



Red Hat Hyperconverged Infrastructure for Virtualization 1.7

『Maintaining Red Hat Hyperconverged Infrastructure for Virtualization』

Common maintenance tasks for Virtualization (仮想化用の Red Hat Lake
Infrastructure の一般的なメンテナンスタスク)

Red Hat Hyperconverged Infrastructure for Virtualization 1.7 『Maintaining Red Hat Hyperconverged Infrastructure for Virtualization』

Common maintenance tasks for Virtualization (仮想化用の Red Hat Lake Infrastructure の一般的なメンテナンスタスク)

Enter your first name here. Enter your surname here.

Enter your organisation's name here. Enter your organisational division here.

Enter your email address here.

法律上の通知

Copyright © 2022 | You need to change the HOLDER entity in the en-US/Maintaining_Red_Hat_Hyperconverged_Infrastructure_for_Virtualization.ent file |.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

概要

Red Hat IaaS Infrastructure for Virtualization(RHHI for Virtualization)は、コンピュー、ストレージ、ネットワーク、管理機能を単一のソリューションに統合し、デプロイメントを簡素化し、企業とメンテナンスのコストが削減します。本書では、Red Hat IaaS Infrastructure for Virtualization 固有のメンテナンスタスクを実行する方法を説明します。

目次

多様性を受け入れるオープンソースの強化	4
パート I. 設定タスク	5
第1章 コンピュートリソースおよびストレージリソースの追加	6
1.1. ANSIBLE を使用した新しいブリックの作成	6
1.2. ANSIBLE を使用した VDO レイヤーでの新しいブリックの作成	9
1.3. RED HAT VIRTUALIZATION MANAGER からのボリュームの拡張	12
1.4. WEB コンソールを使用して新規ノードに新規ボリュームを追加してハイパーコンバードクラスターの拡張	13
第2章 フェンスポリシーを使用した高可用性の設定	17
2.1. クラスターでのフェンシングポリシーの設定	17
2.2. ホストでのフェンシングパラメーターの設定	17
第3章 バックアップオプションおよびリカバリーオプションの設定	21
3.1. 前提条件	21
3.1.1. geo レプリケーションの前提条件	21
3.1.2. フェイルオーバーおよびフェイルバック設定の前提条件	21
3.2. サポート対象のバックアップおよびリカバリー設定	21
3.3. セカンダリーボリュームへのバックアップの設定	22
3.3.1. 前提条件	22
3.3.1.1. ソースボリュームで共有ストレージを有効にします。	22
3.3.1.2. ソースおよびターゲットボリュームでの暗号化の一致	23
3.3.2. Geo レプリケーションに適したターゲットボリュームを作成します。	23
3.3.3. ボリュームをバックアップするための geo レプリケーションの設定	23
3.3.3.1. geo レプリケーションセッションの作成	23
3.3.3.2. Geo レプリケーションセッションの作成の確認	24
3.3.3.3. 管理ポータルを使用したボリュームの状態の同期	24
3.3.4. Geo レプリケーションを使用した通常のバックアップのスケジューリング	24
3.4. セカンダリークラスターへのフェイルオーバーの設定およびフェイルバック	25
3.4.1. フェイルオーバーのためのセカンダリークラスターの作成	25
3.4.2. ソースおよびターゲットクラスター間のマッピングファイルの作成	26
3.4.3. ソースおよびターゲットクラスター間のフェイルオーバー Playbook の作成	26
3.4.4. プライマリークラスターのフェイルオーバークリーンアップ Playbook の作成	27
3.4.5. ソースおよびターゲットクラスター間でフェイルバック Playbook を作成します。	27
第4章 TLS/SSL(TRANSPORT LAYER SECURITY)を使用した暗号化の設定	29
4.1. 自己署名証明書を使用した TLS/SSL の設定	29
4.2. 認証局署名証明書を使用した TLS/SSL の設定	31
第5章 パフォーマンスの向上の設定	35
5.1. シャードサイズの変更によるボリュームパフォーマンスの向上	35
5.1.1. レプリケートされたボリュームのシャードサイズの変更	35
5.1.2. 調整されたボリュームでのシャードサイズの変更	38
5.2. 既存ボリュームの論理ボリュームキャッシュ(LVMCACHE)の設定	41
第6章 モニタリングの設定	44
6.1. イベント通知の設定	44
パート II. メンテナンスタスク	45
第7章 基本的な操作	46
7.1. シャットダウン PLAYBOOK の作成	46

7.2. 仮想化のための RHHI のシャットダウン	46
7.3. ハイパーコンバインドクラスタのインストール	47
第8章 仮想化向けの RED HAT IAAS INFRASTRUCTURE の監視	49
8.1. VDO(VIRTUAL DATA OPTIMIZER)の監視	49
8.1.1. コマンドラインインターフェースを使用した VDO の監視	49
8.1.2. Web コンソールを使用した VDO の監視	49
第9章 FSTRIM を使用してシンプロビジョニングされた論理ボリューム上の領域の解放	51
第10章 RED HAT VIRTUALIZATION MANAGER へのハイパーコンバインドホストの追加	52
第11章 ハイパーコンバインドホストの再インストール	53
第12章 ホストの置き換え	54
12.1. ANSIBLE を使用したプライマリハイパーコンバインドホストの置き換え	54
12.2. ANSIBLE を使用した他のハイパーコンバインドホストの置き換え	57
12.2.1. 別の FQDN を使用するようにハイパーコンバインドホストの置き換え	58
12.2.2. 同じ FQDN を使用するようにハイパーコンバインドホストを置き換える	62
12.3. ANSIBLE を使用した代替ハイパーコンバインドホストの準備	64
第13章 障害からの復旧	68
13.1. バックアップボリュームからのデータの手動による復元	68
13.1.1. 地理的に複製されたバックアップからのボリュームの復元	68
13.2. セカンダリークラスターへのフェイルオーバー	70
13.3. プライマリークラスターへのフェイルバック	70
13.4. RHV MANAGER を使用した GEO レプリケーションセッションの停止	71
13.5. GEO レプリケーションスケジュールを削除してスケジュールされたバックアップをオフにする	71
パート III. トラブルシューティング	72
第14章 自己修復が完了しない	73
14.1. GLUSTER ファイル ID の不一致	73
パート IV. 参考資料	74
付録A RED HAT GLUSTER STORAGE のフェンシングポリシー	75
付録B 用語集	76
B.1. 仮想化用語	76
B.2. ストレージ用語	77
B.3. ハイパーコンバインドインフラストラクチャーの用語	77

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、[Red Hat CTO である Chris Wright のメッセージ](#) をご覧ください。

パート I. 設定タスク

第1章 コンピュートリソースおよびストレージリソースの追加

Red Hat IaaS Infrastructure for Virtualization(RHHI for Virtualization)は、6 ノード、9 ノード、または 12 ノードにスケーリングできます。

コンピューターおよびストレージリソースは、複数の方法で追加できます。

- [新規ノードへの新規ボリュームの追加](#)
- [新規ノード全体で既存ボリュームを拡張します。](#)

また、既存のノードで利用可能な領域を増やして、コンピュートリソースを拡張せずにストレージを拡張できます。

- [Web コンソールを使用したシンプールの拡張](#)
- [Web コンソールを使用した論理ボリュームの拡張](#)
- [Web コンソールを使用した VDO デバイスの論理サイズの拡張](#)



注記

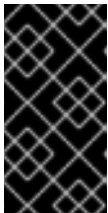
Red Hat Hyperconverged Infrastructure for Virtualization (Red Hat OpenShift Container Platformがインストールされた仮想マシンをホストするハイパーコンバージドノード)の上にOpenShift Container Storageを置くことは、サポートされていない構成です。

1.1. ANSIBLE を使用した新しいブリックの作成

多くのホストでブリックを一度に作成する場合は、ansible Playbook を作成してプロセスを自動化できます。このプロセスに従い、ハイパーコンバージド環境で使用するためにブリックを作成し、フォーマットし、マウントする Playbook を作成し、実行します。

前提条件

- 新しいブリックをホストする物理マシンをインストールします。
「[物理ホストマシンのインストール](#)」の順に従います。
- すべてのノード間のパスワードなしで、鍵ベースの SSH 認証を設定します。
Web コンソールを実行しているノードからすべての新規ノードおよび最初の新規ノードから他のすべての新規ノードにこれを設定します。



重要

RHHI for Virtualization は、IP アドレスと FQDN の両方について、これらのノード間のパスワードなしに鍵ベースの SSH 認証を想定します。これらのマシン間の鍵ベースの SSH 認証を、すべてのストレージおよび管理ネットワークインターフェースの IP アドレスと FQDN に設定するようにしてください。

[鍵ベースの認証の使用](#) の順に従って、パスワードなしで鍵ベースの SSH 認証を設定します。

- ホストが Virtual Disk Optimization(VDO)レイヤーを使用しないことを確認します。VDO レイヤーがある場合は、代わりに「[ansible を使用した VDO レイヤーでの新しいブリックの作成](#)」を使用してください。

手順

1. インベントリー ファイルを作成します。

以下の例を使用して、`/etc/ansible/roles/gluster.infra/playbooks` ディレクトリーに新規 インベントリー ファイルを作成します。

このファイルは、新規ブリックを作成するホストを一覧表示します。

インベントリー ファイルの例

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

2. ブリック.yml 変数ファイルを作成します。

以下の例を使用して、`/etc/ansible/roles/gluster.infra/playbooks` ディレクトリーに新しいブリック.yml ファイルを作成します。

このファイルは、各ホストで作成または使用される基礎となるストレージインフラストラクチャーおよび設定を定義します。

ブリック.yml 変数ファイルの例

```
# 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
```

```

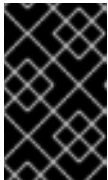
# thinpoolname is optional, if not provided `vgname' followed by _thinpool is
# used for name. poolmetadatasize is optional, default 16G is used
gluster_infra_thinpools:
  - {vgname: 'vg_vdb', thinpoolname: 'foo_thinpool', thinpoolsizesize: '10G', poolmetadatasize:
    '1G' }
  - {vgname: 'vg_vdc', thinpoolname: 'bar_thinpool', thinpoolsizesize: '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',
    cachelvsizesize: '20G', cachemetalvname: 'cachemeta',
    cachemetalvsizesize: '100M', cachemode: 'writethrough' }

# gluster_infra_mount_devices
gluster_infra_mount_devices:
  - {path: '/rhgs/thickl', vgname: 'vg_vdb', lvname: 'vg_vdb_thickl' }
  - {path: '/rhgs/thinlv1', vgname: 'vg_vdb', lvname: 'vg_vdb_thinlv' }
  - {path: '/rhgs/thinlv2', vgname: 'vg_vdc', lvname: 'vg_vdc_thinlv' }

```



重要

path: 定義された path: が **/rhgs** で開始されていない場合、ブリックは管理ポータルによって自動的に検出されません。 **create_brick.yml** Playbook を実行し、新しいブリックを管理ポータルに追加した後にホストストレージを同期します。

3. Playbook ファイル **create_brick.yml** を作成します。

以下の例を使用して、**/etc/ansible/roles/gluster.infra/playbooks** ディレクトリーに新しい **create_brick.yml** ファイルを作成します。

このファイルは、**gluster.infra** ロールおよび上記で作成した変数ファイルを使用してブリックを作成する作業を定義します。

Playbook ファイル **create_brick.yml** の例

```

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

  roles:
    - gluster.infra

```

4. Playbook の実行

`/etc/ansible/roles/gluster.infra/playbooks` ディレクトリーから以下のコマンドを実行し、インベントリーおよび上記で定義した変数ファイルを使用して作成した Playbook を実行します。

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

5. ブリックが利用可能であることを確認します。

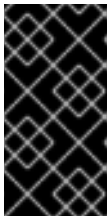
- a. コンピュート → ホスト をクリックして、ホストを選択します。
- b. ストレージ デバイス をクリックし、新しいブリックのストレージデバイスの一覧を確認します。
新しいブリックが表示されない場合は、Sync をクリックし、ストレージデバイスの一覧に表示されるまで待機します。

1.2. ANSIBLE を使用した VDO レイヤーでの新しいブリックの作成

多くのホストでブリックを一度に作成する場合は、ansible Playbook を作成してプロセスを自動化できます。

前提条件

- 新しいブリックをホストする物理マシンをインストールします。
「[物理ホストマシンのインストール](#)」の手順に従います。
- すべてのノード間のパスワードなしで、鍵ベースの SSH 認証を設定します。
Web コンソールを実行しているノードからすべての新規ノードおよび最初の新規ノードから他のすべての新規ノードにこれを設定します。



重要

RHHI for Virtualization は、IP アドレスと FQDN の両方について、これらのノード間のパスワードなしに鍵ベースの SSH 認証を想定します。これらのマシン間の鍵ベースの SSH 認証を、すべてのストレージおよび管理ネットワークインターフェースの IP アドレスと FQDN に設定するようにしてください。

[鍵ベースの認証の使用](#) の手順に従って、パスワードなしで鍵ベースの SSH 認証を設定します。

- ホストが Virtual Disk Optimization(VDO)レイヤーを使用していることを確認します。VDO レイヤーがない場合は、代わりに「[Ansible を使用した新しいブリックの作成](#)」を使用します。

手順

1. インベントリー ファイルを作成します。

以下の例を使用して、`/etc/ansible/roles/gluster.infra/playbooks` ディレクトリーに新規 インベントリー ファイルを作成します。

このファイルは、新規ブリックを作成するホストを一覧表示します。

インベントリー ファイルの例

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

```
server5.example.com
server6.example.com
```

2. ブリック.yml 変数ファイルを作成します。

以下の例を使用して、`/etc/ansible/roles/gluster_infra/playbooks` ディレクトリーに新しいブリック.yml ファイルを作成します。

このファイルは、各ホストで作成または使用される基礎となるストレージインフラストラクチャーおよび設定を定義します。

vdo_bricks.yml 変数ファイルの例

```
# 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' }

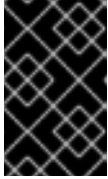
# gluster_infra_volume_groups
# Variables for creating volume group
gluster_infra_volume_groups:
  - { vgname: 'vg_vdb', pvname: '/dev/mapper/hc_vdo_1' }
  - { vgname: 'vg_vdc', pvname: '/dev/mapper/hc_vdo_2' }

# 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', thinpoolsizesize: '10G', poolmetadatasize:
'1G' }
  - {vgname: 'vg_vdc', thinpoolname: 'bar_thinpool', thinpoolsizesize: '20G', poolmetadatasize:
'1G' }
```

```
# gluster_infra_lv_logicalvols
# Thin volumes 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/thickl', vgname: 'vg_vdb', lvname: 'vg_vdb_thickl1' }
  - { path: '/rhgs/thinlv1', vgname: 'vg_vdb', lvname: 'vg_vdb_thinlv' }
  - { path: '/rhgs/thinlv2', vgname: 'vg_vdc', lvname: 'vg_vdc_thinlv' }
```



重要

path: 定義された path: が **/rhgs** で開始されていない場合、ブリックは管理ポータルによって自動的に検出されません。 **create_brick.yml** Playbook を実行し、新しいブリックを管理ポータルに追加した後にホストストレージを同期します。

3. Playbook ファイル **create_brick.yml** を作成します。

以下の例を使用して、**/etc/ansible/roles/gluster.infra/playbooks** ディレクトリーに新しい **create_brick.yml** ファイルを作成します。

このファイルは、**gluster.infra** ロールおよび上記で作成した変数ファイルを使用してブリックを作成する作業を定義します。

Playbook ファイル **create_brick.yml** の例

```
---
- 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. Playbook の実行

/etc/ansible/roles/gluster.infra/playbooks ディレクトリーから以下のコマンドを実行し、インベントリーおよび上記で定義した変数ファイルを使用して作成した Playbook を実行します。

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

5. ブリックが利用可能であることを確認します。

- a. コンピュート → ホスト をクリックして、ホストを選択します。
- b. ストレージ デバイス をクリックし、新しいブリックのストレージデバイスの一覧を確認します。
新しいブリックが表示されない場合は、**Sync** をクリックし、ストレージデバイスの一覧に表示されるまで待機します。

1.3. RED HAT VIRTUALIZATION MANAGER からのボリュームの拡張

このセクションに従って、新しいハイパーコンバージドノードの新規ブリック全体に既存のボリュームを拡張します。

前提条件

- スケーリング計画がサポートされていることを確認します ([スケーリングの要件](#))。
- 既存のデプロイメントで、暗号化に認証局が署名した証明書を使用する場合は、新規ノードに必要な証明書を準備します。
- 新しいハイパーコンバージドノードとして機能する 3 台の物理マシンをインストールします。
「[物理ホストマシンのインストール](#)」の手順に従います。
- パスワードなしで鍵ベースの SSH 認証を設定します。
Web コンソールを実行しているノードからすべての新規ノードおよび最初の新規ノードから他のすべての新規ノードにこれを設定します。



重要

RHHI for Virtualization は、IP アドレスと FQDN の両方について、これらのノード間のパスワードなしに鍵ベースの SSH 認証を想定します。これらのマシン間の鍵ベースの SSH 認証を、すべてのストレージおよび管理ネットワークインターフェースの IP アドレスと FQDN に設定するようにしてください。

[鍵ベースの認証の使用](#) の手順に従って、パスワードなしで鍵ベースの SSH 認証を設定します。

手順

1. 新規ブリックの作成

「[ansible を使用したブリックの作成](#)」または「[要件に応じて VDO レイヤーの上にブリックの作成](#)」の手順に従い、ボリュームを拡張するサーバーにブリックを作成します。



重要

path: 定義された path: が `/rhgs` で開始されていない場合、ブリックは管理ポータルによって自動的に検出されません。**create_brick.yml** Playbook を実行し、新しいブリックを管理ポータルに同期した後にホストストレージを同期します。

1. **コンピュー**ト → **ホスト** をクリックして、ホストを選択します。
2. **ストレージデバイス** をクリックします。
3. **同期** をクリックします。

新しいブリックを持つホストごとに繰り返します。

2. ボリュームへの新しいブリックの追加

- a. RHV 管理コンソールにログインします。
- b. **Storage** → **Volumes** の順にクリックし、展開するボリュームを選択します。

- c. **Bricks** タブをクリックします。
- d. **Add** をクリックします。 **Add Bricks** ウィンドウが開きます。
- e. 新しいブリックを追加します。
 - i. **Host** ドロップダウンメニューからブリックホストを選択します。
 - ii. **Brick Directory** ドロップダウンメニューから追加するブリックを選択し、**Add** をクリックします。
- f. すべてのブリックが一覧表示されたら、**OK** をクリックしてボリュームにブリックを追加します。

ボリュームは新しいブリックを自動的に同期します。

1.4. WEB コンソールを使用して新規ノードに新規ボリュームを追加してハイパーコンバージドクラスタの拡張

以下の手順に従って、Web コンソールを使用して、新規ノードに新しいボリュームを使用してハイパーコンバージドクラスタを拡張します。

前提条件

- スケーリング計画がサポートされていることを確認します ([スケーリングの要件](#))。
- 既存のデプロイメントで、暗号化に認証局が署名した証明書を使用する場合は、新規ノードに必要な証明書を準備します。
- 新しいハイパーコンバージドノードとして機能する 3 台の物理マシンをインストールします。
『[Red Hat Red Hat Virtualization 用インフラストラクチャーのデプロイ](#)』の手順に従います。
- パスワードなしで鍵ベースの SSH 認証を設定します。
Web コンソールを実行しているノードからすべての新規ノードおよび最初の新規ノードから他のすべての新規ノードにこれを設定します。



重要

RHHI for Virtualization は、IP アドレスと FQDN の両方について、これらのノード間のパスワードなしに鍵ベースの SSH 認証を想定します。これらのマシン間の鍵ベースの SSH 認証を、すべてのストレージおよび管理ネットワークインターフェースの IP アドレスと FQDN に設定するようにしてください。

[鍵ベースの認証の使用](#) の手順に従って、パスワードなしで鍵ベースの SSH 認証を設定します。

手順

1. Web コンソールにログインします。
2. **Virtualization** → **Hosted Engine** をクリックしてから **Manage Gluster** をクリックします。
3. **Expand Cluster** をクリックします。 **Gluster Deployment** ウィンドウが開きます。

- a. **ホスト** タブで、新しいハイパーコンバージドノードの FQDN または IP アドレスを入力し、**Next** をクリックします。

Expand Cluster [X]

Hosts Volumes Bricks Review

① ————— ② ————— ③ ————— ④

Host1 newhost1.example.com

Host2 newhost2.example.com

Host3 ⓘ newhost3.example.com

Cancel < Back Next >

- b. **ボリューム** タブで、作成するボリュームの詳細を指定します。

Expand Cluster [X]

Hosts Volumes Bricks Review

① ————— ② ————— ③ ————— ④

Name Volume Type Arbitrator Brick Dirs

new_volume Replicate /gluster_bricks/new_volume/n...

⊕ Add Volume

Cancel < Back Next >

- c. **Bricks** タブで、Gluster ボリュームの作成に使用するディスクの詳細を指定します。

Expand Cluster
✕

Hosts Volumes Bricks Review

① ————— ② ————— ③ ————— ④

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
<input type="text" value="new_volume"/>	<input type="text" value="sdb"/>	<input type="text" value="500"/>	<input checked="" type="checkbox"/>	<input type="text" value="/gluster_bricks/new_volume"/>	<input type="checkbox"/>

[+ Add Bricks](#)

Configure LV Cache

ⓘ Arbitrator bricks will be created on the third host in the host list.

Cancel < Back Next >

- d. **Review** タブで、生成されたファイルで問題の有無を確認します。問題がなければ、**Deploy** をクリックします。

Expand Cluster
✕

Hosts Volumes Bricks Review

① ————— ② ————— ③ ————— ④

Generated Ansible inventory : /etc/ansible/hc_wizard_inventory.yml
[Edit](#) [Reload](#)

```

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


Cancel < Back Deploy

デプロイメントが完了するまでしばらく時間がかかります。クラフターが正常に展開されたら

ノブロ1ノブトが元ノリノるよじしはつ、時間がかかりヨリ。ノブスツノが正市に放閉せれると、以下の画面が表示されます。

Expand Cluster ✕

Hosts (1) — Volumes (2) — Bricks (3) — Review (4)



Successfully expanded cluster

[Close](#)

[Cancel](#) [← Back](#) [Close](#)

第2章 フェンスポリシーを使用した高可用性の設定

フェンシングにより、クラスターがパフォーマンスおよび可用性ポリシーを適用し、ハイパーコンバージドホストを自動的にリブートして予期せぬホスト障害に対応できます。

Red Hat Gluster Storage 固有のいくつかのポリシーを有効にして、フェンシングアクティビティが Red Hat ブリック（仮想化用の RHHI for Virtualization）インフラストラクチャーデプロイメントでストレージサービスを中断しないようにする必要があります。

これには、クラスターレベルとホストレベルの両方でフェンシングを有効化および設定する必要があります。詳細は、以下のセクションを参照してください。

2.1. クラスターでのフェンシングポリシーの設定

1. 管理ポータルで、**Compute** → **Clusters** をクリックします。
2. クラスターを選択し、**編集** をクリックします。 **Edit Cluster** ウィンドウが開きます。
3. **フェンシングポリシー** タブをクリックします。
4. **フェンシングの有効化** チェックボックスを選択します。
5. 少なくとも以下のフェンシングポリシーのチェックボックスを選択します。
 - gluster ブリックが起動している場合は、フェンシングをスキップ
 - Gluster クォーラムが満たされない場合は、フェンシングをスキップします。

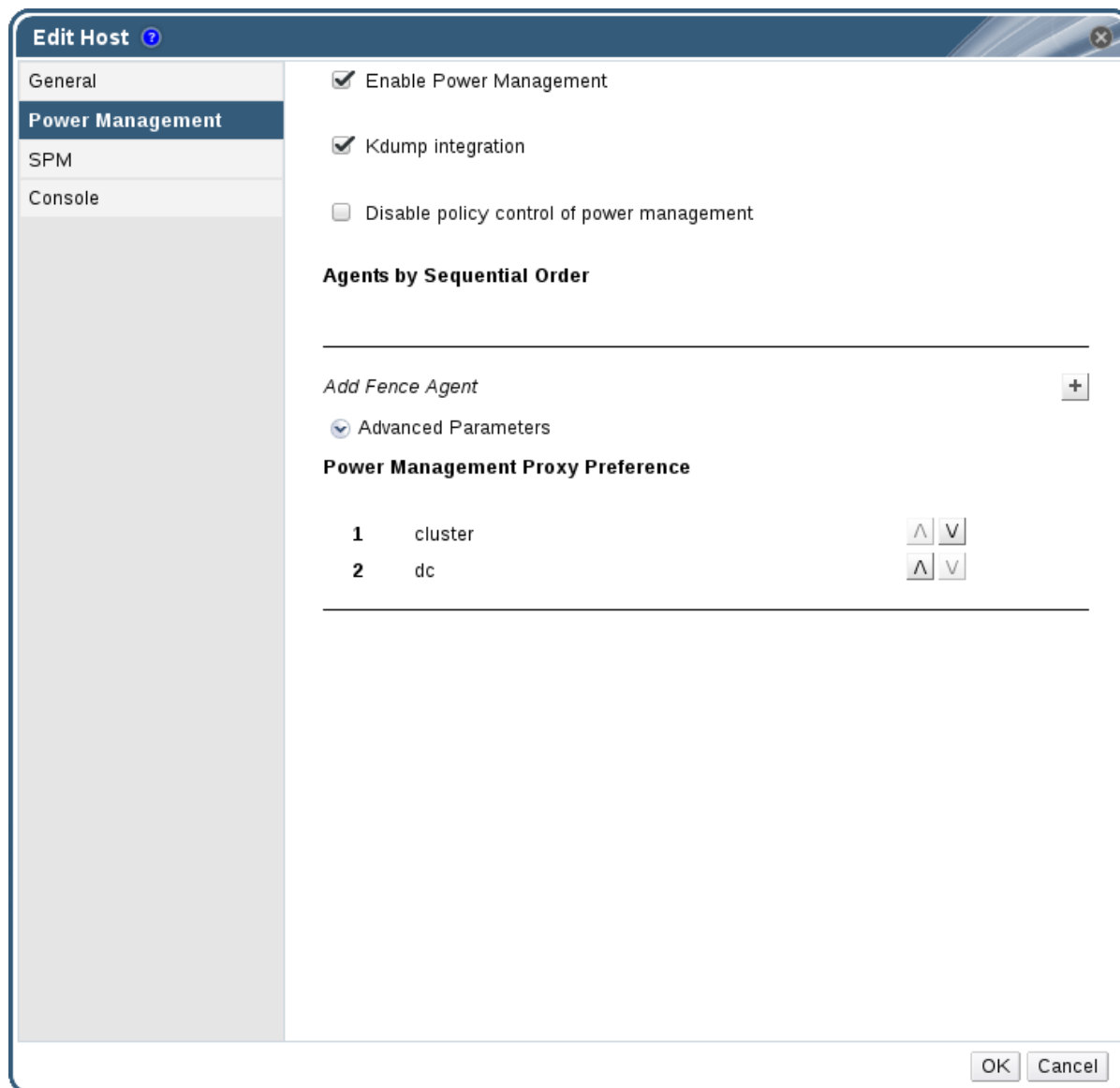
これらのポリシーの影響についての詳細は、[付録A Red Hat Gluster Storage のフェンシングポリシー](#)を参照してください。

6. **OK** をクリックして設定を保存します。

2.2. ホストでのフェンシングパラメーターの設定

1. 管理ポータルで **コンピューター** → **ホスト** をクリックします。
2. 設定するホストを選択し、**編集** をクリックしてホストの **編集 ウィンドウ** を開きます。
3. **Power Management** タブをクリックします。

図2.1 電源管理の設定



4. **電源管理の有効化** チェックボックスにチェックを入れます。これにより、タブの他のフィールドが有効になります。
5. **Kdump 統合** チェックボックスにチェックを入れて、カーネルクラッシュダンプの実行中にホストがフェンシングされないようにします。



重要

既存のホストで Kdump 統合を有効にする場合、kdump を設定するためにホストを再インストールする必要があります。ホストの再インストール方法は、[11章 ハイパーコンバージドホストの再インストール](#)を参照してください。

1. 新しい電源管理デバイスを追加するには、プラス(+)ボタンをクリックします。フェンスエージェントの編集ウィンドウが開きます。

図2.2 フェンスエージェントの編集

Edit fence agent

Address

User Name

Password

Type

SSH Port

Slot

Options

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

Secure

Test

OK Cancel

- a. 電源管理装置のアドレス、ユーザー名、パスワードを入力します。
- b. ドロップダウンリストから電源管理デバイスの種類を選択します。
- a. 電源管理装置がホストとの通信に使用する SSH ポート 番号を入力します。
- b. パワーマネジメントデバイスのブレードを識別するための Slot 番号を入力してください。
- c. パワーマネジメントデバイスのオプションを入力します。key=value エントリーのコンマ区切りリストを使用します。
- d. セキュア チェックボックスにチェックを入れて、電源管理デバイスをホストに安全に接続できるようにします。
- e. Test ボタンをクリックして、設定が正しいことを確認します。テスト Succeeded、Host Status is: on は検証が正常に実行されると表示されます。



警告

電源管理パラメーター（userid、password、オプションなど）は、セットアップ時と Red Hat Virtualization Manager で手動で変更された 2 つの状況で Red Hat Virtualization Manager によってテストされます。誤ったパラメーターに関するアラートを無視するか、Red Hat Virtualization Manager で対応する変更なしに電源管理ハードウェアでパラメーターを変更すると、フェンシングが失敗する可能性が高くなります。

- f. OK をクリックして、フェンスエージェントの追加を終了します。
1. OK をクリックして、ホスト設定を保存します。

ホストのリストに戻ります。ホストの名前の横にある感嘆符(!)が表示され、電源管理が正常に設定されています。

第3章 バックアップオプションおよびリカバリーオプションの設定

本章では、ディスクまたはサーバーの障害後にクラスターを作業状態に復元できるように、Red Hat Warehouse Infrastructure for Virtualization デプロイメントに障害復旧機能を追加する方法を説明します。

3.1. 前提条件

3.1.1. geo レプリケーションの前提条件

geo レプリケーションを設定するには、以下の要件と制限に注意してください。

1つの geo レプリケーションボリュームのみ

Red Hat Converged Infrastructure for Virtualization(RHHI for Virtualization)は、1つの geo レプリケーションボリュームのみをサポートします。通常、これには最も価値のあるデータが含まれているため、Red Hat は仮想マシンのデータを格納するボリュームをバックアップすることを推奨します。

2つの異なるマネージャーが必要です

geo レプリケーションのソースおよび宛先ボリュームは、Red Hat Virtualization Manager の異なるインスタンスで管理される必要があります。

3.1.2. フェイルオーバーおよびフェイルバック設定の前提条件

バージョンが環境間で一致している必要があります

プライマリー環境とセカンダリー環境には、同じバージョンの Red Hat Virtualization Manager が含まれており、同じデータセンターの互換バージョン、クラスターの互換バージョン、および PostgreSQL のバージョンを使用します。

ホスト型エンジンストレージドメインに仮想マシンディスクがない

ホスト型エンジン仮想マシンで使用されるストレージドメインはフェイルオーバーしないため、このストレージドメインの仮想マシンディスクはすべて失われます。

別のマスターノードから Ansible Playbook を手動で実行する

Ansible マスターノードとして機能する別のマシンから Ansible Playbook を手動で生成し、実行します。

3.2. サポート対象のバックアップおよびリカバリー設定

災害復旧機能を Red Hat Warehouse Infrastructure for Virtualization デプロイメントに追加する方法は2つあります。

セカンダリーボリュームへのバックアップのみの設定

リモートのセカンダリーボリュームにデータを定期的に同期することで、ディスクまたはサーバーに障害が発生した場合にデータが失われないようにするのに役立ちます。

このオプションは、以下のステートメントがデプロイメントに当てはまる場合に適しています。

- 障害復旧に必要なデータのバックアップのみが必要です。
- 高可用性ストレージは必要ありません。
- セカンダリークラスターを維持する必要はありません。
- データを手動で復元し、障害の発生後にバックアップソリューションを再設定します。

「[セカンダリーボリュームにバックアップを設定する](#)」の手順に従い、このオプションを設定します。

セカンダリークラスターへのフェイルオーバーとフェイルオーバーの設定

このオプションは、リモートボリュームのデータのバックアップに加えて、フェイルオーバーおよびフェイルバック機能を提供します。プライマリークラスターの操作およびストレージドメインのフェイルオーバーをセカンダリークラスターに設定すると、プライマリークラスターでディスクまたはサーバーに障害が発生した場合にデータが利用可能な状態に保つことができます。

このオプションは、以下のステートメントがデプロイメントに当てはまる場合に適しています。

- 高可用性ストレージが必要です。
- セカンダリークラスターを維持します。
- データを手動で復元したり、障害の発生後にバックアップソリューションを再設定したりしないでください。

[セカンダリークラスターからフェイルオーバーの設定 および フェイルバック の手順](#)に従って、このオプションを設定します。

Red Hat は、実稼働デプロイメント用に少なくともバックアップボリュームを設定することを推奨します。

3.3. セカンダリーボリュームへのバックアップの設定

このセクションでは、geo レプリケーションを使用して、gluster ボリュームをセカンダリー gluster ボリュームにバックアップする方法を説明します。

これを実行するには、以下を行う必要があります。

1. すべての [前提条件](#) を満たしていることを確認します。
2. Geo [レプリケーションターゲット](#)として使用するのに適したボリュームを作成します。
3. ソースボリューム とターゲットボリューム間の [Geo レプリケーションセッションを設定](#) します。
4. Geo レプリケーションプロセスを [スケジュール](#) します。

3.3.1. 前提条件

3.3.1.1. ソースボリュームで共有ストレージを有効にします。

バックアップするボリューム（ソースボリューム）で共有ストレージが有効になっていることを確認します。ソースボリュームをホストするすべてのサーバーで以下のコマンドを実行し、共有ストレージを有効にします。

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

gluster_shared_storage という名前の gluster ボリュームがソースクラスターに作成され、ソースクラスターの全ノードの `/var/run/gluster/shared_storage` にマウントされていることを確認します。詳細は、「[Configuring Up Shared Storage](#)」を参照してください。

3.3.1.2. ソースおよびターゲットボリュームでの暗号化の一致

バックアップするボリュームで暗号化が有効になっている場合は、バックアップデータを保持するボリュームでも暗号化を有効にする必要があります。

詳細は、「[TLS/SSL\(Transport Layer Security\)を使用した暗号化の設定](#)」を参照してください。

3.3.2. Geo レプリケーションに適したターゲットボリュームを作成します。

ソースボリュームの geo-replicated コピーを保持するために、セカンダリー gluster ボリュームを準備します。このターゲットボリュームは、別のサイトでホストされる別のクラスターに置く必要があります。これにより、ソースおよびターゲットボリュームが同じ停止による影響を受けるリスクが最小限に抑えられます。

geo レプリケーションのターゲットボリュームがシャード化が有効にされていることを確認します。ターゲットボリュームをホストするノードで以下のコマンドを実行し、そのボリュームでシャード化を有効にします。

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

3.3.3. ボリュームをバックアップするための geo レプリケーションの設定

3.3.3.1. geo レプリケーションセッションの作成

アクティブなソースボリュームからパッシブターゲットボリュームにデータを複製するには、geo レプリケーションセッションが必要です。



重要

rsync ベースの geo レプリケーションのみが Red Hat IaaS Infrastructure for Virtualization でサポートされます。

1. 一般的な **pem pub** ファイルを作成します。
ターゲットノードに対してパスワードを設定せずに、キーベースの SSH 認証があるソースノードで以下のコマンドを実行します。

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

2. geo レプリケーションセッションの作成
以下のコマンドを実行して、作成した **pem pub** ファイルを使用して、ソースボリュームとターゲットボリューム間に geo レプリケーションセッションを作成します。

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

たとえば、以下のコマンドは、ソースボリューム **prodvol** から **backupvol** というターゲットボリュームに geo レプリケーションセッションを作成します。これは **backup.example.com** がホストします。

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

デフォルトでは、このコマンドは、ターゲットボリュームが利用可能な領域を持つ有効なターゲットであることを確認します。コマンドに **force** オプションを追加して、失敗した検証を無視します。

3. メタボリュームの設定

これは、「[前提条件](#)」で説明されているように、共有ストレージが設定されたソースボリュームに依存します。

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



重要

geo レプリケーションセッションを開始しないでください。Geo レプリケーションセッションを開始すると、ソースボリュームからターゲットボリュームへのレプリケーションが開始されます。

3.3.3.2. Geo レプリケーションセッションの作成の確認

1. 任意のソースノードで管理ポータルにログインします。
2. **Storage** → **Volumes** をクリックします。
3. **Info** 列で geo レプリケーションアイコンを確認します。
このアイコンがある場合、geo レプリケーションはそのボリュームに設定された。

このアイコンが存在しない場合は、[ボリュームの同期を試行](#)します。

3.3.3.3. 管理ポータルを使用したボリュームの状態の同期

1. 管理ポータルにログインします。
2. **コンピュー**ト → **ボリューム** をクリックします。
3. 同期するボリュームを選択します。
4. **Geo-replication** サブタブをクリックします。
5. **同期** をクリックします。

3.3.4. Geo レプリケーションを使用した通常のバックアップのスケジューリング

1. 任意のソースノードで管理ポータルにログインします。
2. **Storage** から **Domains** をクリックします。
3. バックアップするストレージドメインの名前をクリックします。
4. **Remote Data Sync Setup** サブタブをクリックします。
5. **Setup** をクリックします。
Setup Remote Data Synchronization ウィンドウが開きます。
 - a. **Geo-replicated to** フィールドで、バックアップターゲットを選択します。

- b. **Recurrence** フィールドで、繰り返しの間隔を選択します。
有効な値は、最低でも1週間のチェックボックスが選択されている **WEEKLY** または **DAILY** です。
- c. **Hours** フィールドおよび **Minutes** フィールドに、同期を開始する時間を指定します。



注記

この時間は、Hosted Engine のタイムゾーンに基づいています。

- d. **OK** をクリックします。
6. 指定したタイミングでソースボリュームの **Events** サブタブをチェックして、同期が正常に機能していることを確認します。

3.4. セカンダリークラスターへのフェイルオーバーの設定およびフェイルバック

本セクションでは、サーバーに障害が発生した場合に、リモートセカンダリークラスターにフェイルオーバーするようにクラスターを設定する方法を説明します。

これを実行するには、以下を行う必要があります。

1. [リモートボリュームへのバックアップの設定](#)
2. [フェイルオーバーターゲットとして使用するのに適したクラスターを作成](#) します。
3. ソースおよびターゲットクラスターの [マッピングファイルを準備](#) します。
4. [フェイルオーバーの Playbook を準備](#) します。
5. [プライマリークラスターのクリーンアップ Playbook を準備](#) します。
6. [フェイルバック Playbook を作成](#) します。

3.4.1. フェイルオーバーのためのセカンダリークラスターの作成

障害が発生したときにプライマリークラスターの代わりに使用できるセカンダリークラスターをインストールおよび設定します。

このセカンダリークラスターは、以下の設定のいずれかになります。

Red Hat Hyperconverged Infrastructure

詳細は、『[Red Hat ハイパーコンバージドインフラストラクチャー のデプロイ](#)』を参照してください。

Red Hat Virtualization ストレージドメインとして使用するよう設定された Red Hat Gluster Storage

詳細は、『[Configuring Red Hat Virtualization with Red Hat Gluster Storage](#)』を参照してください。このユースケースにはストレージドメインを作成する必要はありません。ストレージドメインはフェイルオーバープロセスの一環としてインポートされます。

セカンダリークラスターのストレージはデータセンターにアタッチしないでください。これにより、フェイルオーバープロセス中にセカンダリーサイトのデータセンターに追加できるようになります。

3.4.2. ソースおよびターゲットクラスター間のマッピングファイルの作成

このセクションでは、ソースクラスターのストレージをターゲットクラスターのストレージにマップするファイルを作成します。

Red Hat では、最初にストレージをデプロイする直後にこのファイルを作成し、デプロイメントの変更時に最新の状態に保つことを推奨します。これは、障害の発生時にクラスター内のすべてのセキュリティーを安全に失敗させるのに役立ちます。

1. マッピングファイルを生成する Playbook を作成します。
サイト、ユーザー名、パスワード、および ca 変数を使用して、クラスターに関する情報を **oVirt.disaster-recovery** ロールに渡す Playbook を作成します。

Red Hat は、**ansible** を提供し、フェイルオーバーおよびフェイルバックを管理するサーバーの **/usr/share/ansible/roles/oVirt.disaster-recovery** ディレクトリーにこのファイルを作成することを推奨します。

Playbook ファイルの例 : 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.disaster-recovery
```

2. **generate_mapping** タグを指定して Playbook を実行してマッピングファイルを生成します。

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

これにより、マッピングファイル **failure_recovery_vars.yml** が作成されます。

3. **disaster_recovery_vars.yml** を編集し、セカンダリークラスターに関する情報を追加します。
マッピングファイルで使用される属性の詳細は、『Red Hat Virtualization 災害復旧ガイド』の「付録 A: マッピングファイルの属性」を参照してください。

3.4.3. ソースおよびターゲットクラスター間のフェイルオーバー Playbook の作成

dr_target_host および **dr_source_map** 変数を使用して、ハイパーコンバージドホストの一覧をフェイルオーバーソースとして使用し、ターゲットを **oVirt.disaster-recovery** ロールに渡す Playbook ファイルを作成します。

Red Hat は、**ansible** を提供し、フェイルオーバーおよびフェイルバックを管理するサーバーの **/usr/share/ansible/roles/oVirt.disaster-recovery** ディレクトリーにこのファイルを作成することを推奨します。

Playbook ファイルの例 : 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.disaster-recovery

```

フェイルオーバーの実行に関する情報は、「[セカンダリクラスターへのフェイルオーバー](#)」を参照してください。

3.4.4. プライマリークラスターのフェイルオーバークリーンアップ Playbook の作成

プライマリークラスターをクリーンアップし、フェイルバックターゲットとして使用できるようにする Playbook ファイルを作成します。

Red Hat は、**ansible** を提供し、フェイルオーバーおよびフェイルバックを管理するサーバーの `/usr/share/ansible/roles/oVirt.disaster-recovery` ディレクトリーにこのファイルを作成することを推奨します。

Playbook ファイルの例 : dr-cleanup.yml

```

---
- name: Clean RHV
  hosts: localhost
  connection: local
  vars:
    dr_source_map: primary
  vars_files:
    - disaster_recovery_vars.yml
  roles:
    - oVirt.disaster-recovery

```

フェイルバックの実行に関する情報は、「[プライマリークラスターにフェイルバック](#)」を参照してください。

3.4.5. ソースおよびターゲットクラスター間でフェイルバック Playbook を作成します。

dr_target_host および **dr_source_map** 変数を使用して、ハイパーコンバージドホストの一覧をフェイルバックソースとして使用し、ターゲットを **oVirt.disaster-recovery** ロールに渡す Playbook ファイルを作成します。

Red Hat は、**ansible** を提供し、フェイルオーバーおよびフェイルバックを管理するサーバーの `/usr/share/ansible/roles/oVirt.disaster-recovery` ディレクトリーにこのファイルを作成することを推奨します。

Playbook ファイルの例 : 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.disaster-recovery
```

フェイルバックの実行に関する情報は、「[プライマリークラスターにフェイルバック](#)」を参照してください。

第4章 TLS/SSL(TRANSPORT LAYER SECURITY)を使用した暗号化の設定

トランスポート層セキュリティ(TLS/SSL)を使用して、ノード間の管理およびストレージ層の通信を暗号化することができます。これにより、データがプライベートのままになるようにすることができます。

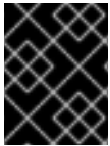
暗号化は、自己署名証明書または認証局が署名した証明書のいずれかを使用して設定できます。

本書では、既存のデプロイメントで暗号化を有効にすることを前提としています。ただし、暗号化はデプロイメントプロセスの一部として設定することもできます。詳細は、「[仮想化用の Red Hat IaaS インフラストラクチャーのデプロイ](#)」を参照してください。

<https://access.redhat.com/documentation/ja->

[jp/red_hat_hyperconverged_infrastructure_for_virtualization/1.7/html/deploying_red_hat_hyperconverged](https://access.redhat.com/documentation/ja-jp/red_hat_hyperconverged_infrastructure_for_virtualization/1.7/html/deploying_red_hat_hyperconverged)

4.1. 自己署名証明書を使用した TLS/SSL の設定



重要

暗号化の有効化または無効化は、仮想マシンおよび Hosted Engine のシャットダウンが必要な中断プロセスです。

1. すべての仮想マシンをシャットダウンする
詳細は、Red Hat Virtualization ドキュメントの「[仮想マシンのシャットダウン](#)」(https://access.redhat.com/documentation/ja-jp/red_hat_virtualization/4.3/html/virtual_machine_management_guide/chap-administrative_tasks)を参照してください。
2. ホストエンジンストレージドメイン以外のすべてのストレージドメインをメンテナンスモードに移動します。
詳細は、Red Hat Virtualization ドキュメントの「[ストレージドメインのメンテナンスモードへの移行](#)」を参照してください。
3. ホストエンジンをグローバルメンテナンスモードに移行
ホストエンジンをホストするハイパーコンバージドホストで以下のコマンドを実行します。

```
# hosted-engine --set-maintenance --mode=global
```

4. ホストエンジンの仮想マシンをシャットダウンする
ホストエンジンをホストするハイパーコンバージドホストで以下のコマンドを実行します。

```
# hosted-engine --vm-shutdown
```

以下のコマンドを実行して、ホストエンジンがシャットダウンしていることを確認します。

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

5. すべての高可用性サービスを停止します。
すべてのハイパーコンバージドホストで以下のコマンドを実行します。

```
# systemctl stop ovirt-ha-agent
# systemctl stop ovirt-ha-broker
```

-
- 6. ホストエンジンストレージドメインをすべてのハイパーコンバージドホストからアンマウントします。

```
# hosted-engine --disconnect-storage
```

- 7. すべてのボリュームがアンマウントされていることを確認します。
各ハイパーコンバージドホストで、すべての gluster ボリュームがマウントされていないことを確認します。

```
# mount
```

- 8. 自己署名証明書の準備
Procedure 23.1に従います。『Red Hat Gluster Storage管理ガイド』の「自己署名証明書の準備」を参照してください。

- 9. すべてのボリュームを停止

```
# gluster v stop <VolumeName>
```

- 10. すべてのノードで glusterd を再起動します。

```
# systemctl restart glusterd
```

- 11. すべてのボリュームでの TLS/SSL 暗号化の有効化

```
# gluster volume set <volname> client.ssl on
# gluster volume set <volname> server.ssl on
```

- 12. すべてのホストでアクセスパーミッションを指定します。

```
# gluster volume set <volname> auth.ssl-allow "host1,host2,host3"
```

- 13. すべてのボリュームを起動

```
# gluster v start <VolumeName>
```

- 14. TLS/SSL エラーが発生していないことを確認します。
各物理マシンの `/var/log/glusterfs/glusterd.log` ファイルをチェックして、TLS/SSL 関連のエラーが発生せず、セットアップが正常に完了していることを確認します。

- 15. すべての高可用性サービスを起動する
すべてのハイパーコンバージドホストで以下のコマンドを実行します。

```
# systemctl start ovirt-ha-agent
# systemctl start ovirt-ha-broker
```

- 16. ホストエンジンのグローバルメンテナンスモードから移動

```
# hosted-engine --set-maintenance --mode=none
```

ホストエンジンは、短い待機時間の後に自動的に起動します。

17. ノードが同期するのを待機します。

最初のハイパーコンバインドホストで以下のコマンドを実行し、同期のステータスを確認します。エンジンのステータスが **unknown stale-data** として一覧表示される場合、同期が完了するまで数分かかります。

以下の出力は同期が完了したことを示しています。

```
# hosted-engine --vm-status | grep 'Engine status'
Engine status : {"health": "good", "vm": "up", "detail": "up"}
Engine status : {"reason": "vm not running on this host",
  "health": "bad", "vm": "down", "detail": "unknown"}
Engine status : {"reason": "vm not running on this host",
  "health": "bad", "vm": "down", "detail": "unknown"}
```

18. すべてのストレージドメインをアクティベートします。

マスターストレージドメインを最初にアクティベートし、その後に他のすべてのストレージドメインをアクティブ化します。

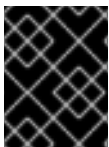
ストレージドメインのアクティブ化の詳細は、Red Hat Virtualization ドキュメントの「**Enabling Storage Domains from Maintenance Mode**」を参照してください(

https://access.redhat.com/documentation/ja-jp/red_hat_virtualization/4.3/html/administration_guide/sect-storage_tasks)。

19. すべての仮想マシンの起動

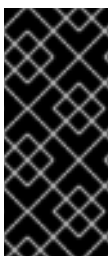
詳細は、Red Hat Virtualization ドキュメントの「**Starting a Virtual Machine**」を参照してください。https://access.redhat.com/documentation/ja-jp/red_hat_virtualization/4.3/html/virtual_machine_management_guide/sect-starting_the_virtual_machine

4.2. 認証局署名証明書を使用した TLS/SSL の設定



重要

暗号化の有効化または無効化は、仮想マシンおよび Hosted Engine のシャットダウンが必要な中断プロセスです。



重要

次のステップに進む前に、認証局が署名した適切な証明書があることを確認してください。証明書の取得については本書では扱いませんが、『Red Hat Gluster Storage Administration Guide:https://access.redhat.com/documentation/ja-jp/red_hat_gluster_storage/3.5/html/administration_guide/chap-network_encryption#chap-Network_Encryption-Prereqs』を参照してください。

1. すべての仮想マシンをシャットダウンする

詳細は、Red Hat Virtualization ドキュメントの「**仮想マシンのシャットダウン**」(https://access.redhat.com/documentation/ja-jp/red_hat_virtualization/4.3/html/virtual_machine_management_guide/chap-administrative_tasks)を参照してください。

2. ホストエンジンストレージドメイン以外のすべてのストレージドメインをメンテナンスモードに移動します。

詳細は、Red Hat Virtualization ドキュメントの「**ストレージドメインのメンテナンスモードへの移行**」を参照してください。

3. ホストエンジンをグローバルメンテナンスモードに移行
ホストエンジンをホストするハイパーコンバージドホストで以下のコマンドを実行します。

```
# hosted-engine --set-maintenance --mode=global
```

4. ホストエンジンの仮想マシンをシャットダウンする
ホストエンジンをホストするハイパーコンバージドホストで以下のコマンドを実行します。

```
# hosted-engine --vm-shutdown
```

以下のコマンドを実行して、ホストエンジンがシャットダウンしていることを確認します。

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

5. すべての高可用性サービスを停止します。
すべてのハイパーコンバージドホストで以下のコマンドを実行します。

```
# systemctl stop ovirt-ha-agent  
# systemctl stop ovirt-ha-broker
```

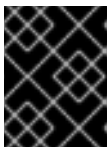
6. ホストエンジンストレージドメインをすべてのハイパーコンバージドホストからアンマウントします。

```
# hosted-engine --disconnect-storage
```

7. すべてのボリュームがアンマウントされていることを確認します。
各ハイパーコンバージドホストで、すべての gluster ボリュームがマウントされていないことを確認します。

```
# mount
```

8. 認証局が署名した暗号化の設定



重要

次のステップに進む前に、認証局が署名した適切な証明書があることを確認してください。証明書の取得については、本書の対象外です。

- a. 全ノードの以下の場所に証明書を配置します。

```
/etc/ssl/glusterfs.key
```

ノードの秘密鍵。

```
/etc/ssl/glusterfs.pem
```

認証局により署名された証明書。ノードの証明書になります。

```
/etc/ssl/glusterfs.ca
```

認証局の証明書。

- b. すべてのボリュームを停止

```
# gluster v stop <VolumeName>
```

- c. すべてのノードで glusterd を再起動します。

```
# systemctl restart glusterd
```

- d. すべてのボリュームでの TLS/SSL 暗号化の有効化

```
# gluster volume set <volname> client.ssl on
# gluster volume set <volname> server.ssl on
```

- e. すべてのホストでアクセスパーミッションを指定します。

```
# gluster volume set <volname> auth.ssl-allow "host1,host2,host3"
```

- f. すべてのボリュームを起動

```
# gluster v start <VolumeName>
```

9. TLS/SSL エラーが発生していないことを確認します。

各物理マシンの `/var/log/glusterfs/glusterd.log` ファイルをチェックして、TLS/SSL 関連のエラーが発生せず、セットアップが正常に完了していることを確認します。

10. すべての高可用性サービスを起動する

すべてのハイパーコンバージドホストで以下のコマンドを実行します。

```
# systemctl start ovirt-ha-agent
# systemctl start ovirt-ha-broker
```

11. ホストエンジンのグローバルメンテナンスモードから移動

```
# hosted-engine --set-maintenance --mode=none
```

ホストエンジンは、短い待機時間の後に自動的に起動します。

12. ノードが同期するのを待機します。

最初のハイパーコンバージドホストで以下のコマンドを実行し、同期のステータスを確認します。エンジンのステータスが **unknown stale-data** として一覧表示される場合、同期が完了するまで数分かかります。

以下の出力は同期が完了したことを示しています。

```
# hosted-engine --vm-status | grep 'Engine status'
Engine status : {"health": "good", "vm": "up", "detail": "up"}
Engine status : {"reason": "vm not running on this host",
  "health": "bad", "vm": "down", "detail": "unknown"}
Engine status : {"reason": "vm not running on this host",
  "health": "bad", "vm": "down", "detail": "unknown"}
```

13. すべてのストレージドメインをアクティベートします。

マスターストレージドメインを最初にアクティベートし、その後に他のすべてのストレージドメインをアクティブ化します。

ストレージドメインのアクティブ化の詳細は、Red Hat Virtualization ドキュメントの「[Enabling Storage Domains from Maintenance Mode](#)」を参照してください。

https://access.redhat.com/documentation/ja-jp/red_hat_virtualization/4.3/html/administration_guide/sect-storage_tasks)。

14. すべての仮想マシンの起動

詳細は、Red Hat Virtualization ドキュメントの「**Starting a Virtual Machine**」を参照してください。https://access.redhat.com/documentation/ja-jp/red_hat_virtualization/4.3/html/virtual_machine_management_guide/sect-starting_the_virtual_machine

第5章 パフォーマンスの向上の設定

一部のデプロイメントは、最適なパフォーマンスを実現するための追加設定の恩恵を受けます。本セクションでは、特定のデプロイメントに推奨される追加の設定を説明します。

5.1. シャードサイズの変更によるボリュームパフォーマンスの向上

バージョン 1.0 から 1.1 の間の **shard-block-size** パラメーターのデフォルト値は、**4MB** から **64 MB** に変更されました。これは、すべての新規ボリュームが **shard-block-size** の値が 64MB で作成されることを意味します。ただし、既存のボリュームには、元の **shard-block-size** 値 4MB を保持します。

データを含むボリュームの **shard-block-size** 値を変更する安全な方法はありません。シャードのブロックサイズは値の設定後に発生する書き込みのみ適用されるため、データが含まれるボリュームの値を変更しようとする、シャードのブロックサイズが混在するため、パフォーマンスが低下します。

本セクションでは、大規模なシャードサイズのパフォーマンス上の利点を活用するために、Red Hat IaaS Infrastructure for Virtualization 1.1 以降に既存のボリュームでシャードブロックサイズを安全に変更する方法を説明します。

5.1.1. レプリケートされたボリュームのシャードサイズの変更

1. インベントリーファイルを作成します。

以下の例に基づいて、**normal_replicated_inventory.yml** という名前のインベントリーファイルを作成します。

host1、**host2**、および **host3** は、ホストの FQDN に置き換え、お使いの環境に合わせてデバイスの詳細を編集します。

normal_replicated_inventory.yml インベントリーファイルの例

```
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:
      # - vgroup: <volgroup_name>
      #   pvname: /dev/mapper/vdo_sdb

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

      gluster_infra_mount_devices:
      - path: <brick_mountpoint>
        lvname: <lv_name>
```

```

    vgroupname: <volgroup_name>

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

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

# Mount the devices
gluster_infra_mount_devices:
  - { path: '<brick_mountpoint>', vgroupname: <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:
  # - vgroupname: <volgroup_name>
  #   pvname: /dev/mapper/vdo_sdb

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

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

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

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

# Mount the devices
gluster_infra_mount_devices:
  - { path: '<brick_mountpoint>', vgroupname: <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:
  # - vgroup: <volgroup_name>
  #   pvname: /dev/mapper/vdo_sdb

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

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

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

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

  # Mount the devices
  gluster_infra_mount_devices:
  - { path: '<brick_mountpoint>', vgroup: <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. `normal_replicated.yml` Playbook を作成します。

以下の例を使用して `normal_replicated.yml` Playbook ファイルを作成します。

`normal_replicated.yml` Playbook の例

```

---

# 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. Playbook の実行

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

5.1.2. 調整されたボリュームでのシャードサイズの変更

1. インベントリーファイルを作成します。

以下の例に基づいて、`arbitrated_replicated_inventory.yml` という名前のインベントリーファイルを作成します。

`host1`、`host2`、および `host3` は、ホストの FQDN に置き換え、お使いの環境に合わせてデバイスの詳細を編集します。

`arbitrated_replicated_inventory.yml` インベントリーファイルの例

```

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:
      # - vgroup: <volgroup_name>
      #   pvname: /dev/mapper/vdo_sdb

```

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

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

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

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

# Mount the devices
gluster_infra_mount_devices:
  - { path: '<brick_mountpoint>', vname: <volgroup_name>, lvname: <lv_name> }
  - { path: '<brick_mountpoint>', vname: <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:
# - vname: <volgroup_name>
#   pvname: /dev/mapper/vdo_sdb

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

gluster_infra_mount_devices:
  - path: <brick_mountpoint>
    lvname: <lv_name>
    vname: <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>

```

```

    lvsizes: <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',
      server.ssl: 'on',
      client.ssl: 'on',
      auth.ssl-allow: '<host1>;<host2>;<host3>'
    }

```

2. **arbitrated_replicated.yml** Playbook を作成します。

以下の例を使用して **arbitrated_replicated.yml** Playbook ファイルを作成します。

arbitrated_replicated.yml Playbook の例

```

---

# 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. Playbook の実行

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

5.2. 既存ボリュームの論理ボリュームキャッシュ(LVMCACHE)の設定

メインストレージデバイスが Solid State Disks(SSD)ではない場合、Red Hat は、仮想化デプロイメントに必要なパフォーマンスを実現するために論理ボリュームキャッシュ(lvmcache)を設定することを推奨します。

1. インベントリーファイルの作成

以下の例に基づいて、**cache_inventory.yml** という名前のインベントリーファイルを作成します。

<host1>、<host2 >、および <host3 > を、キャッシュを設定するホストの FQDN に置き換えます。

ファイル全体で以下の値を置き換えます。

<slow_device>,<fast_device>

キャッシュが接続すべきデバイスを指定し、次にキャッシュデバイスを指定します（例：**cachedisk: '/dev/sdb,/dev/sde'**）。

<fast_device_name>

作成するキャッシュ論理ボリュームの名前を指定します（例：**cachelv_thinpool_gluster_vg_sde**）。

<fast_device_thinpool>

作成するキャッシュシンプールの名前を指定します（例：**gluster_thinpool_gluster_vg_sde**）。

cache_inventory.yml ファイルの例

```
hc_nodes:
  hosts:
    # Host1
    <host1>:
      gluster_infra_cache_vars:
        - vname: 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:
        - vname: 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:
        - vname: gluster_vg_sdb
          cachedisk: '<slow_device>,<fast_device>'
          cachelvname: <fast_device_name>
```

```
cachethinpoolname: <fast_device_thinpool>
cachelsize: '10G'
cachemode: writethrough
```

2. Playbook ファイルを作成します。

lvm_cache.yml という名前の Ansible Playbook ファイルを作成します。

lvm_cache.yml ファイルの例

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

  roles:
    - gluster.infra
```

3. cachesetup タグを使用した Playbook の実行

以下のコマンドを実行して、**lvm_cache.yml** で指定された設定を **cache_inventory.yml** で指定されたホストおよびデバイスに適用します。

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

第6章 モニタリングの設定

6.1. イベント通知の設定

管理ポータルに表示される通知を設定するには、『Red Hat Virtualization 4.3 Administration **Guide**』の「[Configuring Event Notifications in the Administration Portal](#)」を参照してください。

パート II. メンテナンスタスク

第7章 基本的な操作

多くの管理タスクおよびトラブルシューティングタスクには、いくつかの基本的な操作が必要です。本項では、ハイパーコンバージドクラスタのシャットダウンおよび起動などの基本的なタスクを安全に実行する方法を説明します。

7.1. シャットダウン PLAYBOOK の作成

ハイパーコンバージド環境は、特定の順序でシャットダウンする必要があります。最も簡単な方法は、Hosted Engine 仮想マシンから実行できるシャットダウン Playbook を作成することです。

`ovirt.shutdown_env` ロールは Global Maintenance Mode を有効にし、クラスタ内のすべての仮想マシンおよびホストのシャットダウンを開始します。ホストのシャットダウンは非同期です。Playbook は、ハイパーコンバージドホストが実際にシャットダウンする前に終了します。

前提条件

- **ovirt.shutdown_env** Ansible ロールが Hosted Engine 仮想マシンで利用できることを確認します。

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

手順

1. ホストエンジンの仮想マシンにログインします。
2. 環境用のシャットダウン Playbook を作成します。
以下のテンプレートを使用して Playbook ファイルを作成します。
 - **ovirt-engine.example.com** を、Hosted Engine 仮想マシンの FQDN に置き換えます。
 - **123456** を **admin@internal** アカウントのパスワードに置き換えます。

Playbook ファイルの例： 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.shutdown_env
```

7.2. 仮想化のための RHHI のシャットダウン

ハイパーコンバージド環境は、特定の順序でシャットダウンする必要があります。Ansible Playbook を使用してこのプロセスを自動化し、環境を安全にシャットダウンします。

前提条件

- 「シャットダウン Playbook の作成」で説明されているように [シャットダウン Playbook を作成](#) します。
- **ovirt.shutdown_env** Ansible ロールが Hosted Engine 仮想マシンで利用できることを確認します。

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

手順

1. Hosted Engine 仮想マシンに対してシャットダウン Playbook を実行します。

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

7.3. ハイパーコンバージドクラスタのインストール

ハイパーコンバージドクラスタの起動は、従来のコンピュータまたはストレージクラスタを起動する場合よりも複雑です。以下の手順に従って、ハイパーコンバージドクラスタを安全に起動します。

1. クラスタ内の全ホストの電源を入れます。
2. 必要なサービスが利用可能であることを確認します。
 - a. すべてのホストで **glusterd** サービスが正常に起動していることを確認します。

```
# 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
   [...]

```

glusterd が起動していない場合は、起動します。

```
# systemctl start glusterd
```

- b. ホストネットワークが利用可能であり、ホストに IP アドレスが必要なインターフェースに割り当てられていることを確認します。

```
# ip addr show
```

- c. すべてのホストがストレージクラスタの一部であることを確認します（**Connected** で **Peer** としてリストされている）。

```
# 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. すべてのブリックがオンラインに表示されていることを確認します。

```
# gluster volume status engine
Status of volume: engine
Gluster process                TCP Port  RDMA Port  Online  Pid
-----
Brick 10.70.37.28:/gluster_bricks/engine/engine 49153    0          Y      23160
Brick 10.70.37.29:/gluster_bricks/engine/engine 49160    0          Y      12392
Brick 10.70.37.30:/gluster_bricks/engine/engine 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. ホストエンジンの仮想マシンを起動します。
- a. ホストエンジンノードとして使用するホストで以下のコマンドを実行します。

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

- b. ホストエンジンの仮想マシンが正しく起動していることを確認します。

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

4. ホストエンジンの仮想マシンを Global Maintenance モードから外します。
- a. 管理ポータルにログインします。
- b. **Compute** → **Hosts** をクリックし、Hosted Engine ノードを選択します。
- c. pidgin → **Disable Global HA Maintenance** をクリックします。
5. Web コンソールを使用してその他の仮想マシンを起動します。
- a. **Compute** → **Virtualization** をクリックします。
- b. 起動する仮想マシンを選択し、**Run** をクリックします。

第8章 仮想化向けの RED HAT IAAS INFRASTRUCTURE の監視

8.1. VDO(VIRTUAL DATA OPTIMIZER)の監視

VDO の監視は、物理ストレージの容量が不足しているときに理解するのに役立ちます。VDO の物理領域は、シンプロビジョニングストレージのように監視する必要があります。論理領域が多いため、VDO デバイスはシンプロビジョニングを使用する必要があります。VDO 領域はより効果的な方法で使用されます。デフォルトでは、シンプロビジョニングは有効になっており、必要に応じて選択を解除することができます。

View Details をクリックして、利用可能なブロック、使用する領域、およびデバイス情報を確認できます。

8.1.1. コマンドラインインターフェースを使用した VDO の監視

コマンドラインインターフェースを使用して VDO を監視するオプションは複数あります。

vdostats コマンド

このコマンドは、VDO ボリュームで利用可能なブロック、使用されるブロックの数、デバイス名、保存した物理ブロックの割合、VDO ボリュームの物理ブロックの割合など、ボリューム統計を表示します。vdostats の詳細は、man ページの **man vdostats** を参照してください。

vdo status コマンド

このコマンドは、VDO システムおよびボリュームのステータスを YAML 形式で報告します。

/sys/kvdo/<vdo_volume>/statistics ディレクトリー

このディレクトリーのファイルには、VDO のボリューム統計が含まれます。**vdostats** コマンドを使用する代わりに、これらのファイルを読み取ることができます。

8.1.2. Web コンソールを使用した VDO の監視

VDO の使用に関連するイベントは **Notifications** タブに表示されます。イベントでは、VDO ボリュームにある残りの物理領域に関する情報が提供され、物理領域がさらに必要かどうかは最新の状態に保たれます。

表8.1 イベント通知の種類

タイプ	テキスト	アクション
-----	------	-------

タイプ	テキスト	アクション
警告	警告で、ディスク領域が未確認です。StorageDomainName ドメインには、ディスク領域を確認する DiskSpace GB があります。	<ul style="list-style-type: none">● 必要のないデータを削除します。● 既存のディスクを大きなディスクに置き換えます。● Red Hat Gluster Storage サーバーをさらに追加し、新しいサーバー全体でボリュームを拡張します。● ディスクを追加して、Gluster ボリュームの基礎となる論理ボリュームを展開します。

第9章 FSTRIM を使用してシンプロビジョニングされた論理ボリューム上の領域の解放

fstrim を手動で実行して、未使用の論理ボリューム領域をシンプールに返し、他の論理ボリュームで使用できるようにします。

Red Hat は、**fstrim** を毎日実行することを推奨します。

前提条件

- シンプール論理ボリュームが破棄動作に対応していることを確認します。基礎となるデバイスの以下のコマンドの出力がゼロでない場合は、破棄がサポートされます。

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

手順

- fstrim** を実行して、物理領域をシンプールに復元します。

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

たとえば、以下のコマンドは、**/gluster_bricks/data/data** にマウントされた論理ボリュームで見つかった未使用の領域を破棄し、詳細な出力(-v)を提供します。

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

関連情報

- 自動的に繰り返されるタスクを設定する方法についての詳細は、[cron を使用した繰り返しジョブのスケジュール設定](#)について参照してください。

第10章 RED HAT VIRTUALIZATION MANAGER へのハイパーコンバージドホストの追加

このプロセスに従い、Red Hat Virtualization Manager が既存のハイパーコンバージドホストを管理できるようにします。

1. 管理ポータルにログインします。
2. **Compute** → **Hosts** をクリックします。
3. **New** をクリックします。 **新規ホスト 画面**が開きます。
4. **全般** タブで、ハイパーコンバージドホストに関する以下の情報を指定します。
 - **Host Cluster**
 - **名前**
 - **Hostname**
 - **パスワード**
5. **General** タブで、**Advanced Parameters** ドロップダウンをクリックし、**Automatically configure host firewall** チェックボックスの選択を解除します。
6. **OK** をクリックします。

第11章 ハイパーコンバージドホストの再インストール

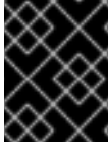
一部の設定の変更では、設定変更を有効にするには、ハイパーコンバージドホストを再インストールする必要があります。以下の手順に従って、ハイパーコンバージドホストを再インストールします。

1. 管理ポータルにログインします。
2. **Compute** → **Hosts** をクリックします。
3. ホストを選択し、**管理** > **メンテナンス** > **OK** をクリックして、このホストをメンテナンスモードに切り替えます。
4. **インストール** > **再インストール** をクリックして、再インストール画面を開きます。
5. General タブで、**Automatically Configure Host firewall** チェックボックスの選択を解除します。
6. **Hosted Engine** タブで、**Choose hosted engine deployment action** の値を **Deploy** に設定します。
7. **OK** をクリックして、ホストを再インストールします。

第12章 ホストの置き換え

12.1. ANSIBLE を使用したプライマリーハイパーコンバージドホストの置き換え

本項では、すべてのデプロイメント操作の実行に使用したハイパーコンバージドホストを置き換えます。



重要

自己署名暗号化が有効な場合には、ノードを置き換えるという中断プロセスで、仮想マシンおよび Hosted Engine のシャットダウンが必要になります。

1. (オプション) 認証局を使用した暗号化が有効な場合は、『Red Hat Gluster Storage 3.5 Administration Guide』の「Expanding Volumes in the [Network Encryption](#)」の章に記載の手順に従ってください。
2. サーバーを移動してメンテナンスモードに切り替えます。
 - a. 管理ポータルで **コンピュー**ト → **ホスト** をクリックし、置き換えるホストを選択します。
 - b. **管理** → **メンテナンス** をクリックし、**OK** をクリックしてホストをメンテナンスモードに移動します。
3. 代替ホストをインストールします。
「仮想化向けの Red Hat ハイパーコンバージドインフラストラクチャーのデプロイ」の手順に従い、物理マシンをインストールし、ストレージを新規ホストに設定します。
 - a. [ハイパーコンバージドホストのインストール](#)
 - b. [公開鍵ベースの SSH 認証の設定](#)
4. 代替ホストの設定
「[Ansible を使用した代替ハイパーコンバージドホストの準備](#)」に記載の手順に従います。
5. (オプション) 自己署名証明書を使用した暗号化が有効な場合は、以下を行います。
 - a. 代替ホストで秘密鍵と自己署名証明書を生成します。詳しくは、『Red Hat Gluster Storage Administration Guide』を参照してください：
https://access.redhat.com/documentation/ja-jp/red_hat_gluster_storage/3.5/html/administration_guide/chap-network_encryption#chap-Network_Encryption-Prereqs
 - b. 正常なホストで、`/etc/ssl/glusterfs.ca` ファイルのコピーを作成します。


```
# cp /etc/ssl/glusterfs.ca /etc/ssl/glusterfs.ca.bk
```
 - c. 新しいホストの証明書を元の `/etc/ssl/glusterfs.ca` ファイルの内容に追加します。
 - d. 新規ホストを含む、`/etc/ssl/glusterfs.ca` ファイルをクラスター内のすべてのホストに配布します。
 - e. 代替のホストで以下のコマンドを実行し、管理暗号化を有効にします。

```
# touch /var/lib/glusterd/secure-access
```

- f. 各ボリュームについて以下のコマンドを実行して、新しいホストを `auth.ssl-allow` ボリュームオプションの値に追加します。

```
# gluster volume set <volname> auth.ssl-allow "<old_host1>,<old_host2>,<new_host>"
```

- g. すべてのホストで `glusterd` サービスを再起動します。

```
# systemctl restart glusterd
```

- h. 「[自己署名証明書を使用した TLS/SSL の設定](#)」の手順に従って、すべての `gluster` プロセスを再マウントします。
6. 置き換えホストをクラスターに追加します。
クラスターにすでにあるホストから以下のコマンドを実行します。

```
# gluster peer probe <new_host>
```

7. ホストエンジンをメンテナンスモードに切り替えます。
クラスターにすでにあるホストから以下のコマンドを実行します。

```
# hosted-engine --set-maintenance --mode=global
```

8. `ovirt-engine` サービスを停止します。
RHV Manager が実行されている Hosted Engine 仮想マシンで以下のコマンドを実行します。

```
# systemctl stop ovirt-engine
```

9. データベースを更新します。
クラスターにすでにあるホストから以下のコマンドを実行します。

```
# hosted-engine --set-shared-config storage <new_host_IP>:/engine --type=he_local
```

```
# hosted-engine --set-shared-config storage <new_host_IP>:/engine --type=he_shared
```

10. `ovirt-engine` サービスを起動します。
Hosted Engine 仮想マシンで以下のコマンドを実行します。

```
# systemctl start ovirt-engine
```

11. ホストエンジン以外の仮想マシンをすべて停止します。
12. Hosted Engine ドメイン 以外 のすべてのストレージドメインをメンテナンスモードに移動します。
13. メンテナンスモードのすべてのストレージドメインの接続詳細を更新します。
- a. 管理ポータルで ストレージ → ドメイン → ストレージドメイン を選択します。

- b. 右上の Add Domain ボタンをクリックして、新しい置換ホストでパスを更新します。
 - c. OK をクリックします。
 - d. メンテナンスモードのすべてのストレージドメインに対して、上記の 3 つの手順を繰り返します。
14. Hosted Engine ストレージの接続を更新します。
Hosted Engine 仮想マシンで以下のコマンドを実行します。

```
# /usr/share/ovirt-engine/dbscripts/engine-psql.sh -c "UPDATE
storage_server_connections SET connection = '<new_server_FQDN>:/engine' WHERE
id = (SELECT storage FROM storage_domains WHERE is_hosted_engine_storage =
't');"

```

15. Hosted Engine 仮想マシンを停止します。
ホストエンジンをホストする既存のサーバーで以下のコマンドを実行します。

```
# hosted-engine --vm-shutdown

```

16. すべてのホストで高可用性サービスを停止します。

```
# systemctl stop ovirt-ha-agent
# systemctl stop ovirt-ha-broker

```

17. Hosted Engine ストレージをハイパーコンバージドホストから切断します。
すべてのホストで以下のコマンドを実行します。

```
# hosted-engine --disconnect-storage

```

18. Hosted Engine 設定ファイルを更新します。
/etc/ovirt-hosted-engine/hosted-engine.conf ファイルの storage パラメーターを編集して、代替のホストを使用します。

```
storage=<new_server_IP>:/engine

```

19. すべてのホストで高可用性サービスを再起動します。

```
# systemctl restart ovirt-ha-agent
# systemctl restart ovirt-ha-broker

```

20. 既存のホストおよび代替ホストを再起動します。
すべてのホストが利用可能になるまで待機してから続行します。

21. ホストエンジンをメンテナンスモードから外します。
ホストのいずれかで以下のコマンドを実行します。

```
# hosted-engine --set-maintenance --mode=none

```

22. 代替ホストが使用されていることを確認します。
すべてのハイパーコンバージドホストで、mount コマンドの出力にある IP アドレスをチェックして、エンジン ボリュームが交換ホストからマウントされていることを確認します。

23. RHV 管理ポータルからストレージドメインをアクティベートします。
ストレージドメインが代替ホストの FQDN/IP アドレスを使用してマウントされていることを確認します。
24. RHV 管理ポータルを使用して、代替ホストを Default クラスタに追加します。
 - a. コンピュート → ホスト → ホスト → New ボタン → 新規ホストの詳細を入力します。
 - b. OK をクリックします。
25. RHV 管理ポータルから、切り替えるホストをメンテナンスモードに移動します。
26. Gluster ボリュームブリックを置き換えます。
ボリュームに属する以前のホストのブリックを、交換ホストの新しいブリックに置き換えます。
 - a. Storage → Volumes の順にクリックし、ボリュームを選択します。
 - b. Bricks サブタブをクリックします。
 - c. 置き換えるブリックを選択し、置換 をクリックします。
 - d. 置き換えるブリックをホストするホストを選択します。
 - e. Replaceブリック ウィンドウで、新しいブリックへのパスを指定します。
 - f. このクラスタ内のすべてのボリュームについて、上記の5つの手順を繰り返します。
27. 古いホストを削除します。
 - a. コンピュート → ホスト をクリックして、古いホストを選択します。
 - b. 管理 → メンテナンス をクリックして、ホストをメンテナンスモードに移動します。
 - c. 削除 をクリックします。ホストの削除の確認ダイアログが表示されます。
 - d. このホストにボリュームブリックがまだある場合や、ホストが応答しない場合は、Force Remove チェックボックスにチェックを入れます。
 - e. OK をクリックします。
 - f. 古いホストをクラスタからデタッチします。

```
# gluster peer detach <old_host_IP> force
```

28. すべてのホストで、以下のコマンドを実行して以前のホストからメタデータを削除します。

```
# hosted-engine --clean-metadata --host-id=<old_host_id> --force-clean
```

12.2. ANSIBLE を使用した他のハイパーコンバージドホストの置き換え

最初のホストではないハイパーコンバージドホストを置き換える方法は2つあります。

1. 「別の FQDN を使用するようにハイパーコンバージドホストの置き換え」の説明に従って、別の完全修飾ドメイン名を持つ新規ホストに置き換えます。

2. 「同じ FQDN を使用するようにハイパーコンバージドホストを置き換える」の説明に従って、同じ完全修飾ドメイン名を持つ新規ホストに置き換えます。

デプロイメントに適したセクションの手順に従います。

12.2.1. 別の FQDN を使用するようにハイパーコンバージドホストの置き換え



重要

自己署名暗号化が有効な場合には、ノードを置き換えるという中断プロセスで、仮想マシンおよび Hosted Engine のシャットダウンが必要になります。

1. 代替ホストをインストールします。
「仮想化向けの Red Hat Warehouse インフラストラクチャーのデプロイ」に記載の手順に従って、物理マシンをインストールします。

- a. [ホストの物理マシンのインストール](#)
- b. [公開鍵ベースの SSH 認証の設定](#)

2. 既存の geo レプリケーションセッションを停止する

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

詳細は、『Red Hat Gluster Storage Administration Guide:https://access.redhat.com/documentation/ja-jp/red_hat_gluster_storage/3.5/html/administration_guide/sect-starting_geo-replication#Stopping_a_Geo-replication_Session』を参照してください。

3. 置換するホストをメンテナンスモードに切り替えます。
管理ポータルで以下の手順を実行します。
 - a. コンピュート → ホスト をクリックし、結果一覧でハイパーコンバージドホストを選択します。
 - b. 管理 → メンテナンス をクリックし、OK をクリックしてホストをメンテナンスモードに移動します。
4. 代替ホストの準備
 - a. パスワードなしで鍵ベースの SSH 認証を設定する
物理マシンのパスワードなしで鍵ベースの SSH 認証をクラスター内に設定して、交換ホストに設定します。詳細は、https://access.redhat.com/documentation/ja-jp/red_hat_hyperconverged_infrastructure_for_virtualization/1.7/html/deploying_red_hat_configure-key-based-ssh-auth を参照してください。
 - b. 代替ホストの準備
「[Ansible を使用した代替ハイパーコンバージドホストの準備](#)」に記載の手順に従います。
5. 代替となるブリックディレクトリーを作成します。
新しいディレクトリーが vdsmd ユーザーおよび kvm グループによって所有されていることを確認します。

```
# mkdir /gluster_bricks/engine/engine
# chmod vdsmd:kvm /gluster_bricks/engine/engine
```

```
# mkdir /gluster_bricks/data/data
# chmod vdsm:kvm /gluster_bricks/data/data
# mkdir /gluster_bricks/vmstore/vmstore
# chmod vdsm:kvm /gluster_bricks/vmstore/vmstore
```

6. (オプション) 暗号化が有効な場合

- a. 『Red Hat Gluster Storage Administration Guide』の「[:https://access.redhat.com/documentation/ja-jp/red_hat_gluster_storage/3.5/html/administration_guide/chap-network_encryption#chap-Network_Encryption-Prereqs](https://access.redhat.com/documentation/ja-jp/red_hat_gluster_storage/3.5/html/administration_guide/chap-network_encryption#chap-Network_Encryption-Prereqs)」の手順を使用して、新しいサーバーで秘密鍵と自己署名証明書を生成します。
認証局を使用した暗号化が有効な場合は、『Red Hat Gluster Storage 3.5 Administration Guide』の「Expanding Volumes in the [Network Encryption](#)」の章に記載の手順に従います。
- b. 新しいホストの証明書を既存の証明書に追加します。
 - i. 正常なホストで、`/etc/ssl/glusterfs.ca` ファイルのバックアップコピーを作成します。
 - ii. 新規ホストの証明書を正常なホストの `/etc/ssl/glusterfs.ca` ファイルに追加します。
 - iii. 更新した `/etc/ssl/glusterfs.ca` ファイルを、新規ホストを含む他のすべてのホストに配布します。
- c. 管理暗号化の有効化
新しいホストで以下のコマンドを実行し、管理暗号化を有効にします。

```
# touch /var/lib/glusterd/secure-access
```

- d. 各ボリュームについて以下のコマンドを実行して、新しいホストを `auth.ssl-allow` ボリュームオプションの値に追加します。

```
# gluster volume set <volname> auth.ssl-allow "<old_host1>,<old_host2>,<new_host>"
```

- e. すべてのホストで `glusterd` サービスを再起動します。

```
# systemctl restart glusterd
```

- f. 暗号化で自己署名証明書を使用している場合は、「[自己署名証明書を使用した TLS/SSL の設定](#)」の手順に従って、すべての `gluster` プロセスを再マウントします。

7. 新規ホストを既存クラスターに追加します。

- a. 正常なホストのいずれかから以下のコマンドを実行します。

```
# gluster peer probe <new_host>
```

- b. 新規ホストを既存クラスターに追加します。

- i. コンピュート → ホスト をクリックしてから 新規 をクリックして 新規ホストダイアログを開きます。
- ii. 新規ホストの名前、アドレス、およびパスワードを指定します。

- iii. ファイアウォールルールが gdeploy によってすでに設定されているので、ホストファイアウォールを自動構成チェックボックスのチェックを外します。
 - iv. 新規 ホスト ダイアログの Hosted Engine タブで、Choose hosted engine デプロイメントアクションの値を Deploy に設定します。
 - v. OK をクリックします。
 - vi. ホストが利用できるようになったら、新規ホストの名前をクリックします。
 - vii. ネットワークインターフェースサブタブをクリックし、ホストネットワークの設定をクリックします。ホストネットワークの設定ダイアログが表示されます。
 - viii. gluster 用に作成したネットワークを、このホストに関連付けられた IP にドラッグアンドドロップして、OK をクリックします。
詳細は、『Red Hat Virtualization 4.3 Self-Hosted Engine Guide』を参照してください：
https://access.redhat.com/documentation/ja-jp/red_hat_virtualization/4.3/html/self-hosted_engine_guide/chap-installing_additional_hosts_to_a_self-hosted_environment
8. ストレージドメインマウントオプションの古いホストを置き換えます。
Gluster ストレージドメインは、マウントオプション `backup-volfile-server` を使用して作成されます。これには古いホストの名前が含まれます。古いホストの値は、新しいホスト名に置き換える必要があります。
- a. SSH を使用して Hosted Engine 仮想マシンにログインします。
 - b. gluster ストレージドメインを Unique Identifier(UUID)で一覧表示します。

```
# /usr/share/ovirt-engine/dbscripts/engine-psql.sh -c "SELECT id, connection,
mount_options FROM storage_server_connections WHERE storage_type = 7;"
```

以下に例を示します。

```
# /usr/share/ovirt-engine/dbscripts/engine-psql.sh -c "SELECT id, connection,
mount_options FROM storage_server_connections WHERE storage_type = 7;"
```

```

      id          |      connection      |
-----+-----+
78f68999-b3aa-44ec-b010-0b630a1a0177 | host1.example.com:/engine |
8b287c52-082e-4263-b99f-4654569d623b | host1.example.com:/vmstore |
|          mount_options          |
+-----+-----+
|backup-volfile-servers=old_host.example.com:host3.example.com |
|backup-volfile-servers=old_host.example.com:host3.example.com |

```

上記の例では、UUID 78f68999-b3aa-44ec-b010-0b630a1a0177 および 8b 287c52-082e-4263-b99f-4654569d623b の 2 つの gluster ストレージドメインがあります。

ストレージドメインは、古いホストの値を含む `mount_options` を使用していることに注意してください。

- c. すべての gluster ストレージドメインの `mount_options` を、前の手順で見つかった UUID で更新します。


```
# /usr/share/ovirt-engine/dbscripts/engine-psql.sh -c "UPDATE
storage_server_connections SET mount_options = 'backup-volfile-servers=<new_host>:
<existing_host>' WHERE id = *<UUID>* AND storage_type = 7;"
```

以下に例を示します。

```
# /usr/share/ovirt-engine/dbscripts/engine-psql.sh -c "UPDATE
storage_server_connections SET mount_options = 'backup-volfile-
servers=newhost.example.com:host3.example.com' WHERE id = '78f68999-b3aa-44ec-
b010-0b630a1a0177' AND storage_type = 7;"
UPDATE 1
```

9. ホストエンジン HA 設定の更新

- a. ハイパーコンバージドノードの1つで以下のコマンドを実行して、共有ストレージのホスト型エンジンストレージマウントオプションを更新します。

```
# hosted-engine --set-shared-config mnt_options "backup-volfile-servers=
<newhost.example.com>:<host3>" --type=he_shared
```

以下に例を示します。

```
# hosted-engine --set-shared-config mnt_options "backup-volfile-
servers=newhost.example.com:host3.example.com" --type=he_shared
```

- b. 以下を実行して、各ハイパーコンバージドホストの1つで Hosted Engine ストレージマウントオプションを更新します。

```
# hosted-engine --set-shared-config mnt_options "backup-volfile-servers=
<newhost.example.com>:<host3>" --type=he_local
```

以下に例を示します。

```
# hosted-engine --set-shared-config mnt_options "backup-volfile-
servers=newhost.example.com:host3.example.com" --type=he_local
```

10. 新規ホストでの共有ストレージの設定およびマウント

```
# cp /etc/fstab /etc/fstab.bk
# echo "<new_host>:/gluster_shared_storage /var/run/gluster/shared_storage/
glusterfs defaults 0 0" >> /etc/fstab
# mount /gluster_shared_storage
```

11. 古いブリックを新規ホストのブリックに置き換えます。

- a. 管理ポータルでストレージ → ボリューム をクリックし、ボリュームを選択します。
- b. Bricks サブタブをクリックします。
- c. 置き換えるブリックを選択し、Replace Brick をクリックします。Replace Brick ダイアログが表示されます。
- d. 新しいブリックのホストと Brick ディレクトリーを指定します。

e. ブリックの修復が正常に完了していることを確認します。

12. Compute → Hosts をクリックします。

13. 古いホストを選択し、削除 をクリックします。

`gluster peer status` を使用して、古いホストがクラスターに含まれなくなったことを確認します。古いホストがステータス出力にまだある場合は、以下のコマンドを実行して強制的に削除します。

```
# gluster peer detach <old_host> force
```

14. 古いホストメタデータを消去します。

```
# hosted-engine --clean-metadata --host-id=<old_host_id> --force-clean
```

15. 新しいブリックの geo レプリケーションに新しい SSH キーを設定します。

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

16. Geo レプリケーションセッションを再作成し、新しい SSH キーを配布します。

```
# gluster volume geo-replication <MASTER_VOL> <SLAVE_HOST>::<SLAVE_VOL>
create push-pem force
```

17. geo レプリケーションセッションを開始します。

```
# gluster volume geo-replication <MASTER_VOL> <SLAVE_HOST>::<SLAVE_VOL>
start
```

12.2.2. 同じ FQDN を使用するようにハイパーコンバージドホストを置き換える



重要

自己署名暗号化が有効な場合には、ノードを置き換えるという中断プロセスで、仮想マシンおよび Hosted Engine のシャットダウンが必要になります。

1. (オプション) 認証局を使用した暗号化が有効な場合は、『Red Hat Gluster Storage 3.5 Administration Guide』の「Expanding Volumes in the [Network Encryption](#)」の章に記載の手順に従ってください。
2. 置換するホストをメンテナンスモードに切り替えます。
 - a. 管理ポータルでコンピュート → ホスト をクリックし、ハイパーコンバージドホストを選択します。
 - b. 管理 → メンテナンス をクリックします。
 - c. OK をクリックして、ホストをメンテナンスモードに移動します。
3. 代替ホストをインストールします。
「仮想化向けの Red Hat ハイパーコンバージドインフラストラクチャーのデプロイ」の手順に従い、物理マシンをインストールし、ストレージを新規ホストに設定します。

- a. [ホストの物理マシンのインストール](#)
- b. [公開鍵ベースの SSH 認証の設定](#)
4. 代替ホストの設定
「[Ansible を使用した代替ハイパーコンバージドホストの準備](#)」に記載の手順に従います。
5. (オプション) 自己署名証明書を使用した暗号化が有効な場合
 - a. 代替ホストで秘密鍵と自己署名証明書を生成します。詳しくは、『Red Hat Gluster Storage Administration Guide』を参照してください：
https://access.redhat.com/documentation/ja-jp/red_hat_gluster_storage/3.5/html/administration_guide/chap-network_encryption#chap-Network_Encryption-Prereqs
 - b. 正常なホストで、`/etc/ssl/glusterfs.ca` ファイルのバックアップコピーを作成します。

```
# cp /etc/ssl/glusterfs.ca /etc/ssl/glusterfs.ca.bk
```
 - c. 新しいホストの証明書を `/etc/ssl/glusterfs.ca` ファイルの内容に追加します。
 - d. 新規ホストを含む、`/etc/ssl/glusterfs.ca` ファイルをクラスター内のすべてのホストに配布します。
 - e. 代替のホストで以下のコマンドを実行し、管理暗号化を有効にします。

```
# touch /var/lib/glusterd/secure-access
```
6. ホストマシンの置き換え
『Red Hat Gluster Storage Administration Guide』の手順に従ってホストを置き換えます(https://access.redhat.com/documentation/ja-jp/red_hat_gluster_storage/3.5/html/administration_guide/sect-replacing_hosts#Replacing_a_Host_Machine_with_the_Same_Hostname)。
7. すべてのホストで `glusterd` サービスを再起動します。

```
# systemctl restart glusterd
```
8. すべてのホストが再接続していることを確認します。

```
# gluster peer status
```
9. (オプション) 暗号化で自己署名証明書を使用している場合は、「[自己署名証明書を使用した TLS/SSL の設定](#)」の手順にしたがって、すべての `gluster` プロセスを再マウントします。
10. すべてのホストが再接続し、ブリックの修復が正常に完了していることを確認します。

```
# gluster peer status
```
11. フィンガープリントの更新
 - a. 管理ポータルで コンピュート → ホスト をクリックし、新規ホストを選択します。
 - b. 編集 をクリックします。

- c. 全般 タブで詳細パラメーター をクリックします。
 - d. fetch をクリックして、ホストからフィンガープリントを取得します。
 - e. OK をクリックします。
12. Installation → Reinstall をクリックし、プロンプトが表示されたら root パスワードを入力します。
 13. Hosted Engine タブで、Choose hosted engine deployment action の値を Deploy に設定します。
 14. gluster ネットワークをホストに割り当てます。
 - a. コンピュート → ホスト をクリックし、ホストの名前をクリックします。
 - b. ネットワークインターフェース サブタブをクリックし、ホストネットワークの設定 をクリックします。
 - c. 新規作成されたネットワークを正しいインターフェースにドラッグアンドドロップします。
 - d. ホストとエンジン間の接続の確認 チェックボックスにチェックマークが付いていることを確認します。
 - e. Save network configuration チェックボックスにチェックマークが付いていることを確認します。
 - f. OK をクリックして保存します。
 15. ネットワークの正常性の確認
 ネットワーク インターフェース タブ をクリックして、ホストのネットワークの状態を確認します。ネットワークインターフェースが「Out of sync」状態になるか、IP アドレスがない場合は、Management → Refresh Capabilities をクリックします。

12.3. ANSIBLE を使用した代替ハイパーコンバージドホストの準備

このプロセスに従い、クラスター内のハイパーコンバージドホストを置き換えます。

前提条件

- 置き換えるホストが、新規ホストに使用する FQDN に関連付けられていないことを確認します。
- 新規ホストが、使用する FQDN に関連付けられていることを確認します。

手順

1. node_prep_inventory.yml インベントリーファイルを作成します。
 以下の例に基づいて、node_prep_inventory.yml という名前のインベントリーファイルを作成します。

host1 は新規ホストに使用する FQDN に置き換え、デバイスの詳細をお使いのホストに適した詳細に置き換えます。

node_prep_inventory.yml ファイルの例

```
hc_nodes:
  hosts:
    # New host
    newhost.example.com:

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

    # With Dedupe & Compression
    #gluster_infra_volume_groups:
    # - vname: gluster_vg_sdc
    #   pvname: /dev/mapper/vdo_sdc

    # Without Dedupe & Compression
    gluster_infra_volume_groups:
    - vname: gluster_vg_sdc
      pvname: /dev/sdc

    gluster_infra_mount_devices:
    - path: /gluster_bricks/engine
      lvname: gluster_lv_engine
      vgname: gluster_vg_sdc
    - path: /gluster_bricks/data
      lvname: gluster_lv_data
      vgname: gluster_vg_sdc
    - path: /gluster_bricks/vmstore
      lvname: gluster_lv_vmstore
      vgname: gluster_vg_sdc

    gluster_infra_thinpools:
    - {vgname: 'gluster_vg_sdc', thinpoolname: 'thinpool_gluster_vg_sdc', thinpoolsizesize:
'500G', poolmetadatasize: '4G'}

    # This is optional
    gluster_infra_cache_vars:
    - vgname: gluster_vg_sdc
      cachedisk: /dev/sde
      cachelvname: cachelv_thinpool_vg_sdc
      cachethinpoolname: thinpool_gluster_vg_sdc # cachethinpoolname is equal to the
already created thinpool which you want to attach
      cachelvsize: '10G'
      cachemetalvsize: '2G'
      cachemetalvname: cache_thinpool_vg_sdc
      cachemode: writethrough

    gluster_infra_thick_lvs:
    - vgname: gluster_vg_sdc
      lvname: gluster_lv_engine
      size: 100G

    gluster_infra_lv_logicalvols:
```

```

- vname: gluster_vg_sdc
  thinpool: thinpool_gluster_vg_sdc
  lvname: gluster_lv_data
  lvsize: 500G
- vname: gluster_vg_sdc
  thinpool: thinpool_gluster_vg_sdc
  lvname: gluster_lv_vmstore
  lvsize: 500G

# Mount the devices
gluster_infra_mount_devices:
  - { path: '/gluster_bricks/data', vname: gluster_vg_sdc, lvname: gluster_lv_data }
  - { path: '/gluster_bricks/vmstore', vname: gluster_vg_sdc, lvname: gluster_lv_vmstore
}
  - { path: '/gluster_bricks/engine', vname: gluster_vg_sdc, lvname: gluster_lv_engine }

# Common configurations
vars:
  # Firewall setup
  gluster_infra_fw_ports:
    - 2049/tcp
    - 54321/tcp
    - 5900/tcp
    - 5900-6923/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
  gluster_infra_disktype: RAID6
  gluster_infra_diskcount: 12
  gluster_infra_stripe_unit_size: 128

```

2. **node_prep.yml** Playbook を作成します。
以下の例に基づいて **node_prep.yml** Playbook ファイルを作成します。

node_prep.yml Playbook の例

```

---

# Prepare Node for replace
- name: Setup backend
  hosts: hc_nodes
  remote_user: root
  gather_facts: no
  any_errors_fatal: true

roles:
  - gluster.infra
  - gluster.features

```

3. **node_prep.yml** Playbook の実行

```
# ansible-playbook -i node_prep_inventory.yml node_prep.yml
```

第13章 障害からの復旧

本章では、ディスクまたはサーバーの障害後にクラスターを作業状態に復元する方法を説明します。

本章を使用するには、障害復旧オプションを以前に設定しておく必要があります。詳細は、「[バックアップオプションおよびリカバリーオプションの設定](#)」を参照してください。

13.1. バックアップボリュームからのデータの手動による復元

本セクションでは、仮想化用に、リモートバックアップボリュームから Red Hat IaaS Infrastructure の置き換えデプロイメントにデータを復元する方法を説明します。

これを実行するには、以下を行う必要があります。

1. 『[Red Hat IaaS Infrastructure for Virtualization のデプロイ](#)』の説明に従って、代替のデプロイメントをインストールおよび設定します。

13.1.1. 地理的に複製されたバックアップからのボリュームの復元

1. 代替となるハイパーコンバージドインフラストラクチャーのデプロイメントのインストールおよび設定
手順は、「[Red Hat Virtualization 向けの Red Hat IaaS インフラストラクチャーのデプロイ](https://access.redhat.com/documentation/ja-jp/red_hat_hyperconverged_infrastructure_for_virtualization/1.7/html/deploying_red_hat_hy)」を参照してください。
2. バックアップボリュームで読み取り専用モードを無効にする
データが変更されないように、各同期後にジオ複製ボリュームが読み取り専用を設定されます。Red Hat Virtualization には、ボリュームをストレージドメインとしてインポートするために書き込み権限が必要です。

以下のコマンドを実行して、バックアップボリュームで読み取り専用モードを無効にします。

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

3. ストレージドメインのバックアップのインポート
管理ポータルで、新しい Infrastructure のデプロイメントから以下を行います。
 - a. Storage から Domains をクリックします。
 - b. Import Domain をクリックします。「ドメインのインポート」ウィンドウが開きます。
 - c. Storage Type フィールドで GlusterFS を指定します。
 - d. 名前 フィールドには、バックアップボリュームから作成される新規ボリュームの名前を指定します。
 - e. パス フィールドには、バックアップボリュームへのパスを指定します。
 - f. OK をクリックします。以下の警告が表示され、以下のアクティブなデータセンターが表示されます。

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. Approve operation チェックボックスをチェックして、OK をクリックします。

4. インポートする仮想マシンの一覧の決定

- a. 以下のコマンドを実行して、インポートされたドメインの識別子を確認します。

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

以下に例を示します。

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

- b. 以下のコマンドを実行して、登録解除したディスクの一覧を確認します。

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

以下に例を示します。

```
# 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. 各仮想マシンのストレージドメインへの部分的なインポートを実行する。

- a. クラスタ識別子の特定
以下のコマンドは、クラスタ ID を返します。

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

以下に例を示します。

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

- b. 仮想マシンのインポート
以下のコマンドは、すべてのディスクをストレージドメインに使用せずに仮想マシンをインポートします。

```
# 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"
```

以下に例を示します。

```
# 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"
```

詳細は、『Red Hat Virtualization REST API ガイド』の

「https://access.redhat.com/documentation/ja-jp/red_hat_virtualization/4.3/html/rest_api_guide/」を参照してください。

- 部分的にインポートされたディスクを新しいストレージドメインに移行します。管理ポータルで Storage → Disks をクリックし、Move Disk オプションをクリックします。インポートされたディスクを同期ボリュームから置き換えクラスターのストレージドメインに移動します。詳細は『Red Hat Virtualization [Administration Guide](#)』を参照してください。
- 復元されたディスクを新しい仮想マシンに割り当てる
『Red Hat Virtualization [Virtual Machine Management Guide](#)』の手順に従って、各仮想マシンに交換ディスクを割り当てます。

13.2. セカンダリークラスターへのフェイルオーバー

本セクションでは、サーバーに障害が発生した場合に、プライマリークラスターからリモートセカンダリークラスターにフェイルオーバーする方法を説明します。

- [リモートクラスターへのフェイルオーバーを設定](#) します。
- ソースおよびターゲットクラスターのマッピングファイルが正確であることを確認します。
- バックアップボリュームで読み取り専用モードを無効にする
データが変更されないように、各同期後にジオ複製ボリュームが読み取り専用を設定されます。Red Hat Virtualization には、ボリュームをストレージドメインとしてインポートするために書き込み権限が必要です。

以下のコマンドを実行して、バックアップボリュームで読み取り専用モードを無効にします。

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

- [fail_over](#) タグを指定してフェイルオーバー Playbook を実行します。

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

13.3. プライマリークラスターへのフェイルバック

本セクションでは、サーバー障害の原因を修正した後に、セカンダリークラスターからプライマリークラスターにフェイルバックする方法を説明します。

- [clean_engine](#) タグを指定してクリーンアップ Playbook を実行して、フェイルバックに向けてプライマリークラスターを準備します。

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

- ソースおよびターゲットクラスターのマッピングファイルが正確であることを確認します。

3. failback を実行して fail_back タグを付け、フェイルバック Playbook を実行します。

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

13.4. RHV MANAGER を使用した GEO レプリケーションセッションの停止

Geo レプリケーションを介して、データをアクティブなソースボリュームからパッシブターゲットボリュームに複製されないようにする場合は、geo レプリケーションセッションを停止します。

1. データが現在同期されていないことを確認する。
Manager の右上にある Tasks アイコンをクリックして、Tasks ページを確認します。

データ同期に関連する継続的なタスクがないことを確認します。

データ同期タスクが存在する場合は、それらが完了するまで待ちます。

2. geo レプリケーションセッションを停止する
 - a. Storage → Volumes をクリックします。
 - b. geo レプリケーションを防ぐボリュームの名前をクリックします。
 - c. Geo-replication サブタブをクリックします。
 - d. 停止するセッションを選択し、Stop をクリックします。

13.5. GEO レプリケーションスケジュールを削除してスケジュールされたバックアップをオフにする

geo レプリケーションのスケジュールを削除して、geo レプリケーションを使用してスケジュールされたバックアップを停止できます。

1. 任意のソースノードで管理ポータルにログインします。
2. Storage から Domains をクリックします。
3. バックアップするストレージドメインの名前をクリックします。
4. Remote Data Sync Setup サブタブをクリックします。
5. Setup をクリックします。
Setup Remote Data Synchronization ウィンドウが開きます。
6. Recurrence フィールドで、NONE の繰り返し間隔のタイプを選択し、OK をクリックします。
7. (オプション) geo レプリケーションセッションを削除します。
geo レプリケーションマスターノードから以下のコマンドを実行します。

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

reset-sync-time パラメーターを使用してこのコマンドを実行することもできます。このパラメーターの詳細と geo レプリケーションセッションの削除は、『Red Hat Gluster Storage 3.5 管理ガイド』の「[Geo-replication Session の削除](#)」を参照してください。

パート III. トラブルシューティング

第14章 自己修復が完了しない

自己修復操作が完了しない場合、原因は Gluster File ID(GFID)の不一致になる可能性があります。

14.1. GLUSTER ファイル ID の不一致

診断

1. 自己修復状態を確認します。
以下のコマンドを数分にわたり実行します。表示されるエントリーに注目してください。

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

同じエントリーが毎回表示されると、これらのエントリーに GFID が一致しなくなる可能性があります。

2. 各ホストの各エントリーの GFID を確認します。
各ホストで、各エントリーに対して以下のコマンドを実行します。

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

エントリーの <backend_path> は、ブリックパスとエントリーで構成されます。たとえば、エンジンボリュームのブリックに /gluster_bricks/engine/engine のパスがあり、heal info に示されるエントリーが 58d392a6-e5b1-4aed-9bbc-952210a7137d/ha_agent/hosted-engine.metadata の場合は、backend_ です。使用するパスは /gluster_bricks/engine/engine/58d392a6-e5b1-4aed-9bbc-952210a7137d/ha_agent/hosted-engine.metadata です。

3. 各ホストの出力を比較します。
エントリーの trusted.gfid がすべてのホストで同じでない場合は、GFID 不一致があります。

解決方法

1. GFID が最新の変更時間に優先され、不一致を解決します。

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

以下に例を示します。

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

2. ボリュームで修復を手動でトリガーします。

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

パート IV. 参考資料

付録A RED HAT GLUSTER STORAGE のフェンシングポリシー

Virtualization 用 RHHI (仮想化用の RHHI) デプロイメントには、以下のフェンシングポリシーが必要です。ブリックプロセスがまだ実行中である場合や、ホストをシャットダウンすると、クォーラムに到達できるボリューム機能が削除されます。

このポリシーは、Red Hat Gluster Storage 機能が有効な場合は、管理ポータルでの New Cluster または Edit Cluster ウィンドウで設定できます。

gluster ブリックが起動している場合は、フェンシングをスキップ

ブリックが実行され、他のピアから到達できる場合にはフェンシングは省略されます。

Gluster クォーラムが満たされない場合は、フェンシングをスキップします。

ブリックが実行中で、ホストがシャットダウンすると、クォーラムが失われるとフェンシングが省略されます。

このポリシーは、ノードがフェンスされるかどうかを判断する際に、その他すべてのフェンシングポリシー後に確認されます。

追加のフェンシングポリシーは、デプロイメントに役に立ちます。フェンシングに関する詳しい情報は、『Red Hat Virtualization Technical Reference』を参照してください。

付録B 用語集

B.1. 仮想化用語

Administration Portal

oVirt engine Web ユーザーインターフェースをベースとする Red Hat Virtualization Manager が提供する Web ユーザーインターフェース。これにより、管理者はネットワーク、ストレージドメイン、仮想マシンテンプレートなどのクラスターリソースを管理および監視できます。

ホスト型エンジン

RHHI for Virtualization を管理する Red Hat Virtualization Manager のインスタンス。

ホスト型エンジン仮想マシン

Red Hat Virtualization Manager として機能する仮想マシン。ホスト型 Engine 仮想マシンは、ホスト型 Engine 仮想マシンで実行されている Red Hat Virtualization Manager のインスタンスが管理する仮想化ホストで実行されます。

Manager ノード

ホスト型エンジンの仮想マシンで実行するのではなく、Red Hat Virtualization Manager を直接実行する仮想化ホスト。

Red Hat Enterprise Linux ホスト

Red Hat Enterprise Linux でインストールした物理マシンと、Red Hat Virtualization ホストと同じ機能を提供する追加パッケージ。このタイプのホストは、RHHI for Virtualization での使用はサポートされません。

Red Hat Virtualization

Linux および Microsoft Windows ワークロードのリソース、プロセス、およびアプリケーションを仮想化するためのオペレーティングシステムと管理インターフェース。

Red Hat Virtualization ホスト

Red Hat Virtualization でインストールされた物理マシン。Linux および Microsoft Windows ワークロードのリソース、プロセス、およびアプリケーションの仮想化をサポートする物理リソースを提供します。これは、RHHI for Virtualization でサポートされる唯一のタイプのホストです。

Red Hat Virtualization Manager

Red Hat Virtualization の管理および監視機能を実行するサーバー

セルフホストエンジンのノード

ホスト型エンジン仮想マシンを含む仮想化ホスト。RHHI for Virtualization デプロイメントのすべてのホストはセルフホストエンジンノードを持つことができますが、一度にセルフホストエンジンノードが1つしかありません。

ストレージドメイン

イメージ、テンプレート、スナップショット、およびメタデータの名前付きコレクション。ストレージドメインは、ブロックデバイスまたはファイルシステムから構成できます。ストレージドメインは、データセンターにアタッチされ、データセンター内のホストにはイメージ、テンプレートなどのコレクションにアクセスできます。

virtualization host

クライアントアクセスのために物理リソース、プロセス、およびアプリケーションを仮想化する機能がある物理マシン。

VM ポータル

Red Hat Virtualization Manager が提供する Web ユーザーインターフェースこれにより、ユーザーは仮想マシンの管理および監視を行うことができます。

B.2. ストレージ用語

ブリック

信頼できるストレージプール内のサーバーにエクスポートされるディレクトリー。

キャッシュ論理ボリューム

大規模で低速な論理ボリュームのパフォーマンスを改善するのに使用される小規模で高速な論理ボリューム。

geo-replication

ソースの Gluster ボリュームからターゲットボリュームへのデータの非同期レプリケーション。Geo レプリケーションは、ローカル領域ネットワークおよびインターネット全体で機能します。ターゲットボリュームは、異なる信頼できるストレージプールの Gluster ボリューム、または別のタイプのストレージになります。

Gluster ボリューム

ワークロードの要件に応じてデータを分散、複製、または分散するように設定できるブリックの論理グループ。

論理ボリューム管理(LVM)

物理ディスクを大きな仮想パーティションに統合する方法。物理ボリュームはボリュームグループに配置され、必要に応じて論理ボリュームに分割できるストレージのプールを形成します。

Red Hat Gluster Storage

Red Hat Enterprise Linux をベースとするオペレーティングシステムは、分散型のソフトウェア定義のストレージをサポートする追加パッケージが含まれています。

ソースボリューム

データが geo レプリケーション時にコピーされる Gluster ボリューム。

ストレージホスト

クライアントアクセス用にストレージを提供する物理マシン。

ターゲットボリューム

geo レプリケーション時にデータがコピーされる Gluster ボリュームまたは他のストレージボリューム。

シンプロビジョニング

必要な領域のみが作成時に割り当てられるように、プロビジョニングストレージは、時間の経過に応じてさらに領域を動的に割り当てられるようにします。

シックプロビジョニング

すぐに領域が必要かどうかに関係なく、すべての領域が作成時に割り当てられるようにプロビジョニングストレージをプロビジョニングします。

信頼できるストレージプール

信頼されるピアとして相互に認識する Red Hat Gluster Storage サーバーのグループ。

B.3. ハイパーコンバージドインフラストラクチャーの用語

Red Hat Hyperconverged Infrastructure(RHHI)for Virtualization

RHHI for Virtualization は、仮想コンピューティングおよび仮想ストレージリソースの両方を提供する単一の製品です。Red Hat Virtualization および Red Hat Gluster Storage は、接続設定にインストールされます。ここでは、両方の製品のサービスがクラスター内の各物理マシンで利用できます。

ハイパーコンバージドホスト

仮想化プロセスとアプリケーションが同じホスト上で実行する物理ストレージを提供する物理マシン。RHHI for Virtualization と共にインストールされるすべてのホストは、ハイパーコンバージドホストです。

Web コンソール

RHHI for Virtualization のデプロイ、管理、監視用の Web ユーザーインターフェース。Web コンソールは、Red Hat Virtualization Manager の Web コンソールサービスおよびプラグインによって提供されます。