Lab 8 - Setting up Model Monitoring

Setup Prometheus and Grafana with HELM

Installing Helm

To install helm version 3 on Linux or MacOS, you can follow following instructions.

```
curl https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3 |
bash
```

You could further refer to Official HELM Install Instructions for alternative options.

Verify the installtion is successful,

```
helm --help
helm version
```

Deploy Prometheus Stack with HELM

Read about kube-prometheus-stack 33.1.0 · prometheus/prometheus-community chart at artifacthub.io

Add helm repository using ,

```
helm repo add prometheus-community https://prometheus-community.github.io/
helm-charts

helm repo update
```

Install the helm chart to setup Prometheus and Grafana as,

```
helm upgrade --install prom \
  -n monitoring \
  --create-namespace \
  prometheus-community/kube-prometheus-stack \
  --set grafana.service.type=NodePort \
  --set grafana.service.nodePort=30200 \
  --set prometheus.service.type=NodePort \
  --set prometheus.service.nodePort=30300
```

validate

```
helm list -A
kubectl get all -n monitoring
```

You should be able to access

- prometheus at http://localhost:30300/
- grafana at http://localhost:30200/

Login to grafana with

```
Username: admin
Password: prom-operator
```

Build Model Monitoring Dashboard

Add Instrumentation for FastAPI

`File: src/api/main.py`

```
from fastapi import FastAPI
from fastapi.middleware.cors import CORSMiddleware
from inference import predict_price, batch_predict
from schemas import HousePredictionRequest, PredictionResponse
from prometheus_fastapi_instrumentator import Instrumentator  # 👈 Add this
```

```
# Add CORS middleware
app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

# Initialize and instrument Prometheus metrics
Instrumentator().instrument(app).expose(app)  # 👈 Add this
```

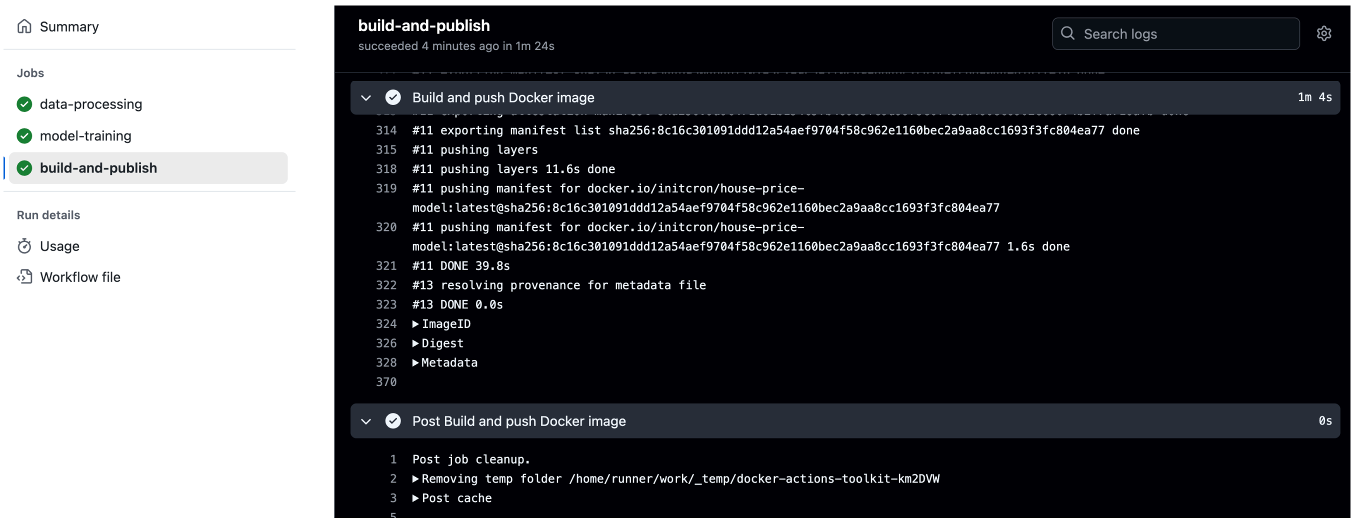
apend this to `requirements.txt`

```
prometheus-fastapi-instrumentator==6.1.0
```

commit and push the changes

```
git add src/api/main.py src/api/requirements.txt
git commit -am "add fastapi instrumentation"
git push origin
```

After pushing the changes, you should see GitHub Actions Pipeline getting triggered ultimately building a new image and publishing it to the DockerHub Repo.



build-and-publish
succeeded 4 minutes ago in 1m 24s

Build and push Docker image 1m 4s

```
#11 exporting manifest list sha256:8c16c301091ddd12a54aef9704f58c962e1160bec2a9aa8cc1693f3fc804ea77 done
#11 pushing layers
#11 pushing layers 11.6s done
#11 pushing manifest for docker.io/initcron/house-price-
model:latest@sha256:8c16c301091ddd12a54aef9704f58c962e1160bec2a9aa8cc1693f3fc804ea77
#11 pushing manifest for docker.io/initcron/house-price-
model:latest@sha256:8c16c301091ddd12a54aef9704f58c962e1160bec2a9aa8cc1693f3fc804ea77 1.6s done
#11 DONE 39.8s
#13 resolving provenance for metadata file
#13 DONE 0.0s
▶ImageID
▶Digest
▶Metadata
```

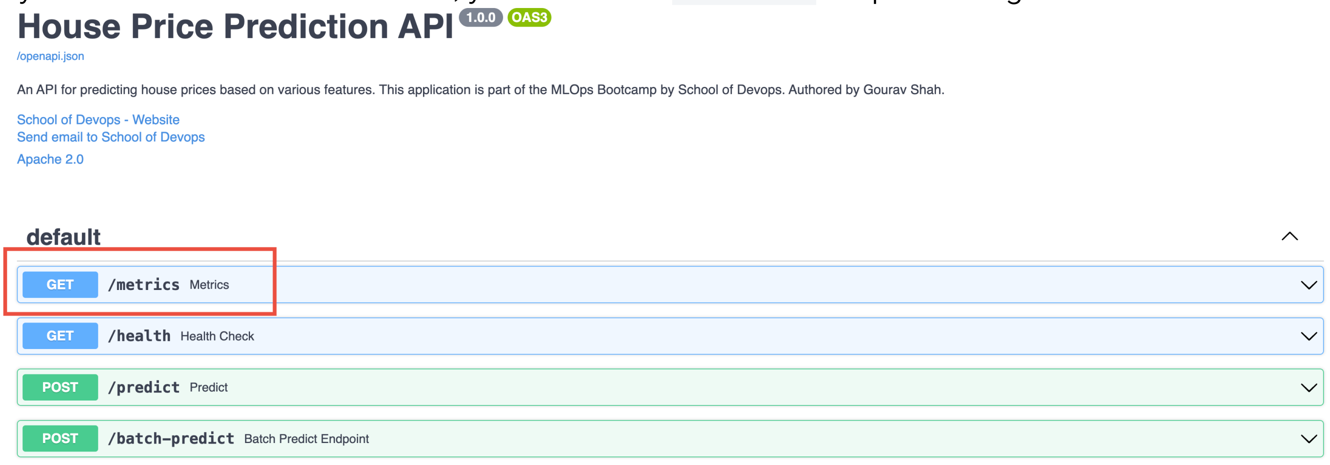
Post Build and push Docker image 0s

```
Post job cleanup.
▶Removing temp folder /home/runner/work/_temp/docker-actions-toolkit-km2DVW
▶Post cache
```

Deploy this change to kubernetes with

```
kubectl rollout restart deployment model
```

If you visit the FastAPI service now, you should see `/metrics` endpoint being added.



House Price Prediction API

An API for predicting house prices based on various features. This application is part of the MLOps Bootcamp by School of Devops. Authored by Gourav Shah.

GET `/metrics` Metrics
GET `/health` Health Check
POST `/predict` Predict
POST `/batch-predict` Batch Predict Endpoint

Scrape FastAPI Metrics using Prometheus Service Monitor

File : `deployment/monitoring/servicemonitor.yaml`

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: house-price-api-monitor
  labels:
    release: prom  # Match the label of your Prometheus instance as per helm
release
spec:
  selector:
    matchLabels:
      app: model
  namespaceSelector:
    matchNames:
      - default  # or your namespace
  endpoints:
    - port: "8000"  # or match name of your service port
      path: /metrics
      interval: 15s
```

apply using

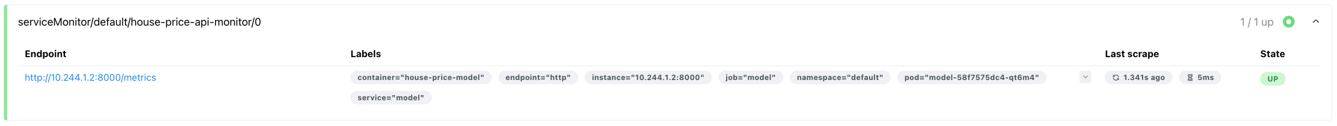
```
kubectl apply -f deployment/monitoring/servicemonitor.yaml
```

check if its been created

```
kubectl get servicemonitor
kubectl describe servicemonitor
```

Validate from Prometheus that the metrics are being sent to prometheus

http://localhost:30300/targets



serviceMonitor/default/house-price-api-monitor/0 — 1 / 1 up

Endpoint	Labels	Last scrape	State
http://10.244.1.2:8000/metrics	container="house-price-model" endpoint="http" instance="10.244.1.2:8000" job="model" namespace="default" pod="model-58f7575dc4-qt6m4" service="model"	1.341s ago 5ms	UP

Also try running following queries on prometheus http://localhost:30300/ if you see the results

```
http_requests_total

histogram_quantile(0.95, sum(rate(http_request_duration_seconds_bucket[1m]))
by (le, handler))
```

```
rate(http_request_size_bytes_sum[1m])
```

[sample output for http_requests_total]

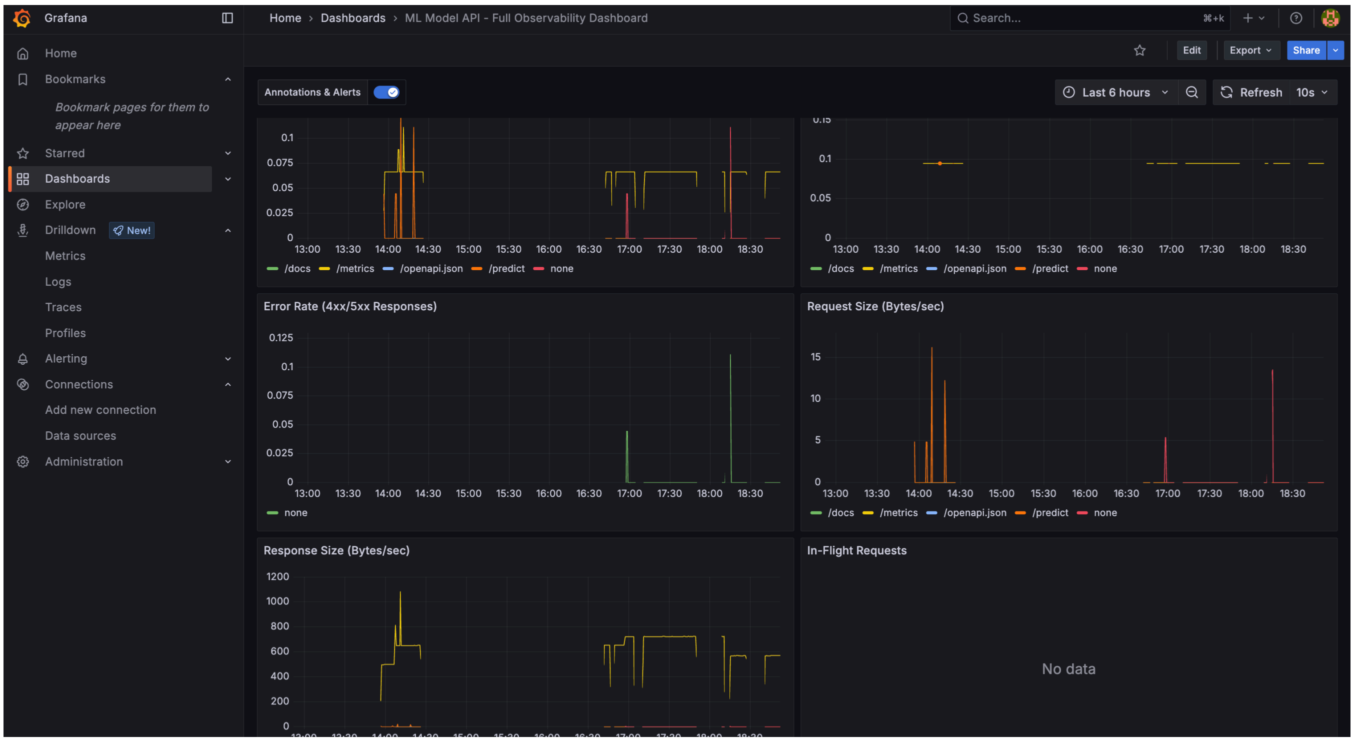
```
http_requests_total{container="house-price-
model", endpoint="8000", handler="none", instance="10.244.2.8:8000",
job="model", method="GET", namespace="default", pod="model-6798556657-
kn6zp", service="model", status="4xx"}
22
http_requests_total{container="house-price-model", endpoint="8000", handler="/
predict", instance="10.244.2.8:8000", job="model", method="POST",
namespace="default", pod="model-6798556657-
kn6zp", service="model", status="2xx"}
2
http_requests_total{container="house-price-model", endpoint="8000", handler="/
metrics", instance="10.244.2.8:8000", job="model", method="GET",
namespace="default", pod="model-6798556657-
kn6zp", service="model", status="2xx"}
29
```

If you see the above queries returning some output, its a confirmation that prometheus is collecting the data from fatapi. Now, you just need to visualize it with Grafana.

To do that,

- Login to Grafana
- Go to Dashboards ➔ New ➔ Import
- In the box which reads `Import via dashboard JSON model` paste the code from enhanced_fastapi_ml_dashboard

You should see a dashboard such as this



This gives you the monitoring which is specific to your model’s performance in terms of latency, number of requests, error rates etc. which could be very useful for you to scale your inference later.

#courses/mlops