CSYE 7220 – Final Project

Team Members:

- **Anjali Sajeevan - 001563277**
- **Somanwita Dey - 001399438**

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:

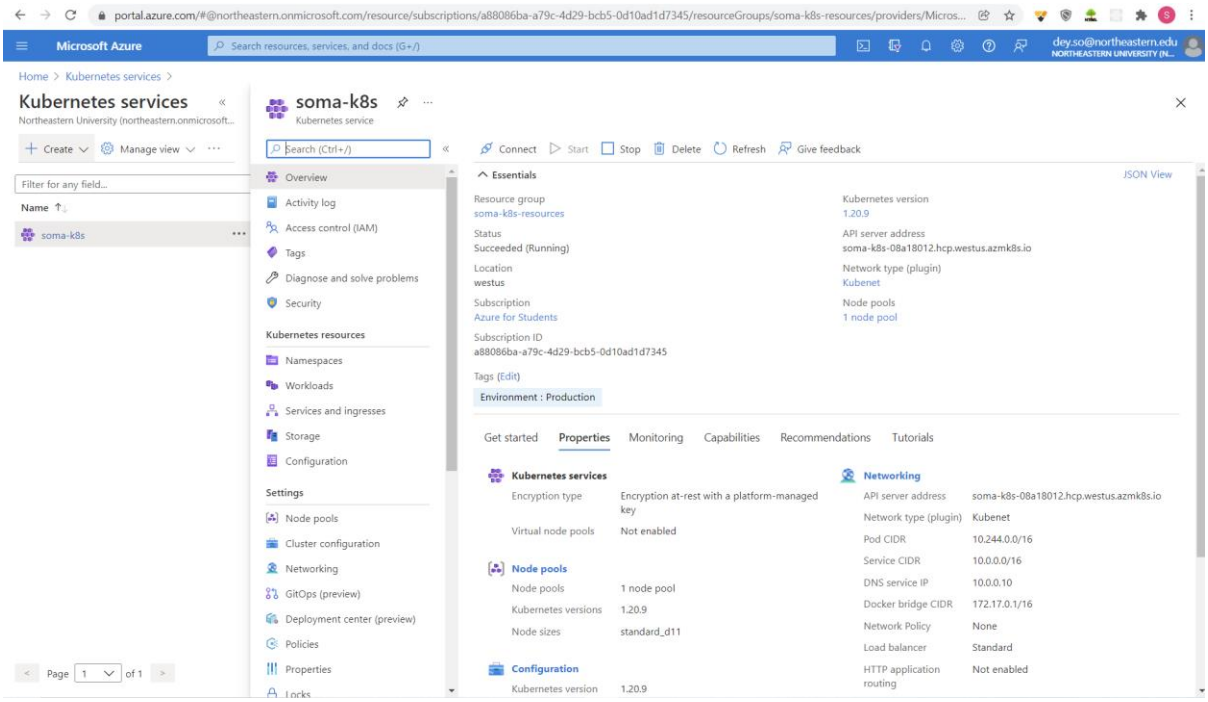
The twtr-be application docker image was created using the below commands:

- 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:



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

twtr-combo.yaml | twtr-scaler.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: twtr-be
spec:
  selector:
    matchLabels:
      app: twtr-be
  replicas: 1
  template:
    metadata:
      annotations:
        prometheus.io/path: "/status/format/prometheus"
        prometheus.io/scrape: "true"
        prometheus.io/port: "80"
      labels:
        app: twtr-be
    spec:
      containers:
      - name: twtr-be
        image: "anjalisajeev/twtr-be"
        ports:
        - containerPort: 5000
        resources:
          limits:
            cpu: .5
            memory: 1Gi
          requests:
            memory: 0.5Gi
            cpu: .2
---
apiVersion: v1
kind: Service
metadata:
  name: twtr-be
  labels:
    app: twtr-be
spec:
  type: LoadBalancer
  ports:
  - port: 80
    targetPort: 5000
  selector:
    app: twtr-be
```

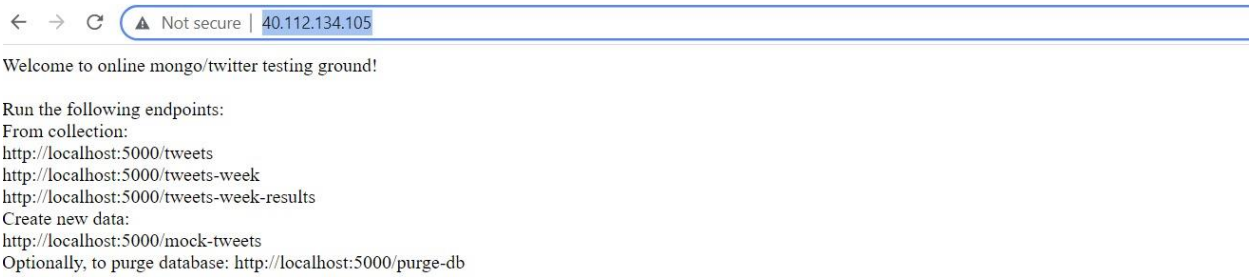
The below screenshots show the nodes, deploy, pods and service available.

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

```
C:\Users\paulr\Documents\STUDY\NEU\NEU\Fourth Sem\CSYE7220\Assignments\Project\tf-azure>terraform output kube_config > c:/Users/paulr/.kube/config-terraform-aks-prometheus

C:\Users\paulr\Documents\STUDY\NEU\NEU\Fourth Sem\CSYE7220\Assignments\Project\tf-azure>kubectl get nodes
NAME                                STATUS   ROLES   AGE    VERSION
aks-default-25096251-vmss000000   Ready    agent   3m7s   v1.20.9

C:\Users\paulr\Documents\STUDY\NEU\NEU\Fourth Sem\CSYE7220\Assignments\Project\tf-azure>kubectl top nodes
W1214 12:28:07.269282   32156 top_node.go:119] Using json format to get metrics. Next release will switch to protocol-buffers, switch early by passing --use-protocol-buffers flag
NAME                                CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
aks-default-25096251-vmss000000   201m         10%    790Mi           7%

C:\Users\paulr\Documents\STUDY\NEU\NEU\Fourth Sem\CSYE7220\Assignments\Project\tf-azure>kubectl get deploy --all-namespaces
NAMESPACE     NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
kube-system   coredns               2/2     2            2           4m52s
kube-system   coredns-autoscaler    1/1     1            1           4m52s
kube-system   metrics-server        1/1     1            1           4m52s
kube-system   tunnelfront           1/1     1            1           4m52s
```

```
C:\Users\paulr\Documents\STUDY\NEU\NEU\Fourth Sem\CSYE7220\Lecture 11\lab\FWO-TEST>kubectl get pods
NAME                                                READY   STATUS    RESTARTS   AGE
twtr-be-6776758b96-d2wd8                            1/1     Running   0          7h31m
twtr-be-release-prometheus-adapter-c695bdd76-bq5gr  1/1     Running   0          44h

C:\Users\paulr\Documents\STUDY\NEU\NEU\Fourth Sem\CSYE7220\Lecture 11\lab\FWO-TEST>kubectl get svc
NAME                                TYPE           CLUSTER-IP    EXTERNAL-IP      PORT(S)        AGE
kubernetes                          ClusterIP      10.0.0.1      <none>           443/TCP        45h
twtr-be                             LoadBalancer   10.0.231.49   40.112.134.105   80:32645/TCP   8h
twtr-be-release-prometheus-adapter  ClusterIP      10.0.137.59   <none>           443/TCP        45h

C:\Users\paulr\Documents\STUDY\NEU\NEU\Fourth Sem\CSYE7220\Lecture 11\lab\FWO-TEST>
```

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



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 get deployments kube-state-metrics -n kube-system

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19042.1348]
(c) Microsoft Corporation. All rights reserved.

C:\Users\paulr\Documents\STUDY\NEU\NEU\Fourth Sem\CSYE7220\Assignments\Project>git clone https://github.com/devopscube/kube-state-metrics-configs.git
Cloning into 'kube-state-metrics-configs'...
remote: Enumerating objects: 10, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 10 (delta 0), reused 7 (delta 0), pack-reused 0
Receiving objects: 100% (10/10), done.

C:\Users\paulr\Documents\STUDY\NEU\NEU\Fourth Sem\CSYE7220\Assignments\Project>kubectl apply -f kube-state-metrics-configs/
clusterrolebinding.rbac.authorization.k8s.io/kube-state-metrics created
clusterrole.rbac.authorization.k8s.io/kube-state-metrics created
deployment.apps/kube-state-metrics created
serviceaccount/kube-state-metrics created
service/kube-state-metrics created

C:\Users\paulr\Documents\STUDY\NEU\NEU\Fourth Sem\CSYE7220\Assignments\Project>kubectl get deployments kube-state-metrics -n kube-system
NAME                 READY   UP-TO-DATE   AVAILABLE   AGE
kube-state-metrics   1/1     1            1           46s
```

Horizontal Autoscaling:

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

The Horizontal Pod Autoscaler (HPA) 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

twtr-combo.yaml | twtr-scaler.yaml

```yaml
apiVersion: autoscaling/v2beta2
kind: HorizontalPodAutoscaler
metadata:
  name: twtr-be
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: twtr-be
  minReplicas: 1
  maxReplicas: 10
  metrics:
    - type: Resource
      resource:
        name: cpu
        target:
          type: Utilization
          averageUtilization: 50
    - type: Resource
      resource:
        name: memory
        target:
          type: Utilization
          averageUtilization: 50
  status:
    observedGeneration: 1
    lastScaleTime: <some-time>
    currentReplicas: 1
    desiredReplicas: 1
    currentMetrics:
    - type: Resource
    resource:
      name: cpu
      current:
        averageUtilization: 0
        averageValue: 0
```

```
C:\Users\paulr\Documents\STUDY\NEU\NEU\Fourth Sem\CSYE7220\Assignments\newrepo\CSYE7220\Project>kubectl describe deploy twtr-be
Name:                   twtr-be
Namespace:              default
CreationTimestamp:      Thu, 16 Dec 2021 02:11:14 -0500
Labels:                 <none>
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=twtr-be
Replicas:               1 desired | 1 updated | 1 total | 1 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:       app=twtr-be
  Annotations:  prometheus.io/path: /status/format/prometheus
                prometheus.io/port: 80
                prometheus.io/scrape: true
  Containers:
   twtr-be:
    Image:      anjalisajeev/twtr-be
    Port:       5000/TCP
    Host Port:  0/TCP
    Limits:
      cpu:     500m
      memory:  1Gi
    Requests:
      cpu:        200m
      memory:     512Mi
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Progressing    True    NewReplicaSetAvailable
  Available      True    MinimumReplicasAvailable
OldReplicaSets:  <none>
NewReplicaSet:   twtr-be-6776758b96 (1/1 replicas created)
Events:          <none>

C:\Users\paulr\Documents\STUDY\NEU\NEU\Fourth Sem\CSYE7220\Assignments\newrepo\CSYE7220\Project>
```

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.

```
C:\Users\paulr\Documents\STUDY\NEU\NEU\Fourth Sem\CSYE7220\Assignments\newrepo\CSYE7220\Project>kubectl describe hpa twtr-be
Name:                                                     twtr-be
Namespace:                                                default
Labels:                                                   <none>
Annotations:                                              <none>
CreationTimestamp:                                        Thu, 16 Dec 2021 02:12:28 -0500
Reference:                                                Deployment/twtr-be
Metrics:                                                  ( current / target )
  resource memory on pods  (as a percentage of request):  14% (75902976) / 50%
  resource cpu on pods  (as a percentage of request):     5% (10m) / 50%
Min replicas:                                             1
Max replicas:                                             10
Deployment pods:                                          1 current / 1 desired
Conditions:
  Type            Status  Reason              Message
  ----            ------  ------              -------
  AbleToScale     True    ReadyForNewScale    recommended size matches current size
  ScalingActive   True    ValidMetricFound    the HPA was able to successfully calculate a replica count from memory resource utilization (percentage of request)
  ScalingLimited  False   DesiredWithinRange  the desired count is within the acceptable range
Events:           <none>

C:\Users\paulr\Documents\STUDY\NEU\NEU\Fourth Sem\CSYE7220\Assignments\newrepo\CSYE7220\Project>
```

Prometheus:

Prometheus 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

```
C:\Users\paulr\Documents\STUDY\NEU\NEU\Fourth Sem\CSYE7220\Assignments\Project\monitoring>kubectl create -f clusterRole.yaml
clusterrole.rbac.authorization.k8s.io/prometheus created
clusterrolebinding.rbac.authorization.k8s.io/prometheus created
```

```
C:\Users\paulr\Documents\STUDY\NEU\NEU\Fourth Sem\CSYE7220\Assignments\Project\monitoring\prometheus>kubectl create -f config-map.yaml
configmap/prometheus-server-conf created
```

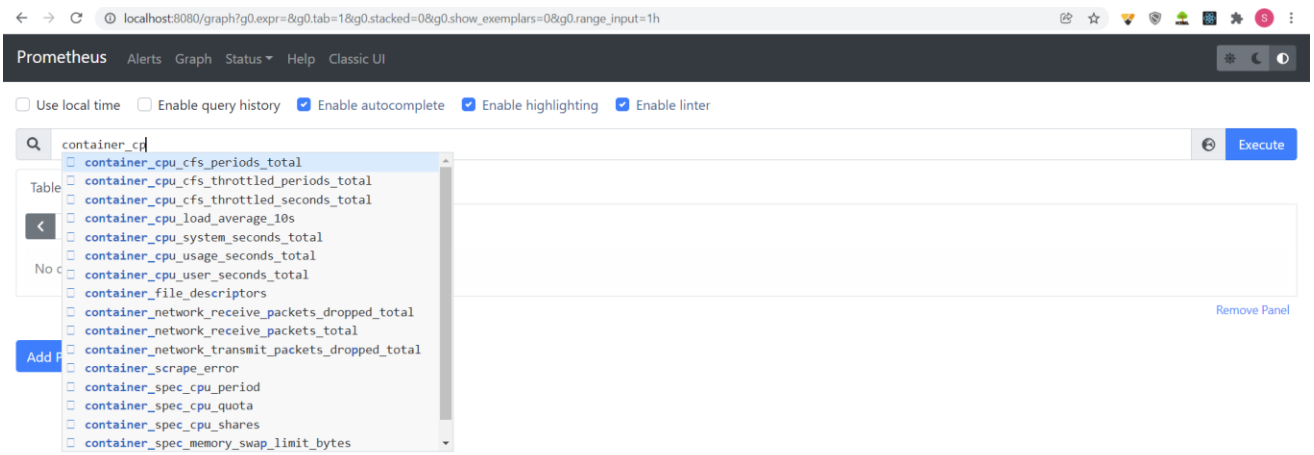
```
C:\Users\paulr\Documents\STUDY\NEU\NEU\Fourth Sem\CSYE7220\Assignments\Project\monitoring\prometheus>kubectl create -f prometheus-deployment.yaml --namespace=monitoring
deployment.apps/prometheus-deployment created

C:\Users\paulr\Documents\STUDY\NEU\NEU\Fourth Sem\CSYE7220\Assignments\Project\monitoring\prometheus>kubectl get deployments --namespace=monitoring
NAME                   READY   UP-TO-DATE   AVAILABLE   AGE
prometheus-deployment  1/1     1            1           17s

C:\Users\paulr\Documents\STUDY\NEU\NEU\Fourth Sem\CSYE7220\Assignments\Project\monitoring\prometheus>kubectl get deployments --all-namespaces
NAMESPACE     NAME                                   READY   UP-TO-DATE   AVAILABLE   AGE
default       twtr-be                                1/1     1            1           3h55m
default       twtr-be-release-prometheus-adapter     1/1     1            1           3h34m
kube-system   coredns                                2/2     2            2           4h6m
kube-system   coredns-autoscaler                     1/1     1            1           4h6m
kube-system   metrics-server                         1/1     1            1           4h6m
kube-system   tunnelfront                            1/1     1            1           4h6m
monitoring    prometheus-deployment                  1/1     1            1           33s
```

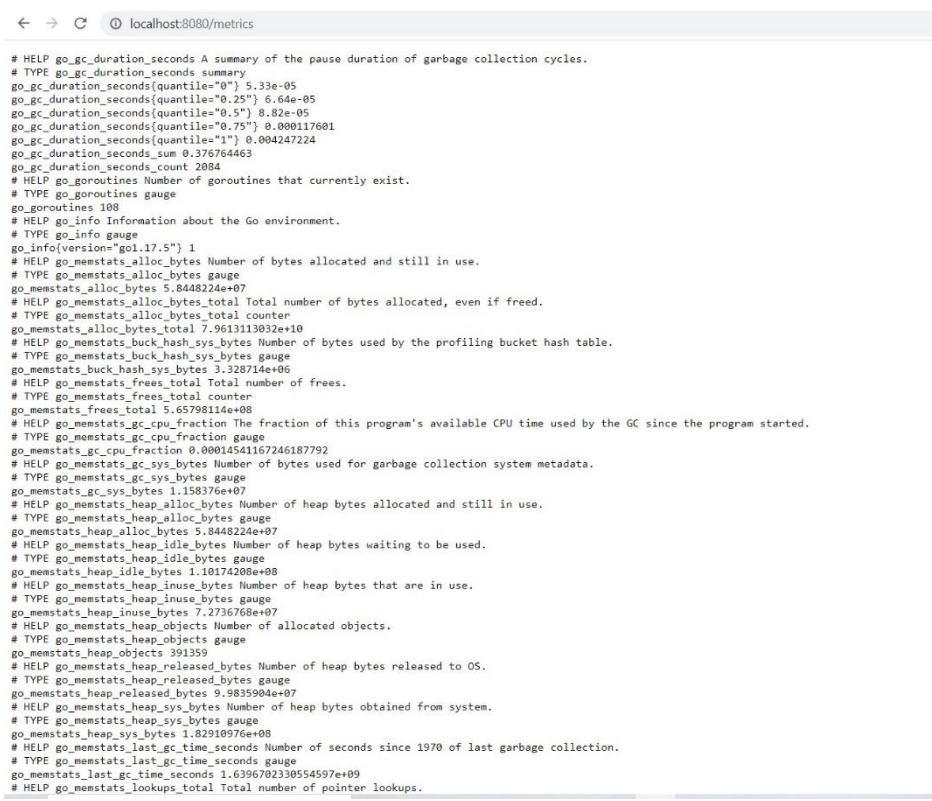
```
C:\Users\paulr\Documents\STUDY\NEU\NEU\Fourth Sem\CSYE7220\Assignments\Project\monitoring\prometheus>kubectl get pods --namespace=monitoring
NAME                                   READY   STATUS    RESTARTS   AGE
prometheus-deployment-87cc8fb88-gp8zz  1/1     Running   0          37m

C:\Users\paulr\Documents\STUDY\NEU\NEU\Fourth Sem\CSYE7220\Assignments\Project\monitoring\prometheus>kubectl port-forward prometheus-deployment-87cc8fb88-gp8zz 8080:9090 -n monitoring
Forwarding from 127.0.0.1:8080 -> 9090
Forwarding from [::1]:8080 -> 9090
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
E1214 19:39:32.630734   33804 portforward.go:233] lost connection to pod
```

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

```
C:\Users\paulr\Documents\STUDY\NEU\NEU\Fourth Sem\CSYE7220\Assignments\Project\monitoring\prometheus>kubectl create -f prometheus-service.yaml --namespace=monitoring
service/prometheus-service created

C:\Users\paulr\Documents\STUDY\NEU\NEU\Fourth Sem\CSYE7220\Assignments\Project\monitoring\prometheus>kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.4.0/aio/deploy/recommended.yaml
namespace/kubernetes-dashboard created
serviceaccount/kubernetes-dashboard created
service/kubernetes-dashboard created
secret/kubernetes-dashboard-certs created
secret/kubernetes-dashboard-csrf created
secret/kubernetes-dashboard-key-holder created
configmap/kubernetes-dashboard-settings created
role.rbac.authorization.k8s.io/kubernetes-dashboard created
clusterrole.rbac.authorization.k8s.io/kubernetes-dashboard created
rolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created
clusterrolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created
deployment.apps/kubernetes-dashboard created
service/dashboard-metrics-scraper created
deployment.apps/dashboard-metrics-scraper created

C:\Users\paulr\Documents\STUDY\NEU\NEU\Fourth Sem\CSYE7220\Assignments\Project\monitoring\prometheus>kubectl proxy
Starting to serve on 127.0.0.1:8001
```

The Prometheus dashboard was accessed using localhost:8080





Grafana:

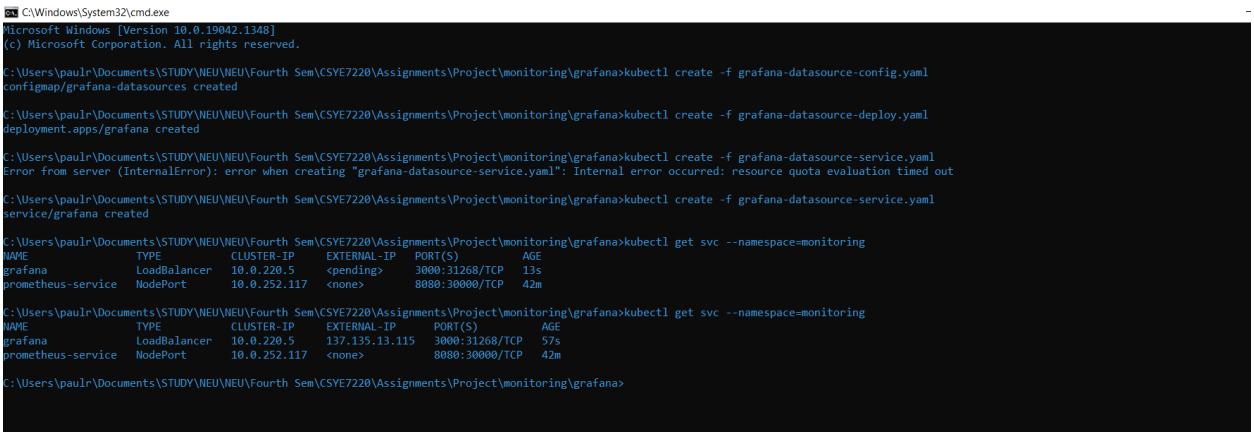
Grafana is a Web dashboard used by many organizations to monitor Kubernetes

Created the below yaml files for grafana configuration:

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

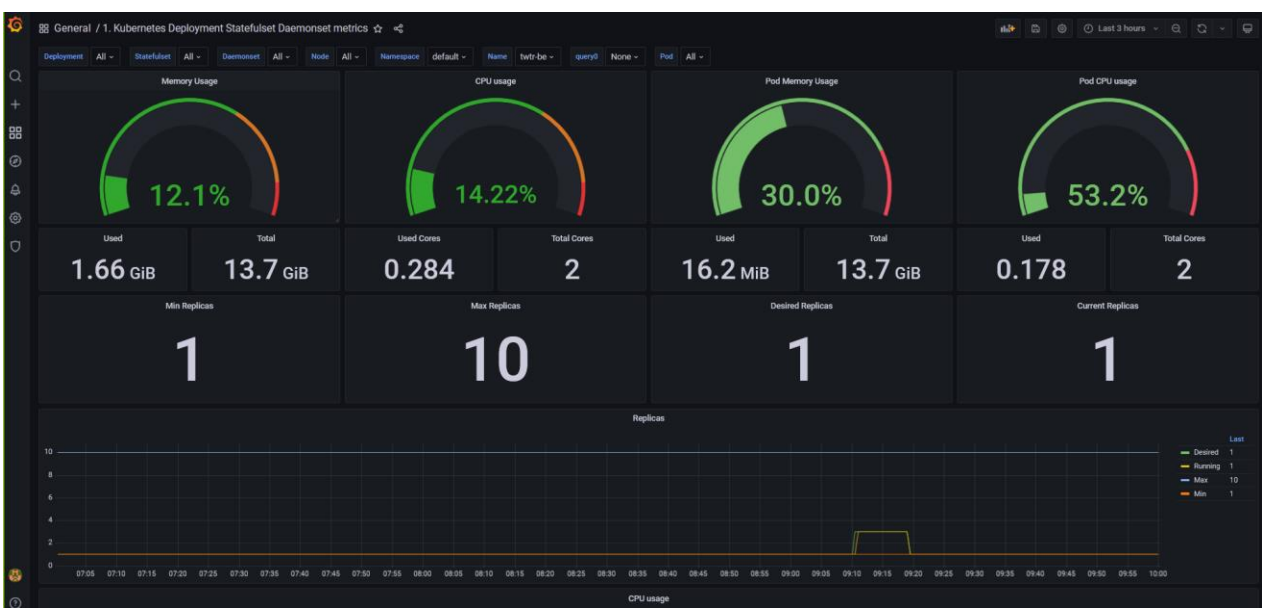
After creating the service file, the below command was run to view the external IP.

- kubectl get svc --namespace=monitoring



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19042.1348]
(c) Microsoft Corporation. All rights reserved.

C:\Users\paulr\Documents\STUDY\NEU\NEU\Fourth Sem\CSYE7220\Assignments\Project\monitoring\grafana>kubectl create -f grafana-datasource-config.yaml
configmap/grafana-datasources created

C:\Users\paulr\Documents\STUDY\NEU\NEU\Fourth Sem\CSYE7220\Assignments\Project\monitoring\grafana>kubectl create -f grafana-datasource-deploy.yaml
deployment.apps/grafana created

C:\Users\paulr\Documents\STUDY\NEU\NEU\Fourth Sem\CSYE7220\Assignments\Project\monitoring\grafana>kubectl create -f grafana-datasource-service.yaml
Error from server (InternalError): error when creating "grafana-datasource-service.yaml": Internal error occurred: resource quota evaluation timed out

C:\Users\paulr\Documents\STUDY\NEU\NEU\Fourth Sem\CSYE7220\Assignments\Project\monitoring\grafana>kubectl create -f grafana-datasource-service.yaml
service/grafana created

C:\Users\paulr\Documents\STUDY\NEU\NEU\Fourth Sem\CSYE7220\Assignments\Project\monitoring\grafana>kubectl get svc --namespace=monitoring
NAME                 TYPE           CLUSTER-IP     EXTERNAL-IP   PORT(S)          AGE
grafana              LoadBalancer   10.0.220.5     <pending>     3000:31268/TCP   13s
prometheus-service   NodePort       10.0.252.117   <none>        8080:30000/TCP   42m

C:\Users\paulr\Documents\STUDY\NEU\NEU\Fourth Sem\CSYE7220\Assignments\Project\monitoring\grafana>kubectl get svc --namespace=monitoring
NAME                 TYPE           CLUSTER-IP     EXTERNAL-IP       PORT(S)          AGE
grafana              LoadBalancer   10.0.220.5     137.135.13.115    3000:31268/TCP   57s
prometheus-service   NodePort       10.0.252.117   <none>            8080:30000/TCP   42m

C:\Users\paulr\Documents\STUDY\NEU\NEU\Fourth Sem\CSYE7220\Assignments\Project\monitoring\grafana>
```

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.



Locust:

Locust was used for load testing.

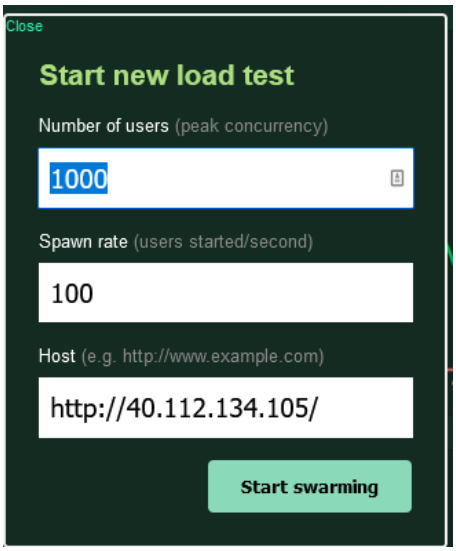
A locust.py file was created, and the below command was run:

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

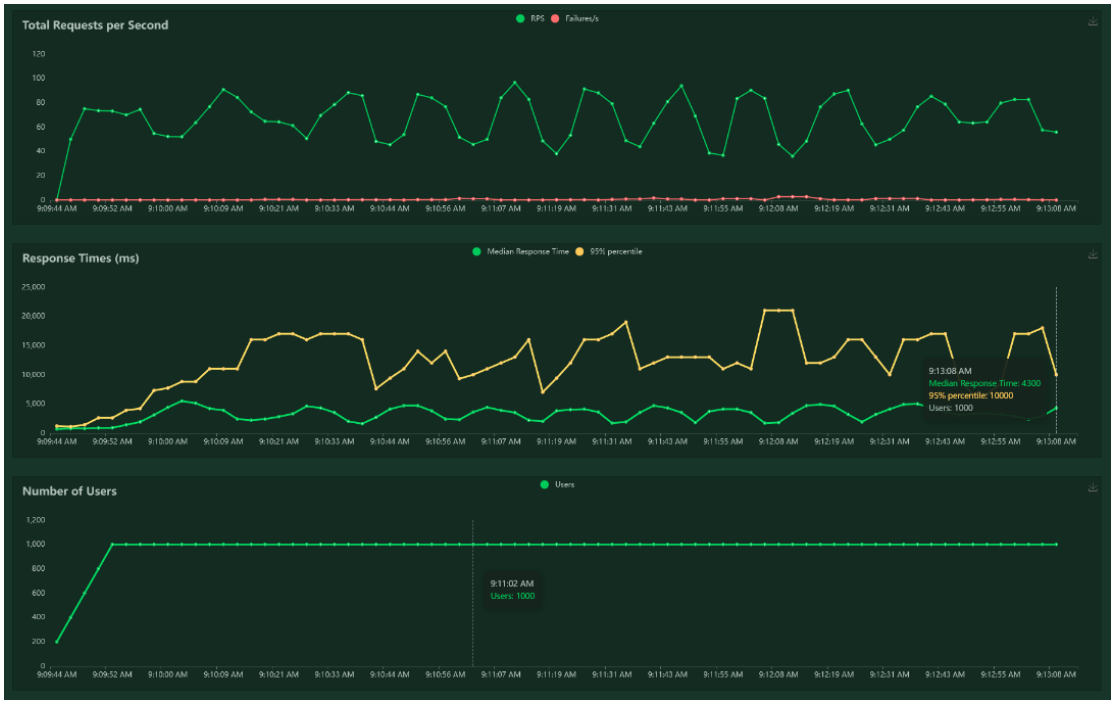
```
C:\Users\anjal\Desktop\Files\Study Materials\Northeastern University\Fall2021\Devops\Assignments\Assignment8>cd locust

C:\Users\anjal\Desktop\Files\Study Materials\Northeastern University\Fall2021\Devops\Assignments\Assignment8\locust>C:\Users\anjal\AppData\Local\Programs\Python\Python310\S
cripts\locust.exe -f locustfile.py --host=http://localhost:5000
[2021-12-16 09:09:04,893] Anjali/INFO/locust.main: Starting web interface at http://0.0.0.0:8089 (accepting connections from all network interfaces)
[2021-12-16 09:09:04,904] Anjali/INFO/locust.main: Starting Locust 2.5.0
[2021-12-16 09:09:42,715] Anjali/INFO/locust.runners: Ramping to 1000 users at a rate of 100.00 per second
[2021-12-16 09:09:51,784] Anjali/INFO/locust.runners: All users spawned: {"MyWebsiteUser": 1000} (1000 total users)
```

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

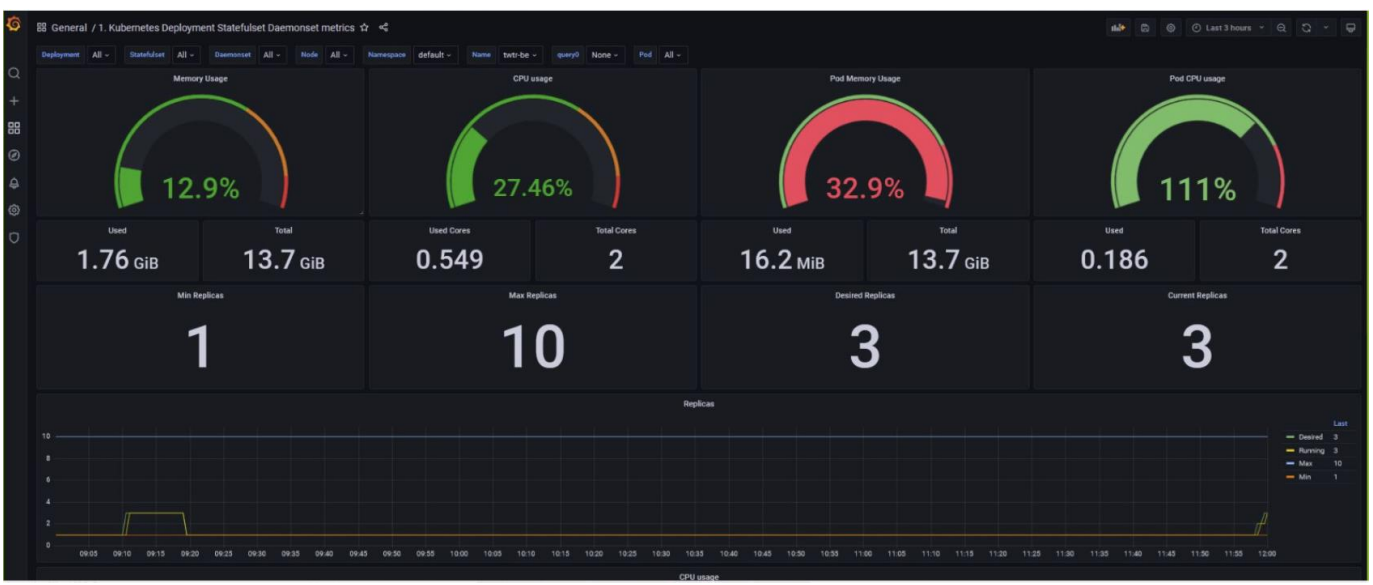
A new test was started to increase the cpu usage:



Once it started swamping, we monitored the requests per second:



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



```
C:\Users\paulr\Documents\STUDY\NEU\NEU\Fourth Sem\CSYE7220\Assignments\Project>kubectl describe hpa twtr-be
Name:                                                     twtr-be
Namespace:                                                default
Labels:                                                   <none>
Annotations:                                              <none>
CreationTimestamp:                                        Tue, 14 Dec 2021 13:25:53 -0500
Reference:                                                Deployment/twtr-be
Metrics:                                                  ( current / target )
  resource cpu on pods  (as a percentage of request):  127% (254m) / 50%
Min replicas:                                             1
Max replicas:                                             10
Deployment pods:                                          3 current / 3 desired
Conditions:
  Type            Status  Reason              Message
  ----            ------  ------              -------
  AbleToScale     True    ReadyForNewScale    recommended size matches current size
  ScalingActive   True    ValidMetricFound    the HPA was able to successfully calculate a replica count from cpu resource utilization (percentage of request)
  ScalingLimited  False   DesiredWithinRange  the desired count is within the acceptable range
Events:
  Type     Reason                        Age                From                       Message
  ----     ------                        ----               ----                       -------
  Warning  FailedGetScale                37m (x3 over 38m)  horizontal-pod-autoscaler  deployments/scale.apps "twtr-be" not found
  Warning  FailedGetResourceMetric       37m (x2 over 37m)  horizontal-pod-autoscaler  failed to get cpu utilization: unable to get metrics for resource cpu: no metrics returned from resource metrics API
  Warning  FailedComputeMetricsReplicas  37m (x2 over 37m)  horizontal-pod-autoscaler  invalid metrics (1 invalid out of 1), first error is: failed to get cpu utilization: unable to get metrics for resource cpu:
 no metrics returned from resource metrics API
  Warning  FailedComputeMetricsReplicas  36m (x4 over 36m)  horizontal-pod-autoscaler  invalid metrics (1 invalid out of 1), first error is: failed to get cpu utilization: did not receive metrics for any ready p
ods
  Warning  FailedGetResourceMetric       36m (x4 over 36m)  horizontal-pod-autoscaler  failed to get cpu utilization: did not receive metrics for any ready pods
  Normal   SuccessfulRescale             15m                horizontal-pod-autoscaler  New size: 2; reason: cpu resource utilization (percentage of request) above target
  Normal   SuccessfulRescale             10m (x3 over 14h)  horizontal-pod-autoscaler  New size: 1; reason: All metrics below target
  Normal   SuccessfulRescale             46s (x2 over 15h)  horizontal-pod-autoscaler  New size: 3; reason: cpu resource utilization (percentage of request) above target
```

ALERT MANAGER

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

Created the below yaml files for alert manager configuration:

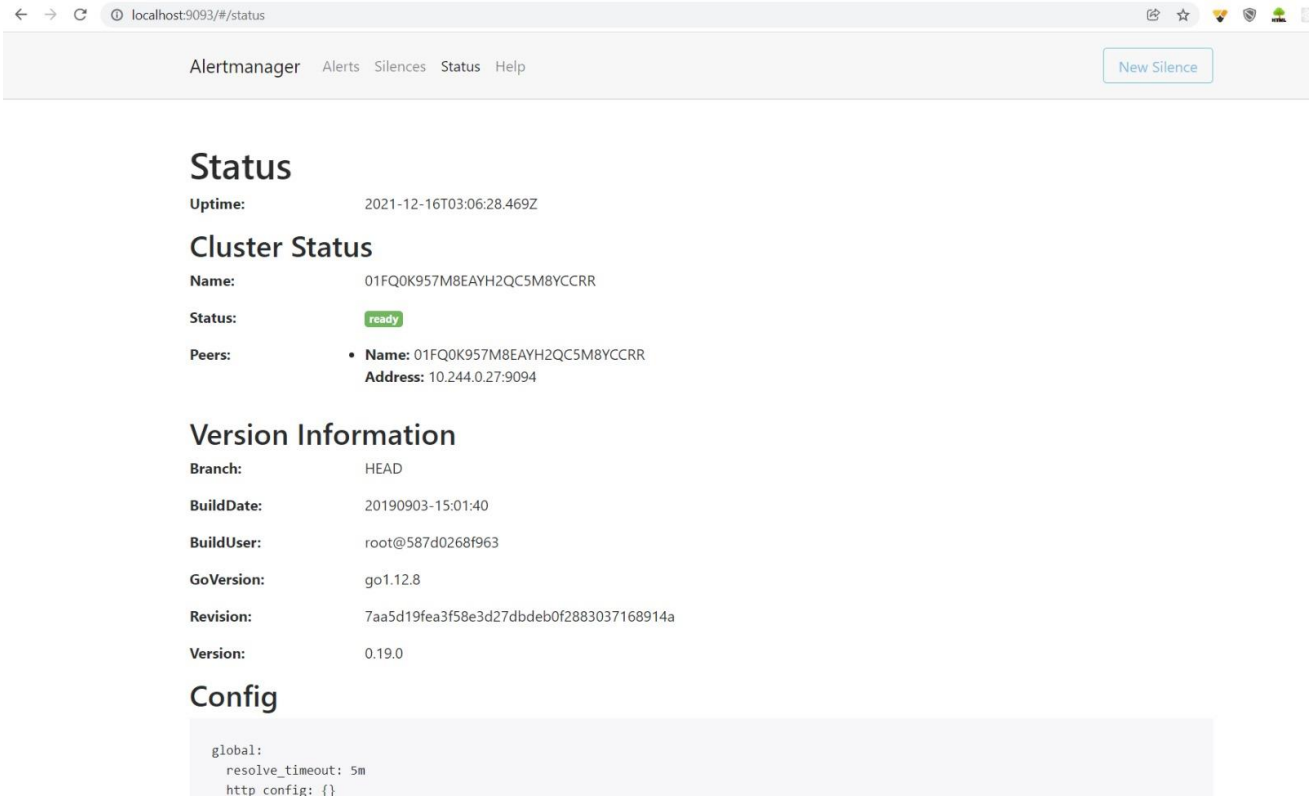
- AlertManagerConfigmap.yaml
- AlertManagerDeployment.yaml
- AlertManagerService.yaml

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

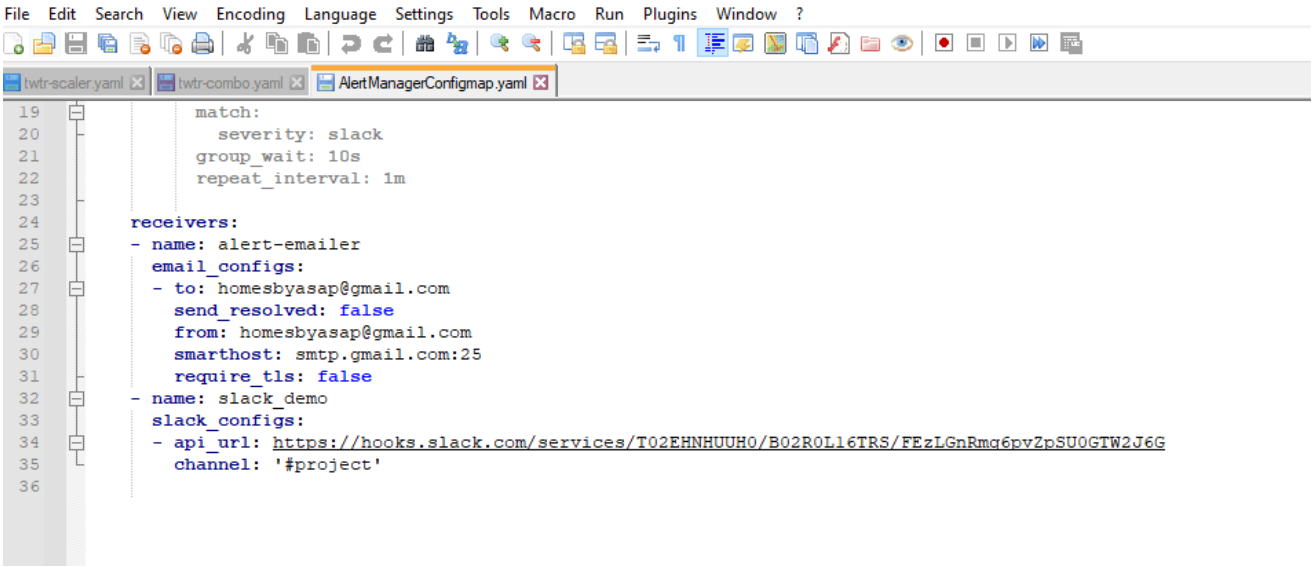
```
C:\Users\paulr\Documents\STUDY\NEU\NEU\Fourth Sem\CSYE7220\Assignments\newrepo\CSYE7220\Project\monitoring\alertmanager>kubectl create -f AlertManagerConfigmap.yaml
configmap/alertmanager-config created

C:\Users\paulr\Documents\STUDY\NEU\NEU\Fourth Sem\CSYE7220\Assignments\newrepo\CSYE7220\Project\monitoring\alertmanager>kubectl create -f AlertTemplateConfigMap.yaml
configmap/alertmanager-templates created

C:\Users\paulr\Documents\STUDY\NEU\NEU\Fourth Sem\CSYE7220\Assignments\newrepo\CSYE7220\Project\monitoring\alertmanager>kubectl create -f AlertManagerDeployment.yaml
deployment.apps/alertmanager created

C:\Users\paulr\Documents\STUDY\NEU\NEU\Fourth Sem\CSYE7220\Assignments\newrepo\CSYE7220\Project\monitoring\alertmanager>kubectl create -f AlertManagerService.yaml
service/alertmanager created

C:\Users\paulr\Documents\STUDY\NEU\NEU\Fourth Sem\CSYE7220\Assignments\newrepo\CSYE7220\Project\monitoring\alertmanager>kubectl get pods -f AlertManagerService.yaml
error: when paths, URLs, or stdin is provided as input, you may not specify resource arguments as well

C:\Users\paulr\Documents\STUDY\NEU\NEU\Fourth Sem\CSYE7220\Assignments\newrepo\CSYE7220\Project\monitoring\alertmanager>kubectl get pods --namespace=monitoring
NAME                                    READY   STATUS    RESTARTS   AGE
alertmanager-896fd948-c4k6j             1/1     Running   0          29s
grafana-6cb5cf456f-9h8v8                1/1     Running   0          28h
prometheus-deployment-87cc8fb88-gp8zz   1/1     Running   0          29h
```

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

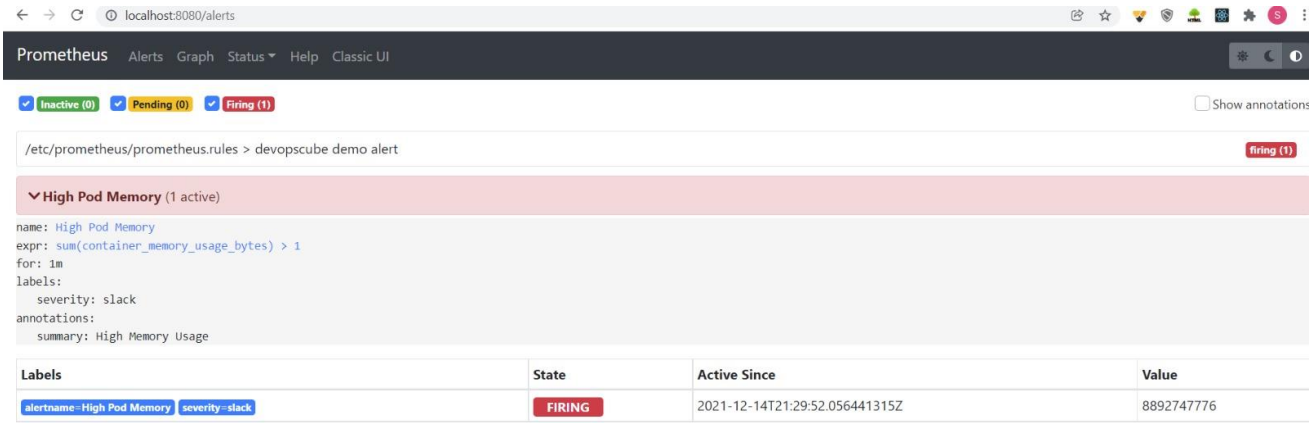
The alertmanager dashboard can be viewed using localhost:9093



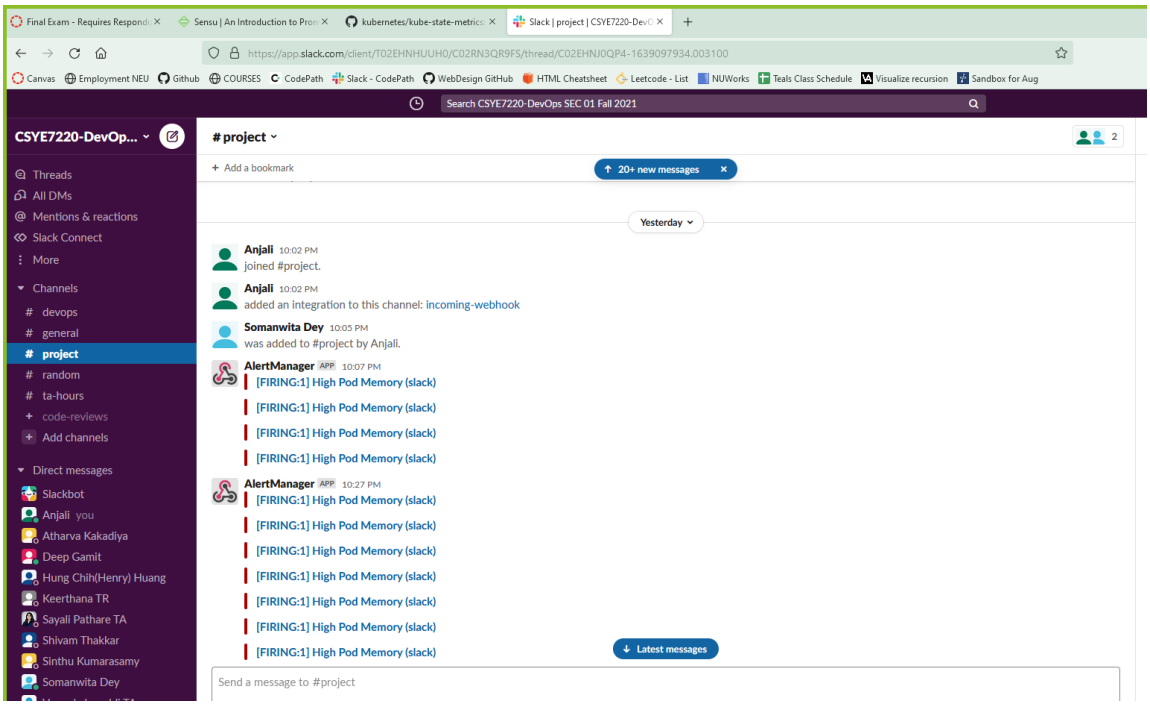
In the Prometheus config-map.yaml we had configured the high pod memory alert if its >1



The alerts are visible in Prometheus alert tab



And also available in slack:



Telemetry:

Number Of users	Spawn rate	Number of Min Replica	Number of Max Replica	Number Of Desired Replica	Number Of Current Replica
1	1	1	10	1	1
1000	100	1	10	3	3

	A	B	C	D	E	F	G	H	I	J	K
1	Type	Name	Request Count	Failure Coun	Median Response Time	Average Response Time	Min Response Time	Max Response Time	Average Content Size	Requests/s	Failures/s
2	GET	//	5507	39	4900	5905.953781	386.4755	32477.7317	433.9052115	64.8160419	0.45902045
3		Aggregated	5507	39	4900	5905.953781	386.4755	32477.7317	433.9052115	64.8160419	0.45902045
4											
5											