

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:

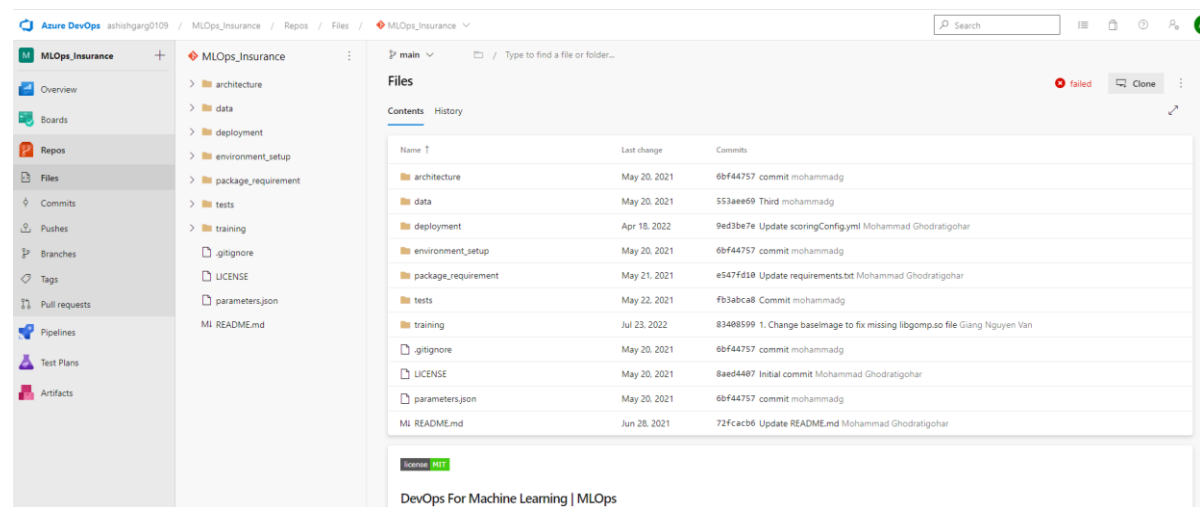
- Make new Project.
- **Make a service connection with your Azure Subscription**, by going under project settings.
- We will write the code to setup the infrastructure for Azure Machine Learning Resources. Go to project → Pipelines → Library → Specify variable group name. Define variables under group 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):

```
1 pr: none
2 trigger: none
3
4 variables:
5   - group: mlops-ash-vg
6
7 stages:
8   - stage: 'Dev'
9     displayName: 'Dev'
10    jobs:
11      - job: 'Provision_Dev'
12        displayName: 'Provision Dev resources'
13        pool:
14          vmImage: 'ubuntu-latest'
15          timeoutInMinutes: 0
16        steps:
17          - task: AzureResourceGroupDeployment@2
18            inputs:
19              azureSubscription: '$(AZURE_RM_SVC_CONNECTION)'
20              action: 'Create Or Update Resource Group'
21              resourceGroupName: '$(RESOURCE_GROUP)'
22              location: $(LOCATION)
23              templateLocation: 'Linked artifact'
24              csmFile: '$(Build.SourcesDirectory)/environment_setup/cloud-environment.json'
25              overrideParameters: '-baseName $(BASE_NAME) -location $(LOCATION) -workspace $(WORKSPACE_NAME)'
26              deploymentMode: 'Incremental'
27              displayName: 'Deploy OH resources to Azure'
28
```

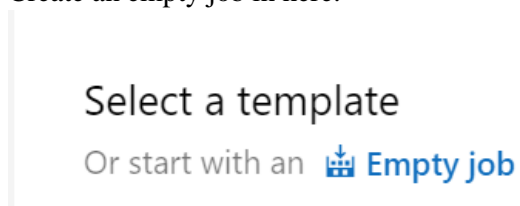
Run the pipeline and azure resources will be created as below:

<input type="checkbox"/>	 mlops-ash-aml1	Azure Machine Learning workspace	Central US
<input type="checkbox"/>	 mlopsash1-AML-AI	Application Insights	Central US
<input type="checkbox"/>	 mlopsash1-AML-KV	Key vault	Central US
<input type="checkbox"/>	 mlopsash1amlcr	Container registry	Central US
<input type="checkbox"/>	 mlopsash1amlsa	Storage account	Central US

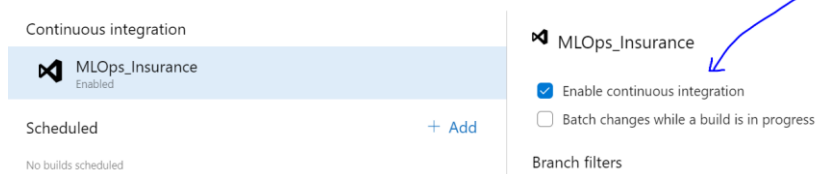
- Data Store contains the different endpoints towards data points.

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

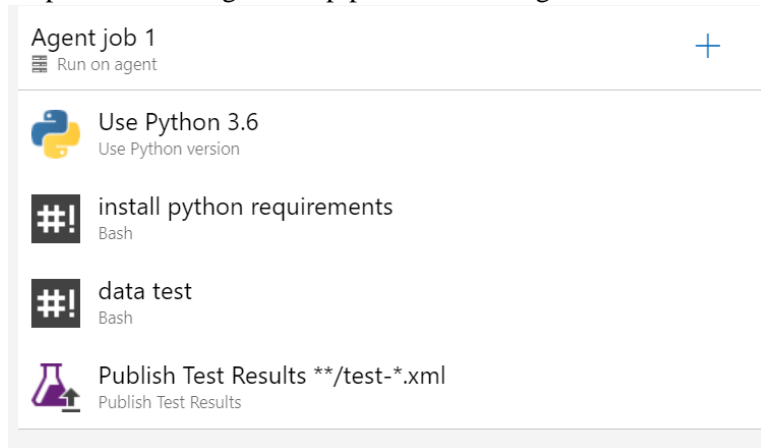
- Create a separate CI pipe for it. Use classic editor this time.
- Create an empty job in here:



- Enable Continuous integration:



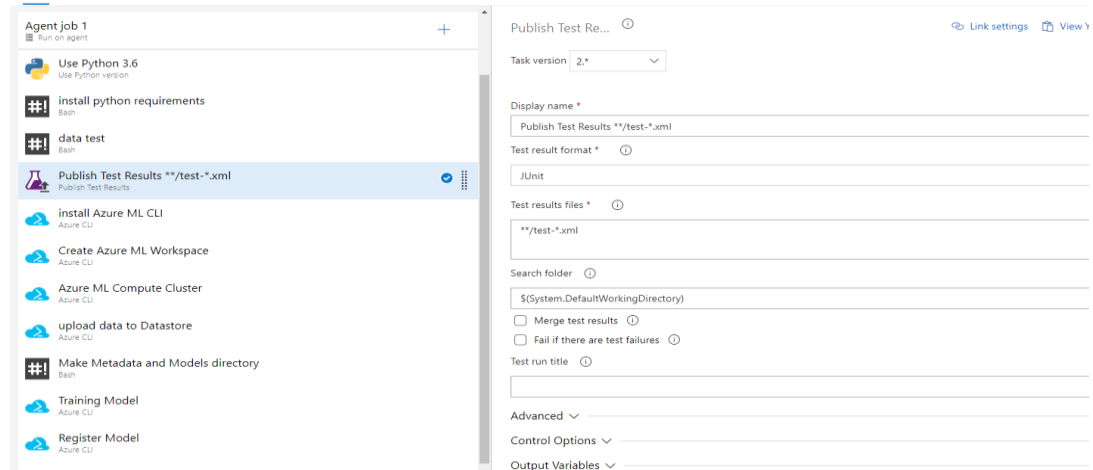
- Steps can be configured in pipeline something like this:



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.

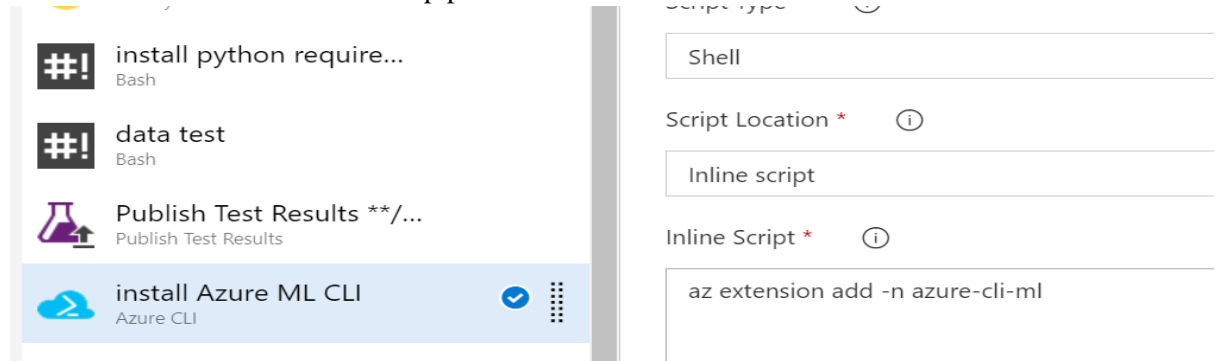
Visualization of test results are done like this:



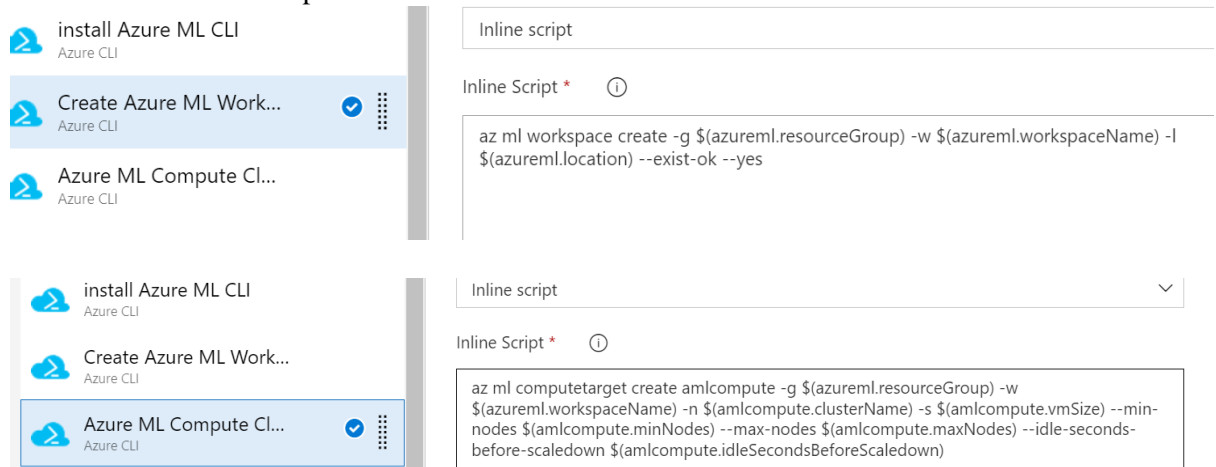
- Azure also has CLI to interact with. Install it in your local computer and excess it through terminal. Command for azure cli starts with az. Specific extensions can be installed like this through terminal.

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

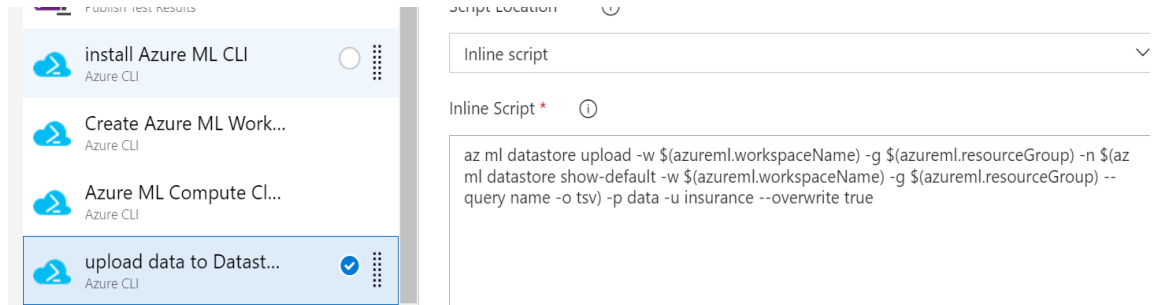
- User can also add azure CLI in the pipeline like this for various tasks:



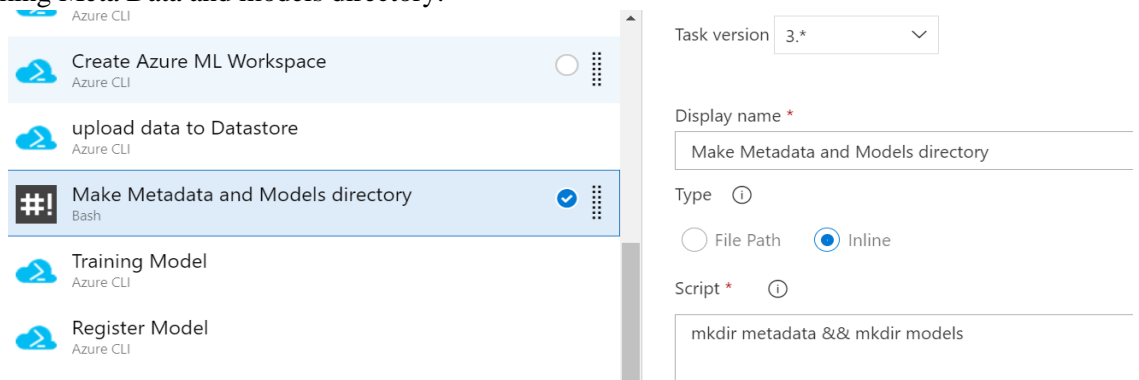
- Create Azure ML Workspace.



- 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`

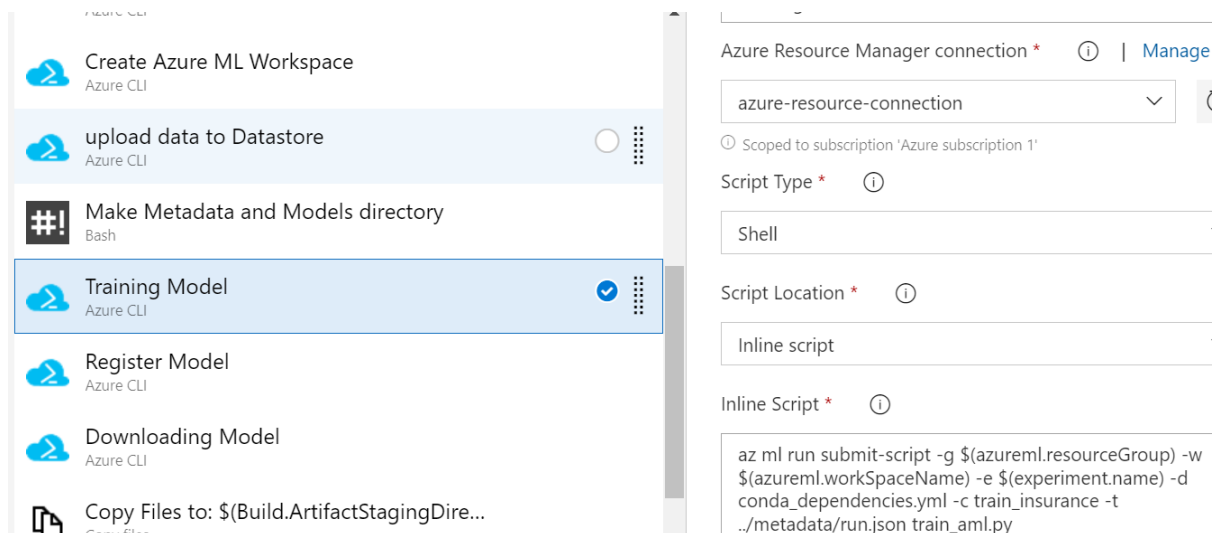


- Making Meta Data and models directory.



- 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
```

The screenshot shows a sequence of steps in an Azure ML pipeline:

- Create Azure ML Workspace (Azure CLI)
- upload data to Datastore (Azure CLI)
- Make Metadata and Models directory (Bash)
- Training Model (Azure CLI)
- Register Model (Azure CLI)** - This step is highlighted with a blue bar and a checkmark.
- Downloading Model (Azure CLI)
- Copy Files to: \$(Build.ArtifactStagingDire... (Copy files)

Scoped to subscription 'Azure subscription 1'

Script Type * ⓘ

Shell

Script Location * ⓘ

Inline script

Inline Script * ⓘ

```
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:

The screenshot shows the same sequence of steps as before, but with 'Downloading Model' highlighted:

- Training Model (Azure CLI)
- Register Model (Azure CLI)
- Downloading Model (Azure CLI)** - This step is highlighted with a blue bar and a checkmark.
- Copy Files to: \$(Build.ArtifactStagingDire... (Copy files)
- Publish Pipeline Artifact

Script Location * ⓘ

Inline script

Inline Script * ⓘ

```
az ml model download -g $(azureml.resourceGroup) -w
$(azureml.workspaceName) -i $(jq -r .modelId
metadata/model.json) -t ./models --overwrite
```

- Copy the content within relevant directories for staging:

Directories:

```
**/metadata/*
**/models/*
**/deployment/*
**/tests/integration/*
**/package_requirement/*
```

The screenshot shows the pipeline steps with 'Copy Files to: \$(Build.ArtifactStagingDirectory)' highlighted:

- Make Metadata and Models directory (Bash)
- Training Model (Azure CLI)
- Register Model (Azure CLI)
- Downloading Model (Azure CLI)
- Copy Files to: \$(Build.ArtifactStagingDirectory) (Copy files)** - This step is highlighted with a blue bar and a checkmark.
- Publish Pipeline Artifact

Display name *

Copy Files to: \$(Build.ArtifactStagingDirectory)

Source Folder ⓘ

\$(Build.SourcesDirectory)

Contents * ⓘ

**/metadata/*
 **/models/*
 **/deployment/*

Target Folder * ⓘ

\$(Build.ArtifactStagingDirectory)

- Publish the artifact that can be used under release.

upload data to Datastore
Azure CLI

Make Metadata and Models directory
Bash

Training Model
Azure CLI

Register Model
Azure CLI

Downloading Model
Azure CLI

Copy Files to: \$(Build.ArtifactStagingDire...
Copy files

Publish Pipeline Artifact
Publish Pipeline Artifacts

Display name *

Publish Pipeline Artifact

File or directory path * ⓘ

\$(Build.ArtifactStagingDirectory)

Artifact name ⓘ

landing

Artifact publish location * ⓘ

Azure Pipelines

Custom properties ⓘ

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

Add ML Extension
Azure CLI

Deploy to ACI
Azure CLI

Azure Resource Manager connection * ⓘ | Manage

azure-resource-connection

ⓘ Scoped to subscription 'Azure subscription 1'

Script Type * ⓘ

Shell

Script Location * ⓘ

Inline script

Inline Script * ⓘ

az extension add -n azure-cli-ml

* Deploy to ACI (Azure container instance).

Add ML Extension
Azure CLI

Deploy to ACI
Azure CLI

Azure Resource Manager connection * ⓘ | Manage

azure-resource-connection

ⓘ Scoped to subscription 'Azure subscription 1'

Script Type * ⓘ

Shell

Script Location * ⓘ

Inline script

Inline Script * ⓘ

az ml model deploy -g \$(azureml.resourceGroup) -w \$(azureml.workspaceName) -n \$(service.name.staging) -f ./metadata/model.json --dc aciDeploymentConfigStaging.yml --ic inferenceConfig.yml --overwrite

*Install Python Requirements.

Deploy to AKS
Azure CLI

Install python requirement
Bash

Prod Test
Azure CLI

Script Path * ⓘ

\$System.DefaultWorkingDirectory/_MLOps_Insurance-CI/landing/package_requirement/install_requirements.sh

Arguments ⓘ

Advanced

Control Options

Environment Variables

Output Variables

*Production test:

The screenshot shows the configuration for a pipeline job named 'Prod Test'. On the left, a list of tasks is shown, with 'Prod Test' selected. The main area on the right contains the configuration for this task.

Task List (Left):

- Use Python 3.6 (Use Python version)
- Azure CLI ML Extension (Azure CLI)
- Create AKS (Azure CLI)
- Deploy to AKS (Azure CLI)
- Install python requirement (Bash)
- Prod Test (Azure CLI)**

Task Configuration (Right):

- Display name ***: Prod Test
- Azure Resource Manager connection ***: azure-resource-connection (Manage)
- Script Type ***: Shell
- Script Location ***: Inline script
- Inline Script ***:

```
pytest prod_test.py --doctest-modules --junitxml=junit/test-results.xml --cov=integration_test --cov-report=yml --cov-report=html --scoreurl $(az ml service show -g $(azureml.resourceGroup) -w $(azureml.workspaceName) -n $(service.name.prod) --query scoringUri -o tsv) --scorekey $(az ml service get-keys -g $(azureml.resourceGroup) -w $(azureml.workspaceName) -n $(service.name.prod) --query primaryKey -o tsv)
```
- Script Arguments**: (Empty field)
- Advanced**: (Expanded section with various options)

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