Publish Python packages(.whl) to Azure Artifacts feed using Azure Pipelines and install packages into Databricks

If you have worked with Python, you must be familiar with the pip command used to install packages. This article will show you how to create a wheel file for your custom packages and import it in other projects.

When you use pip to install modules or packages, you must have unknowingly installed a few wheel files as well. A wheel file is like a zip file in many ways, you compress all your python files and dependencies into a single file. You can use this file in different projects or the cloud. Installing a wheel file is as simples as installing a package using pip. They can also be helpful when you are collaborating with others or when you need to deploy your projects.

Step 1: Create a Wheel package (.whl) project

Step 2: Publish python package (.whl) into Azure Artifact Feed using Azure Pipelines

Step 3: Deploy python package into Azure Databricks DBFS and install packages

Step 4: Connect to Azure Artifact Feed from local system

Step 1: Create a Wheel package (.whl) project

Prerequisites:

In cmd prompt -

python --version –> To check python version installed

pip --version -> To check pip (python package manager) version installed

pip list –> to get list of packages installed on your system

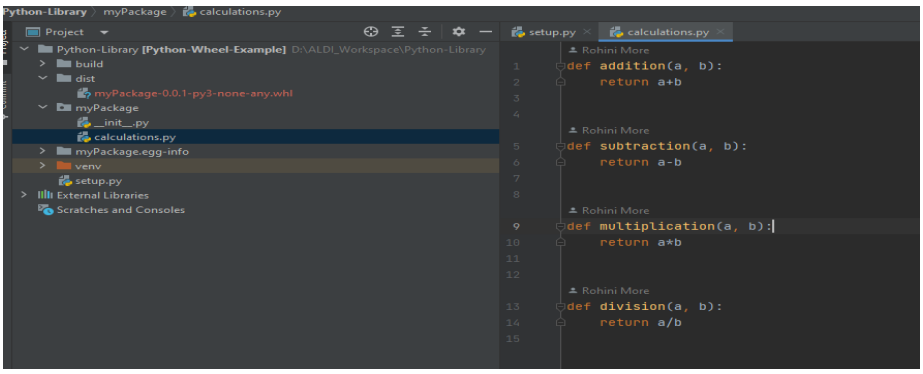
Packages are required to install to create wheel(.whl) package file -

pip install build setuptools twine

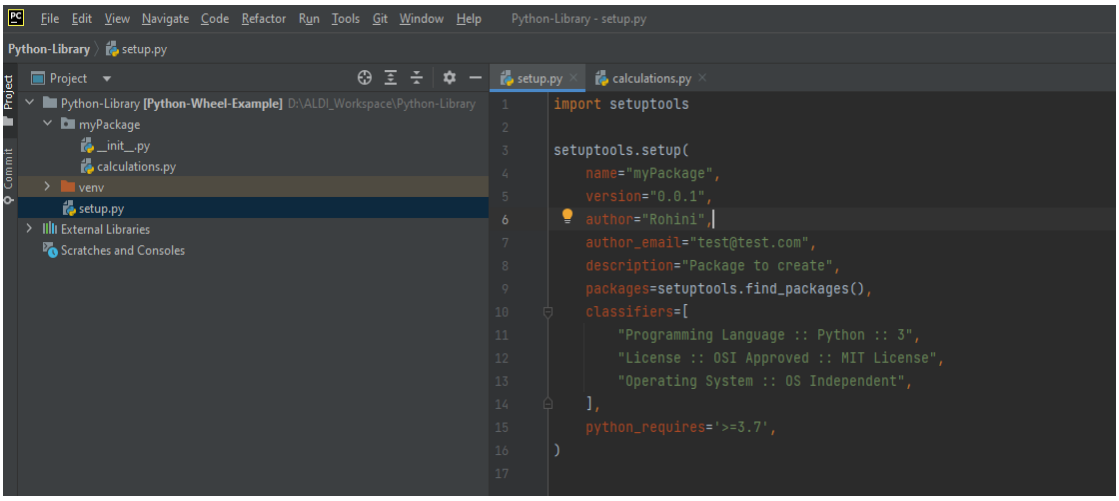
pip install wheel

1. **Create a sample project in python- e.g., Python-Library**

- Add __init__.py file
- Add calculations.py file (add simple functions like below)



2. Add setup.py file outside main folder as given below



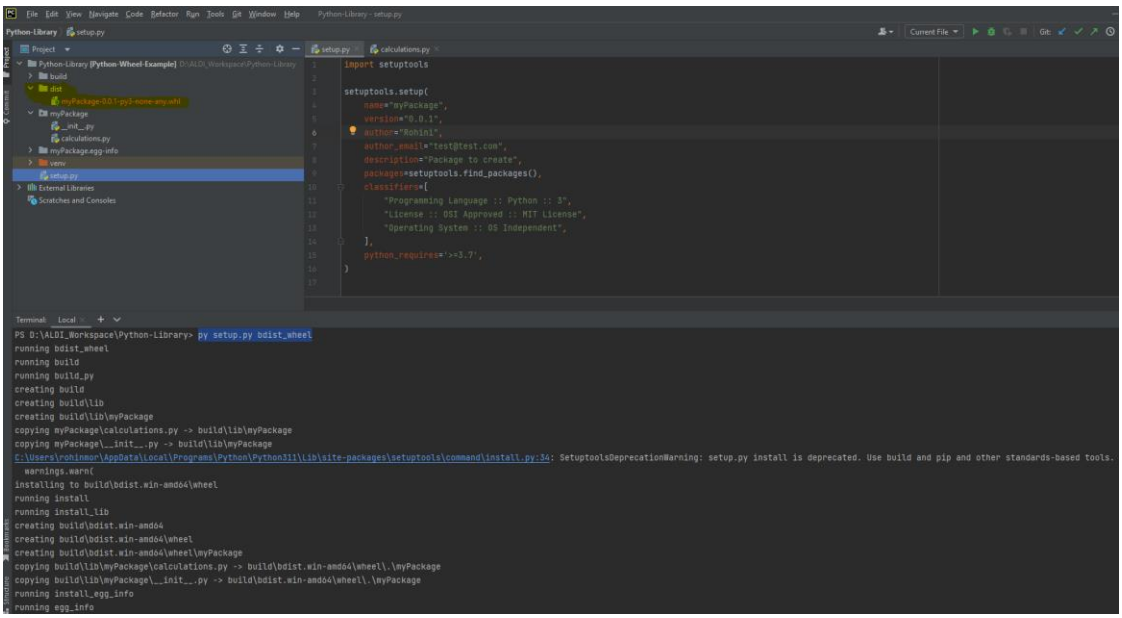
3. Setup virtual environment (venv)
4. Run the below command – This will create a .whl package file in dist folder for 'myPackage' project.

   > py setup.py bdist_wheel –universal (use py or python)

   OR

   > pip install build

   > py -m build

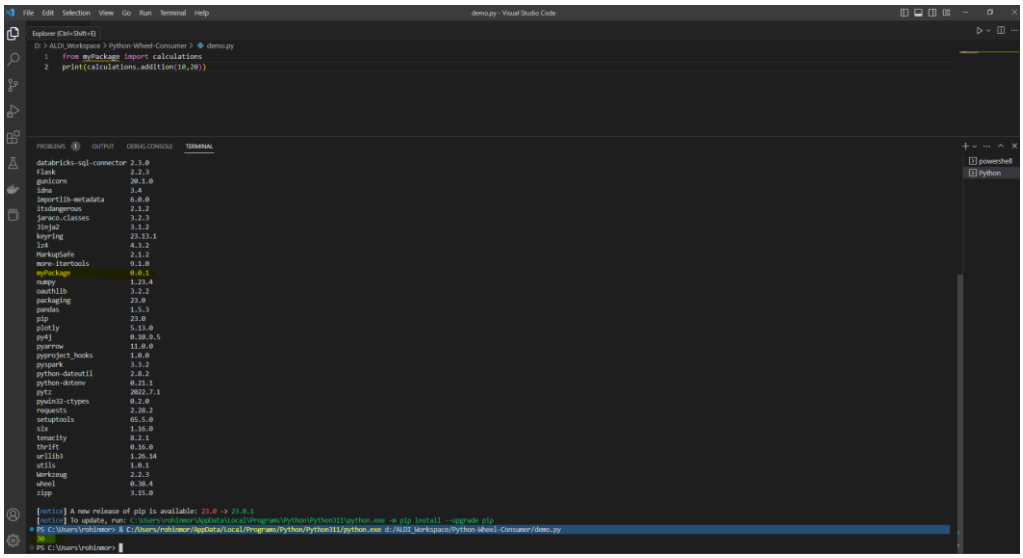


5. To test on local system
    a. Create a file demo.py in different folder
    b. Add below two lines of code i.e import calculations from 'myPackage' and call method addition()
    c. Run below command to install .whl package – Go to project folder path \Python-Library

    > pip install dist\myPackage-0.0.1-py3-none-any.whl

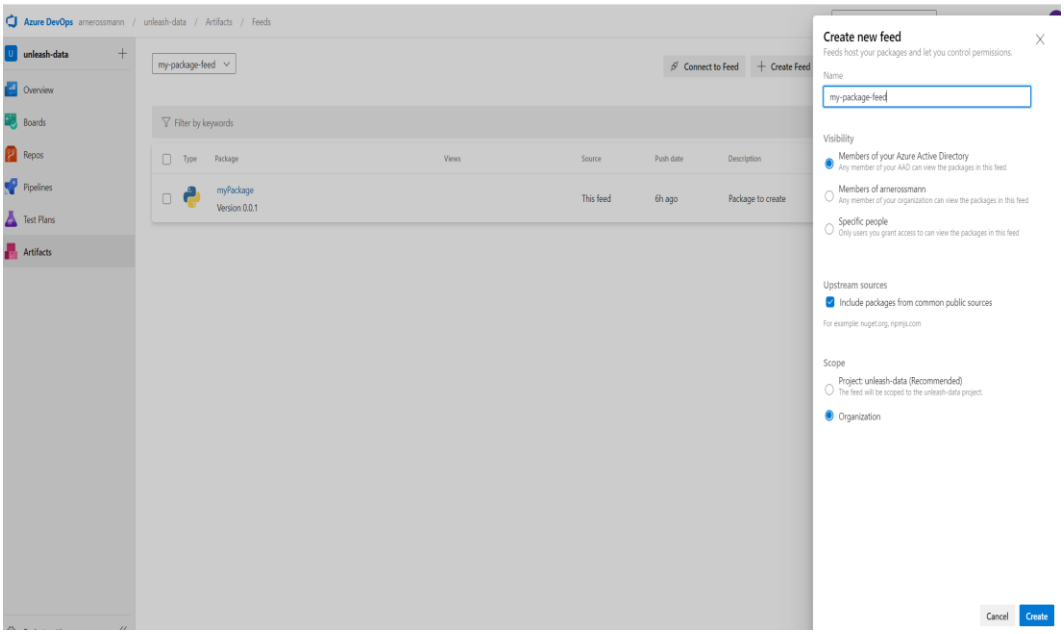
    > pip list - to check that 'myPackage' is installed successfully on local system

    d. Run the demo.py file – import 'myPackage' and print result.

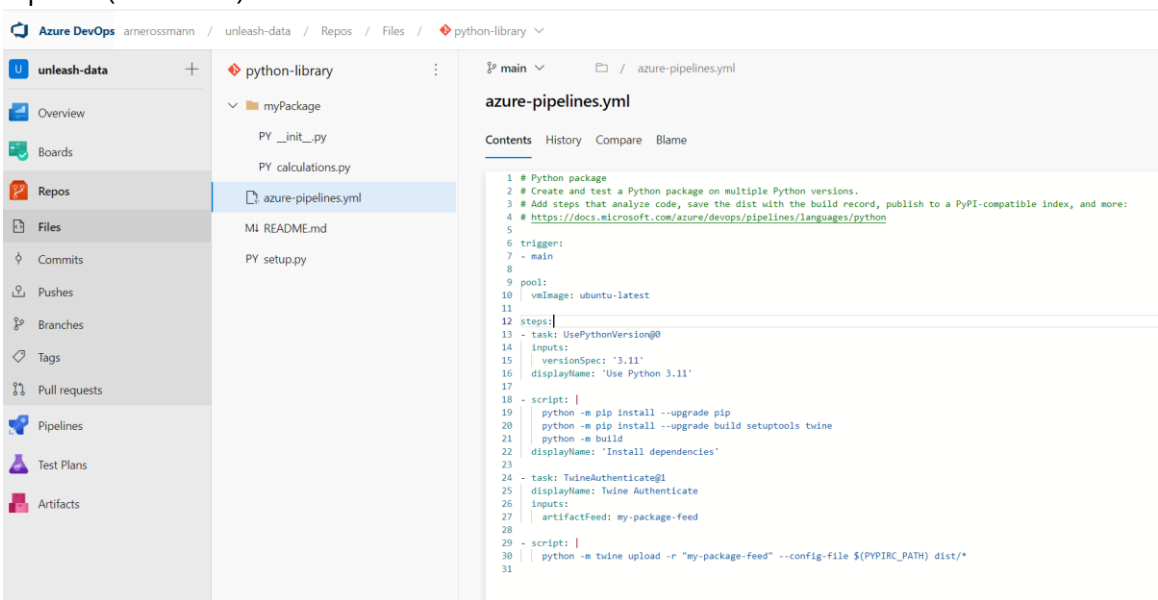


Step 2: Publish python wheel package (.whl) into Azure Artifact Feed using Azure Pipelines

1. Create a new feed under Artifacts as shown below



2. Create a new Repository and push your 'Python-Library' project then create a Azure Pipeline(YAML file) as shown below

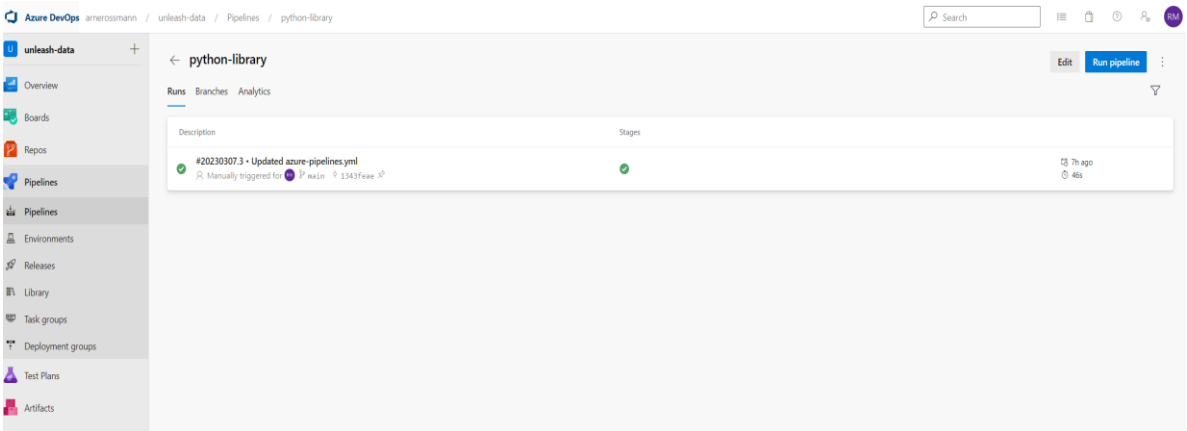


(Note – other way to create .whl package in the azure pipeline)

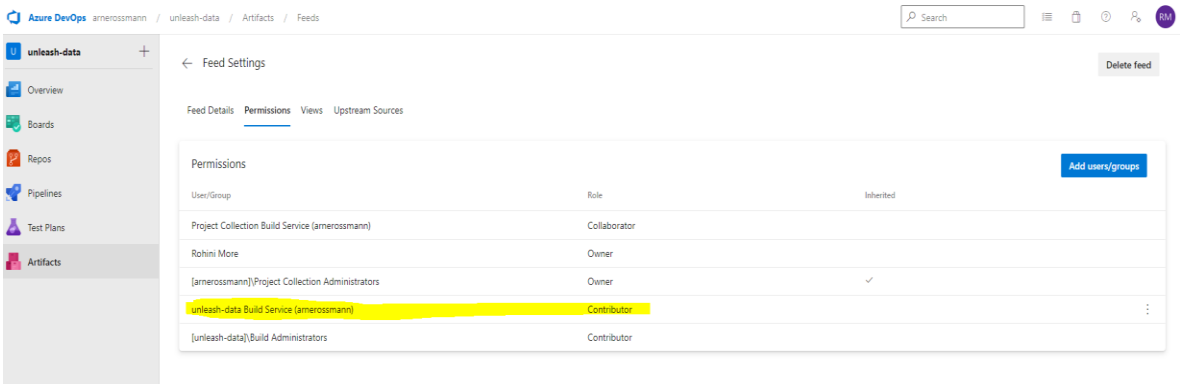
YAML Copy

```yaml
- script: |
    pip install wheel
    pip install twine

- script: |
    python setup.py bdist_wheel

- task: TwineAuthenticate@1
  displayName: Twine Authenticate
  inputs:
    artifactFeed: <PROJECT_NAME/FEED_NAME>          #For an organization-scoped feed, artifactFeed: <FEE

- script: |
    python -m twine upload -r feedName --config-file $(PYPIRC_PATH) dist/*.whl
```

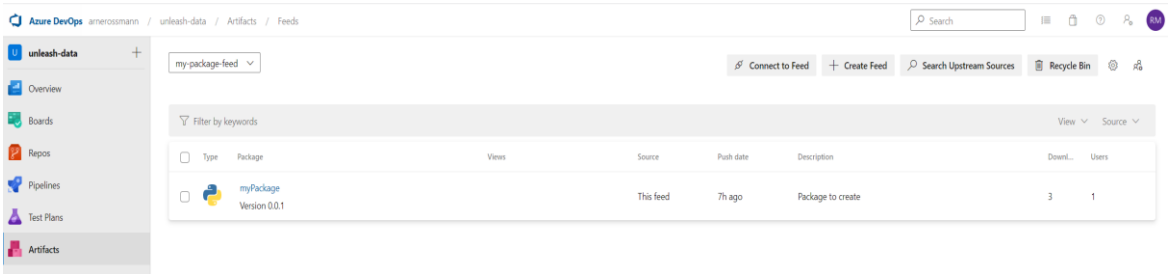
3. Build new Pipeline and run it



4. If build fails, then add permissions to Feed for build service as a contributor role and re-run build.

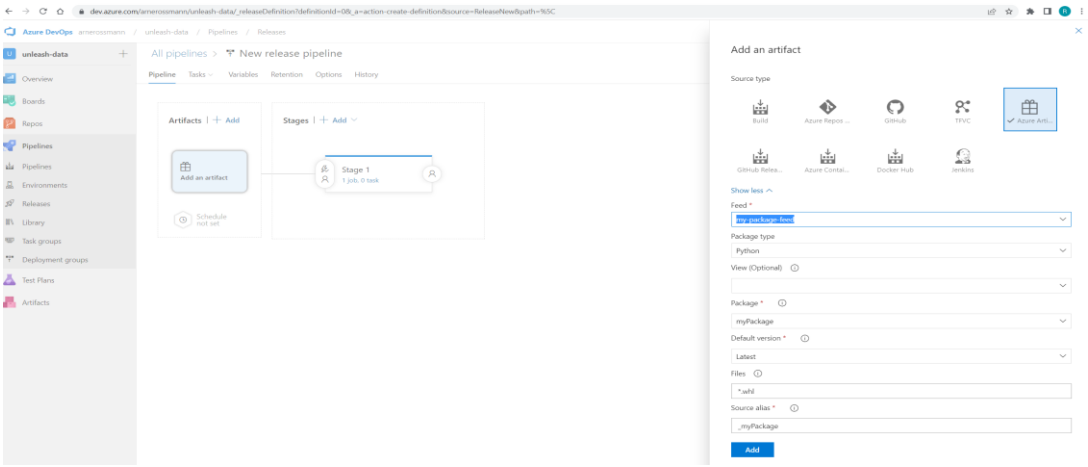


5. Once the pipeline successfully run, it will create a package 'myPackage' (.whl) in Artifact Feed

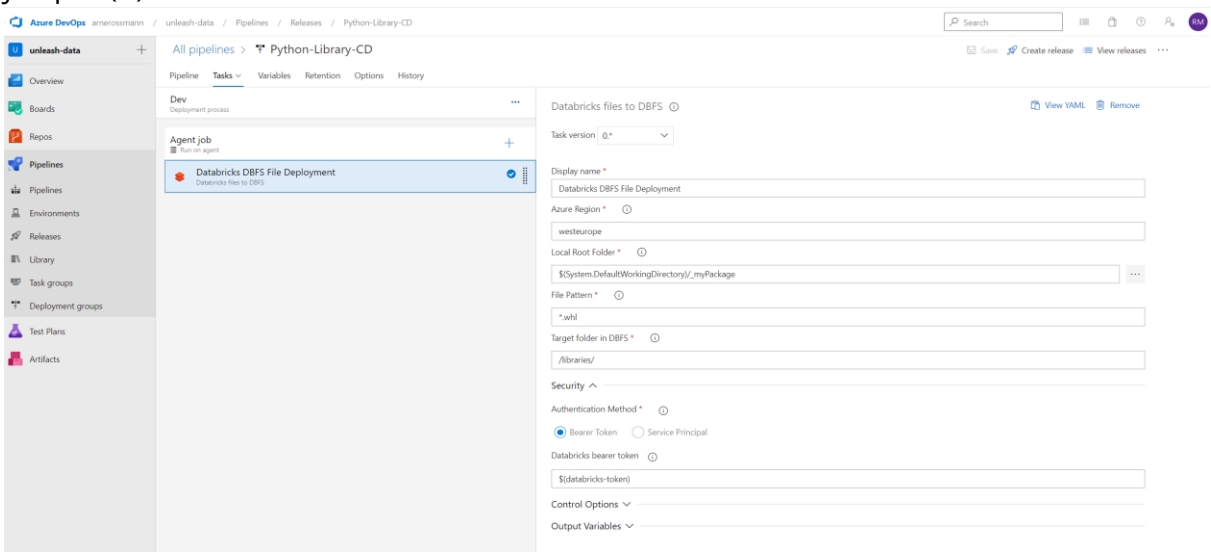


Step 3: Deploy python wheel package into Azure Databricks DBFS and install package

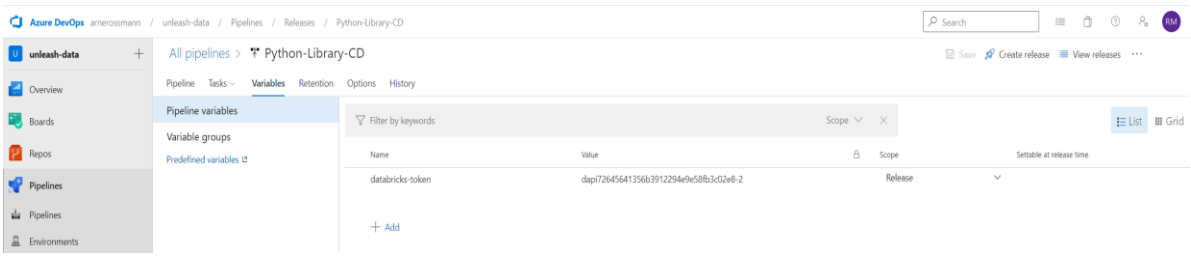
1. Create a Release pipeline - Select Empty job as a template
2. Click on Artifacts - select Azure Artifact as a source type and select your Feed and fill other details

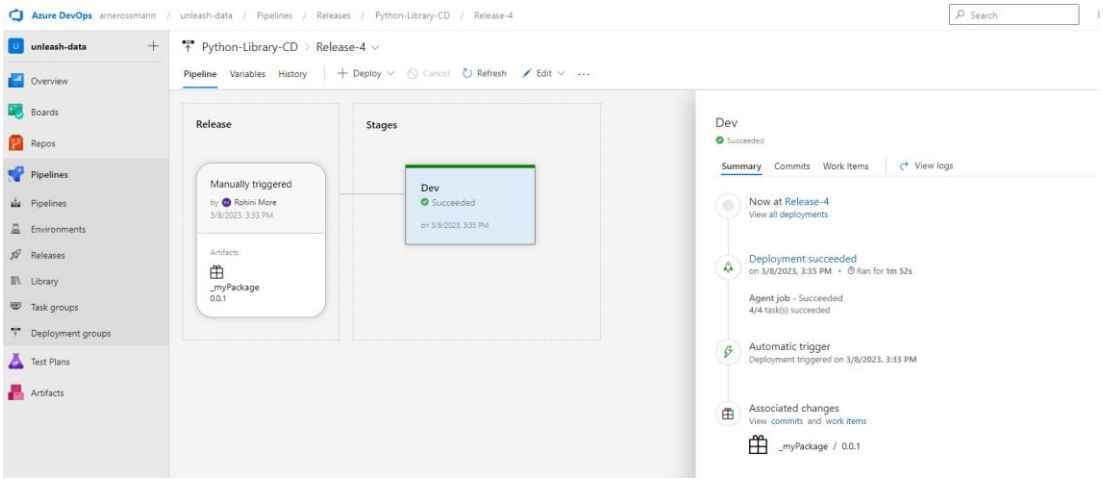


3. Click on Stages, rename it to Dev and Add tasks a 'Databricks DBFS File Deployment' from Agent job plus(+) icon and fill other details as below.



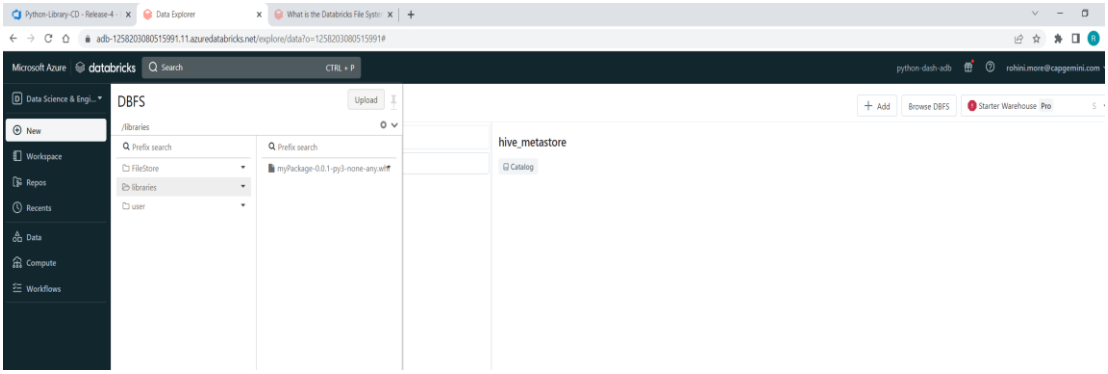
4. Get the access token from databricks and add it into 'databricks-token' variable





5. Run your Release pipeline and after deployment succeeded - go to your Azure Databricks DBFS, now you can see .whl packages are available inside 'libraries' folder.

6. Install package 'pip install myPackage==0.0.1' and import in any notebook



Step 4: Connect to Azure Artifact Feed from local system

There are two primary ways to connect to a feed to publish or consume your Python packages:

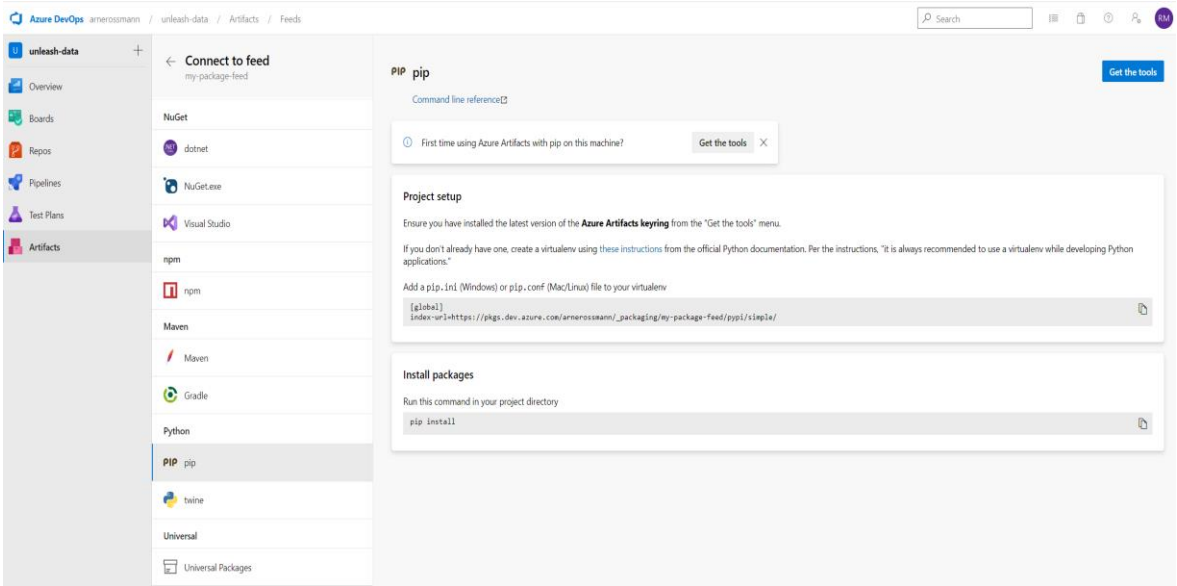
1. Install and use the artifacts-keyring package, which will automatically set up authentication for you.

OR

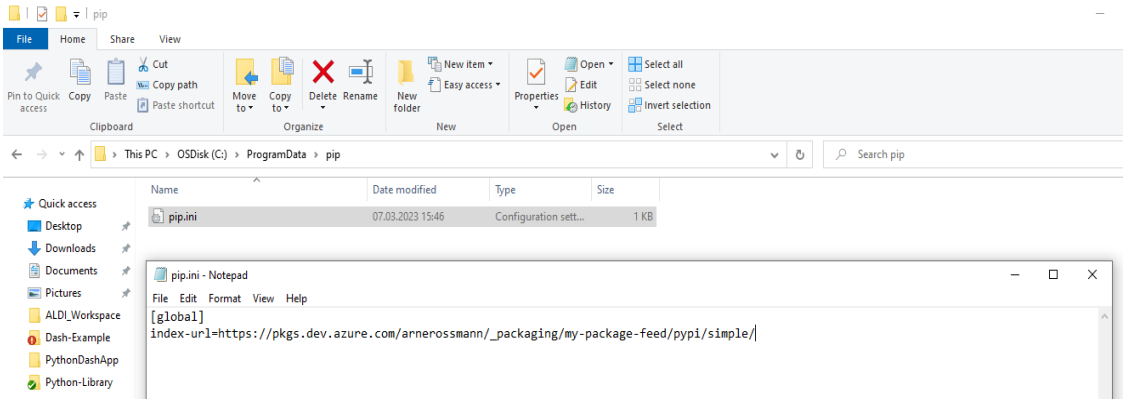
2. Manually set up credentials for *pip.ini/pip.conf*

We will go with second approach –

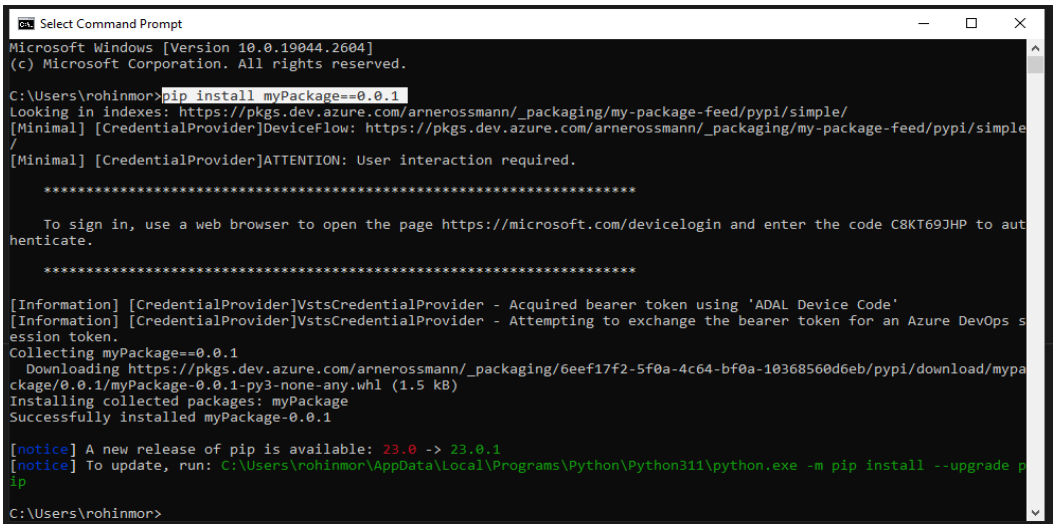
1. Go to Azure DevOps - Artifact tab and click on 'myPackage', click on 'Connect to feed', go to pip, this will give you details about how to connect or consume packages



2. As mentioned in the 'Connect to feed' project setup, create a 'pip' folder in C:\ProgramData\ and add pip.ini file and add given lines.
(Note: ProgramData folder is a hidden so make sure the hidden items are ticked)

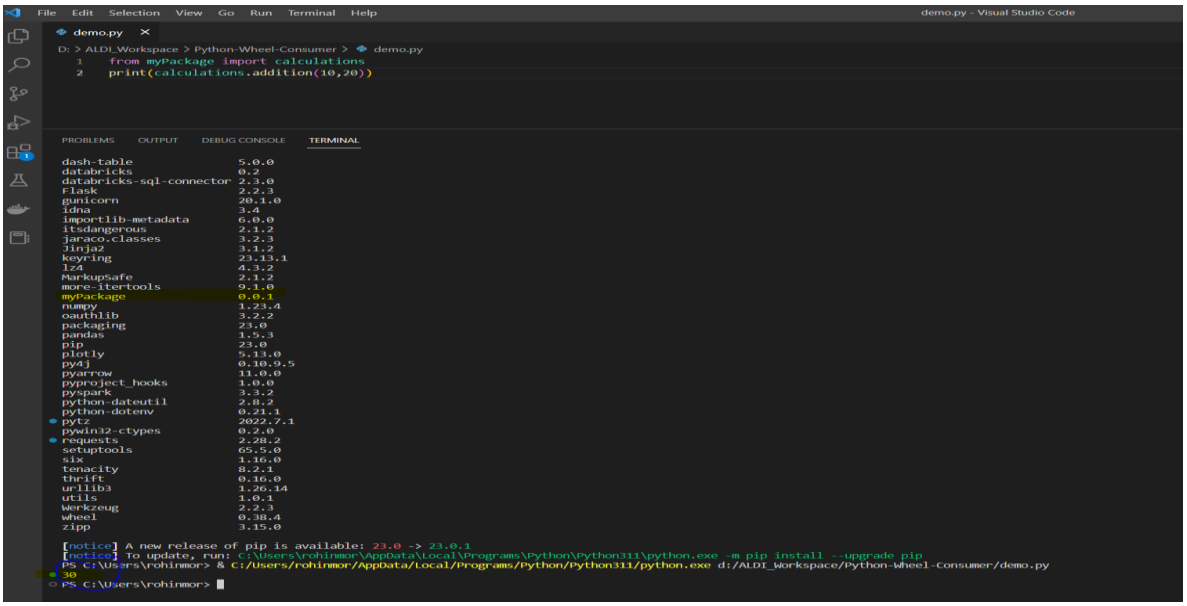


3. Install myPackage using below command, first time it will ask for credentials
> pip install myPackage==0.0.1 (uninstall myPackage if you have installed already while testing locally)



(Note: When you connect to Azure DevOps for the first time, you'll be prompted for credentials. Enter your username (any string) and your personal access token in the appropriate fields (or just sign in by following the terminal provided comments). The credentials will be cached locally and used to automatically sign you in the next time you use the service.)

4. Run the sample demo.py file – import 'myPackage' and it will print the addition numbers result (here 'myPackage' is imported from Azure Artifact Feed)



References:

https://learn.microsoft.com/en-us/azure/devops/artifacts/quickstarts/python-packages?view=azure-devops

https://learn.microsoft.com/en-us/azure/devops/pipelines/artifacts/pypi?view=azure-devops&tabs=yaml

https://menziess.github.io/howto/enhance/your-databricks-workflow/#6-copy-the-package-to-dbfs