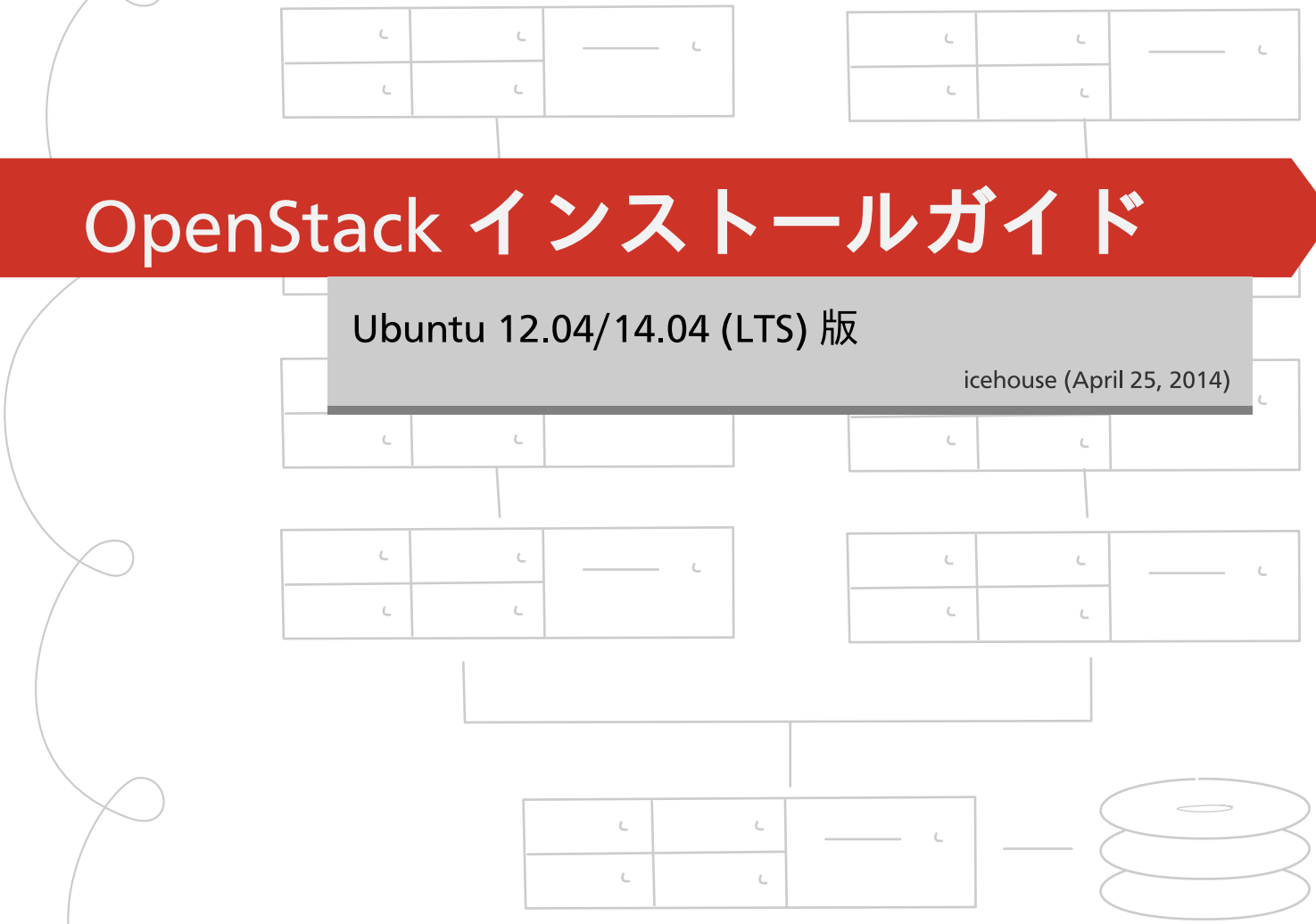
OpenStack インストールガイド

Ubuntu 12.04/14.04 (LTS) 版

icehouse (April 25, 2014)



BUILT FOR

openstack™

CLOUD SOFTWARE

openstack™

docs.openstack.org

OpenStack インストールガイド Ubuntu 12.04/14.04 (LTS) 版

[FAMILY Given]

icehouse (2014-04-25)

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, Orchestration, and Database. You can install any of these projects separately and configure them stand-alone or as connected entities. This guide walks through an installation by using packages available through Ubuntu 12.04 (LTS) or 14.04 (LTS). 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

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		
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。

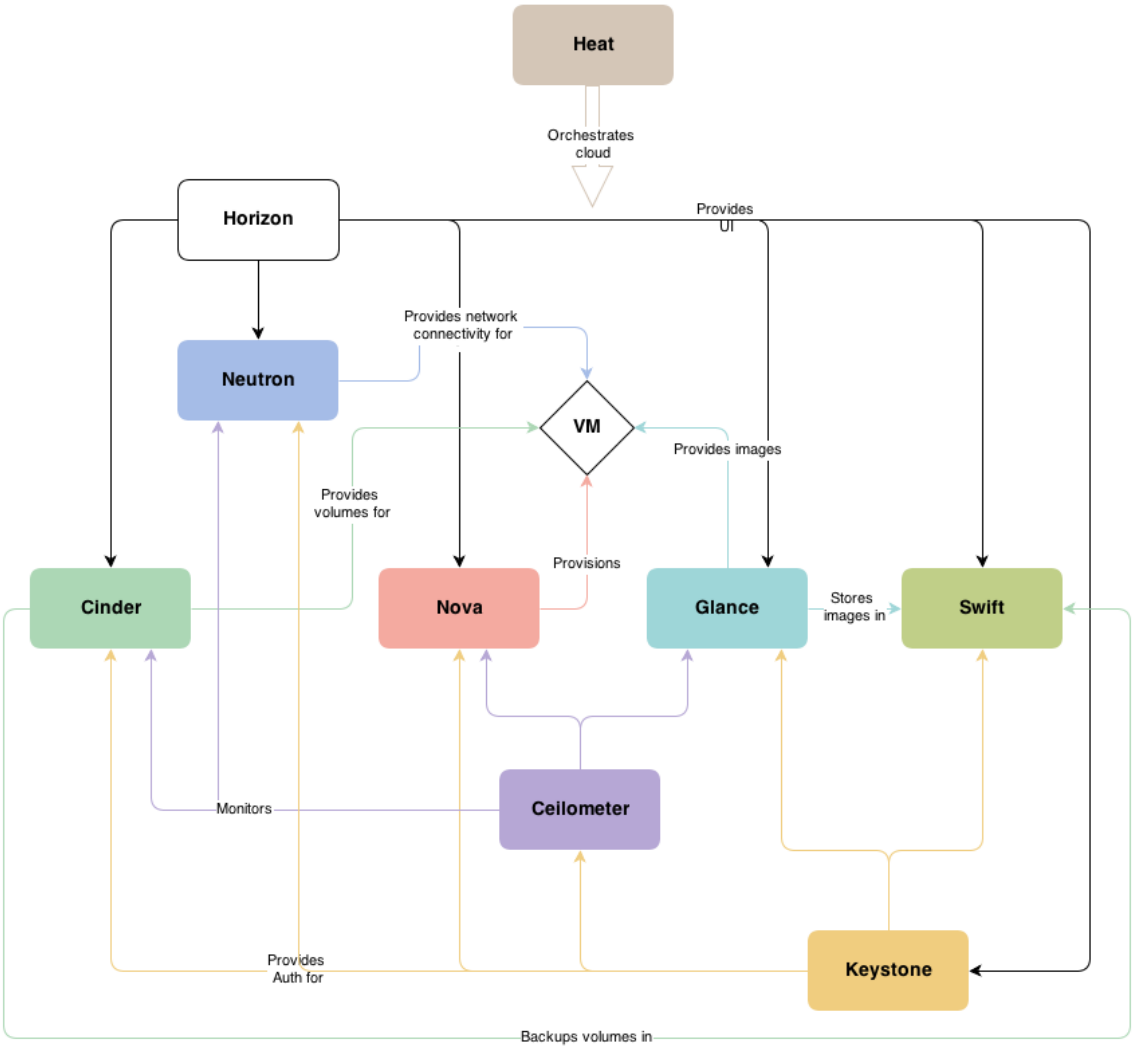
サービス	プロジェクト名	説明
高レベルサービス		
Orchestration	Heat	Orchestrates multiple composite cloud applications by using either the native HOT template format or the AWS CloudFormation template format, through both an OpenStack-native REST API and a CloudFormation-compatible Query API.
Database Service	Trove	Provides 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:

- OpenStack Networking (Neutron) を持つ 3 ノードアーキテクチャー。図 1.2「OpenStack Networking (Neutron) を持つ 3 ノードアーキテクチャー」 [4]を参照してください。

  - The basic controller node runs the Identity service, Image Service, management portions of Compute and Networking, Networking plug-in, and the dashboard. It also includes supporting services such as a database, message broker, and Network Time Protocol (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 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.

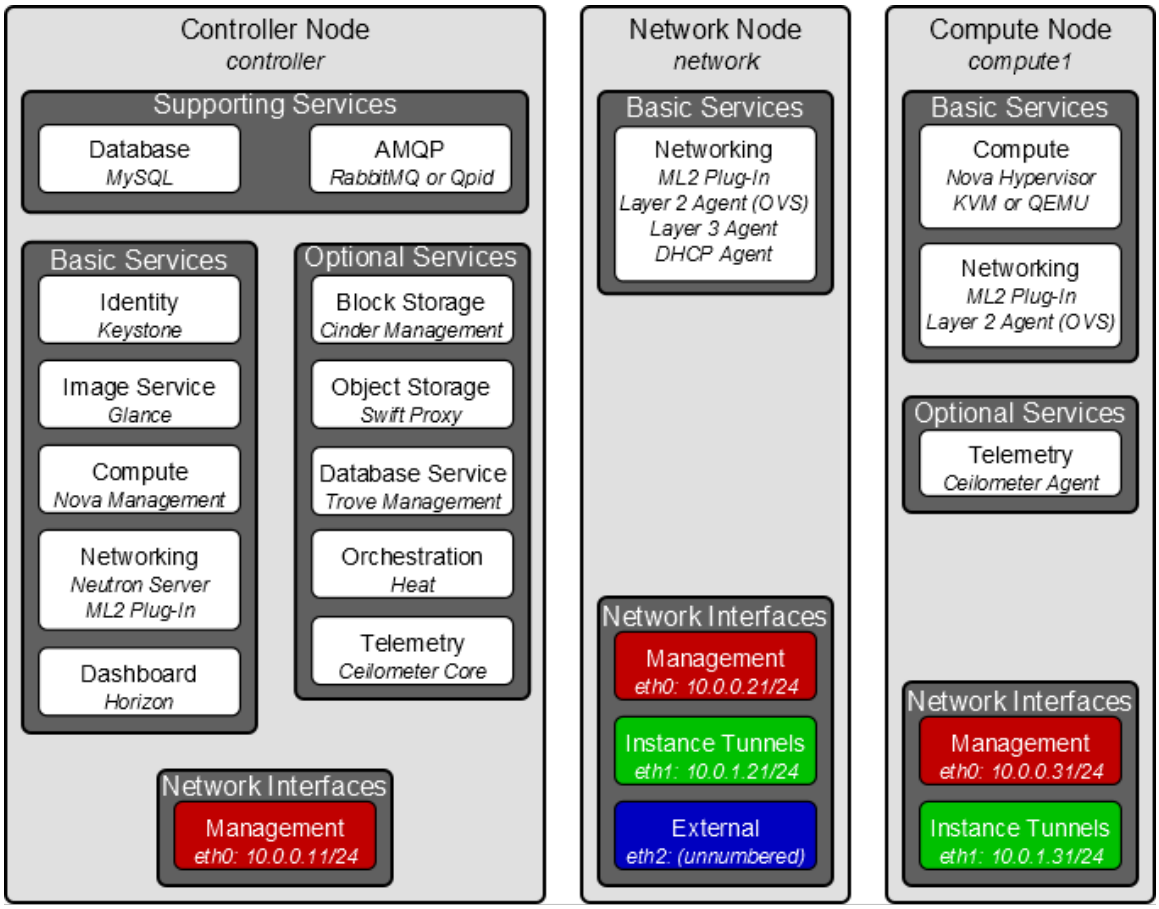
  - The compute node runs the hypervisor portion of Compute, which operates tenant virtual machines or instances. By default Compute uses KVM as the hypervisor. The compute node also runs the Networking plug-in and layer 2 agent which operate tenant networks and implement security groups. You can run more 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 (nova-network)」 [67] 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 OpenStack Networking (Neutron) を持つ 3 ノードアーキテクチャー



- Two-node architecture with legacy networking (nova-network). See 図1.3「Two-node architecture with legacy networking (nova-network)」 [5].

  - 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 a database, message broker, 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.

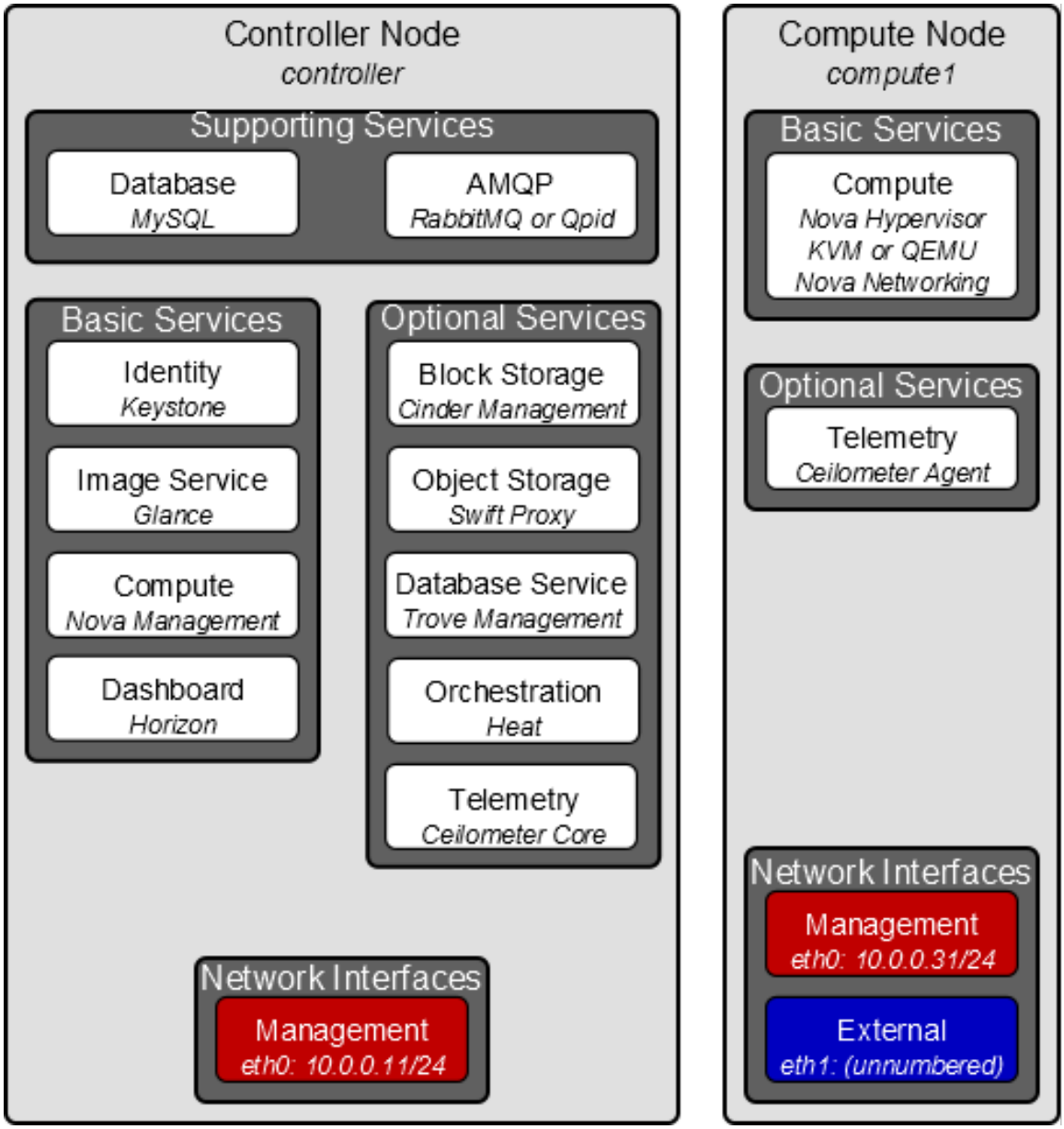
  - The basic compute node runs the hypervisor portion of Compute, which operates tenant virtual machines or instances. By default, Compute uses KVM as the hypervisor. Compute also provisions and operates tenant networks and implements security groups. You can run more than one compute node.

    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 「OpenStack 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1.3 Two-node architecture with legacy networking (nova-network)



第2章 環境の基本設定

目次

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 「Object Storage のシステム要件」 [84]. 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

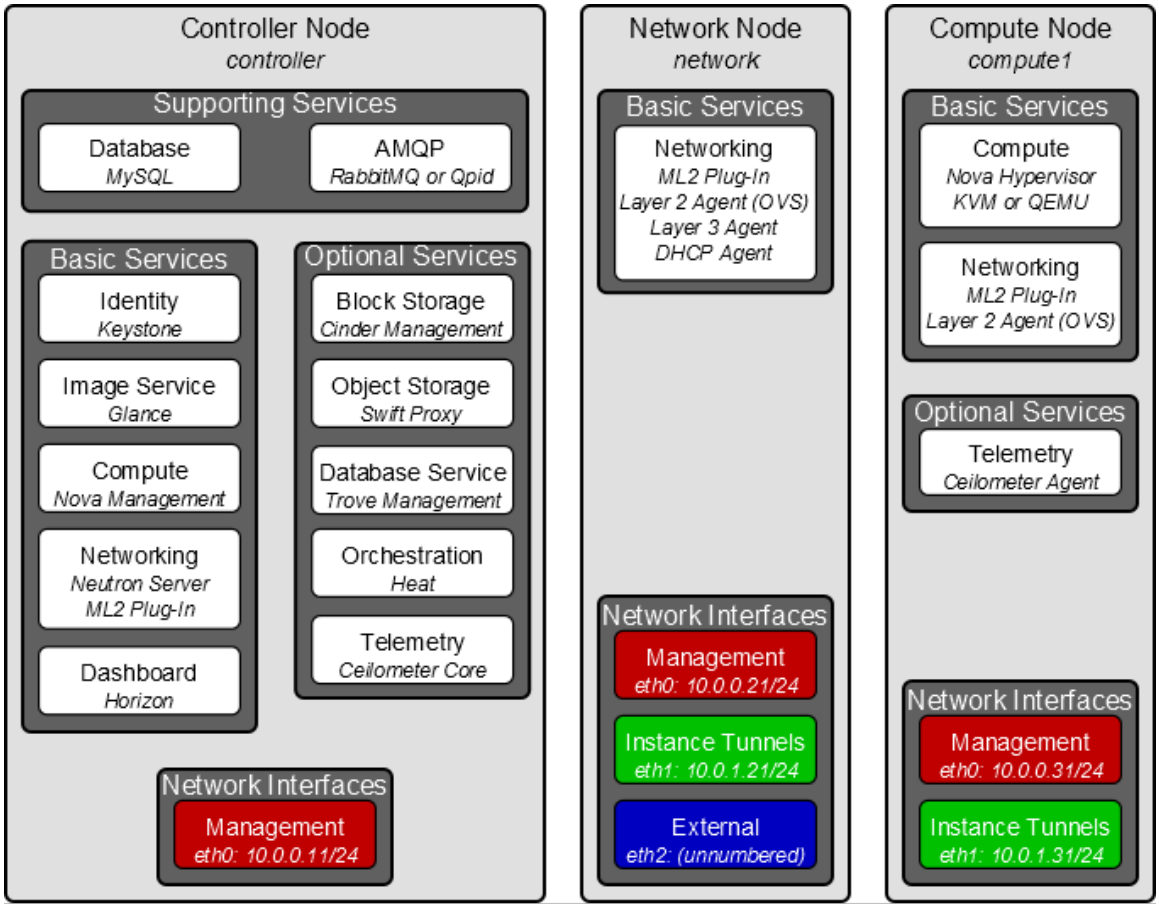
導入するアーキテクチャーに合わせて、各ノードにオペレーティングシステムをインストールした後、ネットワークインターフェースを設定する必要があります。すべての自動ネットワーク管理ツールを無効化し、お使いのディストリビューションに合わせて適切な設定ファイルを手動で編集することを推奨します。お使いのディストリビューションでネットワークを設定する方法に関する詳細は、ドキュメントを参照してください。

Proceed to network configuration for the example OpenStack Networking (neutron) or legacy networking (nova-network) architecture.

OpenStack Networking (neutron)

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 OpenStack Networking (Neutron) を持つ 3 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

# network
10.0.0.21       network

# compute1
10.0.0.31       compute1
```

警告

127.0.1.1 から始まる行を削除するかコメントアウトする必要があります。

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. インスタンスのトンネルインターフェースを設定します。

   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/network/interfaces ファイルを編集し、以下の内容を含めます。

   ```
   # The external network interface
   auto eth2
   iface eth2 inet manual
           up ip link set dev $IFACE up
           down ip link set dev $IFACE down
   ```

4. ネットワークを再起動します。

```
# service networking stop && service networking start
```

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
```

警告

127.0.1.1 から始まる行を削除するかコメントアウトする必要があります。

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. インスタンスのトンネルインターフェースを設定します。

   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
```

警告

127.0.1.1 から始まる行を削除するかコメントアウトする必要があります。

接続性の検証

続行する前に、インターネットとノード間のネットワーク接続性を検証することを推奨します。

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:

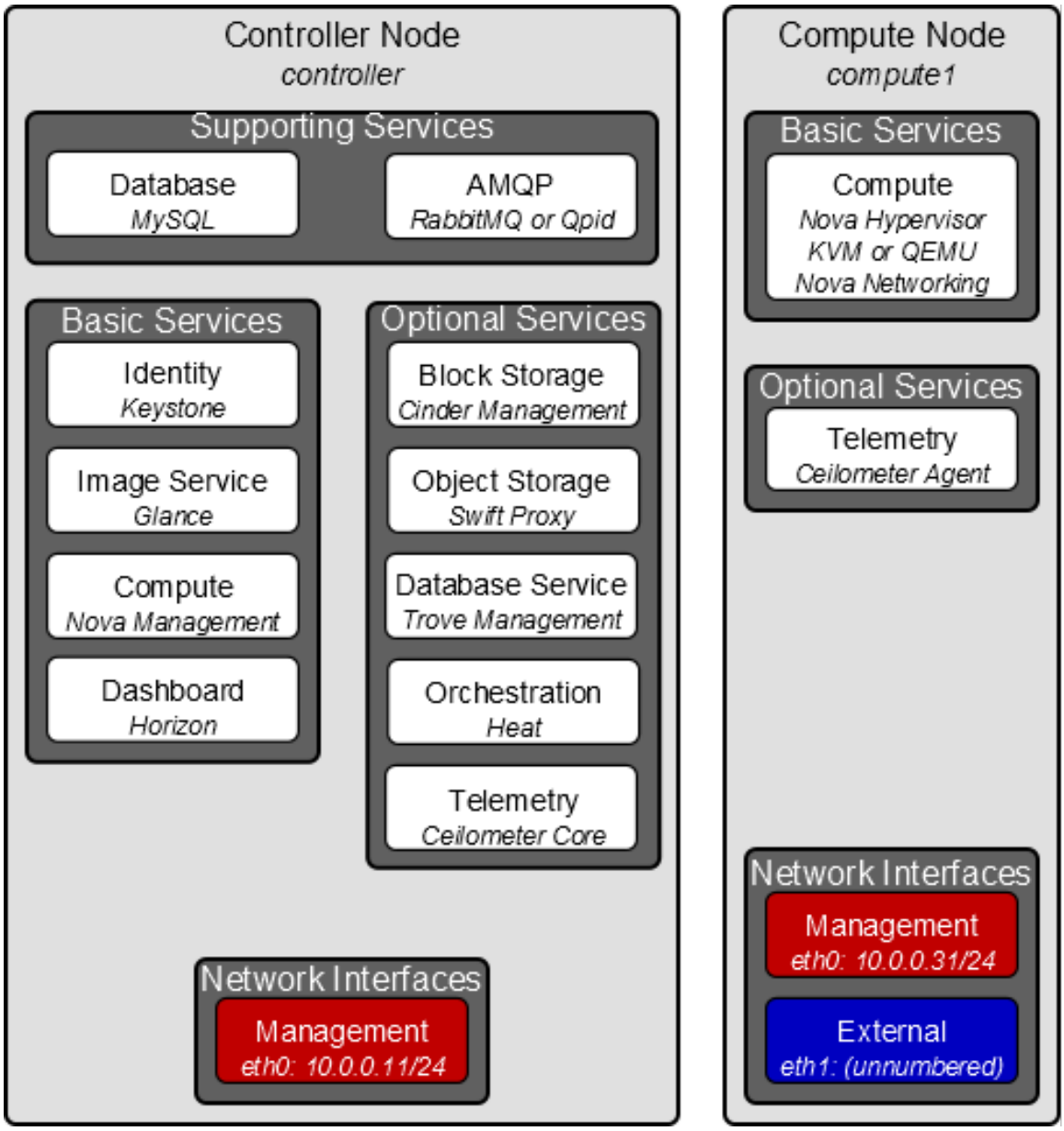
```
# ping -c 4 10.0.1.21
PING 10.0.1.21 (10.0.1.21) 56(84) bytes of data.
64 bytes from 10.0.1.21 (10.0.1.21): icmp_seq=1 ttl=64 time=0.263 ms
64 bytes from 10.0.1.21 (10.0.1.21): icmp_seq=2 ttl=64 time=0.202 ms
64 bytes from 10.0.1.21 (10.0.1.21): icmp_seq=3 ttl=64 time=0.203 ms
64 bytes from 10.0.1.21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

The example architecture with legacy networking (nova-network) requires a 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 Two-node architecture with legacy networking (nova-network)



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
```

警告

127.0.1.1 から始まる行を削除するかコメントアウトする必要があります。

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。

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

```
# The external network interface
auto eth1
iface eth1 inet manual
        up ip link set dev $IFACE up
        down ip link set dev $IFACE down
```

3. ネットワークを再起動します。

```
# service networking stop && service networking start
```

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
```

警告

127.0.1.1 から始まる行を削除するかコメントアウトする必要があります。

接続性の検証

続行する前に、インターネットとノード間のネットワーク接続性を検証することを推奨します。

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:

```
# apt-get install ntp
```

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

Password name	Description
RABBIT_PASS	Password of user guest of RabbitMQ
KEYSTONE_DBPASS	Database password of Identity service
DEMO_PASS	Password of user demo
ADMIN_PASS	Password of user admin
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。

```
# apt-get install python-mysqldb mysql-server
```

注記

サーバーパッケージをインストールするとき、データベースの root パスワードを入力するよう求められます。強いパスワードを選択し、それを覚えておきます。

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/mysql/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
```

変更を適用するために MySQL サービスを再起動します。

```
# service mysql restart
```

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
```

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:

```
# apt-get install python-mysqldb
```

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。

Icehouse 用 Ubuntu Cloud Archive の使用法

Ubuntu Cloud Archive は、Ubuntu の安定サポートバージョンで OpenStack の最新版をインストールできるようにするための特別なリポジトリです。

注記

Icehouse は 14.04 のメインリポジトリにあります。この手順は不要です。

1. Install the Ubuntu Cloud Archive for Icehouse:

```
# apt-get install python-software-properties
# add-apt-repository cloud-archive:icehouse
```

2. Update the package database and upgrade your system:

```
# apt-get update
# apt-get dist-upgrade
```

3. If you intend to use OpenStack Networking with Ubuntu 12.04, you should install a backported Linux kernel to improve the stability of your system. This installation is not needed if you intend to use the legacy networking service.

   Install the Ubuntu 13.10 backported kernel:

```
# apt-get install linux-image-generic-lts-saucy linux-headers-generic-lts-saucy
```

4. Reboot the system for all changes to take effect:

```
# reboot
```

メッセージングサーバー

OpenStack uses a message broker to coordinate operations and status information among services. The message broker service typically runs on the controller node. OpenStack supports several message brokers including RabbitMQ, Qpid, and ZeroMQ. However, most distributions that package OpenStack support a particular message broker. This guide covers the message broker supported by each distribution. If you prefer to implement a different message broker, consult the documentation associated with it.

- RabbitMQ
- Qpid
- ZeroMQ

To install the message broker service

- Ubuntu and Debian use RabbitMQ.

```
# apt-get install rabbitmq-server
```

To configure the message broker service

- The message broker creates a default account that uses guest for the username and password. To simplify installation of your test environment, we recommend that you use this account, but change the password for it.

  Run the following command:

  RABBIT_PASS を適切なパスワードに置き換えます。

```
# rabbitmqctl change_password guest RABBIT_PASS
```

You must configure the rabbit_password key in the configuration file for each OpenStack service that uses the message broker.

注記

For production environments, you should create a unique account with suitable password. For more information on securing the message broker, see the documentation.

If you decide to create a unique account with suitable password for your test environment, you must configure the rabbit_userid and rabbit_password keys in the configuration file of each OpenStack service that uses the message broker.

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。

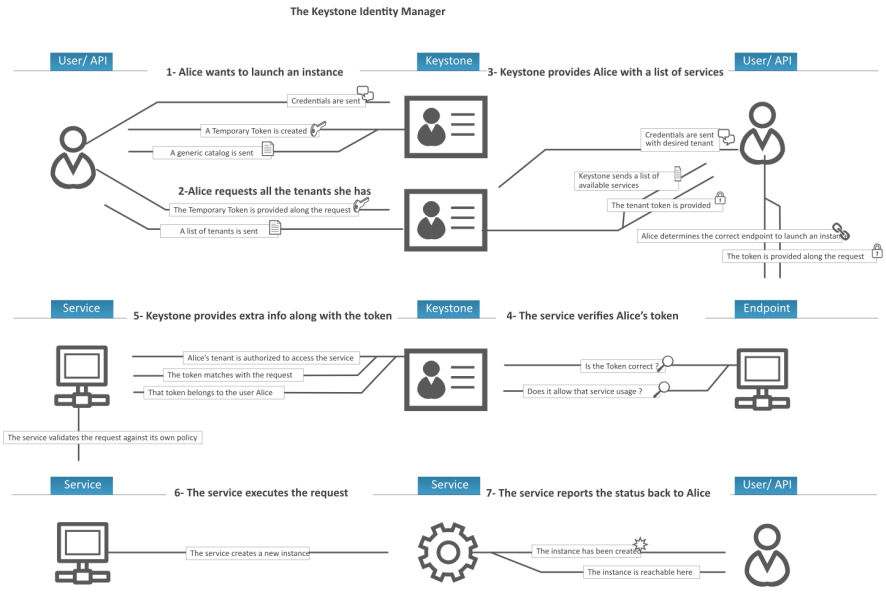
認証
: ユーザーの同一性を確認する動作。Identity Service は、ユーザーに提供された一組のクレデンシャルを検証することにより、送られてきたリクエストを確認します。

  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。

```
# apt-get install keystone
```

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。

   /etc/keystone/keystone.conf を編集し、[database] セクションを変更します。

```
...
[database]
# The SQLAlchemy connection string used to connect to the database
connection = mysql://keystone:KEYSTONE_DBPASS@controller/keystone
...
```

3. Ubuntu パッケージはデフォルトで SQLite データベースを作成します。誤って使用されないように、/var/lib/keystone/ ディレクトリに作成された keystone.db ファイルを削除します。

```
# rm /var/lib/keystone/keystone.db
```

4. 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
```

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 "keystone-manage db_sync" keystone
```

6. 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:

```
# openssl rand -hex 10
```

   /etc/keystone/keystone.conf を編集し、[DEFAULT] セクションを変更します。ADMIN_TOKEN をコマンドの結果で置き換えます。

```
[DEFAULT]
# A "shared secret" between keystone and other openstack services
admin_token = ADMIN_TOKEN
...
```

7. ログディレクトリを設定します。/etc/keystone/keystone.conf ファイルを編集し、[DEFAULT] セクションを更新します。

```
[DEFAULT]
...
log_dir = /var/log/keystone
```

8. Identity Service を再起動します。

```
# service keystone restart
```

9. 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。

   - 1 時間ごとに期限切れトークンを削除し、出力を /var/log/keystone/keystone-tokenflush.log に記録するために、以下のコマンドを実行します。

```
# (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/crontabs/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」 [24] 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. コマンドラインの使用を簡単にするために、お使いの環境で --os-* 変数を設定することもできます。admin クレデンシャルと admin エンドポイントを用いて admin-openrc.sh ファイルをセットアップします。

   ```
   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. admin-openrc.sh ファイルが正しく設定されていることを検証します。同じコマンドを --os-* 引数なしで実行します。

```
$ 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 [30] 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.

残りのインストールを完了するために、クライアントツールをインストールする必要があります。

ユーザーと同じように使用するために、サーバーではなく、デスクトップでクライアントを設定します。

概要

You can use the OpenStack command-line clients to run simple commands that make API calls. You can run these commands from the command line or in scripts to automate tasks. If you provide OpenStack credentials, you can run these commands on any computer.

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。

These open-source Python clients run on Linux or Mac OS X systems and are easy to learn and use. Each OpenStack service has its own command-line client. On some client commands, you can specify a debug parameter to show the underlying API request for the command. This is a good way to become familiar with the OpenStack API calls.

The following table lists the command-line client for each OpenStack service with its package name and description.

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	Configure networks for guest servers. This client was previously called quantum.
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。

Install the prerequisite software

The following table lists the software that you need to have to run the command-line clients, and provides installation instructions as needed.

表4.2 前提ソフトウェア

前提	説明
Python 2.6 or later	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	To install the clients on a Linux, Mac OS X, or Microsoft Windows system, use pip. It is easy to use, ensures that you get the latest version of the clients from the Python Package Index, and lets you update or remove the packages later on. 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。 MacOS.

前提	説明

	`# easy_install pip`
	
	Microsoft Windows. Ensure 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 12.04/14.04. 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`
	
	Red Hat Enterprise Linux, CentOS, or Fedora. A packaged version available in RDO enables you to use yum to install the clients, or you can install pip and use it to manage client installation:
	
	`# 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 clients, or you can install pip and use it to manage client installation:
	
	`# zypper install python-pip`
	
	openSUSE 12.3 and later. A packaged version enables you to use rpm or zypper to install the clients. See 「クライアントのインストール」 [32]

クライアントのインストール

When following the instructions in this section, replace PROJECT with the lowercase name of the client to install, such as nova. Repeat for each client. The following values are valid:

- 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 and extensions
- swift - Object Storage API
- trove - Database Service API

The following example shows the command for installing the nova client with pip.

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

Installing with pip

Use pip to install the OpenStack clients on a Linux, Mac OS X, or Microsoft Windows system. It is easy to use and ensures that you get the latest version of the client from the Python Package Index. Also, pip enables you to update or remove a package.

Install each client separately by using the following command:

- Mac OS X または Linux の場合:

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

- Microsoft Windows の場合:

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

Installing from packages

RDO and openSUSE have client packages that can be installed without pip.

On Red Hat Enterprise Linux, CentOS, or Fedora, use yum to install the clients from the packaged versions available in RDO:

```
# yum install python-PROJECTclient
```

For openSUSE, use rpm or zypper to install the clients from the packaged versions available in the Open Build Service:

```
# zypper install python-PROJECT
```

Upgrade or remove clients

To upgrade a client, add the --upgrade option to the pip install command:

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

To remove the a client, run the pip uninstall command:

```
# pip uninstall python-PROJECTclient
```

Set environment variables using the OpenStack RC file

To set the required environment variables for the OpenStack command-line clients, you must create an environment file called an OpenStack rc file, or openrc.sh file. This project-specific environment file contains the credentials that all OpenStack services use.

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。

注記

Defining environment variables using an environment file is not a common practice on Microsoft Windows. Environment variables are usually defined in the Advanced tab of the System Properties dialog box.

OpenStack RC ファイルの作成と読み込み

1. In a text editor, create a file named PROJECT-openrc.sh file and add the following authentication information:

   The following example shows the information for a project called admin, where the OS username is also admin, and the identity host is located at controller.

   ```
   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 which 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 help output of the various client commands. For example, you can override the OS_PASSWORD setting in the PROJECT-openrc.sh file by specifying a password on a keystone command, as follows:

```
$ keystone --os-password PASSWORD service-list
```

Where PASSWORD is your password.

openrc.sh ファイルの作成

「OpenStack RC ファイルの作成と読み込み」 [34]で説明したように、「ユーザー、プロジェクト、ロールの定義」 [25]にあるクレデンシャルを使用し、以下の PROJECT-openrc.shファイルを作成します。

- 管理ユーザー用の admin-openrc.sh
- 一般ユーザー用の demo-openrc.sh:

```
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

OpenStack Image Service により、ユーザーが仮想マシンイメージを検索、登録、取得できるようになります。Glance プロジェクトとしても知られている Image Service は、仮想マシンイメージを問い合わせ、実際のイメージを取得できるよう、REST API を提供します。Image Service 経由で利用可能な仮想マシンイメージは、単なるファイルシステムから OpenStack Object Storage のようなオブジェクトストレージシステムまで、さまざまな場所に保存できます。

重要

簡単のため、このガイドは Image Service が file バックエンドを使用するよう設定します。これは、Image Service にアップロードされたイメージがこのサービスをホストしているシステムにあるディレクトリに保存されることを意味します。このディレクトリはデフォルトで /var/lib/glance/images/ です。

続行する前に、仮想マシンイメージとスナップショットを保存するために、このディレクトリに十分な空き容量がシステムにあることを確認してください。最小限で、検証環境で Image Service により使用するために、数ギガバイトの容量が利用可能であるべきです。他のバックエンドの要件を確認するために、設定リファレンスを参照してください。

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」 [2]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。

```
# apt-get install glance python-glanceclient
```

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。

   /etc/glance/glance-api.conf と /etc/glance/glance-registry.conf の [database] セクションを編集します。

```
...
[database]
connection = mysql://glance:GLANCE_DBPASS@controller/glance
```

3. Image Service がメッセージブローカーを使用するよう設定します。

   - /etc/glance/glance-api.conf ファイルを編集し、以下のキーを [DEFAULT] セクションに追加します。

     RABBIT_PASS を RabbitMQ の guest アカウント用に選んだパスワードで置き換えます。

```
[DEFAULT]
...
rpc_backend = rabbit
rabbit_host = controller
rabbit_password = RABBIT_PASS
```

4. Ubuntu パッケージはデフォルトで SQLite データベースを作成します。誤って使用されないように、/var/lib/glance/ に作成された glance.sqlite ファイルを削除します。

```
# rm /var/lib/glance/glance.sqlite
```

5. 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';
```

6. 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
```

7. Image Service が Identity Service で認証するために使用する glance 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=glance --pass=GLANCE_PASS ¥
  --email=glance@example.com
$ keystone user-role-add --user=glance --tenant=service --role=admin
```

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

   /etc/glance/glance-api.conf ファイルと /etc/glance/glance-registry.conf ファイルを編集します。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。

   a. 以下のキーを [keystone_authtoken] セクションに追加または修正します。

```
[keystone_authtoken]
auth_uri = http://controller:5000
auth_host = controller
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = glance
admin_password = GLANCE_PASS
```

   b. 以下のキーを [paste_deploy] セクションに修正します。

```
[paste_deploy]
...
flavor = keystone
```

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

```
$ 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
```

10. 新しい設定を用いて glance サービスを再起動します。

```
# service glance-registry restart
# service glance-api restart
```

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 client. A set of command-line interpreter commands for managing cloud resources. Though not an OpenStack module, you can configure nova-api to support this EC2 interface. For more information, see the Eucalyptus 3.4 Documentation.

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。

```
# apt-get install nova-api nova-cert nova-conductor nova-consoleauth ¥
  nova-novncproxy 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。

   Edit the [database] section in the /etc/nova/nova.conf file, adding it if necessary, to modify this key:

```
[database]
connection = mysql://nova:NOVA_DBPASS@controller/nova
```

3. /etc/nova/nova.conf ファイルの [DEFAULT] 設定グループにこれらの設定キーを設定することにより、Compute サービスが RabbitMQ メッセージブローカーを使用するよう設定します。

```
[DEFAULT]
...
rpc_backend = rabbit
rabbit_host = controller
rabbit_password = RABBIT_PASS
```

4. my_ip、vncserver_listen、vncserver_proxyclient_address 設定オプションをコントローラーノードの管理インターフェース IP アドレスに設定します。

   /etc/nova/nova.conf ファイルを編集し、[DEFAULT] セクションにこれらの行を追加します。

```
[DEFAULT]
...
my_ip = 10.0.0.11
vncserver_listen = 10.0.0.11
vncserver_proxyclient_address = 10.0.0.11
```

5. By default, the Ubuntu packages create an SQLite database. Delete the nova.sqlite file created in the /var/lib/nova/ directory so that it does not get used by mistake:

```
# rm /var/lib/nova/nova.sqlite
```

6. 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';
```

7. 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
```

8. 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
```

9. 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。

   このキーを追加するために /etc/nova/nova.conf ファイルの [DEFAULT] セクションを編集します。

```
[DEFAULT]
...
auth_strategy = keystone
```

   これらのキーを [keystone_authtoken] セクションに追加します。

```
[keystone_authtoken]
...
auth_uri = http://controller:5000
auth_host = controller
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = nova
admin_password = NOVA_PASS
```

10. 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
```

11. Compute サービスを再起動します。

```
# service nova-api restart
# service nova-cert restart
# service nova-consoleauth restart
# service nova-scheduler restart
# service nova-conductor restart
# service nova-novncproxy restart
```

12. 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 [6] 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」 [10]

     If you run legacy networking (nova-network), 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」 [15].

   - 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)」 [17]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」 [19] 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。

   このコマンドを実行します。

```
# apt-get install nova-compute-kvm python-guestfs
```

When prompted to create a supermin appliance, respond yes.

3. For security reasons, the Linux kernel is not readable by normal users which restricts hypervisor services such as qemu and libguestfs. For details, see this bug. To make the current kernel readable, run:

```
# dpkg-statoverride  --update --add root root 0644 /boot/vmlinuz-$(uname -r)
```

To also enable this override for all future kernel updates, create the file /etc/kernel/postinst.d/statoverride containing:

```
#!/bin/sh
version="$1"
# passing the kernel version is required
[ -z "${version}" ] && exit 0
dpkg-statoverride --update --add root root 0644 /boot/vmlinuz-${version}
```

Remember to make the file executable:

```
# chmod +x /etc/kernel/postinst.d/statoverride
```

4. /etc/nova/nova.conf 設定ファイルを編集し、これらの行を適切なセクションに追加します。

```
[DEFAULT]
...
auth_strategy = keystone
...
[database]
# The SQLAlchemy connection string used to connect to the database
connection = mysql://nova:NOVA_DBPASS@controller/nova

[keystone_authtoken]
auth_uri = http://controller:5000
auth_host = controller
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = nova
admin_password = NOVA_PASS
```

5. /etc/nova/nova.conf ファイルの [DEFAULT] 設定グループにこれらの設定キーを設定することにより、Compute サービスが RabbitMQ メッセージブローカーを使用するよう設定します。

```
[DEFAULT]
...
rpc_backend = rabbit
rabbit_host = controller
rabbit_password = RABBIT_PASS
```

6. 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。

/etc/nova/nova.conf を編集し、以下のキーを [DEFAULT] セクションに追加します。

```
[DEFAULT]
...
my_ip = 10.0.0.31
vnc_enabled = True
vncserver_listen = 0.0.0.0
vncserver_proxyclient_address = 10.0.0.31
novncproxy_base_url = http://controller:6080/vnc_auto.html
```

7. Image Service を実行するホストを指定します。/etc/nova/nova.conf ファイルを編集し、これらの行を [DEFAULT] セクションに追加します。

```
[DEFAULT]
...
glance_host = controller
```

8. 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 を返したならば、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。この項目を変更するために /etc/nova/nova-compute.conf ファイルの [libvirt] セクションを編集します。

```
[libvirt]
...
virt_type = qemu
```

9. パッケージにより作成された SQLite データベースを削除します。

```
# rm /var/lib/nova/nova.sqlite
```

10. Compute Service を再起動します。

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

第7章 Networking Service の追加

目次

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.

OpenStack Networking (neutron)

Networking concepts

OpenStack Networking (neutron) manages all of the networking facets for the Virtual Networking Infrastructure (VNI) and the access layer aspects of the Physical Networking Infrastructure (PNI) in your OpenStack environment. OpenStack Networking allows tenants to create advanced virtual network topologies including services such as firewalls, load balancers, and virtual private networks (VPNs).

Networking provides the following object abstractions: networks, subnets, and routers. Each has functionality that mimics its physical counterpart: networks contain subnets, and routers route traffic between different subnet and networks.

Any given Networking set up has at least one external network. This network, unlike the other networks, is not merely a virtually defined network. Instead, it represents the view into a slice of the external network that is accessible outside the OpenStack installation. IP addresses on the Networking external network are accessible by anybody physically on the outside network. Because this network merely represents a slice of the outside network, DHCP is disabled on this network.

In addition to external networks, any Networking set up has one or more internal networks. These software-defined networks connect directly to the VMs. Only the VMs on any given internal network, or those on subnets connected through interfaces to a similar router, can access VMs connected to that network directly.

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。

Networking also supports security groups. Security groups enable administrators to define firewall rules in groups. A VM can belong to one or more security groups, and Networking applies the rules in those security groups to block or unblock ports, port ranges, or traffic types for that VM.

Each plug-in that Networking uses has its own concepts. While not vital to operating Networking, understanding these concepts can help you set up Networking. All Networking installations use a core plug-in and a security group plug-in (or just the No-Op security group plug-in). Additionally, Firewall-as-a-service (FWaaS) and Load-balancing-as-a-service (LBaaS) plug-ins are available.

Modular Layer 2 (ML2) プラグイン

コントローラーノードの設定

前提

Before you configure OpenStack Networking (neutron), 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. neutron サービスを作成します。

```
$ 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

- `# apt-get install neutron-server neutron-plugin-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. Networking がデータベースに使用するよう設定します。

   - /etc/neutron/neutron.conf ファイルを編集し、以下のキーを [database] セクションに追加します。

     NEUTRON_PASS をデータベース用に選んだパスワードで置き換えます。

     ```
     [database]
     ...
     connection = mysql://neutron:NEUTRON_DBPASS@controller/neutron
     ```

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

   - Edit the /etc/neutron/neutron.conf file and add the following key to the [DEFAULT] section:

     ```
     [DEFAULT]
     ...
     auth_strategy = keystone
     ```

     Add the following keys to the [keystone_authtoken] section:

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

     ```
     [keystone_authtoken]
     ...
     auth_uri = http://controller:5000
     auth_host = controller
     auth_protocol = http
     auth_port = 35357
     admin_tenant_name = service
     admin_user = neutron
     admin_password = NEUTRON_PASS
     ```

3. Configure Networking to use the message broker:

- Edit the /etc/neutron/neutron.conf file and add the following keys to the [DEFAULT] section:

  RABBIT_PASS を RabbitMQ の guest アカウント用に選んだパスワードで置き換えます。

```
[DEFAULT]
...
rpc_backend = neutron.openstack.common.rpc.impl_kombu
rabbit_host = controller
rabbit_password = RABBIT_PASS
```

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

   Replace SERVICE_TENANT_ID with the service tenant identifier (id) in the Identity service and NOVA_PASS with the password you chose for the nova user in the Identity service.

   - Edit the /etc/neutron/neutron.conf file and add the following keys to the [DEFAULT] section:

```
[DEFAULT]
...
notify_nova_on_port_status_changes = True
notify_nova_on_port_data_changes = True
nova_url = http://controller:8774/v2
nova_admin_username = nova
nova_admin_tenant_id = SERVICE_TENANT_ID
nova_admin_password = NOVA_PASS
nova_admin_auth_url = http://controller:35357/v2.0
```

注記

To obtain the service tenant identifier (id):

```
$ source admin-openrc.sh
$ keystone tenant-get service
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |          Service Tenant          |
|   enabled   |               True               |
|      id     | f727b5ec2ceb4d71bad86dfc414449bf |
|     name    |             service              |
+-------------+----------------------------------+
```

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

   - Edit the /etc/neutron/neutron.conf file and add the following keys to the [DEFAULT] section:

```
[DEFAULT]
...
core_plugin = ml2
service_plugins = router
allow_overlapping_ips = True
```

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.

- /etc/neutron/plugins/ml2/ml2_conf.ini ファイルを編集します。

  以下のキーを [ml2] セクションに追加します。

```
[ml2]
...
type_drivers = gre
tenant_network_types = gre
mechanism_drivers = openvswitch
```

  以下のキーを [ml2_type_gre] セクションに追加します。

```
[ml2_type_gre]
...
tunnel_id_ranges = 1:1000
```

  [securitygroup] セクションを追加し、そこに以下のキーを追加します。

```
[securitygroup]
...
firewall_driver = neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver
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 Networking.

- Edit the /etc/nova/nova.conf and add the following keys to the [DEFAULT] section:

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

```
[DEFAULT]
...
network_api_class = nova.network.neutronv2.api.API
neutron_url = http://controller:9696
neutron_auth_strategy = keystone
neutron_admin_tenant_name = service
neutron_admin_username = neutron
neutron_admin_password = NEUTRON_PASS
neutron_admin_auth_url = http://controller:35357/v2.0
linuxnet_interface_driver = nova.network.linux_net.LinuxOVSInterfaceDriver
firewall_driver = nova.virt.firewall.NoopFirewallDriver
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. Compute のサービスを再起動します。

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

2. Networking のサービスを再起動します。

```
# service neutron-server restart
```

ネットワークノードの設定

前提

Before you configure OpenStack 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

-
```
# apt-get install neutron-plugin-ml2 neutron-plugin-openvswitch-agent openvswitch-datapath-dkms ¥
  neutron-l3-agent neutron-dhcp-agent
```

注記

Ubuntu installations using Linux kernel version 3.11 or newer do not require the openvswitch-datapath-dkms package.

To configure the Networking common components

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

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

   - Edit the /etc/neutron/neutron.conf file and add the following key to the [DEFAULT] section:

     ```
     [DEFAULT]
     ...
     auth_strategy = keystone
     ```

     Add the following keys to the [keystone_authtoken] section:

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

     ```
     [keystone_authtoken]
     ...
     auth_uri = http://controller:5000
     auth_host = controller
     auth_protocol = http
     auth_port = 35357
     admin_tenant_name = service
     admin_user = neutron
     admin_password = NEUTRON_PASS
     ```

2. Configure Networking to use the message broker:

   - Edit the /etc/neutron/neutron.conf file and add the following keys to the [DEFAULT] section:

     RABBIT_PASS を RabbitMQ の guest アカウント用に選んだパスワードで置き換えます。

     ```
     [DEFAULT]
     ...
     rpc_backend = neutron.openstack.common.rpc.impl_kombu
     rabbit_host = controller
     rabbit_password = RABBIT_PASS
     ```

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

   - Edit the /etc/neutron/neutron.conf file and add the following keys to the [DEFAULT] section:

```
[DEFAULT]
...
core_plugin = ml2
service_plugins = router
allow_overlapping_ips = True
```

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.

- Edit the /etc/neutron/l3_agent.ini file and add the following keys to the [DEFAULT] section:

```
[DEFAULT]
...
interface_driver = neutron.agent.linux.interface.OVSInterfaceDriver
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.

- Edit the /etc/neutron/dhcp_agent.ini file and add the following keys to the [DEFAULT] section:

```
[DEFAULT]
...
interface_driver = neutron.agent.linux.interface.OVSInterfaceDriver
dhcp_driver = neutron.agent.linux.dhcp.Dnsmasq
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. Edit the /etc/neutron/metadata_agent.ini file and add the following keys to the [DEFAULT] section:

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.

```
[DEFAULT]
...
auth_url = http://controller:5000/v2.0
auth_region = regionOne
admin_tenant_name = service
admin_user = neutron
admin_password = NEUTRON_PASS
nova_metadata_ip = controller
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, edit the /etc/nova/nova.conf file and add the following keys to the [DEFAULT] section:

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

```
[DEFAULT]
...
service_neutron_metadata_proxy = true
neutron_metadata_proxy_shared_secret = METADATA_SECRET
```

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

```
# service 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.

- /etc/neutron/plugins/ml2/ml2_conf.ini ファイルを編集します。

  以下のキーを [ml2] セクションに追加します。

```
[ml2]
...
type_drivers = gre
tenant_network_types = gre
mechanism_drivers = openvswitch
```

  以下のキーを [ml2_type_gre] セクションに追加します。

```
[ml2_type_gre]
...
tunnel_id_ranges = 1:1000
```

Add the [ovs] section and the following keys to it:

Replace INSTANCE_TUNNELS_INTERFACE_IP_ADDRESS with the IP address of the instance tunnels network interface on your network node.

```
[ovs]
...
local_ip = INSTANCE_TUNNELS_INTERFACE_IP_ADDRESS
tunnel_type = gre
enable_tunneling = True
```

[securitygroup] セクションを追加し、そこに以下のキーを追加します。

```
[securitygroup]
...
firewall_driver = neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver
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. OVS サービスを再起動します。

```
# service openvswitch-switch restart
```

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

- Networking サービスを再起動します。

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

コンピュートノードの設定

前提

Before you configure OpenStack 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

-
```
# apt-get install neutron-common neutron-plugin-ml2 neutron-plugin-openvswitch-agent ¥
  openvswitch-datapath-dkms
```

注記

Ubuntu installations using Linux kernel version 3.11 or newer do not require the openvswitch-datapath-dkms package.

To configure the Networking common components

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

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

    - Edit the /etc/neutron/neutron.conf file and add the following key to the [DEFAULT] section:

```
[DEFAULT]
...
auth_strategy = keystone
```

      Add the following keys to the [keystone_authtoken] section:

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

```
[keystone_authtoken]
...
auth_uri = http://controller:5000
auth_host = controller
auth_protocol = http
auth_port = 35357
admin_tenant_name = service
admin_user = neutron
admin_password = NEUTRON_PASS
```

2. Configure Networking to use the message broker:

   - Edit the /etc/neutron/neutron.conf file and add the following keys to the [DEFAULT] section:

     RABBIT_PASS を RabbitMQ の guest アカウント用に選んだパスワードで置き換えます。

     ```
     [DEFAULT]
     ...
     rpc_backend = neutron.openstack.common.rpc.impl_kombu
     rabbit_host = controller
     rabbit_password = RABBIT_PASS
     ```

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

   - Edit the /etc/neutron/neutron.conf file and add the following keys to the [DEFAULT] section:

     ```
     [DEFAULT]
     ...
     core_plugin = ml2
     service_plugins = router
     allow_overlapping_ips = True
     ```

     ### 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.

- /etc/neutron/plugins/ml2/ml2_conf.ini ファイルを編集します。

  以下のキーを [ml2] セクションに追加します。

  ```
  [ml2]
  ...
  type_drivers = gre
  tenant_network_types = gre
  mechanism_drivers = openvswitch
  ```

以下のキーを [ml2_type_gre] セクションに追加します。

```
[ml2_type_gre]
...
tunnel_id_ranges = 1:1000
```

Add the [ovs] section and the following keys to it:

Replace INSTANCE_TUNNELS_INTERFACE_IP_ADDRESS with the IP address of the instance tunnels network interface on your compute node.

```
[ovs]
...
local_ip = INSTANCE_TUNNELS_INTERFACE_IP_ADDRESS
tunnel_type = gre
enable_tunneling = True
```

[securitygroup] セクションを追加し、そこに以下のキーを追加します。

```
[securitygroup]
...
firewall_driver = neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver
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. OVS サービスを再起動します。

```
# service openvswitch-switch restart
```

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 Networking.

- Edit the /etc/nova/nova.conf and add the following keys to the [DEFAULT] section:

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

```
[DEFAULT]
...
network_api_class = nova.network.neutronv2.api.API
neutron_url = http://controller:9696
neutron_auth_strategy = keystone
neutron_admin_tenant_name = service
neutron_admin_username = neutron
neutron_admin_password = NEUTRON_PASS
neutron_admin_auth_url = http://controller:35357/v2.0
linuxnet_interface_driver = nova.network.linux_net.LinuxOVSInterfaceDriver
firewall_driver = nova.virt.firewall.NoopFirewallDriver
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. Compute Service を再起動します。

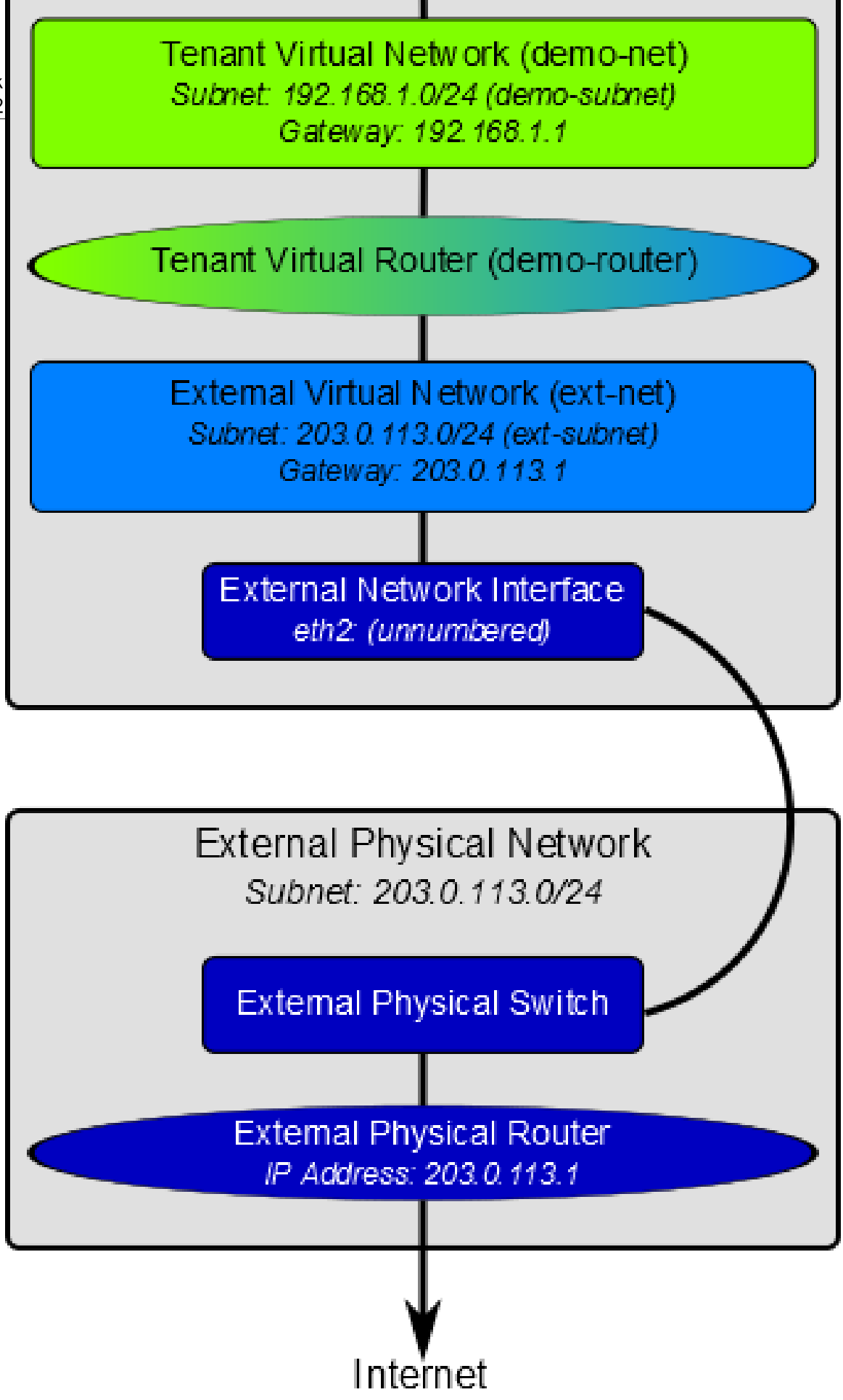
```
# service nova-compute restart
```

2. Open vSwitch (OVS) エージェントを再起動します。

```
# service neutron-plugin-openvswitch-agent 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」 [63]. 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

これらのコマンドをコントローラーノードで実行します。

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

これらのコマンドをコントローラーノードで実行します。

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 (nova-network)

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. Edit the /etc/nova/nova.conf file and add the following keys to the [DEFAULT] section:

```
[DEFAULT]
...
network_api_class = nova.network.api.API
security_group_api = nova
```

2. Compute のサービスを再起動します。

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

- `# apt-get install nova-network nova-api-metadata`

To configure legacy networking

1. Edit the /etc/nova/nova.conf file and add the following keys to the [DEFAULT] section:

```
[DEFAULT]
...
network_api_class = nova.network.api.API
security_group_api = nova
firewall_driver = nova.virt.libvirt.firewall.IptablesFirewallDriver
network_manager = nova.network.manager.FlatDHCPManager
network_size = 254
allow_same_net_traffic = False
multi_host = True
send_arp_for_ha = True
share_dhcp_address = True
force_dhcp_release = True
flat_network_bridge = br100
flat_interface = eth1
public_interface = eth1
```

2. サービスを再起動します。

```
# service nova-network restart
# service nova-api-metadata 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. 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

これらのコマンドをコントローラーノードで実行します。

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。

   NETWORK_CIDR を物理ネットワークのサブネットに置き換えます。

```
$ 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  |
+--------------------------------------+----------+------------------+
```

次の手順

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」［72］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」 [70] 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。

```
# apt-get install apache2 memcached libapache2-mod-wsgi openstack-dashboard
```

Ubuntu ユーザー向け注記

Remove the openstack-dashboard-ubuntu-theme package. This theme prevents translations, several menus as well as the network map from rendering correctly:

```
# apt-get remove --purge openstack-dashboard-ubuntu-theme
```

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

   /etc/openstack-dashboard/local_settings.py 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/memcached.conf 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.py 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。

   /etc/openstack-dashboard/local_settings.py を編集します。

```
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.py を編集し、OPENSTACK_HOST を Identity Service のホスト名に変更します。

```
OPENSTACK_HOST = "controller"
```

5. Apache ウェブサーバーと memcached を起動します。

```
# service apache2 restart
# service memcached restart
```

6. これで Dashboard に http://controller/horizon 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 PRIVILEGES ON dash.* TO 'dash'@'%' IDENTIFIED BY 'DASH_DBPASS';
   mysql> GRANT ALL PRIVILEGES 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。

次の手順

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

OpenStack Block Storage Service はホストマシンに永続的に存在する cinder-* という名前の一連のデーモンプロセスと通信して動作します。単一ノードや複数のノードに渡りバイナリを実行できます。他の OpenStack サービスのように同じノードで実行することもできます。以下のセクションは Block Storage Service のコンポーネントと概念について紹介し、Block Storage Service の設定方法とインストール方法を説明します。

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.

Block Storage Service コントローラーの設定

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 「Block Storage Service ノードの設定」 [79].

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

1. Block Storage Service のために適切なパッケージをインストールします。

```
# apt-get install cinder-api cinder-scheduler
```

2. Configure Block Storage to use your database.

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

```
[database]
...
connection = mysql://cinder:CINDER_DBPASS@controller/cinder
```

注記

In some distributions, the /etc/cinder/cinder.conf file does not include the [database] section header. You must add this section header to the end of the file before you proceed.

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 and add this section for keystone credentials:

```
...
[keystone_authtoken]
auth_uri = http://controller:5000
auth_host = controller
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = cinder
admin_password = CINDER_PASS
```

7. Configure Block Storage to use the RabbitMQ message broker.

   In the [DEFAULT] section in the /etc/cinder/cinder.conf file, set these configuration keys and replace RABBIT_PASS with the password you chose for RabbitMQ:

```
[DEFAULT]
...
rpc_backend = cinder.openstack.common.rpc.impl_kombu
rabbit_host = controller
rabbit_port = 5672
rabbit_userid = guest
rabbit_password = RABBIT_PASS
```

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. Restart the Block Storage services with the new settings:

```
# service cinder-scheduler restart
# service cinder-api restart
```

Block Storage Service ノードの設定

コントローラーノードでサービスを設定した後、Block Storage Service のノードとなる 2 番目のシステムを設定します。このノードはボリュームを取り扱うディスクを有します。

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 [6] 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)」 [17] にある説明に従います。

2. 必要となる LVM パッケージがまだインストールされていなければ、それらをインストールします。

```
# apt-get install lvm2
```

3. 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
```

4. 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。

5. オペレーティングシステムの設定後、Block Storage Service 向けに適切なパッケージをインストールします。

```
# apt-get install cinder-volume
```

6. Edit the /etc/cinder/cinder.conf configuration file and add this section for keystone credentials:

```
...
[keystone_authtoken]
auth_uri = http://controller:5000
auth_host = controller
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = cinder
admin_password = CINDER_PASS
```

7. Configure Block Storage to use the RabbitMQ message broker.

   In the [DEFAULT] configuration section of the /etc/cinder/cinder.conf file, set these configuration keys and replace RABBIT_PASS with the password you chose for RabbitMQ:

   ```
   [DEFAULT]
   ...
   rpc_backend = cinder.openstack.common.rpc.impl_kombu
   rabbit_host = controller
   rabbit_port = 5672
   rabbit_userid = guest
   rabbit_password = RABBIT_PASS
   ```

8. 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:

   ```
   [database]
   ...
   connection = mysql://cinder:CINDER_DBPASS@controller/cinder
   ```

   注記

   In some distributions, the /etc/cinder/cinder.conf file does not include the [database] section header. You must add this section header to the end of the file before you proceed.

9. 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:

   ```
   [DEFAULT]
   ...
   glance_host = controller
   ```

10. Restart the Block Storage services with the new settings:

    ```
    # service cinder-volume restart
    # service tgt restart
    ```

Block Storage のインストールの検証

To verify that the Block Storage is installed and configured properly, create a new volume.

For more information about how to manage volumes, see the OpenStack User Guide.

1. demo-openrc.sh ファイルを読み込みます。

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

2. Use the cinder create command to create a new volume:

```
$ cinder create --display-name myVolume 1
+---------------------+--------------------------------------+
|       Property      |                Value                 |
+---------------------+--------------------------------------+
|     attachments     |                  []                  |
|  availability_zone  |                 nova                 |
|       bootable      |                false                 |
|      created_at     |      2014-04-17T10:28:19.615050      |
| display_description |                 None                 |
|     display_name    |               myVolume               |
|      encrypted      |                False                 |
|          id         | 5e691b7b-12e3-40b6-b714-7f17550db5d1 |
|       metadata      |                  {}                  |
|         size        |                  1                   |
|     snapshot_id     |                 None                 |
|     source_volid    |                 None                 |
|        status       |               creating               |
|     volume_type     |                 None                 |
+---------------------+--------------------------------------+
```

3. Make sure that the volume has been correctly created with the cinder list command:

```
$ cinder list
--------------------------------------+-----------+--------------+------+-------------
+----------+-------------+
|                  ID                  |   Status  | Display Name | Size | Volume Type |
 Bootable | Attached to |
+--------------------------------------+-----------+--------------+------+-------------
+----------+-------------+
| 5e691b7b-12e3-40b6-b714-7f17550db5d1 | available |   myVolume   |  1   |     None    |
 false    |             |
+--------------------------------------+-----------+--------------+------+-------------
+----------+-------------+
```

If the status value is not available, the volume creation failed. Check the log files in the /var/log/cinder/ directory on the controller and volume nodes to get information about the failure.

次の手順

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。

Object Storage のシステム要件

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。

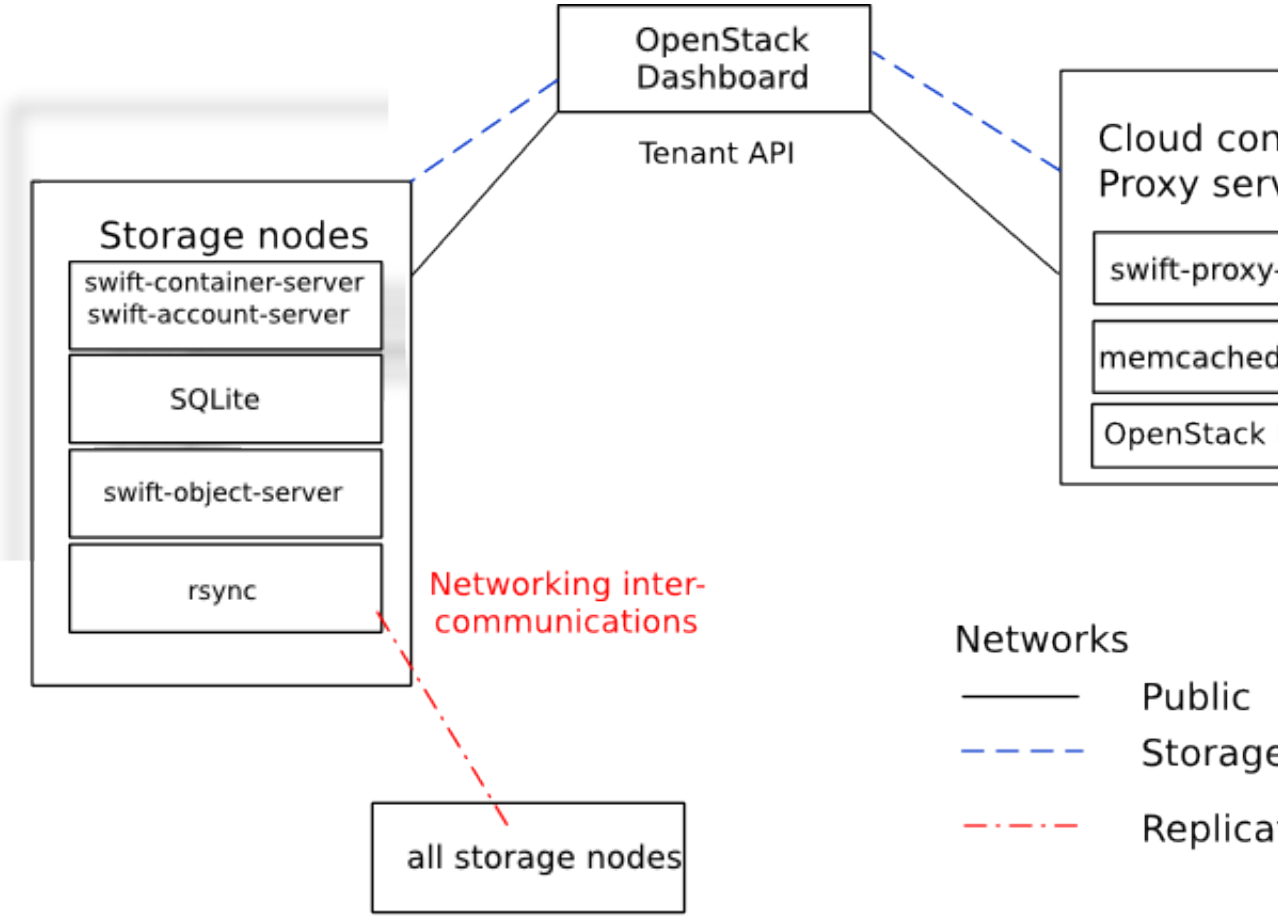
Object Storage 用ネットワークの計画

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。

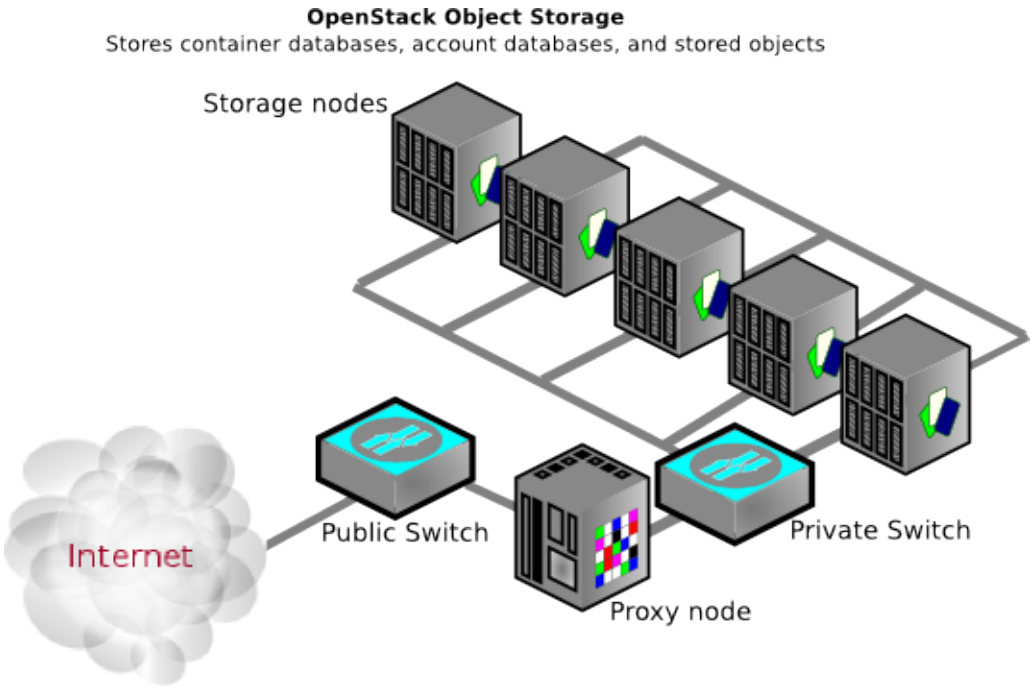
Example of Object Storage installation architecture

- 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. ストレージノードのパッケージをインストールします。

```
# apt-get install swift swift-account swift-container swift-object xfsprogs
```

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/default/rsync で以下の行を編集します。

```
RSYNC_ENABLE=true
```

6. rsync サービスを起動します。

```
# service rsync 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

Object Storage のプロセスは専用のユーザーとグループで動作します。設定オプションにより設定され、swift:swift として参照されます。規定のユーザーは swift です。

1. swift-proxy サービスをインストールします。

```
# apt-get install swift-proxy memcached python-keystoneclient python-swiftclient python-
webob
```

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/memcached.conf ファイルのこの行を編集します。

```
-l 127.0.0.1
```

これを次のように変更します。

```
-l PROXY_LOCAL_NET_IP
```

3. memcached サービスを再起動します。

```
# service memcached restart
```

4. /etc/swift/proxy-server.conf を作成します。

```
[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再起動します。

```
# service swift-proxy restart
```

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 ¥
  swift-object swift-object-replicator swift-object-updater swift-object-auditor ¥
  swift-container swift-container-replicator swift-container-updater swift-container-auditor ¥
  swift-account swift-account-replicator swift-account-reaper swift-account-auditor; do ¥
      service $service start; 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。

次の手順

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。

```
# apt-get install heat-api heat-api-cfn heat-engine
```

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。

   Edit /etc/heat/heat.conf and modify the [database] section:

```
[database]
# The SQLAlchemy connection string used to connect to the database
connection = mysql://heat:HEAT_DBPASS@controller/heat
```

3. By default, the Ubuntu packages create an SQLite database. Delete the heat.sqlite file that was created in the /var/lib/heat/ directory so that it does not get used by mistake:

```
# rm /var/lib/heat/heat.sqlite
```

4. 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';
```

5. 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。

6. Ubuntu パッケージはロギングを正しくセットアップしません。/etc/heat/heat.conf ファイルを編集し、[DEFAULT] セクションを変更します。

```
[DEFAULT]
...
# Print more verbose output (set logging level to INFO instead
# of default WARNING level). (boolean value)
verbose = True
...
# (Optional) The base directory used for relative --log-file
# paths (string value)
log_dir=/var/log/heat
```

7. Orchestration Service が RabbitMQ メッセージブローカーを使用するよう設定します。

   /etc/heat/heat.conf を編集し、[DEFAULT] セクションを変更します。

```
rabbit_host = controller
rabbit_password = RABBIT_PASS
```

8. 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
```

9. 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
```

10. 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
```

11. 新しい設定を用いてサービスを再起動します。

```
# service heat-api restart
# service heat-api-cfn restart
# service heat-engine restart
```

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 |
+--------------------------------------+------------+-----------------+----------------------+
```

次の手順

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。

```
# apt-get install ceilometer-api ceilometer-collector ceilometer-agent-central ¥
  ceilometer-alarm-evaluator ceilometer-alarm-notifier 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。

```
# apt-get install mongodb-server
```

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。

MongoDB サービスを初めて起動するときに、このファイルが作成されるので、この変更を反映するためにサービスを停止して、このファイルを削除する必要があります。

```
# service mongodb stop
# rm /var/lib/mongodb/journal/prealloc.*
# service mongodb start
```

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. Configure MongoDB to make it listen on the controller management IP address. Edit the /etc/mongodb.conf file and modify the bind_ip key:

```
bind_ip = 10.0.0.11
```

4. 設定の変更を適用するために MongoDB のサービスを再起動します。

```
# service mongodb restart
```

5. 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" ]})'
```

6. Telemetry Service がデータベースを使用するよう設定します。

/etc/ceilometer/ceilometer.conf ファイルを編集し、[database] セクションを変更します。

```
[database]
# The SQLAlchemy connection string used to connect to the
# database (string value)
connection = mongodb://ceilometer:CEILOMETER_DBPASS@controller:27017/ceilometer
```

7. 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:

```
# openssl rand -hex 10
```

Edit the /etc/ceilometer/ceilometer.conf file and change the [publisher] section. Replace CEILOMETER_TOKEN with the results of the openssl command:

```
[publisher]
# Secret value for signing metering messages (string value)
metering_secret = CEILOMETER_TOKEN
```

8. RabbitMQ のアクセス権を設定します。

/etc/ceilometer/ceilometer.conf ファイルを編集し、[DEFAULT] セクションを更新します。

```
rabbit_host = controller
rabbit_password = RABBIT_PASS
```

9. ログディレクトリを設定します。

/etc/ceilometer/ceilometer.conf ファイルを編集し、[DEFAULT] セクションを更新します。

```
[DEFAULT]
log_dir = /var/log/ceilometer
```

10. 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
```

11. クレデンシャルを Telemetry Service の設定ファイルに追加します。

    /etc/ceilometer/ceilometer.conf ファイルを編集し、[keystone_authtoken] セクションを変更します。

```
[keystone_authtoken]
auth_host = controller
auth_port = 35357
auth_protocol = http
auth_uri = http://controller:5000
admin_tenant_name = service
admin_user = ceilometer
admin_password = CEILOMETER_PASS
```

    [service_credentials] セクションも設定します。

```
[service_credentials]
os_auth_url = http://controller:5000/v2.0
os_username = ceilometer
os_tenant_name = service
os_password = CEILOMETER_PASS
```

12. 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
```

13. 新しい設定を用いてサービスを再起動します。

```
# service ceilometer-agent-central restart
# service ceilometer-api restart
# service ceilometer-collector restart
# service ceilometer-alarm-evaluator restart
# service ceilometer-alarm-notifier restart
```

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。

```
# apt-get install ceilometer-agent-compute
```

2. /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]
...
instance_usage_audit = True
instance_usage_audit_period = hour
notify_on_state_change = vm_and_task_state
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 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。

   Edit the /etc/ceilometer/ceilometer.conf file and change these lines in the [publisher] section. Replace CEILOMETER_TOKEN with the ceilometer token that you created previously:

```
[publisher]
# Secret value for signing metering messages (string value)
metering_secret = CEILOMETER_TOKEN
```

5. RabbitMQ のアクセス権を設定します。

   /etc/ceilometer/ceilometer.conf ファイルを編集し、[DEFAULT] セクションを更新します。

```
[DEFAULT]
rabbit_host = controller
rabbit_password = RABBIT_PASS
```

6. Identity Service のクレデンシャルを追加します。

   /etc/ceilometer/ceilometer.conf ファイルを編集し、[keystone_authtoken] セクションを変更します。

```
[keystone_authtoken]
auth_host = controller
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = ceilometer
admin_password = CEILOMETER_PASS
```

[service_credentials] セクションも設定します。

```
[service_credentials]
os_auth_url = http://controller:5000/v2.0
os_username = ceilometer
os_tenant_name = service
os_password = CEILOMETER_PASS
```

7. ログディレクトリを設定します。

   /etc/ceilometer/ceilometer.conf ファイルを編集し、[DEFAULT] セクションを更新します。

```
[DEFAULT]
log_dir = /var/log/ceilometer
```

8. 新しい設定を用いてサービスを再起動します。

```
# service ceilometer-agent-compute restart
```

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。

   /etc/glance/glance-api.conf を編集し、[DEFAULT] セクションを変更します。

```
notification_driver = messaging
rpc_backend = rabbit
rabbit_host = controller
rabbit_password = RABBIT_PASS
```

2. 新しい設定を用いて Image サービスを再起動します。

```
# service glance-registry restart
# service glance-api restart
```

Telemetry 用 Block Storage Service エージェントの追加

1. ボリュームのサンプルを取得するために、Block Storage 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。

   Edit /etc/cinder/cinder.conf and add in the [DEFAULT] section on the controller and volume nodes:

```
control_exchange = cinder
notification_driver = cinder.openstack.common.notifier.rpc_notifier
```

2. Restart the Block Storage services with their new settings.

   On the controller node:

```
# service cinder-api restart
# service cinder-scheduler restart
```

   On the volume node:

```
# service 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 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-x86_64" > 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 |
+--------+---------------------+---------------------+-------+------------
+------------+------------+------------+----------+---------------------------
+----------------------------+
```

次の手順

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 Database サービスの追加

目次

クラウドデータベースリソースを作成するために、Database モジュールを使用します。統合プロジェクトの名前は trove です。

警告

本章は作業中です。不正確な情報を含む可能性があり、頻繁に更新されます。

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.

Database サービスのインストール

この手順は、コントローラーノードに Database モジュールをインストールします。

前提. 本章は、すでに OpenStack 環境が動作していること、少なくとも Compute、Image Service、Identity がインストールされていることを仮定しています。

Ubuntu 14.04 のみ

Database モジュールは Ubuntu 14.04 のみで利用可能です。パッケージは 12.04 や Ubuntu Cloud Archive から取得できません。

Database モジュールをコントローラーにインストールする方法:

1. 必要パッケージをインストールします。

```
# apt-get install python-trove python-troveclient python-glanceclient ¥
  trove-common trove-api trove-taskmanager
```

2. OpenStack を準備します。

   a. admin-openrc.sh ファイルを読み込みます。

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

   b. Compute が Identity Service で認証するために使用する trove 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=trove --pass=TROVE_PASS ¥
  --email=trove@example.com
$ keystone user-role-add --user=trove --tenant=service --role=admin
```

3. Edit 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 and set appropriate values for the OpenStack service URLs, logging and messaging configuration, and SQL connections:

   ```
   [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_
   sql_connection = trove:TROVE_DBPASS@controller/trove
   notifier_queue_hostname = controller
   ```

   b. Configure the Database module to use the RabbitMQ message broker by setting the rabbit_password in the [DEFAULT] configuration group of each file:

   ```
   [DEFAULT]
   ...
   rabbit_password = RABBIT_PASS
   ...
   ```

4. Edit the [filter:authtoken] section of the api-paste.ini file so it matches the listing shown below:

   ```
   [filter:authtoken]
   auth_host = controller
   auth_port = 35357
   auth_protocol = http
   admin_user = trove
   admin_password = ADMIN_PASS
   admin_token = ADMIN_TOKEN
   admin_tenant_name = service
   signing_dir = /var/cache/trove
   ```

5. Edit the trove.conf file so it includes appropriate values for the default datastore and network label regex as 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$
   ....
   ```

6. Edit the trove-taskmanager.conf file so it includes the appropriate service credentials required to connect to the OpenStack Compute service as 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_PASS
nova_proxy_admin_tenant_name = service
...
```

7. trove 管理データベースを準備します。

```
$ 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. Database サービスを準備します。

   a. データベースを初期化します。

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

   b. データストアを作成します。使用したいデータベースの種類に応じて、別々のデータストアを作成する必要があります。例えば、MySQL、MongoDB、Cassandra です。この例は、MySQL データベース用のデータストアを作成する方法です。

   ```
   # su -s /bin/sh -c "trove-manage datastore_update mysql ''" trove
   ```

9. trove イメージを作成します。

   使用したいデータベースの種類に応じてイメージを作成します。例えば、MySQL、MongoDB、Cassandra です。

   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, follow these steps on the guest instance you are using to build your image:

   - 以下の行を trove-guestagent.conf に追加します。

   ```
   rabbit_host = controller
   rabbit_password = RABBIT_PASS
   nova_proxy_admin_user = admin
   nova_proxy_admin_pass = ADMIN_PASS
   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.

    この例は MySQL 5.5 データストアを作成する方法です。

    ```
    # 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. 他の OpenStack サービスから使用できるように、Database モジュールを 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=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. Database サービスを再起動します。

```
# service trove-api restart
# service trove-taskmanager restart
# service trove-conductor restart
```

Database サービスのインストールを検証します。

Database サービスがインストールされ、正しく設定されていることを検証するために、Trove コマンドを実行してみます。

1. demo-openrc.sh ファイルを読み込みます。

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

2. Trove インスタンスの一覧を取得します。

```
$ trove list
```

このように出力されます。

```
+----+------+-----------+-------------------+--------+-----------+------+
| 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.

この例は MySQL 5.5 データベースを作成する方法です。

```
$ trove create 名前 2 --size=2 --databases=DBNAME ¥
  --users USER:PASSWORD --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 [35] 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 OpenStack 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 OpenStack 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 |
+--------------------------------------+---------------------+--------+--------+
| acafc7c0-40aa-4026-9673-b879898e1fc2 | 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-0.3.2-x86_64 (acafc7c0-40aa-4026-9673-
b879898e1fc2) |
| 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 |
+--------------------------------------+---------------------+--------+--------+
| acafc7c0-40aa-4026-9673-b879898e1fc2 | 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-0.3.2-x86_64 (acafc7c0-40aa-4026-9673-
b879898e1fc2) |
| 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。

注記

いくつかの OpenStack パッケージはインストール中にユーザー ID を自動的に生成し、割り当てます。これらのケースで、ユーザー ID 値が重要ではありません。ユーザー ID の存在が重要です。

表A.1 予約済みユーザー ID

名前	説明	ID
ceilometer	OpenStack Ceilometer デーモン	パッケージインストール中の割り当て
cinder	OpenStack Cinder デーモン	パッケージインストール中の割り当て
glance	OpenStack Glance デーモン	パッケージインストール中の割り当て
heat	OpenStack Heat デーモン	パッケージインストール中の割り当て
keystone	OpenStack Keystone デーモン	パッケージインストール中の割り当て
neutron	OpenStack Neutron デーモン	パッケージインストール中の割り当て
nova	OpenStack Nova デーモン	パッケージインストール中の割り当て
swift	OpenStack Swift デーモン	パッケージインストール中の割り当て
trove	OpenStack Trove Daemons	パッケージインストール中の割り当て

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
- Installation Guide for Ubuntu 12.04/14.04 (LTS)

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.

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.

Icehouse
: OpenStack の 9 回目リリースのプロジェクト名。

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.

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.

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.

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.