Troubleshooting

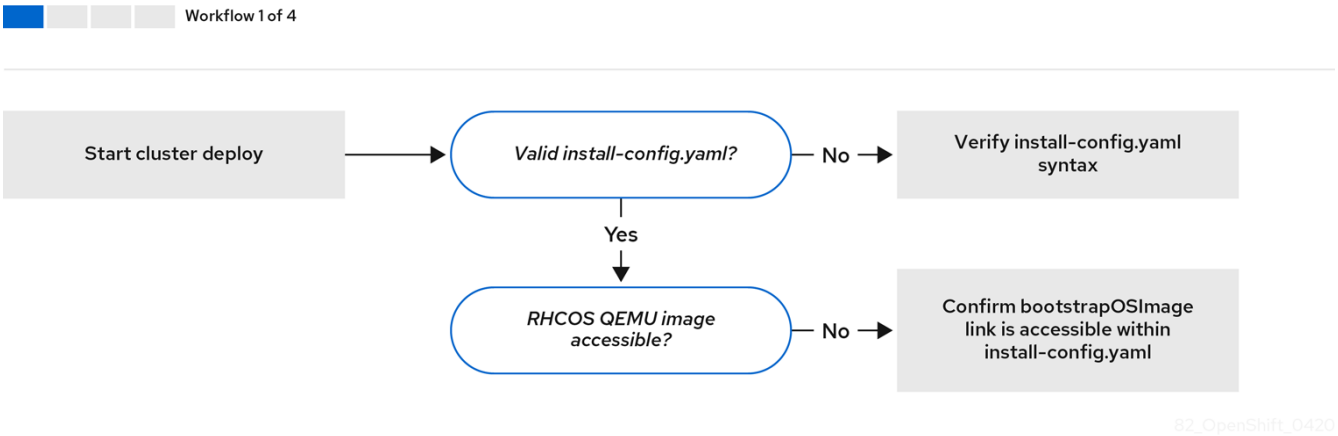
Deployment Integration Team

Download the PDF version of this document or visit https://openshift-kni.github.io/baremetal-deploy/

While attempting to deploy Installer Provisioned Infrastructure (IPI) of OpenShift on Bare Metal (BM), you may run into a situation where you need to troubleshoot your environment. This document provides troubleshooting guidance and tips in solving common issues that may arise.

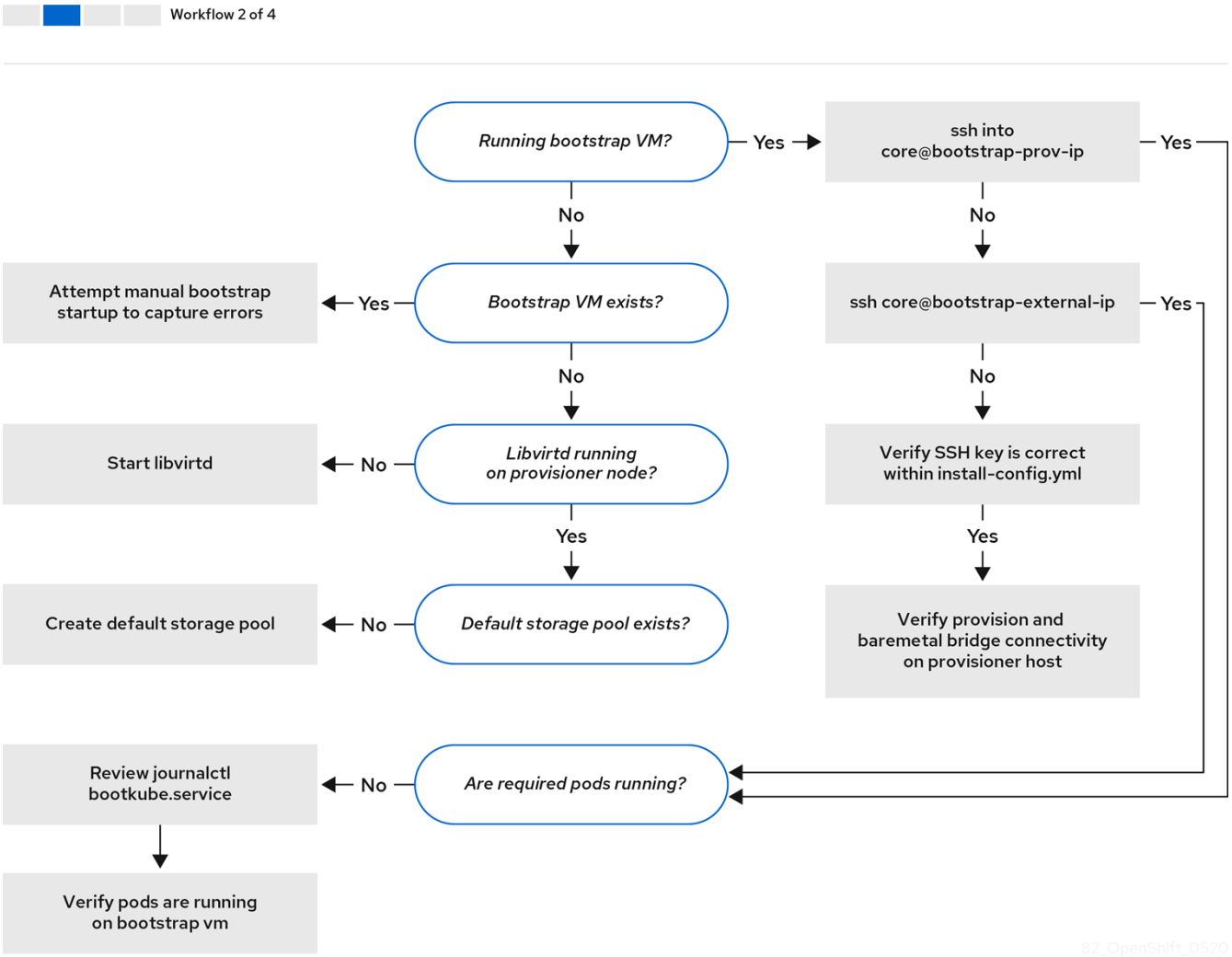
Chapter 1. Troubleshooting the installer workflow

Prior to troubleshooting the installation environment, it is critical to understand the overall flow of the IPI installation on bare metal. The diagrams below provide a troubleshooting flow with a step-by-step breakdown for the environment.

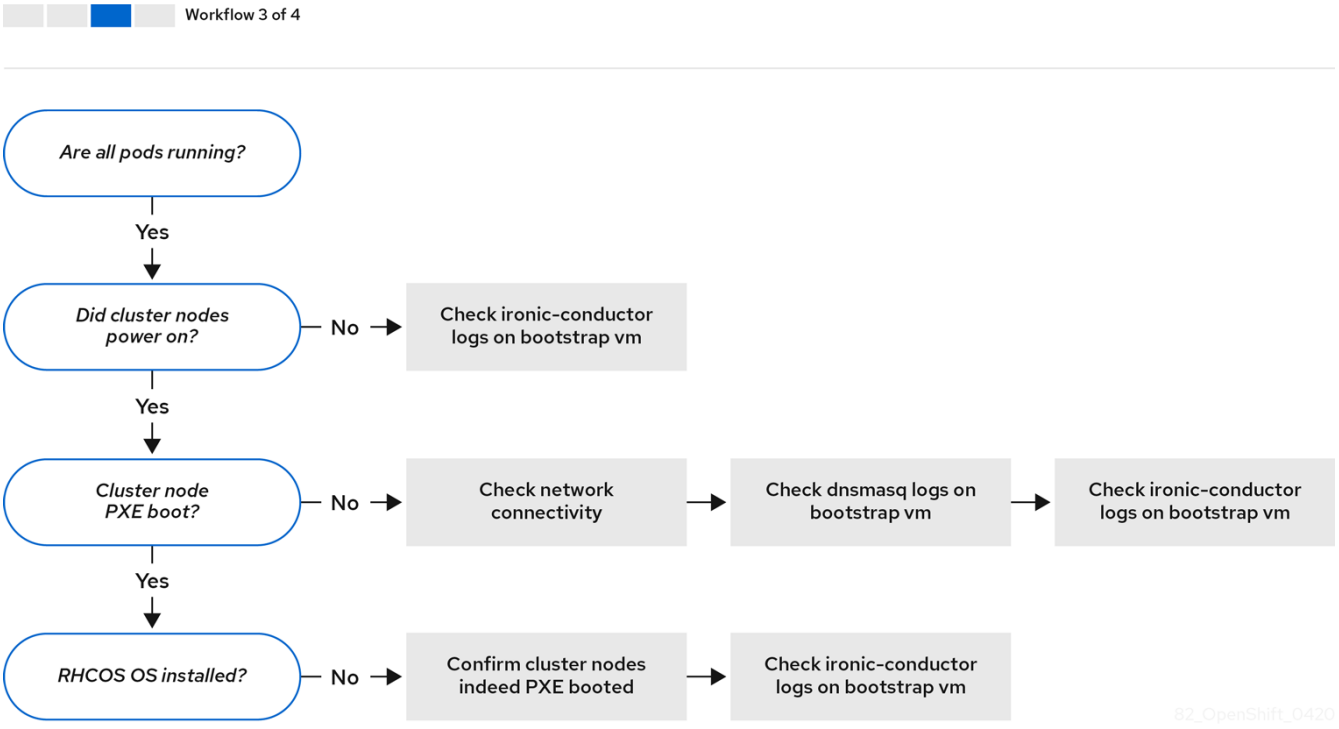


Workflow 1 of 4 illustrates a troubleshooting workflow when the `install-config.yaml` file has errors or the Red Hat Enterprise Linux CoreOS (RHCOS) images are inaccessible. Troubleshooting suggestions can be found at

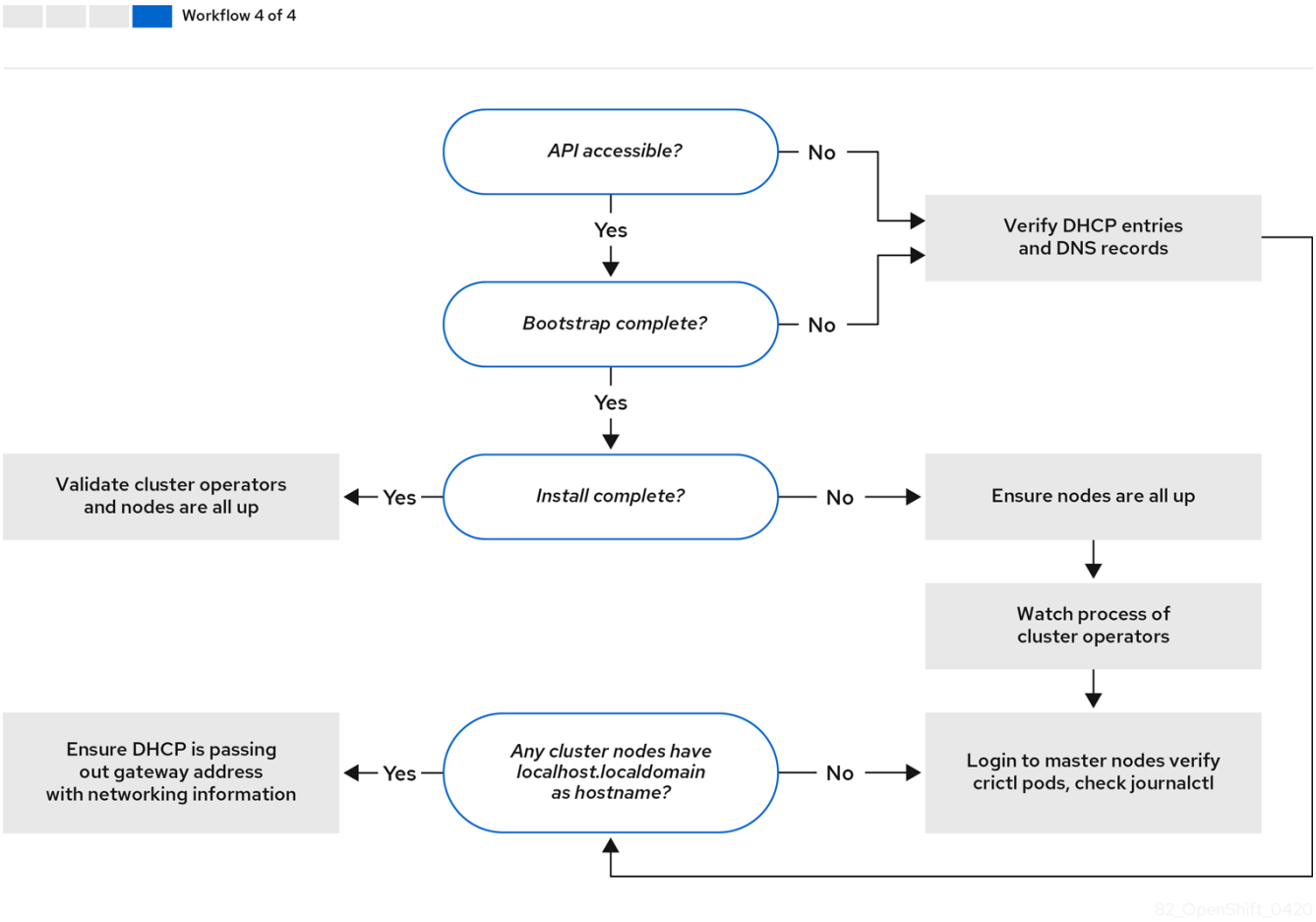
Troubleshooting `install-config.yaml.



Workflow 2 of 4 illustrates a troubleshooting workflow for bootstrap VM issues, bootstrap VMs that cannot boot up the cluster nodes, and inspecting logs.



Workflow 3 of 4 illustrates a troubleshooting workflow for cluster nodes that will not PXE boot.



Workflow 4 of 4 illustrates a troubleshooting workflow from a non-accessible API to a validated installation.

Chapter 2. Troubleshooting `install-config.yaml`

The `install-config.yaml` configuration file represents all of the nodes that are part of the OpenShift Container Platform cluster. The file contains the necessary options consisting of but not limited to `apiVersion`, `baseDomain`, `imageContentSources` and virtual IP addresses. If errors occur early in the deployment of the OpenShift Container Platform cluster, the errors are likely in the `install-config.yaml` configuration file.

Procedure

1. Use the guidelines in YAML-tips.
2. Verify the YAML syntax is correct using syntax-check.
3. Verify the Red Hat Enterprise Linux CoreOS (RHCOS) QEMU images are properly defined and accessible via the URL provided in the `install-config.yaml`. For example:

```
[kni@provisioner ~]$ curl -s -o /dev/null -I -w "%{http_code}\n"
http://webserver.example.com:8080/rhcos-44.81.202004250133-0-
qemu.x86_64.qcow2.gz?sha256=7d884b46ee54fe87bbc3893bf2aa99af3b2d31f2e19ab5529c60636
fbd0f1ce7
```

   If the output is `200`, there is a valid response from the webserver storing the bootstrap VM image.

Chapter 3. Bootstrap VM issues

The OpenShift Container Platform installer spawns a bootstrap node virtual machine, which handles provisioning the OpenShift Container Platform cluster nodes.

Procedure

1. About 10 to 15 minutes after triggering the installer, check to ensure the bootstrap VM is operational using the `virsh` command:

```
[kni@provisioner ~]$ sudo virsh list
```

```
 Id    Name                           State
----------------------------------------------
 12    openshift-xf6fq-bootstrap      running
```

The name of the bootstrap VM is always the cluster name followed by random set of characters and ending in the word "bootstrap."

If the bootstrap VM is not running after 10-15 minutes, troubleshoot why it was not created. Possible issues include:

1. Verify `libvirtd` is running on the system:

```
[kni@provisioner ~]$ systemctl status libvirtd
```

```
● libvirtd.service - Virtualization daemon
   Loaded: loaded (/usr/lib/systemd/system/libvirtd.service; enabled; vendor
preset: enabled)
   Active: active (running) since Tue 2020-03-03 21:21:07 UTC; 3 weeks 5 days ago
     Docs: man:libvirtd(8)
           https://libvirt.org
 Main PID: 9850 (libvirtd)
    Tasks: 20 (limit: 32768)
   Memory: 74.8M
   CGroup: /system.slice/libvirtd.service
           ├─ 9850 /usr/sbin/libvirtd
```

   If the bootstrap VM is operational, log into it.

2. Use the `virsh console` command to find the IP address of the bootstrap VM:

```
[kni@provisioner ~]$ sudo virsh console example.com
```

```
Connected to domain example.com
Escape character is ^]

Red Hat Enterprise Linux CoreOS 43.81.202001142154.0 (Ootpa) 4.3
SSH host key: SHA256:BRWJktXZgQQRY5zjuAV0IKZ4WM7i4TiUyMVanqu9Pqg (ED25519)
SSH host key: SHA256:7+iKGA7VtG5szmk2jB5gl/5EZ+SNcJ3a2g23o0lnIio (ECDSA)
SSH host key: SHA256:DH5VWhvhvagOTaLsYiVNse9ca+ZSW/30OOMed8rIGOc (RSA)
ens3:  fd35:919d:4042:2:c7ed:9a9f:a9ec:7
ens4: 172.22.0.2 fe80::1d05:e52e:be5d:263f
localhost login:
```

3. Once you obtain the IP address, log in to the bootstrap VM using the `ssh` command:

In the console output of the previous step, the IPv6 IP provided by `ens3` or the IPv4 IP provided by `ens4` can be used.

```
[kni@provisioner ~]$ ssh core@172.22.0.2
```

If you are not successful logging in to the bootstrap VM, you have likely encountered of the following scenarios:

- You cannot reach the `172.22.0.0/24` network. Verify network connectivity on the `provisioner` host specifically around the `provisioning` network bridge.
- You cannot reach the bootstrap VM via the public network. When attempting to SSH via `baremetal` network, verify connectivity on the `provisioner` host specifically around the `baremetal` network bridge.
- You encountered `Permission denied (publickey,password,keyboard-interactive)`. When attempting to access the bootstrap VM a `Permission denied` error might occur. Verify that the SSH key for the user attempting to log into the VM is set within the `install-config.yaml` file.

3.1. Bootstrap VM cannot boot up the cluster nodes

During the deployment, it is possible for the bootstrap VM to fail to boot the cluster nodes, which prevents the VM from provisioning the nodes with the RHCOS image. This scenario can arise due to:

- A problem with the `install-config.yaml` file.
- Issues with out-of-band network access via the baremetal network.

To verify the issue, there are three containers related to `ironic`:

- `ironic-api`
- `ironic-conductor`
- `ironic-inspector`

Procedure

1. Log in to the bootstrap VM:

```
[kni@provisioner ~]$ ssh core@172.22.0.2
```

2. To check the container logs, execute the following:

```
[core@localhost ~]$ sudo podman logs -f <container-name>
```

   Replace `<container-name>` with one of `ironic-api`, `ironic-conductor`, or `ironic-inspector`. If you encounter an issue where the master nodes are not booting up via PXE, check the `ironic-conductor` Pod. The `ironic-conductor` Pod contains the most detail about the attempt to boot the cluster nodes, because it attempts to log in to the node over IPMI.

Potential reason

The cluster nodes might be in the `ON` state when deployment started.

Solution

Power off the OpenShift Container Platform cluster nodes before you begin the installation over IPMI:

```
[kni@provisioner ~]$ ipmitool -I lanplus -U root -P <password> -H <out-of-band-ip>
power off
```

3.2. Inspecting logs

When experiencing issues downloading or accessing the RHCOS images, first verify that the URL is correct in the `install-config.yaml` configuration file.

Example of internal webserver hosting RHCOS images

```
bootstrapOSImage: http://<ip:port>/rhcos-43.81.202001142154.0-
qemu.x86_64.qcow2.gz?sha256=9d999f55ff1d44f7ed7c106508e5deecd04dc3c06095d34d36bf1cd127
837e0c
clusterOSImage: http://<ip:port>/rhcos-43.81.202001142154.0-
openstack.x86_64.qcow2.gz?sha256=a1bda656fa0892f7b936fdc6b6a6086bddaed5dafacedcd7a1e81
1abb78fe3b0
```

The `ipa-downloader` and `coreos-downloader` containers download resources from a webserver or the external quay.io registry, whichever is specified in the `install-config.yaml` configuration file. Verify the following two containers are up and running and inspect their logs as needed:

- `ipa-downloader`
- `coreos-downloader`

Procedure

1. Log in to the bootstrap VM:

```
[kni@provisioner ~]$ ssh core@172.22.0.2
```

2. Check the status of the `ipa-downloader` and `coreos-downloader` containers within the bootstrap VM:

```
[core@localhost ~]$ podman logs -f ipa-downloader
```

```
[core@localhost ~]$ podman logs -f coreos-downloader
```

   If the bootstrap VM cannot access the URL to the images, use the `curl` command to verify that the VM can access the images.

3. To inspect the `bootkube` logs that indicate if all the containers launched during the deployment phase, execute the following:

```
[core@localhost ~]$ journalctl -xe
```

```
[core@localhost ~]$ journalctl -b -f -u bootkube.service
```

4. Verify all the Pods, including `dnsmasq`, `mariadb`, `httpd`, and `ironic`, are running:

```
[core@localhost ~]$ sudo podman ps
```

5. If there are issues with the Pods, check the logs of the containers with issues. To check the log of the `ironic-api`, execute the following:

```
[core@localhost ~]$ sudo podman logs <ironic-api>
```

Chapter 4. Cluster nodes will not PXE boot

When OpenShift Container Platform cluster nodes will not PXE boot, execute the following checks on the cluster nodes that will not PXE boot.

Procedure

1. Check the network connectivity to the `provisioning` network.
2. Ensure PXE is enabled on the NIC for the `provisioning` network and PXE is disabled for all other NICs.
3. Verify that the `install-config.yaml` configuration file has the proper hardware profile and boot MAC address for the NIC connected to the `provisioning` network. For example:

   Master node settings

   ```
   bootMACAddress: 24:6E:96:1B:96:90 # MAC of bootable provisioning NIC
   hardwareProfile: default          #master node settings
   ```

   Worker node settings

   ```
   bootMACAddress: 24:6E:96:1B:96:90 # MAC of bootable provisioning NIC
   hardwareProfile: unknown          #worker node settings
   ```

Chapter 5. The API is not accessible

When the cluster is running and clients cannot access the API, domain name resolution issues might impede access to the API.

Procedure

1. **Hostname Resolution:** Check the cluster nodes to ensure they have a fully qualified domain name, and not just `localhost.localdomain`. For example:

   ```
   [kni@provisioner ~]$ hostname
   ```

   If a hostname is not set, set the correct hostname. For example:

   ```
   [kni@provisioner ~]$ hostnamectl set-hostname <hostname>
   ```

2. **Incorrect Name Resolution:** Ensure that each node has the correct name resolution in the DNS server using `dig` and `nslookup`. For example:

   ```
   [kni@provisioner ~]$ dig api.<cluster-name>.example.com
   ```

   ```
   ; <<>> DiG 9.11.4-P2-RedHat-9.11.4-26.P2.el8 <<>> api.<cluster-name>.example.com
   ;; global options: +cmd
   ;; Got answer:
   ;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 37551
   ;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

   ;; OPT PSEUDOSECTION:
   ; EDNS: version: 0, flags:; udp: 4096
   ; COOKIE: 866929d2f8e8563582af23f05ec44203d313e50948d43f60 (good)
   ;; QUESTION SECTION:
   ;api.<cluster-name>.example.com. IN A

   ;; ANSWER SECTION:
   api.<cluster-name>.example.com. 10800 IN  A 10.19.13.86

   ;; AUTHORITY SECTION:
   <cluster-name>.example.com. 10800 IN NS   <cluster-name>.example.com.

   ;; ADDITIONAL SECTION:
   <cluster-name>.example.com. 10800 IN A    10.19.14.247

   ;; Query time: 0 msec
   ;; SERVER: 10.19.14.247#53(10.19.14.247)
   ;; WHEN: Tue May 19 20:30:59 UTC 2020
   ;; MSG SIZE  rcvd: 140
   ```

The output in the foregoing example indicates that the appropriate IP address for the `api.<cluster-name>.example.com` VIP is `10.19.13.86`. This IP address should reside on the `baremetal` network.

Chapter 6. Cleaning up previous installations

In the event of a previous failed deployment, remove the artifacts from the failed attempt before attempting to deploy OpenShift Container Platform again.

Procedure

1. Power off all bare metal nodes prior to installing the OpenShift Container Platform cluster:

```
[kni@provisioner ~]$ ipmitool -I lanplus -U <user> -P <password> -H <management-
server-ip> power off
```

2. Remove all old bootstrap resources if any are left over from a previous deployment attempt:

```
for i in $(sudo virsh list | tail -n +3 | grep bootstrap | awk {'print $2'});
do
  sudo virsh destroy $i;
  sudo virsh undefine $i;
  sudo virsh vol-delete $i --pool default;
  sudo virsh vol-delete $i.ign --pool default;
done
```

3. Remove the following from the `clusterconfigs` directory to prevent Terraform from failing:

```
[kni@provisioner ~]$ rm -rf ~/clusterconfigs/auth ~/clusterconfigs/terraform*
~/clusterconfigs/tls ~/clusterconfigs/metadata.json
```

Chapter 7. Issues with creating the registry

When creating a disconnected registry, you might encounter a "User Not Authorized" error when attempting to mirror the registry. This error might occur if you fail to append the new authentication to the existing `pull-secret.txt` file.

Procedure

1. Check to ensure authentication is successful:

```
[user@registry ~]$ /usr/local/bin/oc adm release mirror \
  -a pull-secret-update.json
  --from=$UPSTREAM_REPO \
  --to-release-image=$LOCAL_REG/$LOCAL_REPO:${VERSION} \
  --to=$LOCAL_REG/$LOCAL_REPO
```

Example output of the variables used to mirror the install images:

```
UPSTREAM_REPO=${RELEASE_IMAGE}
LOCAL_REG=<registry_FQDN>:<registry_port>
LOCAL_REPO='ocp4/openshift4'
```

The values of `RELEASE_IMAGE` and `VERSION` were set during the **Retrieving OpenShift Installer** step of the **Setting up the environment for an OpenShift installation** section.

2. After mirroring the registry, confirm that you can access it in your disconnected environment:

```
[kni@provisioner ~]$ curl -k -u <user>:<password>
https://registry.example.com:<registry-port>/v2/_catalog
{"repositories":["<Repo-Name>"]}
```

Chapter 8. Miscellaneous issues

8.1. Addressing the `runtime network not ready` error

After the deployment of a cluster you might receive the following error:

```
`runtime network not ready: NetworkReady=false reason:NetworkPluginNotReady
message:Network plugin returns error: Missing CNI default network`
```

The Cluster Network Operator is responsible for deploying the networking components in response to a special object created by the installer. It runs very early in the installation process, after the Control Plane (master) nodes have come up, but before the bootstrap control plane has been torn down. It can be indicative of more subtle installer issues, such as long delays in bringing up Control Plane (master) nodes or issues with `apiserver` communication.

Procedure

1. Inspect the Pods in the `openshift-network-operator` namespace:

```
[kni@provisioner ~]$ oc get all -n openshift-network-operator
```

```
NAME                                    READY STATUS            RESTARTS   AGE
pod/network-operator-69dfd7b577-bg89v   0/1   ContainerCreating 0          149m
```

2. On the `provisioner` node, determine that the network configuration exists:

```
[kni@provisioner ~]$ kubectl get network.config.openshift.io cluster -oyaml
```

```
apiVersion: config.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  serviceNetwork:
  - 172.30.0.0/16
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  networkType: OpenShiftSDN
```

   If it does not exist, the installer did not create it. To determine why the installer did not create it, execute the following:

```
[kni@provisioner ~]$ openshift-install create manifests
```

3. Check that the `network-operator` is running:

```
[kni@provisioner ~]$ kubectl -n openshift-network-operator get pods
```

4. Retrieve the logs:

```
[kni@provisioner ~]$ kubectl -n openshift-network-operator logs -l "name=network-operator"
```

On high availability clusters with three or more Control Plane (master) nodes, the Operator will perform leader election and all other Operators will sleep. For additional details, see Troubleshooting.

8.2. Cluster nodes not getting the correct IPv6 address over DHCP

If the cluster nodes are not getting the correct IPv6 address over DHCP, check the following:

1. Ensure the reserved IPv6 addresses reside outside the DHCP range.
2. In the IP address reservation on the DHCP server, ensure the reservation specifies the correct DHCP Unique Identifier (DUID). For example:

```
# This is a dnsmasq dhcp reservation, 'id:00:03:00:01' is the client id and
'18:db:f2:8c:d5:9f' is the MAC Address for the NIC
id:00:03:00:01:18:db:f2:8c:d5:9f,openshift-master-1,[2620:52:0:1302::6]
```

3. Ensure that Route Announcements are working.
4. Ensure that the DHCP server is listening on the required interfaces serving the IP address ranges.

8.3. Cluster nodes not getting the correct hostname over DHCP

During IPv6 deployment, cluster nodes must get their hostname over DHCP. Sometimes the `NetworkManager` does not assign the hostname immediately. A Control Plane (master) node might report an error such as:

```
Failed Units: 2
  NetworkManager-wait-online.service
  nodeip-configuration.service
```

This error indicates that the cluster node likely booted without first receiving a hostname from the DHCP server, which causes `kubelet` to boot with a `localhost.localdomain` hostname. To address the error, force the node to renew the hostname.

Procedure

1. Retrieve the `hostname`:

   ```
   [core@master-X ~]$ hostname
   ```

   If the hostname is `localhost`, proceed with the following steps.

   Where `X` is the master node number.

2. Force the cluster node to renew the DHCP lease:

   ```
   [core@master-X ~]$ sudo nmcli con up "<bare-metal-nic>"
   ```

   Replace `<bare-metal-nic>` with the wired connection corresponding to the `baremetal` network.

3. Check `hostname` again:

   ```
   [core@master-X ~]$ hostname
   ```

4. If the hostname is still `localhost.localdomain`, restart `NetworkManager`:

   ```
   [core@master-X ~]$ sudo systemctl restart NetworkManager
   ```

5. If the hostname is still `localhost.localdomain`, wait a few minutes and check again. If the hostname remains `localhost.localdomain`, repeat the previous steps.
6. Restart the `nodeip-configuration` service:

   ```
   [core@master-X ~]$ sudo systemctl restart nodeip-configuration.service
   ```

   This service will reconfigure the `kubelet` service with the correct hostname references.

7. Reload the unit files definition since the kubelet changed in the previous step:

```
[core@master-X ~]$ sudo systemctl daemon-reload
```

8. Restart the `kubelet` service:

```
[core@master-X ~]$ sudo systemctl restart kubelet.service
```

9. Ensure `kubelet` booted with the correct hostname:

```
[core@master-X ~]$ sudo journalctl -fu kubelet.service
```

If the cluster node is not getting the correct hostname over DHCP after the cluster is up and running, such as during a reboot, the cluster will have a pending `csr`. **Do not** approve a `csr`, or other issues might arise.

Addressing a `csr`

1. Get CSRs on the cluster:

```
[kni@provisioner ~]$ oc get csr
```

2. Verify if a pending `csr` contains `Subject Name: localhost.localdomain`:

```
[kni@provisioner ~]$ oc get csr <pending_csr> -o jsonpath='{.spec.request}' |
base64 -d | openssl req -noout -text
```

3. Remove any `csr` that contains `Subject Name: localhost.localdomain`:

```
[kni@provisioner ~]$ oc delete csr <wrong_csr>
```

8.4. Routes do not reach endpoints

During the installation process, it is possible to encounter a Virtual Router Redundancy Protocol (VRRP) conflict. This conflict might occur if a previously used OpenShift Container Platform node that was once part of a cluster deployment using a specific cluster name is still running but not part of the current OpenShift Container Platform cluster deployment using that same cluster name. For example, a cluster was deployed using the cluster name `openshift`, deploying three Control Plane (master) nodes and three worker nodes. Later, a separate install uses the same cluster name `openshift`, but this redeployment only installed three Control Plane (master) nodes, leaving the three worker nodes from a previous deployment in an `ON` state. This might cause a Virtual Router IDentifier (VRID) conflict and a VRRP conflict.

1. Get the route:

```
[kni@provisioner ~]$ oc get route oauth-openshift
```

2. Check the service endpoint:

```
[kni@provisioner ~]$ oc get svc oauth-openshift
```

```
NAME              TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
oauth-openshift   ClusterIP   172.30.19.162   <none>        443/TCP   59m
```

3. Attempt to reach the service from a Control Plane (master) node:

```
[core@master0 ~]$ curl -k https://172.30.19.162
```

```
{
  "kind": "Status",
  "apiVersion": "v1",
  "metadata": {
  },
  "status": "Failure",
  "message": "forbidden: User \"system:anonymous\" cannot get path \"/\"",
  "reason": "Forbidden",
  "details": {
  },
  "code": 403
```

4. Identify the `authentication-operator` errors from the `provisioner` node:

```
[kni@provisioner ~]$ oc logs deployment/authentication-operator -n openshift-
authentication-operator
```

```
Event(v1.ObjectReference{Kind:"Deployment", Namespace:"openshift-authentication-
operator", Name:"authentication-operator", UID:"225c5bd5-b368-439b-9155-
5fd3c0459d98", APIVersion:"apps/v1", ResourceVersion:"", FieldPath:""}): type:
'Normal' reason: 'OperatorStatusChanged' Status for clusteroperator/authentication
changed: Degraded message changed from "IngressStateEndpointsDegraded: All 2
endpoints for oauth-server are reporting"
```

Solution

1. Ensure that the cluster name for every deployment is unique, ensuring no conflict.
2. Turn off all the rogue nodes which are not part of the cluster deployment that are using the same cluster name. Otherwise, the authentication Pod of the OpenShift Container Platform

cluster might never start successfully.

8.5. Failed Ignition during Firstboot

During the Firstboot, the Ignition configuration may fail.

Procedure

1. Connect to the node where the Ignition configuration failed:

```
Failed Units: 1
  machine-config-daemon-firstboot.service
```

2. Restart the `machine-config-daemon-firstboot` service:

```
[core@worker-X ~]$ sudo systemctl restart machine-config-daemon-firstboot.service
```

8.6. NTP out of sync

The deployment of OpenShift Container Platform clusters depends on NTP synchronized clocks among the cluster nodes. Without synchronized clocks, the deployment may fail due to clock drift if the time difference is greater than two seconds.

Procedure

1. Check for differences in the `AGE` of the cluster nodes. For example:

```
[kni@provisioner ~]$ oc get nodes
```

```
NAME                         STATUS   ROLES    AGE   VERSION
master-0.cloud.example.com   Ready    master   145m   v1.16.2
master-1.cloud.example.com   Ready    master   135m   v1.16.2
master-2.cloud.example.com   Ready    master   145m   v1.16.2
worker-2.cloud.example.com   Ready    worker   100m   v1.16.2
```

2. Check for inconsistent timing delays due to clock drift. For example:

```
[kni@provisioner ~]$ oc get bmh -n openshift-machine-api
```

```
master-1   error registering master-1  ipmi://<out-of-band-ip>
```

```
[kni@provisioner ~]$ sudo timedatectl
```

```
               Local time: Tue 2020-03-10 18:20:02 UTC
           Universal time: Tue 2020-03-10 18:20:02 UTC
                 RTC time: Tue 2020-03-10 18:36:53
                Time zone: UTC (UTC, +0000)
System clock synchronized: no
              NTP service: active
          RTC in local TZ: no
```

Addressing clock drift in existing clusters

1. Create a `chrony.conf` file and encode it as `base64` string. For example:

```
[kni@provisioner ~]$ cat << EOF | base 64
server <NTP-server> iburst①
stratumweight 0
driftfile /var/lib/chrony/drift
rtcsync
makestep 10 3
bindcmdaddress 127.0.0.1
bindcmdaddress ::1
keyfile /etc/chrony.keys
commandkey 1
generatecommandkey
noclientlog
logchange 0.5
logdir /var/log/chrony
EOF
```

① Replace `<NTP-server>` with the IP address of the NTP server. Copy the output.

```
[text-in-base-64]
```

2. Create a `MachineConfig` file, replacing the `base64` string with the `[text-in-base-64]` string generated in the output of the previous step. The following example adds the file to the Control Plane (master) nodes. You can modify the file for worker nodes or make an additional `MachineConfig` for the worker role.

```
[kni@provisioner ~]$ cat << EOF > ./99_masters-chrony-configuration.yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  creationTimestamp: null
  labels:
    machineconfiguration.openshift.io/role: master
  name: 99-master-etc-chrony-conf
spec:
  config:
    ignition:
      config: {}
      security:
        tls: {}
      timeouts: {}
      version: 2.2.0
    networkd: {}
    passwd: {}
    storage:
      files:
      - contents:
          source: data:text/plain;charset=utf-8;base64,[text-in-base-64]①
          verification: {}
        filesystem: root
        group:
          name: root
        mode: 420
        path: /etc/chrony.conf
        user:
          name: root
    systemd: {}
  osImageURL: ""
```

① Replace `[text-in-base-64]` with the base64 string.

3. Make a backup copy of the configuration file. For example:

```
[kni@provisioner ~]$ cp 99_masters-chrony-configuration.yaml 99_masters-chrony-
configuration.yaml.backup
```

4. Apply the configuration file:

```
[kni@provisioner ~]$ oc apply -f ./masters-chrony-configuration.yaml
```

5. Ensure the `System clock synchronized` value is **yes**:

```
[kni@provisioner ~]$ sudo timedatectl
```

```
               Local time: Tue 2020-03-10 19:10:02 UTC
           Universal time: Tue 2020-03-10 19:10:02 UTC
                 RTC time: Tue 2020-03-10 19:36:53
                Time zone: UTC (UTC, +0000)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
```

To setup clock synchronization prior to deployment, generate the manifest files and add this file to the `openshift` directory. For example:

```
[kni@provisioner ~]$ cp chrony-masters.yaml ~/clusterconfigs/openshift/99_masters-
chrony-configuration.yaml
```

Then, continue to create the cluster.

Chapter 9. Reviewing the installation

After installation, ensure the installer deployed the nodes and pods successfully.

Procedure

1. When the OpenShift Container Platform cluster nodes are installed appropriately, the following `Ready` state is seen within the `STATUS` column:

```
[kni@provisioner ~]$ oc get nodes
```

```
NAME                   STATUS   ROLES           AGE  VERSION
master-0.example.com   Ready    master,worker   4h   v1.16.2
master-1.example.com   Ready    master,worker   4h   v1.16.2
master-2.example.com   Ready    master,worker   4h   v1.16.2
```

2. Confirm all pods are successfully deployed. The following command removes any pods that are still running or have completed as part of the output.

```
[kni@provisioner ~]$ oc get pods --all-namespaces | grep -iv running | grep -iv
complete
```