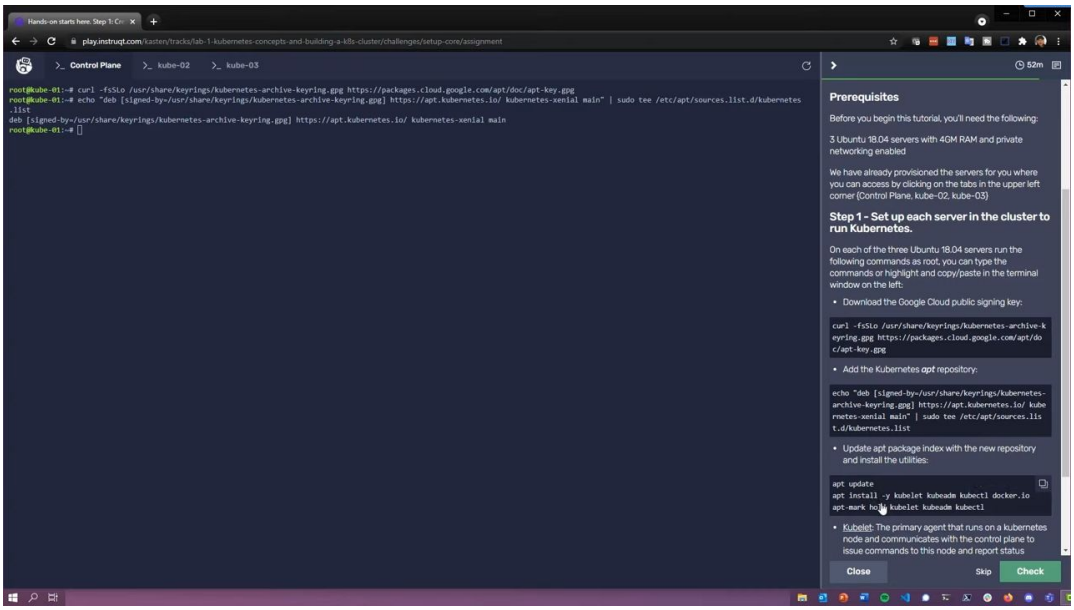
View, add and edit your notes in the app

# Lab 1 Video Overview Building Your First Cluster

Generated on December 3, 2023

## Summary

| Notes | Screenshots | Bookmarks |
| --- | --- | --- |
| 7 | 29 | 0 |

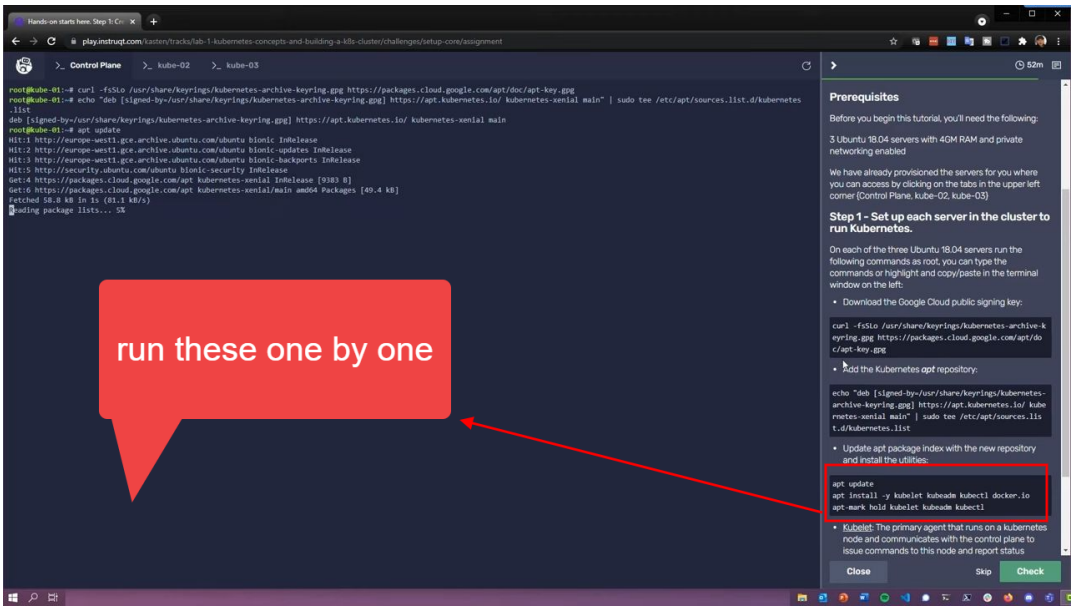



4:58

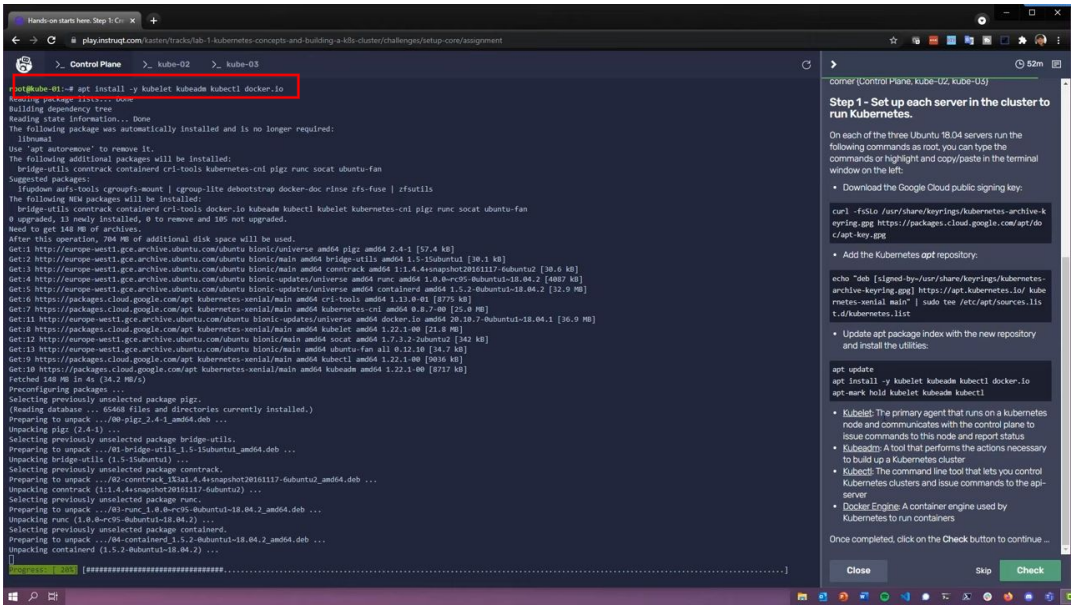
copied 1st and 2nd

and paste

4:58




5:11



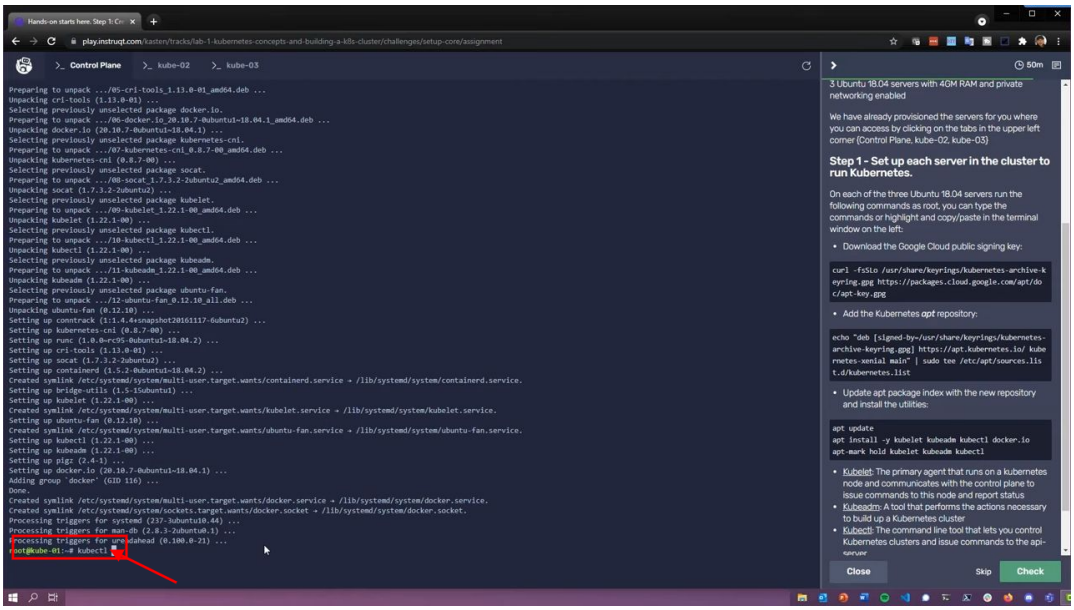

5:20

cube ctl : access into our cluster

5:28

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

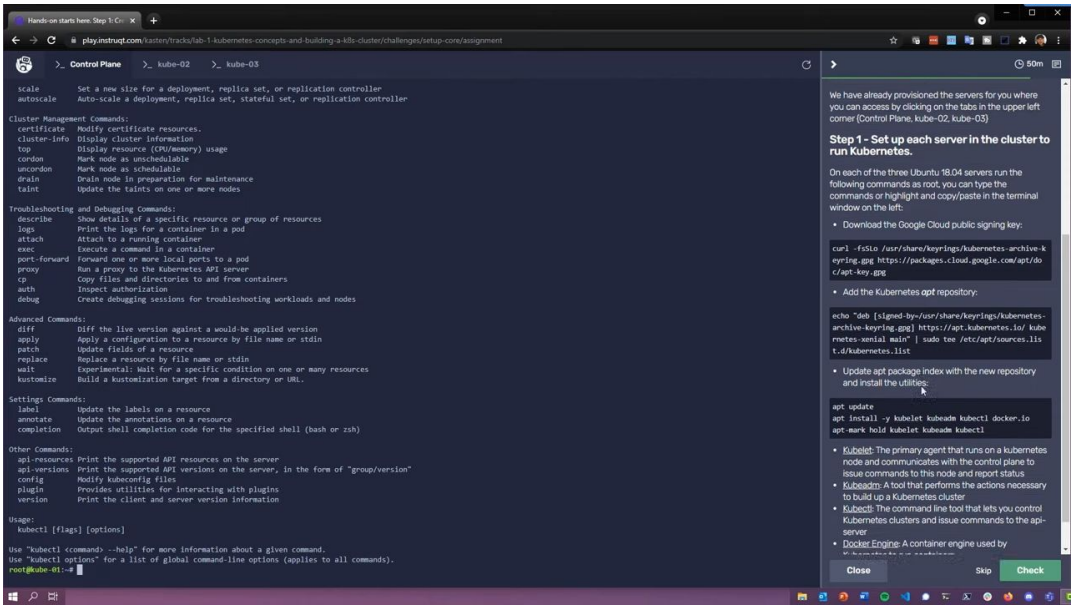
5:55



6:40

successfully installed

6:46

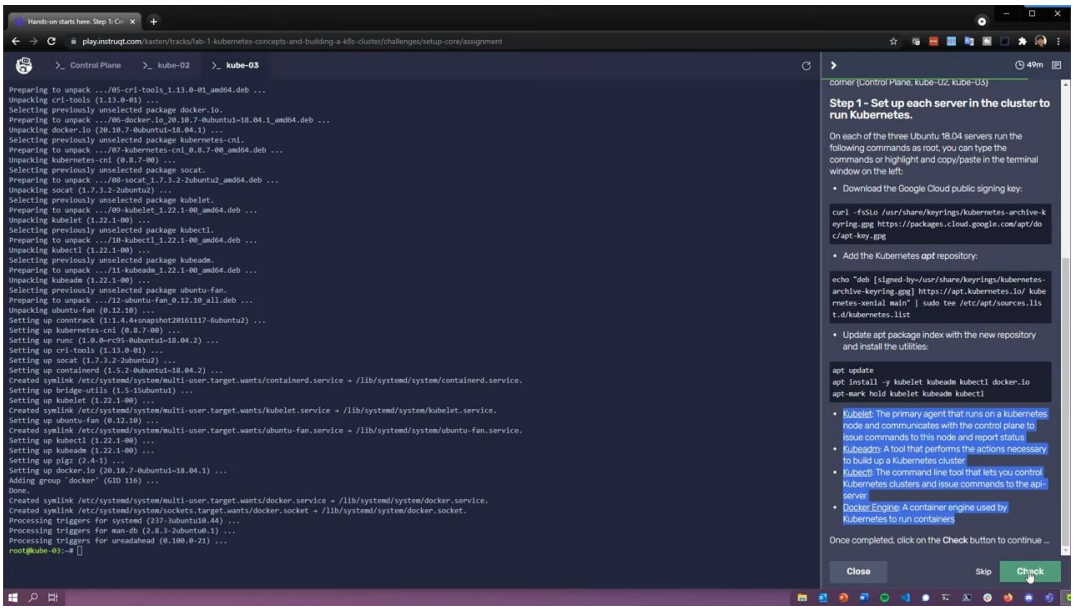


6:46

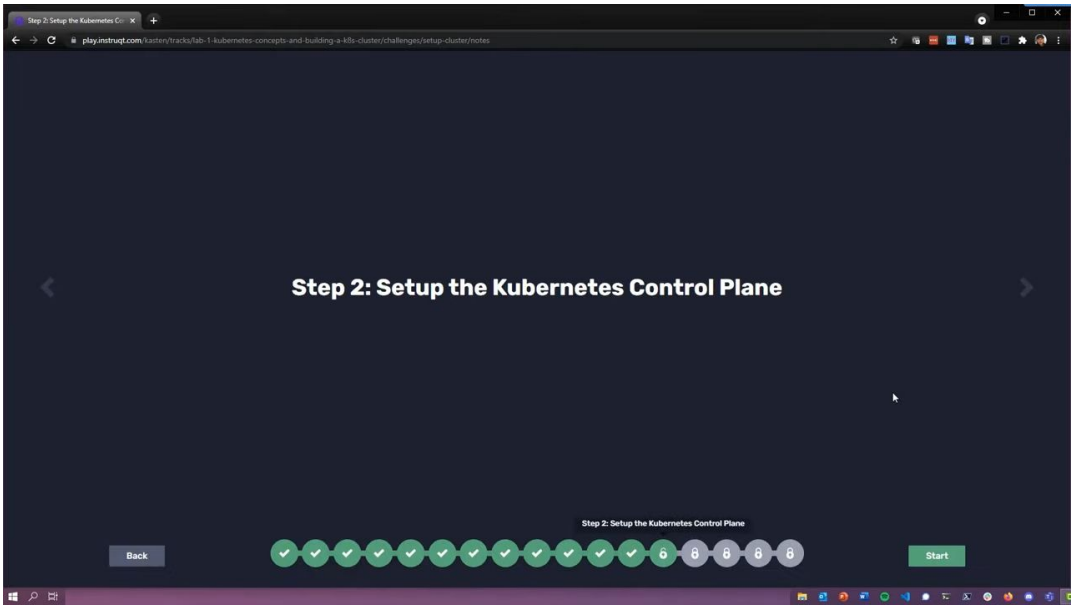
when the install is completed on all three

click on check button and continue to build the cluster

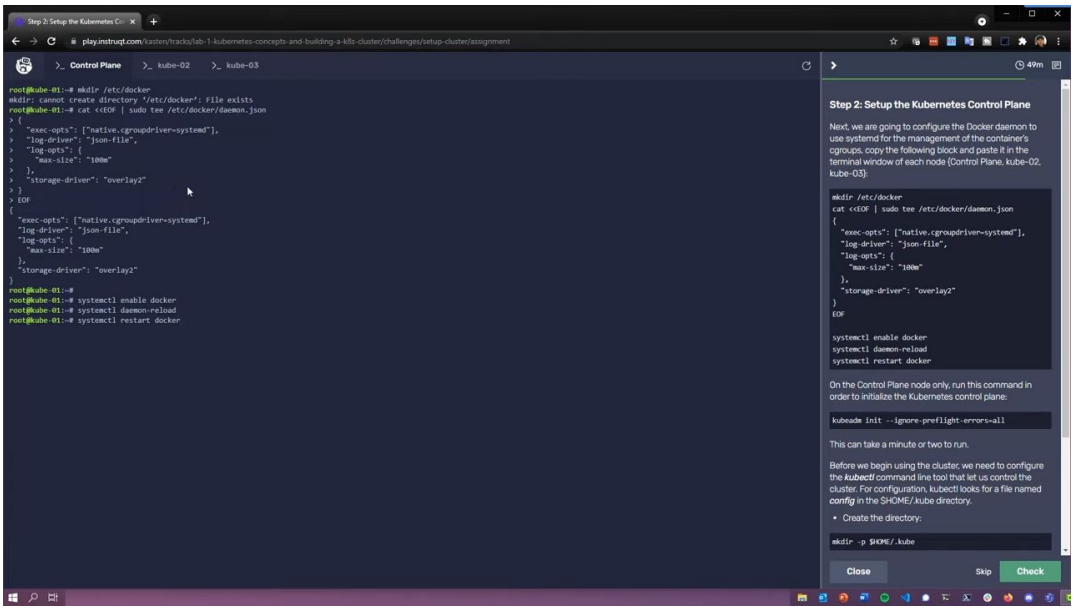
7:28


play.instruqt.com/kasten/tracks/lab-1-kubernetes-concepts-and-building-a-k8s-cluster/challenges/setup-core/assignment
Control Plane
kube-02
kube-03
Preparing to unpack .../05-cri-tools_1.13.0-01_amd64.deb ...
Unpacking cri-tools (1.13.0-01) ...
Selecting previously unselected package docker.io.
Preparing to unpack .../06-docker.io_20.10.7-0ubuntu1~18.04.1_amd64.deb ...
Unpacking docker.io (20.10.7-0ubuntu1~18.04.1) ...
Selecting previously unselected package kubernetes-cni.
Preparing to unpack .../07-kubernetes-cni_0.8.7-00_amd64.deb ...
Unpacking kubernetes-cni (0.8.7-00) ...
Selecting previously unselected package socat.
Preparing to unpack .../08-socat_1.7.3.2-2ubuntu2_amd64.deb ...
Unpacking socat (1.7.3.2-2ubuntu2) ...
Selecting previously unselected package kubelet.
Preparing to unpack .../09-kubelet_1.22.1-00_amd64.deb ...
Unpacking kubelet (1.22.1-00) ...
Selecting previously unselected package kubectl.
Preparing to unpack .../10-kubectl_1.22.1-00_amd64.deb ...
Unpacking kubectl (1.22.1-00) ...
Selecting previously unselected package kubeadm.
Preparing to unpack .../11-kubeadm_1.22.1-00_amd64.deb ...
Unpacking kubeadm (1.22.1-00) ...
Selecting previously unselected package ubuntu-fan.
Preparing to unpack .../12-ubuntu-fan_0.12.10_all.deb ...
Unpacking ubuntu-fan (0.12.10) ...
Setting up conntrack (1:1.4.4+snapshot20161117-6ubuntu2) ...
Setting up kubernetes-cni (0.8.7-00) ...
Setting up runc (1.0.0~rc95-0ubuntu1~18.04.2) ...
Setting up cri-tools (1.13.0-01) ...
Setting up socat (1.7.3.2-2ubuntu2) ...
Setting up containerd (1.5.2-0ubuntu1~18.04.2) ...
Created symlink /etc/systemd/system/multi-user.target.wants/containerd.service → /lib/systemd/system/containerd.service.
Setting up bridge-utils (1.5-15ubuntu1) ...
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 ubuntu-fan (0.12.10) ...
Created symlink /etc/systemd/system/multi-user.target.wants/ubuntu-fan.service → /lib/systemd/system/ubuntu-fan.service.
Setting up kubectl (1.22.1-00) ...
Setting up kubeadm (1.22.1-00) ...
Setting up pigz (2.4-1) ...
Setting up docker.io (20.10.7-0ubuntu1~18.04.1) ...
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 (237-3ubuntu10.44) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Processing triggers for ureadahead (0.100.0-21) ...
root@kube-03:~#
49m
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 in the terminal window on the left:
Download the Google Cloud public signing key:
curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg https://packages.cloud.google.com/apt/doc/apt-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 kubelet kubeadm kubectl docker.io
apt-mark hold kubelet kubeadm kubectl
Kubelet: The primary agent that runs on a kubernetes node and communicates with the control plane to issue commands to this node and report status
Kubeadm: A tool that performs the actions necessary to build up a Kubernetes cluster
Kubectl: The command line tool that lets you control Kubernetes clusters and issue commands to the api-server
Docker Engine: A container engine used by Kubernetes to run containers
Once completed, click on the Check button to continue ...
Close
Skip
Check

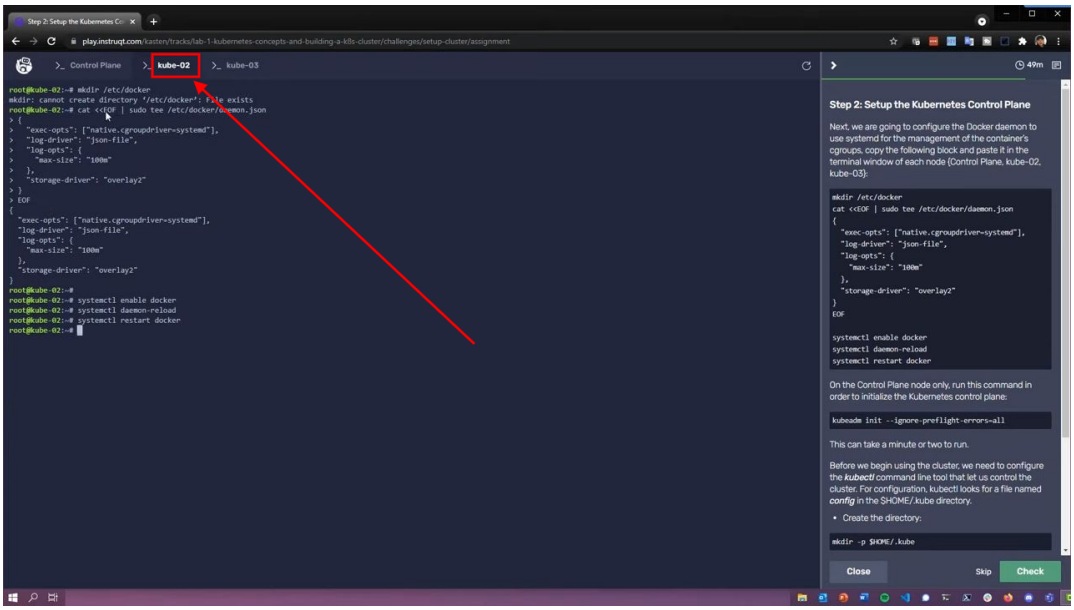
7:28


play.instruqt.com/kasten/tracks/lab-1-kubernetes-concepts-and-building-a-k8s-cluster/challenges/setup-cluster/notes
Step 2: Setup the Kubernetes Control Plane
Step 2: Setup the Kubernetes Control Plane
Back
Start

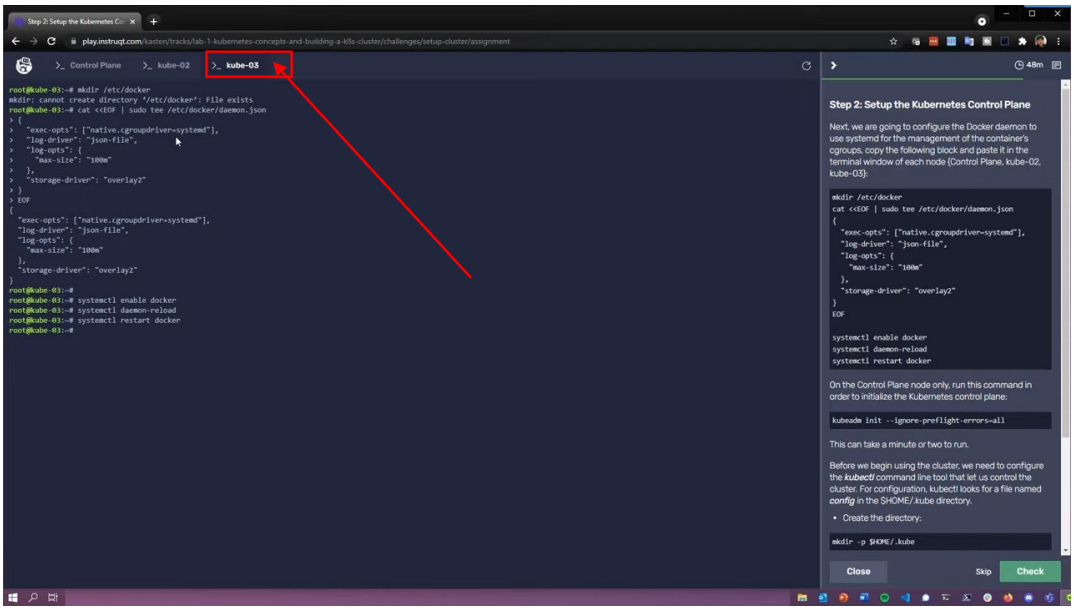
7:42


Step 2: Setup the Kubernetes Co
play.instruqt.com/kasten/tracks/lab-1-kubernetes-concepts-and-building-a-k8s-cluster/challenges/setup-cluster/assignment
Control Plane
kube-02
kube-03
49m
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:~#
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 kubectl command line tool that let us control the cluster. For configuration, kubectl looks for a file named config in the $HOME/.kube directory.
Create the directory:
mkdir -p $HOME/.kube
Close
Skip
Check

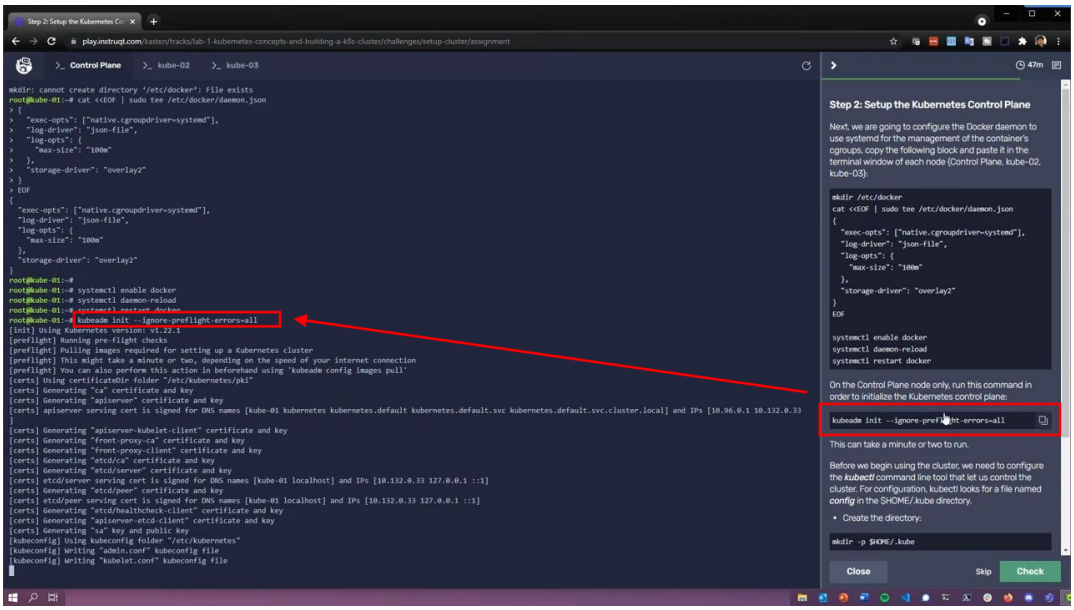
8:14


Step 2: Setup the Kubernetes Co
play.instruqt.com/kasten/tracks/lab-1-kubernetes-concepts-and-building-a-k8s-cluster/challenges/setup-cluster/assignment
Control Plane
kube-02
kube-03
49m
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:~#
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 kubectl command line tool that let us control the cluster. For configuration, kubectl looks for a file named config in the $HOME/.kube directory.
Create the directory:
mkdir -p $HOME/.kube
Close
Skip
Check

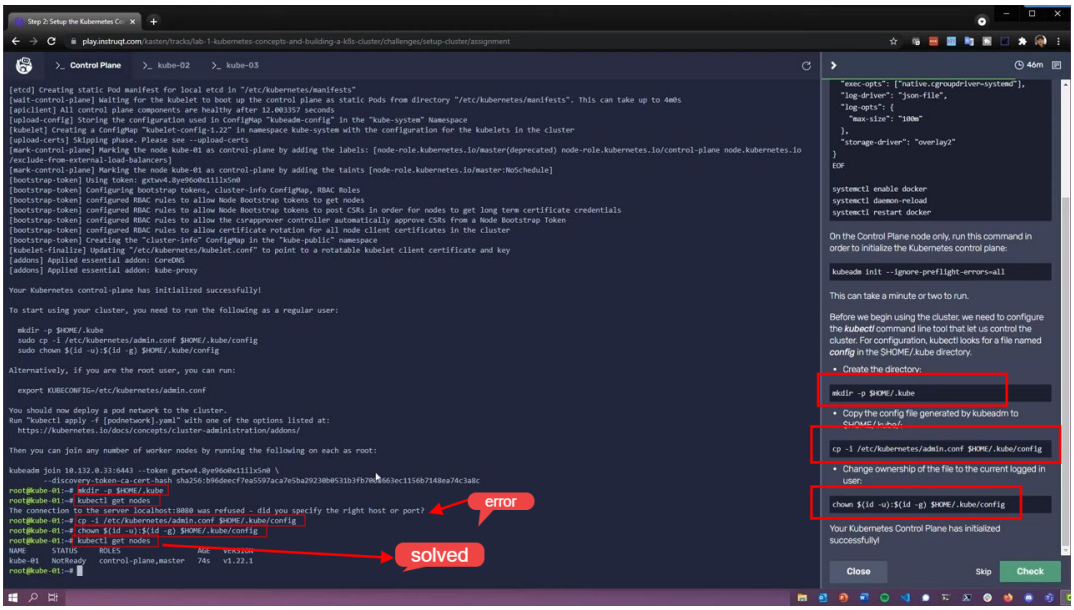
8:23


play.instruqt.com/kasten/tracks/lab-1-kubernetes-concepts-and-building-a-k8s-cluster/challenges/setup-cluster/assignment
Control Plane
kube-02
kube-03
48m
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 kubectl command line tool that let us control the cluster. For configuration, kubectl looks for a file named config in the $HOME/.kube directory.
Create the directory:
mkdir -p $HOME/.kube
Close
Skip
Check

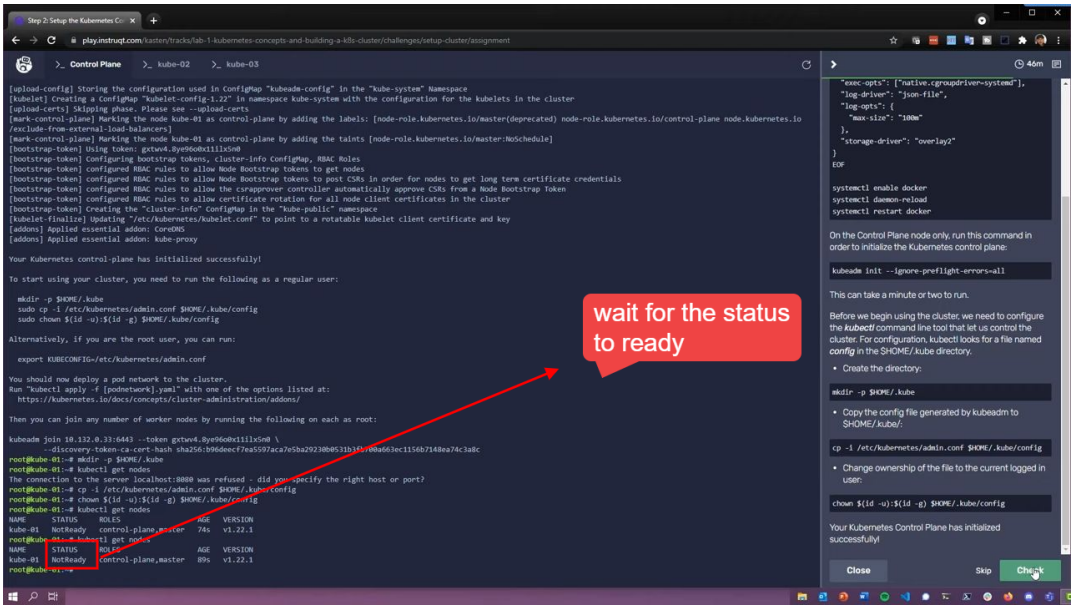
8:48


play.instruqt.com/kasten/tracks/lab-1-kubernetes-concepts-and-building-a-k8s-cluster/challenges/setup-cluster/assignment
Control Plane
kube-02
kube-03
47m
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):
On the Control Plane node only, run this command in order to initialize the Kubernetes control plane:
This can take a minute or two to run.
Before we begin using the cluster, we need to configure the kubectl command line tool that let us control the cluster. For configuration, kubectl looks for a file named config in the $HOME/.kube directory.
Create the directory:
mkdir -p $HOME/.kube
Close
Skip
Check

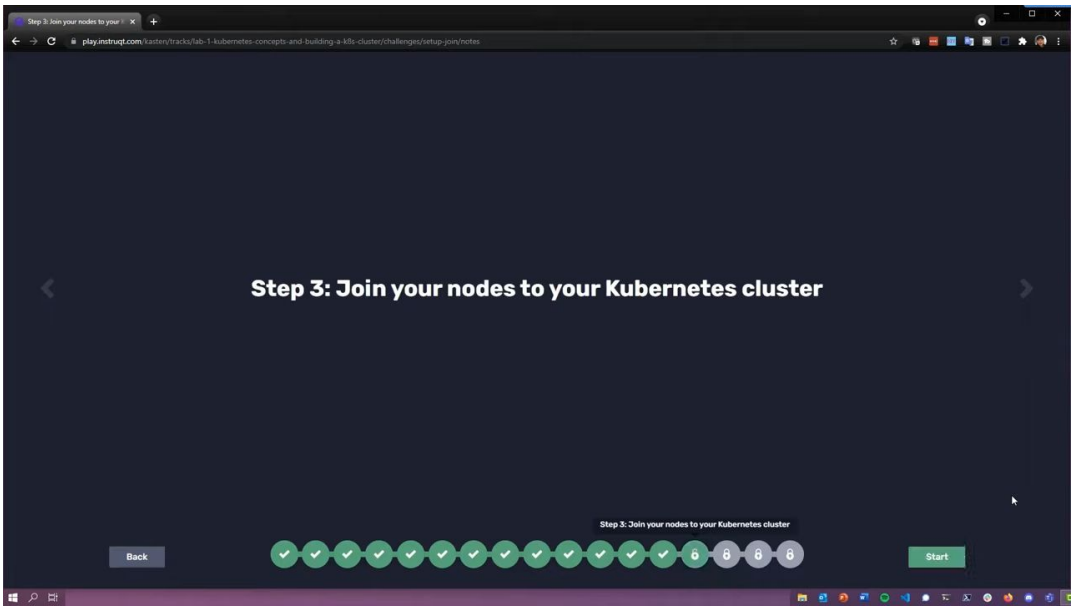
9:30



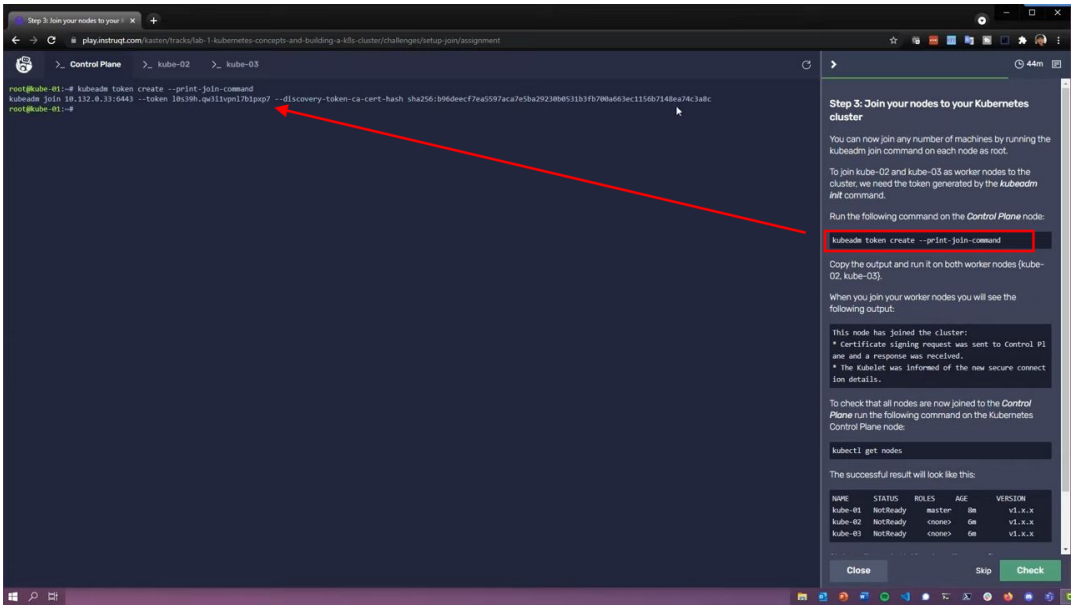

10:56



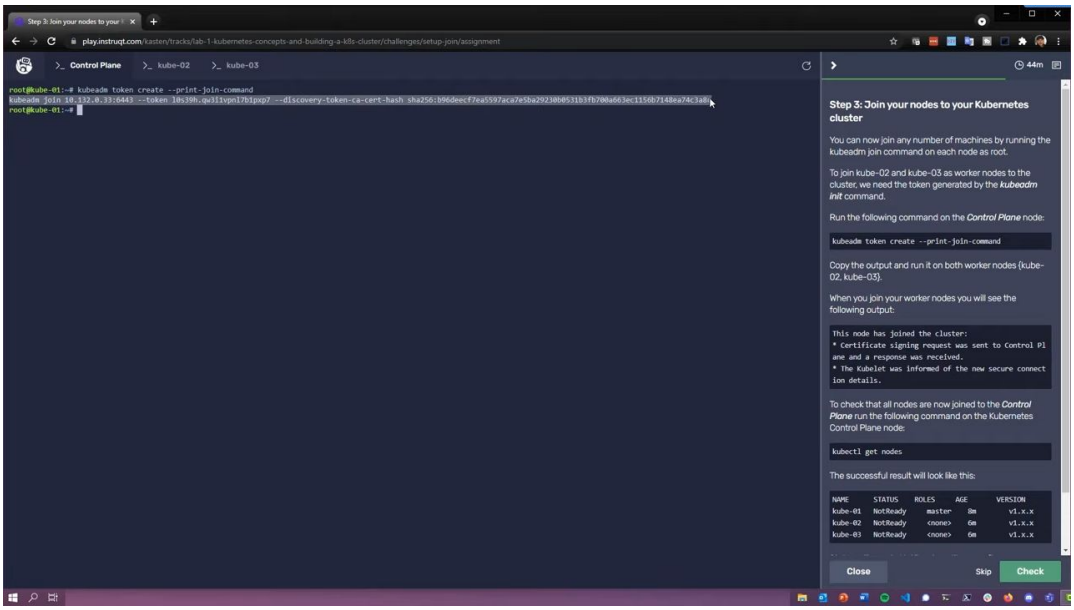

11:18


Step 3: Join your nodes to your Kubernetes cluster
Step 3: Join your nodes to your Kubernetes cluster
Back
Start

11:30


Control Plane
kube-02
kube-03
root@kube-01:~# kubeadm token create --print-join-command
kubeadm join 10.132.0.33:6443 --token 10s30h.qw311vpnl7b1pxp7 --discovery-token-ca-cert-hash sha256:b96deecf7ea5597aca7e5ba29230b0531b3fb700a663ec1156b7148ea74c3a8c
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



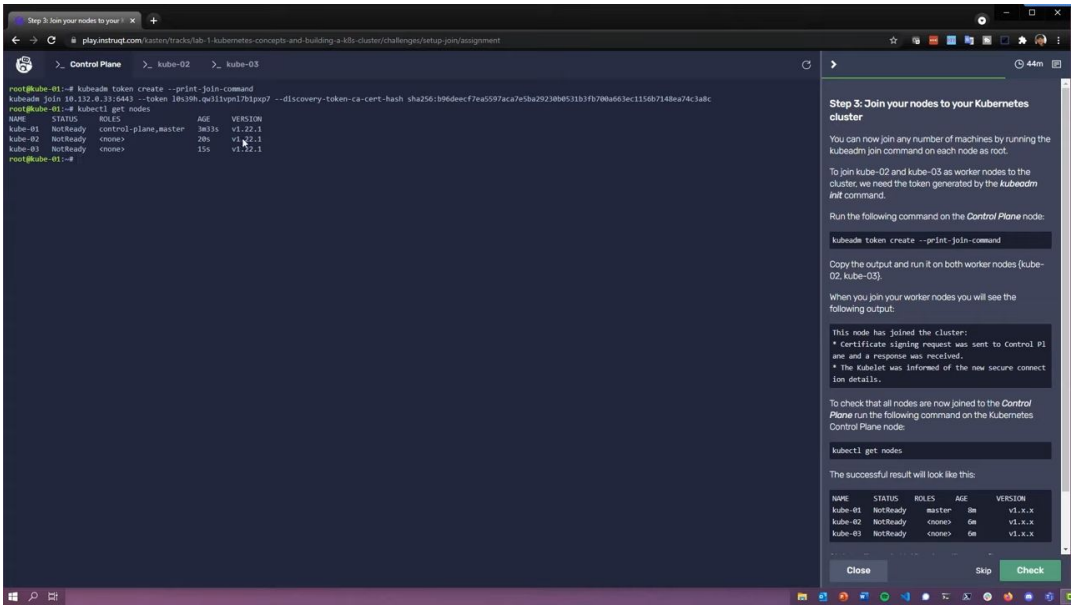

12:47

copy the output that is generated from the

above command

and paste in Kube-02 and Kube-03

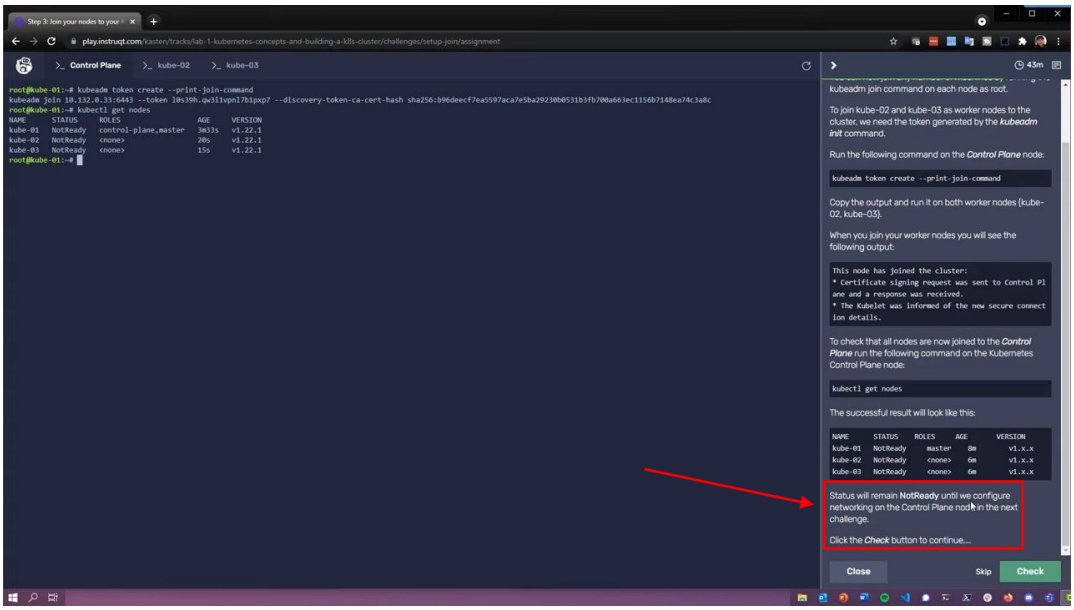
12:47



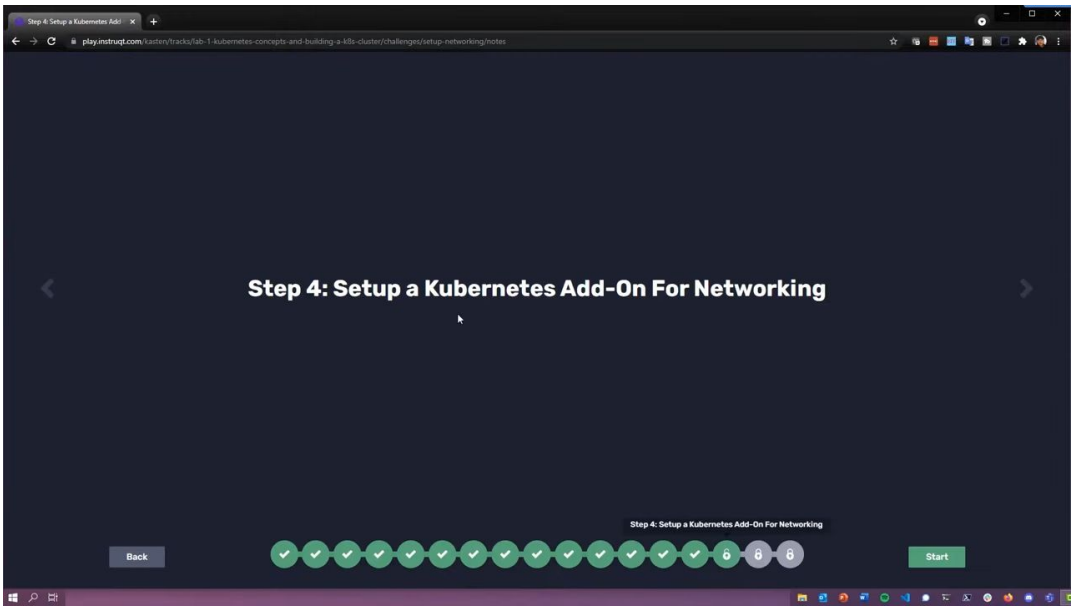

13:18

you will see two other working nodes

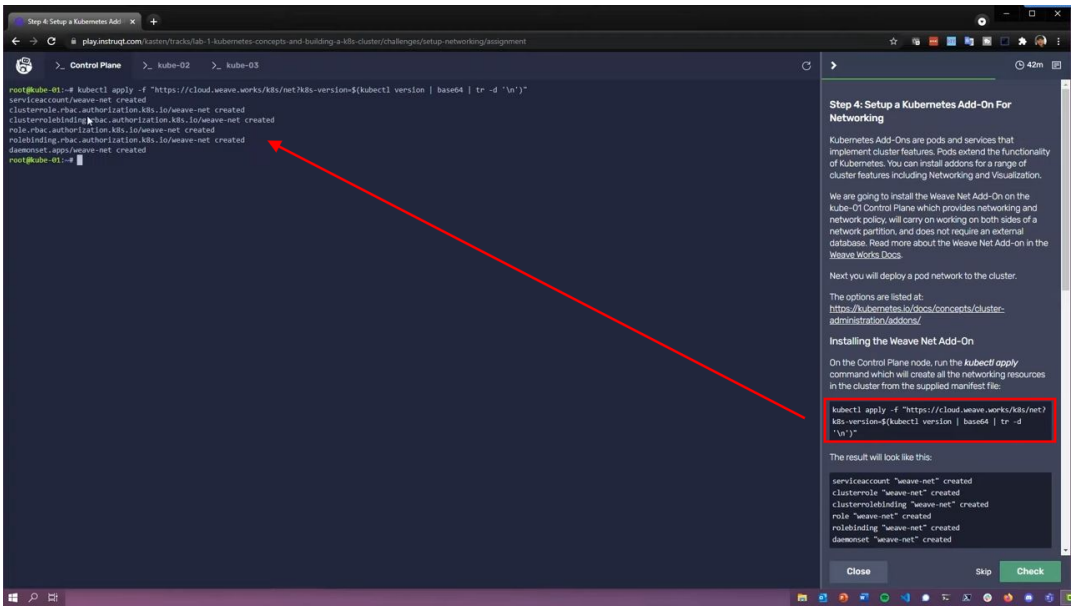
13:19



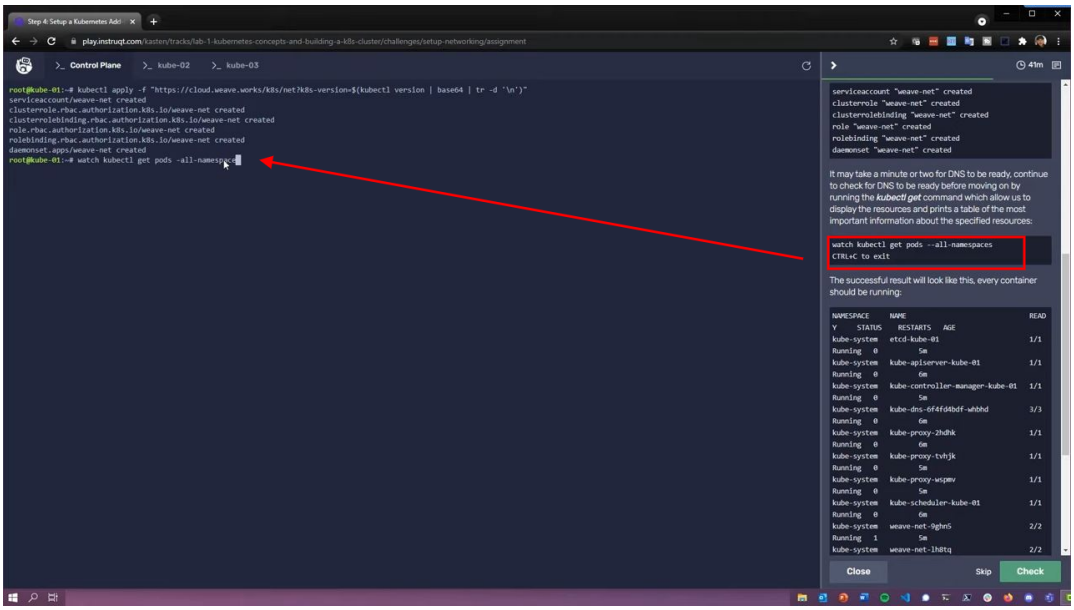

13:31



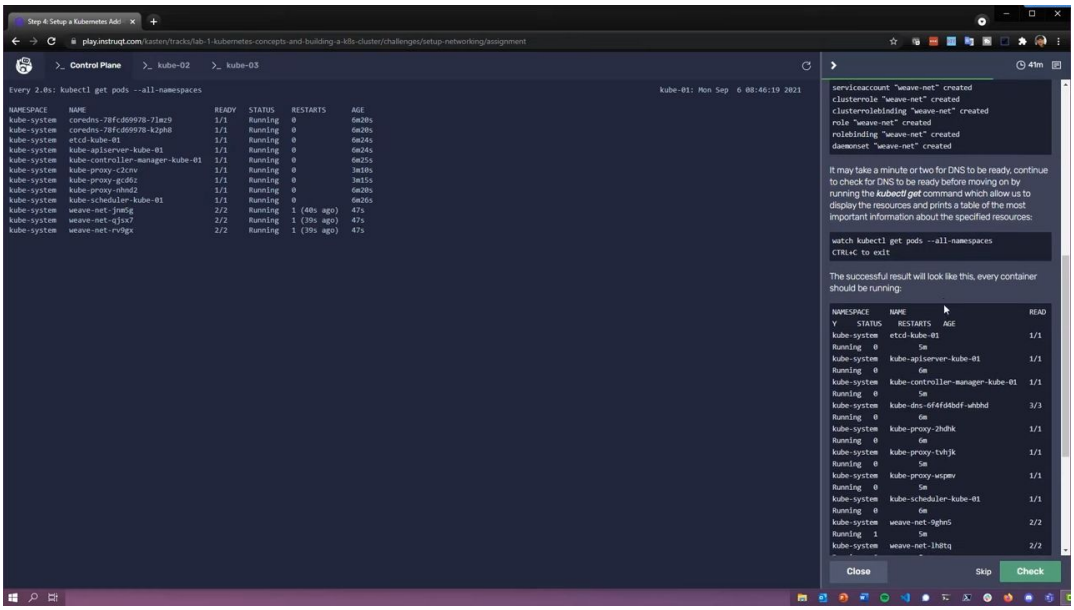

13:54


Step 4: Setup a Kubernetes Add
play.instruqt.com/kasten/tracks/lab-1-kubernetes-concepts-and-building-a-k8s-cluster/challenges/setup-networking/assignment
Control Plane
kube-02
kube-03
root@kube-01:~# kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')"
serviceaccount/weave-net created
clusterrole.rbac.authorization.k8s.io/weave-net created
clusterrolebinding.rbac.authorization.k8s.io/weave-net created
role.rbac.authorization.k8s.io/weave-net created
rolebinding.rbac.authorization.k8s.io/weave-net created
daemonset.apps/weave-net created
root@kube-01:~#
42m
Step 4: Setup a Kubernetes Add-On For Networking
Kubernetes Add-Ons are pods and services that implement cluster features. Pods extend the functionality of Kubernetes. You can install addons for a range of cluster features including Networking and Visualization.
We are going to install the Weave Net Add-On on the kube-01 Control Plane which provides networking and network policy, will carry on working on both sides of a network partition, and does not require an external database. Read more about the Weave Net Add-on in the Weave Works Docs.
Next you will deploy a pod network to the cluster.
The options are listed at: https://kubernetes.io/docs/concepts/cluster-administration/addons/
Installing the Weave Net Add-On
On the Control Plane node, run the kubectl apply command which will create all the networking resources in the cluster from the supplied manifest file:
kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(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

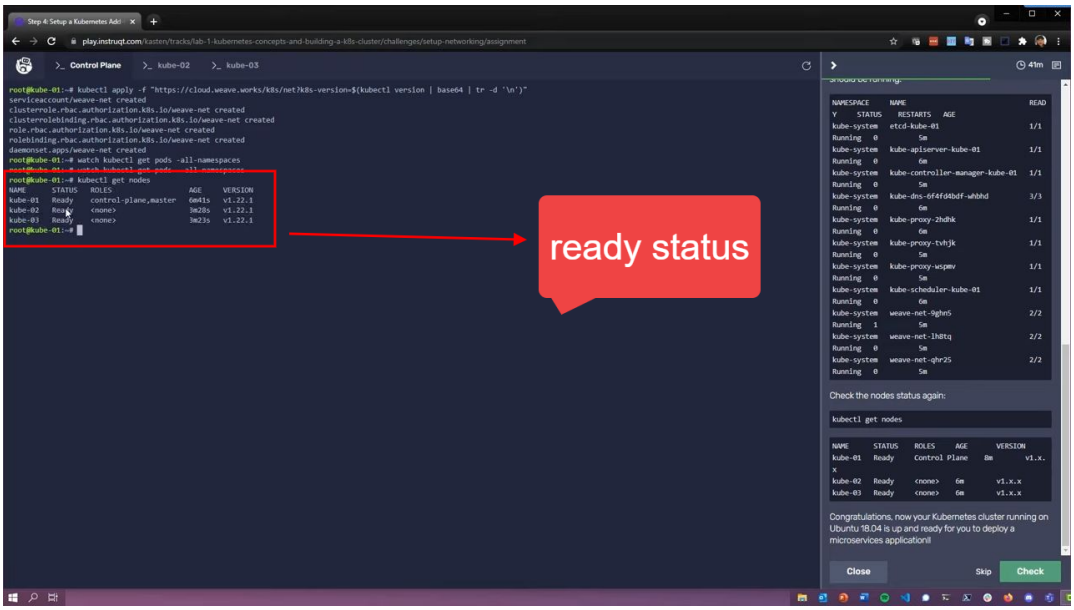
15:20


Step 4: Setup a Kubernetes Add
play.instruqt.com/kasten/tracks/lab-1-kubernetes-concepts-and-building-a-k8s-cluster/challenges/setup-networking/assignment
Control Plane
kube-02
kube-03
root@kube-01:~# kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')"
serviceaccount/weave-net created
clusterrole.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
41m
serviceaccount "weave-net" created
clusterrole "weave-net" created
clusterrolebinding "weave-net" created
role "weave-net" created
rolebinding "weave-net" created
daemonset "weave-net" created
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 READ
Y STATUS RESTARTS AGE
kube-system etcd-kube-01 1/1
Running 0 5m
kube-system kube-apiserver-kube-01 1/1
Running 0 6m
kube-system kube-controller-manager-kube-01 1/1
Running 0 5m
kube-system kube-dns-6f4fd4bdf-whbhd 3/3
Running 0 6m
kube-system kube-proxy-2hdhk 1/1
Running 0 6m
kube-system kube-proxy-tvhjk 1/1
Running 0 5m
kube-system kube-proxy-wspmv 1/1
Running 0 5m
kube-system kube-scheduler-kube-01 1/1
Running 0 6m
kube-system weave-net-9ghn5 2/2
Running 1 5m
kube-system weave-net-lh8tq 2/2
Close
Skip
Check

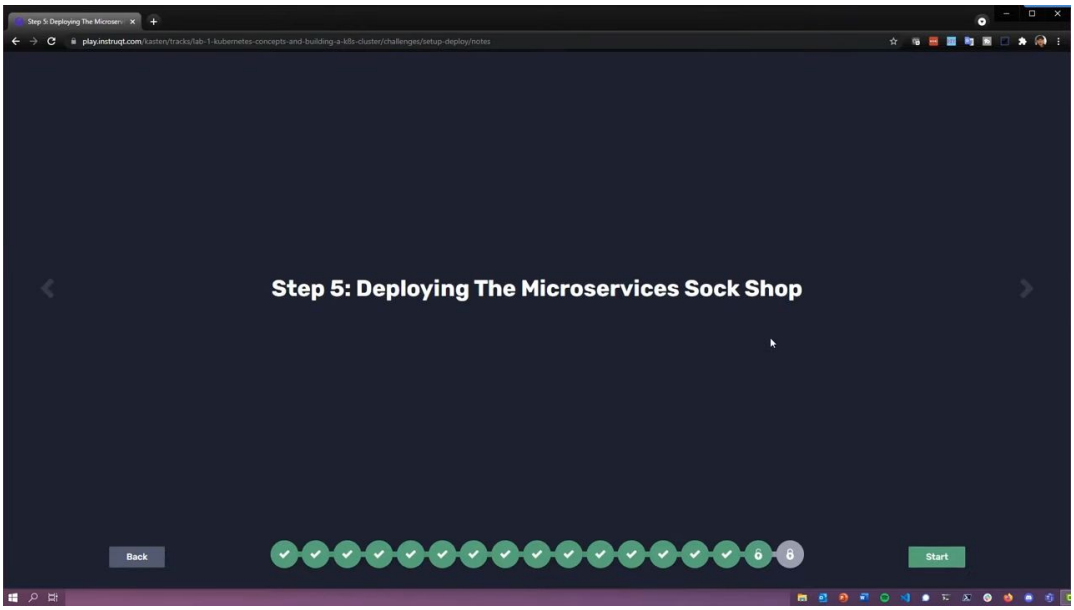
15:51


Step 4: Setup a Kubernetes Add
play.instruqt.com/kasten/tracks/lab-1-kubernetes-concepts-and-building-a-k8s-cluster/challenges/setup-networking/assignment
Control Plane
kube-02
kube-03
41m
Every 2.0s: kubectl get pods --all-namespaces
kube-01: Mon Sep 6 08:46:19 2021
NAMESPACE NAME READY STATUS RESTARTS AGE
kube-system coredns-78fcd69978-7lmz9 1/1 Running 0 6m20s
kube-system coredns-78fcd69978-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-c2cnv 1/1 Running 0 3m10s
kube-system kube-proxy-gcd6z 1/1 Running 0 3m15s
kube-system kube-proxy-nhnd2 1/1 Running 0 6m20s
kube-system kube-scheduler-kube-01 1/1 Running 0 6m26s
kube-system weave-net-jnm5g 2/2 Running 1 (40s ago) 47s
kube-system weave-net-qjsx7 2/2 Running 1 (39s ago) 47s
kube-system weave-net-rv9gx 2/2 Running 1 (39s ago) 47s
serviceaccount "weave-net" created
clusterrole "weave-net" created
clusterrolebinding "weave-net" created
role "weave-net" created
rolebinding "weave-net" created
daemonset "weave-net" created
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 READ
Y STATUS RESTARTS AGE
kube-system etcd-kube-01 1/1
Running 0 5m
kube-system kube-apiserver-kube-01 1/1
Running 0 6m
kube-system kube-controller-manager-kube-01 1/1
Running 0 5m
kube-system kube-dns-6f4fd4bdf-whbhd 3/3
Running 0 6m
kube-system kube-proxy-2hdhk 1/1
Running 0 6m
kube-system kube-proxy-tvhjk 1/1
Running 0 5m
kube-system kube-proxy-wspmv 1/1
Running 0 5m
kube-system kube-scheduler-kube-01 1/1
Running 0 6m
kube-system weave-net-9ghn5 2/2
Running 1 5m
kube-system weave-net-1h8tq 2/2
Close
Skip
Check

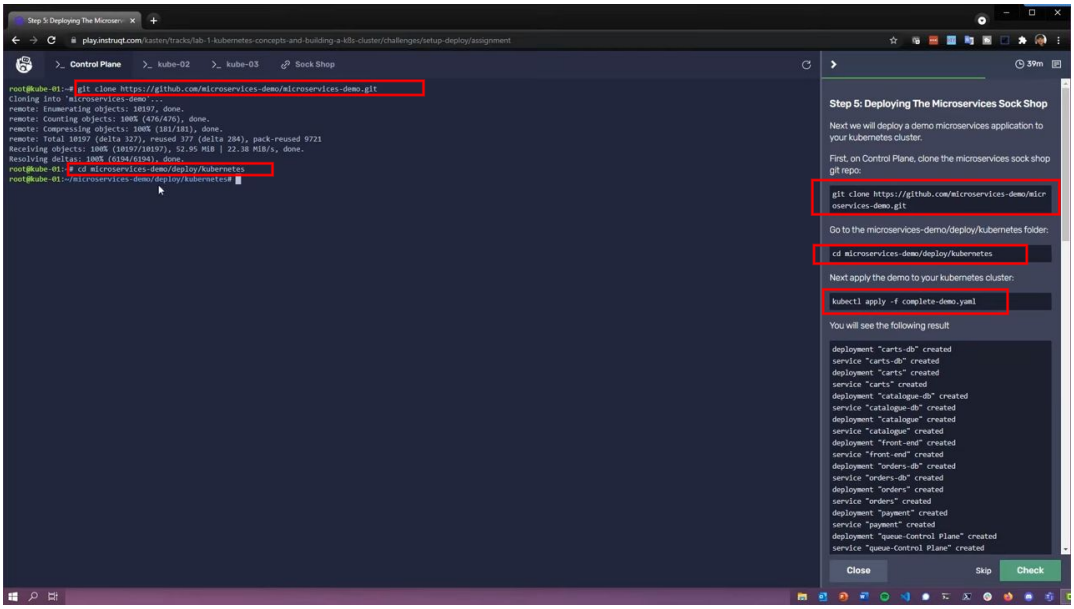
16:06


Step 4: Setup a Kubernetes Add
play.instruqt.com/kasten/tracks/lab-1-kubernetes-concepts-and-building-a-k8s-cluster/challenges/setup-networking/assignment
Control Plane
kube-02
kube-03
41m
root@kube-01:~# kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')"
serviceaccount/weave-net created
clusterrole.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 6m41s v1.22.1
kube-02 Ready <none> 3m28s v1.22.1
kube-03 Ready <none> 3m23s v1.22.1
root@kube-01:~#
ready status
NAMESPACE NAME READ
Y STATUS RESTARTS AGE
kube-system etcd-kube-01 1/1
Running 0 5m
kube-system kube-apiserver-kube-01 1/1
Running 0 6m
kube-system kube-controller-manager-kube-01 1/1
Running 0 5m
kube-system kube-dns-6f4fd4bdf-whbhd 3/3
Running 0 6m
kube-system kube-proxy-2hdhk 1/1
Running 0 6m
kube-system kube-proxy-tvhjk 1/1
Running 0 5m
kube-system kube-proxy-wspmv 1/1
Running 0 5m
kube-system kube-scheduler-kube-01 1/1
Running 0 6m
kube-system weave-net-9ghn5 2/2
Running 1 5m
kube-system weave-net-1h8tq 2/2
Running 0 5m
kube-system weave-net-qhr25 2/2
Running 0 5m
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> 6m v1.x.x
kube-03 Ready <none> 6m v1.x.x
Congratulations, now your Kubernetes cluster running on Ubuntu 18.04 is up and ready for you to deploy a microservices application!
Close
Skip
Check

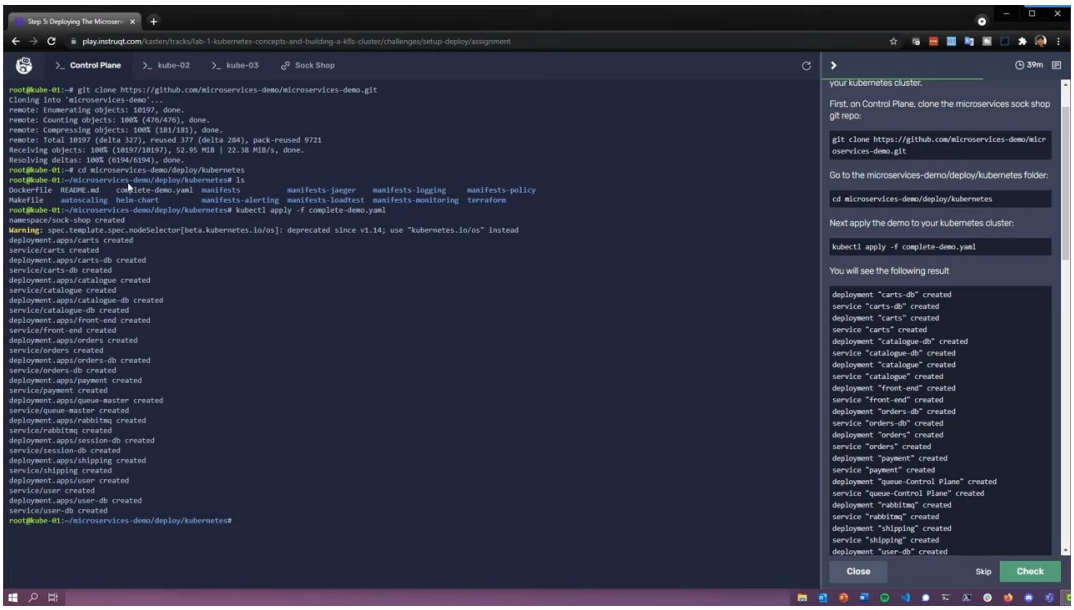
16:20


Step 5: Deploying The Microservices Sock Shop
Back
Start

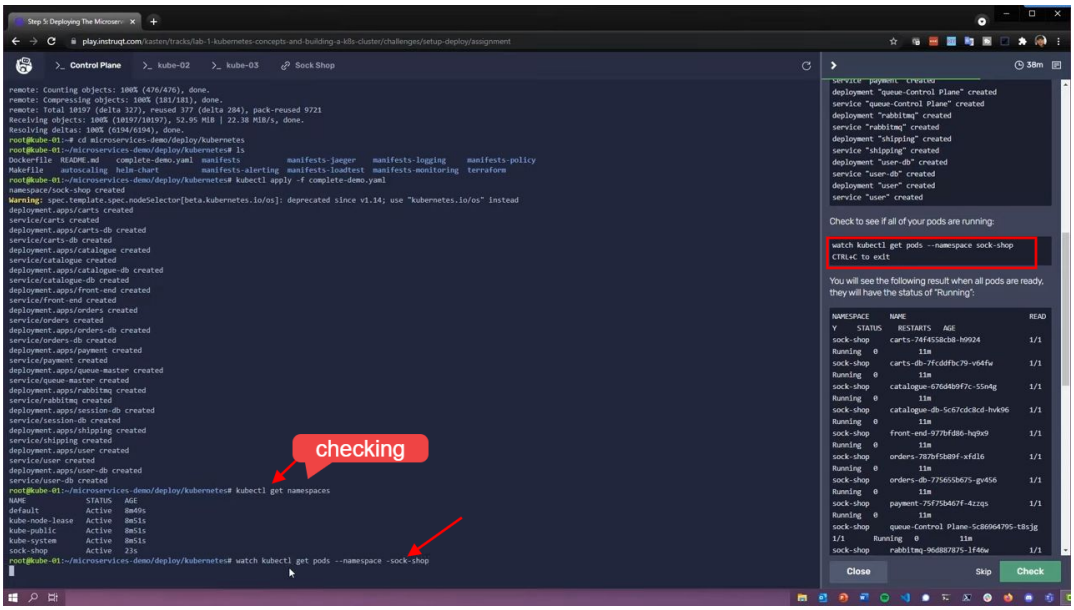
16:31


Control Plane
kube-02
kube-03
Sock Shop
root@kube-01:~# git clone https://github.com/microservices-demo/microservices-demo.git
Cloning into 'microservices-demo'...
remote: Enumerating objects: 10197, done.
remote: Counting objects: 100% (476/476), done.
remote: Compressing objects: 100% (181/181), done.
remote: Total 10197 (delta 327), reused 377 (delta 284), pack-reused 9721
Receiving objects: 100% (10197/10197), 52.95 MiB | 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#
39m
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 "payment" created
service "payment" created
deployment "queue-Control Plane" created
service "queue-Control Plane" created
Close
Skip
Check

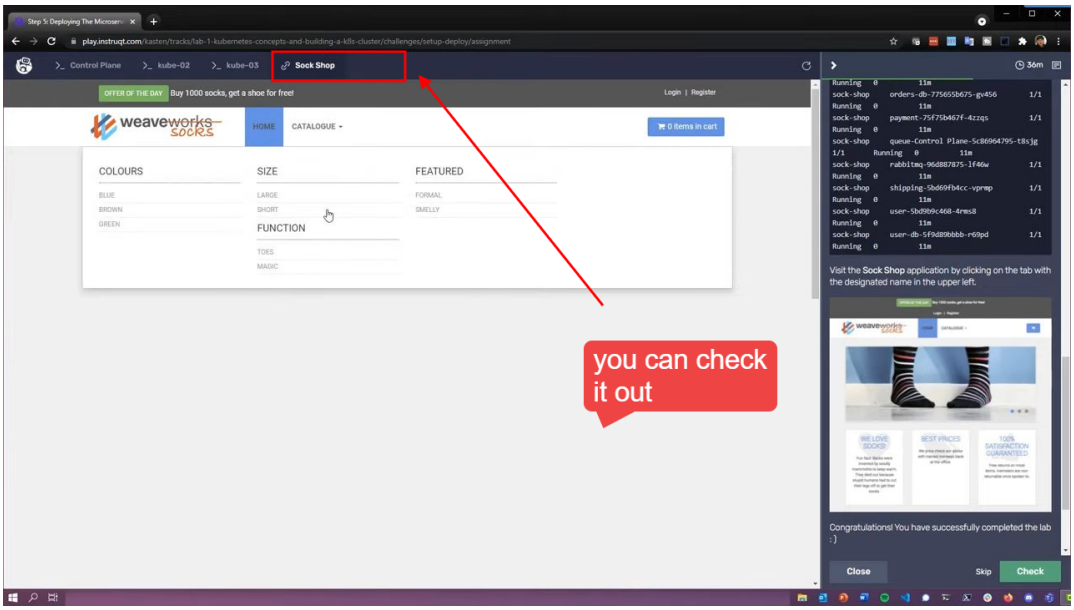
17:33


Step 5: Deploying The Microserv
play.instruqt.com/kasten/tracks/lab-1-kubernetes-concepts-and-building-a-k8s-cluster/challenges/setup-deploy/assignment
Control Plane
kube-02
kube-03
Sock Shop
39m
root@kube-01:~# git clone https://github.com/microservices-demo/microservices-demo.git
Cloning into 'microservices-demo'...
remote: Enumerating objects: 10197, done.
remote: Counting objects: 100% (476/476), done.
remote: Compressing objects: 100% (181/181), done.
remote: Total 10197 (delta 327), reused 377 (delta 284), pack-reused 9721
Receiving objects: 100% (10197/10197), 52.95 MiB | 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# ls
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/carts-db created
service/carts-db created
deployment.apps/catalogue created
service/catalogue created
deployment.apps/catalogue-db created
service/catalogue-db created
deployment.apps/front-end created
service/front-end created
deployment.apps/orders created
service/orders created
deployment.apps/orders-db created
service/orders-db created
deployment.apps/payment created
service/payment 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/user-db created
service/user-db created
root@kube-01:~/microservices-demo/deploy/kubernetes#
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 "payment" created
service "payment" created
deployment "queue-Control Plane" created
service "queue-Control Plane" created
deployment "rabbitmq" created
service "rabbitmq" created
deployment "shipping" created
service "shipping" created
deployment "user-db" created
Close
Skip
Check

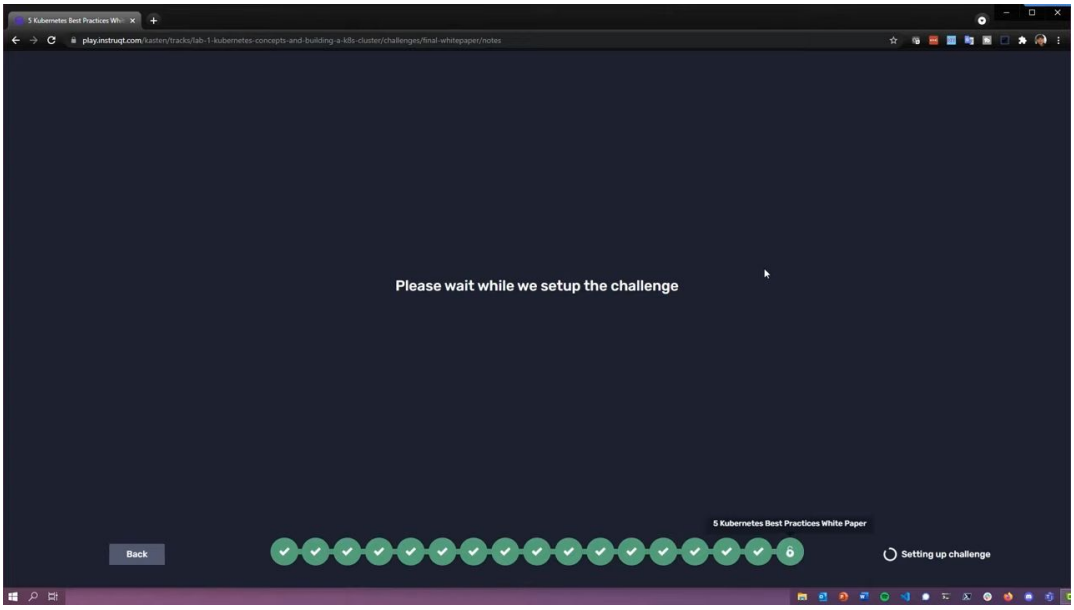
18:03


Step 5: Deploying The Microserv
play.instruqt.com/kasten/tracks/lab-1-kubernetes-concepts-and-building-a-k8s-cluster/challenges/setup-deploy/assignment
Control Plane
kube-02
kube-03
Sock Shop
38m
remote: Counting objects: 100% (476/476), done.
remote: Compressing objects: 100% (181/181), done.
remote: Total 10197 (delta 327), reused 377 (delta 284), pack-reused 9721
Receiving objects: 100% (10197/10197), 52.95 MiB | 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# ls
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/carts-db created
service/carts-db created
deployment.apps/catalogue created
service/catalogue created
deployment.apps/catalogue-db created
service/catalogue-db created
deployment.apps/front-end created
service/front-end created
deployment.apps/orders created
service/orders created
deployment.apps/orders-db created
service/orders-db created
deployment.apps/payment created
service/payment 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/user-db created
service/user-db created
checking
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# watch kubectl get pods --namespace sock-shop
deployment "queue-Control Plane" created
service "queue-Control Plane" created
deployment "rabbitmq" created
service "rabbitmq" created
deployment "shipping" created
service "shipping" created
deployment "user-db" created
service "user-db" created
deployment "user" created
service "user" created
Check to see if all of your pods are running:
watch kubectl get pods --namespace sock-shop
CTRL+C to exit
You will see the following result when all pods are ready, they will have the status of "Running":
NAMESPACE NAME READ
Y STATUS RESTARTS AGE
sock-shop carts-74f455bcb8-h9924 1/1
Running 0 11m
sock-shop carts-db-7fcddfbc79-v64fw 1/1
Running 0 11m
sock-shop catalogue-676d4b9f7c-5sn4g 1/1
Running 0 11m
sock-shop catalogue-db-5c67cdc8cd-hvk96 1/1
Running 0 11m
sock-shop front-end-977bfd86-hq9x9 1/1
Running 0 11m
sock-shop orders-787bf5b89f-xfdl6 1/1
Running 0 11m
sock-shop orders-db-775655b675-gv456 1/1
Running 0 11m
sock-shop payment-75f75b467f-4zzqs 1/1
Running 0 11m
sock-shop queue-Control Plane-5c86964795-t8sjg
1/1 Running 0 11m
sock-shop rabbitmq-96d887875-1f46w 1/1
Close
Skip
Check

18:43


Control Plane
kube-02
kube-03
Sock Shop
OFFER OF THE DAY
Buy 1000 socks, get a shoe for free!
Login | Register
weaveworks socks
HOME
CATALOGUE
0 items in cart
COLOURS
SIZE
FEATURED
BLUE
BROWN
GREEN
LARGE
SHORT
FUNCTION
TOES
MAGIC
FORMAL
SMELLY
you can check it out
Visit the Sock Shop application by clicking on the tab with the designated name in the upper left.
Congratulations! You have successfully completed the lab :)
Close
Skip
Check

20:26


Please wait while we setup the challenge
5 Kubernetes Best Practices White Paper
Back
Setting up challenge

21:09