Red Hat OpenStack Platform 16.2

スタンドアロンデプロイメントガイド

テストおよび概念実証環境用のオールインワン OpenStack クラウドの作成

Last Updated: 2022-05-14

Red Hat OpenStack Platform 16.2 スタンドアロンデプロイメントガイド

テストおよび概念実証環境用のオールインワン OpenStack クラウドの作成

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律上の通知

概要

Red Hat OpenStack Platform のスタンドアロン環境で、テスト環境で Red Hat OpenStack Platform をインストール、設定、およびデプロイします。本ガイドを使用して、シンプルな単一ノードの OpenStack クラウドをデプロイします。

目次

前書き

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、弊社 の CTO、Chris Wright のメッセージ を参照してください。

RED HAT ドキュメントへのフィードバック (英語のみ)

弊社ドキュメントに対するご意見をお聞かせください。ドキュメントの改善点があればお知らせください。

ドキュメントへのダイレクトフィードバック (DDF) 機能の使用 (英語版のみ)

特定の文章、段落、またはコードブロックに対して直接コメントを送付するには、DDF の **Add Feedback** 機能を使用してください。なお、この機能は英語版のドキュメントでのみご利用いただけます。

1. **Multi-page HTML** 形式でドキュメントを表示します。
2. ドキュメントの右上隅に **Feedback** ボタンが表示されていることを確認してください。
3. コメントするテキスト部分をハイライト表示します。
4. **Add Feedback** をクリックします。
5. **Add Feedback** フィールドにコメントを入力します。
6. (オプション) ドキュメントチームが連絡を取り問題についてお伺いできるように、ご自分のメールアドレスを追加します。
7. **Submit** をクリックします。

第1章 オールインワン RED HAT OPENSTACK PLATFORM のインストール

オールインワンのインストール手法では、TripleO を使用して、シンプルな単一ノード環境で Red Hat OpenStack Platform および関連サービスをデプロイします。このインストールを使用して、単一ノードの概念実証、開発、およびテスト用デプロイメントを有効にします。この場合、フォローアップ操作は限定的であるか、あるいはまったく行われません。

注記

この機能は、本リリースでは **テクノロジープレビュー** として提供しているため、Red Hat では全面的にはサポートしていません。これは、テスト目的のみでご利用いただく機能で、実稼働環境にデプロイすべきではありません。テクノロジープレビュー機能についての詳しい情報は、「対象範囲の詳細」を参照してください。

1.1. 前提条件

- システムに Red Hat Enterprise Linux 8.4 ベースオペレーティングシステムがインストールされている。
- TripleO が 2 番目のインターフェースを設定する間にインターネット接続が中断されないように、システムには 2 つのネットワークインターフェースが必要です。
- システムには、4 つの CPU、8 GB のRAM、および 30 GB のディスク領域が必要です。

ネットワーク構成の例

- **default** ネットワーク 192.168.122.0/24 に割り当てられたインターフェース **eth0**。一般的な接続には、このインターフェースを使用します。このインターフェースには、インターネットアクセスが必要です。
- **management** ネットワークの 192.168.25.0/24 に割り当てられたインターフェース **eth1**。TripleO は、OpenStack サービス用にこのインターフェースを使用します。

第2章 オールインワン RED HAT OPENSTACK PLATFORM 環境の概要

本セクションでは、シンプルな単一ノードの Red Hat OpenStack Platform 環境をインストール、設定、およびデプロイする方法について説明します。このシナリオでは、既存のアンダークラウドとの依存関係はありません。その代わりに、インストーラーはインラインの heat-all インスタンスを実行してデプロイプロセスのブートストラップを行い、選択した heat テンプレートをローカルマシンで実行可能な Ansible Playbook に変換します。

基本的なテストおよび開発用に、オールインワンのインストールを使用します。オールインワンのインストールは、Red Hat OpenStack Platform を初めて使用する場合やテスト環境を構築する場合には適していますが、複雑な操作を実行する場合は、プロダクションレベルのスケーリングに対応したクラウドをデプロイする必要があります。

ワークフロー

シンプルな単一ノードの Red Hat OpenStack Platform 環境をインストール、設定、およびデプロイするには、以下に示す基本ワークフローのタスクを完了します。

1. 環境を準備する。
2. オールインワン環境用のパッケージをインストールする。
3. オールインワン環境を設定する。
4. オールインワン環境をデプロイする。

オールインワンインストールのメリット

- コンポーザブルサービス
- 事前定義済みのロール
- 凝縮された単一ノード環境
- コンテナーでフットプリントの小さなインストーラーを実行し、Ansible Playbook を生成するのに使用できる Playbook

設定

試験的にロールおよびサービスを設定する場合は、「8章 *カスタムロールおよびカスタムサービスの使用*」および「コア heat テンプレート」を参照してください。

コンポーザブルロール

カスタムのコンポーザブルロールを作成し、それぞれのロールに特定のサービスをデプロイすることができます。

Ansible

このインストールでは、デプロイメントコマンドにより Ansible Playbooks が自動的に適用されます。デプロイコマンドの設定により、他の環境で使用することのできる Ansible Playbook を出力することもできます。たとえば、オールインワンのインストールでテストを完了し、検証済みの Ansible Playbook を他の環境に適用することができます。

第3章 オールインワン RED HAT OPENSTACK PLATFORM 環境のインストール

オールインワン環境の設定、デプロイ、およびテストを開始する前に、root 以外のユーザーを設定し、必要なパッケージおよび依存関係をインストールする必要があります。

1. オールインワンのホストに root 以外のユーザーを作成します。

```
[root@all-in-one]# useradd stack
```

2. **stack** ユーザーのパスワードを設定します。

```
[root@all-in-one]# passwd stack
```

3. **sudo** を **stack** ユーザーとして使用する場合は、パスワードを要求されないようにします。

```
[root@all-in-one]# echo "stack ALL=(root) NOPASSWD:ALL" | tee -a /etc/sudoers.d/stack
[root@all-in-one]# chmod 0440 /etc/sudoers.d/stack
```

4. オールインワンのホストに root 以外のユーザーとしてログインします。

```
$ ssh stack@<all-in-one>
```

5. マシンを Red Hat Subscription Manager に登録します。要求されたら、Red Hat サブスクリプションの認証情報を入力します。

```
[stack@all-in-one]$ sudo subscription-manager register
```

6. Red Hat サブスクリプションをエンタイトルメントサーバーにアタッチします。

```
[stack@all-in-one]$ sudo subscription-manager attach --auto
```

注記

--auto オプションを使用すると、正しいサブスクリプションプールをサブスクライブしない場合があります。正しいプールをサブスクライブするようにしてください。そうでないと、このインストールに必要なすべてのリポジトリーを有効にすることができない場合があります。**subscription-manager list --all --available** コマンドを使用して、正しいプール ID を特定します。

7. アンダークラウドを Red Hat Enterprise Linux 8.4 にロックします。

```
$ sudo subscription-manager release --set=8.4
```

8. 以下のコマンドを実行し、**dnf-utils** をインストールし、すべてのデフォルトリポジトリーを無効にしてから、必要なリポジトリーを有効にします。

```
[stack@all-in-one]$ sudo dnf install -y dnf-utils
[stack@all-in-one]$ sudo subscription-manager repos --disable=*
[stack@all-in-one]$ sudo subscription-manager repos \
--enable=rhel-8-for-x86_64-baseos-eus-rpms \
```

```
--enable=rhel-8-for-x86_64-appstream-eus-rpms \
--enable=rhel-8-for-x86_64-highavailability-eus-rpms \
--enable=ansible-2.9-for-rhel-8-x86_64-rpms \
--enable=openstack-16.2-for-rhel-8-x86_64-rpms \
--enable=fast-datapath-for-rhel-8-x86_64-rpms
```

注記

オールインワン環境は、Red Hat OpenStack Platform 16 ではテクノロジープレビューの機能です。

9. **container-tools** モジュールバージョンを設定します。

```
[stack@all-in-one]$ sudo dnf module disable -y container-tools:rhel8
[stack@all-in-one]$ sudo dnf module enable -y container-tools:3.0
```

10. ベースオペレーティングシステムを更新し、システムをリブートします。

```
[stack@all-in-one]$ sudo dnf update
[stack@all-in-one]$ sudo reboot
```

11. リブート後に再びホストにログインします。

12. TripleO コマンドラインインターフェース (CLI) をインストールします。

```
[stack@all-in-one]$ sudo dnf install -y python3-tripleoclient
```

第4章 オールインワン RED HAT OPENSTACK PLATFORM 環境の設定

オールインワンの Red Hat OpenStack Platform 環境をデプロイする前に、以下の設定ファイルを手動で作成する必要があります。

- **$HOME/containers-prepare-parameters.yaml**
- **$HOME/standalone_parameters.yaml**

開発またはテスト用にオールインワン環境をカスタマイズする場合は、以下の設定ファイルを編集します。

- **/usr/share/openstack-tripleo-heat-templates/environments/standalone/standalone-tripleo.yaml**
- **/usr/share/openstack-tripleo-heat-templates/roles/Standalone.yaml**

4.1. オールインワン RED HAT OPENSTACK PLATFORM (RHOSP) 環境用の YAML ファイルの生成

containers-prepare-parameter.yaml および **standalone_parameters.yaml** ファイルを生成するには、以下の手順を実行します。

1. デフォルトの **ContainerImagePrepare** パラメーターが含まれる **containers-prepare-parameters.yaml** ファイルを生成します。

```
[stack@all-in-one]$ sudo openstack tripleo container image prepare default --output-env-file $HOME/containers-prepare-parameters.yaml
```

2. **containers-prepare-parameters.yaml** ファイルを編集し、**ContainerImageRegistryCredentials** パラメーターにご自分の Red Hat 認証情報を追加します。これにより、デプロイプロセスで registry.redhat.io との認証を行い、コンテナーイメージを正常にプルすることができます。

```
parameter_defaults:
  ContainerImagePrepare:
  ...
  ContainerImageRegistryCredentials:
    registry.redhat.io:
      <USERNAME>: "<PASSWORD>"
```

注記

パスワードをプレーンテキストで入力するのを避けるためには、Red Hat サービスアカウントを作成します。詳しくは、「Red Hat コンテナーレジストリーの認証」を参照してください。

3. **containers-prepare-parameters.yaml** で **ContainerImageRegistryLogin** パラメーターを **true** に設定します。

```
parameter_defaults:
  ContainerImagePrepare:
```

```
...
ContainerImageRegistryCredentials:
  registry.redhat.io:
    <USERNAME>: "<PASSWORD>"
ContainerImageRegistryLogin: true
```

オールインワンのホストをコンテナーレジストリーとして使用する場合は、このパラメーターを省略し、**openstack tripleo container image prepare** コマンドに **--local-push-destination** を追加します。詳細については、「Preparing container images」を参照してください。

4. **$HOME/standalone_parameters.yaml** ファイルを作成し、ネットワーク設定や一部のデプロイメントオプション等、オールインワン RHOSP 環境の基本パラメーターを設定します。以下の例では、ネットワークインターフェース **eth1** は、RHOSP のデプロイに使用する管理ネットワーク上のインターフェースです。**eth1** の IP アドレスは 192.168.25.2 です。

```
[stack@all-in-one]$ export IP=192.168.25.2
[stack@all-in-one]$ export NETMASK=24
[stack@all-in-one]$ export INTERFACE=eth1
[stack@all-in-one]$ export DNS1=1.1.1.1
[stack@all-in-one]$ export DNS2=8.8.8.8

[stack@all-in-one]$ cat <<EOF > $HOME/standalone_parameters.yaml
parameter_defaults:
  CloudName: $IP
  CloudDomain: <DOMAIN_NAME>
  ControlPlaneStaticRoutes: []
  Debug: true
  DeploymentUser: $USER
  DnsServers:
    - $DNS1
    - $DNS2
  NeutronPublicInterface: $INTERFACE
  NeutronDnsDomain: localdomain
  NeutronBridgeMappings: datacentre:br-ctlplane
  NeutronPhysicalBridge: br-ctlplane
  StandaloneEnableRoutedNetworks: false
  StandaloneHomeDir: $HOME
  StandaloneLocalMtu: 1500
EOF
```

DnsServers パラメーターにご自分の DNS アドレスを設定する必要があります。このアドレスは、**/etc/resolv.conf** ファイルで確認することができます。

```
[stack@all-in-one]$ cat /etc/resolv.conf
192.168.122.1
```

単一のネットワークインターフェースのみを使用する場合は、デフォルトのルートを定義する必要があります。

```
ControlPlaneStaticRoutes:
  - ip_netmask: 0.0.0.0/0
    next_hop: $GATEWAY
    default: true
```

内部の時刻ソースがある場合、または環境が外部の時刻ソースへのアクセスをブロックする場合は、**NtpServer** パラメーターを使用して、使用する時刻ソースを定義します。

```
parameter_defaults:
  NtpServer: clock.example.com
```

仮想環境でオールインワンの RHOSP インストールを使用する場合は、**StandaloneExtraConfig** パラメーターで仮想化の種別を定義する必要があります。

```
parameter_defaults:
  StandaloneExtraConfig:
    NovaComputeLibvirtType: qemu
```

Load-balancing サービス (octavia) には、SSH の設定は必要ありません。ただし、負荷分散インスタンス (amphorae) への SSH アクセスが必要な場合は、**OctaviaAmphoraSshKeyFile** パラメーターを追加して stack ユーザーの公開鍵ファイルへの絶対パスの値を指定します (例: **OctaviaAmphoraSshKeyFile: "/home/stack/.ssh/id_rsa.pub"**)。

第5章 オールインワン RED HAT OPENSTACK PLATFORM 環境のデプロイメント

オールインワン環境をデプロイするには、次の手順を実施します。

1. Red Hat の認証情報を使用して registry.redhat.io にログインします。

```
[stack@all-in-one]$ sudo podman login registry.redhat.io
```

2. デプロイメントコマンドで使用する環境変数をエクスポートします。以下の例では、IP アドレスに管理ネットワーク上の 192.168.25.2 が設定された **eth1** インターフェースと共に、オールインワン環境をデプロイします。

```
[stack@all-in-one]$ export IP=192.168.25.2
[stack@all-in-one]$ export NETMASK=24
[stack@all-in-one]$ export INTERFACE=eth1
```

3. プロイメントコマンドを実行します。ご自分の環境に該当するすべての **.yaml** ファイルを指定するようにしてください。

```
[stack@all-in-one]$ sudo openstack tripleo deploy \
  --templates \
  --local-ip=$IP/$NETMASK \
  -e /usr/share/openstack-tripleo-heat-templates/environments/standalone/standalone-tripleo.yaml \
  -r /usr/share/openstack-tripleo-heat-templates/roles/Standalone.yaml \
  -e $HOME/containers-prepare-parameters.yaml \
  -e $HOME/standalone_parameters.yaml \
  --output-dir $HOME \
  --standalone
```

デプロイメントが正常に完了したら、**/home/$USER/.config/openstack** ディレクトリーの **clouds.yaml** 設定ファイルを使用して、OpenStack サービスにクエリーを行い検証することができます。

```
[stack@all-in-one]$ export OS_CLOUD=standalone
[stack@all-in-one]$ openstack endpoint list
```

ダッシュボードにアクセスするには、http://192.168.25.2/dashboard にアクセスし、**~/standalone-passwords.conf** ファイルのデフォルトのユーザー名 **admin** および **undercloud_admin_password** を使用します。

```
[stack@all-in-one]$ cat standalone-passwords.conf | grep undercloud_admin_password:
```

第6章 オールインワン RED HAT OPENSTACK PLATFORM 環境を使用した ANSIBLE PLAYBOOK の作成

デプロイメントコマンドは、Ansible Playbook を自動的に環境に適用します。ただし、デプロイメントコマンドを変更して、Ansible Playbook をデプロイメントに適用せずに生成し、後で Playbook を実行することができます。

deploy コマンドに **--output-only** オプションを追加して、**standalone-ansible-XXXXX** ディレクトリーを生成します。このディレクトリーには、他のホストで実行することのできる Ansible Playbook のセットが含まれます。

1. Ansible Playbook のディレクトリーを生成するには、オプション **--output-only** を指定してデプロイコマンドを実行します。

```
[stack@all-in-one]$ sudo openstack tripleo deploy \
  --templates \
  --local-ip=$IP/$NETMASK \
  -e /usr/share/openstack-tripleo-heat-templates/environments/standalone/standalone-tripleo.yaml \
  -r /usr/share/openstack-tripleo-heat-templates/roles/Standalone.yaml \
  -e $HOME/containers-prepare-parameters.yaml \
  -e $HOME/standalone_parameters.yaml \
  --output-dir $HOME \
  --standalone \
  --output-only
```

2. Ansible Playbook を実行するには、**ansible-playbook** コマンドを実行し、**inventory.yaml** ファイルおよび **deploy_steps_playbook.yaml** ファイルを指定します。

```
[stack@all-in-one]$ cd standalone-ansible-XXXXX
[stack@all-in-one]$ sudo ansible-playbook -i inventory.yaml deploy_steps_playbook.yaml
```

第7章 HEAT テンプレートの使用

本章のカスタム設定では、heat テンプレートおよび環境ファイルを使用して、オーバークラウドの特定の機能を定義します。本章では、Red Hat OpenStack Platform に関して、heat テンプレートの構成の基本概要について説明します。テンプレートの目的は、heat が作成するリソースのコレクションであるスタックを定義および作成し、リソースを設定することです。リソースとは、コンピュートリソース、ネットワーク設定、セキュリティーグループ、スケーリングルール、カスタムリソースなどの OpenStack のオブジェクトを指します。

heat テンプレートは、3 つの主要なセクションで構成されます。

パラメーター

パラメーターは、heat に渡される設定です。これらのパラメーターを使用して、デフォルト値およびデフォルト以外の値の両方を定義およびカスタマイズします。テンプレートの parameters セクションで、これらのパラメーターを定義します。

リソース

リソースは、スタックの一部として作成および設定する固有のオブジェクトです。OpenStack には、全コンポーネントに対応するコアリソースのセットが含まれています。テンプレートの resources セクションで、リソースを定義します。

アウトプット

これらは、スタックの作成後に heat から渡される値です。これらの値には、heat API またはクライアントツールのいずれかを使用してアクセスすることができます。テンプレートの output セクションで、これらの値を定義します。

heat がテンプレートを処理する際には、テンプレートのスタックとリソーステンプレートの子スタックセットを作成します。このスタック階層は、テンプレートで定義するメインのスタックに由来します。以下のコマンドを使用して、スタックの階層を表示することができます。

```
$ heat stack-list --show-nested
```

7.1. コア HEAT テンプレート

Red Hat OpenStack Platform には、オーバークラウド用のコア heat テンプレートコレクションが含まれています。このコレクションは、**/usr/share/openstack-tripleo-heat-templates** ディレクトリーにあります。

このコレクションには、多数の heat テンプレートおよび環境ファイルが含まれます。本セクションでは、デプロイメントをカスタマイズするのに使用できる主要なファイルおよびディレクトリーについて説明します。

overcloud.j2.yaml

このファイルは、オーバークラウド環境の作成に使用するメインのテンプレートファイルです。このファイルでは Jinja2 構文を使用してテンプレートの特定セクションを繰り返し、カスタムロールを作成します。Jinja2 フォーマットは、オーバークラウドのデプロイメントプロセス中に YAML にレンダリングされます。

overcloud-resource-registry-puppet.j2.yaml

このファイルは、オーバークラウド環境の作成に使用するメインの環境ファイルです。このファイルには、オーバークラウドイメージ上の Puppet モジュールの設定セットが含まれます。director により各ノードにオーバークラウドのイメージが書き込まれると、heat はこの環境ファイルに登録されているリ

ソースを使用して各ノードの Puppet 設定を開始します。このファイルでは Jinja2 構文を使用してテンプレートの特定セクションを繰り返し、カスタムロールを作成します。Jinja2 フォーマットは、オーバークラウドのデプロイメントプロセス中に YAML にレンダリングされます。

roles_data.yaml

このファイルにはオーバークラウド内のロールの定義が含まれ、サービスを各ロールにマッピングします。

network_data.yaml

このファイルには、オーバークラウド内のネットワーク、およびそれらのサブネット、割り当てプール、VIP ステータス等の属性の定義が含まれます。デフォルトの **network_data.yaml** ファイルにはデフォルトのネットワーク (External、Internal Api、Storage、Storage Management、Tenant、Management) のみが含まれます。カスタムの **network_data.yaml** ファイルを作成し、**-n** オプションを使用して **openstack overcloud deploy** コマンドに追加することができます。

plan-environment.yaml

このファイルには、プラン名、使用するメインのテンプレート、オーバークラウドに適用する環境ファイル等、オーバークラウドプランのメタデータの定義が含まれます。

capabilities-map.yaml

このファイルには、オーバークラウドプランの環境ファイルのマッピングが含まれます。director の Web UI で環境ファイルを記述および有効化するには、このファイルを使用します。**environments** ディレクトリーにカスタム環境ファイルを含めても、これらのファイルを **capabilities-map.yaml** ファイルに定義しない場合、これらの環境ファイルは Web UI の **全体の設定** ページの **Other** サブタブに表示されます。

environments

このディレクトリーには、オーバークラウドの作成に使用可能なその他の heat 環境ファイルが含まれます。これらの環境ファイルは、Red Hat OpenStack Platform 環境の追加の機能を有効にします。たとえば、**cinder-netapp-config.yaml** 環境ファイルを使用して、Block Storage サービス (cinder) の NetApp バックエンドストレージを有効にすることができます。**environments** ディレクトリーにカスタム環境ファイルを含めても、これらのファイルを **capabilities-map.yaml** ファイルに定義しない場合、これらの環境ファイルは Web UI の **全体の設定** ページの **Other** サブタブに表示されます。

network

このディレクトリーには、分離ネットワークおよびポートを作成するのに使用できる heat テンプレートのセットが含まれます。

puppet

このディレクトリーには、puppet テンプレートが含まれます。**overcloud-resource-registry-puppet.j2.yaml** 環境ファイルは、**puppet** ディレクトリーのファイルを使用して、各ノードに Puppet の設定が適用されるようにします。

puppet/services

このディレクトリーには、コンポーザブルサービスアーキテクチャー内の全サービス用の heat テンプレートが含まれます。

extraconfig

このディレクトリーには、追加機能を有効にするのに使用できるテンプレートが含まれます。たとえば、**extraconfig/pre_deploy/rhel-registration** ディレクトリーを使用して、ノードを Red Hat コンテンツ配信ネットワークまたはご自分の Red Hat Satellite サーバーに登録することができます。

第8章 カスタムロールおよびカスタムサービスの使用

通常 Red Hat OpenStack Platform は、事前定義済みロールのノード (例: Controller ロール、Compute ロール、さまざまなストレージロール種別のノード) で構成されます。これらのデフォルトロールには、それぞれコアの heat テンプレートコレクションで定義するサービスのセットが含まれます。ただし、オールインワンの Red Hat OpenStack Platform インストールは、すべての OpenStack サービスが含まれる単一のノード上で実行されます。**/usr/share/openstack-tripleo-heat-templates/roles** ディレクトリーの **Standalone.yaml** ロールファイルは、オールインワンインストールのすべてのサービスが含まれる設定ファイルです。**Standalone.yaml** ロールファイルのコピーを作成して変更し、インストール内のサービスを有効および無効にすることができます。

Standalone.yaml ファイルには、ロール **Standalone** にサービスの一覧が含まれます。以下の例を使用して、このファイルの構文を説明します。

```
- name: Standalone
  description: |
    A standalone role that includes a minimal set of services. Use this role for testing in a single node
configuration with the 'openstack tripleo deploy --standalone' command, or with the 'openstack
overcloud deploy' command.
  CountDefault: 1
  tags:
    - primary
    - controller
  disable_constraints: True
  ServicesDefault:
    - OS::TripleO::Services::Aide
    - OS::TripleO::Services::AodhApi
    - OS::TripleO::Services::AodhEvaluator
    ...
    - OS::TripleO::Services::Tuned
    - OS::TripleO::Services::Vpp
    - OS::TripleO::Services::Zaqar
```

このロールファイルをデプロイコマンドに追加して、Standalone ロールでスタックを設定します。このロールには、ロールファイルの **ServicesDefault:** セクションに含めたサービスが含まれます。

```
[stack@all-in-one]$ sudo openstack tripleo deploy --templates -r /usr/share/openstack-tripleo-heat-
templates/roles/Standalone.yaml
```

ただし、プロダクションのマルチノード Red Hat OpenStack Platform 環境では、すべてのサービスを単一のノードに含めるのではなく、OpenStack サービスの一部が含まれるロールを各ノードに割り当てます。たとえば、デフォルトの Controller ロールには管理、ネットワーク、および高可用性サービスが含まれ、デフォルトの Compute ロールにはコンピューティングサービスが含まれます。マルチノード環境のデフォルトのロールファイルは、**/usr/share/openstack-tripleo-heat-templates/roles_data.yaml** ファイルです。このファイルで、以下のロール種別を定義します。

- Controller
- Compute
- BlockStorage
- ObjectStorage
- CephStorage

以下の例を使用して、マルチノード環境におけるロールの構文を説明します。

```
- name: Controller
  description: |
    Controller role that contains all of the services for database, messaging and network functions.
  ServicesDefault:
    - OS::TripleO::Services::AuditD
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CephClient
    ...
- name: Compute
  description: |
    Basic Compute Node role
  ServicesDefault:
    - OS::TripleO::Services::AuditD
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CephClient
    ...
```

デプロイメントコマンドを実行するたびに、ロールファイルを指定する必要があります。デプロイメントコマンドで **-r** 引数を使用して、このファイルをオーバーライドしてカスタムロールファイルを使用することができます。

```
[stack@all-in-one]$ sudo openstack tripleo deploy --templates -r ~/templates/roles_data-custom.yaml
```

8.1. オールインワン RED HAT OPENSTACK PLATFORM 環境でのサービスの有効化および無効化

/usr/share/openstack-tripleo-heat-templates/roles ディレクトリーの **Standalone.yaml** ロールファイルは、オールインワンインストールのすべてのサービスが含まれる設定ファイルです。個々のサービスを有効または無効にすることができます。

手順

1. サービスを無効にするには、新しい環境ファイルを作成し、無効にするサービスに値 **OS::Heat::None** を含めます。

   ```
   - OS::TripleO::Services::<SERVICE>: OS::Heat::None
   ```

   この環境ファイルをデプロイメントコマンドに追加します。

2. サービスを有効にするには、新しい環境ファイルを作成し、有効にするサービスから値 **OS::Heat::None** を削除します。

   ```
   - OS::TripleO::Services::<SERVICE>:
   ```

   この環境ファイルをデプロイメントコマンドに追加します。

第9章 例

以下の例を使用して、デプロイメント後に、さまざまなネットワーク設定でコンピュートインスタンスを起動する方法を説明します。

9.1. 例 1: プロジェクトネットワークおよびプロバイダーネットワークに 1 つの NIC を持つコンピュートノードの起動

この例を使用して、オールインワンの Red Hat OpenStack Platform 環境をデプロイした後に、プライベートプロジェクトネットワークおよびプロバイダーネットワークを設定してコンピュートノードを起動する方法を説明します。この例は単一 NIC 構成がベースで、少なくとも 3 つの IP アドレスが必要です。

前提条件

この例を正常に完了するには、ご自分の環境で以下の IP アドレスが利用可能でなければなりません。

- OpenStack サービス用に 1 つの IP アドレス
- プロジェクトネットワークへの接続を提供するために、仮想ルーター用に 1 つの IP アドレス。この例では、この IP アドレスは自動的に割り当てられます。
- プロバイダーネットワーク上の Floating IP 用に、少なくとも 1 つの IP アドレス

手順

1. 設定ヘルパー変数を作成します。

```
# standalone with project networking and provider networking
export OS_CLOUD=standalone
export GATEWAY=192.168.25.1
export STANDALONE_HOST=192.168.25.2
export PUBLIC_NETWORK_CIDR=192.168.25.0/24
export PRIVATE_NETWORK_CIDR=192.168.100.0/24
export PUBLIC_NET_START=192.168.25.4
export PUBLIC_NET_END=192.168.25.15
export DNS_SERVER=1.1.1.1
```

2. 基本のフレーバーを作成します。

```
$ openstack flavor create --ram 512 --disk 1 --vcpu 1 --public tiny
```

3. CirrOS をダウンロードし、OpenStack イメージを作成します。

```
$ wget https://download.cirros-cloud.net/0.4.0/cirros-0.4.0-x86_64-disk.img
$ openstack image create cirros --container-format bare --disk-format qcow2 --public --file
cirros-0.4.0-x86_64-disk.img
```

4. SSH を設定します。

```
$ ssh-keygen -m PEM -t rsa -b 2048 -f ~/.ssh/id_rsa_pem
$ openstack keypair create --public-key ~/.ssh/id_rsa_pem.pub default
```

5. シンプルなネットワークセキュリティーグループを作成します。

```
$ openstack security group create basic
```

6. 新しいネットワークセキュリティーグループを設定します。

   a. SSH を有効にします。

   ```
   $ openstack security group rule create basic --protocol tcp --dst-port 22:22 --remote-ip
   0.0.0.0/0
   ```

   b. ping を有効にします。

   ```
   $ openstack security group rule create --protocol icmp basic
   ```

   c. DNS を有効にします。

   ```
   $ openstack security group rule create --protocol udp --dst-port 53:53 basic
   ```

7. Neutron ネットワークを作成します。

```
$ openstack network create --external --provider-physical-network datacentre --provider-
network-type flat public
$ openstack network create --internal private
$ openstack subnet create public-net \
    --subnet-range $PUBLIC_NETWORK_CIDR \
    --no-dhcp \
    --gateway $GATEWAY \
    --allocation-pool start=$PUBLIC_NET_START,end=$PUBLIC_NET_END \
    --network public
$ openstack subnet create private-net \
    --subnet-range $PRIVATE_NETWORK_CIDR \
    --network private
```

8. 仮想ルーターを作成します。

```
# NOTE: In this case an IP will be automatically assigned
# from the allocation pool for the subnet.
$ openstack router create vrouter
$ openstack router set vrouter --external-gateway public
$ openstack router add subnet vrouter private-net
```

9. Floating IP を作成します。

```
$ openstack floating ip create public
```

10. インスタンスを起動します。

```
$ openstack server create --flavor tiny --image cirros --key-name default --network private --
security-group basic myserver
```

11. Floating IP を割り当てます。

```
$ openstack server add floating ip myserver <FLOATING_IP>
```

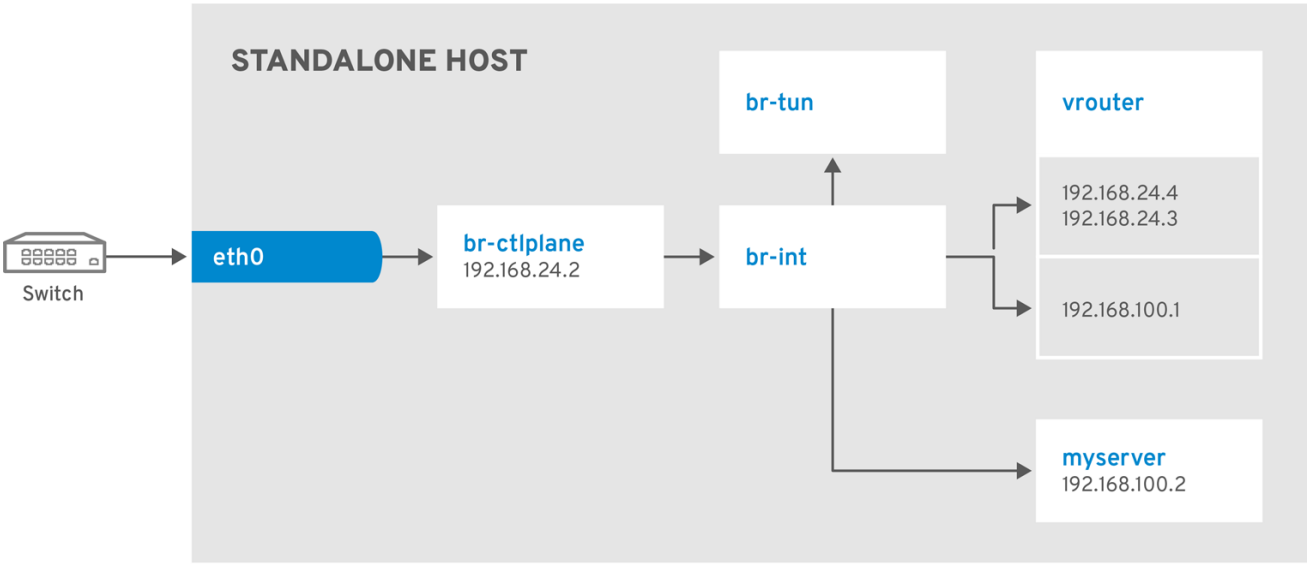
FLOATING_IP を、前のステップで作成した Floating IP のアドレスに置き換えます。

12. SSH をテストします。

```
ssh cirros@<FLOATING_IP>
```

FLOATING_IP を、前のステップで作成した Floating IP のアドレスに置き換えます。

ネットワークアーキテクチャー



9.2. 例 2: プロバイダーネットワークに 1 つの NIC を持つコンピュートノードの起動

この例を使用して、オールインワンの Red Hat OpenStack Platform 環境をデプロイした後に、プロバイダーネットワークを設定してコンピュートノードを起動する方法を説明します。この例は単一 NIC 構成がベースで、少なくとも 4 つの IP アドレスが必要です。

前提条件

この例を正常に完了するには、ご自分の環境で以下の IP アドレスが利用可能でなければなりません。

- OpenStack サービス用に 1 つの IP アドレス
- プロジェクトネットワークへの接続を提供するために、仮想ルーター用に 1 つの IP アドレス。この例では、この IP アドレスは自動的に割り当てられます。
- プロバイダーネットワーク上の DHCP 用に 1 つの IP アドレス
- プロバイダーネットワーク上の Floating IP 用に、少なくとも 1 つの IP アドレス

手順

1. 設定ヘルパー変数を作成します。

```
# standalone with project networking and provider networking
export OS_CLOUD=standalone
export GATEWAY=192.168.25.1
export STANDALONE_HOST=192.168.25.2
```

```
export VROUTER_IP=192.168.25.3
export PUBLIC_NETWORK_CIDR=192.168.25.0/24
export PUBLIC_NET_START=192.168.25.4
export PUBLIC_NET_END=192.168.25.15
export DNS_SERVER=1.1.1.1
```

2. 基本のフレーバーを作成します。

```
$ openstack flavor create --ram 512 --disk 1 --vcpu 1 --public tiny
```

3. CirrOS をダウンロードし、OpenStack イメージを作成します。

```
$ wget https://download.cirros-cloud.net/0.4.0/cirros-0.4.0-x86_64-disk.img
$ openstack image create cirros --container-format bare --disk-format qcow2 --public --file
cirros-0.4.0-x86_64-disk.img
```

4. SSH を設定します。

```
$ ssh-keygen -m PEM -t rsa -b 2048 -f ~/.ssh/id_rsa_pem
$ openstack keypair create --public-key ~/.ssh/id_rsa_pem.pub default
```

5. シンプルなネットワークセキュリティーグループを作成します。

```
$ openstack security group create basic
```

6. 新しいネットワークセキュリティーグループを設定します。

   a. SSH を有効にします。

```
$ openstack security group rule create basic --protocol tcp --dst-port 22:22 --remote-ip
0.0.0.0/0
```

   b. ping を有効にします。

```
$ openstack security group rule create --protocol icmp basic
```

   c. DNS を有効にします。

```
$ openstack security group rule create --protocol udp --dst-port 53:53 basic
```

7. Neutron ネットワークを作成します。

```
$ openstack network create --external --provider-physical-network datacentre --provider-
network-type flat public
$ openstack network create --internal private
$ openstack subnet create public-net \
    --subnet-range $PUBLIC_NETWORK_CIDR \
    --gateway $GATEWAY \
    --allocation-pool start=$PUBLIC_NET_START,end=$PUBLIC_NET_END \
    --network public \
    --host-route destination=169.254.169.254/32,gateway=$VROUTER_IP \
    --host-route destination=0.0.0.0/0,gateway=$GATEWAY \
    --dns-nameserver $DNS_SERVER
```

8. 仮想ルーターを作成します。

```
# NOTE: In this case an IP will be automatically assigned
# from the allocation pool for the subnet.
$ openstack router create vrouter
$ openstack port create --network public --fixed-ip subnet=public-net,ip-
address=$VROUTER_IP vrouter-port
$ openstack router add port vrouter vrouter-port
```

9. インスタンスを起動します。

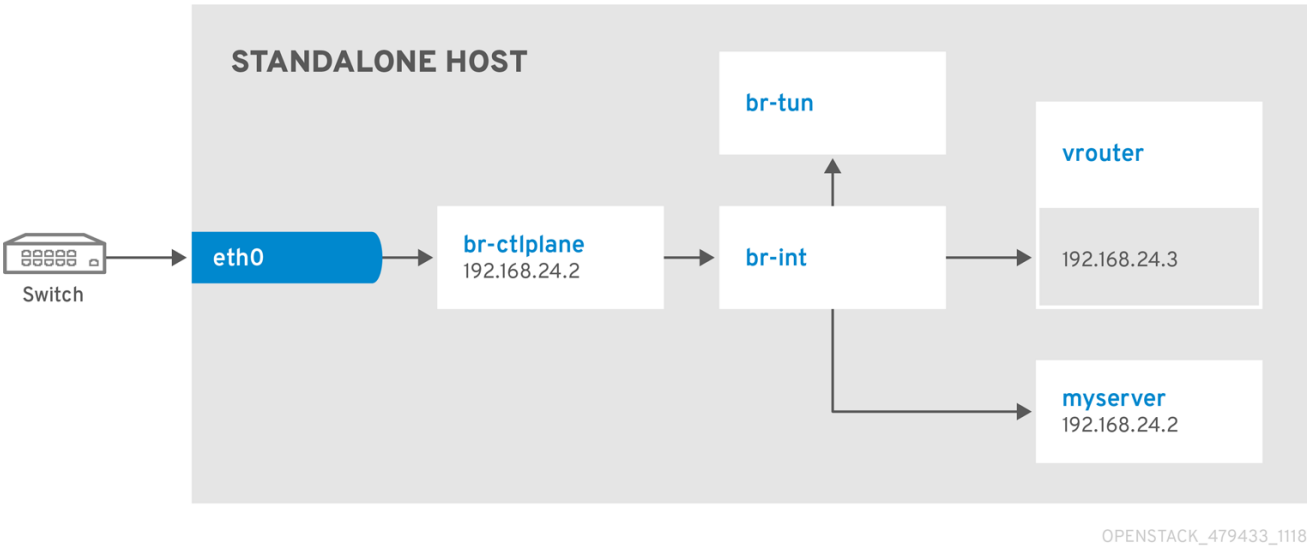
```
$ openstack server create --flavor tiny --image cirros --key-name default --network public --
security-group basic myserver
```

10. SSH をテストします。

```
ssh cirros@<VM_IP>
```

VM_IP を、前のステップで作成した仮想マシンのアドレスに置き換えます。

ネットワークアーキテクチャー



9.3. 例 3: プロジェクトネットワークおよびプロバイダーネットワークに 2 つの NIC を持つコンピュートノードの起動

この例を使用して、オールインワンの Red Hat OpenStack Platform 環境をデプロイした後に、プライベートプロジェクトネットワークおよびプロバイダーネットワークを設定してコンピュートノードを起動する方法を説明します。この例はデュアル NIC 構成がベースで、プロバイダーネットワークに少なくとも 4 つの IP アドレスが必要です。

前提条件

- プロバイダーネットワーク上のゲートウェイ用に 1 つの IP アドレス
- OpenStack のエンドポイント用に 1 つの IP アドレス

- プロジェクトネットワークへの接続を提供するために、仮想ルーター用に 1 つの IP アドレス。この例では、この IP アドレスは自動的に割り当てられます。
- プロバイダーネットワーク上の Floating IP 用に、少なくとも 1 つの IP アドレス

手順

1. 設定ヘルパー変数を作成します。

```
# standalone with project networking and provider networking
export OS_CLOUD=standalone
export GATEWAY=192.168.25.1
export STANDALONE_HOST=192.168.0.2
export PUBLIC_NETWORK_CIDR=192.168.25.0/24
export PRIVATE_NETWORK_CIDR=192.168.100.0/24
export PUBLIC_NET_START=192.168.25.3
export PUBLIC_NET_END=192.168.25.254
export DNS_SERVER=1.1.1.1
```

2. 基本のフレーバーを作成します。

```
$ openstack flavor create --ram 512 --disk 1 --vcpu 1 --public tiny
```

3. CirrOS をダウンロードし、OpenStack イメージを作成します。

```
$ wget https://download.cirros-cloud.net/0.4.0/cirros-0.4.0-x86_64-disk.img
$ openstack image create cirros --container-format bare --disk-format qcow2 --public --file
cirros-0.4.0-x86_64-disk.img
```

4. SSH を設定します。

```
$ ssh-keygen -m PEM -t rsa -b 2048 -f ~/.ssh/id_rsa_pem
$ openstack keypair create --public-key ~/.ssh/id_rsa_pem.pub default
```

5. シンプルなネットワークセキュリティーグループを作成します。

```
$ openstack security group create basic
```

6. 新しいネットワークセキュリティーグループを設定します。
   a. SSH を有効にします。

```
$ openstack security group rule create basic --protocol tcp --dst-port 22:22 --remote-ip
0.0.0.0/0
```

   b. ping を有効にします。

```
$ openstack security group rule create --protocol icmp basic
```

   c. DNS を有効にします。

```
$ openstack security group rule create --protocol udp --dst-port 53:53 basic
```

7. Neutron ネットワークを作成します。

```
$ openstack network create --external --provider-physical-network datacentre --provider-
network-type flat public
$ openstack network create --internal private
$ openstack subnet create public-net \
    --subnet-range $PUBLIC_NETWORK_CIDR \
    --no-dhcp \
    --gateway $GATEWAY \
    --allocation-pool start=$PUBLIC_NET_START,end=$PUBLIC_NET_END \
    --network public
$ openstack subnet create private-net \
    --subnet-range $PRIVATE_NETWORK_CIDR \
    --network private
```

8. 仮想ルーターを作成します。

```
# NOTE: In this case an IP will be automatically assigned
# from the allocation pool for the subnet.
$ openstack router create vrouter
$ openstack router set vrouter --external-gateway public
$ openstack router add subnet vrouter private-net
```

9. Floating IP を作成します。

```
$ openstack floating ip create public
```

10. インスタンスを起動します。

```
$ openstack server create --flavor tiny --image cirros --key-name default --network private --
security-group basic myserver
```

11. Floating IP を割り当てます。

```
$ openstack server add floating ip myserver <FLOATING_IP>
```

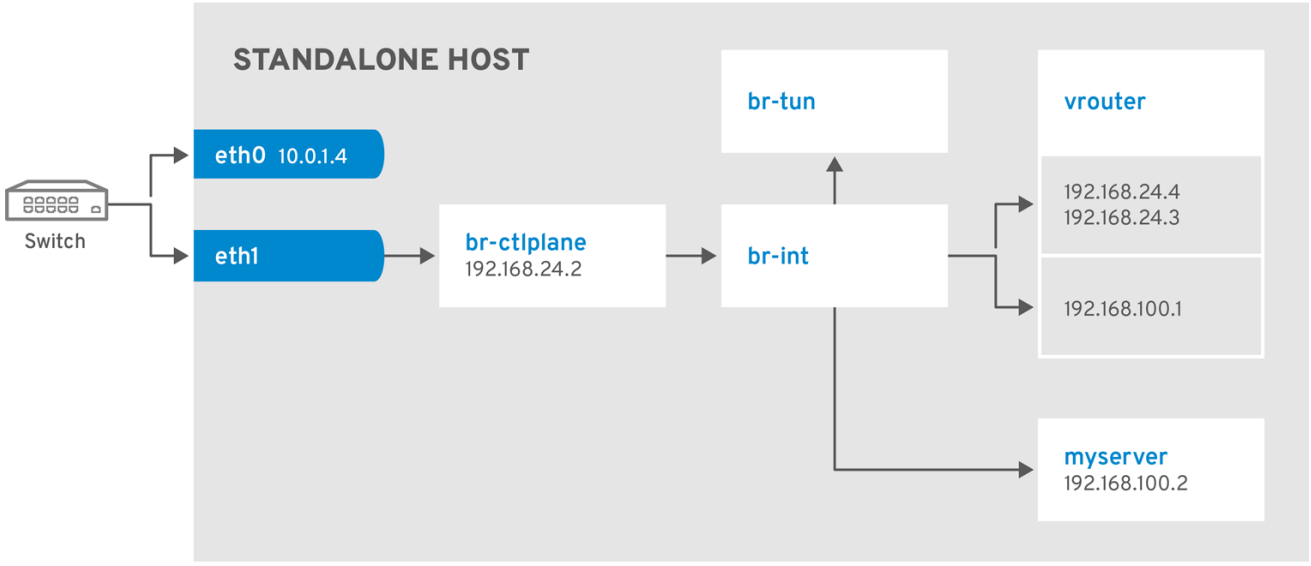
FLOATING_IP を、前のステップで作成した Floating IP のアドレスに置き換えます。

12. SSH をテストします。

```
ssh cirros@<FLOATING_IP>
```

FLOATING_IP を、前のステップで作成した Floating IP のアドレスに置き換えます。

ネットワークアーキテクチャー



OPENSTACK_479433_1118