EXPERIMENT: 8

AIM- Introduction to kubernetes

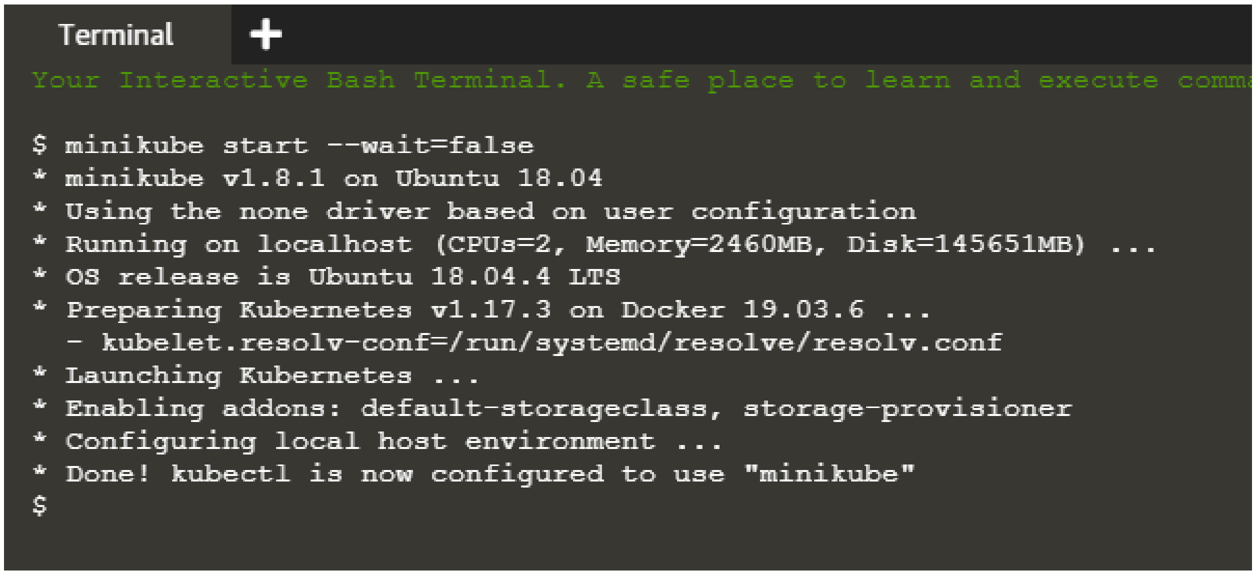
Step 1 - Launch Cluster

To start we need to launch a Kubernetes cluster.

Execute the command below to start the cluster components and download the Kubectl CLI.

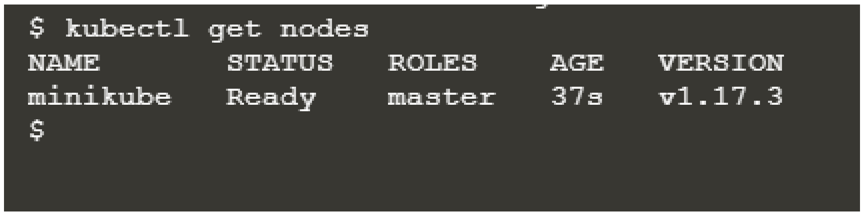
`minikube start --wait=false`

Wait for the Node to become Ready by checking `kubectl get node`



Terminal

```
Your Interactive Bash Terminal. A safe place to learn and execute comma

$ minikube start --wait=false
* minikube v1.8.1 on Ubuntu 18.04
* Using the none driver based on user configuration
* Running on localhost (CPUs=2, Memory=2460MB, Disk=145651MB) ...
* OS release is Ubuntu 18.04.4 LTS
* Preparing Kubernetes v1.17.3 on Docker 19.03.6 ...
  - kubelet.resolv-conf=/run/systemd/resolve/resolv.conf
* Launching Kubernetes ...
* Enabling addons: default-storageclass, storage-provisioner
* Configuring local host environment ...
* Done! kubectl is now configured to use "minikube"
$
```



```
$ kubectl get nodes
NAME        STATUS   ROLES    AGE   VERSION
minikube    Ready    master   37s   v1.17.3
$
```

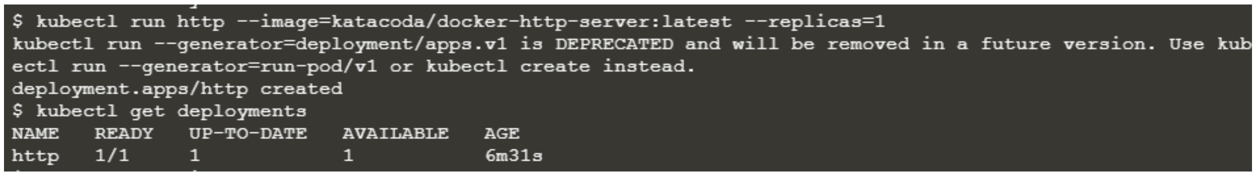
Step 2 - Kubectl Run

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

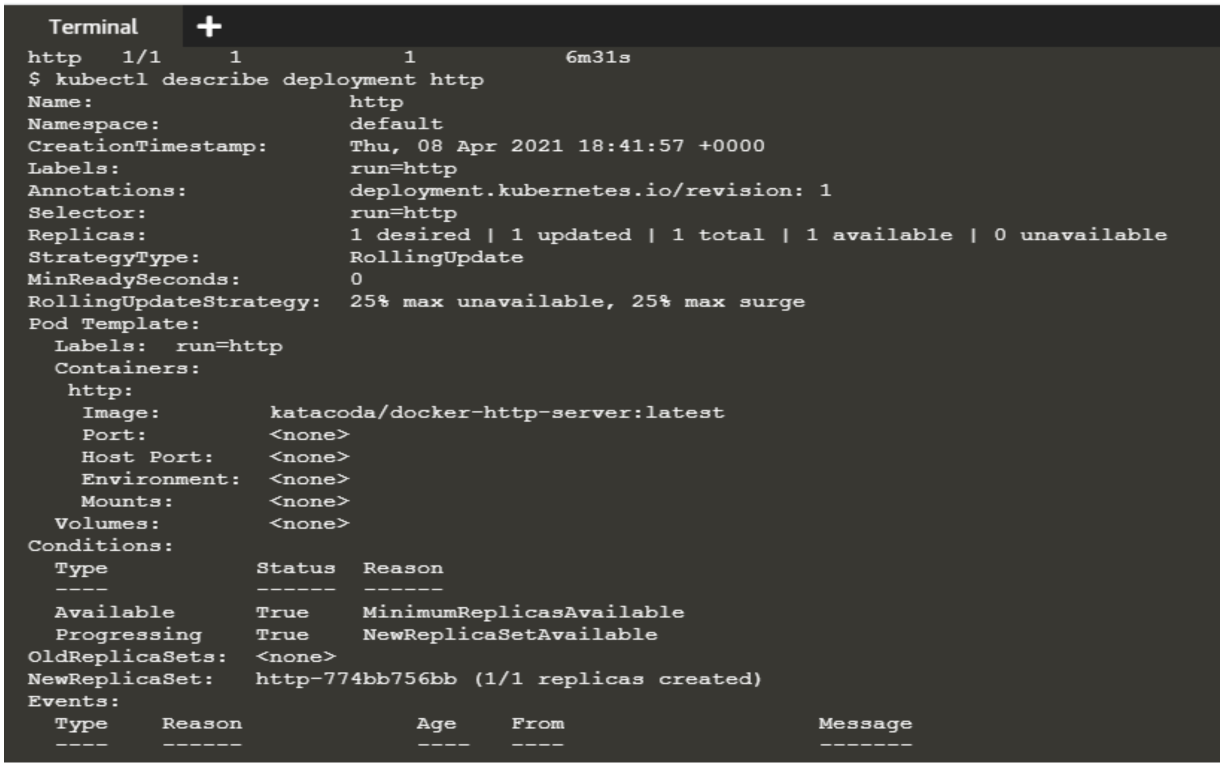
```
kubectl get deployments
```



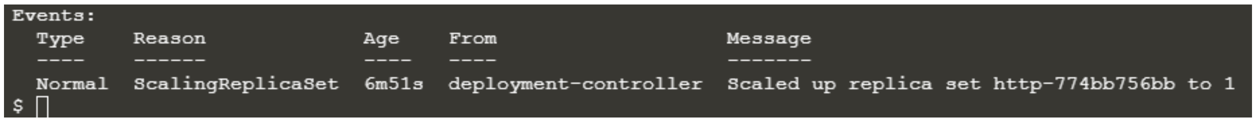
```
$ kubectl run http --image=katacoda/docker-http-server:latest --replicas=1
kubectl run --generator=deployment/apps.v1 is DEPRECATED and will be removed in a future version. Use kub
ectl run --generator=run-pod/v1 or kubectl create instead.
deployment.apps/http created
$ kubectl get deployments
NAME   READY   UP-TO-DATE   AVAILABLE   AGE
http   1/1     1            1           6m31s
```

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

```
kubectl describe deployment http
```



```
http   1/1     1            1           6m31s
$ kubectl describe deployment http
Name:                   http
Namespace:              default
CreationTimestamp:      Thu, 08 Apr 2021 18:41:57 +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
  ----    ------             ----   ----                   -------
```



```
Events:
  Type    Reason             Age    From                   Message
  ----    ------             ----   ----                   -------
  Normal  ScalingReplicaSet  6m51s  deployment-controller  Scaled up replica set http-774bb756bb to 1
$ 
```

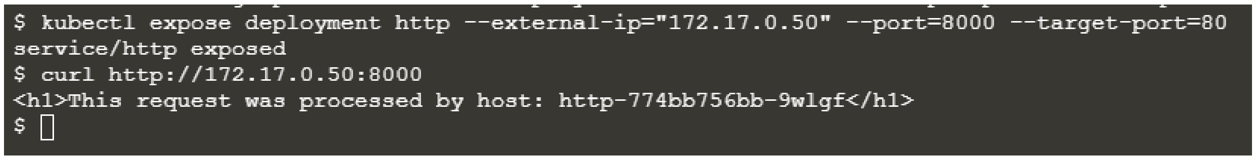
Step 3 - Kubectl Expose

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="172.17.0.50" --port=8000 --target-port=80
```

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

```
curl http://172.17.0.50:8000
```



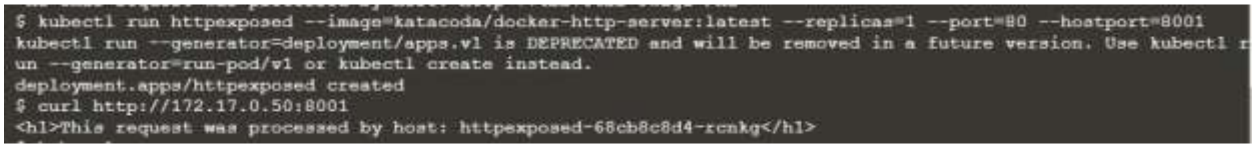
```
$ kubectl expose deployment http --external-ip="172.17.0.50" --port=8000 --target-port=80
service/http exposed
$ curl http://172.17.0.50:8000
<h1>This request was processed by host: http-774bb756bb-9wlgf</h1>
$
```

Step 4 - Kubectl Run and Expose

Use the command 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
```

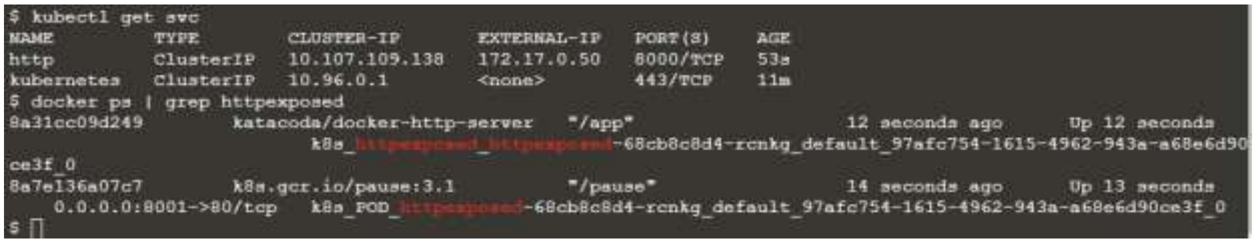
You should be able to access it using `curl http://172.17.0.50:8001`



```
$ 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 r
un --generator=run-pod/v1 or kubectl create instead.
deployment.apps/httpexposed created
$ curl http://172.17.0.50:8001
<h1>This request was processed by host: httpexposed-68cb8c8d4-rcnkg</h1>
```

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`



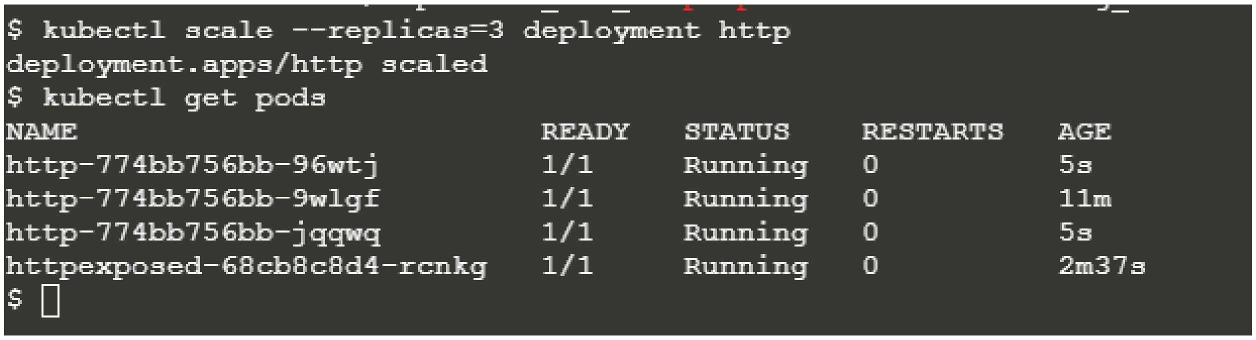
```
$ kubectl get svc
NAME         TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
http         ClusterIP   10.107.109.138   172.17.0.50   8000/TCP   53s
kubernetes   ClusterIP   10.96.0.1        <none>        443/TCP    11m
$ docker ps | grep httpexposed
8a31cc09d249        katacoda/docker-http-server   "/app"                   12 seconds ago      Up 12 seconds
                    k8s_httpexposed_httpexposed-68cb8c8d4-rcnkg_default_97afc754-1615-4962-943a-a68e6d90
ce3f_0
8a7e136a07c7        k8s.gcr.io/pause:3.1          "/pause"                 14 seconds ago      Up 13 seconds
    0.0.0.0:8001->80/tcp   k8s_POD_httpexposed-68cb8c8d4-rcnkg_default_97afc754-1615-4962-943a-a68e6d90ce3f_0
$ 
```

Step 5 - 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`



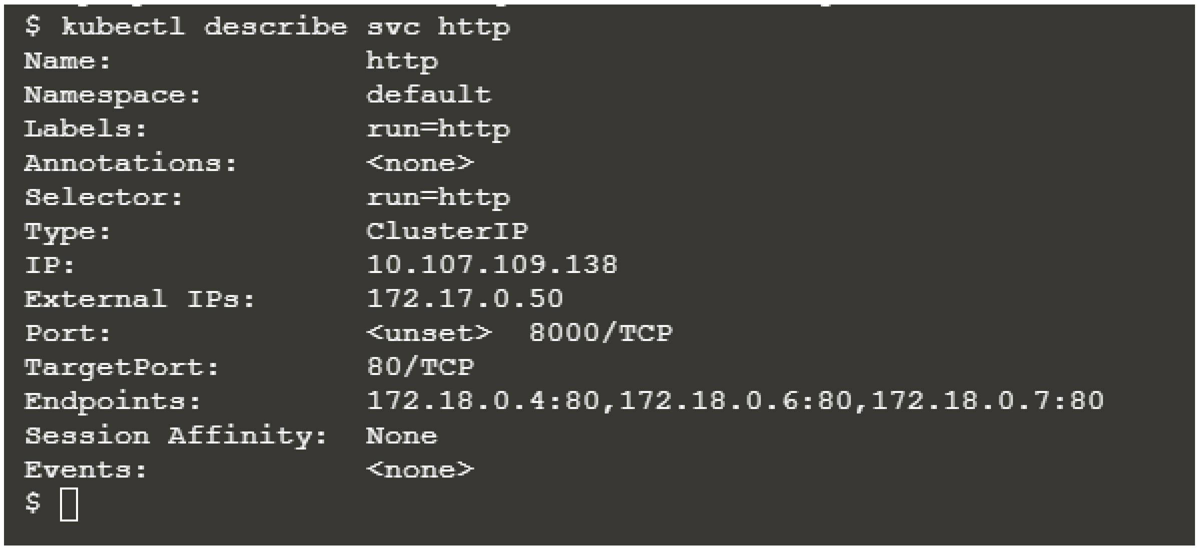
```
$ kubectl scale --replicas=3 deployment http
deployment.apps/http scaled
$ kubectl get pods
NAME                          READY   STATUS    RESTARTS   AGE
http-774bb756bb-96wtj         1/1     Running   0          5s
http-774bb756bb-9wlgf         1/1     Running   0          11m
http-774bb756bb-jqqwq         1/1     Running   0          5s
httpexposed-68cb8c8d4-rcnkg   1/1     Running   0          2m37s
$ 
```

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.

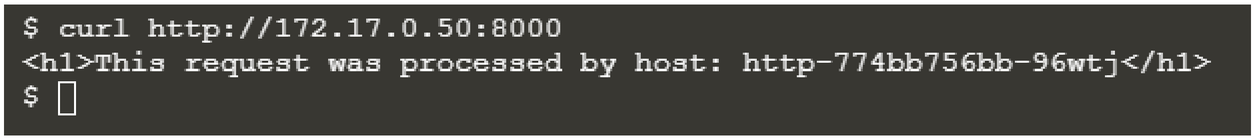
`kubectl describe svc http`

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

`curl http://172.17.0.50:8000`



```
$ kubectl describe svc http
Name:              http
Namespace:         default
Labels:            run=http
Annotations:       <none>
Selector:          run=http
Type:              ClusterIP
IP:                10.107.109.138
External IPs:      172.17.0.50
Port:              <unset>  8000/TCP
TargetPort:        80/TCP
Endpoints:         172.18.0.4:80,172.18.0.6:80,172.18.0.7:80
Session Affinity:  None
Events:            <none>
$
```



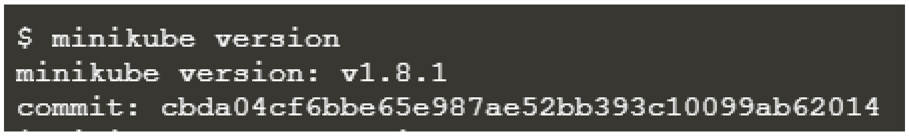
```
$ curl http://172.17.0.50:8000
<h1>This request was processed by host: http-774bb756bb-96wtj</h1>
$
```

b) Launch Single Node Kubernetes Cluster

Step 1 - Start Minikube

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

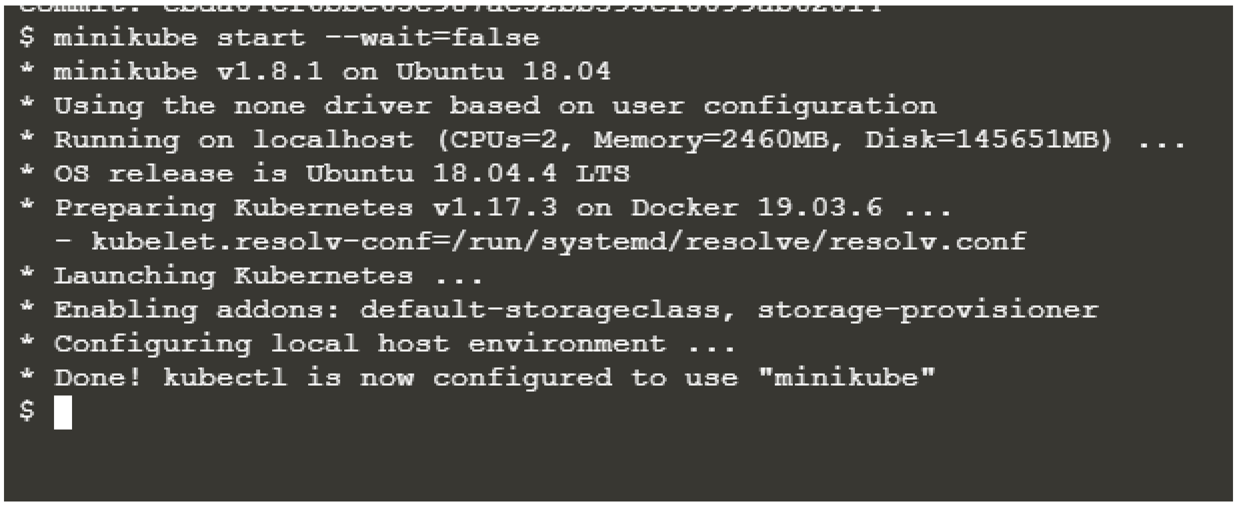
`minikube version`



```
$ minikube version
minikube version: v1.8.1
commit: cbda04cf6bbe65e987ae52bb393c10099ab62014
```

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

`minikube start --wait=false`

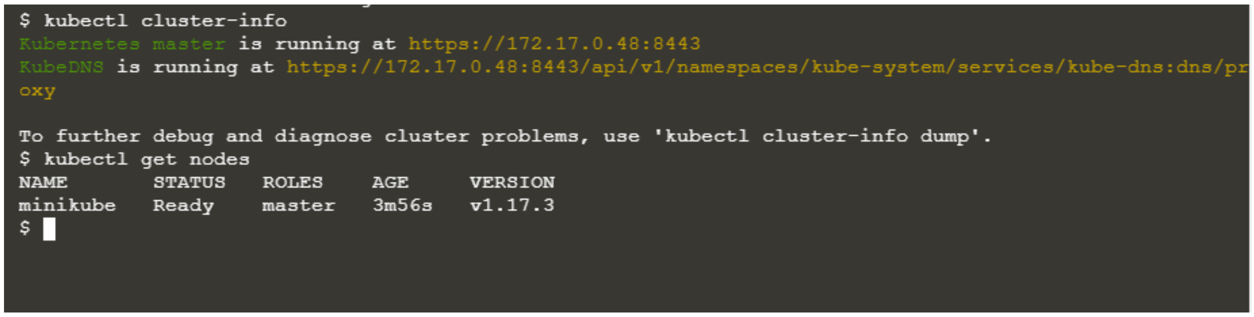


```
$ minikube start --wait=false
* minikube v1.8.1 on Ubuntu 18.04
* Using the none driver based on user configuration
* Running on localhost (CPUs=2, Memory=2460MB, Disk=145651MB) ...
* OS release is Ubuntu 18.04.4 LTS
* Preparing Kubernetes v1.17.3 on Docker 19.03.6 ...
  - kubelet.resolv-conf=/run/systemd/resolve/resolv.conf
* Launching Kubernetes ...
* Enabling addons: default-storageclass, storage-provisioner
* Configuring local host environment ...
* Done! kubectl is now configured to use "minikube"
$
```

Step 2 - 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`



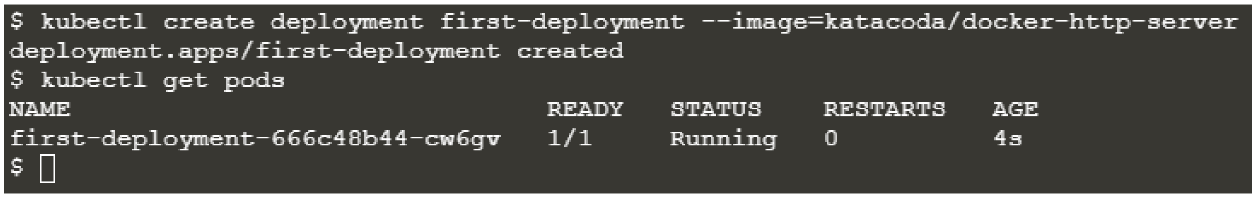
```
$ kubectl cluster-info
Kubernetes master is running at https://172.17.0.48:8443
KubeDNS is running at https://172.17.0.48:8443/api/v1/namespaces/kube-system/services/kube-dns:dns/pr
oxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
$ kubectl get nodes
NAME       STATUS   ROLES    AGE     VERSION
minikube   Ready    master   3m56s   v1.17.3
$
```

Step 3 - Deploy Containers

With a running Kubernetes cluster, containers can now be deployed.

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`

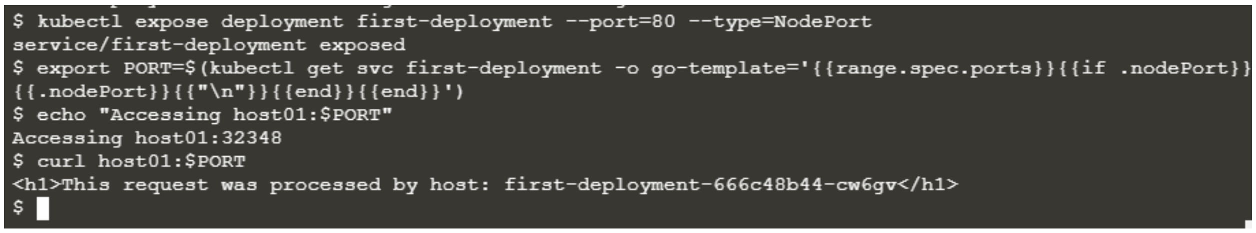

```
$ kubectl create deployment first-deployment --image=katacoda/docker-http-server
deployment.apps/first-deployment created
$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
first-deployment-666c48b44-cw6gv    1/1     Running   0          4s
$ 
```

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
```

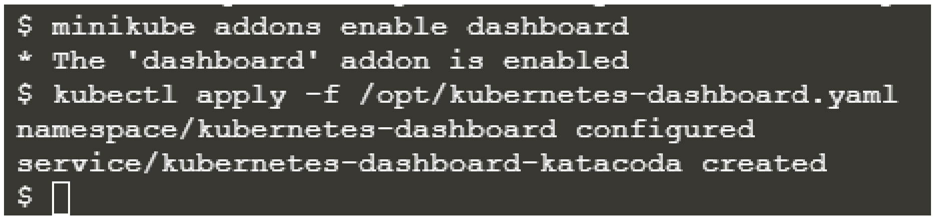

```
$ kubectl expose deployment first-deployment --port=80 --type=NodePort
service/first-deployment exposed
$ export PORT=$(kubectl get svc first-deployment -o go-template='{{range.spec.ports}}{{if .nodePort}}
{{.nodePort}}{{"\n"}}{{end}}{{end}}')
$ echo "Accessing host01:$PORT"
Accessing host01:32348
$ curl host01:$PORT
<h1>This request was processed by host: first-deployment-666c48b44-cw6gv</h1>
$ 
```

Step 4 - Dashboard

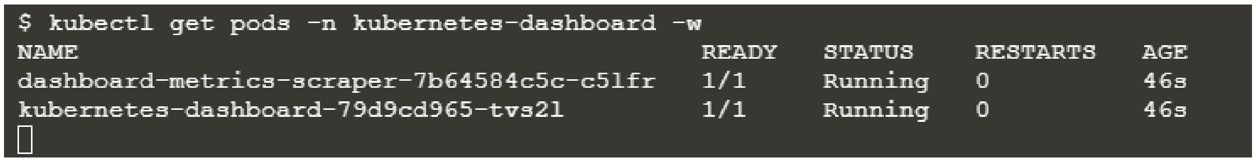
Enable the dashboard using Minikube with the command `minikube addons enable dashboard`

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
```


```
$ minikube addons enable dashboard
* The 'dashboard' addon is enabled
$ kubectl apply -f /opt/kubernetes-dashboard.yaml
namespace/kubernetes-dashboard configured
service/kubernetes-dashboard-katacoda created
$ 
```

To see the progress of the Dashboard starting, watch the Pods within the *kube-system* namespace using `kubectl get pods -n kubernetes-dashboard –w`



```
$ kubectl get pods -n kubernetes-dashboard -w
NAME                                         READY   STATUS    RESTARTS   AGE
dashboard-metrics-scraper-7b64584c5c-c5lfr   1/1     Running   0          46s
kubernetes-dashboard-79d9cd965-tvs2l         1/1     Running   0          46s
```