Table of Contents

OpenShift Installation Lab

Overview

- **Prepare to Deploy OpenShift Enterprise**

  In this section, you prepare hosts for installing OpenShift Enterprise, configure Domain Name System (DNS) and Network File System (NFS) servers on the administration host, configure the settings, and install Docker.

- **Install OpenShift Enterprise**

  In this section, you install OpenShift Enterprise with the Quick installer.

- **Configure and Set Up OpenShift**

  In this section, you configure OpenShift Enterprise: Label nodes, configure authentication, and deploy the registry and default router containers on the Infranode node host.

- **Set Up Persistent Storage**

  In this section, you prepare the OpenShift Cluster to use NFS storage as a Persistent Volume provider.

Notes

Read these notes. They help you successfully navigate the lab.

- You run many commands in this lab remotely from the `oselab` or `master00` host. The instructions and command-line prompts always specify the host from which to run the commands.
- Many steps require you to run a command on a host (usually `master00`) and then run the same command with a `for` loop on the rest of the nodes. When that occurs, examine the output of those commands to ensure that they completed correctly.
- This lab is long and extremely important. All future labs will depend on the success of your deployment.
- Depending on your workstation's screen size, resolution, and browser, some commands may lose their intended formatting, introducing extra line breaks. Even though we have endeavored to avoid this pitfall, the formatting still "breaks" a command sometimes. Pay attention to the command you paste in and correct the formatting, if necessary.

Known Issues

Issue	Description	Workaround
Failure of Docker storage	The Docker daemon cannot access the storage disks following the `docker-storage-setup` storage configuration. This problem sometimes occurs when the hosting platform reuses disks by accident.	Run `lvremove`, `vgremove`, and `pvremove` to free the `/dev/vdb` disks and then run `fdisk` to delete the `/dev/vdb1` partition. Subsequently, run `docker-storage-setup` again and restart the Docker daemon.
Failure of `Infranode00` Containers	The router and registry (containers on `infranode00`) stop responding or do not deploy—sometimes after an automatic shutdown of the virtual machine, sometimes at random.	Reboot `infranode00` and redeploy the containers.

The master daemon does not start after you have configured authentication.	This problem is most likely due to YAML's space sensitivity.	Revert to the original configuration (a copy is available from before) and try again. YAML is space sensitive and might take a couple of tries to work.

1. Prepare to Deploy OpenShift Enterprise

In this section, you deploy OpenShift Enterprise on a master and two nodes, as follows:

- Configure the Secure Shell (SSH) keys.
- Configure the repositories.
- Configure a DNS on the `oselab` server for your OpenShift Enterprise environment.
- Configure the network settings.
- Install Docker on all the hosts.

Reminder: The administration host is the only host you can access outside the lab environment with SSH. External SSH for all other hosts is blocked on all other hosts. Once on the administration host, you can access the other hosts internally through SSH. As described earlier, you must access the system (not `root`) with your private SSH key and OPENTLC login.

Each lab is assigned a global unique identifier (GUID) with four characters, which you receive by email while provisioning your lab environment. **From this point on, replace GUID with your lab's GUID.**

Your Environment

- The lab environment consists of the following five virtual machines (VMs):
  - `oselab-GUID.oslab.opentlc.com`: This is the administration host, which acts as a DNS server and an NFS server and host, from which you install the environment.
  - `master00-GUID.oslab.opentlc.com`: This is the master host, which contains `Etcd` and the management console.
  - `infranode00-GUID.oslab.opentlc.com`: This is the Infranode host, a regular node for running only the infrastructure containers (Registry and Router).
  - `node00-GUID.oslab.opentlc.com`: This is a node host (Region: primary, Zone: east).
  - `node01-GUID.oslab.opentlc.com`: This is another node host (Region: primary, Zone: west).
- In the labs in this section, use the `oselab` host as your DNS and NFS server. Run remote commands on the OpenShift environment on the provisioning and staging host.
- `oselab` is **not** an OpenShift Cluster member or part of the OpenShift environment. That host mimics your client's infrastructure or your laptop or desktop that is connected to the client's local area network (LAN).

Important Details

- Run most, **but not all**, of your commands from the `oselab` host.
- When executing instructions on all the nodes or hosts:
  - As a rule, run the commands on a specific server and examine the output.
  - Execute the commands on the rest of the nodes or hosts with a `for` loop to save time and effort.
  - In some cases, in the interest of time, feel free to run the commands directly on the nodes or hosts instead of using the `for` loop.
- The `$guid/$GUID` environment variables are already defined on all the hosts.
  - For the GUID variable in links or file definitions, replace GUID with its value.
  - Here is an administration host example:

```
[root@oselab-GUID ~]# command
```

  - Here is a master host example:

```
[root@master00-GUID ~]# command
```

In each step, ensure that you are running the step on the required host. Each step contains the host name. The example code contains the host name in the shell prompt.

Red Hat highly recommends that you use a terminal multiplexing tool, such as `tmux` or `screen`, which keeps your place in the session if you are disconnected from your environment. You can install packages after setting up the

`rhel` repositories.

To enter "scroll mode" in `tmux`, type **Ctrl+B**. Page up or down to scroll and use the **Esc** to exit scroll mode.

1.1. Connect to your environment

1. Connect to your administration host `oselab-GUID.oslab.opentlc.com`. Note that your private key location may vary.

```
yourdesktop$ ssh -i ~/.ssh/id_rsa your-opentlc-login@oselab-GUID.oslab.opentlc.com
```

   - Here is an example of a successful connection:

```
[sborenst@desktop01 ~]$ ssh -i ~/.ssh/id_rsa shacharb-redhat.com@oselab-c0fe.oslab.opentlc.com
################################################################################
################################################################################
################################################################################
Environment Deployment Is Completed : Wed Nov 25 20:03:55 EST 2015
################################################################################
################################################################################
################################################################################

-bash-4.2$
```

2. Run `sudo` to become the `root` user on the administration host:

```
-bash-4.2$ su - root
```

1.2. Configure SSH Keys

The OpenShift Enterprise installer configure hosts with SSH. In this section, you create and install an SSH key pair on the `oselab` host and add the public key to the `authorized_hosts` file on all the OpenShift hosts.

1. Create an SSH key pair for the `root` user and overwrite the existing key:

```
[root@oselab-GUID ~]# ssh-keygen -f /root/.ssh/id_rsa -N ''
```

In a different environment, you can adopt a nonroot user with `sudo` capabilities. For example, in Amazon Web Services (AWS), you adopt the `ec2-user` user.

2. On the `oselab` host, add the public SSH key locally to `/root/.ssh/authorized_keys`:

```
[root@oselab-GUID ~]# cat /root/.ssh/id_rsa.pub >> /root/.ssh/authorized_keys
```

3. Configure `/etc/ssh/ssh_conf` to disable `StrictHostKeyChecking` on the `oselab` host and the master host:

```
[root@oselab-GUID ~]# echo StrictHostKeyChecking no >> /etc/ssh/ssh_config
[root@oselab-GUID ~]# ssh master00-$guid "echo StrictHostKeyChecking no >> /etc/ssh/ssh_config"
```

This configuration saves you having to disable strict host-checking and to reply yes when running remote commands on unknown hosts. You will run many commands from both the `oselab` and `master00` hosts.

4. On the `oselab` host, test the new SSH key by connecting it to itself over the loopback interface without a keyboard prompt:

```
[root@oselab-GUID ~]# ssh 127.0.0.1
...[output omitted]...
[root@oselab-GUID ~]# exit
```

5. Copy the SSH key to the rest of the nodes in the environment. When prompted, specify the root password for each of the nodes.

```
[root@oselab-GUID ~]# for node in   master00-$GUID.oslab.opentlc.com \
                                    infranode00-$guid.oslab.opentlc.com \
                                    node00-$guid.oslab.opentlc.com \
                                    node01-$guid.oslab.opentlc.com; \
                                    do \
                                    ssh-copy-id root@$node ; \
                                    done
```

Remember: The default `root` password is `r3dh4t1!`.

1.3. Configure Repositories on All Hosts

OpenShift Enterprise requires four software repositories:

- `rhel-7-server-rpms`
- `rhel-7-server-extras-rpms`
- `rhel-7-server-optional-rpms`
- `rhel-7-server-ose-3.x-rpms`

Normally, you obtain those repositories through `subscription-manager`. For the sake of expediency, a mirror is available for you. Configure it as follows:

1. On the `oselab` host, set up the `yum` repository configuration file `/etc/yum.repos.d/open.repo` with the following repositories:

```
[root@oselab-GUID ~]# cat << EOF > /etc/yum.repos.d/open.repo
[rhel-x86_64-server-7]
name=Red Hat Enterprise Linux 7
baseurl=http://www.opentlc.com/repos/ose/3.1/rhel-7-server-rpms
enabled=1
gpgcheck=0

[rhel-x86_64-server-extras-7]
name=Red Hat Enterprise Linux 7 Extras
baseurl=http://www.opentlc.com/repos/ose/3.1/rhel-7-server-extras-rpms
enabled=1
gpgcheck=0

[rhel-x86_64-server-optional-7]
name=Red Hat Enterprise Linux 7 Optional
baseurl=http://www.opentlc.com/repos/ose/3.1/rhel-7-server-optional-rpms
enabled=1
gpgcheck=0

# This repo is added for the OPENTLC environment not OSE
[rhel-x86_64-server-rh-common-7]
name=Red Hat Enterprise Linux 7 Common
baseurl=http://www.opentlc.com/repos/ose/3.1/rhel-7-server-rh-common-rpms
enabled=1
gpgcheck=0

EOF
```

2. Add the OpenShift Enterprise repository mirror to the `oselab` host.

```
[root@oselab-GUID ~]# cat << EOF >> /etc/yum.repos.d/open.repo
[rhel-7-server-ose-3.1-rpms]
name=Red Hat Enterprise Linux 7 OSE 3.1
baseurl=http://www.opentlc.com/repos/ose/3.1/rhel-7-server-ose-3.1-rpms
enabled=1
gpgcheck=0

EOF
```

3. List the repositories on the `oselab` host:

```
[root@oselab-GUID ~]# yum clean all ; yum repolist
```

   - The output is as follows:

```
Loaded plugins: product-id
...[output omitted]...
repo id                                  repo name                                          status
rhel-7-server-ose-3.1-rpms               Red Hat Enterprise Linux 7 OSE 3                     323
rhel-x86_64-server-7                     Red Hat Enterprise Linux 7                         4,391
rhel-x86_64-server-extras-7              Red Hat Enterprise Linux 7 Extras                     45
rhel-x86_64-server-optional-7            Red Hat Enterprise Linux 7 Optional                4,220
rhel-x86_64-server-rh-common-7           Red Hat Enterprise Linux 7 Common                     19
repolist: 8,998

...[output omitted]...
```

4. Configure the master nodes by copying the `open.repo` file to all the nodes directly from the `oselab` host:

```
[root@oselab-GUID ~]# for node in master00-$guid.oslab.opentlc.com \
                                  infranode00-$guid.oslab.opentlc.com \
                                  node00-$guid.oslab.opentlc.com \
```

```
node01-$guid.oslab.opentlc.com; \
do \
  echo Copying open.repot to $node ; \
  scp /etc/yum.repos.d/open.repo ${node}:/etc/yum.repos.d/open.repo ;
  yum clean all
  yum repolist
done
```

1.4. Configure Wildcard DNS Entry on `oselab` for OpenShift

OpenShift Enterprise requires a wildcard DNS A record, which must point to the publicly available IP address of a node or nodes that are hosting the OpenShift default router container.

In the OpenShift environment, the OpenShift default router is deployed on the `infranode00` host.

1. Install the `bind` and `bind-utils` packages on the administration host:

```
[root@oselab-GUID ~]# yum -y install bind bind-utils
```

2. Verify that you have correctly configured the `$GUID` and `$guid` environment variables:

```
[root@oselab-GUID ~]# echo GUID is $GUID and guid is $GUID
```

   - The output is similar to this:

```
GUID is c0fe and guid is c0fe
```

   - If the environment variables `$GUID` and `$guid` **are not set**, run the following commands:

```
[root@oselab-GUID ~]# export GUID=`hostname|cut -f2 -d-|cut -f1 -d.`
[root@oselab-GUID ~]# export guid=`hostname|cut -f2 -d-|cut -f1 -d.`
```

3. On the administration host, `oselab`, collect and define the environment's information. Also, define the public IP address of `InfraNode00` as the target of the wildcard record:

```
[root@oselab-GUID ~]# host infranode00-$GUID.oslab.opentlc.com  ipa.opentlc.com |grep infranode | awk '{print $4}'
[root@oselab-GUID ~]# HostIP=`host infranode00-$GUID.oslab.opentlc.com  ipa.opentlc.com |grep infranode | awk '{print $4}'`
[root@oselab-GUID ~]# domain="cloudapps-$GUID.oslab.opentlc.com"
```

Perform the steps below on the administration host.

4. Create the zone file with the wildcard DNS:

```
[root@oselab-GUID ~]# mkdir /var/named/zones
[root@oselab-GUID ~]# echo "\$ORIGIN  .
\$TTL 1  ;  1 seconds (for testing only)
${domain} IN SOA master.${domain}.  root.${domain}.  (
  2011112904  ;  serial
  60  ;  refresh (1 minute)
  15  ;  retry (15 seconds)
  1800  ;  expire (30 minutes)
  10  ; minimum (10 seconds)
)
  NS master.${domain}.
\$ORIGIN ${domain}.
test A ${HostIP}
* A ${HostIP}"  >  /var/named/zones/${domain}.db
```

5. Configure `named.conf`:

```
[root@oselab-GUID ~]# echo "// named.conf
options {
  listen-on port 53 { any; };
  directory \"/var/named\";
  dump-file \"/var/named/data/cache_dump.db\";
  statistics-file \"/var/named/data/named_stats.txt\";
  memstatistics-file \"/var/named/data/named_mem_stats.txt\";
  allow-query { any; };
  recursion yes;
  /* Path to ISC DLV key */
  bindkeys-file \"/etc/named.iscdlv.key\";
};
```

```
logging {
  channel default_debug {
    file \"data/named.run\";
    severity dynamic;
  };
};
zone \"${domain}\" IN {
  type master;
  file \"zones/${domain}.db\";
  allow-update { key ${domain} ; } ;
};" > /etc/named.conf
```

6. Correct the file permissions and start the DNS server:

```
[root@oselab-GUID ~]#  chgrp named -R /var/named ; \
 chown named -R /var/named/zones ; \
 restorecon -R /var/named ; \
 chown root:named /etc/named.conf ; \
 restorecon /etc/named.conf ;
```

7. Enable and start `named`:

```
[root@oselab-GUID ~]# systemctl enable named ; \
 systemctl start named
```

8. Configure `firewalld` to allow inbound DNS queries:

```
[root@oselab-GUID bin]# firewall-cmd --zone=public --add-service=dns --permanent ; \
 firewall-cmd --reload
```

1.5. Verify DNS Configuration

A test DNS entry called `test.cloudapps-GUID.oslab.opentlc.com` is available.

1. Test the DNS server on the administration host:

```
[root@oselab-GUID ~]# host test.cloudapps-$GUID.oslab.opentlc.com 127.0.0.1
```

2. Test with an external name server:

```
[root@oselab-GUID ~]# host test.cloudapps-$GUID.oslab.opentlc.com 8.8.8.8
```

The first time you query `8.8.8.8`, you might notice some lag and see the error message `Connection timed out; trying next origin Host test.cloudapps-GUID.oslab.opentlc.com not found`
That phenomenon is normal. Rerunning the test results in faster performance and no errors.

3. Test DNS from your laptop or desktop. Be sure to replace GUID with the correct value. The update may take a few minutes.

```
Desktop$ nslookup test.cloudapps-$GUID.oslab.opentlc.com
```

1.6. Verify Network Configuration

Here, you verify that the master host is correctly configured for resolving internal and external DNS names.

1. Connect to the `master00` host:

```
[root@oselab-GUID ~]# ssh master00-$guid
```

2. Verify the host name for the master host:

```
[root@master00-GUID ~]# hostname -f
```

   - The output is as follows:

```
master00-GUID.oslab.opentlc.com
```

3. Take note of the master host's internal IP address:

```
[root@master00-GUID ~]# ip address show dev eth0|grep "inet "|awk '{print $2}'|cut -f1 -d/
```

4. Ensure that the master host's internal DNS entry matches the internal IP address:

```
[root@master00-GUID ~]# host `hostname -f`
```

5. Take note of the master host's external IP address:

```
[root@master00-GUID ~]# curl http://www.opentlc.com/getip
```

6. Ensure that the master host's external DNS entry matches the external IP address:

```
[root@master00-GUID ~]# host `hostname -f` 8.8.8.8
```

If errors occur on your first try, try again after a short while. It may take some time for the global DNS servers to update.

1.7. Install and Remove Packages

1. Back on the `oselab` host, run the following `for` loop to remove `NetworkManager` from the master and all the nodes:

```
[root@oselab-GUID ~]# for node in   master00-$guid.oslab.opentlc.com \
                                    infranode00-$guid.oslab.opentlc.com \
                                    node00-$guid.oslab.opentlc.com \
                                    node01-$guid.oslab.opentlc.com; \
                                    do \
                                    echo removing NetworkManager on $node ; \
                                      ssh $node "yum -y  remove NetworkManager*"
                                   done
```

You can configure `NetworkManager` so you need not remove it.

2. Install the following tools and utilities on the `master00` host:

```
[root@oselab-GUID ~]# ssh master00-$guid "yum -y install wget git net-tools bind-utils iptables-services bridge-utils python-
virtualenv gcc"
```

3. Install `bash-completion` on both the `oselab` host and the master host. This step is highly recommended.

```
[root@oselab-GUID ~]# yum -y install "bash-completion"
[root@oselab-GUID ~]# ssh master00-$guid "yum -y install bash-completion"
```

`bash-completion` becomes available for use only after you have restarted the `bash` shell.

4. Run `yum update` on the master and all the nodes:

```
[root@oselab-GUID ~]# for node in master00-$guid.oslab.opentlc.com \
                                  infranode00-$guid.oslab.opentlc.com \
                                  node00-$guid.oslab.opentlc.com \
                                  node01-$guid.oslab.opentlc.com; \
                                  do \
                                  echo Running yum update on $node ; \
                                  ssh $node "yum -y update " ; \
                                  done
```

1.8. Install Docker

OpenShift Enterprise stores and manages container images on Docker. Install Docker as follows:

1. Connect to the `master00` host:

```
[root@oselab-GUID ~]# ssh master00-$guid
```

2. Install the `docker` package on the master host:

```
[root@master00-GUID ~]# yum -y install docker
```

3. Configure the Docker registry on the master host to allow insecure (no-certificate) connections to the Docker registries within your network:

```
[root@master00-GUID ~]# sed -i "s/OPTIONS.*/OPTIONS='--selinux-enabled --insecure-registry 172.30.0.0\/16'/"
/etc/sysconfig/docker
```

4. Install the `docker` package on the other nodes:

```
[root@master00-GUID ~]# for node in   infranode00-$guid.oslab.opentlc.com \
                                  node00-$guid.oslab.opentlc.com \
                                  node01-$guid.oslab.opentlc.com; \
                                  do \
                                  echo Installing docker on $node ; \
                                  ssh $node "yum -y install docker"  ;
                                  done
```

The IP address of the Openshift default service network is `172.30.0.0`, which is in the above command line to deploy the local registry under that subnet.

5. Configure the Docker registry on the other nodes:

```
[root@master00-GUID ~]# for node in infranode00-$guid.oslab.opentlc.com \
                                  node00-$guid.oslab.opentlc.com \
                                  node01-$guid.oslab.opentlc.com; \
                                  do \
                                  echo Overwriting docker configuration file on $node ; \
                                  scp  /etc/sysconfig/docker $node:/etc/sysconfig/docker ;
                                  done
```

1.9. Configure Docker Storage

Next, configure the Docker storage pool.

The default configuration of loopback devices for the Docker storage does not support production. Red Hat considers the `dm.thinpooldev` storage option to be the only appropriate configuration for production.

1. Stop the Docker daemon and delete any files from `/var/lib/docker`:

```
[root@master00-GUID ~]# systemctl stop docker
[root@master00-GUID ~]# rm -rf /var/lib/docker/*
```

2. Do the same for the other nodes:

```
[root@master00-GUID ~]# for node in infranode00-$guid.oslab.opentlc.com \
                                  node00-$guid.oslab.opentlc.com \
                                  node01-$guid.oslab.opentlc.com; \
                                  do
                                  echo Cleaning up Docker on $node ; \
                                  ssh $node "systemctl stop docker ; rm -rf /var/lib/docker/*"  ;
                                  done
```

3. Specify the `/dev/vdb` hard drive as the Docker volume group for `docker-storage setup`:

```
[root@master00-GUID ~]# cat <<EOF > /etc/sysconfig/docker-storage-setup
DEVS=/dev/vdb
VG=docker-vg
EOF
```

4. Run `docker-storage-setup` on the `master00` host to create logical volumes for Docker:

```
[root@master00-GUID ~]#  docker-storage-setup
```

- The output is as follows:

```
Checking that no-one is using this disk right now ...
OK

Disk /dev/vdb: 20805 cylinders, 16 heads, 63 sectors/track
sfdisk:  /dev/vdb: unrecognized partition table type

Old situation:
sfdisk: No partitions found

New situation:
Units: sectors of 512 bytes, counting from 0

   Device Boot    Start       End   #sectors  Id  System
/dev/vdb1          2048  20971519   20969472  8e  Linux LVM
/dev/vdb2             0         -          0   0  Empty
/dev/vdb3             0         -          0   0  Empty
/dev/vdb4             0         -          0   0  Empty
```

```
  Warning: partition 1 does not start at a cylinder boundary
  Warning: partition 1 does not end at a cylinder boundary
  Warning: no primary partition is marked bootable (active)
  This does not matter for LILO, but the DOS MBR will not boot this disk.
  Successfully wrote the new partition table

  Re-reading the partition table ...

  If you created or changed a DOS partition, /dev/foo7, say, then use dd(1)
  to zero the first 512 bytes:  dd if=/dev/zero of=/dev/foo7 bs=512 count=1
  (See fdisk(8).)
    Physical volume "/dev/vdb1" successfully created
    Volume group "docker-vg" successfully created
    Rounding up size to full physical extent 12.00 MiB
    Logical volume "docker-poolmeta" created.
    Logical volume "docker-pool" created.
    WARNING: Converting logical volume docker-vg/docker-pool and docker-vg/docker-poolmeta to pool's data and metadata
  volumes.
    THIS WILL DESTROY CONTENT OF LOGICAL VOLUME (filesystem etc.)
    Converted docker-vg/docker-pool to thin pool.
    Logical volume "docker-pool" changed.
```

In a real environment, exercise caution when running `docker-storage-setup` because that command, by default, locates unused extents in the volume group (VG) that contain your root file system to create the pool. You can specify a VG or block device, but that can be a destructive process for the specified VG or block device. See the OpenShift documentation for details.

5. On the master host, examine the newly created logical volume `docker-pool`:

```
[root@master00-GUID ~]#  lvs
```

   - The output is as follows:

```
  LV          VG                  Attr       LSize  Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  docker-pool docker-vg           twi-a-t---  3.99g             0.00   0.29
  root        rhel_host2cc260760b15 -wi-ao---- 17.51g
  swap        rhel_host2cc260760b15 -wi-ao----  2.00g
```

6. On the master host, examine the configuration of `docker storage`:

```
[root@master00-GUID ~]# cat /etc/sysconfig/docker-storage
```

   - The output is as follows:

```
  DOCKER_STORAGE_OPTIONS=--storage-driver devicemapper --storage-opt dm.fs=xfs --storage-opt
  dm.thinpooldev=/dev/mapper/docker--vg-docker--pool
```

7. Run this `for` loop to configure docker storage on the other nodes, enable Docker, and restart the node:

```
[root@master00-GUID ~]# for node in infranode00-$guid.oslab.opentlc.com \
                                    node00-$guid.oslab.opentlc.com \
                                    node01-$guid.oslab.opentlc.com; \
                                    do
                                      echo Configuring Docker Storage and rebooting $node
                                      scp /etc/sysconfig/docker-storage-setup ${node}:/etc/sysconfig/docker-storage-setup
                                      ssh $node "
                                            docker-storage-setup ;
                                            systemctl enable docker;
                                            reboot"
                                    done
```

`Broken Pipeline` messages in the output are normal and not an indication of errors.

8. Enable Docker service on the master host:

```
[root@master00-GUID ~]# systemctl enable docker
```

9. Reboot the master host:

```
[root@master00-GUID ~]# reboot
```

See the *[Known Issues]* section if you have problems with Docker's storage setup.

1.10. Populate Local Docker Registry

1. Log back in to the `osehost` host after rebooting the nodes and the master.
2. Verify that the Docker service has started on all the nodes, you might have to wait for hosts to finish rebooting and for the docker daemon to start:

```
[root@oselab-GUID ~]# for node in   master00-$guid.oslab.opentlc.com \
                                    infranode00-$guid.oslab.opentlc.com \
                                    node00-$guid.oslab.opentlc.com \
                                    node01-$guid.oslab.opentlc.com; \
                                    do
                                      echo Checking docker status on $node
                                      ssh $node "
                                            systemctl status docker | grep Active"
                                    done
```

   - The output is as follows:

```
Checking docker status on master00-c0fe.oslab.opentlc.com
   Active: active (running) since Thu 2015-11-26 01:03:14 EST; 2min 24s ago
Checking docker status on infranode00-c0fe.oslab.opentlc.com
   Active: active (running) since Thu 2015-11-26 01:02:15 EST; 3min 24s ago
Checking docker status on node00-c0fe.oslab.opentlc.com
   Active: active (running) since Thu 2015-11-26 01:02:17 EST; 3min 23s ago
Checking docker status on node01-c0fe.oslab.opentlc.com
   Active: active (running) since Thu 2015-11-26 01:02:20 EST; 3min 21s ago
```

Ensure the status is `enabled` and `active (running)`.

3. On the `oselab` host, pull down the Docker images to **all the nodes** in the primary region (`node00` and `node01`):

```
[root@oselab-GUID ~]# REGISTRY="registry.access.redhat.com";PTH="openshift3"
[root@oselab-GUID ~]# for node in  node00-$guid.oslab.opentlc.com \
                                   node01-$guid.oslab.opentlc.com; \
do
ssh $node "
docker pull $REGISTRY/$PTH/ose-deployer:v3.1.0.4 ; \
docker pull $REGISTRY/$PTH/ose-sti-builder:v3.1.0.4 ; \
docker pull $REGISTRY/$PTH/ose-pod:v3.1.0.4 ; \
docker pull $REGISTRY/$PTH/ose-keepalived-ipfailover:v3.1.0.4 ; \
docker pull $REGISTRY/$PTH/ruby-20-rhel7 ; \
docker pull $REGISTRY/$PTH/mysql-55-rhel7 ; \
docker pull openshift/hello-openshift:v1.0.6 ;
"
done
```

You are downloading these images to save time later. Unless otherwise configured, if a node does not have a local image, it downloads it.

This process takes about 10 minutes to complete on **each node**. For the sake of efficiency, do not wait for the process to complete. Just connect to each node, run `pull`, and continue with the other tasks.

4. On `oselab`, pull only the basic images and the registry and router images to the `Infranode00` host:

```
[root@oselab-GUID ~]# REGISTRY="registry.access.redhat.com";PTH="openshift3"
[root@oselab-GUID ~]# node=infranode00-$guid.oslab.opentlc.com
[root@oselab-GUID ~]# ssh $node "
docker pull $REGISTRY/$PTH/ose-haproxy-router:v3.1.0.4  ; \
docker pull $REGISTRY/$PTH/ose-deployer:v3.1.0.4 ; \
docker pull $REGISTRY/$PTH/ose-pod:v3.1.0.4 ; \
docker pull $REGISTRY/$PTH/ose-docker-registry:v3.1.0.4 ;
"
```

We aren't "pulling" any images on the Master host because it is not meant to run any containers.

```
. Examine the information in the Docker pool on the `node0X` (`node00`, `node01`, etc.) host:
+
```

[root@oselab-GUID ~]# ssh node00-$guid docker info

```
* The output is as follows:
+
```

Containers: 0 Images: 15 Storage Driver: devicemapper Pool Name: docker—vg-docker—pool Pool Blocksize: 524.3 kB Backing Filesystem: xfs Data file: Metadata file: Data Space Used: 1.481 GB Data Space Total: 10.72 GB Data Space Available: 9.24 GB Metadata Space Used: 323.6 kB Metadata Space Total: 29.36 MB Metadata Space Available: 29.04 MB Udev Sync Supported: true Deferred Removal Enabled: false Library Version: 1.02.93-RHEL7 (2015-01-28) Execution Driver: native-0.2 Logging Driver: json-file Kernel Version: 3.10.0-229.el7.x86_64 Operating System: Red Hat Enterprise Linux Server 7.1 (Maipo) CPUs: 2 Total Memory: 1.797 GiB Name: node00-c0fe.oslab.opentlc.com ID: RXVI:JKOO:3U4X:LHDE:QXPN:FSQC:TTBL:UCWP:MCEH:2KU6:GWSD:IRIN …

```
. On the `node0X` host, examine the `docker-pool` logical volume again:
+
```

[root@oselab-GUID ~]# ssh node00-$guid.oslab.opentlc.com "lvs"

```
* The output is similar to below. Note that the `docker-pool` LV now contains data.
+
```

LV VG Attr LSize Pool Origin Data% Meta% Move Log Cpy%Sync Convert docker-pool docker-vg twi-a-t--- 9.98g 13.81 1.10 root rhel_host2cc260760b15 -wi-ao---- 17.51g swap rhel_host2cc260760b15 -wi-ao---- 2.00g

```
== Install OpenShift Enterprise

In this section, you download and install the installer and then verify your environment.

=== Download Installer

In this lab, you run the Installer from the `oselab` host, which, in a real-world scenario, could be a laptop or a staging or
provisioning server. No packages are deployed directly from `oselab` to the OpenShift nodes or master.

. On the `oselab` host, install the OpenShift utility package:
+
```

[root@oselab-GUID ~]# yum -y install atomic-openshift-utils

```
. *(Optional)* Copy and paste the master and node names to a local file on your machine to save time and effort copying and
pasting them in later steps.
+
```

[root@oselab-GUID ~]# for node in master00-$guid.oslab.opentlc.com \ infranode00-$guid.oslab.opentlc.com \ node00-$guid.oslab.opentlc.com \ node01-$guid.oslab.opentlc.com; do echo $node ; done

```
=== Run Installer

. Execute the installation utility to interactively configure one or more hosts:
+
```

[root@oselab-GUID ~]# atomic-openshift-installer install

```
. Follow the instructions of the Installer:
+
```

Welcome to the OpenShift Enterprise 3 installation.

Please confirm that following prerequisites have been met:

- All systems where OpenShift will be installed are running Red Hat Enterprise Linux 7.
- All systems are properly subscribed to the required OpenShift Enterprise 3 repositories.
- All systems have run docker-storage-setup (part of the Red Hat docker RPM).
- All systems have working DNS that resolves not only from the perspective of the installer but also from within the cluster.

When the process completes you will have a default configuration for Masters and Nodes. For ongoing environment maintenance it's recommended that the official Ansible playbooks be used.

For more information on installation prerequisites please see:
https://docs.openshift.com/enterprise/latest/admin_guide/install/prerequisites.html

Are you ready to continue? [y/N]:

```
. Type `y`. The output is as follows. Type `root` at the prompt `User for ssh access [root]`.
+
```

This installation process will involve connecting to remote hosts via ssh. Any account may be used however if a non-root account is used it must have passwordless sudo access.

User for ssh access [root]: root

```
+
CAUTION: Pay attention while typing your input. If you make a mistake, type *Ctrl+C*
 to exit the Installer and try again.

* The output is as follows:
+
```

Host Configuration

The OpenShift Master serves the API and web console. It also coordinates the jobs that have to run across the environment. It can even run the datastore. For wizard based installations the database will be embedded. It's possible to change this later using etcd from Red Hat Enterprise Linux 7.

Any Masters configured as part of this installation process will also be configured as Nodes. This is so that the Master will be able to proxy to Pods from the API. By default this Node will be unscheduleable but this can be changed after installation with *oadm manage-node*.

The OpenShift Node provides the runtime environments for containers. It will host the required services to be managed by the Master.

http://docs.openshift.com/enterprise/latest/architecture/infrastructure_components/kubernetes_infrastructure.html#master
http://docs.openshift.com/enterprise/latest/architecture/infrastructure_components/kubernetes_infrastructure.html#node

Enter hostname or IP address: []: master00-3191.oslab.opentlc.com Will this host be an OpenShift Master? [y/N]: y Do you want to add additional hosts? [y/N]: y Enter hostname or IP address: []: infranode00-3191.oslab.opentlc.com Do you want to add additional hosts? [y/N]: y Enter hostname or IP address: []: node00-3191.oslab.opentlc.com Do you want to add additional hosts? [y/N]: y Enter hostname or IP address: []: node01-3191.oslab.opentlc.com Do you want to add additional hosts? [y/N]: n

```
+
. Answer the questions at the bottom as follows:
.. Paste or type the name of your master host `master00-GUID.oslab.opentlc.com`.
.. Type `y` to confirm that this host is an OpenShift Master.
.. Type `y` to add more hosts.
.. Paste or type the name of your `infra` host `infranode00-GUID.oslab.opentlc.com`.
.. Type `y` to add more hosts.
.. Paste or type the name of your `node00` host `node00-GUID.oslab.opentlc.com`.
.. Type `y` to add more hosts.
.. Paste or type the name of your `node01` host `node01-GUID.oslab.opentlc.com`
.. Type `n` to stop adding OpenShift hosts.

. Type `2` to choose OpenShift Enterprise 3.1 as the variant:
+
```

Which variant would you like to install?

(1) OpenShift Enterprise 3.0 (2) OpenShift Enterprise 3.1 (3) Atomic Enterprise Platform 3.1 Choose a variant from above: [1]: 2

```
* The Installer then collects information on your environment and displays the following:
+
```

Gathering information from hosts… …This might take a few minutes… A list of the facts gathered from the provided hosts follows. Because it is often the case that the hostname for a system inside the cluster is different from the hostname that is resolveable from command line or web clients these settings cannot be validated automatically.

For some cloud providers the installer is able to gather metadata exposed in the instance so reasonable defaults will be provided.

Please confirm that they are correct before moving forward.

master00-GUID.oslab.opentlc.com,192.168.0.100,192.168.0.100,master00-GUID.oslab.opentlc.com,master00-GUID.oslab.opentlc.com infranode00-GUID.oslab.opentlc.com,192.168.0.101,192.168.0.101,infranode00-GUID.oslab.opentlc.com,infranode00-GUID.oslab.opentlc.com node00-GUID.oslab.opentlc.com,192.168.0.200,192.168.0.200,node00-GUID.oslab.opentlc.com,node00-GUID.oslab.opentlc.com node01-GUID.oslab.opentlc.com,192.168.0.201,192.168.0.201,node01-GUID.oslab.opentlc.com,node01-GUID.oslab.opentlc.com

Format:

connect_to,IP,public IP,hostname,public hostname

Notes: * The installation host is the hostname from the installer's perspective. * The IP of the host should be the internal IP of the

instance. * The public IP should be the externally accessible IP associated with the instance * The hostname should resolve to the internal IP from the instances themselves. * The public hostname should resolve to the external ip from hosts outside of the cloud.

Do the above facts look correct? [y/N]: y Ready to run installation process. If changes are needed to the values recorded by the installer please update /root/.config/openshift/installer.cfg.yml. Are you ready to continue? [y/N]: y

```
. Type `y` to confirm the collected facts and,  after the location of the configuration file is displayed, type `y` again to
continue.

+
. Watch the Installer run:
+
```

PLAY RECAP ** infranode00-GUID.oslab.opentlc.com : ok=58 changed=22 unreachable=0 failed=0 localhost : ok=11 changed=0 unreachable=0 failed=0 master00-GUID.oslab.opentlc.com : ok=206 changed=58 unreachable=0 failed=0 node00-GUID.oslab.opentlc.com : ok=58 changed=22 unreachable=0 failed=0 node01-GUID.oslab.opentlc.com : ok=58 changed=22 unreachable=0 failed=0

The installation was successful!

If this is your first time installing please take a look at the Administrator Guide for advanced options related to routing, storage, authentication and much more:

http://docs.openshift.com/enterprise/latest/admin_guide/overview.html

Press any key to continue .

```
. Read the install configuration file below. Note that you could have created this file instead of stepping through the
interactive setup.
+
```

[root@master00-GUID ~]# cat /root/.config/openshift/installer.cfg.yml ansible_config: /usr/share/atomic-openshift-utils/ansible.cfg ansible_log_path: /tmp/ansible.log ansible_ssh_user: root hosts: - connect_to: master00-GUID.oslab.opentlc.com hostname: master00-GUID.oslab.opentlc.com ip: 192.168.0.100 master: true node: true public_hostname: master00-GUID.oslab.opentlc.com public_ip: 192.168.0.100 - connect_to: infranode00-GUID.oslab.opentlc.com hostname: infranode00-GUID.oslab.opentlc.com ip: 192.168.0.101 node: true public_hostname: infranode00-GUID.oslab.opentlc.com public_ip: 192.168.0.101 - connect_to: node00-GUID.oslab.opentlc.com hostname: node00-GUID.oslab.opentlc.com ip: 192.168.0.200 node: true public_hostname: node00-GUID.oslab.opentlc.com public_ip: 192.168.0.200 - connect_to: node01-GUID.oslab.opentlc.com hostname: node01-GUID.oslab.opentlc.com ip: 192.168.0.201 node: true public_hostname: node01-GUID.oslab.opentlc.com public_ip: 192.168.0.201 variant: openshift-enterprise variant_version: *3.1* version: v1

```
. Restart the master and all the nodes:
+
```

[root@oselab-GUID ~]# for node in master00-$guid.oslab.opentlc.com \ infranode00-$guid.oslab.opentlc.com \ node00-$guid.oslab.opentlc.com \ node01-$guid.oslab.opentlc.com; \ do \ echo Rebooting $node ; \ ssh $node "reboot" done

```
+
NOTE: `Broken Pipeline` messages in the output are normal and not an indication
 of errors.

=== Verify Your Environment

. After the hosts finish rebooting, connect to the `master00` host:
+
```

[root@oselab-GUID ~]# ssh master00-$guid

```
. Run `oc get nodes` to check the status of your hosts:
+
```

[root@master-GUID ~]# oc get nodes NAME LABELS STATUS AGE infranode00-GUID.oslab.opentlc.com kubernetes.io/hostname=192.168.0.101 Ready 1m master00-GUID.oslab.opentlc.com kubernetes.io/hostname=192.168.0.100 Ready,SchedulingDisabled 1m node00-GUID.oslab.opentlc.com kubernetes.io/hostname=192.168.0.200 Ready 1m node01-GUID.oslab.opentlc.com kubernetes.io/hostname=192.168.0.201 Ready 1m

```
+
NOTE: If you see an error message that connection to the master host cannot be established, wait a few more seconds for the master
daemon to start.

. Use your browser to connect to the OpenShift web console at link:https://master00-GUID.oslab.opentlc.com:8443[`https://master00-
GUID.oslab.opentlc.com:8443`] and accept the Untrusted Certificate.
```

```
NOTE: You cannot log in yet because you have yet to set up authentication.

== Configure and Set Up OpenShift Enterprise

In this section, you establish regions and zones, configure OpenShift
 Enterprise, set up authentication, add development users, and configure
  `htpasswd` authentication. Subsequently, you deploy the registry and router
   and populate OpenShift Enterprise.

=== Set Regions and Zones

. Label the nodes:
+
```

[root@master00-GUID]# oc label node infranode00-$GUID.oslab.opentlc.com region="infra" zone="infranodes" [root@master00-GUID]# oc label node node00-$GUID.oslab.opentlc.com region="primary" zone="east" [root@master00-GUID ~]# oc label node node01-$GUID.oslab.opentlc.com region="primary" zone="west"

```
. On the `master00` host, run `oc get nodes` to learn how the labels were
 implemented:
+
```

[root@oselab-GUID ~]# oc get nodes

```
* The output is as follows:
+
```

NAME LABELS STATUS AGE infranode00-GUID.oslab.opentlc.com kubernetes.io/hostname=192.168.0.101,region=infra,zone=infranodes Ready 6m master00-GUID.oslab.opentlc.com kubernetes.io/hostname=192.168.0.100 Ready,SchedulingDisabled 6m node00-GUID.oslab.opentlc.com kubernetes.io/hostname=192.168.0.200,region=primary,zone=east Ready 6m node01-GUID.oslab.opentlc.com kubernetes.io/hostname=192.168.0.201,region=primary,zone=west Ready 6m

```
You now have a running OpenShift Enterprise environment across three hosts with
 one master and three nodes, divided into two regions: infra and primary.

=== OpenShift Enterprise Configuration Tips

. Create a copy of your master's configuration file:
+
```

[root@master00-GUID ~]# cp /etc/origin/master/master-config.yaml /etc/origin/master/master-config.yaml.original

```
.Setting the Default Subdomain
* To set a default Route, change the `routingConfig` attribute's `subdomain` command:
+
[source,bash]
```

[root@master00-GUID]# sed -i "s/subdomain: \"\"/subdomain: \"cloudapps-${guid}.oslab.opentlc.com\"/g" /etc/origin/master/master-config.yaml [root@master00-GUID]# systemctl restart atomic-openshift-master

```
.Setting the Default NodeSelector
* To set a default `NodeSelector`, change the `projectConfig` attribute's `defaultNodeSelector` command:
+
[source,bash]
```

[root@master00-GUID]# sed -i *s/defaultNodeSelector: ""/defaultNodeSelector: "region=primary"*/ /etc/origin/master/master-config.yaml [root@master00-GUID]# systemctl restart atomic-openshift-master [root@master00-GUID ~]# systemctl status atomic-openshift-master tomic-openshift-master.service - Atomic OpenShift Master Loaded: loaded (/usr/lib/systemd/system/atomic-openshift-master.service; enabled) Active: active (running) since Mon 2015-12-21 20:19:58 EST; 19s ago Docs: https://github.com/openshift/origin Main PID: 2948 (openshift) CGroup: /system.slice/atomic-openshift-master.service └─2948 /usr/bin/openshift start master --config=/etc/origin/master/master-config.yaml --loglevel=2

```
.Configuring the Default Namespace to Use the `infra` Region

. In annotations section, add the following line in the default namespace object:
+
```

openshift.io/node-selector: region=infra

```
. Edit the default namespace with the following command. To exit, type `:wq`.
```

```
+
```

[root@master00-GUID ~]# oc edit namespace default

```
* Your object looks similar to this:
+
```

apiVersion: v1 kind: Namespace metadata: annotations: openshift.io/node-selector: region=infra openshift.io/sa.initialized-roles: "true" openshift.io/sa.scc.mcs: s0:c3,c2 openshift.io/sa.scc.supplemental-groups: 1000010000/10000 openshift.io/sa.scc.uid-range: 1000010000/10000 creationTimestamp: 2015-11-20T02:10:35Z name: default resourceVersion: "217" selfLink: /api/v1/namespaces/default uid: e304c204-8f2b-11e5-9223-2cc260072896 spec: finalizers: - kubernetes - openshift.io/origin status: phase: Active

```
. Check that your changes were updated in the "default" `namespace`.
+
```

[root@master00-GUID ~]# oc get namespace default -o yaml apiVersion: v1 kind: Namespace metadata: annotations: openshift.io/node-selector: region=infra openshift.io/sa.initialized-roles: "true" openshift.io/sa.scc.mcs: s0:c5,c0 openshift.io/sa.scc.supplemental-groups: 1000020000/10000 openshift.io/sa.scc.uid-range: 1000020000/10000 creationTimestamp: 2015-12-22T01:00:48Z name: default resourceVersion: "752" selfLink: /api/v1/namespaces/default uid: 70779bd0-a847-11e5-b12e-2cc2605128d8 spec: finalizers: - kubernetes - openshift.io/origin status: phase: Active

```
.Setting Up Processes for Logs (Reference Only)
* Because the `systemd` and `journal` commands are for browsing logs in Red Hat Enterprise Linux 7, do not browse them with
`/var/log/messages`. Run `journalctl` instead.

* Given that Red Hat Enterprise Linux 7 runs all components in higher log levels, Red Hat recommends that you set up windows for
each process in your terminal emulator. That is, on the master host, run each of the following command lines in its own window:
+
```

[root@master00-GUID]# journalctl -f -u atomic-openshift-master [root@master00-GUID]# journalctl -f -u atomic-openshift-node

```
[NOTE]
To run the above commands on the other nodes, you do not need the `atomic-openshift-master` service. You may also want to watch
the Docker logs.

=== Configure Authentication

CAUTION: The commands in this section are case-sensitive. If you are new to YAML, it may take a few tries for the configuration
file to parse correctly.

. Create another copy of your master's configuration file:
+
```

[root@master00-GUID ~]# cp /etc/origin/master/master-config.yaml /etc/origin/master/master-config.yaml.preauth.original

```
. Edit the `/etc/origin/master/master-config.yaml` file so that the `oauthConfig` section reads as follows:
+
```

oauthConfig: assetPublicURL: https://master00-GUID.oslab.opentlc.com:8443/console/ grantConfig: method: auto identityProviders: - name: htpasswd_auth challenge: true login: true provider: apiVersion: v1 kind: HTPasswdPasswordIdentityProvider file: /etc/origin/openshift-passwd masterPublicURL: https://master00-GUID.oslab.opentlc.com:8443 masterURL: https://master00-GUID.oslab.opentlc.com:8443 sessionConfig: sessionMaxAgeSeconds: 3600 sessionName: ssn sessionSecretsFile: tokenConfig: accessTokenMaxAgeSeconds: 86400 authorizeTokenMaxAgeSeconds: 500

```
. Run `sed` to replace GUID with its actual value.
+
```

[root@master00-GUID ~]# sed -i s/GUID/${guid}/g /etc/origin/master/master-config.yaml

```
=== Add Development Users

Real-world developers are likely to use the OpenShift Enterprise tools (`oc` and the web console) on their own machines. In this
course, you create accounts for two nonprivileged OpenShift Enterprise users, `andrew` and `marina`, on the master.

On the master host, add two Linux accounts:
```

[root@master00-GUID]# useradd andrew [root@master00-GUID]# useradd marina

```
NOTE: Feel free to create those users on any machine in which the `oc` command is available. The master's API port (8443) is
```

```
available to the public network.

=== Configure `htpasswd` Authentication

OpenShift Enterprise 3 supports several authentication mechanisms. The simplest use case for testing is `htpasswd`-based
authentication.

As a preliminary requirement, you need the `htpasswd` binary in the `httpd-tools` package. Do the following:

. Install `httpd-tools` on the master host:
+
```

[root@master00-GUID ~]# yum -y install httpd-tools

```
. Create a password for users `andrew` and `marina` on the master host:
+
```

[root@master00-GUID]# htpasswd -cb /etc/origin/openshift-passwd andrew r3dh4t1! [root@master00-GUID]# htpasswd -b /etc/origin/openshift-passwd marina r3dh4t1!

```
. Restart `atomic-openshift-master` for the changes to take effect:
+
```

[root@master00-GUID ~]# systemctl restart atomic-openshift-master

```
. Check the status of your `atomic-openshift-master` daemon:
+
```

[root@master00-GUID ~]# systemctl status atomic-openshift-master

```
. Verify that you can authenticate as `andrew` in the OpenShift web console:
.. Connect to `https://master00-GUID.oslab.opentlc.com:8443/`.
.. Log in as `andrew` with the password `r3dh4t1!`.

* Do not create any projects or applications yet. That comes later.

[NOTE]
Many students have encountered problems with this section. For help, see the _<<Known Issues>>_ section near the top. Also,
remember that you created a copy of `master-config.yaml` called `master-config-yaml.original`, to which you can always revert and
then try again.

=== Registry and Router

In this lab scenario, `infranode00` is the target for both the _registry_ and the _default router_.

==== Deploy Registry

. Deploy `registry`:
+
```

[root@master00-GUID ~]# oadm registry --create --credentials=/etc/origin/master/openshift-registry.kubeconfig

```
+
NOTE: To pin down the registry for a specific region, specify the `--selector` flag. However, you can skip this step because you
already set the default namespace to be the default `nodeSelector`.

. Check the status of your pod with the following commands:
+
```

[root@master00-GUID ~]# oc get pods
NAME   READY   STATUS   RESTARTS   AGE
docker-registry-1-deploy   1/1   Pending   0   11s

```
i. Wait a few seconds … [root@master00-GUID ~]# oc get pods

```
NAME                       READY     STATUS    RESTARTS   AGE
docker-registry-1-deploy   1/1       Running   0          31s
docker-registry-1-diqlc    0/1       Pending   0          4s
```

ii. Wait a few seconds … [root@master00-GUID ~]# oc get pods NAME READY STATUS RESTARTS AGE docker-registry-1-diqlc 1/1 Running 0 14s

```
+
[NOTE]
```
```

```
This process may take a few minutes the first time around because the images are pulled from the registry.

. Run `oc status`:
+
```

[root@master00-GUID master]# oc status In project default on server https://master00-GUID.oslab.opentlc.com:8443

```
svc/docker-registry - 172.30.41.32:5000
  dc/docker-registry deploys docker.io/openshift3/ose-docker-registry:v3.1.0.4
    #1 deployed 5 minutes ago - 1 pod
```

```
svc/kubernetes - 172.30.0.1 ports 443, 53, 53
```

```
To see more, use 'oc describe <resource>/<name>'.
You can use 'oc get all' to see a list of other objects.
```

```
. Test the status of the registry with the `curl` command to communicate with the registry's service port, for example, `curl -v
172.30.41.32:5000/healthz`.
+
To test the registry for connectivity, run these commands:
+
```

```
[root@master00-GUID ~]# echo `oc get service docker-registry --template '{{.spec.portalIP}}:{{index .spec.ports 0
"port"}}/healthz'`
172.30.42.118:5000/healthz
[root@master00-GUID ~]# curl -v `oc get service docker-registry --template '{{.spec.portalIP}}:{{index .spec.ports 0
"port"}}/healthz'`
```

```
* The output looks like this:
+
```

- About to connect() to 172.30.42.118 port 5000 (#0)
- Trying 172.30.42.118…
- Connected to 172.30.42.118 (172.30.42.118) port 5000 (#0) > GET /healthz HTTP/1.1 > User-Agent: curl/7.29.0 > Host: 172.30.42.118:5000 > Accept: / > < HTTP/1.1 200 OK < Content-Type: application/json; charset=utf-8 < Docker-Distribution-Api-Version: registry/2.0 < Date: Thu, 26 Nov 2015 06:56:11 GMT < Content-Length: 3 < {}
- Connection #0 to host 172.30.42.118 left intact

```
==== Deploy Default router

. Create a `CA` Certificate for the default router:
+
```

[root@master00-GUID]# CA=/etc/origin/master [root@master00-GUID]# oadm ca create-server-cert --signer-cert=$CA/ca.crt \ --signer-key=$CA/ca.key --signer-serial=$CA/ca.serial.txt \ --hostnames=*.cloudapps-$guid.oslab.opentlc.com* \ --cert=cloudapps.crt --key=cloudapps.key

```
. Combine `cloudapps.crt` and `cloudapps.key` with `CA` into a single Privacy Enhanced Mail (PEM) format file, which the router
needs in the next step:
+
```

[root@master00-GUID ~]# cat cloudapps.crt cloudapps.key $CA/ca.crt > /etc/origin/master/cloudapps.router.pem

```
. Deploy the default router:
+
```

[root@master00-GUID ~]# oadm router trainingrouter --replicas=1 \ --credentials=*/etc/origin/master/openshift-router.kubeconfig* \ --service-account=router --stats-password=*r3dh@t1!*

```
* The output is as follows:
+
```

password for stats user admin has been set to r3dh@t1! DeploymentConfig "trainingrouter" created Service "trainingrouter" created

```
. On a separate terminal, watch the status of your pods:
+
```

[root@master00-06d0 ~]# oc get pods -w NAME READY STATUS RESTARTS AGE docker-registry-1-diqlc 1/1 Running 0 11m router-1-mpzxx 1/1 Running 0 23s

```
* The Docker registry pods are likely also listed in the above output.

NOTE: Type *Ctrl+C* to exit the "watch" on `oc get pods`.


=== Populate OpenShift Enterprise (Reference Only)

OpenShift Enterprise ships with _image streams_ and _templates_, which reside in `/usr/share/openshift/examples/`.  The installer
imports all the image streams and templates for you from that directory.

* To browse the JSON files, see `/usr/share/openshift/examples`.

[IMPORTANT]
The commands below are for reference only. Run them only if you would like to perform the task in question for some reason.

* To create or delete the core set of image streams whose images are based on Red Hat Enterprise Linux 7:
+
```

oc create|delete -f /usr/share/openshift/examples/image-streams/image-streams-rhel7.json -n openshift

```
* To create or delete the core set of database templates:
+
```

oc create|delete or remove -f /usr/share/openshift/examples/db-templates -n openshift

```
* To create or delete the core QuickStart templates:
+
```

oc create|delete -f /usr/share/openshift/examples/quickstart-templates -n openshift

```
== Set Up Persistent Storage

Having a database for development is handy, but what if you actually want the data you store to persist after redeploying the
database pod? Pods are ephemeral and, by default, so is their storage. For shared or persistent storage, you must be able to
mandate that pods use external volumes.

For the purpose of this course, you learn how to have `oselab` act as your NFS server to export NFS mounts as `PersistentVolume`
targets.

=== Prepare for NFS Persistent Storage Back End

CAUTION: Notice that we are switching to the `oselab` host

As `root` on the `oselab` host, ensure that `nfs-utils` is installed on *all* the nodes:
```

[root@oselab-GUID ~]# for node in infranode00-$guid.oslab.opentlc.com \ node00-$guid.oslab.opentlc.com \ node01-$guid.oslab.opentlc.com; \ do \ echo installing nfs-utils on $node ssh $node "yum -y install nfs-utils" ; done

```
=== Export NFS Volume for Persistent Storage

On the `oselab` administration host, create a directory for each volume that you wish to export through NFS.

. Create 100 directory exports as persistent volumes:
+
```

[root@oselab-GUID]# mkdir -p /var/export/pvs/pv{1..100} [root@oselab-GUID]# chown -R nfsnobody:nfsnobody /var/export/pvs/ [root@oselab-GUID ~]# chmod -R 700 /var/export/pvs/

```
. Add a line for each export directory to `/etc/exports`:
+
```

[root@oselab-GUID ~]# for volume in pv{1..100} ; do echo Creating export for volume $volume; echo "/var/export/pvs/${volume} 192.168.0.0/24(rw,sync,all_squash)" >> /etc/exports; done;

```
. Enable and start NFS services:
+
```

[root@oselab-GUID]# systemctl enable rpcbind nfs-server [root@oselab-GUID]# systemctl start rpcbind nfs-server nfs-lock nfs-idmap [root@oselab-

GUID]# systemctl stop firewalld [root@oselab-GUID]# systemctl disable firewalld

```
+
NOTE: The volume is owned by `nfsnobody` and access by all remote users is "squashed" (through the `all_squash` command) for
access by `nfsnobody`. Essentially, this scenario disables user permissions for clients who mount the volume. While another
configuration may be preferable, one problem you may encounter is that the container cannot modify the permissions of the actual
volume directory when mounted. In the case of MySQL below, MySQL desires that the volume belong to the `mysql` user and assumes
that it is, which causes errors later. Arguably, the container should operate differently for the better. In the long run, Red Hat
may present best practices for use of NFS from containers.


=== Allow NFS Access in SELinux Policy

By policy default, containers cannot write to NFS-mounted directories. However in your lab environment, you want to allow write
access for some of your pods.

. Run the following to allow containers to write to NFS-mounted directories on all the nodes where the pod could land (that is,
all of them):
+
```

[root@oselab-GUID ~]# for node in infranode00-$guid.oslab.opentlc.com \ node00-$guid.oslab.opentlc.com \ node01-$guid.oslab.opentlc.com; \ do echo Setting SElinux Policy on $node ssh $node " setsebool -P virt_use_nfs=true;" done

```
. To ensure NFS Access, connect to a node and verify that you can mount a volume from the `oselab` host:
+
```

[root@oselab-GUID]# ssh 192.168.0.20x [root@node0X-GUID]# mkdir /tmp/test [root@node0X-GUID]# mount -v 192.168.0.254:/var/export/pvs/pv98 /tmp/test # Check if any errors accure and unmount. [root@node0X-GUID]# umount /tmp/test [root@node0X-GUID ~]# exit

```
=== Create Definition Files for Volumes

. Connect to the `master00` host:
+
```

[root@oselab-GUID ~]# ssh master00-$guid

```
. Create a `pvs` directory to store definition files for the persistent volumes in
 your environment:
+
```

[root@master00-GUID ~]# mkdir /root/pvs

```
. Create 25 instances of `PersistentVolume` (`pv1` to `pv25`) with a size of 5 GB:
+
```

[root@master00-GUID]# export volsize="5Gi" [root@master00-GUID]# for volume in pv{1..25} ; do cat << EOF > /root/pvs/${volume} { "apiVersion": "v1", "kind": "PersistentVolume", "metadata": { "name": "${volume}" }, "spec": { "capacity": { "storage": "${volsize}" }, "accessModes": ["ReadWriteOnce"], "nfs": { "path": "/var/export/pvs/${volume}", "server": "192.168.0.254" }, "persistentVolumeReclaimPolicy": "Recycle" } } EOF echo "Created def file for ${volume}"; done;

```
. Create 25 more instances of `PersistentVolume` (`pv26` to `pv50`) with a size of 10 GB:
+
```

[root@master00-GUID]# export volsize="10Gi" [root@master00-GUID]# for volume in pv{26..50} ; do cat << EOF > /root/pvs/${volume} { "apiVersion": "v1", "kind": "PersistentVolume", "metadata": { "name": "${volume}" }, "spec": { "capacity": { "storage": "${volsize}" }, "accessModes": ["ReadWriteOnce"], "nfs": { "path": "/var/export/pvs/${volume}", "server": "192.168.0.254" }, "persistentVolumeReclaimPolicy": "Recycle" } } EOF echo "Created def file for ${volume}"; done;

```
. Create 50 more instances of `PersistentVolume` (`pv51` to `pv100`) with a size of 1 GB:
+
```

[root@master00-GUID]# export volsize="1Gi" [root@master00-GUID]# for volume in pv{51..100} ; do cat << EOF > /root/pvs/${volume} { "apiVersion": "v1", "kind": "PersistentVolume", "metadata": { "name": "${volume}" }, "spec": { "capacity": { "storage": "${volsize}" }, "accessModes": ["ReadWriteOnce"], "nfs": { "path": "/var/export/pvs/${volume}", "server": "192.168.0.254" }, "persistentVolumeReclaimPolicy": "Recycle" } } EOF echo "Created def file for ${volume}"; done;

```
. Allocate three of the 5-GB volumes--`pv21`, `pv22`, and `pv23`--to the default project:
+
```

[root@master00-GUID]# cd /root/pvs [root@master00-GUID]# cat pv21 pv22 pv23 | oc create -f - -n default

```
. Run `oc get pvs` to ensure that your `pvs` volumes have been added and are available:
+
```

[root@master00-GUID pvs]# oc get pv NAME LABELS CAPACITY ACCESSMODES STATUS CLAIM REASON pv21 <none> 5368709120 RWO Available pv22 <none> 5368709120 RWO Available pv23 <none> 5368709120 RWO Available

```
[NOTE]
Although this process is fairly manual here, it can be easily automated to create volumes on request.

The infrastructure for persistent volumes is complete. Learn how to use them in future labs.
```