

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 simple 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

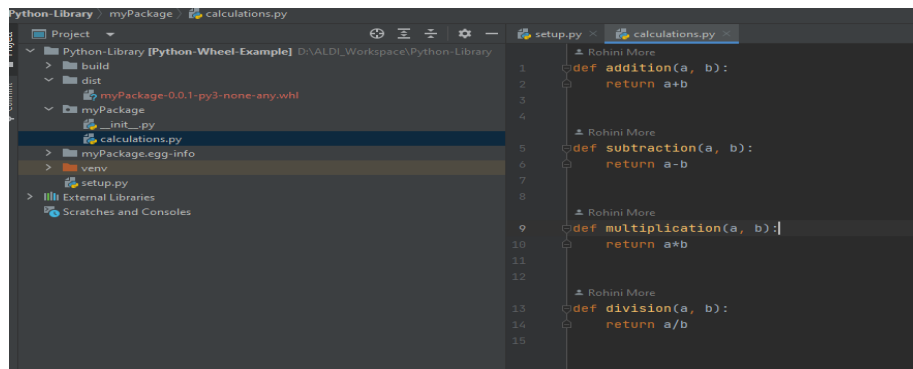
Packages are required to install to create wheel(.whl) package file -

pip install build setuptools twine

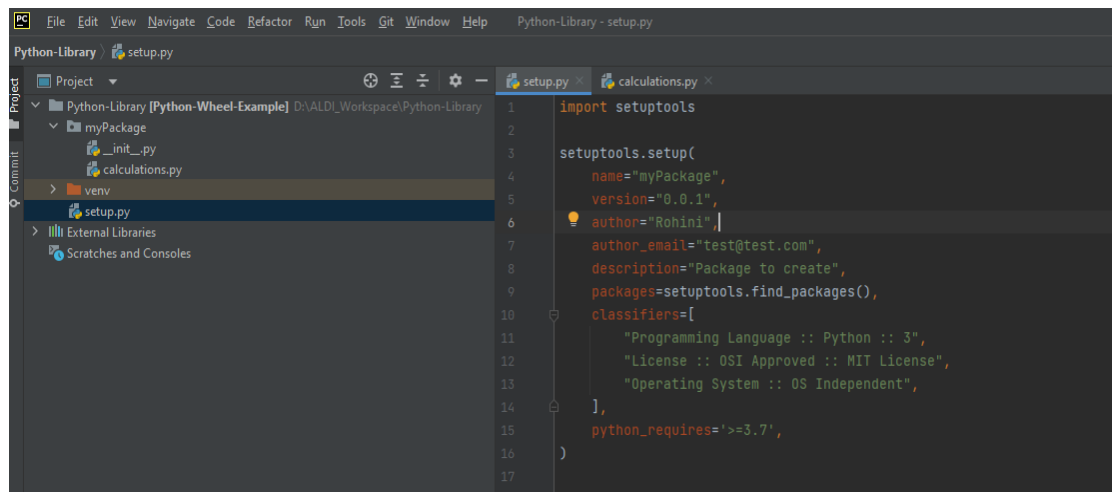
pip install wheel

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

```

1 import setuptools
2
3 setup(
4     name='myPackage',
5     version='0.0.1',
6     author='Rohini',
7     author_email='test@test.com',
8     description='Package to create',
9     packages=setuptools.find_packages(),
10    classifiers=[
11        'Programming Language :: Python :: 3',
12        'License :: OSI Approved :: MIT License',
13        'Operating System :: OS Independent',
14    ],
15    python_requires='>=3.7',
16)

```

```

PS D:\ALDI_Workspace\Python-Library> py setup.py bdist_wheel
running bdist_wheel
running build
running build_py
creating build
creating build\lib
creating build\lib\mypackage
copying mypackage\calculations.py -> build\lib\mypackage
copying mypackage\__init__.py -> build\lib\mypackage
C:\Users\rohini\AppData\Local\Programs\Python\Python311\lib\site-packages\setuptools\command\install.py:36: SetuptoolsDeprecationWarning: setup.py install is deprecated. Use build and pip and other standards-based tools.
  warnings.warn(
installing to build\bdist.win-amd64\wheel
running install
running install_lib
creating build\bdist.win-amd64
creating build\bdist.win-amd64\wheel
creating build\bdist.win-amd64\wheel\mypackage
copying build\lib\mypackage\calculations.py -> build\bdist.win-amd64\wheel\mypackage
copying build\lib\mypackage\__init__.py -> build\bdist.win-amd64\wheel\mypackage
running install_egg_info
running egg_info

```

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.

```

1 from mypackage import calculations
2 print(calculations.addition(10, 20))

```

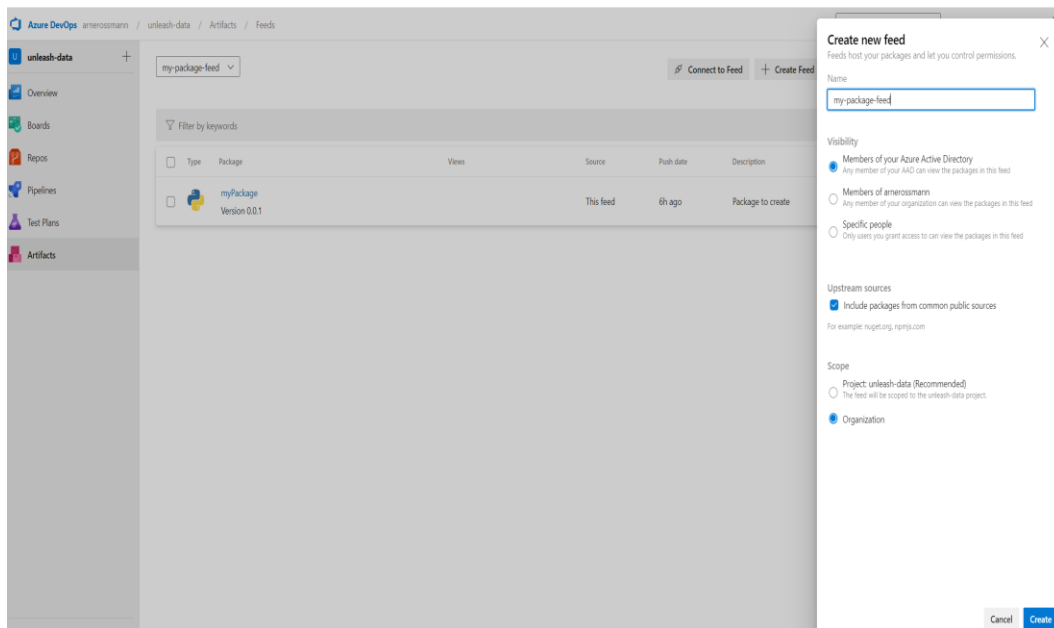
```

[notice] A new release of pip is available: 23.0 -> 23.0.1
[notice] To update, run: python -m pip install --upgrade pip
PS C:\Users\rohini> pip list
Package            Version
-----            -
datatables-qsl-con 2.3.0
flask               2.2.3
gunicorn            20.1.0
idna                3.4
importlib-metadata 4.6.0
itsdangerous        2.1.2
jinja2              3.1.2
jmespath            1.0.1
keyring             23.11.1
lib                 0.1.2
MarkupSafe          2.1.2
more-itertools      8.12.0
myPackage           0.0.1
numpy               1.24.4
oauthlib            3.2.2
packaging           23.0
pandas              1.5.3
pip                23.0
platform           0.12.0
py4j                0.10.9.5
pyarrow            11.0.0
pyarrow_hotels     1.0.0
pyquark            1.2.2
python-dateutil     2.8.2
python-distdev      0.21.1
pytz               2022.7.1
pywin32-ctypes     0.2.0
requests            2.28.2
setuptools          65.5.0
six                 1.16.0
tenacity            8.2.1
thrift              0.16.0
urllib3             1.26.14
utils              1.0.1
werkzeug            2.2.3
wheel               0.38.4
zipp               3.15.0

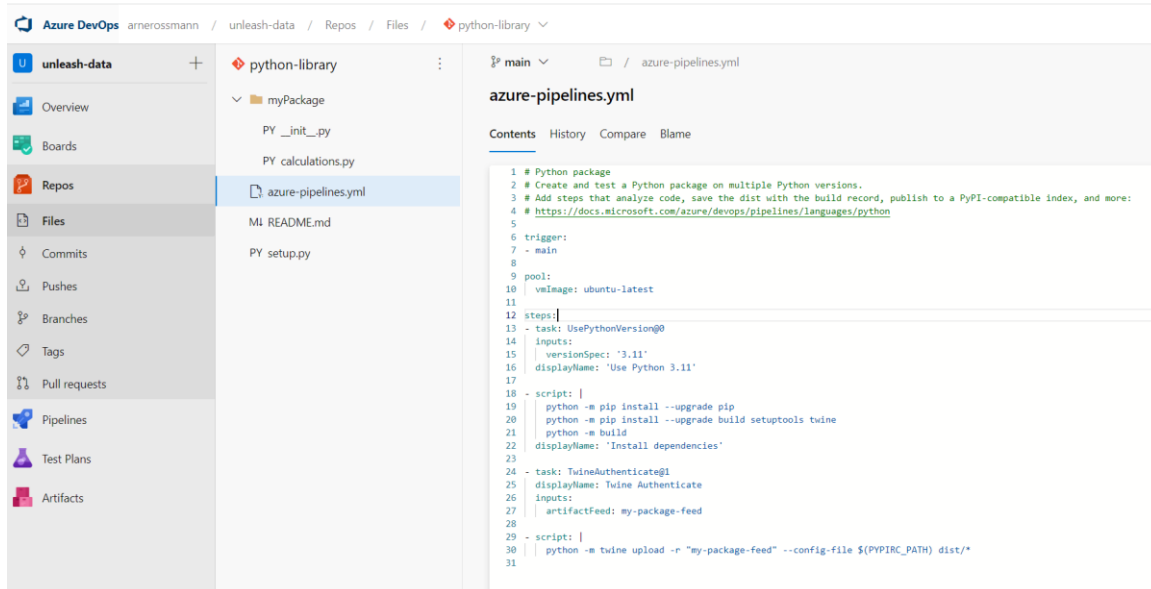
```

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

- 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: <FEED_NAME>

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

```

3. Build new Pipeline and run it

The screenshot shows the 'python-library' pipeline in the 'Runs' tab. A single run is listed with a green status icon, indicating it was successful. The run was manually triggered for the 'main' branch. The left sidebar shows the 'Pipelines' section is active. The top right corner includes a search bar and a user profile icon.

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

The screenshot shows the 'Feed Settings' page for 'my-package-feed'. The 'Permissions' tab is active, displaying a table of permissions. The 'unleash-data Build Service (amersommann)' is highlighted in yellow, showing it has a 'Contributor' role. The table also lists other users and groups like 'Project Collection Build Service (amersommann)' and 'Rohini More'. The left sidebar shows the 'Artifacts' section is active. The top right corner includes a search bar and a user profile icon.

User/Group	Role	Inherited
Project Collection Build Service (amersommann)	Collaborator	
Rohini More	Owner	
[amersommann]Project Collection Administrators	Owner	✓
unleash-data Build Service (amersommann)	Contributor	
[unleash-data]Build Administrators	Contributor	

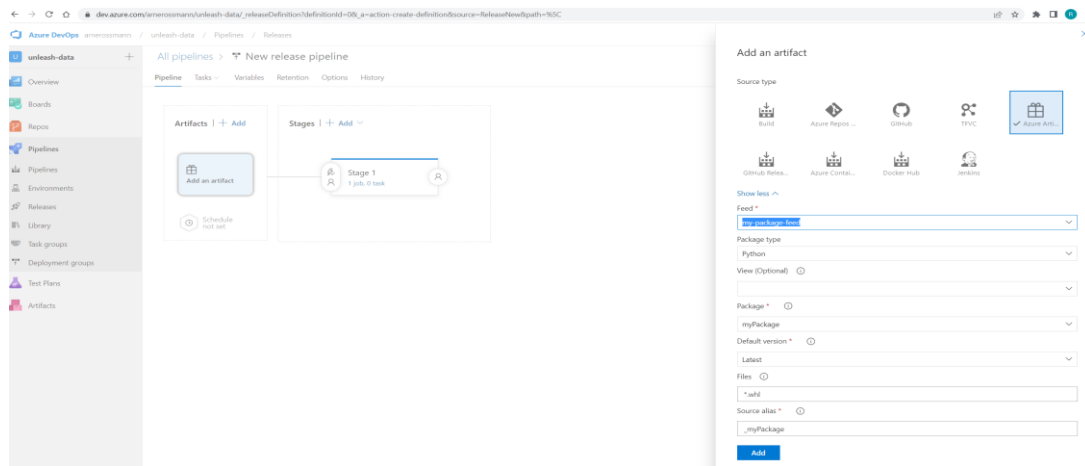
5. Once the pipeline successfully run, it will create a package 'myPackage' (.whl) in Artifact Feed

The screenshot shows the 'my-package-feed' page. The 'my-package-feed' dropdown is selected. The table displays a package named 'myPackage' (Version 0.0.1) created 7h ago. The table has columns for Type, Package, Views, Source, Push date, Description, Download, and Users. The left sidebar shows the 'Artifacts' section is active. The top right corner includes a search bar and a user profile icon.

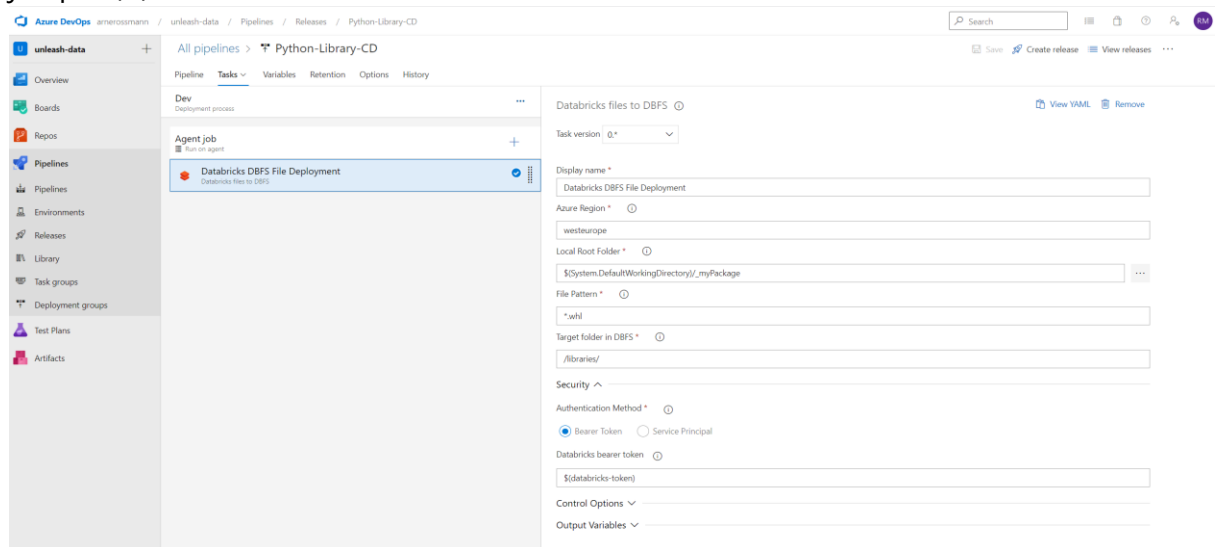
Type	Package	Views	Source	Push date	Description	Download	Users
	myPackage Version 0.0.1		This feed	7h ago	Package to create	3	1

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

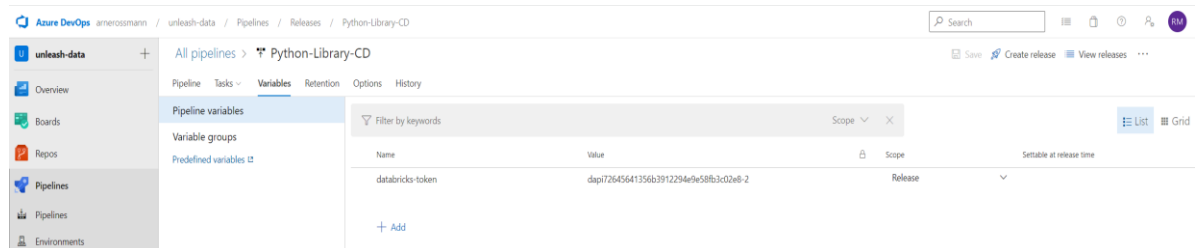
1. Create a Release pipeline - Select Empty job as a template
2. Click on Artifacts - select Azure Artifact as a source type and select your Feed and fill other details

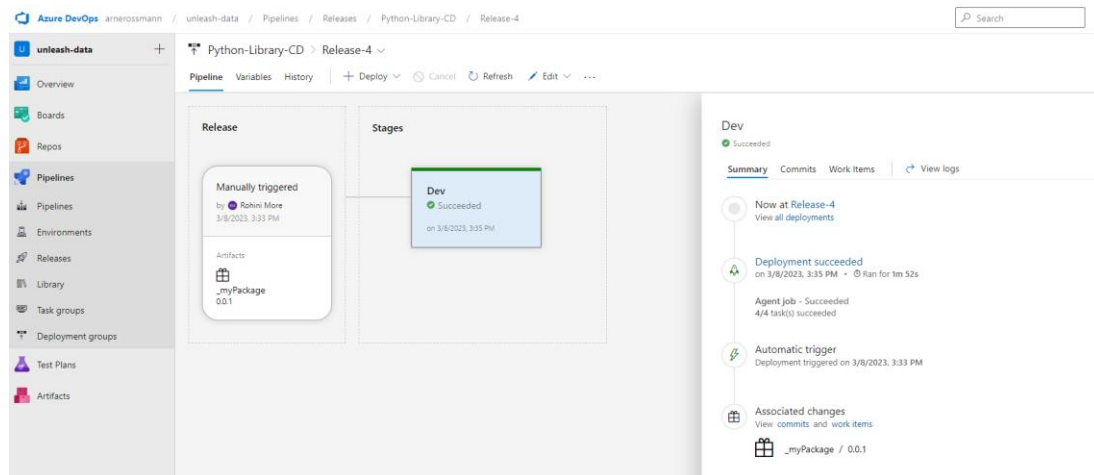


3. Click on Stages, rename it to Dev and Add tasks a 'Databricks DBFS File Deployment' from Agent job plus(+) icon and fill other details as below.

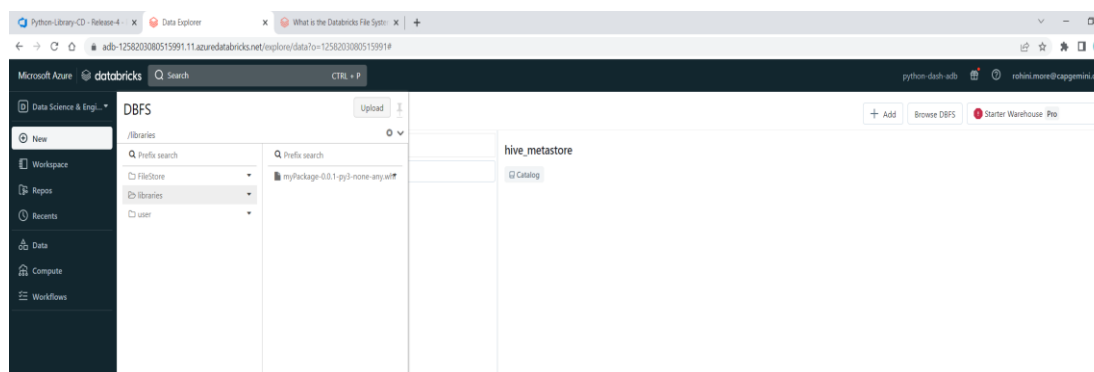


4. Get the access token from databricks and add it into 'databricks-token' variable





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:

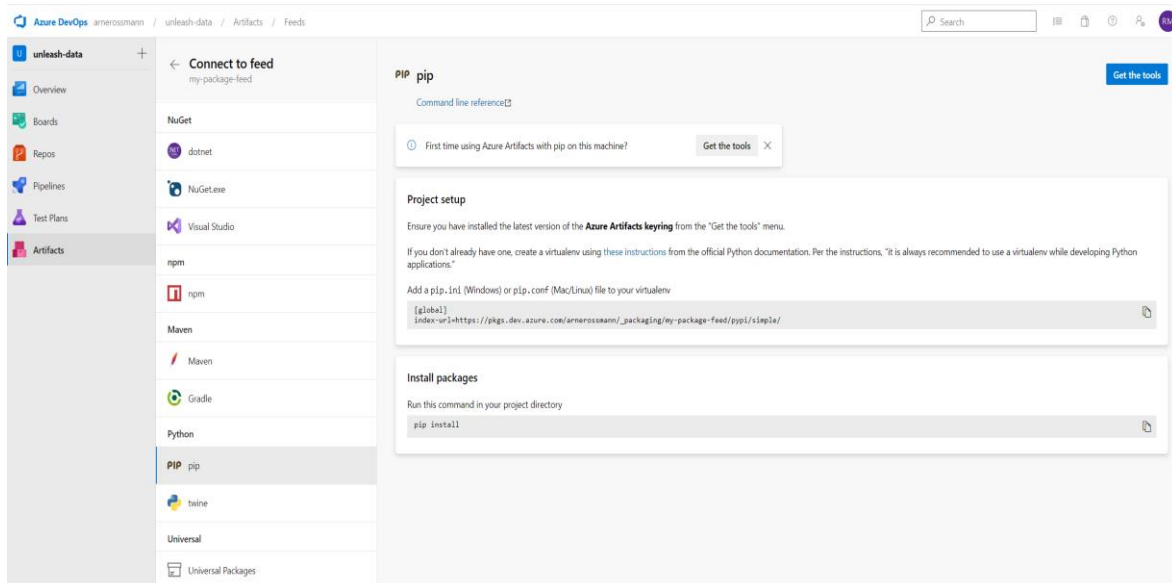
1. Install and use the artifacts-keyring package, which will automatically set up authentication for you.

OR

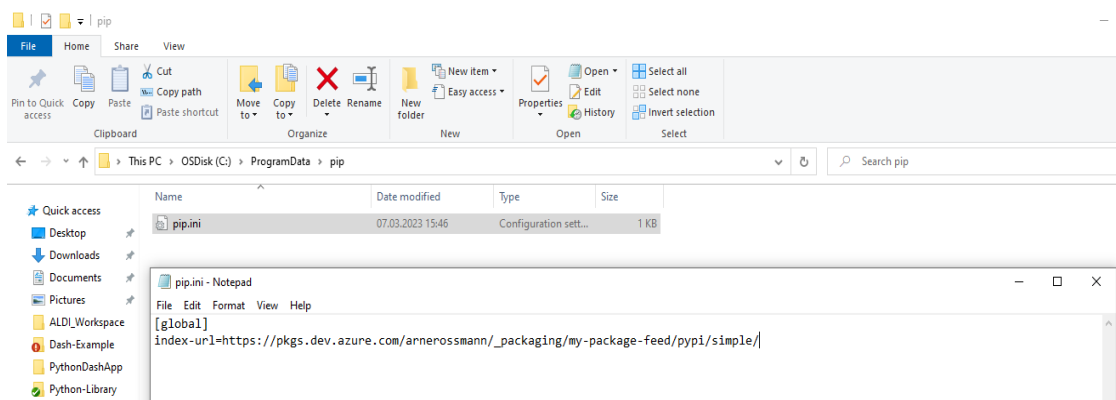
2. Manually set up credentials for *pip.ini/pip.conf*

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



- 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)



- 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)

```

Select Command Prompt
Microsoft Windows [Version 10.0.19044.2604]
(c) Microsoft Corporation. All rights reserved.

C:\Users\rohinmor>pip install myPackage==0.0.1
Looking in indexes: https://pkgs.dev.azure.com/arnerossmann/_packaging/my-package-feed/pypi/simple/
[Minimal] [CredentialProvider]DeviceFlow: https://pkgs.dev.azure.com/arnerossmann/_packaging/my-package-feed/pypi/simple/
[Minimal] [CredentialProvider]ATTENTION: User interaction required.

*****

To sign in, use a web browser to open the page https://microsoft.com/devicelogin and enter the code C8KT69JHP to aut
henticate.

*****

[Information] [CredentialProvider]VstsCredentialProvider - Acquired bearer token using 'ADAL Device Code'
[Information] [CredentialProvider]VstsCredentialProvider - Attempting to exchange the bearer token for an Azure DevOps s
ession token.
Collecting myPackage==0.0.1
  Downloading https://pkgs.dev.azure.com/arnerossmann/_packaging/6eef17f2-5f0a-4c64-bf0a-10368560d6eb/pypi/download/mypa
ckage/0.0.1/myPackage-0.0.1-py3-none-any.whl (1.5 kB)
Installing collected packages: myPackage
Successfully installed myPackage-0.0.1

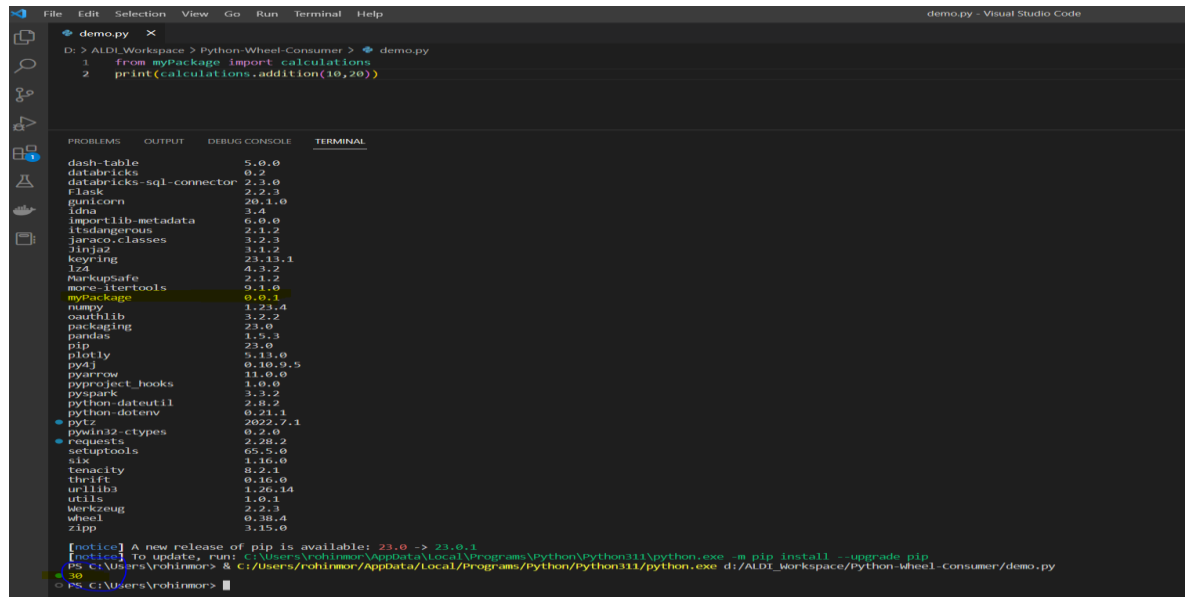
[notice] A new release of pip is available: 23.0 -> 23.0.1
[notice] To update, run: C:\Users\rohinmor\AppData\Local\Programs\Python\Python311\python.exe -m pip install --upgrade p
ip

C:\Users\rohinmor>

```


(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)



The screenshot shows the Visual Studio Code interface. The editor window displays a file named `demo.py` with the following content:

```
D:\> ALDI_Workspace > Python-Wheel-Consumer > demo.py
1 from myPackage import calculations
2 print(calculations.addition(10,20))
```

The terminal window at the bottom shows a list of installed packages and their versions:

Package	Version
dash-table	5.0.0
databricks	0.2
databricks-sql-connector	2.3.0
Flask	2.2.3
gunicorn	20.1.0
idna	3.4
importlib-metadata	6.0.0
itsdangerous	2.1.2
jaraco.classes	3.2.3
jinja2	3.1.2
keyring	23.13.1
lza	4.3.2
MarkupSafe	2.1.2
more-itertools	9.1.0
myPackage	0.0.1
numpy	1.23.4
oauthlib	3.2.2
packaging	23.0
pandas	1.5.3
pip	23.0
plotly	5.13.0
py4j	0.10.9.5
pyarrow	11.0.0
pyproject_hooks	1.0.0
pyspark	3.5.2
python-dateutil	2.8.2
python-dotenv	0.21.1
pytz	2022.7.1
pywin32-ctypes	0.2.0
requests	2.28.2
setuptools	65.5.0
six	1.16.0
tenacity	8.2.1
thrift	0.16.0
urllib3	1.26.14
utils	1.0.1
werkzeug	2.2.3
wheel	0.38.4
zipp	3.15.0

At the bottom of the terminal, a notice states: "[notice] A new release of pip is available: 23.0 -> 23.0.1". Below this, a command is shown: "C:\Users\rohinimor> & C:\Users\rohinimor\AppData\Local\Programs\python\python311\python.exe d:\ALDI_workspace\python-wheel-consumer\demo.py".

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>