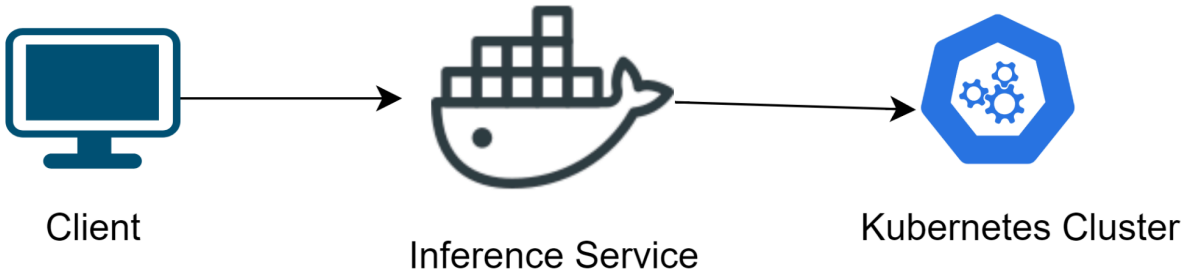
Architecture Diagram



Components:

1. **Client**: Represents the user or application sending requests (e.g., image data) to the inference service.

2. **Inference Servic**e: Flask application running on a Docker container, serving the trained CNN model for predicting digit labels based on input images.

3. **Kubernetes Cluster**: Orchestrates and manages the deployment of the inference service, ensuring scalability and availability.

4. **MLflow**: Tracks and logs experiment metadata and metrics during model training.

5. **Model Training**: Script (train.py) that trains a CNN model using TensorFlow/Keras on the MNIST dataset and logs training details with MLflow.

Flow of Requests:

1. **Client sends image data:** The client (e.g., user interface, external application) sends a POST request containing image data (e.g., handwritten digit) to the deployed inference service endpoint (/predict).

2. **Inference Service processes the request**: The Flask application running on the inference service receives the image data. The inference.py script preprocesses the image (resizing, converting to grayscale) and performs inference using the trained CNN model.

3. **Model Prediction**: The preprocessed image is passed to the loaded model (model.predict), which predicts the digit label (0-9).

4. **Response sent back to Client**: The predicted digit label is returned as JSON in the response to the client.

Interaction with Kubernetes:

The Kubernetes cluster manages the deployment and scaling of the inference service (mnist-inference) based on the defined deployment and service configurations (deployment.yaml, service.yaml).

Directory Structure:

```
mnist_classification/
|
├── train.py # Script for training the CNN model and logging with MLflow
├── inference.py # Script for serving the trained model using Flask
├── Dockerfile # Dockerfile for building the inference image
├── deployment.yaml # Kubernetes deployment configuration
├── service.yaml # Kubernetes service configuration
├── requirements.txt # Python dependencies
├── mlruns/ # MLflow tracking directory (contains experiment logs)
├── model.h5 # Trained CNN model saved in HDF5 format
```

Building and Deploying the MNIST Inference Service:

cd mnist_classification

docker build -t mnist-inference .

docker tag mnist-inference darshan8950/mnist-inference:latest

docker push darshan8950/mnist-inference:latest

kubectl apply -f deployment.yaml

kubectl apply -f service.yaml

Commands Explanation:

1. **docker build -t mnist-inference .:** Builds a Docker image named mnist-inference from the Dockerfile in the current directory (.).

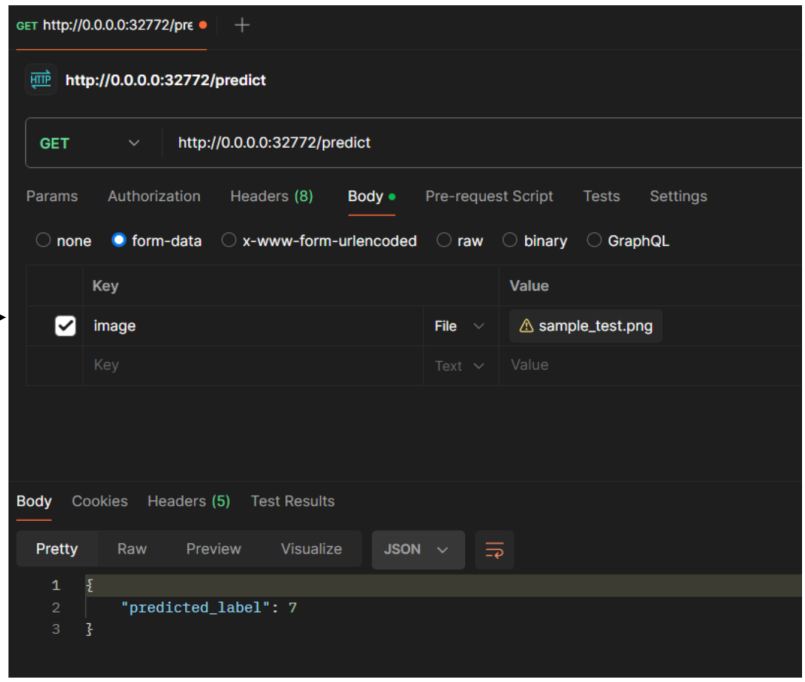
2. **docker tag mnist-inference darshan8950/mnist-inference:latest:** Tags the mnist-inference image with the Docker Hub repository name darshan8950/mnist-inference:latest.

3. **docker push darshan8950/mnist-inference:latest**: Pushes the tagged Docker image to Docker Hub, making it accessible for deployment.

4. **kubectl apply -f deployment.yaml**: Applies the Kubernetes deployment configuration (deployment.yaml) to deploy the inference service.

5. **kubectl apply -f service.yaml**: Applies the Kubernetes service configuration (service.yaml) to expose the deployed service internally.

Input



Output