Red Hat Enterprise Linux 7

パブリッククラウドプラットフォームへの Red Hat Enterprise Linux 7 のデプロイメント

Red Hat Enterprise Linux のカスタムイメージの作成およびパブリッククラウドプラットフォーム用の Red Hat High Availability クラスターの設定

Last Updated: 2023-04-10

Red Hat Enterprise Linux 7 パブリッククラウドプラットフォームへの Red Hat Enterprise Linux 7 のデプロイメント

Red Hat Enterprise Linux のカスタムイメージの作成およびパブリッククラウドプラットフォーム用の Red Hat High Availability クラスターの設定

法律上の通知

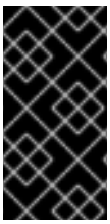
概要

カスタムの Red Hat Enterprise Linux イメージを作成して、Microsoft Azure、Amazon Web Services (AWS)、Google Cloud Platform (GCP) などのさまざまなクラウドプラットフォームにデプロイできます。 各クラウドプラットフォームに Red Hat High Availability クラスターを作成し、設定することもできます。本ガイドでは、イメージを作成する 2 つの選択肢 (Cloud Access イメージおよびオンデマンド (ベンチマーク) イメージ) を説明します。これには、必要なパッケージおよびエージェントのインストール、フェンシングの設定、およびネットワークリソースエージェントのインストールなど、HA クラスターを作成する手順が含まれます。 各クラウドプロバイダーには、カスタムイメージの作成およびデプロイを記述する独自の章があります。また、各クラウドプロバイダーの HA クラスターを設定する章も別途用意されています。

目次

第1章 MICROSOFT AZURE での仮想マシンとして RED HAT ENTERPRISE LINUX 7 イメージをデプロイ

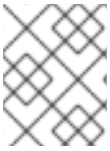
Azure に Red Hat Enterprise Linux (RHEL) 7 イメージをデプロイするオプションは複数あります。本章では、イメージを選択するオプションを説明し、ホストシステムおよび仮想マシンのシステム要件の一覧を紹介します。本章では、ISO イメージからカスタム VM を作成し、そのイメージを Azure にアップロードして、Azure 仮想インスタンスを起動する手順も説明します。



重要

ISO イメージからカスタム仮想マシンを作成することは可能ですが、Red Hat Image Builder 製品を使用して、特定のクラウドプロバイダーで使用するようにカスタマイズされたイメージを作成することを推奨します。詳細は Image Builder ガイド を参照してください。

この章では、Azure のドキュメントを参照している箇所が多数あります。多くの手順に関する詳細は、参照している Azure ドキュメントを確認してください。



注記

Azure でセキュアに使用できる Red Hat 製品の一覧は、Red Hat on Microsoft Azure を参照してください。

前提条件

- Red Hat カスタマーポータル のアカウントに登録している。
- Microsoft Azure アカウントに登録している。
- Red Hat Cloud Access プログラムでサブスクリプションが有効になっている。Red Hat Cloud Access プログラムでは、Red Hat サブスクリプションを、物理システムまたはオンプレミスシステムから、Red Hat のフルサポートのある Azure へ移動できます。

関連情報

- Red Hat in the Public Cloud
- Red Hat Cloud Access Reference Guide
- Frequently Asked Questions and Recommended Practices for Microsoft Azure

1.1. AZURE での RED HAT ENTERPRISE LINUX イメージオプション

以下の表には、イメージの選択肢を記載し、イメージオプションの相違点を示しています。

表1.1 イメージオプション

イメージオプション	サブスクリプション	サンプルシナリオ	留意事項

イメージオプション	サブスクリプション	サンプルシナリオ	留意事項
Red Hat Gold Image のデプロイを選択する	既存の Red Hat サブスクリプションを活用する	Red Hat Cloud Access プログラム を使用してサブスクリプションを有効にし、Azure で Red Hat Gold Image を選択します。Gold イメージおよび Azure でイメージにアクセスする方法は、Red Hat Cloud Access Reference Guide を参照してください。	このサブスクリプションには、Red Hat のコストが含まれていますが、他のインスタンスのコストは、Microsoft 社に支払うことになります。 Red Hat Gold Image は、既存の Red Hat サブスクリプションを活用するため、クラウドアクセスイメージと呼ばれています。Red Hat は、クラウドアクセスイメージを直接サポートします。
Azure に移動するカスタムイメージをデプロイすることを選択する	既存の Red Hat サブスクリプションを活用する	Red Hat Cloud Access プログラム を使用してサブスクリプションを有効にし、カスタムイメージをアップロードし、サブスクリプションを割り当てます。	このサブスクリプションには、Red Hat のコストが含まれていますが、他のインスタンスのコストは、Microsoft 社に支払うことになります。 Azure に移行するカスタムイメージは、既存の Red Hat サブスクリプションを活用するため、クラウドアクセスイメージと呼ばれています。Red Hat は、クラウドアクセスイメージを直接サポートします。
RHEL を含む既存の Azure イメージをデプロイすることを選択する	Azure イメージには、Red Hat 製品が含まれる	Azure コンソールを使用して仮想マシンを作成する際に RHEL イメージを選択するか、Azure Marketplace から仮想マシンを選択します。	従量課金モデルで 1 時間ごとに Microsoft 社に支払います。このようなイメージはオンデマンドと呼ばれています。 Azure は、サポート契約に基づいてオンデマンドイメージのサポートを提供します。 Red Hat は、イメージの更新を提供します。Azure により、Red Hat Update Infrastructure (RHUI) から更新を利用できるようにします。

注記

Red Hat Image Builder を使用して、Azure 用のカスタムイメージを作成できます。詳細は Image Builder ガイド を参照してください。

本章の残りの部分には、Red Hat Enterprise Linux カスタムイメージに関連する情報および手順が記載されています。

関連情報

- Red Hat Gold Images on Microsoft Azure
- Red Hat Cloud Access プログラム
- Azure Marketplace
- Billing options in the Azure Marketplace
- Red Hat Enterprise Linux Bring-Your-Own-Subscription Gold Images in Azure

1.2. ベースイメージの理解

本セクションでは、事前設定されたベースイメージおよびその設定を使用する方法を説明します。

1.2.1. カスタムベースイメージの使用

仮想マシンを手動で設定するには、ベース (スタートアップ) 仮想マシンイメージで開始します。ベース仮想マシンイメージを作成したら、設定設定を変更して、仮想マシンがクラウドで動作するために必要なパッケージを追加できます。イメージのアップロード後に、特定のアプリケーションに追加の設定変更を行うことができます。

RHEL の Hyper-V クラウドイメージを準備するには、Hyper-V Manager から RHEL 7 仮想マシンの準備 を参照してください。

関連情報

Red Hat Enterprise Linux

1.2.2. 必要なシステムパッケージ

本章の手順では、Red Hat Enterprise Linux を実行しているホストシステムを使用していることを前提としています。この手順を正常に行うには、ホストシステムに以下のパッケージをインストールする必要があります。

表1.2 システムパッケージ

パッケージ	詳細	コマンド
qemu-kvm	このパッケージは、ユーザーレベルの KVM エミュレーターを使用し、ホストとゲスト仮想マシン間の通信を容易にします。	**# yum install qemu-kvm libvirt**

パッケージ	詳細	コマンド
qemu-img	このパッケージは、ゲスト仮想マシンのディスク管理を提供します。qemu-img パッケージは qemu-kvm パッケージの依存関係としてインストールされます。	
libvirt	このパッケージは、ハイパーバイザーおよびホストシステムと対話するためにサーバーおよびホスト側のライブラリーを提供し、ライブラリー呼び出しの処理、仮想マシンの管理およびハイパーバイザーの制御を行う libvirtd デーモンも提供します。	

表1.3 その他の仮想化パッケージ

パッケージ	詳細	コマンド
virt-install	このパッケージは、コマンドラインからの仮想マシンを作成するために **virt-install** コマンドを提供します。	**# yum install virt-install libvirt-python virt-manager virt-install libvirt-client**
libvirt-python	このパッケージは、Python プログラミング言語で書かれているアプリケーションが libvirt API で提供されるインターフェイスを使用できるようにするモジュールが含まれています。	
virt-manager	このパッケージには、仮想マシンマネージャー (VMM) と呼ばれる virt-manager ツールが含まれます。VMM は、仮想マシンを管理するためのグラフィカルツールです。管理 API として libvirt-client ライブラリーを使用します。	

パッケージ	詳細	コマンド
libvirt-client	このパッケージは、libvirt サーバーにアクセスするためのクライアント側の API とライブラリーを提供します。libvirt-client パッケージには、コマンドラインまたは特別な仮想化シェルから仮想マシンとハイパーバイザーを管理および制御する **virsh** コマンドラインツールが含まれます。	

関連情報

- 仮想化パッケージの手動インストール

1.2.3. Azure VM 設定

Azure 仮想マシンには、以下の設定設定が必要です。設定の一部は、最初の仮想マシン作成時に有効になります。Azure 用の仮想マシンイメージのプロビジョニング時に、その他の設定が設定されます。エラーが発生する場合に、手順を進める時に、必要に応じてこの手順を再度参照してください。

表1.4 仮想マシンの設定設定

設定	推奨事項
ssh	Azure VM へのリモートアクセスを確立するには、**ssh** を有効にする必要があります。
dhcp	プライマリー仮想アダプターは、dhcp (IPv4 のみ) 用に設定する必要があります。
swap 領域	専用 swap ファイルまたは swap パーティションは作成しないでください。Windows Azure Linux Agent (WALinuxAgent) を使用してスワップ領域を設定できます。
NIC	プライマリー仮想ネットワークアダプター用に **virtio** を選択します。
暗号化	カスタムイメージで RHEL 7.5 以降を実行している場合には、Azure で完全なディスク暗号化に Network Bound Disk Encryption (NBDE) を使用します。NBDE は、RHEL 7.5 以降でのみサポートされます。

1.2.4. ISO イメージからのベースイメージの作成

以下の手順は、カスタム ISO イメージ作成の手順と初期設定の要件を示しています。イメージを設定したら、このイメージを、追加の仮想マシンインスタンスを作成するためのテンプレートとして使用できます。

手順

1. Red Hat カスタマーポータル から最新の Red Hat Enterprise Linux 7 Binary DVD ISO イメージをダウンロードします。
2. 仮想化用のホストマシンを有効にしていることを確認します。詳細および手順は、仮想化スタートガイド を参照してください。
3. 基本的な Red Hat Enterprise Linux 仮想マシンを作成し、起動します。手順は、仮想化コマンドラインインターフェイスの使用 を参照してください。
   a. コマンドラインを使用して仮想マシンを作成する場合は、デフォルトのメモリーと CPU を仮想マシンの容量に設定するようにしてください。仮想ネットワークインターフェイスを **virtio** に設定します。
   基本的なコマンドラインの例を以下に示します。

   ```
   virt-install --name isotest --memory 2048 --vcpus 2 --disk size=8,bus=virtio --location rhel-7.0-x86_64-dvd.iso --os-variant=rhel7.0
   ```

   b. VMM アプリケーションを使用して仮想マシンを作成する場合は、仮想マシンマネージャーの使用 の手順を行います。以下の点に注意してください。
      - **仮想マシンをすぐに起動** は選択しないでください。
      - **メモリー** と **ストレージのサイズ** を、希望の設定に変更します。
      - インストールを開始する前に、**仮想ネットワークインターフェイス設定** で **モデル** を **virtio** に変更し、**vCPUs** を仮想マシンの容量設定に変更していることを確認します。
4. 以下の追加インストールの選択と変更を確認します。
   - **標準 RHEL** オプションを使用して **最小インストール** を選択します。
   - **インストール先** で、**カスタムストレージ設定** を選択します。以下の設定情報を使用して選択を行います。
     - **/boot** で、500 MB 以上であることを確認してください。
     - ファイルシステムの場合は、**boot** パーティションおよび **root** パーティションの両方に xfs、ext4、ext3 のいずれかを使用します。
     - swap 領域を削除します。swap 領域は、WALinuxAgent により Azure 内の物理ブレードサーバー上で設定されます。
   - **インストール概要** 画面で、**ネットワークおよびホスト名** を選択します。**イーサネット** を **オン** に切り替えます。
5. インストール開始時に、以下を行います。
   - **root** のパスワードを作成します。
   - 管理者ユーザーアカウントを作成します。

6. インストールが完了したら、仮想マシンを再起動して root アカウントにログインします。

7. **root** でログインしたら、イメージを設定できます。

1.3. MICROSOFT AZURE のベースイメージの設定

ベースイメージには、Azure で RHEL 7 仮想マシンイメージとして機能するための設定変更が必要です。以下のセクションでは、Azure で必要な追加の設定変更を説明します。

1.3.1. Hyper-V デバイスドライバーのインストール

Microsoft は、Linux Integration Services (LIS) for Hyper-V パッケージの一部として、ネットワークおよびストレージデバイスのドライバーを提供しています。Hyper-V デバイスドライバーを Azure 仮想マシンとしてプロビジョニングする前に、仮想マシンイメージへのインストールが必要になる場合があります。**lsinitrd | grep hv** コマンドを使用して、ドライバーがインストールされていることを確認します。

手順

1. 以下の **grep** コマンドを実行して、必要な Hyper-V デバイスドライバーがインストールされているかどうかを確認します。

```
# lsinitrd | grep hv
```

以下の例では、必要なドライバーがすべてインストールされています。

```
# lsinitrd | grep hv
drwxr-xr-x   2 root     root            0 Aug 12 14:21 usr/lib/modules/3.10.0-
932.el7.x86_64/kernel/drivers/hv
-rw-r--r--   1 root     root        31272 Aug 11 08:45 usr/lib/modules/3.10.0-
932.el7.x86_64/kernel/drivers/hv/hv_vmbus.ko.xz
-rw-r--r--   1 root     root        25132 Aug 11 08:46 usr/lib/modules/3.10.0-
932.el7.x86_64/kernel/drivers/net/hyperv/hv_netvsc.ko.xz
-rw-r--r--   1 root     root         9796 Aug 11 08:45 usr/lib/modules/3.10.0-
932.el7.x86_64/kernel/drivers/scsi/hv_storvsc.ko.xz
```

すべてのドライバーがインストールされていない場合は、残りの手順を完了してください。

注記

hv_vmbus ドライバーは、すでにこの環境に追加されている可能性があります。このドライバーが存在する場合でも、仮想マシンで次の手順を実行してください。

2. **/etc/hv.conf.d** に **hv.conf** という名前のファイルを作成します。

3. 以下のドライバーパラメーターを **dracut.conf** ファイルに追加します。

```
add_drivers+=" hv_vmbus "
add_drivers+=" hv_netvsc "
add_drivers+=" hv_storvsc "
```

注記

引用符の前後に空白に注意してください (例: **add_drivers+=" hv_VMBus "**)。これにより、環境内にその他の Hyper-V ドライバーが存在している場合に、一意のドライバーが読み込まれます。

4. **initramfs** イメージを再生成します。

```
# dracut -f -v --regenerate-all
```

検証手順

1. マシンを再起動します。
2. **lsinitrd | grep hv** コマンドを実行して、ドライバーがインストールされていることを確認します。

1.3.2. 追加の設定設定の変更

仮想マシンを Azure で動作するには、さらなる設定変更が必要です。追加の変更を行うには、以下の手順を行います。

手順

1. 必要な場合は、仮想マシンの電源を入れます。
2. 仮想マシンを登録し、Red Hat Enterprise Linux 7 リポジトリーを有効にします。

```
# subscription-manager register --auto-attach
```

cloud-init (存在する場合) の停止および削除

1. **cloud-init** サービスを停止します。

```
# systemctl stop cloud-init
```

2. **cloud-init** ソフトウェアを削除します。

```
# yum remove cloud-init
```

その他の仮想マシン変更の完了

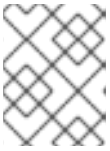
1. **/etc/ssh/sshd_config** ファイルを編集し、パスワード認証を有効にします。

```
PasswordAuthentication yes
```

2. 一般的なホスト名を設定してください。

```
# hostnamectl set-hostname localhost.localdomain
```

3. **/etc/sysconfig/network-scripts/ifcfg-eth0** ファイルを編集 (または作成) します。以下のパラメーターのみを使用してください。



注記

ifcfg-eth0 ファイルは、RHEL 7 DVD ISO イメージには存在しないため、作成する必要があります。

```
DEVICE="eth0"
ONBOOT="yes"
BOOTPROTO="dhcp"
TYPE="Ethernet"
USERCTL="yes"
PEERDNS="yes"
IPV6INIT="no"
```

4. すべての永続的なネットワークデバイスルールがある場合は削除します。

```
# rm -f /etc/udev/rules.d/70-persistent-net.rules
# rm -f /etc/udev/rules.d/75-persistent-net-generator.rules
# rm -f /etc/udev/rules.d/80-net-name-slot-rules
```

5. **ssh** が自動的に起動するように設定します。

```
# systemctl enable sshd
# systemctl is-enabled sshd
```

6. カーネルブートパラメーターを変更します。

   a. **/etc/default/grub** ファイルの **GRUB_CMDLINE_LINUX** 行の先頭に **crashkernel=256M** を追加します。**crashkernel=auto** が存在する場合は、**crashkernel=256M** に変更します。

   b. 次の行がない場合は、**GRUB_CMDLINE_LINUX** 行の末尾に追加します。

```
earlyprintk=ttyS0
console=ttyS0
rootdelay=300
```

   c. 以下のオプションが存在する場合は削除します。

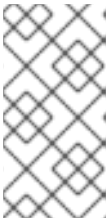
```
rhgb
quiet
```

7. **grub.cfg** ファイルを再生成します。

```
# grub2-mkconfig -o /boot/grub2/grub.cfg
```

8. Windows Azure Linux Agent (WALLinuxAgent) をインストールして有効にします。

```
# yum install WALinuxAgent -y
# systemctl enable waagent
```



注記

No package WALinuxAgent availableというエラーメッセージが表示された場合は、**rhel-7-server-extras-rpms** リポジトリーをインストールします。再度インストールを試行する前に、**# subscription-manager repos --enable=rhel-7-server-extras-rpms** コマンドを実行します。

9. **/etc/waagent.conf** で以下の行を編集して、プロビジョニングされた仮想マシン用にスワップ領域を設定します。プロビジョニングされた仮想マシンに適した swap 領域を設定します。

```
Provisioning.DeleteRootPassword=n
ResourceDisk.Filesystem=ext4
ResourceDisk.EnableSwap=y
ResourceDisk.SwapSizeMB=2048
```

プロビジョニングの準備

1. Red Hat Subscription Manager から仮想マシンの登録を解除します。

```
# subscription-manager unregister
```

2. 既存のプロビジョニングの詳細をクリーンアップして、Azure プロビジョニング用に仮想マシンを準備します。Azure は、仮想マシンを Azure に再プロビジョニングします。このコマンドは、データ損失の警告を生成しますが、これは正常です。

```
# waagent -force -deprovision
```

3. シェル履歴をクリーンアップし、仮想マシンをシャットダウンします。

```
# export HISTSIZE=0
# poweroff
```

1.4. イメージの固定 VHD 形式への変換

すべての Microsoft Azure 仮想マシンイメージは、固定 **VHD** 形式である必要があります。イメージは、VHD に変換する前に 1 MB の境界で調整する必要があります。このセクションでは、必要に応じて、イメージを、**qcow2** 形式から固定の **VHD** 形式に変換して配置する方法を説明します。イメージを変換したら、Azure にアップロードできます。

手順

1. イメージを **qcow2** 形式から **raw** 形式に変換します。

```
$ qemu-img convert -f qcow2 -O raw <image-name>.qcow2 <image-name>.raw
```

2. 以下のコンテンツを使用してシェルスクリプトを作成します。

```
#!/bin/bash
MB=$((1024 * 1024))
size=$(qemu-img info -f raw --output json "$1" | gawk 'match($0, /"virtual-size": ([0-9]+),/, val)
{print val[1]}')
rounded_size=$((($size/$MB + 1) * $MB))
```

```
if [ $(($size % $MB)) -eq  0 ]
then
 echo "Your image is already aligned. You do not need to resize."
 exit 1
fi
echo "rounded size = $rounded_size"
export rounded_size
```

3. スクリプトを実行します。この例では **align.sh** という名前を使用します。

```
$ sh align.sh <image-xxx>.raw
```

   - **Your image is already aligned. You do not need to resize. (イメージはすでに整列しています。サイズを変更する必要はありません。)**と表示されたら、次の手順に進みます。
   - 値が表示されると、イメージは調整されません。**イメージの調整** セクションの手順を使用してイメージのサイズを変更してから、次の手順に進むようにしてください。

4. 次のコマンドを使用して、ファイルを固定 **VHD** 形式に変換します。
サンプルでは qemu-img バージョン 2.12.0 を使用します。

```
$ qemu-img convert -f raw -o subformat=fixed,force_size -O vpc <image-xxx>.raw
<image.xxx>.vhd
```

   変換されると、**VHD** ファイルは Azure にアップロードする準備が整います。

イメージのアライメント

以下の手順は、**raw** ファイルの整列が行われていない場合にのみ行います。

1. 検証スクリプトの実行時に表示される丸め値を使用して、**raw** ファイルのサイズを変更します。

```
$ qemu-img resize -f raw <image-xxx>.raw <rounded-value>
```

2. **raw** イメージファイルを **VHD** 形式に変換します。
サンプルでは qemu-img バージョン 2.12.0 を使用します。

```
$ qemu-img convert -f raw -o subformat=fixed,force_size -O vpc <image-xxx>.raw
<image.xxx>.vhd
```

   変換されると、**VHD** ファイルは Azure にアップロードする準備が整います。

1.5. AZURE CLI のインストール

ホストマシンで Azure コマンドラインインターフェイス (Azure CLI 2.1) をインストールするには、以下の手順を実行します。Azure CLI 2.1 は、Azure で仮想マシンを作成し、管理する Python ベースのユーティリティーです。

前提条件

- Azure CLI を使用するための Microsoft Azure のアカウントがある。
- Azure CLI をインストールするために Python 3.x がインストールされている。

手順

1. Microsoft リポジトリーキーをインポートします。

```
$ sudo rpm --import https://packages.microsoft.com/keys/microsoft.asc
```

2. ローカル Azure CLI リポジトリーエントリーを作成します。

```
$ sudo sh -c 'echo -e "[azure-cli]\nname=Azure
CLI\nbaseurl=https://packages.microsoft.com/yumrepos/azure-
cli\nenabled=1\ngpgcheck=1\ngpgkey=https://packages.microsoft.com/keys/microsoft.asc" >
/etc/yum.repos.d/azure-cli.repo'
```

3. **yum** パッケージインデックスを更新します。

```
$ yum check-update
```

4. Python バージョンを確認 (**python --version**) し、必要に応じて Python 3.x をインストールします。

```
$ sudo yum install python3
```

5. Azure CLI をインストールします。

```
$ sudo yum install -y azure-cli
```

6. Azure CLI を実行します。

```
$ az
```

関連情報

- Azure CLI
- Azure CLI コマンドリファレンス

1.6. AZURE でのリソースの作成

VHD ファイルをアップロードして Azure イメージを作成する前に、以下の手順に従って、必要な Azure リソースを作成します。

手順

1. 次のコマンドを実行して、システムを Azure で認証し、ログインします。

```
$ az login
```

注記

お使いの環境でブラウザーが利用可能な場合、CLI は Azure サインインページでブラウザーを開きます。詳細およびオプションは、Sign in with Azure CLI を参照してください。

2. Azure リージョンにリソースグループを作成します。

```
$ az group create --name <resource-group> --location <azure-region>
```

以下に例を示します。

```
$ az group create --name azrhelclirsgrp --location southcentralus
{
  "id": "/subscriptions//resourceGroups/azrhelclirsgrp",
  "location": "southcentralus",
  "managedBy": null,
  "name": "azrhelclirsgrp",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null
}
```

3. ストレージアカウントを作成します。有効な SKU 値の詳細は、SKU Types を参照してください。

```
$ az storage account create -l <azure-region> -n <storage-account-name> -g <resource-
group> --sku <sku_type>
```

以下に例を示します。

```
$ az storage account create -l southcentralus -n azrhelclistact -g azrhelclirsgrp --sku
Standard_LRS
{
  "accessTier": null,
  "creationTime": "2017-04-05T19:10:29.855470+00:00",
  "customDomain": null,
  "encryption": null,
  "id":
"/subscriptions//resourceGroups/azrhelclirsgrp/providers/Microsoft.Storage/storageAccounts/azr
helclistact",
  "kind": "StorageV2",
  "lastGeoFailoverTime": null,
  "location": "southcentralus",
  "name": "azrhelclistact",
  "primaryEndpoints": {
    "blob": "https://azrhelclistact.blob.core.windows.net/",
    "file": "https://azrhelclistact.file.core.windows.net/",
    "queue": "https://azrhelclistact.queue.core.windows.net/",
    "table": "https://azrhelclistact.table.core.windows.net/"
},
"primaryLocation": "southcentralus",
"provisioningState": "Succeeded",
"resourceGroup": "azrhelclirsgrp",
"secondaryEndpoints": null,
"secondaryLocation": null,
"sku": {
  "name": "Standard_LRS",
  "tier": "Standard"
},
```

```
"statusOfPrimary": "available",
"statusOfSecondary": null,
"tags": {},
  "type": "Microsoft.Storage/storageAccounts"
}
```

4. ストレージアカウントの接続文字列を取得します。

```
$ az storage account show-connection-string -n <storage-account-name> -g <resource-
group>
```

以下に例を示します。

```
[clouduser@localhost]$ az storage account show-connection-string -n azrhelclistact -g
azrhelclirsgrp
{
  "connectionString":
"DefaultEndpointsProtocol=https;EndpointSuffix=core.windows.net;AccountName=azrhelclistact
AccountKey=NreGk...=="
}
```

5. 接続文字列をコピーし、次のコマンドに貼り付けて、接続文字列をエクスポートします。この文字列は、システムをストレージアカウントに接続します。

```
$ export AZURE_STORAGE_CONNECTION_STRING="<storage-connection-string>"
```

以下に例を示します。

```
[clouduser@localhost]$ export
AZURE_STORAGE_CONNECTION_STRING="DefaultEndpointsProtocol=https;EndpointSuffi
x=core.windows.net;AccountName=azrhelclistact;AccountKey=NreGk...=="
```

6. ストレージコンテナーを作成します。

```
$ az storage container create -n <container-name>
```

以下に例を示します。

```
[clouduser@localhost]$ az storage container create -n azrhelclistcont
{
  "created": true
}
```

7. 仮想ネットワークを作成します。

```
$ az network vnet create -g <resource group> --name <vnet-name> --subnet-name <subnet-
name>
```

以下に例を示します。

```
[clouduser@localhost]$ az network vnet create --resource-group azrhelclirsgrp --name
azrhelclivnet1 --subnet-name azrhelclisubnet1
{
```

```
  "newVNet": {
    "addressSpace": {
      "addressPrefixes": [
      "10.0.0.0/16"
      ]
  },
  "dhcpOptions": {
    "dnsServers": []
  },
  "etag": "W/\"\"",
  "id":
"/subscriptions//resourceGroups/azrhelclirsgrp/providers/Microsoft.Network/virtualNetworks/azr
helclivnet1",
  "location": "southcentralus",
  "name": "azrhelclivnet1",
  "provisioningState": "Succeeded",
  "resourceGroup": "azrhelclirsgrp",
  "resourceGuid": "0f25efee-e2a6-4abe-a4e9-817061ee1e79",
  "subnets": [
    {
      "addressPrefix": "10.0.0.0/24",
      "etag": "W/\"\"",
      "id":
"/subscriptions//resourceGroups/azrhelclirsgrp/providers/Microsoft.Network/virtualNetworks/azr
helclivnet1/subnets/azrhelclisubnet1",
      "ipConfigurations": null,
      "name": "azrhelclisubnet1",
      "networkSecurityGroup": null,
      "provisioningState": "Succeeded",
      "resourceGroup": "azrhelclirsgrp",
      "resourceNavigationLinks": null,
      "routeTable": null
    }
  ],
  "tags": {},
  "type": "Microsoft.Network/virtualNetworks",
  "virtualNetworkPeerings": null
  }
}
```


関連情報

- Azure geographies
- Sign in with Azure CLI
- Azure Managed Disks Overview
- SKU Types

1.7. AZURE イメージのアップロードおよび作成

以下の手順に従って、**VHD** ファイルをコンテナーにアップロードし、Azure カスタムイメージを作成します。

注記

システムを再起動すると、エクスポートしたストレージ接続文字列は維持されません。以下の手順でいずれかのコマンドが失敗した場合は、再び接続文字列をエクスポートしてください。

手順

1. ストレージコンテナーに **VHD** ファイルをアップロードします。ストレージコンテナーの一覧を表示するには、**az storage container list** を実行します。

```
$ az storage blob upload --account-name <storage-account-name> --container-name
<container-name> --type page --file <path-to-vhd> --name <image-name>.vhd
```

以下に例を示します。

```
$ az storage blob upload --account-name azrhelclistact --container-name azrhelclistcont --
type page --file rhel-image-7.vhd --name rhel-image-7.vhd
Percent complete: %100.0
```

2. アップロードした **VHD** ファイルの URL を以下の手順で取得します。

```
$ az storage blob url -c <container-name> -n <image-name>.vhd
```

以下に例を示します。

```
$ az storage blob url -c azrhelclistcont -n rhel-image-7.vhd
"https://azrhelclistact.blob.core.windows.net/azrhelclistcont/rhel-image-7.vhd"
```

3. Azure カスタムイメージを作成します。

```
$ az image create -n <image-name> -g <resource-group> -l <azure-region> --source <URL>
--os-type linux
```

注記

仮想マシンのハイパーバイザーのデフォルトの生成は V1 です。必要に応じて、**--hyper-v-generation V2** オプションを使用して V2 ハイパーバイザーの世代を指定できます。第 2 世代の仮想マシンは、UEFI ベースのブートアーキテクチャーを使用します。第 2 世代の仮想マシンの詳細は Support for generation 2 VMs on Azure を参照してください。

このコマンドは、**Only blobs formatted as VHDs can be imported**(VHD としてフォーマットされたブロブのみがインポート可能) というエラーを返す場合があります。 このエラーは、イメージが **VHD** に変換される前に最も近い 1 MB の境界に合致していないことを意味します。

以下に例を示します。

```
$ az image create -n rhel7 -g azrhelclirsgrp2 -l southcentralus --source
https://azrhelclistact.blob.core.windows.net/azrhelclistcont/rhel-image-7.vhd --os-type linux
```

1.8. AZURE での仮想マシンの作成および起動

以下の手順では、イメージから管理ディスクの Azure 仮想マシンを作成するための最小限のコマンドオプションを説明します。追加オプションは、az vm create を参照してください。

手順

1. 次のコマンドを実行して仮想マシンを作成します。

```
$ az vm create -g <resource-group> -l <azure-region> -n <vm-name> --vnet-name <vnet-name> --subnet <subnet-name> --size Standard_A2 --os-disk-name <simple-name> --admin-username <administrator-name> --generate-ssh-keys --image <path-to-image>
```

注記

--generate-ssh-keys オプションを指定すると、秘密鍵と公開鍵のペアが作成されます。秘密鍵ファイルと公開鍵ファイルが、システムの ~/**.ssh** に作成されます。公開鍵は、**--admin-username** オプションで指定したユーザーの仮想マシン上の **authorized_keys** ファイルに追加されます。詳細は、その他の認証方法 を参照してください。

たとえば、以下のようになります。

```
[clouduser@localhost]$ az vm create -g azrhelclirsgrp2 -l southcentralus -n rhel-azure-vm-1 --vnet-name azrhelclivnet1 --subnet azrhelclisubnet1  --size Standard_A2 --os-disk-name vm-1-osdisk --admin-username clouduser --generate-ssh-keys --image rhel7
{
  "fqdns": "",
  "id":
"/subscriptions//resourceGroups/azrhelclirsgrp/providers/Microsoft.Compute/virtualMachines/rhel-azure-vm-1",
  "location": "southcentralus",
  "macAddress": "",
  "powerState": "VM running",
  "privateIpAddress": "10.0.0.4",
  "publicIpAddress": "<public-IP-address>",
  "resourceGroup": "azrhelclirsgrp2"
```

2. SSH セッションを開始し、仮想マシンにログインします。

```
[clouduser@localhost]$ ssh  -i /home/clouduser/.ssh/id_rsa clouduser@<public-IP-address>.
The authenticity of host,  '<public-IP-address>' can't be established.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '<public-IP-address>' (ECDSA) to the list of known hosts.
```

ユーザープロンプトが表示された場合は、Azure VM が正常にデプロイされます。

これで、Azure ポータルにアクセスして、リソースの監査ログとプロパティーを確認できます。このポータルでは、仮想マシンを直接管理できます。複数の仮想マシンを管理している場合は、Azure CLI を使用する必要があります。Azure CLI では、Azure 内のリソースに強力なインターフェイスを利用できます。Microsoft Azure で仮想マシンを管理するために使用するコマンドの詳細は、CLI で **az --help** コマンドを実行するか、Azure CLI コマンドリファレンス を参照してください。

1.9. その他認証方法

セキュリティーを強化するために推奨されますが、Azure 生成のキーペアを使用する必要はありません。以下の例は、SSH 認証用の 2 つの方法を示しています。

例 1: 次のコマンドオプションは、公開鍵ファイルを生成せずに新しい仮想マシンをプロビジョニングします。これにより、SSH は、パスワードを使用した SSH 認証を許可できます。

```
$ az vm create -g <resource-group> -l <azure-region> -n <vm-name> --vnet-name <vnet-name> --subnet <subnet-name> --size Standard_A2 --os-disk-name <simple-name> --authentication-type password --admin-username <administrator-name> --admin-password <ssh-password> --image <path-to-image>
```

```
$ ssh <admin-username>@<public-ip-address>
```

例 2: このコマンドオプションにより、新しい Azure 仮想マシンがプロビジョニングされ、既存の公開鍵ファイルを使用した SSH 認証が許可されます。

```
$ az vm create -g <resource-group> -l <azure-region> -n <vm-name> --vnet-name <vnet-name> --subnet <subnet-name> --size Standard_A2 --os-disk-name <simple-name> --admin-username <administrator-name> --ssh-key-value <path-to-existing-ssh-key> --image <path-to-image>
```

```
$ ssh -i <path-to-existing-ssh-key> <admin-username>@<public-ip-address>
```

1.10. RED HAT サブスクリプションの割り当て

Red Hat Cloud Access プログラムで有効になっているサブスクリプションを割り当てるには、以下の手順を行います。

前提条件

サブスクリプションが有効になっている。

手順

1. システムを登録します。

```
subscription-manager register --auto-attach
```

2. サブスクリプションを割り当てます。

- アクティベーションキーを使用して、サブスクリプションを割り当てることができます。カスタマーポータルのアクティベーションキーを作成する を参照してください。
- または、サブスクリプションプール (Pool ID) の ID を使用してサブスクリプションを手動で割り当てることができます。コマンドラインでのサブスクリプションのアタッチと削除 を参照してください。

関連情報

- カスタマーポータルのアクティベーションキーを作成する
- コマンドラインでのサブスクリプションのアタッチと削除

- Red Hat Subscription Manager の使用および設定

第2章 MICROSOFT AZURE での RED HAT HIGH AVAILABILITY クラスターの設定

Red Hat は、Red Hat Enterprise Linux(RHEL)7.4 以降で高可用性 (HA) をサポートします。本章では、仮想マシンインスタンスをクラスターノードとして使用して Microsoft Azure に Red Hat HA クラスターを設定するための情報および手順を説明します。本章の手順では、Azure 用のカスタムイメージを作成していることを前提としています。クラスターに使用する RHEL 7 イメージを取得するオプションは複数あります。Azure のイメージオプションに関する詳細は、Azure における Red Hat Enterprise Linux Image オプション を参照してください。

本章では、Azure の環境設定の前提条件を説明します。環境を設定したら、Azure 仮想マシンインスタンスを作成および設定できます。

本章では、個別の仮想マシンノードを Azure の HA ノードのクラスターに変換する HA クラスターの作成に固有の手順も説明します。これには、各クラスターノードに高可用性パッケージおよびエージェントをインストールし、フェンシングを設定し、Azure ネットワークリソースエージェントをインストールする手順が含まれます。

本章では、Microsoft Azure のドキュメントを参照している箇所が多数あります。多くの手順に関する詳細は、参照している Azure ドキュメントを確認してください。

前提条件

- Azure コマンドラインインターフェイス (CLI) をインストールする必要があります。詳細は、Azure CLI のインストール を参照してください。
- Red Hat Cloud Access プログラム でサブスクリプションを有効にします。Red Hat Cloud Access プログラムでは、Red Hat サブスクリプションを、物理システムまたはオンプレミスシステムから、Red Hat のフルサポートのある Azure へ移動できます。

関連情報

- Support Policies for RHEL High Availability Clusters - Microsoft Azure Virtual Machines as Cluster Members
- High Availability アドオンの概要

2.1. AZURE でのリソースの作成

可用性セットを作成するには、以下の手順を実施します。本章の後続のタスクを完了するには、これらのリソースが必要です。

手順

- 可用性セットを作成します。すべてのクラスターノードは同じ可用性セットにある必要があります。

  ```
  $ az vm availability-set create --name _MyAvailabilitySet_ --resource-group
  _MyResourceGroup_
  ```

  以下に例を示します。

  ```
  [clouduser@localhost]$ az vm availability-set create --name rhelha-avset1 --resource-group
  azrhelclirsgrp
  ```

```
{
  "additionalProperties": {},
    "id":
"/subscriptions/.../resourceGroups/azrhelclirsgrp/providers/Microsoft.Compute/availabilitySets/rh
elha-avset1",
    "location": "southcentralus",
    "name": “rhelha-avset1",
    "platformFaultDomainCount": 2,
    "platformUpdateDomainCount": 5,

...omitted
```

関連情報

- Sign in with Azure CLI
- SKU Types
- Azure Managed Disks Overview

2.2. AZURE ACTIVE DIRECTORY アプリケーションの作成

Azure Active Directory アプリケーションを作成するには、以下の手順を行います。Azure AD アプリケーションは、クラスター内のすべてのノードに対する HA 操作のアクセスを承認し、自動化します。

前提条件

Azure コマンドラインインターフェイス (CLI) をインストールする必要があります。

手順

1. Microsoft Azure サブスクリプションの管理者または所有者であることを確認します。Azure AD アプリケーションを作成するには、この承認が必要です。
2. Azure アカウントにログインします。

```
$ az login
```

3. 次のコマンドを実行して Azure AD アプリケーションを作成します。独自のパスワードを使用するには、**--password** オプションをコマンドに追加します。強固なパスワードを作成してください。

```
$ az ad sp create-for-rbac --name _FencingApplicationName_ --role owner --scopes
"/subscriptions/_SubscriptionID_/resourceGroups/_MyResourseGroup_"
```

以下に例を示します。

```
[clouduser@localhost ~] $ az ad sp create-for-rbac --name FencingApp --role owner --
scopes "/subscriptions/2586c64b-xxxxxx-xxxxxxx-xxxxxxx/resourceGroups/azrhelclirsgrp"
Retrying role assignment creation: 1/36
Retrying role assignment creation: 2/36
Retrying role assignment creation: 3/36
{
  "appId": "1a3dfe06-df55-42ad-937b-326d1c211739",
```

```
  "displayName": "FencingApp",
  "name": "http://FencingApp",
  "password": "43a603f0-64bb-482e-800d-402efe5f3d47",
  "tenant": "77ecefb6-xxxxxxxxxx-xxxxxxx-757a69cb9485"
}
```

4. 続行する前に、以下の情報を保存します。フェンスエージェントを設定するには、この情報が必要です。
   - Azure AD アプリケーション ID
   - Azure AD アプリケーションのパスワード
   - テナント ID
   - Microsoft Azure サブスクリプション ID

関連情報

View the access a user has to Azure resources

2.3. RED HAT HA パッケージおよびエージェントのインストール

すべてのノードで以下の手順を実行します。

手順

1. 仮想マシンを Red Hat に登録します。

```
$ sudo -i
# subscription-manager register --auto-attach
```

2. すべてのリポジトリーを無効にします。

```
# subscription-manager repos --disable=*
```

3. RHEL 7 Server リポジトリーおよび RHEL 7 Server HA リポジトリーを有効にします。

```
# subscription-manager repos --enable=rhel-7-server-rpms
# subscription-manager repos --enable=rhel-ha-for-rhel-7-server-rpms
```

4. すべてのパッケージを更新します。

```
# yum update -y
```

5. カーネルを更新したら再起動します。

```
# reboot
```

6. **pcs**、**pacemaker**、**fence agent**、**resource agent** および **nmap-ncat** をインストールします。

```
# yum install -y pcs pacemaker fence-agents-azure-arm resource-agents nmap-ncat
```

2.4. HA サービスの設定

すべてのノードで以下の手順を実行します。

手順

1. ユーザー **hacluster** は、前のセクションの **pcs** および **pacemaker** のインストール時に作成されました。すべてのクラスターノードに **hacluster** のパスワードを作成します。すべてのノードに同じパスワードを使用します。

```
# passwd hacluster
```

2. **firewalld.service** が有効化されている場合は、RHEL ファイアウォールに **high availability** サービスを追加します。

```
# firewall-cmd --permanent --add-service=high-availability
# firewall-cmd --reload
```

3. **pcs** サービスを起動し、システムの起動時に開始できるようにします。

```
# systemctl enable pcsd.service --now
```

検証手順

- **pcs** サービスが実行していることを確認します。

```
# systemctl is-active pcsd.service
```

2.5. クラスターの作成

ノードのクラスターを作成するには、以下の手順を実施します。

手順

1. ノードのいずれかで以下のコマンドを実行し、pcs ユーザー **hacluster** を認証します。クラスターの各ノードの名前を指定します。

```
# pcs host auth  _hostname1_ _hostname2_ _hostname3_
```

たとえば、以下のようになります。

```
[root@node01 clouduser]# pcs host auth node01 node02 node03
Username: hacluster
Password:
node01: Authorized
node02: Authorized
node03: Authorized
```

2. クラスターを作成します。

```
# pcs cluster setup --name _hostname1_ _hostname2_ _hostname3_
```

たとえば、以下のようになります。

```
[root@node01 clouduser]# pcs cluster setup --name newcluster node01 node02 node03

...omitted

Synchronizing pcsd certificates on nodes node01, node02, node03...
node02: Success
node03: Success
node01: Success
Restarting pcsd on the nodes in order to reload the certificates...
node02: Success
node03: Success
node01: Success
```

検証手順

1. クラスターを有効にします。

```
# pcs cluster enable --all
```

2. クラスターを起動します。

```
# pcs cluster start --all
```

たとえば、以下のようになります。

```
[root@node01 clouduser]# pcs cluster enable --all
node02: Cluster Enabled
node03: Cluster Enabled
node01: Cluster Enabled

[root@node01 clouduser]# pcs cluster start --all
node02: Starting Cluster...
node03: Starting Cluster...
node01: Starting Cluster...
```

2.6. フェンスデバイスの作成

クラスター内の任意のノードからフェンシングを設定するには、以下の手順を実施します。

手順

1. 利用可能なインスタンスで、フェンスが可能なものを特定します。

```
# fence_azure_arm -l [appid] -p [authkey] --resourceGroup=[name] --subscriptionId=[name] --tenantId=[name] -o list
```

たとえば、以下のようになります。

```
[root@node1 ~]# fence_azure_arm -l XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX -p XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX --resourceGroup=hacluster-rg --subscriptionId=XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX --tenantId=XXXXXXXX-
```

```
XXXX-XXXX-XXXX-XXXXXXXXXXXX -o list
node01-vm,
node02-vm,
node03-vm,
```

2. フェンスデバイスを作成します。**pcmk_host_map** コマンドを使用して、RHEL ホスト名をインスタンス ID にマッピングします。

```
# pcs stonith create _clusterfence_ fence_azure_arm login=_AD-Application-ID_
passwd=_AD-passwd_ pcmk_host_map="_pcmk-host-map_ resourcegroup=
_myresourcegroup_ tenantid=_tenantid_ subscriptionid=_subscriptionid_
```

検証手順

1. 他のノードのいずれかに対してフェンスエージェントをテストします。

```
# pcs stonith fence _azurenodename_
```

たとえば、以下のようになります。

```
[root@node01 ~]# pcs stonith fence fenceazure
 Resource: fenceazure (class=stonith type=fence_azure_arm)
  Attributes: login=XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX passwd=XXXXXXXX-
XXXX-XXXX-XXXX-XXXXXXXXXXXX pcmk_host_map=nodea:nodea-vm;nodeb:nodeb-
vm;nodec:nodec-vm pcmk_reboot_retries=4 pcmk_reboot_timeout=480 power_timeout=240
resourceGroup=rg subscriptionId=XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
tenantId=XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
  Operations: monitor interval=60s (fenceazure-monitor-interval-60s)
[root@node01 ~]# pcs stonith
 fenceazure     (stonith:fence_azure_arm):      Started nodea
```

2. ステータスを確認して、ノードが起動したことを確認します。

```
# watch pcs status
```

以下に例を示します。

```
[root@node01 ~]# watch pcs status
 fenceazure     (stonith:fence_azure_arm):      Started nodea
```

関連情報

- Red Hat High Availability クラスターでのフェンシングの設定
- High Availability Add-On の管理

2.7. AZURE 内部ロードバランサーの作成

Azure 内部ロードバランサーは、ヘルスプローブ要求に応答しないクラスターノードを削除します。

以下の手順を実行し、Azure 内部ロードバランサーを作成します。各ステップは特定の Microsoft 手順を参照し、HA のロードバランサーをカスタマイズするための設定が含まれます。

前提条件

Azure コントロールパネル へのアクセス

手順

1. 基本ロードバランサーを作成。IP アドレスの割り当てタイプの場合は、**内部ロードバランサー**、**基本 SKU**、および **動的** を選択します。
2. バックエンドのアドレスプールを作成します。バックエンドプールを HA に Azure リソースを作成した時に作成された可用性セットに関連付けます。ターゲットネットワーク IP 設定は設定しないでください。
3. ヘルスプローブを作成します。ヘルスプローブの場合は **TCP** を選択し、ポート **61000** を入力します。別のサービスに干渉しない TCP ポート番号を使用できます。特定の HA 製品アプリケーション (SAP HANA や SQL Server など) については、Microsoft と連携して使用する正しいポートを指定する必要がある場合があります。
4. ロードバランサールールを作成します。ロードバランシングルールを作成する場合は、事前に設定されているデフォルト値を使用します。**フローティング IP (ダイレクトサーバーを返す)** を **有効** に設定してください。

2.8. ロードバランサーリソースエージェントの設定

ヘルスプローブを作成したら、**ロードバランサー** リソースエージェントを設定する必要があります。このリソースエージェントは、Azure ロードバランサーからヘルスプローブ要求に応答し、要求に応答しないクラスターノードを削除するサービスを実行します。

手順

1. **Azure id** コマンドを入力して、Azure ロードバランサーリソースエージェントの説明を表示します。これは、このエージェントのオプションとデフォルトの操作を示しています。

```
# pcs resource describe _azure-id_
```

2. ノード上の IP を管理する **Ipaddr2** リソースを作成します。

```
# pcs resource create _resource-id_ IPaddr2 ip=_virtual/floating-ip_
cidr_netmask=_virtual/floating-mask_ --group _group-id_ nic=_network-interface_ op monitor
interval=30s
```

たとえば、以下のようになります。

```
[root@node01 ~]# pcs resource create ClusterIP ocf:heartbeat:IPaddr2 ip=172.16.66.99
cidr_netmask=24 --group CloudIP nic=eth0 op monitor interval=30s
```

3. **ロードバランサー** リソースエージェントを設定します。

```
# pcs resource create _resource-loadbalancer-name_ azure-lb port=_port-number_ --group
_cluster-resources-group_
```

検証手順

- **pcs status** コマンドを実行して結果を表示します。

```
[root@node01 clouduser]# pcs status
```

たとえば、以下のようになります。

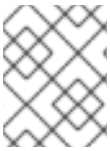
```
[root@node01 ~]# pcs status
Cluster name: hacluster

WARNINGS:
No stonith devices and stonith-enabled is not false

Stack: corosync
Current DC: nodeb (version 1.1.22-1.el7-63d2d79005) - partition with quorum
Last updated: Wed Sep  9 16:47:07 2020
Last change: Wed Sep  9 16:44:32 2020 by hacluster via crmd on nodeb

3 nodes configured
0 resource instances configured

Online: [ node01 node02 node03 ]

No resources


Daemon Status:
  corosync: active/enabled
  pacemaker: active/enabled
  pcsd: active/enabled
```

関連情報

- クラスター操作

2.9. 共有ブロックストレージの設定

このセクションでは、Microsoft Azure Shared Disks を使用して Red Hat High Availability クラスターに共有ブロックストレージを設定するオプションの手順を説明します。この手順では、1TB 共有ディスクを持つ 3 つの Azure 仮想マシン (3 ノードクラスター) を想定しています。



注記

これは、ブロックストレージを設定するスタンドアロンの例です。この手順では、クラスターを作成していないことを前提としています。

前提条件

- ホストシステムに Azure CLI をインストールし、SSH キーを作成している。
- 以下の作成を含むクラスター環境を Azure で作成している。リンクは、Microsoft Azure のドキュメントにあります。
  - リソースグループ
  - 仮想ネットワーク

- ネットワークセキュリティーグループ
- ネットワークセキュリティーグループルール
- サブネット
- ロードバランサー (オプション)
- ストレージアカウント
- 近接配置グループ
- 可用性セット

手順

1. Azure コマンド **az disk create** を使用して、共有ブロックボリュームを作成します。

```
$ az disk create -g resource_group -n shared_block_volume_name --size-gb disk_size --max-shares number_vms -l location
```

たとえば、以下のコマンドは、Azure Availability Zone **westcentralus** 内 のリソースグループ **sharedblock** に **shared-block-volume.vhd** という名前の共有ブロックボリュームを作成します。

```
$ az disk create -g sharedblock-rg -n shared-block-volume.vhd --size-gb 1024 --max-shares 3 -l westcentralus

{
  "creationData": {
    "createOption": "Empty",
    "galleryImageReference": null,
    "imageReference": null,
    "sourceResourceId": null,
    "sourceUniqueId": null,
    "sourceUri": null,
    "storageAccountId": null,
    "uploadSizeBytes": null
  },
  "diskAccessId": null,
  "diskIopsReadOnly": null,
  "diskIopsReadWrite": 5000,
  "diskMbpsReadOnly": null,
  "diskMbpsReadWrite": 200,
  "diskSizeBytes": 1099511627776,
  "diskSizeGb": 1024,
  "diskState": "Unattached",
  "encryption": {
    "diskEncryptionSetId": null,
    "type": "EncryptionAtRestWithPlatformKey"
  },
  "encryptionSettingsCollection": null,
  "hyperVgeneration": "V1",
  "id": "/subscriptions/12345678910-12345678910/resourceGroups/sharedblock-rg/providers/Microsoft.Compute/disks/shared-block-volume.vhd",
  "location": "westcentralus",
```

```
  "managedBy": null,
  "managedByExtended": null,
  "maxShares": 3,
  "name": "shared-block-volume.vhd",
  "networkAccessPolicy": "AllowAll",
  "osType": null,
  "provisioningState": "Succeeded",
  "resourceGroup": "sharedblock-rg",
  "shareInfo": null,
  "sku": {
    "name": "Premium_LRS",
    "tier": "Premium"
  },
  "tags": {},
  "timeCreated": "2020-08-27T15:36:56.263382+00:00",
  "type": "Microsoft.Compute/disks",
  "uniqueId": "cd8b0a25-6fbe-4779-9312-8d9cbb89b6f2",
  "zones": null
}
```


2. Azure コマンド **az disk show** を使用して共有ブロックボリュームを作成していることを確認します。

```
$ az disk show -g resource_group -n shared_block_volume_name
```

たとえば、次のコマンドは、リソースグループ **sharedblock-rg** 内の共有ブロックボリューム **shared-block-volume.vhd** の詳細を表示します。

```
$ az disk show -g sharedblock-rg -n shared-block-volume.vhd

{
  "creationData": {
    "createOption": "Empty",
    "galleryImageReference": null,
    "imageReference": null,
    "sourceResourceId": null,
    "sourceUniqueId": null,
    "sourceUri": null,
    "storageAccountId": null,
    "uploadSizeBytes": null
  },
  "diskAccessId": null,
  "diskIopsReadOnly": null,
  "diskIopsReadWrite": 5000,
  "diskMbpsReadOnly": null,
  "diskMbpsReadWrite": 200,
  "diskSizeBytes": 1099511627776,
  "diskSizeGb": 1024,
  "diskState": "Unattached",
  "encryption": {
    "diskEncryptionSetId": null,
    "type": "EncryptionAtRestWithPlatformKey"
  },
  "encryptionSettingsCollection": null,
  "hyperVgeneration": "V1",
```

```
  "id": "/subscriptions/12345678910-12345678910/resourceGroups/sharedblock-
rg/providers/Microsoft.Compute/disks/shared-block-volume.vhd",
  "location": "westcentralus",
  "managedBy": null,
  "managedByExtended": null,
  "maxShares": 3,
  "name": "shared-block-volume.vhd",
  "networkAccessPolicy": "AllowAll",
  "osType": null,
  "provisioningState": "Succeeded",
  "resourceGroup": "sharedblock-rg",
  "shareInfo": null,
  "sku": {
    "name": "Premium_LRS",
    "tier": "Premium"
  },
  "tags": {},
  "timeCreated": "2020-08-27T15:36:56.263382+00:00",
  "type": "Microsoft.Compute/disks",
  "uniqueId": "cd8b0a25-6fbe-4779-9312-8d9cbb89b6f2",
  "zones": null
}
```


3. Azure コマンド **az network nic create** を使用して、3 つのネットワークインターフェイスを作成します。それぞれ異なる **<nic_name>** を使用して、以下のコマンドを実行します。

```
$ az network nic create -g resource_group -n nic_name --subnet subnet_name --vnet-
name virtual_network --location location --network-security-group
network_security_group --private-ip-address-version IPv4
```

たとえば、以下のコマンドは、**shareblock-nodea-vm-nic-protected** という名前のネットワークインターフェイスを作成します。

```
$ az network nic create -g sharedblock-rg -n sharedblock-nodea-vm-nic-protected --subnet
sharedblock-subnet-protected --vnet-name sharedblock-vn --location westcentralus --
network-security-group sharedblock-nsg --private-ip-address-version IPv4
```

4. Azure コマンド **az vm create** を使用して 3 つの仮想マシンを作成し、共有ブロックボリュームを割り当てます。オプションの値は、各仮想マシンに独自の **<vm_name>**、**<new_vm_disk_name>**、**<nic_name>** を持つ点で異なります。

```
$ az vm create -n vm_name -g resource_group --attach-data-disks
shared_block_volume_name --data-disk-caching None --os-disk-caching ReadWrite --os-
disk-name new-vm-disk-name --os-disk-size-gb disk_size --location location --size
virtual_machine_size --image image_name --admin-username vm_username --
authentication-type ssh --ssh-key-values ssh_key --nics -nic_name_ --availability-set
availability_set --ppg proximity_placement_group
```

たとえば、次のコマンドは、**sharedblock-nodea-vm** という名前の仮想マシンを作成します。

```
$ az vm create -n sharedblock-nodea-vm -g sharedblock-rg --attach-data-disks shared-
block-volume.vhd --data-disk-caching None --os-disk-caching ReadWrite --os-disk-name
sharedblock-nodea-vm.vhd --os-disk-size-gb 64 --location westcentralus --size
Standard_D2s_v3 --image /subscriptions/12345678910-
```

```
12345678910/resourceGroups/sample-
azureimagesgroupwestcentralus/providers/Microsoft.Compute/images/sample-azure-rhel-
7.0-20200713.n.0.x86_64 --admin-username sharedblock-user --authentication-type ssh --
ssh-key-values @sharedblock-key.pub --nics sharedblock-nodea-vm-nic-protected --
availability-set sharedblock-as --ppg sharedblock-ppg

{
  "fqdns": "",
  "id": "/subscriptions/12345678910-12345678910/resourceGroups/sharedblock-
rg/providers/Microsoft.Compute/virtualMachines/sharedblock-nodea-vm",
  "location": "westcentralus",
  "macAddress": "00-22-48-5D-EE-FB",
  "powerState": "VM running",
  "privateIpAddress": "198.51.100.3",
  "publicIpAddress": "",
  "resourceGroup": "sharedblock-rg",
  "zones": ""
}
```

検証手順

1. クラスター内の各仮想マシンについて、仮想マシン **<ip_address>** で SSH コマンドを使用して、ブロックデバイスが利用できることを確認します。

```
# ssh ip_address "hostname ; lsblk -d | grep ' 1T '"
```

たとえば、次のコマンドは、仮想マシン IP **198.51.100.3** のホスト名およびブロックデバイスを含む詳細を一覧表示します。

```
# ssh 198.51.100.3 "hostname ; lsblk -d | grep ' 1T '"

nodea
sdb    8:16   0    1T  0 disk
```

2. SSH コマンドを使用して、クラスター内の各仮想マシンが同じ共有ディスクを使用していることを確認します。

```
# ssh _ip_address_s "hostname ; lsblk -d | grep ' 1T ' | awk '{print \$1}' | xargs -i udevadm info
--query=all --name=/dev/{} | grep '^E: ID_SERIAL='"
```

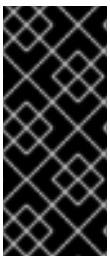
たとえば、以下のコマンドは、インスタンス IP アドレス **198.51.100.3** のホスト名および共有ディスクボリューム ID が含まれる詳細を一覧表示します。

```
# ssh 198.51.100.3 "hostname ; lsblk -d | grep ' 1T ' | awk '{print \$1}' | xargs -i udevadm info -
-query=all --name=/dev/{} | grep '^E: ID_SERIAL='"

nodea
E: ID_SERIAL=3600224808dd8eb102f6ffc5822c41d89
```

共有ディスクが各仮想マシンに割り当てられていることを確認したら、クラスターの回復性の高いストレージを設定できます。Red Hat High Availability クラスターに回復性の高いストレージを設定する方法は、クラスターに GFS2 ファイルシステムを設定 を参照してください。GFS2 ファイルシステムに関する一般的な情報は、GFS2 ファイルシステムの設定と管理 を参照してください。

第3章 AMAZON WEB SERVICES での EC2 インスタンスとしての RED HAT ENTERPRISE LINUX イメージのデプロイメント

Amazon Web Services (AWS) に EC2 インスタンスとして Red Hat Enterprise Linux (RHEL) 7 イメージをデプロイするオプションは多数あります。本章では、イメージを選択するオプションを説明し、ホストシステムおよび仮想マシンのシステム要件の一覧を紹介します。本章では、ISO イメージからカスタム VM を作成し、そのイメージを EC2 にアップロードして、EC2 インスタンスを起動する手順も説明します。



重要

ISO イメージからカスタム仮想マシンを作成することは可能ですが、Red Hat Image Builder 製品を使用して、特定のクラウドプロバイダーで使用するようにカスタマイズされたイメージを作成することを推奨します。Image Builder を使用すると、AMI (Amazon Machine Image **ami** 形式) を作成およびアップロードできます。詳細は Image Builder ガイド を参照してください。

この章では、Amazon のドキュメントを参照している箇所が多数あります。手順の詳細は、参照している Amazon ドキュメントを確認してください。

注記

AWS でセキュアに使用できる Red Hat 製品の一覧は、Red Hat on Amazon Web Services を参照してください。

前提条件

- Red Hat カスタマーポータル のアカウントに登録している。
- AWS にサインアップして、AWS リソースを設定します。詳細は Setting Up with Amazon EC2 を参照してください。
- Red Hat Cloud Access プログラム でサブスクリプションを有効にします。Red Hat Cloud Access プログラムでは、Red Hat のサブスクリプションを、物理システムまたはオンプレミスシステムから、Red Hat のフルサポートのある AWS へ移動できます。

関連情報

- Red Hat Cloud Access Reference Guide
- Red Hat in the Public Cloud
- Red Hat Enterprise Linux on Amazon EC2 - FAQs
- Amazon EC2 での設定
- Red Hat on Amazon Web Services

3.1. AWS での RED HAT ENTERPRISE LINUX イメージオプション

以下の表には、イメージの選択肢を記載し、イメージオプションの相違点を示しています。

表3.1 イメージオプション

イメージオプション	サブスクリプション	サンプルシナリオ	留意事項
Red Hat Gold Image のデプロイを選択する	既存の Red Hat サブスクリプションを活用する	Red Hat Cloud Access プログラム を使用してサブスクリプションを有効にし、AWS で Red Hat Gold Image を選択します。	このサブスクリプションには、Red Hat のコストが含まれていますが、他のインスタンスのコストは、Amazon 社に支払うことになります。 Red Hat Gold Image は、既存の Red Hat サブスクリプションを活用するため、クラウドアクセスイメージと呼ばれています。Red Hat は、クラウドアクセスイメージを直接サポートします。
AWS に移動するカスタムイメージをデプロイすることを選択する	既存の Red Hat サブスクリプションを活用する	Red Hat Cloud Access プログラム を使用してサブスクリプションを有効にし、カスタムイメージをアップロードし、サブスクリプションを割り当てます。	このサブスクリプションには、Red Hat のコストが含まれていますが、他のインスタンスのコストは、Amazon 社に支払うことになります。 AWS に移行するカスタムイメージは、既存の Red Hat サブスクリプションを活用するため、クラウドアクセスイメージと呼ばれています。Red Hat は、クラウドアクセスイメージを直接サポートします。
RHEL を含む既存の Amazon イメージをデプロイすることを選択する	AWS EC2 イメージには Red Hat 製品が含まれる	AWS マネジメントコンソール でのインスタンスの起動時に RHEL イメージを選択するか、AWS Marketplace からイメージを選択します。	Amazon 社に、従量課金モデルで 1 時間ごとに支払います。このようなイメージはオンデマンドイメージと呼ばれています。Amazon 社はオンデマンドイメージをサポートします。 Red Hat は、イメージの更新を提供します。AWS により、Red Hat Update Infrastructure (RHUI) から更新を利用できるようにします。

注記

Red Hat Image Builder を使用して、AWS 用のカスタムイメージを作成できます。詳細は Image Builder ガイド を参照してください。

重要

オンデマンドインスタンスは、Red Hat Cloud Access インスタンスに変換できません。オンデマンドイメージから Red Hat Cloud Access bring-your-own-subscription (BYOS) イメージに変更するには、Red Hat Cloud Access インスタンスを新たに作成し、オンデマンドインスタンスからデータを移行します。データを移行した後に、オンデマンドのインスタンスをキャンセルして二重請求を回避します。

本章の残りの部分には、カスタムイメージに関する情報および手順が記載されています。

関連情報

- Red Hat Gold Image の使用
- Red Hat Cloud Access プログラム
- Image Builder ガイド
- AWS マネジメントコンソール
- AWS Marketplace

3.2. AWS CLI のインストール

本章の多くの手順には、AWS CLI の使用が含まれます。AWS CLI をインストールするには、以下の手順を実行します。

前提条件

AWS アクセスキー ID および AWS シークレットアクセスキーを作成していてアクセスできる。情報および手順は、AWS CLI の設定 を参照してください。

手順

1. Python 3 および **pip** ツールをインストールします。

```
# yum install python3
# yum install python3-pip
```

2. **pip** コマンドを使用して、AWS コマンドラインツール をインストールします。

```
# pip3 install awscli
```

3. **aws --version** コマンドを実行して、AWS CLI をインストールしたことを確認します。

```
$ aws --version
aws-cli/1.16.182 Python/2.7.5 Linux/3.10.0-957.21.3.el7.x86_64 botocore/1.12.172
```

4. AWS アクセスの詳細に従って、AWS コマンドラインクライアントを設定します。

```
$ aws configure
AWS Access Key ID [None]:
AWS Secret Access Key [None]:
Default region name [None]:
Default output format [None]:
```

関連情報

- AWS CLI の設定
- AWS コマンドラインツール

3.3. 仮想マシン設定設定

クラウド仮想マシンには、以下の設定設定が必要です。

表3.2 仮想マシンの設定設定

設定	推奨事項
ssh	仮想マシンへのリモートアクセスを確立するには、**ssh** を有効にする必要があります。
dhcp	プライマリー仮想アダプターは、dhcp 用に設定する必要があります。

3.4. ISO イメージからのベース仮想マシンの作成

このセクションの手順に従って、ISO イメージからベースイメージを作成します。

前提条件

仮想化の 導入および管理ガイド に従って、Red Hat Enterprise Linux 7 ホストマシンの仮想化を有効にします。

3.4.1. ISO イメージのダウンロード

手順

1. Red Hat カスタマーポータル から最新の Red Hat Enterprise Linux ISO Image をダウンロードします。
2. イメージを **/var/lib/libvirt/images** ディレクトリーに移動します。

3.4.2. ISO イメージからの仮想マシンの作成

手順

1. 仮想化用のホストマシンを有効にしていることを確認します。仮想化パッケージのインストールに関する情報および手順は、既存の Red Hat Enterprise Linux システム上への仮想化パッケージのインストール を参照してください。

2. 基本的な Red Hat Enterprise Linux 仮想マシンを作成し、起動します。仮想マシンの作成手順は、仮想マシンの作成 を参照してください。

   a. コマンドラインを使用して仮想マシンを作成する場合は、デフォルトのメモリーと CPU を仮想マシンの容量に設定するようにしてください。仮想ネットワークインターフェイスを **virtio** に設定します。
   基本的なコマンドラインの例を以下に示します。

   ```
   virt-install --name _vmname_ --memory 2048 --vcpus 2 --disk size=8,bus=virtio --location
   rhel-7.0-x86_64-dvd.iso --os-variant=rhel7.0
   ```

   b. virt-manager アプリケーションを使用して仮想マシンを作成する場合は、virt-manager を使用したゲストの作成 の手順を実行します。以下の点に注意してください。

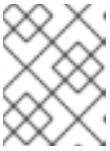
   - **仮想マシンをすぐに起動** は選択しないでください。
   - **メモリー** と **ストレージのサイズ** を、希望の設定に変更します。
   - インストールを開始する前に、**仮想ネットワークインターフェイス設定** で **モデル** を **virtio** に変更し、**vCPUs** を仮想マシンの容量設定に変更していることを確認します。

3.4.3. RHEL インストールの完了

仮想マシンが起動したら、以下の手順を実行してインストールを完了し、root アクセスを有効にします。

手順

1. インストールプロセス中に使用する言語を選択します。
2. **インストール概要** ビューで、以下を行います。
   a. **ソフトウェアの選択** をクリックし、**最小インストール** を選択します。
   b. **完了** をクリックします。
   c. **インストール先** をクリックし、**ストレージ設定** で **カスタム** を選択します。
   - **/boot** で、500 MB 以上であることを確認してください。残りの領域は、root / に使用できます。
   - 標準のパーティションが推奨されますが、論理ボリューム管理 (LVM) を使用することも可能です。
   - ファイルシステムには、xfs、ext4、ext3 などを使用できます。
   - 変更が完了したら、**完了** をクリックします。
3. **インストールの開始** をクリックします。
4. **Root パスワード** を設定します。
5. インストールが完了したら、仮想マシンを再起動して **root** でログインします。
6. イメージを設定します。



注記

cloud-init パッケージがインストールされ、有効になっていることを確認します。

7. 仮想マシンの電源をオフにします。

3.5. RED HAT ENTERPRISE LINUX イメージの AWS へのアップロード

ホストマシンで本セクションにある手順に従って、イメージを AWS にアップロードします。

3.5.1. S3 バケットの作成

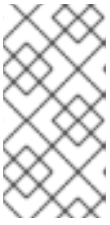
AWS にインポートするには、Amazon S3 バケットが必要です。Amazon S3 バケットは、オブジェクトを格納する Amazon リソースです。イメージのアップロードプロセスの一環として、S3 バケットを作成し、イメージをバケットに移動します。以下の手順に従って、バケットを作成します。

前提条件

- AWS CLI がインストールされていること。詳細は、AWS CLI のインストール を参照してください。

手順

1. Amazon S3 コンソール を起動します。
2. **Create Bucket** をクリックします。**Create Bucket** ダイアログが表示されます。
3. **Name and region** ビューで、以下を行います。
   a. **Bucket name** を入力します。
   b. **Region** を選択します。フィールドにリージョンを入力するか、ドロップダウンをクリックして、利用可能な全リージョンから使用するリージョンを選択します。
   c. **次へ** をクリックします。
4. **Configure options** ビューでは、目的のオプションを選択し、**Next** をクリックします。
5. **Set permissions** ビューで、デフォルトのオプションを変更または受け入れ、**Next** をクリックします。
6. バケットの設定を確認します。
7. **Create bucket** をクリックします。



注記

AWS CLI を使用してバケットを作成することもできます。たとえば、**aws s3 mb s3://my-new-bucket** は、**my-new-bucket** という名前の S3 バケットを作成します。**mb** コマンドの詳細は、AWS CLI コマンドリファレンス を参照してください。

関連情報

- Amazon S3 Console
- AWS CLI コマンドリファレンス

3.5.2. vmimport ロールの作成

仮想マシンのインポートに必要な **vmimport** ロールを作成するには、以下の手順を実行します。詳細は、Amazon ドキュメントの VM Import Service Role を参照してください。

手順

1. **trust-policy.json** という名前のファイルを作成し、以下のポリシーを追加します。システムの任意の場所にファイルを保存し、その場所を書き留めます。

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Principal": { "Service": "vmie.amazonaws.com" },
         "Action": "sts:AssumeRole",
         "Condition": {
            "StringEquals":{
               "sts:Externalid": "vmimport"
            }
         }
      }
   ]
}
```

2. **create role** コマンドを実行して **vmimport** ロールを作成します。**trust-policy.json** ファイルの場所への完全なパスを指定します。**file:**// の接頭辞をパスに設定します。以下に例を示します。

```
aws iam create-role --role-name vmimport --assume-role-policy-document
file:///home/sample/ImportService/trust-policy.json
```

3. **role-policy.json** という名前のファイルを作成し、以下のポリシーを追加します。**s3-bucket-name** を、S3 バケットの名前に置き換えます。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetBucketLocation",
            "s3:GetObject",
            "s3:ListBucket"
         ],
         "Resource":[
            "arn:aws:s3:::s3-bucket-name",
            "arn:aws:s3:::s3-bucket-name/*"
         ]
      },
      {
```

```
            "Effect":"Allow",
            "Action":[
               "ec2:ModifySnapshotAttribute",
               "ec2:CopySnapshot",
               "ec2:RegisterImage",
               "ec2:Describe*"
            ],
            "Resource":"*"
         }
      ]
}
```

4. **put-role-policy** コマンドを使用して、作成したロールにポリシーを割り当てます。**role-policy.json** ファイルの完全パスを指定します。以下に例を示します。

```
aws iam put-role-policy --role-name vmimport --policy-name vmimport --policy-document file:///home/sample/ImportService/role-policy.json
```

関連情報

- VM インポートサービス出力
- 必要なサービス出力

3.5.3. AMI の S3 への変換およびプッシュ

Amazon Machine Image (AMI) を S3 に変換してプッシュするには、以下の手順を実行します。サンプルは典型的なもので、**qcow2** ファイル形式でフォーマットされたイメージを **raw** 形式に変換します。Amazon では、**OVA**、**VHD**、**VHDX**、**VMDK**、および **raw** の形式のイメージを利用できます。Amazon で利用できるイメージ形式の詳細は、VM import/Export の仕組み を参照してください。

手順

1. **qemu-img** コマンドを実行してイメージを変換します。以下に例を示します。

```
qemu-img convert -f qcow2 -O raw rhel-server-7.7-1-x86_64-kvm.qcow2 rhel-server-7.7-1-x86_64-kvm.raw
```

2. イメージを S3 にプッシュします。

```
aws s3 cp rhel-server-7.7.1-x86_64-kvm.raw s3://s3-_bucket-name_
```

注記

この手順では数分かかる場合があります。完了したら、AWS S3 コンソール を使用して、イメージが S3 バケットに正常にアップロードされたことを確認できます。

関連情報

- VM のインポート/エクスポートの仕組み
- AWS S3 コンソール

3.5.4. raw イメージからの AMI の作成

raw イメージから AMI を作成するには、以下の手順を実行します。

前提条件

- AWS CLI がインストールされていること。詳細は、AWS CLI のインストール を参照してください。

手順

- AWS CLI で **aws ec2 import-image** コマンドを実行すると、raw イメージから AMI を作成できます。

```
# aws ec2 import-image --platform Linux --license-type BYOL --no-encrypted --description
_imagedescription_ --architecture x86_64 --disk-containers Format=Raw,UserBucket="
{S3Bucket=virtqes1,S3Key=rhel-server-ec2-7.9-30.x86_64.raw}" --region _regionname_
```

関連情報

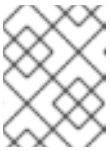
- 仮想マシンをイメージとしてインポート

3.5.5. AMI からのインスタンスの起動

AMI からインスタンスを起動して設定するには、以下の手順を行います。

手順

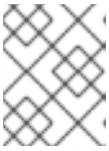
1. AWS EC2 Dashboard から、**Images** を選択して、**AMI** を選択します。
2. イメージを右クリックして、**Launch** を選択します。
3. ワークロードの要件を満たす、もしくは超過する **Instance Type** を選択します。
   インスタンスタイプに関する情報は、Amazon EC2 Instance Types を参照してください。
4. **Next:Configure Instance Details** をクリックします。
   a. 作成する **インスタンス数** を入力します。
   b. **Network** で、AWS 環境でのセットアップ の際に作成した VPC を選択します。インスタンスのサブネットを選択するか、新しいサブネットを作成します。
   c. 自動割り当てパブリック IP では、**Enable** を選択します。



注記

これらは、基本インスタンスの作成に必要な最小限の設定オプションです。アプリケーション要件に応じて追加オプションを確認します。

5. **Next: Add Storage** をクリックします。デフォルトのストレージが十分であることを確認してください。
6. **Next: Add Tags** をクリックします。



注記

タグを使用すると、AWS リソースの管理に役立ちます。タグ付けの詳細は、Amazon EC2 リソースにタグを付ける を参照してください。

7. **Next:Configure Security Group** をクリックします。AWS 環境でのセットアップ の際に作成したセキュリティーグループを選択します。
8. **Review and Launch** をクリックします。選択内容を確認します。
9. **Launch** をクリックします。既存の鍵のペアの選択、または新しい鍵のペアの作成に関するダイアログが表示されます。AWS 環境でのセットアップ 時に作成した鍵のペアを選択します。

注記

秘密鍵のパーミッションが正しいことを確認します。必要に応じて **chmod 400 <keyname>.pem** コマンドオプションを使用してパーミッションを変更します。

10. **Launch Instances** をクリックします。
11. **View Instances** をクリックします。インスタンスに名前を付けることができます。インスタンスを選択して **Connect** をクリックすると、インスタンスへの SSH セッションを開始できます。**A standalone SSH client** に記載されているコマンドの例を使用してください。

注記

または、AWS CLI を使用してインスタンスを起動することもできます。詳細は、Amazon 社のドキュメントの Launching, Listing, and Terminating Amazon EC2 Instances を参照してください。

関連情報

- AWS マネジメントコンソール
- Amazon EC2 での設定
- Amazon EC2 インスタンス
- Amazon EC2 インスタンスタイプ

3.5.6. Red Hat サブスクリプションの割り当て

Red Hat Cloud Access プログラムで有効になっているサブスクリプションを割り当てるには、以下の手順を行います。

前提条件

サブスクリプションが有効になっている。

手順

1. システムを登録します。

```
subscription-manager register --auto-attach
```

2. サブスクリプションを割り当てます。

    - アクティベーションキーを使用して、サブスクリプションを割り当てることができます。カスタマーポータルのアクティベーションキーを作成する を参照してください。

    - または、サブスクリプションプール (Pool ID) の ID を使用してサブスクリプションを手動で割り当てることができます。コマンドラインでのサブスクリプションのアタッチと削除を参照してください。

関連情報

- カスタマーポータルのアクティベーションキーを作成する
- コマンドラインでのサブスクリプションのアタッチと削除
- Red Hat Subscription Manager の使用および設定

第4章 AWS での RED HAT HIGH AVAILABILITY クラスターの設定

本章では、EC2 インスタンスをクラスターノードとして使用し、Amazon Web Services (AWS) での Red Hat High Availability (HA) クラスターを設定する情報および手順を説明します。クラスターに使用する Red Hat Enterprise Linux (RHEL) イメージを取得するオプションは複数あります。AWS のイメージオプションの詳細は、AWS での Red Hat Enterprise Linux Image オプション を参照してください。

本章では、AWS の環境設定の前提条件を説明します。環境を設定したら、EC2 インスタンスを作成および設定できます。

本章では、個別のノードを AWS の HA ノードのクラスターに変換する HA クラスターの作成に固有の手順も説明します。これには、各クラスターノードに高可用性パッケージおよびエージェントをインストールし、フェンシングを設定し、AWS ネットワークリソースエージェントをインストールする手順が含まれます。

この章では、Amazon のドキュメントを参照している箇所が多数あります。多くの手順に関する詳細は、参照している Amazon ドキュメントを確認してください。

前提条件

- AWS コマンドラインインターフェイス (CLI) をインストールする必要があります。AWS CLI のインストールに関する詳細は、AWS CLI のインストール を参照してください。
- Red Hat Cloud Access プログラム でサブスクリプションを有効にします。Red Hat Cloud Access プログラムでは、Red Hat のサブスクリプションを、物理システムまたはオンプレミスシステムから、Red Hat のフルサポートのある AWS へ移動できます。

関連情報

- Red Hat Cloud Access Reference Guide
- Red Hat in the Public Cloud
- Red Hat Enterprise Linux on Amazon EC2 - FAQs
- Amazon EC2 での設定
- Red Hat on Amazon Web Services
- Support Policies for RHEL High Availability Clusters

4.1. AWS アクセスキーおよび AWS シークレットアクセスキーの作成

AWS CLI をインストールする前に、AWS アクセスキーおよび AWS シークレットアクセスキーを作成する必要があります。フェンシングおよびリソースエージェントの API は AWS アクセスキーおよびシークレットアクセスキーを使用してクラスター内の各ノードに接続します。

以下の手順に従って、キーを作成します。

前提条件

IAM ユーザーアカウントに Programmatic アクセスがある。詳細は、Setting up the AWS Environment を参照してください。

手順

手順

1. AWS コンソール を起動します。
2. AWS アカウント ID をクリックしてドロップダウンメニューを表示し、**My Security Credentials** を選択します。
3. **Users** をクリックします。
4. ユーザーを選択し、**Summary** 画面を開きます。
5. **Security credentials** タブをクリックします。
6. **Create access key**をクリックします。
7. **.csv** ファイルをダウンロード (または両方の鍵を保存) します。フェンスデバイスの作成時に、この鍵を入力する必要があります。

4.2. HA パッケージおよびエージェントのインストール

全ノードで以下の手順を実行し、HA パッケージおよびエージェントをインストールします。

手順

1. 以下のコマンドを入力して、AWS Red Hat Update Infrastructure (RHUI) クライアントを削除します。Red Hat Cloud Access サブスクリプションを使用するため、サブスクリプションに加えて AWS RHUI を使用しないでください。

```
$ sudo -i
# yum -y remove rh-amazon-rhui-client*
```

2. 仮想マシンを Red Hat に登録します。

```
# subscription-manager register --auto-attach
```

3. すべてのリポジトリーを無効にします。

```
# subscription-manager repos --disable=*
```

4. RHEL 7 Server リポジトリーおよび RHEL 7 Server HA リポジトリーを有効にします。

```
# subscription-manager repos --enable=rhel-7-server-rpms
# subscription-manager repos --enable=rhel-ha-for-rhel-7-server-rpms
```

5. すべてのパッケージを更新します。

```
# yum update -y
```

6. カーネルを更新したら再起動します。

```
# reboot
```

7. pcs、pacemaker、fence agent およびリソースエージェントをインストールします。

```
# yum -y install pcs pacemaker fence-agents-aws resource-agents
```

8. **hacluster** ユーザーは、前の手順で **pcs** および **pacemaker** のインストール時に作成されました。すべてのクラスターノードに **hacluster** のパスワードを作成します。すべてのノードに同じパスワードを使用します。

```
# passwd hacluster
```

9. **firewalld.service** が有効化されている場合は、RHEL ファイアウォールに **high availability** サービスを追加します。

```
# firewall-cmd --permanent --add-service=high-availability
# firewall-cmd --reload
```

10. **pcs** サービスを起動し、システムの起動時に開始できるようにします。

```
# systemctl enable pcsd.service --now
```

検証手順

pcs サービスが実行していることを確認します。

```
# systemctl is-active pcsd.service
```

4.3. クラスターの作成

ノードのクラスターを作成するには、以下の手順を実施します。

手順

1. ノードのいずれかで以下のコマンドを実行し、pcs ユーザー **hacluster** を認証します。クラスターの各ノードの名前を指定します。

```
# pcs host auth  _hostname1_ _hostname2_ _hostname3_
```

たとえば、以下のようになります。

```
[root@node01 clouduser]# pcs host auth node01 node02 node03
Username: hacluster
Password:
node01: Authorized
node02: Authorized
node03: Authorized
```

2. クラスターを作成します。

```
# pcs cluster setup --name _hostname1_ _hostname2_ _hostname3_
```

たとえば、以下のようになります。

```
[root@node01 clouduser]# pcs cluster setup --name newcluster node01 node02 node03
```

```
...omitted

Synchronizing pcsd certificates on nodes node01, node02, node03...
node02: Success
node03: Success
node01: Success
Restarting pcsd on the nodes in order to reload the certificates...
node02: Success
node03: Success
node01: Success
```

検証手順

1. クラスターを有効にします。

```
# pcs cluster enable --all
```

2. クラスターを起動します。

```
# pcs cluster start --all
```

たとえば、以下のようになります。

```
[root@node01 clouduser]# pcs cluster enable --all
node02: Cluster Enabled
node03: Cluster Enabled
node01: Cluster Enabled

[root@node01 clouduser]# pcs cluster start --all
node02: Starting Cluster...
node03: Starting Cluster...
node01: Starting Cluster...
```

4.4. フェンスデバイスの作成

フェンシングを設定するには、以下の手順を実行します。

手順

1. 以下の AWS メタデータクエリーを入力し、各ノードのインスタンス ID を取得します。フェンスデバイスを設定するには、これらの ID が必要です。詳細は Instance Metadata and User Data を参照してください。

```
# echo $(curl -s http://169.254.169.254/latest/meta-data/instance-id)
```

たとえば、以下のようになります。

```
[root@ip-10-0-0-48 ~]# echo $(curl -s http://169.254.169.254/latest/meta-data/instance-id) i-07f1ac63af0ec0ac6
```

2. フェンスデバイスを作成します。**pcmk_host_map** コマンドを使用して、RHEL ホスト名をインスタンス ID にマッピングします。AWS アクセスキーおよび AWS シークレットアクセスキーの作成 で先ほど設定した AWS アクセスキーおよび AWS シークレットアクセスキーを使用し

ます。

```
# pcs stonith create cluster_fence fence_aws access_key=access-key secret_key=_secret-access-key_ region=_region_ pcmk_host_map="rhel-hostname-1:Instance-ID-1;rhel-hostname-2:Instance-ID-2;rhel-hostname-3:Instance-ID-3"
```

たとえば、以下のようになります。

```
[root@ip-10-0-0-48 ~]# pcs stonith create clusterfence fence_aws access_key=AKIAI*******6MRMJA secret_key=a75EYIG4RVL3h*******K7koQ8dzaDyn5yoIZ/ region=us-east-1 pcmk_host_map="ip-10-0-0-48:i-07f1ac63af0ec0ac6;ip-10-0-0-46:i-063fc5fe93b4167b2;ip-10-0-0-58:i-08bd39eb03a6fd2c7" power_timeout=240 pcmk_reboot_timeout=480 pcmk_reboot_retries=4
```

検証手順

1. 他のノードのいずれかに対してフェンスエージェントをテストします。

```
# pcs stonith fence _awsnodename_
```

たとえば、以下のようになります。

```
[root@ip-10-0-0-48 ~]# pcs stonith fence ip-10-0-0-58
Node: ip-10-0-0-58 fenced
```

2. ステータスを確認して、ノードがフェンスされていることを確認します。

```
# watch pcs status
```

たとえば、以下のようになります。

```
[root@ip-10-0-0-48 ~]# pcs status
Cluster name: newcluster
Stack: corosync
Current DC: ip-10-0-0-46 (version 1.1.18-11.el7-2b07d5c5a9) - partition with quorum
Last updated: Fri Mar  2 20:01:31 2018
Last change: Fri Mar  2 19:24:59 2018 by root via cibadmin on ip-10-0-0-48

3 nodes configured
1 resource configured

Online: [ ip-10-0-0-46 ip-10-0-0-48 ip-10-0-0-58 ]

Full list of resources:

  clusterfence  (stonith:fence_aws):    Started ip-10-0-0-46

Daemon Status:
  corosync: active/disabled
  pacemaker: active/disabled
  pcsd: active/enabled
```

4.5. クラスターノードへの AWS CLI のインストール

以前のバージョンでは、AWS CLI をホストシステムにインストールしていました。ネットワークリソースエージェントを設定する前に、クラスターノードに AWS CLI をインストールする必要があります。

各クラスターノードで以下の手順を実行します。

前提条件

AWS アクセスキーおよび AWS シークレットアクセスキーを作成している。詳細は、AWS アクセスキーおよび AWS シークレットアクセスキーの作成 を参照してください。

手順

1. AWS CLI のインストール の手順を行います。
2. 以下のコマンドを実行して、AWS CLI が適切に設定されていることを確認します。インスタンス ID およびインスタンス名が表示されます。
   以下に例を示します。

```
[root@ip-10-0-0-48 ~]# aws ec2 describe-instances --output text --query
'Reservations[*].Instances[*].[InstanceId,Tags[?Key==`Name`].Value]'
i-07f1ac63af0ec0ac6
ip-10-0-0-48
i-063fc5fe93b4167b2
ip-10-0-0-46
i-08bd39eb03a6fd2c7
ip-10-0-0-58
```

4.6. ネットワークリソースエージェントのインストール

HA 操作が機能するために、クラスターは AWS ネットワークリソースエージェントを使用してフェイルオーバー機能を有効にします。設定された時間内にノードがハートビートチェックに応答しない場合、ノードはフェンスされ、操作はクラスター内の追加のノードにフェイルオーバーします。これを使用するには、ネットワークリソースエージェントを設定する必要があります。

順序 と **コロケーション** の制約を適用するため、同じグループ に 2 つのリソースを追加します。

セカンダリープライベート IP リソースと仮想 IP リソースを作成

セカンダリープライベート IP アドレスを追加し、仮想 IP を作成するには、以下の手順を行います。この手順は、クラスター内の任意のノードから実行できます。

手順

1. 以下のコマンドを実行して、**AWS Secondary Private IP Address** リソースエージェント (awsvip) の説明を表示します。これは、このエージェントのオプションとデフォルトの操作を示しています。

```
# pcs resource describe awsvip
```

2. 次のコマンドを実行して、**VPC CIDR** ブロックで未使用のプライベート IP アドレスを使用して 2 番目のプライベート IP アドレスを作成します。

```
# pcs resource create privip awsvip secondary_private_ip=_Unused-IP-Address_ --group
_group-name_
```

以下に例を示します。

```
[root@ip-10-0-0-48 ~]# pcs resource create privip awsvip secondary_private_ip=10.0.0.68 --
group networking-group
```

3. 仮想 IP リソースを作成します。これは、フェンシングされたノードからフェイルオーバーノードに即時に再マッピングできる VPC IP アドレスで、サブネット内のフェンスされたノードの失敗をマスクします。

```
# pcs resource create vip IPaddr2 ip=_secondary-private-IP_ --group _group-name_
```

たとえば、以下のようになります。

```
root@ip-10-0-0-48 ~]# pcs resource create vip IPaddr2 ip=10.0.0.68 --group networking-
group
```

検証手順

pcs status コマンドを実行して、リソースが実行していることを確認します。

```
# pcs status
```

以下に例を示します。

```
[root@ip-10-0-0-48 ~]# pcs status
Cluster name: newcluster
Stack: corosync
Current DC: ip-10-0-0-46 (version 1.1.18-11.el7-2b07d5c5a9) - partition with quorum
Last updated: Fri Mar  2 22:34:24 2018
Last change: Fri Mar  2 22:14:58 2018 by root via cibadmin on ip-10-0-0-46

3 nodes configured
3 resources configured

Online: [ ip-10-0-0-46 ip-10-0-0-48 ip-10-0-0-58 ]

Full list of resources:

clusterfence    (stonith:fence_aws):    Started ip-10-0-0-46
 Resource Group: networking-group
     privip (ocf::heartbeat:awsvip):    Started ip-10-0-0-48
     vip    (ocf::heartbeat:IPaddr2):   Started ip-10-0-0-58

Daemon Status:
  corosync: active/disabled
  pacemaker: active/disabled
  pcsd: active/enabled
```

Elastic IP アドレスの作成

Elastic IP アドレスは、フェンシングされたノードからフェイルオーバーノードに即時に再マッピングできるパブリック IP アドレスで、フェンスされたノードの障害をマスクします。

これは、先に作成した仮想 IP リソースとは異なることに注意してください。Elastic IP アドレスは、サブネット接続ではなく、公開用インターネット接続に使用されます。

1. 上で作成した 同じグループ に 2 つのリソースを追加して、**順序** と **コロケーション** の制約を適用します。
2. 以下の AWS CLI コマンドを入力して、Elastic IP アドレスを作成します。

```
[root@ip-10-0-0-48 ~]# aws ec2 allocate-address --domain vpc --output text
eipalloc-4c4a2c45   vpc 35.169.153.122
```

3. 以下のコマンドを実行して、AWS のセカンダリー Elastic IP Address リソースエージェント (awseip) の説明を表示します。これは、このエージェントのオプションとデフォルトの操作を示しています。

```
# pcs resource describe awseip
```

4. ステップ 1 で作成して割り当てられた IP アドレスを使用して、セカンダリー Elastic IP アドレスリソースを作成します。

```
# pcs resource create elastic awseip elastic_ip=_Elastic-IP-Address_allocation_id=_Elastic-IP-Association-ID_ --group networking-group
```

たとえば、以下のようになります。

```
# pcs resource create elastic awseip elastic_ip=35.169.153.122 allocation_id=eipalloc-4c4a2c45 --group networking-group
```

検証手順

pcs status コマンドを実行して、リソースが実行していることを確認します。

```
# pcs status
```

以下に例を示します。

```
[root@ip-10-0-0-58 ~]# pcs status
Cluster name: newcluster
Stack: corosync
Current DC: ip-10-0-0-58 (version 1.1.18-11.el7-2b07d5c5a9) - partition with quorum
Last updated: Mon Mar  5 16:27:55 2018
Last change: Mon Mar  5 15:57:51 2018 by root via cibadmin on ip-10-0-0-46

3 nodes configured
4 resources configured

Online: [ ip-10-0-0-46 ip-10-0-0-48 ip-10-0-0-58 ]

Full list of resources:

 clusterfence   (stonith:fence_aws):    Started ip-10-0-0-46
```

```
 Resource Group: networking-group
     privip (ocf::heartbeat:awsvip):  Started ip-10-0-0-48
     vip    (ocf::heartbeat:IPaddr2):    Started ip-10-0-0-48
     elastic (ocf::heartbeat:awseip):    Started ip-10-0-0-48

 Daemon Status:
   corosync: active/disabled
   pacemaker: active/disabled
   pcsd: active/enabled
```

Elastic IP アドレスのテスト

以下のコマンドを入力して、仮想 IP (awsvip) および Elastic IP (awseip) のリソースが機能していることを確認します。

手順

1. ローカルワークステーションから、上で作成した Elastic IP アドレスへの SSH セッションを開始します。

   ```
   $ ssh -l ec2-user -i ~/.ssh/<KeyName>.pem elastic-IP
   ```

   以下に例を示します。

   ```
   $ ssh -l ec2-user -i ~/.ssh/cluster-admin.pem 35.169.153.122
   ```

2. SSH 経由で接続したホストが、作成された elastic リソースに関連付けられたホストであることを確認します。

関連情報

- High Availability アドオンの概要
- High Availability Add-On の管理
- High Availability Add-On リファレンス

4.7. 共有ブロックストレージの設定

このセクションでは、Amazon EBS のマルチアタッチボリュームを使用する Red Hat High Availability クラスターの共有ブロックストレージを設定するオプションの手順を説明します。この手順では、1TB 共有ディスクを持つ 3 つのインスタンス (3 ノードクラスター) を想定しています。

手順

1. AWS コマンド create-volume を使用して共有ブロックボリュームを作成 します。

   ```
   $ aws ec2 create-volume --availability-zone availability_zone --no-encrypted --size 1024 --volume-type io1 --iops 51200 --multi-attach-enabled
   ```

   たとえば、以下のコマンドは、**us-east-1a** アベイラビリティーゾーンにボリュームを作成します。

   ```
   $ aws ec2 create-volume --availability-zone us-east-1a --no-encrypted --size 1024 --volume-
   ```

```
type io1 --iops 51200 --multi-attach-enabled

{
    "AvailabilityZone": "us-east-1a",
    "CreateTime": "2020-08-27T19:16:42.000Z",
    "Encrypted": false,
    "Size": 1024,
    "SnapshotId": "",
    "State": "creating",
    "VolumeId": "vol-042a5652867304f09",
    "Iops": 51200,
    "Tags": [ ],
    "VolumeType": "io1"
}
```

注記

次の手順で **VolumeId** が必要になります。

2. クラスターの各インスタンスについて、AWS コマンド **attach-volume** を使用して共有ブロックボリュームを割り当てます。**<instance_id>** および **<volume_id>** を使用します。

```
$ aws ec2 attach-volume --device /dev/xvdd --instance-id instance_id --volume-id
volume_id
```

たとえば、以下のコマンドは共有ブロックボリューム **vol-042a5652867304f09** を **instance i-0eb803361c2c887f2** に接続します。

```
$ aws ec2 attach-volume --device /dev/xvdd --instance-id i-0eb803361c2c887f2 --volume-id
vol-042a5652867304f09

{
    "AttachTime": "2020-08-27T19:26:16.086Z",
    "Device": "/dev/xvdd",
    "InstanceId": "i-0eb803361c2c887f2",
    "State": "attaching",
    "VolumeId": "vol-042a5652867304f09"
}
```

検証手順

1. クラスター内の各インスタンスについて、インスタンス **<ip_address>** で SSH コマンドを使用して、ブロックデバイスが利用できるようにされていることを確認します。

```
# ssh <ip_address> "hostname ; lsblk -d | grep ' 1T '"
```

たとえば、以下のコマンドは、インスタンス IP **198.51.100.3** のホスト名およびブロックデバイスを含む詳細を一覧表示します。

```
# ssh 198.51.100.3 "hostname ; lsblk -d | grep ' 1T '"

nodea
nvme2n1 259:1    0   1T  0 disk
```

2. **SSH** コマンドを使用して、クラスター内の各インスタンスが同じ共有ディスクを使用していることを確認します。

```
# ssh ip_address "hostname ; lsblk -d | grep ' 1T ' | awk '{print \$1}' | xargs -i udevadm info --query=all --name=/dev/{} | grep '^E: ID_SERIAL='"
```

たとえば、以下のコマンドは、インスタンス IP アドレス **198.51.100.3** のホスト名および共有ディスクボリューム ID が含まれる詳細を一覧表示します。

```
 # ssh 198.51.100.3 "hostname ; lsblk -d | grep ' 1T ' | awk '{print \$1}' | xargs -i udevadm info --query=all --name=/dev/{} | grep '^E: ID_SERIAL='"

nodea
E: ID_SERIAL=Amazon Elastic Block Store_vol0fa5342e7aedf09f7
```

共有ディスクが各インスタンスに接続されていることを確認したら、クラスターの回復性の高いストレージを設定できます。Red Hat High Availability クラスターに回復性の高いストレージを設定する方法は、クラスターに GFS2 ファイルシステムを設定 を参照してください。GFS2 ファイルシステムに関する一般的な情報は、GFS2 ファイルシステムの設定と管理 を参照してください。

第5章 GOOGLE CLOUD PLATFORM での GOOGLE COMPUTE ENGINE インスタンスとしての RED HAT ENTERPRISE LINUX イメージのデプロイメント

Google Cloud Platform (GCP) に Google Compute Engine (GCE) インスタンスとして Red Hat Enterprise Linux (RHEL) 7 イメージをデプロイするオプションは複数あります。本章では、イメージを選択するオプションを説明し、ホストシステムおよび仮想マシンのシステム要件の一覧を紹介します。本章では、ISO イメージからカスタム VM を作成し、GCE にアップロードして、インスタンスを起動する手順を説明します。

この章では、Google のドキュメントを参照している箇所が多数あります。多くの手順に関する詳細は、参照している Google ドキュメントを参照してください。

注記

GCP 向けの Red Hat 製品認定の一覧は、Red Hat on Google Cloud Platform を参照してください。

前提条件

- Red Hat カスタマーポータル のアカウントがある (本章の手順を完了するのに必要)。
- Google Cloud Platform コンソールにアクセスするために、GCP でアカウントを作成している。詳細は Google Cloud を参照してください。
- Red Hat Cloud Access プログラム で、Red Hat サブスクリプションを有効にしている。Red Hat Cloud Access プログラムでは、Red Hat のサブスクリプションを、物理システムまたはオンプレミスシステムから、Red Hat のフルサポートのある GCP へ移動できます。

関連情報

- Red Hat in the Public Cloud
- Google Cloud

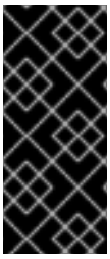
5.1. GCP での RED HAT ENTERPRISE LINUX イメージオプション

以下の表には、イメージの選択肢を記載し、イメージオプションの相違点を示しています。

表5.1 イメージオプション

イメージオプション	サブスクリプション	サンプルシナリオ	留意事項

イメージオプション	サブスクリプション	サンプルシナリオ	留意事項
GCP に移動するカスタムイメージをデプロイすることを選択する	既存の Red Hat サブスクリプションを活用する	Red Hat Cloud Access プログラム を使用してサブスクリプションを有効にし、カスタムイメージをアップロードし、サブスクリプションを割り当てます。	その他のすべてのインスタンスコストを支払うことになりますが、サブスクリプション自体には Red Hat 製品コストが含まれます。 GCP に移行するカスタムイメージは、既存の Red Hat サブスクリプションを活用するため、クラウドアクセスイメージと呼ばれています。Red Hat は、クラウドアクセスイメージを直接サポートします。
RHEL を含む既存の GCP イメージをデプロイすることを選択する	GCP イメージには、Red Hat 製品が含まれる	GCP Compute Engine でインスタンスを起動する時に RHEL イメージを選択するか、Google Cloud Platform Marketplace からイメージを選択します。	従量課金モデルでは、GCP に 1 時間ごとに支払います。このようなイメージはオンデマンドイメージと呼ばれています。GCP は、サポート契約に基づいてオンデマンドイメージのサポートを提供します。



重要

オンデマンドインスタンスは、Red Hat Cloud Access インスタンスに変換できません。オンデマンドイメージから Red Hat Cloud Access bring-your-own subscription (BYOS) イメージに変更するには、Red Hat Cloud Access インスタンスを新たに作成し、オンデマンドインスタンスからデータを移行します。データを移行した後に、オンデマンドのインスタンスをキャンセルして二重請求を回避します。

本章の残りの部分には、カスタムイメージに関する情報および手順が記載されています。

関連情報

- Red Hat in the Public Cloud
- イメージ
- Red Hat Cloud Access Reference Guide
- Creating an instance from a custom image

5.2. ベースイメージの理解

本セクションでは、事前設定されたベースイメージおよびその設定を使用する方法を説明します。

5.2.1. カスタムベースイメージの使用

仮想マシンを手動で設定するには、ベース (スタートアップ) 仮想マシンイメージで開始します。ベース仮想マシンイメージを作成したら、設定設定を変更して、仮想マシンがクラウドで動作するために必要なパッケージを追加できます。イメージのアップロード後に、特定のアプリケーションに追加の設定変更を行うことができます。

関連情報

Red Hat Enterprise Linux

5.2.2. 仮想マシン設定設定

クラウド仮想マシンには、以下の設定設定が必要です。

表5.2 仮想マシンの設定設定

設定	推奨事項
ssh	仮想マシンへのリモートアクセスを確立するには、**ssh** を有効にする必要があります。
dhcp	プライマリー仮想アダプターは、dhcp 用に設定する必要があります。

5.3. ISO イメージからのベース仮想マシンの作成

このセクションの手順に従って、ISO イメージからベースイメージを作成します。

前提条件

仮想化の 導入および管理ガイド に従って、Red Hat Enterprise Linux 7 ホストマシンの仮想化を有効にします。

5.3.1. ISO イメージのダウンロード

手順

1. Red Hat カスタマーポータル から最新の Red Hat Enterprise Linux ISO Image をダウンロードします。
2. イメージを **/var/lib/libvirt/images** ディレクトリーに移動します。

5.3.2. ISO イメージからの仮想マシンの作成

手順

1. 仮想化用のホストマシンを有効にしていることを確認します。仮想化パッケージのインストールに関する情報および手順は、既存の Red Hat Enterprise Linux システム上への仮想化パッケージのインストール を参照してください。
2. 基本的な Red Hat Enterprise Linux 仮想マシンを作成し、起動します。仮想マシンの作成手順は、仮想マシンの作成 を参照してください。

a. コマンドラインを使用して仮想マシンを作成する場合は、デフォルトのメモリーと CPU を仮想マシンの容量に設定するようにしてください。仮想ネットワークインターフェイスを **virtio** に設定します。
基本的なコマンドラインの例を以下に示します。

```
virt-install --name _vmname_ --memory 2048 --vcpus 2 --disk size=8,bus=virtio --location rhel-7.0-x86_64-dvd.iso --os-variant=rhel7.0
```

b. virt-manager アプリケーションを使用して仮想マシンを作成する場合は、virt-manager を使用したゲストの作成 の手順を実行します。以下の点に注意してください。

- **仮想マシンをすぐに起動** は選択しないでください。
- **メモリー** と **ストレージのサイズ** を、希望の設定に変更します。
- インストールを開始する前に、**仮想ネットワークインターフェイス設定** で **モデル** を **virtio** に変更し、**vCPUs** を仮想マシンの容量設定に変更していることを確認します。

5.3.3. RHEL インストールの完了

仮想マシンが起動したら、以下の手順を実行してインストールを完了し、root アクセスを有効にします。

手順

1. インストールプロセス中に使用する言語を選択します。
2. **インストール概要** ビューで、以下を行います。
   a. **ソフトウェアの選択** をクリックし、**最小インストール** を選択します。
   b. **完了** をクリックします。
   c. **インストール先** をクリックし、**ストレージ設定** で **カスタム** を選択します。
      - **/boot** で、500 MB 以上であることを確認してください。残りの領域は、root / に使用できます。
      - 標準のパーティションが推奨されますが、論理ボリューム管理 (LVM) を使用することも可能です。
      - ファイルシステムには、xfs、ext4、ext3 などを使用できます。
      - 変更が完了したら、**完了** をクリックします。
3. **インストールの開始** をクリックします。
4. **Root パスワード** を設定します。
5. インストールが完了したら、仮想マシンを再起動して **root** でログインします。
6. イメージを設定します。

注記

cloud-init パッケージがインストールされ、有効になっていることを確認します。

7. 仮想マシンの電源をオフにします。

5.4. RHEL イメージの GCP へのアップロード

ホストマシンで本セクションにある手順に従って、イメージを GCP にアップロードします。

5.4.1. GCP での新規プロジェクトの作成

GCP で新規プロジェクトを作成するには、以下の手順を行います。

前提条件

GCP でアカウントを作成している。作成していない場合は、Google Cloud で詳細を参照してください。

手順

1. GCP コンソール を起動します。
2. **Google Cloud Platform** の右側にあるドロップダウンメニューをクリックします。
3. ポップアップメニューから **新しいプロジェクト** をクリックします。
4. **新しいプロジェクト** ウィンドウに、新規プロジェクトの名前を入力します。
5. **組織** を確認します。ドロップダウンメニューをクリックして、必要に応じて組織を変更します。
6. 親組織またはフォルダーの **場所** を確認します。**参照** をクリックして検索し、必要に応じてこの値を変更します。
7. **作成** をクリックして、新しい GCP プロジェクトを作成します。

注記

Google Cloud SDK をインストールしたら、CLI コマンドの **gcloud projects create** を使用してプロジェクトを作成できます。簡単な例を以下に示します。

```
gcloud projects create my-gcp-project3 --name project3
```

この例では、プロジェクト ID **my-gcp-project3** とプロジェクト名 **project3** のプロジェクトを作成します。詳細は、gcloud project create を参照してください。

関連情報

リソースの作成と管理

5.4.2. Google Cloud SDK のインストール

Google Cloud SDK をインストールするには、以下の手順を行います。

前提条件

- GCP でプロジェクトがない場合は、これを作成します。詳細は Google Cloud Platform でプロジェクトの新規作成 を参照してください。
- ホストシステムに Python 2.7 以降が含まれることを確認します。そうでない場合は、Python 2.7 をインストールします。

手順

1. GCP の説明に従って、Google Cloud SDK アーカイブをダウンロードし、抽出します。詳細は、GCP ドキュメント Quickstart for Linux を参照してください。
2. Google Cloud SDK の初期化と同じ手順に従います。

注記

Google Cloud SDK を初期化すると、**gcloud** CLI コマンドを使用してタスクを実行でき、プロジェクトとインスタンスに関する情報を取得できます。たとえば、**gcloud compute project-info describe --project <project-name>** コマンドを使用してプロジェクト情報を表示できます。

関連情報

- Linux 用のクイックスタート
- gcloud command reference
- gcloud コマンドライン ツールの概要

5.4.3. Google Compute Engine の SSH 鍵の作成

以下の手順に従って、GCE で SSH 鍵を生成して登録し、そのパブリック IP アドレスを使用してインスタンスに直接 SSH 接続できるようにします。

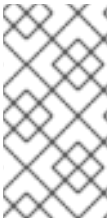
手順

1. **ssh-keygen** コマンドを使用して、GCE で使用する SSH 鍵ペアを生成します。

```
# ssh-keygen -t rsa -f ~/.ssh/google_compute_engine
```

2. GCP Console Dashboard ページ から、Google の **Cloud Console バナー** の左側にある **ナビゲーション** メニューをクリックし、**Compute Engine** を選択して **Metadata** を選択します。
3. **SSH 鍵** をクリックして、**編集** をクリックします。
4. **~/.ssh/google_compute_engine.pub** ファイルから生成された出力を入力し、**保存** をクリックします。
これで、標準の SSH を使用してインスタンスに接続できます。

```
# ssh -i ~/.ssh/google_compute_engine <username>@<instance_external_ip>
```



注記

gcloud compute config-ssh コマンドを実行すると、インスタンスのエイリアスを設定ファイルに追加できます。エイリアスは、インスタンス名による単純な SSH 接続を許可します。**gcloud compute config-ssh** コマンドの詳細は、gcloud compute config-ssh を参照してください。

関連情報

- gcloud compute config-ssh
- インスタンスへの接続

5.4.4. GCP ストレージでのストレージバケットの作成

GCP にインポートするには、GCP Storage バケットが必要です。以下の手順に従って、バケットを作成します。

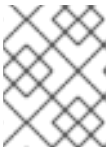
手順

1. GCP にログインしていない場合は、次のコマンドを実行してログインします。

```
# gcloud auth login
```

2. ストレージバケットを作成します。

```
# gsutil mb gs://bucket_name
```



注記

Google Cloud コンソールを使用してバケットを作成することもできます。詳細は、バケットの作成 を参照してください。

関連情報

バケットの作成

5.4.5. GCP バケットへのイメージの変換およびアップロード

以下の手順に従って、GCP バケットにイメージを変換してアップロードします。サンプルは典型的なものです。**qcow2** イメージを **raw** 形式に変換してから、そのイメージをアップロードするために tar を使用します。

手順

1. **qemu-img** コマンドを実行してイメージを変換します。変換されたイメージは、**disk.raw** という名前になるはずです。

```
# qemu-img convert -f qcow2 -O raw rhel-sample.qcow2 disk.raw
```

2. イメージに tar コマンドを実行します。

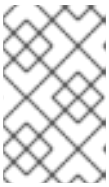
```
# tar --format=oldgnu -Sczf disk.raw.tar.gz disk.raw
```

3. そのイメージを、以前作成したバケットにアップロードします。アップロードには数分かかる場合があります。

```
# gsutil cp disk.raw.tar.gz gs://bucket_name
```

検証手順

1. **Google Cloud Platform** のホーム画面で、折りたたまれたメニューアイコンをクリックし、**ストレージ** を選択してから **ブラウザー** を選択します。
2. バケットの名前をクリックします。
バケット名の下に、tar コマンドを実行したイメージが表示されます。



注記

GCP コンソール を使用してイメージをアップロードすることもできます。これを行うには、バケットの名前をクリックしてから **ファイルのアップロード** をクリックします。

関連情報

- 仮想ディスクの手動インポート
- インポート方法の選択

5.4.6. GCP バケットのオブジェクトからイメージの作成

以下の手順に従って、GCP バケットのオブジェクトからイメージを作成します。

手順

- 以下のコマンドを実行して GCE のイメージを作成します。作成するイメージの名前、バケット名、および tar コマンドを実行したイメージの名前を指定します。

```
# gcloud compute images create my-image-name --source-uri gs://my-bucket-
name/disk.raw.tar.gz
```

注記

または、Google Cloud コンソールを使用して、イメージを作成することもできます。詳細は、カスタムイメージの作成、削除、および中止 を参照してください。

- 必要に応じて、GCP コンソールでイメージを検索します。
  a. **Google Cloud Console** バナーの左側にある **ナビゲーション** メニューをクリックします。
  b. **Compute Engine** を選択し、**イメージ** を選択します。

関連情報

- カスタム イメージの作成、削除、使用中止

- gcloud compute images create

5.4.7. イメージからの Google Compute Engine インスタンスの作成

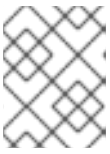
GCP コンソールを使用して GCE 仮想マシンインスタンスを設定するには、以下の手順を行います。

注記

以下の手順では、GCP コンソールを使用して基本的な仮想マシンインスタンスを作成する方法を説明します。GCE 仮想マシンインスタンスおよびその設定オプションの詳細は、VM インスタンスの作成と起動 を参照してください。

手順

1. GCP Console Dashboard ページ から、Google の **Cloud Console バナー** の左側にある **ナビゲーション** メニューをクリックし、**Compute Engine** を選択して **イメージ** を選択します。
2. イメージを選択します。
3. **インスタンスの作成** をクリックします。
4. **インスタンスの作成** ページで、インスタンスの **名前** を入力します。
5. **地域** と **ゾーン** を選択します。
6. ワークロードの要件を満たす、もしくは超過する **マシン設定** を選択します。
7. **ブートディスク** にイメージ名が指定されていることを確認します。
8. 必要に応じて、**ファイアウォール** で **HTTP トラフィックを許可する** か、**HTTPS トラフィックを許可する** を選択します。
9. **作成** をクリックします。



注記

これらは、基本インスタンスの作成に必要な最小限の設定オプションです。アプリケーション要件に応じて追加オプションを確認します。

10. **VM インスタンス** にあるイメージを見つけます。
11. GCP Console Dashboard から、Google の **Cloud Console バナー** の左側にある **ナビゲーション** メニューをクリックし、**Compute Engine** を選択してから **仮想マシンインスタンス** を選択します。

注記

または、CLI コマンド **gcloud compute instances create** を使用して、イメージから GCE 仮想マシンインスタンスを作成することもできます。簡単な例を以下に示します。

```
gcloud compute instances create myinstance3 --zone=us-central1-a --image
test-iso2-image
```

この例では、既存のイメージ **test-iso2-image** に基づいて、ゾーン **us-central1-a** に **myinstance3** という名前の仮想マシンインスタンスを作成します。詳細は、gcloud compute instances create を参照してください。

5.4.8. インスタンスへの接続

以下の手順に従って、そのパブリック IP アドレスを使用して GCE インスタンスに接続します。

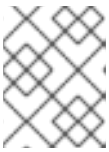
手順

1. 次のコマンドを実行して、インスタンスが実行されていることを確認します。このコマンドは、インスタンスが実行しているかどうかを示し、実行している場合は実行中のインスタンスのパブリック IP アドレスなど、GCE インスタンスに関する情報一覧を表示します。

```
# gcloud compute instances list
```

2. 標準の SSH を使用してインスタンスに接続します。この例では、上述の手順で作成した **google_compute_engine** キーを使用します。

```
# ssh -i ~/.ssh/google_compute_engine <user_name>@<instance_external_ip>
```



注記

GCP は、インスタンスに対して SSH を多数実行する方法を提供します。詳細は、Connecting to instances を参照してください。

関連情報

- gcloud compute instances list
- インスタンスへの接続

5.4.9. Red Hat サブスクリプションの割り当て

Red Hat Cloud Access プログラムで有効になっているサブスクリプションを割り当てるには、以下の手順を行います。

前提条件

サブスクリプションが有効になっている。

手順

1. システムを登録します。

```
subscription-manager register --auto-attach
```

2. サブスクリプションを割り当てます。

   - アクティベーションキーを使用して、サブスクリプションを割り当てることができます。カスタマーポータルのアクティベーションキーを作成する を参照してください。
   - または、サブスクリプションプール (Pool ID) の ID を使用してサブスクリプションを手動で割り当てることができます。コマンドラインでのサブスクリプションのアタッチと削除を参照してください。

関連情報

- カスタマーポータルのアクティベーションキーを作成する
- コマンドラインでのサブスクリプションのアタッチと削除
- Red Hat Subscription Manager の使用および設定

第6章 GOOGLE CLOUD PLATFORM での RED HAT HIGH AVAILABILITY クラスターの設定

本章では、Google Compute Engine (GCE) 仮想マシンインスタンスをクラスターノードとして使用して、Google Cloud Platform (GCP) に Red Hat High Availability (HA) クラスターを設定する情報および手順を説明します。

本章には、GCP の環境を設定するための前提条件の手順が含まれています。環境を設定したら、GCP 仮想マシンインスタンスを作成および設定できます。

本章では、個別のノードを GCP の HA ノードのクラスターに変換する HA クラスターの作成に固有の手順も説明します。これには、各クラスターノードに高可用性パッケージおよびエージェントをインストールし、フェンシングを設定し、GCP ネットワークリソースエージェントをインストールする手順が含まれます。

この章では、GCP のドキュメントを参照している箇所が多数あります。詳細は、記載の GCP ドキュメントを参照してください。

前提条件

- GCP ソフトウェア開発きっと (SDK) をインストールする必要があります。詳細は、Google クラウド SDK のインストール を参照してください。
- Red Hat Cloud Access プログラム でサブスクリプションを有効にします。Red Hat Cloud Access プログラムでは、Red Hat のサブスクリプションを、物理システムまたはオンプレミスシステムから、Red Hat のフルサポートのある GCP へ移動できます。
- アクティブな GCP プロジェクトに属し、プロジェクトでリソースを作成するのに十分なパーミッションを持っている必要があります。
- プロジェクトには、個別ユーザーではなく、仮想マシンインスタンスに属する サービスアカウント が必要です。別のサービスアカウントを作成する代わりに、デフォルトのサービスアカウントを使用する方法は、Using the Compute Engine Default Service Account を参照してください。

またはプロジェクト管理者がカスタムサービスアカウントを作成する場合、サービスアカウントは以下のロールに設定する必要があります。

- クラウドトレースエージェント
- コンピュート管理者
- コンピュートネットワーク管理者
- クラウドデータベースユーザー
- ロギング管理者
- 監視エディター
- 監視メトリックライター
- サービスアカウント管理者
- ストレージ管理者

関連情報

- Support Policies for RHEL High Availability Clusters - Google Cloud Platform Virtual Machines as Cluster Members
- Support Policies for RHEL High Availability clusters - Transport Protocols
- VPC network overview
- Exploring RHEL High Availability's Components, Concepts, and Features - Overview of Transport Protocols
- Design Guidance for RHEL High Availability Clusters - Selecting the Transport Protocol
- Quickstart for Red Hat and Centos

6.1. GCP での RED HAT ENTERPRISE LINUX イメージオプション

以下の表には、イメージの選択肢を記載し、イメージオプションの相違点を示しています。

表6.1 イメージオプション

イメージオプション	サブスクリプション	サンプルシナリオ	留意事項
GCP に移動するカスタムイメージをデプロイすることを選択する	既存の Red Hat サブスクリプションを活用する	Red Hat Cloud Access プログラム を使用してサブスクリプションを有効にし、カスタムイメージをアップロードし、サブスクリプションを割り当てます。	その他のすべてのインスタンスコストを支払うことになりますが、サブスクリプション自体には Red Hat 製品コストが含まれます。 GCP に移行するカスタムイメージは、既存の Red Hat サブスクリプションを活用するため、クラウドアクセスイメージと呼ばれています。Red Hat は、クラウドアクセスイメージを直接サポートします。
RHEL を含む既存の GCP イメージをデプロイすることを選択する	GCP イメージには、Red Hat 製品が含まれる	GCP Compute Engine でインスタンスを起動する時に RHEL イメージを選択するか、Google Cloud Platform Marketplace からイメージを選択します。	従量課金モデルでは、GCP に 1 時間ごとに支払います。このようなイメージはオンデマンドイメージと呼ばれています。GCP は、サポート契約に基づいてオンデマンドイメージのサポートを提供します。

重要

オンデマンドインスタンスは、Red Hat Cloud Access インスタンスに変換できません。オンデマンドイメージから Red Hat Cloud Access bring-your-own subscription (BYOS) イメージに変更するには、Red Hat Cloud Access インスタンスを新たに作成し、オンデマンドインスタンスからデータを移行します。データを移行した後に、オンデマンドのインスタンスをキャンセルして二重請求を回避します。

本章の残りの部分には、カスタムイメージに関する情報および手順が記載されています。

関連情報

- Red Hat in the Public Cloud
- イメージ
- Red Hat Cloud Access Reference Guide
- Creating an instance from a custom image

6.2. 必要なシステムパッケージ

本章の手順では、Red Hat Enterprise Linux を実行しているホストシステムを使用していることを前提としています。この手順を正常に行うには、ホストシステムに以下のパッケージをインストールする必要があります。

表6.2 システムパッケージ

パッケージ	詳細	コマンド
qemu-kvm	このパッケージは、ユーザーレベルの KVM エミュレーターを使用し、ホストとゲスト仮想マシン間の通信を容易にします。	**# yum install qemu-kvm libvirt**
qemu-img	このパッケージは、ゲスト仮想マシンのディスク管理を提供します。qemu-img パッケージは qemu-kvm パッケージの依存関係としてインストールされます。	
libvirt	このパッケージは、ハイパーバイザーおよびホストシステムと対話するためにサーバーおよびホスト側のライブラリーを提供し、ライブラリー呼び出しの処理、仮想マシンの管理およびハイパーバイザーの制御を行う libvirtd デーモンも提供します。	

表6.3 その他の仮想化パッケージ

パッケージ	詳細	コマンド
virt-install	このパッケージは、コマンドラインからの仮想マシンを作成するために **virt-install** コマンドを提供します。	**# yum install virt-install libvirt-python virt-manager virt-install libvirt-client**
libvirt-python	このパッケージは、Python プログラミング言語で書かれているアプリケーションが libvirt API で提供されるインターフェイスを使用できるようにするモジュールが含まれています。	
virt-manager	このパッケージには、仮想マシンマネージャー (VMM) と呼ばれる virt-manager ツールが含まれます。VMM は、仮想マシンを管理するためのグラフィカルツールです。管理 API として libvirt-client ライブラリーを使用します。	
libvirt-client	このパッケージは、libvirt サーバーにアクセスするためのクライアント側の API とライブラリーを提供します。libvirt-client パッケージには、コマンドラインまたは特別な仮想化シェルから仮想マシンとハイパーバイザーを管理および制御する **virsh** コマンドラインツールが含まれます。	

関連情報

- 仮想化パッケージの手動インストール

6.3. HA パッケージおよびエージェントのインストール

全ノードで以下の手順を実行し、High Availability パッケージおよびエージェントをインストールします。

手順

1. すべてのリポジトリーを無効にします。

```
# subscription-manager repos --disable=*
```

2. RHEL 7 Server リポジトリーおよび RHEL 7 Server HA リポジトリーを有効にします。

```
# subscription-manager repos --enable=rhel-7-server-rpms
# subscription-manager repos --enable=rhel-ha-for-rhel-7-server-rpms
```

3. すべてのパッケージを更新します。

```
# yum update -y
```

4. **pcs pacemaker**、fence agent およびリソースエージェントをインストールします。

```
# yum install -y pcs pacemaker fence-agents-gce resource-agents-gcp
```

5. カーネルを更新したら、マシンを再起動します。

```
# reboot
```

6.4. HA サービスの設定

高可用性サービスを設定するには、すべてのノードで以下の手順を実行します。

手順

1. **hacluster** ユーザーは、前の手順で **pcs** および **pacemaker** のインストール時に作成されました。すべてのクラスターノードに **hacluster** ユーザーのパスワードを作成します。すべてのノードに同じパスワードを使用します。

```
# passwd hacluster
```

2. **firewalld** サービスが有効になっている場合は、RHEL に高可用性サービスを追加します。

```
# firewall-cmd --permanent --add-service=high-availability

# firewall-cmd --reload
```

3. **pcs** サービスを起動し、システムの起動時に開始できるようにします。

```
# systemctl enable pcsd.service --now
```

検証手順

1. **pcs** サービスが実行していることを確認します。

```
# systemctl is-active pcsd.service
```

6.5. クラスターの作成

ノードのクラスターを作成するには、以下の手順を実施します。

手順

1. ノードのいずれかで以下のコマンドを実行し、pcs ユーザー **ha cluster** を認証します。クラスターの各ノードの名前を指定します。

```
# pcs cluster auth  _hostname1_ _hostname2_ _hostname3_ -u hacluster
```

たとえば、以下のようになります。

```
[root@node01 ~]# pcs cluster auth node01 node02 node03 -u hacluster
node01: Authorized
node02: Authorized
node03: Authorized
```

2. クラスターを作成します。

```
# pcs cluster setup --name cluster-name _hostname1_ _hostname2_  _hostname3_
```

検証手順

1. クラスターを有効にします。

```
# pcs cluster enable --all
```

2. クラスターを起動します。

```
# pcs cluster start --all
```

6.6. フェンスデバイスの作成

ほとんどのデフォルト設定では、GCP インスタンス名と RHEL ホスト名は同じです。

クラスター内の任意のノードからフェンシングを設定するには、以下の手順を実施します。

手順

1. クラスターの任意のノードから GCP インスタンス名を取得します。出力には、インスタンスの内部 ID も表示されることに注意してください。

```
# fence_gce --zone _gcp_ _region_ --project= _gcp_ _project_ -o list
```

たとえば、以下のようになります。

```
[root@rhel71-node-01 ~]# fence_gce --zone us-west1-b --project=rhel-ha-testing-on-gcp -o
list
44358**********3181,InstanceName-3
40819**********6811,InstanceName-1
71736**********3341,InstanceName-2
```

2. フェンスデバイスを作成します。**pcmk_host-name** コマンドを使用して、RHEL ホスト名をインスタンス ID にマッピングします。

```
# pcs stonith create _clusterfence_ fence_gce pcmk_host_map=_pcmk-hpst-map_
fence_gce zone=_gcp-zone_ project=_gcpproject_
```

たとえば、以下のようになります。

```
[root@node01 ~]# pcs stonith create fencegce fence_gce
pcmk_host_map="node01:node01-vm;node02:node02-vm;node03:node03-vm"
project=hacluster zone=us-east1-b
```

検証手順

1. 他のノードのいずれかに対してフェンスエージェントをテストします。

```
# pcs stonith fence gcp nodename
```

2. ステータスを確認して、ノードがフェンスされていることを確認します。

```
# watch pcs status
```

たとえば、以下のようになります。

```
[root@node01 ~]# watch pcs status
Cluster name: gcp-cluster
Stack: corosync
Current DC: rhel71-node-02 (version 1.1.18-11.el7_5.3-2b07d5c5a9) - partition with quorum
Last updated: Fri Jul 27 12:53:25 2018
Last change: Fri Jul 27 12:51:43 2018 by root via cibadmin on rhel71-node-01

3 nodes configured
3 resources configured

Online: [ rhel71-node-01 rhel71-node-02 rhel71-node-03 ]

Full list of resources:

us-east1-b-fence    (stonith:fence_gce):    Started rhel71-node-01

Daemon Status:
corosync: active/enabled
pacemaker: active/enabled
pcsd: active/enabled
```

6.7. GCP ノードの承認の設定

アカウント認証情報を使用して GCP にアクセスするようにクラウドの SDK ツールを設定します。

手順

各ノードで次のコマンドを実行し、プロジェクト ID およびアカウントの認証情報で各ノードを初期化します。

```
# gcloud-ra init
```

6.8. GCP ネットワークリソースエージェントの設定

クラスターは、実行中のインスタンスにセカンダリー IP アドレス (エイリアス IP) を 割り当てた GCP ネットワークリソースエージェントを使用します。これは、クラスター内の異なるノード間で渡すことのできるフローティング IP アドレスです。

手順

以下のコマンドを入力して、GCP 仮想 IP アドレスリソースエージェント (gcp-vpc-move-vip) の説明を表示します。これは、このエージェントのオプションとデフォルトの操作を示しています。

```
# pcs resource describe gcp-vpc-move-vip
```

リソースエージェントを、プライマリーサブネットアドレス範囲またはセカンダリーサブネットアドレス範囲を使用するように設定できます。本セクションでは、両方の手順を説明します。

プライマリーサブネットアドレス範囲

手順

プライマリー VPC サブネットのリソースを設定するには、以下の手順を実行します。

1. **aliasip** リソースを作成します。未使用の内部 IP アドレスを含めます。コマンドに CIDR ブロックを含めます。

   ```
   # pcs resource create aliasip gcp-vpc-move-vip  alias_ip=_UnusedIPaddress/CIDRblock_ --group _group-name_ --group _networking-group_
   ```

2. ノード上の IP を管理する **IPaddr2** リソースを作成します。

   ```
   # pcs resource create vip IPaddr2 nic=_interface_ ip=_AliasIPaddress_ cidr_netmask=32 --group _group-name_ --group _networking-group_
   ```

3. **vipgrp** でネットワークリソースをグループ化します。

   ```
   # pcs resource group add vipgrp aliasip vip
   ```

検証手順

1. リソースが起動し、**vipgrp** でグループ化されていることを確認します。

   ```
   # pcs status
   ```

2. リソースが別のノードに移動できることを確認します。

   ```
   # pcs resource move vip _Node_
   ```

   たとえば、以下のようになります。

   ```
   # pcs resource move vip rhel71-node-03
   ```

3. 別のノードで **vip** が正常に起動することを確認します。

   ```
   # pcs status
   ```

セカンダリーサブネットアドレス範囲

セカンダリーサブネットアドレス範囲のリソースを設定するには、以下の手順を実施します。

前提条件

カスタムネットワークおよびサブネットを作成している。

手順

1. セカンダリーサブネットアドレス範囲を作成します。

```
# gcloud-ra compute networks subnets update _SubnetName_ --region _RegionName_ --add-secondary-ranges _SecondarySubnetName_=_SecondarySubnetRange_
```

   たとえば、以下のようになります。

```
# gcloud-ra compute networks subnets update range0 --region us-west1 --add-secondary-ranges range1=10.10.20.0/24
```

2. **aliasip** リソースを作成します。セカンダリーサブネットアドレス範囲に未使用の内部 IP アドレスを作成します。コマンドに CIDR ブロックを含めます。

```
# pcs resource create aliasip gcp-vpc-move-vip alias_ip=_UnusedIPaddress/CIDRblock_ --group _group-name_ --group _networking-group_
```

3. ノード上の IP を管理する **IPaddr2** リソースを作成します。

```
# pcs resource create vip IPaddr2 nic=_interface_ ip=_AliasIPaddress_ cidr_netmask=32 --group _group-name_ --group _networking-group_
```

検証手順

1. リソースが起動し、**vipgrp** でグループ化されていることを確認します。

```
# pcs status
```

2. リソースが別のノードに移動できることを確認します。

```
# pcs resource move vip _Node_
```

   たとえば、以下のようになります。

```
[root@rhel71-node-01 ~]# pcs resource move vip rhel71-node-03
```

3. 別のノードで **vip** が正常に起動することを確認します。

```
# pcs status
```