MLOps helps us to understand how to build a Continuous Integration and Continuous Delivery pipeline for an ML/AI project.

Let us use the Azure DevOps Project for build and release/deployment pipelines along with Azure ML services for model retraining pipeline, model management and operationalization.

**Architecture:**

Diagram

Description automatically generated with medium confidence

**PREREQUISITE:**

* Active Azure subscription
* At least contributor access to Azure subscription

In this example, we are using a sample ML project **diabetes\_regression** to set up MLOPSPython. The project creates a linear regression model to predict diabetes and has CI/CD DevOps practices enabled for model training and serving.

For the purpose of demo, I am using a “Pay as you go” subscription model of azure to be able to run CI/CD jobs.

**STEP1: CREATION OF AZURE ACCOUNT**

For details on how to create an azure account please use the link below

<https://azure.microsoft.com/en-in/free/>

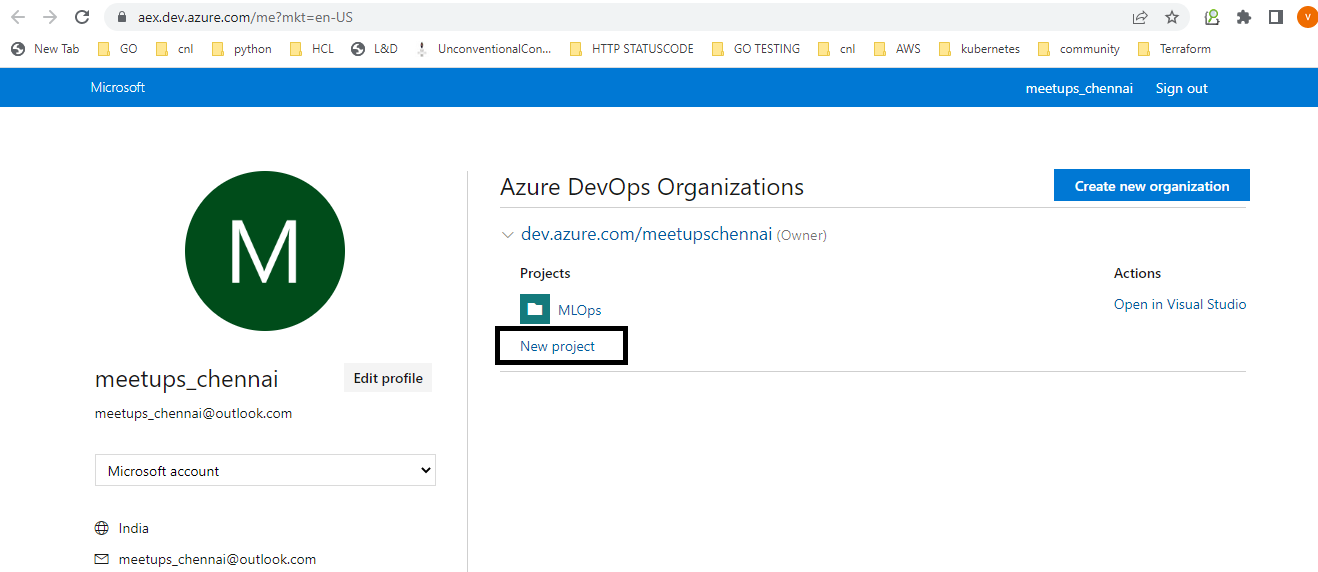
**STEP2:** **CREATION OF DEVOPS ORGANIZATION**

To create an Azure DevOps organization, please refer to

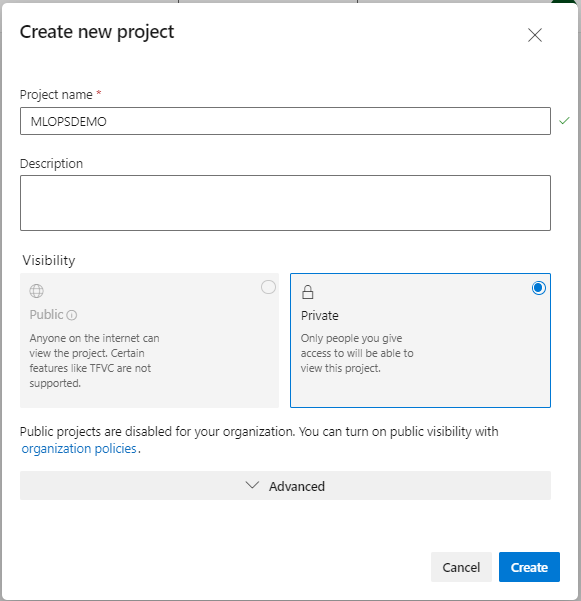
<https://learn.microsoft.com/en-us/azure/devops/organizations/accounts/create-organization?view=azure-devops>

**STEP3: CREATION OF PROJECT IN AZURE DEVOPS**

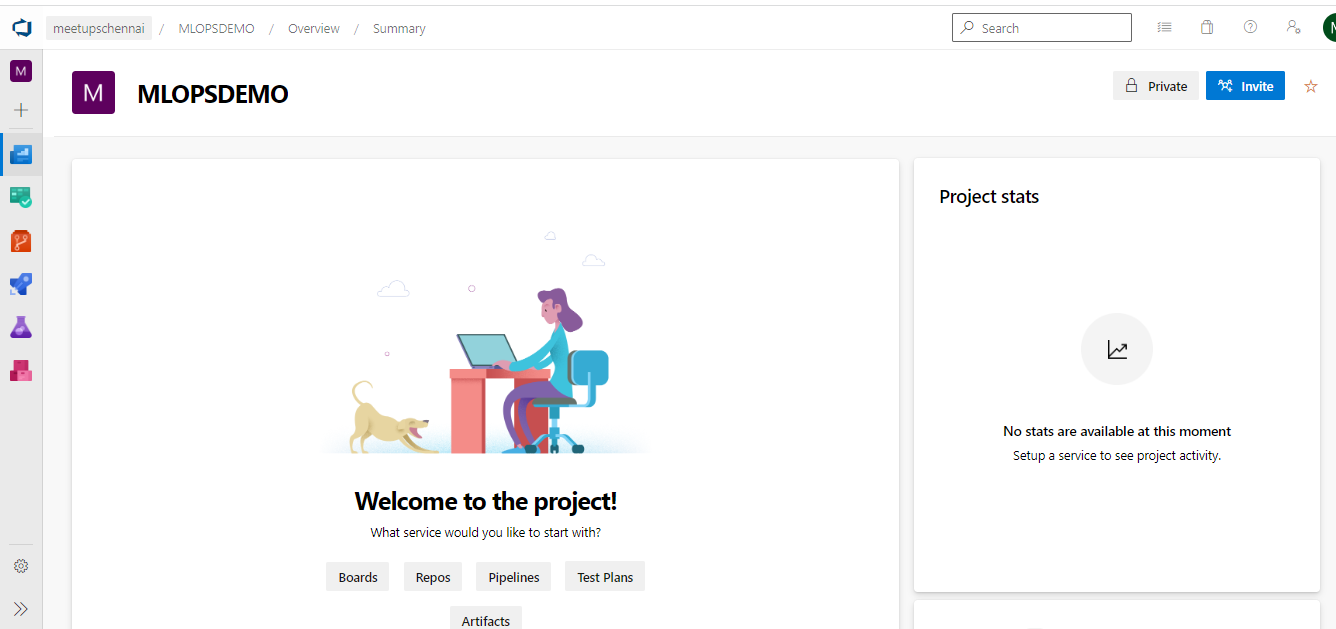
1. Navigate to Azure DevOps Organization, Click on “New Project”



1. Enter the name of the project, click create



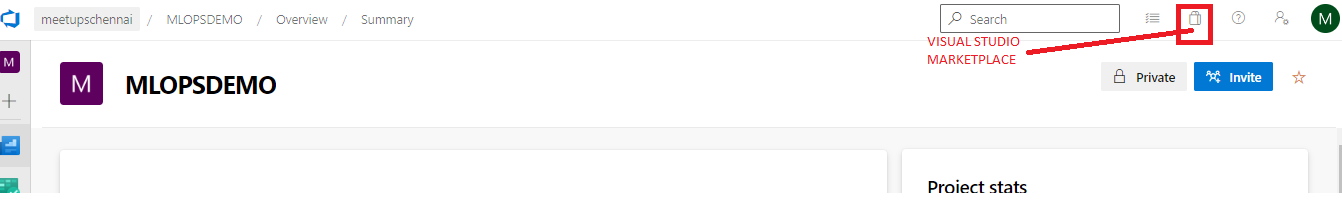
Now we have the Azure DevOps Project Created.



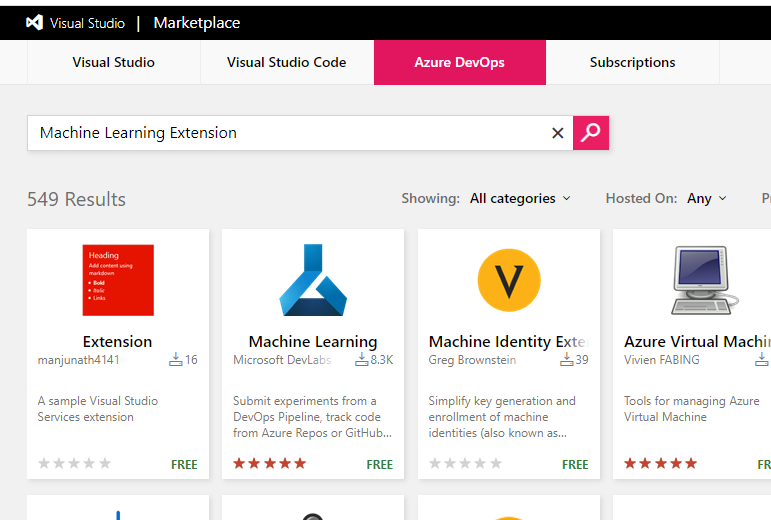
**Install the Azure Machine Learning extension**

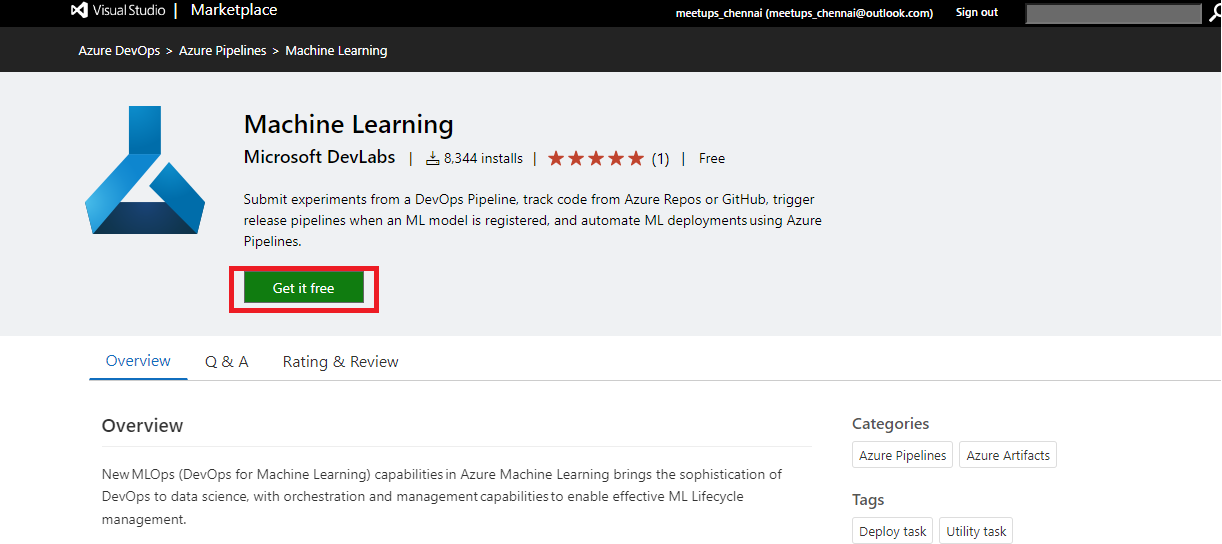
Install the **Azure Machine Learning** extension to your Azure DevOps organization from the [Visual Studio Marketplace](https://marketplace.visualstudio.com/items?itemName=ms-air-aiagility.vss-services-azureml) by clicking "Get it free" and following the steps. The UI will tell you if try to add it and it's already installed.

This extension contains the Azure ML pipeline tasks and adds the ability to create Azure ML Workspace service connections.



Click on it 🡪 Go to “Browse Marketplace” 🡪 search for “Machine Learning Extension”





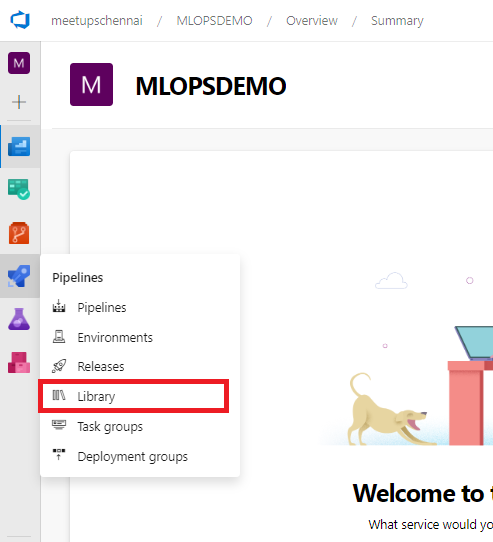
Once it is installed, go back to the project that has been created and navigate to

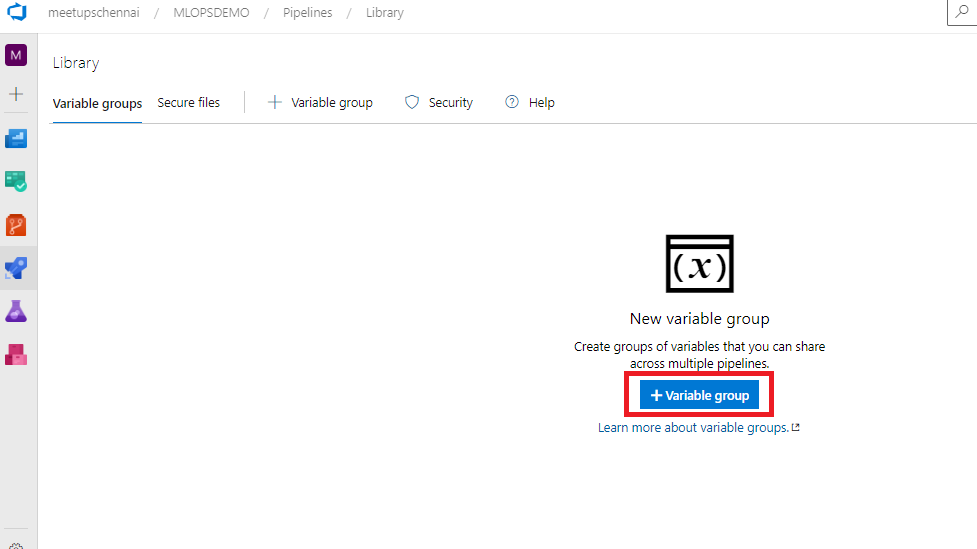
Pipelines 🡪 Library 🡪 Create Variable Group

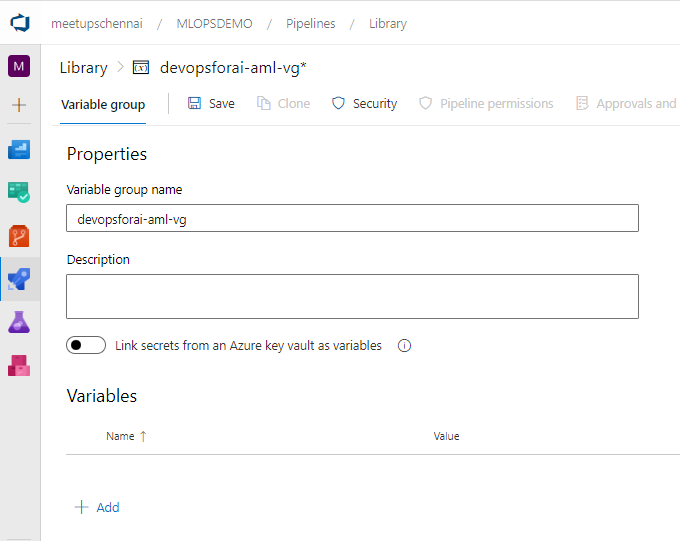
**Create a Variable Group for your Pipeline:**

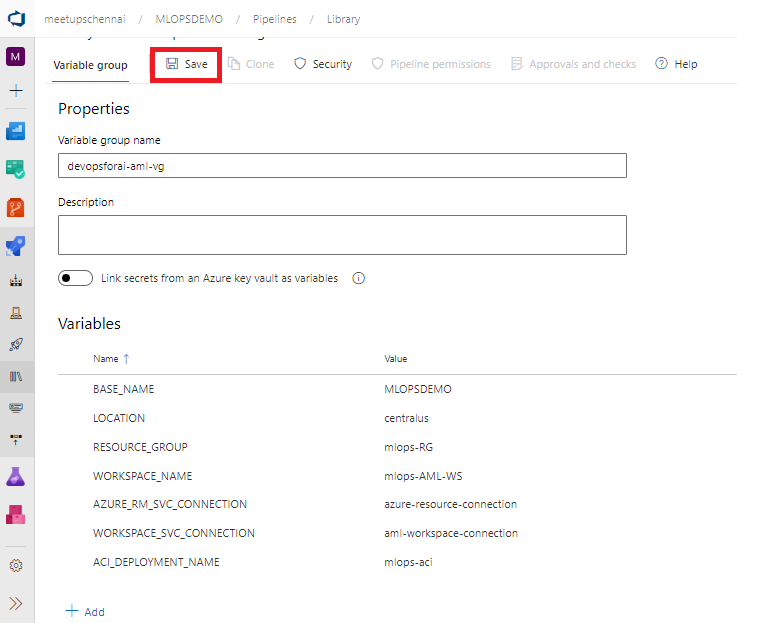
Create a variable group named **devopsforai-aml-vg**. The YAML pipeline definitions in this repository refer to this variable group by name.

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 az account list-locations -o table 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](https://github.com/VijayalakshmiBakthavachalam/MLOpsPython/blob/master/docs/getting_started.md#create-an-azure-devops-service-connection-for-the-azure-resource-manager) name |
| WORKSPACE\_SVC\_CONNECTION | aml-workspace-connection | [Azure ML Workspace Service Connection](https://github.com/VijayalakshmiBakthavachalam/MLOpsPython/blob/master/docs/getting_started.md#create-an-azure-devops-azure-ml-workspace-service-connection) name |
| ACI\_DEPLOYMENT\_NAME | mlops-aci | [Azure Container Instances](https://azure.microsoft.com/en-us/services/container-instances/) name |

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.

**BASE\_NAME** is used as a prefix for naming Azure resources and should be unique. When sharing an Azure subscription, the prefix allows you to avoid naming collisions for resources that require unique names, for example, Azure Blob Storage and Registry DNS. Make sure to set BASE\_NAME to a unique name so that created resources will have unique names, for example, MyUniqueMLamlcr, MyUniqueML-AML-KV, and so on. The length of the BASE\_NAME value shouldn't exceed 10 characters and must contain letters and numbers only.

**LOCATION** is the name of the [Azure location](https://azure.microsoft.com/en-us/global-infrastructure/locations/) for your resources. There should be no spaces in the name. For example, central, westus, northeurope. You can list all the region codes by running az account list-locations -o table in the Azure CLI.

**RESOURCE\_GROUP** is used as the name for the resource group that will hold the Azure resources for the solution. If providing an existing Azure ML Workspace, set this value to the corresponding resource group name.

**WORKSPACE\_NAME** is used for creating the Azure Machine Learning Workspace. While you should be able to provide an existing Azure ML Workspace if you have one, you will run into problems if this has been provisioned manually and the naming of the associated storage account doesn't follow the convention followed in this repo -- as the environment provisioning will try to associate it with a new Storage Account and this is not supported. To avoid these problems, specify a new workspace/unique name.

**AZURE\_RM\_SVC\_CONNECTION** is used by the [Azure Pipeline](https://github.com/VijayalakshmiBakthavachalam/MLOpsPython/blob/master/environment_setup/iac-create-environment-pipeline.yml) in Azure DevOps that creates the Azure ML workspace and associated resources through Azure Resource Manager. You'll create the connection in a [step below](https://github.com/VijayalakshmiBakthavachalam/MLOpsPython/blob/master/docs/getting_started.md#create-an-azure-devops-service-connection-for-the-azure-resource-manager).

**WORKSPACE\_SVC\_CONNECTION** is used to reference a service connection for the Azure ML workspace. You'll create the connection after [provisioning the workspace](https://github.com/VijayalakshmiBakthavachalam/MLOpsPython/blob/master/docs/getting_started.md#provisioning-resources-using-azure-pipelines) in the Create an Azure DevOps Service Connection for the Azure ML Workspace section below.

**ACI\_DEPLOYMENT\_NAME** is used for naming the scoring service during deployment to [Azure Container Instances](https://azure.microsoft.com/en-us/services/container-instances/).

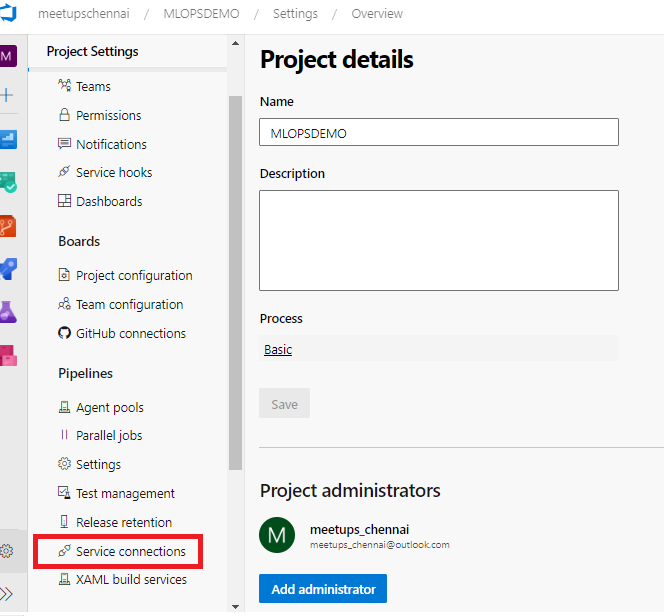
**Provisioning resources using Azure Pipelines**

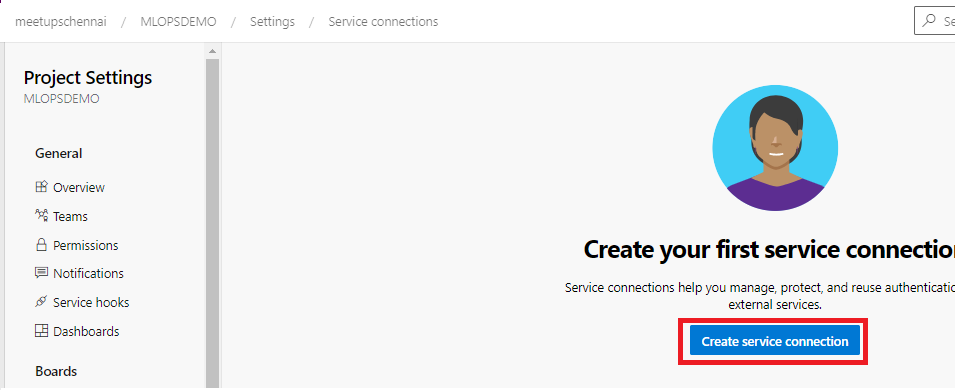
The easiest way to create required Azure resources (Resource Group, Azure ML Workspace, Container Registry, and others) is to use the **Infrastructure as Code (IaC)** pipeline with ARM templates or the pipeline with Terraform templates. The pipeline takes care of setting up all required resources based on these Azure Resource Manager templates, or based on these Terraform templates.

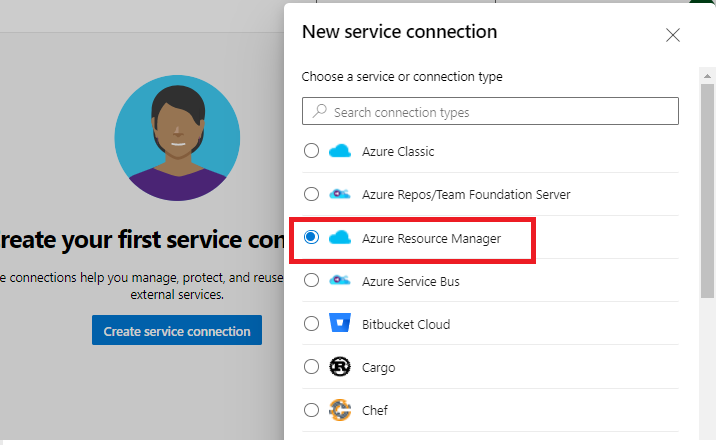
**Create an Azure DevOps Service Connection for the Azure Resource Manager**

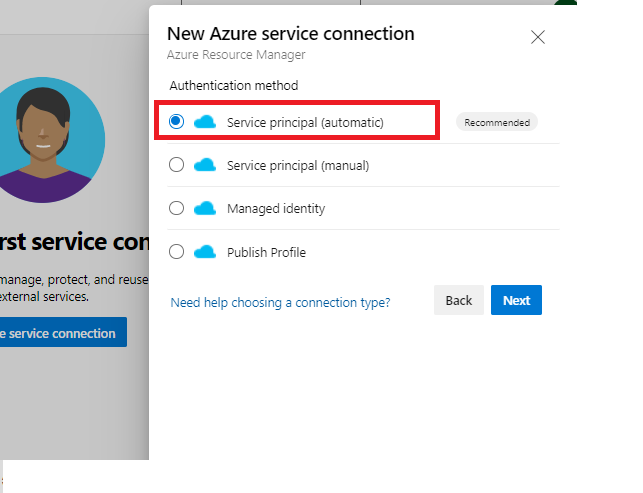
The IaC provisioning pipeline requires an **Azure Resource Manager** service connection. To create one, in Azure DevOps select **Project Settings**, then **Service Connections**, and create a new one, where:

* Type in **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**

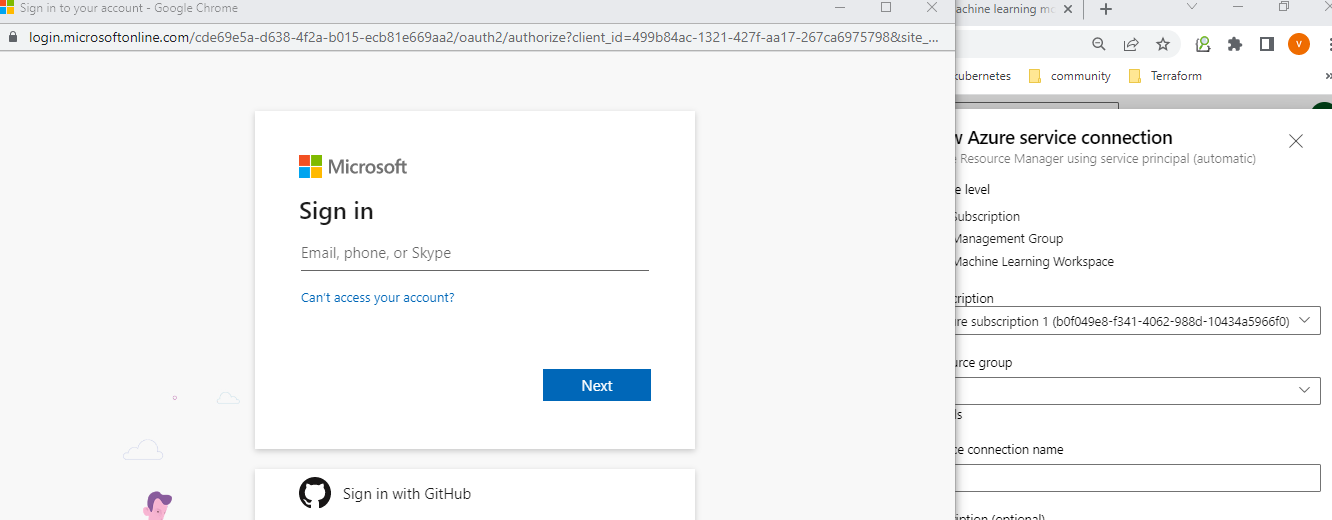


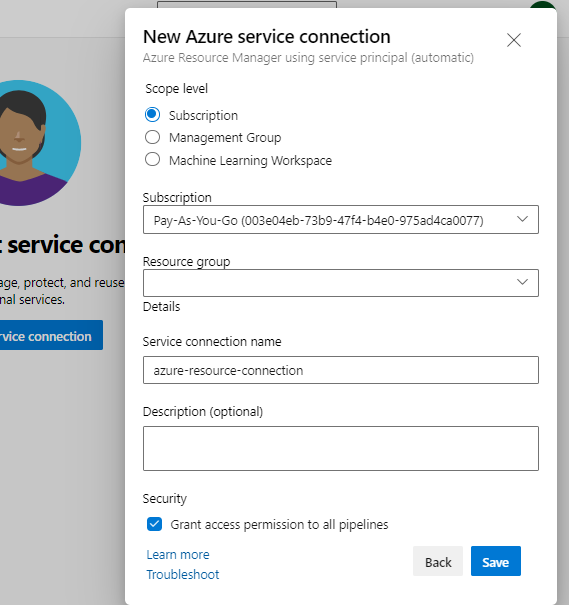


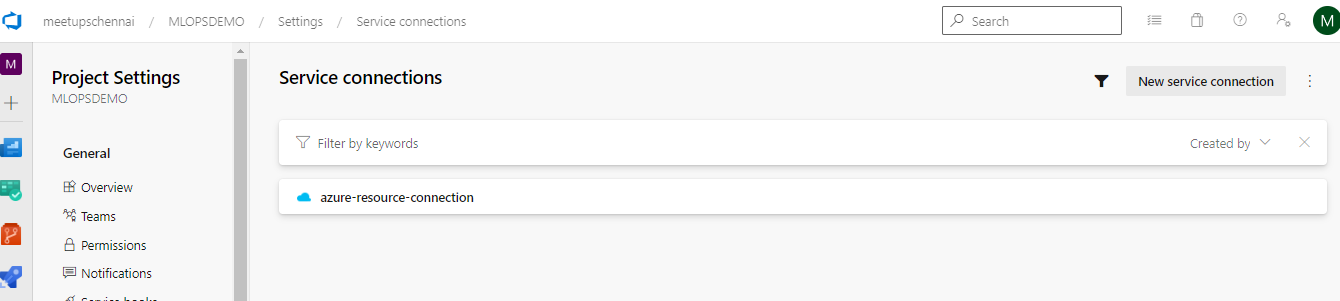




While trying to add service connection, we will be asked to authenticate using the azure account.



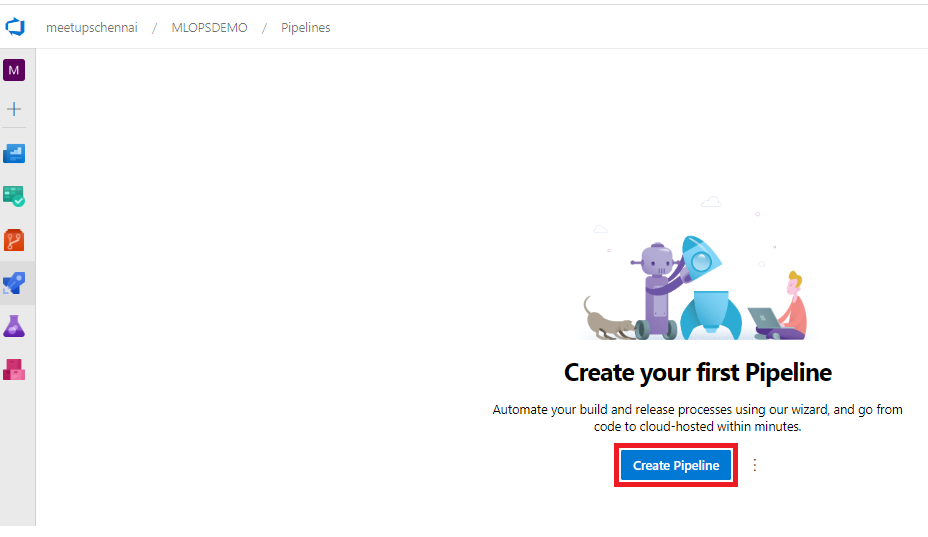


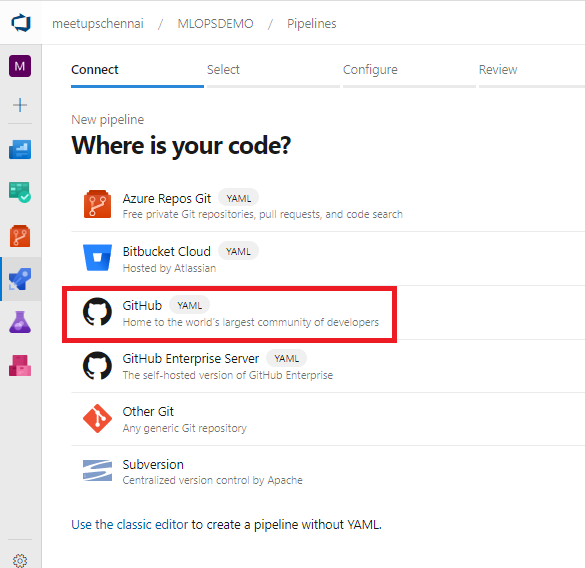


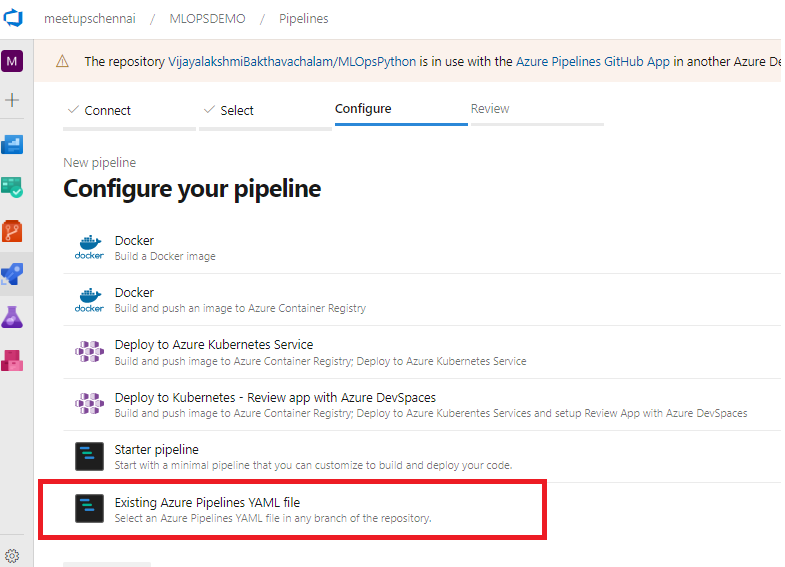
**Create the IaC Pipeline**

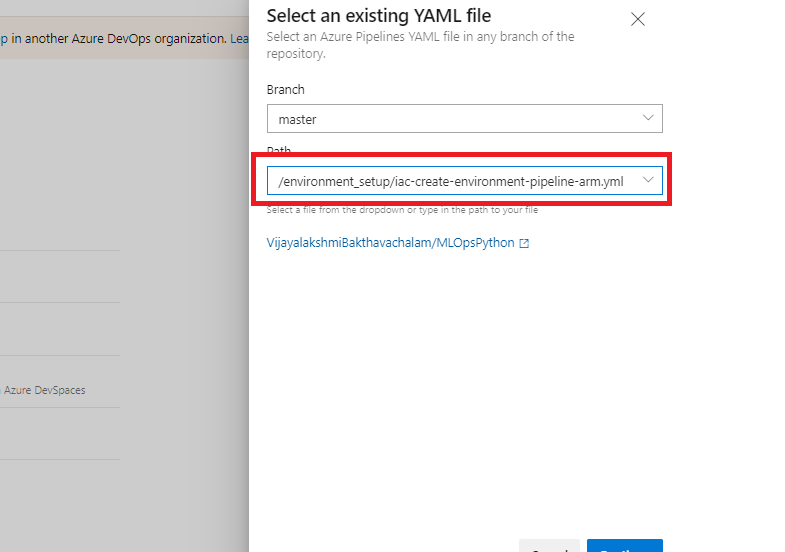
* In your Azure DevOps project, create a build pipeline from the repository:
* Navigate to Pipelines 🡪 Pipelines 🡪 Create Pipeline Select the Source code repository (I am choosing GitHub – when choosing it for the first time, the authorization will be checked on) 🡪 Choose the repository where the code resides 🡪 then choose “Existing Azure Pipelines from YAML” 🡪 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:
* Save the Pipeline 🡪 Rename the Pipeline for ease of reference and reuse 🡪 Pipelines 🡪 Pipelines 🡪Click on 3 dots and hit rename

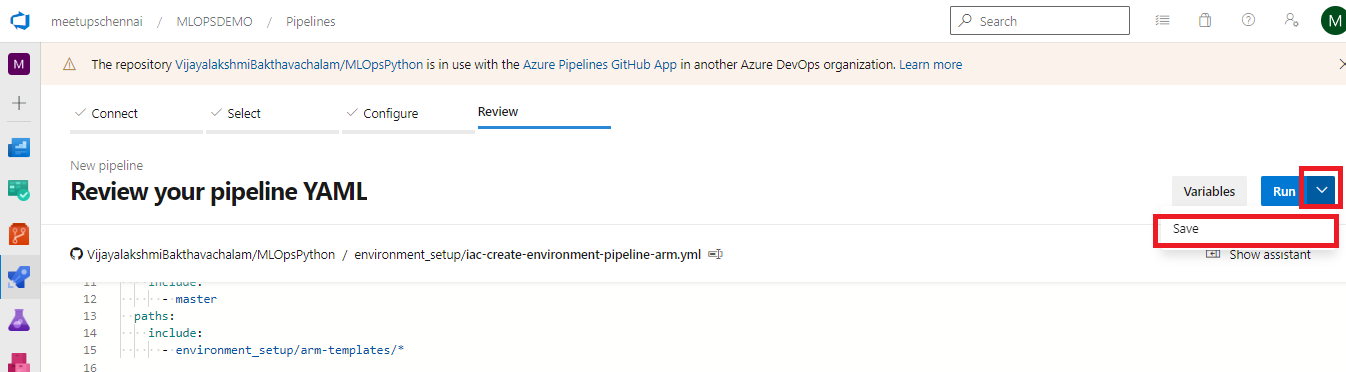
If you decide to use Terraform, make sure the ['Terraform Build & Release Tasks' from Charles Zipp](https://marketplace.visualstudio.com/items?itemName=charleszipp.azure-pipelines-tasks-terraform) is installed.

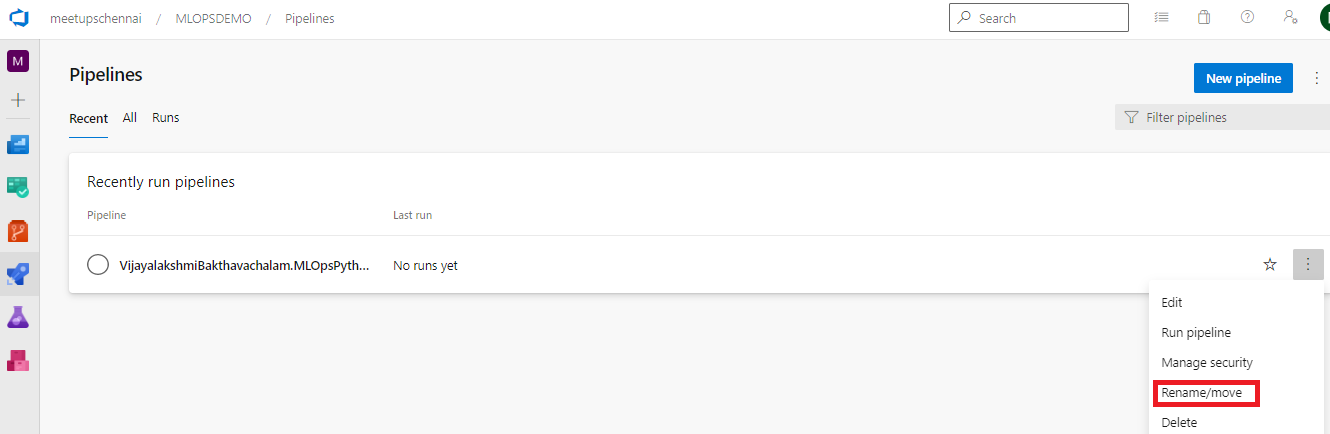


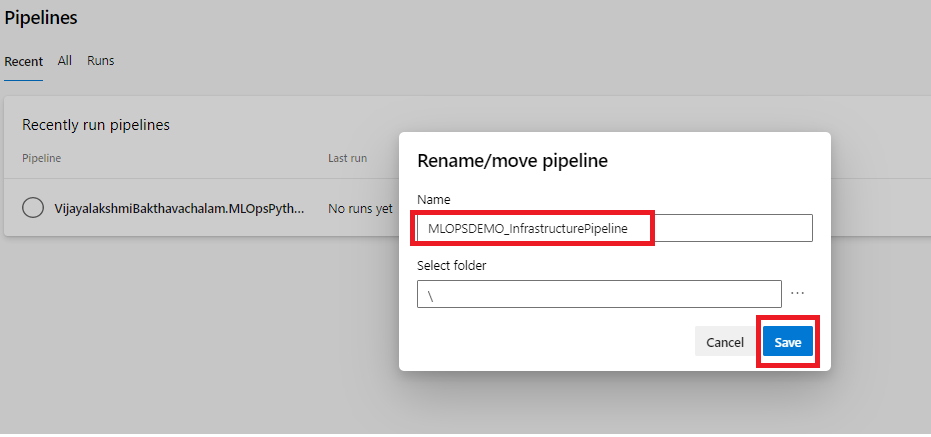


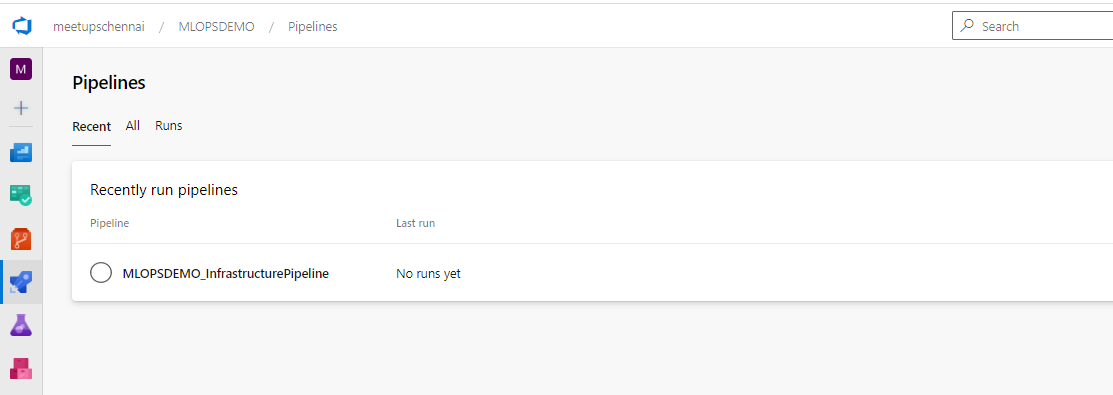




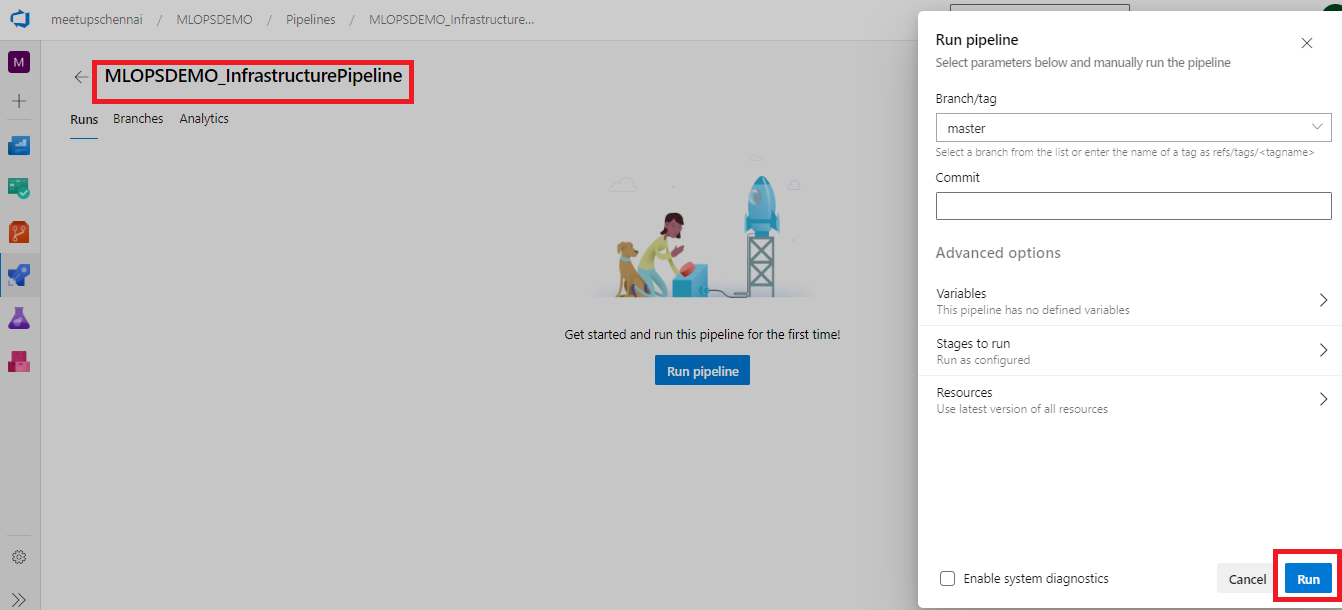


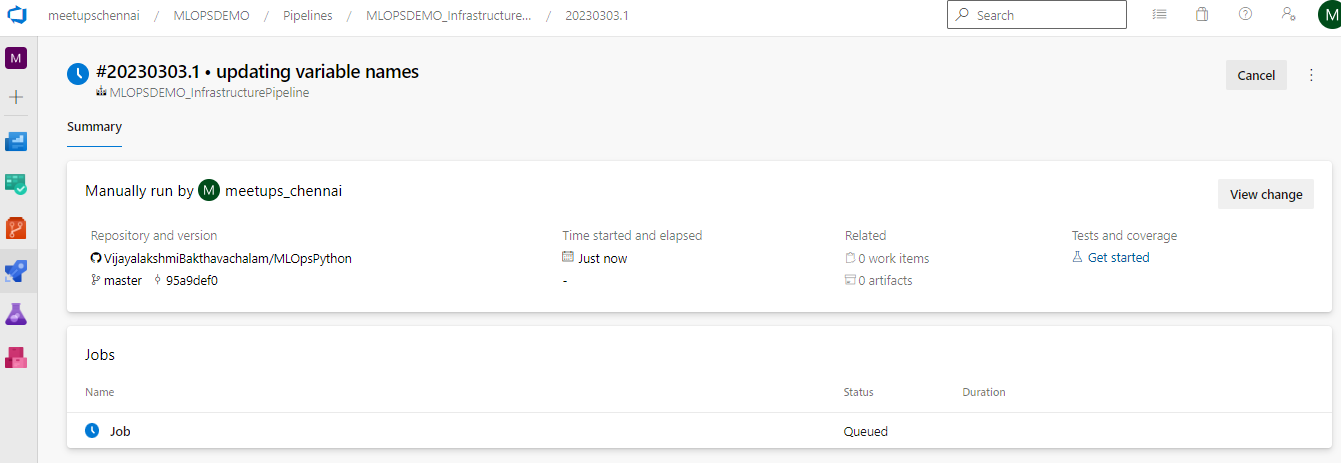


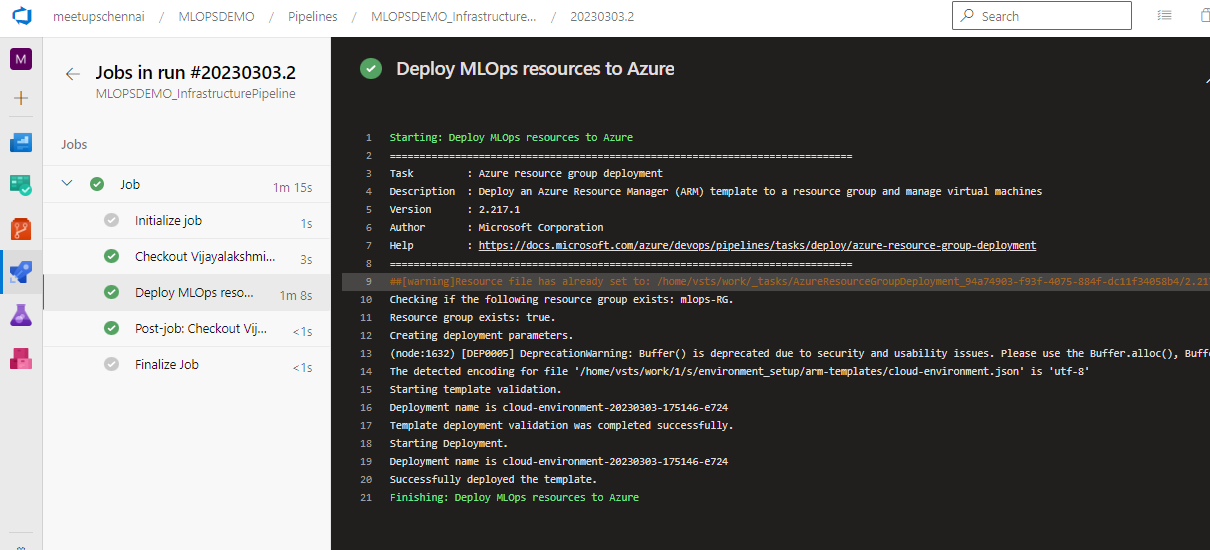




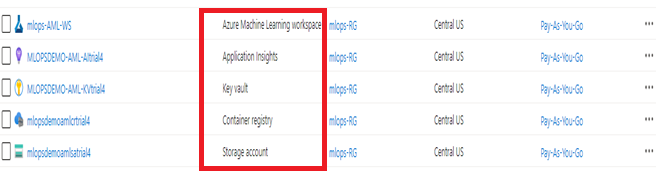
Having done that, run the pipeline: - this should create a resource group, a Key Vault, a Storage Account, a Container Registry, an Application Insights and a Machine Learning workspace.







Once the build job is successful, Check on the resources created via Azure portal

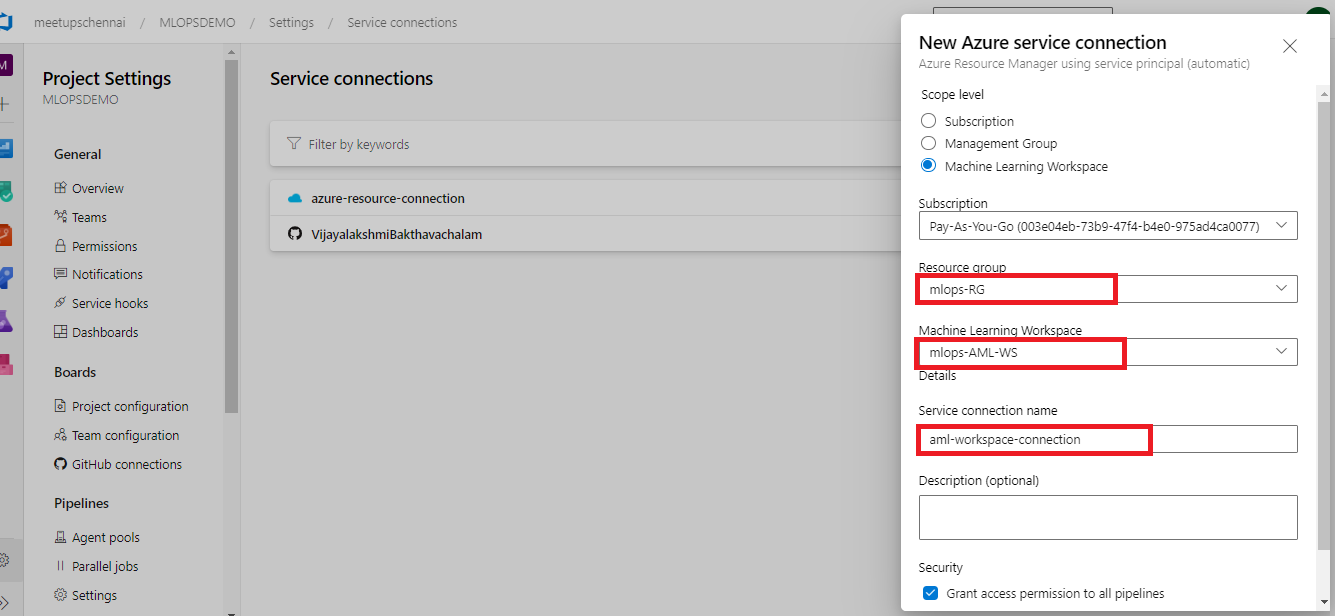


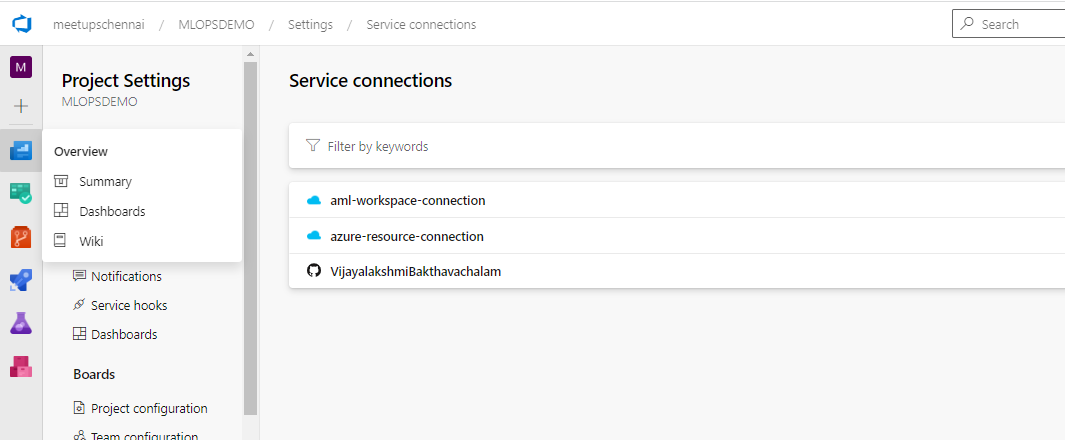
Now we have the infrastructure ready for our AI/ML app CI/CD Pipeline, before we move on to create the CI Pipeline using YAML file, let us create the Azure ML workspace Service Connection

**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](https://marketplace.visualstudio.com/items?itemName=ms-air-aiagility.vss-services-azureml) 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').





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

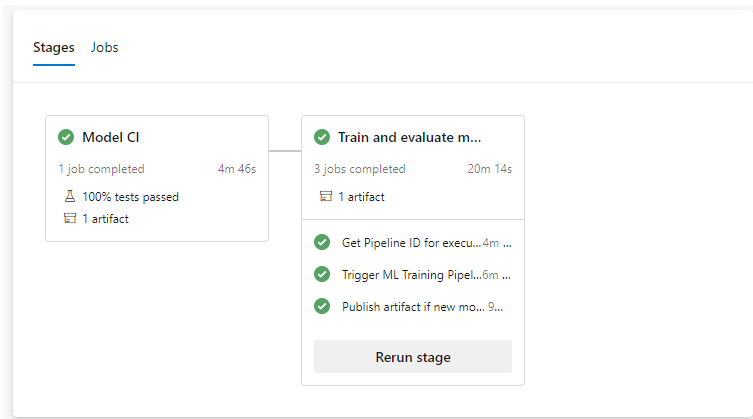
Now that you've provisioned all the required Azure resources and service connections, you can set up the pipelines for training (Continuous Integration - **CI**) and deploying (Continuous Deployment - **CD**) your machine learning model to production.

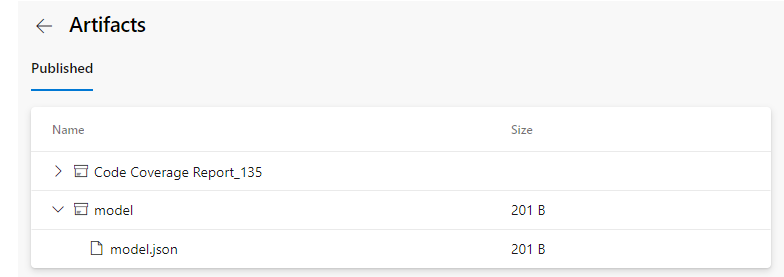
1. **Model CI, training, evaluation, and registration** - triggered on code changes to master branch on GitHub. Runs linting, unit tests, code coverage, and publishes and runs the training pipeline. If a new model is registered after evaluation, it creates a build artifact containing the JSON metadata of the model. Definition: diabetes\_regression-ci.yml.
2. **Release deployment** - consumes the artifact of the previous pipeline and deploys a model to either [Azure Container Instances (ACI)](https://azure.microsoft.com/en-us/services/container-instances/), [Azure Kubernetes Service (AKS)](https://azure.microsoft.com/en-us/services/kubernetes-service), or [Azure App Service](https://docs.microsoft.com/en-us/azure/machine-learning/service/how-to-deploy-app-service) environments

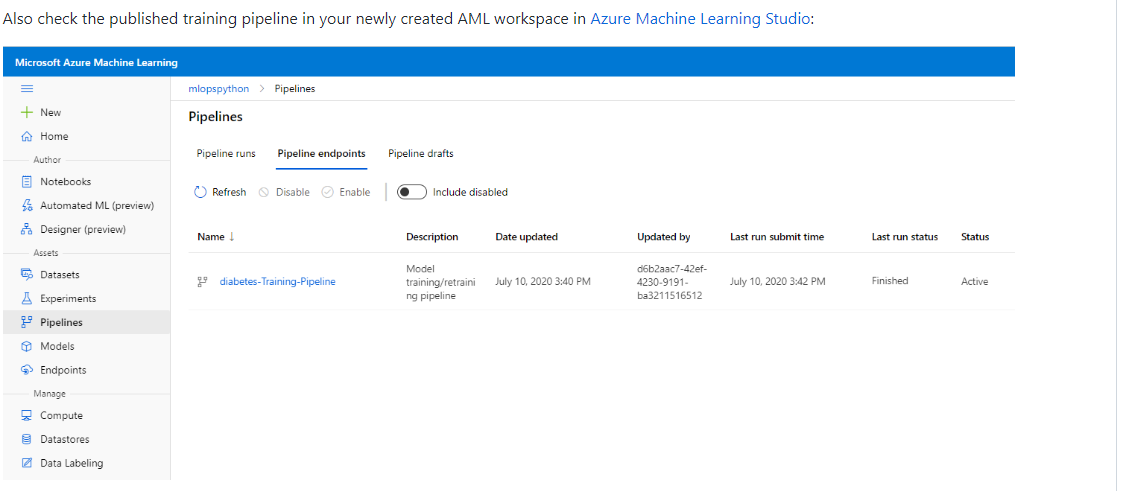
**Set up the Model CI, training, evaluation, and registration pipeline**

In your Azure DevOps project, create and run a new build pipeline based on the [./pipelines/diabetes\_regression-ci.yml](https://github.com/VijayalakshmiBakthavachalam/MLOpsPython/blob/master/.pipelines/diabetes_regression-ci.yml) pipeline definition from the code repository.

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:

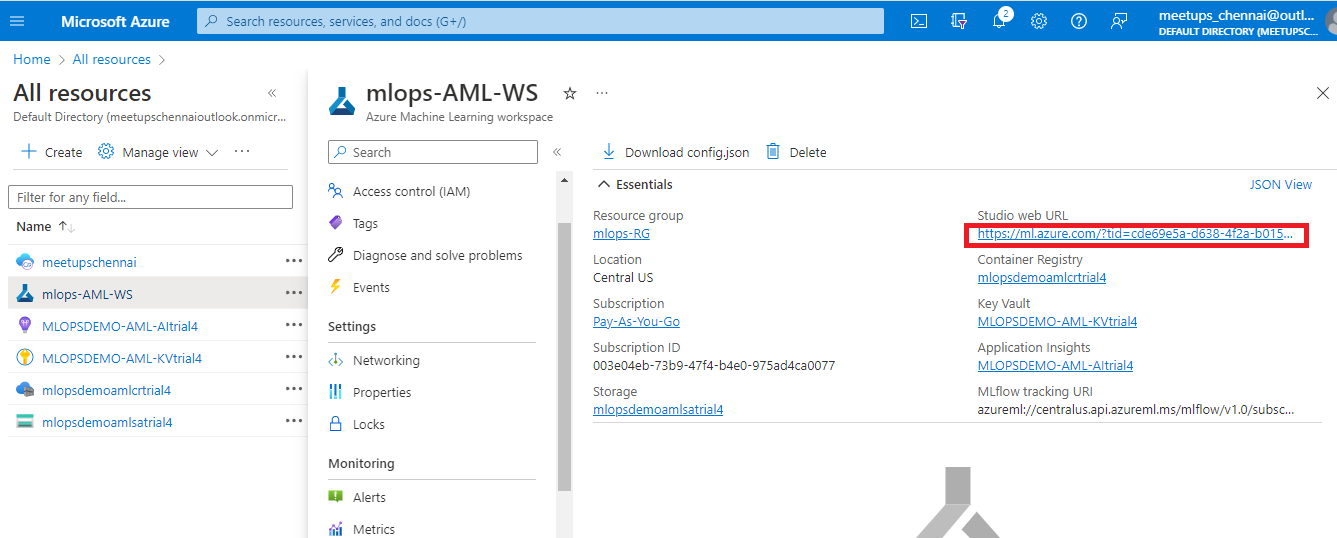


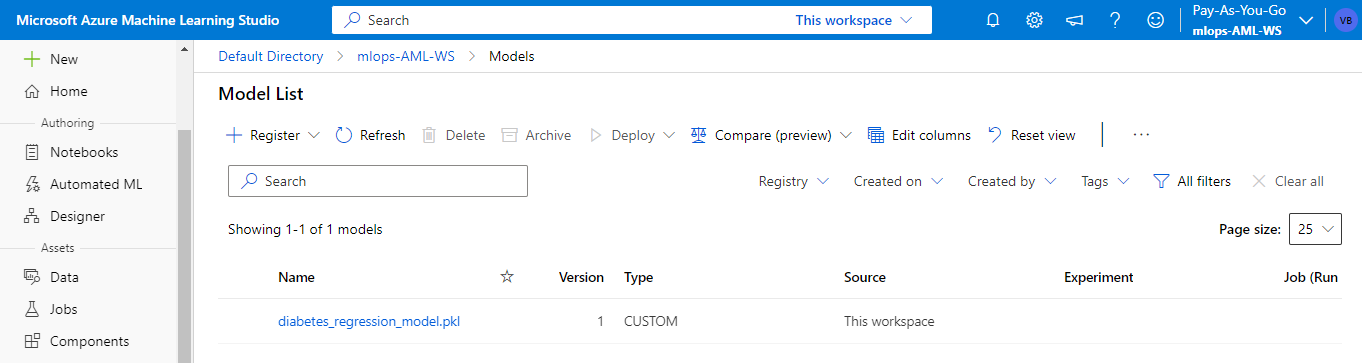




Great, you now have the build pipeline for training set up which automatically triggers every time there's a change in the master branch!

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





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 waits for it to complete.
  + 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.

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.

**Set up the Release Deployment**

**PRE-REQUISITES**

In order 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.

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

* 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. See the second screenshot below to verify where this number would be used.

**Set up the Release Deployment pipeline:**

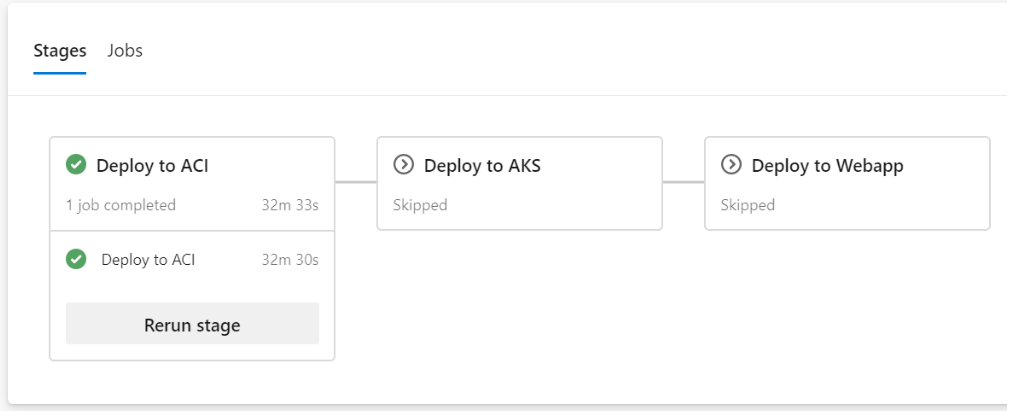
In your Azure DevOps project, create and run a new **build** pipeline based on the [./pipelines/diabetes\_regression-cd.yml](https://github.com/VijayalakshmiBakthavachalam/MLOpsPython/blob/master/.pipelines/diabetes_regression-cd.yml) pipeline definition from the repository.

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.

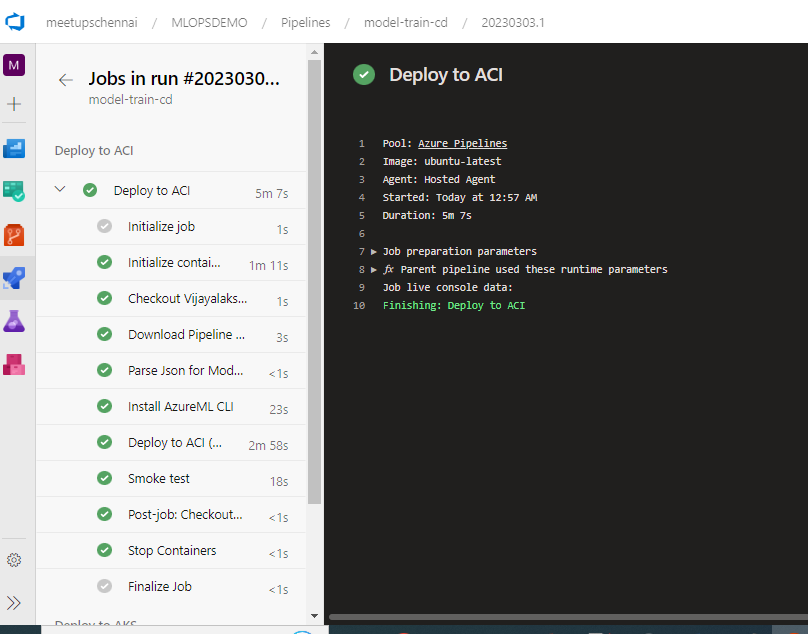
Once the pipeline is finished, check the execution result:

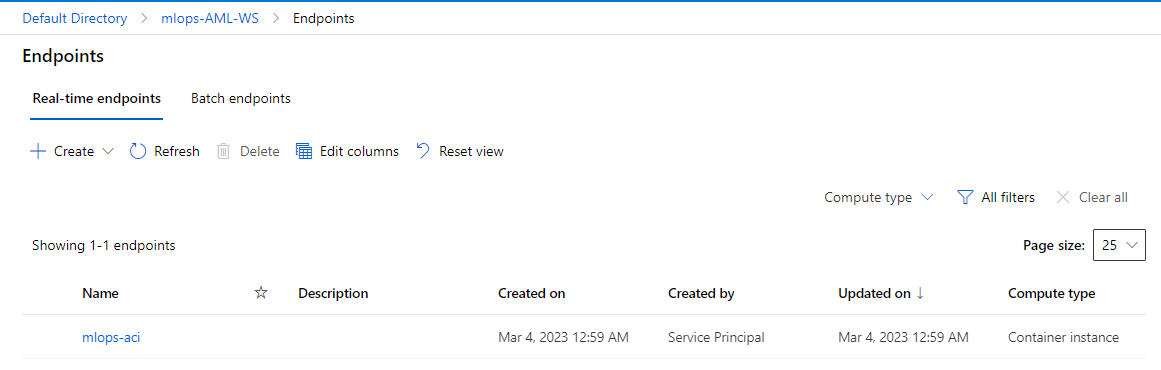


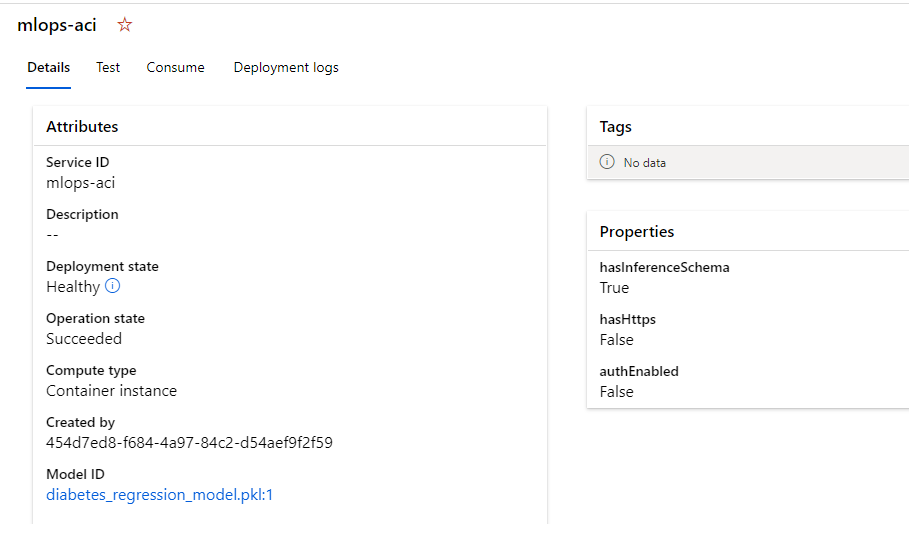
The pipeline has the following stage:

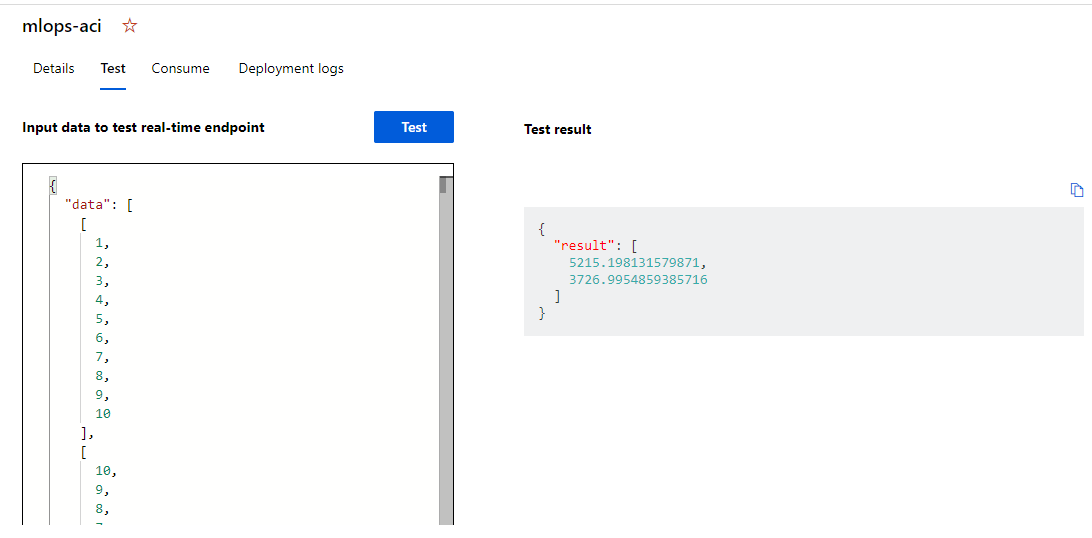
Deploy to ACI

* Deploy the model to the QA environment in Azure Container Instances.
* Smoke test
  + The test sends a sample query to the scoring web service and verifies that it returns the expected response. Have a look at the smoke test code for an example.
* You can verify that an ACI instance was created in the same resource group you specified:









Further exploration of deployment to AKS Clusters and Azure Webapp can as well be tried using the YAML scripts available.