NAME – SHUBHAM SINHA

COURSE – B.TECH CS CCV

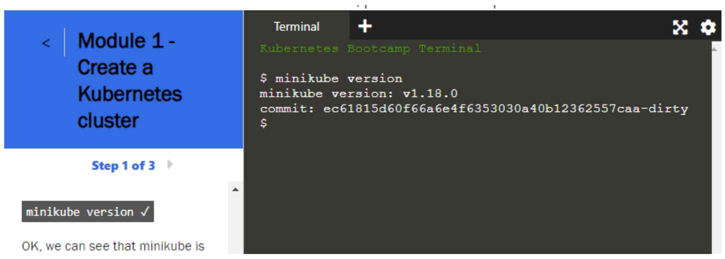
UNIVERSITY ROLL – 201510020

SECTION – O-20

KUBERNETS ASSIGNMENT

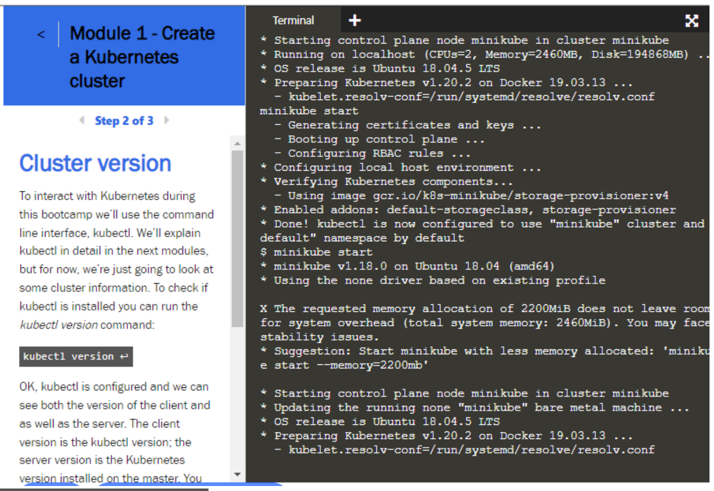
Module 1- Create a Kubernetes cluster

`minikube version`


Module 1 - Create a Kubernetes cluster
Step 1 of 3
minikube version ✓
OK, we can see that minikube is
Terminal

```
Kubernetes Bootcamp Terminal

$ minikube version
minikube version: v1.18.0
commit: ec61815d60f66a6e4f6353030a40b12362557caa-dirty
$
```

1. `minikube start`



Cluster version

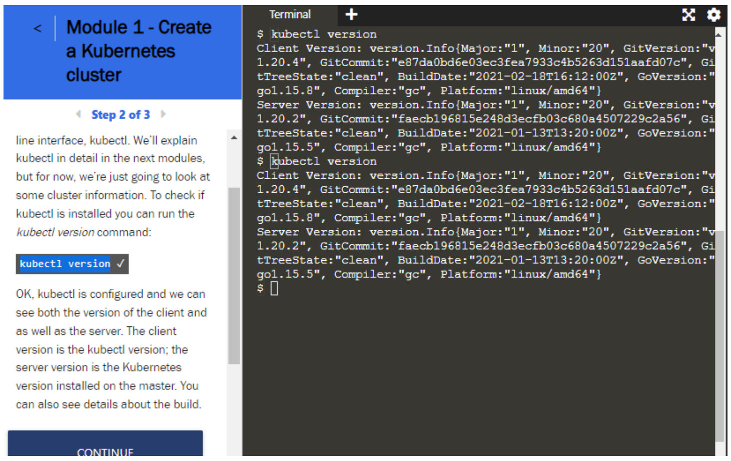
To interact with Kubernetes during this bootcamp we'll use the command line interface, kubectl. We'll explain kubectl in detail in the next modules, but for now, we're just going to look at some cluster information. To check if kubectl is installed you can run the *kubectl version* command:

`kubectl version ↵`

OK, kubectl is configured and we can see both the version of the client and as well as the server. The client version is the kubectl version; the server version is the Kubernetes version installed on the master. You

```
* Starting control plane node minikube in cluster minikube
* Running on localhost (CPUs=2, Memory=2460MB, Disk=194868MB) ..
* OS release is Ubuntu 18.04.5 LTS
* Preparing Kubernetes v1.20.2 on Docker 19.03.13 ...
  - kubelet.resolv-conf=/run/systemd/resolve/resolv.conf
minikube start
  - Generating certificates and keys ...
  - Booting up control plane ...
  - Configuring RBAC rules ...
* Configuring local host environment ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v4
* Enabled addons: default-storageclass, storage-provisioner
* Done! kubectl is now configured to use "minikube" cluster and
default" namespace by default
$ minikube start
* minikube v1.18.0 on Ubuntu 18.04 (amd64)
* Using the none driver based on existing profile

X The requested memory allocation of 2200MiB does not leave room
for system overhead (total system memory: 2460MiB). You may face
stability issues.
* Suggestion: Start minikube with less memory allocated: 'miniku
e start --memory=2200mb'

* Starting control plane node minikube in cluster minikube
* Updating the running none "minikube" bare metal machine ...
* OS release is Ubuntu 18.04.5 LTS
* Preparing Kubernetes v1.20.2 on Docker 19.03.13 ...
  - kubelet.resolv-conf=/run/systemd/resolve/resolv.conf
```

2. `kubectl version`



< Module 1 - Create a Kubernetes cluster

Step 2 of 3

line interface, kubectl. We'll explain kubectl in detail in the next modules, but for now, we're just going to look at some cluster information. To check if kubectl is installed you can run the *kubectl version* command:

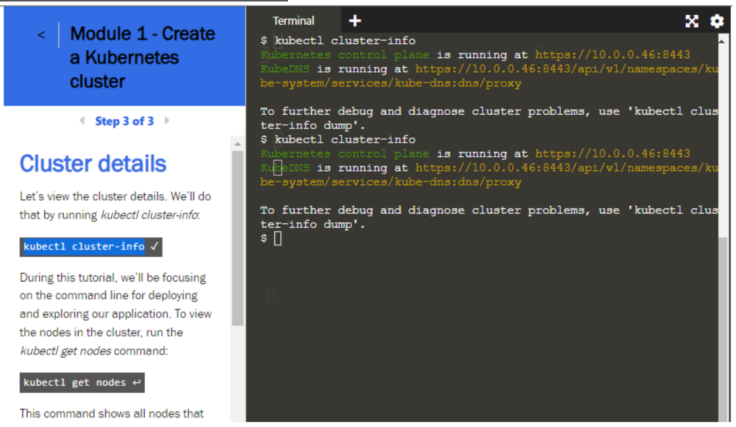
`kubectl version` ✓

OK, kubectl is configured and we can see both the version of the client and as well as the server. The client version is the kubectl version; the server version is the Kubernetes version installed on the master. You can also see details about the build.

CONTINUE

```
$ kubectl version
Client Version: version.Info{Major:"1", Minor:"20", GitVersion:"v
1.20.4", GitCommit:"e87da0bd6e03ec3fea7933c4b5263d151aafd07c", Gi
tTreeState:"clean", BuildDate:"2021-02-18T16:12:00Z", GoVersion:"
go1.15.8", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"20", GitVersion:"v
1.20.2", GitCommit:"faecb196815e248d3ecfb03c680a4507229c2a56", Gi
tTreeState:"clean", BuildDate:"2021-01-13T13:20:00Z", GoVersion:"
go1.15.5", Compiler:"gc", Platform:"linux/amd64"}
$ kubectl version
Client Version: version.Info{Major:"1", Minor:"20", GitVersion:"v
1.20.4", GitCommit:"e87da0bd6e03ec3fea7933c4b5263d151aafd07c", Gi
tTreeState:"clean", BuildDate:"2021-02-18T16:12:00Z", GoVersion:"
go1.15.8", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"20", GitVersion:"v
1.20.2", GitCommit:"faecb196815e248d3ecfb03c680a4507229c2a56", Gi
tTreeState:"clean", BuildDate:"2021-01-13T13:20:00Z", GoVersion:"
go1.15.5", Compiler:"gc", Platform:"linux/amd64"}
$
```

3. `kubectl cluster-info`



< Module 1 - Create a Kubernetes cluster

Step 3 of 3

Cluster details

Let's view the cluster details. We'll do that by running *kubectl cluster-info*:

`kubectl cluster-info` ✓

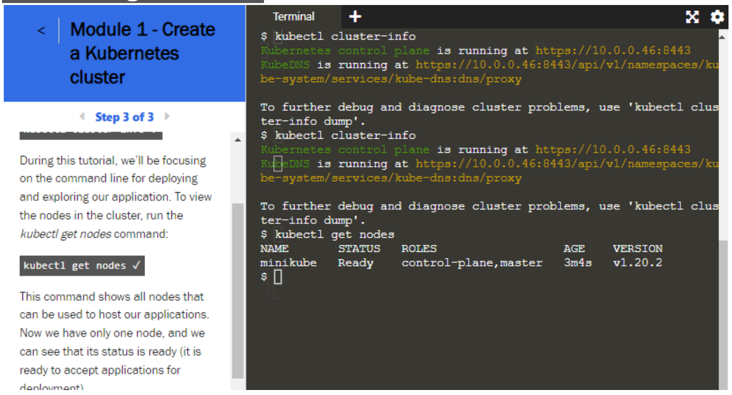
During this tutorial, we'll be focusing on the command line for deploying and exploring our application. To view the nodes in the cluster, run the *kubectl get nodes* command:

`kubectl get nodes` ↵

This command shows all nodes that

```
$ kubectl cluster-info
Kubernetes control plane is running at https://10.0.0.46:8443
KubeDNS is running at https://10.0.0.46:8443/api/v1/namespaces/ku
be-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl clus
ter-info dump'.
$ kubectl cluster-info
Kubernetes control plane is running at https://10.0.0.46:8443
KubeDNS is running at https://10.0.0.46:8443/api/v1/namespaces/ku
be-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl clus
ter-info dump'.
$
```

4. `kubectl get nodes`



< Module 1 - Create a Kubernetes cluster

Step 3 of 3

During this tutorial, we'll be focusing on the command line for deploying and exploring our application. To view the nodes in the cluster, run the *kubectl get nodes* command:

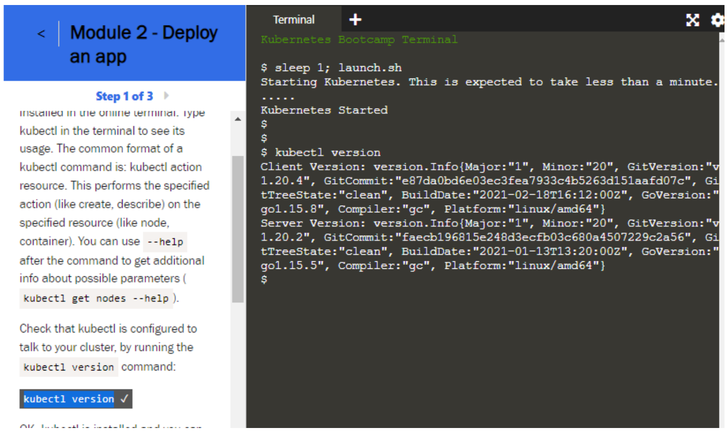
`kubectl get nodes` ✓

This command shows all nodes that can be used to host our applications. Now we have only one node, and we can see that its status is ready (it is ready to accept applications for deployment).

```
$ kubectl cluster-info
Kubernetes control plane is running at https://10.0.0.46:8443
KubeDNS is running at https://10.0.0.46:8443/api/v1/namespaces/ku
be-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl clus
ter-info dump'.
$ kubectl cluster-info
Kubernetes control plane is running at https://10.0.0.46:8443
KubeDNS is running at https://10.0.0.46:8443/api/v1/namespaces/ku
be-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl clus
ter-info dump'.
$ kubectl get nodes
NAME       STATUS   ROLES                  AGE    VERSION
minikube   Ready    control-plane,master   3m4s   v1.20.2
$
```

Module 2 - Deploy an app

1. `kubectl version`



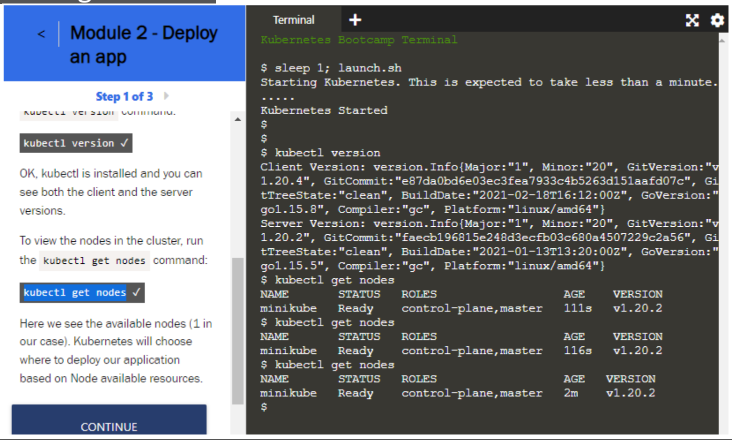
installed in the online terminal. Type kubectl in the terminal to see its usage. The common format of a kubectl command is: kubectl action resource. This performs the specified action (like create, describe) on the specified resource (like node, container). You can use `--help` after the command to get additional info about possible parameters (`kubectl get nodes --help`).

Check that kubectl is configured to talk to your cluster, by running the `kubectl version` command:

`kubectl version` ✓

```
Kubernetes Bootcamp Terminal

$ sleep 1; launch.sh
Starting Kubernetes. This is expected to take less than a minute.
.....
Kubernetes Started
$
$
$ kubectl version
Client Version: version.Info{Major:"1", Minor:"20", GitVersion:"v
1.20.4", GitCommit:"e87da0bd6e03ec3fea7933c4b5263d151aafd07c", Gi
tTreeState:"clean", BuildDate:"2021-02-18T16:12:00Z", GoVersion:"
go1.15.8", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"20", GitVersion:"v
1.20.2", GitCommit:"faecb196815e248d3ecfb03c680a4507229c2a56", Gi
tTreeState:"clean", BuildDate:"2021-01-13T13:20:00Z", GoVersion:"
go1.15.5", Compiler:"gc", Platform:"linux/amd64"}
$
```

2. `kubectl get nodes`



`kubectl version` ✓

OK, kubectl is installed and you can see both the client and the server versions.

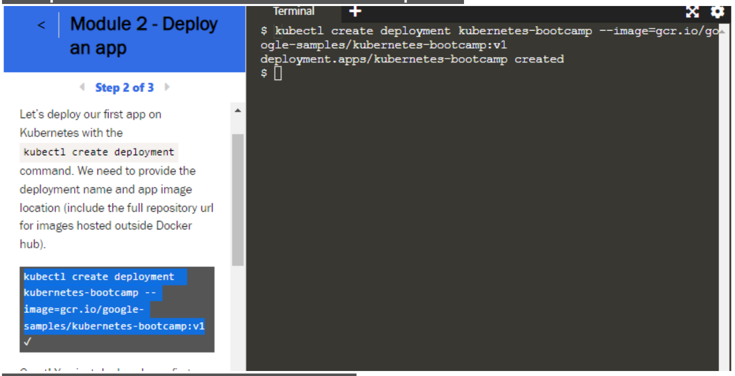
To view the nodes in the cluster, run the `kubectl get nodes` command:

`kubectl get nodes` ✓

Here we see the available nodes (1 in our case). Kubernetes will choose where to deploy our application based on Node available resources.

```
Kubernetes Bootcamp Terminal

$ sleep 1; launch.sh
Starting Kubernetes. This is expected to take less than a minute.
.....
Kubernetes Started
$
$
$ kubectl version
Client Version: version.Info{Major:"1", Minor:"20", GitVersion:"v
1.20.4", GitCommit:"e87da0bd6e03ec3fea7933c4b5263d151aafd07c", Gi
tTreeState:"clean", BuildDate:"2021-02-18T16:12:00Z", GoVersion:"
go1.15.8", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"20", GitVersion:"v
1.20.2", GitCommit:"faecb196815e248d3ecfb03c680a4507229c2a56", Gi
tTreeState:"clean", BuildDate:"2021-01-13T13:20:00Z", GoVersion:"
go1.15.5", Compiler:"gc", Platform:"linux/amd64"}
$ kubectl get nodes
NAME       STATUS   ROLES                  AGE    VERSION
minikube   Ready    control-plane,master   111s   v1.20.2
$ kubectl get nodes
NAME       STATUS   ROLES                  AGE    VERSION
minikube   Ready    control-plane,master   116s   v1.20.2
$ kubectl get nodes
NAME       STATUS   ROLES                  AGE   VERSION
minikube   Ready    control-plane,master   2m    v1.20.2
$
```

3. `kubectl create deployment kubernetes-bootcamp --image=gcr.io/google-samples/kubernetes-bootcamp:v1`

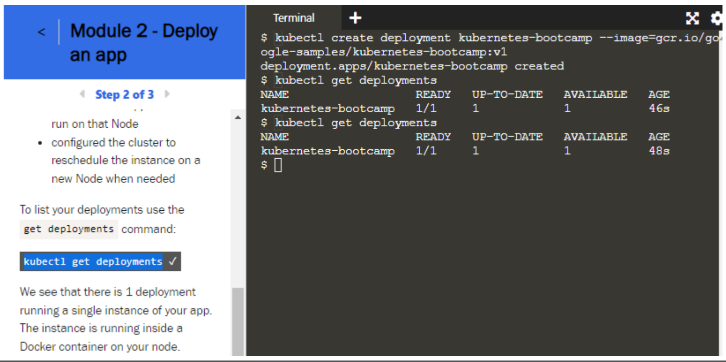


Let's deploy our first app on Kubernetes with the `kubectl create deployment` command. We need to provide the deployment name and app image location (include the full repository url for images hosted outside Docker hub).

```
kubectl create deployment
kubernetes-bootcamp --
image=gcr.io/google-
samples/kubernetes-bootcamp:v1
```
✓

```
$ kubectl create deployment kubernetes-bootcamp --image=gcr.io/go
ogle-samples/kubernetes-bootcamp:v1
deployment.apps/kubernetes-bootcamp created
$ 
```

4. `kubectl get deployments`



run on that Node

- configured the cluster to reschedule the instance on a new Node when needed

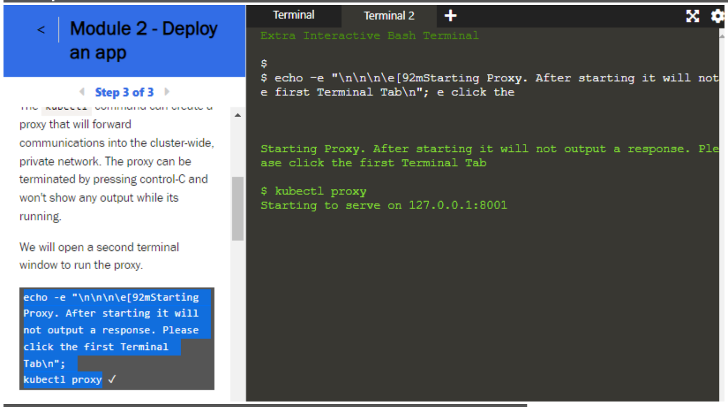
To list your deployments use the `get deployments` command:

`kubectl get deployments` ✓

We see that there is 1 deployment running a single instance of your app. The instance is running inside a Docker container on your node.

```
$ kubectl create deployment kubernetes-bootcamp --image=gcr.io/go
ogle-samples/kubernetes-bootcamp:v1
deployment.apps/kubernetes-bootcamp created
$ kubectl get deployments
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp   1/1     1            1           46s
$ kubectl get deployments
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp   1/1     1            1           48s
$ 
```

5. `echo -e "\n\n\n\e[92mStarting Proxy. After starting it will not output a response. Please click the first Terminal Tab\n"; kubectl proxy`



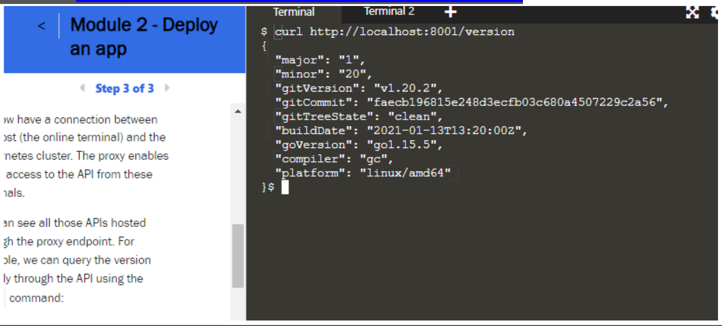
proxy that will forward communications into the cluster-wide, private network. The proxy can be terminated by pressing control-C and won't show any output while its running.

We will open a second terminal window to run the proxy.

```
echo -e "\n\n\n\e[92mStarting
Proxy. After starting it will
not output a response. Please
click the first Terminal
Tab\n";
kubectl proxy ✓
```

```
Extra Interactive Bash Terminal

$
$ echo -e "\n\n\n\e[92mStarting Proxy. After starting it will not
e first Terminal Tab\n"; e click the



Starting Proxy. After starting it will not output a response. Ple
ase click the first Terminal Tab

$ kubectl proxy
Starting to serve on 127.0.0.1:8001
```

6. `curl http://localhost:8001/version`

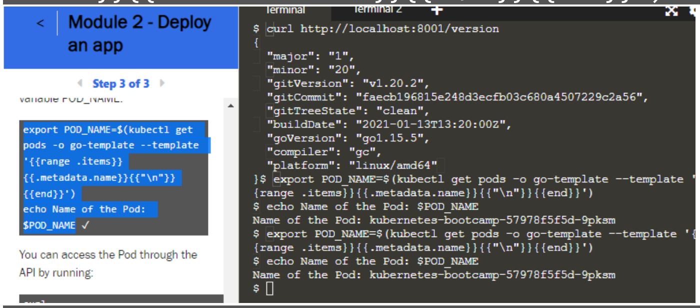


ɔw have a connection between ɔst (the online terminal) and the ɪnetes cluster. The proxy enables access to the API from these ıals.

ɜn see all those APIs hosted ʒh the proxy endpoint. For ɔle, we can query the version ly through the API using the command:

```
$ curl http://localhost:8001/version
{
  "major": "1",
  "minor": "20",
  "gitVersion": "v1.20.2",
  "gitCommit": "faecb196815e248d3ecfb03c680a4507229c2a56",
  "gitTreeState": "clean",
  "buildDate": "2021-01-13T13:20:00Z",
  "goVersion": "go1.15.5",
  "compiler": "gc",
  "platform": "linux/amd64"
}$ 
```

7. `export POD_NAME=$(kubectl get pods -o go-template --template '{{range .items}}{{.metadata.name}}{{"\n"}}{{end}}') echo Name of the Pod: $POD_NAME`

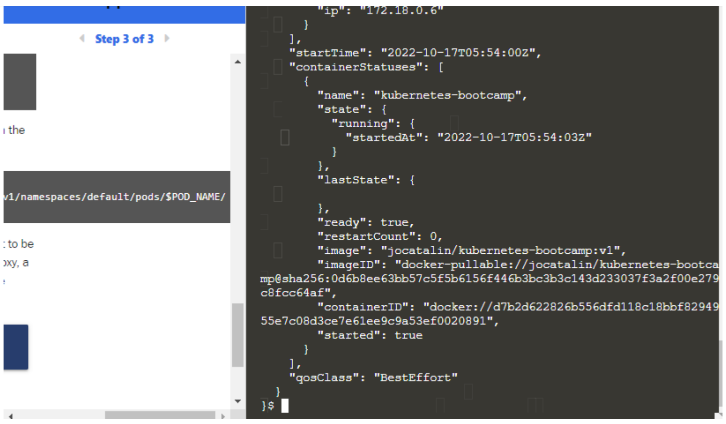


```
export POD_NAME=$(kubectl get
pods -o go-template --template
'{{range .items}}
{{.metadata.name}}{{"\n"}}
{{end}}')
echo Name of the Pod:
$POD_NAME ✓
```

You can access the Pod through the API by running:

```
$ curl http://localhost:8001/version
{
  "major": "1",
  "minor": "20",
  "gitVersion": "v1.20.2",
  "gitCommit": "faecb196815e248d3ecfb03c680a4507229c2a56",
  "gitTreeState": "clean",
  "buildDate": "2021-01-13T13:20:00Z",
  "goVersion": "go1.15.5",
  "compiler": "gc",
  "platform": "linux/amd64"
}$ export POD_NAME=$(kubectl get pods -o go-template --template '
{range .items}}{{.metadata.name}}{{"\n"}}{{end}}')
$ echo Name of the Pod: $POD_NAME
Name of the Pod: kubernetes-bootcamp-57978f5f5d-9pksm
$ export POD_NAME=$(kubectl get pods -o go-template --template '{
{range .items}}{{.metadata.name}}{{"\n"}}{{end}}')
$ echo Name of the Pod: $POD_NAME
Name of the Pod: kubernetes-bootcamp-57978f5f5d-9pksm
$ 
```

8. `curl http://localhost:8001/api/v1/namespaces/default/pods/$POD_NAME/`


Step 3 of 3

the

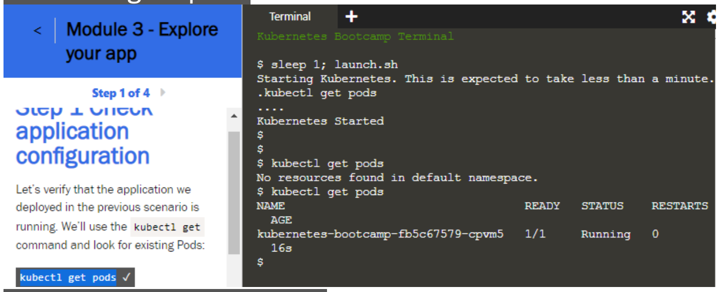
```
v1/namespaces/default/pods/$POD_NAME/
```

to be
xy, a

```
        "ip": "172.18.0.6"
      }
    ],
    "startTime": "2022-10-17T05:54:00Z",
    "containerStatuses": [
      {
        "name": "kubernetes-bootcamp",
        "state": {
          "running": {
            "startedAt": "2022-10-17T05:54:03Z"
          }
        },
        "lastState": {

        },
        "ready": true,
        "restartCount": 0,
        "image": "jocatalin/kubernetes-bootcamp:v1",
        "imageID": "docker-pullable://jocatalin/kubernetes-bootca
mp@sha256:0d6b8ee63bb57c5f5b6156f446b3bc3b3c143d233037f3a2f00e279
c8fcc64af",
        "containerID": "docker://d7b2d622826b556dfd118c18bbf82949
55e7c08d3ce7e61ee9c9a53ef0020891",
        "started": true
      }
    ],
    "qosClass": "BestEffort"
  }
}$
```

Module 3 - Explore your app

1. `kubectl get pods`


Module 3 - Explore your app

Step 1 of 4

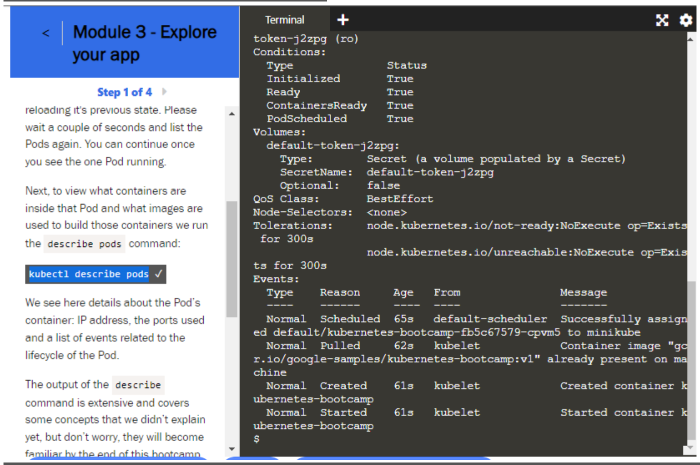
Step 1 Check application configuration

Let's verify that the application we deployed in the previous scenario is running. We'll use the `kubectl get` command and look for existing Pods:

`kubectl get pods` ✓

```
Kubernetes Bootcamp Terminal

$ sleep 1; launch.sh
Starting Kubernetes. This is expected to take less than a minute.
.kubectl get pods
....
Kubernetes Started
$
$
$ kubectl get pods
No resources found in default namespace.
$ kubectl get pods
NAME                                   READY   STATUS    RESTARTS
  AGE
kubernetes-bootcamp-fb5c67579-cpvm5    1/1     Running   0
  16s
$
```

2. `kubectl describe pods`


Module 3 - Explore your app

Step 1 of 4

reloading it's previous state. Please wait a couple of seconds and list the Pods again. You can continue once you see the one Pod running.

Next, to view what containers are inside that Pod and what images are used to build those containers we run the `describe pods` command:

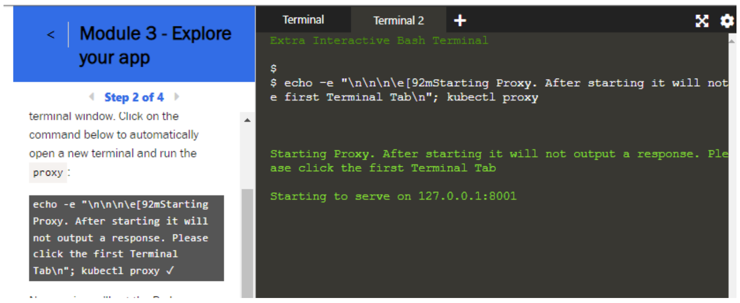
`kubectl describe pods` ✓

We see here details about the Pod's container: IP address, the ports used and a list of events related to the lifecycle of the Pod.

The output of the `describe` command is extensive and covers some concepts that we didn't explain yet, but don't worry, they will become familiar by the end of this bootcamp.

```
token-j2zpg (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  default-token-j2zpg:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-j2zpg
    Optional:    false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute op=Exists
 for 300s
                 node.kubernetes.io/unreachable:NoExecute op=Exis
ts for 300s
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  65s   default-scheduler  Successfully assign
ed default/kubernetes-bootcamp-fb5c67579-cpvm5 to minikube
  Normal  Pulled     62s   kubelet            Container image "gc
r.io/google-samples/kubernetes-bootcamp:v1" already present on ma
chine
  Normal  Created    61s   kubelet            Created container k
ubernetes-bootcamp
  Normal  Started    61s   kubelet            Started container k
ubernetes-bootcamp
$
```

3. `echo -e "\n\n\n\e[92mStarting Proxy. After starting it will not output a response. Please click the first Terminal Tab\n"; kubectl proxy`


Module 3 - Explore your app

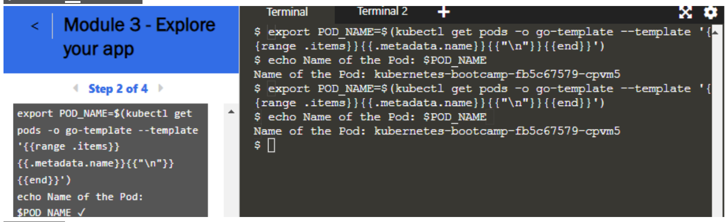
Step 2 of 4

terminal window. Click on the command below to automatically open a new terminal and run the `proxy`:

```
echo -e "\n\n\n\e[92mStarting Proxy. After starting it will not output a response. Please click the first Terminal Tab\n"; kubectl proxy ✓
```

```
Extra Interactive Bash Terminal

$
$ echo -e "\n\n\n\e[92mStarting Proxy. After starting it will not
e first Terminal Tab\n"; kubectl proxy



Starting Proxy. After starting it will not output a response. Ple
ase click the first Terminal Tab

Starting to serve on 127.0.0.1:8001
```

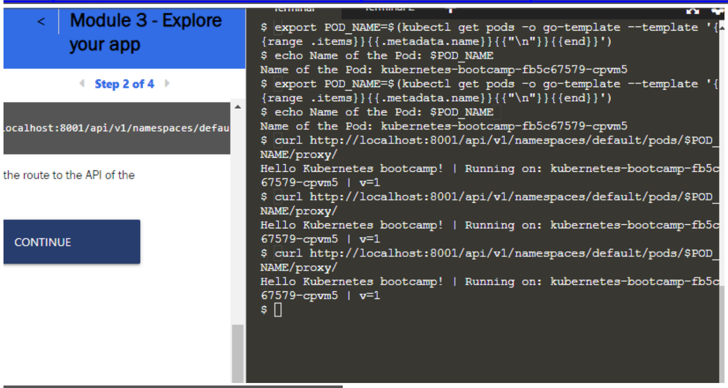
4. `export POD_NAME=$(kubectl get pods -o go-template --template '{{range .items}}{{.metadata.name}}{{"\n"}}{{end}}') echo Name of the Pod: $POD_NAME`



```
export POD_NAME=$(kubectl get
pods -o go-template --template
'{{range .items}}
{{.metadata.name}}{{"\n"}}
{{end}}')
echo Name of the Pod:
$POD_NAME ✓
```

```
$ export POD_NAME=$(kubectl get pods -o go-template --template '{
{range .items}}{{.metadata.name}}{{"\n"}}{{end}}')
$ echo Name of the Pod: $POD_NAME
Name of the Pod: kubernetes-bootcamp-fb5c67579-cpvm5
$ export POD_NAME=$(kubectl get pods -o go-template --template '{
{range .items}}{{.metadata.name}}{{"\n"}}{{end}}')
$ echo Name of the Pod: $POD_NAME
Name of the Pod: kubernetes-bootcamp-fb5c67579-cpvm5
$ 
```

5. `curl` http://localhost:8001/api/v1/namespaces/default/pods/$POD_NAME/proxy/

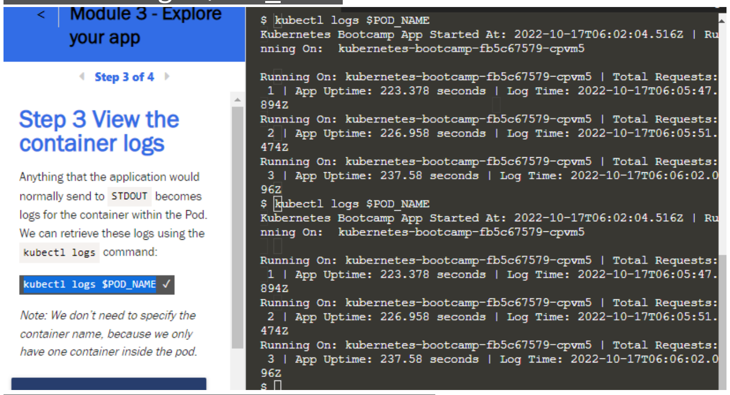


```
ocalhost:8001/api/v1/namespaces/defau
```

the route to the API of the

```
$ export POD_NAME=$(kubectl get pods -o go-template --template '{
{range .items}}{{.metadata.name}}{{"\n"}}{{end}}')
$ echo Name of the Pod: $POD_NAME
Name of the Pod: kubernetes-bootcamp-fb5c67579-cpvm5
$ export POD_NAME=$(kubectl get pods -o go-template --template '{
{range .items}}{{.metadata.name}}{{"\n"}}{{end}}')
$ echo Name of the Pod: $POD_NAME
Name of the Pod: kubernetes-bootcamp-fb5c67579-cpvm5
$ curl http://localhost:8001/api/v1/namespaces/default/pods/$POD_
NAME/proxy/
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-fb5c
67579-cpvm5 | v=1
$ curl http://localhost:8001/api/v1/namespaces/default/pods/$POD_
NAME/proxy/
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-fb5c
67579-cpvm5 | v=1
$ curl http://localhost:8001/api/v1/namespaces/default/pods/$POD_
NAME/proxy/
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-fb5c
67579-cpvm5 | v=1
$ 
```

6. `kubectl logs $POD_NAME`



Step 3 View the container logs

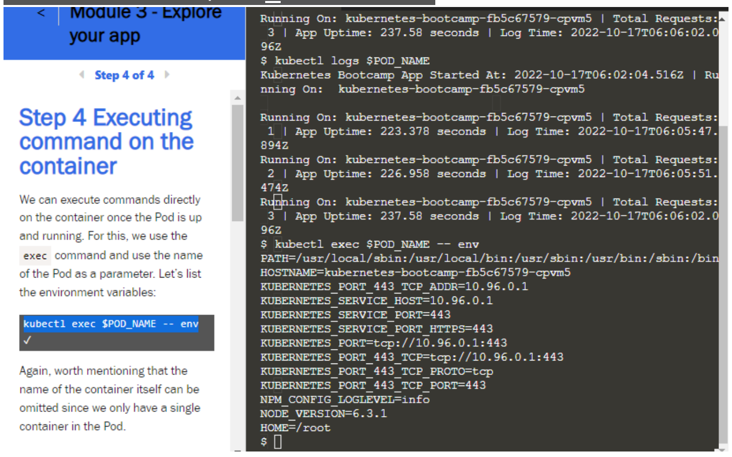
Anything that the application would normally send to `STDOUT` becomes logs for the container within the Pod. We can retrieve these logs using the `kubectl logs` command:

`kubectl logs $POD_NAME` ✓

Note: We don't need to specify the container name, because we only have one container inside the pod.

```
$ kubectl logs $POD_NAME
Kubernetes Bootcamp App Started At: 2022-10-17T06:02:04.516Z | Ru
nning On:  kubernetes-bootcamp-fb5c67579-cpvm5

Running On: kubernetes-bootcamp-fb5c67579-cpvm5 | Total Requests:
 1 | App Uptime: 223.378 seconds | Log Time: 2022-10-17T06:05:47.
894Z
Running On: kubernetes-bootcamp-fb5c67579-cpvm5 | Total Requests:
 2 | App Uptime: 226.958 seconds | Log Time: 2022-10-17T06:05:51.
474Z
Running On: kubernetes-bootcamp-fb5c67579-cpvm5 | Total Requests:
 3 | App Uptime: 237.58 seconds | Log Time: 2022-10-17T06:06:02.0
96Z
$ kubectl logs $POD_NAME
Kubernetes Bootcamp App Started At: 2022-10-17T06:02:04.516Z | Ru
nning On:  kubernetes-bootcamp-fb5c67579-cpvm5

Running On: kubernetes-bootcamp-fb5c67579-cpvm5 | Total Requests:
 1 | App Uptime: 223.378 seconds | Log Time: 2022-10-17T06:05:47.
894Z
Running On: kubernetes-bootcamp-fb5c67579-cpvm5 | Total Requests:
 2 | App Uptime: 226.958 seconds | Log Time: 2022-10-17T06:05:51.
474Z
Running On: kubernetes-bootcamp-fb5c67579-cpvm5 | Total Requests:
 3 | App Uptime: 237.58 seconds | Log Time: 2022-10-17T06:06:02.0
96Z
$ 
```

7. `kubectl exec $POD_NAME – env`



Step 4 Executing command on the container

We can execute commands directly on the container once the Pod is up and running. For this, we use the `exec` command and use the name of the Pod as a parameter. Let's list the environment variables:

`kubectl exec $POD_NAME -- env` ✓

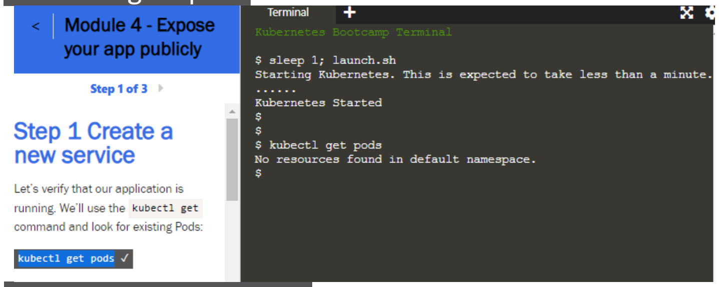
Again, worth mentioning that the name of the container itself can be omitted since we only have a single container in the Pod.

```
Running On: kubernetes-bootcamp-fb5c67579-cpvm5 | Total Requests:
 3 | App Uptime: 237.58 seconds | Log Time: 2022-10-17T06:06:02.0
96Z
$ kubectl logs $POD_NAME
Kubernetes Bootcamp App Started At: 2022-10-17T06:02:04.516Z | Ru
nning On:  kubernetes-bootcamp-fb5c67579-cpvm5

Running On: kubernetes-bootcamp-fb5c67579-cpvm5 | Total Requests:
 1 | App Uptime: 223.378 seconds | Log Time: 2022-10-17T06:05:47.
894Z
Running On: kubernetes-bootcamp-fb5c67579-cpvm5 | Total Requests:
 2 | App Uptime: 226.958 seconds | Log Time: 2022-10-17T06:05:51.
474Z
Running On: kubernetes-bootcamp-fb5c67579-cpvm5 | Total Requests:
 3 | App Uptime: 237.58 seconds | Log Time: 2022-10-17T06:06:02.0
96Z
$ kubectl exec $POD_NAME -- env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=kubernetes-bootcamp-fb5c67579-cpvm5
KUBERNETES_PORT_443_TCP_ADDR=10.96.0.1
KUBERNETES_SERVICE_HOST=10.96.0.1
KUBERNETES_SERVICE_PORT=443
KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_PORT=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP_PROTO=tcp
KUBERNETES_PORT_443_TCP_PORT=443
NPM_CONFIG_LOGLEVEL=info
NODE_VERSION=6.3.1
HOME=/root
$ 
```

<

Module 4 - Expose your app publicly

1. `kubectl get pods`



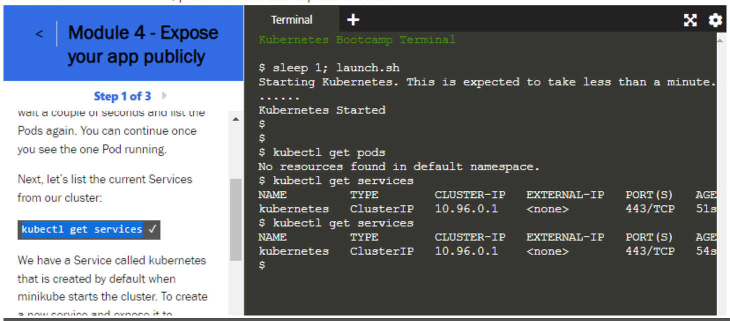
Step 1 Create a new service

Let's verify that our application is running. We'll use the `kubectl get` command and look for existing Pods:

`kubectl get pods` ✓

```
Kubernetes Bootcamp Terminal

$ sleep 1; launch.sh
Starting Kubernetes. This is expected to take less than a minute.
......
Kubernetes Started
$
$
$ kubectl get pods
No resources found in default namespace.
$
```

2. `kubectl get services`



wait a couple of seconds and list the Pods again. You can continue once you see the one Pod running.

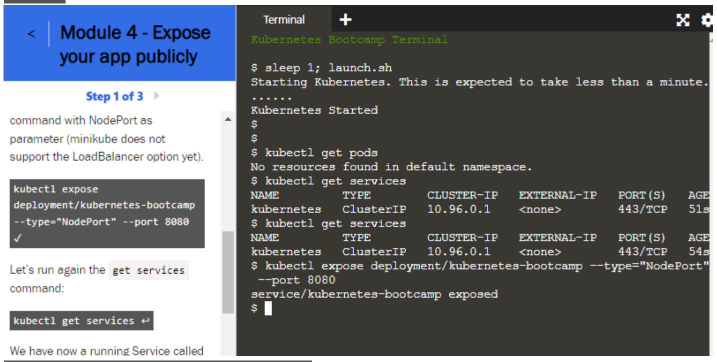
Next, let's list the current Services from our cluster:

`kubectl get services` ✓

We have a Service called kubernetes that is created by default when minikube starts the cluster. To create a new service and expose it to

```
Kubernetes Bootcamp Terminal

$ sleep 1; launch.sh
Starting Kubernetes. This is expected to take less than a minute.
......
Kubernetes Started
$
$
$ kubectl get pods
No resources found in default namespace.
$ kubectl get services
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   51s
$ kubectl get services
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   54s
$
```

3. `kubectl expose deployment/kubernetes-bootcamp --type="NodePort" --port 8080`



command with NodePort as parameter (minikube does not support the LoadBalancer option yet).

```
kubectl expose
deployment/kubernetes-bootcamp
--type="NodePort" --port 8080
✓
```

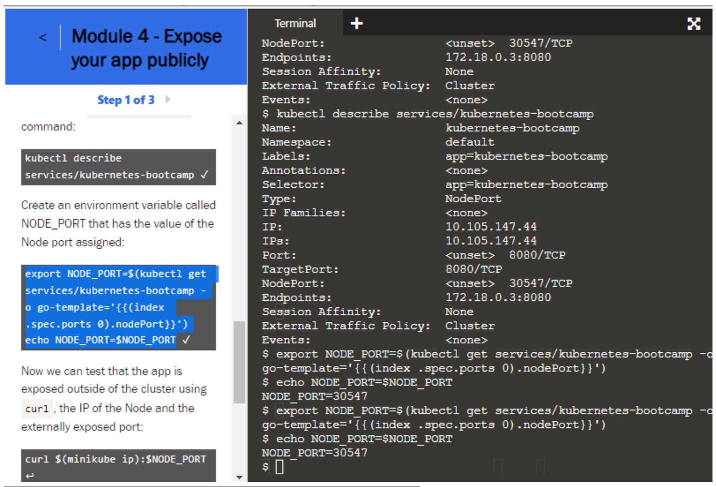
Let's run again the `get services` command:

`kubectl get services` ↩

We have now a running Service called

```
Kubernetes Bootcamp Terminal

$ sleep 1; launch.sh
Starting Kubernetes. This is expected to take less than a minute.
......
Kubernetes Started
$
$
$ kubectl get pods
No resources found in default namespace.
$ kubectl get services
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   51s
$ kubectl get services
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   54s
$ kubectl expose deployment/kubernetes-bootcamp --type="NodePort"
 --port 8080
service/kubernetes-bootcamp exposed
$ █
```

4. `kubectl get services`
5. `kubectl describe services/kubernetes-bootcamp`
6. `export NODE_PORT=$(kubectl get services/kubernetes-bootcamp -o go-template='{{(index .spec.ports 0).nodePort}}') echo NODE_PORT=$NODE_PORT`



command:

```
kubectl describe services/kubernetes-bootcamp ✓
```

Create an environment variable called NODE_PORT that has the value of the Node port assigned:

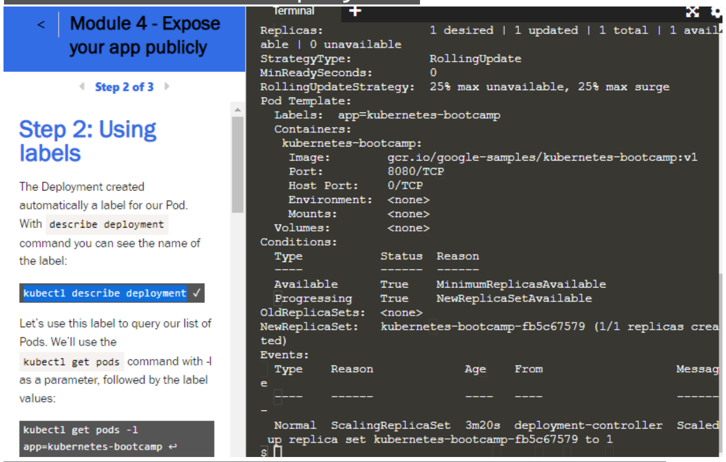
```
export NODE_PORT=$(kubectl get services/kubernetes-bootcamp -o go-template='{{(index .spec.ports 0).nodePort}}')
echo NODE_PORT=$NODE_PORT ✓
```

Now we can test that the app is exposed outside of the cluster using `curl`, the IP of the Node and the externally exposed port:

```
curl $(minikube ip):$NODE_PORT ↵
```

```
NodePort:                 <unset>  30547/TCP
Endpoints:                172.18.0.3:8080
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
$ kubectl describe services/kubernetes-bootcamp
Name:                     kubernetes-bootcamp
Namespace:                default
Labels:                   app=kubernetes-bootcamp
Annotations:              <none>
Selector:                 app=kubernetes-bootcamp
Type:                     NodePort
IP Families:              <none>
IP:                       10.105.147.44
IPs:                      10.105.147.44
Port:                     <unset>  8080/TCP
TargetPort:               8080/TCP
NodePort:                 <unset>  30547/TCP
Endpoints:                172.18.0.3:8080
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
$ export NODE_PORT=$(kubectl get services/kubernetes-bootcamp -o
go-template='{{(index .spec.ports 0).nodePort}}')
$ echo NODE_PORT=$NODE_PORT
NODE_PORT=30547
$ export NODE_PORT=$(kubectl get services/kubernetes-bootcamp -o
go-template='{{(index .spec.ports 0).nodePort}}')
$ echo NODE_PORT=$NODE_PORT
NODE_PORT=30547
$
```

7. `kubectl describe deployment`



Step 2: Using labels

The Deployment created automatically a label for our Pod. With `describe deployment` command you can see the name of the label:

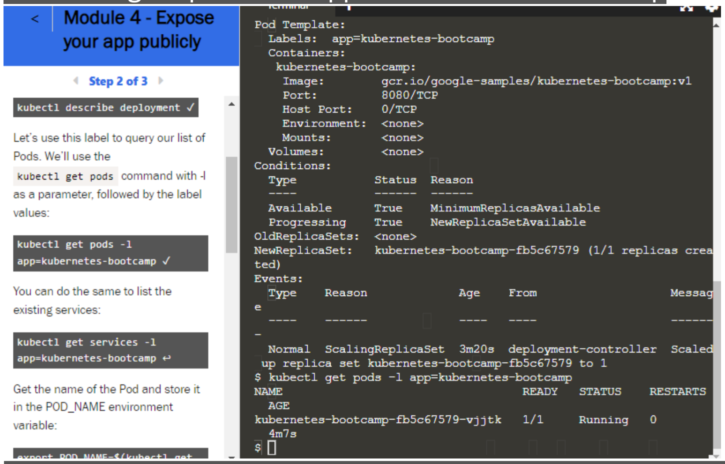
```
kubectl describe deployment ✓
```

Let's use this label to query our list of Pods. We'll use the `kubectl get pods` command with -l as a parameter, followed by the label values:

```
kubectl get pods -l app=kubernetes-bootcamp ↵
```

```
Replicas:               1 desired | 1 updated | 1 total | 1 avail
able | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=kubernetes-bootcamp
  Containers:
   kubernetes-bootcamp:
    Image:        gcr.io/google-samples/kubernetes-bootcamp:v1
    Port:         8080/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   kubernetes-bootcamp-fb5c67579 (1/1 replicas crea
ted)
Events:
  Type    Reason             Age    From                   Messag
e
  ----    ------             ----   ----                   ------
-
  Normal  ScalingReplicaSet  3m20s  deployment-controller  Scaled
 up replica set kubernetes-bootcamp-fb5c67579 to 1
$
```

8. `kubectl get pods -l app=kubernetes-bootcamp`



```
kubectl describe deployment ✓
```

Let's use this label to query our list of Pods. We'll use the `kubectl get pods` command with -l as a parameter, followed by the label values:

```
kubectl get pods -l app=kubernetes-bootcamp ✓
```

You can do the same to list the existing services:

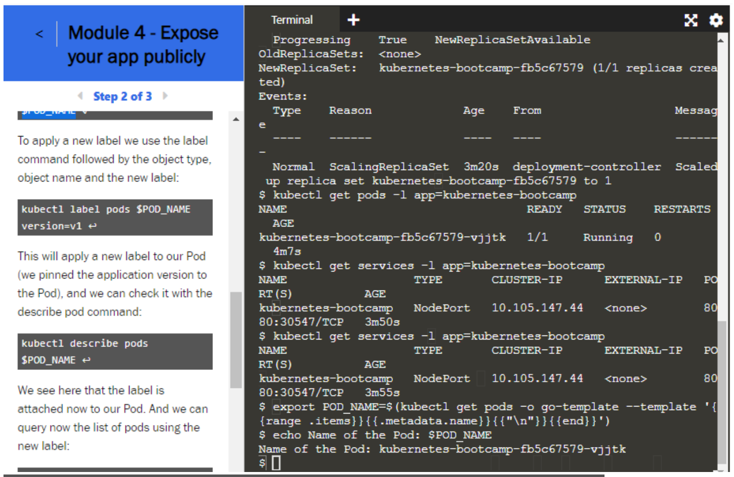
```
kubectl get services -l app=kubernetes-bootcamp ↵
```

Get the name of the Pod and store it in the POD_NAME environment variable:

```
export POD_NAME=$(kubectl get
```

```
Pod Template:
  Labels:  app=kubernetes-bootcamp
  Containers:
   kubernetes-bootcamp:
    Image:        gcr.io/google-samples/kubernetes-bootcamp:v1
    Port:         8080/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   kubernetes-bootcamp-fb5c67579 (1/1 replicas crea
ted)
Events:
  Type    Reason             Age    From                   Messag
e
  ----    ------             ----   ----                   ------
-
  Normal  ScalingReplicaSet  3m20s  deployment-controller  Scaled
 up replica set kubernetes-bootcamp-fb5c67579 to 1
$ kubectl get pods -l app=kubernetes-bootcamp
NAME                                   READY   STATUS    RESTARTS
  AGE
kubernetes-bootcamp-fb5c67579-vjjtk    1/1     Running   0
  4m7s
$
```

9. `kubectl get services -l app=kubernetes-bootcamp`
10. `export POD_NAME=$(kubectl get pods -o go-template --template '{{range .items}}{{.metadata.name}}{{"\n"}}{{end}}') echo Name of the Pod: $POD_NAME`



To apply a new label we use the label command followed by the object type, object name and the new label:

```
kubectl label pods $POD_NAME version=v1 ↩
```

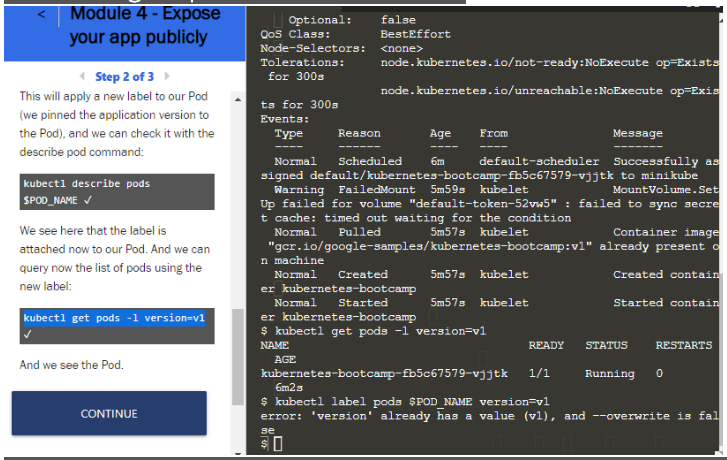
This will apply a new label to our Pod (we pinned the application version to the Pod), and we can check it with the describe pod command:

```
kubectl describe pods $POD_NAME ↩
```

We see here that the label is attached now to our Pod. And we can query now the list of pods using the new label:

```
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   kubernetes-bootcamp-fb5c67579 (1/1 replicas crea
ted)
Events:
  Type    Reason             Age    From                   Messag
e
  ----    ------             ----   ----                   ------
-
  Normal  ScalingReplicaSet  3m20s  deployment-controller  Scaled
 up replica set kubernetes-bootcamp-fb5c67579 to 1
$ kubectl get pods -l app=kubernetes-bootcamp
NAME                                  READY   STATUS    RESTARTS
  AGE
kubernetes-bootcamp-fb5c67579-vjjtk   1/1     Running   0
  4m7s
$ kubectl get services -l app=kubernetes-bootcamp
NAME                   TYPE       CLUSTER-IP      EXTERNAL-IP   PO
RT(S)          AGE
kubernetes-bootcamp    NodePort   10.105.147.44   <none>        80
80:30547/TCP   3m50s
$ kubectl get services -l app=kubernetes-bootcamp
NAME                   TYPE       CLUSTER-IP      EXTERNAL-IP   PO
RT(S)          AGE
kubernetes-bootcamp    NodePort   10.105.147.44   <none>        80
80:30547/TCP   3m55s
$ export POD_NAME=$(kubectl get pods -o go-template --template '{
{range .items}}{{.metadata.name}}{{"\n"}}{{end}}')
$ echo Name of the Pod: $POD_NAME
Name of the Pod: kubernetes-bootcamp-fb5c67579-vjjtk
$
```

11. `kubectl label pods $POD_NAME version=v1`
12. `kubectl describe pods $POD_NAME`
13. `kubectl get pods -l version=v1`



This will apply a new label to our Pod (we pinned the application version to the Pod), and we can check it with the describe pod command:

```
kubectl describe pods $POD_NAME ✓
```

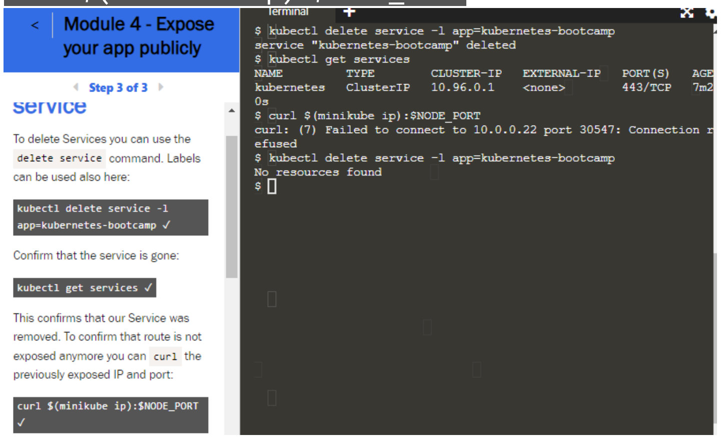
We see here that the label is attached now to our Pod. And we can query now the list of pods using the new label:

```
kubectl get pods -l version=v1 ✓
```

And we see the Pod.

```
  Optional:    false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute op=Exists
 for 300s
                 node.kubernetes.io/unreachable:NoExecute op=Exis
ts for 300s
Events:
  Type     Reason       Age    From                Message
  ----     ------       ----   ----                -------
  Normal   Scheduled    6m     default-scheduler   Successfully as
signed default/kubernetes-bootcamp-fb5c67579-vjjtk to minikube
  Warning  FailedMount  5m59s  kubelet             MountVolume.Set
Up failed for volume "default-token-52vw5" : failed to sync secre
t cache: timed out waiting for the condition
  Normal   Pulled       5m57s  kubelet             Container image
 "gcr.io/google-samples/kubernetes-bootcamp:v1" already present o
n machine
  Normal   Created      5m57s  kubelet             Created contain
er kubernetes-bootcamp
  Normal   Started      5m57s  kubelet             Started contain
er kubernetes-bootcamp
$ kubectl get pods -l version=v1
NAME                                  READY   STATUS    RESTARTS
  AGE
kubernetes-bootcamp-fb5c67579-vjjtk   1/1     Running   0
  6m2s
$ kubectl label pods $POD_NAME version=v1
error: 'version' already has a value (v1), and --overwrite is fal
se
$
```

14. `kubectl delete service -l app=kubernetes-bootcamp`
15. `kubectl get services`
16. `curl $(minikube ip):$NODE_PORT`



service

To delete Services you can use the `delete service` command. Labels can be used also here:

```
kubectl delete service -l app=kubernetes-bootcamp ✓
```

Confirm that the service is gone:

```
kubectl get services ✓
```

This confirms that our Service was removed. To confirm that route is not exposed anymore you can `curl` the previously exposed IP and port:

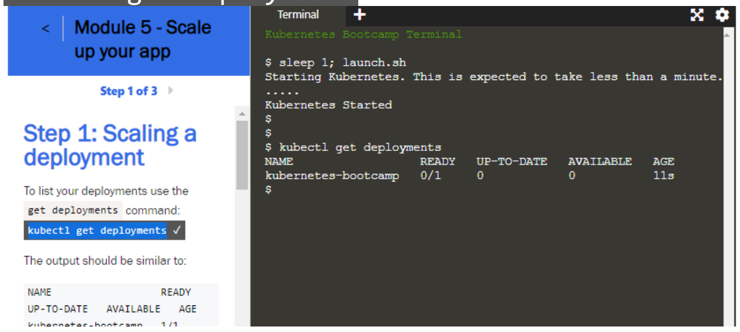
```
curl $(minikube ip):$NODE_PORT ✓
```

```
$ kubectl delete service -l app=kubernetes-bootcamp
service "kubernetes-bootcamp" deleted
$ kubectl get services
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   7m2
0s
$ curl $(minikube ip):$NODE_PORT
curl: (7) Failed to connect to 10.0.0.22 port 30547: Connection r
efused
$ kubectl delete service -l app=kubernetes-bootcamp
No resources found
$
```

<

Module 5 - Scale up your app

1. `kubectl get deployments`



Step 1: Scaling a deployment

To list your deployments use the `get deployments` command:

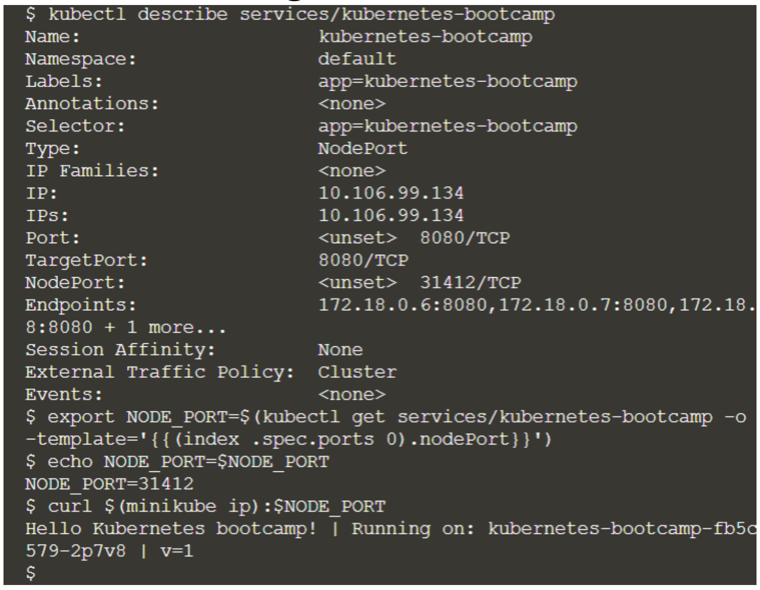
`kubectl get deployments` ✓

The output should be similar to:

```
NAME                            READY
UP-TO-DATE   AVAILABLE   AGE
```

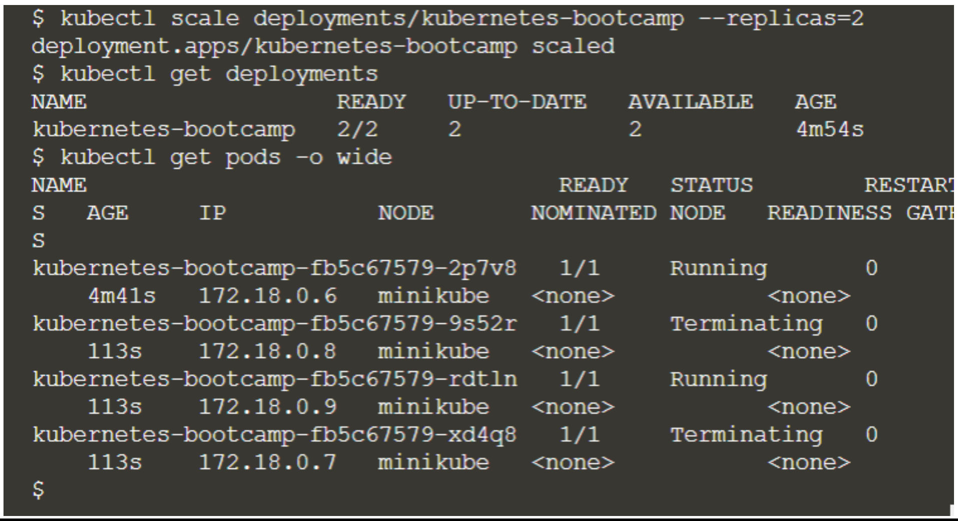
```
Kubernetes Bootcamp Terminal

$ sleep 1; launch.sh
Starting Kubernetes. This is expected to take less than a minute.
.....
Kubernetes Started
$
$
$ kubectl get deployments
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp   0/1     0            0           11s
$
```

2: Load balancing



```
$ kubectl describe services/kubernetes-bootcamp
Name:                     kubernetes-bootcamp
Namespace:                default
Labels:                   app=kubernetes-bootcamp
Annotations:              <none>
Selector:                 app=kubernetes-bootcamp
Type:                     NodePort
IP Families:              <none>
IP:                       10.106.99.134
IPs:                      10.106.99.134
Port:                     <unset>  8080/TCP
TargetPort:               8080/TCP
NodePort:                 <unset>  31412/TCP
Endpoints:                172.18.0.6:8080,172.18.0.7:8080,172.18.
8:8080 + 1 more...
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
$ export NODE_PORT=$(kubectl get services/kubernetes-bootcamp -o
-template='{{(index .spec.ports 0).nodePort}}')
$ echo NODE_PORT=$NODE_PORT
NODE_PORT=31412
$ curl $(minikube ip):$NODE_PORT
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-fb5c
579-2p7v8 | v=1
$
```

3: Scale down



```
$ kubectl scale deployments/kubernetes-bootcamp --replicas=2
deployment.apps/kubernetes-bootcamp scaled
$ kubectl get deployments
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp   2/2     2            2           4m54s
$ kubectl get pods -o wide
NAME                                   READY   STATUS        RESTART
S   AGE     IP           NODE       NOMINATED NODE   READINESS GATE
S
kubernetes-bootcamp-fb5c67579-2p7v8   1/1     Running       0
    4m41s   172.18.0.6   minikube   <none>           <none>
kubernetes-bootcamp-fb5c67579-9s52r   1/1     Terminating   0
    113s    172.18.0.8   minikube   <none>           <none>
kubernetes-bootcamp-fb5c67579-rdtln   1/1     Running       0
    113s    172.18.0.9   minikube   <none>           <none>
kubernetes-bootcamp-fb5c67579-xd4q8   1/1     Terminating   0
    113s    172.18.0.7   minikube   <none>           <none>
$
```