**Lesson 02 Demo 01**

**Setting up Kubeflow and Deploying a Model**

**Objective:** To set up Kubeflow and deploy a model

**Tools Required:** Python FSD lab

**Prerequisites:** None

Steps to be followed:

1. Set up an environment
2. Set up a Kubernetes cluster
3. Create a sample machine learning model
4. Dockerize machine learning model
5. Create requirements.txt file
6. Build Docker image
7. Deploy the Docker container using Kubernetes
8. Expose the service
9. Access the MLflow UI

**Step 1: Set up an environment**

* 1. Go to the project directory using the below command:

**cd /home/nidhissimplilea/Desktop/MLOps/demo1/**



* 1. Create a virtual environment with the following command:

**python -m venv mlflow-env**



**Note:** If you get the error shown below, then follow the troubleshooting steps 1.3 and 1.4.

**A white background with black text

Description automatically generated**

* 1. Run the below command to update apt:

**sudo apt update**

A screenshot of a computer

Description automatically generated

* 1. Run the below command to install Python:

**sudo apt install python3.8-venv**

A screenshot of a computer

Description automatically generated

**Note:** Remember when it asks *if you want to continue?,* write Y (uppercase).

* 1. Continue with the creation of virtual environment using the below command:

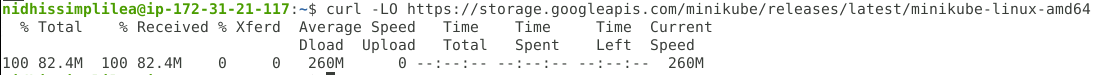
**python -m venv mlflow-env**



**Step 2: Set up a kubernetes cluster**

2.1 Enter the command to setup minikube:

**curl -LO** [**https://storage.googleapis.com/minikube/releases/latest/minikube- linux-amd64**](https://storage.googleapis.com/minikube/releases/latest/minikube-%20%20%20%20%20%20%20%20%20linux-amd64)



2.2 Install a minikube using the below command:

**sudo install minikube-linux-amd64/usr/local/bin/minikube**



2.3 Start the minikube using the below given command:

**minikube start**

A white background with black text

Description automatically generated

**Step 3: Create a simple machine learning model**

3.1 Write a simple Python script for your machine learning model. For this example, let's

create a basic scikit-learn model and save it in a file named app.py. This script predicts a simple numeric value:

# app.py

**import numpy as np**

**from sklearn.linear\_model import LinearRegression**

# Sample data

**X = np.array([1, 2, 3, 4, 5]).reshape(-1, 1)**

**y = np.array([2, 4, 5, 4, 5])**

# Create a Linear Regression model

**model = LinearRegression()**

**model.fit(X, y)**

# Predict a value

**prediction = model.predict([[6]])**

**print("Prediction:", prediction[0])**

**Step 4: Dockerize machine learning model**

4.1 Create a Dockerfile in your project directory to package your model and its

dependencies. Here's an example of a Dockerfile:

# Dockerfile

# Use an official Python runtime as a parent image

**FROM python:3.8-slim**

# Set the working directory to /app

**WORKDIR /app**

# Copy the current directory contents into the container at /app

**COPY . /app**

# Install any needed packages specified in requirements.txt

**RUN pip install -r requirements.txt**

# Make port 80 available to the world outside this container

**EXPOSE 80**

# Define environment variable

**ENV NAME World**

# Run app.py when the container launches

**CMD ["python", "app.py"]**

**Step 5: Create requirements.txt file**

5.1 Create a requirements.txt file using the below command:

**scikit-learn==0.24.2 # A popular machine learning library**

**numpy==1.20.3 # For numerical operations**

**Step 6: Build Docker image**

6.1 Create the virtual environment with the below command:

**python -m venv mlflow-env**

6.2 Activate the virtual environment using the below command:

**source mlflow-env/bin/activate**

6.3 Execute the following command to install mlflow:

**pip install mlflow**



**Note**: Docker file must be without extensions.

6.4 Build the Docker using the below command:

**docker build -t demo1 .**

A screenshot of a computer

Description automatically generated

**Step 7: Deploy the Docker container using Kubernetes**

7.1 Create a Kubernetes deployment YAML file, for example, deployment.yaml

**apiVersion: apps/v1**

**kind: Deployment**

**metadata:**

**name: ml-model-deployment**

**spec:**

**replicas: 1**

**selector:**

**matchLabels:**

**app: ml-model**

**template:**

**metadata:**

**labels:**

**app: ml-model**

**spec:**

**containers:**

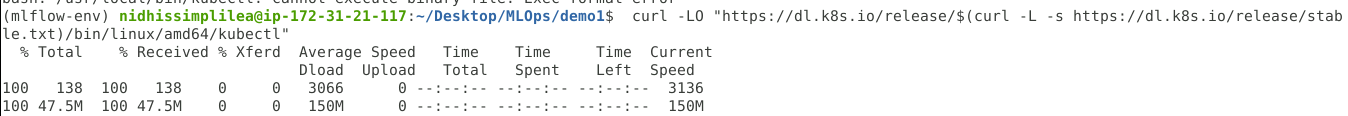
**- name: ml-model-container**

**Image: demo1 # Use the image name from step 4**

7.2 To install Kubectl, use the following command:

**curl -LO "**[**https://dl.k8s.io/release/$(curl**](https://dl.k8s.io/release/$%28curl)

[**https://dl.k8s.io/release/stable.txt**](https://dl.k8s.io/release/stable.txt)[**)/bin/linux/amd64/kubectl**](https://dl.k8s.io/release/stable.txt%29/bin/linux/amd64/kubectl)**"**



**sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl**



7.3 Apply Kubectl using the below command:

**kubectl apply -f deployment.yaml**



**Step 8: Expose the service**

8.1 Create a service YAML file to access MLflow web UI, for example, service.yaml

**apiVersion: v1**

**kind: Service**

**metadata:**

**name: mlflow-service**

**spec:**

**selector:**

**app: ml-model**

**ports:**

**- protocol: TCP**

**port: 80**

**targetPort: 80**

**type: LoadBalancer**

8.2 Apply the service using the below command:

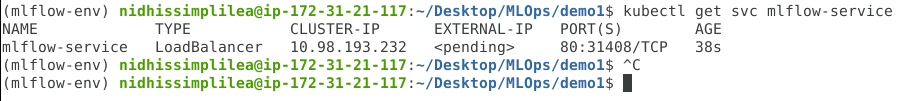
**Kubectl apply -f service.yaml**



**Step 9: Access the MLflow UI**

9.1Find the external IP address of the service using the below command:

**kubectl get svc mlflow-service**



You have successfully set up Kubeflow and deployed a model.