MLflow

NetApp Solutions

NetApp
August 14, 2024

Table of Contents

MLflow

MLflow Deployment

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

> It is possible to deploy MLflow on platforms other than Kubernetes. Deploying MLflow on platforms other than Kubernetes is outside of the scope of this solution.

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.
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.

Install Helm

MLflow is deployed using Helm, a popular package manager for Kubernetes. Before you deploy MLflow, you must install Helm on your Kubernetes control node. To install Helm, follow the installation instructions in the official Helm documentation.

Set Default Kubernetes StorageClass

Before you deploy MLflow, you must designate a default StorageClass within your Kubernetes cluster. To designate a default StorageClass within your cluster, follow the instructions outlined in the Kubeflow Deployment section. If you have already designated a default StorageClass within your cluster, then you can skip this step.

Deploy MLflow

Once the pre-requisites have been met, you can start with MLflow deployment using the helm chart.

Configure MLflow Helm Chart Deployment.

Before we deploy MLflow using the Helm chart, we can configure the deployment to use NetApp Trident Storage Class and change other parameters to suit our needs using a **config.yaml** file. An example of **config.yaml** file can be found at: https://github.com/bitnami/charts/blob/main/bitnami/mlflow/values.yaml

> You can set the Trident storageClass under the **global.defaultStorageClass** parameter in the config.yaml file (e.g. storageClass: "ontap-flexvol").

Installing the Helm Chart

The Helm chart can be installed with the custom **config.yaml** file for MLflow using the following command:

```
helm install oci://registry-1.docker.io/bitnamicharts/mlflow -f
config.yaml --generate-name --namespace jupyterhub
```

The command deploys MLflow on the Kubernetes cluster in the custom configuration via the provided **config.yaml** file. MLflow is deployed in the given namespace and a random release name is given via kubernetes for the release.

Check Deployment

After the Helm chart is done deploying, you can check if the service is accessible using:

```
kubectl get service -n jupyterhub
```

Replace **jupyterhub** with the namespace you used during deployment.

You should see the following services:

```
NAME                              TYPE        CLUSTER-IP      EXTERNAL-IP
PORT(S)           AGE
mlflow-1719843029-minio           ClusterIP   10.233.22.4     <none>
80/TCP,9001/TCP   25d
mlflow-1719843029-postgresql      ClusterIP   10.233.5.141    <none>
5432/TCP          25d
mlflow-1719843029-postgresql-hl   ClusterIP   None            <none>
5432/TCP          25d
mlflow-1719843029-tracking        NodePort    10.233.2.158    <none>
30002:30002/TCP   25d
```

We edited the config.yaml file to use NodePort service to access MLflow on port 30002.

Access MLflow

Once all the services related to MLflow are up and running you can access it using the given NodePort or LoadBalancer IP address (e.g. http://10.61.181.109:30002)

Dataset-to-model Traceability with NetApp and MLflow

The NetApp DataOps Toolkit for Kubernetes can be used in conjunction with MLflow's experiment tracking capabilities in order to implement dataset-to-model or workspace-to-model traceability.

To implement dataset-to-model or workspace-to-model traceability, simply create a snapshot of your dataset or workspace volume using the DataOps Toolkit as part of your training run, as shown the following example code snippet. This code will save the data volume name and snapshot name as tags associated with the specific training run that you are logging to your MLflow experiment tracking server.

```
...
from netapp_dataops.k8s import create_volume_snapshot

with mlflow.start_run() :
    ...

    namespace = "my_namespace" # Kubernetes namespace in which dataset
volume PVC resides
    dataset_volume_name = "project1" # Name of PVC corresponding to
dataset volume
    snapshot_name = "run1" # Name to assign to your new snapshot

    # Create snapshot
    create_volume_snapshot(
        namespace=namespace,
        pvc_name=dataset_volume_name,
        snapshot_name=snapshot_name,
        printOutput=True
    )

    # Log data volume name and snapshot name as "tags"
    # associated with this training run in mlflow.
    mlflow.set_tag("data_volume_name", dataset_volume_name)
    mlflow.set_tag("snapshot_name", snapshot_name)

    ...
```

Copyright information

Trademark information