**CSYE 7220 – Final Project**

**Team Members:**

* **Anjali Sajeevan**
* **Somanwita Dey**

**Aim of the Project:**

Deploy a twitter application using docker image on Azure Elastic Kubernetes service (aks) with terraform, perform HPA autoscaling based on cpu and memory, monitoring with Prometheus and Grafana and send messages to slack using alert manager. Locust was used for load testing.

**Result:**

The twtr-be application docker image was created using the below commands:

* docker build -f Dockerfile-dev -t anjalisajeev/twtr-be .
* docker push anjalisajeev/twtr-be

Created the terraform files for Azure Elastic Kubernetes service and ran the terraform commands:

* terraform init
* terraform plan
* terraform apply

It was successfully created and viewed in azure as shown below:

Graphical user interface, text, application, email

Description automatically generated

After creating, the below command was run to create a config-terraform-aks-prometheus file in the current folder. Then copy that file and go to users/anjal/.kube and replace the config file with this one(or rename it to config)

* terraform output kube\_config > config-terraform-aks-prometheus

then, ran the below kubectl commands to create the deployment and service with the combined yaml file.

* kubectl apply -f twtr-combo.yaml

The file contains the limits and request values for memory and cpu resources

Table

Description automatically generated

The below screenshots show the nodes, deploy, pods and service available.

* kubectl get nodes
* kubectl top nodes
* kubectl get deploy --all-namespaces
* kubectl get pods

A picture containing graphical user interface

Description automatically generated

Graphical user interface, text

Description automatically generated

After getting the external ip, open the webpage using the ip : 40.112.134.105

Graphical user interface, text, application

Description automatically generated with medium confidence

**Metric Server:**

Metrics server was installed to collect the resource metrics

* kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml
* kubectl get pods -n kube-system -l k8s-app=metrics-server

**Kube state metrics:**

Kube state metrics was installed to talk to Kubernetes API server to get all the details about all the API objects like deployments, pods etc.

* git clone https://github.com/devopscube/kube-state-metrics-configs.git
* kubectl apply -f kube-state-metrics-configs/
* kubectl getdeployments kube-state-metrics -n kube-system

A computer screen capture

Description automatically generated with medium confidence

**Horizontal Autoscaling:**

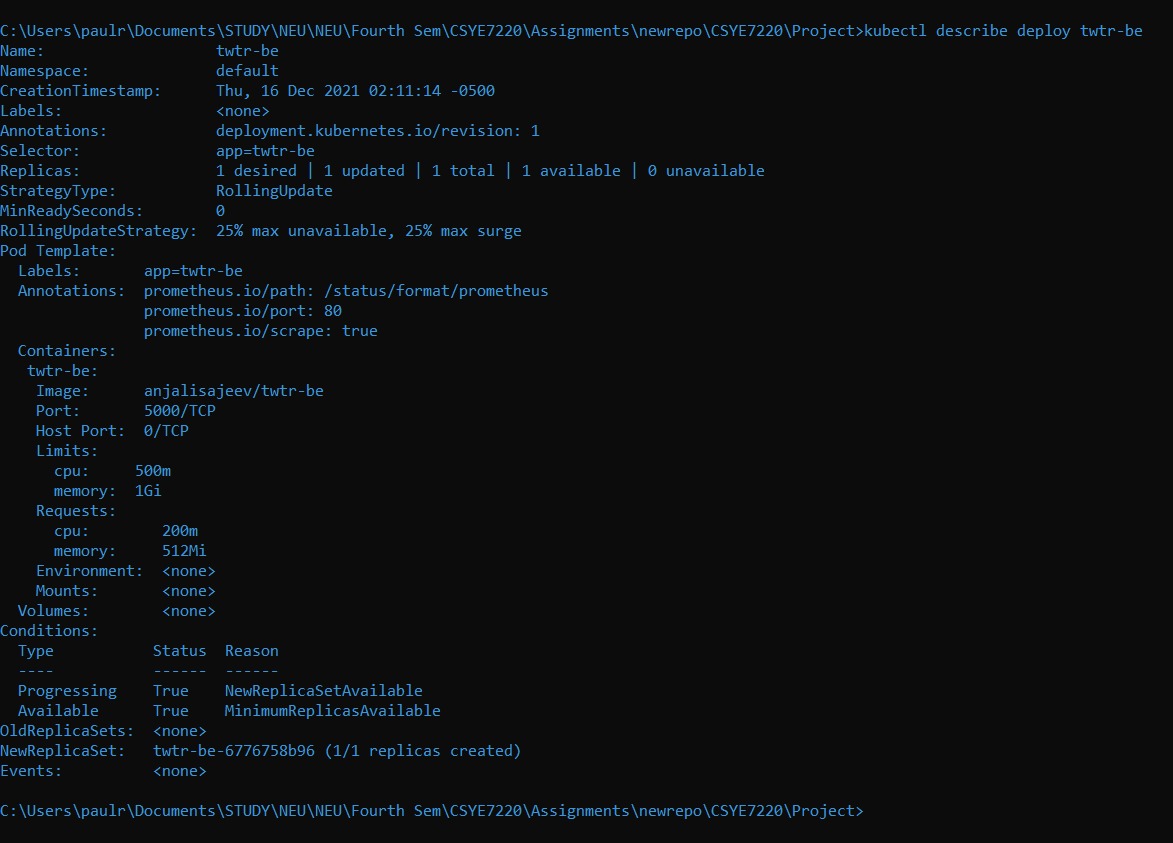
It means raising the amount of your instance after a target value is reached.

The Horizontal Pod Autoscaler automatically scales the number of Pods in a replication controller, deployment, replica set or stateful set based on observed CPU utilization

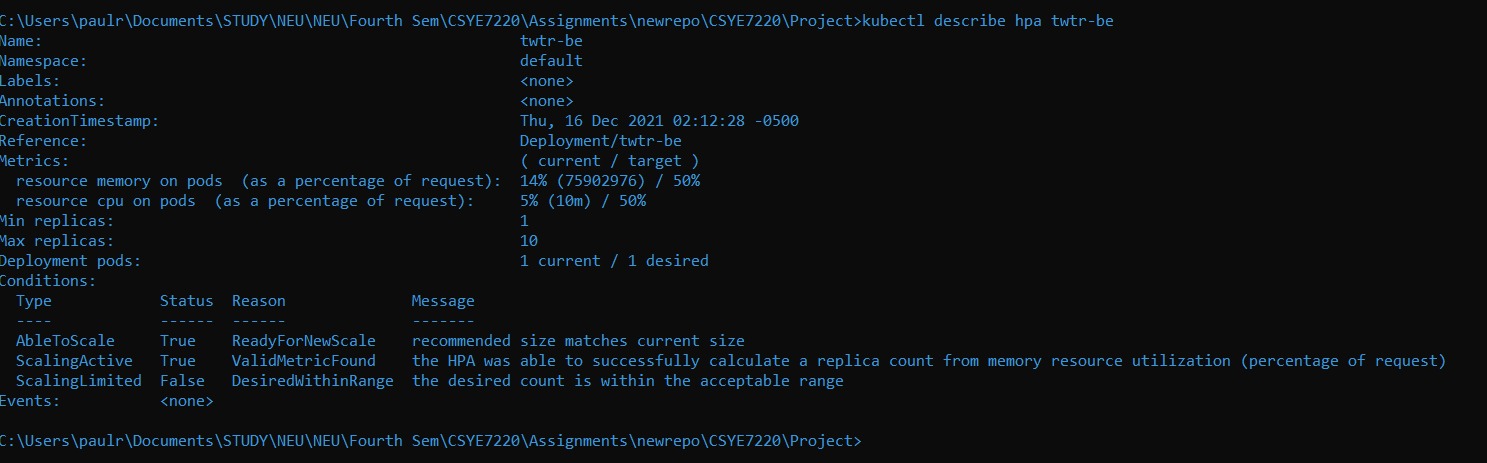
A twtr-scalar.yaml file was created specifying the cpu and memory resources

A picture containing text

Description automatically generated



After the hpa file was created ran the below command, we get a detailed monitoring of the current cpu and memory availability with the desired levels, we can also see the current replica and the desired replicas.



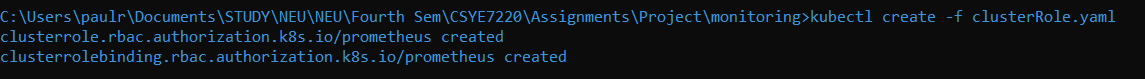
**Prometheus:**

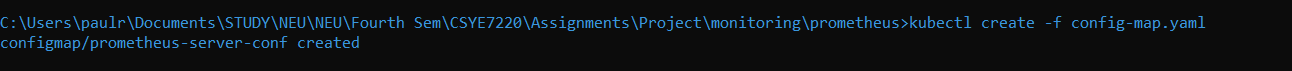
[Prometheus](https://sensu.io/blog/how-to-collect-prometheus-metrics-and-store-them-anywhere-with-sensu) is a monitoring solution for recording and processing any purely numeric time-series. It gathers, organizes, and stores metrics along with unique identifiers and timestamps

Created a Kubernetes namespace for all monitoring components:

Created the below yaml files for prometheus configuration:

* clusterRole.yaml
* config-map.yaml
* prometheus-deployment.yaml
* prometheus-service.yaml





A computer screen capture

Description automatically generated with medium confidence

Text

Description automatically generated

Port forwarding was done to access the Prometheus dashboard from the workstation

Text

Description automatically generated with low confidence

The Prometheus dashboard was accessed using localhost:8080

Graphical user interface, text, application

Description automatically generated

Table

Description automatically generated

**Grafana:**

Grafana is a Web dashboard used by many organizations to monitor Kubernetes

Created the below yaml files for grafana configuration:

* grafana-datasource-config.yaml
* grafana-datasource-deploy.yaml
* grafana-datasource-service.yaml

After creating the service file, the below command was run to view the external IP.

* kubectl get svc --namespace=monitoring

Text

Description automatically generated

The Grafana login page opens with the external ip and then we can login to view the dashboard. A template was imported, and the data source selected as Prometheus.

Graphical user interface

Description automatically generated

**Locust:**

Locust was used for load testing.

A locust.py file was created, and the below command was run:

* C:\Users\anjal\AppData\Local\Programs\Python\Python310\Scripts\locust.exe -f locustfile.py --host=http://40.83.204.246/

A screenshot of a computer screen

Description automatically generated with medium confidence

The locust homepage was opened using <http://localhost:8089/>

A new test was started to increase the cpu usage:

Graphical user interface, text, application, chat or text message

Description automatically generated

Once it started swamping, we monitored the rps:

Chart, histogram

Description automatically generated

Text

Description automatically generated with medium confidence

Once the locust started swamping, the replicas increased to 3 due to the increased load. This was also monitored by Grafana.

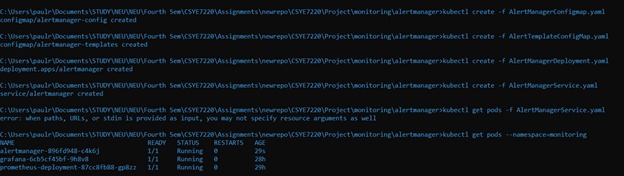
**ALERT MANAGER**

Alert Manager is an open-source configuration that works with Prometheus.

Created the below yaml files for alert manager configuration:

* AlertManagerConfigmap.yaml
* AlertManagerDeployment.yaml
* AlertManagerService.yaml

The configuration map is configured to send alerts to a project channel in CSYE6220 slack.



Port forwarding was done to access the alertmanager dashboard from the workstation

The alertmanager dashboard can be viewed using localhost:9093

Graphical user interface, text, application, email

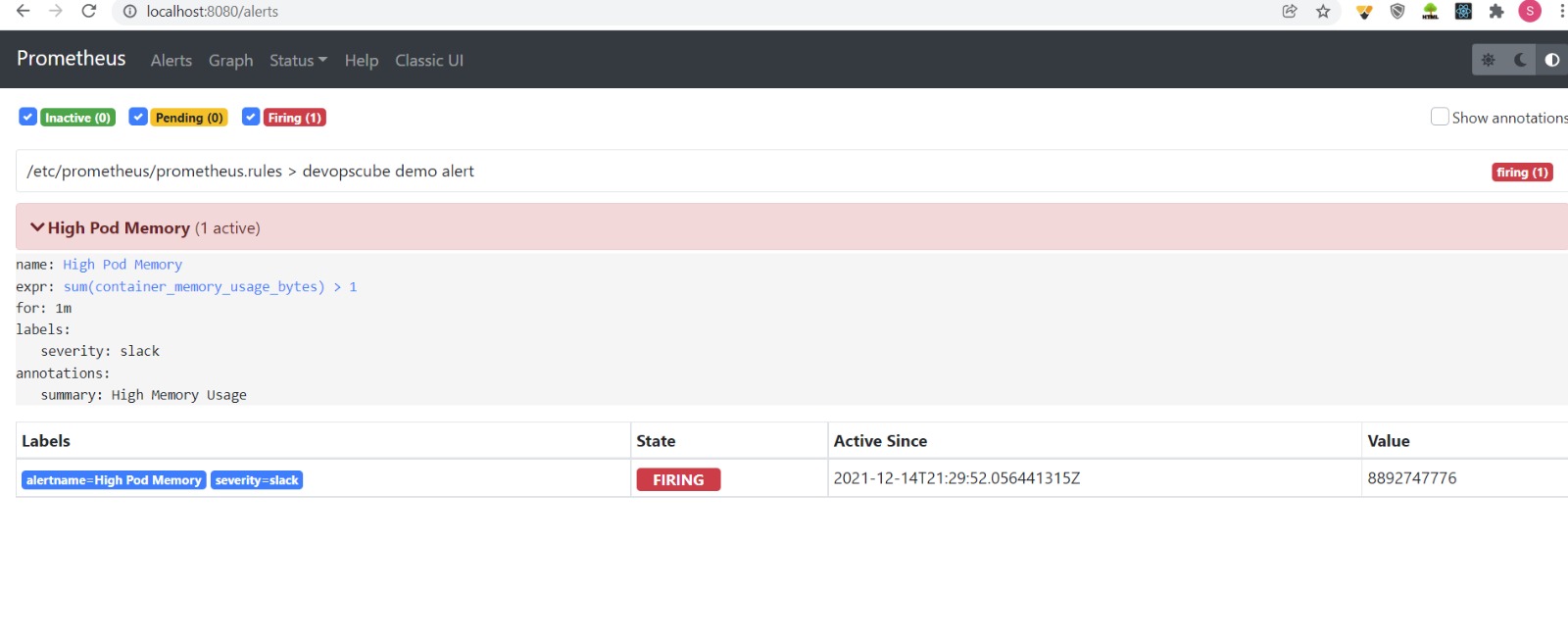
Description automatically generated

In the Prometheus config-map.yaml we had configured the high pod memory alert if its >1

Graphical user interface, application, Word

Description automatically generated

The alerts are visible in Prometheus alert tab



And also available in slack:

Graphical user interface, text, application, email

Description automatically generated