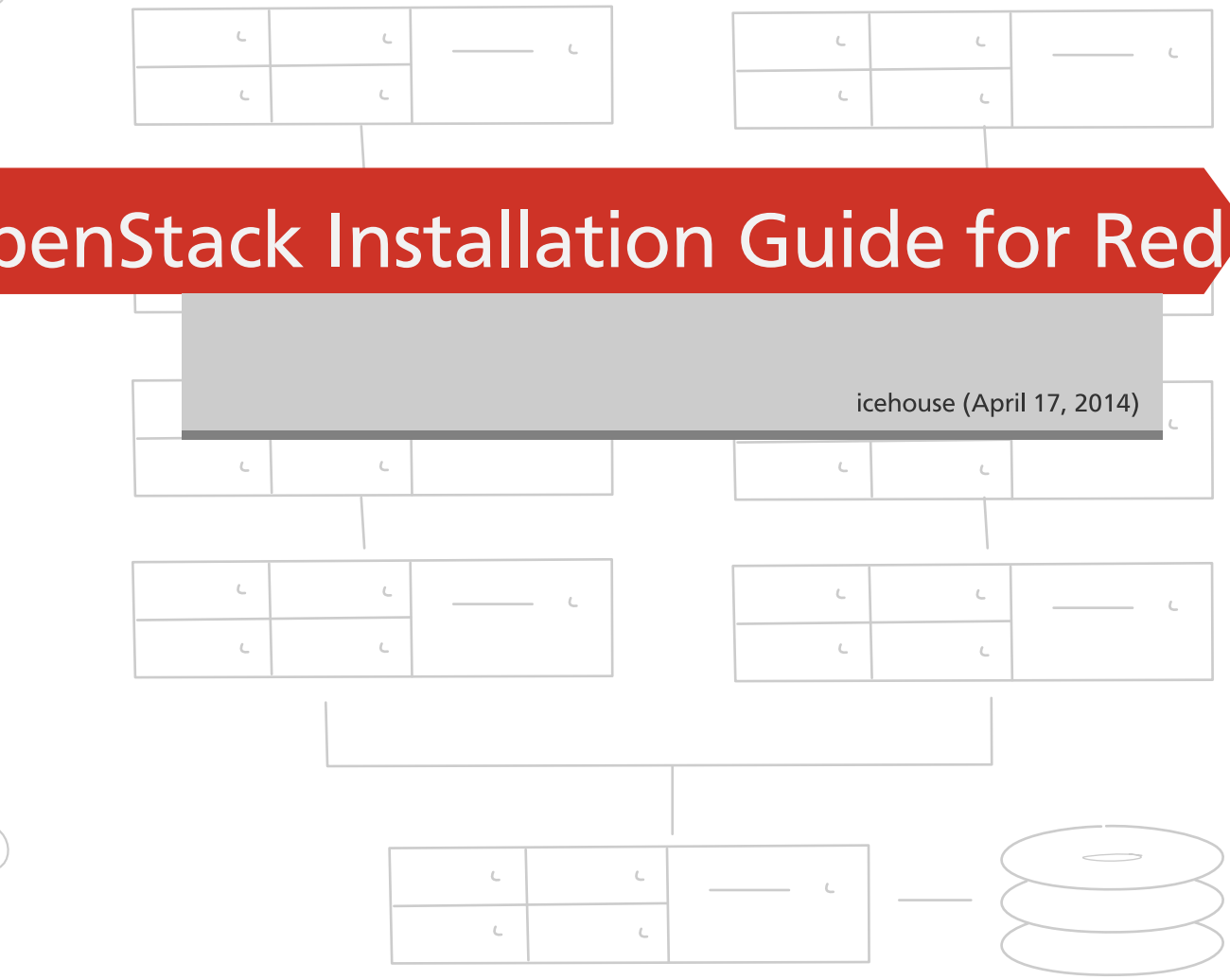
OpenStack Installation Guide for Red Ha

icehouse (April 17, 2014)



BUILT FOR

openstack™

CLOUD SOFTWARE

openstack™

docs.openstack.org

OpenStack Installation Guide for Red Hat Enterprise Linux, CentOS, and Fedora

[FAMILY Given]

icehouse (2014-04-17)

概要

The OpenStack® system consists of several key projects that you install separately but that work together depending on your cloud needs. These projects include Compute, Identity Service, Networking, Image Service, Block Storage, Object Storage, Telemetry, and Orchestration. You can install any of these projects separately and configure them stand-alone or as connected entities. This guide shows you how to install OpenStack by using packages available through Fedora 19 as well as on Red Hat Enterprise Linux and its derivatives through the EPEL repository. Explanations of configuration options and sample configuration files are included.

目次

図の一覧

表の一覧

はじめに

表記規則

The OpenStack documentation uses several typesetting conventions.

Notices

Notices take three forms:

注記

The information in a note is usually in the form of a handy tip or reminder.

重要

The information in an important notice is something you must be aware of before proceeding.

警告

The information in warnings is critical. Warnings provide additional information about risk of data loss or security issues.

コマンドプロンプト

Commands prefixed with the # prompt are to be executed by the root user. These examples can also be executed by using the sudo command, if available.

$ プロンプトから始まるコマンドは、root を含む、すべてのユーザーにより実行できます。

ドキュメント変更履歴

このバージョンのガイドはすべての旧バージョンを置き換え、廃止します。以下の表はもっとも最近の変更点を記載しています。

Revision Date	Summary of Changes
April 16, 2014	• Update for Icehouse, rework Networking setup to use ML2 as plugin, add new chapter for Database Service setup, improved basic configuration.
October 25, 2013	• Debian の初期サポートの追加。
October 17, 2013	• Havana リリース。
October 16, 2013	• SUSE Linux Enterprise のサポートの追加。
October 8, 2013	• Havana 向け再構成の完了。
September 9, 2013	• openSUSE 版の作成。
August 1, 2013	• Object Storage 検証手順の修正。バグ 1207347 の修正。
July 25, 2013	• cinder ユーザーの作成と service プロジェクトへの追加。バグ 1205057 の修正。

Revision Date	Summary of Changes
May 8, 2013	・ 一貫性のために文書名の更新。
May 2, 2013	・ 表紙の更新と付録の小さなミスの修正。

第1章 アーキテクチャー

目次

警告

Icehouse 向けにこのドキュメントを更新中です。この作業中、構造や内容の問題が見つかるかもしれません。

概要

OpenStack プロジェクトは、あらゆる種類のクラウド環境をサポートする、オープンソースのクラウドコンピューティングプラットフォームです。シンプルな実装、大規模なスケーラビリティ、豊富な機能を目指しています。世界中のクラウドコンピューティング技術者がプロジェクトに貢献しています。

OpenStack はさまざまな相補サービスを通して Infrastructure-as-a-Service (IaaS) ソリューションを提供します。各サービスはこの統合を促す Application Programming Interface (API) を提供します。以下の表は OpenStack サービスの一覧です。

表1.1 OpenStack のサービス

サービス	プロジェクト名	説明
Dashboard	Horizon	インスタンスの起動、IP アドレスの割り当て、アクセス制御の設定など、基礎となる OpenStack サービスを操作するために、ウェブベースのセルフサービスポータルを提供します。
Compute	Nova	Manages the lifecycle of compute instances in an OpenStack environment. Responsibilities include spawning, scheduling and decomissioning of virtual machines on demand.
Networking	Neutron	OpenStack Compute のような他の OpenStack サービスに対してサービスとしてのネットワーク接続性を可能にします。ユーザーがネットワークやそれらへの接続を定義するための API を提供します。数多くの人気のあるネットワークベンダーや技術をサポートする、プラグイン可能なアーキテクチャーを持ちます。
ストレージ		
Object Storage	Swift	RESTful、HTTP ベースの API 経由で任意の非構造データオブジェクトを保存および取得します。そのデータ複製およびスケールアウトアーキテクチャーで高い耐障害性を持ちます。その実装はマウント可能なディレクトリを持つファイルサーバーのようではありません。
Block Storage	Cinder	実行中のインスタンスに永続的なブロックストレージを提供します。そのプラグイン可能なドライバーアーキテクチャーにより、ブロックストレージデバイスの作成と管理が容易になります。
共有サービス		

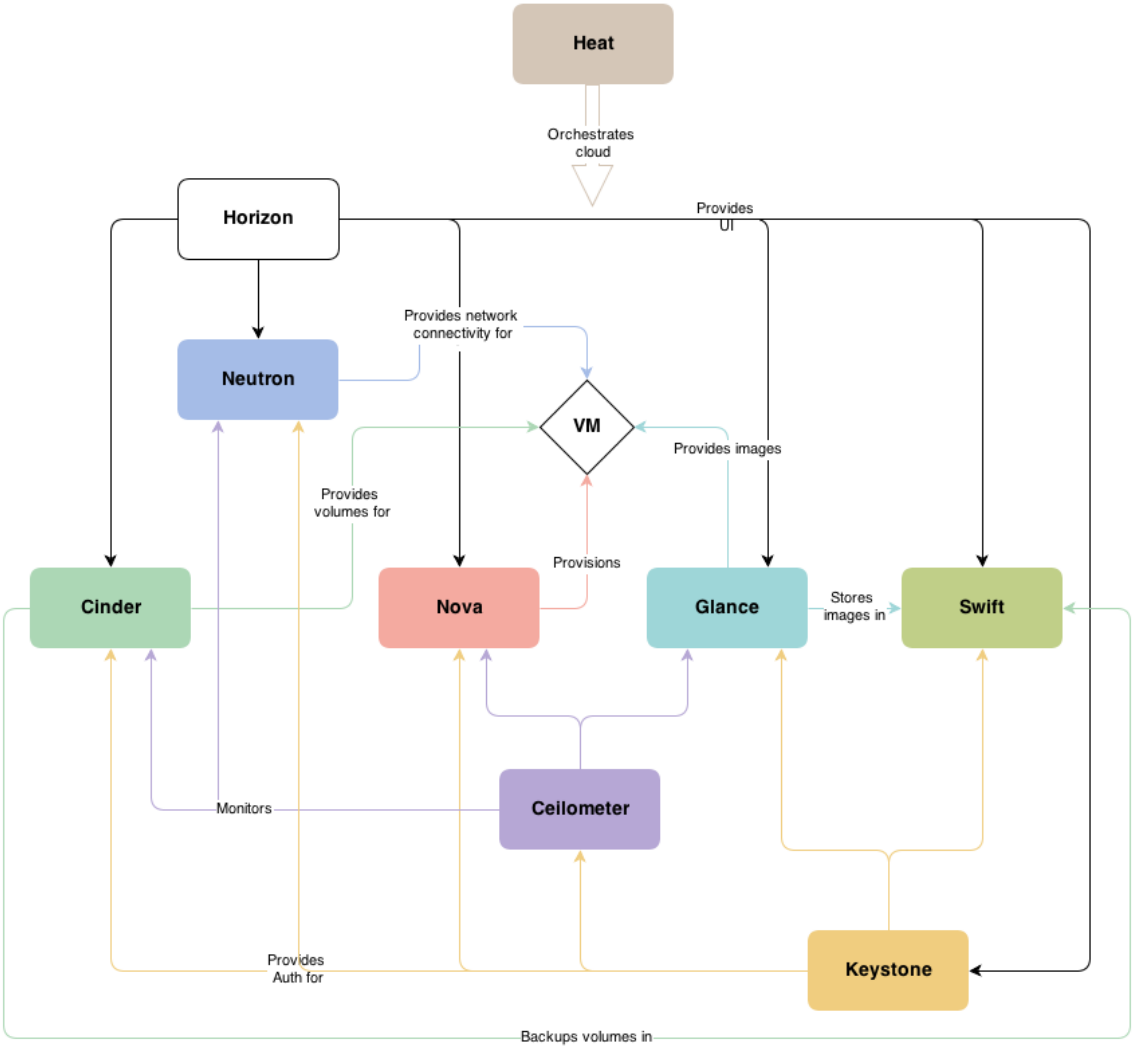
サービス	プロジェクト名	説明
Identity service	Keystone	他の OpenStack サービスに対して認証および認可サービスを提供します。すべての OpenStack サービスに対してエンドポイントのカタログを提供します。
Image Service	Glance	仮想マシンディスクイメージを保存および取得します。OpenStack Compute がインスタンスの配備中に使用します。
Telemetry	Ceilometer	課金、ベンチマーク、スケーラビリティ、統計などの目的のために、OpenStack クラウドを監視および測定します。
高レベルサービス		
Orchestration	Heat	Orchestrates multiple composite cloud applications by using either the native HOT template format or the AWS CloudFormation template format, through both an OpenStack-native REST API and a CloudFormation-compatible Query API.
Database Service	Trove	Provides scalable and reliable Cloud Database-as-a-Service functionality for both relational and non-relational database engines.

このガイドはこれらのサービスを機能テスト環境に導入する方法について説明します。例えば、本番環境を構築する方法を教えます。

概念アーキテクチャー

仮想マシンやインスタンスの起動には、いくつかのサービスがいくつも通信します。以下の図は一般的な OpenStack 環境の概念アーキテクチャーです。

図1.1 概念アーキテクチャー



サンプルアーキテクチャー

OpenStack is highly configurable to meet different needs with various compute, networking, and storage options. This guide enables you to choose your own OpenStack adventure using a combination of basic and optional services. This guide uses the following example architectures:

- レガシーなネットワークを持つ 2 ノードアーキテクチャー。図1.2「レガシーなネットワークを持つ 2 ノードアーキテクチャー」 [5]を参照してください。

  - The basic controller node runs the Identity service, Image Service, management portion of Compute, and the dashboard necessary to launch a simple instance. It also includes supporting services such as MySQL, AMQP, and NTP.

Optionally, the controller node also runs portions of Block Storage, Object Storage, Database Service, Orchestration, and Telemetry. These components provide additional features for your environment.

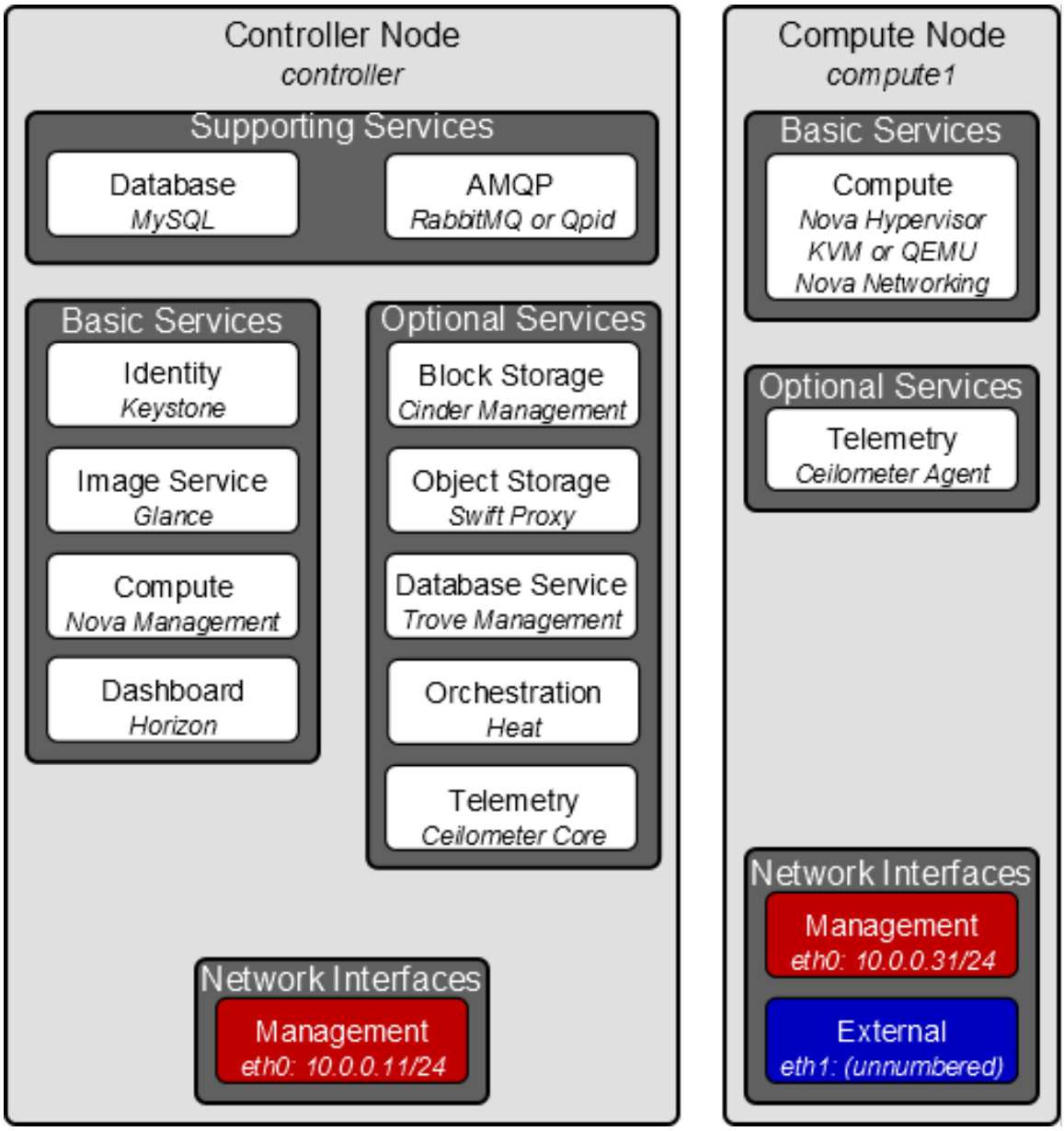
- The basic compute node runs the hypervisor portion of Compute, which operates tenant virtual machines or instances. By default, Compute uses KVM as the hypervisor. Compute also provisions and operates tenant networks and implements security groups. You can run more than one compute node.

  Optionally, the compute node also runs the Telemetry agent. This component provides additional features for your environment.

注記

When you implement this architecture, skip 「Networking (neutron)」 [49] in 7章Networking Service の追加 [49]. To use optional services, you might need to install additional nodes, as described in subsequent chapters.

図1.2 レガシーなネットワークを持つ 2 ノードアーキテクチャー



- Three-node architecture with OpenStack Networking (neutron). See 図 1.3「OpenStack Networking (Neutron) を持つ 3 ノードアーキテクチャー」 [6].

  - 基本的なコントローラーノードは、Identity Service、Image Service、および Compute、Networking、Networking プラグイン、ダッシュボードの管理部分を実行します。MySQL、AMQP、NTP のようなサポートサービスも含まれます。

    Optionally, the controller node also runs portions of Block Storage, Object Storage, Database Service, Orchestration, and Telemetry. These components provide additional features for your environment.

- The network node runs the Networking plug-in, layer 2 agent, and several layer 3 agents that provision and operate tenant networks. Layer 2 services include provisioning of virtual networks and tunnels. Layer 3 services include routing, NAT , and DHCP. This node also handles external (internet) connectivity for tenant virtual machines or instances.

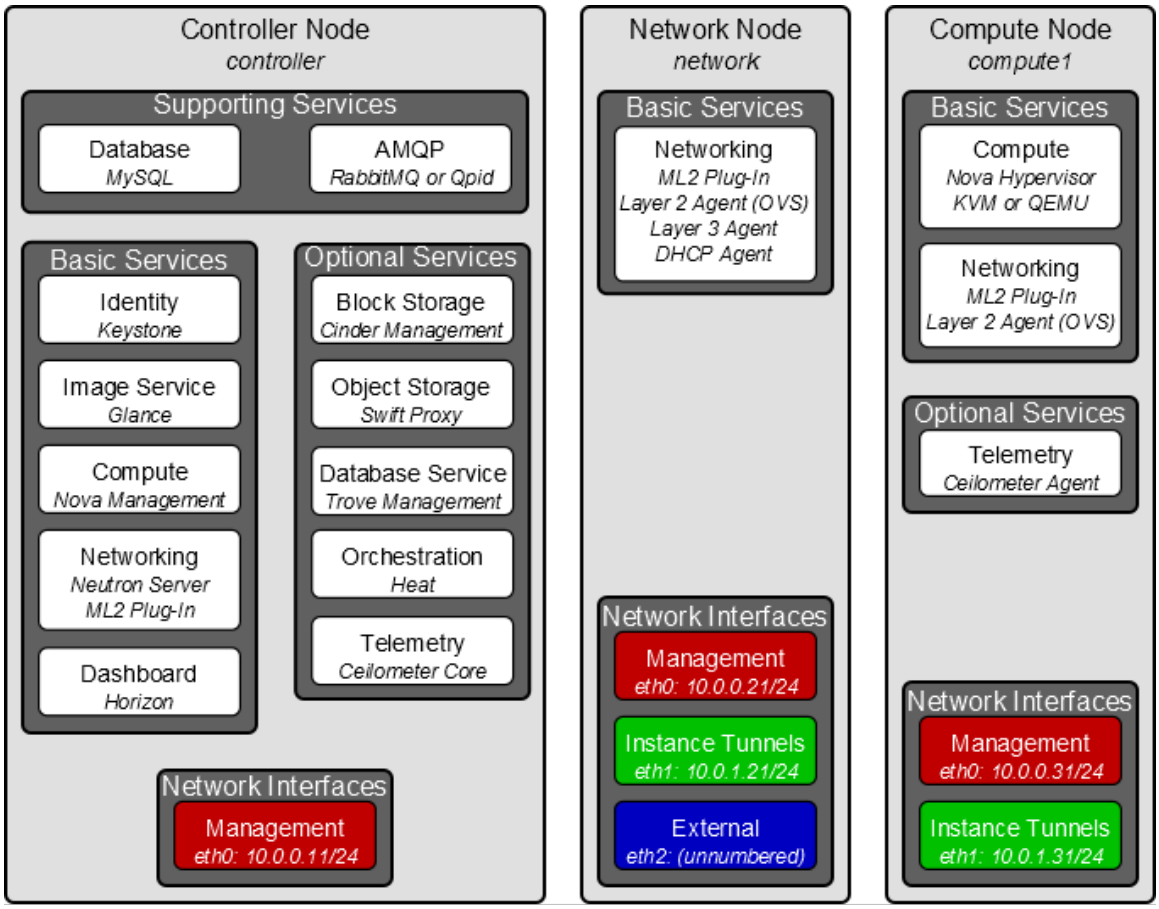
- The compute node runs the hypervisor portion of Compute, which operates tenant virtual machines or instances. By default Compute uses KVM as the hypervisor. The compute node also runs the Networking plug-in and layer 2 agent which operate tenant networks and implement security groups. You can run more than one compute node.

  Optionally, the compute node also runs the Telemetry agent. This component provides additional features for your environment.

注記

When you implement this architecture, skip 「Legacy networking」 [81] in 7章Networking Service の追加 [49]. To use optional services, you might need to install additional nodes, as described in subsequent chapters.

図1.3 OpenStack Networking (Neutron) を持つ 3 ノードアーキテクチャー



第2章 環境の基本設定

目次

警告

Icehouse 向けにこのドキュメントを更新中です。この作業中、構造や内容の問題が見つかるかもしれません。

This chapter explains how to configure each node in the example architectures including the two-node architecture with legacy networking and three-node architecture with OpenStack Networking (neutron).

注記

Although most environments include OpenStack Identity, Image Service, Compute, at least one networking service, and the dashboard, OpenStack Object Storage can operate independently of most other services. If your use case only involves Object Storage, you can skip to 「System requirements for Object Storage」 [98]. However, the dashboard will not work without at least OpenStack Image Service and Compute.

注記

You must use an account with administrative privileges to configure each node. Either run the commands as the root user or configure the sudo utility.

始める前に

For a functional environment, OpenStack doesn't require a significant amount of resources. We recommend that your environment meets or exceeds the following minimum requirements which can support several minimal CirrOS instances:

- コントローラーノード: 1 CPU、2 GB メモリ、5 GB ストレージ
- ネットワークノード: 1 CPU、512 MB メモリ、5 GB ストレージ
- コンピュートノード: 1 CPU、2 GB メモリ、10 GB ストレージ

To minimize clutter and provide more resources for OpenStack, we recommend a minimal installation of your Linux distribution. Also, we strongly recommend that you install a 64-bit version of your distribution on at least the compute node. If you install a 32-bit version of your distribution on the compute node, attempting to start an instance using a 64-bit image will fail.

注記

A single disk partition on each node works for most basic installations. However, you should consider Logical Volume Manager (LVM) for installations with optional services such as Block Storage.

Many users build their test environments on virtual machines (VMs). The primary benefits of VMs include the following:

- One physical server can support multiple nodes, each with almost any number of network interfaces.
- Ability to take periodic "snap shots" throughout the installation process and "roll back" to a working configuration in the event of a problem.

However, VMs will reduce performance of your instances, particularly if your hypervisor and/or processor lacks support for hardware acceleration of nested VMs.

注記

If you choose to install on VMs, make sure your hypervisor permits promiscuous mode on the external network.

システム要件の詳細は OpenStack 運用ガイドを参照してください。

ネットワーク

導入するアーキテクチャーに合わせて、各ノードにオペレーティングシステムをインストールした後、ネットワークインターフェースを設定する必要があります。すべての自動ネットワーク管理ツールを無効化し、お使いのディストリビューションに合わせて適切な設定ファイルを手動で編集することを推奨します。お使いのディストリビューションでネットワークを設定する方法に関する詳細は、ドキュメントを参照してください。

NetworkManager を無効化し、network サービスを有効化する方法:

- ```
  # service NetworkManager stop
  # service network start
  # chkconfig NetworkManager off
  # chkconfig network on
  ```

RHEL および、CentOS や Scientific Linux のような派生物は、標準で制限的なファイアウォールを有効化しています。このインストール中に、この設定を変更するか、ファイアウォールを無効化しなければ、特定の手順が失敗します。インストール環境をセキュア化することに関する詳細は OpenStack セキュリティガイド を参照してください。
  ```

Fedora の場合、firewalld が標準のファイアウォールシステムとして iptables の代わりに使用されています。firewalld を正常に使用できている限り、このガイドは他のディストリビューションとの互換性のために iptables を参照しています。

firewalld を無効化し、iptables を有効化する方法:

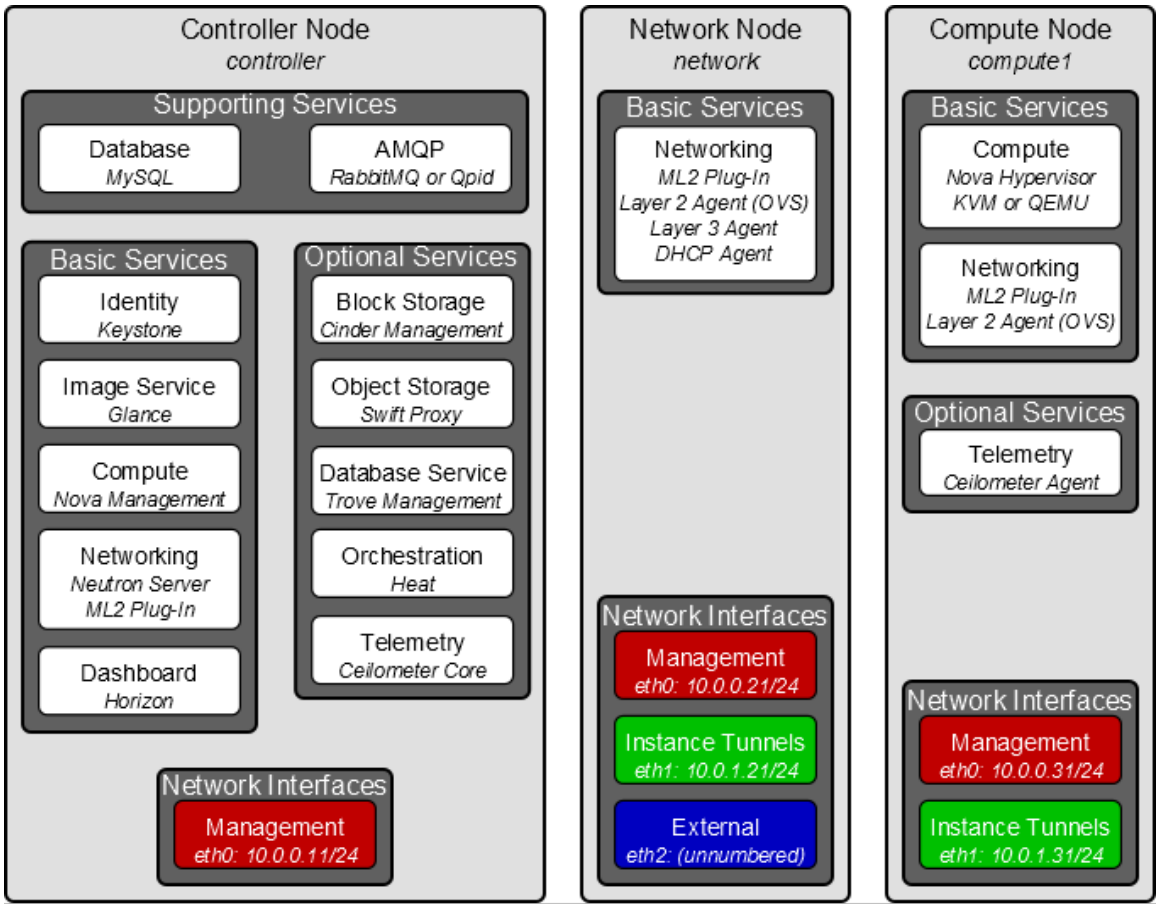
- ```
  # service firewalld stop
  # service iptables start
  # chkconfig firewalld off
  # chkconfig iptables on
  ```

OpenStack Networking や レガシーネットワークアーキテクチャーの例のネットワーク設定に進みます。

OpenStack Networking

The example architecture with OpenStack Networking (neutron) requires one controller node, one network node, and at least one compute node. The controller node contains one network interface on the management network. The network node contains one network interface on the management network, one on the instance tunnels network, and one on the external network. The compute node contains one network interface on the management network and one on the instance tunnels network.

図2.1 Three-node architecture with OpenStack Networking


  ```

Unless you intend to use the exact configuration provided in this example architecture, you must modify the networks in this procedure to match your environment. Also, each node must resolve the other nodes by name in addition to IP address. For example, the controller name must resolve to 10.0.0.11, the IP address of the management interface on the controller node.

警告

Reconfiguring network interfaces will interrupt network connectivity. We recommend using a local terminal session for these procedures.

コントローラーノード

ネットワークを設定する方法:

- 管理インターフェースを設定します。

  IP アドレス: 10.0.0.11

  ネットマスク: 255.255.255.0 (または /24)

  デフォルトゲートウェイ: 10.0.0.1

名前解決を設定する方法:

- /etc/hosts ファイルを編集し、以下の内容を含めます。

```
# controller
10.0.0.11       controller

# network
10.0.0.21       network

# compute1
10.0.0.31       compute1
```

ネットワークノード

ネットワークを設定する方法:

1. 管理インターフェースを設定します。

   IP アドレス: 10.0.0.21

   ネットマスク: 255.255.255.0 (または /24)

   デフォルトゲートウェイ: 10.0.0.1

2. Configure the instance tunnels interface:

   IP アドレス: 10.0.1.21

   ネットマスク: 255.255.255.0 (または /24)

3. 外部インターフェースは、IP アドレスを割り当てない特別な設定を使用します。外部インターフェースを設定します。

- /etc/sysconfig/network-scripts/ifcfg-eth2 ファイルを編集し、以下の内容を含めます。

  HWADDR と UUID の項目を変更してはいけません。

```
DEVICE=eth2
TYPE=Ethernet
ONBOOT="yes"
BOOTPROTO="none"
```

4. ネットワークを再起動します。

```
# service network restart
```

名前解決を設定する方法:

- /etc/hosts ファイルを編集し、以下の内容を含めます。

```
# network
10.0.0.21       network

# controller
10.0.0.11       controller

# compute1
10.0.0.31       compute1
```

コンピュートノード

ネットワークを設定する方法:

1. 管理インターフェースを設定します。

   IP アドレス: 10.0.0.31

   ネットマスク: 255.255.255.0 (または /24)

   デフォルトゲートウェイ: 10.0.0.1

注記

追加のコンピュートノードは 10.0.0.32、10.0.0.33 などを使用すべきです。

2. Configure the instance tunnels interface:

   IP アドレス: 10.0.1.31

   ネットマスク: 255.255.255.0 (または /24)

注記

追加のコンピュートノードは 10.0.1.32、10.0.1.33 などを使用すべきです。

名前解決を設定する方法:

- /etc/hosts ファイルを編集し、以下の内容を含めます。

```
# compute1
10.0.0.31       compute1

# controller
10.0.0.11       controller

# network
10.0.0.21       network
```

接続性の検証

We recommend that you verify network connectivity to the internet and among the nodes before proceeding further.

1. From the controller node, ping a site on the internet:

```
# ping -c 4 openstack.org
PING openstack.org (174.143.194.225) 56(84) bytes of data.
64 bytes from 174.143.194.225: icmp_seq=1 ttl=54 time=18.3 ms
64 bytes from 174.143.194.225: icmp_seq=2 ttl=54 time=17.5 ms
64 bytes from 174.143.194.225: icmp_seq=3 ttl=54 time=17.5 ms
64 bytes from 174.143.194.225: icmp_seq=4 ttl=54 time=17.4 ms

--- openstack.org ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3022ms
rtt min/avg/max/mdev = 17.489/17.715/18.346/0.364 ms
```

2. From the controller node, ping the management interface on the network node:

```
# ping -c 4 Network
PING network (10.0.0.21) 56(84) bytes of data.
64 bytes from network (10.0.0.21): icmp_seq=1 ttl=64 time=0.263 ms
64 bytes from network (10.0.0.21): icmp_seq=2 ttl=64 time=0.202 ms
64 bytes from network (10.0.0.21): icmp_seq=3 ttl=64 time=0.203 ms
64 bytes from network (10.0.0.21): icmp_seq=4 ttl=64 time=0.202 ms

--- network ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3000ms
rtt min/avg/max/mdev = 0.202/0.217/0.263/0.030 ms
```

3. From the controller node, ping the management interface on the compute node:

```
# ping -c 4 compute1
PING compute1 (10.0.0.31) 56(84) bytes of data.
64 bytes from compute1 (10.0.0.31): icmp_seq=1 ttl=64 time=0.263 ms
64 bytes from compute1 (10.0.0.31): icmp_seq=2 ttl=64 time=0.202 ms
64 bytes from compute1 (10.0.0.31): icmp_seq=3 ttl=64 time=0.203 ms
64 bytes from compute1 (10.0.0.31): icmp_seq=4 ttl=64 time=0.202 ms

--- network ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3000ms
rtt min/avg/max/mdev = 0.202/0.217/0.263/0.030 ms
```

4. From the network node, ping a site on the internet:

```
# ping -c 4 openstack.org
PING openstack.org (174.143.194.225) 56(84) bytes of data.
64 bytes from 174.143.194.225: icmp_seq=1 ttl=54 time=18.3 ms
64 bytes from 174.143.194.225: icmp_seq=2 ttl=54 time=17.5 ms
64 bytes from 174.143.194.225: icmp_seq=3 ttl=54 time=17.5 ms
64 bytes from 174.143.194.225: icmp_seq=4 ttl=54 time=17.4 ms

--- openstack.org ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3022ms
rtt min/avg/max/mdev = 17.489/17.715/18.346/0.364 ms
```

5. From the network node, ping the management interface on the controller node:

```
# ping -c 4 controller
PING controller (10.0.0.11) 56(84) bytes of data.
64 bytes from controller (10.0.0.11): icmp_seq=1 ttl=64 time=0.263 ms
64 bytes from controller (10.0.0.11): icmp_seq=2 ttl=64 time=0.202 ms
64 bytes from controller (10.0.0.11): icmp_seq=3 ttl=64 time=0.203 ms
64 bytes from controller (10.0.0.11): icmp_seq=4 ttl=64 time=0.202 ms

--- controller ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3000ms
rtt min/avg/max/mdev = 0.202/0.217/0.263/0.030 ms
```

6. From the network node, ping the instance tunnels interface on the compute node:

```
# ping -c 4 10.0.1.31
PING 10.0.1.31 (10.0.1.31) 56(84) bytes of data.
64 bytes from 10.0.1.31 (10.0.1.31): icmp_seq=1 ttl=64 time=0.263 ms
64 bytes from 10.0.1.31 (10.0.1.31): icmp_seq=2 ttl=64 time=0.202 ms
64 bytes from 10.0.1.31 (10.0.1.31): icmp_seq=3 ttl=64 time=0.203 ms
64 bytes from 10.0.1.31 (10.0.1.31): icmp_seq=4 ttl=64 time=0.202 ms

--- 10.0.1.31 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3000ms
rtt min/avg/max/mdev = 0.202/0.217/0.263/0.030 ms
```

7. From the compute node, ping a site on the internet:

```
# ping -c 4 openstack.org
PING openstack.org (174.143.194.225) 56(84) bytes of data.
64 bytes from 174.143.194.225: icmp_seq=1 ttl=54 time=18.3 ms
64 bytes from 174.143.194.225: icmp_seq=2 ttl=54 time=17.5 ms
64 bytes from 174.143.194.225: icmp_seq=3 ttl=54 time=17.5 ms
64 bytes from 174.143.194.225: icmp_seq=4 ttl=54 time=17.4 ms

--- openstack.org ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3022ms
rtt min/avg/max/mdev = 17.489/17.715/18.346/0.364 ms
```

8. From the compute node, ping the management interface on the controller node:

```
# ping -c 4 controller
PING controller (10.0.0.11) 56(84) bytes of data.
64 bytes from controller (10.0.0.11): icmp_seq=1 ttl=64 time=0.263 ms
64 bytes from controller (10.0.0.11): icmp_seq=2 ttl=64 time=0.202 ms
64 bytes from controller (10.0.0.11): icmp_seq=3 ttl=64 time=0.203 ms
64 bytes from controller (10.0.0.11): icmp_seq=4 ttl=64 time=0.202 ms
```

```
--- controller ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3000ms
rtt min/avg/max/mdev = 0.202/0.217/0.263/0.030 ms
```

9. From the compute node, ping the instance tunnels interface on the network node:

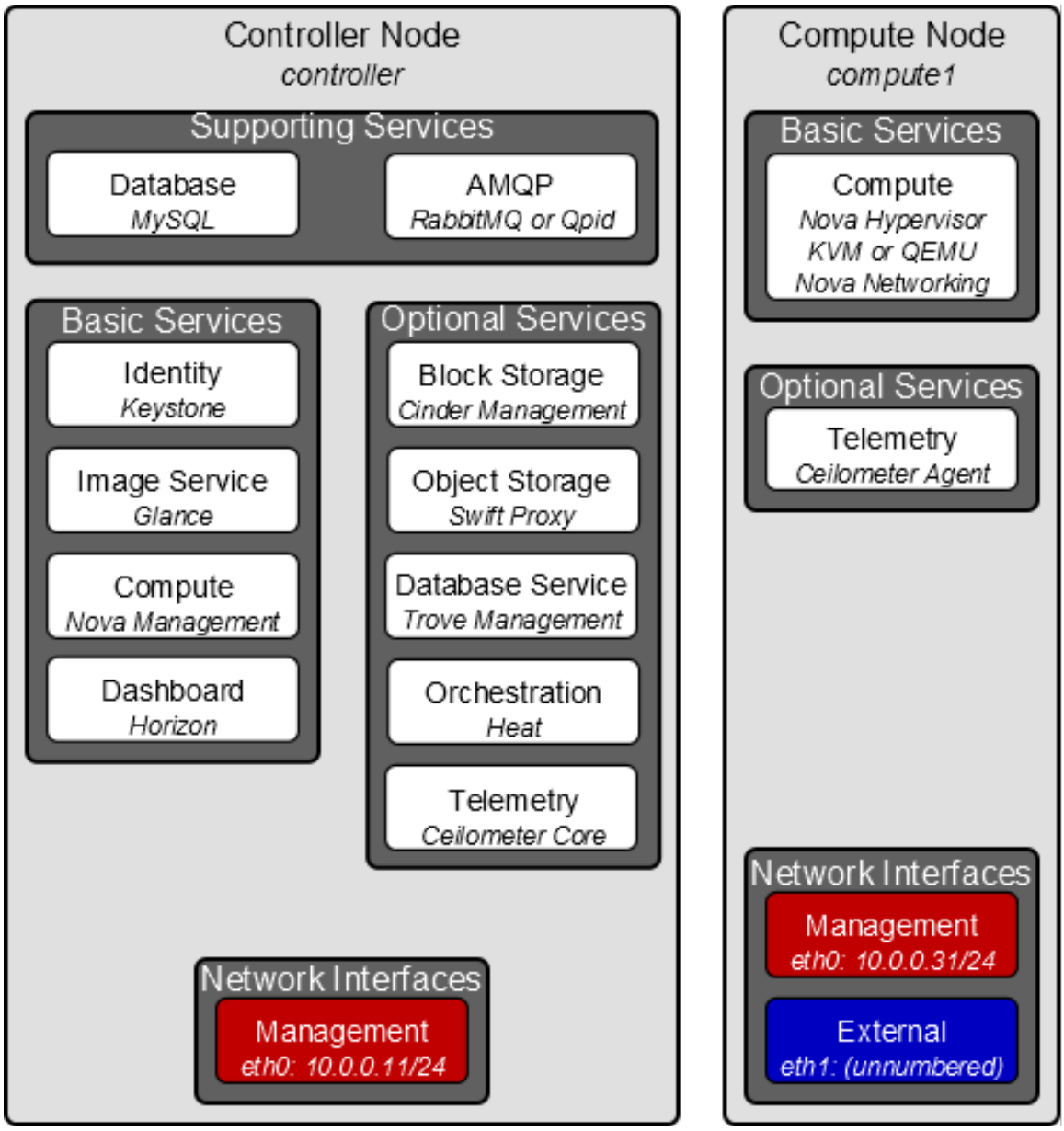
```
# ping -c 4 10.0.1.21
PING 10.0.1.21 (10.0.1.21) 56(84) bytes of data.
64 bytes from 10.0.1.21 (10.0.1.21): icmp_seq=1 ttl=64 time=0.263 ms
64 bytes from 10.0.1.21 (10.0.1.21): icmp_seq=2 ttl=64 time=0.202 ms
64 bytes from 10.0.1.21 (10.0.1.21): icmp_seq=3 ttl=64 time=0.203 ms
64 bytes from 10.0.1.21 (10.0.1.21): icmp_seq=4 ttl=64 time=0.202 ms

--- 10.0.1.21 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3000ms
rtt min/avg/max/mdev = 0.202/0.217/0.263/0.030 ms
```

Legacy networking

The example architecture with legacy networking (nova) requires one controller node and at least one compute node. The controller node contains one network interface on the management network. The compute node contains one network interface on the management network and one on the external network.

図2.2 レガシーなネットワークを持つ 2 ノードアーキテクチャー



Unless you intend to use the exact configuration provided in this example architecture, you must modify the networks in this procedure to match your environment. Also, each node must resolve the other nodes by name in addition to IP address. For example, the controller name must resolve to 10.0.0.11, the IP address of the management interface on the controller node.

警告

Reconfiguring network interfaces will interrupt network connectivity. We recommend using a local terminal session for these procedures.

コントローラーノード

ネットワークを設定する方法:

- 管理インターフェースを設定します。

  IP アドレス: 10.0.0.11

  ネットマスク: 255.255.255.0 (または /24)

  デフォルトゲートウェイ: 10.0.0.1

名前解決を設定する方法:

- /etc/hosts ファイルを編集し、以下の内容を含めます。

```
# controller
10.0.0.11       controller

# compute1
10.0.0.31       compute1
```

コンピュートノード

ネットワークを設定する方法:

1. 管理インターフェースを設定します。

   IP アドレス: 10.0.0.31

   ネットマスク: 255.255.255.0 (または /24)

   デフォルトゲートウェイ: 10.0.0.1

   ### 注記

   追加のコンピュートノードは 10.0.0.32、10.0.0.33 などを使用すべきです。

2. 外部インターフェースは、IP アドレスを割り当てない特別な設定を使用します。外部インターフェースを設定します。

   - Edit the /etc/sysconfig/network-scripts/ifcfg-eth1 file to contain the following:

     HWADDR と UUID の項目を変更してはいけません。

     ```
     DEVICE=eth1
     TYPE=Ethernet
     ONBOOT="yes"
     BOOTPROTO="none"
     ```

3. ネットワークを再起動します。

   ```
   # service network restart
   ```

名前解決を設定する方法:

- /etc/hosts ファイルを編集し、以下の内容を含めます。

```
# compute1
10.0.0.31       compute1

# controller
10.0.0.11       controller
```

接続性の検証

We recommend that you verify network connectivity to the internet and among the nodes before proceeding further.

1. From the controller node, ping a site on the internet:

```
# ping -c 4 openstack.org
PING openstack.org (174.143.194.225) 56(84) bytes of data.
64 bytes from 174.143.194.225: icmp_seq=1 ttl=54 time=18.3 ms
64 bytes from 174.143.194.225: icmp_seq=2 ttl=54 time=17.5 ms
64 bytes from 174.143.194.225: icmp_seq=3 ttl=54 time=17.5 ms
64 bytes from 174.143.194.225: icmp_seq=4 ttl=54 time=17.4 ms

--- openstack.org ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3022ms
rtt min/avg/max/mdev = 17.489/17.715/18.346/0.364 ms
```

2. From the controller node, ping the management interface on the compute node:

```
# ping -c 4 compute1
PING compute1 (10.0.0.31) 56(84) bytes of data.
64 bytes from compute1 (10.0.0.31): icmp_seq=1 ttl=64 time=0.263 ms
64 bytes from compute1 (10.0.0.31): icmp_seq=2 ttl=64 time=0.202 ms
64 bytes from compute1 (10.0.0.31): icmp_seq=3 ttl=64 time=0.203 ms
64 bytes from compute1 (10.0.0.31): icmp_seq=4 ttl=64 time=0.202 ms

--- compute1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3000ms
rtt min/avg/max/mdev = 0.202/0.217/0.263/0.030 ms
```

3. From the compute node, ping a site on the internet:

```
# ping -c 4 openstack.org
PING openstack.org (174.143.194.225) 56(84) bytes of data.
64 bytes from 174.143.194.225: icmp_seq=1 ttl=54 time=18.3 ms
64 bytes from 174.143.194.225: icmp_seq=2 ttl=54 time=17.5 ms
64 bytes from 174.143.194.225: icmp_seq=3 ttl=54 time=17.5 ms
64 bytes from 174.143.194.225: icmp_seq=4 ttl=54 time=17.4 ms

--- openstack.org ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3022ms
rtt min/avg/max/mdev = 17.489/17.715/18.346/0.364 ms
```

4. From the compute node, ping the management interface on the controller node:

```
# ping -c 4 controller
PING controller (10.0.0.11) 56(84) bytes of data.
64 bytes from controller (10.0.0.11): icmp_seq=1 ttl=64 time=0.263 ms
```

```
64 bytes from controller (10.0.0.11): icmp_seq=2 ttl=64 time=0.202 ms
64 bytes from controller (10.0.0.11): icmp_seq=3 ttl=64 time=0.203 ms
64 bytes from controller (10.0.0.11): icmp_seq=4 ttl=64 time=0.202 ms

--- controller ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3000ms
rtt min/avg/max/mdev = 0.202/0.217/0.263/0.030 ms
```

Network Time Protocol (NTP)

複数のマシンにわたりサービスを同期するために、NTP をインストールする必要があります。このガイドの例は、コントローラーノードを参照サーバーとして設定し、他のすべてのノードはコントローラーノードから時刻を設定するよう設定します。

Install the ntp package on each system running OpenStack services:

```
# yum install ntp
```

Set up the NTP server on your controller node so that it receives data by modifying the ntp.conf file and restarting the service:

```
# service ntpd start
# chkconfig ntpd on
```

It is advised that you configure additional nodes to synchronize their time from the controller node rather than from outside of your LAN. To do so, install the ntp daemon as above, then edit /etc/ntp.conf and change the server directive to use the controller node as internet time source.

パスワード

The various OpenStack services and the required software like the database and the messaging server have to be password protected. You use these passwords when configuring a service and then again to access the service. You have to choose a password while configuring the service and later remember to use the same password when accessing it. Optionally, you can generate random passwords with the pwgen program. Or, to create passwords one at a time, use the output of this command repeatedly:

```
$ openssl rand -hex 10
```

This guide uses the convention that SERVICE_PASS is the password to access the service SERVICE and SERVICE_DBPASS is the database password used by the service SERVICE to access the database.

The complete list of passwords you need to define in this guide are:

表2.1 Passwords

Password name	Description
Database password (no variable used)	Root password for the database
KEYSTONE_DBPASS	Database password of Identity service
DEMO_PASS	Password of user demo
ADMIN_PASS	Password of user admin

Password name	Description
GLANCE_DBPASS	Database password for Image Service
GLANCE_PASS	Password of Image Service user glance
NOVA_DBPASS	Database password for Compute service
NOVA_PASS	Password of Compute service user nova
DASH_DBPASS	Database password for the dashboard
CINDER_DBPASS	Database password for the Block Storage service
CINDER_PASS	Password of Block Storage service user cinder
NEUTRON_DBPASS	Database password for the Networking service
NEUTRON_PASS	Password of Networking service user neutron
HEAT_DBPASS	Database password for the Orchestration service
HEAT_PASS	Password of Orchestration service user heat
CEILOMETER_DBPASS	Database password for the Telemetry service
CEILOMETER_PASS	Password of Telemetry service user ceilometer
TROVE_DBPASS	Database password of Database service
TROVE_PASS	Password of Database Service user trove

データベース

Most OpenStack services require a database to store information. These examples use a MySQL database that runs on the controller node. You must install the MySQL database on the controller node. You must install the MySQL Python library on any additional nodes that access MySQL.

コントローラーのセットアップ

コントローラーノードに MySQL クライアントとサーバーパッケージ、Python ライブラリをインストールします。

```
# yum install mysql mysql-server MySQL-python
```

OpenStack を扱うために、いくつかの MySQL の設定変更が必要になります。

- /etc/my.cnf ファイルを編集します。

  a. 管理ネットワーク経由で他のノードからアクセスできるようにするために、[mysqld] セクションの下で、bind-address キーにコントローラーノードの管理 IP アドレスを設定します。

  ```
  [mysqld]
  ...
  bind-address = 10.0.0.11
  ```

  b. Under the [mysqld] section, set the following keys to enable InnoDB, UTF-8 character set, and UTF-8 collation by default:

  ```
  [mysqld]
  ...
  default-storage-engine = innodb
  collation-server = utf8_general_ci
  init-connect = 'SET NAMES utf8'
  character-set-server = utf8
  ```

Start the MySQL database server and set it to start automatically when the system boots:

```
# service mysqld start
# chkconfig mysqld on
```

最後に MySQL データベースの root パスワードを設定すべきです。データベースとテーブルをセットアップする OpenStack のプログラムは、パスワードが設定されているとプロンプトを表示します。

You must delete the anonymous users that are created when the database is first started. Otherwise, database connection problems occur when you follow the instructions in this guide. To do this, use the mysql_secure_installation command. Note that if mysql_secure_installation fails you might need to use mysql_install_db first:

```
# mysql_install_db
# mysql_secure_installation
```

If you have not already set a root database password, press ENTER when you are prompted for the password. This command presents a number of options for you to secure your database installation. Respond yes to all prompts unless you have a good reason to do otherwise.

ノードのセットアップ

On all nodes other than the controller node, install the MySQL Python library:

```
# yum install MySQL-python
```

OpenStack パッケージ

ディストリビューションはその一部として OpenStack パッケージをリリースしているかもしれません。または、OpenStack とディストリビューションのリリース間隔がお互いに独立しているため、他の方法によりリリースしているかもしれません。

このセクションは、最新の OpenStack パッケージをインストールするために、マシンを設定した後に完了する必要がある設定について説明します。

The examples in this guide use the OpenStack packages from the RDO repository. These packages work on Red Hat Enterprise Linux 6, compatible versions of CentOS, and Fedora 20. To enable the RDO repository, download and install the rdo-release-icehouse package:

```
# yum install http://repos.fedorapeople.org/repos/openstack/openstack-icehouse/rdo-release-icehouse-2.noarch.rpm
```

EPEL パッケージはパッケージ署名とリポジトリ情報のために GPG キーを含みます。これは Red Hat Enterprise Linux と CentOS のみにインストールすべきです。Fedora では必要がありません。最新の epel-release パッケージ（http://download.fedoraproject.org/pub/epel/6/x86_64/repoview/epel-release.html 参照）をインストールします。例:

```
# yum install http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
```

The openstack-utils package contains utility programs that make installation and configuration easier. These programs are used throughout this guide. Install openstack-utils. This verifies that you can access the RDO repository:

```
# yum install openstack-utils
```

警告

openstack-utils パッケージにある openstack-config プログラムは設定ファイルを操作するために crudini を使用します。しかしながら、crudini バージョン 3.0 は複数の値を持つオプションをサポートしません。https://bugs.launchpad.net/openstack-manuals/+bug/1269271 を参照してください。回避策として、複数の値を持つオプションを手動で設定する必要があります。または、新しいオプションを作成する代わりに、新しい値で前の値を上書きします。

The openstack-selinux package includes the policy files that are required to configure SELinux during OpenStack installation on RHEL and CentOS. This step is not required during OpenStack installation on Fedora. Install openstack-selinux:

```
# yum install openstack-selinux
```

お使いのシステムのパッケージをアップグレードします。

```
# yum upgrade
```

アップグレードが新しいカーネルパッケージを含む場合、確実に新しいカーネルを実行するために、システムを再起動します。

```
# reboot
```

メッセージングサーバー

On the controller node, install the messaging queue server. Typically this is Qpid but RabbitMQ and ZeroMQ (0MQ) are also available:

```
# yum install qpid-cpp-server
```

Disable Qpid authentication by editing /etc/qpidd.conf file and changing the auth option to no:

```
auth=no
```

注記

設定を簡単にするために、このガイドの Qpid 例は認証を使用しません。しかしながら、本番環境では認証を有効化することを強く推奨します。Qpid のセキュア化に関する詳細は Qpid のドキュメントを参照してください。

Qpid 認証を有効化した後、qpid_username と qpid_password 設定キーがそれぞれ有効な Qpid のユーザー名とパスワードを確実に参照するよう、各 OpenStack サービスの設定ファイルを更新する必要があります。

Start Qpid and set it to start automatically when the system boots:

```
# service qpidd start
# chkconfig qpidd on
```

おめでとうございます。これで OpenStack サービスをインストールする準備ができました。

第3章 Identity Service の設定

目次

Identity Service の概念

Identity Service は以下の機能を実行します。

- ユーザー管理。ユーザーとその権限を追跡します。
- サービスカタログ。利用可能なサービスのカタログとその API エンドポイントを提供します。

Identity Service を理解するために、以下の概念を理解する必要があります。

ユーザー

人、システム、または OpenStack クラウドサービスを使用するサービスの電子的な表現。Identity Service は遅れられてきたリクエストがどのユーザーにより行われているかを検証します。ユーザーはログインでき、リソースにアクセスするためにトークンを割り当てられるかもしれません。ユーザーは特定のテナントに直接割り当てられ、そのテナントに含まれているかのように振る舞います。

クレデンシャル

ユーザーが誰であるかを証明するために、ユーザーのみにより知られているデータ。Identity Service では、次のようなものがあります。ユーザー名とパスワード、ユーザー名と API キー、Identity Service により発行された認証トークン。

認証

ユーザーの同一性を確認する動作。Identity Service は、ユーザーに提供された一組のクレデンシャルを検証することにより、送られてきたリクエストを確認します。

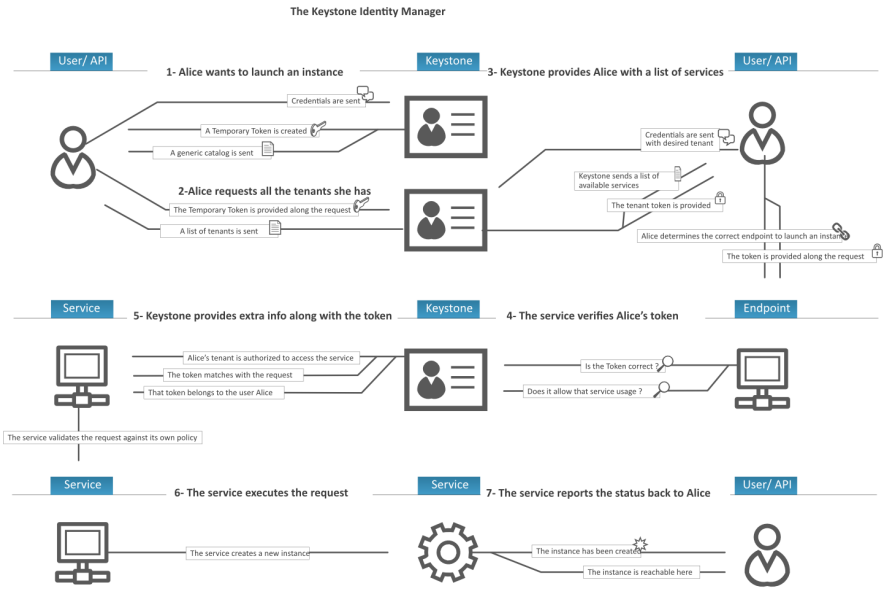
これらのクレデンシャルは最初にユーザー名とパスワード、またはユーザー名と API トークンです。これらのクレデンシャルの応答で、Identity Service がユーザーに認証トークンを発行します。ユーザーはこれ以降のリクエストでこのトークンを提供します。

トークン

リソースにアクセスするために使用される任意のビット数のテキスト。各トークンはアクセス可能なリソースを記述した範囲を持ちます。トークンは適宜失効しているかもしれません。また、有限の期間だけ有効です。

	Identity Service はこのリリースでトークンによる認証をサポートしますが、その意図は将来的にさらなるプロトコルをサポートすることです。意図は真っ先に統合サービスになるためですが、十分に成熟した認証ストアや管理ソリューションにある熱意はありません。
テナント	リソース、主体オブジェクト、またはその組み合わせをグループ化、または分離するために使用されるコンテナー。サービス操作者に依存して、テナントが顧客、アカウント、組織、プロジェクトに対応付けられるかもしれません。
サービス	Compute (Nova)、Object Storage (Swift)、Image Service (Glance) のような OpenStack サービス。ユーザーがリソースにアクセスでき、操作を実行できる 1 つ以上のエンドポイントを提供します。
エンドポイント	サービスにアクセスするところからネットワークアクセス可能なアドレス。通常は URL により記載されます。テンプレート用の拡張を使用している場合、エンドポイントのテンプレートを作成できます。これはリージョンを越えて利用できる、すべての消費できるサービスのテンプレートです。
役割	ユーザーが特定の操作の組を実行できると仮定する人格。ロールは一組の権利と権限を含みます。そのロールを仮定しているユーザーは、それらの権利と権限を継承します。 Identity Service では、ユーザーに発行されたトークンはユーザーが持つロールの一覧を含みます。そのユーザーにより呼び出されたサービスは、ユーザーが持つロール一覧を解釈する方法と、各ロールがアクセス権を持つ操作やリソースを判断します。

以下の図は Identity Service のプロセスフローを示します。



Identity Service のインストール

1. OpenStack Identity Service と python-keystoneclient（依存関係）をコントローラーノードにインストールします。

```
# yum install openstack-keystone python-keystoneclient
```

2. Identity Service は情報を保存するためにデータベースを使用します。設定ファイルでデータベースの場所を指定します。このガイドでは、コントローラーノードにユーザー名 keystone で MySQL データベースを使用します。KEYSTONE_DBPASS をデータベースのユーザーの適切なパスワードで置き換えます。

```
# openstack-config --set /etc/keystone/keystone.conf ¥
   database connection mysql://keystone:KEYSTONE_DBPASS@controller/keystone
```

3. root としてログインするために、前に設定したパスワードを使用します。keystone データベースユーザーを作成します。

```
$ mysql -u root -p
mysql> CREATE DATABASE keystone;
mysql> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost' ¥
  IDENTIFIED BY 'KEYSTONE_DBPASS';
mysql> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'%' ¥
  IDENTIFIED BY 'KEYSTONE_DBPASS';
mysql> exit
```

4. Image Service 用のデータベーステーブルを作成します。

```
# su -s /bin/sh -c "keystone-manage db_sync" keystone
```

5. Define an authorization token to use as a shared secret between the Identity Service and other OpenStack services. Use openssl to generate a random token and store it in the configuration file:

```
# ADMIN_TOKEN=$(openssl rand -hex 10)
# echo $ADMIN_TOKEN
# openstack-config --set /etc/keystone/keystone.conf DEFAULT ¥
   admin_token $ADMIN_TOKEN
```

6. By default, Keystone uses PKI tokens. Create the signing keys and certificates and restrict access to the generated data:

```
# keystone-manage pki_setup --keystone-user keystone --keystone-group keystone
# chown -R keystone:keystone /etc/keystone/ssl
# chmod -R o-rwx /etc/keystone/ssl
```

7. Identity Service を起動し、システム起動時に起動するよう有効化します。

```
# service openstack-keystone start
# chkconfig openstack-keystone on
```

8. Identity Service は標準で、期限切れトークンをデータベースに無期限に保存します。本番環境で監査のために有用であるかもしれませんが、期限切れトークンが蓄積すると、データベースの容量がかなり大きくなり、サービスの性能を劣化させるかもしれません。とくにリソースが限られたテスト環境で顕著かもしれません。期限切れトークンを 1 時間おきに削除するために、cron を使用して定期タスクを設定することを推奨します。

- Run the following command to purge expired tokens every hour and log the output to /var/log/keystone/keystone-tokenflush.log:

```
# (crontab -l 2>&1 | grep -q token_flush) || ¥
echo '@hourly /usr/bin/keystone-manage token_flush >/var/log/keystone/keystone-
tokenflush.log 2>&1' >> /var/spool/cron/root
```

ユーザー、プロジェクト、ロールの定義

Identity Service をインストールした後、認証するユーザー、プロジェクト、ロールをセットアップします。これらは次のセクションに記述されるサービスとエンドポイントへのアクセスを許可するために使用されます。

Typically, you would indicate a user and password to authenticate with the Identity Service. At this point, however, you have not created any users, so you have to use the authorization token created in an earlier step, see 「Identity Service のインストール」 [25] for further details. You can pass this with the --os-token option to the keystone command or set the OS_SERVICE_TOKEN environment variable. Set OS_SERVICE_TOKEN, as well as OS_SERVICE_ENDPOINT to specify where the Identity Service is running. Replace ADMIN_TOKEN with your authorization token.

```
$ export OS_SERVICE_TOKEN=ADMIN_TOKEN
$ export OS_SERVICE_ENDPOINT=http://controller:35357/v2.0
```

管理ユーザーの作成

管理ユーザー、ロール、プロジェクトを作成するために、以下の手順を実行します。OpenStack クラウドの管理操作のために、このアカウントを使用します。

Identity Service は標準で特別な _member_ ロールを作成します。OpenStack ダッシュボードは自動的にこのロールを持つユーザーにアクセス権を与えます。admin ユーザーのアクセス権に、このロールに加えて admin ロールを与えます。

注記

作成するすべてのロールは、各 OpenStack サービスに含まれる policy.json ファイルで指定されたロールにマップすべきです。多くのサービス用の標準ポリシーファイルは、管理アクセスを admin ロールに許可します。

1. admin ユーザーを作成します。

```
$ keystone user-create --name=admin --pass=ADMIN_PASS --email=ADMIN_EMAIL
```

ADMIN_PASS を安全なパスワードに置き換え、ADMIN_EMAIL をこのアカウントに関連付ける電子メールアドレスに置き換えます。

2. admin ロールを作成します。

```
$ keystone role-create --name=admin
```

3. admin プロジェクトを作成します。

```
$ keystone tenant-create --name=admin --description="Admin Tenant"
```

4. ここで user-role-add オプションを使用して、admin ユーザー、admin ロール、admin プロジェクトをリンクする必要があります。

```
$ keystone user-role-add --user=admin --tenant=admin --role=admin
```

5. admin ユーザー、_member_ ロール、admin プロジェクトをリンクします。

```
$ keystone user-role-add --user=admin --role=_member_ --tenant=admin
```

一般ユーザーの作成

一般ユーザーとプロジェクトを作成し、それらと特別な _member_ ロールをリンクするために、以下の手順を実行します。OpenStack クラウドの日々の非管理操作のために、このアカウントを使用します。別のユーザー名とパスワードを持つユーザーを作成するために、この手順を繰り返すことができます。これらのユーザーを作成するときに、プロジェクトを作成する手順を省略します。

1. demo ユーザーを作成します。

```
$ keystone user-create --name=demo --pass=DEMO_PASS --email=DEMO_EMAIL
```

DEMO_PASS を安全なパスワードに置き換え、DEMO_EMAIL をこのアカウントに関連付ける電子メールアドレスに置き換えます。

2. demo プロジェクトを作成します。

```
$ keystone tenant-create --name=demo --description="Demo Tenant"
```

注記

別のユーザーを追加するときに、この手順を繰り返さないでください。

3. demo ユーザー、_member_ ロール、demo プロジェクトをリンクします。

```
$ keystone user-role-add --user=demo --role=_member_ --tenant=demo
```

service プロジェクトの作成

OpenStack のサービスは、他の OpenStack のサービスにアクセスするために、ユーザー名、プロジェクト、ロールを必要とします。基本的なインストールでは、OpenStack のサービスは一般的に同じ service という名前のプロジェクトを共有します。

各サービスをインストールし、設定するので、このプロジェクトの下に追加のユーザー名とロールを作成します。

- service プロジェクトを作成します。

```
$ keystone tenant-create --name=service --description="Service Tenant"
```

サービスと API エンドポイントの定義

Identity Service が、どの OpenStack サービスがインストールされているか、それらがネットワークのどこにあるかを追跡できるよう、OpenStack インストール環境の各サービスを登録する必要があります。サービスを登録するために、これらのコマンドを実行します。

- keystone service-create. Describes the service.
- keystone endpoint-create. Associates API endpoints with the service.

Identity Service 自身も登録する必要があります。前に設定した OS_SERVICE_TOKEN 環境変数を認証のために使用します。

1. Identity Service のサービスエントリーを作成します。

```
$ keystone service-create --name=keystone --type=identity ¥
  --description="OpenStack Identity"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description | OpenStack Identity               |
| id          | 15c11a23667e427e91bc31335b45f4bd |
| name        | keystone                         |
| type        | identity                         |
+-------------+----------------------------------+
```

サービス ID はランダムに生成され、ここに表示されているものとは異なります。

2. 返されたサービス ID を使用することにより、Identity Service の API エンドポイントを指定します。エンドポイントを指定するとき、パブリック API、内部 API、管理 API の URL を指定します。このガイドでは、controller というホスト名を使用します。Identity Service は管理 API 用に異なるポートを使用することに注意してください。

```
$ keystone endpoint-create ¥
  --service-id=$(keystone service-list | awk '/ identity / {print $2}') ¥
  --publicurl=http://controller:5000/v2.0 ¥
  --internalurl=http://controller:5000/v2.0 ¥
  --adminurl=http://controller:35357/v2.0
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| adminurl    | http://controller:35357/v2.0     |
| id          | 11f9c625a3b94a3f8e66bf4e5de2679f |
| internalurl | http://controller:5000/v2.0      |
| publicurl   | http://controller:5000/v2.0      |
| region      | regionOne                        |
| service_id  | 15c11a23667e427e91bc31335b45f4bd |
+-------------+----------------------------------+
```

注記

お使いの OpenStack 環境に追加した各サービス用の追加のエンドポイントを作成することが必要になります。各サービスのインストールと関連したこのガイドのセクションに、サービスへのエンドポイントの具体的な作成手順があります。

Identity Service のインストールの検証

1. Identity Service が正しくインストールされ、設定されていることを確認するためには、OS_SERVICE_TOKEN 環境変数と OS_SERVICE_ENDPOINT 環境変数にある値を削除します。

```
$ unset OS_SERVICE_TOKEN OS_SERVICE_ENDPOINT
```

管理ユーザーをブートストラップし、Identity Service に登録するために使用された、これらの変数はもはや必要ありません。

2. これで通常のユーザー名による認証を使用できます。

   admin ユーザーと、そのユーザー用に選択したパスワードを使用して認証トークンを要求します。

   ```
   $ keystone --os-username=admin --os-password=ADMIN_PASS ¥
     --os-auth-url=http://controller:35357/v2.0 token-get
   ```

   応答で、ユーザー ID とペアになったトークンを受け取ります。これにより、Identity Service が期待したエンドポイントで実行されていて、ユーザーアカウントが期待したクレデンシャルで確立されていることを検証できます。

3. 認可が期待したとおり動作することを検証します。そうするために、プロジェクトで認可を要求します。

   ```
   $ keystone --os-username=admin --os-password=ADMIN_PASS ¥
     --os-tenant-name=admin --os-auth-url=http://controller:35357/v2.0 ¥
     token-get
   ```

   応答で、指定したプロジェクトの ID を含むトークンを受け取ります。これにより、ユーザーアカウントが指定したプロジェクトで明示的に定義したロールを持ち、プロジェクトが期待したとおりに存在することを検証します。

4. You can also set your --os-* variables in your environment to simplify command-line usage. Set up a admin-openrc.sh file with the admin credentials and admin endpoint:

   ```
   export OS_USERNAME=admin
   export OS_PASSWORD=ADMIN_PASS
   export OS_TENANT_NAME=admin
   export OS_AUTH_URL=http://controller:35357/v2.0
   ```

5. 環境変数を読み込むために、このファイルを source します。

   ```
   $ source admin-openrc.sh
   ```

6. Verify that your admin-openrc.sh file is configured correctly. Run the same command without the --os-* arguments:

   ```
   $ keystone token-get
   ```

   コマンドはトークンと指定されたプロジェクトの ID を返します。これにより、環境変数が正しく設定されていることを確認します。

7. admin アカウントが管理コマンドを実行する権限があることを検証します。

   ```
   $ keystone user-list
   +----------------------------------+-------+---------+-------------------+
   |                id                |  name | enabled |    email          |
   +----------------------------------+-------+---------+-------------------+
   | afea5bde3be9413dbd60e479fddf9228 | admin |   True  | admin@example.com |
   | 32aca1f9a47540c29d6988091f76c934 |  demo |   True  | demo@example.com  |
   ```

```
+----------------------------------+-------+---------+----------------------------------+
```

```
$ keystone user-role-list --user admin --tenant admin
+----------------------------------+----------+----------------------------------
+----------------------------------+
|                id                |   name   |             user_id              |
     tenant_id             |
+----------------------------------+----------+----------------------------------
+----------------------------------+
| 9fe2ff9ee4384b1894a90878d3e92bab | _member_ | afea5bde3be9413dbd60e479fddf9228 |
 e519b772cb43474582fa303da62559e5 |
| 5d3b60b66f1f438b80eaae41a77b5951 |  admin   | afea5bde3be9413dbd60e479fddf9228 |
 e519b772cb43474582fa303da62559e5 |
+----------------------------------+----------+----------------------------------
+----------------------------------+
```

Seeing that the id in the output from the keystone user-list command matches the user_id in the keystone user-role-list command, and that the admin role is listed for that user, for the related tenant, this verifies that your user account has the admin role, which matches the role used in the Identity Service policy.json file.

注記

コマンドラインや環境変数経由でクレデンシャルと Identity Service エンドポイントを定義する限り、すべてのマシンからすべての OpenStack クライアントコマンドを実行できます。詳細は 4章OpenStack クライアントのインストールと設定 [31] を参照してください。

第4章 OpenStack クライアントのインストールと設定

目次

The following sections contain information about working with the OpenStack clients. Recall: in the previous section, you used the keystone client.

You must install the client tools to complete the rest of the installation.

Configure the clients on your desktop rather than on the server so that you have a similar experience to your users.

概要

API コールを行う簡単なコマンドを実行するために、OpenStack コマンドラインクライアントを使用できます。コマンドラインから、または作業を自動化するためのスクリプトでこれらのコマンドを実行できます。OpenStack クレデンシャルを提供する限り、そのマシンでもこれらのコマンドを実行できます。

内部的に、各クライアントコマンドは API リクエストを組み込んだ cURL コマンドを実行します。OpenStack API は、メソッド、URI、メディアタイプ、応答コードを含む HTTP プロトコルを使用する RESTful API です。

これらのオープンソースの Python クライアントは、Linux または Mac OS X システムで実行します。これらは簡単に習得し、使用できます。OpenStack の各サービスは自身のコマンドラインクライアントを持ちます。いくつかのクライアントコマンドでは、コマンドのベースになる API リクエストを表示するために、debug パラメーターを指定できます。これは OpenStack API コールに慣れるために良い方法です。

以下の表は、各 OpenStack サービスのコマンドラインクライアント、そのパッケージ名、説明の一覧です。

表4.1 OpenStack のサービスとクライアント

サービス	クライアント	パッケージ	説明
Block Storage	cinder	python-cinderclient	ボリュームを作成、管理します。
Compute	nova	python-novaclient	イメージ、インスタンス、フレーバーを作成、管理します。
Database Service	trove	python-troveclient	Create and manage databases.

サービス	クライアント	パッケージ	説明
Identity	keystone	python-keystoneclient	ユーザー、プロジェクト、ロール、エンドポイント、クレデンシャルを作成、管理します。
Image Service	glance	python-glanceclient	イメージを作成、管理します。
Networking	neutron	python-neutronclient	ゲストサーバー用のネットワークを設定します。このクライアントは以前 quantum として知られていました。
Object Storage	swift	python-swiftclient	統計情報を収集し、項目を一覧表示し、メタデータを更新し、Object Storage サービスにより保存されたファイルをアップロード、ダウンロード、削除します。
Orchestration	heat	python-heatclient	テンプレートからスタックを起動し、イベントやリソースを含む実行中のスタックの詳細を表示し、スタックを更新、削除します。
Telemetry	ceilometer	python-ceilometerclient	OpenStack 全体の測定項目を作成、収集します。

An OpenStack common client is in development.

OpenStack コマンドラインクライアントのインストール

前提ソフトウェアと各 OpenStack クライアント用の Python パッケージをインストールします。

注記

各コマンドに対して、nova のように、インストールするクライアントの小文字の名前で PROJECT を置き換えます。各クライアントに対して繰り返します。

表4.2 前提ソフトウェア

前提	説明
Python 2.6 またはそれ以降	現在、クライアントは Python 3 をサポートしません。
setuptools パッケージ	Mac OS X に標準でインストールされます。 多くの Linux ディストリビューションはインストールしやすい setuptools パッケージを提供します。インストールパッケージを検索するために、パッケージマネージャーで setuptools を検索します。見つけられない場合、http://pypi.python.org/pypi/setuptools から setuptools パッケージを直接ダウンロードします。 Microsoft Windows に setuptools をインストールする推奨の方法は setuptools ウェブサイト で提供されているドキュメントに従うことです。他の選択肢は hristoph Gohlke (http://www.lfd.uci.edu/~gohlke/pythonlibs/#setuptools) によりメンテナンスされている非公式のバイナリインストーラーを使用することです。
pip パッケージ	Linux、Mac OS X、Microsoft Windows システムにクライアントをインストールするために、pip を使用します。これは使いやすく、必ず Python Package Index から最新バージョンのクライアントを取得します。後からパッケージの更新や削除ができます。 お使いのシステムのパッケージマネージャーを利用して pip をインストールします。

前提	説明

	Mac OS X.
	
	`# easy_install pip`
	
	Microsoft Windows. Make sure that the C:¥Python27¥Scripts directory is defined in the PATH environment variable, and use the easy_install command from the setuptools package:
	
	`C:¥>easy_install pip`
	
	Another option is to use the unofficial binary installer provided by Christoph Gohlke (http://www.lfd.uci.edu/~gohlke/pythonlibs/#pip).
	
	Ubuntu 12.04. A packaged version enables you to use dpkg or aptitude to install the python-novaclient:
	
	`# aptitude install python-novaclient`
	
	Ubuntu and Debian.
	
	`# aptitude install python-pip`
	
	RHEL、CentOS、Fedora. A packaged version available in RDO enables you to use yum to install the clients:
	
	`# yum install python-PROJECTclient`
	
	あるいは、クライアントのインストールを管理するために pip をインストールして使用します。
	
	`# yum install python-pip`
	
	openSUSE 12.2 およびそれ以前. A packaged version available in the Open Build Service enables you to use rpm or zypper to install the python-novaclient:
	
	`# zypper install python-PROJECT`
	
	Alternatively, install pip and use it to manage client installation:
	
	`# zypper install python-pip`
	
	openSUSE 12.3 およびそれ以降. A packaged version enables you to use rpm or zypper to install the clients:
	
	`# zypper install python-PROJECTclient`

クライアントのインストール

Use pip to install the OpenStack clients on a Linux, Mac OS X or Microsoft Windows system. It is easy and ensures that you get the latest version of the client from the Python Package Index. Also, pip lets you update or remove a package. After you install the clients, you must source an PROJECT-openrc.sh file to set required environment variables before you can request OpenStack services through the clients or the APIs.

- それぞれ以下のとおりクライアントをインストールします。
  - Mac OS X または Linux の場合:

    ```
    # pip install python-PROJECTclient
    ```

  - Microsoft Windows の場合:

    ```
    C:¥>pip install python-PROJECTclient
    ```

ここで PROJECT はプロジェクトの名前で、以下の値のどれかです。

- ceilometer - Telemetry API。
- cinder - Block Storage API and extensions.
- glance - Image Service API。
- heat - Orchestration API。
- keystone - Identity service API and extensions.
- neutron - Networking API。
- nova - Compute API とその拡張。
- swift - Object Storage API。
- trove - Database Service API.

たとえば、nova クライアントをインストールする場合、このコマンドを実行します。

```
# pip install python-novaclient
```

nova クライアントを削除する場合、このコマンドを実行します。

```
# pip uninstall python-novaclient
```

注記

To upgrade a package, add the --upgrade option to the pip command.

たとえば、nova クライアントを更新する場合、このコマンドを実行します。

```
# pip install --upgrade python-novaclient
```

OpenStack RC ファイル

OpenStack コマンドラインクライアントに必要な環境変数を設定するために、環境ファイルを作成する必要があります。このプロジェクト固有の環境ファイルは、すべての OpenStack サービスを使用するクレデンシャルを含みます。

このファイルを読み込むと、環境変数が現在のシェルに対して設定されます。この変数により OpenStack クライアントコマンドがクラウドで実行中の OpenStack サービスとやりとりできるようになります。

Microsoft Windows における環境変数

環境変数ファイルを用いて環境変数を定義することは、Microsoft Windows で一般的な手法ではありません。環境変数は通常、システムのプロパティダイアログの詳細設定タブで定義されます。

OpenStack RC ファイルの作成と読み込み

1. Create the PROJECT-openrc.sh file and add the authentication information:

```
export OS_USERNAME=admin
export OS_PASSWORD=ADMIN_PASS
export OS_TENANT_NAME=admin
export OS_AUTH_URL=http://controller:35357/v2.0
```

2. On any shell from where you want to run OpenStack commands, source the PROJECT-openrc.sh file for the respective project. In this example, you source the admin-openrc.sh file for the admin project:

```
$ source admin-openrc.sh
```

環境変数値の上書き

When you run OpenStack client commands, you can override some environment variable settings by using the options that are listed at the end of the nova help output. For example, you can override the OS_PASSWORD setting in the PROJECT-openrc.sh file by specifying a password on a nova command, as follows:

```
$ nova --password <password> image-list
```

ここで password はお使いのパスワードです。

Create openrc.sh files

As explained in 「OpenStack RC ファイルの作成と読み込み」 [35], use the credentials from 「ユーザー、プロジェクト、ロールの定義」 [26] and create the following PROJECT-openrc.sh files:

- admin-openrc.sh for the administrative user
- demo-openrc.sh for the normal user:

```
export OS_USERNAME=demo
export OS_PASSWORD=DEMO_PASS
export OS_TENANT_NAME=demo
export OS_AUTH_URL=http://controller:35357/v2.0
```

第5章 Image Service の設定

目次

The OpenStack Image Service enables users to discover, register, and retrieve virtual machine images. Also known as the glance project, the Image Service offers a REST API that enables you to query virtual machine image metadata and retrieve an actual image. You can store virtual machine images made available through the Image Service in a variety of locations from simple file systems to object-storage systems like OpenStack Object Storage.

重要

For simplicity, this guide configures the Image Service to use the file back end. This means that images uploaded to the Image Service are stored in a directory on the same system that hosts the service. By default, this directory is /var/lib/glance/images/.

Before you proceed, ensure that the system has sufficient space available in this directory to store virtual machine images and snapshots. At an absolute minimum, several gigabytes of space should be available for use by the Image Service in a proof of concept deployment. To see requirements for other back ends, see Configuration Reference.

Image Service の概要

Image Service は以下のコンポーネントを含みます。

- glance-api。イメージの検索・取得・保存に対する Image API コールを受け付けます。
- glance-registry。イメージに関するメタデータを保存・処理・取得します。メタデータは容量や形式などの項目を含みます。

Security note

The registry is a private internal service meant only for use by the Image Service itself. Do not expose it to users.

- データベース。イメージのメタデータを保存します。お好みに合わせてデータベースを選択できます。多くの環境では MySQL か SQlite を使用します。
- Storage repository for image files. The Image Service supports a variety of repositories including normal file systems, Object Storage, RADOS block

devices, HTTP, and Amazon S3. Some types of repositories support only read-only usage.

キャッシュをサポートするために Image Service で実行されるいくつかの定期的なプロセス。複製サービスにより、クラスター全体で一貫性と可用性が確保されます。他の定期的なプロセスにオーディター、アップデーター、リーパーなどがあります。

図1.1「概念アーキテクチャー」 [3]に示されているように、Image Service は IaaS 全体像の中で中心になります。エンドユーザーや Compute のコンポーネントからイメージやイメージのメタデータに対する API リクエストを受け付けます。また、そのディスクファイルを Object Storage Service に保存できます。

Image Service のインストール

OpenStack Image Service は仮想ディスクイメージの登録管理者として動作します。ユーザーは新しいイメージを追加できます。イメージのスナップショットを既存のサーバーの直接ストレージから取得できます。バックアップのため、または新しいサーバーを起動するためのテンプレートとしてスナップショットを使用します。登録済みイメージを Object Storage に保存できます。例えば、イメージをシンプルなファイルシステムや外部ウェブサーバーに保存できます。

注記

この手順は 「Identity Service のインストールの検証」 [28] に記載されているとおり、適切な環境変数にクレデンシャルを設定していると仮定しています。

1. コントローラーノードに Image Service をインストールします。

```
# yum install openstack-glance
```

2. Image Service はイメージに関する情報をデータベースに保存します。このガイドの例は、他の OpenStack サービスにより使用されている MySQL データベースを使用します。

   データベースの位置を設定します。Image Service はそれぞれの設定ファイルを用いて glance-api サービスと glance-registry サービスを提供します。このセクションを通して両方の設定ファイルを更新する必要があります。GLANCE_DBPASS をお使いの Image Service データベースのパスワードで置き換えます。

```
# openstack-config --set /etc/glance/glance-api.conf database ¥
  connection mysql://glance:GLANCE_DBPASS@controller/glance
# openstack-config --set /etc/glance/glance-registry.conf database ¥
  connection mysql://glance:GLANCE_DBPASS@controller/glance
```

3. Configure the Image Service to use the message broker:

```
# openstack-config --set /etc/glance/glance-api.conf DEFAULT ¥
  rpc_backend qpid
# openstack-config --set /etc/glance/glance-api.conf DEFAULT ¥
  qpid_hostname controller
```

4. root としてログインするために、作成したパスワードを使用します。glance データベースユーザーを作成します。

```
$ mysql -u root -p
mysql> CREATE DATABASE glance;
mysql> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'localhost' ¥
IDENTIFIED BY 'GLANCE_DBPASS';
mysql> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'%' ¥
IDENTIFIED BY 'GLANCE_DBPASS';
```

5. Image Service 用のデータベーステーブルを作成します。

```
# su -s /bin/sh -c "glance-manage db_sync" glance
```

6. Create a glance user that the Image Service can use to authenticate with the Identity service. Choose a password and specify an email address for the glance user. Use the service tenant and give the user the admin role:

```
$ keystone user-create --name=glance --pass=GLANCE_PASS ¥
  --email=glance@example.com
$ keystone user-role-add --user=glance --tenant=service --role=admin
```

7. Image Service が認証用に Identity Service を使用するよう設定します。

   以下のコマンドを実行します。Identity Service で glance ユーザー用に選択したパスワードで GLANCE_PASS を置き換えます。

```
# openstack-config --set /etc/glance/glance-api.conf keystone_authtoken ¥
  auth_uri http://controller:5000
# openstack-config --set /etc/glance/glance-api.conf keystone_authtoken ¥
  auth_host controller
# openstack-config --set /etc/glance/glance-api.conf keystone_authtoken ¥
  admin_tenant_name service
# openstack-config --set /etc/glance/glance-api.conf keystone_authtoken ¥
  admin_user glance
# openstack-config --set /etc/glance/glance-api.conf keystone_authtoken ¥
  admin_password GLANCE_PASS
# openstack-config --set /etc/glance/glance-api.conf paste_deploy ¥
  flavor keystone
# openstack-config --set /etc/glance/glance-registry.conf keystone_authtoken ¥
  auth_uri http://controller:5000
# openstack-config --set /etc/glance/glance-registry.conf keystone_authtoken ¥
  auth_host controller
# openstack-config --set /etc/glance/glance-registry.conf keystone_authtoken ¥
  admin_tenant_name service
# openstack-config --set /etc/glance/glance-registry.conf keystone_authtoken ¥
  admin_user glance
# openstack-config --set /etc/glance/glance-registry.conf keystone_authtoken ¥
  admin_password GLANCE_PASS
# openstack-config --set /etc/glance/glance-registry.conf paste_deploy ¥
  flavor keystone
```

8. Register the Image Service with the Identity service so that other OpenStack services can locate it. Register the service and create the endpoint:

```
$ keystone service-create --name=glance --type=image ¥
  --description="OpenStack Image Service"
$ keystone endpoint-create ¥
  --service-id=$(keystone service-list | awk '/ image / {print $2}') ¥
  --publicurl=http://controller:9292 ¥
  --internalurl=http://controller:9292 ¥
  --adminurl=http://controller:9292
```

9. glance-api、glance-registry サービスを起動し、システムの起動時にそれらが起動するよう設定します。

```
# service openstack-glance-api start
# service openstack-glance-registry start
# chkconfig openstack-glance-api on
# chkconfig openstack-glance-registry on
```

Image Service のインストールの検証

Image Service のインストールをテストするために、OpenStack で動作することが知られている仮想マシンイメージを何かしらダウンロードします。例えば、CirrOS（CirrOS ダウンロード）は OpenStack 環境をテストするためによく使用される小さなテストイメージです。ここでは 64 ビット CirrOS QCOW2 イメージを使用します。

ダウンロード方法とイメージ構築の詳細はOpenStack 仮想マシンイメージガイドを参照してください。イメージの管理方法の詳細はOpenStack ユーザーガイドを参照してください。

1. Download the image into a dedicated directory using wget or curl:

```
$ mkdir images
$ cd images/
$ wget http://cdn.download.cirros-cloud.net/0.3.2/cirros-0.3.2-x86_64-disk.img
```

2. イメージを Image Service にアップロードします。

```
$ glance image-create --name=imageLabel --disk-format=fileFormat ¥
  --container-format=containerFormat --is-public=accessValue < imageFile
```

各項目:

imageLabel　任意のラベル。ユーザーがイメージを参照する名前。

fileFormat　イメージファイルの形式を指定します。有効な形式は qcow2, raw, vhd, vmdk, vdi, iso, aki, ari, ami です。

You can verify the format using the file command:

```
$ file cirros-0.3.2-x86_64-disk.img
cirros-0.3.2-x86_64-disk.img: QEMU QCOW Image (v2), 41126400
 bytes
```

containerFormat　コンテナーの形式を指定します。有効な形式は bare, ovf, aki, ari, ami です。

仮想マシンに関するメタデータを含むイメージファイルがファイル形式ではないことを示すために bare を指定します。この項目が現在必須となっていますが、実際はすべての OpenStack により使用されるわけではなく、システム動作に影響を与えません。この値がどこでも使用されないため、常に bare をコンテナー形式として指定すると安全です。

accessValue　イメージのアクセス権を指定します。

- true - すべてのユーザーがイメージを表示および使用できます。
- false - 管理者のみがイメージを表示および使用できます。

imageFile ダウンロードしたイメージファイルの名前を指定します。

例:

```
$ source admin-openrc.sh
$ glance image-create --name "cirros-0.3.2-x86_64" --disk-format qcow2 ¥
  --container-format bare --is-public True --progress < cirros-0.3.2-x86_64-disk.img
+------------------+--------------------------------------+
| Property         | Value                                |
+------------------+--------------------------------------+
| checksum         | 64d7c1cd2b6f60c92c14662941cb7913     |
| container_format | bare                                 |
| created_at       | 2014-04-08T18:59:18                  |
| deleted          | False                                |
| deleted_at       | None                                 |
| disk_format      | qcow2                                |
| id               | acafc7c0-40aa-4026-9673-b879898e1fc2 |
| is_public        | True                                 |
| min_disk         | 0                                    |
| min_ram          | 0                                    |
| name             | cirros-0.3.2-x86_64                  |
| owner            | efa984b0a914450e9a47788ad330699d     |
| protected        | False                                |
| size             | 13167616                             |
| status           | active                               |
| updated_at       | 2014-01-08T18:59:18                  |
+------------------+--------------------------------------+
```

注記

返されたイメージ ID は動的に変更されるため、導入環境によりこの例で示されているものと異なる ID が生成されます。

3. イメージがアップロードされたことを確認し、その属性を表示します。

```
$ glance image-list
```

```
+--------------------------------------+---------------------+-------------
+------------------+----------+--------+
| ID                                   | Name                | Disk Format | Container
 Format | Size     | Status |
+--------------------------------------+---------------------+-------------
+------------------+----------+--------+
| acafc7c0-40aa-4026-9673-b879898e1fc2 | cirros-0.3.2-x86_64 | qcow2       | bare
     | 13167616 | active |
+--------------------------------------+---------------------+-------------
+------------------+----------+--------+
```

代わりに、Image Service にアップロードしたものは、--copy-from パラメーターを使用することにより、ファイルを保存するためのローカルディスク領域を使用する必要なく実行できます。

例:

```
$ glance image-create --name="cirros-0.3.2-x86_64" --disk-format=qcow2 ¥
  --container-format=bare --is-public=true ¥
  --copy-from http://cdn.download.cirros-cloud.net/0.3.2/cirros-0.3.2-x86_64-disk.img
+------------------+--------------------------------------+
| Property         | Value                                |
+------------------+--------------------------------------+
| checksum         | 64d7c1cd2b6f60c92c14662941cb7913     |
| container_format | bare                                 |
| created_at       | 2014-04-08T06:13:18                  |
| deleted          | False                                |
| disk_format      | qcow2                                |
| id               | 3cce1e32-0971-4958-9719-1f92064d4f54 |
| is_public        | True                                 |
| min_disk         | 0                                    |
| min_ram          | 0                                    |
| name             | cirros-0.3.2-x86_64                  |
| owner            | efa984b0a914450e9a47788ad330699d     |
| protected        | False                                |
| size             | 13167616                             |
| status           | active                               |
| updated_at       | 2014-04-08T06:13:20                  |
+------------------+--------------------------------------+
```

第6章 Compute Service の設定

目次

Compute Service

Compute Service はクラウドコンピューティングのファブリックコントローラーです。これは Iaas システムの中心部です。クラウドコンピューティングシステムをホストして管理するために使用します。主要なモジュールは Python で実装されます。

Compute は、認証のために Identity Service と、イメージのために Image Service と、ユーザーと管理者のインターフェースのために Dashboard とやりとりします。イメージへのアクセスはプロジェクトやユーザーにより制限されます。クォータはプロジェクトごとに制限されます（例: インスタンス数）。Compute Service は、標準的なハードウェアで水平的にスケールし、必要に応じてインスタンスを起動するためにイメージをダウンロードします。

The Compute service is made up of the following functional areas and their underlying components:

API

- nova-api サービス。エンドユーザーの Compute API コールを受け付けて処理します。OpenStack Compute API、Amazon EC2 API、および管理操作を実行するための特権ユーザー用の特別な Admin API をサポートします。また、インスタンスの実行やいくつかのポリシーの強制など、多くのオーケストレーション作業を開始します。

- nova-api-metadata サービス。インスタンスからメタデータリクエストを受け取ります。nova-api-metadata サービスは一般的に、nova-network を用いてマルチホストモードで実行しているときのみ使用されます。詳細は クラウド管理者ガイドのメタデータサービスを参照してください。

  Debian システムの場合、nova-api パッケージに含まれます。debconf 経由で選択できます。

Compute コア

- nova-compute プロセス。ハイパーバイザーの API 経由で仮想マシンインスタンスを作成および終了するワーカーデーモンです。たとえば、XenServer/XCP 用の XenAPI、KVM や QEMU 用の libvirt、VMware 用の VMwareAPI などです。そのように実行されるプロセスはかなり複雑ですが、基本はシンプルです。キューから操作を受け取り、KVM インスタンスの起動などの一連のシステムコマンドを実行し、データベースで状態を更新している間にそれらを実施します。

- nova-scheduler プロセス。Compute のコードの中で概念的に最も簡単なものです。キューから仮想マシンインスタンスのリクエストを受け取り、どのコンピュートノードで実行すべきかを判断します。

- nova-conductor モジュール。nova-compute とデータベースの間のやりとりを取り次ます。nova-compute により行われるクラウドデータベースへの直接アクセスを削減することが目標です。nova-conductor モジュールは水平的にスケールします。しかしながら、nova-compute を実行しているノードに導入しません。詳細は A new Nova service: nova-conductor を参照してください。

仮想マシン用ネットワーク

- nova-network ワーカーデーモン。nova-compute と同じように、キューからネットワークのタスクを受け取り、ネットワークを操作するためにタスクを実行します。ブリッジインターフェースのセットアップや iptables ルールの変更などです。この機能は別の OpenStack サービスである OpenStack Networking に移行されています。

- nova-dhcpbridge スクリプト。dnsmasq dhcp-script 機能を使用して、IP アドレスのリース情報を追跡し、それらをデータベースに記録します。この機能は OpenStack Networking に移行されています。OpenStack Networking は別のスクリプトを提供します。

コンソールインターフェース

- nova-consoleauth デーモン。コンソールプロキシを提供するユーザーのトークンを認可します。nova-novncproxy と nova-xvpnvcproxy を参照してください。このサービスはコンソールプロキシを動作させるために実行する必要があります。どちらの種類の多くのプロキシもクラスター設定で単一の nova-consoleauth サービスに対して実行されます。詳細は nova-consoleauth について を参照してください。

- nova-novncproxy デーモン。VNC 接続で実行中の仮想マシンにアクセスするためのプロキシを提供します。ブラウザーベースの novnc クライアントをサポートします。

- nova-xvpnvncproxy デーモン。VNC 接続で実行中の仮想マシンにアクセスするためのプロキシを提供します。OpenStack 向けに特別に設計された Java クライアントをサポートします。

- nova-cert デーモン。x509 証明書を管理します。

イメージ管理（EC2 シナリオ）

- nova-objectstore デーモン。イメージを Image Service に登録するための S3 インターフェースを提供します。主に euca2ools をサポートする必要があるインストール環境のために使用されます。euca2ools は S3 言語 で nova-objectstore とやりとりします。また、nova-objectstore は S3 リクエストを Image Service リクエストに変換します。

- euca2ools クライアント。クラウドリソースを管理するための一組のコマンドラインインタプリターコマンドです。OpenStack のモジュールではありませんが、この EC2 インターフェースをサポートするために、nova-api を設定できます。詳細は Eucalyptus 2.0 のドキュメント を参照してください。

コマンドラインクライアントと他のインターフェース

- nova クライアント。ユーザーがプロジェクト管理者やエンドユーザーとしてコマンドを投入できます。

- nova-manage クライアント。クラウド管理者がコマンドを投入できます。

他のコンポーネント

- The queue. A central hub for passing messages between daemons. Usually implemented with RabbitMQ, but could be any AMQP message queue, such as Apache Qpid or Zero MQ.

- SQL database. Stores most build-time and runtime states for a cloud infrastructure. Includes instance types that are available for use, instances in use, available networks, and projects. Theoretically, OpenStack Compute can support any database that SQL-Alchemy supports, but the only databases widely used are SQLite3 databases (only appropriate for test and development work), MySQL, and PostgreSQL.

The Compute service interacts with other OpenStack services: Identity Service for authentication, Image Service for images, and the OpenStack dashboard for a web interface.

Compute コントローラーサービスのインストール

Compute は仮想マシンインスタンスを起動できるようにするためのサービス群です。これらのサービスを別々のノードで実行することも同じノードで実行することも設定できます。このガイドでは、多くのサービスはコントローラーノードで実行し、仮想マシンを起動するサービスはコンピュート専用ノードで実行します。このセクションは、コントローラーノードにこれらのサービスをインストールし、設定する方法を示します。

1. コントローラーノードに必要な Compute のパッケージをインストールします。

```
# yum install openstack-nova-api openstack-nova-cert openstack-nova-conductor ¥
  openstack-nova-console openstack-nova-novncproxy openstack-nova-scheduler ¥
  python-novaclient
```

2. Compute は情報を保存するためにデータベースを使用します。このガイドでは、コントローラーノードで MySQL データベースを使用します。Compute をデータベースの位置とクレデンシャルで設定します。NOVA_DBPASS を後のステップで作成するデータベース用パスワードで置き換えます。

```
# openstack-config --set /etc/nova/nova.conf ¥
  database connection mysql://nova:NOVA_DBPASS@controller/nova
```

3. Compute が Qpid メッセージブローカーを使用するよう設定するために、これらの設定キーを設定します。

```
# openstack-config --set /etc/nova/nova.conf ¥
  DEFAULT rpc_backend nova.openstack.common.rpc.impl_qpid
# openstack-config --set /etc/nova/nova.conf DEFAULT qpid_hostname controller
```

4. Set the my_ip, vncserver_listen, and vncserver_proxyclient_address configuration options to the management interface IP address of the controller node:

```
# openstack-config --set /etc/nova/nova.conf DEFAULT my_ip 10.0.0.11
# openstack-config --set /etc/nova/nova.conf DEFAULT vncserver_listen 10.0.0.11
# openstack-config --set /etc/nova/nova.conf DEFAULT vncserver_proxyclient_address 10.0.
0.11
```

5. root としてログインするために、前に作成したパスワードを使用します。nova データベースユーザーを作成します。

```
$ mysql -u root -p
mysql> CREATE DATABASE nova;
mysql> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost' ¥
IDENTIFIED BY 'NOVA_DBPASS';
mysql> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%' ¥
IDENTIFIED BY 'NOVA_DBPASS';
```

6. Compute サービスのテーブルを作成します。

```
# su -s /bin/sh -c "nova-manage db sync" nova
```

7. Compute が Identity Service で認証するために使用する nova ユーザーを作成します。service プロジェクトを使用し、ユーザーに admin ロールを与えます。

```
$ keystone user-create --name=nova --pass=NOVA_PASS --email=nova@example.com
$ keystone user-role-add --user=nova --tenant=service --role=admin
```

8. コントローラーで実行している Identity Service でこれらのクレデンシャルを使用するよう Compute を設定します。NOVA_PASS をお使いの Compute パスワードで置き換えます。

```
# openstack-config --set /etc/nova/nova.conf DEFAULT auth_strategy keystone
# openstack-config --set /etc/nova/nova.conf keystone_authtoken auth_uri http:/
/controller:5000
# openstack-config --set /etc/nova/nova.conf keystone_authtoken auth_host controller
# openstack-config --set /etc/nova/nova.conf keystone_authtoken auth_protocol http
# openstack-config --set /etc/nova/nova.conf keystone_authtoken auth_port 35357
# openstack-config --set /etc/nova/nova.conf keystone_authtoken admin_user nova
# openstack-config --set /etc/nova/nova.conf keystone_authtoken admin_tenant_name service
# openstack-config --set /etc/nova/nova.conf keystone_authtoken admin_password NOVA_PASS
```

9. 他の OpenStack サービスから使用できるように、Compute を Identity Service に登録します。サービスを登録し、エンドポイントを指定します。

```
$ keystone service-create --name=nova --type=compute ¥
  --description="OpenStack Compute"
$ keystone endpoint-create ¥
  --service-id=$(keystone service-list | awk '/ compute / {print $2}') ¥
  --publicurl=http://controller:8774/v2/%¥(tenant_id¥)s ¥
  --internalurl=http://controller:8774/v2/%¥(tenant_id¥)s ¥
  --adminurl=http://controller:8774/v2/%¥(tenant_id¥)s
```

10. Compute サービスを起動し、システム起動時に起動するよう設定します。

```
# service openstack-nova-api start
# service openstack-nova-cert start
# service openstack-nova-consoleauth start
# service openstack-nova-scheduler start
# service openstack-nova-conductor start
# service openstack-nova-novncproxy start
# chkconfig openstack-nova-api on
# chkconfig openstack-nova-cert on
# chkconfig openstack-nova-consoleauth on
# chkconfig openstack-nova-scheduler on
# chkconfig openstack-nova-conductor on
# chkconfig openstack-nova-novncproxy on
```

11. 設定を検証するために、使用可能なイメージを一覧表示します。

```
$ nova image-list
+--------------------------------------+---------------------+--------+--------+
| ID                                   | Name                | Status | Server |
+--------------------------------------+---------------------+--------+--------+
| acafc7c0-40aa-4026-9673-b879898e1fc2 | cirros-0.3.2-x86_64 | ACTIVE |        |
+--------------------------------------+---------------------+--------+--------+
```

コンピュートノードの設定

コントローラーノードで Compute サービスを設定した後、他のシステムをコンピュートノードとして設定する必要があります。コンピュートノードはコントローラーノードからリクエストを受け取り、仮想マシンインスタンスをホストします。単一ノードですべてのサービスを実行することもできます。しかし、このガイドの例では分離したシステムを使用します。これにより、このセクションにある説明に従って、追加のコンピュートノードを追加して、水平的にスケールさせることが容易になります。

Compute サービスは仮想マシンインスタンスを実行するためにハイパーバイザーに依存します。OpenStack はさまざまなハイパーバイザーを使用できますが、このガイドは KVM を使用します。

1. システムを設定します。2章環境の基本設定 [7] にある方法を使用します。以下の項目はコントローラーノードと異なることに注意してください。

   - Use different IP addresses when you configure eth0. This guide uses 10.0.0.31 for the management network of the first compute node.

     If you run OpenStack Networking (neutron), configure eth1 as instance tunnels interface with IP address 10.0.1.31 for the first compute node. For details, see the instructions in 「コンピュートノード」 [11]

     If you run legacy networking (nova-compute), do not configure eth1 with a static IP address. The networking component of OpenStack assigns and configures an IP address. For details, see the instructions in 「コンピュートノード」 [16].

   - ホスト名を compute1 に設定します。確認するために、uname -n を使用します。両方のノードの IP アドレスとホスト名が各システムの /etc/hosts にあることを確認します。

   - コントローラーノードから同期します。「Network Time Protocol (NTP)」 [18]にある手順に従ってください。

   - MySQL クライアントライブラリをインストールします。MySQL データベースサーバーをインストールする必要や MySQL サービスを起動する必要がありません。

   - 使用しているディストリビューションの OpenStack パッケージを有効化します。「OpenStack パッケージ」 [20] を参照してください。

2. オペレーティングシステムの設定後、Compute サービス向けに適切なパッケージをインストールします。

```
# yum install openstack-nova-compute
```

3. /etc/nova/nova.conf 設定ファイルを編集します。

```
# openstack-config --set /etc/nova/nova.conf database connection mysql://
nova:NOVA_DBPASS@controller/nova
# openstack-config --set /etc/nova/nova.conf DEFAULT auth_strategy keystone
# openstack-config --set /etc/nova/nova.conf keystone_authtoken auth_uri http:/
/controller:5000
# openstack-config --set /etc/nova/nova.conf keystone_authtoken auth_host controller
# openstack-config --set /etc/nova/nova.conf keystone_authtoken auth_protocol http
# openstack-config --set /etc/nova/nova.conf keystone_authtoken auth_port 35357
# openstack-config --set /etc/nova/nova.conf keystone_authtoken admin_user nova
# openstack-config --set /etc/nova/nova.conf keystone_authtoken admin_tenant_name service
# openstack-config --set /etc/nova/nova.conf keystone_authtoken admin_password NOVA_PASS
```

4. これらの設定キーを設定することにより、Compute サービスが Qpid メッセージブローカーを使用するよう設定します。

```
# openstack-config --set /etc/nova/nova.conf ¥
  DEFAULT rpc_backend nova.openstack.common.rpc.impl_qpid
```

```
# openstack-config --set /etc/nova/nova.conf DEFAULT qpid_hostname controller
```

5. インスタンスへのリモートコンソールアクセスを提供するよう Compute を設定します。

```
# openstack-config --set /etc/nova/nova.conf DEFAULT my_ip 10.0.0.31
# openstack-config --set /etc/nova/nova.conf DEFAULT vnc_enabled True
# openstack-config --set /etc/nova/nova.conf DEFAULT vncserver_listen 0.0.0.0
# openstack-config --set /etc/nova/nova.conf DEFAULT vncserver_proxyclient_address 10.0.
0.31
# openstack-config --set /etc/nova/nova.conf ¥
  DEFAULT novncproxy_base_url http://controller:6080/vnc_auto.html
```

6. Image Service を実行するホストを指定します。

```
# openstack-config --set /etc/nova/nova.conf DEFAULT glance_host controller
```

7. Compute をテスト目的で仮想マシンにインストールする場合、ハイパーバイザーと CPU がネストハードウェア支援をサポートするかどうかを、以下のコマンドを使用して確認する必要があります。

```
$ egrep -c '(vmx|svm)' /proc/cpuinfo
```

このコマンドが 1 以上の値を返す場合、ハイパーバイザーと CPU がネストハードウェア支援をサポートし、追加の設定は必要ありません。

このコマンドが 0 を返す場合、ハイパーバイザーと CPU がネストハードウェア支援をサポートしません。libvirt は KVM の代わりに QEMU を使用する必要があります。QEMU を使用するために libvirt を設定します。

```
# openstack-config --set /etc/nova/nova.conf libvirt virt_type qemu
```

8. Start the Compute service and configure it to start when the system boots:

```
# service libvirtd start
# service messagebus start
# chkconfig libvirtd on
# chkconfig messagebus on
# service openstack-nova-compute start
# chkconfig openstack-nova-compute on
```

第7章 Networking Service の追加

目次

警告

Icehouse 向けにこのドキュメントを更新中です。この作業中、構造や内容の問題が見つかるかもしれません。

Configuring networking in OpenStack can be a bewildering experience. This guide provides step-by-step instructions for both OpenStack Networking (neutron) and the legacy networking (nova-network) service. If you are unsure which to use, we recommend trying OpenStack Networking because it offers a considerable number of features and flexibility including plug-ins for a variety of emerging products supporting virtual networking. See the Networking chapter of the OpenStack Cloud Administrator Guide for more information.

Networking (neutron)

Neutron の概念

Nova Networking のように、Neutron は OpenStack インストール環境の SDN を管理します。しかしながら、Nova Networking と異なり、テナントごとのプライベートネットワークなど、より高度な仮想ネットワークトポロジー向けに Neutron を設定できます。

Neutron はネットワーク、サブネット、ルーターのオブジェクトの抽象化を実現します。それぞれ、対応する物理的なものの機能を模倣します。ネットワークがサブネットを含みます。ルーターがサブネットやネットワーク間の通信を中継します。

すべての Neutron 環境は少なくとも 1 つの外部ネットワークを持ちます。このネットワークは、他のネットワークと異なり、ほとんど仮想的に定義されたネットワークではありません。これは OpenStack インストール環境の外部からアクセス可能な外部ネットワークの一部のビューであることを意味します。Neutron 外部ネットワークの IP アドレスは外部ネットワークにある何らかの物理的なものによりアクセスできます。このネットワークがほとんど外部ネットワークの一部を表すため、DHCP はこのネットワークで無効化されます。

外部ネットワークに加えて、あらゆる Neutron のセットアップ環境は 1 つ以上の内部ネットワークを持ちます。これらの SDN は仮想マシンに直接接続します。あらゆる指定された内部ネットワークにある仮想マシン、またはインターフェース経由で同様のルーターに接続されたサブネットにある仮想マシンのみが、そのネットワークに接続された仮想マシンに直接アクセスできます。

外部ネットワークが仮想マシンにアクセスするため、またその逆のため、ネットワーク間のルーターが必要になります。各ルーターはネットワークに接続された 1 つのゲート

ウェイとサブネットに接続された多くのインターフェースを持ちます。物理ルーターのように、同じルーターに接続された他のサブネットにあるマシンにサブネットがアクセスできます。また、マシンはルーターに対するゲートウェイ経由で外部ネットワークにアクセスできます。

さらに、内部ネットワークにたどり着くために外部ネットワークに IP アドレスを割り当てることができます。何かがサブネットに接続されたとき必ず、その接続がポートと呼ばれます。外部ネットワークの IP アドレスを仮想マシンのポートに関連づけられます。このように、外部ネットワークのものが仮想マシンにアクセスできます。

Neutron は セキュリティグループ もサポートします。セキュリティグループにより、管理者がグループでファイアウォールルールを定義できます。仮想マシンは 1 つ以上のセキュリティグループに属します。Neutron が、ポート、ポート範囲、または通信種別をブロックするかブロックしないかのために、これらのセキュリティグループにあるルールを仮想マシンに対して適用します。

Neutron が使用する各プラグインはそれぞれ独自の概念を持ちます。Neutron を稼働されるために必須ではありませんが、これらの概念を理解することにより、Neutron をセットアップする役に立つでしょう。すべての Neutron インストール環境は、コアプラグインとセキュリティグループプラグイン（またはただの No-Op セキュリティグループプラグイン）を使用します。さらに、Firewall-as-a-service (FWaaS) と Load-balancing-as-a-service (LBaaS) プラグインが利用可能です。

Open vSwitch の概念

Open vSwitch プラグインは最も人気のあるコアプラグインの一つです。Open vSwitch の設定はブリッジとポートから構成されます。ポートは物理インターフェースやパッチケーブルのような他のものへの接続を意味します。ブリッジのあらゆるポートからのパケットは、そのブリッジにあるすべての他のポートと共有されます。ブリッジは Open vSwitch 仮想パッチケーブルまたは Linux 仮想イーサネットケーブル（veth）から接続されます。また、ブリッジは Linux にネットワークインターフェースとして認識されるため、それらに IP アドレスを割り当てることができます。

Neutron では、br-int という統合ブリッジが仮想マシンおよび関連するサービスを直接接続します。br-ex という外部ブリッジが外部ネットワークに接続します。最後に、Open vSwitch プラグインの VLAN 設定が各物理ネットワークと関連づけられたブリッジを使用します。

ブリッジの定義に加えて、Open vSwitch は OpenFlow に対応しています。これにより、ネットワークのフロールールを定義できるようになります。特定の設定が VLAN 間のパケットを転送するために、これらのルールを使用します。

最後に、Open vSwitch のいくつかの設定はネットワーク名前空間を使用します。この名前空間により、Linux が他の名前空間に認識されない一意な名前空間の中にアダプターをグループ化できます。これで、同じネットワークノードが複数の Neutron ルーターを管理できるようになります。

Open vSwitch を用いると、仮想ネットワークを作成するために、2 つの異なる技術 GRE と VLAN を使用できます。

Generic Routing Encapsulation (GRE) は多くの VLAN で使用される技術です。異なるルーティング情報を用いて新しいパケット全体を作成するために IP パケットをラップできます。新しいパケットがその宛て先に到達したとき、ラップが外され、元のパケットが

中継されます。Open vSwitch を用いて GRE を使用するために、Neutron が GRE トンネルを作成します。これらのトンネルはブリッジにポートを作成し、異なるシステムにあるブリッジが 1 つのブリッジのように動作できるようにします。これにより、コンピュートノードとネットワークノードがルーティング目的に 1 つのものとして動作できます。

一方、Virtual LAN (VLAN) はイーサネットヘッダーに対する特別な変更を使用します。1 から 4096 までの範囲で 4 バイトの VLAN タグを追加します (タグ 0 は特別です。すべてのものからなるタグ 4095 はタグなしパケットにあたります)。特別な NIC、スイッチ、ルーターは Open vSwitch が実行するように、VLAN タグを解釈する方法について理解しています。1 つの VLAN にタグ付けされたパケットは、すべてのデバイスが同じ物理ネットワークにあるときさえ、その VLAN 上に設定された他のデバイスのみと共有されます。

Open vSwitch とともに使用される最も一般的なセキュリティグループはハイブリッド iptables/Open vSwitch プラグインです。これは iptables ルールと OpenFlow ルールの組み合わせを使用します。Linux でファイアウォールを作成し、NAT をセットアップするために iptables ツールを使用します。このツールは、Neutron セキュリティグループにより必要となる複雑なルールを実現するために、複雑なルールシステムとルールのチェインを使用します。

Modular Layer 2 (ML2) プラグイン

注記

We primarily tested the Modular Layer 2 (ML2) plug-in on Icehouse and suggest that you implement it instead of the traditional Open vSwitch (OVS) plug-in.

コントローラーノードの設定

前提

Before you configure Networking, you must create a database and Identity service credentials including a user and service.

1. Connect to the database as the root user, create the neutron database, and grant the proper access to it:

   Replace NEUTRON_DBPASS with a suitable password.

   ```
   $ mysql -u root -p
   mysql> CREATE DATABASE neutron;
   mysql> GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'localhost' ¥
   IDENTIFIED BY 'NEUTRON_DBPASS';
   mysql> GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'%' ¥
   IDENTIFIED BY 'NEUTRON_DBPASS';
   ```

2. Create Identity service credentials for Networking:

   a. Create the neutron user:

      Replace NEUTRON_PASS with a suitable password and neutron@example.com with a suitable e-mail address.

```
$ keystone user-create --name neutron --pass NEUTRON_PASS --email neutron@example.com
```

b. Link the neutron user to the service tenant and admin role:

```
$ keystone user-role-add --user neutron --tenant service --role admin
```

c. Create the neutron service:

```
$ keystone service-create --name neutron --type network --description "OpenStack Networking"
```

d. サービスエンドポイントを作成します。

```
$ keystone endpoint-create ¥
  --service-id $(keystone service-list | awk '/ network / {print $2}') ¥
  --publicurl http://controller:9696 ¥
  --adminurl http://controller:9696 ¥
  --internalurl http://controller:9696
```

To install the Networking components

- `# yum install openstack-neutron openstack-neutron-ml2`

To configure the Networking server component

The Networking server component configuration includes the database, authentication mechanism, message broker, topology change notifier, and plug-in.

1. Configure Networking to use the database:

   Replace NEUTRON_DBPASS with a suitable password.

```
# openstack-config --set /etc/neutron/neutron.conf database connection ¥
  mysql://neutron:NEUTRON_DBPASS@controller/neutron
```

2. Configure Networking to use the Identity service for authentication:

   Replace NEUTRON_PASS with the password you chose for the neutron user in the Identity service.

```
# openstack-config --set /etc/neutron/neutron.conf DEFAULT ¥
  auth_strategy keystone
# openstack-config --set /etc/neutron/neutron.conf keystone_authtoken ¥
  auth_uri http://controller:5000
# openstack-config --set /etc/neutron/neutron.conf keystone_authtoken ¥
  auth_host controller
# openstack-config --set /etc/neutron/neutron.conf keystone_authtoken ¥
  auth_protocol http
# openstack-config --set /etc/neutron/neutron.conf keystone_authtoken ¥
  auth_port 35357
# openstack-config --set /etc/neutron/neutron.conf keystone_authtoken ¥
  admin_tenant_name service
# openstack-config --set /etc/neutron/neutron.conf keystone_authtoken ¥
  admin_user neutron
# openstack-config --set /etc/neutron/neutron.conf keystone_authtoken ¥
  admin_password NEUTRON_PASS
```

3. Configure Networking to use the message broker:

```
# openstack-config --set /etc/neutron/neutron.conf DEFAULT ¥
  rpc_backend neutron.openstack.common.rpc.impl_qpid
# openstack-config --set /etc/neutron/neutron.conf DEFAULT ¥
  qpid_hostname controller
```

4. Configure Networking to notify Compute about network topology changes:

```
# openstack-config --set /etc/neutron/neutron.conf DEFAULT ¥
  notify_nova_on_port_status_changes True
# openstack-config --set /etc/neutron/neutron.conf DEFAULT ¥
  notify_nova_on_port_data_changes True
# openstack-config --set /etc/neutron/neutron.conf DEFAULT ¥
  nova_url http://controller:8774/v2
# openstack-config --set /etc/neutron/neutron.conf DEFAULT ¥
  nova_admin_username nova
# openstack-config --set /etc/neutron/neutron.conf DEFAULT ¥
  nova_admin_tenant_id $(keystone tenant-list | awk '/ service / { print $2 }')
# openstack-config --set /etc/neutron/neutron.conf DEFAULT ¥
  nova_admin_password NOVA_PASS
# openstack-config --set /etc/neutron/neutron.conf DEFAULT ¥
  nova_admin_auth_url http://controller:35357/v2.0
```

5. Configure Networking to use the Modular Layer 2 (ML2) plug-in and associated services:

```
# openstack-config --set /etc/neutron/neutron.conf DEFAULT ¥
  core_plugin ml2
# openstack-config --set /etc/neutron/neutron.conf DEFAULT ¥
  service_plugins router
```

注記

We recommend adding verbose = True to the [DEFAULT] section in /etc/neutron/neutron.conf to assist with troubleshooting.

6. Comment out any lines in the [service_providers] section.

To configure the Modular Layer 2 (ML2) plug-in

The ML2 plug-in uses the Open vSwitch (OVS) mechanism (agent) to build the virtual networking framework for instances. However, the controller node does not need the OVS agent or service because it does not handle instance network traffic.

- 以下のコマンドを実行します。

```
# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini ml2 ¥
  type_drivers gre
# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini ml2 ¥
  tenant_network_types gre
# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini ml2 ¥
  mechanism_drivers openvswitch
# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini ml2_type_gre ¥
  tunnel_id_ranges 1:1000
# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini securitygroup ¥
  firewall_driver neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver
# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini securitygroup ¥
```

```
enable_security_group True
```

To configure Compute to use Networking

By default, most distributions configure Compute to use legacy networking. You must reconfigure Compute to manage networks through OpenStack Networking.

- 以下のコマンドを実行します。

  Replace NEUTRON_PASS with the password you chose for the neutron user in the Identity service.

```
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  network_api_class nova.network.neutronv2.api.API
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  neutron_url http://controller:9696
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  neutron_auth_strategy keystone
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  neutron_admin_tenant_name service
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  neutron_admin_username neutron
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  neutron_admin_password NEUTRON_PASS
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  neutron_admin_auth_url http://controller:35357/v2.0
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  linuxnet_interface_driver nova.network.linux_net.LinuxOVSInterfaceDriver
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  firewall_driver nova.virt.firewall.NoopFirewallDriver
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  security_group_api neutron
```

注記

By default, Compute uses an internal firewall service. Since Networking includes a firewall service, you must disable the Compute firewall service by using the nova.virt.firewall.NoopFirewallDriver firewall driver.

To finalize installation

1. The Networking service initialization scripts expect a symbolic link /etc/neutron/plugin.ini pointing to the configuration file associated with your chosen plug-in. Using ML2, for example, the symbolic link must point to /etc/neutron/plugins/ml2/ml2_conf.ini. If this symbolic link does not exist, create it using the following commands:

```
# ln -s plugins/ml2/ml2_conf.ini /etc/neutron/plugin.ini
```

2. Compute のサービスを再起動します。

```
# service openstack-nova-api restart
# service openstack-nova-scheduler restart
# service openstack-nova-conductor restart
```

3. Networking サービスを起動し、システム起動時に起動するよう設定します。

```
# service neutron-server start
# chkconfig neutron-server on
```

ネットワークノードの設定

前提

Before you configure Networking, you must enable certain kernel networking functions.

1. /etc/sysctl.conf を編集し、以下の内容を含めます。

```
net.ipv4.ip_forward=1
net.ipv4.conf.all.rp_filter=0
net.ipv4.conf.default.rp_filter=0
```

2. 変更を実装します。

```
# sysctl -p
```

To install the Networking components

-
```
# yum install openstack-neutron openstack-neutron-ml2 ¥
  openstack-neutron-openvswitch python-novaclient
```

To configure the Networking common components

The Networking common component configuration includes the authentication mechanism, message broker, and plug-in.

1. Configure Networking to use the Identity service for authentication:

   Replace NEUTRON_PASS with the password you chose for the neutron user in the Identity service.

```
# openstack-config --set /etc/neutron/neutron.conf DEFAULT ¥
  auth_strategy keystone
# openstack-config --set /etc/neutron/neutron.conf keystone_authtoken ¥
  auth_uri http://controller:5000
# openstack-config --set /etc/neutron/neutron.conf keystone_authtoken ¥
  auth_host controller
# openstack-config --set /etc/neutron/neutron.conf keystone_authtoken ¥
  auth_protocol http
# openstack-config --set /etc/neutron/neutron.conf keystone_authtoken ¥
  auth_port 35357
# openstack-config --set /etc/neutron/neutron.conf keystone_authtoken ¥
  admin_tenant_name service
# openstack-config --set /etc/neutron/neutron.conf keystone_authtoken ¥
  admin_user neutron
# openstack-config --set /etc/neutron/neutron.conf keystone_authtoken ¥
  admin_password NEUTRON_PASS
```

2. Configure Networking to use the message broker:

```
# openstack-config --set /etc/neutron/neutron.conf DEFAULT ¥
  rpc_backend neutron.openstack.common.rpc.impl_qpid
# openstack-config --set /etc/neutron/neutron.conf DEFAULT ¥
  qpid_hostname controller
```

3. Configure Networking to use the Modular Layer 2 (ML2) plug-in and associated services:

```
# openstack-config --set /etc/neutron/neutron.conf DEFAULT ¥
  core_plugin ml2
# openstack-config --set /etc/neutron/neutron.conf DEFAULT ¥
  service_plugins router
```

注記

We recommend adding verbose = True to the [DEFAULT] section in /etc/neutron/neutron.conf to assist with troubleshooting.

4. Comment out any lines in the [service_providers] section.

To configure the Layer-3 (L3) agent

The Layer-3 (L3) agent provides routing services for instance virtual networks.

- 以下のコマンドを実行します。

```
# openstack-config --set /etc/neutron/l3_agent.ini DEFAULT ¥
  interface_driver neutron.agent.linux.interface.OVSInterfaceDriver
# openstack-config --set /etc/neutron/l3_agent.ini DEFAULT ¥
  use_namespaces True
```

注記

We recommend adding verbose = True to the [DEFAULT] section in /etc/neutron/l3_agent.ini to assist with troubleshooting.

To configure the DHCP agent

The DHCP agent provides DHCP services for instance virtual networks.

- 以下のコマンドを実行します。

```
# openstack-config --set /etc/neutron/dhcp_agent.ini DEFAULT ¥
  interface_driver neutron.agent.linux.interface.OVSInterfaceDriver
# openstack-config --set /etc/neutron/dhcp_agent.ini DEFAULT ¥
  dhcp_driver neutron.agent.linux.dhcp.Dnsmasq
# openstack-config --set /etc/neutron/dhcp_agent.ini DEFAULT ¥
  use_namespaces True
```

注記

We recommend adding verbose = True to the [DEFAULT] section in /etc/neutron/dhcp_agent.ini to assist with troubleshooting.

To configure the metadata agent

The metadata agent provides configuration information such as credentials for remote access to instances.

1. 以下のコマンドを実行します。

Replace NEUTRON_PASS with the password you chose for the neutron user in the Identity service. Replace METADATA_SECRET with a suitable secret for the metadata proxy.

```
# openstack-config --set /etc/neutron/metadata_agent.ini DEFAULT ¥
  auth_url http://controller:5000/v2.0
# openstack-config --set /etc/neutron/metadata_agent.ini DEFAULT ¥
  auth_region regionOne
# openstack-config --set /etc/neutron/metadata_agent.ini DEFAULT ¥
  admin_tenant_name service
# openstack-config --set /etc/neutron/metadata_agent.ini DEFAULT ¥
  admin_user neutron
# openstack-config --set /etc/neutron/metadata_agent.ini DEFAULT ¥
  admin_password NEUTRON_PASS
# openstack-config --set /etc/neutron/metadata_agent.ini DEFAULT ¥
  nova_metadata_ip controller
# openstack-config --set /etc/neutron/metadata_agent.ini DEFAULT ¥
  metadata_proxy_shared_secret METADATA_SECRET
```

注記

We recommend adding verbose = True to the [DEFAULT] section in /etc/neutron/metadata_agent.ini to assist with troubleshooting.

2. 

注記

Perform the next two steps on the controller node.

3. On the controller node, configure Compute to use the metadata service:

   Replace METADATA_SECRET with the secret you chose for the metadata proxy.

```
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  service_neutron_metadata_proxy true
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  neutron_metadata_proxy_shared_secret METADATA_SECRET
```

4. On the controller node, restart the Compute API service:

```
# service openstack-nova-api restart
```

To configure the Modular Layer 2 (ML2) plug-in

The ML2 plug-in uses the Open vSwitch (OVS) mechanism (agent) to build virtual networking framework for instances.

- 以下のコマンドを実行します。

  Replace INSTANCE_TUNNELS_INTERFACE_IP_ADDRESS with the IP address of the instance tunnels network interface on your network node. This guide uses 10.0.1.21 for the IP address of the instance tunnels network interface on the network node.

```
# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini ml2 ¥
  type_drivers gre
# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini ml2 ¥
  tenant_network_types gre
```

```
# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini ml2 ¥
  mechanism_drivers openvswitch
# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini ml2_type_gre ¥
  tunnel_id_ranges 1:1000
# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini ovs ¥
  local_ip INSTANCE_TUNNELS_INTERFACE_IP_ADDRESS
# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini ovs ¥
  tunnel_type gre
# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini ovs ¥
  enable_tunneling True
# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini securitygroup ¥
  firewall_driver neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver
# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini securitygroup ¥
  enable_security_group True
```

To configure the Open vSwitch (OVS) service

The OVS service provides the underlying virtual networking framework for instances. The integration bridge br-int handles internal instance network traffic within OVS. The external bridge br-ext handles external instance network traffic within OVS. The external bridge requires a port on the physical external network interface to provide instances with external network access. In essence, this port bridges the virtual and physical external networks in your environment.

1. Start the OVS service and configure it to start when the system boots:

```
# service openvswitch start
# chkconfig openvswitch on
```

2. 統合ブリッジを追加します。

```
# ovs-vsctl add-br br-int
```

3. 外部ブリッジを追加します。

```
# ovs-vsctl add-br br-ex
```

4. Add a port to the external bridge that connects to the physical external network interface (eth2):

```
# ovs-vsctl add-port br-ex eth2
```

注記

Depending on your network interface driver, you may need to disable Generic Receive Offload (GRO) to achieve suitable throughput between your instances and the external network.

To temporarily disable GRO on the external network interface while testing your environment:

```
# ethtool -K eth2 gro off
```

To finalize the installation

1. The Networking service initialization scripts expect a symbolic link /etc/neutron/plugin.ini pointing to the configuration file associated with your

chosen plug-in. Using the ML2 plug-in, for example, the symbolic link must point to /etc/neutron/plugins/ml2/ml2_conf.ini. If this symbolic link does not exist, create it using the following commands:

```
# ln -s plugins/ml2/ml2_conf.ini /etc/neutron/plugin.ini
```

Due to a packaging bug, the Open vSwitch agent initialization script explicitly looks for the Open vSwitch plug-in configuration file rather than a symbolic link /etc/neutron/plugin.ini pointing to the ML2 plug-in configuration file. Run the following commands to resolve this issue:

```
# cp /etc/init.d/neutron-openvswitch-agent /etc/init.d/neutron-openvswitch-agent.orig
# sed -i 's,plugins/openvswitch/ovs_neutron_plugin.ini,plugin.ini,g' /etc/init.d/neutron-openvswitch-agent
```

2. Start the Networking services and configure them to start when the system boots:

```
# service neutron-openvswitch-agent start
# service neutron-l3-agent start
# service neutron-dhcp-agent start
# service neutron-metadata-agent start
# chkconfig neutron-openvswitch-agent on
# chkconfig neutron-l3-agent on
# chkconfig neutron-dhcp-agent on
# chkconfig neutron-metadata-agent on
```

コンピュートノードの設定

前提

Before you configure Networking, you must enable certain kernel networking functions.

1. /etc/sysctl.conf を編集し、以下の内容を含めます。

```
net.ipv4.conf.all.rp_filter=0
net.ipv4.conf.default.rp_filter=0
```

2. 変更を実装します。

```
# sysctl -p
```

To install the Networking components

-
```
# yum install openstack-neutron-ml2 openstack-neutron-openvswitch
```

To configure the Networking common components

The Networking common component configuration includes the authentication mechanism, message broker, and plug-in.

1. Configure Networking to use the Identity service for authentication:

   Replace NEUTRON_PASS with the password you chose for the neutron user in the Identity service.

```
# openstack-config --set /etc/neutron/neutron.conf DEFAULT ¥
  auth_strategy keystone
# openstack-config --set /etc/neutron/neutron.conf keystone_authtoken ¥
  auth_uri http://controller:5000
# openstack-config --set /etc/neutron/neutron.conf keystone_authtoken ¥
  auth_host controller
# openstack-config --set /etc/neutron/neutron.conf keystone_authtoken ¥
  auth_protocol http
# openstack-config --set /etc/neutron/neutron.conf keystone_authtoken ¥
  auth_port 35357
# openstack-config --set /etc/neutron/neutron.conf keystone_authtoken ¥
  admin_tenant_name service
# openstack-config --set /etc/neutron/neutron.conf keystone_authtoken ¥
  admin_user neutron
# openstack-config --set /etc/neutron/neutron.conf keystone_authtoken ¥
  admin_password NEUTRON_PASS
```

2. Configure Networking to use the message broker:

```
# openstack-config --set /etc/neutron/neutron.conf DEFAULT ¥
  rpc_backend neutron.openstack.common.rpc.impl_qpid
# openstack-config --set /etc/neutron/neutron.conf DEFAULT ¥
  qpid_hostname controller
```

3. Configure Networking to use the Modular Layer 2 (ML2) plug-in and associated services:

```
# openstack-config --set /etc/neutron/neutron.conf DEFAULT ¥
  core_plugin ml2
# openstack-config --set /etc/neutron/neutron.conf DEFAULT ¥
  service_plugins router
```

注記

We recommend adding verbose = True to the [DEFAULT] section in /etc/neutron/neutron.conf to assist with troubleshooting.

4. Comment out any lines in the [service_providers] section.

To configure the Modular Layer 2 (ML2) plug-in

The ML2 plug-in uses the Open vSwitch (OVS) mechanism (agent) to build the virtual networking framework for instances.

- 以下のコマンドを実行します。

  Replace INSTANCE_TUNNELS_INTERFACE_IP_ADDRESS with the IP address of the instance tunnels network interface on your compute node. This guide uses 10.0.1.31 for the IP address of the instance tunnels network interface on the first compute node.

```
# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini ml2 ¥
  type_drivers gre
# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini ml2 ¥
  tenant_network_types gre
# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini ml2 ¥
```

```
  mechanism_drivers openvswitch
# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini ml2_type_gre ¥
  tunnel_id_ranges 1:1000
# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini ovs ¥
  local_ip INSTANCE_TUNNELS_INTERFACE_IP_ADDRESS
# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini ovs ¥
  tunnel_type gre
# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini ovs ¥
  enable_tunneling True
# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini securitygroup ¥
  firewall_driver neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver
# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini securitygroup ¥
  enable_security_group True
```

To configure the Open vSwitch (OVS) service

The OVS service provides the underlying virtual networking framework for instances. The integration bridge br-int handles internal instance network traffic within OVS.

1. Start the OVS service and configure it to start when the system boots:

```
# service openvswitch start
# chkconfig openvswitch on
```

2. 統合ブリッジを追加します。

```
# ovs-vsctl add-br br-int
```

To configure Compute to use Networking

By default, most distributions configure Compute to use legacy networking. You must reconfigure Compute to manage networks through OpenStack Networking.

- 以下のコマンドを実行します。

  Replace NEUTRON_PASS with the password you chose for the neutron user in the Identity service.

```
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  network_api_class nova.network.neutronv2.api.API
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  neutron_url http://controller:9696
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  neutron_auth_strategy keystone
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  neutron_admin_tenant_name service
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  neutron_admin_username neutron
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  neutron_admin_password NEUTRON_PASS
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  neutron_admin_auth_url http://controller:35357/v2.0
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  linuxnet_interface_driver nova.network.linux_net.LinuxOVSInterfaceDriver
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  firewall_driver nova.virt.firewall.NoopFirewallDriver
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
```

```
security_group_api neutron
```

注記

By default, Compute uses an internal firewall service. Since Networking includes a firewall service, you must disable the Compute firewall service by using the nova.virt.firewall.NoopFirewallDriver firewall driver.

To finalize the installation

1. The Networking service initialization scripts expect a symbolic link /etc/neutron/plugin.ini pointing to the configuration file associated with your chosen plug-in. Using the ML2 plug-in, for example, the symbolic link must point to /etc/neutron/plugins/ml2/ml2_conf.ini. If this symbolic link does not exist, create it using the following commands:

   ```
   # ln -s plugins/ml2/ml2_conf.ini /etc/neutron/plugin.ini
   ```

   Due to a packaging bug, the Open vSwitch agent initialization script explicitly looks for the Open vSwitch plug-in configuration file rather than a symbolic link /etc/neutron/plugin.ini pointing to the ML2 plug-in configuration file. Run the following commands to resolve this issue:

   ```
   # cp /etc/init.d/neutron-openvswitch-agent /etc/init.d/neutron-openvswitch-agent.orig
   # sed -i 's,plugins/openvswitch/ovs_neutron_plugin.ini,plugin.ini,g' /etc/init.d/neutron-openvswitch-agent
   ```

2. Compute Service を再起動します。

   ```
   # service openstack-nova-compute restart
   ```

3. Start the Open vSwitch (OVS) agent and configure it to start when the system boots:

   ```
   # service neutron-openvswitch-agent start
   # chkconfig neutron-openvswitch-agent on
   ```

Open vSwitch (OVS) プラグイン

警告

We suggest that you implement the Modular Layer 2 (ML2) plug-in on Icehouse until we completely test the traditional Open vSwitch (OVS) plug-in.

コントローラーノードの設定

警告

system-config-firewall 自動ファイアウォール設定ツールが RHEL にデフォルトで入っています。このグラフィカルツール（および名前の最後に -tui を付けた端末スタイルのインターフェース）により、基本的なファイ

アウォールとして iptables を設定できます。基礎となるネットワーク技術に詳しくなければ、Neutron を利用しているときに、これを無効化すべきです。これは Neutron にとって重要であるさまざまな種類のネットワーク通信を遮断するためです。これを無効化するには、単にプログラムを起動し、有効化チェックボックスを解除します。

OpenStack を Neutron と一緒に正常にセットアップした後、ツールを再び有効化し、設定できます。しかしながら、Networking のセットアップ中は、ネットワークの問題をデバッグしやすくするためにツールを無効化します。

前提

個々のノードを Networking 用に設定する前に、必要となる OpenStack コンポーネント（ユーザー、サービス、データベース、1 つ以上のエンドポイント）を作成する必要があります。コントローラーノードでこれらの手順を完了した後、OpenStack Networking ノードをセットアップするために、このガイドにある説明に従います。

1. MySQL データベースに root ユーザーとして接続し、neutron データベースを作成し、適切なアクセス権限を与えます。

```
$ mysql -u root -p
mysql> CREATE DATABASE neutron;
mysql> GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'localhost' ¥
IDENTIFIED BY 'NEUTRON_DBPASS';
mysql> GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'%' ¥
IDENTIFIED BY 'NEUTRON_DBPASS';
```

2. Networking が Identity Service と通信できるよう、必要となるユーザー、サービス、エンドポイントを作成します。

   neutron ユーザーを作成します。

```
$ keystone user-create --name=neutron --pass=NEUTRON_PASS --email=neutron@example.com
```

   ユーザーロールを neutron ユーザーに追加します。

```
$ keystone user-role-add --user=neutron --tenant=service --role=admin
```

   neutron サービスを作成します。

```
$ keystone service-create --name=neutron --type=network ¥
  --description="OpenStack Networking"
```

   Networking エンドポイントを作成します。

```
$ keystone endpoint-create ¥
  --service-id $(keystone service-list | awk '/ network / {print $2}') ¥
  --publicurl http://controller:9696 ¥
  --adminurl http://controller:9696 ¥
  --internalurl http://controller:9696
```

サーバーコンポーネントのインストールと設定

1. Networking および依存関係のあるサーバーコンポーネントをインストールします。

```
# yum install openstack-neutron python-neutron python-neutronclient
```

2. Networking がデータベースに接続するよう設定します。

```
# openstack-config --set /etc/neutron/neutron.conf database connection ¥
  mysql://neutron:NEUTRON_DBPASS@controller/neutron
```

3. Networking が認証用に Identity Service として keystone を使用するよう設定します。

   a. このファイルの DEFAULT セクションで auth_strategy 設定キーを keystone に設定します。

```
# openstack-config --set /etc/neutron/neutron.conf DEFAULT auth_strategy keystone
```

   b. keystone 認証用に neutron を設定します。

```
# openstack-config --set /etc/neutron/neutron.conf keystone_authtoken ¥
  auth_uri http://controller:5000
# openstack-config --set /etc/neutron/neutron.conf keystone_authtoken ¥
  auth_host controller
# openstack-config --set /etc/neutron/neutron.conf keystone_authtoken ¥
  auth_protocol http
# openstack-config --set /etc/neutron/neutron.conf keystone_authtoken ¥
  auth_port 35357
# openstack-config --set /etc/neutron/neutron.conf keystone_authtoken ¥
  admin_tenant_name service
# openstack-config --set /etc/neutron/neutron.conf keystone_authtoken ¥
  admin_user neutron
# openstack-config --set /etc/neutron/neutron.conf keystone_authtoken ¥
  admin_password NEUTRON_PASS
```

4. Qpid メッセージキューのアクセス権を設定します。

```
# openstack-config --set /etc/neutron/neutron.conf DEFAULT ¥
  rpc_backend neutron.openstack.common.rpc.impl_qpid
# openstack-config --set /etc/neutron/neutron.conf DEFAULT ¥
  qpid_hostname controller
# openstack-config --set /etc/neutron/neutron.conf DEFAULT ¥
  qpid_port 5672
# openstack-config --set /etc/neutron/neutron.conf DEFAULT ¥
  qpid_username guest
# openstack-config --set /etc/neutron/neutron.conf DEFAULT ¥
  qpid_password guest
```

Open vSwitch (OVS) プラグインのインストールと設定

OpenStack Networking はさまざまなプラグインをサポートします。簡単のために、最も一般的なプラグイン Open vSwitch を取り扱うことにし、プロジェクトのネットワーク通信のために基本的な GRE トンネルを使用するよう設定します。

1. Open vSwitch プラグインをインストールします。

```
# yum install openstack-neutron-openvswitch
```

2. どのネットワーク技術を Open vSwitch と一緒に使用するかによらず、いくつかの共通設定オプションを設定する必要があります。OVS を使用するために Networking コアを設定する必要があります。/etc/neutron/neutron.conf ファイルを編集します。

```
core_plugin = neutron.plugins.openvswitch.ovs_neutron_plugin.OVSNeutronPluginV2
```

注記

コントローラー専用ノードは Open vSwitch や Open vSwitch エージェントを実行する必要がありません。

3. OVS プラグインに GRE トンネリングを使用するよう設定します。/etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini ファイルを編集します。

```
[ovs]
tenant_network_type = gre
tunnel_id_ranges = 1:1000
enable_tunneling = True
```

Compute サービスの Networking 用設定

1. Compute が OpenStack Networking サービスを使用するよう設定します。以下のそれぞれの説明にあるとおり、/etc/nova/nova.conf ファイルを設定します。

```
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  network_api_class nova.network.neutronv2.api.API
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  neutron_url http://controller:9696
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  neutron_auth_strategy keystone
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  neutron_admin_tenant_name service
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  neutron_admin_username neutron
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  neutron_admin_password NEUTRON_PASS
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  neutron_admin_auth_url http://controller:35357/v2.0
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  linuxnet_interface_driver nova.network.linux_net.LinuxOVSInterfaceDriver
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  firewall_driver nova.virt.firewall.NoopFirewallDriver
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  security_group_api neutron
```

注記

- ネットワークノードとコンピュートノードを設定するときに、どのファイアウォールドライバーを選択しても、このドライバーを No-Op ファイアウォールとして設定します。このファイアウォールは nova ファイアウォールです。neutron がファイアウォールを取り扱うので、nova にこれを使用しないよう通知する必要があります。

  Networking がファイアウォールを取り扱うとき、firewall_driver オプションは指定したプラグインに合わせて設定されるべきです。例えば、OVS を用いる場合、/etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini ファイルを編集します。

```
# openstack-config --set ¥
   /etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini securitygroup
 firewall_driver ¥
   neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver
```

- If you do not want to use a firewall in Compute or Networking, set firewall_driver=nova.virt.firewall.NoopFirewallDriver in both config files, and comment out or remove security_group_api=neutron in the /etc/nova/nova.conf file, otherwise you may encounter ERROR: The server has either erred or is incapable of performing the requested operation. (HTTP 500) when issuing nova list commands.

2. neutron-server 初期化スクリプトは、選択したプラグインと関連する設定ファイルを指し示すシンボリックリンク /etc/neutron/plugin.ini を予期しています。例えば、Open vSwitch を使用する場合、シンボリックリンクが /etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini を指し示す必要があります。このシンボリックリンクが存在しなければ、以下のコマンドを使用して作成します。

```
# cd /etc/neutron
# ln -s plugins/openvswitch/ovs_neutron_plugin.ini plugin.ini
```

インストールの完了

1. Compute のサービスを再起動します。

```
# service openstack-nova-api restart
# service openstack-nova-scheduler restart
# service openstack-nova-conductor restart
```

2. Networking サービスを起動し、システム起動時に起動するよう設定します。

```
# service neutron-server start
# chkconfig neutron-server on
```

ネットワークノードの設定

注記

Before you start, set up a machine as a dedicated network node. Dedicated network nodes have a MGMT_INTERFACE NIC, a DATA_INTERFACE NIC, and an EXTERNAL_INTERFACE NIC.

管理ネットワークはノード間の通信を処理します。データネットワークは仮想マシンとの通信を処理します。外部 NIC は仮想マシンが外部と接続できるようネットワークノードを接続します。オプションとしてコントローラーノードに接続します。

警告

By default, the system-config-firewall automated firewall configuration tool is in place on RHEL. This graphical interface (and a curses-style interface with -tui on the end of the name) enables you to configure IP tables as a basic firewall. You should disable it when you work with Networking unless you are familiar with the underlying network technologies. By default, it blocks various types of network traffic that are important to Networking. To disable it, simply launch the program and clear the Enabled check box.

OpenStack Networking を正常にセットアップした後、ツールを再び有効化し、設定できます。しかしながら、Networking のセットアップ中は、ネットワークの問題をデバッグしやすくするためにツールを無効化します。

エージェントのインストールおよび共通コンポーネントの設定

1. Networking パッケージおよび依存関係のあるパッケージをインストールします。

```
# yum install openstack-neutron
```

2. Networking エージェントをブート時に起動するよう設定します。

```
# for s in neutron-{dhcp,metadata,l3}-agent; do chkconfig $s on; done
```

3. ネットワークノードが仮想マシンの通信を制御できるように、パケット転送を有効化し、パケット宛先フィルタリングを無効化します。/etc/sysctl.conf を以下のように編集します。

```
net.ipv4.ip_forward=1
net.ipv4.conf.all.rp_filter=0
net.ipv4.conf.default.rp_filter=0
```

Use the sysctl command to ensure the changes made to the /etc/sysctl.conf file take effect:

```
# sysctl -p
```

注記

Networking 関連の設定の値を変更した後、Networking Service を再起動することを推奨します。これにより、すべての変更した値がすぐに確実に適用されます。

```
# service network restart
```

4. Networking が認証用に keystone を使用するよう設定します。

   a. このファイルの DEFAULT セクションで auth_strategy 設定キーを keystone に設定します。

```
# openstack-config --set /etc/neutron/neutron.conf DEFAULT auth_strategy keystone
```

   b. keystone 認証用に neutron を設定します。

```
# openstack-config --set /etc/neutron/neutron.conf keystone_authtoken ¥
  auth_uri http://controller:5000
# openstack-config --set /etc/neutron/neutron.conf keystone_authtoken ¥
  auth_host controller
# openstack-config --set /etc/neutron/neutron.conf keystone_authtoken ¥
  auth_protocol http
# openstack-config --set /etc/neutron/neutron.conf keystone_authtoken ¥
  auth_port 35357
# openstack-config --set /etc/neutron/neutron.conf keystone_authtoken ¥
  admin_tenant_name service
# openstack-config --set /etc/neutron/neutron.conf keystone_authtoken ¥
  admin_user neutron
# openstack-config --set /etc/neutron/neutron.conf keystone_authtoken ¥
```

```
admin_password NEUTRON_PASS
```

5. Qpid メッセージキューのアクセス権を設定します。

```
# openstack-config --set /etc/neutron/neutron.conf DEFAULT ¥
  rpc_backend neutron.openstack.common.rpc.impl_qpid
# openstack-config --set /etc/neutron/neutron.conf DEFAULT ¥
  qpid_hostname controller
# openstack-config --set /etc/neutron/neutron.conf DEFAULT ¥
  qpid_port 5672
# openstack-config --set /etc/neutron/neutron.conf DEFAULT ¥
  qpid_username guest
# openstack-config --set /etc/neutron/neutron.conf DEFAULT ¥
  qpid_password guest
```

Install and configure the Open vSwitch (OVS) plug-in

OpenStack Networking はさまざまなプラグインをサポートします。簡単のために、最も一般的なプラグイン Open vSwitch を取り扱うことにし、プロジェクトのネットワーク通信のために基本的な GRE トンネルを使用するよう設定します。

1. Open vSwitch プラグインと依存パッケージをインストールします。

```
# yum install openstack-neutron-openvswitch
```

2. Open vSwitch を起動します。

```
# service openvswitch start
```

システム起動時に起動するよう設定します。

```
# chkconfig openvswitch on
```

3. どのネットワーク技術を使用するにしても、br-int 統合ブリッジを追加する必要があります。このブリッジは、仮想マシンと、外部に接続する br-ex 外部ブリッジを接続します。

```
# ovs-vsctl add-br br-int
# ovs-vsctl add-br br-ex
```

4. EXTERNAL_INTERFACE インターフェースの ポート（接続）を br-ex インターフェースに追加します。

```
# ovs-vsctl add-port br-ex EXTERNAL_INTERFACE
```

警告

ホストは EXTERNAL_INTERFACE 以外にインターフェースと関連づけられた IP アドレスを持つ必要があります。リモートターミナルセッションがこの他の IP アドレスと関連づけられる必要があります。

If you associate an IP address with EXTERNAL_INTERFACE, that IP address stops working after you issue the ovs-vsctl add-port br-ex EXTERNAL_INTERFACE command. If you associate a remote terminal session with that IP address, you lose connectivity with the host.

この動作に関する詳細は、Open vSwitch FAQ の Configuration Problems（接続の問題）を参照してください。

5. EXTERNAL_INTERFACE を IP アドレスなしでプロミスキャスモードに設定します。さらに、前に EXTERNAL_INTERFACE に含めた IP アドレスを持つよう、新しく作成した br-ex インターフェースを設定する必要があります。

   /etc/sysconfig/network-scripts/ifcfg-EXTERNAL_INTERFACE ファイルを作成します。

   ```
   DEVICE_INFO_HERE
   ONBOOT=yes
   BOOTPROTO=none
   PROMISC=yes
   ```

6. /etc/sysconfig/network-scripts/ifcfg-br-ex を作成し、編集します。

   ```
   DEVICE=br-ex
   TYPE=Bridge
   ONBOOT=no
   BOOTPROTO=none
   IPADDR=EXTERNAL_INTERFACE_IP
   NETMASK=EXTERNAL_INTERFACE_NETMASK
   GATEWAY=EXTERNAL_INTERFACE_GATEWAY
   ```

7. どのネットワーク技術を Open vSwitch と一緒に使用するかによらず、いくつかの共通設定オプションを設定する必要があります。OVS と名前空間を使用するために L3 エージェントと DHCP エージェントを設定する必要があります。それぞれ /etc/neutron/l3_agent.ini と /etc/neutron/dhcp_agent.ini ファイルを編集します。

   ```
   interface_driver = neutron.agent.linux.interface.OVSInterfaceDriver
   use_namespaces = True
   ```

   ### 注記

   このガイドの例はデフォルトでネットワークの名前空間を有効化するとしても、問題が発生したり、カーネルがそれらをサポートしなかったりする場合、それらを無効化できます。/etc/neutron/l3_agent.ini ファイルと /etc/neutron/dhcp_agent.ini ファイルをそれぞれ編集します。

   ```
   use_namespaces = False
   ```

   IP アドレスのオーバーラップを無効化するために /etc/neutron/neutron.conf を編集します。

   ```
   allow_overlapping_ips = False
   ```

   ネットワーク名前空間が無効化されているとき、各ネットワークノードに対してルーターを一つのみ持つことができ、IP アドレスのオーバーラップがサポートされないことに注意してください。

   初期 Neutron 仮想ネットワークとルーターを作成した後、追加のステップを完了する必要があります。

8. 同様に、OVS を使用するよう Neutron コアに通知する必要があります。/etc/neutron/neutron.conf ファイルを編集します。

```
core_plugin = neutron.plugins.openvswitch.ovs_neutron_plugin.OVSNeutronPluginV2
```

9. ファイアウォールプラグインを設定します。OpenStack によりセキュリティグループと呼ばれるファイアウォールルールを強制したくない場合、neutron.agent.firewall.NoopFirewall を使用できます。そうでなければ、Networking ファイアウォールプラグインのどれかを選択できます。最も一般的な選択は OVS-iptables ハイブリッドドライバーですが、FWaaS（ファイアウォールアズアサービス）ドライバーを使用することもできます。/etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini ファイルを編集します。

```
[securitygroup]
# Firewall driver for realizing neutron security group function.
firewall_driver = neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver
```

警告

少なくとも No-Op ファイアウォールを使用する必要があります。そうでなければ、Horizon と他の OpenStack サービスが必要となる仮想マシンのブートオプションを取得および設定できません。

10. OVS プラグインをシステム起動時に起動するよう設定します。

```
# chkconfig neutron-openvswitch-agent on
```

11. GRE トンネリング、br-int 統合ブリッジ、br-tun トンネリングブリッジ、DATA_INTERFACE トンネル IP 用ローカル IP を使用するよう OVS プラグインを設定します。/etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini ファイルを編集します。

```
[ovs]
...
tenant_network_type = gre
tunnel_id_ranges = 1:1000
enable_tunneling = True
integration_bridge = br-int
tunnel_bridge = br-tun
local_ip = DATA_INTERFACE_IP
```

エージェントの設定

1. SDN で DHCP を実行するために、Networking はいくつかのプラグインをサポートします。しかしながら一般的に、 dnsmasq プラグインを使用します。

   /etc/neutron/dhcp_agent.ini ファイルを設定します。

```

# openstack-config --set /etc/neutron/dhcp_agent.ini DEFAULT ¥
  dhcp_driver neutron.agent.linux.dhcp.Dnsmasq
```

2. 仮想マシンが Compute メタデータ情報にアクセスできるようにするために、Networking メタデータエージェントが有効化されて設定される必要があります。エージェントが Compute メタデータサービスのプロキシとして動作します。

   Compute サービスと Networking メタデータエージェントの間で共有される秘密鍵を定義するために、コントローラーで /etc/nova/nova.conf ファイルを編集します。

neutron_metadata_proxy_shared_secret キーを設定します。

```
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  neutron_metadata_proxy_shared_secret METADATA_PASS
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  service_neutron_metadata_proxy true
```

nova-api サービスを再起動します。

```
# service openstack-nova-api restart
```

ネットワークノードでメタデータエージェント設定を変更します。

必要となるキーを設定します。

```
# openstack-config --set /etc/neutron/metadata_agent.ini DEFAULT ¥
  auth_url http://controller:5000/v2.0
# openstack-config --set /etc/neutron/metadata_agent.ini DEFAULT ¥
  auth_region regionOne
# openstack-config --set /etc/neutron/metadata_agent.ini DEFAULT ¥
  admin_tenant_name service
# openstack-config --set /etc/neutron/metadata_agent.ini DEFAULT ¥
  admin_user neutron
# openstack-config --set /etc/neutron/metadata_agent.ini DEFAULT ¥
  admin_password NEUTRON_PASS
# openstack-config --set /etc/neutron/metadata_agent.ini DEFAULT ¥
  nova_metadata_ip controller
# openstack-config --set /etc/neutron/metadata_agent.ini DEFAULT ¥
  metadata_proxy_shared_secret METADATA_PASS
```

注記

auth_region の値は大文字小文字を区別します。Keystone で定義されたエンドポイントのリージョンと一致する必要があります。

注記

自己署名証明書を用いた HTTPS 経由で OpenStack Networking API を提供する場合、Networking がサービスカタログから SSL 証明書を検証できないため、メタデータエージェントに追加の設定をする必要があります。

必要となるキーを設定します。

```
# openstack-config --set /etc/neutron/metadata_agent.ini DEFAULT
 neutron_insecure True
```

インストールの完了

1. neutron-server 初期化スクリプトは、選択したプラグインと関連する設定ファイルを指し示すシンボリックリンク /etc/neutron/plugin.ini を予期しています。例えば、Open vSwitch を使用する場合、シンボリックリンクが /etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini を指し示す必要があります。このシンボリックリンクが存在しなければ、以下のコマンドを使用して作成します。

```
# cd /etc/neutron
```

```
# ln -s plugins/openvswitch/ovs_neutron_plugin.ini plugin.ini
```

2. Networking サービスを再起動します。

```
# service neutron-dhcp-agent restart
# service neutron-l3-agent restart
# service neutron-metadata-agent restart
# service neutron-openvswitch-agent restart
```

コンピュートノードの設定

注記

このセクションは nova-compute コンポーネントを実行するあらゆるノードのセットアップについて詳細に説明しますが、すべてのネットワークスタックを実行しません。

警告

system-config-firewall 自動ファイアウォール設定ツールが RHEL にデフォルトで入っています。このグラフィカルツール（および名前の最後に -tui を付けた端末スタイルのインターフェース）により、基本的なファイアウォールとして iptables を設定できます。基礎となるネットワーク技術に詳しくなければ、OpenStack Networking を利用しているときに、これを無効化すべきです。これは neutron サービスにとって重要であるさまざまな種類のネットワーク通信を遮断するためです。これを無効化する場合、プログラムを起動し、有効化チェックボックスを解除します。

Neutron を用いて OpenStack Networking を正常にセットアップした後、ツールを再び有効化し、設定できます。しかしながら、OpenStack Networking のセットアップ中は、ネットワークの問題をデバッグしやすくするためにツールを無効化します。

前提

- Networking Service が仮想マシンへの通信をルーティングできるように、パケット宛先フィルタリング（ルート検証）を無効化します。/etc/sysctl.conf を編集し、変更を有効化するために以下のコマンドを実行します。

```
net.ipv4.conf.all.rp_filter=0
net.ipv4.conf.default.rp_filter=0
```

```
# sysctl -p
```

Open vSwitch プラグインのインストール

OpenStack Networking はさまざまなプラグインをサポートします。簡単のために、最も一般的なプラグイン Open vSwitch を取り扱うことにし、プロジェクトのネットワーク通信のために基本的な GRE トンネルを使用するよう設定します。

1. Open vSwitch プラグインと依存パッケージをインストールします。

```
# yum install openstack-neutron-openvswitch
```

2. Open vSwitch を起動し、システム起動時に起動するよう設定します。

```
# service openvswitch start
# chkconfig openvswitch on
```

3. どのネットワーク技術を Open vSwitch と一緒に使用するかによらず、いくつかの共通設定オプションを設定する必要があります。仮想マシンを接続する br-int 統合ブリッジを追加する必要があります。

```
# ovs-vsctl add-br br-int
```

4. どのネットワーク技術を OVS と一緒に使用するかによらず、いくつかの共通設定オプションを設定する必要があります。/etc/neutron/neutron.conf ファイルを編集します。

```
core_plugin = neutron.plugins.openvswitch.ovs_neutron_plugin.OVSNeutronPluginV2
```

5. 同様にファイアウォールを設定する必要があります。ネットワークノードをセットアップするときに選択したものと同じファイアウォールプラグインを使用すべきです。そうするために、/etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini ファイルを編集し、securitygroup の下にある firewall_driver 値をネットワークノードで使用したものと同じ値に設定します。例えば、ハイブリッド OVS iptables プラグインを使用したい場合、設定はこのようになるでしょう。

```
[securitygroup]
# Firewall driver for realizing neutron security group function.
firewall_driver = neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver
```

警告

少なくとも No-Op ファイアウォールを使用する必要があります。そうでなければ、Horizon と他の OpenStack サービスが必要となる仮想マシンのブートオプションを取得および設定できません。

6. OVS プラグインをシステム起動時に起動するよう設定します。

```
# chkconfig neutron-openvswitch-agent on
```

7. br-int 統合ブリッジを持つ GRE トンネリング、br-tun トンネリングブリッジ、DATA_INTERFACE の IP のトンネル用ローカル IP を使用するよう OVS プラグインに通知します。/etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini ファイルを編集します。

```
[ovs]
...
tenant_network_type = gre
tunnel_id_ranges = 1:1000
enable_tunneling = True
integration_bridge = br-int
tunnel_bridge = br-tun
local_ip = DATA_INTERFACE_IP
```

共通コンポーネントの設定

1. Networking が認証用に keystone を使用するよう設定します。

   a. このファイルの [DEFAULT] セクションで auth_strategy 設定キーを keystone に設定します。

```
# openstack-config --set /etc/neutron/neutron.conf DEFAULT auth_strategy keystone
```

b. keystone 認証用に neutron を設定します。

```
# openstack-config --set /etc/neutron/neutron.conf keystone_authtoken ¥
  auth_uri http://controller:5000
# openstack-config --set /etc/neutron/neutron.conf keystone_authtoken ¥
  auth_host controller
# openstack-config --set /etc/neutron/neutron.conf keystone_authtoken ¥
  auth_protocol http
# openstack-config --set /etc/neutron/neutron.conf keystone_authtoken ¥
  auth_port 35357
# openstack-config --set /etc/neutron/neutron.conf keystone_authtoken ¥
  admin_tenant_name service
# openstack-config --set /etc/neutron/neutron.conf keystone_authtoken ¥
  admin_user neutron
# openstack-config --set /etc/neutron/neutron.conf keystone_authtoken ¥
  admin_password NEUTRON_PASS
```

2. Qpid メッセージキューのアクセス権を設定します。

```
# openstack-config --set /etc/neutron/neutron.conf DEFAULT ¥
  rpc_backend neutron.openstack.common.rpc.impl_qpid
# openstack-config --set /etc/neutron/neutron.conf DEFAULT ¥
  qpid_hostname controller
# openstack-config --set /etc/neutron/neutron.conf DEFAULT ¥
  qpid_port 5672
# openstack-config --set /etc/neutron/neutron.conf DEFAULT ¥
  qpid_username guest
# openstack-config --set /etc/neutron/neutron.conf DEFAULT ¥
  qpid_password guest
```

Compute サービスの Networking 用設定

- OpenStack Compute が OpenStack Networking Service を使用するよう設定します。以下のそれぞれの説明にあるとおり、/etc/nova/nova.conf ファイルを編集します。

```
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  network_api_class nova.network.neutronv2.api.API
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  neutron_url http://controller:9696
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  neutron_auth_strategy keystone
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  neutron_admin_tenant_name service
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  neutron_admin_username neutron
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  neutron_admin_password NEUTRON_PASS
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  neutron_admin_auth_url http://controller:35357/v2.0
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  linuxnet_interface_driver nova.network.linux_net.LinuxOVSInterfaceDriver
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  firewall_driver nova.virt.firewall.NoopFirewallDriver
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  security_group_api neutron
```

注記

- ネットワークとコンピュートノードを設定するときに、どのファイアウォールドライバーを選択しても、ファイアウォールドライバーを nova.virt.firewall.NoopFirewallDriver に設定するために、/etc/nova/nova.conf ファイルを編集する必要があります。OpenStack Networking はファイアウォールを取り扱うので、このステートメントは Compute がファイアウォールを使用しないことを指定します。
- Networking にファイアウォールを取り扱わせたい場合、firewall_driver オプションをプラグイン用のファイアウォールに設定するために、/etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini ファイルを編集します。例えば、OVS を使用する場合、ファイルを次のとおり編集します。

```
# openstack-config --set ¥
  /etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini securitygroup
 firewall_driver ¥
  neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver
```

- Compute や Networking でファイアウォールを使用したくない場合、両方の設定ファイルを編集し、firewall_driver=nova.virt.firewall.NoopFirewallDriver を設定します。また、/etc/nova/nova.conf ファイルを編集し、security_group_api=neutron ステートメントをコメントアウトまたは削除します。

  Otherwise, when you issue nova list commands, the ERROR: The server has either erred or is incapable of performing the requested operation. (HTTP 500) error might be returned.

インストールの完了

1. neutron-server 初期化スクリプトは、選択したプラグインと関連する設定ファイルを指し示すシンボリックリンク /etc/neutron/plugin.ini を予期しています。例えば、Open vSwitch を使用する場合、シンボリックリンクが /etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini を指し示す必要があります。このシンボリックリンクが存在しなければ、以下のコマンドを使用して作成します。

```
# cd /etc/neutron
# ln -s plugins/openvswitch/ovs_neutron_plugin.ini plugin.ini
```

2. Networking サービスを再起動します。

```
# service neutron-openvswitch-agent restart
```

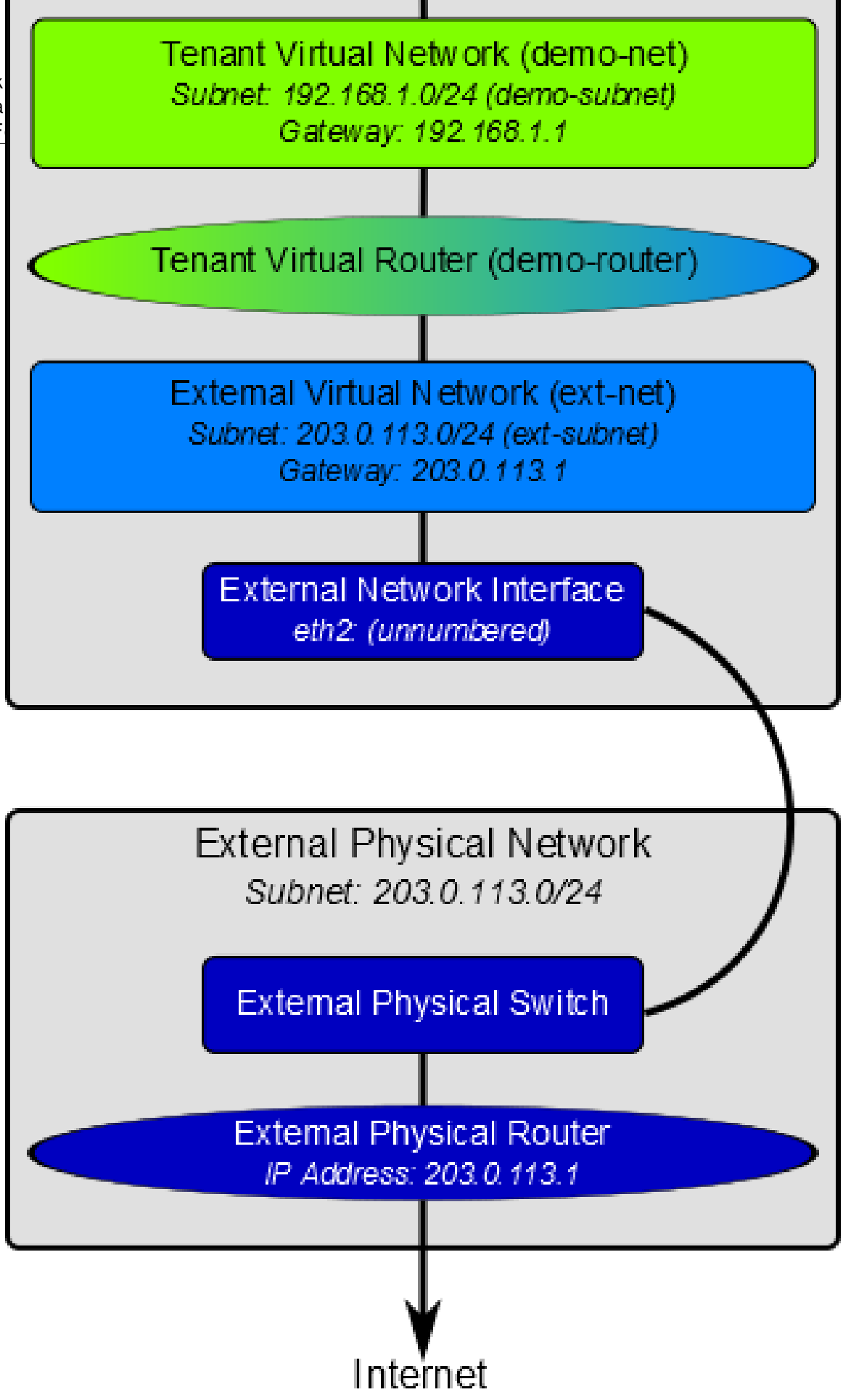
3. Compute Service を再起動します。

```
# service openstack-nova-compute restart
```

初期ネットワークの作成

Before launching your first instance, you must create the necessary virtual network infrastructure to which the instance will connect, including the

external network and tenant network. See 図7.1「初期ネットワーク」 [77]. After creating this infrastructure, we recommend that you verify connectivity and resolve any issues before proceeding further.


Tenant Virtual Network (demo-net)
Subnet: 192.168.1.0/24 (demo-subnet)
Gateway: 192.168.1.1
Tenant Virtual Router (demo-router)
External Virtual Network (ext-net)
Subnet: 203.0.113.0/24 (ext-subnet)
Gateway: 203.0.113.1
External Network Interface
eth2: (unnumbered)
External Physical Network
Subnet: 203.0.113.0/24
External Physical Switch
External Physical Router
IP Address: 203.0.113.1
Internet

外部ネットワーク

The external network typically provides internet access for your instances. By default, this network only allows internet access from instances using Network Address Translation (NAT). You can enable internet access to individual instances using a floating IP address and suitable security group rules. The admin tenant owns this network because it provides external network access for multiple tenants. You must also enable sharing to allow access by those tenants.

注記

Perform these commands on the controller node.

To create the external network

1. admin プロジェクトのクレデンシャルを読み込みます。

```
$ source admin-openrc.sh
```

2. ネットワークを作成します。

```
$ neutron net-create ext-net --shared --router:external=True
Created a new network:
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | True                                 |
| id                        | 893aebb9-1c1e-48be-8908-6b947f3237b3 |
| name                      | ext-net                              |
| provider:network_type     | gre                                  |
| provider:physical_network |                                      |
| provider:segmentation_id  | 1                                    |
| router:external           | True                                 |
| shared                    | True                                 |
| status                    | ACTIVE                               |
| subnets                   |                                      |
| tenant_id                 | 54cd044c64d5408b83f843d63624e0d8     |
+---------------------------+--------------------------------------+
```

Like a physical network, a virtual network requires a subnet assigned to it. The external network shares the same subnet and gateway associated with the physical network connected to the external interface on the network node. You should specify an exclusive slice of this subnet for router and floating IP addresses to prevent interference with other devices on the external network.

Replace FLOATING_IP_START and FLOATING_IP_END with the first and last IP addresses of the range that you want to allocate for floating IP addresses. Replace EXTERNAL_NETWORK_CIDR with the subnet associated with the physical network. Replace EXTERNAL_NETWORK_GATEWAY with the gateway associated with the physical network, typically the ".1" IP address. You should disable DHCP on this subnet because instances do not connect directly to the external network and floating IP addresses require manual assignment.

To create a subnet on the external network

- サブネットを作成します。

```
$ neutron subnet-create ext-net --name ext-subnet ¥
  --allocation-pool start=FLOATING_IP_START,end=FLOATING_IP_END ¥
  --disable-dhcp --gateway EXTERNAL_NETWORK_GATEWAY EXTERNAL_NETWORK_CIDR
```

For example, using 203.0.113.0/24 with floating IP address range 203.0.113.101 to 203.0.113.200:

```
$ neutron subnet-create ext-net --name ext-subnet ¥
  --allocation-pool start=203.0.113.101,end=203.0.113.200 ¥
  --disable-dhcp --gateway 203.0.113.1 203.0.113.0/24
Created a new subnet:
+-------------------+------------------------------------------------------+
| Field             | Value                                                |
+-------------------+------------------------------------------------------+
| allocation_pools  | {"start": "203.0.113.101", "end": "203.0.113.200"}   |
| cidr              | 203.0.113.0/24                                       |
| dns_nameservers   |                                                      |
| enable_dhcp       | False                                                |
| gateway_ip        | 203.0.113.1                                          |
| host_routes       |                                                      |
| id                | 9159f0dc-2b63-41cf-bd7a-289309da1391                 |
| ip_version        | 4                                                    |
| ipv6_address_mode |                                                      |
| ipv6_ra_mode      |                                                      |
| name              | ext-subnet                                           |
| network_id        | 893aebb9-1c1e-48be-8908-6b947f3237b3                 |
| tenant_id         | 54cd044c64d5408b83f843d63624e0d8                     |
+-------------------+------------------------------------------------------+
```

テナントネットワーク

The tenant network provides internal network access for instances. The architecture isolates this type of network from other tenants. The demo tenant owns this network because it only provides network access for instances within it.

注記

Perform these commands on the controller node.

To create the tenant network

1. Source the demo tenant credentials:

```
$ source demo-openrc.sh
```

2. ネットワークを作成します。

```
$ neutron net-create demo-net
Created a new network:
+---------------+--------------------------------------+
| Field         | Value                                |
+---------------+--------------------------------------+
| admin_state_up | True                                |
| id            | ac108952-6096-4243-adf4-bb6615b3de28 |
| name          | demo-net                             |
| shared        | False                                |
```

```
| status         | ACTIVE                               |
| subnets        |                                      |
| tenant_id      | cdef0071a0194d19ac6bb63802dc9bae     |
+----------------+--------------------------------------+
```

Like the external network, your tenant network also requires a subnet attached to it. You can specify any valid subnet because the architecture isolates tenant networks. Replace TENANT_NETWORK_CIDR with the subnet you want to associate with the tenant network. Replace TENANT_NETWORK_GATEWAY with the gateway you want to associate with this network, typically the ".1" IP address. By default, this subnet will use DHCP so your instances can obtain IP addresses.

To create a subnet on the tenant network

- サブネットを作成します。

```
$ neutron subnet-create demo-net --name demo-subnet ¥
  --gateway TENANT_NETWORK_GATEWAY TENANT_NETWORK_CIDR
```

Example using 192.168.1.0/24:

```
$ neutron subnet-create demo-net --name demo-subnet ¥
  --gateway 192.168.1.1 192.168.1.0/24
Created a new subnet:
+-------------------+------------------------------------------------------+
| Field             | Value                                                |
+-------------------+------------------------------------------------------+
| allocation_pools  | {"start": "192.168.1.2", "end": "192.168.1.254"}     |
| cidr              | 192.168.1.0/24                                       |
| dns_nameservers   |                                                      |
| enable_dhcp       | True                                                 |
| gateway_ip        | 192.168.1.1                                          |
| host_routes       |                                                      |
| id                | 69d38773-794a-4e49-b887-6de6734e792d                 |
| ip_version        | 4                                                    |
| ipv6_address_mode |                                                      |
| ipv6_ra_mode      |                                                      |
| name              | demo-subnet                                          |
| network_id        | ac108952-6096-4243-adf4-bb6615b3de28                 |
| tenant_id         | cdef0071a0194d19ac6bb63802dc9bae                     |
+-------------------+------------------------------------------------------+
```

A virtual router passes network traffic between two or more virtual networks. Each router requires one or more interfaces and/or gateways that provide access to specific networks. In this case, you will create a router and attach your tenant and external networks to it.

To create a router on the tenant network and attach the external and tenant networks to it

1. ルーターを作成します。

```
$ neutron router-create demo-router
Created a new router:
+-----------------------+--------------------------------------+
| Field                 | Value                                |
+-----------------------+--------------------------------------+
```

```
| admin_state_up        | True                                 |
| external_gateway_info |                                      |
| id                    | 635660ae-a254-4feb-8993-295aa9ec6418 |
| name                  | demo-router                          |
| status                | ACTIVE                               |
| tenant_id             | cdef0071a0194d19ac6bb63802dc9bae     |
+-----------------------+--------------------------------------+
```

2. Attach the router to the demo tenant subnet:

```
$ neutron router-interface-add demo-router demo-subnet
Added interface b1a894fd-aee8-475c-9262-4342afdc1b58 to router demo-router.
```

3. Attach the router to the external network by setting it as the gateway:

```
$ neutron router-gateway-set demo-router ext-net
Set gateway for router demo-router
```

接続性の検証

We recommend that you verify network connectivity and resolve any issues before proceeding further. Following the external network subnet example using 203.0.113.0/24, the tenant router gateway should occupy the lowest IP address in the floating IP address range, 203.0.113.101. If you configured your external physical network and virtual networks correctly, you you should be able to ping this IP address from any host on your external physical network.

注記

If you are building your OpenStack nodes as virtual machines, you must configure the hypervisor to permit promiscuous mode on the external network.

To verify network connectivity

- プロジェクトのゲートウェイに ping します。

```
$ ping -c 4 203.0.113.101
PING 203.0.113.101 (203.0.113.101) 56(84) bytes of data.
64 bytes from 203.0.113.101: icmp_req=1 ttl=64 time=0.619 ms
64 bytes from 203.0.113.101: icmp_req=2 ttl=64 time=0.189 ms
64 bytes from 203.0.113.101: icmp_req=3 ttl=64 time=0.165 ms
64 bytes from 203.0.113.101: icmp_req=4 ttl=64 time=0.216 ms

--- 203.0.113.101 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2999ms
rtt min/avg/max/mdev = 0.165/0.297/0.619/0.187 ms
```

Legacy networking

コントローラーノードの設定

Legacy networking primarily involves compute nodes. However, you must configure the controller node to use it.

To configure legacy networking

1. 以下のコマンドを実行します。

```
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  network_api_class nova.network.api.API
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  security_group_api nova
```

2. Compute のサービスを再起動します。

```
# service openstack-nova-api restart
# service openstack-nova-scheduler restart
# service openstack-nova-conductor restart
```

コンピュートノードの設定

This section covers deployment of a simple flat network that provides IP addresses to your instances via DHCP. If your environment includes multiple compute nodes, the multi-host feature provides redundancy by spreading network functions across compute nodes.

To install legacy networking components

- `# yum install openstack-nova-network openstack-nova-api`

To configure legacy networking

1. 以下のコマンドを実行します。

```
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  network_api_class nova.network.api.API
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  security_group_api nova
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  network_manager nova.network.manager.FlatDHCPManager
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  firewall_driver nova.virt.libvirt.firewall.IptablesFirewallDriver
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  network_size 254
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  allow_same_net_traffic False
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  multi_host True
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  send_arp_for_ha True
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  share_dhcp_address True
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  force_dhcp_release True
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  flat_interface eth1
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  flat_network_bridge br100
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  public_interface eth1
```

2. Start the services and configure them to start when the system boots:

```
# service openstack-nova-network start
# service openstack-nova-metadata-api start
# chkconfig openstack-nova-network on
# chkconfig openstack-nova-metadata-api on
```

初期ネットワークの作成

Before launching your first instance, you must create the necessary virtual network infrastructure to which the instance will connect. This network typically provides internet access from instances. You can enable internet access to individual instances using a floating IP address and suitable security group rules. The admin tenant owns this network because it provides external network access for multiple tenants.

This network shares the same subnet associated with the physical network connected to the external interface on the compute node. You should specify an exclusive slice of this subnet to prevent interference with other devices on the external network.

注記

Perform these commands on the controller node.

To create the network

1. admin プロジェクトのクレデンシャルを読み込みます。

```
$ source admin-openrc.sh
```

2. ネットワークを作成します。

   Replace NETWORK_CIDR with the subnet associated with the physical network.

```
$ nova network-create demo-net --bridge br100 --multi-host T ¥
  --fixed-range-v4 NETWORK_CIDR
```

   For example, using an exclusive slice of 203.0.113.0/24 with IP address range 203.0.113.24 to 203.0.113.32:

```
$ nova network-create demo-net --bridge br100 --multi-host T ¥
  --fixed-range-v4 203.0.113.24/29
```

注記

This command provides no output.

3. Verify creation of the network:

```
$ nova net-list
+--------------------------------------+----------+------------------+
| ID                                   | Label    | CIDR             |
+--------------------------------------+----------+------------------+
| 84b34a65-a762-44d6-8b5e-3b461a53f513 | demo-net | 203.0.113.24/29  |
+--------------------------------------+----------+------------------+
```

Next steps

Your OpenStack environment now includes the core components necessary to launch a basic instance. You can launch an instance or add more services to your environment in the following chapters.

第8章 Dashboard の追加

目次

OpenStack Dashboard は Horizon としても知られ、クラウド管理者やユーザーがさまざまな OpenStack のリソースとサービスを管理できるようになるウェブインターフェースです。

Dashboard は OpenStack API を経由して OpenStack Compute クラウドコントローラーとウェブベースで操作できます。

ここからの説明は Apache ウェブサーバーを用いて設定する導入例を示します。

Dashboard のインストールと設定をした後、以下の作業を完了できます。

- Dashboard のカスタマイズ。OpenStack クラウド管理者ガイドの Dashboard のカスタマイズセクション参照。
- Dashboard 用セッションストレージのセットアップ。「Dashboard 用セッションストレージのセットアップ」 [87] 参照。

システム要件

OpenStack Dashboard をインストールする前に、以下のシステム要件を満たしている必要があります。

- OpenStack Compute のインストール。ユーザーとプロジェクトの管理用の Identity Service の有効化。

  Identity Service と Compute のエンドポイントの URL を記録します。

- sudo 権限を持つ Identity Service のユーザー。Apache は root ユーザーのコンテンツを処理しないため、ユーザーは sudo 権限を持つ Identity Service のユーザーとしてダッシュボードを実行する必要があります。
- Python 2.6 または 2.7。Python が Django をサポートするバージョンである必要があります。この Python のバージョンは Mac OS X を含め、あらゆるシステムで実行すべきです。インストールの前提条件はプラットフォームにより異なるかもしれません。

そして、Identity Service と通信できるノードに Dashboard をインストールし、設定します。

ユーザーのローカルマシンからウェブブラウザー経由で Dashboard にアクセスできるよう、以下の情報をユーザーに提供します。

- Dashboard にアクセスできるパブリック IP アドレス。
- Dashboard にアクセスできるユーザー名とパスワード。

お使いのウェブブラウザーが HTML5 をサポートし、クッキーと JavaScript を有効化されている必要があります。

注記

Dashboard で VNC クライアントを使用する場合、ブラウザーが HTML5 Canvas と HTML5 WebSockets をサポートする必要があります。

noVNC をサポートするブラウザーの詳細はそれぞれ https://github.com/kanaka/noVNC/blob/master/README.md と https://github.com/kanaka/noVNC/wiki/Browser-support を参照してください。

Dashboard のインストール

Dashboard をインストールし、設定する前に 「システム要件」 [85] にある要件を満たしている必要があります。

注記

bject Storage と Identity Service のみをインストールしたとき、Dashboard をインストールしても、プロジェクトが表示されず、使用することもできません。

Dashboard の導入方法の詳細は deployment topics in the developer documentation を参照してください。

1. Identity Service と通信できるノードに root として Dashboard をインストールします。

```
# yum install memcached python-memcached mod_wsgi openstack-dashboard
```

2. /etc/sysconfig/memcached に設定したものと一致されるために、/etc/openstack-dashboard/local_settings の CACHES['default']['LOCATION'] の値を変更します。

   /etc/openstack-dashboard/local_settings を開き、この行を探します。

```
CACHES = {
'default': {
'BACKEND' : 'django.core.cache.backends.memcached.MemcachedCache',
'LOCATION' : '127.0.0.1:11211'
}
}
```

注

- アドレスとポートは /etc/sysconfig/memcached に設定したものと一致する必要があります。

  memcached 設定を変更する場合、変更を反映するために Apache ウェブサーバーを再起動する必要があります。

- セッションストレージのために memcached 以外のオプションを使用することもできます。SESSION_ENGINE オプションによりセッションバックエンドを設定します。
- タイムゾーンを変更する場合、ダッシュボードを使用します。または /etc/openstack-dashboard/local_settings ファイルを編集します。

  次のパラメーターを変更します。 TIME_ZONE = "UTC"

3. Dashboard にアクセスしたいアドレスを含めるために local_settings.py の ALLOWED_HOSTS を更新します。

   filename os="centos;fedora;rhel">/etc/openstack-dashboard/local_settings

   ```
   ALLOWED_HOSTS = ['localhost', 'my-desktop']
   ```

4. このガイドはコントローラーノードで Dashboard を実行していると仮定します。local_settings.py の設定を適切に変更することにより、別のサーバーで Dashboard を簡単に実行できます。

   /etc/openstack-dashboard/local_settings を編集し、OPENSTACK_HOST を Identity Service のホスト名に変更します。

   ```
   OPENSTACK_HOST = "controller"
   ```

5. システムの SELinux ポリシーが HTTP サーバーにネットワーク接続を許可するよう設定されていることを確認します。

   ```
   # setsebool httpd_can_network_connect on
   ```

6. Apache ウェブサーバーと memcached を起動します。

   ```
   # service httpd start
   # service memcached start
   # chkconfig httpd on
   # chkconfig memcached on
   ```

7. これで Dashboard に http://controller/dashboard からアクセスできます。

   OpenStack Identity Service で作成したどれかのユーザーのクレデンシャルでログインします。

Dashboard 用セッションストレージのセットアップ

Dashboard はユーザーのセッションデータを処理するために Django セッションフレームワーク を使用します。しかしながら、あらゆる利用可能なセッションバックエンドを使用できます。local_settings ファイル (Fedora/RHEL/CentOS の場合: /etc/openstack-dashboard/local_settings、Ubuntu/Debian の場合: /etc/openstack-dashboard/local_settings.py、openSUSE の場合: /srv/www/openstack-dashboard/openstack_dashboard/local/local_settings.py) にある SESSION_ENGINE 設定によりセッションバックエンドをカスタマイズします。

以下のセクションは、Dashboard の導入に関する各選択肢の賛否について記載します。

ローカルメモリキャッシュ

ローカルメモリストレージは、外部にまったく何も依存しないため、セットアップすることが最速かつ容易なバックエンドです。以下の重大な弱点があります。

- プロセスやワーカーをまたがる共有ストレージがありません。
- プロセス終了後の永続性がありません。

ローカルメモリバックエンドは、依存関係がないため、Horizon 単体のデフォルトとして有効化されています。本番環境や深刻な開発作業の用途に推奨しません。以下のように有効化します。

```
SESSION_ENGINE = 'django.contrib.sessions.backends.cache'
CACHES = {
    'BACKEND': 'django.core.cache.backends.locmem.LocMemCache'
}
```

キーバリューストア

外部キャッシュのために Memcached や Redis のようなアプリケーションを使用できます。これらのアプリケーションは永続性と共有ストレージを提供します。小規模な環境や開発環境に有用です。

Memcached

Memcached is a high-performance and distributed memory object caching system providing in-memory key-value store for small chunks of arbitrary data.

要件:

- Memcached サービスが実行中であり、アクセス可能であること。
- Python モジュール python-memcached がインストールされていること。

以下のように有効化します。

```
SESSION_ENGINE = 'django.contrib.sessions.backends.cache'
CACHES = {
    'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache'
    'LOCATION': 'my_memcached_host:11211',
}
```

Redis

Redis はオープンソースで BSD ライセンスの高度なキーバリューストアです。しばしばデータ構造サーバーとして参照されます。

要件:

- Redis サービスが実行中であり、アクセス可能であること。
- Python モジュール redis と django-redis がインストールされていること。

以下のように有効化します。

```
SESSION_ENGINE = 'django.contrib.sessions.backends.cache'
CACHES = {
    "default": {
        "BACKEND": "redis_cache.cache.RedisCache",
        "LOCATION": "127.0.0.1:6379:1",
        "OPTIONS": {
            "CLIENT_CLASS": "redis_cache.client.DefaultClient",
        }
    }
}
```

データベースの初期化と設定

データベースのセッションバックエンドはスケーラブルかつ永続的です。高い多重度と高可用性を実現できます。

しかしながら、データベースのセッションバックエンドは、より低速なセッションストレージの一つであり、高負荷環境で大きなオーバーヘッドを引き起こします。データベース環境の適切な設定は大きな仕事であり、このドキュメントの範囲を越えています。

1. mysql コマンドラインクライアントを実行します。

   ```
   $ mysql -u root -p
   ```

2. プロンプトが表示されたら、MySQL の root ユーザのパスワードを入力します。

3. MySQL データベースを設定するために、dash データベースを作成します。

   ```
   mysql> CREATE DATABASE dash;
   ```

4. 新しく作成した dash データベース用の MySQL ユーザーを作成し、データベースのフルアクセスを許可します。DASH_DBPASS を新しいユーザー用のパスワードで置き換えます。

   ```
   mysql> GRANT ALL ON dash.* TO 'dash'@'%' IDENTIFIED BY 'DASH_DBPASS';
   mysql> GRANT ALL ON dash.* TO 'dash'@'localhost' IDENTIFIED BY 'DASH_DBPASS';
   ```

5. mysql> プロンプトで quit と入力し、MySQL から抜けます。

6. local_settings ファイル (Fedora/RHEL/CentOS の場合: /etc/openstack-dashboard/local_settings、Ubuntu/Debian の場合: /etc/openstack-dashboard/local_settings.py、openSUSE の場合: /srv/www/openstack-dashboard/openstack_dashboard/local/local_settings.py) で、これらのオプションを変更します。

   ```
   SESSION_ENGINE = 'django.core.cache.backends.db.DatabaseCache'
   DATABASES = {
       'default': {
           # Database configuration here
           'ENGINE': 'django.db.backends.mysql',
           'NAME': 'dash',
           'USER': 'dash',
           'PASSWORD': 'DASH_DBPASS',
           'HOST': 'localhost',
           'default-character-set': 'utf8'
       }
   }
   ```

7. After configuring the local_settings as shown, you can run the manage.py syncdb command to populate this newly-created database.

```
$ /usr/share/openstack-dashboard/manage.py syncdb
```

openSUSE ではパスが /srv/www/openstack-dashboard/manage.py であることに注意してください。

結果として、以下の出力が返されます。

```
Installing custom SQL ...
Installing indexes ...
DEBUG:django.db.backends:(0.008) CREATE INDEX `django_session_c25c2c28` ON
 `django_session` (`expire_date`);; args=()
No fixtures found.
```

8. Ubuntu の場合: apache2 を再起動するときに、警告を避けたい場合、以下のようにダッシュボードのディレクトリにブラックホールディレクトリを作成します。

```
# mkdir -p /var/lib/dash/.blackhole
```

9. デフォルトのサイトとシンボリックの設定を取得するために Apache を再起動します。

On Ubuntu:

```
# /etc/init.d/apache2 restart
```

On Fedora/RHEL/CentOS:

```
# service httpd restart
```

```
# service apache2 restart
```

On openSUSE:

```
# systemctl restart apache2.service
```

10. Ubuntu の場合、API サーバーがエラーなくダッシュボードに接続できることを確実にするために nova-api サービスを再起動します。

```
# service nova-api restart
```

キャッシュ付きデータベース

To mitigate the performance issues of database queries, you can use the Django cached_db session back end, which utilizes both your database and caching infrastructure to perform write-through caching and efficient retrieval.

前に説明したように、データベースとキャッシュの両方を設定することにより、このハイブリッド設定を有効化します。そして、以下の値を設定します。

```
SESSION_ENGINE = "django.contrib.sessions.backends.cached_db"
```

クッキー

If you use Django 1.4 or later, the signed_cookies back end avoids server load and scaling problems.

このバックエンドは、ユーザーのブラウザーにより保存されるクッキーにセッションデータを保存します。バックエンドは、セッションデータが転送中に改ざんされていないことを保証するために、暗号的な署名技術を使用します。これは暗号化とは違います。セッションデータは攻撃者により読み取りできます。

このエンジンのいいところは、追加の依存関係や環境のオーバーヘッドが必要ないことです。また、保存されるセッションデータの量が通常のクッキーに収まる限り、どこまでもスケールします。

最大の欠点は、ユーザーのマシンのストレージにセッションデータを保存し、ネットワーク経由で送信されることです。また、保存できるセッションデータの量に限りがあります。

Django cookie-based sessions ドキュメントを参照してください。

Next steps

Your OpenStack environment now includes the dashboard. You can launch an instance or add more services to your environment in the following chapters.

第9章 Block Storage Service の追加

目次

The OpenStack Block Storage service works through the interaction of a series of daemon processes named cinder-* that reside persistently on the host machine or machines. You can run the binaries from a single node or across multiple nodes. You can also run them on the same node as other OpenStack services. The following sections introduce Block Storage service components and concepts and show you how to configure and install the Block Storage service.

Block Storage

The Block Storage service enables management of volumes, volume snapshots, and volume types. It includes the following components:

- cinder-api: Accepts API requests and routes them to cinder-volume for action.
- cinder-volume: Responds to requests to read from and write to the Block Storage database to maintain state, interacting with other processes (like cinder-scheduler) through a message queue and directly upon block storage providing hardware or software. It can interact with a variety of storage providers through a driver architecture.
- cinder-scheduler daemon: Like the nova-scheduler, picks the optimal block storage provider node on which to create the volume.
- Messaging queue: Routes information between the Block Storage service processes.

The Block Storage service interacts with Compute to provide volumes for instances.

Configure a Block Storage service controller

注記

This scenario configures OpenStack Block Storage services on the Controller node and assumes that a second node provides storage through the cinder-volume service.

For instructions on how to configure the second node, see 「Configure a Block Storage service node」 [94].

You can configure OpenStack to use various storage systems. This example uses LVM.

1. Install the appropriate packages for the Block Storage service:

```
# yum install openstack-cinder
```

2. Configure Block Storage to use your database.

   In the /etc/cinder/cinder.conf file, add this key in the [database] section and replace CINDER_DBPASS with the password for the Block Storage database that you will create in a later step:

```
# openstack-config --set /etc/cinder/cinder.conf ¥
  database connection mysql://cinder:CINDER_DBPASS@controller/cinder
```

3. Use the password that you set to log in as root to create a cinder database:

```
# mysql -u root -p
mysql> CREATE DATABASE cinder;
mysql> GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'localhost' ¥
  IDENTIFIED BY 'CINDER_DBPASS';
mysql> GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'%' ¥
  IDENTIFIED BY 'CINDER_DBPASS';
```

4. Create the database tables for the Block Storage service:

```
# su -s /bin/sh -c "cinder-manage db sync" cinder
```

5. cinder ユーザーを作成します。

   The Block Storage service uses this user to authenticate with the Identity service.

   Use the service tenant and give the user the admin role:

```
$ keystone user-create --name=cinder --pass=CINDER_PASS --email=cinder@example.com
$ keystone user-role-add --user=cinder --tenant=service --role=admin
```

6. Edit the /etc/cinder/cinder.conf configuration file:

```
# openstack-config --set /etc/cinder/cinder.conf DEFAULT ¥
  auth_strategy keystone
# openstack-config --set /etc/cinder/cinder.conf keystone_authtoken ¥
  auth_uri http://controller:5000
# openstack-config --set /etc/cinder/cinder.conf keystone_authtoken ¥
  auth_host controller
# openstack-config --set /etc/cinder/cinder.conf keystone_authtoken ¥
  auth_protocol http
# openstack-config --set /etc/cinder/cinder.conf keystone_authtoken ¥
  auth_port 35357
# openstack-config --set /etc/cinder/cinder.conf keystone_authtoken ¥
  admin_user cinder
# openstack-config --set /etc/cinder/cinder.conf keystone_authtoken ¥
  admin_tenant_name service
# openstack-config --set /etc/cinder/cinder.conf keystone_authtoken ¥
  admin_password CINDER_PASS
```

7. Configure Block Storage to use the Qpid message broker:

```
# openstack-config --set /etc/cinder/cinder.conf ¥
  DEFAULT rpc_backend cinder.openstack.common.rpc.impl_qpid
# openstack-config --set /etc/cinder/cinder.conf ¥
  DEFAULT qpid_hostname controller
```

8. Register the Block Storage service with the Identity service so that other OpenStack services can locate it:

```
$ keystone service-create --name=cinder --type=volume --description="OpenStack Block
 Storage"
```

```
$ keystone endpoint-create ¥
  --service-id=$(keystone service-list | awk '/ volume / {print $2}') ¥
  --publicurl=http://controller:8776/v1/%¥(tenant_id¥)s ¥
  --internalurl=http://controller:8776/v1/%¥(tenant_id¥)s ¥
  --adminurl=http://controller:8776/v1/%¥(tenant_id¥)s
```

9. Register a service and endpoint for version 2 of the Block Storage service API:

```
$ keystone service-create --name=cinderv2 --type=volumev2 --description="OpenStack Block
 Storage v2"
```

```
$ keystone endpoint-create ¥
  --service-id=$(keystone service-list | awk '/ volumev2 / {print $2}') ¥
  --publicurl=http://controller:8776/v2/%¥(tenant_id¥)s ¥
  --internalurl=http://controller:8776/v2/%¥(tenant_id¥)s ¥
  --adminurl=http://controller:8776/v2/%¥(tenant_id¥)s
```

10. Start and configure the Block Storage services to start when the system boots:

```
# service openstack-cinder-api start
# service openstack-cinder-scheduler start
# chkconfig openstack-cinder-api on
# chkconfig openstack-cinder-scheduler on
```

Configure a Block Storage service node

After you configure the services on the controller node, configure a second system to be a Block Storage service node. This node contains the disk that serves volumes.

You can configure OpenStack to use various storage systems. This example uses LVM.

1. システムを設定するために 2章環境の基本設定 [7] にある方法を使用します。以下の項目はコントローラーノードのインストール説明と異なることに注意してください。

   - Set the host name to block1 and use 10.0.0.41 as IP address on the management network interface. Ensure that the IP addresses and host names for both controller node and Block Storage service node are listed in the /etc/hosts file on each system.

   - コントローラーノードから同期するために、「Network Time Protocol (NTP)」 [18] にある説明に従います。

2. LVM の物理ボリュームと論理ボリュームを作成します。このガイドはこの目的のために使用される 2 番目のディスク /dev/sdb を仮定します。

```
# pvcreate /dev/sdb
# vgcreate cinder-volumes /dev/sdb
```

3. Add a filter entry to the devices section in the /etc/lvm/lvm.conf file to keep LVM from scanning devices used by virtual machines:

```
devices {
...
filter = [ "a/sda1/", "a/sdb/", "r/.*/"]
...
}
```

注記

You must add required physical volumes for LVM on the Block Storage host. Run the pvdisplay command to get a list or required volumes.

フィルター配列にある各項目は、許可するために a から、拒否するために r から始まります。Block Storage のホストで必要となる物理ボリュームは a から始まる名前を持つ必要があります。配列は一覧に無いすべてのデバイスを拒否するために "r/.*/" で終わる必要があります。

この例では、/dev/sda1 がノードのオペレーティングシステム用のボリュームが置かれるボリュームです。/dev/sdb は cinder-volumes のために予約されたボリュームです。

4. After you configure the operating system, install the appropriate packages for the Block Storage service:

```
# yum install openstack-cinder scsi-target-utils
```

5. Copy the /etc/cinder/cinder.conf configuration file from the controller, or perform the following steps to set the keystone credentials:

```
# openstack-config --set /etc/cinder/cinder.conf DEFAULT ¥
  auth_strategy keystone
# openstack-config --set /etc/cinder/cinder.conf keystone_authtoken ¥
  auth_uri http://controller:5000
# openstack-config --set /etc/cinder/cinder.conf keystone_authtoken ¥
  auth_host controller
# openstack-config --set /etc/cinder/cinder.conf keystone_authtoken ¥
  auth_protocol http
# openstack-config --set /etc/cinder/cinder.conf keystone_authtoken ¥
  auth_port 35357
# openstack-config --set /etc/cinder/cinder.conf keystone_authtoken ¥
  admin_user cinder
# openstack-config --set /etc/cinder/cinder.conf keystone_authtoken ¥
  admin_tenant_name service
# openstack-config --set /etc/cinder/cinder.conf keystone_authtoken ¥
  admin_password CINDER_PASS
```

6. Configure Block Storage to use the Qpid message broker:

```
# openstack-config --set /etc/cinder/cinder.conf ¥
```

```
  DEFAULT rpc_backend cinder.openstack.common.rpc.impl_qpid
# openstack-config --set /etc/cinder/cinder.conf ¥
    DEFAULT qpid_hostname controller
```

7. Configure Block Storage to use your MySQL database. Edit the /etc/cinder/cinder.conf file and add the following key to the [database] section. Replace CINDER_DBPASS with the password you chose for the Block Storage database:

```
# openstack-config --set /etc/cinder/cinder.conf ¥
  database connection mysql://cinder:CINDER_DBPASS@controller/cinder
```

8. Configure Block Storage to use the Image Service. Block Storage needs access to images to create bootable volumes. Edit the /etc/cinder/cinder.conf file and update the glance_host option in the [DEFAULT] section:

```
# openstack-config --set /etc/cinder/cinder.conf ¥
  DEFAULT glance_host controller
```

9. Configure the iSCSI target service to discover Block Storage volumes. Add the following line to the beginning of the /etc/tgt/targets.conf file, if it is not already present:

```
include /etc/cinder/volumes/*
```

10. Start and configure the Block Storage services to start when the system boots:

```

# service openstack-cinder-volume start
# service tgtd start
# chkconfig openstack-cinder-volume on
# chkconfig tgtd on
```

Next steps

Your OpenStack environment now includes Block Storage. You can launch an instance or add more services to your environment in the following chapters.

第10章 Object Storage の追加

目次

OpenStack Object Storage Service はオブジェクトストレージと REST API 経由の取得を提供するために一緒に動作します。このアーキテクチャー例は、Keystone として知られる Identity Service がすでにインストールされている必要があります。

Object Storage Service

Object Storage Service は高いスケーラビリティを持つ、永続的なマルチテナントのオブジェクトストレージシステムです。RESTful HTTP API 経由で利用する低コストで大規模な非構造データに向いています。

以下のコンポーネントを含みます。

- プロキシサーバー (swift-proxy-server)。ファイルのアップロード、メタデータの変更、コンテナーの作成をするために、Object Storage API と生の HTTP リクエストを受け付けます。ウェブブラウザーにファイルやコンテナーを一覧表示します。パフォーマンスを改善するために、プロキシサーバーはオプションとしてキャッシュを使用できます。通常は memcache を用います。

- アカウントサーバー (swift-account-server)。Object Storage Service で定義されたアカウントを管理します。

- コンテナーサーバー (swift-container-server)。Object Storage Service の中で、コンテナーやフォルダーの対応付けを管理します。

- オブジェクトサーバー (swift-object-server)。ストレージノードでファイルのような実際のオブジェクトを管理します。

- いくつかの定期的なプロセス。大規模なデータストアでハウスキーピング作業を実行します。複製サービスにより、クラスター全体で一貫性と可用性が確保されます。他の定期的なプロセスにオーディター、アップデーター、リーパーなどがあります。

- 認証を処理する、設定可能な WSGI ミドルウェア。通常は Identity Service。

System requirements for Object Storage

ハードウェア: OpenStack Object Storage は一般的なハードウェアで実行するために設計されています。

注記

Object Storage と Identity Service のみをインストールするとき、Compute と Image Service もインストールしなければ、ダッシュボードを使用できません。

表10.1 ハードウェア推奨事項

Server	推奨ハードウェア	注
Object Storage オブジェクトサーバー	プロセッサー: 4 コア 2 個 メモリ: 8 ～ 12 GB RAM ディスク容量: 容量単価に最適なもの ネットワーク: 1 GB NIC 1 個	ディスクの合計容量はどのくらいラック効率良く収容できるかに依存します。エンタープライズ向けの一般的な故障率を達成しながら、GB 単価に最適なものにしたいです。Rackspace の場合、ストレージサーバーは現在、24 本の 2TB SATA ディスクと 8 コアのプロセッサーを持つごく一般的な 4U サーバーを実行しています。ストレージディスクの RAID は必要ではなく、推奨しません。Swift のディスク利用パターンは RAID に対して考えられる最悪のケースです。RAID 5 や 6 を使用すると、パフォーマンスが非常にすぐに劣化します。 例として、Rackspace は 24 本の 2TB SATA ディスクと 8 コアのプロセッサーを持つ Cloud Files ストレージサーバーを稼働しています。多くのサービスは、設定でワーカーと多重度をサポートします。これにより、サービスが利用可能なコアを効率的に使用できます。
Object Storage コンテナー/アカウントサーバー	プロセッサー: 4 コア 2 個 メモリ: 8 ～ 12 GB RAM ネットワーク: 1 GB NIC 1 個	SQLite データベースと関わるため IOPS に最適化します。
Object Storage プロキシサーバー	プロセッサー: 4 コア 2 個 ネットワーク: 1 GB NIC 1 個	より高いネットワークスループットにより、多くの API リクエストをサポートするためのより良いパフォーマンスを提供します。 最高の CPU パフォーマンスのためにプロキシサーバーを最適化します。プロキシサービスはより多くの CPU 処理とネットワーク I/O 集中が発生します。10 ギガネットワークをプロキシに使用している場合、または SSL 通信をプロキシで終了している場合、さらに多くの CPU パワーが必要になります。

オペレーティングシステム: OpenStack Object Storage は現在 Ubuntu、RHEL、CentOS、Fedora、openSUSE、SLES で動作します。

ネットワーク: 内部的に 1Gpbs か 10 Gbps が推奨されます。OpenStack Object Storage の場合には、外部ネットワークが外部とプロキシサーバーを接続すべきです。また、ストレージネットワークがプライベートネットワークで分離されていることを意図しています。

データベース: OpenStack Object Storage の場合には、SQLite データベースが OpenStack Object Storage のコンテナーとアカウントの管理プロセスの一部です。

権限: OpenStack Object Storage を root としてインストールできます。または、すべての権限を有効化するよう sudoers ファイルを設定する場合、sudo 権限を持つユーザーとしてインストールできます。

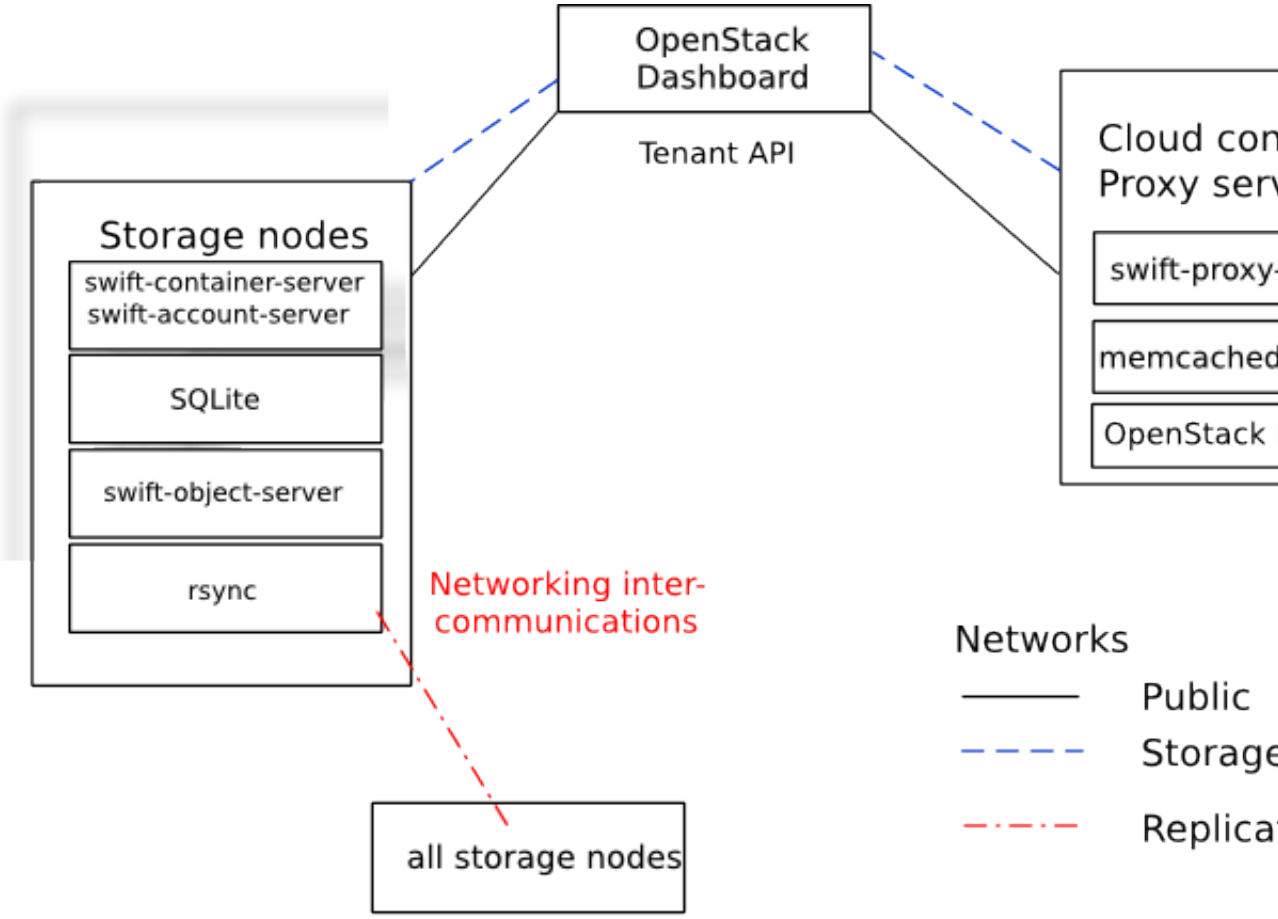
Object Storage 用ネットワークの計画

ネットワークリソースの節約のため、およびネットワーク管理者が必要に応じて API とストレージのネットワークへのアクセスを提供するためのネットワークとパブリック IP アドレスの必要性について確実に理解するために、このセクションは推奨量と必須の最小量を提供します。少なくとも 1000 Mbps のスループットが推奨されます。

このガイドは以下のネットワークを記載します。

- A mandatory public network. Connects to the proxy server.
- 必須のストレージネットワーク。クラスターの外部からアクセスできません。すべてのノードがこのネットワークに接続されます。
- An optional replication network. Not accessible from outside the cluster. Dedicated to replication traffic among storage nodes. Must be configured in the Ring.

This figure shows the basic architecture for the public network, the storage network, and the optional replication network.



By default, all of the OpenStack Object Storage services, as well as the rsync daemon on the storage nodes, are configured to listen on their STORAGE_LOCAL_NET IP addresses.

リングで複製ネットワークを設定する場合、アカウントサーバー、コンテナーサーバー、オブジェクトサーバーが STORAGE_LOCAL_NET と STORAGE_REPLICATION_NET の IP アドレスをリッスンします。rsync デーモンは STORAGE_REPLICATION_NET IP アドレスのみをリッスンします。

パブリックネットワーク（パブリックにルーティング可能な IP 範囲）	クラウドインフラストラクチャーの中で API エンドポイントにアクセス可能なパブリック IP を提供します。 最小量: 各プロキシサーバーに対して IP アドレス 1 つ。
ストレージネットワーク (RFC1918 IP 範囲、パブリックにルーティングできません)	Object Storage インフラストラクチャーの中ですべてのサーバー間通信を管理します。 最小量: 各ストレージノードとプロキシサーバーに対して IP アドレス 1 つ。 推奨量: 上のとおり、クラスターの最大量に拡張するための余地を持ちます。例えば、255 や CIDR /24 です。
複製ネットワーク（RFC1918 IP 範囲、パブリックにルーティングできません）	Object Storage インフラストラクチャーの中でストレージサーバー間の複製関連の通信を管理します。 推奨量: STORAGE_LOCAL_NET に限ります。

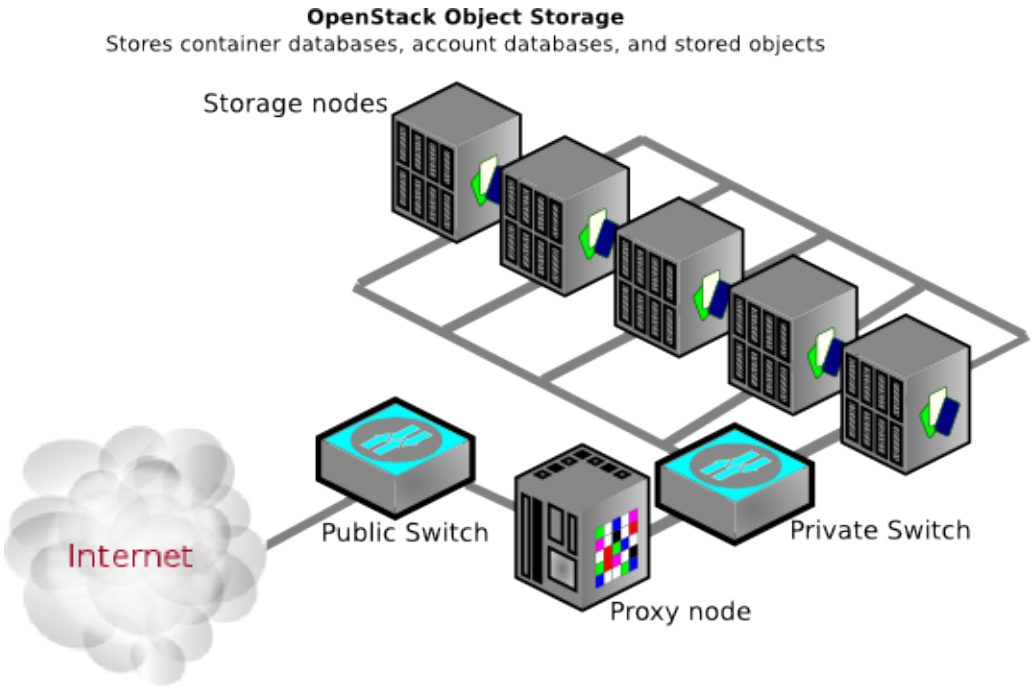
Example of Object Storage installation architecture

- Node: A host machine that runs one or more OpenStack Object Storage services.
- Proxy node: Runs proxy services.
- Storage node: Runs account, container, and object services. Contains the SQLite databases.
- Ring: A set of mappings between OpenStack Object Storage data to physical devices.
- Replica: A copy of an object. By default, three copies are maintained in the cluster.
- Zone: A logically separate section of the cluster, related to independent failure characteristics.
- Region (optional): A logically separate section of the cluster, representing distinct physical locations such as cities or countries. Similar to zones but

representing physical locations of portions of the cluster rather than logical segments.

信頼性とパフォーマンスを向上させるために、追加のプロキシノードを追加できます。

This document describes each storage node as a separate zone in the ring. At a minimum, five zones are recommended. A zone is a group of nodes that are as isolated as possible from other nodes (separate servers, network, power, even geography). The ring guarantees that every replica is stored in a separate zone. This diagram shows one possible configuration for a minimal installation:



Object Storage のインストール

OpenStack Object Storage を開発もしくはテスト目的で一つのサーバーにインストールすることができますが、複数のサーバーにインストールすることで、本番環境の分散オブジェクトストレージシステムに期待する高可用性と冗長性を実現できます。

開発目的でソースコードから単一ノードのインストールを実行するために、 Swift All In One 手順（Ubuntu）や DevStack（複数のディストリビューション）を使用します。手動インストールは http://swift.openstack.org/development_saio.html を参照してください。Identity Service（keystone）を用いた認証を含む、オールインワンは http://devstack.org を参照してください。

始める前に

新規サーバーにインストールしている場合、利用可能なオペレーティングシステムのインストールメディアを準備します。

これらの手順は OpenStack パッケージに示されている、お使いのオペレーティングシステム用のパッケージのリポジトリをセットアップしていることを仮定します。

このドキュメントは以下の種類のノードを使用したクラスターをインストールする方法を説明しています。

- swift-proxy-server プロセスを実行する 1 台のプロキシノード。このプロキシサーバーは適切なストレージノードにリクエストを中継します。
- swift-account-server、swift-container-server、swift-object-server プロセスを実行する 5 台のストレージノード。これはアカウントデータベース、コンテナーデータベース、実際のオブジェクトの保存を制御します。

注記

最初はより少ない台数のストレージノードを使用することができますが、本番環境のクラスターは少なくとも 5 台が推奨されます。

一般的なインストール手順

1. Object Storage Service が Identity Service で認証するために使用する swift ユーザーを作成します。swift ユーザー用のパスワードと電子メールアドレスを選択します。service プロジェクトを使用し、ユーザーに admin ロールを与えます。

```
$ keystone user-create --name=swift --pass=SWIFT_PASS ¥
  --email=swift@example.com
$ keystone user-role-add --user=swift --tenant=service --role=admin
```

2. Object Storage Service のサービスエントリーを作成します。

```
$ keystone service-create --name=swift --type=object-store ¥
  --description="OpenStack Object Storage"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |     OpenStack Object Storage     |
|      id     | eede9296683e4b5ebfa13f5166375ef6 |
|     name    |              swift               |
|     type    |           object-store           |
+-------------+----------------------------------+
```

注記

サービス ID はランダムに生成され、ここに表示されているものとは異なります。

3. 返されたサービス ID を使用することにより、Object Storage Service の API エンドポイントを指定します。エンドポイントを指定するとき、パブリック API、内部 API、管理 API の URL を指定します。このガイドでは、controller というホスト名を使用します。

```
$ keystone endpoint-create ¥
  --service-id=$(keystone service-list | awk '/ object-store / {print $2}') ¥
  --publicurl='http://controller:8080/v1/AUTH_%(tenant_id)s' ¥
  --internalurl='http://controller:8080/v1/AUTH_%(tenant_id)s' ¥
  --adminurl=http://controller:8080
+-------------+----------------------------------------------------+
|   Property  |                       Value                        |
+-------------+----------------------------------------------------+
```

```
|   adminurl  |            http://controller:8080/                |
|      id     |          9e3ce428f82b40d38922f242c095982e         |
| internalurl | http://controller:8080/v1/AUTH_%(tenant_id)s      |
|  publicurl  | http://controller:8080/v1/AUTH_%(tenant_id)s      |
|    region   |                      regionOne                    |
|  service_id |          eede9296683e4b5ebfa13f5166375ef6         |
+-------------+---------------------------------------------------+
```

4. すべてのノードに設定用ディレクトリを作成します。

```
# mkdir -p /etc/swift
```

5. すべてのノードで /etc/swift/swift.conf を作成します。

```
[swift-hash]
# random unique string that can never change (DO NOT LOSE)
swift_hash_path_suffix = fLIbertYgibbitZ
```

注記

/etc/swift/swift.conf のサフィックス値は、リングでマッピングを決めるためのハッシュをするときに、ソルトとして使用するために何かランダムな文字列に設定すべきです。このファイルはクラスター上のすべてのノードで同じにする必要があります。

次にストレージノードとプロキシノードをセットアップします。この例では、共通の認証部品として Identity Service を使用します。

ストレージノードのインストールと設定

注記

Object Storage works on any file system that supports Extended Attributes (XATTRS). XFS shows the best overall performance for the swift use case after considerable testing and benchmarking at Rackspace. It is also the only file system that has been thoroughly tested. See the OpenStack Configuration Reference for additional recommendations.

1. Install storage node packages:

```
# yum install openstack-swift-account openstack-swift-container ¥
  openstack-swift-object xfsprogs xinetd
```

2. ストレージ用に使用したいノードで各デバイスに対して、XFS ボリュームをセットアップします（例として /dev/sdb が使用されます）。ドライブに単一のパーティションを使用します。例えば、12 本のディスクを持つサーバーで、この手順で触れませんが、オペレーティングシステム用に1〜2 本のディスクを使用するかもしれません。他の 10〜11 本のディスクは単一のパーティションを持ち、XFS でフォーマットされるべきです。

```
# fdisk /dev/sdb
# mkfs.xfs /dev/sdb1
# echo "/dev/sdb1 /srv/node/sdb1 xfs noatime,nodiratime,nobarrier,logbufs=8 0 0" >> /etc/
fstab
```

```
# mkdir -p /srv/node/sdb1
# mount /srv/node/sdb1
# chown -R swift:swift /srv/node
```

3. /etc/rsyncd.conf を作成します。

```
uid = swift
gid = swift
log file = /var/log/rsyncd.log
pid file = /var/run/rsyncd.pid
address = STORAGE_LOCAL_NET_IP

[account]
max connections = 2
path = /srv/node/
read only = false
lock file = /var/lock/account.lock

[container]
max connections = 2
path = /srv/node/
read only = false
lock file = /var/lock/container.lock

[object]
max connections = 2
path = /srv/node/
read only = false
lock file = /var/lock/object.lock
```

4. （オプション）rsync と複製の通信を複製ネットワークと分離したい場合、STORAGE_LOCAL_NET_IP の代わりに STORAGE_REPLICATION_NET_IP を設定します。

```
address = STORAGE_REPLICATION_NET_IP
```

5. /etc/xinetd.d/rsync で以下の行を編集します。

```
disable = false
```

6. xinetd サービスを起動します。

```
# service xinetd start
```

注記

rsync サービスは認証を必要としないため、ローカルのプライベートネットワークで実行します。

7. swift recon キャッシュディレクトリを作成し、そのパーミッションを設定します。

```
# mkdir -p /var/swift/recon
# chown -R swift:swift /var/swift/recon
```

プロキシノードのインストールと設定

プロキシサーバーは各リクエストを受け取り、アカウント、コンテナー、オブジェクトの位置を検索し、リクエストを正しくルーティングします。プロキシサーバーは API リク

エストも処理します。/etc/swift/proxy-server.conf ファイルでアカウント管理を設定することにより有効化できます。

注記

The Object Storage processes run under a separate user and group, set by configuration options, and referred to as swift:swift. The default user is swift.

1. swift-proxy サービスをインストールします。

```
# yum install openstack-swift-proxy memcached python-swiftclient python-keystone-auth-token
```

2. memcached が標準のインターフェースでローカルの非パブリックなネットワークをリッスンするよう変更します。/etc/sysconfig/memcached ファイルを編集します。

```
OPTIONS="-l PROXY_LOCAL_NET_IP"
```

3. memcached サービスを起動し、システム起動時に起動するよう設定します。

```
# service memcached start
# chkconfig memcached on
```

4. /etc/swift/proxy-server.conf を編集します。

```
[DEFAULT]
bind_port = 8080
user = swift

[pipeline:main]
pipeline = healthcheck cache authtoken keystoneauth proxy-server

[app:proxy-server]
use = egg:swift#proxy
allow_account_management = true
account_autocreate = true

[filter:keystoneauth]
use = egg:swift#keystoneauth
operator_roles = Member,admin,swiftoperator

[filter:authtoken]
paste.filter_factory = keystoneclient.middleware.auth_token:filter_factory

# Delaying the auth decision is required to support token-less
# usage for anonymous referrers ('.r:*').
delay_auth_decision = true

# cache directory for signing certificate
signing_dir = /home/swift/keystone-signing

# auth_* settings refer to the Keystone server
auth_protocol = http
auth_host = controller
auth_port = 35357

# the service tenant and swift username and password created in Keystone
```

```
admin_tenant_name = service
admin_user = swift
admin_password = SWIFT_PASS

[filter:cache]
use = egg:swift#memcache

[filter:catch_errors]
use = egg:swift#catch_errors

[filter:healthcheck]
use = egg:swift#healthcheck
```

注記

複数の memcache サーバーを実行している場合、/etc/swift/proxy-server.conf ファイルの [filter:cache] セクションで複数の IP:port の一覧を置きます。

```
10.1.2.3:11211,10.1.2.4:11211
```

プロキシサーバーのみが memcache を使用します。

5. アカウント、コンテナー、オブジェクトリングを作成します。builder コマンドがいくつかのパラメーターを用いてビルダーファイルを作成します。18 という値を持つパラメーターは、パーティションの大きさが 2 の 18 乗となるを意味します。この “partition power”（パーティションのべき乗）の値は、リング全体が使用したいストレージの合計量に依存します。3 という値は各オブジェクトの複製数を表します。最後の値は一度ならずパーティションが移動することを制限する時間数です。

```
# cd /etc/swift
# swift-ring-builder account.builder create 18 3 1
# swift-ring-builder container.builder create 18 3 1
# swift-ring-builder object.builder create 18 3 1
```

6. 各ノードですべてのストレージデバイスに対して、各リングに項目を追加します。

```
# swift-ring-builder account.builder add
 zZONE-STORAGE_LOCAL_NET_IP:6002[RSTORAGE_REPLICATION_NET_IP:6005]/DEVICE 100
# swift-ring-builder container.builder add
 zZONE-STORAGE_LOCAL_NET_IP_1:6001[RSTORAGE_REPLICATION_NET_IP:6004]/DEVICE 100
# swift-ring-builder object.builder add
 zZONE-STORAGE_LOCAL_NET_IP_1:6000[RSTORAGE_REPLICATION_NET_IP:6003]/DEVICE 100
```

注記

複製のために専用のネットワークを使用したくなれば、オプションの STORAGE_REPLICATION_NET_IP パラメーターを省略する必要があります。

例えば、ストレージノードが IP 10.0.0.1 でゾーン 1 にパーティションを持つならば、ストレージノードは複製ネットワークのアドレス 10.0.1.1 を持ちます。このパーティションのマウントポイントは /srv/node/sdb1 です。/etc/rsyncd.conf のパスは /srv/node/ です。DEVICE が sdb1 になり、コマンドは次のとおりです。

```
# swift-ring-builder account.builder add z1-10.0.0.1:6002R10.0.1.1:6005/sdb1 100
# swift-ring-builder container.builder add z1-10.0.0.1:6001R10.0.1.1:6004/sdb1 100
# swift-ring-builder object.builder add z1-10.0.0.1:6000R10.0.1.1:6003/sdb1 100
```

注記

各ゾーンに対して 1 つのノードを持つ 5 つのゾーンを仮定する場合、ZONE を 1 から始めます。それぞれの追加ノードに対して、ZONE を 1 増やします。

7. 各リングのリングコンテンツを検証します。

```
# swift-ring-builder account.builder
# swift-ring-builder container.builder
# swift-ring-builder object.builder
```

8. リングを再バランスします。

```
# swift-ring-builder account.builder rebalance
# swift-ring-builder container.builder rebalance
# swift-ring-builder object.builder rebalance
```

注記

リングの再バランスには少し時間がかかります。

9. account.ring.gz、container.ring.gz、object.ring.gz ファイルをそれぞれのプロキシノードとストレージノードの /etc/swift にコピーします。

10. swift ユーザーがすべての設定ファイルを所有していることを確認します。

```
# chown -R swift:swift /etc/swift
```

11. プロキシサービスを起動し、システム起動時に起動するよう設定します。

```
# service openstack-swift-proxy start
# chkconfig openstack-swift-proxy on
```

ストレージノードでのサービスの起動

これで、リングファイルが各ストレージノードに存在するので、サービスを起動できます。各ストレージノードで以下のコマンドを実行します。

```
# for service in ¥
  openstack-swift-object openstack-swift-object-replicator openstack-swift-object-updater
 openstack-swift-object-auditor ¥
  openstack-swift-container openstack-swift-container-replicator openstack-swift-container-
updater openstack-swift-container-auditor ¥
  openstack-swift-account openstack-swift-account-replicator openstack-swift-account-reaper
 openstack-swift-account-auditor; do ¥
    service $service start; chkconfig $service on; done
```

注記

すべての Swift サービスを起動するために、次のコマンドを実行します。

```
# swift-init all start
```

swift-init コマンドについて詳しく知りたい場合、以下を実行します。

```
$ man swift-init
```

インストールの検証

プロキシサーバー、または Identity Service にアクセスできるすべてのサーバーから、これらのコマンドを実行できます。

1. Make sure that your credentials are set up correctly in the admin-openrc.sh file and source it:

   ```
   $ source admin-openrc.sh
   ```

2. Run the following swift command:

   ```
   $ swift stat
   Account: AUTH_11b9758b7049476d9b48f7a91ea11493
   Containers: 0
      Objects: 0
        Bytes: 0
   Content-Type: text/plain; charset=utf-8
   X-Timestamp: 1381434243.83760
   X-Trans-Id: txdcdd594565214fb4a2d33-0052570383
   X-Put-Timestamp: 1381434243.83760
   ```

3. Run the following swift commands to upload files to a container. Create the test.txt and test2.txt test files locally if needed.

   ```
   $ swift upload myfiles test.txt
   $ swift upload myfiles test2.txt
   ```

4. Run the following swift command to download all files from the myfiles container:

   ```
   $ swift download myfiles
   test2.txt [headers 0.267s, total 0.267s, 0.000s MB/s]
   test.txt [headers 0.271s, total 0.271s, 0.000s MB/s]
   ```

Add another proxy server

To provide additional reliability and bandwidth to your cluster, you can add proxy servers. You can set up an additional proxy node the same way that you set up the first proxy node but with additional configuration steps.

After you have more than two proxies, you must load balance them; your storage endpoint (what clients use to connect to your storage) also changes. You can select from different strategies for load balancing. For example, you could use round-robin DNS, or a software or hardware load balancer (like pound) in front of the two proxies. You can then point your storage URL to the load balancer, configure an initial proxy node and complete these steps to add proxy servers.

1. 追加のプロキシサーバーのために /etc/swift/proxy-server.conf ファイルにある memcache サーバーの一覧を更新します。複数の memcache サーバーを実行している場合、各プロキシサーバー設定ファイルで複数の IP:port の一覧に対してこのパターンを使用します。

```
10.1.2.3:11211,10.1.2.4:11211

[filter:cache]
use = egg:swift#memcache
memcache_servers = PROXY_LOCAL_NET_IP:11211
```

2. 新しいプロキシノードを含め、すべてのノードにリング情報をコピーします。また、リング情報がすべてのストレージノードに到達していることを確認します。

3. すべてのノードを同期した後、管理者が /etc/swift にあるキーを持ち、リングファイルの所有者が正しいことを確認します。

Next steps

Your OpenStack environment now includes Object Storage. You can launch an instance or add more services to your environment in the following chapters.

第11章 Orchestration Service の追加

目次

HOT と呼ばれるテンプレート言語を使用してクラウドリソースを作成するために Orchestration モジュールを使用します。統合プロジェクト名は Heat です。

Orchestration Service 概要

Orchestration Service は、クラウドアプリケーションを稼働済みにして生成するために OpenStack API コールを実行することにより、クラウドアプリケーションを記載するためのテンプレートベースのオーケストレーションを提供します。このソフトウェアは OpenStack の他のコアコンポーネントを一つのテンプレートシステムに統合します。テンプレートにより、インスタンス、Floating IP、ボリューム、セキュリティグループ、ユーザーなどのような、多くの OpenStack リソース種別を作成できます。また、インスタンスの高可用化、インスタンスのオートスケール、入れ子のスタックなどのより高度な機能をいくつか提供します。他の OpenStack コアプロジェクトと非常に緊密に統合することにより、すべての OpenStack コアプロジェクトが大規模なユーザーグループを受け取れます。

このサービスにより、開発者が Orchestration Service 直接、またはカスタムプラグイン経由で統合できるようになります。

Orchestration Service は以下のコンポーネントから構成されます。

- heat コマンドラインクライアント。AWS CloudFormation API を実行するために、heat-api と通信する CLI です。エンドの開発者は直接 Orchestration REST API を使用することもできます。
- heat-api コンポーネント。RPC 経由で API リクエストを heat-engine に送信して処理する OpenStack ネイティブの REST API を提供します。
- heat-api-cfn コンポーネント。AWS CloudFormation と互換性があり、RPC 経由で API リクエストを heat-engine に送信して処理する AWS Query API を提供します。
- heat-engine。テンプレートの開始を指示し、API コンシューマーにイベントを送り返します。

Orchestration Service のインストール

1. コントローラーノードに Orchestration モジュールをインストールします。

```
# yum install openstack-heat-api openstack-heat-engine ¥
```

```
  openstack-heat-api-cfn
```

2. Orchestration Service がデータを保存するデータベースの場所を設定ファイルで指定します。これらの例はコントローラーノードにユーザー名 heat で MySQL データベースを使用します。HEAT_DBPASS をデータベースのユーザーの適切なパスワードで置き換えます。

```
# openstack-config --set /etc/heat/heat.conf ¥
  database connection mysql://heat:HEAT_DBPASS@controller/heat
```

3. root としてログインするために前に設定したパスワードを使用し、heat データベースユーザーを作成します。

```
$ mysql -u root -p
mysql> CREATE DATABASE heat;
mysql> GRANT ALL PRIVILEGES ON heat.* TO 'heat'@'localhost' ¥
IDENTIFIED BY 'HEAT_DBPASS';
mysql> GRANT ALL PRIVILEGES ON heat.* TO 'heat'@'%' ¥
IDENTIFIED BY 'HEAT_DBPASS';
```

4. heat サービスのテーブルを作成します。

```
# su -s /bin/sh -c "heat-manage db_sync" heat
```

注記

DeprecationWarning エラーを無視します。

5. Orchestration サービスが Identity Service で認証するために使用する heat ユーザーを作成します。service プロジェクトを使用し、ユーザーに admin ロールを与えます。

```
$ keystone user-create --name=heat --pass=HEAT_PASS ¥
  --email=heat@example.com
$ keystone user-role-add --user=heat --tenant=service --role=admin
```

6. Edit the /etc/heat/heat.conf file to change the [keystone_authtoken] and [ec2authtoken] sections to add credentials to the Orchestration Service:

```
[keystone_authtoken]
auth_host = controller
auth_port = 35357
auth_protocol = http
auth_uri = http://controller:5000/v2.0
admin_tenant_name = service
admin_user = heat
admin_password = HEAT_PASS

[ec2authtoken]
auth_uri = http://controller:5000/v2.0
```

7. Register the Heat and CloudFormation APIs with the Identity Service so that other OpenStack services can locate these APIs. Register the services and specify the endpoints:

```
$ keystone service-create --name=heat --type=orchestration ¥
  --description="Orchestration"
$ keystone endpoint-create ¥
```

```
  --service-id=$(keystone service-list | awk '/ orchestration / {print $2}') ¥
  --publicurl=http://controller:8004/v1/%¥(tenant_id¥)s ¥
  --internalurl=http://controller:8004/v1/%¥(tenant_id¥)s ¥
  --adminurl=http://controller:8004/v1/%¥(tenant_id¥)s
$ keystone service-create --name=heat-cfn --type=cloudformation ¥
  --description="Orchestration CloudFormation"
$ keystone endpoint-create ¥
  --service-id=$(keystone service-list | awk '/ cloudformation / {print $2}') ¥
  --publicurl=http://controller:8000/v1 ¥
  --internalurl=http://controller:8000/v1 ¥
  --adminurl=http://controller:8000/v1
```

8. heat-api、heat-api-cfn、heat-engine サービスを起動し、システムの起動時にそれらが起動するよう設定します。

```
# service openstack-heat-api start
# service openstack-heat-api-cfn start
# service openstack-heat-engine start
# chkconfig openstack-heat-api on
# chkconfig openstack-heat-api-cfn on
# chkconfig openstack-heat-engine on
```

Orchestration Service のインストールの検証

To verify that the Orchestration service is installed and configured correctly, make sure that your credentials are set up correctly in the demo-openrc.sh file. Source the file, as follows:

```
$ source demo-openrc.sh
```

The Orchestration Module uses templates to describe stacks. To learn about the template languages, see the Template Guide in the Heat developer documentation.

Create a test template in the test-stack.yml file with the following content:

```
heat_template_version: 2013-05-23

description: Test Template

parameters:
  ImageID:
    type: string
    description: Image use to boot a server
  NetID:
    type: string
    description: Network ID for the server

resources:
  server1:
    type: OS::Nova::Server
    properties:
      name: "Test server"
      image: { get_param: ImageID }
      flavor: "m1.tiny"
      networks:
      - network: { get_param: NetID }

outputs:
```

```
  server1_private_ip:
    description: IP address of the server in the private network
    value: { get_attr: [ server1, first_address ] }
```

Use the heat stack-create command to create a stack from this template:

```
$ NET_ID=$(nova net-list | awk '/ demo-net / { print $2 }')
$ heat stack-create -f test-stack.yml ¥
  -P "ImageID=cirros-0.3.2-x86_64;NetID=$NET_ID" testStack
+--------------------------------------+------------+--------------------
+---------------------+
| id                                   | stack_name | stack_status       | creation_time
   |
+--------------------------------------+------------+--------------------
+---------------------+
| 477d96b4-d547-4069-938d-32ee990834af | testStack  | CREATE_IN_PROGRESS |
 2014-04-06T15:11:01Z |
+--------------------------------------+------------+--------------------
+---------------------+
```

Verify that the stack was created successfully with the heat stack-list command:

```
$ heat stack-list
+--------------------------------------+------------+-----------------+----------------------+
| id                                   | stack_name | stack_status    | creation_time        |
+--------------------------------------+------------+-----------------+----------------------+
| 477d96b4-d547-4069-938d-32ee990834af | testStack  | CREATE_COMPLETE | 2014-04-06T15:11:01Z |
+--------------------------------------+------------+-----------------+----------------------+
```

Next steps

Your OpenStack environment now includes Orchestration. You can launch an instance or add more services to your environment in the following chapters.

第12章 Telemetry モジュールの追加

目次

Telemetry は OpenStack クラウドのモニタリングとメータリングのフレームワークを提供します。これは Ceilometer プロジェクトとしても知られています。

Telemetry

The Telemetry module:

- CPU とネットワークのコストに関する統計データを効率的に収集します。
- サービスから送られた通知を監視すること、またはインフラストラクチャーをポーリングすることにより、データを収集します。
- さまざまな運用環境に適合するよう、収集するデータの種類を設定します。REST API 経由で統計データにアクセスおよび追加をします。
- 追加のプラグインによりカスタム利用データを収集するためにフレームワークを拡張します。
- 否認できない書名付き統計情報メッセージを作成します。

システムは以下の基本的なコンポーネントから構成されます。

- コンピュートエージェント (ceilometer-agent-compute)。各コンピュートノードで実行され、リソースの使用状況の統計情報を収集します。将来的に別の種類のエージェントができるかもしれませんが、今のところコンピュートエージェントの作成に注力しています。
- 中央エージェント (ceilometer-agent-central)。インスタンスやコンピュートノードに結びつけられていないリソースに対して、リソースの利用状況の統計情報を収集するために、中央管理サーバーで実行されます。
- コレクター (ceilometer-collector)。(エージェントから送られてくる通知や統計情報に対する) メッセージキューを監視するために、一つまたは複数の中央管理サーバーで実行されます。通知メッセージが処理され、統計情報メッセージに変えられます。適切なトピックを使用してメッセージバスの中に送り返されます。Telemetry メッセージは変更せずにデータストアに書き込まれます。

- アラーム通知（ceilometer-alarm-notifier）。いくつかの標本に対する閾値評価に基づいてアラームを設定できるようにするために、一つまたは複数の中央管理サーバーで実行されます。
- データストア。（一つまたは複数のコレクターインスタンスからの）同時書き込みや（API サーバーからの）同時読み込みを処理できる能力のあるデータベースです。
- An API server (ceilometer-api). Runs on one or more central management servers to provide access to the data from the data store.

これらのサービスは標準的な OpenStack メッセージバスを使用して通信します。コレクターと API サーバーのみがデータストアにアクセスできます。

Telemetry モジュールのインストール

Telemetry は情報収集機能とさまざまな種類のエージェントを提供する API サービスです。コンピュートノードのようなノードにこれらのエージェントをインストールする前に、コントローラーノードにコアコンポーネントをインストールするために、この手順を使用する必要があります。

1. コントローラーノードに Telemetry Service をインストールします。

```
# yum install openstack-ceilometer-api openstack-ceilometer-collector openstack-
ceilometer-central python-ceilometerclient
```

2. Telemetry Service は情報を保存するためにデータベースを使用します。設定ファイルでデータベースの場所を指定します。この例はコントローラーノードで MongoDB データベースを使用します。

```
# yum install mongodb-server mongodb
```

注記

MongoDB はデフォルトで、データベースのジャーナリングをサポートするために、/var/lib/mongodb/journal/ ディレクトリにいくつかの 1GB のファイルを作成するよう設定されます。

データベースのジャーナリングをサポートするために割り当てられる領域を最小化する必要がある場合、/etc/mongodb.conf 設定ファイルにある smallfiles 設定キーを true に設定します。この設定により、各ジャーナルファイルの容量が 512MB に減ります。

smallfiles 設定キーの詳細は MongoDB のドキュメント http://docs.mongodb.org/manual/reference/configuration-options/#smallfiles を参照してください。

データベースのジャーナリング自体を無効化する手順の詳細は http://docs.mongodb.org/manual/tutorial/manage-journaling/ を参照してください。

3. MongoDB サーバーを起動し、システム起動時に起動するよう設定します。

```
# service mongod start
# chkconfig mongod on
```

4. データベースと ceilometer データベースユーザーを作成します。

```
# mongo --host controller --eval '
db = db.getSiblingDB("ceilometer");
db.addUser({user: "ceilometer",
            pwd: "CEILOMETER_DBPASS",
            roles: [ "readWrite", "dbAdmin" ]})'
```

5. Telemetry Service がデータベースを使用するよう設定します。

```
# openstack-config --set /etc/ceilometer/ceilometer.conf ¥
  database connection mongodb://ceilometer:CEILOMETER_DBPASS@controller:27017/ceilometer
```

6. You must define a secret key that is used as a shared secret among Telemetry service nodes. Use openssl to generate a random token and store it in the configuration file:

```
# CEILOMETER_TOKEN=$(openssl rand -hex 10)
# echo $CEILOMETER_TOKEN
# openstack-config --set /etc/ceilometer/ceilometer.conf publisher metering_secret
 $CEILOMETER_TOKEN
```

7. Telemetry Service が Identity Service で認証するために使用する ceilometer ユーザーを作成します。service プロジェクトを使用し、ユーザーに admin ロールを与えます。

```
$ keystone user-create --name=ceilometer --pass=CEILOMETER_PASS --
email=ceilometer@example.com
$ keystone user-role-add --user=ceilometer --tenant=service --role=admin
```

8. クレデンシャルを Telemetry Service の設定ファイルに追加します。

```
# openstack-config --set /etc/ceilometer/ceilometer.conf ¥
  keystone_authtoken auth_host controller
# openstack-config --set /etc/ceilometer/ceilometer.conf ¥
  keystone_authtoken admin_user ceilometer
# openstack-config --set /etc/ceilometer/ceilometer.conf ¥
  keystone_authtoken admin_tenant_name service
# openstack-config --set /etc/ceilometer/ceilometer.conf ¥
  keystone_authtoken auth_protocol http
# openstack-config --set /etc/ceilometer/ceilometer.conf ¥
  keystone_authtoken auth_uri http://controller:5000
# openstack-config --set /etc/ceilometer/ceilometer.conf ¥
  keystone_authtoken admin_password CEILOMETER_PASS
# openstack-config --set /etc/ceilometer/ceilometer.conf ¥
    service_credentials os_auth_url http://controller:5000/v2.0
# openstack-config --set /etc/ceilometer/ceilometer.conf ¥
  service_credentials os_username ceilometer
# openstack-config --set /etc/ceilometer/ceilometer.conf ¥
  service_credentials os_tenant_name service
# openstack-config --set /etc/ceilometer/ceilometer.conf ¥
  service_credentials os_password CEILOMETER_PASS
```

9. Register the Telemetry service with the Identity Service so that other OpenStack services can locate it. Use the keystone command to register the service and specify the endpoint:

```
$ keystone service-create --name=ceilometer --type=metering ¥
  --description="Telemetry"
$ keystone endpoint-create ¥
```

```
  --service-id=$(keystone service-list | awk '/ metering / {print $2}') ¥
  --publicurl=http://controller:8777 ¥
  --internalurl=http://controller:8777 ¥
  --adminurl=http://controller:8777
```

10. openstack-ceilometer-api、openstack-ceilometer-central、openstack-ceilometer-collector、、 サービスを起動し、システムの起動時にそれらが起動するよう設定します。

```
# service openstack-ceilometer-api start
# service openstack-ceilometer-central start
# service openstack-ceilometer-collector start
# chkconfig openstack-ceilometer-api on
# chkconfig openstack-ceilometer-central on
# chkconfig openstack-ceilometer-collector on
```

Telemetry 用 Compute エージェントのインストール

Telemetry は情報収集機能とさまざまな種類のエージェントを提供する API サービスを提供します。この手順はコンピュートノードで実行するエージェントをインストールする方法を詳細に説明します。

1. コンピュートノードに Telemetry Service をインストールします。

```
# yum install openstack-ceilometer-compute
```

2. /etc/nova/nova.conf ファイルに以下のオプションを設定します。

```
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  instance_usage_audit True
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  instance_usage_audit_period hour
# openstack-config --set /etc/nova/nova.conf DEFAULT ¥
  notify_on_state_change vm_and_task_state
```

注記

notification_driver オプションは複数の値を持つオプションです。openstack-config はこれを正しく設定できません。「OpenStack パッケージ」 [20]を参照してください。

/etc/nova/nova.conf ファイルを編集し、[DEFAULT] セクションに以下の行を追加します。

```
[DEFAULT]
...
notification_driver = nova.openstack.common.notifier.rpc_notifier
notification_driver = ceilometer.compute.nova_notifier
```

3. Compute Service を再起動します。

```
# service openstack-nova-compute restart
```

4. 前に設定したシークレットキーを設定する必要があります。Telemetry Service ノードは共有シークレットとしてこのキーを共有します。

```
# openstack-config --set /etc/ceilometer/ceilometer.conf publisher ¥
  metering_secret CEILOMETER_TOKEN
```

5. Qpid のアクセス権を設定します。

```
# openstack-config --set /etc/ceilometer/ceilometer.conf DEFAULT qpid_hostname controller
```

6. Identity Service のクレデンシャルを追加します。

```
# openstack-config --set /etc/ceilometer/ceilometer.conf ¥
  keystone_authtoken auth_host controller
# openstack-config --set /etc/ceilometer/ceilometer.conf ¥
  keystone_authtoken admin_user ceilometer
# openstack-config --set /etc/ceilometer/ceilometer.conf ¥
  keystone_authtoken admin_tenant_name service
# openstack-config --set /etc/ceilometer/ceilometer.conf ¥
  keystone_authtoken auth_protocol http
# openstack-config --set /etc/ceilometer/ceilometer.conf ¥
  keystone_authtoken admin_password CEILOMETER_PASS
# openstack-config --set /etc/ceilometer/ceilometer.conf ¥
  service_credentials os_username ceilometer
# openstack-config --set /etc/ceilometer/ceilometer.conf ¥
  service_credentials os_tenant_name service
# openstack-config --set /etc/ceilometer/ceilometer.conf ¥
  service_credentials os_password CEILOMETER_PASS
# openstack-config --set /etc/ceilometer/ceilometer.conf ¥
  service_credentials os_auth_url http://controller:5000/v2.0
```

7. サービスを起動し、システム起動時に起動するよう設定します。

```
# service openstack-ceilometer-compute start
# chkconfig openstack-ceilometer-compute on
```

Telemetry 用 Image Service の設定

1. イメージのサンプルを取得するために、Image Service がバスに通知を送信するよう設定する必要があります。

   以下のコマンドを実行します。

```
# openstack-config --set /etc/glance/glance-api.conf DEFAULT notification_driver
 messaging
# openstack-config --set /etc/glance/glance-api.conf DEFAULT rpc_backend qpid
```

2. Restart the Image Services with their new settings:

```
# service openstack-glance-api restart
# service openstack-glance-registry restart
```

Add the Block Storage service agent for Telemetry

1. To retrieve volume samples, you must configure the Block Storage service to send notifications to the bus.

   Run the following commands on the controller and volume nodes:

```
# openstack-config --set /etc/cinder/cinder.conf DEFAULT control_exchange cinder
# openstack-config --set /etc/cinder/cinder.conf DEFAULT notification_driver cinder.
openstack.common.notifier.rpc_notifier
```

2. Restart the Block Storage services with their new settings.

   On the controller node:

   ```
   # service openstack-cinder-api restart
   # service openstack-cinder-scheduler restart
   ```

   On the volume node:

   ```
   # service openstack-cinder-volume restart
   ```

Telemetry 用 Object Storage Service の設定

1. オブジェクトストアの統計情報を取得するためには、Telemetry Service が ResellerAdmin ロールで Object Storage にアクセスする必要があります。このロールを os_tenant_name プロジェクトの os_username ユーザーに与えます。

   ```
   $ keystone role-create --name=ResellerAdmin
   +----------+----------------------------------+
   | Property |              Value               |
   +----------+----------------------------------+
   |    id    | 462fa46c13fd4798a95a3bfbe27b5e54 |
   |   name   |          ResellerAdmin           |
   +----------+----------------------------------+
   ```

   ```
   $ keystone user-role-add --tenant service --user ceilometer ¥
        --role 462fa46c13fd4798a95a3bfbe27b5e54
   ```

2. 入力通信と出力通信を処理するために、Telemetry ミドルウェアを Object Storage に追加する必要もあります。これらの行を /etc/swift/proxy-server.conf ファイルに追加します。

   ```
   [filter:ceilometer]
   use = egg:ceilometer#swift
   ```

3. ceilometer を同じファイルの pipeline パラメーターに追加します。

   ```
   [pipeline:main]
   pipeline = healthcheck cache authtoken keystoneauth ceilometer proxy-server
   ```

4. 新しい設定を用いてサービスを再起動します。

   ```
   # service openstack-swift-proxy restart
   ```

Telemetry のインストールの検証

To test the Telemetry installation, download an image from the Image Service, and use the ceilometer command to display usage statistics.

1. Telemetry へのアクセスをテストするために ceilometer meter-list コマンドを使用します。

```
$ ceilometer meter-list
```

```
+------------+-------+-------+--------------------------------------+---------
+----------------------------------+
| Name       | Type  | Unit  | Resource ID                          | User ID | Project
 ID                         |
+------------+-------+-------+--------------------------------------+---------
+----------------------------------+
| image      | gauge | image | acafc7c0-40aa-4026-9673-b879898e1fc2 | None    |
 efa984b0a914450e9a47788ad330699d |
| image.size | gauge | B     | acafc7c0-40aa-4026-9673-b879898e1fc2 | None    |
 efa984b0a914450e9a47788ad330699d |
+------------+-------+-------+--------------------------------------+---------
+----------------------------------+
```

2. Image Service からイメージをダウンロードします。

```
$ glance image-download "CirrOS 0.3.2" > cirros.img
```

3. このダウンロードが Telemetry により検知され、保存されていることを検証するために ceilometer meter-list コマンドを呼び出します。

```
$ ceilometer meter-list
```

```
+---------------+-------+-------+--------------------------------------+---------
+----------------------------------+
| Name          | Type  | Unit  | Resource ID                          | User ID |
 Project ID                       |
+---------------+-------+-------+--------------------------------------+---------
+----------------------------------+
| image         | gauge | image | acafc7c0-40aa-4026-9673-b879898e1fc2 | None    |
 efa984b0a914450e9a47788ad330699d |
| image.download | delta | B     | acafc7c0-40aa-4026-9673-b879898e1fc2 | None    |
 efa984b0a914450e9a47788ad330699d |
| image.serve    | delta | B     | acafc7c0-40aa-4026-9673-b879898e1fc2 | None    |
 efa984b0a914450e9a47788ad330699d |
| image.size     | gauge | B     | acafc7c0-40aa-4026-9673-b879898e1fc2 | None    |
 efa984b0a914450e9a47788ad330699d |
+---------------+-------+-------+--------------------------------------+---------
+----------------------------------+
```

4. さまざまなメーターの使用量の統計情報を取得できるようになりました。

```
$ ceilometer statistics -m image.download -p 60
```

```
+--------+---------------------+---------------------+-------+------------
+------------+------------+------------+---------+----------------------------
+----------------------------+
| Period | Period Start        | Period End          | Count | Min        | Max
 Sum        | Avg        | Duration | Duration Start              | Duration End
     |
+--------+---------------------+---------------------+-------+------------
+------------+------------+------------+---------+----------------------------
+----------------------------+
| 60     | 2013-11-18T18:08:50 | 2013-11-18T18:09:50 | 1     | 13167616.0 | 13167616.0 |
 13167616.0 | 13167616.0 | 0.0      | 2013-11-18T18:09:05.334000 | 2013-11-18T18:09:05.
334000 |
+--------+---------------------+---------------------+-------+------------
+------------+------------+------------+---------+----------------------------
+----------------------------+
```

Next steps

Your OpenStack environment now includes Telemetry. You can launch an instance or add more services to your environment in the previous chapters.

第13章 Add the Database service

目次

Use the Database module to create cloud database resources. The integrated project name is trove.

警告

This chapter is a work in progress. It may contain incorrect information, and will be updated frequently.

Database service overview

The Database service provides scalable and reliable cloud provisioning functionality for both relational and non-relational database engines. Users can quickly and easily utilize database features without the burden of handling complex administrative tasks. Cloud users and database administrators can provision and manage multiple database instances as needed.

The Database service provides resource isolation at high performance levels, and automates complex administrative tasks such as deployment, configuration, patching, backups, restores, and monitoring.

Process flow example. Here is a high-level process flow example for using Database services:

1. Administrator sets up infrastructure:
    a. OpenStack administrator installs the Database service.
    b. She creates one image for each type of database the administrator wants to have (one for MySQL, one for MongoDB, and so on).
    c. OpenStack administrator updates the datastore to use the new images, using the trove-manage command.
2. End user uses database service:
    a. Now that the basic infrastructure is set up, an end user can create a Trove instance (database) whenever the user wants, using the trove create command.
    b. The end user gets the IP address of the Trove instance by using the trove list command to get the ID of the instance, and then using the trove show instanceID command to get the IP address.

c. The end user can now access the Trove instance using typical database access commands. MySQL example:

```
$ mysql -u myuser -pmypass -h trove_ip_address mydb
```

Components: The Database service includes the following components:

- python-troveclient command-line client. A CLI that communicates with the trove-api component.
- trove-api component. Provides an OpenStack-native RESTful API that supports JSON to provision and manage Trove instances.
- trove-conductor service. Runs on the host, and receives messages from guest instances that want to update information on the host.
- trove-taskmanager service. Instruments the complex system flows that support provisioning instances, managing the lifecycle of instances, and performing operations on instances.
- trove-guestagent service. Runs within the guest instance. Manages and performs operations on the database itself.

Install the Database service

This procedure installs the Database module on the controller node.

前提. This chapter assumes that you already have a working OpenStack environment with at least the following components installed: Compute, Image Service, Identity.

To install the Database module on the controller:

1. Install required packages:

```
# yum install openstack-trove FIXME
```

2. Prepare OpenStack:

   a. Source the admin-openrc.sh file.

```
$ source ~/admin-openrc.sh
```

   b. Create a trove user that Compute uses to authenticate with the Identity service. Use the service tenant and give the user the admin role:

```
$ keystone user-create --name=trove --pass=TROVE_PASS --email=trove@example.com
$ keystone user-role-add --user=trove --tenant=service --role=admin
```

3. Edit the the following configuration files, taking the below actions for each file:

   - trove.conf

- trove-taskmanager.conf
- trove-conductor.conf

a. Edit the [DEFAULT] section of each file for the, client URLs and logging configuration

```
[DEFAULT]
log_dir=/var/log/trove
trove_auth_url = http://controller:5000/v2.0
nova_compute_url = http://controller:8774/v2
cinder_url = http://controller:8776/v1
swift_url = http://controller:8080/v1/AUTH_
```

b. Set these configuration keys to configure thee Database module to use the Qpid message broker:

```
# openstack-config --set /etc/trove/trove-api.conf ¥
  DEFAULT rpc_backend rabbit
# openstack-config --set /etc/trove/trove-taskmaster.conf ¥
  DEFAULT rpc_backend rabbit
# openstack-config --set /etc/trove/trove-conductor.conf ¥
  DEFAULT rpc_backend rabbit
# openstack-config --set /etc/trove/trove-api.conf DEFAULT qpid_hostname controller
# openstack-config --set /etc/trove/trove-taskmaster.conf DEFAULT
 qpid_hostname controller
# openstack-config --set /etc/trove/trove-conductor.conf DEFAULT
 qpid_hostname controller
```

c. Edit the [keystone_authtoken] section of each file so it matches the listing shown below:

```
[keystone_authtoken]
auth_host = controller
auth_port = 35357
auth_protocol = http
admin_user = trove
admin_password = TROVE_PASS
admin_tenant_name = service
auth_uri = https://controller:5000/v2.0
```

d. Edit the [database] section of each file, adding it if necessary, so it matches the listing shown below:

```
[database]
connection = mysql://trove:TROVE_DBPASS@controller/trove
```

4. Edit the trove.conf file so it matches the listing shown below:

```
[DEFAULT]
default_datastore = mysql
....
# Config option for showing the IP address that nova doles out
add_addresses = True
network_label_regex = ^NETWORK_LABEL$
....
# ============ notifer queue kombu connection options ========================
notifier_queue_hostname = controller
```

5. Edit the trove-taskmanager.conf file so it matches the listing shown below:

```
[DEFAULT]
....
# Configuration options for talking to nova via the novaclient.
# These options are for an admin user in your keystone config.
# It proxy's the token received from the user to send to nova via this admin users creds,
# basically acting like the client via that proxy token.
nova_proxy_admin_user = admin
nova_proxy_admin_pass = ADMIN_PASSS
nova_proxy_admin_tenant_name = service
....
# ============ notifer queue kombu connection options ========================
notifier_queue_hostname = controller
...
```

6. Create a trove_nolog.conf file:

```
# cp /etc/trove/trove.conf /etc/trove/trove_nolog.conf
```

Edit the trove_nolog.conf file:

```
# remove line starting with logdir and add
use_syslog=False
```

7. Prepare the trove admin database:

```
$ mysql -u root -p
mysql> CREATE DATABASE trove;
mysql> GRANT ALL PRIVILEGES ON trove.* TO trove@'localhost' IDENTIFIED BY 'TROVE_DBPASS';
mysql> GRANT ALL PRIVILEGES ON trove.* TO trove@'%' IDENTIFIED BY 'TROVE_DBPASS';
```

8. Prepare the Database service:

   a. Initialize the database:

   ```
   $ trove-manage --config-file=/etc/trove/trove_nolog.conf db_sync
   ```

   b. Create a datastore. You need to create a separate datastore for each type of database you want to use, for example, MySQL, MongoDB, Cassandra. This example shows you how to create a datastore for a MySQL database:

   ```
   $ trove-manage --config-file=/etc/trove/trove_nolog.conf datastore_update mysql ""
   ```

9. Create a trove image.

   Create an image for the type of database you want to use, for example, MySQL, MongoDB, Cassandra.

This image must have the trove guest agent installed, and it must have the trove-guestagent.conf file configured to connect to your OpenStack environment. To correctly configure the trove-guestagent.conf file, do these steps on the guest instance you are using to build your image:

- Add the following lines to trove-guestagent.conf:

```
rabbit_host = controller
rabbit_password = RABBIT_PASS

nova_proxy_admin_user = admin
nova_proxy_admin_pass = ADMIN_PASSS
nova_proxy_admin_tenant_name = service
trove_auth_url = http://controller:35357/v2.0
```

10. Update the datastore to use the new image, using the trove-manage command.

    This example shows you how to create a MySQL 5.5 datastore:

```
# trove-manage --config-file=/etc/trove/trove.conf datastore_version_update ¥
  mysql mysql-5.5 mysql glance_image_ID mysql-server-5.5 1
```

11. You must register the Database module with the Identity service so that other OpenStack services can locate it. Register the service and specify the endpoint:

```
$ keystone service-create --name=trove --type=database ¥
  --description="OpenStack Database Service"
$ keystone endpoint-create ¥
  --service-id=$(keystone service-list | awk '/ trove / {print $2}') ¥
  --publicurl=http://controller:8779/v1.0/%¥(tenant_id¥)s ¥
  --internalurl=http://controller:8779/v1.0/%¥(tenant_id¥)s ¥
  --adminurl=http://controller:8779/v1.0/%¥(tenant_id¥)s
```

12. Start Database services and configure them to start when the system boots:

```
# service openstack-trove-api start
# service openstack-trove-taskmanager start
# service openstack-trove-conductor start
# chkconfig openstack-trove-api on
# chkconfig openstack-trove-taskmanager on
# chkconfig openstack-trove-conductor on
```

Verify the Database service installation

To verify that the Database service is installed and configured correctly, try executing a Trove command:

1. Source the demo-openrc.sh file.

```
$ source ~/demo-openrc.sh
```

2. Retrieve the Trove instances list:

```
$ trove list
```

You should see output similar to this:

```
+----+------+-----------+-------------------+--------+-----------+------+
| id | name | datastore | datastore_version | status | flavor_id | size |
+----+------+-----------+-------------------+--------+-----------+------+
+----+------+-----------+-------------------+--------+-----------+------+
```

3. Assuming you have created an image for the type of database you want, and have updated the datastore to use that image, you can now create a Trove instance (database). To do this, use the trove create command.

   This example shows you how to create a MySQL 5.5 database:

```
$ trove create 名前 2 --size=2 --databases=dbname ¥
  --users ユーザー:pass --datastore_version mysql-5.5 ¥
  --datastore mysql
```

第14章 インスタンスの起動

目次

An instance is a VM that OpenStack provisions on a compute node. This guide shows you how to launch a minimal instance using the CirrOS image that you added to your environment in the 5章Image Service の設定 [36] chapter. In these steps, you use the command-line interface (CLI) on your controller node or any system with the appropriate OpenStack client libraries. To use the dashboard, see the OpenStack User Guide.

Launch an instance using Networking (neutron) or legacy networking (nova-network). For more information, see the OpenStack User Guide.

注記

These steps reference example components created in previous chapters. You must adjust certain values such as IP addresses to match your environment.

Launch an instance with Networking (neutron)

To generate a keypair

Most cloud images support public key authentication rather than conventional username/password authentication. Before launching an instance, you must generate a public/private key pair using ssh-keygen and add the public key to your OpenStack environment.

1. Source the demo tenant credentials:

```
$ source demo-openrc.sh
```

2. キーペアを生成します。

```
$ ssh-keygen
```

3. Add the public key to your OpenStack environment:

```
$ nova keypair-add --pub-key ~/.ssh/id_rsa.pub demo-key
```

注記

This command provides no output.

4. Verify addition of the public key:

```
$ nova keypair-list
```

```
+----------+-------------------------------------------------+
| Name     | Fingerprint                                     |
+----------+-------------------------------------------------+
| demo-key | 6c:74:ec:3a:08:05:4e:9e:21:22:a6:dd:b2:62:b8:28 |
+----------+-------------------------------------------------+
```

To launch an instance

To launch an instance, you must at least specify the flavor, image name, network, security group, key, and instance name.

1. A flavor specifies a virtual resource allocation profile which includes processor, memory, and storage.

   List available flavors:

```
$ nova flavor-list
+----+-----------+-----------+------+-----------+------+-------+-------------+-----------
+
| ID | Name      | Memory_MB | Disk | Ephemeral | Swap | VCPUs | RXTX_Factor | Is_Public
 |
+----+-----------+-----------+------+-----------+------+-------+-------------+-----------
+
| 1  | m1.tiny   | 512       | 1    | 0         |      | 1     | 1.0         | True
 |
| 2  | m1.small  | 2048      | 20   | 0         |      | 1     | 1.0         | True
 |
| 3  | m1.medium | 4096      | 40   | 0         |      | 2     | 1.0         | True
 |
| 4  | m1.large  | 8192      | 80   | 0         |      | 4     | 1.0         | True
 |
| 5  | m1.xlarge | 16384     | 160  | 0         |      | 8     | 1.0         | True
 |
+----+-----------+-----------+------+-----------+------+-------+-------------+-----------
+
```

   Your first instance uses the m1.tiny flavor.

   #### 注記

   You can also reference a flavor by ID.

2. 利用可能なイメージを一覧表示します。

```
$ nova image-list
+--------------------------------------+---------------------+--------+--------+
| ID                                   | Name                | Status | Server |
+--------------------------------------+---------------------+--------+--------+
| e4d5edea-f07e-4faa-8390-b71e9b747267 | cirros-0.3.2-x86_64 | ACTIVE |        |
+--------------------------------------+---------------------+--------+--------+
```

   Your first instance uses the cirros-0.3.2-x86_64 image.

3. List available networks:

```
$ neutron net-list
+--------------------------------------+----------
+------------------------------------------------------+
```

```
| id                                   | name     | subnets
                |
+--------------------------------------+----------
+-----------------------------------------------------+
| 3c612b5a-d1db-498a-babb-a4c50e344cb1 | demo-net | 20bcd3fd-5785-41fe-ac42-55ff884e3180
 192.168.1.0/24   |
| 9bce64a3-a963-4c05-bfcd-161f708042d1 | ext-net  | b54a8d85-b434-4e85-a8aa-74873841a90d
 203.0.113.0/24   |
+--------------------------------------+----------
+-----------------------------------------------------+
```

Your first instance uses the demo-net tenant network. However, you must reference this network using the ID instead of the name.

4. List available security groups:

```
$ nova secgroup-list
+--------------------------------------+---------+-------------+
| Id                                   | Name    | Description |
+--------------------------------------+---------+-------------+
| ad8d4ea5-3cad-4f7d-b164-ada67ec59473 | default | default     |
+--------------------------------------+---------+-------------+
```

Your first instance uses the default security group. By default, this security group implements a firewall that blocks remote access to instances. If you would like to permit remote access to your instance, launch it and then configure remote access.

5. インスタンスを起動します。

Replace DEMO_NET_ID with the ID of the demo-net tenant network.

```
$ nova boot --flavor m1.tiny --image cirros-0.3.2-x86_64 --nic net-id=DEMO_NET_ID ¥
  --security-group default --key-name demo-key demo-instance1
+--------------------------------------
+------------------------------------------------------------+
| Property                             | Value
          |
+--------------------------------------
+------------------------------------------------------------+
| OS-DCF:diskConfig                    | MANUAL
          |
| OS-EXT-AZ:availability_zone          | nova
          |
| OS-EXT-STS:power_state               | 0
          |
| OS-EXT-STS:task_state                | scheduling
          |
| OS-EXT-STS:vm_state                  | building
          |
| OS-SRV-USG:launched_at               | -
          |
| OS-SRV-USG:terminated_at             | -
          |
| accessIPv4                           |
          |
| accessIPv6                           |
          |
```

```
| adminPass                           | vFW7Bp8PQGNo
          |
| config_drive                        |
          |
| created                             | 2014-04-09T19:24:27Z
          |
| flavor                              | m1.tiny (1)
          |
| hostId                              |
          |
| id                                  | 05682b91-81a1-464c-8f40-8b3da7ee92c5
          |
| image                               | cirros-0.3.1-x86_64 (e4d5edea-f07e-4faa-8390-
b71e9b747267) |
| key_name                            | demo-key
          |
| metadata                            | {}
          |
| name                                | demo-instance1
          |
| os-extended-volumes:volumes_attached | []
          |
| progress                            | 0
          |
| security_groups                     | default
          |
| status                              | BUILD
          |
| tenant_id                           | 7cf50047f8df4824bc76c2fdf66d11ec
          |
| updated                             | 2014-04-09T19:24:27Z
          |
| user_id                             | 0e47686e72114d7182f7569d70c519c9
          |
+-------------------------------------
+-------------------------------------------------------------+
```

6. インスタンスの状態を確認します。

```
$ nova list
+--------------------------------------+---------------+--------+------------
+-------------+-----------------------+
| ID                                   | Name          | Status | Task State | Power
 State | Networks               |
+--------------------------------------+---------------+--------+------------
+-------------+-----------------------+
| 05682b91-81a1-464c-8f40-8b3da7ee92c5 | demo-instance1 | ACTIVE | -          | Running
   | demo-net=192.168.1.3    |
+--------------------------------------+---------------+--------+------------
+-------------+-----------------------+
```

The status changes from BUILD to ACTIVE when your instance finishes the build process.

To access your instance using a virtual console

- Obtain a Virtual Network Computing (VNC) session URL for your instance and access it from a web browser:

```
$ nova get-vnc-console demo-instance1 novnc
```

```
+-------
+-----------------------------------------------------------------------------------+
| Type  | Url
    |
+-------
+-----------------------------------------------------------------------------------+
| novnc | http://controller:6080/vnc_auto.html?token=2f6dd985-f906-4bfc-b566-e87ce656375b
    |
+-------
+-----------------------------------------------------------------------------------+
```

注記

If your web browser runs on a host that cannot resolve the controller host name, you can replace controller with the IP address of the management interface on your controller node.

The CirrOS image includes conventional username/password authentication and provides these credentials at the login prompt. After logging into CirrOS, we recommend that you verify network connectivity using ping.

Verify the demo-net tenant network gateway:

```
$ ping -c 4 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_req=1 ttl=64 time=0.357 ms
64 bytes from 192.168.1.1: icmp_req=2 ttl=64 time=0.473 ms
64 bytes from 192.168.1.1: icmp_req=3 ttl=64 time=0.504 ms
64 bytes from 192.168.1.1: icmp_req=4 ttl=64 time=0.470 ms

--- 192.168.1.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2998ms
rtt min/avg/max/mdev = 0.357/0.451/0.504/0.055 ms
```

Verify the ext-net external network:

```
$ ping -c 4 openstack.org
PING openstack.org (174.143.194.225) 56(84) bytes of data.
64 bytes from 174.143.194.225: icmp_req=1 ttl=53 time=17.4 ms
64 bytes from 174.143.194.225: icmp_req=2 ttl=53 time=17.5 ms
64 bytes from 174.143.194.225: icmp_req=3 ttl=53 time=17.7 ms
64 bytes from 174.143.194.225: icmp_req=4 ttl=53 time=17.5 ms

--- openstack.org ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 17.431/17.575/17.734/0.143 ms
```

To access your instance remotely

1. default セキュリティグループにルールを追加します。

   a. ICMP (ping) を許可します。

```
$ nova secgroup-add-rule default icmp -1 -1 0.0.0.0/0
+-------------+-----------+---------+-----------+--------------+
| IP Protocol | From Port | To Port | IP Range  | Source Group |
+-------------+-----------+---------+-----------+--------------+
| icmp        | -1        | -1      | 0.0.0.0/0 |              |
```

```
+-------------+----------+---------+----------+-------------+
```

b. secure shell (SSH) アクセスを許可します。

```
$ nova secgroup-add-rule default tcp 22 22 0.0.0.0/0
+-------------+-----------+---------+-----------+-------------+
| IP Protocol | From Port | To Port | IP Range  | Source Group |
+-------------+-----------+---------+-----------+-------------+
| tcp         | 22        | 22      | 0.0.0.0/0 |              |
+-------------+-----------+---------+-----------+-------------+
```

2. Create a floating IP address on the ext-net external network:

```
$ neutron floatingip-create ext-net
Created a new floatingip:
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| fixed_ip_address    |                                      |
| floating_ip_address | 203.0.113.102                        |
| floating_network_id | 9bce64a3-a963-4c05-bfcd-161f708042d1 |
| id                  | 05e36754-e7f3-46bb-9eaa-3521623b3722 |
| port_id             |                                      |
| router_id           |                                      |
| status              | DOWN                                 |
| tenant_id           | 7cf50047f8df4824bc76c2fdf66d11ec     |
+---------------------+--------------------------------------+
```

3. Associate the floating IP address with your instance:

```
$ nova floating-ip-associate demo-instance1 203.0.113.102
```

注記

This command provides no output.

4. Floating IP アドレスの状態を確認します。

```
$ nova list
+--------------------------------------+---------------+--------+------------
+-------------+------------------------------------------+
| ID                                   | Name          | Status | Task State | Power
 State | Networks                                 |
+--------------------------------------+---------------+--------+------------
+-------------+------------------------------------------+
| 05682b91-81a1-464c-8f40-8b3da7ee92c5 | demo-instance1 | ACTIVE | -          | Running
   | demo-net=192.168.1.3, 203.0.113.102     |
+--------------------------------------+---------------+--------+------------
+-------------+------------------------------------------+
```

5. Verify network connectivity using ping from the controller node or any host on the external network:

```
$ ping -c 4 203.0.113.102
PING 203.0.113.102 (203.0.113.112) 56(84) bytes of data.
64 bytes from 203.0.113.102: icmp_req=1 ttl=63 time=3.18 ms
64 bytes from 203.0.113.102: icmp_req=2 ttl=63 time=0.981 ms
64 bytes from 203.0.113.102: icmp_req=3 ttl=63 time=1.06 ms
64 bytes from 203.0.113.102: icmp_req=4 ttl=63 time=0.929 ms
```

```
--- 203.0.113.102 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3002ms
rtt min/avg/max/mdev = 0.929/1.539/3.183/0.951 ms
```

6. Access your instance using SSH from the controller node or any host on the external network:

```
$ ssh cirros@203.0.113.102
The authenticity of host '203.0.113.102 (203.0.113.102)' can't be established.
RSA key fingerprint is ed:05:e9:e7:52:a0:ff:83:68:94:c7:d1:f2:f8:e2:e9.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '203.0.113.102' (RSA) to the list of known hosts.
$
```

注記

If your host does not contain the public/private key pair created in an earlier step, SSH prompts for the default password associated with the cirros user.

If your instance does not launch or seem to work as you expect, see the OpenStack Operations Guide for more information or use one of the many other options to seek assistance. We want your environment to work!

Launch an instance with legacy networking (nova-network)

To generate a keypair

Most cloud images support public key authentication rather than conventional username/password authentication. Before launching an instance, you must generate a public/private key pair using ssh-keygen and add the public key to your OpenStack environment.

1. Source the demo tenant credentials:

```
$ source demo-openrc.sh
```

2. キーペアを生成します。

```
$ ssh-keygen
```

3. Add the public key to your OpenStack environment:

```
$ nova keypair-add --pub-key ~/.ssh/id_rsa.pub demo-key
```

注記

This command provides no output.

4. Verify addition of the public key:

```
$ nova keypair-list
+----------+-------------------------------------------------+
| Name     | Fingerprint                                     |
+----------+-------------------------------------------------+
```

```
| demo-key | 6c:74:ec:3a:08:05:4e:9e:21:22:a6:dd:b2:62:b8:28 |
+----------+-------------------------------------------------+
```

To launch an instance

To launch an instance, you must at least specify the flavor, image name, network, security group, key, and instance name.

1. A flavor specifies a virtual resource allocation profile which includes processor, memory, and storage.

   List available flavors:

```
$ nova flavor-list
+----+-----------+-----------+------+-----------+------+-------+-------------+-----------
+
| ID | Name      | Memory_MB | Disk | Ephemeral | Swap | VCPUs | RXTX_Factor | Is_Public
 |
+----+-----------+-----------+------+-----------+------+-------+-------------+-----------
+
| 1  | m1.tiny   | 512       | 1    | 0         |      | 1     | 1.0         | True
 |
| 2  | m1.small  | 2048      | 20   | 0         |      | 1     | 1.0         | True
 |
| 3  | m1.medium | 4096      | 40   | 0         |      | 2     | 1.0         | True
 |
| 4  | m1.large  | 8192      | 80   | 0         |      | 4     | 1.0         | True
 |
| 5  | m1.xlarge | 16384     | 160  | 0         |      | 8     | 1.0         | True
 |
+----+-----------+-----------+------+-----------+------+-------+-------------+-----------
+
```

   Your first instance uses the m1.tiny flavor.

   #### 注記

   You can also reference a flavor by ID.

2. 利用可能なイメージを一覧表示します。

```
$ nova image-list
+--------------------------------------+---------------------+--------+--------+
| ID                                   | Name                | Status | Server |
+--------------------------------------+---------------------+--------+--------+
| e4d5edea-f07e-4faa-8390-b71e9b747267 | cirros-0.3.2-x86_64 | ACTIVE |        |
+--------------------------------------+---------------------+--------+--------+
```

   Your first instance uses the cirros-0.3.2-x86_64 image.

3. List available networks:

   #### 注記

   You must source the admin tenant credentials for this step and then source the demo tenant credentials for the remaining steps.

```
$ source admin-openrc.sh
```

```
$ nova net-list
+--------------------------------------+----------+------------------+
| ID                                   | Label    | CIDR             |
+--------------------------------------+----------+------------------+
| 7f849be3-4494-495a-95a1-0f99ccb884c4 | demo-net | 203.0.113.24/29  |
+--------------------------------------+----------+------------------+
```

Your first instance uses the demo-net tenant network. However, you must reference this network using the ID instead of the name.

4. List available security groups:

```
$ nova secgroup-list
+--------------------------------------+---------+-------------+
| Id                                   | Name    | Description |
+--------------------------------------+---------+-------------+
| ad8d4ea5-3cad-4f7d-b164-ada67ec59473 | default | default     |
+--------------------------------------+---------+-------------+
```

Your first instance uses the default security group. By default, this security group implements a firewall that blocks remote access to instances. If you would like to permit remote access to your instance, launch it and then configure remote access.

5. インスタンスを起動します。

Replace DEMO_NET_ID with the ID of the demo-net tenant network.

```
$ nova boot --flavor m1.tiny --image cirros-0.3.2-x86_64 --nic net-id=DEMO_NET_ID ¥
  --security-group default --key-name demo-key demo-instance1
+--------------------------------------
+------------------------------------------------------------+
| Property                             | Value
          |
+--------------------------------------
+------------------------------------------------------------+
| OS-DCF:diskConfig                    | MANUAL
          |
| OS-EXT-AZ:availability_zone          | nova
          |
| OS-EXT-STS:power_state               | 0
          |
| OS-EXT-STS:task_state                | scheduling
          |
| OS-EXT-STS:vm_state                  | building
          |
| OS-SRV-USG:launched_at               | -
          |
| OS-SRV-USG:terminated_at             | -
          |
| accessIPv4                           |
          |
| accessIPv6                           |
          |
| adminPass                            | ThZqrg7ach78
          |
| config_drive                         |
          |
```

```
| created                              | 2014-04-10T00:09:16Z
          |
| flavor                               | m1.tiny (1)
          |
| hostId                               |
          |
| id                                   | 45ea195c-c469-43eb-83db-1a663bbad2fc
          |
| image                                | cirros-0.3.1-x86_64
 (081dab35-690e-419b-8ce0-7fe232e5fec6) |
| key_name                             | demo-key
          |
| metadata                             | {}
          |
| name                                 | demo-instance1
          |
| os-extended-volumes:volumes_attached | []
          |
| progress                             | 0
          |
| security_groups                      | default
          |
| status                               | BUILD
          |
| tenant_id                            | 93849608fe3d462ca9fa0e5dbfd4d040
          |
| updated                              | 2014-04-10T00:09:16Z
          |
| user_id                              | 8397567baf4746cca7a1e608677c3b23
          |
+--------------------------------------
+------------------------------------------------------------+
```

6. インスタンスの状態を確認します。

```
$ nova list
+--------------------------------------+----------------+--------+------------
+-------------+----------------------+
| ID                                   | Name           | Status | Task State | Power
 State | Networks              |
+--------------------------------------+----------------+--------+------------
+-------------+----------------------+
| 45ea195c-c469-43eb-83db-1a663bbad2fc | demo-instance1 | ACTIVE | -          | Running
   | demo-net=203.0.113.26  |
+--------------------------------------+----------------+--------+------------
+-------------+----------------------+
```

The status changes from BUILD to ACTIVE when your instance finishes the build process.

To access your instance using a virtual console

- Obtain a Virtual Network Computing (VNC) session URL for your instance and access it from a web browser:

```
$ nova get-vnc-console demo-instance1 novnc
+-------
+-------------------------------------------------------------------------------------+
| Type  | Url
    |
```

```
+-------
+--------------------------------------------------------------------------------------+
| novnc | http://controller:6080/vnc_auto.html?token=2f6dd985-f906-4bfc-b566-e87ce656375b
    |
+-------
+--------------------------------------------------------------------------------------+
```

注記

If your web browser runs on a host that cannot resolve the controller host name, you can replace controller with the IP address of the management interface on your controller node.

The CirrOS image includes conventional username/password authentication and provides these credentials at the login prompt. After logging into CirrOS, we recommend that you verify network connectivity using ping.

demo-net ネットワークを検証します。

```
$ ping -c 4 openstack.org
PING openstack.org (174.143.194.225) 56(84) bytes of data.
64 bytes from 174.143.194.225: icmp_req=1 ttl=53 time=17.4 ms
64 bytes from 174.143.194.225: icmp_req=2 ttl=53 time=17.5 ms
64 bytes from 174.143.194.225: icmp_req=3 ttl=53 time=17.7 ms
64 bytes from 174.143.194.225: icmp_req=4 ttl=53 time=17.5 ms

--- openstack.org ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 17.431/17.575/17.734/0.143 ms
```

To access your instance remotely

1. default セキュリティグループにルールを追加します。

   a. ICMP (ping) を許可します。

   ```
   $ nova secgroup-add-rule default icmp -1 -1 0.0.0.0/0
   +-------------+-----------+---------+-----------+--------------+
   | IP Protocol | From Port | To Port | IP Range  | Source Group |
   +-------------+-----------+---------+-----------+--------------+
   | icmp        | -1        | -1      | 0.0.0.0/0 |              |
   +-------------+-----------+---------+-----------+--------------+
   ```

   b. secure shell (SSH) アクセスを許可します。

   ```
   $ nova secgroup-add-rule default tcp 22 22 0.0.0.0/0
   +-------------+-----------+---------+-----------+--------------+
   | IP Protocol | From Port | To Port | IP Range  | Source Group |
   +-------------+-----------+---------+-----------+--------------+
   | tcp         | 22        | 22      | 0.0.0.0/0 |              |
   +-------------+-----------+---------+-----------+--------------+
   ```

2. Verify network connectivity using ping from the controller node or any host on the external network:

   ```
   $ ping -c 4 203.0.113.26
   PING 203.0.113.26 (203.0.113.26) 56(84) bytes of data.
   64 bytes from 203.0.113.26: icmp_req=1 ttl=63 time=3.18 ms
   ```

```
64 bytes from 203.0.113.26: icmp_req=2 ttl=63 time=0.981 ms
64 bytes from 203.0.113.26: icmp_req=3 ttl=63 time=1.06 ms
64 bytes from 203.0.113.26: icmp_req=4 ttl=63 time=0.929 ms

--- 203.0.113.26 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3002ms
rtt min/avg/max/mdev = 0.929/1.539/3.183/0.951 ms
```

3. Access your instance using SSH from the controller node or any host on the external network:

```
$ ssh cirros@203.0.113.26
The authenticity of host '203.0.113.26 (203.0.113.26)' can't be established.
RSA key fingerprint is ed:05:e9:e7:52:a0:ff:83:68:94:c7:d1:f2:f8:e2:e9.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '203.0.113.26' (RSA) to the list of known hosts.
$
```

注記

If your host does not contain the public/private key pair created in an earlier step, SSH prompts for the default password associated with the cirros user.

If your instance does not launch or seem to work as you expect, see the OpenStack Operations Guide for more information or use one of the many other options to seek assistance. We want your environment to work!

付録A 予約済みユーザー ID

OpenStack では、特定のユーザー ID が特定の OpenStack サービスを実行し、特定の OpenStack ファイルを所有するために、予約され、使用されます。これらのユーザーはディストリビューションのパッケージにより設定されます。以下の表はその概要です。

表A.1 予約済みユーザー ID

名前	説明	ID
ceilometer	OpenStack Ceilometer デーモン	166
cinder	OpenStack Cinder デーモン	165
glance	OpenStack Glance デーモン	161
heat	OpenStack Heat デーモン	187
keystone	OpenStack Keystone デーモン	163
neutron	OpenStack Neutron デーモン	164
nova	OpenStack Nova デーモン	162
swift	OpenStack Swift デーモン	160
trove	OpenStack Trove Daemons	Unknown FIXME

各ユーザーはユーザーと同じ名前のユーザーグループに所属します。

付録B コミュニティのサポート

目次

The following resources are available to help you run and use OpenStack. The OpenStack community constantly improves and adds to the main features of OpenStack, but if you have any questions, do not hesitate to ask. Use the following resources to get OpenStack support, and troubleshoot your installations.

ドキュメント

OpenStackのドキュメントは、 docs.openstack.orgを参照してください。

ドキュメントにフィードバックするには、 OpenStack Documentation Mailing Listの<openstack-docs@lists.openstack.org>か、Launchpadのreport a bugを活用してください。

OpenStackクラウドと関連コンポーネントの導入ガイド:

- Installation Guide for Debian 7.0
- Installation Guide for openSUSE and SUSE Linux Enterprise Server
- Red Hat Enterprise Linux, CentOS, and Fedora向けインストールガイド
- Ubuntu 12.04 (LTS)向けインストールガイド

OpenStackクラウドの構成と実行ガイド:

- Cloud Administrator Guide
- Configuration Reference
- Operations Guide
- High Availability Guide
- Security Guide

- Virtual Machine Image Guide

OpenStackダッシュボードとCLIクライアントガイド

- API Quick Start
- End User Guide
- Admin User Guide
- コマンドラインインターフェースのリファレンス

OpenStack APIのリファレンスガイド

- OpenStack API Complete Reference (HTML)
- API Complete Reference (PDF)
- OpenStack Block Storage Service API v2 Reference
- OpenStack Compute API v2 and Extensions Reference
- OpenStack Identity Service API v2.0 Reference
- OpenStack Identity Service API v2.0 Reference
- OpenStack Networking API v2.0 Reference
- OpenStack Object Storage API v1 Reference

トレーニングガイドはクラウド管理者向けのソフトウェアトレーニングを提供します。

ask.openstack.org

During the set up or testing of OpenStack, you might have questions about how a specific task is completed or be in a situation where a feature does not work correctly. Use the ask.openstack.org site to ask questions and get answers. When you visit the http://ask.openstack.org site, scan the recently asked questions to see whether your question has already been answered. If not, ask a new question. Be sure to give a clear, concise summary in the title and provide as much detail as possible in the description. Paste in your command output or stack traces, links to screen shots, and any other information which might be useful.

OpenStack メーリングリスト

回答やヒントを得るとっておきの方法は、OpenStackメーリングリストへ質問や問題の状況を投稿することです。同様の問題に対処したことのある仲間が助けてくれることでしょう。購読の手続き、アーカイブの参照はhttp://lists.openstack.org/cgi-bin/mailman/listinfo/openstackで行ってください。特定プロジェクトや環境についてのメーリングリストは、on the wikiで探してみましょう。すべてのメーリングリストは、http://wiki.openstack.org/MailingListsで参照できます。

OpenStack wiki

OpenStack wikiは広い範囲のトピックを扱っていますが、情報によっては、探すのが難しかったり、情報が少なかったりします。幸いなことに、wikiの検索機能にて、タイトルと内容で探せます。もし特定の情報、たとえばネットワークや novaについて探すのであれば、多くの関連情報を見つけられます。日々追加されているため、こまめに確認してみてください。OpenStack wikiページの右上に、その検索窓はあります。

Launchpad バグエリア

OpenStackコミュニティはあなたのセットアップ、テストの取り組みに価値を感じており、フィードバックを求めています。バグを登録するには、https://launchpad.net/+loginでLaunchpadのアカウントを作成してください。Launchpadバグエリアにて、既知のバグの確認と報告ができます。すでにそのバグが報告、解決されていないかを判断するため、検索機能を活用してください。もしそのバグが報告されていなければ、バグレポートを入力しましょう。

使いこなすヒント:

- 明瞭で簡潔なまとめを!
- Provide as much detail as possible in the description. Paste in your command output or stack traces, links to screen shots, and any other information which might be useful.
- Be sure to include the software and package versions that you are using, especially if you are using a development branch, such as, "Juno release" vs git commit bc79c3ecc55929bac585d04a03475b72e06a3208.
- Any deployment specific information is helpful, such as Ubuntu 14.04 or multi-node install.

Launchpadバグエリアは下記リンクを参照してください。

- Bugs: OpenStack Block Storage (cinder)
- Bugs: OpenStack Compute (nova)
- Bugs : OpenStack Dashboard (horizon)
- Bugs : OpenStack Identity (keystone)
- Bugs : OpenStack Image Service (glance)
- Bugs : OpenStack Networking (neutron)
- Bugs : OpenStack Object Storage (swift)
- Bugs: Bare Metal (ironic)
- Bugs: Data Processing Service (sahara)
- Bugs: Database Service (trove)

- Bugs: Orchestration (heat)
- Bugs: Telemetry (ceilometer)
- Bugs: Queue Service (marconi)
- Bugs: OpenStack API Documentation (api.openstack.org)
- Bugs: OpenStack Documentation (docs.openstack.org)

OpenStack IRC チャネル

OpenStackコミュニティはFreenode上の#openstack IRCチャネルを活用しています。あなたはそこに訪れ、質問することで、差し迫った問題へのフィードバックを迅速に得られます。IRCクライアントをインストール、もしくはブラウザベースのクライアントを使うには、http://webchat.freenode.net/にアクセスしてください。また、Colloquy (Mac OS X, http://colloquy.info/), mIRC (Windows, http://www.mirc.com/), or XChat (Linux)なども使えます。IRCチャネル上でコードやコマンド出力結果を共有したい時には、Paste Binが多く使われています。OpenStackプロジェクトのPaste Binはhttp://paste.openstack.orgです。長めのテキストやログであっても、webフォームに貼り付けてURLを得るだけです。OpenStack IRCチャネルは、#openstack on irc.freenode.netです。OpenStack関連IRCチャネルは、https://wiki.openstack.org/wiki/IRCにリストがあります。

ドキュメントへのフィードバック

ドキュメントにフィードバックするには、 OpenStack Documentation Mailing Listの<openstack-docs@lists.openstack.org>か、Launchpadのreport a bugを活用してください。

OpenStackディストリビューション

OpenStackのコミュニティサポート版を提供しているディストリビューション

- Debian: http://wiki.debian.org/OpenStack
- CentOS、Fedora、およびRed Hat Enterprise Linux: http://openstack.redhat.com/
- openSUSEとSUSE Linux Enterprise Server: http://en.opensuse.org/Portal:OpenStack
- Ubuntu: https://wiki.ubuntu.com/ServerTeam/CloudArchive

用語集

API
: アプリケーションプログラミングインターフェース。

認証
: ユーザー、プロセスまたはクライアントが、秘密鍵、秘密トークン、パスワード、指紋または同様の方式により示されている主体と本当に同じであることを確認するプロセス。

CirrOS
: A minimal Linux distribution designed for use as a test image on clouds such as OpenStack.

クレデンシャル
: Data that is only known to or accessible by a user and used to verify that the user is who they say they are. Credentials are presented to the server during authentication. Examples include a password, secret key, digital certificate, fingerprint, and so on.

Database Service
: An integrated project that provide scalable and reliable Cloud Database-as-a-Service functionality for both relational and non-relational database engines. The project name of Database Service is trove.

DHCP
: Dynamic Host Configuration Protocol. A network protocol that configures devices that are connected to a network so that they can communicate on that network by using the Internet Protocol (IP). The protocol is implemented in a client-server model where DHCP clients request configuration data such as, an IP address, a default route, and one or more DNS server addresses from a DHCP server.

DHCP agent
: OpenStack Networking agent that provides DHCP services for virtual networks.

エンドポイント
: API エンドポイントを参照。

external network
: A network segment typically used for instance Internet access.

ファイアウォール
: Used to restrict communications between hosts and/or nodes, implemented in Compute using iptables, arptables, ip6tables, and etables.

Generic Receive Offload (GRO)
: Feature of certain network interface drivers that combines many smaller received packets into a large packet before delivery to the kernel IP stack.

IaaS
: Infrastructure-as-a-Service. IaaS is a provisioning model in which an organization outsources physical components of a data center such as storage, hardware, servers

and networking components. A service provider owns the equipment and is responsible for housing, operating and maintaining it. The client typically pays on a per-use basis. IaaS is a model for providing cloud services.

ICMP
Internet Control Message Protocol, used by network devices for control messages. For example, ping uses ICMP to test connectivity.

Image Service
An OpenStack core project that provides discovery, registration, and delivery services for disk and server images. The project name of the Image Service is glance.

instance tunnels network
A network segment used for instance traffic tunnels between compute nodes and the network node.

interface
A physical or virtual device that provides connectivity to another device or medium.

kernel-based VM (KVM)
OpenStack がサポートするハイパーバイザーの１つ。

Layer-3 (L3) agent
OpenStack Networking agent that provides layer-3 (routing) services for virtual networks.

Logical Volume Manager (LVM)
Provides a method of allocating space on mass-storage devices that is more flexible than conventional partitioning schemes.

multi-host
High-availability mode for legacy (nova) networking. Each compute node handles NAT and DHCP and acts as a gateway for all of the VMs on it. A networking failure on one compute node doesn't affect VMs on other compute nodes.

Network Address Translation (NAT)
The process of modifying IP address information while in-transit. Supported by Compute and Networking.

Network Time Protocol (NTP)
A method of keeping a clock for a host or node correct through communications with a trusted, accurate time source.

OpenStack
OpenStack is a cloud operating system that controls large pools of compute, storage, and networking resources throughout a data center, all managed through a dashboard that gives administrators control while empowering their users to provision resources through a web interface. OpenStack is an open source project licensed under the Apache License 2.0.

plug-in
Software component providing the actual implementation for Networking APIs, or for Compute APIs, depending on the context.

ポート
A virtual network port within Networking; VIFs / vNICs are connected to a port.

プロジェクト
A logical grouping of users within Compute, used to define quotas and access to VM images.

promiscuous mode
Causes the network interface to pass all traffic it receives to the host rather than passing only the frames addressed to it.

public key authentication
Authentication method that uses keys rather than passwords.

Qpid
Message queue software supported by OpenStack; an alternative to RabbitMQ.

RabbitMQ
OpenStackでデフォルトで採用されているメッセージキューのソフトウェア。

RESTful
A kind of web service API that uses REST, or Representational State Transfer. REST is the style of architecture for hypermedia systems that is used for the World Wide Web.

role
ユーザーが特定の操作の組を実行できると仮定する人格。ロールは一組の権利と権限を含みます。そのロールを仮定しているユーザーは、それらの権利と権限を継承します。

router
A physical or virtual network device that passes network traffic between different networks.

セキュリティグループ
A set of network traffic filtering rules that are applied to a Compute instance.

サービスカタログ
Alternative term for the Identity Service catalog.

subnet
Logical subdivision of an IP network.

Telemetry
An integrated project that provides metering and measuring facilities for OpenStack. The project name of Telemetry is ceilometer.

テナント
A group of users, used to isolate access to Compute resources. An alternative term for a project.

トークン
OpenStack API やリソースへのアクセスに使用される英数字文字列。

trove
データベースサービスをアプリケーションに提供する OpenStack のプロジェクト。

ユーザー
: In Identity Service, each user is associated with one or more tenants, and in Compute can be associated with roles, projects, or both.

virtual networking
: A generic term for virtualization of network functions such as switching, routing, load balancing, and security using a combination of VMs and overlays on physical network infrastructure.

Virtual Network Computing (VNC)
: Open source GUI and CLI tools used for remote console access to VMs. Supported by Compute.

ZeroMQ
: OpenStack によりサポートされるメッセージキューソフトウェア。RabbitMQ の代替。0MQ とも表記。