ML & DevOps Workshop

Introduction

This lab uses Azure DevOps and Azure Machine Learning and will leverage the git repository here: https://github.com/KBaroni/mlopsworkshop2.

Objectives

This lab covers the following topics.

1. Setup and configure your MLOps environment to automatically execute Azure Machine Learning pipelines from Azure DevOps
2. Create three machine learning pipelines: one to create the azure resources, on to create the build and training pipelines, and one to deploy the AI model into QA and Production.

Prerequisites

The following are required to complete this hands-on lab:

- An active Microsoft Azure subscription with at least **contributor** role access. If you don't have one, sign up for a free trial.
- A **Service Principal** provisioned and assigned **contributor** role access to the Azure subscription. A service principal and credentials will be used for authentication within the AML pipelines.
- Create an **Azure Resource Group** in which you have **owner** level permissions
- Access to an **Azure DevOps account**. The build and release pipelines will be run using Azure DevOps. To verify you have an existing DevOps account, navigate to http://devops.azure.com and sign in. If you don't already have a DevOps account, you can create one by following the '*Start free*' prompt.
- **Document** the following parameters and values and bring them with you to the workshop:
    - Service principal name
    - Service principal application id
    - Service principal password
    - Service principal tenant id
    - Resource Group name

If you need step-by-step guidance on how to affect these pre-requisites, please see the *WorkshopPreReqs* reference in this repository: https://github.com/KBaroni/mlopsworkshop2.

Exercises

This hands-on lab includes the following exercises:

- Exercise 1: Download and unzip the workshop repo to your local machine
- Exercise 2: Create a new project in your Azure DevOps account
- Exercise 3: Create a service connection (Azure Subscription scope)
- Exercise 4: Store secrets in a variable group
- Exercise 5: Setup a repository in Azure DevOps

Estimated time to complete this lab: **120-160 mins.**

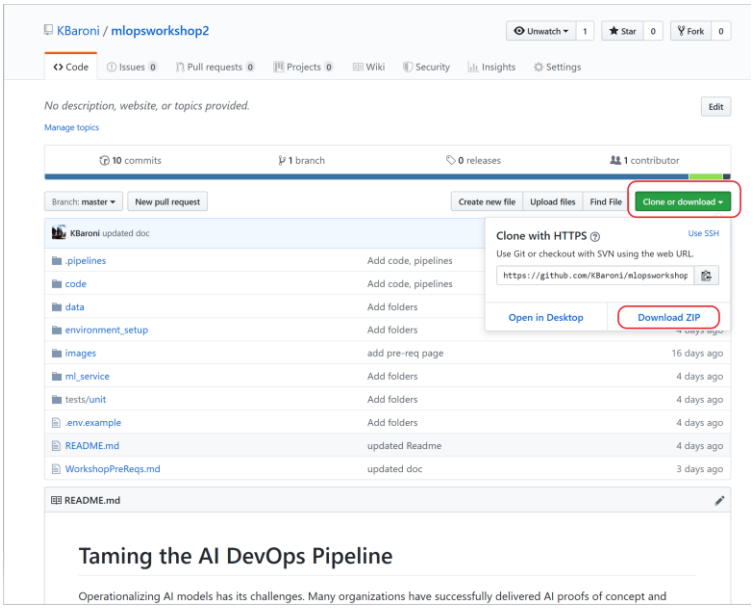
Exercise 1: Download and unzip the workshop repo to your local machine

In this exercise, you will download the contents of the workshop GitHub repo and unzip it to your local machine.

Caution: do not clone the GitHub repo, download the zip as shown below.

Content for this workshop is in a GitHub repo here: https://github.com/kbaroni/mlopsworkshop2 and includes the slides, documents, code, and scripts.

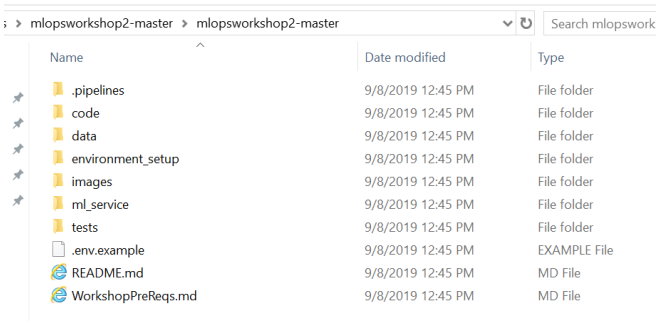
1. Navigate to the git repository and click on **Clone or download**, then click on **Download ZIP** and save the zip to your local machine:



GitHub repo for Workshop

2. Unzip and extract the files on your local machine

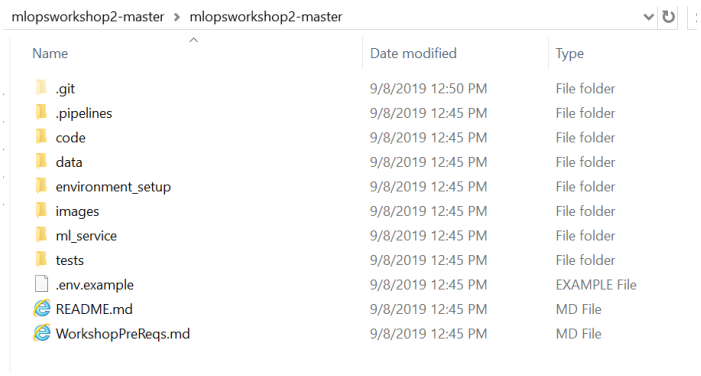
3. Bring up a command line window and change directory into the workshop folder **mopsworkshop2-master** that contains the code. *Note: There will be two levels of folders with the same name (mopsworkshop2-master), be sure to navigate to the folder that has the code below.*



4. On the command line, execute a series of git commands to initialize the directory as a version control project and prepare it to use with Azure DevOps:

```
git init          <to initialize the directory for version control>
git status        <to check the current status of the directory>
git add .         <will add all the files into the staging area>
git commit -m "Initial Build"      <creates a commit>
```

5. On Windows, the directory will now resemble the following and is ready to be pushed to Azure DevOps:



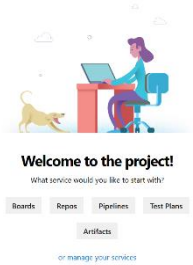
Exercise 2: Create a new project in your Azure DevOps account

In this exercise, you will setup an Azure DevOps project as the 1st step in setting up the pipelines.

1. Navigate to http://devops.azure.com
2. If there are multiple organizations, select the one you will be using to host this project. Alternatively, you can create a new organization and create your project in the new organization.
3. Create a project and name it

   Project name: <Your initials>_DevOps

   The project is available when a welcome screen appears similar to the below image.



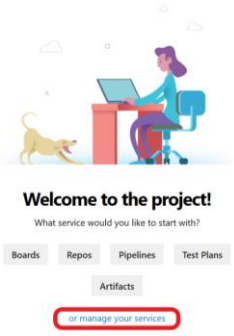
Exercise 3: Create a service connection (Azure subscription level)

To deploy the pipelines to Azure, you need to create an Azure Resource Manager service connection in Azure DevOps. To create the service connection, you need the information about the service principal required in this lab's pre-requisites. To complete this exercise, you need the following parameters:

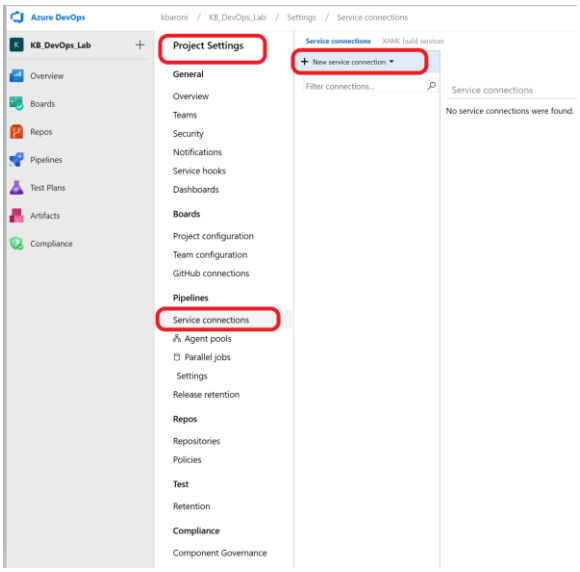
- Azure subscription ID
- Azure subscription name
- Service principal client ID
- Service principal password/key
- Tenant ID

To create a service connection:

1. Navigate to Project Settings by clicking on 'manage your services'



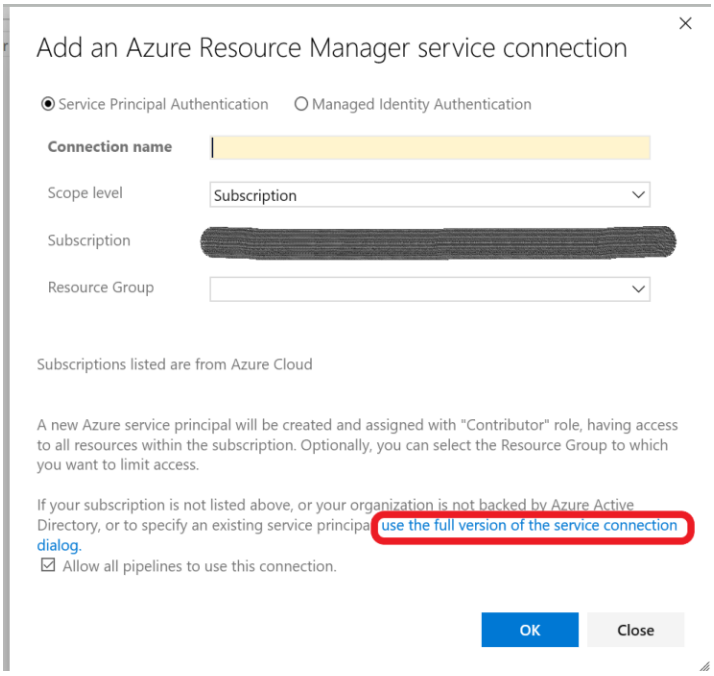
2. Under Project Settings, select **Service Connections** -> **New service connection** and select **Azure Resource Manager** from the drop down.



3. When the connection screen comes up, enter in the following parameters.

   Connection name: AzureResourceConnection
   Scope level: Subscription
   Subscription: Select the Azure subscription you will use for the lab
   Resource Group: (leave it blank)

And then click on '**use the full version of the service connection dialogue**' as shown below.

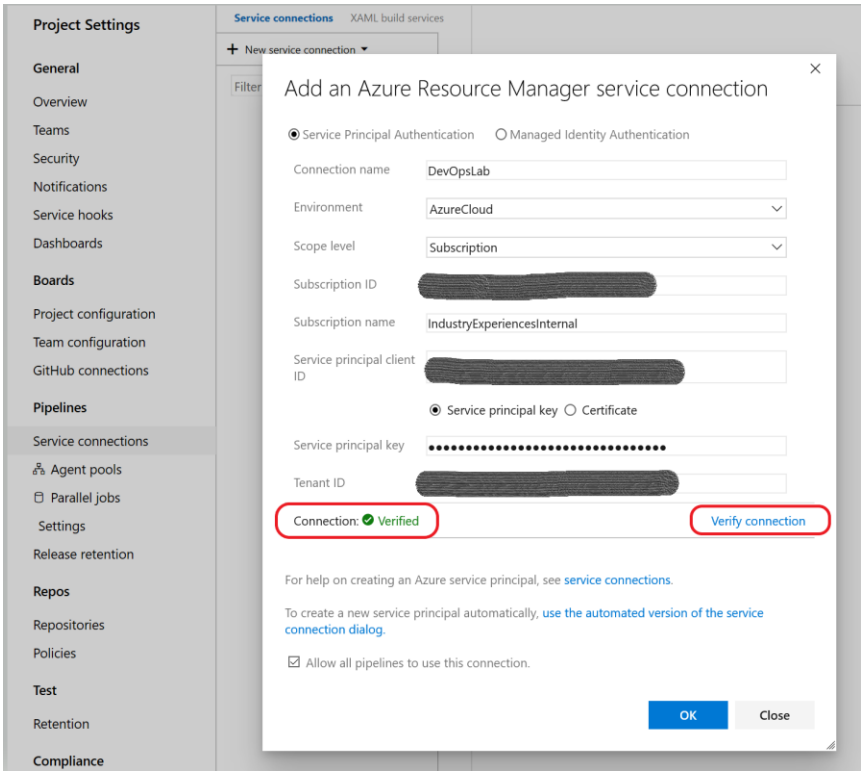


4. When the screen expands, add in the following parameters:

   Service principal client ID: <the service principal ID>

Service principal key: <the service principal password>
Tenant ID: this may auto-populate <your tenant ID>

5. Click on **Verify connection** to confirm a valid service connection. Your screen should look similar to the image below.



Service connection configuration is checked and verified

6. When the connection is verified, click on **OK**.

   If you encounter an error after entering valid values for the parameters above, it is likely the service principal has not been granted contributor level role at the subscription level. The subscription admin must do before you can proceed with the lab.

Exercise 4: Store secrets in a variable group

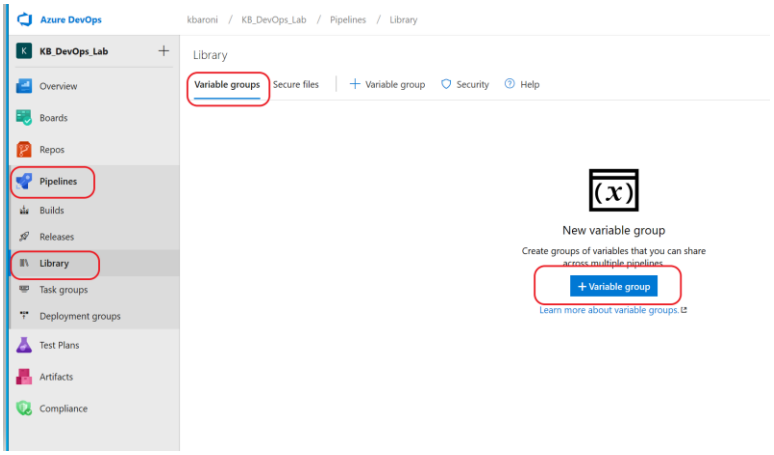
A variable group is a logical container that stores variables and their properties (like key/value pairs). They can contain API keys, database connection strings, or API endpoints. For this lab, we will create a variable group to store the credentials and secrets needed to access the Azure subscription and provision resources. This variable group will be shared across the pipelines. To complete this exercise, you need the following variables and values:

- Azure subscription ID
- Service principal client ID
- Service principal password/key

- Tenant ID
- The name of the Resource Group for which you have owner role

To create a variable group:

1. In your Azure DevOps project, navigate to the **Pipelines** icon on the left panel and select **Library** and click on **+Variable group** as shown below.



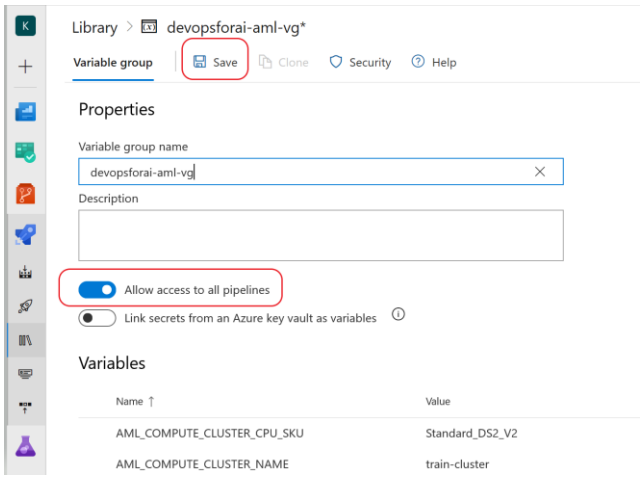
2. Name the variable group: **devopsforai-aml-vg** (this is hard coded in the build file)

3. Add each of the following variables to the variable group with their corresponding value:

Variable Name	Value
AML_COMPUTE_CLUSTER_CPU_SKU	Standard_DS2_V2
AML_COMPUTE_CLUSTER_NAME	train-cluster
BASE_NAME	<unique name – for instance <your initials> + ops (eg: kbops)
AML_WORKSPACE_NAME	<BASE_NAME>-AML-WS
EVALUATE_SCRIPT_PATH	evaluate/evaluate_model.py
EXPERIMENT_NAME	mlopspython
LOCATION	centralus
MODEL_NAME	sklearn_regression_model.pkl
REGISTER_SCRIPT_PATH	register/register_model.py
RESOURCE_GROUP	[resource group in which you have owner perms]
SOURCES_DIR_TRAIN	code
SP_APP_ID	<your service principal id>
SP_APP_SECRET	<your service principal key/password>*
SUBSCRIPTION_ID	<your azure subscription id>
TENANT_ID	<your tenant id>
TRAIN_SCRIPT_PATH	training/train.py

*Mark the SP_APP_SECRET variable as a secret by clicking the lock icon to the left of your variable

Make sure the variable group is selected to allow access to all pipelines:



BE SURE to click **SAVE** before navigating away from the Variable group page.

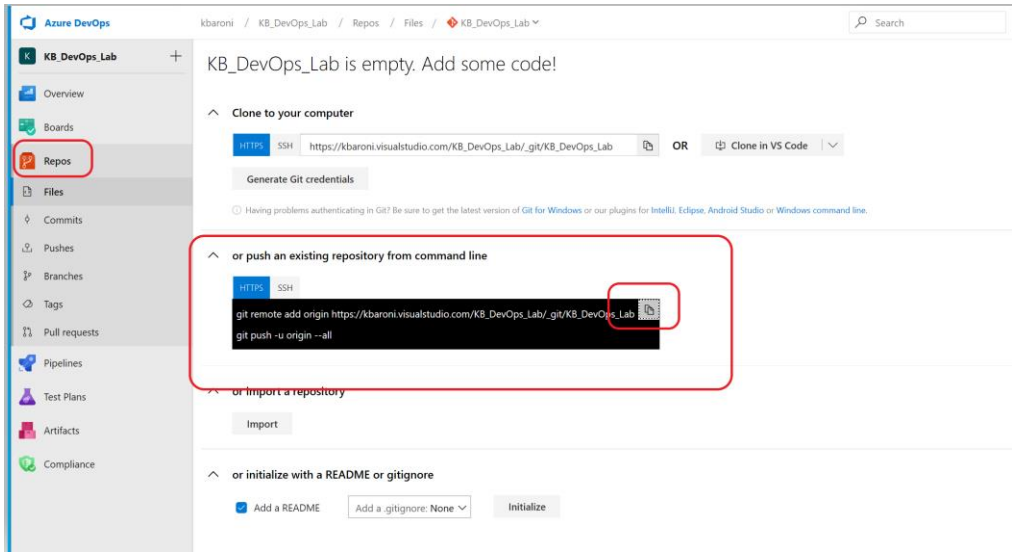
Exercise 5: Setup a repository in Azure DevOps

The DevOps project reads from a git repository. In this exercise we will push the repo we created earlier on our laptops to the repo in this Azure DevOps project.

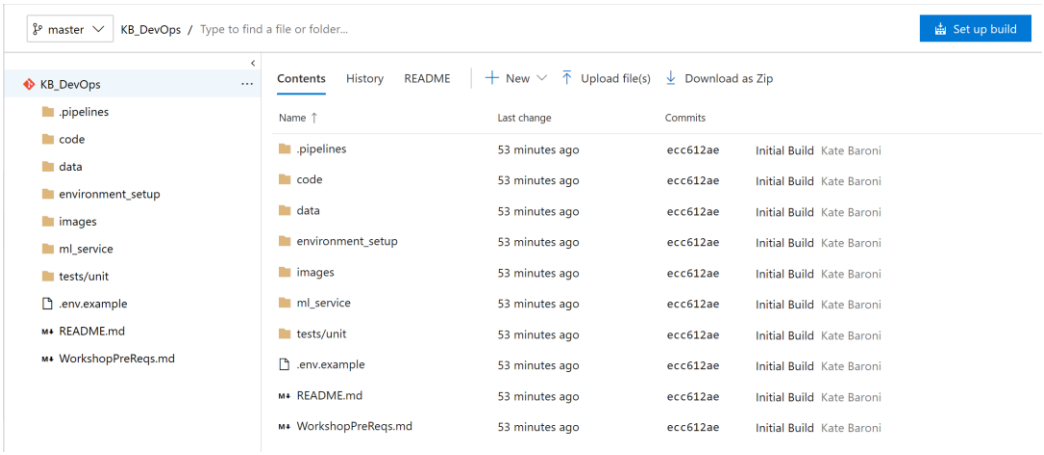
1. Navigate to the **Repos** icon on the left pane.

2. Copy the git commands under the section **push an existing repository from command line**.

3. On your laptop, navigate to the directory where you prepared the workshop repo and issue the 2 commands you copied from the DevOps Repos page as shown in the image below.



4. Navigate back to your DevOps project and Select **Repos** from the left pane to confirm the code and scripts have been uploaded as shown below.

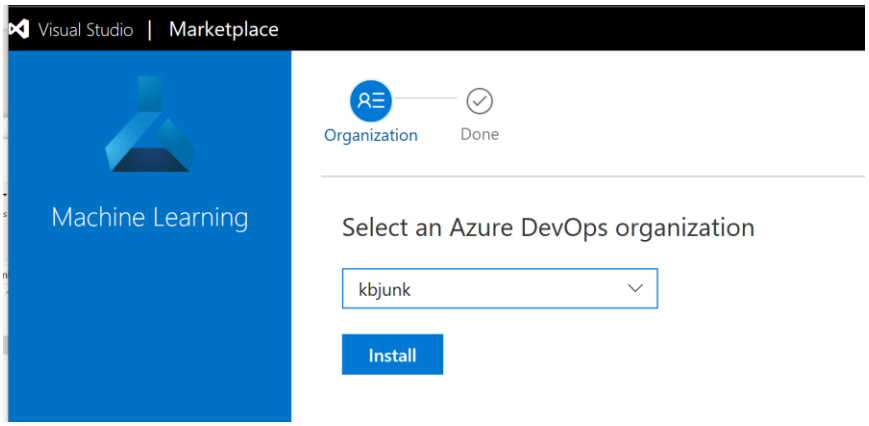


Exercise 6: Add Azure Machine Learning extension to Azure DevOps

The pipelines in this lab leverage the Azure Machine Learning extension for Azure DevOps. This extension enables integration with artifacts that are created in your AzureML Workspace. With this extension installed, the AzureML Workspace artifacts can be referenced in Azure Devops and used to trigger pipeline activity (like model training or deployment).

Install the extension directly into your Azure DevOps organization from the Azure Marketplace here: https://marketplace.visualstudio.com/items?itemName=ms-air-aiagility.vss-services-azureml

When you click on 'Get it free', you will be prompted to select an Azure DevOps organization from the drop down. Select the organization you are using for this lab and click **Install**:

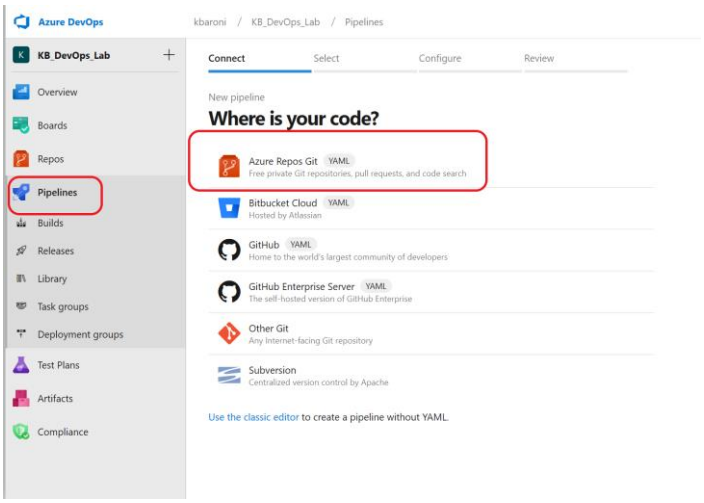


With this exercise completed, you are ready to setup the build and release pipelines.

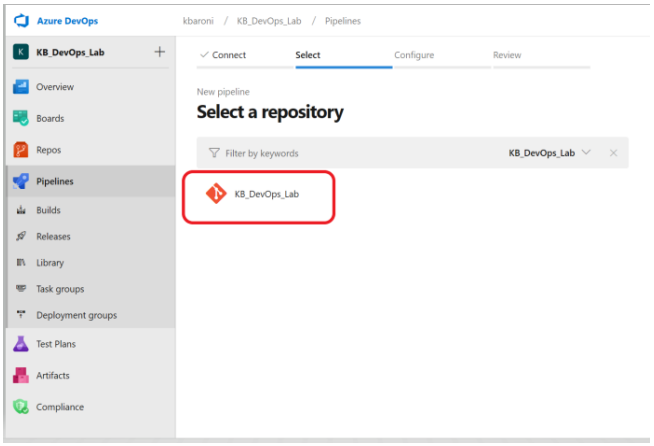
Exercise 7: 1st Pipeline: Provision the Azure resources

The 1st pipeline will provision the Azure resources and infrastructure needed to execute the remaining pipelines. To do that we will be using the *service connection* configured in a previous step and executing a YAML pipeline that already exists. To get started:

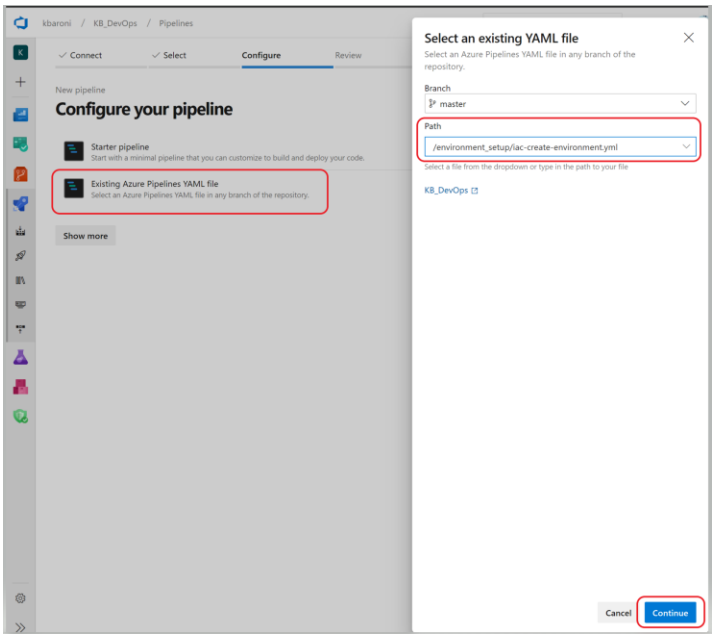
1. Select **Pipelines** from the left pane and click on **New pipeline** and you will see a screen similar to the one below. Select **Azure Repos Git** which will display the repo you associated with this project:



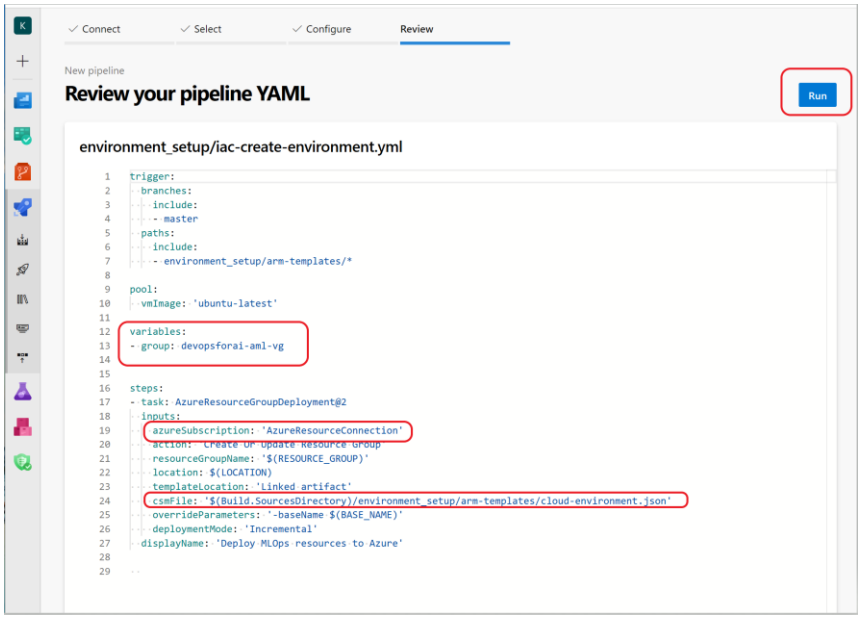
2. Select the repo you pushed to this project in an earlier step:



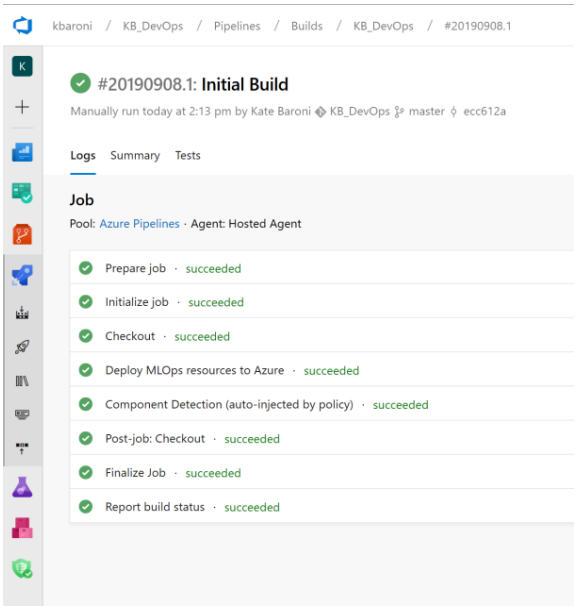
3. Choose '*Existing Azure Pipelines YAML file*' and select */environment_setup/iac-create-environment.yml* from the **Path** drop-down box and click **Continue:**



4. When you click **Continue**, the YAML file will be loaded and you can review it to see that it refers to the *variable group* and the *service connection* we just created. Scroll through to see the json file that will execute the ARM templates that will be used to provision the infrastructure.



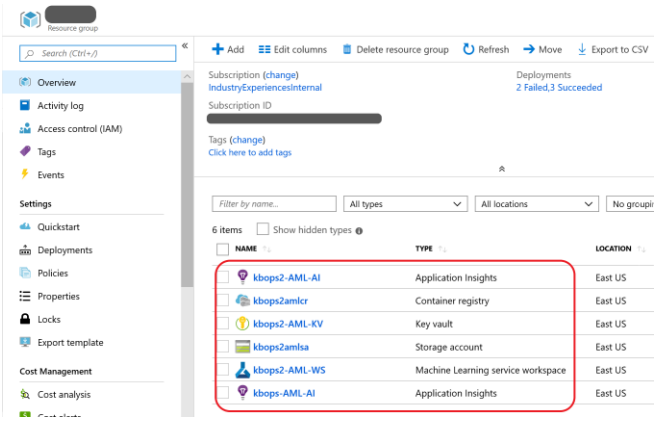
When you are ready, click **Run** to manually execute the pipeline. As they are executed, the status of the tasks will visible on your console:



Provisioning will take 5-10 minutes to complete and you will get an email confirming the success (or failure) of the pipeline:

Azure DevOps Notifications
[Build succeeded] KB_DevOps_Lab2 - KB_DevOps_Lab2... **7:36 PM**
[Build succeeded] KB_DevOps_Lab2 -

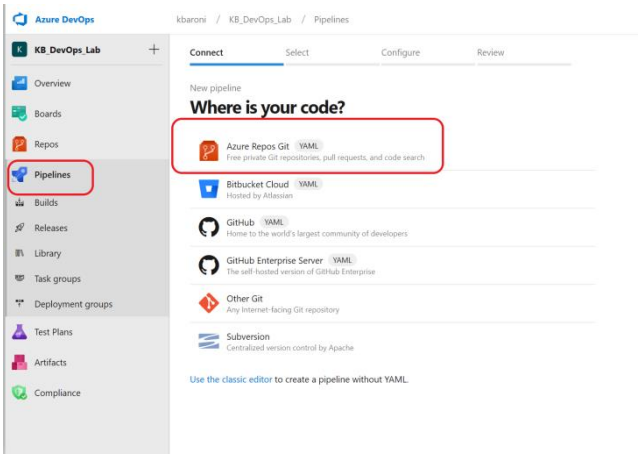
At this point, you can navigate to your Azure subscription and see the resources that have been provisioned under your Resource Group:



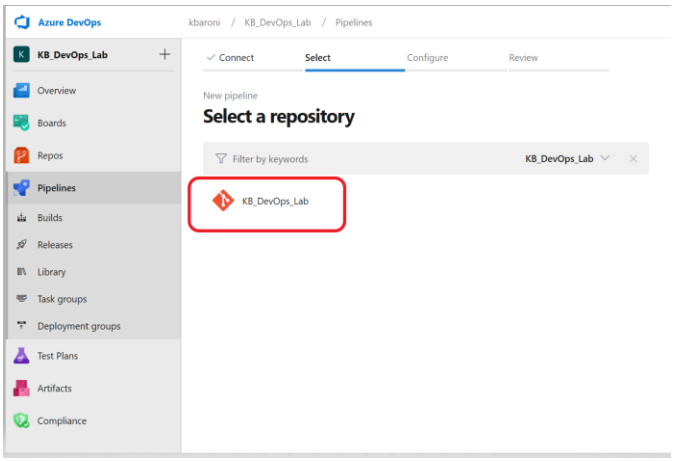
Exercise 8: 2nd Pipeline: Setup and run the Build Pipeline

This pipeline defines the steps to set up the build process and create the AzureML Training Pipeline that will be used in subsequent pipelines. In this exercise, you will create and run a build pipeline that configures an Azure ML training pipeline in Azure ML Services Workspace. It uses an existing YAML file named *azdo-ci-build-train.yml*. To get started:

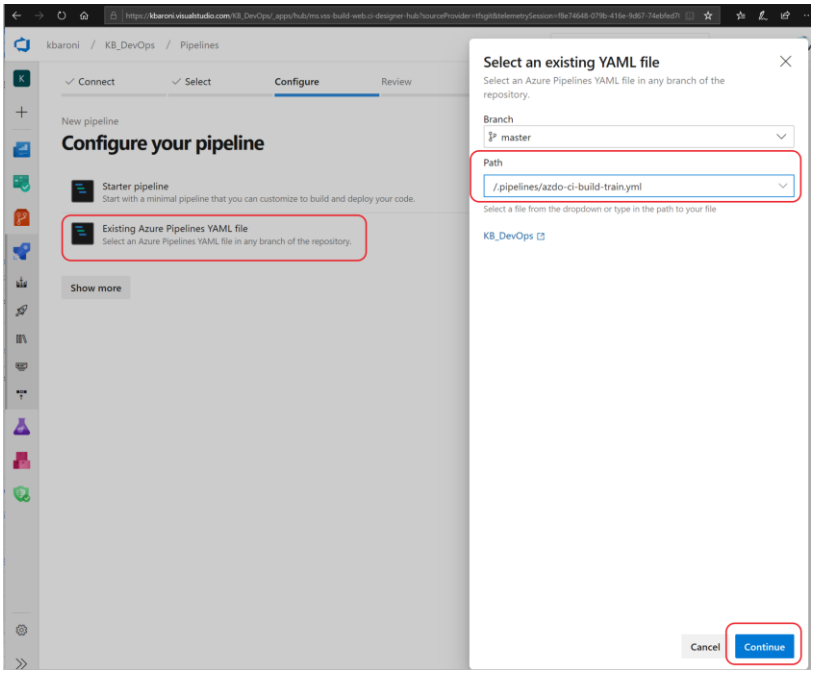
1. Select **Pipelines** from the left pane and click on **New pipeline** and you will see a screen similar to the one below. Select **Azure Repos Git** which will display the repo you associated with this project:



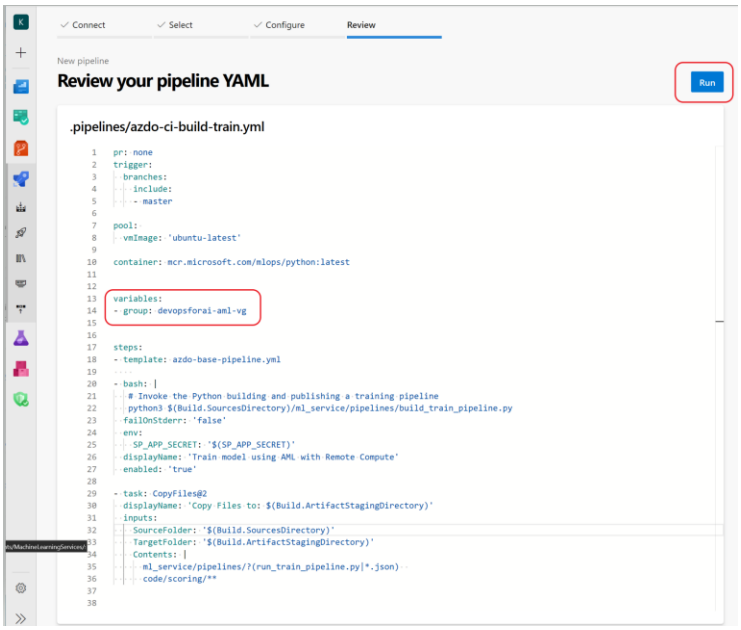
2. Select the repo you pushed to this project in an earlier step:



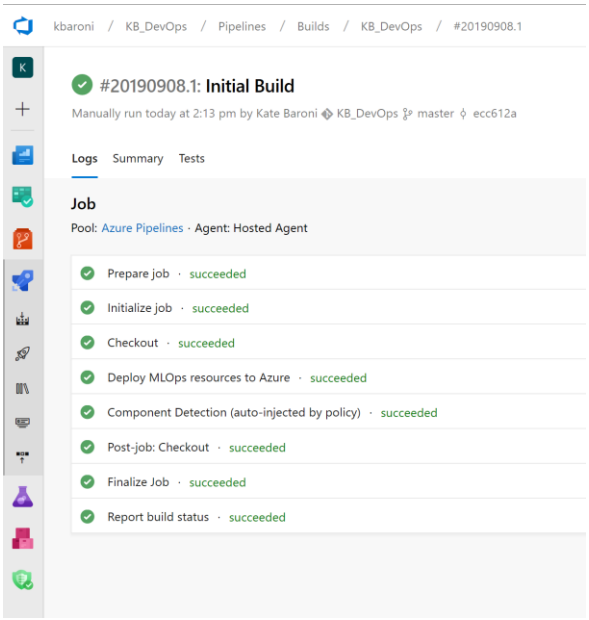
3. Configure the pipeline by selecting '*Existing Azure Pipelines YAML file*' and in the PATH drop down, navigating to the YAML file named: */.pipelines/azdo-ci-build-train.ym.*). Click **Continue**.



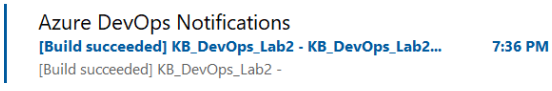
4. When you click **Continue**, the YAML file will be loaded and you can review it to see that it refers to the *variable group* we just created, will execute a python script (*run_train_pipeline.py*) to initiate the AzureML Training Pipeline, and copy all the build artifacts to a staging directory.



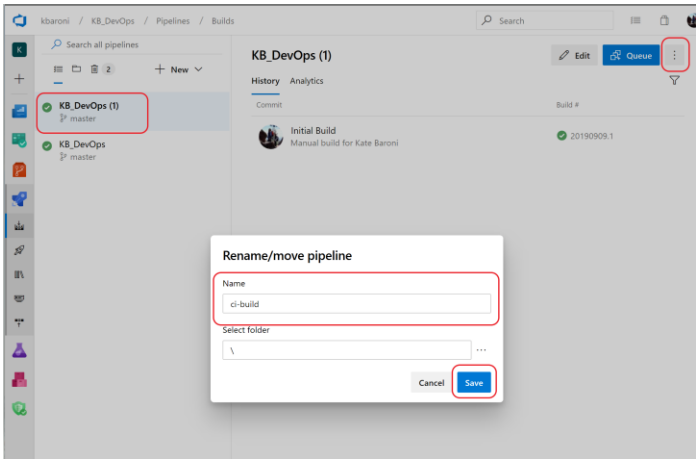
5. When you are ready, click **Run** to manually execute the pipeline. While the build is executing, the status of the steps will be posted to your window: Notice the pipeline does a quality check on the data, initializes a container, runs unit tests, trains and publishes an ML model, and reports the build status as the build pipeline executes through the steps. One of the steps incudes **Component Governance Detection** which will scan the build for vulnerable components and report findings.



The training process will take a few minutes to complete. Once it does, you will get an email confirming the build success in your inbox, similar to the image below.



6. Once the build has completed successfully, rename the pipeline to ci-build. Navigate back to the Build screen, highlight the pipeline you just executed, and click on the ellipses in the upper right corner. Select **Rename/Move** and when the dialogue box pops up, change the pipeline name to **ci-build** and **Save**.



The next step is to create the 2nd service connection that will be used in the subsequent pipelines to authenticate to Azure Machine Learning Workspace.

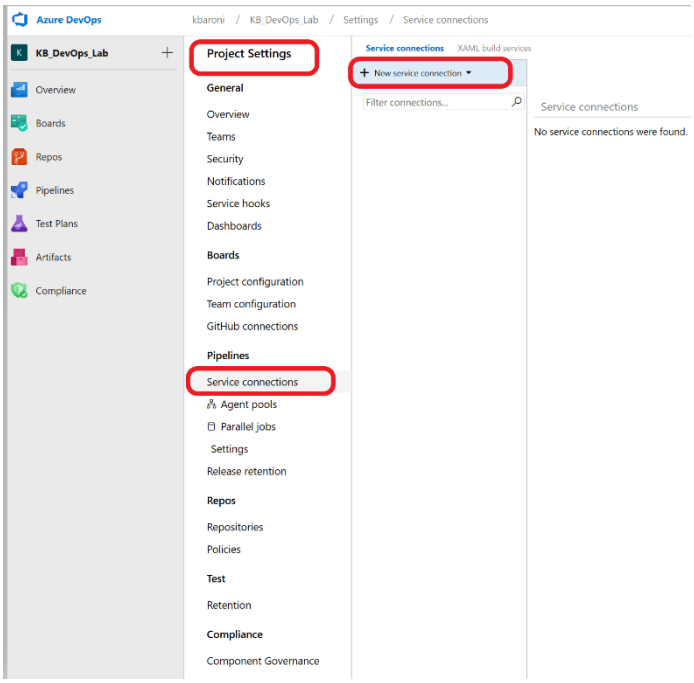
Exercise 9: Create a service connection (AzureML Workspace scope)

To reference artifacts in Azure Machine Learning Services, you need to create another Azure Resource Manager service connection in Azure DevOps but this time it needs to be scoped specifically to the Azure Machine Learning Workspace. To create the service connection, you need:

- the name of the *Resource Group* in which you are designated as an owner
- the name of the Azure Machine Learning Workspace that was created in the Build exercise (if you followed the suggestion while setting up the Variable Group, the name of the workspace will be <base name>-AML-WS.

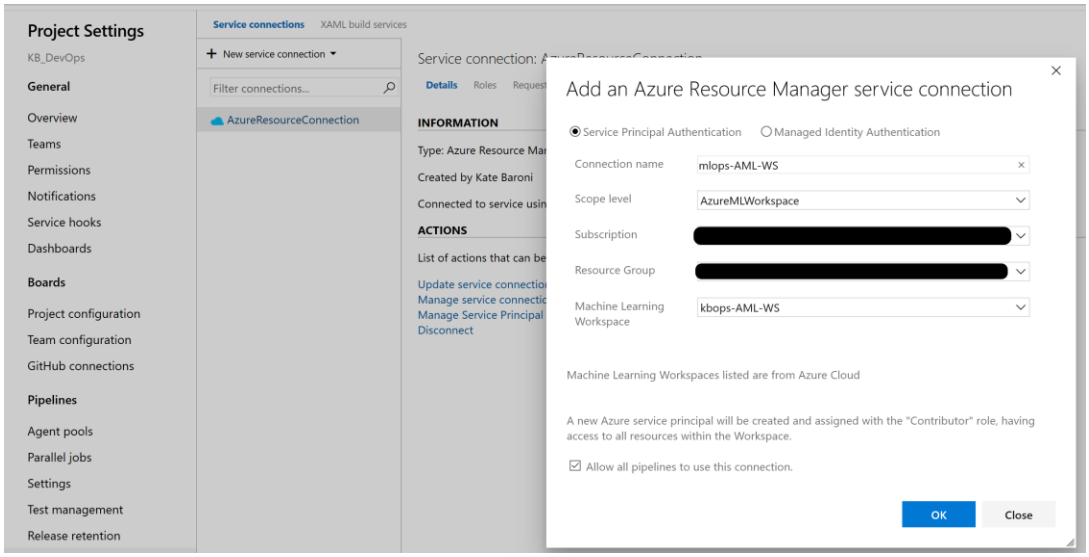
To create a 2nd service connection:

1. Navigate to Project Settings -> Service Connections -> Select **New service connection** and select **Azure Resource Manager** from the drop down.



2. When the connection screen comes up, enter in the following parameters.

   - Connection name: mlops-AML-WS
   - Scope level: AzureMLWorkspace
   - Subscription: Select the Azure subscription you will use for the lab
   - Resource Group: From the drop-down, select the name of the Resource Group in which you are an owner
   - Machine Learning Workspace: From the drop-down, select the name of the workspace



3. Click on **OK**.

If you encounter errors, it is likely you do not have owner permissions for the resource group that you have specified. The Azure subscription admin must grant you owner permissions before you can proceed with the lab.

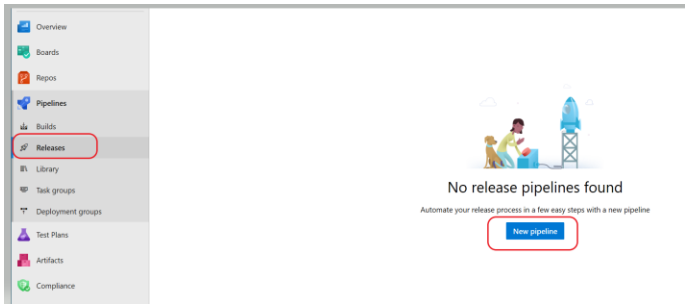
The next step is to create the model training pipeline that will be triggered every time a successful build executes.

Exercise 10: 3rd Pipeline: Setup the Model Training Pipeline

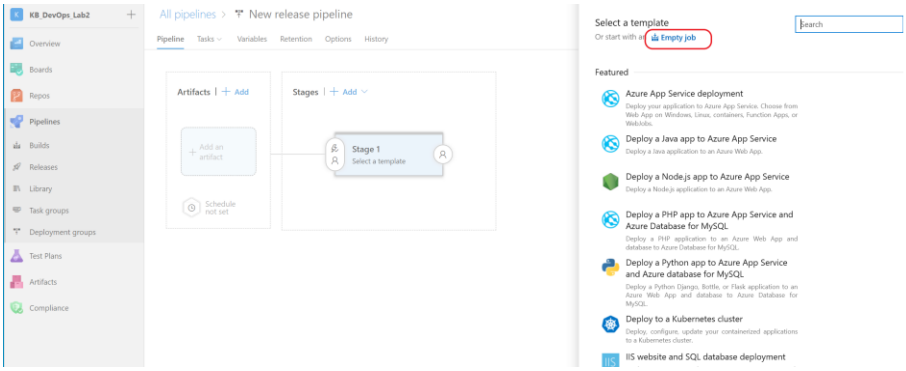
In this exercise you will invoke the Azure ML training pipeline that was created in the previous steps. This is a release pipeline that will use the build artifacts to deploy a model to a target platform. Whenever a new build and training pipeline is published, this model training pipeline will be triggered. This pipeline will initiate the execution of the training pipeline in AzureML Workspace. If this step executes successfully, it means the pipeline is running in AzureML Workspaces but may not have completed yet.

To get started with the model training pipeline:

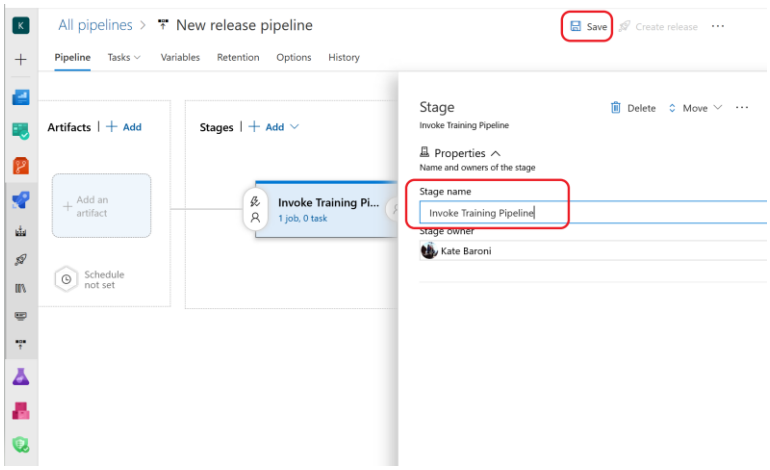
1. Navigate to the 'Releases' icon on the left panel and click 'New pipeline'.



2. Select **Empty Job** instead of selecting a template as shown below.



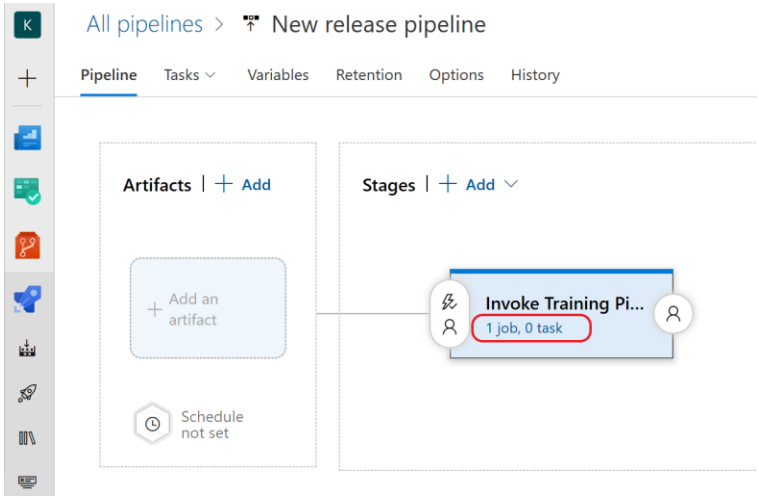
3. On the next screen, change the name from 'Stage 1' to 'Invoke Training Pipeline', select **Save** to save the empty job and then **OK**.



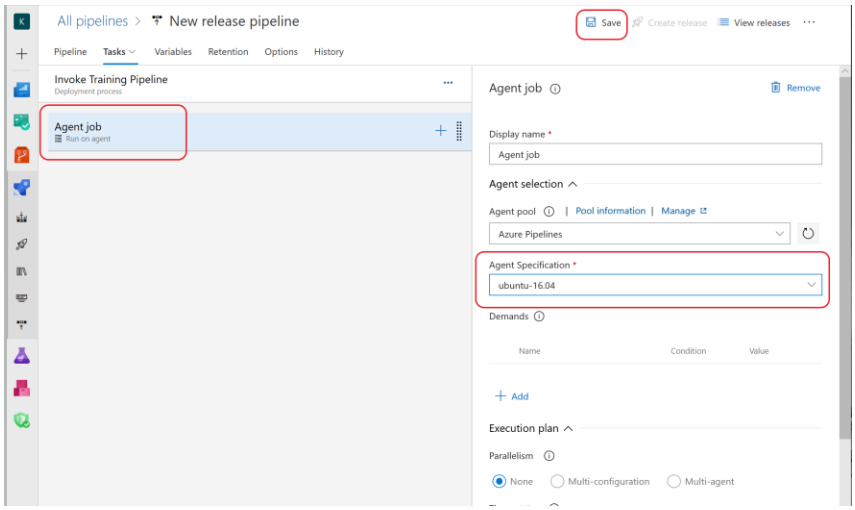
The next step is to configure the pipeline.

4. To build or deploy a release pipeline in Azure DevOps, you need at add at least one agent to run the job. The agent job will be defined under stages and will be configured to run as a hosted agent and each time the pipeline is run, you will get a fresh virtual machine to run the job. In this step you specify the pool from which the agent job will run.

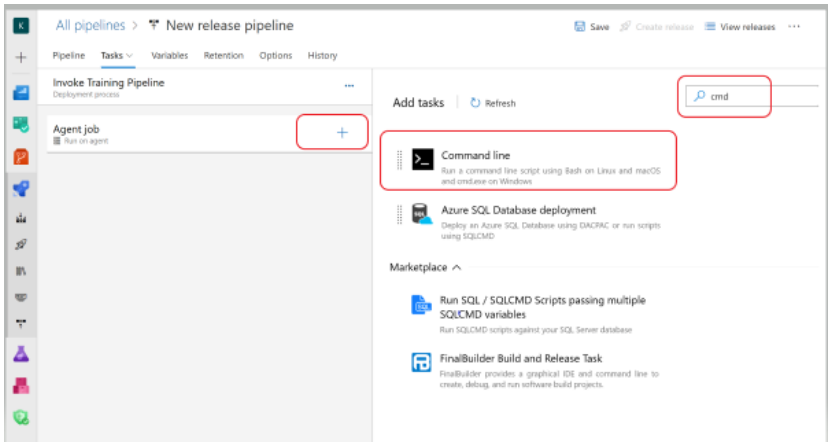
   Click on the link under the name of the pipeline to bring up the screen to configure the agent job.



Click on **Agent job** to bring up the settings and select **Hosted Ubuntu-16.04** from the Agent Specification drop-down**.** Click **Save** and **OK**.

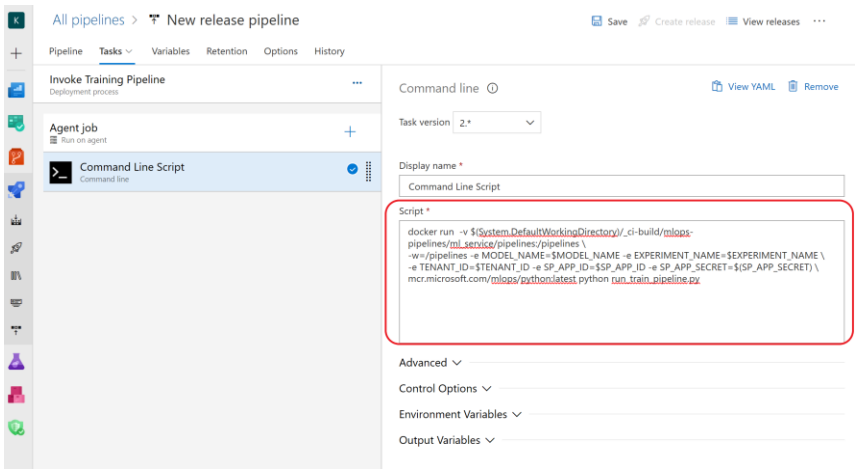


5. Add a *command line* task to the Agent job. The *command line task* will execute docker and Initiate the *Azure ML Training pipeline*. Click on + to add a task to the Agent Job and search for **cmd.** Add the **Command Line** template when it appears in the search results:



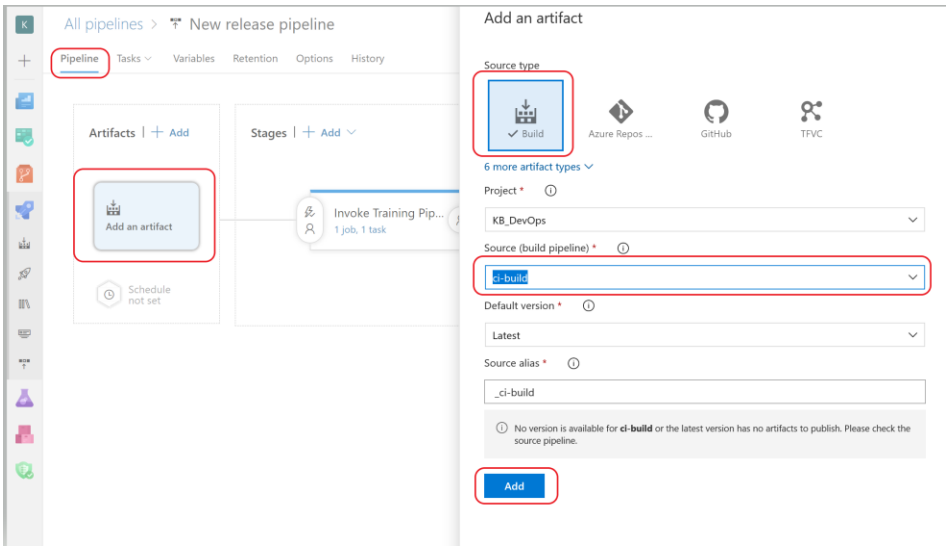
Configure the Command Line Script by clicking on it and entering the following block of code into the Script Box:

```
docker run  -v $(System.DefaultWorkingDirectory)/_ci-build/mlops-pipelines/ml_service/pipelines:/pipelines \

-w=/pipelines -e MODEL_NAME=$MODEL_NAME -e EXPERIMENT_NAME=$EXPERIMENT_NAME \

-e TENANT_ID=$TENANT_ID -e SP_APP_ID=$SP_APP_ID -e SP_APP_SECRET=$(SP_APP_SECRET) \
mcr.microsoft.com/mlops/python:latest python run_train_pipeline.py
```



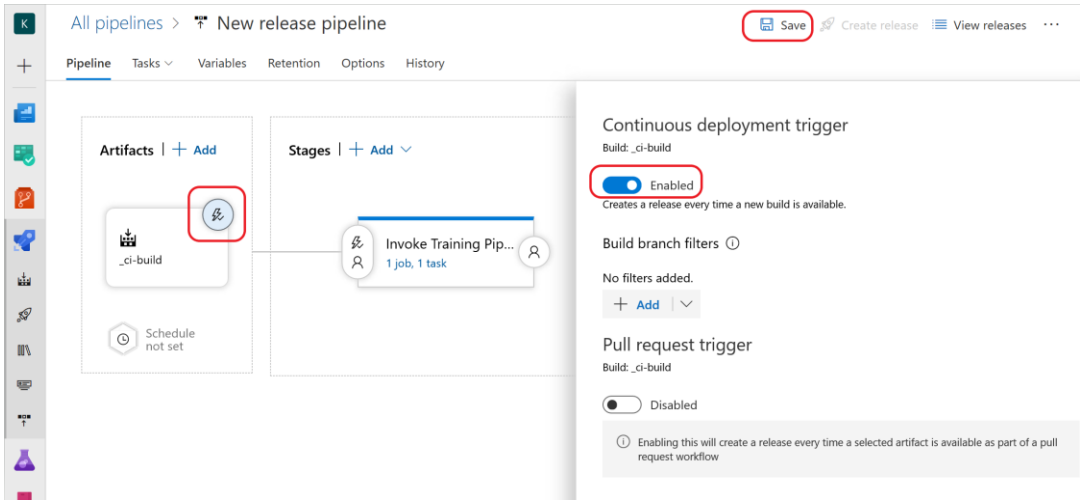

```
Click Save and OK to complete the steps for configuring the Agent Job.
```

6. To add the Build artifacts to the pipeline, navigate back to the **Pipelines** tab and click on '**Add an Artifact'.** Select the **Build** artifact, select the **ci-build** pipeline from the Source drop down. Click **Add** to add the artifact to your Training pipeline.



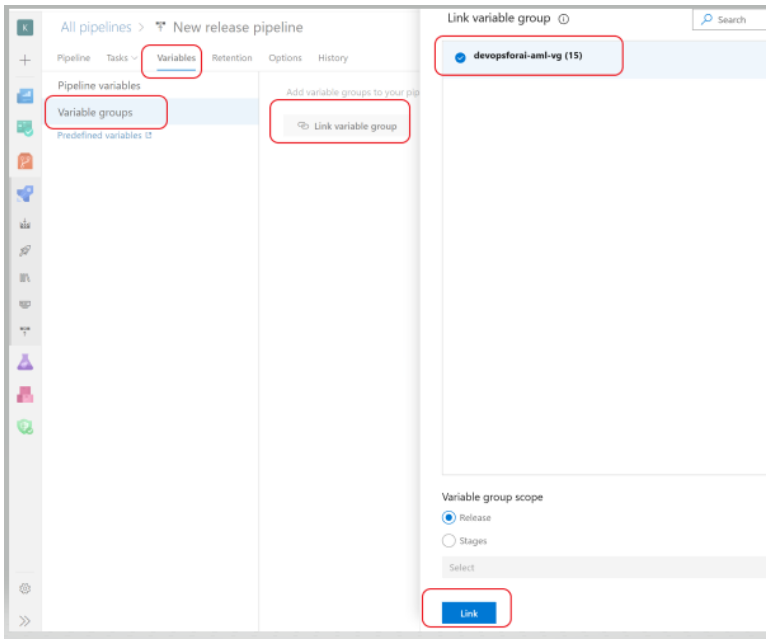
7. Click on **Add** and then **Save** the pipeline:

8. To trigger this pipeline every time a build is executed, enable the *Continuous Deployment Trigger* by clicking on the lightening sign, and sliding the bar for **Continuous Deployment** to **Enabled**. Click **Save** and then **OK.**

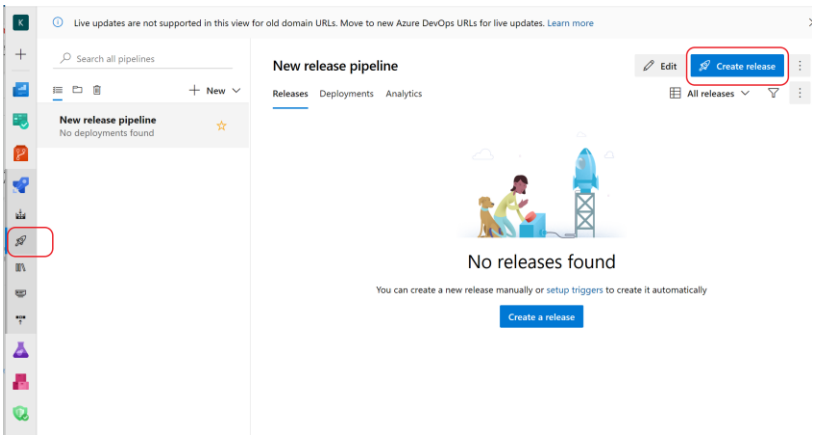


Note: Although we won't be doing it as part of this lab, to schedule a pipeline you would click on **Schedule set.**

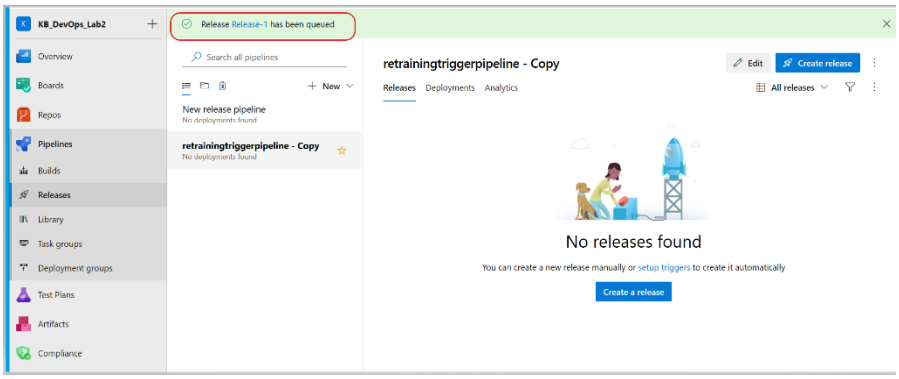
9. This pipeline needs access to the credentials and IDs that are stored in the variables group. Click on the **Variables** tab, then click on **Variable groups** and select **Link variable group**. Select the variable group created earlier (devopsforai-aml-vg), click on **Link**, and click on **Save** and then **OK**.



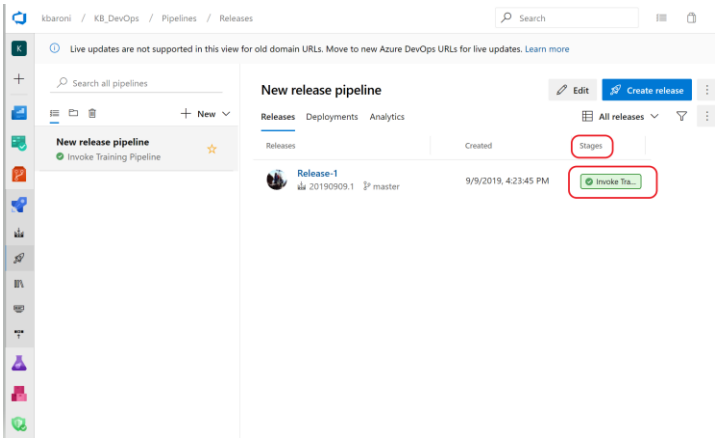
10. To verify the pipeline works as expected, initially you will manually trigger it. Navigate to the left pane and select **Releases.** Click on the training pipeline we just created and then click on **Create a release** as shown below**.**



11. On the next screen, click **Create** to initiate a manual release. Notice in the image below the release has been queued. Your screen should look similar.

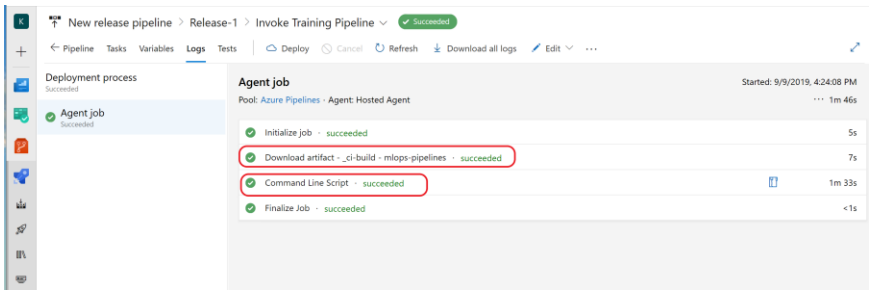


It will take a few minutes for the training pipeline to execute and when it is completed, it will show up under **Stages.** A successful execution will be highlighted in green and a failed execution will be highlighted in red. The following image shows a successful execution.

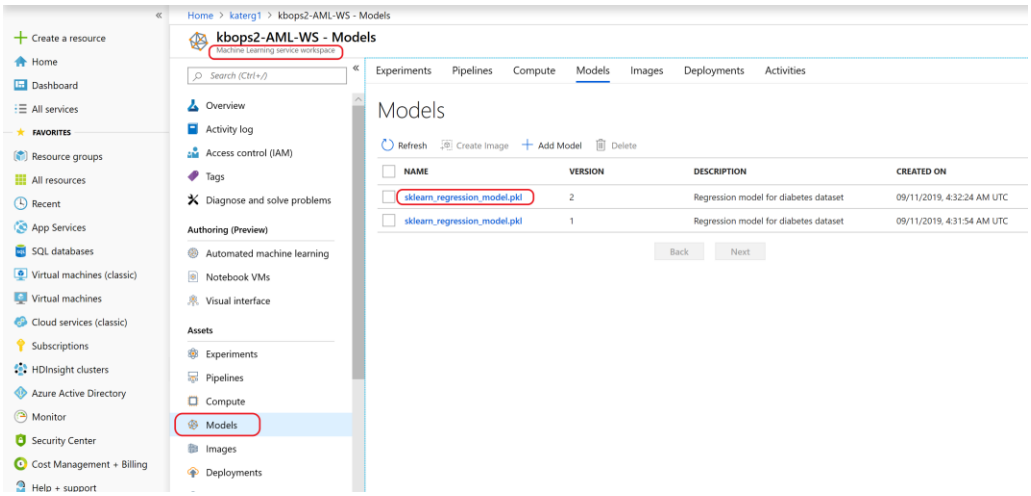


12. Examine some of the execution details by clicking into the green (or red) hyperlink **Invoke Train**. A window will pop up and you can review the steps and the execution time for each step. For

additional details, click on the individual step to see the specific command and scripts that were executed.



This pipeline executes the model training pipeline in the Azure ML Workspace. The model training pipeline has executed successfully when the model is registered in the workspace:



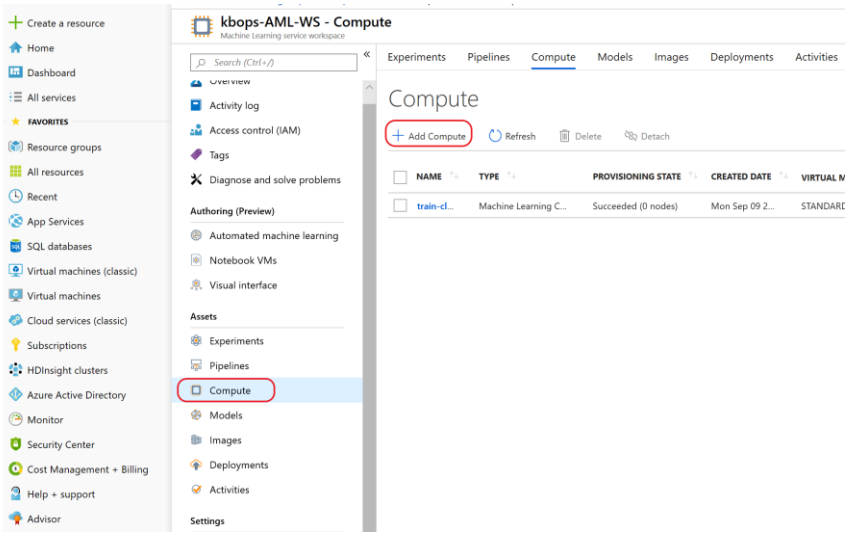
The next step is to create a Kubernetes cluster for model deployment in Production.

Exercise 11: Provision an Azure Kubernetes Cluster

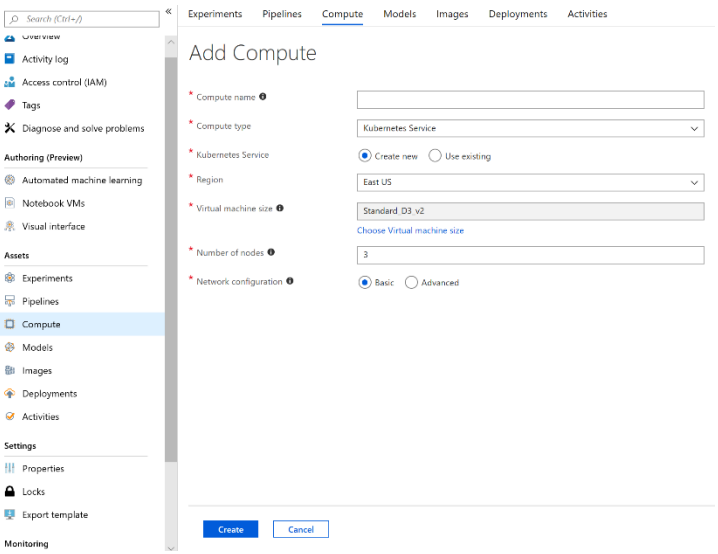
The model deployment pipeline has two stages: QA and Production. For QA stage, the ML model will be deployed to a docker container. For the Production stage, the ML model will be deployed to a Kubernetes cluster within the Azure Machine Learning Workspace. In this exercise you will create an AKS cluster that is connected to your workspace.

1. Navigate to the azure portal http://portal.azure.com
2. On the left pane, click on **Resource groups** and scroll until you see the Resource Group for this lab.
3. Click into the resource group and view the services that have been provisioned including Application Insights, Container registry, Storage Account, etc.

4. Click into **Machine learning service workspace** and scroll through to see all the artifacts that have been created including an experiment, the training pipeline, the compute platforms, and information about the deployment assets.
5. Click into **Compute** and **+ Add Compute**:



6. Give your cluster a name (note it down you will need it later) and select Kubernetes Service from the drop down for **Compute type**. Select a region from the drop down (East US) and leave the defaults. Click Create. It will take about 15 minutes for the cluster to create.



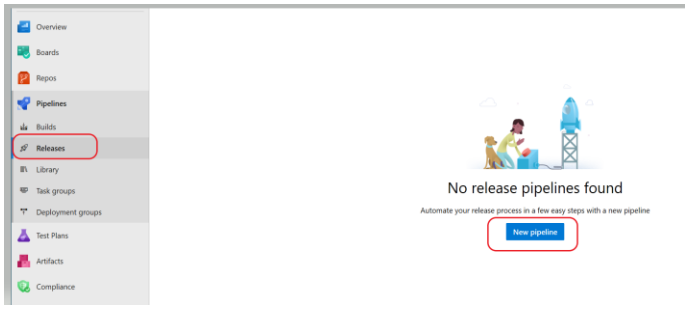
7. Monitor the status of the cluster provisioning by clicking on **Refresh**. When the provisioning state changes to **Succeeded**, the cluster is ready and the next step is to setup the deployment pipeline.

Exercise 12: 4th Pipeline: Setup the Deployment Pipeline

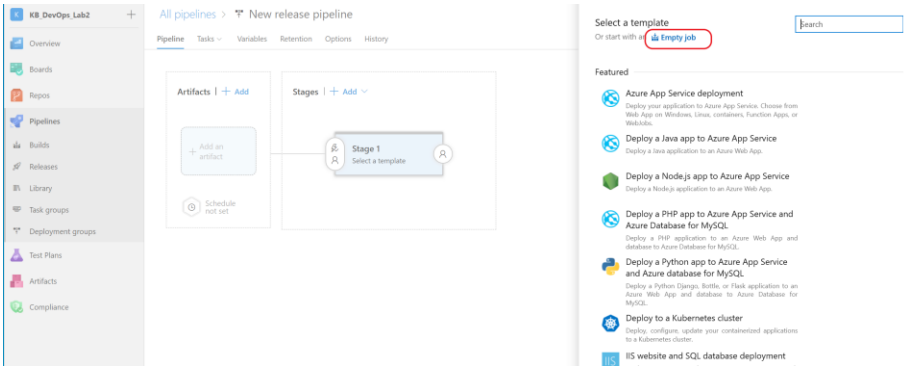
For the final pipeline – the deployment pipeline – you will create another new release pipeline. To successfully execute the deployment pipeline all three of the previous pipelines will have had to been successfully executed.

To set up the deployment pipeline:

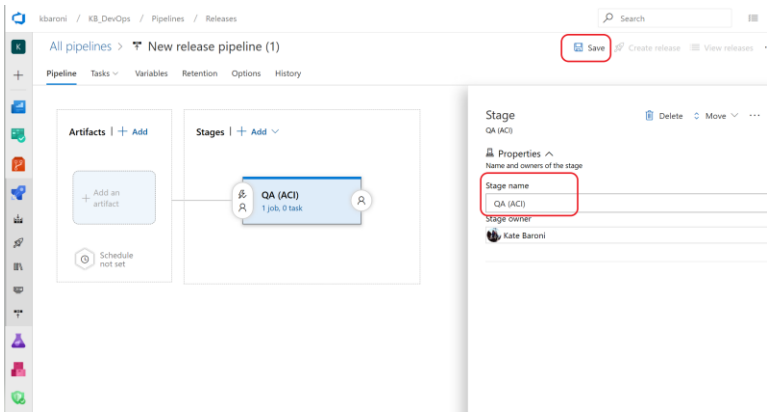
1. Navigate to the **Releases** icon on the left panel and click **New** -> **New release pipeline**:



2. Select **Empty Job** instead of selecting a template:



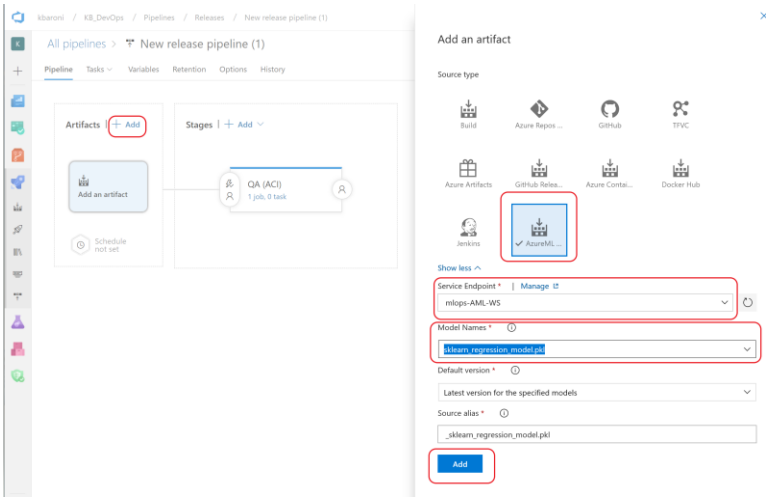
3. On the next screen, change the name from 'Stage 1' to **'QA (ACI)**', select **Save** to save the empty job and then **OK**.



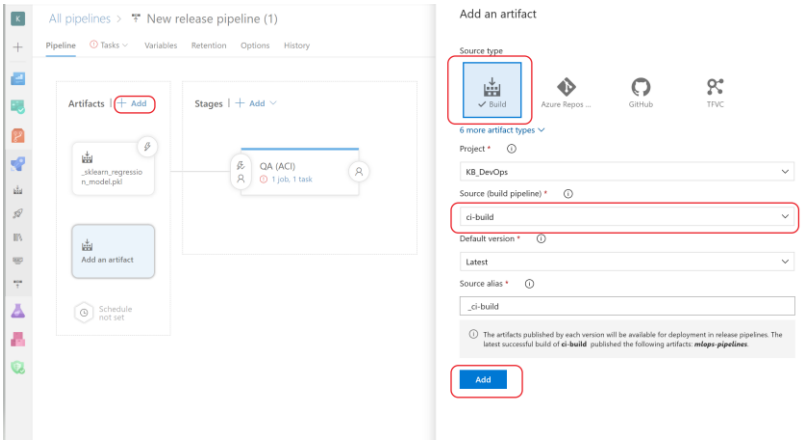
With the empty job created, you can now configure the pipeline.

13. The next step is to add an artifact to the pipeline. Click on **+ Add** and select **AzureML Model Artifact** from the source types. (*If the source type is not available to be selected, you need to go back to the exercise and Add the AzureML Extension into your DevOps organization*.)

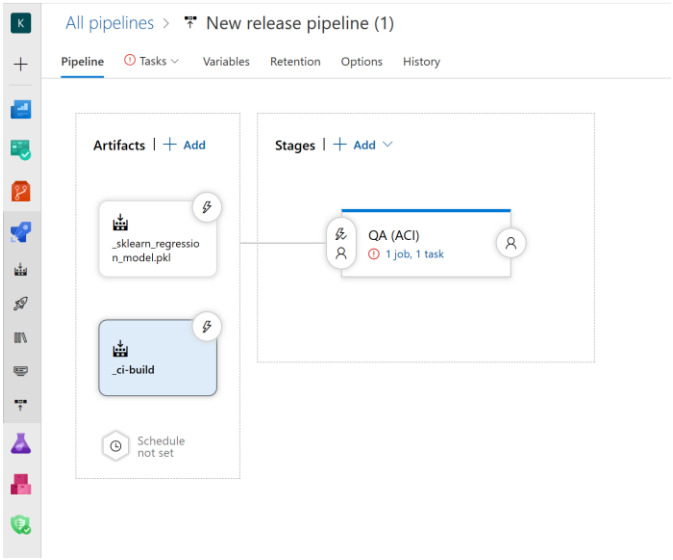
14. In the **Service Endpoint** drop down select the connection service you created for Azure ML(mlops-AML-WS) -> select *sklearn_regression_model.pkl* from the **Model Names** dropdown. (*Note: If there is no model name in the drop down, then the ML model has not been added to the AzureML Workspace. The training pipeline may still be running or the pipeline has failed. Recommend you review the status of the **Pipelines** and **Models** in the Azure ML Workspace*). Click **Add**:



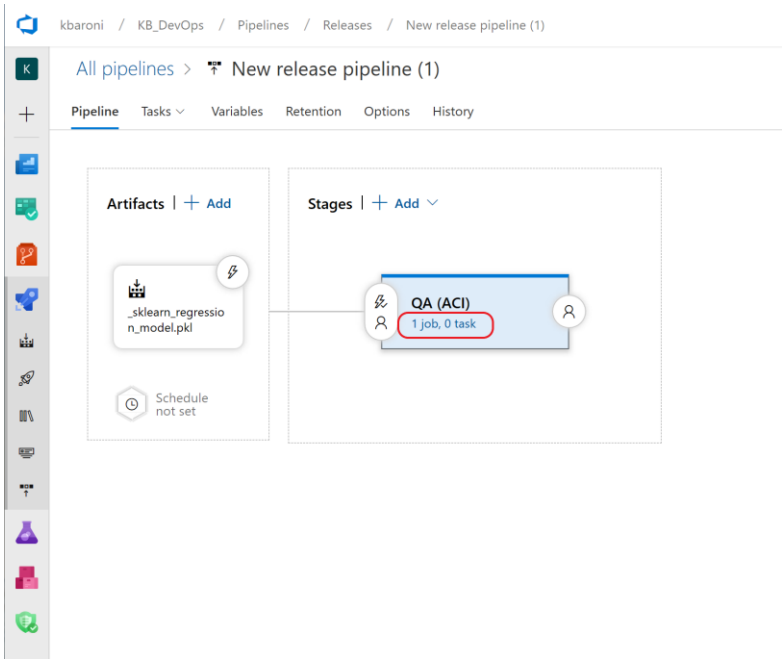
The next step is to add the build artifact. Click on **+ Add** Artifact and this time add a **Build** artifact and select **ci-build** from the **Source** drop-down:



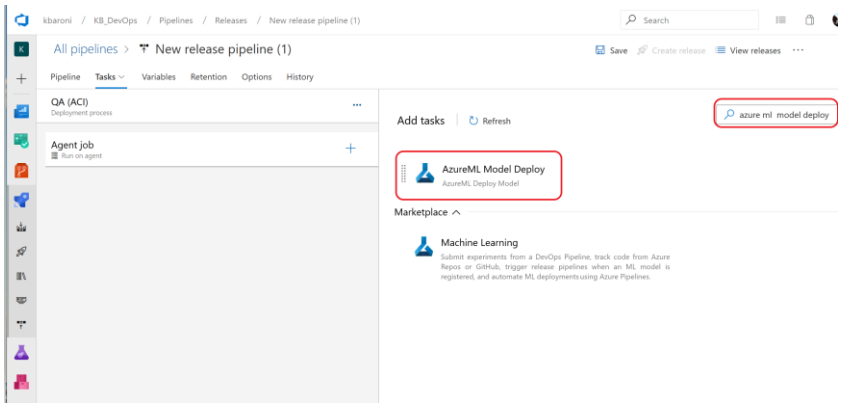
With both artifacts added, the pipeline will look like this: :



15. The next step is to configure the Agent job defined under **Stages.** Click on the link under the name of the pipeline to bring up the screen to configure the agent job.



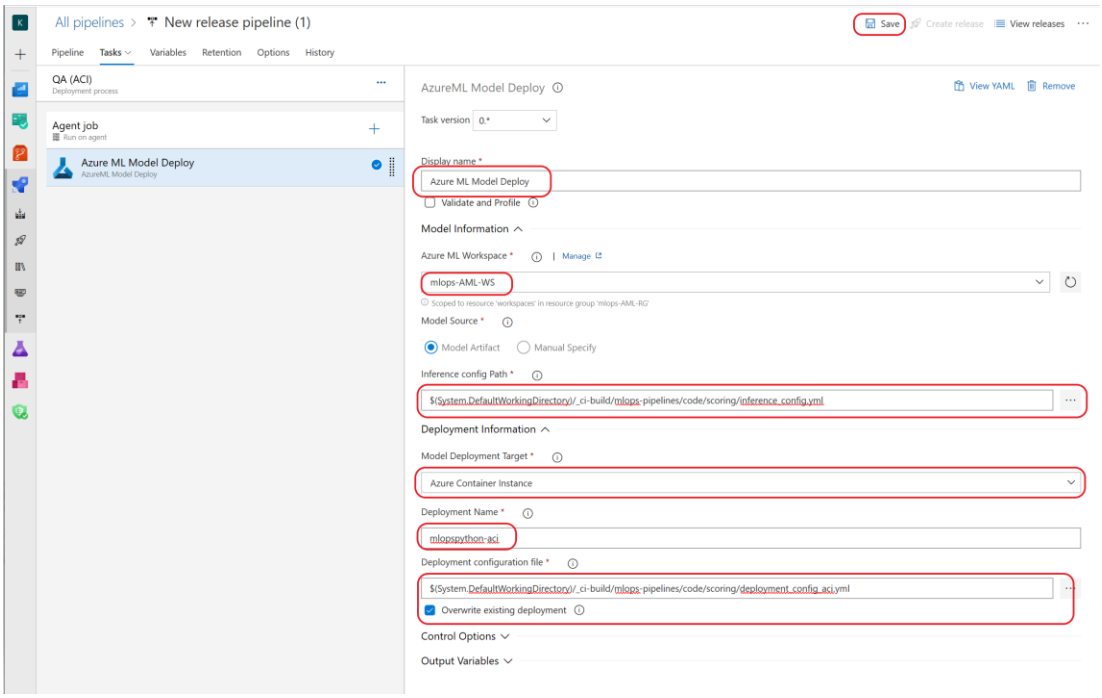
8. Click on **+** to add a task to the Agent Job, search for **Azure ML Model Deploy** and select the template when it appears in the search results:



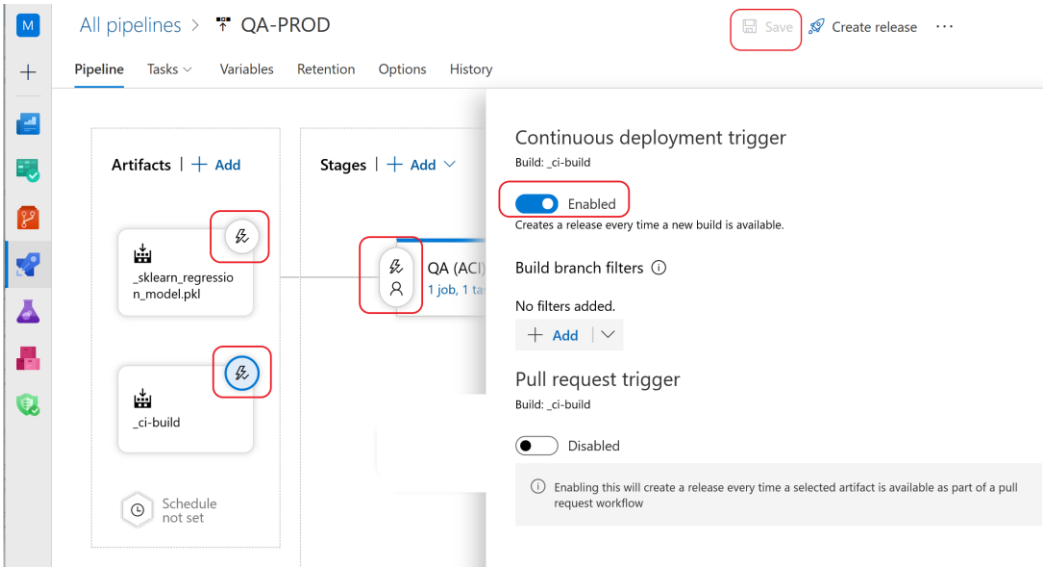
16. Configure the **Azure ML Model Deploy** task by clicking into *Some settings need attention* and specifying the following parameters:

Parameter	**Value**
Display Name	Azure ML Model Deploy
Azure ML Workspace	From the drop down, select *mlops-AML-WS*
Inference config path	From the drop down, select: $(System.DefaultWorkingDirectory/_ci-build/mlops-pipelines/code/scoring/inference_config.yml
Model Deployment Target	Azure Container Instance
Deployment Name	mlopspython-aci
Deployment Configuration File	From the drop down, select: $(System.DefaultWorkingDirectory)/_ci-build/mlops-pipelines/code/scoring/deployment_config_aci.yml
Overwrite existing deployment	(check)

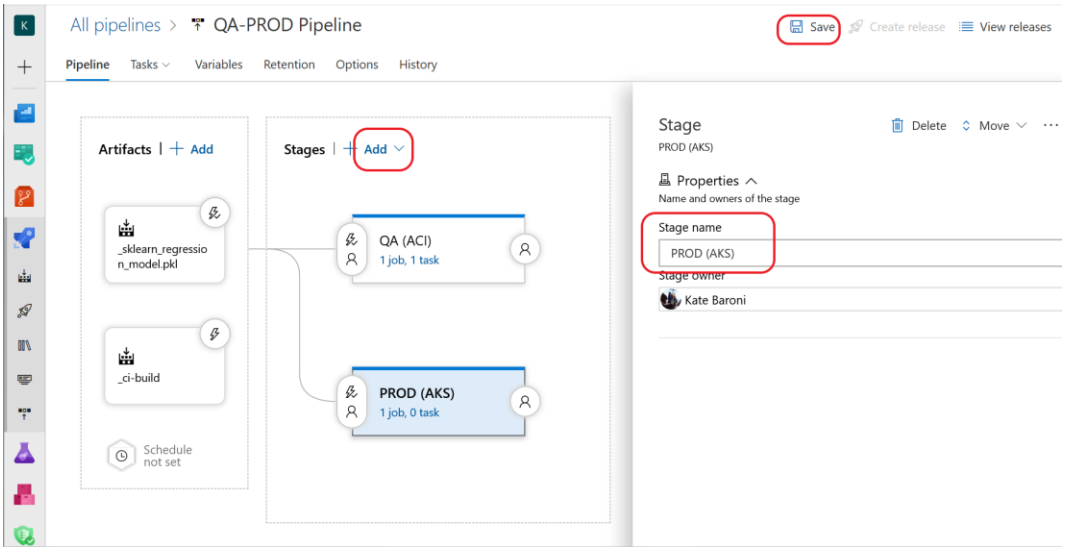
The screen will look similar to the one below. Click **Save** and **OK** to complete the steps for configuring the task.



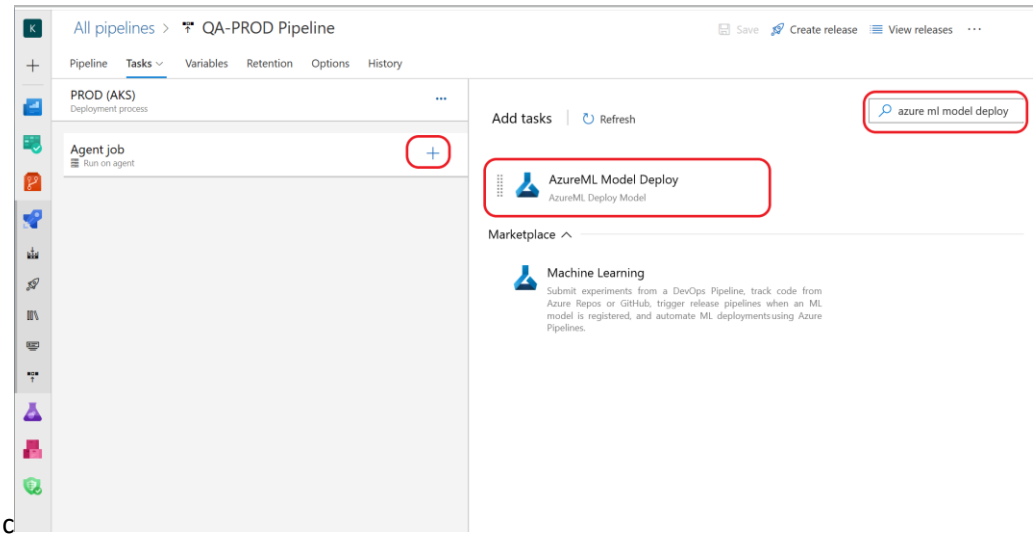
17. To trigger this pipeline every time a new AzureML model is registered, enable the *Continuous Deployment Trigger*. Navigate back to the Pipeline tab and click on the lightening sign and slide the bar for **Continuous Deployment** to **Enabled**. Do this for both the *build* and the *model deploy artifacts*. Click **Save** and then **OK.**



18. Next step is to add the Production stage into the pipeline. Add a new stage to the pipeline with an **empty job**, name the job **PROD (AKS)** and **Save**:



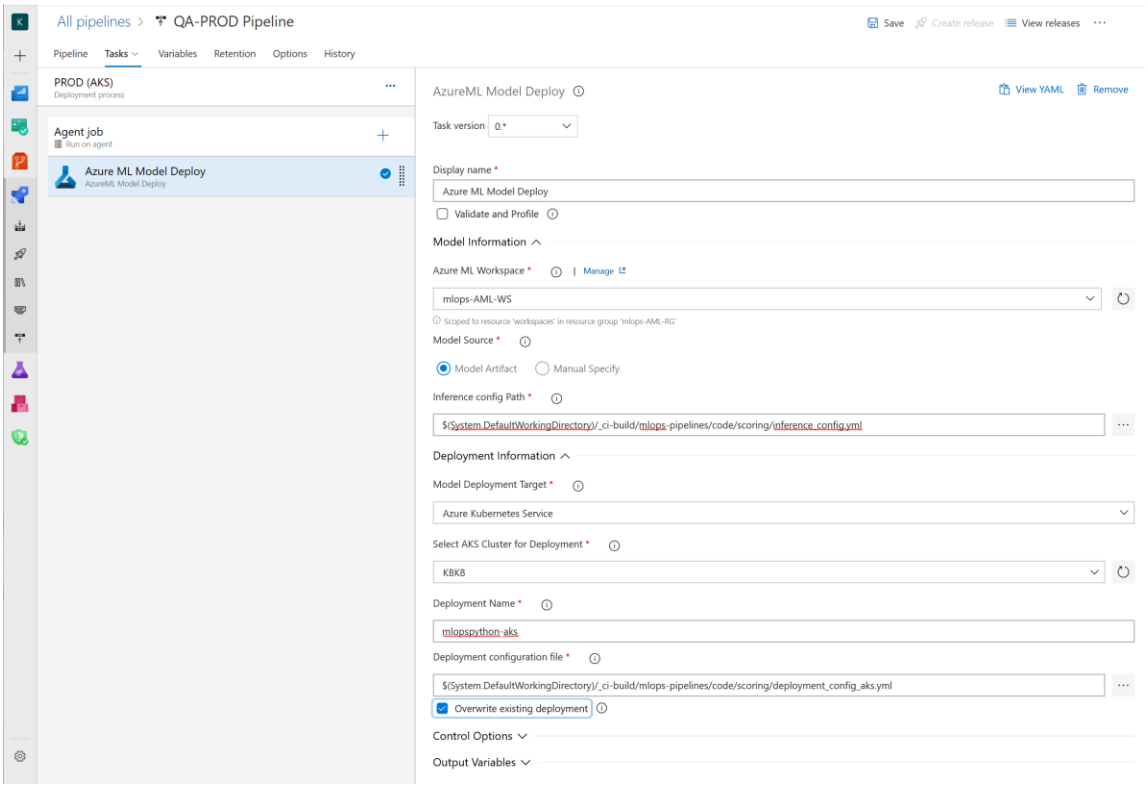
19. Add a single task to the agent job: Search for the template **Azure ML Model Deploy** task and add it to the stage:



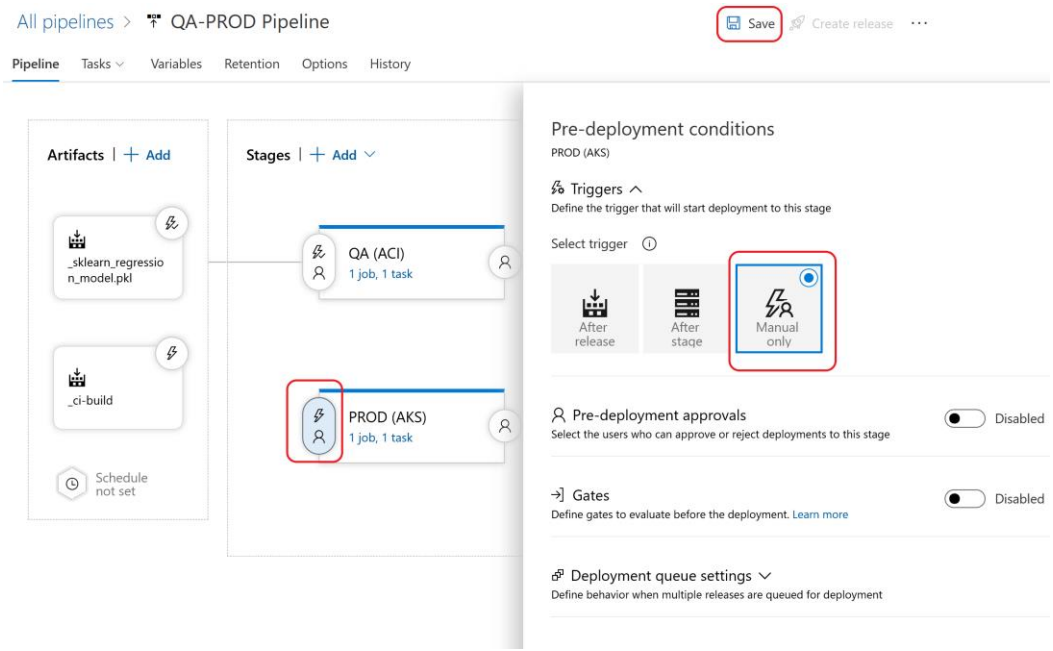
20. Configure the **Azure ML Model Deploy** task by clicking into *'Some settings need attention'* and specifying the following parameters:

Parameter	**Value**
Display Name	Azure ML Model Deploy
Azure ML Workspace	From the drop down, select *mlops-AML-WS*
Inference config path	From the drop down, select: $(System.DefaultWorkingDirectory)/_ci-build/mlops-pipelines/code/scoring/inference_config.yml
Model Deployment Target	Azure Kubernetes Service
Select AKS Cluster for Deployment	From the drop down, select the Kubernetes cluster you provisioned in an earlier step
Deployment Name	mlopspython-aks
Deployment Configuration File	From the drop down, select: $(System.DefaultWorkingDirectory)/_ci-build/mlops-pipelines/code/scoring/deployment_config_aks.yml
Overwrite existing deployment	(check)

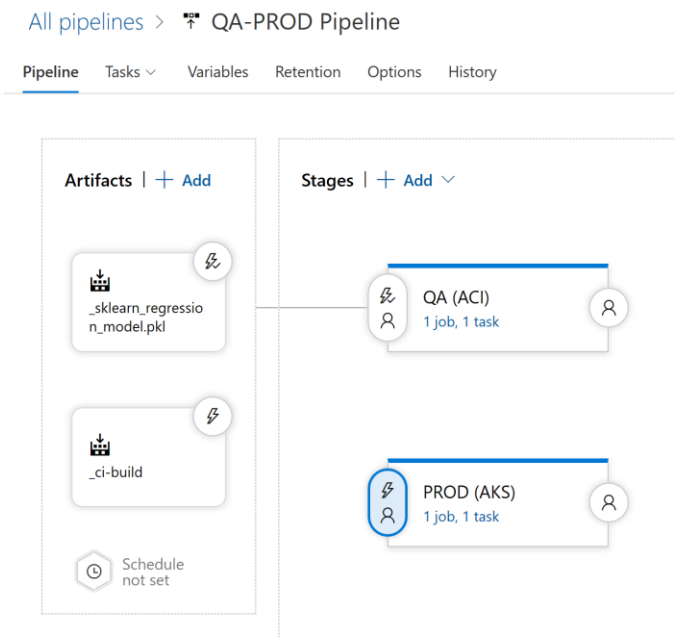
The task configuration screen will look similar to the screen shot below. Click **Save** and **OK**.



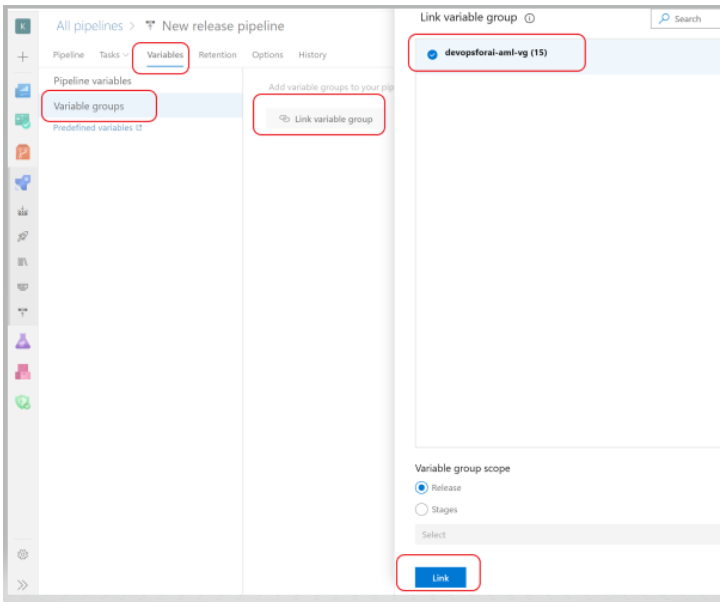
21. The pipeline now looks like the configuration below. Notice that the continuous deployment icon (the lighting icon) has been configured for both the QA and the PROD stages. Many teams prefer to have a manual-only trigger for the release into production. To turn off the automatic trigger for the PROD stage, click on the *lightening icon* to the left of the PROD stage, select **Manual only** as the trigger option, click on **Save** and then OK.



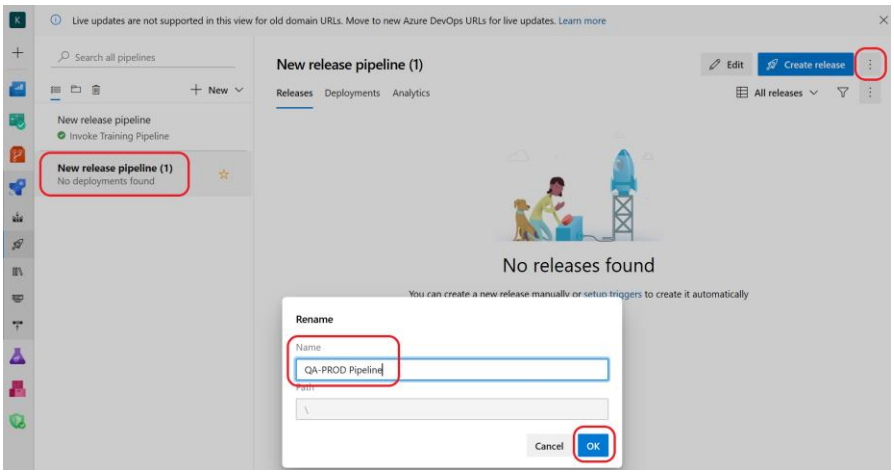
The updated pipeline will now look like this (*no checkmark next to the PROD lightening icon*):



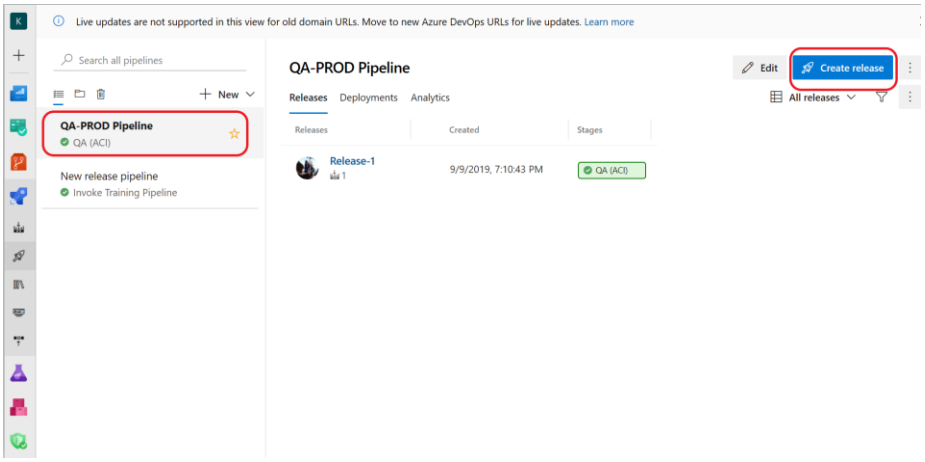
22. This pipeline also needs access to the credentials and IDs that are stored in the variables group. Click on the **Variables** tab, then click on **Variable groups** and select **Link variable group**. Select the variable group created earlier, click on **Link**, and click on **Save** and then **OK**.



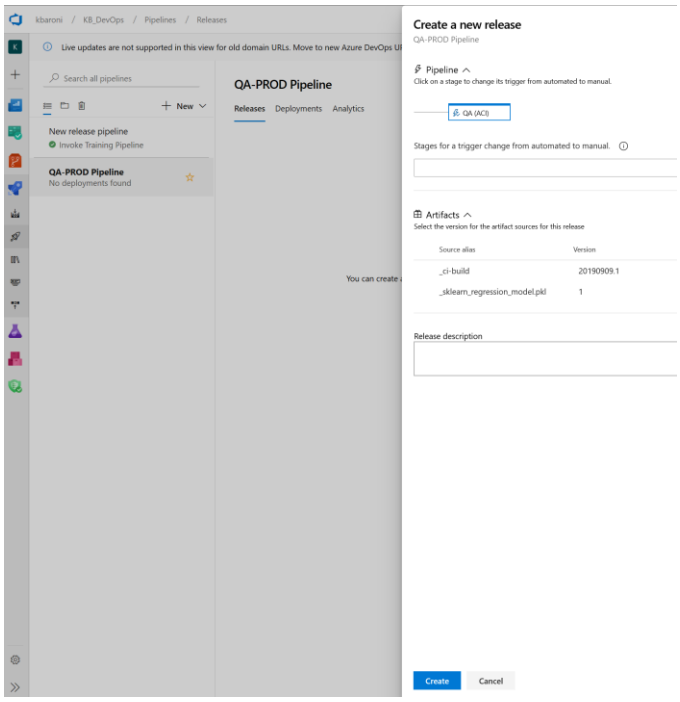
23. Rename the pipeline to QA-Prod



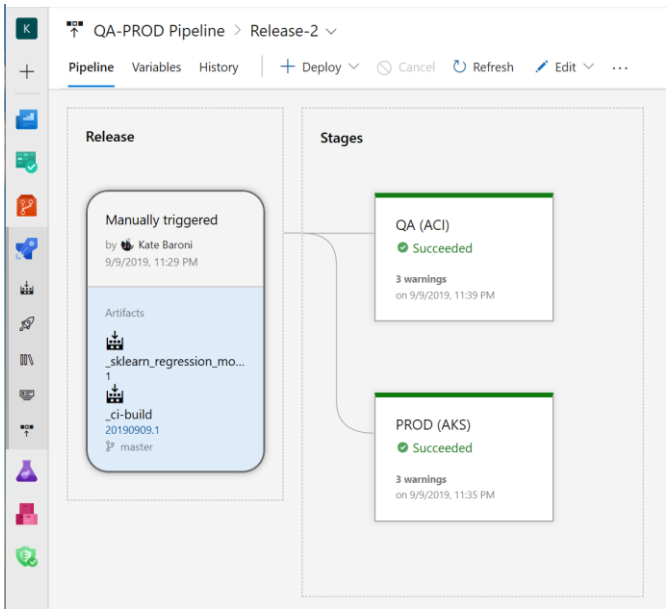
24. To verify the pipeline works as expected, initially you will manually trigger it. Navigate to the left pane and select **Releases.** Click on the **QA-PROD** pipeline we just created and then click on **Create a release** as shown below**.**



25. Click on **Create** to queue and execute the **QA-PROD** release:



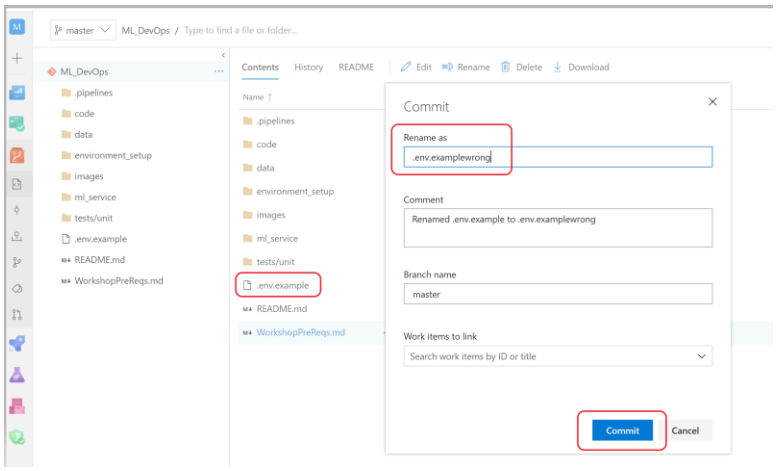
When the pipeline has successfully executed, the screen will be similar to:



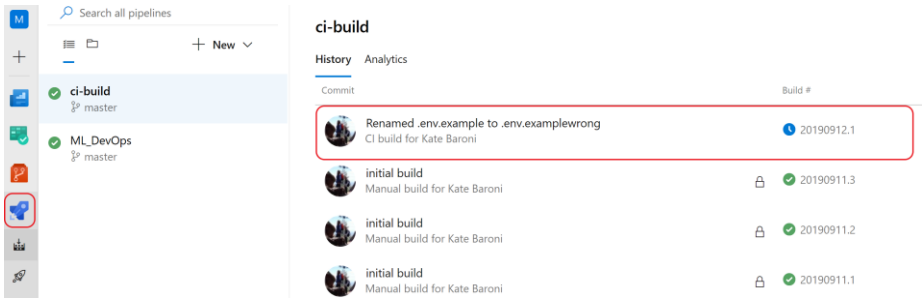
Exercise 13: Test the end-to-end CI/CD pipeline triggers

Now that the pipelines have all been configured and tested individually, try making a change that will trigger the ci-build pipeline – and subsequently both release pipelines. For instance:

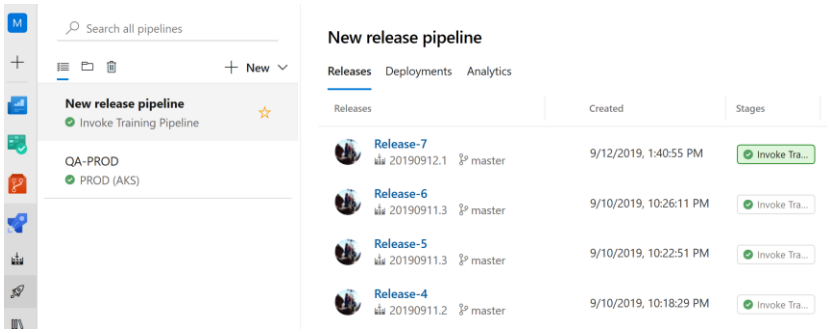
1. Navigate to the Repo icon on the left side panel, click on the file named *.env.example* and change the name of the file, then commit the change:



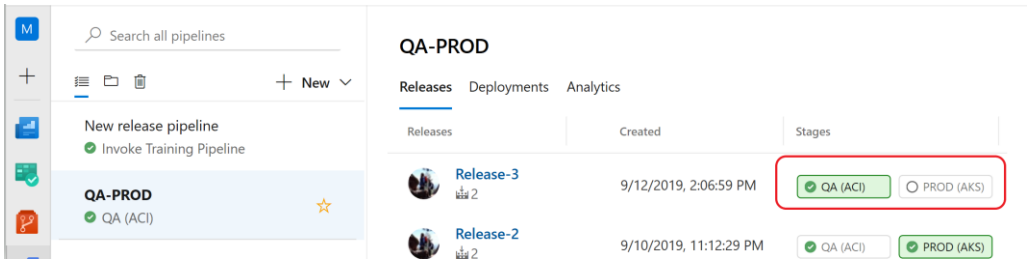
2. Navigate back to the build screen to see that **ci-build** pipeline has been queued to execute:



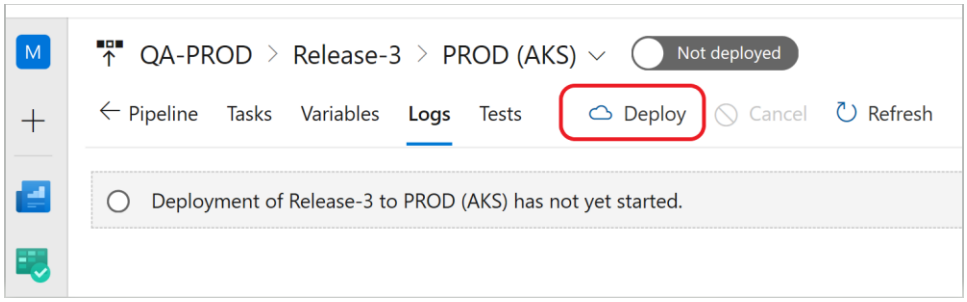
3. When the ci-build pipeline completes, the **Invoke Training** pipeline is automatically triggered:



4. When the **Invoke Training** pipeline completes, the **QA-PROD** pipeline executes. Notice the **QA (ACI)** stage is triggered to execute while the **PROD** stage does not execute. This is expected because the **PROD** stage was configured to execute manually.



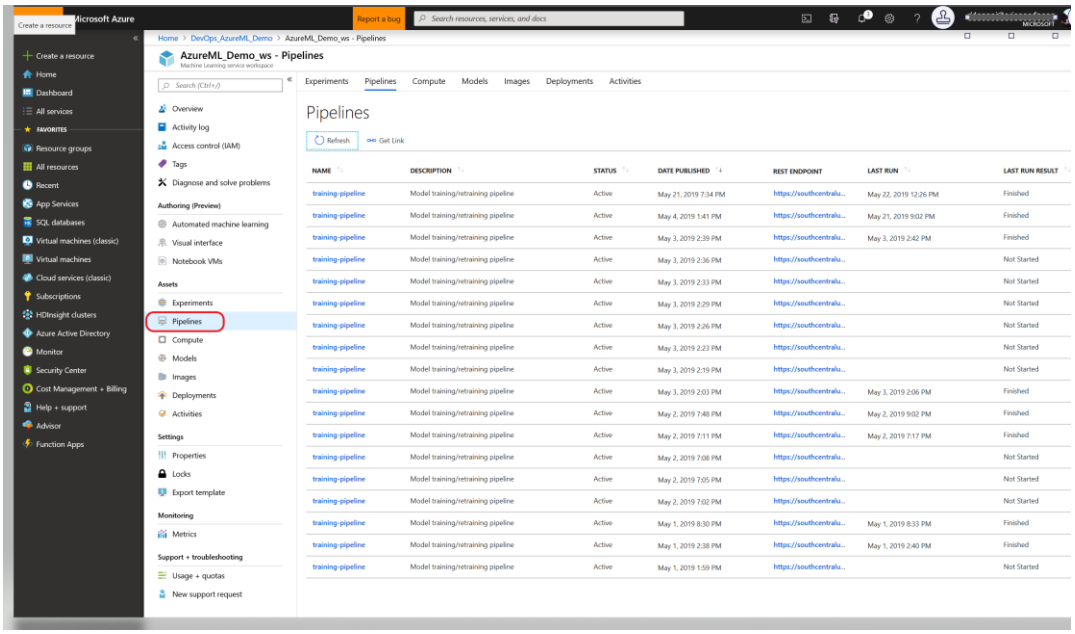
5. To execute the PROD stage, click into the stage and click Deploy:



Exercise 14 (optional): View resources from Azure Portal

View the artifacts and resources that have been provisioned through the Azure DevOps pipeline processing. To view the resources:

1. Navigate to the azure portal http://portal.azure.com
2. On the left pane, click on Resource groups and scroll until you see the Resource Group you created for this lab.
3. Click into the resource group and view the services that have been provisioned including **Container instances**, **Kubernetes**, **Key vault**, an **ACR**, etc.
4. Click into **Machine learning service workspace** and scroll through to see all the artifacts that have been created including an experiment, the training pipeline, the compute platforms, and information about the deployment assets.



Congratulations! You have completed the lab!