Azure MLops: Azure Machine Learning + Azure Devops.

Azure Machine Learning Studio:

- A collaborative tool for the team members to work with.
- **pip freeze** → This shows the list of installed packages.
- Activities are automatically logged under Azure Machine Learning Platform.
- **Auto-ML:** Automated machine learning is the process of automating the tasks of applying machine learning to real-world problems. Auto-ML potentially includes every stage from beginning with a raw dataset to building a machine learning model ready for deployment.

Services of importance: Azure Devops.

Go to Azure Devops Service →

- Create new Project.
- The Repos folder is like a GitHub containing relevant codes for the project, user can work directly over it and do updates accordingly.
- Why is a workspace needed?
  ➔ Workspace helps to collaborate when working over a project.

Project:

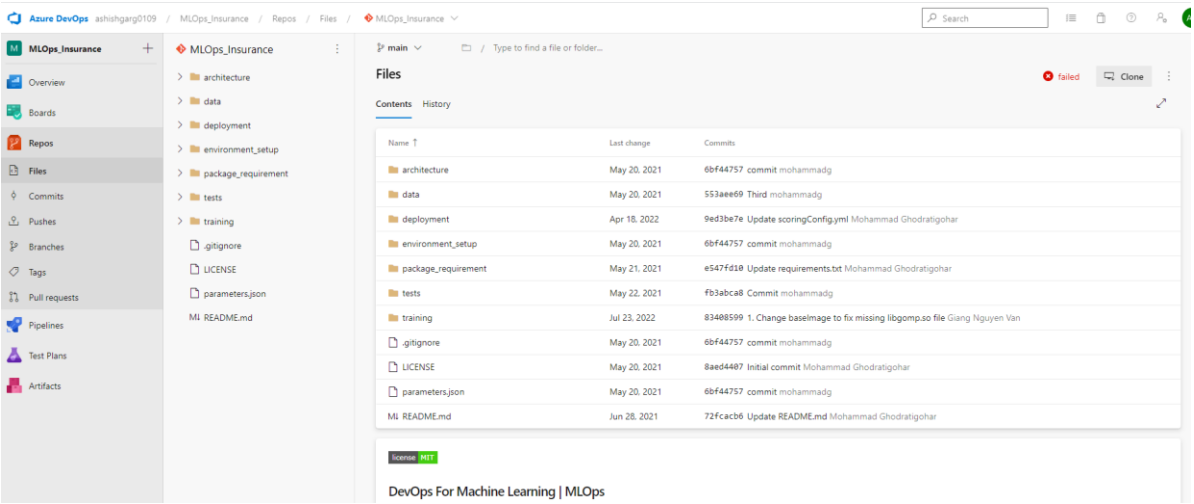
- Make new Project.
- **Make a service connection with your Azure Subscription**, by going under project settings.
- We will write the code to setup the infrastructure for Azure Machine Learning Resources.
  Go to project → Pipelines → Library → Specify variable group name.
  Define variables under group like:
  BASE_NAME: mlopsash1(This will be the name of azure resource this is creating Ex:
  WORKSPACE_NAME: mlops-ash-aml1
  AZURE_RM_SVC_CONNECTION: azure-resource-connection (This is the name of service connection created earlier).

Variables

Name ↑	Value
AZURE_RM_SVC_CONNECTION	azure-resource-connection
BASE_NAME	mlopswsh1
LOCATION	centralus
RESOURCE_GROUP	mlops-wsh-rg1
WORKSPACE_NAME	mlops-wsh-aml1
WORKSPACE_SVC_CONNECTION	aml-workspace-connection

Under Azure Devops:

All the relevant code is in one place under Repos Folder.



Set the yaml pipeline configuration as shown below to create infrastructure under azure portal (Infrastructure as a code):

```yaml
pr: none
trigger: none

variables:
- group: mlops-ash-vg

stages:
- stage: 'Dev'
  displayName: 'Dev'
  jobs:
  - job: "Provision_Dev"
    displayName: "Provision Dev resources"
    pool:
      vmImage: 'ubuntu-latest'
    timeoutInMinutes: 0
    steps:
      Settings
    - task: AzureResourceGroupDeployment@2
      inputs:
        azureSubscription: '$(AZURE_RM_SVC_CONNECTION)'
        action: 'Create Or Update Resource Group'
        resourceGroupName: '$(RESOURCE_GROUP)'
        location: $(LOCATION)
        templateLocation: 'Linked artifact'
        csmFile: '$(Build.SourcesDirectory)/environment_setup/cloud-environment.json'
        overrideParameters: '-baseName $(BASE_NAME) -location $(LOCATION) -workspace $(WORKSPACE_NAME)'
        deploymentMode: 'Incremental'
      displayName: 'Deploy OH resources to Azure'
```

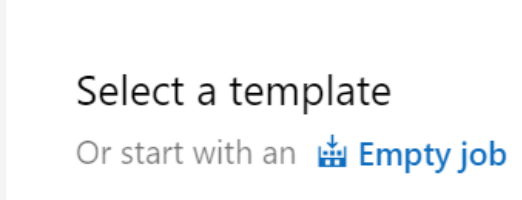
Run the pipeline and azure resources will be created as below:

mlops-ash-aml1	Azure Machine Learning workspace	Central US
mlopsash1-AML-AI	Application Insights	Central US
mlopsash1-AML-KV	Key vault	Central US
mlopsash1amlcr	Container registry	Central US
mlopsash1amlsa	Storage account	Central US

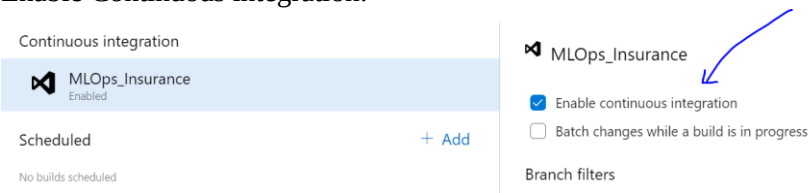
- Data Store contains the different endpoints towards data points.

Suppose if a data-scientist did some manual change, then that change needs to trigger the pipeline and create the new model.

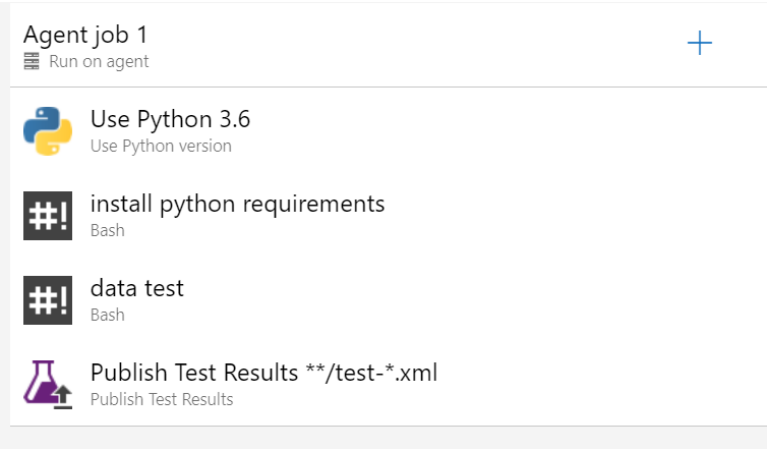
- Create a separate CI pipe for it. Use classic editor this time.
- Create an empty job in here:



- Enable Continuous integration:



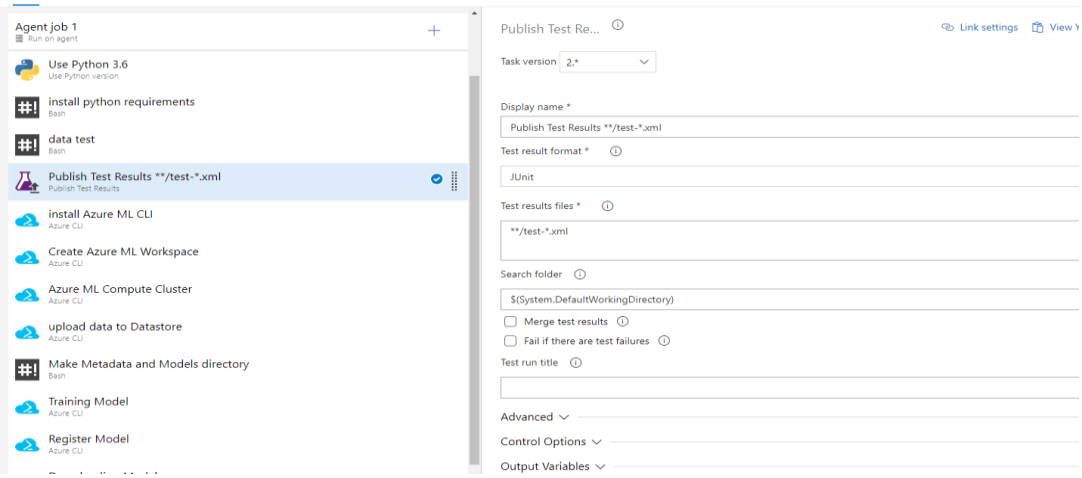
- Steps can be configured in pipeline something like this:



For tests: Use command: pytest training/train_test.py --doctest-modules --junitxml=junit/test-results.xml --cov=data_test --cov-report=xml --cov-report=html. Under data test inline option in order to write script command.

This command will also generate the report of executed tests in form of html.

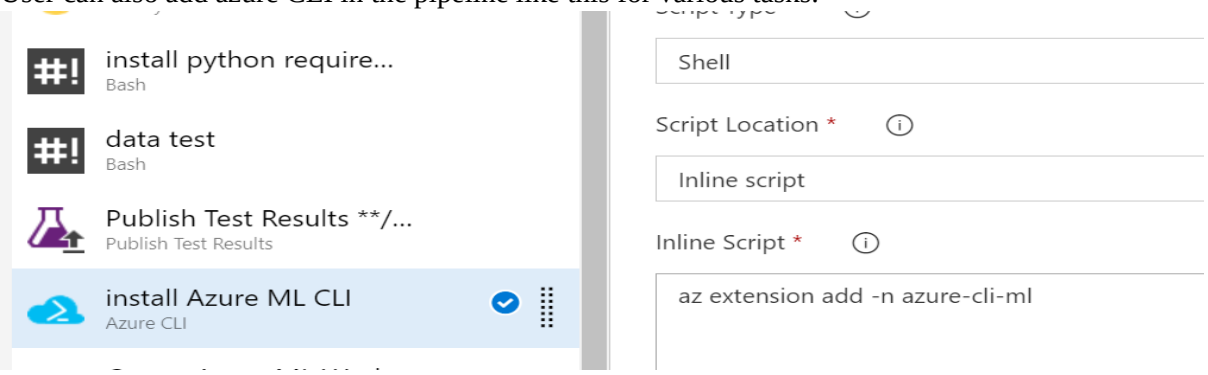
Visualization of test results are done like this:



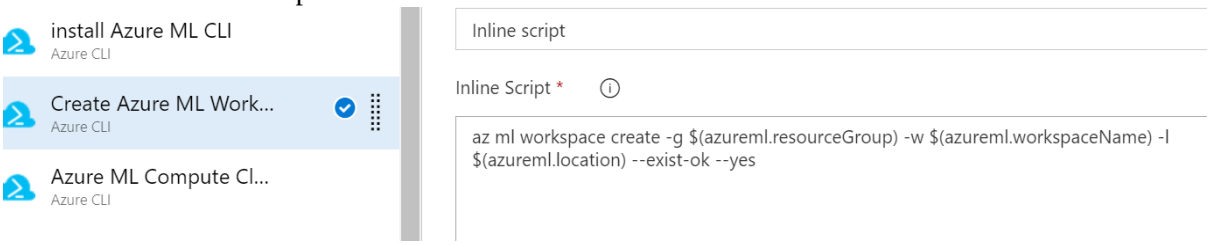
- Azure also has CLI to interact with. Install it in your local computer and excess it through terminal. Command for azure cli starts with az.
  Specific extensions can be installed like this through terminal.

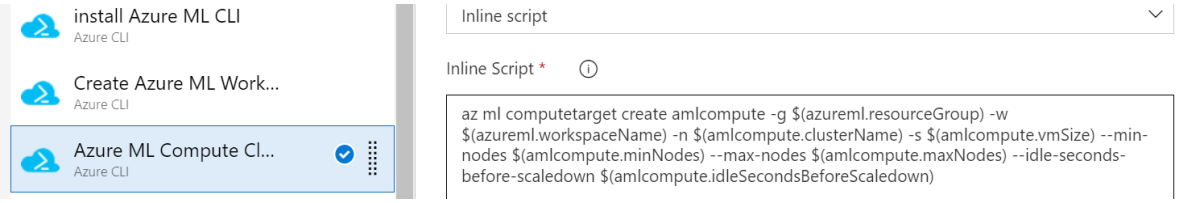
```
PS C:\Users\asgarg> az extension add -n azure-cli-ml
```

- User can also add azure CLI in the pipeline like this for various tasks:

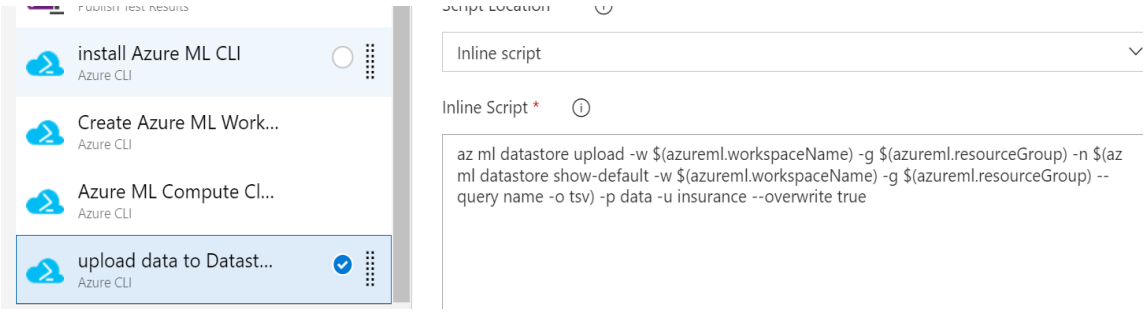


- Create Azure ML Workspace.

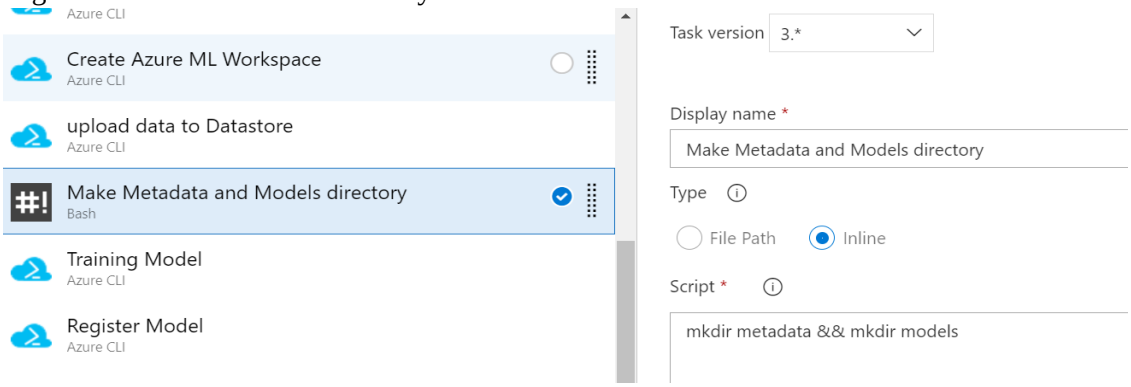




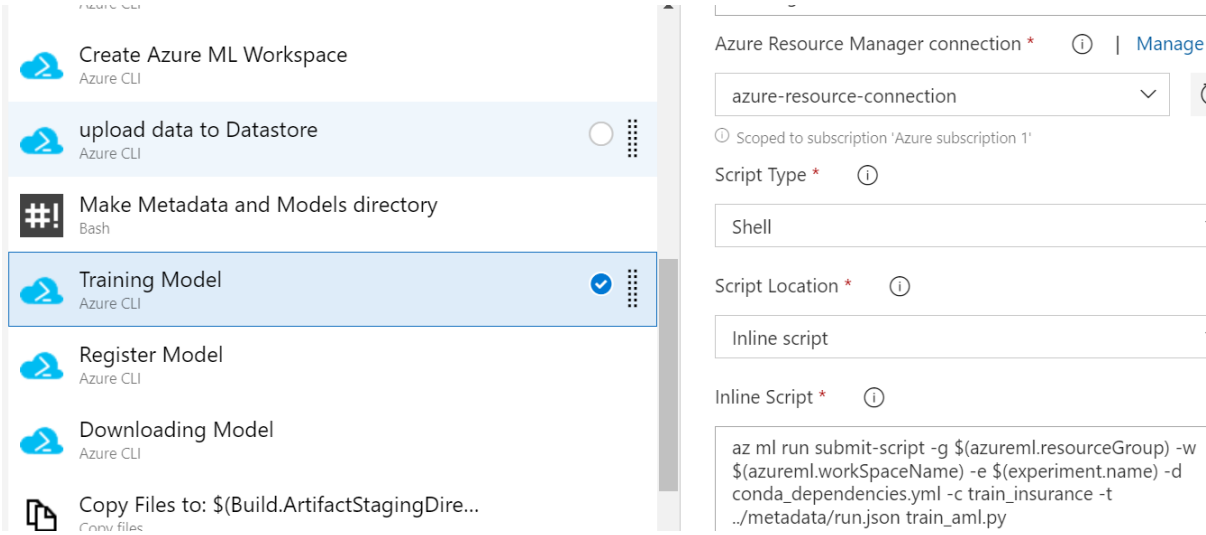
- Upload Data to datastore(A process to automatically update data, if new data comes in).
  Command: az ml datastore upload -w $(azureml.workspaceName) -g $(azureml.resourceGroup) -n $(az ml datastore show-default -w $(azureml.workspaceName) -g $(azureml.resourceGroup) --query name -o tsv) -p data -u insurance --overwrite true



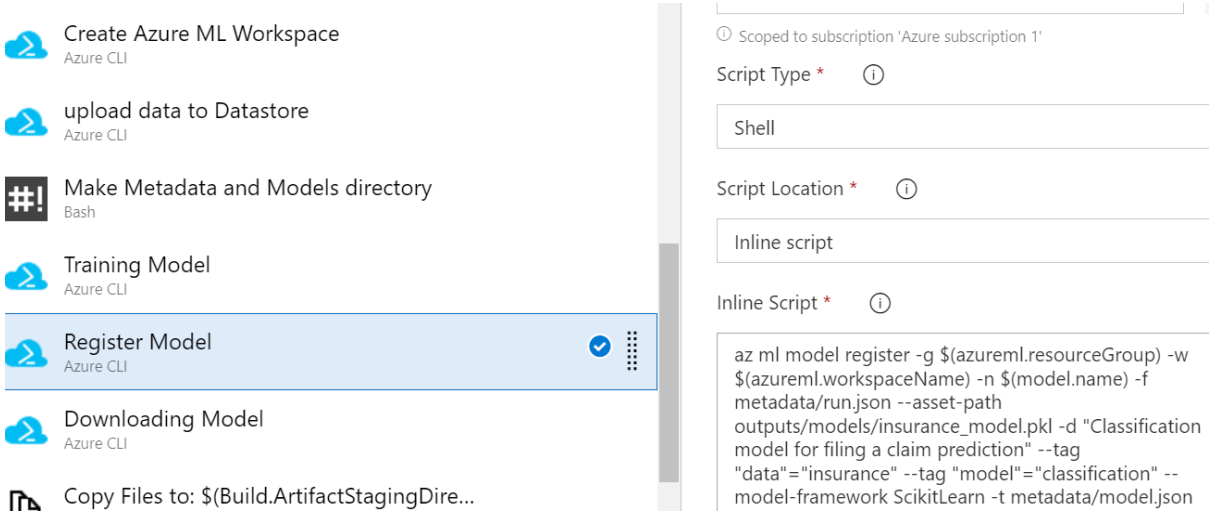
- Making Meta Data and models directory.



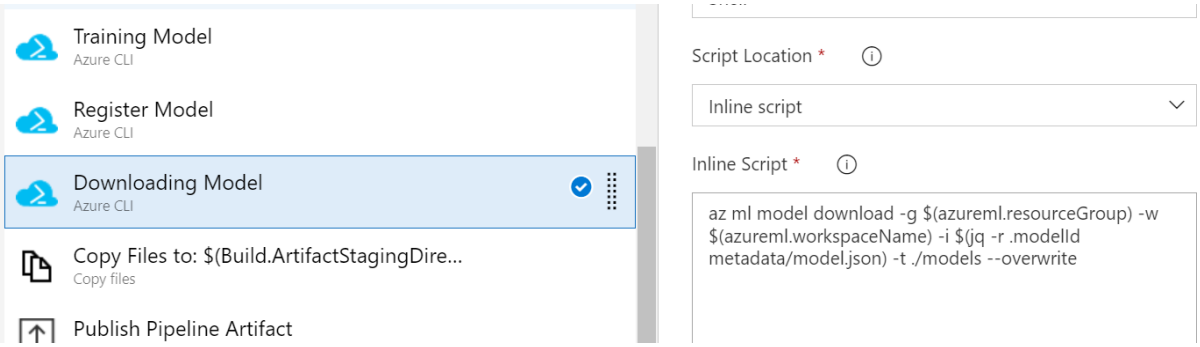
- Training the model:
  Command: az ml run submit-script -g $(azureml.resourceGroup) -w $(azureml.workSpaceName) -e $(experiment.name) -d conda_dependencies.yml -c train_insurance -t ../metadata/run.json train_aml.py



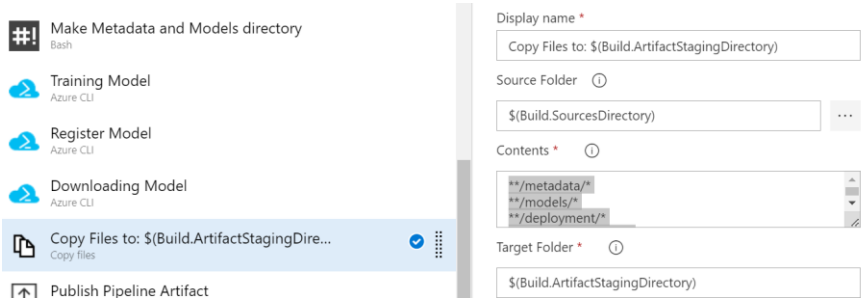
- Register the trained model:
  az ml model register -g $(azureml.resourceGroup) -w $(azureml.workspaceName) -n $(model.name) -f metadata/run.json --asset-path outputs/models/insurance_model.pkl -d "Classification model for filing a claim prediction" --tag "data"="insurance" --tag "model"="classification" --model-framework ScikitLearn -t metadata/model.json



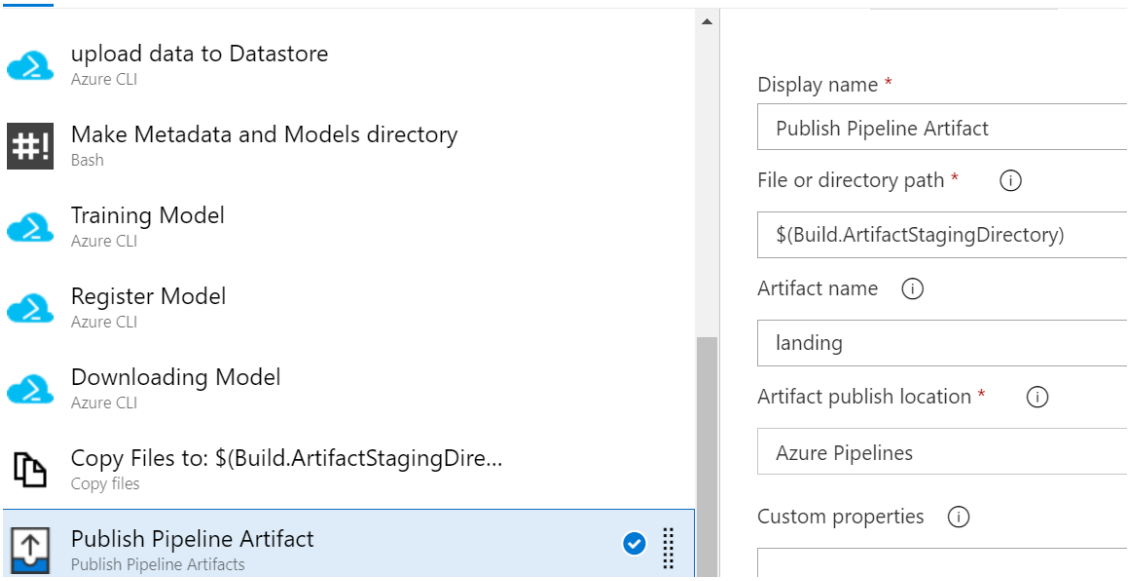
- Download the trained model:



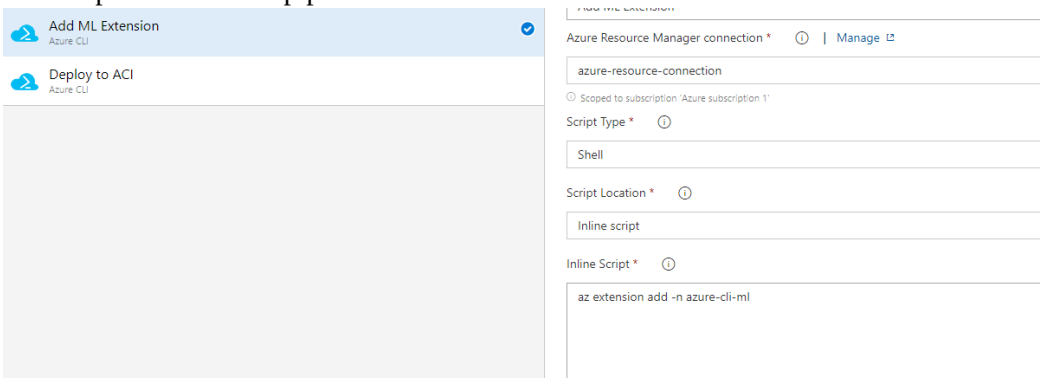
- Copy the content within relevant directories for staging:
  Directories:
  **/metadata/*
  **/models/*
  **/deployment/*
  **/tests/integration/*
  **/package_requirement/*



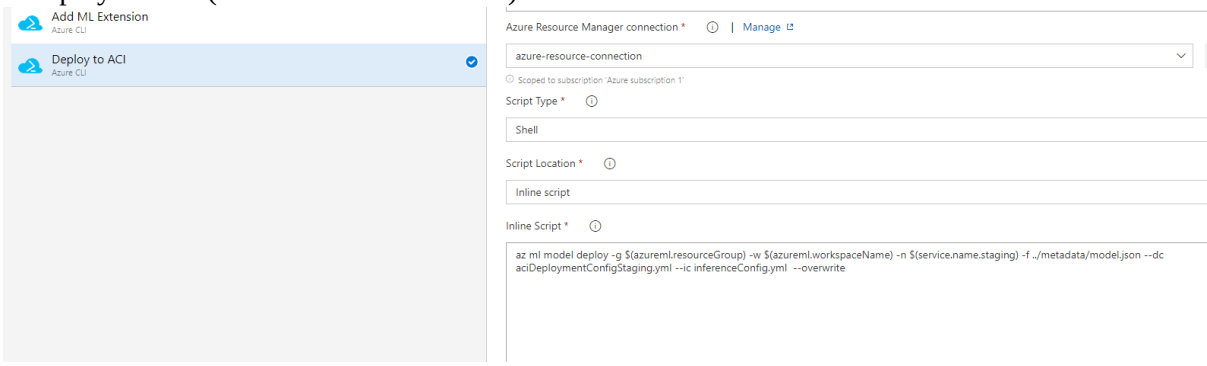
- Publish the artifact that can be used under release.



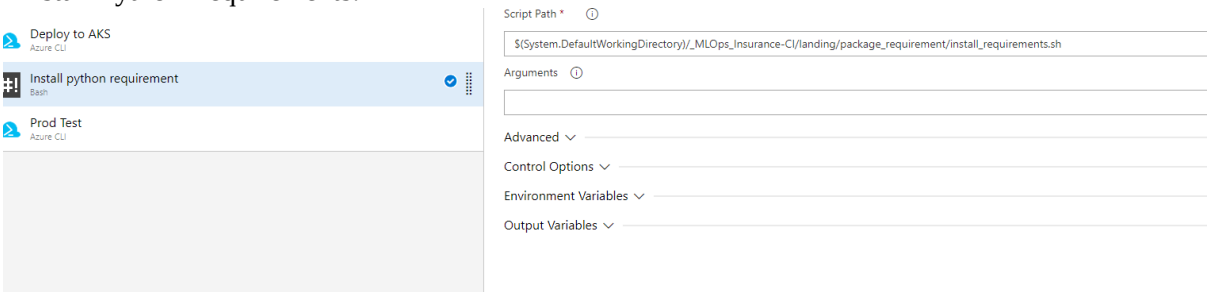
- After Publishing the artifact, go to the release pipeline.
  - ➔ Create a new pipeline with MLOpsInsurance_CI as an source artifact.
  - ➔ Add respective tasks in pipeline like:



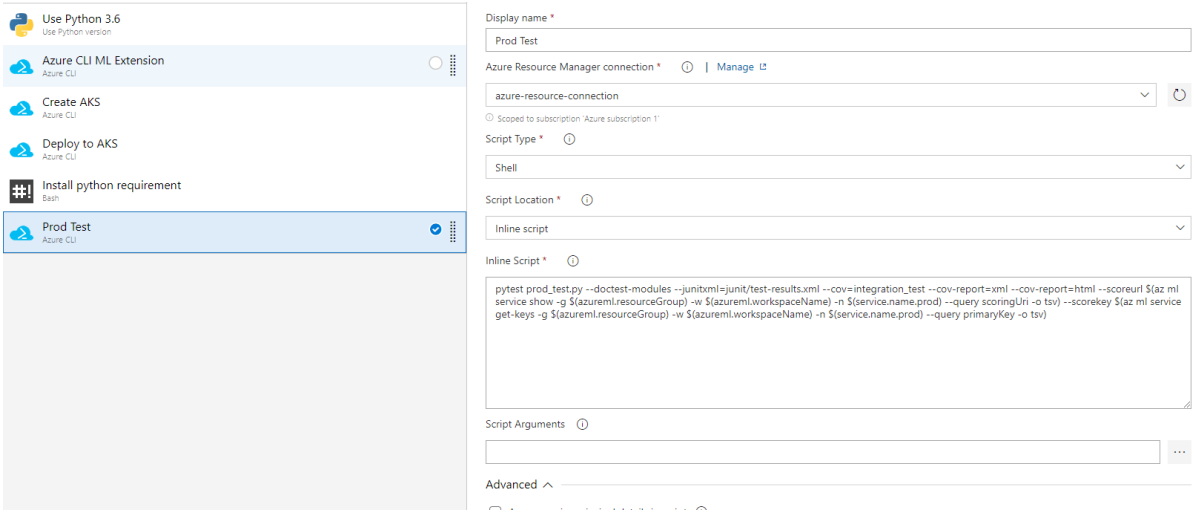
* Deploy to ACI (Azure container instance).



*Install Python Requirements.



*Production test:



- A user can define triggers in the pipeline i.e if some changes happened, pipeline will be triggered automatically.