Juniper Networks | Engineering Simplicity

BBE Cloudsetup Installation Guide

Published
2024-07-24

Juniper Networks, Inc.
1133 Innovation Way
Sunnyvale, California 94089
USA
408-745-2000
www.juniper.net

BBE Cloudsetup Installation Guide

The information in this document is current as of the date on the title page.

YEAR 2000 NOTICE

Juniper Networks hardware and software products are Year 2000 compliant. Junos OS has no known time-related limitations through the year 2038. However, the NTP application is known to have some difficulty in the year 2036.

END USER LICENSE AGREEMENT

The Juniper Networks product that is the subject of this technical documentation consists of (or is intended for use with) Juniper Networks software. Use of such software is subject to the terms and conditions of the End User License Agreement ("EULA") posted at https://support.juniper.net/support/eula/. By downloading, installing or using such software, you agree to the terms and conditions of that EULA.

Table of Contents

About This Guide

Use this guide to install and configure BBE Cloudsetup.

1
CHAPTER

BBE Cloudsetup Installation

Install BBE Cloudsetup

SUMMARY

This section describes installation procedures and system requirements for BBE Cloudsetup.

IN THIS SECTION

BBE Cloudsetup Installation Overview

BBE Cloudsetup facility constructs a Kubernetes cluster from a set of dedicated Ubuntu hosts. You can use the BBE Cloudsetup facility to create multiple Kubernetes clusters managed from a single jump host.

We recommend that you create a cluster with at least three control plane nodes. BBE Cloudsetup control plane nodes are hybrid nodes. Meaning, that they take on the roles of a control plane, etcd (Kubernetes state database) and worker nodes. Additional nodes can be added to the cluster that are pure worker nodes (single worker role).

After you set up the cluster (using BBE Cloudsetup), you can then install the desired Juniper BBE cloud applications in the cluster.

You can deploy BBE Cloudsetup on any hardware that meets the installation requirements. The following sections describe:

- BBE Cloudsetup hardware and software requirements
- How to install BBE Cloudsetup

BBE Cloudsetup Installation Requirements

Before you begin installing BBE Cloudsetup, make sure you have the following:

- A juniper.net user account with permissions to download the BBE Cloudsetup software package.
- A Linux host (jump host) running Ubuntu 22.04 LTS (required) for running the `bbecloudsetup` utility. The jump host must meet the following system requirements:
  - CPU cores—2
  - RAM—8 GB
  - Disk space—128 GB of free disk storage
- If you are performing an offline installation, you will need a Linux host system where you will move the downloaded BBE Cloudsetup images package to. The host can be the same as the jump host, but it must meet the following requirements:
  - Docker must be installed.
  - Must have access to an existing container registry that can be used as the system registry for the cluster. You must be logged into the registry through `docker login` *`registry_address`*.

    NOTE: The system registry should exist and be available for the cluster to use for the lifespan of the cluster. The cluster will need to pull system infrastructure images from the host when nodes are added or restarted.
- Nodes that will be used as part of the cluster (either virtual or physical machines). A node is a Linux system running Ubuntu 22.04 LTS (required) that has a management address and a domain name.
- The cluster nodes and the jump host must be able to access each other through SSH.
- All cluster nodes must have a user account with sudo access.
- We recommend that all cluster nodes meet the following minimum system requirements:
  - CPU cores—8 (hyperthreading preferred)
  - RAM—64 GB
  - Disk space—512 GB of free disk storage (solid state recommended)

    We recommend that you use the storage space to partition your disk accordingly:
    - 128 GB to the root (/) partition for the operating system

- 128 GB to **/var/lib/docker** for the Docker cache
- 256 GB to **/mnt/longhorn** for the application data. This is the default location, you can specify a different location during configuration.

- Node access and authentication. You must have root-level SSH access, using key-based authentication, to all nodes.

  NOTE: You should not run BBE cloudsetup as root. You should configure BBE Cloudsetup to connect as a user on each node with sudo permissions and a Bash login shell.

- You must create your own partitions on all the host systems. On the jump host, you should create a Docker partition. On the cluster nodes you must create both a Docker partition and a Longhorn (data) partition.

NOTE: Be aware that BBE Cloudsetup may modify elements of your host system's configurations. This is required so that it can become a cluster node.

Install BBE Cloudsetup

SUMMARY

Use this procedure to install BBE Cloudsetup.

Before you begin, make sure you complete the following:

- Confirm that you have met all the requirements for the BBE Cloudsetup installation (see "BBE Cloudsetup Installation Requirements" on page 3).
- You must create SSH keys before installing BBE Cloudsetup. The SSH keys must be copied to each cluster member with the exact hostname used in the configuration file. See "Configure SSH" on page 14.

Install the BBE Cloudsetup utility.

1. Download the BBE Cloudsetup software package from the Juniper Networks software download page, and save it to the jump host system.

   BBE Cloudsetup is available as a compressed tarball image (**.tgz**). The filename includes the release number as part of the name. The release number has the format: <Major>.<Minor>.<Maintenance>

   - *major* is the main release number of the product.
   - *minor* is the minor release number of the product.
   - *maintainance* is the revision number.

2. Unpack the BBE Cloudsetup tarball (**.tgz**) file on the jump host by entering:

   ```
   $ tar -zxf bbecloudsetup-m.m.m.tgz
   ```

   This unpacks the tarball into a directory named bbecloudsetup **bbecloudsetup**. If this directory already exists, remove it before you unpack the tarball to avoid loading any previously installed **bbecloudsetup** files.

3. Make a uniquely named copy of the **exampleconfig.yaml** file in the **bbecloudsetup** directory to preserve its values. We recommended that you follow the format ***clustername*-config.yaml** (for example, **mycluster-config.yaml**). Untaring another BBE Cloudsetup package results in the **exampleconfig.yaml** getting overwritten, as a new **exampleconfig.yaml** file is created. Creating a uniquely named copy helps in restoring the information contained in the **exampleconfig.yaml** file.

   NOTE: The supplied **exampleconfig.yaml** file (located in the **bbecloudsetup** directory) is used to define the pararmeters of the Kubernetes cluster. Untaring another BBE Cloudsetup package results in the existing **exampleconfig.yaml** file getting overwritten, as a new **exampleconfig.yaml** file is created during untaring. Creating a uniquely named copy helps you in restoring the information contained in the **exampleconfig.yaml** file.

4. If you are installing BBE Cloudsetup in an environment with Internet access, go to Step5.

   If you are installing BBE Cloudsetup in an air gapped environment (no Internet access), perform the following and then proceed to Step 5:

   a. Identify an existing separate private system registry (or create one), to serve as the default registry from where the cluster nodes can pull the system infrastructure images.

      For example, to create a new Docker registry, run the following command:

      ```
      $ docker run -d -p host.example.com:port:5000 --name registry name registry:2.8.1
      ```

- *host.example.com*—The qualified domain name of the host for the registry. This node cannot be a cluster member (as this creates a cyclical dependency), but it should have contact with the cluster. The jump host can serve as the host for the system registry.
- *port*—The port number of the host on which the registry is started on. If you are using the default registry, then the port number should be 5000. Otherwise, any available port number can be used.

  For further information on deploying a registry server, see: https://distribution.github.io/distribution/about/deploying/.

b. Download the BBE Cloudsetup images software package from the Juniper Networks software download page, and save it to a separate host system that has access to an existing container registry that can be used as the system registry for the cluster (for all host system requirements see "BBE Cloudsetup Installation Requirements" on page 3).

c. Unpack the BBE Cloudsetup images tarball (**.tgz**) file on your host system by entering:

```
$ tar -zxf bbecloudsetup-images-m.m.m.tgz
```

d. Enter the **bbecloudsetup-images** directory, using the following command:

```
cd bbecloudsetup-images
```

e. Confirm you are able to execute Docker commands and are logged into the target registry by entering the following command:

```
[sudo] docker login registry-url
```

f. Run the following command to load and tag the images with the target registry and then push the images:

```
[sudo] ./loadimages registry-url
```

The `loadimages` script takes exactly one positional argument (either the registry URL, **help**, or **version**). The registry URL should include the port (if needed) and should not include **http(s)://** or a trailing forward slash (**/**).

When completed, a **Done** message appears.

> **NOTE**: The `loadimages` script must remain located with both the **manifest** file and the **bcs_images.tar** package inside the **bbecloudsetup-images** directory. If any of these objects are moved out of that location, the script produces an error message stating which object was not found.

g. On the jump host in the renamed **.yaml** configuration file (***clustername*-config.yaml**), make sure that the following parameters are filled out:

- `systemRegistry: - #registry.example.com:port`

  In the format *host.example.com*:*port-number*

- `systemRegistryUser: - #user`

- `systemRegistryPassword: - #secret`

5. Fill out the ***clustername*-config.yaml** file with the parameters of the Kubernetes cluster. Table 1 on page 10 describes the information that you need to enter into the configuration file.
6. Generate a new SSH key for BBE Cloudsetup by running the following command:

```
$ ssh-keygen -t rsa
```

7. Copy the SSH key to the cluster. Using the fully qualified domain name of the cluster in place of *clustername*, run the following command:

```
$ ssh-copy-id bcs@clustername
```

The password that you are prompted for is the password for the user on the remote host to which the key is being copied.

8. Stop any orphaned SSH agents by running the following command:

```
$ pkill ssh-agent
```

9. Start a new SSH agent by running the following command:

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

10. Add the SSH key to the agent by running the following command:

```
$ ssh-add
```

11. Run the BBE Cloudsetup executable. The executable is located in the **bbecloudsetup** directory.

```
sudo -E ./bbecloudsetup <flags> install -template </bbecloudsetup/clustername-
config.yaml><flags>
```

12. Verify the installation.
    - Observe that the BBE Cloudsetup installation has completed and no errors have occurred
    - Run the `kubectl get nodes` command. All nodes should report their status as *Ready*.

```
xxxyyyy.testlab.juniper.net Ready controlplane,etcd,worker 194d v1.24.4
xxxyyyy.testlab.juniper.net Ready controlplane,etcd,worker 194d v1.24.4
xxxyyyy.testlab.juniper.net Ready controlplane,etcd,worker 194d v1.24.4
```

    - Run the `kubectl get pods -A` command. All pods in the cluster should report their status as either *Running* or *Completed*.

```
ngress-nginx nginx-ingress-controller-b7dxg 1/1 Running 7 (8d ago) 194d
ingress-nginx nginx-ingress-controller-kp7rl 1/1 Running 5 (8d ago) 194d
ingress-nginx nginx-ingress-controller-mpqmm 1/1 Running 14 (8d ago) 194d
jnpr-registry jnpr-registry-khcws 1/1 Running 0 7d23h
jnpr-registry jnpr-registry-qpt5k 1/1 Running 0 7d23h
jnpr-registry jnpr-registry-xh7nq 1/1 Running 0 7d23h
kube-system coredns-59499769fb-685q9 1/1 Running 5 (8d ago) 194d
kube-system coredns-59499769fb-z225q 1/1 Running 4 (8d ago) 194d
kube-system coredns-autoscaler-67cbd4599c-kfxzd 1/1 Running 5 (8d ago) 194d
kube-system kube-flannel-56vrr 2/2 Running 10 (8d ago) 194d
kube-system kube-flannel-6pns5 2/2 Running 28 (8d ago) 194d
kube-system kube-flannel-jjzxh 2/2 Running 8 (8d ago) 194d
kube-system metrics-server-585b7cc746-xkgjl 1/1 Running 15 (8d ago) 194d
kube-system rke-coredns-addon-deploy-job-r7wch 0/1 Completed 0 194d
kube-system rke-ingress-controller-deploy-job-hxbd8 0/1 Completed 0 194d
kube-system rke-metrics-addon-deploy-job-xmqsc 0/1 Completed 0 194d
kube-system rke-network-plugin-deploy-job-5q9f7 0/1 Completed 0 194d
longhorn-system csi-attacher-5b6cc49f97-9wndg 1/1 Running 6 (8d ago) 57d
longhorn-system csi-attacher-5b6cc49f97-cbdrv 1/1 Running 7 (7d23h ago) 57d
```

```
longhorn-system csi-attacher-5b6cc49f97-lcnr9 1/1 Running 18 (8d ago) 132d
longhorn-system csi-provisioner-5d8dd96b57-rbs7t 1/1 Running 5 (7d23h ago) 57d
longhorn-system csi-provisioner-5d8dd96b57-vp2vk 1/1 Running 13 (8d ago) 132d
longhorn-system csi-provisioner-5d8dd96b57-zhh2h 1/1 Running 17 (8d ago) 132d
longhorn-system csi-resizer-7c5bb5fd65-bg48m 1/1 Running 14 (8d ago) 132d
longhorn-system csi-resizer-7c5bb5fd65-cwtzn 1/1 Running 6 (7d23h ago) 57d
longhorn-system csi-resizer-7c5bb5fd65-qrdm9 1/1 Running 17 (8d ago) 132d
longhorn-system csi-snapshotter-5586bc7c79-7zl2m 1/1 Running 15 (8d ago) 132d
longhorn-system csi-snapshotter-5586bc7c79-fb5gn 1/1 Running 3 (8d ago) 57d
longhorn-system csi-snapshotter-5586bc7c79-rdm6g 1/1 Running 4 (8d ago) 132d
longhorn-system engine-image-ei-9bf563e8-fnkdt 1/1 Running 4 (8d ago) 194d
longhorn-system engine-image-ei-9bf563e8-hzg7w 1/1 Running 5 (8d ago) 194d
longhorn-system engine-image-ei-9bf563e8-k27f4 1/1 Running 14 (8d ago) 194d
longhorn-system instance-manager-e-16643925 1/1 Running 0 8d
longhorn-system instance-manager-e-f4810551 1/1 Running 0 8d
longhorn-system instance-manager-e-f842b1a7 1/1 Running 0 8d
longhorn-system instance-manager-r-d8844e77 1/1 Running 0 8d
longhorn-system instance-manager-r-dff34848 1/1 Running 0 8d
longhorn-system instance-manager-r-f1ad6f4a 1/1 Running 0 8d
longhorn-system longhorn-csi-plugin-59hjv 2/2 Running 9 (8d ago) 194d
longhorn-system longhorn-csi-plugin-6flcd 2/2 Running 13 (8d ago) 194d
longhorn-system longhorn-csi-plugin-k89vc 2/2 Running 31 (8d ago) 194d
longhorn-system longhorn-driver-deployer-6ffcd48d5d-55lhf 1/1 Running 4 (8d ago) 57d
longhorn-system longhorn-manager-8djqj 1/1 Running 14 (8d ago) 194d
longhorn-system longhorn-manager-khx6m 1/1 Running 5 (8d ago) 194d
longhorn-system longhorn-manager-kpkh7 1/1 Running 4 (8d ago) 194d
longhorn-system longhorn-ui-6ddb78495f-dtcv6 1/1 Running 4 (8d ago) 132d
longhorn-system share-manager-pvc-289a1a2e-f79a-42af-9324-49a8c00c7c9a 1/1 Running 0 8d
longhorn-system share-manager-pvc-39921efe-87ce-47ac-864d-076d6fb4b069 1/1 Running 0 170m
longhorn-system share-manager-pvc-962745a8-e08f-4726-84c8-45f151a432d7 1/1 Running 0 170m
longhorn-system share-manager-pvc-d9c9d08c-08c5-44e9-91fa-bea34240bebe 1/1 Running 0 170m
metallb-system metallb-controller-7f6b8b7fdd-ljfck 1/1 Running 4 (8d ago) 132d
metallb-system metallb-speaker-h6t9q 1/1 Running 15 (8d ago) 194d
metallb-system metallb-speaker-jr76q 1/1 Running 5 (8d ago) 194d
metallb-system metallb-speaker-whz4w 1/1 Running 6 (8d ago) 194d
```

13. Stop the SSH agent.

```
$ eval $(ssh-agent -k)
```

Configuration File

This section describes the fields in the configuration file (**exampleconfig.yaml**).

Table 1: BBE Cloudsetup Configuration File Fields

Field Name	Description
Cluster name	Enter the name of the cluster.
NTP server	Enter the name of the NTP server that is used to synchronize the time across all nodes in the cluster.
List of nodes in the cluster • **Controlplane node details**. These nodes can be control plane nodes, etcd nodes, and worker nodes. • **Worker node details**	Enter the following information for each node. • **hostname**—Name of the node. • **user**—User name. • **sudoPassword**—Password for the sudo command on the node. • **role**—Either controlplane or worker. • **port**—SSH port number.

Table 1: BBE Cloudsetup Configuration File Fields *(Continued)*

Field Name	Description

General system configuration
 	
A registry is a central resource for pulling container images for cluster workloads. The registry may be a preexisting registry that the cluster nodes have access to, or you can install the registry as part of the cluster.
 	
NOTE: The default action is to install and set up a registry as a pod in the cluster. You can enter an alternative address if you want to host your own registry.
	Enter the following system information:
	
	• **installRegistry**—Set to **true** if you want to install a private registry on the cluster.
	
	• **registrySize**—Size of the persistent volume for the registry in GB.
	
	• **registryPort**—Port number of the registry.
	
	• **address**—The DNS name of the Keepalived VIP address (or the address itself if no DNS is available).
	
	• **user**—Username for the registry.
	
	• **password**—Password for the registry.
	
	• **generateCerts**—By default this field is set to **true** and self-signed certificates for the registry are generated automatically. If this field is disabled, you will need to provide the paths to the key and the certificates to be used. You can do this, either in the configuration file or during the installation process.
	
	  • **key**—Path to the registration key.
	
	  • **cert**—Path to the registration certificate.
	
	  • **caCert**—Path to the registration CA certificate.
	
	• If the cluster can not pull system images from DockerHub, the images will need to be hosted in a separate private registry. This registry will be configured as the default registry for the cluster to pull images from. Enter the following information:
	
	  • **systemRegistry**—Address for the registry.
	
	  • **systemRegistryUser**—Username for the registry.
	
	  • **systemRegistryPassword**—Password for the registry.

Table 1: BBE Cloudsetup Configuration File Fields *(Continued)*

Field Name	Description
Load balancer configuration metalLB A L2 network load balancer provides external (outside the cluster) access to application load balancing services. It is responsible for assigning external IP addresses to load balancing services when they are created as part of application startup. The number of addresses that you configure in the Load Balancer's address pool is dependent on the applications you intend to run: • APM requires at least one IP address for access to APMi. • BBE Event Collection and Visualization may optionally require an external address. If BBE Event Collection and Visualization is configured to analyze syslogs from elements that are not part of the cluster (for example, a BNG User Plane), then it requires an external address. • BNG CUPS Controller requires two addresses. The subnet for the addresses must be reachable from all the cluster nodes and the remote devices that need access to the application's services.	Enter the following information: • **install**—Set to **true**, to install MetalLB for load balancing. • **addresses**—Enter a list of IP addresses for the Load Balancer's address pool. Addresses in the list can be entered as a prefix (for example, 10.0.0.0/24), as individual addresses (for example 10.0.1.2/32), as a range of addresses (for example, 10.1.2.3-10.1.2.5), or as a combination of any of these options.

Table 1: BBE Cloudsetup Configuration File Fields *(Continued)*

Field Name	Description

Network configuration
 	
It may be desirable to segregate intracluster network traffic (used to maintain and manage the cluster) from network traffic (used to communicate with external systems). In a multi-node cluster, a virtual IP address should be assigned to the cluster so that the cluster control plane can be addressed as one system.
	network—Enter the following network information:
	
	NOTE: All nodes in the cluster must have the same interfaces available.
	
	• **cniInterface**—Network interface for the cluster network. This interface is also used as the keepalived interface, therefore must be defined.
	
	• **sans** (Subject Alternative Name)—A list of hostnames (IP addresses) to add to the SAN Kubernetes certificate as *allowed* names where to contact the Kubernetes API server.
	  • controlplane—Subject Alternative Name for TLS certificates.
	  • worker.example.com—Subject Alternative Name for TLS certificates.
	
	• **keepalived**—Must be enabled.
	  • **install**—Enter **true** to install Keepalived for virtual IP configurations.
	  • **vip**—Virtual IP address for the cluster
	  • **vrid**—Virtual Router ID (1 through 255) that keepalived uses for the cluster. This value should be unique for each cluster and each network. By default if no value is given, BBE Cloudsetup uses the last octet of the keepalived address. So, if you know that the last octet of the keepalived address is already a Virtual Router ID used by another cluster, make sure to set this **vrid** to a unique value.

Table 1: BBE Cloudsetup Configuration File Fields *(Continued)*

Field Name	Description

Storage configuration	Enter the following information to configure storage:
	• **installLonghorn**—Enter **true** to install Longhorn for storage.
	• **storageClass**—The storage class name for Longhorn.
	• **Partitions**—Enter the following partition information (Docker and Longhorn partitions):
	  • **name**—Name of partition.
	  • **createPartition**—Do not use.
	  • **devicePath**—Path to the block device for the volume.
	  • **location**—Mount point for the volume.
	  • **partitionSize**—Size (in GB) of the partition for the volume.

Configure SSH

Configure SSH password-less access from the jump host to the control plane and worker nodes.

1. Create an SSH key.

```
ssh-keygen -t rsa
```

   Use the default path and enter a password.

2. Copy the key to the control plane and worker nodes. For example:

```
ssh-copy-id <user-id>@<host address>
```

   - *<user-id>*—The user ID that BBE Cloudesetup will use when connecting to the nodes.
   - *<host address>*—The IP address of the node that BBE Cloudesetup will connect to.

   Repeat the above procedure for each node in the cluster.

3. Start the SSH agent.

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

4. Add all known keys on the system to the agent. If this is a fresh system, there will only be the one key created. If there are other keys, you can either add them all or you can specify the path to the key to load as an argument. Enter the key passphrase as prompted.

```
ssh-add
```

Upgrade BBE Cloudsetup

SUMMARY

Use this procedure to upgrade BBE Cloudsetup.

1. On the jump host system, back up your current **bbecloudsetup** directory containing your current configuration file (renamed **exampleconfig.yaml** file) to a directory with a different name.
2. Download the BBE Cloudsetup software package from the Juniper Networks software download page, and save it to your jump host server.

   BBE Cloudsetup is available as a compressed tarball image (**.tgz**). The filename includes the release number as part of the name. The release number has the format: <Major>.<Minor>.<Maintenance>

   - *major* is the main release number of the product.
   - *minor* is the minor release number of the product.
   - *maintainance* is the revision number.
3. Unpack the BBE Cloudsetup tarball (**.tgz**) file on the jump host by entering:

```
$ tar -zxf bbecloudsetup-m.m.m.tgz
```

   The **bbecloudsetup** directory will be populated with new files.
4. Compare your old configuration file to the new configuration file, and make any desired changes to the file. See Table 1 on page 10.

5. Once you have the configuration file (located in the **bbecloudsetup** directory) ready, run the BBE Cloudsetup executable. The executable is located in the **bbecloudsetup** directory.

```
sudo -E./bbecloudsetup <flags> install -template </bbecloudsetup/exampleconfig.yaml><flags>
```

   If you did not enter the passwords for the hosts (sudo password) or the registry (user password) in the configuration file, you will be prompted for them during installation.

6. Verify the installation.

   - Observe that the BBE Cloudsetup installation has completed and no errors have occurred
   - Run the `kubectl get nodes` command. All nodes should report their status as *Ready*.

```
xxxyyyy.testlab.juniper.net Ready controlplane,etcd,worker 194d v1.24.4
xxxyyyy.testlab.juniper.net Ready controlplane,etcd,worker 194d v1.24.4
xxxyyyy.testlab.juniper.net Ready controlplane,etcd,worker 194d v1.24.4
```

   - Run the `kubectl get pods -A` command. All pods in the cluster should report their status as either *Running* or *Completed*.

```
ngress-nginx nginx-ingress-controller-b7dxg 1/1 Running 7 (8d ago) 194d
ingress-nginx nginx-ingress-controller-kp7rl 1/1 Running 5 (8d ago) 194d
ingress-nginx nginx-ingress-controller-mpqmm 1/1 Running 14 (8d ago) 194d
jnpr-registry jnpr-registry-khcws 1/1 Running 0 7d23h
jnpr-registry jnpr-registry-qpt5k 1/1 Running 0 7d23h
jnpr-registry jnpr-registry-xh7nq 1/1 Running 0 7d23h
kube-system coredns-59499769fb-685q9 1/1 Running 5 (8d ago) 194d
kube-system coredns-59499769fb-z225q 1/1 Running 4 (8d ago) 194d
kube-system coredns-autoscaler-67cbd4599c-kfxzd 1/1 Running 5 (8d ago) 194d
kube-system kube-flannel-56vrr 2/2 Running 10 (8d ago) 194d
kube-system kube-flannel-6pns5 2/2 Running 28 (8d ago) 194d
kube-system kube-flannel-jjzxh 2/2 Running 8 (8d ago) 194d
kube-system metrics-server-585b7cc746-xkgjl 1/1 Running 15 (8d ago) 194d
kube-system rke-coredns-addon-deploy-job-r7wch 0/1 Completed 0 194d
kube-system rke-ingress-controller-deploy-job-hxbd8 0/1 Completed 0 194d
kube-system rke-metrics-addon-deploy-job-xmqsc 0/1 Completed 0 194d
kube-system rke-network-plugin-deploy-job-5q9f7 0/1 Completed 0 194d
longhorn-system csi-attacher-5b6cc49f97-9wndg 1/1 Running 6 (8d ago) 57d
longhorn-system csi-attacher-5b6cc49f97-cbdrv 1/1 Running 7 (7d23h ago) 57d
longhorn-system csi-attacher-5b6cc49f97-lcnr9 1/1 Running 18 (8d ago) 132d
longhorn-system csi-provisioner-5d8dd96b57-rbs7t 1/1 Running 5 (7d23h ago) 57d
```

```
longhorn-system csi-provisioner-5d8dd96b57-vp2vk 1/1 Running 13 (8d ago) 132d
longhorn-system csi-provisioner-5d8dd96b57-zhh2h 1/1 Running 17 (8d ago) 132d
longhorn-system csi-resizer-7c5bb5fd65-bg48m 1/1 Running 14 (8d ago) 132d
longhorn-system csi-resizer-7c5bb5fd65-cwtzn 1/1 Running 6 (7d23h ago) 57d
longhorn-system csi-resizer-7c5bb5fd65-qrdm9 1/1 Running 17 (8d ago) 132d
longhorn-system csi-snapshotter-5586bc7c79-7zl2m 1/1 Running 15 (8d ago) 132d
longhorn-system csi-snapshotter-5586bc7c79-fb5gn 1/1 Running 3 (8d ago) 57d
longhorn-system csi-snapshotter-5586bc7c79-rdm6g 1/1 Running 4 (8d ago) 132d
longhorn-system engine-image-ei-9bf563e8-fnkdt 1/1 Running 4 (8d ago) 194d
longhorn-system engine-image-ei-9bf563e8-hzg7w 1/1 Running 5 (8d ago) 194d
longhorn-system engine-image-ei-9bf563e8-k27f4 1/1 Running 14 (8d ago) 194d
longhorn-system instance-manager-e-16643925 1/1 Running 0 8d
longhorn-system instance-manager-e-f4810551 1/1 Running 0 8d
longhorn-system instance-manager-e-f842b1a7 1/1 Running 0 8d
longhorn-system instance-manager-r-d8844e77 1/1 Running 0 8d
longhorn-system instance-manager-r-dff34848 1/1 Running 0 8d
longhorn-system instance-manager-r-f1ad6f4a 1/1 Running 0 8d
longhorn-system longhorn-csi-plugin-59hjv 2/2 Running 9 (8d ago) 194d
longhorn-system longhorn-csi-plugin-6flcd 2/2 Running 13 (8d ago) 194d
longhorn-system longhorn-csi-plugin-k89vc 2/2 Running 31 (8d ago) 194d
longhorn-system longhorn-driver-deployer-6ffcd48d5d-55lhf 1/1 Running 4 (8d ago) 57d
longhorn-system longhorn-manager-8djqj 1/1 Running 14 (8d ago) 194d
longhorn-system longhorn-manager-khx6m 1/1 Running 5 (8d ago) 194d
longhorn-system longhorn-manager-kpkh7 1/1 Running 4 (8d ago) 194d
longhorn-system longhorn-ui-6ddb78495f-dtcv6 1/1 Running 4 (8d ago) 132d
longhorn-system share-manager-pvc-289a1a2e-f79a-42af-9324-49a8c00c7c9a 1/1 Running 0 8d
longhorn-system share-manager-pvc-39921efe-87ce-47ac-864d-076d6fb4b069 1/1 Running 0 170m
longhorn-system share-manager-pvc-962745a8-e08f-4726-84c8-45f151a432d7 1/1 Running 0 170m
longhorn-system share-manager-pvc-d9c9d08c-08c5-44e9-91fa-bea34240bebe 1/1 Running 0 170m
metallb-system metallb-controller-7f6b8b7fdd-ljfck 1/1 Running 4 (8d ago) 132d
metallb-system metallb-speaker-h6t9q 1/1 Running 15 (8d ago) 194d
metallb-system metallb-speaker-jr76q 1/1 Running 5 (8d ago) 194d
metallb-system metallb-speaker-whz4w 1/1 Running 6 (8d ago) 194d
```

Clean Registry

You can use this proceed to clean your registry of images that are not currently being used.
BBE Cloudsetup does not automatically perform any cleanup of old images. If needed, you can remove old images manually by using the `bbecloudsetup registry clean` command.

1. On the jump host system, run the `bbecloudsetup registry clean` command by entering:

```
user@host:~/bbecloudsetup$ ./bbecloudsetup registry clean --context host-name
```

Options:

- **--kubeconfig** (or **-k**)—The path to the kubeconfig file.
- **--context** (or **-c**)—The context to use from the kubeconfig file.
- **--dry-run** (or **-n**)—Lets you see the existing images without performing any actions.

Running the `registry clean` command checks the local registry (if present) for any images that are currently not in use by any cluster resources. If you run the command with the **dry-run** option, you can see the images that exist without deleting them.

2. Verify that the images have been deleted.

```
user@host:~/bbecloudsetup$ ./bbecloudsetup registry clean --context host-name -n
INFO[0000] No images to delete
```