You make **possible**

Deploying a Cloud Native App

Using Docker and Kubernetes

Hector Morales
@ekktor

LTRPRG-1200

Cisco live!
June 9-13, 2019 • San Diego, CA
#CLUS

Agenda

- What is Kubernetes?
- Create a Cluster
- Create a Deployment
- Viewing Pods and Nodes
- Using Services, Labels and Selectors
- Scaling your App
- Updating the App
- Try and play
- Summary

Cisco Webex Teams

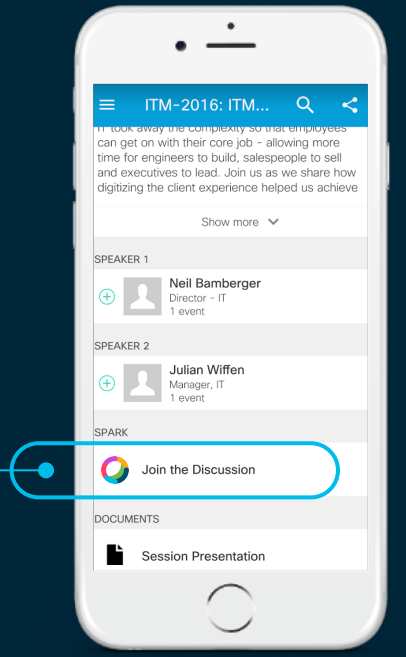
Questions?

Use Cisco Webex Teams to chat with the speaker after the session

How

1. Find this session in the Cisco Live Mobile App
2. Click “Join the Discussion”
3. Install Webex Teams or go directly to the team space
4. Enter messages/questions in the team space

Webex Teams will be moderated by the speaker until June 16, 2019.



cs.co/ciscolivebot#LTRPRG-1200

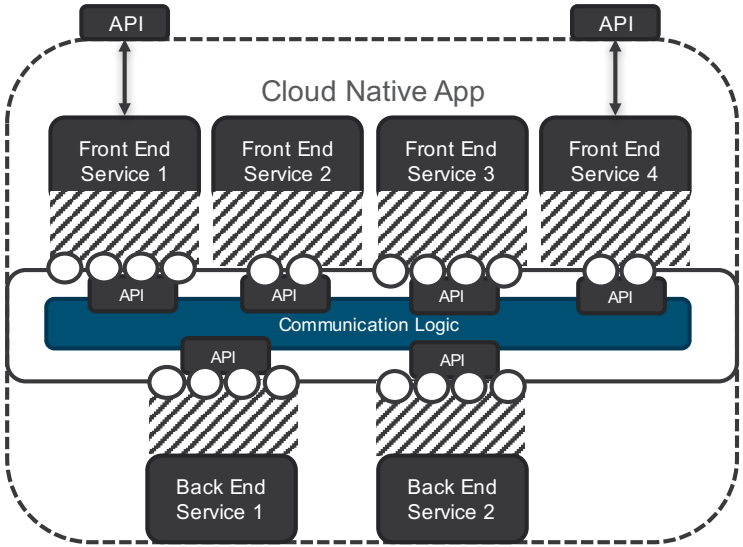
Expected schedule

	Topic	Lab	Duration
13:00 – 13:20	Introduction, What is Kubernetes?	No Lab	20 minutes
13:20 – 13:50	Installing Kubernetes and Create a Cluster	Lab 1	30 minutes
13:50 – 14:10	Create a Deployment	Lab 2	20 minutes
14:10 – 14:30	Viewing Pods and Nodes	Lab 3	20 minutes
14:30 - 15:00	Afternoon break		
15:00 – 15:25	Using Services, Labels and Selectors	Lab 4	25 minutes
15:25 – 15:45	Scaling your App	Lab 5	20 minutes
15:45 – 16:10	Updating the App	Lab 6	25 minutes
16:10 – 16:40	Try and play	Lab 7	30 minutes
16:40 – 17:00	Summary, Q&A, Lab 8	Lab 8 (if there is time)	20 minutes

What is Kubernetes?

You make networking **possible**

Distributed Applications and Services



- In a distributed application, different pieces of the app are called “services.”
- **Services** are **also called microservices** because they are small and dedicated to specific functions as we have shown in first section
- The services or microservices are really just “containers in production.”
- A service only runs one image, but it codifies the way that image runs—what ports it should use, how many replicas of the container should run so the service has the capacity it needs, and so on.
- Scaling a service changes the number of container instances running that piece of software, assigning more computing resources to the service in the process.
- **API’s** – is a software interface used to communicate microservices.

NOTE: For our discussion, a ***microservice*** is a cloud-hosted service that delivers a specific narrowly scope capability and responds to a business need or requirement

Container Technologies

An Architecture View

Container Platforms

RED HAT OPENSHIFT Container Platform

docker DOCKER DATACENTER

Orchestration

Kubernetes

docker SWARM

MESOS

Nomad

Runtime

rkt

OPEN CONTAINER INITIATIVE

docker

Infrastructure

cisco

APIC

HyperFlex All Flash

HyperFlex

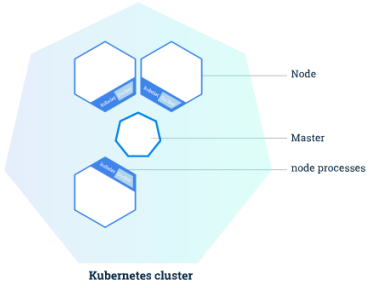
vmware

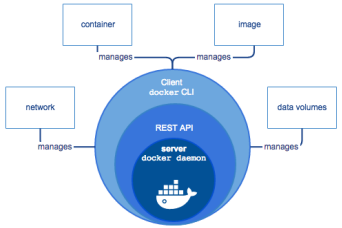
Azure Stack

openstack

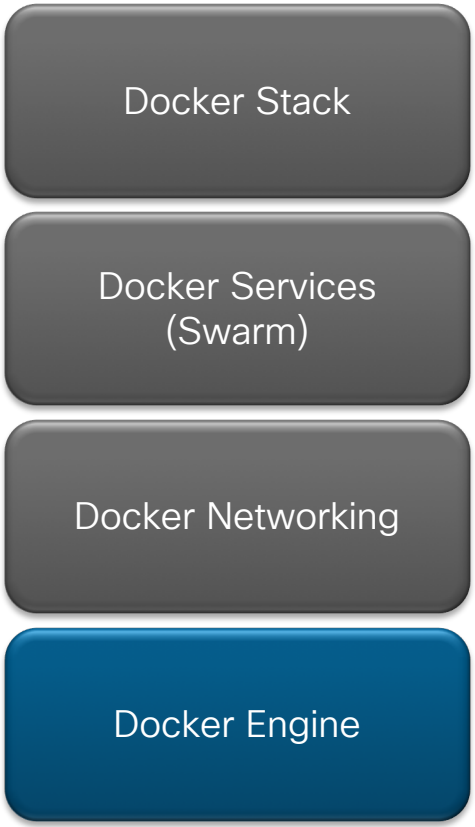
Google Cloud Platform

amazon web services





A quick Docker 101



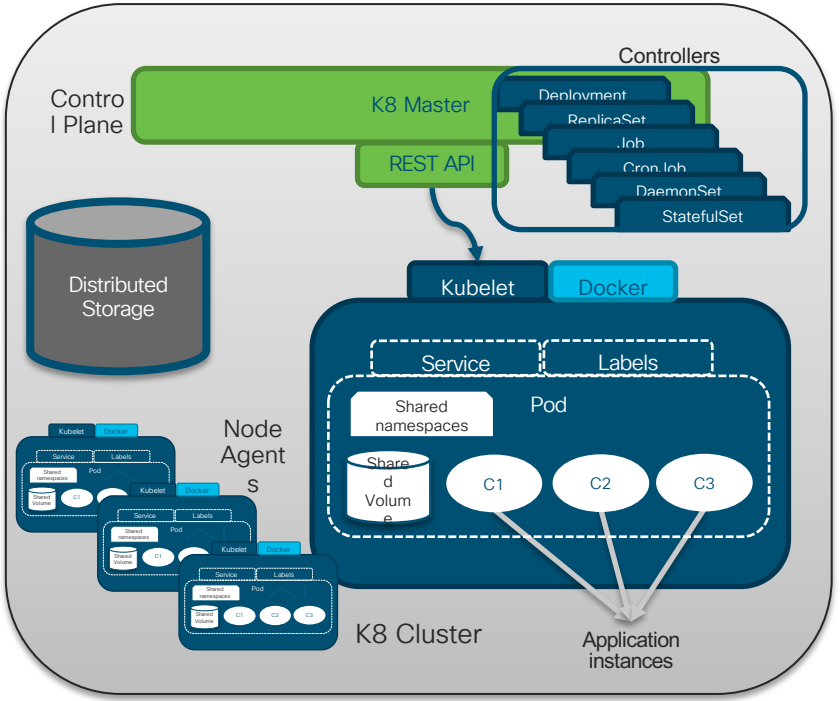
- Deploying a Docker Service Stack in a Cluster means, distributing Docker Swarms in several physical or virtual nodes
- The concept of Services and a Service Cluster managed by Docker Swarms is way Docker manages load-balancing in containers
- Docker container networking principles is based on IPTables Linux networking and supports certaing 3rd party plugins
- Docker engine is the daemon that makes possible to port your app stacked in layers using lightweight composites of a running instance

What is Kubernetes?

- Kubernetes is a portable, extensible open-source platform for managing containerized workloads and services
- You can think of Kubernetes as:
  - a container platform
  - a microservices platform
  - a portable cloud platform
- Provides a **container-centric** management environment
- Orchestrates computing, networking, and storage infrastructure on behalf of user workloads
- It is a platform for developers, who create components and tools to deploy, scale and manage applications
- So, in summary is THE PLATFORM for Cloud Native Applications

Kubernetes at a Glance

- A **Kubernetes (or K8)** is an open-source platform designed to automate deploying, scaling, and operating application containers.
- Kubernetes components:
  - Cluster (at least 3 nodes)
  - Container engine (Docker, Rkt or Open Container are supported engines)
- Kubernetes cluster:
  - Master – coordinates the cluster and also schedules the Pods
  - Nodes – workers that run applications
- Kubelet – agent that communicates with Kubernetes Master
- Nodes must have a container engine
- Pods are the atomic unit on the Kubernetes platform
- Deployment creates Pods with containers and other resources like volumes inside them
- Services is the way Pods communicates with external world



A quick Kubernetes 101

- A **Controller** is a they way to build a Cluster. An example of controller is a Deployment, which is a declarative Kubernetes controller, where you described a desired state of a specific Pod image with specific needs.
- A **Service** is the way to expose a Pod to the external world using the controller instructions and elements. It provides connectivity and ways to scale and update the application.
- A **Pod** is the smallest and simplest Kubernetes unit, which runs on the Kubernetes nodes and represents a running process. Inside the Pod there are one or many containers
- **Docker engine** is reused as the the daemon that makes possible to port your app stacked in layers using lightweight composites of a running instance. Kubernetes supports other container engines such as RKT and Open Containers

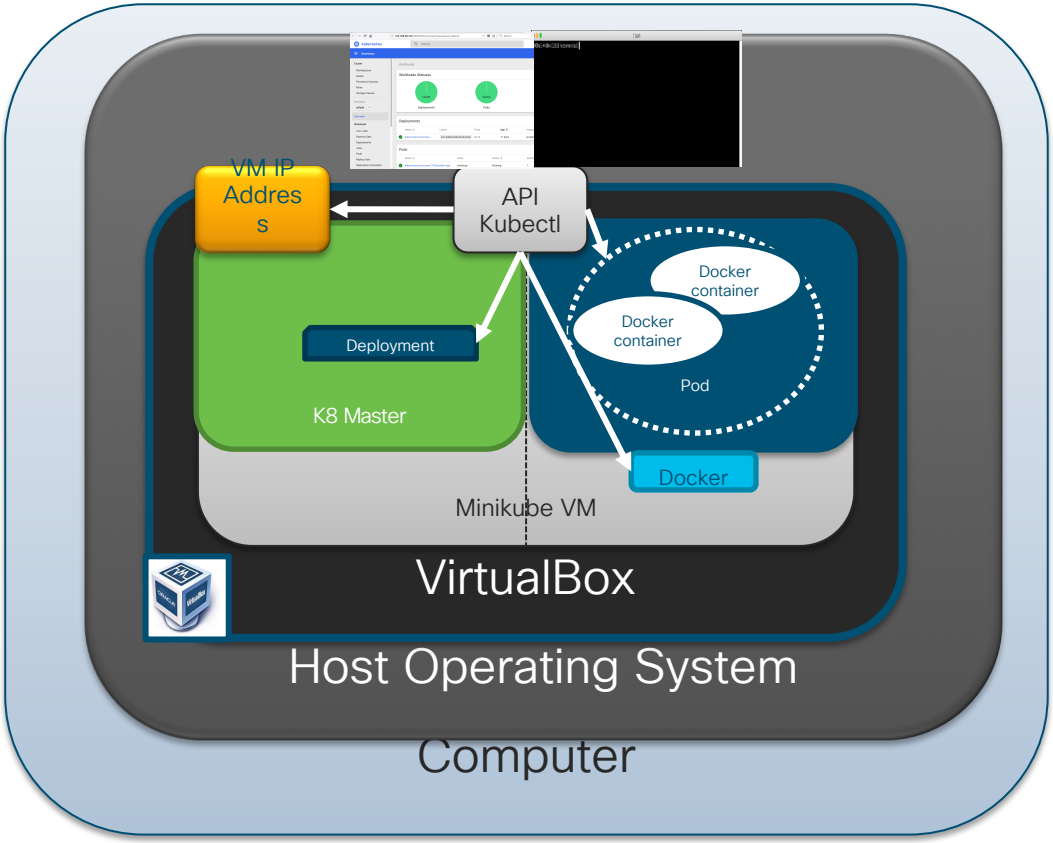
Installing Kubernetes Cluster

Minikube Single Node

You make networking **possible**

Cisco live!

Lab concept – Minikube Single-Node Kubernetes



- **Minikube** is a tool that makes it easy to run Kubernetes locally.
- Minikube runs a single-node Kubernetes cluster inside a VM on your laptop or computer for users looking to try out Kubernetes or develop with it day-to-day.
- Minikube creates a VM where a Node with docker is running
- What you do in this cluster, will work on a scale-out deployment
- Requirements:
  - Hypervisor (i.e. Virtual Box)
  - Kubectl (kubelet application)
  - Minikube (Kubernetes VM)
  - Optionally, docker installed locally (however, Minikube comes with its own docker engine so no need to install)

Updated lab guide and material

https://github.com/hemorale/CLUS19-LTRPRG-1200

Create a Minikube Cluster on Windows

Prerequisites

- Install hypervisor
  - Virtual Box - recommended
  - Hyper-V – only if you have Windows 10
- Install kubectl
  - Use the same version of your server. Using a newer version may cause errors
- Install minikube (Kubernetes single node installation)
- Optional - Install docker for Windows 10 ONLY
  - For more information and installation documentation https://store.docker.com/editions/community/docker-ce-desktop-windows

Create a Minikube Cluster on Windows

Hypervisor

- Install hypervisor
  - Virtual Box – we will use Virtual Box. Download from this URL:
  - https://download.virtualbox.org/virtualbox/5.2.30/VirtualBox-5.2.30-130521-Win.exe
  - Hyper-V – if you have windows 10, this is fine. If you install hyper-v, this will be the default hypervisor for Minikube

Create a Minikube Cluster on Windows

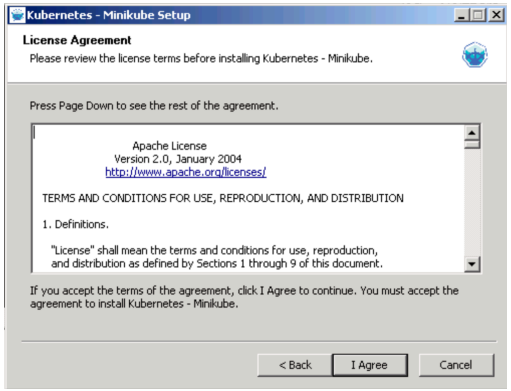
Kubectl

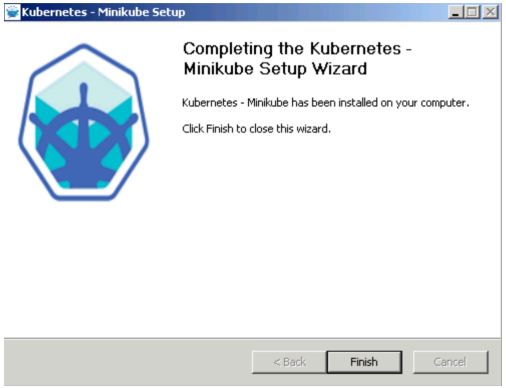
- Install kubectl
  - Download this release from:
  - https://storage.googleapis.com/kubernetes-release/release/v1.14.0/bin/windows/amd64/kubectl.exe
  - Use the same version of your server. Using a newer version may cause errors

Install Minikube on Windows

Minikube

- Download latest release (currently v1.1.1) and install under C:\Program Files (x86)\Kubernetes\Minikube\ (Preferred and quicker method)
  - https://github.com/kubernetes/minikube/releases/download/v1.1.1/minikube-windows-amd64.exe
- Or Run the installer





Create a Minikube Cluster on Windows

Summary

- Install hypervisor
  - Virtual Box – we will use Virtual Box. Download from this URL:
  - https://download.virtualbox.org/virtualbox/5.2.30/VirtualBox-5.2.30-130521-Win.exe
  - Hyper-V – if you have windows 10, this is fine. If you install hyper-v, this will be the default hypervisor for Minukube
- Install kubectl
  - Download this release from:
  - https://storage.googleapis.com/kubernetes-release/release/v1.14.0/bin/windows/amd64/kubectl.exe
  - Use the same version of your server. Using a newer version may cause errors
- Install Minikube
  - Download latest release (currently 1.1.1):
  - https://github.com/kubernetes/minikube/releases/download/v1.1.1/minikube-windows-amd64.exe
- Optional - Install Docker. Only if you have Windows 10
  - https://download.docker.com/win/edge/Docker%20for%20Windows%20Installer.exe
  - For more information and installation documentation https://store.docker.com/editions/community/docker-ce-desktop-windows

Starting Minikube

Windows

- Start Minikube. With no defined driver it will use Virtual Box.

```
$ minikube start
```

```
PS C:\Users\Lenovo> minikube version
minikube version: v1.1.0
PS C:\Users\Lenovo> minikube start
* minikube v1.1.0 on windows (amd64)
* minikube will upgrade the local cluster from Kubernetes 1.10.0 to 1.14.2
* Downloading Minikube ISO ...
 21.58 MB / 131.28 MB [======>------
```

```
PS C:\Users\Lenovo> minikube start
* minikube v1.1.0 on windows (amd64)
* minikube will upgrade the local cluster from Kubernetes 1.10.0 to 1.14.2
* Downloading Minikube ISO ...
 131.28 MB / 131.28 MB [============================================] 100.00% 0s
* Tip: Use 'minikube start -p <name>' to create a new cluster, or 'minikube delete' to delete this one.
* Re-using the currently running virtualbox VM for "minikube" ...
* Waiting for SSH access ...
* Configuring environment for Kubernetes v1.14.2 on Docker 17.12.1-ce
* Downloading kubeadm v1.14.2
* Downloading kubelet v1.14.2
* Pulling images ...
* Relaunching Kubernetes v1.14.2 using kubeadm ...
* Verifying: apiserver proxy etcd scheduler controller dns
* Done! kubectl is now configured to use "minikube"
PS C:\Users\Lenovo>
PS C:\Users\Lenovo>
PS C:\Users\Lenovo>
```

Starting Minikube

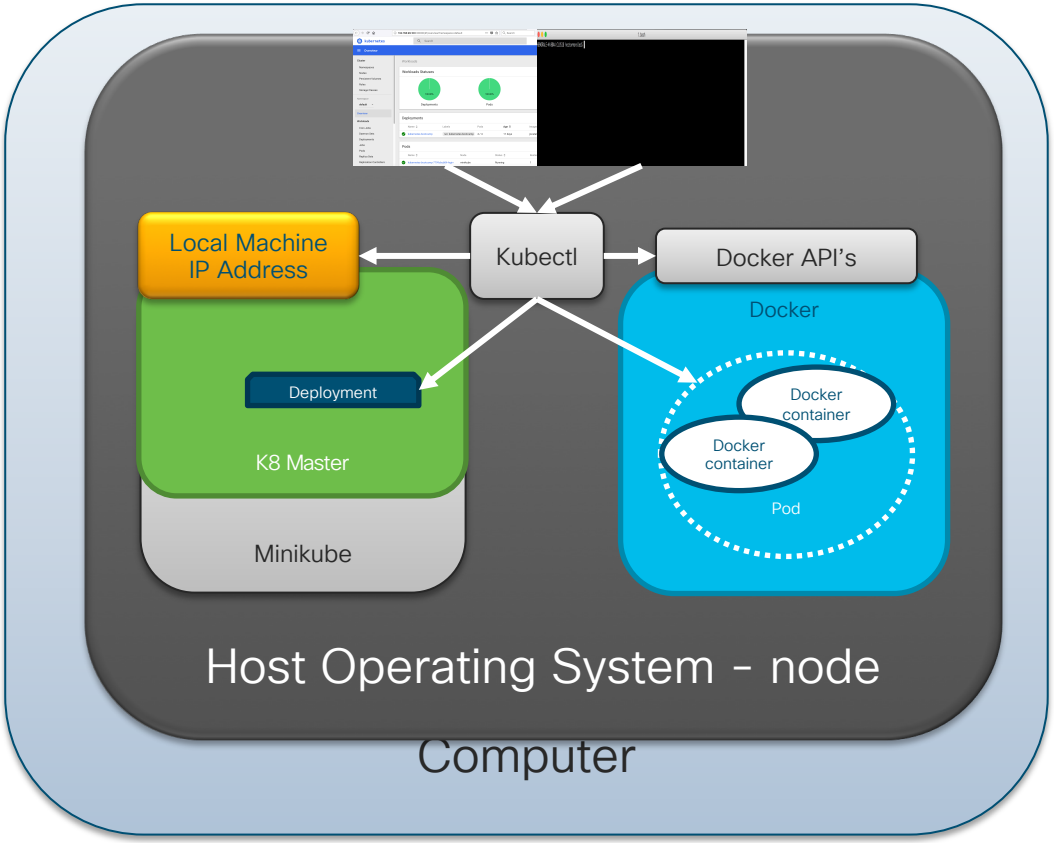
Windows

- Status of Minikube when it has been recently installed

```
$ minikube status
```

```
PS C:\Users\Lenovo> minikube status
There is a newer version of minikube available (v1.1.1).  Download it here:
https://github.com/kubernetes/minikube/releases/tag/v1.1.1

To disable this notification, run the following:
minikube config set WantUpdateNotification false
host: Running
kubelet: Running
apiserver: Running
kubectl: Correctly Configured: pointing to minikube-vm at 192.168.99.100
```

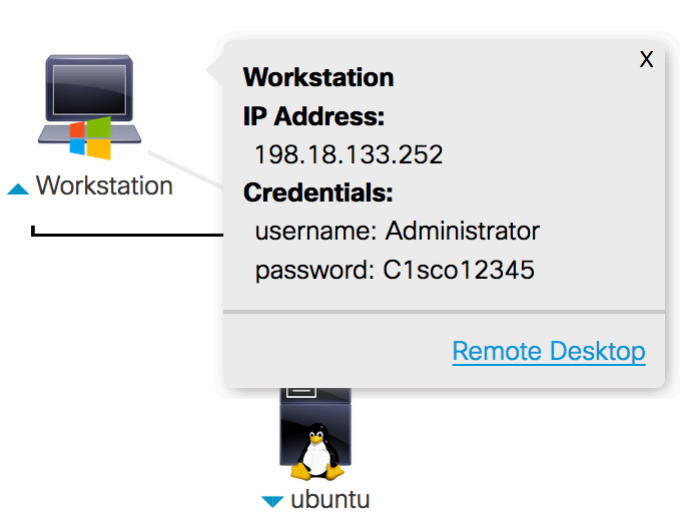
Minikube start --vm-driver=none

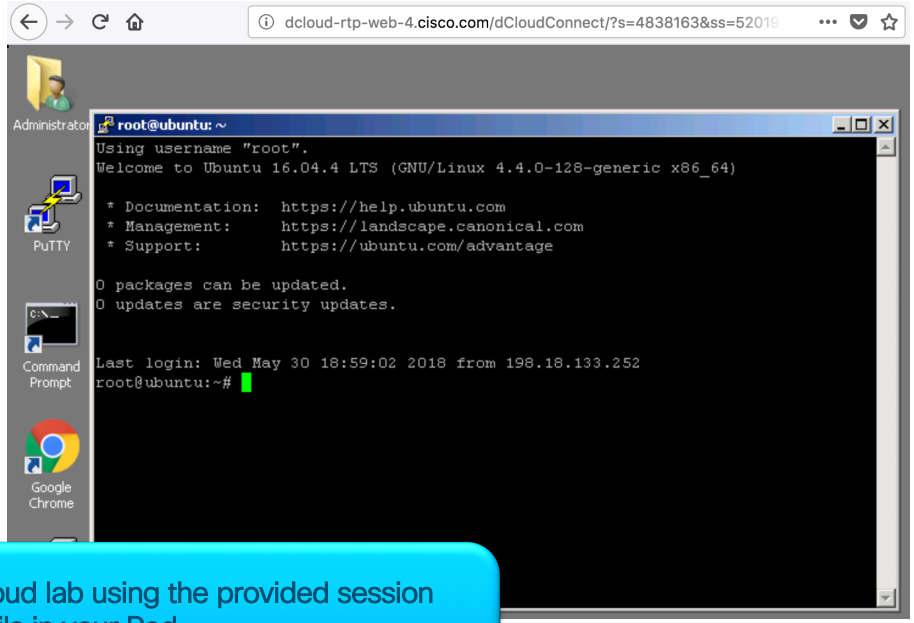


- **Minikube** also runs without any hypervisor.
- The VM has two main functions:
  - Create a Master and a Node locally
  - Provide a copy of Docker to run locally
- When Minikube starts with no VM driver, we need to have Docker installed locally
- Your machine becomes the node that runs Docker
- Kubectl uses the Docker API's to create and manage the containers and to talk to Master node, Minikube
- When you run Minikube in this mode, you need to be very careful with the certificates, which are installed at root level and may represent a security hole. **Don't use this mode in your personal machines**
- **This is just for lab purposes**

Running Minikube with no VM driver

dCloud Lab





You can access our dCloud lab using the provided session details in your Pod

- Anyconnect to dCloud Server
- SSH to Ubuntu VM where minikube is installed

Running Minikube with no VM driver

dCloud Lab

- Start Minikube. With no defined driver it will use Virtual Box.

  `$ minikube start [--vm-driver=none]`

```
HEMORALE-M-80A4:~ hectormorales$ minikube start --vm-driver=none
Starting local Kubernetes v1.10.0 cluster...
Starting VM...
Getting VM IP address...
Moving files into cluster...
Setting up certs...
Connecting to cluster...
Setting up kubeconfig...
Starting cluster components...
Kubectl is now configured to use the cluster.
```

NOTE: It is possible to use no VM but requires Docker installed locally. Please note there are some risks

```
===================
WARNING: IT IS RECOMMENDED NOT TO RUN THE NONE DRIVER ON PERSONAL WORKSTATIONS
        The 'none' driver will run an insecure kubernetes apiserver as root that may leave the host vulnerable to CSRF attacks

When using the none driver, the kubectl config and credentials generated will be root owned and will appear in the root home directory.
You will need to move the files to the appropriate location and then set the correct permissions.  An example of this is below:

        sudo mv /root/.kube $HOME/.kube # this will write over any previous configuration
        sudo chown -R $USER $HOME/.kube
        sudo chgrp -R $USER $HOME/.kube

        sudo mv /root/.minikube $HOME/.minikube # this will write over any previous configuration
        sudo chown -R $USER $HOME/.minikube
        sudo chgrp -R $USER $HOME/.minikube

This can also be done automatically by setting the env var CHANGE_MINIKUBE_NONE_USER=true
```

- Status of Minikube when it has been recently installed

  `$ minikube status`

```
HEMORALE-M-80A4:~ hectormorales$ minikube status
E0425 18:49:39.361798   44634 status.go:85] Error cluster status: Error: Unrecognized output from ClusterStatus:
E0425 18:49:39.362876   44634 util.go:151] Error formatting error message: Error msg with no stack trace cannot be reported
```

Using Minikube with Kubectl

- Now set the Minikube context. The context is what determines which cluster kubectl is interacting with:

  ```
  $ kubectl config use-context minikube
  ```

- Check again minikube status

  ```
  $ minikube status
  ```

```
PS C:\Users\Lenovo> minikube status
There is a newer version of minikube available (v1.1.1).  Download it here:
https://github.com/kubernetes/minikube/releases/tag/v1.1.1

To disable this notification, run the following:
minikube config set WantUpdateNotification false
host: Running
kubelet: Running
apiserver: Running
kubectl: Correctly Configured: pointing to minikube-vm at 192.168.99.100
```

- At this point, Minikube Server is up and running on a VM in Virtual Box with your local Docker Engine as your Kubernetes Worker. Please observe this is just for demo purposes. Don't use it in running environments

Kubernetes Cluster

- Now check the status of the Kubernetes cluster:

  `$ kubectl cluster-info`

```
PS C:\Users\Lenovo> kubectl cluster-info
Kubernetes master is running at https://192.168.99.100:8443
KubeDNS is running at https://192.168.99.100:8443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

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

Summary

- Install Kubernetes cluster on your machine using Single-Node Minikube
- Configure Minikube to use it on your Machine
- Test Minikube is correctly installed
- Learn more:
  - https://kubernetes.io/docs/tasks/tools/install-minikube/
  - https://kubernetes.io/docs/getting-started-guides/minikube/

Create a Kubernetes Cluster

You make networking **possible**

Create a Kubernetes Cluster with Minikube

- A **Kubernetes** cluster consists of two types of resources:
  - The Master coordinates the cluster
  - Nodes are the workers that run applications
- Cluster is managed from the master using the Kubernetes API. Nodes are never accessed directly, but they used an agent called Kubelet that allows Master to manage the node
- Nodes also need to run any Container Engine such as Docker, Rkt or CRI-O
- **Minikube** is a lightweight Kubernetes implementation that creates a VM on your local machine and deploys a simple cluster containing only one node
- We will use Minikube for the rest of the lab as our Kubernetes cluster

Lab 1 – Create a Cluster

Commands

- Get Minukube version

```
$ minikube version
```

- Kubectl version

```
$ kubectl version      #Client = kubectl version, Server = K8 version
```

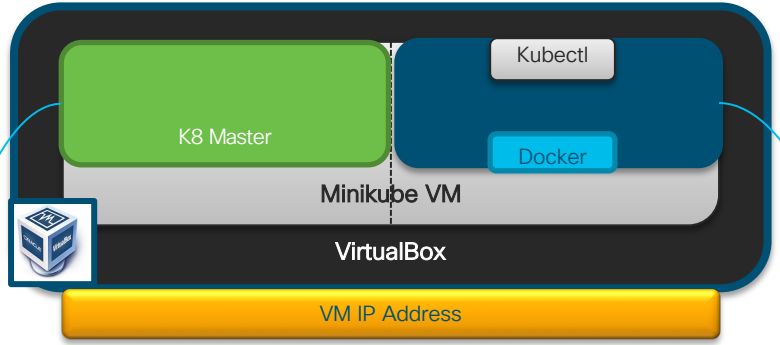
```
PS C:\Users\Lenovo> minikube version
minikube version: v1.1.0
PS C:\Users\Lenovo> kubectl version
Client Version: version.Info{Major:"1", Minor:"14", GitVersion:"v1.14.0", GitCommit:"641856db18352033a0d96dbc99153fa3b27298e5", GitTreeSta
te:"clean", BuildDate:"2019-03-25T15:53:57Z", GoVersion:"go1.12.1", Compiler:"gc", Platform:"windows/amd64"}
Server Version: version.Info{Major:"1", Minor:"14", GitVersion:"v1.14.2", GitCommit:"66049e3b21efe110454d67df4fa62b08ea79a19b", GitTreeSta
te:"clean", BuildDate:"2019-05-16T16:14:56Z", GoVersion:"go1.12.5", Compiler:"gc", Platform:"linux/amd64"}
```

- View nodes on the cluster

```
$ kubectl get nodes
```

```
PS C:\Users\Lenovo> kubectl get nodes
NAME       STATUS   ROLES    AGE     VERSION
minikube   Ready    <none>   4d16h   v1.14.2
```

Lab concept




```
HEMORALE-M-80A4:~ hectormorales$ minikube service list
|-------------|----------------------|-----------------------------|
|  NAMESPACE  |         NAME         |             URL             |
|-------------|----------------------|-----------------------------|
| default     | kubernetes           | No node port                |
| default     | kubernetes-bootcamp  | http://192.168.99.100:30927 |
| kube-system | heapster             | No node port                |
| kube-system | kube-dns             | No node port                |
| kube-system | kubernetes-dashboard | http://192.168.99.100:30000 |
| kube-system | monitoring-grafana   | http://192.168.99.100:30002 |
| kube-system | monitoring-influxdb  | No node port                |
|-------------|----------------------|-----------------------------|
```

```
HEMORALE-M-80A4:~ hectormorales$ kubectl get all
NAME                                        READY     STATUS    RESTARTS   AGE
pod/kubernetes-bootcamp-7799cbcb86-dg8nm    1/1       Running   2          11d
pod/kubernetes-bootcamp-7799cbcb86-hqjrv    1/1       Running   2          11d
pod/kubernetes-bootcamp-7799cbcb86-pm4hw    1/1       Running   2          11d
pod/kubernetes-bootcamp-7799cbcb86-tnwx6    1/1       Running   2          11d

NAME                          TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)          AGE
service/kubernetes            ClusterIP   10.96.0.1      <none>        443/TCP          19d
service/kubernetes-bootcamp   NodePort    10.106.65.63   <none>        8080:30927/TCP   12d

NAME                                   DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/kubernetes-bootcamp    4         4         4            4           16d

NAME                                              DESIRED   CURRENT   READY   AGE
replicaset.apps/kubernetes-bootcamp-5c69669756    0         0         0       16d
replicaset.apps/kubernetes-bootcamp-5f76cd7b94    0         0         0       11d
replicaset.apps/kubernetes-bootcamp-7799cbcb86    4         4         4       11d
```

Basics of Kubectl and Minikube

Commands

- Use of Minikube

```
$ minikube [command]
$ minikube –help
$ minikube start      #starts a local minikube cluster using default VirtualBox Driver
$ minikube status     #status of local minikube cluster
$ minikube service list        #minikube services running in the cluster
$ minikube dashboard   #opens local cluster Kubernetes dashboard
$ minikube stop        #stops a local minikube cluster
$ minikube delete      #delete a local minikube cluster, includes the VM and all files
```

Basics of Kubectl and Minikube

Commands

- Use of Kubectl

```
$ kubectl [command]    #controls Kubernetes cluster manager
```

- Format of Kubectl commands

```
$ kubectl [command] [options]
```

- Useful commands

```
$ kubectl --help       #kubectl help
$ kubectl config use-context minikube   #configures kubectl to use minikube cluster
$ kubectl config view #display current kubeconfig file settings
$ kubectl config current-context       #display current context
$ kubectl config get-contexts  #display all existing contexts
$ kubectl cluster-info         #display cluster information
$ kubectl top [node | pod]     #display resource usage
```

More information on:
https://kubernetes.io/docs/reference/kubectl/cheatsheet/

Summary

- Learned how to create a Kubernetes Cluster with Minikube
- Understood Minikube and Kubectl functionality
- Learned basic Minikube and Kubectl commands
- Learn more of Minikube https://kubernetes.io/docs/getting-started-guides/minikube/

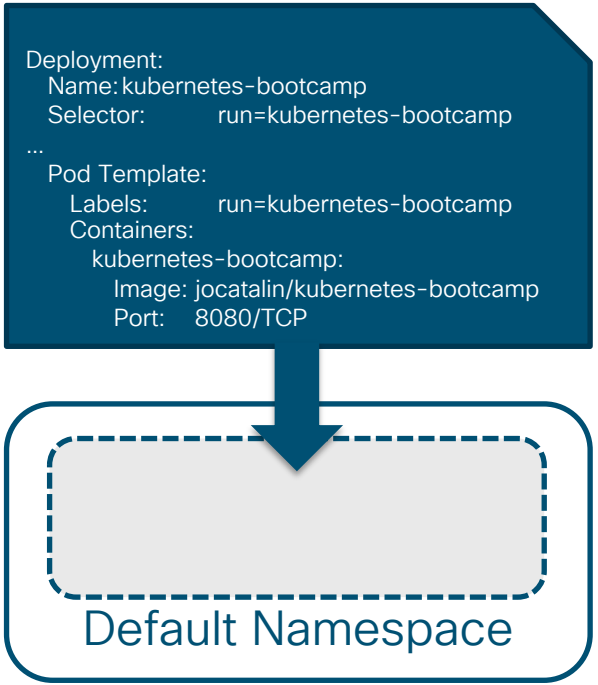
Create a Deployment

You make networking **possible**

Cisco live!

What is a Deployment?

- Kubernetes uses two types of workloads to control your apps. These are **Pods** and **Controllers**
- A Controller is a logical function which drives your app on a specific way using a specific set of instructions
- A **Deployment** is a declarative Kubernetes controller, where you described a desired state of a specific Pod image with specific needs
- A deployment controller is the recommended form to run a cloud-native stateless application as it has the following features:
  - Declare a desired state of your Pods, including the number of replicas of your App
  - Rollback to an earlier version of the Pods
  - Scale up the deployment
  - Pause the deployment in case your App needs a fix
  - Know the status of your deployment and make proper actions/fixes
  - Clean up older replicas



REFERENCE: https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

Example of Deployment File

nginx-deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
   selector:
      matchLabels:
         app: nginx
   replicas: 3
   template:
      metadata:
         labels:
           app: nginx
      spec:
         containers:
          - name: nginx
            image: nginx:1.7.9
            ports:
            - containerPort: 80
```

- This is the standard Kubernetes file
- In this case, it is a Deployment file called nginx-deployment
- It will have 3 replicas
- It will use a container named nginx and will used a specific version
- It will use container port 80

5 minutes

Creating a Deployment

Your first deployment

- Working directory c:\Kubernetes\LTRDEV-1200
- Create your deployment file:

```
$ touch nginx-deployment.yaml
```

- Create the deployment file as shown and save it
- Create the deployment using the following command:

```
$ kubectl create -f nginx-deployment.yaml
```

- Verify your deployment status

```
$ kubectl get deploy
$ kubectl get rs                    #replicasets
$ kubectl rollout status deployment/nginx-
  deployment
$ kubectl get pods
```

- Finally, delete your deployment

```
$ kubectl delete deployment/nginx-deployment
```

nginx-deployment.yaml

nginx-deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
   selector:
      matchLabels:
         app: nginx
   replicas: 3
   template:
      metadata:
         labels:
           app: nginx
      spec:
         containers:
          - name: nginx
            image: nginx:1.7.9
            ports:
            - containerPort: 80
```

A copy of this file is git cloned in your local directory

10 minutes

Lab 2 – Create a Deployment

Commands

- Alternatively, Kubernetes can create a Deployment using the run command (similar to Docker run) from an existing image

```
$ #Use Kubectl run <image_name> --image=<image_location> [parameters]
$ kubectl run kubernetes-bootcamp --image=gcr.io/google-samples/kubernetes-bootcamp:v1 --port=8080
```

```
PS C:\CLUS19\LTRPRG-1200> kubectl run kubernetes-bootcamp --image=gcr.io/google-samples/kubernetes-bootcamp:v1 --port=8080
kubectl run --generator=deployment/apps.v1 is DEPRECATED and will be removed in a future version. Use kubectl run --generator=run-pod/v1 or kubectl create instead.
deployment.apps/kubernetes-bootcamp created
```

- List your deployments

```
$ kubectl get deployments
```

```
PS C:\CLUS19\LTRPRG-1200> kubectl get deployments
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp   1/1     1            1           66m
```

```
$ kubectl get pod
```

```
PS C:\CLUS19\LTRPRG-1200> kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-b94cb9bff-cjwkh   1/1     Running   0          11h
PS C:\CLUS19\LTRPRG-1200> kubectl get pods -o wide
NAME                                  READY   STATUS    RESTARTS   AGE   IP           NODE       NOMINATED NODE   READINESS GATES
kubernetes-bootcamp-b94cb9bff-cjwkh   1/1     Running   0          11h   172.17.0.5   minikube   <none>           <none>
```

Lab 2 – Create a Deployment

Kubectl Proxy

- This command starts a Proxy to the Kubernetes API Server. Open another terminal window and run:

```
$ kubectl proxy --port 8002          #connection between the "terminal" and the Kubernetnetes API Server. No -port option, default is port=8001
```

```
PS C:\CLUS19\LTRPRG-1200> kubectl proxy --port 8002
Starting to serve on 127.0.0.1:8002
```

- Use of direct API endpoints via the Proxy to query Pods. Type in your browser: http://localhost:8002/version

```
$ curl http://localhost:8002/version          #query kubectl version via API, only on Linux hosts
```

```
PS C:\CLUS19\LTRPRG-1200> curl http://localhost:8002/version


StatusCode        : 200
StatusDescription : OK
Content           : {
                      "major": "1",
                      "minor": "14",
                      "gitVersion": "v1.14.3",
                      "gitCommit": "5e53fd6bc17c0dec8434817e69b04a25d8ae0ff0",
                      "gitTreeState": "clean",
                      "buildDate": "2019-06-06T01:36:19Z",
                      "goVersion"...
```

Lab 2 – Create a Deployment

Commands

- Now we can make an HTTP request to the application running in that pod

```
$ # Powershell Commands
$ $POD_NAME = (kubectl get pod -o jsonpath="{.items[0].metadata.name}”)
```

```
PS C:\CLUS19\LTRPRG-1200> $POD_NAME = (kubectl get pod -o jsonpath="{.items[0].metadata.name}")
PS C:\CLUS19\LTRPRG-1200> curl http://localhost:8002/api/v1/namespaces/default/pods/$POD_NAME/


StatusCode        : 200
StatusDescription : OK
Content           : {
                        "kind": "Pod",
                        "apiVersion": "v1",
                        "metadata": {
                          "name": "kubernetes-bootcamp-b94cb9bff-cjwkh",
                          "generateName": "kubernetes-bootcamp-b94cb9bff-",
                          "namespace": "default",
                          "selfL...
```

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

Lab 2 – Create a Deployment

Commands

- Now we can make an HTTP request to the application running in that pod. It can be done either using CURL on the terminal window or typing the complete route in your browser (HINT, copy the $POD_NAME variable content and paste at the end in your browser)

```
$ curl http://localhost:8002/api/v1/namespaces/default/pods/$POD_NAME/
```

```
PS C:\CLUS19\LTRPRG-1200> $POD_NAME = (kubectl get pod -o jsonpath="{.items[0].metadata.name}")
PS C:\CLUS19\LTRPRG-1200> curl http://localhost:8002/api/v1/namespaces/default/pods/$POD_NAME/


StatusCode        : 200
StatusDescription : OK
Content           : {
                      "kind": "Pod",
                      "apiVersion": "v1",
                      "metadata": {
                        "name": "kubernetes-bootcamp-b94cb9bff-cjwkh",
                        "generateName": "kubernetes-bootcamp-b94cb9bff-",
                        "namespace": "default",
                        "selfL...
```

```
PS C:\CLUS19\LTRPRG-1200> kubectl get pod -o jsonpath="{.items[0].metadata.name}"
kubernetes-bootcamp-b94cb9bff-cjwkh
```

localhost:8002/api/v1/namespaces/default/pods/kubernetes-bootcamp-b94cb9bff-cjwkh


```
{
  "kind": "Pod",
  "apiVersion": "v1",
  "metadata": {
    "name": "kubernetes-bootcamp-b94cb9bff-cjwkh",
    "generateName": "kubernetes-bootcamp-b94cb9bff-",
    "namespace": "default",
    "selfLink": "/api/v1/namespaces/default/pods/kubernetes-bootcamp-b94cb9bff-cjwkh",
    "uid": "7449453d-8b4b-11e9-a8dd-0800272f8b28",
    "resourceVersion": "27436",
    "creationTimestamp": "2019-06-10T06:46:20Z",
    "labels": {
      "pod-template-hash": "b94cb9bff",
      "run": "kubernetes-bootcamp"
    },
    "ownerReferences": [
```

Summary

- Learned what a deployment is and how it helps to control workloads
- Learned the basics of Kubernetes config files
- Understood Kubectl Proxy and API Server

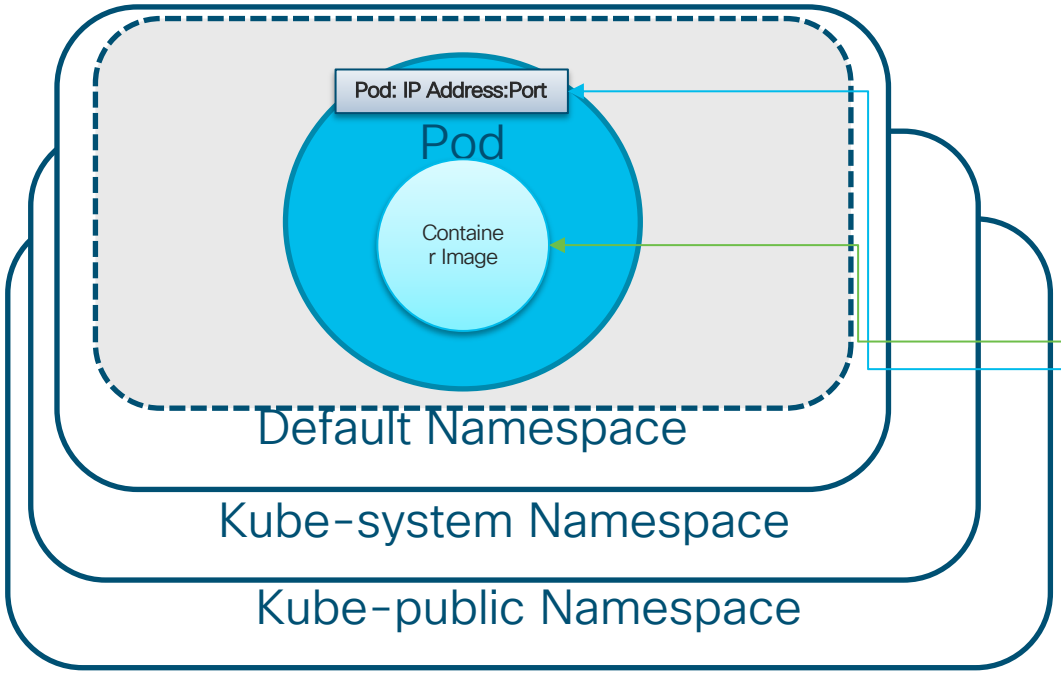
Viewing Pods and Nodes

You make networking **possible**

What is a Pod?

- Pod is the smallest and simplest Kubernetes unit, which runs on the Kubernetes nodes
- A Pod represents a running process:
  - Container or containers
  - Storage resources
  - An IP Address to access these resources
  - A mechanism that indicates how your container should run
- In a few words, the Pod is where you place your app and resources and the unit you manage
- The Pod in Kubernetes is the equivalent to a Container in Docker
- Nodes are the hosts where Namespaces allocates Pods
- Namespaces are Virtual Hosts that can be used for multiple Pod operations (we won't discuss Namespaces in this lab)

How a Pod works




```
Deployment:
  Name: kubernetes-bootcamp
  Selector:        run=kubernetes-bootcamp
...
  Pod Template:
    Labels:        run=kubernetes-bootcamp
    Containers:
      kubernetes-bootcamp:
        Image: jocatalin/kubernetes-bootcamp
        Port:  8080/TCP
```

REFERENCE: https://kubernetes.io/docs/concepts/workloads/pods/pod-overview/

Namespaces and Pods examples

```
HEMORALE-M-80A4:nginx-deploy hectormorales$ kubectl get namespaces
NAME          STATUS    AGE
default       Active    27d
kube-public   Active    27d
kube-system   Active    27d
HEMORALE-M-80A4:nginx-deploy hectormorales$ kubectl --namespace=default get pods
NAME                                   READY     STATUS    RESTARTS   AGE
kubernetes-bootcamp-7799cbcb86-dg8nm   1/1       Running   6          19d
kubernetes-bootcamp-7799cbcb86-hqjrv   1/1       Running   6          19d
kubernetes-bootcamp-7799cbcb86-pm4hw   1/1       Running   6          19d
kubernetes-bootcamp-7799cbcb86-tnwx6   1/1       Running   6          19d
HEMORALE-M-80A4:nginx-deploy hectormorales$ kubectl --namespace=kube-system get pods
NAME                                    READY     STATUS    RESTARTS   AGE
etcd-minikube                           1/1       Running   0          5m
heapster-hwkgg                          1/1       Running   6          19d
influxdb-grafana-chws6                  2/2       Running   14         19d
kube-addon-manager-minikube             1/1       Running   15         27d
kube-apiserver-minikube                 1/1       Running   0          5m
kube-controller-manager-minikube        1/1       Running   0          5m
kube-dns-86f4d74b45-s5mbs               3/3       Running   56         27d
kube-proxy-jj48z                        1/1       Running   0          4m
kube-scheduler-minikube                 1/1       Running   4          8d
kubernetes-dashboard-5498ccf677-2rpmh   1/1       Running   31         27d
storage-provisioner                     1/1       Running   29         27d
```

Default namespace

Pods

kube-system namespace

Pods

Lab 3 – Viewing Pods and Nodes

Commands

- Look for existing Pods inside the

```
$  kubectl get pods
```

```
PS C:\CLUS19\LTRPRG-1200> kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-b94cb9bff-cjwkh   1/1     Running   0          11h
PS C:\CLUS19\LTRPRG-1200> kubectl get pods -o wide
NAME                                READY   STATUS    RESTARTS   AGE   IP           NODE       NOMINATED NODE   READINESS GATES
kubernetes-bootcamp-b94cb9bff-cjwkh   1/1     Running   0          11h   172.17.0.5   minikube   <none>           <none>
```

- Describe the containers and resources inside the Pods. The output is extensive and provides a full view of what has happened within inside the pod and its containers and resources

```
$  kubectl describe pods #just showing a brief piece
```

```
PS C:\Users\Lenovo> kubectl describe pods
Name:               kubernetes-bootcamp-b94cb9bff-2lm54
Namespace:          default
Priority:           0
PriorityClassName:  <none>
Node:               minikube/10.0.2.15
Start Time:         Mon, 10 Jun 2019 19:23:53 -0500
Labels:             pod-template-hash=b94cb9bff
                    run=kubernetes-bootcamp
Annotations:        <none>
Status:             Running
IP:                 172.17.0.5
Controlled By:      ReplicaSet/kubernetes-bootcamp-b94cb9bff
```

Step 3 – Viewing Pods and Nodes

Commands

- Run this command or type in your browser (don't forget to take out the "curl" command and type in the full pod name)

```
$ curl
  http://localhost:8002/api/v1/namespaces/default/pods/$POD_NAME/proxy/
```

```
PS C:\CLUS19\LTRPRG-1200> curl http://localhost:8002/api/v1/namespaces/default/pods/$POD_NAME/proxy/


StatusCode        : 200
StatusDescription : OK
Content           : Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-b94cb9bff-cjwkh | v=1
```

Step 3 – Viewing Pods and Nodes

Commands

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

```
$ kubectl logs $POD_NAME
```

```
PS C:\CLUS19\LTRPRG-1200> kubectl logs $POD_NAME
Kubernetes Bootcamp App Started At: 2019-06-10T06:46:22.067Z | Running On:  kubernetes-bootcamp-b94cb9bff-cjwkh

Running On: kubernetes-bootcamp-b94cb9bff-cjwkh | Total Requests: 1 | App Uptime: 42061.463 seconds | Log Time: 2019-06-10T18:27:23.531Z
Running On: kubernetes-bootcamp-b94cb9bff-cjwkh | Total Requests: 2 | App Uptime: 42230.029 seconds | Log Time: 2019-06-10T18:30:12.096Z
```

Step 3 – Viewing Pods and Nodes

Commands

- We can execute a command inside the container using kubectl exec

```
$ kubectl exec $POD_NAME env          #execute the command inside the container to get environment variables
$ kubectl exec -ti $POD_NAME bash     #start a bash session into the Pod's container. We can run any command inside the
  Pod's container
```

```
PS C:\CLUS19\LTRPRG-1200> kubectl exec $POD_NAME env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=kubernetes-bootcamp-b94cb9bff-cjwkh
KUBERNETES_SERVICE_HOST=10.96.0.1
KUBERNETES_SERVICE_PORT=443
KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_PORT=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP_PROTO=tcp
KUBERNETES_PORT_443_TCP_PORT=443
KUBERNETES_PORT_443_TCP_ADDR=10.96.0.1
NPM_CONFIG_LOGLEVEL=info
NODE_VERSION=6.3.1
HOME=/root
```

```
PS C:\CLUS19\LTRPRG-1200> kubectl exec -ti $POD_NAME bash
root@kubernetes-bootcamp-b94cb9bff-cjwkh:/# 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(
:",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-b94cb9bff-cjwkh:/# curl localhost:8080
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-b94cb9bff-cjwkh | v=1
root@kubernetes-bootcamp-b94cb9bff-cjwkh:/# exit
exit
```

We are inside the container!!

Step 3 – Viewing Pods and Nodes

Commands

- Let's jump inside our container within the Pod and run a few commands

```
$ cat server.js                    #check our Web Server app inside the containers
$ curl localhost:8080              #check Web Server is up and running inside the container
$ exit                             #exit container connection
```

```
PS C:\CLUS19\LTRPRG-1200> kubectl exec -ti $POD_NAME bash
root@kubernetes-bootcamp-b94cb9bff-cjwkh:/# 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(
:",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-b94cb9bff-cjwkh:/# curl localhost:8080
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-b94cb9bff-cjwkh | v=1
root@kubernetes-bootcamp-b94cb9bff-cjwkh:/# exit
exit
```

Understanding Minikube Cluster Services

Kubernetes Cluster basics

- As we could see in the previous step, we were able to access the Pod's container (our app) either using the kubectl API or the minikube cluster. So let's understand a bit more:

```
$ minikube service list
```

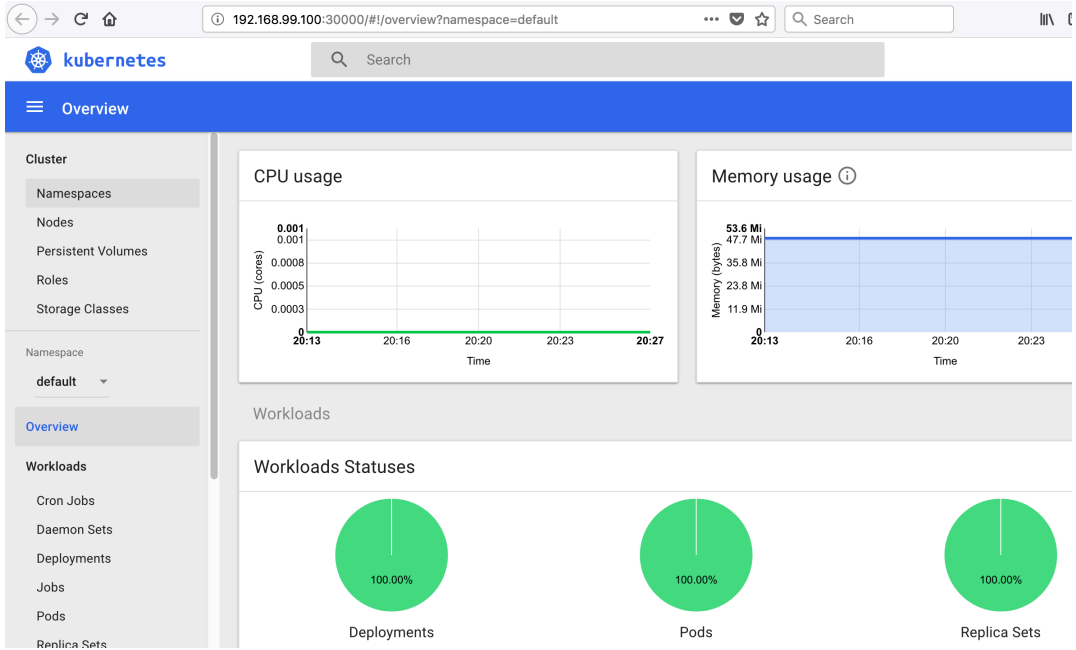
```
PS C:\CLUS19\LTRPRG-1200> minikube service list
|-------------|----------------------|--------------|
|  NAMESPACE  |         NAME         |     URL      |
|-------------|----------------------|--------------|
| default     | kubernetes           | No node port |
| kube-system | kube-dns             | No node port |
| kube-system | kubernetes-dashboard | No node port |
|-------------|----------------------|--------------|
```

```
$ minikube dashboard
```

Understanding Minikube Cluster Services

Kubernetes Cluster basics

```
$ minikube dashboard
```



Summary

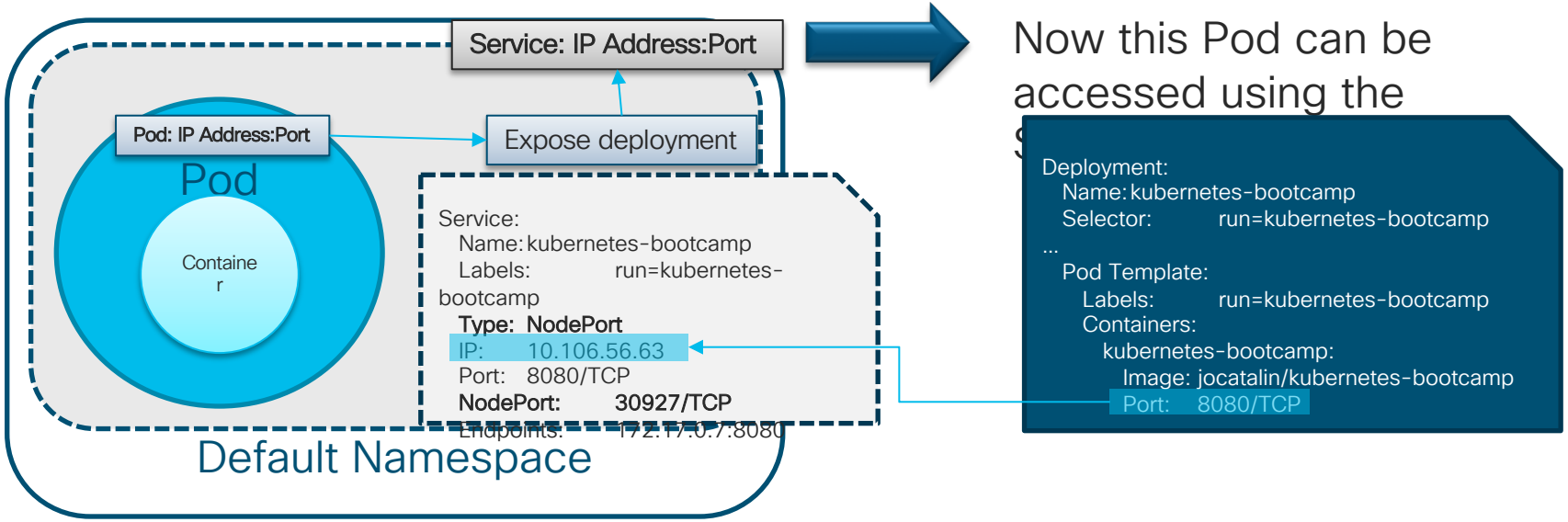
- Learned what a Pod is and how it stores our app
- Learned how the Deployment controller works to control Pods
- Learned how to access a container inside a Pod
- Learned how Minikube implements Pods and Namespaces

Using Services, Labels and Selectors

You make networking **possible**

Cisco live!

Services




```
HEMORALE-M-80A4:nginx-deploy hectormorales$ kubectl get pods -o wide
NAME                                   READY     STATUS    RESTARTS   AGE       IP           NODE
kubernetes-bootcamp-7799cbcb86-dg8nm   1/1       Running   8          20d       172.17.0.7   minikube
```

REFERENCE: https://kubernetes.io/docs/concepts/services-networking/service/

Labels, Services and Selectors

```
HEMORALE-M-80A4:nginx-deploy hectormorales$ kubectl get services -o wide
NAME                  TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE       SELECTOR
kubernetes            ClusterIP   10.96.0.1       <none>        443/TCP          28d       <none>
kubernetes-bootcamp   NodePort    10.106.65.63    <none>        8080:30927/TCP   21d       run=kubernetes-bootcamp
```

```
HEMORALE-M-80A4:nginx-deploy hectormorales$ kubectl describe pods
Name:           kubernetes-bootcamp-7799cbcb86-dg8nm
Namespace:      default
Node:           minikube/10.0.2.15
Start Time:     Wed, 02 May 2018 18:15:48 -0500
Labels:         pod-template-hash=3355767642
                run=kubernetes-bootcamp
Annotations:    <none>
Status:         Running
IP:             172.17.0.7
Controlled By:  ReplicaSet/kubernetes-bootcamp-7799cbcb86
```

Label used to map a **Service** with a **Pod**
Label is used to create the Pod Template Hash as well (don't modified this)

```
Volumes:
  default-token-42x57:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-42x57
    Optional:    false
QoS Class:       BestEffort
Node-Selectors:  <none>
```

Selectors is a special form of Label where we can create a key/pair for a specific requirement. Node-selectors is a good example, where we define a Pod to be placed in a specific hardware node, for example, with a GPU

See more on https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/

Pods and interactions

- Pods that are running inside Kubernetes are running on a private, isolated network
- When we use kubectl, we're interacting through an API endpoint to communicate with our application.
- The API server will automatically create an endpoint for each pod, based on the pod name, that is also accessible through the Kubectl proxy.

Step 4 – Using Services, Labels and Selectors

Commands

- Look for existing Pods

  ```
  $ kubectl get pods
  ```

- List the current services from within our cluster. By default, Kubernetes creates a Service type ClusterIP

  ```
  $ kubectl get services
  ```

```
PS C:\CLUS19\LTRPRG-1200> kubectl get pods -o wide
NAME                                  READY   STATUS    RESTARTS   AGE   IP           NODE       NOMINATED NODE   READINESS GATES
kubernetes-bootcamp-b94cb9bff-cjwkh   1/1     Running   0          12h   172.17.0.5   minikube   <none>           <none>
PS C:\CLUS19\LTRPRG-1200> kubectl get services
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   6d17h
```

Step 4 – Using Services, Labels and Selectors

Commands

- Let's create a new Service using our current Deployment. This service will expose the Pod to the external world using port 8080

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

$ kubectl get services
```

```
PS C:\CLUS19\LTRPRG-1200> kubectl expose deployment/kubernetes-bootcamp --type="NodePort" --port 8080
service/kubernetes-bootcamp exposed
PS C:\CLUS19\LTRPRG-1200> kubectl get services
NAME                  TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
kubernetes            ClusterIP   10.96.0.1       <none>        443/TCP          6d17h
kubernetes-bootcamp   NodePort    10.103.200.60   <none>        8080:32499/TCP   10s
```

Step 4 – Using Services, Labels and Selectors

Commands

- Let's take a look to this created Service:

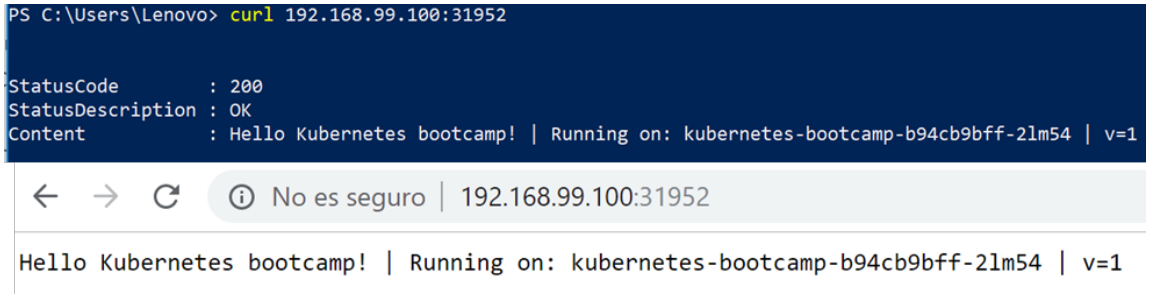
```
$ kubectl describe services/kubernetes-bootcamp

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

$ echo NODE_PORT=$NODE_PORT
```

Step 4 – Using Services, Labels and Selectors

Commands

- In the previous step, we learned how to get the exposed port by the cluster (the VM) so it is accessible from the external world. Let's do a couple of experiments. From within the console and from your browser. Which other command produces the SAME output? (HINT: Step 3 – Pod API's)

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



Step 4 – Using Services, Labels and Selectors

Commands

- Let's understand what a Deployment means. If we type

  ```
  $ kubectl describe deployments
  ```

- We see a bunch of details regarding our Pod. So, a deployment is the actual Pod description and the way it has to work within our cluster. See the label in the Pod
  - `Labels:  run=kubernetes-bootcamp`

```
PS C:\CLUS19\LTRPRG-1200> kubectl describe deployments/kubernetes-bootcamp
Name:                   kubernetes-bootcamp
Namespace:              default
CreationTimestamp:      Mon, 10 Jun 2019 19:23:53 -0500
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-b94cb9bff (4/4 replicas created)
Events:
  Type    Reason             Age   From                   Message
  ----    ------             ----  ----                   -------
  Normal  ScalingReplicaSet  2m9s  deployment-controller  Scaled up replica set kubernetes-bootcamp-b94cb9bff to 4
```

Step 4 – Using Services, Labels and Selectors

Commands

- Run the following commands:

```
$ kubectl get pods -l run=kubernetes-bootcamp
$ kubectl get services -l run=kubernetes-bootcamp
```

- What is the output?

```
PS C:\CLUS19\LTRPRG-1200> kubectl get pods -l run=kubernetes-bootcamp
NAME                                  READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-b94cb9bff-2lm54   1/1     Running   0          44m
PS C:\CLUS19\LTRPRG-1200> kubectl get services -l run=kubernetes-bootcamp
NAME                  TYPE       CLUSTER-IP     EXTERNAL-IP   PORT(S)          AGE
kubernetes-bootcamp   NodePort   10.98.92.163   <none>        8080:31952/TCP   14m
```

Step 4 – Using Services, Labels and Selectors

Commands

- Let's apply a new label to our Pod
- Run the following commands:

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

$ echo Name of the Pod: $POD_NAME

$ kubectl label pod $POD_NAME app=v1
```

```
PS C:\CLUS19\LTRPRG-1200> kubectl label pod $POD_NAME app=v1
pod/kubernetes-bootcamp-b94cb9bff-2lm54 labeled
```

Step 4 – Using Services, Labels and Selectors

Commands

- Let's explore again our Pod:

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

- We can now list our Pod using the label we added:

  ```
  $   kubectl get pods –l app=v1
  ```

```
PS C:\CLUS19\LTRPRG-1200> kubectl describe pods $POD_NAME
Name:               kubernetes-bootcamp-b94cb9bff-2lm54
Namespace:          default
Priority:           0
PriorityClassName:  <none>
Node:               minikube/10.0.2.15
Start Time:         Mon, 10 Jun 2019 19:23:53 -0500
Labels:             app=v1
                    pod-template-hash=b94cb9bff
                    run=kubernetes-bootcamp
Annotations:        <none>
Status:             Running
IP:                 172.17.0.5
Controlled By:      ReplicaSet/kubernetes-bootcamp-b94cb9bff
Containers:
  kubernetes-bootcamp:
    Container ID:   docker://ddb7eebdf694c1ee3ff8042ed40ec37f
```

```
PS C:\CLUS19\LTRPRG-1200> kubectl get pods -l app=v1
NAME                                   READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-b94cb9bff-2lm54   1/1     Running   0          49m
```

Step 4 – Using Services, Labels and Selectors

Commands

- Let's delete the service we created and run a few commands:

```
$ kubectl delete service -l run=kubernetes-bootcamp
$ kubectl get services
$ curl $(minikube ip):$NODE_PORT
```

- Let's run this one, what happens?:

```
$ kubectl exec -ti $POD_NAME curl localhost:8080
```

```
PS C:\CLUS19\LTRPRG-1200> curl 192.168.99.100:31952
curl :  Can't connect to remote server

+ curl 192.168.99.100:31952
+ ~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : InvalidOperation: (System.Net.HttpWebRequest:HttpWebRequest) [Invoke-WebRequest], WebException
    + FullyQualifiedErrorId : WebCmdletWebResponseException,Microsoft.PowerShell.Commands.InvokeWebRequestCommand
```

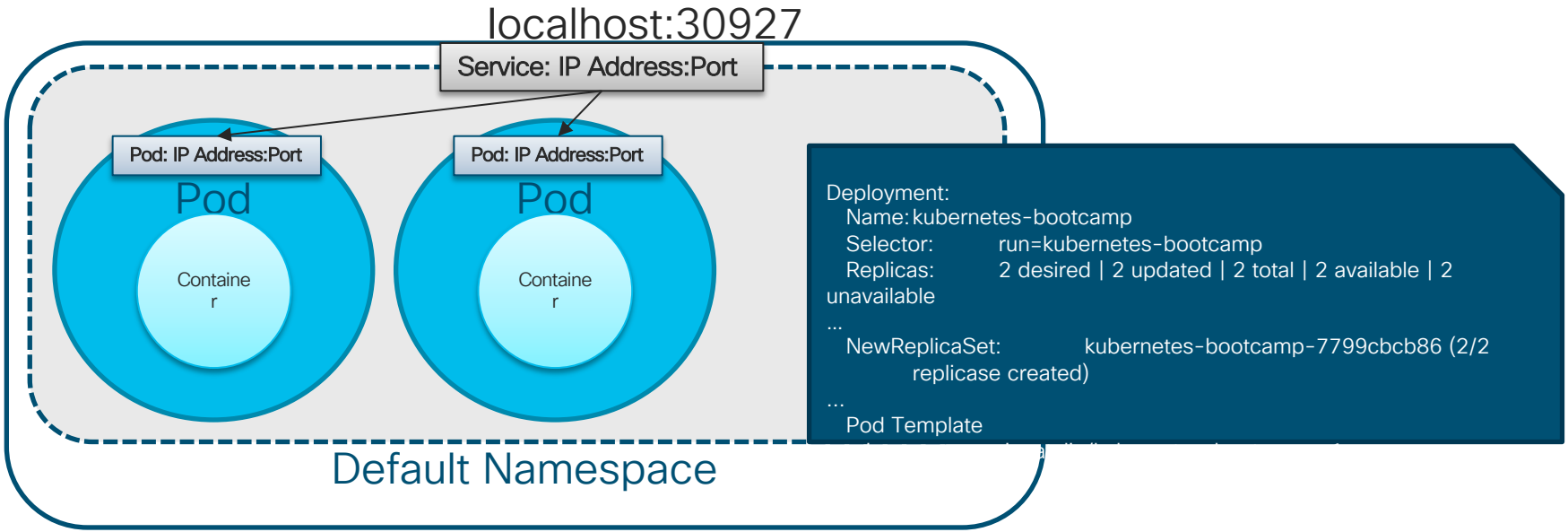
```
PS C:\CLUS19\LTRPRG-1200> kubectl exec -ti $POD_NAME curl localhost:8080
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-b94cb9bff-2lm54 | v=1
```

Scaling a Deployment

You make networking **possible**

Cisco live!

Scaling a deployment: Pod replicas




```
HEMORALE-M-80A4:nginx-deploy hectormorales$ kubectl get services -o wide
NAME                  TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)          AGE       SELECTOR
kubernetes            ClusterIP   10.96.0.1      <none>        443/TCP          28d       <none>
kubernetes-bootcamp   NodePort    10.106.65.63   <none>        8080:30927/TCP   21d       run=kubernetes-bootcamp
```

Step 5 – Scaling a Deployment

Commands

- In the previous step, we learned how to expose our Deployment using Services. In this section we will learn how to scale our application. First, let's check our deployment

```
$ kubectl get deployments
```

```
PS C:\CLUS19\LTRPRG-1200> kubectl get deployments
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp   1/1     1            1           66m
```

- We can see that our deployment has 1 Pod
  - DESIRED = number of desired replicas
  - CURRENT = how many replicas are running
  - UP-TO-DATE = number of replicas updated to get to the desired state
  - AVAILABLE = how many replicas are available to be used
- We may check the current Pods and Status

```
$ kubectl get pods
$ Kubectl get pods –o wide
```

```
PS C:\CLUS19\LTRPRG-1200> kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-b94cb9bff-2lm54   1/1     Running   0          68m
PS C:\CLUS19\LTRPRG-1200> kubectl get pods -o wide
NAME                                  READY   STATUS    RESTARTS   AGE   IP           NODE       NOMINATED NODE   READINESS GATES
kubernetes-bootcamp-b94cb9bff-2lm54   1/1     Running   0          68m   172.17.0.5   minikube   <none>           <none>
```

Step 5 – Scaling a Deployment

Commands

- Now, let's increase the number of replicas to 4 in the Deployment. We will use the kubectl scale command:

```
$ kubectl scale deployments/kubernetes-bootcamp --replicas=4
```

```
PS C:\CLUS19\LTRPRG-1200> kubectl scale deployments/kubernetes-bootcamp --replicas=4
deployment.extensions/kubernetes-bootcamp scaled
PS C:\CLUS19\LTRPRG-1200> kubectl get deployments
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
kubernetes-bootcamp   4/4     4            4           70m
PS C:\CLUS19\LTRPRG-1200> kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-b94cb9bff-2lm54   1/1     Running   0          70m
kubernetes-bootcamp-b94cb9bff-7dhnt   1/1     Running   0          14s
kubernetes-bootcamp-b94cb9bff-p7sfc   1/1     Running   0          14s
kubernetes-bootcamp-b94cb9bff-pkwxl   1/1     Running   0          13s
PS C:\CLUS19\LTRPRG-1200> kubectl get pods -o wide
NAME                                  READY   STATUS    RESTARTS   AGE   IP           NODE       NOMINATED NODE   READINESS GATES
kubernetes-bootcamp-b94cb9bff-2lm54   1/1     Running   0          70m   172.17.0.5   minikube   <none>           <none>
kubernetes-bootcamp-b94cb9bff-7dhnt   1/1     Running   0          20s   172.17.0.6   minikube   <none>           <none>
kubernetes-bootcamp-b94cb9bff-p7sfc   1/1     Running   0          20s   172.17.0.7   minikube   <none>           <none>
kubernetes-bootcamp-b94cb9bff-pkwxl   1/1     Running   0          19s   172.17.0.8   minikube   <none>           <none>
```

Step 5 – Scaling a Deployment

Commands

- You c…loyment comm…
  - $ kub…

```
PS C:\CLUS19\LTRPRG-1200> kubectl describe deployments/kubernetes-bootcamp
Name:                   kubernetes-bootcamp
Namespace:              default
CreationTimestamp:      Mon, 10 Jun 2019 19:23:53 -0500
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-b94cb9bff (4/4 replicas created)
Events:
  Type    Reason             Age   From                   Message
  ----    ------             ----  ----                   -------
  Normal  ScalingReplicaSet  2m9s  deployment-controller  Scaled up replica set kubernetes-bootcamp-b94cb9bff to 4
```

See the replicas messages!!

Step 5 – Scaling a Deployment

Commands

- From Step 4, we deleted the Service that expose our deployment. Let's create it back:

```
$ kubectl expose deployment/kubernetes-bootcamp --type="NodePort" --port 8080
$ kubectl get services
```

```
HEMORALE-M-80A4:~ hectormorales$ kubectl expose deployment/kubernetes-bootcamp --type="NodePort" --port 8080
service "kubernetes-bootcamp" exposed
HEMORALE-M-80A4:~ hectormorales$ kubectl get services
NAME                  TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)          AGE
kubernetes            ClusterIP   10.96.0.1      <none>        443/TCP          7d
kubernetes-bootcamp   NodePort    10.106.65.63   <none>        8080:30927/TCP   3s
```

- Now, let's update the NODE_PORT variable with the new port associated to the service just created

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

Step 5 – Scaling a Deployment

Commands

- Let's check if the service is load balancing between the 4 replicas we created. Every time you hit the command, there is a different replica running

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

```
HEMORALE-M-80A4:~ hectormorales$ curl $(minikube ip):$NODE_PORT
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-5c69669756-dnftl | v=1
HEMORALE-M-80A4:~ hectormorales$ curl $(minikube ip):$NODE_PORT
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-5c69669756-7jwjr | v=1
HEMORALE-M-80A4:~ hectormorales$ curl $(minikube ip):$NODE_PORT
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-5c69669756-j7785 | v=1
HEMORALE-M-80A4:~ hectormorales$ curl $(minikube ip):$NODE_PORT
Hello Kubernetes bootcamp! | Running on: kubernetes-bootcamp-5c69669756-dnftl | v=1
HEMORALE-M-80A4:~ hectormorales$
```

Step 5 – Scaling a Deployment

Commands

- Now, let's decrease the number of replicas to 2 in the Deployment:

```
$ kubectl scale deployments/kubernetes-bootcamp --replicas=2
$ kubectl get pods
$ kubectl  get pods -o wide
```

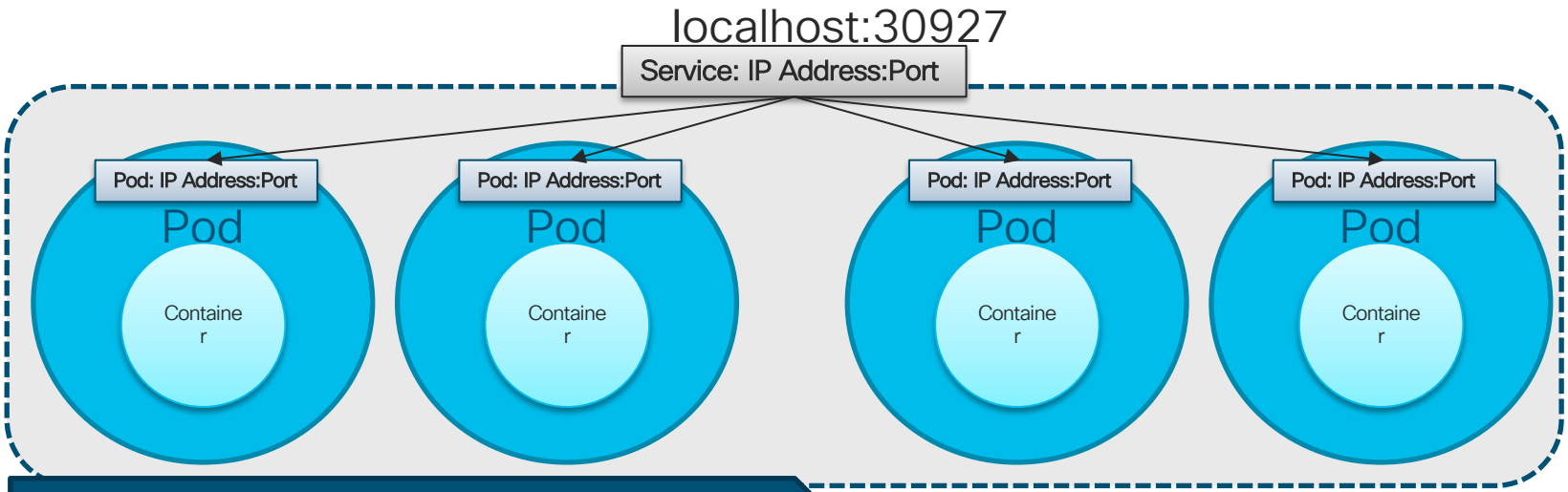
```
HEMORALE-M-80A4:~ hectormorales$ kubectl scale deployments/kubernetes-bootcamp --replicas=2
deployment.extensions "kubernetes-bootcamp" scaled
HEMORALE-M-80A4:~ hectormorales$ kubectl get pods
NAME                                  READY     STATUS        RESTARTS   AGE
kubernetes-bootcamp-5c69669756-7jwjr  1/1       Terminating   0          27m
kubernetes-bootcamp-5c69669756-dnftl  1/1       Running       0          27m
kubernetes-bootcamp-5c69669756-j7785  1/1       Running       2          4d
kubernetes-bootcamp-5c69669756-ld4wx  1/1       Terminating   0          27m
HEMORALE-M-80A4:~ hectormorales$
HEMORALE-M-80A4:~ hectormorales$
HEMORALE-M-80A4:~ hectormorales$
HEMORALE-M-80A4:~ hectormorales$ kubectl  get pods -o wide
NAME                                  READY     STATUS        RESTARTS   AGE       IP           NODE
kubernetes-bootcamp-5c69669756-7jwjr  1/1       Terminating   0          28m       172.17.0.5   minikube
kubernetes-bootcamp-5c69669756-dnftl  1/1       Running       0          28m       172.17.0.6   minikube
kubernetes-bootcamp-5c69669756-j7785  1/1       Running       2          4d        172.17.0.4   minikube
kubernetes-bootcamp-5c69669756-ld4wx  1/1       Terminating   0          28m       172.17.0.7   minikube
```

Updating your App

You make networking **possible**

Cisco live!

Updating your App




```
Deployment:
   Name: kubernetes-bootcamp
   Selector:          run=kubernetes-bootcamp
   Replicas:          2 desired | 2 updated | 2 total | 2 available | 2
unavailable
...
   NewReplicaSet:             kubernetes-bootcamp-7799cbcb86 (2
            replicas created)
...
   Pod Template
      Image:          jocatalin/kubernetes-bootcamp:v1
```

```
Deployment:
   Name: kubernetes-bootcamp
   Selector:          run=kubernees-bootcamp
   Replicas:          4 desired | 4 updated | 4 total | 4 available | 4
unavailable
...
   NewReplicaSet:             kubernetes-bootcamp-7799cbcb86 (2/4
            replicas created)
...
   Pod Template
      Image:          jocatalin/kubernetes-bootcamp:v2
```

Step 6 – Updating your App

Commands

- In the previous section, we learned how to scale in and out our deployment by decreasing or increasing the replica count. In this section we will learn how to update our app without any disruption. Let's check the current deployment status:

```
$ kubectl get pods
$ kubectl get deployment –o wide                    #describe the current image and deployment version
```

- Now, lets create version 2 of our image:

```
$ kubectl set image deployments/kubernetes-bootcamp kubernetes-bootcamp=jocatalin/kubernetes-bootcamp:v2
$ kubectl get pods                        #note each container id now is different to previous command
$ kubectl get deployment –o wide          #you will see the new running version
```

```
PS C:\CLUS19\LTRPRG-1200> kubectl rollout status deployment/kubernetes-bootcamp
deployment "kubernetes-bootcamp" successfully rolled out
PS C:\CLUS19\LTRPRG-1200> kubectl get pods -o wide
NAME                                  READY   STATUS    RESTARTS   AGE   IP           NODE       NOMINATED NODE   READINESS GATES
kubernetes-bootcamp-64bfc9b489-dzvxw   1/1     Running   0          10m   172.17.0.8   minikube   <none>           <none>
kubernetes-bootcamp-64bfc9b489-j55sl   1/1     Running   0          10m   172.17.0.7   minikube   <none>           <none>
PS C:\CLUS19\LTRPRG-1200> kubectl get deployment –o wide
Error from server (NotFound): deployments.extensions "–o" not found
Error from server (NotFound): deployments.extensions "wide" not found
PS C:\CLUS19\LTRPRG-1200> kubectl get deployment -o wide
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE    CONTAINERS            IMAGES                              SELECTOR
kubernetes-bootcamp   2/2     2            2           103m   kubernetes-bootcamp   jocatalin/kubernetes-bootcamp:v2   run=kubernetes-bootcamp
```

Step 6 – Updating your App

Commands

- Let's check the current service detail:

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

- It's running the 2 replicas. We expect it is running version 2 of the deployment. Before proceed, please check that your $NODE_PORT variable is updated. Otherwise please run the following commands:
  - `export NODE_PORT=$(kubectl get services/kubernetes-bootcamp -o go-template='{{(index .spec.ports 0).nodePort}}')`
  - `echo NODE_PORT=$NODE_PORT`

Step 6 – Updating your App

Commands

- Let's do a curl to the specified IP:Port. Alternatively you can put the URL on your browser (HINT: do a "minikube IP" if you don't remember the cluster IP). Do it several times to verify that version 2 is actually running

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

Step 6 – Updating your App

Commands

- We can confirm the rollout of the new version using:

```
$ kubectl rollout status deployment/kubernetes-bootcamp
```

- And now let's check the image id on the pods:

```
$ kubectl get pods -o wide
$ kubectl get deployment –o wide
```

```
PS C:\CLUS19\LTRPRG-1200> kubectl rollout undo deployments/kubernetes-bootcamp
deployment.extensions/kubernetes-bootcamp rolled back
PS C:\CLUS19\LTRPRG-1200> kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-64bfc9b489-dzvxw   1/1     Running   0          17m
kubernetes-bootcamp-64bfc9b489-j55sl   1/1     Running   0          17m
```

Step 6 – Updating your App

Commands

- Now, let's learn how Kubernetes deals with rollbacks. Let's do another update and deploy another image, say v10. Run these commands:

```
$ kubectl set image deployments/kubernetes-bootcamp kubernetes-bootcamp=gcr.io/google-samples/kubernetes-bootcamp:v10
$ kubectl get deployments
$ kubectl get pods
```

Step 6 – Updating your App

Commands

- What happened? Lets get more detail. Run:

```
$ kubectl describe pods
```

- Take a good look at the Events section in every container. We observe that the v10 image simply does not exist, then Kubernetes doesn't load it and is backing this off.

```
Events:
  Type     Reason     Age                From               Message
  ----     ------     ----               ----               -------
  Normal   Scheduled  87s                default-scheduler  Successfully assigned default/kubernetes-bootcamp-66ffd96587-dwf4s to minikube
  Normal   Pulling    41s (x3 over 86s)  kubelet, minikube  Pulling image "gcr.io/google-samples/kubernetes-bootcamp:v10"
  Warning  Failed     40s (x3 over 85s)  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:v10 not found
  Warning  Failed     40s (x3 over 85s)  kubelet, minikube  Error: ErrImagePull
  Normal   BackOff    13s (x5 over 85s)  kubelet, minikube  Back-off pulling image "gcr.io/google-samples/kubernetes-bootcamp:v10"
  Warning  Failed     13s (x5 over 85s)  kubelet, minikube  Error: ImagePullBackOff
```

Step 6 – Updating your App

Commands

- Let's rollback to the previous version:

```
$ kubectl rollout undo deployments/kubernetes-bootcamp
$ kubectl get pods
```

- You can see the events again if you run the `kubectl describe pods` command.

```
PS C:\CLUS19\LTRPRG-1200> kubectl rollout undo deployments/kubernetes-bootcamp
deployment.extensions/kubernetes-bootcamp rolled back
PS C:\CLUS19\LTRPRG-1200> kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
kubernetes-bootcamp-64bfc9b489-dzvxw  1/1     Running   0          17m
kubernetes-bootcamp-64bfc9b489-j55sl  1/1     Running   0          17m
```

Try and Play

You make networking **possible**

Cisco live!

Do it on your own using what you learned

- Your task is to create a web server app using node.js in a Docker container and your recently created Kubernetes Cluster in Minikube
- See Lab 7 on your Lab Guide and try on!
- I added an extra lab for you to understand more on Minikube. Try if you have time.
- Good luck!

Complete your online session evaluation

- Please complete your session survey after each session. Your feedback is very important.
- Complete a minimum of 4 session surveys and the Overall Conference survey (starting on Thursday) to receive your Cisco Live water bottle.
- All surveys can be taken in the Cisco Live Mobile App or by logging in to the Session Catalog on ciscolive.cisco.com/us.

Cisco Live sessions will be available for viewing on demand after the event at ciscolive.cisco.com.

Continue your education

Demos in the Cisco campus

Walk-in self-paced labs

Meet the engineer 1:1 meetings

Related sessions

Thank you

You make **possible**