**DEVOps – Create CI/CD Pipeline for SPFX**

1. Create local Project and add to the Repo in Github.

Graphical user interface

Description automatically generated

**Step 1 – Build the Continuous Integration (CI) Pipeline**

In Azure DevOps, click on “Pipelines” in the left navigation area:

Click on the “New pipeline” button in the upper right of the screen:

In the New pipeline window, select “Other git” and select the Project and folder. Click on the “Continue”.

Click on “Empty job”:

Graphical user interface, text, application, email

Description automatically generated

Let’s change the name of this Pipeline. Click the name of the Pipeline at the top and change it to something more meaningful:

Graphical user interface, text, application

Description automatically generated

Click on the plus sign next to Agent Job 1 to add task.

In the search box, type in “node”. Hover over “Node.js tool installer”, then click on the “Add” button:

Graphical user interface, application

Description automatically generated

Click on the Node task you just added and change the version to 10.x:

Graphical user interface, application

Description automatically generated

Add another task by clicking on the plus sign again. This time, we are going to type “npm” in the search box. Hover over “npm” and click on the “Add” button:

Click on the plus sign to add another task. In the search box, type in “gulp”. Hover over “Gulp” and click on the “Add” button:

Click on the Gulp task you just added and fill out the following:  
**Display name:** gulp bundle --ship  
**Gulp File Path:** gulpfile.js  
**Gulp Task(s):** bundle  
**Arguments:** --ship  
  
Add another gulp task  
**Display name:** gulp package-solution  
**Gulp File Path:** gulpfile.js  
**Gulp Task(s):** package-solution  
**Arguments:** --ship

Add another task for “copy files”:

**Display name:** Copy Files to: $(build.artifactstagingdirectory)/drop  
**Contents:** \*\*\\*.sppkg  
**Target Folder:** $(build.artifactstagingdirectory)/drop

Add the last task. Search for “publish build artifacts”:

**Display name:** Publish Artifact: drop  
**Path to publish:** $(Build.ArtifactStagingDirectory)/drop

Graphical user interface, text, application

Description automatically generated

Save the pipeline by clicking on the “Save & Queue” drop down menu and selecting “Save”:

Keep the default path and click on the “Save” button:

**Step 2 – Building the release pipeline for Continuous Deployment (CD)**

Click on “Releases” in the left navigation area:

Click on the “New” drop down menu and select “New release pipeline”:

Lets select an “Empty job” for our template:  
Click on “Add an artifact”:

In the “Source (build pipeline)” drop down menu, select the name of the build pipeline we created in step 1 above:

In the “Source alias” field type in an alias that you’ll need to remember for future steps…Write it down. Click “Add” when you are done:

Click on “1 job, 0 task”:

We will now add tasks like we did in step 1.

Click on the plus sign to add a task. Search for “node” and add the “Node.js tool installer”.  
**Version Spec:** 10.x  
  
Click on the plus sign and add the “npm” task.  
**Display name:** npm custom  
**Command:** custom  
**Command and arguments:** install -g @pnp/office365-cli

Add a “Command line” task by clicking the plus sign and searching for “command line”.  
**Display name:** Connect to SharePoint App Catalog  
**Script:** o365 login https://$(tenant).sharepoint.com/$(catalogsite) --authType password --userName $(username) --password $(password)  
  
Add another “Command line” task.  
**Display name:** Add Solution Package to App Catalog  
**Script:** o365 spo app add -p $(System.DefaultWorkingDirectory)/ReactDemo-CI/drop/sharepoint/solution/react-demo.sppkg --overwrite --appCatalogUrl https://$(tenant).sharepoint.com/$(catalogsite) --scope sitecollection

In the above script, ReactDemo-CI is the alias that we wrote down. The solution package for my code in the script is react-demo.sppkg. You’ll want to make sure you have the name of your package correct from your SPFx project. To locate the name, you can do a gulp build, gulp bundle --ship, and a gulp package-solution --ship in Visual Studio Code. You will find the package in your sharepoint/solution folder. If your App Catalog is a site collection app catalog then you MUST include --scope sitecollection. If it is a tenant app catalog, then you can remove --scope sitecollection.

Add a “Command line” task by clicking the plus sign and searching for “command line”.  
**Display name:** Deploy App  
**Script:** o365 spo app deploy --name react-demo.sppkg --appCatalogUrl https://$(tenant).sharepoint.com/$(catalogsite) --scope sitecollection  
  
In the above script, make sure you use the name of your package. I am using in my example, react-demo.sppkg. If your App Catalog is a site collection app catalog then you MUST include --scope sitecollection. If it is a tenant app catalog, then you can remove --scope sitecollection.

A picture containing diagram

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Click on the “Variables” tab:

Add the following variables:  
**Name:** catalogsite  
**Value:** sites/johndemo/appcatalog  
**Name:** password  
**Value:** Whatever your password is. Click on the lock to lock/hide your password  
**Name:** tenant  
**Value:** Whatever your tenant name is  
**Name:** username  
**Value:** Your login email address

Graphical user interface, application

Description automatically generated

Click on the “Pipeline” tab, then click on the “Continuous deployment trigger” button:

Graphical user interface, application

Description automatically generated

Click on the “Save” button:

To have your pipeline be triggered when you push your code to the repository, click on “Pipelines” in the left navigation area and select the CI Pipeline we created:

Click on the “Edit”button in the upper right corner:

Click on the “Triggers” tab:

Select the “Enable continuous integration” checkbox, then click on the “Save & queue” and select “Save”:

Graphical user interface, application

Description automatically generated

**Step 3 – Test**

Click on “Pipelines” in the left navigation area.

Select the CI Pipeline we created

Click on the “Run pipeline” button:

As soon as the job has completed successfully, you can also check the progress of the CD pipeline. Click on “Releases:

Click on the current release. In my example it is “Release-5”, then click on “In progress”:

After a successful run, you should see that your app is now deployed in the app catalog:

Graphical user interface, text, application, email

Description automatically generated