:** This step must be done for both **dev** and **test**.





Set below flag to **ON** and click **Save**.

## Power Apps component framework for canvas apps

Enables Power Apps component framework feature that allows the execution of code that may not be generated by Microsoft when a maker adds code components to an app. Make sure that the code component solution is from a trusted source.

[Learn more](#)

Allow publishing of canvas apps with code components

☒ On

5. Now that is done. Go to GitHub Codespaces tab. Let us do a **pac pcf push** command at the TERMINAL prompt from the **LinearInputControl/controls** folder (Command: *pac pcf push --publisher-prefix prefix*).

**NOTE:** We used "lic" as prefix (*prefix must be between 2-8 letters*)

This will push the content into your environment.

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

Time Elapsed 00:00:39.85
Building temporary solution wrapper: done.
Dropping temporary solution wrapper zip file: done.
  File at C:\Users\kartikka\Documents\GitHub\control\LinearInputControl\obj\PowerAppsTools_linp
Debug\PowerAppsTools_linp.zip.
Importing the temporary solution wrapper into the current org.

Solution Importing...

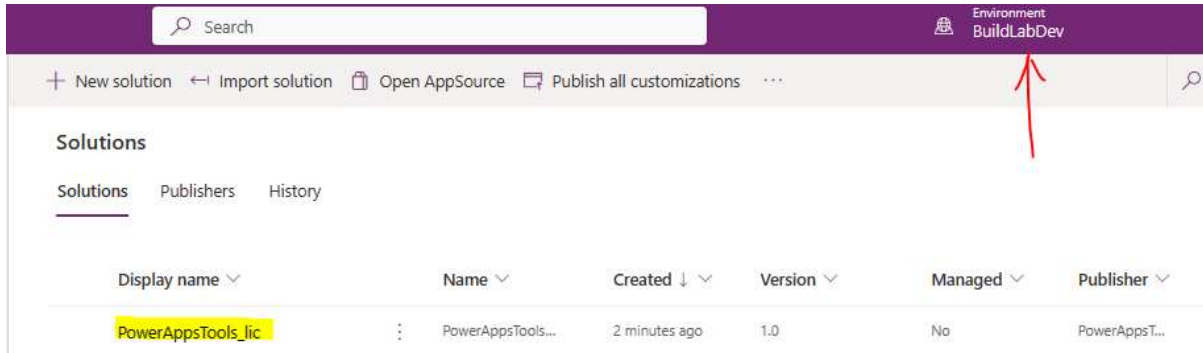
Solution Imported successfully.

Publishing All Customizations...
█
```

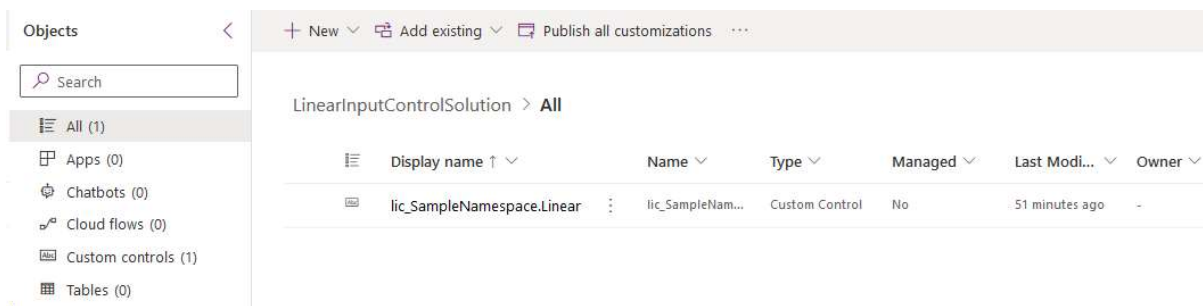
And now PCF control has been pushed into the environment.

## 6. Now let us build an application.

Open a new browser tab and using Office 365 Tenant credentials provided in the Lab Resources area, go to <https://make.powerapps.com> and select DEV environment at top right. Now select **Solutions** on the left. We should see the imported solution in our DEV environment.



Clicking the **solution** will show Code component imported into your DEV environment as part of the PCF PUSH in this solution (see image below).



## Add the PCF control to an App

Now we are going to create a Canvas app and insert/import the PCF control that we built.

Starting at the screen from the last step (**Environments > Solutions >** select **PowerAppsTools\_lic**) in previous section.

1. To build a new Canvas app, click on **New > App > Canvas app**.



+ New

Add existing

Publish all customizations

App

Automation

Chatbot

Dashboard

Report

Security

Table

More

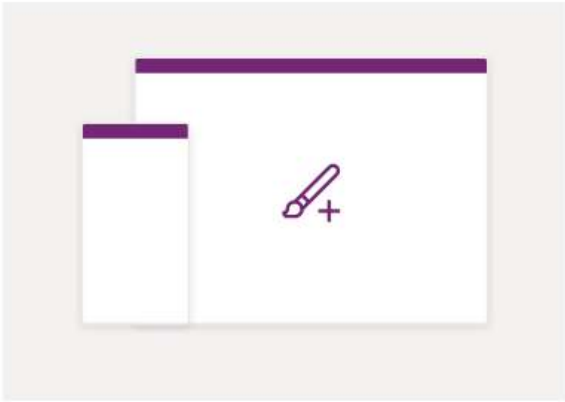
Canvas app

Model-driven app

Page

Name	Type
Namespace.Linear	lic_SampleNam... Custom Control

Canvas app from blank



App name \*

BuildLab App

Format

Tablet

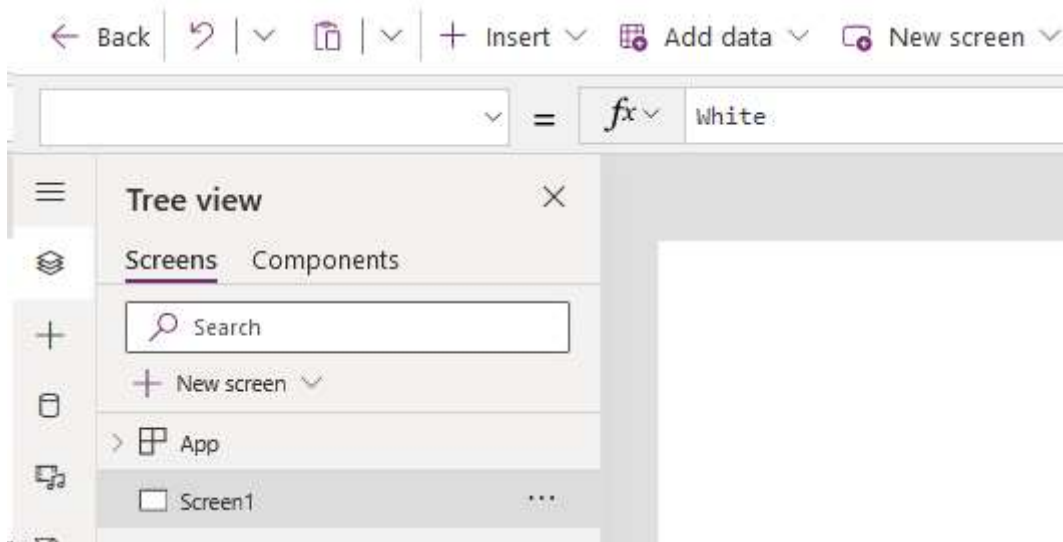
Phone

Design the app you want, and connect it to hundreds of data sources.

Canvas app

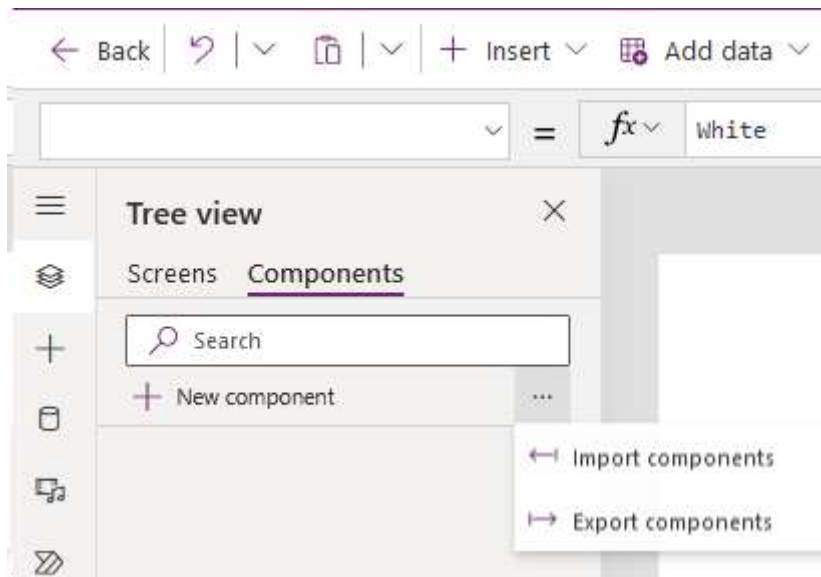
Create

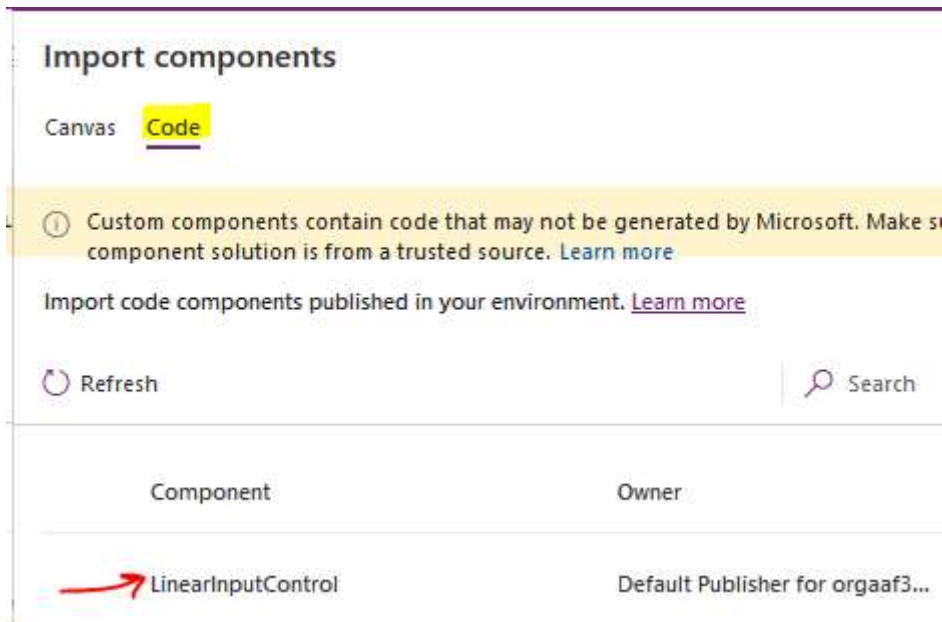
Cancel



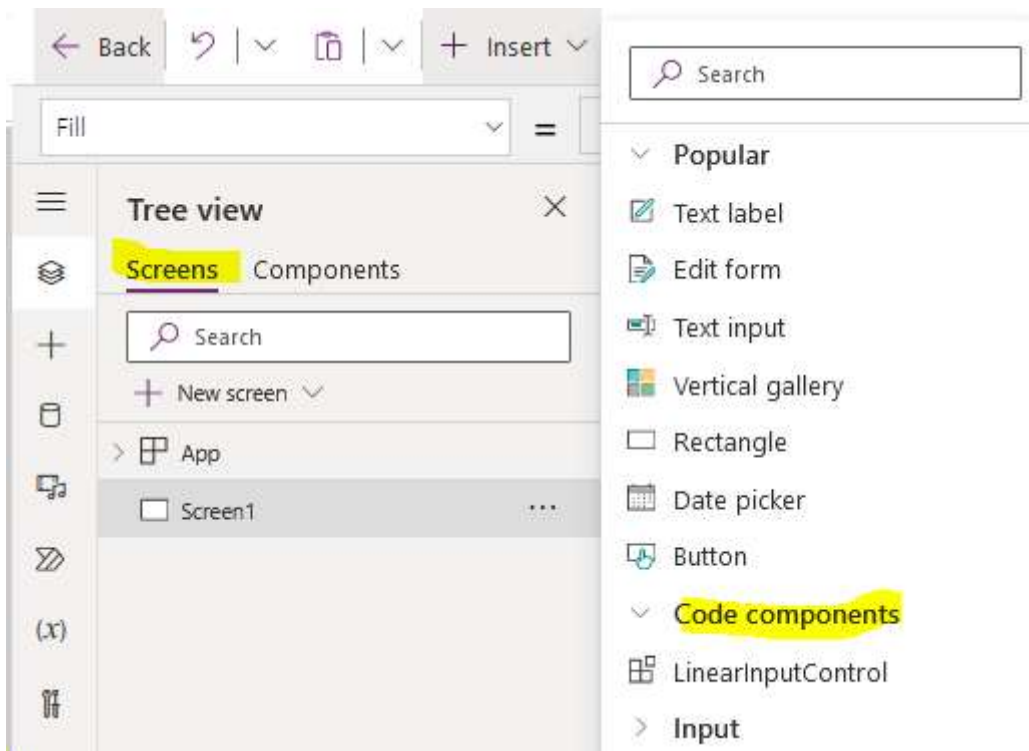
2. Now we will import the component that we built to be used in the Canvas App.

Click **Components > Import components > Code** > select component (*LinearInputControl*)

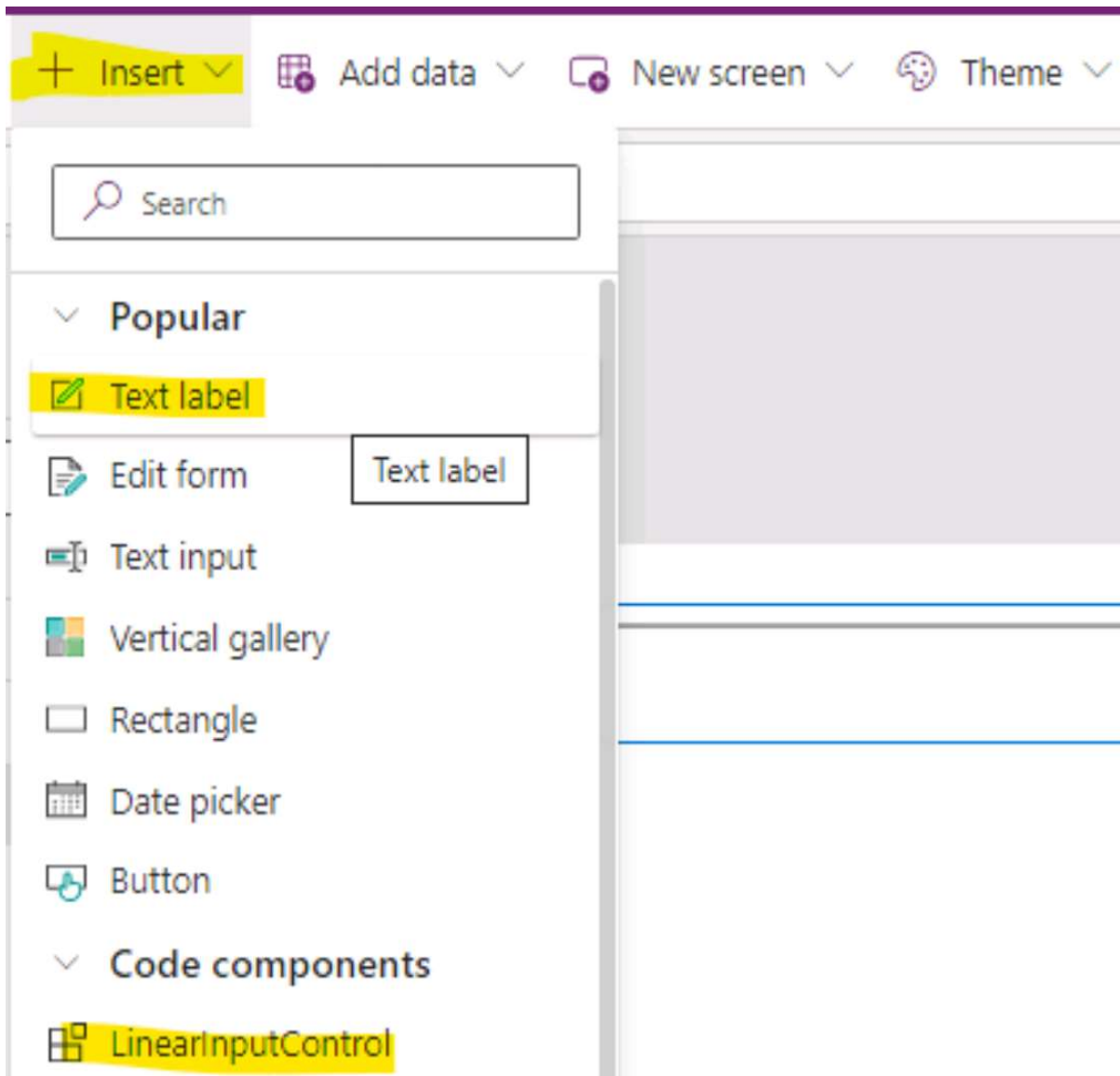




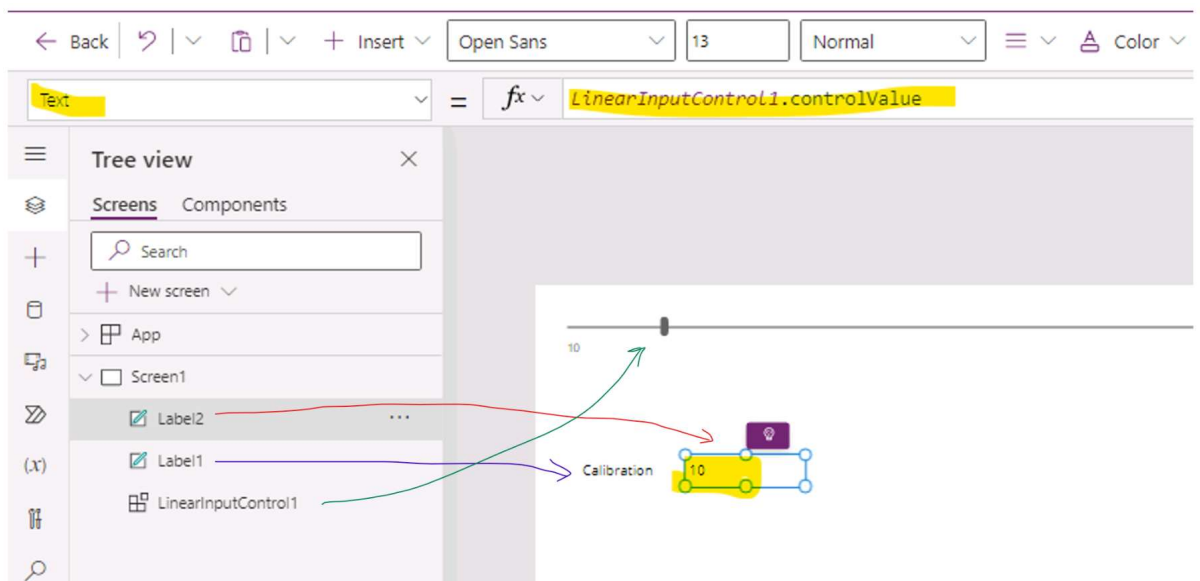
Click **Insert > Code components > Select LinearInputControl > Import** to add to the screen.



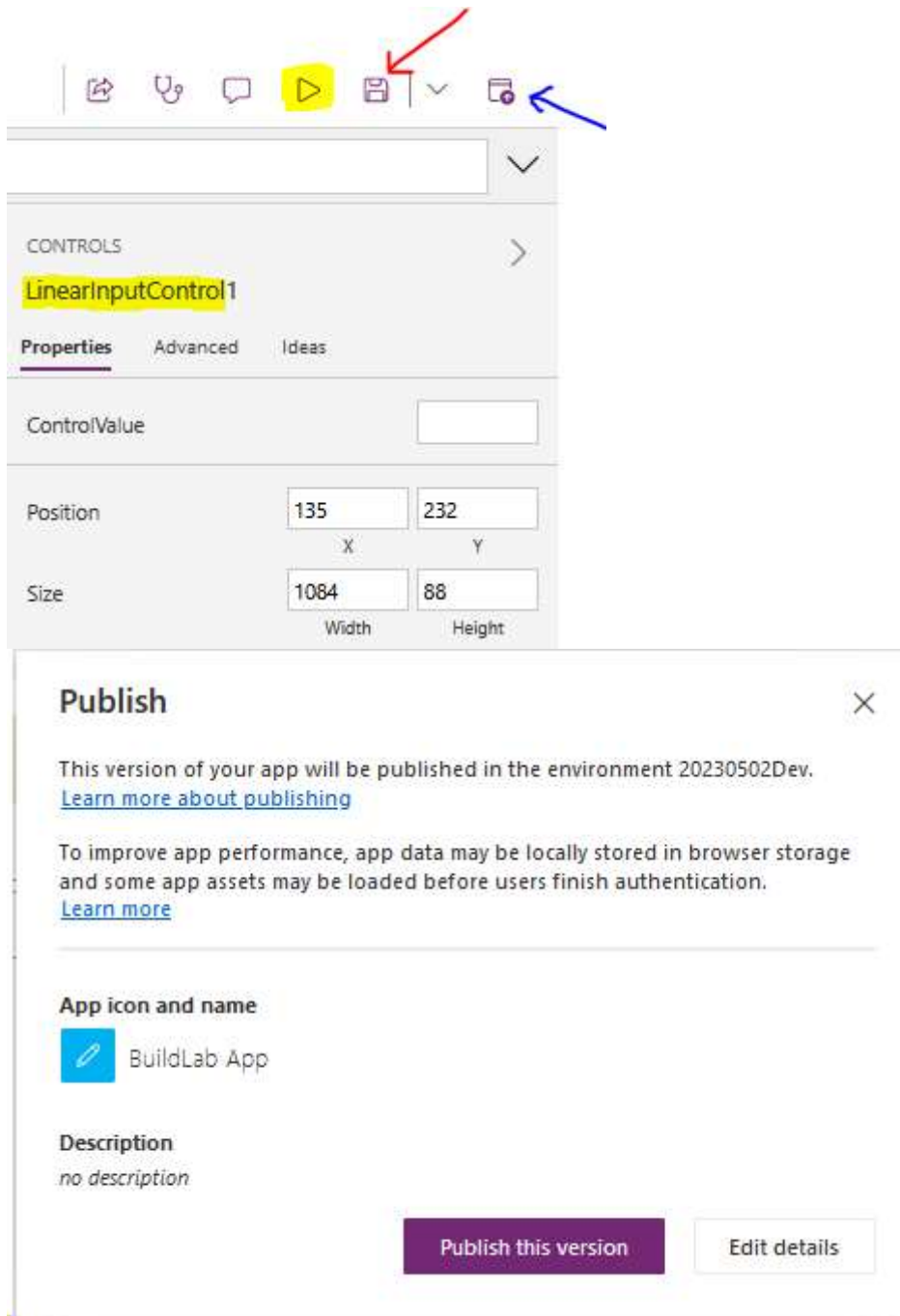
3. Insert a **Code component** that we imported and resize it as you see fit. Also, change the "Text" value in the formula bar to **<ControlName>.controlValue**. In the below case, ControlName is *LinearInputControl1*.
4. Now you are on a **Screens** tab in **Tree view**. Insert two(2) **Text labels** as shown below.
  - (a) One for word Calibration



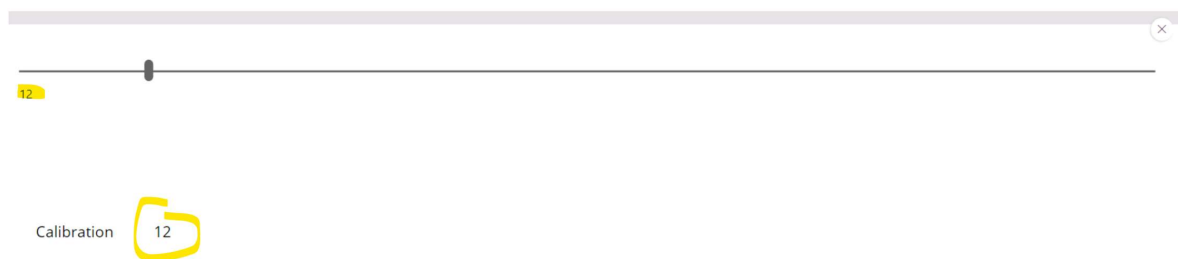
(b) Second for Slider value



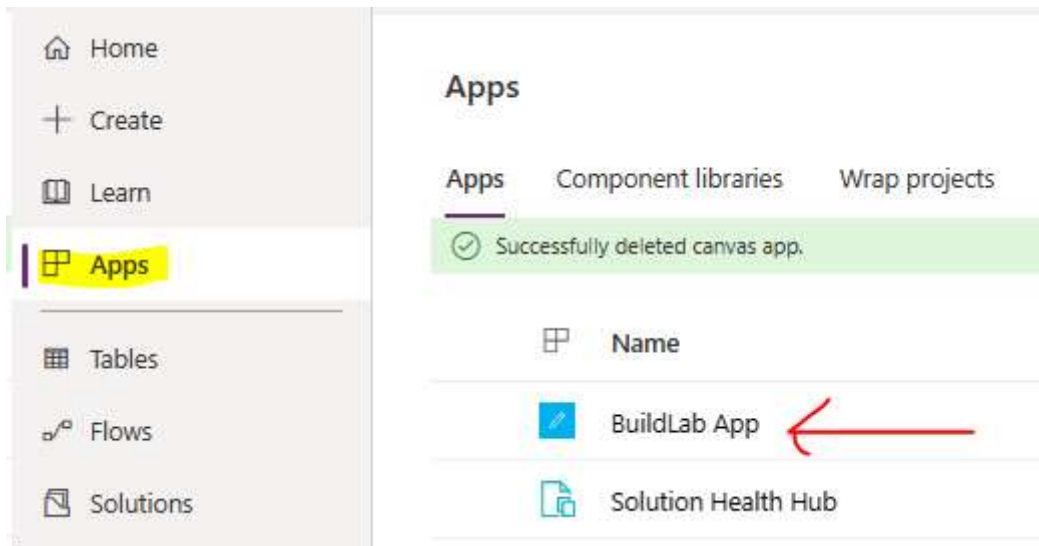
5. Now **Save** and then **Publish** your app.



6. Now **Play** the app to test.



You should see your App in Apps on the left.



Now we have created a code component and imported solution with the code component to the Power Platform environment.

## Export the new app with PCF control to GitHub

Let's use GitHub Actions to promote our Canvas app solution embedded in the PCF control to the TEST environment.

1. On **GitHub webpage**, click **<> Code > .github/workflows** and you shall see 2 workflows. (1) export-solution-on-main and (2) deploy-unmanaged-application.

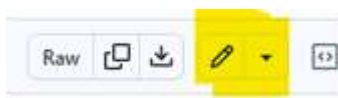
We will **make the changes** listed below to **both of the workflows** and run them in order. Be sure to change the values (*SOLUTION\_NAME*, *ENVIRONMENT\_URL*, *CLIENT\_ID/ApplicationId*, *TENANT\_ID*) in both of your workflows.

**NOTE:** On GitHub Codespaces tab, click on *Power Platform Extension*, under left bottom side of your screen you will see 2 environments under "ENVIRONMENTS & SOLUTIONS".

**For export-solution-on-main workflow.** Right-click on the **DEV environment** to get then Environment\_URL.

**For Deploy Unmanaged Application workflow.** Right-click on the **TEST environment** to get then Environment\_URL.

Click **Edit** icon (*like below*) to edit workflow.



```

solution_name:
  description: "name of the Solution"
  required: true
  default: <SOLUTION_NAME, i.e, PowerAppsTools_lic>

```

Get **Environment URL** from Codespaces under Power Platform (*left bottom window*)

```

environment_url:
  description: "http endpoint of your Dataverse environment"
  required: true
  default: "<ENVIRONMENT_URL>"

```

Get **Client ID** (*ApplicationId at the top*) and **Tenant ID** from Lab Instructions (*top*)

```

CLIENT_ID: '<CLIENT_ID_FROM_THE_LAB_INSTRUCTIONS>'
TENANT_ID: '<TENANT_ID_FROM_THE_LAB_INSTRUCTIONS>'

```

Once updated SOLUTION\_NAME, CLIENT\_ID (*which is Application Id*), and TENANT\_ID, click **Commit changes** and then **Commit changes** again to save.

- Under the **Actions** menu in GitHub, you will see your 2 workflows. Let's run workflow **export-solution-on-main**.

The screenshot shows the GitHub Actions interface. On the left, the 'Actions' menu is open, showing 'All workflows' and 'Deploy Umanaged Application'. The 'export-solution-on-main' workflow is selected. The main area displays the workflow details for 'export-solution-on-main' (export-solution-on-main.yml). It includes a search bar for 'Filter workflow runs' and a 'New workflow' button. Below this, there is a section for '1 workflow run' with filters for Event, Status, Branch, and Actor. A message states 'This workflow has a workflow\_dispatch event trigger.' with a 'Run workflow' button. A list of workflow runs shows a successful run for 'export-solution-on-main' (export-solution-on-main #3: Manually run by snizar00) completed 13 minutes ago in 1m 47s.

Your GitHub Actions (*workflows*) should be completed successfully.





Manually triggered 15 minutes ago

Status

Total duration

Artifacts

 snizar00  958cf64


Success

1m 47s




—

export-solution-on-main.yml


on: workflow\_dispatch


 export-from-dev

1m 38s




← export-solution-on-main

 export-solution-on-main #3

Re-run all jobs 

Summary

Jobs

 export-from-dev




Run details


Usage

Workflow file


export-from-dev

succeeded 14 minutes ago in 1m 38s


 Search logs  

>  Set up job


1m 5s

>  Run actions/checkout@v3


1s

>  who-am-i action


8s

>  export-solution action


21s

>  unpack-solution action


0s

>  Commit changes

1s

>  Post Run actions/checkout@v3

1s

>  Complete job

0s

## Have a GitHub workflow to deploy from Dev to Test

In the last section, we exported our App with PCF control. In this section, we will promote our solution with the PCF control and the App to the Test environment.

1. Now let's run a second workflow **Deploy Unmanaged Application** and that should complete successfully.

← Deploy Umanaged Application

✓ Packing PowerAppsTools\_lic for https://orga6dedd14.crm.dynamics.com/ environment #1 Re-run all jobs ...

Summary

Jobs

- ✓ pack-umanged-solution-import

Run details

- Usage
- Workflow file

Manually triggered 11 minutes ago

snizar00 → b5f5eeb

Status: **Success**

Total duration: **1m 55s**

Artifacts: —

deploy-unmanaged-application.yml  
on: workflow\_dispatch

✓ pack-umanged-solutio... 1m 46s

← Deploy Umanaged Application

✓ Packing PowerAppsTools\_lic for https://orga6dedd14.crm.dynamics.com/ environment #1 Re-run all jobs ...

Summary

Jobs

- ✓ pack-umanged-solution-imp...

Run details

- Usage
- Workflow file

pack-umanged-solution-import  
succeeded 12 minutes ago in 1m 46s

Search logs

- > ✓ Set up job 3s
- > ✓ Run actions/checkout@v3 1s
- > ✓ Pack the solution 5s
- > ✓ Import solution as unmanaged to build env 1m 35s
- > ✓ Post Run actions/checkout@v3 0s
- > ✓ Complete job 0s

2. You will see the solution imported to into your **TEST** environment. Click on **Solutions** in left pane, select your **Solution** > click **All** > select your app > **click on three dots** in front of your app and select **Play**

Power Apps

Environment: BuildLabTest

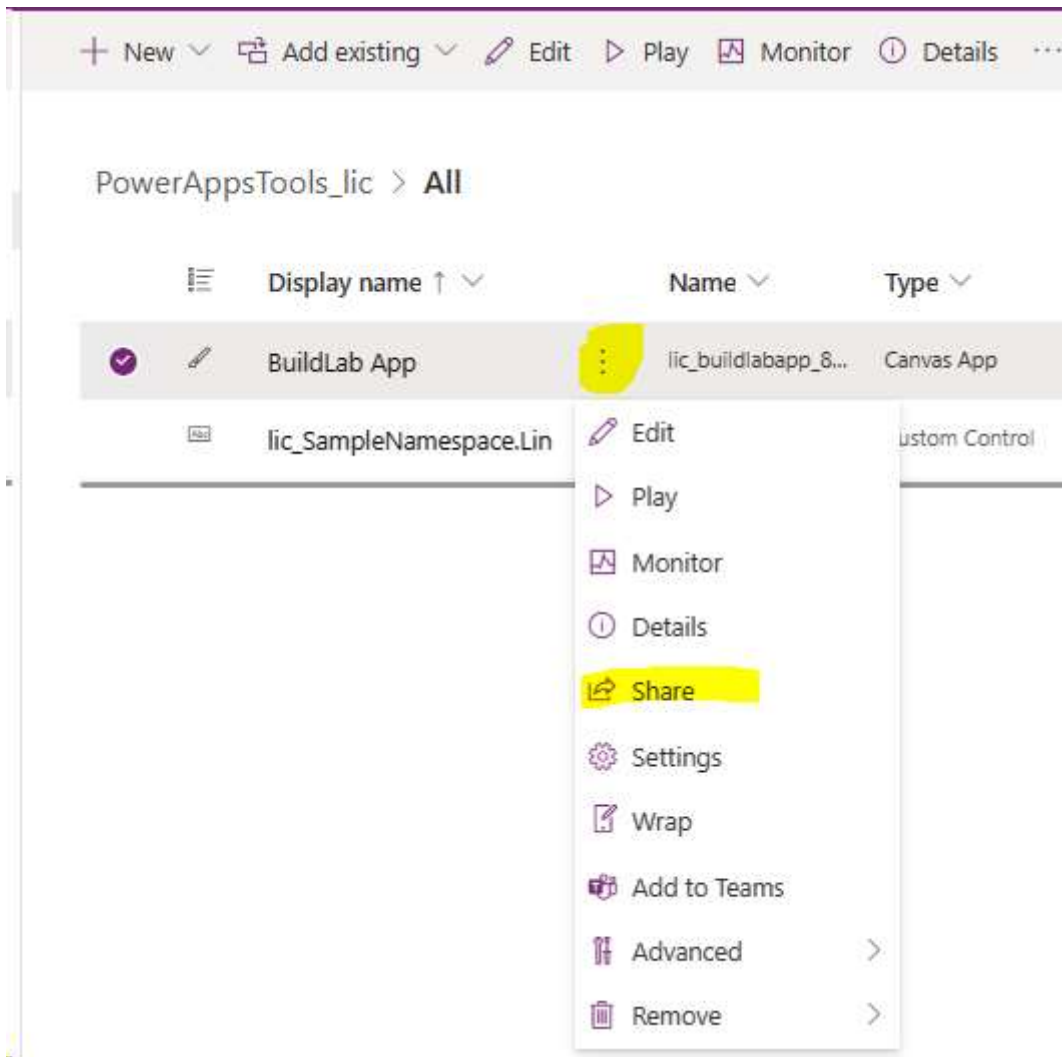
Objects

- All (2)
- Apps (1)
- Chatbots (0)
- Cloud flows (0)
- Custom controls (1)
- Tables (0)

PowerAppsTools\_lic > All

Display name	Name	Type	Managed	Last Modif...
BuildLab App	lic_buildlabapp_8...	Canvas App	No	4 minutes ago
lic_SampleNamespace.Lin	lic_SampleName...	Custom Control	No	4 minutes ago

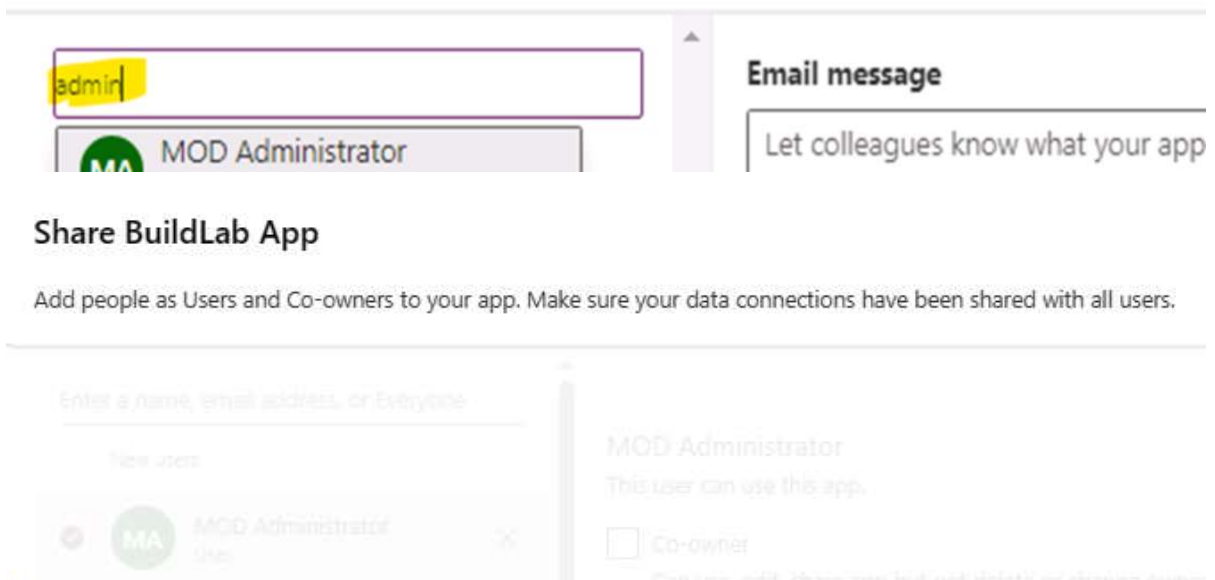
You may have to share the App in order to play using the below steps. Click *three dots* > **Share** (as highlighted below).



Type **admin** in the field and click **Share**.

## Share BuildLab App

Add people as Users and Co-owners to your app. Make sure your data connections have been



After sharing, click on **Cancel** to exit the screen. Now **Play** the App, you should be able to test it.

## Congratulations!

---

You have successfully completed this Module, to mark the lab as complete click **End**.