Brief Report For Question-2 Solutions

1. Kubernetes configuration files for deploying and scaling the containerized application.

The Kubernetes configuration files for deploying and scaling the containerized application include a Horizontal Pod Autoscaler (HPA) for the CoAP server deployment. The HPA is configured to automatically adjust the number of replicas between 1 and 10, based on CPU utilization. If the average CPU usage exceeds 80%, the HPA will increase the number of replicas. Conversely, if CPU usage falls below 80%, the number of replicas will decrease, within the specified range.

The Kubernetes configuration files for the CoAP server, MQTT broker, Prometheus, and Grafana can be found in the `kubernetes_app/` directory. Configuration files were used to deploy the pods to Kubernetes cluster using:

```
kubectl apply -f deployment.yml
```

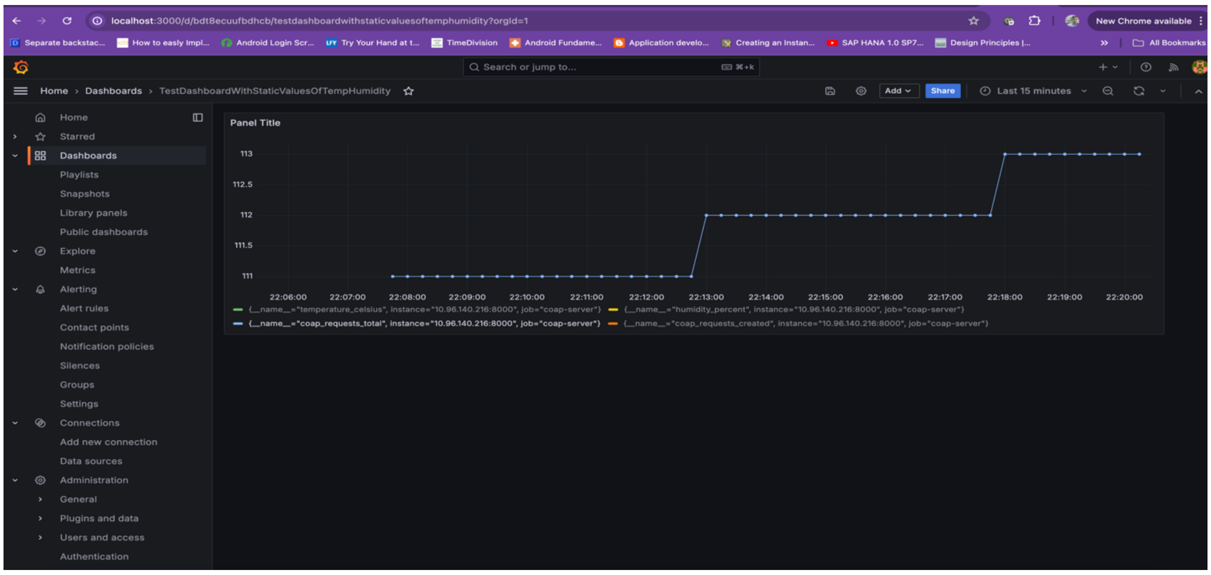
Please use respective yml file in above command.

2. Prometheus and Grafana setup and configuration files, along with screenshots of performance metrics.

Prometheus and Grafana were set up using Helm, the package manager for Kubernetes. To complete the setup, account and role bindings were required, and the necessary YAML files for this configuration are available in the `kubernetes_app/PrometheusConfig_Deployment` directory. Below documentation was used as reference:

https://grafana.com/docs/grafana-cloud/monitor-infrastructure/kubernetes-monitoring/configuration/config-other-methods/prometheus/prometheus-operator/

Below is a screenshot illustrating the number of CoAP requests received in the last 15 minutes.


localhost:3000/d/bdt8ecuufbdhcb/testdashboardwithstaticvaluesoftemphumidity?orgId=1
Home › Dashboards › TestDashboardWithStaticValuesOfTempHumidity
Last 15 minutes
Panel Title
113
112.5
112
111.5
111
22:06:00
22:07:00
22:08:00
22:09:00
22:10:00
22:11:00
22:12:00
22:13:00
22:14:00
22:15:00
22:16:00
22:17:00
22:18:00
22:19:00
22:20:00
{__name__="temperature_celsius", instance="10.96.140.216:8000", job="coap-server"}
{__name__="humidity_percent", instance="10.96.140.216:8000", job="coap-server"}
{__name__="coap_requests_total", instance="10.96.140.216:8000", job="coap-server"}
{__name__="coap_requests_created", instance="10.96.140.216:8000", job="coap-server"}

3. Source code for the machine learning model (Python with TensorFlow/PyTorch) and data processing scripts (Pandas).

The source code for the machine learning model, developed using TensorFlow, is available in the data_science/ directory. The weather_prediction.py script is responsible for loading weather data, processing it, and training a linear regression model to predict temperature. The script also evaluates the model's performance using various metrics, visualizes the predictions, saves the trained model, and provides an example of how to load the model for future use.

4. Integration of the machine learning model with the IoT system to provide real-time forecasts.

The machine learning model is integrated with the IoT system through a Flask API, which is hosted in a Docker container deployed on a Kubernetes cluster. The IoT system can send real-time data to this API, and the model provides real-time predictions for forecasting purposes. While this setup is adequate for basic deployment and real-time forecasting, additional considerations such as security, scalability, and monitoring are necessary for a production environment.

The machine learning model is implemented in data_science/weather_prediction.py and is loaded from a pickle file within the Flask API, which is handled in data_science/flask_app.py.