Faculty of Engineering & Applied Science

SOFE4790U – Distributed Systems

Lab 2 – CRN 44425

Due Date: 10/09/2022

First Name	Last Name	Student ID
Abdul	Bhutta	100785884

Changes: RED

1.0 Part 1 – Gateway Routing Pattern

Read the following document (Gateway Routing pattern - Azure Architecture Center | Microsoft Learn). Focus on the problem being solved by the pattern, how is it solved? and the requirements needed for the solution.

The problem is that when a client is accessing multiple services at once, numerous service instances, or both, the client must be updated when there is a change in the services (Add or remove). An example is when the user is accessing an e-commerce website, they will have various services such as searching for an item, adding an item to the cart, and checkout. If a new service is added, such as order history, it must also be updated on the client side. In the table, you can see the different scenarios that may occur,

Scenario	Description
Multiple disparate services	Each service requires an API call that the client will have to communicate with, while each endpoint will be different. Each time a change is made in the code, the client and the service must be updated.
Multiple instances of the same service	The same services can be run multiple times for load balancing, or the service can be offered in different regions.
Multiple versions of the same service	A new version of the service can be deployed with the existing version.

The solution

One of the solutions to the problem is to place a gateway between the client and services where it will route the request to the desired service. The client only requires one endpoint and will communicate with the gateway. Since there is only one endpoint, the client side will communicate with a single endpoint. The table below shows the solutions to the different scenarios and how to integrate the gateway.

Multiple disparate services

When the gateway routing protocol is applied, the client will send a request to the gateway, which routes it to the correct instance. If a change is made in service or removed, the client does not need to change; only the routing will be updated.

Multiple instances of the same service

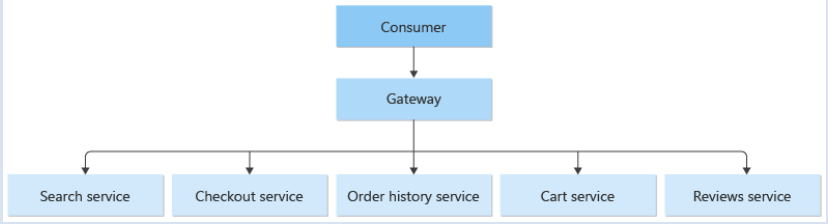
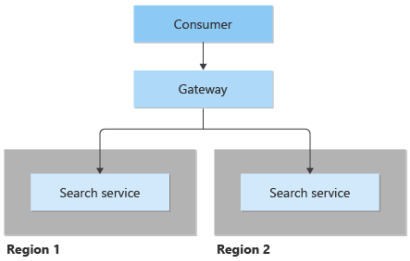
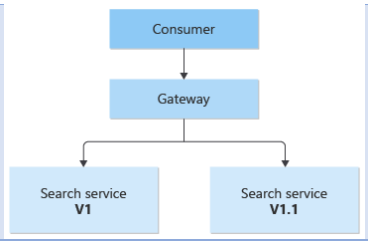
Elasticity in cloud computing is extremely useful as services are never consistent throughout the year and fluctuate between various times and regions. The gateway routing protocol allows an increase in the number of services to meet the demand or a decrease in the number of services to save money while keeping it hidden from the user. The gateway will route the client to the best endpoint for their region.

Multiple versions of the same service

Multiple versions can be deployed, which helps the developer release them in parallel or incremental updates, while the gateway routing allows us to choose which service can be presented to the client. If the new deployments have issues, they can be changed back to the old version without affecting the clients.

Requirements:

- Elasticity
- Availability
- Reliability
- Latency
- Greater than 1 or 2 services

Scenario	Solution Diagrams
Multiple disparate services	
Multiple instances of the same service	
Multiple versions of the same service	

2.0 Part 2

You will be guided through the steps to deploy a request-splitting ambassador that will split 10% of the incoming HTTP requests to an experimental server.

i. Create web server and experiment server while making them accessible.

```
bhutta_abdul@cloudshell:~ (sofe4790u-lab2)$ kubectl delete deployment web-deployment
deployment.apps "web-deployment" deleted
bhutta_abdul@cloudshell:~ (sofe4790u-lab2)$ kubectl delete deployment experiment-deployment
deployment.apps "experiment-deployment" deleted
bhutta_abdul@cloudshell:~ (sofe4790u-lab2)$ kubectl delete services experiment-deployment
service "experiment-deployment" deleted
bhutta_abdul@cloudshell:~ (sofe4790u-lab2)$ kubectl create -f web-deployment.yaml
deployment.apps/web-deployment created
bhutta_abdul@cloudshell:~ (sofe4790u-lab2)$ kubectl expose deployment web-deployment --port=80 --type=ClusterIP --name web- deployment
Error from server (Invalid): Service "web-" is invalid: metadata.name: Invalid value: "web-": a DNS-1035 label must consist of lower case alphanumeric characters or '-', start with an alphabetic ch
aracter, and end with an alphanumeric character (e.g. 'my-name',  or 'abc-123', regex used for validation is '[a-z]([-a-z0-9]*[a-z0-9])?')
Error from server (NotFound): deployments.apps "deployment" not found
bhutta_abdul@cloudshell:~ (sofe4790u-lab2)$ kubectl expose deployment web-deployment --port=80 --type=ClusterIP --name web-deployment
service/web-deployment exposed
bhutta_abdul@cloudshell:~ (sofe4790u-lab2)$ kubectl create -f experiment-deployment.yaml
deployment.apps/experiment-deployment created
bhutta_abdul@cloudshell:~ (sofe4790u-lab2)$  kubectl expose deployment experiment-deployment --port=80 --type=ClusterIP --name experiment-deployment
service/experiment-deployment exposed
bhutta_abdul@cloudshell:~ (sofe4790u-lab2)$
```

ii. Generate configMap

```
bhutta_abdul@cloudshell:~$ kubectl create configmap ambassador-config --from-file=conf.d
configmap/ambassador-config created
```

iii. Deploy the ambassador service

```
bhutta_abdul@cloudshell:~$ kubectl create -f ambassador-deployment.yaml
deployment.apps/ambassador-deployment created
bhutta_abdul@cloudshell:~$
```

iv. Check if the pods, deployments, and services are running

```
bhutta_abdul@cloudshell:~$ kubectl get pods
NAME                                    READY   STATUS    RESTARTS   AGE
ambassador-deployment-66db4f7766-br4vq  1/1     Running   0          113s
ambassador-deployment-66db4f7766-nmhzk  1/1     Running   0          113s
experiment-deployment-7b47cbd668-jt4b4  1/1     Running   0          10m
experiment-deployment-7b47cbd668-t6sgs  1/1     Running   0          10m
mongodb-deployment-5c6cb86f45-8lqqq     1/1     Running   0          4d21h
mysql-deployment-5496fdc956-994jm       1/1     Running   0          4d21h
web-deployment-6fdbb5c6bb-5zjvc         1/1     Running   0          33m
web-deployment-6fdbb5c6bb-pwp7f         1/1     Running   0          33m
bhutta_abdul@cloudshell:~$ kubectl get deployments
NAME                   READY   UP-TO-DATE   AVAILABLE   AGE
ambassador-deployment  2/2     2            2           2m6s
experiment-deployment  2/2     2            2           10m
mongodb-deployment     1/1     1            1           14d
mysql-deployment       1/1     1            1           21d
web-deployment         2/2     2            2           33m
bhutta_abdul@cloudshell:~$ kubectl get services
NAME                   TYPE           CLUSTER-IP    EXTERNAL-IP      PORT(S)          AGE
ambassador-deployment  LoadBalancer   10.84.6.158   34.171.252.47    80:32216/TCP     91s
experiment-deployment  ClusterIP      10.84.3.33    <none>           80/TCP           11m
kubernetes             ClusterIP      10.84.0.1     <none>           443/TCP          23d
mongodb-service        LoadBalancer   10.84.8.48    35.226.185.35    3306:30647/TCP   14d
mysql-service          LoadBalancer   10.84.0.54    34.171.222.236   3306:30136/TCP   21d
web-deployment         ClusterIP      10.84.9.181   <none>           80/TCP           15m
bhutta_abdul@cloudshell:~$
```

Finally, check that all pods, deployments, and services are running with no issues. What commands should you use? What's the external IP address associated with the **ambassador-deployment** service?

1. Check pods are running -> **kubectl get pods**
2. Check deployments are running -> **kubectl get deployments**
3. Check services are running -> **kubectl get services**
4. External IP address to the ambassador deployment -> **34.171.252.47**

v. View the output

vi. Run the script to call the server many times and view the output.txt file.

```
bhutta_abdul@cloudshell:~$ for _ in {1..20}; do curl http://34.171.252.47 -s > output.txt; done
bhutta_abdul@cloudshell:~$
bhutta_abdul@cloudshell:~$
```

output.txt

```
<html>
<head>
  <title>Welcome to Azure Container Instances!</title>
</head>
<style>
h1 {
    color: darkblue;
    font-family:arial, sans-serif;
    font-weight: lighter;
}
</style>

<body>

<div align="center">
<h1>Welcome to Azure Container Instances!</h1>

<svg id="Layer_1" data-name="Layer 1" xmlns="http://www.w3.org/2000/svg" viewBox="0 0 49.8 49.9" width="250px" height="250px">
  <title>ContainerInstances_rgb_UI</title>
  <path d="M41.9,11.368A11.929,11.929,0,0,0,20.3,5.061a9.444,9.444,0,0,0-14.932,9.8A8.969,8.969,0,0,0,0,9.064,32H39.442A10.463,10.463,0,0,0,41.9,11.368Z" transform="t
    <path d="M41.9,11.368A11.929,11.929,0,0,0,20.3,5.061a9.444,9.444,0,0,0-14.932,9.8A8.969,8.969,0,0,0,0,9.064,32H39.442A10.463,10.463,0,0,0,41.9,11.368Z" transform=
    <path d="M13,22a1,1,0,0,0-1,1V49a1,1,0,0,0,1,1H37a1,1,0,0,0,1-1V23a1,1,0,0,0-1-1Z" transform="translate(-0.1 -0.1)" fill="#672a7a"/>
    <path d="M26.95,16" transform="translate(-0.1 -0.1)" fill="none"/>
    <path d="M34.95,20" transform="translate(-0.1 -0.1)" fill="none"/>
    <polygon points="22.9 21.9 22.9 14.9 19.9 14.9 24.9 7.9 29.9 14.9 26.9 14.9 26.9 21.9 22.9 21.9" fill="#fff"/>
    <path d="M26.95,16" transform="translate(-0.1 -0.1)" fill="#814a98"/>
    <path d="M33,25H15V47H35V25ZM21,45H17V27h4Zm6,0H23V27h4Zm6,0H29V27h4Z" transform="translate(-0.1 -0.1)" fill="#b92025" opacity="0.3" style="isolation:isolate"/>
    <path d="M33,25H15V47H35V25ZM21,45H17V27h4Zm6,0H23V27h4Zm6,0H29V27h4Z" transform="translate(-0.1 -0.1)" fill="#fff" style="isolation:isolate"/>
</svg>
</div>
```

vii. Read logs to view how many times each server was called.

```
bhutta_abdul@cloudshell:~$ kubectl logs -l run=web-deployment
::ffff:10.80.3.6 - - [07/Oct/2022:17:59:14 +0000] "GET / HTTP/1.0" 200 1663 "-" "curl/7.74.0"
::ffff:10.80.3.6 - - [07/Oct/2022:17:59:14 +0000] "GET / HTTP/1.0" 200 1663 "-" "curl/7.74.0"
::ffff:10.80.3.6 - - [07/Oct/2022:17:59:14 +0000] "GET / HTTP/1.0" 200 1663 "-" "curl/7.74.0"
::ffff:10.80.2.10 - - [07/Oct/2022:17:59:14 +0000] "GET / HTTP/1.0" 200 1663 "-" "curl/7.74.0"
::ffff:10.80.3.6 - - [07/Oct/2022:17:59:14 +0000] "GET / HTTP/1.0" 200 1663 "-" "curl/7.74.0"
::ffff:10.80.3.6 - - [07/Oct/2022:17:59:14 +0000] "GET / HTTP/1.0" 200 1663 "-" "curl/7.74.0"
::ffff:10.80.2.10 - - [07/Oct/2022:17:59:15 +0000] "GET / HTTP/1.0" 200 1663 "-" "curl/7.74.0"
::ffff:10.80.3.6 - - [07/Oct/2022:17:59:15 +0000] "GET / HTTP/1.0" 200 1663 "-" "curl/7.74.0"
::ffff:10.80.3.6 - - [07/Oct/2022:17:59:15 +0000] "GET / HTTP/1.0" 200 1663 "-" "curl/7.74.0"
::ffff:10.80.2.10 - - [07/Oct/2022:17:59:15 +0000] "GET / HTTP/1.0" 200 1663 "-" "curl/7.74.0"
::ffff:10.80.3.6 - - [07/Oct/2022:17:49:59 +0000] "POST / HTTP/1.0" 404 140 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit
::ffff:10.80.3.6 - - [07/Oct/2022:17:51:46 +0000] "GET /favicon.ico HTTP/1.0" 404 150 "http://34.171.252.47/" "Mozilla/5.0 (
ri/605.1.15"
::ffff:10.80.3.6 - - [07/Oct/2022:17:52:26 +0000] "GET / HTTP/1.0" 200 1663 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; r
::ffff:10.80.2.10 - - [07/Oct/2022:17:59:14 +0000] "GET / HTTP/1.0" 200 1663 "-" "curl/7.74.0"
::ffff:10.80.2.10 - - [07/Oct/2022:17:59:14 +0000] "GET / HTTP/1.0" 200 1663 "-" "curl/7.74.0"
::ffff:10.80.2.10 - - [07/Oct/2022:17:59:14 +0000] "GET / HTTP/1.0" 200 1663 "-" "curl/7.74.0"
::ffff:10.80.3.6 - - [07/Oct/2022:17:59:15 +0000] "GET / HTTP/1.0" 200 1663 "-" "curl/7.74.0"
::ffff:10.80.3.6 - - [07/Oct/2022:17:59:15 +0000] "GET / HTTP/1.0" 200 1663 "-" "curl/7.74.0"
::ffff:10.80.3.6 - - [07/Oct/2022:17:59:15 +0000] "GET / HTTP/1.0" 200 1663 "-" "curl/7.74.0"
::ffff:10.80.3.6 - - [07/Oct/2022:17:59:15 +0000] "GET / HTTP/1.0" 200 1663 "-" "curl/7.74.0"
bhutta_abdul@cloudshell:~$
bhutta_abdul@cloudshell:~$
 bhutta_abdul@cloudshell:~$ kubectl logs -l run=experiment-deployment
 listening on port 80
 ::ffff:10.80.2.10 - - [07/Oct/2022:17:59:15 +0000] "GET / HTTP/1.0" 200 1663 "-" "curl/7.74.0"
 listening on port 80
 ::ffff:10.80.3.6 - - [07/Oct/2022:17:59:14 +0000] "GET / HTTP/1.0" 200 1663 "-" "curl/7.74.0"
 ::ffff:10.80.3.6 - - [07/Oct/2022:17:59:15 +0000] "GET / HTTP/1.0" 200 1663 "-" "curl/7.74.0"
```

3.0 Part 3

i. Create the deployments.

```
bhutta_abdul@cloudshell:~$ kubectl create -f loadbalancer-deployment.yaml
deployment.apps/loadbalancer-deployment created
bhutta_abdul@cloudshell:~$
```

ii. Check the status of the created pods.

```
bhutta_abdul@cloudshell:~$ kubectl get pods --output=wide
NAME                                     READY   STATUS    RESTARTS   AGE     IP           NODE                                          NOMINATED NODE   READINESS GATES
ambassador-deployment-66db4f7766-br4vq   1/1     Running   0          46m     10.80.2.10   gke-openfaas-default-pool-98fbf41c-xgnr       <none>           <none>
ambassador-deployment-66db4f7766-nmhzk   1/1     Running   0          46m     10.80.3.6    gke-openfaas-default-pool-98fbf41c-07wg       <none>           <none>
experiment-deployment-7b47cbd668-jt4b4   1/1     Running   0          55m     10.80.2.9    gke-openfaas-default-pool-98fbf41c-xgnr       <none>           <none>
experiment-deployment-7b47cbd668-t6sgs   1/1     Running   0          55m     10.80.1.5    gke-openfaas-default-pool-98fbf41c-l7x6       <none>           <none>
loadbalancer-deployment-6676f9ccf6-59ct7 1/1     Running   0          46s     10.80.1.6    gke-openfaas-default-pool-98fbf41c-l7x6       <none>           <none>
loadbalancer-deployment-6676f9ccf6-7q7k4 1/1     Running   0          46s     10.80.2.11   gke-openfaas-default-pool-98fbf41c-xgnr       <none>           <none>
loadbalancer-deployment-6676f9ccf6-gt712 1/1     Running   0          46s     10.80.3.7    gke-openfaas-default-pool-98fbf41c-07wg       <none>           <none>
mongodb-deployment-5c6cb86f45-81qqq      1/1     Running   0          4d22h   10.80.3.3    gke-openfaas-default-pool-98fbf41c-07wg       <none>           <none>
mysql-deployment-5496fdc956-994jm        1/1     Running   0          4d22h   10.80.2.2    gke-openfaas-default-pool-98fbf41c-xgnr       <none>           <none>
web-deployment-6fdbb5c6bb-5zjvc          1/1     Running   0          78m     10.80.2.8    gke-openfaas-default-pool-98fbf41c-xgnr       <none>           <none>
web-deployment-6fdbb5c6bb-pwp7f          1/1     Running   0          78m     10.80.1.4    gke-openfaas-default-pool-98fbf41c-l7x6       <none>           <none>
bhutta_abdul@cloudshell:~$
```

iii. Associating a load balancer.

```
bhutta_abdul@cloudshell:~$ kubectl expose deployment loadbalancer-deployment --port=8080 --type=LoadBalancer
service/loadbalancer-deployment exposed
bhutta_abdul@cloudshell:~$
```

iv. Get external IP of the load balancer.

```
bhutta_abdul@cloudshell:~$ kubectl get services --watch
NAME                        TYPE           CLUSTER-IP     EXTERNAL-IP      PORT(S)          AGE
ambassador-deployment       LoadBalancer   10.84.6.158    34.171.252.47    80:32216/TCP     50m
experiment-deployment       ClusterIP      10.84.3.33     <none>           80/TCP           59m
kubernetes                  ClusterIP      10.84.0.1      <none>           443/TCP          24d
loadbalancer-deployment     LoadBalancer   10.84.7.236    35.232.122.111   8080:30458/TCP   58s
mongodb-service             LoadBalancer   10.84.8.48     35.226.185.35    3306:30647/TCP   14d
mysql-service               LoadBalancer   10.84.0.54     34.171.222.236   3306:30136/TCP   21d
web-deployment              ClusterIP      10.84.9.181    <none>           80/TCP           64m
```

v. Running the dictionary application through the replicated load-balancing service.

```
bhutta_abdul@cloudshell:~$ curl http://35.232.122.111:8080/dog
A quadruped of the genus Canis, esp. the domestic dog (C.familiaris).bhutta_abdul@cloudshell:~$ curl http://35.232.122.111:8080/storey
See Story.bhutta_abdul@cloudshell:~$
bhutta_abdul@cloudshell:~$ curl http://35.232.122.111:8080/story
A set of rooms on the same floor or level; a floor, or thespace between two floors. Also, a horizontal division of a building'sexterior cons
idered architecturally, which need not correspondexactly with the stories within. [Written also storey.]bhutta_abdul@cloudshell:~$
```

```
See Story.bhutta_abdul@cloudshell:~$ curl http://35.232.122.111:8080/story
A set of rooms on the same floor or level; a floor, or thespace between two floors. Also, a horizontal division of a building'sexterior considered architecturally, which need not correspondexactly with the stories wit
hin. [Written also storey.]bhutta_abdul@cloudshell:~$
bhutta_abdul@cloudshell:~$ curl http://35.232.122.111:8080/dog
A quadruped of the genus Canis, esp. the domestic dog (C.familiaris).bhutta_abdul@cloudshell:~$
```

4.0 Discussion

Summarize the problem, the solution, and the requirements for the pattern given in part 1. Which of these requirements can be achieved by the procedures shown in parts 2 and 3?

A gateway routing pattern can route multiple services or instances to a single endpoint by placing a gateway in front of the services. This solves the issue with multiple disparate services, multiple instances of the same service, and multiple versions of the same service. The solution allows the request from the client to reach the gateway, which routes to the appropriate instance or services. In part 2, the ambassador deployment is used as a gateway to split the traffic between the web server and the experiment server. The client will communicate with a single endpoint while the gateway will manage the routes. This is an example of multiple versions of the same service. A new pod should not receive traffic until it is entirely up, which is why the readiness probe is used in part 3 to wait until the application is fully up and running before sending the traffic every 5 seconds. The readiness probe acts as a gateway before routing the traffic.

5.0 Design

Autoscaling is another pattern that can be implemented by GKE. Your task is to configure a Yaml file that autoscaling the deployment of a given Pod. Why autoscaling is usually used? How autoscaling is implemented? How autoscaling is different from load balancing and request splitter?

Autoscaling is helpful as it can help the user manage the resource load, such as using more resources during peak hours and less during downtime. Also, it gives the user more availability in case of a hardware or application failure. Autoscaling in google cloud is implemented using Kubernetes, where you apply the autoscaling option to the deployment. You must specify the resource utilization target and the minimum and maximum pods. Autoscaling allows the user to scale the resources at a lower cost. In contrast, load balancing is used to distribute the server workload for efficiency and performance. Autoscaling and load balancer work together to provide efficiency and performance. A request splitter will always split the traffic based on the user's specification to another server. In part 2, we sent 90% of the traffic to the main server while sending 10% of the traffic to the experimental server. This allows the user to conduct testing on the experimental server between versions before deploying them. In this lab, to demonstrate autoscaling, we will use a deployment which runs a container with the image “autoscaling-example” and use that as a service. The YAML file deployed was found through the Kubernetes documentation website, which runs a container of the image “hpa-example” and uses it as a service. The PHP-Apache server will listen to incoming requests. The horizontal pod scaler was integrated to manage the CPU utilization to 50 percent while allowing for autoscaling to increase or decrease the number of pods (1 – 10). Multiple requests were sent to the server through a different terminal infinitely while monitoring the horizontal scaling done by Kubernetes. Once the requests were stopped, the auto-scaling took the replicas back to 1 as CPU utilization was 0. An example of autoscaling is shown below.

- It allows the service to have greater availability, reliability and elasticity, even when there are hardware failures. With auto scaling a more optimized application environment is achieved through a dynamic hardware environment

i. Create deployment and service of the YAML file.

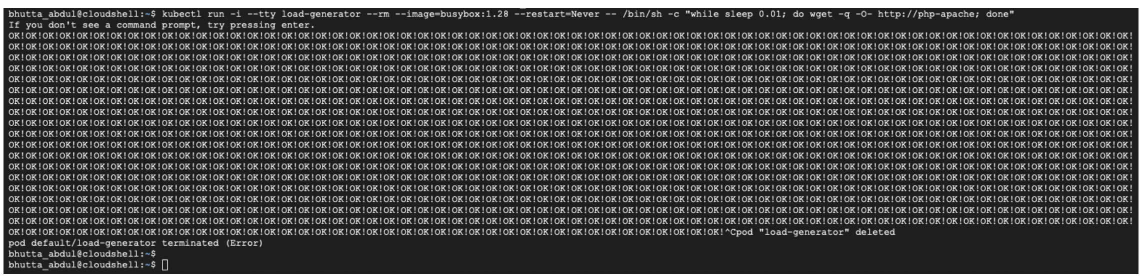
```
bhutta_abdul@cloudshell:~ (sofe4790u-lab2)$ kubectl apply -f https://k8s.io/examples/application/php-apache.yaml
deployment.apps/php-apache created
service/php-apache created
bhutta_abdul@cloudshell:~ (sofe4790u-lab2)$
```

ii. Create an auto scaler to maintain CPU utilization at 40% and balance the load between 1 and 10 pods.

```
bhutta_abdul@cloudshell:~ (sofe4790u-lab2)$ kubectl autoscale deployment php-apache --cpu-percent=60 --min=1 --max=10
horizontalpodautoscaler.autoscaling/php-apache autoscaled
bhutta_abdul@cloudshell:~ (sofe4790u-lab2)$
```

```
bhutta_abdul@cloudshell:~ (sofe4790u-lab2)$ kubectl get hpa
NAME         REFERENCE               TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
php-apache   Deployment/php-apache   0%/50%    1         10        1          34s
```

iii. Send unlimited requests to the PHP server, which will increase the workload on the CPU.



iv. Check-in real-time the usage of the CPU and the autoscaling.

```
bhutta_abdul@cloudshell:~ (sofe4790u-lab2)$ kubectl get hpa php-apache --watch
NAME         REFERENCE               TARGETS    MINPODS   MAXPODS   REPLICAS   AGE
php-apache   Deployment/php-apache   250%/50%   1         10        4          112s
php-apache   Deployment/php-apache   109%/50%   1         10        5          2m2s
php-apache   Deployment/php-apache   58%/50%    1         10        5          2m32s
php-apache   Deployment/php-apache   61%/50%    1         10        5          3m2s
php-apache   Deployment/php-apache   61%/50%    1         10        7          3m17s
php-apache   Deployment/php-apache   35%/50%    1         10        7          3m32s
php-apache   Deployment/php-apache   41%/50%    1         10        7          4m2s
^Cbhutta_abdul@cloudshell:~ (sofe4790u-lab2)kubectl get deployment php-apachehe
NAME         READY   UP-TO-DATE   AVAILABLE   AGE
php-apache   7/7     7            7           4m28s
bhutta_abdul@cloudshell:~ (sofe4790u-lab2)$
```

v. Kubernetes scaling down will bring the replica's pod back down to 1 as the CPU utilization returns to 0 because the infinite requests were stopped.

```
bhutta_abdul@cloudshell:~ (sofe4790u-lab2)$ kubectl get hpa php-apache --watch

NAME         REFERENCE               TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
php-apache   Deployment/php-apache   0%/50%    1         10        7          10m
php-apache   Deployment/php-apache   0%/50%    1         10        1          10m
^Cbhutta_abdul@cloudshell:~ (sofe4790u-lab2)$ kubectl get deployment php-apache
NAME         READY   UP-TO-DATE   AVAILABLE   AGE
php-apache   1/1     1            1           10m
bhutta_abdul@cloudshell:~ (sofe4790u-lab2)$
```

Videos links

Part 2 and 3 Final Results: https://drive.google.com/file/d/1tNuS1CO7dWAhjza9_rbD_l4S-DPiWI4i/view?usp=sharing

Auto Scaling - https://drive.google.com/file/d/1IFn7dYaQQ--kEs-A1CsprZnJFPcT0YMK/view?usp=sharing