+Learning Kubenetes Basics

https://kubernetes.io/docs/tutorials/kubernetes-basics/

Notes to share with anyone interesting in taking above tutorial – see License at Kubernetes.io. Notes provided as is as courtesy for your review from https://longislandnewtechnology.blogspot.com

Companion notes to https://kubernetes.io/docs/tutorials/kubernetes-basics

Notes intended to be shared with anyone interesting in taking above tutorial -> see Tutorial License at Kubernetes.io.

Annotated Notes with screenshots provided "AS IS" courtesy of https://longislandnewtechnology.blogspot.com

PDF document is "annotated guide" from K8SSG (tm - longislandnewtechnology) aka Kubernetes Study Group of Long Island, New York, USA

Any screenshots used elsewhere require link back to blog address https://longislandnewtechnology.blogspot.com

https://kubernetes.io/docs/tutorials/kubernetes-basics/

+Learning Kubenetes Basics

https://kubernetes.io/docs/tutorials/kubernetes-basics/

Using Minikube to create a cluster

https://kubernetes.io/docs/tutorials/kubernetes-basics/create-cluster/cluster-intro/

CREATE A CLUSTER

Once minikube is installed as a local VM, check to see if minikube is installed properly: type ***minikube version***

```
Kubernetes Bootcamp Terminal

$
$ minikube version
minikube version: v1.6.2
commit: 54f28ac5d3a815d1196cd5d57d707439ee4bb392
```

Start the cluster by typing "minikube start" …. there will be a delay here … when complete it will look like:

```
$ minikube start
* minikube v1.6.2 on Ubuntu 18.04
* Selecting 'none' driver from user configuration (alternates: [])
* Running on localhost (CPUs=2, Memory=2461MB, Disk=47990MB) ...
* OS release is Ubuntu 18.04.3 LTS
* Preparing Kubernetes v1.17.0 on Docker '18.09.7' ...
  - kubelet.resolv-conf=/run/systemd/resolve/resolv.conf

* Pulling images ...
* Launching Kubernetes ...
* Configuring local host environment ...
* Waiting for cluster to come online ...
* Done! kubectl is now configured to use "minikube"
$
$
```

You can run some ad hoc kubectl commands if you know some…

```
$ kubectl get pod
No resources found in default namespace.
$ kubectl get service
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   88s
$
```

https://kubernetes.io/docs/tutorials/kubernetes-basics/

Module 2 – check your cluster version by typing ***kubectl version***

```
$ kubectl version
Client Version: version.Info{Major:"1", Minor:"17", GitVersion:"v1.17.0", GitCommi
t:"70132b0f130acc0bed193d9ba59dd186f0e634cf", GitTreeState:"clean", BuildDate:"201
9-12-07T21:20:10Z", GoVersion:"go1.13.4", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"17", GitVersion:"v1.17.0", GitCommi
t:"70132b0f130acc0bed193d9ba59dd186f0e634cf", GitTreeState:"clean", BuildDate:"201
9-12-07T21:12:17Z", GoVersion:"go1.13.4", Compiler:"gc", Platform:"linux/amd64"}
$
```

The client version is the **kubectl version (v1.17.0)** while the server version is the **kubernetes version (v1.17.0)**

LEARN MORE ABOUT YOUR CLUSTER

kubectl cluster-info

```
$ kubectl cluster-info
Kubernetes master is running at https://172.17.0.10:8443
KubeDNS is running at https://172.17.0.10:8443/api/v1/namespaces/kube-system/servi
ces/kube-dns:dns/proxy

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

Powered by Katacoda

Notes: Cluster has a single note. You can use CLI or Dashboard (the latter allows you to view your applications in a UI)

kubectl get nodes

```
$ kubectl get nodes
NAME       STATUS   ROLES    AGE   VERSION
minikube   Ready    master   15m   v1.17.0
```

A status of ready means “node is ready to accept applications for deployment”

https://kubernetes.io/docs/tutorials/kubernetes-basics/

Using kubectl to Create a Deployment

https://kubernetes.io/docs/tutorials/kubernetes-basics/deploy-app/deploy-intro/
https://kubernetes.io/docs/tutorials/hello-minikube/

$ sleep 1; launch.sh

kubectl get nodes --help

```
$ kubectl get nodes --help
Display one or many resources

 Prints a table of the most important information about the specified resources.
You can filter the list using a label selector and the --selector flag. If the
desired resource type is namespaced you will only see results in your current
namespace unless you pass --all-namespaces.

 Uninitialized objects are not shown unless --include-uninitialized is passed.

 By specifying the output as 'template' and providing a Go template as the value
of the --template flag, you can filter the attributes of the fetched resources.

Use "kubectl api-resources" for a complete list of supported resources.

Examples:
  # List all pods in ps output format.
  kubectl get pods

  # List all pods in ps output format with more information (such as node name).
  kubectl get pods -o wide
  # List a single replication controller with specified NAME in ps output format.
  kubectl get replicationcontroller web

  # List deployments in JSON output format, in the "v1" version of the "apps" API group:
  kubectl get deployments.v1.apps -o json

  # List a single pod in JSON output format.
  kubectl get -o json pod web-pod-13je7

  # List a pod identified by type and name specified in "pod.yaml" in JSON output format.
  kubectl get -f pod.yaml -o json

  # List resources from a directory with kustomization.yaml - e.g. dir/kustomization.yaml.
  kubectl get -k dir/

  # Return only the phase value of the specified pod.
  kubectl get -o template pod/web-pod-13je7 --template={{.status.phase}}

  # List resource information in custom columns.
  kubectl get pod test-pod -o custom-columns=CONTAINER:.spec.containers[0].name,IMAGE:.spec.c
ontainers[0].image
```

https://kubernetes.io/docs/tutorials/kubernetes-basics/

```
  # List all replication controllers and services together in ps output format.
  kubectl get rc,services

  # List one or more resources by their type and names.
  kubectl get rc/web service/frontend pods/web-pod-13je7

Options:
  -A, --all-namespaces=false: If present, list the requested object(s) across all namespaces.
 Namespace in current context is ignored even if specified with --namespace.
      --allow-missing-template-keys=true: If true, ignore any errors in templates when a fiel
d or map key is missing in the template. Only applies to golang and jsonpath output formats.
      --chunk-size=500: Return large lists in chunks rather than all at once. Pass 0 to disab
le. This flag is beta and may change in the future.
      --field-selector='': Selector (field query) to filter on, supports '=', '==', and '!='.
(e.g. --field-selector key1=value1,key2=value2). The server only supports a limited number of
 field queries per type.
  -f, --filename=[]: Filename, directory, or URL to files identifying the resource to get fro
m a server.
      --ignore-not-found=false: If the requested object does not exist the command will retur
n exit code 0.
  -k, --kustomize='': Process the kustomization directory. This flag can't be used together w
ith -f or -R.
  -L, --label-columns=[]: Accepts a comma separated list of labels that are going to be prese
nted as columns. Names are case-sensitive. You can also use multiple flag options like -L lab
el1 -L label2...
      --no-headers=false: When using the default or custom-column output format, don't print
headers (default print headers).
  -o, --output='': Output format. One of: json|yaml|wide|name|custom-columns=...|custom-colum
ns-file=...|go-template=...|go-template-file=...|jsonpath=...|jsonpath-file=... See custom co
lumns [http://kubernetes.io/docs/user-guide/kubectl-overview/#custom-columns], golang templat
e [http://golang.org/pkg/text/template/#pkg-overview] and jsonpath template [http://kubernete
s.io/docs/user-guide/jsonpath].
      --raw='': Raw URI to request from the server.  Uses the transport specified by the kube
config file.
  -R, --recursive=false: Process the directory used in -f, --filename recursively. Useful whe
n you want to manage related manifests organized within the same directory.
  -l, --selector='': Selector (label query) to filter on, supports '=', '==', and '!='.(e.g.
-l key1=value1,key2=value2)
      --server-print=true: If true, have the server return the appropriate table output. Supp
orts extension APIs and CRDs.
      --show-kind=false: If present, list the resource type for the requested object(s).
      --show-labels=false: When printing, show all labels as the last column (default hide la
bels column)
      --sort-by='': If non-empty, sort list types using this field specification.  The field
specification is expressed as a JSONPath expression (e.g. '{.metadata.name}'). The field in t
he API resource specified by this JSONPath expression must be an integer or a string.
      --template='': Template string or path to template file to use when -o=go-template, -o=
go-template-file. The template format is golang templates [http://golang.org/pkg/text/templat
e/#pkg-overview].
  -w, --watch=false: After listing/getting the requested object, watch for changes. Uninitial
ized objects are excluded if no object name is provided.
      --watch-only=false: Watch for changes to the requested object(s), without listing/getti
ng first.

Usage:
  kubectl get [(-o|--output=)json|yaml|wide|custom-columns=...|custom-columns-file=...|go-tem
plate=...|go-template-file=...|jsonpath=...|jsonpath-file=...] (TYPE[.VERSION][.GROUP] [NAME
| -l label] | TYPE[.VERSION][.GROUP]/NAME ...) [flags] [options]

Use "kubectl options" for a list of global command-line options (applies to all commands).
$
```

https://kubernetes.io/docs/tutorials/kubernetes-basics/

$ kubectl version

```
$ kubectl version
Client Version: version.Info{Major:"1", Minor:"15", GitVersion:"v1.15.2", GitCommit:"f6278300be
bbb750328ac16ee6dd3aa7d3549568", GitTreeState:"clean", BuildDate:"2019-08-05T09:23:26Z", GoVers
ion:"go1.12.5", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"15", GitVersion:"v1.15.0", GitCommit:"e8462b5b5d
c2584fdcd18e6bcfe9f1e4d970a529", GitTreeState:"clean", BuildDate:"2019-06-19T16:32:14Z", GoVers
ion:"go1.12.5", Compiler:"gc", Platform:"linux/amd64"}
$
```

Note: This says kubelet is version 1.15.2 and kubernetes (server) is v1.15.0

$ kubectl get nodes

```
$ kubectl get nodes
NAME       STATUS   ROLES    AGE   VERSION
minikube   Ready    master   10m   v1.15.0
$
```

DEPLOYING AN APPLICATION TO A KUBERNETES CLUSTER requires a image to pull down and the number of replicas.

Before using **kubernetes create deployment** command, we will need to provide:

- the deployment name
- the app image location (including the full repository URL for images hosted outside of Docker hub)

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

```
$ kubectl create deployment kubernetes-bootcamp --image=gcr.io/google-samples/kubernetes-bootca
mp:v1
deployment.apps/kubernetes-bootcamp created
$
```

What happened? Several steps occurred while creating your deployment:

1. Kubenetes (KDC) searched for a suitable node where the instance of the application could be run (which was trivial here since we only have one available node)
2. Kubernetes scheduled the application to run on the Node
3. Kubernetes configured the cluster to reschedule the instance on a new Node when needed

What's next? Confirmed your deployment using **$ kubectl get deployments** to list your deployments

```
$ kubectl get deployments
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp   1/1     1            1           4m10s
$
```

What's the result: You have a single deployment, called "kubernetes-bootcamp" and it is running a single instance of your app. The app instance is running inside a Docker container on your node.

https://kubernetes.io/docs/tutorials/kubernetes-basics/

View your app

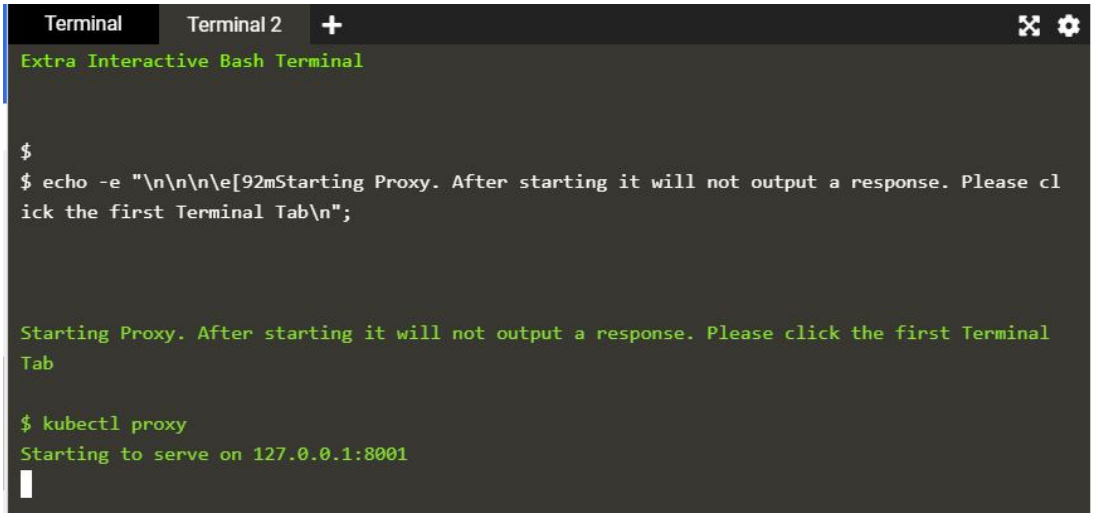
Actually as you know, your container is running within a Pod on inside of Kubernetes (in this case on the Node). The Pod is running on a private, isolated network. By default, the they are visible to other pods and services within the same Kubernetes cluster, but not outside the network.

When we use kubectl, we are using an API endpoint to communicate to our application.

Using kubectl, we can create a Kube Proxy that will forward communciations into the cluster-wide private network. The proxy can be terminated by pressing Ctrl-C and won't show any output while it is running.

LAB: Open a second terminal window to run the proxy: (use + to create 2nd terminal and enter supplied command)

```
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
```

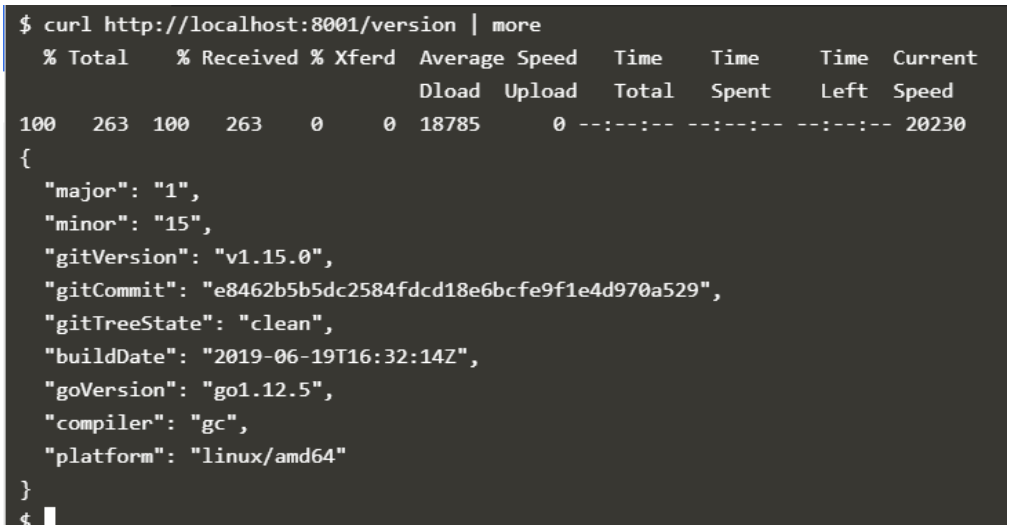


Click on Terminal to continue:

Note: we now have a connection between our host (the online terminal) and the Kubernetes cluster. The Kube Proxy enables direct access to the API from the provided Lab Terminals.

NEXT

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



https://kubernetes.io/docs/tutorials/kubernetes-basics/

Since the Kube Proxy is running, the API Server will automatically create an endpoint for each pod, based on the pod name that is also accessible through the Proxy. You will need the Pod name, and we'll store it in the environment variable POD_NAME.

One way:

```
$ kubectl get pods
NAME                                   READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-75bccb7d87-xss9k   1/1     Running   0          27m
$
```

Lab way:

export POD_NAME=$(kubectl get pods –o go-template --template '{{.metadata.name}}{{"\n"}}{{end}}')

echo Name of the Pod: $POD_NAME

```
$ kubectl get pods
NAME                                   READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-75bccb7d87-xss9k   1/1     Running   0          27m
$ 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-75bccb7d87-xss9k
$
```

KEY FACT: localhost:8001 requires the Kube Proxy to be running (as shown in Terminal 2). While the proxy still runs, our curl commands can work using localhost:8001/path.

KEY FINDING: In order for the new deployment to be accessible without using the Proxy, a **Service is required to be created. [Explained in future lab]**

First we need to get the Pod name, and we'll store in the environment variable 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
```

Note: Check the top of the terminal. The proxy was run in a new tab (Terminal 2), and the recent commands were executed the original tab (Terminal 1). The proxy still runs in the second tab, and this allowed our curl command to work using `localhost:8001`.

https://kubernetes.io/docs/tutorials/kubernetes-basics/

Module 3 – Viewing Pods and Nodes; Troubleshooting deployed applications…

https://kubernetes.io/docs/tutorials/kubernetes-basics/explore/explore-intro/

A Pod is a group of one or more application containers (such as Docker or rkt) and includes shared storage (volumes), IP address and information about how to run them.

A "Pod" is a Kubernetes abstraction that represents a group of one or more application containers (like Docker or rkt) and some shared resources for those containers such as:

- shared storage or "Volumes"
- networking, such as a "unique cluster wide IP address"
- Information or metadata about each container, such as the container image version and specific port to use

Each Pod models an application-specific "logical host" and can contain different application containers that are relatively tightly coupled.

For instance a Pod might contain both the container with your Node.js app as well as a different container that feeds data to be published by the Node.js webserver. All containers within a single Pod share an IP address and common port space; are always co-located (same node) and co-scheduled (exist at same time); and run in the shared context on the same Worker Node.

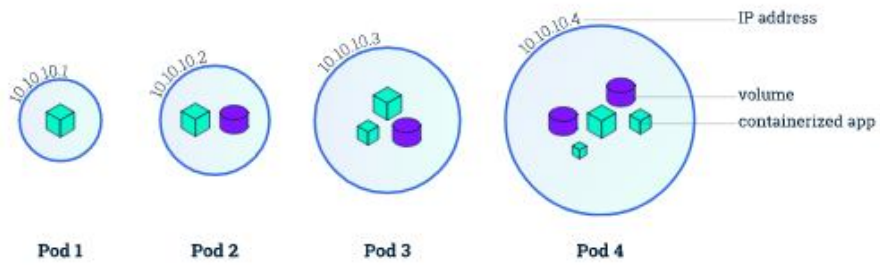
Pods are the atomic unit on the Kubernetes platform. Pods are also referred to the unit of scheduling like a VM in VMware or a Process ID (pid) in Linux, Windows and UNIX.

When we create a Deployment on Kubernetes, that Deployment creates Pods with containers inside of them (as opposed to creating the containers directly).

Each Pod is tied to the Node where it is scheduled, and remains there until termination (according to the restart policy) or Pod deletion. In the case of a Node Failure, then identical Pods are scheduled on other available Nodes in the cluster.

Note: Other features of the cluster (Load balancing, etc) ensure that the externally available IP address doesn't' change and the Cluster will handle re-routing network traffic to the newly established Pod once it is available after Node Failure. In general cases, applications should have N+1 or N+2 instances (that translates to minimum of 3 or 4 instances) at all times for maximum balance of availability and maintainability.

Pods

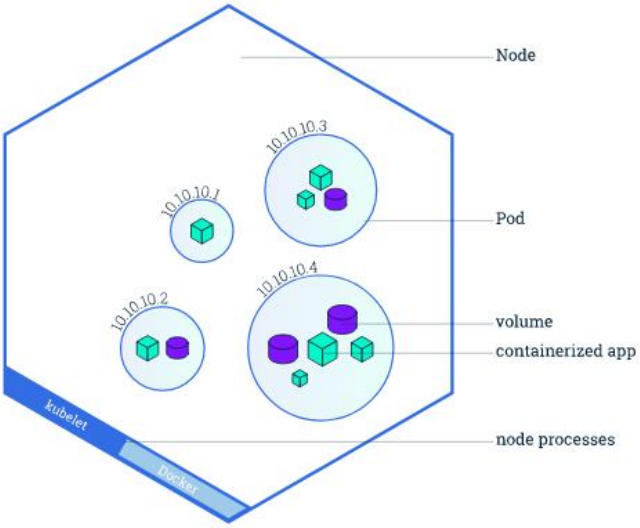


Containers should only be scheduled together in a single Pod if they are tightly coupled and need to share resources such as disk. In the above diagrams, Pod1, Pod2 and Pod3 are more typical, with Pod 3 example having a dedicated helper node.

Note: A Volume exists or is deleted when the Pod terminates or is deleted. Learn more about "Persistent Volume" where are tied to the Cluster lifetime later to have more persistence across any given instance's life cycle. External data stores can be arranged by the Kubernetes Administrator such as PostgreSQL, MemDB (temporary in-memory DB great for DevOps proof-of-concepts, etc), etc.

https://kubernetes.io/docs/tutorials/kubernetes-basics/

Node Overview



Nodes

A Pod always runs on a **Node**. The Node is a worker machine (i.e., "worker node") in Kubernetes and may be either a virtual machine or a physical server, depending on the cluster. Each Node is managed by the master. A Node can have multiple pods, and the Kubernetes master automatically handles all scheduling of the pods across all available Nodes in the cluster, based on weighted formulas, available resources that match the deployment plan, etc. As stated, The Master's automatic scheduling takes into account the available resources on each Node and maintains this metadata.

Every Kubernetes Node runs at least:

- Kubelet—a process responsible for communication between the Node and the Kubernetes Master; the process also manages the local Pods and containers running on a machine where it resides.
- A container runtime (like Docker, rkt), also known as container orchestration engine,--which is a container framework responsible for pulling the container image from a registry (repository), unpacking the container, and running the application in the Pod instance provided by Kubernetes.

Troubleshooting with Kubectl

In a previous lab, you starting using the kubectl command-line interface and interacted with the Master Node. In Module 3 you will contain to use 'kubectl' to get more infmromation about the Pods, the deployed applications and the environment that has been step.

In particular, you will use the most common operations or subcommands assocated with kubectl, such as:

- **kubectl get** list cluster resources
- kubectl describe show more detailed information about a resource
- kubectl logs print the logs from a container within a pod
- kubectl exec execute commands directly on a container within a pod

Learn how to see when applications were deployed, what the applications' current status are; where they are presently running in the Cluster and what their configuration are. Later with labels and other metadata, you will learn how to refine your commands to search for what you need with precision across larger and larger clusters with hundreds or thousands of nodes.

Given what we have discovered about our cluster components and the command line, start lab for Module 3.

https://kubernetes.io/docs/tutorials/kubernetes-basics/

Lab 3 “Exploring Your App” as companion to the Module 3 discussion

```
Terminal +
Kubernetes Bootcamp Terminal

$
$ sleep 1; launch.sh
Starting Kubernetes. This is expected to take less than a minute
Kubernetes Started
$
```

$ kubectl get pods *Note: If you didn’t see the single Pod, then please wait for Kubernetes cluster to start and try the command again*

```
$ kubectl get pods
NAME                                   READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-5b48cfdcbd-vnnzx   1/1     Running   0          100s
```

As mentioned the next command **$ kubectl describe pods** will get significantly more information.

```
$ kubectl describe pods
Name:           kubernetes-bootcamp-5b48cfdcbd-vnnzx
Namespace:      default
Priority:       0
Node:           minikube/172.17.0.13
Start Time:     Tue, 14 Jan 2020 03:32:33 +0000
Labels:         pod-template-hash=5b48cfdcbd
                run=kubernetes-bootcamp
Annotations:    <none>
Status:         Running
IP:             172.18.0.4
Controlled By:  ReplicaSet/kubernetes-bootcamp-5b48cfdcbd
Containers:
  kubernetes-bootcamp:
    Container ID:   docker://67f0d0e6a435e5ff49790db9908a351e73ae75873d68fb326759b269a623efc1
    Image:          gcr.io/google-samples/kubernetes-bootcamp:v1
    Image ID:       docker-pullable://jocatalin/kubernetes-bootcamp@sha256:0d6b8ee63bb57c5f5b6156f446b3bc3b3c143d233037f3a2f00
e279c8fcc64af
    Port:           8080/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Tue, 14 Jan 2020 03:32:34 +0000
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-cb2mp (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  default-token-cb2mp:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-cb2mp
    Optional:    false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute for 300s
                 node.kubernetes.io/unreachable:NoExecute for 300s
Events:
  Type    Reason     Age    From               Message
  ----    ------     ----   ----               -------
  Normal  Scheduled  3m34s  default-scheduler  Successfully assigned default/kubernetes-bootcamp-5b48cfdcbd-vnnzx to minikube
  Normal  Pulled     3m33s  kubelet, minikube  Container image "gcr.io/google-samples/kubernetes-bootcamp:v1" already present
on machine
  Normal  Created    3m33s  kubelet, minikube  Created container kubernetes-bootcamp
  Normal  Started    3m33s  kubelet, minikube  Started container kubernetes-bootcamp
$
```

Using **$ kubectl describe pod** we can see more details about the Pod's container such as:

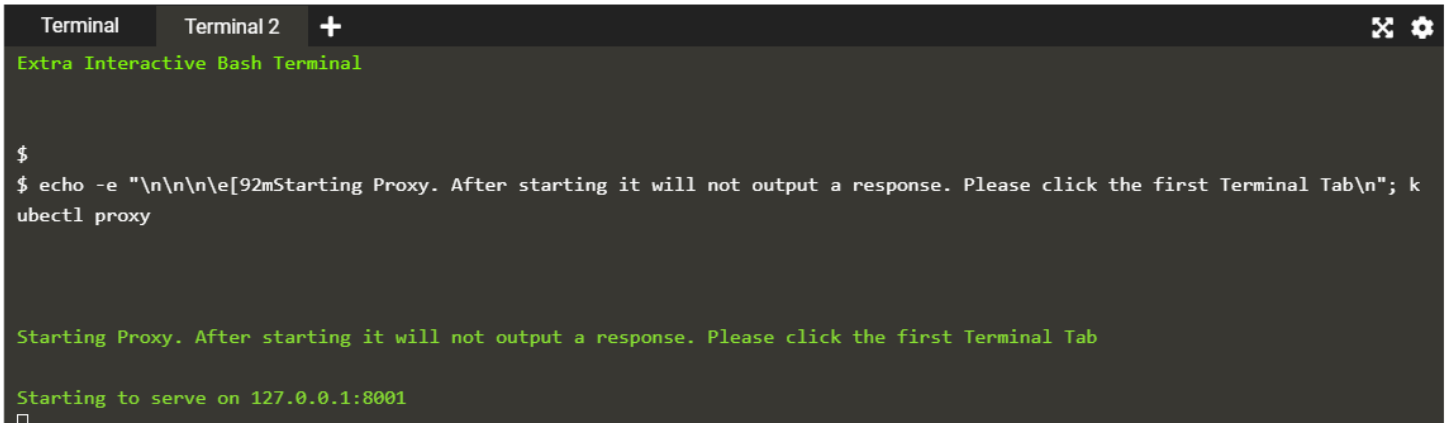
- IP address
- the ports in use and used
- a list of events related to the lifecycle of the Pod (get used to the order of events for alter troubleshooting)

In addition, you can use the **kubectl describe *{other Kubernetes primatives}*** to find out about most Kubernetes primatives such as nodes, pods, deployments, etc. The goal of **describe output** is it is designed to be **human readable, not to be scripted against – which won't stop anyone from trying.**

Lab 3.2 Show the App in the Terminal

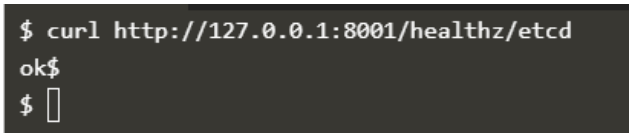
As in the previous lab, we will use the "+" in the Session to create "Terminal 2" so we can establish a Kube Proxy somce the pods are running on an isolated, private network and we need to "proxy" access to them so we can interact and debug with them.

$ echo –e "\n\n\n\e[92mStarting Proxy. After starting it will not output a response. Please click the first Terminat Tab\n"; kubectl proxy



As before click on "Terminal", and I like to verify you are able to access the resources; and that you can test the API:

Here is one of my simple API Test commands after some exploring after Module 2 (see Appendix)



https://kubernetes.io/docs/tutorials/kubernetes-basics/

Lab 3.2 continues

Now again, we'll get the Pod name and query that pod directly through the proxy. To get the Pod name and store it in the POD_NAME environment variable:

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

Note: you can see that the "Name of the Pod" output is same as what kubectl get pods returned above.

```
$ 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-5b48cfdcbd-vnnzx
$
```

To see the output of our application, run a curl request.

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

The url is the route to the API of the Pod. Here is the output:

```
$ curl http://localhost:8001/api/v1/namespaces/default/pods/$POD_NAME/proxy/
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-5b48cfdcbd-vnnzx | v=1
$
```

Lab 3.3 View the Container Logs

Since the prior lab created an environment variable, we can simply use $POD_NAME instead of dealing with the longer name of the pod and avoid mistakes. **$ kubectl logs $POD_NAME** will provide the following output:

```
$ kubectl logs $POD_NAME
Kubernetes Bootcamp App Started At: 2020-01-14T03:32:34.740Z | Running On:  kubernetes-bootcamp-5b48cfdcbd-vnnzx

Running On: kubernetes-bootcamp-5b48cfdcbd-vnnzx | Total Requests: 1 | App Uptime: 1361.899 seconds | Log Time: 2020-01-14T03:
55:16.639Z
$
```

Lab 3.4 Executing command on the actual Container

Since the Pod is up and running, you can address commands, using **kubectl exec {pod-name} {command} as follows:**

$ kubectl exec $POD_NAME env

```
$ kubectl exec $POD_NAME env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=kubernetes-bootcamp-5b48cfdcbd-vnnzx
KUBERNETES_PORT_443_TCP_PORT=443
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
NPM_CONFIG_LOGLEVEL=info
NODE_VERSION=6.3.1
HOME=/root
$
```

https://kubernetes.io/docs/tutorials/kubernetes-basics/

Note: It is good habit to specify the POD NAME even though, with a single Pod on a single Node it could be omitted.

Creating a Bash Session on the Pod's container:

$ kubectl exec –ti $POD_NAME bash

```
$ kubectl exec -ti $POD_NAME bash
root@kubernetes-bootcamp-5b48cfdcbd-vnnzx:/# wc -l server.js
19 server.js
root@kubernetes-bootcamp-5b48cfdcbd-vnnzx:/#
```

Since we now have an open console on the container hosting our NodeJS application, we can show the source code of our app located in server.js file. But first, I was curious on how many lines of code are in this app, server.js
$ wc –l server.js above is a simple Linux command that Ubuntu supports that says the app is 19 lines long.

However, the lab wants you to run **/# cat server.js** to see the entire app in NodeJS container

```
root@kubernetes-bootcamp-5b48cfdcbd-vnnzx:/# cat server.js
var http = require('http');
var requests=0;
var podname= process.env.HOSTNAME;
var startTime;
var host;
var handleRequest = function(request, response) {
  response.setHeader('Content-Type', 'text/plain');
  response.writeHead(200);
  response.write("Hello Kubernetes bootcamp! | Running on: ");
  response.write(host);
  response.end(" | v=1\n");
  console.log("Running On:" ,host, "| Total Requests:", ++requests,"| App Uptime:", (new Date() - startTime)/1000 , "seconds", "| Log Ti
me:",new Date());
}
var www = http.createServer(handleRequest);
www.listen(8080,function () {
    startTime = new Date();;
    host = process.env.HOSTNAME;
    console.log ("Kubernetes Bootcamp App Started At:",startTime, "| Running On: " ,host, "\n" );
});
root@kubernetes-bootcamp-5b48cfdcbd-vnnzx:/#
```

Now let's confirm the application, server.js, is up and running by testing the API again:

```
root@kubernetes-bootcamp-5b48cfdcbd-vnnzx:/# curl localhost:8080
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-5b48cfdcbd-vnnzx | v=1
root@kubernetes-bootcamp-5b48cfdcbd-vnnzx:/#
```

Note: here we used localhost because we executed the command inside the NodeJS Pod. If you cannot connect to localhost:8080, check to make sure you have run the kubectl exec command and are launching the command from within the Pod.

After typing exit, you will leave the connection to Nodejs container and can review the logs for the Pod again:

```
$ kubectl exec -ti $POD_NAME bash
root@kubernetes-bootcamp-5b48cfdcbd-vnnzx:/# ls
bin  boot  core  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  server.js  srv  sys  tmp  usr  var
root@kubernetes-bootcamp-5b48cfdcbd-vnnzx:/# exit
exit
$
```

You can see the logs below:

```
$ kubectl logs $POD_NAME
Kubernetes Bootcamp App Started At: 2020-01-14T03:32:34.740Z | Running On:  kubernetes-bootcamp-5b48cfdcbd-vnnzx

Running On: kubernetes-bootcamp-5b48cfdcbd-vnnzx | Total Requests: 1 | App Uptime: 1361.899 seconds | Log Time: 2020-01-14T03:55:16.639Z
Running On: kubernetes-bootcamp-5b48cfdcbd-vnnzx | Total Requests: 2 | App Uptime: 2234.447 seconds | Log Time: 2020-01-14T04:09:49.187Z
$
```

https://kubernetes.io/docs/tutorials/kubernetes-basics/

Module 4 – Using a Service to Expose your App PUBLICLY

https://kubernetes.io/docs/tutorials/kubernetes-basics/explore/explore-intro/

All Kubernetes Pods have a lifecycle. When a worker node dies, the Pods running on the Node are also lost. Any Volumes attached to the Pod are lost. The state is gone.

With a failed Node, a ReplicaSet might then dynamically drive the cluster back to the desired state via creation of new Pods identical to the lost Pods to keep your application running for the next set of input received.

Another example might be an image-processing backend with 3 replicas. Those replicas are exchangeable; the front-end system should not care about backend replicas or even if a Pod is lost and recreated.

Summary – each Pod in a Kubernetes cluster has a unique IP address, even Pods on the same Node, so there needs to be a way of automatically reconciling changes among Pods so that your applications can continue to function.

A Pod is a group of one or more application containers (such as Docker or rkt) and includes shared storage (volumes), IP address and information about how to run them. The previous lab used Kube Proxy to provide a way of reaching into the unique private address space assigned to a given Pod and we tested the access using curl command. However Kubernetes provides the means to expose an application outside the Kubenetes cluster using a ***SERVICE.***

A Kubernetes Service is an abstraction layer which defines a logical set of Pods and enables external traffic exposure, load balancing and service discovery for those Pods.

A "SERVICE" in Kubernetes is an abstraction which defines a **logical set of Pods and a policy by which to access them**. Services enable a *loose coupling* between dependent Pods**. A Service is defined using YAML (preferred, or JSON as last resort)**, like all Kubernetes objects. The set of Pods targeted by a Service is usually determined by a ***LabelSelector*** (see below why you might want a Service without including **selector** in the spec).

A SERVICE ALLOWS YOUR APPLICATIONS TO RECEIVE TRAFFIC. Without a Service, even though each Pod has a unique IP address, it is private to the cluster, and those IPs are not exposed outside of the cluster without a Service. Services can be exposed in different ways by specifying a **type** in the **ServiceSpec:**

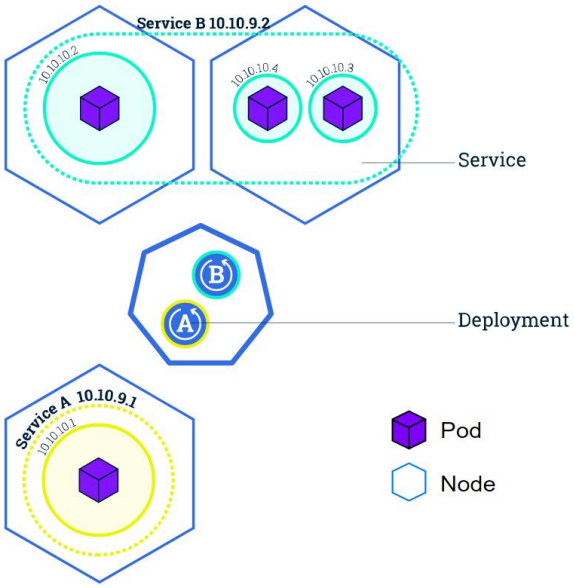
- **ClusterIP (default)**—exposes the service on the internal IP in the cluster. This type makes the service only reachable from within the cluster itself.
- **NodePort**—exposes the Service on the same port of each selected Node in the cluster using NAT. Makes a Service accessible from outside the cluster using <NodeIP>:<NodePort>. This is a SuperSet of ClsuterIP.
- **LoadBalancer**—creates an external load balancer in the current cloud (if supported) and assigns a fixed, external IP to the Service. Superset of NodePort.
- **ExternalName**—exposes the Service using an arbitrary name (specified by **externalName** in the spce) by returning a CNAME record with the name. No proxy is used. This type requires Kubernetes v1.7 or higher of kube-dns.

Refer to "Using Source IP" (https://kubernetes.io/docs/tutorials/services/source-ip/) tutorial for more details on the different types of Services. Also see "Connecting Applications with Services" (https://kubernetes.io/docs/concepts/services-networking/connect-applications-service).

Note: A service created without **selector** will also not create the corresponding Endpoints object. This allows users to manually map a Service to specific endpoints. Another possibility is that you are strictly using **type: ExternalName** and there is no selector.

https://kubernetes.io/docs/tutorials/kubernetes-basics/

Services and Labels

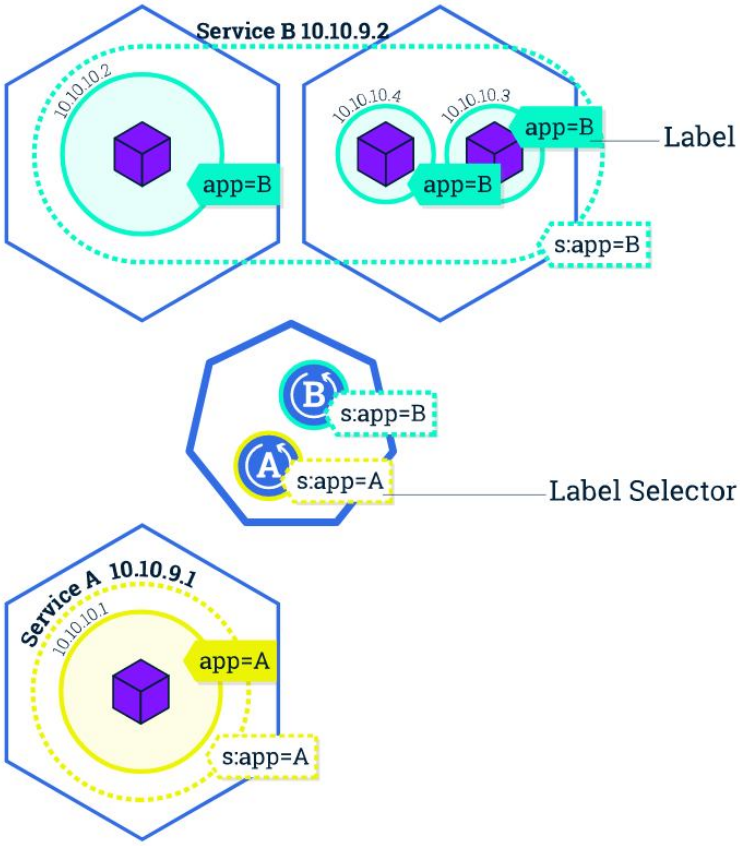


Did you know…
You can create a Service at the same time you create a Deployment by using `--expose` in kubectl.

The role of a Service is to route network traffic across a set of Pods. Services are abstractions that allow pods to die and replicate in Kubernetes without impacting your application. Discovery and routing among dependent Pods (such as the front-end and back-end components in an application) is handled by Kubernetes Services.

Services match a set of Pods using **labels ( )** and **selectors** ( ), which is a grouping primitive that allows logical operations on objects in Kubernetes. Labels are key/value pairs attached to objects and can be used in many ways:

- designate objects for development, test, build and production
- embed version tags
- classify an object using tags



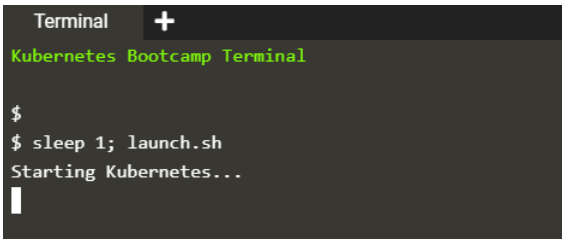
In summary, Labels can be attached to objects at creation time or can be added later on.

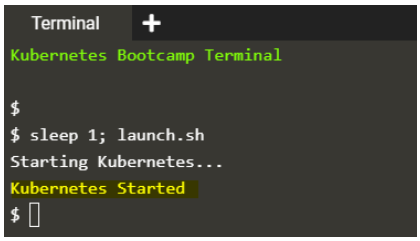
Labels can be modified at any time.

The Lab 4 for Module 4 will expose our application publically using a Service and then spend some time applying labels.

https://kubernetes.io/docs/tutorials/kubernetes-basics/

Lab 4 “Exposing Your App Publically using a Service” as companion to the Module 4 discussion





Some times you have to wait for your provisioning …. (45 seconds…) …. before you see “**Kubernetes Started”.** Now, run a few commands to know the state of your cluster that took time to start up:

Suggested: **$ kubectl get pods**
$ kubectl get deployments
$ kubectl get services

$ kubectl get pods *Note: If you didn’t see the single Pod, then please wait for Kubernetes cluster to start and try again*

```
$ kubectl get pods
NAME                                 READY     STATUS    RESTARTS   AGE
kubernetes-bootcamp-5c69669756-rk8zp   1/1       Running   0          1m
$ kubectl get deployments
NAME                  DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp   1         1         1            1           1m
$ kubectl get services
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   1m
$
```

Currently, we are running a Service called kubernetes-bootcamp. The Service received a unique cluster-IP, an internal port and an external IP (the IP of the Node). – however, we must use “expose” option for kubectl

```
$ kubectl expose deployment/kubernetes-bootcamp --type="NodePort" --port 8080
service/kubernetes-bootcamp exposed
$ kubectl get services
NAME                  TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
kubernetes            ClusterIP   10.96.0.1       <none>        443/TCP          10m
kubernetes-bootcamp   NodePort    10.100.94.226   <none>        8080:31991/TCP   6s
```

To find out what port was opened externally (by the NodePort option), please run the **describe service** command

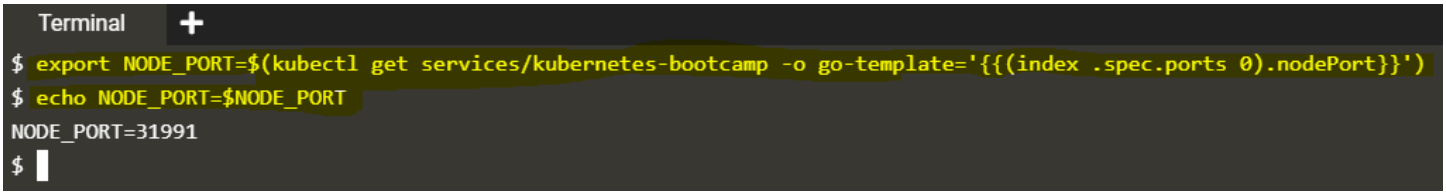
```
$ kubectl describe services/kubernetes-bootcamp
Name:                     kubernetes-bootcamp
Namespace:                default
Labels:                   run=kubernetes-bootcamp
Annotations:              <none>
Selector:                 run=kubernetes-bootcamp
Type:                     NodePort
IP:                       10.100.94.226
Port:                     <unset>  8080/TCP
TargetPort:               8080/TCP
NodePort:                 <unset>  31991/TCP
Endpoints:                172.18.0.4:8080
Session Affinity:         None
```

https://kubernetes.io/docs/tutorials/kubernetes-basics/

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
```

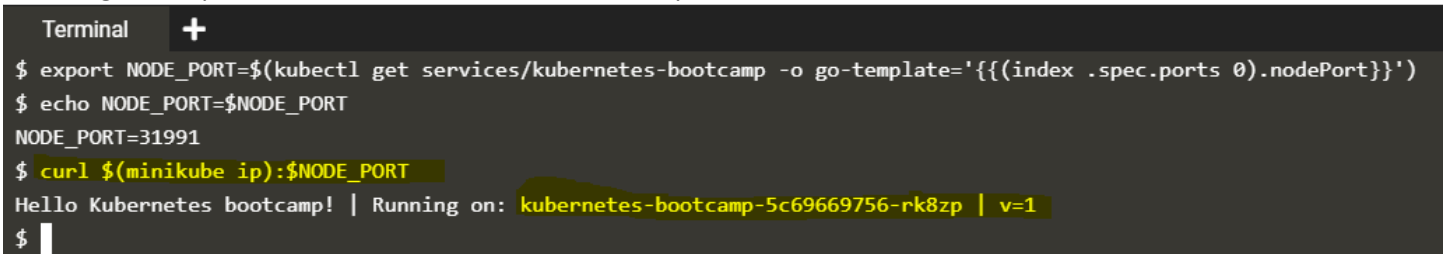
Results:



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

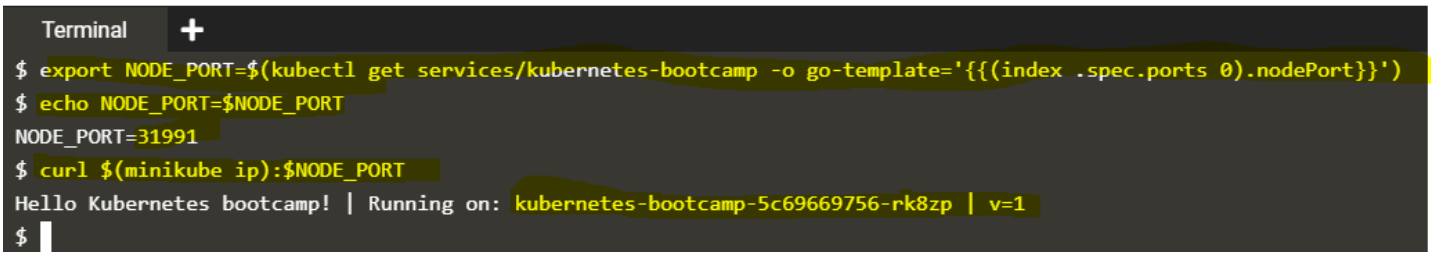
```
curl $(minikube ip):$NODE_PORT
```

And we get a response from the server. The Service is exposed.



Lab 4.2 Using Labels (as part of Exposing your App Publicly with Services)

So far, exposed our service using the kubectl expose command, we were able to describe the service and identify the NodePort assigned and put that information into an environment variable $NODE_PORT we subsequently used to use the **curl $(minikube ip):$NODE_PORT** command to test that the app is actually exposed outside of the cluster:



https://kubernetes.io/docs/tutorials/kubernetes-basics/

Lab 4.2 continues

In this lab, 4.2, we are going to use labels – and to do that, we have to learn the automatically created label the Deployment assigned to our Pod. We do this with the **$ kubectl describe deployment**

```
$ kubectl describe deployment | more
Name:                   kubernetes-bootcamp
Namespace:              default
CreationTimestamp:      Tue, 14 Jan 2020 04:57:37 +0000
Labels:                 run=kubernetes-bootcamp
Annotations:            deployment.kubernetes.io/revision=1
Selector:               run=kubernetes-bootcamp
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=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-5c69669756 (1/1 replicas created)
Events:
  Type    Reason             Age   From                   Message
  ----    ------             ----  ----                   -------
  Normal  ScalingReplicaSet  27m   deployment-controller  Scaled up replica set kubernetes-bootcamp-5c69669756 to 1
$
```

The fourth line provides the value “run=kubernetes-bootcamp” as the label

```
Labels:                 run=kubernetes-bootcamp
```

Use this value with the –l option on “kubectl get pods” and “kubectl get services” as follows:

```
$ kubectl get pods -l run=kubernetes-bootcamp
NAME                                   READY     STATUS    RESTARTS   AGE
kubernetes-bootcamp-5c69669756-rk8zp   1/1       Running   0          31m
$ kubectl get services -l run=kubernetes-bootcamp
NAME                  TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
kubernetes-bootcamp   NodePort   10.100.94.226   <none>        8080:31991/TCP   20m
```

Once again, put the name of this Pod in $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-5c69669756-rk8zp
$
```

https://kubernetes.io/docs/tutorials/kubernetes-basics/

To apply a new label we use the label command followed by the object type, object name and the new label: `kubectl label pod $POD_NAME app=v1`

```
$ kubectl label pod $POD_NAME app=v1
pod/kubernetes-bootcamp-5c69669756-rk8zp labeled
```

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`

```
$ kubectl describe pods $POD_NAME | more
Name:           kubernetes-bootcamp-5c69669756-rk8zp
Namespace:      default
Node:           minikube/172.17.0.26
Start Time:     Tue, 14 Jan 2020 04:57:55 +0000
Labels:         app=v1
                pod-template-hash=1725225312
                run=kubernetes-bootcamp
Annotations:    <none>
Status:         Running
IP:             172.18.0.4
Controlled By:  ReplicaSet/kubernetes-bootcamp-5c69669756
Containers:
  kubernetes-bootcamp:
    Container ID:   docker://87498c04a988734a0a1794e2863dfe4db86f6f160f910535546325133eec5955
    Image:          gcr.io/google-samples/kubernetes-bootcamp:v1
    Image ID:       docker-pullable://gcr.io/google-samples/kubernetes-bootcamp@sha256:0d6b8ee63bb57c5f5b6156f446b3bc3b3c143d2330
37f3a2f00e279c8fcc64af
    Port:           8080/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Tue, 14 Jan 2020 04:57:56 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-jdzdx (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  PodScheduled      True
Volumes:
  default-token-jdzdx:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-jdzdx
    Optional:    false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute for 300s
                 node.kubernetes.io/unreachable:NoExecute for 300s
Events:
  Type     Reason                  Age                From               Message
  ----     ------                  ----               ----               -------
  Warning  FailedScheduling        35m (x5 over 35m)  default-scheduler  0/1 nodes are available: 1 node(s) were not ready.
  Normal   Scheduled               35m                default-scheduler  Successfully assigned kubernetes-bootcamp-5c69669756-rk8z
p to minikube
  Normal   SuccessfulMountVolume   35m                kubelet, minikube  MountVolume.SetUp succeeded for volume "default-token-jdz
dx"
  Normal   Pulled                  35m                kubelet, minikube  Container image "gcr.io/google-samples/kubernetes-bootcam
p:v1" already present on machine
  Normal   Created                 35m                kubelet, minikube  Created container
  Normal   Started                 35m                kubelet, minikube  Started container
$
```

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 app=v1
```

And we see the Pod.

```
$ kubectl get pods -l app=v1
NAME                                   READY     STATUS    RESTARTS   AGE
kubernetes-bootcamp-5c69669756-rk8zp   1/1       Running   0          37m
$
```

Lab 4.3 Deleting a Service (as part of Exposing your App Publicly with Services)

Kubectl has a "delete service" option that will allow you to clean up. During the test, you should validate what you intended did occur:

```
$ kubectl get services
NAME                  TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
kubernetes            ClusterIP   10.96.0.1       <none>        443/TCP          39m
kubernetes-bootcamp   NodePort    10.100.94.226   <none>        8080:31991/TCP   28m
$ kubectl delete service -l run=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   39m
$
```

You can further demonstrate the Service was removed using the **$ curl $(minikube i):$NODE_PORT** command again:

```
$ curl $(minikube ip):$NODE_PORT
curl: (7) Failed to connect to 172.17.0.26 port 31991: Connection refused
$
```

Last, is your app still running? You can verify in one command by having the Container within the Pod execute it:

$ kubectl exec –ti $POD_NAME curl localhost:8080

```
$ kubectl exec -ti $POD_NAME curl localhost:8080
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-5c69669756-rk8zp | v=1
$
```

So we see the application is up and running. Remember, the deployment is managing the application. To complete the lifecycle, you will have to shut down the application which means you need to delete the Deployment as well.

https://kubernetes.io/docs/tutorials/kubernetes-basics/

Module 5 – Running Multiple Instances of Your App

Objectives: **Scale an app using kubectl.**

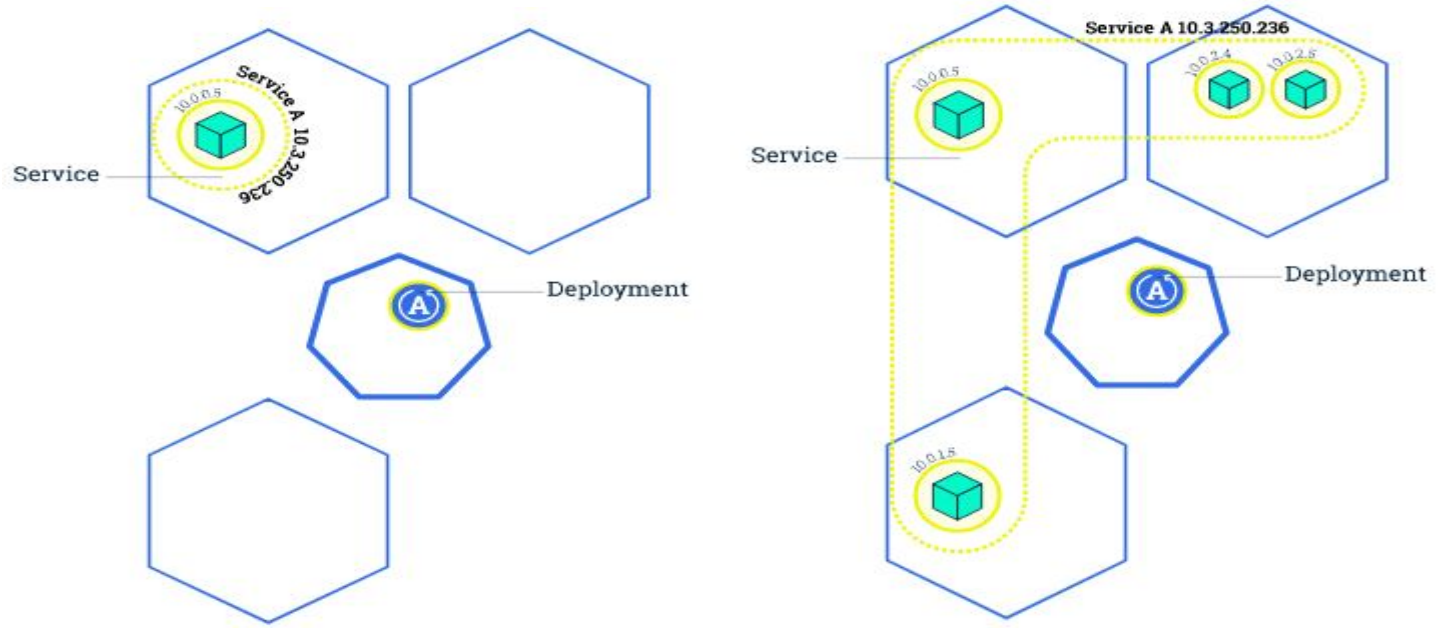
https://kubernetes.io/docs/tutorials/kubernetes-basics/scale/scale-intro/

Scaling an application

In the previous modules we created a Deployment, and then exposed it publicly via a Service. The Deployment created only one Pod for running our application. When traffic increases, we will need to scale the application to keep up with user demand. **Scaling** is accomplished by changing the number of replicas in a Deployment

You can scale from the start, using the --replica parameter for the kubectl run command, when you create a Deployment with multiple instances

Scaling overview



Scaling out a Deployment will ensure new Pods are created and scheduled to Nodes with available resources. Scaling will increase the number of Pods to the new desired state. Kubernetes also supports autoscaling of Pods, but it is outside of the scope of this tutorial. Scaling to zero is also possible, and it will terminate all Pods of the specified Deployment.

Running multiple instances of an application will require a way to distribute the traffic to all of them. Services have an integrated load-balancer that will distribute network traffic to all Pods of an exposed Deployment. Services will monitor continuously the running Pods using endpoints, to ensure the traffic is sent only to available Pods.

Once you have multiple instances of an Application running, you would be able to do Rolling updates without downtime. We'll cover that in the next module. Now, let's go to the online terminal and scale our application.

➔ *Scaling is accomplished by changing the number of replicas in a Deployment.*

https://kubernetes.io/docs/tutorials/kubernetes-basics/

Lab 5 "Scaling Your App" as companion to the Module 5 discussion

The goal of this interactive scenario is to scale a deployment with kubectl scale and to see the load balancing in action
The online terminal is a pre-configured Linux environment that can be used as a regular console (you can type commands). Clicking on the blocks of code followed by the ENTER sign will execute that command in the terminal.

```
Kubernetes Bootcamp Terminal
$
$ sleep 1; launch.sh
Starting Kubernetes. This is expected to take less than a minute
Kubernetes Started
$
```

Some times you have to wait for your provisioning …. (45 seconds…) …. before you see "**Kubernetes Started".** Now, run a few commands to know the state of your cluster that took time to start up:

Suggested: **$ kubectl get pods**
$ kubectl get deployments
$ kubectl get services

```
Kubernetes Bootcamp Terminal
$
$ sleep 1; launch.sh
Starting Kubernetes. This is expected to take less than a minute
Kubernetes Started
$ kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-5b48cfdcbd-dhjm8   1/1     Running   0          61s
$ kubectl get nodes
NAME       STATUS   ROLES    AGE   VERSION
minikube   Ready    master   76s   v1.15.0
$ kubectl get deployments
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp   1/1     1            1           82s
$ kubectl get services
NAME                  TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
kubernetes            ClusterIP   10.96.0.1       <none>        443/TCP          88s
kubernetes-bootcamp   NodePort    10.111.171.95   <none>        8080:32466/TCP   86s
$
```

Lab for Step 5.1: Scaling a deployment

Confirm you deployments by using the **$ kubectl get deployments** *(see output above).* This shows we have once 1 Pod with the Ready column sure the ratio of ***Current*** *to* ***Desired*** *replicas. It is important in large environments to monitor that CURRENT is the number of replicas running now, and DESIRED will be the configured of number of replicas that should be running.* Likewise, UP-TO-DATE is the number of replicas updated to match the desired (configured) state; and AVAILABLE state shows how many replicas are actually AVAILABLE to the users.

https://kubernetes.io/docs/tutorials/kubernetes-basics/

Lab for Lab 5.1 continues

Now, run the following commands:

$ kubectl get deployments
$ kubectl scale deployments/kubernetes-bootcamp –replicas=4
$ kubectl get deployments
$ kubectl get pods –o wide

Here is a perfect result – you can see the interim stages

- READY = 1/4 right after the scale command; then
- 2 Pods are ready (1/1) with the Status of Running, with an IPv4 address each; but the other
- 2 Pods are not ready (0/1) with the Status of ContainerCreating, and <none> as the IPv4 value
- Finally a subsequent **$ kubectl get deployments** shows READY is 4/4, Update is 4 and AVAILABLE is 4

```
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp   1/1     1            1           9m9s
$ kubectl scale deployments/kubernetes-bootcamp --replicas=4
deployment.extensions/kubernetes-bootcamp scaled
$ kubectl get deployments
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp   1/4     4            1           9m12s
$ kubectl get pods -o wide
NAME                                  READY   STATUS              RESTARTS   AGE    IP           NODE       NOMINATED NODE
READINESS GATES
kubernetes-bootcamp-5b48cfdcbd-79n7x  1/1     Running             0          2s     172.18.0.6   minikube   <none>
<none>
kubernetes-bootcamp-5b48cfdcbd-dhjm8  1/1     Running             0          9m7s   172.18.0.4   minikube   <none>
<none>
kubernetes-bootcamp-5b48cfdcbd-hpmfb  0/1     ContainerCreating   0          2s     <none>       minikube   <none>
<none>
kubernetes-bootcamp-5b48cfdcbd-vslzm  0/1     ContainerCreating   0          2s     <none>       minikube   <none>
<none>
$ kubectl get deployments
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp   4/4     4            4           9m17s
```

For good meansure, I ran the **$ kubectl get pods –o wide** which shows the correct READY ratio and a valid IPv4 address filled in

```
$ kubectl get pods -o wide
NAME                                  READY   STATUS    RESTARTS   AGE     IP           NODE       NOMINATED NODE   READINESS
 GATES
kubernetes-bootcamp-5b48cfdcbd-79n7x  1/1     Running   0          5m23s   172.18.0.6   minikube   <none>           <none>
kubernetes-bootcamp-5b48cfdcbd-dhjm8  1/1     Running   0          14m     172.18.0.4   minikube   <none>           <none>
kubernetes-bootcamp-5b48cfdcbd-hpmfb  1/1     Running   0          5m23s   172.18.0.7   minikube   <none>           <none>
kubernetes-bootcamp-5b48cfdcbd-vslzm  1/1     Running   0          5m23s   172.18.0.8   minikube   <none>           <none>
$
```

There are 4 Pods now, with different IP addresses. The change was registered in the Deployment events log. To check that, use the describe command:

```
kubectl describe deployments/kubernetes-bootcamp
```

https://kubernetes.io/docs/tutorials/kubernetes-basics/

```
$ kubectl describe deployments/kubernetes-bootcamp | more
Name:                   kubernetes-bootcamp
Namespace:              default
CreationTimestamp:      Fri, 17 Jan 2020 04:56:11 +0000
Labels:                 run=kubernetes-bootcamp
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               run=kubernetes-bootcamp
Replicas:               4 desired | 4 updated | 4 total | 4 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  run=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
  ----           ------  ------
  Progressing    True    NewReplicaSetAvailable
  Available      True    MinimumReplicasAvailable
OldReplicaSets:  <none>
NewReplicaSet:   kubernetes-bootcamp-5b48cfdcbd (4/4 replicas created)
Events:
  Type    Reason             Age   From                   Message
  ----    ------             ----  ----                   -------
  Normal  ScalingReplicaSet  17m   deployment-controller  Scaled up replica set kubernetes-bootcamp-5b48cfdcbd to 1
  Normal  ScalingReplicaSet  8m7s  deployment-controller  Scaled up replica set kubernetes-bootcamp-5b48cfdcbd to 4
$
```

You can also view in the output of this command that there are 4 replicas now.

```
Replicas:               4 desired | 4 updated | 4 total | 4 available | 0 unavailable
```

https://kubernetes.io/docs/tutorials/kubernetes-basics/

Lab for Step 5.2: Load Balancing

Let's verify whether the Service is load-balancing the traffic like it showed for Service A in the Module 5 diagram. Once again, use the same describe command, but let's look for the "exposed IP" and "exposed Port", but this time use **services/kubernetes-bootcamp** instead of **deployments/kubernetes-bootcamp**.

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

```
$ kubectl get services
NAME                  TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
kubernetes            ClusterIP   10.96.0.1       <none>        443/TCP          23m
kubernetes-bootcamp   NodePort    10.111.171.95   <none>        8080:32466/TCP   23m
$ kubectl describe services/kubernetes-bootcamp
Name:                     kubernetes-bootcamp
Namespace:                default
Labels:                   run=kubernetes-bootcamp
Annotations:              <none>
Selector:                 run=kubernetes-bootcamp
Type:                     NodePort
IP:                       10.111.171.95
Port:                     <unset>  8080/TCP
TargetPort:               8080/TCP
NodePort:                 <unset>  32466/TCP
Endpoints:                172.18.0.4:8080,172.18.0.6:8080,172.18.0.7:8080 + 1 more...
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
$
```

As before, let's create an environment variable called NODE_PORT for the value of NODE_PORT above so we can use curl to confirm the exposed IP and port is working (run curl command multiple times)

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

```
$ export NODE_PORT=$(kubectl get services/kubernetes-bootcamp -o go-template='{{(index .spec.ports 0).nodePort}}')
$ echo NODE_PORT=$NODE_PORT
NODE_PORT=32466
$ curl $(minikube ip):$NODE_PORT
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-5b48cfdcbd-79n7x | v=1
$ curl $(minikube ip):$NODE_PORT
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-5b48cfdcbd-vslzm | v=1
$ curl $(minikube ip):$NODE_PORT
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-5b48cfdcbd-vslzm | v=1
$ curl $(minikube ip):$NODE_PORT
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-5b48cfdcbd-hpmfb | v=1
$ curl $(minikube ip):$NODE_PORT
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-5b48cfdcbd-hpmfb | v=1
$ curl $(minikube ip):$NODE_PORT
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-5b48cfdcbd-hpmfb | v=1
```

We can see, we have 3 different Pods values for the six times we ran the **`$ curl $minikube ip):$NODE_PORT`** command. It isn't exactly round-robin (79n7x = 16%, vs1zm = 34% and hpmfb = 50%), but is is load balancing.

https://kubernetes.io/docs/tutorials/kubernetes-basics/

Lab for Step 5.3: Scale Down

While the peaks and valleys of network traffic from the Internet will normally shape your Kubenetes deployment, we can demonstrate by manually **scaling down the cluster.**

Let's alter the cluster by reducing the deployment by 50 percent (going from 4 replicas to 2 replicas).

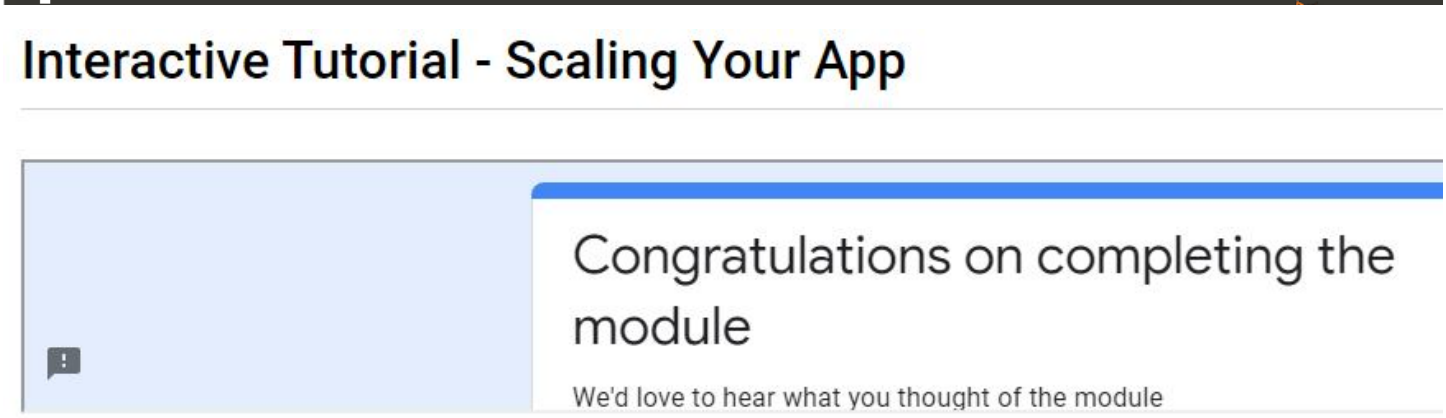
You accomplish this by using the same exact SCALE parameter, but use **replicas=2** instead of **replicas=4**

$ kubectl scale deployments/kubernetes-bootcamp –replicas=2

```
$ kubectl get deployments
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp   4/4     4            4           36m
$ kubectl scale deployments/kubernetes-bootcamp --replicas=2
deployment.extensions/kubernetes-bootcamp scaled
$ kubectl get deployments
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp   2/2     2            2           36m
$ kubectl get pods -o wide
NAME                                  READY   STATUS        RESTARTS   AGE   IP           NODE       NOMINATED NODE   READINE
SS GATES
kubernetes-bootcamp-5b48cfdcbd-79n7x  1/1     Running       0          27m   172.18.0.6   minikube   <none>           <none>
kubernetes-bootcamp-5b48cfdcbd-dhjm8  1/1     Running       0          36m   172.18.0.4   minikube   <none>           <none>
kubernetes-bootcamp-5b48cfdcbd-hpmfb  1/1     Terminating   0          27m   172.18.0.7   minikube   <none>           <none>
kubernetes-bootcamp-5b48cfdcbd-vslzm  1/1     Terminating   0          27m   172.18.0.8   minikube   <none>           <none>
$
```

And now we run the get pods to see final state of deployment – the 2 pods that were terminating are truly destroyed.

```
$ kubectl get pods -o wide
NAME                                  READY   STATUS    RESTARTS   AGE   IP           NODE       NOMINATED NODE   READINESS G
ATES
kubernetes-bootcamp-5b48cfdcbd-79n7x  1/1     Running   0          28m   172.18.0.6   minikube   <none>           <none>
kubernetes-bootcamp-5b48cfdcbd-dhjm8  1/1     Running   0          37m   172.18.0.4   minikube   <none>           <none>
$
```



https://kubernetes.io/docs/tutorials/kubernetes-basics/

Module 6 – Performing a Rolling Update

Objectives: **Perform a rolling update using kubectl.**

https://kubernetes.io/docs/tutorials/kubernetes-basics/scale/scale-intro/

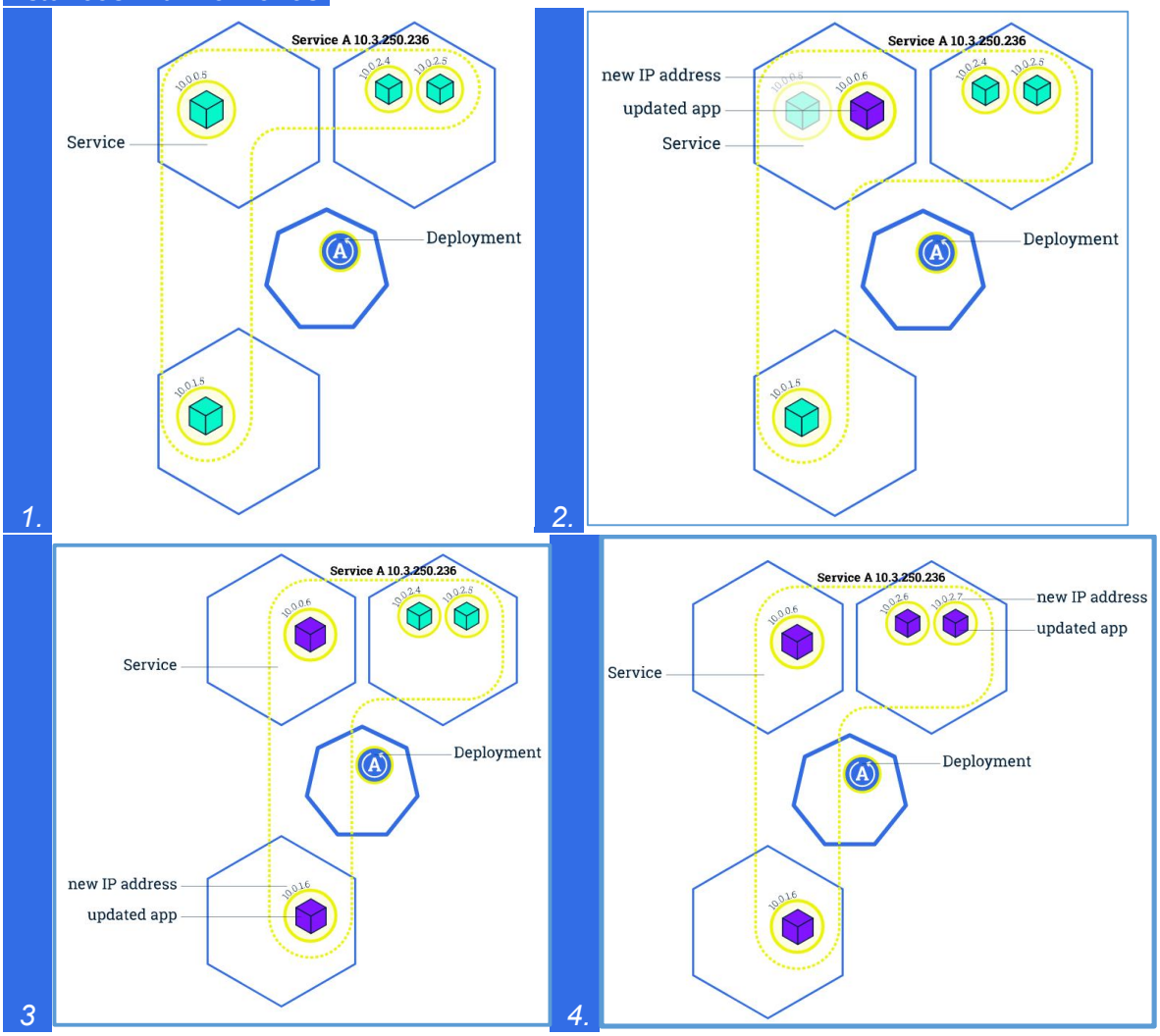
Updating an application

Users expect applications to be available all the time and developers are expected to deploy new versions of them several times a day. In Kubernetes this is done with rolling updates. **Rolling updates** allow Deployments' update to take place with zero downtime by incrementally updating Pods instances with new ones. The new Pods will be scheduled on Nodes with available resources.

In the previous module we scaled our application to run multiple instances. **This is a requirement for performing updates without affecting application availability**. By default, the maximum number of Pods that can be unavailable during the update and the maximum number of new Pods that can be created, is one. Both options can be configured to either numbers or percentages (of Pods). In Kubernetes, updates are versioned and *any Deployment update can be reverted to previous (stable) version*.

Rolling updates overview

Rolling updates allow Deployments' update to take place with zero downtime by incrementally updating Pods instances with new ones.



https://kubernetes.io/docs/tutorials/kubernetes-basics/

Similar to application Scaling, if a Deployment is exposed publicly, the Service will load-balance the traffic only to available Pods during the update. An available Pod is an instance that is available to the users of the application.

Rolling updates allow the following actions:

- Promote an application from one environment to another (via container image updates)
- Rollback to previous versions
- Continuous Integration and Continuous Delivery of applications with zero downtime

If a Deployment is exposed publicly, the Service will load-balance the traffic only to available Pods during the update.

In the following interactive tutorial, we'll update our application to a new version, and also perform a rollback.

https://kubernetes.io/docs/tutorials/kubernetes-basics/

Lab 6 “Updating Your App” as companion to the Module 6 discussion

The goal of this scenario is to update a deployed application with kubectl set image and to rollback with the rollout undo command.

The online terminal is a pre-configured Linux environment that can be used as a regular console (you can type commands). Clicking on the blocks of code followed by the ENTER sign will execute that command in the terminal.

```
Kubernetes Bootcamp Terminal

$
$ sleep 1; launch.sh
Starting Kubernetes. This is expected to take less than a minute
Kubernetes Started
$
```

Before any updated, you want to be sure you know your deployments, the running Pods and can confirm the current image version of the application running .

Now, run the kubectl commands to get this information before you update your app:

$ kubectl get deployments

$ kubectl get pods

```
$ kubectl get deployments
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp   4/4     4            4           4m22s
$ kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-5b48cfdcbd-tnq5x   1/1     Running   0          4m15s
kubernetes-bootcamp-5b48cfdcbd-xb5p2   1/1     Running   0          4m15s
kubernetes-bootcamp-5b48cfdcbd-xtqbb   1/1     Running   0          4m15s
kubernetes-bootcamp-5b48cfdcbd-z4wv5   1/1     Running   0          4m15s
```

$ kubectl describe pods

```
$ kubectl describe pods
Name:           kubernetes-bootcamp-5b48cfdcbd-tnq5x
Namespace:      default
Priority:       0
Node:           minikube/172.17.0.7
Start Time:     Fri, 17 Jan 2020 05:48:59 +0000
Labels:         pod-template-hash=5b48cfdcbd
```

https://kubernetes.io/docs/tutorials/kubernetes-basics/

```
IP:             172.18.0.2
Controlled By:  ReplicaSet/kubernetes-bootcamp-5b48cfdcbd
Containers:
  kubernetes-bootcamp:
    Container ID:   docker://14f310d3c25505395bef10d91cb48e5dcb5a15973101e83c3fa26e0b65c95cb8
    Image:          gcr.io/google-samples/kubernetes-bootcamp:v1
    Image ID:       docker-pullable://jocatalin/kubernetes-bootcamp@sha256:0d6b8ee63bb57c5f5b6156f446b3bc3b3c143d233037f3a2f0
0e279c8fcc64af
    Port:           8080/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Fri, 17 Jan 2020 05:49:02 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-l89lh (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
```

```
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-l89lh
    Optional:    false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute for 300s
                 node.kubernetes.io/unreachable:NoExecute for 300s
Events:
  Type    Reason     Age    From               Message
  ----    ------     ----   ----               -------
  Normal  Scheduled  4m14s  default-scheduler  Successfully assigned default/kubernetes-bootcamp-5b48cfdcbd-tnq5x to minikube
  Normal  Pulled     4m12s  kubelet, minikube  Container image "gcr.io/google-samples/kubernetes-bootcamp:v1" already present
 on machine
  Normal  Created    4m12s  kubelet, minikube  Created container kubernetes-bootcamp
  Normal  Started    4m11s  kubelet, minikube  Started container kubernetes-bootcamp


Name:           kubernetes-bootcamp-5b48cfdcbd-xb5p2
Namespace:      default
Priority:       0
Node:           minikube/172.17.0.7
Start Time:     Fri, 17 Jan 2020 05:48:59 +0000
```

```
Status:         Running
IP:             172.18.0.5
Controlled By:  ReplicaSet/kubernetes-bootcamp-5b48cfdcbd
Containers:
  kubernetes-bootcamp:
    Container ID:   docker://983efe4995ea7f60d4c212b711424b9dbe77c0bb92de4eaef16fc0543d8f6ee8
    Image:          gcr.io/google-samples/kubernetes-bootcamp:v1
    Image ID:       docker-pullable://jocatalin/kubernetes-bootcamp@sha256:0d6b8ee63bb57c5f5b6156f446b3bc3b3c143d233037f3a2f0
0e279c8fcc64af
    Port:           8080/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Fri, 17 Jan 2020 05:49:02 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-l89lh (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
```

https://kubernetes.io/docs/tutorials/kubernetes-basics/

```
  default-token-189lh:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-189lh
    Optional:    false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute for 300s
                 node.kubernetes.io/unreachable:NoExecute for 300s
Events:
  Type    Reason     Age    From               Message
  ----    ------     ----   ----               -------
  Normal  Scheduled  4m14s  default-scheduler  Successfully assigned default/kubernetes-bootcamp-5b48cfdcbd-xb5p2 to minikube
  Normal  Pulled     4m11s  kubelet, minikube  Container image "gcr.io/google-samples/kubernetes-bootcamp:v1" already present
 on machine
  Normal  Created    4m11s  kubelet, minikube  Created container kubernetes-bootcamp
  Normal  Started    4m11s  kubelet, minikube  Started container kubernetes-bootcamp


Name:           kubernetes-bootcamp-5b48cfdcbd-xtqbb
Namespace:      default
Priority:       0
Node:           minikube/172.17.0.7
```

```
Annotations:    <none>
Status:         Running
IP:             172.18.0.6
Controlled By:  ReplicaSet/kubernetes-bootcamp-5b48cfdcbd
Containers:
  kubernetes-bootcamp:
    Container ID:   docker://dff7d0dd55724475957383f08f293da71933df92fd6f89877c55285273e8a7db
    Image:          gcr.io/google-samples/kubernetes-bootcamp:v1
    Image ID:       docker-pullable://jocatalin/kubernetes-bootcamp@sha256:0d6b8ee63bb57c5f5b6156f446b3bc3b3c143d233037f3a2f0
0e279c8fcc64af
    Port:           8080/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Fri, 17 Jan 2020 05:49:02 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-189lh (ro)
Conditions:
  Type              Status
  Initialized       True
```

```
Volumes:
  default-token-189lh:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-189lh
    Optional:    false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute for 300s
                 node.kubernetes.io/unreachable:NoExecute for 300s
Events:
  Type    Reason     Age    From               Message
  ----    ------     ----   ----               -------
  Normal  Scheduled  4m14s  default-scheduler  Successfully assigned default/kubernetes-bootcamp-5b48cfdcbd-xtqbb to minikube
  Normal  Pulled     4m12s  kubelet, minikube  Container image "gcr.io/google-samples/kubernetes-bootcamp:v1" already present
 on machine
  Normal  Created    4m12s  kubelet, minikube  Created container kubernetes-bootcamp
  Normal  Started    4m11s  kubelet, minikube  Started container kubernetes-bootcamp


Name:           kubernetes-bootcamp-5b48cfdcbd-z4wv5
Namespace:      default
Priority:       0
```

https://kubernetes.io/docs/tutorials/kubernetes-basics/

```
                run=kubernetes-bootcamp
Annotations:    <none>
Status:         Running
IP:             172.18.0.4
Controlled By:  ReplicaSet/kubernetes-bootcamp-5b48cfdcbd
Containers:
  kubernetes-bootcamp:
    Container ID:   docker://828de037ce548641a8be6f5cc56351011afa2f38d5b6a06f6d22c9beede9e92b
    Image:          gcr.io/google-samples/kubernetes-bootcamp:v1
    Image ID:       docker-pullable://jocatalin/kubernetes-bootcamp@sha256:0d6b8ee63bb57c5f5b6156f446b3bc3b3c143d233037f3a2f0
0e279c8fcc64af
    Port:           8080/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Fri, 17 Jan 2020 05:49:02 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-l89lh (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  default-token-l89lh:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-l89lh
    Optional:    false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute for 300s
                 node.kubernetes.io/unreachable:NoExecute for 300s
Events:
  Type    Reason     Age    From               Message
  ----    ------     ----   ----               -------
  Normal  Scheduled  4m14s  default-scheduler  Successfully assigned default/kubernetes-bootcamp-5b48cfdcbd-z4wv5 to minikube
  Normal  Pulled     4m12s  kubelet, minikube  Container image "gcr.io/google-samples/kubernetes-bootcamp:v1" already present
 on machine
  Normal  Created    4m11s  kubelet, minikube  Created container kubernetes-bootcamp
  Normal  Started    4m11s  kubelet, minikube  Started container kubernetes-bootcamp
$
```

https://kubernetes.io/docs/tutorials/kubernetes-basics/

Lab for Step 6.1: Update Your App

Next, to update your app, you will need to update the application to version 2, which you can do with the **set image** command followed by the deployment name and the new image version you wish to deploy.

$ kubectl get deployment
$ kubectl set image deployments/kubernetes-bootcamp kubernetes-bootcamp=jocatalin/kubernetes-bootcamp:v2

```
$ kubectl get deployment
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp   4/4     4            4           12m
$ kubectl set image deployments/kubernetes-bootcamp kubernetes-bootcamp=jocatalin/kubernetes-bootcamp:v2
deployment.extensions/kubernetes-bootcamp image updated
```

$ kubectl get pods

```
$ kubectl get pods
NAME                                  READY   STATUS        RESTARTS   AGE
kubernetes-bootcamp-5b48cfdcbd-tnq5x  1/1     Terminating   0          12m
kubernetes-bootcamp-5b48cfdcbd-xb5p2  1/1     Terminating   0          12m
kubernetes-bootcamp-5b48cfdcbd-xtqbb  1/1     Terminating   0          12m
kubernetes-bootcamp-5b48cfdcbd-z4wv5  1/1     Terminating   0          12m
kubernetes-bootcamp-cfc74666-2hkl2    1/1     Running       0          10s
kubernetes-bootcamp-cfc74666-lrw97    1/1     Running       0          9s
kubernetes-bootcamp-cfc74666-tjxzc    1/1     Running       0          10s
kubernetes-bootcamp-cfc74666-w6vhl    1/1     Running       0          8s
$
```

Wait, then check again

```
$ kubectl get deployment
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp   4/4     4            4           13m
$ kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-cfc74666-2hkl2    1/1     Running   0          76s
kubernetes-bootcamp-cfc74666-lrw97    1/1     Running   0          75s
kubernetes-bootcamp-cfc74666-tjxzc    1/1     Running   0          76s
kubernetes-bootcamp-cfc74666-w6vhl    1/1     Running   0          74s
$
```

$ kubectl get pods –o wide

```
$ kubectl get pods -o wide
NAME                                  READY   STATUS    RESTARTS   AGE    IP            NODE       NOMINATED NODE   READINESS
GATES
kubernetes-bootcamp-cfc74666-2hkl2    1/1     Running   0          113s   172.18.0.9    minikube   <none>           <none>
kubernetes-bootcamp-cfc74666-lrw97    1/1     Running   0          112s   172.18.0.11   minikube   <none>           <none>
kubernetes-bootcamp-cfc74666-tjxzc    1/1     Running   0          113s   172.18.0.10   minikube   <none>           <none>
kubernetes-bootcamp-cfc74666-w6vhl    1/1     Running   0          111s   172.18.0.12   minikube   <none>           <none>
$
```

https://kubernetes.io/docs/tutorials/kubernetes-basics/

Lab for Step 6.2: Verify an Update of Your App

So, now let's check that the app is running and find out the exported IP and exposed Port, we'll start with the **describe service** command once again:

```
$ kubectl get deployments
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp   4/4     4            4           17m
$ kubectl describe services/kubernetes-bootcamp | more
Name:                     kubernetes-bootcamp
Namespace:                default
Labels:                   run=kubernetes-bootcamp
Annotations:              <none>
Selector:                 run=kubernetes-bootcamp
Type:                     NodePort
IP:                       10.106.128.13
Port:                     <unset>  8080/TCP
TargetPort:               8080/TCP
NodePort:                 <unset>  30031/TCP
Endpoints:                172.18.0.10:8080,172.18.0.11:8080,172.18.0.12:8080 + 1 more...
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
$
```

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
```

Next, we'll do a curl to the the exposed IP and port:

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

We hit a different Pod with every request and we see that all Pods are running the latest version (v2).

```
$ export NODE_PORT=$(kubectl get services/kubernetes-bootcamp -o go-template='{{(index .spec.ports 0).nodePort}}')
$ echo NODE_PORT=$NODE_PORT
NODE_PORT=30031
$ curl $(minikube ip):$NODE_PORT
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-cfc74666-w6vhl | v=2
$ curl $(minikube ip):$NODE_PORT
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-cfc74666-2hkl2 | v=2
$ curl $(minikube ip):$NODE_PORT
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-cfc74666-tjxzc | v=2
$ curl $(minikube ip):$NODE_PORT
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-cfc74666-lrw97 | v=2
$
```

https://kubernetes.io/docs/tutorials/kubernetes-basics/

Lab for Step 6.2: Rollback an Update of Your App

This time, let's perform another update, and deploy and image tagged as v10. As before, we'll use the **get deployments** command to confirm the state of the deployment before and update the update.

```
$ kubectl get deployments
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp   4/4     4            4           22m
$ kubectl set image deployments/kubernetes-bootcamp kubernetes-bootcamp=gcr.io/google-samples/kubernetes-bootcamp:v10
deployment.extensions/kubernetes-bootcamp image updated
$ kubectl get deployments
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp   3/4     2            3           22m
$ kubectl get deployments
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp   3/4     2            3           22m
```

Something went wrong. Let's check the pods with both **get pods** and **describe pods** to get a better look at what's going on.

```
$ kubectl get deployments
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp   3/4     2            3           24m
$ kubectl get pods
NAME                                  READY   STATUS             RESTARTS   AGE
kubernetes-bootcamp-547469f5dd-6rf59  0/1     ErrImagePull       0          100s
kubernetes-bootcamp-547469f5dd-8x26j  0/1     ImagePullBackOff   0          100s
kubernetes-bootcamp-cfc74666-2hkl2    1/1     Running            0          11m
kubernetes-bootcamp-cfc74666-lrw97    1/1     Running            0          11m
kubernetes-bootcamp-cfc74666-tjxzc    1/1     Running            0          11m
$
```

$ kubectl describe pods | more

```
$ kubectl describe pods | more
Name:           kubernetes-bootcamp-547469f5dd-6rf59
Namespace:      default
Priority:       0
Node:           minikube/172.17.0.7
Start Time:     Fri, 17 Jan 2020 06:11:36 +0000
Labels:         pod-template-hash=547469f5dd
                run=kubernetes-bootcamp
Annotations:    <none>
Status:         Pending
IP:             172.18.0.4
Controlled By:  ReplicaSet/kubernetes-bootcamp-547469f5dd
Containers:
  kubernetes-bootcamp:
    Container ID:
    Image:          gcr.io/google-samples/kubernetes-bootcamp:v10
    Image ID:
    Port:           8080/TCP
    Host Port:      0/TCP
    State:          Waiting
      Reason:       ImagePullBackOff
    Ready:          False
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-189lh (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             False
  ContainersReady   False
  PodScheduled      True
Volumes:
  default-token-189lh:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-189lh
    Optional:    false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute for 300s
                 node.kubernetes.io/unreachable:NoExecute for 300s
Events:
  Type     Reason     Age                From               Message
```

https://kubernetes.io/docs/tutorials/kubernetes-basics/

```
  Normal   Scheduled  2m41s                default-scheduler  Successfully assigned default/kubernetes-bootcamp-547469f5dd-6r
f59 to minikube
  Normal   Pulling    81s (x4 over 2m40s)  kubelet, minikube  Pulling image "gcr.io/google-samples/kubernetes-bootcamp:v10"
  Warning  Failed     81s (x4 over 2m39s)  kubelet, minikube  Failed to pull image "gcr.io/google-samples/kubernetes-bootcamp
:v10": rpc error: code = Unknown desc = Error response from daemon: manifest for gcr.io/google-samples/kubernetes-bootcamp:v1
0 not found
  Warning  Failed     81s (x4 over 2m39s)  kubelet, minikube  Error: ErrImagePull
  Normal   BackOff    53s (x6 over 2m39s)  kubelet, minikube  Back-off pulling image "gcr.io/google-samples/kubernetes-bootca
mp:v10"
  Warning  Failed     42s (x7 over 2m39s)  kubelet, minikube  Error: ImagePullBackOff
```

Above – the events tell us a lot – and we know why we have 2 pods (original and new) in a bad state

```
Name:           kubernetes-bootcamp-547469f5dd-8x26j
Namespace:      default
Priority:       0
Node:           minikube/172.17.0.7
Start Time:     Fri, 17 Jan 2020 06:11:36 +0000
Labels:         pod-template-hash=547469f5dd
                run=kubernetes-bootcamp
Annotations:    <none>
Status:         Pending
IP:             172.18.0.2
Controlled By:  ReplicaSet/kubernetes-bootcamp-547469f5dd
Containers:
  kubernetes-bootcamp:
    Container ID:
    Image:          gcr.io/google-samples/kubernetes-bootcamp:v10
    Image ID:
    Port:           8080/TCP
    Host Port:      0/TCP
    State:          Waiting
      Reason:       ImagePullBackOff
    Ready:          False
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-l89lh (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             False
  ContainersReady   False
  PodScheduled      True
Volumes:
  default-token-l89lh:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-l89lh
    Optional:    false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute for 300s
                 node.kubernetes.io/unreachable:NoExecute for 300s
Events:
```

https://kubernetes.io/docs/tutorials/kubernetes-basics/

```
Events:
  Type     Reason     Age                From               Message
  ----     ------     ----               ----               -------
  Normal   Scheduled  2m41s              default-scheduler  Successfully assigned default/kubernetes-bootcamp-547469f5dd-8x
26j to minikube
  Normal   Pulling    69s (x4 over 2m40s)  kubelet, minikube  Pulling image "gcr.io/google-samples/kubernetes-bootcamp:v10"
  Warning  Failed     69s (x4 over 2m40s)  kubelet, minikube  Failed to pull image "gcr.io/google-samples/kubernetes-bootcamp
:v10": rpc error: code = Unknown desc = Error response from daemon: manifest for gcr.io/google-samples/kubernetes-bootcamp:v1
0 not found
  Warning  Failed     69s (x4 over 2m40s)  kubelet, minikube  Error: ErrImagePull
  Normal   BackOff    56s (x6 over 2m39s)  kubelet, minikube  Back-off pulling image "gcr.io/google-samples/kubernetes-bootca
mp:v10"
  Warning  Failed     41s (x7 over 2m39s)  kubelet, minikube  Error: ImagePullBackOff


Name:           kubernetes-bootcamp-cfc74666-2hkl2
Namespace:      default
Priority:       0
Node:           minikube/172.17.0.7
Start Time:     Fri, 17 Jan 2020 06:01:24 +0000
Labels:         pod-template-hash=cfc74666
                run=kubernetes-bootcamp
Annotations:    <none>
Status:         Running
IP:             172.18.0.9
Controlled By:  ReplicaSet/kubernetes-bootcamp-cfc74666
Containers:
  kubernetes-bootcamp:
    Container ID:   docker://59dff6b522615e446e3a5d31107de1302da523653a2ad2fad1c5f226edce28df
    Image:          jocatalin/kubernetes-bootcamp:v2
    Image ID:       docker-pullable://jocatalin/kubernetes-bootcamp@sha256:fb1a3ced00cecfc1f83f18ab5cd14199e30adc1b49aa4244f5
d65ad3f5feb2a5
    Port:           8080/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Fri, 17 Jan 2020 06:01:25 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-l89lh (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  default-token-l89lh:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-l89lh
    Optional:    false
```

.....

https://kubernetes.io/docs/tutorials/kubernetes-basics/

```
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  default-token-189lh:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-189lh
    Optional:    false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute for 300s
                 node.kubernetes.io/unreachable:NoExecute for 300s
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  19m   default-scheduler  Successfully assigned default/kubernetes-bootcamp-cfc74666-tjxzc to minikube
  Normal  Pulled     19m   kubelet, minikube  Container image "jocatalin/kubernetes-bootcamp:v2" already present on machine
  Normal  Created    19m   kubelet, minikube  Created container kubernetes-bootcamp
  Normal  Started    19m   kubelet, minikube  Started container kubernetes-bootcamp
$
```

Use “rollback undo” command

```
$  kubectl get pods
NAME                                   READY   STATUS             RESTARTS   AGE
kubernetes-bootcamp-547469f5dd-6rf59   0/1     ImagePullBackOff   0          11m
kubernetes-bootcamp-547469f5dd-8x26j   0/1     ErrImagePull       0          11m
kubernetes-bootcamp-cfc74666-2hkl2     1/1     Running            0          21m
kubernetes-bootcamp-cfc74666-lrw97     1/1     Running            0          21m
kubernetes-bootcamp-cfc74666-tjxzc     1/1     Running            0          21m
$ kubectl rollout undo deployments/kubernetes-bootcamp
deployment.extensions/kubernetes-bootcamp rolled back
$ kubectl get pods
NAME                                   READY   STATUS              RESTARTS   AGE
kubernetes-bootcamp-547469f5dd-6rf59   0/1     Terminating         0          11m
kubernetes-bootcamp-547469f5dd-8x26j   0/1     Terminating         0          11m
kubernetes-bootcamp-cfc74666-2hkl2     1/1     Running             0          21m
kubernetes-bootcamp-cfc74666-lrw97     1/1     Running             0          21m
kubernetes-bootcamp-cfc74666-sfjpm     0/1     ContainerCreating   0          3s
kubernetes-bootcamp-cfc74666-tjxzc     1/1     Running             0          21m
$
```

Wait and check again to give the Terminating Pods time to clear, and the new Container to be created

```
$ kubectl get pods
NAME                                   READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-cfc74666-2hkl2     1/1     Running   0          22m
kubernetes-bootcamp-cfc74666-lrw97     1/1     Running   0          22m
kubernetes-bootcamp-cfc74666-sfjpm     1/1     Running   0          47s
kubernetes-bootcamp-cfc74666-tjxzc     1/1     Running   0          22m
$
```

https://kubernetes.io/docs/tutorials/kubernetes-basics/

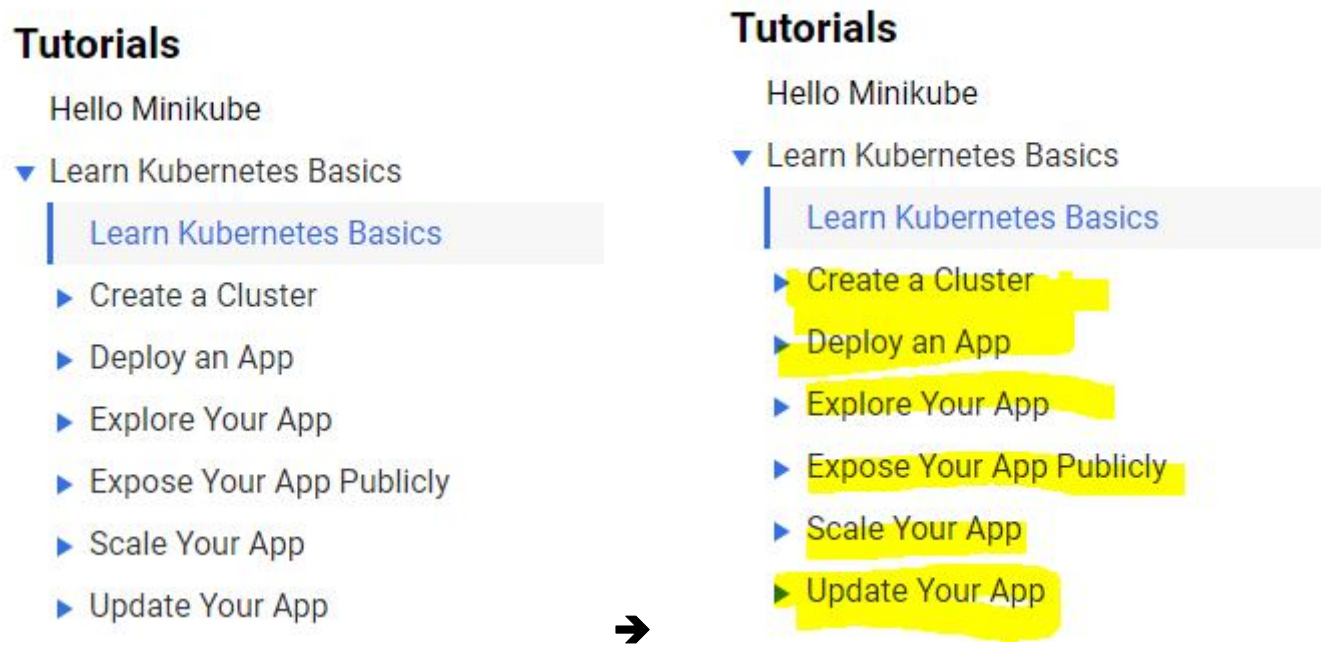
Four Pods are running. Check again the image deployed on them:

```
kubectl describe pods
```

```
...
$ kubectl describe pods | grep v2
    Image:          jocatalin/kubernetes-bootcamp:v2
  Normal  Pulled     25m   kubelet, minikube  Container image "jocatalin/kubernetes-bootcamp:v2" already present on machine
    Image:          jocatalin/kubernetes-bootcamp:v2
  Normal  Pulled     25m   kubelet, minikube  Container image "jocatalin/kubernetes-bootcamp:v2" already present on machine
    Image:          jocatalin/kubernetes-bootcamp:v2
  Normal  Pulled     3m36s  kubelet, minikube  Container image "jocatalin/kubernetes-bootcamp:v2" already present on machine
    Image:          jocatalin/kubernetes-bootcamp:v2
  Normal  Pulled     25m   kubelet, minikube  Container image "jocatalin/kubernetes-bootcamp:v2" already present on machine
$
```

We see that the deployment is using a stable version of the app (v2). The Rollback was successful.

SUCCESS – ALL OBJECTIVES MET



https://kubernetes.io/docs/tutorials/kubernetes-basics/

APPENDIX – Notes during Module 2

What part of the API Server is exposed?

NOTE: Finding out the allowed "paths" of the API is easy – type anything that will not be normal to be there (like 'curl')

```
$ curl http://localhost:8001/curl
{
  "paths": [
    "/apis",
    "/apis/",
    "/apis/apiextensions.k8s.io",
    "/apis/apiextensions.k8s.io/v1beta1",
    "/healthz",
    "/healthz/etcd",
    "/healthz/log",
    "/healthz/ping",
    "/healthz/poststarthook/crd-informer-synced",
    "/healthz/poststarthook/generic-apiserver-start-informers",
    "/healthz/poststarthook/start-apiextensions-controllers",
    "/healthz/poststarthook/start-apiextensions-informers",
    "/metrics",
    "/openapi/v2",
    "/version"
  ]
}$
```

You can see why /version provided the earlier output.

```
$ curl http://localhost:8001/healthz/etcd
ok$
$
```

Some output, like /metrics will provide more output than can be captured here, but you can pipe that output in to grep (e.g., " curl http://localhost:8001/metrica | grep etcd | wc –l " and you can see there is a lot of output still…

```
$ curl http://localhost:8001/metrics | grep etcd | wc -l
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 2557k    0 2557k    0     0  26.2M      0 --:--:-- --:--:-- --:--:-- 26.2M
2827
$
```

https://kubernetes.io/docs/tutorials/kubernetes-basics/

https://kubernetes.io/docs/tutorials/kubernetes-basics/

Appendix – Notes on Module 3

Enter container with Bash and exit

```
$ kubectl exec -ti $POD_NAME bash
root@kubernetes-bootcamp-5b48cfdcbd-vnnzx:/# ls
bin  boot  core  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  server.js  srv  sys  tmp  usr  var
root@kubernetes-bootcamp-5b48cfdcbd-vnnzx:/# exit
exit
$
```

```
$ kubectl logs $POD_NAME
Kubernetes Bootcamp App Started At: 2020-01-14T03:32:34.740Z | Running On:  kubernetes-bootcamp-5b48cfdcbd-vnnzx

Running On: kubernetes-bootcamp-5b48cfdcbd-vnnzx | Total Requests: 1 | App Uptime: 1361.899 seconds | Log Time: 2020-01-14T03:55:16.639Z
Running On: kubernetes-bootcamp-5b48cfdcbd-vnnzx | Total Requests: 2 | App Uptime: 2234.447 seconds | Log Time: 2020-01-14T04:09:49.187Z
$ kubectl exec -ti $POD_NAME bash
root@kubernetes-bootcamp-5b48cfdcbd-vnnzx:/# ls
bin  boot  core  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  server.js  srv  sys  tmp  usr  var
root@kubernetes-bootcamp-5b48cfdcbd-vnnzx:/# exit
exit
$
```

```
root@kubernetes-bootcamp-5b48cfdcbd-vnnzx:/# cat server.js
var http = require('http');
var requests=0;
var podname= process.env.HOSTNAME;
var startTime;
var host;
var handleRequest = function(request, response) {
  response.setHeader('Content-Type', 'text/plain');
  response.writeHead(200);
  response.write("Hello Kubernetes bootcamp! | Running on: ");
  response.write(host);
  response.end(" | v=1\n");
  console.log("Running On:" ,host, "| Total Requests:", ++requests,"| App Uptime:", (new Date() - startTime)/1000 , "seconds", "| Log Ti
me:",new Date());
}
var www = http.createServer(handleRequest);
www.listen(8080,function () {
    startTime = new Date();;
    host = process.env.HOSTNAME;
    console.log ("Kubernetes Bootcamp App Started At:",startTime, "| Running On: " ,host, "\n" );
});
root@kubernetes-bootcamp-5b48cfdcbd-vnnzx:/#
```

```
root@kubernetes-bootcamp-5b48cfdcbd-vnnzx:/# curl localhost:8080
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-5b48cfdcbd-vnnzx | v=1
root@kubernetes-bootcamp-5b48cfdcbd-vnnzx:/#
```