Oracle® Solaris 11.2 での OpenStack のインストールと構成

Part No: E56871-03
2015 年 4 月

ORACLE

Part No: E56871-03

ドキュメントのアクセシビリティについて

オラクルのアクセシビリティについての詳細情報は、Oracle Accessibility ProgramのWeb サイト(http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc)を参照してください。

Oracle Supportへのアクセス

サポートをご契約のお客様には、My Oracle Supportを通して電子支援サービスを提供しています。詳細情報は(http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info)か、聴覚に障害のあるお客様は http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs)を参照してください。

目次

このドキュメントの使用法

- **概要** – Oracle Solaris 11.2 システムに OpenStack をインストールし OpenStack 仮想マシンを配備する方法について説明します。
- **対象読者** – 大規模インストールシステムの管理者。
- **前提知識** – Solaris ネットワークおよび大規模システムの管理。OpenStack の知識が役立ちます。

製品ドキュメントライブラリ

この製品および関連製品のドキュメントとリソースは http://www.oracle.com/pls/topic/lookup?ctx=E56342 で入手可能です。

フィードバック

このドキュメントに関するフィードバックを http://www.oracle.com/goto/docfeedback からお聞かせください。

♦ ♦ ♦ 第 1 章

1 Oracle Solaris 11.2 の OpenStack の概要

この章では、Oracle Solaris 11.2 の OpenStack で Oracle Solaris の機能がどのように使用されるかについて説明します。

この章では、次の内容について説明します。

Oracle Solaris 11.2 の新機能

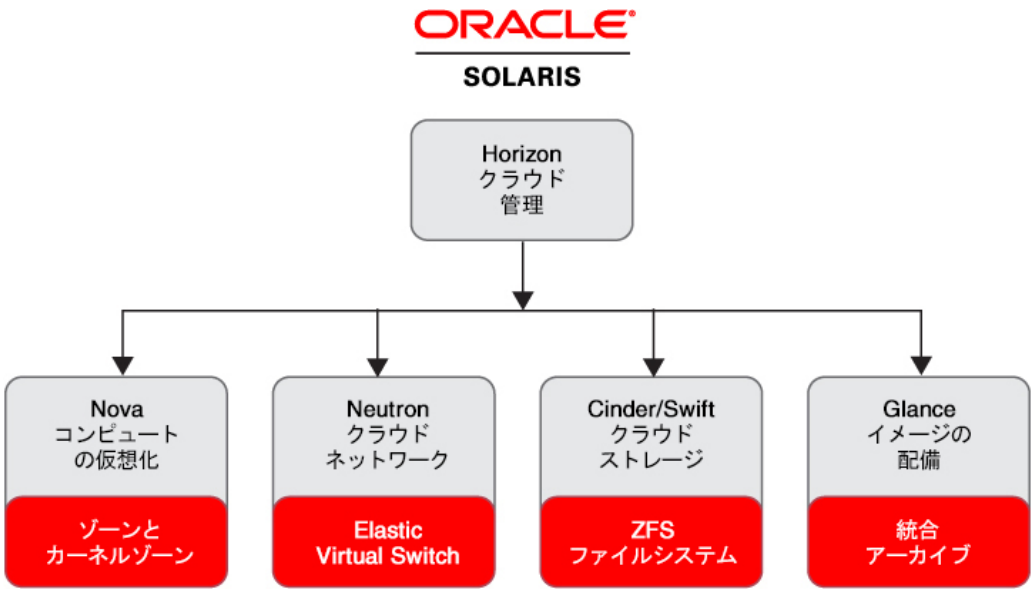
Oracle Solaris 11.2 SRU 10 以降、Juno バージョンの OpenStack がサポートされます。このドキュメントには、Havana または Juno 特有の OpenStack の構成手順が含まれています。

Oracle Solaris への OpenStack 統合のしくみ

Oracle Solaris 11.2 には、Oracle Solaris 11 のコアテクノロジに完全に統合された OpenStack ディストリビューションが含まれています。Oracle Solaris 11.2 の OpenStack では、エンタープライズ対応の IaaS (Infrastructure as a Service) プライベートクラウドを作成できるため、ユーザーは一元化された Web ベースポータルを使用して仮想ネットワークリソースと仮想コンピュートリソースをすばやく作成できます。

次の図は、OpenStack のサービスを実装するために使用される Oracle Solaris 11.2 の機能を示しています。これらの関係は図の下に説明されています。

図 1-1 Oracle Solaris と OpenStack の統合



Oracle Solaris 11.2 では次の OpenStack サービスが提供されます。

- Nova

  Nova コンピュート仮想化サービスは、さまざまな仮想化テクノロジをサポートするクラウドコンピュートファブリックコントローラを提供します。Solaris では、仮想マシン (VM) インスタンスはカーネルゾーンまたは非大域ゾーンです。ゾーンは、仮想化のオーバーヘッドが少ない、スケーラブルな高密度仮想環境です。カーネルゾーンには独立したカーネルバージョンも用意されており、マルチテナントクラウドの場合に望ましい VM インスタンスの独立アップグレードが可能です。

  Oracle Solaris ゾーンの詳細は、Oracle Solaris 11.2 Library のゾーンに関する各種ドキュメントを参照してください。

- Neutron

  Neutron ネットワーク仮想化サービスは、複数の OpenStack システム上にあるほかの OpenStack サービス、および VM インスタンスにネットワーク接続を提供します。Solaris

では、Elastic Virtual Switch (EVS) 機能を介してネットワーク仮想化サービスが提供されます。EVS は、複数の物理サーバーにまたがる仮想スイッチを作成、構成、およびモニターするための単一制御点として機能します。アプリケーションは、クラウドのネットワークトラフィックに優先順位を付ける独自の動作を実行できます。Neutron には、仮想ネットワークを動的に要求および構成するための API が用意されています。これらのネットワークは、Nova VM インスタンスの VNIC などのインタフェースを接続します。

Elastic Virtual Switch の詳細は、『Oracle Solaris 11.2 での仮想ネットワークとネットワークリソースの管理 』の第 5 「エラスティック仮想スイッチについて」を参照してください。

- Cinder

  Cinder ブロックストレージサービスは、OpenStack のブロックストレージボリュームを管理するためのインフラストラクチャーを提供します。Cinder では、ブロック型デバイスを公開することや VM インスタンスにブロック型デバイスを接続することにより、ストレージの拡張、パフォーマンスの向上、およびエンタープライズストレージプラットフォームとの統合が可能です。Solaris では、Cinder はストレージに ZFS を使用し、リモートアクセスに iSCSI またはファイバチャネルを使用します。ZFS は、スナップショット、暗号化、複製解除などの統合データサービスを提供します。Cinder ドライバは、ZFS Storage Appliance でも使用できます。

  ZFS の詳細は、『Oracle Solaris 11.2 での ZFS ファイルシステムの管理 』を参照してください。ZFS Storage Appliance のドキュメントは `https://docs.oracle.com/en/storage/` にあります。

- Swift

  Swift オブジェクトストレージサービスは、OpenStack プロジェクトおよびユーザーに冗長かつスケーラブルなオブジェクトストレージサービスを提供します。Swift は ZFS を使用して任意の非構造化データの保存と取得を行います。また、データには RESTful API を介してアクセスできます。

- Glance

  Glance イメージストアサービスは、仮想マシンのディスクイメージを保存します。これらは VM インスタンスの配備に使用されます。Solaris では、Glance イメージは統合アーカイブです。単純なファイルシステムから OpenStack Swift のようなオブジェクトストレージシステムまで、さまざまな場所にイメージを保存できます。Glance には、イメージメタデータの照会とイメージの取得を可能にする RESTful API があります。

  統合アーカイブを使用すると、規格に準拠したセキュアかつスケーラブルな配備をすばやく行うことができます。同じ統合アーカイブを使用して、ベアメタルシステムと仮想システムのどちらでも配備できます。統合アーカイブを Automated Installer (AI) とともに使用すると、多数のシステムのプロビジョニングをすばやく行うことができます。

詳細については、『Oracle Solaris 11.2 でのシステム復旧とクローン 』を参照してください。AI インストールは、メディアまたはサーバー上で AI イメージを使用して自動インストールを実行する方法です。詳細は、『Oracle Solaris 11.2 システムのインストール 』の第 5 「メディアからブートする自動インストール」を参照してください。

- Horizon

  Horizon は、複数の VM インスタンスをサポートするためにクラウドインフラストラクチャーおよびコンピューティングインフラストラクチャーを管理できる、OpenStack のダッシュボードです。ダッシュボードは OpenStack サービスに対する Web ベースのユーザーインタフェースを提供します。例については、29 ページの「OpenStack ダッシュボードの使用」を参照してください。

- Keystone

  Keystone アイデンティティーサービスは、ユーザー、管理者、および OpenStack サービスの間の認証と承認のサービスを提供します。

- 開発者は、Heat オーケストレーションサービスエンジンを使用して OpenStack インフラストラクチャーの実装を自動化できます。このエンジンは、カスタマイズされた構成を配備するための構成情報とインストール後処理を含むテンプレートに従って動作します。

各 OpenStack サービスは、1 つ以上のサービス管理機能 (SMF) サービスで表されます。たとえば、表3-1「コントローラ、ネットワーク、およびコンピュートノードにインストールされた SMF サービス」 の SMF サービスのリストを見てください。SMF は OpenStack サービスを制御します。たとえば、障害発生時に自動的にサービスを再起動したり、サービス依存関係の完全チェックを実行したりすることにより、サービス起動の精度と効率を向上させます。

Image Packaging System (IPS) により、OpenStack システムを簡単に配備でき、失敗のないアップグレードをすばやく実行できます。ブート環境 (BE) を使用すると、OpenStack システムを更新するときにバックアップ環境を簡単に保持できます。最小化も含めてインストールの柔軟性を提供するために、OpenStack サービスはそれぞれ独自の IPS パッケージで配布されます。各 OpenStack サービスパッケージは、その OpenStack サービス用の一意のユーザーとグループ、およびその OpenStack サービスを管理するための RBAC プロファイルを提供します。

このドキュメントの使用法

このドキュメントでは主に、Solaris とほかのプラットフォームの間で異なっている OpenStack の情報を扱います。Solaris とほかのプラットフォームで同様に動作する機能、および Solaris

とほかのプラットフォームで同様に実行される操作は、通常このドキュメントでは扱われません。

第2章「評価構成のインストール」では、OpenStack を評価するために単一の Solaris システムにすばやくインストールする方法について説明します。完全なインストールは統合アーカイブで配布され、構成の大部分は自動的に実行されます。

第3章「マルチノード Havana OpenStack 構成での複数システムにまたがるインストール」では、OpenStack を 3 つの Solaris システム (コントローラノード、ネットワークノード、コンピュートノード) にインストールして構成する方法について説明します。

第5章「仮想マシンインスタンスの作成」では、VM インスタンスの作成と使用に関する Solaris 固有の情報を提供します。テナントと関連ユーザーの作成などのタスクは、Solaris とほかのプラットフォームで同じなので、説明されていません。

この Oracle Solaris リリースの OpenStack の基になっている OpenStack に関する一般的な情報については、OpenStack のドキュメントサイトにある次のリソースおよびその他を参照してください。

- *OpenStack トレーニングガイド*
- *エンドユーザーガイド* (OpenStack コマンド行インタフェースチートシートを含む)
- *管理ユーザーガイド*
- *コマンド行インタフェースリファレンス*
- *構成リファレンス*
- *クラウド管理者ガイド*

Solaris の詳細については、Oracle Solaris 11.2 情報ライブラリを参照してください。Solaris の OpenStack の詳細については、OpenStack for Oracle Solaris 11 を参照してください。

OpenStack コミュニティーでは、異なる用語が同じ意味を持っていることがあります。たとえば、クラウドの仮想マシンは、サーバー、インスタンス、またはコンピュート VM と呼ばれることがあります。コンピュートやネットワークといった OpenStack の機能部分は、モジュール、コンポーネント、またはサービスと呼ばれることがあります。OpenStack では、*プロジェクト*と*テナント*は同じ意味で使われます。このドキュメントでは次の用語を使用します。

サービス	OpenStack サービス (Nova や Compute サービスなど)。
SMF サービス	Solaris サービス `svc:/application/openstack/nova/nova-compute:default` など)。「サービスを有効にします」などの句は、SMF サービスを指しています。

ノード	OpenStack サービスをホストするシステム。たとえば、コントローラノードは Keystone、Glance、および Horizon サービスをホストします。
プロジェクト	Oracle Solaris ゾーンでは、プロジェクトは関連する作業に対するネットワーク規模の管理識別子です。ただし、このドキュメントでは、この用語を OpenStack の定義に従ってコンピュートモジュール内の論理的なユーザーグループという意味で使用します。プロジェクトは、割り当て制限と VM イメージへのアクセスを定義します。
VM インスタンス	クラウド内の仮想マシン。VM インスタンスは、ハードウェアサーバーのように使用できる、実行中の VM または一時停止中など既知の状態の VM です。
ゾーン	オペレーティングシステムを仮想化し、隔離されたセキュアなアプリケーション実行環境を提供する Oracle Solaris のテクノロジ。この用語は、仮想化された環境自体を指すこともあります。Oracle Solaris では、OpenStack のコンピュート仮想化はゾーンのテクノロジに基づいて構築されています。

OpenStack の用語の詳細は、`http://docs.openstack.org/glossary/content/glossary.html` を参照してください。

OpenStack のインストール要件

OpenStack をインストールする Solaris システムは、次の要件を満たす必要があります。

- **オペレーティングシステム。**OpenStack システムは Oracle Solaris 11.2 を実行している必要があります。Oracle Solaris 11.2 をインストールするには、『Oracle Solaris 11.2 システムのインストール 』を参照してください。インストールイメージをダウンロードするには、Oracle Solaris 11.2 のダウンロードページを参照してください。Oracle Solaris 11 の以前のバージョンから Oracle Solaris 11.2 にアップグレードするには、『Oracle Solaris 11.2 への更新 』および『Oracle Solaris 11.2 ソフトウェアの追加と更新 』の第 4 「Oracle Solaris イメージの更新またはアップグレード」を参照してください。
- **ハードウェア。**使用しているシステムが Oracle Solaris 11.2 をサポートしていることを確認するには、Oracle Solaris 11.2 のシステム要件または『Oracle Solaris 11.2 ご使用にあたって 』の「Oracle Solaris 11.2 をインストールするためのシステム要件」を参照してください。OpenStack をインストールするには、各ノードにどの OpenStack サービスをインストールするかに応じて、最大 5G バイトの追加容量が必要です。必要な数の VM インスタンスをサポートするために十分な CPU、メモリー、およびディスク容量がコンピュートノードに

あることを確認してください。システムには、VM インスタンスのイメージのため、および VM インスタンスの作成のために、100 から 200G バイトの ZFS ストレージが必要です。

- **仮想化のサポート。**使用している OpenStack システムがカーネルゾーンをサポートしている必要があります。VM インスタンスは非大域ゾーンかカーネルゾーンのどちらかです。

  使用しているシステムが仮想化をサポートするかどうかを確認するには、端末ウィンドウで `virtinfo` コマンドを入力します。コマンドの出力に、次の例に示すような情報が表示されます。

```
# virtinfo
NAME              CLASS
non-global-zone   supported
kernel-zone       supported
```

  カーネルゾーンをサポートするには、使用しているシステムが次の追加要件を満たす必要があります。

  - 8G バイト以上の物理 RAM。
  - メモリーエラーを回避するための、ホストでの ZFS Adaptive Replacement Cache (ARC) の十分なチューニング。詳細は、『Oracle Solaris カーネルゾーンの作成と使用 』の「カーネルゾーンホストでのホスト ZFS ARC キャッシュのチューニング」を参照してください。

注記 - カーネルゾーンは、Oracle VM Server for x86 ゲストまたは Oracle VM VirtualBox では実行できません。

カーネルゾーンのインストール要件の詳細については、次のリソースを参照してください。

- 『Oracle Solaris カーネルゾーンの作成と使用 』の「Oracle Solaris カーネルゾーンのハードウェアおよびソフトウェア要件」
- Oracle Solaris 11.2 OpenStack 統合アーカイブのダウンロードページの Oracle Solaris 11.2 OpenStack 統合アーカイブに関するセクションにある `README` ファイル

OpenStack のインストールに影響する可能性がある追加情報については、125 ページの「既知の制限事項」を参照してください。

♦ ♦ ♦ 第 2 章

2 評価構成のインストール

評価のために、OpenStack を単一の Oracle Solaris システムにインストールできます。このタイプの OpenStack インストールは、シングルノードインストールとも呼ばれます。OpenStack のバージョンは、システム上の Oracle Solaris に応じて、Havana か Juno のどちらかになります。

マルチノード OpenStack のインストールの説明については、第3章「マルチノード Havana OpenStack 構成での複数システムにまたがるインストール」または第4章「マルチノード Juno OpenStack 構成での複数システムにまたがるインストール」を参照してください。

この章では、統合アーカイブを使用してシングルノードをインストールする方法、およびブラウザインタフェースを使用して VM インスタンスを簡単に作成する方法について説明します。この章で扱う内容は、次のとおりです。

注記 - 特に記載がないかぎり、この章の内容は Havana と Juno の両方のバージョンに適用されます。

OpenStack 統合アーカイブの配備

Unified Archives のダウンロードページには、Oracle Solaris アーカイブのみ、または OpenStack アーカイブ付き Oracle Solaris をダウンロードする場合のオプションが用意されています。

OpenStack 付き Oracle Solaris アーカイブでは、オペレーティングシステム、および 11 ページの「Oracle Solaris への OpenStack 統合のしくみ」に一覧表示されているすべての OpenStack サービスの両方がインストールされます。

Oracle Solaris OpenStack 統合アーカイブは次の機能を提供します。

- 実行する必要がある構成作業の量を減らすため、大部分が事前に構成されている OpenStack サービス
- 実行する必要がある追加構成を自動化するスクリプト
- Solaris システムテンプレート
- 非大域ゾーンとカーネルゾーンの 2 つのイメージが事前にロードされた Glance イメージストア

OpenStack 統合アーカイブは、ベアメタルシステムまたはカーネルゾーンのどちらかに配備できます。オプションについては、21 ページの「単一システムのインストール」を参照してください。

イメージファイルのダウンロード

Oracle Solaris OS と OpenStack の両方を提供する単一のイメージファイルをダウンロードするには、次の手順を使用します。

▼ イメージファイルをダウンロードする方法

1. **インターネットブラウザで、統合アーカイブのダウンロードページに移動します。**
2. **ライセンス契約書を読み、「同意する」ボタンをクリックします。**
3. **実行しようとしているインストールのタイプに適したファイルをダウンロードします。**

   - **使用しているシステムアーキテクチャーに対応する統合アーカイブファイルをダウンロードします。**

     このファイルを使用して、次のいずれかのタイプのインストールを実行します。

     - この `.uar` ファイルを `zoneadm install` コマンドの引数として使用して、カーネルゾーンを直接インストールします。
     - この `.uar` ファイルを AI インストールマニフェストで参照して、AI インストールサービスを使った AI インストールを実行します。
     - この `.uar` ファイルを使用して、AI ブート可能メディアを作成します。

- **使用しているシステムアーキテクチャーに対応する USB ブートイメージをダウンロードします。**
  この `usb` ファイルを使用して、ブート可能メディアから統合アーカイブをインストールします。

4. **ダウンロードしたファイルの整合性を確認します。**
   MD5 チェックサムリンクをクリックします。次の `digest` コマンドを実行して、その出力をチェックサムファイルの対応するチェックサムと比較します。

```
$ digest -a md5 file
```

単一システムのインストール

このセクションでは、単一システムの OpenStack 評価構成をインストールする方法について説明します。最初に説明する 3 つの方法では、ベアメタルにインストールします。4 つ目の方法では、カーネルゾーンにインストールします。

- 21 ページの「ダウンロードした USB ファイルを使用してインストールする方法」
- 23 ページの「統合アーカイブファイルと AI インストールサービスを使用してインストールする方法」
- 24 ページの「統合アーカイブファイルから作成した AI ブート可能メディアを使用してインストールする方法」
- 25 ページの「Havana 統合アーカイブファイルを使用してカーネルゾーンにインストールする方法」

▼ ダウンロードした USB ファイルを使用してインストールする方法

この手順では、ダウンロードした USB ファイルを使用して Oracle Solaris と OpenStack をベアメタルにインストールする方法について説明します。この方法は、AI サーバーを必要とせず、もっとも直接的なベアメタルインストールの方法です。

1. **ターゲットのシステムアーキテクチャーに対応する USB ファイルをダウンロードします。**
   20 ページの「イメージファイルをダウンロードする方法」を参照してください。

2. **USB ファイルを USB フラッシュドライブに転送します。**

   - **`usbcopy` コマンドを使用する。**

Oracle Solaris 11.2 システムにアクセスできる場合は、usbcopy ユーティリティーを使用します。usbcopy(1M) のマニュアルページを参照してください。

Oracle Solaris 11.2 の usbcopy コマンドを使用する必要があります。以前のバージョンの Solaris の usbcopy は使用できません。

- **dd コマンドを使用する。**

  Oracle Solaris 11.2 システムにアクセスできない場合は、dd コマンドを使用できます。

  dd を使用するときは、適切なディスク (フラッシュドライブ) を正しく指定するように特に注意してください。

  - **Oracle Solaris 11 の場合:**

    a. **HAL サービスを無効にします。**

    ```
    # svcadm disable -t hal
    ```

    b. **フラッシュドライブを挿入し、該当するデバイスを特定します。**

    ```
    # rmformat
    ```

    c. **イメージをコピーします。**

    ```
    # dd if=/path/image.usb of=/dev/rdsk/device bs=16k
    ```

    d. **HAL サービスを有効にします。**

    ```
    # svcadm enable hal
    ```

  - **Linux の場合:**

    a. **フラッシュドライブを挿入し、該当するデバイスを特定します。**

    ```
    # dmesg | tail
    ```

    b. **イメージをコピーします。**

    ```
    # dd if=/path/image.usb of=/dev/diskN bs=16k
    ```

  - **MacOSX の場合:**

a. **ドライブ `/dev/disk`*N*、*N* はディスク番号) を識別します。**

```
# diskutil list
# diskutil unmountDisk /dev/diskN
```

b. **イメージをコピーします。**

```
# dd if=/path/image.usb of=/dev/diskN bs=16k
```

3. **フラッシュドライブをシステムに挿入し、USB からブートします。**

   対話式システム構成 (SCI) ツールが表示されます。SCI ツールが表示されない場合は、Enter キーまたは Ctrl+L を押して画面表示を更新します。

▼ 統合アーカイブファイルと AI インストールサービスを使用してインストールする方法

この手順では、ダウンロードした統合アーカイブファイルと AI を使用して Oracle Solaris と OpenStack をベアメタルにインストールする方法について説明します。

1. **ターゲットのシステムアーキテクチャーに対応する統合アーカイブファイルをダウンロードします。**

   20 ページの「イメージファイルをダウンロードする方法」を参照してください。

2. **AI マニフェストを作成します。**

   Oracle Solaris AI インストールサーバーで、AI の設定に従って `/usr/share/auto_install/manifest/default_archive.xml` をコピーし、変更します。`ARCHIVE software` セクションで、ダウンロードした `.uar` ファイルの場所を指定します。

3. **AI インストールサービスを設定します。**

   前の手順で作成した AI マニフェストを使用して AI インストールサービスを設定します。『Oracle Solaris 11.2 システムのインストール 』のパート III「インストールサーバーを使用したインストール」を参照してください。

4. **ネットワーク経由でシステムをブートします。**

```
ok boot net -install
```

5. **インストールが完了したら、システムをリブートします。**

システムを構成するための SCI ツールが表示されます。SCI ツールが表示されない場合は、Enter キーまたは Ctrl+L を押して画面表示を更新します。

▼ 統合アーカイブファイルから作成した AI ブート可能メディアを使用してインストールする方法

この手順では、ブート可能 AI イメージを作成して Oracle Solaris 11.2 と OpenStack をベアメタルにインストールする方法について説明します。ブート可能 USB イメージは、ダウンロードした統合アーカイブファイルから作成します。この方法の詳細は、『Oracle Solaris 11.2 システムのインストール 』の第 5 「メディアからブートする自動インストール」を参照してください。

1. **ターゲットのシステムアーキテクチャーに対応する統合アーカイブファイルをダウンロードします。**
20 ページの「イメージファイルをダウンロードする方法」を参照してください。

2. **統合アーカイブファイルから AI USB を作成します。**

```
# archiveadm create-media -s http://pkg.oracle.com/solaris/release \
-f usb -o workdir/usb-filename
 \
workdir/uar-file
```

workdir は、統合アーカイブファイルをダウンロードした場所です。同じディレクトリに AI USB ファイルが作成されます。

3. **USB ファイルを USB フラッシュドライブに転送します。**

- Oracle Solaris 11.2 システムに USB ファイルをダウンロードした場合は、`usbcopy` コマンドを使用します。
- Oracle Solaris OS が Oracle Solaris 11.2 より前のバージョンであるシステムに USB ファイルをダウンロードした場合は、次のように `dd` コマンドを使用します。

    1. HAL サービスを無効にします。

    ```
    # svcadm disable -t hal
    ```

    2. フラッシュドライブを挿入し、該当するデバイスを特定します。

    ```
    # rmformat
    ```

    3. イメージをフラッシュドライブにコピーします。

    ```
    # dd if=/path/image.usb of=/dev/rdsk/device bs=16k
    ```

4. HAL サービスを有効にします。

```
# svcadm enable hal
```

4. **デフォルトの AI マニフェストを確認します。**

   デフォルトのマニフェストを使用することも、カスタムマニフェストを作成することもできます。カスタムマニフェストを作成する場合は、インストールするシステムから到達可能な場所にカスタムマニフェストを格納します。

5. **USB フラッシュドライブをシステムに挿入し、USB からブートします。**

   デフォルトの AI マニフェストを使用するか、カスタムマニフェストの場所を指定するように求められます。

   システムを構成するための SCI ツールが表示されます。SCI ツールが表示されない場合は、Enter キーまたは Ctrl+L を押して画面表示を更新します。

▼ Havana 統合アーカイブファイルを使用してカーネルゾーンにインストールする方法

この手順では、ダウンロードした統合アーカイブファイルを使用して Oracle Solaris 11.2 と Havana OpenStack をカーネルゾーンに直接インストールする方法について説明します。

始める前に カーネルゾーンをホストするシステムが 16 ページの「OpenStack のインストール要件」で指定されている仮想化の要件を満たしていることを確認してください。

1. **ターゲットのシステムアーキテクチャーに対応する統合アーカイブファイルをダウンロードします。**

   20 ページの「イメージファイルをダウンロードする方法」を参照してください。

2. **カーネルゾーンを作成します。**

```
# zonecfg -z OpenStackKZ create -t SYSsolaris-kz
```

3. **カーネルゾーンを構成します。**

   カーネルゾーンに十分な仮想 CPU、RAM、ストレージ、および MAC アドレスがあることを確認します。カーネルゾーンの内部に作成された非大域ゾーンは、これらの余分な MAC アドレスを自動的に消費できます。

次の例では、8 個の仮想 CPU、最大 8G バイトの物理メモリー、および MAC アドレスの自動割り当てを使用してゾーンを構成しています。構成可能なリソースの詳細は、zonecfg(1M) のマニュアルページを参照してください。

```
# zonecfg -z OpenStackKZ
zonecfg:OpenStackKZ> add virtual-cpu
zonecfg:OpenStackKZ:virtual-cpu> set ncpus=8
zonecfg:OpenStackKZ:virtual-cpu> end
zonecfg:OpenStackKZ> select capped-memory
zonecfg:OpenStackKZ:capped-memory> set physical=8g
zonecfg:OpenStackKZ:capped-memory> end
zonecfg:OpenStackKZ> select anet id=0
zonecfg:OpenStackKZ:anet> add mac
zonecfg:OpenStackKZ:anet:mac> set mac-address=auto
zonecfg:OpenStackKZ:anet:mac> end
zonecfg:OpenStackKZ:anet> end
zonecfg:OpenStackKZ> exit
```

4. **構成を確認します。**

```
# zonecfg -z OpenStackKZ info
```

5. **カーネルゾーンをインストールします。**

次の例は、.uar ファイルの x86 バージョンのインストールを示しています。

```
# zoneadm -z OpenStackKZ install -a path/uar-file
```

ここで、*uar-file* は Havana OpenStack アーカイブ付き Oracle Solaris 11.2 です。

6. **ゾーンをブートします。**

```
# zoneadm -z OpenStackKZ boot
```

7. **ゾーンコンソールにログインして構成を完了します。**

```
# zlogin -C OpenStackKZ
```

システムを構成するための SCI ツールが表示されます。SCI ツールが表示されない場合は、Enter キーまたは Ctrl+L を押して画面表示を更新します。

8. **カーネルゾーンに IP アドレスがあることを確認します。**

統合アーカイブでは、DHCP サーバーがカーネルゾーンに IP アドレスを割り当てることが期待されます。DHCP を使用している場合は、カーネルゾーンに MAC アドレスが割り当てられていることを確認してください。前の手順で SCI ツールのネットワークページの「自動」を選択すると、MAC アドレスが割り当てられます。DHCP を使用していない場合は、必ずカーネルゾーンに IPv4 アドレスを割り当ててください。

現在のところカーネルゾーンに対するシステムリポジトリは存在しないため、この IP アドレスは重要です。IP アドレスがあれば、必要に応じてカーネルゾーンから IPS パッケージリポジトリに接続できます。

Elastic Virtual Switch の構成

インストールが完了し、システムがリブートされたあとは、Elastic Virtual Switch (EVS) を構成する必要があります。EVS を使って、複数の物理サーバーにまたがる仮想スイッチを作成、構成、およびモニターできます。EVS は、同じ EVS の一部である VM どうしを接続します。EVS の詳細については、『Oracle Solaris 11.2 での仮想ネットワークとネットワークリソースの管理 』の第 5 「エラスティック仮想スイッチについて」を参照してください。

EVS の構成を自動化するためのスクリプトが用意されています。次のコマンドを使用してスクリプトを実行します。

```
# /usr/demo/openstack/configure_evs.py
```

このスクリプトによって次の構成が実行されます。

- 必要なすべてのユーザー `root`、`evsuser`、`neutron` UNIX ユーザーなど) の Secure Shell (SSH) 鍵を作成し、それらの公開鍵を `evsuser` ユーザーの `/var/user/evsuser/.ssh/authorized_keys` ファイルに追加します。
- EVS を構成します。
- `neutron-server:default` および `neutron-dhcp-agent:default` SMF サービスを有効にします。
- 使用する仮想 LAN テクノロジ (VLAN または VXLAN) とそれに対応する ID またはセグメントを指定します。

Juno OpenStack 統合アーカイブの配備

Juno OpenStack アーカイブを配備するには、まず使用中のシステムが Oracle Solaris 11.2 SRU 10 リリースを実行していることを確認する必要があります。以前の Oracle Solaris 11.2 リリースでは Juno をサポートしていません。

▼ 統合アーカイブファイルを使用して Juno OpenStack をカーネルゾーンにインストールする方法

始める前に カーネルゾーンをホストするシステムが、16 ページの「OpenStack のインストール要件」で指定されている仮想化の要件を満たしている必要があります。OpenStack 統合アーカイブファイルがシステムにダウンロードされていることも確認してください。20 ページの「イメージファイルをダウンロードする方法」を参照してください。

1. **カーネルゾーンを作成します。**

   ```
   # zonecfg -z kzone-name create -t SYSsolaris-kz
   ```

   この手順では、`SYSsolaris-kz` と呼ばれる Oracle Solaris テンプレートに基づいてカーネルゾーンを作成します。

2. **カーネルゾーンを構成します。**
   次の例では、8 個の仮想 CPU と最大 12G バイトの物理メモリーを使用してゾーンを構成しています。構成可能なリソースの詳細は、`zonecfg`(1M) のマニュアルページを参照してください。

   ```
   # zonecfg -z kzone-name
   zonecfg:OpenStackKZ> add virtual-cpu
   zonecfg:OpenStackKZ:virtual-cpu> set ncpus=8
   zonecfg:OpenStackKZ:virtual-cpu> end
   zonecfg:OpenStackKZ> select capped-memory
   zonecfg:OpenStackKZ:capped-memory> set physical=12g
   zonecfg:OpenStackKZ:capped-memory> end
   zonecfg:OpenStackKZ> verify
   zonecfg:OpenStackKZ> exit
   ```

3. **(オプション) 構成を確認します。**

   ```
   # zonecfg -z kzone-name info
   ```

4. **カーネルゾーンをインストールします。**
   次の例では、カーネルゾーンに 50G ビットのディスク容量を使用して、VM インスタンス用のボリュームを作成するための十分な容量を確保します。

   ```
   # zoneadm -z kzone-name install -a archive-path -x install-size=50g
   ```

   ここで、*archive-path* はダウンロードされる Juno OpenStack 統合アーカイブの場所のフルパス名を指します。

5. **ゾーンをブートします。**

   ```
   # zoneadm -z kzone-name boot
   ```

6. **ゾーンコンソールにログインして構成を完了します。**

   ```
   # zlogin -C kzone-name
   ```

   システムを構成するための SCI ツールが表示されます。SCI ツールが表示されない場合は、Enter キーまたは Ctrl+L を押して画面表示を更新します。

OpenStack ダッシュボードの使用

OpenStack のインストールタスクとインストール後の構成タスクが完了したあとは、OpenStack ダッシュボードにログインして、利用可能なリソースを表示したり、VM インスタンスを作成してブートしたりします。

▼ OpenStack ダッシュボードにアクセスする方法

1. **OpenStack システムに接続できる任意のシステムにログインします。**
2. **ブラウザを構成します。**
   a. **JavaScript を有効にします。**
   b. **Cookie を保持します。**
3. **ブラウザの場所フィールドまたはアドレスフィールドに、次の場所を入力します。**

   ```
   http://system/horizon/
   ```

   system は、OpenStack 統合アーカイブがインストールされ、Apache Web サーバーの下で Horizon OpenStack サービスを実行している OpenStack システムの名前または IP アドレスです。

   カーネルゾーンに統合アーカイブをインストールした場合、OpenStack システムはカーネルゾーンであり、*system* はカーネルゾーンの名前または IP アドレスです。

4. **ログイン画面で次の情報を入力します。**
   - ユーザー名: `admin`
   - パスワード: `secrete`

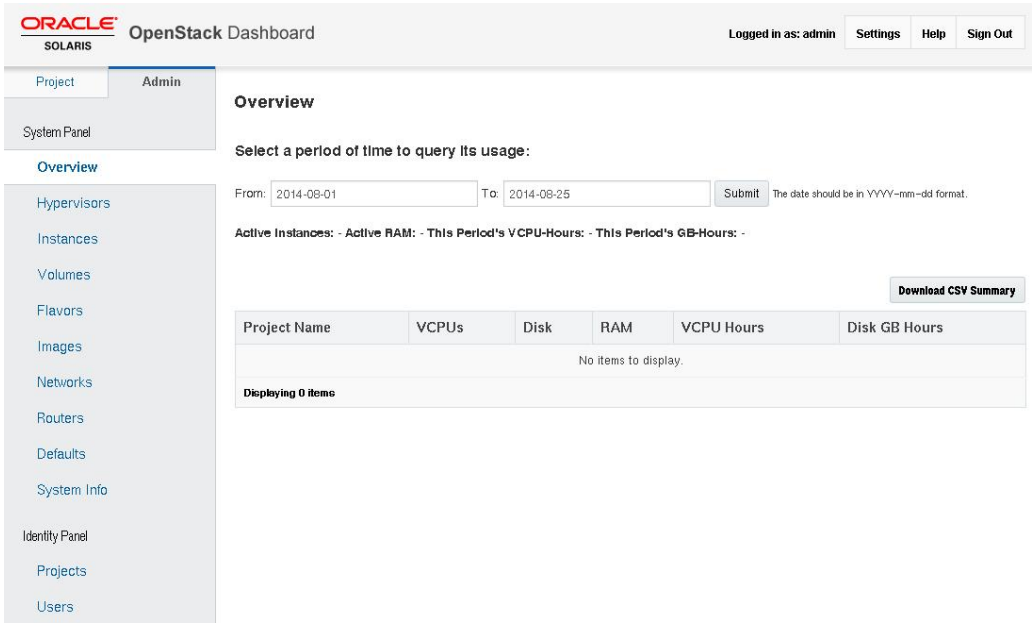
ダッシュボードで使用できる機能は、ログイン時に使用したユーザーのアクセス権 (または役割) によって異なります。

ダッシュボードの詳細

クラウド管理者としてログインすると、画面の左側に「プロジェクト」と「管理」の 2 つのタブを含むパネルが表示されます。「管理」パネルはデフォルトのクラウド管理者ビューです。「管理」パネルでの選択に応じて、次の機能が実行されます。

- クラウド内で使用されている Nova インスタンスと Cinder ボリュームの全体的な表示
- 次のような VM インスタンスの特性を定義するフレーバ定義を表示および編集する機能。
  - 仮想 CPU の数
  - メモリーの量
  - 割り当てられたディスク容量
  - 基となる Solaris ゾーンのブランド: 非大域ゾーンは `solaris`、カーネルゾーンは `solaris-kz`
- クラウド管理者が使用する仮想ネットワークおよびルーターを作成する機能
- 仮想コンピューティングリソースの所有権をグループ化したり分離したりすることでプロジェクトを表示および編集する機能
- クラウドのリソースを使用する人またはサービスであるユーザーを表示および編集する機能

図 2-1 OpenStack ダッシュボードの「管理」の「概要」ウィンドウ

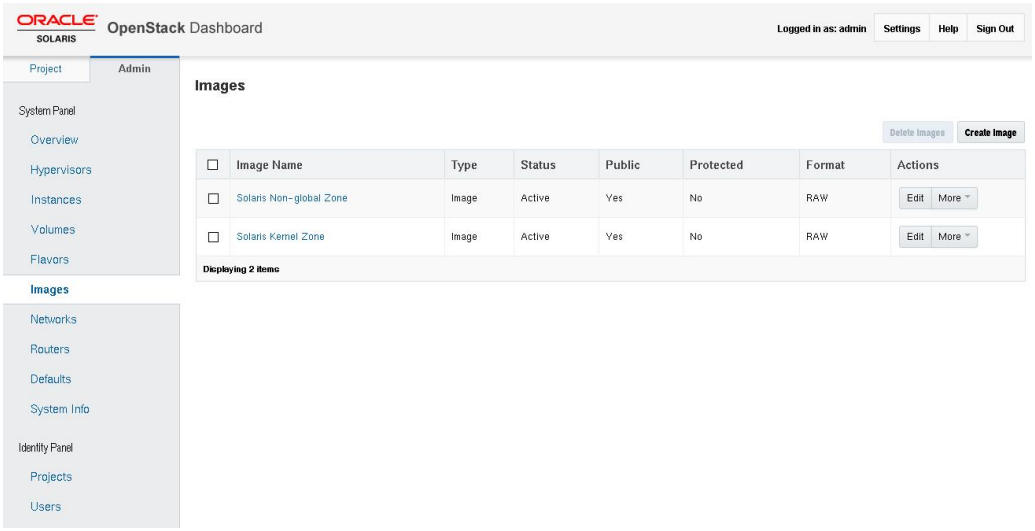


21 ページの「単一システムのインストール」の説明に従って OpenStack をインストールした場合、OpenStack システムは次のリソースを使用して事前構成されます。

- 2 つのイメージ: Solaris 非大域ゾーンと Solaris カーネルゾーン
- 2 つのプロジェクト: `demo` と `service`
- 10 個のフレーバ

次の図はイメージを示しています。

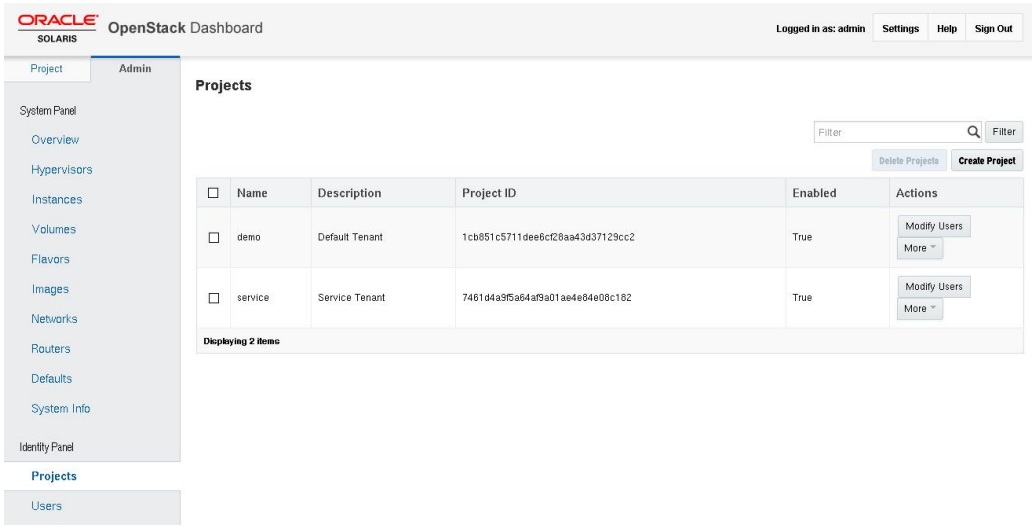
図 2-2 OpenStack ダッシュボードの「イメージ」画面



次の図はプロジェクト (テナントとも呼ばれる) を示しています。

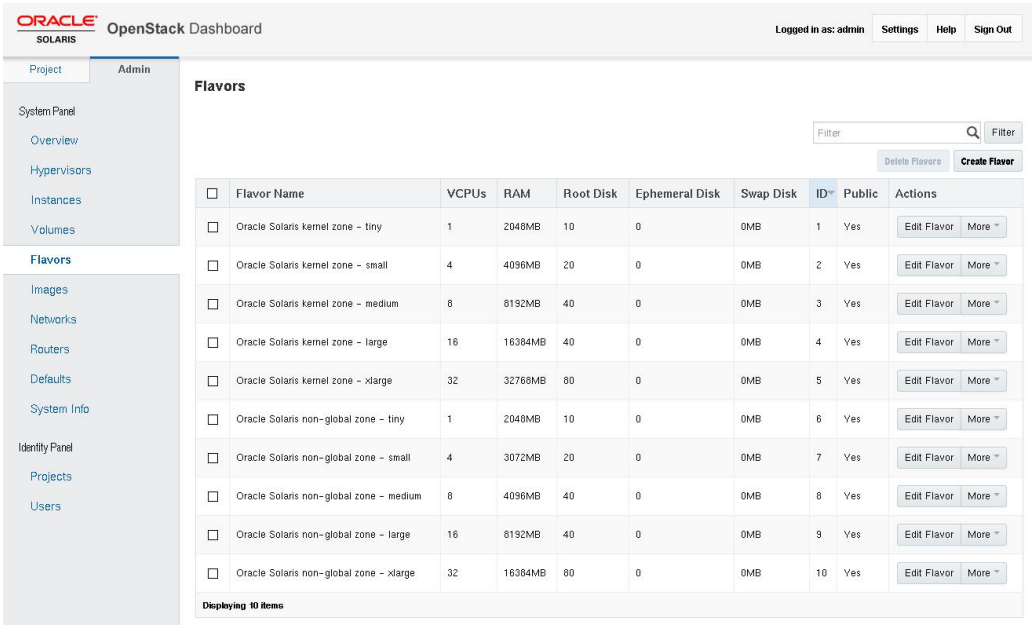
- `demo` テナントはデフォルトのテナントです。デフォルトでは、`demo` テナントは単一のユーザー (`admin`) をメンバーとして作成されます。
- `service` テナントは、クラウド管理者が複数のテナントで共有されるリソースを作成するために使用するテナントです。たとえば、このドキュメントの例とシナリオでは、`service` テナント内に Neutron ルーターを作成して、すべてのテナントでルーターを共有します。OpenStack 設定では、`service` テナントをほかの目的に使用しないでください。OpenStack サービスはサービス固有のユーザーを使用して相互に通信しますが、これらのユーザーはすべて `admin` 役割を持っており、service テナントの一部です。

図 2-3 OpenStack ダッシュボードの「プロジェクト」画面



次の図はフレーバを示しています。フレーバのテキストリストについては、116 ページの「フレーバに関する情報の表示」を参照してください。

図 2-4 OpenStack ダッシュボードの「フレーバー」画面



VM インスタンスの作成とブート

VM インスタンスを作成するには、ダッシュボードの「プロジェクト」パネルを使用します。

▼ ダッシュボードを使用して VM インスタンスを作成する方法

始める前に SSH 鍵ペアがあることを確認します。39 ページの「SSH 鍵ペアを作成する方法」を参照してください。

外部ネットワークが定義されていることを確認します。67 ページの「Neutron L3 エージェントの構成」を参照してください。

1. **ダッシュボードの左側のパネルで、「プロジェクト」タブをクリックします。**

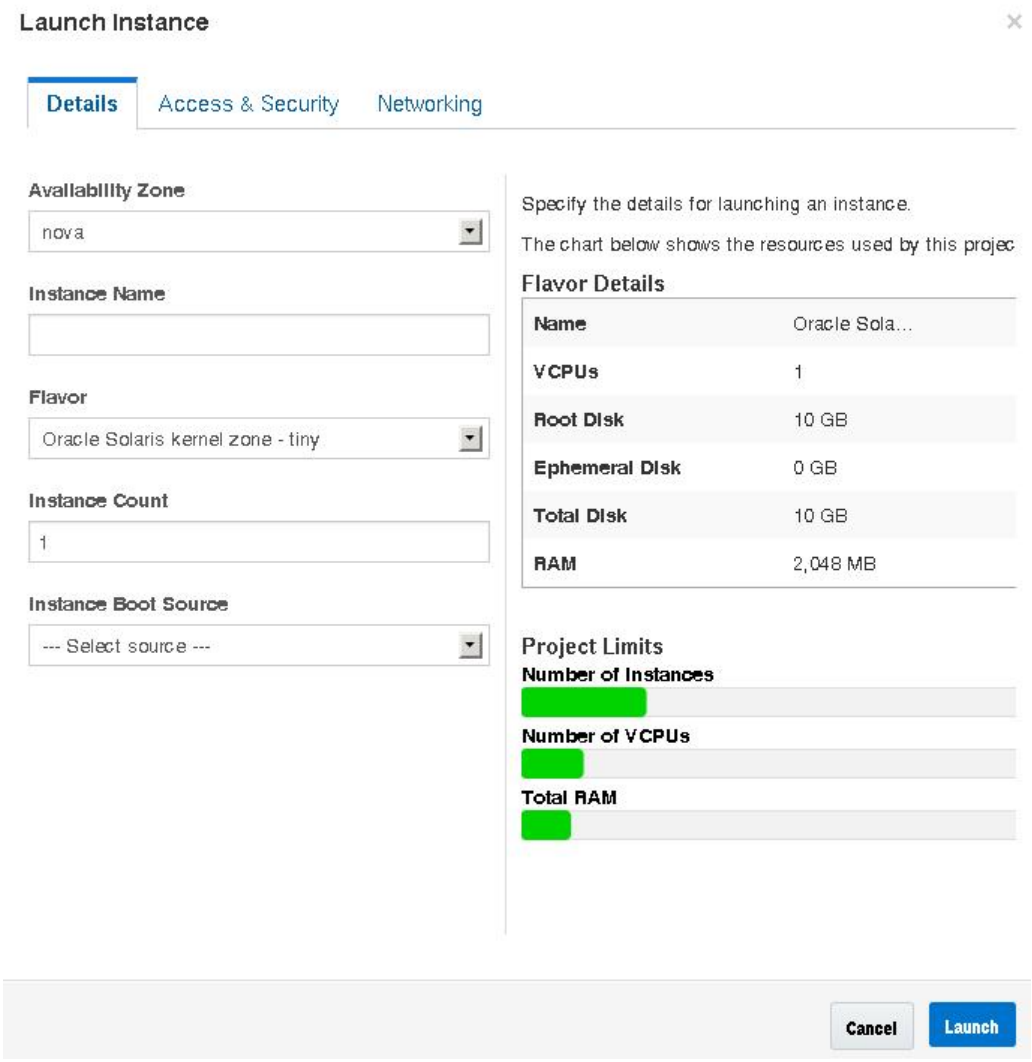
   ユーザーが現在使用しているプロジェクトがパネルの最上部に表示されます。デフォルトでは、この OpenStack 構成の `admin` ユーザーが `demo` プロジェクトを使用します。

2. **パネルの「コンピュートの管理」セクションで、「インスタンス」をクリックします。**

3. **「インスタンス」パネルの右側にある「インスタンスの起動」ボタンをクリックします。**

   次の「インスタンスの起動」ダイアログボックスが表示されます。ここで、新しい VM インスタンスの名前を指定したり、インスタンスのフレーバとイメージのタイプを選択したりできます。

図 2-5 「インスタンスの起動」ダイアログボックス



4. **「インスタンス名」フィールドに、新しい VM インスタンスの名前を入力します。**

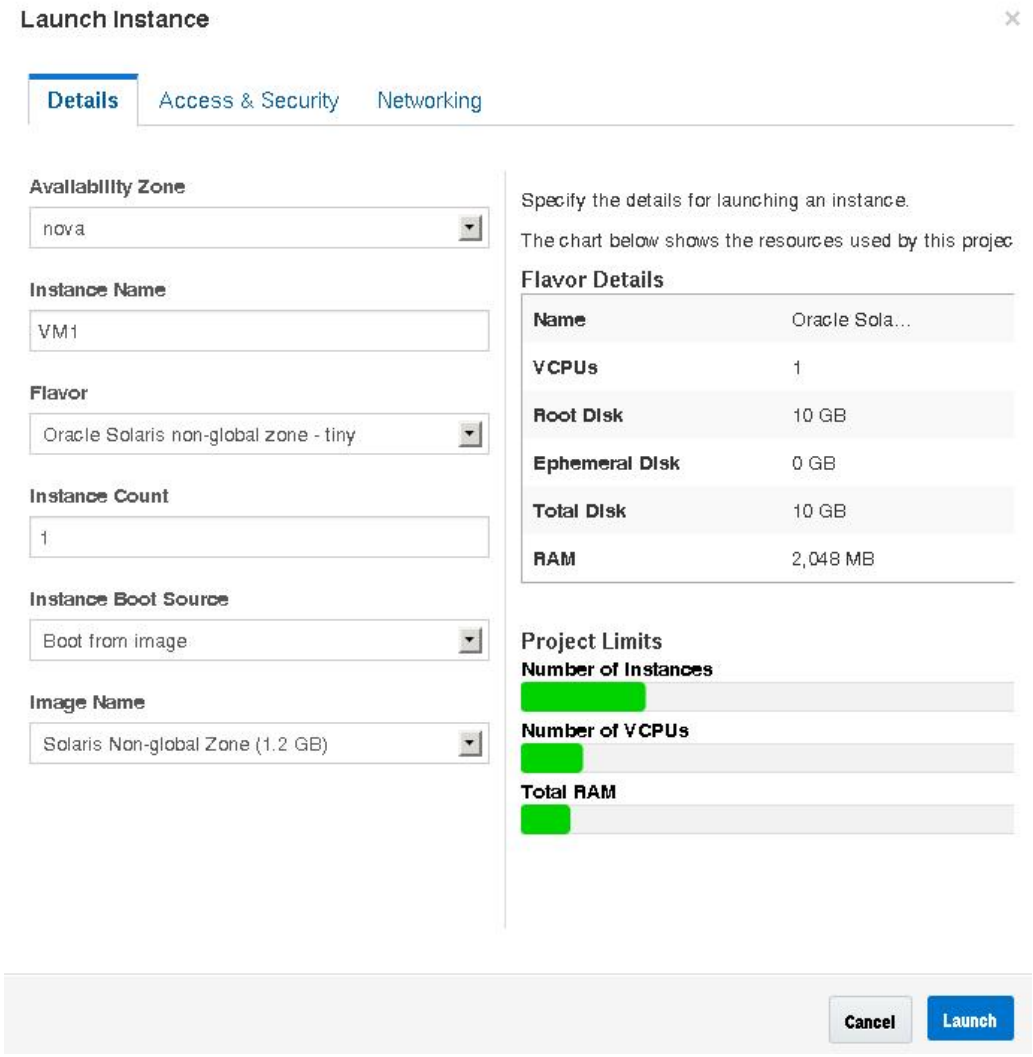
5. **「フレーバー」ドロップダウンリストからフレーバを選択します。**

   この OpenStack システムがベアメタルシステムではなくカーネルゾーンである場合は、非大域ゾーンのフレーバを選択する必要があります。

6. **「インスタンスのブートソース」で、「イメージから起動」を選択します。**

   「イメージ名」の選択内容が表示されます。選択したフレーバと同じゾーンタイプのイメージを選択します。フレーバとイメージの両方が、`solaris` 非大域ゾーンまたは `solaris-kz` カーネルゾーンのどちらかに対応している必要があります。この OpenStack システムがベアメタルシステムではなくカーネルゾーンである場合は、非大域ゾーンのイメージを選択する必要があります。

図 2-6　選択内容が表示された「インスタンスの起動」ダイアログ



7. **ダイアログボックスの「アクセスとセキュリティ」タブをクリックします。**

   新しい VM インスタンスにインストールする SSH 鍵ペアを選択します。

8. **ダイアログボックスの「ネットワーク処理」タブをクリックします。**

新しい VM インスタンスを接続するネットワークを選択します。

9. **ダイアログボックスの下部にある「起動」ボタンをクリックします。**

   新しい VM インスタンスが作成、インストール、およびブートされます。

   新しいインスタンスが利用可能になるまでに必要な時間は、イメージのサイズ、フレーバで提供されるリソース、OpenStack が新しい VM インスタンスのルートファイルシステムをどこに配置するかなど、いくつかの要因によって異なります。

10. **フローティング IP アドレスを新しい VM インスタンスに関連付けます。**

    これらの手順は、新しい VM インスタンスのインストール中に実行できます。ユーザーが VM インスタンスにログインできるようにするには、VM インスタンスに関連付けられたフローティング IP アドレスが必要です。

    a. **「アクション」列の「Floating IP の割り当て」ボタンをクリックします。**

       「Floating IP の割り当ての管理」ダイアログが開きます。

    b. **「IP アドレス」ドロップダウンメニューからアドレスを選択します。**

       「IP アドレス」フィールドに利用可能な IP アドレスがないことを示すメッセージが表示された場合は、+ ボタンをクリックします。40 ページの「フローティング IP アドレスをプロジェクトに関連付ける方法」を参照してください。

    c. **関連付けられたポートを選択します。**

       ポートのリストに、VM インスタンスの固定 IP アドレスが表示されます。

    d. **ダイアログボックスの下部にある「割り当て」ボタンをクリックします。**

次の手順

- インスタンスの詳細情報を表示したり、インスタンスのコンソールログを表示したりするには、「インスタンス」をクリックし、インスタンスの名前をクリックします。ログの更新内容を表示するには、ページをリロードします。
- 作成済みの Cinder ボリュームを表示するには、「ボリューム」をクリックします。
- クラウドネットワークの図を表示するには、「ネットワークトポロジー」をクリックします。この図には、すべてのサブネットセグメント、仮想ルーター、およびアクティブなインスタンスが含まれています。
- Glance イメージストアにアップロードされた統合アーカイブを表示するには、「イメージとスナップショット」をクリックします。

- 新しい VM インスタンスがインストールを完了し、「アクティブ」ステータスに達したら、インスタンスにログインします。次のコマンドは、手順 7 の鍵と手順 10 のフローティング IP アドレスを使用してゾーンに root としてログインします。

  ```
  # ssh root@floating-ip-address
  ```

▼ SSH 鍵ペアを作成する方法

1. **ダッシュボードの左側のパネルで、「プロジェクト」タブをクリックします。**
2. **パネルの「コンピュートの管理」セクションで、「アクセスとセキュリティー」をクリックします。**
3. **「キーペア」タブをクリックします。**
4. **「キーペアの作成」ボタンをクリックします。**

   「キーペアの作成」ダイアログが開きます。
5. **ダイアログの「キーペア名」フィールドに鍵ペアの名前を入力します。**
6. **ダイアログの「キーペアの作成」ボタンをクリックします。**

   新しい鍵ペアが自動的にダウンロードされます。

   新しい鍵ペアが自動的にダウンロードされない場合は、表示される「キーペアのダウンロード」リンクをクリックします。

   新しい鍵ペアが「アクセスとセキュリティ」パネルの「キーペア」タブに一覧表示されます。

▼ テナント用のネットワークを作成する方法

1. **ダッシュボードの左側のパネルで、「プロジェクト」タブをクリックします。**
2. **パネルの「ネットワークの管理」セクションで、「ネットワーク」をクリックします。**
3. **「ネットワークの作成」ボタンをクリックします。**

   「ネットワークの作成」ダイアログが開きます。
4. **「ネットワーク」タブの「名前」フィールドにネットワークの名前を入力します。**
5. **「サブネット」タブと「サブネットの詳細」タブで、必要な情報を指定します。**

6. **ダイアログボックスの下部にある「作成」ボタンをクリックします。**
   新しいネットワークとサブネットが「ネットワーク」パネルに一覧表示されます。

▼ フローティング IP アドレスをプロジェクトに関連付ける方法

1. **ダッシュボードの左側のパネルで、「プロジェクト」タブをクリックします。**
2. **パネルの「コンピュートの管理」セクションで、「アクセスとセキュリティー」をクリックします。**
3. **「Floating IP」タブをクリックします。**
4. **「Floating IP の確保」ボタンをクリックします。**
   「Floating IP の確保」ダイアログが開きます。
5. **ダイアログのドロップダウンメニューから、割り当てるフローティング IP が含まれるプールを選択します。**
6. **ダイアログの「IP の確保」ボタンをクリックします。**

♦ ♦ ♦ 第 3 章

3 マルチノード Havana OpenStack 構成での複数システムにまたがるインストール

この章ではマルチノード OpenStack 構成のインストール方法について説明します。各クラウドに、1 つのダッシュボードインスタンス、1 つのイメージストア、および 1 つのアイデンティティーサービスのみが必要です。各クラウドでは任意の数のストレージとコンピュートインスタンスを使用できます。本番環境では、これらのサービスは複数のノード全体に構成します。特定のクラウドデプロイメントのニーズに関連して、各コンポーネントを評価し、そのコンポーネントを個別のノードにインストールすべきかどうか、およびそのタイプのノードをどのくらい必要とするかを決定します。

注記 - この章は、Havana 固有の OpenStack 構成に該当します。Juno バージョンの OpenStack は、Oracle Solaris 11.2 SRU10 以降のリリースでのみサポートされます。

- Oracle Solaris 11.2 SRU10 リリースの入手および既存の Havana 構成の Juno へのアップグレードについては、Havana から Juno へ OpenStack をアップグレードする手順を参照してください。
- Oracle Solaris 11.2 SRU10 リリースを実行中で、最新の Juno 構成を実行する場合は、第4章「マルチノード Juno OpenStack 構成での複数システムにまたがるインストール」を参照してください。

この章では、個別の物理システムにデプロイするアーキテクチャーについて説明します。単一の Oracle SPARC サーバーをパーティション分割し、OVM Server for SPARC (LDoms) を実行するサーバーにマルチノード OpenStack を構成するには、SPARC サーバー上のマルチノード Solaris 11.2 OpenStack に関する記事を参照してください。

3 ノードアーキテクチャーの概要

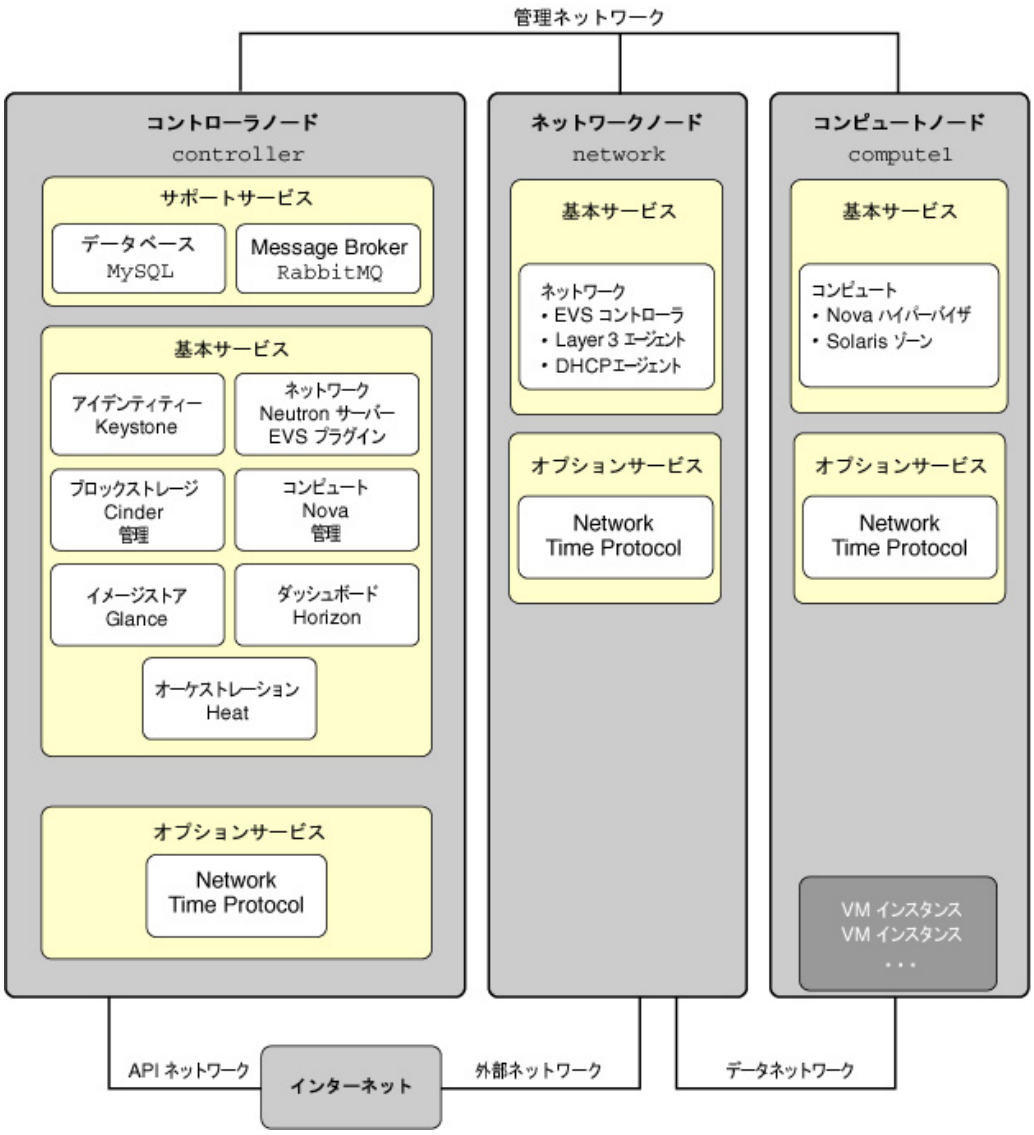
この章で説明するアーキテクチャーは次の 3 つのシステムにデプロイされています。

- コントローラノード。コントローラノードで、ほとんどの共有 OpenStack サービスおよびその他のツールが実行されます。コントローラノードはクラウドに API、スケジュール、およびその他の共有サービスを提供します。コントローラノードには、ダッシュボード、イメージストア、およびアイデンティティーサービスがあります。さらに、Nova コンピュート管理サービスと Neutron サーバーもこのノードに構成されます。
- ネットワークノード。ネットワークノードは、Neutron Layer 3 および DHCP ネットワークサービスを使用して、Nova インスタンスに仮想ネットワークおよびネットワークサービスを提供します。
- コンピュートノード。コンピュートノードには VM インスタンス (Nova コンピュートインスタンス) がインストールされています。VM インスタンスは Cinder ボリュームサービスによってプロビジョニングされた iSCSI ターゲットを使用します。

このアーキテクチャーで、3 つのノードは管理サブネットと呼ばれる共通のサブネットを共有しています。コントローラノードと各コンピュートノードは、データサブネットと呼ばれる別の共通のサブネットを共有しています。各システムは、その `net0` 物理インタフェースを介して管理ネットワークに接続されています。ネットワークノードとコンピュートノードは、それらの `net1` 物理インタフェースを介してデータネットワークに接続されています。

次の図に、この章で説明するアーキテクチャーの概要図を示します。

図 3-1 3 ノード構成のリファレンスアーキテクチャー



次の表に、各ノードにインストールされている OpenStack に関連する SMF サービスを示します。リストに、`svcadm` コマンドなどのコマンドで使用できる、各 SMF サービスの名前の最小部分を示しています。SMF サービスのインスタンス名は、名前にインスタンス名が含まれておらず、あいまいになりそうな場合のみ示しています。

表 3-1 コントローラ、ネットワーク、およびコンピュートノードにインストールされた SMF サービス

コントローラノード	ネットワークノード	コンピュートノード
mysql	neutron-dhcp-agent	nova-compute
rabbitmq	neutron-l3-agent	ntp
keystone	evs-controller	
cinder-api	ntp	
cinder-db		
cinder-db		
cinder-scheduler		
cinder-volume:default		
cinder-volume:setup		
glance-api		
glance-db		
glance-registry		
glance-scrubber		
neutron-server		
evs		
nova-api-ec2		
nova-api-osapi-compute		
nova-cert		
nova-conductor		
nova-objectstore		
nova-scheduler		
http		
ntp		
heat-api		
heat-db		

コントローラノード	ネットワークノード	コンピュートノード
heat-api-cfn heat-api-cloudwatch heat-engine		

この例のアーキテクチャーでは、Swift オブジェクトストレージサービスを示していません。Swift の構成に関する一般的な情報については、OpenStack 構成参照などの OpenStack コミュニティサイトの情報を参照してください。OpenStack システムで Swift サービスを構成する方法および、Solaris 上の OpenStack に関するほかの情報については、OpenStack for Oracle Solaris 11 に関するドキュメントを参照してください。

Oracle Solaris システムへの OpenStack のデプロイメントに役立つ OpenStack 構成パラメーターのリストについては、Oracle Solaris 11.2 での OpenStack のスタートガイドの OpenStack の一般的な構成パラメーターに関するセクションを参照してください。

3 ノードのサンプル OpenStack 構成の実装を準備するには、次の情報が手元にあることを確認してください。

- コントローラノードの IP アドレスとホスト名。
- ネットワークノードの IP アドレスとホスト名。
- コンピュートノードの IP アドレスとホスト名。
- 各サービスユーザーのパスワード (必要に応じて)。

サンプル構成では、3 つのノードの名前は `controller`、`network`、および `compute1` です。

コントローラノードの構成

コントローラノードには 1 つのダッシュボードサービス、1 つのイメージストア、および 1 つのアイデンティティーサービスがあります。このノードには、MySQL、RabbitMQ、およびコンピュート、ブロックストレージ、ネットワークの各サービスも含まれています。

Oracle Solaris 11 で ZFS とアプリケーションの間のメモリー使用をより効率的に管理するには、次の例に示すように、ノード上で `usr_reserve_hint_pct` パラメータを設定します。

```
# echo "set user_reserve_hint_pct=80" >>/etc/system.d/site:kernel-zones-reserve
# reboot
```

ここで、*site* にはユーザーの会社を指定することもあります。

ほかの OpenStack ノードでも同様にこのパラメータを設定するようにしてください。

このパラメータの詳細については、`https://support.oracle.com` で MOS にログインし、*Oracle Solaris 11.2 での ZFS とアプリケーションの間のメモリー管理*に関するドキュメント 1663862.1 を確認してください。

OpenStack サービス間の通信は Advanced Message Queuing Protocol (AMQP) によって実行されます。Solaris で AMQP は RabbitMQ によって実装されます。RabbitMQ は必須サービスです。通常、クラウド内の 1 つのノードで RabbitMQ を実行するように構成します。このアーキテクチャーでは、RabbitMQ がコントローラノードで実行するように構成します。

▼ コントローラノードを構成する方法

1. **(オプション) NTP をインストールし、構成します。**
   47 ページの「Network Time Protocol のインストール」を参照してください。

2. **(オプション) MySQL をインストールし、構成します。**
   48 ページの「MySQL のインストール」を参照してください。

3. **RabbitMQ をインストールします。**

   a. **RabbitMQ パッケージをインストールします。**

   ```
   controller# pkg install rabbitmq
   ```

   b. **RabbitMQ SMF サービスを有効にします。**

   ```
   controller# svcadm enable rabbitmq
   ```

4. **Keystone をインストールし、構成します。**
   49 ページの「Keystone のインストール」を参照してください。

5. **Cinder をインストールし、構成します。**
   52 ページの「Cinder のインストール」を参照してください。

6. **Glance をインストールし、構成します。**
   57 ページの「Glance のインストール」を参照してください。

7. **Neutron をインストールし、構成します。**
   58 ページの「コントローラノードでの Neutron のインストールと構成」を参照してください。

8. **Nova をインストールし、構成します。**
   59 ページの「Nova のインストール」を参照してください。

9. **Horizon を構成します。**
   60 ページの「Horizon を構成する方法」を参照してください。

Network Time Protocol のインストール

Network Time Protocol (NTP) のインストールはオプションですが強くお勧めします。NTP をインストールする場合、クラウドデプロイメント内の各サービスノードにインストールします。

NTP はクラウド内のすべてのサービスノード全体で一貫した時間を確保するのに役立ちます。ネットワークで NTP を有効にする場合、サービスノードがネットワークを介して時間を取得するように構成します。

- サービスノードが存在する IP サブネットで IP マルチキャストが有効にされている場合、IP マルチキャストを利用して、NTP を構成できます。
- サービスノードが存在する IP サブネットで IP マルチキャストが有効にされていない場合は、NTP を手動で構成します。

▼ Network Time Protocol をインストールし、構成する方法

1. **NTP パッケージをインストールします。**

```
controller# pkg install ntp
```

2. **構成ファイルをインストールします。**

```
controller# cp /etc/inet/ntp.client /etc/inet/ntp.conf
```

3. **NTP を構成します。**
   必要な構成は、ノードのサブネットで IP マルチキャストが有効にされているかどうかによって異なります。

   - **IP マルチキャストが有効にされている場合、追加の構成は必要ありません。**

- **IP マルチキャストが有効にされていない場合は、既存の NTP サーバーのホスト名または IP アドレスを構成します。**

  a. **`/etc/inet/ntp.conf` ファイル内の `multicastclient` オプションをコメントアウトします。**

  ```
  # multicastclient 224.0.1.1
  ```

  b. **`/etc/inet/ntp.conf` ファイル内の 1 つ以上のサーバーオプションをコメント解除します。**

  ```
  server ntp_server_1 iburst
  server ntp_server_2 iburst
  ```

4. **NTP サーバーの SMF サービスを有効にします。**

```
controller# svcadm enable ntp
```

MySQL のインストール

多くの OpenStack サービスは、重要なリソース、使用状況、およびその他の情報を追跡するためにデータベースを保持します。デフォルトでは個々の SQLite データベースがこの目的で指定されますが、単一ノード構成の場合に便利です。マルチノード構成の場合は、この情報を格納するために、MySQL データベースをお勧めします。

▼ MySQL データベースをインストールする方法

1. **コントローラノードに関連付けられるプライマリ名を決定します。**
   コントローラノードのプライマリ IP アドレスを使用して、そのノードに関連付けられるプライマリ名を決定します。

```
controller# getent hosts controller-IP
controller-IP  controller-name
```

2. **MySQL サーバーパッケージをインストールします。**

```
controller# pkg install mysql-55
```

3. **MySQL クライアントパッケージをインストールします。**

```
controller# pkg install mysql-55/client
```

4. **MySQL サーバー SMF サービスを有効にします。**

```
controller# svcadm enable mysql:version_55
```

5. **MySQL サーバーの root パスワードを設定します。**

```
controller# mysqladmin –u root password MySQL-root-password
```

6. **MySQL サーバーを構成します。**

OpenStack によって使用されるテーブルを作成します。これらのデータベースへの排他的アクセスを提供するために、コントローラノード上のサービスに特権を付与します。以前の getent hosts コマンドによって出力された *controller-name* を使用します。

```
controller# mysql –u root –p
Enter password: MySQL-root-password
mysql> create database cinder;
mysql> grant all privileges on cinder.*
-> to 'cinder'@'controller-name'
-> identified by 'cinder';
mysql> create database glance;
mysql> grant all privileges on glance.*
-> to 'glance'@'controller-name'
-> identified by 'glance';
mysql> create database keystone;
mysql> grant all privileges on keystone.*
-> to 'keystone'@'controller-name'
-> identified by 'keystone';
mysql> create database nova;
mysql> grant all privileges on nova.*
-> to 'nova'@'controller-name'
-> identified by 'nova';
mysql> flush privileges;
mysql> quit
```

7. **MySQL Python クライアントライブラリパッケージをインストールします。**

```
controller# pkg install python-mysql
```

Keystone のインストール

Keystone サービスはコントローラノードにインストールし、構成してください。

▼ Keystone をインストールし、構成する方法

1. **Keystone パッケージをインストールします。**

```
controller# pkg install keystone
```

2. **Keystone 構成ファイルを変更します。**

   /etc/keystone/keystone.conf ファイル内の次の 2 つのパラメータをコメント解除し、設定します。

   a. **admin_token パラメータを設定します。**

      admin_token パラメータは Keystone とその他の OpenStack サービス間の「共有シークレット」です。このパラメータの値には任意の文字列を使用できますが、この値を公開または配布しないでください。そのような文字列を作成する 1 つの方法は、次のコマンドに示すように OpenSSL を使用することです。

      ```
      controller# openssl rand -hex 10
      random_string
      ```

      この出力値を使用して、/etc/keystone/keystone.conf ファイル内に admin_token パラメータを設定します。

      ```
      admin_token = random_string
      ```

   b. **connection パラメータを設定します。**

      connection パラメータは Keystone データベースの場所および使用されるデータベースの種類を表す URI です。

      以前の getent hosts コマンドによって出力された *controller-name* を使用して、/etc/keystone/keystone.conf ファイル内に connection パラメータを設定します。

      ```
      connection = mysql://keystone:keystone@controller-name/keystone
      ```

3. **公開鍵インフラストラクチャー (PKI) トークンを生成します。**

   ```
   controller# su - keystone -c "keystone-manage pki_setup"
   ```

4. **Keystone SMF サービスを有効にします。**

   ```
   controller# svcadm enable keystone
   ```

5. **Keystone データベースを生成します。**

   このステップは手動で行うか、または次の例に示すように sample_data.sh スクリプトを使用できます。以前の getent hosts コマンドによって出力された *controller-name* を使用します。

   ```
   controller# su - keystone -c "env
   ```

```
CONTROLLER_ADMIN_ADDRESS=controller-name
CONTROLLER_INTERNAL_ADDRESS=controller-name
CONTROLLER_PUBLIC_ADDRESS=controller-name
/usr/demo/openstack/keystone/sample_data.sh"
```

sample_data.sh スクリプトは、各 API サービスが存在するノードおよび各サービスのパスワードを定義する環境変数サポートします。環境から設定できるパラメータの詳細については、スクリプトを確認してください。デフォルトで、Keystone ユーザーはユーザー名と同じパスワードで service テナントのサービスごとに作成されます。たとえば、nova ユーザーはパスワード nova で作成されます。

Heat のインストールと構成

Heat は、作成されたテンプレートに基づいてクラウドアプリケーションを配備できるようにする OpenStack のオーケストレーションエンジンです。Heat は Keystone と同じノードにインストールします。

▼ Heat を構成する方法

始める前に このタスクを実行する前に、まず「Keystone をインストールし、構成する方法」の説明に従って Keystone を構成する必要があります。

1. **Heat パッケージをインストールします。**

   ```
   controller# pkg install heat
   ```

2. **Heat 設定スクリプトを実行します。**

   ```
   # /usr/demo/openstack/keystone/heat-keystone-setup
   ```

3. **/etc/heat/api-past.ini を編集して、ファイル内の次の情報を更新します。**

   ```
   # Auth middleware that validates token against keystone
   [filter:authtoken]
   paste.filter_factory = heat.common.auth_token:filter_factory
   auth_uri = http://controller-IP:5000/v2.0
   identity_uri = http://controller-IP:35357
   admin_tenant_name = keystone
   admin_user = heat
   admin_password = heat-password
   ```

4. **Heat サービスを有効にします。**

   ```
   # svcadm enable -rs heat-api heat-db heat-engine heat-api-cfn heat-api-cloudwatch
   ```

Cinder のインストール

Cinder 構成では少なくとも次の情報を指定する必要があります。

- Keystone で認証するための認可情報。
- 作成するボリュームのクラス。

▼ Cinder をインストールし、構成する方法

1. Cinder パッケージをインストールします。

```
controller# pkg install cinder
```

2. 認証構成情報を指定します。

`/etc/cinder/api-paste.ini` ファイル内の次のパラメータをコメント解除し、設定します。これらのパラメータは、Keystone API サービスと Cinder 認証情報の場所を指定します。

```
auth_uri = http://controller-name:5000/v2.0
identity_uri = http://controller-name:35357
admin_tenant_name = service
admin_user = cinder
admin_password = cinder-password
```

3. Cinder ボリュームサービスが作成すべきボリュームのクラスを指定します。

`/etc/cinder/cinder.conf` ファイル内の該当する `volume_driver` パラメータをコメント解除します。次の 4 つのクラスのボリュームがサポートされています。

`ZFSVolumeDriver`

Cinder ボリュームサービスと同じノード上の Nova によって使用されるローカルボリュームの作成をサポートします。

`ZFSISCSIDriver`

リモート Nova コンピュートノードによって使用される iSCSI ターゲットの作成とエクスポートをサポートします。

`ZFSFCDriver`

リモート Nova コンピュートノードによって使用されるファイバチャネル LUN の作成とエクスポートをサポートします。

`ZFSSAISCSIDriver`

リモート Nova コンピュートノードによって使用されるリモート Oracle ZFS Storage Appliance からの iSCSI ターゲットの作成とエクスポートをサポートします。このドライバの追加パラメータを `/etc/cinder/cinder.conf` ファイルに設定する必要があります。

この章で説明する例では、Nova インスタンスによって使用されるボリュームを提供するために iSCSI を使用しています。`ZFSVolumeDriver` のデフォルトの選択をコメントアウトし、`ZFSISCSIDriver` の選択をコメント解除します。

```
# Driver to use for volume creation (string value)
# The local ZFS driver provides direct access to ZFS volumes that it
# creates. The other listed drivers provide access to ZFS volumes via
# iSCSI or Fibre Channel and are suitable for cases where block storage
# for Nova compute instances is shared.
#volume_driver=cinder.volume.drivers.solaris.zfs.ZFSVolumeDriver
volume_driver=cinder.volume.drivers.solaris.zfs.ZFSISCSIDriver
#volume_driver=cinder.volume.drivers.solaris.zfs.ZFSFCDriver
#volume_driver=cinder.volume.drivers.zfssa.zfssaiscsi.ZFSSAISCSIDriver
```

4. **追加の構成パラメータを設定します。**

`/etc/cinder/cinder.conf` ファイル内の次のパラメータをコメント解除し、設定します。これらのパラメータは、Glance API サービス、Cinder の対応するデータベース、および RabbitMQ サービスの場所を指定します。

```
glance_host=controller-name
sql_connection=mysql://cinder:cinder@controller-name/cinder
rabbit_host=controller-name
volume_driver=cinder.volume.drivers.solaris.zfs.ZFSISCSIDriver
```

5. **iSCSI ターゲットを構成する場合は、対応する SMF サービスを有効にします。**

```
controller# svcadm enable iscsi/target stmf
```

6. **Cinder SMF サービスを有効にします。**

```
controller# svcadm enable cinder-db
controller# svcadm enable cinder-api cinder-scheduler
controller# svcadm enable cinder-volume:default cinder-volume:setup
```

参照 ZFS に OpenStack Block Storage を構築する方法に関するドキュメントも参照してください。

▼ ZFS Storage Appliance iSCSI Cinder ドライバの構成方法

Oracle ZFS Storage Appliance iSCSI Cinder ドライバにより、Oracle ZFS Storage Appliance (ZFSSA) を Cinder のブロックストレージリソースとしてシームレスに使用できます。このドライバは、Nova サービスによってインスタンス化された任意の仮想マシンに、Cinder サーバーによって割り当て可能な iSCSI ボリュームを作成する機能を提供します。ドライバは `cloud/openstack/cinder` パッケージで配布されます。アプライアンスでは ZFSSA ソフトウェアリリース 2013.1.2.0 以上を実行している必要があります。

始める前に Oracle ZFS Storage Appliance 上にプールを構成します。既存のプールを使用するように選択できます。

1. **ワークフロー `cinder.akwf` を実行します。**

   Cinder ドライバ操作を実行するためのロール認可を持つ既存のユーザーを使用するか、新しいユーザーを作成できます。

   `cinder.akwf` ワークフローは次のタスクを実行します。

   - ユーザーが存在しない場合はユーザーを作成します。
   - Cinder ドライバの操作を実行するためのロール認可を設定します。
   - RESTful サービスが現在無効になっている場合、サービスを有効にします。

   コマンド行インタフェース (CLI) またはアプライアンスのブラウザユーザーインタフェース (BUI) からワークフローを実行できます。

   - **CLI からワークフローを実行します。**

```
zfssa:maintenance workflows> download
zfssa:maintenance workflows download (uncommitted)> show
Properties:
                          url = (unset)
                         user = (unset)
                     password = (unset)

zfssa:maintenance workflows download (uncommitted)> set url="url to the cinder.akwf file"
                          url = "url to the cinder.akwf file"
zfssa:maintenance workflows download (uncommitted)> commit
Transferred 2.64K of 2.64K (100%) ... done

zfssa:maintenance workflows> ls
Properties:
                    showhidden = false

Workflows:
```

```
WORKFLOW     NAME                                              OWNER SETID ORIGIN
 VERSION
workflow-000 Clear locks                                       root  false Oracle Corporation
 1.0.0
workflow-001 Configuration for OpenStack Cinder Driver root  false Oracle Corporation
 1.0.0

zfssa:maintenance workflows> select workflow-001

zfssa:maintenance workflow-001 execute (uncommitted)> set name=openstack
                          name = openstack
zfssa:maintenance workflow-001 execute (uncommitted)> set password=openstack-password
                       password = ********
zfssa:maintenance workflow-001 execute (uncommitted)> commit
User openstack created.
```

- **BUI からワークフローを実行します。**

  a. **「保守」 -> 「ワークフロー」を選択して、プラスアイコンを使用して、新しいワークフローをアップロードします。**

  b. **参照ボタンをクリックして、`cinder.akwf` ファイルを選択します。**

  c. **アップロードボタンをクリックして、ワークフローのアップロードを完了します。**

  d. **BUI の「ワークフロー」ページに表示される新しい行をクリックし、Cinder ドライバワークフローを実行します。**

     ワークフローによって、ユーザー名とパスワードの入力が求められます。このユーザー名とパスワードは、`zfssa_auth_user` および `zfssa_auth_password` として、`cinder.conf` ファイルでも使用されます。

2. **`cinder.conf` ファイルにパラメータを設定します。**

   `cinder.conf` ファイルに次の必須プロパティーを指定します。

   - `volume_driver` - `cinder.volume.drivers.zfssa.zfssaiscsi.ZFSSAISCSIDriver` がコメント解除されていることを確認してください。ほかの 3 つの選択がコメントアウトされていることを確認してください。
   - `zfssa_host` – ZFSSA 管理ホストの名前または IP アドレス。
   - `zfssa_auth_user` – ZFSSA 上の Cinder ユーザーのユーザー名。
   - `zfssa_auth_password` – ZFSSA 上の Cinder ユーザーのパスワード。

- `zfssa_pool` – ボリュームを割り当てるために使用するプール。
- `zfssa_target_portal` – ZFSSA iSCSI ターゲットポータル `data-ip:port`)。デフォルトのポートは 3260 です。
- `zfssa_project` – ZFSSA プロジェクトの名前。プロジェクトがアプライアンスに存在していない場合は、起動時にドライバによって、その名前でプロジェクトが作成されます。このプロジェクトにはドライバによって作成されたすべてのボリュームが含まれます。ボリュームの特性 (ブロックサイズなど) やアクセス (イニシエータ、ターゲット、セキュリティーなど) を設定するために追加の ZFSSA プロパティーが提供されています。
- `zfssa_initiator_group` – イニシエータグループの名前。イニシエータグループがアプライアンスに存在していない場合は、起動時にドライバによって、その名前でイニシエータグループが作成されます。`default` イニシエータグループを使用する場合は、このパラメータの値を `default` に設定します。`default` イニシエータグループは評価目的で役立つことがあります。`default` イニシエータグループは、不要なイニシエータまたは競合するイニシエータにボリュームが公開される可能性があるため、通常は使用しないでください。
- `zfssa_target_interfaces` – ZFSSA iSCSI ターゲットネットワークインタフェース。インタフェースを表示するには次のコマンドを使用します。

```
zfssa:configuration net interfaces> show
Interfaces:

  INTERFACE STATE CLASS LINKS    ADDRS          LABEL
  e1000g0   up    ip    e1000g0  1.10.20.30/24  Untitled Interface
```

- `connection` – `connection` を `sql_connection` に変更します。

  次の行を見つけます。

```
connection=mysql://cinder:cinder...
```

  この行を下に示すように変更します。

```
sql_connection=mysql://cinder:cinder...
```

3. **ZFSSA iSCSI サービスがオンラインであることを確認します。**

   ZFSSA iSCSI サービスがオンラインでない場合は、アプライアンスの BUI または CLI を使用して、それを有効にします。次の例では、アプライアンスの CLI の使用を示します。

```
zfssa:> configuration services iscsi
zfssa:configuration services iscsi> enable
zfssa:configuration services iscsi> show
```

```
Properties:
<status> = online
...
```

4. **Cinder ボリューム SMF サービスを有効にします。**

```
controller# svcadm enable cinder-volume:default cinder-volume:setup
```

Glance のインストール

Cinder 構成と同様に、Glance の設定では、認証情報と MySQL および RabbitMQ サービスの場所に関する情報を構成する必要があります。

▼ Glance をインストールし、構成する方法

1. **Glance パッケージをインストールします。**

```
controller# pkg install glance
```

2. **Glance 構成パラメータを設定します。**

次の各ファイル内の次のパラメータのコメントを解除して設定します。

- /etc/glance/glance-api.conf
- /etc/glance/glance-cache.conf
- /etc/glance/glance-registry.conf
- /etc/glance/glance-scrubber.conf

```
auth_uri = http://controller-name:5000/v2.0
identity_uri = http://controller-name:35357
admin_tenant_name = service
admin_user = glance-password
admin_password = glance
```

3. **MySQL データベースの URI を指定します。**

/etc/glance/glance-api.conf ファイルと /etc/glance/glance-registry.conf ファイルの両方で、MySQL データベースの URI を指定します。

```
connection=mysql://glance:glance@controller-name/glance
```

4. **RabbitMQ サービスの場所を指定します。**

/etc/glance/glance-api.conf に、RabbitMQ サービスの場所を設定します。

```
rabbit_host = controller-name
```

5. **Glance SMF サービスを有効にします。**

```
controller# svcadm enable glance-db
controller# svcadm enable glance-api glance-registry glance-scrubber
```

コントローラノードでの Neutron のインストールと構成

この章で説明するアーキテクチャーでは、Neutron API サービスはコントローラノードで実行します。このサービスがネットワークノードにインストールされている EVS コントローラと通信できるようにするには、コントローラの Neutron ユーザーの SSH 公開鍵をネットワークノードの evsuser ユーザーの authorized_keys ファイルにデポジットする必要があります。

▼ Neutron をインストールし、構成する方法

1. **Neutron パッケージをインストールします。**

```
controller# pkg install neutron
```

2. **コントローラノード上に neutron ユーザーの SSH 公開鍵を作成します。**

この鍵は Neutron API サービスが EVS コントローラにアクセスできるようにします。

neutron ユーザーとして ssh-keygen コマンドを使用して、neutron ユーザーの鍵を作成します。

```
controller# su - neutron
-c "ssh-keygen -N '' -f /var/lib/neutron/.ssh/id_rsa -t rsa"
```

3. **EVS コントローラノードに鍵をコピーします。**

前の手順で生成された SSH 公開鍵 /var/lib/neutron/.ssh/id_rsa.pub) を EVS コントローラが実行されているノードにコピーします。この鍵は、EVS コントローラを構成するときに参照されます。

4. **Neutron 構成ファイルにパラメータを設定します。**

Keystone 認証情報および RabbitMQ サービスの場所を指定します。/etc/neutron/neutron.conf ファイルで、次のパラメータのコメントを解除して設定します。

```
rabbit_host = controller-name

auth_uri = http://controller-name:5000/v2.0
```

```
identity_uri = http://controller-name:35357
admin_tenant_name = service
admin_user = neutron
admin_password = neutron-password
```

5. **EVS コントローラの場所を指定します。**

   a. **EVS コントローラの場所を取得します。**

   `getent hosts` コマンドでネットワークノードの IP アドレスを使用して、ネットワークノードの名前を取得します。

   ```
   network# getent hosts network-IP
   network-IP  network-name
   ```

   b. **EVS コントローラの場所を設定します。**

   `/etc/neutron/plugins/evs/evs_plugin.ini` ファイルで、コメントを解除するか次のパラメータを設定します。`getent hosts` コマンドからの出力を使用して、EVS コントローラの場所を設定します。

   ```
   evs_controller = ssh://evsuser@network-name
   sql_connection = path-to-database
   ```

6. **Neutron サーバーサービスを有効にします。**

   ```
   controller# svcadm enable neutron-server
   ```

Nova のインストール

コントローラノード上の Nova 構成には、通常の認証およびその他のサービス情報を構成する必要もあります。

▼ Nova をインストールし、構成する方法

1. **Nova パッケージをインストールします。**

   ```
   controller# pkg install nova
   ```

2. **認証構成情報を指定します。**

   `/etc/nova/api-paste.ini` ファイル内の次のパラメータをコメント解除し、設定します。これらのパラメータは、Keystone API サービスと Nova 認証情報の場所を指定します。

```
auth_uri = http://controller-name:5000/v2.0
identity_uri = http://controller-name:35357
admin_tenant_name = service
admin_user = nova
admin_password = nova-password
```

3. **追加の構成パラメータを設定します。**

   /etc/nova/nova.conf ファイル内の次のパラメータをコメント解除し、設定します。これらのパラメータは、追加の Keystone 認証サービスエンドポイント、Glance API サービス、Neutron API サービス、RabbitMQ サービス、および Nova 固有のデータベースのデータベース URI を指定します。

```
keystone_ec2_url=http://controller-name:5000/v2.0/ec2tokens
glance_host=controller-name
neutron_url=http://controller-name:9696
neutron_admin_username=neutron
neutron_admin_password=neutron-password
neutron_admin_tenant_name=service
neutron_admin_auth_url=http://controller-name:5000/v2.0
rabbit_host=controller-name
connection=mysql://nova:nova@controller-name/nova
```

4. **Nova SMF サービスを有効にします。**

```
controller# svcadm enable nova-conductor
controller# svcadm enable nova-api-ec2 nova-api-osapi-compute
nova-cert nova-conductor nova-objectstore nova-scheduler
```

▼ Horizon を構成する方法

1. **Horizon パッケージをインストールします。**

```
horizon# pkg install horizon
```

2. **Horizon で使用する証明書を生成します。**

   次のコマンドは Horizon によって使用される自己署名証明書を生成し、OpenStack ダッシュボード構成ファイルを Apache 構成ファイルディレクトリにコピーします。自己署名証明書の作成の詳細については、SSL/TLS 強力な暗号化に関する FAQ を参照してください。

```
controller# DASHBOARD=/etc/openstack_dashboard
controller# openssl req -new -x509 -nodes
-out horizon.crt -keyout horizon.key
controller# mv horizon.crt horizon.key ${DASHBOARD}
controller# chmod 0600 ${DASHBOARD}/horizon.*
```

```
controller# sed
-e "/SSLCertificateFile/s:/path.*:${DASHBOARD}/horizon.crt:"
-e "/SSLCACertificateFile/d"
-e "/SSLCertificateKeyFile/s:/path.*:${DASHBOARD}/horizon.key:"
< /etc/apache2/2.2/samples-conf.d/openstack-dashboard-tls.conf
> /etc/apache2/2.2/conf.d/openstack-dashboard-tls.conf
```

3. **~/conf.d/openstack-dashboard-tls.conf ファイルの次のパラメータに、Horizon パッケージのサイトアドレスとサーバー名を指定します。**

```
RedirectPermanent=site-address
ServerName=server-name
```

注記 - 現在の 3 ノードのサンプル構成では、この 2 つのパラメータによってコントローラノードのシステムが指定されます。

4. **新しい構成ファイルを読み取るには、次のいずれかの操作を実行します。**

- Apache サービスが無効になっている場合は有効にします。

```
controller# svcadm enable apache22
```

- Apache サービスがオンラインの場合は再起動します。

```
controller# svcs apache22
STATE          STIME    FMRI
online         Jul_07   svc:/network/http:apache22
controller# svcadm restart apache22
```

コンピュートノードの構成

コンピュートノードには VM インスタンスがインストールされています。クラウドには多数のコンピュートノードが必要になる可能性があります。

Oracle Solaris 11 で ZFS とアプリケーションの間のメモリー使用をより効率的に管理するには、次の例に示すように、ノード上で `usr_reserve_hint_pct` パラメータを設定します。

```
# echo "set user_reserve_hint_pct=80" >>/etc/system.d/site:kernel-zones-reserve
# reboot
```

ここで、*site* にはユーザーの会社を指定することもあります。

ほかの OpenStack ノードでも同様にこのパラメータを設定するようにしてください。

このパラメータの詳細については、https://support.oracle.com で MOS にログインし、*Oracle Solaris 11.2 での ZFS とアプリケーションの間のメモリー管理*に関するドキュメント 1663862.1 を確認してください。

▼ コンピュートノードを構成する方法

1. **(オプション) NTP をインストールし、構成します。**

   47 ページの「Network Time Protocol のインストール」を参照してください。

2. **Nova パッケージをインストールします。**

```
compute1# pkg install nova
```

3. **Remote Access Daemon (RAD) を再起動します。**

   Nova は RAD を使用して、Oracle Solaris ゾーンフレームワークと通信します。

```
compute1# svcadm restart rad:local
```

4. **認証構成情報を指定します。**

   /etc/nova/api-paste.ini ファイル内の次のパラメータをコメント解除し、設定します。これらのパラメータは、Keystone API サービスと Nova 認証情報の場所を指定します。

```
auth_uri = http://controller-name:5000/v2.0
identity_uri = http://controller-name:35357
admin_tenant_name = service
admin_user = nova
admin_password = nova-password
```

5. **認証パラメータ、データベースパラメータ、および関連サービスを構成します。**

   /etc/nova/nova.conf ファイル内の次のパラメータをコメント解除し、設定します。これらのパラメータは、追加の Keystone 認証サービスエンドポイント、Glance API サービス、Neutron API サービス、RabbitMQ サービス、および Nova 固有のデータベースのデータベース URI を指定します。

```
keystone_ec2_url=http://controller-name:5000/v2.0/ec2tokens
glance_host=controller-name
neutron_url=http://controller-name:9696
neutron_admin_username=neutron
neutron_admin_password=neutron-password
neutron_admin_tenant_name=service
neutron_admin_auth_url=http://controller-name:5000/v2.0
rabbit_host=controller-name
```

```
connection=mysql://nova:nova@controller-name/nova
```

6. **コンピュートノードに EVS パッケージをインストールします。**

```
# pkg install evs
```

7. **コンピュートノード上に root ユーザーの SSH 公開鍵を作成します。**

この鍵は Solaris ゾーンフレームワークが EVS コントローラにアクセスできるようにします。

root ユーザーとして ssh-keygen コマンドを使用して、root ユーザーの鍵を作成します。

```
compute1# su - root -c "ssh-keygen -N '' -f /root/.ssh/id_rsa -t rsa"
```

8. **EVS コントローラノードに鍵をコピーします。**

前のステップで生成された SSH 公開鍵 /root/.ssh/id_rsa.pub を EVS コントローラが実行しているノード (この構成ではネットワークノード) にコピーします。この鍵はネットワークノードに EVS コントローラを構成する際に参照されます。

この手順のあとで、Glance イメージを登録できます。120 ページの「イメージの作成」および120 ページの「イメージストアへのイメージの追加」を参照してください。

9. **Nova コンピュートサービスを有効にします。**

```
compute1# svcadm enable nova-compute
```

ネットワークノードの構成

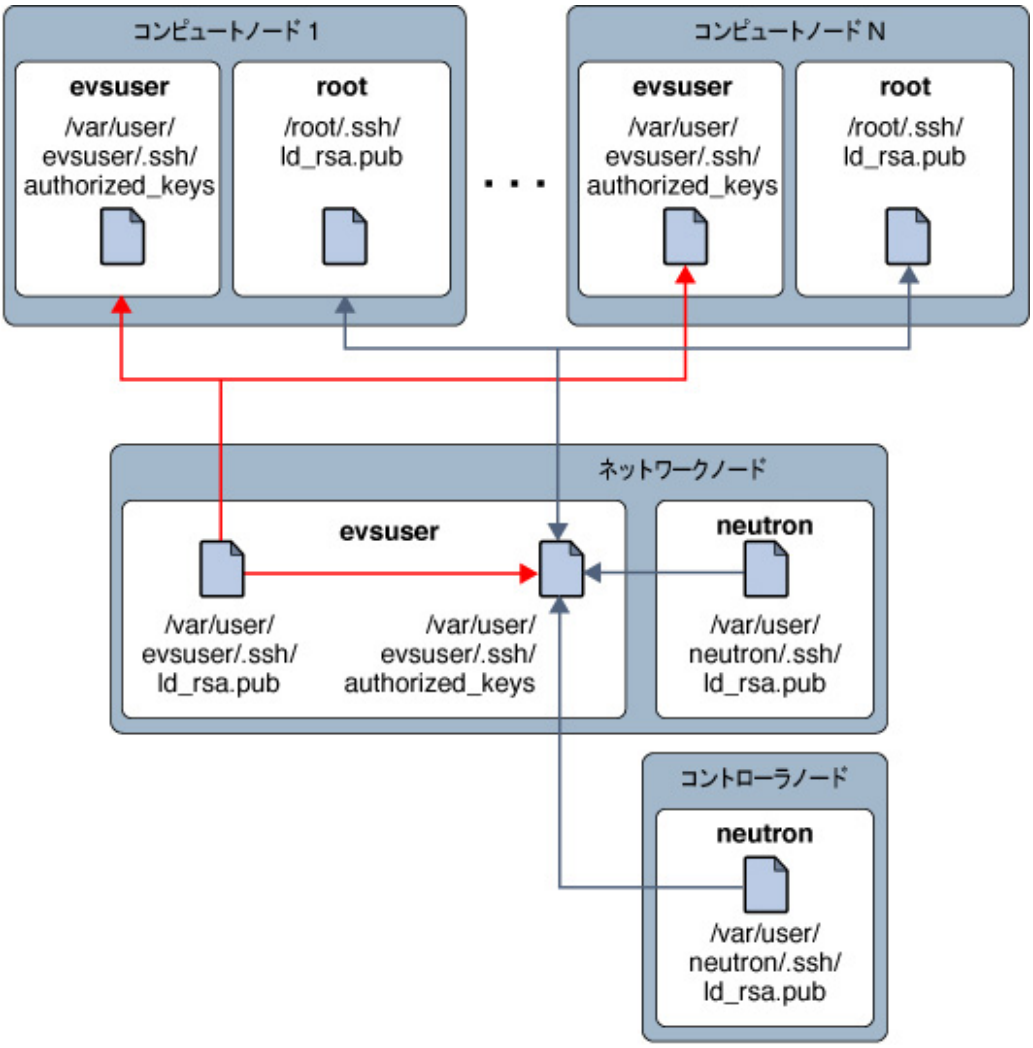
ネットワークノードの構成では、Elastic Virtual Switch (EVS) と Neutron DHCP エージェントの両方を構成する必要があります。オプションで Neutron Layer (L3) エージェントを構成できます。

EVS は OpenStack ネットワークのバックエンドを形成し、VLAN または VXLAN を使用して、VM インスタンス間の通信を容易にします。VM インスタンスは、同じコンピュートノードまたは複数のコンピュートノードに配置できます。EVS の詳細については、『Oracle Solaris 11.2 での仮想ネットワークとネットワークリソースの管理 』の第 5 「エラスティック仮想スイッチについて」を参照してください。

ネットワークノードを構成するときは、必ず evsuser の SSH 公開鍵をコンピュートノードとネットワークノードにある evsuser の authorized_keys ファイルのそれぞれにコピーしてください。

SSH 公開鍵の配布を示す次のイメージを参照してください。このイメージでは、複数のコンピュートノードが構成されていると仮定しています。

図 3-2 EVS コントローラの SSH 鍵の配布



Oracle Solaris 11 で ZFS とアプリケーションの間のメモリー使用をより効率的に管理するには、次の例に示すように、ノード上で `usr_reserve_hint_pct` パラメータを設定します。

```
# echo "set user_reserve_hint_pct=80" >>/etc/system.d/site:kernel-zones-reserve
```

```
# reboot
```

ここで、*site* にはユーザーの会社を指定することもあります。

ほかの OpenStack ノードでも同様にこのパラメータを設定するようにしてください。

このパラメータの詳細については、https://support.oracle.com で MOS にログインし、*Oracle Solaris 11.2 での ZFS とアプリケーションの間のメモリー管理*に関するドキュメント 1663862.1 を確認してください。

▼ ネットワークノードを構成する方法

1. **(オプション) NTP をインストールし、構成します。**

   47 ページの「Network Time Protocol のインストール」を参照してください。

2. **Neutron パッケージをインストールします。**

   ```
   network# pkg install neutron
   ```

3. **ネットワークノード上に、neutron ユーザーと evsuser ユーザーの SSH 公開鍵を作成します。**

   ```
   network# su - neutron \
   -c "ssh-keygen -N '' -f /var/lib/neutron/.ssh/id_rsa -t rsa"

   network# su - neutron \
   -c "ssh-keygen -N '' -f /var/lib/neutron/.ssh/id_rsa -t rsa"
   ```

4. **鍵ファイルを EVS 認証鍵ファイルに組み合わせます。**

   以前にコントローラノードとコンピュートノードから作成された SSH 公開鍵をこれらの 2 つの新しい鍵と組み合わせ、evsuser の認可された鍵ファイルにこの組み合わせを連結します。

   ```
   network# cat \
   path-to-neutron@controller/id_rsa.pub
    \
   path-to-root@compute1/id_rsa.pub \
   /var/lib/neutron/.ssh/id_rsa.pub \
   /var/user/evsuser/.ssh/id_rsa.pub \
   >> /var/user/evsuser/.ssh/authorized_keys
   ```

5. **evsuser の SSH 公開鍵 /var/user/evsuser/.ssh/id_rsa.pub) をコンピュートノードとネットワークノードにある evsuser の authorized_keys ファイルのそれぞれにコピーします。**

クラウド構成全体のコンテキストにおける evsuser の SSH 公開鍵の配布の概要については、図3-2「EVS コントローラの SSH 鍵の配布」を参照してください。

6. **SSH 接続が正しく動作しているか確認します。**

   ssh コマンドを使用して、EVS コントローラに接続し、接続ごとにプロンプトが表示されたら yes と答えます。OpenStack アーキテクチャーの各ノードでこのステップを実行します。

```
controller# su - neutron -c "ssh evsuser@network-name whoami"
compute1# su - root -c "ssh evsuser@network-name whoami"
network# su - neutron -c "ssh evsuser@network-name whoami"
network# su - root -c "ssh evsuser@network-name whoami"
```

7. **EVS コントローラパッケージをインストールします。**

```
network# pkg install rad-evs-controller
```

8. **RAD を再起動します。**

```
network# svcadm restart rad:local
```

9. **EVS コントローラを構成します。**

   Nova インスタンスが通信する仮想ネットワークは VLAN にするかまたは VXLAN にするかを決定します。次の例では、次の VLAN 構成を行います。

   - ID が 13 の VLAN をクラウド外部との接続に使用する外部ネットワーク用に構成します。
   - 1000 から 2000 の VLAN ID の範囲を EVS によって作成されるサブネット用に構成します。

```
network# evsadm set-prop -p controller=ssh://evsuser@network-name
network# evsadm
network# evsadm set-controlprop -p l2-type=vlan
network# evsadm set-controlprop -p uplink-port=net1
network# evsadm set-controlprop -p vlan-range=13,1000-2000
```

10. **認証パラメータと RabbitMQ サービスの場所を構成します。**

    /etc/neutron/neutron.conf ファイル内の次のパラメータをコメント解除し、設定します。これらのパラメータは、追加の Keystone 認証サービスエンドポイント、Glance API サービス、Neutron API サービス、RabbitMQ サービス、および Neutron 固有のデータベースのデータベース URI を指定します。

```
[keystone_authtoken]
signing_dir = /var/lib/neutron/keystone-signing
auth_host = 127.0.0.1
auth_port = 35357
```

```
auth_protocol = http
auth_uri = http://controller-IP:5000/v2.0
admin_tenant_name = service
admin_user = neutron
admin_password = neutron-password
identity_uri = http://controller-IP:35357
.
[DEFAULT]
core_plugin = neutron.plugins.evs.plugin.EVSNeutronPluginV2
allow_overlapping_ips = False

quotas]
quota_driver = neutron.plugins.evs.db.quotas_db.EVSDbQuotaDriver
```

11. **Neutron DHCP エージェントを構成します。**

    /etc/neutron/dhcp_agent.ini ファイル内の次のパラメータをコメント解除し、設定して、EVS コントローラの場所を指定します。

    ```
    evs_controller = ssh://evsuser@network-name
    ```

12. **(オプション) Nova インスタンスによって DNS リクエストを解決するために使用するデフォルトのドメインを指定します。**

    /etc/neutron/dhcp_agent.ini ファイルで、dhcp_domain パラメータのコメントを解除して、Nova インスタンスによって DNS リクエストを解決するために使用するデフォルトのドメインに設定します。

13. **DHCP エージェントを有効にします。**

    ```
    network# svcadm enable neutron-dhcp-agent
    ```

14. **Neutron L3 エージェントを構成します。**

    このステップはオプションですが強くお勧めします。

Neutron L3 エージェントの構成

このセクションでは、外部ネットワークを表す仮想ネットワークを作成する方法を示します。この仮想ネットワークでは DHCP が使用されません。代わりにフローティング IP アドレスが作成されます。これらのフローティング IP アドレスは特定のテナントに割り当てられ、そのテナントで、ユーザーによって使用される Nova VM インスタンスに割り当てることができます。Neutron L3 エージェントは、Nova インスタンスに割り当てられたアドレスとフローティング IP アドレス間の 1 対 1 の NAT マッピングを自動的に作成します。

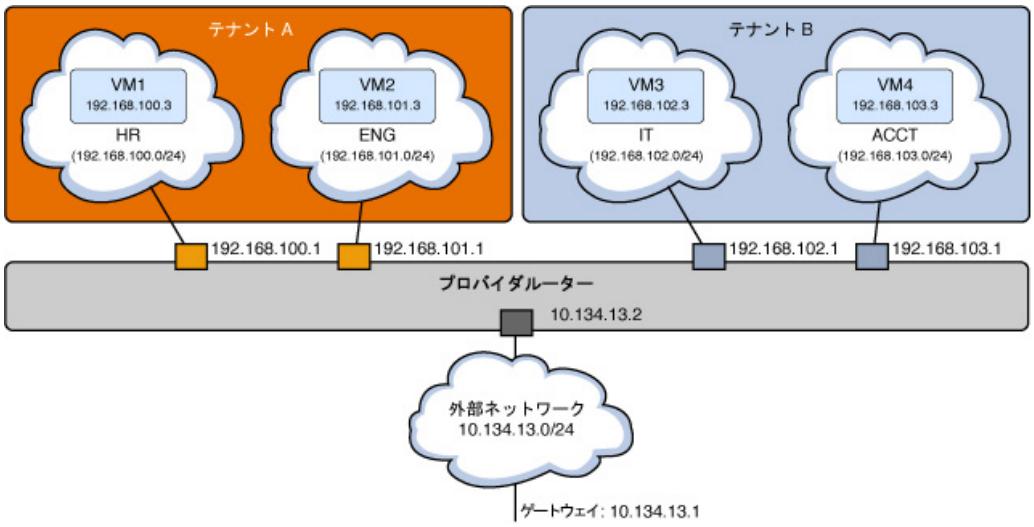
OpenStack Neutron の Oracle Solaris 11.2 実装は、プライベートネットワークデプロイメントモデルによって、プロバイダルーターをサポートしています。このモデルの詳細は、『*運用者トレーニングガイド*』の第 7 にある OpenStack Neutron のユースケースを参照してください。このデプロイメントモデルで、各テナントは 1 つ以上のプライベートネットワークを使用でき、すべてのテナントネットワークが同じルーターを共有します。このルーターはデータ管理者によって作成、所有、および管理されます。ルーターはテナントのネットワークトポロジビューに表示されません。単一のルーターしかないため、テナントネットワークでは重複する IP アドレスを使用できません。管理者はテナントの代わりにプライベートネットワークを作成します。

デフォルトで、このモデル内のルーターは同じテナントに含まれるプライベートネットワーク間のルーティングを妨げます。あるプライベートネットワーク内の VM インスタンスは別のプライベートネットワーク内の VM インスタンスと、それらがすべて同じテナントに含まれる場合でも通信できません。この動作を変更するには、`/etc/neutron/l3_agent.ini` 構成ファイルで `allow_forwarding_between_networks` を `True` に設定し、`neutron-l3-agent` SMF サービスを再起動します。

このモデルのルーターはテナントの VM インスタンスの外部への接続を提供します。ルーターは、外部ネットワークにルーターを接続するインタフェース上で双方向の NAT を実行します。テナントは、それらが必要なだけ、またはフローティング IP の割り当て制限によって許可されるだけの数のフローティング IP (パブリック IP) を作成し、これらのフローティング IP を、外部接続を必要とする VM インスタンスに関連付けます。

次の図に、Oracle Solaris 11.2 Neutron デプロイメントモデルを示します。図の説明は図の後にあります。

図 3-3　プライベートネットワークモデルでのプロバイダルーター



前の図に示したモデルで、各テナントには 2 つの内部ネットワークと 2 つの VM インスタンスがあります。

テナント

- サブネット 192.168.100.0/24 とゲートウェイ 192.168.100.1 の HR ネットワーク
- サブネット 192.168.101.0/24 とゲートウェイ 192.168.101.1 の ENG ネットワーク
- 192.168.100.3 の固定 IP アドレスを持つ HR に接続された VM1
- 192.168.101.3 の固定 IP アドレスを持つ ENG に接続された VM2

テナント

- サブネット 192.168.102.0/24 とゲートウェイ 192.168.102.1 の IT ネットワーク
- サブネット 192.168.103.0/24 とゲートウェイ 192.168.103.1 の ACCT ネットワーク
- 192.168.102.3 の固定 IP アドレスを持つ IT に接続された VM3
- 192.168.103.3 の固定 IP アドレスを持つ ACCT に接続された VM4

双方向 NAT テーブルは次の表に示すように構成されます。

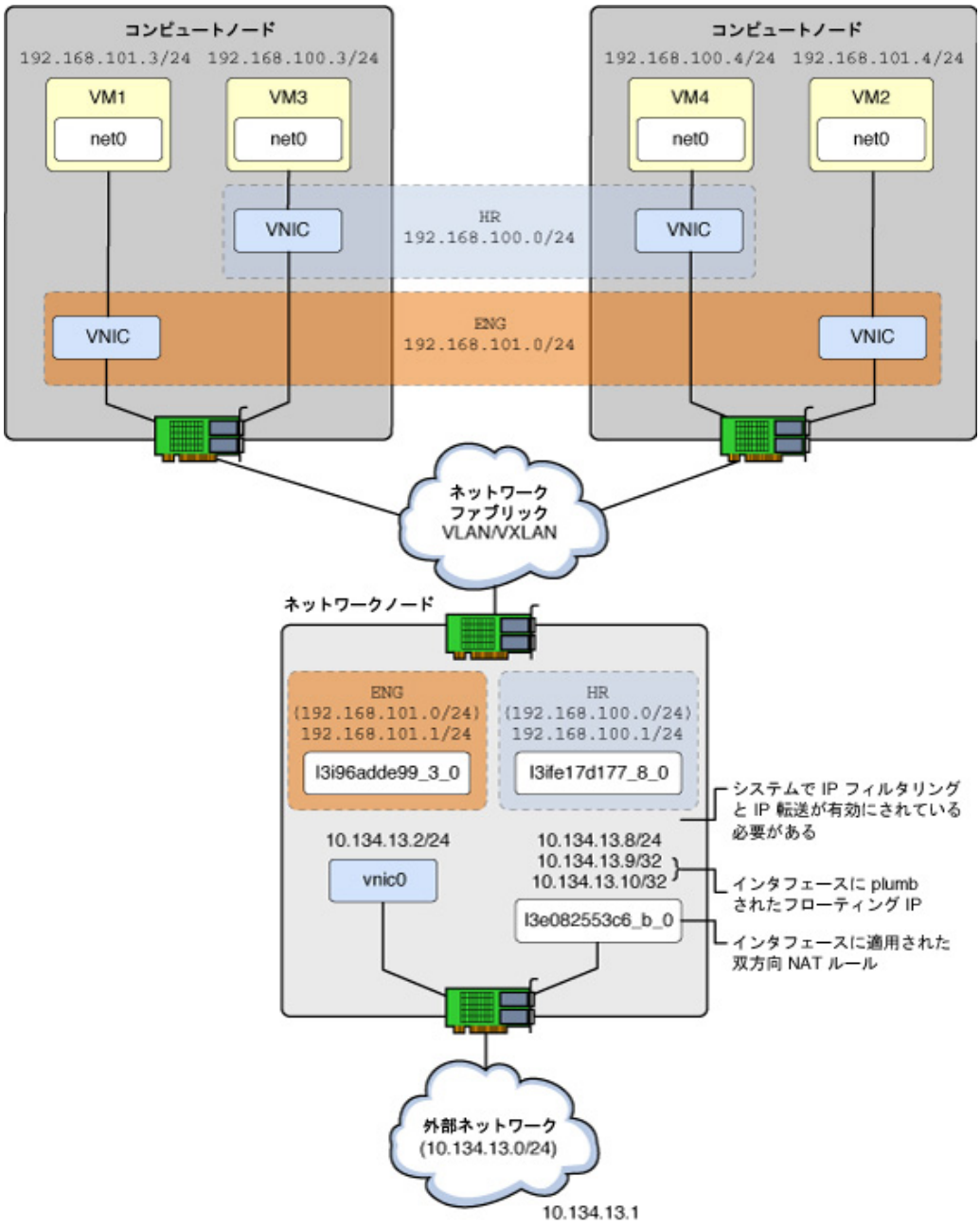
内部 IP	フローティング IP
192.168.100.3	10.134.13.40
192.168.101.3	10.134.13.9

すべてのゲートウェイインタフェースが `neutron-l3-agent` SMF サービスを実行しているノードでインスタンス化されます。クラウドでは多くの Neutron インスタンスを使用できますが、クラウドあたりに必要な `neutron-l3-agent` は 1 つだけです。

外部ネットワークは、外部から到達可能なサブネット 10.134.13.0/24 24 に関連付けられたプロバイダネットワークです。テナントはこのネットワークからフローティング IP アドレスを作成し、それらを VM インスタンスに関連付けます。VM1 と VM2 にはフローティング IP 10.134.13.40 と 10.134.13.9 がそれぞれ関連付けられています。VM1 と VM2 はこれらの IP アドレスによって、外部から到達可能です。

次のダイアグラムに、これらのテナントリソースがネットワークノードおよび 2 つのコンピュートノードにどのように構成されているかを示します。図に示している機能の一部の説明は図の後にあります。

図 3-4 Neutron L3 エージェント構成



VNIC	仮想ネットワークインタフェース。
`l3e...`	双方向 NAT が発生する外部 (「e」) ネットワーク上に VNIC を作成した L3 エージェント。
`l3i...`	デフォルトのゲートウェイ IP アドレスを持つ内部 (「i」) ネットワーク上に VNIC を作成した L3 エージェント。

次のリストに、この例の構成で IP アドレスがどのように使用されるかを示します。

10.134.13.1
: デフォルトゲートウェイ

10.134.13.2 - 10.134.13.7
: OpenStackAPI (Nova、Cinder、Glance など) をテナントに公開するために確保されている IP アドレス

10.134.13.9 - 10.134.13.254
: テナント VM インスタンス用のフローティング IP アドレス

▼ Neutron L3 エージェントを構成する方法

この手順では、`service` テナントを使用して、ルーター、外部ネットワーク、およびデータセンター内のすべてのテナントによって使用される外部サブネットを作成する方法を示します。

この手順は、データセンターの管理者によって実行されます。OpenStack ダッシュボードでは一度に 1 つのテナントのリソースしか管理できないため、単一の共有ルーターを構成し、異なるテナントのネットワークとサブネットを関連付けるには、コマンド行を使用する必要があります。

次の手順を実行する場合は、図3-4「Neutron L3 エージェント構成」を参照してください。

始める前に この手順を実行する前に、内部ネットワークの構成を完了する必要があります。

1. **Solaris IP フィルタ機能を有効にします。**

```
network# svcadm enable ipfilter
```

2. **ホスト全体で IP 転送を有効にします。**

```
network# ipadm set-prop -p forwarding=on ipv4
network# ipadm set-prop -p forwarding=on ipv6
```

3. **EVS が正しく構成されており、外部ネットワークに必要な VLAN ID を持つことを確認します。**

次の例の VLAN ID と範囲値は、ステップ 9 で完了した EVS 構成に基づいています。

```
network# evsadm show-controlprop -p vlan-range,l2-type
PROPERTY      PERM    VALUE         DEFAULT    HOST
l2-type       rw      vlan          vlan       --
vlan-range    rw      13,1000-2000  --         --
```

4. **service テナントが存在することを確認します。**

```
network# keystone tenant-list
```

5. **プロバイダルーターを作成します。**

プロバイダルーターは service テナントに、OpenStack neutron ユーザーとして作成します。

新しいルーターの UUID id) をメモします。これは次のステップで使用します。

```
network# export OS_USERNAME=neutron
network# export OS_PASSWORD=neutron-password
network# export OS_TENANT_NAME=service
network# export OS_AUTH_URL=http://controller-name:5000/v2.0
network# neutron router-create provider_router
Created a new router:
+-----------------------+--------------------------------------+
| Field                 | Value                                |
+-----------------------+--------------------------------------+
| admin_state_up        | True                                 |
| external_gateway_info |                                      |
| id                    | 181543df-40d1-4514-ea77-fddd78c389ff |
| name                  | provider_router                      |
| status                | ACTIVE                               |
| tenant_id             | f164220cb02465db929ce520869895fa     |
+-----------------------+--------------------------------------+
```

6. **L3 エージェント構成ファイルを更新します。**

前のステップからのルーター UUID id) を使用して、/etc/neutron/l3_agent.ini ファイル内の router_id の値を更新します。

```
router_id = 181543df-40d1-4514-ea77-fddd78c389ff
```

7. **neutron-l3-agent SMF サービスを有効にします。**

```
network# svcadm enable neutron-l3-agent
```

8. **外部ネットワークを作成します。**

仮想ネットワークは service テナントに、OpenStack neutron ユーザーとして作成します。

```
network# neutron net-create --provider:network_type=vlan
```

```
--provider:segmentation_id=13 --router:external=true external_network
Created a new network:
+--------------------------+--------------------------------------+
| Field                    | Value                                |
+--------------------------+--------------------------------------+
| admin_state_up           | True                                 |
| id                       | f67f0d72-0ddf-11e4-9d95-e1f29f417e2f |
| name                     | external_network                     |
| provider:network_type    | vlan                                 |
| provider:segmentation_id | 13                                   |
| router:external          | True                                 |
| shared                   | False                                |
| status                   | ACTIVE                               |
| subnets                  |                                      |
| tenant_id                | f164220cb02465db929ce520869895fa     |
+--------------------------+--------------------------------------+
```

9. **外部ネットワークにサブネットを関連付けます。**

   外部ネットワークに関連付けられるサブネットネットワークを作成し、DHCP が無効であることとサブネットが外部ルーティングに使用されることを指定します。フローティング IP アドレスが割り当てられるルート可能 IP サブネットを指定します。この例では、このサブネットは VLAN ID 13 に関連付けられています。

```
network# neutron subnet-create  --enable-dhcp=False \
--allocation-pool start=10.134.13.8,end=10.134.13.254 \
--name external_subnet  external_network 10.134.13.0/24
Created a new subnet:

+------------------+--------------------------------------------------+
| Field            | Value                                            |
+------------------+--------------------------------------------------+
| allocation_pools | {"start": "10.134.13.8", "end": "10.134.13.254"} |
| cidr             | 10.134.13.0/24                                   |
| dns_nameservers  |                                                  |
| enable_dhcp      | False                                            |
| gateway_ip       | 10.134.13.1                                      |
| host_routes      |                                                  |
| id               | 5d9c8958-0de0-11e4-9d96-e1f29f417e2f             |
| ip_version       | 4                                                |
| name             | external_subnet                                  |
| network_id       | f67f0d72-0ddf-11e4-9d95-e1f29f417e2f             |
| tenant_id        | f164220cb02465db929ce520869895fa                 |
+------------------+--------------------------------------------------+
```

10. **ルーターに外部ネットワークを追加します。**

    次のコマンドで、最初の UUID は `provider_router` UUID で、2 番目の UUID は `external_network` UUID です。

```
network# neutron router-gateway-set
181543df-40d1-4514-ea77-fddd78c389ff
```

```
f67f0d72-0ddf-11e4-9d95-e1f29f417e2f
Set gateway for router 181543df-40d1-4514-ea77-fddd78c389ff
network# neutron router-list -c name -c external_gateway_info
+----------------+-------------------------------------------------------+
| name           | external_gateway_info                                 |
+----------------+-------------------------------------------------------+
| provider_router | {"network_id": "f67f0d72-0ddf-11e4-9d95-e1f29f417e2f"} |
+----------------+-------------------------------------------------------+
```

11. **ルーターにテナントのプライベートネットワークを追加します。**

    neutron net-list コマンドによって表示されるネットワークは、以前に構成されました。

```
network# keystone tenant-list
+----------------------------------+---------+---------+
|                id                |   name  | enabled |
+----------------------------------+---------+---------+
| 511d4cb9ef6c40beadc3a664c20dc354 |   demo  |   True  |
| f164220cb02465db929ce520869895fa | service |   True  |
+----------------------------------+---------+---------+
network# neutron net-list --tenant-id=511d4cb9ef6c40beadc3a664c20dc354
+------------------------------+------+------------------------------+
| id                           | name | subnets                      |
+------------------------------+------+------------------------------+
| c0c15e0a-0def-11e4-9d9f-     | HR   | c0c53066-0def-11e4-9da0-     |
|  e1f29f417e2f                |      | e1f29f417e2f 192.168.100.0/24|
| ce64b430-0def-11e4-9da2-     | ENG  | ce693ac8-0def-11e4-9da3-     |
|  e1f29f417e2f                |      | e1f29f417e2f 192.168.101.0/24|
+------------------------------+------+------------------------------+
```

    次のコマンドで、最初の UUID は provider_router UUID で、2 番目の UUID は HR サブネット UUID です。

```
network# neutron router-interface-add
181543df-40d1-4514-ea77-fddd78c389ff
c0c53066-0def-11e4-9da0-e1f29f417e2f       (HR subnet UUID)
Added interface 7843841e-0e08-11e4-9da5-e1f29f417e2f to router 181543df-40d1-4514-ea77-
fddd78c389ff.
```

    次のコマンドで、最初の UUID は provider_router UUID で、2 番目の UUID は ENG サブネット UUID です。

```
network# neutron router-interface-add
181543df-40d1-4514-ea77-fddd78c389ff
ce693ac8-0def-11e4-9da3-e1f29f417e2f
Added interface 89289b8e-0e08-11e4-9da6-e1f29f417e2f to router 181543df-40d1-4514-ea77-
fddd78c389ff.
```

参照

- 77 ページの「L3 エージェント構成を監視する方法」
- 125 ページの「既知の制限事項」

▼ フローティング IP アドレスをテナントユーザーとして作成および関連付ける方法

この手順は OpenStack Horizon ダッシュボードを使用してテナントユーザーによって実行します。

1. **OpenStack ダッシュボードにログインします。**

   テナントユーザーの資格証明を使用して、29 ページの「OpenStack ダッシュボードにアクセスする方法」の説明に従って、ログインします。

2. **「プロジェクト」 -> 「アクセスとセキュリティ」 -> 「Floating IP」を選択します。**

3. **`external_network` を選択します。**

4. **「IP の確保」ボタンをクリックします。**

   「Floating IP」タブに、フローティング IP アドレス 10.134.13.9 が割り当てられていることが示されます。

5. **「割り当て」ボタンをクリックします。**

6. **プルダウンメニューから VM インスタンスのポートを選択します。**

   「プロジェクト」 -> 「インスタンス」ウィンドウにはフローティング IP アドレスが VM インスタンスに関連付けられていることが示されます。

   VM インスタンスの起動中に、鍵ペア (SSH 公開鍵) を選択した場合、その SSH 鍵は VM インスタンスの `root` ユーザーの `authorized_keys` ファイルに追加されます。

7. **実行中の VM インスタンスにログインします。**

```
global# ssh root@10.134.13.9
Last login: Fri Jul 18 00:37:39 2014 from 10.132.146.13
Oracle Corporation      SunOS 5.11      11.2     June 2014
root@host-192-168-101-3:~# uname -a
SunOS host-192-168-101-3 5.11 11.2 i86pc i386 i86pc
root@host-192-168-101-3:~# zoneadm list -cv
ID NAME              STATUS      PATH                 BRAND      IP
 2 instance-00000001 running     /                    solaris    excl
root@host-192-168-101-3:~# ipadm
NAME             CLASS/TYPE STATE     UNDER      ADDR
lo0              loopback   ok        --         --
   lo0/v4        static     ok        --         127.0.0.1/8
   lo0/v6        static     ok        --         ::1/128
net0             ip         ok        --         --
```

```
   net0/dhcp     inherited  ok        --         192.168.101.3/24
```

▼ L3 エージェント構成を監視する方法

ipf、ippool、および ipnat などの IP フィルターコマンドおよび dladm や ipadm などのネットワークコマンドを使用すると、neturon-l3-agent によって実行された構成を監視し、トラブルシューティングすることができます。

1. **neutron-l3-agent によって作成された VNIC を表示します。**

```
network# dladm show-vnic
LINK                OVER         SPEED  MACADDRESS        MACADDRTYPE VIDS
l3i7843841e_0_0     net1         1000   2:8:20:42:ed:22   fixed       200
l3i89289b8e_0_0     net1         1000   2:8:20:7d:87:12   fixed       201
l3ed527f842_0_0     net0         100    2:8:20:9:98:3e    fixed
```

2. **neutron-l3-agent によって作成された IP アドレスを表示します。**

```
network# ipadm
NAME                 CLASS/TYPE STATE   UNDER      ADDR
l3ed527f842_0_0      ip         ok      --         --
  l3ed527f842_0_0/v4 static     ok      --         10.134.13.8/24
  l3ed527f842_0_0/v4a static    ok      --         10.134.13.9/32
l3i7843841e_0_0      ip         ok      --         --
  l3i7843841e_0_0/v4 static     ok      --         192.168.100.1/24
l3i89289b8e_0_0      ip         ok      --         --
  l3i89289b8e_0_0/v4 static     ok      --         192.168.101.1/24
```

3. **IP フィルタルールを表示します。**

```
network# ipfstat -io
empty list for ipfilter(out)
block in quick on l3i7843841e_0_0 from 192.168.100.0/24 to pool/4386082
block in quick on l3i89289b8e_0_0 from 192.168.101.0/24 to pool/8226578
network# ippool -l
table role = ipf type = tree number = 8226578
{ 192.168.100.0/24; };
table role = ipf type = tree number = 4386082
{ 192.168.101.0/24; };
```

4. **IP NAT ルールを表示します。**

```
network# ipnat -l
List of active MAP/Redirect filters:
bimap l3ed527f842_0_0 192.168.101.3/32 -> 10.134.13.9/32
List of active sessions:
BIMAP 192.168.101.3  22  <- -> 10.134.13.9  22 [10.132.146.13 36405]
```

♦ ♦ ♦ 第 4 章

4 マルチノード Juno OpenStack 構成での複数システムにまたがるインストール

この章では、マルチノード OpenStack 構成のインストール方法について説明します。シングルノードインストールについては、第2章「評価構成のインストール」を参照してください。

この章の内容は次のとおりです。

注記 - この章は、Oracle Solaris 11.2 SRU10 リリース以降でのみサポートされる Juno 固有の OpenStack 構成に適用されます。

- Oracle Solaris 11.2 SRU10 リリースの入手および既存の Havana 構成の Juno へのアップグレードについては、Havana から Juno へ OpenStack をアップグレードする手順を参照してください。
- Oracle Solaris 11.2 SRU10 リリースを実行していない場合は、代わりに Havana 構成の指示を使用してください。第3章「マルチノード Havana OpenStack 構成での複数システムにまたがるインストール」を参照してください

3 ノードアーキテクチャーの概要

通常、OpenStack のインストールと構成は複数のシステムやノードをまたいで行います。シングルノード構成は、OpenStack を製品としてテストし、その機能を理解するために役立ちます。ただし、シングルノード構成は本番環境には適していません。

各クラウドに、1 つのダッシュボードインスタンス、1 つのイメージストア、および 1 つのアイデンティティーサービスのみが必要です。各クラウドでは任意の数のストレージとコンピュートインスタンスを使用できます。本番環境では、これらのサービスは複数のノード全体に構成します。特定のクラウドデプロイメントのニーズに関連して、各コンポーネントを評価し、そのコンポーネントを個別のノードにインストールすべきかどうか、およびそのタイプのノードをどのくらい必要とするかを決定します。

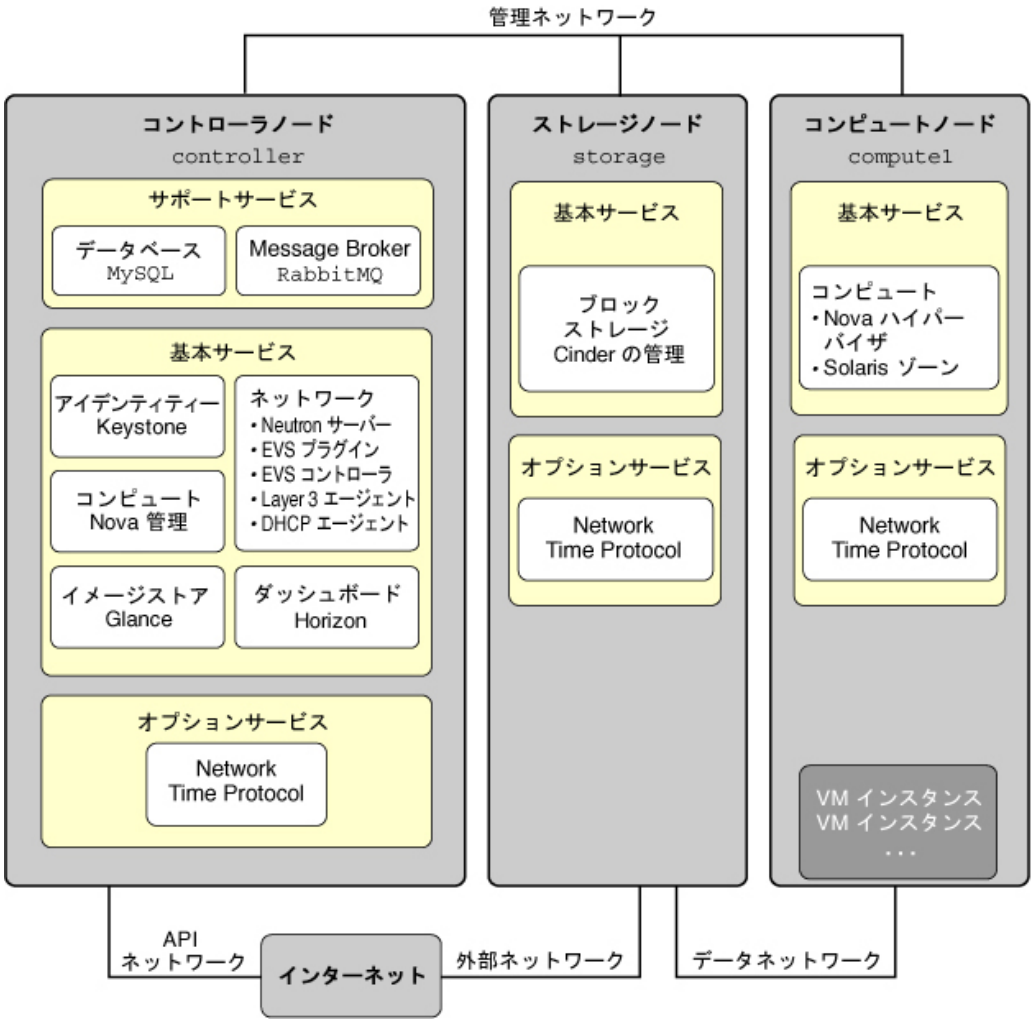
- コントローラノード – ほとんどの共有 OpenStack サービスおよびその他のツールが実行されるノード。コントローラノードはクラウドに API、スケジュール、およびその他の共有サービスを提供します。コントローラノードには、ダッシュボード、イメージストア、およびアイデンティティーサービスがあります。さらに、Nova コンピュート管理サービスと Neutron サーバーもこのノードに構成されます。
- コンピュートノード – VM インスタンス (Nova コンピュートインスタンス) がインストールされているノード。このノードは、これらの VM インスタンスを管理するコンピュートデーモンを実行します。
- ストレージノード – データをホストするノード。

この章で説明するアーキテクチャーは次の 3 つのシステムにデプロイされています。

注記 - このドキュメントでは、個別の物理システムにデプロイするアーキテクチャーについて説明します。単一の Oracle SPARC サーバーをパーティション分割し、OVM Server for SPARC (LDoms) を実行するサーバーにマルチノード OpenStack を構成するには、SPARC サーバー上のマルチノード Solaris 11.2 OpenStack に関する記事を参照してください。この記事は Havana バージョンの OpenStack 向けに書かれています。ただし、一般的な手順は現在のバージョンにも適用されます。

次の図に、この章で説明するアーキテクチャーの概要図を示します。

図 4-1 3 ノード構成のリファレンスアーキテクチャー



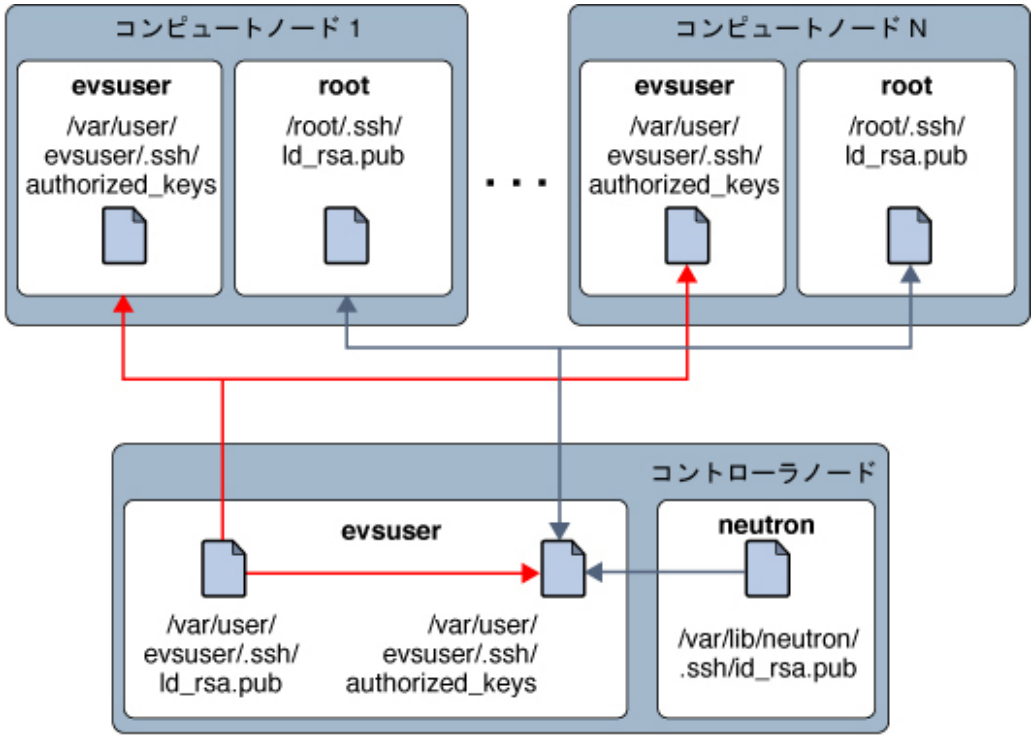
この例のアーキテクチャーでは、Swift オブジェクトストレージサービスを示していません。Swift の構成に関する一般的な情報については、OpenStack 構成参照などの OpenStack コミュニティサイトの情報を参照してください。Oracle Solaris システムで Swift サービスを構成する方法および、Oracle Solaris 上の OpenStack に関するほかの情報については、OpenStack for Oracle Solaris 11 に関するドキュメントを参照してください。

Oracle Solaris では、Elastic Virtual Switch (EVS) が OpenStack ネットワーキングのバックエンドを形成します。EVS は、VLAN または VXLAN 上の VM インスタンス間の通信を容易にします。VM インスタンスは、同じコンピュートノードまたは複数のコンピュートノードに配置できます。EVS の詳細については、『Oracle Solaris 11.2 での仮想ネットワークとネットワークリソースの管理 』の第 5 「エラスティック仮想スイッチについて」を参照してください。

異なるノード同士が通信するには、コントローラノードにある `evsuser`、`neutron`、および `root` の SSH 公開鍵が、構成されたすべてのコンピュートノード内の `evsuser` の `authorized_keys` ファイルそれぞれに格納されている必要があります。SSH 公開鍵の配布を示す次のイメージを参照してください。このイメージでは、複数のコンピュートノードが構成されていると仮定しています。

Oracle Solaris システムへの OpenStack のデプロイメントに役立つ OpenStack 構成パラメータのリストについては、`http://www.oracle.com/technetwork/articles/servers-storage-admin/getting-started-openstack-os11-2-2195380.html` の OpenStack の一般的な構成パラメータに関するセクションを参照してください。

図 4-2 EVS コントローラの SSH 鍵の配布



コントローラノードの構成

コントローラノードには 1 つのダッシュボードサービス、1 つのイメージストア、および 1 つのアイデンティティーサービスがあります。このノードには、MySQL、RabbitMQ、およびコンピュート、ブロックストレージ、ネットワークの各サービスも含まれています。

次のリストは、コントローラノードを構成するためのタスクを示しています。

- 84 ページの「準備の手順」
- 85 ページの「Network Time Protocol のインストール」
- 86 ページの「MySQL のインストール」
- 88 ページの「Keystone のインストール」

準備の手順

マルチノード OpenStack 構成の実装を準備するには、次の点に注意してください。

- 単一の OpenStack ノードまたはシステムで複数のネットワークインタフェースを使用する場合は、これらのネットワークインタフェース用にホスト名を作成します。

  たとえば、OpenStack の管理および API トラフィック (OpenStack ネットワーク)、コンピュートノードと L3 ルーターの間のトラフィック (テナントネットワーク)、およびクラウド構成の外側のより大きなネットワークのトラフィック (外部ネットワーク) を処理する個別のインタフェースがある場合があります。そのため、`host-on`、`host-tn`、`host-en` のように、それぞれに対応するホスト名を作成します。

  これらすべてのホスト名および IP アドレスが、ノードの `/etc/hosts` ファイルまたは DNS 構成に含まれていることを確認します。
- OpenStack サービスの構成を容易にするために、次のような変数を設定します。
  - `$CONTROLLER_ADMIN_NODE` - OpenStack 管理サービスが接続されているコントローラノード内のインタフェースまたは IP アドレスのホスト名。
  - `$CONTROLLER_ADMIN_NODE_IP` - OpenStack 管理サービスおよびトラフィックを処理するコントローラポートの IP アドレス。
  - `$COMPUTE_ADMIN_NODE_IP` - OpenStack 管理サービスおよびトラフィックを処理するコンピュートポートの IP アドレス。
  - `$VOLUME_IP` - ストレージノードのホスト名。
- 必要に応じてパスワードが割り当て可能であることも確認してください。

Oracle Solaris 11 で ZFS とアプリケーションの間のメモリー使用をより効率的に管理するには、次の例に示すように、ノード上で `usr_reserve_hint_pct` パラメータを設定します。

```
# echo "set user_reserve_hint_pct=80" >>/etc/system.d/site:kernel-zones-reserve
# reboot
```

ここで、*site* にはユーザーの会社を指定することもあります。

ほかの OpenStack ノードでも同様にこのパラメータを設定するようにしてください。

このパラメータの詳細については、https://support.oracle.com で MOS にログインし、*Oracle Solaris 11.2 での ZFS とアプリケーションの間のメモリー管理*に関するドキュメント 1663862.1 を確認してください。

Network Time Protocol のインストール

Network Time Protocol (NTP) のインストールはオプションですが強くお勧めします。NTP をインストールする場合、クラウドデプロイメント内の各サービスノードにインストールします。

NTP はクラウド内のすべてのサービスノード全体で一貫した時間を確保するのに役立ちます。ネットワークで NTP を有効にする場合、サービスノードがネットワークを介して時間を取得するように構成します。

- サービスノードが存在する IP サブネットで IP マルチキャストが有効にされている場合、IP マルチキャストを利用して、NTP を構成できます。
- サービスノードが存在する IP サブネットで IP マルチキャストが有効にされていない場合は、NTP を手動で構成します。

▼ Network Time Protocol をインストールし、構成する方法

始める前に NTP のマルチキャストを使用しない場合は、NTP サーバーを構成する必要があります。

1. **NTP パッケージをインストールします。**

```
controller# pkg install ntp
```

2. **構成ファイルをインストールします。**

```
controller# cp /etc/inet/ntp.client /etc/inet/ntp.conf
```

3. **(オプション) マルチキャストを使用する場合は `/etc/inet/ntp.conf` ファイルを構成します。**

   a. **`multicastclient` オプションをコメントアウトします。**

   b. **1 つまたは複数のサーバーオプションをコメント解除して、NTP サーバーの特定の名前またはその IP アドレスを指定します。**

構成ファイルは次の例のようになります。

```
# multicastclient 224.0.1.1
...
server system1.example.com iburst
server system2.example.com iburst
# server server_name3 iburst
```

4. **NTP サーバーの SMF サービスを有効にします。**

```
controller# svcadm enable ntp
```

MySQL のインストール

多くの OpenStack サービスは、重要なリソース、使用状況、およびその他の情報を追跡するためにデータベースを保持します。デフォルトでは個々の SQLite データベースがこの目的で指定されますが、単一ノード構成の場合に便利です。特に本番環境でマルチノード構成を使用する場合は、この情報の格納に MySQL データベースなどの別のデータベースを使用することをお薦めします。

OpenStack サービス間の通信は Advanced Message Queuing Protocol (AMQP) によって実行されます。Oracle Solaris で AMQP は RabbitMQ によって実装されます。RabbitMQ は必須サービスです。通常、クラウド内の 1 つのノードで RabbitMQ を実行するように構成します。このアーキテクチャーでは、RabbitMQ がコントローラノードで実行するように構成します。

▼ MySQL データベースをインストールする方法

1. **MySQL サーバーとパッケージをインストールします。**

```
controller# pkg install mysql-55 mysql-55/client python-mysql \
rabbitmq markupsafe rad-evs-controller
```

2. **RabbitMQ サービスを有効にします。**

```
controller# svcadm enable rabbitmq
controller# svcadm restart rad:local
```

3. **(オプション) 管理および API トラフィック専用の IP アドレスを使用している場合は、そのアドレスを `/etc/mysql/5.5/my.cnf` に追加します。**

```
bind-address=$CONTROLLER_ADMIN_NODE_IP
```

4. **MySQL サービスを有効にします。**

```
controller# svcadm enable mysql
```

5. **MySQL サーバーの root パスワードを設定します。**

```
controller# mysqladmin -u root password MySQL-root-password
```

6. **MySQL を構成します。**

OpenStack によって使用されるテーブルを作成します。これらのデータベースへの排他的アクセスを提供するために、コントローラノード上のサービスに特権を付与します。

```
controller# mysql -u root -p
Enter password: MySQL-root-password
mysql> drop database if exists nova;
mysql> drop database if exists cinder;
mysql> drop database if exists glance;
mysql> drop database if exists keystone;
mysql> drop database if exists neutron;
mysql> drop database if exists heat;
mysql> create database cinder;
mysql> default character set utf8
mysql> default collate utf8_general_ci;
mysql> grant all privileges on cinder.* to 'cinder'@'$CONTROLLER_ADMIN_NODE' \
identified by service-password';
mysql> grant all privileges on cinder.* to 'cinder'@'$VOLUME_IP' \
identified by service-password';
mysql> create database glance;
mysql> default character set utf8
mysql> default collate utf8_general_ci;
mysql> grant all privileges on glance.* to 'glance'@'$CONTROLLER_ADMIN_NODE' \
identified by service-password';
mysql> create database keystone;
mysql> default character set utf8
mysql> default collate utf8_general_ci;
mysql> grant all privileges on keystone.* to 'keystone'@'$CONTROLLER_ADMIN_NODE' \
identified by service-password';
mysql> create database nova;
mysql> default character set utf8
mysql> default collate utf8_general_ci;
mysql> grant all privileges on nova.* to 'nova'@'$CONTROLLER_ADMIN_NODE' \
identified by service-password';
mysql> create database neutron;
mysql> default character set utf8
mysql> default collate utf8_general_ci;
mysql> grant all privileges on neutron.* to 'neutron'@'$CONTROLLER_ADMIN_NODE' \
identified by service-password';
mysql> create database heat
```

```
mysql> default character set utf8
mysql> default collate utf8_general_ci;
mysql> grant all privileges on heat.* to 'heat'@'$CONTROLLER_ADMIN_NODE' \
mysql> identified by service-password';
mysql> flush privileges;
mysql> quit
```

Keystone のインストール

Keystone サービスはコントローラノードにインストールし、構成してください。

サンプル Keystone スクリプト

Keystone データベースへの入力をすばやく行うために、サンプルスクリプト `/usr/demo/openstack/keystone/sample_data.sh` を使用できます。このスクリプトは、次に示す最初のテナントを作成します。

- `service`: ここには、各 OpenStack サービスの Keystone ユーザーが作成されます。
- `demo`: ここには、デフォルトパスワードでユーザー `admin` が作成されます。

また、このスクリプトは各 API サービスおよびそれぞれのサービスのパスワードが格納されるノードを定義する環境変数も設定します。デフォルトでは、テナント `service` の各サービスのサービス名、ユーザー名、およびパスワードは同一です。たとえば、Nova サービスでは、ユーザー `nova` がパスワード `nova` で作成されます。

スクリプトを実行する前に変数を変更して、サービスのユーザー名とそれに対応するパスワード、および最初のテナント名をカスタマイズできます。環境に設定できるパラメータの詳細については、スクリプトを確認してください。

▼ Keystone をインストールし、構成する方法

1. **Keystone パッケージをインストールします。**

   ```
   controller# pkg install keystone
   ```

2. **Keystone およびその他の OpenStack サービス用の共有トークンを作成します。**

   トークンはランダムな文字列で構成されます。

```
controller# openssl rand -hex 10
token-string
```

3. **環境変数にトークンを設定します。**

```
controller# export SERVICE_TOKEN=token-string
```

4. **/etc/keystone/keystone.conf ファイル内のパラメータを変更します。**

構成は次の例のようになります。

```
[DEFAULT]
admin_token = token-string
qpid_hostname=$CONTROLLER_ADMIN_NODE
rabbit_host=$CONTROLLER_ADMIN_NODE
...
[database]
connection = mysql://keystone:service-password@$CONTROLLER_ADMIN_NODE/keystone
```

5. **Keystone SMF サービスを有効にします。**

```
controller# svcadm enable keystone
```

6. **公開鍵インフラストラクチャー (PKI) トークンを生成します。**

```
controller# su - keystone -c "keystone-manage pki_setup"
```

7. **Keystone データベースを生成します。**

サンプルスクリプトを使用するには、次のコマンドを発行します。

```
controller# CONTROLLER_PUBLIC_ADDRESS=$CONTROLLER_ADMIN_NODE \
CONTROLLER_ADMIN_ADDRESS=$CONTROLLER_ADMIN_NODE \
CONTROLLER_INTERNAL_ADDRESS=$CONTROLLER_ADMIN_NODE \
SERVICE_TOKEN=token-string \
/usr/demo/openstack/keystone/sample_data.sh
```

Glance のインストール

Glance を設定するには、認証用にいくつかの情報を構成し、MySQL および RabbitMQ サービスの場所を指定する必要があります。

▼ Glance をインストールし、構成する方法

1. **Glance パッケージをインストールします。**

```
controller# pkg install glance
```

2. **これらの構成ファイル内でコメント解除するかパラメータを設定して Glance を構成します。**

- /etc/glance/glance-api.conf

```
[DEFAULT]
registry_host = $CONTROLLER_ADM_NODE
admin_user =glance
admin_password = service-password
admin_tenant_name = tenant
auth_url =http://$CONTROLLER_ADM_NODE:5000/v2.0
auth_strategy = keystone
default_publisher_id =image.$CONTROLLER_ADM_NODE
rabbit_host = $CONTROLLER_ADM_NODE
qpid_hostname =$CONTROLLER_ADM_NODE

[database]
connection = mysql://glance:service-password@$CONTROLLER_ADM_NODE/glance

[keystone_authtoken]
auth_uri= http://$CONTROLLER_ADM_NODE:5000/v2.0
identity_uri = http://$CONTROLLER_ADM_NODE:35357
admin_tenant_name = tenant
admin_user = glance
admin_password = service-password
```

- /etc/glance/glance-cache.conf

```
[DEFAULT]
auth_url = http://$CONTROLLER_ADM_NODE:5000/v2.0/
identity_uri = http://$CONTROLLER_ADM_NODE:35357
admin_tenant_name = tenant
admin_user = glance
admin_password = service-password
```

- /etc/glance/glance-registry.conf

```
[DEFAULT]
default_publisher_id = image.$CONTROLLER_ADM_NODE
```

```
rabbit_host = $CONTROLLER_ADM_NODE
qpid_hostname = $CONTROLLER_ADM_NODE

[database]
connection = mysql://glance:glance@$CONTROLLER_ADM_NODE/glance

[keystone_authtoken]
auth_uri = http://$CONTROLLER_ADM_NODE:5000/v2.0
identity_uri = http://$CONTROLLER_ADM_NODE:35357
admin_tenant_name = tenant
admin_user = glance
admin_password = service-password
```

- /etc/glance/glance-api-paste.ini

```
[filter:authtoken]
auth_uri = http://$CONTROLLER_ADM_NODE:5000/v2.0/
identity_uri = http://$CONTROLLER_ADM_NODE:35357
admin_tenant_name = tenant
admin_user = glance
admin_password = service-password
```

- /etc/glance/glance-registry-paste.ini

```
[filter:authtoken]
auth_uri = http://$CONTROLLER_ADM_NODE:5000/v2.0/
identity_uri = http://$CONTROLLER_ADM_NODE:35357
admin_tenant_name = tenant
admin_user = glance
admin_password = service-password
```

- /etc/glance/glance-scrubber.conf

```
[DEFAULT]
auth_url = http://$CONTROLLER_ADM_NODE:5000/v2.0/
identity_uri = http://$CONTROLLER_ADM_NODE:35357
admin_tenant_name = tenant
admin_user = glance
admin_password = service-password
```

```
[database]
connection=mysql://glance:glance@$CONTROLLER_ADM_NODE/glance
```

3. **Glance SMF サービスを有効にします。**

```
controller# svcadm enable -rs glance-api glance-db glance-registry glance-scrubber
```

Nova のインストール

コントローラノード上の Nova 構成には、通常の認証およびその他のサービス情報を構成する必要もあります。このセクションは、コンピュートノード自体ではなく Nova エンドポイントサービス構成に関連しています。

▼ Nova をインストールし、構成する方法

1. **Nova パッケージをインストールします。**

```
controller# pkg install nova
```

2. **/etc/nova/nova.conf ファイル内でコメント解除するかパラメータを設定して Nova を構成します。**

```
[DEFAULT]
qpid_hostname=$CONTROLLER_ADM_NODE
rabbit_host=$CONTROLLER_ADM_NODE
my_ip=$CONTROLLER_ADMIN_NODE_IP
host=$CONTROLLER_ADMIN_NODE
firewall_driver=nova.virt.firewall.NoopFirewallDriver

[database]
connection = mysql://nova:nova@$CONTROLLER_ADM_NODE/nova

[glance]
host=$CONTROLLER_ADM_NODE

[keystone_authtoken]
auth_uri=http:/$CONTROLLER_ADM_NODE:5000/v2.0/
identity_uri=http://$CONTROLLER_ADM_NODE:35357/
admin_user=nova
admin_password=service-password
admin_tenant_name=tenant

[neutron]
url=http://$CONTROLLER_ADM_NODE:9696
admin_username=neutron
admin_password=service-password
```

```
admin_tenant_name=tenant
admin_auth_url=http://$CONTROLLER_ADM_NODE:5000/v2.0
```

3. **Nova SMF サービスを有効にします。**

```
controller# svcadm enable -rs nova-conductor
controller# svcadm enable -rs nova-api-osapi-compute
nova-cert nova-scheduler
```

Horizon のインストール

Horizon は OpenStack の Web ポータルとして機能します。次の手順は、SSL/TLS ではなく Horizon への通常の HTTP アクセスを想定しています。

▼ Horizon を構成する方法

1. **Horizon パッケージをインストールします。**

```
controller# pkg install horizon
```

2. **使用している構成に応じて、次の一連の手順のいずれかを実行します。**

- 構成が HTTP を使用している。

  1. `/etc/openstack_dashboard/local_settings.py` スクリプトに設定を入力します。

```
controller# gsed -i -e s@SECURE_PROXY_SSL_HEADER@#SECURE_PROXY_SSL_HEADER@ \
-e s@CSRF_COOKIE_SECURE@#CSRF_COOKIE_SECURE@ \
-e s@SESSION_COOKIE_SECURE@#SESSION_COOKIE_SECURE@ \
/etc/openstack_dashboard/local_settings.py
```

  2. OpenStack 用の HTTP バージョンの `http.conf` ファイルをコピーします。

```
controller# cp /etc/apache2/2.2/samples-conf.d/openstack-dashboard-http.conf \
/etc/apache24/2.2/conf.d/
```

- 構成が SSL/TLS を使用している。

  1. Horizon で使用する証明書を生成します。

     次のコマンドは Horizon によって使用される自己署名証明書を生成し、OpenStack ダッシュボード構成ファイルを Apache 構成ファイルディレクトリにコピーします。自己署名証明書の作成の詳細については、SSL/TLS 強力な暗号化に関する FAQ を参照してください。

```
controller# export DASHBOARD=/etc/openstack_dashboard
controller# openssl req -new -x509 -nodes \
-out horizon.crt -keyout horizon.key

controller# mv horizon.crt horizon.key ${DASHBOARD}
controller# chmod 0600 ${DASHBOARD}/horizon.*

controller# sed \
-e "/SSLCertificateFile/s:/path.*:${DASHBOARD}/horizon.crt:" \
-e "/SSLCACertificateFile/d" \
-e "/SSLCertificateKeyFile/s:/path.*:${DASHBOARD}/horizon.key:" \
< /etc/apache2/2.2/samples-conf.d/openstack-dashboard-tls.conf \
> /etc/apache2/2.2/conf.d/openstack-dashboard-tls.conf
```

2. ~/conf.d/openstack-dashboard-tls.conf ファイルの次のパラメータに、Horizon パッケージのサイトアドレスとサーバー名を指定します。

```
RedirectPermanent=controller-IP
ServerName=controller-name
```

3. **適用可能な次のいずれかのコマンドを使用して、Apache サービスを開始します。**

```
controller# svcadm enable apache22
```

Cinder のインストール

Cinder 構成では少なくとも次の情報を指定する必要があります。

- Keystone で認証するための認可情報。
- 作成するボリュームのクラス。

▼ Cinder をインストールし、構成する方法

この手順のステップは、Cinder またはボリュームノードではなく、Cinder エンドポイントサービスの構成を指しています。

1. **Cinder パッケージをインストールします。**

```
controller# pkg install cinder
```

2. **/etc/cinder/cinder.conf ファイル内でコメント解除するかパラメータを設定して Cinder を構成します。**

```
[DEFAULT]
qpid_hostname=$CONTROLLER_ADM_NODE
rabbit_host=$CONTROLLER_ADM_NODE
my_ip=$CONTROLLER_ADM_NODE

[database]
connection = mysql://cinder:cinder@$CONTROLLER_ADM_NODE/cinder

[keystone_authtoken
auth_uri = http://$CONTROLLER_ADM_NODE:5000/v2.0
identity_uri = http://$CONTROLLER_ADM_NODE:35357
admin_tenant_name = tenant
admin_user = cinder
admin_password = service-password
```

3. **Cinder SMF サービスを有効にします。**

```
controller# svcadm enable -rs cinder-db
controller# svcadm enable -rs cinder-api cinder-scheduler
```

参照 ZFS に OpenStack Block Storage を構築する方法に関するドキュメントも参照してください。

▼ ZFS Storage Appliance iSCSI Cinder ドライバの構成方法

Oracle ZFS Storage Appliance iSCSI Cinder ドライバにより、Oracle ZFS Storage Appliance (ZFSSA) を Cinder のブロックストレージリソースとしてシームレスに使用できます。このドライバは、Nova サービスによってインスタンス化された任意の仮想マシンに、Cinder サーバーによって割り当て可能な iSCSI ボリュームを作成する機能を提供します。ドライバは `cloud/openstack/cinder` パッケージで配布されます。アプライアンスでは ZFSSA ソフトウェアリリース 2013.1.2.0 以上を実行している必要があります。

始める前に Oracle ZFS Storage Appliance 上にプールを構成します。既存のプールを使用するように選択できます。

1. **ワークフロー `cinder.akwf` を実行します。**

Cinder ドライバ操作を実行するためのロール認可を持つ既存のユーザーを使用するか、新しいユーザーを作成できます。

`cinder.akwf` ワークフローは次のタスクを実行します。

- ユーザーが存在しない場合はユーザーを作成します。
- Cinder ドライバの操作を実行するためのロール認可を設定します。
- RESTful サービスが現在無効になっている場合、サービスを有効にします。

コマンド行インタフェース (CLI) またはアプライアンスのブラウザユーザーインタフェース (BUI) からワークフローを実行できます。

■ **CLI からワークフローを実行します。**

```
zfssa:maintenance workflows> download
zfssa:maintenance workflows download (uncommitted)> show
Properties:
                          url = (unset)
                         user = (unset)
                     password = (unset)

zfssa:maintenance workflows download (uncommitted)> set url="url to the cinder.akwf file"
                          url = "url to the cinder.akwf file"
zfssa:maintenance workflows download (uncommitted)> commit
Transferred 2.64K of 2.64K (100%) ... done

zfssa:maintenance workflows> ls
Properties:
                   showhidden = false

Workflows:

WORKFLOW     NAME                                        OWNER SETID ORIGIN
 VERSION
workflow-000 Clear locks                                 root  false Oracle Corporation
 1.0.0
workflow-001 Configuration for OpenStack Cinder Driver root  false Oracle Corporation
 1.0.0

zfssa:maintenance workflows> select workflow-001

zfssa:maintenance workflow-001 execute (uncommitted)> set name=openstack
                         name = openstack
zfssa:maintenance workflow-001 execute (uncommitted)> set password=openstack-password
                      password = ********
zfssa:maintenance workflow-001 execute (uncommitted)> commit
User openstack created.
```

■ **BUI からワークフローを実行します。**

a. **「保守」 -> 「ワークフロー」を選択して、プラスアイコンを使用して、新しいワークフローをアップロードします。**

b. **参照ボタンをクリックして、`cinder.akwf` ファイルを選択します。**

c. **アップロードボタンをクリックして、ワークフローのアップロードを完了します。**

d. **BUI の「ワークフロー」ページに表示される新しい行をクリックし、Cinder ドライバワークフローを実行します。**

ワークフローによって、ユーザー名とパスワードの入力が求められます。このユーザー名とパスワードは、`san_login` および `san_password` として、`cinder.conf` ファイルでも使用されます。

2. **`/etc/cinder/cinder.conf` ファイルにパラメータを設定します。**

`cinder.conf` ファイルに次の必須プロパティーを指定します。

- `volume_driver` - `cinder.volume.drivers.zfssa.zfssaiscsi.ZFSSAISCSIDriver` がコメント解除されていることを確認してください。ほかの 3 つの選択がコメントアウトされていることを確認してください。
- `san_ip` – ZFSSA 管理ホストの名前または IP アドレス。
- `san_login` – ZFSSA 上の Cinder ユーザーのユーザー名。
- `san_password` – ZFSSA 上の Cinder ユーザーのパスワード。
- `zfssa_pool` – ボリュームを割り当てるために使用するプール。
- `zfssa_target_portal` – ZFSSA iSCSI ターゲットポータル `data-ip:port`)。デフォルトのポートは 3260 です。
- `zfssa_project` – ZFSSA プロジェクトの名前。プロジェクトがアプライアンスに存在していない場合は、起動時にドライバによって、その名前でプロジェクトが作成されます。このプロジェクトにはドライバによって作成されたすべてのボリュームが含まれます。ボリュームの特性 (ブロックサイズなど) やアクセス (イニシエータ、ターゲット、セキュリティーなど) を設定するために追加の ZFSSA プロパティーが提供されています。
- `zfssa_initiator_group` – イニシエータグループの名前。イニシエータグループがアプライアンスに存在していない場合は、起動時にドライバによって、その名前でイニシエータグループが作成されます。`default` イニシエータグループを使用する場合は、このパラメータの値を `default` に設定します。`default` イニシエータグループは評価目的で役立つことがあります。`default` イニシエータグループは、不要なイニシエータまたは競合するイニシエータにボリュームが公開される可能性があるため、通常は使用しないでください。

- `zfssa_target_interfaces` – ZFSSA iSCSI ターゲットネットワークインタフェース。インタフェースを表示するには次のコマンドを使用します。

```
zfssa:configuration net interfaces> show
Interfaces:

  INTERFACE STATE CLASS LINKS    ADDRS          LABEL
  e1000g0   up    ip    e1000g0  1.10.20.30/24  Untitled Interface
```

- `connection` – パラメータを次のように設定します。

```
connection=mysql://cinder:service-password@controller-fqdn/cinder
```

3. **ZFSSA iSCSI サービスがオンラインであることを確認します。**

   ZFSSA iSCSI サービスがオンラインでない場合は、アプライアンスの BUI または CLI を使用して、それを有効にします。次の例では、アプライアンスの CLI の使用を示します。

```
zfssa:> configuration services iscsi
zfssa:configuration services iscsi> enable
zfssa:configuration services iscsi> show
Properties:
<status> = online
...
```

4. **Cinder ボリューム SMF サービスを有効にします。**

```
controller# svcadm enable cinder-volume:default cinder-volume:setup
```

Neutron のインストールと構成

この章で説明するアーキテクチャーでは、Neutron API サービスはコントローラノードで実行します。

▼ Neutron をインストールし、構成する方法

1. **Neutron パッケージをインストールします。**

```
controller# pkg install neutron
```

2. **これらの構成ファイル内でコメント解除するかパラメータを設定して Neutron を構成します。**

- `/etc/neutron/neutron.conf`

```
qpid_hostname=/$CONTROLLER_ADM_NODE
rabbit_host=/$CONTROLLER_ADM_NODE

# Host to locate redis. (string value)
# host=127.0.0.1
host=$CONTROLLER_ADM_NODE

[keystone_authtoken]
auth_uri = http://$CONTROLLER_ADM_NODE:5000/v2.0
identity_uri = http://$CONTROLLER_ADM_NODE:35357
admin_tenant_name = tenant
admin_user = neutron
admin_password = service-password

[database]
connection = mysql://neutron:neutron@$CONTROLLER_ADM_NODE/neutron
```

- /etc/neutron/plugins/evs/evs_plugin.ini

```
[EVS]
evs_controller = ssh://evsuser@$CONTROLLER_ADM_NODE
```

- /etc/neutron/dhcp_agent.ini

```
[DEFAULT]
evs_controller = ssh://evsuser@$CONTROLLER_ADM_NODE
```

3. Elastic Virtual Switch (EVS) を構成します。

   a. EVS プロパティーを設定して EVS コントローラの場所を指定します。

   ```
   controller# evsadm set-prop -p controller=ssh://evsuser@$CONTROLLER_ADM_NODE
   ```

   b. evsuser、neutron、および root ユーザーの SSH 鍵ペアを作成します。

   ```
   controller# su - evsuser -c "ssh-keygen -N '' \
   -f /var/user/evsuser/.ssh/id_rsa -t rsa"
   controller# su - neutron -c "ssh-keygen -N '' -f /var/lib/neutron/.ssh/id_rsa -t rsa"
   controller# ssh-keygen -N '' -f /root/.ssh/id_rsa -t rsa
   ```

c. **evsuser の authorized_keys ファイル内で、evsuser、neutron、および root ユーザーの SSH 鍵を組み合わせます。**

```
controller# cat /var/user/evsuser/.ssh/id_rsa.pub \
/var/lib/neutron/.ssh/id_rsa.pub /root/.ssh/id_rsa.pub >> \
/var/user/evsuser/.ssh/authorized_keys
```

d. **known_host ファイル内に格納されるフィンガープリントを受け入れるように SSH 接続をテストします。**

すべての確認用プロンプトで Yes を指定します。

```
controller# su - evsuser -c "ssh evsuser@$CONTROLLER_ADM_NODE true"
controller# su - neutron -c "ssh evsuser@$CONTROLLER_ADM_NODE true"
controller# ssh evsuser@$CONTROLLER_ADM_NODE true
```

e. **.ssh ディレクトリの所有権を設定します。**

```
controller# chown -R evsuser:evsgroup /var/user/evsuser/.sshcontroller
controller# chown -R neutron:neutron /var/lib/neutron/.ssh
```

f. **EVS コントローラの l2-type、uplink-port、および vlan-range プロパティーを構成します。**

```
controller# evsadm set-controlprop -p property=value
```

次の例に、これらのプロパティーの設定方法を示します。オプションとして、最後のコマンドを使用するとすべての EVS プロパティーを表示できます。

```
controller# evsadm set-controlprop -p l2-type=vlan
controller# evsadm set-controlprop -p vlan-range=1,200-300
controller# evsadm set-controlprop -p uplink-port=net0

controller# evsadm show-controlprop -o all
```

4. **IP 転送を有効にします。**

```
controller# ipadm set-prop -p forwarding=on ipv4
```

5. **IP フィルタサービスを開始します。**

```
controller# svcadm enable -rs ipfilter
```

6. **Neutron サーバーサービスを有効にします。**

```
controller# svcadm enable -rs neutron-server neutron-dhcp-agent
```

Heat のインストールと構成

Heat は、作成されたテンプレートに基づいてクラウドアプリケーションを配備できるようにする OpenStack のオーケストレーションエンジンです。Heat は Keystone と同じノードにインストールします。

▼ Heat を構成する方法

始める前に このタスクを実行する前に、まず「Keystone をインストールし、構成する方法」の説明に従って Keystone を構成する必要があります。

1. **Heat パッケージをインストールします。**

```
controller# pkg install heat
```

2. **Heat 設定スクリプトを実行します。**

```
controller# OS_SERVICE_ENDPOINT=http://$CONTROLLER_ADM_NODE \
SERVICE_HOST=$CONTROLLER_ADM_NODE \
OS_AUTH_URL=http://$CONTROLLER_ADM_NODE:5000/v2.0 \
OS_USERNAME=admin OS_PASSWORD=secrete OS_TENANT_NAME=demo \
/usr/demo/openstack/keystone/heat-keystone-setup
```

3. **これらの構成ファイル内でコメント解除するかパラメータを設定して Heat を構成します。**

- `/etc/heat/heat.conf`

```
[database]
connection = mysql://heat:heat@$CONTROLLER_ADM_NODE/heat

[keystone_authtoken]
auth_uri = http://$CONTROLLER_ADM_NODE:5000/v2.0
identity_uri = http://$CONTROLLER_ADM_NODE:35357
admin_tenant_name = tenant
admin_user = heat
admin_password = service-password
```

- `/etc/heat/api-paste.ini`

```
[filter:authtoken]
auth_uri = http://$CONTROLLER_ADM_NODE:5000/v2.0/
```

```
identity_uri = http://$CONTROLLER_ADM_NODE:35357
admin_tenant_name = tenant
admin_user = heat
admin_password = service-password
```

4. **Heat サービスを有効にします。**

```
controller# svcadm enable -rs heat-api heat-db heat-engine \
heat-api-cfn heat-api-cloudwatch
```

コンピュートノードの構成

コンピュートノードに VM インスタンスと nova-compute デーモンをインストールします。VM インスタンスは、Web アプリケーションや分析などの幅広いサービスを提供します。コンピュートノードはクラウドに必要な数だけ構成できます。

Oracle Solaris 11 で ZFS とアプリケーションの間のメモリー使用をより効率的に管理するには、次の例に示すように、ノード上で usr_reserve_hint_pct パラメータを設定します。

```
# echo "set user_reserve_hint_pct=80" >>/etc/system.d/site:kernel-zones-reserve
# reboot
```

ここで、*site* にはユーザーの会社を指定することもあります。

ほかの OpenStack ノードでも同様にこのパラメータを設定するようにしてください。

このパラメータの詳細については、https://support.oracle.com で MOS にログインし、*Oracle Solaris 11.2 での ZFS とアプリケーションの間のメモリー管理*に関するドキュメント 1663862.1 を確認してください。

▼ コンピュートノードを構成する方法

1. **(オプション) NTP をインストールし、構成します。**

47 ページの「Network Time Protocol のインストール」を参照してください。

2. **Nova パッケージをインストールします。**

```
compute1# pkg install nova
```

3. **Remote Access Daemon (RAD) を再起動します。**

   Nova は RAD を使用して、Oracle Solaris ゾーンフレームワークと通信します。

```
compute1# svcadm restart rad:local
```

4. **/etc/nova/nova.conf ファイル内でコメント解除するか次のパラメータを設定して Nova を構成します。**

```
[DEFAULT]
rabbit_host=$CONTROLLER_ADM_NODE
my_ip=$COMPUTE_ADMIN_NODE_IP
host=$COMPUTE_ADMIN_NODE_X
firewall_driver=nova.virt.firewall.NoopFirewallDriver
keystone_ec2_url=http:/$CONTROLLER_ADM_NODE:5000/v2.0/ec2tokens

[database]
connection = mysql://nova:nova@$CONTROLLER_ADM_NODE/nova

[glance]
host=$CONTROLLER_ADM_NODE

[keystone_authtoken]
auth_uri=http://$CONTROLLER_ADM_NODE:5000/v2.0/
identity_uri=http://$CONTROLLER_ADM_NODE:35357/
admin_usr=nova
admin_password=service-password
admin_tenant_name=tenant

[neutron]
url=http://$CONTROLLER_ADM_NODE:9696
admin_username=neutron
admin_password=service-password
admin_tenant_name=tenant
admin_auth_url=http://$CONTROLLER_ADM_NODE:5000/v2.0
```

5. **コンピュートノードで EVS を設定します。**

   a. **EVS パッケージをインストールします。**

```
compute1# pkg install evs
```

   b. **EVS プロパティーを設定して EVS コントローラの場所を指定します。**

```
compute1# evsadm set-prop -p controller=ssh://evsuser@$CONTROLLER_ADM_NODE
```

6. **コントローラとコンピュートノードの間の通信を構成します。**

a. **コンピュートノード上に root ユーザーの SSH 公開鍵を作成します。**

```
compute1# su - root -c "ssh-keygen -N '' -f /root/.ssh/id_rsa -t rsa"
```

b. **(オプション) SSH 鍵の内容を確認します。**

```
compute1# cat /root/.ssh/id_rsa.pub
```

c. **SSH 鍵 /root/.ssh/id_rsa.pub をコントローラノード内の場所にコピーします。**

d. **コントローラノード上で、evsuser の authorized_keys ファイルに SSH 鍵を追加します。**

```
controller# cat location/id_rsa.pub >> /var/user/evsuser/.ssh/authorized_keys
```

e. **(オプション) コンピュートノードの SSH 鍵が authorized_keys ファイルに追加されていることを確認します。**

```
controller# cat /var/user/evsuser/.ssh/authorized_keys
```

出力には、コンピュートノード上で生成され、ステップ 6.b で表示された SSH 鍵の内容が含まれます。

f. **コンピュートノード上で、コントローラへのコンピュートノードの SSH 接続をテストし、known_host ファイルへのフィンガープリントの格納を受け入れます。**

確認用プロンプトで Yes を指定します。

```
compute1# ssh evsuser@$CONTROLLER_ADM_NODE true
```

7. **Nova コンピュートサービスを有効にします。**

```
compute1# svcadm enable nova-compute
```

ストレージノードの構成

ストレージノードは、OpenStack の設定でトランザクション処理されるすべてのデータのリポジトリです。

Oracle Solaris 11 で ZFS とアプリケーションの間のメモリー使用をより効率的に管理するには、次の例に示すように、ノード上で usr_reserve_hint_pct パラメータを設定します。

```
# echo "set user_reserve_hint_pct=80" >>/etc/system.d/site:kernel-zones-reserve
# reboot
```

ここで、*site* にはユーザーの会社を指定することもあります。

ほかの OpenStack ノードでも同様にこのパラメータを設定するようにしてください。

このパラメータの詳細については、https://support.oracle.com で MOS にログインし、*Oracle Solaris 11.2 での ZFS とアプリケーションの間のメモリー管理*に関するドキュメント 1663862.1 を確認してください。

▼ ブロックストレージノードを構成する方法

1. **適切なパッケージをインストールします。**

```
storage# pkg install cinder python-mysql mysql-55/client
```

2. **/etc/cinder/cinder.conf ファイル内でコメント解除するかパラメータを設定して Cinder を構成します。**

```
[DEFAULT]
san_is_local=true
my_ip=storage-IP
rabbit_host=controller-fqdn
glance_host=controller-IP
zfs_volume_base=cinder/cinder

[database]
connection = mysql://cinder:service-password@controller-fqdn/cinder

[DEFAULT]
san_is_local=true
my_ip=$VOLUME_IP
rabbit_host=$CONTROLLER_ADM_NODE
glance_host=$CONTROLLER_ADM_NODE
zfs_volume_base=cinder/cinder

[database]
connection = mysql://cinder:cinder@$CONTROLLER_ADM_NODE/cinder

[keystone_authtoken]
auth_uri = http://$CONTROLLER_ADM_NODE:5000/v2.0
identity_uri = http://$CONTROLLER_ADM_NODE:35357
admin_user = cinder
admin_password = service-password
admin_tenant_name = tenant
```

3. **Cinder サービスを開始します**

```
storage# svcadm enable -rs cinder-db cinder-volume:default cinder-volume:setup
storage# svcadm enable -rs iscsi/target
```

OpenStack 内の内部ネットワークの構成

内部ネットワークでは、あとから作成する VM インスタンスは相互通信のみ可能です。これらのインスタンスは、より広いネットワークには接続できないか、または外部からアクセスできます。

注記 - テナントに対して外部ネットワークを構成するには、107 ページの「外部ネットワークを使用した OpenStack の構成」を参照してください。

▼ 内部ネットワークを作成する方法

Neutron サービスが実行されているノード (この場合はコントローラノード) でこれらの手順を実行します。

1. **既存のテナントをリストします。**

   ```
   controller# keystone tenant-list
   ```

   テナントのリストにはテナントおよびそれらに対応する ID が含まれます。内部ネットワーク作成の対象となるテナントの ID を選択します。

2. **選択したテナントのネットワークを作成します。**

   ```
   controller# neutron net-create --tenant-id tenant-ID network-name
   ```

   ここで、*tenant-ID* は前の手順のテナントリストから取得されるものです。

3. **同じテナントのサブネットを作成します。**

   ```
   controller# neutron subnet-create --name subnet-name \
   --tenant-id tenant-ID network-name subnet-IP
   ```

例 4-1 内部ネットワークの作成

この例では、88 ページの「サンプル Keystone スクリプト」でサンプル Keystone スクリプトによってデフォルトで作成されたテナント `demo` の内部ネットワークを作成する方法を示します。

```
controller# keystone tenant-list
```

```
+------------+---------+---------+
|     id     |   name  | enabled |
+------------+---------+---------+
| abcde12345 |   demo  |  True   |
| fghij67890 | service |  True   |
+------------+---------+---------+

controller# neutron net-create --tenant-id abcde12345 demo_internal_net
Created a new network:
+--------------------------+------------------+
| Field                    | Value            |
+--------------------------+------------------+
| admin_state_up           | True             |
| id                       | 9999             |
| name                     | demo_internal_net |
| provider:network_type    | vlan             |
| provider:segmentation_id | 300              |
| router:external          | False            |
| shared                   | False            |
| status                   | ACTIVE           |
| subnets                  |                  |
| tenant_id                | abcde12345       |
+--------------------------+------------------+

controller# neutron subnet-create --name demo_int_subnet --tenant-id abcde12345 \
demo_internal_net 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               | 07f9b37c-ae4e-11e4-8000-db57d0041a2c             |
| ip_version       | 4                                                |
| name             | demo_int_subnet                                  |
| network_id       | 99999|                                           |
| tenant_id        | abcde12345                                       |
+------------------+--------------------------------------------------+
```

外部ネットワークを使用した OpenStack の構成

外部ネットワークを作成すると、クラウド内のプライベートネットワークがさらに広いネットワークと通信できるようになります。クラウドでは、テナントのプライベートネットワークは 1 つ以上になる場合があります。クラウドの外部ネットワークを作成すると、すべてのテナントネットワークで共有されるプロバイダルーターが作成されます。このルーターについては、管理者が作成、所有、およ

び管理します。ルーターはテナントのネットワークトポロジビューに表示されません。単一のルーターしかないため、テナントネットワークでは重複する IP アドレスを使用できません。

外部ネットワークの作成には、Neutron L3 エージェントの構成も含まれます。Neutron L3 エージェントは、Nova インスタンスに割り当てられたアドレスとフローティング IP アドレス間の 1 対 1 の NAT マッピングを自動的に作成します。L3 エージェントによって、プライベートネットワーク間の通信も可能になります。デフォルトでは、同じテナントの一部であるプライベートネットワーク間のルーティングは無効になっています。この動作を変更するには、`/etc/neutron/l3_agent.ini` 構成ファイルで `allow_forwarding_between_networks` を `True` に設定し、`neutron-l3-agent` SMF サービスを再起動します。

ルーターはテナントの VM インスタンスの外部への接続を提供します。ルーターは、外部ネットワークにルーターを接続するインタフェース上で双方向の NAT を実行します。テナントは、それらが必要なだけ、またはフローティング IP の割り当て制限によって許可されるだけの数のフローティング IP (パブリック IP) を作成し、これらのフローティング IP を、外部接続を必要とする VM インスタンスに関連付けます。

OpenStack 内での内部ネットワークと外部ネットワークの間の関係図は、図3-3「プライベートネットワークモデルでのプロバイダルーター」を参照してください。

▼ OpenStack 内で外部ネットワークを構成する方法

この手順では、外部ネットワークを表す仮想ネットワークを作成する方法を示します。この仮想ネットワークでは DHCP が使用されません。代わりにフローティング IP アドレスが作成されます。これらのフローティング IP アドレスは特定のテナントに割り当てられ、そのテナントで、ユーザーによって使用される Nova VM インスタンスに割り当てることができます。

この章のマルチノードアーキテクチャーのサンプルではコントローラに Neutron サービスが含まれているため、コントローラノードで次の手順を実行します。

始める前に この手順を実行するには、テナントの内部ネットワークがすでに存在している必要があります。プライベートネットワークを作成する手順については、106 ページの「内部ネットワークを作成する方法」を参照してください。

また、Elastic Virtual Switch についても、その `l2-type` および `vlan-range` プロパティーが構成されているなど、構成が完了している必要があります。次の例は、ステップ 9 でこれらのプロパティーを構成したときに、その設定を表示する方法を示しています。

```
controller# evsadm show-controlprop -p l2-type -p vlan-range
```

```
PROPERTY   PERM  VALUE      DEFAULT  HOST
l2-type    rw    vlan       vlan     --
vlan-range rw    1,200-300  --       --
```

また、外部ネットワークに接続するプライベートネットワークのテナントに関する情報も必要です。テナント情報は、次のコマンドを使用するといつでも表示できます。

```
keystone tenant-list
```

1. **Solaris IP フィルタを有効にします。**

```
controller# svcadm enable ipfilter
```

2. **ホスト全体で IP 転送を有効にします。**

```
controller# ipadm set-prop -p forwarding=on ipv4
```

3. **必要な環境変数を設定します。**

```
controller# export OS_USERNAME=neutron
controller# export OS_PASSWORD=service-password
controller# export OS_TENANT_NAME=service-name
controller# export OS_AUTH_URL=http://controller-name:5000/v2.0
```

4. **プロバイダルーターを作成します。**

```
controller# neutron router-create router-name
```

このコマンドは、ルーター名とそれに対応する ID を表示します。次の手順で、この ID を使用して構成ファイルを更新します。

5. **L3 エージェント構成ファイルを更新します。**

`/etc/neutron/l3_agent.ini` ファイルで、`router_id` パラメータの値を前の手順のルーター UUID に設定します。

```
router_id = router-ID
```

6. **`neutron-l3-agent` SMF サービスを有効にします。**

```
controller# svcadm enable neutron-l3-agent
```

7. **外部ネットワークを作成します。**

```
controller# neutron net-create --provider:network_type=vlan \
--provider:segmentation_id=VLAN-nbr \
--router:external=true network-name
```

ここで、`segmentation_id` は VLAN 範囲の最初の数字です。

8. **サブネットを作成して外部ネットワークに関連付けます。**

   このドキュメントのサンプル構成では、DHCP は無効になっています。割り当てプールはサブネットに割り当てられているフローティング IP アドレスの範囲で構成されます。

   ```
   controller# neutron subnet-create --enable-dhcp=false --name subnet-name \
   --allocation-pool start=start-IP, end=end-IP network-name subnet-IP
   ```

9. **ルーターに外部ネットワークを追加します。**

   ```
   controller# neutron router-gateway-set router-ID network-ID \
   ```

   注記 - `/etc/neutron/l3_agent.ini` ファイルから *router-ID* を取得します。必要に応じて `neutron net-list` コマンドを使用すると、*network-ID* を取得できます。

10. **ルーターにテナントのプライベートネットワークを追加します。**

    この手順では、ルーター ID とテナントのサブネット ID が必要です。次のようにして情報を取得できます。

    a. **テナントのサブネット ID を取得するには、まずテナントとそれらの ID を表示し、次に特定のテナント ID のサブネットを表示します。**

       ```
       # keystone tenant-list
       # neutron net-list --tenant-id tenant-ID
       ```

    b. **ルーターにプライベートネットワークを追加します。**

       この手順は、外部ネットワークに追加するテナントのプライベートネットワークの数に応じて繰り返します。

       ```
       controller# neutron router-interface-add router-ID subnet-ID
       ```

例 4-2 `service` テナントに対する外部ネットワークの作成

次の例は参照として 図3-3「プライベートネットワークモデルでのプロバイダルーター」 の一部を使用します。図では、テナント A には 2 つの VM インスタンスがあり、それぞれがプライベートネットワークに属しています。この 2 つのプライベートネットワークは `HR` と `ENG` です。2 つの VM インスタンスがさらに広いネットワークと通信できるように、外部ネットワークにこれら 2 つのサブネットが追加されます。この例では、Neutron サービスのデフォルトのユーザー名とパスワードを使用することを想定しています。

```
controller# svcadm enable ipfilter
controller# ipadm set-prop -p forwarding=on ipv4
```

```
controller# export OS_USERNAME=neutron
controller# export OS_PASSWORD=neutron
controller# export OS_TENANT_NAME=TenantA
controller# export OS_AUTH_URL=http://controller-name:5000/v2.0

controller# neutron router-create ext-router
Created a new router:
+-----------------------+-----------------+
| Field                 | Value           |
+-----------------------+-----------------+
| admin_state_up        | True            |
| external_gateway_info |                 |
| id                    | 97ro5-ut3er     |
| name                  | ext-router      |
| status                | ACTIVE          |
| tenant_id             | abcde12345      |
+-----------------------+-----------------+
```

この時点で、router_ID パラメータを 97ro5-ut3er に設定して /etc/neutron/l3_agent.ini ファイルを更新します。

ファイルを更新したあと、残りの手順に進みます。

```
controller# svcadm enable neutron-l3-agent

controller# neutron net-create --provider:network_type=vlan \
--provider:segmentation_id=1 --router:external=true ext_network
Created a new network:
+--------------------------+------------------+
| Field                    | Value            |
+--------------------------+------------------+
| admin_state_up           | True             |
| id                       | 555ext-net555    |
| name                     | ext_network      |
| provider:network_type    | vlan             |
| provider:segmentation_id | 1                |
| router:external          | True             |
| shared                   | False            |
| status                   | ACTIVE           |
| subnets                  |                  |
| tenant_id                | abcde12345       |
+--------------------------+------------------+

controller# neutron subnet-create  --enable-dhcp=False \
--name ext_subnet --allocation-pool start=10.134.13.8,end=10.134.13.254 \
ext_network 10.134.13.0/24
Created a new subnet:

+------------------+--------------------------------------------------+
| Field            | Value                                            |
+------------------+--------------------------------------------------+
| allocation_pools | {"start": "10.134.13.8", "end": "10.134.13.254"} |
| cidr             | 10.134.13.0/24                                   |
```

```
| dns_nameservers  |                                                  |
| enable_dhcp      | False                                            |
| gateway_ip       | 10.134.13.1                                      |
| host_routes      |                                                  |
| id               | 444sub-net444                                    |
| ip_version       | 4                                                |
| name             | ext_subnet                                       |
| network_id       | 555ext-net555                                    |
| tenant_id        | abcde12345                                       |
+------------------+--------------------------------------------------+

controller# neutron router-gateway-set 97ro5-ut3er 555ext-net555
Set gateway for router 97ro5-ut3er

controller# keystone tenant-list
+------------+---------+---------+
|     id     |   name  | enabled |
+------------+---------+---------+
| 12345abcde | TenantA |  True   |
| 67890fghij | TenantB |  True   |
+------------+---------+---------+

controller# neutron net-list --tenant-id 12345abcde
+------------+------+-------------+
|     id     | name |   subnets   |
+------------+------+-------------+
| 1a3b5c7d9e | HR   | xyz-123-uvw |
| 2f4g6h8i0j | ENG  | 098-r2d2-56 |
+------------+------+-------------+

controller# neutron router-interface-add 97ro5-ut3er xyz-123-uvw     HR added to the router.
Added interface xyz-123-uvw to router 97ro5-ut3er.

controller# neutron router-interface-add 97ro5-ut3er 098-r2d2-56     ENG added to the router.
Added interface 098-r2d2-56 to router 97ro5-ut3er.
```

参照

- 77 ページの「L3 エージェント構成を監視する方法」
- 125 ページの「既知の制限事項」

▼ フローティング IP アドレスをテナントユーザーとして作成および関連付ける方法

この手順は OpenStack Horizon ダッシュボードを使用してテナントユーザーによって実行します。

1. **OpenStack ダッシュボードにログインします。**

テナントユーザーの資格証明を使用して、29 ページの「OpenStack ダッシュボードにアクセスする方法」の説明に従って、ログインします。

2. **「プロジェクト」 -> 「アクセスとセキュリティ」 -> 「Floating IP」を選択します。**

3. **外部ネットワーク名を選択します。**

4. **「IP の確保」ボタンをクリックします。**

   「Floating IP」タブに、割り当てられたフローティング IP アドレスが表示されます。

5. **「割り当て」ボタンをクリックします。**

6. **プルダウンメニューから VM インスタンスのポートを選択します。**

   「プロジェクト」 -> 「インスタンス」ウィンドウにはフローティング IP アドレスが VM インスタンスに関連付けられていることが示されます。

   VM インスタンスの起動中に、鍵ペア (SSH 公開鍵) を選択した場合、その SSH 鍵は VM インスタンスの `root` ユーザーの `authorized_keys` ファイルに追加されます。

7. **実行中の VM インスタンスにログインします。**

   ```
   # ssh root@IP-address
   ```

   ここで、*IP-address* は VM インスタンスに関連付けられているフローティング IP アドレスです。

   ログイン後、アプリケーションをインストールする場合など、ほかのシステムで通常行うようにインスタンスを使用できます。

```
global# ssh root@10.134.13.9
Last login: Fri Jul 18 00:37:39 2014 from 10.132.146.13
Oracle Corporation      SunOS 5.11      11.2     June 2014
root@host-192-168-101-3:~# uname -a
SunOS host-192-168-101-3 5.11 11.2 i86pc i386 i86pc
root@host-192-168-101-3:~# zoneadm list -cv
ID NAME             STATUS      PATH                BRAND      IP
 2 instance-00000001 running     /                   solaris    excl
root@host-192-168-101-3:~# ipadm
NAME             CLASS/TYPE STATE     UNDER      ADDR
lo0              loopback   ok        --         --
   lo0/v4        static     ok        --         127.0.0.1/8
   lo0/v6        static     ok        --         ::1/128
net0             ip         ok        --         --
   net0/dhcp     inherited  ok        --         192.168.101.3/24
```

▼ L3 エージェント構成を監視する方法

ipf、ippool、および ipnat などの IP フィルターコマンドおよび dladm や ipadm などのネットワークコマンドを使用すると、neturon-l3-agent によって実行された構成を監視し、トラブルシューティングすることができます。

1. **neutron-l3-agent によって作成された VNIC を表示します。**

```
network# dladm show-vnic
LINK                OVER         SPEED  MACADDRESS        MACADDRTYPE VIDS
l3i7843841e_0_0     net1         1000   2:8:20:42:ed:22   fixed       200
l3i89289b8e_0_0     net1         1000   2:8:20:7d:87:12   fixed       201
l3ed527f842_0_0     net0         100    2:8:20:9:98:3e    fixed
```

2. **neutron-l3-agent によって作成された IP アドレスを表示します。**

```
network# ipadm
NAME                  CLASS/TYPE STATE   UNDER      ADDR
l3ed527f842_0_0       ip         ok      --         --
  l3ed527f842_0_0/v4  static     ok      --         10.134.13.8/24
  l3ed527f842_0_0/v4a static     ok      --         10.134.13.9/32
l3i7843841e_0_0       ip         ok      --         --
  l3i7843841e_0_0/v4  static     ok      --         192.168.100.1/24
l3i89289b8e_0_0       ip         ok      --         --
  l3i89289b8e_0_0/v4  static     ok      --         192.168.101.1/24
```

3. **IP フィルタルールを表示します。**

```
network# ipfstat -io
empty list for ipfilter(out)
block in quick on l3i7843841e_0_0 from 192.168.100.0/24 to pool/4386082
block in quick on l3i89289b8e_0_0 from 192.168.101.0/24 to pool/8226578
network# ippool -l
table role = ipf type = tree number = 8226578
{ 192.168.100.0/24; };
table role = ipf type = tree number = 4386082
{ 192.168.101.0/24; };
```

4. **IP NAT ルールを表示します。**

```
network# ipnat -l
List of active MAP/Redirect filters:
bimap l3ed527f842_0_0 192.168.101.3/32 -> 10.134.13.9/32
List of active sessions:
BIMAP 192.168.101.3  22  <- -> 10.134.13.9  22 [10.132.146.13 36405]
```

♦ ♦ ♦ 第 5 章

5 仮想マシンインスタンスの作成

この章では、OpenStack クラウド内の仮想マシンインスタンス (VM インスタンス) をプロビジョニングする方法について説明します。各 VM インスタンスはテナントに属しています。ユーザーは 1 つ以上のテナント内に 1 つ以上の VM インスタンスを作成し、そこで作業できます。OpenStack VM インスタンスの作成および管理に関する一般的な情報については、OpenStack エンドユーザーガイドを参照してください。

VM インスタンスを作成するには、フレーバとイメージが必要です。この章では、次の内容について説明します。

ファイルを作成し、そこに OpenStack コマンドで使用する環境変数を設定することもできます。環境変数を設定しない場合は、各コマンドでオプションを指定する必要があります。詳細については、OpenStack コマンド行インタフェースリファレンスを参照してください。

フレーバの管理

*フレーバ*は VM インスタンスのタイプ、つまり仮想ハードウェアテンプレートです。フレーバは、VM インスタンスに割り当てられた仮想 CPU の数、メモリー量、ディスク容量など、一連の仮想マシンリソースを指定します。Solaris では、基盤となるゾーンのブランドもフレーバに含まれ、`solaris` は非大域ゾーン、`solaris-kz` はカーネルゾーンを表します。インスタンスフレーバの一例は、16 個の仮想 CPU と 16384M バイトの RAM を備えたカーネルゾーンです。

フレーバの一般的な情報については、*OpenStack クラウド管理者ガイド*のフレーバに関するセクションを参照してください。

フレーバに関する情報の表示

nova flavor-list コマンドは、使用可能なフレーバのリストを表示します。フレーバの名前または ID は、VM インスタンスを作成するときに使用します。フレーバリストに extra-specs を含めるには、nova flavor-list コマンドに --extra-specs オプションを指定します。extra_specs 値の詳細については、117 ページの「フレーバ仕様の変更」を参照してください。

次の例では、RXTX_Factor および Is_Public 列は領域を節約するために出力から取り除かれています。これらの列については、OpenStack コマンド行インタフェースリファレンスを参照してください。

```
$ nova flavor-list
+----+-----------------------------+-----------+------+-----------+------+-------+
| ID | Name                        | Memory_MB | Disk | Ephemeral | Swap | VCPUs |
+----+-----------------------------+-----------+------+-----------+------+-------+
| 1  | Oracle Solaris              |           |      |           |      |       |
|    |  kernel zone - tiny         | 2048      | 10   | 0         |      | 1     |
| 10 | Oracle Solaris              |           |      |           |      |       |
|    |  non-global zone - xlarge   | 16384     | 80   | 0         |      | 32    |
| 2  | Oracle Solaris              |           |      |           |      |       |
|    |  kernel zone - small        | 4096      | 20   | 0         |      | 4     |
| 3  | Oracle Solaris              |           |      |           |      |       |
|    |  kernel zone - medium       | 8192      | 40   | 0         |      | 8     |
| 4  | Oracle Solaris              |           |      |           |      |       |
|    |  kernel zone - large        | 16384     | 40   | 0         |      | 16    |
| 5  | Oracle Solaris              |           |      |           |      |       |
|    |  kernel zone - xlarge       | 32768     | 80   | 0         |      | 32    |
| 6  | Oracle Solaris              |           |      |           |      |       |
|    |  non-global zone - tiny     | 2048      | 10   | 0         |      | 1     |
| 7  | Oracle Solaris              |           |      |           |      |       |
|    |  non-global zone - small    | 3072      | 20   | 0         |      | 4     |
| 8  | Oracle Solaris              |           |      |           |      |       |
|    |  non-global zone - medium   | 4096      | 40   | 0         |      | 8     |
| 9  | Oracle Solaris              |           |      |           |      |       |
|    |  non-global zone - large    | 8192      | 40   | 0         |      | 16    |
+----+-----------------------------+-----------+------+-----------+------+-------+
```

次のコマンドは、指定されたフレーバに関する詳細情報を表示します。コマンドの 1 番目のバージョンではフレーバ名を指定し、2 番目のバージョンではフレーバ ID を指定しています。これらのコマンドのどちらでも出力は同じになります。

```
$ nova flavor-show 'Oracle Solaris kernel zone - large'
$ nova flavor-show
+---------------------------+-----------------------------------+
| Property                  | Value                             |
+---------------------------+-----------------------------------+
| name                      | Oracle Solaris kernel zone - large |
| ram                       | 16384                             |
| OS-FLV-DISABLED:disabled  | False                             |
```

```
| vcpus                      | 16                                 |
| extra_specs                | {u'zonecfg:brand': u'solaris-kz'}  |
| swap                       |                                    |
| os-flavor-access:is_public | True                               |
| rxtx_factor                | 1.0                                |
| OS-FLV-EXT-DATA:ephemeral  | 0                                  |
| disk                       | 40                                 |
| id                         | 4                                  |
+----------------------------+------------------------------------+
```

フレーバ仕様の変更

フレーバ仕様を変更するには、`flavor-key` サブコマンドを使用して `extra_specs` 値を変更します。

`nova flavor-key` *flavor action key=value key=value* `...]`

flavor フレーバの名前または ID。

action `set` または `unset`

key=value *key* は仕様の名前です。*value* はその仕様の新しい値です。*action* が `unset` の場合は、*key* だけを指定します。

```
$ nova flavor-key 4 set zonecfg:bootargs=-v
$ nova flavor-show
+----------------------------+-----------------------------------------------+
| Property                   | Value                                         |
+----------------------------+-----------------------------------------------+
| name                       | Oracle Solaris                                |
|                            |  kernel zone - large                          |
| ram                        | 16384                                         |
| OS-FLV-DISABLED:disabled   | False                                         |
| vcpus                      | 16                                            |
| extra_specs                | {u'zonecfg:brand':                            |
|                            |    u'solaris-kz', u'zonecfg:bootargs': u'-v'}  |
| swap                       |                                               |
| os-flavor-access:is_public | True                                          |
| rxtx_factor                | 1.0                                           |
| OS-FLV-EXT-DATA:ephemeral  | 0                                             |
| disk                       | 40                                            |
| id                         | 4                                             |
+----------------------------+-----------------------------------------------+
```

次のプロパティーはカーネルゾーンと非大域ゾーンの両方でサポートされます。

- `zonecfg:bootargs`
- `zonecfg:brand`

- zonecfg:hostid

次のプロパティーは非大域ゾーンでのみサポートされます。

- zonecfg:file-mac-profile
- zonecfg:fs-allowed
- zonecfg:limitpriv

ほかのゾーン構成プロパティーは OpenStack ではサポートされていません。これらのゾーン構成プロパティーについては、zonecfg(1M) のマニュアルページを参照してください。

システム構成プロファイルを指定するには、sc_profile キーを使用します。

```
$ nova flavor-key 4 set sc_profile=/system/volatile/profile/sc_profile.xml
```

nova flavor-key コマンドで変更または追加できるのは、extra_specs 値だけです。既存のフレーバのほかの仕様 (RAM の量など) を変更するには、そのフレーバを削除し、変更したフレーバを同じ名前で作成する必要があります。フレーバの削除と作成については、OpenStack 管理ユーザーガイドを参照してください。

イメージの管理

仮想マシンイメージ *イメージ*) は、ブート可能なオペレーティングシステムがインストールされている仮想ディスクを含む単一のファイルです。イメージは仮想マシンのファイルシステムのテンプレートを提供します。

イメージの管理には、ダッシュボードのほかに、glance および nova コマンド行クライアント、または Image Service API と Compute API を使用できます。

イメージに関する情報の表示

nova image-list コマンドは、使用可能なイメージのリストを表示します。イメージの名前または ID は、VM インスタンスを作成するときに使用します。

```
$ nova image-list
+--------------------------------------+-------------------------+--------+--------+
| ID                                   | Name                    | Status | Server |
```

```
+--------------------------------------+------------------------+--------+--------+
| e422aae1-b0ba-618c-85d3-a214059800e2 | Solaris Kernel Zone    | ACTIVE |        |
| e82aa857-ec92-4859-f530-deb89274863e | Solaris Non-global Zone | ACTIVE |        |
+--------------------------------------+------------------------+--------+--------+
```

glance image-list コマンドは、ディスクフォーマット、コンテナフォーマット、イメージサイズなどの追加情報を表示します。

```
$ glance image-list --human-readable
```

nova image-show および glance image-show コマンドは、指定されたイメージに関する詳細情報を表示します。

```
$ nova image-show 'Solaris Kernel Zone'
+-------------------------+--------------------------------------+
| Property                | Value                                |
+-------------------------+--------------------------------------+
| OS-EXT-IMG-SIZE:size    | 1547458560                           |
| created                 | 2014-06-29T15:40:49Z                 |
| id                      | e422aae1-b0ba-618c-85d3-a214059800e2 |
| metadata architecture   | x86_64                               |
| metadata hypervisor_type | solariszones                         |
| metadata vm_mode        | solariszones                         |
| minDisk                 | 0                                    |
| minRam                  | 0                                    |
| name                    | Solaris Kernel Zone                  |
| progress                | 100                                  |
| status                  | ACTIVE                               |
| updated                 | 2014-06-29T15:40:55Z                 |
+-------------------------+--------------------------------------+
$ glance image-show 'Solaris Kernel Zone'
+---------------------------+--------------------------------------+
| Property                  | Value                                |
+---------------------------+--------------------------------------+
| Property 'architecture'   | x86_64                               |
| Property 'hypervisor_type' | solariszones                         |
| Property 'vm_mode'        | solariszones                         |
| checksum                  | b2fc9560c15603c7663326db82d5ddaa     |
| container_format          | bare                                 |
| created_at                | 2014-06-29T15:40:49.108578           |
| deleted                   | False                                |
| disk_format               | raw                                  |
| id                        | e422aae1-b0ba-618c-85d3-a214059800e2 |
| is_public                 | True                                 |
| min_disk                  | 0                                    |
| min_ram                   | 0                                    |
| name                      | Solaris Kernel Zone                  |
| owner                     | 7461d4a9f5a64af9a01ae4e84e08c182     |
| protected                 | False                                |
| size                      | 1547458560                           |
| status                    | active                               |
| updated_at                | 2014-06-29T15:40:55.769756           |
+---------------------------+--------------------------------------+
```

イメージの作成

Solaris では、OpenStack イメージは統合アーカイブです。また、Oracle Solaris 11.2 を実行している必要があります。`archiveadm` コマンドを使用すると、Oracle Solaris 11.2 を実行している大域ゾーン、非大域ゾーン、およびカーネルゾーンから新しい統合アーカイブを作成できます。OpenStack で使用できるようにイメージを Glance リポジトリにアップロードします。

統合アーカイブはクローンアーカイブか復旧用アーカイブのどちらかです。現在アクティブなブート環境に基づいてクローンアーカイブを作成するか、すべてのブート環境とシステム構成情報を含む復旧用アーカイブを作成します。クローンアーカイブには OS インスタンスのシステム構成情報は含まれません。クローンアーカイブの場合は、インストーラで強制的に再構成を行うか、ユーザーがシステム構成 (SC) プロファイルで構成を指定できます。クローンアーカイブには、アクティブでない BE なども含まれません。システムのすべてが必要な場合は、復旧用の統合アーカイブを使用してください。統合アーカイブの詳細については、『Oracle Solaris 11.2 でのシステム復旧とクローン 』を参照してください。

次のコマンドは、`myzone` という実行中の非大域ゾーンの統合アーカイブを取得します。

```
global# zonecfg -z myzone create
global# zoneadm -z myzone install
global# zlogin myzone
'sed /^PermitRootLogin/s/no$/without-password/
< /etc/ssh/sshd_config > /system/volatile/sed.$$ ;
cp /system/volatile/sed.$$ /etc/ssh/sshd_config'
global# archiveadm create -z myzone /var/tmp/myzone.uar
```

既存の VM インスタンスの*スナップショット*を作成することによって OpenStack イメージを作成することもできます。実行中の VM インスタンスのスナップショットを取得することでイメージを作成するには、`nova image-create` コマンドを使用します。

VM インスタンスの作成に使用するイメージを作成するほかに、データのバックアップや VM インスタンスの修復のためにカスタムイメージを使用することもできます。*レスキューイメージ*は、VM インスタンスが `rescue` モードにされたときにブートされる、特殊なタイプのイメージです。管理者は、問題を修正するために、レスキューイメージを使用して VM インスタンスのファイルシステムをマウントできます。

イメージストアへのイメージの追加

OpenStack のイメージサービスである Glance は、ディスクイメージとサーバーイメージの保存、発見、登録、および配布のサービスを提供します。*レジストリサーバー*は、クライアントにイメージ

のメタデータ情報を提供するイメージサービスです。*イメージキャッシュ*は、要求されるたびにイメージをイメージサーバーから再度ダウンロードする代わりに、ローカルホスト上でイメージを取得するためにイメージサービスが使用します。

次のコマンドは、前のセクションで作成された統合アーカイブを Glance リポジトリにアップロードします。フォーマットの種類として raw を使用します。必ず architecture プロパティーを指定してください。

```
global# glance image-create --container-format bare --disk-format raw
--is-public true --name "Oracle Solaris 11.2 x86 NGZ"
--property architecture=x86_64
--property hypervisor_type=solariszones
--property vm_mode=solariszones < /var/tmp/myzone.uar
```

glance image-create コマンドでは、イメージのアップロードとすべてのプロパティー値の設定を一度に行うことができます。次のスクリプトは、確実に architecture プロパティーを現在のホストのアーキテクチャーに設定してイメージをアップロードする方法を示しています。

```
#!/bin/ksh

# Upload Unified Archive image to glance with proper Solaris decorations

arch=$(archiveadm info -p $1|grep ^archive|cut -d '|' -f 4)

if [[ "$arch" == "i386" ]]; then
        imgarch=x86_64
else
        imgarch=sparc64
fi

name=$(basename $1 .uar)

export OS_USERNAME=glance
export OS_PASSWORD=glance
export OS_TENANT_NAME=service
export OS_AUTH_URL=http://controller-name:5000/v2.0

glance image-create --name $name --container-format bare --disk-format raw --owner service
--file $1 --is-public True --property architecture=$imgarch --property
 hypervisor_type=solariszones
--property vm_mode=solariszones --progress
```

VM インスタンスの作成

VM インスタンスを作成するには、フレーバ、イメージ、およびネットワークが必要です。

▼ コマンド行インタフェースを使用して VM インスタンスを作成する方法

1. **新しい VM インスタンスを作成するテナントを選択します。**

   VM インスタンスを作成するコマンドで、テナントの名前または ID を指定する必要があります。

```
$ keystone tenant-list
+----------------------------------+---------+---------+
|                id                |   name  | enabled |
+----------------------------------+---------+---------+
| 6ea34f7dafa5ce3c9a1b9de659e59d77 |   demo  |   True  |
| 0bda9b63b800ca808031a38637d50f3e | service |   True  |
+----------------------------------+---------+---------+
```

2. **新しい VM インスタンスの作成元となるイメージを選択します。**

   VM インスタンスを作成するコマンドで、イメージの名前または ID を指定する必要があります。イメージ ID など、各イメージの仕様を表示する方法については、118 ページの「イメージに関する情報の表示」を参照してください。

   大域ゾーンよりリリースレベルの低い非大域ゾーンの VM インスタンスを配備する場合、インストール時に VM インスタンスは大域ゾーンのリリースレベルに自動的にアップグレードされます。大域ゾーンよりリリースレベルの高い非大域ゾーンの VM インスタンスを配備しようとすると失敗します。

3. **新しい VM インスタンスの作成元となるフレーバを選択します。**

   必要な仕様を備えたフレーバが存在することを確認します。必要な仕様をフレーバに追加するか、必要な仕様を備えた新しいフレーバを作成します。フレーバ ID や `extra-specs` など、各フレーバの仕様を表示する方法については、116 ページの「フレーバに関する情報の表示」を参照してください。

4. **新しい VM インスタンスで使用するネットワークを選択します。**

   VM インスタンスを作成するコマンドで、ネットワークの名前または ID を指定する必要があります。`neutron net-list` コマンドで、手順 1 で選択したテナントのネットワークが表示されない場合は、`neutron net-create` コマンドを使用してこのテナントのネットワークを作成します。詳細については、OpenStack コマンド行インタフェースリファレンスを参照してください。ネットワークの ID をメモします。

5. **インスタンスを作成します。**

`nova boot` コマンドを使用して、コンピュートインスタンスを作成してブートします。*imageID* は手順 2、*flavorID* は手順 3、*nicID* は手順 4 のものです。詳細については、OpenStack コマンド行インタフェースリファレンスを参照してください。

```
# nova boot --image imageID --flavor flavorID --nic net-id=nicID
```

6. **新しい VM インスタンスで使用するフローティング IP アドレスを選択します。**

   `neutron floatingip-list` コマンドを使用して、手順 1 で選択したテナントのフローティング IP アドレスを表示します。必要に応じて、`neutron floatingip-create` コマンドを使用してこのテナントのフローティング IP アドレスを作成します。フローティング IP アドレスの ID をメモします。

7. **新しい VM インスタンスにフローティング IP アドレスを関連付けます。**

   `neutron floatingip-associate` コマンドを使用して、手順 6 で選択したフローティング IP アドレスを新しい VM インスタンスに関連付けます。

♦ ♦ ♦ 第 6 章

6 OpenStack のトラブルシューティング

この章では、次の内容について説明します。

- このリリースの既知の問題
- OpenStack に関連付けられたログファイルの使用
- 問題の調査および解決

既知の制限事項

次に、Oracle Solaris 11.2 の OpenStack (Havana 2013.2.3) に関する既知の問題を示します。

- OpenStack ダッシュボードを使用した VM インスタンスへのリモートコンソールアクセスはサポートされていません。代わりに、ダッシュボードを使用して SSH 鍵ペアをアップロードします。この鍵ペアは、その VM インスタンスの `root` の `authorized_keys` ファイルに挿入されます。
- Neutron ではネットワーク仮想化のプラグインは 1 つしかサポートされないため、完全にサポートされるのは Solaris を実行している Nova ノードだけです。
- 非大域ゾーンでは、Cinder ボリュームの接続は現在サポートされていません。
- VM インスタンスは Oracle Solaris 11.2 を実行している必要があります。
- VM インスタンスのサイズ変更はサポートされていません。

  `nova resize` コマンドはサポートされていません。`nova resize` コマンドでは、コマンドが完了したと出力に表示される場合がありますが、`nova resize-confirm` コマンドでは、インスタンスのサイズ変更は確定できないと報告され、`nova show` コマンドでは、インスタンスのサイズは変更されていないことが示されます。
- VM インスタンスのライブ移行はサポートされていません。

  `nova live-migration` コマンドはサポートされていません。
- Cinder バックアップはサポートされていません。

cinder パッケージをインストールすると cinder-backup サービスがインストールされますが、サービスは無効になっています。無効のままにしておいてください。

- ダッシュボードの「インスタンスの起動」ダイアログボックスで、「インスタンスのブートソース」には「イメージから起動」だけがサポートされています。「プロジェクト」->「イメージとスナップショット」->「アクション」メニューで、CreateVolumeFromImage はサポートされていません。Solaris OpenStack ダッシュボードのカスタマイズについては、Solaris OpenStack Horizon のカスタマイズに関する記事を参照してください。
- /etc/neutron/l3_agent.ini ファイル内の external_network_datalink オプションの値として VXLAN データリンクはサポートされていません。external_network_datalink オプションの値として VXLAN データリンクを設定した場合、Neutron L3 エージェントは外部ネットワーク上の VNIC の作成および plumb に失敗します。
- プロジェクトのネットワークリソースの割り当て制限を変更するには、コマンド行を使用する必要があります。

  ネットワークリソースの割り当て制限を Horizon から変更することはできません。プロジェクトを作成する場合、または既存のプロジェクトの非ネットワークリソースを変更する場合には、Horizon ダッシュボードを使用できます。プロジェクトのネットワーク、サブネット、ポート、ルーターまたはフローティング IP アドレスの割り当て制限を変更するには、neutron quota-update コマンドを使用する必要があります。

  非ネットワークリソースを変更する場合でも、次のエラーメッセージが表示されます。このメッセージは無視してかまいません。このメッセージに反して、非ネットワークリソースの割り当て制限は適用されています。

  ```
  Error: Modified project information and members, but unable to modify project quotas.
  ```

- SMF と OpenStack とでは、報告されるサービスのステータスが異なる場合があります。

  次の例では、nova-cert サービスが OpenStack では無効になっているにもかかわらず、SMF では online として表示されています。

  ```
  root@c190-133:~# nova service-disable c190-133 nova-cert
  +----------+-----------+----------+
  | Host     | Binary    | Status   |
  +----------+-----------+----------+
  | c190-133 | nova-cert | disabled |
  +----------+-----------+----------+
  root@c190-133:~# svcs nova-cert
  STATE          STIME    FMRI
  online         21:14:11 svc:/application/openstack/nova/nova-cert:default
  ```

- neutron-l3-agent SMF サービスが再起動時に保守に入ります。

  回避方法: ipfilter サービスを再起動し、neutron-l3-agent をクリアします。

  ```
  network# svcadm restart ipfilter:default
  network# svcadm clear neutron-l3-agent:default
  ```

- ネットワークノードのデフォルトのゲートウェイが特定の設定で削除されます。

  ネットワークノードの IP アドレスが external_network アドレス空間から派生している場合に、neutron router-gateway-clear コマンドを使用して provider_router から external_network を削除すると、ネットワークノードのデフォルトのゲートウェイが削除され、ネットワークノードにアクセスできなくなります。

  ```
  network# neutron router-gateway-clear router_UUID
  ```

  回避方法: コンソールを介してネットワークノードに接続し、デフォルトのゲートウェイを再度追加します。

- 複数のインスタンスが同時に作成されると、Nova sqlite データベースが動作しなくなります。

  多数のインスタンス (たとえば 10 個以上) が同時に作成されると、nova list コマンドはしばらく動作を停止し、次のエラーメッセージを表示します。

  ```
  $ nova list
  ERROR: The server has either erred or is incapable of performing the
  requested operation. (HTTP 500) (Request-ID:
  req-0ad63452-6753-c9fc-8275-e80604d42569)
  ```

  Horizon でもインスタンスを見つけることができません。

  この問題の原因は、Nova sqlite データベースが動作しなくなることです。しばらくすると、データベースは遅れを取り戻し、nova list と Horizon は予期したとおりに機能します。

  回避方法: sqlite の代わりに MySQL データベースを使用します。62 ページの「コンピュートノードを構成する方法」を参照してください。

ログファイルの調査

SMF サービスとさまざまな Solaris プロセスが生成するログファイルでは、エラーメッセージを探したり、画面に表示されたメッセージの詳細情報を集めたりすることができます。SMF サービスのログファイルには貴重なデバッグ情報が含まれています。

問題によっては、nova-compute、nova-scheduler、および neutron-server SMF サービスのログファイルが役立つことがあります。SMF サービスのログファイルファイルの名前を検索するには、svcs -L コマンドを使用します。

```
$ svcs -L neutron-server
/var/svc/log/application-openstack-neutron-neutron-server:default.log
```

特権ユーザーとして svcs -Lv コマンドを使用すると、サービスのログファイルを表示できます。

```
# svcs -Lv neutron-server
```

svcs -xv コマンドは、サービスの状態とともにログファイルの名前を表示します。

```
$ svcs -xv neutron-server
svc:/application/openstack/neutron/neutron-server:default (OpenStack Neutron Network Service)
 State: online since Fri Jul 25 12:11:16 2014
   See: /var/svc/log/application-openstack-neutron-neutron-server:default.log
Impact: None.
```

SMF サービスのログファイルのほかに、OpenStack サービスがログファイルを生成し、多くの Solaris プロセスが独自のログファイルを持っています。一部の OpenStack サービスは、/var/log ディレクトリ内の OpenStack サービス名の下に情報を記録します。たとえば、OpenStack イメージストアのログファイルは /var/log/glance にあります。VM インスタンスの作成とブートで問題が発生している場合は、/var/log/zones ディレクトリを調べてください。

ほとんどの OpenStack 構成ファイルは、/etc ディレクトリ内の OpenStack サービス名の下にあります。たとえば、OpenStack ネットワークの構成ファイルは /etc/neutron にあります。Horizon の構成ファイルは /etc/openstack_dashboard にあります。

より多くの情報を OpenStack サービスのログファイルに受け取るには、そのサービスの構成ファイルで verbose オプションを設定します。構成ファイルでは、verbose オプションがすでに false に設定されているかコメントアウトされている場合があります。コメントを解除するか verbose オプションを追加し、verbose=true を設定します。同様に、構成ファイルで debug=true を設定すると、その構成ファイルの影響を受ける操作からより多くの出力が表示されます。Oracle Solaris 11.2 での OpenStack の入門ガイドにある OpenStack の一般的な構成パラメータに関するセクション、または OpenStack ドキュメントサイトにある *OpenStack 構成リファレンス*で、構成オプションの表を参照してください。

構成ファイルで debug=true を設定するのと同様に、OpenStack サービスの個々のコマンドに --debug オプションを指定できます。

OpenStack サービスのログファイルを表示したり、pfedit コマンドを使用して OpenStack サービスの構成ファイルを変更したりするには、適切な RBAC プロファイルを引き受けてください。次のプロファイルを割り当てることができます。

- OpenStack ブロックストレージの管理
- OpenStack コンピューティングの管理
- OpenStack アイデンティティーの管理
- OpenStack イメージの管理
- OpenStack ネットワークの管理
- OpenStack オブジェクトストレージの管理
- OpenStack の管理

問題の調査および解決

サービス名を指定せずに svcs -x コマンドを使用すると、maintenance 状態になっているすべてのサービスが表示されます。すべての OpenStack ノードで svcs コマンドを使用して、必要なサービスがすべて online 状態になっていることを確認してください。必要なサービスが online 状態になっていない場合は、127 ページの「ログファイルの調査」の説明に従ってログファイルを確認します。

SMF サービスに関する問題を修正するための一般的なヘルプについては、『Oracle Solaris 11.2 でのシステムサービスの管理 』の「サービス問題のトラブルシューティング」を参照してください。

ネットワークの問題と考えられる場合は、すべてのノードで evsadm コマンドを実行します。

OpenStack ダッシュボードの使用中に操作を正常に完了できない問題が発生している場合は、同じ操作をコマンド行で試みてください。コマンド行では、より有用なエラーメッセージが表示されることがあります。コマンドに詳細出力オプションが指定されているかどうかを確認してください。

コントローラノード以外のノードで問題が発生している場合は、そのノード上で nova list などの単純なコマンドをいくつかコマンド行で実行して、そのノードがコントローラノードと通信できることを確認します。

OpenStack のインストールと構成

このセクションでは、OpenStack をインストールして構成する際に発生する可能性のあるエラーについて説明します。

ダッシュボードのエラー

次のメッセージのような「承認されていない」というエラーメッセージが表示される場合は、RSA ホスト鍵が変更されたかどうかを確認してください。

```
Error: Unauthorized: Unable to retrieve usage information.
Error: Unauthorized: Unable to retrieve quota information.
Error: Unauthorized: Unable to retrieve project list information.
Error: Unauthorized: Unable to retrieve instance list information.
```

VM インスタンスのインストールと構成

このセクションで説明する問題は、特に VM インスタンスに関連しています。

VM インスタンスがエラー状態になっている

VM のインスタンスがエラー状態になる理由の 1 つは、ホストシステムとは異なるアーキテクチャーの VM インスタンスをインストールしようとしたというものです。この場合、アーキテクチャーの不一致を明確に示すエラーメッセージは表示されない可能性があります。この問題を回避するには、`glance` イメージストアにイメージをアップロードする際に、イメージの `architecture` プロパティーを正しく設定してください。Horizon を使用してイメージをアップロードする場合は、アップロード後にイメージのプロパティーを設定する必要があります。あるいは、コマンド行を使用すれば、イメージのアップロードとプロパティー値の設定を 1 つの `glance image-create` コマンドで行うことができます。例については、120 ページの「イメージストアへのイメージの追加」を参照してください。

VM インスタンスのプロパティー値がゾーンのプロパティー値と一致しない

VM インスタンスに関して OpenStack が報告する情報の一部は、対応するゾーンに関して Solaris が報告する情報と一致しません。Horizon に表示される情報と `nova` コマンドで表示

される情報は、zoneadm コマンドやほかの Solaris コマンドで表示される情報と一致しない場合があります。

名前
: Horizon または nova list コマンドで表示される VM インスタンスの名前は、インスタンスを作成したときに割り当てた名前です example-instance など)。zoneadm list コマンドで表示されるゾーンの名前は instance-00000001 のようになります。どのゾーンがどの VM インスタンスに関連付けられているかを特定するには、nova show コマンドを使用します。nova show の出力の中で、OS-EXT-SRV-ATTR:instance_name プロパティーの値はゾーンの名前、name プロパティーの値は VM インスタンスの名前です。

UUID
: Horizon または nova show コマンドで表示される VM インスタンスの UUID は、zoneadm list -p コマンドで表示される同じゾーンの UUID と一致しません。zoneadm コマンドが表示する UUID は、Nova に使用される識別子とは異なる識別子です。

CPU
: Horizon に表示される VM インスタンスの VCPU 数は、そのインスタンスで使用できる分割 CPU 数の限度においてのみ仮想化される、*上限が定義された* CPU の数です。この数からは、インスタンス内部の制限前の数を把握することはできません。psrinfo コマンドは、ゾーンに割り当てられている専用 CPU 数を報告します。

メモリー
: Horizon に表示される VM インスタンスのメモリー量は、その VM インスタンスにログインしているときに prtconf コマンドで表示されるメモリー量とは異なる場合があります。Horizon は、VM インスタンスの作成に使用されたフレーバが指定しているメモリー量を表示します。prtconf コマンドは、すべてのシステムメモリーを報告します。

ストレージ
: VM インスタンスが Zones on Shared Storage (ZOSS) を使用した非大域ゾーンである場合を除き、Horizon に表示される VM インスタンスのストレージ量は、その VM インスタンスにログインしているときに表示されるストレージ量とは異なる場合があります。

ネットワークの廃棄

ネットワークノードに Neutron を構成するときに問題が発生し、最初からやり直すために構成を廃棄する必要がある場合は、この手順に従います。どのポイントから構成の取り消しを始める必要があるかに応じて、この手順に示す順序に従ってください。

▼ Neutron のネットワーク構成を削除する方法

1. **Horizon ダッシュボードでこの手順を実行します。**

a. すべてのフローティング IP アドレスの関連付けを解除します。

b. すべてのフローティング IP アドレスを削除します。

2. 端末ウィンドウで、次のコマンドを入力します。

```
# neutron router-gateway-clear router-id external-network-id
# neutron router-interface-delete router-id subnet-id
```

a. ルーターのゲートウェイインタフェースを削除するため、次のコマンドを入力します。

```
# neutron router-gateway-interface-delete router-id external-network-id
```

b. ルーターの残りのインタフェースを削除するため、次のコマンドを入力します。

```
# neutron router-interface-delete router-id subnet-id
```

3. Horizon ダッシュボードで次を実行します。

a. すべての VM インスタンスを終了します。

b. サブネットを削除します。

サブネットの削除中に問題が発生した場合は、132 ページの「仮想ポートを削除する方法」を参照してください。

c. ネットワークを削除します。

▼ 仮想ポートを削除する方法

サブネットを削除できない問題が発生した場合は、この手順を使用してください。

1. どの仮想ポートが現在使用されているかを特定します。

```
# evsadm
```

2. 使用されている仮想ポートをリセットします。

```
# evsadm reset-vport vport
```

3. 仮想ポートを削除します。

```
# evsadm delete-vport vport
```

索引