Name – Laxmi Praveena Velagapudi

Technical challenge

Create the Kubernetes cluster of your choice (eg. minikube) and provision with the following requirements:

- ingress-nginx controller to route the internal traffic in the dedicated namespace (LoadBalancer)
- Enable Horizontal Pod Autoscaller for Nginx controller (HPA) based on the CPU usage
- Create one example application and route external traffic to it under custom /app path
- Install kubernetes-dashboard using Helm chart and make it accessible through ingress (default / path)

To perform the technical challenge you will get temporary access to the custom EKS cluster hosted by storelab.

Requirements:

- aws cli
- kubectl
- helm

Please follow the instruction to configure your environment:

1. Configure your AWS credentials

```
aws configure

AWS Access Key ID:
AWS Secret Access Key:
Default region name: eu-west-2
```

2. Configure kubeconfig

```
aws eks update-kubeconfig --region eu-west-2 --name [cluster-
name]
```

3. Make sure you have access to the cluster

```
kubectl get nodes
```

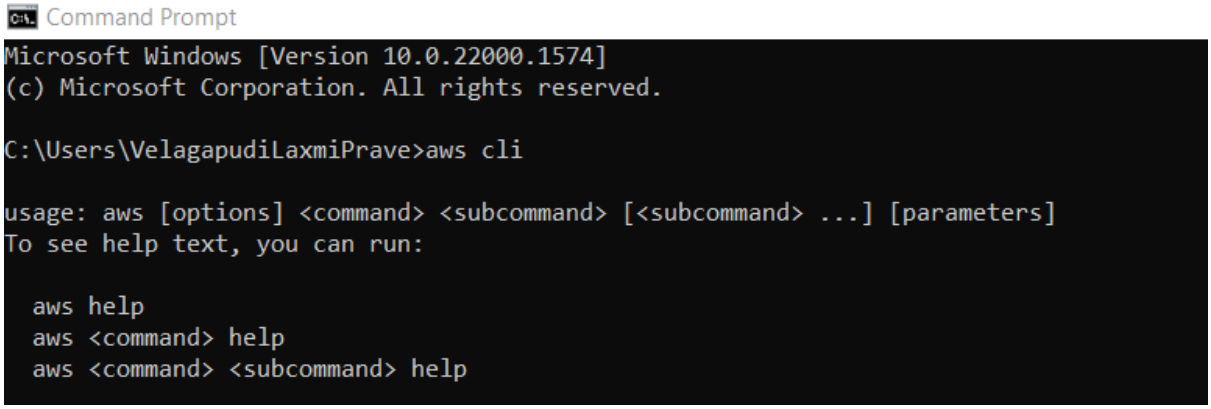
Solution

AWS Credentials-

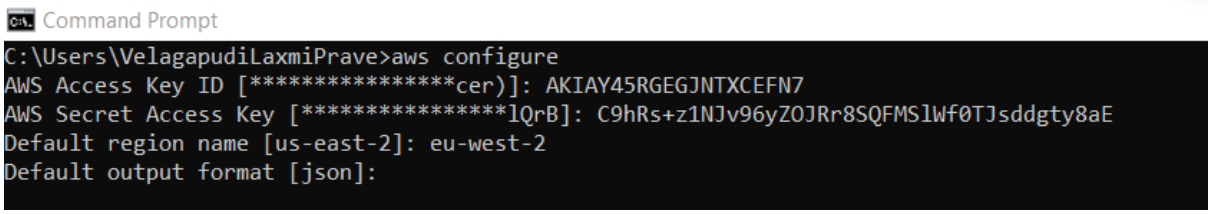
Interview sandbox 1
Access Key: AKIAY45RGEGJNTXCEFN7
Secret Key: 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
Cluster name: adorable-badger-1678788027

Step1- Connecting to EKS cluster using AWS CLI

Install AWS CLI on PowerShell and make sure aws commands are working. Then configure to the Interview sandbox1 environment using aws configure command.



Execute aws configure command to configure the credentials



Step2- kubectl Installation

Download the kubectl binary to install Kubernetes command line utility.

Command Prompt

```
curl: try 'curl --help' for more information

C:\Users\VelagapudiLaxmiPrave>curl https://kubernetes.io/docs/tasks/tools/install-kubectl-windows/
<!doctype html><html lang=en class=no-js><head><meta name=robots content="index, follow"><link rel=alterna
ternate hreflang=ko href=https://kubernetes.io/ko/docs/tasks/tools/install-kubectl-windows/><link rel=alte
tf-8><meta name=viewport content="width=device-width,initial-scale=1,shrink-to-fit=no"><meta name=generato
e-touch-icon href=/favicons/apple-touch-icon-180x180.png sizes=180x180><link rel=manifest href=/manifest.w
tl on Windows | Kubernetes</title><meta property="og:title" content="Install and Set Up kubectl on Windows
 one minor version difference of your cluster. For example, a v1.26 client can communicate with v1.25, v1.
s.
Install kubectl on Windows The following methods exist for installing kubectl on Windows:
Install kubectl binary with curl on Windows Install on Windows using Chocolatey, Scoop, or winget Install
ticle"><meta property="og:url" content="https://kubernetes.io/docs/tasks/tools/install-kubectl-windows/"><
T19:39:15+05:30"><meta property="og:site_name" content="Kubernetes"><meta itemprop=name content="Install a
 version that is within one minor version difference of your cluster. For example, a v1.26 client can comm
 avoid unforeseen issues.
Install kubectl on Windows The following methods exist for installing kubectl on Windows:
Install kubectl binary with curl on Windows Install on Windows using Chocolatey, Scoop, or winget Install
"2023-02-13T19:39:15+05:30"><meta itemprop=wordCount content="935"><meta itemprop=keywords content><meta n
s"><meta name=twitter:description content="Before you begin You must use a kubectl version that is within
26, and v1.27 control planes. Using the latest compatible version of kubectl helps avoid unforeseen issues
Install kubectl on Windows The following methods exist for installing kubectl on Windows:
Install kubectl binary with curl on Windows Install on Windows using Chocolatey, Scoop, or winget Install
etagmanager.com/gtag/js?id=G-JPP6RFM2BP"></script>
```

Execute the binary file to install kubectl and check the installation using **kubectl version –client** command.

Command Prompt

```
<script src=/js/main.min.5c0bf7f21dc4f66485f74efbbeeff28a7e4f8cddaac1bae47043159c922f
C:\Users\VelagapudiLaxmiPrave>curl.exe -LO "https://dl.k8s.io/release/v1.26.0/bin/win
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   138  100   138    0     0    252      0 --:--:-- --:--:-- --:--:--   253
100 46.4M  100 46.4M    0     0  6811k      0  0:00:06  0:00:06 --:--:-- 7864k

C:\Users\VelagapudiLaxmiPrave>curl.exe -LO "https://dl.k8s.io/v1.26.0/bin/windows/amd
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   138  100   138    0     0    738      0 --:--:-- --:--:-- --:--:--   741
100    64  100    64    0     0    166      0 --:--:-- --:--:-- --:--:--   166

C:\Users\VelagapudiLaxmiPrave>kubectl version --client
WARNING: This version information is deprecated and will be replaced with the output
Client Version: version.Info{Major:"1", Minor:"26", GitVersion:"v1.26.0", GitCommit:"
ler:"gc", Platform:"windows/amd64"}
Kustomize Version: v4.5.7
```

Step3- Connect to the eks cluster and fetch the nodes of the eks cluster.

Command Prompt

```
C:\Users\VelagapudiLaxmiPrave>aws eks update-kubeconfig --region eu-west-2 --name adorable-badger-1678788027
Added new context arn:aws:eks:eu-west-2:611867566482:cluster/adorable-badger-1678788027 to C:\Users\VelagapudiLaxmiPrave\.kube\config
```

Command Prompt

```
Kustomize Version: v4.5.7

C:\Users\VelagapudiLaxmiPrave>kubectl get nodes
NAME                                          STATUS   ROLES    AGE     VERSION
ip-192-168-15-85.eu-west-2.compute.internal   Ready    <none>   4d21h   v1.24.10-eks-48e63af
ip-192-168-37-156.eu-west-2.compute.internal  Ready    <none>   4d21h   v1.24.10-eks-48e63af
```

Step4- Creation of ingress-nginx controller

Now we will create an ingress-nginx controller to route traffic internally.

Command Prompt

```
C:\Users\VelagapudiLaxmiPrave>kubectl create namespace ingress-nginx
namespace/ingress-nginx created
```

Command Prompt

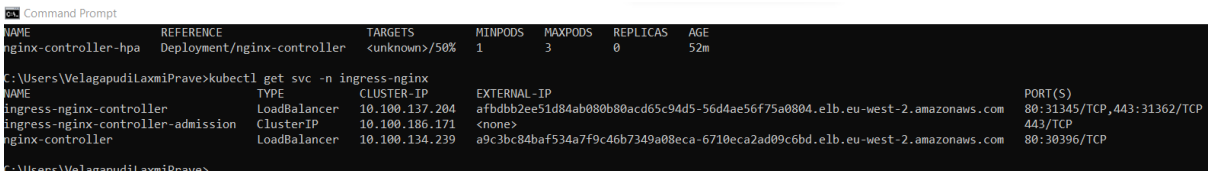
```
C:\Users\VelagapudiLaxmiPrave>
C:\Users\VelagapudiLaxmiPrave>curl https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.1.0/deploy/static/provider/aws/deploy.yaml -o deploy.yaml
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 19520  100 19520    0     0  47638      0 --:--:-- --:--:-- --:--:-- 47609
```

Command Prompt

```
operable program or batch file.

C:\Users\VelagapudiLaxmiPrave>kubectl apply -f deploy.yaml -n ingress-nginx
Warning: resource namespaces/ingress-nginx is missing the kubectl.kubernetes.i
claratively by either kubectl create --save-config or kubectl apply. The missi
namespace/ingress-nginx configured
serviceaccount/ingress-nginx created
configmap/ingress-nginx-controller created
clusterrole.rbac.authorization.k8s.io/ingress-nginx created
clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx created
role.rbac.authorization.k8s.io/ingress-nginx created
rolebinding.rbac.authorization.k8s.io/ingress-nginx created
service/ingress-nginx-controller-admission created
service/ingress-nginx-controller created
deployment.apps/ingress-nginx-controller created
ingressclass.networking.k8s.io/nginx created
validatingwebhookconfiguration.admissionregistration.k8s.io/ingress-nginx-admi
serviceaccount/ingress-nginx-admission created
clusterrole.rbac.authorization.k8s.io/ingress-nginx-admission created
clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx-admission created
role.rbac.authorization.k8s.io/ingress-nginx-admission created
rolebinding.rbac.authorization.k8s.io/ingress-nginx-admission created
job.batch/ingress-nginx-admission-create created
job.batch/ingress-nginx-admission-patch created
```

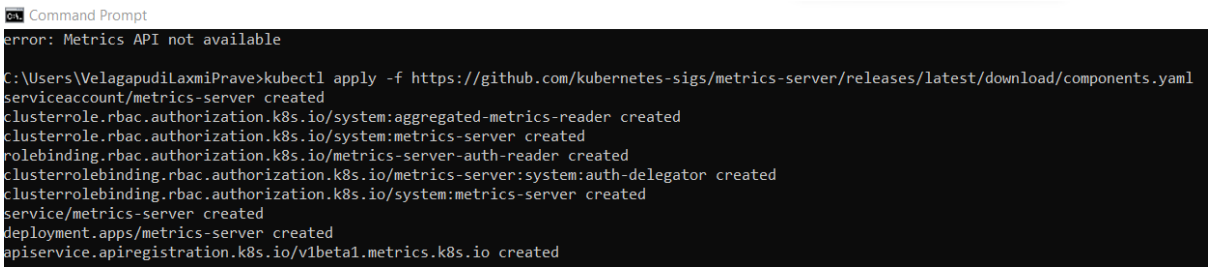
Step5- Create a service of type LoadBalancer to nginx-controller to route internal traffic using the below yaml file.

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-controller
  annotations:
    service.beta.kubernetes.io/aws-load-balancer-type: "nlb"
    service.beta.kubernetes.io/aws-load-balancer-internal: "true"
spec:
  selector:
    app: nginx-controller
  type: LoadBalancer
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
```



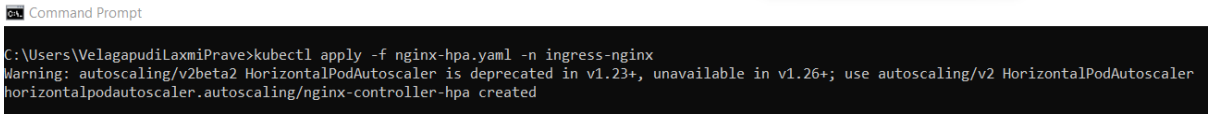
Verify the services of the ingress-controller and the traffic is internally routed.

Step-6 Install the metrics-server on the cluster to enable autoscaling

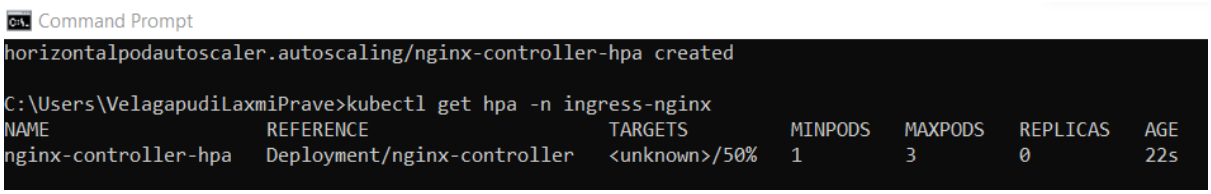


Step7- Enable horizontal pod autoscaling on nginx controller based on CPU usage.

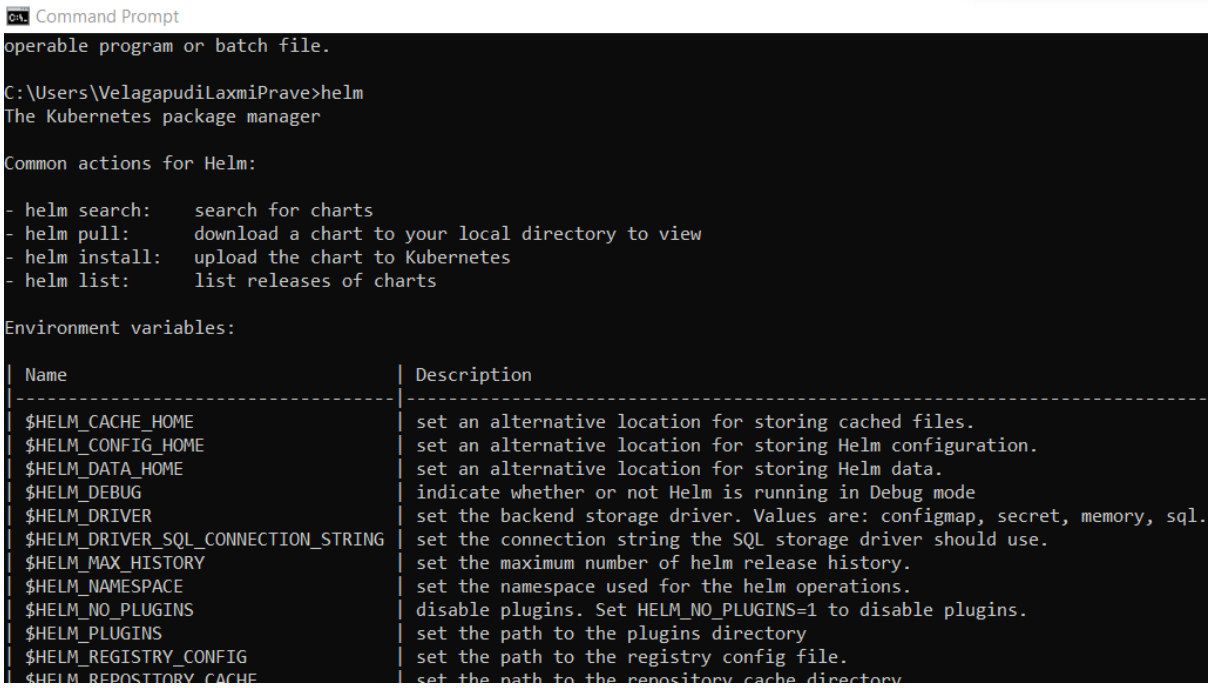
```yaml
apiVersion: autoscaling/v2beta2
kind: HorizontalPodAutoscaler
metadata:
  name: nginx-controller-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: nginx-controller
  minReplicas: 1
  maxReplicas: 3
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 50
```



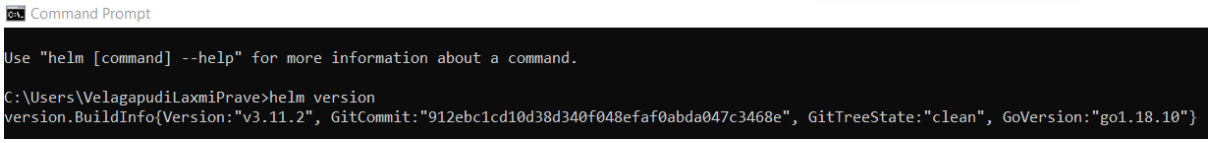
Verify the horizontal pod autoscaling for the deployment.



Step8- Download & Install helm package manager and Kubernetes-dashboard namespace. Also verify pods and services



```
Command Prompt
operable program or batch file.

C:\Users\VelagapudiLaxmiPrave>helm
The Kubernetes package manager

Common actions for Helm:

- helm search:    search for charts
- helm pull:      download a chart to your local directory to view
- helm install:   upload the chart to Kubernetes
- helm list:      list releases of charts

Environment variables:

| Name                               | Description                                                                  |
|------------------------------------|------------------------------------------------------------------------------|
| $HELM_CACHE_HOME                   | set an alternative location for storing cached files.                        |
| $HELM_CONFIG_HOME                  | set an alternative location for storing Helm configuration.                  |
| $HELM_DATA_HOME                    | set an alternative location for storing Helm data.                           |
| $HELM_DEBUG                        | indicate whether or not Helm is running in Debug mode                        |
| $HELM_DRIVER                       | set the backend storage driver. Values are: configmap, secret, memory, sql. |
| $HELM_DRIVER_SQL_CONNECTION_STRING | set the connection string the SQL storage driver should use.                 |
| $HELM_MAX_HISTORY                  | set the maximum number of helm release history.                              |
| $HELM_NAMESPACE                    | set the namespace used for the helm operations.                              |
| $HELM_NO_PLUGINS                   | disable plugins. Set HELM_NO_PLUGINS=1 to disable plugins.                   |
| $HELM_PLUGINS                      | set the path to the plugins directory                                        |
| $HELM_REGISTRY_CONFIG              | set the path to the registry config file.                                    |
| $HELM_REPOSITORY_CACHE             | set the path to the repository cache directory                               |
```



```
Command Prompt
Use "helm [command] --help" for more information about a command.

C:\Users\VelagapudiLaxmiPrave>helm version
version.BuildInfo{Version:"v3.11.2", GitCommit:"912ebc1cd10d38d340f048efaf0abda047c3468e", GitTreeState:"clean", GoVersion:"go1.18.10"}
```

The below yaml file is to install Kubernetes dashboard.

```yaml
apiVersion: v1
kind: Namespace
metadata:
  name: kubernetes-dashboard

---

apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
    k8s-app: kubernetes-dashboard
```

```yaml
  name: kubernetes-dashboard
  namespace: kubernetes-dashboard

---

kind: Service
apiVersion: v1
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard
  namespace: kubernetes-dashboard
spec:
  ports:
    - port: 443
      targetPort: 8443
      nodePort: 32000
  selector:
    k8s-app: kubernetes-dashboard
  type: NodePort

---

apiVersion: v1
kind: Secret
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard-certs
  namespace: kubernetes-dashboard
type: Opaque

---

apiVersion: v1
kind: Secret
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard-csrf
  namespace: kubernetes-dashboard
type: Opaque
data:
  csrf: ""

---

apiVersion: v1
```

```yaml
kind: Secret
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard-key-holder
  namespace: kubernetes-dashboard
type: Opaque

---

kind: ConfigMap
apiVersion: v1
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard-settings
  namespace: kubernetes-dashboard

---

kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard
  namespace: kubernetes-dashboard
rules:
  # Allow Dashboard to get, update and delete Dashboard exclusive secrets.
  - apiGroups: [""]
    resources: ["secrets"]
    resourceNames: ["kubernetes-dashboard-key-holder", "kubernetes-
dashboard-certs", "kubernetes-dashboard-csrf"]
    verbs: ["get", "update", "delete"]
    # Allow Dashboard to get and update 'kubernetes-dashboard-settings'
config map.
  - apiGroups: [""]
    resources: ["configmaps"]
    resourceNames: ["kubernetes-dashboard-settings"]
    verbs: ["get", "update"]
    # Allow Dashboard to get metrics.
  - apiGroups: [""]
    resources: ["services"]
    resourceNames: ["heapster", "dashboard-metrics-scraper"]
    verbs: ["proxy"]
  - apiGroups: [""]
    resources: ["services/proxy"]
```

```
    resourceNames: ["heapster", "http:heapster:", "https:heapster:",
"dashboard-metrics-scraper", "http:dashboard-metrics-scraper"]
    verbs: ["get"]

---

kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard
rules:
  # Allow Metrics Scraper to get metrics from the Metrics server
  - apiGroups: ["metrics.k8s.io"]
    resources: ["pods", "nodes"]
    verbs: ["get", "list", "watch"]

---

apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard
  namespace: kubernetes-dashboard
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: kubernetes-dashboard
subjects:
  - kind: ServiceAccount
    name: kubernetes-dashboard
    namespace: kubernetes-dashboard

---

apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: kubernetes-dashboard
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: kubernetes-dashboard
subjects:
  - kind: ServiceAccount
```

```
    name: kubernetes-dashboard
    namespace: kubernetes-dashboard

---

kind: Deployment
apiVersion: apps/v1
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard
  namespace: kubernetes-dashboard
spec:
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      k8s-app: kubernetes-dashboard
  template:
    metadata:
      labels:
        k8s-app: kubernetes-dashboard
    spec:
      containers:
        - name: kubernetes-dashboard
          image: kubernetesui/dashboard:v2.4.0
          imagePullPolicy: Always
          ports:
            - containerPort: 8443
              protocol: TCP
          args:
            - --auto-generate-certificates
            - --namespace=kubernetes-dashboard
            # Uncomment the following line to manually specify Kubernetes
API server Host
            # If not specified, Dashboard will attempt to auto discover the
API server and connect
            # to it. Uncomment only if the default does not work.
            # - --apiserver-host=http://my-address:port
          volumeMounts:
            - name: kubernetes-dashboard-certs
              mountPath: /certs
              # Create on-disk volume to store exec logs
            - mountPath: /tmp
              name: tmp-volume
          livenessProbe:
            httpGet:
              scheme: HTTPS
```

```
            path: /
            port: 8443
          initialDelaySeconds: 30
          timeoutSeconds: 30
        securityContext:
          allowPrivilegeEscalation: false
          readOnlyRootFilesystem: true
          runAsUser: 1001
          runAsGroup: 2001
      volumes:
        - name: kubernetes-dashboard-certs
          secret:
            secretName: kubernetes-dashboard-certs
        - name: tmp-volume
          emptyDir: {}
      serviceAccountName: kubernetes-dashboard
      nodeSelector:
        "kubernetes.io/os": linux
      # Comment the following tolerations if Dashboard must not be deployed
on master
      tolerations:
        - key: node-role.kubernetes.io/master
          effect: NoSchedule

---

kind: Service
apiVersion: v1
metadata:
  labels:
    k8s-app: dashboard-metrics-scraper
  name: dashboard-metrics-scraper
  namespace: kubernetes-dashboard
spec:
  ports:
    - port: 8000
      targetPort: 8000
  selector:
    k8s-app: dashboard-metrics-scraper

---

kind: Deployment
apiVersion: apps/v1
metadata:
  labels:
    k8s-app: dashboard-metrics-scraper
  name: dashboard-metrics-scraper
```

```yaml
  namespace: kubernetes-dashboard
spec:
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      k8s-app: dashboard-metrics-scraper
  template:
    metadata:
      labels:
        k8s-app: dashboard-metrics-scraper
    spec:
      securityContext:
        seccompProfile:
          type: RuntimeDefault
      containers:
        - name: dashboard-metrics-scraper
          image: kubernetesui/metrics-scraper:v1.0.7
          ports:
            - containerPort: 8000
              protocol: TCP
          livenessProbe:
            httpGet:
              scheme: HTTP
              path: /
              port: 8000
            initialDelaySeconds: 30
            timeoutSeconds: 30
          volumeMounts:
          - mountPath: /tmp
            name: tmp-volume
          securityContext:
            allowPrivilegeEscalation: false
            readOnlyRootFilesystem: true
            runAsUser: 1001
            runAsGroup: 2001
      serviceAccountName: kubernetes-dashboard
      nodeSelector:
        "kubernetes.io/os": linux
      # Comment the following tolerations if Dashboard must not be deployed
on master
      tolerations:
        - key: node-role.kubernetes.io/master
          effect: NoSchedule
      volumes:
        - name: tmp-volume
          emptyDir: {}
---
```

```yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: kops-admin
  namespace: kube-system
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: kops-admin
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: kops-admin
  namespace: kube-system
---

apiVersion: v1
kind: Secret
type: kubernetes.io/service-account-token
metadata:
  namespace: kube-system
  name: kops-admin
  annotations:
    kubernetes.io/service-account.name: "kops-admin"
```

```
C:\Users\VelagapudiLaxmiPrave>kubectl apply -f k8sdashboard.yaml
Warning: resource namespaces/kubernetes-dashboard is missing the kubectl.kubernetes.io/
ated declaratively by either kubectl create --save-config or kubectl apply. The missing
namespace/kubernetes-dashboard configured
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
serviceaccount/kops-admin created
clusterrolebinding.rbac.authorization.k8s.io/kops-admin created
secret/kops-admin created
```

```
C:\Users\VelagapudiLaxmiPrave>kubectl -n kubernetes-dashboard get svc
NAME                        TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)         AGE
dashboard-metrics-scraper   ClusterIP   10.100.42.42    <none>        8000/TCP        2m36s
kubernetes-dashboard        NodePort    10.100.163.14   <none>        443:32000/TCP   2m36s

C:\Users\VelagapudiLaxmiPrave>kubectl -n kubernetes-dashboard get pod -o wide
NAME                                         READY   STATUS    RESTARTS   AGE     IP              NODE                                              NOMINATED NODE   READINESS GATES
dashboard-metrics-scraper-7bfdf779ff-k6lqm   1/1     Running   0          3m18s   192.168.26.205  ip-192-168-15-85.eu-west-2.compute.internal   <none>           <none>
kubernetes-dashboard-b65bbf7d9-l84js         1/1     Running   0          3m19s   192.168.2.12    ip-192-168-15-85.eu-west-2.compute.internal   <none>           <none>
```

Step9- Deploy the sample nginx deployment and expose the application.

Command Prompt

```
error: failed to create deployment: namespaces "kubernetes-dashbaord" not found

C:\Users\VelagapudiLaxmiPrave>kubectl -n kubernetes-dashboard create deploy my-app --image nginx --port 80
deployment.apps/my-app created
```

Command Prompt

```
deployment.apps/my-app created

C:\Users\VelagapudiLaxmiPrave>kubectl -n kubernetes-dashboard expose deploy my-app --name my-app-svc --port 80 --target-port 80 --type NodePort
service/my-app-svc exposed
```

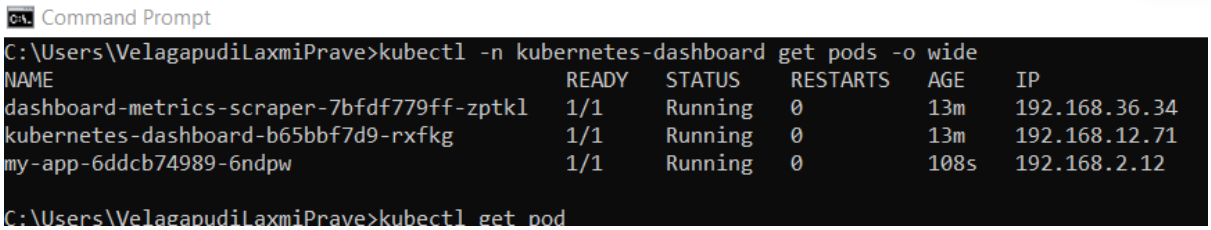
Step10- Create an ingress to split the traffic between Kubernetes dashboard and sample application based on the rules specified.

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: my-ingress
  namespace: kubernetes-dashboard
  annotations:
    kubernetes.io/ingress.class: alb
spec:
  rules:
  - http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: kubernetes-dashboard
            port:
              number: 443
      - path: /app
        pathType: Prefix
        backend:
          service:
            name: my-app-svc
            port:
              number: 80
```

Step11- Verify the traffic split by describing the ingress created in above step and also verify with pods IP address

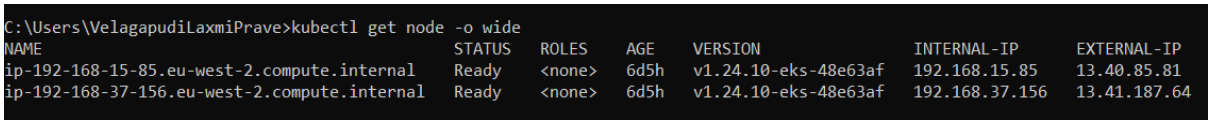
Command Prompt

```
service/my-app-svc exposed

C:\Users\VelagapudiLaxmiPrave>kubectl describe ingress my-ingress -n kubernetes-dashboard
Name:             my-ingress
Labels:           <none>
Namespace:        kubernetes-dashboard
Address:
Ingress Class:    <none>
Default backend:  <default>
Rules:
  Host        Path  Backends
  ----        ----  --------
  *
              /      kubernetes-dashboard:443 (192.168.12.71:8443)
              /app   my-app-svc:80 (192.168.2.12:80)
Annotations:  nginx.ingress.kubernetes.io/rewrite-target: /
Events:       <none>
```

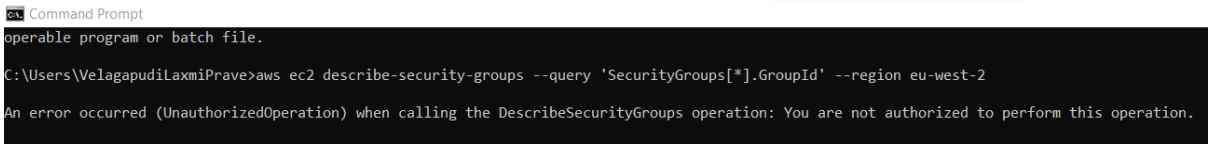


Through this process we could split the traffic through ingress to dashboard and the sample application.

You could also copy the external IP address of the ec2-instance and check the output in the browser.



Note-Make sure desired ports are opened in the security group, to view the traffic. Here I don't have permission to add ports to security group, could not do that step.



Once the ports are opened in the security group, we can view the dashboard and application traffic through browser.