IBM Storage Ceph 7.1

Installation Guide 2024-05-16

Important

DRAFT BETA PROVIDED WITH NO SUPPORT: Do not upgrade a production installation to a beta release or from a beta release. Upgrade support is not provided between beta releases. No automated upgrade path is provided from a beta release to the final GA release. The cluster will need to be rebuilt for production use.

⚠ **Attention:** When copying and pasting commands from the beta PDF, hyphens at the end of the lines get removed and the content no longer pastes as expected, creating broken commands and links. This specifically occurs in longer commands, such as the **`curl https://public.dhe.ibm.com/ibmdl/export/pub/storage/ceph/beta/ibm-storage-ceph-7-beta-rhel-9.repo`** command. Where the `ibm-storage` copies and pastes as `ibmstorage`.

Be sure that when copying and pasting commands, they paste as is written in this document, and add in any necessary hyphens.

List of known issues

- BZ 2266035
- BZ 2270193
- BZ 2276038
- BZ 2277830
- BZ 2279339
- BZ 2279461

Last updated: 2024-05-16

Contents

Figures

Tables

Installing

This information provides instructions on installing IBM Storage Ceph on Red Hat Enterprise Linux running on AMD64 and Intel 64 architectures.

For more information, see the **Day 1 and Day 2 operations** chapter, within the IBM Storage Ceph Solutions Guide Redpaper publication.

Installing Pro Edition for free

Install IBM Storage Ceph Pro Edition for a free 60-day trial.

About this task

You can install an IBM Storage Ceph trial version by following the steps provided.

Procedure

1. Acquire and install Red Hat Enterprise Linux (RHEL) 9.
   a. Create and or log in to your Red Hat account.
   b. Acquire a Red Hat Enterprise Linux 9 subscription from Red Hat for free here.
      i) Confirm that your version meets the Operating system requirements for IBM Storage Ceph 7 or later.
   c. Install Red Hat Enterprise Linux 9 on your storage nodes or hosts.
2. Install IBM Storage Ceph Pro Edition.
   a. Register for a free 60-day IBM Storage Ceph Pro trial here.
   b. Log in to your account with the username and password from the previous step and obtain your IBM entitlement key from the container software library in My IBM.
   c. Register your Red Hat Enterprise Linux nodes and enable the IBM Storage Ceph repositories by following the steps mentioned here.
   d. Run **`dnf install cephadm -y`** command to install `cephadm`.
   e. Run the preflight playbook.
   f. Bootstrap the IBM Storage Ceph cluster that uses `cephadm`.
   g. Use the dashboard to expand your cluster with OSD, Ceph Object Gateway, S3, and other daemons and services.

Initial installation

As a storage administrator, you can use the `cephadm` utility to deploy new IBM Storage Ceph clusters.

About this task

The `cephadm` utility manages the entire life cycle of a Ceph cluster. Installation and management tasks comprise two types of operations:

- Day One operations involve installing and bootstrapping a bare-minimum, containerized Ceph storage cluster, running on a single node. Day One also includes deploying the Monitor and Manager daemons and adding Ceph OSDs.
- Day Two operations use the Ceph orchestration interface, `cephadm orch`, or the IBM Storage Ceph Dashboard to expand the storage cluster by adding other Ceph services to the storage cluster.

Before you begin

- At least one running virtual machine (VM) or bare-metal server with an active internet connection.
- Red Hat Enterprise Linux 9.2 with `ansible-core` bundled into AppStream.
- A valid IBM subscription with the appropriate entitlements.
- Root-level access to all nodes.
- An active IBM Network or service account to access the IBM Registry.
- Remove troubling configurations in iptables so that refresh of iptables services does not cause issues to the cluster. For an example, see Verifying firewall rules are configured for default Ceph ports.

For the latest supported Red Hat Enterprise Linux versions, see Compatibility matrix.

`cephadm` utility

The `cephadm` utility deploys and manages a Ceph storage cluster. It is tightly integrated with both the command-line interface (CLI) and the IBM Storage Ceph Dashboard web interface so that you can manage storage clusters from either environment. `cephadm` uses SSH to connect to hosts from the manager daemon to add, remove, or update Ceph daemon containers. It does not rely on external configuration or orchestration tools such as Ansible or Rook.

Note: The `cephadm` utility is available after running the preflight playbook on a host.

The `cephadm` utility consists of two main components:

- The `cephadm` shell.
- The `cephadm` orchestrator.

The `cephadm` shell

The `cephadm` shell starts a `bash` shell within a container. Use the shell to complete “Day One” cluster setup tasks, such as installation and bootstrapping, and to use `ceph` commands.

For more information about how to start the `cephadm` shell, see “Starting the cephadm shell” on page 38.

The `cephadm` orchestrator

Use the `cephadm` orchestrator to perform “Day Two” Ceph functions, such as expanding the storage cluster and provisioning Ceph daemons and services. You can use the `cephadm` orchestrator through either the command-line interface (CLI) or the web-based IBM Storage Ceph Dashboard. Orchestrator commands take the form `ceph orch`.

The `cephadm` script interacts with the Ceph orchestration module that is used by the Ceph Manager.

How `cephadm` works

Use the `cephadm` command to manage the full lifecycle of an IBM Storage Ceph cluster.

The `cephadm` command can perform the following operations:

- Bootstrap a new IBM Storage Ceph cluster.
- Launch a containerized shell that works with the IBM Storage Ceph command-line interface (CLI).
- Aid in debugging containerized daemons.

The `cephadm` command uses `ssh` to communicate with the nodes in the storage cluster. This allows you to add, remove, or update IBM Storage Ceph containers without using external tools. Generate the `ssh` key pair during the bootstrapping process, or use your own `ssh` key.

The `cephadm` bootstrapping process creates a small storage cluster on a single node, consisting of one Ceph Monitor and one Ceph Manager, as well as any required dependencies. You then use the orchestrator CLI or the IBM Storage Ceph Dashboard to expand the storage cluster to include nodes, and to provision all of the IBM Storage Ceph daemons and services. You can perform management functions through the CLI or from the IBM Storage Ceph Dashboard web interface.

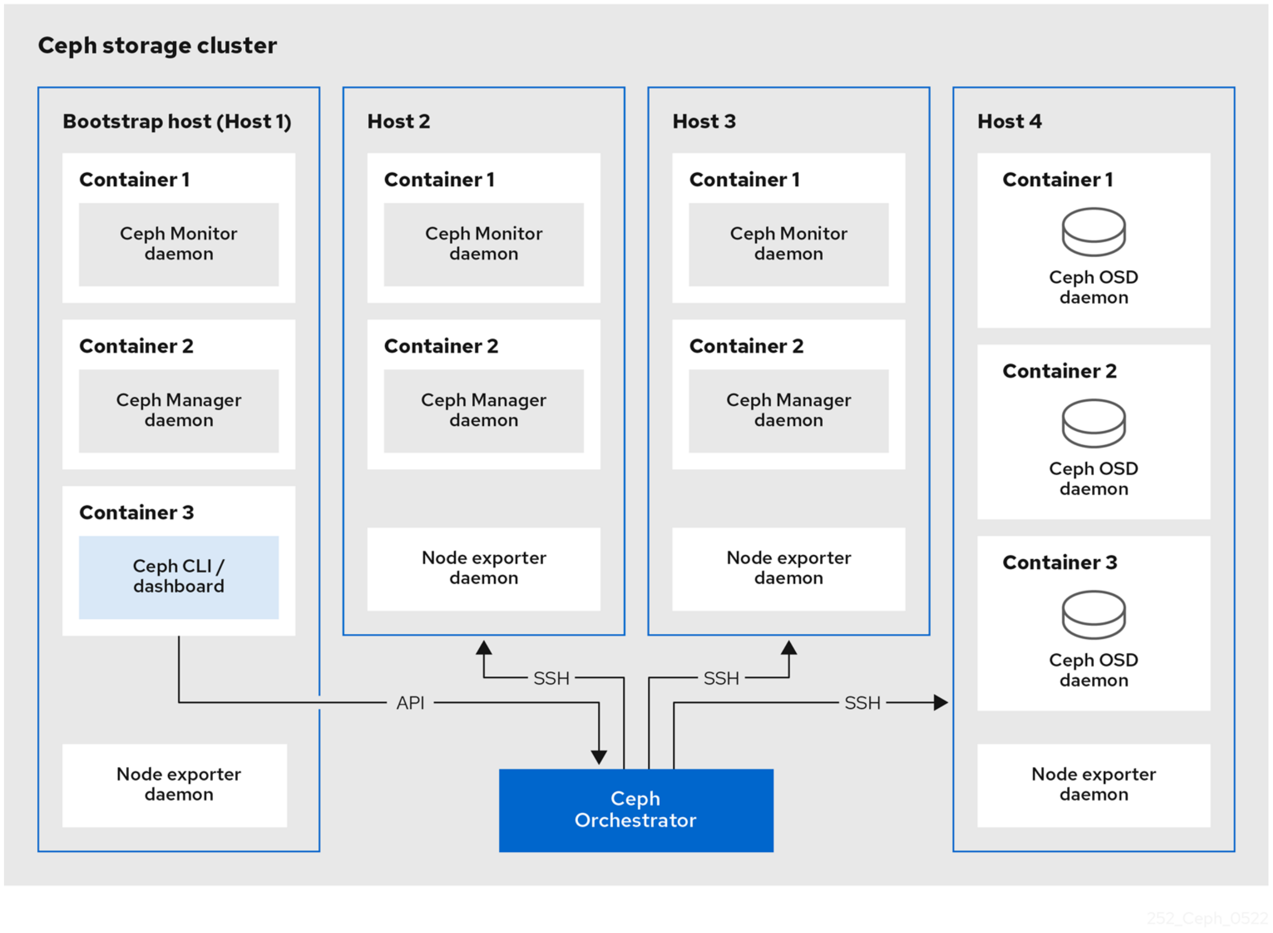

Figure 1. Ceph storage cluster deployment

`cephadm-ansible` playbooks

The `cephadm-ansible` package is a collection of Ansible playbooks to simplify workflows that are not covered by `cephadm`. After installation, the playbooks are located in `/usr/share/cephadm-ansible/`.

The `cephadm-ansible` package includes the following playbooks:

- `cephadm-preflight.yml`
- `cephadm-clients.yml`
- `cephadm-purge-cluster.yml`

The `cephadm-preflight` playbook

Use the `cephadm-preflight` playbook to initially setup hosts before bootstrapping the storage cluster and before adding new nodes or clients to your storage cluster. This playbook configures the Ceph repository and installs some prerequisites such as `podman`, `lvm2`, `chronyd`, and `cephadm`.

For more information, see Running the preflight playbook.

The `cephadm-clients` playbook

Use the `cephadm-clients` playbook to set up client hosts. This playbook handles the distribution of configuration and keyring files to a group of Ceph clients.

For more information, see "Deploying client nodes" on page 61.

The `cephadm-purge-cluster` playbook

Use the `cephadm-purge-cluster` playbook to remove a Ceph cluster. This playbook purges a Ceph cluster that is managed with cephadm.

For more information, see Purging the Ceph storage cluster.

Registering the IBM Storage Ceph nodes

Register the IBM Storage Ceph nodes on Red Hat Enterprise Linux.

About this task

IBM Storage Ceph 7.1 is supported on Red Hat Enterprise Linux 8.10 and 9.2.

Note: When using Red Hat Enterprise Linux 8.x, the Admin node must be running a supported Red Hat Enterprise Linux 9.x version for your IBM Storage Ceph cluster installation.

Before you begin

Before registering the IBM Storage Ceph nodes, be sure that you have:

- At least one running virtual machine (VM) or bare-metal server with an active internet connection.
- Red Hat Enterprise Linux 8.10 or 9.4 with `ansible-core` bundled into AppStream.
- A valid IBM subscription with the appropriate entitlements.
- Root-level access to all nodes.

For the latest supported Red Hat Enterprise Linux versions, see Compatibility matrix.

Procedure

1. Register the system, and when prompted, enter your Red Hat customer portal credentials.

   Example

   ```
   [root@admin ~]# subscription-manager register
   ```

2. Pull the latest subscription.

   Syntax

   ```
   [root@admin ~]# subscription-manager refresh
   ```

3. Identify the appropriate subscription and retrieve its pool ID.
4. Attach the pool ID to gain access to the software entitlements.

   Use the pool ID you identified in the previous step.

   Example

   ```
   [root@admin ~]# subscription-manager attach --pool=POOL_ID
   ```

5. Disable the software repositories.

   Example

   ```
   [root@admin ~]# subscription-manager repos --disable=*
   ```

6. Enable the Red Hat Enterprise Linux baseos and appstream repositories.

   Example

   ```
   [root@admin ~]# subscription-manager repos --enable=rhel-9-for-x86_64-baseos-rpms
   [root@admin ~]# subscription-manager repos --enable=rhel-9-for-x86_64-appstream-rpms
   ```

7. Update the system.

   Example

   ```
   [root@admin ~]# dnf update
   ```

8. Enable the `ceph-tools` repository.

   Example

   ```
   [root@admin ~]# curl https://public.dhe.ibm.com/ibmdl/export/pub/storage/ceph/beta/ibm-
   storage-ceph-7-beta-rhel-9.repo | sudo tee /etc/yum.repos.d/ibm-storage-ceph-7-rhel-9.repo
   ```

9. Repeat steps "1" on page 12 through "8" on page 13 on all the nodes of the storage cluster.
10. Add license to install IBM Storage Ceph and click **Accept**.

    ```
    [root@admin ~]# dnf install ibm-storage-ceph-license
    ```

11. Accept the provisions.

    ```
    [root@admin ~]# touch /usr/share/ibm-storage-ceph-license/accept
    ```

12. Repeat "10" on page 13 and "11" on page 13 on all the nodes of the storage cluster.
13. Install `cephadm-ansible`.

    ```
    dnf install cephadm-ansible
    ```

Configuring Ansible inventory location

You can configure inventory location files for the `cephadm-ansible` staging and production environments. The Ansible inventory hosts file contains all the hosts that are part of the storage cluster.

You can list nodes individually in the inventory hosts file or you can create groups such as `[mons]`, `[osds]`, and `[rgws]` to provide clarity to your inventory and ease the usage of the `--limit` option to target a group or node when running a playbook.

Note: If deploying clients, client nodes must be defined in a dedicated `[clients]` group.

Prerequisites

- An Ansible administration node.
- Root-level access to the Ansible administration node.
- The `cephadm-ansible` package is installed on the node.

Procedure

1. Navigate to the `/usr/share/cephadm-ansible/` directory:

   ```
   [root@admin ~]# cd /usr/share/cephadm-ansible
   ```

2. Optional: Create subdirectories for staging and production:

   ```
   [root@admin cephadm-ansible]# mkdir -p inventory/staging inventory/production
   ```

3. Optional: Edit the `ansible.cfg` file and add the following line to assign a default inventory location:

   ```
   [defaults]
   inventory = ./inventory/staging
   ```

4. Optional: Create an inventory `hosts` file for each environment:

   ```
   [root@admin cephadm-ansible]# touch inventory/staging/hosts
   [root@admin cephadm-ansible]# touch inventory/production/hosts
   ```

5. Open and edit each `hosts` file and add the nodes and `[admin]` group:

   Syntax

   ```
   NODE_NAME_1
   NODE_NAME_2
   ```

```
[admin]
ADMIN_NODE_NAME_1
```

- Replace *NODE_NAME_1* and *NODE_NAME_2* with the Ceph nodes such as monitors, OSDs, MDSs, and gateway nodes.
- Replace *ADMIN_NODE_NAME_1* with the name of the node where the admin keyring is stored.

  Example

  ```
  host02
  host03
  host04

  [admin]
  host01
  ```

  Note: If you set the inventory location in the `ansible.cfg` file to staging, you need to run the playbooks in the staging environment as follows:

  Syntax

  ```
  ansible-playbook -i inventory/staging/hosts PLAYBOOK.yml
  ```

  To run the playbooks in the production environment:

  Syntax

  ```
  ansible-playbook -i inventory/production/hosts PLAYBOOK.yml
  ```

Enabling SSH login as root user on Red Hat Enterprise Linux 9

Enabling SSH login as root user on Red Hat Enterprise Linux 9, as it is not automatically supported.

Red Hat Enterprise Linux 9 does not support SSH login as a root user even if `PermitRootLogin` parameter is set to `yes` in the `/etc/ssh/sshd_config file`. You get the following error:

Example

```
[root@host01 ~]# ssh root@myhostname
root@myhostname password:
Permission denied, please try again.
```

You can run one of the following methods to enable login as a root user:

- Use the `Allow root SSH login with password` flag while setting the root password during installation of Red Hat Enterprise Linux 9.
- Manually set the `PermitRootLogin` parameter after Red Hat Enterprise Linux 9 installation.

This section describes manual setting of the `PermitRootLogin` parameter.

Prerequisites

- Root-level access to all nodes.

Procedure

1. Open the `etc/ssh/sshd_config` file and set the `PermitRootLogin` to `yes`:

   Example

   ```
   [root@admin ~]# echo 'PermitRootLogin yes' >> /etc/ssh/sshd_config.d/01-permitrootlogin.conf
   ```

2. Restart the SSH service:

   Example

```
[root@admin ~]# systemctl restart sshd.service
```

3. Login to the node as the `root` user:

   Syntax

   ```
   ssh root@HOST_NAME
   ```

   Replace HOST_NAME with the host name of the Ceph node.

   Example

   ```
   [root@admin ~]# ssh root@host01
   ```

   Enter the `root` password when prompted.

Reference

- For more information, see Not able to login as root user via ssh in RHEL 9 server

Creating an Ansible user with sudo access

You can create an Ansible user with password-less `root` access on all nodes in the storage cluster to run the `cephadm-ansible` playbooks.

About this task

The Ansible user must be able to log into all the IBM Storage Ceph nodes as a user that has `root` privileges to install software and create configuration files without prompting for a password.

Important: If you are using Red Hat Enterprise Linux 9 only use these steps if you are a non-root user. If you are a root user, see the "Enabling SSH login as root user on Red Hat Enterprise Linux 9" on page 14 section.

For more information about creating user accounts, see **Configuring basic system settings** > **Getting started with managing user accounts** in the Red Hat Enterprise Linux documentation on the Red Hat Customer Portal.

Before you begin

- Root-level access to all nodes.
- For Red Hat Enterprise 9, to log in as a root user, see "Enabling SSH login as root user on Red Hat Enterprise Linux 9" on page 14.

Procedure

Complete these steps on each node in the storage cluster.

1. Log in to the node as the `root` user.

   ```
   ssh root@HOST_NAME
   ```

   Replace *HOST_NAME* with the host name of the Ceph node.

   For example,

   ```
   [root@admin ~]# ssh root@host01
   ```

   Enter the `root` password when prompted.

2. Create a new Ansible user.

   ```
   adduser USER_NAME
   ```

Replace *USER_NAME* with the new user name for the Ansible user.

Important: Do not use ceph as the user name. The ceph user name is reserved for the Ceph daemons. A uniform user name across the cluster can improve ease of use, but avoid using obvious user names, because intruders typically use them for brute-force attacks.

For example,

```
[root@host01 ~]# adduser ceph-admin
```

3. Set a new password for this user.

   ```
   passwd USER_NAME
   ```

   Replace *USER_NAME* with the new user name for the Ansible user.

   For example,

   ```
   [root@host01 ~]# passwd ceph-admin
   ```

   Enter the new password twice when prompted.
4. Configure sudo access for the newly created user.

   ```
   cat << EOF >/etc/sudoers.d/USER_NAME
   $USER_NAME ALL = (root) NOPASSWD:ALL
   EOF
   ```

   Replace *USER_NAME* with the user name for the Ansible user, created in step "3" on page 16.

   For example,

   ```
   [root@host01 ~]# cat << EOF >/etc/sudoers.d/ceph-admin
   ceph-admin ALL = (root) NOPASSWD:ALL
   EOF
   ```
5. Assign the correct file permissions to the new file.

   ```
   chmod 0440 /etc/sudoers.d/USER_NAME
   ```

   Replace *USER_NAME* with the user name for the Ansible user, created in step "3" on page 16.

   For example,

   ```
   [root@host01 ~]# chmod 0440 /etc/sudoers.d/ceph-admin
   ```

Related information

Managing sudo access

Enabling password-less SSH for Ansible

Generate an SSH key pair on the Ansible administration node and distribute the public key to each node in the storage cluster so that Ansible can access the nodes without being prompted for a password.

Important: If you are using Red Hat Enterprise Linux 9 only use these steps if you are a non-root user. If you are a root user, go to "Bootstrapping a new storage cluster" on page 19.

Prerequisites

- Access to the Ansible administration node.
- Ansible user with sudo access to all nodes in the storage cluster.
- For Red Hat Enterprise 9, to log in as a root user, see "Enabling SSH login as root user on Red Hat Enterprise Linux 9" on page 14

Procedure

1. Generate the SSH key pair, accept the default file name and leave the passphrase empty:

   ```
   [ansible@admin ~]$ ssh-keygen
   ```

2. Copy the public key to all nodes in the storage cluster:

   ```
   ssh-copy-id USER_NAME@HOST_NAME
   ```

   Replace *USER_NAME* with the new user name for the Ansible user. Replace *HOST_NAME* with the host name of the Ceph node.

   Example

   ```
   [ansible@admin ~]$ ssh-copy-id ceph-admin@host01
   ```

3. Create the user's SSH `config` file:

   ```
   [ansible@admin ~]$ touch ~/.ssh/config
   ```

4. Open for editing the `config` file. Set values for the `Hostname` and `User` options for each node in the storage cluster:

   Syntax

   ```
   Host host01
      Hostname HOST_NAME
      User USER_NAME
   Host host02
      Hostname HOST_NAME
      User USER_NAME
   ...
   ```

   Replace *HOST_NAME* with the host name of the Ceph node. Replace *USER_NAME* with the new user name for the Ansible user.

   Example

   ```
   Host host01
      Hostname host01
      User ceph-admin
   Host host02
      Hostname host02
      User ceph-admin
   Host host03
      Hostname host03
      User ceph-admin
   ```

   Important: By configuring the `~/.ssh/config` file you do not have to specify the `-u _USER_NAME_` option each time you execute the `ansible-playbook` command.

5. Set the correct file permissions for the `~/.ssh/config` file:

   ```
   [ansible@admin ~]$ chmod 600 ~/.ssh/config
   ```

Reference

- The `ssh_config(5)` manual page.
- See *Using secure communications between two systems with OpenSSH*.

Running the preflight playbook

This Ansible playbook configures the Ceph repository and prepares the storage cluster for bootstrapping. It also installs some prerequisites, such as podman, `lvm2`, `chronyd`, and `cephadm`. The default location for `cephadm-ansible` and `cephadm-preflight.yml` is `/usr/share/cephadm-ansible`.

About this task

The preflight playbook uses the `cephadm-ansible` inventory file to identify all the admin and nodes in the storage cluster.

The default location for the inventory file is `/usr/share/cephadm-ansible/hosts`.

The following example shows the structure of a typical inventory file:

```
host02
host03
host04

[admin]
host01
```

The `[admin]` group in the inventory file contains the name of the node where the admin keyring is stored. On a new storage cluster, the node in the `[admin]` group will be the bootstrap node. To add additional admin hosts after bootstrapping the cluster see "Setting up the admin node" on page 57.

Important: If you are performing a disconnected installation, see "Running the preflight playbook for a disconnected installation" on page 34.

Note: Run the preflight playbook before you bootstrap the initial host.

Before you begin

- `cephadm-ansible` package installed using the `dnf install cephadm-ansible` command.
- Root-level access to the Ansible administration node.
- Ansible user with sudo and password-less `ssh` access to all nodes in the storage cluster.
- The IBM license installed on all the nodes.

Procedure

Note: In the following procedure, `host01` is the bootstrap node.

1. Navigate to the `/usr/share/cephadm-ansible` directory.
2. Open and edit the `hosts` file and add your nodes.
   For example,

```
host02
host03
host04

[admin]
host01
```

3. Run the preflight playbook.

   Note: When running the preflight playbook, `cephadm-ansible` automatically installs `chronyd` and `ceph-common` on the client nodes.

   Important: Since this is a BETA release, you need to use the **`ceph_origin`** parameter set to `custom` to use the repository.

```
ansible-playbook -i INVENTORY_FILE cephadm-preflight.yml ---extra-vars "ceph_origin=custom"
-e "custom_repo_url=BETA_URL"
```

   For example,

```
[ansible@admin cephadm-ansible]$ ansible-playbook -i hosts cephadm-preflight.yml --extra-vars
"ceph_origin=custom" -e "custom_repo_url=https://public.dhe.ibm.com/ibmdl/export/pub/storage/
ceph/beta/rhel9/x86_64/"
```

   After installation is complete, `cephadm` resides in the `/usr/sbin/` directory.

- To run the preflight playbook on a selected set of hosts in a storage cluster, use the **`--limit`** option.

  ```
  ansible-playbook -i INVENTORY_FILE cephadm-preflight.yml --extra-vars "ceph_origin=custom"
  -e "custom_repo_url=BETA_URL" --limit GROUP_NAME|NODE_NAME
  ```

  Replace *GROUP_NAME* with a group name from your inventory file and *NODE_NAME* with a specific node name from your inventory file.

  For example,

  ```
  [ansible@admin cephadm-ansible]$ ansible-playbook -i hosts cephadm-preflight.yml --extra-vars
  "ceph_origin=custom" -e "custom_repo_url=https://public.dhe.ibm.com/ibmdl/export/pub/storage/
  ceph/beta/rhel9/x86_64/" --limit clients
  [ansible@admin cephadm-ansible]$ ansible-playbook -i hosts cephadm-preflight.yml --extra-vars
  "ceph_origin=custom" -e "custom_repo_url=https://public.dhe.ibm.com/ibmdl/export/pub/storage/
  ceph/beta/rhel9/x86_64/" --limit host01
  ```

When running the preflight playbook, `cephadm-ansible` automatically installs `chronyd` and `ceph-common` on the client nodes.

If you want to configure multiple sources or if you have a disconnected environment, see the following documentation for more information:

How to install chrony?

Best practices for NTP

Basic chrony NTP troubleshooting

Bootstrapping a new storage cluster

Use the `cephadm` utility to bootstrap a new storage cluster.

About this task

The `cephadm` utility performs the following tasks during the bootstrap process:

- Installs and starts a Ceph Monitor daemon and a Ceph Manager daemon for a new IBM Storage Ceph cluster on the local node as containers.
- Creates the `/etc/ceph` directory.
- Writes a copy of the public key to `/etc/ceph/ceph.pub` for the IBM Storage Ceph cluster and adds the SSH key to the root user's `/root/.ssh/authorized_keys` file.
- Applies the `_admin` label to the bootstrap node.
- Writes a minimal configuration file needed to communicate with the new cluster to `/etc/ceph/ceph.conf`.
- Writes a copy of the `client.admin` administrative secret key to `/etc/ceph/ceph.client.admin.keyring`.
- Deploys a basic monitoring stack with Prometheus, Grafana, and other tools such as `node-exporter` and `alert-manager`.

Important:

- If you are performing a disconnected installation, see "Performing a disconnected installation" on page 35.
- If you are deploying a monitoring stack, see Deploying the monitoring stack using the Ceph Orchestrator.
- Bootstrapping provides the default user name and password for the initial login to the Dashboard. Bootstrap requires you to change the password after you log in.
- Before you begin the bootstrapping process, make sure that the container image that you want to use has the same version of IBM Storage Ceph as `cephadm`. If the two versions do not match, bootstrapping fails at the `Creating initial admin user` stage.

Note:

- If you have an existing Prometheus services that you want to run with the new storage cluster, or if you are running Ceph with Rook, use the `--skip-monitoring-stack` option with the `cephadm bootstrap` command. This option bypasses the basic monitoring stack so that you can manually configure it later.
- Before you begin the bootstrapping process, you must create a username and password for the `cp.icr.io/cp` container registry.

Note: You can enable Call Home and Storage Insights during installation. For bootstrap command options for Call Home and IBM Storage Insights, see "Bootstrap command options" on page 25.

Note: By enabling Call Home utilizing these parameters, you agree to allow IBM and its subsidiaries to store and use your contact information and certain system information anywhere you do business worldwide as described in the Program license agreement and documentation. For more information, refer to the Program license agreement and documentation.

For more information, see:

- "Recommended cephadm bootstrap command options" on page 21
- "Using a JSON file to protect login information" on page 21

Before you begin

- An IP address for the first Ceph Monitor container, which is also the IP address for the first node in the storage cluster.
- Login access to `cp.icr.io/cp`. For information about obtaining credentials for `cp.icr.io/cp`, see "Obtaining entitlement key" on page 28.
- A minimum of 10 GB of free space for `/var/lib/containers/`.
- Root-level access to all nodes.

Procedure

Important:

– Run **`cephadm bootstrap`** on the node that you want to be the initial Monitor node in the cluster. The *IP_ADDRESS* option should be the IP address of the node you are using to run **`cephadm bootstrap`**.

Note:

– If the storage cluster includes multiple networks and interfaces, be sure to choose a network that is accessible by any node that uses the storage cluster.

– If the local node uses fully-qualified domain names (FQDN), then add the **`--allow-fqdn-hostname`** option to the **`cephadm bootstrap`** command.

– To deploy a storage cluster using IPV6 addresses, use the IPV6 address format for the `--mon-ip` *`IP_ADDRESS`* option. For example: **`cephadm bootstrap --mon-ip 2620:52:0:880:225:90ff:fefc:2536 --registry-json /etc/mylogin.json`**.

– To route the internal cluster traffic over the public network, omit the `--cluster-network` *`NETWORK_CIDR`* option.

- Bootstrap a storage cluster.

```
cephadm bootstrap --cluster-network NETWORK_CIDR --mon-ip IP_ADDRESS --registry-url
cp.icr.io/cp --registry-username USER_NAME --registry-password PASSWORD --yes-i-know
```

For example,

```
[root@host01 ~]# cephadm bootstrap --cluster-network 10.10.128.0/24 --mon-ip 10.10.128.68 --
registry-url cp.icr.io/cp --registry-username myuser1 --registry-password mypassword1 --yes-
i-know
```

The script takes a few minutes to complete. Once the script completes, it provides the credentials to the IBM Storage Ceph Dashboard URL, a command to access the Ceph command-line interface (CLI), and a request to enable telemetry.

```
Ceph Dashboard is now available at:

             URL: https://host01:8443/
            User: admin
        Password: i8nhu7zham

Enabling client.admin keyring and conf on hosts with "admin" label
You can access the Ceph CLI with:

        sudo /usr/sbin/cephadm shell --fsid 266ee7a8-2a05-11eb-b846-5254002d4916 -c /etc/ceph/
ceph.conf -k /etc/ceph/ceph.client.admin.keyring

Please consider enabling telemetry to help improve Ceph:

        ceph telemetry on

For more information see:

        https://docs.ceph.com/docs/master/mgr/telemetry/

Bootstrap complete.
```

Recommended `cephadm bootstrap` command options

IBM recommends that you use a basic set of command options for `cephadm bootstrap`. You can configure additional options after your initial cluster is up and running.

The **`cephadm bootstrap`** command has multiple options that allow you to specify file locations, configure `ssh` settings, set passwords, and perform other initial configuration tasks.

The following examples show how to specify the recommended options.

Syntax

```
cephadm bootstrap --ssh-user USER_NAME --mon-ip IP_ADDRESS --allow-fqdn-hostname --registry-
json REGISTRY_JSON
```

Example

```
[root@host01 ~]# cephadm bootstrap --ssh-user ceph --mon-ip 10.10.128.68 --allow-fqdn-hostname
--registry-json /etc/mylogin.json
```

For non-root users, see "Creating an Ansible user with sudo access" on page 15 and "Enabling password-less SSH for Ansible" on page 16 for more details.

Reference

- For more information about the `--registry-json` option, see "Using a JSON file to protect login information" on page 21.
- For more information about all available `cephadm bootstrap` options, see "Bootstrap command options" on page 25.
- For more information about bootstrapping the storage cluster as a non-root user, see "Bootstrapping the storage cluster as a non-root user" on page 24.

Using a JSON file to protect login information

As a storage administrator, you might choose to add login and password information to a JSON file, and then refer to the JSON file for bootstrapping. This protects the login credentials from exposure.

Note: A JSON file can also be used with the **`cephadm --registry-login`** command.

Prerequisites

- An IP address for the first Ceph Monitor container, which is also the IP address for the first node in the storage cluster.
- Login access to `cp.icr.io/cp`.
- A minimum of 10 GB of free space for `/var/lib/containers/`.
- Root-level access to all nodes.

Procedure

1. Create the JSON file. In this example, the file is named `mylogin.json`.

   Syntax

   ```
   {
    "url":"REGISTRY_URL",
    "username":"USER_NAME",
    "password":"PASSWORD"
   }
   ```

   Example

   ```
   {
    "url":"cp.icr.io/cp",
    "username":"myuser1",
    "password":"mypassword1"
   }
   ```

2. Bootstrap a storage cluster:

   Syntax

   ```
   cephadm bootstrap --mon-ip IP_ADDRESS --registry-json /etc/mylogin.json
   ```

   Example

   ```
   [root@host01 ~]# cephadm bootstrap --mon-ip 10.10.128.68 --registry-json /etc/mylogin.json
   ```

Bootstrapping a storage cluster using a service configuration file

Use the **`cephadm boostrap`** command with the **`--apply-spec`** option to bootstrap the storage cluster and configure additional hosts and daemons using a service configuration file. The configuration file is a `.yaml` file that contains the service type, placement, and designated nodes for services that you want to deploy.

Note: If you want to use a non-default realm or zone for applications such as multi-site, configure your Ceph Object Gateway daemons after you bootstrap the storage cluster, instead of adding them to the configuration file and using the `--apply-spec` option. This gives you the opportunity to create the realm or zone you need for the Ceph Object Gateway daemons before deploying them.

Note: If deploying a NFS-Ganesha gateway, or Metadata Server (MDS) service, configure them after bootstrapping the storage cluster.

- To deploy a Ceph NFS-Ganesha gateway, you must create a RADOS pool first.
- To deploy the MDS service, you must create a CephFS volume first.

Note: If you run the bootstrap command with `--apply-spec` option, ensure to include the IP address of the bootstrap host in the specification file. This prevents resolving the IP address to loopback address while re-adding the bootstrap host where active Ceph Manager is already running. If you do not use the `--apply spec` option during bootstrap and instead use `ceph orch apply` command with another specification file which includes re-adding the host and contains an active Ceph Manager running, then ensure to explicitly provide the addr field. This is applicable for applying any specification file after bootstrapping.

For more information, see Operations.

Prerequisites

- At least one running virtual machine (VM) or server.
- Root-level access to all nodes.
- Login access to `cp.icr.io/cp`.
- Passwordless `ssh` is set up on all hosts in the storage cluster.
- `cephadm` is installed on the node that you want to be the initial Monitor node in the storage cluster.

For the latest supported Red Hat Enterprise Linux versions, see Compatibility matrix.

Procedure

1. Log in to the bootstrap host.
2. Create the service configuration `.yaml` file for your storage cluster. The example file directs `cephadm bootstrap` to configure the initial host and two additional hosts, and it specifies that OSDs be created on all available disks.

   Example

   ```
   service_type: host
   addr: host01
   hostname: host01
   ---
   service_type: host
   addr: host02
   hostname: host02
   ---
   service_type: host
   addr: host03
   hostname: host03
   ---
   service_type: host
   addr: host04
   hostname: host04
   ---
   service_type: mon
   placement:
     host_pattern: "host[0-2]"
   ---
   service_type: osd
   service_id: my_osds
   placement:
     host_pattern: "host[1-3]"
   data_devices:
     all: true
   ```

3. Bootstrap the storage cluster with the `--apply-spec` option:

   Syntax

   ```
   cephadm bootstrap --apply-spec CONFIGURATION_FILE_NAME --mon-ip MONITOR_IP_ADDRESS --
   registry-url cp.icr.io/cp --registry-username USER_NAME --registry-password PASSWORD
   ```

   Example

   ```
   [root@host01 ~]# cephadm bootstrap --apply-spec initial-config.yaml --mon-ip 10.10.128.68
   --registry-url cp.icr.io/cp --registry-username myuser1 --registry-password mypassword1
   ```

   The script takes a few minutes to complete. Once the script completes, it provides the credentials to the IBM Storage Ceph Dashboard URL, a command to access the Ceph command-line interface (CLI), and a request to enable telemetry.

Once your storage cluster is up and running, see Operations for more information about configuring additional daemons and services.

Reference

- “Bootstrap command options” on page 25

Bootstrapping the storage cluster as a non-root user

You can bootstrap the storage cluster as a non-root user if you have passwordless sudo privileges.

To bootstrap the IBM Storage Ceph cluster as a non-root user on the bootstrap node, use the `--ssh-user` option with the **`cephadm bootstrap`** command. `--ssh-user` specifies a user for SSH connections to cluster nodes.

Non-root users must have passwordless sudo access. For more information, see “Creating an Ansible user with sudo access” on page 15 and “Enabling password-less SSH for Ansible” on page 16.

Prerequisites

- An IP address for the first Ceph Monitor container, which is also the IP address for the initial Monitor node in the storage cluster.
- Login access to `cp.icr.io/cp`.
- A minimum of 10 GB of free space for `/var/lib/containers/`.
- Optional: SSH public and private keys.
- Passwordless sudo access to the bootstrap node.
- Non-root users have passwordless sudo access on all nodes intended to be part of the cluster.

Procedure

1. Change to sudo on the bootstrap node:

   Syntax

   ```
   su - SSH_USER_NAME
   ```

   Example

   ```
   [root@host01 ~]$ su - ceph
   Last login: Tue Sep 14 12:00:29 EST 2021 on pts/0
   ```

2. Check the SSH connection to the bootstrap node:

   Example

   ```
   [ceph@host01 ~]$ ssh host01
   Last login: Tue Sep 14 12:03:29 EST 2021 on pts/0
   ```

3. Optional: Invoke the `cephadm bootstrap` command.

   Note: Using private and public keys is optional.

   If SSH keys have not previously been created, these can be created during this step.

   Include the `--ssh-private-key` and `--ssh-public-key` options:

   Syntax

   ```
   sudo cephadm bootstrap --ssh-user USER_NAME --mon-ip IP_ADDRESS --ssh-private-key
   PRIVATE_KEY --ssh-public-key PUBLIC_KEY --registry-url cp.icr.io/cp --registry-username
   USER_NAME --registry-password PASSWORD
   ```

   Example

   ```
   sudo cephadm bootstrap --ssh-user ceph --mon-ip 10.10.128.68 --ssh-private-key /home/
   ceph/.ssh/id_rsa --ssh-public-key /home/ceph/.ssh/id_rsa.pub --registry-url cp.icr.io/cp --
   registry-username myuser1 --registry-password mypassword1
   ```

Reference

- “Bootstrap command options” on page 25
- For more information about utilizing Ansible to automate bootstrapping a rootless cluster, see the knowledge base article *Red Hat Ceph Storage 5.3 rootless deployment utilizing ansible ad-hoc commands*.
- For more information about sudo privileges, see Managing sudo access

Bootstrap command options

The **`cephadm bootstrap`** command bootstraps a IBM Storage Ceph cluster on the local host. It deploys a MON daemon and a MGR daemon on the bootstrap node, automatically deploys the monitoring stack on the local host, and calls **`ceph orch host add HOSTNAME`**.

The following table lists the available options for **`cephadm bootstrap`**.

`cephadm bootstrap` **option**	**Description**
--config *CONFIG_FILE*, -c *CONFIG_FILE*	*CONFIG_FILE* is the `ceph.conf` file to use with the bootstrap command
--cluster-network *NETWORK_CIDR*	Use the subnet defined by *NETWORK_CIDR* for internal cluster traffic. This is specified in CIDR notation. For example: `10.10.128.0/24`.
--mon-id *MON_ID*	Bootstraps on the host named *MON_ID*. Default value is the local host.
--mon-addrv *MON_ADDRV*	mon IPs (e.g., [v2:localipaddr:3300,v1:localipaddr:6789])
--mon-ip *IP_ADDRESS*	IP address of the node you are using to run `cephadm bootstrap`.
--mgr-id *MGR_ID*	Host ID where a MGR node should be installed. Default: randomly generated.
--fsid *FSID*	Cluster FSID.
--output-dir *OUTPUT_DIR*	Use this directory to write config, keyring, and pub key files.
--output-keyring *OUTPUT_KEYRING*	Use this location to write the keyring file with the new cluster admin and mon keys.
--output-config *OUTPUT_CONFIG*	Use this location to write the configuration file to connect to the new cluster.
--output-pub-ssh-key *OUTPUT_PUB_SSH_KEY*	Use this location to write the public SSH key for the cluster.
--skip-ssh	Skip the setup of the ssh key on the local host.
--initial-dashboard-user *INITIAL_DASHBOARD_USER*	Initial user for the dashboard.

`cephadm bootstrap` option	Description
--initial-dashboard-password *INITIAL_DASHBOARD_PASSWORD*	Initial password for the initial dashboard user.
--ssl-dashboard-port *SSL_DASHBOARD_PORT*	Port number used to connect with the dashboard using SSL.
--dashboard-key *DASHBOARD_KEY*	Dashboard key.
--dashboard-crt *DASHBOARD_CRT*	Dashboard certificate.
--ssh-config *SSH_CONFIG*	SSH config.
--ssh-private-key *SSH_PRIVATE_KEY*	SSH private key.
--ssh-public-key *SSH_PUBLIC_KEY*	SSH public key.
--ssh-user *SSH_USER*	Sets the user for SSH connections to cluster hosts. Passwordless sudo is needed for non-root users.
--skip-mon-network	Sets mon public_network based on the bootstrap mon ip.
--skip-dashboard	Do not enable the Ceph Dashboard.
--dashboard-password-noupdate	Disable forced dashboard password change.
--no-minimize-config	Do not assimilate and minimize the configuration file.
--skip-ping-check	Do not verify that the mon IP is pingable.
--skip-pull	Do not pull the latest image before bootstrapping.
--skip-firewalld	Do not configure firewalld.
--allow-overwrite	Allow the overwrite of existing –output-* config/keyring/ssh files.
--allow-fqdn-hostname	Allow fully qualified host name.
--skip-prepare-host	Do not prepare host.
--orphan-initial-daemons	Do not create initial mon, mgr, and crash service specs.
--skip-monitoring-stack	Do not automatically provision the monitoring stack] (prometheus, grafana, alertmanager, node-exporter).
--apply-spec *APPLY_SPEC*	Apply cluster spec file after bootstrap (copy ssh key, add hosts and apply services).

`cephadm bootstrap` option	Description
--registry-url *REGISTRY_URL*	Specifies the URL of the custom registry to log into. For example: `cp.icr.io/cp`.
--registry-username *REGISTRY_USERNAME*	User name of the login account to the custom registry.
--registry-password *REGISTRY_PASSWORD*	Password of the login account to the custom registry.
--registry-json *REGISTRY_JSON*	JSON file containing registry login information.

The following table lists the available options that are required to enable Call home via the bootstrap command -

`cephadm bootstrap` option	Description
--enable-ibm-call-home	Flag to opt-in to IBM Call Home. Syntax: `cephadm -v bootstrap --mon-ip 10.0.210.25 --registry-json registry.json --enable-ibm-call-home --call-home-config call_home.json`
`--call-home-icn`	IBM Customer Number (7 alphanumeric characters long).
`--ceph-call-home-contact-first-name`	Storage help desk contact first name.
`--ceph-call-home-contact-last-name`	Storage help desk contact last name.
`--ceph-call-home-contact-email`	Storage help desk email. For example, *storage_helpdesk@ibm.com*.
`--ceph-call-home-contact-phone`	Storage help desk phone number.
`--ceph-call-home-country-code`	2-letter country code.
`--call-home-config`	File path to a JSON config file which includes the following keys: `{ "icn": "<IBM_CUSTOMER_NUMBER>", "email": "<CALL_HOME_CONTACT_EMAIL_ADDRESS>", "phone": "<CALL_HOME_CONTACT_PHONE_NUMBER>", "first_name": "<CALL_HOME_CONTACT_FIRST_NAME>", "last_name": "<CALL_HOME_CONTACT_LAST_NAME>", "country_code": "<CUSTOMER_COUNTRY_CODE>" }` Use either the direct parameters with their options, or the `--call-home-config` flag with the file path.

Enable IBM Storage Insights too by providing the following flags and parameters:

`cephadm bootstrap` option	Description
`--enable-storage-insights`	Flag to opt-in to IBM Storage Insights. Syntax: `cephadm -v bootstrap --mon-ip 10.0.210.25 --registry-json registry.json --enable-ibm-call-home --call-home-config call_home.json --enable-storage-insights --storage-insights-config si_opt.json`
`--storage-insights-tenant-id`	A Storage Insights tenant ID to be associated with this deployment.
`--storage-insights-config`	File path to a JSON config file which includes the following key: `{"tenant_id": "<STORAGE_INSIGHTS_TENANT_ID>"}` Use either the direct parameters with their options, or the `--storage-insights-config` flag and specify its file path.

Reference

- For more information about the `--skip-monitoring-stack` option, see "Adding hosts" on page 44.
- For more information about logging into the registry with the `registry-json` option, see help for the `registry-login` command.
- For more information about `cephadm` options, see help for `cephadm`.

Obtaining entitlement key

Entitlement keys determine whether IBM Storage Ceph can automatically pull the required container default images. During installation, image pull failures can occur due to an invalid entitlement key or a key belonging to an account that does not have entitlement to IBM Storage Ceph.

Procedure

1. Log in to the IBM container software library with the IBMid and password that is associated with the entitled IBM Storage Ceph software.
2. In the navigation bar, click **Get entitlement key**.
3. On the **Access your container software** page, click **Copy key** to copy the generated entitlement key.
4. Save the key to a secure location for future use.

Distributing SSH keys

You can use the `cephadm-distribute-ssh-key.yml` playbook to distribute the SSH keys instead of creating and distributing the keys manually.

Before you begin

- Ansible is installed on the administration node.
- Access to the Ansible administration node.
- Ansible user with sudo access to all nodes in the storage cluster.
- Bootstrapping is completed. See "Bootstrapping a new storage cluster" on page 19 for more details.

About this task

The playbook distributes an SSH public key over all hosts in the inventory. You can also generate an SSH key pair on the Ansible administration node and distribute the public key to each node in the storage cluster so that Ansible can access the nodes without being prompted for a password.

Procedure

1. Navigate to the `/usr/share/cephadm-ansible` directory on the Ansible administration node.

```
[ansible@admin ~]$ cd /usr/share/cephadm-ansible
```

2. From the Ansible administration node, distribute the SSH keys. The optional `cephadm_pubkey_path` parameter is the full path name of the SSH public key file on the ansible controller host.

   Note:

   If `cephadm_pubkey_path` is not specified, the playbook gets the key from the `cephadm get-pub-key` command. This implies that you have at least bootstrapped a minimal cluster.

```
ansible-playbook -i INVENTORY_HOST_FILE cephadm-distribute-ssh-key.yml -e
cephadm_ssh_user=USER_NAME -e cephadm_pubkey_path= home/cephadm/ceph.key -e
admin_node=ADMIN_NODE_NAME_1
```

```
[ansible@admin cephadm-ansible]$ ansible-playbook -i hosts cephadm-distribute-ssh-key.yml -e
cephadm_ssh_user=ceph-admin -e cephadm_pubkey_path=/home/cephadm/ceph.key -e admin_node=host01

[ansible@admin cephadm-ansible]$ ansible-playbook -i hosts cephadm-distribute-ssh-key.yml -e
cephadm_ssh_user=ceph-admin -e admin_node=host01
```

Disconnected installation

As a storage administrator, you can use a disconnected installation procedure to install `cephadm` and bootstrap your storage cluster on a private network. A disconnected installation uses a private registry for installation.

Configuring a private registry for a disconnected installation

Use this procedure when the IBM Storage Ceph nodes do NOT have access to the Internet during deployment.

Before you begin

- At least one running virtual machine (VM) or server with an active internet connection.
- Red Hat Enterprise Linux 8.10 or 9.4 with `ansible-core` bundled into AppStream.
- Login access to `cp.icr.io/cp`.
- Root-level access to all nodes.

For the latest supported Red Hat Enterprise Linux versions, see Compatibility matrix.

About this task

Follow this procedure to set up a secure private registry using authentication and a self-signed certificate. Perform these steps on a node that has both Internet access and access to the local cluster.

Note: Do not use an insecure registry for production.

Procedure

1. Register the hosts, and when prompted, enter the appropriate Red Hat customer portal credentials:

Example

```
[root@admin ~]# subscription-manager register
```

2. Pull the latest subscription:

   Example

   ```
   subscription-manager refresh
   ```

3. Identify the appropriate subscription and retrieve its Pool ID.
4. Attach the pool ID to gain access to the software entitlements. Use the Pool ID you identified in the previous step.

   Example

   ```
   [root@admin ~]# subscription-manager attach --pool=POOL_ID
   ```

5. Disable the software repositories:

   Example

   ```
   [root@admin ~]# subscription-manager repos --disable=*
   ```

6. Enable the Red Hat Enterprise Linux baseos and appstream repositories.

   Example

   ```
   [root@admin ~]# subscription-manager repos --enable=rhel-9-for-x86_64-baseos-rpms

   [root@admin ~]# subscription-manager repos --enable=rhel-9-for-x86_64-appstream-rpms
   ```

7. Update the system:

   Example

   ```
   [root@admin ~]# dnf update
   ```

8. Enable the `ceph-tools` repository.

   Example

   ```
   [root@admin ~]# curl https://public.dhe.ibm.com/ibmdl/export/pub/storage/ceph/beta/ibm-
   storage-ceph-7-beta-rhel-9.repo | sudo tee /etc/yum.repos.d/ibm-storage-ceph-7-rhel-9.repo
   ```

9. Install the podman and `httpd-tools` packages.

   ```
   [root@admin ~]# dnf install -y podman httpd-tools
   ```

10. Create folders for the private registry.

    ```
    [root@admin ~]# mkdir -p /opt/registry/{auth,certs,data}
    ```

    The registry is stored in `/opt/registry` and the directories are mounted in the container running the registry.

    - The `auth` directory stores the `htpasswd` file the registry uses for authentication.
    - The `certs` directory stores the certificates the registry uses for authentication.
    - The `data` directory stores the registry images.

11. Create credentials for accessing the private registry.

    ```
    htpasswd -bBc /opt/registry/auth/htpasswd PRIVATE_REGISTRY_USERNAME
    PRIVATE_REGISTRY_PASSWORD
    ```

    - The b option provides the password from the command line.
    - The B option stores the password using `Bcrypt` encryption.

- The c option creates the `htpasswd` file.
- Replace *PRIVATE_REGISTRY_USERNAME* with the username to create for the private registry.
- Replace *PRIVATE_REGISTRY_PASSWORD* with the password to create for the private registry username.

For example:

```
[root@admin ~]# htpasswd -bBc /opt/registry/auth/htpasswd myregistryusername
myregistrypassword1
```

12. Create a self-signed certificate.

```
openssl req -newkey rsa:4096 -nodes -sha256 -keyout /opt/registry/certs/domain.key -x509
-days 365 -out /opt/registry/certs/domain.crt -addext "subjectAltName = DNS:LOCAL_NODE_FQDN"
```

Replace *LOCAL_NODE_FQDN* with the fully qualified host name of the private registry node.

There is a prompt for the respective options for your certificate. The CN= value is the host name of your node and should be resolvable by DNS or the `/etc/hosts` file.

For example:

```
[root@admin ~] # openssl req -newkey rsa:4096 -nodes -sha256 -keyout /opt/registry/certs/
domain.key -x509 -days 365 -out /opt/registry/certs/domain.crt -addext "subjectAltName =
DNS:admin.lab.ibm.com"
```

Note: When creating a self-signed certificate, be sure to create a certificate with a proper Subject Alternative Name (SAN). Podman commands that require TLS verification for certificates that do not include a proper SAN, return the following error:

```
x509: certificate relies on legacy Common Name field, use SANs or temporarily enable Common
Name matching with GODEBUG=x509ignoreCN=0
```

13. Create a symbolic link to `domain.cert`.

This allows skopeo to locate the certificate with the file extension `.cert`.

For example:

```
[root@admin ~]# ln -s /opt/registry/certs/domain.crt /opt/registry/certs/domain.cert
```

14. Add the certificate to the trusted list on the private registry node.

```
cp /opt/registry/certs/domain.crt /etc/pki/ca-trust/source/anchors/
 update-ca-trust
 trust list | grep -i "LOCAL_NODE_FQDN"
```

Replace *LOCAL_NODE_FQDN* with the FQDN of the private registry node.

For example:

```
[root@admin ~]# cp /opt/registry/certs/domain.crt /etc/pki/ca-trust/source/anchors/
 [root@admin ~]# update-ca-trust
 [root@admin ~]# trust list | grep -i "admin.lab.ibm.com"

     label: admin.lab.ibm.com
```

15. Copy the certificate to any nodes that will access the private registry for installation and update the trusted list.

For example:

```
[root@admin ~]# scp /opt/registry/certs/domain.crt root@host01:/etc/pki/ca-trust/source/
anchors/
 [root@admin ~]# ssh root@host01
 [root@host01 ~]# update-ca-trust
 [root@host01 ~]# trust list | grep -i "admin.lab.ibm.com"

     label: admin.lab.ibm.com
```

16. Start the local secure private registry.

```
[root@admin ~]# podman run --restart=always --name NAME_OF_CONTAINER \
-p 5000:5000 -v /opt/registry/data:/var/lib/registry:z \
-v /opt/registry/auth:/auth:z \
-v /opt/registry/certs:/certs:z \
-e "REGISTRY_AUTH=htpasswd" \
-e "REGISTRY_AUTH_HTPASSWD_REALM=Registry Realm" \
-e REGISTRY_AUTH_HTPASSWD_PATH=/auth/htpasswd \
-e "REGISTRY_HTTP_TLS_CERTIFICATE=/certs/domain.crt" \
-e "REGISTRY_HTTP_TLS_KEY=/certs/domain.key" \
-e REGISTRY_COMPATIBILITY_SCHEMA1_ENABLED=true \
-d registry:2
```

Replace *NAME_OF_CONTAINER* with a name to assign to the container.

This starts the private registry on port 5000 and mounts the volumes of the registry directories in the container running the registry.

For example:

```
[root@admin ~]# podman run --restart=always --name myprivateregistry \
-p 5000:5000 -v /opt/registry/data:/var/lib/registry:z \
-v /opt/registry/auth:/auth:z \
-v /opt/registry/certs:/certs:z \
-e "REGISTRY_AUTH=htpasswd" \
-e "REGISTRY_AUTH_HTPASSWD_REALM=Registry Realm" \
-e REGISTRY_AUTH_HTPASSWD_PATH=/auth/htpasswd \
-e "REGISTRY_HTTP_TLS_CERTIFICATE=/certs/domain.crt" \
-e "REGISTRY_HTTP_TLS_KEY=/certs/domain.key" \
-e REGISTRY_COMPATIBILITY_SCHEMA1_ENABLED=true \
-d registry:2
```

17. On the local registry node, verify that `cp.icr.io/cp` is in the container registry search path.
    a) Open `/etc/containers/registries.conf` for editing.
    b) Optional: If needed, add `cp.icr.io/cp` to the `unqualified-search-registries` list.

    ```
    unqualified-search-registries = ["cp.icr.io/cp"]
    ```

18. With your IBM Customer Portal credentials, login to `cp.icr.io/cp`.

    ```
    podman login cp.icr.io/cp
    ```

19. You need to have access to `registry.redhat.io` to pull custom images. Log into the Red Hat registry.

    ```
    podman login registry.redhat.io
    ```

20. Copy the following images from the IBM Customer Portal to the private registry.

Table 1. Custom image details for monitoring stack

Monitoring stack component	**Image details**
Ceph image	cp.icr.io/cp/ibm-ceph/ceph-7-rhel9:latest
Prometheus	cp.icr.io/cp/ibm-ceph/prometheus:v4.12
Grafana	cp.icr.io/cp/ibm-ceph/grafana-rhel9:9.4.7-2
Node-exporter	cp.icr.io/cp/ibm-ceph/prometheus-node-exporter:v4.12
AlertManager	cp.icr.io/cp/ibm-ceph/prometheus-alertmanager:v4.12
HAProxy	cp.icr.io/cp/ibm-ceph/haproxy-rhel9:latest

Table 1. Custom image details for monitoring stack (continued)

Monitoring stack component	**Image details**
Keepalived	cp.icr.io/cp/ibm-ceph/keepalived-rhel9:latest
SNMP Gateway	cp.icr.io/cp/ibm-ceph/snmp-notifier-rhel9:latest

```
podman run -v /CERTIFICATE_DIRECTORY_PATH:/certs:Z -v /CERTIFICATE_DIRECTORY_PATH/
domain.cert:/certs/domain.cert:Z --rm registry.redhat.io/rhel9/skopeo skopeo copy --
remove-signatures --src-creds IBM_CUSTOMER_PORTAL_LOGIN:IBM_CUSTOMER_PORTAL_PASSWORD --dest-
cert-dir=./certs/ --dest-creds PRIVATE_REGISTRY_USERNAME:PRIVATE_REGISTRY_PASSWORD docker://
cp.icr.io/cp/SRC_IMAGE:SRC_TAG docker://LOCAL_NODE_FQDN:5000/DST_IMAGE:pDST_TAG
```

- Replace *CERTIFICATE_DIRECTORY_PATH* with the directory path to the self-signed certificates.
- Replace *CERTIFICATE_DIRECTORY_PATH* and *IBM_CUSTOMER_PORTAL_PASSWORD* with your IBM Customer Portal credentials.
- Replace *PRIVATE_REGISTRY_USERNAME* and *PRIVATE_REGISTRY_PASSWORD* with the private registry credentials.
- Replace *SRC_IMAGE* and *SRC_TAG* with the name and tag of the image to copy from cp.icr.io/cp.
- Replace *DST_IMAGE* and *DST_TAG* with the name and tag of the image to copy to the private registry.
- Replace *LOCAL_NODE_FQDN* with the FQDN of the private registry.

For example:

```
[root@admin ~]# podman run -v /opt/registry/certs:/certs:Z -v /opt/registry/certs/
domain.cert:/certs/domain.cert:Z --rm registry.redhat.io/rhel9/skopeo skopeo copy --remove-
signatures --src-creds myusername:mypassword1 --dest-cert-dir=./certs/ --dest-creds
myregistryusername:myregistrypassword1 docker://cp.icr.io/cp/ibm-ceph/ceph-7-rhel9:latest
docker://admin.lab.ibm.com:5000/ibm-ceph/ceph-7-rhel9:latest

[root@admin ~]# podman run -v /opt/registry/certs:/certs:Z -v /opt/registry/certs/
domain.cert:/certs/domain.cert:Z --rm registry.redhat.io/rhel9/skopeo skopeo copy
--remove-signatures --src-creds myusername:mypassword1 --dest-cert-dir=./certs/
--dest-creds myregistryusername:myregistrypassword1 docker://cp.icr.io/cp/ibm-ceph/ose-
prometheus-node-exporter:v4.12 docker://admin.lab.ibm.com:5000/ibm-ceph/ose-prometheus-node-
exporter:v4.12

[root@admin ~]# podman run -v /opt/registry/certs:/certs:Z -v /opt/registry/certs/
domain.cert:/certs/domain.cert:Z --rm registry.redhat.io/rhel9/skopeo skopeo copy --remove-
signatures --src-creds myusername:mypassword1 --dest-cert-dir=./certs/ --dest-creds
myregistryusername:myregistrypassword1 docker://cp.icr.io/cp/ibm-ceph/grafana-rhel9:9.4.7-2
docker://admin.lab.ibm.com:5000/ibm-ceph/grafana-rhel9:9.4.7-2

[root@admin ~]# podman run -v /opt/registry/certs:/certs:Z -v /opt/registry/certs/
domain.cert:/certs/domain.cert:Z --rm registry.redhat.io/rhel9/skopeo skopeo copy --remove-
signatures --src-creds myusername:mypassword1 --dest-cert-dir=./certs/ --dest-creds
myregistryusername:myregistrypassword1 docker://cp.icr.io/cp/ibm-ceph/ose-prometheus:v4.12
docker://admin.lab.ibm.com:5000/ibm-ceph/ose-prometheus:v4.12

[root@admin ~]# podman run -v /opt/registry/certs:/certs:Z -v /opt/registry/certs/
domain.cert:/certs/domain.cert:Z --rm registry.redhat.io/rhel9/skopeo skopeo copy --
remove-signatures --src-creds myusername:mypassword1 --dest-cert-dir=./certs/ --dest-
creds myregistryusername:myregistrypassword1 docker://cp.icr.io/cp/ibm-ceph/ose-prometheus-
alertmanager:v4.12 docker://admin.lab.ibm.com:5000/ibm-ceph/ose-prometheus-alertmanager:v4.12
```

Note: For more image Ceph package versions, see What are the Red Hat Ceph Storage releases and corresponding Ceph package versions? within the Red Hat Customer Portal..

21. Verify that the images reside in the local registry.

    Use the **curl** command.

```
curl -u PRIVATE_REGISTRY_USERNAME:PRIVATE_REGISTRY_PASSWORD https://LOCAL_NODE_FQDN:5000/v2/
catalog
```

For example:

```
[root@admin ~]# curl -u myregistryusername:myregistrypassword1 https://
admin.lab.ibm.com:5000/v2/_catalog

{"repositories":["ibm-ceph/prometheus","ibm-ceph/prometheus-alertmanager","ibm-ceph/
prometheus-node-exporter","ibm-ceph/ceph-7-dashboard-rhel9","ibm-ceph/ceph-7-rhel9"]}
```

Running the preflight playbook for a disconnected installation

Use the `cephadm-preflight.yml` Ansible playbook to configure the Ceph repository and prepare the storage cluster for bootstrapping. It also installs some prerequisites, such as `podman`, `lvm2`, `chronyd`, and `cephadm`.

About this task

The preflight playbook uses the `cephadm-ansible` inventory `hosts` file to identify all the nodes in the storage cluster. The default location for `cephadm-ansible`, `cephadm-preflight.yml`, and the inventory `hosts` file is `/usr/share/cephadm-ansible/`.

The following example shows the structure of a typical inventory file:

```
host02
host03
host04

[admin]
host01
```

The `[admin]` group in the inventory file contains the name of the node where the admin keyring is stored.

Note: Run the preflight playbook before you bootstrap the initial host.

Before you begin

- Nodes configured to access a local YUM repository server with the following repositories enabled on respective Red Hat Enterprise Linux versions.

  -
    ```
    rhel-9-for-x86_64-baseos-rpms
    ```

  -
    ```
    rhel-9-for-x86_64-appstream-rpms
    ```

  -
    ```
    curl https://public.dhe.ibm.com/ibmdl/export/pub/storage/ceph/beta/ibm-storage-ceph-7-beta-
    rhel-9.repo | sudo tee /etc/yum.repos.d/ibm-storage-ceph-7-rhel-9.repo
    ```

  Note:

  For more information about setting up a local YUM repository, see the Red Hat knowledge base article Creating a Local Repository and Sharing with Disconnected/Offline/Air-gapped Systems
- The `cephadm-ansible` package is installed on the Ansible administration node.

  ```
  [root@admin ~]# dnf install cephadm-ansible
  ```
- Root-level access to all nodes in the storage cluster.
- Passwordless `ssh` is set up on all hosts in the storage cluster.

Procedure

1. Navigate to the `/usr/share/cephadm-ansible` directory on the Ansible administration node.
2. Open and edit the `hosts` file and add your nodes.
3. Add license to install IBM Storage Ceph and click **Accept** on all nodes.

   ```
   dnf install ibm-storage-ceph-license
   ```

For example,

```
[root@admin ~]# dnf install ibm-storage-ceph-license
```

4. Accept the provisions.

```
sudo touch /usr/share/ibm-storage-ceph-license/accept
```

For example,

```
[root@admin ~]# sudo touch /usr/share/ibm-storage-ceph-license/accept
```

5. Run the preflight playbook.

Note: When running the preflight playbook, `cephadm-ansible` automatically installs `chronyd` and `ceph-common` on the client nodes.

Use **`ceph_origin`** parameter set to `custom` to use a local YUM repository.

```
ansible-playbook -i INVENTORY_FILE cephadm-preflight.yml --extra-vars "ceph_origin=custom"
-e "custom_repo_url=CUSTOM_REPO_URL"
```

For example,

```
[ansible@admin cephadm-ansible]$ ansible-playbook -i hosts cephadm-preflight.yml --extra-vars
"ceph_origin=custom" -e "custom_repo_url=http://mycustomrepo.lab.ibm.com/x86_64/os/"
```

After installation is complete, `cephadm` resides in the `/usr/sbin/` directory.

- Use the **`--limit`** option to run the preflight playbook on a selected set of hosts in the storage cluster.

```
ansible-playbook -i INVENTORY_FILE cephadm-preflight.yml --extra-vars "ceph_origin=custom"
-e "custom_repo_url=CUSTOM_REPO_URL" --limit GROUP_NAME|NODE_NAME
```

Replace *GROUP_NAME* with a group name from your inventory file and *NODE_NAME* with a specific node name from your inventory file.

For example,

```
[ansible@admin cephadm-ansible]$ ansible-playbook -i hosts cephadm-preflight.yml --extra-vars
"ceph_origin=custom" -e "custom_repo_url=http://mycustomrepo.lab.ibm.com/x86_64/os/" --limit
clients
[ansible@admin cephadm-ansible]$ ansible-playbook -i hosts cephadm-preflight.yml --extra-vars
"ceph_origin=custom" -e "custom_repo_url=http://mycustomrepo.lab.ibm.com/x86_64/os/" --limit
host02
```

Performing a disconnected installation

Before you can perform the installation, you must obtain an IBM Storage Ceph container image, either from a proxy host that has access to the IBM registry or by copying the image to your local registry.

Important: Before you begin the bootstrapping process, make sure that the container image that you want to use has the same version of IBM Storage Ceph as `cephadm`. If the two versions do not match, bootstrapping fails at the `Creating initial admin user` stage.

Note: If your local registry uses a self-signed certificate with a local registry, ensure you have added the trusted root certificate to the bootstrap host. For more information, see "Configuring a private registry for a disconnected installation" on page 29.

Prerequisites

- At least one running virtual machine (VM) or server.
- Root-level access to all nodes.

- Passwordless ssh is set up on all hosts in the storage cluster.
- The preflight playbook has been run on the bootstrap host in the storage cluster. For more information, see "Running the preflight playbook for a disconnected installation" on page 34.
- A private registry has been configured and the bootstrap node has access to it. For more information, see "Configuring a private registry for a disconnected installation" on page 29
- An IBM Storage Ceph container image resides in the custom registry.

Procedure

1. Log in to the bootstrap host.
2. Bootstrap the storage cluster:

   Syntax

   ```
   cephadm --image PRIVATE_REGISTRY_NODE_FQDN:5000/CUSTOM_IMAGE_NAME:IMAGE_TAG bootstrap --
   mon-ip IP_ADDRESS --registry-url PRIVATE_REGISTRY_NODE_FQDN:5000 --registry-username
   PRIVATE_REGISTRY_USERNAME --registry-password PRIVATE_REGISTRY_PASSWORD
   ```

   - Replace *PRIVATE_REGISTRY_NODE_FQDN* with the fully qualified domain name of your private registry.
   - Replace *CUSTOM_IMAGE_NAME* and *IMAGE_TAG* with the name and tag of the IBM Storage Ceph container image that resides in the private registry.
   - Replace *IP_ADDRESS* with the IP address of the node you are using to run `cephadm bootstrap`.
   - Replace *PRIVATE_REGISTRY_USERNAME* with the username to create for the private registry.
   - Replace *PRIVATE_REGISTRY_PASSWORD* with the password to create for the private registry username.

     Example

     ```
     [root@host01 ~]# cephadm --image admin.lab.ibm.com:5000/ibm-ceph/ceph-7-rhel9:latest
     bootstrap --mon-ip 10.10.128.68 --registry-url admin.lab.ibm.com:5000 --registry-username
     myregistryusername --registry-password myregistrypassword1
     ```

     The script takes a few minutes to complete. Once the script completes, it provides the credentials to the IBM Storage Ceph Dashboard URL, a command to access the Ceph command-line interface (CLI), and a request to enable telemetry.

   ```
   Ceph Dashboard is now available at:

                URL: https://host01:8443/
               User: admin
           Password: i8nhu7zham

   Enabling client.admin keyring and conf on hosts with "admin" label
   You can access the Ceph CLI with:

           sudo /usr/sbin/cephadm shell --fsid 266ee7a8-2a05-11eb-b846-5254002d4916 -c /etc/ceph/
   ceph.conf -k /etc/ceph/ceph.client.admin.keyring

   Please consider enabling telemetry to help improve Ceph:

           ceph telemetry on

   For more information see:

           https://docs.ceph.com/docs/master/mgr/telemetry/

   Bootstrap complete.
   ```

After the bootstrap process is complete, configure the container images, as detailed in "Changing configurations of custom container images for disconnected installations" on page 37.

Once your storage cluster is up and running, configure additional daemons and services. For more information, see Operations

Changing configurations of custom container images for disconnected installations

After you perform the initial bootstrap for disconnected nodes, you must specify custom container images for monitoring stack daemons. You can override the default container images for monitoring stack daemons, since the nodes do not have access to the default container registry.

Note: Make sure that the bootstrap process on the initial host is complete before making any configuration changes.

By default, the monitoring stack components are deployed based on the primary Ceph image. For disconnected environment of the storage cluster, you can use the latest available monitoring stack component images.

Note: When using a custom registry, be sure to log in to the custom registry on newly added nodes before adding any Ceph daemons.

Syntax

```
ceph cephadm registry-login --registry-url CUSTOM_REGISTRY_NAME --registry_username
REGISTRY_USERNAME --registry_password REGISTRY_PASSWORD
```

Example

```
# ceph cephadm registry-login --registry-url myregistry --registry_username myregistryusername --
registry_password myregistrypassword1
```

For more information, see "Performing a disconnected installation" on page 35.

Prerequisites

- At least one running virtual machine (VM) or server.
- Red Hat Enterprise Linux 8.10 or 9.4 with `ansible-core` bundled into AppStream.
- Root-level access to all nodes.
- Passwordless `ssh` is set up on all hosts in the storage cluster.

For the latest supported Red Hat Enterprise Linux versions, see Compatibility matrix.

Procedure

1. Set the custom container images with the `ceph config` command:

   Syntax

   ```
   ceph config set mgr mgr/cephadm/OPTION_NAME CUSTOM_REGISTRY_NAME/IMAGE_NAME
   ```

   Use the following options for *OPTION_NAME*:

   ```
   container_image_prometheus
   container_image_grafana
   container_image_alertmanager
   container_image_node_exporter
   ```

   Example

   ```
   [root@host01 ~]# ceph config set mgr mgr/cephadm/container_image_prometheus private_registry/
   prometheus
   [root@host01 ~]# ceph config set mgr mgr/cephadm/container_image_grafana private_registry/
   grafana
   [root@host01 ~]# ceph config set mgr mgr/cephadm/container_image_alertmanager
   private_registry/alertmanager
   [root@host01 ~]# ceph config set mgr mgr/cephadm/container_image_node_exporter
   private_registry/node_exporter
   ```

2. Redeploy `node-exporter`:

Syntax

```
ceph orch redeploy node-exporter
```

Note:

- If any of the services do not deploy, you can redeploy them with the **ceph orch redeploy** command.
- By setting a custom image, the default values for the configuration image name and tag will be overridden, but not overwritten. The default values change when updates become available. By setting a custom image, you will not be able to configure the component for which you have set the custom image for automatic updates. You will need to manually update the configuration image name and tag to be able to install updates.
- If you choose to revert to using the default configuration, you can reset the custom container image. Use **ceph config rm** to reset the configuration option:

  Syntax

  ```
  ceph config rm mgr mgr/cephadm/OPTION_NAME
  ```

  Example

  ```
  ceph config rm mgr mgr/cephadm/container_image_prometheus
  ```

Reference

- "Performing a disconnected installation" on page 35

Adding hosts in disconnected deployments

If you are running a storage cluster on a private network and your host domain name server (DNS) cannot be reached through private IP, you must include both the host name and the IP address for each host you want to add to the storage cluster.

Before you begin

- A running IBM Storage Ceph cluster.
- Root-level access to all hosts in the storage cluster.

Procedure

1. Log into the cephadm shell.

   Syntax

   ```
   [root@host01 ~]# cephadm shell
   ```

2. Add the host.

   Syntax

   ```
   ceph orch host add HOST_NAME HOST_ADDRESS
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph orch host add host03 10.10.128.70
   ```

Starting the cephadm shell

The **cephadm shell** command opens a bash shell in a container with all Ceph packages installed. Use the shell to run "Day One" cluster setup tasks, such as installation and bootstrapping, and to run ceph commands.

About this task

Note: If the node contains configuration and keyring files in /etc/ceph/, the container environment uses the values in those files as defaults for the cephadm shell. If you run the cephadm shell on a MON node, the cephadm shell inherits its default configuration from the MON container, instead of using the default configuration.

Before you begin

- A running IBM Storage Ceph cluster.
- Root-level access to all nodes in the storage cluster.

Procedure

Open the cephadm shell in one of the following ways.

- Enter **cephadm shell** at the system prompt.

  This example runs the **ceph -s** command from within the shell.

  For example,

```
[root@host01 ~]# cephadm shell
[ceph: root@host01 /]# ceph -s
```

- At the system prompt, type cephadm shell and the command you want to run.
  For example,

```
[root@host01 ~]# cephadm shell ceph -s
cluster:
    id:     f64f341c-655d-11eb-8778-fa163e914bcc
    health: HEALTH_OK

services:
    mon: 3 daemons, quorum host01,host02,host03 (age 94m)
    mgr: host01.lbnhug(active, since 59m), standbys: host02.rofgay, host03.ohipra
    mds: 1/1 daemons up, 1 standby
    osd: 18 osds: 18 up (since 10m), 18 in (since 10m)
    rgw: 4 daemons active (2 hosts, 1 zones)

data:
    volumes: 1/1 healthy
    pools:   8 pools, 225 pgs
    objects: 230 objects, 9.9 KiB
    usage:   271 MiB used, 269 GiB / 270 GiB avail
    pgs:     225 active+clean

io:
    client:   85 B/s rd, 0 op/s rd, 0 op/s wr
```

Exiting the cephadm shell

To exit the cephadm shell, use the **exit** command.

```
[ceph: root@host01 /]# exit
[root@host01 ~]#
```

cephadm commands

The cephadm is a command line tool to manage the local host for the Cephadm Orchestrator. It provides commands to investigate and modify the state of the current host.

Some of the commands are generally used for debugging.

Note: cephadm is not required on all hosts, however, it is useful when investigating a particular daemon. The cephadm-ansible-preflight playbook installs cephadm on all hosts and the cephadm-ansible purge playbook requires cephadm be installed on all hosts to work properly.

Table 2. The ***cephadm*** *commands*

Command	Description	Syntax	Example
adopt	Convert an upgraded storage cluster daemon to run `cephadm`.	`cephadm adopt [-h] --name DAEMON_NAME --style STYLE [--cluster CLUSTER] --legacy-dir [LEGACY_DIR] --config-json CONFIG_JSON] [--skip-firewalld] [--skip-pull]`	`[root@host01 ~]#` **`cephadm adopt --style=legacy --name prometheus.host02`**
ceph-volume	This command is used to list all the devices on the particular host. Run the `ceph-volume` command inside a container Deploys OSDs with different device technologies like `lvm` or physical disks using pluggable tools and follows a predictable, and robust way of preparing, activating, and starting OSDs.	`cephadm ceph-volume inventory/simple/raw/lvm [-h] [--fsid FSID] [--config-json CONFIG_JSON] [--config CONFIG, -c CONFIG] [--keyring KEYRING, -k KEYRING]`	`[root@host01 ~]# cephadm ceph-volume inventory --fsid f64f341c-655d-11eb-8778-fa163e914bcc`
check-host	Check the host configuration that is suitable for a Ceph cluster.	`cephadm check-host [--expect-hostname HOSTNAME]`	`[root@host01 ~]# cephadm check-host --expect-hostname host02`
deploy	Deploys a daemon on the local host.	`cephadm shell deploy DAEMON_TYPE [-h] [--name DAEMON_NAME] [--fsid FSID] [--config CONFIG, -c CONFIG] [--config-json CONFIG_JSON] [--keyring KEYRING] [--key KEY] [--osd-fsid OSD_FSID] [--skip-firewalld] [--tcp-ports TCP_PORTS] [--reconfig] [--allow-ptrace] [--memory-request MEMORY_REQUEST] [--memory-limit MEMORY_LIMIT] [--meta-json META_JSON]`	`[root@host01 ~]# cephadm shell deploy mon --fsid f64f341c-655d-11eb-8778-fa163e914bcc`
enter	Run an interactive shell inside a running daemon container.	`cephadm enter [-h] [--fsid FSID] --name NAME [command [command …]]`	`[root@host01 ~]# cephadm enter --name 52c611f2b1d9`
help	View all the commands supported by `cephadm`.	`cephadm help`	`[root@host01 ~]# cephadm help`

Table 2. The **cephadm** commands (continued)

Command	Description	Syntax	Example
`install`	Install the packages.	`cephadm install PACKAGES`	`[root@host01 ~]# cephadm install ceph-common ceph-osd`
`inspect-image`	Inspect the local Ceph container image.	`cephadm --image IMAGE_ID inspect-image`	`[root@host01 ~]# cephadm --image 13ea90216d0be03003d12d7869f72ad9de5cec9e54a27fd308e01e467c0d4a0a inspect-image`
`list-networks`	List the IP networks.	`cephadm list-networks`	`[root@host01 ~]# cephadm list-networks`
`ls`	List daemon instances known to `cephadm` on the hosts. You can use `--no-detail` for the command to run faster, which gives details of the daemon name, fsid, style, and systemd unit per daemon. You can use `--legacy-dir` option to specify a legacy base directory to search for daemons.	`cephadm ls [--no-detail] [--legacy-dir LEGACY_DIR]`	`[root@host01 ~]# cephadm ls --no-detail`
`logs`	Print `journald` logs for a daemon container. This is similar to the `journalctl` command.	`cephadm logs [--fsid FSID] --name DAEMON_NAME` `cephadm logs [--fsid FSID] --name DAEMON_NAME -- -n NUMBER # Last N lines` `cephadm logs [--fsid FSID] --name DAEMON_NAME -- -f # Follow the logs`	`[root@host01 ~]# cephadm logs --fsid 57bddb48-ee04-11eb-9962-001a4a000672 --name osd.8` `[root@host01 ~]# cephadm logs --fsid 57bddb48-ee04-11eb-9962-001a4a000672 --name osd.8 -- -n 20` `[root@host01 ~]# cephadm logs --fsid 57bddb48-ee04-11eb-9962-001a4a000672 --name osd.8 -- -f`
`prepare-host`	Prepare a host for `cephadm`.	`cephadm prepare-host [--expect-hostname HOSTNAME]`	`[root@host01 ~]# cephadm prepare-host` `[root@host01 ~]# cephadm prepare-host --expect-hostname host01`
`pull`	Pull the Ceph image.	`cephadm [-h] [--image IMAGE_ID] pull`	`[root@host01 ~]# cephadm --image 13ea90216d0be03003d12d7869f72ad9de5cec9e54a27fd308e01e467c0d4a0a pull`

*Table 2. The **cephadm** commands (continued)*

Command	Description	Syntax	Example

`registry-login`	Give cephadm login information for an authenticated registry. Cephadm attempts to log the calling host into that registry.	`cephadm registry-login --registry-url REGISTRY_URL --registry-username USERNAME --registry-password PASSWORD [--fsid FSID] [--registry-json JSON_FILE]`
		You can also use a JSON registry file containing the login info formatted as:
		`cat REGISTRY_FILE`
		`{`
		`"url":"REGISTRY_URL",`
		`"username":"REGISTRY_USERNAME",`
		`"password":"REGISTRY_PASSWORD"`
		`}`
			`[root@host01 ~]# cephadm registry-login --registry-url cp.icr.io/cp --registry-username myuser1 --registry-password mypassword1`
			`[root@host01 ~]# cat registry_file`
			`{`
			` "url":"cp.icr.io/cp",`
			`"username":"myuser",`
			` "password":"mypass"`
			`}`
			`[root@host01 ~]# cephadm registry-login -i registry_file`

`rm-daemon`	Remove a specific daemon instance. If you run the `cephadm rm-daemon` command on the host directly, although the command removes the daemon, the `cephadm mgr` module notices that the daemon is missing and redeploys it. This command is problematic and should be used only for experimental purposes and debugging.	`cephadm rm-daemon --fsid FSID --name DAEMON_NAME [--force ] [--force-delete-data]`	`[root@host01 ~]# cephadm rm-daemon --fsid f64f341c-655d-11eb-8778-fa163e914bcc --name osd.8`

*Table 2. The **cephadm** commands (continued)*

Command	Description	Syntax	Example
rm-cluster	Remove all the daemons from a storage cluster on that specific host where it is run. Similar to `rm-daemon`, if you remove a few daemons this way and the Ceph Orchestrator is not paused and some of those daemons belong to services that are not unmanaged, the `cephadm` orchestrator just redeploys them there.	`cephadm rm-cluster --fsid FSID [--force]`	`[root@host01 ~]# cephadm rm-cluster --fsid f64f341c-655d-11eb-8778-fa163e914bcc`
rm-repo	Remove a package repository configuration. This is mainly used for the disconnected installation of IBM Storage Ceph.	`cephadm rm-repo [-h]`	`[root@host01 ~]# cephadm rm-repo`
run	Run a Ceph daemon, in a container, in the foreground.	`cephadm run [--fsid FSID] --name DAEMON_NAME`	`[root@host01 ~]# cephadm run --fsid f64f341c-655d-11eb-8778-fa163e914bcc --name osd.8`
shell	Run an interactive shell with access to Ceph commands over the inferred or specified Ceph cluster. You can enter the shell using the `cephadm shell` command and run all the orchestrator commands within the shell.	`cephadm shell [--fsid FSID] [--name DAEMON_NAME, -n DAEMON_NAME] [--config CONFIG, -c CONFIG] [--mount MOUNT, -m MOUNT] [--keyring KEYRING, -k KEYRING] [--env ENV, -e ENV]`	`[root@host01 ~]# cephadm shell -- ceph orch ls` `[root@host01 ~]# cephadm shell`
unit	Start, stop, restart, enable, and disable the daemons with this operation. This operates on the daemon's `systemd` unit.	`cephadm unit [--fsid FSID] --name DAEMON_NAME start/stop/restart/enable/disable`	`[root@host01 ~]# cephadm unit --fsid f64f341c-655d-11eb-8778-fa163e914bcc --name osd.8 start`
version	Provides the version of the storage cluster.	`cephadm version`	`[root@host01 ~]# cephadm version`

Verifying the cluster installation

Once the cluster installation is complete, you can verify that the IBM Storage Ceph installation is running properly.

There are two ways of verifying the storage cluster installation as a root user:

- Run the **podman ps** command.
- Run the **cephadm shell ceph -s** command.

Prerequisites

- Root-level access to all nodes in the storage cluster.

Procedure

- Run the **podman ps** command:

  Example

  ```
  [root@host01 ~]# podman ps
  ```

  Note: In the NAMES column, the unit files now include the `FSID`.

- Run the **cephadm shell ceph -s** command:

  Example

  ```
  [root@host01 ~]# cephadm shell ceph -s

    cluster:
      id:     f64f341c-655d-11eb-8778-fa163e914bcc
      health: HEALTH_OK

    services:
      mon: 3 daemons, quorum host01,host02,host03 (age 94m)
      mgr: host01.lbnhug(active, since 59m), standbys: host02.rofgay, host03.ohipra
      mds: 1/1 daemons up, 1 standby
      osd: 18 osds: 18 up (since 10m), 18 in (since 10m)
      rgw: 4 daemons active (2 hosts, 1 zones)

    data:
      volumes: 1/1 healthy
      pools:   8 pools, 225 pgs
      objects: 230 objects, 9.9 KiB
      usage:   271 MiB used, 269 GiB / 270 GiB avail
      pgs:     225 active+clean

    io:
      client:   85 B/s rd, 0 op/s rd, 0 op/s wr
  ```

  Note: The health of the storage cluster is in *HEALTH_WARN* status as the hosts and the daemons are not added.

Adding hosts

Bootstrapping the IBM Storage Ceph installation creates a working storage cluster, consisting of one Monitor daemon and one Manager daemon within the same container. As a storage administrator, you can add additional hosts to the storage cluster and configure them.

Important: For adding hosts with disconnected installations, see "Adding hosts in disconnected deployments" on page 38.

Note:

- Running the preflight playbook installs `podman`, `lvm2`, `chronyd`, and `cephadm` on all hosts listed in the Ansible inventory file.
- When using a custom registry, be sure to log in to the custom registry on newly added nodes before adding any Ceph daemons.

  Syntax

  ```
  ceph cephadm registry-login --registry-url CUSTOM_REGISTRY_NAME --registry_username
  REGISTRY_USERNAME --registry_password REGISTRY_PASSWORD
  ```

Example

```
# ceph cephadm registry-login --registry-url myregistry --registry_username myregistryusername
--registry_password myregistrypassword1
```

Prerequisites

- A running IBM Storage Ceph cluster.
- Root-level or user with sudo access to all nodes in the storage cluster.
- Register the nodes to IBM subscription.
- Ansible user with sudo and passwordless `ssh` access to all nodes in the storage cluster.

Procedure

Note: In the following procedure, use either `root`, as indicated, or the username with which the user is bootstrapped.

1. From the node that contains the admin keyring, install the storage cluster's public SSH key in the root user's `authorized_keys` file on the new host:

   Syntax

   ```
   ssh-copy-id -f -i /etc/ceph/ceph.pub user@NEWHOST
   ```

   Example

   ```
   [root@host01 ~]# ssh-copy-id -f -i /etc/ceph/ceph.pub root@host02
   [root@host01 ~]# ssh-copy-id -f -i /etc/ceph/ceph.pub root@host03
   ```

2. Navigate to the `/usr/share/cephadm-ansible` directory on the Ansible administration node.

   Example

   ```
   [ansible@admin ~]$ cd /usr/share/cephadm-ansible
   ```

3. From the Ansible administration node, add the new host to the Ansible inventory file. The default location for the file is `/usr/share/cephadm-ansible/hosts`. The following example shows the structure of a typical inventory file:

   Note: If you have previously added the new host to the Ansible inventory file and run the preflight playbook on the host, skip to step 4.

   Example

   ```
   [ansible@admin ~]$ cat hosts

   host02
   host03
   host04

   [admin]
   host01
   ```

4. Run the preflight playbook with the `--limit` option:

   Syntax

   ```
   ansible-playbook -i INVENTORY_FILE cephadm-preflight.yml --extra-vars "ceph_origin=ibm" --
   limit NEWHOST
   ```

   Example

   ```
   [ansible@admin cephadm-ansible]$ ansible-playbook -i hosts cephadm-preflight.yml --extra-
   vars "ceph_origin=ibm" --limit host02
   ```

The preflight playbook installs `podman`, `lvm2`, `chronyd`, and `cephadm` on the new host. After installation is complete, `cephadm` resides in the `/usr/sbin/` directory.

For Red Hat Enterprise Linux 9, install `podman`, `lvm2`, `chronyd`, and `cephadm` manually:

Example

```
[root@host01 ~]# dnf install podman lvm2 chronyd cephadm
```

5. From the bootstrap node, use the `cephadm` orchestrator to add the new host to the storage cluster:

   Syntax

   ```
   ceph orch host add NEWHOST
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph orch host add host02
   Added host 'host02' with addr '10.10.128.69'
   [ceph: root@host01 /]# ceph orch host add host03
   Added host 'host03' with addr '10.10.128.70'
   ```

6. Optional: You can also add nodes by IP address, before and after you run the preflight playbook. If you do not have DNS configured in your storage cluster environment, you can add the hosts by IP address, along with the host names.

   Syntax

   ```
   ceph orch host add HOSTNAME IP_ADDRESS
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph orch host add host02 10.10.128.69
   Added host 'host02' with addr '10.10.128.69'
   ```

   - View the status of the storage cluster and verify that the new host has been added. The *STATUS* of the hosts is blank, in the output of the `ceph orch host ls` command.

     Example

     ```
     [ceph: root@host01 /]# ceph orch host ls
     ```

Reference

- "Registering the IBM Storage Ceph nodes" on page 12
- "Creating an Ansible user with sudo access" on page 15

Using the `addr` option to identify hosts

The `addr` option offers an additional way to contact a host. Add the IP address of the host to the `addr` option. If `ssh` cannot connect to the host by its hostname, then it uses the value stored in `addr` to reach the host by its IP address.

Prerequisites

- A storage cluster that has been installed and bootstrapped.
- Root-level access to all nodes in the storage cluster.

Procedure

1. Log in to the `cephadm` shell:

Example

```
[root@host01 ~]# cephadm shell
```

2. Add the IP address:

   Syntax

   ```
   ceph orch host add HOSTNAME IP_ADDR
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph orch host add host02 10.10.128.68
   ```

   Note: If adding a host by hostname results in that host being added with an IPv6 address instead of an IPv4 address, use the **`ceph orch host`** command to specify the IP address of that host:

   Syntax

   ```
   ceph orch host set-addr HOSTNAME IP_ADDR
   ```

   To convert the IP address from IPv6 format to IPv4 format for a host you have added, use the following command:

   ```
   ceph orch host set-addr HOSTNAME IPV4_ADDRESS
   ```

Adding multiple hosts

Add multiple hosts at a time to the storage cluster, by using a YAML file.

Note: Be sure to create the `hosts.yaml` file within a host container, or create the file on the local host and then use the `cephadm` shell to mount the file within the container. The `cephadm` shell automatically places mounted files in `/mnt`. If you create the file directly on the local host and then apply the `hosts.yaml` file instead of mounting it, you might see a `File does not exist` error.

Prerequisites

- A storage cluster that has been installed and bootstrapped.
- Root-level access to all nodes in the storage cluster.

Procedure

1. Copy over the public `ssh` key to each of the hosts that you want to add.
2. Use a text editor to create a `hosts.yaml` file.
3. Add the host descriptions to the `hosts.yaml` file, as shown in the following example. Include the labels to identify placements for the daemons that you want to deploy on each host. Separate each host description with three dashes (---).

   Example

   ```
   service_type: host
   addr:
   hostname: host02
   labels:
   - mon
   - osd
   - mgr
   ---
   service_type: host
   addr:
   hostname: host03
   labels:
   - mon
   - osd
   - mgr
   ---
   ```

```
service_type: host
addr:
hostname: host04
labels:
- mon
- osd
```

4. If you created the `hosts.yaml` file directly on the local host, use the `cephadm` shell to mount the file:

   Example

   ```
   [root@host01 ~]# cephadm shell --mount hosts.yaml -- ceph orch apply -i /mnt/hosts.yaml
   ```

5. If you created the `hosts.yaml` file within the host container, invoke the `ceph orch apply` command:

   Example

   ```
   [ceph: root@host01 /]# ceph orch apply -i hosts.yaml
   Added host 'host02' with addr '10.10.128.69'
   Added host 'host03' with addr '10.10.128.70'
   Added host 'host04' with addr '10.10.128.71'
   ```

6. View the list of hosts and their labels:

   Example

   ```
   [ceph: root@host01 /]# ceph orch host ls
   HOST      ADDR      LABELS          STATUS
   host02   host02    mon osd mgr
   host03   host03    mon osd mgr
   host04   host04    mon osd
   ```

   Note: If a host is online and operating normally, its status is blank. An offline host shows a status of OFFLINE, and a host in maintenance mode shows a status of MAINTENANCE.

Removing hosts

Remove hosts of a Ceph cluster with the Ceph Orchestrators.

All daemons are removed with the `drain` option which adds the `_no_schedule` label to ensure that you cannot deploy any daemons or a cluster till the operation is complete.

Important: If you are removing the bootstrap host, be sure to copy the admin keyring and the configuration file to another host in the storage cluster before you remove the host.

Prerequisites

- A running IBM Storage Ceph cluster.
- Root-level access to all the nodes.
- Hosts are added to the storage cluster.
- All the services are deployed.
- Cephadm is deployed on the nodes where the services have to be removed.

Procedure

1. Log into the Cephadm shell:

   Example

   ```
   [root@host01 ~]# cephadm shell
   ```

2. Fetch the host details:

Example

```
[ceph: root@host01 /]# ceph orch host ls
```

3. Drain all the daemons from the host:

   Syntax

   ```
   ceph orch host drain HOSTNAME
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph orch host drain host02
   ```

   The `_no_schedule` label is automatically applied to the host which blocks deployment.

4. Check the status of OSD removal:

   Example

   ```
   [ceph: root@host01 /]# ceph orch osd rm status
   ```

   When no placement groups (PGs) remain on the OSD, the OSD is decommissioned and removed from the storage cluster.

5. Check if all the daemons are removed from the storage cluster:

   Syntax

   ```
   ceph orch ps HOSTNAME
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph orch ps host02
   ```

6. Remove the host:

   Syntax

   ```
   ceph orch host rm HOSTNAME
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph orch host rm host02
   ```

Reference

- Adding hosts using the Ceph Orchestrator
- Listing hosts using the Ceph Orchestrator

Labeling hosts

The Ceph Orchestrator supports assigning labels to hosts. Labels are free-form and have no specific meanings. This means that you can use `mon`, `monitor`, `mycluster_monitor`, or any other text string. Each host can have multiple labels.

For example, the `mon` label can be applied to all hosts on which you want to deploy Ceph Monitor daemons, `mgr` for all hosts on which you want to deploy Ceph Manager daemons, `rgw` for Ceph Object Gateway daemons, and so on.

Labeling all the hosts in the storage cluster helps to simplify system management tasks by allowing you to quickly identify the daemons running on each host. In addition, you can use the Ceph orchestrator or a YAML file to deploy or remove daemons on hosts that have specific host labels.

Prerequisites

- A storage cluster that has been installed and bootstrapped.

Adding labels to a host

Use the Ceph Orchestrator to add a label to a host. Labels can be used to specify placement of daemons.

A few examples of labels are `mgr`, `mon`, and `osd` based on the service deployed on the hosts. Each host can have multiple labels.

You can also add the following host labels that have special meaning to `cephadm` and they begin with `_`:

- `_no_schedule`: This label prevents `cephadm` from scheduling or deploying daemons on the host. If it is added to an existing host that already contains Ceph daemons, it causes `cephadm` to move those daemons elsewhere, except OSDs which are not removed automatically. When a host is added with the `_no_schedule` label, no daemons are deployed on it. When the daemons are drained before the host is removed, the `_no_schedule` label is set on that host.
- `_no_autotune_memory`: This label does not autotune memory on the host. It prevents the daemon memory from being tuned even when the `osd_memory_target_autotune` option or other similar options are enabled for one or more daemons on that host.
- `_admin`: By default, the `_admin` label is applied to the bootstrapped host in the storage cluster and the `client.admin` key is set to be distributed to that host with the `ceph orch client-keyring {ls|set|rm}` function. Adding this label to additional hosts normally causes `cephadm` to deploy configuration and keyring files in the `/etc/ceph` directory.

Prerequisites

- A storage cluster that has been installed and bootstrapped.
- Root-level access to all nodes in the storage cluster.
- Hosts are added to the storage cluster.

Procedure

1. Launch the `cephadm` shell:

   ```
   [root@host01 ~]# cephadm shell
   [ceph: root@host01 /]#
   ```

2. Add a label to a host:

   Syntax

   ```
   ceph orch host label add HOSTNAME LABEL
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph orch host label add host02 mon
   ```

Verification

- List the hosts:

  Example

  ```
  [ceph: root@host01 /]# ceph orch host ls
  ```

Removing labels from a host

Use the Ceph Orchestrator to remove a label from a host.

Before you begin

- A IBM Storage Ceph cluster that has been installed and boostrapped.
- Root-level access to all nodes in the storage cluster.
- Hosts are added to the storage cluster.

Procedure

1. Log into the Cephadm shell.

```
[root@host01 ~]# cephadm shell
```

2. Remove the label.

```
ceph orch host label rm HOST_NAME LABEL_NAME
```

For example,

```
[ceph: root@host01 /]# ceph orch host label rm host02 mon
```

What to do next

Verify that the label has been moved from the host, by using the **`ceph orch host ls`** command.

Using host labels to deploy daemons on specific hosts

Use host labels to deploy daemons to specific hosts.

About this task

There are two ways to use host labels to deploy daemons on specific hosts:

- By using the **`--placement`** option from the command line.

  For example, **`ceph orch apply DAEMON --placement="label:LABEL"`**
- By using a YAML file.

Before you begin

- A storage cluster that has been installed and bootstrapped.
- Root-level access to all nodes in the storage cluster.

Procedure

1. Log into the Cephadm shell.

   Example

```
[root@host01 ~]# cephadm shell
```

2. List all current hosts and labels.
   Example

```
[ceph: root@host01 /]# ceph orch host ls

HOST      ADDR     LABELS               STATUS
host01              _admin mon osd mgr
host02              mon osd mgr mylabel
```

3. Optional: Use the **`--placement`** option to deploy a daemon from the command line.

Deploying daemons from the command line

Procedure

1. Log into the Cephadm shell.

   Example

   ```
   [root@host01 ~]# cephadm shell
   ```

2. List all current hosts and labels.
   Example

   ```
   [ceph: root@host01 /]# ceph orch host ls

   HOST      ADDR     LABELS               STATUS
   host01              _admin mon osd mgr
   host02              mon osd mgr mylabel
   ```

3. Use the **`--placement`** option to deploy a daemon from the command line.

   Syntax

   ```
   ceph orch apply DAEMON --placement="label:LABEL"
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph orch apply prometheus --placement="label:mylabel"
   ```

Deploying daemons with a YAML file

Procedure

1. Log into the Cephadm shell.

   Example

   ```
   [root@host01 ~]# cephadm shell
   ```

2. List all current hosts and labels.
   Example

   ```
   [ceph: root@host01 /]# ceph orch host ls

   HOST      ADDR     LABELS               STATUS
   host01              _admin mon osd mgr
   host02              mon osd mgr mylabel
   ```

3. Assign the daemon to a specific host label in a YAML file.

   Be sure to specify the service type and label.

   a) Create the `placement.yml` file.

   ```
   [ceph: root@host01 /]# vi placement.yml
   ```

   b) Specify the service type and label within the YAML file.

   Example

   ```
   service_type: prometheus
   placement:
     label: "mylabel"
   ```

   c) Apply the daemon placement file.

Syntax

```
ceph orch apply -i FILENAME
```

Example

```
[ceph: root@host01 /]# ceph orch apply -i placement.yml
Scheduled prometheus update…
```

What to do next

Verify the status of the daemons.

Syntax

```
ceph orch ps --daemon_type=DAEMON_NAME
```

Example

```
[ceph: root@host01 /]# ceph orch ps --daemon_type=prometheus
NAME               HOST    PORTS    STATUS         REFRESHED  AGE  MEM USE  MEM LIM  VERSION
IMAGE ID      CONTAINER ID
prometheus.host02  host02  *:9095  running (2h)     8m ago   2h    85.3M        -  2.22.2
ac25aac5d567  ad8c7593d7c0
```

Adding Monitor service

A typical IBM Storage Ceph storage cluster has three or five monitor daemons deployed on different hosts. If your storage cluster has five or more hosts, deploy five Monitor nodes.

About this task

Note:

- In the case of a firewall, see Firewall settings for Ceph Monitor node.
- The bootstrap node is the initial monitor of the storage cluster. Be sure to include the bootstrap node in the list of hosts to which you want to deploy.
- If you want to apply Monitor service to more than one specific host, be sure to specify all of the host names within the same **`ceph orch apply`** command. If you specify **`ceph orch apply mon --placement host1`** and then specify **`ceph orch apply mon --placement host2`**, the second command removes the Monitor service on host1 and applies a Monitor service to host2.

If your Monitor nodes or your entire cluster are located on a single subnet, then `cephadm` automatically adds up to five Monitor daemons as you add new hosts to the cluster. `cephadm` automatically configures the Monitor daemons on the new hosts. The new hosts reside on the same subnet as the first (bootstrap) host in the storage cluster. `cephadm` can also deploy and scale monitors to correspond to changes in the size of the storage cluster.

Before you begin

- Root-level access to all hosts in the storage cluster.
- A running storage cluster.

Procedure

1. Apply five Monitor daemons to five random hosts in the storage cluster.

   ```
   ceph orch apply mon 5
   ```

2. Disable automatic Monitor deployment.

```
ceph orch apply mon --unmanaged
```

Deploying Ceph monitor nodes using host labels

A typical IBM Storage Ceph storage cluster has three or five Ceph Monitor daemons deployed on different hosts. If your storage cluster has five or more hosts, deploy five Ceph Monitor nodes. Use host labels to identify the hosts that contain Ceph Monitor nodes.

If your Ceph Monitor nodes or your entire cluster are located on a single subnet, then `cephadm` automatically adds up to five Ceph Monitor daemons as you add new nodes to the cluster. `cephadm` automatically configures the Ceph Monitor daemons on the new nodes. The new nodes reside on the same subnet as the first (bootstrap) node in the storage cluster. `cephadm` can also deploy and scale monitors to correspond to changes in the size of the storage cluster.

Prerequisites

- Root-level access to all nodes in the storage cluster.
- A running storage cluster.

Procedure

1. Assign the mon label to the host:

   Syntax

   ```
   ceph orch host label add HOSTNAME mon
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph orch host label add host02 mon
   [ceph: root@host01 /]# ceph orch host label add host03 mon
   ```

2. View the current hosts and labels:

   Syntax

   ```
   ceph orch host ls
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph orch host ls
   HOST   ADDR   LABELS  STATUS
   host01        mon,mgr,_admin
   host02        mon
   host03        mon
   host04
   host05
   host06
   ```

   - Deploy Ceph Monitor daemons based on the host label:

     Syntax

     ```
     ceph orch apply mon label:mon
     ```

   - Deploy Ceph Monitor daemons on a specific set of hosts:

     Syntax

     ```
     ceph orch apply mon HOSTNAME1,HOSTNAME2,HOSTNAME3
     ```

     Example

     ```
     [ceph: root@host01 /]# ceph orch apply mon host01,host02,host03
     ```

Note: Be sure to include the bootstrap node in the list of hosts to which you want to deploy.

Adding Ceph Monitor nodes by IP address or network name

A typical IBM Storage Ceph storage cluster has three or five Ceph Monitor daemons deployed on different hosts. If your storage cluster has five or more hosts, deploy five Ceph Monitor nodes. Add Ceph Monitor nodes by IP address or network name.

A typical IBM Storage Ceph storage cluster has three or five monitor daemons deployed on different hosts. If your storage cluster has five or more hosts, IBM recommends that you deploy five Monitor nodes.

If your Monitor nodes or your entire cluster are located on a single subnet, then `cephadm` automatically adds up to five Monitor daemons as you add new nodes to the cluster. You do not need to configure the Monitor daemons on the new nodes. The new nodes reside on the same subnet as the first node in the storage cluster. The first node in the storage cluster is the bootstrap node. `cephadm` can also deploy and scale monitors to correspond to changes in the size of the storage cluster.

Prerequisites

- Root-level access to all nodes in the storage cluster.
- A running storage cluster.

Procedure

- To deploy each additional Ceph Monitor node:

  Syntax

  ```
  ceph orch apply mon NODE:IP_ADDRESS_OR_NETWORK_NAME [NODE:IP_ADDRESS_OR_NETWORK_NAME...]
  ```

  Example

  ```
  [ceph: root@host01 /]# ceph orch apply mon host02:10.10.128.69 host03:mynetwork
  ```

Setting up a custom SSH key on an existing cluster

As a storage administrator, with Cephadm, you can use an SSH key to securely authenticate with remote hosts. The SSH key is stored in the monitor to connect to remote hosts.

About this task

When the cluster is bootstrapped, this SSH key is generated automatically and no additional configuration is necessary. However, you can generate a new SSH key with the **`ceph cephadm generate-key`** command.

Before you begin

- A running IBM Storage Ceph cluster.
- An Ansible administration node.
- Root-level access to the Ansible administration node.
- The `cephadm-ansible` package is installed on the node.

Procedure

1. Navigate to the `cephadm-ansible` directory.
2. Generate a new SSH key.

   ```
   ceph cephadm generate-key
   ```

   For example,

```
[ceph-admin@admin cephadm-ansible]$ ceph cephadm generate-key
```

3. Retrieve the public portion of the SSH key.

   ```
   ceph cephadm get-pub-key
   ```

   For example,

   ```
   [ceph-admin@admin cephadm-ansible]$ ceph cephadm get-pub-key
   ```

4. Delete the currently stored SSH key.

   ```
   ceph cephadm clear-key
   ```

   For example,

   ```
   [ceph-admin@admin cephadm-ansible]$ceph cephadm clear-key
   ```

5. Restart the `mgr` daemon to reload the configuration.

   ```
   ceph mgr fail
   ```

   For example,

   ```
   [ceph-admin@admin cephadm-ansible]$ ceph mgr fail
   ```

Configuring a different SSH user

As a storage administrator, you can configure a non-root SSH user who can log in to all the Ceph cluster nodes with enough privileges to download container images, start containers, and run commands without prompting for a password.

Important: Before configuring a non-root SSH user, the cluster SSH key needs to be added to the user's `authorized_keys` file and non-root users must have `passwordless` sudo access.

Prerequisites

- A running IBM Storage Ceph cluster.
- An Ansible administration node.
- Root-level access to the Ansible administration node.
- The `cephadm-ansible` package is installed on the node.
- Add the cluster SSH keys to the user's `authorized_keys`.
- Enable `passwordless` sudo access for the non-root users.

Procedure

1. Go to the `cephadm-ansible` directory.
2. Provide Cephadm the name of the user who is going to perform all the Cephadm operations.

   Syntax

   ```
   ceph cephadm set-user USER
   ```

   Example

   ```
   [ceph-admin@admin cephadm-ansible]$ ceph cephadm set-user user
   ```

3. Retrieve the SSH public key.

Syntax

```
ceph cephadm get-pub-key > ~/ceph.pub
```

Example

```
[ceph-admin@admin cephadm-ansible]$ ceph cephadm get-pub-key > ~/ceph.pub
```

4. Copy the SSH keys to all the hosts.

   Syntax

   ```
   ssh-copy-id -f -i ~/ceph.pub USER@HOST
   ```

   Example

   ```
   [ceph-admin@admin cephadm-ansible]$ ssh-copy-id ceph-admin@host01
   ```

Setting up the admin node

Use an admin node to administer the storage cluster.

An admin node contains both the cluster configuration file and the admin keyring. Both of these files are stored in the directory `/etc/ceph` and use the name of the storage cluster as a prefix.

For example, the default ceph cluster name is `ceph`. In a cluster using the default name, the admin keyring is named `/etc/ceph/ceph.client.admin.keyring`. The corresponding cluster configuration file is named `/etc/ceph/ceph.conf`.

To set up additional hosts in the storage cluster as admin nodes, apply the `_admin` label to the host you want to designate as an administrator node.

Note: By default, after applying the `_admin` label to a node, `cephadm` copies the `ceph.conf` and `client.admin` keyring files to that node. The `_admin` label is automatically applied to the bootstrap node unless the `--skip-admin-label` option was specified with the `cephadm bootstrap` command.

Prerequisites

- A running storage cluster with `cephadm` installed.
- The storage cluster has running Monitor and Manager nodes.
- Root-level access to all nodes in the cluster.

Procedure

1. Use **`ceph orch host ls`** to view the hosts in your storage cluster:

   Example

   ```
   [root@host01 ~]# ceph orch host ls
   HOST   ADDR   LABELS  STATUS
   host01        mon,mgr,_admin
   host02        mon
   host03        mon,mgr
   host04
   host05
   host06
   ```

2. Use the `_admin` label to designate the admin host in your storage cluster. For best results, this host should have both Monitor and Manager daemons running.

   Syntax

   ```
   ceph orch host label add HOSTNAME _admin
   ```

Example

```
[root@host01 ~]#  ceph orch host label add host03 _admin
```

3. Verify that the admin host has the `_admin` label.

   Example

```
[root@host01 ~]#  ceph orch host ls
HOST   ADDR   LABELS  STATUS
host01        mon,mgr,_admin
host02        mon
host03        mon,mgr,_admin
host04
host05
host06
```

4. Log in to the admin node to manage the storage cluster.

Removing the admin label from a host

You can use the Ceph orchestrator to remove the admin label from a host.

Prerequisites

- A running storage cluster with `cephadm` installed and bootstrapped.
- The storage cluster has running Monitor and Manager nodes.
- Root-level access to all nodes in the cluster.

Procedure

1. Use the **`ceph orch host ls`** command to view the hosts and identify the admin host in your storage cluster:

   Example

```
 [root@host01 ~]# ceph orch host ls
 HOST   ADDR   LABELS  STATUS
 host01        mon,mgr,_admin
 host02        mon
 host03        mon,mgr,_admin
 host04
 host05
 host06
```

2. Log into the Cephadm shell:

   Example

```
 [root@host01 ~]# cephadm shell
```

3. Use the ceph orchestrator to remove the admin label from a host:

   Syntax

```
 ceph orch host label rm HOSTNAME LABEL
```

   Example

```
 [ceph: root@host01 /]# ceph orch host label rm host03 _admin
```

4. Verify that the admin host has the `_admin` label.

   Example

```
 [root@host01 ~]#  ceph orch host ls
 HOST   ADDR   LABELS  STATUS
 host01        mon,mgr,_admin
```

```
host02        mon
host03        mon,mgr
host04
host05
host06
```

Important: After removing the admin label from a node, ensure you remove the `ceph.conf` and `client.admin` keyring files from that node. Also, the node must be removed from the admin Ansible inventory file.

Adding Manager service

`cephadm` automatically installs a Manager daemon on the bootstrap node during the bootstrapping process. Use the Ceph orchestrator to deploy additional Manager daemons.

The Ceph orchestrator deploys two Manager daemons by default. To deploy a different number of Manager daemons, specify a different number. If you do not specify the hosts where the Manager daemons should be deployed, the Ceph orchestrator randomly selects the hosts and deploys the Manager daemons to them.

Note: If you want to apply Manager daemons to more than one specific host, be sure to specify all of the host names within the same **`ceph orch apply`** command. If you specify **`ceph orch apply mgr --placement host1`** and then specify **`ceph orch apply mgr --placement host2`**, the second command removes the Manager daemon on host1 and applies a Manager daemon to host2.

Use the **`--placement`** option to deploy to specific hosts.

Prerequisites

- A running storage cluster.

Procedure

- To specify that you want to apply a certain number of Manager daemons to randomly selected hosts:

  Syntax

  ```
  ceph orch apply mgr NUMBER_OF_DAEMONS
  ```

  Example

  ```
  [ceph: root@host01 /]# ceph orch apply mgr 3
  ```

- To add Manager daemons to specific hosts in your storage cluster:

  Syntax

  ```
  ceph orch apply mgr --placement "HOSTNAME1 HOSTNAME2 HOSTNAME3"
  ```

  Example

  ```
  [ceph: root@host01 /]# ceph orch apply mgr --placement "host02 host03 host04"
  ```

Adding OSDs

Use this information to deploy OSDs on a specific device or host.

Cephadm does not provision an OSD on a device that is not available. A storage device is considered available if it meets all of the following conditions:

- The device must have no partitions.
- The device must not be mounted.
- The device must not contain a file system.

- The device must not contain a Ceph BlueStore OSD.
- The device must be larger than 5 GB.

Important: By default, the **`osd_memory_target_autotune`** parameter is set to `true` in IBM Storage Ceph.

For information about tuning OSD memory automatically, see Automatically tuning OSD memory.

Prerequisites

- A running IBM Storage Ceph cluster.

Procedure

1. List the available devices to deploy OSDs:

   Syntax

   ```
   ceph orch device ls [--hostname=HOSTNAME1 HOSTNAME2] [--wide] [--refresh]
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph orch device ls --wide --refresh
   ```

2. You can either deploy the OSDs on specific hosts or on all the available devices:

   - To create an OSD from a specific device on a specific host:

     Syntax

     ```
     ceph orch daemon add osd HOSTNAME:DEVICE_PATH
     ```

     Example

     ```
     [ceph: root@host01 /]# ceph orch daemon add osd host02:/dev/sdb
     ```

   - To deploy OSDs on any available and unused devices, use the `--all-available-devices` option.

     Example

     ```
     [ceph: root@host01 /]# ceph orch apply osd --all-available-devices
     ```

     Note: This command creates colocated WAL and DB daemons. If you want to create non-colocated daemons, do not use this command.

Reference

- For more information about drive specifications for OSDs, see Advanced service specifications and filters for deploying OSDs
- For more information about zapping devices to clear data on devices, see Zapping devices for Ceph OSD deployment

Purging the Ceph storage cluster

Purging the Ceph storage cluster clears any data or connections that remain from previous deployments on your server. For Red Hat Enterprise Linux 9, use the **`cephadm rm-cluster`** command, since Ansible is not supported.

Before you begin

- A running IBM Storage Ceph.

Procedure

1. Disable `cephadm` to stop all the orchestration operations to avoid deploying new daemons.

   ```
   ceph mgr module disable cephadm
   ```

   For example,

   ```
   [ceph: root#host01 /]# ceph mgr module disable cephadm
   ```

2. Get the FSID of the cluster.

   ```
   ceph fsid
   ```

   For example,

   ```
   [ceph: root#host01 /]# ceph fsid
   ```

3. Exit the cephadm shell.
   For example,

   ```
   [ceph: root@host01 /]# exit
   ```

4. Purge the Ceph daemons from all hosts in the cluster.

   ```
   cephadm rm-cluster --force --zap-osds --fsid FSID
   ```

   For example,

   ```
   [root@host01 ~]# cephadm rm-cluster --force --zap-osds  --fsid 2d2fd136-6df1-11ea-
   ae74-002590e526e8
   ```

Deploying client nodes

As a storage administrator, you can deploy client nodes by running the `cephadm-preflight.yml` and `cephadm-clients.yml` playbooks.

The `cephadm-preflight.yml` playbook configures the Ceph repository and prepares the storage cluster for bootstrapping. It also installs some prerequisites, such as `podman`, `lvm2`, `chronyd`, and `cephadm`.

The `cephadm-clients.yml` playbook handles the distribution of configuration and keyring files to a group of Ceph clients.

Note: If you are not using the `cephadm-ansible` playbooks, after upgrading your Ceph cluster, you must upgrade the `ceph-common` package and client libraries on your client nodes. For more information, see Upgrading the IBM Storage Ceph cluster.

Prerequisites

- Root-level access to the Ansible administration node.
- Ansible user with sudo and passwordless `ssh` access to all nodes in the storage cluster.
- The `cephadm-ansible` package is installed.
- The prefight playbook has been run on the initial host in the storage cluster. For more information, see "Running the preflight playbook" on page 17.
- The *client_group* variable must be specified in the Ansible inventory file.
- The `[admin]` group is defined in the inventory file with a node where the admin keyring is present at `/etc/ceph/ceph.client.admin.keyring`.

Procedure

1. As an Ansible user, navigate to the `/usr/share/cephadm-ansible` directory on the Ansible administration node.

   Example

   ```
    [ceph-admin@admin ~]$ cd /usr/share/cephadm-ansible
   ```

2. Open and edit the `hosts` inventory file and add the `[clients]` group and clients to your inventory:

   Example

   ```
    host02
    host03
    host04

    [clients]
    client01
    client02
    client03

    [admin]
    host01
   ```

3. Run the `cephadm-preflight.yml` playbook to install the prerequisites on the clients:

   Syntax

   ```
    ansible-playbook -i INVENTORY_FILE cephadm-preflight.yml --limit CLIENT_GROUP_NAME|
   CLIENT_NODE_NAME
   ```

   Example

   ```
    [ceph-admin@admin cephadm-ansible]$ ansible-playbook -i hosts cephadm-preflight.yml --limit
   clients
   ```

4. Run the `cephadm-clients.yml` playbook to distribute the keyring and Ceph configuration files to a set of clients.

   a. To copy the keyring with a custom destination keyring name:

      Syntax

      ```
       ansible-playbook -i INVENTORY_FILE cephadm-clients.yml --extra-vars
      '{"fsid":"FSID","keyring":"KEYRING_PATH","client_group":"CLIENT_GROUP_NAME","conf":"CEPH_C
      ONFIGURATION_PATH","keyring_dest":"KEYRING_DESTINATION_PATH"}'
      ```

      - Replace INVENTORY_FILE with the Ansible inventory file name.
      - Replace FSID with the FSID of the cluster.
      - Replace KEYRING_PATH with the full path name to the keyring on the admin host that you want to copy to the client.
      - Optional: Replace CLIENT_GROUP_NAME with the Ansible group name for the clients to set up.
      - Optional: Replace CEPH_CONFIGURATION_PATH with the full path to the Ceph configuration file on the admin node.
      - Optional: Replace KEYRING_DESTINATION_PATH with the full path name of the destination where the keyring will be copied.

      Note: If you do not specify a configuration file with the `conf` option when you run the playbook, the playbook generates and distributes a minimal configuration file. By default, the generated file is located at `/etc/ceph/ceph.conf`.

      Example

      ```
       [ceph-admin@host01 cephadm-ansible]$ ansible-playbook -i hosts cephadm-
      clients.yml --extra-vars '{"fsid":"266ee7a8-2a05-11eb-b846-5254002d4916","keyring":"/etc/
      ```

```
ceph/ceph.client.admin.keyring","client_group":"clients","conf":"/etc/ceph/
ceph.conf","keyring_dest":"/etc/ceph/custom.name.ceph.keyring"}'
```

b. To copy a keyring with the default destination keyring name of `ceph.keyring` and using the default group of `clients`:

Syntax

```
 ansible-playbook -i INVENTORY_FILE cephadm-clients.yml --extra-vars
'{"fsid":"FSID","keyring":"KEYRING_PATH","conf":"CONF_PATH"}'
```

Example

```
 [ceph-admin@host01 cephadm-ansible]$ ansible-playbook -i hosts cephadm-
clients.yml --extra-vars '{"fsid":"266ee7a8-2a05-11eb-b846-5254002d4916","keyring":"/etc/
ceph/ceph.client.admin.keyring","conf":"/etc/ceph/ceph.conf"}'
```

Verification

- Log into the client nodes and verify that the keyring and configuration files exist.

  Example

```
[user@client01 ~]# ls -l /etc/ceph/

-rw-------. 1 ceph ceph 151 Jul 11 12:23 custom.name.ceph.keyring
-rw-------. 1 ceph ceph 151 Jul 11 12:23 ceph.keyring
-rw-------. 1 ceph ceph 269 Jul 11 12:23 ceph.conf
```

Reference

- For more information about admin keys, see Ceph User Management.
- For more information about the `cephadm-preflight` playbook, see "Running the preflight playbook" on page 17.

Managing an IBM Storage Ceph cluster using cephadm-ansible modules

As a storage administrator, you can use `cephadm-ansible` modules in Ansible playbooks to administer your IBM Storage Ceph cluster. The `cephadm-ansible` package provides several modules that wrap `cephadm` calls to let you write your own unique Ansible playbooks to administer your cluster.

Note: `cephadm-ansible` modules currently only support the most important tasks. Any operation not covered by `cephadm-ansible` modules must be completed using either the `command` or `shell` Ansible modules in your playbooks.

`cephadm-ansible` modules

The `cephadm-ansible` modules are a collection of modules that simplify writing Ansible playbooks by providing a wrapper around `cephadm` and `ceph orch` commands. You can use the modules to write your own unique Ansible playbooks to administer your cluster using one or more of the modules.

The `cephadm-ansible` package includes the following modules:

- `cephadm_bootstrap`
- `ceph_orch_host`
- `ceph_config`
- `ceph_orch_apply`
- `ceph_orch_daemon`
- `cephadm_registry_login`

`cephadm-ansible` modules options

This section lists the available options for the `cephadm-ansible` modules.

Options listed as required need to be set when using the modules in your Ansible playbooks. Options listed with a default value of `true` indicate that the option is automatically set when using the modules and you do not need to specify it in your playbook. For example, for the `cephadm_bootstrap` module, the Ceph Dashboard is installed unless you set `dashboard: false`.

Table 3. Available options for the `cephadm_bootstrap` module.

`cephadm_bootstrap`	**Description**	**Required**	**Default**
`mon_ip`	Ceph Monitor IP address.	true	
`image`	Ceph container image.	false	
`docker`	Use `docker` instead of `podman`.	false	
`fsid`	Define the Ceph FSID.	false	
`pull`	Pull the Ceph container image.	false	true
`dashboard`	Deploy the Ceph Dashboard.	false	true
`dashboard_user`	Specify a specific Ceph Dashboard user.	false	
`dashboard_password`	Ceph Dashboard password.	false	
`monitoring`	Deploy the monitoring stack.	false	true
`firewalld`	Manage firewall rules with firewalld.	false	true
`allow_overwrite`	Allow overwrite of existing --output-config, --output-keyring, or --output-pub-ssh-key files.	false	false
`registry_url`	URL for custom registry.	false	
`registry_username`	Username for custom registry.	false	
`registry_password`	Password for custom registry.	false	

Table 3. Available options for the `cephadm_bootstrap` module. (continued)

cephadm_bootstrap	**Description**	**Required**	**Default**
`registry_json`	JSON file with custom registry login information.	false	
`ssh_user`	SSH user to use for `cephadm` ssh to hosts.	false	
`ssh_config`	SSH config file path for `cephadm` SSH client.	false	
`allow_fqdn_hostname`	Allow hostname that is a fully-qualified domain name (FQDN).	false	false
`cluster_network`	Subnet to use for cluster replication, recovery and heartbeats.	false	

Table 4. Available options for the `ceph_orch_host` module.

ceph_orch_host	**Description**	**Required**	**Default**
`fsid`	The FSID of the Ceph cluster to interact with.	false	
`image`	The Ceph container image to use.	false	
`name`	Name of the host to add, remove, or update.	true	
`address`	IP address of the host.	true when `state` is `present`.	
`set_admin_label`	Set the `_admin` label on the specified host.	false	false
`labels`	The list of labels to apply to the host.	false	[]
`state`	If set to `present`, it ensures the name specified in `name` is present. If set to `absent`, it removes the host specified in `name`. If set to `drain`, it schedules to remove all daemons from the host specified in `name`.	false	present

Table 5. Available options for the `ceph_config` module

ceph_config	**Description**	**Required**	**Default**
`fsid`	The FSID of the Ceph cluster to interact with.	false	
`image`	The Ceph container image to use.	false	
`action`	Whether to `set` or `get` the parameter specified in `option`.	false	set
`who`	Which daemon to set the configuration to.	true	
`option`	Name of the parameter to `set` or `get`.	true	
`value`	Value of the parameter to set.	true if action is `set`	

Table 6. Available options for the `ceph_orch_apply` module.

ceph_orch_apply	**Description**	**Required**
`fsid`	The FSID of the Ceph cluster to interact with.	false
`image`	The Ceph container image to use.	false
`spec`	The service specification to apply.	true

Table 7. Available options for the `ceph_orch_daemon` module.

ceph_orch_daemon	**Description**	**Required**
`fsid`	The FSID of the Ceph cluster to interact with.	false
`image`	The Ceph container image to use.	false
`state`	The desired state of the service specified in `name`.	true If `started`, it ensures the service is started. If `stopped`, it ensures the service is stopped. If `restarted`, it will restart the service.
`daemon_id`	The ID of the service.	true

Table 7. Available options for the ceph_orch_daemon module. (continued)

ceph_orch_daemon	**Description**	**Required**
daemon_type	The type of service.	true

Table 8. Available options for the cephadm_registry_login module

cephadm_registry_login	**Description**	**Required**	**Default**
state	Login or logout of a registry.	false	login
docker	Use docker instead of podman.	false	
registry_url	The URL for custom registry.	false	
registry_username	Username for custom registry.	true when state is login.	
registry_password	Password for custom registry.	true when state is login.	
registry_json	The path to a JSON file. This file must be present on remote hosts prior to running this task. This option is currently not supported.		

Bootstrapping a storage cluster using the cephadm_ansible modules

As a storage administrator, you can bootstrap a storage cluster using the cephadm-ansible modules such as cephadm_bootstrap and cephadm_registry_login playbook.

Prerequisites

- An IP address for the first Ceph Monitor container, which is also the IP address for the first node in the storage cluster.
- Login access to cp.icr.io/cp.
- A minimum of 10 GB of free space for /var/lib/containers/.
- Red Hat Enterprise Linux 8.10 or 9.4 or later with ansible-core bundled into AppStream.
- Installation of the cephadm-ansible package on the Ansible administration node.
- Passwordless SSH is set up on all hosts in the storage cluster.
- Hosts are registered with CDN.

For the latest supported Red Hat Enterprise Linux versions, see Compatibility matrix.

Procedure

1. Log in to the Ansible administration node.
2. Navigate to the /usr/share/cephadm-ansible directory on the Ansible administration node:

Example

```
[ansible@admin ~]$ cd /usr/share/cephadm-ansible
```

3. Create the `hosts` file and add hosts, labels, and monitor IP address of the first host in the storage cluster:

   Syntax

```
sudo vi INVENTORY_FILE

HOST1 labels="[LABEL1, LABEL2]"
HOST2 labels="[LABEL1, LABEL2]"
HOST3 labels="[LABEL1]"

[admin]
ADMIN_HOST monitor_address=MONITOR_IP_ADDRESS labels="[ADMIN_LABEL, LABEL1, LABEL2]"
```

   Example

```
[ansible@admin cephadm-ansible]$ sudo vi hosts

host02 labels="['mon', 'mgr']"
host03 labels="['mon', 'mgr']"
host04 labels="['osd']"
host05 labels="['osd']"
host06 labels="['osd']"

[admin]
host01 monitor_address=10.10.128.68 labels="['_admin', 'mon', 'mgr']"
```

4. Run the preflight playbook:

   Syntax

```
ansible-playbook -i INVENTORY_FILE cephadm-preflight.yml --extra-vars "ceph_origin=ibm"
```

   Example

```
[ansible@admin cephadm-ansible]$ ansible-playbook -i hosts cephadm-preflight.yml --extra-
vars "ceph_origin=ibm"
```

5. Create a playbook to bootstrap your cluster:

   Syntax

```
sudo vi PLAYBOOK_FILENAME.yml

---
- name: NAME_OF_PLAY
  hosts: BOOTSTRAP_HOST
  become: USE_ELEVATED_PRIVILEGES
  gather_facts: GATHER_FACTS_ABOUT_REMOTE_HOSTS
  tasks:
    -name: NAME_OF_TASK
     cephadm_registry_login:
       state: STATE
       registry_url: REGISTRY_URL
       registry_username: REGISTRY_USER_NAME
       registry_password: REGISTRY_PASSWORD

    - name: NAME_OF_TASK
      cephadm_bootstrap:
        mon_ip: "{{ monitor_address }}"
        dashboard_user: DASHBOARD_USER
        dashboard_password: DASHBOARD_PASSWORD
        allow_fqdn_hostname: ALLOW_FQDN_HOSTNAME
        cluster_network: NETWORK_CIDR
```

   Example

```
[ansible@admin cephadm-ansible]$ sudo vi bootstrap.yml
```

```
---
- name: bootstrap the cluster
  hosts: host01
  become: true
  gather_facts: false
  tasks:
    - name: login to registry
      cephadm_registry_login:
        state: login
        registry_url: cp.icr.io/cp
        registry_username: user1
        registry_password: mypassword1

    - name: bootstrap initial cluster
      cephadm_bootstrap:
        mon_ip: "{{ monitor_address }}"
        dashboard_user: mydashboarduser
        dashboard_password: mydashboardpassword
        allow_fqdn_hostname: true
        cluster_network: 10.10.128.0/28
```

6. Run the playbook:

   Syntax

   ```
   ansible-playbook -i INVENTORY_FILE PLAYBOOK_FILENAME.yml -vvv
   ```

   Example

   ```
   ansible@admin cephadm-ansible]$ ansible-playbook -i hosts bootstrap.yml -vvv
   ```

- Review the Ansible output after running the playbook.

Adding or removing hosts using the `ceph_orch_host` module

Add and remove hosts in your storage cluster by using the `ceph_orch_host` module in your Ansible playbook.

Prerequisites

- A running IBM Storage Ceph cluster.
- Register the nodes to the CDN and attach subscriptions.
- Ansible user with sudo and passwordless SSH access to all nodes in the storage cluster.
- Installation of the `cephadm-ansible` package on the Ansible administration node.
- New hosts have the storage cluster's public SSH key. For more information about copying the storage cluster's public SSH keys to new hosts, see "Adding hosts" on page 44.

Procedure

1. Use the following procedure to add new hosts to the cluster:

   a. Log in to the Ansible administration node.

   b. Navigate to the `/usr/share/cephadm-ansible` directory on the Ansible administration node:

      Example

      ```
      [ansible@admin ~]$ cd /usr/share/cephadm-ansible
      ```

   c. Add the new hosts and labels to the Ansible inventory file.

      Syntax

      ```
      sudo vi INVENTORY_FILE

      NEW_HOST1 labels="[LABEL1, LABEL2]"
      NEW_HOST2 labels="[LABEL1, LABEL2]"
      NEW_HOST3 labels="[LABEL1]"
      ```

```
[admin]
ADMIN_HOST monitor_address=MONITOR_IP_ADDRESS labels="[ADMIN_LABEL, LABEL1, LABEL2]"
```

Example

```
[ansible@admin cephadm-ansible]$ sudo vi hosts

host02 labels="['mon', 'mgr']"
host03 labels="['mon', 'mgr']"
host04 labels="['osd']"
host05 labels="['osd']"
host06 labels="['osd']"

[admin]
host01 monitor_address= 10.10.128.68 labels="['_admin', 'mon', 'mgr']"
```

d. Run the preflight playbook with the `--limit` option:

Syntax

```
ansible-playbook -i INVENTORY_FILE cephadm-preflight.yml --extra-vars "ceph_origin=ibm"
--limit NEWHOST
```

Example

```
[ansible@admin cephadm-ansible]$ ansible-playbook -i hosts cephadm-preflight.yml --extra-
vars "ceph_origin=ibm" --limit host02
```

The preflight playbook installs `podman`, `lvm2`, `chronyd`, and `cephadm` on the new host. After installation is complete, `cephadm` resides in the `/usr/sbin/` directory.

e. Create a playbook to add the new hosts to the cluster:

Syntax

```
sudo vi PLAYBOOK_FILENAME.yml

---
- name: PLAY_NAME
  hosts: HOSTS_OR_HOST_GROUPS
  become: USE_ELEVATED_PRIVILEGES
  gather_facts: GATHER_FACTS_ABOUT_REMOTE_HOSTS
  tasks:
    - name: NAME_OF_TASK
      ceph_orch_host:
        name: "{{ ansible_facts[hostname] }}"
        address: "{{ ansible_facts[default_ipv4][address] }}"
        labels: "{{ labels }}"
      delegate_to: HOST_TO_DELEGATE_TASK_TO

    - name: NAME_OF_TASK
      when: inventory_hostname in groups[admin]
      ansible.builtin.shell:
        cmd: CEPH_COMMAND_TO_RUN
      register: REGISTER_NAME

    - name: NAME_OF_TASK
      when: inventory_hostname in groups[admin]
      debug:
        msg: "{{ REGISTER_NAME.stdout }}"
```

Note: By default, Ansible runs all tasks on the host that matches the `hosts` line of your playbook. The `ceph orch` commands must run on the host that contains the admin keyring and the Ceph configuration file. Use the `delegate_to` keyword to specify the admin host in your cluster.

Example

```
[ansible@admin cephadm-ansible]$ sudo vi add-hosts.yml

---
- name: add additional hosts to the cluster
  hosts: all
  become: true
  gather_facts: true
  tasks:
    - name: add hosts to the cluster
      ceph_orch_host:
        name: "{{ ansible_facts['hostname'] }}"
        address: "{{ ansible_facts['default_ipv4']['address'] }}"
        labels: "{{ labels }}"
      delegate_to: host01

    - name: list hosts in the cluster
      when: inventory_hostname in groups['admin']
      ansible.builtin.shell:
        cmd: ceph orch host ls
      register: host_list

    - name: print current list of hosts
      when: inventory_hostname in groups['admin']
      debug:
        msg: "{{ host_list.stdout }}"
```

In this example, the playbook adds the new hosts to the cluster and displays a current list of hosts.

f. Run the playbook to add additional hosts to the cluster:

Syntax

```
ansible-playbook -i INVENTORY_FILE PLAYBOOK_FILENAME.yml
```

Example

```
[ansible@admin cephadm-ansible]$ ansible-playbook -i hosts add-hosts.yml
```

2. Use the following procedure to remove hosts from the cluster:

a. Log in to the Ansible administration node.

b. Navigate to the `/usr/share/cephadm-ansible` directory on the Ansible administration node:

Example

```
[ansible@admin ~]$ cd /usr/share/cephadm-ansible
```

c. Create a playbook to remove a host or hosts from the cluster:

Syntax

```
sudo vi PLAYBOOK_FILENAME.yml

---
- name: NAME_OF_PLAY
  hosts: ADMIN_HOST
  become: USE_ELEVATED_PRIVILEGES
  gather_facts: GATHER_FACTS_ABOUT_REMOTE_HOSTS
  tasks:
    - name: NAME_OF_TASK
      ceph_orch_host:
        name: HOST_TO_REMOVE
        state: STATE

    - name: NAME_OF_TASK
      ceph_orch_host:
        name: HOST_TO_REMOVE
        state: STATE
      retries: NUMBER_OF_RETRIES
      delay: DELAY
      until: CONTINUE_UNTIL
      register: REGISTER_NAME

    - name: NAME_OF_TASK
      ansible.builtin.shell:
        cmd: ceph orch host ls
      register: REGISTER_NAME

    - name: NAME_OF_TASK
        debug:
          msg: "{{ REGISTER_NAME.stdout }}"
```

Example

```
[ansible@admin cephadm-ansible]$ sudo vi remove-hosts.yml

---
- name: remove host
  hosts: host01
  become: true
  gather_facts: true
  tasks:
    - name: drain host07
      ceph_orch_host:
        name: host07
        state: drain

    - name: remove host from the cluster
      ceph_orch_host:
        name: host07
        state: absent
      retries: 20
      delay: 1
      until: result is succeeded
      register: result

     - name: list hosts in the cluster
       ansible.builtin.shell:
         cmd: ceph orch host ls
       register: host_list

     - name: print current list of hosts
       debug:
         msg: "{{ host_list.stdout }}"
```

In this example, the playbook tasks drain all daemons on `host07`, removes the host from the cluster, and displays a current list of hosts.

d. Run the playbook to remove host from the cluster:

Syntax

```
ansible-playbook -i INVENTORY_FILE PLAYBOOK_FILENAME.yml
```

Example

```
[ansible@admin cephadm-ansible]$ ansible-playbook -i hosts remove-hosts.yml
```

Verification

- Review the Ansible task output displaying the current list of hosts in the cluster:

Example

```
TASK [print current hosts]
**********************************************************************************************
********
Friday 24 June 2022  14:52:40 -0400 (0:00:03.365)       0:02:31.702 ***********
ok: [host01] =>
  msg: |-
    HOST    ADDR           LABELS          STATUS
    host01  10.10.128.68   _admin mon mgr
    host02  10.10.128.69   mon mgr
    host03  10.10.128.70   mon mgr
    host04  10.10.128.71   osd
    host05  10.10.128.72   osd
    host06  10.10.128.73   osd
```

Setting configuration options using the `ceph_config` module

As a storage administrator, you can set or get IBM Storage Ceph configuration options using the `ceph_config` module.

Prerequisites

- A running IBM Storage Ceph cluster.
- Ansible user with sudo and passwordless SSH access to all nodes in the storage cluster.
- Installation of the `cephadm-ansible` package on the Ansible administration node.
- The Ansible inventory file contains the cluster and admin hosts. For more information about adding hosts to your storage cluster, see "Adding or removing hosts using the ceph_orch_host module" on page 69.

Procedure

1. Log in to the Ansible administration node.
2. Navigate to the `/usr/share/cephadm-ansible` directory on the Ansible administration node:

   Example

   ```
   [ansible@admin ~]$ cd /usr/share/cephadm-ansible
   ```

3. Create a playbook with configuration changes:

   Syntax

   ```
   sudo vi PLAYBOOK_FILENAME.yml

   ---
   - name: PLAY_NAME
     hosts: ADMIN_HOST
     become: USE_ELEVATED_PRIVILEGES
     gather_facts: GATHER_FACTS_ABOUT_REMOTE_HOSTS
     tasks:
       - name: NAME_OF_TASK
         ceph_config:
           action: GET_OR_SET
           who: DAEMON_TO_SET_CONFIGURATION_TO
           option: CEPH_CONFIGURATION_OPTION
           value: VALUE_OF_PARAMETER_TO_SET

       - name: NAME_OF_TASK
         ceph_config:
           action: GET_OR_SET
           who: DAEMON_TO_SET_CONFIGURATION_TO
           option: CEPH_CONFIGURATION_OPTION
         register: REGISTER_NAME

       - name: NAME_OF_TASK
         debug:
           msg: "MESSAGE_TO_DISPLAY {{ REGISTER_NAME.stdout }}"
   ```

   Example

   ```
   [ansible@admin cephadm-ansible]$ sudo vi change_configuration.yml

   ---
   - name: set pool delete
     hosts: host01
     become: true
     gather_facts: false
     tasks:
       - name: set the allow pool delete option
         ceph_config:
           action: set
           who: mon
           option: mon_allow_pool_delete
           value: true

       - name: get the allow pool delete setting
         ceph_config:
           action: get
           who: mon
           option: mon_allow_pool_delete
         register: verify_mon_allow_pool_delete
   ```

```
     - name: print current mon_allow_pool_delete setting
       debug:
         msg: "the value of 'mon_allow_pool_delete' is
{{ verify_mon_allow_pool_delete.stdout }}"
```

In this example, the playbook first sets the `mon_allow_pool_delete` option to `false`. The playbook then gets the current `mon_allow_pool_delete` setting and displays the value in the Ansible output.

4. Run the playbook:

   Syntax

   ```
   ansible-playbook -i INVENTORY_FILE _PLAYBOOK_FILENAME.yml
   ```

   Example

   ```
   [ansible@admin cephadm-ansible]$ ansible-playbook -i hosts change_configuration.yml
   ```

Verification

- Review the output from the playbook tasks.

  Example

```
TASK [print current mon_allow_pool_delete setting]
*************************************************************
Wednesday 29 June 2022  13:51:41 -0400 (0:00:05.523)       0:00:17.953 ********
ok: [host01] =>
  msg: the value of 'mon_allow_pool_delete' is true
```

Reference

- For more information, see Configuring.

Applying a service specification using the `ceph_orch_apply` module

As a storage administrator, you can apply service specifications to your storage cluster using the `ceph_orch_apply` module in your Ansible playbooks. A service specification is a data structure to specify the service attributes and configuration settings that is used to deploy the Ceph service. You can use a service specification to deploy Ceph service types like `mon`, `crash`, `mds`, `mgr`, `osd`, `rdb`, or `rbd-mirror`.

Prerequisites

- A running IBM Storage Ceph cluster.
- Ansible user with sudo and passwordless SSH access to all nodes in the storage cluster.
- Installation of the `cephadm-ansible` package on the Ansible administration node.
- The Ansible inventory file contains the cluster and admin hosts. For more information about adding hosts to your storage cluster, see "Adding or removing hosts using the ceph_orch_host module" on page 69.

Procedure

1. Log in to the Ansible administration node.
2. Navigate to the `/usr/share/cephadm-ansible` directory on the Ansible administration node:

   Example

   ```
   [ansible@admin ~]$ cd /usr/share/cephadm-ansible
   ```

3. Create a playbook with the service specifications:

Syntax

```
sudo vi PLAYBOOK_FILENAME.yml

---
- name: PLAY_NAME
  hosts: HOSTS_OR_HOST_GROUPS
  become: USE_ELEVATED_PRIVILEGES
  gather_facts: GATHER_FACTS_ABOUT_REMOTE_HOSTS
  tasks:
    - name: NAME_OF_TASK
      ceph_orch_apply:
        spec: |
          service_type: SERVICE_TYPE
          service_id: UNIQUE_NAME_OF_SERVICE
          placement:
            host_pattern: HOST_PATTERN_TO_SELECT_HOSTS
            label: LABEL
          spec:
            SPECIFICATION_OPTIONS:
```

Example

```
[ansible@admin cephadm-ansible]$ sudo vi deploy_osd_service.yml

---
- name: deploy osd service
  hosts: host01
  become: true
  gather_facts: true
  tasks:
    - name: apply osd spec
      ceph_orch_apply:
        spec: |
          service_type: osd
          service_id: osd
          placement:
            host_pattern: '*'
            label: osd
          spec:
            data_devices:
              all: true
```

In this example, the playbook deploys the Ceph OSD service on all hosts with the label `osd`.

4. Run the playbook:

   Syntax

```
ansible-playbook -i INVENTORY_FILE _PLAYBOOK_FILENAME.yml
```

   Example

```
[ansible@admin cephadm-ansible]$ ansible-playbook -i hosts deploy_osd_service.yml
```

Verification

- Review the output from the playbook tasks.

Managing Ceph daemon states using the `ceph_orch_daemon` module

Start, stop, and restart Ceph daemons on hosts using the `ceph_orch_daemon` module in your Ansible playbooks.

Prerequisites

- A running IBM Storage Ceph cluster.
- Ansible user with sudo and passwordless SSH access to all nodes in the storage cluster.
- Installation of the `cephadm-ansible` package on the Ansible administration node.

- The Ansible inventory file contains the cluster and admin hosts. For more information about adding hosts to your storage cluster, see "Adding or removing hosts using the ceph_orch_host module" on page 69.

Procedure

1. Log in to the Ansible administration node.
2. Navigate to the `/usr/share/cephadm-ansible` directory on the Ansible administration node:

   Example

   ```
   [ansible@admin ~]$ cd /usr/share/cephadm-ansible
   ```

3. Create a playbook with daemon state changes:

   Syntax

   ```
   sudo vi PLAYBOOK_FILENAME.yml

   ---
   - name: PLAY_NAME
     hosts: ADMIN_HOST
     become: USE_ELEVATED_PRIVILEGES
     gather_facts: GATHER_FACTS_ABOUT_REMOTE_HOSTS
     tasks:
       - name: NAME_OF_TASK
         ceph_orch_daemon:
           state: STATE_OF_SERVICE
           daemon_id: DAEMON_ID
           daemon_type: TYPE_OF_SERVICE
   ```

   Example

   ```
   [ansible@admin cephadm-ansible]$ sudo vi restart_services.yml

   ---
   - name: start and stop services
     hosts: host01
     become: true
     gather_facts: false
     tasks:
       - name: start osd.0
         ceph_orch_daemon:
           state: started
           daemon_id: 0
           daemon_type: osd

       - name: stop mon.host02
         ceph_orch_daemon:
           state: stopped
           daemon_id: host02
           daemon_type: mon
   ```

   In this example, the playbook starts the OSD with an ID of `0` and stops a Ceph Monitor with an id of `host02`.

4. Run the playbook:

   Syntax

   ```
   ansible-playbook -i INVENTORY_FILE _PLAYBOOK_FILENAME.yml
   ```

   Example

   ```
   [ansible@admin cephadm-ansible]$ ansible-playbook -i hosts restart_services.yml
   ```

Verification

- Review the output from the playbook tasks.

Comparison between Ceph Ansible and Cephadm

Understand the differences between Cephadm and Ceph-Ansible playbooks for the containerized deployment of the storage cluster.

The tables compare Cephadm with Ceph-Ansible playbooks for managing the containerized deployment of a Ceph cluster for day one and day two operations.

Table 9. Day one operations

Description	**Ceph-Ansible**	**Cephadm**
Installation of the IBM Storage Ceph cluster	Run the `site-container.yml` playbook.	Run `cephadm bootstrap` command to bootstrap the cluster on the admin node.
Addition of hosts	Use the Ceph Ansible inventory.	Run `ceph orch add host` HOST_NAME to add hosts to the cluster.
Gathering Ceph logs	Run the `gather ceph logs` playbook.	Run the `journalctl` command.
Addition of monitors	Run the `add-mon.yml` playbook.	Run the `ceph orch apply mon` command.
Addition of managers	Run the `site-container.yml` playbook.	Run the `ceph orch apply mgr` command.
Addition of OSDs	Run the `add-osd.yml` playbook.	Run the `ceph orch apply osd` command to add OSDs on all available devices or on specific hosts.
Addition of OSDs on specific devices	Select the `devices` in the `osd.yml` file and then run the `add-osd.yml` playbook.	Select the `paths` filter under the `data_devices` in the `osd.yml` file and then run `ceph orch apply -i FILE_NAME.yml` command.
Addition of MDS	Run the `site-container.yml` playbook.	Run the `ceph orch apply` FILESYSTEM_NAME command to add MDS.
Addition of Ceph Object Gateway	Run the `site-container.yml` playbook.	Run the `ceph orch apply rgw` commands to add Ceph Object Gateway.

Table 10. Day two operations

Description	**Ceph-Ansible**	**Cephadm**
Removing hosts	Use the Ansible inventory.	Run `ceph orch host rm` HOST_NAME to remove the hosts.
Removing monitors	Run the `shrink-mon.yml` playbook.	Run `ceph orch apply mon` to redeploy other monitors.

Table 10. Day two operations (continued)

Description	**Ceph-Ansible**	**Cephadm**
Removing managers	Run the `shrink-mon.yml` playbook.	Run `ceph orch apply mgr` to redeploy other managers.
Removing OSDs	Run the `shrink-osd.yml` playbook.	Run `ceph orch osd rm OSD_ID` to remove the OSDs.
Removing MDS	Run the `shrink-mds.yml` playbook.	
Deployment of Ceph Object Gateway	Run the `site-container.yml` playbook.	Run `ceph orch apply rgw _SERVICE_NAME_` to deploy Ceph Object Gateway service.
Removing Ceph Object Gateway	Run the `shrink-rgw.yml` playbook.	Run `ceph orch rm SERVICE_NAME` to remove the specific service.
Block device mirroring	Run the `site-container.yml` playbook.	Run `ceph orch apply rbd-mirror` command.
Minor version upgrade of IBM Storage Ceph	Run the `infrastructure-playbooks/rolling_update.yml` playbook.	Run `ceph orch upgrade start` command.
Deployment of monitoring stack	Edit the `all.yml` file during installation.	Run the `ceph orch apply -i FILE.yml` after specifying the services.

What to do next? Day 2

As a storage administrator, once you have installed and configured IBM Storage Ceph, you are ready to perform "Day Two" operations for your storage cluster. These operations include adding metadata servers (MDS) and Ceph Object Gateways (RGW), and configuring services.

For more information about how to use the `cephadm` orchestrator to perform "Day Two" operations, see Operations.

To deploy, configure, and administer the Ceph Object Gateway on "Day Two" operations, see Ceph Object Gateway.