



[View, add and edit your notes in the app](#)

# Lab 1 Video Overview Building Your First Cluster

Generated on December 3, 2023

## Summary

Notes

Screenshots

Bookmarks

7

29

0

The screenshot shows a browser window with two tabs open. The left tab is a terminal session on a Ubuntu 18.04 server (kube-01) with root privileges. It displays commands to download Google Cloud's public key and add the Kubernetes APT repository to /etc/apt/sources.list.d/kubernetes.list. The right tab is a help page titled 'Hands-on starts here. Step 1: Create a cluster' from play.instruct.com. It provides prerequisites (3 Ubuntu 18.04 servers with 40M RAM and private networking enabled), steps for setting up each server, and a checklist for running Kubernetes. The checklist includes steps like downloading the key, adding the repository, updating apt, installing kubelet, and marking it as a kubernetes node.

▶ 4:58

copied 1st and 2nd

and paste

▶ 4:58

```

root@kube-01:~# curl -fsSL /usr/share/keyrings/kubernetes-archive-keyring.gpg https://packages.cloud.google.com/apt/doc/apt-key.gpg
root@kube-01:~# echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg] https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee /etc/apt/sources.list.d/kubernetes.list
root@kube-01:~# apt update
Hit:1 http://europe-west1.gce.archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 http://europe-west1.gce.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:3 http://europe-west1.gce.archive.ubuntu.com/ubuntu bionic-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu bionic-security InRelease
Get:5 https://packages.cloud.google.com/apt/kubernetes-xenial/main amd64 Packages [49.4 kB]
Fetched 58.8 kB in 1s (81.1 kB/s)
Reading package lists...

```

▷ 5:11

```

root@kube-01:~# apt install -y kubelet kubeadm kubectl docker.io
Reading package lists... done
Building dependency tree
Reading state information... done
The following package was automatically installed and is no longer required:
  liblomak
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  bridge-utils contrack cri-tools kubernetes-cni pigz runc socat ubuntu-fan
Suggested packages:
  ifupdown rsf-tools groups-mount | group-lite debbootstrap docker-doc rimage zfs-fuse | rsfsutils
The following NEW packages will be installed:
  bridge-utils contrack cri-tools docker.io kubeadm kubectl kubernetes-cni pigz runc socat ubuntu-fan
0 upgraded, 144 newly installed, 0 to remove and 165 not upgraded.
Need to get 148 MB of archives.
After this operation, 704 MB of additional disk space will be used.
Get:1 https://packages.cloud.google.com/apt/kubernetes-xenial/main amd64 kubelet amd64 2.4.1 [57.4 kB]
Get:2 http://europe-west1.gce.archive.ubuntu.com/ubuntu bionic/main amd64 bridge-utils amd64 1.5.1ubuntu1 [30.1 kB]
Get:3 http://europe-west1.gce.archive.ubuntu.com/ubuntu bionic/main amd64 contrack amd64 1.1.4.4+snapshot20161117~ubuntu2 [30.6 kB]
Get:4 http://europe-west1.gce.archive.ubuntu.com/ubuntu bionic-updates/main amd64 contrack amd64 1.1.4.4+snapshot20161117~ubuntu2_18.04.2 [4887 kB]
Get:5 http://europe-west1.gce.archive.ubuntu.com/ubuntu bionic-backports/main amd64 contrack amd64 1.5.2.1~ubuntu18.04.2 [32.8 kB]
Get:6 https://packages.cloud.google.com/apt/kubernetes-xenial/main amd64 cri-tools amd64 1.13.0-0~[875 kB]
Get:7 https://packages.cloud.google.com/apt/kubernetes-xenial/main amd64 docker.io amd64 2.4.1-0~[25.0 kB]
Get:8 https://packages.cloud.google.com/apt/kubernetes-xenial/main amd64 kubeadm amd64 0.8.7-0~[25.0 kB]
Get:9 https://packages.cloud.google.com/apt/kubernetes-xenial/main amd64 kubectl amd64 1.22.1-0~[21.8 kB]
Get:10 https://packages.cloud.google.com/apt/kubernetes-xenial/main amd64 runc amd64 1.7.3-2~[342 kB]
Get:11 http://europe-west1.gce.archive.ubuntu.com/ubuntu bionic/main amd64 socat amd64 0.10.12.1-0~[341 kB]
Get:12 http://europe-west1.gce.archive.ubuntu.com/ubuntu bionic/main amd64 ubuntu-fan all 0.12.2-0~[341 kB]
Get:13 http://europe-west1.gce.archive.ubuntu.com/ubuntu bionic/main amd64 udev amd64 245-1~[341 kB]
Get:14 https://packages.cloud.google.com/apt/kubernetes-xenial/main amd64 kubeadm amd64 1.22.1.0~[8717 kB]
Fetched 148 MB in 4s (34.2 MB/s)
Preconfiguring packages...
Selecting previously unselected package pigz.
(Reading database ... 85648 files and directories currently installed.)
Preparing to unpack .../00_pigz_2.4.1_amd64.deb ...
Unpacking pigz (2.4.1) ...
Selecting previously unselected package bridge-utils.
Preparing to unpack .../01_bridge-utils_1.5.1ubuntu1_amd64.deb ...
Unpacking bridge-utils (1.5.1ubuntu1) ...
Selecting previously unselected package contrack.
Preparing to unpack .../02_contrack_1.1_4.4+snapshot20161117~ubuntu2_amd64.deb ...
Unpacking contrack (1.1_4.4+snapshot20161117~ubuntu2) ...
Selecting previously unselected package runc.
Preparing to unpack .../03_runc_1.0.0-95~ubuntu18.04.2 ...
Unpacking runc (1.0.0-95~ubuntu18.04.2) ...
Selecting previously unselected package kubeadm.
Preparing to unpack .../04_kubeadm_0.8.7-0~ ...
Unpacking kubeadm (0.8.7-0~) ...
Selecting previously unselected package kubectl.
Preparing to unpack .../05_kubectl_1.22.1-0~ ...
Unpacking kubectl (1.22.1-0~) ...
Selecting previously unselected package docker.io.
Preparing to unpack .../06_docker.io_2.4.1_amd64.deb ...
Unpacking docker.io (2.4.1) ...
Selecting previously unselected package ubuntu-fan.
Preparing to unpack .../07_ubuntu-fan_0.12.2-0~ ...
Unpacking ubuntu-fan (0.12.2-0~) ...
Selecting previously unselected package cri-tools.
Preparing to unpack .../08_cri-tools_1.13.0-0~ ...
Unpacking cri-tools (1.13.0-0~) ...
Selecting previously unselected package socat.
Preparing to unpack .../09_socat_1.7.3-2~ ...
Unpacking socat (1.7.3-2~) ...
Selecting previously unselected package udev.
Preparing to unpack .../10_udev_245-1 ...
Unpacking udev (245-1) ...

```

▷ 5:20

**cube ctl : access into our cluster**

▷ 5:28

now we do the exact same thing on Kube-02 and Kube-03

▷ 5:55

```
Hands-on starts here. Step 1: Click here to start the lab

https://play.instruct.com/vanri/tracks/lab-1-kubernetes-concepts-and-building-a-k8s-cluster/challenges/setup-core/assignment

Control Plane > kube-02 > kube-03

Preparing to unpack .../05-crl-tools_1.13.0-01_amd64.deb ...
Unpacking crl-tools (1.13.0-01) ...
Selecting previously unpacked package docker-in.
Preparing to unpack .../06_docker.io_1.20.7-7~ubuntus1=1.20.7-8_amd64.deb ...
Unpacking docker.io (1.20.7-7~ubuntus1=1.20.7-8_amd64) ...
Selecting previously unpacked package kubelet=1.22.1-00_amd64-cni.
Preparing to unpack .../07_kubernetes-cni_0.8.7-00_amd64.deb ...
Unpacking kubernetes-cni (0.8.7-00) ...
Selecting previously unpacked package socat.
Preparing to unpack .../08_socat_1.7.3-2~ubuntus2_amd64.deb ...
Unpacking socat (1.7.3-2~ubuntus2) ...
Selecting previously unpacked package kubelet.
Preparing to unpack .../09_kubelet_1.22.1-00_amd64.deb ...
Unpacking kubelet (1.22.1-00) ...
Selecting previously unpacked package kubelet.
Preparing to unpack .../10_kubelet_1.22.1-00_amd64.deb ...
Unpacking kubelet (1.22.1-00) ...
Selecting previously unpacked package kubelet.
Preparing to unpack .../11_kubelet_1.22.1-00_amd64.deb ...
Unpacking kubelet (1.22.1-00) ...
Selecting previously unpacked package ubuntu-fan.
Preparing to unpack .../12_ubuntu-fan_0.12.10_all.deb ...
Unpacking ubuntu-fan (0.12.10) ...
Setting up kubelet (1.22.1-00) ...
Setting up kubernetes-cni (0.8.7-00) ...
Setting up runit (1.9.0-0rc5~Ubuntu-18.04.2) ...
Setting up socat (1.7.3-2~ubuntus2) ...
Setting up socat (1.7.3-2~ubuntus2) ...
Setting up contained (1.5.2~ubuntu18.04.2) ...
Created symlink /etc/systemd/system/multi-user.target.wants/containerd.service → /lib/systemd/system/containerd.service.
Setting up containerd (1.5.2~ubuntu18.04.2) ...
Setting up kubelet (1.22.1-00) ...
Created symlink /etc/systemd/system/multi-user.target.wants/kubelet.service → /lib/systemd/system/kubelet.service.
Setting up kubelet (1.22.1-00) ...
Created symlink /etc/systemd/system/multi-user.target.wants/ubuntu-fan.service → /lib/systemd/system/ubuntu-fan.service.
Setting up kubelet (1.22.1-00) ...
Setting up kubelet (1.22.1-00) ...
Setting up kubelet (1.22.1-00) ...
Adding group docker (GID 116) ...
Creating socket ...
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /lib/systemd/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /lib/systemd/system/docker.socket.
Processing triggers for systemd (237-1ubuntu14.40) ...
Processing triggers for libsystemd-dbus (1.10.0-2~ubuntus1) ...
Processing triggers for man-db (2.8.3-2~ubuntus1) ...
Processing triggers for unbound (0.6.30-0.2) ...
[unbound@kube-01 ~]# kubectl
```

▶ 6:40

successfully installed

▶ 6:46

Hands-on starts here: Step 1: Go to [play.instruct.com](#)

Control Plane > kube-02 > kube-03

scale Set a new size for a deployment, replica set, or replication controller  
autoscale Auto-scale a deployment, replica set, stateful set, or replication controller

Cluster Management Commands:  
certificate Create self-signed certificates.  
cluster-info Display cluster information  
top Display resource (CPU/memory) usage  
cordon Mark nodes as unschedulable  
eviction Evict pods from a node  
drain Drain node in preparation for maintenance  
taint Update the taints on one or more nodes

Troubleshooting and Debugging Commands:  
describe Show details of a specific resource or group of resources  
logs Print the logs for a container in a pod  
attach Attach to a running container  
exec Execute a command in a container  
port-forward Forward one or more local ports to a pod  
proxy Proxy to the Kubernetes API server  
cp Copy files and directories to and from containers  
auth Inspect authorization  
debug Create debugging sessions for troubleshooting workloads and nodes

Advanced Commands:  
diff Diff the live version against a would-be applied version  
apply Apply a configuration to a resource by file name or stdin  
patch Patch a resource in place  
replace Replace a resource by file name or stdin  
wait Experimental: Wait for a specific condition on one or many resources  
kustomize Build a kustomization target from a directory or URL

Settings Commands:  
label Update the labels on a resource  
annotate Update the annotations on a resource  
completion Output shell completion code for the specified shell (bash or zsh)

Other Commands:  
api-resources Print the supported API resources on the server  
api-versions Print the supported API versions on the server, in the form of "group/version"  
config Modify kubeconfig files  
plugin Provides utilities for interacting with plugins  
version Print the client and server version information

Usage:  
kubectl [flags] [options]

Use "kubectl command --help" for more information about a given command.  
Use "kubectl options" for a list of global command-line options (applies to all commands).

root@kube-01:~#

We have already provisioned the servers for you where you can access by clicking on the tabs in the upper left corner (Control Plane, kube-02, kube-03)

Step 1 - Set up each server in the cluster to run Kubernetes.

On each of the three Ubuntu 18.04 servers run the following commands as root, you can type the commands or highlight and copy/paste from the terminal window on the left:

- Download the Google Cloud public signing key:  
`curl -fsSL https://storage.googleapis.com/kubernetes-archive-keyring/gpg https://packages.cloud.google.com/apt/doc/gpg-key.gpg`
- Add the Kubernetes apt repository:  
`echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg] https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee /etc/apt/sources.list.d/kubernetes.list`
- Update apt package index with the new repository and install the utilities:  
`apt update  
apt install -y kubectl kubeadm kubectl dockershim  
apt-mark hold kubectl kubeadm kubectl`

▪ **kubeadm**: The primary agent that runs on a kubernetes node and communicates with the control plane to lease an IP address and report status

▪ **Kubelet**: A tool that performs the actions necessary to build up a Kubernetes cluster

▪ **Kubectl**: The command tool that lets you control Kubernetes clusters and issue commands to the api-server

▪ **Docker Engine**: A container engine used by Kubernetes to run workloads

**Close** **Skip** **Check**

▶ 6:46

when the install is completed on all three

click on check button and continue to build the cluster

▶ 7:28

Hands-on starts here: Step 1: C... playinstruct.com/casten/tracks/lab-1-kubernetes-concepts-and-building-a-k8s-cluster/challenges/setup-core/assignment

Control Plane > kube-02 > kube-03

```

Preparing to unpack .../0/crl-tools_1.13.0-0_amd64.deb ...
Unpacking crt-tools (1.13.0-0) ...
Selecting previously unselected package docker.io.
Preparing to unpack .../1/docker.io_19.03.1~3-0ubuntu1~18.04.1_amd64.deb ...
Unpacking docker.io (19.03.1~3-0ubuntu1~18.04.1) ...
Selecting previously unselected package kubernetes-cni.
Preparing to unpack .../2/kubernetes-cni_0.8.8-7~0_amd64.deb ...
Unpacking kubernetes-cni (0.8.8-7~0) ...
Selecting previously unselected package socat.
Preparing to unpack .../3/socat_1.7.3-2_amd64.deb ...
Unpacking socat (1.7.3-2) ...
Selecting previously unselected package kubelet.
Preparing to unpack .../4/kubelet_1.22.1-0_amd64.deb ...
Unpacking kubelet (1.22.1-0) ...
Selecting previously unselected package kubeadm.
Preparing to unpack .../5/kubeadm_1.22.1-0_amd64.deb ...
Unpacking kubeadm (1.22.1-0) ...
Selecting previously unselected package ubuntu-fan.
Preparing to unpack .../6/ubuntu-fan_0.12.10_all.deb ...
Unpacking ubuntu-fan (0.12.10) ...
Setting up coreutils (1:8.30-0ubuntu17.6ubuntu2) ...
Setting up debconf (0.8.7.0) ...
Setting up runit (1.8.0-rc5~Ubuntu18.04.2) ...
Setting up crt-tools (1.13.0-0) ...
Setting up kubernetes-cni (0.8.8-7~0) ...
Setting up containerd (1.5.2-2~Ubuntu-0.0.2) ...
Created symlink /etc/systemd/system/multi-user.target.wants/containerd.service → /lib/systemd/system/containerd.service.
Setting up kubelet (1.22.1-0) ...
Setting up kubeadm (1.22.1-0) ...
Created symlink /etc/systemd/system/multi-user.target.wants/kubelet.service → /lib/systemd/system/kubelet.service.
Setting up ubuntu-fan (0.12.10) ...
Creating /etc/systemd/system/multi-user.target.wants/ubuntu-fan.service → /lib/systemd/system/ubuntu-fan.service.
Setting up kubelet (1.22.1-0) ...
Setting up kubeadm (1.22.1-0) ...
Setting up podman (2.0.0-1~Ubuntu-0.0.1) ...
Setting up docker.io (19.03.1~3-0ubuntu1~18.04.1) ...
Adding group 'docker' (GID 116) ...
Done.
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /lib/systemd/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /lib/systemd/system/docker.socket.
Processing triggers for systemd (237-3ubuntu0.44) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Processing triggers for ureadahead (0.100.0-21) ...
root@kube-03: ~ []

```

▶ 7:28



▶ 7:42

```
root@kube-01:~# mkdir /etc/docker
mkdir: cannot create directory '/etc/docker': File exists
root@kube-01:~# cat <>EOF | sudo tee /etc/docker/daemon.json
{
    "exec-opts": ["native.cgroupdriver=systemd"],
    "log-driver": "json-file",
    "log-opts": {
        "max-size": "100m"
    },
    "storage-driver": "overlay2"
}
EOF
root@kube-01:~# systemctl enable docker
root@kube-01:~# systemctl daemon-reload
root@kube-01:~# systemctl restart docker
```

Step 2: Setup the Kubernetes Control Plane

Next, we are going to configure the Docker daemon to use systemd for the management of the container's cgroups. Copy the following block and paste it in the terminal window of each node (Control Plane, kube-02, kube-03):

```
mkdir /etc/docker
cat <>EOF | sudo tee /etc/docker/daemon.json
{
    "exec-opts": ["native.cgroupdriver=systemd"],
    "log-driver": "json-file",
    "log-opts": {
        "max-size": "100m"
    },
    "storage-driver": "overlay2"
}
EOF
```

systemctl enable docker  
systemctl daemon-reload  
systemctl restart docker

On the Control Plane node only, run this command in order to initialize the Kubernetes control plane:

```
kubeadm init --ignore-preflight-errors=all
```

This can take a minute or two to run.

Before we begin using the cluster, we need to configure the `kubeadm` command line tool that lets us control the cluster. For configuration, `kubeadm` looks for a file named `config` in the `$HOME/.kube` directory.

- Create the directory:

```
mkdir -p $HOME/.kube
```

**Close** **Skip** **Check**

▶ 8:14

```
root@kube-02:~# mkdir /etc/docker
mkdir: cannot create directory '/etc/docker': File exists
root@kube-02:~# cat <>EOF | sudo tee /etc/docker/daemon.json
{
    "exec-opts": ["native.cgroupdriver=systemd"],
    "log-driver": "json-file",
    "log-opts": {
        "max-size": "100m"
    },
    "storage-driver": "overlay2"
}
EOF
root@kube-02:~# systemctl enable docker
root@kube-02:~# systemctl daemon-reload
root@kube-02:~# systemctl restart docker
root@kube-02:~# 
```

Step 2: Setup the Kubernetes Control Plane

Next, we are going to configure the Docker daemon to use systemd for the management of the container's cgroups. Copy the following block and paste it in the terminal window of each node (Control Plane, kube-02, kube-03):

```
mkdir /etc/docker
cat <>EOF | sudo tee /etc/docker/daemon.json
{
    "exec-opts": ["native.cgroupdriver=systemd"],
    "log-driver": "json-file",
    "log-opts": {
        "max-size": "100m"
    },
    "storage-driver": "overlay2"
}
EOF
```

systemctl enable docker  
systemctl daemon-reload  
systemctl restart docker

On the Control Plane node only, run this command in order to initialize the Kubernetes control plane:

```
kubeadm init --ignore-preflight-errors=all
```

This can take a minute or two to run.

Before we begin using the cluster, we need to configure the `kubeadm` command line tool that lets us control the cluster. For configuration, `kubeadm` looks for a file named `config` in the `$HOME/.kube` directory.

- Create the directory:

```
mkdir -p $HOME/.kube
```

**Close** **Skip** **Check**

▶ 8:23

```

root@kube-02:~# mkdir /etc/docker
mkdir: cannot create directory '/etc/docker': File exists
root@kube-02:~# cat <>EOF | sudo tee /etc/docker/daemon.json
{
    "exec-opts": ["native.cgroupdriver=systemd"],
    "log-driver": "json-file",
    "log-opt": {
        "max-size": "100m"
    },
    "storage-driver": "overlay2"
}
EOF
root@kube-01:~#
root@kube-01:~# systemctl enable docker
root@kube-01:~# systemctl daemon-reload
root@kube-01:~# systemctl restart docker
root@kube-01:~#

```

**Step 2: Setup the Kubernetes Control Plane**

Next, we are going to configure the Docker daemon to use systemd for the management of the container's cgroups. Copy the following block and paste it in the terminal window of each node (control Plane, kube-02, kube-03).

```

mkdir /etc/docker
cat <>EOF | sudo tee /etc/docker/daemon.json
{
    "exec-opts": ["native.cgroupdriver=systemd"],
    "log-driver": "json-file",
    "log-opt": {
        "max-size": "100m"
    },
    "storage-driver": "overlay2"
}
EOF

```

systemctl enable docker  
systemctl daemon-reload  
systemctl restart docker

On the Control Plane node only, run this command in order to initialize the Kubernetes control plane:

```

kubeadm init --ignore-preflight-errors=all

```

This can take a minute or two to run.

Before we begin using the cluster, we need to configure the `kubecfg` command line tool that lets us control the cluster. For configuration, kubect looks for a file named `config` in the `$HOME/.kube` directory.

- Create the directory.

```

mkdir -p $HOME/.kube

```

▶ 8:48

```

root@kube-02:~# cat <>EOF | sudo tee /etc/docker/daemon.json
{
    "exec-opts": ["native.cgroupdriver=systemd"],
    "log-driver": "json-file",
    "log-opt": {
        "max-size": "100m"
    },
    "storage-driver": "overlay2"
}
EOF
root@kube-01:~#
root@kube-01:~# systemctl enable docker
root@kube-01:~# systemctl daemon-reload
root@kube-01:~#
root@kube-01:~# kubeadm init --ignore-preflight-errors=all
[init] Using Kubernetes version: v1.22.3
[preflight] Running pre-flight checks
[preflight] Pulling images required to set up a Kubernetes Cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action in beforehand using 'kubeadm config images pull'
[certs] Using certificate-dir folder "/etc/kubernetes/pki"
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [kube-01 localhost] and IPs [10.96.0.1 10.132.0.33]
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "etcd-peer" certificate and key
[certs] Generating "etcd/serve" certificate and key
[certs] etcd/serve serving cert is signed for DNS names [kube-01 localhost] and IPs [10.132.0.33 127.0.0.1 ::1]
[certs] etcd/peer serving cert is signed for DNS names [kube-01 localhost] and IPs [10.132.0.33 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "apiserver-etcd-peer" certificate and key
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
[kubeconfig] Writing "admin.conf" kubeconfig file
[kubeconfig] Writing "kublet.conf" kubeconfig file

```

**Step 2: Setup the Kubernetes Control Plane**

Next, we are going to configure the Docker daemon to use systemd for the management of the container's cgroups. Copy the following block and paste it in the terminal window of each node (control Plane, kube-02, kube-03).

```

mkdir /etc/docker
cat <>EOF | sudo tee /etc/docker/daemon.json
{
    "exec-opts": ["native.cgroupdriver=systemd"],
    "log-driver": "json-file",
    "log-opt": {
        "max-size": "100m"
    },
    "storage-driver": "overlay2"
}
EOF

```

systemctl enable docker  
systemctl daemon-reload  
systemctl restart docker

On the Control Plane node only, run this command in order to initialize the Kubernetes control plane:

```

kubeadm init --ignore-preflight-errors=all

```

This can take a minute or two to run.

Before we begin using the cluster, we need to configure the `kubecfg` command line tool that lets us control the cluster. For configuration, kubect looks for a file named `config` in the `$HOME/.kube` directory.

- Create the directory.

```

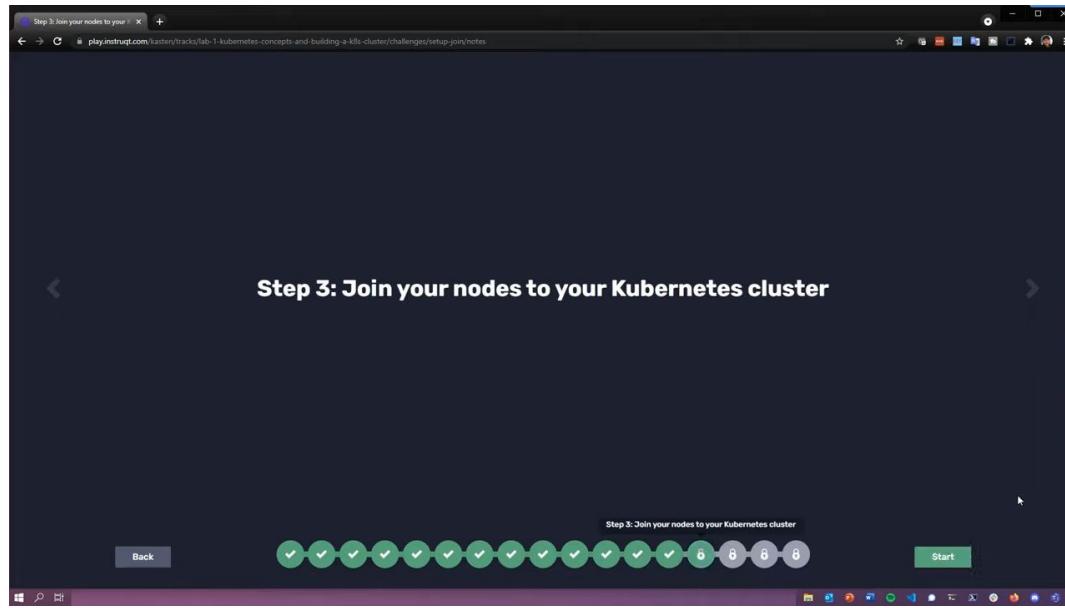
mkdir -p $HOME/.kube

```

▶ 9:30

▶ 10:56

▶ 11:18



▶ 11:30

```
root@kube-01:~# kubeadm token create --print-join-command
kubeadm join 10.132.0.33:6443 --token 10s39h.qe311vpnl7b1pg7 --discovery-token-ca-cert-hash sha256:b0f6deecf7ea5997aca7e5ba29230b0531b3fb700a663ec1156b7148ea74c3a8c
root@kube-01:~#
```

Step 3: Join your nodes to your Kubernetes cluster

You can now join any number of machines by running the `kubeadm join` command on each node as root.

To join `kube-02` and `kube-03` as worker nodes to the cluster, we need the token generated by the `kubeadm init` command.

Run the following command on the `Control Plane` node:

```
kubeadm token create --print-join-command
```

Copy the output and run it on both worker nodes (`kube-02`, `kube-03`).

When you join your worker nodes you will see the following output:

```
This node has joined the cluster:
  * Certificate signing request was sent to Control Plane and a response was received.
  * The Kubelet was informed of the new secure connection details.
```

To check that all nodes are now joined to the `Control Plane` run the following command on the Kubernetes Control Plane node:

```
kubectl get nodes
```

The successful result will look like this:

NAME	STATUS	ROLES	AGE	VERSION
kube-01	NotReady	master	8m	v1.x.x
kube-02	NotReady	<none>	6m	v1.x.x
kube-03	NotReady	<none>	6m	v1.x.x

Close Skip Check

▶ 12:33

```

Step 3: Join your nodes to your Kubernetes cluster
root@kube-01:~# kubeadm token create --print-join-command
kubeadm join 10.132.0.33:6443 --token 10539h.qw111vpn17bjpg7 --discovery-token-ca-cert-hash sha256:b96deecf7ea5597aca7e5ba292340d531b3fb700a663ec1156b7148ea74c3ab
root@kube-01:~#

```

The terminal shows the command `kubeadm token create --print-join-command` being run on the Control Plane node. The output includes the token value and its expiration details.

▶ 12:47

copy the output that is generated from the above command and paste in Kube-02 and Kube-03

▶ 12:47

```

Step 3: Join your nodes to your Kubernetes cluster
root@kube-02:~# kubeadm join 10.132.0.33:6443 --token 10539h.qw111vpn17bjpg7 --discovery-token-ca-cert-hash sha256:b96deecf7ea5597aca7e5ba292340d531b3fb700a663ec1156b7148ea74c3ab
root@kube-02:~# kubectl get nodes
NAME     STATUS   ROLES    AGE     VERSION
kube-01  NotReady control-plane,master 3m33s  v1.22.x
kube-02  NotReady <none>        29s   v1.22.x
kube-03  NotReady <none>        15s   v1.22.x
root@kube-02:~#

```

The terminal shows the command `kubeadm join` being run on Kube-02. The output indicates that the node has joined the cluster.

▶ 13:18

you will see two other working nodes

▷ 13:19

Step 3: Join your nodes to your cluster

Kubelet token created for node kube-01: 10.132.0.33:4443 --token 10b39h.gq311vpnl78ppg7 -discovery-token-ca-cert-hash sha256:b96deef7ea559aca7e5ba292300d531bf700a663ec1156b7148ea74c3a8c

```
root@kube-01:~# kubeadm token create --print-join-command
kubeadm join 10.132.0.33:4443 --token 10b39h.gq311vpnl78ppg7 -discovery-token-ca-cert-hash sha256:b96deef7ea559aca7e5ba292300d531bf700a663ec1156b7148ea74c3a8c
root@kube-01:~# kubectl get nodes
NAME     STATUS    ROLES      AGE   VERSION
kube-01   NotReady   control-plane,master   36m33s   v1.22.1
kube-02   NotReady   <none>      28s   v1.22.1
kube-03   NotReady   <none>      15s   v1.22.1
root@kube-01:~#
```

To join kube-02 and kube-03 as worker nodes to the cluster, we need the token generated by the `kubeadm join` command.

Run the following command on the `Control Plane` node:

```
kubeadm token create --print-join-command
```

Copy the output and run it on both worker nodes (kube-02, kube-03).

When you join your worker nodes you will see the following output:

This node has joined the cluster:

- Certificate signing request was sent to Control Plane and a response was received.
- The Kubelet was informed of the new secure connection details.

To check that all nodes are now joined to the `Control Plane` run the following command on the Kubernetes Control Plane node:

```
kubectl get nodes
```

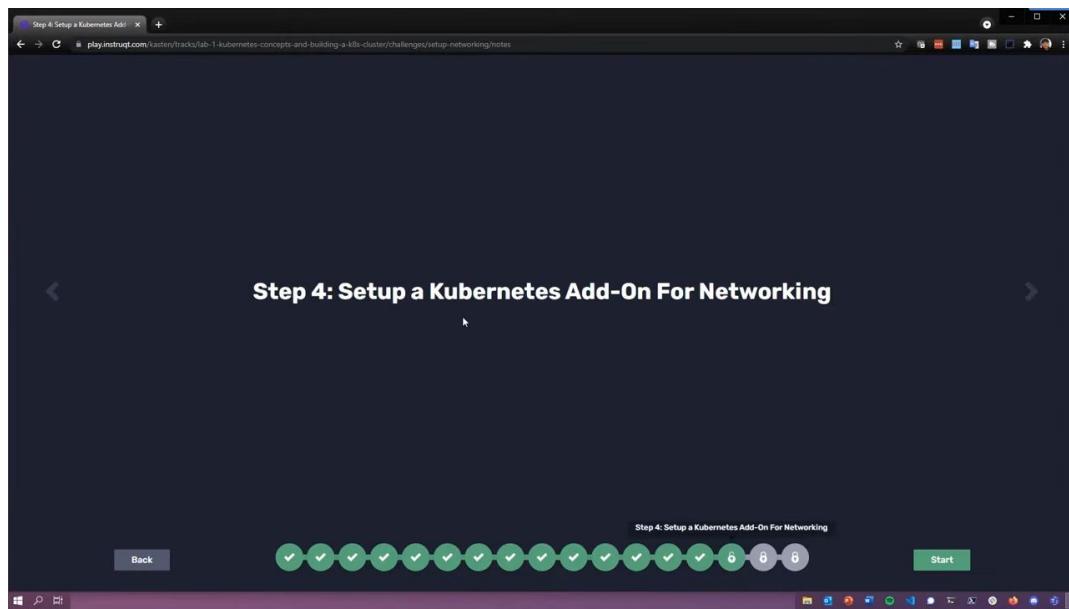
The successful result will look like this:

NAME	STATUS	ROLES	AGE	VERSION
kube-01	NotReady	master	8m	v1.x.x
kube-02	NotReady	<none>	0s	v1.x.x
kube-03	NotReady	<none>	0s	v1.x.x

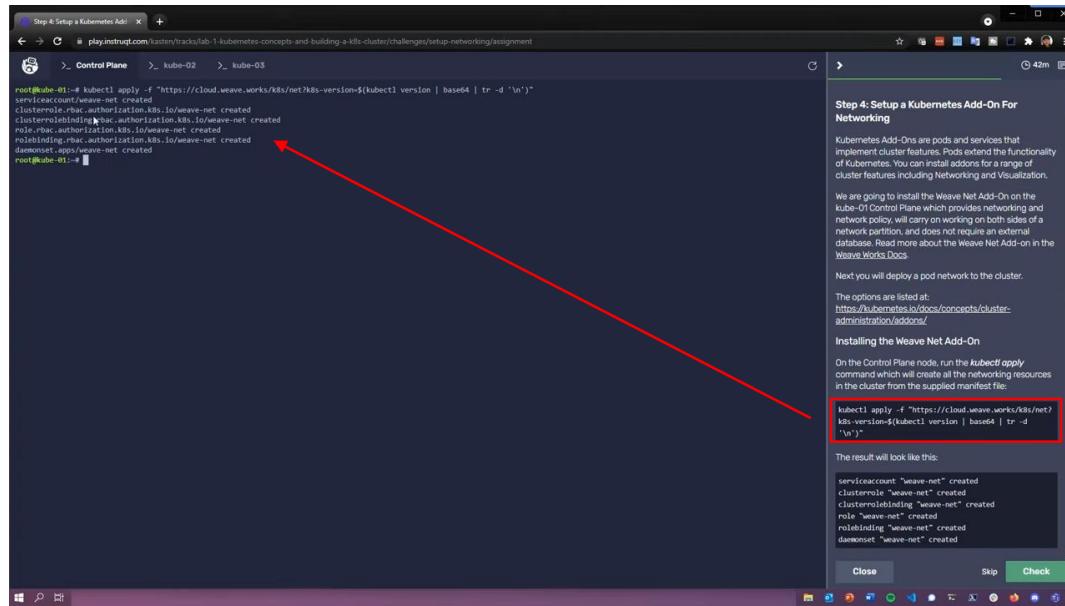
Status will remain `NotReady` until we configure networking on the `Control Plane` node in the next challenge.

Click the `Check` button to continue...

▷ 13:31



▷ 13:54



```

root@kube-01:~# kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')"
serviceaccount/weave-net created
clusterrolebinding.k8s.io/weave-net created
clusterrolebinding.rbac.authorization.k8s.io/weave-net created
role.rbac.authorization.k8s.io/weave-net created
rolebinding.rbac.authorization.k8s.io/weave-net created
daemonset.apps/weave-net created
root@kube-01:~#

```

**Step 4: Setup a Kubernetes Add-On For Networking**

Kubernetes Add-Ons are pods and services that implement cluster features and extend the functionality of Kubernetes. You can install add-ons for a range of cluster features including Networking and Visualization.

We are going to install the Weave Net Add-On on the kube-control-plane which provides network and network policy, will be working on both sides of a network partition, and does not require an external database. Read more about the Weave Net Add-on in the [Weave Works Docs](#).

Next you will deploy a pod network to the cluster.

The options are listed at: <https://kubernetes.io/docs/concepts/cluster-administration/addons/>

**Installing the Weave Net Add-On**

On the Control Plane node, run the `kubectl apply` command which will create all the networking resources in the cluster from the supplied manifest file.

```

kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')"

```

The result will look like this:

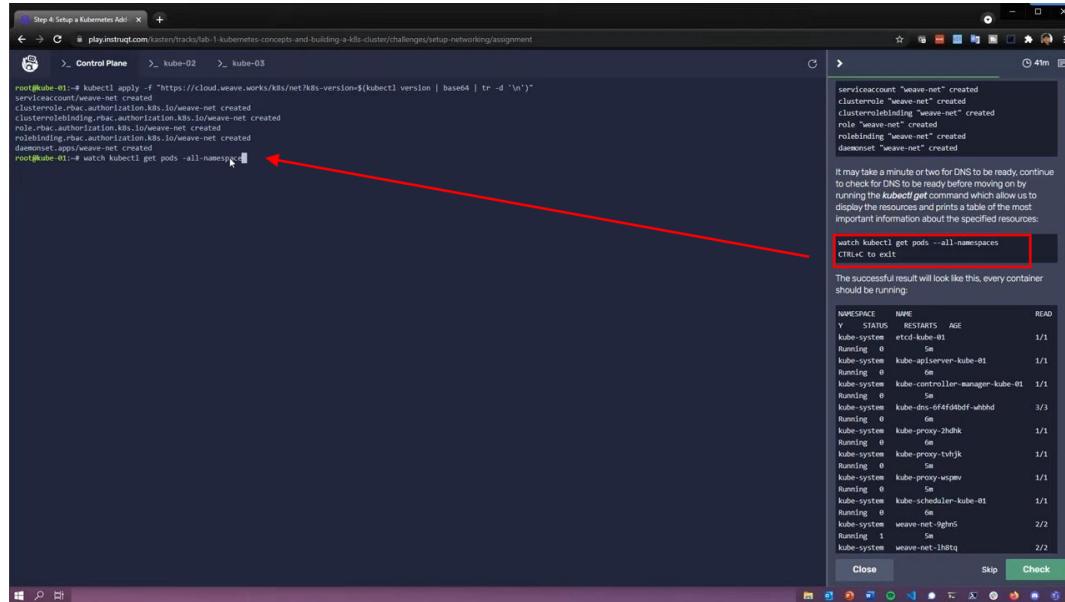
```

serviceaccount "weave-net" created
clusterrole "weave-net" created
clusterrolebinding "weave-net" created
role "weave-net" created
rolebinding "weave-net" created
daemonset "weave-net" created

```

**Close Skip Check**

▶ 15:20



```

root@kube-01:~# kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')"
serviceaccount/weave-net created
clusterrolebinding.k8s.io/weave-net created
clusterrolebinding.rbac.authorization.k8s.io/weave-net created
role.rbac.authorization.k8s.io/weave-net created
rolebinding.rbac.authorization.k8s.io/weave-net created
daemonset.apps/weave-net created
root@kube-01:~# watch kubectl get pods --all-namespaces

```

**Step 4: Setup a Kubernetes Add-On For Networking**

It may take a minute or two for DNS to be ready, continue to check for DNS to be ready before moving on by running the `kubectl get` command which allow us to display the resources and prints a table of the most important information about the specified resources.

```

watch kubectl get pods --all-namespaces
CTRL+C to exit

```

The successful result will look like this, every container should be running:

NAMESPACE	NAME	STATUS	RESTARTS	AGE	READ
kube-system	etcd-kube-01	Running	0	5m	1/1
kube-system	kube-apiserver-kube-01	Running	0	5m	1/1
kube-system	kube-controller-manager-kube-01	Running	0	5m	1/1
kube-system	kube-dns-6f4fd6bf-whhbd	Running	0	5m	3/3
kube-system	kube-node-local-2hdhk	Running	0	5m	1/1
kube-system	kube-proxy-tvhjk	Running	0	5m	1/1
kube-system	kube-proxy-wsper	Running	0	5m	1/1
kube-system	kube-scheduler-kube-01	Running	0	5m	1/1
kube-system	weave-net-9ghn5	Running	1	5m	2/2
kube-system	weave-net-1h8tq	Running	1	5m	2/2

**Close Skip Check**

▶ 15:51

```

Step 4: Setup a Kubernetes Add-on
Control Plane > kube-02 > kube-03
Every 0.6s: kubectl get pods --all-namespaces
NAME      NAME    STATUS  RESTARTS  AGE
kube-system   Coredns-78f4d69578-71exc9  1/1   Running   0   6m20s
kube-system   coredns-78f4d69578-k2ph8  1/1   Running   0   6m20s
kube-system   etcd-kube-01  1/1   Running   0   6m24s
kube-system   kube-apiserver-kube-01  1/1   Running   0   6m24s
kube-system   kube-controller-manager-kube-01  1/1   Running   0   6m25s
kube-system   kube-proxy-z2ov  1/1   Running   0   3m10s
kube-system   kube-proxy-gcf0  1/1   Running   0   3m10s
kube-system   kube-proxy-hz2  1/1   Running   0   6m20s
kube-system   kube-scheduler-kube-01  1/1   Running   0   6m26s
kube-system   weave-net-jm6g  2/2   Running   1 (48s ago)  47s
kube-system   weave-net-qjso  2/2   Running   1 (39s ago)  47s
kube-system   weave-net-rvng  2/2   Running   1 (39s ago)  47s

```

It may take a minute or two for DNS to be ready before moving on by running the `kubectl get pods` command which allow us to display the resources and print a table of the most important information about the specified resources:

```

watch kubectl get pods --all-namespaces
CTRL+C to exit

```

The successful result will look like this, every container should be running.

NAMESPACE	NAME	STATUS	RESTARTS	AGE
Y				
kube-system	etcd-kube-01	Running	0	1/1
kube-system	kube-apiserver-kube-01	Running	0	1/1
kube-system	kube-controller-manager-kube-01	Running	0	1/1
kube-system	kube-dns-6f4fd0bf-whhd	Running	0	3/3
kube-system	kube-proxy-2hdhk	Running	0	1/1
kube-system	kube-proxy-tvhjk	Running	0	1/1
kube-system	kube-proxy-wspv	Running	0	1/1
kube-system	kube-scheduler-kube-01	Running	0	1/1
kube-system	weave-net-9ghns	Running	1	5s
kube-system	weave-net-1h8tq	Running	1	2/2

▶ 16:06

```

root@kube-01:~# kubectl apply -f https://cloud.weave.works/k8s/net/7k8s-version=$(kubectl version | base64 | tr -d '\n')
serviceaccount/weave-net created
clusterrolebinding:rbac.authorization.k8s.io/weave-net created
clusterrolebinding:rbac.authorization.k8s.io/weave-net created
role:rbac.authorization.k8s.io/weave-net created
rolebinding:rbac.authorization.k8s.io/weave-net created
daemonset.apps/weave-net created
root@kube-01:~# watch kubectl get pods --all-namespaces
root@kube-01:~# kubectl get nodes
NAME      STATUS  ROLES   AGE     VERSION
Kube-01   Ready   control-plane,master  6m6ls   v1.22.1
Kube-02   Ready   <none>  3m23s  v1.22.1
Kube-03   Ready   <none>  3m23s  v1.22.1
root@kube-01:~#

```

ready status

NAMESPACE	NAME	STATUS	RESTARTS	AGE	READ
Y					
kube-system	etcd-kube-01	Running	0	1/1	
kube-system	kube-apiserver-kube-01	Running	0	1/1	
kube-system	kube-controller-manager-kube-01	Running	0	1/1	
kube-system	kube-dns-6f4fd0bf-whhd	Running	0	3/3	
kube-system	kube-proxy-2hdhk	Running	0	1/1	
kube-system	kube-proxy-tvhjk	Running	0	1/1	
kube-system	kube-proxy-wspv	Running	0	1/1	
kube-system	kube-scheduler-kube-01	Running	0	1/1	
kube-system	weave-net-9ghns	Running	1	5s	
kube-system	weave-net-1h8tq	Running	0	2/2	
kube-system	weave-net-9hr2s	Running	0	2/2	

Check the nodes status again:

```

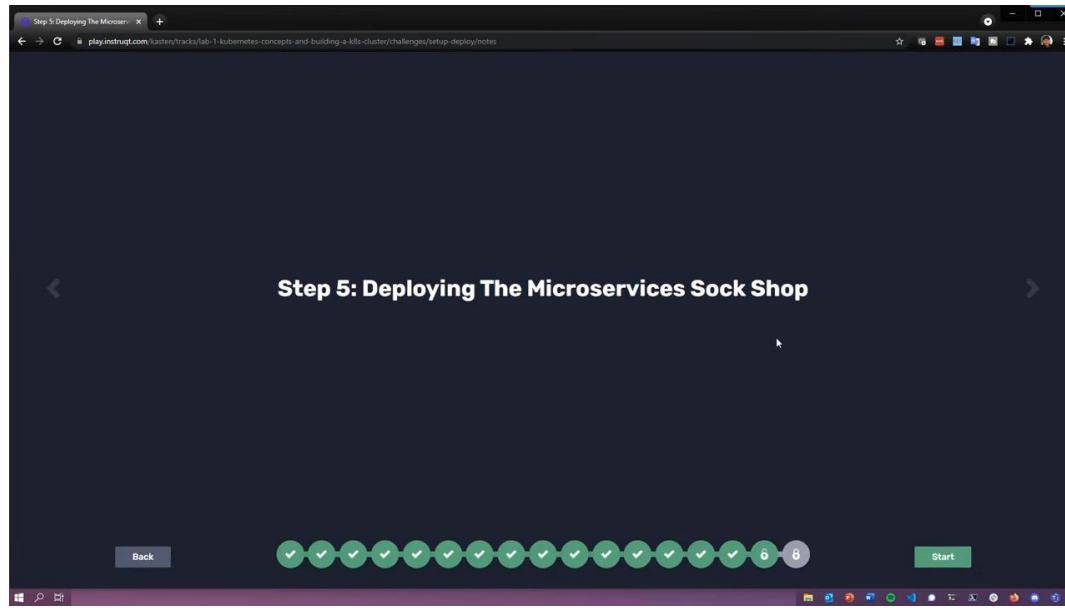
kubectl get nodes

```

NAME	STATUS	ROLES	AGE	VERSION
Kube-01	Ready	Control Plane	8m	v1.x.x
Kube-02	Ready	<none>	6s	v1.x.x
Kube-03	Ready	<none>	6s	v1.x.x

Congratulations, now your Kubernetes cluster running on Ubuntu 20.04 is up and ready for you to deploy a microservices application!

▶ 16:20



▶ 16:31

Step 5: Deploying The Microservices Sock Shop

Next we will deploy a demo microservices application to your Kubernetes cluster.

First, on Control Plane, clone the microservices sock shop git repo:

```
git clone https://github.com/microservices-demo/microservices-demo.git
```

Go to the microservices-demo/deploy/kubernetes folder:

```
cd microservices-demo/deploy/kubernetes
```

Next apply the demo to your Kubernetes cluster:

```
kubectl apply -f complete-demo.yaml
```

You will see the following result:

```
deployment "carts-db" created
service "carts-db" created
deployment "carts" created
service "carts" created
deployment "catalogue-db" created
service "catalogue-db" created
deployment "catalogue" created
service "catalogue" created
deployment "front-end" created
service "front-end" created
deployment "orders-db" created
service "orders-db" created
deployment "orders" created
service "orders" created
deployment "cart" created
deployment "payment" created
service "payment" created
deployment "queue-control-plane" created
service "queue-control-plane" created
```

Close Skip Check

▶ 17:33

Step 5: Deploying The Microservices

```

root@kube-01:~# git clone https://github.com/microservices-demo/microservices-demo.git
Cloning into 'microservices-demo'...
remote: Enumerating objects: 1008, done.
remote: Counting objects: 1008 (delta 470), done.
remote: Compressing objects: 1008 (delta 284), pack-reused 9721
remote: Total 1008 (delta 327), reused 377 (delta 284), pack-reused 9721
Resolving deltas: 100% (327/327), reused 52.95 MB | 22.38 MiB/s, done.
Resolving deltas: 100% (6194/6194), done.
root@kube-01:~# cd microservices-demo/deploy/kubernetes
root@kube-01:~/microservices-demo/deploy/kubernetes# kubectl apply -f complete-demo.yaml
Warning: spec.template.spec.nodeSelector[beta.kubernetes.io/os]: deprecated since v1.14; use "kubernetes.io/os" instead
deployment.apps/carts created
service/carts created
deployment.apps/orders created
deployment.apps/queue-db created
service/carts created
deployment.apps/payment created
service/payment created
deployment.apps/session-db created
service/session-db created
deployment.apps/shipping created
service/shipping created
deployment.apps/user created
service/user created
deployment.apps/queue-master created
service/queue-master created
deployment.apps/rabbitmq created
service/rabbitmq created
deployment.apps/session-db created
service/session-db created
deployment.apps/shipping created
service/shipping created
deployment.apps/user created
service/user created
deployment.apps/queue-shop created
Warning: spec.template.spec.nodeSelector[beta.kubernetes.io/os]: deprecated since v1.14; use "kubernetes.io/os" instead
deployment.apps/carts created
service/carts created
deployment.apps/orders created
deployment.apps/queue-db created
service/carts created
deployment/catalogue created
deployment.apps/catalogue-db created
service/catalogue-db created
deployment/catalogue-db created
deployment.apps/payment created
service/payment created
deployment.apps/session-db created
service/session-db created
deployment.apps/shipping created
service/shipping created
deployment.apps/user created
service/user created
root@kube-01:~/microservices-demo/deploy/kubernetes# kubectl apply -f complete-demo.yaml

```

18:03

Step 5: Deploying The Microservices

```

root@kube-01:~# git clone https://github.com/microservices-demo/microservices-demo.git
Cloning into 'microservices-demo'...
remote: Counting objects: 1008 (476/476), done.
remote: Compressing objects: 1008 (181/181), done.
remote: Total 1008 (delta 470), reused 9721
Resolving deltas: 100% (6194/6194), done.
root@kube-01:~# cd microservices-demo/deploy/kubernetes
root@kube-01:~/microservices-demo/deploy/kubernetes# kubectl get namespaces
Warning: kubectl get resources is deprecated, use kubectl get --resource instead
Dockerfile README.md complete-demo.yaml manifests manifests-jaeger manifests-logging manifests-policy
Makefile autoscaling helm-chart manifests-alerting manifests-loadtest manifests-monitoring terraform
root@kube-01:~/microservices-demo/deploy/kubernetes# kubectl apply -f complete-demo.yaml
namespace/sock-shop created
Warning: spec.template.spec.nodeSelector[beta.kubernetes.io/os]: deprecated since v1.14; use "kubernetes.io/os" instead
deployment.apps/carts created
service/carts created
deployment.apps/orders created
deployment.apps/queue-db created
service/carts created
deployment/catalogue created
deployment.apps/catalogue-db created
service/catalogue-db created
deployment/catalogue-db created
deployment.apps/payment created
service/payment created
deployment.apps/session-db created
service/session-db created
deployment.apps/shipping created
service/shipping created
deployment.apps/user created
service/user created
root@kube-01:~/microservices-demo/deploy/kubernetes# kubectl get namespaces
NAME          STATUS   AGE
default        Active   8m49s
kube-node-lease Active   8m51s
kube-public    Active   8m51s
kube-system    Active   8m51s
sock-shop      Active   23s
root@kube-01:~/microservices-demo/deploy/kubernetes# kubectl get pods --namespace=sock-shop
NAME                STATUS  AGE
carts-74f458cb8-h9924  Running  1m
carts-8fbccdfbc9-vy4fw  Running  0s
catalogue-676d6b9f7c-55nqg  Running  0s
catalogue-85c67cd8cd-hvk96  Running  0s
front-end-877bfdb8-hqpx9  Running  0s
orders-787bf3fb9f-xfd16  Running  0s
orders-db-77565b675-gv456  Running  0s
orders-db-79340f1-4zzqs  Running  0s
payment-75840f1-4zzqs  Running  0s
queue-control-plane-5c86964795-18sjg  1/1  Running  0  1m
rabbitmq-96d887875-1f46w  1/1
root@kube-01:~/microservices-demo/deploy/kubernetes# watch kubectl get pods --namespace=sock-shop

```

18:43

The screenshot shows a Windows desktop with a browser window open. The address bar indicates the URL is [playinstruct.com/kasten/tracks/lab-1-kubernetes-concepts-and-building-a-k8s-cluster/challenges/setup-deploy/assignment](https://playinstruct.com/kasten/tracks/lab-1-kubernetes-concepts-and-building-a-k8s-cluster/challenges/setup-deploy/assignment). The browser has three tabs: 'Control Plane', 'Index-02', and 'Index-03'. The 'Index-03' tab is active and displays the 'Sock Shop' application from 'weaveworks socks'. A red box highlights the 'Sock Shop' tab in the browser's tab bar. A red arrow points from the text 'you can check it out' to the highlighted tab. To the right of the browser, a separate window shows the Kubernetes cluster status with several pods running, including 'orders-8b-37565bb75-gv456', 'payment-75f75d467f-4zqzs', 'queue-control-plane-5c869d795-t8sjg', 'rabbitmq-96d887975-1f46w', 'shipping-5bd69fbfc-c-vpmcp', 'user-5bd99b946b-4rcs8', and 'user-db-5fd89bbb-b-r0pd'. Below the browser, a red button displays the time '20:26'.

The screenshot shows a dark-themed web browser window. The address bar is visible at the top. The main content area displays the text 'Please wait while we setup the challenge'. At the bottom, there is a navigation bar with a 'Back' button on the left, a series of green circular progress indicators with checkmarks, and a 'Setting up challenge' button on the right. The number '6' is also present near the progress indicators. A red button at the bottom left displays the time '21:09'.