Red Hat build of MicroShift 4.15

Installing

Installing and configuring MicroShift clusters

Last Updated: 2024-11-01

Red Hat build of MicroShift 4.15 Installing

Installing and configuring MicroShift clusters

Legal Notice

Abstract

This document provides information about installing MicroShift and details about some configuration processes.

Table of Contents

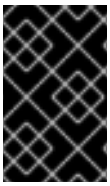
CHAPTER 1. INSTALLING FROM AN RPM PACKAGE

You can install MicroShift from an RPM package on a machine with a supported version of Red Hat Enterprise Linux (RHEL).

1.1. SYSTEM REQUIREMENTS FOR INSTALLING MICROSHIFT

The following conditions must be met prior to installing MicroShift:

- A compatible version of RHEL or RHEL for Edge.
- AArch64 or x86_64 system architecture.
- 2 CPU cores.
- 2 GB RAM for MicroShift or 3 GB RAM, required by RHEL for networked-based HTTPs or FTP installations.
- 10 GB of storage.
- You have an active MicroShift subscription on your Red Hat account. If you do not have a subscription, contact your sales representative for more information.
- You have a Logical Volume Manager (LVM) Volume Group (VG) with sufficient capacity for the Persistent Volumes (PVs) of your workload.



IMPORTANT

Secure access must be configured before running MicroShift to ensure proper functionality and security. For more information, see Using secure communications between two systems with OpenSSH.

1.2. COMPATIBILITY TABLE

Plan to pair a supported version of RHEL for Edge with the MicroShift version you are using as described in the following compatibility table:

Red Hat Device Edge release compatibility matrix

The two products of Red Hat Device Edge work together as a single solution for device-edge computing. To successfully pair your products, use the verified releases together for each as listed in the following table:

RHEL for Edge Version(s)	MicroShift Version	MicroShift Release Status	MicroShift Supported Updates
9.2, 9.3	4.15	Generally Available	4.15.0→4.15.z and 4.15→future minor version
9.2, 9.3	4.14	Generally Available	4.14.0→4.14.z and 4.14→4.15

9.2	4.13	Technology Preview	None
8.7	4.12	Developer Preview	None

1.3. BEFORE INSTALLING MICROSHIFT FROM AN RPM PACKAGE

Preparation of the host machine is recommended prior to installing MicroShift for memory configuration and FIPS mode.

1.3.1. Configuring volume groups

MicroShift uses the logical volume manager storage (LVMS) Container Storage Interface (CSI) plugin for providing storage to persistent volumes (PVs). LVMS relies on the Linux logical volume manager (LVM) to dynamically manage the backing logical volumes (LVs) for PVs. For this reason, your machine must have an LVM volume group (VG) with unused space in which LVMS can create the LVs for your workload's PVs.

To configure a volume group (VG) that allows LVMS to create the LVs for your workload's PVs, lower the **Desired Size** of your root volume during the installation of RHEL. Lowering the size of your root volume allows unallocated space on the disk for additional LVs created by LVMS at runtime.

1.3.2. Prepare for FIPS mode

If your use case requires running MicroShift containers in FIPS mode, you must install RHEL with FIPS enabled. After the worker machine is configured to run in FIPS mode, your MicroShift containers are automatically configured to also run in FIPS mode.

IMPORTANT

Because FIPS must be enabled before the operating system that your cluster uses starts for the first time, you cannot enable FIPS after you deploy a cluster.

Additional resources

- Using FIPS mode with MicroShift

1.4. PREPARING TO INSTALL MICROSHIFT FROM AN RPM PACKAGE

Configure your RHEL machine to have a logical volume manager (LVM) volume group (VG) with sufficient capacity for the persistent volumes (PVs) of your workload.

Prerequisites

- The system requirements for installing MicroShift have been met.
- You have root user access to your machine.
- You have configured your LVM VG with the capacity needed for the PVs of your workload.

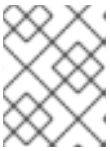
Procedure

1. In the graphical installer under **Installation Destination** in the **Storage Configuration** subsection, select **Custom → Done** to open the dialog for configuring partitions and volumes. The Manual Partitioning window is displayed.

2. Under **New Red Hat Enterprise Linux 9.x Installation**, select **Click here to create them automatically**.

3. Select the root partition, **/**, reduce **Desired Capacity** so that the VG has sufficient capacity for your PVs, and then click **Update Settings**.

4. Complete your installation.



NOTE

For more options on partition configuration, read the guide linked in the Additional information section for Configuring Manual Partitioning.

5. As a root user, verify the VG capacity available on your system by running the following command:

```
$ sudo vgs
```

Example output:

```
VG   #PV #LV #SN Attr   VSize    VFree
rhel   1   2   0 wz--n- <127.00g 54.94g
```

Additional resources

- Download the pull secret from the Red Hat Hybrid Cloud Console.
- Configuring MicroShift.
- For more options on partition configuration, read Configuring Manual Partitioning.
- For more information about resizing your existing LVs to free up capacity in your VGs, read Managing LVM Volume Groups.
- For more information about creating VGs and PVs, read Overview of logical volume management.

1.5. INSTALLING MICROSHIFT FROM AN RPM PACKAGE

Use the following procedure to install MicroShift from an RPM package.

Prerequisites

- The system requirements for installing MicroShift have been met.
- You have completed the steps of preparing to install MicroShift from an RPM package.

Procedure

1. As a root user, enable the MicroShift repositories by running the following command:

```
$ sudo subscription-manager repos \
    --enable rhocp-4.15-for-rhel-9-$(uname -m)-rpms \
    --enable fast-datapath-for-rhel-9-$(uname -m)-rpms
```

2. Install MicroShift by running the following command:

```
$ sudo dnf install -y microshift
```

3. Download your installation pull secret from the Red Hat Hybrid Cloud Console to a temporary folder, for example, **$HOME/openshift-pull-secret**. This pull secret allows you to authenticate with the container registries that serve the container images used by MicroShift.

4. To copy the pull secret to the **/etc/crio** folder of your RHEL machine, run the following command:

```
$ sudo cp $HOME/openshift-pull-secret /etc/crio/openshift-pull-secret
```

5. Make the root user the owner of the **/etc/crio/openshift-pull-secret** file by running the following command:

```
$ sudo chown root:root /etc/crio/openshift-pull-secret
```

6. Make the **/etc/crio/openshift-pull-secret** file readable and writeable by the root user only by running the following command:

```
$ sudo chmod 600 /etc/crio/openshift-pull-secret
```

7. If your RHEL machine has a firewall enabled, you must configure a few mandatory firewall rules. For **firewalld**, run the following commands:

```
$ sudo firewall-cmd --permanent --zone=trusted --add-source=10.42.0.0/16
```

```
$ sudo firewall-cmd --permanent --zone=trusted --add-source=169.254.169.1
```

```
$ sudo firewall-cmd --reload
```

If the Volume Group (VG) that you have prepared for MicroShift used the default name **rhel**, no further configuration is necessary. If you have used a different name, or if you want to change more configuration settings, see the Configuring MicroShift section.

1.5.1. Installing the Operator Lifecycle Manager (OLM) from an RPM package

When you install MicroShift, the Operator Lifecycle Manager (OLM) package is not installed by default. You can install the OLM on your MicroShift instance using a RPM package.

Procedure

1. Install the OLM package by running the following command:

```
$ sudo dnf install microshift-olm
```

2. To apply the manifest from the package to an active cluster, run the following command:

```
$ sudo systemctl restart microshift
```

Additional resources

- System requirements for installing MicroShift
- Preparing to install MicroShift from an RPM package
- Using Operator Lifecycle Manager with MicroShift

1.5.2. Installing the GitOps Argo CD manifests from an RPM package

You can use a lightweight version of the OpenShift GitOps with MicroShift to help manage your applications. Install the necessary Argo CD manifests using an RPM package. This RPM package included the necessary manifests that runs core Argo CD.

IMPORTANT

This process installs the basic GitOps functionalities, the Argo CD CLI is not currently available on MicroShift.

Prerequisites

- You installed MicroShift version 4.14 or higher
- Additional RAM storage of 250MB recommended

Procedure

1. Enable the GitOps repository with the subscription manager by running the following command:

```
$ sudo subscription-manager repos --enable=gitops-1.12-for-rhel-9-$(uname -m)-rpms
```

2. Install the GitOps package by running the following command:

```
$ sudo dnf install -y microshift-gitops
```

3. To deploy Argo CD pods, restart the MicroShift service by running the following command:

```
$ sudo systemctl restart microshift
```

Verification

- You can verify that your pods are running properly by running the following command:

```
$ oc get pods -n openshift-gitops
```

Example output

```
NAME                                READY   STATUS    RESTARTS   AGE
```

```
argocd-application-controller-0       1/1     Running   0          4m11s
argocd-redis-56844446bc-dzmhf         1/1     Running   0          4m12s
argocd-repo-server-57b4f896cf-7qk8l   1/1     Running   0          4m12s
```

1.6. STARTING THE MICROSHIFT SERVICE

Use the following procedure to start the MicroShift service.

Prerequisites

- You have installed MicroShift from an RPM package.

Procedure

1. As a root user, start the MicroShift service by entering the following command:

```
$ sudo systemctl start microshift
```

2. Optional: To configure your RHEL machine to start MicroShift when your machine starts, enter the following command:

```
$ sudo systemctl enable microshift
```

3. Optional: To disable MicroShift from automatically starting when your machine starts, enter the following command:

```
$ sudo systemctl disable microshift
```

NOTE

The first time that the MicroShift service starts, it downloads and initializes the container images for MicroShift. As a result, it can take several minutes for MicroShift to start the first time that the service is deployed. Boot time is reduced for subsequent starts of the MicroShift service.

1.7. STOPPING THE MICROSHIFT SERVICE

Use the following procedure to stop the MicroShift service.

Prerequisites

- The MicroShift service is running.

Procedure

1. Enter the following command to stop the MicroShift service:

```
$ sudo systemctl stop microshift
```

2. Workloads deployed on MicroShift might continue running even after the MicroShift service has been stopped. Enter the following command to display running workloads:

```
$ sudo crictl ps -a
```

3. Enter the following commands to stop the deployed workloads:

```
$ sudo systemctl stop kubepods.slice
```

1.8. HOW TO ACCESS THE MICROSHIFT CLUSTER

Use the procedures in this section to access the MicroShift cluster, either from the same machine running the MicroShift service or remotely from a workstation. You can use this access to observe and administrate workloads. When using these steps, choose the **kubeconfig** file that contains the host name or IP address you want to connect with and place it in the relevant directory. As listed in each procedure, you use the OpenShift Container Platform CLI tool (**oc**) for cluster activities.

Additional resources

- Installing the OpenShift CLI tool.

1.8.1. Accessing the MicroShift cluster locally

Use the following procedure to access the MicroShift cluster locally by using a **kubeconfig** file.

Prerequisites

- You have installed the **oc** binary.

Procedure

1. Optional: to create a ~/**.kube**/ folder if your RHEL machine does not have one, run the following command:

```
$ mkdir -p ~/.kube/
```

2. Copy the generated local access **kubeconfig** file to the ~/**.kube**/ directory by running the following command:

```
$ sudo cat /var/lib/microshift/resources/kubeadmin/kubeconfig > ~/.kube/config
```

3. Update the permissions on your ~/**.kube**/**config** file by running the following command:

```
$ chmod go-r ~/.kube/config
```

Verification

- Verify that MicroShift is running by entering the following command:

```
$ oc get all -A
```

1.8.2. Opening the firewall for remote access to the MicroShift cluster

Use the following procedure to open the firewall so that a remote user can access the MicroShift cluster. This procedure must be completed before a workstation user can access the cluster remotely.

For this procedure, **user@microshift** is the user on the MicroShift host machine and is responsible for setting up that machine so that it can be accessed by a remote user on a separate workstation.

Prerequisites

- You have installed the **oc** binary.
- Your account has cluster administration privileges.

Procedure

- As **user@microshift** on the MicroShift host, open the firewall port for the Kubernetes API server (**6443/tcp**) by running the following command:

```
[user@microshift]$ sudo firewall-cmd --permanent --zone=public --add-port=6443/tcp &&
sudo firewall-cmd --reload
```

Verification

- As **user@microshift**, verify that MicroShift is running by entering the following command:

```
[user@microshift]$ oc get all -A
```

1.8.3. Accessing the MicroShift cluster remotely

Use the following procedure to access the MicroShift cluster from a remote workstation by using a **kubeconfig** file.

The **user@workstation** login is used to access the host machine remotely. The **<user>** value in the procedure is the name of the user that **user@workstation** logs in with to the MicroShift host.

Prerequisites

- You have installed the **oc** binary.
- The **user@microshift** has opened the firewall from the local host.

Procedure

1. As **user@workstation**, create a **~/.kube/** folder if your RHEL machine does not have one by running the following command:

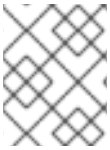
```
[user@workstation]$ mkdir -p ~/.kube/
```

2. As **user@workstation**, set a variable for the hostname of your MicroShift host by running the following command:

```
[user@workstation]$ MICROSHIFT_MACHINE=<name or IP address of MicroShift machine>
```

3. As **user@workstation**, copy the generated **kubeconfig** file that contains the host name or IP address you want to connect with from the RHEL machine running MicroShift to your local machine by running the following command:

```
[user@workstation]$ ssh <user>@$MICROSHIFT_MACHINE "sudo cat
/var/lib/microshift/resources/kubeadmin/$MICROSHIFT_MACHINE/kubeconfig" >
~/.kube/config
```



NOTE

To generate **kubeconfig** files for this step, see the "Generating additional kubeconfig files for remote access" link in the additional resources section.

1. As **user@workstation**, update the permissions on your **~/.kube/config** file by running the following command:

```
$ chmod go-r ~/.kube/config
```

Verification

- As **user@workstation**, verify that MicroShift is running by entering the following command:

```
[user@workstation]$ oc get all -A
```

CHAPTER 2. USING FIPS MODE WITH MICROSHIFT

You can use FIPS mode with RPM-based installations of MicroShift on Red Hat Enterprise Linux (RHEL) 9.

- To enable FIPS mode in MicroShift containers, the worker machine kernel must be enabled to run in FIPS mode before the machine starts.
- Using FIPS with Red Hat Enterprise Linux for Edge (RHEL for Edge) images is not supported.

2.1. FIPS MODE WITH RHEL RPM-BASED INSTALLATIONS

Using FIPS with MicroShift requires enabling the cryptographic module self-checks in your Red Hat Enterprise Linux (RHEL) installation. After the host operating system has been configured to start with the FIPS modules, MicroShift containers are automatically enabled to run in FIPS mode.

- When RHEL is started in FIPS mode, MicroShift core components use the RHEL cryptographic libraries that have been submitted to NIST for FIPS 140-2/140-3 validation on only the x86_64 architectures.
- You must enable FIPS mode when you install RHEL 9 on the machines that you plan to use as worker machines.

IMPORTANT

Because FIPS must be enabled before the operating system that your cluster uses starts for the first time, you cannot enable FIPS after you deploy a cluster.

- MicroShift uses a FIPS-compatible Golang compiler.
- FIPS is supported in the CRI-O container runtime.

2.1.1. Limitations

- TLS implementation FIPS support is not complete.
- The FIPS implementation does not offer a single function that both computes hash functions and validates the keys that are based on that hash. This limitation continues to be evaluated for improvement in future MicroShift releases.

2.1.2. Installing RHEL in FIPS mode

To install RHEL with FIPS, follow the guidance in the Installing the system in FIPS mode of the RHEL documentation.

2.2. ADDITIONAL RESOURCES

- Installing the system in FIPS mode
- Enabling FIPS mode in a container
- Federal Information Processing Standards 140 and FIPS mode

CHAPTER 3. MIRRORING CONTAINER IMAGES FOR DISCONNECTED INSTALLATIONS

You can use a custom container registry when you deploy MicroShift in a disconnected network. Running your cluster in a restricted network without direct internet connectivity is possible by installing the cluster from a mirrored set of container images in a private registry.

3.1. MIRROR CONTAINER IMAGES INTO AN EXISTING REGISTRY

Using a custom air-gapped container registry, or mirror, is necessary with certain user environments and workload requirements. Mirroring allows for the transfer of container images and updates to air-gapped environments where they can be installed on a MicroShift instance.

To create an air-gapped mirror registry for MicroShift containers, you must complete the following steps:

- Get the container image list to be mirrored.
- Configure the mirroring prerequisites.
- Download images on a host with internet access.
- Copy the downloaded image directory to an air-gapped site.
- Upload images to a mirror registry in an air-gapped site.
- Configure your MicroShift hosts to use the mirror registry.

Additional resources

- Creating a mirror registry with mirror registry for Red Hat OpenShift

3.2. GETTING THE MIRROR REGISTRY CONTAINER IMAGE LIST

To use a mirror registry, you must know which container image references are used by a specific version of MicroShift. These references are provided in the **release-<arch>.json** files that are part of the **microshift-release-info** RPM package.

NOTE

To mirror the Operator Lifecycle Manager (OLM) in disconnected environments, add the references provided in the **release-olm-$ARCH.json** that is included in the **microshift-olm** RPM and follow the same procedure. Use **oc-mirror** for mirroring Operator catalogs and Operators.

Prerequisites

- You have installed jq.

Procedure

1. Access the list of container image references by using one of the following methods:

- If the package is installed on the MicroShift host, get the location of the files by running the following command:

```
$ rpm -ql microshift-release-info
```

Example output

```
/usr/share/microshift/release/release-x86_64.json
```

- If the package is not installed on a MicroShift host, download and unpack the RPM package without installing it by running the following command:

```
$ rpm2cpio microshift-release-info*.noarch.rpm | cpio -idmv
```

Example output

```
/usr/share/microshift/release/release-x86_64.json
```

2. Extract the list of container images into the **microshift-container-refs.txt** file by running the following commands:

```
$ RELEASE_FILE=/usr/share/microshift/release/release-$(uname -m).json
```

```
$ jq -r '.images | .[]' ${RELEASE_FILE} > microshift-container-refs.txt
```

NOTE

After the **microshift-container-refs.txt** file is created with the MicroShift container image list, you can append the file with other user-specific image references before running the mirroring procedure.

3.3. CONFIGURING MIRRORING PREREQUISITES

You must create a container image registry credentials file that allows the mirroring of images from your internet-connected mirror host to your air-gapped mirror. Follow the instructions in the "Configuring credentials that allow images to be mirrored" link provided in the "Additional resources" section. These instructions guide you to create a **~/.pull-secret-mirror.json** file on the mirror registry host that includes the user credentials for accessing the mirror.

3.3.1. Example mirror registry pull secret entry

For example, the following section is added to the pull secret file for the **microshift_quay:8443** mirror registry using **microshift:microshift** as username and password.

Example mirror registry section for pull secret file

```
"<microshift_quay:8443>": { 1
   "auth": "<microshift_auth>", 2
   "email": "<microshift_quay@example.com>" 3
},
```

1. Replace the **<registry_host>:<port>** value **microshift_quay:8443** with the host name and port of your mirror registry server.
2. Replace the **<microshift_auth>** value with the user password.
3. Replace the **</microshift_quay@example.com>** value with the user email.

Additional resources

- Configuring credentials that allow images to be mirrored

3.4. DOWNLOADING CONTAINER IMAGES

After you have located the container list and completed the mirroring prerequisites, download the container images to a host with internet access.

Prerequisites

- You are logged into a host with access to the internet.
- You have ensured that the **.pull-secret-mirror.json** file and **microshift-containers** directory contents are available locally.

Procedure

1. Install the **skopeo** tool used for copying the container images by running the following command:

```
$ sudo dnf install -y skopeo
```

2. Set the environment variable that points to the pull secret file:

```
$ PULL_SECRET_FILE=~/.pull-secret-mirror.json
```

3. Set the environment variable that points to the list of container images:

```
$ IMAGE_LIST_FILE=~/microshift-container-refs.txt
```

4. Set the environment variable that points to the destination directory for storing the downloaded data:

```
$ IMAGE_LOCAL_DIR=~/microshift-containers
```

5. Run the following script to download the container images to the **${IMAGE_LOCAL_DIR}** directory:

```
while read -r src_img ; do
  # Remove the source registry prefix
  dst_img=$(echo "${src_img}" | cut -d '/' -f 2-)

  # Run the image download command
  echo "Downloading '${src_img}' to '${IMAGE_LOCAL_DIR}'"
```

```
    mkdir -p "${IMAGE_LOCAL_DIR}/${dst_img}"
    skopeo copy --all --quiet \
      --preserve-digests \
      --authfile "${PULL_SECRET_FILE}" \
      docker://"${src_img}" dir://"${IMAGE_LOCAL_DIR}/${dst_img}"

done < "${IMAGE_LIST_FILE}"
```

6. Transfer the image set to the target environment, such as air-gapped site. Then you can upload the image set into the mirror registry.

3.5. UPLOADING CONTAINER IMAGES TO A MIRROR REGISTRY

To use your container images at an air-gapped site, upload them to the mirror registry using the following procedure.

Prerequisites

- You are logged into a host with access to **microshift-quay**.
- The **.pull-secret-mirror.json** file is available locally.
- The **microshift-containers** directory contents are available locally.

Procedure

1. Install the **skopeo** tool used for copying the container images by running the following command:

```
$ sudo dnf install -y skopeo
```

2. Set the environment variables pointing to the pull secret file:

```
$ IMAGE_PULL_FILE=~/.pull-secret-mirror.json
```

3. Set the environment variables pointing to the local container image directory:

```
$ IMAGE_LOCAL_DIR=~/microshift-containers
```

4. Set the environment variables pointing to the mirror registry URL for uploading the container images:

```
$ TARGET_REGISTRY=<registry_host>:<port> 1
```

1 Replace **<registry_host>:<port>** with the host name and port of your mirror registry server.

5. Run the following script to upload the container images to the **${TARGET_REGISTRY}** mirror registry:

```
image_tag=mirror-$(date +%y%m%d%H%M%S)
image_cnt=1
```

```
  # Uses timestamp and counter as a tag on the target images to avoid
  # their overwrite by the 'latest' automatic tagging

pushd "${IMAGE_LOCAL_DIR}" >/dev/null
while read -r src_manifest ; do
  # Remove the manifest.json file name
  src_img=$(dirname "${src_manifest}")
  # Add the target registry prefix and remove SHA
  dst_img="${TARGET_REGISTRY}/${src_img}"
  dst_img=$(echo "${dst_img}" | awk -F'@' '{print $1}')

  # Run the image upload command
  echo "Uploading '${src_img}' to '${dst_img}'"
  skopeo copy --all --quiet \
    --preserve-digests \
    --authfile "${IMAGE_PULL_FILE}" \
    dir://"${IMAGE_LOCAL_DIR}/${src_img}" docker://"${dst_img}:${image_tag}-
${image_cnt}"
  # Increment the counter
  (( image_cnt += 1 ))

done < <(find . -type f -name manifest.json -printf '%P\n')
popd >/dev/null
```

3.6. CONFIGURING HOSTS FOR MIRROR REGISTRY ACCESS

To configure a MicroShift host to use a mirror registry, you must give the MicroShift host access to the registry by creating a configuration file that maps the Red Hat registry host names to the mirror.

Prerequisites

- Your mirror host has access to the internet.
- The mirror host can access the mirror registry.
- You configured the mirror registry for use in your restricted network.
- You downloaded the pull secret and modified it to include authentication to your mirror repository.

Procedure

1. Log into your MicroShift host.
2. Enable the SSL certificate trust on any host accessing the mirror registry by completing the following steps:
   a. Copy the **rootCA.pem** file from the mirror registry, for example, **<registry_path>/quay-rootCA**, to the MicroShift host at the **/etc/pki/ca-trust/source/anchors** directory.
   b. Enable the certificate in the system-wide trust store configuration by running the following command:

      ```
      $ sudo update-ca-trust
      ```

3. Create the **/etc/containers/registries.conf.d/999-microshift-mirror.conf** configuration file that maps the Red Hat registry host names to the mirror registry:

Example mirror configuration file

```
[[registry]]
   prefix = ""
   location = "<registry_host>:<port>" (1)
   mirror-by-digest-only = true
   insecure = false

[[registry]]
   prefix = ""
   location = "quay.io"
   mirror-by-digest-only = true
[[registry.mirror]]
   location = "<registry_host>:<port>"
   insecure = false

[[registry]]
   prefix = ""
   location = "registry.redhat.io"
   mirror-by-digest-only = true
[[registry.mirror]]
   location = "<registry_host>:<port>"
   insecure = false

[[registry]]
   prefix = ""
   location = "registry.access.redhat.com"
   mirror-by-digest-only = true
[[registry.mirror]]
   location = "<registry_host>:<port>"
   insecure = false
```

(1) Replace **<registry_host>:<port>** with the host name and port of your mirror registry server, for example, **<microshift-quay:8443>**.

4. Enable the MicroShift service by running the following command:

```
$ sudo systemctl enable microshift
```

5. Reboot the host by running the following command:

```
$ sudo reboot
```

CHAPTER 4. EMBEDDING IN A RHEL FOR EDGE IMAGE

You can embed MicroShift into a Red Hat Enterprise Linux for Edge (RHEL for Edge) image. Use this guide to build a RHEL image containing MicroShift.

4.1. SYSTEM REQUIREMENTS FOR INSTALLING MICROSHIFT

The following conditions must be met prior to installing MicroShift:

- A compatible version of RHEL or RHEL for Edge.
- AArch64 or x86_64 system architecture.
- 2 CPU cores.
- 2 GB RAM for MicroShift or 3 GB RAM, required by RHEL for networked-based HTTPs or FTP installations.
- 10 GB of storage.
- You have an active MicroShift subscription on your Red Hat account. If you do not have a subscription, contact your sales representative for more information.
- You have a Logical Volume Manager (LVM) Volume Group (VG) with sufficient capacity for the Persistent Volumes (PVs) of your workload.

IMPORTANT

Secure access must be configured before running MicroShift to ensure proper functionality and security. For more information, see Using secure communications between two systems with OpenSSH.

Plan to use a supported version of RHEL paired with your version of MicroShift as described in the following table.

4.2. COMPATIBILITY TABLE

Plan to pair a supported version of RHEL for Edge with the MicroShift version you are using as described in the following compatibility table:

Red Hat Device Edge release compatibility matrix

The two products of Red Hat Device Edge work together as a single solution for device-edge computing. To successfully pair your products, use the verified releases together for each as listed in the following table:

RHEL for Edge Version(s)	MicroShift Version	MicroShift Release Status	MicroShift Supported Updates
9.2, 9.3	4.15	Generally Available	4.15.0→4.15.z and 4.15→future minor version

9.2, 9.3	4.14	Generally Available	4.14.0→4.14.z and 4.14→4.15
9.2	4.13	Technology Preview	None
8.7	4.12	Developer Preview	None

4.3. PREPARING FOR IMAGE BUILDING

Read Composing, installing, and managing RHEL for Edge images .

To build an Red Hat Enterprise Linux for Edge (RHEL for Edge) 9.2 image for a given CPU architecture, you need a RHEL 9.2 build host of the same CPU architecture that meets the Image Builder system requirements.

Follow the instructions in Installing Image Builder to install Image Builder and the **composer-cli** tool.

4.4. ADDING MICROSHIFT REPOSITORIES TO IMAGE BUILDER

Use the following procedure to add the MicroShift repositories to Image Builder on your build host.

Prerequisites

- Your build host meets the Image Builder system requirements.
- You have installed and set up Image Builder and the **composer-cli** tool.
- You have root-user access to your build host.

Procedure

1. Create an Image Builder configuration file for adding the **rhocp-4.15** RPM repository source required to pull MicroShift RPMs by running the following command:

```
cat > rhocp-4.15.toml <<EOF
id = "rhocp-4.15"
name = "Red Hat OpenShift Container Platform 4.15 for RHEL 9"
type = "yum-baseurl"
url = "https://cdn.redhat.com/content/dist/layered/rhel9/$(uname -m)/rhocp/4.15/os"
check_gpg = true
check_ssl = true
system = false
rhsm = true
EOF
```

2. Create an Image Builder configuration file for adding the **fast-datapath** RPM repository by running the following command:

```
cat > fast-datapath.toml <<EOF
id = "fast-datapath"
name = "Fast Datapath for RHEL 9"
```

```
type = "yum-baseurl"
url = "https://cdn.redhat.com/content/dist/layered/rhel9/$(uname -m)/fast-datapath/os"
check_gpg = true
check_ssl = true
system = false
rhsm = true
EOF
```

3. Add the sources to the Image Builder by running the following commands:

```
$ sudo composer-cli sources add rhocp-4.15.toml
```

```
$ sudo composer-cli sources add fast-datapath.toml
```

Verification

- Confirm that the sources were added properly by running the following command:

```
$ sudo composer-cli sources list
```

Example output

```
appstream
baseos
fast-datapath
rhocp-4.15
```

Additional resources

- Image Builder system requirements
- Installing Image Builder

4.5. ADDING THE MICROSHIFT SERVICE TO A BLUEPRINT

Adding the MicroShift RPM package to an Image Builder blueprint enables the build of a RHEL for Edge image with MicroShift embedded.

- Start with step 1 to create your own minimal blueprint file which results in a faster MicroShift installation.
- Start with step 2 to use the generated blueprint for installation which includes all the RPM packages and container images. This is a longer installation process, but a faster start up because container references are accessed locally.

IMPORTANT

- Replace *<microshift_blueprint.toml>* in the following procedures with the name of the TOML file you are using.
- Replace *<microshift_blueprint>* in the following procedures with the name you want to use for your blueprint.

Procedure

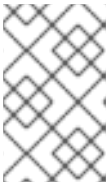
1. Use the following example to create your own blueprint file:

Custom Image Builder blueprint example

```
cat > <microshift_blueprint.toml> <<EOF (1)
name = "<microshift_blueprint>" (2)

description = ""
version = "0.0.1"
modules = []
groups = []

[[packages]]
name = "microshift"
version = "*"

[customizations.services]
enabled = ["microshift"]
EOF
```

(1) *<microshift_blueprint.toml>* is the name of the TOML file.

(2) *<microshift_blueprint>* is the name of your blueprint.



NOTE

The wildcard * in the commands uses the latest MicroShift RPMs. If you need a specific version, substitute the wildcard for the version you want. For example, insert **4.15.0** to download the MicroShift 4.15.0 RPMs.

2. Optional. Use the blueprint installed in the **/usr/share/microshift/blueprint** directory that is specific to your platform architecture. See the following example snippet for an explanation of the blueprint sections:

Generated Image Builder blueprint example snippet

```
name = "microshift_blueprint"
description = "MicroShift 4.15.1 on x86_64 platform"
version = "0.0.1"
modules = []
groups = []

[[packages]] (1)
name = "microshift"
version = "4.15.1"
...
...

[customizations.services] (2)
enabled = ["microshift"]
```

```
[customizations.firewall]
ports = ["22:tcp", "80:tcp", "443:tcp", "5353:udp", "6443:tcp", "30000-32767:tcp", "30000-
32767:udp"]
...
...

[[containers]] 3
source = "quay.io/openshift-release-dev/ocp-v4.0-art-
dev@sha256:f41e79c17e8b41f1b0a5a32c3e2dd7cd15b8274554d3f1ba12b2598a347475f4"

[[containers]]
source = "quay.io/openshift-release-dev/ocp-v4.0-art-
dev@sha256:dbc65f1fba7d92b36cf7514cd130fe83a9bd211005ddb23a8dc479e0eea645fd"
...
…
EOF
```

1. References for all non-optional MicroShift RPM packages using the same version compatible with the **microshift-release-info** RPM.
2. References for automatically enabling MicroShift on system startup and applying default networking settings.
3. References for all non-optional MicroShift container images necessary for an offline deployment.

3. Add the blueprint to the Image Builder by running the following command:

```
$ sudo composer-cli blueprints push <microshift_blueprint.toml> 1
```

1. Replace *<microshift_blueprint.toml>* with the name of your TOML file.

Verification

1. Verify the Image Builder configuration listing only MicroShift packages by running the following command:

```
$ sudo composer-cli blueprints depsolve <microshift_blueprint> | grep microshift 1
```

1. Replace *<microshift_blueprint>* with the name of your blueprint.

Example output

```
blueprint: microshift_blueprint v0.0.1
    microshift-greenboot-4.15.1-202305250827.p0.g4105d3b.assembly.4.15.1.el9.noarch
    microshift-networking-4.15.1-202305250827.p0.g4105d3b.assembly.4.15.1.el9.x86_64
    microshift-release-info-4.15.1-202305250827.p0.g4105d3b.assembly.4.15.1.el9.noarch
    microshift-4.15.1-202305250827.p0.g4105d3b.assembly.4.15.1.el9.x86_64
    microshift-selinux-4.15.1-202305250827.p0.g4105d3b.assembly.4.15.1.el9.noarch
```

2. Optional: Verify the Image Builder configuration listing all components to be installed by running the following command:

```
$ sudo composer-cli blueprints depsolve <microshift_blueprint> 1
```

1 Replace *<microshift_blueprint>* with the name of your blueprint.

4.5.1. Adding the Operator Lifecycle Manager (OLM) service to a blueprint

When you install MicroShift, the Operator Lifecycle Manager (OLM) package is not installed by default. You can add the **microshift-olm** package in the ostree blueprint to enable OLM in MicroShift.

1. Edit your ostree blueprint by running the following example command:

```
$ vi <microshift_blueprint.toml> 1
```

1 Specify the name of the blueprint file you used when adding the MicroShift service.

2. Add the following example text to your ostree blueprint:

```
[[packages]]
name = "microshift-olm"
version = "*"
```

3. To apply the manifest from the package to an active cluster, you must build a new OSTree system then deploy it on the machine. To update your OSTree system, use the instruction's in "Applying updates on an OSTree system"

Additional resources

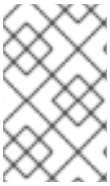
- Using Operator Lifecycle Manager with MicroShift
- Applying updates on an OSTree system

4.6. ADDING A CERTIFICATE AUTHORITY BUNDLE

MicroShift uses the host trust bundle when clients evaluate server certificates. You can also use a customized security certificate chain to improve the compatibility of your endpoint certificates with clients specific to your deployments. To do this, you can add a certificate authority (CA) bundle with root and intermediate certificates to the Red Hat Enterprise Linux for Edge (RHEL for Edge) system-wide trust store.

4.6.1. Adding a certificate authority bundle to an rpm-ostree image

You can include additional trusted certificate authorities (CAs) to the Red Hat Enterprise Linux for Edge (RHEL for Edge) **rpm-ostree** image by adding them to the blueprint that you use to create the image. Using the following procedure sets up additional CAs to be trusted by the operating system when pulling images from an image registry.



NOTE

This procedure requires you to configure the CA bundle customizations in the blueprint, and then add steps to your kickstart file to enable the bundle. In the following steps, **data** is the key, and **<value>** represents the PEM-encoded certificate.

Prerequisites

- You have root user access to your build host.
- Your build host meets the Image Builder system requirements.
- You have installed and set up Image Builder and the **composer-cli** tool.

Procedure

1. Add the following custom values to your blueprint to add a directory.
   a. Add instructions to your blueprint on the host where the image is built to create the directory, for example, **/etc/pki/ca-trust/source/anchors/** for your certificate bundles.

```
[[customizations.directories]]
path = "/etc/pki/ca-trust/source/anchors"
```

   b. After the image has booted, create the certificate bundles, for example, **/etc/pki/ca-trust/source/anchors/cert1.pem**:

```
[[customizations.files]]
path = "/etc/pki/ca-trust/source/anchors/cert1.pem"
data = "<value>"
```

2. To enable the certificate bundle in the system-wide trust store configuration, use the **update-ca-trust** command on the host where the image you are using has booted, for example:

```
$ sudo update-ca-trust
```

NOTE

The **update-ca-trust** command might be included in the **%post** section of a kickstart file used for MicroShift host installation so that all the necessary certificate trust is enabled on the first boot. You must configure the CA bundle customizations in the blueprint before adding steps to your kickstart file to enable the bundle.

```
%post
# Update certificate trust storage in case new certificates were
# installed at /etc/pki/ca-trust/source/anchors directory
update-ca-trust
%end
```

Additional resources

- Creating the RHEL for Edge image
- Using Shared System Certificates (RHEL 9)
- Supported image customizations (RHEL 9)

4.7. CREATING THE RHEL FOR EDGE IMAGE

Use the following procedure to create the ISO. The RHEL for Edge Installer image pulls the commit from the running container and creates an installable boot ISO with a Kickstart file configured to use the embedded **rpm-ostree** commit.

Prerequisites

- Your build host meets the Image Builder system requirements.
- You have installed and set up Image Builder and the **composer-cli** tool.
- You have root-user access to your build host.
- You have installed the **podman** tool.

Procedure

1. Start an **ostree** container image build by running the following command:

```
$ BUILDID=$(sudo composer-cli compose start-ostree --ref "rhel/{op-system-version-major}/$(uname -m)/edge" <microshift_blueprint> edge-container | awk '{print $2}') 1
```

1 Replace *<microshift_blueprint>* with the name of your blueprint.

   This command also returns the identification (ID) of the build for monitoring.

2. You can check the status of the build periodically by running the following command:

```
$ sudo composer-cli compose status
```

Example output of a running build

```
ID                                   Status    Time                  Blueprint             Version   Type
Size
cc3377ec-4643-4483-b0e7-6b0ad0ae6332 RUNNING   Wed Jun 7 12:26:23 2023
microshift_blueprint     0.0.1     edge-container
```

Example output of a completed build

```
ID                                   Status    Time                  Blueprint           Version   Type
Size
cc3377ec-4643-4483-b0e7-6b0ad0ae6332 FINISHED  Wed Jun 7 12:32:37 2023
microshift_blueprint  0.0.1     edge-container
```

NOTE

You can use the **watch** command to monitor your build if you are familiar with how to start and stop it.

3. Download the container image using the ID and get the image ready for use by running the following command:

```
$ sudo composer-cli compose image ${BUILDID}
```

4. Change the ownership of the downloaded container image to the current user by running the following command:

```
$ sudo chown $(whoami). ${BUILDID}-container.tar
```

5. Add read permissions for the current user to the image by running the following command:

```
$ sudo chmod a+r ${BUILDID}-container.tar
```

6. Bootstrap a server on port 8085 for the **ostree** container image to be consumed by the ISO build by completing the following steps:

   a. Get the **IMAGEID** variable result by running the following command:

```
$ IMAGEID=$(cat < "./${BUILDID}-container.tar" | sudo podman load | grep -o -P '(?
<=sha256[@:])[a-z0-9]*')
```

   b. Use the **IMAGEID** variable result to execute the podman command step by running the following command:

```
$ sudo podman run -d --name=minimal-microshift-server -p 8085:8080 ${IMAGEID}
```

   This command also returns the ID of the container saved in the **IMAGEID** variable for monitoring.

7. Generate the installer blueprint file by running the following command:

```
cat > microshift-installer.toml <<EOF
name = "microshift-installer"

description = ""
version = "0.0.0"
modules = []
groups = []
packages = []
EOF
```

4.8. ADD THE BLUEPRINT TO IMAGE BUILDER AND BUILD THE ISO

1. Add the blueprint to the Image Builder by running the following command:

```
$ sudo composer-cli blueprints push microshift-installer.toml
```

2. Start the **ostree** ISO build by running the following command:

```
$ BUILDID=$(sudo composer-cli compose start-ostree --url http://localhost:8085/repo/ --ref
"rhel/9/$(uname -m)/edge" microshift-installer edge-installer | awk '{print $2}')
```

   This command also returns the identification (ID) of the build for monitoring.

3. You can check the status of the build periodically by running the following command:

```
$ sudo composer-cli compose status
```

Example output for a running build

```
ID                                   Status    Time                  Blueprint           Version  Type
Size
c793c24f-ca2c-4c79-b5b7-ba36f5078e8d  RUNNING   Wed Jun 7 13:22:20 2023
microshift-installer  0.0.0    edge-installer
```

Example output for a completed build

```
ID                                   Status    Time                  Blueprint           Version  Type
Size
c793c24f-ca2c-4c79-b5b7-ba36f5078e8d  FINISHED  Wed Jun 7 13:34:49 2023
microshift-installer  0.0.0    edge-installer
```

4.9. DOWNLOAD THE ISO AND PREPARE IT FOR USE

1. Download the ISO using the ID by running the following command:

```
$ sudo composer-cli compose image ${BUILDID}
```

2. Change the ownership of the downloaded container image to the current user by running the following command:

```
$ sudo chown $(whoami). ${BUILDID}-installer.iso
```

3. Add read permissions for the current user to the image by running the following command:

```
$ sudo chmod a+r ${BUILDID}-installer.iso
```

Additional resources

- Creating a RHEL for Edge Container blueprint using image builder CLI
- Supported image customizations
- Building OSTree image
- Blueprint Reference
- Installing podman

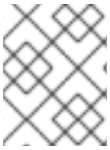
4.10. PROVISIONING A MACHINE FOR MICROSHIFT

Provision a machine with your RHEL for Edge image by using the procedures from the RHEL for Edge documentation.

To use MicroShift, you must provision the system so that it meets the following requirements:

- The machine you are provisioning must meet the system requirements for installing MicroShift.
- The file system must have a logical volume manager (LVM) volume group (VG) with sufficient capacity for the persistent volumes (PVs) of your workload.

- A pull secret from the Red Hat Hybrid Cloud Console must be present as **/etc/crio/openshift-pull-secret** and have root user-only read/write permissions.
- The firewall must be configured with the required settings.



NOTE

If you are using a Kickstart such as the RHEL for Edge Installer (ISO) image, you can update your Kickstart file to meet the provisioning requirements.

Prerequisites

1. You have created a RHEL for Edge Installer (ISO) image containing your RHEL for Edge commit with MicroShift.
   a. This requirement includes the steps of composing an RFE Container image, creating the RFE Installer blueprint, starting the RFE container, and composing the RFE Installer image.
2. Create a Kickstart file or use an existing one. In the Kickstart file, you must include:
   a. Detailed instructions about how to create a user.
   b. How to fetch and deploy the RHEL for Edge image.

For more information, read "Additional resources."

Procedure

1. In the main section of the Kickstart file, update the setup of the filesystem such that it contains an LVM volume group called **rhel** with at least 10GB system root. Leave free space for the LVMS CSI driver to use for storing the data for your workloads.

Example kickstart snippet for configuring the filesystem

```
# Partition disk such that it contains an LVM volume group called `rhel` with a
# 10GB+ system root but leaving free space for the LVMS CSI driver for storing data.
#
# For example, a 20GB disk would be partitioned in the following way:
#
# NAME          MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
# sda             8:0    0  20G  0 disk
# ├─sda1          8:1    0 200M  0 part /boot/efi
# ├─sda1          8:1    0 800M  0 part /boot
# └─sda2          8:2    0  19G  0 part
#   └─rhel-root  253:0    0  10G  0 lvm  /sysroot
#
ostreesetup --nogpg --osname=rhel --remote=edge \
--url=file:///run/install/repo/ostree/repo --ref=rhel/<RHEL VERSION NUMBER>/x86_64/edge
zerombr
clearpart --all --initlabel
part /boot/efi --fstype=efi --size=200
part /boot --fstype=xfs --asprimary --size=800
# Uncomment this line to add a SWAP partition of the recommended size
#part swap --fstype=swap --recommended
part pv.01 --grow
volgroup rhel pv.01
```

```
logvol / --vgname=rhel --fstype=xfs --size=10000 --name=root
# To add users, use a line such as the following
user --name=<YOUR_USER_NAME> \
--password=<YOUR_HASHED_PASSWORD> \
--iscrypted --groups=<YOUR_USER_GROUPS>
```

2. In the **%post** section of the Kickstart file, add your pull secret and the mandatory firewall rules.

   Example Kickstart snippet for adding the pull secret and firewall rules

```
%post --log=/var/log/anaconda/post-install.log --erroronfail

# Add the pull secret to CRI-O and set root user-only read/write permissions
cat > /etc/crio/openshift-pull-secret << EOF
YOUR_OPENSHIFT_PULL_SECRET_HERE
EOF
chmod 600 /etc/crio/openshift-pull-secret

# Configure the firewall with the mandatory rules for MicroShift
firewall-offline-cmd --zone=trusted --add-source=10.42.0.0/16
firewall-offline-cmd --zone=trusted --add-source=169.254.169.1

%end
```

3. Install the **mkksiso** tool by running the following command:

```
$ sudo yum install -y lorax
```

4. Update the Kickstart file in the ISO with your new Kickstart file by running the following command:

```
$ sudo mkksiso <your_kickstart>.ks <your_installer>.iso <updated_installer>.iso
```

Additional resources

- RHEL for Edge documentation
- System requirements for installing MicroShift
- Red Hat Hybrid Cloud Console pull secret
- Required firewall settings
- Creating a Kickstart file
- How to embed a Kickstart file into an ISO image

4.11. HOW TO ACCESS THE MICROSHIFT CLUSTER

Use the procedures in this section to access the MicroShift cluster, either from the same machine running the MicroShift service or remotely from a workstation. You can use this access to observe and administrate workloads. When using these steps, choose the **kubeconfig** file that contains the host name or IP address you want to connect with and place it in the relevant directory. As listed in each procedure, you use the OpenShift Container Platform CLI tool (**oc**) for cluster activities.

4.11.1. Accessing the MicroShift cluster locally

Use the following procedure to access the MicroShift cluster locally by using a **kubeconfig** file.

Prerequisites

- You have installed the **oc** binary.

Procedure

1. Optional: to create a ~/**.kube**/ folder if your RHEL machine does not have one, run the following command:

```
$ mkdir -p ~/.kube/
```

2. Copy the generated local access **kubeconfig** file to the ~/**.kube**/ directory by running the following command:

```
$ sudo cat /var/lib/microshift/resources/kubeadmin/kubeconfig > ~/.kube/config
```

3. Update the permissions on your ~/**.kube**/**config** file by running the following command:

```
$ chmod go-r ~/.kube/config
```

Verification

- Verify that MicroShift is running by entering the following command:

```
$ oc get all -A
```

4.11.2. Opening the firewall for remote access to the MicroShift cluster

Use the following procedure to open the firewall so that a remote user can access the MicroShift cluster. This procedure must be completed before a workstation user can access the cluster remotely.

For this procedure, **user@microshift** is the user on the MicroShift host machine and is responsible for setting up that machine so that it can be accessed by a remote user on a separate workstation.

Prerequisites

- You have installed the **oc** binary.
- Your account has cluster administration privileges.

Procedure

- As **user@microshift** on the MicroShift host, open the firewall port for the Kubernetes API server (**6443**/**tcp**) by running the following command:

```
[user@microshift]$ sudo firewall-cmd --permanent --zone=public --add-port=6443/tcp &&
sudo firewall-cmd --reload
```

Verification

- As **user@microshift**, verify that MicroShift is running by entering the following command:

```
[user@microshift]$ oc get all -A
```

4.11.3. Accessing the MicroShift cluster remotely

Use the following procedure to access the MicroShift cluster from a remote workstation by using a **kubeconfig** file.

The **user@workstation** login is used to access the host machine remotely. The **<user>** value in the procedure is the name of the user that **user@workstation** logs in with to the MicroShift host.

Prerequisites

- You have installed the **oc** binary.
- The **user@microshift** has opened the firewall from the local host.

Procedure

1. As **user@workstation**, create a **~/.kube/** folder if your RHEL machine does not have one by running the following command:

```
[user@workstation]$ mkdir -p ~/.kube/
```

2. As **user@workstation**, set a variable for the hostname of your MicroShift host by running the following command:

```
[user@workstation]$ MICROSHIFT_MACHINE=<name or IP address of MicroShift machine>
```

3. As **user@workstation**, copy the generated **kubeconfig** file that contains the host name or IP address you want to connect with from the RHEL machine running MicroShift to your local machine by running the following command:

```
[user@workstation]$ ssh <user>@$MICROSHIFT_MACHINE "sudo cat
/var/lib/microshift/resources/kubeadmin/$MICROSHIFT_MACHINE/kubeconfig" >
~/.kube/config
```

NOTE

To generate **kubeconfig** files for this step, see the "Generating additional kubeconfig files for remote access" link in the additional resources section.

1. As **user@workstation**, update the permissions on your **~/.kube/config** file by running the following command:

```
$ chmod go-r ~/.kube/config
```

Verification

- As **user@workstation**, verify that MicroShift is running by entering the following command:

```
[user@workstation]$ oc get all -A
```

Additional resources

- Generating additional kubeconfig files for remote access

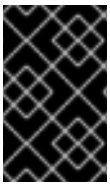
CHAPTER 5. EMBEDDING IN A RHEL FOR EDGE IMAGE FOR OFFLINE USE

Embedding MicroShift containers in an **rpm-ostree** commit means that you can run a cluster in air-gapped, disconnected, or offline environments. You can embed MicroShift containers in a Red Hat Enterprise Linux for Edge (RHEL for Edge) image so that container engines do not need to pull images over a network from a container registry. Workloads can start up immediately without network connectivity.

5.1. SYSTEM REQUIREMENTS FOR INSTALLING MICROSHIFT

The following conditions must be met prior to installing MicroShift:

- A compatible version of RHEL or RHEL for Edge.
- AArch64 or x86_64 system architecture.
- 2 CPU cores.
- 2 GB RAM for MicroShift or 3 GB RAM, required by RHEL for networked-based HTTPs or FTP installations.
- 10 GB of storage.
- You have an active MicroShift subscription on your Red Hat account. If you do not have a subscription, contact your sales representative for more information.
- You have a Logical Volume Manager (LVM) Volume Group (VG) with sufficient capacity for the Persistent Volumes (PVs) of your workload.



IMPORTANT

Secure access must be configured before running MicroShift to ensure proper functionality and security. For more information, see Using secure communications between two systems with OpenSSH.

5.2. COMPATIBILITY TABLE

Plan to pair a supported version of RHEL for Edge with the MicroShift version you are using as described in the following compatibility table:

Red Hat Device Edge release compatibility matrix

The two products of Red Hat Device Edge work together as a single solution for device-edge computing. To successfully pair your products, use the verified releases together for each as listed in the following table:

RHEL for Edge Version(s)	MicroShift Version	MicroShift Release Status	MicroShift Supported Updates
9.2, 9.3	4.15	Generally Available	4.15.0→4.15.z and 4.15→future minor version

9.2, 9.3	4.14	Generally Available	4.14.0→4.14.z and 4.14→4.15
9.2	4.13	Technology Preview	None
8.7	4.12	Developer Preview	None

5.3. EMBEDDING MICROSHIFT CONTAINERS FOR OFFLINE DEPLOYMENTS

You can use Image Builder to create **rpm-ostree** system images with embedded MicroShift container images. To embed container images, you must add the image references to your Image Builder blueprint.

Prerequisites

- You have root-user access to your build host.
- Your build host meets the Image Builder system requirements.
- You have installed and set up Image Builder and the **composer-cli** tool.
- You have created a RHEL for Edge image blueprint.
- You have installed jq.

Procedure

1. Get the exact list of container image references used by the MicroShift version you are deploying. You can either install the **microshift-release-info** RPM package by following step 2 or download and unpack the RPM by following step 3.
2. To install the **microshift-release-info** RPM package:
   a. Install the **microshift-release-info** RPM package by running the following command:

      ```
      $ sudo dnf install -y microshift-release-info-<release_version>
      ```

      Replace **<release_version>** with the numerical value of the release you are deploying, using the entire version number, such as **4.15.0**.

   b. List the contents of the **/usr/share/microshift/release** directory to verify the presence of the release information files by running the following command:

      ```
      $ ls /usr/share/microshift/release
      ```

      Example output

      ```
      release-x86_64.json
      release-aarch64.json
      ```

      If you installed the **microshift-release-info** RPM, you can proceed to step 4.

3. If you did not complete step 2, download and unpack the **microshift-release-info** RPM without installing it:

   a. Download the RPM package by running the following command:

   ```
   $ sudo dnf download microshift-release-info-<release_version>
   ```

   Replace **<release_version>** with the numerical value of the release you are deploying, using the entire version number, such as **4.15.0**.

   Example rpm

   ```
   microshift-release-info-4.15.0.*.el9.noarch.rpm (1)
   ```

   (1) The * represents the date and commit ID. Your output should contain both, for example **-202311101230.p0.g7dc6a00.assembly.4.15.0**.

   b. Unpack the RPM package without installing it by running the following command:

   ```
   $ rpm2cpio <my_microshift_release_info> | cpio -idmv (1)
   ./usr/share/microshift/release/release-aarch64.json
   ./usr/share/microshift/release/release-x86_64.json
   ```

   (1) Replace **<my_microshift_release_info>** with the name of the RPM package from the previous step.

4. Define the location of your JSON file, which contains the container reference information, by running the following command:

   ```
   $ RELEASE_FILE=</path/to/your/release-$(uname -m).json>
   ```

   Replace **</path/to/your/release-$(uname -m).json>** with the full path to your JSON file. Be sure to use the file needed for your architecture.

5. Define the location of your TOML file, which contains instructions for building the image, by running the following command:

   ```
   $ BLUEPRINT_FILE=</path/to/your/blueprint.toml>
   ```

   Replace **</path/to/your/blueprint.toml>** with the full path to your JSON file.

6. Generate and then embed the container image references in your blueprint TOML file by running the following command:

   ```
   $  jq -r '.images | .[] | ("[[containers]]\nsource = \"" + . + "\"\n")' "${RELEASE_FILE}" >>
   "${BLUEPRINT_FILE}"
   ```

   Example resulting <my_blueprint.toml> fragment showing container references

   ```
   [[containers]]
   source = "quay.io/openshift-release-dev/ocp-v4.0-art-
   dev@sha256:82cfef91557f9a70cff5a90accba45841a37524e9b93f98a97b20f6b2b69e5db"
   ```

```
[[containers]]
source = "quay.io/openshift-release-dev/ocp-v4.0-art-
dev@sha256:82cfef91557f9a70cff5a90accba45841a37524e9b93f98a97b20f6b2b69e5db"
```

7. You can manually embed any container image by adding it to the Image Builder blueprint using the following example:

Example section for manually embedding container image to Image Builder

```
[[containers]]
source = "<my_image_pullspec_with_tag_or_digest>"
```

Replace **<my_image_pullspec_with_tag_or_digest>** with the exact reference to a container image used by the MicroShift version you are deploying.

5.4. UPDATING OSBUILDER WORKER CONFIGURATION TO PREPARE FOR IMAGE BUILDING

After you have updated the blueprint, you must update the osbuilder worker configuration to prepare for building the image with embedded MicroShift containers.

Prerequisites

- You have root-user access to your build host.
- Your build host meets the Image Builder system requirements.
- You have installed and set up Image Builder and the **composer-cli** tool.

NOTE

You can create an **/etc/osbuild-worker/osbuild-worker.toml** directory and configuration file if they do not exist.

Procedure

1. Add a pull secret for authenticating to the registry by setting the **auth_file_path** in the **[containers]** section of the **/etc/osbuild-worker/osbuild-worker.toml** osbuilder worker configuration file:

```
[containers]
auth_file_path = "/etc/osbuild-worker/pull-secret.json"
```

2. Restart the **osbuild-worker** to apply configuration changes by restarting the host. Restarting the host ensures that all **osbuild-worker** services currently running are restarted.

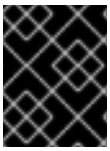
5.5. BUILD AND USE THE RPM-OSTREE IMAGE FOR OFFLINE DEPLOYMENTS

You can use Image Builder to create **rpm-ostree** system images with embedded MicroShift container images. To embed container images, you must add the image references to your Image Builder blueprint. You can create the commit and ISO as needed for your use case.

Add the prerequisites listed here to the ones that are included in the procedures that follow.

5.5.1. Additional prerequisites for offline deployments

- You have created and updated a RHEL for Edge image blueprint for offline use. The following procedures use the example of a blueprint created with container images. You must use the updated blueprint you created in the "Embedding MicroShift containers for offline deployments" procedure.
- You have updated the **/etc/osbuild-worker/osbuild-worker.toml** configuration file for offline use.



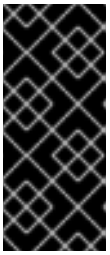
IMPORTANT

Replace **minimal-microshift.toml** in the following procedures with the name of the TOML you updated for offline use, <my_blueprint_name>.

5.5.2. Adding the MicroShift service to a blueprint

Adding the MicroShift RPM package to an Image Builder blueprint enables the build of a RHEL for Edge image with MicroShift embedded.

- Start with step 1 to create your own minimal blueprint file which results in a faster MicroShift installation.
- Start with step 2 to use the generated blueprint for installation which includes all the RPM packages and container images. This is a longer installation process, but a faster start up because container references are accessed locally.



IMPORTANT

- Replace *<microshift_blueprint.toml>* in the following procedures with the name of the TOML file you are using.
- Replace *<microshift_blueprint>* in the following procedures with the name you want to use for your blueprint.

Procedure

1. Use the following example to create your own blueprint file:

Custom Image Builder blueprint example

```
cat > <microshift_blueprint.toml> <<EOF (1)
name = "<microshift_blueprint>" (2)

description = ""
version = "0.0.1"
modules = []
groups = []
```

```
[[packages]]
name = "microshift"
version = "*"

[customizations.services]
enabled = ["microshift"]
EOF
```

1. *<microshift_blueprint.toml>* is the name of the TOML file.
2. *<microshift_blueprint>* is the name of your blueprint.

NOTE

The wildcard * in the commands uses the latest MicroShift RPMs. If you need a specific version, substitute the wildcard for the version you want. For example, insert **4.15.0** to download the MicroShift 4.15.0 RPMs.

2. Optional. Use the blueprint installed in the **/usr/share/microshift/blueprint** directory that is specific to your platform architecture. See the following example snippet for an explanation of the blueprint sections:

Generated Image Builder blueprint example snippet

```
name = "microshift_blueprint"
description = "MicroShift 4.15.1 on x86_64 platform"
version = "0.0.1"
modules = []
groups = []

[[packages]] 1
name = "microshift"
version = "4.15.1"
...
...

[customizations.services] 2
enabled = ["microshift"]

[customizations.firewall]
ports = ["22:tcp", "80:tcp", "443:tcp", "5353:udp", "6443:tcp", "30000-32767:tcp", "30000-
32767:udp"]
...
...

[[containers]] 3
source = "quay.io/openshift-release-dev/ocp-v4.0-art-
dev@sha256:f41e79c17e8b41f1b0a5a32c3e2dd7cd15b8274554d3f1ba12b2598a347475f4"

[[containers]]
source = "quay.io/openshift-release-dev/ocp-v4.0-art-
dev@sha256:dbc65f1fba7d92b36cf7514cd130fe83a9bd211005ddb23a8dc479e0eea645fd"
```

```
...
...
EOF
```

1. References for all non-optional MicroShift RPM packages using the same version compatible with the **microshift-release-info** RPM.
2. References for automatically enabling MicroShift on system startup and applying default networking settings.
3. References for all non-optional MicroShift container images necessary for an offline deployment.

3. Add the blueprint to the Image Builder by running the following command:

```
$ sudo composer-cli blueprints push <microshift_blueprint.toml> 1
```

1. Replace *<microshift_blueprint.toml>* with the name of your TOML file.

Verification

1. Verify the Image Builder configuration listing only MicroShift packages by running the following command:

```
$ sudo composer-cli blueprints depsolve <microshift_blueprint> | grep microshift 1
```

1. Replace *<microshift_blueprint>* with the name of your blueprint.

Example output

```
blueprint: microshift_blueprint v0.0.1
    microshift-greenboot-4.15.1-202305250827.p0.g4105d3b.assembly.4.15.1.el9.noarch
    microshift-networking-4.15.1-202305250827.p0.g4105d3b.assembly.4.15.1.el9.x86_64
    microshift-release-info-4.15.1-202305250827.p0.g4105d3b.assembly.4.15.1.el9.noarch
    microshift-4.15.1-202305250827.p0.g4105d3b.assembly.4.15.1.el9.x86_64
    microshift-selinux-4.15.1-202305250827.p0.g4105d3b.assembly.4.15.1.el9.noarch
```

2. Optional: Verify the Image Builder configuration listing all components to be installed by running the following command:

```
$ sudo composer-cli blueprints depsolve <microshift_blueprint> 1
```

1. Replace *<microshift_blueprint>* with the name of your blueprint.

5.5.3. Creating the RHEL for Edge image

Use the following procedure to create the ISO. The RHEL for Edge Installer image pulls the commit from the running container and creates an installable boot ISO with a Kickstart file configured to use the embedded **rpm-ostree** commit.

Prerequisites

- Your build host meets the Image Builder system requirements.
- You have installed and set up Image Builder and the **composer-cli** tool.
- You have root-user access to your build host.
- You have installed the **podman** tool.

Procedure

1. Start an **ostree** container image build by running the following command:

```
$ BUILDID=$(sudo composer-cli compose start-ostree --ref "rhel/{op-system-version-major}/$(uname -m)/edge" <microshift_blueprint> edge-container | awk '{print $2}') 1
```

1 Replace *<microshift_blueprint>* with the name of your blueprint.

   This command also returns the identification (ID) of the build for monitoring.

2. You can check the status of the build periodically by running the following command:

```
$ sudo composer-cli compose status
```

Example output of a running build

```
ID                                   Status    Time                  Blueprint             Version   Type
Size
cc3377ec-4643-4483-b0e7-6b0ad0ae6332 RUNNING   Wed Jun 7 12:26:23 2023
microshift_blueprint     0.0.1     edge-container
```

Example output of a completed build

```
ID                                   Status    Time                  Blueprint             Version   Type
Size
cc3377ec-4643-4483-b0e7-6b0ad0ae6332 FINISHED  Wed Jun 7 12:32:37 2023
microshift_blueprint  0.0.1     edge-container
```

NOTE

You can use the **watch** command to monitor your build if you are familiar with how to start and stop it.

3. Download the container image using the ID and get the image ready for use by running the following command:

```
$ sudo composer-cli compose image ${BUILDID}
```

4. Change the ownership of the downloaded container image to the current user by running the following command:

```
$ sudo chown $(whoami). ${BUILDID}-container.tar
```

5. Add read permissions for the current user to the image by running the following command:

```
$ sudo chmod a+r ${BUILDID}-container.tar
```

6. Bootstrap a server on port 8085 for the **ostree** container image to be consumed by the ISO build by completing the following steps:

   a. Get the **IMAGEID** variable result by running the following command:

```
$ IMAGEID=$(cat < "./${BUILDID}-container.tar" | sudo podman load | grep -o -P '(?
<=sha256[@:])[a-z0-9]*')
```

   b. Use the **IMAGEID** variable result to execute the podman command step by running the following command:

```
$ sudo podman run -d --name=minimal-microshift-server -p 8085:8080 ${IMAGEID}
```

   This command also returns the ID of the container saved in the **IMAGEID** variable for monitoring.

7. Generate the installer blueprint file by running the following command:

```
cat > microshift-installer.toml <<EOF
name = "microshift-installer"

description = ""
version = "0.0.0"
modules = []
groups = []
packages = []
EOF
```

5.6. ADDITIONAL RESOURCES

- Pushing a container to a registry and embedding it into an image
- Container registry credentials
- Configuring network settings for fully disconnected hosts
- Using Operator Lifecycle Manager with MicroShift
- Creating custom catalogs using the oc-mirror plugin

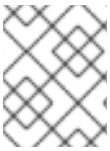
CHAPTER 6. THE GREENBOOT HEALTH CHECK FRAMEWORK

Greenboot is the generic health check framework for the **systemd** service on **rpm-ostree** systems such as Red Hat Enterprise Linux for Edge (RHEL for Edge). This framework is included in MicroShift installations with the **microshift-greenboot** and **greenboot-default-health-checks** RPM packages.

Greenboot health checks run at various times to assess system health and automate a rollback to the last healthy state in the event of software trouble, for example:

- Default health check scripts run each time the system starts.
- In addition the to the default health checks, you can write, install, and configure application health check scripts to also run every time the system starts.
- Greenboot can reduce your risk of being locked out of edge devices during updates and prevent a significant interruption of service if an update fails.
- When a failure is detected, the system boots into the last known working configuration using the **rpm-ostree** rollback capability. This feature is especially useful automation for edge devices where direct serviceability is either limited or non-existent.

A MicroShift application health check script is included in the **microshift-greenboot** RPM. The **greenboot-default-health-checks** RPM includes health check scripts verifying that DNS and **ostree** services are accessible. You can create your own health check scripts for the workloads you are running. You can write one that verifies that an application has started, for example.



NOTE

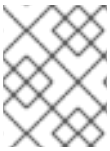
Rollback is not possible in the case of an update failure on a system not using **rpm-ostree**. This is true even though health checks might run.

6.1. HOW GREENBOOT USES DIRECTORIES TO RUN SCRIPTS

Health check scripts run from four **/etc/greenboot** directories. These scripts run in alphabetical order. Keep this in mind when you configure the scripts for your workloads.

When the system starts, greenboot runs the scripts in the **required.d** and **wanted.d** directories. Depending on the outcome of those scripts, greenboot continues the startup or attempts a rollback as follows:

1. System as expected: When all of the scripts in the **required.d** directory are successfully run, greenboot runs any scripts present in the **/etc/greenboot/green.d** directory.
2. System trouble: If any of the scripts in the **required.d** directory fail, greenboot runs any prerollback scripts present in the **red.d** directory, then restarts the system.



NOTE

Greenboot redirects script and health check output to the system log. When you are logged in, a daily message provides the overall system health output.

6.1.1. Greenboot directories details

Returning a nonzero exit code from any script means that script has failed. Greenboot restarts the system a few times to retry the scripts before attempting to roll back to the previous version.

- **/etc/greenboot/check/required.d** contains the health checks that must not fail.
    - If the scripts fail, greenboot retries them three times by default. You can configure the number of retries in the **/etc/greenboot/greenboot.conf** file by setting the **GREENBOOT_MAX_BOOTS** parameter to the desired number of retries.
    - After all retries fail, greenboot automatically initiates a rollback if one is available. If a rollback is not available, the system log output shows that manual intervention is required.
    - The **40_microshift_running_check.sh** health check script for MicroShift is installed into this directory.
- **/etc/greenboot/check/wanted.d** contains health scripts that are allowed to fail without causing the system to be rolled back.
    - If any of these scripts fail, greenboot logs the failure but does not initiate a rollback.
- **/etc/greenboot/green.d** contains scripts that run after greenboot has declared the start successful.
- **/etc/greenboot/red.d** contains scripts that run after greenboot has declared the startup as failed, including the **40_microshift_pre_rollback.sh** prerollback script. This script is executed right before a system rollback. The script performs MicroShift pod and OVN-Kubernetes cleanup to avoid potential conflicts after the system is rolled back to a previous version.

6.2. THE MICROSHIFT HEALTH CHECK SCRIPT

The **40_microshift_running_check.sh** health check script only performs validation of core MicroShift services. Install your customized workload health check scripts in the greenboot directories to ensure successful application operations after system updates. Scripts run in alphabetical order.

MicroShift health checks are listed in the following table:

Table 6.1. Validation statuses and outcome for MicroShift

Validation	Pass	Fail
Check that the script runs with **root** permissions	Next	**exit 0**
Check that the **microshift.service** is enabled	Next	**exit 0**
Wait for the **microshift.service** to be active (!failed)	Next	**exit 1**
Wait for Kubernetes API health endpoints to be working and receiving traffic	Next	**exit 1**
Wait for any pod to start	Next	**exit 1**
For each core namespace, wait for images to be pulled	Next	**exit 1**

Validation	Pass	Fail
For each core namespace, wait for pods to be ready	Next	**exit 1**
For each core namespace, check if pods are not restarting	**exit 0**	**exit 1**

6.2.1. Validation wait period

The wait period in each validation is five minutes by default. After the wait period, if the validation has not succeeded, it is declared a failure. This wait period is incrementally increased by the base wait period after each boot in the verification loop.

- You can override the base-time wait period by setting the **MICROSHIFT_WAIT_TIMEOUT_SEC** environment variable in the **/etc/greenboot/greenboot.conf** configuration file. For example, you can change the wait time to three minutes by resetting the value to 180 seconds, such as **MICROSHIFT_WAIT_TIMEOUT_SEC=180**.

6.3. ENABLING SYSTEMD JOURNAL SERVICE DATA PERSISTENCY

The default configuration of the **systemd** journal service stores the data in the volatile **/run/log/journal** directory. To view system logs across system starts and restarts, you must enable log persistence and set limits on the maximal journal data size.

Procedure

1. Make the directory by running the following command:

```
$ sudo mkdir -p /etc/systemd/journald.conf.d
```

2. Create the configuration file by running the following command:

```
cat <<EOF | sudo tee /etc/systemd/journald.conf.d/microshift.conf &>/dev/null
[Journal]
Storage=persistent
SystemMaxUse=1G
RuntimeMaxUse=1G
EOF
```

3. Edit the configuration file values for your size requirements.

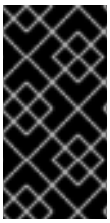
Additional resources

- Auto applying manifests

6.4. UPDATES AND THIRD-PARTY WORKLOADS

Health checks are especially useful after an update. You can examine the output of greenboot health checks and determine whether the update was declared valid. This health check can help you determine if the system is working properly.

Health check scripts for updates are installed into the **/etc/greenboot/check/required.d** directory and are automatically executed during each system start. Exiting scripts with a nonzero status means the system start is declared as failed.



IMPORTANT

Wait until after an update is declared valid before starting third-party workloads. If a rollback is performed after workloads start, you can lose data. Some third-party workloads create or update data on a device before an update is complete. Upon rollback, the file system reverts to its state before the update.

6.5. CHECKING THE RESULTS OF AN UPDATE

After a successful start, greenboot sets the variable **boot_success=** to **1** in GRUB. You can view the overall status of system health checks after an update in the system log by using the following procedure.

Procedure

- To access the overall status of system health checks, run the following command:

```
$ sudo grub2-editenv - list | grep ^boot_success
```

Example output for a successful system start

```
boot_success=1
```

6.6. ACCESSING HEALTH CHECK OUTPUT IN THE SYSTEM LOG

You can manually access the output of health checks in the system log by using the following procedure.

Procedure

- To access the results of a health check, run the following command:

```
$ sudo journalctl -o cat -u greenboot-healthcheck.service
```

Example output of a failed health check

```
...
...
Running Required Health Check Scripts...
STARTED
GRUB boot variables:
boot_success=0
boot_indeterminate=0
boot_counter=2
...
```

```
...
Waiting 300s for MicroShift service to be active and not failed
FAILURE
...
...
```

6.7. ACCESSING PREROLLBACK HEALTH CHECK OUTPUT IN THE SYSTEM LOG

You can access the output of health check scripts in the system log. For example, check the results of a prerollback script using the following procedure.

Procedure

- To access the results of a prerollback script, run the following command:

```
$ sudo journalctl -o cat -u redboot-task-runner.service
```

Example output of a prerollback script

```
...
...
Running Red Scripts...
STARTED
GRUB boot variables:
boot_success=0
boot_indeterminate=0
boot_counter=0
The ostree status:
* rhel c0baa75d9b585f3dd989a9cf05f647eb7ca27ee0dbd4b94fe8c93ed3a4b9e4a5.0
    Version: 9.1
    origin: <unknown origin type>
  rhel 6869c1347b0e0ba1bbf0be750cdf32da5138a1fcbc5a4c6325ab9eb647b64663.0 (rollback)
    Version: 9.1
    origin refspec: edge:rhel/9/x86_64/edge
System rollback imminent - preparing MicroShift for a clean start
Stopping MicroShift services
Removing MicroShift pods
Killing conmon, pause and OVN processes
Removing OVN configuration
Finished greenboot Failure Scripts Runner.
Cleanup succeeded
Script '40_microshift_pre_rollback.sh' SUCCESS
FINISHED
redboot-task-runner.service: Deactivated successfully.
```

6.8. CHECKING UPDATES WITH A HEALTH CHECK SCRIPT

Access the output of greenboot health check scripts in the system log after an update by using the following procedure.

Procedure

- To access the result of update checks, run the following command:

```
$ sudo grub2-editenv - list | grep ^boot_success
```

Example output for a successful update

```
boot_success=1
```

If your command returns **boot_success=0**, either the greenboot health check is still running, or the update is a failure.

6.9. ADDITIONAL RESOURCES

- Greenboot workload health check scripts

CHAPTER 7. TROUBLESHOOTING INSTALLATION ISSUES

To troubleshoot a failed MicroShift installation, you can run an sos report. Use the **sos report** command to generate a detailed report that shows all of the enabled plugins and data from the different components and applications in a system.

7.1. GATHERING DATA FROM AN SOS REPORT

Prerequisites

- You must have the **sos** package installed.

Procedure

1. Log into the failing host as a root user.
2. Perform the debug report creation procedure by running the following command:

```
$ microshift-sos-report
```

Example output

```
sosreport (version 4.5.1)

This command will collect diagnostic and configuration information from
this Red Hat Enterprise Linux system and installed applications.

An archive containing the collected information will be generated in
/var/tmp/sos.o0sznf_8 and may be provided to a Red Hat support
representative.

Any information provided to Red Hat will be treated in accordance with
the published support policies at:

        Distribution Website : https://www.redhat.com/
        Commercial Support   : https://www.access.redhat.com/

The generated archive may contain data considered sensitive and its
content should be reviewed by the originating organization before being
passed to any third party.

No changes will be made to system configuration.


 Setting up archive ...
 Setting up plugins ...
 Running plugins. Please wait ...

  Starting 1/2   microshift      [Running: microshift]
  Starting 2/2   microshift_ovn  [Running: microshift microshift_ovn]
  Finishing plugins              [Running: microshift]

  Finished running plugins
```

```
Found 1 total reports to obfuscate, processing up to 4 concurrently

sosreport-microshift-rhel9-2023-03-31-axjbyxw :    Beginning obfuscation...
sosreport-microshift-rhel9-2023-03-31-axjbyxw :    Obfuscation completed

Successfully obfuscated 1 report(s)

Creating compressed archive...

A mapping of obfuscated elements is available at
 /var/tmp/sosreport-microshift-rhel9-2023-03-31-axjbyxw-private_map

Your sosreport has been generated and saved in:
 /var/tmp/sosreport-microshift-rhel9-2023-03-31-axjbyxw-obfuscated.tar.xz

 Size 444.14KiB
 Owner root
 sha256 922e5ff2db25014585b7c6c749d2c44c8492756d619df5e9838ce863f83d4269

Please send this file to your support representative.
```

7.2. ADDITIONAL RESOURCES

- About MicroShift sos reports
- Generating an sos report for technical support