ß SHOW

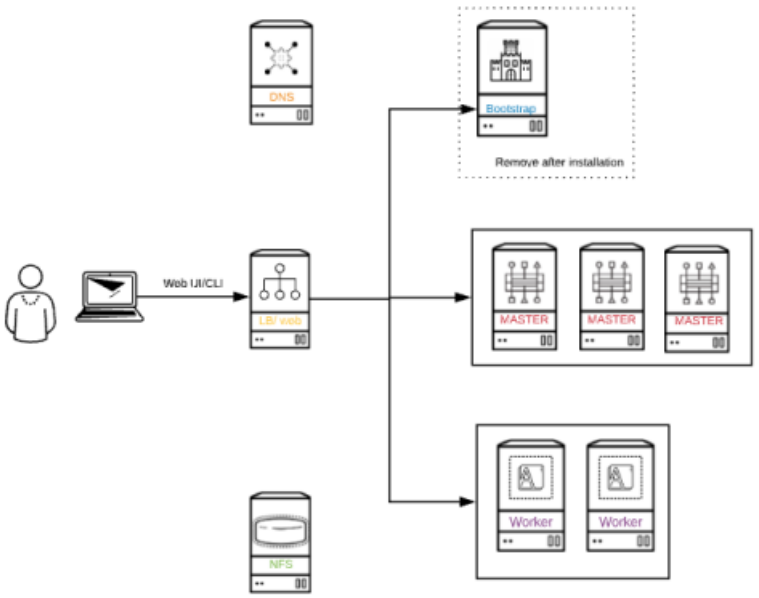
OpenShift Lessons Learned

OpenShift4.3: Retest Static IP configuration on vSphere

March 16, 2020March 18, 2020
Lesson learned from the last test (https://shanna-chan.blog/2019/07/26/openshift4-vsphere-static-ip/), and I got questions around clarification on using static IP. My apologies for the confusion from my last test since it was my test without any real documentation. I want to record all my errors so I can help others to troubleshoot.

Anyway, I decided to retest the installation of OCP 4.3 using static IP. The goal to clarify the installation instructions my last note from the last blog if you are trying to install OCP4 on the VMware environment manually using static IP.

Environment:



- OCP 4.3.5
- vSphere 6.7

List of VMs:

- Bootstrap 192.168.1.110
- Master0 192.168.1.111
- Master1 192.168.1.112

- Master2 192.168.1.113
- Worker0 192.168.1.114
- Worker1 192.168.1.115

Prerequisites:

The following components are already running in my test environment.

DNS Server

1. Add Zone /etc/named.conf. An example can be found here https://github.com/christianh814/openshift-toolbox/blob/master/ocp4_upi/docs/0.prereqs.md#dns
2. Configures the zone files for all the DNS entries. An example configuration is shown below.

```
; The api points to the IP of your load balancer
api.ocp43       IN      A       192.168.1.72
api-int.ocp43   IN      A       192.168.1.72
;
; The wildcard also points to the load balancer
*.apps.ocp43    IN      A       192.168.1.72
;
; Create entry for the bootstrap host
bootstrap0.ocp43        IN      A       192.168.1.110
;
; Create entries for the master hosts
master01.ocp43  IN      A       192.168.1.111
master02.ocp43  IN      A       192.168.1.112
master03.ocp43  IN      A       192.168.1.113
;
; Create entries for the worker hosts
worker01.ocp43  IN      A       192.168.1.114
worker02.ocp43  IN      A       192.168.1.115
;
; The ETCd cluster lives on the masters...so point these to the IP of the masters
etcd-0.ocp43    IN      A       192.168.1.111
etcd-1.ocp43    IN      A       192.168.1.112
etcd-2.ocp43    IN      A       192.168.1.113
;
; The SRV records are IMPORTANT....make sure you get these right...note the trailing dot
_etcd-server-ssl._tcp.ocp43     IN      SRV     0 10 2380 etcd-0.ocp43.example.com.
_etcd-server-ssl._tcp.ocp43     IN      SRV     0 10 2380 etcd-1.ocp43.example.com.
_etcd-server-ssl._tcp.ocp43     IN      SRV     0 10 2380 etcd-2.ocp43.example.com.
```

Load balancer

1. Update /etc/haproxy/haproxy.cfg with cluster information. An example is shown below.

```
#---------------------------------------------------------------------

listen stats
    bind *:9000
    mode http
    stats enable
    stats uri /
    monitor-uri /healthz

#---------------------------------------------------------------------
#Cluster ocp43 - static ip test
frontend openshift-api-server
    bind *:6443
    default_backend openshift-api-server
    mode tcp
    option tcplog

backend openshift-api-server
    balance source
    mode tcp
    #server bootstrap0.ocp43.example.com 192.168.1.110:6443 check
    server master01.ocp43.example.com 192.168.1.111:6443 check
    server master02.ocp43.example.com 192.168.1.112:6443 check
    server master03.ocp43.example.com 192.168.1.113:6443 check

frontend machine-config-server
    bind *:22623
    default_backend machine-config-server
    mode tcp
    option tcplog

backend machine-config-server
    balance source
    mode tcp
    # server bootstrap0.ocp43.example.com 192.168.1.110:22623 check
    server master01.ocp43.example.com 192.168.1.111:22623 check
    server master02.ocp43.example.com 192.168.1.112:22623 check
    server master03.ocp43.example.com 192.168.1.113:22623 check

frontend ingress-http
    bind *:80
    default_backend ingress-http
    mode tcp
    option tcplog

backend ingress-http
    balance source
    mode tcp
    server worker01.ocp43.example.com 192.168.1.114:80 check
    server worker02.ocp43.example.com 192.168.1.115:80 check

frontend ingress-https
    bind *:443
    default_backend ingress-https
    mode tcp
    option tcplog

backend ingress-https
    balance source
    mode tcp
    server worker01.ocp43.example.com 192.168.1.114:443 check
    server worker02.ocp43.example.com 192.168.1.115:443 check
```

Web Server

1. Configure a web server. In my example, I configure httpd on an RHEL VM.

```
yum -y install httpd
systemctl enable --now httpd
firewall-cmd --add-service=8080/tcp --permanent
firewall-cmd --reload
```

Installation downloads

- From https://mirror.openshift.com/pub/openshift-v4/clients/ocp/4.3.5/
  - openshift-install-mac-4.3.5.tar.gz
  - openshift-client-mac-4.3.5.tar.gz
- From https://mirror.openshift.com/pub/openshift-v4/dependencies/rhcos/4.3/latest/
  - rhcos-4.3.0-x86_64-installer.iso
  - rhcos-4.3.0-x86_64-metal.raw.gz

Installation Using Static IP address

Prepare installation

1. Generate SSH key:

```
$ ssh-keygen -t rsa -b 4096 -N '' -f ~/.ssh/vsphere-ocp43
```

2. Start ssh-agent:

```
$ eval "$(ssh-agent -s)"
```

3. Add ssh private key to the ssh-agent:

```
$ ssh-add ~/.ssh/vsphere-ocp43
Identity added: /Users/shannachan/.ssh/vsphere-ocp43 (shannachan@MacBook-Pro)
```

4. Download & extract OpenShift Installer:

```
wget https://mirror.openshift.com/pub/openshift-v4/clients/ocp/4.3.5/openshift-install-ma
tar zxvf openshift-install-mac-4.3.5.tar.gz
```

5. Download & extract OpenShift CLI:

```
wget wget https://mirror.openshift.com/pub/openshift-v4/clients/ocp/4.3.5/openshift-clien
tar zxvf openshift-client-mac-4.3.5.tar.gz
```

6. Copy or download the pull secret from cloud.redhat.com
    1. Go to cloud.redhat.com
    2. Login with your credential (create an account if you don't have one)
    3. Click "Create Cluster"
    4. Click OpenShift Container Platform
    5. Scroll down and click "VMware vSphere"
    6. Click on "Download Pull Secret" to download the secret

Create Installation manifests and ignition files

1. Create an installation directory:

```
mkdir ocp43
```

2. Create `install-config.yaml` as shown below.

```
apiVersion: v1
baseDomain: example.com
compute:
- name: worker
  replicas: 0
controlPlane:
  hyperthreading: Enabled
  name: master
  replicas: 3
metadata:
  name: ocp43
platform:
  vsphere:
    vcenter: 192.168.1.200
    username: vsphereadmin
    password: xxxx
    datacenter: Datacenter
    defaultDatastore: datastore3T
pullSecret: '<copy your pull secret here>'
sshKey: '<copy your public key here>'
```

3. Backup install-config.yaml and copy it into the installation directory
4. Generate Kubernetes manifests for the cluster:

```
$./openshift-install create manifests --dir=./ocp43
INFO Consuming Install Config from target directory
WARNING Making control-plane schedulable by setting MastersSchedulable to true for Schedu
```

5. Modify <installation directory>/manifests/cluster-scheduler-02-config.yml
6. Update mastersSchedulable to `false`
7. Obtain Ignition files:

```
$ ./openshift-install create ignition-configs --dir=./ocp43
INFO Consuming Common Manifests from target directory
INFO Consuming Worker Machines from target directory
INFO Consuming Master Machines from target directory
INFO Consuming OpenShift Install (Manifests) from target directory
INFO Consuming Openshift Manifests from target directory
```

8. Files that were created:

```
$ tree ocp43
ocp43
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── master.ign
├── metadata.json
└── worker.ign
```

Upload files to the webserver

1. Upload the rhcos-4.3.0-x86_64-metal.raw.gz to web server location
2. Upload all the ignition files to the webserver location
3. Update the file permission on the *.ign files on the webserver:

```
chmod 644 *.ign
```

Note: check and make sure that you can download the ignition files and gz file for the webserver.

Custom ISO

Create all custom ISO files with the parameters that you need for each VMs. This step can skip if you plan to type all the kernel parameters by hand when prompt.

1. Download rhcos-4.3.0-x86_64-installer.iso and rhcos-4.3.0-x86_64-metal.raw.gz
2. Extract ISO to a temporary location:

```
sudo mount rhcos-410.8.20190425.1-installer.iso /mnt/
mkdir /tmp/rhcos
rsync -a /mnt/* /tmp/rhcos/
cd /tmp/rhcos
vi isolinux/isolinux.cfg
```

3. Modify the boot entry similar to this:

```
label linux
  menu label ^Install RHEL CoreOS
  kernel /images/vmlinuz
  append initrd=/images/initramfs.img nomodeset rd.neednet=1 coreos.inst=yes ip=192.168.1
```

where:

ip=<ip address of the VM>::<gateway>:<netmask>:<hostname of the VM>:<interface>:none

nameserver=<DNS>

coreos.inst.image_url=http://<webserver host:port>/rhcos-4.3.0-x86_64-metal.raw.gz

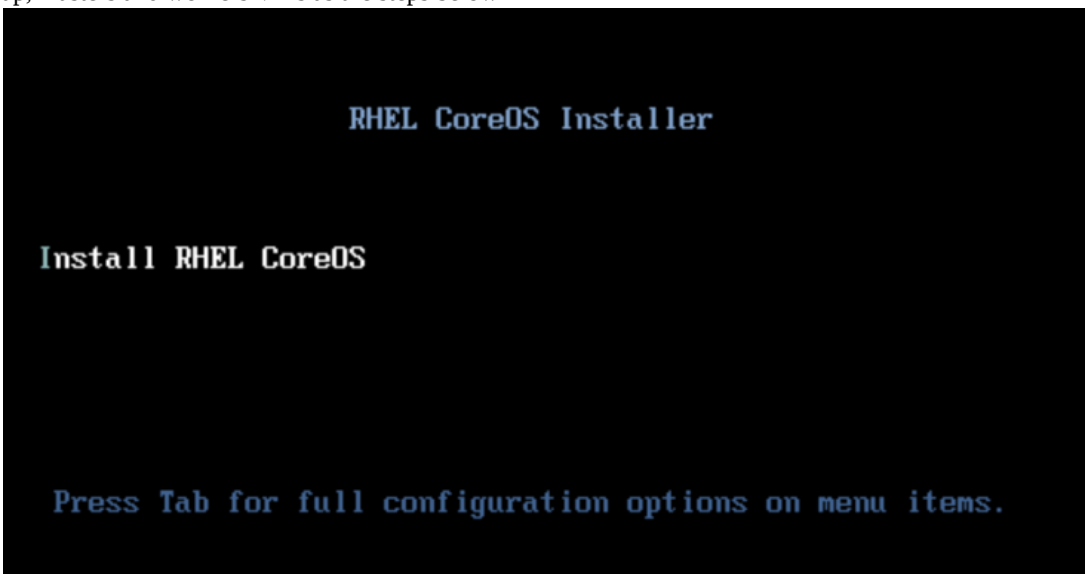
coreos.inst.ignition_url=http://<webserver host:port>/<bootstrap, master or worker ignition>.ign

4. Create new ISO as /tmp/rhcos_install.iso:

```
sudo mkisofs -U -A "RHCOS-x86_64" -V "RHCOS-x86_64" -volset "RHCOS-x86_64" -J -joliet-lon
```

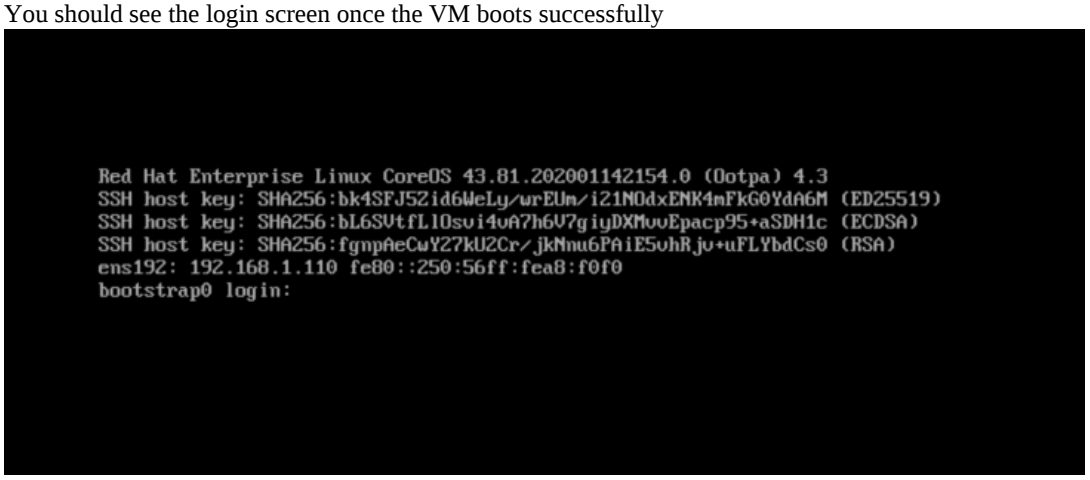
5. Upload all the custom ISOs to the datastore for VM creation via vCenter
6. You will repeat the steps for all VMs with the specific IP and ign file. You only need to create individual VM for the cluster if you don't want to type the kernel parameters at the prompt when installing via the ISO. I would recommend that since it actually takes less time to do that than typing the kernel parameters each time.

Create VM using custom ISO

1. Create a resource folder
   - Action -> New folder -> New VM or Template folder
   - I normally give the name as the cluster id
2. Create VM with 4 CPU and 16 RAM
   - Action -> New Virtual Machine
   - Select `Create New Virtual Machine` -> click Next
   - Add name
   - Select the VM folder -> Next
   - Select datacenter -> Next
   - Select storage -> Next
   - Use ESXi 6.7 -> Next
   - Select Linux and RHEL 7 -> Next
   - Use these parameters:
     - CPU: 4
     - Memory: 16 (Reserve all guest memory)
     - 120 GB disk
     - Select the corresponding ISO from Datastore and check "connect"
     - VMOption -> advantage -> Edit configuration -> Add configuration Params -> Add "disk.EnableUUID": Specify TRUE
     - Click OK
     - Click Next
     - Click Finish
3. Power on the bootstrap, masters and workers VMs as the steps below
4. Go the VM console: 

5. Hit Enter

6. You should see the login screen once the VM boots successfully



7. repeat on all servers and make sure the specific ISO for the given VM is used.

Tips: you can clone the existing VM and just modify the ISO files for VM creation.

Creating Cluster

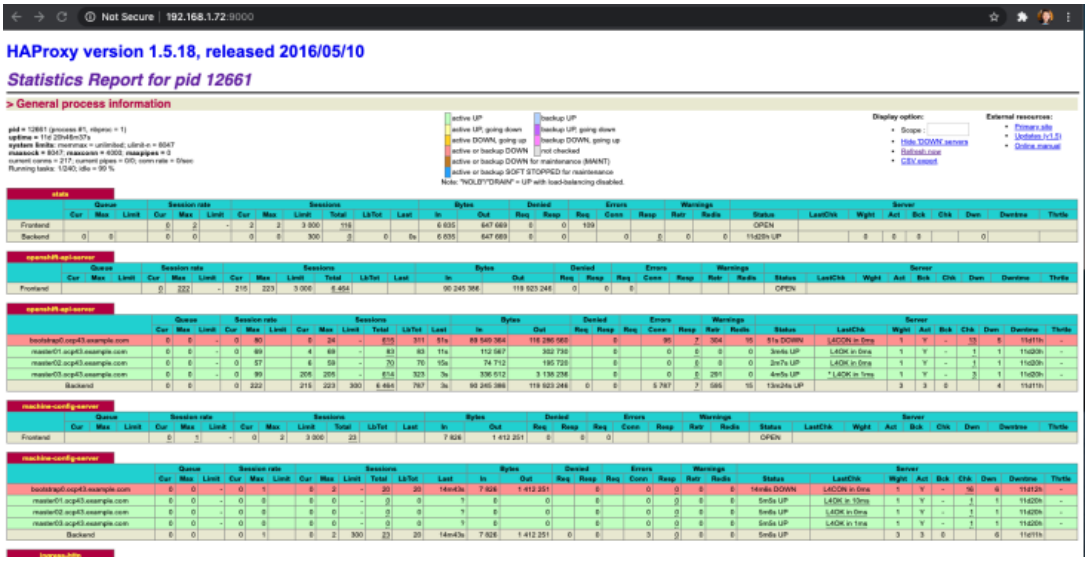
1. Monitor the cluster:

```
./openshift-install --dir=<installation_directory> wait-for bootstrap-complete --log-leve
INFO Waiting up to 30m0s for the Kubernetes API at https://api.ocp43.example.com:6443...
INFO API v1.16.2 up
INFO Waiting up to 30m0s for bootstrapping to complete...
INFO It is now safe to remove the bootstrap resources
```

2. From the bootstrap VM, similar log messages are shown:

```
$ journalctl -b -f -u bootkube.service
...
Mar 16 20:03:57 bootstrap0.ocp43.example.com bootkube.sh[2816]: Tearing down temporary bo
Mar 16 20:03:57 bootstrap0.ocp43.example.com podman[18629]: 2020-03-16 20:03:57.232567868
Mar 16 20:03:57 bootstrap0.ocp43.example.com podman[18629]: 2020-03-16 20:03:57.379721836
Mar 16 20:03:57 bootstrap0.ocp43.example.com bootkube.sh[2816]: bootkube.service complete
```

3. Load balancer status
4. Remove the bootstrap from the Load Balancer. You can check the status of LB from the status page



Logging in to the Cluster

1. Export the kubeadmiin credentials:

```
export KUBECONFIG=./ocp43/auth/kubeconfig
```

2. Verify cluster role via oc CLI

```
$ oc whoami
system:admin
```

3. Approving the CSRs

```
$ oc get nodes
NAME                         STATUS   ROLES    AGE   VERSION
master01.ocp43.example.com   Ready    master   60m   v1.16.2
master02.ocp43.example.com   Ready    master   60m   v1.16.2
master03.ocp43.example.com   Ready    master   60m   v1.16.2
worker01.ocp43.example.com   Ready    worker   52m   v1.16.2
worker02.ocp43.example.com   Ready    worker   51m   v1.16.2

$ oc get csr
NAME        AGE   REQUESTOR
csr-66l6l   60m   system:node:master02.ocp43.example.com
csr-8r2dc   52m   system:serviceaccount:openshift-machine-config-operator:node-bootstrapp
csr-hvt2d   51m   system:node:worker02.ocp43.example.com
csr-k2ggg   60m   system:node:master03.ocp43.example.com
csr-kg72s   52m   system:serviceaccount:openshift-machine-config-operator:node-bootstrapp
csr-qvbg2   60m   system:serviceaccount:openshift-machine-config-operator:node-bootstrapp
csr-rtncq   52m   system:node:worker01.ocp43.example.com
csr-tsfxx   60m   system:serviceaccount:openshift-machine-config-operator:node-bootstrapp
csr-wn7rp   60m   system:serviceaccount:openshift-machine-config-operator:node-bootstrapp
csr-zl87q   60m   system:node:master01.ocp43.example.com
```

4. If there is `pending` CSR, approve the CSR via the command below.

```
oc adm certificate approve <csr_name>
```

5. Validate the cluster components all available:

```
$ oc get co
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
authentication                             4.3.5     True        False         False
cloud-credential                           4.3.5     True        False         False
cluster-autoscaler                         4.3.5     True        False         False
console                                    4.3.5     True        False         False
dns                                        4.3.5     True        False         False
image-registry                             4.3.5     True        False         False
ingress                                    4.3.5     True        False         False
insights                                   4.3.5     True        False         False
kube-apiserver                             4.3.5     True        False         False
kube-controller-manager                    4.3.5     True        False         False
kube-scheduler                             4.3.5     True        False         False
machine-api                                4.3.5     True        False         False
machine-config                             4.3.5     True        False         False
marketplace                                4.3.5     True        False         False
monitoring                                 4.3.5     True        False         False
network                                    4.3.5     True        False         False
node-tuning                                4.3.5     True        False         False
openshift-apiserver                        4.3.5     True        False         False
openshift-controller-manager               4.3.5     True        False         False
openshift-samples                          4.3.5     True        False         False
operator-lifecycle-manager                 4.3.5     True        False         False
operator-lifecycle-manager-catalog         4.3.5     True        False         False
operator-lifecycle-manager-packageserver   4.3.5     True        False         False
service-ca                                 4.3.5     True        False         False
service-catalog-apiserver                  4.3.5     True        False         False
service-catalog-controller-manager         4.3.5     True        False         False
storage                                    4.3.5     True        False         False
```

Configure the Image Registry to use ephemeral storage for now.

I will update the image registry in the other blog since I want to focus on the completion of the installation.

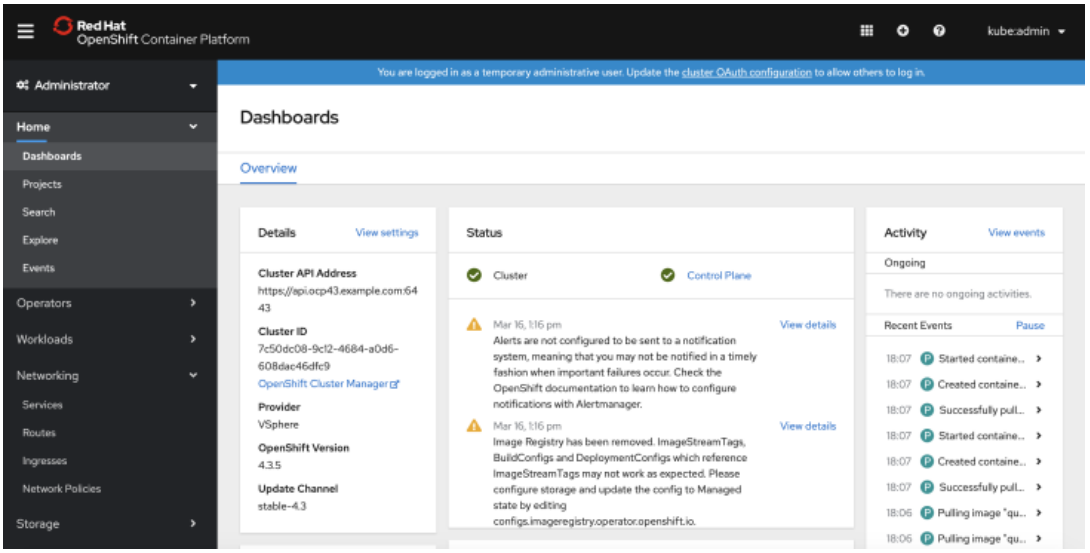
To set emptyDir for the image registry:

```
oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":{
```

Completing the installation:

```
$ ./openshift-install --dir=./ocp43 wait-for install-complete
INFO Waiting up to 30m0s for the cluster at https://api.ocp43.example.com:6443 to initialize
INFO Waiting up to 10m0s for the openshift-console route to be created...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export KUBECONFIG=
INFO Access the OpenShift web-console here: https://console-openshift-console.apps.ocp43.exa
INFO Login to the console with user: kubeadmin, password: xxxxxxxxxxxxx
```

Congratulation Cluster is up!



Troubleshoot tips:

Reference:

https://docs.openshift.com/container-platform/4.3/installing/installing_bare_metal/installing-bare-metal.html

https://docs.openshift.com/container-platform/4.3/installing/installing_vsphere/installing-vsphere.html

https://shanna-chan.blog/2019/07/26/openshift4-vsphere-static-ip/

Published by shannachan

Shanna Chan is a passionate and self driven technologist who enjoy solving problems and share knowledge with others. Strong engineering professional skilled in presales, middleware, OpenShift, Docker, Kubernetes, open source technologies, IT Strategy, DevOps, Professional Services, Java, and Platform as a Service (PaaS). View all posts by shannachan

2 thoughts on “OpenShift4.3: Retest Static IP configuration on vSphere”

Mario says:
April 6, 2020 at 2:24 pm
Excellent document Shanna…it was very useful ..Thx a lot..Ciao..Mario

Reply

shannachan says:
April 9, 2020 at 3:11 pm
You are welcome! Happy to share!

Reply