Red Hat build of MicroShift 4.16

Troubleshooting

Troubleshooting common issues

Last Updated: 2024-06-28

Red Hat build of MicroShift 4.16 Troubleshooting

Troubleshooting common issues

Abstract

Information about troubleshooting common Red Hat build of MicroShift issues.

Table of Contents

CHAPTER 1. CHECKING WHICH VERSION YOU HAVE INSTALLED

To begin troubleshooting, determine which version of Red Hat build of MicroShift you have installed.

1.1. CHECKING THE VERSION USING THE COMMAND-LINE INTERFACE

To begin troubleshooting, you must know your MicroShift version. One way to get this information is by using the CLI.

Procedure

- Run the following command to check the version information:

  ```
  $ microshift version
  ```

  Example output

  ```
  Red Hat build of MicroShift Version: 4.16-0.microshift-e6980e25
  Base OCP Version: 4.16
  ```

1.2. CHECKING THE MICROSHIFT VERSION USING THE API

To begin troubleshooting, you must know your MicroShift version. One way to get this information is by using the API.

Procedure

- To get the version number using the OpenShift CLI (**oc**), view the **kube-public/microshift-version** config map by running the following command:

  ```
  $ oc get configmap -n kube-public microshift-version -o yaml
  ```

  Example output

  ```
  apiVersion: v1
  data:
    major: "4"
    minor: "13"
    version: 4.13.8-0.microshift-fa441af87431
  kind: ConfigMap
  metadata:
    creationTimestamp: "2023-08-03T21:06:11Z"
    name: microshift-version
    namespace: kube-public
  ```

1.3. CHECKING THE ETCD VERSION

You can get the version information for the etcd database included with your MicroShift by using one or both of the following methods, depending on the level of information that you need.

Procedure

- To display the base database version information, run the following command:

  ```
  $ microshift-etcd version
  ```

  Example output

  ```
  microshift-etcd Version: 4.16.0
  Base etcd Version: 3.5.13
  ```

- To display the full database version information, run the following command:

  ```
  $ microshift-etcd version -o json
  ```

  Example output

  ```
  {
    "major": "4",
    "minor": "16",
    "gitVersion": "4.16.0~rc.1",
    "gitCommit": "140777711962eb4e0b765c39dfd325fb0abb3622",
    "gitTreeState": "clean",
    "buildDate": "2024-05-10T16:37:53Z",
    "goVersion": "go1.21.9"
    "compiler": "gc",
    "platform": "linux/amd64",
    "patch": "",
    "etcdVersion": "3.5.13"
  }
  ```

CHAPTER 2. TROUBLESHOOTING A CLUSTER

To begin troubleshooting a MicroShift cluster, first access the cluster status.

2.1. CHECKING THE STATUS OF A CLUSTER

You can check the status of a MicroShift cluster or see active pods. Given in the following procedure are three different commands you can use to check cluster status. You can choose to run one, two, or all commands to help you get the information you need to troubleshoot the cluster.

Procedure

- Check the system status, which returns the cluster status, by running the following command:

  ```
  $ sudo systemctl status microshift
  ```

  If MicroShift fails to start, this command returns the logs from the previous run.

  Example healthy output

  ```
  ● microshift.service - MicroShift
       Loaded: loaded (/usr/lib/systemd/system/microshift.service; enabled; preset: disabled)
       Active: active (running) since <day> <date> 12:39:06 UTC; 47min ago
     Main PID: 20926 (microshift)
        Tasks: 14 (limit: 48063)
       Memory: 542.9M
          CPU: 2min 41.185s
       CGroup: /system.slice/microshift.service
               └─20926 microshift run

  <Month-Day> 13:23:06 i-06166fbb376f14a8b.<hostname> microshift[20926]: kube-apiserver
  I0528 13:23:06.876001   20926 controll>
  <Month-Day> 13:23:06 i-06166fbb376f14a8b.<hostname> microshift[20926]: kube-apiserver
  I0528 13:23:06.876574   20926 controll>
  # ...
  ```

- Optional: Get comprehensive logs by running the following command:

  ```
  $ sudo journalctl -u microshift
  ```

  NOTE

  The default configuration of the **systemd** journal service stores data in a volatile directory. To persist system logs across system starts and restarts, enable log persistence and set limits on the maximum journal data size.

- Optional: If MicroShift is running, check the status of active pods by entering the following command:

  ```
  $ oc get pods -A
  ```

  Example output

```
NAMESPACE                  NAME                                                READY   STATUS
RESTARTS  AGE
default                 i-06166fbb376f14a8bus-west-2computeinternal-debug-qtwcr  1/1
Running  0      46m
kube-system             csi-snapshot-controller-5c6586d546-lprv4               1/1     Running  0
51m
kube-system             csi-snapshot-webhook-6bf8ddc7f5-kz6k9                  1/1     Running
0      51m
openshift-dns           dns-default-45jl7                                      2/2     Running  0      50m
openshift-dns           node-resolver-7wmzf                                    1/1     Running  0      51m
openshift-ingress       router-default-78b86fbf9d-qvj9s                        1/1     Running  0
51m
openshift-ovn-kubernetes    ovnkube-master-5rfhh                               4/4     Running  0
51m
openshift-ovn-kubernetes    ovnkube-node-gcnt6                                 1/1     Running  0
51m
openshift-service-ca       service-ca-bf5b7c9f8-pn6rk                          1/1     Running  0
51m
openshift-storage       topolvm-controller-549f7fbdd5-7vrmv                    5/5     Running  0
51m
openshift-storage       topolvm-node-rht2m                                     3/3     Running  0
50m
```

NOTE

This example output shows basic MicroShift. If you have installed optional RPMs, the status of pods running those services is also expected to be shown in your output.

CHAPTER 3. TROUBLESHOOTING DATA BACKUP AND RESTORE

To troubleshoot failed data backups and restorations, check the basics first, such as data paths, storage configuration, and storage capacity.

3.1. BACKING UP DATA FAILED

Data backups are automatic on **rpm-ostree** systems. If you are not using an **rpm-ostree** system and attempted to create a manual backup, the following reasons can cause the backup to fail:

- Not waiting several minutes after a system start to successfully stop MicroShift. The system must complete health checks and any other background processes before a back up can succeed.
- If MicroShift stopped running because of an error, you cannot perform a backup of the data.
  - Make sure the system is healthy.
  - Stop it in a healthy state before attempting a backup.
- If you do not have sufficient storage for the data, the backup fails. Ensure that you have enough storage for the MicroShift data.
- If you do not have sufficient permissions, a backup can fail. Ensure you have the correct user permissions to create a backup and perform the required configurations.

3.2. BACKUP LOGS

- Logs print to the terminal console during manual backups.
- Logs are automatically generated for **rpm-ostree** system automated backups as part of the MicroShift journal logs. You can check the logs by running the following command:

```
$ sudo journalctl -u microshift
```

3.3. RESTORING DATA FAILED

The restoration of data can fail for many reasons, including storage and permission issues. Mismatched data versions can cause failures when MicroShift restarts.

3.3.1. RPM-OSTree-based systems data restore failed

Data restorations are automatic on **rpm-ostree** systems, but can fail, for example:

- The only backups that are restored on **rpm-ostree** systems are backups from the current deployment or a rollback deployment. Backups are not taken on an unhealthy system.
  - Only the latest backups that have corresponding deployments are retained. Outdated backups that do not have a matching deployment are automatically removed.
  - Data is usually not restored from a newer version of MicroShift.

- Ensure that the data you are restoring follows same versioning pattern as the update path. For example, if the destination version of MicroShift is an older version than the version of the MicroShift data you are currently using, the restoration can fail.

3.3.2. RPM-based manual data restore failed

If you are using an RPM system that is not **rpm-ostree** and tried to restore a manual backup, the following reasons can cause the restoration to fail:

- If MicroShift stopped running because of an error, you cannot restore data.
  - Make sure the system is healthy.
  - Start it in a healthy state before attempting to restore data.
- If you do not have enough storage space allocated for the incoming data, the restoration fails.
  - Make sure that your current system storage is configured to accept the restored data.
- You are attempting to restore data from a newer version of MicroShift.
  - Ensure that the data you are restoring follows same versioning pattern as the update path. For example, if the destination version of MicroShift is an older version than the version of the MicroShift data you are attempting to use, the restoration can fail.

3.4. STORAGE MIGRATION FAILED

Storage migration failures are typically caused by substantial changes in custom resources (CRs) from one MicroShift to the next. If a storage migration fails, there is usually an unresolvable discrepancy between versions that requires manual review.

CHAPTER 4. TROUBLESHOOT UPDATES

To troubleshoot MicroShift updates, use the following guide.

4.1. TROUBLESHOOTING MICROSHIFT UPDATES

In some cases, MicroShift might fail to update. In these events, it is helpful to understand failure types and how to troubleshoot them.

4.1.1. Update path is blocked by version incompatibility

RPM dependency errors result if a MicroShift update is incompatible with the version of Red Hat Enterprise Linux for Edge (RHEL for Edge) or Red Hat Enterprise Linux (RHEL).

4.1.1.1. Compatibility table

Check the following compatibility table:

Red Hat Device Edge release compatibility matrix

Red Hat Enterprise Linux (RHEL) and MicroShift work together as a single solution for device-edge computing. You can update each component separately, but the product versions must be compatible. For example, an update of MicroShift from 4.14 to 4.16 requires a RHEL update. Supported configurations of Red Hat Device Edge use verified releases for each together as listed in the following table:

RHEL for Edge Version(s)	MicroShift Version	MicroShift Release Status	Supported MicroShift Version→MicroShift Version Updates
9.4	4.16	Generally Available	4.16.0→4.16.z, 4.14→4.16 and 4.15→4.16
9.2, 9.3	4.15	Generally Available	4.15.0→4.15.z, 4.14→4.15 and 4.15→4.16
9.2, 9.3	4.14	Generally Available	4.14.0→4.14.z, 4.14→4.15 and 4.14→4.16
9.2	4.13	Technology Preview	None
8.7	4.12	Developer Preview	None

4.1.1.2. Version compatibility

Check the following update paths:

Red Hat build of MicroShift update paths

- Generally Available Version 4.16.0 to 4.16.z on RHEL for Edge 9.4
- Generally Available Version 4.15.0 from RHEL 9.2 to 4.16.0 on RHEL 9.4

- Generally Available Version 4.14.0 from RHEL 9.2 to 4.16.0 on RHEL 9.4

4.1.2. OSTree update failed

If you updated on an OSTree system, the Greenboot health check automatically logs and acts on system health. A failure can be indicated by a system rollback by Greenboot. In cases where the update failed, but Greenboot did not complete a system rollback, you can troubleshoot using the RHEL for Edge documentation linked in the "Additional resources" section that follows this content.

Checking the Greenboot logs manually

- Manually check the Greenboot logs to verify system health by running the following command:

```
$ sudo systemctl restart --no-block greenboot-healthcheck && sudo journalctl -fu
greenboot-healthcheck
```

4.1.3. Manual RPM update failed

If you updated by using RPMs on a non-OSTree system, an update failure can be indicated by Greenboot, but the health checks are only informative. Checking the system logs is the next step in troubleshooting a manual RPM update failure. You can use Greenboot and **sos report** to check both the MicroShift update and the host system.

Additional resources

- Enabling **systemd** journal service data persistency
- Checking the MicroShift version
- Stopping the MicroShift service
- Starting the MicroShift service
- Composing, installing, and managing RHEL for Edge images
- Rolling back RHEL for Edge images

4.2. CHECKING JOURNAL LOGS AFTER UPDATES

In some cases, MicroShift might fail to update. In these events, it is helpful to understand failure types and how to troubleshoot them. The journal logs can assist in diagnosing update failures.

NOTE

The default configuration of the **systemd** journal service stores data in a volatile directory. To persist system logs across system starts and restarts, enable log persistence and set limits on the maximum journal data size.

Procedure

- Get comprehensive MicroShift journal logs by running the following command:

```
$ sudo journalctl -u microshift
```

- Check the Greenboot journal logs by running the following command:

```
$ sudo journalctl -u greenboot-healthcheck
```

- Examining the comprehensive logs of a specific boot uses three steps. First list the boots, then select the one you want from the list you obtained:
  - List the boots present in the journal logs by running the following command:

    ```
    $ sudo journalctl --list-boots
    ```

    Example output

    ```
    IDX  BOOT ID                          FIRST ENTRY                 LAST ENTRY
     0   681ece6f5c3047e183e9d43268c5527f  <Day> <Date> 12:27:58 UTC  <Day>
    <Date>> 13:39:41 UTC
    #....
    ```

  - Check the journal logs for the specific boot you want by running the following command:

    ```
    $ sudo journalctl --boot <-my_boot_ID> (1)
    ```

    (1) Replace ←*my-boot-ID>* with the number assigned to the specific boot that you want to check.

  - Check the journal logs for the boot of a specific service by running the following command:

    ```
    $ sudo journalctl --boot <-my_boot_ID> -u <service_name> (1) (2)
    ```

    (1) Replace ←*my-boot-ID>* with the number assigned to the specific boot that you want to check.

    (2) Replace *<service_name>* with the name of the service that you want to check.

4.3. CHECKING THE STATUS OF GREENBOOT HEALTH CHECKS

Check the status of Greenboot health checks before making changes to the system or during troubleshooting. You can use any of the following commands to help you ensure that Greenboot scripts have finished running.

Procedure

- To see a report of health check status, use the following command:

```
$ systemctl show --property=SubState --value greenboot-healthcheck.service
```

  - An output of **start** means that Greenboot checks are still running.

  - An output of **exited** means that checks have passed and Greenboot has exited. Greenboot runs the scripts in the **green.d** directory when the system is a healthy state.
  - An output of **failed** means that checks have not passed. Greenboot runs the scripts in **red.d** directory when the system is in this state and might restart the system.
- To see a report showing the numerical exit code of the service where **0** means success and non-zero values mean a failure occurred, use the following command:

```
$ systemctl show --property=ExecMainStatus --value greenboot-healthcheck.service
```

- To see a report showing a message about boot status, such as **Boot Status is GREEN - Health Check SUCCESS**, use the following command:

```
$ cat /run/motd.d/boot-status
```

CHAPTER 5. CHECKING AUDIT LOGS

You can use audit logs to identify pod security violations.

5.1. IDENTIFYING POD SECURITY VIOLATIONS THROUGH AUDIT LOGS

You can identify pod security admission violations on a workload by viewing the server audit logs. The following procedure shows you how to access the audit logs and parse them to find pod security admission violations in a workload.

Prerequisites

- You have installed **jq**.
- You have access to the cluster as a user with the **cluster-admin** role.

Procedure

1. To retrieve the node name, run the following command:

```
$ <node_name>=$(oc get node -ojsonpath='{.items[0].metadata.name}')
```

2. To view the audit logs, run the following command:

```
$ oc adm node-logs <node_name> --path=kube-apiserver/
```

Example output

```
rhel-92.lab.local audit-2023-08-18T18-25-41.663.log
rhel-92.lab.local audit-2023-08-19T11-21-29.225.log
rhel-92.lab.local audit-2023-08-20T04-16-09.622.log
rhel-92.lab.local audit-2023-08-20T21-11-41.163.log
rhel-92.lab.local audit-2023-08-21T14-06-10.402.log
rhel-92.lab.local audit-2023-08-22T06-35-10.392.log
rhel-92.lab.local audit-2023-08-22T23-26-27.667.log
rhel-92.lab.local audit-2023-08-23T16-52-15.456.log
rhel-92.lab.local audit-2023-08-24T07-31-55.238.log
```

3. To parse the affected audit logs, enter the following command:

```
$ oc adm node-logs <node_name> --path=kube-apiserver/audit.log \
  | jq -r 'select((.annotations["pod-security.kubernetes.io/audit-violations"] != null) and
(.objectRef.resource=="pods")) | .objectRef.namespace + " " + .objectRef.name + " " +
.objectRef.resource' \
  | sort | uniq -c
```

CHAPTER 6. TROUBLESHOOT ETCD

To troubleshoot etcd and improve performance, configure the memory allowance for the service.

6.1. CONFIGURING THE MEMORYLIMITMB VALUE TO SET PARAMETERS FOR THE ETCD SERVER

By default, etcd uses as much memory as necessary to handle the load on the system. In memory-constrained systems, you might need to limit the amount of memory etcd uses.

Procedure

- Edit the **/etc/microshift/config.yaml** file to set the **memoryLimitMB** value.

```
etcd:
  memoryLimitMB: 128
```

NOTE

The minimum required value for **memoryLimitMB** on MicroShift is 128 MB. Values close to the minimum value are more likely to impact etcd performance. The lower the limit, the longer etcd takes to respond to queries. If the limit is too low or the etcd usage is high, queries time out.

Verification

1. After modifying the **memoryLimitMB** value in **/etc/microshift/config.yaml**, restart MicroShift by running the following command:

```
$ sudo systemctl restart microshift
```

2. Verify the new **memoryLimitMB** value is in use by running the following command:

```
$ systemctl show --property=MemoryHigh microshift-etcd.scope
```

CHAPTER 7. RESPONSIVE RESTARTS AND SECURITY CERTIFICATES

Red Hat build of MicroShift responds to system configuration changes and restarts after alterations are detected, including IP address changes, clock adjustments, and security certificate age.

7.1. IP ADDRESS CHANGES OR CLOCK ADJUSTMENTS

MicroShift depends on device IP addresses and system-wide clock settings to remain consistent during its runtime. However, these settings may occasionally change on edge devices, such as DHCP or Network Time Protocol (NTP) updates.

When such changes occur, some MicroShift components may stop functioning properly. To mitigate this situation, MicroShift monitors the IP address and system time and restarts if either setting change is detected.

The threshold for clock changes is a time adjustment of greater than 10 seconds in either direction. Smaller drifts on regular time adjustments performed by the Network Time Protocol (NTP) service do not cause a restart.

7.2. SECURITY CERTIFICATE LIFETIME

MicroShift certificates are separated into two basic groups:

1. Short-lived certificates having certificate validity of one year.

2. Long-lived certificates having certificate validity of 10 years.

Most server or leaf certificates are short-term.

An example of a long-lived certificate is the client certificate for **system:admin user** authentication, or the certificate of the signer of the **kube-apiserver** external serving certificate.

7.2.1. Certificate rotation

Certificates that are expired or close to their expiration dates need to be rotated to ensure continued MicroShift operation. When MicroShift restarts for any reason, certificates that are close to expiring are rotated. A certificate that is set to expire imminently, or has expired, can cause an automatic MicroShift restart to perform a rotation.

NOTE

If the rotated certificate is a Certificate Authority, all of the certificates it signed rotate.

7.2.1.1. Short-term certificates

The following situations describe MicroShift actions during short-term certificate lifetimes:

1. No rotation:

   a. When a short-term certificate is up to 5 months old, no rotation occurs.

2. Rotation at restart:

   a. When a short-term certificate is 5 to 8 months old, it is rotated when MicroShift starts or

a. When a short-term certificate is 5 to 8 months old, it is rotated when MicroShift starts or restarts.

3. Automatic restart for rotation:

    a. When a short-term certificate is more than 8 months old, MicroShift can automatically restart to rotate and apply a new certificate.

7.2.1.2. Long-term certificates

The following situations describe MicroShift actions during long-term certificate lifetimes:

1. No rotation:

    a. When a long-term certificate is up to 8.5 years old, no rotation occurs.

2. Rotation at restart:

    a. When a long-term certificate is 8.5 to 9 years old, it is rotated when MicroShift starts or restarts.

3. Automatic restart for rotation:

    a. When a long-term certificate is more than 9 years old, MicroShift can automatically restart to rotate and apply a new certificate.