Prometheus Assignment **Vikas K R**

Question:

1. **Export the metrics (like request per second, memory usage, cpu usage etc) in the existing mini project given to Interns**
2. **Install Prometheus and Grafana using Docker (with docker-compose)**
3. **Configure prometheus (scrape configs) such way that it can scrape the metrics from default metric path of the application job**
4. **Validate the entire configuration to check if the data is coming or not in Prometheus UI**
5. **Create the Dashboards in Grafana on top of the metrics exported by adding the Prometheus as a Datasource.**

Solution:

Step 1:

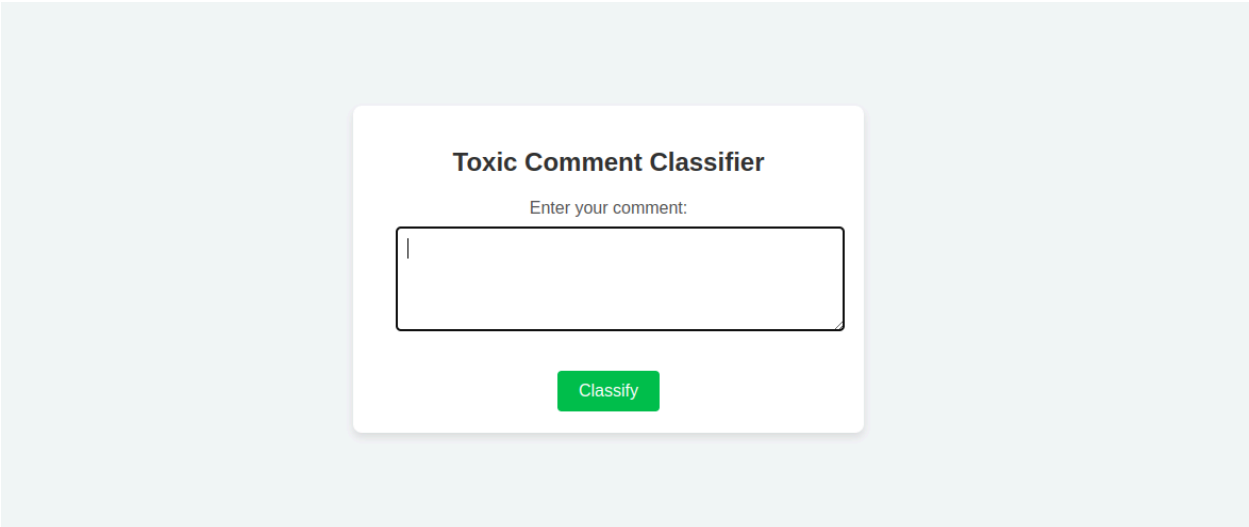
Application overview:

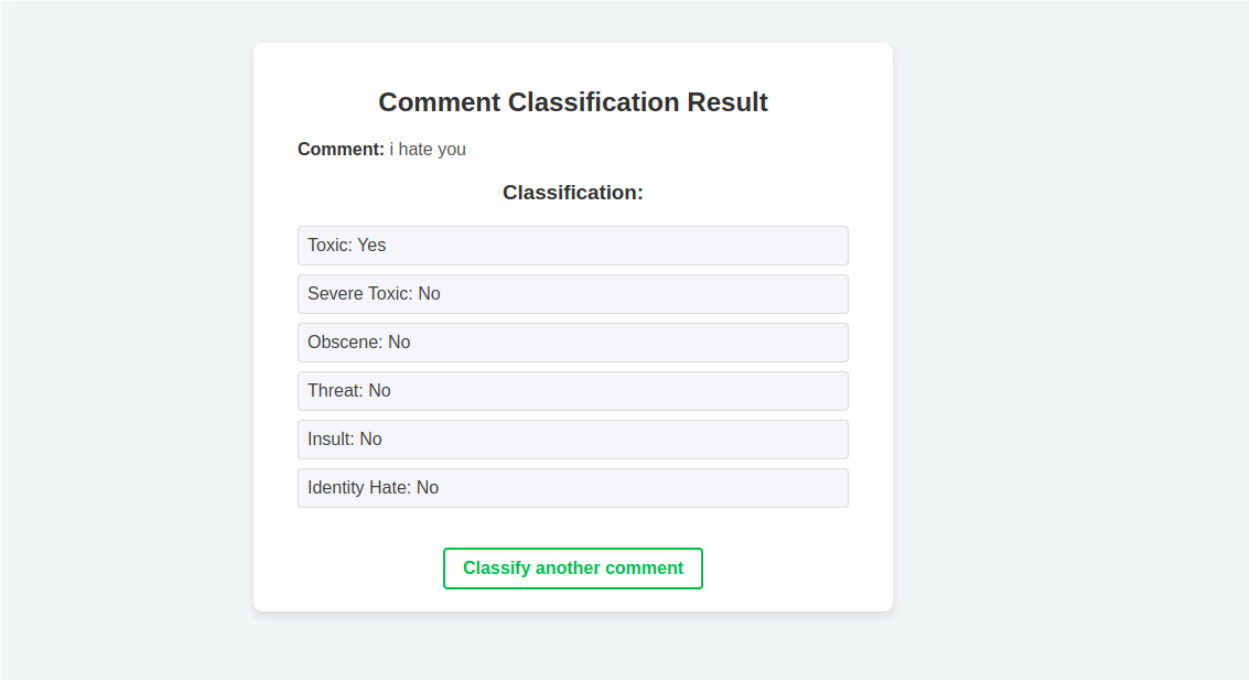
I created an ML-based Flask application that collects a comment from the user and detects whether the comment is toxic. If it is toxic, the application classifies it as severe toxicity, insult, threat, or other categories using logistic regression. I also integrated MLflow for experiment tracking, and my model achieved an accuracy of 93%.

```
from flask import Flask, render_template, request
import joblib
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
import prometheus_client  # Import Prometheus client for metrics
# import mlflow
app = Flask(__name__)


# model_uri = "models:/toxic_comment_model/1"
# model = mlflow.pyfunc.load_model(model_uri)
```

```
# Load model and vectorizer
model = joblib.load("toxic_comment_model.pkl")
tfidf = joblib.load("tfidf_vectorizer.pkl")

# Initialize Prometheus metrics
REQUEST_COUNT = prometheus_client.Counter('request_count', 'Total
number of requests')
COMMENT_COUNT = prometheus_client.Counter('comment_count', 'Total
number of comments submitted')

@app.route('/')
def home():
   return render_template('index.html')

@app.route('/predict', methods=['POST'])
def predict():
   if request.method == 'POST':
       comment = request.form['comment']

       # Increment comment count
       COMMENT_COUNT.inc()

       # Increment request count
       REQUEST_COUNT.inc()

       comment_tfidf = tfidf.transform([comment])
       prediction = model.predict(comment_tfidf).tolist()[0]
          categories = ["Toxic", "Severe Toxic", "Obscene", "Threat",
"Insult", "Identity Hate"]
             result = {categories[i]: bool(prediction[i]) for i in
range(len(categories))}

               return render_template('result.html', comment=comment,
result=result)

@app.route('/metrics')
def metrics():
   return prometheus_client.generate_latest()
```

```
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5001, debug=True)
```





Below two collect number comment and number of request

```
REQUEST_COUNT = prometheus_client.Counter('request_count', 'Total number of requests')
COMMENT_COUNT = prometheus_client.Counter('comment_count', 'Total number of comments submitted')
```

And then i created docker file which contain

```
FROM python:3.9-slim
WORKDIR /app
COPY . /app
RUN pip install --no-cache-dir -r requirements.txt
EXPOSE 5001
CMD ["python", "app.py"]
```

My application's Docker image **`vikaskarbail/toxic-comment-classification`**. Below are the Kubernetes deployment and service files for the application, where I exposed the service using NodePort.

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
 name: toxic-comment-classification
 labels:
   app: toxic-comment-classification
spec:
 replicas: 1
 selector:
   matchLabels:
     app: toxic-comment-classification
 template:
   metadata:
     labels:
       app: toxic-comment-classification
   spec:
     containers:
     - name: toxic-comment-classification
       image: vikaskarbail/toxic-comment-classification:latest
       ports:
       - containerPort: 5001
```

Service.yaml

```yaml
apiVersion: v1
kind: Service
```

```yaml
metadata:
 name: toxic-comment-classification-service
spec:
 type: NodePort
 selector:
   app: toxic-comment-classification
 ports:
   - port: 5001
     targetPort: 5001
     nodePort: 30000
```

to deploy the deployment file
kubectl apply -f todo-deployment.yaml
to deploy the service file
kubectl apply -f todo-service.yaml

Step 2: Helm Chart

To install Prometheus and Grafana in your Kubernetes cluster, use Helm. First, add the necessary Helm repositories and update them. Then, install Prometheus and Grafana using the following commands:

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

# Install Prometheus
helm install prometheus prometheus-community/prometheus

# Install Grafana
helm install grafana grafana/grafana
```

Step 3: Configure Prometheus Scrape Configs

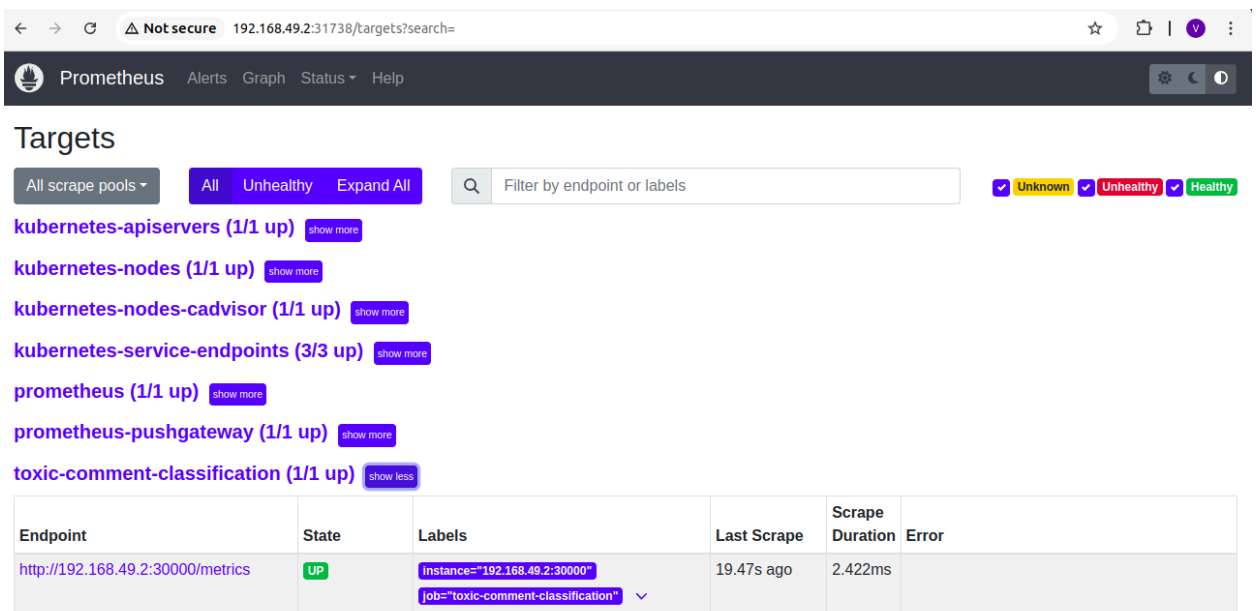
We can customize the scrape configuration by editing the values.yaml of the Prometheus Helm release or overriding it during installation.

```
- job_name: toxic-comment-classification
    static_configs:
        - targets: ["192.168.49.2:30000"]
```

Step 4: Accessing and Configuring Prometheus and Grafana in Minikube

To access the Prometheus UI, Grafana UI, and Toxic Comment Application UI, use the Minikube IP along with their respective NodePorts.

To add Prometheus as a data source in the Grafana UI, navigate to **Configuration > Data Sources**, click **Add data source**, and select **Prometheus**. Set the URL to http://prometheus-server

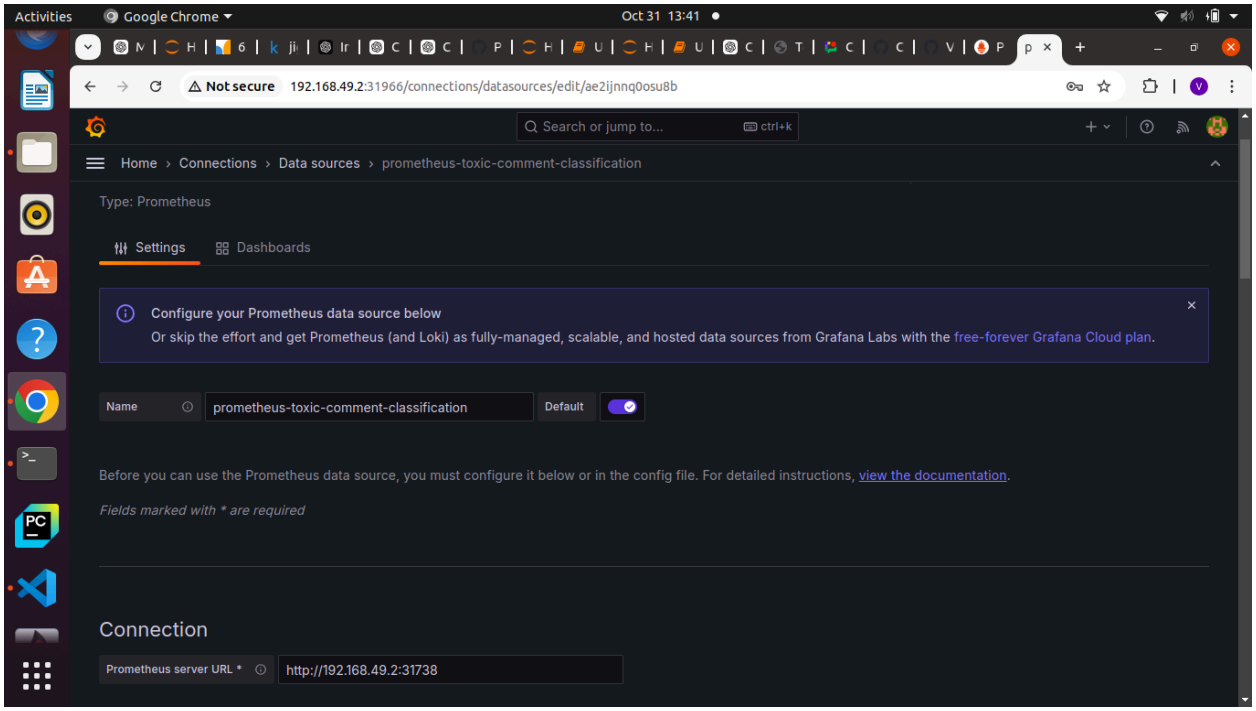


Step 5: Create Dashboards in Grafana

To create dashboards in Grafana, navigate to **Create > Dashboard**. From there, add panels to visualize key metrics such as *Total HTTP Requests*, *Memory Usage*, and *CPU Usage*.

> Metrics from the application

> Grafana Setting



> Dashboard: Cpu usage, Memory Usage, Number of comments