CS5830: Big Data Laboratory

Assignment 7

Report

Course Instructor: Balaraman Ravindran

Submitted By: Vishal V
Roll Number: ME20B204
Date: 5/05/2024

Indian Institute of Technology Madras

Chennai 600036, India

REPO: https://github.com/visvig/cs5830-a7-2024

Task 1: [25 pts]

Check the modified code for task1.py that now collects counters and gauges

```
app = FastAPI()

# Initialize Prometheus metrics
num_requests = Counter('num_requests', 'Number of requests received', ['method',
'endpoint', 'ip_address'])
processing_time_per_char = Gauge('processing_time_per_char', 'Processing time per
character in microseconds', ['method', 'endpoint'])
memory_usage = Gauge('memory_usage_bytes', 'Memory usage in bytes')
cpu_usage = Gauge('cpu_usage_percent', 'CPU usage percentage')
network_io_sent = Counter('network_io_sent_bytes_total', 'Total number of bytes sent
via network')
network_io_received = Counter('network_io_received_bytes_total', 'Total number of
bytes received via network')

# Setup Prometheus instrumentation for FastAPI
Instrumentator().instrument(app).expose(app)
```

Use `prometheus_task1.yaml` in the repo.

```
# my global config
global:
 scrape_interval: 15s # Set the scrape interval to every 15 seconds. Default is every
1 minute.
 evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1
minute.

# Alertmanager configuration
alerting:
 alertmanagers:
   - static_configs:
       - targets:
         # Uncomment and modify if Alertmanager is configured
         # - "alertmanager:9093"
```

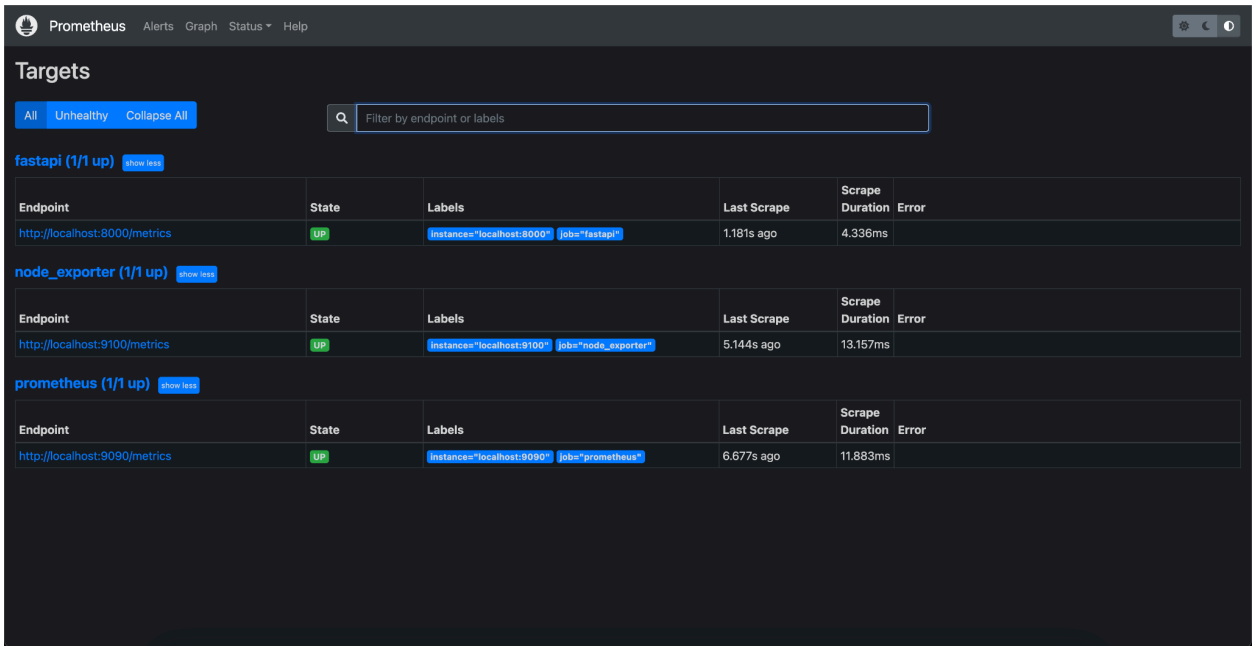
```yaml
# Load rules once and periodically evaluate them according to the global
'evaluation_interval'.
rule_files:
 # Uncomment and list any rule files
 # - "first_rules.yml"
 # - "second_rules.yml"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
 - job_name: "prometheus"
   static_configs:
     - targets: ["localhost:9090"]

 # Add a job to scrape metrics from Node Exporter
 - job_name: "node_exporter"
   static_configs:
     - targets: ["localhost:9100"]

 # Add this job to scrape metrics from FastAPI application
 - job_name: "fastapi"
   scrape_interval: 5s # Scrape more frequently than the default
   static_configs:
     - targets: ["localhost:8000"]  # FastAPI exposes metrics at this port
```

Run in terminal

To check the seamless working of Task 1

python task1.py mnist_model.h5

Prometheus Connection - Successful!



Rough Graphana Dashboard - Successful!

Now Collecting..

```
num_requests_total
processing_time_per_char
memory_usage_bytes
cpu_usage_percent
network_io_sent_bytes_total
```

Graphana Dashboard

Task 2: [25 pts]

Use `prometheus_task2_docker.yaml` from repo

`Requirements.txt`

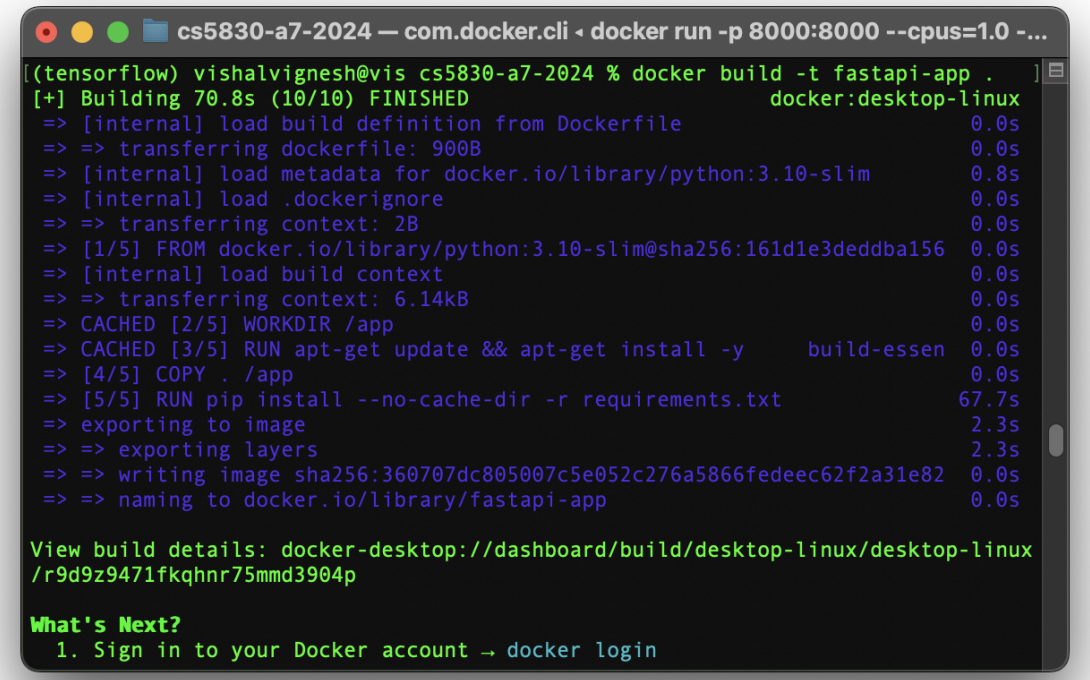
```
fastapi==0.94.1
uvicorn==0.29.0
tensorflow==2.15.0
keras==2.15.0
numpy==1.24.4
Pillow==9.4.0
psutil==5.9.7
prometheus-fastapi-instrumentator==6.1.0
prometheus-client==0.16.0
h5py==3.8.0
python-multipart
```

Dockerfile

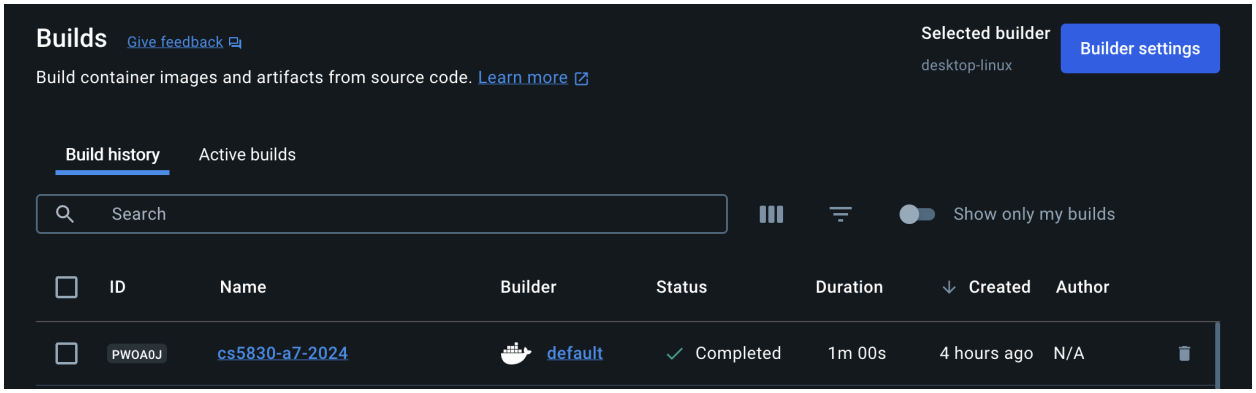
```
# Use an official Python runtime as a parent image
FROM python:3.10-slim

# Set the working directory in the container
WORKDIR /app

# Install system dependencies including HDF5 and pkg-config
RUN apt-get update && apt-get install -y \
   build-essential \
   libhdf5-dev \
   pkg-config  # This package is necessary for h5py installation

# Optionally set HDF5_DIR if h5py cannot find the HDF5 installation
ENV HDF5_DIR=opt/homebrew

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

# Install any needed packages specified in requirements.txt
RUN pip install --no-cache-dir -r requirements.txt

# Make port 8000 available to the world outside this container
```

```
EXPOSE 8000

# Define environment variable
ENV NAME World

# Run app.py when the container launches, including the model path
CMD ["python", "task1.py", "/app/mnist_model.h5"]
```

Run docker build -t fastapi-app

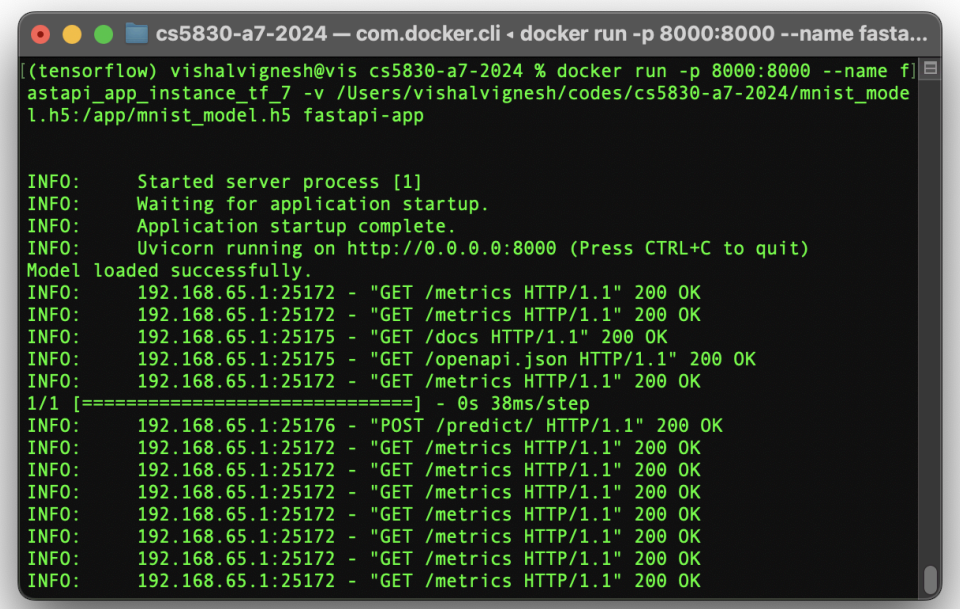


Docker Image Built successfully!



After completing the docker build, check run - Successful!

Run docker run -p 8000:8000 --name fastapi_app_instance_tf_7 -v /Users/vishalvignesh/codes/cs5830-a7-2024/mnist_model.h5:/app/mnist_model.h5 fastapi-app

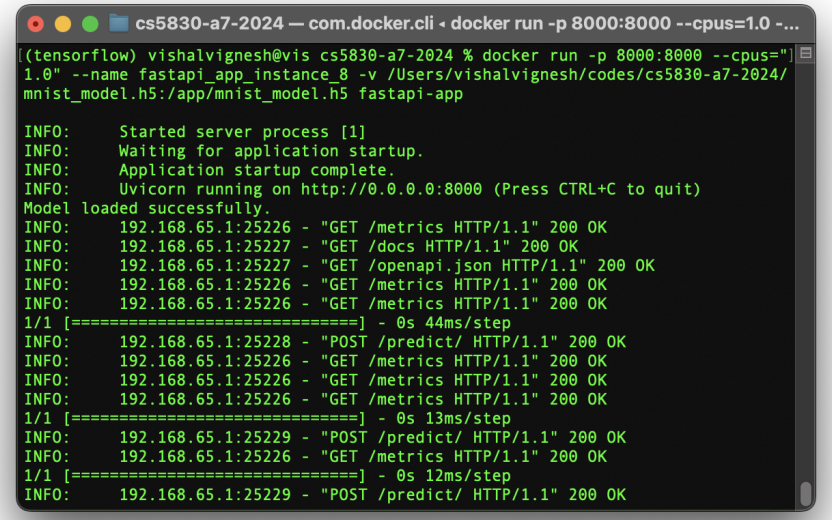


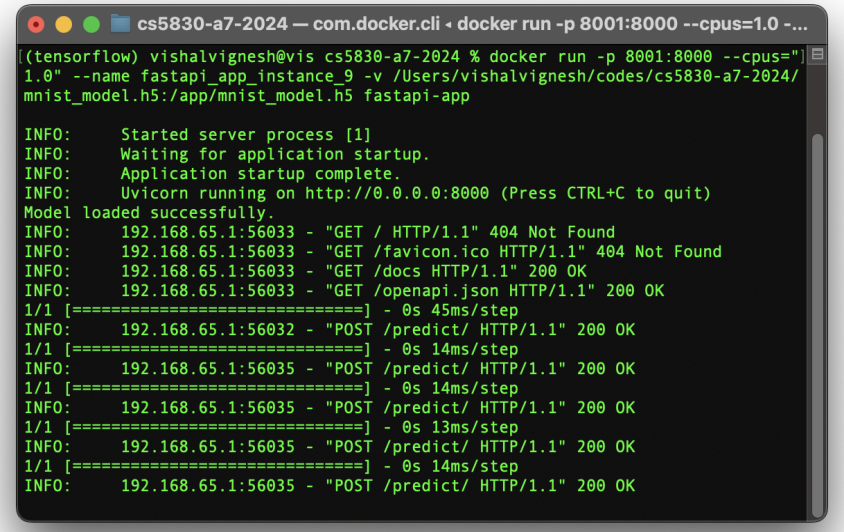
Now 2 docker instances - Successful!

Run docker run -p 8000:8000 --cpus="1.0" --name fastapi_app_instance_8 -v /Users/vishalvignesh/codes/cs5830-a7-2024/mnist_model.h5:/app/mnist_model.h5 fastapi-app

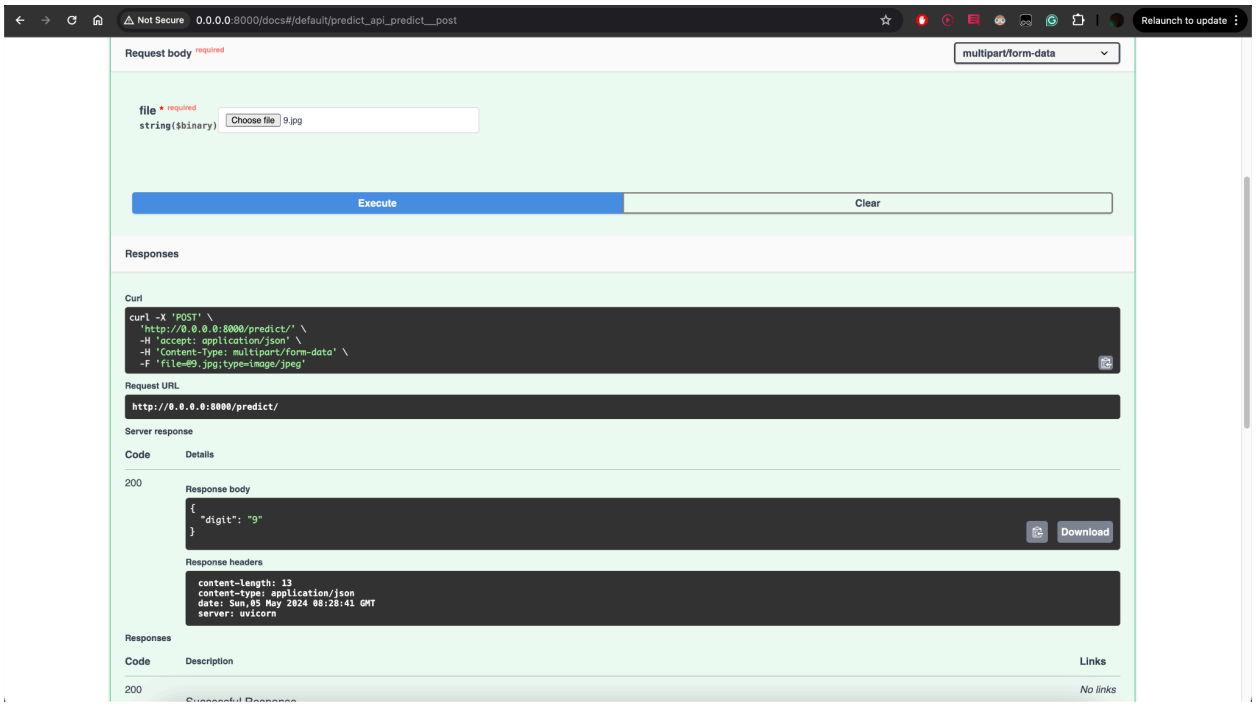
Open another terminal

Run docker run -p 8001:8000 --cpus="1.0" --name fastapi_app_instance_9 -v /Users/vishalvignesh/codes/cs5830-a7-2024/mnist_model.h5:/app/mnist_model.h5 fastapi-app

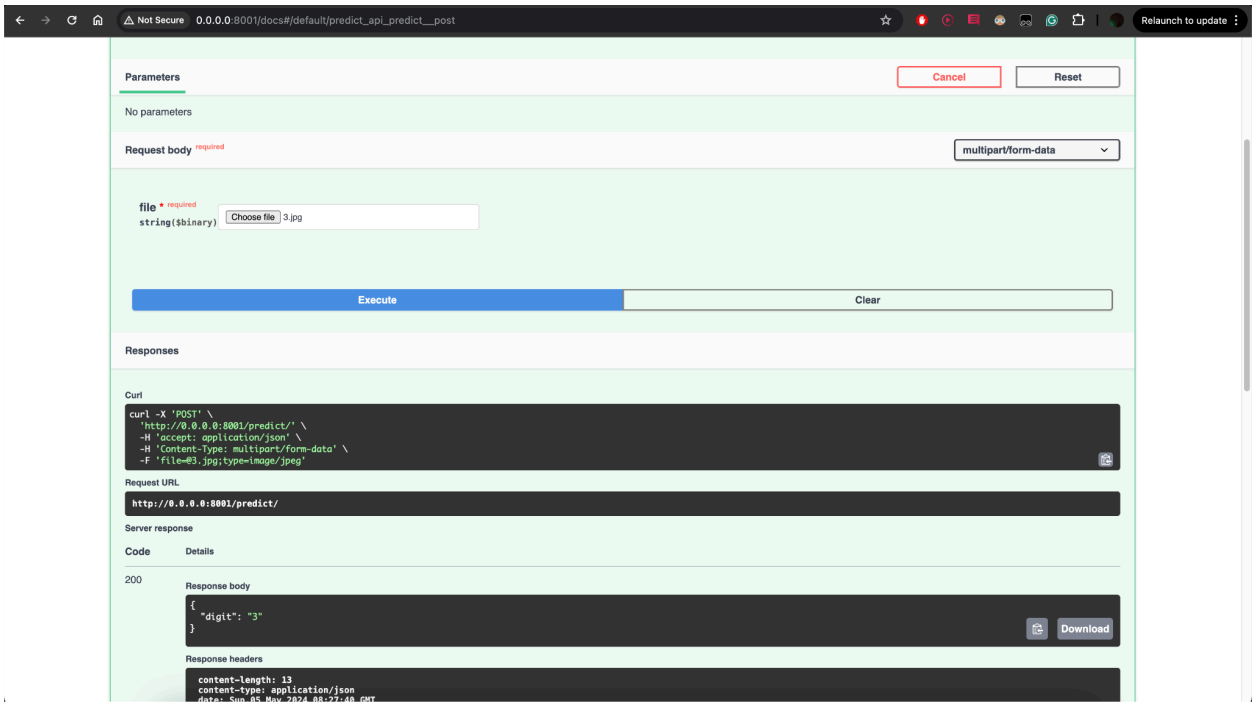




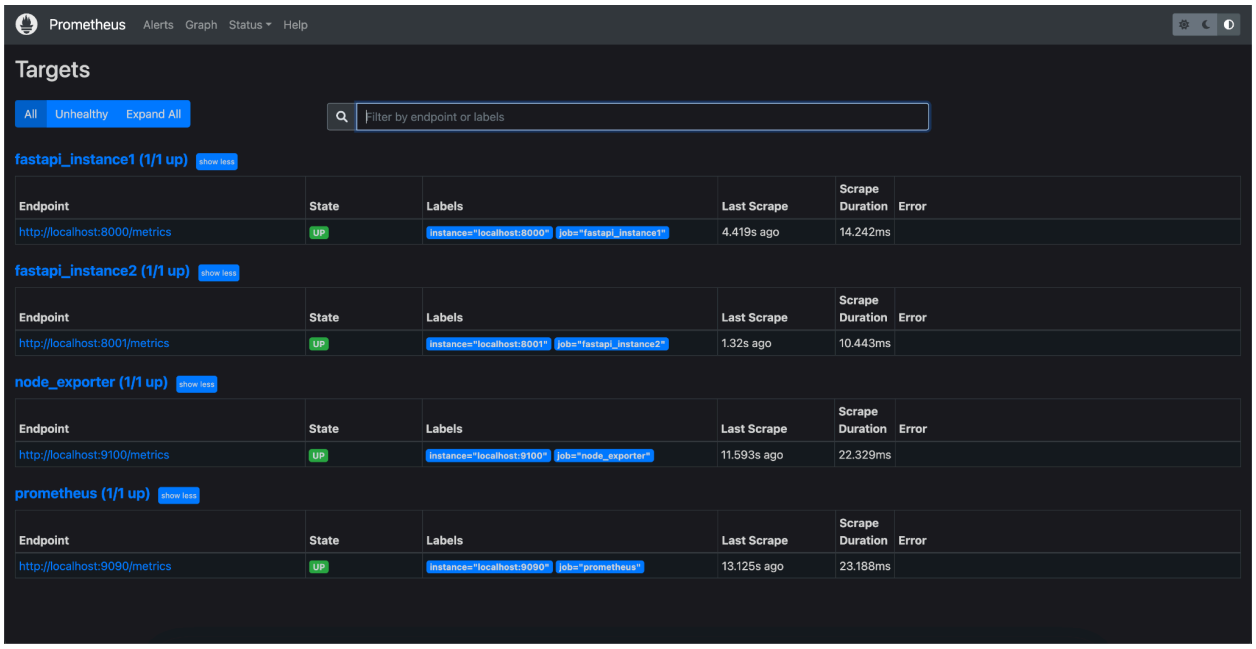
Port 8000



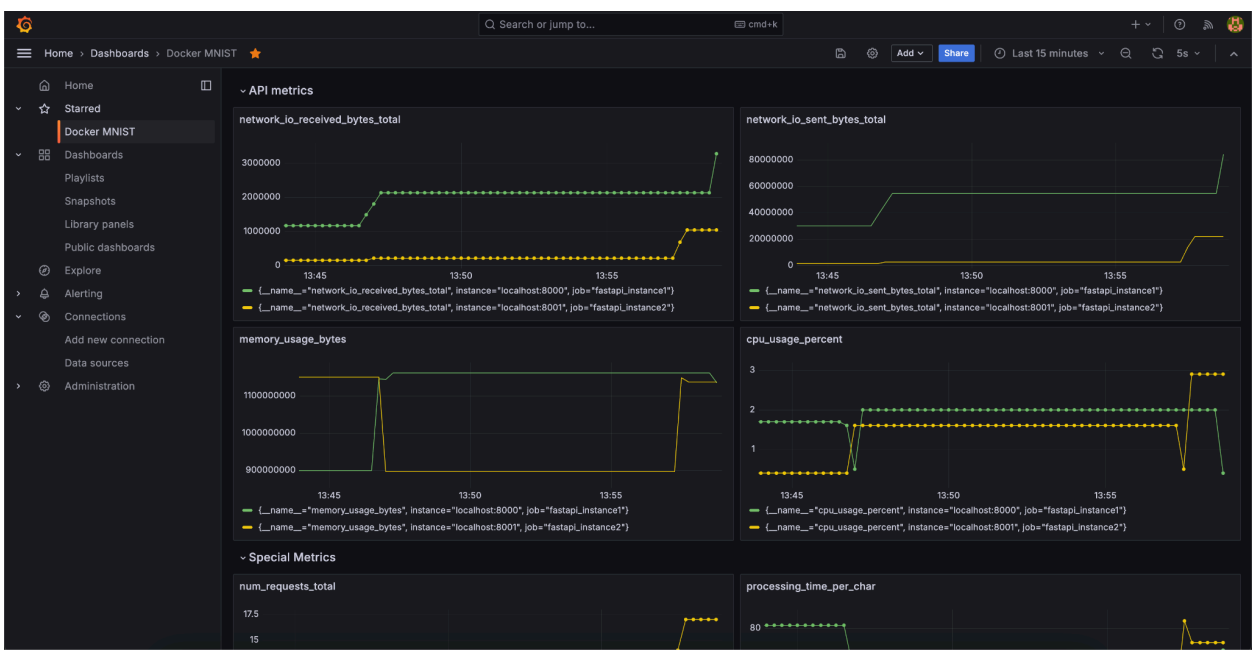
Port 8001

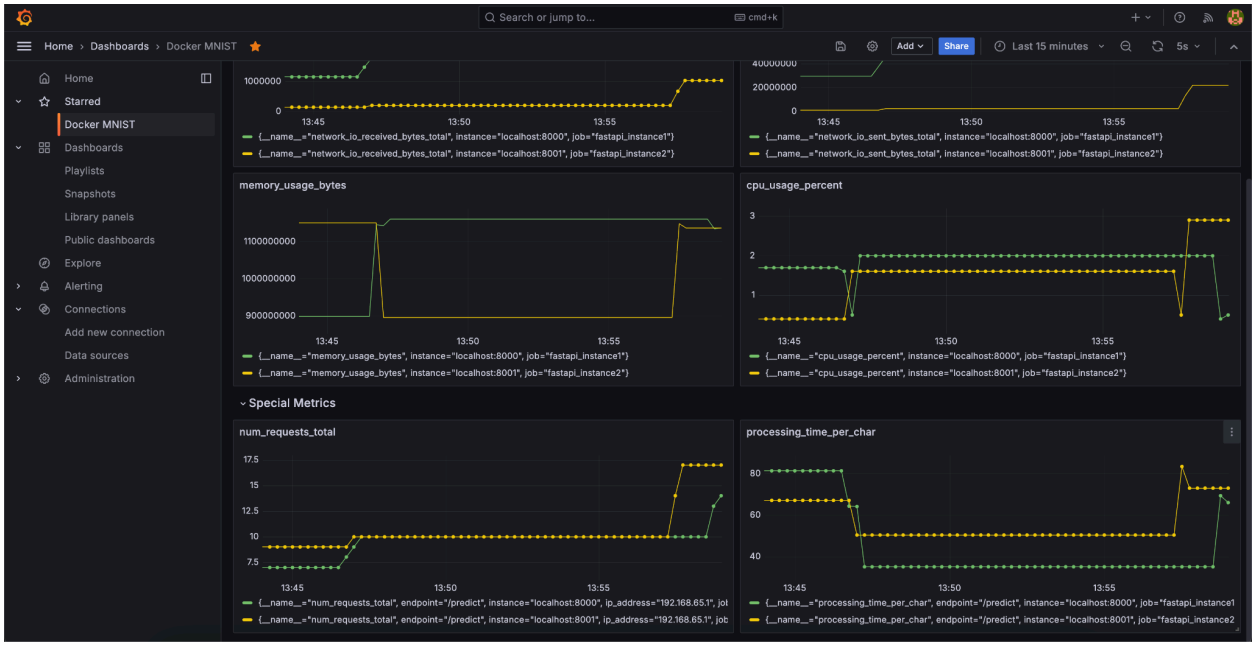


Prometheus now showing 2 instances - Successful!



Graphana Dashboard




Search or jump to...
cmd+k
Home › Dashboards › Docker MNIST
Add
Share
Last 15 minutes
5s
Home
Starred
Docker MNIST
Dashboards
Playlists
Snapshots
Library panels
Public dashboards
Explore
Alerting
Connections
Add new connection
Data sources
Administration
1000000
0
13:45
13:50
13:55
{__name__="network_io_received_bytes_total", instance="localhost:8000", job="fastapi_instance1"}
{__name__="network_io_received_bytes_total", instance="localhost:8001", job="fastapi_instance2"}
40000000
20000000
0
13:45
13:50
13:55
{__name__="network_io_sent_bytes_total", instance="localhost:8000", job="fastapi_instance1"}
{__name__="network_io_sent_bytes_total", instance="localhost:8001", job="fastapi_instance2"}
memory_usage_bytes
1100000000
1000000000
900000000
13:45
13:50
13:55
{__name__="memory_usage_bytes", instance="localhost:8000", job="fastapi_instance1"}
{__name__="memory_usage_bytes", instance="localhost:8001", job="fastapi_instance2"}
cpu_usage_percent
3
2
1
13:45
13:50
13:55
{__name__="cpu_usage_percent", instance="localhost:8000", job="fastapi_instance1"}
{__name__="cpu_usage_percent", instance="localhost:8001", job="fastapi_instance2"}
Special Metrics
num_requests_total
17.5
15
12.5
10
7.5
13:45
13:50
13:55
{__name__="num_requests_total", endpoint="/predict", instance="localhost:8000", ip_address="192.168.65.1", job
{__name__="num_requests_total", endpoint="/predict", instance="localhost:8001", ip_address="192.168.65.1", job
processing_time_per_char
80
60
40
13:45
13:50
13:55
{__name__="processing_time_per_char", endpoint="/predict", instance="localhost:8000", job="fastapi_instance1
{__name__="processing_time_per_char", endpoint="/predict", instance="localhost:8001", job="fastapi_instance2

All Done! Fun!