

Contents

Problem Statement.....	1
Capabilities of the Pipeline	2
Setting Up Azure Machine Learning Workspace.....	2
Setting Up Azure DevOps:.....	3
Create an azure devops organization	3
If you already have an organization then create a new project.....	4
Install azure machine learning extension to azure devops organization, by clicking “get it free”.	4
Get the code by using the <i>repository template</i>	4
Create a variable group for your pipeline in azure devops.	5
Create an Azure DevOps Service Connection for the Azure Resource Manager	8
Create the IaC pipeline	10
Create an Azure DevOps Service Connection for the Azure ML Workspace.....	13
Set up Build, Release Trigger, and Release Multi-Stage Pipelines.....	14
Set up the Model CI, training, evaluation, and registration pipeline	15
Set up the Release Deployment and/or Batch Scoring pipelines.....	19
Set up the Release Deployment pipeline	20
Set up the Batch Scoring pipeline.....	21
Data Drift Monitoring script	21

POC: MLOPS

Problem Statement

Calculate the progression of the diabetes of 442 patients, given 10 attributes that contribute to the predictions.

Ten baseline variables, age, sex, body mass index, average blood pressure, and six blood serum measurements were obtained for each of $n = 442$ diabetes patients, as well as the response of interest, a quantitative measure of disease progression one year after baseline

It is a linear regression problem.

Each of these 10 feature variables have been mean centered and scaled by the standard deviation times the square root of $n_samples$ (i.e., the sum of squares of each column totals 1).

Data Set Characteristics:

Number of Instances:	442
Number of Attributes:	First 10 columns are numeric predictive values
Target:	Column 11 is a quantitative measure of disease progression one year after baseline
Attribute Information:	<ul style="list-style-type: none"> • age age in years • sex • bmi body mass index • bp average blood pressure • s1 tc, total serum cholesterol • s2 ldl, low-density lipoproteins • s3 hdl, high-density lipoproteins • s4 tch, total cholesterol / HDL • s5 ltg, possibly log of serum triglycerides level • s6 glu, blood sugar level

Capabilities of the Pipeline

This pipeline contains steps to

- Convert the experimentation code to the azure-ml production code
- Creates and manages the azure-ml connection with azure devops.

This pipeline will:

trigger CI (continuous Integration) on each code commit, to validate the code for code quality, and runs unit tests and linting tests, and will trigger the CD (Continuous Deployment) to train, score, evaluate and register the model in .pkl format to the azure storage account.

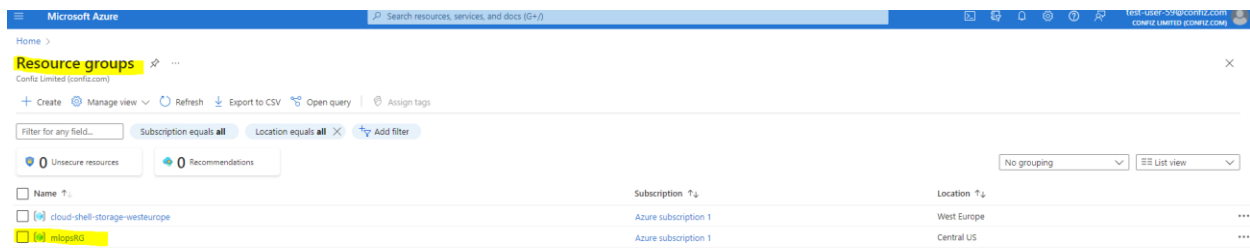
This pipeline does the Batch Scoring over a given set of batches and produces the results and saves them into the prediction.csv file into the azure-ml storage account.

Setting Up Azure Machine Learning Workspace

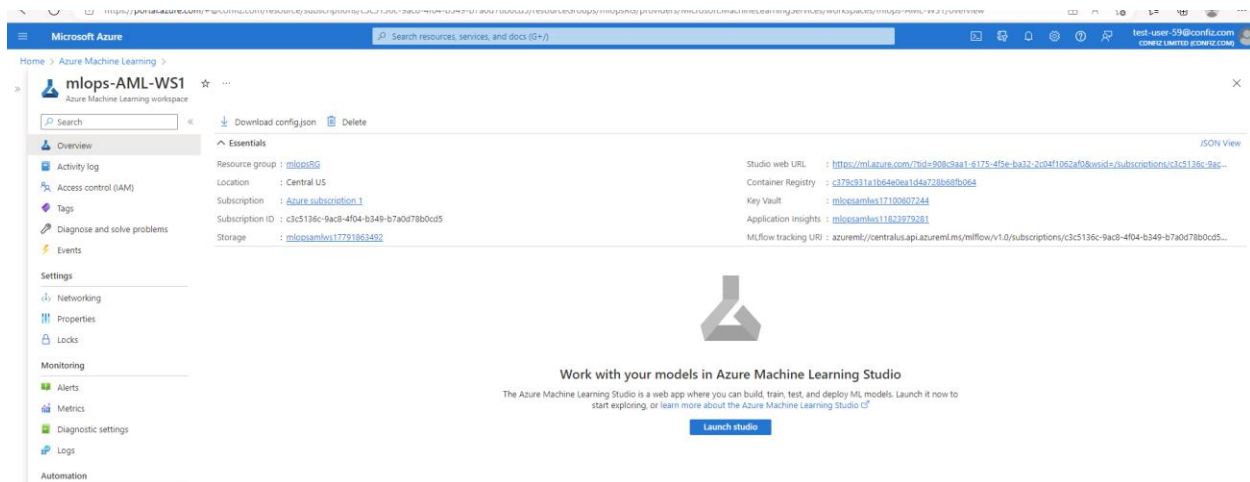
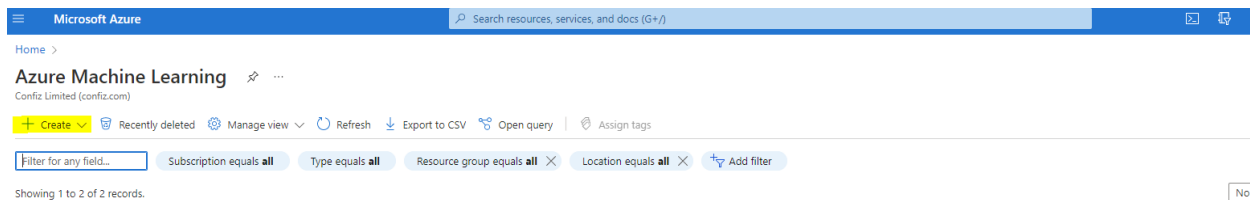
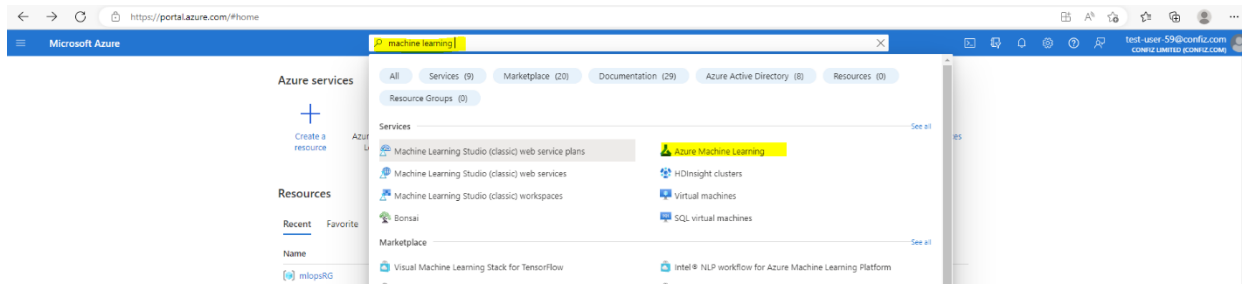
Open the Azure: <https://portal.azure.com>

Register for a subscription of a free account, create your free account and access the portal home page.

Create a resource group named “mlopsRG”, with this subscription, with the highlighted icon on the screen.



Create an “Azure Machine Learning Workspace” the same way named “mlops-AML-WS1”.

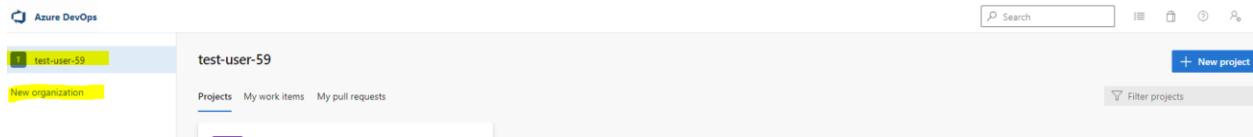


Setting Up Azure DevOps:

[MLOpsPython/getting_started.md at master · microsoft/MLOpsPython \(github.com\)](https://github.com/microsoft/MLOpsPython/blob/master/MLOpsPython/getting_started.md)

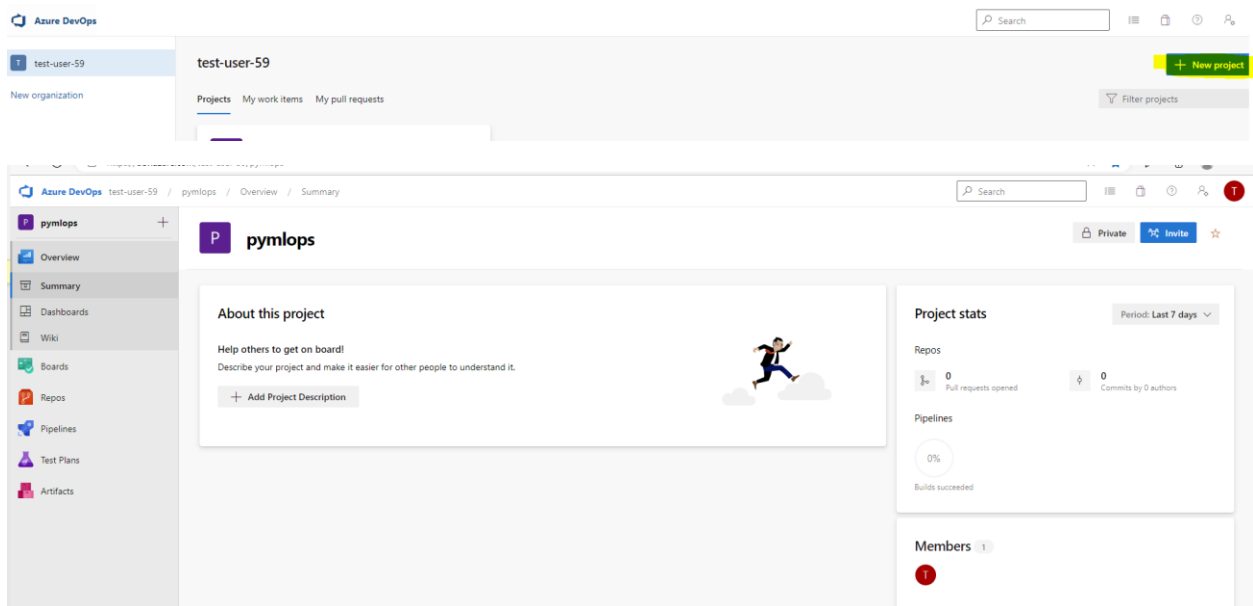
Create an azure devops organization

<https://docs.microsoft.com/en-us/azure/devops/organizations/accounts/create-organization?view=azure-devops>

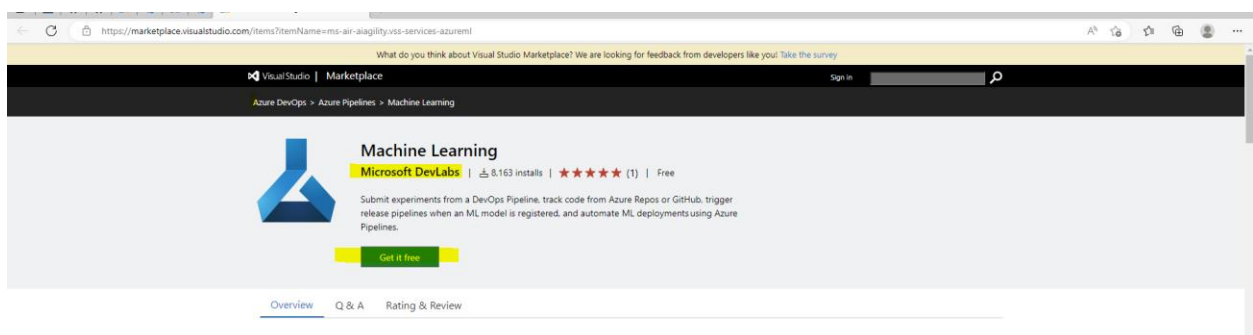


If you already have an organization then create a new project

<https://docs.microsoft.com/en-us/azure/devops/organizations/projects/create-project?view=azure-devops>



Install azure machine learning extension to azure devops organization, by clicking “get it free”.



<https://marketplace.visualstudio.com/items?itemName=ms-air-aiagility.vss-services-azureml>

Get the code by using the *repository template*

<https://github.com/microsoft/MLOpsPython/generate>

Create a new repository from MLOpsPython

The new repository will start with the same files and folders as `microsoft/MLOpsPython`.

Owner * `HiraNisarWarsi` / Repository name *

Great repository names are short and memorable. Need inspiration? How about `psychic-guide`?

Description (optional)

☒ Public
Anyone on the internet can see this repository. You choose who can commit.

☐ Private
You choose who can see and commit to this repository.

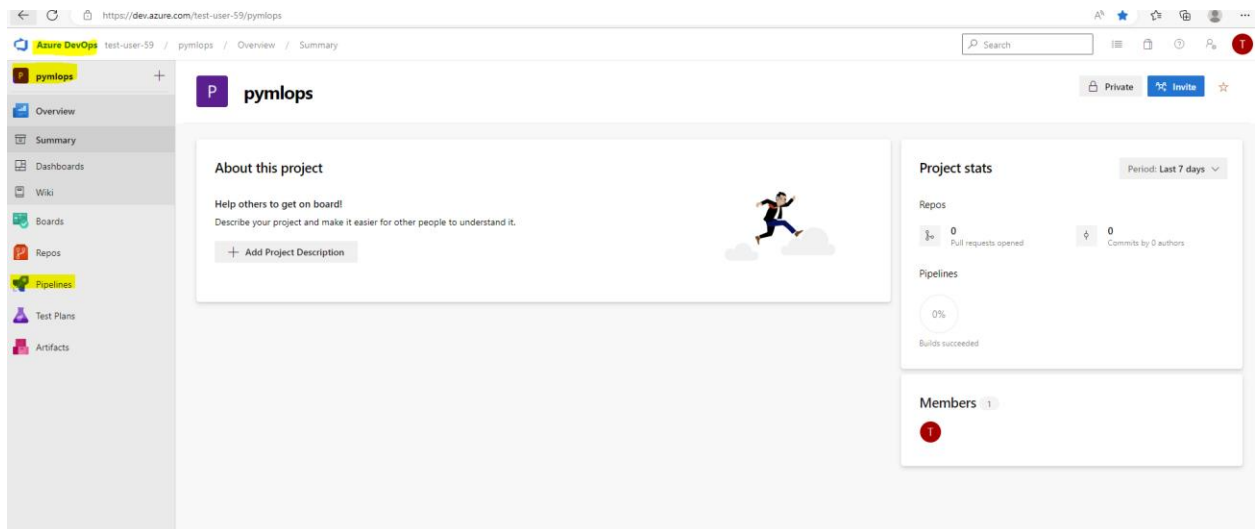
☐ Include all branches
Copy all branches from `microsoft/MLOpsPython` and not just master.

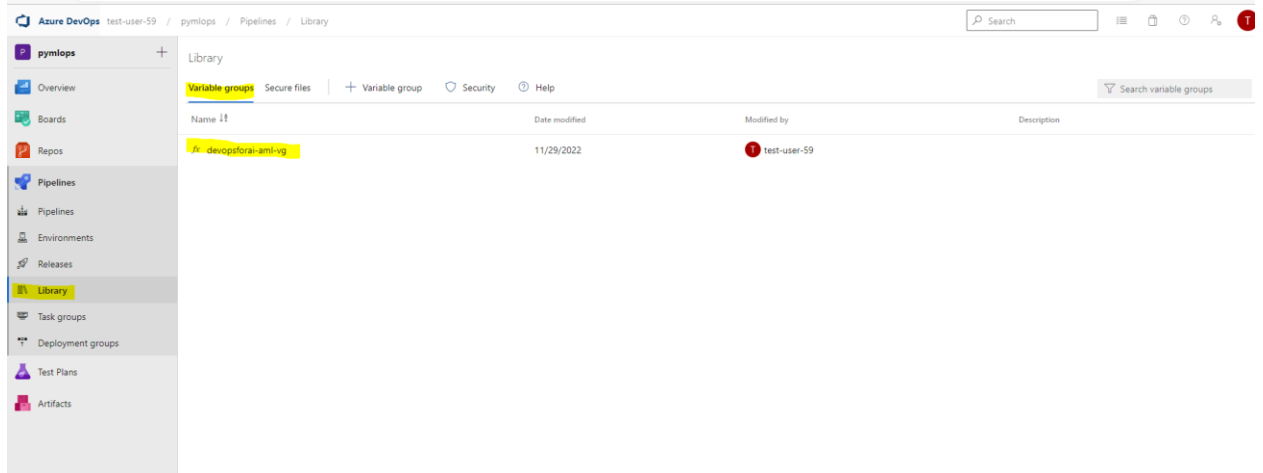
☐ You are creating a public repository in your personal account.

Create repository from template

Create a variable group for your pipeline in azure devops.

- In azure devops, navigate to **library** in the **pipelines** section, and create a variable group named **“devopsforai-aml-vg”**. The YAML pipeline definitions in this repository refer to this variable group by name.





- The variable group should contain the following required variables.

Variable Name	Suggested Value	Short description
BASE_NAME	[your project name]	Unique naming prefix for created resources - max 10 chars, letters and numbers only
LOCATION	centralus	Azure location , no spaces. You can list all the region codes by running <code>az account list-locations -o table</code> in the Azure CLI
RESOURCE_GROUP	mlops-RG	Azure Resource Group name
WORKSPACE_NAME	mlops-AML-WS	Azure ML Workspace name
AZURE_RM_SVC_CONNECTION	azure-resource-connection	Azure Resource Manager Service Connection name
WORKSPACE_SVC_CONNECTION	aml-workspace-connection	Azure ML Workspace Service Connection name
ACI_DEPLOYMENT_NAME	mlops-aci	Azure Container Instances name

Library > devopstorai-aml-vg

Variable group | Save | Clone | Security | Pipeline permissions | Approvals and checks | Help

Properties

Variable group name

Description

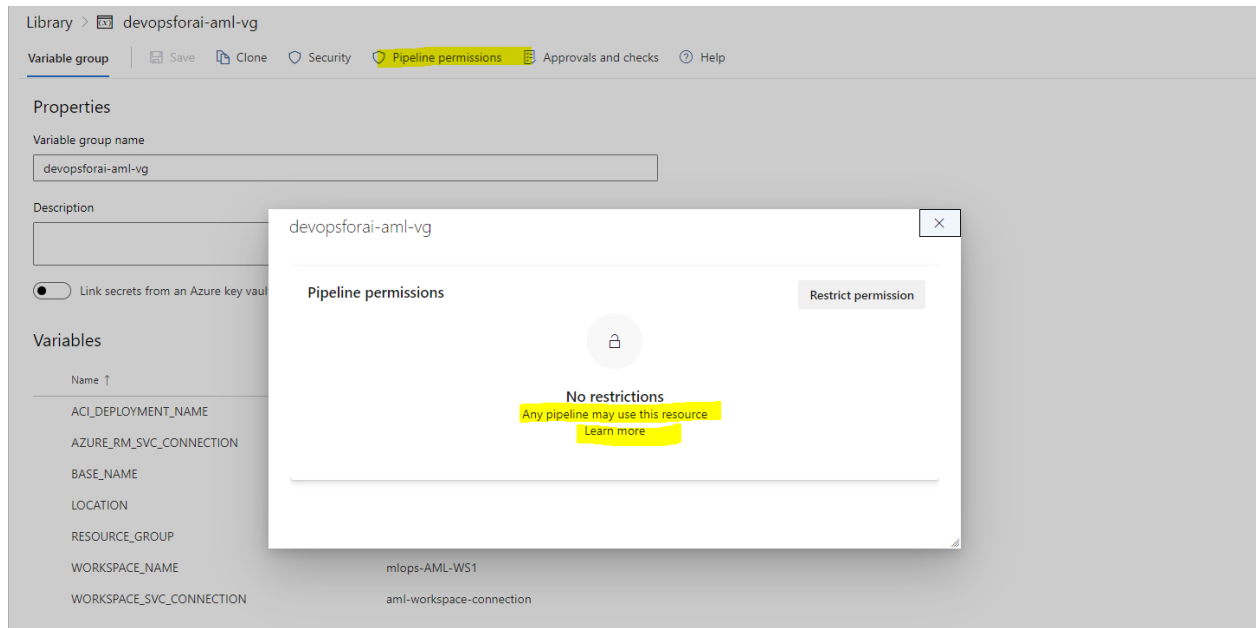
☒ Link secrets from an Azure key vault as variables ⓘ

Variables

Name ↑	Value	
ACI_DEPLOYMENT_NAME	mlops-aci	
AZURE_RM_SVC_CONNECTION	azure-resource-connection	
BASE_NAME	pymlops	
LOCATION	centralus	
RESOURCE_GROUP	mlopsRG	
WORKSPACE_NAME	mlops-AML-WS1	
WORKSPACE_SVC_CONNECTION	aml-workspace-connection	

Add

- Make sure you select the **Allow access to all pipelines** checkbox in the variable group configuration. To do this, first **Save** the variable group, then click **Pipeline Permissions**, then the button with 3 vertical dots, and then **Open access** button



Create an Azure DevOps Service Connection for the Azure Resource Manager

The [IaC provisioning pipeline](#) requires an **Azure Resource Manager service connection**. To create one, in Azure DevOps select **Project Settings**, then **Service Connections**, and create a new one, where:

test-user-59 / pymlops / Settings / Overview

Project Settings

pymlops

Project details

Name: pymlops

Description:

Process: Basic

Save

Project administrators

test-user-59
test-user-59@confiz.com

Add administrator

Azure DevOps services

Service	Description	Status
Boards	Flexible agile planning with boards and cross-product issues	On
Repos	Repos, pull requests, advanced file management and more	On
Pipelines	Build, manage, and scale your deployments to the cloud	On

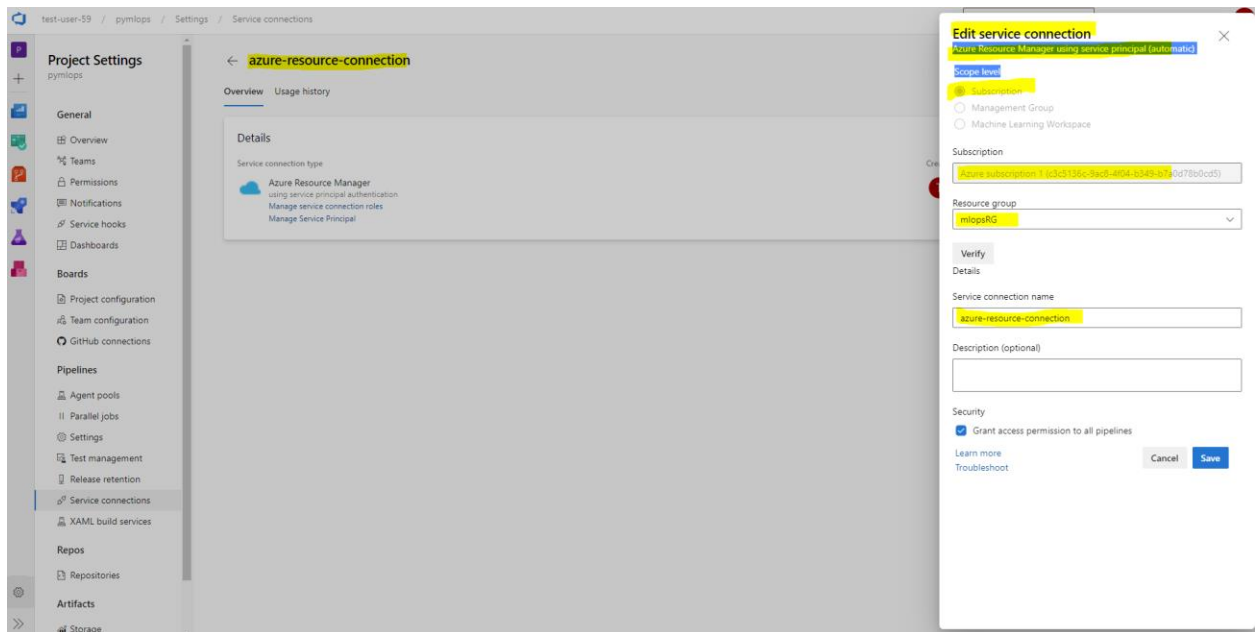
- Type is **Azure Resource Manager**
- Authentication method is **Service principal (automatic)**
- Scope level is **Subscription**
- Leave **Resource Group** empty after selecting your subscription in the dropdown
- Use the same **Service Connection Name** that you used in the variable group you created
- Select **Grant access permission to all pipelines**

Service connections

Filter by keywords

Created by

aml-workspace-connection	
azure-resource-connection	



Create the IaC pipeline

Follow: https://github.com/microsoft/MLOpsPython/blob/master/docs/getting_started.md#create-the-iac-pipeline

In your Azure DevOps project, create a build pipeline from your forked repository:

Connect Select Configure Create pipeline

New pipeline

Where is your code?



Azure Repos Git YAML

Free private Git repositories, pull requests, and code search



Bitbucket Cloud

Hosted by Atlassian



GitHub YAML

Home to the world's largest community of developers



GitHub Enterprise Server YAML

The self-hosted version of GitHub Enterprise



Other Git

Any Internet-facing Git repository



Subversion

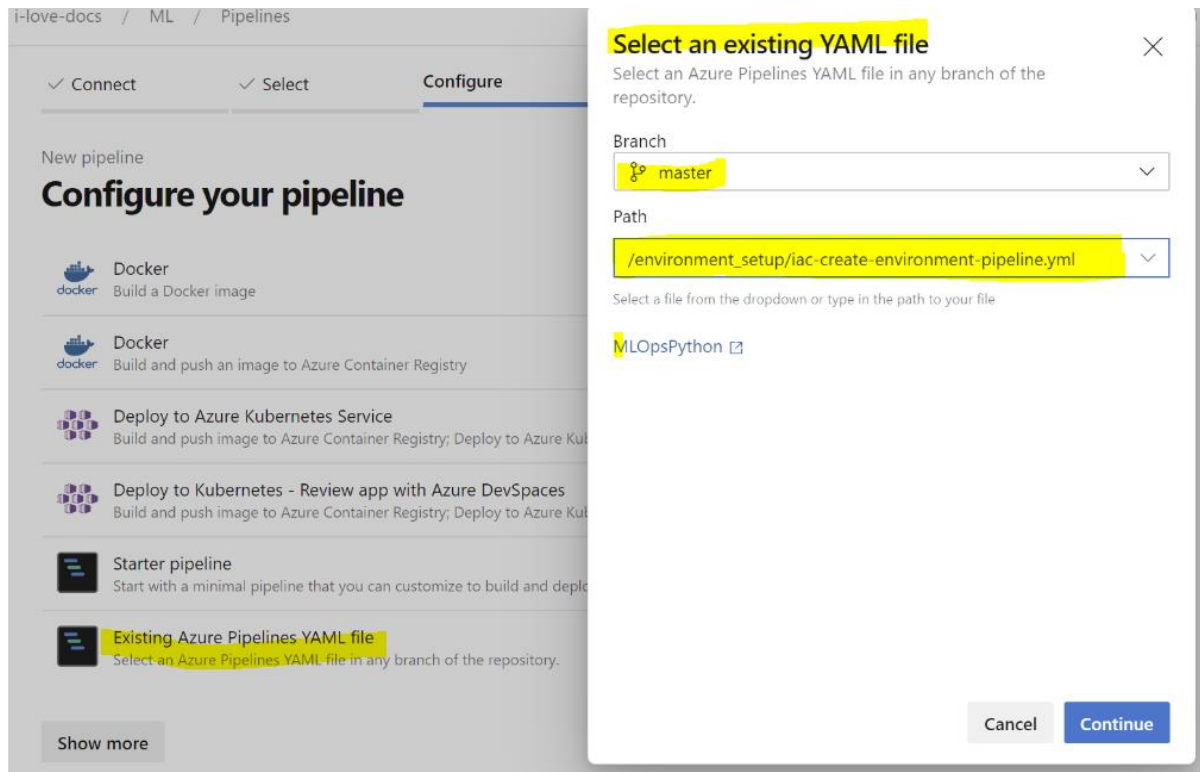
Centralized version control by Apache

Use the classic editor to create a pipeline without YAML.

Name	Last change	Commits
.pipelines	Dec 8, 2022	c2605722 Updated diabetes_regression-clyml test-user-59
bootstrap	Jul 7, 2020	2767d2a7 Update docs and pipeline status badge (#303) Jenny So
charts	Mar 10, 2020	86c6a1f8 Canary pipeline fixes (#224) Tom Care
data	Sep 24, 2020	74885302 Fix Batch Scoring docs (#333) Jenny So
diabetes_regression	Nov 29, 2022	738d0481 Updated deployment_config_aks.yml test-user-59
docs	Dec 15, 2021	98a610ff Clarified/Fixed getting started instructions for WebApp/AppService deplo...
environment_setup	Nov 29, 2022	b257cd92 Updated cloud-environment.json test-user-59
experimentation	Apr 2, 2020	319cae66 Making changes to experiment notebook based on changes to tutorial do...
ml_service	Dec 15, 2021	4b2667e3 Improve readability of exceptions in build pipeline script (#357) Tim Johns...
.env.example	Feb 19, 2021	771c8ef1 fix TRAIN_SCRIPT_PATH value in .env.example (#348) SATO Naoki (Neo)
.gitignore	Feb 1, 2020	962778c8 Manage environments in conda YAML files (#158) Alexandre Gattiker
LICENSE	Jan 30, 2019	c0f0f92a Initial commit Microsoft Open Source
ML README.md	Jul 7, 2020	2767d2a7 Update docs and pipeline status badge (#303) Jenny So

If you are using GitHub, after picking the option above, you'll be asked to authorize GitHub and select the repo you forked. Then you'll have to select your forked repository on GitHub under the **Repository Access** section and click **Approve and Install**.

After the above, and when you're redirected back to Azure DevOps, select the Existing Azure Pipelines YAML file option and set the path to **`/environment_setup/iac-create-environment-pipeline-arm.yml`** or to **`/environment_setup/iac-create-environment-pipeline-tf.yml`**, depending on if you want to deploy your infrastructure using ARM templates or Terraform:



This will create the resources from the variable group (azure devops/pipelines/library) to azure machine learning workspace in Azure.

Contents History Compare Blame

```

1 # CI/PR Pipeline that deploys an ARM template to create or update the resources needed by the other pipelines.
2 trigger:
3   branches:
4     include:
5       - master
6   paths:
7     include:
8       - environment_setup/arm-templates/*
9 pr:
10  branches:
11    include:
12      - master
13  paths:
14    include:
15      - environment_setup/arm-templates/*
16
17 pool:
18   vmImage: "ubuntu-latest"
19
20 variables:
21   - group: devopsforai-aml-vg
22   - name: WORKSPACE_SKU # https://docs.microsoft.com/en-us/azure/machine-learning/overview-what-is-azure-ml#sku
23     value: basic
24
25 steps:
26   - task: AzureResourceGroupDeployment@2
27     inputs:
28       azureSubscription: "$(AZURE_RM_SVC_CONNECTION)"
29       action: "Create Or Update Resource Group"
30       resourceGroupname: "$(RESOURCE_GROUP)"
31       location: $(LOCATION)
32       templateLocation: "Linked artifact"
33       configFile: "$(Build.SourcesDirectory)/environment_setup/arm-templates/cloud-environment.json"
34       overrideParameters: "-baseName $(BASE_NAME) -location $(LOCATION) -workspace $(WORKSPACE_NAME) -sku $(WORKSPACE_SKU)"
35       deploymentMode: "Incremental"
36       displayName: "Deploy ML Ops resources to Azure"
37

```

Azure services

Create a resource, Azure Machine Learning, Resource groups, Azure DevOps organizations, Event Hubs, Event Grid, Storage accounts, Marketplace, Subscriptions, More services

Resources

Recent Favorite

Name	Type	Last Viewed
mlops-AML-WS1	Azure Machine Learning workspace	27 minutes ago
mlopsRG	Resource group	2 weeks ago
Azure subscription 1	Subscription	a month ago
mlopsamlws17791863492	Storage account	a month ago
pymlopsamlss1	Storage account	2 months ago
mlopsamlws17100607244	Key vault	2 months ago
c379c931a1b64e0ea1d4a728b66fb054	Container registry	2 months ago
mlopsamlws11823979281	Application Insights	2 months ago
mlops-AML-WS	Azure Machine Learning workspace	2 months ago
pymlops-AML-KV1	Key vault	2 months ago

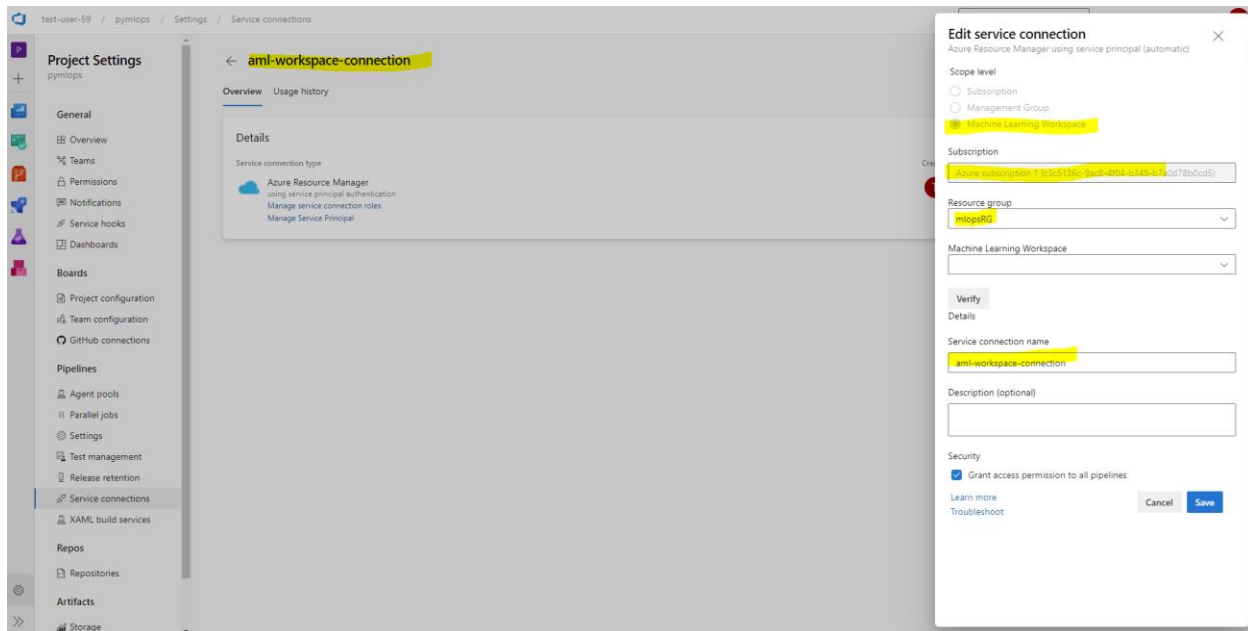
[See all](#)

Note: If you have other errors, one good thing to check is what you used in the variable names. If you end up running the pipeline multiple times, you may also run into errors and need to delete the Azure services and re-run the pipeline -- this should include a resource group, a KeyVault, a Storage Account, a Container Registry, an Application Insights and a Machine Learning workspace.

Create an Azure DevOps Service Connection for the Azure ML Workspace

At this point, you should have an Azure ML Workspace created. Similar to the Azure Resource Manager service connection, you need to create an additional one for the Azure ML Workspace.

Create a new service connection to your Azure ML Workspace using the Machine Learning Extension instructions to enable executing the Azure ML training pipeline. The connection name needs to match `WORKSPACE_SVC_CONNECTION` that you set in the variable group above (e.g., 'aml-workspace-connection').



Note: Similar to the Azure Resource Manager service connection you created earlier, creating a service connection with Azure Machine Learning workspace scope requires 'Owner' or 'User Access Administrator' permissions on the Workspace. You'll need sufficient permission to register an application with your Azure AD tenant, or you can get the ID and secret of a service principal from your Azure AD Administrator. That principal must have Contributor permissions on the Azure ML Workspace.

Set up Build, Release Trigger, and Release Multi-Stage Pipelines

Now that you've provisioned all the required Azure resources and service connections, you can set up the pipelines for training (**Continuous Integration - CI**) and deploying (**Continuous Deployment - CD**) your machine learning model to production. Additionally, you can set up a pipeline for batch scoring.

- **Model CI, training, evaluation, and registration** - triggered on code changes to master branch on GitHub. Runs linting, unit tests, code coverage, and publishes and runs the training pipeline. If a new model is registered after evaluation, it creates a build artifact containing the JSON metadata of the model. Definition: [diabetes_regression-ci.yml](#).
- **Release deployment** - consumes the artifact of the previous pipeline and deploys a model to either [Azure Container Instances \(ACI\)](#), [Azure Kubernetes Service \(AKS\)](#), or [Azure App Service](#) environments. See [Further Exploration](#) for other deployment types. Definition: [diabetes_regression-cd.yml](#).

Note: Edit the pipeline definition to remove unused stages. For example, if you're deploying to Azure Container Instances and Azure Kubernetes Service only, you'll need to delete the unused Deploy_Webapp stage.

- **Batch Scoring Code Continuous Integration** - consumes the artifact of the model training pipeline. Runs linting, unit tests, code coverage, publishes a batch scoring pipeline, and invokes the published batch scoring pipeline to score a model.

These pipelines use a Docker container on the Azure Pipelines agents to accomplish the pipeline steps. The container image `mcr.microsoft.com/mlops/python:latest` is built with [this Dockerfile](#) and has all the necessary dependencies installed for MLOpsPython and **diabetes_regression**. This image is an example of a custom Docker image with a pre-baked environment. The environment is guaranteed to be the same on any building agent, VM, or local machine. *In your project, you'll want to build your own Docker image that only contains the dependencies and tools required for your use case. Your image will probably be smaller and faster, and it will be maintained by your team.*

Set up the Model CI, training, evaluation, and registration pipeline

In your Azure DevOps project, create and run a new build pipeline based on the `./pipelines/diabetes_regression-ci.yml` pipeline definition in your forked repository.

If you plan to use the release deployment pipeline (in the next section), you will need to rename this pipeline to Model-Train-Register-CI.

Note: To rename your pipeline, after you saved it, click Pipelines on the left menu on Azure DevOps, then All to see all the pipelines, then click the menu with the 3 vertical dots that appears when you hover the name of the new pipeline, and click it to pick "Rename/move pipeline".

Start a run of the pipeline if you haven't already, and once the pipeline is finished, check the execution result. Note that the run can take 20 minutes, with time mostly spent in Trigger ML Training Pipeline > Invoke ML Pipeline step. You can track the execution of the AML pipeline by opening the AML Workspace user interface. Screenshots are below:

Azure DevOps test-user-59 / pymlops / Pipelines / Model-Train-Register-CI / 20221129.1

#20221129.1 • Updated cloud-environment.json

Model-Train-Register-CI

This run is being retained as one of 3 recent runs by master (Branch). [View retention leases](#)

Summary Tests Code Coverage Associated pipelines

Manually run by test-user-59 [View 250 changes](#)

Repository and version	Time started and elapsed	Related	Tests and coverage
pymlops master b257cd92	Nov 29, 2022 at 1:01 PM 24m 15s	0 work items 2 published	100% passed 47.44% covered

Warnings 1

Please install dotnet core to enable automatic generation of Html report.
Model CI • Model CI Pipeline • Publish coverage report

Stages Jobs

Model CI
1 job completed 3m 24s
100% tests passed
1 artifact

Train and evaluate m...
3 jobs completed 20m 36s
1 artifact

Azure DevOps test-user-59 / pymlops / Pipelines / pymlops (1) / 20221129.1

#20221129.1 • Updated cloud-environment.json

pymlops (1)

This run is being retained as one of 3 recent runs by master (Branch). [View retention leases](#)

Summary

Manually run by test-user-59 [View 500 changes](#)

Repository and version	Time started and elapsed	Related	Tests and coverage
pymlops master b257cd92	Nov 29, 2022 at 2:23 PM 7m 21s	0 work items 2 consumed	Get started

Stages Jobs

Deploy to ACI
1 job completed 7m 19s

Deploy to AKS
Skipped

Deploy to Webapp
Skipped

Azure DevOps test-user-59 / pymlops / Pipelines / batch_scoring / 20221202.1

#20221202.1 • Updated diabetes_regression-variables-template.yml
batch_scoring

This run is being retained as one of 3 recent runs by master (Branch). [View retention leases](#)

Summary Tests Code Coverage

Manually run by test-user-59

Repository and version	Time started and elapsed	Related	Tests and coverage
pymlops master 11b39874	Dec 2, 2022 at 11:54 AM 8m 16s	0 work items 1 published: 2 consumed	100% passed 47.44% covered

Warnings 1

Please install dotnet core to enable automatic generation of Html report.
Build Batch Scoring Pipeline • Publish coverage report

Jobs

Name	Status	Duration
Build Batch Scoring Pipeline	Success	3m 16s
Run Batch Scoring Pipeline	Success	4m 50s

Azure DevOps test-user-59 / pymlops / Pipelines / batch_scoring / 20221202.1

Jobs in run #20221202.1
batch_scoring

Batch Scoring Pipeline CI

- Build Batch Scoring Pipeline 3m 16s
 - Initialize job 4s
 - Initialize containers 1m 22s
 - Checkout pymlops@master to s 2s
 - Run lint tests 1s
 - Run unit tests 1s
 - Publish test results 3s
 - Publish coverage report 1s
 - Download Pipeline Artifacts 4s
 - Parse Json for Model Name and ... <1s
 - Publish Batch Scoring Pipeline 1m 26s
 - Post-job: Checkout pymlops@m... <1s
 - Stop Containers <1s
 - Finalize Job <1s
- Run Batch Scoring Pipeline 4m 50s
 - Invoke Batch Scoring pipeline 4m 50s
- Finalize build
 - Report build status <1s

Run Batch Scoring Pipeline

```

1 Started: Dec 2, 2022 at 11:58 AM
2 Duration: 4m 50s
3
4 Job preparation parameters
5 A Parent pipeline used these runtime parameters

```

Also check the published training pipeline in your newly created AML workspace in [Azure Machine Learning Studio](#):

Microsoft Azure Machine Learning Studio

Confiz Limited > mlops-AML-WS1 > Pipelines

Subscription disabled. This subscription is disabled and therefore marked as read only. You cannot perform any write actions on this subscription until it is re-enabled. View subscription

Pipelines

Pipeline jobs Pipeline endpoints Pipeline drafts

Refresh Disable Enable Edit columns Reset view Include disabled

Search

Updated by Status: Active All filters Clear all

Showing 1-4 of 4 pipeline endpoints

Name	Description	Date updated	Updated by	Last job submit time	Last job status	Status
diabetes-scoring-pipeline	Diabetes Batch Scoring Pipeline	Dec 2, 2022 3:50 PM	40a180f1-41ec-477c-8284-b5afa055282a	Dec 2, 2022 3:45 PM	Finished	Active
diabetes-scoring-pipeline	Diabetes Batch Scoring Pipeline	Dec 2, 2022 12:03 PM	40a180f1-41ec-477c-8284-b5afa055282a	Dec 2, 2022 11:58 AM	Finished	Active
diabetes-scoring-pipeline	Diabetes Batch Scoring Pipeline	Dec 1, 2022 2:40 PM	40a180f1-41ec-477c-8284-b5afa055282a	Dec 1, 2022 2:37 PM	Failed	Active
diabetes-training-pipeline	Model training/retraining pipeline	Nov 29, 2022 1:23 PM	40a180f1-41ec-477c-8284-b5afa055282a	Nov 29, 2022 1:08 PM	Finished	Active

Great, you now have the build pipeline for training set up which automatically triggers every time there's a change in the master branch!

After the pipeline is finished, you'll also see a new model in the AML Workspace model registry section:

Microsoft Azure Machine Learning Studio

Confiz Limited > mlops-AML-WS1 > Models

Subscription disabled. This subscription is disabled and therefore marked as read only. You cannot perform any write actions on this subscription until it is re-enabled. View subscription

Model List

Register Refresh Delete Archive Deploy Compare (preview) Edit columns Reset view Show latest versions only Include archived

Search

Created on Created by Tags All filters Clear all

Showing 1-1 of 1 models

Name	Version	Experiment	Job (Run ID)	Created on	Tags	Properties	Created by
diabetes_regression_model.pkl	1			Nov 29, 2022 1:22 PM	area : diabetes_regression		Service Principal

To disable the automatic trigger of the training pipeline, change the auto-trigger-training variable as listed in the `.pipelines\diabetes_regression-ci.yml` pipeline to `false`. You can also override the variable at runtime execution of the pipeline. The pipeline stages are summarized below:

Model CI

- Linting (code quality analysis)
- Unit tests and code coverage analysis
- Build and publish *ML Training Pipeline* in an *ML Workspace*

Train model

- Determine the ID of the *ML Training Pipeline* published in the previous stage.
- Trigger the *ML Training Pipeline* and wait for it to be completed.
 - This is an **agentless** job. The CI pipeline can wait for ML pipeline completion for hours or even days without using agent resources.
- Determine if a new model was registered by the *ML Training Pipeline*.
 - If the model evaluation step of the AML Pipeline determines that the new model doesn't perform any better than the previous one, the new model won't register, and the *ML Training Pipeline* will be **canceled**. In this case, you'll see a message in the 'Train Model' job under the 'Determine if evaluation succeeded and new model is registered' step saying '**Model was not registered for this run.**'
 - See [evaluate_model.py](#) for the evaluation logic. This is a simplified test that just looks at MSE to decide whether or not to register a new model. A more realistic verification would also do some error analysis and verify the inferences/error distribution against a test dataset, for example.
 - **Note:** *while it's possible to do an Evaluation Step as part of the ADO pipeline, this evaluation is logically part of the work done by Data Scientists, and as such the recommendation is that this step is done as part of the AML Pipeline and not ADO pipelines.*
 - [Additional Variables and Configuration](#) for configuring this and other behavior.

Create pipeline artifact

- Get the info about the registered model
- Create an Azure DevOps pipeline artifact called model that contains a model.json file containing the model information, for example:

Set up the Release Deployment and/or Batch Scoring pipelines

PRE-REQUISITES

To use these pipelines:

1. Follow the steps to set up the Model CI, training, evaluation, and registration pipeline.
2. You **must** rename your model CI/train/eval/register pipeline to Model-Train-Register-CI.

These pipelines rely on the model CI pipeline and reference it by name.

If you would like to change the name of your model CI pipeline, you must edit this section of yml for the CD and batch scoring pipeline, where it says `source: Model-Train-Register-CI` to use your own name.

```
trigger: none
resources:
  containers:
  - container: mlops
    image: mcr.microsoft.com/mlops/python:latest
  pipelines:
  - pipeline: model-train-ci
    source: Model-Train-Register-CI # Name of the triggering pipeline
    trigger:
      branches:
        include:
          - master
```

The release deployment and batch scoring pipelines have the following behaviors:

- The pipeline will **automatically trigger** on completion of the Model-Train-Register-CI pipeline for the master branch.
- The pipeline will default to using the latest successful build of the Model-Train-Register-CI pipeline. It will deploy the model produced by that build.
- You can specify a Model-Train-Register-CI build ID when running the pipeline manually. You can find this in the url of the build, and the model registered from that build will also be tagged with the build ID. This is useful to skip model training and registration and deploy/score a model successfully registered by a Model-Train-Register-CI build.
 - For example, if you navigate to a specific run of your CI pipeline, the URL should be something like `https://dev.azure.com/yourOrgName/yourProjectName/_build/results?buildId=653&view=results`. **653** is the build ID in this case.

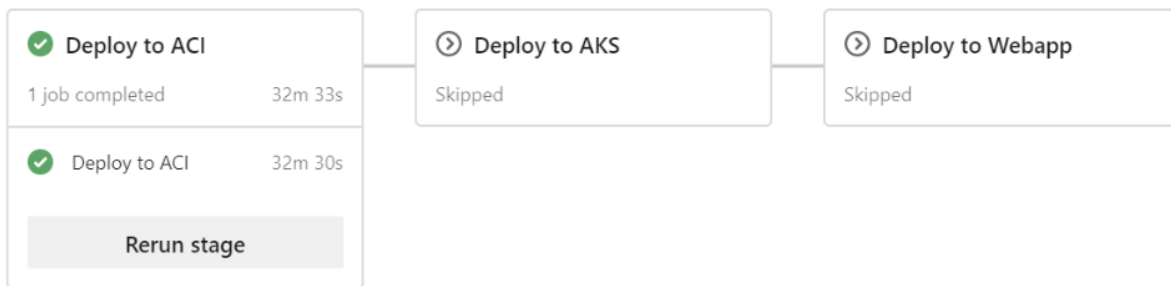
Set up the Release Deployment pipeline

In your Azure DevOps project, create and run a new **build** pipeline based on the [./pipelines/diabetes_regression-cd.yml](#) pipeline definition in your forked repository. It is recommended you rename this pipeline to something like Model-Deploy-CD for clarity.

Note: While Azure DevOps supports both Build and Release pipelines, when using YAML you don't usually need to use Release pipelines. This repository assumes the usage only of Build pipelines.

Your first run will use the latest model created by the Model-Train-Register-CI pipeline. Once the pipeline is finished, check the execution result:

Stages Jobs



Set up the Batch Scoring pipeline

https://github.com/microsoft/MLOpsPython/blob/master/docs/getting_started.md#set-up-the-batch-scoring-pipeline

Data Drift Monitoring script

Open the “Notebooks” tab in the workspace.
use the following script:

```

import json
import pandas as pd
import numpy as np
import requests
import zipfile
import io
import plotly.offline as py #working offline
import plotly.graph_objs as go
from evidently.pipeline.column_mapping import ColumnMapping
from evidently.report import Report
from evidently.metric_preset import DataDriftPreset
import mlflow
import mlflow.sklearn
from mlflow.tracking import MlflowClient

from sklearn.datasets import load_diabetes

import warnings
warnings.filterwarnings('ignore')
warnings.simplefilter('ignore')

sample_data = load_diabetes()
raw_data = pd.DataFrame(

```

```

data=sample_data.data,
columns=sample_data.feature_names)
raw_data.head()

raw_data['date'] = pd.date_range(start='1/1/2021', periods=len(raw_data), freq='D')
raw_data.head()

raw_data = raw_data.set_index('date')
raw_data.head()

data_columns = ColumnMapping()
data_columns.numerical_features = ['age', 'sex', 'bmi', 'bp', 's1']

#set reference dates
reference_dates = ('2021-01-01', '2021-01-28')

#set experiment batches dates
experiment_batches = [
    ('2021-02-01', '2021-02-28'),
    ('2021-03-01', '2021-03-31'),
    ('2021-04-01', '2021-04-30'),
    ('2021-05-01', '2021-05-31'),
    ('2021-06-01', '2021-06-30'),
    ('2021-07-01', '2021-07-31'),
]

data_drift_report = Report(metrics=[DataDriftPreset()])
data_drift_report.run(reference_data=raw_data[:100], current_data=raw_data[100:200], column_mapping=data_columns)
report = data_drift_report.as_dict()

report["metrics"][1]["result"]["drift_by_columns"]

#evaluate data drift with Evidently Profile
def detect_dataset_drift(reference, production, column_mapping, get_ratio=False):
    """
    Returns True if Data Drift is detected, else returns False.
    If get_ratio is True, returns the share of drifted features.
    The Data Drift detection depends on the confidence level and the threshold.
    For each individual feature Data Drift is detected with the selected confidence (default value is 0.95).
    Data Drift for the dataset is detected if share of the drifted features is above the selected threshold (default value is 0.5).
    """

```

```

data_drift_report = Report(metrics=[DataDriftPreset()])
data_drift_report.run(reference_data=reference, current_data=production, column_mapping=column_mapping
)
report = data_drift_report.as_dict()

if get_ratio:
    return report["metrics"][0]["result"]["drift_share"]
else:
    return report["metrics"][0]["result"]["dataset_drift"]

```

#evaluate data drift with Evidently Profile

```

def detect_features_drift(reference, production, column_mapping, get_scores=False):
    """
    Returns True if Data Drift is detected, else returns False.
    If get_scores is True, returns scores value (like p-value) for each feature.
    The Data Drift detection depends on the confidence level and the threshold.
    For each individual feature Data Drift is detected with the selected confidence (default value is 0.95
    ).
    """
    data_drift_report = Report(metrics=[DataDriftPreset()])
    data_drift_report.run(reference_data=reference, current_data=production, column_mapping=column_mapping
    )
    report = data_drift_report.as_dict()

    drifts = []
    num_features = column_mapping.numerical_features if column_mapping.numerical_features else []
    cat_features = column_mapping.categorical_features if column_mapping.categorical_features else []
    for feature in num_features + cat_features:
        drift_score = report["metrics"][1]["result"]["drift_by_columns"][feature]["drift_score"]
        if get_scores:
            drifts.append((feature, drift_score))
        else:
            drifts.append((feature, report["metrics"][1]["result"]["drift_by_columns"][feature]["drift_det
ected"]))
    return drifts

```

```
features_historical_drift = []
```

```

for date in experiment_batches:
    drifts = detect_features_drift(raw_data.loc[reference_dates[0]:reference_dates[1]],
                                raw_data.loc[date[0]:date[1]],
                                column_mapping=data_columns)

```

```

features_historical_drift.append([x[1] for x in drifts])

features_historical_drift_frame = pd.DataFrame(features_historical_drift,
                                              columns = data_columns.numerical_features)

fig = go.Figure(data=go.Heatmap(
    z = features_historical_drift_frame.astype(int).transpose(),
    x = [x[1] for x in experiment_batches],
    y = data_columns.numerical_features,
    hoverongaps = False,
    xgap = 1,
    ygap = 1,
    zmin = 0,
    zmax = 1,
    showscale = False,
    colorscale = 'Bluered'
))

fig.update_xaxes(side="top")

fig.update_layout(
    xaxis_title = "Timestamp",
    yaxis_title = "Feature Drift"
)

fig.show()

features_historical_drift_pvalues = []

for date in experiment_batches:
    drifts = detect_features_drift(raw_data.loc[reference_dates[0]:reference_dates[1]],
                                  raw_data.loc[date[0]:date[1]],
                                  column_mapping=data_columns,
                                  get_scores=True)

    features_historical_drift_pvalues.append([x[1] for x in drifts])

features_historical_drift_pvalues_frame = pd.DataFrame(features_historical_drift_pvalues,
                                                       columns = data_columns.numerical_features)

fig = go.Figure(data=go.Heatmap(
    z = features_historical_drift_pvalues_frame.transpose(),
    x = [x[1] for x in experiment_batches],
    y = features_historical_drift_pvalues_frame.columns,
    hoverongaps = False,
    xgap = 1,

```



```

        ygap = 1,
        zmin = 0,
        zmax = 1,
        colorscale = 'reds_r'
    )
)

```

```
fig.update_xaxes(side="top")
```

```

fig.update_layout(
    xaxis_title = "Timestamp",
    yaxis_title = "p-value"
)

```

```
fig.show()
```

Dataset Drift

```
dataset_historical_drift = []
```

```

for date in experiment_batches:
    dataset_historical_drift.append(detect_dataset_drift(raw_data.loc[reference_dates[0]:reference_dates[1]
    ],
    raw_data.loc[date[0]:date[1]],
    column_mapping=data_columns))

```

```

fig = go.Figure(data=go.Heatmap(
    z = [[1 if x == True else 0 for x in dataset_historical_drift]],
    x = [x[1] for x in experiment_batches],
    y = [''],
    hoverongaps = False,
    xgap = 1,
    ygap = 1,
    zmin = 0,
    zmax = 1,
    colorscale = 'Bluered',
    showscale = False
)
)
fig.update_xaxes(side="top")
fig.update_layout(
    xaxis_title = "Timestamp",
    yaxis_title = "Dataset Drift"
)
fig.show()

```

```
dataset_historical_drift_ratio = []
```

```

for date in experiment_batches:
    dataset_historical_drift_ratio.append(detect_dataset_drift(raw_data.loc[reference_dates[0]:reference_d
ates[1]],
                                                                raw_data.loc[date[0]:date[1]],
                                                                column_mapping=data_columns,
                                                                get_ratio=True))

fig = go.Figure(data=go.Heatmap(
    z = [dataset_historical_drift_ratio],
    x = [x[1] for x in experiment_batches],
    y = [''],
    hoverongaps = False,
    xgap = 1,
    ygap = 1,
    zmin = 0.5,
    zmax = 1,
    colorscale = 'reds'
))
fig.update_xaxes(side="top")
fig.update_layout(
    xaxis_title = "Timestamp",
    yaxis_title = "Dataset Drift"
)
fig.show()

```

Log Dataset Drift in MLflow

```

#log into MLflow
client = MlflowClient()

#set experiment
mlflow.set_experiment('Dataset Drift Analysis with Evidently')

#start new run
for date in experiment_batches:
    with mlflow.start_run() as run:
        # Log parameters
        mlflow.log_param("begin", date[0])
        mlflow.log_param("end", date[1])

        # Log metrics
        metric = detect_dataset_drift(raw_data.loc[reference_dates[0]:reference_dates[1]],
                                      raw_data.loc[date[0]:date[1]],

```

```
        column_mapping=data_columns,  
        get_ratio=True)  
  
mlflow.log_metric('dataset drift', metric)  
  
print(run.info)
```