Red Hat Enterprise Linux 9

Microsoft Azure への RHEL 9 のデプロイ

RHEL システムイメージの取得と Azure 上での RHEL インスタンスの作成

Last Updated: 2025-11-27

Red Hat Enterprise Linux 9 Microsoft Azure への RHEL 9 のデプロイ

RHEL システムイメージの取得と Azure 上での RHEL インスタンスの作成

Legal Notice

Abstract

Red Hat Enterprise Linux (RHEL) をパブリッククラウド環境で使用するには、RHEL システムイメージを作成し、Microsoft Azure 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。Azure 上で Red Hat High Availability (HA) クラスターを作成および設定することもできます。次の章では、Azure 上でクラウド RHEL インスタンスと HA クラスターを作成する手順を説明します。これらのプロセスには、必要なパッケージとエージェントのインストール、フェンシングの設定、ネットワークリソースエージェントのインストールが含まれます。

Table of Contents

RED HAT ドキュメントへのフィードバック (英語のみ)

Red Hat ドキュメントに関するご意見やご感想をお寄せください。また、改善点があればお知らせください。

Jira からのフィードバック送信 (アカウントが必要)

1. Jira の Web サイトにログインします。
2. 上部のナビゲーションバーで **Create** をクリックします。
3. **Summary** フィールドにわかりやすいタイトルを入力します。
4. **Description** フィールドに、ドキュメントの改善に関するご意見を記入してください。ドキュメントの該当部分へのリンクも追加してください。
5. ダイアログの下部にある **Create** をクリックします。

第1章 パブリッククラウドプラットフォームでの RHEL の導入

パブリッククラウドプラットフォームは、コンピューティングリソースをサービスとして提供します。オンプレミスのハードウェアを使用する代わりに、Red Hat Enterprise Linux (RHEL) システムなどの IT ワークロードをパブリッククラウドインスタンスとして実行できます。

1.1. パブリッククラウドで RHEL を使用する利点

パブリッククラウドプラットフォーム上に配置されたクラウドインスタンスとしての RHEL には、RHEL オンプレミスの物理システムまたは仮想マシン (VM) に比べて次の利点があります。

- **リソースの柔軟性と詳細な割り当て**
  RHEL のクラウドインスタンスは、クラウドプラットフォーム上の仮想マシンとして実行されます。この仮想マシンは通常、クラウドサービスのプロバイダーによって維持管理されるリモートサーバーのクラスターです。したがって、特定のタイプの CPU やストレージなどのハードウェアリソースのインスタンスへの割り当ては、ソフトウェアレベルで行われ、簡単にカスタマイズできます。

  また、ローカルの RHEL システムと比較すると、物理ホストの機能によって制限されることがありません。むしろ、クラウドプロバイダーが提供する選択肢に基づいて、さまざまな機能から選択できます。

- **領域とコスト効率**
  クラウドワークロードをホストするためにオンプレミスサーバーを所有する必要がありません。これにより、物理ハードウェアに関連するスペース、電力、メンテナンスの要件が回避されます。

  代わりに、パブリッククラウドプラットフォームでは、クラウドインスタンスの使用料をクラウドプロバイダーに直接支払います。通常、コストはインスタンスに割り当てられたハードウェアとその使用時間に基づきます。したがって、要件に基づいてコストを最適化できます。

- **ソフトウェアで制御される設定**
  クラウドインスタンスの設定全体がクラウドプラットフォーム上にデータとして保存され、ソフトウェアによって制御されます。したがって、インスタンスの作成、削除、クローン作成、または移行を簡単に行うことができます。また、クラウドインスタンスは、クラウドプロバイダーのコンソールでリモートで操作され、デフォルトでリモートストレージに接続されます。

  さらに、クラウドインスタンスの現在の状態をいつでもスナップショットとしてバックアップできます。その後、スナップショットをロードしてインスタンスを保存した状態に復元できます。

- **ホストからの分離とソフトウェアの互換性**
  ローカルの仮想マシンと同様に、クラウドインスタンス上の RHEL ゲストオペレーティングシステムは仮想化されたカーネル上で実行されます。このカーネルは、ホストオペレーティングシステムや、インスタンスへの接続に使用する **クライアント** システムとは別のものです。

  したがって、任意のオペレーティングシステムをクラウドインスタンスにインストールできます。つまり、RHEL パブリッククラウドインスタンスでは、ローカルオペレーティングシステムでは使用できない RHEL 固有のアプリケーションを実行できます。

  さらに、インスタンスのオペレーティングシステムが不安定になったり侵害されたりした場合でも、クライアントシステムには一切影響がありません。

関連情報

- パブリッククラウドとは
- ハイパースケーラーとは
- クラウドコンピューティングの種類
- RHEL のパブリッククラウドのユースケース
- パブリッククラウドデプロイメント用の RHEL の入手

1.2. RHEL のパブリッククラウドのユースケース

パブリッククラウドへのデプロイには多くの利点がありますが、すべてのシナリオにおいて最も効率的なソリューションであるとは限りません。RHEL デプロイメントをパブリッククラウドに移行するかどうかを評価している場合は、ユースケースがパブリッククラウドの利点を享受できるかどうかを検討してください。

有益なユースケース

- パブリッククラウドインスタンスのデプロイは、デプロイメントのアクティブなコンピューティング能力を柔軟に増減する (**スケールアップ** および **スケールダウン** とも呼ばれます) 場合に非常に効果的です。したがって、次のシナリオではパブリッククラウドで RHEL を使用することを推奨します。
  - ピーク時のワークロードが高く、一般的なパフォーマンス要件が低いクラスター。要求に応じてスケールアップおよびスケールダウンすることで、リソースコストの面で高い効率が得られる場合があります。
  - クラスターを迅速にセットアップまたは拡張できます。これにより、ローカルサーバーのセットアップにかかる高額な初期費用が回避されます。
- クラウドインスタンスは、ローカル環境で何が起こっても影響を受けません。したがって、バックアップや障害復旧に使用できます。

問題が発生する可能性のあるユースケース

- 調整不可能な既存の環境を運用している場合。既存のデプロイメントの特定のニーズに合わせてクラウドインスタンスをカスタマイズすることは、現在のホストプラットフォームと比較して費用対効果が低い可能性があります。
- 厳しい予算制限の中で運用している場合。通常、ローカルデータセンターでデプロイメントを維持管理すると、パブリッククラウドよりも柔軟性は低くなりますが、最大リソースコストをより細かく制御できます。

次のステップ

- パブリッククラウドデプロイメント用の RHEL の入手

関連情報

- アプリケーションをクラウドに移行すべきか?また、その決定方法

1.3. パブリッククラウドへの移行時によくある懸念事項

RHEL ワークロードをローカル環境からパブリッククラウドプラットフォームに移行すると、それに伴う変更に懸念が生じる可能性があります。よくある質問は次のとおりです。

RHEL は、クラウドインスタンスとして動作する場合、ローカル仮想マシンとして動作する場合とは異なる動作になりますか?

パブリッククラウドプラットフォーム上の RHEL インスタンスは、ほとんどの点で、オンプレミスサーバーなどのローカルホスト上の RHEL 仮想マシンと同じように機能します。注目すべき例外には次のようなものがあります。

- パブリッククラウドインスタンスは、プライベートオーケストレーションインターフェイスの代わりに、プロバイダー固有のコンソールインターフェイスを使用してクラウドリソースを管理します。
- ネストされた仮想化などの特定の機能が正しく動作しない可能性があります。特定の機能がデプロイメントにとって重要な場合は、選択したパブリッククラウドプロバイダーとその機能の互換性を事前に確認してください。

ローカルサーバーと比べて、パブリッククラウドではデータは安全に保たれますか?

RHEL クラウドインスタンス内のデータの所有権はユーザーにあり、パブリッククラウドプロバイダーはデータにアクセスできません。さらに、主要なクラウドプロバイダーは転送中のデータ暗号化をサポートしているため、仮想マシンをパブリッククラウドに移行する際のデータのセキュリティーが向上します。

RHEL パブリッククラウドインスタンスの一般的なセキュリティーは次のように管理されます。

- パブリッククラウドプロバイダーは、クラウドハイパーバイザーのセキュリティーを担当します。
- Red Hat は、RHEL ゲストオペレーティングシステムのセキュリティー機能をインスタンスに提供します。
- ユーザーは、クラウドインフラストラクチャーにおける特定のセキュリティー設定とプラクティスを管理します。

ユーザーの地理的リージョンは、RHEL パブリッククラウドインスタンスの機能にどのように影響しますか?

RHEL インスタンスは、地理的な場所に関係なく、パブリッククラウドプラットフォームで使用できます。したがって、オンプレミスサーバーと同じリージョンでインスタンスを実行できます。

ただし、物理的に離れたリージョンでインスタンスをホストすると、操作時に待ち時間が長くなる可能性があります。さらに、パブリッククラウドプロバイダーによっては、特定のリージョンで、追加機能が提供される場合や、より高いコスト効率が得られる場合があります。RHEL インスタンスを作成する前に、選択したクラウドプロバイダーで利用可能なホスティングリージョンのプロパティーを確認してください。

1.4. パブリッククラウドデプロイメント用の RHEL の入手

RHEL システムをパブリッククラウド環境にデプロイするには、次の手順を実行する必要があります。

1. 要件と市場の現在のオファーに基づいて、ユースケースに最適なクラウドプロバイダーを選択します。
   現在、RHEL インスタンスの実行が認定されているクラウドプロバイダーは次のとおりです。
   - Amazon Web Services (AWS)
   - Google Cloud

- Microsoft Azure

注記

このドキュメントでは、Microsoft Azure への RHEL のデプロイについて特に説明します。

2. 選択したクラウドプラットフォーム上に RHEL クラウドインスタンスを作成します。詳細は、RHEL クラウドインスタンスを作成する方法 を参照してください。
3. RHEL デプロイメントを最新の状態に保つには、Red Hat Update Infrastructure (RHUI) を使用します。

関連情報

- RHUI ドキュメント
- Red Hat Open Hybrid Cloud

1.5. RHEL クラウドインスタンスを作成する方法

RHEL インスタンスをパブリッククラウドプラットフォームにデプロイするには、次のいずれかの方法を使用できます。

RHEL のシステムイメージを作成し、クラウドプラットフォームにインポートします。

- システムイメージを作成するには、RHEL Image Builder を使用するか、イメージを手動で構築します。
- これは既存の RHEL サブスクリプションを使用する方法で、**bring your own subscription** (BYOS) とも呼ばれます。
- 年間サブスクリプションを前払いすると、Red Hat お客様割引を利用できます。
- カスタマーサービスは Red Hat によって提供されます。
- 複数のイメージを効率的に作成するには、**cloud-init** ツールを使用できます。

RHEL インスタンスをクラウドプロバイダーマーケットプレイスから直接購入します。

- サービスの利用に対して時間料金を後払いで支払います。したがって、この方法は **従量課金制** (PAYG) とも呼ばれます。
- カスタマーサービスはクラウドプラットフォームプロバイダーによって提供されます。

注記

さまざまな方法を使用して Microsoft Azure に RHEL インスタンスをデプロイする詳細な手順は、このドキュメントの次の章を参照してください。

関連情報

- ゴールデンイメージとは

- RHEL 9 の cloud-init の設定および管理

第2章 VHD イメージを作成して MICROSOFT AZURE クラウドに自動的にアップロードする

RHEL Image Builder を使用して **.vhd** イメージを作成すると、Microsoft Azure クラウドサービスプロバイダーの Blob Storage に自動的にアップロードされます。

前提条件

- システムへの root アクセス権があります。
- RHEL Web コンソールの RHEL Image Builder インターフェイスにアクセスできる。
- ブループリントを作成している。Web コンソールインターフェイスでの RHEL Image Builder ブループリントの作成 を参照してください。
- Microsoft ストレージアカウント が作成されました。
- 書き込み可能な Blob Storage が準備されました。

手順

1. RHEL Image Builder ダッシュボードで、使用するブループリントを選択します。
2. **Images** タブをクリックします。
3. **Create Image** をクリックして、カスタマイズした **.vhd** イメージを作成します。
   Create image ウィザードが開きます。
   a. **Type** ドロップダウンメニューリストから **Microsoft Azure (.vhd)** を選択します。
   b. イメージを Microsoft Azure クラウドにアップロードするには、**Upload to Azure** チェックボックスをオンします。
   c. **Image Size** を入力し、**Next** をクリックします。
4. **Upload to Azure** ページで、次の情報を入力します。
   a. 認証ページで、次のように入力します。
      i. **Storage account** の名前。これは、Microsoft Azure portal の **Storage account** ページにあります。
      ii. **Storage access key**. これは、**Access Key** ストレージページにあります。
      iii. **Next** をクリックします。
   b. **Authentication** ページで、次のように入力します。
      i. イメージ名
      ii. **Storage container**。これは、イメージのアップロード先の Blob コンテナーです。Microsoft Azure portal の **Blob service** セクションにあります。
      iii. **Next** をクリックします。

5. **Review** ページで **Create** をクリックします。RHEL Image Builder が起動し、アップロードプロセスが開始します。
   Microsoft Azure Cloud にプッシュしたイメージにアクセスします。
6. Microsoft Azure portal にアクセスします。
7. 検索バーに "storage account" と入力し、リストから **Storage accounts** をクリックします。
8. 検索バーに "Images" と入力し、**Services** の下にある最初のエントリーを選択します。**Image dashboard** にリダイレクトされます。
9. ナビゲーションパネルで、**Containers** をクリックします。
10. 作成したコンテナーを見つけます。コンテナー内には、RHEL Image Builder を使用して作成およびプッシュした **.vhd** ファイルがあります。

検証

1. 仮想マシンイメージを作成して起動できることを確認します。
   a. 検索バーに images account と入力し、リストから **Images** をクリックします。
   b. **+Create** をクリックします。
   c. ドロップダウンリストから、前に使用したリソースグループを選択します。
   d. イメージの名前を入力します。
   e. **OS type** で **Linux** を選択します。
   f. **VM generation** で **Gen 2** を選択します。
   g. **Storage Blob** で **Browse** をクリックし、VHD ファイルに到達するまでストレージアカウントとコンテナーをクリックします。
   h. ページの最後にある **Select** をクリックします。
   i. Account Type を選択します (例: **Standard SSD)**。
   j. **Review + Create** をクリックし、**Create** をクリックします。イメージが作成されるまでしばらく待機します。
2. 仮想マシンを起動するには、次の手順に従います。
   a. **Go to resource** をクリックします。
   b. ヘッダーのメニューバーから **+ Create VM** をクリックします。
   c. 仮想マシンの名前を入力します。
   d. **Size** セクションと **Administrator account** セクションに入力します。
   e. **Review + Create** をクリックし、**Create** をクリックします。デプロイメントの進行状況を確認できます。
   デプロイメントが完了したら、仮想マシン名をクリックしてインスタンスのパブリック IP アドレスを取得し、SSH を使用して接続します。

f. ターミナルを開いて SSH 接続を作成し、仮想マシンに接続します。

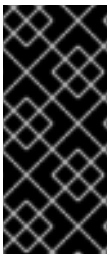
関連情報

- Microsoft Azure ストレージのドキュメント
- Microsoft Azure ストレージアカウントの作成
- Red Hat カスタマーポータルでのケースの作成
- Help + support
- Red Hat サポートへの問い合わせ

第3章 MICROSOFT AZURE での仮想マシンとして RED HAT ENTERPRISE LINUX イメージをデプロイ

Microsoft Azure に Red Hat Enterprise Linux 9(RHEL 9) イメージをデプロイするには、以下の情報に従ってください。この章の内容は次のとおりです。

- イメージを選ぶ際の選択肢について
- ホストシステムおよび仮想マシン (VM) のシステム要件のリストまたは参照
- ISO イメージからカスタム仮想マシンを作成し、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



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、Azure Disk Image (VHD 形式) を作成してアップロードできます。詳細は Composing a Customized RHEL System Image を参照してください。

Azure でセキュアに使用できる Red Hat 製品のリストは、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。

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

以下の表には、Microsoft Azure 上での RHEL 9 用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 をデプロイする	既存の Red Hat サブスクリプションを使用する	Azure で Red Hat Gold Image を選択します。Gold Image の詳細と、Azure で Gold Image 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。

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
Azure に移動するカスタムイメージをデプロイする	既存の Red Hat サブスクリプションを使用する	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。
RHEL を含む既存の Azure イメージをデプロイする	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。	Microsoft 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" 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。

関連情報

- Using Red Hat Gold Images on Microsoft Azure
- Azure Marketplace
- Billing options in the Azure Marketplace
- Red Hat Enterprise Linux Bring-Your-Own-Subscription Gold Images in Azure
- Red Hat Cloud Access リファレンスガイド

3.2. ベースイメージの理解

このセクションでは、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。

3.2.1. カスタムベースイメージの使用

仮想マシン (VM) を手動で設定するには、まずベース (スターター) となる仮想マシンイメージを作成します。続いて、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 のクラウドイメージを準備するには、以下のセクションの手順に従います。RHEL の Hyper-V クラウドイメージを準備するには、Hyper-V Manager から Red Hat ベースの仮想マシンの準備 を参照してください。

3.2.2. 必要なシステムパッケージ

RHEL の基本イメージを作成して設定するには、ホストシステムに次のパッケージがインストールされている必要があります。

表3.2 システムパッケージ

パッケージ	リポジトリー	説明
libvirt	rhel-9-for-x86_64-appstream-rpms	プラットフォーム仮想化を管理するためのオープンソース API、デーモン、および管理ツール。
virt-install	rhel-9-for-x86_64-appstream-rpms	仮想マシンを構築するためのコマンドラインユーティリティー
libguestfs	rhel-9-for-x86_64-appstream-rpms	仮想マシンファイルシステムにアクセスして変更するためのライブラリー
guestfs-tools	rhel-9-for-x86_64-appstream-rpms	仮想マシン用のシステム管理ツール。**virt-customize** ユーティリティーが含まれています

3.2.3. Azure VM 設定

Azure 仮想マシン(VM)には、以下の設定が必要です。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。この手順を進める際には、この設定に留意してください。必要に応じてこの設定を参照します。

表3.3 仮想マシンの設定

設定	推奨事項
SSH	Azure 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 (IPv4 のみ) 用に設定する必要があります。
スワップ領域	インストール中に、オペレーティングシステム(OS)ディスクまたはストレージディスクに専用の **swap** ファイルまたはスワップパーティションを作成しないでください。**cloud-init** ユーティリティーを設定して、仮想マシンの一時ディスクに **swap** パーティションを自動的に作成します。一時ディスクは仮想マシンのローカルストレージであり、リソースディスクは仮想マシン自体にストレージをマウントしています。どちらのストレージタイプも一時的にデータを保存します。

設定	推奨事項
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	カスタムイメージの場合には、Azure で完全なディスク暗号化に Network Bound Disk Encryption (NBDE) を使用します。

3.2.4. Azure での cloud-init を使用したスワップ領域の設定

Microsoft Azure 上の Red Hat Enterprise Linux (RHEL)仮想マシン(VM)にスワップ領域を使用するには、一時ディスクに swap パーティションを作成する必要があります。オペレーティングシステム(OS)ディスクまたはデータ（ストレージ）ディスクではなく、swap パーティションの作成には一時ディスクのみを使用してください。仮想マシンの削除時に一時ディスクが削除されるため、swap パーティションも削除されます。

cloud-init ユーティリティーを使用して、一時ディスクに swap パーティションをオンデマンドで設定できます。一時ディスクは仮想マシンのローカルストレージであり、リソースディスクは仮想マシン自体にストレージをマウントします。どちらのストレージタイプも一時的にデータを保存します。仮想マシンを削除、移動、停止、または障害が発生すると、一時またはリソースディスクに保存されているデータが失われます。

重要

永続データに一時ディスクを使用しないでください。仮想マシンが停止または移動されると、スワップパーティションを含むすべてのコンテンツが削除されます。

前提条件

- 仮想マシンに **cloud-init** ユーティリティーがインストールされている。
- **/etc/waagent.conf** ファイルにパラメーターを設定して、Windows Azure Linux Agent (WALA) でスワップ設定を無効にしている。

```
ResourceDisk.Format=n
ResourceDisk.EnableSwap=n
ResourceDisk.SwapSizeMB=0
```

- 仮想マシンに一時ディスクが利用可能である。

手順

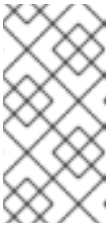
1. 仮想マシンにログインします。
2. **/etc/cloud/cloud.cfg.d/00-azure-swap.cfg** 設定ファイルを作成および編集し、次の **cloud-init** 設定を追加します。

```
# vi /etc/cloud/cloud.cfg.d/00-azure-swap.cfg
```

```
#cloud-config
disk_setup:
  ephemeral0:
    table_type: gpt
    layout: [66, [33,82]]
    overwrite: true
fs_setup:
  - device: ephemeral0.1
    filesystem: ext4
  - device: ephemeral0.2
    filesystem: swap
mounts:
  - ["ephemeral0.1", "/mnt"]
  - ["ephemeral0.2", "none", "swap", "sw,nofail,x-systemd.requires=cloud-init.service", "0", "0"]
```

この設定は、以下のようになります。

- 一時ディスク(**ephemeral0**)を GPT パーティションテーブルでパーティションします。
- ファイルシステムの場合は 66%(/**mnt**にマウント)と **スワップ** 領域の場合は 33% の 2 つのパーティションを作成します。
- 最初のパーティションを **ext4** と 2 番目のパーティションを **swap** としてフォーマットします。
- 起動時に両方のパーティションの自動マウントを設定します。



注記

パーティションレイアウト **[66, [33,82]** はディスクの 66% を最初のパーティションに割り当て、33% を 2 番目のパーティションに割り当てます。2 番目のパーティション仕様の **82** は、Linux swap パーティションタイプを示します。これらの割合は、要件に応じて調整できます。

3. 設定ファイルでエラーの有無を確認します。

```
# cloud-init devel schema --config-file /etc/cloud/cloud.cfg.d/00-azure-swap.cfg
```

設定が有効な場合、コマンドはエラーを返しません。

検証

- 仮想マシンを再起動したら、**/etc/fstab** ファイルのアクティブな swap 領域、swap 使用量、および swap パーティションエントリーを確認して、swap パーティションが設定され、アクティブであることを確認します。
  - アクティブなスワップ領域を確認します。

```
$ swapon -s
```

出力には、**ephemeral0.2** からの swap パーティションが表示されるはずです。

```
Filename                Type        Size     Used    Priority
/dev/ephemeral0.2       partition   8388604  0       -2
```

- スワップの使用状況を確認します。

```
$ free -h
```

出力には、**Swap** 行にスワップ領域が表示されるはずです。

```
               total        used        free      shared  buffered/cache   available
Mem:           7.8Gi       1.2Gi       5.8Gi        16MiB       800MiB       6.3Gi
Swap:          8.0Gi          0B       8.0Gi
```

- swap パーティションが **/etc/fstab** ファイルに存在することを確認します。

```
$ grep swap /etc/fstab
```

出力には、swap パーティションのエントリーが含まれている必要があります。次に例を示します。

```
/dev/ephemeral0.2   none    swap  sw,nofail,x-systemd.requires=cloud-init.service   0
0
```

関連情報

- Azure Linux 仮想マシンで cloud-init を使用してスワップファイルを設定する
- Azure の Linux 仮想マシンでスワップファイルを作成して設定
- cloud-init のディスク設定

3.2.5. 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。

前提条件

- 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。設定および手順は、RHEL 9 で仮想化を有効にする を参照してください。

手順

1. Red Hat カスタマーポータル から最新の Red Hat Enterprise Linux 9 DVD ISO イメージをダウンロードします。
2. 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。
   たとえば、次のコマンドは **rhel-9.0-x86_64-kvm.qcow2** イメージを使用して **kvmtest** 仮想マシンを作成します。

```
# virt-install \
    --name kvmtest --memory 2048 --vcpus 2 \
    --disk rhel-9.0-x86_64-kvm.qcow2,bus=virtio \
    --import --os-variant=rhel9.0
```

b. Web コンソールを使用して仮想マシンを作成する場合は、Web コンソールで仮想マシンの作成 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** を希望のサイズに変更します。
- 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. 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** には 500 MB 以上であることを確認してください。ただし、1 GB 以上で十分です。
  - 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。
  - インストール時に、OS ディスクからスワップ領域を削除します。デプロイメント後に一時ディスクで **cloud-init** を使用してスワップ領域を設定します。
- **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。

4. 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。

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 アカウントにログインします。

6. **root** でログインしたら、イメージを設定できます。

3.3. MICROSOFT AZURE のカスタムベースイメージの設定

仮想マシン (VM) のカスタムベースイメージを作成し、Azure で特定の設定を使用して RHEL 9 VM をデプロイすることができます。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。

3.3.1. 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 (VM) 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

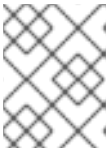
1. 以下の **grep** コマンドを実行して、必要な Hyper-V デバイスドライバーがインストールされているかどうかを確認します。

```
# 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.el9.x86_64/kernel/drivers/hv
-rw-r--r--   1 root     root        31272 Aug 11 08:45 usr/lib/modules/3.10.0-
932.el9.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.el9.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.el9.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。



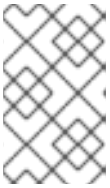
注記

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、次の手順を実行してください。

2. **/etc/dracut.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 **hv.conf** ファイルに追加します。

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



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

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。

3.3.2. Microsoft Azure のデプロイメントに必要な設定変更を行う

カスタムベースイメージを Azure にデプロイする前に、追加の設定変更を実行して、仮想マシン (VM) が Azure で適切に動作できるようにする必要があります。

手順

1. 仮想マシンにログインします。
2. 仮想マシンを登録し、Red Hat Enterprise Linux 9 リポジトリーを有効にします。

```
# subscription-manager register
Installed Product Current Status:
Product Name: Red Hat Enterprise Linux for x86_64
Status: Subscribed
```

3. **cloud-init** および **hyperv-daemons** パッケージがインストールされていることを確認します。

```
# dnf install cloud-init hyperv-daemons -y
```

4. Azure サービスとの統合に必要な **cloud-init** 設定ファイルを作成します。
   a. Hyper-V Data Exchange Service (KVP) へのログ記録を有効にするには、**/etc/cloud/cloud.cfg.d/10-azure-kvp.cfg** 設定ファイルを作成し、そのファイルに次の行を追加します。

```
reporting:
    logging:
        type: log
    telemetry:
        type: hyperv
```

   b. Azure をデータソースとして追加するには、**/etc/cloud/cloud.cfg.d/91-azure_datasource.cfg** 設定ファイルを作成し、そのファイルに次の行を追加します。

```
datasource_list: [ Azure ]
datasource:
    Azure:
        apply_network_config: False
```

   c. 一時ディスクにスワップ領域を設定するには、**/etc/cloud/cloud.cfg.d/00-azure-swap.cfg** 設定ファイルを作成し、次の行を追加します。

重要

一時ディスクは一時ストレージです。したがって、スワップ領域を含む 保存したデータは、VM が割り当て解除されたり、移動されたりすると失われます。一時ディスクは、スワップ領域などの一時データにのみ使用します。

```
#cloud-config
disk_setup:
  ephemeral0:
    table_type: gpt
```

```
    layout: [66, [33,82]]
    overwrite: true
fs_setup:
  - device: ephemeral0.1
    filesystem: ext4
  - device: ephemeral0.2
    filesystem: swap
mounts:
  - ["ephemeral0.1", "/mnt"]
  - ["ephemeral0.2", "none", "swap", "sw,nofail,x-systemd.requires=cloud-init.service", "0",
"0"]
```

5. 特定のカーネルモジュールが自動的にロードされないようにするには、**/etc/modprobe.d/blocklist.conf** ファイルを編集または作成し、そのファイルに次の行を追加します。

```
blacklist nouveau
blacklist lbm-nouveau
blacklist floppy
blacklist amdgpu
blacklist skx_edac
blacklist intel_cstate
```

6. **udev** ネットワークデバイスルールを変更します。

   a. 次の永続的なネットワークデバイスルールが存在す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
   ```

   b. Azure で Accelerated Networking が意図したとおりに動作するようにするには、新しいネットワークデバイスルール **/etc/udev/rules.d/68-azure-sriov-nm-unmanaged.rules** を作成し、次の行を追加します。

   ```
   SUBSYSTEM=="net", DRIVERS=="hv_pci", ACTION=="add",
   ENV{NM_UNMANAGED}="1"
   ```

7. **sshd** サービスが自動的に起動するように設定します。

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

8. カーネルブートパラメーターを変更します。

   a. **/etc/default/grub** ファイルを開き、**GRUB_TIMEOUT** 行に次の値があることを確認します。

   ```
   GRUB_TIMEOUT=10
   ```

   b. 次のオプションがある場合は、**GRUB_CMDLINE_LINUX** 行の末尾から削除します。

   ```
   rhgb quiet
   ```

c. **/etc/default/grub** ファイルに、指定されたすべてのオプションを含む次の行が含まれていることを確認します。

```
GRUB_CMDLINE_LINUX="loglevel=3 crashkernel=auto console=tty1 console=ttyS0
earlyprintk=ttyS0 rootdelay=300"
GRUB_TIMEOUT_STYLE=countdown
GRUB_TERMINAL="serial console"
GRUB_SERIAL_COMMAND="serial --speed=115200 --unit=0 --word=8 --parity=no --
stop=1"
```

注記

HDD でワークロードを実行していない場合は、**GRUB_CMDLINE_LINUX** 行の最後に **elevator=none** を追加します。これにより、I/O スケジューラーが **none** に設定され、SSD ベースのシステムでの I/O パフォーマンスが向上します。

d. **grub.cfg** ファイルを再生成します。

- BIOS ベースのマシンの場合:
  - RHEL 9.2 以前

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

  - RHEL 9.3 以降の場合:

```
# grub2-mkconfig -o /boot/grub2/grub.cfg --update-bls-cmdline
```

- UEFI ベースのマシンの場合:
  - RHEL 9.2 以前

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

  - RHEL 9.3 以降の場合:

```
# grub2-mkconfig -o /boot/grub2/grub.cfg --update-bls-cmdline
```

警告

grub.cfg を再構築するパスは、BIOS ベースと UEFI ベースの両マシンで同じです。実際の **grub.cfg** は BIOS パスにのみ存在します。UEFI パスには、**grub2-mkconfig** コマンドを使用して変更または再作成してはならないスタブファイルがあります。

システムが **grub.cfg** にデフォルト以外の場所を使用している場合は、それに応じ

システムが **grub.cfg** にデフォルト以外の場所を使用している場合は、それに応じてコマンドを調整してください。

9. Windows Azure Linux Agent (**WALinuxAgent**) を設定します。

   a. **WALinuxAgent** パッケージをインストールして有効にします。

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

   b. WALinuxAgent でスワップ設定を無効にするには(**cloud-init** を使用して swap を管理する場合に必要)、**/etc/waagent.conf** ファイルの以下の行を編集します。

   ```
   Provisioning.DeleteRootPassword=y
   ResourceDisk.Format=n
   ResourceDisk.EnableSwap=n
   ResourceDisk.SwapSizeMB=0
   ```

   注記

   WALinuxAgent で swap を無効にすると、**cloud-init** が一時ディスクのスワップ設定を管理できます。

10. Azure プロビジョニング用に VM を準備します。

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

    ```
    # subscription-manager unregister
    ```

    b. 既存のプロビジョニングの詳細をクリーンアップします。

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

    注記

    このコマンドは警告を生成しますが、Azure が VM のプロビジョニングを自動的に処理するため、これは想定されています。

    c. 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
    ```

3.4. 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。イメージを **qcow2** から固定 **VHD** 形式に変換し、イメージを整列するには、次の手順を参照してください。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. 以下の内容で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."**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。

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。

5. **raw** イメージが整列していない場合は、以下の手順で整列させてください。

a. 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>
```

b. **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。

3.5. AZURE CLI のインストール

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. **dnf** パッケージインデックスを更新します。

```
$ dnf 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 dnf install python3
```

5. Azure CLI をインストールします。

```
$ sudo dnf 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

3.6. 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. Azure でシステムを認証し、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。

```
[clouduser@localhost]$ 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。

```
[clouduser@localhost]$ 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/azrhelclistact",
  "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/azrhelclivnet1",
  "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/azrhelclivnet1/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 Managed Disks Overview
- SKU Types

3.7. 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。これには数分かかる場合があります。ストレージコンテナーのリストを表示するには、**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。

```
[clouduser@localhost]$ az storage blob upload \
--account-name azrhelclistact --container-name azrhelclistcont \
--type page --file rhel-image-{ProductNumber}.vhd --name rhel-image-{ProductNumber}.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-9.vhd
"https://azrhelclistact.blob.core.windows.net/azrhelclistcont/rhel-image-9.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 rhel9 -g azrhelclirsgrp2 -l southcentralus --source
https://azrhelclistact.blob.core.windows.net/azrhelclistcont/rhel-image-9.vhd --os-type linux
```

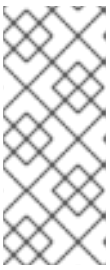
3.8. AZURE での仮想マシンの作成および起動

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、その他認証方法 を参照してください。

以下に例を示します。

```
[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 rhel9

{
  "fqdns": "",
  "id":
"/subscriptions//resourceGroups/azrhelclirsgrp/providers/Microsoft.Compute/virtualMachines/rhe
```

```
l-azure-vm-1",
  "location": "southcentralus",
  "macAddress": "",
  "powerState": "VM running",
  "privateIpAddress": "10.0.0.4",
  "publicIpAddress": "<public-IP-address>",
  "resourceGroup": "azrhelclirsgrp2"
```

publicIpAddress を書き留めます。以下の手順で仮想マシンにログインするには、このアドレスが必要です。

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.

[clouduser@rhel-azure-vm-1 ~]$
```

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、Microsoft 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** 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。

3.9. 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、パスワードを使用した 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> --authentication-type password \
   --admin-username <administrator-name> --admin-password <ssh-password> --image <path-to-image>
```

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

例 2: これら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>
```

3.10. RED HAT サブスクリプションの割り当て

subscription-manager コマンドを使用すると、Red Hat サブスクリプションを登録して RHEL インスタンスに割り当てることができます。

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
```

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。詳細は、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、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もできます。ホストベースのサブスクリプションのハイパーバイザーへの割り当て を参照してください。
3. オプション： Red Hat Hybrid Cloud Console のインスタンスに関するさまざまなシステムメトリックを収集するには、インスタンスを Red Hat Lightspeed に登録できます。

```
# insights-client register --display-name <display_name_value>
```

   Red Hat Lightspeed の詳細な設定の詳細は、Client Configuration Guide for Red Hat Lightspeed 参照してください。

関連情報

- Simple Content Access とは何ですか？
- Creating Red Hat Customer Portal Activation Keys
- ホストベースのサブスクリプションのハイパーバイザーへの割り当て
- Red Hat Lightspeed のクライアント設定ガイド

3.11. AZURE GOLD IMAGE の自動登録の設定

Micorsoft Azure 上に RHEL 9 の仮想マシン (VM) をより早く、より快適にデプロイするために、RHEL 9 の Gold Image を Red Hat Subscription Manager(RHSM) に自動的に登録するように設定することができます。

前提条件

- RHEL 9 Gold Image は Microsoft Azure で利用できます。手順は、Azure でのゴールドイメージの使用 を参照してください。

注記

Microsoft Azure アカウントは、一度に 1 つの Red Hat アカウントにしか割り当てできません。そのため、Red Hat アカウントにアタッチする前に、他のユーザーが Azure アカウントへのアクセスを必要としていないことを確認してください。

手順

1. Gold Image を使用して、Azure インスタンスに RHEL 9 仮想マシンを作成します。手順については、Creating and starting the VM in Azure を参照してください。
2. 作成した仮想マシンを起動します。
3. RHEL 9 仮想マシンで、自動登録を有効にします。

```
# subscription-manager config --rhsmcertd.auto_registration=1
```

4. **rhsmcertd** サービスを有効にします。

```
# systemctl enable rhsmcertd.service
```

5. **redhat.repo** リポジトリーを無効にします。

```
# subscription-manager config --rhsm.manage_repos=0
```

6. 仮想マシンの電源を切り、Azure 上で管理イメージとして保存します。手順については、How to create a managed image of a virtual machine or VHD を参照してください。
7. 管理イメージを使用して仮想マシンを作成します。自動的に RHSM に登録されます。

検証

- 上記の手順で作成した RHEL 9 仮想マシンで、**subscription-manager identity** コマンドを実行して、システムが RHSM に登録されていることを確認します。登録に成功したシステムでは、システムの UUID が表示されます。以下に例を示します。

```
# subscription-manager identity
system identity: fdc46662-c536-43fb-a18a-bbcb283102b7
name: 192.168.122.222
org name: 6340056
org ID: 6340056
```

関連情報

- Azure での Red Hat ゴールドイメージ
- Overview of RHEL images in Azure
- Red Hat サービスのクラウド統合の設定

3.12. MICROSOFT AZURE インスタンス用の KDUMP の設定

RHEL インスタンスでカーネルクラッシュが発生した場合は、**kdump** サービスを使用してクラッシュの原因を特定できます。インスタンスカーネルが予期せず終了したときに **kdump** が正しく設定されていれば、**kdump** はクラッシュダンプまたは **vmcore** ファイルと呼ばれるダンプファイルを生成します。その後、ファイルを分析してクラッシュが発生した原因を見つけ、システムをデバッグできます。

kdump を Microsoft Azure インスタンスで動作させるには、仮想マシンのサイズと RHEL バージョンに合わせて、**kdump** の予約メモリーと **vmcore** ターゲットの調整が必要になる場合があります。

前提条件

- **kdump** をサポートする Microsoft Azure 環境を使用している。
  - Standard_DS2_v2 VM
  - Standard NV16as v4
  - Standard M416-208s v2
  - Standard M416ms v2
- システムの **root** 権限がある。
- システムが、**kdump** の設定とターゲットの要件を満たしている。詳細は サポートされている kdump 設定とターゲット を参照してください。

手順

1. **kdump** およびその他の必要なパッケージがシステムにインストールされていることを確認します。

```
# dnf install kexec-tools
```

2. クラッシュダンプファイルのデフォルトの場所が **kdump** 設定ファイルで設定されており、**/var/crash** ファイルが使用可能であることを確認します。

```
# grep -v "#" /etc/kdump.conf

path /var/crash
core_collector makedumpfile -l --message-level 7 -d 31
```

3. RHEL 仮想マシン (VM) インスタンスのサイズとバージョンに基づいて、**/mnt/crash** などのより多くの空き領域を持つ **vmcore** ターゲットが必要かどうかを判断します。これを行うには、次の表を使用します。

表3.4 Azure 上の GEN2 VM でテストされた仮想マシンのサイズ

RHEL バージョン	Standard DS1 v2 (1 vCPU、3.5GiB)	Standard NV16as v4 (16 vCPU、56 GiB)	Standard M416-208s v2 (208 vCPU、5700 GiB)	Standard M416ms v2 (416 vCPU、11400 GiB)
RHEL 9.0 - RHEL 9.3	デフォルト	デフォルト	Target	Target

- **Default** は、デフォルトのメモリーとデフォルトの **kdump** ターゲットで **kdump** が期待どおりに動作することを示します。デフォルトの **kdump** ターゲットは **/var/crash** です。
- **Target** は、**kdump** がデフォルトのメモリーで期待どおりに動作することを示します。ただし、より多くの空き領域を持つターゲットを割り当てる必要がある場合があります。

4. インスタンスで必要な場合は、**/mnt/crash** など、より多くの空き領域を持つターゲットを割り当てます。これを行うには、**/etc/kdump.conf** ファイルを編集し、デフォルトのパスを置き換えます。

```
$ sed s/"path /var/crash"/"path /mnt/crash"
```

オプションパス **/mnt/crash** は、**kdump** がクラッシュダンプファイルを保存するファイルシステムへのパスを表します。

クラッシュダンプファイルを別のパーティションに書き込む、デバイスに直接書き込む、リモートマシンに保存するなどのその他のオプションは、kdump ターゲットの設定 を参照してください。

5. インスタンスで必要な場合は、それぞれのブートパラメーターを追加して、クラッシュカーネルのサイズを、**kdump** が **vmcore** をキャプチャーするのに十分なサイズまで増やします。
たとえば、Standard M416-208s v2 仮想マシンの場合、十分なサイズは 512 MB なので、ブートパラメーターは **crashkernel=512M** になります。

   a. GRUB 設定ファイルを開き、ブートパラメーター行に **crashkernel=512M** を追加します。

```
# vi /etc/default/grub

GRUB_CMDLINE_LINUX="console=tty1 console=ttyS0 earlyprintk=ttyS0 rootdelay=300
crashkernel=512M"
```

   b. GRUB 設定ファイルを更新します。

   - RHEL 9.2 以前

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

   - RHEL 9.3 以降の場合:

```
# grub2-mkconfig -o /boot/grub2/grub.cfg --update-bls-cmdline
```

6. 仮想マシンを再起動して、仮想マシンに個別のカーネルクラッシュメモリーを割り当てます。

検証

- **kdump** がアクティブで実行されていることを確認します。

```
# systemctl status kdump
● kdump.service - Crash recovery kernel arming
   Loaded: loaded (/usr/lib/systemd/system/kdump.service; enabled; vendor prese>
   Active: active (exited) since Fri 2024-02-09 10:50:18 CET; 1h 20min ago
  Process: 1252 ExecStart=/usr/bin/kdumpctl start (code=exited, status=0/SUCCES>
 Main PID: 1252 (code=exited, status=0/SUCCESS)
    Tasks: 0 (limit: 16975)
   Memory: 512B
   CGroup: /system.slice/kdump.service
```

関連情報

- kdump のインストール
- How to troubleshoot kernel crashes, hangs, or reboots with kdump on Red Hat Enterprise Linux (Red Hat ナレッジベース)
- kdump メモリー要件

3.13. 関連情報

- Red Hat in the Public Cloud
- Red Hat Cloud Access リファレンスガイド
- Frequently Asked Questions and Recommended Practices for Microsoft Azure

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

ノード障害が発生した場合に RHEL ノードがワークロードを自動的に再分配するクラスターを作成するには、Red Hat High Availability Add-On を使用します。このような高可用性 (HA) クラスターは、Microsoft Azure などのパブリッククラウドプラットフォームでもホストできます。Azure 上での RHEL HA クラスターの作成は、特定の詳細を除けば、非クラウド環境での HA クラスターの作成と同様です。

Azure 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 Azure 上に Red Hat HA クラスターを設定するには、以下のセクションを参照してください。これらのセクションの手順では、Azure 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 9 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。

次のセクションで説明します。

- Azure 用の環境を設定するための前提手順。環境を設定したら、Azure VM インスタンスを作成して設定できます。
- 個々のノードを Azure 上の 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。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。

前提条件

- Red Hat カスタマーポータル のアカウントにサインアップします。
- 管理者権限で Microsoft Azure アカウント にサインアップします。
- 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。

4.1. パブリッククラウドプラットフォームで高可用性クラスターを使用する利点

高可用性 (HA) クラスターは、特定のワークロードを実行するためにリンクされた一連のコンピューター (**ノード** と呼ばれます) です。HA クラスターの目的は、ハードウェアまたはソフトウェア障害が発生した場合に備えて、冗長性を提供することです。HA クラスター内のノードに障害が発生しても、Pacemaker クラスターリソースマネージャーがワークロードを他のノードに分散するため、クラスター上で実行されているサービスに顕著なダウンタイムが発生することはありません。

HA クラスターはパブリッククラウドプラットフォームで実行することもできます。この場合、クラウド内の仮想マシン (VM) インスタンスを個々のクラスターノードとして使用します。パブリッククラウドプラットフォームで HA クラスターを使用すると、次の利点があります。

- 可用性の向上: VM に障害が発生した場合、ワークロードがすぐに他のノードに再分散されるため、実行中のサービスが中断されません。
- スケーラビリティー: 需要が高いときに追加のノードを起動し、需要が低いときにノードを停止することができます。
- 費用対効果: 従量課金制の料金設定では、実行中のノードに対して料金を支払うだけで済みます。

- 管理の簡素化: 一部のパブリッククラウドプラットフォームでは、HA クラスターの設定を容易にする管理インターフェイスが提供されています。

Red Hat Enterprise Linux (RHEL) システムで HA を有効にするために、Red Hat は High Availability Add-On を提供しています。High Availability Add-On は、RHEL システム上で HA クラスターを作成するために必要なすべてのコンポーネントを提供します。コンポーネントには、高可用性サービス管理ツールとクラスター管理ツールが含まれます。

関連情報

- High Availability Add-On の概要

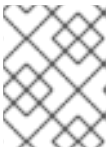
4.2. AZURE でのリソースの作成

リージョン、リソースグループ、ストレージアカウント、仮想ネットワーク、および可用性セットを作成するには、以下の手順を実施します。Microsoft Azure でクラスターをセットアップするには、これらのリソースが必要です。

手順

1. Azure でシステムを認証し、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。

以下に例を示します。

```
[clouduser@localhost]$ az login

To sign in, use a web browser to open the page https://aka.ms/devicelogin and enter the code FDMSCMETZ to authenticate.
 [
   {
     "cloudName": "AzureCloud",
     "id": "Subscription ID",
     "isDefault": true,
     "name": "MySubscriptionName",
     "state": "Enabled",
     "tenantId": "Tenant ID",
     "user": {
       "name": "clouduser@company.com",
       "type": "user"
     }
   }
 ]
```

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。

```
[clouduser@localhost]$ 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。

```
$ az storage account create -l azure-region -n storage-account-name -g resource-group --sku sku_type --kind StorageV2
```

以下に例を示します。

```
[clouduser@localhost]$ az storage account create -l southcentralus -n azrhelclistact -g azrhelclirsgrp --sku Standard_LRS --kind StorageV2

{
  "accessTier": null,
  "creationTime": "2017-04-05T19:10:29.855470+00:00",
  "customDomain": null,
  "encryption": null,
  "id":
"/subscriptions//resourceGroups/azrhelclirsgrp/providers/Microsoft.Storage/storageAccounts/azrhelclistact",
  "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。すべてのクラスターノードが同じ仮想ネットワークにある必要があります。

```
$ 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
  }
}
```


8. 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

4.3. 高可用性に必要なシステムパッケージ

この手順では、Red Hat Enterprise Linux を使用して Azure HA 用の仮想マシンイメージを作成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、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4.1 システムパッケージ

パッケージ	リポジトリー	説明
libvirt	rhel-9-for-x86_64-appstream-rpms	プラットフォーム仮想化を管理するためのオープンソース API、デーモン、および管理ツール。
virt-install	rhel-9-for-x86_64-appstream-rpms	仮想マシンを構築するためのコマンドラインユーティリティー
libguestfs	rhel-9-for-x86_64-appstream-rpms	仮想マシンファイルシステムにアクセスして変更するためのライブラリー
guestfs-tools	rhel-9-for-x86_64-appstream-rpms	仮想マシン用のシステム管理ツール。**virt-customize** ユーティリティーが含まれています

4.4. AZURE VM 設定

Azure 仮想マシン(VM)には、以下の設定が必要です。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。この手順を進める際には、この設定に留意してください。必要に応じてこの設定を参照します。

表4.2 仮想マシンの設定

設定	推奨事項
SSH	Azure 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。

設定	推奨事項
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。
スワップ領域	インストール中に、オペレーティングシステム(OS)ディスクまたはストレージディスクに専用の **swap** ファイルまたはスワップパーティションを作成しないでください。**cloud-init** ユーティリティーを設定して、仮想マシンの一時ディスクに **swap** パーティションを自動的に作成します。一時ディスクは仮想マシンのローカルストレージであり、リソースディスクは仮想マシン自体にストレージをマウントしています。どちらのストレージタイプも一時的にデータを保存します。
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	カスタムイメージの場合には、Azure で完全なディスク暗号化に Network Bound Disk Encryption (NBDE) を使用します。

4.5. 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 (VM) 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.el9.x86_64/kernel/drivers/hv
-rw-r--r--   1 root     root        31272 Aug 11 08:45 usr/lib/modules/3.10.0-
932.el9.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.el9.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.el9.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、次の手順を実行してください。

2. **/etc/dracut.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 **hv.conf** ファイルに追加します。

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

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

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。

4.6. MICROSOFT AZURE のデプロイメントに必要な設定変更を行う

カスタムベースイメージを Azure にデプロイする前に、追加の設定変更を実行して、仮想マシン (VM) が Azure で適切に動作できるようにする必要があります。

手順

1. 仮想マシンにログインします。
2. 仮想マシンを登録し、Red Hat Enterprise Linux 9 リポジトリーを有効にします。

```
# subscription-manager register
Installed Product Current Status:
Product Name: Red Hat Enterprise Linux for x86_64
Status: Subscribed
```

3. **cloud-init** および **hyperv-daemons** パッケージがインストールされていることを確認します。

```
# dnf install cloud-init hyperv-daemons -y
```

4. Azure サービスとの統合に必要な **cloud-init** 設定ファイルを作成します。

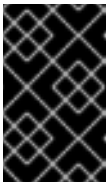
   a. Hyper-V Data Exchange Service (KVP) へのログ記録を有効にするには、**/etc/cloud/cloud.cfg.d/10-azure-kvp.cfg** 設定ファイルを作成し、そのファイルに次の行を追加します。

   ```
   reporting:
     logging:
       type: log
     telemetry:
       type: hyperv
   ```

   b. Azure をデータソースとして追加するには、**/etc/cloud/cloud.cfg.d/91-azure_datasource.cfg** 設定ファイルを作成し、そのファイルに次の行を追加します。

   ```
   datasource_list: [ Azure ]
   datasource:
     Azure:
       apply_network_config: False
   ```

   c. 一時ディスクにスワップ領域を設定するには、**/etc/cloud/cloud.cfg.d/00-azure-swap.cfg** 設定ファイルを作成し、次の行を追加します。

   重要

   一時ディスクは一時ストレージです。したがって、スワップ領域を含む 保存したデータは、VM が割り当て解除されたり、移動されたりすると失われます。一時ディスクは、スワップ領域などの一時データにのみ使用します。

   ```
   #cloud-config
   disk_setup:
     ephemeral0:
       table_type: gpt
       layout: [66, [33,82]]
       overwrite: true
   fs_setup:
     - device: ephemeral0.1
       filesystem: ext4
     - device: ephemeral0.2
       filesystem: swap
   mounts:
     - ["ephemeral0.1", "/mnt"]
     - ["ephemeral0.2", "none", "swap", "sw,nofail,x-systemd.requires=cloud-init.service", "0", "0"]
   ```

5. 特定のカーネルモジュールが自動的にロードされないようにするには、**/etc/modprobe.d/blocklist.conf** ファイルを編集または作成し、そのファイルに次の行を追加します。

   ```
   blacklist nouveau
   blacklist lbm-nouveau
   blacklist floppy
   blacklist amdgpu
   ```

```
blacklist skx_edac
blacklist intel_cstate
```

6. **udev** ネットワークデバイスルールを変更します。

   a. 次の永続的なネットワークデバイスルールが存在す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
   ```

   b. Azure で Accelerated Networking が意図したとおりに動作するようにするには、新しいネットワークデバイスルール **/etc/udev/rules.d/68-azure-sriov-nm-unmanaged.rules** を作成し、次の行を追加します。

   ```
   SUBSYSTEM=="net", DRIVERS=="hv_pci", ACTION=="add",
   ENV{NM_UNMANAGED}="1"
   ```

7. **sshd** サービスが自動的に起動するように設定します。

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

8. カーネルブートパラメーターを変更します。

   a. **/etc/default/grub** ファイルを開き、**GRUB_TIMEOUT** 行に次の値があることを確認します。

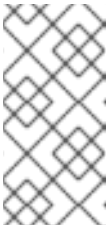
   ```
   GRUB_TIMEOUT=10
   ```

   b. 次のオプションがある場合は、**GRUB_CMDLINE_LINUX** 行の末尾から削除します。

   ```
   rhgb quiet
   ```

   c. **/etc/default/grub** ファイルに、指定されたすべてのオプションを含む次の行が含まれていることを確認します。

   ```
   GRUB_CMDLINE_LINUX="loglevel=3 crashkernel=auto console=tty1 console=ttyS0
   earlyprintk=ttyS0 rootdelay=300"
   GRUB_TIMEOUT_STYLE=countdown
   GRUB_TERMINAL="serial console"
   GRUB_SERIAL_COMMAND="serial --speed=115200 --unit=0 --word=8 --parity=no --
   stop=1"
   ```

   注記

   HDD でワークロードを実行していない場合は、**GRUB_CMDLINE_LINUX** 行の最後に **elevator=none** を追加します。これにより、I/O スケジューラーが **none** に設定され、SSD ベースのシステムでの I/O パフォーマンスが向上します。

   d. **grub.cfg** ファイルを再生成します。

- BIOS ベースのマシンの場合:
  - RHEL 9.2 以前

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

  - RHEL 9.3 以降の場合:

```
# grub2-mkconfig -o /boot/grub2/grub.cfg --update-bls-cmdline
```

- UEFI ベースのマシンの場合:
  - RHEL 9.2 以前

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

  - RHEL 9.3 以降の場合:

```
# grub2-mkconfig -o /boot/grub2/grub.cfg --update-bls-cmdline
```

警告

grub.cfg を再構築するパスは、BIOS ベースと UEFI ベースの両マシンで同じです。実際の **grub.cfg** は BIOS パスにのみ存在します。UEFI パスには、**grub2-mkconfig** コマンドを使用して変更または再作成してはならないスタブファイルがあります。

システムが **grub.cfg** にデフォルト以外の場所を使用している場合は、それに応じてコマンドを調整してください。

9. Windows Azure Linux Agent (**WALinuxAgent**) を設定します。
   a. **WALinuxAgent** パッケージをインストールして有効にします。

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

   b. WALinuxAgent でスワップ設定を無効にするには(**cloud-init** を使用して swap を管理する場合に必要)、**/etc/waagent.conf** ファイルの以下の行を編集します。

```
Provisioning.DeleteRootPassword=y
ResourceDisk.Format=n
ResourceDisk.EnableSwap=n
ResourceDisk.SwapSizeMB=0
```

注記

WALinuxAgent で swap を無効にすると、**cloud-init** が一時ディスクのスワップ設定を管理できます。

10. Azure プロビジョニング用に VM を準備します。

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

    ```
    # subscription-manager unregister
    ```

    b. 既存のプロビジョニングの詳細をクリーンアップします。

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

    注記

    このコマンドは警告を生成しますが、Azure が VM のプロビジョニングを自動的に処理するため、これは想定されています。

    c. 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
    ```

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

Azure Active Directory (AD) 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 操作のアクセスを許可し、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) がシステムにインストールされている。
- Microsoft Azure サブスクリプションの管理者または所有者である。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。

手順

1. HA クラスター内の任意のノードで、Azure アカウントにログインします。

    ```
    $ az login
    ```

2. Azure フェンスエージェントのカスタムロールの **json** 設定ファイルを作成します。次の設定を使用します。ただし、**<subscription-id>** はご自身のサブスクリプション ID に置き換えます。

    ```
    {
        "Name": "Linux Fence Agent Role",
        "description": "Allows to power-off and start virtual machines",
        "assignableScopes": [
            "/subscriptions/<subscription-id>"
    ```

```
        ],
        "actions": [
                "Microsoft.Compute/*/read",
                "Microsoft.Compute/virtualMachines/powerOff/action",
                "Microsoft.Compute/virtualMachines/start/action"
        ],
        "notActions": [],
        "dataActions": [],
        "notDataActions": []
}
```

3. Azure フェンスエージェントのカスタムロールを定義します。前の手順で作成した **json** ファイルを使用します。

```
$ az role definition create --role-definition azure-fence-role.json

{
  "assignableScopes": [
    "/subscriptions/<my-subscription-id>"
  ],
  "description": "Allows to power-off and start virtual machines",
  "id": "/subscriptions/<my-subscription-
id>/providers/Microsoft.Authorization/roleDefinitions/<role-id>",
  "name": "<role-id>",
  "permissions": [
    {
      "actions": [
        "Microsoft.Compute/*/read",
        "Microsoft.Compute/virtualMachines/powerOff/action",
        "Microsoft.Compute/virtualMachines/start/action"
      ],
      "dataActions": [],
      "notActions": [],
      "notDataActions": []
    }
  ],
  "roleName": "Linux Fence Agent Role",
  "roleType": "CustomRole",
  "type": "Microsoft.Authorization/roleDefinitions"
}
```

4. Azure Web コンソールインターフェイスで、**Virtual Machine** を選択 → 左側のメニューの **Identity** をクリックします。
5. **On** を選択→ **Save** をクリック→ **Yes** をクリックして確認します。
6. **Azure role assignments** → **Add role assignment** をクリックします。
7. ロールに必要な **Scope** (例: **Resource Group**) を選択します。
8. 必要な **Resource Group** を選択します。
9. オプション: 必要に応じて **Subscription** を変更します。
10. **Linux Fence Agent Role** ロールを選択します。

11. **Save** をクリックします。

検証

- Azure AD に表示されるノードを表示します。

```
# fence_azure_arm --msi -o list
node1,
node2,
[...]
```

このコマンドでクラスター上のすべてのノードが出力された場合、AD アプリケーションは正常に設定されています。

関連情報

- View the access a user has to Azure resources
- Create a custom role for the fence agent
- Assign Azure roles by using Azure CLI

4.8. 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。イメージを **qcow2** から固定 **VHD** 形式に変換し、イメージを整列するには、次の手順を参照してください。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. 以下の内容で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."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。

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。

5. **raw** イメージが整列していない場合は、以下の手順で整列させてください。
   a. 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>
```

   b. **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。

4.9. 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。これには数分かかる場合があります。ストレージコンテナーのリストを表示するには、**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。

```
[clouduser@localhost]$ az storage blob upload \
--account-name azrhelclistact --container-name azrhelclistcont \
```

```
--type page --file rhel-image-{ProductNumber}.vhd --name rhel-image-{ProductNumber}.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。

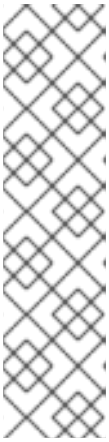
```
$ az storage blob url -c <container-name> -n <image-name>.vhd
```

以下に例を示します。

```
$ az storage blob url -c azrhelclistcont -n rhel-image-9.vhd
"https://azrhelclistact.blob.core.windows.net/azrhelclistcont/rhel-image-9.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 rhel9 -g azrhelclirsgrp2 -l southcentralus --source
https://azrhelclistact.blob.core.windows.net/azrhelclistcont/rhel-image-9.vhd --os-type linux
```

4.10. 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. SSH ターミナルセッションを起動し、管理者名とパブリック IP アドレスを使用して仮想マシンに接続します。

```
$ ssh administrator@PublicIP
```

Azure 仮想マシンのパブリック IP アドレスを取得するには、Azure ポータルで仮想マシンプロパティーを開くか、以下の Azure CLI コマンドを入力します。

```
$ az vm list -g <resource_group> -d --output table
```

以下に例を示します。

```
[clouduser@localhost ~] $ az vm list -g azrhelclirsgrp -d --output table
Name    ResourceGroup          PowerState      PublicIps       Location
------  ---------------------  --------------  -------------    --------------
node01  azrhelclirsgrp         VM running      192.98.152.251    southcentralus
```

2. 仮想マシンを Red Hat に登録します。

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

3. すべてのリポジトリーを無効にします。

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

4. RHEL9 サーバーの HA リポジトリーを有効にします。

```
# subscription-manager repos --enable=rhel-9-for-x86_64-highavailability-rpms
```

5. すべてのパッケージを更新します。

```
# dnf update -y
```

6. High Availability チャネルから、Red Hat High Availability Add-On ソフトウェアパッケージと Azure フェンスエージェントをインストールします。

```
# dnf install pcs pacemaker fence-agents-azure-arm
```

7. **hacluster** ユーザーは、最後に 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
```

8. **firewalld.service** がインストールされている場合は、RHEL ファイアウォールに **高可用性** サービスを追加します。

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

9. **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 start pcsd.service
# systemctl enable pcsd.service

Created symlink from /etc/systemd/system/multi-user.target.wants/pcsd.service to
/usr/lib/systemd/system/pcsd.service.
```

検証

- **pcs** サービスが実行していることを確認します。

```
# systemctl status pcsd.service
pcsd.service - PCS GUI and remote configuration interface
Loaded: loaded (/usr/lib/systemd/system/pcsd.service; enabled; vendor preset: disabled)
Active: active (running) since Fri 2018-02-23 11:00:58 EST; 1min 23s ago
Docs: man:pcsd(8)
        man:pcs(8)
Main PID: 46235 (pcsd)
  CGroup: /system.slice/pcsd.service
          └─46235 /usr/bin/ruby /usr/lib/pcsd/pcsd > /dev/null &
```

4.11. 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。コマンドで、クラスター内の各ノードの名前を指定します。

```
# pcs host auth <hostname1> <hostname2> <hostname3>
```

以下に例を示します。

```
[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 <cluster_name> <hostname1> <hostname2> <hostname3>
```

以下に例を示します。

```
[root@node01 clouduser]# pcs cluster setup new_cluster node01 node02 node03

[...]

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

1. クラスターを有効にします。

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

2. クラスターを起動します。

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

4.12. フェンシングの概要

クラスター内のノードの 1 つと通信が失敗した場合に、障害が発生したクラスターノードがアクセスする可能性があるリソースへのアクセスを、その他のノードが制限したり、解放したりできるようにする必要があります。クラスターノードが応答しない可能性があるため、そのクラスターノードと通信しても成功しません。代わりに、フェンスエージェントを使用した、フェンシングと呼ばれる外部メソッドを指定する必要があります。

応答しないノードがデータへのアクセスを続けている可能性があります。データが安全であることを確認する場合は、STONITH を使用してノードをフェンスすることが唯一の方法になります。STONITH は "Shoot The Other Node In The Head" の頭字語で、不安定なノードや同時アクセスによるデータの破損を防ぐことができます。STONITH を使用すると、別のノードからデータをアクセスする前に、そのノードが完全にオフラインであることを確認できます。

関連情報

- Fencing in Red Hat High Availability Cluster（Red Hat ナレッジベース）

4.13. フェンシングデバイスの作成

フェンシングを設定するには、以下の手順を実行します。クラスターの任意のノードからこのコマンドを完了します。

前提条件

クラスタープロパティー **stonith-enabled** を **true** に設定する必要があります。

手順

1. 各 RHEL 仮想マシンの Azure ノード名を特定します。Azure ノード名を使用してフェンスデバイスを設定します。

```
# fence_azure_arm \
   -l <AD-Application-ID> -p <AD-Password> \
   --resourceGroup <MyResourceGroup> --tenantId <Tenant-ID> \
   --subscriptionId <Subscription-ID> -o list
```

以下に例を示します。

```
[root@node01 clouduser]# fence_azure_arm \
-l e04a6a49-9f00-xxxx-xxxx-a8bdda4af447 -p
z/a05AwCN0IzAjVwXXXXXXXEWIoeVp0xg7QT//JE=
```

```
--resourceGroup azrhelclirsgrp --tenantId 77ecefb6-cff0-XXXX-XXXX-757XXXX9485
--subscriptionId XXXXXXXX-38b4-4527-XXXX-012d49dfc02c -o list

node01,
node02,
node03,
```

2. Azure ARM STONITH エージェントのオプションを表示します。

```
# pcs stonith describe fence_azure_arm
```

以下に例を示します。

```
# pcs stonith describe fence_apc
Stonith options:
password: Authentication key
password_script: Script to run to retrieve password
```

警告

method オプションを提供するフェンスエージェントでは、cycle の値を指定しないため、データの破損が発生する可能性があるため、この値は指定しないでください。

1 つのノードのみをフェンスできるフェンスデバイスや、複数のノードをフェンスできるデバイスもあります。フェンスデバイスの作成時に指定するパラメーターは、フェンスデバイスが対応しているか、必要としているかにより異なります。

フェンシングデバイスの作成時に **pcmk_host_list** パラメーターを使用すると、フェンスデバイスで制御されるすべてのマシンを指定できます。

フェンシングデバイスの作成時に **pcmk_host_map** パラメーターを使用すると、フェンスデバイスに関する仕様にホスト名をマッピングできます。

3. フェンスデバイスを作成します。

```
# pcs stonith create clusterfence fence_azure_arm
```

4. 即時かつ完全なフェンシングを確実に行うために、すべてのクラスターノードで ACPI Soft-Off を無効にします。ACPI Soft-Off を無効にする方法については、統合フェンスデバイスで使用する ACPI の無効化 を参照してください。

検証

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
```

以下に例を示します。

```
[root@node01 clouduser]# pcs status
Cluster name: newcluster
Stack: corosync
Current DC: node01 (version 1.1.18-11.el7-2b07d5c5a9) - partition with quorum
Last updated: Fri Feb 23 11:44:35 2018
Last change: Fri Feb 23 11:21:01 2018 by root via cibadmin on node01

3 nodes configured
1 resource configured

Online: [ node01 node03 ]
OFFLINE: [ node02 ]

Full list of resources:

  clusterfence  (stonith:fence_azure_arm):  Started node01

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

2. 前の手順でフェンシングされたノードを起動します。

```
# pcs cluster start <hostname>
```

3. 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。

```
# pcs status
```

以下に例を示します。

```
[root@node01 clouduser]# pcs status
Cluster name: newcluster
Stack: corosync
Current DC: node01 (version 1.1.18-11.el7-2b07d5c5a9) - partition with quorum
Last updated: Fri Feb 23 11:34:59 2018
Last change: Fri Feb 23 11:21:01 2018 by root via cibadmin on node01

3 nodes configured
1 resource configured

Online: [ node01 node02 node03 ]

Full list of resources:

clusterfence    (stonith:fence_azure_arm):  Started node01

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

関連情報

- Fencing in a Red Hat High Availability Cluster（Red Hat ナレッジベース）
- フェンシングデバイスの一般的なプロパティー

4.14. 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

手順

1. 基本ロードバランサーを作成 します。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、デフォルト値が事前に設定されます。**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。

4.15. 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. 全ノードに **nmap-ncat** リソースエージェントをインストールします。

```
# dnf install nmap-ncat resource-agents-cloud
```

   単一ノードで以下の手順を実行します。

2. **pcs** リソースおよびグループを作成します。IPaddr2 アドレスにロードバランサーの FrontendIP を使用します。

```
# pcs resource create resource-name IPaddr2 ip="10.0.0.7" --group cluster-resources-group
```

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

- **pcs status** を実行して結果を表示します。

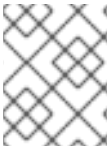
```
[root@node01 clouduser]# pcs status
```

出力例:

```
Cluster name: clusterfence01
Stack: corosync
Current DC: node02 (version 1.1.16-12.el7_4.7-94ff4df) - partition with quorum
Last updated: Tue Jan 30 12:42:35 2018
Last change: Tue Jan 30 12:26:42 2018 by root via cibadmin on node01

3 nodes configured
3 resources configured

Online: [ node01 node02 node03 ]

Full list of resources:

clusterfence (stonith:fence_azure_arm):      Started node01
Resource Group: g_azure
    vip_azure  (ocf::heartbeat:IPaddr2):       Started node02
    lb_azure   (ocf::heartbeat:azure-lb):      Started node02

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

4.16. 共有ブロックストレージの設定

Microsoft Azure 共有ディスクを使用して 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には、次の手順を使用します。この手順はオプションであり、以下の手順では、1 TB の共有ディスクを備えた 3 つの Azure 仮想マシン (3 ノードクラスター) を想定していることに注意してください。



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。
- 以下リソース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が作成されたことを確認します。

```
$ 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 3 回実行します。

```
$ 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>** および **<nic_name>** が指定される点で異なります。

```
$ 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-9.3.0-20200713.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

1. クラスター内の VM ごとに、**ssh** コマンドと VM の IP アドレスを使用して、ブロックデバイスが使用可能であ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リスト表示します。

```
# 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> "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、インスタンスの IP アドレスが **198.51.100.3** のホスト名および共有ディスクボリューム ID が含まれる詳細をリスト表示します。

```
# 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。

関連情報

- クラスターに GFS2 ファイルシステムを設定
- GFS2 ファイルシステムの設定

4.17. 関連情報

- Support Policies for RHEL High Availability Clusters - Microsoft Azure Virtual Machines as Cluster Members
- 高可用性クラスターの設定および管理

第5章 セキュアブートを使用した AZURE での RHEL の設定

Secure Boot は、システムの起動時にプログラムの実行を制御する UEFI (Unified Extensible Firmware Interface)仕様のメカニズムです。セキュアブートは、ブート時にブートローダーとそのコンポーネントのデジタル署名を検証することで、信頼できる許可されたプログラムだけを実行するとともに、許可されていないプログラムのロードを防止します。

Azure プラットフォームで公開されている RHEL イメージでは、セキュアブートが有効になっています。デフォルトでは、Microsoft 証明書を持つ Allowed Signature データベース(**db**)があります。Microsoft Azure では、**Azure Compute Gallery** に新しいイメージバージョンが登録されるときに、UEFI セキュアブート変数にカスタム証明書を追加できます。

5.1. クラウド上の RHEL のセキュアブートについて

セキュアブートは、Unified Extensible Firmware Interface (UEFI) の機能であり、ブートローダーやカーネルなど、デジタル署名された信頼できるプログラムとコンポーネントだけがブート時に実行されるようにします。セキュアブートでは、ハードウェアに保存されている信頼できる鍵に照らしてデジタル署名を検証し、改ざんされたコンポーネントや信頼できないエンティティーによって署名されたコンポーネントが検出されると、ブートプロセスを中止します。これにより、悪意のあるソフトウェアがオペレーティングシステムを侵害するのを防ぎます。

セキュアブートは、信頼できるエンティティーだけがブートチェーン内に存在することを保証するため、Confidential Virtual Machine (CVM) を設定するのに不可欠なコンポーネントです。定められたインターフェイスを通じて特定のデバイスパスへの認証アクセスを提供します。これにより、最新の設定だけが確実に使用され、以前の設定が永続的に上書きされます。さらに、セキュアブートを有効にして Red Hat Enterprise Linux カーネルを起動すると、カーネルが **lockdown** モードに入ります。これにより、信頼できるベンダーによって署名されたカーネルモジュールだけが確実にロードされます。したがって、セキュアブートにより、オペレーティングシステムのブートシーケンスのセキュリティーが向上します。

5.1.1. セキュアブートのコンポーネント

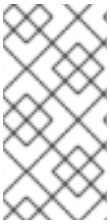
セキュアブートメカニズムは、ファームウェア、署名データベース、暗号鍵、ブートローダー、ハードウェアモジュール、およびオペレーティングシステムで構成されます。以下は、UEFI の信頼済み変数の構成要素です。

- Key Exchange Key データベース (KEK): RHEL オペレーティングシステムと仮想マシンファームウェア間の信頼を確立するための公開鍵の交換。これらの鍵を使用して、Allowed Signature データベース (**db**) と Forbidden Signature データベース (**dbx**) を更新することもできます。

- Platform Key データベース (PK): 仮想マシンファームウェアとクラウドプラットフォーム間の信頼を確立するための自己署名のシングルキーデータベース。また、PK は KEK データベースを更新します。

- Allowed Signature データベース (**db**): バイナリーファイルがシステム上で起動できるかどうかを確認するための証明書またはバイナリーハッシュのリストを保持するデータベース。さらに、**db** からのすべての証明書は、RHEL カーネルの **.platform** キーリングにインポートされます。この機能を使用すると、**lockdown** モードで署名されたサードパーティーのカーネルモジュールを追加およびロードすることができます。

- Forbidden Signature データベース (**dbx**): システム上で起動することが禁止されている証明書またはバイナリーハッシュのリストを保持するデータベース。



注記

バイナリーファイルは、**dbx** データベースと Secure Boot Advanced Targeting (SBAT) メカニズムに照らしてチェックされます。SBAT を使用すると、バイナリーに署名した証明書を有効なままに保ちながら、特定のバイナリーの古いバージョンを無効にできます。

5.1.2. クラウド上の RHEL におけるセキュアブートの段階

RHEL インスタンスが Unified Kernel Image (UKI) モードで起動し、セキュアブートが有効な場合、RHEL インスタンスは次の順序でクラウドサービスインフラストラクチャーとやり取りします。

1. **初期化**: RHEL インスタンスが起動すると、クラウドでホストされているファームウェアが最初に起動し、セキュアブートメカニズムを実装します。

2. **変数ストアの初期化**: ファームウェアは、変数ストアから UEFI 変数を初期化します。このストアは、ブートプロセスと実行時の操作のためにファームウェアが管理する必要がある情報専用のストレージ領域です。RHEL インスタンスが初めて起動すると、ストアは仮想マシンイメージに関連付けられたデフォルト値によって初期化されます。

3. **ブートローダー**: ブート時に、ファームウェアは第 1 段階のブートローダーをロードします。x86 UEFI 環境の RHEL インスタンスの場合、第 1 段階のブートローダーは shim です。shim ブートローダーは、ブートプロセスの次の段階を認証して読み込み、UEFI と GRUB 間の橋渡し役として機能します。

    a. RHEL の shim x86 バイナリーは、現在 **Microsoft Corporation UEFI CA 2011** Microsoft 証明書によって署名されています。これは、Allowed Signature データベース (**db**) にデフォルトの Microsoft 証明書が含まれているさまざまなハードウェアおよび仮想化プラットフォーム上で、RHEL インスタンスをセキュアブート対応モードで起動できるようにするためです。

    b. shim バイナリーは、Red Hat Secure Boot CA と、必要に応じて Machine Owner Key (**MOK**) を使用して、信頼済み証明書のリストを拡張します。

4. **UKI**: shim バイナリーは、RHEL UKI (**kernel-uki-virt** パッケージ) をロードします。UKI は、**redhat-sb-certs** パッケージにある対応する証明書 (x86_64 アーキテクチャーでは **Red Hat Secure Boot Signing 504**) によって署名されます。この証明書は Red Hat Secure Boot CA によって署名されているため、チェックに合格します。

5. **UKI アドオン**: UKI の **cmdline** 拡張機能を使用するために、RHEL カーネルが、拡張機能の署名を **db**、**MOK**、および shim に同梱されている証明書と照らしてチェックして、その拡張機能がオペレーティングシステムのベンダーである RHEL またはユーザーによって署名されていることを確認します。

RHEL カーネルがセキュアブートモードで起動すると、カーネルは **lockdown** モードになります。**lockdown** に入ると、RHEL カーネルは **db** の鍵を **.platform** キーリングに追加し、**MOK** の鍵を **.machine** キーリングに追加します。カーネルのビルドプロセス中に、**kernel-modules-core**、**kernel-modules**、**kernel-modules-extra** などの標準 RHEL カーネルモジュールが、秘密鍵と公開鍵で構成される一時的な鍵で署名されます。各カーネルビルドが完了すると、この秘密鍵はサードパーティーモジュールの署名には使用できなくなります。この署名のためには、**db** および **MOK** の証明書を使用できます。

関連情報

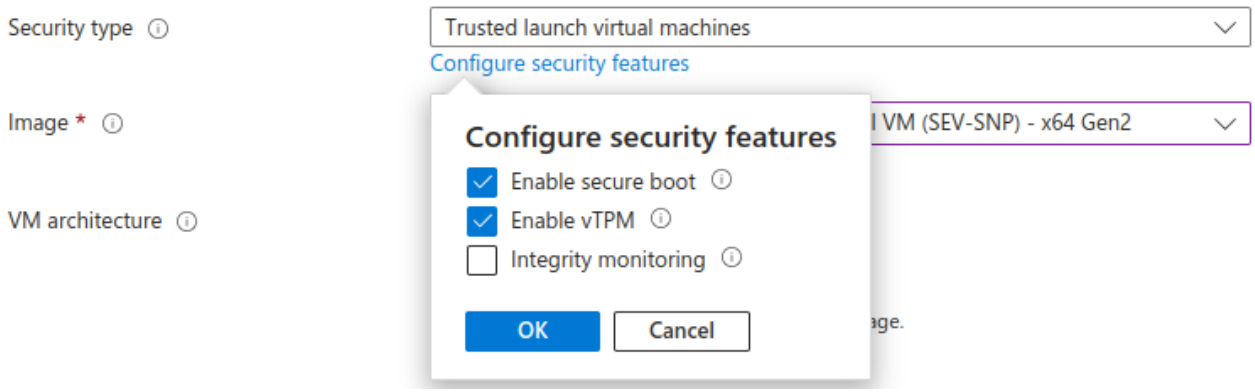
- SBAT メカニズム

- クラウドでの Red Hat Enterprise Linux およびセキュアブート

5.2. セキュアブートを使用した AZURE での RHEL 仮想マシンの設定

Azure クラウドプラットフォーム上の Red Hat Enterprise Linux インスタンスにセキュアなオペレーティングシステムの起動プロセスがあることを確認するには、セキュアブート を使用します。カスタム RHEL Azure イメージが登録されると、そのイメージはセキュアブート用に事前保存された Unified Extensible Firmware Interface (UEFI) 変数で構成されます。これにより、RHEL イメージから起動されたすべてのインスタンスが、最初の起動時に必要な変数を使用してセキュアブートメカニズムを使用できるようになります。

Microsoft Azure は、Trusted Launch 仮想マシンによるセキュアブートをサポートしています。これらの仮想マシンは、ルートキットやブートキットから保護するためのセキュリティーメカニズムを提供するとともに、Virtual Trusted Platform Manager (vTPM) などの追加機能も提供します。GUI を使用してインスタンスを作成する場合、**Enable secure boot** オプションは、**Configure security features** 設定の下にあります。



前提条件

- パッケージをインストールしている。
  - **python3**
  - **openssl**
  - **efivar**
  - **keyutils**
  - **python3-virt-firmware**
- **azure-cli** ユーティリティーがインストールされている。詳細は、Installing the Azure CLI on Linux を参照してください。

手順

1. **openssl** ユーティリティーを使用してカスタム証明書 **custom_db.cer** を生成します。

```
$ openssl req -quiet \
-newkey rsa:4096 \
-nodes -keyout custom_db.key \
-new -x509 \
-sha256 -days 3650 \
```

```
-subj "/CN=Signature Database key/" \
--outform DER \
-out custom_db.cer
```

2. 証明書を **base64** エンコード形式に変換します。

```
$ echo base64 -w0 custom_db.cer

MIIFIjCCAwqgAwIBAgITNf23J4k0d8c0NR ....
```

3. 新しい Azure Compute Gallery イメージバージョンを登録するための **azure-example-template.json** Azure Resource Manager (ARM) ファイルを作成して編集します。

```
$ vi azure-example-template.json

{
   "$schema": "https://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
   "contentVersion": "1.0.0.0",
   "resources": [
      {
         "type": "Microsoft.Compute/galleries/images/versions",
         "apiVersion": "2023-07-03",
         "name": "<your compute gallery/your image definition/version>",
         "location": "<location of the VHD>",
         "properties": {
            "storageProfile": {
               "osDiskImage": {
                  "source": {
                     "id": "<your-storage-account-id>",
                     "uri": "<url-with-the-vhd>"
                  },
                  "hostCaching": "ReadOnly"
               }
            },
            "securityProfile": {
               "uefiSettings": {
                  "signatureTemplateNames": [
                     "MicrosoftUefiCertificateAuthorityTemplate"
                  ],
                  "additionalSignatures": {
                     "db": [
                        {
                           "type": "x509",
                           "value": [
                              "<base64 of custom_db.cer>"
                           ]
                        }
                     ]
                  }
               }
            }
         }
```

```
        }
    ]
}
```

4. **azure-cli** ユーティリティーを使用して、イメージバージョンを登録します。

```
$ az deployment group create --name <example-deployment> \
--resource-group <example-resource-group> \
--template-file <example-template.json>
```

5. Azure ポータルからインスタンスを再起動します。

検証

1. 新しく作成された RHEL インスタンスでセキュアブートが有効になっているか確認します。

```
$ mokutil --sb-state
SecureBoot enabled
```

2. **keyctl** ユーティリティーを使用して、カスタム証明書のカーネルキーリングを確認します。

```
$ sudo keyctl list %:.platform

keys in keyring:
...
586621657: ---lswrv   0   0
asymmetric: Signature Database key: f064979641c24e1b935e402bdbc3d5c4672a1acc
...
```

関連情報

- Trusted Launch in Azure

第6章 INTEL TDX を使用したパブリッククラウドプラットフォームでの RHEL の設定

Intel Trust Domain Extensions (TDX)は、Confidential Virtual Machine (CVM)のセキュリティータイプのことです。これは、仮想マシンに安全で分離された環境を提供します。このアプローチは、以前のテクノロジーである Intel Software Guard Extensions (SGX)の進歩です。

SGX は、ハイパーバイザーとクラウドサービスプロバイダーからの仮想マシンの分離を、enclaves と呼ばれる安全なメモリー領域を作成して提供します。保存されたアプリケーションコードは、エンリート内に保存されたメモリーおよびデータにアクセスでき、外部エンティティーからはアクセスできなくなります。

TDX は、Trusted Domains (TD)と呼ばれるハードウェア分離仮想マシンを作成します。これにより、メモリーにアクセスし、TD 仮想マシンのみが仮想マシンマネージャー(VMM)、ハイパーバイザー、その他の仮想マシン、およびホストから分離されます。これにより、ハイパーバイザー、CPU、TD VM のリソースを使用する一方で、データの機密性と整合性を維持することで安全性が保たれます。

SGX と TDX の主な違いは、SGX はアプリケーションレベルで動作し、TDX はハイパーバイザーのアクセスを制限することで仮想化レベルで機能する点です。

注記

パブリッククラウドプラットフォームに Red Hat Enterprise Linux (RHEL)をデプロイする前に、特定の RHEL インスタンスタイプのサポートステータスと認定について、対応するクラウドサービスプロバイダーに常に確認してください。

6.1. INTEL TDX セキュアブートプロセスについて

1. **初期化と測定**: TDX 対応ハイパーバイザーは、仮想マシンの初期状態を設定します。このハイパーバイザーは、ファームウェアバイナリーファイルを仮想マシンメモリーに読み込み、初期レジスター状態を設定します。Intel プロセッサーは、仮想マシンの初期状態を測定し、仮想マシンの初期状態を確認する詳細を提供します。

2. **ファームウェア**: 仮想マシンは UEFI ファームウェアを開始します。ファームウェアには、ステートフルまたはステートレスの Virtual Trusted Platform Module (vTPM)実装が含まれる可能性があります。stateful vTPM は、VM の再起動と移行後も永続的な暗号化状態を維持しますが、ステートレス vTPM は、永続性のない各 VM セッションの新しい暗号化状態を生成します。仮想マシンの権限レベル(VMPL)テクノロジーは、vTPM をゲストから分離します。VMPL は、異なる仮想マシンコンポーネントとハイパーバイザーの間で、ハードウェアで強制された権限の分離を提供します。

3. **vTPM**: クラウドサービスプロバイダーによっては、ステートフル vTPM 実装によっては、UEFI ファームウェアがリモートアテステーションを実行して、vTPM の永続的な状態を復号化する可能性があります。vTPM は、セキュアブート状態、ブートアーティファクトの署名に使用される証明書、UEFI バイナリーハッシュなど、ブートプロセスに関するデータを収集します。

4. **shim** : UEFI ファームウェアが初期化プロセスを終了すると、拡張ファームウェアインターフェイス(EFI)システムパーティションを検索します。次に、UEFI ファームウェアは、そこから最初のステージブートローダーを検証し、実行します。RHEL の場合、これは **shim** です。**shim** プログラムでは、Microsoft 以外のオペレーティングシステムが EFI システムパーティションから第 2 段階のブートローダーをロードすることができます。

    a. **shim** は Red Hat 証明書を使用して、2 番目のステージブートローダー(**grub**)または Red Hat Unified Kernel Image (UKI)を検証します。

b. **GRUB** または **UKI** は、Linux カーネルと initramfs、およびカーネルコマンドラインをアンパックして検証し、実行します。このプロセスにより、Linux カーネルが信頼できるセキュアな環境に読み込まれるようになります。

5. **initramfs**: initramfs では、完全なディスク暗号化テクノロジーの場合、vTPM 情報は暗号化されたルートパーティションを自動的にアンロックします。

   a. ルートボリュームが利用可能になると、**initramfs** はそこで実行フローを転送します。

6. アテステーション：VM テナントはシステムにアクセスし、リモート認証を実行して、アクセスされた VM が改ざんされていない Confidential Virtual Machine (CVM)であることを確認できます。**認証は、Intel プロセッサーと vTPM からの情報に基づいて実行されます。このプロセスは、RHEL インスタンスと Intel プロセッサーの初期 CPU およびメモリーの状態の信頼性および信頼性を確認します。**

7. **TEE: このプロセスは、Trusted Execution Environment (TEE)を作成し、仮想マシンの起動が信頼できるセキュアな環境にあることを確認します。**

6.2. INTEL TDX を使用した AZURE での RHEL 仮想マシンの設定

Intel TDX を使用すると、信頼できるドメイン(TD)として知られるハードウェア支援型 VM を作成できます。これにより、仮想マシンのみがそのリソースにアクセスでき、ハイパーバイザーやホストからアクセスできなくなります。

前提条件

- **openssh パッケージおよび openssh -clients パッケージがインストールされている。**
- **Azure CLI ユーティリティーをインストールしている。詳細は、Installing the Azure CLI on Linux を参照してください。**
- **サポートされている Azure インスタンスタイプから RHEL インスタンスを起動している。詳細は、Azure Confidential VM options を参照してください。**

手順

1. **azure cli ユーティリティーを使用して Azure にログインします。**

```
$ az login
```

2. **選択したアベイラビリティーゾーンの Azure リソースグループを作成します。**

```
$ az group create --name <example_resource_group> --location westeurope
```

3. **TDX を有効にして RHEL インスタンスをデプロイします（例： Standard_DC2eds_v5 インスタンスタイプ）。**

```
$ az vm create --resource-group <example_resource_group> \
--name <example_rhel_instance> \
--image <"RedHat:rhel-cvm:9_5_cvm:latest"> \
--size <Standard_DC2eds_v5> \
--admin-username <example_azure_user> \
--generate-ssh-keys \
--security-type ConfidentialVM \
--os-disk-security-encryption-type DiskWithVMGuestState
```

4. **RHEL インスタンスに接続します。**

```
$ ssh <example_azure_user>@<example_ip_address_of_the_instance>
```

検証

- **カーネルログをチェックして、TDX のステータスを確認します。**

```
$ dmesg | grep -i tdx
```

```
...
[    0.733613] Memory Encryption Features active: Intel TDX
[    4.320222] systemd[1]: Detected confidential virtualization tdx.
[    5.977432] systemd[1]: Detected confidential virtualization tdx.
...
```

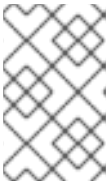
- **RHEL インスタンス設定のメタデータを確認します。**

```
$ az vm show --resource-group <example_resource_group> \
--name <example_rhel_instance> \
--query "securityProfile.enableTrustedDomainExtensions" \
--output json
```

第7章 AMD SEV SNP を使用したパブリッククラウドプラットフォームでの RHEL の設定

AMD Secure Encrypted Virtualization with Secure Nested Paging (SEV-SNP)は、仮想マシンの整合性ベースの攻撃を防ぎ、メモリー整合性違反の危険を低減することを目的としています。セキュアブートプロセスの場合、AMD プロセッサーは 3 つのハードウェアベースのセキュリティーメカニズムを提供します。SEV (Secure Encrypted Virtualization)、SEV-ES (SEV-ES)、および SEV Secure Nested Paging (SEV-SNP)です。

- SEV: SEV メカニズムは、ハイパーバイザーが仮想マシンデータにアクセスできないように、仮想マシン(VM)メモリーを暗号化します。
- SEV-ES: SEV with Encrypted State (SEV-ES)は、CPU レジスター状態を暗号化することで SEV を拡張します。このメカニズムにより、ハイパーバイザーが仮想マシンの CPU レジスターにアクセスまたは変更できなくなります。ハイパーバイザーと仮想マシンを分離しているにもかかわらず、メモリー整合性攻撃に対して脆弱です。
- SEV-SNP: SEV-SNP は、仮想マシンの暗号化と共にメモリー整合性保護を追加する SEV-ES の拡張機能です。このメカニズムにより、ハイパーバイザーが仮想マシンメモリーアクセスをリダイレクトするためにページテーブルを変更することが防止され、リプレイ攻撃やメモリーの改ざんを防ぎます。



注記

パブリッククラウドプラットフォームに Red Hat Enterprise Linux (RHEL)をデプロイする前に、特定の RHEL インスタンスタイプのサポートステータスと認定について、対応するクラウドサービスプロバイダーに常に確認してください。

7.1. SEV-SNP のプロパティー

- セキュアなプロセッサー: AMD **EPYC** プロセッサーは、セキュアプロセッサー(SP)サブシステムを統合します。AMD SP は、鍵と暗号化操作を管理する専用のハードウェアコンポーネントです。
- メモリーの整合性: 仮想化と分離を管理するため、メモリー管理ユニット(MMU)はページテーブルを利用して仮想アドレスをゲストの物理アドレスに変換します。SEV-SNP は、ネストされたページテーブルを使用して、ゲスト物理アドレスをホスト物理アドレスに変換します。ネストされたページテーブルを定義すると、ハイパーバイザーまたはホストがページテーブルを変更して異なるページにアクセスすることができず、メモリーの整合性が保護されます。SEV-SNP は、この方法を使用して、再生攻撃や仮想マシンメモリーへの悪意のある変更に対する保護を提供します。
- メモリー の暗号 化：AMD **EPYC** プロセッサーは、メモリー暗号化キーを非表示にします。メモリー暗号化キーは、ホストと仮想マシンの両方で非表示にされます。
- 検証用のテストレポート: 承認された暗号化形式の RHEL インスタンス情報に関する CPU 生成レポート。このプロセスは、RHEL インスタンスと AMD プロセッサーの初期 CPU およびメモリーの状態の信頼性および信頼性を確認します。

注記

ハイパーバイザーが仮想マシンのプライマリーメモリーと CPU レジスター状態を作成する場合でも、VM の初期化後も、ハイパーバイザーが非表示になり、ハイパーバイザーからはアクセスできなくなります。

7.2. AMD SEV SNP セキュアブートプロセスについて

1. 初期化と測定: SEV-SNP 対応ハイパーバイザーは、仮想マシンの初期状態を設定します。このハイパーバイザーは、ファームウェアバイナリーを仮想マシンメモリーに読み込み、初期レジスター状態を設定します。AMD Secure Processor (SP)は、仮想マシンの初期状態を測定し、仮想マシンの初期状態を検証する情報を提供します。

2. ファームウェア: 仮想マシンは UEFI ファームウェアを開始します。ファームウェアには、ステートフルまたはステートレスの Virtual Trusted Platform Module (vTPM)実装が含まれる可能性があります。stateful vTPM は、VM の再起動と移行後も永続的な暗号化状態を維持しますが、ステートレス vTPM は、永続性のない各 VM セッションの新しい暗号化状態を生成します。仮想マシンの権限レベル(VMPL)テクノロジーは、vTPM をゲストから分離します。VMPL は、異なる仮想マシンコンポーネントとハイパーバイザーの間で、ハードウェアで強制された権限の分離を提供します。

3. vTPM: クラウドサービスプロバイダーによっては、ステートフル vTPM 実装によっては、UEFI ファームウェアがリモートアテステーションを実行して、vTPM の永続的な状態を復号化する可能性があります。

    a. vTPM は、セキュアブート状態、ブートアーティファクトの署名に使用される証明書、UEFI バイナリーハッシュなどのブートプロセスに関するファクトも測定します。

4. shim: UEFI ファームウェアが初期化プロセスを終了すると、拡張ファームウェアインターフェイス(EFI)システムパーティションを検索します。次に、UEFI ファームウェアは、そこから最初のステージブートローダーを検証し、実行します。RHEL の場合、これは **shim** です。**shim** プログラムでは、Microsoft 以外のオペレーティングシステムが EFI システムパーティションから第 2 段階のブートローダーをロードすることができます。

    a. **shim** は Red Hat 証明書を使用して、2 番目のステージブートローダー(**grub**)または Red Hat Unified Kernel Image (UKI)を検証します。

    b. **GRUB** または **UKI** は、Linux カーネルと初期 RAM ファイルシステム(**initramfs**)、およびカーネルコマンドラインをアンパックし、検証し、実行します。このプロセスにより、Linux カーネルが信頼できるセキュアな環境に読み込まれるようになります。

5. initramfs: **initramfs** では、ディスク暗号化技術が完全な場合に、vTPM 情報は暗号化されたルートパーティションを自動的にアンロックします。

    a. ルートボリュームが利用可能になると、**initramfs** により実行フローがルートボリュームに転送されます。

6. アテステーション：VM テナントはシステムにアクセスし、リモート認証を実行して、アクセスされた VM が改ざんされていない Confidential Virtual Machine (CVM)であることを確認できます。テストは、AMD SP および vTPM からの情報に基づいて実行されます。このプロセスは、RHEL インスタンスと AMD プロセッサーの初期 CPU およびメモリーの状態の信頼性および信頼性を確認します。

7. TEE: このプロセスは、Trusted Execution Environment (TEE)を作成し、仮想マシンの起動が信頼できるセキュアな環境にあることを確認します。

7.3. AMD SEV SNP を使用した AZURE での RHEL 仮想マシンの設定

AMD Secure Encrypted Virtualization with Secure Nested Paging (SEV-SNP)は、Azure 仮想マシン(VM)上の Red Hat Enterprise Linux (RHEL)用の Confidential Virtual Machine (CVM)テクノロジーのセキュリティータイプです。また、AMD **EPYC** プロセッサーファミリーにのみ利用可能です。SEV-

SNP は、信頼できるブート環境を提供し、ハイパーバイザーとクラウドプロバイダーのデータにアクセスできないようにプロセス全体が保護され、保護されます。

前提条件

- openssh パッケージおよび openssh -clients パッケージがインストールされている。
- Azure CLI ユーティリティーをインストールしている。詳細は、Azure CLI のインストール を参照してください。
- インスタンスは、前述の Azure インスタンスタイプからのみ起動しました。詳細は Supported VM size for CVM を 参照してください。

手順

1. Azure CLI ユーティリティーを使用して Azure にログインします。

```
$ az login
```

2. 選択したアベイラビリティーゾーンの Azure リソースグループを作成します。

```
$ az group create --name <example_resource_group> --location eastus
```

3. SEV-SNP を使用して RHEL インスタンスをデプロイします（例： Standard_DC4as_V5 インスタンスタイプ）。

```
$ az vm create --resource-group <example_resource_group> \
--name <example-rhel-9-instance> \
--image <"RedHat:rhel-cvm:9_5_cvm:latest"> \
--size <Standard_DC4as_V5> \
--admin-username <example_azure_user> \
--generate-ssh-keys \
--security-type ConfidentialVM \
--os-disk-security-encryption-type DiskWithVMGuestState
```

4. RHEL インスタンスに接続します。

```
$ ssh <example_azure_user>@<example_ip_address_of_VM>
```

検証

- カーネルログをチェックして、SEV-SNP のステータスを確認します。

```
$ sudo dmesg | grep -i sev
```

```
...
[    0.547223] Memory Encryption Features active: AMD SEV
[    4.843171] kvm-guest: setup_efi_kvm_sev_migration : EFI live migration variable not found
...
```

第8章 RHEL システムロールを使用した HPC クラスターの AZURE へのデプロイ

Microsoft Azure の高パフォーマンスコンピューティング(HPC)ワークロードには、最適なパフォーマンスとスケーラビリティーのために特殊な設定が必要です。HPC RHEL システムロールは、InfiniBand サポート、パフォーマンスチューニング、必要なライブラリーなど、HPC 固有の最適化を使用した RHEL イメージの設定を自動化します。

HPC 対応イメージを設定した後、仮想マシンを一般化し、Azure Compute Gallery で再利用可能なイメージバージョンを作成できます。これらのイメージは、Azure CycleCloud を使用して Microsoft Azure に HPC クラスターをデプロイするための基盤です。これは、Slurm ワークロードマネージャーと統合して計算ジョブをスケジュールおよび管理するクラスターオーケストレーションツールです。環境モジュールは、HPC クラスターノード間で複数のソフトウェアバージョンとその依存関係を管理するための柔軟なフレームワークを提供します。

8.1. HPC RHEL システムロールを使用した HPC 用の RHEL AZURE 仮想マシンの設定

カスタマイズされた Red Hat Enterprise Linux (RHEL)イメージで高パフォーマンスコンピューティング(HPC) RHEL システムロールを設定するには、**cloud-init** ユーティリティーを使用できます。**cloud-init** を使用すると、Ansible コレクションの設定を自動化し、Microsoft Azure で Ansible Playbook を実行できます。

以下のいずれかの方法を使用して、カスタマイズされた RHEL イメージで HPC RHEL システムロールを設定します。

8.1.1. Azure ポータルを使用した RHEL HPC 仮想マシンの設定

ansible-core ユーティリティーで Ansible を使用すると、イメージ構築プロセス中に RHEL システムロールを適用することにより、Azure のカスタム RHEL イメージの設定を自動化できます。**cloud-init** のようなユーティリティーは、HPC (High-performance computing) RHEL システムロール設定を組み込み、Azure にデプロイする前に HPC RHEL イメージを作成および設定します。

前提条件

- アクティブな Azure クラウドサブスクリプションがある。

手順

1. Azure Console に移動します。
2. Virtual Machines → Create Virtual Machineをクリックします。
3. Basics タブから仮想マシンについて、以下の設定を選択します。
   a. Virtual machine nameフィールドに仮想マシン名を入力します。
   b. セキュリティータイプ: Standard
   c. Image → See All Images → Search forRed Hat Enterprise Linux (RHEL) for High Performance Computing (HPC) for High Performance Computing (HPC) → SelectRed Hat Enterprise Linux for HPC 9.6 VM - x64 Gen2
   d. 仮想マシンアーキテクチャー: x64

e. サイズ: Standard_NC4as_T4_v3 - 4 vcpus、28 GiB メモリー

注記

最適なパフォーマンスを得るには、NC や ND シリーズなどの GPU 用に最適化された仮想マシンのみを使用してください。詳細は、Virtual machine sizes in Azure を参照してください。

4. Advanced タブに移動し、Custom data フィールドに以下の詳細を入力します。

```
#cloud-config
#Please check RHEL HPC Ansible system role documentation for all available options

write_files:
 - path: /root/hpc_full_install.yaml
   permissions: 644
   content: |
     ---
     - name: Install and configure HPC
       hosts: localhost
       become: true
       vars:
         hpc_reboot_ok: false
         hpc_update_all_packages: true
         hpc_manage_firewall: true
       roles:
         - redhat.rhel_system_roles.hpc
 - path: /etc/dnf/azure-rhel9-eus.config
   permissions: 644
   content: |
     [rhui-microsoft-azure-rhel9]
     name=Microsoft Azure RPMs for Red Hat Enterprise Linux 9 (rhel9-eus)
     baseurl=https://rhui4-1.microsoft.com/pulp/repos/unprotected/microsoft-azure-
rhel9-eus
     enabled=1
     gpgcheck=1
     sslverify=1
     gpgkey=/etc/pki/rpm-gpg/RPM-GPG-KEY-microsoft-azure-release
 - path: /etc/dnf/vars/releasever
   permissions: 644
   content: |
     9.6

# Run custom commands
runcmd:
  # lock VM to RHEL9.6 and enable EUS channels
  # https://learn.microsoft.com/en-us/azure/virtual-machines/workloads/redhat/redhat-
rhui
  - dnf --assumeyes --disablerepo='*' remove "rhui-azure-rhel9"
  - dnf --assumeyes --config /etc/dnf/azure-rhel9-eus.config install rhui-azure-rhel9-eus
  - dnf --assumeyes clean all
  - dnf --assumeyes install rhel-system-roles
```

5. Review + create ボタンをクリックして、指定の設定で仮想マシンを作成します。

6. **Azure コンソールで、仮想マシンが正常にデプロイされ、使用できる状態であることを確認します。**
7. **Go to resourceボタンをクリックします。**
8. **パブリック IP アドレスをコピーします。**
9. **Azure コンソールで、VM が実行されていることを確認します。**
10. **仮想マシンに接続します。**

```
$ ssh -i ~/.ssh/azure_hpc <example_azureuser>@<192.0.2.101>
```

11. **仮想マシンのステータスを確認します。**

```
$ sudo cloud-init status --wait
```

12. **準備ができたら、HPC RHEL システムロールを実行します。**

```
$ sudo ANSIBLE_LOG_PATH=/var/log/ansible_hpc_full_install.log ansible-playbook
/root/hpc_full_install.yaml --verbose
```

13. **仮想マシンを再起動します。**

重要

このフェーズでは、HPC RHEL システムロール設定が最終終了するのを待ちます。

検証

1. **SSH 経由で仮想マシンに接続します。**

```
$ ssh -i <example_private_key.pem> <example_azureuser>@<192.0.2.101>
```

2. **インストールされているパッケージのリストを確認します。**

```
$ sudo dnf list installed| grep -i -E 'nvidia-driver|cuda-toolkit|nccl|fabric-
manager|rdma|openmpi'
```

```
cuda-toolkit-12-9.x86_64                12.9.1-1           @nvidia-cuda
cuda-toolkit-12-9-config-common.noarch  12.9.79.1          @nvidia-cuda
cuda-toolkit-12-config-common.noarch    12.9.79.1          @nvidia-cuda
cuda-toolkit-config-common.noarch       12.9.79.1          @nvidia-cuda
libnccl.x86_64                          2.27.5-1+cuda12.9  @nvidia-cuda
libnccl-devel.x86_64                    2.27.5-1+cuda12.9  @nvidia-cuda
librdma.x86_64                          54.0.1-e19         @rhel-9-for-x86_64-baseos-rhui-rpms
nvidia-driver.x86_64                    3:575.57.08-1.e19  @nvidia-cuda
nvidia-driver-cuda.x86_64               3:575.57.08-1.e19  @nvidia-cuda
nvidia-driver-cuda-libs.x86_64          3:575.57.08-1.e19  @nvidia-cuda
nvidia-driver-libs.x86_64               3:575.57.08-1.e19  @nvidia-cuda
nvidia-fabric-manager.x86_64            575.57.08-1        @nvidia-cuda
openmpi.x86_64                          2:4.1.7-7.e19      @rhel-9-for-x86_64-appstream-rhui-rpms
openmpi-devel.x86_64                    2:4.1.7-7.e19      @rhel-9-for-x86_64-appstream-rhui-
```

```
rpms
rdma-core.x86_64                    54.0.1-e19          @rhel-9-for-x86_64-baseos-rhui-rpms
```

3. **インストールされた Lmod 環境モジュールを確認します。**

```
$ ml available
```

```
------------------------/usr/share/modulefiles-------------------------
   mpi/hpcx-2.24.1-pmix-4.2.9    mpi/openmpi-5.0.8-cuda12-gpu (L,D)
   mpi/hpcx-2.24.1               pmix/pmix-4.2.9              (L)
   mpi/openmpi-x86_64

-----------------/usr/share/lmod/lmod/modulefiles/Core----------------
   lmod    settarg
```

ここでは、以下のようになります。

- **L: モジュールがロードされている。**
- **d: デフォルトモジュール**

次のステップ

- **イメージ作成のための Azure 仮想マシンの一般化**

8.1.2. Azure CLI を使用した RHEL HPC 仮想マシンの設定

ansible-core ユーティリティーで Ansible を使用すると、イメージのビルドプロセス中に RHEL システムロールを適用することにより、Azure のカスタム Red Hat Enterprise Linux (RHEL)イメージの設定を自動化できます。cloud-init や Azure CLI などのユーティリティーは、Azure にデプロイする前に HPC RHEL イメージを作成および設定するための高パフォーマンスコンピューティング(HPC) RHEL システムロールを管理します。

前提条件

- **アクティブな Azure クラウドサブスクリプションがある。**

手順

1. **Azure ポータルに接続します。**

```
$ az login
```

2. **キーペアを作成します。**

```
$ ssh-keygen -t ed25519 -b 3072 -C "<azureuser@hpc>" -f ~/.ssh/azure_hpc
```

3. **以下の詳細で user-data.yml ファイルを編集します。**

```
$ vi user-data.yml
```

```
#cloud-config
```

```
# Please check RedHat HPC Ansible system role documentation for all available
options

write_files:
 - path: /root/hpc_full_install.yaml
   permissions: 644
   content: |
     ---
     - name: Install and configure HPC
       hosts: localhost
       become: true
       vars:
         hpc_reboot_ok: false
         hpc_update_all_packages: true
         hpc_manage_firewall: true
       roles:
         - redhat.rhel_system_roles.hpc
 - path: /etc/dnf/azure-rhel9-eus.config
   permissions: 644
   content: |
     [rhui-microsoft-azure-rhel9]
     name=Microsoft Azure RPMs for Red Hat Enterprise Linux 9 (rhel9-eus)
     baseurl=https://rhui4-1.microsoft.com/pulp/repos/unprotected/microsoft-azure-
rhel9-eus
     enabled=1
     gpgcheck=1
     sslverify=1
     gpgkey=/etc/pki/rpm-gpg/RPM-GPG-KEY-microsoft-azure-release
 - path: /etc/dnf/vars/releasever
   permissions: 644
   content: |
     9.6

# Run custom commands
runcmd:
  # lock VM to RHEL9.6 and enable EUS channels
  # https://learn.microsoft.com/en-us/azure/virtual-machines/workloads/redhat/redhat-
rhui
  - dnf --assumeyes --disablerepo='*' remove "rhui-azure-rhel9"
  - dnf --assumeyes --config /etc/dnf/azure-rhel9-eus.config install rhui-azure-rhel9-eus
  - dnf --assumeyes clean all
  - dnf --assumeyes install rhel-system-roles
```

4. リソースグループを作成します。

```
$ az group create --name <example_vm_resource_group>
```

5. アカウントに関連するイメージの契約条件を選択し、同意します。

- 北アメリカ(NA)またはグローバルアカウントの場合は、以下を使用します。

```
$ az vm image terms accept --urn "redhat:rh-rhel-hpc:rh-rhel-hpc96:latest"
```

- ヨーロッパの場合は、中東 および アフリカ(EMEA)アカウントは以下を使用します。

```
$ az vm image terms accept --urn "redhat-limited:rh-rhel-hpc:rh-rhel-hpc96:latest"
```

6. **最後のステップで指定した設定に基づいて、イメージを作成します。**

```
$ az vm create \
--resource-group <example_vm_resource_group> \
--name <example_vm_name> \
--image <example_rhel_hpc_image_urn> \
--size <Standard_NC4as_T4_v3> \
--admin-username <example_azureuser> \
--ssh-key-values ~/.ssh/azure_hpc.pub \
--custom-data user-data.yaml \
--security-type Standard \
--public-ip-address-dns-name <example_vm_name>-$(openssl rand -hex 4) \
--tags owner=$USER project=hpc
```

7. **仮想マシンが実行されている場合は、Azure Console を確認します。**
8. **SSH 経由で仮想マシンに接続します。**

```
$ ssh -i ~/.ssh/azure_hpc <example_azureuser>@<192.0.2.101>
```

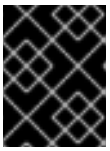
9. **仮想マシンのステータスを確認します。**

```
$ sudo cloud-init status --wait
```

10. **準備ができたら、HPC RHEL システムロールを実行します。**

```
$ sudo ANSIBLE_LOG_PATH=/var/log/ansible_hpc_full_install.log ansible-playbook
/root/hpc_full_install.yaml --verbose
```

11. **仮想マシンを再起動します。**



重要

このフェーズでは、HPC RHEL システムロール設定が最終終了するのを待ちます。

検証

1. **SSH 経由で仮想マシンに接続します。**

```
$ ssh -i <example_private_key.pem> <example_azureuser>@<192.0.2.101>
```

2. **インストールされているパッケージのリストを確認します。**

```
$ sudo dnf list installed| grep -i -E 'nvidia-driver|cuda-toolkit|nccl|fabric-
manager|rdma|openmpi'
```

```
cuda-toolkit-12-9.x86_64                12.9.1-1          @nvidia-cuda
cuda-toolkit-12-9-config-common.noarch  12.9.79.1         @nvidia-cuda
cuda-toolkit-12-config-common.noarch    12.9.79.1         @nvidia-cuda
```

```
cuda-toolkit-config-common.noarch       12.9.79.1          @nvidia-cuda
libnccl.x86_64                          2.27.5-1+cuda12.9  @nvidia-cuda
libnccl-devel.x86_64                    2.27.5-1+cuda12.9  @nvidia-cuda
librdma.x86_64                          54.0.1-e19         @rhel-9-for-x86_64-baseos-rhui-rpms
nvidia-driver.x86_64                    3:575.57.08-1.e19  @nvidia-cuda
nvidia-driver-cuda.x86_64               3:575.57.08-1.e19  @nvidia-cuda
nvidia-driver-cuda-libs.x86_64          3:575.57.08-1.e19  @nvidia-cuda
nvidia-driver-libs.x86_64               3:575.57.08-1.e19  @nvidia-cuda
nvidia-fabric-manager.x86_64            575.57.08-1        @nvidia-cuda
openmpi.x86_64                          2:4.1.7-7.e19      @rhel-9-for-x86_64-appstream-rhui-rpms
openmpi-devel.x86_64                    2:4.1.7-7.e19      @rhel-9-for-x86_64-appstream-rhui-
rpms
rdma-core.x86_64                        54.0.1-e19         @rhel-9-for-x86_64-baseos-rhui-rpms
```

3. **インストールされた Lmod 環境モジュールを確認します。**

```
$ ml available
```

```
------------------------/usr/share/modulefiles-------------------------
   mpi/hpcx-2.24.1-pmix-4.2.9    mpi/openmpi-5.0.8-cuda12-gpu (L,D)
   mpi/hpcx-2.24.1               pmix/pmix-4.2.9              (L)
   mpi/openmpi-x86_64

-----------------/usr/share/lmod/lmod/modulefiles/Core----------------
   lmod    settarg
```

ここでは、以下のようになります。

- **L: モジュールがロードされている。**
- **d: デフォルトモジュール**

関連情報

- **イメージ作成のための Azure 仮想マシンの一般化**

8.2. イメージ作成のための AZURE 仮想マシンの一般化

仮想マシン(VM)を一般化することにより、イメージのバージョン管理用のテンプレートまたはベースイメージとして使用する仮想マシンを準備します。このプロセスでは、特定のデータを削除し、VM を停止し、リソースの割り当てを解除し、仮想マシンを一般化する必要があります。汎用イメージを使用すると、同じイメージから複数のイメージバージョンを作成できます。

前提条件

- **RHEL HPC イメージがすでに設定されている。詳細は、HPC RHEL システムロールを使用した RHEL HPC イメージの設定 を参照してください。**

手順

1. **SSH 経由で仮想マシンに接続します。**

```
$ ssh -i <example_private_key.pem> <example_azureuser>@<192.0.2.101>
```

2. 一時的なユーザー、ネットワーク、およびホストの情報を削除します。

```
$ sudo waagent -deprovision+user -force
```

3. ログアウトするか、Ctrl + D を押して SSH セッションを閉じます。

4. 仮想マシンを停止します。

```
$ az vm stop --name <example_vm_name> --resource-group
<example_vm_resource_group>
```

5. Azure 課金を停止するためにリソースの割り当てを解除します。

```
$ az vm deallocate --name <example_vm_name> --resource-group
<example_vm_resource_group>
```

6. 仮想マシンを一般化して、このイメージが汎用的で、クローン作成の準備ができていることを確認します。

```
$ az vm generalize --name <example_vm_name> --resource-group
<example_vm_resource_group>
```

8.3. 一般化された仮想マシンからの AZURE イメージバージョンの準備

一般的な仮想マシンから再利用可能な Azure イメージバージョンを作成するには、最初に リソースグループ を作成して、コンピュート、ネットワーク、ストレージなどの関連リソースを整理する必要があります。このリソースグループ内で、Azure Compute Gallery を設定して、組織全体でイメージを管理し、共有します。ギャラリーで、イメージを論理的にグループ化するためにイメージ定義を定義し、そのプロパティーと要件を指定します。これらのイメージ定義に基づいて、一貫性とスケーラビリティーのために複数のイメージバージョンを作成できます。

イメージバージョンを使用すると、レプリカと、同じイメージの複数のバージョンを作成できます。Azure Compute Gallery を使用すると、マーケットプレイスと互換性のあるカスタムイメージを作成して、組織全体で共有することができます。Azure CLI または Azure Cloud Shell を使用できます。Azure Cloud Shell の詳細は、Get started with Azure Cloud Shellを参照してください。

前提条件

- Azure CLI をインストールしている。詳細は、Azure CLI のインストール を参照してください。
- 一般化された仮想マシンを作成している。詳細は、Generalizing an Azure virtual machine for image creation を参照してください。

手順

1. ギャラリーをホストするためのリソースグループを作成します。

```
$ az group create --name <example_image_resource_group>
```

2. 上記のリソースグループにギャラリーを作成します。

```
$ az sig create --resource-group <example_image_resource_group> \
--gallery-name <example_image_gallery_name>
```

3. **サブスクリプションのセキュリティータイプを Standard に設定します。**

```
$ az feature register --name UseStandardSecurityType \
--namespace Microsoft.Compute
```

4. **プロバイダーを登録します。**

```
$ az provider register --namespace Microsoft.Compute
```

5. **イメージのバージョンを管理するイメージ定義を作成します。**

```
$ az sig image-definition create --resource-group <example_image_resource_group> \
--gallery-name <example_image_gallery_name> \
--gallery-image-definition <example_image_definition> \
--publisher <example_publisher> \
--offer <example_offer> \
--sku <example_sky> \
--os-type Linux \
--os-state Generalized \
--hyper-v-generation V2 \
--features SecurityType=Standard
```

<publisher>
イメージを提供するエンティティーまたは組織。

<offer>
パブリッシャーからの関連イメージのコレクションです。

<stock Keeping Unit (SKU)>
メジャーリリースを示すオファー内のエディション。

<VERSION>
特定の SKU のバージョン番号。

6. **イメージに関する情報を取得します。**

```
$ az vm list --output table
```

7. **出力から一般化されたイメージ名を使用します。**

```
$ az vm get-instance-view --resource-group <example_vm_resource_group> \
--name <example_vm_name> \
--query id
```

8. **出力からイメージ定義の ID をコピーします。**

9. **前のステップで取得したイメージ ID を使用して、イメージバージョンを作成します。**

```
$ az sig image-version create \
--resource-group <example_image_resource_group> \
--gallery-name <example_image_gallery_name> \
```

```
--gallery-image-definition <example_image_definition> \
--gallery-image-version <example_version> \
--virtual-machine <example_id>
```

10. 必要に応じて、VM および関連するリソースを削除します。

```
$ az vm delete --resource-group <example_image_resource_group> \
--name <example_vm_name>
```

関連情報

- Azure での Linux 仮想マシンのカスタムイメージの作成

8.4. AZURE CYCLECLOUD と SLURM を使用した HPC クラスターのデプロイ

Azure Cloud で Red Hat Enterprise Linux (RHEL)高パフォーマンスコンピューティング(HPC)クラスターを設定できます。HPC クラスターは、ノードとも呼ばれる複数のマシンにタスクを分散することにより、集中的な処理と計算を必要とする複雑な問題を解決するのに役立ちます。

Azure CycleCloud （クラウドネイティブオーケストレーター）は、Azure Cloud の HPC クラスターを管理します。Azure CycleCloud では、HPC クラスターを管理し、適切なワークロードの自動デプロイメントおよびスケーリングを行うことができます。Azure CycleCloud は、並列コンピューティングジョブ、リソースを管理し、Slurm ワークロードマネージャーを設定します。ただし、slurm は、クラスターでタスクをスケジュールして実行するためのリソースの割り当てを管理します。以下の手順では、RHEL HPC クラスターのデプロイおよび管理に、Slurm および Azure CycleCloud 8.x を使用します。

警告

Azure 環境で RHEL HPC クラスターを設定するには、Azure CycleCloud などの Microsoft Azure サービスを使用できます。これらのツールは、自己のリスクで使用してください。

前提条件

- アクティブな Azure クラウドサブスクリプションがある。
- RHEL HPC イメージがある。詳細は、HPC RHEL システムロールを使用した RHEL HPC イメージの設定 を参照してください。
- 一般化された仮想マシンがある。詳細は、Generalizing an Azure virtual machine for image creation を参照してください。
- Azure イメージバージョンの準備が完了している。詳細は、一般化された仮想マシンからの Azure イメージバージョンの準備 を 参照してください。

手順

1. Azure に CycleCloud をインストールし、デプロイします。
   - Azure Marketplace のインストールについては、Install Azure CycleCloud from Azure Marketplace を参照してください。
   - 手動インストールは Azure CycleCloud を手動でインストールする を参照してください。
2. カスタム RHEL HPC イメージの ID を表示します。

```
$ az sig image-version show --resource-group="<example_resource_group>" \
--gallery-name="<example_gallery>" \
--gallery-image-definition="<example_image>" \
--gallery-image-version="<example_version>" \
--query="id" \
--output="tsv"
```

3. Run Slurm with CycleCloudの手順に従って、Slurm ワークロードマネージャーをプロビジョニングします。

警告

IPv4 アドレスの既知の制限により、Public Head Node オプションを選択すると、Slurm ヘッドノードでプロビジョニングが失敗します。回避策として、Public Head Node オプションがオフになっていることを確認し、環境内の Slurm head ノードにアクセスするのに最適な方法を決定します。詳細は、GitHub の関連する Slurm issue を参照してください。

注記

すべてのクラスターノードで、前の手順で取得したカスタム RHEL イメージ ID を使用します。詳細は、カスタム OS イメージの指定方法 を参照してください。

4. CycleCloud ホームページで、既存の Slurm クラスターを選択します。
5. Slurm クラスターを起動するには、Start をクリックします。
6. クラスター ビューを選択し、Connect をクリックして、Slurm ヘッドノードにログインします。標準の Slurm コマンドラインツールを使用して HPC ジョブをスケジュールします。詳細は、How do I submit jobs? を参照してください。(Slurm)セクションを参照してください。

関連情報

- Azure CycleCloud Overview
- Azure CycleCloud Web Services Overview

8.5. HPC クラスターの環境モジュールの管理

環境モジュールサブシステムは、Lua ベースの環境モジュール(Lmod)を使用して、インストールされ

たモジュールを一覧表示します。Lmod を使用すると、さまざまなソフトウェアパッケージとその依存関係を読み込み、アンロードすることで環境を動的に変更できます。高パフォーマンスコンピューティング(HPC)環境で複数のバージョンのコンパイラー、ライブラリー、およびアプリケーションを管理するため、バージョンごとに特定のソフトウェア設定を選択できます。これには、ml に短縮される モジュール ユーティリティーを使用できます。

8.5.1. 環境モジュール管理のコマンド

以下のコマンドを使用して、環境モジュールを管理できます。

- **module ユーティリティーに関連するすべてのコマンドを表示します。**

```
$ module help
```

- **個々のコマンドオプションと構文を表示します。**

```
$ module <command> help
```

- **特定のモジュールの詳細を表示します。**

```
$ ml whatis pmix/pmix-4.2.9
```

```
pmix/pmix-4.2.9   : Description: PMIx 4.2.9 installed in /opt/pmix/4.2.9
pmix/pmix-4.2.9   : Version: 4.2.9-1
```

- **HPC 環境で利用可能なモジュールを一覧表示します。**

```
$ ml available
```

```
-----------------------/usr/share/modulefiles-------------------------
  mpi/hpcx-2.24.1-pmix-4.2.9    mpi/openmpi-5.0.8-cuda12-gpu (L,D)
  mpi/hpcx-2.24.1               pmix/pmix-4.2.9              (L)
  mpi/openmpi-x86_64

----------------/usr/share/lmod/lmod/modulefiles/Core----------------
  lmod    settarg
```

ここでは、以下のようになります。

  - **L: モジュールがロードされている。**
  - **d: デフォルトモジュール**
  - **(L) アノテーションは、mpi/openmpi-5.0.8-cuda12-gpu モジュールが読み込まれていることを示します。**
  - **これは、pmix/pmix-4.2.9 モジュールを依存関係として読み込みます。**
  - **モジュールシステムは、必要に応じて依存するモジュールを自動的に読み込み、アンロードします。**

- **読み込まれたモジュールを一覧表示します。**

```
$ ml list
```

```
Currently Loaded Modules:
1) pmix/pmix-4.2.9
2) mpi/openmpi-5.0.8-cuda12-gpu
```

- **モジュールとその依存関係をアンロードします。**

```
$ ml unload mpi/openmpi-5.0.8-cuda12-gpu
$ ml list
```

```
No modules loaded
```

- **既存のモジュールをアンロードし、新しいモジュールを読み込みます。**

```
$ ml load mpi/openmpi-5.0.8-cuda12-gpu
$ ml list
```

```
Currently Loaded Modules:

1) pmix/pmix-4.2.9
2) mpi/openmpi-5.0.8-cuda12-gpu
```

- **ロードしたモジュールを切り替えます。**

```
$ ml swap mpi/openmpi-x86_64
```

```
The following have been reloaded with a version change:

1) mpi/openmpi-5.0.8-cuda12-gpu => mpi/openmpi-x86_64
```

- **利用可能なモジュールを一覧表示します。**

```
$ ml list
```

```
Currently Loaded Modules:

1) mpi/openmpi-x86_64
```

8.5.2. Modulefiles のレイアウトとルール

モジュールファイルは、HPC システム上でソフトウェア環境の読み込み、アンロード、および切り替えを行うための環境変数を定義および管理するスクリプトです。推奨されるディレクトリー構造および命名規則により、モジュール定義、ラッパーモジュール、RHEL ベースの HPC デプロイメントで複数のソフトウェア環境を管理するための一貫したルールなど、環境モジュールファイルの効率的な編成が可能になります。

モジュール定義の手動方法

モジュール ユーティリティーは、/usr/share/modulefiles ディレクトリーにあるモジュール定義を自動的に検出します。モジュール定義のあるその他のディレクトリーがある場合は、それらを MODULEPATH 環境変数に追加する必要があります。

ラッパーモジュール

環境変数を変更し、**Lmod** がパッケージ固有のモジュールを見つけないようにするには、ラッパーモジュールを **/usr/share/modulefiles** ディレクトリーに配置して、これらのモジュールが **MODULEPATH** を変更し、外部の場所から関連モジュールを読み込むようにします。このスタイルのラッパーの例は、**mpi/hpcx-2.24.1** 環境モジュールです。

モジュールファイル形式

Lmod は、Lua および Tool Command Language (Tcl)形式で記述された環境モジュールをサポートします。lua スクリプトは、環境モジュールを定義するための推奨される方法です。パーミッション **755** が設定された **.lua** 拡張を使用します。**ml** コマンドのモジュール名は、**.lua** の接尾辞を省略します。このドキュメントの例では、Lua スクリプトインターフェイスを使用します。これには、以下の要件があります。

- MPI ライブラリーなどの同じ機能を提供するすべてのパッケージは、共通のモジュール サブディレクトリー内にあります。この場合、.. **/mpi/** （すべての MPI ライブラリーバリアントの場合）。
- package サブディレクトリー（例： **conflict ("mpi")** ）のモジュールに **conflict** （）定義を追加します。これにより、一度に 1 つのパッケージタイプのモジュールのみをロードできます。
- パッケージの命名の一貫性を維持します。
- 機能名はサブディレクトリー名として使用します。一方、各パッケージインスタンスの個々のモジュールは、以下のパターンに従って命名する必要があります。

```
<package provider>-<version>-<build>-<options>
```

MPI ライブラリーのバリアントが複数ある場合には、異なるプロバイダーからのものもあれば、コンパイラーやビルドオプションが異なる特定のパッケージの複数のビルドであるものもあります。このような場合、命名スキームにより一貫性が確保されます。

8.5.3. 利用可能な MPI 環境

HPC Slurm クラスターでは、メッセージを渡すインターフェイス(MPI)環境では、アプリケーションがノード間で通信および同期するためのランタイムサポートを提供します。Slurm は OpenMPI などの MPI 実装と統合され、分散ジョブを効率的に管理し、クラスターリソースの使用を最適化します。Slurm は **mpirun** を使用して、スケーラブルで高パフォーマンスのジョブのために、割り当てられたノード間で実行を調整します。提供されている RHEL HPC クラスターでは、以下の MPI 環境が利用できます。

- openmpi **.x86_64**: CUDA または GPU アクセラレーションサポートを提供しない標準オープン MPI ビルド。
- openmpi- **5.0.8-cuda-gpu**: GPU 対応の MPI 通信の CUDA サポートでコンパイルされた Open MPI モジュール
- **Hpcx-2.24.1**: Open MPI をベースとした NVIDIA が包括的な HPC ソフトウェアスタック。
- **Hpcx-2.24.1-pmix**: Slurm 統合とジョブスケジューリングのために Process Management Interface (PMIx)で設定された HPC-X ビルド。

MPI ライブラリーの命名で一貫性を保つために、MPI インフラストラクチャーの PML (MPI インフラストラクチャーの PML)実装を指すように Unified Communication X (UCX)を指定します。

```
$ mpirun -mca pml ucx .
```

openmpi.x86_64

このモジュールは、OpenMPI 4.1.1 の標準ビルドを提供します。これには PMIx 3.x のサポートが含まれていますが、CUDA または GPU アクセラレーションのサポートは提供されません。RHEL InfiniBand ドライバーおよびインフラストラクチャーは、InfiniBand (IB)/Remote Direct Memory Access (RDMA)機能を提供します。OpenMPI ライブラリーは gcc-11.4 でコンパイルされています。アプリケーションが CUDA 言語拡張機能を使用しない場合や、GPU オフロードサポートが必要な場合は、このモジュールを使用します。

openmpi-5.0.8-cuda-gpu

このモジュールは、NVIDIA HPC-X パッケージライブラリーを使用して、PMIx 4.x サポート、CUDA および NVIDIA GPU アクセラレーションサポートを提供します。RHEL InfiniBand ドライバーおよびインフラストラクチャーは、InfiniBand (IB)/Remote Direct Memory Access (RDMA)機能を提供します。OpenMPI ライブラリーは、gcc-11.5 でコンパイルされています。GPU アクセラレーションを必要とする CUDA 対応アプリケーションを使用している場合にのみ、このモジュールを選択してください。

注記

InfiniBand NIC がない場合、InfiniBand NIC の自動検出の失敗に関する警告が表示されます。

```
$ mpirun -n 2 /lib64/openmpi/bin/mpitests-osu_allreduce
```

```
...
[test-vm] Error: coll_hcoll_module.c:312 - mca_coll_hcoll_comm_query() Hcol
library init failed
...
```

この警告を削除するには、**-mca coll ^hcoll** パラメーターを **mpirun** コマンドに追加します。

```
$ mpirun -mca coll ^hcoll -n 2 /lib64/openmpi/bin/mpitests-osu_allreduce
```

```
# Size   Avg Latency(us)
  1      8.36
  2      6.85
```

hpcx-2.24.1

このモジュールは、NVIDIA で構築された OpenMPI 4.1.5 環境を提供します。PMIx のサポートはありませんが、CUDA および NVIDIA GPU アクセラレーションサポートを提供します。RHEL InfiniBand ドライバーおよびインフラストラクチャーは、InfiniBand (IB)/Remote Direct Memory Access (RDMA)機能を提供します。

hpcx-2.24.1-pmix

このモジュールは、**mpi/hpcx-2.24.1** モジュールと同じ環境を提供し、PMIx 4.x サポートも有効になっています。

関連情報

- Lmod ドキュメント