

Be part of a better internet. [Get 20% off membership for a limited time](#)

This member-only story is on us. [Upgrade](#) to access all of Medium.

★ Member-only story

Guide: OKD 4.5 Single Node Cluster



Craig Robinson · Follow

Published in The Startup · 12 min read · Jul 15, 2020

127

13

+

•

↑

...

Updated: 7/29/2020

After listening to some feedback in the chat on a recent [Twitch stream](#) and the okd-wg mailing list I decided to create a guide for installing an OKD 4.5 SNC (single node cluster). This guide will use no worker nodes and only a single control-plane node with the goal of reducing the total amount of resources needed to get OKD up and running for a test drive.

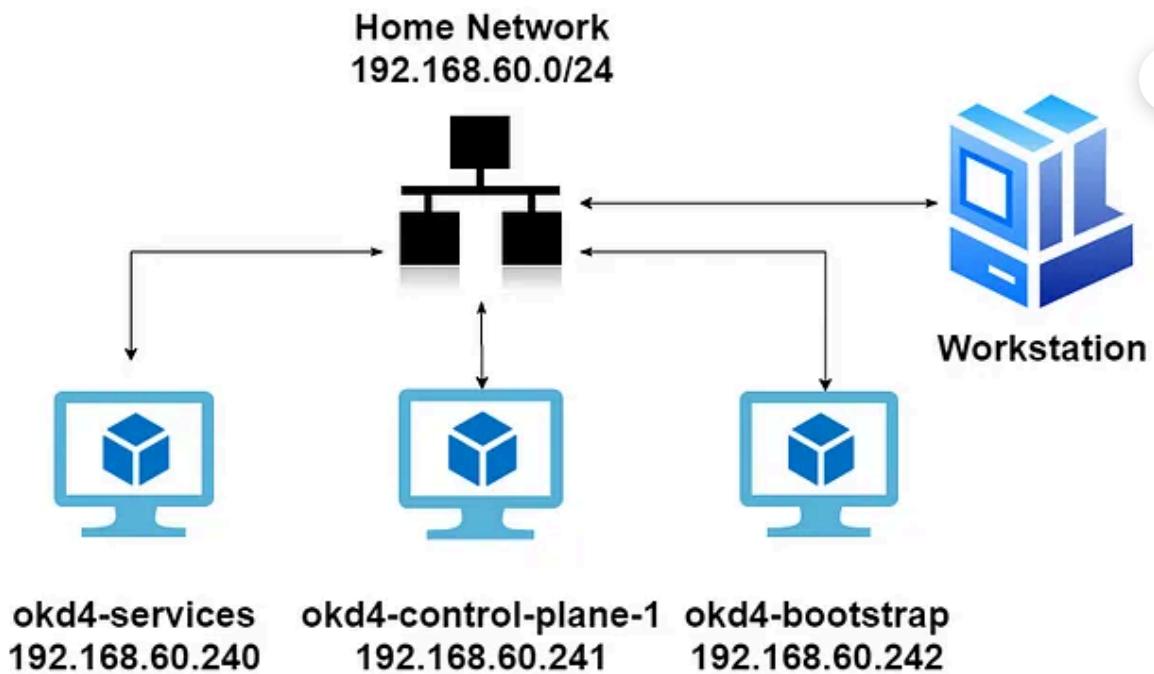
VM Overview:

For this installation, I used an ESXi 6.5 host on my local network and allocated the total virtual machine RAM usage to 22GB. ESXi 6.5 has a minimum of 4GB of RAM but you should be able to run this on an ESXi host

with 24GB of RAM. Here is a breakdown of the virtual machines and network layout:

Machine	Type	OS	vCPU	RAM	Storage	IP Address
okd4-services	DNS/LB/Web/NFS	Fedora Server	2	2	100	192.168.60.240
okd4-control-plane-1	Master	Fedora CoreOS	8	16	120	192.168.60.241
okd4-bootstrap	Bootstrap	Fedora CoreOS	4	4	120	192.168.60.242

Network Layout



Create the okd4-services VM:

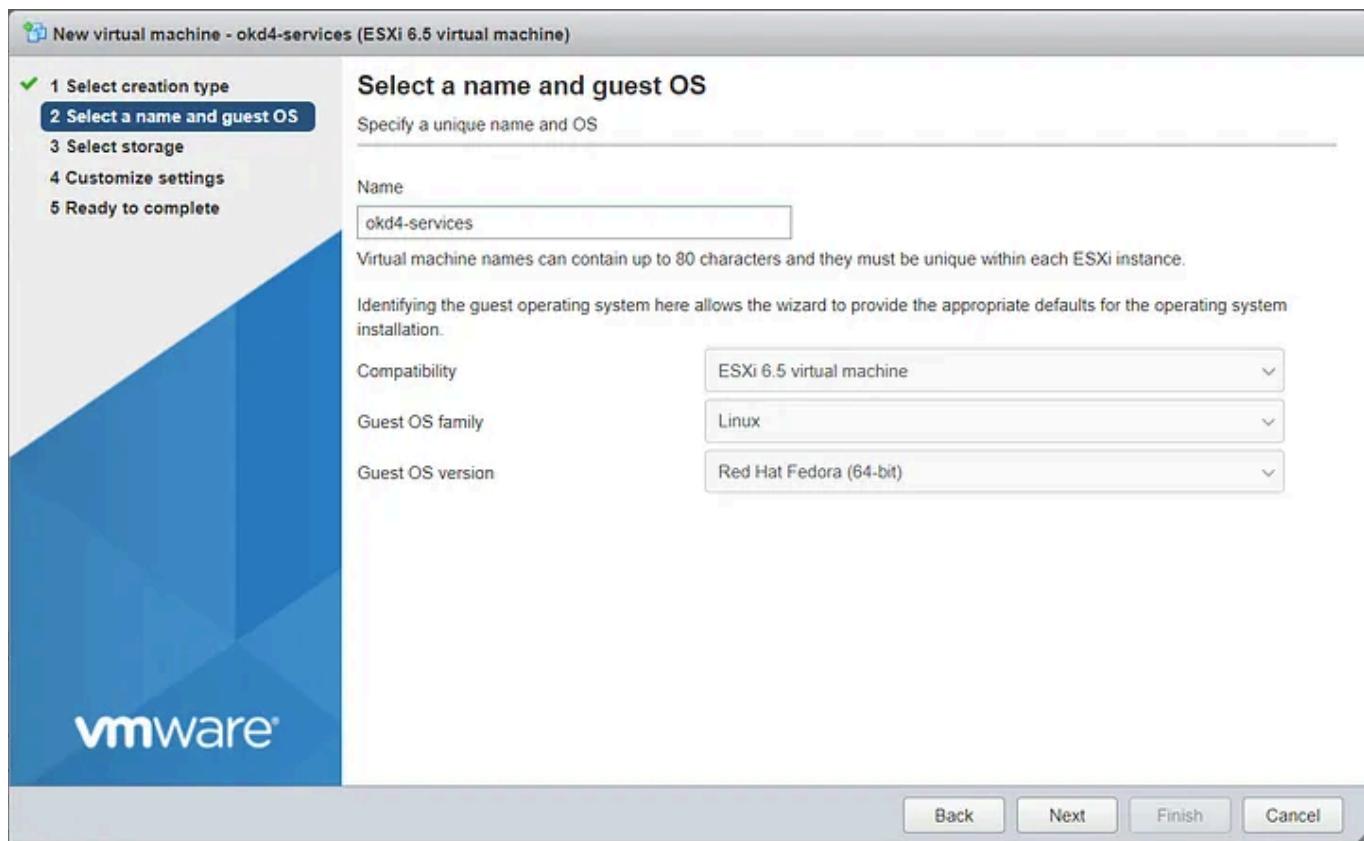
Download the Fedora Server Standard ISO and upload it to your ESXi host datastore.



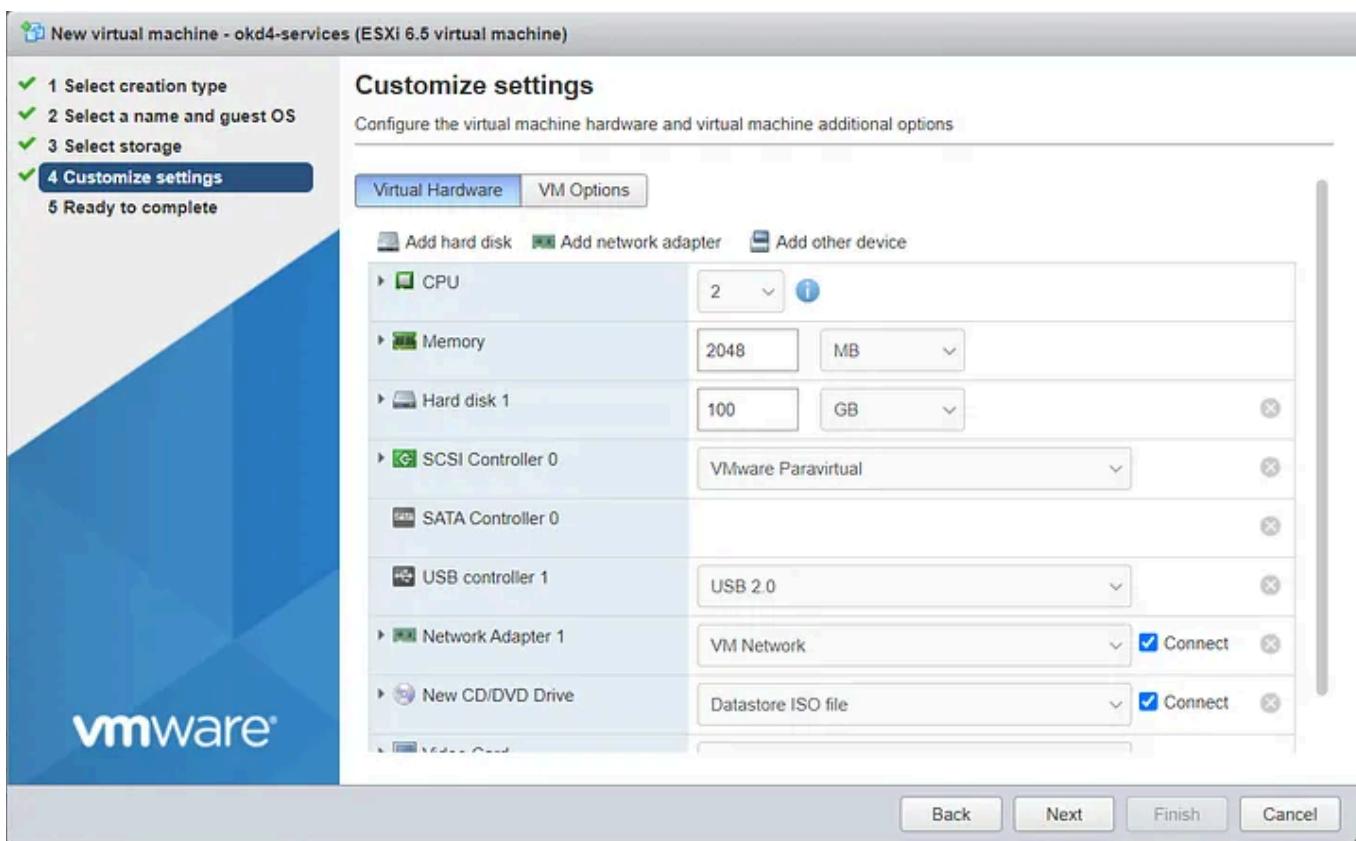
Download Fedora 32 Server.

A screenshot of a web page titled "Fedora Server is available". It features a heading "Using the x86_64 Architecture?" followed by two download options: "Fedora 32: Standard ISO image for x86_64" and "Fedora 32: Netinstall ISO image for x86_64", each with a "Download" button.

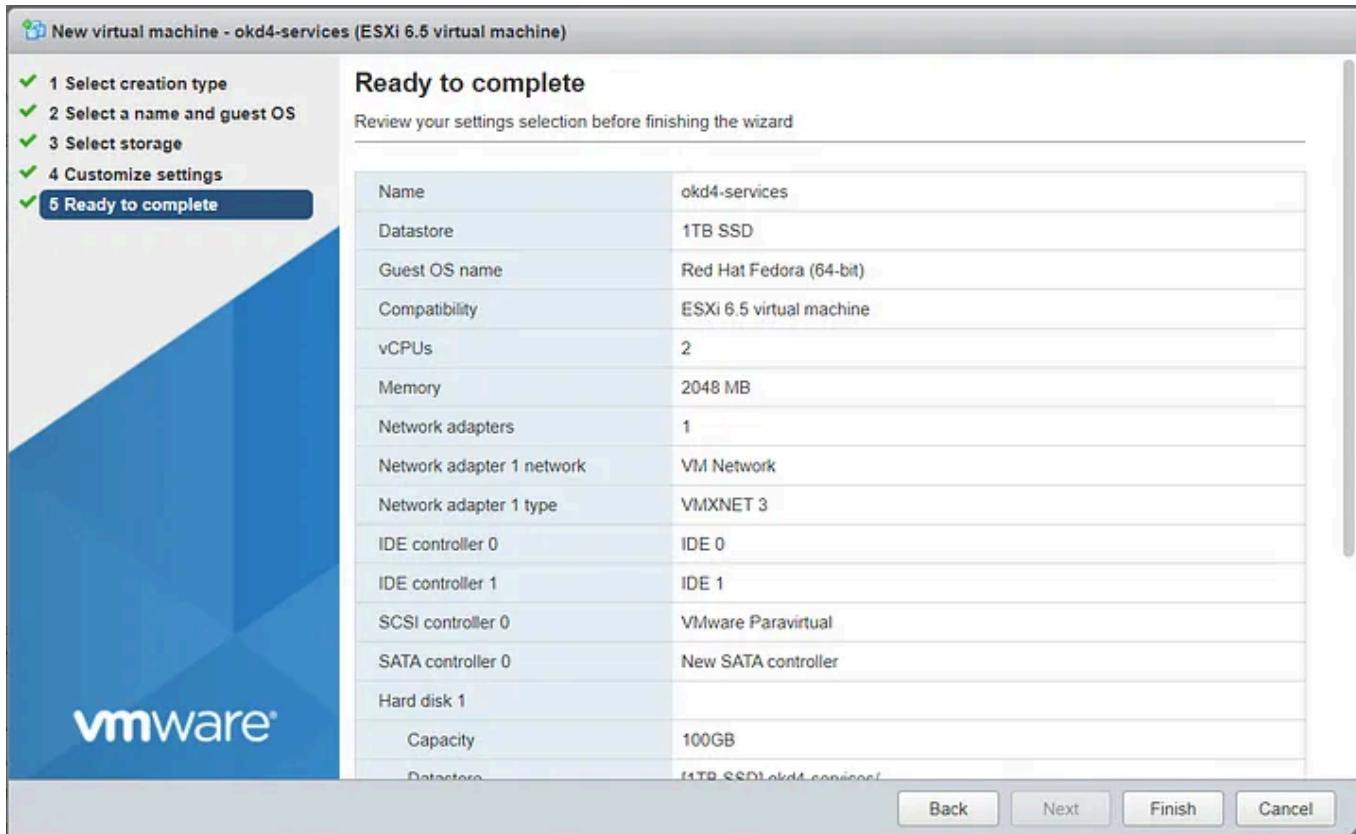
Create a new Virtual Machine. Choose Guest OS as *Linux* and Select CentOS 7 (64-bit).



Select the Datastore and customize the settings to 2 vCPU, 2GB RAM, 100GB HD. Attach the Fedora Server ISO to the CD/DVD drive.

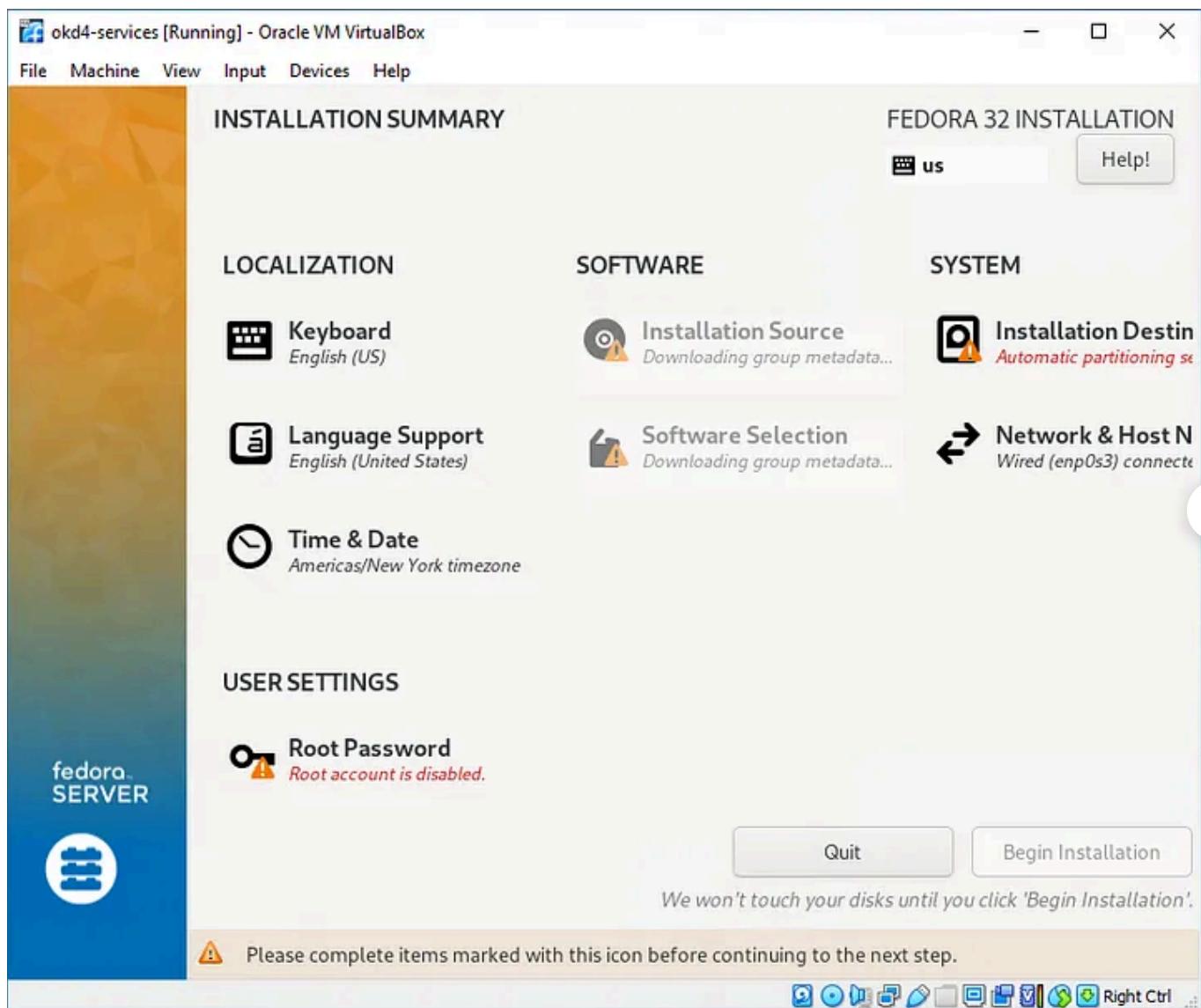


Review your settings and click Finish.



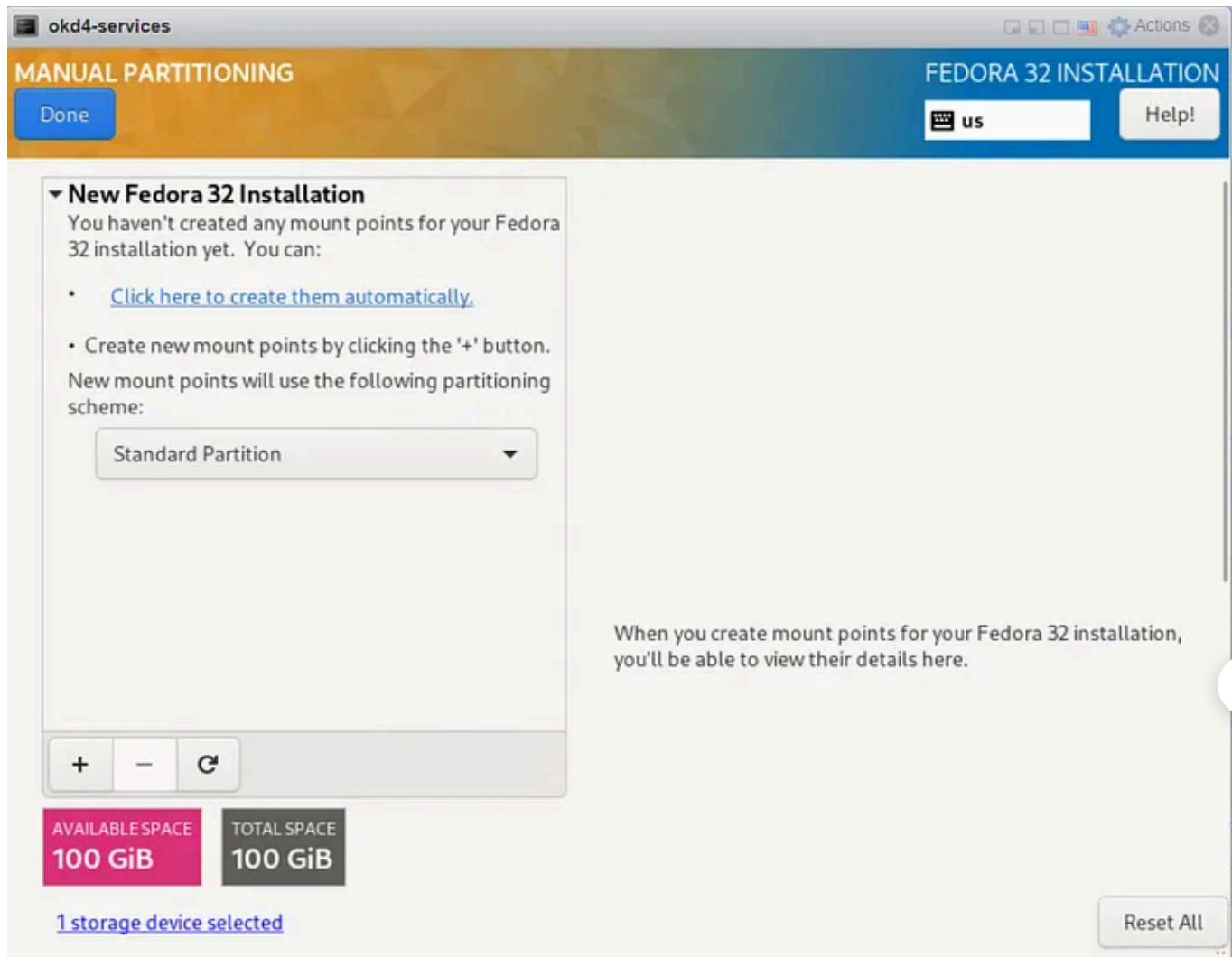
Install Fedora Server 32 on the okd4-services VM:

Run through the installation.

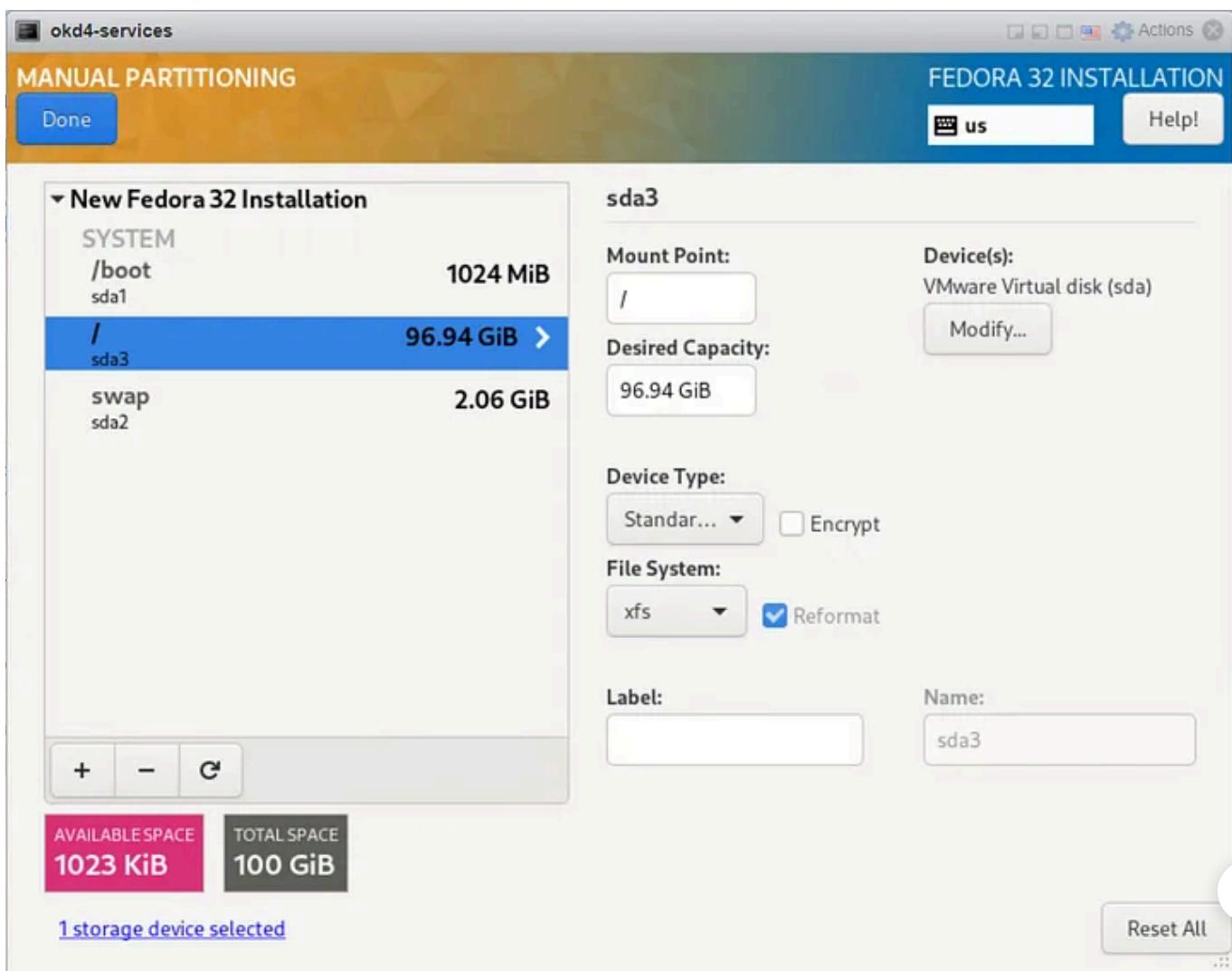


I prefer to use the “Standard Partition” storage configuration to use the entire drive at “/”. On the “Installation Destination” page, click on Custom under Storage Configuration, then Done.

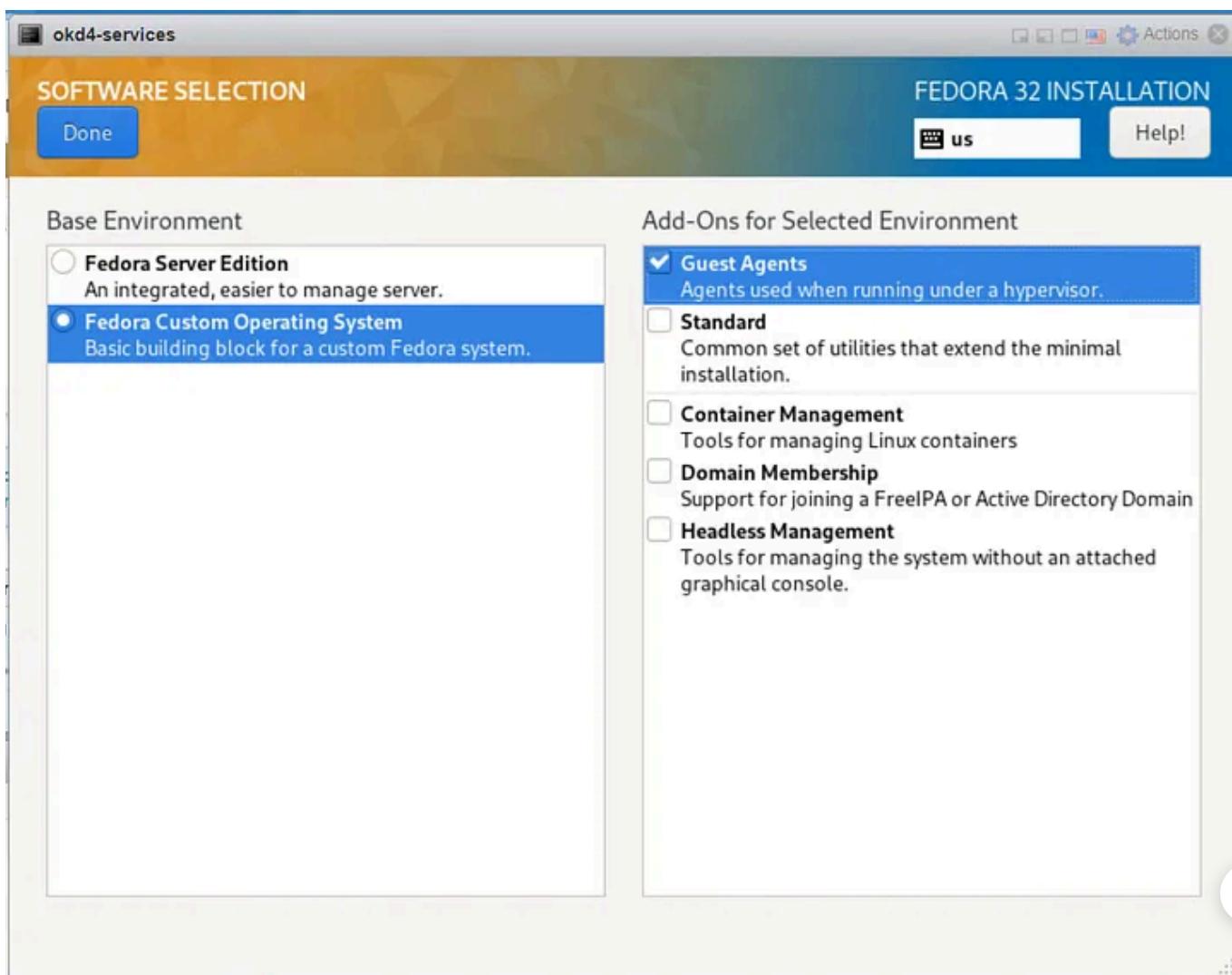
On the Manual Partitioning page, select Standard Partition, then click “Click Here to Create them automatically.”



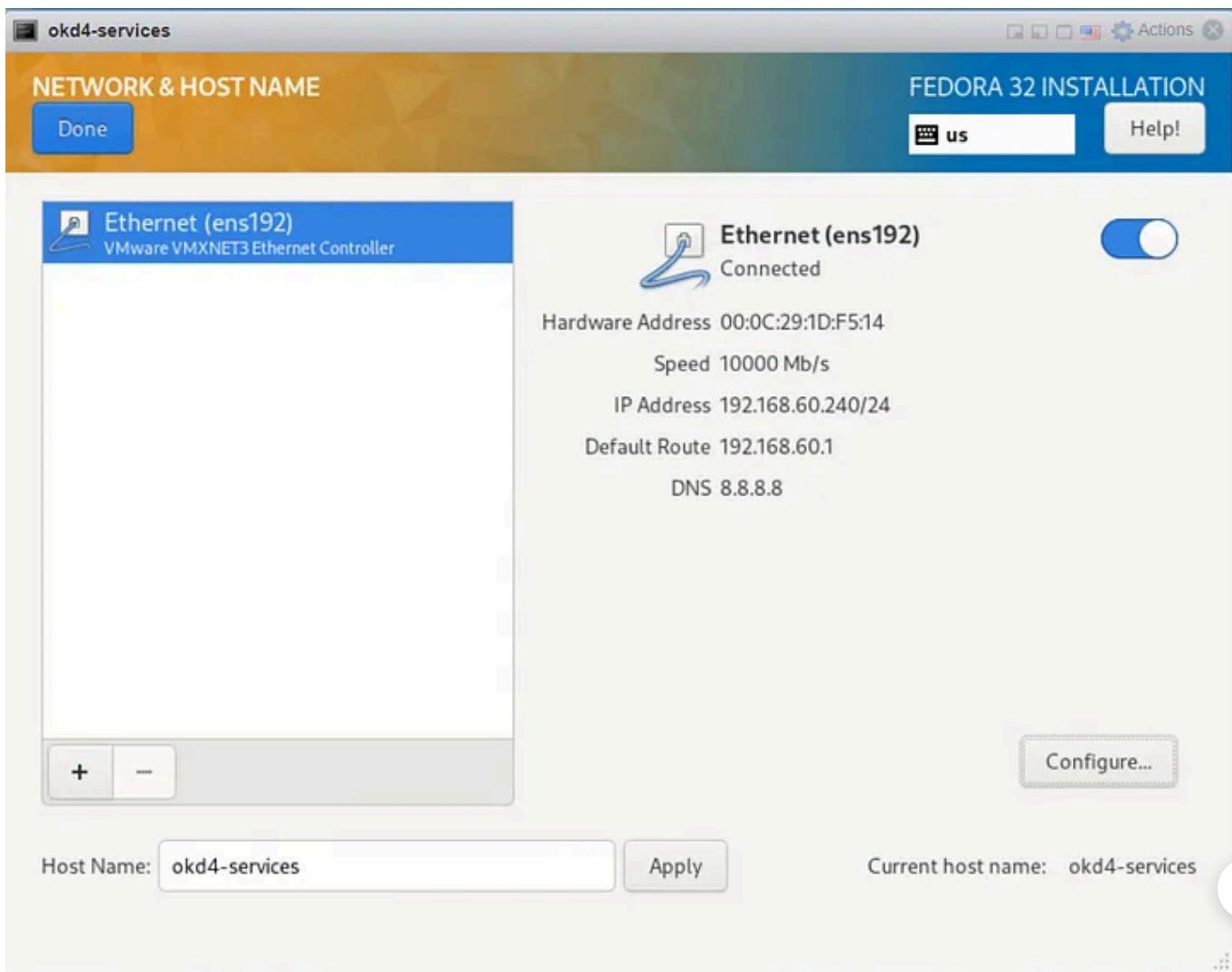
Select the “/” partition and click the “-” to delete it. Then click the “+” to re-add it. At this point, it should use the entire drive. Click Done, then Accept the changes.



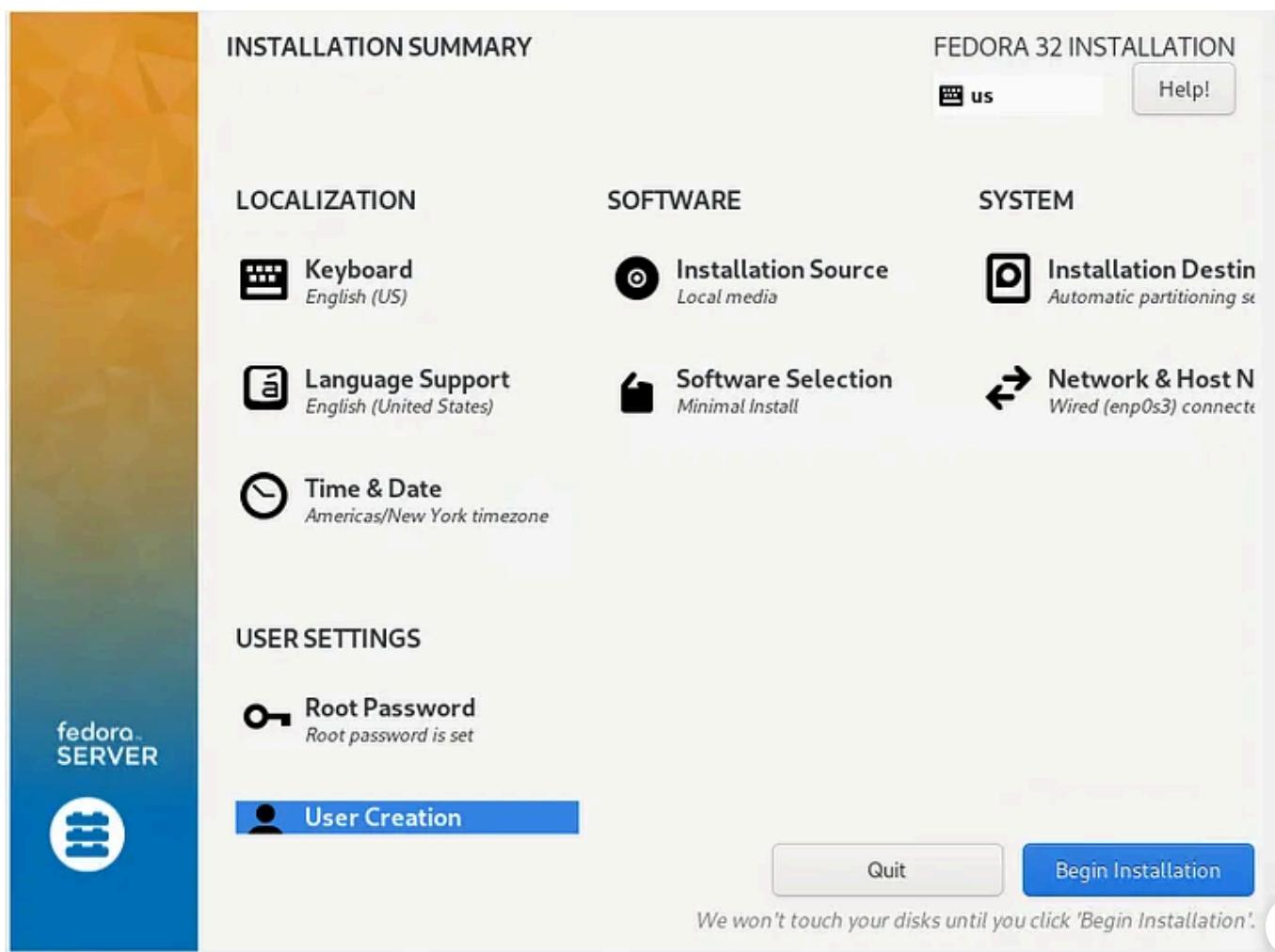
For Software Selection, choose “Fedora Custom Operating System” and include the “Guest Agents” add-on.



Enable the NIC connected to the VM Network and set the hostname as okd4-services, then click Apply and Done.



Set the Root password, and create an admin user. Click “Begin Installation” to start the install.



After the installation has completed, login, and update the OS.

```
sudo dnf update -y
sudo dnf install -y git wget vim
sudo init 6
```

Create bootstrap and master nodes:

Download the [Fedora CoreOS Bare Metal ISO](#) and upload it to your ESXi datastore.

The latest stable version at the time of writing is 32.20200715.3.0

Stable
v 32.20200715.3.0
[JSON](#) — 3 hours ago

The Stable stream should be used by production clusters. Versions of Fedora CoreOS are battle-tested within the Testing and Next streams before being promoted.

[Show Downloads](#)

Testing
v 32.20200726.2.0
[JSON](#) — 3 hours ago

The Testing stream consists of promoted Next releases. Mix a few Testing machines into your production clusters to catch any bugs specific to your hardware or configuration.

[Show Downloads](#)

Currently displayed stream: **Stable** ([View Releases](#))

[Cloud Launchable](#)

[Bare Metal & Virtualized](#)

Bare Metal

ISO

(iso)

32.20200715.3.0 stable

[Download](#)

[Verify signature & SHA256](#)

Virtualized

OpenStack

(qcow2.xz)

32.20200715.3.0 st

[Download](#)

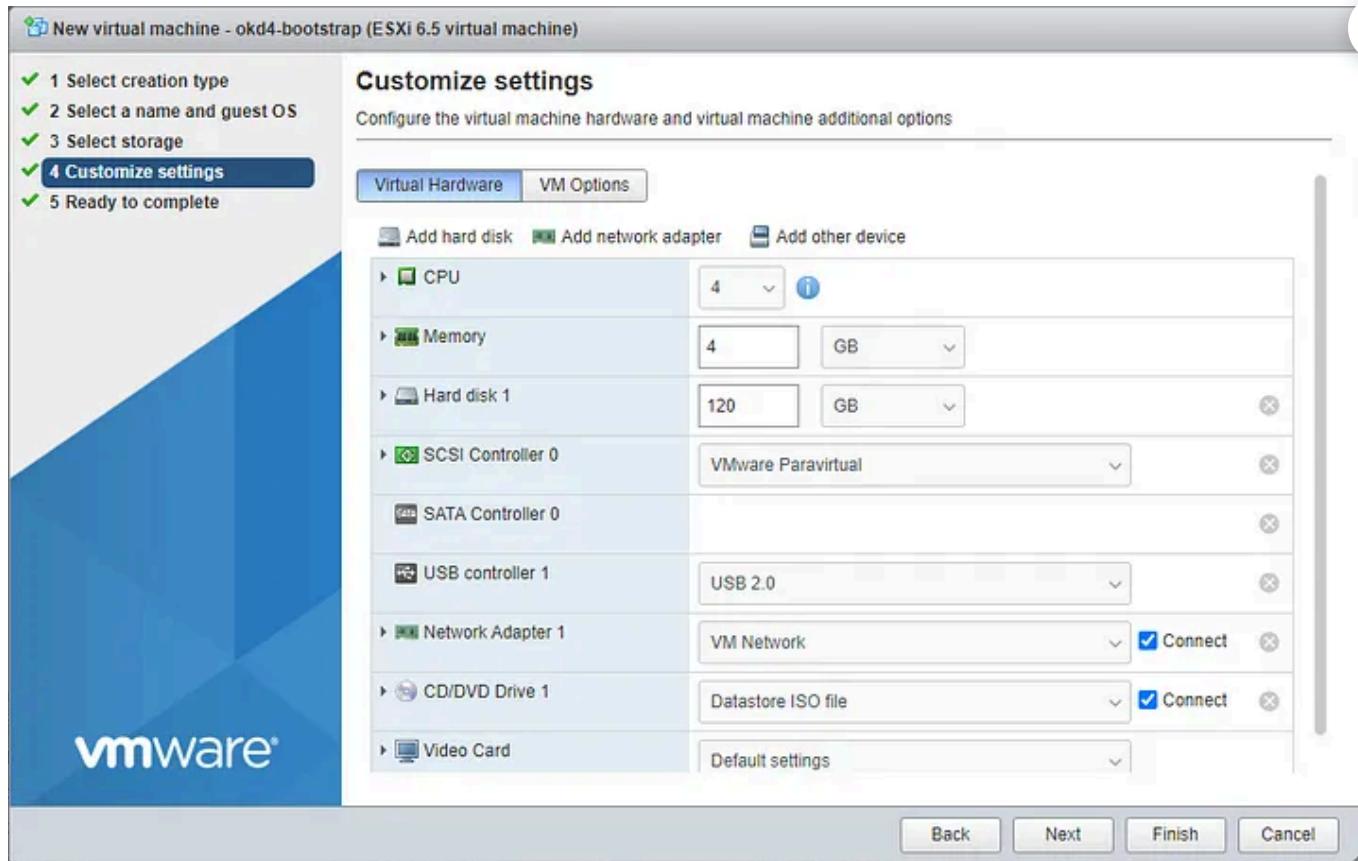
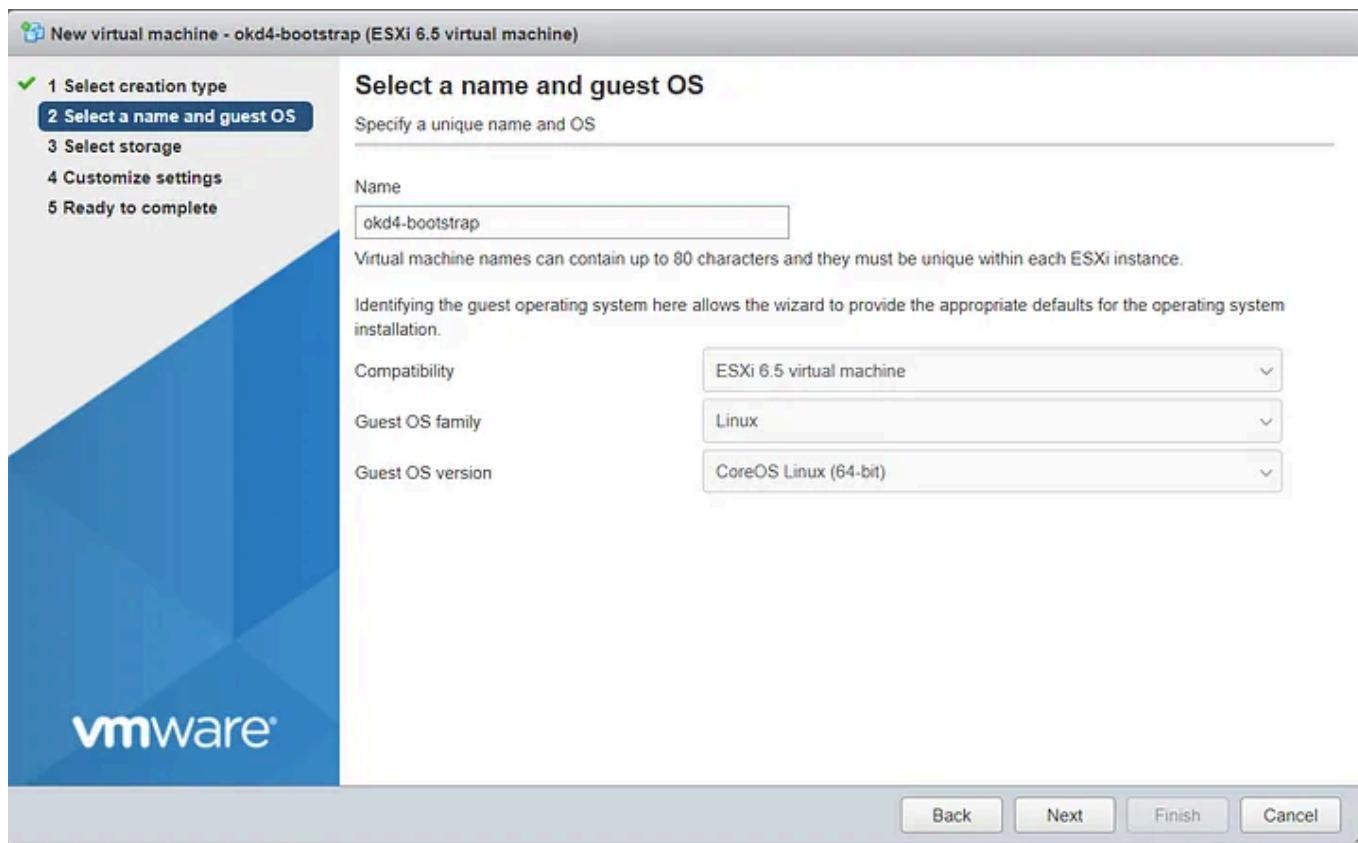
[Verify signature & S](#)

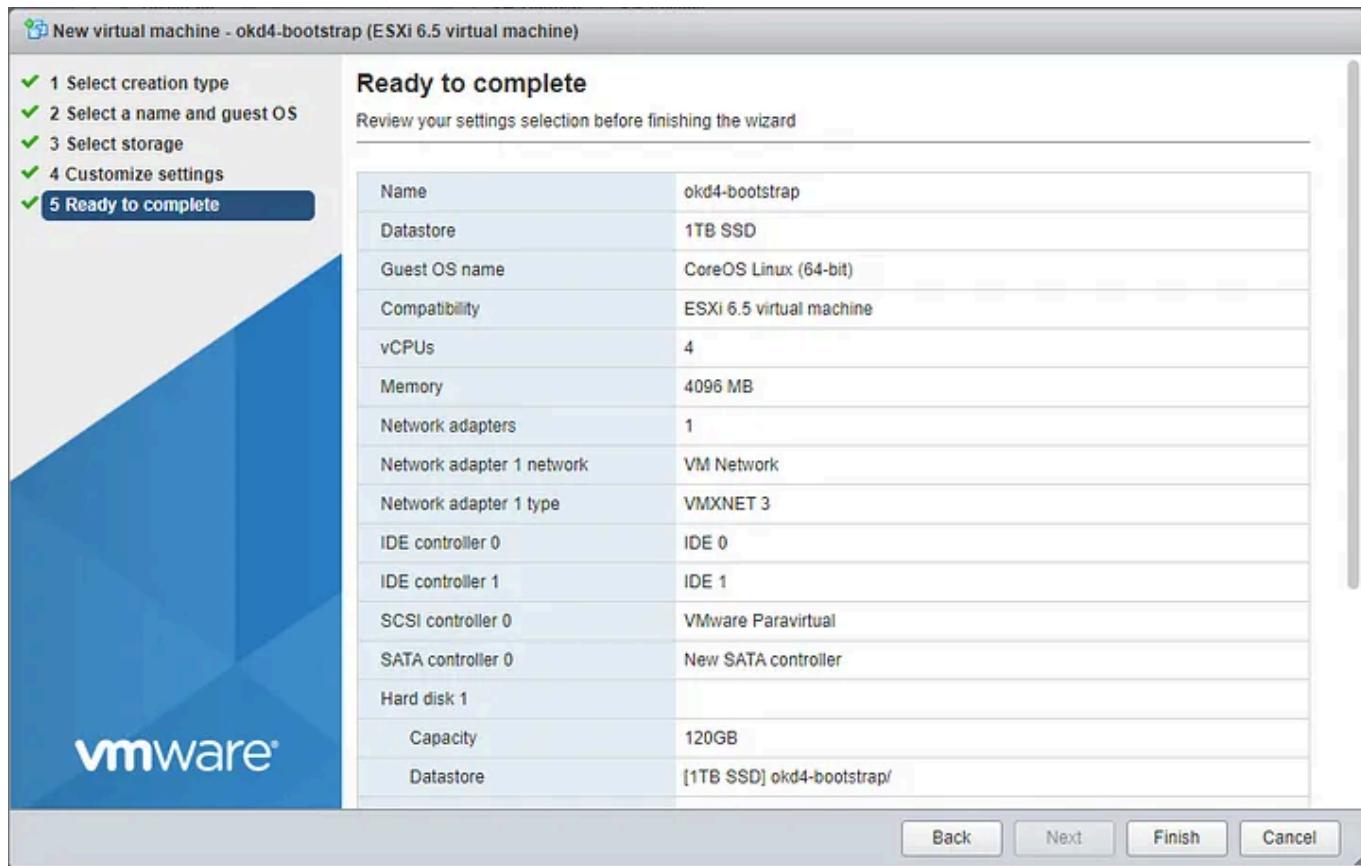
PXE

QEMU

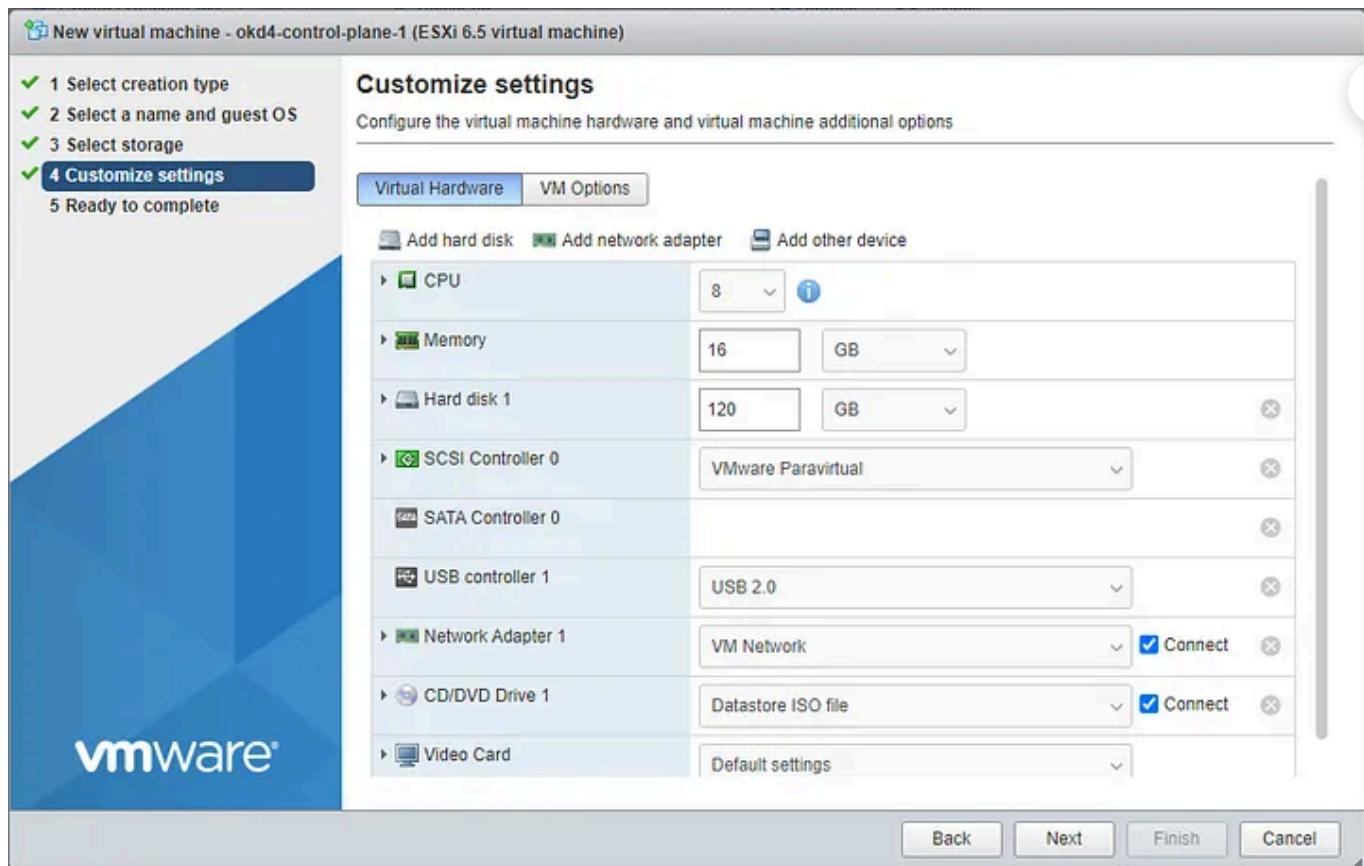
Create the two ODK nodes (okd4-bootstrap and okd4-control-plane-1) on your ESXi host using the values in the spreadsheet at the beginning of this post:

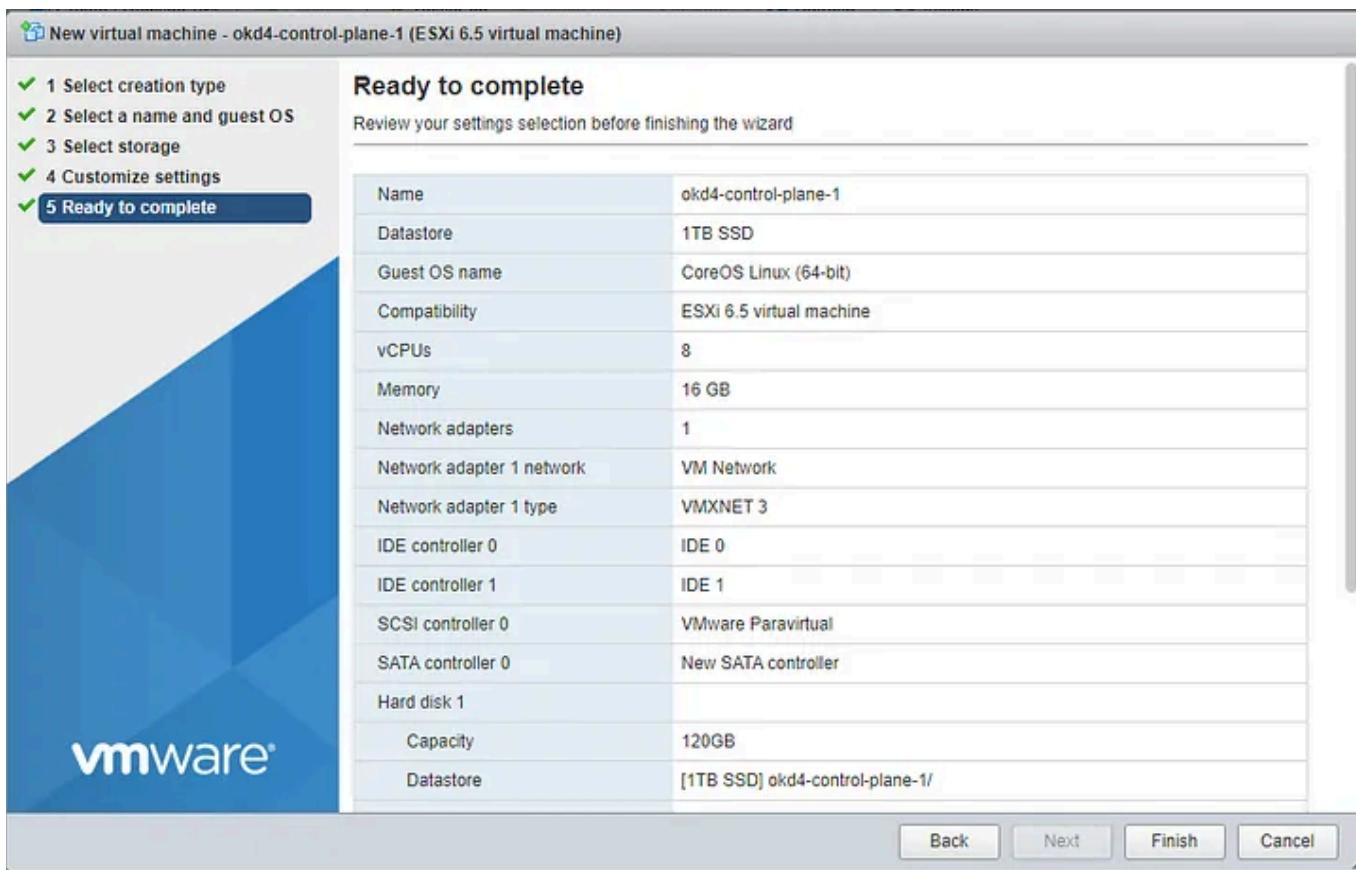
okd4-bootstrap:





okd4-control-plane-1





You should end up with the following VMs:

<input type="checkbox"/> okd4-services	Normal	102.11 GB	Red Hat Fedora (64-bit)	okd4-services	67 MHz	2.05 GB
<input type="checkbox"/> okd4-bootstrap	Normal	0 B	CoreOS Linux (64-bit)	Unknown	0 MHz	0 MB
<input type="checkbox"/> okd4-control-plane-1	Normal	0 B	CoreOS Linux (64-bit)	Unknown	0 MHz	0 MB
Quick filters... <input type="button" value="▼"/>						4 items

Configure okd4-services VM to host various services:

The okd4-services VM is used to provide DNS, NFS exports, web server, and load balancing.

Open a terminal on the okd4-services VM and clone the okd4_files repo (branch snc) that contains the DNS, HAProxy, and install-conf.yaml example files:

```
cd  
git clone -b snc https://github.com/cragr/okd4\_files.git  
cd okd4_files
```

Install bind (DNS)

```
sudo dnf -y install bind bind-utils
```

Copy the named config files and zones:

```
sudo cp named.conf /etc/named.conf  
sudo cp named.conf.local /etc/named/  
sudo mkdir /etc/named/zones  
sudo cp db* /etc/named/zones
```

Enable and start named:

```
sudo systemctl enable named  
sudo systemctl start named  
sudo systemctl status named
```

Create firewall rules:

```
sudo firewall-cmd --permanent --add-port=53/udp
```

```
sudo firewall-cmd --reload
```

Change the DNS on the okd4-service to 127.0.0.1:

```
sudo nmcli connection modify ens192 ipv4.dns "127.0.0.1"
```

Restart the network services on the okd4-services VM:

```
sudo systemctl restart NetworkManager
```

Test DNS on the okd4-services.

```
dig okd.local  
dig -x 192.168.60.240
```

With DNS working correctly, you should see the following results:

```
[crobinson@okd4-services okd_files]$ dig okd.local

; <>> DiG 9.11.13-RedHat-9.11.13-3.el8 <>> okd.local
;; global options: +cmd
;; Got answer:
;; WARNING: .local is reserved for Multicast DNS
;; You are currently testing what happens when an mDNS query is leaked to DNS
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 64424
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 7009863cf0c3194df61cc045f04c51c67e06b2f24fd0618 (good)
; QUESTION SECTION:
;okd.local.          IN      A

;; AUTHORITY SECTION:
okd.local.      604800  IN      SOA      okd4-services.okd.local. admin.o
okd.local. 1 604800 86400 2419200 604800

;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Tue Jul 07 14:55:24 EDT 2020
;; MSG SIZE  rcvd: 122
```

```
[crobinson@okd4-services okd4_files]$ dig -x 192.168.1.210

; <>> DiG 9.11.13-RedHat-9.11.13-3.el8 <>> -x 192.168.1.210
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 65007
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 1, ADDITIONAL: 2

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 23bfcaa3f3140ab1261acff65f04c4d921fa9f8011631ba6 (good)
;; QUESTION SECTION:
;210.1.168.192.in-addr.arpa. IN PTR

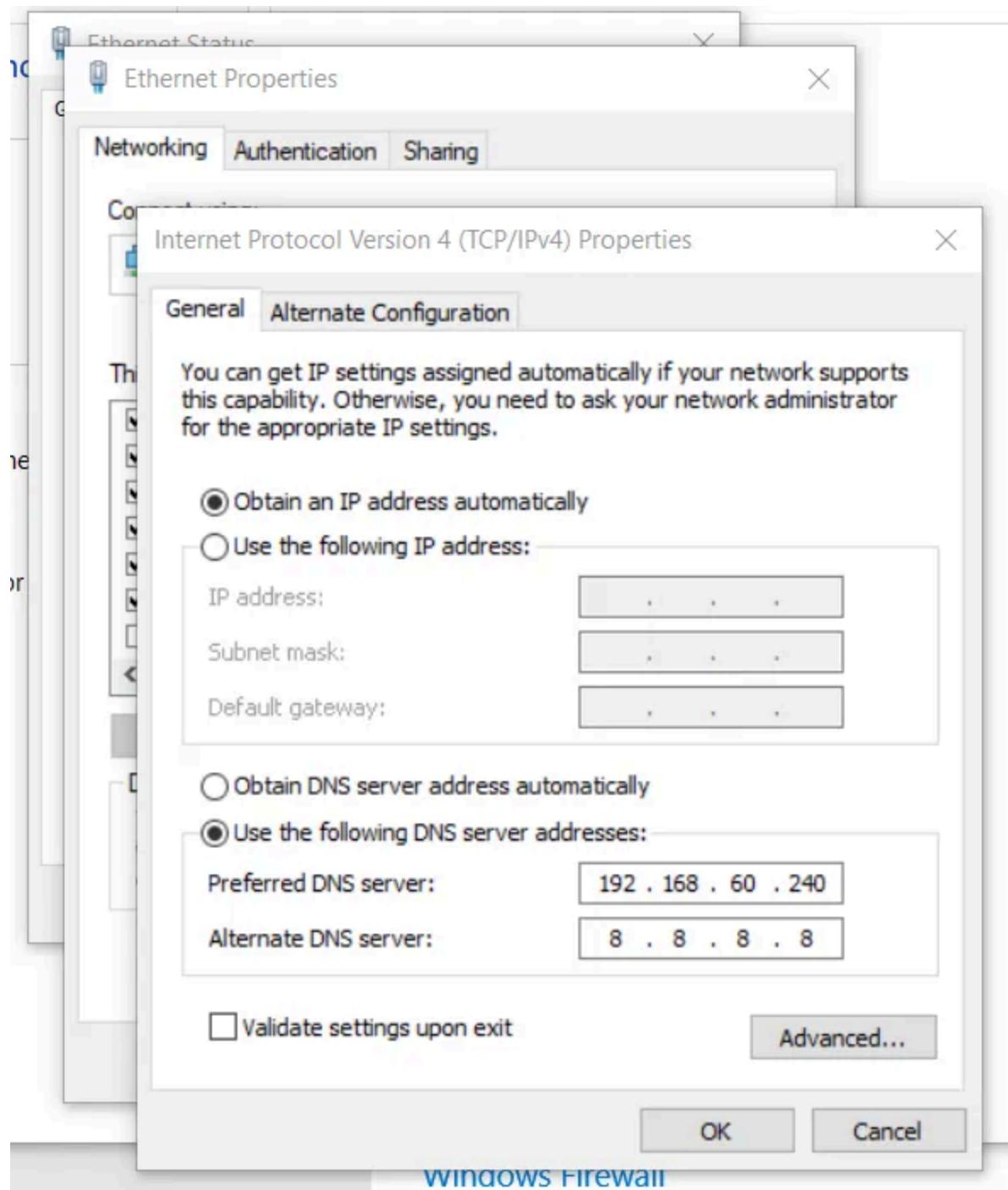
;; ANSWER SECTION:
210.1.168.192.in-addr.arpa. 604800 IN PTR api-int.lab.okd.local.
210.1.168.192.in-addr.arpa. 604800 IN PTR okd4-services.okd.local.
210.1.168.192.in-addr.arpa. 604800 IN PTR api.lab.okd.local.

;; AUTHORITY SECTION:
1.168.192.in-addr.arpa. 604800 IN NS okd4-services.okd.local.

;; ADDITIONAL SECTION:
okd4-services.okd.local. 604800 IN A 192.168.1.210

;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Tue Jul 07 14:54:17 EDT 2020
;; MSG SIZE rcvd: 194
```

Change your workstation's primary DNS to match the okd4-services IP, 192.168.60.240 so you can resolve the DNS entries.



Install HAProxy:

```
sudo dnf install haproxy -y
```

Copy haproxy config from the git okd4_files directory :

```
sudo cp haproxy.cfg /etc/haproxy/haproxy.cfg
```

Start, enable, and verify HA Proxy service:

```
sudo setsebool -P haproxy_connect_any 1  
sudo systemctl enable haproxy  
sudo systemctl start haproxy  
sudo systemctl status haproxy
```

Add OKD firewall ports:

```
sudo firewall-cmd --permanent --add-port=6443/tcp  
sudo firewall-cmd --permanent --add-port=22623/tcp  
sudo firewall-cmd --permanent --add-service=http  
sudo firewall-cmd --permanent --add-service=https  
sudo firewall-cmd --reload
```

Install Apache/HTTPD

```
sudo dnf install -y httpd
```

Change httpd to listen port to 8080:

```
sudo sed -i 's/Listen 80/Listen 8080/' /etc/httpd/conf/httpd.conf
```

Enable and Start httpd service/Allow port 8080 on the firewall:

```
sudo setsebool -P httpd_read_user_content 1  
sudo systemctl enable httpd  
sudo systemctl start httpd  
sudo firewall-cmd --permanent --add-port=8080/tcp  
sudo firewall-cmd --reload
```

Test the webserver:

```
curl localhost:8080
```

A successful curl should look like this:

```
<h2>推广 Apache 及 CentOS</h2>
<p>你可在 Apache 及 CentOS Linux 驱动的 HTTP 服务器上随意采用下列图像。多谢采用 Apache 及 CentOS! </p>
<p>
    <a href="http://httpd.apache.org/">
        
    </a>
    <a href="http://www.centos.org/">
        
    </a>
</p>
</div>
<div class="col-md-6">
</div>
<div class="col-md-6">
    <h2>CentOS 计划</h2>
    <p>CentOS Linux 发行版本是一个稳定、高预测性、高管理性、高重复性的平台，它源于 Red Hat 企业级 Linux (RHEL) 的源代码。</p>
    <p>除了是寄存网站的热门选择外，CentOS 亦为开源社群提供一个可扩展的丰富平台。请拜访 <a href="http://www.centos.org/">CentOS 网站</a> 获取更多信息。</p>
</div>
</section>
<hr/>
</main>
<footer class="container">
    <p>© 2019 CentOS 计划 | <a href="https://www.centos.org/legal/">法律条文</a> | <a href="https://www.centos.org/legal/privacy/">私隐</a></p>
</footer>
</body>
</html>
[crobinson@okd4-services okd4 files]$ $
```

Congratulations, You Are Half Way There!

Download the openshift-installer and oc client:

SSH to the okd4-services VM

To download the latest oc client and openshift-install binaries, you need to use an existing version of the oc client.

Download the 4.5 version of the oc client and openshift-install from the [OKD releases page](#). Example:

```
cd  
wget https://github.com/openshift/okd/releases/download/4.5.0-0.okd-  
2020-07-29-070316/openshift-client-linux-4.5.0-0.okd-2020-07-29-  
070316.tar.gz  
wget https://github.com/openshift/okd/releases/download/4.5.0-0.okd-  
2020-07-29-070316/openshift-install-linux-4.5.0-0.okd-2020-07-29-  
070316.tar.gz
```

[Open in app ↗](#)



Search



Write



```
tar -zxvf openshift-client-linux-4.5.0-0.okd-2020-07-29-070316.tar.gz  
tar -zxvf openshift-install-linux-4.5.0-0.okd-2020-07-29-  
070316.tar.gz
```

Move the kubectl, oc, and openshift-install to /usr/local/bin and show the version:

```
sudo mv kubectl oc openshift-install /usr/local/bin/  
oc version  
openshift-install version
```

The latest and recent releases are available at <https://origin-release.svc.ci.openshift.org>

Setup the openshift-installer:

In the install-config.yaml, you can either use a pull-secret from RedHat or the default of “{“auths”:{“fake”:{“auth”: “bar”}}}" as the pull-secret.

Generate an SSH key if you do not already have one.

```
ssh-keygen
```

Create an install directory and copy the install-config.yaml file:

```
cd  
mkdir install_dir  
cp okd4_files/install-config.yaml ./install_dir
```

Edit the install-config.yaml in the install_dir, add your ssh key, and edit the controlPlane master replicas to 1. Backup the install-config.yaml as it will be removed when generating the manifests:

```

crobinson@okd4-services:~$ cat install-config.yaml
apiVersion: v1
baseDomain: okd.local
metadata:
  name: lab

compute:
- hyperthreading: Enabled
  name: worker
  replicas: 0

controlPlane:
  hyperthreading: Enabled
  name: master
  replicas: 1

networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16

platform:
  none: {}

fips: false

pullSecret: '{"auths":{"fake":{"auth":"bar"}}}'
sshKey: 'ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQgQCfn8hWe2X48kfSj1PgJYFEpMjYRGJR8Tut
F+92BLZjG4/RR6EqbX585JQRiV/fJVmANIDezZy1dTGvgMrznjYZMQbiGinZc5xU0QncxHthMQuDUPWo3
ercnN7CNyCExaYhtFzB8waIQW6e4tR3lnsFF0E3+M72CzYH/ycOs5TPYT1yZXPM900f0foTSFXNhA2NJ+
aaOwKAkH7Vr6xHr0Ba+XAT1Fi/JYQVUdC9fdH4vLVDthTJG9JDP1PbVLAOP07ds9jGcuI1Ez1Htyw/9QE
obinson@okd4-services'

```

```

vim ./install_dir/install-config.yaml
cp ./install_dir/install-config.yaml ./install_dir/install-
config.yaml.bak

```

Generate the Kubernetes manifests for the cluster, ignore the warning:

```
openshift-install create manifests --dir=install_dir/
```

Now you can create the ignition-configs:

```
openshift-install create ignition-configs --dir=install_dir/
```

Note: If you reuse the install_dir, make sure it is empty. Hidden files are created after generating the configs, and they should be removed before you use the same folder on a 2nd attempt.

Host ignition and Fedora CoreOS files on the webserver:

Create okd4 directory in /var/www/html:

```
sudo mkdir /var/www/html/okd4
```

Copy the install_dir contents to /var/www/html/okd4 and set permissions:

```
sudo cp -R install_dir/* /var/www/html/okd4/
sudo chown -R apache: /var/www/html/
sudo chmod -R 755 /var/www/html/
```

Test the webserver:

```
curl localhost:8080/okd4/metadata.json
```

Download the Fedora CoreOS bare-metal bios image and sig files and shorten the file names:

```
cd /var/www/html/okd4/
sudo wget
https://builds.coreos.fedoraproject.org/prod/streams/stable/builds/32
.20200715.3.0/x86_64/fedora-coreos-32.20200715.3.0-
metal.x86_64.raw.xz
sudo wget
https://builds.coreos.fedoraproject.org/prod/streams/stable/builds/32
.20200715.3.0/x86_64/fedora-coreos-32.20200715.3.0-
metal.x86_64.raw.xz.sig
sudo mv fedora-coreos-32.20200715.3.0-metal.x86_64.raw.xz fcose.raw.xz
sudo mv fedora-coreos-32.20200715.3.0-metal.x86_64.raw.xz.sig
fcose.raw.xz.sig
sudo chown -R apache: /var/www/html/
sudo chmod -R 755 /var/www/html/
```

Starting the bootstrap node:

NOTE: I recommend you use a DHCP lease for both the bootstrap and control-plane nodes if possible instead of the line below beginning with “ip=” which will set static IP addresses.

Power on the odk4-bootstrap VM. Press the TAB key to edit the kernel boot options. Once completed, press Enter to boot.

```
ip=192.168.60.242::192.168.60.1:255.255.255.0:::none
nameserver=192.168.60.240 coreos.inst.install_dev=/dev/sda
```

```
coreos.inst.image_url=http://192.168.60.240:8080/okd4/fcos.raw.xz
coreos.inst.ignition_url=http://192.168.60.240:8080/okd4/bootstrap.ign
n
```



You should see that the fcos.raw.xz image and signature are downloading:

```

okd4-control-plane-1
Starting Network Manager...
Starting D-Bus System Message Bus...
[ OK ] Started D-Bus System Message Bus.
[ OK ] Started Network Manager.
[ OK ] Reached target Network.
Starting Network Manager Wait Online...
Starting Hostname Service...
[ OK ] Started Hostname Service.
[ OK ] Listening on Load/Save RF Kill Switch Status /dev/rfkill Watch.
Starting Network Manager Script Dispatcher Service...
[ OK ] Started Network Manager Script Dispatcher Service.
[ OK ] Finished Network Manager Wait Online.
[ OK ] Reached target Network is Online.
Starting CoreOS Installer...
#####
11.552419] coreos-installer-service[977]: coreos-installer install /dev/sda --ignition /tmp/coreos-installer-DjUysn --firstboot-args rd.neednet=1 ip=192.168.60.241::192.168.60.1:255.255.255.0:::none nameserver=192.168.60.240 --image-url http://192.168.60.240:8080/okd4/fcos.raw.xz
11.582360] coreos-installer-service[994]: Downloading image from http://192.168.60.240:8080/okd4/fcos.raw.xz
11.583602] coreos-installer-service[994]: Downloading signature from http://192.168.60.240:8080/okd4/fcos.raw.xz.sig
12.650966] coreos-installer-service[994]: Read disk 29.5 MiB/496.9 MiB (5%)
13.651162] coreos-installer-service[994]: Read disk 64.2 MiB/496.9 MiB (12%)
14.653976] coreos-installer-service[994]: Read disk 74.3 MiB/496.9 MiB (14%)
15.702414] coreos-installer-service[994]: Read disk 74.5 MiB/496.9 MiB (14%)
16.765728] coreos-installer-service[994]: Read disk 74.7 MiB/496.9 MiB (15%)
18.301552] coreos-installer-service[994]: Read disk 74.7 MiB/496.9 MiB (15%)
19.599644] coreos-installer-service[994]: Read disk 75.5 MiB/496.9 MiB (15%)
20.628859] coreos-installer-service[994]: Read disk 76.4 MiB/496.9 MiB (15%)
22.073134] coreos-installer-service[994]: Read disk 76.4 MiB/496.9 MiB (15%)
23.491240] coreos-installer-service[994]: Read disk 76.4 MiB/496.9 MiB (15%)
25.153267] coreos-installer-service[994]: Read disk 76.5 MiB/496.9 MiB (15%)
26.791653] coreos-installer-service[994]: Read disk 76.5 MiB/496.9 MiB (15%)
28.389935] coreos-installer-service[994]: Read disk 76.5 MiB/496.9 MiB (15%)
29.893554] coreos-installer-service[994]: Read disk 76.5 MiB/496.9 MiB (15%)

```

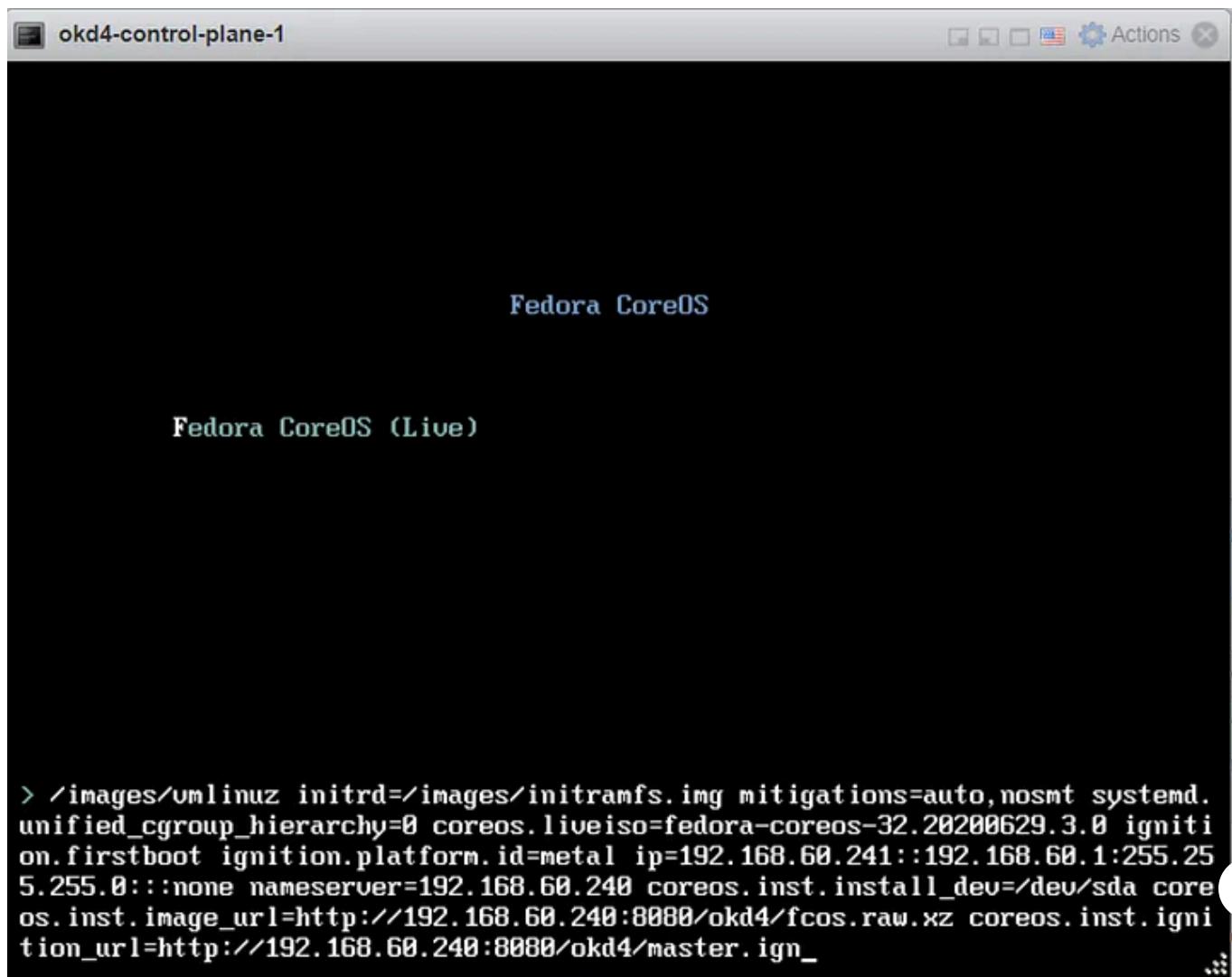
Starting the control-plane-1 node:

Power on the odk4-control-plane-1 VM. Press the TAB key to edit the kernel boot options. Once completed, press Enter to boot.

```

ip=192.168.60.241::192.168.60.1:255.255.255.0:::none
nameserver=192.168.60.240 coreos.inst.install_dev=/dev/sda
coreos.inst.image_url=http://192.168.60.240:8080/okd4/fcos.raw.xz
coreos.inst.ignition_url=http://192.168.60.240:8080/okd4/master.ign

```



You should see that the fcos.raw.xz image and signature are downloading:

```
Starting Network Manager...
Starting D-Bus System Message Bus...
[ OK ] Started D-Bus System Message Bus.
[ OK ] Started Network Manager.
[ OK ] Reached target Network.
Starting Network Manager Wait Online...
Starting Hostname Service...
[ OK ] Started Hostname Service.
[ OK ] Listening on Load/Save RF Kill Switch Status /dev/rfkill Watch.
Starting Network Manager Script Dispatcher Service...
[ OK ] Started Network Manager Script Dispatcher Service.
[ OK ] Finished Network Manager Wait Online.
[ OK ] Reached target Network is Online.
Starting CoreOS Installer...
#####
11.552419] coreos-installer-service[977]: coreos-installer install /dev/sda --ignition /tmp/coreos-installer-DjUysn --firstboot-args rd.neednet=1 ip=192.168.60.241::192.168.60.1:255.255.255.0::none nameserver=192.168.60.240 --image-url http://192.168.60.240:8080/okd4/fcos.raw.xz
11.582360] coreos-installer-service[994]: Downloading image from http://192.168.60.240:8080/okd4/fcos.raw.xz
11.583602] coreos-installer-service[994]: Downloading signature from http://192.168.60.240:8080/okd4/fcos.raw.xz.sig
12.650966] coreos-installer-service[994]: Read disk 29.5 MiB/496.9 MiB (5%)
13.651162] coreos-installer-service[994]: Read disk 64.2 MiB/496.9 MiB (12%)
14.653976] coreos-installer-service[994]: Read disk 74.3 MiB/496.9 MiB (14%)
15.702414] coreos-installer-service[994]: Read disk 74.5 MiB/496.9 MiB (14%)
16.765728] coreos-installer-service[994]: Read disk 74.7 MiB/496.9 MiB (15%)
18.301552] coreos-installer-service[994]: Read disk 74.7 MiB/496.9 MiB (15%)
19.599644] coreos-installer-service[994]: Read disk 75.5 MiB/496.9 MiB (15%)
20.628859] coreos-installer-service[994]: Read disk 76.4 MiB/496.9 MiB (15%)
22.073134] coreos-installer-service[994]: Read disk 76.4 MiB/496.9 MiB (15%)
23.491240] coreos-installer-service[994]: Read disk 76.4 MiB/496.9 MiB (15%)
25.153267] coreos-installer-service[994]: Read disk 76.5 MiB/496.9 MiB (15%)
26.791653] coreos-installer-service[994]: Read disk 76.5 MiB/496.9 MiB (15%)
28.389935] coreos-installer-service[994]: Read disk 76.5 MiB/496.9 MiB (15%)
29.893554] coreos-installer-service[994]: Read disk 76.5 MiB/496.9 MiB (15%)
```

Monitor the bootstrap installation:

You can monitor the bootstrap process from the okd4-services node:

```
openshift-install --dir=install_dir/ wait-for bootstrap-complete --log-level=info
```

```
[crobinson@okd4-services ~]$ openshift-install --dir=install_dir/ wait-for bootstrap-complete --log-level=debug
DEBUG OpenShift Installer 4.5.0-0.okd-2020-07-12-134038-rc
DEBUG Built from commit 290e3b1de6096ecef2133fb071ff3a71c9c78594
INFO Waiting up to 20m0s for the Kubernetes API at https://api.lab.okd.local:6443...
INFO API v1.18.3 up
INFO Waiting up to 40m0s for bootstrapping to complete...
DEBUG Bootstrap status: complete
INFO It is now safe to remove the bootstrap resources
DEBUG Time elapsed per stage:
DEBUG Bootstrap Complete: 3m13s
INFO Time elapsed: 3m13s
[crobinson@okd4-services ~]$
```

Once the bootstrap process is complete, which can take upwards of 30 minutes, you can shutdown your bootstrap node. Now is a good time to edit the /etc/haproxy/haproxy.cfg, comment out the bootstrap node, and reload the haproxy service.

```
sudo sed -i '/ okd4-bootstrap /s/^/#/' /etc/haproxy/haproxy.cfg
sudo systemctl reload haproxy
```

Login to the cluster and check the status:

Now that the control-plane node is online, you should be able to login with the oc client. Use the following commands to log in and check the status of your cluster:

```
export KUBECONFIG=~/.install_dir/auth/kubeconfig
oc whoami
oc get nodes
```

```
[crobinson@okd4-services ~]$ export KUBECONFIG=~/.install_dir/auth/kubeconfig
[crobinson@okd4-services ~]$ oc whoami
system:admin
[crobinson@okd4-services ~]$ oc get nodes
NAME                  STATUS   ROLES      AGE      VERSION
okd4-control-plane-1.lab.okd.local   Ready    master,worker   7m50s   v1.18.3
```

Check the status of the cluster operators.

```
oc get clusteroperators
```

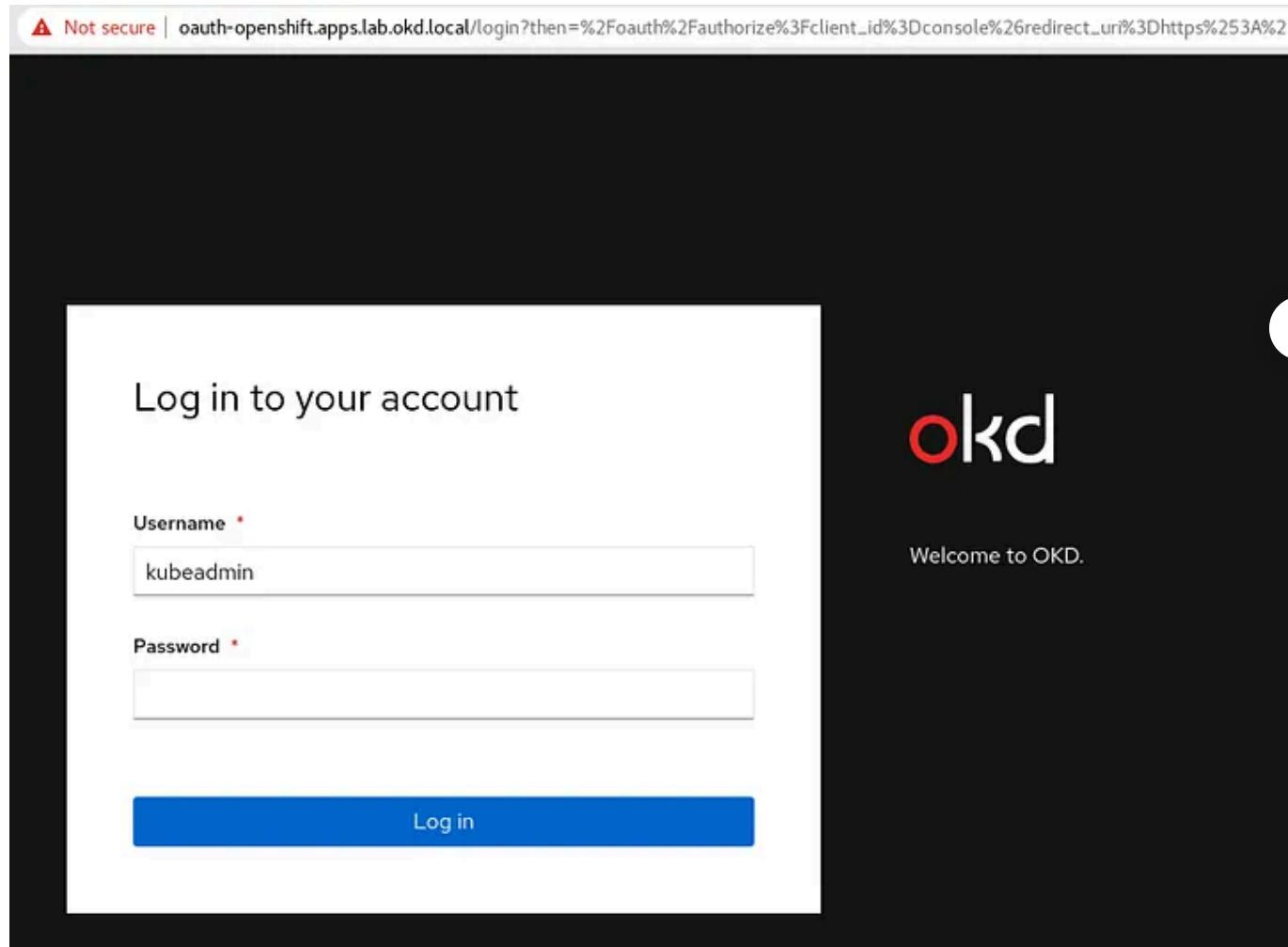
```
[crobinson@okd4-services ~]$ oc get clusteroperators
NAME          VERSION      AVAILABLE PROGRESSING DEGRADED SINCE
authentication           Unknown  Unknown  True   6m32s
cloud-credential         True   False   False
cluster-autoscaler       True   False   False  10m
config-operator          True   False   False
console                 4.5.0-0.okd-2020-07-12-134038-rc False  True   False  5m7s
csi-snapshot-controller 4.5.0-0.okd-2020-07-12-134038-rc True   False   False  4m56s
dns                      4.5.0-0.okd-2020-07-12-134038-rc True   False   False  4m52s
etcd                    4.5.0-0.okd-2020-07-12-134038-rc True   False   False
image-registry           True   True   False
ingress                 True   True   False
insights                True   True   False  4m34s
kube-apiserver          4.5.0-0.okd-2020-07-12-134038-rc True   True   False  4m31s
kube-controller-manager 4.5.0-0.okd-2020-07-12-134038-rc True   True   False
kube-scheduler           4.5.0-0.okd-2020-07-12-134038-rc True   True   False  3m49s
kube-storage-version-migrator 4.5.0-0.okd-2020-07-12-134038-rc True   False   False  4m14s
machine-api              True   False   False
machine approver         4.5.0-0.okd-2020-07-12-134038-rc True   False   False  4m10s
machine-config            4.5.0-0.okd-2020-07-12-134038-rc True   False   False  4m50s
marketplace               True   True   False
monitoring                True   False   False  7m10s
network                  4.5.0-0.okd-2020-07-12-134038-rc True   False   False
node-tuning               4.5.0-0.okd-2020-07-12-134038-rc True   False   False  6m25s
openshift-apiserver      4.5.0-0.okd-2020-07-12-134038-rc False  False   False  6m31s
openshift-controller-manager 4.5.0-0.okd-2020-07-12-134038-rc False  True   False  6m37s
openshift-samples         True   True   False
operator-lifecycle-manager 4.5.0-0.okd-2020-07-12-134038-rc True   True   False  5m27s
operator-lifecycle-manager-catalog 4.5.0-0.okd-2020-07-12-134038-rc True   True   False  5m28s
operator-lifecycle-manager-packageserver 4.5.0-0.okd-2020-07-12-134038-rc False  True   False  5m27s
service-ca                True   False   False
storage                  4.5.0-0.okd-2020-07-12-134038-rc True   False   False  6m18s
[crobinson@okd4-services ~]$ A=
```

Once the console operator is available login to the web console. Get your kubeadmin password from the install_dir/auth folder:

```
cat install_dir/auth/kubeadmin-password
```

```
crobinson@okd4-services:~  
File Edit View Search Terminal Help  
[crobinson@okd4-services ~]$ cat install_dir/auth/kubeadmin-password  
xmLBi-xUC26-n3EU5-Mrtxb[crobinson@okd4-services ~]$ 
```

Open your web browser to <https://console-openshift-console.apps.lab.okd.local/> and login as kubeadmin with the password from above:



The cluster status may still say upgrading, and it continues to finish the installation.

You are logged in as a temporary administrative user. Update the cluster OAuth configuration to allow others to log in.

Overview

Cluster

Details	View settings	Status	View alerts	Activity	View events
Cluster API Address https://api.lab.okd.local:6443		✓ Cluster	✓ Control Plane	✓ Operators	Ongoing There are no ongoing activities.
Cluster ID 6cc2c777-28f2-4e72-bda3-c3e5b09e53d4					Recent Events Pause
Provider None					11:04 openshift-config...
OpenShift Version 4.5.0-0.okd-2020-07-12-134038-rc					11:05 Started container ...
Update Channel stable-4					11:05 Created container...

Cluster Utilization

1Hour

Resource	Usage	11:00	11:02	11:04
CPU	2.01 of 8	8	6	4
Memory	6.36 GiB of 15.63 GiB	8 GiB	6 GiB	4 GiB

Activity

View events

Ongoing

Recent Events Pause

- 11:04 openshift-config...
- 11:05 Started container ...
- 11:05 Created container...
- 11:05 Container image "...
- 11:05 Add eth0 [10.128.0...
- 11:05 Created Pod/revis...
- 11:05 Uncaught event ConfirmM

Persistent Storage:

We need to create some persistent storage for our registry before we can complete this project. Let's configure our okd4-services VM as an NFS server and use it for persistent storage.

Login to your okd4-services VM and begin to set up an NFS server. The following commands install the necessary packages, enable services, and configure file and folder permissions.

```
sudo dnf install -y nfs-utils
sudo systemctl enable nfs-server rpcbind
sudo systemctl start nfs-server rpcbind
sudo mkdir -p /var/nfsshare/registry
sudo chmod -R 777 /var/nfsshare
sudo chown -R nobody:nobody /var/nfsshare
```

Create an NFS Export

Add this line in the new /etc/exports file “/var/nfsshare
192.168.60.0/24(rw,sync,no_root_squash,no_all_squash,no_wdelay)”

```
echo '/var/nfsshare  
192.168.60.0/24(rw,sync,no_root_squash,no_all_squash,no_wdelay)' |  
sudo tee /etc/exports
```

```
/var/nfsshare 192.168.60.0/24(rw,sync,no_root_squash,no_all_squash,no_wdelay)  
~  
~  
~
```

Restart the nfs-server service and add firewall rules:

```
sudo setsebool -P nfs_export_all_rw 1  
sudo systemctl restart nfs-server  
sudo firewall-cmd --permanent --zone=public --add-service mountd  
sudo firewall-cmd --permanent --zone=public --add-service rpc-bind  
sudo firewall-cmd --permanent --zone=public --add-service nfs  
sudo firewall-cmd --reload
```

Registry configuration:

Create a persistent volume on the NFS share. Use the registry_py.yaml in okd4_files folder from the git repo:

```
oc create -f okd4_files/registry_pv.yaml  
oc get pv
```

```
[crobinson@okd4-services ~]$ oc get pv  
No resources found  
[crobinson@okd4-services ~]$ oc create -f okd4_files/registry_pv.yaml  
persistentvolume/registry-pv created  
[crobinson@okd4-services ~]$ oc get pv  
NAME      CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS     CLAIM   STORAGECLASS   REASON   AGE  
registry-pv  100Gi      RWX           Retain          Available     
[crobinson@okd4-services ~]$ █
```

Edit the image-registry operator:

```
oc edit configs.imageregistry.operator.openshift.io
```

Change the managementState: from Removed to Managed. Under storage: add the pvc: and claim: blank to attach the PV and save your changes automatically:

```
managementState: Managed  
storage:  
  pvc:  
    claim:
```

```

File Edit View Search Terminal Help
f:storage: {}
f:storageManaged: {}
manager: cluster-image-registry-operator
operation: Update
time: "2020-07-07T20:29:00Z"
name: cluster
resourceVersion: "28815"
selfLink: /apis/imageregistry.operator.openshift.io/v1/configs/cluster
uid: d04a80b4-dfd4-4f6b-82db-07cf5c7cc136
spec:
  httpSecret: 6dad3c822fe59312785dafa707f3192d30de946cf21b3ac3da9359
  logging: 2
  managementState: Managed
  proxy: {}
  replicas: 1
  requests:
    read:
      maxWaitInQueue: 0s
    write:
      maxWaitInQueue: 0s
  rolloutStrategy: RollingUpdate
  storage:
    pvc:
      claim:
status:
  conditions:

```

Check your persistent volume, and it should now be claimed:

```
oc get pv
```

```
[crobinson@okd4-services ~]$ oc get pv
NAME          CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM
           STORAGECLASS
registry-pv   100Gi      RWX            Retain        Bound   openshift-image-registry/i
mage-registry-storage
[crobinson@okd4-services ~]$
```

Check the export size, and it should be zero. In the next section, we will push to the registry, and the file size should not be zero.

```
du -sh /var/nfsshare/registry
```

```
[crobinson@okd4-services ~]$ du -sh /var/nfsshare/registry/
0      /var/nfsshare/registry/
[crobinson@okd4-services ~]$ █
```

In the next section, we will create a WordPress project and push it to the registry. After the push, the NFS export should show 200+ MB.

Create WordPress Project:

Create a new OKD project.

```
oc new-project wordpress-test
```

```
[crobinson@okd4-services ~]$ oc new-project wordpress-test
Now using project "wordpress-test" on server "https://api.lab.okd.local:6443".
You can add applications to this project with the 'new-app' command. For example, try:
  oc new-app ruby~https://github.com/sclorg/ruby-ex.git
to build a new example application in Ruby. Or use kubectl to deploy a simple Kubernetes application:
  kubectl create deployment hello-node --image=gcr.io/hello-minikube-zero-install/hello-node
[crobinson@okd4-services ~]$ █
```

Create a new app using the centos php73 s2i image from docker hub and use the WordPress GitHub repo for the source. Expose the service to create a route.

```
oc new-app centos/php-73-
centos7~https://github.com/WordPress/WordPress.git
oc expose svc/wordpress
```

```
[crobinson@okd4-services ~]$ oc new-app centos/php-73-centos7~https://github.com/WordPress/WordPress.git

--> Found container image 83d5957 (2 weeks old) from Docker Hub for "centos/php-73-centos7"

  Apache 2.4 with PHP 7.3
  -----
    PHP 7.3 available as container is a base platform for building and running various PHP 7
.3 applications and frameworks. PHP is an HTML-embedded scripting language. PHP attempts to
make it easy for developers to write dynamically generated web pages. PHP also offers built-
in database integration for several commercial and non-commercial database management system
s, so writing a database-enabled webpage with PHP is fairly simple. The most common use of P
HP coding is probably as a replacement for CGI scripts.

  Tags: builder, php, php73, rh-php73

    * An image stream tag will be created as "php-73-centos7:latest" that will track the sou
rce image
    * A source build using source code from https://github.com/WordPress/WordPress.git will
be created
      * The resulting image will be pushed to image stream tag "wordpress:latest"
      * Every time "php-73-centos7:latest" changes a new build will be triggered

--> Creating resources ...
imagestream.image.openshift.io "php-73-centos7" created
imagestream.image.openshift.io "wordpress" created
buildconfig.build.openshift.io "wordpress" created
deployment.apps "wordpress" created
service "wordpress" created
--> Success
Build scheduled, use 'oc logs -f bc/wordpress' to track its progress.
Application is not exposed. You can expose services to the outside world by executing on
e or more of the commands below:
  'oc expose svc/wordpress'
  Run 'oc status' to view your app.
[crobinson@okd4-services ~]$ oc expose svc/wordpress
route.route.openshift.io/wordpress exposed
[crobinson@okd4-services ~]$
```

Create a new app using the centos7 MariaDB image with some environment variables:

```
oc new-app centos/mariadb-103-centos7 --name mariadb --env
MYSQL_DATABASE=wordpress --env MYSQL_USER=wordpress --env
MYSQL_PASSWORD=wordpress
```

```
[crobinson@okd4-services ~]$ oc new-app centos/mariadb-103-centos7 --name mariadb --env MYSQL_DATABASE=wordpress --env MYSQL_USER=wordpress --env MYSQL_PASSWORD=wordpress
--> Found container image 000778a (4 days old) from Docker Hub for "centos/mariadb-103-centos7"

  MariaDB 10.3
  -----
  MariaDB is a multi-user, multi-threaded SQL database server. The container image provides a containerized packaging of the MariaDB mysqld daemon and client application. The mysqld server daemon accepts connections from clients and provides access to content from MariaDB databases on behalf of the clients.

  Tags: database, mysql, mariadb, mariadb103, rh-mariadb103, galera
  * An image stream tag will be created as "mariadb:latest" that will track this image

--> Creating resources ...
imagestream.image.openshift.io "mariadb" created
deployment.apps "mariadb" created
service "mariadb" created
--> Success
  Application is not exposed. You can expose services to the outside world by executing one or more of the commands below:
    'oc expose svc/mariadb'
  Run 'oc status' to view your app.
[crobinson@okd4-services ~]$ █
```

Open the OpenShift console and browse to the WordPress-test project. Once the WordPress image is built and ready, it will be dark blue like the MariaDB instance shown here:

The screenshot shows the OKD 4.5 Single Node Cluster topology interface. The left sidebar has a 'Topology' section selected. The main area displays two service icons: 'wordpress' and 'mariadb'. Both icons are blue circles with white reload symbols. Below each icon is a small green checkmark and a blue circular badge with a white 'D'. The 'wordpress' badge also contains the text 'wordpress'. At the bottom of the main area are several small navigation buttons: a magnifying glass, a double arrow, a plus sign, a minus sign, and two numbered buttons '1' and '2'.

Click on the WordPress object and click on the route to open it in your web browser:

The screenshot shows the OKD 4.5 Single Node Cluster interface. The left sidebar has a 'Developer' dropdown, '+Add' button, and a 'Topology' section which is currently selected. Other options include 'Monitoring', 'Search', 'Builds', 'Helm', 'Project', 'Config Maps', and 'Secrets'. The main content area displays the following information:

- Project:** wordpress-test
- Application:** all applications
- Display Options:** Find by name... (with a search icon)
- Resources:** Details, Resources (selected), Monitoring
- Pods:** A single pod named 'wordpress-674d478fd9-4c7ej' is listed as 'Running'.
- Builds:** A build named 'wordpress' is listed, with a 'Start Build' button and a note that it is complete (3 minutes ago). There is also a 'View logs' link.
- Services:** A service named 'wordpress' is listed, with two mappings: Service port: 8080-tcp → Pod Port: 8080 and Service port: 8443-tcp → Pod Port: 8443.
- Routes:** A route named 'wordpress' is listed, with a location of <http://wordpress-wordpress-test.apps.lab.okd.local>.

You should see the WordPress setup config, click Let's Go.

The screenshot shows a web browser window with the following tabs:

- Topology · OKD
- WordPress · Setup Config
- okd4-pfsense.okd.local
- Log in - VMware ESXi

The address bar indicates the URL is `wordpress-wordpress-test.apps.lab.okd.local/wp-admin/setup-config.php`.

The main content area displays the WordPress logo at the top, followed by the following text:

Welcome to WordPress. Before getting started, we need some information on the database. You will need to know the following items before proceeding.

1. Database name
2. Database username
3. Database password
4. Database host
5. Table prefix (if you want to run more than one WordPress in a single database)

We're going to use this information to create a `wp-config.php` file. If for any reason this automatic file creation doesn't work, don't worry. All this does is fill in the database information to a configuration file. You may also simply open `wp-config-sample.php` in a text editor, fill in your information, and save it as `wp-config.php`. Need more help? [We got it.](#)

In all likelihood, these items were supplied to you by your Web Host. If you don't have this information, then you will need to contact them before you can continue. If you're all ready...

[Let's go!](#)

Fill in the database, username, password, and database host as pictured and run the installation:

Below you should enter your database connection details. If you're not sure about these, contact your host.

Database Name	<input type="text" value="wordpress"/>	The name of the database you want to use with WordPress.
Username	<input type="text" value="wordpress"/>	Your database username.
Password	<input type="text" value="wordpress"/>	Your database password.
Database Host	<input type="text" value="mariadb"/>	You should be able to get this info from your web host, if <code>localhost</code> doesn't work.
Table Prefix	<input type="text" value="wp_"/>	If you want to run multiple WordPress installations in a single database, change this.

Submit

Fill out the welcome information and click Install WordPress.

Welcome

Welcome to the famous five-minute WordPress installation process! Just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform in the world.

Information needed

Please provide the following information. Don't worry, you can always change these settings later.

Site Title test

Username test

Usernames can have only alphanumeric characters, spaces, underscores, hyphens, periods, and the @ symbol.

Password test Hide Very weak

Confirm Password Confirm use of weak password

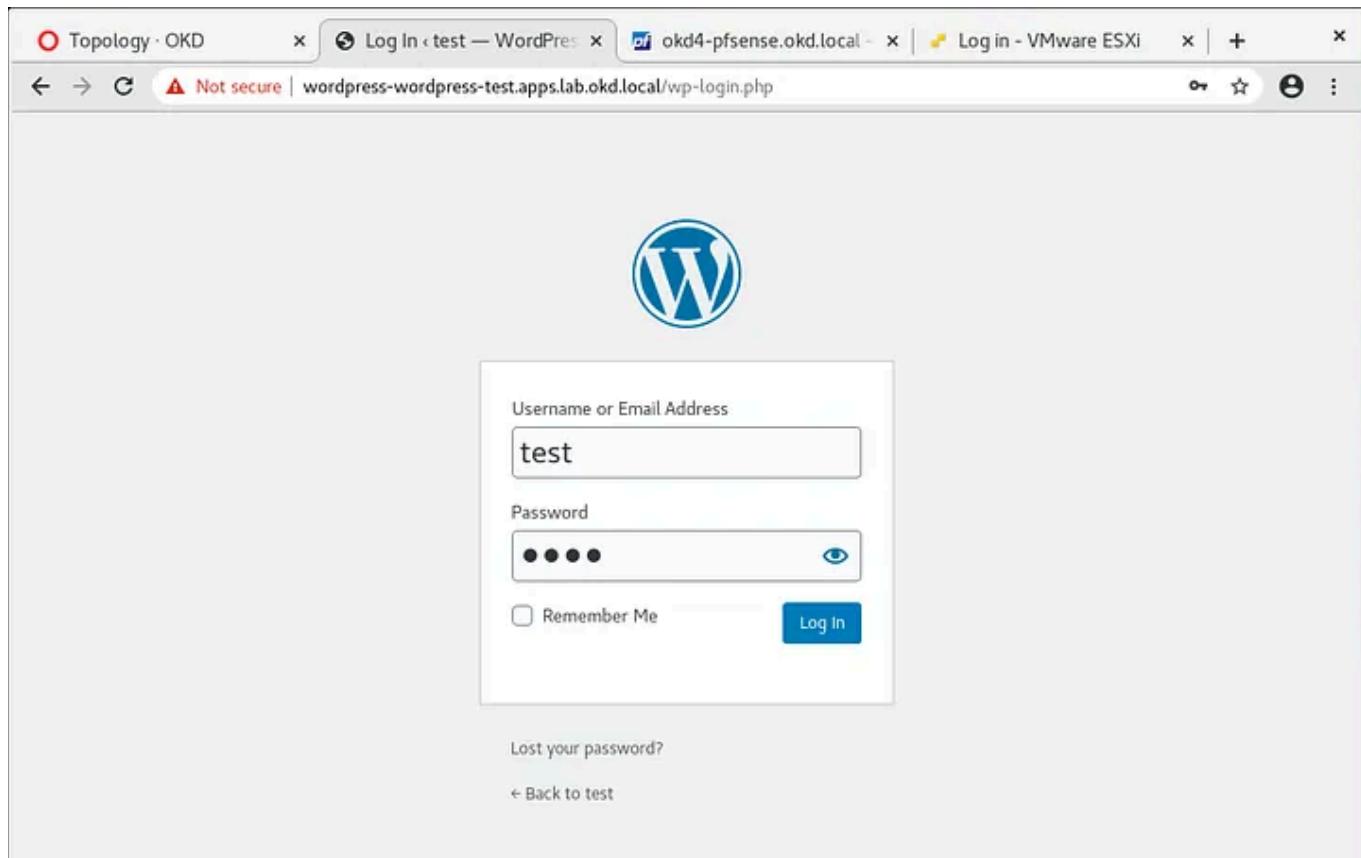
Your Email test@test.com

Double-check your email address before continuing.

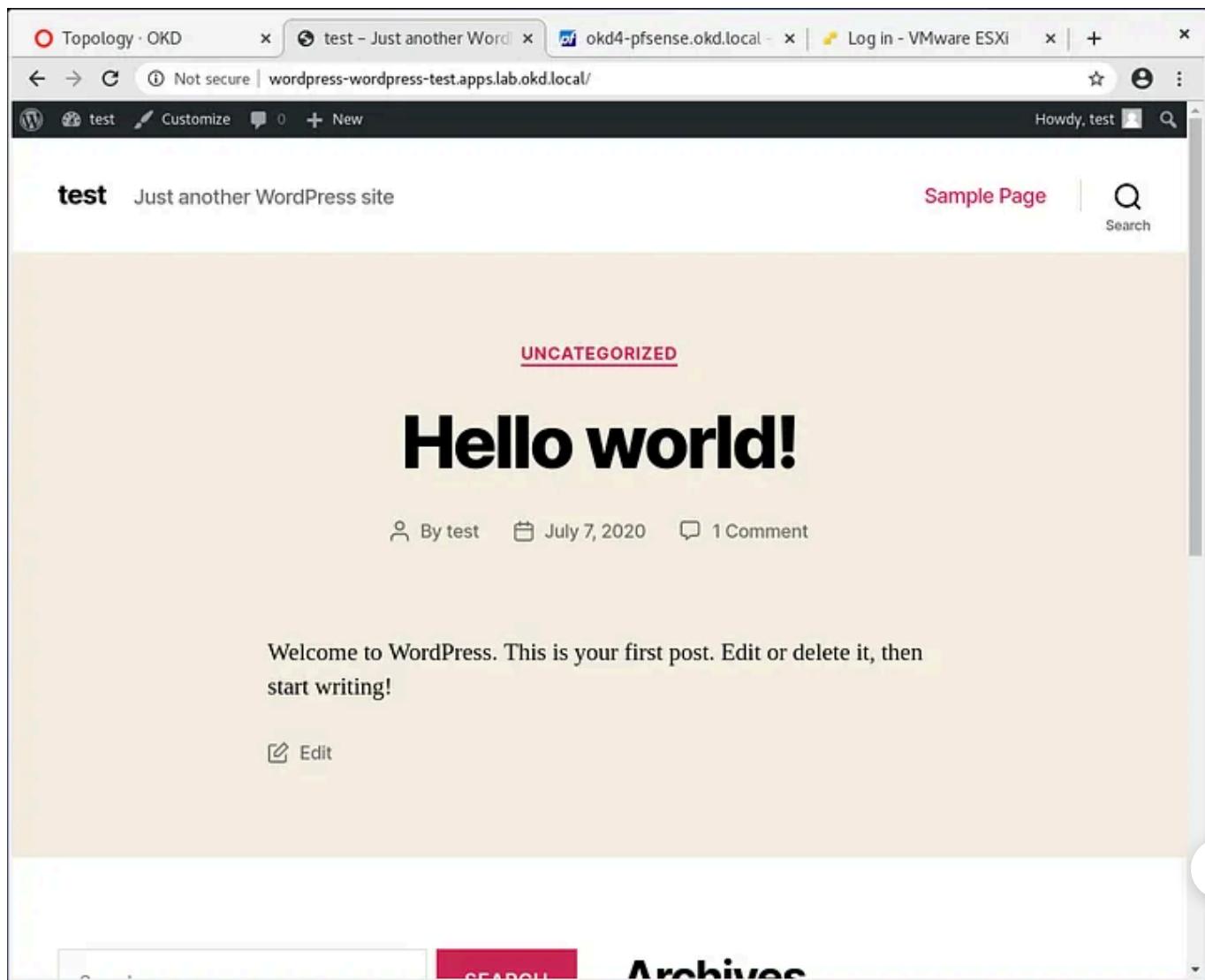
Search engine visibility Discourage search engines from indexing this site
It is up to search engines to honor this request.

Install WordPress

Log in, and you should have a working WordPress installation:



The screenshot shows the WordPress dashboard for a site named 'test'. The top navigation bar includes tabs for 'Topology - OKD', 'Dashboard < test — Wordc', 'okd4-pfsense.okd.local - Log in - VMware ESXi', and 'Howdy, test'. The dashboard features a 'Welcome to WordPress!' message and a 'Get Started' section with a 'Customize Your Site' button. It also displays 'Site Health Status' (No information yet...), 'At a Glance' (1 Post, 1 Page, 1 Comment), and 'Activity' sections. On the right, there's a 'Quick Draft' area for writing posts and a 'WordPress Events and News' section.



Check the size of your NFS export on the okd4-services VM. It should be around 300MB in size.

```
du -sh /var/nfsshare/registry/
```

```
[crobinson@okd4-services ~]$ du -sh /var/nfsshare/registry/
272M    /var/nfsshare/registry/
[crobinson@okd4-services ~]$ █
```

You have just verified your persistent volume is working.

HTPasswd Setup:

The kubeadmin is a temporary user. The easiest way to set up a local user is with htpasswd.

```
cd  
cd okd4_files  
htpasswd -c -B -b users.htpasswd testuser testpassword
```

```
[crobinson@okd4-services okd4_files]$ cd  
[crobinson@okd4-services ~]$ cd okd4_files/  
[crobinson@okd4-services okd4_files]$ htpasswd -c -B -b users.htpasswd testuser testpassword  
Adding password for user testuser  
[crobinson@okd4-services okd4_files]$ █
```

Create a secret in the openshift-config project using the users.htpasswd file you generated:

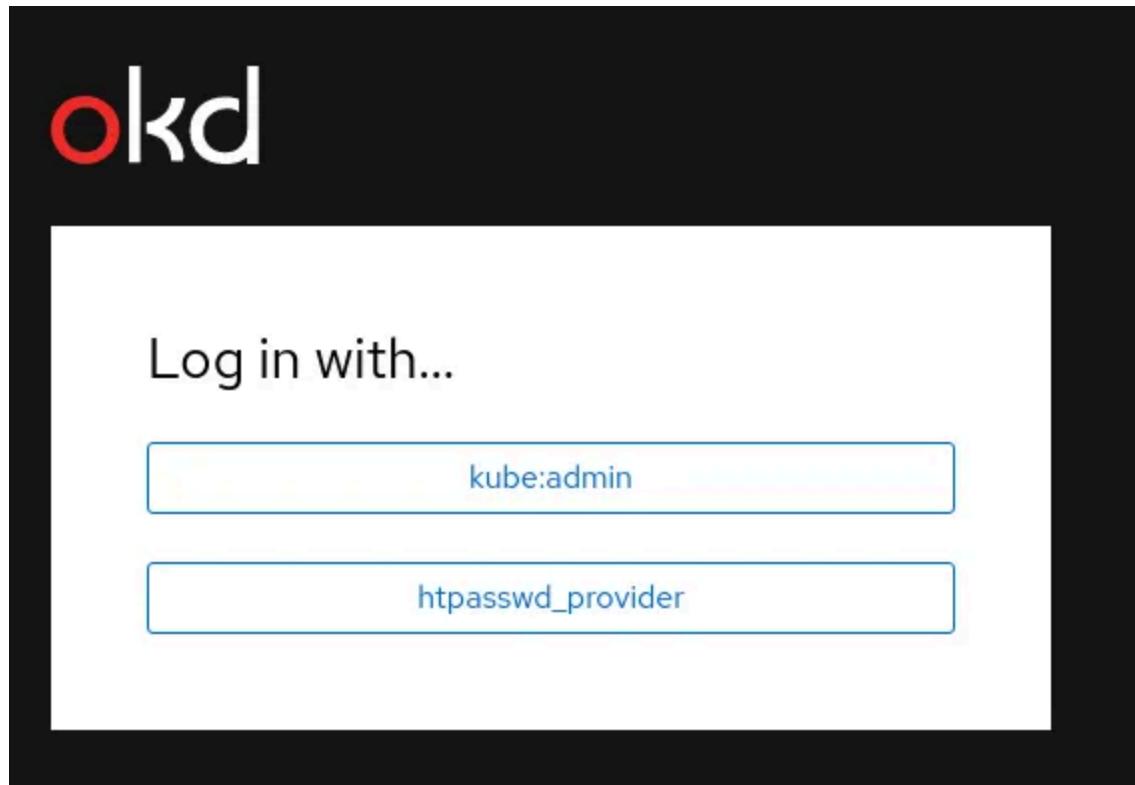
```
oc create secret generic htpass-secret --from=  
file=htpasswd=users.htpasswd -n openshift-config
```

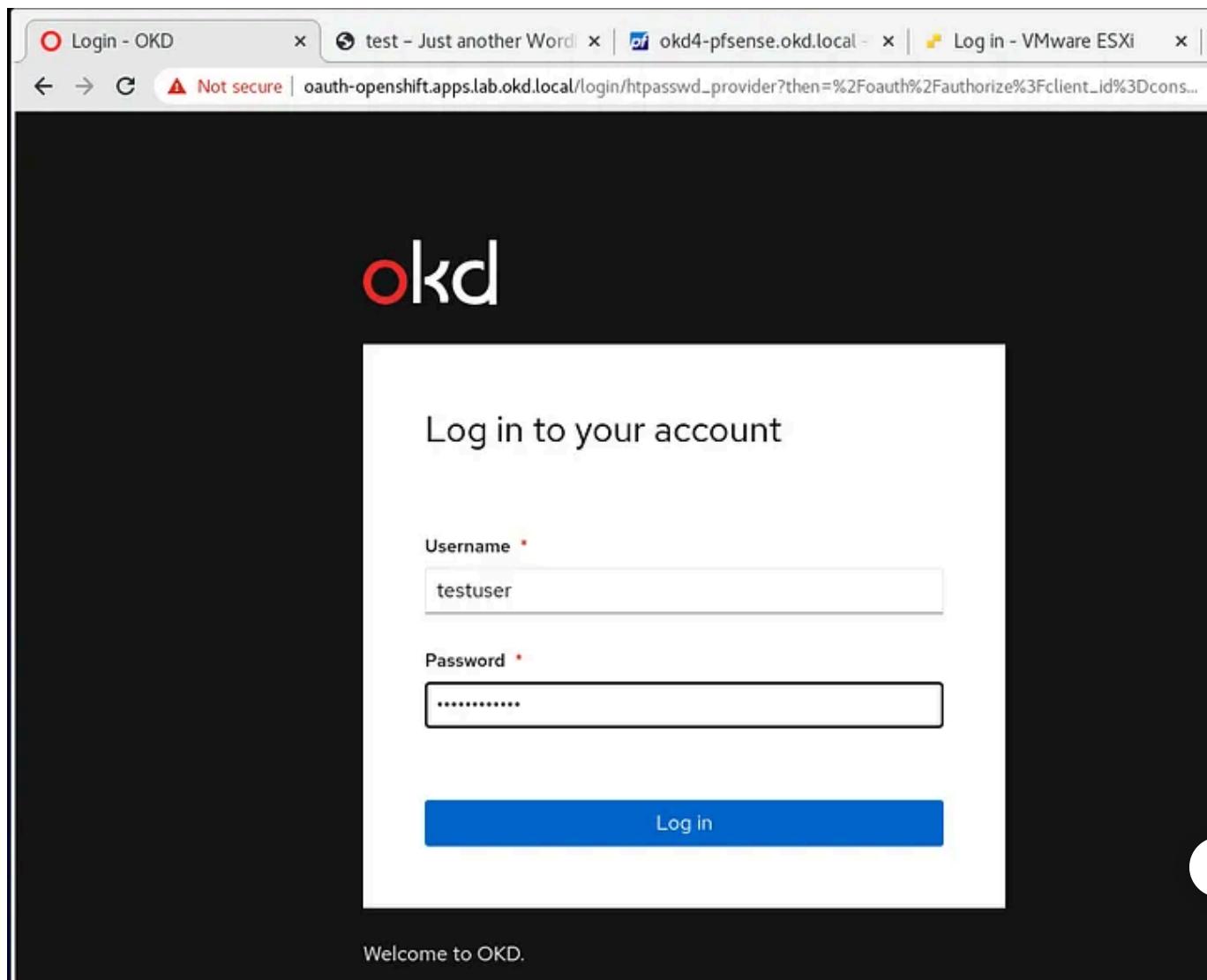
Add the identity provider.

```
oc apply -f htpasswd_provider.yaml
```

```
[crobinson@okd4-services okd4_files]$ oc apply -f htpasswd_provider.yaml
Warning: oc apply should be used on resource created by either oc create --save-config or oc
apply
oauth.config.openshift.io/cluster configured
[crobinson@okd4-services okd4_files]$
```

Logout of the OpenShift Console. Then select htpasswd_provider and login with testuser and testpassword credentials.





If you visit the Administrator page you should see no projects:

The screenshot shows the OpenShift OKD web interface. The left sidebar is dark-themed with white text, showing a navigation menu with 'Administrator' at the top, followed by 'Home' (which is underlined in blue), 'Projects', 'Search', 'Explore', 'Events', 'Operators', 'Workloads', 'Networking', 'Storage', 'Builds', 'User Management', and 'Administration'. The main content area has a light background. At the top right of the content area, there is a user dropdown labeled 'testuser'. Below the user dropdown, there is a 'Create Project' button. The main content area is titled 'Welcome to OpenShift' and contains text: 'OpenShift helps you quickly develop, host, and scale applications. To get started, create a project for your application.' It also includes links to 'OpenShift documentation' and 'Download the command-line tools'. At the bottom right of the content area, there is a link 'Create a new project'.

Give yourself cluster-admin access, and the projects should immediately populate:

```
oc adm policy add-cluster-role-to-user cluster-admin testuser
```

Your user should now have cluster-admin level access:

Name	Display Name	Status	Requester
PR default	No display name	Active	No requester
PR kube-node-lease	No display name	Active	No requester
PR kube-public	No display name	Active	No requester
PR kube-system	No display name	Active	No requester
PR openshift	No display name	Active	No requester
PR openshift-apiserver	No display name	Active	No requester
PR openshift-apiserver-operator	No display name	Active	No requester
PR openshift-authentication	No display name	Active	No requester
PR openshift-authentication-operator	No display name	Active	No requester
PR openshift-cloud-credential-operator	No display name	Active	No requester
PR openshift-cluster-machine approver	No display name	Active	No requester

Congrats! You have created an OKD Single Node Cluster!

Hopefully, you have created an OKD single node cluster and learned a few things along the way. At this point, you should have a decent basis to tinker with OKD and continue to learn.

Here are some resources available to help you along your journey:

To report issues, use the [OKD Github Repo: https://github.com/openshift/okd](https://github.com/openshift/okd)

For support check out the #openshift-users channel on [k8s Slack](#)

The [OKD Working Group](#) meets bi-weekly to discuss the development and next steps. The meeting schedule and location are tracked in the [openshift/community repo](#).

Google group for okd-wg: <https://groups.google.com/forum/#!forum/okd-wg>

Openshift

Okd

Kubernetes

WordPress

Cluster

No rights reserved by the author. 



Written by Craig Robinson

517 Followers · Writer for The Startup

OpenShift Enthusiast

Follow



More from Craig Robinson and The Startup

The screenshot shows the WordPress dashboard. On the left, a sidebar lists navigation options like Home, Posts, Media, Pages, Comments, Appearance, Plugins, Users, Tools, and Settings. The main area displays a 'Welcome to WordPress!' message with links to 'Get Started' (Customize Your Site, Write your first blog post, Add an About page, Set up your homepage, View your site), 'Next Steps' (Manage widgets, Manage menus, Turn comments on or off, Learn more about getting started), and 'Site Health Status' (No information yet...). Below these are sections for 'Quick Draft' (Title, Content, What's on your mind?) and 'At a Glance'.



 Craig Robinson in ITNEXT

Guide: Installing an OKD 4.5 Cluster

Updated 7/7/2020

★ Jul 8, 2020 ⚡ 507 🎙 53

Bookmark  More 



 Greyson Ferguson  in The Startup

Countries You Can Move To Indefinitely Without a Visa

Why deal with all of that visa paperwork when you could just move without filling out a sing...

★ Jul 4 ⚡ 4.8K 🎙 56

Bookmark  More 

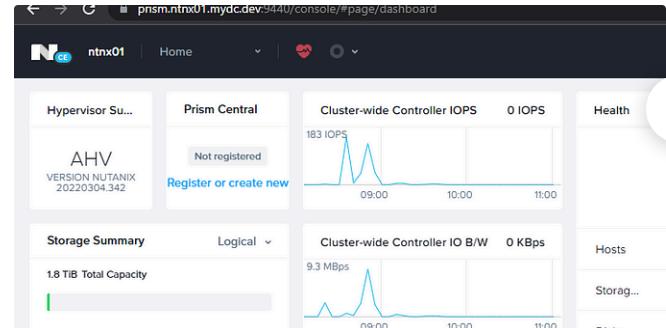
 Onyedikachukwu Czar in The Startup

I Asked Google Gemini To Come Up With Business Ideas Not Found...

How to use AI to iterate ideas and come up with what stands out.

★ Jul 12 ⚡ 2.1K 🎙 42

Bookmark  More 



 Craig Robinson

Installing Prism Central on Nutanix Community Edition 2.0

Installing Prism Central

★ Mar 7, 2023 ⚡ 2

Bookmark  More 

See all from Craig Robinson

See all from The Startup

Recommended from Medium

Software Development Engineer Seattle, WA Mar. 2020 – May 2021

- Developed Amazon checkout and payment services to handle traffic of 10 Million daily global transactions
- Integrated Iframes for credit cards and bank accounts to secure 80% of all consumer traffic and prevent CSRF, cross-site scripting, and cookie-jacking
- Led Your Transactions implementation for JavaScript front-end framework to showcase consumer transactions and reduce call center costs by \$25 Million
- Recovered Saudi Arabia checkout failure impacting 4000+ customers due to incorrect GET form redirection

Projects

NinjaPrep.io (React)

- Platform to offer coding problem practice with built in code editor and written + video solutions in React
- Utilized Nginx to reverse proxy IP address on Digital Ocean hosts
- Developed using Styled-Components for 95% CSS styling to ensure proper CSS scoping
- Implemented Docker and Seccomp to safely run user submitted code with < 2.2s runtime

HeatMap (JavaScript)

- Visualized Google Takeout location data of location history using Google Maps API and Google Maps heatmap code with React
- Included local file system storage to reliably handle 5mb of location history data
- Implemented Express to include routing between pages and jQuery to parse Google Map and implement heatmap overlay

 Alexander Nguyen in Level Up Coding

The resume that got a software engineer a \$300,000 job at Google.

1-page. Well-formatted.

Jun 1 16.1K 252



...



 Michelle Teheux in Minds Without Borders

We Could Learn a Lot About Sex From the Dutch

My Dutch relative's views shocked me, but I immediately realized she was right

Jul 18 25K 299



...

Lists



Natural Language Processing

1618 stories · 1187 saves



Samanta Writes in ILLUMINATION

12 Signs of a Highly Unintelligent Person According to Science

3. Nodding Your Head

★ Jul 19

8.8K

236



...



Unbecoming

10 Seconds That Ended My 20 Year Marriage

It's August in Northern Virginia, hot and humid. I still haven't showered from my...

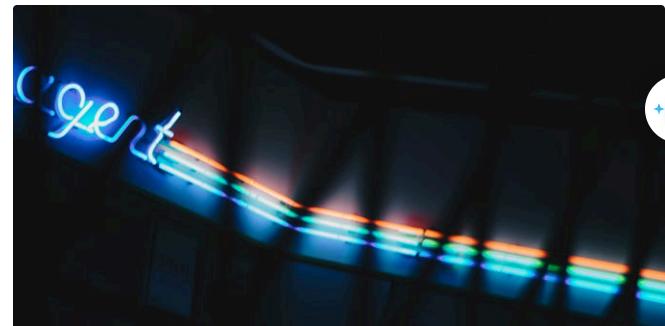
★ Feb 16, 2022

83K

1139



...



Derek Johnson

I'm Unemployed for Over Two Years (as a software engineer)

In 2022, I worked on a contract as a software engineer at Apple. Apple dissolved our entir...

★ Jun 1

5.7K

143



...



kapil rajyaguru

Agent-Based Red Hat OpenShift Cluster Install

With OpenShift 4.11, Red Hat introduced a new agent-based installer for OpenShift to the...

Apr 25



...

[See more recommendations](#)