Red Hat Hyperconverged Infrastructure for Virtualization 1.8

Maintaining Red Hat Hyperconverged Infrastructure for Virtualization

Common maintenance tasks for Red Hat Hyperconverged Infrastructure for Virtualization

Last Updated: 2023-12-05

Red Hat Hyperconverged Infrastructure for Virtualization 1.8 Maintaining Red Hat Hyperconverged Infrastructure for Virtualization

Common maintenance tasks for Red Hat Hyperconverged Infrastructure for Virtualization

Laura Bailey
lbailey@redhat.com

Legal Notice

Abstract

Red Hat Hyperconverged Infrastructure for Virtualization (RHHI for Virtualization) combines compute, storage, networking, and management capabilities into a single solution, simplifying deployment and reducing the cost of acquisition and maintenance. This document explains how to perform maintenance tasks specific to Red Hat Hyperconverged Infrastructure for Virtualization.

Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message .

PART I. CONFIGURATION TASKS

CHAPTER 1. ADD COMPUTE AND STORAGE RESOURCES

Red Hat Hyperconverged Infrastructure for Virtualization (RHHI for Virtualization) can be scaled to 6, 9, or 12 nodes.

You can add compute and storage resources in several ways:

- Add new volumes on new nodes
- Expand an existing volume across new nodes

You can also increase the space available on your existing nodes to expand storage without expanding compute resources.

- Growing a thin pool using the Web Console
- Growing a logical volume using the Web Console
- Growing the logical size of a VDO device using the Web Console

> **NOTE**
>
> OpenShift Container Storage on top of Red Hat Hyperconverged Infrastructure for Virtualization (hyperconverged nodes that host virtual machines installed with Red Hat OpenShift Container Platform) is not a supported configuration.

1.1. CREATING NEW BRICKS USING ANSIBLE

If you want to create bricks on a lot of hosts at once, you can automate the process by creating an ansible playbook. Follow this process to create and run a playbook that creates, formats, and mounts bricks for use in a hyperconverged environment.

Prerequisites

- Install the physical machines to host your new bricks.
  Follow the instructions in Installing hyperconverged hosts.

- Configure key-based SSH authentication without a password between all nodes.
  Configure this from the node that is running the Web Console to all new nodes, and from the first new node to all other new nodes.

  > **IMPORTANT**
  >
  > RHHI for Virtualization expects key-based SSH authentication without a password between these nodes for both IP addresses and FQDNs. Ensure that you configure key-based SSH authentication between these machines for the IP address and FQDN of all storage and management network interfaces.

  Follow the instructions in Using key pairs instead of passwords for SSH authentication to configure key based authentication without a password.

- Verify that your hosts do not use a Virtual Disk Optimization (VDO) layer. If you have a VDO layer, use Section 1.2, “Creating new bricks above VDO layer using ansible” instead.

Procedure

1. **Create an inventory file**
   Create a new **inventory** file in the **/etc/ansible/roles/gluster.infra/playbooks** directory using the following example.

   This file lists the hosts on which to create new bricks.

   #### Example inventory file

   ```
   [hosts]
   server4.example.com
   server5.example.com
   server6.example.com
   ```

2. **Create a bricks.yml variables file**
   Create a new **bricks.yml** file in the **/etc/ansible/roles/gluster.infra/playbooks** directory using the following example.

   This file defines the underlying storage infrastructure and settings to be created or used on each host.

   #### Example bricks.yml variable file

   ```
   # gluster_infra_disktype
   # Set a disk type. Options: JBOD, RAID6, RAID10 - Default: JBOD
   gluster_infra_disktype: RAID10

   # gluster_infra_dalign
   # Dataalignment, for JBOD default is 256K if not provided.
   # For RAID{6,10} dataalignment is computed by multiplying
   # gluster_infra_diskcount and gluster_infra_stripe_unit_size.
   gluster_infra_dalign: 256K

   # gluster_infra_diskcount
   # Required only for RAID6 and RAID10.
   gluster_infra_diskcount: 10

   # gluster_infra_stripe_unit_size
   # Required only in case of RAID6 and RAID10. Stripe unit size always in KiB, do
   # not provide the trailing `K' in the value.
   gluster_infra_stripe_unit_size: 128

   # gluster_infra_volume_groups
   # Variables for creating volume group
   gluster_infra_volume_groups:
     - { vgname: 'vg_vdb', pvname: '/dev/vdb' }
     - { vgname: 'vg_vdc', pvname: '/dev/vdc' }

   # gluster_infra_thick_lvs
   # Variable for thick lv creation
   gluster_infra_thick_lvs:
    - { vgname: 'vg_vdb', lvname: 'vg_vdb_thicklv1', size: '10G' }

   # gluster_infra_thinpools
   ```

```
# thinpoolname is optional, if not provided `vgname' followed by _thinpool is
# used for name. poolmetadatasize is optional, default 16G is used
gluster_infra_thinpools:
  - {vgname: 'vg_vdb', thinpoolname: 'foo_thinpool', thinpoolsize: '10G', poolmetadatasize:
'1G' }
  - {vgname: 'vg_vdc', thinpoolname: 'bar_thinpool', thinpoolsize: '20G', poolmetadatasize:
'1G' }

# gluster_infra_lv_logicalvols
# Thinvolumes for the brick. `thinpoolname' is optional, if omitted `vgname'
# followed by _thinpool is used
gluster_infra_lv_logicalvols:
   - { vgname: 'vg_vdb', thinpool: 'foo_thinpool', lvname: 'vg_vdb_thinlv', lvsize: '500G' }
   - { vgname: 'vg_vdc', thinpool: 'bar_thinpool', lvname: 'vg_vdc_thinlv', lvsize: '500G' }

# Setting up cache using SSD disks
gluster_infra_cache_vars:
   - { vgname: 'vg_vdb', cachedisk: '/dev/vdd',
       cachethinpoolname: 'foo_thinpool', cachelvname: 'cachelv',
       cachelvsize: '20G', cachemetalvname: 'cachemeta',
       cachemetalvsize: '100M', cachemode: 'writethrough' }

# gluster_infra_mount_devices
gluster_infra_mount_devices:
   - { path: '/rhgs/thicklv', vgname: 'vg_vdb', lvname: 'vg_vdb_thicklv1' }
   - { path: '/rhgs/thinlv1', vgname: 'vg_vdb', lvname: 'vg_vdb_thinlv' }
   - { path: '/rhgs/thinlv2', vgname: 'vg_vdc', lvname: 'vg_vdc_thinlv' }
```

IMPORTANT

If the **path:** defined does not begin with **/rhgs** the bricks are not detected automatically by the Administration Portal. Synchronize the host storage after running the **create_brick.yml** playbook to add the new bricks to the Administration Portal.

3. **Create a create_brick.yml playbook file**
   Create a new **create_brick.yml** file in the **/etc/ansible/roles/gluster.infra/playbooks** directory using the following example.

   This file defines the work involved in creating a brick using the **gluster.infra** role and the variable file you created above.

Example create_brick.yml playbook file

```
---
- name: Create a GlusterFS brick on the servers
  remote_user: root
  hosts: all
  gather_facts: false
  vars_files:
    - bricks.yml

  roles:
    - gluster.infra
```

4. **Execute the playbook**
   Run the following command from the **/etc/ansible/roles/gluster.infra/playbooks** directory to run the playbook you created using the inventory and the variables files you defined above.

   ```
   # ansible-playbook -i inventory create_brick.yml
   ```

5. **Verify that your bricks are available**

   a. Click **Compute** → **Hosts** and select the host.

   b. Click **Storage Devices** and check the list of storage devices for your new bricks.
      If you cannot see your new bricks, click **Sync** and wait for them to appear in the list of storage devices.

1.2. CREATING NEW BRICKS ABOVE VDO LAYER USING ANSIBLE

If you want to create bricks on a lot of hosts at once, you can automate the process by creating an ansible playbook.

Prerequisites

- Install the physical machines to host your new bricks.
  Follow the instructions in Installing hyperconverged hosts.

- Configure key-based SSH authentication without a password between all nodes.
  Configure this from the node that is running the Web Console to all new nodes, and from the first new node to all other new nodes.

  > **IMPORTANT**
  >
  > RHHI for Virtualization expects key-based SSH authentication without a password between these nodes for both IP addresses and FQDNs. Ensure that you configure key-based SSH authentication between these machines for the IP address and FQDN of all storage and management network interfaces.

  Follow the instructions in Using key pairs instead of passwords for SSH authentication to configure key based authentication without a password.

- Verify that your hosts use a Virtual Disk Optimization (VDO) layer. If you do not have a VDO layer, use Section 1.1, "Creating new bricks using ansible" instead.

Procedure

1. **Create an inventory file**
   Create a new **inventory** file in the **/etc/ansible/roles/gluster.infra/playbooks** directory using the following example.

   This file lists the hosts on which to create new bricks.

   Example inventory file

   ```
   [hosts]
   server4.example.com
   server5.example.com
   ```

```
server6.example.com
```

2. **Create a bricks.yml variables file**
   Create a new **bricks.yml** file in the **/etc/ansible/roles/gluster.infra/playbooks** directory using the following example.

   This file defines the underlying storage infrastructure and settings to be created or used on each host.

   Example vdo_bricks.yml variable file

```
# gluster_infra_disktype
# Set a disk type. Options: JBOD, RAID6, RAID10 - Default: JBOD
gluster_infra_disktype: RAID10

# gluster_infra_dalign
# Dataalignment, for JBOD default is 256K if not provided.
# For RAID{6,10} dataalignment is computed by multiplying
# gluster_infra_diskcount and gluster_infra_stripe_unit_size.
gluster_infra_dalign: 256K

# gluster_infra_diskcount
# Required only for RAID6 and RAID10.
gluster_infra_diskcount: 10

# gluster_infra_stripe_unit_size
# Required only in case of RAID6 and RAID10. Stripe unit size always in KiB, do
# not provide the trailing `K' in the value.
gluster_infra_stripe_unit_size: 128

# VDO creation
gluster_infra_vdo:
   - { name: 'hc_vdo_1', device: '/dev/vdb' }
   - { name: 'hc_vdo_2', device: '/dev/vdc' }

# gluster_infra_volume_groups
# Variables for creating volume group
gluster_infra_volume_groups:
   - { vgname: 'vg_vdb', pvname: '/dev/mapper/hc_vdo_1' }
   - { vgname: 'vg_vdc', pvname: '/dev/mapper/hc_vdo_2' }

# gluster_infra_thick_lvs
# Variable for thick lv creation
gluster_infra_thick_lvs:
  - { vgname: 'vg_vdb', lvname: 'vg_vdb_thicklv1', size: '10G' }

# gluster_infra_thinpools
# thinpoolname is optional, if not provided `vgname' followed by _thinpool is
# used for name. poolmetadatasize is optional, default 16G is used
gluster_infra_thinpools:
  - {vgname: 'vg_vdb', thinpoolname: 'foo_thinpool', thinpoolsize: '10G', poolmetadatasize:
'1G' }
  - {vgname: 'vg_vdc', thinpoolname: 'bar_thinpool', thinpoolsize: '20G', poolmetadatasize:
'1G' }

# gluster_infra_lv_logicalvols
# Thinvolumes for the brick. `thinpoolname' is optional, if omitted `vgname'
# followed by _thinpool is used
gluster_infra_lv_logicalvols:
  - { vgname: 'vg_vdb', thinpool: 'foo_thinpool', lvname: 'vg_vdb_thinlv', lvsize: '500G' }
  - { vgname: 'vg_vdc', thinpool: 'bar_thinpool', lvname: 'vg_vdc_thinlv', lvsize: '500G' }

# gluster_infra_mount_devices
gluster_infra_mount_devices:
  - { path: '/rhgs/thicklv', vgname: 'vg_vdb', lvname: 'vg_vdb_thicklv1' }
  - { path: '/rhgs/thinlv1', vgname: 'vg_vdb', lvname: 'vg_vdb_thinlv' }
  - { path: '/rhgs/thinlv2', vgname: 'vg_vdc', lvname: 'vg_vdc_thinlv' }
```

IMPORTANT

If the **path:** defined does not begin with **/rhgs** the bricks are not detected automatically by the Administration Portal. Synchronize the host storage after running the **create_brick.yml** playbook to add the new bricks to the Administration Portal.

3. **Create a create_brick.yml playbook file**
   Create a new **create_brick.yml** file in the **/etc/ansible/roles/gluster.infra/playbooks** directory using the following example.

   This file defines the work involved in creating a brick using the **gluster.infra** role and the variable file you created above.

   #### Example create_brick.yml playbook file

   ```
   ---
   - name: Create a GlusterFS brick on the servers
     remote_user: root
     hosts: all
     gather_facts: false
     vars_files:
       - vdo_bricks.yml

     roles:
       - gluster.infra
   ```

4. **Execute the playbook**
   Run the following command from the **/etc/ansible/roles/gluster.infra/playbooks** directory to run the playbook you created using the inventory and the variables files you defined above.

   ```
   # ansible-playbook -i inventory create_brick.yml
   ```

5. **Verify that your bricks are available**

   a. Click **Compute → Hosts** and select the host.

   b. Click **Storage Devices** and check the list of storage devices for your new bricks.
      If you cannot see your new bricks, click **Sync** and wait for them to appear in the list of storage devices.

1.3. EXPANDING VOLUME FROM RED HAT VIRTUALIZATION MANAGER

Follow this section to expand an existing volume across new bricks on new hyperconverged nodes.

Prerequisites

- Verify that your scaling plans are supported: Requirements for scaling.
- Install three physical machines to serve as the new hyperconverged nodes. Follow the instructions in Installing hyperconverged hosts.
- Configure key-based SSH authentication without a password. Configure this from the node that is running the Web Console to all new nodes, and from the first new node to all other new nodes.

  IMPORTANT

  RHHI for Virtualization expects key-based SSH authentication without a password between these nodes for both IP addresses and FQDNs. Ensure that you configure key-based SSH authentication between these machines for the IP address and FQDN of all storage and management network interfaces.

  Follow the instructions in Using key pairs instead of passwords for SSH authentication to configure key based authentication without a password.

Procedure

1. **Create new bricks**
   Create the bricks on the servers you want to expand your volume across by following the instructions in Creating bricks using ansible or Creating bricks above a VDO layer using ansible depending on your requirements.

   IMPORTANT

   If the **path:** defined does not begin with **/rhgs** the bricks are not detected automatically by the Administration Portal. Synchronize the host storage after running the **create_brick.yml** playbook to synchronize the new bricks to the Administration Portal.

   1. Click **Compute → Hosts** and select the host.
   2. Click **Storage Devices**.
   3. Click **Sync**.

   Repeat for each host that has new bricks.

2. **Add new bricks to the volume**

   a. Log in to RHV Administration Console.

   b. Click **Storage → Volumes** and select the volume to expand.

c. Click the **Bricks** tab.

d. Click **Add**. The *Add Bricks* window opens.

e. Add new bricks.

   i. Select the brick host from the **Host** dropdown menu.

   ii. Select the brick to add from the **Brick Directory** dropdown menu and click **Add**.

f. When all bricks are listed, click **OK** to add bricks to the volume.

The volume automatically syncs the new bricks.

1.4. EXPANDING THE HYPERCONVERGED CLUSTER BY ADDING A NEW VOLUME ON NEW NODES USING THE WEB CONSOLE

Follow these instructions to use the Web Console to expand your hyperconverged cluster with a new volume on new nodes.

Prerequisites

- Verify that your scaling plans are supported: Requirements for scaling.
- Install three physical machines to serve as the new hyperconverged nodes.
  Follow the instructions in Installing hyperconverged hosts.
- Configure key-based SSH authentication without a password.
  Configure this from the node that is running the Web Console to all new nodes, and from the first new node to all other new nodes.

  IMPORTANT

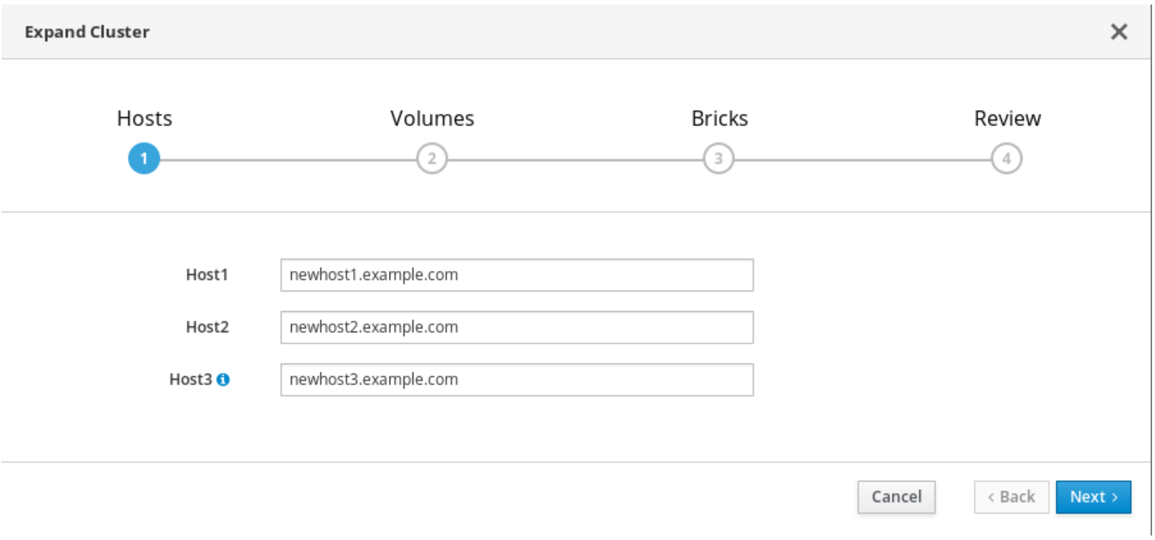
  RHHI for Virtualization expects key-based SSH authentication without a password between these nodes for both IP addresses and FQDNs. Ensure that you configure key-based SSH authentication between these machines for the IP address and FQDN of all storage and management network interfaces.

  Follow the instructions in Using key pairs instead of passwords for SSH authentication to configure key based authentication without a password.

Procedure

1. Log in to the Web Console.
2. Click **Virtualization → Hosted Engine** and then click **Manage Gluster**.
3. Click **Expand Cluster**. The *Gluster Deployment* window opens.

   a. On the *Hosts* tab, enter the FQDN or IP address of the new hyperconverged nodes and click **Next**.

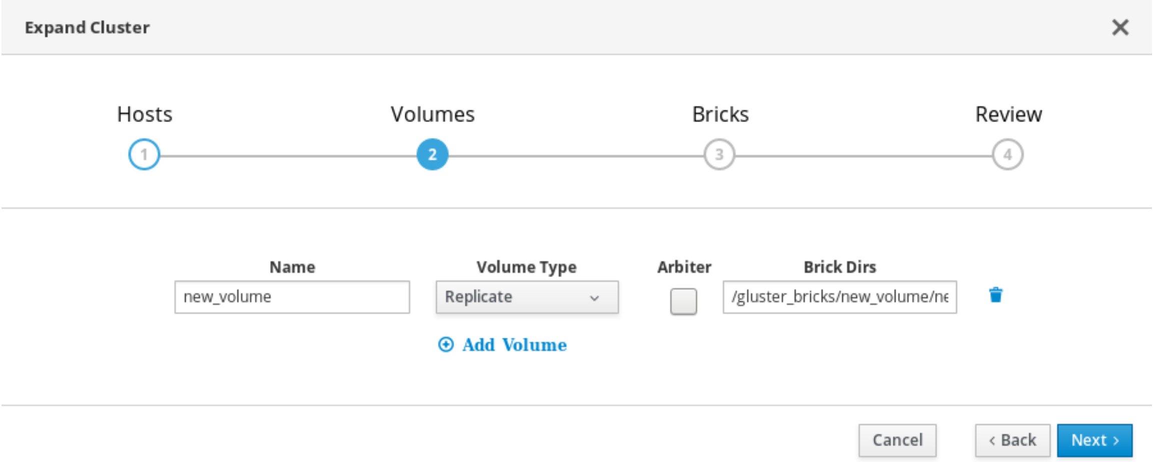


Host1: newhost1.example.com

Host2: newhost2.example.com

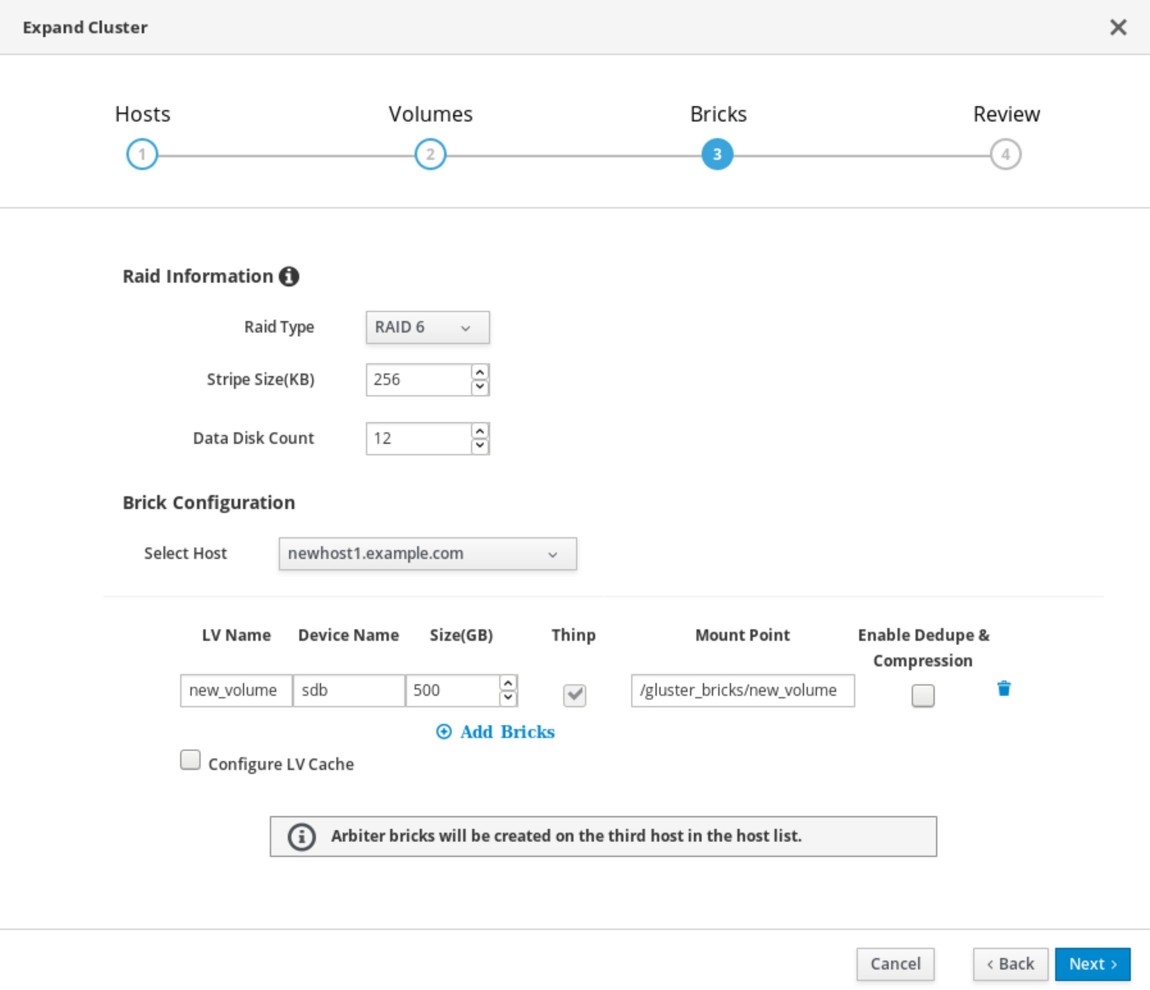
Host3: newhost3.example.com

b. On the *Volumes* tab, specify the details of the volume you want to create.



Name	Volume Type	Arbiter	Brick Dirs
new_volume	Replicate	□	/gluster_bricks/new_volume/ne

c. On the *Bricks* tab, specify the details of the disks to be used to create the Gluster volume.



Raid Information

Raid Type: RAID 6

Stripe Size(KB): 256

Data Disk Count: 12

Brick Configuration

Select Host: newhost1.example.com

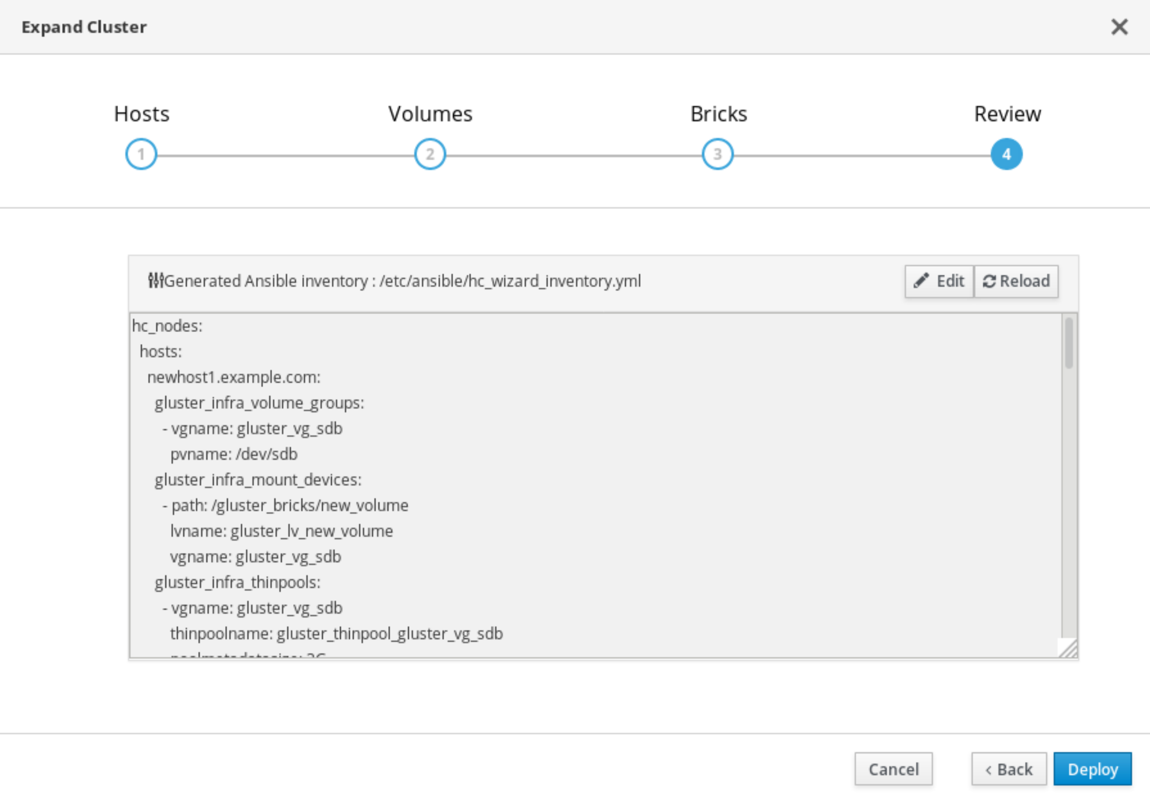
LV Name	Device Name	Size(GB)	Thinp	Mount Point	Enable Dedupe & Compression
new_volume	sdb	500	☑	/gluster_bricks/new_volume	□

⊕ Add Bricks

□ Configure LV Cache

Arbiter bricks will be created on the third host in the host list.

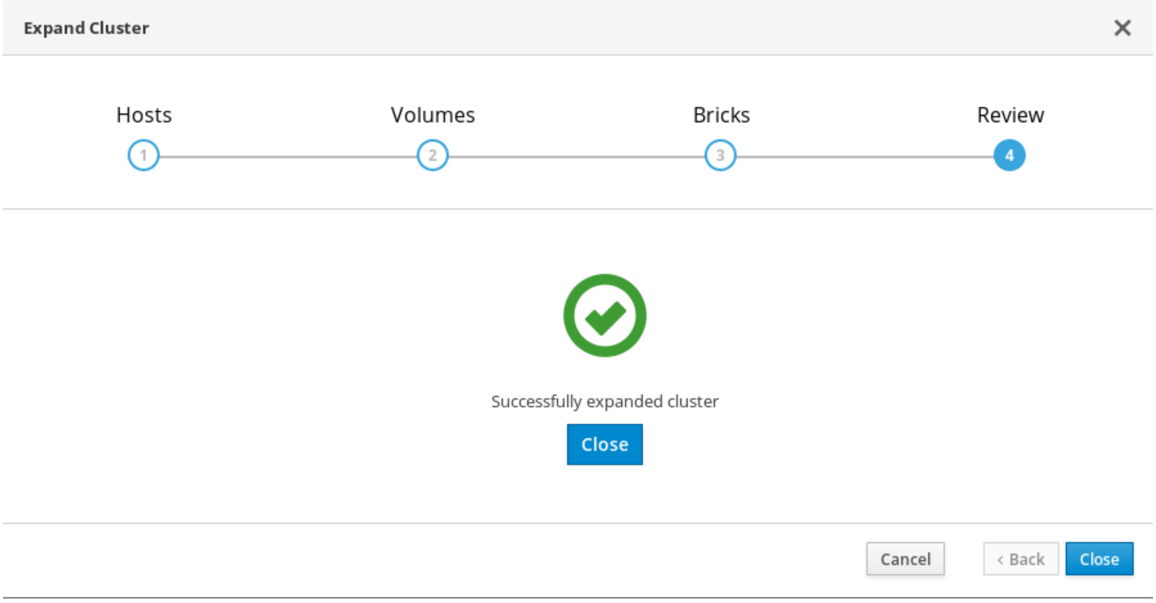
d. On the *Review* tab, check the generated file for any problems. When you are satisfied, click **Deploy**.



Generated Ansible inventory : /etc/ansible/hc_wizard_inventory.yml

```
hc_nodes:
 hosts:
  newhost1.example.com:
   gluster_infra_volume_groups:
    - vgname: gluster_vg_sdb
     pvname: /dev/sdb
   gluster_infra_mount_devices:
    - path: /gluster_bricks/new_volume
     lvname: gluster_lv_new_volume
     vgname: gluster_vg_sdb
   gluster_infra_thinpools:
    - vgname: gluster_vg_sdb
     thinpoolname: gluster_thinpool_gluster_vg_sdb
```

Deployment takes some time to complete. The following screen appears when the cluster has been successfully expanded.



1.4.1. Configure additional hyperconverged hosts

If your environment uses IPv6 addresses, or if you did not specify additional hyperconverged hosts as part of Configure Red Hat Gluster Storage for Hosted Engine using the Web Console , follow these steps in the Administration Portal for each of the other hyperconverged hosts.

1. Click **Compute → Hosts** and then click **New** to open the *New Host* window.
2. Provide the **Name**, **Hostname**, and **Password** for the host that you want to manage.
3. Under **Advanced Parameters**, uncheck the **Automatically configure host firewall** checkbox, as firewall rules are already configured by the deployment process.
4. In the **Hosted Engine** tab of the *New Host* dialog, set the value of **Choose hosted engine deployment action** to **Deploy**. This ensures that the hosted engine can run on the new host.
5. Click **OK**.
6. **Attach the gluster network to all remaining hosts**
   a. Click the name of the newly added host to go to the host page.
   b. Click the **Network Interfaces** subtab and then click **Setup Host Networks**.
   c. Drag and drop the newly created network to the correct interface.
   d. Ensure that the **Verify connectivity** checkbox is checked.
   e. Ensure that the **Save network configuration** checkbox is checked.
   f. Click **OK** to save.
7. In the **General** subtab for this host, verify that the value of **Hosted Engine HA** is *Active*, with a positive integer as a score.

IMPORTANT

If **Score** is listed as **N/A**, you may have forgotten to select the **deploy** action for **Choose hosted engine deployment action**. Follow the steps in Reinstalling a hyperconverged host in *Maintaining Red Hat Hyperconverged Infrastructure for Virtualization* to reinstall the host with the **deploy** action.

8. **Verify the health of the network**
   Click the **Network Interfaces** tab and check the state of the host’s network. If the network interface enters an "Out of sync" state or does not have an IP Address, click **Management → Refresh Capabilities**.

See the Red Hat Virtualization 4.4 *Self-Hosted Engine Guide* for further details: https://access.redhat.com/documentation/en-us/red_hat_virtualization/4.4/html/self-hosted_engine_guide/chap-installing_additional_hosts_to_a_self-hosted_environment

CHAPTER 2. CONFIGURE HIGH AVAILABILITY USING FENCING POLICIES

Fencing allows a cluster to enforce performance and availability policies and react to unexpected host failures by automatically rebooting hyperconverged hosts.

Several policies specific to Red Hat Gluster Storage must be enabled to ensure that fencing activities do not disrupt storage services in a Red Hat Hyperconverged (RHHI for Virtualization) Infrastructure deployment.

This requires enabling and configuring fencing at both the cluster level and at the host level. See the following sections for details.

2.1. CONFIGURING FENCING POLICIES IN THE CLUSTER

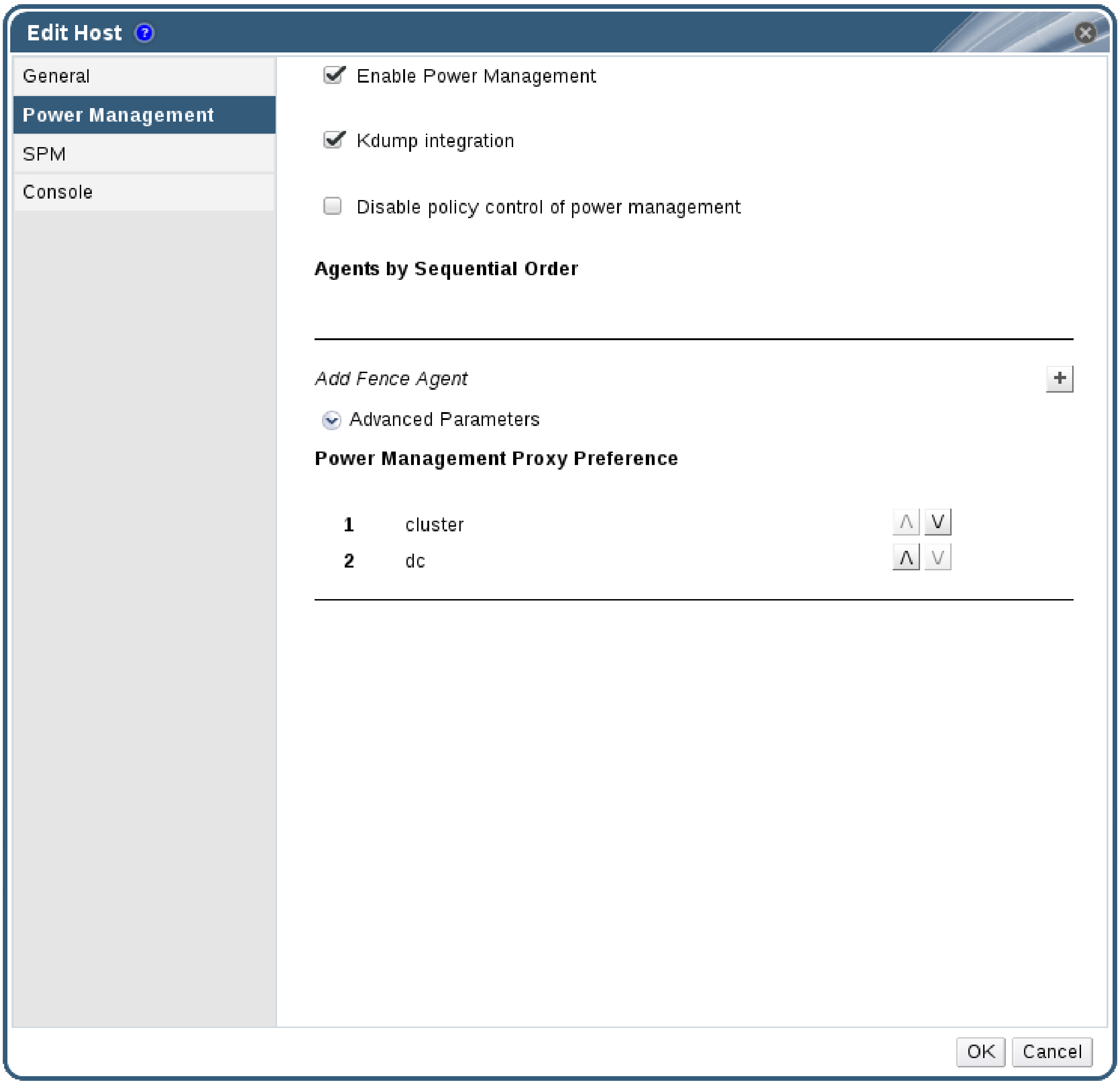
1. In the Administration Portal, click **Compute → Clusters**.
2. Select the cluster and click **Edit**. The *Edit Cluster* window opens.
3. Click the **Fencing policy** tab.
4. Check the **Enable fencing** checkbox.
5. Check the checkboxes for at least the following fencing policies:
    - Skip fencing if gluster bricks are up
    - Skip fencing if gluster quorum not met

    See Appendix B, *Fencing Policies for Red Hat Gluster Storage* for details on the effects of these policies.
6. Click **OK** to save settings.

2.2. CONFIGURING FENCING PARAMETERS ON THE HOSTS

1. In the Administration Portal, click **Compute → Hosts**.
2. Select the host to configure, and click **Edit** to open the **Edit Host** window.
3. Click the **Power Management** tab.

Figure 2.1. Power Management Settings



☑ Enable Power Management

☑ Kdump integration

□ Disable policy control of power management

Agents by Sequential Order

Power Management Proxy Preference

1 cluster

2 dc

4. Check the **Enable Power Management** check box. This enables other fields on the tab.

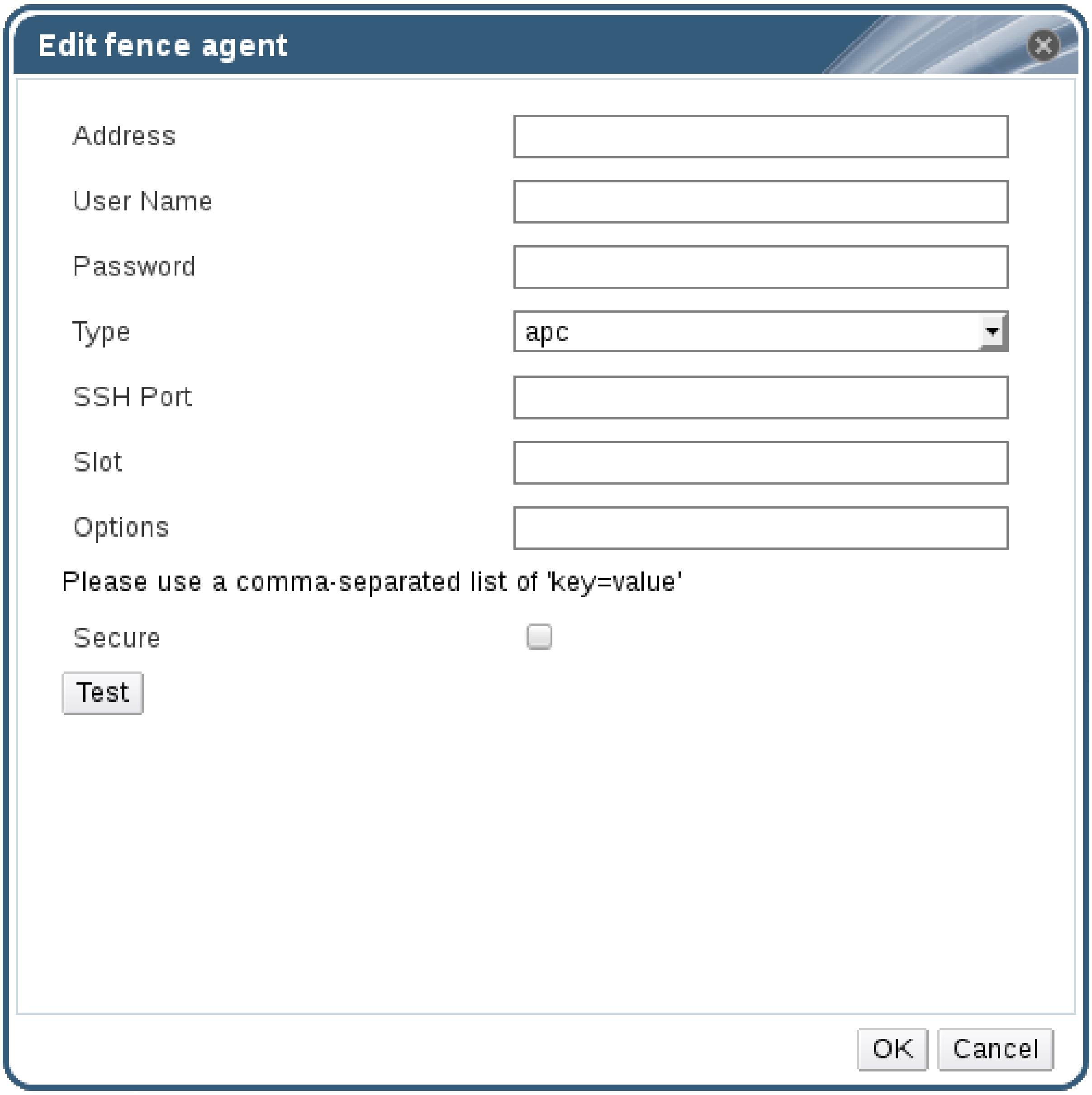
5. Check the **Kdump integration** check box to prevent the host from fencing while performing a kernel crash dump.

IMPORTANT

When you enable Kdump integration on an existing host, the host must be reinstalled for kdump to be configured. See Chapter 11, *Reinstalling a hyperconverged host* for instructions on reinstalling a host.

1. Click the plus (+) button to add a new power management device. The **Edit fence agent** window opens.

Figure 2.2. Edit fence agent



Type: apc

Please use a comma-separated list of 'key=value'

a. Enter the **Address**, **User Name**, and **Password** of the power management device.

b. Select the power management device **Type** from the drop-down list.

a. Enter the **SSH Port** number used by the power management device to communicate with the host.

b. Enter the **Slot** number used to identify the blade of the power management device.

c. Enter the **Options** for the power management device. Use a comma-separated list of *key=value* entries.

d. Check the **Secure** check box to enable the power management device to connect securely to the host.

e. Click the **Test** button to ensure the settings are correct. *Test Succeeded, Host Status is: on* displays upon successful verification.

> **WARNING**
>
> Power management parameters (userid, password, options, etc.) are tested by Red Hat Virtualization Manager in two situations: during setup, and when parameter values are manually changed in Red Hat Virtualization Manager. If you choose to ignore alerts about incorrect parameters, or if the parameters are changed on the power management hardware without the corresponding change in Red Hat Virtualization Manager, fencing is likely to fail.

f. Click **OK** to finish adding the fence agent.

1. Click **OK** to save your host configuration.

You are returned to the list of hosts. Note that the exclamation mark (**!**) next to the host's name has now disappeared, signifying that power management has been successfully configured.

CHAPTER 3. CONFIGURING DATA BACKUP AND RECOVERY OPTIONS

This chapter explains how to add disaster recovery capabilities to your Red Hat Hyperconverged Infrastructure for Virtualization deployment so that you can restore your cluster to a working state after a disk or server failure.

3.1. PREREQUISITES

3.1.1. Prerequisites for geo-replication

Be aware of the following requirements and limitations when configuring geo-replication:

Two different managers required

The source and destination volumes for geo-replication must be managed by different instances of Red Hat Virtualization Manager.

3.1.2. Prerequisites for failover and failback configuration

Versions must match between environments

Ensure that the primary and secondary environments have the same version of Red Hat Virtualization Manager, with identical data center compatibility versions, cluster compatibility versions, and PostgreSQL versions.

No virtual machine disks in the hosted engine storage domain

The storage domain used by the hosted engine virtual machine is not failed over, so any virtual machine disks in this storage domain will be lost.

Execute Ansible playbooks manually from a separate machine

Generate and execute Ansible playbooks manually from a separate machine that acts as an Ansible controller node. This node must have the **ovirt-ansible-collection** package, which provides all required disaster recovery Ansible roles.

NOTE

The **ovirt-ansible-collection** package is installed with the Hosted Engine virtual machine by default. However, during a disaster that affects the primary site, this virtual machine may be down. It is safe to use a machine that is outside the primary site to run this playbook, but for testing purposes these playbooks can be triggered from the Hosted Engine virtual machine.

3.2. SUPPORTED BACKUP AND RECOVERY CONFIGURATIONS

There are two supported ways to add disaster recovery capabilities to your Red Hat Hyperconverged Infrastructure for Virtualization deployment.

Configure backing up to a secondary volume only

Regularly synchronizing your data to a remote secondary volume helps to ensure that your data is not lost in the event of disk or server failure.

This option is suitable if the following statements are true of your deployment.

- You require only a backup of your data for disaster recovery.
- You do not require highly available storage.
- You do not want to maintain a secondary cluster.
- You are willing to manually restore your data and reconfigure your backup solution after a failure has occurred.

Follow the instructions in Configuring backup to a secondary volume to configure this option.

Configure failing over to and failing back from a secondary cluster

This option provides failover and failback capabilities in addition to backing up data on a remote volume. Configuring failover of your primary cluster's operations and storage domains to a secondary cluster helps to ensure that your data remains available in event of disk or server failure in the primary cluster.
This option is suitable if the following statements are true of your deployment.

- You require highly available storage.
- You are willing to maintain a secondary cluster.
- You do not want to manually restore your data or reconfigure your backup solution after a failure has occurred.

Follow the instructions in Configuring failover to and failback from a secondary cluster to configure this option.

Red Hat recommends that you configure at least a backup volume for production deployments.

3.3. CONFIGURING BACKUP TO A SECONDARY VOLUME

This section covers how to back up a gluster volume to a secondary gluster volume using geo-replication.

To do this, you must:

1. Ensure that all prerequisites are met.
2. Create a suitable volume to use as a geo-replication target .
3. Configure a geo-replication session between the source volume and the target volume.
4. Schedule the geo-replication process.

3.3.1. Prerequisites

3.3.1.1. Enable shared storage on the source volume

Ensure that the volume you want to back up (the source volume) has shared storage enabled. Run the following command on any server that hosts the source volume to enable shared storage.

```
# gluster volume set all cluster.enable-shared-storage enable
```

Ensure that a gluster volume named **gluster_shared_storage** is created in the source cluster, and is mounted at **/run/gluster/shared_storage** on all the nodes in the source cluster. See Setting Up Shared Storage for further information.

3.3.1.2. Match network protocol

Ensure that all hosts use the same Internet Protocol version.

If the hosts for your source volume use IPv4, the hosts for the target volume must also use IPv4.

If the hosts for your source volume use IPv6, the hosts for the target volume must use IPv6. Additionally, configure geo-replication using FQDNs instead of IPv6 addresses to avoid Bug 1855965.

3.3.2. Create a suitable target volume for geo-replication

Prepare a secondary gluster volume to hold the geo-replicated copy of your source volume. This target volume should be in a separate cluster, hosted at a separate site, so that the risk of source and target volumes being affected by the same outages is minimised.

Ensure that the target volume for geo-replication has sharding enabled. Run the following command on any node that hosts the target volume to enable sharding on that volume.

```
# gluster volume set <volname> features.shard enable
```

3.3.3. Configuring geo-replication for backing up volumes

3.3.3.1. Creating a geo-replication session

A geo-replication session is required to replicate data from an active source volume to a passive target volume.

> **IMPORTANT**
>
> Only rsync based geo-replication is supported with Red Hat Hyperconverged Infrastructure for Virtualization.

1. Create a common **pem pub** file.
   Run the following command on a source node that has key-based SSH authentication without a password configured to the target nodes.

   ```
   # gluster system:: execute gsec_create
   ```

2. Create the geo-replication session
   Run the following command to create a geo-replication session between the source and target volumes, using the created **pem pub** file for authentication.

   ```
   # gluster volume geo-replication <SOURCE_VOL> <TARGET_NODE>::<TARGET_VOL>
   create push-pem
   ```

   For example, the following command creates a geo-replication session from a source volume **prodvol** to a target volume called **backupvol**, which is hosted by **backup.example.com**.

```
# gluster volume geo-replication prodvol backup.example.com::backupvol create push-pem
```

By default this command verifies that the target volume is a valid target with available space. You can append the **force** option to the command to ignore failed verification.

3. Configure a meta-volume
   This relies on the source volume having shared storage configured, as described in Prerequisites.

```
# gluster volume geo-replication <SOURCE_VOL> <TARGET_HOST>::<TARGET_VOL>
config use_meta_volume true
```

IMPORTANT

Do not start the geo-replication session. Starting the geo-replication session begins replication from your source volume to your target volume.

3.3.3.2. Verifying creation of a geo-replication session

1. Log in to the Administration Portal on any source node.
2. Click **Storage** → **Volumes**.
3. Check the **Info** column for the geo-replication icon.
   If this icon is present, geo-replication has been configured for that volume.

   If this icon is not present, try synchronizing the volume.

3.3.3.3. Synchronizing volume state using the Administration Portal

1. Log in to the Administration Portal.
2. Click **Storage** → **Volumes**.
3. Select the volume that you want to synchronize.
4. Click the **Geo-replication** sub-tab.
5. Click **Sync**.

3.3.4. Scheduling regular backups using geo-replication

1. Log in to the Administration Portal on any source node.
2. Click **Storage** → **Domains**.
3. Click the name of the storage domain that you want to back up.
4. Click the **Remote Data Sync Setup** subtab.
5. Click **Setup**.
   The *Setup Remote Data Synchronization* window opens.
   a. In the **Geo-replicated to** field, select the backup target.

b. In the **Recurrence** field, select a recurrence interval type.
   Valid values are **WEEKLY** with at least one weekday checkbox selected, or **DAILY**.

c. In the **Hours** and **Minutes** field, specify the time to start synchronizing.

> NOTE
>
> This time is based on the Hosted Engine's timezone.

d. Click **OK**.

6. Check the **Events** subtab for the source volume at the time you specified to verify that synchronization works correctly.

3.4. CONFIGURING FAILOVER TO AND FAILBACK FROM A SECONDARY CLUSTER

This section covers how to configure your cluster to fail over to a remote secondary cluster in the event of server failure.

To do this, you must:

1. Configure backing up to a remote volume .
2. Create a suitable cluster to use as a failover target .
3. Prepare a mapping file for the source and target clusters.
4. Prepare a failover playbook .
5. Prepare a cleanup playbook for the primary cluster.
6. Prepare a failback playbook .

3.4.1. Creating a secondary cluster for failover

Install and configure a secondary cluster that can be used in place of the primary cluster in the event of failure.

This secondary cluster can be either of the following configurations:

Red Hat Hyperconverged Infrastructure

See Deploying Red Hat Hyperconverged Infrastructure for details.

Red Hat Gluster Storage configured for use as a Red Hat Virtualization storage domain

See Configuring Red Hat Virtualization with Red Hat Gluster Storage for details. Note that creating a storage domain is not necessary for this use case; the storage domain is imported as part of the failover process.

The storage on the secondary cluster must not be attached to a data center, so that it can be added to the secondary site's data center during the failover process.

3.4.2. Creating a mapping file between source and target clusters

Follow this section to create a file that maps the storage in your source cluster to the storage in your target cluster.

Red Hat recommends that you create this file immediately after you first deploy your storage, and keep it up to date as your deployment changes. This helps to ensure that everything in your cluster fails over safely in the event of disaster.

1. Create a playbook to generate the mapping file.
   Create a playbook that passes information about your cluster to the **ovirt.ovirt.disaster_recovery** role, using the **site**, **username**, **password**, and **ca** variables.

   Example playbook file: dr-ovirt-setup.yml

   ```
   ---
   - name: Collect mapping variables
     hosts: localhost
     connection: local

     vars:
       site: https://example.engine.redhat.com/ovirt-engine/api
       username: admin@internal
       password: my_password
       ca: /etc/pki/ovirt-engine/ca.pem
       var_file: disaster_recovery_vars.yml

     roles:
       - ovirt.ovirt.disaster_recovery
   ```

2. Generate the mapping file by running the playbook with the **generate_mapping** tag.

   ```
   # ansible-playbook dr-ovirt-setup.yml --tags="generate_mapping"
   ```

   This creates the mapping file, **disaster_recovery_vars.yml**.

3. Edit **disaster_recovery_vars.yml** and add information about the secondary cluster. Ensure that you only mention storage domains that have data synchronized to the secondary site; other storage domains can be removed.
   See Appendix A: Mapping File Attributes in the Red Hat Virtualization *Disaster Recovery Guide* for detailed information about attributes used in the mapping file.

3.4.3. Creating a failover playbook between source and target clusters

Create a playbook file to handle failover.

1. Define a password file (for example **passwords.yml**) to store the Manager passwords for the primary and secondary site. For example:

   Example passwords.yml file

   ```
   ---
   # This file is in plain text, if you want to
   # encrypt this file, please execute following command:
   #
   # $ ansible-vault encrypt passwords.yml
   #
   ```

```
# It will ask you for a password, which you must then pass to
# ansible interactively when executing the playbook.
#
# $ ansible-playbook myplaybook.yml --ask-vault-pass
#
dr_sites_primary_password: primary_password
dr_sites_secondary_password: secondary_password
```

NOTE

For extra security you can encrypt the password file. However, you will need to use the --ask-vault-pass parameter when running the playbook. See Working with files encrypted using Ansible Vault for more information.

2. Create a playbook file that passes the lists of hyperconverged hosts to use as a failover source and target to the **ovirt.ovirt.disaster_recovery** role, using the **dr_target_host** and **dr_source_map** variables.

   Example playbook file: dr-rhv-failover.yml

```
---
- name: Failover RHV
  hosts: localhost
  connection: local
  vars:
    dr_target_host: secondary
    dr_source_map: primary
  vars_files:
    - disaster_recovery_vars.yml
    - passwords.yml
  roles:
    - ovirt.ovirt.disaster_recovery
```

For information about executing failover, see Failing over to a secondary cluster .

3.4.4. Creating a failover cleanup playbook for your primary cluster

Create a playbook file that cleans up your primary cluster so that you can use it as a failback target.

Example playbook file: dr-cleanup.yml

```
---
- name: Clean RHV
  hosts: localhost
  connection: local
  vars:
    dr_source_map: primary
  vars_files:
    - disaster_recovery_vars.yml
  roles:
    - ovirt.ovirt.disaster_recovery
```

For information about executing failback, see Failing back to a primary cluster .

3.4.5. Create a failback playbook between source and target clusters

Create a playbook file that passes the lists of hyperconverged hosts to use as a failback source and target to the **ovirt.ovirt.disaster_recovery** role, using the **dr_target_host** and **dr_source_map** variables.

Example playbook file: dr-rhv-failback.yml

```
---
- name: Failback RHV
  hosts: localhost
  connection: local
  vars:
    dr_target_host: primary
    dr_source_map: secondary
  vars_files:
    - disaster_recovery_vars.yml
    - passwords.yml
  roles:
    - ovirt.ovirt.disaster_recovery
```

For information about executing failback, see Failing back to a primary cluster .

CHAPTER 4. CONFIGURE PERFORMANCE IMPROVEMENTS

Some deployments benefit from additional configuration to achieve optimal performance. This section covers recommended additional configuration for certain deployments.

4.1. IMPROVING VOLUME PERFORMANCE BY CHANGING SHARD SIZE

The default value of the **shard-block-size** parameter changed from **4MB** to **64MB** between Red Hat Hyperconverged Infrastructure for Virtualization version 1.0 and 1.1. This means that all new volumes are created with a **shard-block-size** value of 64MB. However, existing volumes retain the original **shard-block-size** value of 4MB.

There is no safe way to modify the **shard-block-size** value on volumes that contain data. Because shard block size applies only to writes that occur after the value is set, attempting to change the value on a volume that contains data results in a mixed shard block size, which results in poor performance.

This section shows you how to safely modify the shard block size on an existing volume after upgrading to Red Hat Hyperconverged Infrastructure for Virtualization 1.1 or higher, in order to take advantage of the performance benefits of a larger shard size.

4.1.1. Changing shard size on replicated volumes

1. **Create an inventory file**
   Create an inventory file called **normal_replicated_inventory.yml** based on the following example.

   Replace **host1**, **host2**, and **host3** with the FQDNs of your hosts, and edit device details to match your environment.

   Example normal_replicated_inventory.yml inventory file

```
hc_nodes:
  hosts:
    # Host1
    host1:
      # Dedupe & Compression config
      # If logicalsize >= 1000G then slabsize=32G else slabsize=2G
      #gluster_infra_vdo:
      #   - { name: 'vdo_sdb', device: '/dev/sdb', logicalsize: '3000G', emulate512: 'on', slabsize:
'32G',
      #       blockmapcachesize:  '128M', readcachesize: '20M', readcache: 'enabled',
writepolicy: 'auto' }

      # With Dedupe & Compression
      #gluster_infra_volume_groups:
      #  - vgname: <volgroup_name>
      #    pvname: /dev/mapper/vdo_sdb

      # Without Dedupe & Compression
      gluster_infra_volume_groups:
        - vgname: <volgroup_name>
          pvname: /dev/sdb

      gluster_infra_mount_devices:
        - path: <brick_mountpoint>
```

```
          lvname: <lv_name>
          vgname: <volgroup_name>

      gluster_infra_thinpools:
        - {vgname: '<volgroup_name>', thinpoolname: 'thinpool_<volgroup_name>',
thinpoolsize: '500G', poolmetadatasize: '4G'}

      gluster_infra_lv_logicalvols:
        - vgname: <volgroup_name>
          thinpool: thinpool_<volgroup_name>
          lvname: <lv_name>
          lvsize: <size>G

      # Mount the devices
      gluster_infra_mount_devices:
         - { path: '<brick_mountpoint>', vgname: <volgroup_name>, lvname: <lv_name> }

    #Host2
    host2:
      # Dedupe & Compression config
      # If logicalsize >= 1000G then slabsize=32G else slabsize=2G
      #gluster_infra_vdo:
      #   - { name: 'vdo_sdb', device: '/dev/sdb', logicalsize: '3000G', emulate512: 'on', slabsize:
'32G',
      #      blockmapcachesize:  '128M', readcachesize: '20M', readcache: 'enabled',
writepolicy: 'auto' }

      # With Dedupe & Compression
      #gluster_infra_volume_groups:
      #  - vgname: <volgroup_name>
      #    pvname: /dev/mapper/vdo_sdb

      # Without Dedupe & Compression
      gluster_infra_volume_groups:
        - vgname: <volgroup_name>
          pvname: /dev/sdb

      gluster_infra_mount_devices:
        - path: <brick_mountpoint>
          lvname: <lv_name>
          vgname: <volgroup_name>

      gluster_infra_thinpools:
        - {vgname: '<volgroup_name>', thinpoolname: 'thinpool_<volgroup_name>',
thinpoolsize: '500G', poolmetadatasize: '4G'}

      gluster_infra_lv_logicalvols:
        - vgname: <volgroup_name>
          thinpool: thinpool_<volgroup_name>
          lvname: <lv_name>
          lvsize: <size>G

      # Mount the devices
      gluster_infra_mount_devices:
         - { path: '<brick_mountpoint>', vgname: <volgroup_name>, lvname: <lv_name> }
    #Host3
    host3:
      # Dedupe & Compression config
      # If logicalsize >= 1000G then slabsize=32G else slabsize=2G
      #gluster_infra_vdo:
      #   - { name: 'vdo_sdb', device: '/dev/sdb', logicalsize: '3000G', emulate512: 'on', slabsize:
'32G',
      #       blockmapcachesize:  '128M', readcachesize: '20M', readcache: 'enabled',
writepolicy: 'auto' }

      # With Dedupe & Compression
      #gluster_infra_volume_groups:
      #  - vgname: <volgroup_name>
      #    pvname: /dev/mapper/vdo_sdb

      # Without Dedupe & Compression
      gluster_infra_volume_groups:
        - vgname: <volgroup_name>
          pvname: /dev/sdb

      gluster_infra_mount_devices:
        - path: <brick_mountpoint>
          lvname: <lv_name>
          vgname: <volgroup_name>

      gluster_infra_thinpools:
        - {vgname: '<volgroup_name>', thinpoolname: 'thinpool_<volgroup_name>',
thinpoolsize: '500G', poolmetadatasize: '4G'}

      gluster_infra_lv_logicalvols:
        - vgname: <volgroup_name>
          thinpool: thinpool_<volgroup_name>
          lvname: <lv_name>
          lvsize: <size>G

      # Mount the devices
      gluster_infra_mount_devices:
         - { path: '<brick_mountpoint>', vgname: <volgroup_name>, lvname: <lv_name> }

  # Common configurations
  vars:
    cluster_nodes:
       - host1
       - host2
       - host3
    gluster_features_hci_cluster: "{{ cluster_nodes }}"
    gluster_features_hci_volumes:
       - { volname: 'data', brick: '<brick_mountpoint>' }
    gluster_features_hci_volume_options:
      {
        group: 'virt',
        storage.owner-uid: '36',
        storage.owner-gid: '36',
        network.ping-timeout: '30',
        performance.strict-o-direct: 'on',
        network.remote-dio: 'off',
        cluster.granular-entry-heal: 'enable',
        features.shard-block-size: '64MB'
      }
```

2. **Create the normal_replicated.yml playbook**
   Create a **normal_replicated.yml** playbook file using the following example:

   Example normal_replicated.yml playbook

```
---

# Safely changing the shard block size parameter value for normal replicated volume
- name: Changing the shard block size
  hosts: hc_nodes
  remote_user: root
  gather_facts: no
  any_errors_fatal: true

  roles:
     - gluster.infra
     - gluster.features
```

3. **Run the playbook**

```
ansible-playbook -i normal_replicated_inventory.yml normal_replicated.yml
```

4.1.2. Changing shard size on arbitrated volumes

1. **Create an inventory file**
   Create an inventory file called **arbitrated_replicated_inventory.yml** based on the following example.

   Replace **host1**, **host2**, and **host3** with the FQDNs of your hosts, and edit device details to match your environment.

   Example arbitrated_replicated_inventory.yml inventory file

```
hc_nodes:
  hosts:
    # Host1
    host1:
      # Dedupe & Compression config
      # If logicalsize >= 1000G then slabsize=32G else slabsize=2G
      #gluster_infra_vdo:
      #   - { name: 'vdo_sdb', device: '/dev/sdb', logicalsize: '3000G', emulate512: 'on', slabsize:
'32G',
      #      blockmapcachesize:  '128M', readcachesize: '20M', readcache: 'enabled',
writepolicy: 'auto' }

      # With Dedupe & Compression
      #gluster_infra_volume_groups:
      #  - vgname: <volgroup_name>
      #    pvname: /dev/mapper/vdo_sdb
      # Without Dedupe & Compression
      gluster_infra_volume_groups:
        - vgname: <volgroup_name>
          pvname: /dev/sdb

      gluster_infra_mount_devices:
        - path: <brick_mountpoint>
          lvname: <lv_name>
          vgname: <volgroup_name>
        - path: <brick_mountpoint>
          lvname: <lv_name>
          vgname: <volgroup_name>

      gluster_infra_thinpools:
        - {vgname: '<volgroup_name>', thinpoolname: 'thinpool_<volgroup_name>',
thinpoolsize: '500G', poolmetadatasize: '4G'}

      gluster_infra_lv_logicalvols:
        - vgname: <volgroup_name>
          thinpool: thinpool_<volgroup_name>
          lvname: <lv_name>
          lvsize: <size>G
        - vgname: <volgroup_name>
          thinpool: thinpool_<volgroup_name>
          lvname: <lv_name>
          lvsize: <size>G

      # Mount the devices
      gluster_infra_mount_devices:
         - { path: '<brick_mountpoint>', vgname: <volgroup_name>, lvname: <lv_name> }
         - { path: '<brick_mountpoint>', vgname: <volgroup_name>, lvname: <lv_name> }

    #Host2
    host2:
      # Dedupe & Compression config
      # If logicalsize >= 1000G then slabsize=32G else slabsize=2G
      #gluster_infra_vdo:
      #   - { name: 'vdo_sdb', device: '/dev/sdb', logicalsize: '3000G', emulate512: 'on', slabsize:
'32G',
      #      blockmapcachesize:  '128M', readcachesize: '20M', readcache: 'enabled',
writepolicy: 'auto' }

      # With Dedupe & Compression
      #gluster_infra_volume_groups:
      #  - vgname: <volgroup_name>
      #    pvname: /dev/mapper/vdo_sdb

      # Without Dedupe & Compression
      gluster_infra_volume_groups:
        - vgname: <volgroup_name>
          pvname: /dev/sdb

      gluster_infra_mount_devices:
        - path: <brick_mountpoint>
          lvname: <lv_name>
          vgname: <volgroup_name>
        - path: <brick_mountpoint>
          lvname: <lv_name>
          vgname: <volgroup_name>

      gluster_infra_thinpools:
        - {vgname: '<volgroup_name>', thinpoolname: 'thinpool_<volgroup_name>',
thinpoolsize: '500G', poolmetadatasize: '4G'}

      gluster_infra_lv_logicalvols:
        - vgname: <volgroup_name>
          thinpool: thinpool_<volgroup_name>
          lvname: <lv_name>
          lvsize: <size>G
        - vgname: <volgroup_name>
          thinpool: thinpool_<volgroup_name>
          lvname: <lv_name>
          lvsize: <size>G

      # Mount the devices
      gluster_infra_mount_devices:
         - { path: '<brick_mountpoint>', vgname: <volgroup_name>, lvname: <lv_name> }
         - { path: '<brick_mountpoint>', vgname: <volgroup_name>, lvname: <lv_name> }

    #Host3
    host3:
      # Dedupe & Compression config
      # If logicalsize >= 1000G then slabsize=32G else slabsize=2G
      #gluster_infra_vdo:
      #   - { name: 'vdo_sdb', device: '/dev/sdb', logicalsize: '3000G', emulate512: 'on', slabsize:
'32G',
      #      blockmapcachesize:  '128M', readcachesize: '20M', readcache: 'enabled',
writepolicy: 'auto' }

      # With Dedupe & Compression
      #gluster_infra_volume_groups:
      #  - vgname: <volgroup_name>
      #    pvname: /dev/mapper/vdo_sdb

      # Without Dedupe & Compression
      gluster_infra_volume_groups:
        - vgname: <volgroup_name>
          pvname: /dev/sdb

      gluster_infra_mount_devices:
        - path: <brick_mountpoint>
          lvname: <lv_name>
          vgname: <volgroup_name>

      gluster_infra_thinpools:
        - {vgname: '<volgroup_name>', thinpoolname: 'thinpool_<volgroup_name>',
thinpoolsize: '500G', poolmetadatasize: '4G'}

      gluster_infra_lv_logicalvols:
        - vgname: <volgroup_name>
          thinpool: thinpool_<volgroup_name>
          lvname: <lv_name>
        lvsize: <size>G

    # Mount the devices
    gluster_infra_mount_devices:
      - { path: '<brick_mountpoint>', vgname: <volgroup_name>, lvname: <lv_name> }

# Common configurations
vars:
  cluster_nodes:
    - host1
    - host2
    - host3
  gluster_features_hci_cluster: "{{ cluster_nodes }}"
  gluster_features_hci_volumes:
    - { volname: 'data_one', brick: '<brick_mountpoint>', arbiter: 1 }
  gluster_features_hci_volume_options:
    {
      group: 'virt',
      storage.owner-uid: '36',
      storage.owner-gid: '36',
      network.ping-timeout: '30',
      performance.strict-o-direct: 'on',
      network.remote-dio: 'off',
      cluster.granular-entry-heal: 'enable',
      features.shard-block-size: '64MB',
    }
```

2. **Create thearbitrated_replicated.yml playbook**
   Create a **arbitrated_replicated.yml** playbook file using the following example:

   Example arbitrated_replicated.yml playbook

```
---

# Safely changing the shard block size parameter value for arbitrated replicated volume
- name: Changing the shard block size
  hosts: hc_nodes
  remote_user: root
  gather_facts: no
  any_errors_fatal: true

  roles:
     - gluster.infra
     - gluster.features
```

3. **Run the playbook**

```
ansible-playbook -i arbitrated_replicated_inventory.yml arbitrated_replicated.yml
```

4.2. CONFIGURING A LOGICAL VOLUME CACHE (LVMCACHE) FOR AN EXISTING VOLUME

If your main storage devices are not Solid State Disks (SSDs), Red Hat recommends configuring a logical volume cache (lvmcache) to achieve the required performance for Red Hat Hyperconverged Infrastructure for Virtualization deployments.

1. **Create inventory file**
   Create an inventory file called **cache_inventory.yml** based on the example below.

   Replace **<host1>**, **<host2>**, and **<host3>** with the FQDNs of the hosts on which to configure the cache.

   Replace the following values throughout the file.

   <slow_device>,<fast_device>

   Specify the device to which the cache should attach, followed by the cache device, as a comma-delimited list, for example, **cachedisk: '/dev/sdb,/dev/sde'**.

   <fast_device_name>

   Specify the name of the cache logical volume to create, for example, **cachelv_thinpool_gluster_vg_sde**

   <fast_device_thinpool>

   Specify the name of the cache thin pool to create, for example, **gluster_thinpool_gluster_vg_sde**.

   ### Example cache_inventory.yml file

```
hc_nodes:
  hosts:
    # Host1
    <host1>:
      gluster_infra_cache_vars:
        - vgname: gluster_vg_sdb
          cachedisk: '<slow_device>,<fast_device>'
          cachelvname: <fast_device_name>
          cachethinpoolname: <fast_device_thinpool>
          cachelvsize: '10G'
          cachemode: writethrough

    #Host2
    <host2>:
      gluster_infra_cache_vars:
        - vgname: gluster_vg_sdb
          cachedisk: '<slow_device>,<fast_device>'
          cachelvname: <fast_device_name>
          cachethinpoolname: <fast_device_thinpool>
          cachelvsize: '10G'
          cachemode: writethrough

    #Host3
    <host3>:
      gluster_infra_cache_vars:
        - vgname: gluster_vg_sdb
          cachedisk: '<slow_device>,<fast_device>'
          cachelvname: <fast_device_name>
          cachethinpoolname: <fast_device_thinpool>
          cachelvsize: '10G'
          cachemode: writethrough
```

2. **Create a playbook file**
   Create an ansible playbook file named **lvm_cache.yml**.

   ### Example lvm_cache.yml file

   ```
   ---
   # Create LVM Cache
   - name: Setup LVM Cache
     hosts: hc_nodes
     remote_user: root
     gather_facts: no
     any_errors_fatal: true

     roles:
       - gluster.infra
   ```

3. **Run the playbook with thecachesetup tag**
   Run the following command to apply the configuration specified in **lvm_cache.yml** to the hosts and devices specified in **cache_inventory.yml**.

   ```
   ansible-playbook -i cache_inventory.yml lvm_cache.yml --tags=cachesetup
   ```

CHAPTER 5. CONFIGURE MONITORING

5.1. CONFIGURING EVENT NOTIFICATIONS

To configure which notifications you want to be displayed in the Administration Portal, see Configuring Event Notifications in the Administration Portal in the Red Hat Virtualization 4.4 *Administration Guide*.

PART II. MAINTENANCE TASKS

CHAPTER 6. BASIC OPERATIONS

Some basic operations are required for many administrative and troubleshooting tasks. This section covers how to safely perform basic tasks like shutting down and starting up the hyperconverged cluster.

6.1. CREATING A SHUTDOWN PLAYBOOK

A hyperconverged environment must be shut down in a particular order. The simplest way to do this is to create a shutdown playbook that can be run from the Hosted Engine virtual machine.

The **ovirt.ovirt.shutdown_env** role enables Global Maintenance Mode, and initiates shutdown for all virtual machines and hosts in the cluster. Host shutdown is asynchronous. The playbook terminates before hyperconverged hosts are actually shut down.

Prerequisites

- Ensure that the **ovirt.ovirt.shutdown_env** ansible role is available on the Hosted Engine virtual machine.

  ```
  # yum install ovirt-ansible-shutdown-env -y
  ```

Procedure

1. Log in to the Hosted Engine virtual machine.

2. Create a shutdown playbook for your environment.
   Use the following template to create the playbook file.

   - Replace **ovirt-engine.example.com** with the FQDN of your Hosted Engine virtual machine.

   - Replace **123456** with the password for the **admin@internal** account.

   Example playbook file: shutdown_rhhi-v.yml

   ```
   ---
   - name: oVirt shutdown environment
     hosts: localhost
     connection: local
     gather_facts: false

     vars:
       engine_url: https://ovirt-engine.example.com/ovirt-engine/api
       engine_user: admin@internal
       engine_password: 123456
       engine_cafile: /etc/pki/ovirt-engine/ca.pem

     roles:
       - ovirt.ovirt.shutdown_env
   ```

6.2. SHUTTING DOWN RHHI FOR VIRTUALIZATION

A hyperconverged environment must be shut down in a particular order. Use an Ansible playbook to automate this process and ensure that your environment is shut down safely.

Prerequisites

- Create a shutdown playbook as described in Creating a shutdown playbook
- Ensure that the **ovirt.ovirt.shutdown_env** ansible role is available on the Hosted Engine virtual machine.

  ```
  # yum install ovirt-ansible-shutdown-env -y
  ```

Procedure

1. Run the shutdown playbook against the Hosted Engine virtual machine.

   ```
   # ansible-playbook -i localhost <shutdown_rhhi-v.yml>
   ```

6.3. STARTING UP A HYPERCONVERGED CLUSTER

Starting up a hyperconverged cluster is more complex than starting up a traditional compute or storage cluster. Follow these instructions to start up your hyperconverged cluster safely.

1. Power on all hosts in the cluster.
2. Ensure that the required services are available.
   a. Verify that the **glusterd** service started correctly on all hosts.

   ```
   # systemctl status glusterd
   ● glusterd.service - GlusterFS, a clustered file-system server
     Loaded: loaded (/usr/lib/systemd/system/glusterd.service; enabled; vendor preset:
   disabled)
    Drop-In: /etc/systemd/system/glusterd.service.d
            └─99-cpu.conf
     Active: active (running) since Wed 2018-07-18 11:15:03 IST; 3min 48s ago
     [...]
   ```

   If glusterd is not started, start it.

   ```
   # systemctl start glusterd
   ```

   b. Verify that host networks are available and hosts have IP addresses assigned to the required interfaces.

   ```
   # ip addr show
   ```

   c. Verify that all hosts are part of the storage cluster (listed as *Peer in Cluster (Connected)*).

   ```
   # gluster peer status

   Number of Peers: 2

   Hostname: 10.70.37.101
   Uuid: 773f1140-68f7-4861-a996-b1ba97586257
   State: Peer in Cluster (Connected)
   ```

```
Hostname: 10.70.37.102
Uuid: fc4e7339-9a09-4a44-aa91-64dde2fe8d15
State: Peer in Cluster (Connected)
```

d. Verify that all bricks are shown as online.

```
# gluster volume status engine
Status of volume: engine
Gluster process                             TCP Port  RDMA Port  Online  Pid
------------------------------------------------------------------------------
Brick 10.70.37.28:/gluster_bricks/engine/en
gine                                        49153     0          Y       23160
Brick 10.70.37.29:/gluster_bricks/engine/en
gine                                        49160     0          Y       12392
Brick 10.70.37.30:/gluster_bricks/engine/en
gine                                        49157     0          Y       15200
Self-heal Daemon on localhost               N/A       N/A        Y       23008
Self-heal Daemon on 10.70.37.30             N/A       N/A        Y       10905
Self-heal Daemon on 10.70.37.29             N/A       N/A        Y       13568

Task Status of Volume engine
------------------------------------------------------------------------------
There are no active volume tasks
```

3. Start the hosted engine virtual machine.

   a. Run the following command on the host that you want to be the hosted engine node.

   ```
   # hosted-engine --vm-start
   ```

   b. Verify that the hosted engine virtual machine has started correctly.

   ```
   # hosted-engine --vm-status
   ```

4. Take the hosted engine virtual machine out of Global Maintenance mode.

   a. Log in to the Administration Portal.

   b. Click **Compute → Hosts** and select the hosted engine node.

   c. Click ⋮ → **Disable Global HA Maintenance**.

5. Start any other virtual machines using the Web Console.

   a. Click **Compute → Virtualization**.

   b. Select any virtual machines you want to start and click **Run**.

CHAPTER 7. BACKING UP IMPORTANT FILES

Backing up important configuration files, inventory files, and modified playbooks makes it easy to restore or redeploy your cluster.

Red Hat recommends backing up your configuration after initial deployment, and after confirming the success of any major changes in your cluster. You can also take backups after a node has failed if necessary.

Prerequisites

- Example playbooks and inventory files are stored in the **/etc/ansible/roles/gluster.ansible/playbooks/hc-ansible-deployment** directory. If you have manually created or modified inventory and playbook files and you are not storing them in this directory, ensure that you know the path to their location.

Procedure

1. Log in to a hyperconverged host as the root user.

2. Change into the **hc-ansible-deployment** directory and back up the default **archive_config_inventory.yml** file.

```
# cd /etc/ansible/roles/gluster.ansible/playbooks/hc-ansible-deployment
# cp archive_config_inventory.yml archive_config_inventory.yml.bk
```

3. Edit the **archive_config_inventory.yml** file with details of the cluster you want to back up.

hosts

The backend FQDN of each host in the cluster that you want to back up.

backup_dir

The directory in which to store backup files.

nbde_setup

If you use Network-Bound Disk Encryption, set this to **true**. Otherwise, set to **false**.

upgrade

Set to **false**.

For example:

```
all:
  hosts:
    host1-backend.example.com:
    host2-backend.example.com:
    host3-backend.example.com:
  vars:
    backup_dir: /rhhi-backup
    nbde_setup: true
    upgrade: false
```

4. Run the **archive_config.yml** playbook using your updated inventory file with the **backupfiles** tag.

```
# ansible-playbook -i archive_config_inventory.yml archive_config.yml --tags=backupfiles
```

This creates an archive in the **/root** directory specific to each host FQDN in the **hosts** section of the inventory, for example, **/root/rhvh-node-host1-backend.example.com-backup.tar.gz**.

5. Transfer the backup archives to a different machine.

```
# scp /root/rhvh-node-host1-backend.example.com-backup.tar.gz backup-
host.example.com:/backups/
```

CHAPTER 8. MONITORING RED HAT HYPERCONVERGED INFRASTRUCTURE FOR VIRTUALIZATION

8.1. MONITORING VIRTUAL DATA OPTIMIZER (VDO)

Monitoring VDO helps in understanding when the physical storage is running out of space. Physical space in VDO needs to be monitored like thin provisioned storage. VDO devices should use thin provisioning because more logical space will be available and VDO space will be used in a more effective way. By default thin provisioning is enabled and it can be unchecked as required.

You can check available blocks, used space, and device information by clicking on **View Details**.

8.1.1. Monitoring VDO using the command line interface

There are several options for monitoring VDO using the command line interface.

The vdostats command

This command displays volume statistics including available blocks, number of blocks used, device name, percentage of physical blocks saved, and percentage of physical blocks on a VDO volume. For more information on vdostats, see the manual page: **man vdostats**.

The vdo status command

This command reports VDO system and volume status in YAML format.

The /sys/kvdo/<vdo_volume>/statistics directory

Files in this directory include volume statistics for VDO. You can read these files instead of using the **vdostats** command.

8.1.2. Monitoring VDO using the Web Console

Events related to VDO usage are displayed under the **Notifications** tab. Events provide information about the physical space remaining on the VDO volume, and keep you up to date about whether more physical space is needed.

Table 8.1. Types of Event Notification

Type	Text	Actions
Warn	Warning, low confirmed disk space. StorageDomainName domain has DiskSpace GB of confirmed free space.	● Delete data that is not required. ● Replace existing disks with larger disks. ● Add more Red Hat Gluster Storage servers and expand the volume across the new servers. ● Add more disks and expand the logical volumes underlying the Gluster volume.

CHAPTER 9. FREEING SPACE ON THINLY-PROVISIONED LOGICAL VOLUMES USING FSTRIM

You can manually run **fstrim** to return unused logical volume space to the thin pool, which can be available for other logical volumes.

Red Hat recommends running **fstrim** daily.

Prerequisites

- Verify that the thin pool logical volume supports discard behavior.
  Discard is supported if the output of the following command for the underlying device is not zero.

  ```
  # cat /sys/block/<device>/queue/discard_max_bytes
  ```

  > NOTE
  >
  > If **discard_max_bytes** returns 0, you can still execute the **fstrim** command on the thin logical volume to reclaim unused blocks to the thin pool. However, the **discard** or **trim** will not be passed down to the block layer.

Procedure

1. Run **fstrim** to restore physical space to the thin pool.

   ```
   # fstrim -v <mountpoint>
   ```

   For example, the following command discards any unused space it finds on the logical volume mounted at **/gluster_bricks/data/data**, and provides verbose output (**-v**).

   ```
   # fstrim -v /gluster_bricks/data/data
   ```

CHAPTER 10. ADD HYPERCONVERGED HOSTS TO RED HAT VIRTUALIZATION MANAGER

Follow this process to allow Red Hat Virtualization Manager to manage an existing hyperconverged host.

1. Log in to the Administration Portal.
2. Click **Compute → Hosts**.
3. Click **New**. The **New Host** window opens.
4. On the **General** tab, specify the following details about your hyperconverged host.
   - **Host Cluster**
   - **Name**
   - **Hostname**
   - **Password**
5. On the **General** tab, click the **Advanced Parameters** dropdown, and uncheck the **Automatically configure host firewall** checkbox.
6. Click **OK**.

CHAPTER 11. REINSTALLING A HYPERCONVERGED HOST

Some configuration changes require a hyperconverged host to be reinstalled before the configuration change can take effect. Follow these steps to reinstall a hyperconverged host.

1. Log in to the Administration Portal.
2. Click **Compute → Hosts**.
3. Select the host and click **Management** > **Maintenance** > **OK** to place this host in Maintenance mode.
4. Click **Installation** > **Reinstall** to open the Reinstall window.
5. On the General tab, uncheck the **Automatically Configure Host firewall** checkbox.
6. On the **Hosted Engine** tab, set the value of **Choose hosted engine deployment action** to **Deploy**.
7. Click **OK** to reinstall the host.

CHAPTER 12. RECOVERING FROM DISASTER

This chapter explains how to restore your cluster to a working state after a disk or server failure.

You must have configured disaster recovery options previously in order to use this chapter. See Configuring backup and recovery options for details.

12.1. MANUALLY RESTORING DATA FROM A BACKUP VOLUME

This section covers how to restore data from a remote backup volume to a freshly installed replacement deployment of Red Hat Hyperconverged Infrastructure for Virtualization.

To do this, you must:

1. Install and configure a replacement deployment according to the instructions in Deploying Red Hat Hyperconverged Infrastructure for Virtualization.

12.1.1. Restoring a volume from a geo-replicated backup

1. Install and configure a replacement Hyperconverged Infrastructure deployment
   For instructions, refer to *Deploying Red Hat Hyperconverged Infrastructure for Virtualization* : https://access.redhat.com/documentation/en-us/red_hat_hyperconverged_infrastructure_for_virtualization/1.8/html/deploying_red_hat_hyperco

2. Disable read-only mode on the backup volume
   Geo-replicated volumes are set to read-only after each sync to ensure that data is not modified. Red Hat Virtualization needs write permissions in order to import the volume as a storage domain.

   Run the following command to disable read-only mode on the backup volume.

   ```
   # gluster volume set <backup-vol> features.read-only off
   ```

3. Import the backup of the storage domain
   From the new Hyperconverged Infrastructure deployment, in the Administration Portal:

   a. Click **Storage → Domains**.

   b. Click **Import Domain**. The *Import Pre-Configured Domain* window opens.

   c. In the **Storage Type** field, specify **GlusterFS**.

   d. In the **Name** field, specify a name for the new volume that will be created from the backup volume.

   e. In the **Path** field, specify the path to the backup volume.

   f. Click **OK**. The following warning appears, with any active data centers listed below:

   ```
   This operation might be unrecoverable and destructive!

   Storage Domain(s) are already attached to a Data Center.
   Approving this operation might cause data corruption if
   both Data Centers are active.
   ```

g. Check the **Approve operation** checkbox and click **OK**.

4. Determine a list of virtual machines to import

a. Determine the imported domain’s identifier by running the following command:

```
# curl -v -k -X GET -u "admin@internal:password" -H "Accept: application/xml"
https://$ENGINE_FQDN/ovirt-engine/api/storagedomains/
```

For example:

```
# curl -v -k -X GET -u "admin@example.com:mybadpassword" -H "Accept:
application/xml" https://10.0.2.1/ovirt-engine/api/storagedomains/
```

b. Determine the list of unregistered disks by running the following command:

```
# curl -v -k -X GET -u "admin@internal:password" -H "Accept: application/xml"
"https://$ENGINE_FQDN/ovirt-
engine/api/storagedomains/DOMAIN_ID/vms;unregistered"
```

For example:

```
# curl -v -k -X GET -u "admin@example.com:mybadpassword" -H "Accept:
application/xml" "https://10.0.2.1/ovirt-engine/api/storagedomains/5e1a37cf-933d-424c-
8e3d-eb9e40b690a7/vms;unregistered"
```

5. Perform a partial import of each virtual machine to the storage domain

a. Determine cluster identifier
The following command returns the cluster identifier.

```
# curl -v -k -X GET -u "admin@internal:password" -H "Accept: application/xml"
https://$ENGINE_FQDN/ovirt-engine/api/clusters/
```

For example:

```
# curl -v -k -X GET -u "admin@example:mybadpassword" -H "Accept: application/xml"
https://10.0.2.1/ovirt-engine/api/clusters/
```

b. Import the virtual machines
The following command imports a virtual machine without requiring all disks to be available in the storage domain.

```
# curl -v -k -u 'admin@internal:password' -H "Content-type: application/xml" -d '<action>
<cluster id="CLUSTER_ID"></cluster>
<allow_partial_import>true</allow_partial_import> </action>'
"https://ENGINE_FQDN/ovirt-
engine/api/storagedomains/DOMAIN_ID/vms/VM_ID/register"
```

For example:

```
# curl -v -k -u 'admin@example.com:mybadpassword' -H "Content-type: application/xml"
-d '<action> <cluster id="bf5a9e9e-5b52-4b0d-aeba-4ee4493f1072"></cluster>
```

```
<allow_partial_import>true</allow_partial_import> </action>' "https://10.0.2.1/ovirt-engine/api/storagedomains/8d21980a-a50b-45e9-9f32-cd8d2424882e/e164f8c6-769a-4cbd-ac2a-ef322c2c5f30/register"
```

For further information, see the Red Hat Virtualization *REST API Guide*: https://access.redhat.com/documentation/en-us/red_hat_virtualization/4.4/html/rest_api_guide/.

6. Migrate the partially imported disks to the new storage domain
   In the Administration Portal, click **Storage** → **Disks**, and Click the **Move Disk** option. Move the imported disks from the synced volume to the replacement cluster’s storage domain. For further information, see the Red Hat Virtualization Administration Guide.

7. Attach the restored disks to the new virtual machines
   Follow the instructions in the Red Hat Virtualization Virtual Machine Management Guide to attach the replacement disks to each virtual machine.

12.2. FAILING OVER TO A SECONDARY CLUSTER

This section covers how to fail over from your primary cluster to a remote secondary cluster in the event of server failure.

1. Configure failover to a remote cluster .

2. Verify that the mapping file for the source and target clusters remains accurate.

3. Disable read-only mode on the backup volume
   Geo-replicated volumes are set to read-only after each sync to ensure that data is not modified. Red Hat Virtualization needs write permissions in order to import the volume as a storage domain.

   Run the following command to disable read-only mode on the backup volume.

   ```
   # gluster volume set <backup-vol> features.read-only off
   ```

   ### IMPORTANT

   Make sure to stop the remote sync schedule at the primary site, before executing the failover playbook.

   If the Red Hat Virtualization Manager Administration portal at the primary site is reachable, then remove the scheduled remote data sync using the following steps:

   1. Login as **admin** to the **Red Hat Virtualization Manager Administration Portal** → Select **Storage** → Select **Domains**.

   2. Select the storage domain configured with remote data sync → Select **Remote Data Sync Setup** tab → Edit → choose the **Recurrence** as none.

4. Run the failover playbook with the **fail_over** tag.

   ```
   # ansible-playbook dr-rhv-failover.yml --tags="fail_over"
   ```

12.3. FAILING BACK TO A PRIMARY CLUSTER

This section covers how to fail back from your secondary cluster to the primary cluster after you have corrected the cause of a server failure.

1. Prepare the primary cluster for failback by running the cleanup playbook with the **clean_engine** tag.

   ```
   # ansible-playbook dr-cleanup.yml --tags="clean_engine"
   ```

2. Verify that the mapping file for the source and target clusters remains accurate.

3. Execute failback by running the failback playbook with the **fail_back** tag.

   ```
   # ansible-playbook dr-cleanup.yml --tags="fail_back"
   ```

12.4. STOPPING A GEO-REPLICATION SESSION USING RHV MANAGER

Stop a geo-replication session when you want to prevent data being replicated from an active source volume to a passive target volume via geo-replication.

1. **Verify that data is not currently being synchronized**
   Click the **Tasks** icon at the top right of the Manager, and review the Tasks page.

   Ensure that there are no ongoing tasks related to Data Synchronization.

   If data synchronization tasks are present, wait until they are complete.

2. **Stop the geo-replication session**

   a. Click **Storage → Volumes**.

   b. Click the name of the volume that you want to prevent geo-replicating.

   c. Click the **Geo-replication** subtab.

   d. Select the session that you want to stop, then click **Stop**.

12.5. TURNING OFF SCHEDULED BACKUPS BY DELETING THE GEO-REPLICATION SCHEDULE

You can stop scheduled backups via geo-replication by deleting the geo-replication schedule.

1. Log in to the Administration Portal on any source node.

2. Click **Storage → Domains**.

3. Click the name of the storage domain that you want to back up.

4. Click the **Remote Data Sync Setup** subtab.

5. Click **Setup**.
   The *Setup Remote Data Synchronization* window opens.

6. In the **Recurrence** field, select a recurrence interval type of **NONE** and click **OK**.

7. (Optional) Remove the geo-replication session
   Run the following command from the geo-replication master node:

   ```
   # gluster volume geo-replication MASTER_VOL SLAVE_HOST::SLAVE_VOL delete
   ```

   You can also run this command with the **reset-sync-time** parameter. For further information about this parameter and deleting a geo-replication session, see Deleting a Geo-replication Session in the Red Hat Gluster Storage 3.5 *Administration Guide*.

PART III. TROUBLESHOOTING

CHAPTER 13. SELF-HEAL DOES NOT COMPLETE

If a self-heal operation never completes, the cause could be a Gluster File ID (GFID) mismatch.

13.1. GLUSTER FILE ID MISMATCH

Diagnosis

1. **Check self-heal state.**
   Run the following command several times over a few minutes. Note the entries that are shown.

   ```
   # gluster volume heal <volname> info
   ```

   If the same entries are shown each time, these entries may have a GFID mismatch.

2. **Check the GFID of each entry on each host.**
   On each host, run the following command for each entry:

   ```
   # getfattr -d -m. -ehex <backend_path> -h
   ```

   The **<backend_path>** for an entry is comprised of the brick path and the entry. For example, if the brick for the **engine** volume has the path of **/gluster_bricks/engine/engine** and the entry shown in heal info is **58d392a6-e5b1-4aed-9bbc-952210a7137d/ha_agent/hosted-engine.metadata**, the **backend_path** to use is **/gluster_bricks/engine/engine/58d392a6-e5b1-4aed-9bbc-952210a7137d/ha_agent/hosted-engine.metadata**.

3. **Compare the output from each host.**
   If the **trusted.gfid** for an entry is not the same on all hosts, there is a GFID mismatch.

Solution

1. Resolve the mismatch in favor of the GFID with the most recent modification time:

   ```
   # gluster volume heal <volume> split-brain latest-mtime <entry>
   ```

   For example:

   ```
   # gluster volume heal engine split-brain latest-mtime /58d392a6-e5b1-4aed-9bbc-
   952210a7137d/ha_agent/hosted-engine.metadata
   ```

2. Manually trigger a heal on the volume.

   ```
   # gluster volume heal <volname>
   ```

PART IV. REFERENCE MATERIAL

APPENDIX A. UNDERSTANDING THE NODE_PREP_INVENTORY.YML FILE

The **node_prep_inventory.yml** file is an example Ansible inventory file that you can use to prepare a replacement host for your Red Hat Hyperconverged Infrastructure for Virtualization cluster.

You can find this file at **/etc/ansible/roles/gluster.ansible/playbooks/hc-ansible-deployment/node_prep_inventory.yml** on any hyperconverged host.

A.1. CONFIGURATION PARAMETERS FOR PREPARING A REPLACEMENT NODE

A.1.1. Hosts to configure

hc_nodes

A list of hyperconverged hosts that uses the back-end FQDN of the host, and the configuration details of those hosts. Configuration that is specific to a host is defined under that host's back-end FQDN. Configuration that is common to all hosts is defined in the **vars:** section.

```
hc_nodes:
  hosts:
    new-host-backend-fqdn.example.com:
      [configuration specific to this host]
  vars:
    [configuration common to all hosts]
```

A.1.2. Multipath devices

blacklist_mpath_devices (optional)

By default, Red Hat Virtualization Host enables multipath configuration, which provides unique multipath names and worldwide identifiers for all disks, even when disks do not have underlying multipath configuration. Include this section if you do not have multipath configuration so that the multipath device names are not used for listed devices. Disks that are not listed here are assumed to have multipath configuration available, and require the path format **/dev/mapper/<WWID>** instead of **/dev/sdx** when defined in subsequent sections of the inventory file.
On a server with four devices (**sda**, **sdb**, **sdc** and **sdd**), the following configuration blacklists only two devices. The path format **/dev/mapper/<WWID>** is expected for devices not in this list.

```
hc_nodes:
  hosts:
    new-host-backend-fqdn.example.com:
      blacklist_mpath_devices:
        - sdb
        - sdc
```

IMPORTANT

Do not list encrypted devices (**luks_*** devices) in **blacklist_mpath_devices**, as they require multipath configuration to work.

A.1.3. Deduplication and compression

gluster_infra_vdo (optional)

Include this section to define a list of devices to use deduplication and compression. These devices require the **/dev/mapper/<name>** path format when you define them as volume groups in **gluster_infra_volume_groups**. Each device listed must have the following information:

name

A short name for the VDO device, for example **vdo_sdc**.

device

The device to use, for example, /**dev**/**sdc**.

logicalsize

The logical size of the VDO volume. Set this to ten times the size of the physical disk, for example, if you have a 500 GB disk, set **logicalsize: '5000G'**.

emulate512

If you use devices with a 4 KB block size, set this to **on**.

slabsize

If the logical size of the volume is 1000 GB or larger, set this to **32G**. If the logical size is smaller than 1000 GB, set this to **2G**.

blockmapcachesize

Set this to **128M**.

writepolicy

Set this to **auto**.

For example:

```
hc_nodes:
  hosts:
    new-host-backend-fqdn.example.com:
      gluster_infra_vdo:
      - { name: 'vdo_sdc', device: '/dev/sdc', logicalsize: '5000G',
        emulate512: 'off', slabsize: '32G', blockmapcachesize: '128M',
        writepolicy: 'auto' }
      - { name: 'vdo_sdd', device: '/dev/sdd', logicalsize: '500G',
        emulate512: 'off', slabsize: '2G', blockmapcachesize: '128M',
        writepolicy: 'auto' }
```

A.1.4. Storage infrastructure

gluster_infra_volume_groups (required)

This section creates the volume groups that contain the logical volumes.

```
hc_nodes:
  hosts:
    new-host-backend-fqdn.example.com:
      gluster_infra_volume_groups:
        - vgname: gluster_vg_sdb
          pvname: /dev/sdb
        - vgname: gluster_vg_sdc
          pvname: /dev/mapper/vdo_sdc
```

▪

gluster_infra_mount_devices (required)

This section creates the logical volumes that form Gluster bricks.

```
hc_nodes:
  hosts:
    new-host-backend-fqdn.example.com:
      gluster_infra_mount_devices:
        - path: /gluster_bricks/engine
          lvname: gluster_lv_engine
          vgname: gluster_vg_sdb
        - path: /gluster_bricks/data
          lvname: gluster_lv_data
          vgname: gluster_vg_sdc
        - path: /gluster_bricks/vmstore
          lvname: gluster_lv_vmstore
          vgname: gluster_vg_sdd
```

gluster_infra_thinpools (optional)

This section defines logical thin pools for use by thinly provisioned volumes. Thin pools are not suitable for the **engine** volume, but can be used for the **vmstore** and **data** volume bricks.

vgname

The name of the volume group that contains this thin pool.

thinpoolname

A name for the thin pool, for example, **gluster_thinpool_sdc**.

thinpoolsize

The sum of the sizes of all logical volumes to be created in this volume group.

poolmetadatasize

Set to **16G**; this is the recommended size for supported deployments.

```
hc_nodes:
  hosts:
    new-host-backend-fqdn.example.com:
      gluster_infra_thinpools:
        - {vgname: 'gluster_vg_sdc', thinpoolname: 'gluster_thinpool_sdc', thinpoolsize: '500G',
poolmetadatasize: '16G'}
        - {vgname: 'gluster_vg_sdd', thinpoolname: 'gluster_thinpool_sdd', thinpoolsize: '500G',
poolmetadatasize: '16G'}
```

gluster_infra_cache_vars (optional)

This section defines cache logical volumes to improve performance for slow devices. A fast cache device is attached to a thin pool, and requires **gluster_infra_thinpool** to be defined.

vgname

The name of a volume group with a slow device that requires a fast external cache.

cachedisk

The paths of the slow and fast devices, separated with a comma, for example, to use a cache device **sde** with the slow device **sdb**, specify **/dev/sdb,/dev/sde**.

cachelvname

A name for this cache logical volume.

cachethinpoolname

The thin pool to which the fast cache volume is attached.

cachelvsize

The size of the cache logical volume. Around 0.01% of this size is used for cache metadata.

cachemode

The cache mode. Valid values are **writethrough** and **writeback**.

```
hc_nodes:
  hosts:
    new-host-backend-fqdn.example.com:
      gluster_infra_cache_vars:
        - vgname: gluster_vg_sdb
          cachedisk: /dev/sdb,/dev/sde
          cachelvname: cachelv_thinpool_sdb
          cachethinpoolname: gluster_thinpool_sdb
          cachelvsize: '250G'
          cachemode: writethrough
```

gluster_infra_thick_lvs (required)

The thickly provisioned logical volumes that are used to create bricks. Bricks for the **engine** volume must be thickly provisioned.

vgname

The name of the volume group that contains the logical volume.

lvname

The name of the logical volume.

size

The size of the logical volume. The **engine** logical volume requires **100G**.

```
hc_nodes:
  hosts:
    new-host-backend-fqdn.example.com:
      gluster_infra_thick_lvs:
        - vgname: gluster_vg_sdb
          lvname: gluster_lv_engine
          size: 100G
```

gluster_infra_lv_logicalvols (required)

The thinly provisioned logical volumes that are used to create bricks.

vgname

The name of the volume group that contains the logical volume.

thinpool

The thin pool that contains the logical volume, if this volume is thinly provisioned.

lvname

The name of the logical volume.

size

The size of the logical volume. The **engine** logical volume requires **100G**.

```
hc_nodes:
  hosts:
    new-host-backend-fqdn.example.com:
      gluster_infra_lv_logicalvols:
        - vgname: gluster_vg_sdc
          thinpool: gluster_thinpool_sdc
          lvname: gluster_lv_data
          lvsize: 200G
        - vgname: gluster_vg_sdd
          thinpool: gluster_thinpool_sdd
          lvname: gluster_lv_vmstore
          lvsize: 200G
```

gluster_infra_disktype (required)

Specifies the underlying hardware configuration of the disks. Set this to the value that matches your hardware: **RAID6**, **RAID5**, or **JBOD**.

```
hc_nodes:
  vars:
    gluster_infra_disktype: RAID6
```

gluster_infra_diskcount (required)

Specifies the number of data disks in the RAID set. For a **JBOD** disk type, set this to **1**.

```
hc_nodes:
  vars:
    gluster_infra_diskcount: 10
```

gluster_infra_stripe_unit_size (required)

The stripe size of the RAID set in megabytes.

```
hc_nodes:
  vars:
    gluster_infra_stripe_unit_size: 256
```

gluster_features_force_varlogsizecheck (required)

Set this to **true** if you want to verify that your **/var/log** partition has sufficient free space during the deployment process. It is important to have sufficient space for logs, but it is not required to verify space requirements at deployment time if you plan to monitor space requirements carefully.

```
hc_nodes:
  vars:
    gluster_features_force_varlogsizecheck: false
```

gluster_set_selinux_labels (required)

Ensures that volumes can be accessed when SELinux is enabled. Set this to **true** if SELinux is enabled on this host.

```
hc_nodes:
  vars:
    gluster_set_selinux_labels: true
```

A.1.5. Firewall and network infrastructure

gluster_infra_fw_ports (required)

A list of ports to open between all nodes, in the format **<port>/<protocol>**.

```
hc_nodes:
  vars:
    gluster_infra_fw_ports:
      - 2049/tcp
      - 54321/tcp
      - 5900-6923/tcp
      - 16514/tcp
      - 5666/tcp
      - 16514/tcp
```

gluster_infra_fw_permanent (required)

Ensures the ports listed in **gluster_infra_fw_ports** are open after nodes are rebooted. Set this to **true** for production use cases.

```
hc_nodes:
  vars:
  gluster_infra_fw_permanent: true
```

gluster_infra_fw_state (required)

Enables the firewall. Set this to **enabled** for production use cases.

```
hc_nodes:
  vars:
    gluster_infra_fw_state: enabled
```

gluster_infra_fw_zone (required)

Specifies the firewall zone to which these **gluster_infra_fw_*** parameters are applied.

```
hc_nodes:
  vars:
    gluster_infra_fw_zone: public
```

gluster_infra_fw_services (required)

A list of services to allow through the firewall. Ensure **glusterfs** is defined here.

```
hc_nodes:
  vars:
  gluster_infra_fw_services:
     - glusterfs
```

A.2. EXAMPLE NODE_PREP_INVENTORY.YML

```
# Section for Host Preparation Phase
hc_nodes:
 hosts:
   # Host - The node which need to be prepared for replacement
   new-host-backend-fqdn.example.com:

    # Blacklist multipath devices which are used for gluster bricks
    # If you omit blacklist_mpath_devices it means all device will be whitelisted.
    # If the disks are not blacklisted, and then its taken that multipath configuration
    # exists in the server and one should provide /dev/mapper/<WWID> instead of /dev/sdx
    blacklist_mpath_devices:
      - sdb
      - sdc

    # Enable this section gluster_infra_vdo, if dedupe & compression is
    # required on that storage volume.
    # The variables refers to:
    # name        - VDO volume name to be used
    # device      - Disk name on which VDO volume to created
    # logicalsize - Logical size of the VDO volume.This value is 10 times
    #               the size of the physical disk
    # emulate512  - VDO device is made as 4KB block sized storage volume(4KN)
    # slabsize    - VDO slab size. If VDO logical size >= 1000G then
    #               slabsize is 32G else slabsize is 2G
    #
    # Following VDO values are as per recommendation and treated as constants:
    # blockmapcachesize - 128M
    # writepolicy       - auto
    #
    # gluster_infra_vdo:
    #   - { name: vdo_sdc, device: /dev/sdc, logicalsize: 5000G, emulate512: off, slabsize: 32G,
    #       blockmapcachesize:  128M, writepolicy: auto }
    #   - { name: vdo_sdd, device: /dev/sdd, logicalsize: 3000G, emulate512: off, slabsize: 32G,
    #       blockmapcachesize:  128M, writepolicy: auto }

    # When dedupe and compression is enabled on the device,
    # use pvname for that device as /dev/mapper/<vdo_device_name> # # The variables refers to: #
vgname - VG to be created on the disk # pvname - Physical disk (/dev/sdc) or VDO volume
(/dev/mapper/vdo_sdc) gluster_infra_volume_groups: - vgname: gluster_vg_sdb pvname: /dev/sdb -
vgname: gluster_vg_sdc pvname: /dev/mapper/vdo_sdc - vgname: gluster_vg_sdd pvname:
/dev/mapper/vdo_sdd gluster_infra_mount_devices: - path: /gluster_bricks/engine lvname:
gluster_lv_engine vgname: gluster_vg_sdb - path: /gluster_bricks/data lvname: gluster_lv_data
vgname: gluster_vg_sdc - path: /gluster_bricks/vmstore lvname: gluster_lv_vmstore vgname:
gluster_vg_sdd # 'thinpoolsize is the sum of sizes of all LVs to be created on that VG
    # In the case of VDO enabled, thinpoolsize is 10 times the sum of sizes
    # of all LVs to be created on that VG. Recommended values for
    # poolmetadatasize is 16GB and that should be considered exclusive of
    # thinpoolsize
    gluster_infra_thinpools:
      - {vgname: gluster_vg_sdc, thinpoolname: gluster_thinpool_sdc, thinpoolsize: 500G,
poolmetadatasize: 16G}
      - {vgname: gluster_vg_sdd, thinpoolname: gluster_thinpool_sdd, thinpoolsize: 500G,
poolmetadatasize: 16G}

    # Enable the following section if LVM cache is to enabled
    # Following are the variables:
    # vgname           - VG with the slow HDD device that needs caching
    # cachedisk        - Comma separated value of slow HDD and fast SSD
```

```
      #                  In this example, /dev/sdb is the slow HDD, /dev/sde is fast SSD
      # cachelvname       - LV cache name
      # cachethinpoolname - Thinpool to which the fast SSD to be attached
      # cachelvsize       - Size of cache data LV. This is the SSD_size - (1/1000) of SSD_size
      #                  1/1000th of SSD space will be used by cache LV meta
      # cachemode         - writethrough or writeback
      # gluster_infra_cache_vars:
      #  - vgname: gluster_vg_sdb
      #    cachedisk: /dev/sdb,/dev/sde
      #    cachelvname: cachelv_thinpool_sdb
      #    cachethinpoolname: gluster_thinpool_sdb
      #    cachelvsize: 250G
      #    cachemode: writethrough

      # Only the engine brick needs to be thickly provisioned
      # Engine brick requires 100GB of disk space
      gluster_infra_thick_lvs:
        - vgname: gluster_vg_sdb
          lvname: gluster_lv_engine
          size: 100G

      gluster_infra_lv_logicalvols:
        - vgname: gluster_vg_sdc
          thinpool: gluster_thinpool_sdc
          lvname: gluster_lv_data
          lvsize: 200G
        - vgname: gluster_vg_sdd
          thinpool: gluster_thinpool_sdd
          lvname: gluster_lv_vmstore
          lvsize: 200G

  # Common configurations
  vars:
    # In case of IPv6 based deployment "gluster_features_enable_ipv6" needs to be enabled,below
line needs to be uncommented, like:
    # gluster_features_enable_ipv6: true

    # Firewall setup
    gluster_infra_fw_ports:
       - 2049/tcp
       - 54321/tcp
       - 5900-6923/tcp
       - 16514/tcp
       - 5666/tcp
       - 16514/tcp
    gluster_infra_fw_permanent: true
    gluster_infra_fw_state: enabled
    gluster_infra_fw_zone: public
    gluster_infra_fw_services:
       - glusterfs
    # Allowed values for gluster_infra_disktype - RAID6, RAID5, JBOD
    gluster_infra_disktype: RAID6

    # gluster_infra_diskcount is the number of data disks in the RAID set.
    #  Note for JBOD its 1
    gluster_infra_diskcount: 10
```

```
gluster_infra_stripe_unit_size: 256
gluster_features_force_varlogsizecheck: false
gluster_set_selinux_labels: true
```

CHAPTER 14. UNDERSTANDING THE NODE_REPLACE_INVENTORY.YML FILE

The **node_replace_inventory.yml** file is an example Ansible inventory file that you can use to prepare a replacement host for your Red Hat Hyperconverged Infrastructure for Virtualization cluster.

You can find this file at **/etc/ansible/roles/gluster.ansible/playbooks/hc-ansible-deployment/node_replace_inventory.yml** on any hyperconverged host.

14.1. CONFIGURATION PARAMETERS FOR NODE REPLACEMENT

hosts (required)

Defines one active host in the cluster using the back-end FQDN.

```
cluster_nodes:
  hosts:
    host2-backend-fqdn.example.com:
  vars:
    [common host configuration]
```

gluster_maintenance_old_node (required)

Defines the backend FQDN of the node being replaced.

```
cluster_nodes:
  hosts:
    host2-backend-fqdn.example.com:
  vars:
    gluster_maintenance_old_node: host1-backend-fqdn.example.com
```

gluster_maintenance_new_node (required)

Defines the backend FQDN of the replacement node.

```
cluster_nodes:
  hosts:
    host2-backend-fqdn.example.com:
  vars:
    gluster_maintenance_new_node: new-host-backend-fqdn.example.com
```

gluster_maintenance_cluster_node (required)

An active node in the cluster. Cannot be the same as **gluster_maintenance_cluster_node_2**.

```
cluster_nodes:
  hosts:
    host2-backend-fqdn.example.com:
  vars:
    gluster_maintenance_cluster_node: host2-backend-fqdn.example.com
```

gluster_maintenance_cluster_node_2 (required)

An active node in the cluster. Cannot be the same as **gluster_maintenance_cluster_node**.

```
cluster_nodes:
  hosts:
    host2-backend-fqdn.example.com:
  vars:
    gluster_maintenance_cluster_node_2: host3-backend-fqdn.example.com
```

14.2. EXAMPLE NODE_REPLACE_INVENTORY.YML

```
cluster_node:
  hosts:
   host2-backend-fqdn.example.com:

  vars:
    gluster_maintenance_old_node: host1-backend-fqdn.example.com
    gluster_maintenance_new_node: new-host-backend-fqdn.example.com
    gluster_maintenance_cluster_node: host2-backend-fqdn.example.com
    gluster_maintenance_cluster_node_2: host3-backend-fqdn.example.com
```

APPENDIX B. FENCING POLICIES FOR RED HAT GLUSTER STORAGE

The following fencing policies are required for Red Hat Hyperconverged Infrastructure for Virtualization (RHHI for Virtualization) deployments. They ensure that hosts are not shut down in situations where brick processes are still running, or when shutting down the host would remove the volume's ability to reach a quorum.

These policies can be set in the **New Cluster** or **Edit Cluster** window in the Administration Portal when Red Hat Gluster Storage functionality is enabled.

Skip fencing if gluster bricks are up

Fencing is skipped if bricks are running and can be reached from other peers.

Skip fencing if gluster quorum not met

Fencing is skipped if bricks are running and shutting down the host will cause loss of quorum

These policies are checked after all other fencing policies when determining whether a node is fenced.

Additional fencing policies may be useful for your deployment. For further details about fencing, see the Red Hat Virtualization *Technical Reference*: https://access.redhat.com/documentation/en-us/red_hat_virtualization/4.4/html/technical_reference/fencing.

APPENDIX C. CONFIGURE RED HAT GLUSTER STORAGE AS A RED HAT VIRTUALIZATION STORAGE DOMAIN

C.1. CREATE THE LOGICAL NETWORK FOR GLUSTER TRAFFIC

1. **Log in to the Administration Portal**
   Browse to the Administration Portal (for example, **http://engine.example.com/ovirt-engine**) and log in using the administrative credentials you configured in Deploy the Hosted Engine using the Web Console.

2. **Create a logical network for gluster traffic**

   a. Click **Network** → **Networks** and then click **New**. The **New Logical Network** wizard appears.

   b. On the **General** tab of the wizard, provide a **Name** for the new logical network, and uncheck the **VM Network** checkbox.

   c. On the **Cluster** tab of the wizard, uncheck the **Required** checkbox.

   d. Click **OK** to create the new logical network.

3. **Enable the new logical network for gluster**

   a. Click the **Network** → **Networks** and select the new logical network.

   b. Click the **Clusters** subtab and then click **Manage Network**. The **Manage Network** dialogue appears.

   c. In the **Manage Network** dialogue, check the **Migration Network** and **Gluster Network** checkboxes.

   d. Click **OK** to save.

4. **Attach the gluster network to the host**

   a. Click **Compute** → **Hosts** and select the host.

   b. Click the **Network Interfaces** subtab and then click **Setup Host Networks**. The **Setup Host Networks** window opens.

   c. Drag and drop the newly created network to the correct interface.

   d. Ensure that the **Verify connectivity between Host and Engine** checkbox is checked.

   e. Ensure that the **Save network configuration** checkbox is checked.

   f. Click **OK** to save.

5. **Verify the health of the network**
   Click the **Network Interfaces** tab and check the state of the host’s network. If the network interface enters an "Out of sync" state or does not have an IP Address, click **Management** → **Refresh Capabilities**.

C.2. CONFIGURE ADDITIONAL HYPERCONVERGED HOSTS

If your environment uses IPv6 addresses, or if you did not specify additional hyperconverged hosts (host2, host3) as part of Configure Red Hat Gluster Storage for Hosted Engine using the Web Console , follow these steps in the Administration Portal for each of the other hyperconverged hosts.

a. Click **Compute → Hosts** and then click **New** to open the *New Host* window.

b. Provide the **Name**, **Hostname**, and **Password** for the host that you want to manage.

c. Under **Advanced Parameters**, uncheck the **Automatically configure host firewall** checkbox, as firewall rules are already configured by the deployment process.

d. In the **Hosted Engine** tab of the *New Host* dialog, set the value of **Choose hosted engine deployment action** to **Deploy**. This ensures that the hosted engine can run on the new host.

e. Click **OK**.

    1. **Attach the gluster network to the new host**

f. Click the name of the newly added host to go to the host page.

g. Click the **Network Interfaces** subtab and then click **Setup Host Networks**.

h. Drag and drop the newly created network to the correct interface.

i. Ensure that the **Verify connectivity** checkbox is checked.

j. Ensure that the **Save network configuration** checkbox is checked.

k. Click **OK** to save.

    1. In the **General** subtab for this host, verify that the value of **Hosted Engine HA** is *Active*, with a positive integer as a score.

> **IMPORTANT**
>
> If **Score** is listed as **N/A**, you may have forgotten to select the **deploy** action for **Choose hosted engine deployment action**. Follow the steps in *Reinstalling a hyperconverged host* in *Maintaining Red Hat Enterprise Linux based RHHI* to reinstall the host with the **deploy** action.

    2. **Verify the health of the network**
    Click the **Network Interfaces** tab and check the state of the host's network. If the network interface enters an "Out of sync" state or does not have an IP Address, click **Management → Refresh Capabilities**.

See the Red Hat Virtualization 4.4 *Self-Hosted Engine Guide* for further details: https://access.redhat.com/documentation/en-us/red_hat_virtualization/4.4/html/self-hosted_engine_guide/chap-installing_additional_hosts_to_a_self-hosted_environment

APPENDIX D. WORKING WITH FILES ENCRYPTED USING ANSIBLE VAULT

Red Hat recommends encrypting the contents of deployment and management files that contain passwords and other sensitive information. Ansible Vault is one method of encrypting these files. More information about Ansible Vault is available in the Ansible documentation.

D.1. ENCRYPTING FILES

You can create an encrypted file by using the **ansible-vault create** command, or encrypt an existing file by using the **ansible-vault encrypt** command.

When you create an encrypted file or encrypt an existing file, you are prompted to provide a password. This password is used to decrypt the file after encryption. You must provide this password whenever you work directly with information in this file or run a playbook that relies on the file's contents.

Creating an encrypted file

```
$ ansible-vault create variables.yml
New Vault password:
Confirm New Vault password:
```

The **ansible-vault create** command prompts for a password for the new file, then opens the new file in the default text editor (defined as **$EDITOR** in your shell environment) so that you can populate the file before saving it.

If you have already created a file and you want to encrypt it, use the **ansible-vault encrypt** command.

Encrypting an existing file

```
$ ansible-vault encrypt existing-variables.yml
New Vault password:
Confirm New Vault password:
Encryption successful
```

D.2. EDITING ENCRYPTED FILES

You can edit an encrypted file using the **ansible-vault edit** command and providing the Vault password for that file.

Editing an encrypted file

```
$ ansible-vault edit variables.yml
New Vault password:
Confirm New Vault password:
```

The **ansible-vault edit** command prompts for a password for the file, then opens the file in the default text editor (defined as **$EDITOR** in your shell environment) so that you can edit and save the file contents.

D.3. REKEYING ENCRYPTED FILES TO A NEW PASSWORD

You can change the password used to decrypt a file by using the **ansible-vault rekey** command.

```
$ ansible-vault rekey variables.yml
Vault password:
New Vault password:
Confirm New Vault password:
Rekey successful
```

The **ansible-vault rekey** command prompts for the current Vault password, and then prompts you to set and confirm a new Vault password.

APPENDIX E. GLOSSARY OF TERMS

E.1. VIRTUALIZATION TERMS

Administration Portal

A web user interface provided by Red Hat Virtualization Manager, based on the oVirt engine web user interface. It allows administrators to manage and monitor cluster resources like networks, storage domains, and virtual machine templates.

Hosted Engine

The instance of Red Hat Virtualization Manager that manages RHHI for Virtualization.

Hosted Engine virtual machine

The virtual machine that acts as Red Hat Virtualization Manager. The Hosted Engine virtual machine runs on a virtualization host that is managed by the instance of Red Hat Virtualization Manager that is running on the Hosted Engine virtual machine.

Manager node

A virtualization host that runs Red Hat Virtualization Manager directly, rather than running it in a Hosted Engine virtual machine.

Red Hat Enterprise Linux host

A physical machine installed with Red Hat Enterprise Linux plus additional packages to provide the same capabilities as a Red Hat Virtualization host. This type of host is not supported for use with RHHI for Virtualization.

Red Hat Virtualization

An operating system and management interface for virtualizing resources, processes, and applications for Linux and Microsoft Windows workloads.

Red Hat Virtualization host

A physical machine installed with Red Hat Virtualization that provides the physical resources to support the virtualization of resources, processes, and applications for Linux and Microsoft Windows workloads. This is the only type of host supported with RHHI for Virtualization.

Red Hat Virtualization Manager

A server that runs the management and monitoring capabilities of Red Hat Virtualization.

Self-Hosted Engine node

A virtualization host that contains the Hosted Engine virtual machine. All hosts in a RHHI for Virtualization deployment are capable of becoming Self-Hosted Engine nodes, but there is only one Self-Hosted Engine node at a time.

storage domain

A named collection of images, templates, snapshots, and metadata. A storage domain can be comprised of block devices or file systems. Storage domains are attached to data centers in order to provide access to the collection of images, templates, and so on to hosts in the data center.

virtualization host

A physical machine with the ability to virtualize physical resources, processes, and applications for client access.

VM Portal

A web user interface provided by Red Hat Virtualization Manager. It allows users to manage and monitor virtual machines.

E.2. STORAGE TERMS

brick

An exported directory on a server in a trusted storage pool.

cache logical volume

A small, fast logical volume used to improve the performance of a large, slow logical volume.

geo-replication

One way asynchronous replication of data from a source Gluster volume to a target volume. Geo-replication works across local and wide area networks as well as the Internet. The target volume can be a Gluster volume in a different trusted storage pool, or another type of storage.

gluster volume

A logical group of bricks that can be configured to distribute, replicate, or disperse data according to workload requirements.

logical volume management (LVM)

A method of combining physical disks into larger virtual partitions. Physical volumes are placed in volume groups to form a pool of storage that can be divided into logical volumes as needed.

Red Hat Gluster Storage

An operating system based on Red Hat Enterprise Linux with additional packages that provide support for distributed, software-defined storage.

source volume

The Gluster volume that data is being copied from during geo-replication.

storage host

A physical machine that provides storage for client access.

target volume

The Gluster volume or other storage volume that data is being copied to during geo-replication.

thin provisioning

Provisioning storage such that only the space that is required is allocated at creation time, with further space being allocated dynamically according to need over time.

thick provisioning

Provisioning storage such that all space is allocated at creation time, regardless of whether that space is required immediately.

trusted storage pool

A group of Red Hat Gluster Storage servers that recognise each other as trusted peers.

E.3. HYPERCONVERGED INFRASTRUCTURE TERMS

Red Hat Hyperconverged Infrastructure (RHHI) for Virtualization

RHHI for Virtualization is a single product that provides both virtual compute and virtual storage resources. Red Hat Virtualization and Red Hat Gluster Storage are installed in a converged configuration, where the services of both products are available on each physical machine in a cluster.

hyperconverged host

A physical machine that provides physical storage, which is virtualized and consumed by virtualized processes and applications run on the same host. All hosts installed with RHHI for Virtualization are hyperconverged hosts.

Web Console

The web user interface for deploying, managing, and monitoring RHHI for Virtualization. The Web Console is provided by the Web Console service and plugins for Red Hat Virtualization Manager.