Red Hat build of MicroShift 4.13

Installing

Installing and configuring MicroShift clusters
Installing and configuring MicroShift clusters
Abstract

This document provides information about installing MicroShift and details about some configuration processes.
# Table of Contents

## CHAPTER 1. INSTALLING RED HAT BUILD OF MICROSHIFT FROM AN RPM PACKAGE

1.1. SYSTEM REQUIREMENTS FOR INSTALLING RED HAT BUILD OF MICROSHIFT ........................................... 3  
1.2. BEFORE INSTALLING RED HAT BUILD OF MICROSHIFT FROM AN RPM PACKAGE ................................. 3  
1.3. PREPARING TO INSTALL RED HAT BUILD OF MICROSHIFT FROM AN RPM PACKAGE ........................ 4  
1.4. INSTALLING RED HAT BUILD OF MICROSHIFT FROM AN RPM PACKAGE ........................................... 5  
1.5. STARTING THE RED HAT BUILD OF MICROSHIFT SERVICE ................................................................. 6  
1.6. STOPPING THE RED HAT BUILD OF MICROSHIFT SERVICE ................................................................. 7  
1.7. HOW TO ACCESS THE RED HAT BUILD OF MICROSHIFT CLUSTER .................................................... 7  
  1.7.1. Accessing the Red Hat build of MicroShift cluster locally ............................................................... 7  
  1.7.2. Opening the firewall for remote access to the Red Hat build of MicroShift cluster ......................... 8  
  1.7.3. Accessing the Red Hat build of MicroShift cluster remotely ............................................................ 9  

## CHAPTER 2. EMBEDDING RED HAT BUILD OF MICROSHIFT IN A RHEL FOR EDGE IMAGE

2.1. PREPARING FOR IMAGE BUILDING ........................................................................................................ 10  
2.2. ADDING RED HAT BUILD OF MICROSHIFT REPOSITORIES TO IMAGE BUILDER ............................. 10  
2.3. ADDING THE RED HAT BUILD OF MICROSHIFT SERVICE TO A BLUEPRINT ................................. 12  
2.4. CREATING THE RED HAT ENTERPRISE LINUX (RHEL) FOR EDGE IMAGE ........................................... 13  
2.5. ADD THE BLUEPRINT TO IMAGE BUILDER AND BUILD THE ISO ..................................................... 15  
2.6. DOWNLOAD THE ISO AND PREPARE IT FOR USE ................................................................................ 15  
2.7. PROVISIONING A MACHINE FOR RED HAT BUILD OF MICROSHIFT .............................................. 16  
2.8. HOW TO ACCESS THE RED HAT BUILD OF MICROSHIFT CLUSTER .................................................. 18  
  2.8.1. Accessing the Red Hat build of MicroShift cluster locally ............................................................... 18  
  2.8.2. Opening the firewall for remote access to the Red Hat build of MicroShift cluster ......................... 19  
  2.8.3. Accessing the Red Hat build of MicroShift cluster remotely ............................................................ 19  

## CHAPTER 3. THE GREENBOOT HEALTH CHECK

3.1. HOW GREENBOOT USES DIRECTORIES TO RUN SCRIPTS .................................................................. 21  
  3.1.1. Greenboot directories details ...................................................................................................... 21  
3.2. THE RED HAT BUILD OF MICROSHIFT HEALTH SCRIPT .................................................................... 22  
  3.2.1. Validation wait period ............................................................................................................ 23  
3.3. ENABLING SYSTEMD JOURNAL SERVICE DATA PERSISTENCY ......................................................... 23  
3.4. UPDATES AND THIRD-PARTY WORKLOADS .................................................................................... 23  
3.5. CHECKING THE RESULTS OF AN UPDATE .......................................................................................... 24  
3.6. ACCESSING HEALTH CHECK OUTPUT IN THE SYSTEM LOG .......................................................... 24  
3.7. ACCESSING PREROLLBACK HEALTH CHECK OUTPUT IN THE SYSTEM LOG .......................... 25  
3.8. CHECKING UPDATES WITH A HEALTH SCRIPT ............................................................................... 25
You can install Red Hat build of MicroShift from an RPM package on a machine with Red Hat Enterprise Linux (RHEL) 9.2.

**IMPORTANT**

Red Hat build of MicroShift is Technology Preview only. This Technology Preview software is not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using Red Hat build of MicroShift in production. Technology Preview provides early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

Red Hat does not support an update path from the Technology Preview version to later versions of Red Hat build of MicroShift. A new installation is necessary.

For more information about the support scope of Red Hat Technology Preview features, read [Technology Preview Features Support Scope](#).

### 1.1. SYSTEM REQUIREMENTS FOR INSTALLING RED HAT BUILD OF MICROSHIFT

The following conditions must be met prior to installing Red Hat build of MicroShift:

- RHEL 9.2
- 2 CPU cores
- 2 GB RAM for Red Hat build of MicroShift or 3 GB RAM, required by RHEL for networked-based HTTPs or FTP installations
- 10 GB of storage
- You have an active Red Hat build of MicroShift subscription on your Red Hat account. If you do not have a subscription, contact your sales representative for more information.
- You have a subscription that includes Red Hat build of MicroShift RPMs.
- You have a Logical Volume Manager (LVM) Volume Group (VG) with sufficient capacity for the Persistent Volumes (PVs) of your workload.

### 1.2. BEFORE INSTALLING RED HAT BUILD OF MICROSHIFT FROM AN RPM PACKAGE

Red Hat build of 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. PREPARING TO INSTALL RED HAT BUILD OF 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 Red Hat build of 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.

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

   ```bash
   $ sudo vgs
   ```

   Example output:

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

**NOTE**

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

**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.

1.4. INSTALLING RED HAT BUILD OF MICROSHIFT FROM AN RPM PACKAGE

Use the following procedure to install Red Hat build of MicroShift from an RPM package.

Prerequisites

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

Procedure

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

   ```bash
   $ sudo subscription-manager repos
   --enable rhocp-4.13-for-rhel-9-$uname -m-rpms
   --enable fast-datapath-for-rhel-9-$uname -m-rpms
   ```

2. Install Red Hat build of MicroShift by running the following command:

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

3. Optional: Install greenboot for Red Hat build of MicroShift by running the following command:

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

4. 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 Red Hat build of MicroShift.

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

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

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

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

7. Make the `/etc/crio/openshift-pull-secret` file readable and writeable by the root user only by running the following command:
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 Red Hat build of 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 Red Hat build of MicroShift section.

**Additional resources**

- [System requirements for installing MicroShift](#).
- [Preparing to install MicroShift from an RPM package](#).

### 1.5. STARTING THE RED HAT BUILD OF MICROSHIFT SERVICE

Use the following procedure to start the Red Hat build of MicroShift service.

**Prerequisites**

- You have installed Red Hat build of MicroShift from an RPM package.

**Procedure**

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

   ```
   $ sudo systemctl start microshift
   ```

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

   ```
   $ sudo systemctl enable microshift
   ```

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

   ```
   $ sudo systemctl disable microshift
   ```
NOTE

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

1.6. STOPPING THE RED HAT BUILD OF MICROSHIFT SERVICE

Use the following procedure to stop the Red Hat build of MicroShift service.

Prerequisites

- The Red Hat build of MicroShift service is running.

Procedure

1. Enter the following command to stop the Red Hat build of MicroShift service:

   ```
   $ sudo systemctl stop microshift
   ```

2. Workloads deployed on Red Hat build of MicroShift will continue running even after the Red Hat build of 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
   ```

1.7. HOW TO ACCESS THE RED HAT BUILD OF MICROSHIFT CLUSTER

Use the procedures in this section to access the Red Hat build of MicroShift cluster, either from the same machine running the Red Hat build of 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.

1.7.1. Accessing the Red Hat build of MicroShift cluster locally

Use the following procedure to access the Red Hat build of 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:

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

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

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

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

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

Verification

- Verify that Red Hat build of MicroShift is running by entering the following command:

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

1.7.2. Opening the firewall for remote access to the Red Hat build of MicroShift cluster

Use the following procedure to open the firewall so that a remote user can access the Red Hat build of 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 Red Hat build of 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 Red Hat build of MicroShift host, open the firewall port for the Kubernetes API server (`6443/tcp`) by running the following command:

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

Verification

- As `user@microshift`, verify that Red Hat build of MicroShift is running by entering the following command:

  ```bash
  [user@microshift]$ oc get all -A
  ```
1.7.3. Accessing the Red Hat build of MicroShift cluster remotely

Use the following procedure to access the Red Hat build of 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 Red Hat build of 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:

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

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

   ```bash
   [user@workstation]$ MICROSHIFT_MACHINE=<name or IP address of Red Hat build 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 Red Hat build of MicroShift to your local machine by running the following command:

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

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

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

**Verification**

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

  ```bash
  [user@workstation]$ oc get all -A
  ```
CHAPTER 2. EMBEDDING RED HAT BUILD OF MICROSHIFT IN A RHEL FOR EDGE IMAGE

You can embed Red Hat build of MicroShift into a Red Hat Enterprise Linux (RHEL) for Edge 9.2 image. Use this guide to build a RHEL image containing Red Hat build of MicroShift.

IMPORTANT

Red Hat build of MicroShift is Technology Preview only. This Technology Preview software is not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using Red Hat build of MicroShift in production. Technology Preview provides early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

Red Hat does not support an update path from the Technology Preview version to later versions of Red Hat build of MicroShift. A new installation is necessary.

For more information about the support scope of Red Hat Technology Preview features, read Technology Preview Features Support Scope.

2.1. PREPARING FOR IMAGE BUILDING

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

IMPORTANT

Red Hat build of MicroShift 4.13 deployments have only been tested with Red Hat Enterprise Linux (RHEL) for Edge 9.2. Other versions of RHEL are not supported.

To build an Red Hat Enterprise Linux (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.

2.2. ADDING RED HAT BUILD OF MICROSHIFT REPOSITORIES TO IMAGE BUILDER

Use the following procedure to add the Red Hat build of 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-4.13` RPM repository source required to pull Red Hat build of MicroShift RPMs by running the following command:

```
$ cat > rhocp-4.13.toml <<EOF
id = "rhocp-4.13"
name = "Red Hat OpenShift Container Platform 4.13 for RHEL 9"
type = "yum-baseurl"
url = "https://cdn.redhat.com/content/dist/layered/rhel9/$(uname -m)/rhocp/4.13/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-4.13.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-4.13
```

**Additional resources**

- [Image Builder system requirements](#)
- [Installing Image Builder](#)
2.3. ADDING THE RED HAT BUILD OF MICROSHIFT SERVICE TO A BLUEPRINT

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

Procedure

1. Use the following example to create your blueprint:

   **Image Builder blueprint example**

   ```
   $ cat > minimal-microshift.toml <<EOF
   name = "minimal-microshift"
   description = ""
   version = "0.0.1"
   modules = []
   groups = []

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

   [[packages]]
   name = "microshift-greenboot" ¹
   version = "*"

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

   ¹ Optional **microshift-greenboot** RPM. For more information, read the "Greenboot health check" guide in the "Running Applications" section.

   **NOTE**

   The wildcard * in the commands uses the latest Red Hat build of MicroShift RPMs. If you need a specific version, substitute the wildcard for the version you want. For example, insert **4.13.1** to download the Red Hat build of MicroShift 4.13.1 RPMs.

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

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

Verification

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

   ```
   $ sudo composer-cli blueprints depsolve minimal-microshift | grep microshift
   ```
2. Optional: Verify the Image Builder configuration listing all components to be installed by running the following command:

```bash
$sudo composer-cli blueprints depsolve minimal-microshift
```

### 2.4. CREATING THE RED HAT ENTERPRISE LINUX (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 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 the `podman` tool.

**Procedure**

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

```bash
$ BUILDID=$(sudo composer-cli compose start-ostree --ref "rhel/9/$(uname -m)/edge" minimal-microshift edge-container | awk '{print $2}')
```

   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:

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

**Example output of a running build**

<table>
<thead>
<tr>
<th>ID</th>
<th>Status</th>
<th>Time</th>
<th>Blueprint</th>
<th>Version</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>cc3377ec-4643-4483-b0e7-6b0ad0ae6332</td>
<td>RUNNING</td>
<td>Wed Jun 7 12:26:23 2023</td>
<td>minimal-microshift</td>
<td>0.0.1</td>
<td>edge-container</td>
</tr>
</tbody>
</table>

**Example output of a completed build**

-
### 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:

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

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

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

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

   ```bash
   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:

      ```bash
      IMAGEID=$(cat ./${BUILDID}-container.tar | 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:

      ```bash
      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:

   ```bash
   cat > microshift-installer.toml <<EOF
   name = "microshift-installer"
   description = ""
   version = "0.0.0"
   modules = []
   groups = []
   packages = []
   EOF
   ```
2.5. 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**

<table>
<thead>
<tr>
<th>ID</th>
<th>Status</th>
<th>Time</th>
<th>Blueprint</th>
<th>Version</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>c793c24f-ca2c-4c79-b5b7-ba36f5078e8d</td>
<td>RUNNING</td>
<td>Wed Jun 7 13:22:20 2023</td>
<td>microshift-installer</td>
<td>0.0.0</td>
<td>edge-installer</td>
</tr>
</tbody>
</table>

   **Example output for a completed build**

<table>
<thead>
<tr>
<th>ID</th>
<th>Status</th>
<th>Time</th>
<th>Blueprint</th>
<th>Version</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>c793c24f-ca2c-4c79-b5b7-ba36f5078e8d</td>
<td>FINISHED</td>
<td>Wed Jun 7 13:34:49 2023</td>
<td>microshift-installer</td>
<td>0.0.0</td>
<td>edge-installer</td>
</tr>
</tbody>
</table>

2.6. 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
2.7. PROVISIONING A MACHINE FOR RED HAT BUILD OF MICROSHIFT

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

To use Red Hat build of 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 Red Hat build of 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 Red Hat build of MicroShift’s required firewall 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 an RHEL for Edge Installer (ISO) image containing your RHEL for Edge commit with Red Hat build of MicroShift.
   - 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:
   - Detailed instructions about how to create a user.
   - 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**

```bash
# 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.
```
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
cat > /etc/crio/openshift-pull-secret

cat > /etc/crio/openshift-pull-secret << EOF
YOUR_OPENSHIFT_PULL_SECRET_HERE
EOF
chown 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
```
2.8. HOW TO ACCESS THE RED HAT BUILD OF MICROSHIFT CLUSTER

Use the procedures in this section to access the Red Hat build of MicroShift cluster, either from the same machine running the Red Hat build of 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.

2.8.1. Accessing the Red Hat build of MicroShift cluster locally

Use the following procedure to access the Red Hat build of 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:

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

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

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

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

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

Verification

- Verify that Red Hat build of MicroShift is running by entering the following command:

  ```bash
  $ oc get all -A
  ```
2.8.2. Opening the firewall for remote access to the Red Hat build of MicroShift cluster

Use the following procedure to open the firewall so that a remote user can access the Red Hat build of 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 Red Hat build of 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 Red Hat build of 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 Red Hat build of MicroShift is running by entering the following command:

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

2.8.3. Accessing the Red Hat build of MicroShift cluster remotely

Use the following procedure to access the Red Hat build of 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 Red Hat build of 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 Red Hat build of MicroShift host by running the following command:

   ```bash
   [user@workstation]$ MICROSHIFT_MACHINE=<name or IP address of Red Hat build 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 Red Hat build of MicroShift to your local machine by running the following command:

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

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

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

Verification

- As `user@workstation`, verify that Red Hat build of MicroShift is running by entering the following command:

  ```bash
  [user@workstation]$ oc get all -A
  ```
CHAPTER 3. THE GREENBOOT HEALTH CHECK

Greenboot is the generic health check framework for the `systemd` service on RPM-OSTree-based systems. The `microshift-greenboot` RPM and `greenboot-default-health-checks` are optional RPM packages you can install. Greenboot is used to assess system health and automate a rollback to the last healthy state in the event of software trouble.

This health check framework is especially useful when you need to check for software problems and perform system rollbacks on edge devices where direct serviceability is either limited or non-existent. When health check scripts are installed and configured, health checks run every time the system starts.

Using 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.

A Red Hat build of MicroShift 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 also create your own health check scripts based on 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 OSTree. This is true even though health checks might run.

3.1. 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 successful, 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.

3.1.1. 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 Red Hat build of 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 Red Hat build of MicroShift pod and OVN-Kubernetes cleanup to avoid potential conflicts after the system is rolled back to a previous version.

### 3.2. THE RED HAT BUILD OF MICROSHIFT HEALTH SCRIPT

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

Red Hat build of MicroShift health checks are listed in the following table:

**Table 3.1. Validation statuses and outcome for Red Hat build of MicroShift**

<table>
<thead>
<tr>
<th>Validation</th>
<th>Pass</th>
<th>Fail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check that the script runs with <strong>root</strong> permissions</td>
<td>Next</td>
<td>exit 0</td>
</tr>
<tr>
<td>Check that the <strong>microshift.service</strong> is enabled</td>
<td>Next</td>
<td>exit 0</td>
</tr>
<tr>
<td>Wait for the <strong>microshift.service</strong> to be active (if failed)</td>
<td>Next</td>
<td>exit 1</td>
</tr>
<tr>
<td>Wait for Kubernetes API health endpoints to be working and receiving traffic</td>
<td>Next</td>
<td>exit 1</td>
</tr>
<tr>
<td>Wait for any pod to start</td>
<td>Next</td>
<td>exit 1</td>
</tr>
<tr>
<td>For each core namespace, wait for images to be pulled</td>
<td>Next</td>
<td>exit 1</td>
</tr>
<tr>
<td>For each core namespace, wait for pods to be ready</td>
<td>Next</td>
<td>exit 1</td>
</tr>
</tbody>
</table>
For each core namespace, check if pods are not restarting

<table>
<thead>
<tr>
<th>Validation</th>
<th>Pass</th>
<th>Fail</th>
</tr>
</thead>
<tbody>
<tr>
<td>For each core namespace, check if pods are not restarting</td>
<td>exit 0</td>
<td>exit 1</td>
</tr>
</tbody>
</table>

3.2.1. 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`.

3.3. 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 persist 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

3.4. 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.

### 3.5. 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:

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

**Example output for a successful system start**

```
boot_success=1
```

### 3.6. 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:

  ```bash
  $ 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
```
3.7. 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:
  ```bash
  $ 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 6869c1347b0e0ba1bbf0be750cd32da5138a1fcbc5a4c6325ab9eb647b64663.0 (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.

3.8. CHECKING UPDATES WITH A HEALTH SCRIPT

Access the output of 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.