

Guide: Installing an OKD 4.5 Cluster



Craig Robinson Following
Jul 7 · 16 min read ★

Updated 7/20/2020

OKD is the upstream and community-supported version of the Red Hat OpenShift Container Platform (OCP). OpenShift expands vanilla Kubernetes into an application platform designed for enterprise use at scale. Starting with the release of OpenShift 4, the default operating system is Red Hat CoreOS, which provides an immutable infrastructure and automated updates. Fedora CoreOS, like OKD, is the upstream version of Red Hat CoreOS.

The screenshot shows the OKD cluster dashboard with the following details:

- Cluster API Address:** https://api.lab.okd.local:6443
- Cluster ID:** fa14cd98-5cd4-4be1-9348-adc9a53a50a2
- Provider:** None
- OpenShift Version:** 4.5.0-0.okd-2020-06-29-110348-beta6
- Update Channel:** stable-4.5
- Cluster Inventory:** 5 Nodes
- Cluster Utilization:**
 - CPU:** 14.23 available, 5.77 of 20 usage over the last 1 hour.
 - Memory:** 64.46 GiB available, 13.69 GiB of 78.16 GiB usage over the last 1 hour.
 - Filesystem:** 2.24 TiB available, 180.1 GiB of 2.41 TiB usage over the last 1 hour.
- Status:** Cluster and Control Plane are green (healthy).
- Activity:** Recent events log showing various status updates and container operations.

You can experience OpenShift in your home lab by using the open-source upstream combination of OKD and FCOS (Fedora CoreOS) to build your cluster. Used hardware

for a home lab that can run an OKD cluster is relatively inexpensive these days (\$250–\$350), especially when compared to a cloud-hosted solution costing over \$250 per month.

The purpose of this guide is to help you successfully build an OKD 4.5 cluster so that you can develop and learn to use OpenShift and OKD.

NOTE: In this guide, I use VMWare ESXi as the hypervisor, but you could use Hyper-V, libvirt, Proxmox, VirtualBox, or even bare metal instead.

• • •

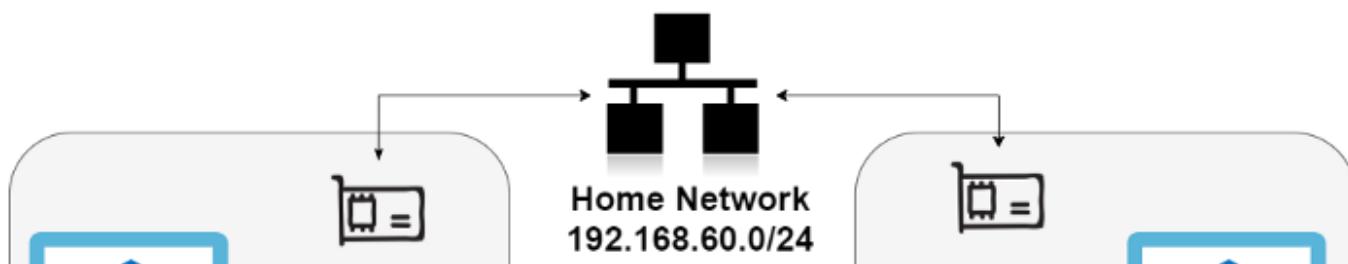
VM Overview:

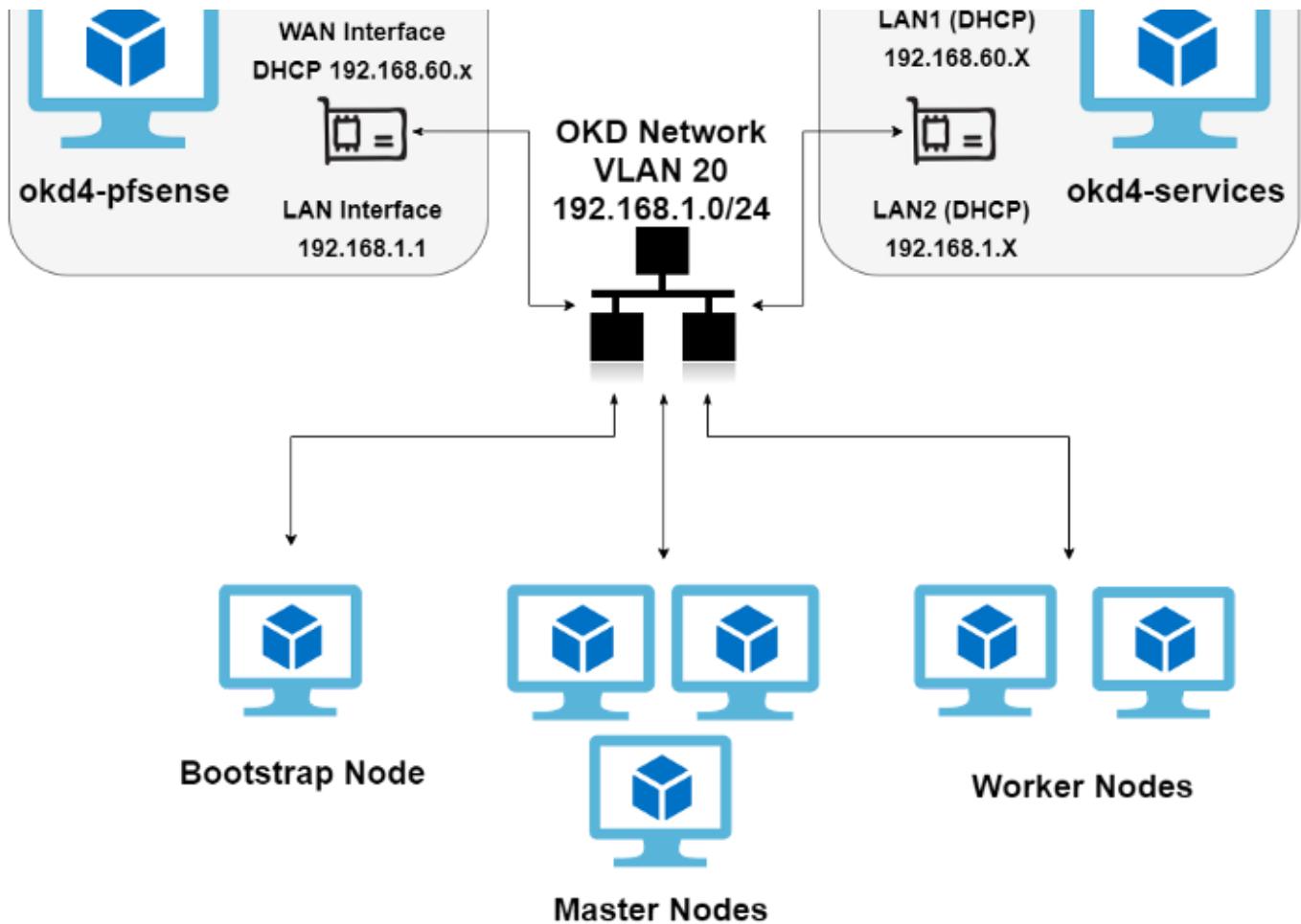
For my installation, I used an ESXi 6.5 host with 96GB of RAM and a separate VLAN configured for OKD. Here is a breakdown of the virtual machines:

Machine	Type	OS	vCPU	RAM	Storage	IP Address
okd4-bootstrap	Bootstrap	Fedora CoreOS	4	16	120	192.168.1.200
okd4-control-plane-1	Master	Fedora CoreOS	4	16	120	192.168.1.201
okd4-control-plane-2	Master	Fedora CoreOS	4	16	120	192.168.1.202
okd4-control-plane-3	Master	Fedora CoreOS	4	16	120	192.168.1.203
okd4-compute-1	Worker	Fedora CoreOS	4	16	120	192.168.1.204
okd4-compute-2	Worker	Fedora CoreOS	4	16	120	192.168.1.205
okd4-services	DNS/LB/Web/NFS	CentOS 8	4	4	100	192.168.1.210
okd4-pfsense	Router/DHCP	FreeBSD	1	1	8	192.168.1.1

• • •

Network Layout



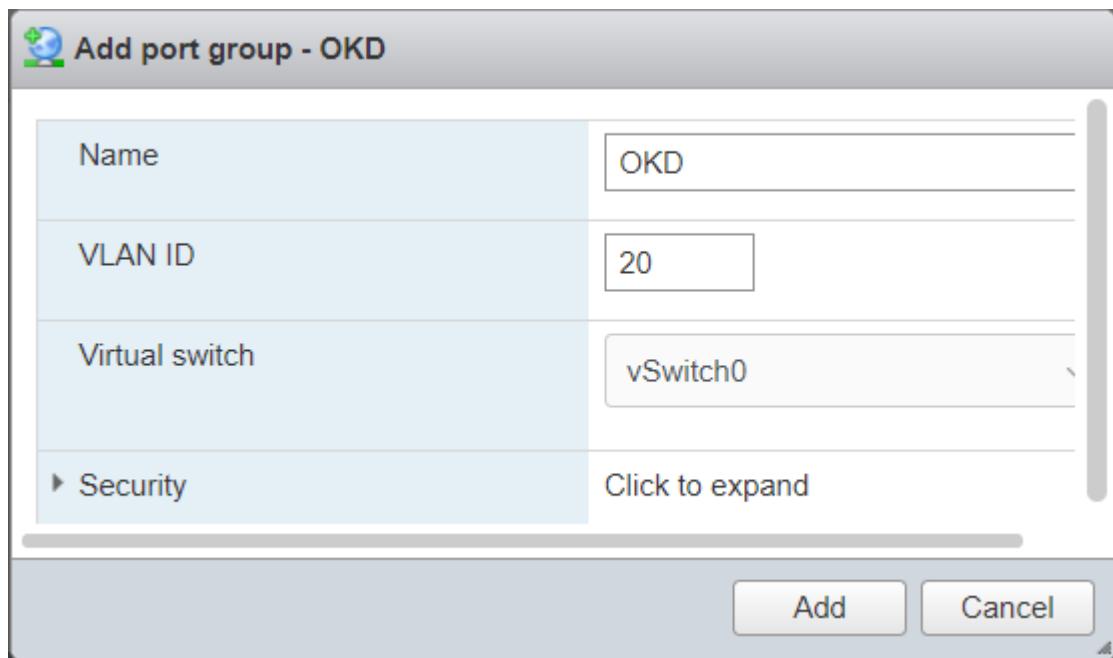


Create a new network in VMWare for OKD:

Login to your VMWare Host. Select Networking → Port Groups → Add port group. Setup an OKD network on an unused VLAN, in my instance, VLAN 20.



Name your Group and set your VLAN ID.



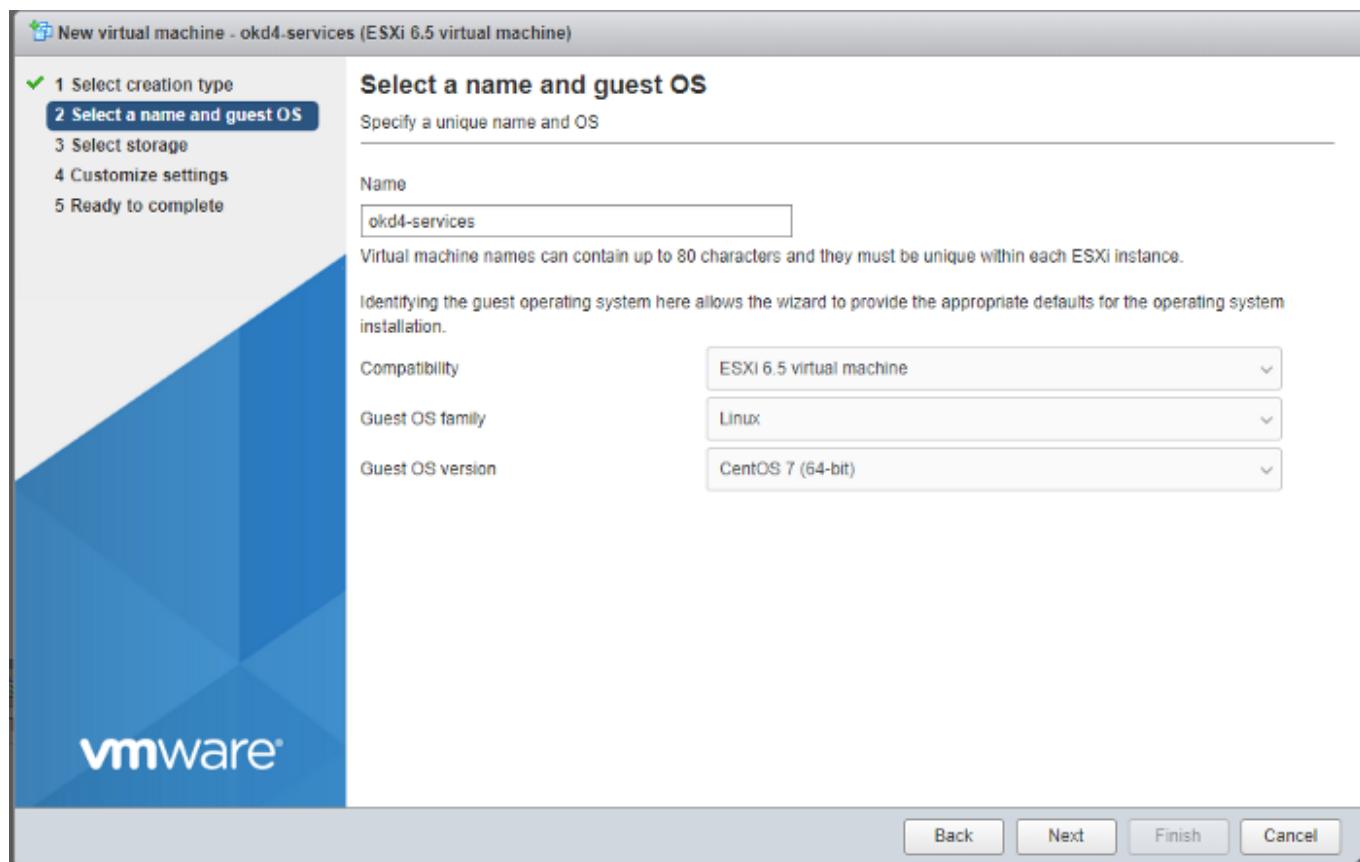
Create the okd4-services VM:

Download the CentOS 8 ISO DVD. Example: CentOS-8.2.2004-x86_64-dvd1.iso and upload it to your ESXi host datastore.

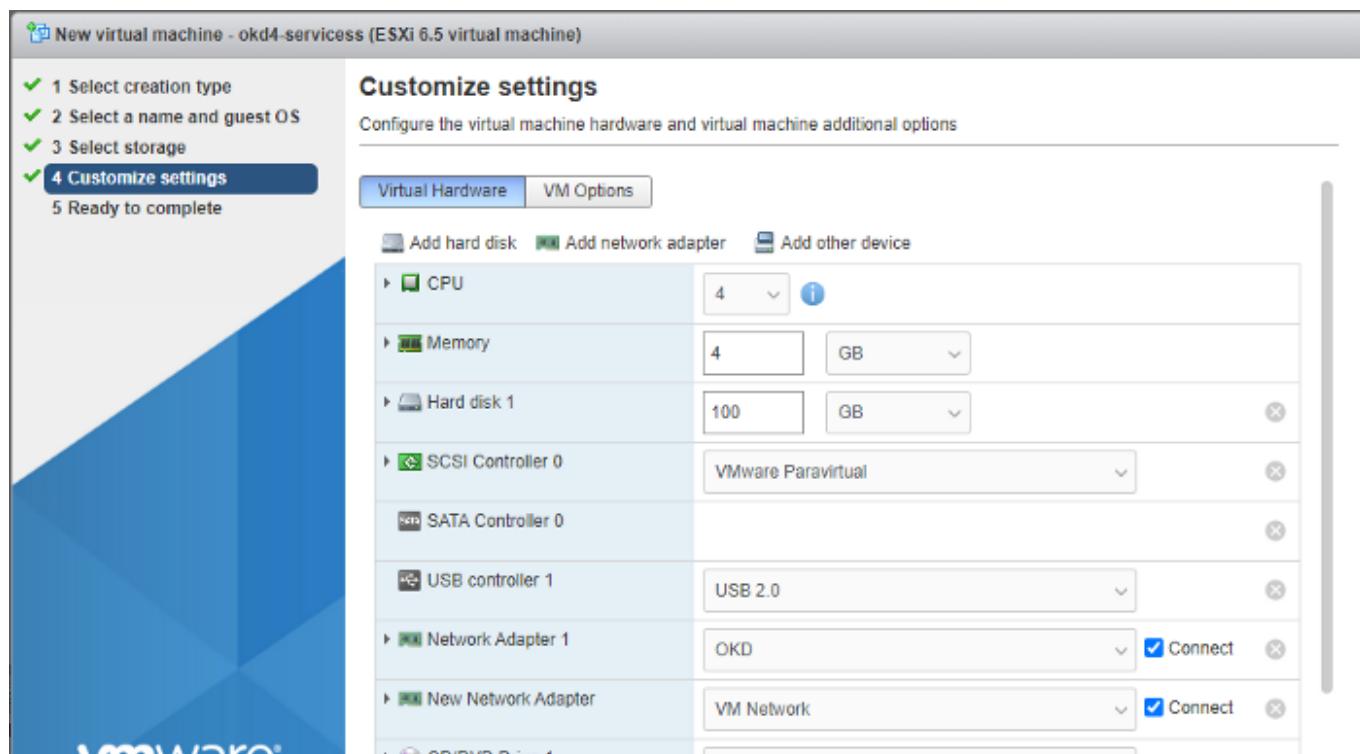
Index of /pub/centos/8.2.2004/isos/x86_64/

Name	Last Modified	Size	Type
..		-	Directory
CentOS-8.2.2004-x86_64-boot.iso	2020-Jun-08 17:26:48	624.0M	application/octet-stream
CentOS-8.2.2004-x86_64-boot.iso.manifest	2020-Jun-08 17:39:02	0.6K	application/octet-stream
CentOS-8.2.2004-x86_64-boot.torrent	2020-Jun-15 10:35:30	24.9K	application/x-bittorrent
<u>CentOS-8.2.2004-x86_64-dvd1.iso</u>	2020-Jun-08 18:11:38	7.6G	application/octet-stream
CentOS-8.2.2004-x86_64-dvd1.iso.manifest	2020-Jun-08 18:11:38	425.0K	application/octet-stream
CentOS-8.2.2004-x86_64-dvd1.torrent	2020-Jun-15 10:35:48	307.1K	application/x-bittorrent
CentOS-8.2.2004-x86_64-minimal.iso	2020-Jun-08 18:09:44	1.6G	application/octet-stream
CentOS-8.2.2004-x86_64-minimal.iso.manifest	2020-Jun-08 18:09:44	91.4K	application/octet-stream
CentOS-8.2.2004-x86_64-minimal.torrent	2020-Jun-15 10:35:51	64.5K	application/x-bittorrent
CHECKSUM	2020-Jun-12 18:42:13	0.4K	application/octet-stream
CHECKSUM.asc	2020-Jun-19 13:10:24	0.8K	text/plain

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

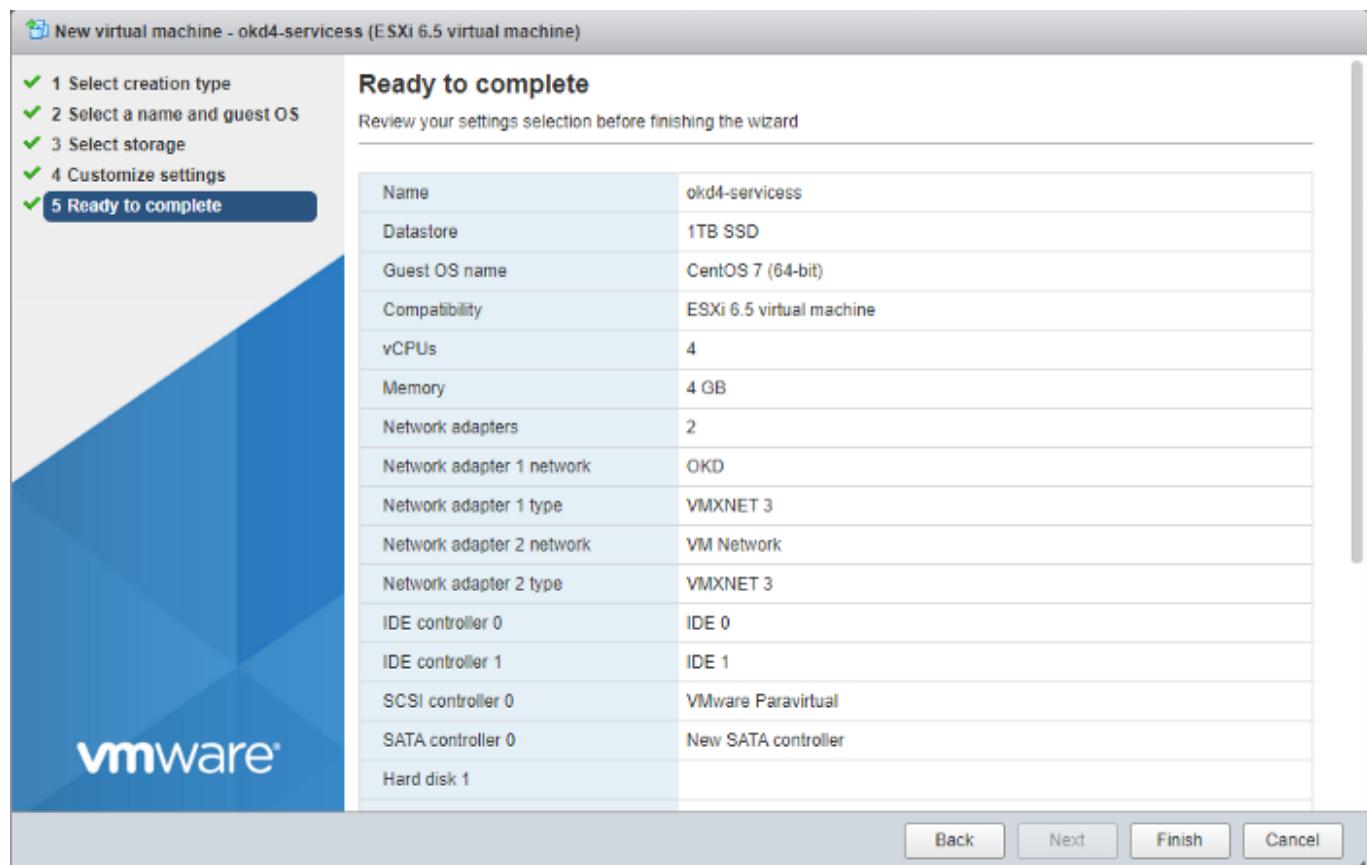


Select the Datastore and customize the settings to 4 vCPU, 4GB RAM, 100GB HD. Add a 2nd network adapter to the OKD network. Attached the CentOS ISO.





Review your settings and click Finish.



Install CentOS 8 on the okd4-services VM:

Run through the CentOS 8 installation.

I prefer to use the “Standard Partition” storage configuration without mounting storage for /home. On the “Installation Destination” page, click on Custom under Storage Configuration, then Done.



Device Selection

Select the device(s) you'd like to install to. They will be left untouched until you click on the main menu's "Begin Installation" button.

Local Standard Disks

100 GiB
 VMware Virtual disk
sda / 100 GiB free

Disks left unselected here will not be touched.

Specialized & Network Disks

 Add a disk...
--

Disks left unselected here will not be touched.

Storage Configuration

Automatic Custom

[Full disk summary and boot loader...](#) 1 disk selected; 100 GiB capacity; 100 GiB free [Refresh...](#)

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

okd4-services Actions ×

MANUAL PARTITIONING

Done Help

CENTOS LINUX 8 INSTALLATION

New CentOS Linux 8 Installation

You haven't created any mount points for your CentOS Linux 8 installation yet. You can:

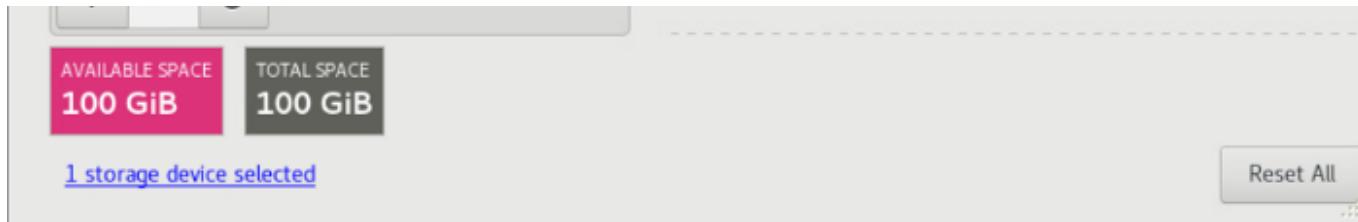
- [Click here to create them automatically.](#)
- Create new mount points by clicking the '+' button.

New mount points will use the following partitioning scheme:

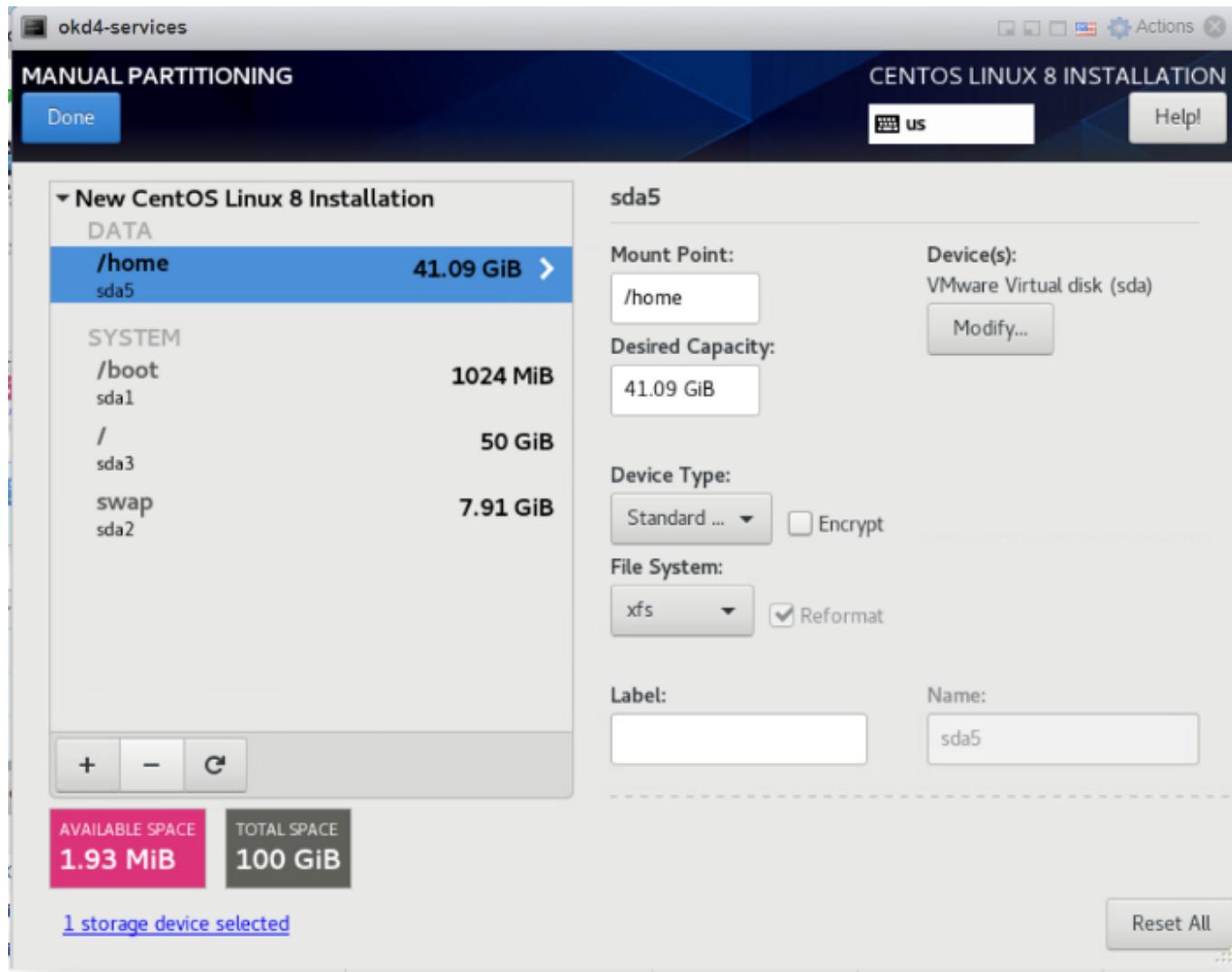
Standard Partition ▾

When you create mount points for your CentOS Linux 8 installation, you'll be able to view their details here.

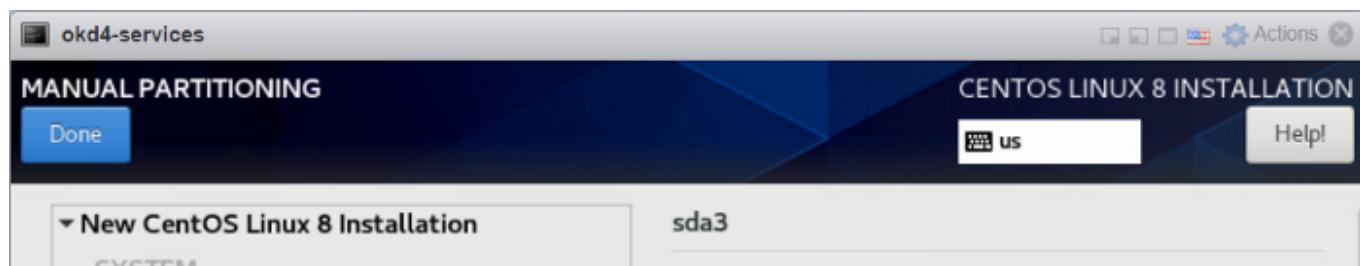
+ - C

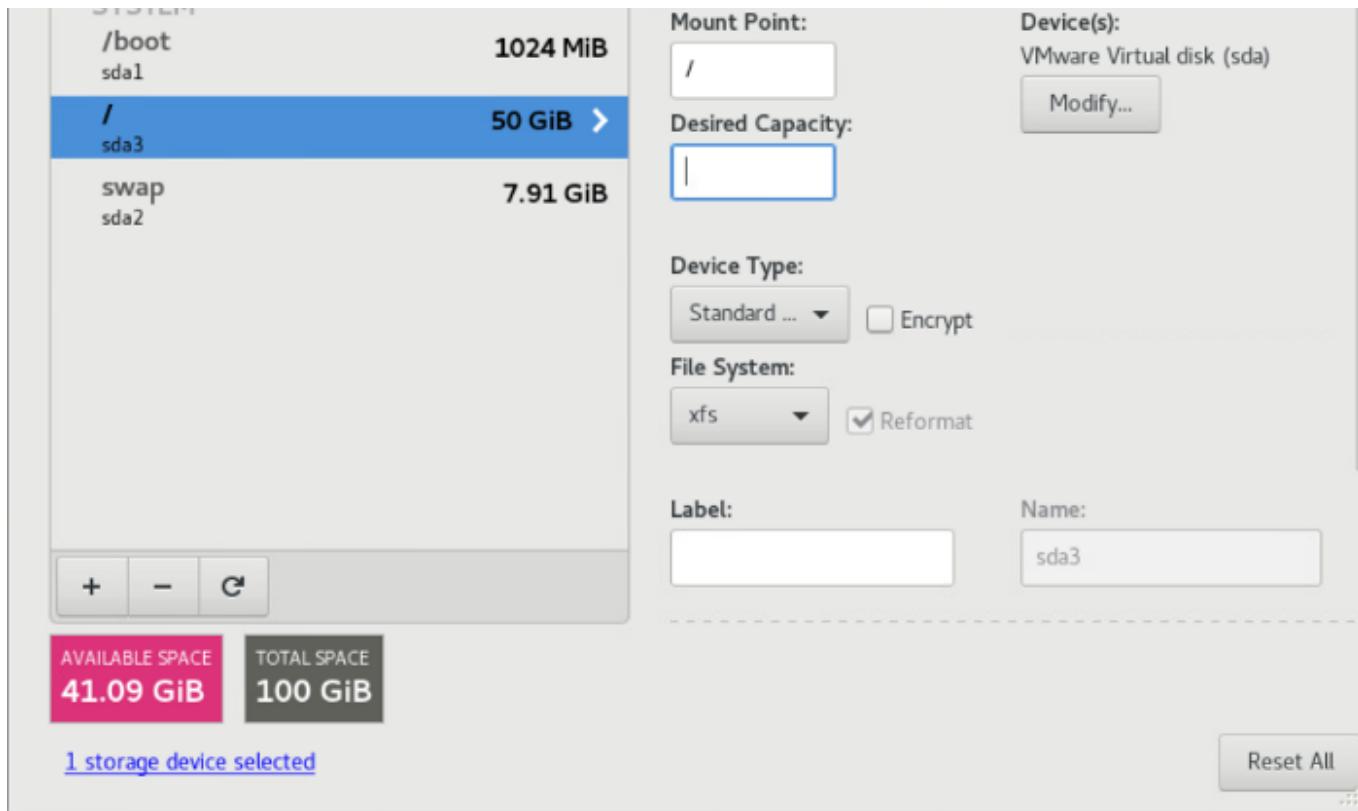


Select the “/home” partition and click the “-” to delete it.

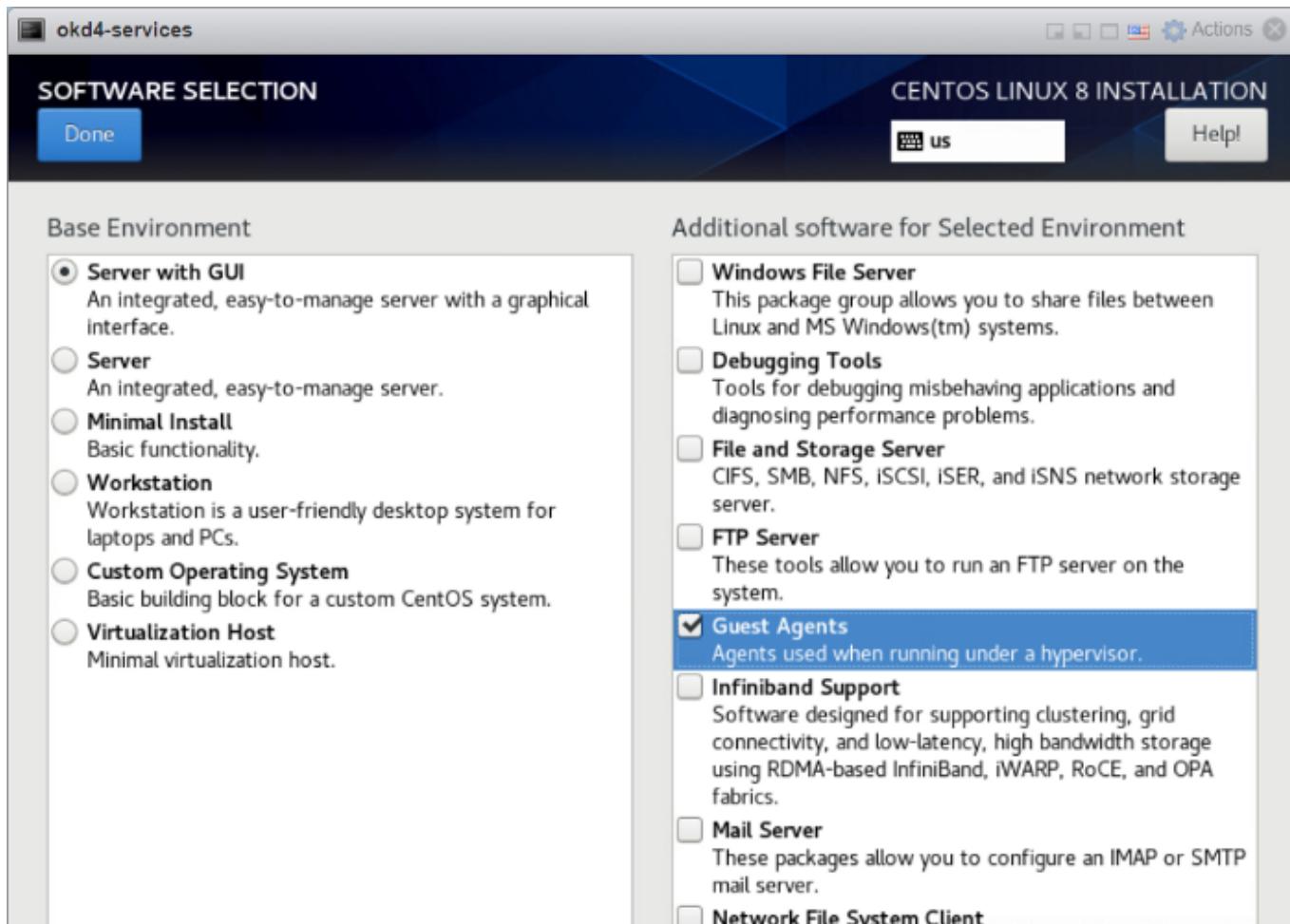


Remove the contents of the “Desired Capacity” field, so it is blank and click Done, then Accept the changes.



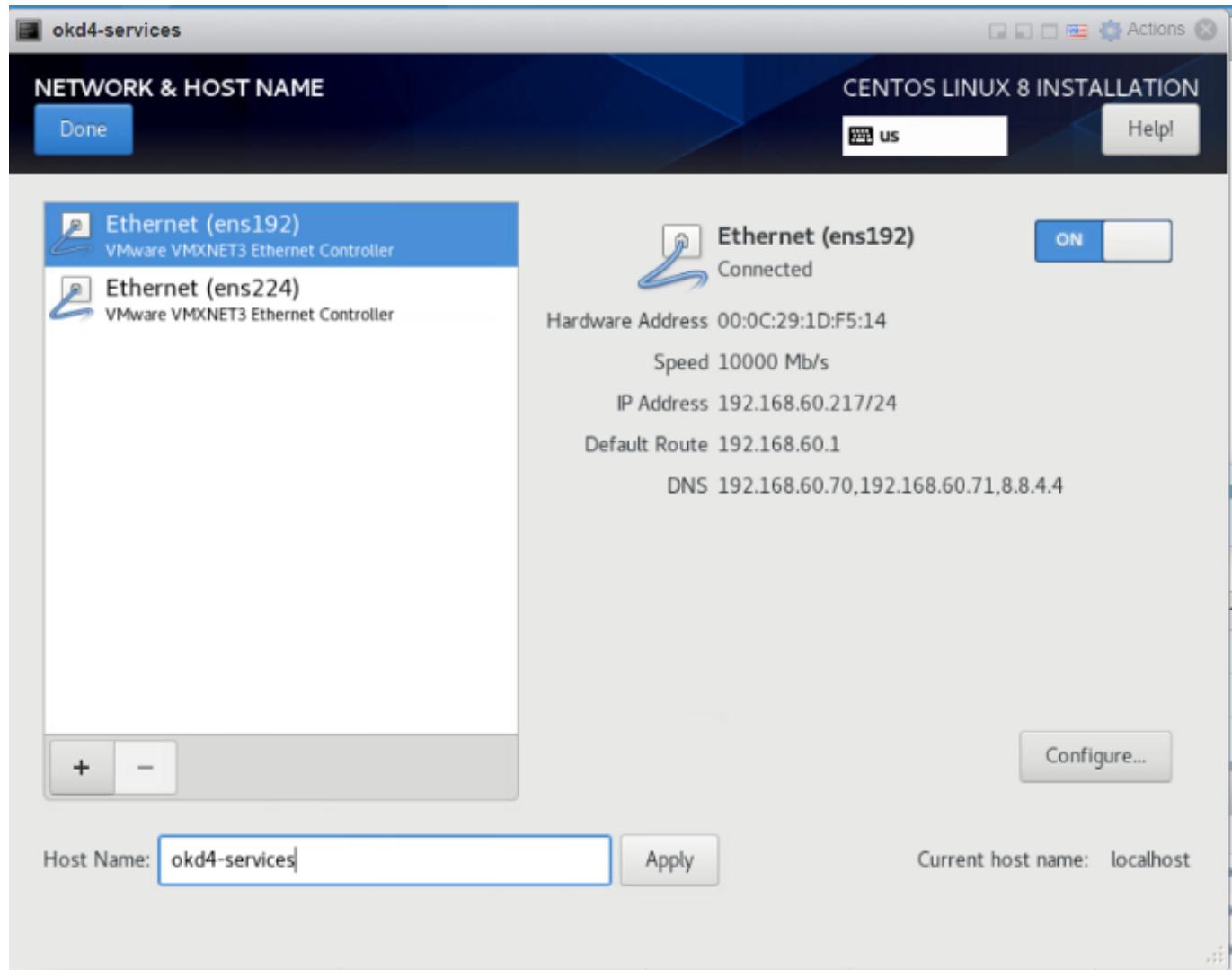


For Software Selection, use Server with GUI and add the Guest Agents.





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

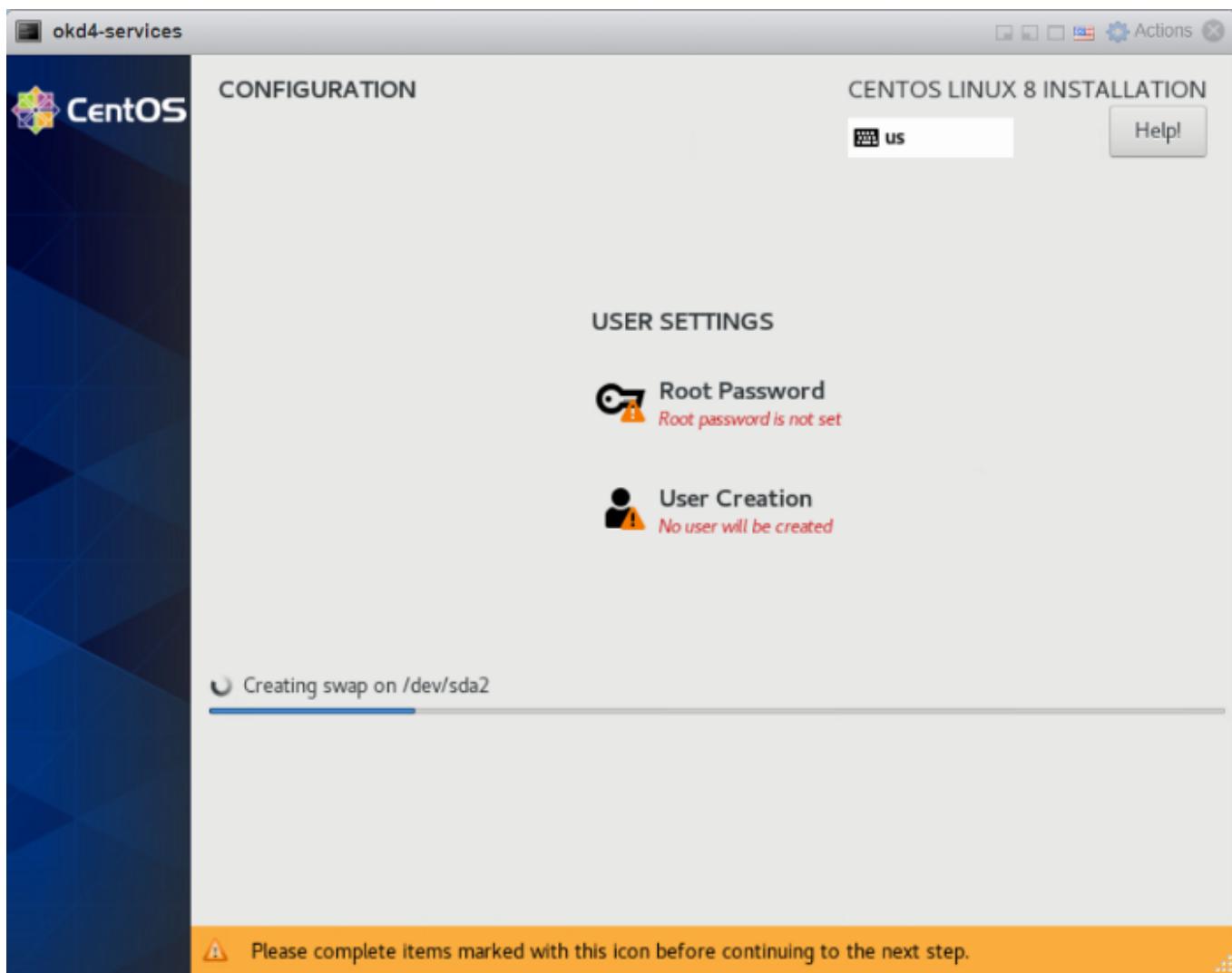


Click “Begin Installation” to start the install.





Set the Root password, and create an admin user.



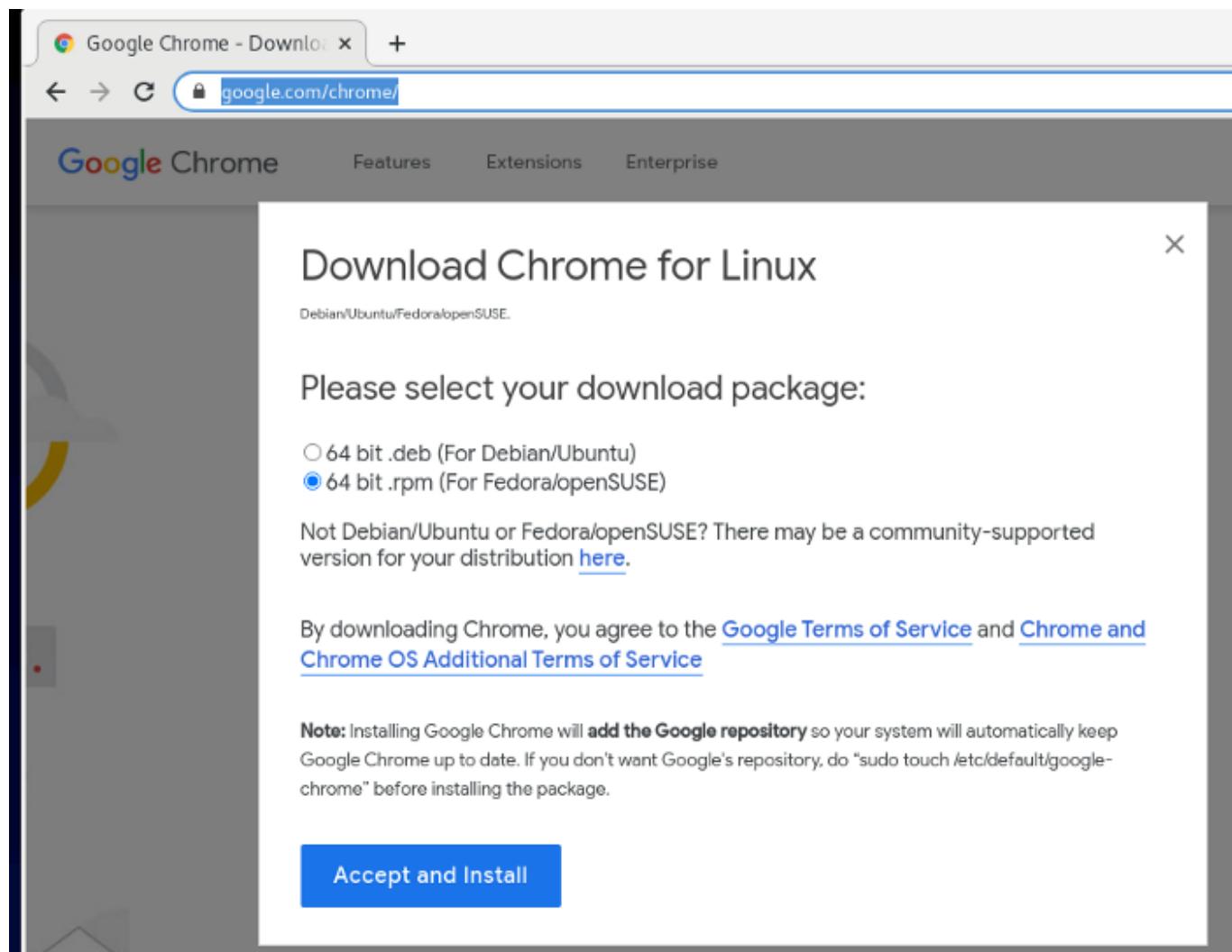
After the installation has completed, login, and update the OS.

```
sudo dnf install -y epel-release  
sudo dnf update -y  
sudo systemctl restart
```

Setup XRDP for Remote Access from Home Network

```
sudo dnf install -y xrdp tigervnc-server  
sudo systemctl enable --now xrxp  
sudo firewall-cmd --zone=public --permanent --add-port=3389/tcp  
sudo firewall-cmd --reload
```

Download Google Chrome rpm and install along with git



```
sudo dnf install -y ~/Downloads/google-chrome-stable_current_x86_64.rpm git
```

• • •

Create the okd4-pfsense VM:

Download the pfSense ISO and upload it to your ESXi host's datastore.

 RELEASE NOTES  SOURCE CODE

Select Image To Download

Version: 2.4.5-p1

Architecture: AMD64 (64-bit) 

Installer: CD Image (ISO) Installer

Mirror: New York City, USA

Supported by 

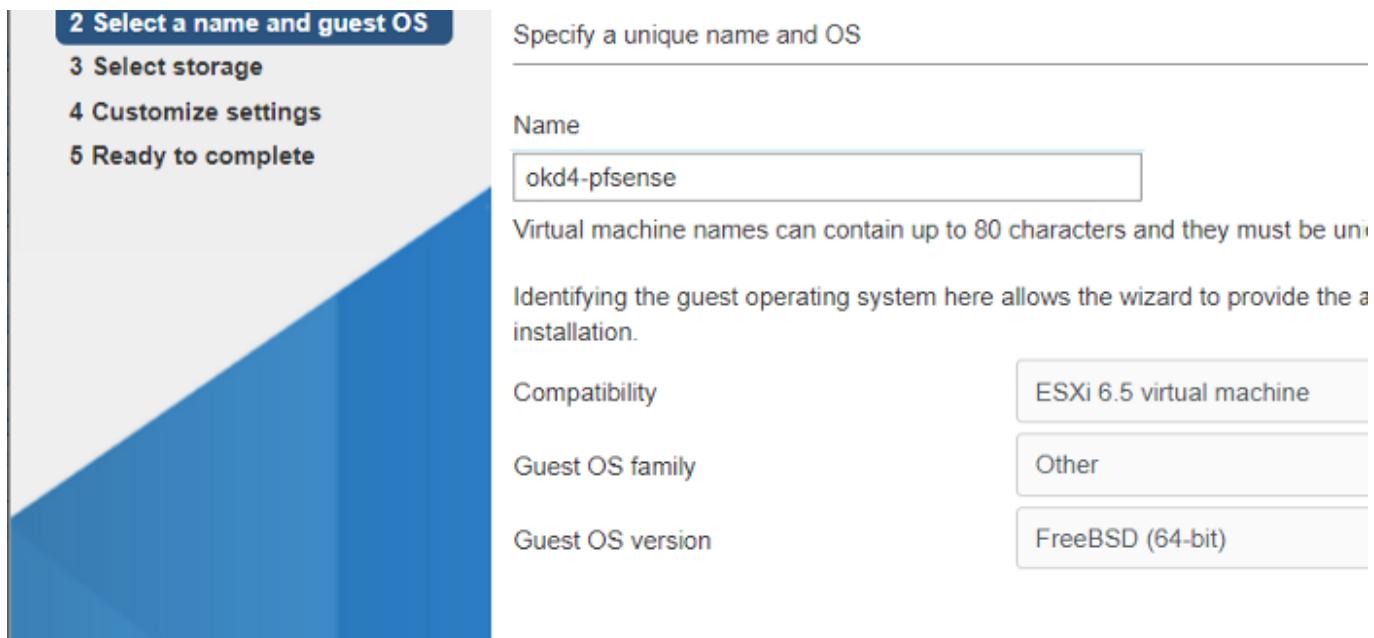
 DOWNLOAD

[SHA256 Checksum](#) for compressed (.gz) file:
0a09a7748419c86c665eb8d908f584e96d54859aa13f4eeb175a60548c70e228

Create a new Virtual Machine. Choose Guest OS as *Other* and Select FreeBSD 64-bit.

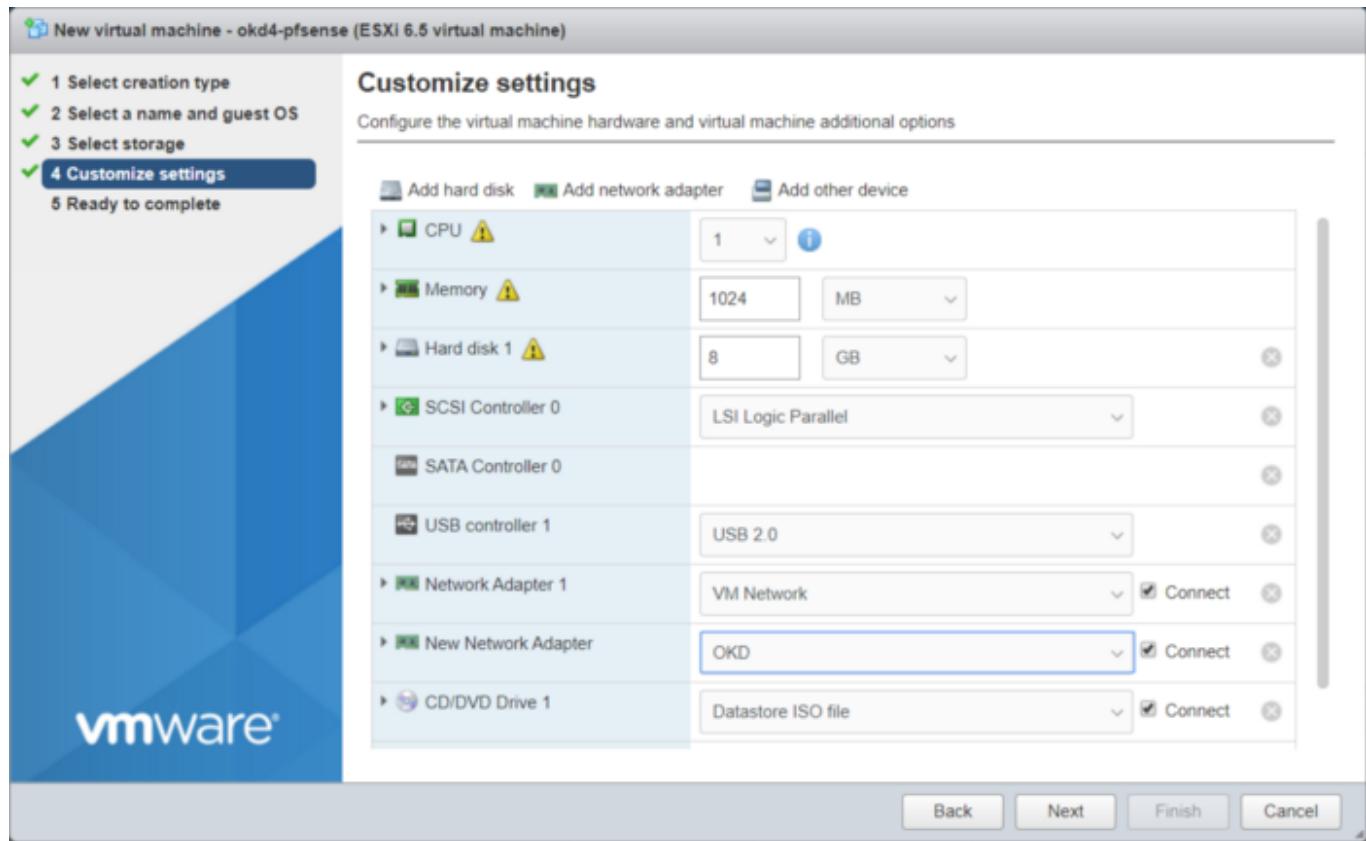
New virtual machine - okd4-pfsense (ESXi 6.5 virtual machine)

✓ 1 Select creation type Select a name and guest OS



Use the default template settings for resources.

Select your home network for Network Adapter 1, and add a new network adapter using the OKD network.

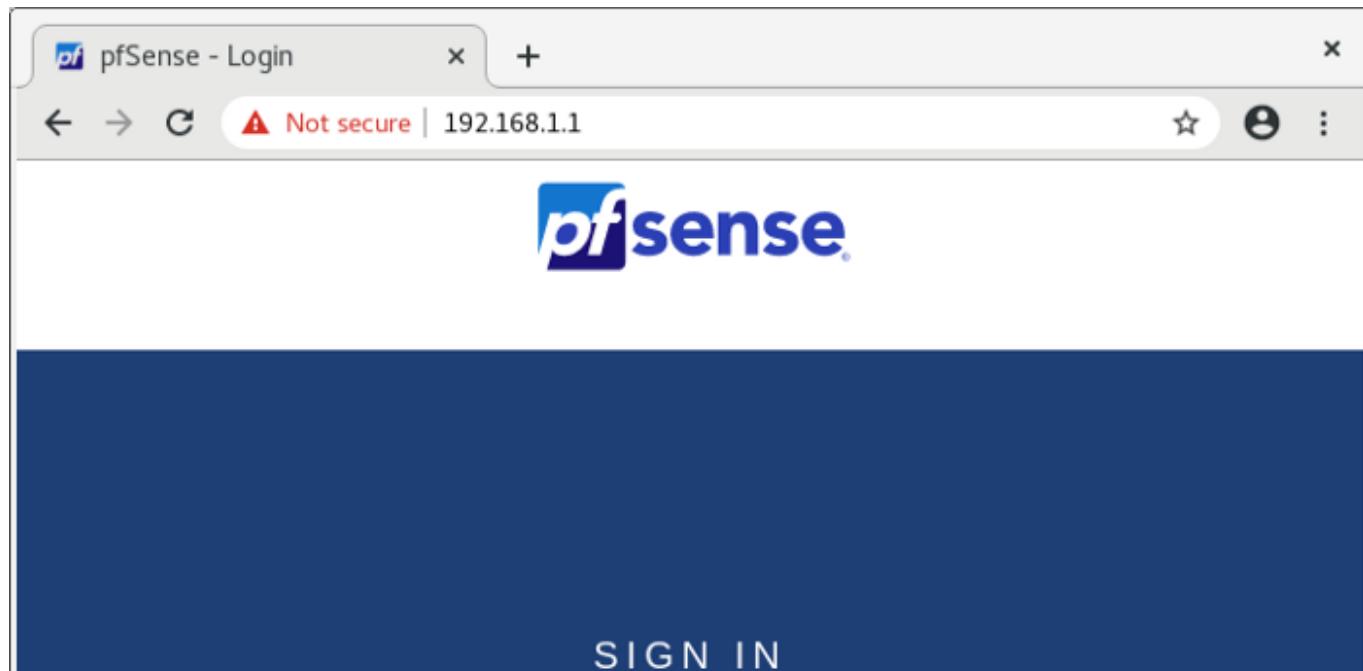


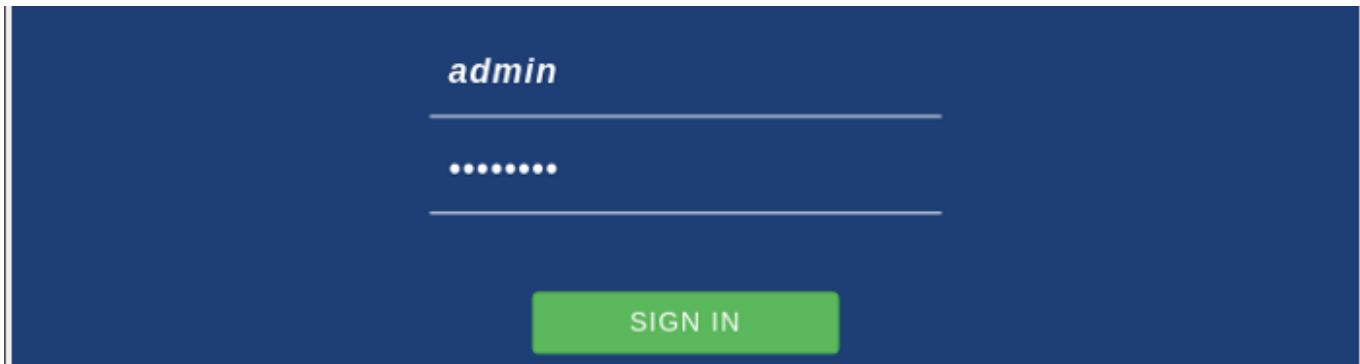
Setup the okd4-pfsense VM:

Power on your pfSense VM and run through the installation using all the default values.
After completion your VM console should look like this:

The screenshot shows a terminal window titled "okd4-pfsense". The output includes the following text:
Starting syslog...done.
Starting CRON... done.
pfSense 2.4.5-RELEASE (Patch 1) amd64 Tue Jun 02 17:51:17 EDT 2020
Bootup complete
FreeBSD/amd64 (okd-pfsense.okd.local) (ttyv0)
VMware Virtual Machine - Netgate Device ID: d728f3f33c3a8370c65c
*** Welcome to pfSense 2.4.5-RELEASE-p1 (amd64) on okd-pfsense ***
WAN (wan) -> em0 -> v4/DHCP4: 192.168.60.239/24
LAN (lan) -> em1 -> v4: 192.168.1.1/24
A list of numbered options follows:
0) Logout (SSH only) 9) pfTop
1) Assign Interfaces 10) Filter Logs
2) Set interface(s) IP address 11) Restart webConfigurator
3) Reset webConfigurator password 12) PHP shell + pfSense tools
4) Reset to factory defaults 13) Update from console
5) Reboot system 14) Enable Secure Shell (sshd)
6) Halt system 15) Restore recent configuration
7) Ping host 16) Restart PHP-FPM
8) Shell
Enter an option: █

Login to pfSense via your web-browser on the okd4-services VM. The default username is “admin” and the password is “pfsense”.





After logging in, click next and use “okd4-pfsense” for hostname and “okd.local” for the domain and add 192.168.1.210 as the Primary DNS server.

Wizard / pfSense Setup / General Information

Step 2 of 9

General Information

On this screen the general pfSense parameters will be set.

Hostname okd4-pfsense
EXAMPLE: myserver

Domain okd.local
EXAMPLE: mydomain.com

The default behavior of the DNS Resolver will ignore manually configured DNS servers for client servers directly. To use the manually configured DNS servers below for client queries, visit [Setup DNS Forwarding](#) after completing the wizard.

Primary DNS Server S

Secondary DNS Server (empty)

Override DNS Allow DNS servers to be overridden by DHCP/PPP on WAN

» Next

Select your Timezone. Next.

Use Defaults for WAN Configuration. Uncheck “Block RFC1918 Private Networks” since your home network is the “WAN” in this setup. Next.

RFC1918 Networks

Block RFC1918 Private Networks

- Block private networks from entering via WAN

When set, this option blocks traffic from IP addresses that are reserved for private networks (10/8, 172.16/12, 192.168/16) as well as loopback addresses (127/8). This option, unless the WAN network lies in such a private address space, too.

Block bogon networks

Block bogon networks

- Block non-Internet routed networks from entering via WAN

When set, this option blocks traffic from IP addresses that are reserved by IANA. Bogons are prefixes that should never appear in the Internet routing table as the source address in any packets received.

» Next

Use the default LAN IP and subnet mask. Set an admin password on the next screen.

Wizard / pfSense Setup / Configure LAN Interface

Step 5 of 9

Configure LAN Interface

On this screen the Local Area Network information will be configured.

LAN IP Address

192.168.1.1

Type dhcp if this interface uses DHCP to obtain its IP address.

Subnet Mask

24

[» Next](#)

• • •

Create bootstrap, master, and worker 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 32.20200629.3.0



Download Fedora CoreOS

Fedora CoreOS is available across 3 different releases:

Stream	Version	Last Updated	Action
Stable	v 32.20200629.3.0	JSON — 5 days ago	Show Downloads
Testing	v 32.20200629.2.0	JSON — 14 days ago	Show Downloads

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.

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.

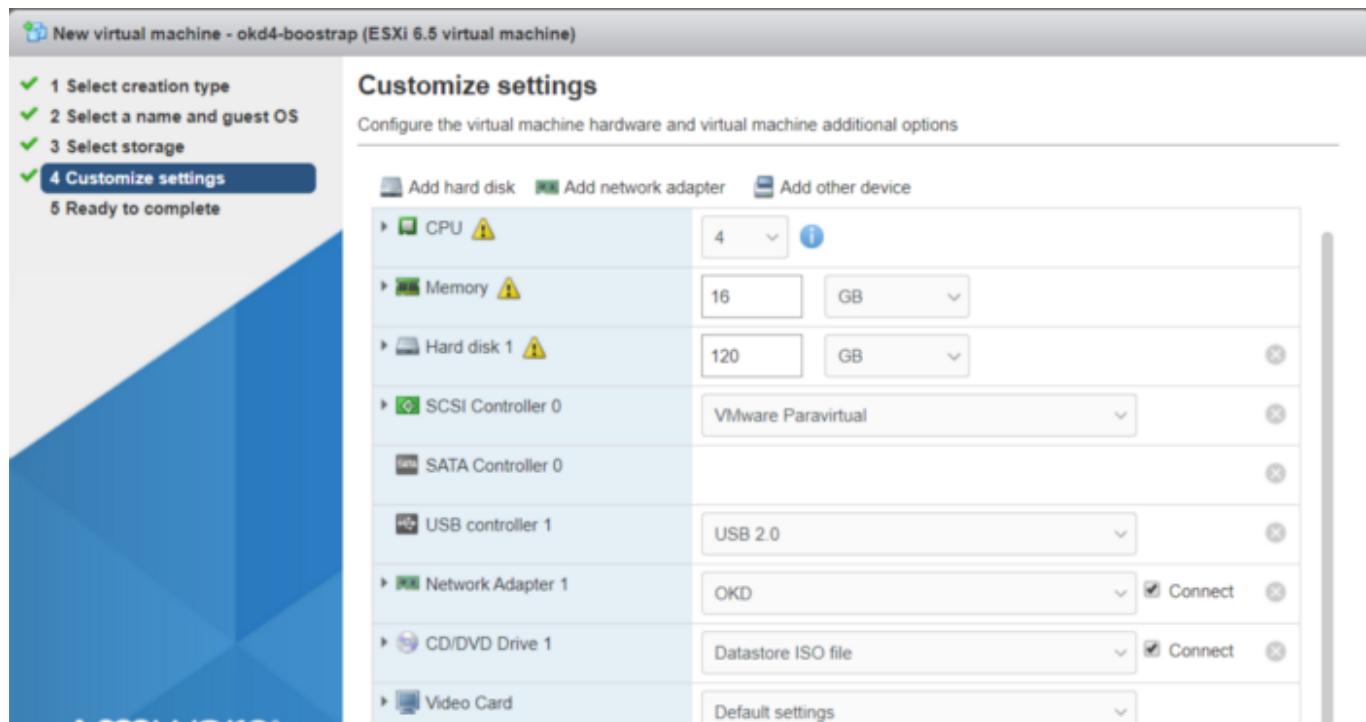
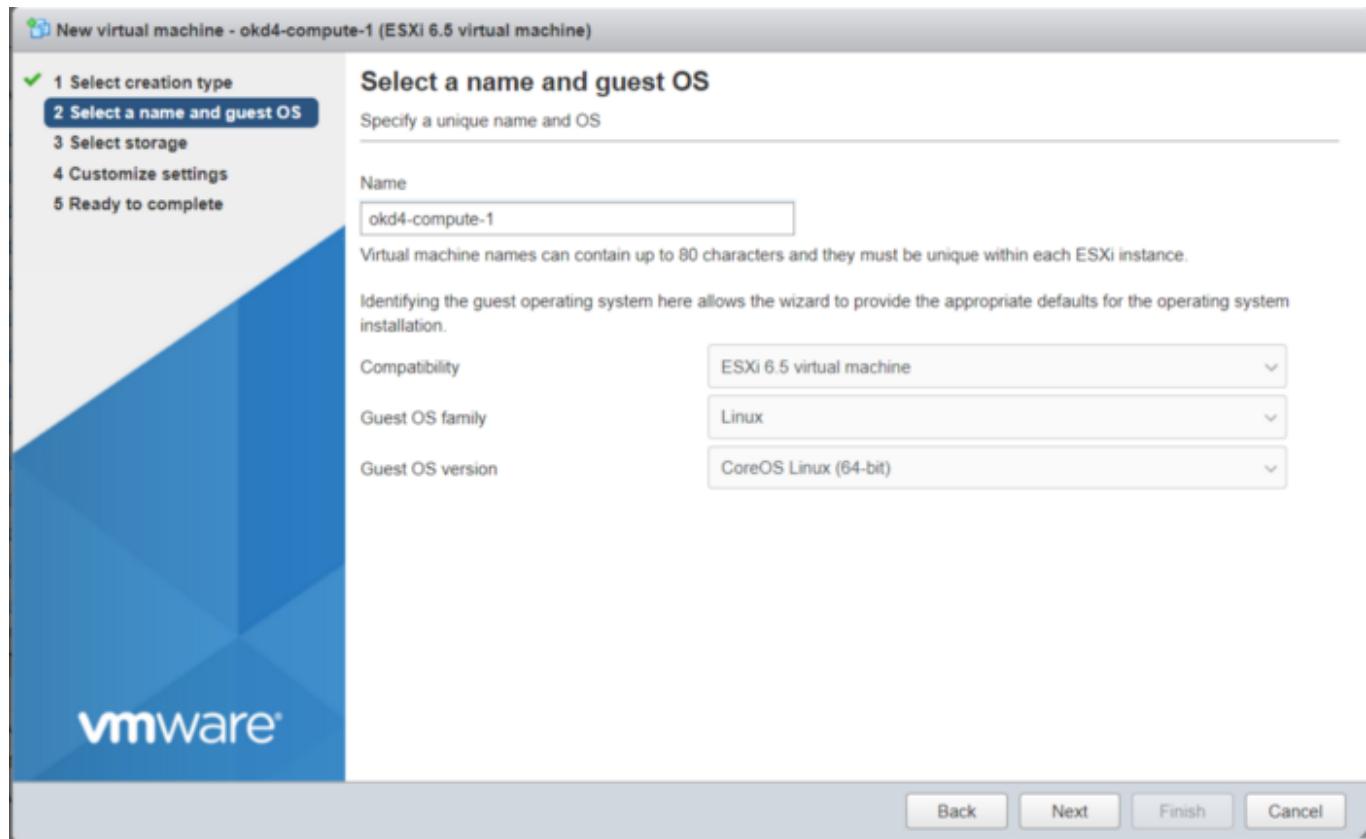
Currently displayed stream: Stable ([View Releases](#))

Cloud Launchable Bare Metal & Virtualized

Bare Metal

Virtualized

Create the six ODK nodes (bootstrap, master, worker) on your ESXi host using the values in the spreadsheet at the beginning of this post:





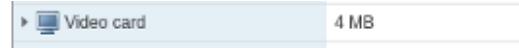
You should end up with the following VMs:

	Virtual machine ▲	Status	Guest OS	Used space	H
1	okd4-bootstrap	Normal	CoreOS Linux (64-bit)	Unknown	U
2	okd4-compute-1	Normal	CoreOS Linux (64-bit)	Unknown	U
3	okd4-compute-2	Normal	CoreOS Linux (64-bit)	Unknown	U
4	okd4-control-plane-1	Normal	CoreOS Linux (64-bit)	Unknown	U
5	okd4-control-plane-2	Normal	CoreOS Linux (64-bit)	Unknown	U
6	okd4-control-plane-3	Normal	CoreOS Linux (64-bit)	Unknown	U
7	okd4-pfsense	Normal	FreeBSD (64-bit)	9.11 GB	U

Setup DHCP reservations:

Compile a list of the OKD nodes MAC addresses by viewing the hardware configuration of your VMs.

The screenshot shows the VMware vSphere Client interface for the 'okd4-bootstrap' VM. The top navigation bar includes 'Console', 'Monitor', 'Power on', 'Power off', 'Suspend', 'Restart', 'Edit', 'Refresh', and 'Actions'. The main summary panel shows the VM's name, guest OS (CoreOS Linux 64-bit), compatibility (ESXI 6.5 and later, VM version 13), and resource allocation (4 vCPUs, 16 GB memory). Below this, two expandable sections provide detailed information: 'General Information' (Networking, VMware Tools, Storage, Notes) and 'Hardware Configuration' (CPU, Memory, Hard disk 1, USB controller, Network adapter 1). The 'Network adapter 1' section is highlighted with a red oval around the 'MAC address' field, which contains the value '00:0c:29:45:cf:d1'.



A screenshot of the pfSense DHCP Server configuration page. It shows two sections:

- Additional Pools:** A table with columns Pool Start, Pool End, and Description. There are no entries yet.
- Servers:** A table with columns WINS servers, DNS servers, and DNS servers. The 'DNS servers' row contains the value '192.168.1.210', which is circled in red.

On the DHCP Server, page click Add at the bottom.

A screenshot of the 'DHCP Static Mappings for this Interface' page. It has a table with columns: Static ARP, MAC address, IP address, Hostname, and Description. A green 'Add' button is located at the bottom right.

Fill in the MAC Address, IP Address, and Hostname, then click save. Do this for each OKD VM. Also, include the okd4-services MAC on the OKD network while you are at it. Click Apply Changes at the top of the page when complete.

DHCP Static Mappings for this Interface				
Static ARP	MAC address	IP address	Hostname	Description
	00:0c:29:45:cf:d1	192.168.1.200	okd4-bootstrap	
	00:0c:29:2f:5b:9d	192.168.1.201	okd4-control-plane-1	
	00:0c:29:be:2f:16	192.168.1.202	okd4-control-plane-2	
	00:0c:29:16:0a:03	192.168.1.203	okd4-control-plane-3	
	00:0c:29:90:2d:76	192.168.1.204	okd4-compute-1	
	00:0c:29:56:be:a9	192.168.1.205	okd4-compute-2	

[Add](#)

Configure okd4-services VM to host various services:

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

Copy the MAC address on the VM Hardware configuration page for the NIC connected to the OKD network and set up a DHCP Reservation for this VM using the IP address 192.168.1.210.

Guest OS	CentOS 7 (64-bit)												
Compatibility	ESXi 6.5 and later (VM version 13)												
VMware Tools	Yes												
CPU	4												
Memory	4 GB												
General Information <ul style="list-style-type: none"> Networking VMware Tools Installed and running Storage 1 disk Notes 													
Performance summary last hour <p>Ready (%)</p> <p>Consumed host CPU Ready Consumed host memory...</p>													
Hardware Configuration <table border="1"> <tbody> <tr> <td>CPU</td> <td>4 vCPUs</td> </tr> <tr> <td>Memory</td> <td>4 GB</td> </tr> <tr> <td>Hard disk 1</td> <td>100 GB</td> </tr> <tr> <td>USB controller</td> <td>USB 2.0</td> </tr> <tr> <td>Network adapter 1</td> <td>VM Network (Connected)</td> </tr> <tr> <td>Network adapter 2</td> <td> Network: OKD (Connected) Connected: Yes MAC address: 00:0c:29:1d:f5:1e Pass-through (Direct-path I/O): Yes </td> </tr> </tbody> </table>		CPU	4 vCPUs	Memory	4 GB	Hard disk 1	100 GB	USB controller	USB 2.0	Network adapter 1	VM Network (Connected)	Network adapter 2	Network: OKD (Connected) Connected: Yes MAC address: 00:0c:29:1d:f5:1e Pass-through (Direct-path I/O): Yes
CPU	4 vCPUs												
Memory	4 GB												
Hard disk 1	100 GB												
USB controller	USB 2.0												
Network adapter 1	VM Network (Connected)												
Network adapter 2	Network: OKD (Connected) Connected: Yes MAC address: 00:0c:29:1d:f5:1e Pass-through (Direct-path I/O): Yes												

Hit “Apply Changes” at the top of the DHCP page when completed.

DHCP Static Mappings for this Interface				
Static ARP	MAC address	IP address	Hostname	Description

00:0c:29:45:cf:d1	192.168.1.200	okd4-bootstrap		
00:0c:29:2f:5b:9d	192.168.1.201	okd4-control-plane-1		
00:0c:29:be:2f:16	192.168.1.202	okd4-control-plane-2		
00:0c:29:16:0a:03	192.168.1.203	okd4-control-plane-3		
00:0c:29:90:2d:76	192.168.1.204	okd4-compute-1		
00:0c:29:56:be:a9	192.168.1.205	okd4-compute-2		
00:0c:29:1d:f5:1e	192.168.1.210	okd4-services		

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

```
cd
git clone 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
```

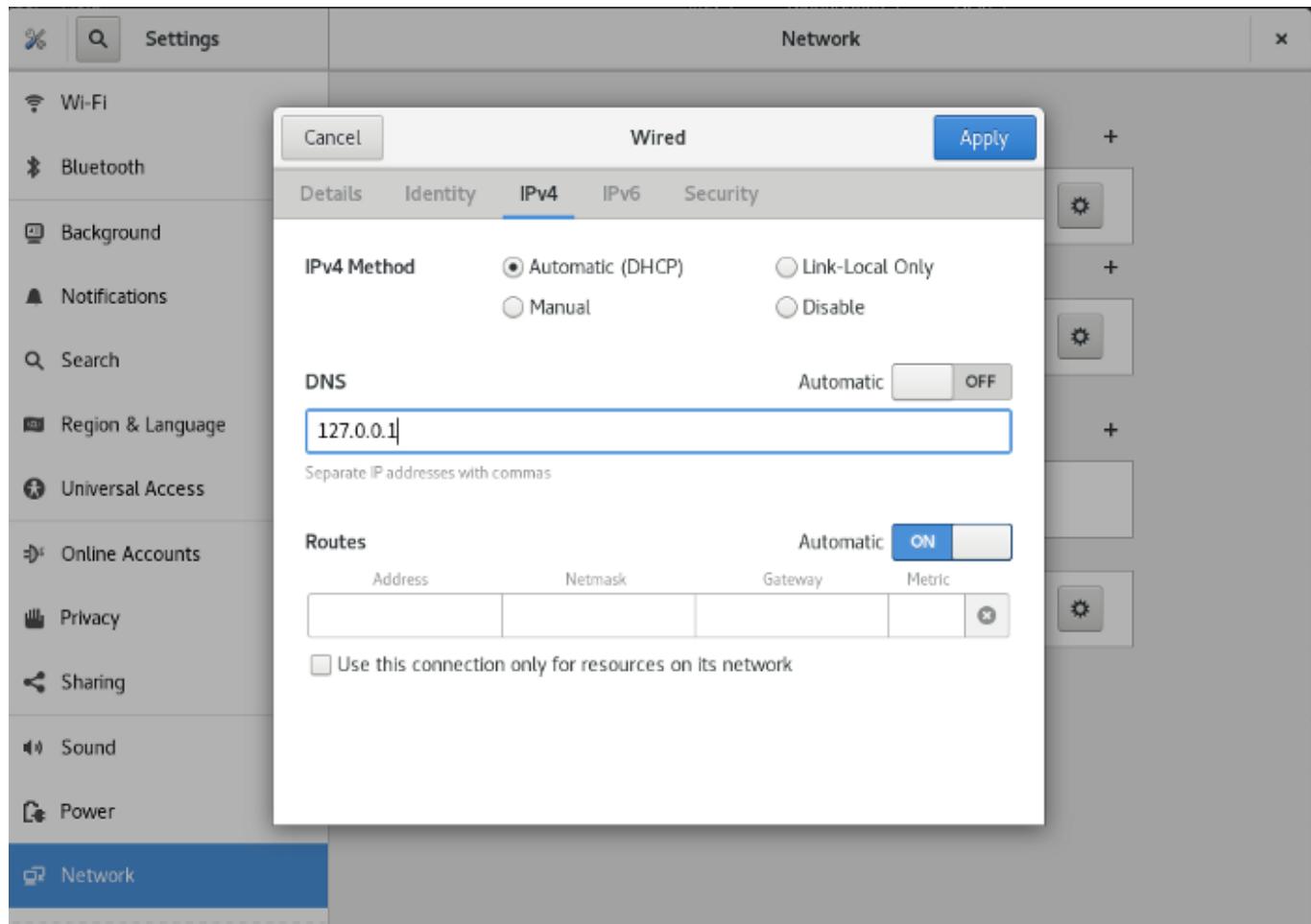
Enable and start named:

```
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 NIC that is attached to the VM Network (not OKD) to 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 192.168.1.210
```

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
kd.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 okd_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:
210.1.168.192.in-addr.arpa. 604800 IN  NS      okd4-services.okd.local.
```

```
1.168.192.100-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  7 14:54:17 EDT 2020
;; MSG SIZE  rcvd: 194
```

• • •

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!

Congrats! You should now have a separate home lab environment setup and ready for OKD. Now we can start the install.

• • •

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:

```
wget https://github.com/openshift/okd/releases/download/4.5.0-0.okd-2020-07-14-153706-ga/openshift-client-linux-4.5.0-0.okd-2020-07-14-153706-ga.tar.gz
wget https://github.com/openshift/okd/releases/download/4.5.0-0.okd-2020-07-14-153706-ga/openshift-install-linux-4.5.0-0.okd-2020-07-14-153706-ga.tar.gz
```

Extract the okd version of the oc client and openshift-install:

```
tar -zxvf openshift-client-linux-4.5.0-0.okd-2020-07-14-153706-
ga.tar.gz
tar -zxvf openshift-install-linux-4.5.0-0.okd-2020-07-14-153706-
ga.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, insert your pull secret and ssh key, and backup the install-config.yaml as it will be deleted in the next step:

```
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/
```

Modify the cluster-scheduler-02-config.yaml manifest file to prevent Pods from being scheduled on the control plane machines:

```
sed -i 's/mastersSchedulable: true/mastersSchedulable: False/'  
install_dir/manifests/cluster-scheduler-02-config.yml
```

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
.20200629.3.0/x86_64/fedora-coreos-32.20200629.3.0-
metal.x86_64.raw.xz
sudo wget
https://builds.coreos.fedoraproject.org/prod/streams/stable/builds/32
.20200629.3.0/x86_64/fedora-coreos-32.20200629.3.0-
metal.x86_64.raw.xz.sig
sudo mv fedora-coreos-32.20200629.3.0-metal.x86_64.raw.xz fcos.raw.xz
sudo mv fedora-coreos-32.20200629.3.0-metal.x86_64.raw.xz.sig
fcos.raw.xz.sig
sudo chown -R apache: /var/www/html/
sudo chmod -R 755 /var/www/html/
```

• • •

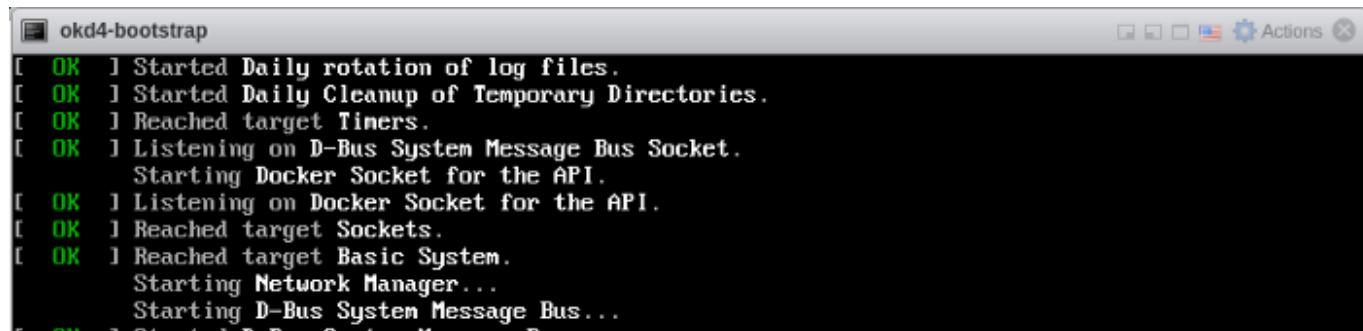
Starting the bootstrap node:

Power on the odk4-bootstrap VM. Press the TAB key to edit the kernel boot options and add the following:

```
coreos.inst.install_dev=/dev/sda
coreos.inst.image_url=http://192.168.1.210:8080/okd4/fcos.raw.xz
coreos.inst.ignition_url=http://192.168.1.210:8080/okd4/bootstrap.ign
```



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



```
[ 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...
#####
# 100.0%
[ 12.392624] coreos-installer-service[1073]: coreos-installer install /dev/sda --ignition /tmp/coreos-installer-fBbRRu --image-url http://192.168.1.210:8080/okd4/fcos.raw.xz
[ 12.423413] coreos-installer-service[1094]: Downloading image from http://192.168.1.210:8080/okd4/fcos.raw.xz
[ 12.424616] coreos-installer-service[1094]: Downloading signature from http://192.168.1.210:8080/okd4/fcos.raw.xz.sig
[ 13.493506] coreos-installer-service[1094]: Read disk 29.6 MiB/481.2 MiB (6%)
[ 14.493888] coreos-installer-service[1094]: Read disk 65.3 MiB/481.2 MiB (13%)
[ 15.494417] coreos-installer-service[1094]: Read disk 74.4 MiB/481.2 MiB (15%)
[ 16.694874] coreos-installer-service[1094]: Read disk 74.5 MiB/481.2 MiB (15%)
[ 17.778351] coreos-installer-service[1094]: Read disk 74.7 MiB/481.2 MiB (15%)
[ 18.779196] coreos-installer-service[1094]: Read disk 75.1 MiB/481.2 MiB (15%)
[ 20.433895] coreos-installer-service[1094]: Read disk 75.6 MiB/481.2 MiB (15%)
.
```

Starting the control plane nodes:

Power on the control-plane nodes and press the TAB key to edit the kernel boot options and add the following, then press enter:

```
coreos.inst.install_dev=/dev/sda
coreos.inst.image_url=http://192.168.1.210:8080/okd4/fcos.raw.xz
coreos.inst.ignition_url=http://192.168.1.210:8080/okd4/master.ign
```



```
> /images/vmlinuz initrd=/images/initramfs.img mitigations=auto,nosmt systemd.unified_cgroup_hierarchy=0 coreos.liveiso=fedora-coreos-32.20200615.3.0 ignition.firstboot ignition.platform.id=metal coreos.inst.install_dev=/dev/sda coreos.inst.image_url=http://192.168.1.210:8080/okd4/fcos.raw.xz coreos.inst.ignition_url=http://192.168.1.210:8080/okd4/master.ign_
```

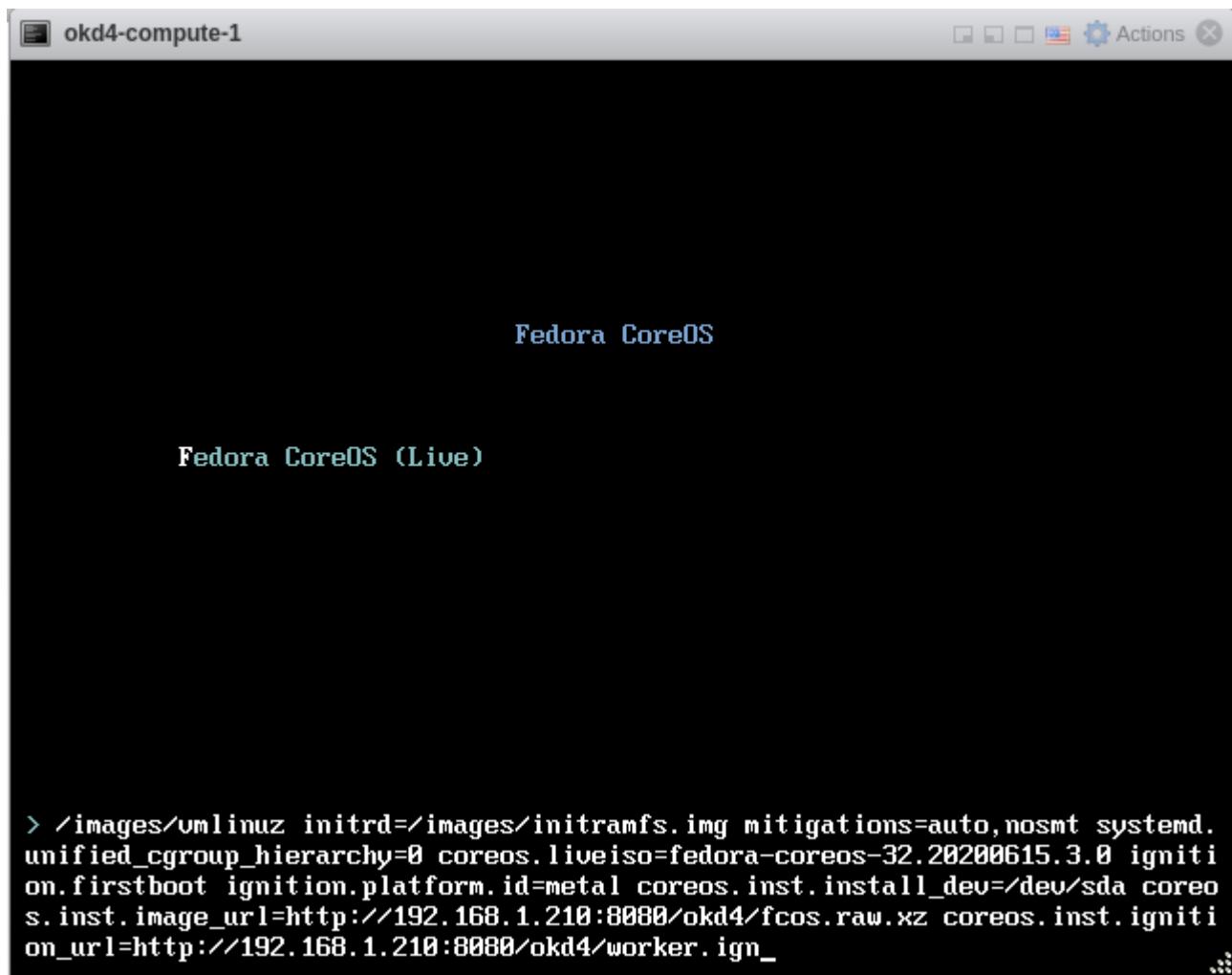
You should see that the fcos.raw.gz image and signature are downloading:

```
okd4-control-plane-2
[ OK ] Reached target Timers.
[ OK ] Listening on D-Bus System Message Bus Socket.
      Starting Docker Socket for the API.
[ OK ] Listening on Docker Socket for the API.
[ OK ] Reached target Sockets.
[ OK ] Reached target Basic System.
      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...
=====
[ 12.678788] coreos-installer-service[1077]: coreos-installer install /dev/sda --ignition /tmp/coreos-installer-f1GqNz --image-url http://192.168.1.210:8080/okd4/fcos.raw.xz
[ 12.718790] coreos-installer-service[1094]: Downloading image from http://192.168.1.210:8080/okd4/fcos.raw.xz
[ 12.720319] coreos-installer-service[1094]: Downloading signature from http://192.168.1.210:8080/okd4/fcos.raw.xz.sig
[ 13.804045] coreos-installer-service[1094]: Read disk 28.0 MiB/481.2 MiB (5%)
[ 14.805276] coreos-installer-service[1094]: Read disk 61.8 MiB/481.2 MiB (12%)
[ 15.893925] coreos-installer-service[1094]: Read disk 74.3 MiB/481.2 MiB (15%)
[ 17.352149] coreos-installer-service[1094]: Read disk 74.5 MiB/481.2 MiB (15%)
[ 18.722067] coreos-installer-service[1094]: Read disk 74.7 MiB/481.2 MiB (15%)
[ 19.881882] coreos-installer-service[1094]: Read disk 74.7 MiB/481.2 MiB (15%)
[ 20.908220] coreos-installer-service[1094]: Read disk 75.5 MiB/481.2 MiB (15%)
[ 21.916272] coreos-installer-service[1094]: Read disk 75.6 MiB/481.2 MiB (15%)
[ 23.173243] coreos-installer-service[1094]: Read disk 87.2 MiB/481.2 MiB (18%)
```

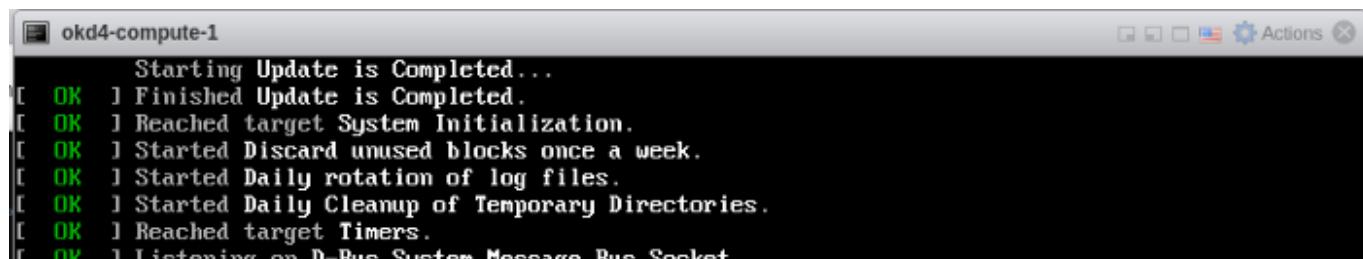
Starting the compute nodes:

Power on the control-plane nodes and press the TAB key to edit the kernel boot options and add the following, then press enter:

```
coreos.inst.install_dev=/dev/sda
coreos.inst.image_url=http://192.168.1.210:8080/okd4/fcos.raw.xz
coreos.inst.ignition_url=http://192.168.1.210:8080/okd4/worker.ign
```



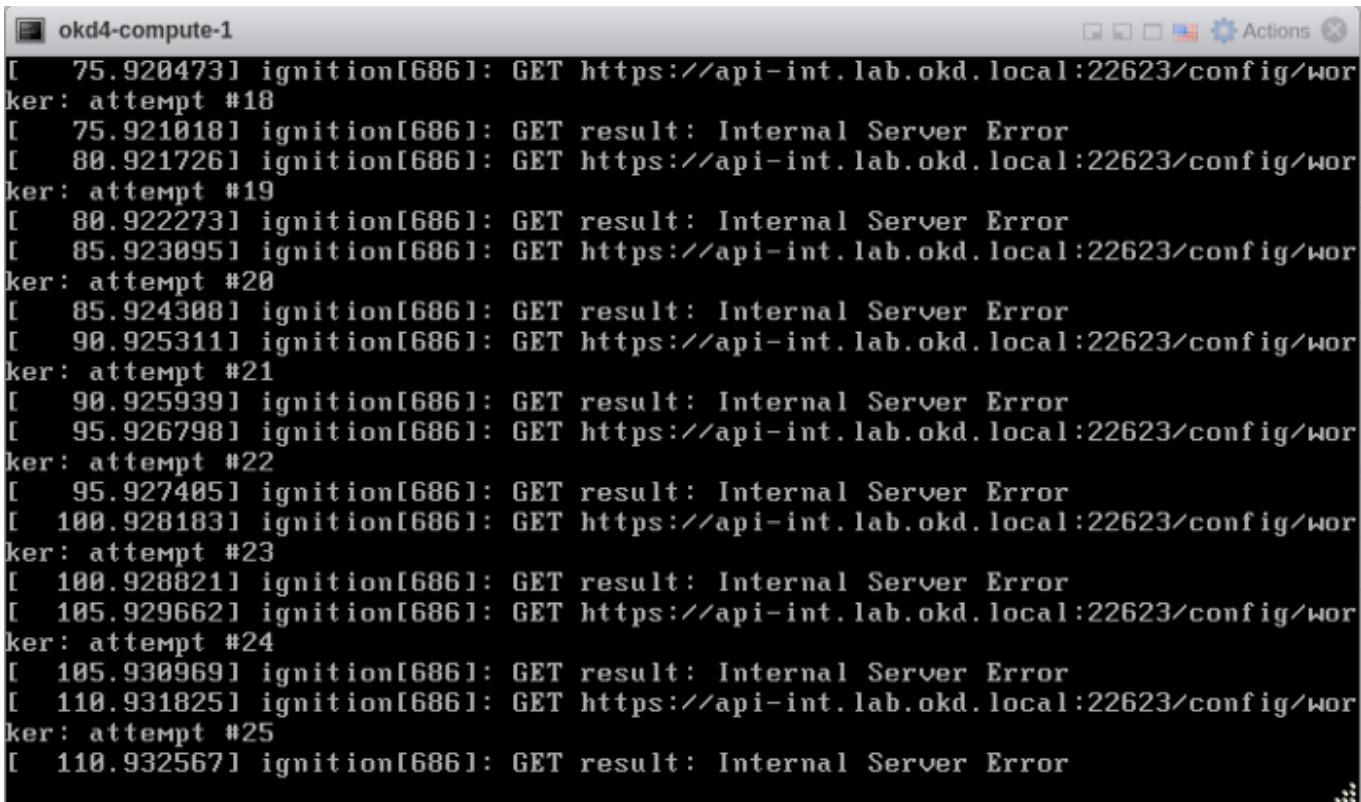
You should see that the fcos.raw.gz image and signature are downloading:



```
[  OK ] Starting Docker Socket for the API.
[  OK ] Listening on Docker Socket for the API.
[  OK ] Reached target Sockets.
[  OK ] Reached target Basic System.
      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...
===== 100.0%
[ 13.690292] coreos-installer-service[1078]: coreos-installer install /dev/sda --ignition /tmp/coreos-installer-0XNqY9 --image-url http://192.168.1.210:8080/okd4/fcos.raw.xz
[ 13.722217] coreos-installer-service[1095]: Downloading image from http://192.168.1.210:8080/okd4/fcos.raw.xz
[ 13.723766] coreos-installer-service[1095]: Downloading signature from http://192.168.1.210:8080/okd4/fcos.raw.xz.sig
[ 14.838985] coreos-installer-service[1095]: Read disk 29.8 MiB/481.2 MiB (6%)
[ 15.839207] coreos-installer-service[1095]: Read disk 64.7 MiB/481.2 MiB (13%)
[ 16.886878] coreos-installer-service[1095]: Read disk 74.3 MiB/481.2 MiB (15%)

```

It is usual for the worker nodes to display the following until the bootstrap process complete:



```
okd4-compute-1
[ 75.920473] ignition[686]: GET https://api-int.lab.okd.local:22623/config/worker: attempt #18
[ 75.921018] ignition[686]: GET result: Internal Server Error
[ 80.921726] ignition[686]: GET https://api-int.lab.okd.local:22623/config/worker: attempt #19
[ 80.922273] ignition[686]: GET result: Internal Server Error
[ 85.923095] ignition[686]: GET https://api-int.lab.okd.local:22623/config/worker: attempt #20
[ 85.924308] ignition[686]: GET result: Internal Server Error
[ 90.925311] ignition[686]: GET https://api-int.lab.okd.local:22623/config/worker: attempt #21
[ 90.925939] ignition[686]: GET result: Internal Server Error
[ 95.926798] ignition[686]: GET https://api-int.lab.okd.local:22623/config/worker: attempt #22
[ 95.927405] ignition[686]: GET result: Internal Server Error
[ 100.928183] ignition[686]: GET https://api-int.lab.okd.local:22623/config/worker: attempt #23
[ 100.928821] ignition[686]: GET result: Internal Server Error
[ 105.929662] ignition[686]: GET https://api-int.lab.okd.local:22623/config/worker: attempt #24
[ 105.930969] ignition[686]: GET result: Internal Server Error
[ 110.931825] ignition[686]: GET https://api-int.lab.okd.local:22623/config/worker: attempt #25
[ 110.932567] ignition[686]: GET result: Internal Server Error
```

• • •

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 boots  
trap-complete --log-level=info  
INFO Waiting up to 20m0s for the Kubernetes API at https://api.lab.okd.local:644  
3...  
INFO API v1.18.3 up  
INFO Waiting up to 40m0s for bootstrapping to complete...  
INFO It is now safe to remove the bootstrap resources
```

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

• • •

Login to the cluster and approve CSRs:

Now that the masters are 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  
oc get csr
```

```
[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 Ready    master   17m    v1.18.3
okd4-control-plane-2 Ready    master   12m    v1.18.3
okd4-control-plane-3 Ready    master   9m54s   v1.18.3
[crobinson@okd4-services ~]$ oc get csr
NAME          AGE      SIGNERNAME             REQUESTOR
              CONDITION
csr-2ggcf    10m     kubernetes.io/kube-apiserver-client-kubelet
g-operator:node-bootstrapper Approved,Issued
csr-2wjcp    17m     kubernetes.io/kube-apiserver-client-kubelet
g-operator:node-bootstrapper Approved,Issued
csr-4lqrm    4m52s   kubernetes.io/kube-apiserver-client-kubelet
g-operator:node-bootstrapper Pending
csr-bxjqn    12m     kubernetes.io/kubelet-serving
                  Approved,Issued
csr-c5mcj    3m33s   kubernetes.io/kube-apiserver-client-kubelet
g-operator:node-bootstrapper Pending
csr-pf92t    17m     kubernetes.io/kubelet-serving
                  Approved,Issued
csr-rf4vx    12m     kubernetes.io/kube-apiserver-client-kubelet
g-operator:node-bootstrapper Approved,Issued
csr-tgfvr    9m56s   kubernetes.io/kubelet-serving
                  Approved,Issued
```

You should only see the master nodes and several CSR's waiting for approval. Install the jq package to assist with approving multiple CSR's at once time.

```
wget -O jq https://github.com/stedolan/jq/releases/download/jq-1.6/jq-linux64
chmod +x jq
sudo mv jq /usr/local/bin/
jq --version
```

```
[crobinson@okd4-services ~]$ wget -O jq https://github.com/stedolan/jq/releases/download/jq-1.6/jq-linux64
--2020-07-07 16:05:41-- https://github.com/stedolan/jq/releases/download/jq-1.6/jq-linux64
Resolving github.com (github.com)... 140.82.114.4
Connecting to github.com (github.com)|140.82.114.4|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://github-production-release-asset-2e65be.s3.amazonaws.com/5101141/6387d980-delf-11e8-8d3e-4455415aa408?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53A%2F20200707%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20200707T200358Z&X-Amz-Expires=300&X-Amz-Signature=5812bef9169856b24b2525bebcb78fe74e755c78319aa542265fdf8a85dcef9&X-Amz-SignedHeaders=host&actor_id=0&repo_id=5101141&response-content-disposition=attachment%3B%20filename%3Djq-linux64&response-content-type=application%2Foctet-stream [following]
--2020-07-07 16:05:41-- https://github-production-release-asset-2e65be.s3.amazonaws.com/5101141/6387d980-delf-11e8-8d3e-4455415aa408?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53A%2F20200707%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20200707T200358Z&X-Amz-Expires=300&X-Amz-Signature=5812bef9169856b24b2525bebcb78fe74e755c78319aa542265fdf8a85dcef9&X-Amz-SignedHeaders=host&actor_id=0&repo_id=5101141&response-content-disposition=attachment%3B%20filename%3Djq-linux64&response-content-type=application%2Foctet-stream
Resolving github-production-release-asset-2e65be.s3.amazonaws.com (github-production-release-asset-2e65be.s3.amazonaws.com)... 52.216.138.43
Connecting to github-production-release-asset-2e65be.s3.amazonaws.com (github-production-rel
```

```
ease-asset-zeb5be.s3.amazonaws.com) | 52.216.138.43] :443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3953824 (3.8M) [application/octet-stream]
Saving to: 'jq'

jq          100%[=====] 3.77M 6.81MB/s in 0.6s

2020-07-07 16:05:42 (6.81 MB/s) - 'jq' saved [3953824/3953824]

[crobinson@okd4-services ~]$ chmod +x jq
[crobinson@okd4-services ~]$ sudo mv jq /usr/local/bin/
[crobinson@okd4-services ~]$ jq --version
jq-1.6
[crobinson@okd4-services ~]$ 
```

Approve all the pending certs and check your nodes:

```
oc get csr -ojson | jq -r '.items[] | select(.status == {} ) | .metadata.name' | xargs oc adm certificate approve
```

```
[crobinson@okd4-services ~]$ oc get csr -ojson | jq -r '.items[] | select(.status == {} ) | .metadata.name' | xargs oc adm certificate approve
certificatesigningrequest.certificates.k8s.io/csr-4lqrm approved
certificatesigningrequest.certificates.k8s.io/csr-c5mcj approved
[crobinson@okd4-services ~]$ oc get nodes
NAME      STATUS   ROLES   AGE   VERSION
okd4-compute-1  NotReady  worker  2s   v1.18.3
okd4-control-plane-1 Ready    master  20m  v1.18.3
okd4-control-plane-2 Ready    master  15m  v1.18.3
okd4-control-plane-3 Ready    master  13m  v1.18.3
[crobinson@okd4-services ~]$ oc get nodes
NAME      STATUS   ROLES   AGE   VERSION
okd4-compute-1  NotReady  worker  10s  v1.18.3
okd4-compute-2  NotReady  worker  8s   v1.18.3
okd4-control-plane-1 Ready    master  20m  v1.18.3
okd4-control-plane-2 Ready    master  16m  v1.18.3
okd4-control-plane-3 Ready    master  13m  v1.18.3
[crobinson@okd4-services ~]$ 
```

Check the status of the cluster operators.

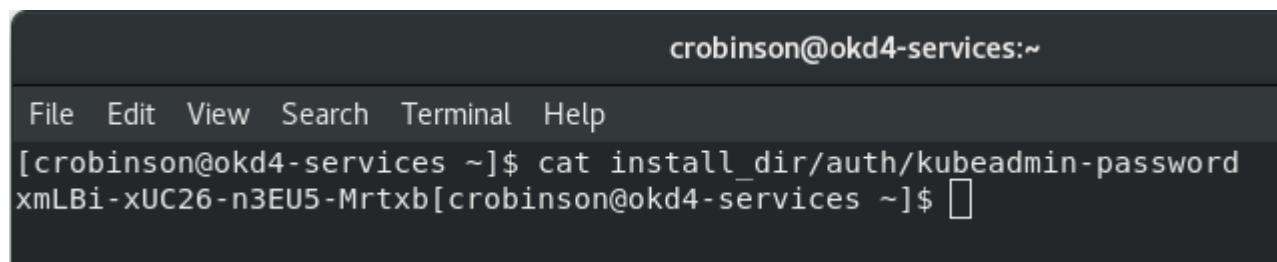
```
oc get clusteroperators
```

NAME	VERSION	AVAILABLE	PROGRESSING	DEGRADED	SINCE
authentication	4.5.0-0.okd-2020-06-29-110348-beta6	False	True	False	2m28s
cloud-credential	4.5.0-0.okd-2020-06-29-110348-beta6	True	False	False	29m
cluster-autoscaler	4.5.0-0.okd-2020-06-29-110348-beta6	True	False	False	9m58s
config-operator	4.5.0-0.okd-2020-06-29-110348-beta6	True	False	False	10m
console	4.5.0-0.okd-2020-06-29-110348-beta6	True	False	False	10s
csi-snapshot-controller	4.5.0-0.okd-2020-06-29-110348-beta6	True	False	False	2m32s
dns	4.5.0-0.okd-2020-06-29-110348-beta6	True	False	False	20m
etcd	4.5.0-0.okd-2020-06-29-110348-beta6	True	False	False	18m
image-registry	4.5.0-0.okd-2020-06-29-110348-beta6	True	False	False	11m

ingress	4.5.0-0.okd-2020-06-29-110348-beta6	True	False	False	2m51s
insights	4.5.0-0.okd-2020-06-29-110348-beta6	True	False	False	11m
kube-apiserver	4.5.0-0.okd-2020-06-29-110348-beta6	True	True	False	14m
kube-controller-manager	4.5.0-0.okd-2020-06-29-110348-beta6	True	False	False	20m
kube-scheduler	4.5.0-0.okd-2020-06-29-110348-beta6	True	False	False	20m
kube-storage-version-migrator	4.5.0-0.okd-2020-06-29-110348-beta6	True	False	False	3m
machine-api	4.5.0-0.okd-2020-06-29-110348-beta6	True	False	False	11m
machine approver	4.5.0-0.okd-2020-06-29-110348-beta6	True	False	False	16m
machine-config	4.5.0-0.okd-2020-06-29-110348-beta6	True	False	False	19m
marketplace	4.5.0-0.okd-2020-06-29-110348-beta6	True	False	False	10m
monitoring	4.5.0-0.okd-2020-06-29-110348-beta6	True	False	False	118s
network	4.5.0-0.okd-2020-06-29-110348-beta6	True	False	False	24m
node-tuning	4.5.0-0.okd-2020-06-29-110348-beta6	True	False	False	23m
openshift-apiserver	4.5.0-0.okd-2020-06-29-110348-beta6	True	False	False	12m
openshift-controller-manager	4.5.0-0.okd-2020-06-29-110348-beta6	True	False	False	11m
openshift-samples	4.5.0-0.okd-2020-06-29-110348-beta6	True	False	False	8m44s
operator-lifecycle-manager	4.5.0-0.okd-2020-06-29-110348-beta6	True	False	False	21m
operator-lifecycle-manager-catalog	4.5.0-0.okd-2020-06-29-110348-beta6	True	False	False	21m
operator-lifecycle-manager-packageserver	4.5.0-0.okd-2020-06-29-110348-beta6	True	False	False	15m
service-ca	4.5.0-0.okd-2020-06-29-110348-beta6	True	False	False	23m
storage	4.5.0-0.okd-2020-06-29-110348-beta6	True	False	False	11m

The console has just become available in my picture above. Get your kubeadmin password from the `install_dir/auth` folder and login to the web console:

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

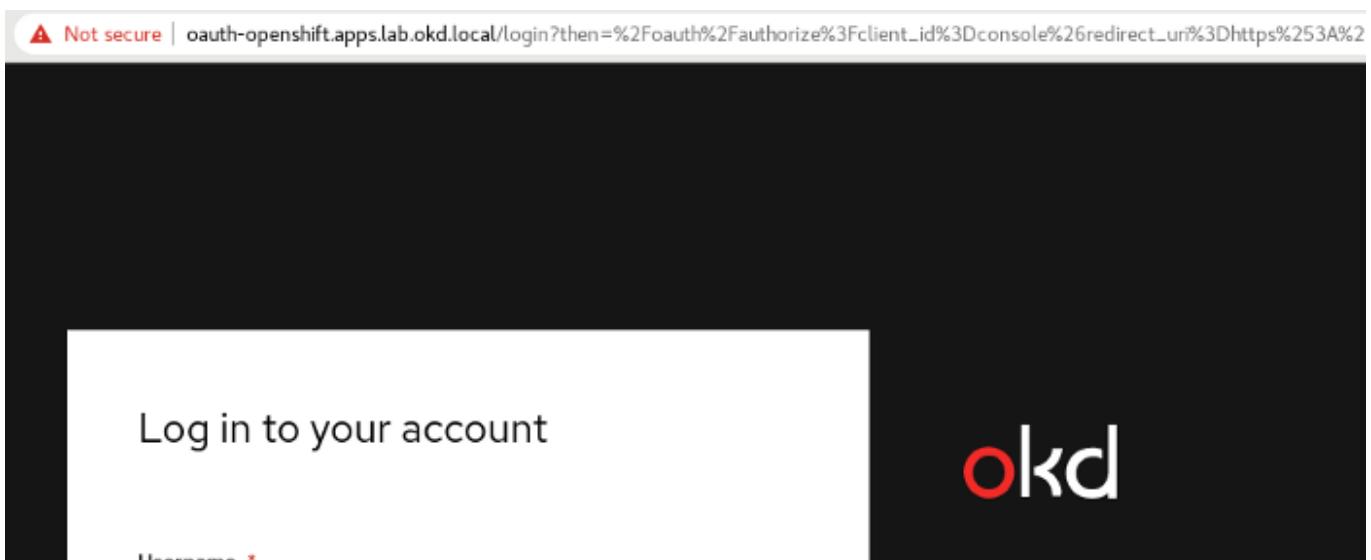


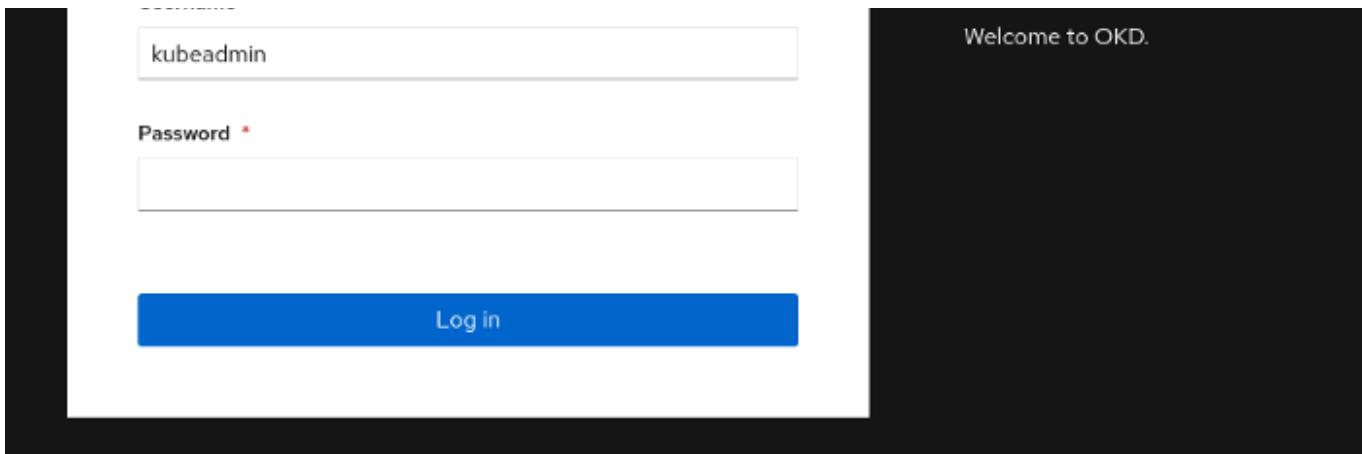
```
crobinson@okd4-services:~
```

```
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.

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.

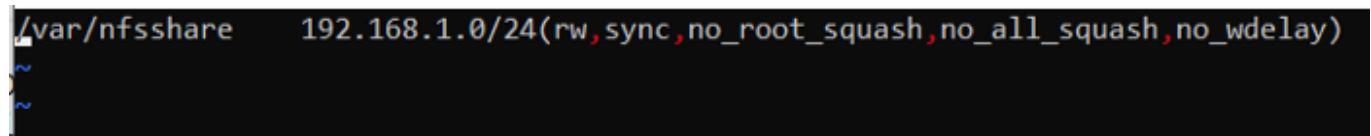
Login to your okd4-services VM and begin to set up an NFS server. The following commands install the necessary packages, enable services, and configure file and folder permissions.

```
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
192.168.1.0/24(rw,sync,no_root_squash,no_all_squash,no_wdelay)”

```
echo '/var/nfsshare
192.168.1.0/24(rw,sync,no_root_squash,no_all_squash,no_wdelay)' |
sudo tee /etc/exports
```



```
/var/nfsshare 192.168.1.0/24(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:
```

```

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 systems, so writing a database-enabled webpage with PHP is fairly simple. The most common use of PHP 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 source 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 one 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 OpenShift Topology interface. The left sidebar has a 'Topology' tab selected. The main area displays two circular icons representing pods: one for 'wordpress' (dark blue) and one for 'mariadb' (light blue). Below the icons, there are navigation buttons for search, refresh, and other actions.

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

The screenshot shows the OpenShift Topology interface with the 'wordpress' pod selected. The right panel displays detailed information about the pod, including its status as 'Running' and a 'View logs' button. It also shows build and service details for the 'wordpress' project.



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

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.

The screenshot shows a configuration form for a WordPress database. It includes fields for Database Name (wordpress), Username (wordpress), Password (wordpress), Database Host (mariadb), and Table Prefix (wp_). A 'Submit' button is at the bottom.

Database Name	wordpress	The name of the database you want to use with WordPress.
Username	wordpress	Your database username.
Password	wordpress	Your database password.
Database Host	mariadb	You should be able to get this info from your web host, if localhost doesn't work.
Table Prefix	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.

The screenshot shows the WordPress installation welcome screen. It displays the title 'Welcome' and a brief introduction: '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.'

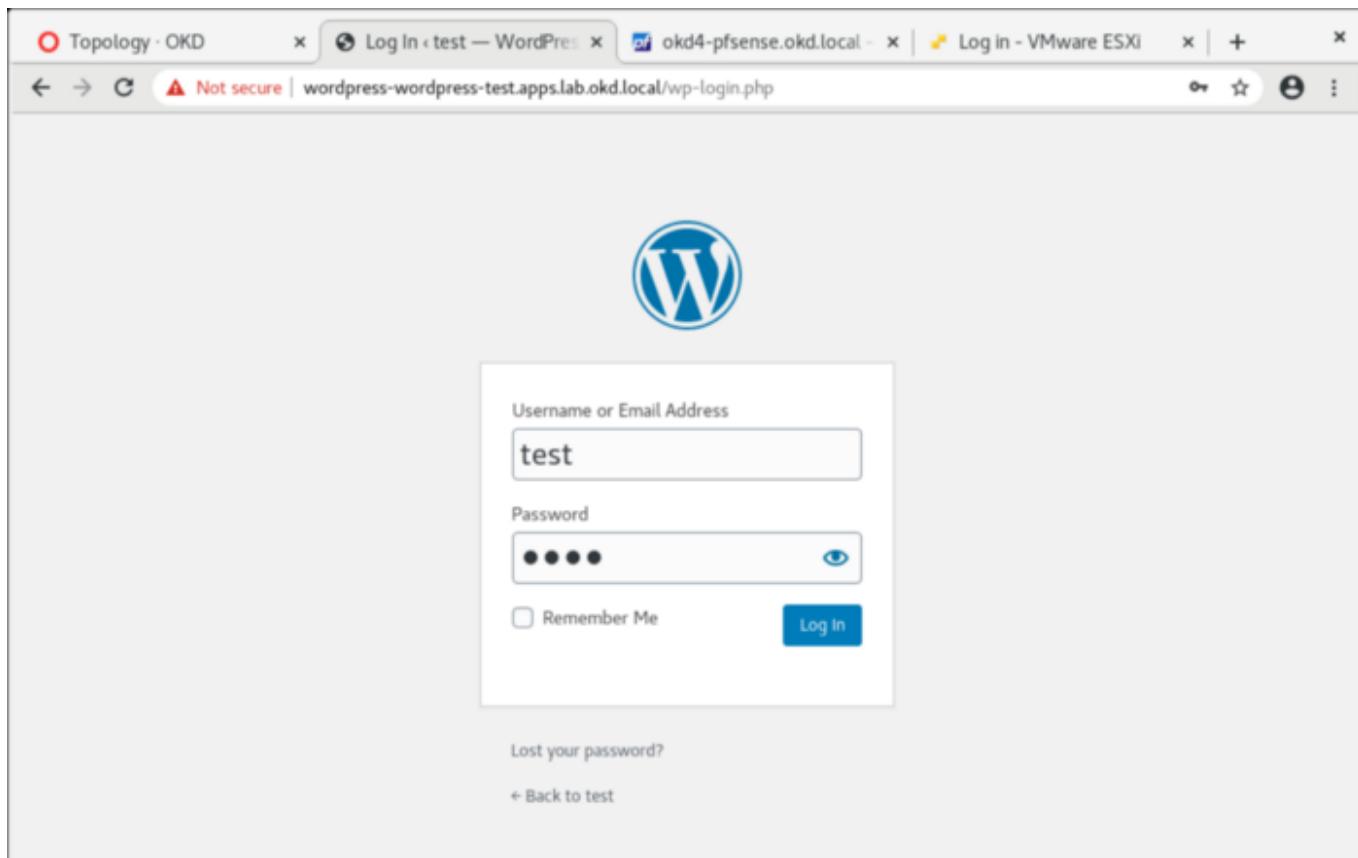
Information needed

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	Very weak (Show/Hide)	
Important: You will need this password to log in. Please store it in a secure location.			
Confirm Password	<input checked="" type="checkbox"/> Confirm use of weak password		
Your Email	test@test.com		Double-check your email address before continuing.
Search engine visibility	<input type="checkbox"/> Discourage search engines from indexing this site It is up to search engines to honor this request.		

Install WordPress

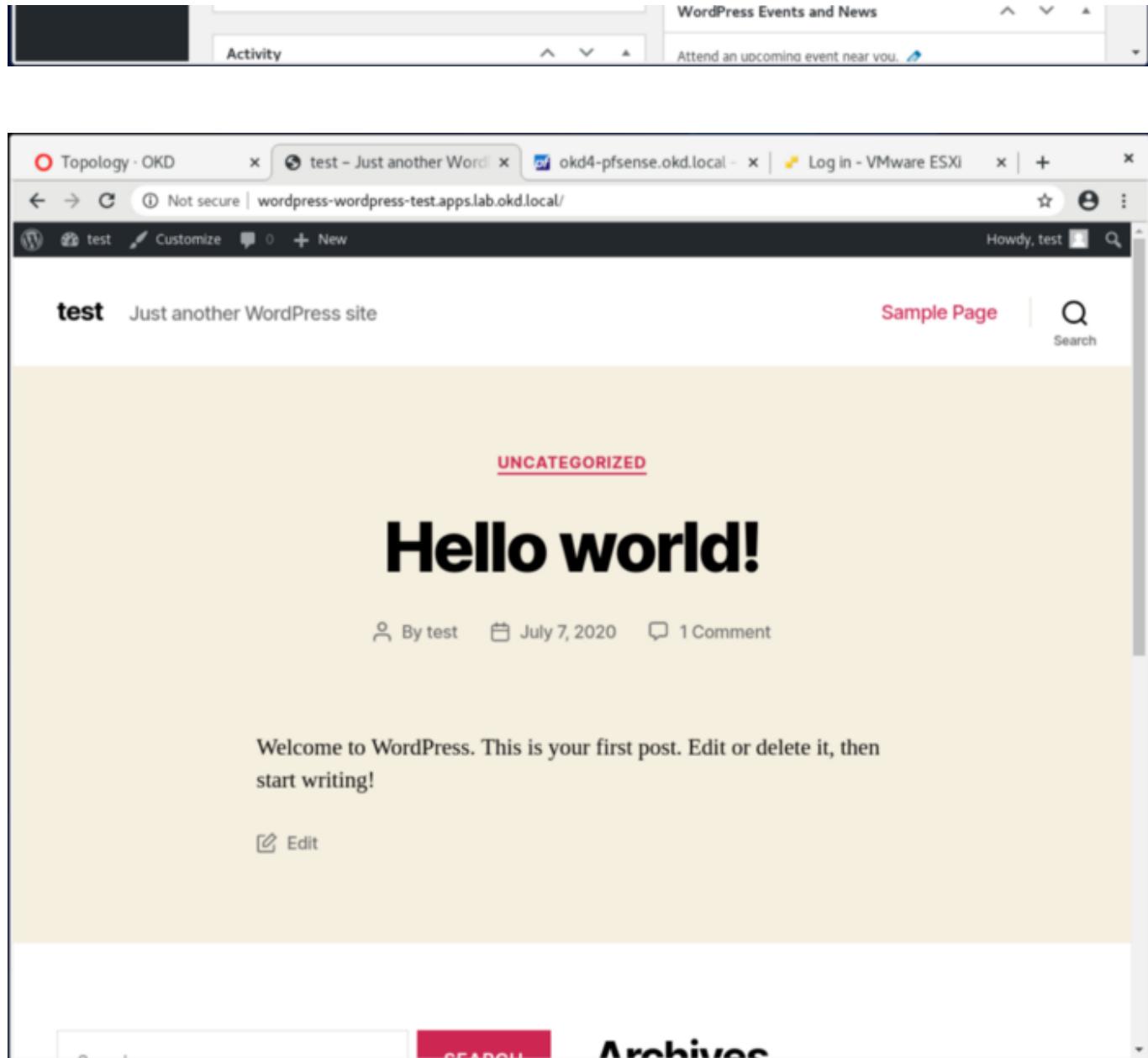
Log in, and you should have a working WordPress installation:



The screenshot shows a web browser window with four tabs open:

- Topology - OKD
- Dashboard < test — WordPress
- okd4-pfsense.okd.local -
- Log in - VMware ESXi

The active tab is "Dashboard < test — WordPress". The URL is "wordpress-wordpress-test.apps.lab.okd.local/wp-admin/". The page displays the WordPress dashboard. On the left is a sidebar with links: Home, Posts, Media, Pages, Comments, Appearance, Plugins, Users, Tools, Settings, and a Collapse menu. The main content area includes a "Welcome to WordPress!" message, "Get Started" options (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), "Site Health Status" (No information yet...), "At a Glance" (1 Post, 1 Page, 1 Comment), and a "Quick Draft" section with fields for Title, Content, and Save Draft.



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:

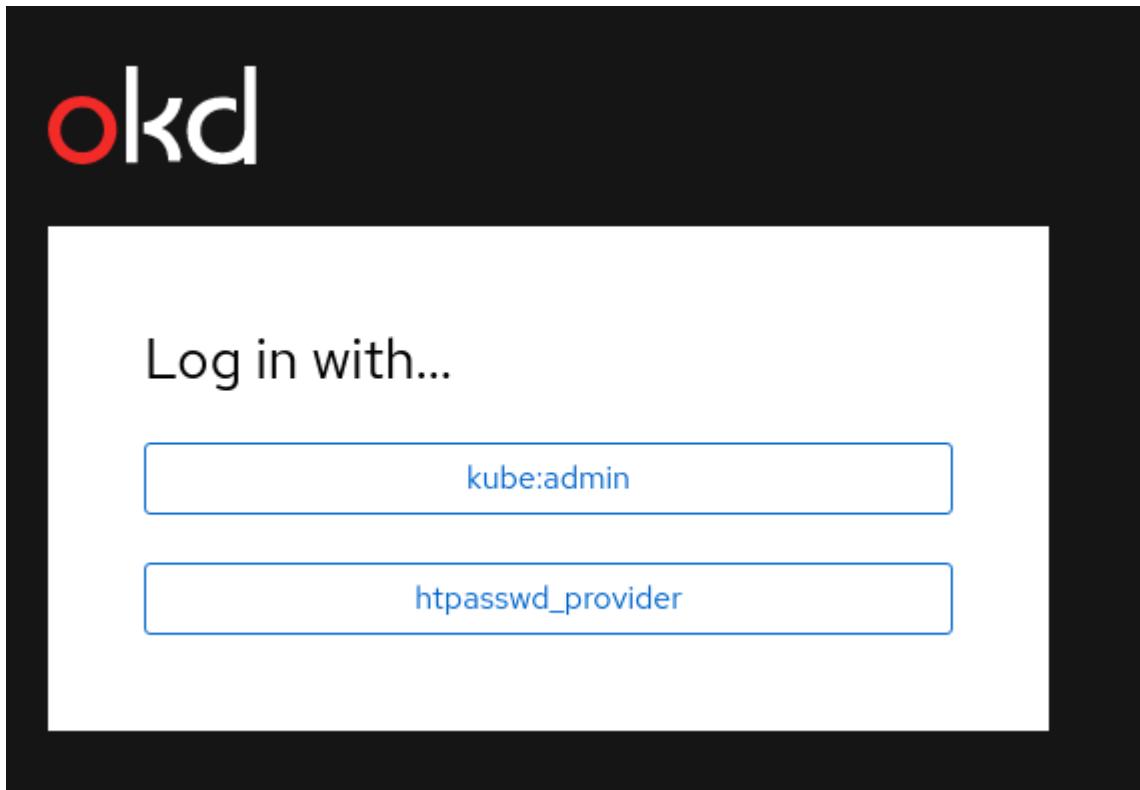
```
oc create secret generic htpass-secret --from-file=htpasswd=users.htpasswd -n openshift-config
```

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.



A screenshot of a web browser window. The address bar shows the URL: "Not secure | oauth-openshift.apps.lab.okd.local/login/htpasswd_provider?then=%2Foauth%2Fauthorize%3Fclient_id%3Dcons...". The main content area displays the OKD login interface. The OKD logo is at the top. Below it, the text "Log in to your account" is centered. There are two input fields: "Username *" with the value "testuser" and "Password *" with a redacted password. A large blue "Log in" button is at the bottom. At the very bottom of the page, the text "Welcome to OKD." is visible.

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

The screenshot shows a web browser window with the URL `okd4-pfsense.okd.local/k8s/cluster/projects`. The title bar says "Projects - OKD". The main content area is titled "Projects" and displays a message: "Welcome to OpenShift". It says "OpenShift helps you quickly develop, host, and scale applications. To get started, create a project for your application." Below this are links to "Documentation", "Download the command-line tools", and a "Create a new project" button. On the left, there is a sidebar menu under "Administrator" with options like Home, Projects, Search, Explore, Events, Operators, Workloads, Networking, Storage, Builds, User Management, and Administration.

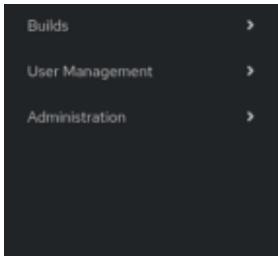
Give yourself cluster-admin access, and the projects should immediately populate:

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

Your user should now have cluster-admin level access:

The screenshot shows the same web browser window after the user has been granted cluster-admin privileges. The "Projects" list now contains several entries: default, kube-node-lease, kube-public, kube-system, openshift, and openshift-apiserver. Each entry includes columns for Name, Display Name, Status (all Active), Requester (No requester), and a small icon.

Name	Display Name	Status	Requester
default	No display name	Active	No requester
kube-node-lease	No display name	Active	No requester
kube-public	No display name	Active	No requester
kube-system	No display name	Active	No requester
openshift	No display name	Active	No requester
openshift-apiserver	No display name	Active	No requester



openshift-apiserver-operator	No display name	Active	No requester	
openshift-authentication	No display name	Active	No requester	
openshift-authentication-operator	No display name	Active	No requester	
openshift-cloud-credential-operator	No display name	Active	No requester	
openshift-cluster-machine approver	No display name	Active	No requester	

• • •

Congrats! You have created an OKD Cluster!

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

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 Kubernetes Homelab Okd Coreos

About Help Legal

Get the Medium app

A button that says 'Download on the App Store', and if clicked it will lead you to the iOS App store

A button that says 'Get it on, Google Play', and if clicked it will lead you to the Google Play store

