UNIVERSITY WITH A PURPOSE

UNIVERSITY OF PETROLEUM & ENERGY STUDIES

Dehradun

Application Containerization

Experiment 8

Name:	**Devashish Choudhary**
Course:	**B-Tech CSE DevOps (2018-22)**
Roll number:	**R171218122**
Sap ID:	**500070510**

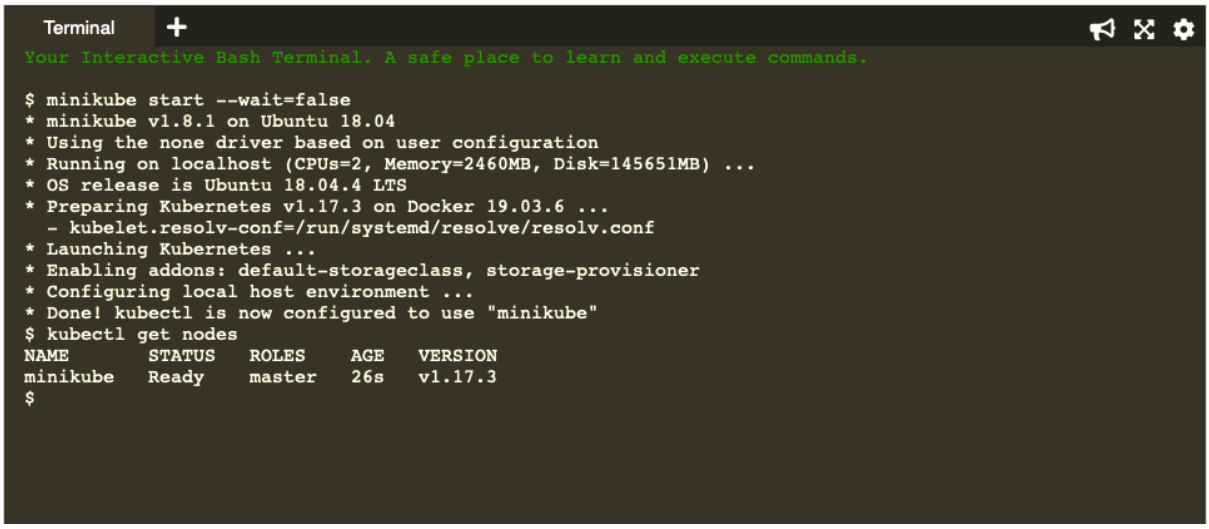
○ Start containers using Kubectl

- **Launch Cluster**
  Execute the command below to start the cluster components and download the Kubectl CLI.

  ```
  minikube start --wait=false
  ```

  Wait for the Node to become Ready by checking

  ```
  kubectl get nodes
  ```



- **Kubectl run**
  The following command will launch a deployment called *http* which will start a container based on the Docker Image *katacoda/docker-http-server:latest*.

  ```
  kubectl run http --image=katacoda/docker-http-server:latest --replicas=1
  ```

  You can then use kubectl to view the status of the deployments

  ```
  kubectl get deployments
  ```

  To find out what Kubernetes created you can describe the deployment process.

  ```
  kubectl describe deployment http
  ```

The description includes how many replicas are available, labels specified and the events associated with the deployment. These events will highlight any problems and errors that might have occurred.

```
Terminal
$ kubectl run http --image=katacoda/docker-http-server:latest --replicas=1
kubectl run --generator=deployment/apps.v1 is DEPRECATED and will be removed in a future version. Use ku
bectl run --generator=run-pod/v1 or kubectl create instead.
deployment.apps/http created
$ kubectl get deployments
NAME   READY   UP-TO-DATE   AVAILABLE   AGE
http   1/1     1            1           4m8s
$ kubectl describe deployment http
Name:                   http
Namespace:              default
CreationTimestamp:      Thu, 08 Apr 2021 18:43:55 +0000
Labels:                 run=http
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               run=http
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:  run=http
  Containers:
   http:
    Image:        katacoda/docker-http-server:latest
    Port:         <none>
    Host Port:    <none>
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   http-774bb756bb (1/1 replicas created)
Events:
  Type    Reason             Age    From                   Message
  ----    ------             ----   ----                   -------
  Normal  ScalingReplicaSet  4m10s  deployment-controller  Scaled up replica set http-774bb756bb to 1
$
```

- **Kubectl Expose**

  Use the following command to expose the container port *80* on the host *8000* binding to the *external-ip* of the host.

```
kubectl expose deployment http --external-ip="172.17.0.9" --
port=8000 --target-port=80
```

  You will then be able to ping the host and see the result from the HTTP service.

```
curl http://172.17.0.9:8000
```

```
Terminal
$ kubectl expose deployment http --external-ip="172.17.0.9" --port=8000 --target-port=80
service/http exposed
$ curl http://172.17.0.9:8000
<h1>This request was processed by host: http-774bb756bb-ljg6n</h1>
$
```

- **Kubectl Run and Expose**
  Use the command to create a second http service exposed on port *8001*.

  ```
  kubectl run httpexposed --image=katacoda/docker-http-server:latest --replicas=1 --port=80 --hostport=8001
  ```

  You should be able to access it using.

  ```
  curl http://172.17.0.9:8001
  ```

  Under the covers, this exposes the Pod via Docker Port Mapping. As a result, you will not see the service listed using

  ```
  Kubectl get svc
  ```

  To find the details you can use

  ```
  docker ps | grep httpexposed
  ```

```
Terminal
$ kubectl run httpexposed --image=katacoda/docker-http-server:latest --replicas=1 --port=80 --hostport=8001
kubectl run --generator=deployment/apps.v1 is DEPRECATED and will be removed in a future version. Use kubectl run
 --generator=run-pod/v1 or kubectl create instead.
deployment.apps/httpexposed created
$ curl http://172.17.0.9:8001
<h1>This request was processed by host: httpexposed-68cb8c8d4-2cx7l</h1>
$ kubectl get svc
NAME          TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
http          ClusterIP   10.106.45.109   172.17.0.9    8000/TCP   4m30s
kubernetes    ClusterIP   10.96.0.1       <none>        443/TCP    21m
$ docker ps | grep httpexposed
86d271b143c2        katacoda/docker-http-server   "/app"                   7 seconds ago       Up 6 seconds
                    k8s_httpexposed_httpexposed-68cb8c8d4-2cx7l_default_76bc9711-c641-41c7-b47d-5ab4e8b8f008
_0
40d4a318e4c6        k8s.gcr.io/pause:3.1          "/pause"                 9 seconds ago       Up 8 seconds
  0.0.0.0:8001->80/tcp   k8s_POD_httpexposed-68cb8c8d4-2cx7l_default_76bc9711-c641-41c7-b47d-5ab4e8b8f008_0
$
```

- **Scale Containers**

The command *kubectl scale* allows us to adjust the number of Pods running for a particular deployment or replication controller.

```
kubectl scale --replicas=3 deployment http
```

Listing all the pods, you should see three running for the *http* deployment

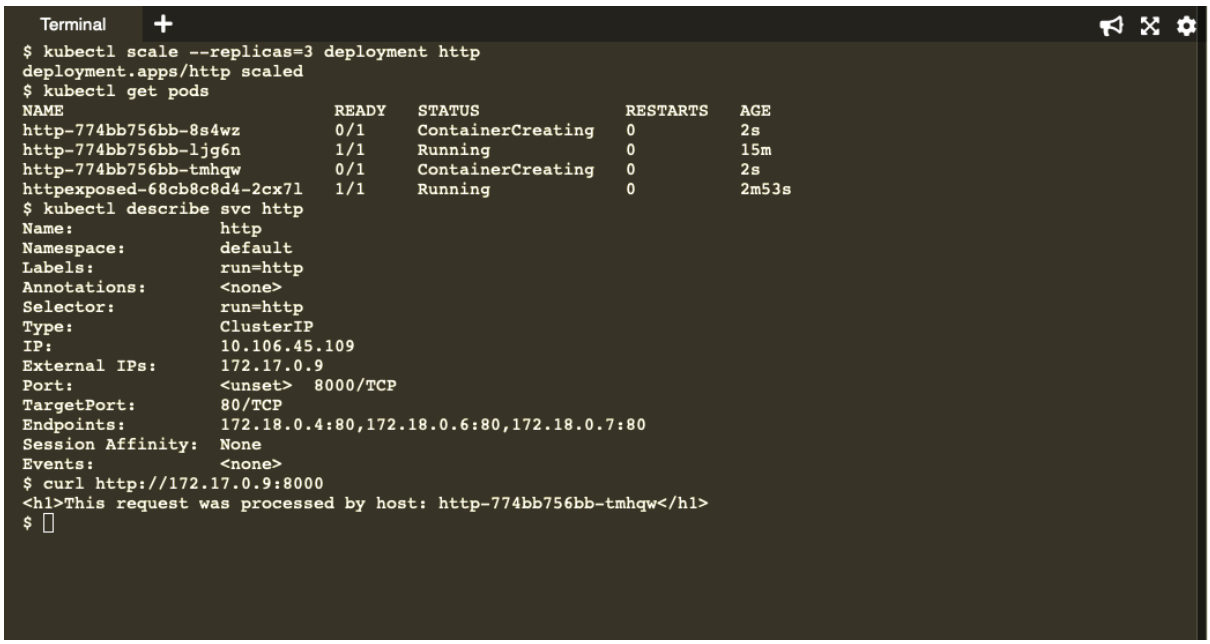
```
kubectl get pods
```

Once each Pod starts it will be added to the load balancer service. By describing the service you can view the endpoint and the associated Pods which are included.

```
kubectl describe svc http
```

Making requests to the service will request in different nodes processing the request.

```
curl http://172.17.0.9:8000
```



○ Launch Single Node Kubernetes Cluster

- **Start Minikube**

  Minikube has been installed and configured in the environment. Check that it is properly installed, by running the *minikube version* command:

  ```
  minikube version
  ```

  Start the cluster, by running the *minikube start* command:

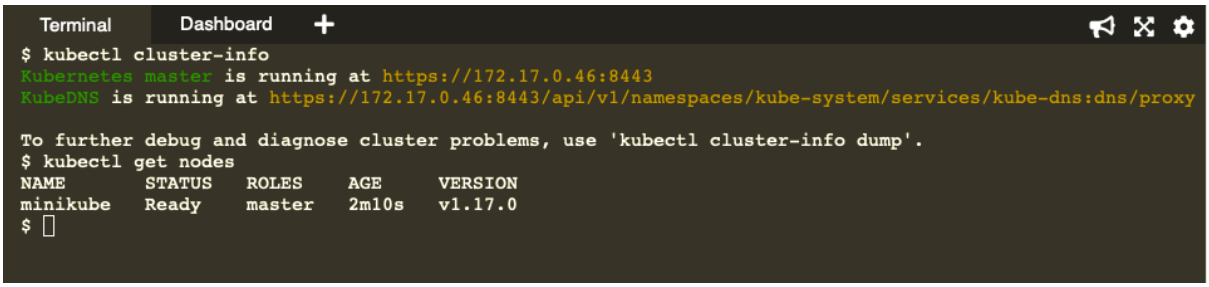
  ```
  minikube start --wait=false
  ```

- **Cluster info**

  Details of the cluster and its health status can be discovered via
  `kubectl cluster-info`

  To view the nodes in the cluster using
  `kubectl get nodes`

  If the node is marked as **NotReady** then it is still starting the components.



- **Deploy Containers**

  Using `kubectl run`, it allows containers to be deployed onto the cluster-

```
kubectl create deployment first-deployment --image=katacoda/docker-http-server
```

The status of the deployment can be discovered via the running Pods -

```
kubectl get pods
```

Once the container is running it can be exposed via different networking options, depending on requirements. One possible solution is NodePort, that provides a dynamic port to a container.

```
kubectl expose deployment first-deployment --port=80 --type=NodePort
```

The command below finds the allocated port and executes a HTTP request.

```
export PORT=$(kubectl get svc first-deployment -o go-template='{{range.spec.ports}}{{if .nodePort}}{{.nodePort}}{{"\n"}}{{end}}{{end}}') echo "Accessing host01:$PORT" curl host01:$PORT
```

- **Dashboard**

  Enable the dashboard using Minikube with the command `minikube addons enable dashboard`

Make the Kubernetes Dashboard available by deploying the following YAML definition. This should only be used on Katacoda.

```
kubectl apply -f /opt/kubernetes-dashboard.yaml
```

The Kubernetes dashboard allows you to view your applications in a UI. In this deployment, the dashboard has been made available on port *30000* but may take a while to start.

To see the progress of the Dashboard starting, watch the Pods within the *kube-system* namespace using

```
kubectl get pods -n kubernetes-dashboard -w
```

Once running, the URL to the dashboard is https://2886795310-30000-jago01.environments.katacoda.com/