1. Prepare to Deploy OpenShift Origin Platform

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

* Configure the Secure Shell (SSH) keys.
* Configure the repositories.
* Configure the DNS on the **toolbox** server for your OpenShift Origin Platform environment.
* Configure the network settings.
* Install Docker on all hosts.

|  |  |
| --- | --- |
|  |  |

**Your Environment**

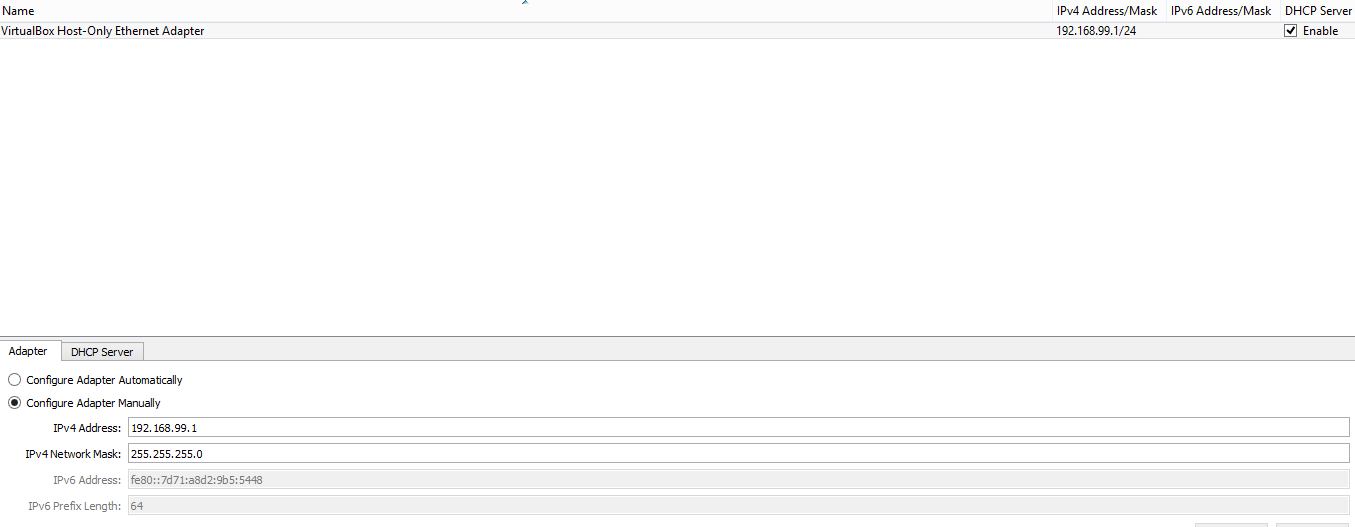
* An OpenShift 3.6 cluster of 4 machine with Centos 7 OS minimal image. Centos 7 minimal install is available from following location:

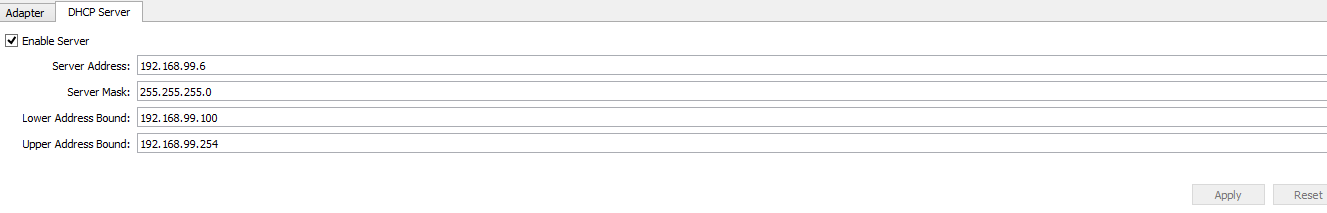
<http://isoredirect.centos.org/centos/7/isos/x86_64/CentOS-7-x86_64-Minimal-1804.iso>

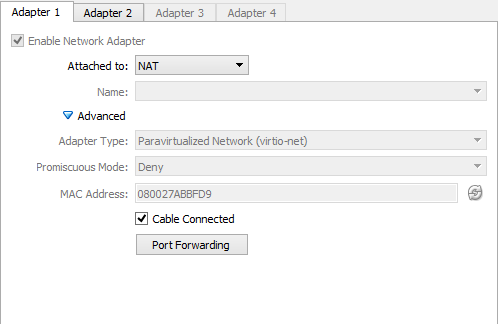
Whenever a VM is created in Oracle VM, choose IDE controller for selecting the Centos 7 ISO image downloaded above and SATA Controller for creating the disk storage.

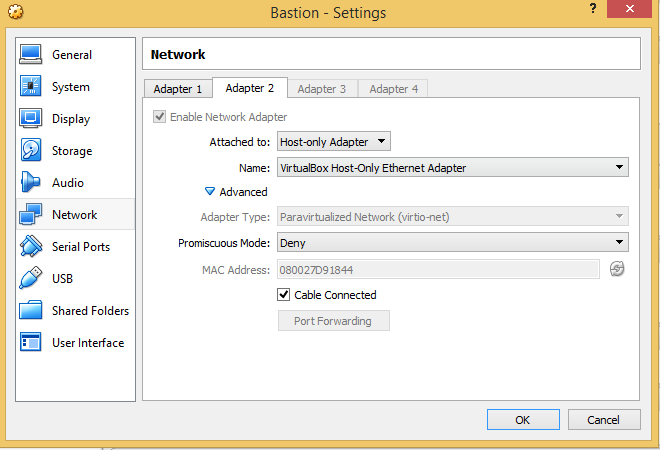
| **Machine** | **Hostname** | **IP** | **Description** |
| --- | --- | --- | --- |
| Toolbox | toolbox.local.net | 192.168.99.100 | Ansible, Bind DNS, and NFS storage |
| Master | master.local.net | 192.168.99.102 | Cluster Master node |
| Infra | infra.local.net | 192.168.99.105 | Infra Node hosting Docker registry and HA Proxy router |
| Node1 | node1.local.net | 192.168.99.106 | Worker node |
|  |  |  |  |

| **Machine** | **CPU** | **Memory** | **Primary Disk** | **Secondary Disk** |
| --- | --- | --- | --- | --- |
| Toolbox | 1 | 512 MB | 40 GB dynamic allocation | NA |
| Master | 1 | 2048 MB | 40 GB dynamic allocation | 15 GB, dynamic, for Docker storage |
| Infra | 1 | 2048 MB | 40 GB dynamic allocation | 20 GB, dynamic, for Docker storage |
| Node1 | 1 | 2048 MB | 40 GB dynamic allocation | 20 GB, dynamic, for Docker storage |







* 

In Windows, add following in C: /Windows/System32/drivers/etc/hosts file:

192.168.99.100 toolbox.local.net

192.168.99.102 master.local.net

* We will be using the **toolbox** host as the DNS and NFS server. Run remote commands on the OpenShift environment on the provisioning and staging host.
* **toolbox** 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 **toolbox** host.
* When executing instructions on 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 saving time, run commands directly on the nodes or hosts instead of using the **for** loop.
  + Administration host example:

[root@toolbox ~]# command

* + Master host example:

[root@master ~]# 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. |
|  |  |

1.1. Connect to Environment

When you connect to **toolbox.local.net** for the first time, you will have to accept the server SSH fingerprint. Reply *yes*: it will be added to your **known\_hosts** and not asked next time you connect.

1. Connect to your administration host **toolbox.local.net**. Note that your private key location may vary.

yourdesktop$ ssh -i ~/.ssh/id\_rsa root@toolbox.local.net

1.2. Configure SSH Keys

The OpenShift Origin installer configures hosts with SSH. In this section you create and install an SSH key pair on the **toolbox** host and add the public key to the **authorized\_hosts** file on the OpenShift hosts.

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

[root@toolbox ~]# ssh-keygen -f /root/.ssh/id\_rsa -N ''

* + The ssh key of root might already exist. If so you will have the following message. You can either recreate the key or keep the current:

root@toolbox ~# ssh-keygen -f /root/.ssh/id\_rsa -N ''

Generating public/private rsa key pair.

/root/.ssh/id\_rsa already exists.

Overwrite (y/n)?

|  |  |
| --- | --- |
|  |  |

1. On the **toolbox** host, add the public SSH key locally to **/root/.ssh/authorized\_keys**:

[root@toolbox ~]# cat /root/.ssh/id\_rsa.pub >> /root/.ssh/authorized\_keys

1. Configure **/etc/ssh/ssh\_config** to disable **StrictHostKeyChecking** on the **toolbox** and master hosts:

[root@toolbox ~]# echo StrictHostKeyChecking no >> /etc/ssh/ssh\_config

[root@toolbox ~]# ssh master.local.net "echo StrictHostKeyChecking no >> /etc/ssh/ssh\_config"

* + This configuration avoids your having to disable strict host-checking and to reply yes when running remote commands on unknown hosts.

1. Add following highlighted in bold in etc/hosts file in toolbox VM:

127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4

**192.168.99.100 localhost localhost.localdomain localhost6 localhost6.localdomain6 toolbox.local.net toolbox.cloudapps.local.net**

::1 localhost localhost.localdomain localhost6 localhost6.localdomain6

**192.168.99.102 master.local.net master.cloudapps.local.net**

**192.168.99.105 infra.local.net infra.cloudapps.local.net**

**192.168.99.106 node1.local.net node1.cloudapps.local.net**

1. On the **toolbox** host, test the new SSH key by connecting it to itself over the loopback interface without a keyboard prompt:

[root@toolbox ~]# ssh 127.0.0.1

...[output omitted]...

[root@toolbox ~]# exit

1. 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@toolbox ~]# for node in master.local.net \

infra.local.net \

node1.local.net; \

do \

ssh-copy-id root@$node ; \

done

|  |  |
| --- | --- |
|  |  |

1.3. Configure Repositories on All Hosts

OpenShift 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**

. Configure it as follows:

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

[root@toolbox ~]# export OWN\_REPO\_PATH=http://admin.na.shared.opentlc.com/repos/ocp/3.6

[root@bastion ~]# cat << EOF > /etc/yum.repos.d/open.repo

[rhel-7-server-rpms]

name=Red Hat Enterprise Linux 7

baseurl=${OWN\_REPO\_PATH}/rhel-7-server-rpms

enabled=1

gpgcheck=0

[rhel-7-server-rh-common-rpms]

name=Red Hat Enterprise Linux 7 Common

baseurl=${OWN\_REPO\_PATH}/rhel-7-server-rh-common-rpms

enabled=1

gpgcheck=0

[rhel-7-server-extras-rpms]

name=Red Hat Enterprise Linux 7 Extras

baseurl=${OWN\_REPO\_PATH}/rhel-7-server-extras-rpms

enabled=1

gpgcheck=0

[rhel-7-server-optional-rpms]

name=Red Hat Enterprise Linux 7 Optional

baseurl=${OWN\_REPO\_PATH}/rhel-7-server-optional-rpms

enabled=1

gpgcheck=0

[rhel-7-fast-datapath-rpms]

name=Red Hat Enterprise Linux 7 Fast Datapath

baseurl=${OWN\_REPO\_PATH}/rhel-7-fast-datapath-rpms

enabled=1

gpgcheck=0

EOF

1. Add the OpenShift Container Platform repository mirror to the **toolbox** host:

[root@toolbox ~]# cat << EOF >> /etc/yum.repos.d/open.repo

[rhel-7-server-ose-3.6-rpms]

name=Red Hat Enterprise Linux 7 OSE 3.6

baseurl=${OWN\_REPO\_PATH}/rhel-7-server-ose-3.6-rpms

enabled=1

gpgcheck=0

EOF

1. Deactivate the previous Red Hat repositories, as they are not needed anymore:

[root@toolbox ~]# mv /etc/yum.repos.d/redhat.{repo,disabled}

1. Clean up and list the repositories on the **toolbox** host:

[root@toolbox ~]# yum clean all ; yum repolist

* + The output is as follows:
  + Loaded plugins: product-id
  + ...[output omitted]...
  + repolist: 10,114
  + repo id repo name status
  + rhel-7-fast-datapath-rpms Red Hat Enterprise Linux 7 Fast Datapath 15
  + rhel-7-server-extras-rpms Red Hat Enterprise Linux 7 Extras 76
  + rhel-7-server-optional-rpms Red Hat Enterprise Linux 7 Optional 4,648
  + rhel-7-server-ose-3.5-rpms Red Hat Enterprise Linux 7 OSE 3.5 415
  + rhel-7-server-rh-common-rpms Red Hat Enterprise Linux 7 Common 82
  + rhel-7-server-rpms Red Hat Enterprise Linux 7 4,878

repolist: 10,114

1. Configure the master and nodes by disabling redhat.repo and copying the **open.repo** file to all of the nodes directly from the **toolbox** host:

[root@toolbox ~]# for node in master.local.net\

infra.local.net \

node1.local.net; do

echo Copying open repos to $node

scp /etc/yum.repos.d/open.repo ${node}:/etc/yum.repos.d/open.repo

ssh ${node} 'mv /etc/yum.repos.d/redhat.{repo,disabled}'

ssh ${node} yum clean all

ssh ${node} yum repolist

done

1.4. Configure Wildcard DNS Entry on **toolbox**

OpenShift Origin 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 **infra** host. |

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

[root@toolbox ~]# yum -y install bind bind-utils

1. On the administration host, **toolbox**, create following files:
   1. named.conf in /etc folder.

//

// named.conf

//

// Provided by Red Hat bind package to configure the ISC BIND named(8) DNS

// server as a caching only nameserver (as a localhost DNS resolver only).

//

// See /usr/share/doc/bind\*/sample/ for example named configuration files.

//

// See the BIND Administrator's Reference Manual (ARM) for details about the

// configuration located in /usr/share/doc/bind-{version}/Bv9ARM.html

options {

listen-on port 53 { any; };

listen-on-v6 port 53 { ::1; };

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; };

/\*

- If you are building an AUTHORITATIVE DNS server, do NOT enable recursion.

- If you are building a RECURSIVE (caching) DNS server, you need to enable

recursion.

- If your recursive DNS server has a public IP address, you MUST enable access

control to limit queries to your legitimate users. Failing to do so will

cause your server to become part of large scale DNS amplification

attacks. Implementing BCP38 within your network would greatly

reduce such attack surface

\*/

recursion yes;

dnssec-enable yes;

dnssec-validation yes;

/\* Path to ISC DLV key \*/

bindkeys-file "/etc/named.iscdlv.key";

managed-keys-directory "/var/named/dynamic";

pid-file "/run/named/named.pid";

session-keyfile "/run/named/session.key";

};

logging {

channel default\_debug {

file "data/named.run";

severity dynamic;

};

};

zone "." IN {

type hint;

file "named.ca";

};

zone "local.net." IN {

type master;

file "forward.local";

allow-update { none; };

};

zone "99.168.192.in-addr.arpa" IN {

type master;

file "reverse.local";

allow-update { none; };

};

zone "cloudapps.local.net." IN {

type master;

file "forward.cloudapps.local";

allow-update { none; };

};

include "/etc/named.rfc1912.zones";

include "/etc/named.root.key";

* 1. forward.local in /var/named folder.

$TTL 86400

@ IN SOA toolbox.local.net. root.local.net. (

2011071001 ;Serial

3600 ;Refresh

1800 ;Retry

604800 ;Expire

86400 ;Minimum TTL

)

@ IN NS toolbox.local.net.

@ IN A 192.168.99.100

@ IN A 192.168.99.102

@ IN A 192.168.99.105

@ IN A 192.168.98.106

toolbox IN A 192.168.99.100

master IN A 192.168.99.102

infra IN A 192.168.99.105

node1 IN A 192.168.99.106

* 1. reverse.local in /var/named folder.

$TTL 86400

@ IN SOA toolbox.local.net. root.local.net. (

2011071001 ;Serial

3600 ;Refresh

1800 ;Retry

604800 ;Expire

86400 ;Minimum TTL

)

@ IN NS toolbox.local.net.

@ IN PTR local.net.

toolbox IN A 192.168.99.100

master IN A 192.168.99.102

infra IN A 192.168.99.105

node1 IN A 192.168.99.106

100 IN PTR toolbox.local.net.

102 IN PTR master.local.net.

105 IN PTR infra.local.net.

106 IN PTR node1.local.net.

* 1. forward.cloudapps.local in /var/named folder.

$TTL 86400

@ IN SOA . root.cloudapps.local.net. (

2011071001 ;Serial

3600 ;Refresh

1800 ;Retry

604800 ;Expire

86400 ;Minimum TTL

)

@ IN NS .

@ IN A 192.168.99.105

\* IN A 192.168.99.105

1. On the administration host, **toolbox**, run following command for giving permission to the created files.

[root@toolbox ~]# chgrp named -R /var/named

[root@toolbox ~]# chown -v root:named /etc/named.conf

[root@toolbox ~]# restorecon -rv /var/named

[root@toolbox ~]# restorecon /etc/named.conf

1. Enable and Start DNS Service

[root@toolbox ~]# systemctl enable named

[root@toolbox ~]# systemctl start named

1. Enable and Start Firewall Service

[root@toolbox ~]# systemctl enable firewalld

[root@toolbox ~]# systemctl start firewalld

In case firewalld is masked, run following command to unmask firewalls before running above commands:

[root@toolbox ~]# systemctl unmask firewalld

1. Add Firewall Rules

[root@toolbox ~]# firewall-cmd --permanent --add-port=53/tcp

[root@toolbox ~]# firewall-cmd --permanent --add-port=53/udp

1. Configure IP Tables

[root@toolbox ~]# iptables -I INPUT 1 -p tcp --dport 53 -s 0.0.0.0/0 -j ACCEPT;

[root@toolbox ~]# iptables-save > /etc/sysconfig/iptables

[root@toolbox ~]# iptables -I INPUT 1 -p udp --dport 53 -s 0.0.0.0/0 -j ACCEPT;

[root@toolbox ~]# iptables-save > /etc/sysconfig/iptables

1. Configure Network Manager

[root@toolbox ~]# sed -i "/\[main\]/a dns=none" /etc/NetworkManager/NetworkManager.conf

[root@toolbox ~]# systemctl restart NetworkManager

[root@toolbox ~]# cat << EOF > /etc/resolv.conf

local.net

nameserver 192.168.99.100

EOF

[root@toolbox ~]# systemctl restart network

1.5. Verify DNS Configuration

[root@toolbox ~]# named-checkconf /etc/named.conf

[root@toolbox ~]# named-checkzone local.net /var/named/forward.local

[root@toolbox ~]# named-checkzone local.net /var/named/reverse.local

[root@toolbox ~]# named-checkzone local.net /var/named/forward.cloudapps.local

[root@toolbox ~]# dig infra.local.net

[root@toolbox ~]# nslookup local.net

[root@toolbox ~]# dig XXX.cloudapps.local.net

[root@toolbox ~]# nslookup cloudapps.local.net

1.6. Install and Configure Ansible on **toolbox**

The advanced installation method is based on Ansible playbooks, so you must be able to directly invoke Ansible.

1. Install Ansible from **yum**:

[root@toolbox ~]# yum -y install ansible

1. Create a simple inventory file with groups used by Ansible:

[root@toolbox ~]# cat << EOF > /etc/ansible/hosts

[masters]

master.local.net

[nodes]

master.local.net

infra.local.net

node1.local.net

EOF

[root@toolbox ~]# cat /etc/ansible/hosts

1. Test the Ansible configuration:

[root@toolbox ~]# ansible nodes -m ping

master.local.net | success >> {

"changed": false,

"ping": "pong"

}

infra.local.net | success >> {

"changed": false,

"ping": "pong"

}

node1.local.net | success >> {

"changed": false,

"ping": "pong"

}

1.7. Install Packages

1. On the **toolbox** host, run the following **for** loop to ensure that **NetworkManager** is installed on the master and all nodes:

[root@toolbox ~]# for node in master.local.net \

infra.local.net \

node1.local.net; \

do \

echo installing NetworkManager on $node ; \

ssh $node "yum -y install NetworkManager"

done

|  |  |
| --- | --- |
|  |  |

1. (Reference Only) Install the following tools and utilities on the **toolbox** host:

[root@toolbox ~]# yum -y install wget git net-tools bind-utils iptables-services bridge-utils

1. (Reference Only) Install **bash-completion** on both the **toolbox** and **master** hosts:

[root@toolbox ~]# yum -y install bash-completion

[root@toolbox ~]# ssh master.local.net yum -y install bash-completion

|  |  |
| --- | --- |
|  | **bash-completion** is not available for use until the **bash** shell is restarted. |

1. Run **yum update** on the master and all nodes:

[root@toolbox ~]# for node in master.local.net \

infra.local.net \

node1.local.net; \

do \

echo Running yum update on $node ; \

ssh $node "yum -y update " ; \

done

|  |  |
| --- | --- |
|  |  |

1.8. Install Docker

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

* Install the **docker** package on the master and all nodes:

[root@bastion ~]# for node in master.local.net \

infra.local.net \

node1.local.net; \ do \

echo Installing docker on $node ; \

ssh $node "yum -y install docker" ;

done

|  |  |
| --- | --- |
|  |  |

1.9. Configure Docker Storage

Next, configure the Docker storage pool.

|  |  |
| --- | --- |
|  | . |

Stop the Docker daemon and delete any files from **/var/lib/docker**:

[root@toolbox ~]# for node in master.local.net\

infra.local.net \

node1.local.net; \

do

echo Cleaning up Docker on $node ; \

ssh $node "systemctl stop docker ; rm -rf /var/lib/docker/\*" ;

done

|  |  |
| --- | --- |
|  |  |

1. connect to master

[root@toolbox ~]# ssh master.local.net

1. identify the block device that will serve as the Docker Physical volume

[root@master1 ~]# lsblk

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT

fd0 2:0 1 4K 0 disk

sr0 11:0 1 1024M 0 rom

sda 252:0 0 10G 0 disk

└─sda1 252:1 0 10G 0 part /

sdb 252:16 0 25G 0 disk

sdc 252:32 0 366K 1 disk

1. Specify the **/dev/sdb** hard drive for the Docker volume group for **docker-storage setup**:

[root@master ~]# cat <<EOF > /etc/sysconfig/docker-storage-setup

DEVS=/dev/sdb

VG=docker-vg

EOF

1. Run **docker-storage-setup** on the **master** host to create logical volumes for Docker:

[root@master ~]# docker-storage-setup

* + The output is as follows:
  + Checking that no-one is using this disk right now ...
  + OK
  + Disk /dev/sdb: 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/sdb1 2048 20971519 20969472 8e Linux LVM
  + /dev/sdb2 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).)
  + INFO: Device node /dev/vdb1 exists.
  + Physical volume "/dev/vdb1" successfully created.
  + Volume group "docker-vg" successfully created
  + Using default stripesize 64.00 KiB.
  + Rounding up size to full physical extent 28.00 MiB
  + Logical volume "docker-pool" created.

Logical volume docker-vg/docker-pool changed.

|  |  |
| --- | --- |
|  |  |

1. On the master host, examine the newly created **docker-pool** logical volume:

[root@master ~]# 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--- 9.94g 0.00 0.14

1. On the master host, examine the configuration of **docker storage**:

[root@master ~]# 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 --storage-opt dm.use\_deferred\_removal=true

1. Enable the Docker service on the master host:

[root@master ~]# systemctl enable docker

Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service to /usr/lib/systemd/system/docker.service.

[root@master ~]# systemctl start docker

1. Go back to toolbox:

[root@master ~]# exit

[root@toolbox ~]#

1. Run the following **for** loop to configure Docker storage on the other nodes, enable Docker, and restart the node:

[root@toolbox ~]# scp root@master.local.net:/etc/sysconfig/docker-storage-setup ./

[root@toolbox ~]# for node in infra.local.net \

node1.local.net; \

node2.example.com; \

do

echo Configuring Docker Storage and rebooting $node

scp docker-storage-setup ${node}:/etc/sysconfig/docker-storage-setup

ssh $node "

docker-storage-setup ;

systemctl enable docker

systemctl start docker"

done

1.10. Populate Local Docker Registry

1. Verify that the Docker service has started on all nodes:

[root@toolbox ~]# for node in master.local.net \

infra.local.net \

node1.example.com \

node1.local.net; \

do

echo Checking docker status on $node

ssh $node "

systemctl status docker | grep Active"

done

|  |  |
| --- | --- |
|  |  |

1. Now on the **toolbox** host, pull down the Docker images to **node1**  in the primary region with the following command:

[root@toolbox ~]# for node in node1.local.net;\

do

ssh $node "

docker pull openshift/origin-deployer:v3.6.1 ; \

docker pull openshift/origin-sti-builder:v3.6.1 ; \

docker pull openshift/origin-pod:v3.6.1 ; \

docker pull openshift/origin-keepalived-ipfailover:v3.6.1 ; \

docker pull openshift/ruby-20-centos7 ; \

docker pull openshift/mysql-55-centos7 ; \

docker pull openshift/hello-openshift:v1.2.1 ;"

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.

|  |  |
| --- | --- |
|  |  |

1. On **toolbox**, pull only the basic images and the registry and router images to the **infra** host:

[root@toolbox ~]# node=infra.local.net

[root@toolbox ~]# ssh $node "

docker pull openshift/origin-haproxy-router:v3.6.1

docker pull openshift/origin-deployer:v3.6.1

docker pull openshift/origin-pod:v3.6.1

docker pull openshift/origin-docker-registry:v3.6.1"

|  |  |
| --- | --- |
|  |  |

* + You are not pulling any images on the master host because it is not meant to run any containers.

1. Examine the information in the Docker pool on the **node1**  host:

[root@toolbox ~]# ssh node1.local.net docker info

* + The **node1** 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: node1.local.net

ID: RXVI:JKOO:3U4X:LHDE:QXPN:FSQC:TTBL:UCWP:MCEH:2KU6:GWSD:IRIN

...

1. On the **node1**  host, examine the **docker-pool** logical volume again:

[root@toolbox ~]# ssh node1.local.net "lvs"

* + The **node1** output is similar to below.
  + 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--- 9.94g 0.00 0.14

2. Install OpenShift Origin Platform

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

2.1. Download Ansible Playbook

In this exercise you run the Ansible playbook from the **toolbox** host, which, in a real-world scenario, could be a laptop or a staging or provisioning server. No packages are deployed directly from **toolbox** to the OpenShift nodes or master.

* On the **toolbox** host, install the OpenShift utility package:

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

2.2. Create Inventory File

The **/etc/ansible/hosts** file is Ansible’s inventory file for the playbook to use during the installation. The inventory file describes the configuration for your OpenShift Container Platform cluster.

* Write the inventory file:

[root@toolbox ~]# cat << EOF > /etc/ansible/hosts[OSEv3:children]

masters

etcd

nodes

nfs

[OSEv3:vars]

ansible\_ssh\_user=root

debug\_level=2

openshift\_deployment\_type=origin

openshift\_release=v3.6.0

openshift\_image\_tag=v3.6.0

openshift\_pkg\_version=-3.6.0

# Skip memory and disk check

openshift\_disable\_check=disk\_availability,memory\_availability,package\_version,docker\_image\_availability

openshift\_master\_cluster\_method=native

openshift\_master\_cluster\_hostname=master.local.net

openshift\_master\_cluster\_public\_hostname=master.local.net

#os\_sdn\_network\_plugin\_name='redhat/openshift-ovs-multitenant'

openshift\_master\_default\_subdomain=cloudapps.local.net

#openshift\_use\_dnsmasq=False

#openshift\_node\_dnsmasq\_additional\_config\_file=/home/bob/ose-dnsmasq.conf

# Configure Security

openshift\_master\_identity\_providers=[{'name': 'htpasswd\_auth', 'login': 'true', 'challenge': 'true', 'kind': 'HTPasswdPasswordIdentityProvider', 'filename': '/etc/origin/master/htpasswd'}]

# default project node selector

osm\_default\_node\_selector='region=primary'

# Configure Router

openshift\_hosted\_router\_selector='zone=infranodes'

openshift\_hosted\_router\_replicas=1

#openshift\_hosted\_router\_certificate={\"certfile\": \"/path/to/router.crt\", \"keyfile\": \"/path/to/router.key\", \"cafile\": \"/path/to/router-ca.crt\"}

# Configure Registry

openshift\_hosted\_registry\_selector='zone=infranodes'

openshift\_hosted\_registry\_replicas=1

openshift\_hosted\_registry\_storage\_kind=nfs

openshift\_hosted\_registry\_storage\_access\_modes=['ReadWriteMany']

openshift\_hosted\_registry\_storage\_host=toolbox.local.net

openshift\_hosted\_registry\_storage\_nfs\_directory=/exports

openshift\_hosted\_registry\_storage\_volume\_name=registry

openshift\_hosted\_registry\_storage\_volume\_size=5Gi

# Configure Logging

#openshift\_hosted\_logging\_deploy=true

#openshift\_hosted\_logging\_elasticsearch\_cluster\_size=1

#openshift\_hosted\_logging\_storage\_kind=nfs

#openshift\_hosted\_logging\_storage\_access\_modes=['ReadWriteOnce']

#openshift\_hosted\_logging\_storage\_host=toolbox.local.net

#openshift\_hosted\_logging\_storage\_nfs\_directory=/exports

#openshift\_hosted\_logging\_storage\_volume\_name=logging

#openshift\_hosted\_logging\_storage\_volume\_size=10Gi

#openshift\_hosted\_logging\_storage\_labels={'storage': 'logging'}

# Configure Metrics

#openshift\_hosted\_metrics\_deploy=true

#openshift\_hosted\_metrics\_storage\_kind=nfs

#openshift\_hosted\_metrics\_storage\_access\_modes=['ReadWriteOnce']

#openshift\_hosted\_metrics\_storage\_host=toolbox.local.net

#openshift\_hosted\_metrics\_storage\_nfs\_directory=/exports

#openshift\_hosted\_metrics\_storage\_volume\_name=metrics

#openshift\_hosted\_metrics\_storage\_volume\_size=10Gi

#openshift\_hosted\_metrics\_storage\_labels={'storage': 'metrics'}

[toolbox]

toolbox.local.net ansible\_host=192.168.99.100 ansible\_connection=local openshift\_hostname=toolbox.local.net openshift\_public\_hostname=toolbox.local.net openshift\_ip=192.168.99.100 openshift\_public\_ip=192.168.99.100

[ansible:children]

toolbox

[nfs:children]

toolbox

[infra]

infra.local.net ansible\_host=192.168.99.105 openshift\_hostname=infra.local.net openshift\_public\_hostname=infra.local.net openshift\_ip=192.168.99.105 openshift\_public\_ip=192.168.99.105 openshift\_node\_labels="{'region': 'infra', 'zone': 'infranodes'}"

[masters]

master.local.net ansible\_host=192.168.99.102 openshift\_hostname=master.local.net openshift\_public\_hostname=master.local.net openshift\_ip=192.168.99.102 openshift\_public\_ip=192.168.99.102 openshift\_node\_labels="{'region': 'infra'}"

[etcd:children]

Masters

[worker\_nodes]

node1.local.net ansible\_host=192.168.99.106 openshift\_hostname=node1.local.net openshift\_public\_hostname=node1.local.net openshift\_ip=192.168.99.106 openshift\_public\_ip=192.168.99.106 openshift\_node\_labels="{'region': 'primary', 'zone': 'east'}"

[nodes:children]

masters

etcd

infra

worker\_nodes

[cluster:children]

ansible

nfs

nodes

EOF

2.3. Run Ansible Playbook

1. After configuring Ansible by defining an inventory file in **/etc/ansible/hosts**, run the installation using the following playbook:

[root@toolbox ~]# ansible-playbook /usr/share/ansible/openshift-ansible/playbooks/byo/config.yml

1. Watch the Ansible playbook run:

[Omitted long output]

PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

infra.local.net : ok=105 changed=29 unreachable=0 failed=0

localhost : ok=21 changed=0 unreachable=0 failed=0

master.local.net : ok=396 changed=73 unreachable=0 failed=0

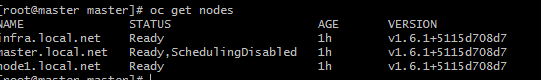
node1.local.net : ok=105 changed=29 unreachable=0 failed=0

2.4. Verify the Environment

1. Connect to the **master** host:

[root@toolbox ~]# ssh master.local.net

1. Run **oc get nodes** to check the status of the hosts:



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

1. Use your browser to connect to the OpenShift web console at **https://master.local.net:8443** and accept the untrusted certificate.
   * You cannot log in yet because you have not set up authentication.
2. Create a new user called developer in the htpasswd file created from the location mentioned in the ansible inventory file by typing the following command in master node VM:

**htpasswd /etc/origin/master/htpasswd developer**

When prompted for password, type a new password.

Now you should be able to login with new login credentials.

You can assign privileges to the user by first logging as cluster admin user.

**oc login –u system:admin –n default**

Then provide appropriate privilege to the user using oc adm policy command.