OntarioTech
UNIVERSITY

Faculty of Engineering and Applied Science

SOFE 4790U Distributed Systems

CRN 44425

Lab #2

Alexander Campbell

100703650

Part 1:

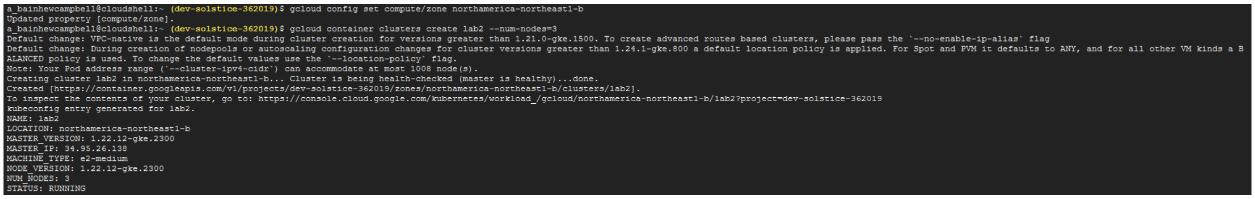
The Gateway Routing pattern is a pattern where a single endpoint can be used to serve multiple services or multiple requests. This is done by placing a gateway in front of your application so that the connected client is only aware of the single endpoint occupied by the gateway. This is useful for three main reasons: exposing several versions of one service, having multiple instances of the same service, or when there's multiple services on one end point.

We can explore the usefulness by taking an e-commerce store as an example. Say a product owner would like to do some AB Testing for a new user interface, you may change the user interface on one version of the service to test if it increases sales by exposing the two versions of the service on one end point for your traffic to use. The gateway would then automatically distribute clients to each version of the website. Having multiple instances is also very useful for e-commerce as there are many regional differences you may want to consider such as high traffic times changing depending on the time zone. In this case, we would want to balance the load to ensure that servers located in high traffic areas have the appropriate amount of availability, while low traffic areas can be decreased and save on computational costs. The gateway would automatically redirect clients towards the endpoint that best suits their region. E-commerce stores also have many services such as search or checkout. Using this pattern, we can expose each end point so that the client is aware of each of the services available to them through the gateway.

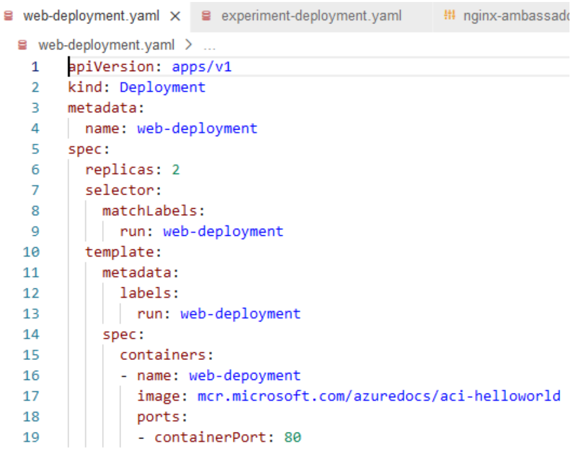
The problems discussed in this can be boiled down to the fact that clients need to be easily redirected to different services without them having to have knowledge about the backend of the application. To utilize this pattern properly one must ensure that the gateway is correctly designed as otherwise it can lead to failure or a bottleneck of the entire system.

Part 2:

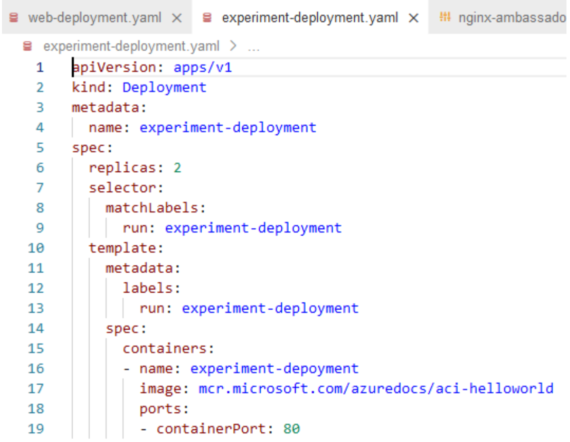
First the pods were initialized, then the web, experiment, and ambassador deployment yaml files were created, deployed, and exposed successfully



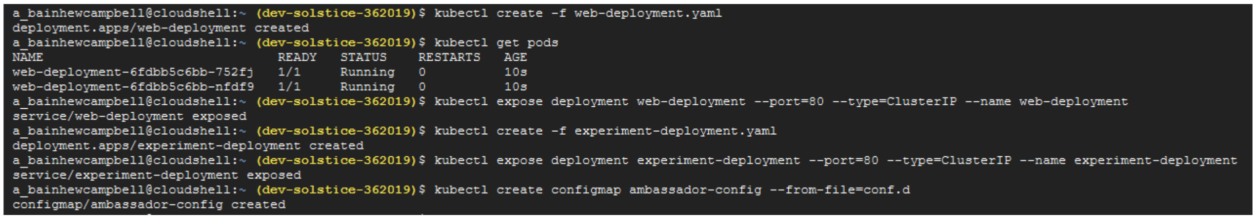
```
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ gcloud config set compute/zone northamerica-northeast1-b
Updated property [compute/zone].
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ gcloud container clusters create lab2 --num-nodes=3
Default change: VPC-native is the default mode during cluster creation for versions greater than 1.21.0-gke.1500. To create advanced routes based clusters, please pass the `--no-enable-ip-alias` flag
Default change: During creation of nodepools or autoscaling configuration changes for cluster versions greater than 1.24.1-gke.800 a default location policy is applied. For Spot and PVM it defaults to ANY, and for all other VM kinds a B
ALANCED policy is used. To change the default values use the `--location-policy` flag.
Note: Your Pod address range (`--cluster-ipv4-cidr`) can accommodate at most 1008 node(s).
Creating cluster lab2 in northamerica-northeast1-b... Cluster is being health-checked (master is healthy)...done.
Created [https://container.googleapis.com/v1/projects/dev-solstice-362019/zones/northamerica-northeast1-b/clusters/lab2].
To inspect the contents of your cluster, go to: https://console.cloud.google.com/kubernetes/workload_/gcloud/northamerica-northeast1-b/lab2?project=dev-solstice-362019
kubeconfig entry generated for lab2.
NAME: lab2
LOCATION: northamerica-northeast1-b
MASTER_VERSION: 1.22.12-gke.2300
MASTER_IP: 34.95.26.138
MACHINE_TYPE: e2-medium
NODE_VERSION: 1.22.12-gke.2300
NUM_NODES: 3
STATUS: RUNNING
```



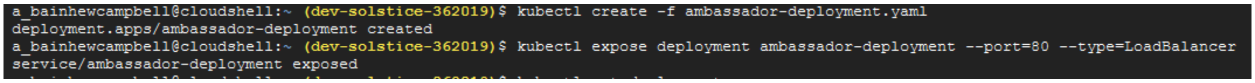
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      run: web-deployment
  template:
    metadata:
      labels:
        run: web-deployment
    spec:
      containers:
      - name: web-depoyment
        image: mcr.microsoft.com/azuredocs/aci-helloworld
        ports:
        - containerPort: 80
```



```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: experiment-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      run: experiment-deployment
  template:
    metadata:
      labels:
        run: experiment-deployment
    spec:
      containers:
      - name: experiment-depoyment
        image: mcr.microsoft.com/azuredocs/aci-helloworld
        ports:
        - containerPort: 80
```

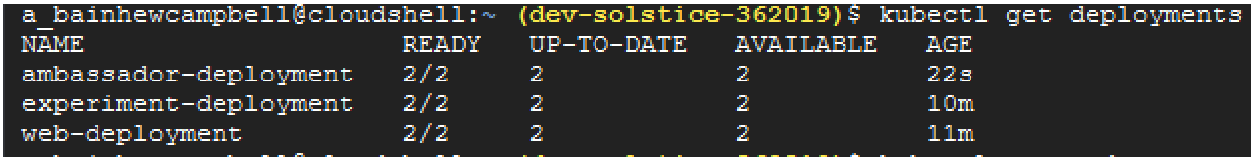


```
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl create -f web-deployment.yaml
deployment.apps/web-deployment created
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl get pods
NAME                              READY   STATUS    RESTARTS   AGE
web-deployment-6fdbb5c6bb-752fj   1/1     Running   0          10s
web-deployment-6fdbb5c6bb-nfdf9   1/1     Running   0          10s
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl expose deployment web-deployment --port=80 --type=ClusterIP --name web-deployment
service/web-deployment exposed
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl create -f experiment-deployment.yaml
deployment.apps/experiment-deployment created
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl expose deployment experiment-deployment --port=80 --type=ClusterIP --name experiment-deployment
service/experiment-deployment exposed
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl create configmap ambassador-config --from-file=conf.d
configmap/ambassador-config created
```

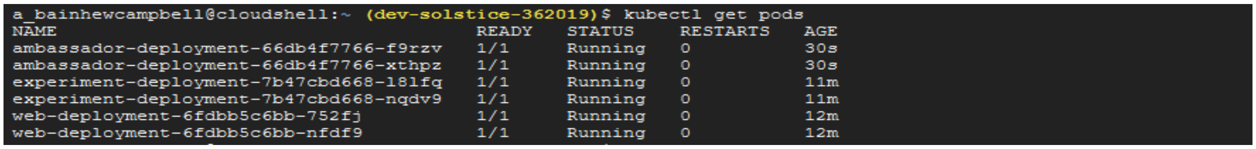


```
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl create -f ambassador-deployment.yaml
deployment.apps/ambassador-deployment created
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl expose deployment ambassador-deployment --port=80 --type=LoadBalancer
service/ambassador-deployment exposed
```

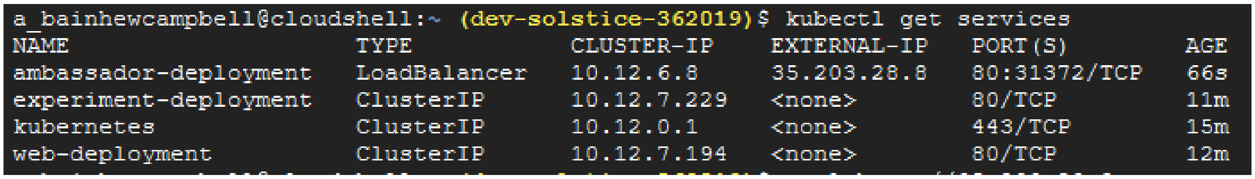
Then the pods, deployments, and services were checked to ensure that they were running without any errors



```
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl get deployments
NAME                    READY   UP-TO-DATE   AVAILABLE   AGE
ambassador-deployment   2/2     2            2           22s
experiment-deployment   2/2     2            2           10m
web-deployment          2/2     2            2           11m
```

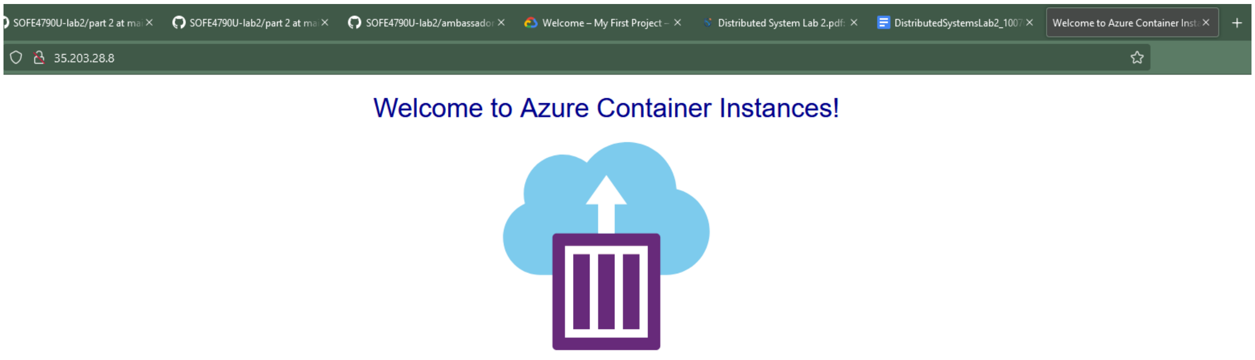


```
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl get pods
NAME                                     READY   STATUS    RESTARTS   AGE
ambassador-deployment-66db4f7766-f9rzv   1/1     Running   0          30s
ambassador-deployment-66db4f7766-xthpz   1/1     Running   0          30s
experiment-deployment-7b47cbd668-181fq   1/1     Running   0          11m
experiment-deployment-7b47cbd668-nqdv9   1/1     Running   0          11m
web-deployment-6fdbb5c6bb-752fj          1/1     Running   0          12m
web-deployment-6fdbb5c6bb-nfdf9          1/1     Running   0          12m
```

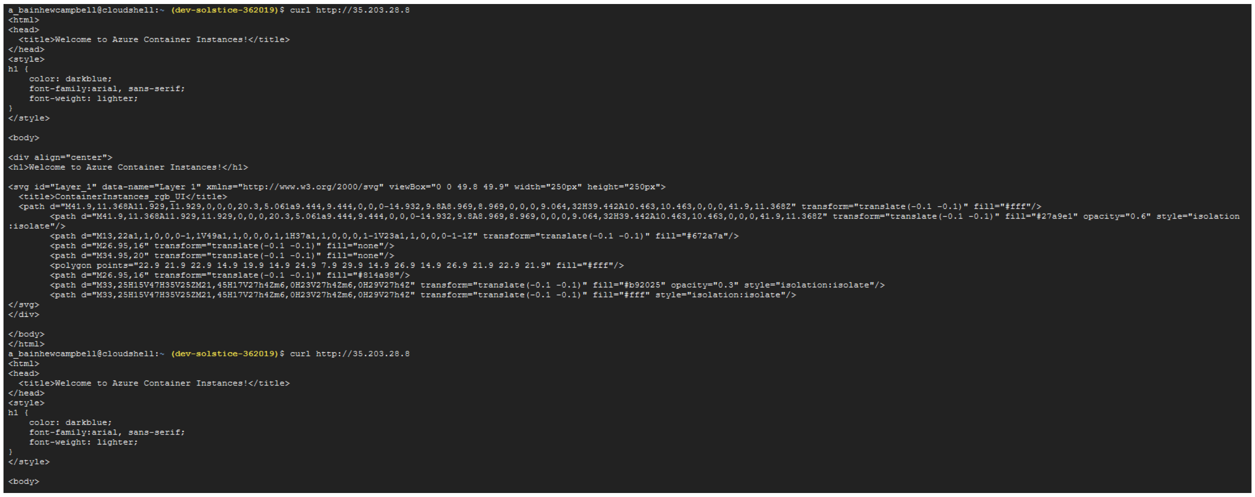


```
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl get services
NAME                    TYPE           CLUSTER-IP    EXTERNAL-IP   PORT(S)        AGE
ambassador-deployment   LoadBalancer   10.12.6.8     35.203.28.8   80:31372/TCP   66s
experiment-deployment   ClusterIP      10.12.7.229   <none>        80/TCP         11m
kubernetes              ClusterIP      10.12.0.1     <none>        443/TCP        15m
web-deployment          ClusterIP      10.12.7.194   <none>        80/TCP         12m
```

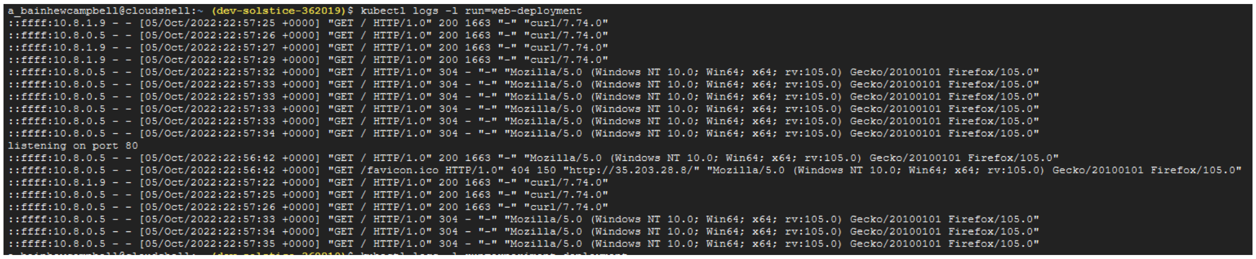
After the pods, deployments, and services were confirmed to be running correctly I opened up the ambassador-deployment IP address in my browser to find the website running and then ran the curl command many times and viewed the logs.


SOFE4790U-lab2/part 2 at ma
SOFE4790U-lab2/part 2 at ma
SOFE4790U-lab2/ambassador
Welcome – My First Project –
Distributed System Lab 2.pdf
DistributedSystemsLab2_1007
Welcome to Azure Container Inst
35.203.28.8

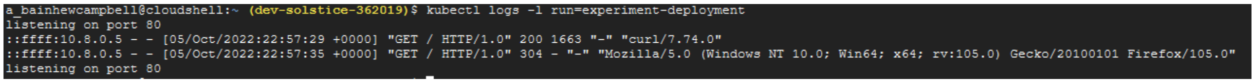
Welcome to Azure Container Instances!



```
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ curl http://35.203.28.8
<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="translate(-0.1 -0.1)" fill="#fff"/>
        <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="translate(-0.1 -0.1)" fill="#27a9e1" opacity="0.6" style="isolation
:isolate"/>
        <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>

</body>
</html>
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ curl http://35.203.28.8
<html>
<head>
  <title>Welcome to Azure Container Instances!</title>
</head>
<style>
h1 {
    color: darkblue;
    font-family:arial, sans-serif;
    font-weight: lighter;
}
</style>

<body>
```



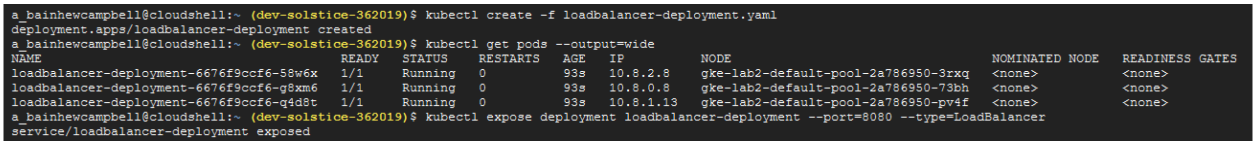
```
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl logs -l run=web-deployment
::ffff:10.8.1.9 - - [05/Oct/2022:22:57:25 +0000] "GET / HTTP/1.0" 200 1663 "-" "curl/7.74.0"
::ffff:10.8.0.5 - - [05/Oct/2022:22:57:26 +0000] "GET / HTTP/1.0" 200 1663 "-" "curl/7.74.0"
::ffff:10.8.1.9 - - [05/Oct/2022:22:57:27 +0000] "GET / HTTP/1.0" 200 1663 "-" "curl/7.74.0"
::ffff:10.8.1.9 - - [05/Oct/2022:22:57:29 +0000] "GET / HTTP/1.0" 200 1663 "-" "curl/7.74.0"
::ffff:10.8.0.5 - - [05/Oct/2022:22:57:32 +0000] "GET / HTTP/1.0" 304 - "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.0) Gecko/20100101 Firefox/105.0"
::ffff:10.8.0.5 - - [05/Oct/2022:22:57:33 +0000] "GET / HTTP/1.0" 304 - "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.0) Gecko/20100101 Firefox/105.0"
::ffff:10.8.0.5 - - [05/Oct/2022:22:57:33 +0000] "GET / HTTP/1.0" 304 - "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.0) Gecko/20100101 Firefox/105.0"
::ffff:10.8.0.5 - - [05/Oct/2022:22:57:33 +0000] "GET / HTTP/1.0" 304 - "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.0) Gecko/20100101 Firefox/105.0"
::ffff:10.8.0.5 - - [05/Oct/2022:22:57:33 +0000] "GET / HTTP/1.0" 304 - "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.0) Gecko/20100101 Firefox/105.0"
::ffff:10.8.0.5 - - [05/Oct/2022:22:57:34 +0000] "GET / HTTP/1.0" 304 - "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.0) Gecko/20100101 Firefox/105.0"
listening on port 80
::ffff:10.8.0.5 - - [05/Oct/2022:22:56:42 +0000] "GET / HTTP/1.0" 200 1663 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.0) Gecko/20100101 Firefox/105.0"
::ffff:10.8.0.5 - - [05/Oct/2022:22:56:42 +0000] "GET /favicon.ico HTTP/1.0" 404 150 "http://35.203.28.8/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.0) Gecko/20100101 Firefox/105.0"
::ffff:10.8.1.9 - - [05/Oct/2022:22:57:22 +0000] "GET / HTTP/1.0" 200 1663 "-" "curl/7.74.0"
::ffff:10.8.0.5 - - [05/Oct/2022:22:57:25 +0000] "GET / HTTP/1.0" 200 1663 "-" "curl/7.74.0"
::ffff:10.8.0.5 - - [05/Oct/2022:22:57:26 +0000] "GET / HTTP/1.0" 200 1663 "-" "curl/7.74.0"
::ffff:10.8.0.5 - - [05/Oct/2022:22:57:33 +0000] "GET / HTTP/1.0" 304 - "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.0) Gecko/20100101 Firefox/105.0"
::ffff:10.8.0.5 - - [05/Oct/2022:22:57:34 +0000] "GET / HTTP/1.0" 304 - "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.0) Gecko/20100101 Firefox/105.0"
::ffff:10.8.0.5 - - [05/Oct/2022:22:57:35 +0000] "GET / HTTP/1.0" 304 - "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.0) Gecko/20100101 Firefox/105.0"
```



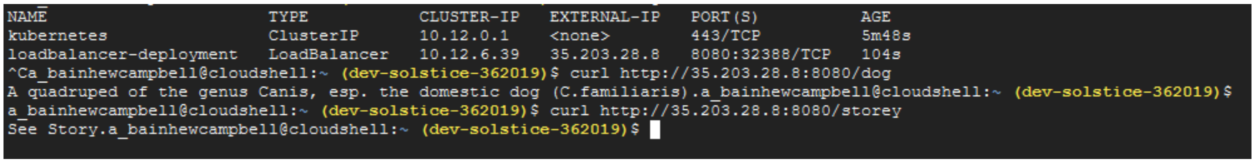
```
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl logs -l run=experiment-deployment
listening on port 80
::ffff:10.8.0.5 - - [05/Oct/2022:22:57:29 +0000] "GET / HTTP/1.0" 200 1663 "-" "curl/7.74.0"
::ffff:10.8.0.5 - - [05/Oct/2022:22:57:35 +0000] "GET / HTTP/1.0" 304 - "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.0) Gecko/20100101 Firefox/105.0"
listening on port 80
```

Part 3:

Since the pods were set up from part 2, I created the yaml file, deployed, and exposed it. Then I verified that it was working correctly by running the specified commands.



```
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl create -f loadbalancer-deployment.yaml
deployment.apps/loadbalancer-deployment created
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl get pods --output=wide
NAME                                       READY   STATUS    RESTARTS   AGE   IP           NODE                                    NOMINATED NODE   READINESS GATES
loadbalancer-deployment-6676f9ccf6-58w6x   1/1     Running   0          93s   10.8.2.8     gke-lab2-default-pool-2a786950-3rxq   <none>           <none>
loadbalancer-deployment-6676f9ccf6-g8xm6   1/1     Running   0          93s   10.8.0.8     gke-lab2-default-pool-2a786950-73bh   <none>           <none>
loadbalancer-deployment-6676f9ccf6-q4d8t   1/1     Running   0          93s   10.8.1.13    gke-lab2-default-pool-2a786950-pv4f   <none>           <none>
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl expose deployment loadbalancer-deployment --port=8080 --type=LoadBalancer
service/loadbalancer-deployment exposed
```



```
NAME                      TYPE           CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
kubernetes                ClusterIP      10.12.0.1     <none>         443/TCP          5m48s
loadbalancer-deployment   LoadBalancer   10.12.6.39    35.203.28.8    8080:32388/TCP   104s
^Ca_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ curl http://35.203.28.8:8080/dog
A quadruped of the genus Canis, esp. the domestic dog (C.familiaris).a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ curl http://35.203.28.8:8080/storey
See Story.a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ █
```

Discussion:

As discussed in part 1, the major issues that the Gateway Routing pattern solve are when you want to expose multiple services on one end point, when there are multiple versions of the same service you want to expose to various users through one end point, and when you want multiple instances of a service on one endpoint. The solution that the Gateway Routing pattern provides is having a gateway service that will route the client to the various instances that an application may have.

The requirements of this pattern are that you have multiple deployments of some service(s) and a gateway service that will route the client to a deployment of their requested service(s).

In part 2 of the lab, this pattern was seen where we had the web deployment and the experimental deployment being kept behind the gateway of the ambassador deployment. The ambassador was used as a gateway such that you would be routed to one of the web or experimental deployments. This was seen after repeatedly using the curl command and viewing the logs of each of the deployments.

In part 3 of the lab, we deploy 3 images of the dictionary-server and use the readiness probe as a gateway. This gateway checks the readiness of all 3 deployed images every 5 seconds to ensure that they are healthy.

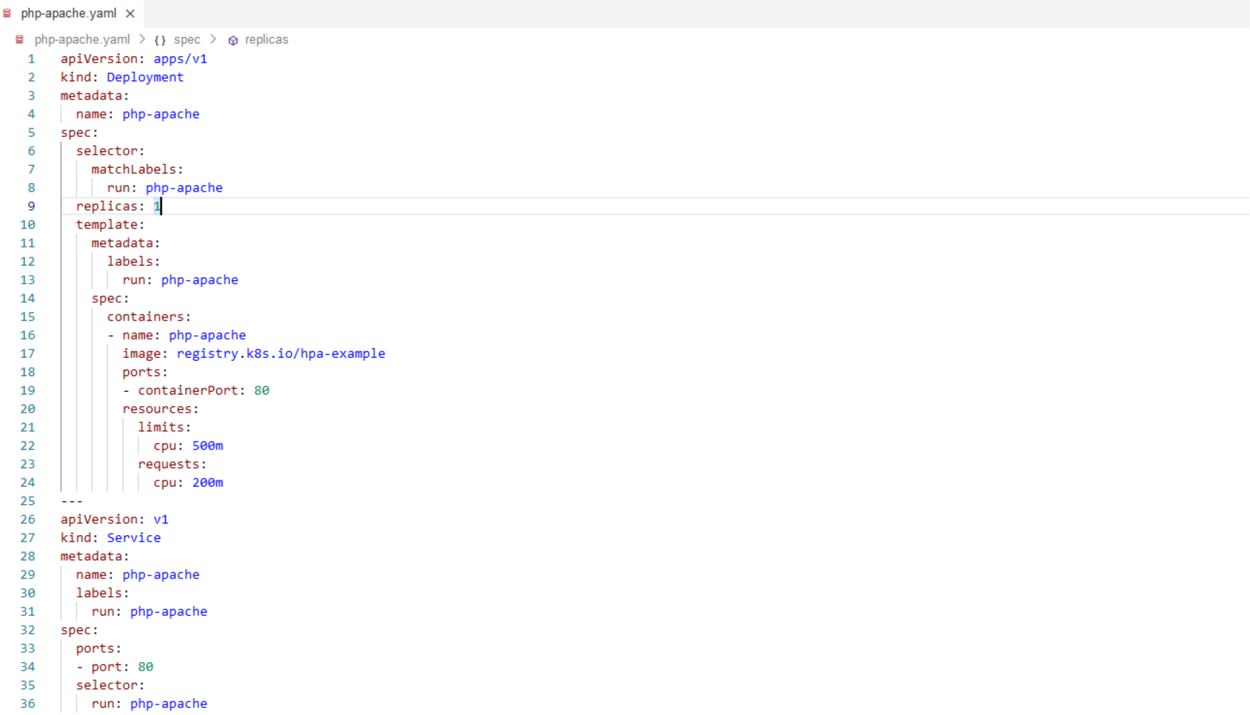
Design:

Autoscaling is a service that initiates new instances of your deployment automatically when your existing deployments are becoming overwhelmed or conversely closes deployments automatically when too many are idle or underutilized. This allows a platform to optimize the server utilization without having client performance degrade.

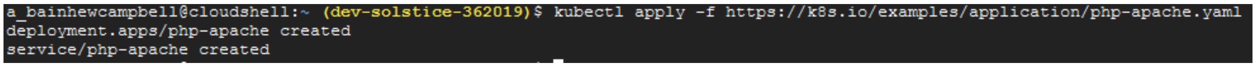
This differs from load balancing such that load balancers will balance the requests being taken in throughout existing deployments, while the autoscaler will create more deployments that the load balancer can route requests towards. Autoscaling is used when traffic to the platform is elastic as it can automatically scale up to meet the demands that a load balancer cannot do on its own, or scale down to save on computational costs.

How autoscaling is implemented can change depending on the business, however generally it involves the platform having a baseline amount of deployments that are scaled up by a load balancing service when it finds there are not enough resources for the incoming requests.

Utilizing a kubernetes autoscaler guide, it was possible to set up an autoscaler and experiment with it and see the scaling done in real-time.

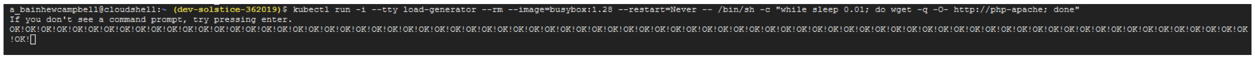


```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: php-apache
spec:
  selector:
    matchLabels:
      run: php-apache
  replicas: 1
  template:
    metadata:
      labels:
        run: php-apache
    spec:
      containers:
      - name: php-apache
        image: registry.k8s.io/hpa-example
        ports:
        - containerPort: 80
        resources:
          limits:
            cpu: 500m
          requests:
            cpu: 200m
---
apiVersion: v1
kind: Service
metadata:
  name: php-apache
  labels:
    run: php-apache
spec:
  ports:
  - port: 80
  selector:
    run: php-apache
```

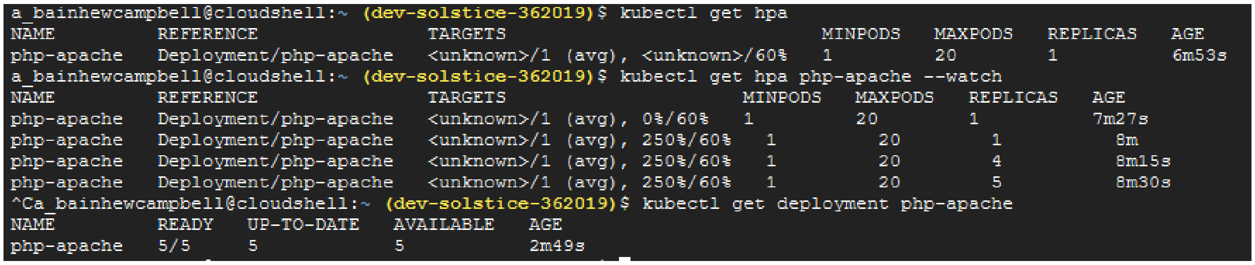


```
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl apply -f https://k8s.io/examples/application/php-apache.yaml
deployment.apps/php-apache created
service/php-apache created
```

Configuration and deployment of yaml file

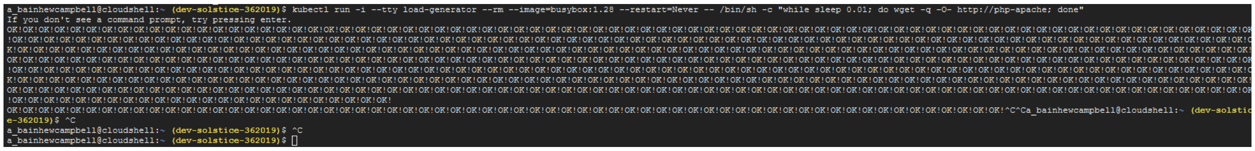


```
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl run -i --tty load-generator --rm --image=busybox:1.28 --restart=Never -- /bin/sh -c "while sleep 0.01; do wget -q -O- http://php-apache; done"
If you don't see a command prompt, try pressing enter.
OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK
!OK!
```

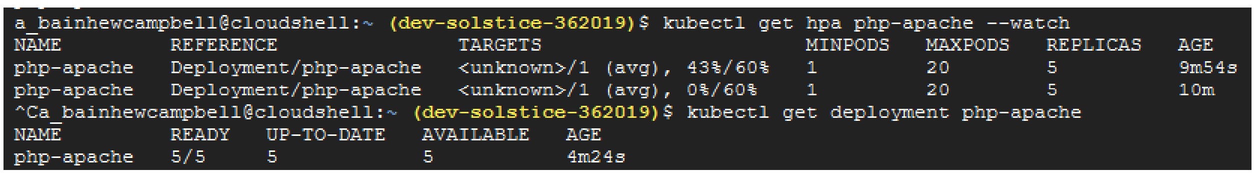


```
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl get hpa
NAME         REFERENCE               TARGETS                            MINPODS   MAXPODS   REPLICAS   AGE
php-apache   Deployment/php-apache   <unknown>/1 (avg), <unknown>/60%   1         20        1          6m53s
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl get hpa php-apache --watch
NAME         REFERENCE               TARGETS                    MINPODS   MAXPODS   REPLICAS   AGE
php-apache   Deployment/php-apache   <unknown>/1 (avg), 0%/60%   1         20        1          7m27s
php-apache   Deployment/php-apache   <unknown>/1 (avg), 250%/60%   1         20        1          8m
php-apache   Deployment/php-apache   <unknown>/1 (avg), 250%/60%   1         20        4          8m15s
php-apache   Deployment/php-apache   <unknown>/1 (avg), 250%/60%   1         20        5          8m30s
^Ca_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl get deployment php-apache
NAME         READY   UP-TO-DATE   AVAILABLE   AGE
php-apache   5/5     5            5           2m49s
```

Using a command that generates load onto the deployments so that the autoscaler kicks in and creates more replicas to meet the need



```
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl run -i --tty load-generator --rm --image=busybox:1.28 --restart=Never -- /bin/sh -c "while sleep 0.01; do wget -q -O- http://php-apache; done"
If you don't see a command prompt, try pressing enter.
OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK
!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!O
K!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!
OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK
!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!O
K!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!
OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK
!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!
OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!OK!^C^Ca_bainhewcampbell@cloudshell:~ (dev-solstic
e-362019)$ ^C
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ ^C
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ 
```



```
a_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl get hpa php-apache --watch
NAME         REFERENCE               TARGETS                      MINPODS   MAXPODS   REPLICAS   AGE
php-apache   Deployment/php-apache   <unknown>/1 (avg), 43%/60%   1         20        5          9m54s
php-apache   Deployment/php-apache   <unknown>/1 (avg), 0%/60%    1         20        5          10m
^Ca_bainhewcampbell@cloudshell:~ (dev-solstice-362019)$ kubectl get deployment php-apache
NAME         READY   UP-TO-DATE   AVAILABLE   AGE
php-apache   5/5     5            5           4m24s
```

After stopping the command we can see CPU Utilization go down