OpenShift Container Platform 3.11

Container-native Virtualization Release Notes

Container-native Virtualization Release Notes

Last Updated: 2019-08-02

OpenShift Container Platform 3.11 Container-native Virtualization Release Notes

Container-native Virtualization Release Notes

Legal Notice

Abstract

Known and fixed issues in this release of CNV

Table of Contents

CHAPTER 1. CONTAINER-NATIVE VIRTUALIZATION RELEASE NOTES 3
1.1. PRODUCT OVERVIEW 3
1.1.1. Introduction to Container-native Virtualization 3
1.2. NEW AND CHANGED FEATURES 3
1.2.1. Introducing Operators 3
1.2.2. Updated KubeVirt API 3
1.3. RESOLVED ISSUES 4
1.4. KNOWN ISSUES 4

CHAPTER 1. CONTAINER-NATIVE VIRTUALIZATION RELEASE NOTES

1.1. PRODUCT OVERVIEW

1.1.1. Introduction to Container-native Virtualization

Container-native Virtualization is an add-on to OpenShift Container Platform that allows virtual machine workloads to run and be managed alongside container workloads. You can create virtual machines from disk images imported using the containerized data importer (CDI) controller, or from scratch within OpenShift Container Platform.

Container-native Virtualization introduces two new objects to OpenShift Container Platform:

- **Virtual Machine**: The virtual machine in OpenShift Container Platform
- **Virtual Machine Instance**: A running instance of the virtual machine

With the Container-native Virtualization add-on, virtual machines run in pods and have the same network and storage capabilities as standard pods.

Existing virtual machine disks are imported into persistent volumes (PVs), which are made accessible to Container-native Virtualization virtual machines using persistent volume claims (PVCs). In OpenShift Container Platform, the virtual machine object can be modified or replaced as needed, without affecting the persistent data stored on the PV.

IMPORTANT

Container-native Virtualization is currently a Technology Preview feature. For details about Red Hat support for Container-native Virtualization, see the Container-native Virtualization - Technology Preview Support Policy.

Technology Preview features are not supported with Red Hat production service level agreements (SLAs), might not be functionally complete, and Red Hat does not recommend to use them for production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information on Red Hat Technology Preview features support scope, see https://access.redhat.com/support/offerings/techpreview/.

1.2. NEW AND CHANGED FEATURES

1.2.1. Introducing Operators

- Operators power the installation of KubeVirt, Containerized Data Importer, and the web-ui in Container-native Virtualization 1.4.

1.2.2. Updated KubeVirt API

- A new version of the KubeVirt API is included in Container-native Virtualization 1.4. Several important changes are reflected in the latest configuration file templates.

- The **apiVersion** has been updated from **kubevirt.io/v1alpha2** to **kubevirt.io/v1alpha3**.
- The **volumeName** attribute no longer exists. Ensure that each disk name matches the corresponding volume name in all configuration files.
- All instances of **registryDisk** must be updated to **containerDisk** in configuration files.

1.3. RESOLVED ISSUES

- The **runc** package version **runc-1.0.0-54** contained a bug that caused the virt-launcher to crash if FIPS was disabled. The version of **runc** containing the fix is now shipped with Red Hat Enterprise Linux 7 Extras. (BZ1650512)
- In the Create Virtual Machine Wizard, using the **PXE** source option with the **Start virtual machine on creation** option resulted in the boot order not changing after stopping and starting the virtual machine. This issue has been resolved. (BZ#1648245) (BZ#1647447)

1.4. KNOWN ISSUES

- CPU Manager, a feature that provides CPU pinning in OpenShift Container Platform, is currently disabled in Container-native Virtualization due to performance regressions. CPU Manager does not consider the physical CPU topology, resulting in sub-optimal pinning when hyper-threading is enabled. (BZ#1667854)
- Red Hat OpenShift Container Storage versions before 3.11.1 are not compatible with Container-native Virtualization. In the incompatible versions, Gluster nodes do not deploy with CRI-O as the container runtime. (BZ#1651270)
- When installing Container-native Virtualization 1.4, the **ansible-playbook** command fails if the **multus** image and its underlying layers are not pulled within the timeout period. As a workaround, wait a few minutes and try the command again. (BZ#1664274)
- The **virtctl image-upload** command fails if the **--uploadproxy-url** value ends with a trailing slash. If you use a custom URL, ensure that it does not end with a trailing slash before running the command. (BZ#1660888)
- The limit for compute node devices is currently 110. This limit cannot be configured, but scaling up to more than 110 devices will be supported in a future release. (BZ#1673438)
- When uploading or importing a disk image to a PVC, the space allocated for the PVC must be at least **2 * actual image size + virtual image size**. Otherwise, the virtual machine does not boot successfully. (BZ#1676824)

  NOTE

  The command **$ ls -sh <image_file>** reports the actual size. For uncompressed raw images, the command **$ ls -lh <image_file>** reports the virtual size. For QCOW2 images, use **$ qemu-img info <image_file>** to get both values.

- If you create a new DataVolume while a PVC already exists with the same name, the DataVolume enters an unrecoverable error state. If this DataVolume is associated with a virtual machine, or if it was created with the **dataVolumeTemplates** section of a virtual machine configuration file, then the virtual machine will fail to start. In these cases, the underlying DataVolume error will not be propagated to the virtual machine. (BZ#1669163)

- If a CDI import into a PVC fails, a request to delete the PVC might not work immediately. Instead, the importer pod gets stuck in a **CrashLoopBackOff** state, causing the PVC to enter a **Terminating** phase. To resolve this issue, find the importer pod associated with the PVC and delete it. The PVC will then be deleted. (BZ#1673683)

- If you use **virtctl image-upload** to upload a QCOW2 image to a PVC, the operation might fail with the error **Unexpected return value 500**, resulting in an unusable PVC. This can be caused by a bug where conversion of certain QCOW2 images during an upload operation exceeds predefined process limits. (BZ#1679134)
  To confirm that the failure was caused by this bug, check the associated **uploadserver** pod logs for a message like this:

  ```
  1 uploadserver.go:203] Saving stream failed: data stream copy failed: Local qcow to raw
  conversion failed: could not convert local qcow2 image to raw: qemu-img execution failed:
  signal: killed
  ```

  As a workaround, locally convert the file to compressed raw format and then upload the result:

  ```
  $ qemu-img convert -f qcow2 -O raw <failing-image.qcow2> image.raw
  $ gzip image.raw
  $ virtctl image-upload ... image.raw.gz
  ```

- When a virtual machine provisioned from a **URL** source is started for the first time, the virtual machine will be in the **Importing** state while Container-native Virtualization imports the container from the endpoint URL.
  Restarting a virtual machine while it is in the **Importing** state results in an error. (BZ#1673921)

- If the kubelet on a node crashes or restarts, this causes the kubelet to incorrectly report 0 KVM devices. Virtual machines are not properly scheduled on affected nodes.
  Verify the number of devices that the kubelet reports by running:

  ```
  $ oc get node $NODE | grep devices.kubevirt
  ```

  The output on an affected node shows **devices.kubevirt.io/kvm: 0**. (BZ#1681175)

  NOTE

  As a workaround, kill the relevant **virt-handler** pod on the affected node. The pod automatically restarts, and the kubelet reports the correct number of available KVM devices.

- If a virtual machine is connected to the pod network by using **bridge** mode, the virtual machine might stop if the kubelet gets restarted. (BZ#1685118)