Kubeflow

NetApp Solutions

NetApp
July 31, 2024

Table of Contents

Kubeflow

Kubeflow Deployment

This section describes the tasks that you must complete to deploy Kubeflow in your Kubernetes cluster.

Prerequisites

Before you perform the deployment exercise that is outlined in this section, we assume that you have already performed the following tasks:

1. You already have a working Kubernetes cluster, and you are running a version of Kubernetes that is supported by the Kubeflow version that you intend to deploy. For a list of supported Kubernetes versions, refer to the dependencies for your Kubeflow version in the official Kubeflow documentation.
2. You have already installed and configured NetApp Astra Trident in your Kubernetes cluster. For more details on Astra Trident, refer to the Astra Trident documentation.

Set Default Kubernetes StorageClass

Before you deploy Kubeflow, we recommend designating a default StorageClass within your Kubernetes cluster. The Kubeflow deployment process may attempt to provision new persistent volumes using the default StorageClass. If no StorageClass is designated as the default StorageClass, then the deployment may fail. To designate a default StorageClass within your cluster, perform the following task from the deployment jump host. If you have already designated a default StorageClass within your cluster, then you can skip this step.

1. Designate one of your existing StorageClasses as the default StorageClass. The example commands that follow show the designation of a StorageClass named `ontap-ai-flexvols-retain` as the default StorageClass.

   The `ontap-nas-flexgroup` Trident Backend type has a minimum PVC size that is fairly large. By default, Kubeflow attempts to provision PVCs that are only a few GBs in size. Therefore, you should not designate a StorageClass that utilizes the `ontap-nas-flexgroup` Backend type as the default StorageClass for the purposes of Kubeflow deployment.

```
$ kubectl get sc
NAME                                PROVISIONER             AGE
ontap-ai-flexgroups-retain          csi.trident.netapp.io   25h
ontap-ai-flexgroups-retain-iface1   csi.trident.netapp.io   25h
ontap-ai-flexgroups-retain-iface2   csi.trident.netapp.io   25h
ontap-ai-flexvols-retain            csi.trident.netapp.io   3s
$ kubectl patch storageclass ontap-ai-flexvols-retain -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'
storageclass.storage.k8s.io/ontap-ai-flexvols-retain patched
$ kubectl get sc
NAME                                 PROVISIONER             AGE
ontap-ai-flexgroups-retain           csi.trident.netapp.io   25h
ontap-ai-flexgroups-retain-iface1    csi.trident.netapp.io   25h
ontap-ai-flexgroups-retain-iface2    csi.trident.netapp.io   25h
ontap-ai-flexvols-retain (default)   csi.trident.netapp.io   54s
```

Kubeflow Deployment Options

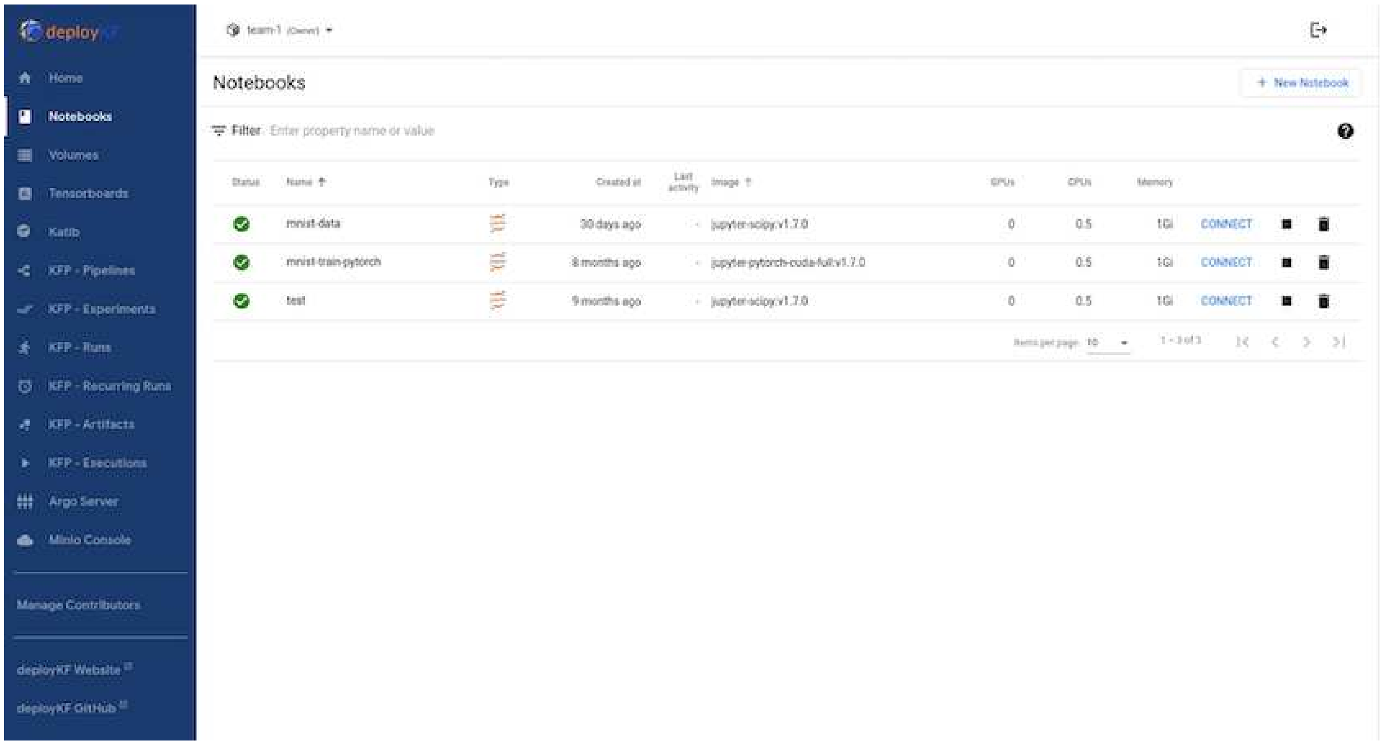
There are many different options for deploying Kubeflow. Refer to the official Kubeflow documentation for a list of deployment options, and choose the option that is the best fit for your needs.

For validation purposes, we deployed Kubeflow 1.7 using deployKF 0.1.1.

Example Kubeflow Operations and Tasks

Provision a Jupyter Notebook Workspace for Data Scientist or Developer Use

Kubeflow is capable of rapidly provisioning new Jupyter Notebook servers to act as data scientist workspaces. For more information about Jupyter Notebooks within the Kubeflow context, see the official Kubeflow documentation.



Notebooks

Status	Name	Type	Created at	Last activity	Image	GPUs	CPUs	Memory			
✓	mnist-data		30 days ago	-	jupyter-scipy:v1.7.0	0	0.5	1Gi	CONNECT	■	🗑
✓	mnist-train-pytorch		8 months ago	-	jupyter-pytorch-cuda-full:v1.7.0	0	0.5	1Gi	CONNECT	■	🗑
✓	test		9 months ago	-	jupyter-scipy:v1.7.0	0	0.5	1Gi	CONNECT	■	🗑

Items per page: 10   1 - 3 of 3

Use the NetApp DataOps Toolkit with Kubeflow

The NetApp Data Science Toolkit for Kubernetes can be used in conjunction with Kubeflow. Using the NetApp Data Science Toolkit with Kubeflow provides the following benefits:

- Data scientists can perform advanced NetApp data management operations, such as creating snapshots and clones, directly from within a Jupyter Notebook.
- Advanced NetApp data management operations, such as creating snapshots and clones, can be incorporated into automated workflows using the Kubeflow Pipelines framework.

Refer to the Kubeflow Examples section within the NetApp Data Science Toolkit GitHub repository for details on using the toolkit with Kubeflow.

Example Workflow - Train an Image Recognition Model Using Kubeflow and the NetApp DataOps Toolkit

This section describes the steps involved in training and deploying a Neural Network for Image Recognition using Kubeflow and the NetApp DataOps Toolkit. This is intended to serve as an example to show a training job that incorporates NetApp storage.

Prerequisites

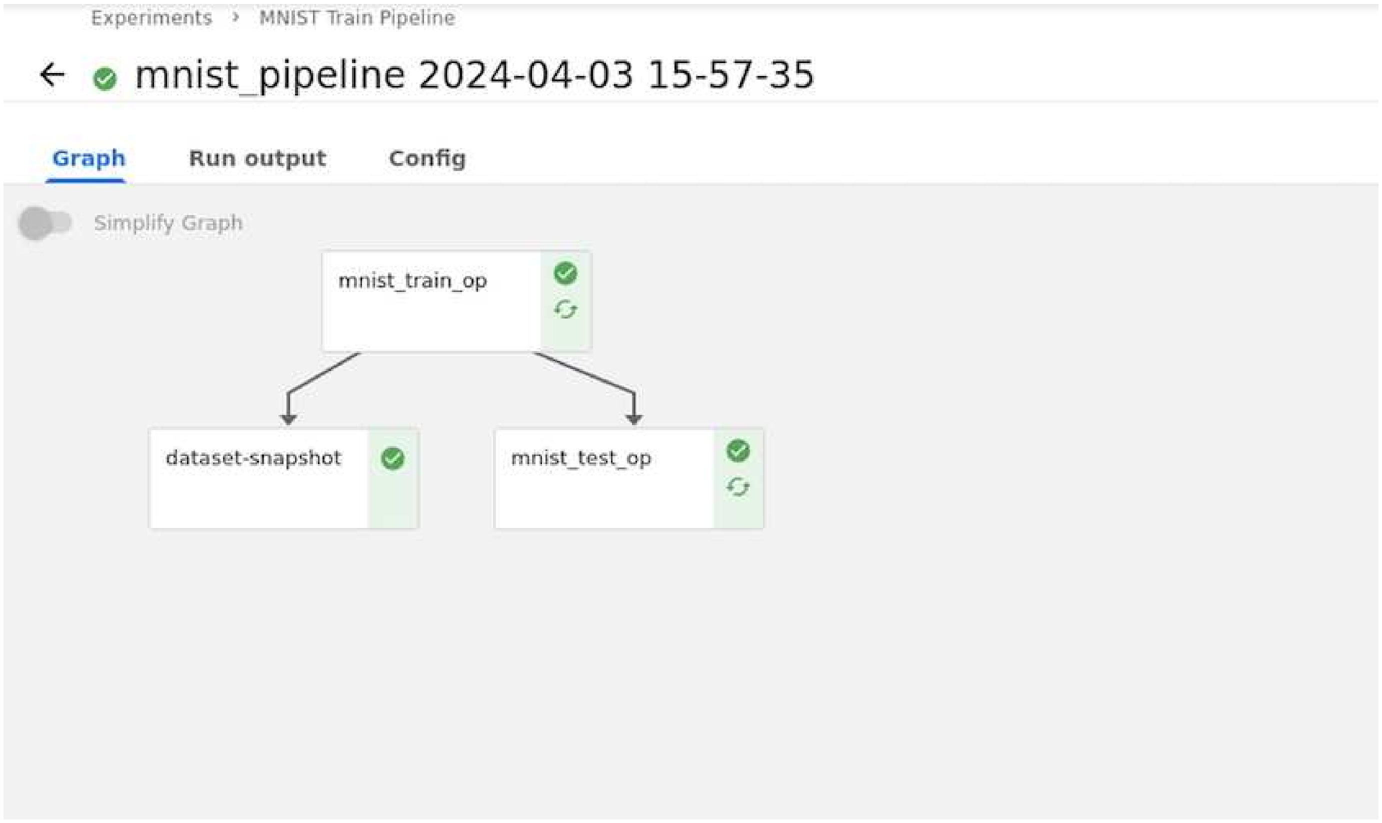
Create a Dockerfile with the required configurations to use for the train and test steps within the Kubeflow pipeline.
Here is an example of a Dockerfile -

```
FROM pytorch/pytorch:latest
RUN pip install torchvision numpy scikit-learn matplotlib tensorboard
WORKDIR /app
COPY . /app
COPY train_mnist.py /app/train_mnist.py
CMD ["python", "train_mnist.py"]
```

Depending on your requirements, install all required libraries and packages needed to run the program. Before you train the Machine Learning model, it is assumed that you already have a working Kubeflow deployment.

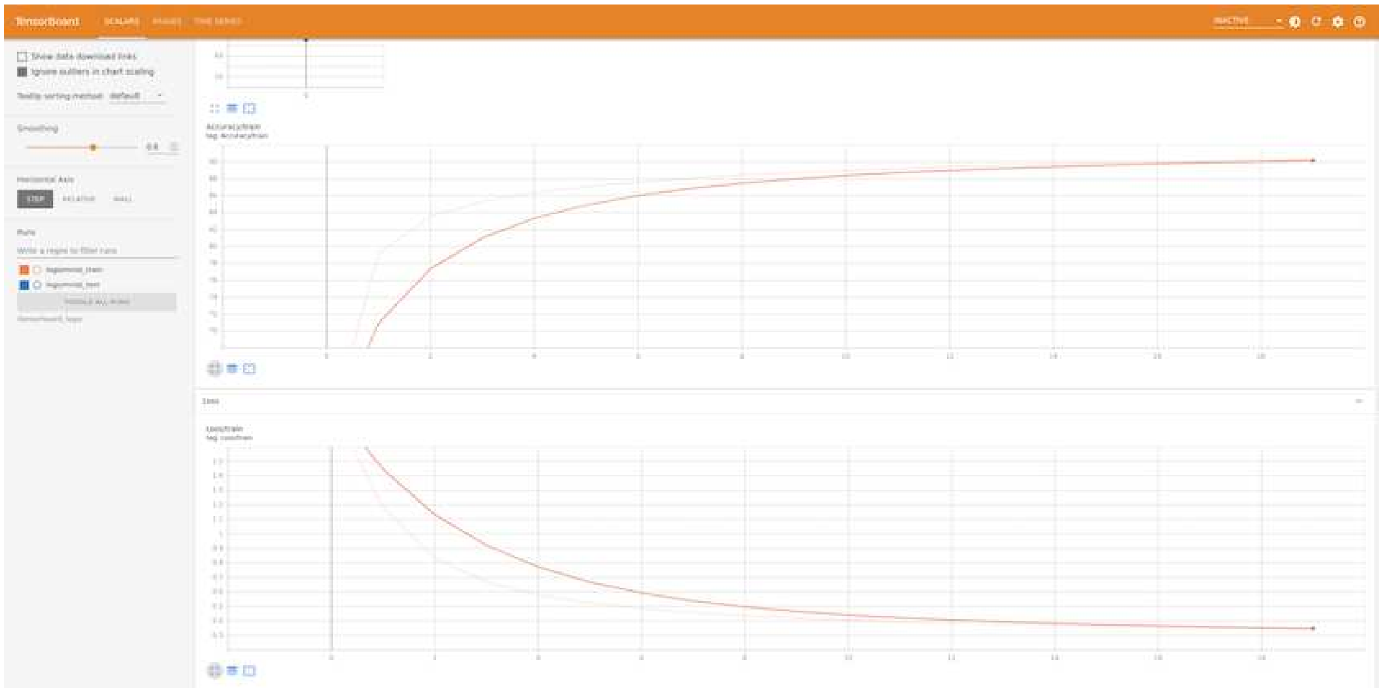
Train a Small NN on MNIST Data Using PyTorch and Kubeflow Pipelines

We use the example of a small Neural Network trained on MNIST data. The MNIST dataset consists of handwritten images of digits from 0-9. The images are 28x28 pixels in size. The dataset is divided into 60,000 train images and 10,000 validation images. The Neural Network used for this experiment is a 2-layer feedforward network. Training is executed using Kubeflow Pipelines. Refer to the documentation here for more information. Our Kubeflow pipeline incorporates the docker image from the Prerequisites section.



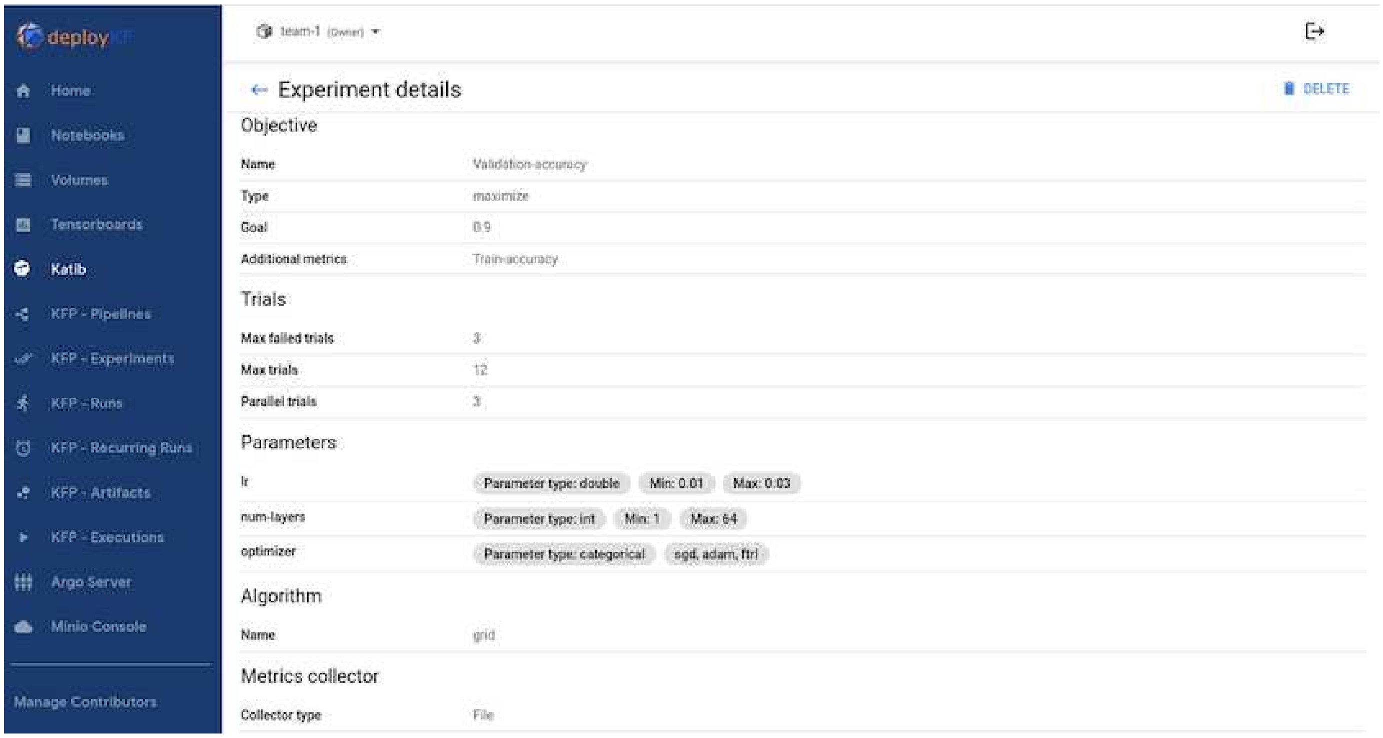
Visualize Results Using Tensorboard

Once the model is trained, we can visualize the results using Tensorboard. Tensorboard is available as a feature on the Kubeflow Dashboard. You can create a custom tensorboard for your job. An example below shows the plot of training accuracy vs. number of epochs and training loss vs. number of epochs.



Experiment with Hyperparameters Using Katib

Katib is a tool within Kubeflow that can be used to experiment with the model hyperparameters. To create an experiment, define a desired metric/goal first. This is usually the test accuracy. Once the metric is defined, choose hyperparameters that you would like to play around with (optimizer/learning_rate/number of layers). Katib does a hyperparameter sweep with the user-defined values to find the best combination of parameters that satisfy the desired metric. You can define these parameters in each section in the UI. Alternatively, you could define a **YAML** file with the necessary specifications. Below is an illustration of a Katib experiment -



← Experiment details

Objective

Name	Validation-accuracy
Type	maximize
Goal	0.9
Additional metrics	Train-accuracy

Trials

Max failed trials	3
Max trials	12
Parallel trials	3

Parameters

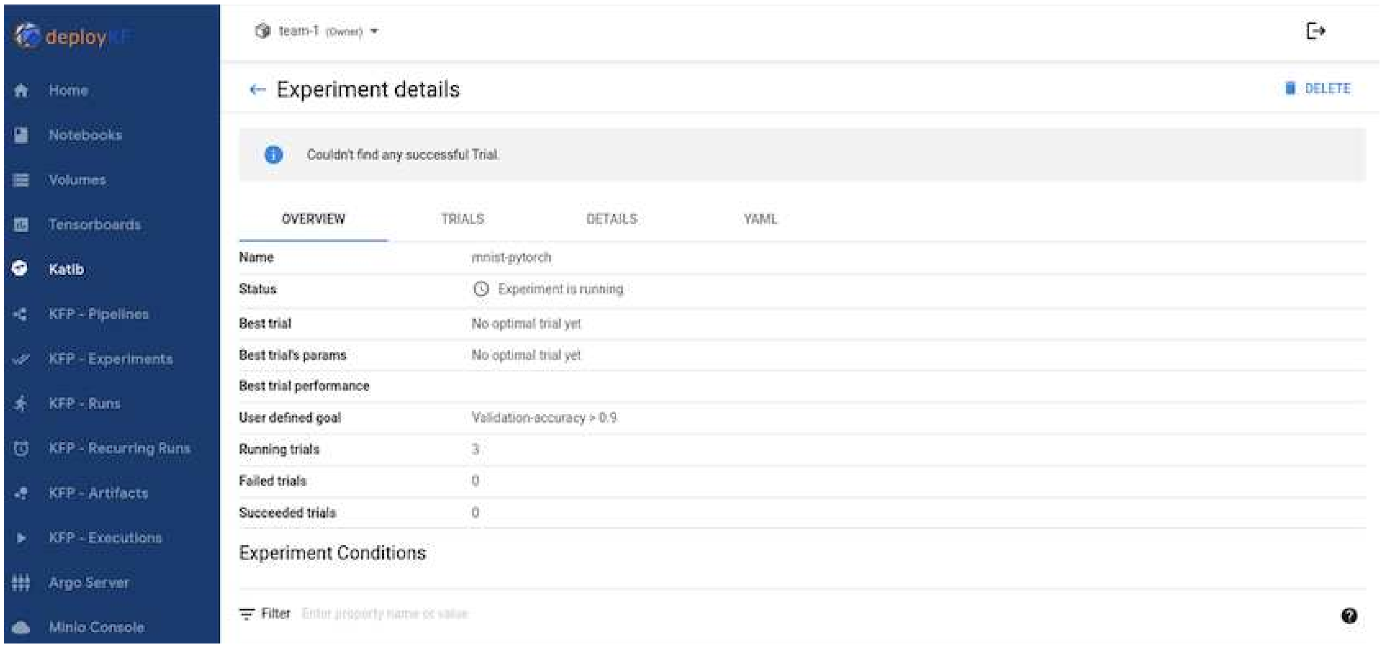
lr	Parameter type: double	Min: 0.01	Max: 0.03
num-layers	Parameter type: int	Min: 1	Max: 64
optimizer	Parameter type: categorical	sgd, adam, ftrl	

Algorithm

Name	grid

Metrics collector

Collector type	File



Experiment details

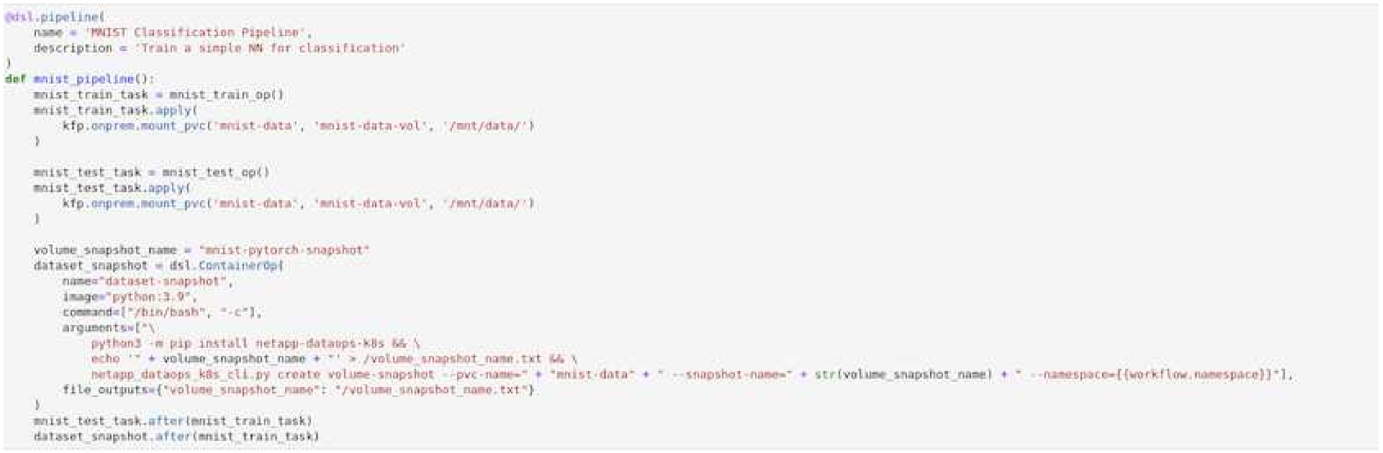
Couldn't find any successful Trial.

Name mnist-pytorch
Status Experiment is running
Best trial No optimal trial yet
Best trial's params No optimal trial yet
Best trial performance
User defined goal Validation-accuracy > 0.9
Running trials 3
Failed trials 0
Succeeded trials 0

Experiment Conditions

Use NetApp Snapshots to Save Data for Traceability

During the model training, we may want to save a snapshot of the training dataset for traceability. To do this, we can add a snapshot step to the pipeline as shown below. To create the snapshot, we can use the NetApp DataOps Toolkit for Kubernetes.



```
@dsl.pipeline(
    name = 'MNIST Classification Pipeline',
    description = 'Train a simple NN for classification'
)
def mnist_pipeline():
    mnist_train_task = mnist_train_op()
    mnist_train_task.apply(
        kfp.onprem.mount_pvc('mnist-data', 'mnist-data-vol', '/mnt/data/')
    )

    mnist_test_task = mnist_test_op()
    mnist_test_task.apply(
        kfp.onprem.mount_pvc('mnist-data', 'mnist-data-vol', '/mnt/data/')
    )

    volume_snapshot_name = "mnist-pytorch-snapshot"
    dataset_snapshot = dsl.ContainerOp(
        name="dataset-snapshot",
        image="python:3.9",
        command=["/bin/bash", "-c"],
        arguments=["\
            python3 -m pip install netapp-dataops-k8s && \
            echo '" + volume_snapshot_name + "' > /volume_snapshot_name.txt && \
            netapp_dataops_k8s_cli.py create volume-snapshot --pvc-name=" + "mnist-data" + " --snapshot-name=" + str(volume_snapshot_name) + " --namespace={{workflow.namespace}}"],
        file_outputs={"volume_snapshot_name": "/volume_snapshot_name.txt"}
    )
    mnist_test_task.after(mnist_train_task)
    dataset_snapshot.after(mnist_train_task)
```

Refer to the NetApp DataOps Toolkit example for Kubeflow for more information.