Red Hat Enterprise Linux 8

Amazon Web Services への RHEL 8 のデプロイ

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

Last Updated: 2024-05-25

Red Hat Enterprise Linux 8 Amazon Web Services への RHEL 8 のデプロイ

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

法律上の通知

概要

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について説明します。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。

目次

多様性を受け入れるオープンソースの強化

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

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
- AWS で Linux を実行する理由

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、それに伴う変更について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、次のことを行う必要があります。

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 Platform (GCP)
- Microsoft Azure

注記

このドキュメントでは、AWS 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

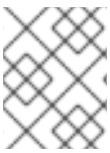
RHEL インスタンスをパブリッククラウドプラットフォームにデプロイするには、次のいずれかの方法を使用できます。

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 Amazon Web Services に RHEL インスタンスをデプロイする詳細な手順については、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 8 の cloud-init の設定と管理

第2章 AWS AMI イメージの作成とアップロード

カスタマイズした RHEL システムイメージを Amazon Web Services (AWS) クラウドで使用するには、Image Builder でそれぞれの出力タイプを使用してシステムイメージを作成し、イメージをアップロードするようにシステムを設定し、イメージを AWS アカウントにアップロードします。

2.1. AWS AMI イメージのアップロードの準備

AWS AMI イメージをアップロードする前に、イメージをアップロードするためのシステムを設定する必要があります。

前提条件

- AWS IAM アカウントマネージャー にアクセスキー ID を設定している。
- 書き込み可能な S3 バケット を準備している。

手順

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

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

2. pip で **AWS コマンドラインツール** をインストールします。

```
# pip3 install awscli
```

3. プロファイルを設定します。ターミナルで、認証情報、リージョン、および出力形式を指定するように求められます。

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

4. バケットの名前を定義し、バケットを作成します。

```
$ BUCKET=bucketname
$ aws s3 mb s3://$BUCKET
```

bucketname は、バケット名に置き換えます。この名前は、グローバルで一意となるように指定する必要があります。上記で、バケットが作成されます。

5. S3 バケットへのアクセス許可を付与するには、AWS Identity and Access Management (IAM) で **vmimport** S3 ロールを作成します (まだ作成していない場合)。

   a. 信頼ポリシーの設定で、JSON 形式で **trust-policy.json** ファイルを作成します。以下に例を示します。

```
{
   "Version": "2022-10-17",
   "Statement": [{
```

```
        "Effect": "Allow",
        "Principal": {
           "Service": "vmie.amazonaws.com"
        },
        "Action": "sts:AssumeRole",
        "Condition": {
           "StringEquals": {
              "sts:Externalid": "vmimport"
           }
        }
     }]
}
```

b. ロールポリシーの設定を含む **role-policy.json** ファイルを JSON 形式で作成します。以下に例を示します。

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Effect": "Allow",
      "Action": ["s3:GetBucketLocation", "s3:GetObject", "s3:ListBucket"],
      "Resource": ["arn:aws:s3:::%s", "arn:aws:s3:::%s/"] }, { "Effect": "Allow", "Action":
["ec2:ModifySnapshotAttribute", "ec2:CopySnapshot", "ec2:RegisterImage",
"ec2:Describe"],
      "Resource": "*"
   }]
}
$BUCKET $BUCKET
```

c. **trust-policy.json** ファイルを使用して、Amazon Web Services アカウントのロールを作成します。

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

d. **role-policy.json** ファイルを使用して、インラインポリシードキュメントを埋め込みます。

```
$ aws iam put-role-policy --role-name vmimport --policy-name vmimport --policy-document file://role-policy.json
```

関連情報

- Using high-level (s3) commands with the AWS CLI

2.2. CLI の使用による AMI イメージの AWS へのアップロード

RHEL Image Builder を使用して **ami** イメージをビルドし、CLI を使用してそれらを Amazon AWS Cloud サービスプロバイダーに直接プッシュできます。

前提条件

- AWS IAM アカウントマネージャーに **Access Key ID** を設定している。
- 書き込み可能な S3 バケット を準備している。

- 定義済みの青写真がある。

手順

1. テキストエディターを使用して、次の内容の設定ファイルを作成します。

```
provider = "aws"

[settings]
accessKeyID = "AWS_ACCESS_KEY_ID"
secretAccessKey = "AWS_SECRET_ACCESS_KEY"
bucket = "AWS_BUCKET"
region = "AWS_REGION"
key = "IMAGE_KEY"
```

   フィールドの値を **accessKeyID**、**secretAccessKey**、**bucket**、および **region** の認証情報に置き換えます。**IMAGE_KEY** 値は、EC2 にアップロードする仮想マシンイメージの名前です。

2. ファイルを **CONFIGURATION-FILE**.toml として保存し、テキストエディターを閉じます。

3. Compose を開始して AWS にアップロードします。

```
# composer-cli compose start blueprint-name image-type image-key configuration-file.toml
```

   以下を置き換えます。

   - **blueprint-name** は、作成したブループリントの名前に置き換えます。
   - **blueprint-name** は、**ami** イメージタイプに置き換えます。
   - **image-key** は、EC2 にアップロードする仮想マシンイメージの名前に置き換えます。
   - **configuration-file**.toml は、クラウドプロバイダーの設定ファイルの名前に置き換えます。

注記

カスタマイズしたイメージの送信先となるバケットの正しい AWS Identity and Access Management (IAM) 設定が必要です。イメージをアップロードする前にバケットにポリシーを設定しておく必要があります。

4. イメージビルドのステータスを確認します。

```
# composer-cli compose status
```

   イメージのアップロードプロセスが完了すると、FINISHED ステータスが表示されます。

検証

イメージのアップロードが成功したことを確認するには、以下を行います。

1. メニューで EC2 にアクセスし、AWS コンソールで正しいリージョンを選択します。イメージが正常にアップロードされたことを示すには、イメージが **available** ステータスになっている必要があります。

2. ダッシュボードでイメージを選択し、**Launch** をクリックします。

関連情報

- 仮想マシンのインポートに必要なサービスロール

2.3. イメージの AWS CLOUD AMI へのプッシュ

RHEL Image Builder を使用して **(.raw)** イメージを作成し、**Upload to AWS** チェックボックスをオンにして、作成した出力イメージを **Amazon AWS Cloud AMI** サービスプロバイダーに直接プッシュすることができます。

前提条件

- **root** または **wheel** グループでシステムにアクセスできる。
- ブラウザーで、RHEL Web コンソールの RHEL Image Builder インターフェイスを開いている。
- 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ブループリントの作成を参照してください。
- AWS IAM アカウントマネージャーにアクセスキー ID を設定している。
- 書き込み可能な S3 バケット を準備している。

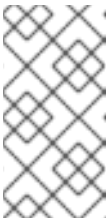
手順

1. RHEL Image Builder のダッシュボードで、以前に作成した **ブループリント名** をクリックします。
2. **Images** タブを選択します。
3. **Create Image** をクリックして、カスタマイズしたイメージを作成します。
   Create Image ウィンドウが開きます。
   a. **Type** ドロップダウンメニューから、**Amazon Machine Image Disk (.raw)** を選択します。
   b. イメージを AWS Cloud にアップロードするには、**Upload to AWS** チェックボックスをオンして、**Next** をクリックします。
   c. AWS へのアクセスを認証するには、対応するフィールドに **AWS access key ID** および **AWS secret access key** を入力します。**Next** をクリックします。

注記

新規アクセスキー ID を作成する場合にのみ、AWS シークレットアクセスキーを表示できます。秘密鍵が分からない場合は、新しいアクセスキー ID を生成します。

   d. **Image name** フィールドにイメージ名を、**Amazon S3 bucket name** フィールドに Amazon バケット名を入力して、カスタマイズイメージを追加するバケットの **AWS region** フィールドを入力します。**Next** をクリックします。
   e. 情報を確認し、**Finish** をクリックします。
   必要に応じて、**Back** をクリックして誤った情報を変更します。



注記

カスタマイズイメージを送信するバケットの正しい IAM 設定が必要です。この手順では IAM のインポートとエクスポートを使用するため、バケットにイメージをアップロードする前にバケットに **ポリシー** を設定する必要があります。詳細は、IAM ユーザーの必要なパーミッション を参照してください。

4. 右上のポップアップで、保存の進行状況が通知されます。イメージ作成の開始、イメージ作成の進捗、およびその後の AWS Cloud にアップロードに関する情報も通知されます。
プロセスが完了すると、**Image build complete** のステータスが表示されます。

5. ブラウザーで、Service→EC2 にアクセスします。

   a. AWS コンソールのダッシュボードメニューで、正しいリージョン を選択します。イメージのステータスは、アップロードされていることを示す **Available** でなければなりません。

   b. AWS ダッシュボードでイメージを選択し、**Launch** をクリックします。

6. 新しいウィンドウが開きます。イメージを開始するために必要なリソースに応じて、インスタンスタイプを選択します。**Review and Launch** をクリックします。

7. インスタンスの開始の詳細を確認します。変更が必要な場合は、各セクションを編集できます。**Launch** をクリックします。

8. インスタンスを起動する前に、インスタンスにアクセスするための公開鍵を選択します。
既存のキーペアを使用するか、キーペアーを新規作成します。

   次の手順に従って、EC2 で新規キーペアを作成し、新規インスタンスにアタッチします。

   a. ドロップダウンメニューリストから、**Create a new key pair**を選択します。

   b. 新しいキーペアに名前を入力します。新しいキーペアが生成されます。

   c. **Download Key Pair** をクリックして、新しいキーペアをローカルシステムに保存します。

9. 次に、**Launch Instance** をクリックしてインスタンスを起動できます。
Initializing と表示されるインスタンスのステータスを確認できます。

10. インスタンスのステータスが **running** になると、**Connect** ボタンが有効になります。

11. **Connect** をクリックします。ウィンドウが表示され、SSH を使用して接続する方法の説明が表示されます。

    a. 優先する接続方法として **スタンドアロン SSH クライアント** を選択し、ターミナルを開きます。

    b. 秘密鍵を保存する場所で、SSH が機能するために鍵が公開されていることを確認してください。これには、以下のコマンドを実行します。

    ```
    $ chmod 400 _your-instance-name.pem_>
    ```

    c. パブリック DNS を使用してインスタンスに接続します。

    ```
    $ ssh -i <_your-instance-name.pem_> ec2-user@<_your-instance-IP-address_>
    ```

d. **yes** と入力して、接続の続行を確定します。
その結果、SSH 経由でインスタンスに接続されます。

検証

- SSH でインスタンスに接続している間にアクションが実行できるかどうかを確認します。

関連情報

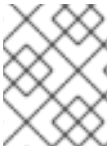
- Red Hat カスタマーポータルでのケースの作成
- Connecting to your Linux instance by using SSH

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

Amazon Web Services (AWS) 上で RHEL の高可用性 (HA) デプロイメントを設定するには、RHEL の EC2 インスタンスを AWS 上のクラスターにデプロイします。

重要

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



注記

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

前提条件

- Red Hat カスタマーポータル のアカウントに登録している。
- AWS にサインアップして、AWS リソースを設定します。詳細は Setting Up with Amazon EC2 を参照してください。

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

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

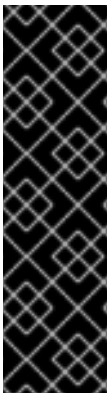
表3.1 イメージオプション

イメージオプション	サブスクリプション	サンプルシナリオ	留意事項
Red Hat Gold Image 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	AWS で Red Hat Gold Image を選択します。Gold Image の詳細と、AWS で Gold Image にアクセスする方法については、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、Amazon 社に支払うことになります。Red Hat は、クラウドアクセスイメージを直接サポートします。

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
AWS 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、Amazon 社に支払うことになります。Red Hat は、カスタム RHEL イメージのサポートを直接提供します。
RHEL を含む既存の Amazon イメージをデプロイする	AWS EC2 イメージには Red Hat 製品が含まれる	AWS マネジメントコンソール でのインスタンスの起動時に RHEL イメージを選択するか、AWS Marketplace からイメージを選択します。	Amazon 社に、**従量課金**モデルで 1 時間ごとに支払います。このようなイメージはオンデマンドイメージと呼ばれています。Amazon 社はオンデマンドイメージをサポートします。 Red Hat は、イメージの更新を提供します。AWS により、Red Hat Update Infrastructure (RHUI) から更新を利用できるようにします。

注記

Red Hat Image Builder を使用して、AWS 用のカスタムイメージを作成できます。詳細は RHEL システムイメージのカスタマイズ を参照してください。



重要

オンデマンドインスタンスをカスタム RHEL インスタンスに変換することはできません。オンデマンドイメージからカスタム RHEL **bring-your-own-subscription** (BYOS) イメージに変更するには、次の手順を実行します。

1. 新しいカスタム RHEL インスタンスを作成し、オンデマンドインスタンスからデータを移行します。
2. データを移行した後に、オンデマンドのインスタンスをキャンセルして二重請求を回避します。

関連情報

- RHEL システムイメージのカスタマイズの作成
- AWS マネジメントコンソール
- AWS Marketplace

3.2. ベースイメージの理解

ISO イメージからベース仮想マシンを作成するには、事前設定されたベースイメージとその設定を使用できます。

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。

関連情報

- Red Hat Enterprise Linux

3.2.2. 仮想マシン設定

クラウド仮想マシンには、以下の設定が必要です。

表3.2 仮想マシンの設定

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

3.3. ISO イメージからのベース仮想マシンの作成

ISO イメージから RHEL 8 ベースイメージを作成するには、ホストマシンの仮想化を有効にし、RHEL 仮想マシン (VM) を作成します。

前提条件

- ホストマシンで 仮想化が有効 になっている。
- Red Hat カスタマーポータル から最新の Red Hat Enterprise Linux ISO イメージをダウンロードし、イメージを **/var/lib/libvirt/images** に移動しました。

3.3.1. RHEL ISO イメージからの仮想マシンの作成

手順

1. 仮想化用のホストマシンを有効にしていることを確認します。設定および手順は、RHEL 8 で仮想化を有効にする を参照してください。
2. 基本的な Red Hat Enterprise Linux 仮想マシンを作成し、起動します。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 **/home/username/Downloads/rhel9.iso** イメージを使用して仮想マシン **kvmtest** を作成します。

```
# virt-install \
    --name kvmtest --memory 2048 --vcpus 2 \
    --cdrom /home/username/Downloads/rhel8.iso,bus=virtio \
    --os-variant=rhel8.0
```

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。

3.3.2. RHEL インストールの完了

Amazon Web Services (AWS) にデプロイする RHEL システムのインストールを完了するには、**インストールの概要** ビューをカスタマイズし、インストールを開始し、仮想マシンの起動後にルートアクセスを有効にします。

手順

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

a. 仮想マシンを登録し、Red Hat Enterprise Linux 8 リポジトリーを有効にします。

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

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

```
# yum install cloud-init
# systemctl enable --now cloud-init.service
```

7. **重要: この手順は、AWS にアップロードする仮想マシンにのみ行います。**

a. AMD64 または Intel 64 (x86_64) 仮想マシンの場合は、**nvme** ドライバー、**xen-netfront** ドライバー、および **xen-blkfront ドライバー** をインストールします。

```
# dracut -f --add-drivers "nvme xen-netfront xen-blkfront"
```

b. ARM 64 (aarch64) 仮想マシンの場合は、**nvme** ドライバーをインストールします。

```
# dracut -f --add-drivers "nvme"
```

これらのドライバーを追加すると、dracut タイムアウトの可能性がなくなります。

ここでは、ドライバーを **/etc/dracut.conf.d/** に追加し、**dracut -f** を入力して既存の **initramfs** ファイルを上書きできます。

8. 仮想マシンの電源をオフにします。

関連情報

- サブスクリプションの自動割り当てと更新
- cloud-init の概要

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

Amazon Web Services (AWS) で RHEL インスタンスを実行できるようにするには、まず RHEL イメージを AWS にアップロードする必要があります。

3.4.1. AWS CLI のインストール

AWS で HA クラスターを管理するために必要な手順の多くには、AWS CLI の使用が含まれます。

前提条件

- AWS アクセスキー ID および AWS シークレットアクセスキーを作成し、それらにアクセスできるようになりました。手順と詳細は、AWS CLI のクイック設定 を参照してください。

手順

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

```
# yum install awscli
```

2. **aws --version** コマンドを使用して、AWS CLI をインストールしたことを確認します。

```
$ aws --version
aws-cli/1.19.77 Python/3.6.15 Linux/5.14.16-201.fc34.x86_64 botocore/1.20.77
```

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

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

関連情報

- AWS CLI の設定
- AWS コマンドラインツール

3.4.2. S3 バケットの作成

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

手順

1. Amazon S3 コンソール を起動します。
2. **Create Bucket** をクリックします。**Create Bucket** ダイアログが表示されます。
3. **Name and region** ビューで、以下を行います。
   a. **Bucket name** を入力します。
   b. **Region** を入力します。
   c. **Next** をクリックします。
4. **Configure options** ビューでは、目的のオプションを選択し、**Next** をクリックします。
5. **Set permissions** ビューで、デフォルトのオプションを変更または受け入れ、**Next** をクリックします。
6. バケットの設定を確認します。
7. **Create bucket** をクリックします。

注記

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

関連情報

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

3.4.3. vmimport ロールの作成

仮想マシンインポートサービスを使用して RHEL 仮想マシン (VM) を Amazon Web Services (AWS) にインポートするには、**vmimport** ロールを作成する必要があります。

詳細は、Amazon ドキュメントの Importing a VM as an image using VM Import/Export を参照してください。

手順

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

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

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

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

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

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetBucketLocation",
            "s3:GetObject",
            "s3:ListBucket"
         ],
         "Resource":[
```

```
            "arn:aws:s3:::s3-bucket-name",
            "arn:aws:s3:::s3-bucket-name/*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "ec2:ModifySnapshotAttribute",
            "ec2:CopySnapshot",
            "ec2:RegisterImage",
            "ec2:Describe*"
         ],
         "Resource":"*"
      }
   ]
}
```

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

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

関連情報

- VM インポートサービスロール
- 必要なサービスロール

3.4.4. イメージの S3 への変換およびプッシュ

qemu-img コマンドを使用すると、イメージを変換して S3 にプッシュできるようになります。サンプルは典型的なもので、**qcow2** ファイル形式でフォーマットされたイメージを **raw** 形式に変換します。Amazon では、**OVA**、**VHD**、**VHDX**、**VMDK**、および **raw** の形式のイメージを利用できます。Amazon で利用できるイメージ形式の詳細は、How VM Import/Export works を参照してください。

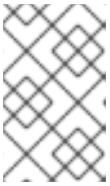
手順

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

```
# qemu-img convert -f qcow2 -O raw rhel-8.0-sample.qcow2 rhel-8.0-sample.raw
```

2. イメージを S3 にプッシュします。

```
$ aws s3 cp rhel-8.0-sample.raw s3://s3-bucket-name
```



注記

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

関連情報

- VM のインポート/エクスポートの仕組み
- AWS S3 コンソール

3.4.5. イメージのスナップショットとしてのインポート

Amazon Elastic Cloud Compute (EC2) サービスで RHEL インスタンスを起動するには、Amazon Machine Image (AMI) が必要です。システムの AMI を作成するには、まず RHEL システムイメージのスナップショットを EC2 にアップロードする必要があります。

手順

1. イメージのバケットとパスを指定するファイルを作成します。**container.json** ファイルに名前を付けます。以下の例では、**s3-bucket-name** をバケット名に置き換え、**s3-key** を鍵に置き換えます。Amazon S3 コンソールを使用して、イメージの鍵を取得できます。

```
{
    "Description": "rhel-8.0-sample.raw",
    "Format": "raw",
    "UserBucket": {
        "S3Bucket": "s3-bucket-name",
        "S3Key": "s3-key"
    }
}
```

2. イメージをスナップショットとしてインポートします。この例では、パブリックの Amazon S3 ファイルを使用しています。Amazon S3 コンソール を使用して、バケットのパーミッション設定を変更できます。

```
$ aws ec2 import-snapshot --disk-container file://containers.json
```

端末は以下のようなメッセージを表示します。メッセージ内の **ImportTaskID** を書き留めます。

```
{
    "SnapshotTaskDetail": {
        "Status": "active",
        "Format": "RAW",
        "DiskImageSize": 0.0,
        "UserBucket": {
            "S3Bucket": "s3-bucket-name",
            "S3Key": "rhel-8.0-sample.raw"
        },
        "Progress": "3",
        "StatusMessage": "pending"
    },
    "ImportTaskId": "import-snap-06cea01fa0f1166a8"
}
```

3. **describe-import-snapshot-tasks** コマンドを使用して、インポートの進行状況を追跡します。**ImportTaskID** を含めます。

```
$ aws ec2 describe-import-snapshot-tasks --import-task-ids import-snap-06cea01fa0f1166a8
```

返されるメッセージには、タスクの現在の状態が表示されます。完了したら、**Status** は **completed** になります。ステータスに記載されているスナップショット ID を書き留めます。

関連情報

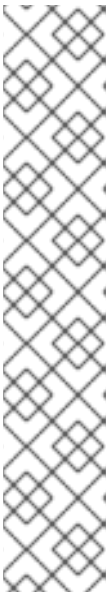
- Amazon S3 Console
- VM Import/Export を使用したスナップショットとしてのディスクのインポート

3.4.6. アップロードしたスナップショットからの AMI の作成

Amazon Elastic Cloud Compute (EC2) サービスで RHEL インスタンスを起動するには、Amazon Machine Image (AMI) が必要です。システムの AMI を作成するには、以前にアップロードした RHEL システムスナップショットを使用できます。

手順

1. AWS EC2 Dashboard に移動します。
2. **Elastic Block Store** で **スナップショット** を選択します。
3. スナップショット ID (例: **snap-0e718930bd72bcda0**) を検索します。
4. スナップショットを右クリックして、**イメージの作成** を選択します。
5. イメージに名前を付けます。
6. **仮想化のタイプ** で、**ハードウェア仮想化支援機能** を選択します。
7. **作成** をクリックします。イメージ作成に関する注意事項に、イメージへのリンクがあります。
8. イメージリンクをクリックします。**Images>AMI** の下にイメージが表示されます。



注記

また、AWS CLI の **register-image** コマンドを使用して、スナップショットから AMI を作成できます。詳細は、register-image を参照してください。以下に例を示します。

```
$ aws ec2 register-image \
   --name "myimagename" --description "myimagedescription" --architecture
x86_64 \
   --virtualization-type hvm --root-device-name "/dev/sda1" --ena-support \
   --block-device-mappings "{\"DeviceName\": \"/dev/sda1\",\"Ebs\":
{\"SnapshotId\": \"snap-0ce7f009b69ab274d\"}}"
```

root デバイスボリューム **/dev/sda1** を **root-device-name** として指定する必要があります。AWS のデバイスマッピングの概念情報は、Example block device mapping を参照してください。

3.4.7. AMI からのインスタンスの起動

Amazon Elastic Compute Cloud (EC2) インスタンスを起動して設定するには、Amazon Machine Image (AMI) を使用します。

手順

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

注記

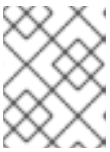
これらは、基本インスタンスの作成に必要な最小限の設定オプションです。アプリケーション要件に応じて追加オプションを確認します。

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

注記

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

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



注記

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

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

注記

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

関連情報

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

3.4.8. 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 --auto-attach
```

2. サブスクリプションを割り当てます。
   - アクティベーションキーを使用して、サブスクリプションを割り当てることができます。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。
   - または、サブスクリプションプール (Pool ID) の ID を使用してサブスクリプションを手動で割り当てることができます。コマンドラインでのサブスクリプションのアタッチと削除 を参照してください。
3. **オプション:** Red Hat Hybrid Cloud Console でインスタンスに関するさまざまなシステムメトリクスを収集するには、インスタンスを Red Hat Insights に登録します。

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

   Red Hat Insights の詳細な設定については、Red Hat Insights のクライアント設定ガイド を参照してください。

関連情報

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

- Red Hat Insights のクライアント設定ガイド

3.4.9. AWS Gold Image の自動登録の設定

Amazon Web Services (AWS) 上に RHEL 8 の仮想マシンをより早く、より快適にデプロイするために、RHEL 8 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

- 最新の AWS 用 RHEL 8 Gold Image をダウンロードしている。手順については、AWS でのゴールドイメージの使用 を参照してください。

注記

AWS 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 AWS 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 AWS にアップロードします。手順については、Uploading the Red Hat Enterprise Linux image to AWS を参照してください。
2. アップロードされたイメージを使用して VM 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 8 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

- AWS マネジメントコンソール
- Hybrid Cloud Console へのクラウドソースの追加

3.5. 関連情報

- Red Hat Cloud Access リファレンスガイド
- Red Hat in the Public Cloud
- Red Hat Enterprise Linux on Amazon EC2 - FAQs
- Amazon EC2 での設定

- Red Hat on Amazon Web Services

第4章 AWS 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、AWS 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。AWS 上で RHEL HA クラスターを作成することは、非クラウド環境で HA クラスターを作成するのと似ています。

EC2 インスタンスをクラスターノードとして使用して Amazon Web Services (AWS) 上に Red Hat HA クラスターを設定するには、次のセクションを参照してください。クラスターに使用する Red Hat Enterprise Linux (RHEL) イメージを取得するオプションは複数あることに注意してください。AWS のイメージオプションの詳細は、AWS での Red Hat Enterprise Linux Image オプション を参照してください。

前提条件

- Red Hat カスタマーポータル のアカウントに登録している。
- AWS にサインアップして、AWS リソースを設定します。詳細は Setting Up with Amazon EC2 を参照してください。

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. AWS アクセスキーおよび AWS シークレットアクセスキーの作成

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

前提条件

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

手順

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

4.3. AWS CLI のインストール

AWS で HA クラスターを管理するために必要な手順の多くには、AWS CLI の使用が含まれます。

前提条件

- AWS アクセスキー ID および AWS シークレットアクセスキーを作成し、それらにアクセスできるようになりました。手順と詳細は、AWS CLI のクイック設定 を参照してください。

手順

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

```
# yum install awscli
```

2. **aws --version** コマンドを使用して、AWS CLI をインストールしたことを確認します。

```
$ aws --version
aws-cli/1.19.77 Python/3.6.15 Linux/5.14.16-201.fc34.x86_64 botocore/1.20.77
```

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

```
$ aws configure
AWS Access Key ID [None]:
AWS Secret Access Key [None]:
```

```
Default region name [None]:
Default output format [None]:
```

関連情報

- AWS CLI の設定
- AWS コマンドラインツール

4.4. HA EC2 インスタンスの作成

HA クラスターノードとして使用するインスタンスを作成するには、以下の手順を実施します。クラスターに使用する RHEL イメージを取得するオプションは複数あることに注意してください。AWS のイメージオプションに関する詳細は、AWS の Red Hat Enterprise Linux Image オプション を参照してください。

クラスターノードに使用するカスタムイメージを作成してアップロードすることも、Gold Image やオンデマンドイメージを使用することもできます。

前提条件

- AWS 環境をセットアップしている。詳細は、Setting Up with Amazon EC2 を参照してください。

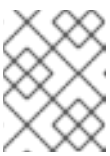
手順

1. AWS EC2 Dashboard から、**Images** を選択して、**AMI** を選択します。
2. イメージを右クリックして、**Launch** を選択します。
3. ワークロードの要件を満たす、もしくは超過する **Instance Type** を選択します。HA アプリケーションによっては、各インスタンスの容量を増やす必要がある場合があります。
   インスタンスタイプの詳細は、Amazon EC2 Instance Types を参照してください。
4. **Next:Configure Instance Details** をクリックします。
   a. クラスターを作成する **インスタンス数** を入力します。この例の手順では、3 つのクラスターノードを使用します。

注記

自動スケーリンググループでは起動しないでください。

   b. **ネットワーク** の場合は、Set up the AWS environment で作成した VPC を選択します。新規サブネットを作成するインスタンスのサブネットを選択します。
   c. 自動割り当てパブリック IP では、**Enable** を選択します。**Configure Instance Details** に必要な最小限の選択です。特定の HA アプリケーションによっては、追加の選択が必要になる場合があります。



注記

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

5. **Next: Add Storage** をクリックし、デフォルトのストレージが十分であることを確認します。HA アプリケーションに他のストレージオプションが必要でない限り、これらの設定を変更する必要はありません。

6. **Next: Add Tags** をクリックします。

注記

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

7. **Next:Configure Security Group** をクリックします。Setting up the AWS environment で作成した既存のセキュリティーグループを選択します。

8. **Review and Launch** をクリックし、選択を確認します。

9. **Launch** をクリックします。既存の鍵のペアの選択、または新しい鍵のペアの作成に関するダイアログが表示されます。AWS 環境でのセットアップ 時に作成した鍵のペアを選択します。

10. **Launch Instances** をクリックします。

11. **View Instances** をクリックします。インスタンスに名前を付けることができます。

注記

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

関連情報

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

4.5. 秘密鍵の設定

プライベート SSH キーファイル (**.pem**) を使用するための以下の設定タスクを完了してからではない、SSH セッションで使用できません。

手順

1. キーファイルを **Downloads** ディレクトリーから **ホーム** ディレクトリーまたは ~/**.ssh ディレクトリー** に移動します。

2. root ユーザーのみが読み取れるように、キーファイルのアクセス許可を変更します。

```
# chmod 400 KeyName.pem
```

4.6. EC2 インスタンスへの接続

すべてのノードで **AWS コンソール** を使用して、EC2 インスタンスに接続できます。

手順

1. AWS コンソール を起動し、EC2 インスタンスを選択します。
2. **Connect** をクリックし、**A standalone SSH client** を選択します。
3. SSH 端末セッションから、ポップアップウィンドウで提供される AWS の例を使用してインスタンスに接続します。パスが例に表示されていない場合は、正しいパスを **KeyName.pem** ファイルに追加します。

4.7. 高可用性パッケージおよびエージェントのインストール

AWS 上で Red Hat High Availability クラスターを設定できるようにするには、各ノードに高可用性パッケージとエージェントをインストールする必要があります。

手順

1. AWS Red Hat Update Infrastructure (RHUI) クライアントを削除します。

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

2. 仮想マシンを Red Hat に登録します。

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

3. すべてのリポジトリーを無効にします。

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

4. RHEL 8 サーバーの HA リポジトリーを有効にします。

```
# subscription-manager repos --enable=rhel-8-for-x86_64-highavailability-rpms
```

5. RHEL AWS インスタンスを更新します。

```
# yum 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 AWS フェンスエージェントをインストールします。

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

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

```
# passwd hacluster
```

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
```

10. **/etc/hosts** を編集し、RHEL ホスト名と内部 IP アドレスを追加します。詳細は、 RHEL クラスターノードに /etc/hosts ファイルを設定する を参照してください。

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 Thu 2018-03-01 14:53:28 UTC; 28min ago
Docs: man:pcsd(8)
man:pcs(8)
Main PID: 5437 (pcsd)
CGroup: /system.slice/pcsd.service
    └─5437 /usr/bin/ruby /usr/lib/pcsd/pcsd > /dev/null &
Mar 01 14:53:27 ip-10-0-0-48.ec2.internal systemd[1]: Starting PCS GUI and remote
configuration interface…
Mar 01 14:53:28 ip-10-0-0-48.ec2.internal systemd[1]: Started PCS GUI and remote
configuration interface.
```

4.8. 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...
```

4.9. フェンシングの設定

フェンシング設定により、AWS クラスター上の誤動作しているノードが自動的に分離され、ノードがクラスターのリソースを消費したり、クラスターの機能が損なわれたりするのを防ぎます。

AWS クラスターでフェンシングを設定するには、複数の方法を使用できます。

- デフォルト設定の標準手順。
- 自動化に焦点を当てた、より高度な設定のための代替設定手順。

前提条件

- **fence_aws** フェンスエージェントを使用している。**fence_aws** を取得するには、クラスターに **resource-agents** パッケージをインストールします。

標準手順

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

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

以下に例を示します。

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

2. 以下のコマンドを実行して、フェンスデバイスを設定します。**pcmk_host_map** コマンドを使用して、RHEL ホスト名をインスタンス ID にマッピングします。以前に設定した AWS アクセスキーおよび AWS シークレットアクセスキーを使用します。

```
# pcs stonith \
    create <name> fence_aws access_key=access-key secret_key=<secret-access-key> \
    region=<region> pcmk_host_map="rhel-hostname-1:Instance-ID-1;rhel-hostname-2:Instance-ID-2;rhel-hostname-3:Instance-ID-3" \
    power_timeout=240 pcmk_reboot_timeout=480 pcmk_reboot_retries=4
```

以下に例を示します。

```
[root@ip-10-0-0-48 ~]# pcs stonith \
create clusterfence fence_aws access_key=AKIAI123456MRMJA
secret_key=a75EYIG4RVL3hdsdAslK7koQ8dzaDyn5yoIZ/ \
region=us-east-1 pcmk_host_map="ip-10-0-0-48:i-07f1ac63af0ec0ac6;ip-10-0-0-46:i-063fc5fe93b4167b2;ip-10-0-0-58:i-08bd39eb03a6fd2c7" \
power_timeout=240 pcmk_reboot_timeout=480 pcmk_reboot_retries=4
```

別の手順

1. クラスターの VPC ID を取得します。

```
# aws ec2 describe-vpcs --output text --filters "Name=tag:Name,Values=<clustername>-vpc" --query 'Vpcs[*].VpcId'
vpc-06bc10ac8f6006664
```

2. クラスターの VPC ID を使用して、VPC インスタンスを取得します。

```
$ aws ec2 describe-instances --output text --filters "Name=vpc-id,Values=vpc-06bc10ac8f6006664" --query 'Reservations[*].Instances[*].{Name:Tags[? Key==Name]|[0].Value,Instance:InstanceId}' | grep "\-node[a-c]"
i-0b02af8927a895137     <clustername>-nodea-vm
i-0cceb4ba8ab743b69     <clustername>-nodeb-vm
i-0502291ab38c762a5     <clustername>-nodec-vm
```

3. 取得したインスタンス ID を使用して、クラスター上の各ノードでフェンシングを設定します。たとえば、クラスター内のすべてのノードでフェンスデバイスを設定するには、次のようにします。

```
[root@nodea ~]# CLUSTER=<clustername> && pcs stonith create fence${CLUSTER} fence_aws access_key=XXXXXXXXXXXXXXXXXXXX pcmk_host_map=$(for NODE \
```

```
in node{a..c}; do ssh ${NODE} "echo -n \${HOSTNAME}:\$(curl -s
http://169.254.169.254/latest/meta-data/instance-id)\;"; done) \
pcmk_reboot_retries=4 pcmk_reboot_timeout=480 power_timeout=240 region=xx-xxxx-x
secret_key=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

フェンスデバイスを作成するための特定のパラメーターについては、**fence_aws の** man ページまたは 高可用性クラスターの設定と管理 ガイドを参照してください。

検証

1. ノードに設定されているフェンスデバイスとそのパラメーターを表示します。

```
[root@nodea ~]# pcs stonith config fence${CLUSTER}

Resource: <clustername> (class=stonith type=fence_aws)
Attributes: access_key=XXXXXXXXXXXXXXXXXXX pcmk_host_map=nodea:i-
0b02af8927a895137;nodeb:i-0cceb4ba8ab743b69;nodec:i-0502291ab38c762a5;
pcmk_reboot_retries=4 pcmk_reboot_timeout=480 power_timeout=240 region=xx-xxxx-x
secret_key=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Operations: monitor interval=60s (<clustername>-monitor-interval-60s)
```

2. クラスターノードのいずれかに対してフェンスエージェントをテストします。

```
# pcs stonith fence <awsnodename>
```

注記

コマンドの応答が表示されるまで数分かかる場合があります。フェンシングしているノードのアクティブな端末セッションを確認する場合は、fence コマンドを入力すると、端末の接続がすぐに終了するようになります。

以下に例を示します。

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

Node: ip-10-0-0-58 fenced
```

3. ステータスを確認して、ノードがフェンスされていることを確認します。

```
# pcs status
```

以下に例を示します。

```
[root@ip-10-0-0-48 ~]# pcs status

Cluster name: newcluster
Stack: corosync
Current DC: ip-10-0-0-46 (version 1.1.18-11.el7-2b07d5c5a9) - partition with quorum
Last updated: Fri Mar  2 19:55:41 2018
Last change: Fri Mar  2 19:24:59 2018 by root via cibadmin on ip-10-0-0-46

3 nodes configured
1 resource configured
```

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

Full list of resources:
clusterfence  (stonith:fence_aws):    Started ip-10-0-0-46

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

4. 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 <awshostname>
```

5. 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@ip-10-0-0-48 ~]# pcs status

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

3 nodes configured
1 resource configured

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

Full list of resources:

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

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

4.10. クラスターノードへの AWS CLI のインストール

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

各クラスターノードで以下の手順を実行します。

前提条件

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

手順

1. AWS CLI をインストールしている。手順については、Installing the AWS CLI を参照してください。
2. AWS CLI が正しく設定されていることを確認します。インスタンス ID およびインスタンス名が表示されます。
以下に例を示します。

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

4.11. AWS での IP アドレスリソースの設定

フェイルオーバーが発生した場合でも、ネットワーク経由でクラスターに管理されるリソースに IP アドレスを使用してアクセスするクライアントがリソースにアクセスできるようにするには、特定のネットワークリソースエージェントを使用する **IP アドレスリソース** をクラスターに含める必要があります。

RHEL HA アドオンは、AWS 上のさまざまな種類の IP アドレスを管理するための IP アドレスリソースを作成するリソースエージェントのセットを提供します。どのリソースエージェントを設定するかを決定するには、HA クラスターで管理する AWS IP アドレスのタイプを考慮してください。

- インターネットに公開されている IP アドレスを管理する必要がある場合は、**awseip** ネットワークリソースを使用 します。
- 単一の AWS アベイラビリティーゾーン (AZ) に制限されたプライベート IP アドレスを管理する必要がある場合は、**awsvip** および **IPaddr2** ネットワークリソースを使用します。
- 同じ AWS リージョン内の複数の AWS AZ 間で移動できる IP アドレスを管理する必要がある場合は、**aws-vpc-move-ip** ネットワークリソースを使用 します。

注記

HA クラスターが IP アドレスを管理しない場合は、AWS 上の仮想 IP アドレスを管理するためのリソースエージェントは必要ありません。特定のデプロイメントに関する詳細なガイダンスが必要な場合は、AWS プロバイダーにご相談ください。

4.11.1. インターネットに公開されている IP アドレスを管理するための IP アドレスリソースの作成

高可用性 (HA) クライアントがパブリックインターネット接続を使用する RHEL 8 ノードにアクセスできるようにするには、Elastic IP アドレスを使用するように **AWS セカンダリー Elastic IP アドレス** (**awseip**) リソースを設定します。

前提条件

前提条件

- 以前に設定されたクラスターがある。
- クラスターノードが RHEL HA リポジトリーにアクセスできる。詳細は、高可用性パッケージとエージェントのインストール を参照してください。
- AWS CLI を設定している。手順については、Installing the AWS CLI を参照してください。

手順

1. **resource-agents** パッケージをインストールします。

```
# yum install resource-agents
```

2. AWS コマンドラインインターフェイス (CLI) を使用して、Elastic IP アドレスを作成します。

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

eipalloc-4c4a2c45    vpc 35.169.153.122
```

3. **オプション: awseip** の説明を表示します。これは、このエージェントのオプションとデフォルトの操作を示しています。

```
# pcs resource describe awseip
```

4. AWS CLI を使用して以前に指定した割り当て済み IP アドレスを使用するセカンダリー Elastic IP アドレスリソースを作成します。さらに、セカンダリー Elastic IP アドレスが属するリソースグループを作成します。

```
# pcs resource create <resource-id> awseip elastic_ip=<Elastic-IP-Address>
allocation_id=<Elastic-IP-Association-ID> --group networking-group
```

以下に例を示します。

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

検証

1. クラスターのステータスを表示して、必要なリソースが実行していることを確認します。

```
# pcs status
```

次の出力は、**vip** リソースおよび **elastic** リソースが **networking-group** リソースグループの一部として起動されている実行中のクラスターの例を示しています。

```
[root@ip-10-0-0-58 ~]# pcs status

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

```
3 nodes configured
4 resources configured

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

Full list of resources:

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

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

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

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

以下に例を示します。

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

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

4.11.2. 単一の AWS アベイラビリティーゾーンに限定されたプライベート IP アドレスを管理するための IP アドレスリソースの作成

AWS 上の高可用性 (HA) クライアントが、単一の AWS アベイラビリティーゾーン (AZ) 内でのみ移動できるプライベート IP アドレスを使用する RHEL 8 ノードにアクセスできるようにするには、仮想 IP アドレスを使用するように **AWS セカンダリープライベート IP アドレス** (**awsvip**) リソースを設定します。

クラスター内の任意のノードで次の手順を実行できます。

前提条件

- 以前に設定されたクラスターがある。
- クラスターノードが RHEL HA リポジトリーにアクセスできる。詳細は、高可用性パッケージとエージェントのインストール を参照してください。
- AWS CLI を設定している。手順については、Installing the AWS CLI を参照してください。

手順

1. **resource-agents** パッケージをインストールします。

```
# yum install resource-agents
```

2. **オプション: awsvip** の説明を表示します。これは、このエージェントのオプションとデフォルトの操作を示しています。

```
# pcs resource describe awsvip
```

3. **VPC CIDR** ブロック内の未使用のプライベート IP アドレスを使用して、セカンダリープライベート IP アドレスを作成します。さらに、セカンダリープライベート IP アドレスが属するリソースグループを作成します。

```
# pcs resource create <resource-id> awsvip secondary_private_ip=<Unused-IP-Address> --group <group-name>
```

以下に例を示します。

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

4. 仮想 IP リソースを作成します。これは、フェンシングされたノードからフェイルオーバーノードに即時に再マッピングできる VPC IP アドレスで、サブネット内のフェンスされたノードの失敗をマスクします。仮想 IP が、前の手順で作成したセカンダリープライベート IP アドレスと同じリソースグループに属していることを確認します。

```
# pcs resource create <resource-id> IPaddr2 ip=<secondary-private-IP> --group <group-name>
```

以下に例を示します。

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

検証

- クラスターのステータスを表示して、必要なリソースが実行していることを確認します。

```
# pcs status
```

次の出力は、**vip** および **privip** リソースが **networking-group** リソースグループの一部として起動している実行中のクラスターの例を示しています。

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

3 nodes configured
3 resources configured

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

Full list of resources:
```

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

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

4.11.3. 複数の AWS アベイラビリティーゾーン間で移動できる IP アドレスを管理するための IP アドレスリソースの作成

AWS 上の高可用性 (HA) クライアントが、同じ AWS リージョン内の複数の AWS アベイラビリティーゾーン間で移動できる RHEL 8 ノードにアクセスできるようにするには、**aws-vpc-move-ip** リソースを設定して、Elastic IP アドレスを使用します。

前提条件

- 以前に設定されたクラスターがある。
- クラスターノードが RHEL HA リポジトリーにアクセスできる。詳細は、高可用性パッケージとエージェントのインストール を参照してください。
- AWS CLI を設定している。手順については、Installing the AWS CLI を参照してください。
- アイデンティティーおよびアクセス管理 (IAM) ユーザーがクラスターに設定されており、次の権限を持ちます。
  - ルーティングテーブルを変更する
  - セキュリティーグループを作成する
  - IAM ポリシーとロールを作成する

手順

1. **resource-agents** パッケージをインストールします。

```
# yum install resource-agents
```

2. **オプション: aws-vpc-move-ip** の説明を表示します。これは、このエージェントのオプションとデフォルトの操作を示しています。

```
# pcs resource describe aws-vpc-move-ip
```

3. IAM ユーザーに対して **OverlayIPAgent** IAM ポリシーを設定します。
   a. AWS コンソールで、**Services → IAM → Policies → Create OverlayIPAgent Policy** に移動します。
   b. 次の設定を入力し、**<region>**、**<account-id>**、および **<ClusterRouteTableID>** の値をクラスターに合わせて変更します。

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
      {
        "Sid": "Stmt1424870324000",
        "Effect": "Allow",
        "Action":  "ec2:DescribeRouteTables",
        "Resource": "*"
      },
      {
        "Sid": "Stmt1424860166260",
        "Action": [
          "ec2:CreateRoute",
          "ec2:ReplaceRoute"
        ],
        "Effect": "Allow",
        "Resource": "arn:aws:ec2:<region>:<account-id>:route-
table/<ClusterRouteTableID>"
      }
    ]
}
```

4. AWS コンソールで、クラスター内のすべてのノードの **Source/Destination Check** 機能を無効にします。
これを行うには、各ノードを右クリックして、**Networking → Change Source/Destination Checks** を選択します。表示されるポップアップメッセージで、**Yes, Disable** をクリックします。

5. クラスターのルートテーブルを作成します。これを行うには、クラスター内の 1 つのノードで次のコマンドを使用します。

```
# aws ec2 create-route --route-table-id <ClusterRouteTableID> --destination-cidr-block
<NewCIDRblockIP/NetMask> --instance-id <ClusterNodeID>
```

コマンドでは、値を次のように置き換えます。

- **ClusterRouteTableID**: 既存のクラスター VPC ルートテーブルのルートテーブル ID。
- **NewCIDRblockIP/NetMask**: VPC Classless Inter-Domain Routing (CIDR) ブロック外の新しい IP アドレスとネットマスク。たとえば、VPC CIDR ブロックが **172.31.0.0/16** の場合、新しい IP アドレス/ネットマスクは **192.168.0.15/32** になります。
- **ClusterNodeID**: クラスター内の別のノードのインスタンス ID。

6. クラスター内のノードの 1 つで、クライアントがアクセスできる空き IP アドレスを使用する **aws-vpc-move-ip** リソースを作成します。次の例では、IP **192.168.0.15** を使用する **vpcip** という名前のリソースを作成します。

```
# pcs resource create vpcip aws-vpc-move-ip ip=192.168.0.15 interface=eth0
routing_table=<ClusterRouteTableID>
```

7. クラスター内のすべてのノードで、**/etc/hosts/** ファイルを編集し、新しく作成されたリソースの IP アドレスを含む行を追加します。以下に例を示します。

```
192.168.0.15 vpcip
```

検証

1. 新しい **aws-vpc-move-ip** リソースのフェイルオーバー機能をテストします。

```
# pcs resource move vpcip
```

2. フェイルオーバーが成功した場合は、**vpcip** リソースの移動後に自動的に作成された制約を削除します。

```
# pcs resource clear vpcip
```

関連情報

- IAM ユーザー

4.11.4. 関連情報

- High Availability Add-On の概要
- 高可用性クラスターの設定および管理
- **aws-vpc-move-ip** の設定例
- **aws-vpc-route53** を使用した高可用性 AWS サービスの DNS 名フェイルオーバー

4.12. 共有ブロックストレージの設定

追加のストレージリソースを作成するには、Amazon Elastic Block Storage (EBS) マルチアタッチボリュームを使用して、Red Hat High Availability クラスターの共有ブロックストレージを設定できます。この手順はオプションであり、以下の手順では、1 TB の共有ディスクを持つ 3 つのインスタンス (3 ノードクラスター) を想定していることに注意してください。

前提条件

- AWS Nitro システムベースの Amazon EC2 インスタンス を使用している必要があります。

手順

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

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

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

```
$ aws ec2 create-volume --availability-zone us-east-1a --no-encrypted --size 1024 --volume-type io1 --iops 51200 --multi-attach-enabled

{
    "AvailabilityZone": "us-east-1a",
    "CreateTime": "2020-08-27T19:16:42.000Z",
    "Encrypted": false,
```

```
    "Size": 1024,
    "SnapshotId": "",
    "State": "creating",
    "VolumeId": "vol-042a5652867304f09",
    "Iops": 51200,
    "Tags": [ ],
    "VolumeType": "io1"
}
```

注記

次の手順で **VolumeId** が必要になります。

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

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

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

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

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

検証

1. クラスター内の各インスタンスについて、インスタンスの **<ip_address>** を指定して **SSH** コマンドを使用して、ブロックデバイスが利用可能になっていることを確認します。

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

たとえば、以下のコマンドは、インスタンス IP が **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
nvme2n1 259:1    0   1T  0 disk
```

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

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

たとえば、以下のコマンドは、インスタンスの 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 が含まれる詳細を一覧表示します。

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

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

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

4.13. 関連情報

- Red Hat Cloud Access リファレンスガイド
- Red Hat in the Public Cloud
- Red Hat Enterprise Linux on Amazon EC2 - FAQs
- Amazon EC2 での設定
- Red Hat on Amazon Web Services