

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:

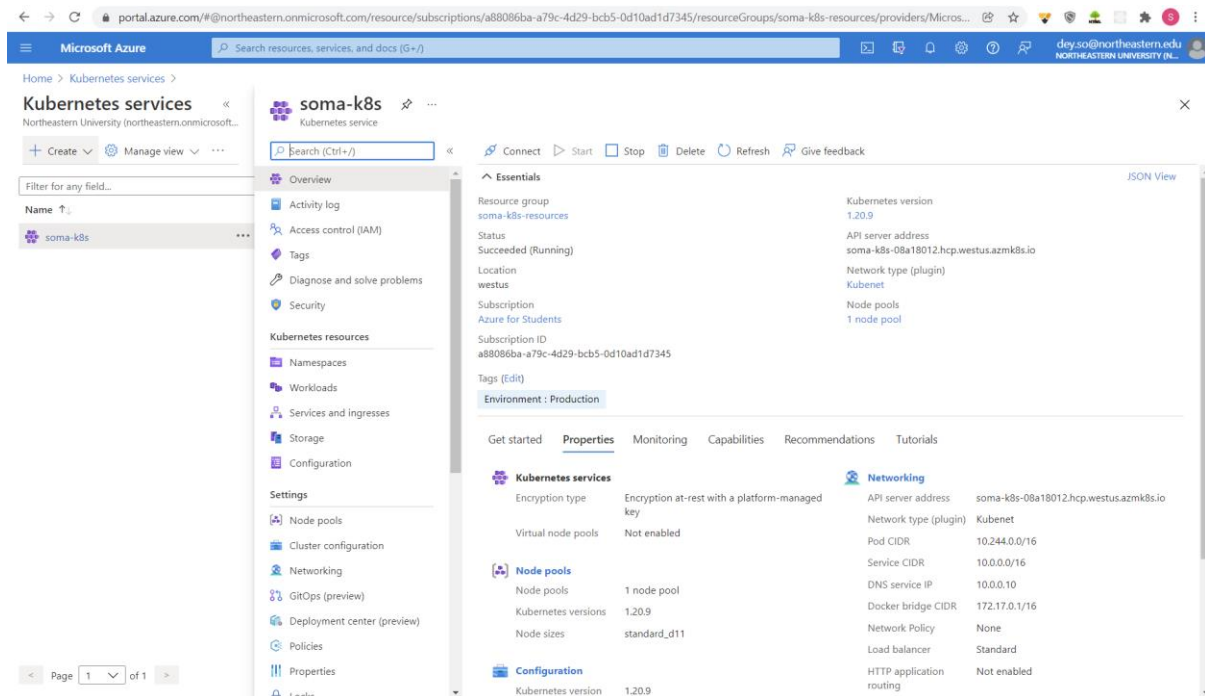
The twtr-be application docker image was created using the below commands:

- `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/anjali/.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`

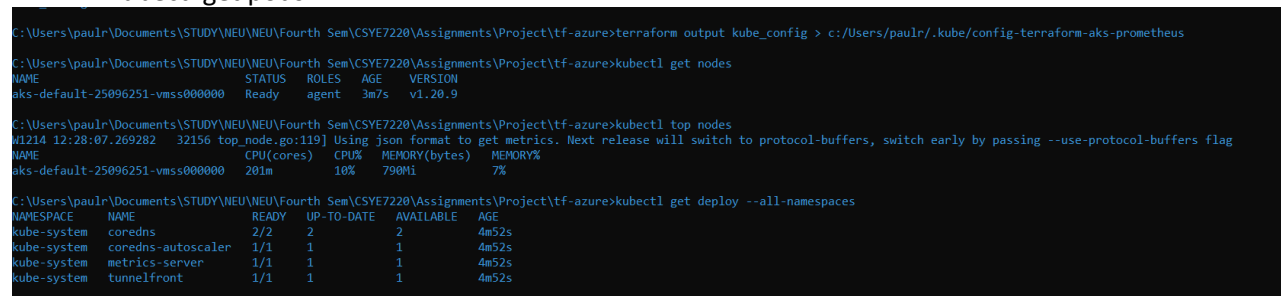
The file contains the limits and request values for memory and cpu resources



```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: twtr-be
5  spec:
6    selector:
7      matchLabels:
8        app: twtr-be
9    replicas: 1
10   template:
11     metadata:
12       annotations:
13         prometheus.io/path: "/status/format/prometheus"
14         prometheus.io/scrape: "true"
15         prometheus.io/port: "80"
16       labels:
17         app: twtr-be
18     spec:
19       containers:
20       - name: twtr-be
21         image: "anjalisajeev/twtr-be"
22         ports:
23         - containerPort: 5000
24         resources:
25           limits:
26             cpu: .5
27             memory: 1Gi
28           requests:
29             memory: 0.5Gi
30             cpu: .2
31   ---
32   apiVersion: v1
33   kind: Service
34   metadata:
35     name: twtr-be
36   labels:
37     app: twtr-be
38   spec:
39     type: LoadBalancer
40     ports:
41     - port: 80
42       targetPort: 5000
43     selector:
44       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    3m/s    v1.20.9

C:\Users\paulr\Documents\STUDY\NEU\NEU\Fourth Sem\CSYE7220\Assignments\Project\tf-azure>kubectl top nodes
M1214 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)    MEMORY(bytes)    MEMORY%
aks-default-25096251-vmss000000    201m         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\FW0-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\FW0-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\FW0-TEST>
```

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

← → ↻ ⚠ Not secure | 40.112.134.105

Welcome to online mongo/twitter testing ground!

Run the following endpoints:
 From collection:
<http://localhost:5000/tweets>
<http://localhost:5000/tweets-week>
<http://localhost:5000/tweets-week-results>
 Create new data:
<http://localhost:5000/mock-tweets>
 Optionally, to purge database: <http://localhost:5000/purge-db>

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-scalar.yaml
1  apiVersion: autoscaling/v2beta2
2  kind: HorizontalPodAutoscaler
3  metadata:
4    name: twtr-be
5  spec:
6    scaleTargetRef:
7      apiVersion: apps/v1
8      kind: Deployment
9      name: twtr-be
10   minReplicas: 1
11   maxReplicas: 10
12   metrics:
13     - type: Resource
14       resource:
15         name: cpu
16         target:
17           type: Utilization
18           averageUtilization: 50
19     - type: Resource
20       resource:
21         name: memory
22         target:
23           type: Utilization
24           averageUtilization: 50
25   status:
26     observedGeneration: 1
27     lastScaleTime: <some-time>
28     currentReplicas: 1
29     desiredReplicas: 1
30     currentMetrics:
31       - type: Resource
32         resource:
33           name: cpu
34           current:
35             averageUtilization: 0
36             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: 16Gi
      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

The screenshot displays the Prometheus web interface in a browser. The address bar shows the URL `localhost:8080/graph?g0.expr=8g0.tab=1&g0.stacked=0&g0.show_exemplars=0&g0.range_input=1h`. The interface includes a navigation bar with links for Alerts, Graph, Status, Help, and Classic UI. Below the navigation bar, there are checkboxes for 'Use local time', 'Enable query history', 'Enable autocomplete', 'Enable highlighting', and 'Enable linter'. The main area is divided into two sections. The top section is the query editor, where the expression `container_cpu` is entered. A dropdown menu is open, showing a list of available metrics starting with `container_cpu`, including `container_cpu_cfs_periods_total`, `container_cpu_cfs_throttled_periods_total`, `container_cpu_load_average_10s`, `container_cpu_system_seconds_total`, `container_cpu_usage_seconds_total`, `container_cpu_user_seconds_total`, `container_file_descriptors`, `container_network_receive_packets_dropped_total`, `container_network_receive_packets_total`, `container_network_transmit_packets_dropped_total`, `container_network_transmit_packets_total`, `container_scrape_error`, `container_spec_cpu_period`, `container_spec_cpu_quota`, `container_spec_cpu_shares`, and `container_spec_memory_swap_limit_bytes`. The bottom section shows the results of the query, displaying a list of metrics and their values. The metrics include `go_gc_duration_seconds`, `go_goroutines`, `go_info`, `go_memstats_alloc_bytes`, `go_memstats_alloc_bytes_total`, `go_memstats_buck_hash_sys_bytes`, `go_memstats_buck_hash_sys_bytes_gauge`, `go_memstats_buck_hash_sys_bytes_gauge`, `go_memstats_frees_total`, `go_memstats_frees_total_counter`, `go_memstats_gc_cpu_fraction`, `go_memstats_gc_cpu_fraction_gauge`, `go_memstats_gc_sys_bytes`, `go_memstats_gc_sys_bytes_gauge`, `go_memstats_heap_alloc_bytes`, `go_memstats_heap_alloc_bytes_gauge`, `go_memstats_heap_alloc_bytes_gauge`, `go_memstats_heap_idle_bytes`, `go_memstats_heap_idle_bytes_gauge`, `go_memstats_heap_inuse_bytes`, `go_memstats_heap_inuse_bytes_gauge`, `go_memstats_heap_objects`, `go_memstats_heap_objects_gauge`, `go_memstats_heap_released_bytes`, `go_memstats_heap_released_bytes_gauge`, `go_memstats_heap_sys_bytes`, `go_memstats_heap_sys_bytes_gauge`, `go_memstats_last_gc_time_seconds`, `go_memstats_last_gc_time_seconds_gauge`, and `go_memstats_lookups_total`.

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.

- `kubect1 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>kubect1 create -f grafana-datasource-config.yaml
configmap/grafana-datasources created

C:\Users\paulr\Documents\STUDY\NEU\NEU\Fourth Sem\CSYE7220\Assignments\Project\monitoring\grafana>kubect1 create -f grafana-datasource-deploy.yaml
deployment.apps/grafana created

C:\Users\paulr\Documents\STUDY\NEU\NEU\Fourth Sem\CSYE7220\Assignments\Project\monitoring\grafana>kubect1 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

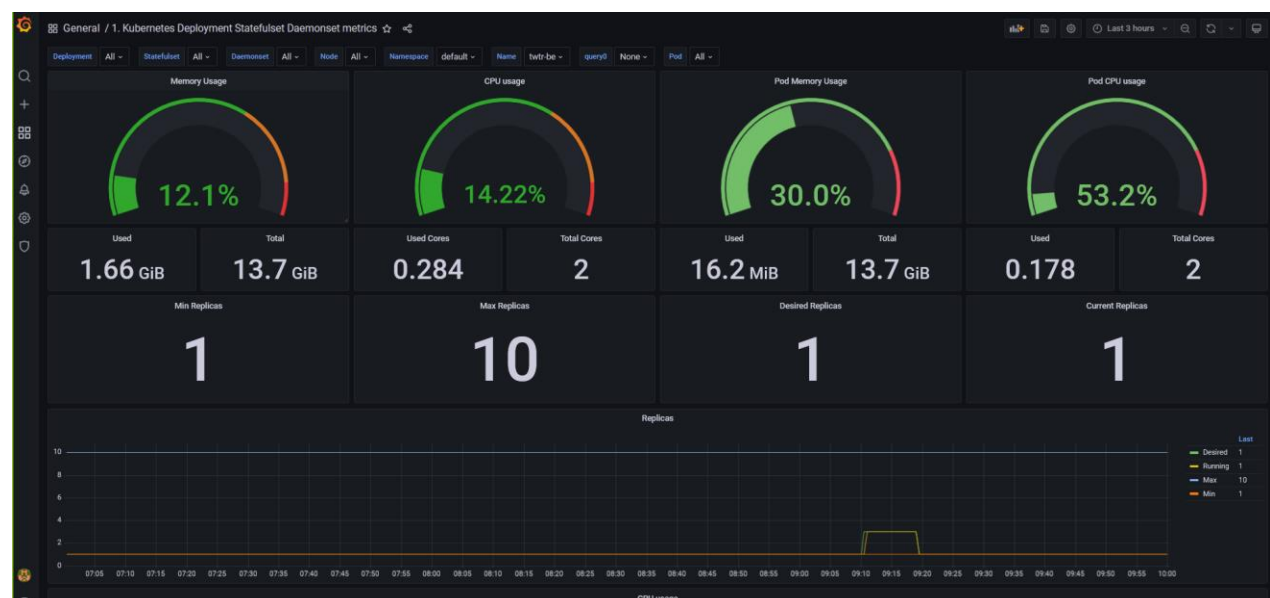
C:\Users\paulr\Documents\STUDY\NEU\NEU\Fourth Sem\CSYE7220\Assignments\Project\monitoring\grafana>kubect1 create -f grafana-datasource-service.yaml
service/grafana created

C:\Users\paulr\Documents\STUDY\NEU\NEU\Fourth Sem\CSYE7220\Assignments\Project\monitoring\grafana>kubect1 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   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>kubect1 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.

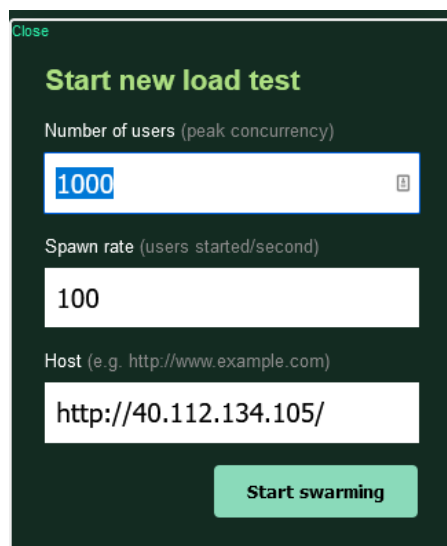
A locust.py file was created, and the below command was run:

- `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\Scripts\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:

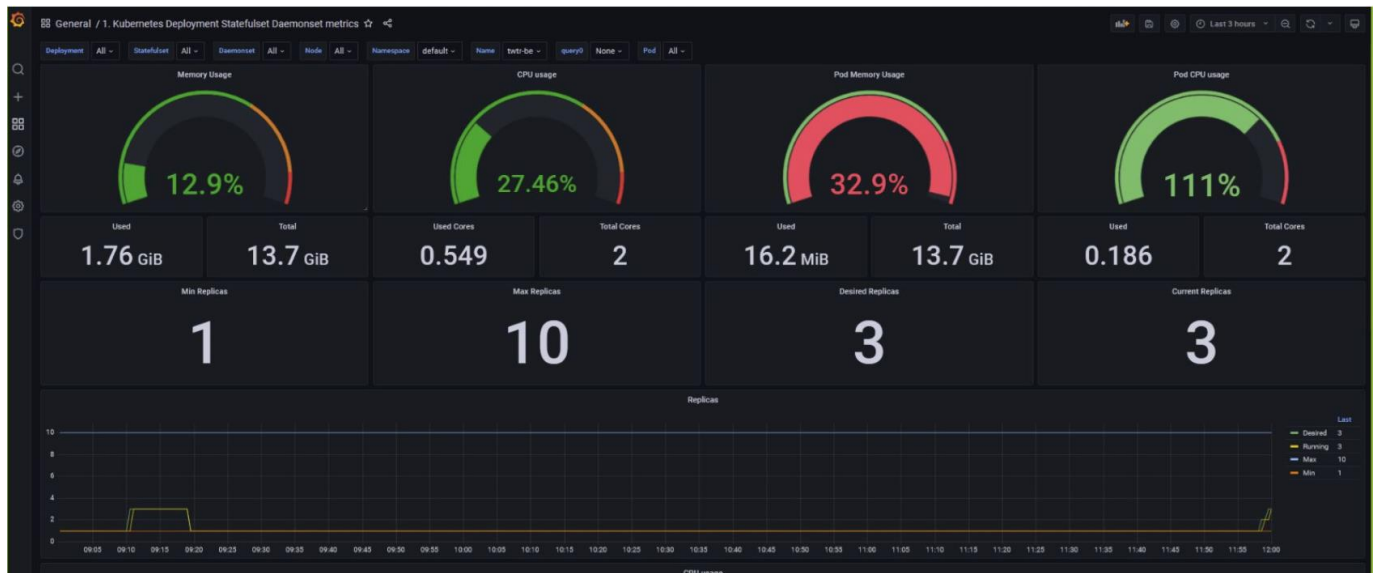


The screenshot shows the Locust web interface with a dark green background. At the top left is a 'Close' button. The main heading is 'Start new load test' in yellow. Below it are three input fields: 'Number of users (peak concurrency)' with the value '1000', 'Spawn rate (users started/second)' with the value '100', and 'Host (e.g. http://www.example.com)' with the value 'http://40.112.134.105/'. At the bottom right is a green button labeled 'Start swarming'.

Once it started swarming, 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 30m) 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 pods
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 path, 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-896fd948-4k6j          1/1 Running 0      29s
grafana-bcb5cf45bf-9h8vd            1/1 Running 0      28s
prometheus-deployment-87cc8f188-gp8zz 1/1 Running 0      29s
```

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

The alertmanager dashboard can be viewed using localhost:9093

← → ↻ localhost:9093/#/status

Alertmanager Alerts Silences Status Help [New Silence](#)

Status

Uptime: 2021-12-16T03:06:28.469Z

Cluster Status

Name: 01FQ0K957M8EAYH2QC5M8YCCRR

Status: ready

Peers:

- Name: 01FQ0K957M8EAYH2QC5M8YCCRR
Address: 10.244.0.27:9094

Version Information

Branch: HEAD

BuildDate: 20190903-15:01:40

BuildUser: root@587d0268f963

GoVersion: go1.12.8

Revision: 7aa5d19fea3f58e3d27dbdeb0f2883037168914a

Version: 0.19.0

Config

```
global:
  resolve_timeout: 5m
  http_config: {}
```

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

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

twtr-scaler.yaml x twtr-combo.yaml x AlertManagerConfigmap.yaml x

```
19 match:
20   severity: slack
21   group_wait: 10s
22   repeat_interval: 1m
23
24 receivers:
25 - name: alert-emailer
26   email_configs:
27   - to: homesbyasap@gmail.com
28     send_resolved: false
29     from: homesbyasap@gmail.com
30     smarthost: smtp.gmail.com:25
31     require_tls: false
32 - name: slack_demo
33   slack_configs:
34   - api_url: https://hooks.slack.com/services/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
35     channel: '#project'
36
```

The screenshot shows the Prometheus web UI at localhost:8080/alerts. The top navigation bar includes 'Prometheus', 'Alerts', 'Graph', 'Status', 'Help', and 'Classic UI'. Below the navigation bar, there are status indicators: 'Inactive (0)', 'Pending (0)', and 'Firing (1)'. A search bar contains the path '/etc/prometheus/prometheus.rules > devopscube demo alert'. The main content area displays an active alert titled 'High Pod Memory (1 active)'. The alert details include the name 'High Pod Memory', the expression 'sum(container_memory_usage_bytes) > 1', the duration 'for: 1m', and labels 'severity: slack'. The annotations section shows 'summary: High Memory Usage'. At the bottom, a table provides a summary of the alert:

Labels	State	Active Since	Value
alertname=High Pod Memory severity=slack	FIRING	2021-12-14T21:29:52.056441315Z	8892747776

The screenshot displays a Slack workspace interface. At the top, a navigation bar shows the channel name "#project" and a search bar. Below this, a sidebar on the left lists various channels and direct messages, with "#project" highlighted. The main content area shows a list of messages in the "#project" channel. The messages include a welcome message from Anjali, a message from Somanwita Dey, and a series of messages from the "AlertManager" bot indicating high pod memory usage. The interface is clean and professional, with a dark theme.

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

[illegible]