

# Oracle® Solaris での OpenStack (Juno) のインストールと構成

ORACLE®

Part No: E62829  
2016 年 6 月



## Part No: E62829

Copyright © 2014, 2016, Oracle and/or its affiliates. All rights reserved.

このソフトウェアおよび関連ドキュメントの使用と開示は、ライセンス契約の制約条件に従うものとし、知的財産に関する法律により保護されています。ライセンス契約で明示的に許諾されている場合もしくは法律によって認められている場合を除き、形式、手段に関係なく、いかなる部分も使用、複写、複製、翻訳、放送、修正、ライセンス供与、送信、配布、発表、実行、公開または表示することはできません。このソフトウェアのリバース・エンジニアリング、逆アセンブル、逆コンパイルは互換性のために法律によって規定されている場合を除き、禁止されています。

ここに記載された情報は予告なしに変更される場合があります。また、誤りが無いことの保証はいたしかねます。誤りを見つけた場合は、オラクルまでご連絡ください。

このソフトウェアまたは関連ドキュメントを、米国政府機関もしくは米国政府機関に代わってこのソフトウェアまたは関連ドキュメントをライセンスされた者に提供する場合は、次の通知が適用されます。

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

このソフトウェアまたはハードウェアは様々な情報管理アプリケーションでの一般的な使用のために開発されたものです。このソフトウェアまたはハードウェアは、危険が伴うアプリケーション(人的傷害を発生させる可能性があるアプリケーションを含む)への用途を目的として開発されていません。このソフトウェアまたはハードウェアを危険が伴うアプリケーションで使用する場合、安全に使用するために、適切な安全装置、バックアップ、冗長性(redundancy)、その他の対策を講じることは使用者の責任となります。このソフトウェアまたはハードウェアを危険が伴うアプリケーションで使用したこと起因して損害が発生しても、Oracle Corporationおよびその関連会社は一切の責任を負いかねます。

OracleおよびJavaはオラクル およびその関連会社の登録商標です。その他の社名、商品名等は各社の商標または登録商標である場合があります。

Intel, Intel Xeonは、Intel Corporationの商標または登録商標です。すべてのSPARCの商標はライセンスをもとに使用し、SPARC International, Inc.の商標または登録商標です。AMD, Opteron, AMDロゴ, AMD Opteronロゴは、Advanced Micro Devices, Inc.の商標または登録商標です。UNIXは、The Open Groupの登録商標です。

このソフトウェアまたはハードウェア、そしてドキュメントは、第三者のコンテンツ、製品、サービスへのアクセス、あるいはそれらに関する情報を提供することがあります。適用されるお客様とOracle Corporationとの間の契約に別段の定めがある場合を除いて、Oracle Corporationおよびその関連会社は、第三者のコンテンツ、製品、サービスに関して一切の責任を負わず、いかなる保証もいたしません。適用されるお客様とOracle Corporationとの間の契約に定めがある場合を除いて、Oracle Corporationおよびその関連会社は、第三者のコンテンツ、製品、サービスへのアクセスまたは使用によって損失、費用、あるいは損害が発生しても一切の責任を負いかねます。

### 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方法 .....	9
<b>1 OpenStack (Juno) の概要 .....</b>	<b>11</b>
Oracle Solaris での Juno バージョンの新機能 .....	11
アップグレードの注意事項 .....	13
このドキュメントの使用方法 .....	13
<b>2 シングルノードへの OpenStack のインストール .....</b>	<b>15</b>
OpenStack 統合アーカイブの配備 .....	15
OpenStack UA イメージについて .....	15
イメージファイルのダウンロード .....	16
単一システム構成のインストール .....	17
▼ 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 .....	17
▼ 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 .....	18
OpenStack ダッシュボードの使用 .....	20
▼ OpenStack ダッシュボードにアクセスする方法 .....	20
ダッシュボードの詳細 .....	20
<b>3 マルチノード OpenStack 構成用に複数のシステムをまたがるインストール .....</b>	<b>23</b>
3 ノードアーキテクチャーの概要 .....	23
準備の手順 .....	26
ホスト名、変数、およびパスワードの準備 .....	26
サンプルの Keystone スクリプト .....	27
構成ファイルの編集について .....	28
メモリー使用の最適化 .....	28
NTP サーバーの構成 .....	29
▼ NTP サーバーを設定する方法 .....	29
コントローラノードの構成 .....	30

NTP クライアントの構成 .....	30
MySQL のインストール .....	31
Keystone のインストール .....	32
Glance のインストール .....	33
Nova のインストール .....	35
Horizon のインストール .....	36
Cinder のインストール .....	37
Neutron のインストール .....	38
コンピュータノードの構成 .....	41
▼ コンピュータノードを構成する方法 .....	41
▼ コンソールアクセスを有効にする方法 .....	44
ストレージノードの構成 .....	46
▼ ブロックストレージノードを構成する方法 .....	46
Swift オブジェクトストレージの構成 .....	47
▼ Swift プロキシコントローラサービスノードを構成する方法 .....	48
▼ オブジェクトストレージノードを構成する方法 .....	49
<b>4 インストール後タスクおよび構成タスク .....</b>	<b>51</b>
OpenStack プロジェクト用の外部ネットワークの準備 .....	51
プロバイダルーターについて .....	51
外部ネットワークの作成 .....	53
Glance リポジトリ用のイメージの準備 .....	57
イメージの作成 .....	58
イメージに関する情報の表示 .....	59
Glance イメージ作成スクリプトの使用 .....	61
<b>5 クラウドの管理 .....</b>	<b>63</b>
プロジェクトとユーザーの作成 .....	63
▼ プロジェクトを作成してユーザーを割り当てる方法 .....	63
▼ 既存のユーザーをプロジェクトに移入する方法 .....	64
プロジェクト用の内部ネットワークの作成 .....	66
▼ プロジェクトにネットワークを構成する方法 .....	66
▼ 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 .....	68
▼ 内部ネットワークに外部接続を提供する方法 .....	69
L3 エージェント構成情報の表示 .....	70
VM インスタンスの作成とブート .....	71
▼ SSH 鍵ペアを作成する方法 .....	71
▼ VM インスタンスを作成する方法 .....	72
▼ VM インスタンスにユーザーを追加する方法 .....	75

---

フレーバの管理 .....	76
フレーバに関する情報の表示 .....	77
フレーバ仕様の変更 .....	77
VM インスタンスの管理 .....	80
VM インスタンスの移行および退避 .....	80
VM インスタンスのサイズ変更 .....	82
<b>6 Cinder の構成と配備のためのオプション .....</b>	<b>85</b>
ストレージのためのリモートシステムの配備 .....	85
cinder.conf ファイルの構成 .....	86
指定されたユーザーへの権利の付与 .....	89
ターゲットとしてのリモートホストの有効化 .....	89
Oracle ZFS Storage Appliance での OpenStack の使用 .....	90
Oracle ZFS Storage Appliance について .....	90
Oracle ZFSSA を使用した OpenStack の構成 .....	90
▼ OpenStack 用に Oracle ZFSSA を構成する方法 .....	91
<b>7 Heat の操作 .....</b>	<b>95</b>
Heat コンポーネントについて .....	95
Heat のインストール .....	96
▼ Heat を構成する方法 .....	96
HOT テンプレートについて .....	96
<b>8 Ironic の操作 .....</b>	<b>99</b>
Ironic コンポーネントについて .....	99
Ironic のインストールおよび構成 .....	100
▼ Ironic をインストールおよび構成する方法 .....	100
概要: Ironic でのベアメタルの配備 .....	104
Ironic を使用したベアメタルの配備 .....	106
▼ UAR ファイルからベアメタルを配備する方法 .....	106
▼ ノードを廃棄する方法 .....	110
<b>9 OpenStack のトラブルシューティング .....</b>	<b>113</b>
コマンド行ヘルプの取得 .....	113
既知の制限事項 .....	114
ログファイルの調査 .....	115
問題の調査および解決 .....	117
VM インスタンスのインストールと構成 .....	118

---

Horizon 関連の問題 .....	121
Heat 関連の問題 .....	122
スケーラビリティの問題 .....	122
ネットワークの廃棄 .....	123
▼ Neutron のネットワーク構成を削除する方法 .....	123
▼ 仮想ポートを削除する方法 .....	124
デバッグに関する一般的なヒントとコツ .....	124
役立つサイト .....	124
<b>A 一般的な OpenStack 構成ファイルおよびサービス .....</b>	<b>127</b>
構成ファイル .....	127
Cinder ファイル .....	127
Glance ファイル .....	127
Keystone ファイル .....	127
Neutron ファイル .....	128
Nova ファイル .....	128
Horizon ファイル .....	128
Swift ファイル .....	128
OpenStack SMF サービス .....	129
Cinder .....	129
Glance .....	129
Keystone .....	129
Neutron .....	129
Nova * .....	129
Swift .....	130
<b>索引 .....</b>	<b>131</b>



## このドキュメントの使用方法

---

- **概要** – Oracle Solaris 11.3 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** – Oracle 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=E69401> 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

## OpenStack (Juno) の概要

---

この章では、OpenStack の Juno バージョンで導入された機能について説明します。また、この本を使用する方法についても説明します。

OpenStack および Oracle Solaris テクノロジーの概要については、『[Overview: OpenStack Components and Oracle Solaris Technologies](#)』を参照してください。Oracle Solaris に OpenStack をインストールするための計画のガイドラインと要件については、『[Planning for an OpenStack Configuration](#)』も参照してください。

この章では、次の内容について説明します。

- 11 ページの「Oracle Solaris での Juno バージョンの新機能」
- 13 ページの「アップグレードの注意事項」
- 13 ページの「このドキュメントの使用法」

## Oracle Solaris での Juno バージョンの新機能

このバージョンで OpenStack コアコンポーネントに導入された機能については、次のリリースノート、および必要に応じてその前のバージョンを参照してください。

- Juno のリリースノート (<https://wiki.openstack.org/wiki/ReleaseNotes/Juno>)
- Icehouse のリリースノート (<https://wiki.openstack.org/wiki/ReleaseNotes/Icehouse>)

現在の Juno リリースでの問題のリストについては、[My Oracle Support \(https://support.oracle.com\)](https://support.oracle.com) にある各バージョンに対応する README ファイルを参照してください。

OpenStack Juno では、次の Oracle Solaris 機能が提供されます。

- VM インスタンスのコンソールサポート。ユーザーのリクエストに基づき、各インスタンスのコンソールにブラウザからアクセスできます。
- Horizon では、システムで利用可能な場合は Apache 2.4 がデフォルトで使用されます。そのため、TLS と HTTP の両方の `httpd.conf` フラグメントが Apache 2.4 の `samples-conf.d` ディレクトリに含まれています。
- Neutron の場合は、テナントネットワークの完全な IPv6 サポート、およびソース NAT (SNAT) のサポート。

デフォルトで、SNAT は Neutron ルーターのゲートウェイインタフェース上で有効です。この機能を無効にするには、`neutron router-gateway-set --disable-snat` コマンドを実行します。

さらに Neutron では、次の機能が Juno でサポートされています。

- メタデータサービス
- フラット (タグなし) レイヤー 2 ネットワークタイプ
- 新しい Neutron サブコマンド

これらの Neutron 機能の詳細は、[Solaris OpenStack Juno Neutron の新機能](#)の項目を参照してください。

- Swift 2.2.2 のアップグレードサービスに対応する追加の SMF サービスが導入され、ネットワークパーティションからの回復後に Swift コンテナ内のオブジェクトを調整します。
- OpenStack および EC2 と互換性のあるメタデータサービスのサポート。
- Oracle Solaris 11.3 SRU 2 リリースからは、Cinder に関連した次の機能が追加されました。

- リモート SAN ストレージの使用

ローカルで管理されている ZFS ボリュームでの動作とは別に、Paramiko のサポートにより、Cinder はまた、リモート Solaris ノード上のこれらの ZFS ボリュームも iSCSI またはファイバチャネルプロトコルを使用した SAN ストレージとしてプロビジョニングできます。リモートシステムは、複数のストレージバックエンドに構成することによって調整できます。

- ボリュームのバックアップおよび復元操作のサポート

Oracle Solaris では現在、Cinder バックアップ SMF サービスが有効になっています。そのため、未接続のボリュームをバックアップし、構成済みバックエンド間で復元できます。現在は、Swift が唯一のサポートされるバックエンドです。

- Cinder ボリュームの移行のサポート

Cinder では、Cinder ボリュームの移行に ZFS 操作を利用します。ZFS の送受信プロセスを使用すると、異なる構成済み Cinder バックエンド間でのボリュームの移行が可能になります。移行先が移行元と同じ zpool に存在する場合は、ZFS の名前変更操作が使用されます。現在、Cinder ボリュームの移行のサポートは単一システムに制限されています。

- Oracle Solaris 11.3 SRU 6 リリースからは、Nova に関連した次の機能が追加されています。

- ライブ移行

Horizon サービスのダッシュボードまたは `nova` コマンドを使用してライブ移行を開始できます。参加しているノードの中からスケジューラが移行のターゲットホストを選択します。これらのノードには、互いのパブリック SSH 鍵が存在する必要があります。[80 ページの「VM インスタンスの移行および退避」](#)を参照してください。

暗号化により、ライブ移行操作は確実にセキュアになります。`/etc/nova.conf` ファイル内のパラメータを使用して、使用する暗号化を手動で選択するか、またはプロセスに適切な暗号化を選択させることができます。

- インスタンス退避のサポート

nova evacuate の機能です。ホストに障害が発生した場合や、ホスト上でサービスが無効になっている場合は、インスタンスを別のノードに安全に移動して、そのノードから回復できます。退避のサポートは、ルートデバイスが共有ストレージ上に存在する場合にのみ利用できます。さらに、退避は非大域ゾーンではなく、カーネルゾーンでのみサポートされます。

- VM インスタンスのサイズ変更

VM インスタンスの新しいフレーバへのサイズ変更がサポートされています。VM インスタンスについて、CPU の数や使用可能なメモリーの量を増やしたり減らしたりできます。さらに、インスタンスで使用できるルートボリュームのサイズを増やすこともできます。ただし、ルートボリュームのサイズを減らすことは現在サポートされていません。

## アップグレードの注意事項

OpenStack の Juno バージョンは、次のバージョンの Oracle Solaris でサポートされています。

- Oracle Solaris 11.2 SRU 10 以降のリリース
- Oracle Solaris 11.3

Oracle Solaris 11.2 の SRU 10 または SRU 11 リリースを実行しているシステム上に既存の Havana 構成が存在する場合は、<https://community.oracle.com/docs/DOC-910993> に記載されている手動の手順に従って Juno バージョンにアップグレードしてください。Oracle Solaris 11.2 SRU 12 からは、Havana から Juno へのアップグレードプロセスが自動的に実行されます。

---

**注記** - 既存の Havana 構成を Juno にアップグレードしている場合は、現在 Neutron に関する問題が存在します。特定の場合に、構成ミスによって、Neutron データベースが MySQL ではなく、SQLite を使用するようにアップグレードされます。SQLite はサポートされておらず、本番モードで使用するには不適切であるため、MySQL にデータベースを手動で移行する必要があります。詳細は、[Oracle OpenStack for Oracle Solaris での Neutron データベースの SQLite から MySQL への移行](#)の項目を参照してください。

---

## このドキュメントの使用法

このドキュメントでは主に Oracle Solaris に固有の OpenStack の情報を説明します。すべてのプラットフォームに共通して実行する機能や操作は、一般に本書の対象外です。

この Oracle Solaris リリースの OpenStack の基になっている Juno 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 Juno 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*

Oracle Solaris の詳細については、使用している Oracle Solaris バージョンの[情報ライブラリ](#)にアクセスしてください。[クラウド管理](#)のサイトでも追加情報を入手できます。

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	Oracle 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 のテクノロジー。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ー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。

## ◆◆◆ 第 2 章

# シングルノードへの OpenStack のインストール

---

この章では、OpenStack を単一システムにインストールし、シングルノード構成を作成することによってすぐに使用し始める方法について説明します。

この章の内容は次のとおりです。

- 15 ページの「OpenStack 統合アーカイブの配備」
- 17 ページの「単一システム構成のインストール」
- 20 ページの「OpenStack ダッシュボードの使用」

シングルノード OpenStack インストールの最適な使用方法は評価の目的や、その動作のしくみに慣れるための使用です。実際の本番での使用では、OpenStack を複数のシステムまたはノードにインストールおよび構成するようにしてください。マルチノード OpenStack インストールについては、[第3章「マルチノード OpenStack 構成用に複数のシステムをまたがるインストール」](#)を参照してください。

## OpenStack 統合アーカイブの配備

Oracle Solaris は、実行中の OpenStack 構成をユーザーがすばやく入手できるようにする OpenStack 統合アーカイブを提供します。

## OpenStack UA イメージについて

このアーカイブをインストールすると、すべての基本的な OpenStack サービスがシステムにインストールおよび事前構成され、すぐに使用できるようになります。Nova インスタンスとも呼ばれる仮想マシン (VM) を作成し、それにブロックストレージを割り当てることができます。これらの VM を仮想ネットワークに接続したあと、Web ブラウザインタフェースを使用してそれらの VM を管理できます。

OpenStack 統合アーカイブは次の機能があります。

- 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。
- Oracle 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。

## イメージファイルのダウンロード

Oracle Solaris 統合アーカイブのページから、USB イメージまたは UAR イメージのどちらかをダウンロードできます。どちらのイメージでも、次のさまざまなインストール方法を使用して、インストールのターゲットに応じたシングルノード構成を作成できます。

- ベアメタルターゲット。
  - Automated Installer ネットワークサービスを使用したインストール。
  - `archiveadm(1M)` コマンドを使用して統合アーカイブから生成された USB イメージを使用したインストール。
  - Oracle Solaris Automated Installer ブートイメージを統合アーカイブと組み合わせて使用した間接的なインストール。
- カーネルゾーンターゲット - 標準のゾーンコマンドを使用したインストール。

簡略化のために、この章では次の 2 つのインストール方法に焦点を絞ります。各方法には、インストールの特定のターゲットがあります。

- Automated Installer ネットワークサービスを使用した、ベアメタルへの統合アーカイブのインストール。
- ゾーンコマンドを使用した、カーネルゾーンへの統合アーカイブのインストール。

---

**注記** - この章では、一般的なインストール手順を示します。詳細な手順については、アーカイブに関連付けられた README ファイルを参照してください。Oracle Solaris のインストール方法の詳細は、使用している Oracle Solaris バージョンの [ライブラリ](#)にあるインストールの本を参照してください。

---

### ▼ 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. ダウンロードのタブから、「Solaris」を選択し、統合アーカイブをクリックします。
3. 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。
4. システムのアーキテクチャー用の適切な OpenStack \*.uar ファイルをダウンロードします。
5. 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
```

## 単一システム構成のインストール

このセクションでは、シングルノード OpenStack 構成を作成するためにダウンロードした統合アーカイブをインストールする方法について説明します。このセクションでは、次の 2 つの方法について説明します。

- Automated Installer ネットワークサービスを使用した、ベアメタルへの統合アーカイブのインストール。  
Automated Installer の詳細は、使用している Oracle Solaris バージョンの[ライブラリ](#)にあるインストールの本を参照してください。
- カーネルゾーンへの統合アーカイブのインストール。  
カーネルゾーンの詳細は、同じ[ライブラリ](#)にある『*Oracle Solaris カーネルゾーンの作成と使用*』を参照してください。

### ▼ 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

**始める前に** この手順を実行するには、AI インストールサーバーに統合アーカイブファイルをダウンロードしている必要があります。[16 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. AI インストールサーバー上で、次の手順を実行します。
  - a. AI マニフェストを作成または変更します。  
たとえば、AI インストールサーバーの `/usr/share/auto_install/manifest/default_archive.xml` を使用し、それを AI 設定に従って変更できます。この OpenStack インストールに適用される次のフラグメントがマニフェストに含まれていることを確認してください。

```
<software type="ARCHIVE">
  <source>
    <file uri="full-path-to-the-JAR-file" />
  </source>
  <software_data action="install">
    <name>*</name>
  </software_data>
</software>
```

b. 前の手順の AI マニフェストを使用する AI インストールサービスを設定します。

SPARC システムの統合アーカイブをダウンロードしたとします。マニフェストを作成したあと、それを既存の default-sparc AI 別名サービスに関連付ける必要があります。次の例は、まず使用可能なサービスをオプションで一覧表示してから、マニフェストを選択されたサービスに関連付ける方法を示しています。最後の手順として、関連付けの結果を確認するために、サービスを再度一覧表示します。

```
# installadm list -m
Service Name      Manifest Name    Type      Status  Criteria
-----
default-sparc    orig_default    derived   default none
solaris11_3-sparc orig_default    derived   default none

# installadm create-manifest -n default-sparc \
-m os_manifest -f openstack_manifest.xml -d
Created Manifest: 'os_manifest'

# installadm list -m
Service Name      Manifest Name    Type      Status  Criteria
-----
default-sparc    os_manifest     xml       default none
                  orig_default    derived   default none
solaris11_3-sparc orig_default     derived   default none
```

2. ターゲットシステムをブートします。

```
{0} ok boot net - install
```

インストールプロセスが完了するまでに少し時間がかかります。

3. 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。

4. SCI ツールから入力を求められたら、情報を指定します。

5. Horizon ダッシュボードにログインします。

[20 ページの「OpenStack ダッシュボードの使用」](#)を参照してください。

## ▼ 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

始める前に 次のことを確認してください。

- カーネルゾーンをホストするターゲットシステムが仮想化の要件を満たしていること。

インストール要件は、『[Planning for an OpenStack Configuration](#)』で説明されています。

- 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。16 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。

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。

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。

```
# zonecfg -z kzone-name
zonecfg:OpenStackKZ> select 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、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するために、カーネルゾーンには 50G バイトのディスク容量が使用されます。

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

ここで、*archive-path* は、OpenStack 統合アーカイブの場所のフルパス名です。

#### 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。

7. **Horizon ダッシュボードにログインします。**  
20 ページの「OpenStack ダッシュボードの使用」を参照してください。

## OpenStack ダッシュボードの使用

OpenStack のインストールおよびインストール後構成タスクを完了したら、OpenStack ダッシュボードにログインして、クラウド内の使用可能なリソースを表示します。

### ▼ 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。

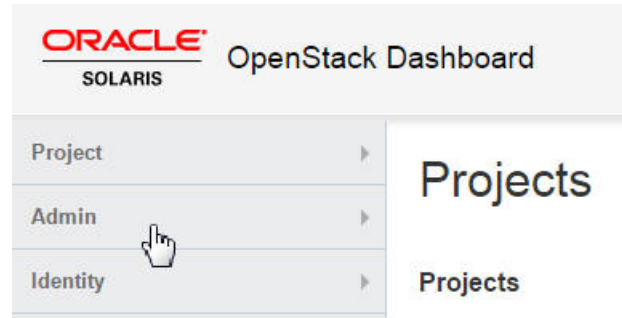
カーネルゾーンに統合アーカイブをインストールした場合、OpenStack システムはカーネルゾーンであり、*system* はカーネルゾーンの名前または IP アドレスです。

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

### ダッシュボードの詳細

最初に、ユーザー admin として Horizon ダッシュボードにログインします。それにより、プロジェクト demo のランディングページが開かれます。admin には demo に対する管理特権があるため、このページの左パネルには「プロジェクト」、「管理」、「識別情報」の 3 つのタブが表示され

ます。管理特権がない場合、ユーザーには「プロジェクト」タブと「識別情報」タブのみが表示されます。



**注記** - ユーザー名 admin は、その役割には関係ありません。たとえば、admin には demo に対する管理特権があります。admin をユーザーとして新しいプロジェクトに追加しても、そのプロジェクトに管理特権が移管されるわけではありません。代わりに、admin にはメンバー特権のみが付与され、「管理」タブにはアクセスできません。

「管理」パネルの「使用法のサマリー」概要ページは、デフォルトのクラウド管理者ビューです。

図 1 OpenStack ダッシュボードの「管理」の「概要」ウィンドウ



「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
  - ベースとなる Oracle 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

シングルノード構成の OpenStack UA では、次の事前に構成されたリソースが提供されます。

- 2 つのイメージ: Solaris 非大域ゾーンと Solaris カーネルゾーン
- 2 つのプロジェクトまたはテナント: demo と service

demo テナントは、admin をその単一のメンバーとして含むデフォルトのテナントです。

service テナントは、複数のテナントにわたって共有されるリソースを作成するためにクラウド管理者によって使用されます。たとえば、このドキュメントでは、service テナント内に Neutron ルーターが作成され、そのルーターがすべてのテナントによって共有されます。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、service テナント内で管理特権を持っています。

- 10 個のフレーバ

OpenStack 統合アーカイブに含まれていたりポジトリ内の事前に構成されたイメージを表示するには、次のタブのいずれかをクリックします。

- 「管理」 > 「システム」 > 「イメージ」タブ。
- 「プロジェクト」 > 「コンピューター」 > 「イメージ」タブ。

自動的に使用可能な Oracle Solaris フレーバを一覧表示するには、「管理」 > 「システム」 > 「フレーバー」タブをクリックします。

次のビデオプレゼンテーションは、ダッシュボードの完全な概要を提供しています。

- [OpenStack Dashboard - パート 1](#)
- [OpenStack Dashboard - パート 2](#)

ダッシュボードで実行できるタスクについては、[第5章「クラウドの管理」](#)を参照してください。

# ◆◆◆ 第 3 章

## マルチノード OpenStack 構成用に複数のシステムをまたがるインストール

---

この章では、マルチノード OpenStack 構成をインストールする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「シングルノードへの OpenStack 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。

- [23 ページの「3 ノードアーキテクチャーの概要」](#)
- [26 ページの「準備の手順」](#)
- [30 ページの「コントローラノードの構成」](#)
- [41 ページの「コンピューターノードの構成」](#)
- [46 ページの「ストレージノードの構成」](#)
- [47 ページの「Swift オブジェクトストレージの構成」](#)

### 3 ノードアーキテクチャーの概要

シングルノード構成は、OpenStack を製品としてテストしたり、その機能に精通したり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適していません。この環境では、OpenStack を複数のシステムまたはノードにまたがってインストールおよび構成します。

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ターインスタンスを使用できます。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。

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** - 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サービスがあります。さらに、Nova コンピューター管理サービスと Neutron サーバーもこのノードに構成されます。

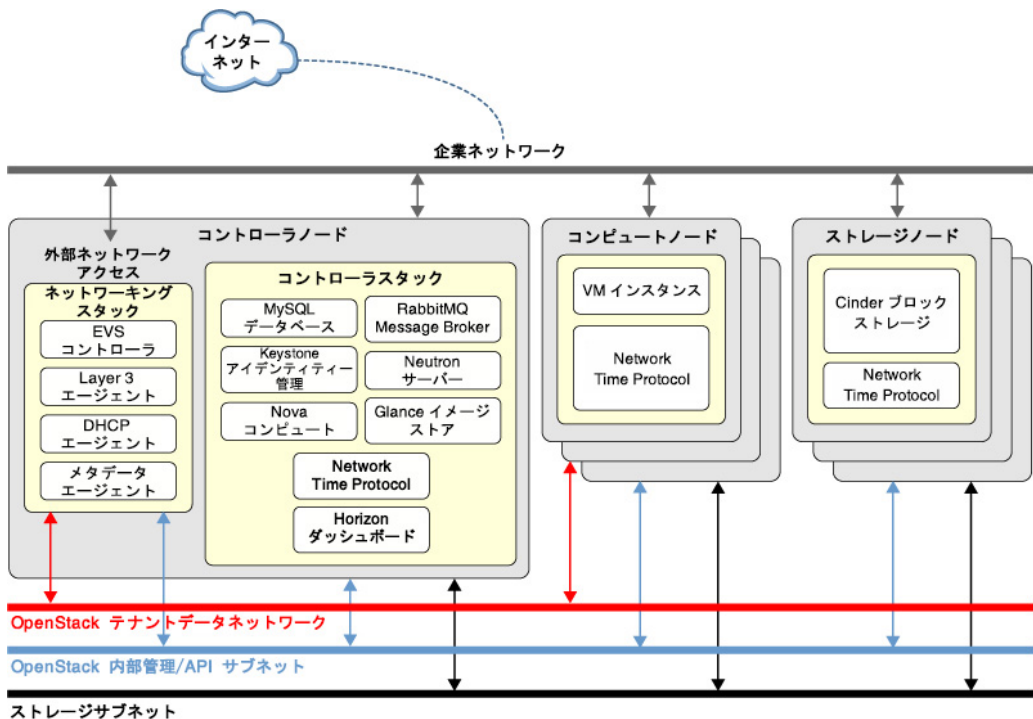
- コンピュートノード - VM インスタンス (Nova コンピュートインスタンスとも呼ばれる) がインストールされているノード。このノードは、これらの VM インスタンスを管理する計算デーモンを実行します。
- ストレージノード - データをホストするノード。

この 3 ノードアーキテクチャーは、複数のシステムに OpenStack を配備する方法の 1 つではありません。柔軟性が高いため、このアーキテクチャー以外の方法でも OpenStack コンポーネントを配布できます。したがって、インストールを開始する前に、クラウド構成を計画する必要があります。計画のガイドについては、『[OpenStack 構成の計画](#)』を参照してください。

**注記** - OVM Server for SPARC (LDom) を実行するサーバー上で単一 Oracle SPARC サーバーを分割してマルチノード 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 に関する説明](#)を参照してください。この記事では、OpenStack の Havana バージョンについて具体的に言及しています。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 2 3 ノード構成のリファレンスアーキテクチャー



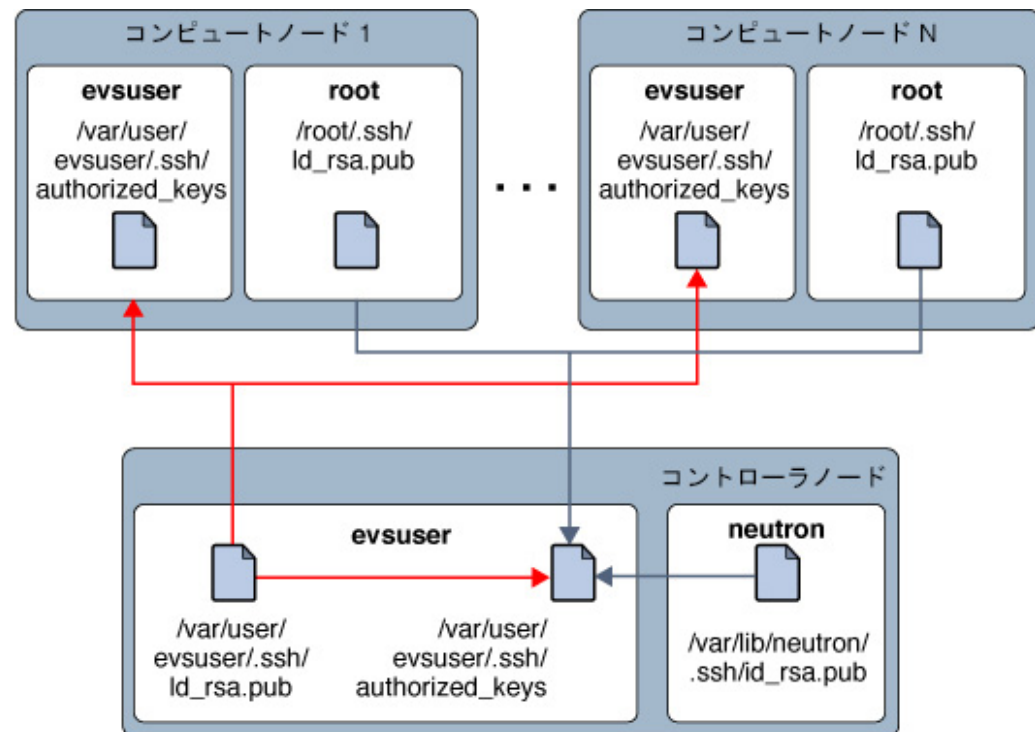


この図は、Cinder を使用してストレージノードを示しています。ただし、Swift オブジェクトストレージサービスも同様に構成できます。

Oracle Solaris では、エラスティック仮想スイッチ (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 上にある 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、同じコンピュータノードまたは複数のコンピュータノードに配置できます。EVS の詳細は、『*Oracle Solaris での仮想ネットワークとネットワークリソースの管理*』の仮想スイッチに関する説明を参照してください。この本は、[オペレーティングシステムのドキュメント](#)内の使用している Oracle Solaris バージョンのライブラリにあります。

異なるノードが互いに通信するには、コントローラノード内の `evsuser`、`neutron`、および `root` の SSH 公開鍵が、構成されているすべてのコンピュータノード内の `evsuser` の各 `authorized_keys` ファイル内に存在する必要があります。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、複数のコンピュータノードが構成されていると仮定しています。

図 3 EVS コントローラの SSH 鍵の配布



Oracle Solaris システムへの OpenStack の配備に役立つ OpenStack 構成パラメータのリストについては、[付録A 一般的な OpenStack 構成ファイルおよびサービス](#)を参照してください。

## 準備の手順

このセクションでは、マルチノード OpenStack 構成を実装する前のいくつかの予備的な考慮事項について説明します。

### ホスト名、変数、およびパスワードの準備

マルチノード構成では、複数のネットワークインターフェースを使用して、クラウド用に作成した異なるサブネットを提供します。これらのインターフェース用のホスト名を準備していることを確認してください。これらの名前とその IP アドレスをシステムの `/etc/hosts` ファイルまたは DNS 構成に含めます。

たとえば、さまざまなタイプのネットワークトラフィックを処理するため、次のホスト名を作成できます。

- 管理トラフィックおよび API トラフィックをホストする OpenStack ネットワーク用の `host-on`。
- コンピュートノードと L3 ルーターの間のトラフィックをホストするプロジェクトネットワーク用の `host-tn`。
- 外部ネットワークトラフィック用の `host-en`。

異なるノード内の OpenStack サービスを構成する場合は、次の例のような、タスクを容易にするための変数を作成します。

- `$CONTROLLER_ADMIN_NODE` - OpenStack 管理サービスが接続されたコントローラノードのインターフェースまたは IP アドレスのホスト名。
- `$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。

構成プロセスではパスワードも必要です。準備する必要があるパスワードのサンプルリストを次に示します。

- MySQL データベースの root パスワード
- RabbitMQ のゲストユーザーのパスワード

- ユーザー admin のパスワード
- 次の OpenStack サービスのデータベースパスワード
  - アイデンティティサービス
  - イメージサービス
  - コンピュートサービス
  - ダッシュボードデータベース
  - ブロックストレージデータベース
  - ネットワークデータベース
  - オーケストレーションデータベース
- 次の OpenStack サービスユーザーのパスワード
  - glance
  - nova
  - cinder
  - neutron
  - heat

---

注記 - ユーザーまたはサービスのグループに共通のパスワードを割り当てることもできます。パスワードの割り当てにどのシステムを採用しているにかかわらず、環境をセキュリティー保護するためのベストプラクティスに必ず従うようにしてください。使用しているリリースの該当するライブラリにある『Oracle Solaris でのシステムおよび接続されたデバイスのセキュリティー保護』を参照してください。

---

## サンプルの Keystone スクリプト

Keystone データベースをすばやく移入するには、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、開始するのに役立つ次の基本的なタスクを実行します。

- 次の初期プロジェクトを作成します。
  - 基本的なサービスまたはコアサービスが作成される基となる service。
  - demo。この下で、ユーザー admin が `secrete` をデフォルトのパスワードとして使用して作成されます。
- Keystone データベースを移入します。
- 次のコアサービスを対応するユーザー名とパスワードで作成します。
  - cinder
  - ec2
  - glance
  - keystone

- neutron
- nova
- swift

対応するユーザー名とパスワードはサービス名と同じです。たとえば、cinder は Cinder サービスの、また ec2 は ec2 サービスのユーザー名とパスワードであり、その他も同じです。これらのパスワードは広く知られているため、セキュアではありません。本番設定では、カスタマイズされたパスワードを作成して、スクリプト内のこれらのデフォルトのパスワードを置き換えるようにしてください。必要に応じて、スクリプト内のすべてのサービスに単一のパスワードを設定できます。スクリプトを実行して Keystone をブートストラップする前に、必要なすべての変更をスクリプトに適用します。

---

**注記** - 環境に設定できるパラメータの詳細は、スクリプトを確認してください。設定に従って、スクリプト内のデフォルト設定を置き換えてください。

このドキュメント全体を通して、各手順ではパスワードを除き、サンプルデータスクリプトが改訂なしで使用され、スクリプトのすべてのデフォルト設定がクラウド構成に適用されることを前提にしています。

---

## 構成ファイルの編集について

OpenStack 構成の主な部分には、コンポーネントの構成ファイルの編集が必要になります。このドキュメントでは、各 \*.conf または \*.ini ファイル内の構成のために、選択されたパラメータのみが識別されます。これらの選択されたパラメータは、クラウド構成を機能させるために必要最小限のものです。ただし、特定のクラウド設定に関連するすべてのパラメータが正しく構成されていることを確認するには、各構成ファイルの内容全体を確認してください。

## メモリー使用の最適化

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。

別の OpenStack ノードでもこのパラメータを設定してください。

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。

## NTP サーバーの構成

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 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。

NTP を使用するには、NTP サーバーと NTP クライアントを構成する必要があります。通常、NTP サーバーは、OpenStack を構成しているほかのシステムとは別のシステムです。NTP クライアントは、OpenStack コンポーネントをホストするノードまたはシステム上でインストールおよび構成されます。

NTP の詳細は、<http://www.ntp.org/documentation.html> にあるドキュメントを参照してください。

### ▼ NTP サーバーを設定する方法

NTP サーバーは、OpenStack ノードとは別のシステム上にあります。

1. NTP パッケージをインストールします。

```
ntp-server# pkg install ntp
```

2. 構成ファイルをインストールします。

```
ntp-server# cp /etc/inet/ntp.server /etc/inet/ntp.conf
```

3. `server` および `driftfile` キーワードを構成することによって `/etc/inet/ntp.conf` ファイルを編集します。

例:

```
server 127.127.1.0 prefer
...
driftfile /var/ntp/ntp.drift
```

---

**注記** - 127.127.1.0 は IP アドレスではありません。これは、サーバーに正確な時間を提供するクロックを参照するために使用される形式です。server キーワードについて説明している `ntp.conf` ファイル内のコメントを必ず参照してください。

---

4. 前の手順で定義したように `/var/ntp/ntp.drift` ファイルを作成します。

```
ntp-server# touch /var/ntp/ntp.drift
```

5. ntp サービスを起動します。

```
ntp-server# svcadm enable ntp
```

## 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。

コントローラノードを構成するには、次のコマンドを使用して、システムに OpenStack コンポーネントおよびサービスをインストールします。

```
controller# pkg install openstack
```

パッケージのインストールが完了したら、ノード上で実行するサービスを構成します。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。

- [30 ページの「NTP クライアントの構成」](#)
- [31 ページの「MySQL のインストール」](#)。
- [32 ページの「Keystone のインストール」](#)。
- [33 ページの「Glance のインストール」](#)。
- [35 ページの「Nova のインストール」](#)。
- [36 ページの「Horizon のインストール」](#)。
- [37 ページの「Cinder のインストール」](#)。
- [38 ページの「Neutron のインストール」](#)。
- [96 ページの「Heat のインストール」](#)。

## NTP クライアントの構成

NTP クライアントサービスは、クラウド配備内の各サービスノードにインストールします。

### ▼ NTP クライアントを構成する方法

このセクションでは、[29 ページの「NTP サーバーを設定する方法」](#)の説明に従って NTP サーバーをすでに設定していることを前提にしています。

1. クライアント構成ファイルを作成します。

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

2. クライアント構成ファイルで、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、NTP サーバーの特定の名前または IP アドレスを指定します。

たとえば、構成した NTP サーバーのホスト名が system1 である場合、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 server_name2 iburst
# server server_name3 iburst
```

3. ntp 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、特に本番環境では、この情報を格納するために MySQL データベースなどのその他のデータベースをお勧めします。

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. RabbitMQ サービスを有効にします。

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

2. (オプション) 管理および API トラフィックに専用の 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
```

3. MySQL サービスを有効にします。

```
controller# svcadm enable mysql
```

#### 4. 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
```

#### 5. 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 default character set utf8 default collate utf8_general_ci;
mysql> grant all privileges on cinder.* to 'cinder'@$CONTROLLER_ADMIN_NODE identified by
'service-password';
mysql> create database glance default character set utf8 default collate utf8_general_ci;
mysql> grant all privileges on glance.* to 'glance'@$CONTROLLER_ADMIN_NODE identified by
'service-password';
mysql> create database keystone default character set utf8 default collate utf8_general_ci;
mysql> grant all privileges on keystone.* to 'keystone'@$CONTROLLER_ADMIN_NODE identified by
'service-password';
mysql> create database nova default character set utf8 default collate utf8_general_ci;
mysql> grant all privileges on nova.* to 'nova'@$CONTROLLER_ADMIN_NODE identified by
'service-password';
mysql> create database neutron default character set utf8 default collate utf8_general_ci;
mysql> grant all privileges on neutron.* to 'neutron'@$CONTROLLER_ADMIN_NODE identified by
'service-password';
mysql> create database heat default character set utf8 default collate utf8_general_ci;
mysql> grant all privileges on heat.* to 'heat'@$CONTROLLER_ADMIN_NODE 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ードにインストールし、構成してください。この手順では、[27 ページの「サンプルの Keystone スクリプト」](#)で説明されているサンプルスクリプトを使用します。スクリプトを使用する前に、そのセクションを参照してください。

### ▼ Keystone をインストールし、構成する方法

1. 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。openssl コマンドから、鍵を構成するコンポーネント (国や州など) の入力を求められます。

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

2. トークンをシェル変数に設定します。

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

3. /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
```

4. Keystone SMF サービスを有効にします。

```
controller# svcadm enable keystone
```

5. Keystone サンプルスクリプトを使用して Keystone データベースを移入します。  
スクリプトを実行する前に、そのスクリプトを確認し、設定に従って変更したことを確認してください。この手順では、サンプルスクリプトがカスタマイズされていないことを前提にしています。

```
controller# CONTROLLER_PUBLIC_ADDRESS=$CONTROLLER_ADMIN_NODE \
CONTROLLER_ADMIN_ADDRESS=$CONTROLLER_ADMIN_NODE \
CONTROLLER_INTERNAL_ADDRESS=$CONTROLLER_ADMIN_NODE \
SERVICE_TOKEN=$MY_SERVICE_TOKEN \
ADMIN_PASSWORD=admin-password \
SERVICE_PASSWORD=service-password \
/usr/demo/openstack/keystone/sample_data.sh
```

## Glance のインストール

Glance を設定するには、認証に関する情報の構成、および MySQL および RabbitMQ サービスの場所の指定が必要です。

### ▼ Glance をインストールし、構成する方法

1. これらの構成ファイルでパラメータをコメント解除または設定することで、Glance を構成します。

■ /etc/glance/glance-api.conf

```
[DEFAULT]
registry_host = $CONTROLLER_ADMIN_NODE

auth_strategy = keystone
default_publisher_id = image.$CONTROLLER_ADMIN_NODE
rabbit_host = $CONTROLLER_ADMIN_NODE
qpuid_hostname = $CONTROLLER_ADMIN_NODE

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

[keystone_auth token]
auth_uri = http://$CONTROLLER_ADMIN_NODE:5000/v2.0
identity_uri = http://$CONTROLLER_ADMIN_NODE:35357
admin_tenant_name = service
admin_user = glance
admin_password = service-password
```

■ /etc/glance/glance-cache.conf

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

■ /etc/glance/glance-registry.conf

```
[DEFAULT]
default_publisher_id = image.$CONTROLLER_ADMIN_NODE
rabbit_host = $CONTROLLER_ADMIN_NODE
qpuid_hostname = $CONTROLLER_ADMIN_NODE

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

[keystone_auth token]
auth_uri = http://$CONTROLLER_ADMIN_NODE:5000/v2.0
identity_uri = http://$CONTROLLER_ADMIN_NODE:35357
admin_tenant_name = service
admin_user = glance
admin_password = service-password
```

■ /etc/glance/glance-scrubber.conf

```
[DEFAULT]
```

```

auth_url = http://$CONTROLLER_ADMIN_NODE:5000/v2.0/
identity_uri = http://$CONTROLLER_ADMIN_NODE:35357
admin_tenant_name = service
admin_user = glance
admin_password = service-password

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

```

## 2. 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

このセクションは、コンピュータ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. `/etc/nova/nova.conf` ファイルでパラメータをコメント解除または設定することで、Nova を構成します。

```

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

[database]
connection = mysql://nova:service-password@$CONTROLLER_ADMIN_NODE/nova

[glance]
host=$CONTROLLER_ADMIN_NODE

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

[neutron]
url=http://$CONTROLLER_ADMIN_NODE:9696
admin_username=neutron

```

```
admin_password=service-password
admin_tenant_name=service
admin_auth_url=http://$CONTROLLER_ADMIN_NODE:5000/v2.0
```

2. `/etc/nova/api-paste.ini` ファイル内のパラメータを設定します。

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

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。

### ▼ Horizon を構成する方法

1. 構成に適用される内容に応じて、次のいずれかの手順を実行します。

- 構成は HTTP を使用します。

- a. `/etc/openstack_dashboard/local_settings.py` スクリプトに設定を指定します。

```
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
```

- b. OpenStack 用 `http.conf` ファイルの HTTP バージョンをコピーします。

```
controller# cp /etc/apache2/2.4/samples-conf.d/openstack-dashboard-http.conf \
/etc/apache2/2.4/conf.d/
```

- 構成は SSL/TLS を使用します。

- a. 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 0644 ${DASHBOARD}/*
controller# chown webservd:webservd ${DASHBOARD}/*

controller# sed \
-e "/SSLCertificateFile/s:/path.*:${DASHBOARD}/horizon.crt:" \
-e "/SSLCertificateFile/d" \
-e "/SSLCertificateKeyFile/s:/path.*:${DASHBOARD}/horizon.key:" \
< /etc/apache2/2.4/samples-conf.d/openstack-dashboard-tls.conf \
> /etc/apache2/2.4/conf.d/openstack-dashboard-tls.conf
```

- b. `~/conf.d/openstack-dashboard-tls.conf` ファイルの次のパラメータで、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 /horizon https://controller-fqdn/horizon
ServerName controller-fqdn
```

2. Apache サービスを起動します。

```
controller# svcadm enable apache24
```

## 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では、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参照します。

1. `/etc/cinder/cinder.conf` ファイルでパラメータをコメント解除または設定することで、Cinder を構成します。

`volume_driver` パラメータでは、複数のドライバを選択できます。次の例では、`volume_driver` に選択されたドライバのみが表示されています。コメントアウトされているその他の使用可能なドライバは除外されています。

```
[DEFAULT]
volume_driver=cinder.volume.drivers.solaris.zfs.ZFSISCSIDriver
my_ip=${CONTROLLER_ADMIN_NODE}

[database]
connection = mysql://cinder:service-password@${CONTROLLER_ADMIN_NODE}/cinder

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

2. `/etc/cinder/api-paste.ini` ファイル内のパラメータを構成します。

```
[filter:authtoken]
admin_tenant_name = service
admin_user = cinder
admin_password = service-password
```

3. iSCSI ターゲットが構成されている場合は、対応する SMF サービスを有効にします。

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

4. 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。

## Neutron のインストール

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. これらの構成ファイルでパラメータをコメント解除または設定することで、Neutron を構成します。

```
■ /etc/neutron/neutron.conf

qpid_hostname=${CONTROLLER_ADMIN_NODE}
```

```

rabbit_host=/${CONTROLLER_ADMIN_NODE}

host=${CONTROLLER_ADMIN_NODE}

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

[database]
connection = mysql://neutron:service-password@${CONTROLLER_ADMIN_NODE}/neutron

```

- /etc/neutron/plugins/evs/evs\_plugin.ini

```

[EVS]
evs_controller = ssh://evsuser@${CONTROLLER_ADMIN_NODE}

```

- /etc/neutron/dhcp\_agent.ini

```

[DEFAULT]
evs_controller = ssh://evsuser@${CONTROLLER_ADMIN_NODE}

```

- /etc/neutron/l3\_agent.ini

```

evs_controller = ssh://evsuser@${CONTROLLER_ADMIN_NODE}

```

## 2. 使用される SSH 鍵ペアを設定します。

### a. 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

```

### b. evsuser の authorized\_keys ファイルで、evsuser、neutron、および root ユーザーの SSH 鍵を結合します。

```

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

```

### c. SSH 接続をテストして、known\_host ファイルに格納されるフィンガープリントを受け入れます。

確認のプロンプトごとに Yes を指定します。

```

controller# su - evsuser -c "ssh evsuser@${CONTROLLER_ADMIN_NODE} true"
controller# su - neutron -c "ssh evsuser@${CONTROLLER_ADMIN_NODE} true"

```

```
controller# ssh evsuser@$CONTROLLER_ADMIN_NODE true
```

### 3. エラスティック仮想スイッチ (EVS) を構成します。

**注記** - 次のサブステップでは、具体的には VLAN ベースのネットワーク用の EVS を構成します。

VXLAN ベースのネットワークを構成するには、[オペレーティングシステムのドキュメント](#)内の使用している Oracle Solaris バージョンのライブラリにある『Oracle Solaris での仮想ネットワークとネットワークリソースの管理』に移動します。そのドキュメントの中の、特に「ユースケース: テナント用のエラスティック仮想スイッチの構成」のセクションを参照してください。

フラットなネットワークを構成する方法の例については、[https://blogs.oracle.com/openstack/entry/configuring\\_the\\_neutron\\_l3\\_agent](https://blogs.oracle.com/openstack/entry/configuring_the_neutron_l3_agent) を参照してください。

#### a. EVS プロパティを設定して、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_ADMIN_NODE
```

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

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

[図2](#)に示すように、通常は、異なるサブネットを提供するために複数のネットワークインタフェースを使用します。uplink-port プロパティを設定する場合は、サブネットにサービスを提供する複数のネットワークポートにわたって VLAN を分割できます。

次の例は、EVS プロパティを設定する方法 (VLAN の分割を含む) を示しています。オプションで、最後のコマンドを使用して、すべての EVS プロパティを表示します。

**注記** - VLAN をネットワークポートに分散させる前に、まず VLAN 範囲を定義します。そうしないと、uplink-port プロパティを構成できません。

```
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,vlan-range=1
controller# evsadm set-controlprop -p uplink-port=net1,vlan-range=200-250
controller# evsadm set-controlprop -p uplink-port=net1,vlan-range=251-300
```

```
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起動します。

```
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
```

## コンピュータノードの構成

コンピュータノードには、VM インスタンスだけでなく nova-compute デーモンも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広範なサービスを提供します。コンピュータノードは、クラウドに必要な数だけ構成できます。

コンピュータノードを構成するには、次のコマンドを使用して、システムに OpenStack コンポーネントおよびサービスをインストールします。

```
compute# pkg install openstack
```

パッケージのインストールが完了したら、ノード上で実行するサービスを構成します。

---

**注記** - 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` はランダムな識別子 (会社の名前など) です。

別の OpenStack ノードでもこのパラメータを設定してください。

このパラメータの詳細については、<https://support.oracle.com> で MOS アカウントにログインし、*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。

---

## ▼ コンピュータノードを構成する方法

### 1. NTP クライアントを構成します。

[30 ページの「NTP クライアントの構成」](#)を参照してください。

### 2. 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
```

### 3. `/etc/nova/nova.conf` ファイル内の次のパラメータをコメント解除または設定することによって Nova を構成します。

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

[database]
connection = mysql://nova:service-password@$CONTROLLER_ADMIN_NODE/nova

[glance]
host=$CONTROLLER_ADMIN_NODE

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

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

4. `/etc/nova/api-paste.ini` ファイル内のパラメータを設定します。

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

5. コンピュータノード上で EVS を設定します。

a. EVS パッケージがインストールされていることを確認します。

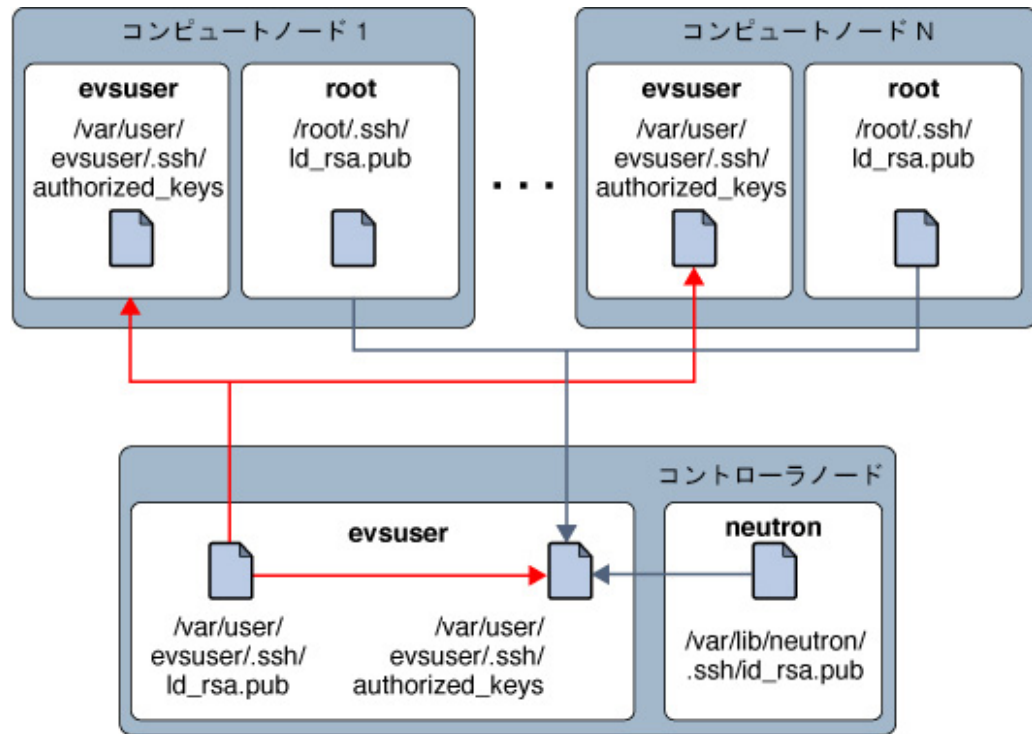
```
compute1# pkg info evs
```

b. 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_ADMIN_NODE
```

6. コントローラとコンピュータノードの間の通信を構成します。

ノード間の通信を確立する SSH 鍵の配布は、次の図のようになります。



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

```
compute1# 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) コンピュータノードの 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
```

この出力には、コンピュータノード上で生成した `/root/.ssh/id_rsa.pub` の内容が含まれるはずですが、

- f. コントローラへのコンピュータノードの 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。

確認のプロンプトで `Yes` を指定します。

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

7. コンピュータノードに Nova サービスアクセスを付与します。

- a. コントローラノードにアクセスします。

- b. 次のコマンドを入力します。

```
controller# mysql -u root -p
Enter password: MySQL-root-password
mysql> grant all privileges on nova.* to 'nova'@'nova-compute' identified by 'service-
password';
mysql> flush privileges
mysql> quit
```

ここで、`nova-compute` はコンピュータノードのシステム名または完全修飾ドメイン名です。

8. コンピュータノード上で、Nova コンピュータサービスを有効にします。

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

## ▼ コンソールアクセスを有効にする方法

ユーザーの要求に基づいて、ブラウザから VM インスタンスのコンソールを使用できるようにするには、この手順を使用します。

1. 各コンピュータノード上で、どのシナリオが適用されるかに応じて、次の手順を実行します。

- コンピュータノードの IP アドレスがパブリックに面したネットワークからアクセスできる場合は、`/etc/nova/nova.conf` ファイルの [DEFAULT] セクション内の次のパラメータを設定します。

```
[DEFAULT]
...
vnc_enabled = true
vncserver_listen = 0.0.0.0
novncproxy_port = 6080
novncproxy_base_url =http://FQDN:6080/vnc_auto.html
```

```
novncproxy_host = 0.0.0.0
...
```

ここで、*FQDN* は、そのコンピュータノードの完全修飾ドメイン名または IP アドレスを表します。

- コンピュータノードがプライベートネットワーク内に存在する場合は、`/etc/nova/nova.conf` ファイルの [DEFAULT] セクション内の次のパラメータを設定します。

```
[DEFAULT]
...
vnc_enabled = true
vncserver_listen = internal-IP
novncproxy_port=6080
novncproxy_base_url = http://public-IP:6080/vnc_auto.html
vncserver_proxyclient_address = internal-IP
```

- *internal-IP* - 内部ネットワーク上のコンピュータノードの IP アドレス。
- *public-IP* - コントローラホストのパブリック IP アドレス。

2. コンピュータノードの IP アドレスがパブリックに面したネットワークからアクセスできる場合は、次のサブステップを実行します。それ以外の場合は、次の手順にスキップします。

- a. `nova-novncproxy` サービスを有効にします。

```
compute# svcadm enable nova-novncproxy
```

- b. `nova-compute` サービスを再起動します。

```
compute# svcadm restart nova-compute
```

3. コントローラノード上で、どのシナリオが適用されるかに応じて、次の手順を実行します。

- コンピュータノードの IP アドレスがパブリックに面したネットワークからアクセスできる場合は、`nova-consoleauth` サービスを有効にします。

```
controller# svcadm enable nova-consoleauth
```

- コンピュータノードがプライベートネットワーク内に存在する場合は、次の手順を実行します。

- a. `/etc/nova/nova.conf` ファイルの [DEFAULT] セクション内の次のパラメータを設定します。

```
novncproxy_base_url=http://public-IP:6080/vnc_auto.html
```

ここで、*public-IP* はコントローラホストのパブリック IP アドレスです。

- b. 次のように `Nova` サービスを有効にします。

```
controller# svcadm enable nova-consoleauth
controller# svcadm enable nova-novncproxy
```

## ストレージノードの構成

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

コンピュータノードを構成するには、次のコマンドを使用して、システムに OpenStack コンポーネントおよびサービスをインストールします。

```
storage# pkg install openstack
```

パッケージのインストールが完了したら、ノード上で実行するサービスを構成します。

---

**注記** - 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* はランダムな識別子 (会社の名前など) です。

別の OpenStack ノードでもこのパラメータを設定してください。

このパラメータの詳細については、<https://support.oracle.com> で MOS アカウントにログインし、*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

この手順では、ブロックストレージの標準的な構成について説明します。ストレージコンポーネントを構成するためのその他のオプションについては、[第6章「Cinder の構成と配備のためのオプション」](#)を参照してください。

1. **NTP クライアントを構成します。**  
[30 ページの「NTP クライアントの構成」](#)を参照してください。
2. `/etc/cinder/cinder.conf` ファイルでパラメータをコメント解除または設定することで、**Cinder を構成します。**

[DEFAULT]

```

san_is_local=true
volume_driver=cinder.volume.drivers.solaris.zfs.ZFSISCSIDriver
my_ip=$VOLUME_IP
rabbit_host=$CONTROLLER_ADMIN_NODE
glance_host=$CONTROLLER_ADMIN_NODE
zfs_volume_base=cinder/cinder

[database]
connection = mysql://cinder:service-password@$CONTROLLER_ADMIN_NODE/cinder

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

```

3. `/etc/cinder/api-paste.ini` ファイル内のパラメータを設定します。

```

[filter:authtoken]
admin_tenant_name = service
admin_user = cinder
admin_password = service-password

```

4. ブロックストレージまたはボリュームノードに Cinder サービスアクセスを付与します。

- a. コントローラノードにアクセスします。

- b. 次のコマンドを入力します。

```

controller# mysql -u root -p
Enter password: MySQL-root-password
mysql> grant all privileges on cinder.* to 'cinder'@$VOLUME_IP identified by 'service-
password';
mysql> flush privileges
mysql> quit

```

5. ストレージノード上で、Cinder サービスを起動します。

```

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

```

## Swift オブジェクトストレージの構成

Swift は、OpenStack オブジェクトストアプロジェクトです。大量のデータを単純な API で格納したり、取得したりできるようにするためのクラウドストレージソフトウェアを提供します。このサービスは、無制限に拡張できる非構造化データの格納に最適です。

Swift の詳細は、OpenStack コミュニティの [OpenStack クラウド管理者ガイド](#) のオブジェクトストレージに関する章を参照してください。

OpenStack コミュニティのドキュメントでは、本番モードでの Swift 配備には少なくとも 6 ノードを推奨しています。これらのノードは、Swift プロキシコントローラと 5 つの Swift ストレージノードで構成されます。ただし、このガイドでは、前の構成手順の基になる 3 ノードアーキテクチャーとの整合性がある 3 ノード配備について説明します。必要に応じて、あとでストレージノードを追加できます。

## ▼ Swift プロキシコントローラサービスノードを構成する方法

このタスクでは、Swift 用に指定されたノードに OpenStack パッケージをすでにインストールしていることを前提にしています。使用するインストールコマンドは、[46 ページの「ストレージノードの構成」](#)で参照してください。

1. Swift パッケージをインストールします。

```
proxy-node # pkg install swift swiftclient
```

2. ZFS データセットを作成します。

```
proxy-node # /usr/sbin/zfs create -o mountpoint=none rpool/export/swift
proxy-node # /usr/sbin/zfs create -o mountpoint=/srv rpool/export/swift/srv
proxy-node # /usr/sbin/zfs create -p rpool/export/swift/srv/node/disk0
proxy-node # /usr/bin/chown -R swift:swift /srv
```

3. 次の 8 進ダンプを実行します。

これらのダンプの値を保持します。これらの 2 つの値は、以降の手順では \$OD\_1 および \$OPD\_2 と呼ばれます。

```
proxy-node # od -t x8 -N 8 -A n < /dev/random
proxy-node # od -t x8 -N 8 -A n < /dev/random
```

4. 次のパラメータを使用して /etc/swift/swift.conf ファイルを編集します。

```
[swift-hash]
swift_hash_path_suffix = $OD_1
swift_hash_path_prefix = $OD_2
```

5. 次のパラメータを使用して /etc/swift/proxy-server.conf ファイルを編集します。

```
[DEFAULT]
bind_port = 8080

[filter:tempauth]
use = egg:swift#tempauth

operator_roles = admin, swiftoperator
```



```
[filter:authtoken]
auth_uri = http://$CONTROLLER_IP:5000/
identity_uri = http://$CONTROLLER_IP:35357
admin_tenant_name = service
admin_user = swift
admin_password = swiftpass
```

```
[filter:cache]
memcache_servers = $CONTROLLER_IP:11211
```

## 6. memcached デーモンを有効にします。

```
proxy-node # svcadm enable -rs memcached
```

## 7. リングを構築します。

```
proxy-node # cd /etc/swift
proxy-node # swift-ring-builder account.builder create 18 3 1
proxy-node # swift-ring-builder container.builder create 18 3 1
proxy-node # swift-ring-builder object.builder create 18 3 1
proxy-node # swift-ring-builder account.builder add r1z1-$STORAGE_IP_1:6002/disk0 100
proxy-node # swift-ring-builder container.builder add r1z1-$STORAGE_IP_1:6001/disk0 100
proxy-node # swift-ring-builder object.builder add r1z1-$STORAGE_IP_1:6000/disk0 100
proxy-node # swift-ring-builder account.builder add r1z1-$STORAGE_IP_2:6002/disk0 100
proxy-node # swift-ring-builder container.builder add r1z1-$STORAGE_IP_2:6001/disk0 100
proxy-node # swift-ring-builder object.builder add r1z1-$STORAGE_IP_2:6000/disk0 100
proxy-node # swift-ring-builder account.builder rebalance
proxy-node # swift-ring-builder container.builder rebalance
proxy-node # swift-ring-builder object.builder rebalance
proxy-node # chown -R swift:swift /etc/swift
```

## 8. Swift サービスを有効にします。

```
proxy-node # svcadm enable swift-proxy-server
```

# ▼ オブジェクトストレージノードを構成する方法

設定する各オブジェクトストレージノード上でこの手順を繰り返します。

## 1. Swift パッケージをインストールします。

```
storage-node # pkg install swift swiftclient
```

## 2. ZFS データセットを作成します。

```
storage-node # /usr/sbin/zfs create -o mountpoint=none rpool/export/swift
storage-node # /usr/sbin/zfs create -o mountpoint=svr rpool/export/swift/srv
storage-node # /usr/sbin/zfs create -p rpool/export/swift/srv/node/disk0
storage-node # /usr/bin/chown -R swift:swift /srv
```

3. 次のように、プロキシサーバーノードからファイルをコピーします。
  - a. プロキシサーバーノードの `/etc/swift/swift.conf` ファイルを現在のノードの `/etc/swift` ディレクトリにコピーします。
  - b. プロキシサーバーノードの次のファイルを現在のノードの `/etc/swift` ディレクトリにコピーします。

- `account.ring.gz`
- `container.ring.gz`
- `object.ring.gz`

4. Swift レプリケーターサービスを有効にします。

```
storage-node # svcadm enable swift-replicator-rsync
```

5. 現在のノードの `/etc/swift` ディレクトリの所有権を設定します。

```
storage-node # chown -R swift:swift /etc/swift
```

6. すべての Swift サービスを有効にします。

```
storage-node # for x in `svcs -a -o SVC | fgrep swift | \
egrep "account|container|object" | sort` \
do \
    echo Starting $x \
    svcadm enable $x \
done
```

7. コントローラノード上で、ユーザーが Swift サービスにアクセスして操作できるようにします。

- a. Swift のグローバルシェル変数を設定します。

```
controller# export OS_USERNAME=swift
controller# export OS_PASSWORD=service-password
controller# export OS_TENANT_NAME=service
controller# export OS_AUTH_URL=http://$CONTROLLER_ADMIN_NODE:5000/v2.0
```

- b. Keystone `swiftoperator` 役割を追加します。

```
controller# keystone role-create --name swiftoperator
```

- c. `swiftoperator` 役割を Swift サービスの承認されたユーザーに割り当てます。

```
controller# keystone user-role-add --user user-name \
--role swiftoperator --tenant tenant-id
```

次の手順 ユーザーがクラウドを使用できるように、[第4章「インストール後タスクおよび構成タスク」](#)で説明されている準備作業を完了します。

# ◆◆◆ 第 4 章

## インストール後タスクおよび構成タスク

---

この章では、初期の OpenStack インストールおよび構成を完了する手順について説明します。この章の内容は次のとおりです。

- 51 ページの「OpenStack プロジェクト用の外部ネットワークの準備」
- 57 ページの「Glance リポジトリ用のイメージの準備」

### OpenStack プロジェクト用の外部ネットワークの準備

外部ネットワークは、クラウド内のプライベートネットワークとパブリックネットワークの間の接続を提供します。

#### プロバイダルーターについて

ルーターは、プロジェクト VM インスタンスに、より広いネットワークとの接続を提供します。ルーターは、すべてのプロジェクトネットワークによって共有されます。ルーターは 1 つしかないため、プロジェクトネットワークは重複した IP アドレスを使用できません。

ルーターは、そのルーターを外部ネットワークに接続するインタフェース上で双方向のネットワークアドレス変換 (NAT) を実行します。プロジェクトには必要なだけ、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ている数だけのフローティング IP (パブリック IP) を割り当てることができます。これらのフローティング IP は、外部接続を必要としている VM インスタンスに関連付けられています。

ルーターを作成するには、Neutron L3 エージェントを構成する必要があります。このエージェントは、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 アドレスとの間の 1 対 1 NAT マッピングを自動的に作成します。L3 エージェントは、プライベートネットワーク間の通信も可能にします。

デフォルトでは、同じプロジェクトの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` 構成ファイル内の

`allow_forwarding_between_networks` を True に設定します。このパラメータを設定したあと、`neutron-l3-agent` SMF サービスを再起動します。

## ▼ 外部ネットワーク用にルーターを構成する方法

この手順では、外部ネットワーク用にルーターを作成する方法を示します。この手順の一部には、構成ファイルの編集が必要です。そのため、この手順では Horizon ダッシュボードより端末ウィンドウを使用した方が便利です。

Neutron サービスがインストールされているノード上で、次の手順を実行します。このドキュメントでは、前の章で説明されているサンプルアーキテクチャーに基づいて、コントローラノード上でこのサービスを見つけます。

始める前に [38 ページの「Neutron をインストールし、構成する方法」](#)の説明に従って Neutron の構成を完了していることを確認してください。

1. Solaris IP フィルタが無効になっている場合は有効にします。

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

2. ホスト上で IP 転送が無効になっている場合は有効にします。

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

3. Neutron のグローバルシェル変数を設定します。

```
controller# export OS_USERNAME=neutron
controller# export OS_PASSWORD=service-password
controller# export OS_TENANT_NAME=service
controller# export OS_AUTH_URL=http://$CONTROLLER_ADMIN_NODE: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. `/etc/neutron/l3_agent.ini` ファイルで、`router_id` の値を前の手順のルーター ID に設定します。

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

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

7. (オプション) ルーターに関する情報を表示します。

ルーターに外部ネットワークを追加すると、そのルーターに関する詳細情報が追加されます。

```
controller# neutron router-show router-name
```

#### 例 1 ルーターの作成

この例では、外部ネットワーク用にルーターを作成する方法を示します。

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

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

```
controller# export OS_USERNAME=neutron
```

```
controller# export OS_PASSWORD=service-password
```

```
controller# export OS_TENANT_NAME=service
```

```
controller# export OS_AUTH_URL=http://$CONTROLLER_ADMIN_NODE:5000/v2.0
```

```
controller# neutron router-create ext-router
```

Created a new router:

```
+-----+
| Field          | Value                                |
+-----+-----+
| admin_state_up | True                                 |
| external_gateway_info |                                     |
| id              | f89b24ed-42dd-48b0-8f4b-fd41887a3370 |
| name           | ext-router                           |
| status         | ACTIVE                               |
| tenant_id      | 7d1caf0854b24becb28df5c5cabf72cc    |
+-----+-----+
```

この時点で、`/etc/neutron/l3_agent.ini` ファイル内の `router_id` を更新します。

```
router_id = f89b24ed-42dd-48b0-8f4b-fd41887a3370
```

次に、L3 エージェントサービスを有効にします。

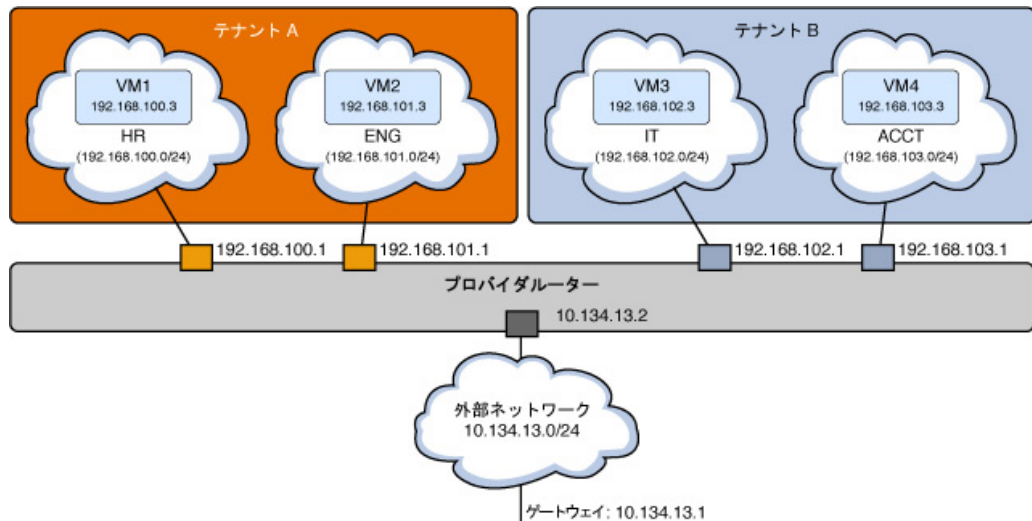
```
controller# svcadm enable neutron-l3-agent
```

## 外部ネットワークの作成

ルーターを作成したあと、次は外部ネットワークを構成します。クラウドの内部ネットワーク内のユーザーは、外部ネットワークのゲートウェイ経由でインターネットに接続できます。

次の図は、単一ルーターがクラウドプロジェクトのネットワーク通信をサポートする方法を示しています。

図 4 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



この図は、次を示しています。

- 2つのプロジェクト (Tenant A と Tenant B)
- 4つの VM (VM1、VM2、VM3、および VM4)
- 4つのサブネット (HR、ENG、IT、および ACCT)
- ルーター
- 外部ネットワーク

外部ネットワークへの内部ネットワークアクセスを提供すると、VM の IP アドレスが、その外部ネットワークに割り当てられるフローティングアドレスのいずれかにマップされます。

## ▼ 外部ネットワークを作成する方法

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は DHCP を使用しません。代わりに、フローティング IP アドレスが作成されてプロジェクトに割り当てられ、それらのプロジェクトの下で Nova VM インスタンスによって使用されます。これらの手順では VLAN タイプのネットワークを作成しますが、この手順は (フラットなネットワークなどの) その他のネットワークタイプの作成にも適用されます。

外部ネットワークは、内部ネットワークの作成に関係なく作成できます。

**始める前に** エラスティック仮想スイッチの構成を完了してください。詳細は、[38 ページの「Neutron をインストールし、構成する方法」](#) (特に、EVS を構成する手順) を参照してください。

## 1. Neutron のグローバルシェル変数を設定します。

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

## 2. (オプション) VLAN 範囲を取得します。

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

## 3. 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 の値は通常、EVS の構成中に定義した VLAN 範囲の最初の番号である 1 です。

---

**注記** - フラットなネットワークを作成している場合は、セグメンテーション ID を指定する必要はありません。

---

## 4. 外部ネットワークのサブネットを作成します。

割り当てプールは、サブネットに割り当てられたフローティング IP アドレスの範囲で構成されます。

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

## 5. 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-name ext-network-id
```

---

**注記** - デフォルトでは、このコマンドを発行するときに SNAT は有効になっています。SNAT が有効になっていると、プライベートネットワーク内の VM は外部ネットワークにアクセスできます。ただし、これらのインスタンス自体にクラウドの外部からアクセスすることはできません。SNAT を無効にするには、neutron router-gateway-set サブコマンドで --disable-snat オプションを指定します。

---

## 6. (オプション) ルーターに関する情報を表示します。

```
controller# neutron router-show router-name
```

## 例 2 外部ネットワークの作成

この例では、外部ネットワークを作成して、それをクラウド内の内部ネットワークで使用できるように準備する方法を示します。

フラットなネットワークを作成するには、<https://blogs.oracle.com/openstack/tags/juno> のセクション 2 で提供されている例も参照してください。

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

```
controller# evsadm show-controlprop -p vlan-range
PROPERTY          PERM VALUE          DEFAULT  HOST
vlan-range        rw  1,200-300        --        --
```

```
controller# neutron net-create --router:external=True \
--provider:network_type=vlan --provider:segmentation_id=1 ext-network
Created a new network:
```

```
+-----+
| Field          | Value |
+-----+
| admin_state_up | True  |
| id             | 08cf49c8-f28f-49c1-933d-bdb1017e0294 |
| name           | ext-network |
| provider:network_type | vlan |
| provider:segmentation_id | 1 |
| router:external | True |
| shared         | False |
| status         | ACTIVE |
| subnets       | |
| tenant_id      | 7d1caf0854b24becb28df5c5cabf72cc |
+-----+
```

```
controller# neutron subnet-create --name ext-subnet --disable-dhcp \
--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              | fce503ff-f483-4024-b122-f2524e3edae1 |
| ip_version      | 4 |
| ipv6_address_mode | |
| ipv6_ra_mode    | |
| name            | ext-subnet |
| network_id      | 08cf49c8-f28f-49c1-933d-bdb1017e0294 |
| tenant_id      | 7d1caf0854b24becb28df5c5cabf72cc |
+-----+
```

```
controller# neutron router-gateway-set ext-router 08cf49c8-f28f-49c1-933d-bdb1017e0294
Set gateway for router ext-router
```



```

controller# neutron router-show ext-router
+-----+-----+
| Field          | Value                                     |
+-----+-----+
| admin_state_up | True                                     |
| external_gateway_info | {"network_id": "08cf49c8-f28f-49c1-933d-bdb1017e0294", |
|                  | "enable_snat": true,                   |
|                  | "external_fixed_ips":                  |
|                  | [{"subnet_id": "fce503ff-f483-4024-b122-f2524e3edae1", |
|                  | "ip_address": "10.134.13.8"}]} |
| id             | f89b24ed-42dd-48b0-8f4b-fd41887a3370   |
| name           | ext-router                             |
| status         | ACTIVE                                  |
| tenant_id     | 7d1caf0854b24becb28df5c5cabf72cc      |
+-----+-----+

```

参照 ■ 114 ページの「既知の制限事項」。

## Glance リポジトリ用のイメージの準備

イメージは、クラウド内の VM インスタンスの基盤です。イメージは、ブート可能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ファイルです。イメージは、1 つまたは複数の VM を作成するためのテンプレートを提供します。そのため、クラウド上で VM をプロビジョニングするには、まずイメージを作成する必要があります。

OpenStack 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。

複数のイメージを Glance リポジトリにアップロードできます。ベストプラクティスとして、クラウド上に配備するさまざまなシステムタイプのイメージをアップロードしてください。たとえば、非大域ゾーン、カーネルゾーン、および大域ゾーンのアーカイブされたイメージを作成します。次に、適切なテンプレートを選択することによって、これらのいずれかのタイプの VM をすばやく配備できます。

---

**注記** - 第2章「[シングルノードへの OpenStack のインストール](#)」に従った場合は、この手順によって 2 つのイメージが自動的に作成されます。この現在の章では、事前に構成されたイメージが存在しないマルチノード構成を前提にしています。

---

## イメージの作成

Oracle Solaris で OpenStack イメージを作成するには、統合アーカイブ機能を使用します。archiveadm コマンドを使用すると、大域、非大域、およびカーネルゾーンから新しい統合アーカイブ (UA) を作成できます。

UA は、クローンアーカイブまたは回復用アーカイブのどちらかです。クローンアーカイブは、現在アクティブなブート環境に基づいています。このアーカイブには、OS インスタンス (非アクティブな BE など) のシステム構成情報は含まれません。代わりに、インストーラは再構成を強制するか、またはシステム構成 (SC) プロファイルで提供する構成情報を使用します。回復用アーカイブには、すべてのブート環境とシステム構成情報が含まれています。そのため、UA にシステムの情報をすべて含める場合は、回復用アーカイブを作成します。UA の詳細は、使用している Oracle Solaris バージョンの [ライブラリ](#) にある該当する『Oracle Solaris でのシステム復旧とクローン』ドキュメントを参照してください。

あとで、構成が完全に動作状態になったら、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りイメージを作成することもできます。この場合、VM インスタンスはすでにクラウド内に存在します。そのため、使用するコマンドは archiveadm ではなく、nova image-create です。nova コマンドは、実行中 VM インスタンスのスナップショットを取ってイメージを作成します。

データバックアップのため、または VM インスタンスをレスキューする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。

Oracle Solaris では、次の 3 つのフェーズで OpenStack イメージを作成します。

1. ゾーンを作成します。
2. ゾーンの UA を作成します。
3. UA を Glance にアップロードします。

次の手順では、これらのフェーズが結合されています。

### ▼ OpenStack 用にイメージを作成する方法

ゾーンを作成するためのコマンド構文を除き、この手順の残りはすべて、非大域ゾーンとカーネルゾーンの両方のイメージの作成とアップロードに使用できます。

ゾーンを作成する手順では、基本的なコマンドのみを提供します。ゾーンを作成する詳細な手順については、使用している Oracle Solaris バージョンの [ライブラリ](#) にあるゾーンのドキュメントを参照してください。

1. **任意のシステム上で、ゾーンを作成してから、そのゾーンにログインします。**  
ログインしたあと、入力を要求されたら情報を指定します。

```
global# zonecfg -z zone-name create
global# zoneadm -z zone-name install
global# zoneadm -z zone-name boot
global# zlogin -C zone-name
```

---

**注記** - この手順は、完了するまでにしばらく時間がかかることがあります。

---

## 2. OpenStack root ログインアクセス用にルート SSH を有効にします。

```
global# zlogin zone-name
root@zone-name# sed /^PermitRootLogin/s/no$/without-password/ < /etc/ssh/sshd_config
> /system/volatile/sed.$$
root@zone-name# cp /etc/ssh/sshd_config /etc/ssh/sshd_config.orig
root@zone-name# cp /system/volatile/sed.$$ /etc/ssh/sshd_config
root@zone-name# exit
```

## 3. ゾーンの UA を作成します。

```
global# archiveadm create -z zone-name /var/tmp/archive-name.uar
```

## 4. UA を Glance がインストールされているシステムに転送します。

このドキュメントでは、Glance がコントローラノード上にあることを前提にしています。

## 5. Glance のグローバルシェル変数を設定します。

```
controller# export OS_USERNAME=glance
controller# export OS_PASSWORD=service-password
controller# export OS_TENANT_NAME=service
controller# export OS_AUTH_URL=$CONTROLLER_ADMIN_NODE:5000/v2.0
```

## 6. UA を Glance リポジトリにアップロードします。

```
controller# glance image-create --container-format bare --disk-format raw \
--is-public true --name "image-name" \
--property architecture=system-arch --property hypervisor_type=solariszones \
--property vm_mode=solariszones < path-to-archive-file
```

ここで、*system-arch* はシステムのアーキテクチャーを指し、x86\_64 または sparc64 のどちらかです。

## イメージに関する情報の表示

イメージ情報を表示するには、nova コマンドまたは glance コマンドのどちらかを使用できます。

```
$ nova image-list
+-----+-----+-----+-----+-----+
| ID | Name | Size | Status | Progress |
+-----+-----+-----+-----+-----+
|    |      |      |        |           |
+-----+-----+-----+-----+-----+
```

```

| ID | Name | Status | Server |
+-----+-----+-----+-----+
| 4dfbfd4f-2de5-4251-832c-e35a4a4145ee | 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、各種イメージのサイズなどの追加情報を表示します。

nova image-show および glance image-show コマンドは、特定のイメージに関する情報を表示します。各コマンドは、イメージに関する異なる出力を生成します。

```
$ nova image-show 'Solaris Non-global Zone'
```

```

+-----+-----+
| Property | Value |
+-----+-----+
| OS-EXT-IMG-SIZE:size | 845025280 |
| created | 2015-11-19T14:46:38Z |
| id | 4dfbfd4f-2de5-4251-832c-e35a4a4145ee |
| metadata architecture | x86_64 |
| metadata hypervisor_type | solariszones |
| metadata vm_mode | solariszones |
| minDisk | 0 |
| minRam | 0 |
| name | Solaris Non-global Zone |
| progress | 100 |
| status | ACTIVE |
| updated | 2015-11-19T14:46:42Z |
+-----+-----+

```

```
$ glance image-show 'Solaris Non-global Zone'
```

```

+-----+-----+
| Property | Value |
+-----+-----+
| Property 'architecture' | x86_64 |
| Property 'hypervisor_type' | solariszones |
| Property 'vm_mode' | solariszones |
| checksum | ba9b9eeddb467833d725c8750a46e004 |
| container_format | bare |
| created_at | 2015-11-19T14:46:38 |
| deleted | False |
| disk_format | raw |
| id | 4dfbfd4f-2de5-4251-832c-e35a4a4145ee |
| is_public | True |
| min_disk | 0 |
| min_ram | 0 |
| name | Solaris Non-global Zone |
| owner | 7d1caf0854b24becb28df5c5cabf72cc |
| protected | False |
| size | 845025280 |
| status | active |
| updated_at | 2015-11-19T14:46:42 |
+-----+-----+

```

---

**注記** - Horizon ダッシュボードから同じイメージ情報を取得できます。

---

## Glance イメージ作成スクリプトの使用

`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
```



# ◆◆◆ 第 5 章

## クラウドの管理

---

この章では、クラウド上でさまざまな管理タスクを実行する方法について説明します。これらのタスクにはダッシュボードまたはコマンド行を使用できます。ダッシュボードで、「プロジェクト」タブの下タスクにはメンバー役割のみが必要で、「管理」タブの下タスクには管理特権が必要です。ダッシュボード上の単一ログインセッションでプロジェクトのすべてのタスクを実行するには、そのプロジェクトのメンバー役割と管理役割の両方を持っているべきです。

この章の内容は次のとおりです。

- 63 ページの「プロジェクトとユーザーの作成」
- 66 ページの「プロジェクト用の内部ネットワークの作成」
- 71 ページの「VM インスタンスの作成とブート」
- 76 ページの「フレーバの管理」
- 80 ページの「VM インスタンスの管理」

## プロジェクトとユーザーの作成

ユーザー admin としてはじめて OpenStack にログインすると、demo プロジェクトのランディングページが表示されます。このプロジェクトから、ほかのプロジェクトの作成に進むことができます。

### ▼ プロジェクトを作成してユーザーを割り当てる方法

新しいプロジェクトまたはテナントを作成し、それらに新しいユーザーを移入するには、この手順を使用します。

1. ブラウザで、次のリンクのような URL でクラウド管理者としてログインします。

`http://system/horizon/`

demo プロジェクトのランディングページが表示されます。

ダッシュボードに関する簡単な概要については、20 ページの「[ダッシュボードの詳細](#)」を参照してください。

2. 左パネルで、「識別情報」>「プロジェクト」タブを選択します。  
デフォルトプロジェクト demo と service が表示されます。
3. 「プロジェクトを作成」をクリックします。
4. 「プロジェクト情報」タブで、プロジェクトの名前とプロジェクトの説明を指定します。  
その作成のあと、新しいプロジェクトがプロジェクトのリストに追加されます。
5. 「識別情報」>「ユーザー」タブを選択します。  
demo と service のデフォルトユーザーが表示されます。
6. 「ユーザーの作成」をクリックします。
7. 該当するフィールドで必要な情報を指定します。
  - a. 新しいユーザー名と割り当てられたパスワードを指定します。
  - b. 「主プロジェクト」ドロップダウンリストから、新しいユーザーが属するプロジェクトを選択します。
  - c. (オプション)「役割」ドロップダウンリストから、プロジェクトユーザーの役割を選択します。  
デフォルトでは、プロジェクトの新しいユーザーにはメンバー役割が与えられます。

## ▼ 既存のユーザーをプロジェクトに移入する方法

新しく作成されたプロジェクトに既存のユーザーを追加するには、この手順を使用します。

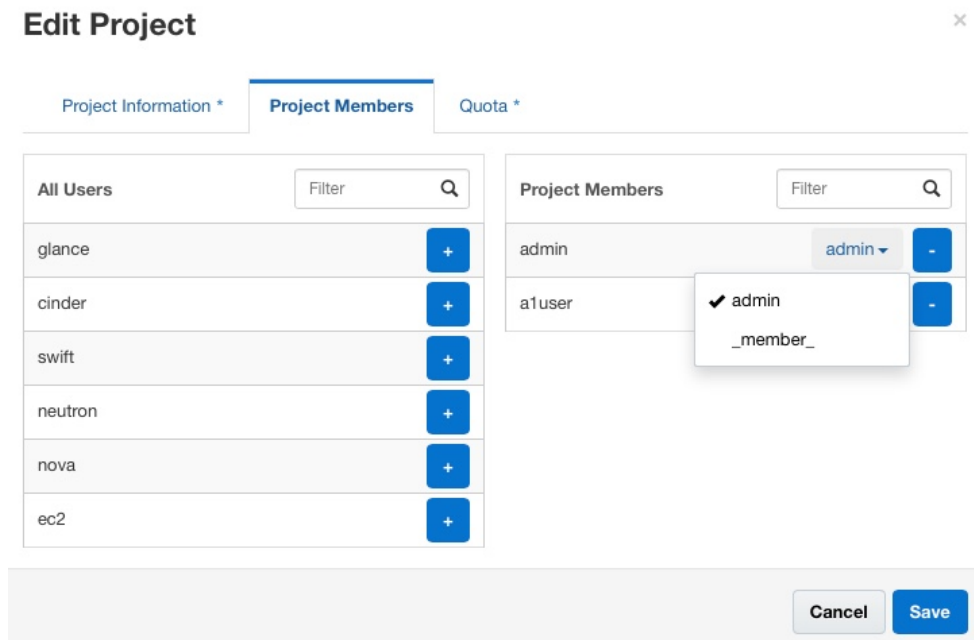
1. デフォルト画面の左パネルで、「識別情報」>「プロジェクト」タブを選択します。
2. 既存ユーザーの追加先プロジェクトの「ユーザーの変更」をクリックします。  
「プロジェクトの編集」ダイアログボックスが表示されます。
3. 「すべてのユーザー」リストで、プロジェクトに追加するユーザー名の右にあるプラス (+) 記号をクリックします。  
デフォルトでは、追加されたユーザーにはそのプロジェクトのメンバー役割が与えられます。



**注記** - service プロジェクトのこれらのユーザーをほかのプロジェクトに追加しないでください。

- glance
- cinder
- swift
- neutron
- nova
- ec2

4. (オプション) プロジェクトのユーザーの役割を変更するには、次の手順を実行します。
- a. プロジェクトメンバーのリストから、役割を変更するユーザーのドロップダウンリストを開きます。
  - b. ユーザーに割り当てる新しい役割を選択します。  
このサンプル図では、現在のプロジェクトの「プロジェクトメンバー」は a1user と admin です。admin ユーザーには、プロジェクトへの管理特権が付与されています。ユーザーにメンバーおよび管理役割の両方を割り当てることができます。



## プロジェクト用の内部ネットワークの作成

プロジェクトは複数の内部ネットワークを持ち、それぞれが対応する仮想マシンインスタンスを提供できます。デフォルトでは、ユーザー通信はそのネットワーク内のみに制限されます。クラウドネットワークを構成するには、サイトに動作しているネットワークが必要です。

### ▼ プロジェクトにネットワークを構成する方法

プロジェクトに内部ネットワークを作成するには、少なくともそのプロジェクトのメンバーであるべきです。この手順を実行するために管理特権は必要ありません。

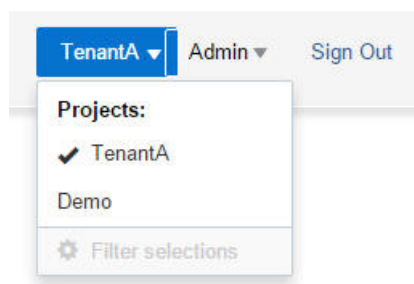
デフォルトでは、このネットワークに外部ネットワークへのアクセス権は付与されていません。内部ネットワークをパブリックネットワークに接続するには、[69 ページの「内部ネットワークに外部接続を提供する方法」](#)を参照してください。

1. ブラウザで、次のアドレスのような URL を使用して Horizon ダッシュボードにログインします。

`http://system/horizon/`

2. デフォルト画面の右上隅にあるプロジェクト名を確認することにより、正しいプロジェクトにログインしていることを確認します。

次の例は、2 つの使用可能なプロジェクトから TenantA が選択されていることを示しています。現在、ユーザー admin がログインしています。



3. 左パネルで、「プロジェクト」>「ネットワーク」>「ネットワーク」タブを選択してから、「ネットワークの作成」をクリックします。

「ネットワークの作成」ダイアログボックスが表示されます。

**Create Network**

Network Subnet Subnet Detail

Network Name

Admin State \*

UP

« Back

Next »

Create a new network. In addition a subnet associated with the network can be created in the next panel.

#### 4. 各タブをクリックして、必要に応じて情報を指定します。

注記 - 「次へ」をクリックすると、各タブの画面が表示されます。

各タブでは、次の情報の入力を求められます。

- 「ネットワーク」タブ
  - ネットワーク名
  - 管理状態 - デフォルト値を受け入れます。
- 「サブネット」タブ
  - サブネット名
  - ネットワークアドレス
  - IP バージョン
  - ゲートウェイ IP - デフォルト値を受け入れるときは空白のままにします。
- 「サブネットの詳細」タブ
  - 「DHCP を使用」 - DHCP を使用しない場合は選択解除します。
  - IP アドレス割り当てプール
  - DNS サーバー名
  - 追加のルート設定

情報を指定して「作成」ボタンをクリックすると、「ネットワーク」画面に、次の例のようなネットワークとそれに関連付けられたサブネットが表示されます。

## Networks

### Networks

<input type="checkbox"/>	Name	Subnets Associated
<input type="checkbox"/>	HR	hr_subnet 10.132.30.0/24
<input type="checkbox"/>	ENG	eng_subnet 10.132.35.0/24

Displaying 2 items

## ▼ 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

ネットワークの構成の一部は、フローティング IP アドレスの作成です。[54 ページの「外部ネットワークを作成する方法」](#)を参照してください。これらの IP アドレスの一部をプロジェクトに割り当てるには、この手順を使用します。

1. 左パネルで、「プロジェクト」>「コンピューター」>「アクセスとセキュリティ」タブを選択します。
2. 「Floating IP」タブをクリックします。
3. 「Floating IP の確保」ボタンをクリックします。  
「Floating IP の確保」ダイアログボックスが開きます。
4. ドロップダウンメニューから、フローティング IP の割り当て元プールを選択します。
5. ダイアログの「IP の確保」ボタンをクリックします。

IP アドレスが「Floating IP」リストに追加されます。IP は必要なだけ、または割り当て制限によって許可されている数だけを割り当てることができます。

## ▼ 内部ネットワークに外部接続を提供する方法

内部ネットワークがより広いパブリックネットワークにアクセスできるようにするには、この手順を使用します。この手順では、内部ネットワークが特定のプロジェクト用にすでに存在していることを前提にしています。ダッシュボードを使用してプロジェクト内部ネットワークを作成するには、[66 ページの「プロジェクト用の内部ネットワークの作成」](#)を参照してください。

始める前に 続行する前に、パブリックアクセスを必要としているサブネット名を取得します。

1. Neutron のグローバルシェル変数を設定します。

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

2. 外部アクセスを必要としているサブネットの ID を識別します。

```
controller# neutron subnet-list | grep subnet-name
```

3. (オプション) ルーターの名前を取得します。

```
controller# neutron router-list
```

4. サブネットの ID をインタフェースとしてルーターに追加します。

```
controller# neutron router-interface-add router-name subnetID
```

### 例 3 内部ネットワークを外部ネットワークに接続

この例では、[66 ページの「プロジェクトにネットワークを構成する方法」](#)で作成された HR 内部ネットワークを使用します。サブネットが HR\_Subnet である HR ネットワークは、パブリックネットワークへのアクセスを必要としています。

```
controller# neutron subnet-list | grep HR_Subnet
| b6feff42-36aa-4235- | HR_Subnet | 10.132.20.0/24 | {"start": "10.132.20.2", |
| 9fe0-ac5de6b43af3 | | | "end": "10.132.20.254"} |
```

```
controller# neutron router-list
+-----+-----+-----+
| id          | name      | external_gateway_info |
+-----+-----+-----+
| f89b24ed-42dd-48b0- | ext-router | {"network_id": "6c4c1823-a203- |
| 8f4b-fd41887a3370 | | 43b1-9674-ddb5ff4185fc", |
| | | "enable_snat": true, |
| | | "external_fixed_ips": |
| | | [{"subnet_id": "83d9b40f-cc61- |
| | | 4696-b22e-b4cbc2aa3872", |
| | | "ip_address": "10.132.10.8"}]} |
+-----+-----+-----+
```

```
controller# neutron router-interface-add ext-router b6feff42-36aa-4235-9fe0-ac5de6b43af3
Added interface b6feff42-36aa-4235-9fe0-ac5de6b43af3 to router ext-router.
```

## L3 エージェント構成情報の表示

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、neutron-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。

- neutron-l3-agent によって作成された VNIC を一覧表示するには、次のようにします。

```
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        0
```

- neutron-l3-agent によって作成された IP アドレスを表示するには、次のようにします。

```
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
```

- IP フィルタ規則を表示するには、次のようにします。

```
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; };
```

- IP NAT 規則を表示するには、次のようにします。

```
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]
```

## VM インスタンスの作成とブート

このセクションの手順を実行するには、少なくともそのプロジェクトの有効なメンバーである必要があります。管理特権は必要ありません。

### ▼ SSH 鍵ペアを作成する方法

1. ブラウザで、次のアドレスのような URL を使用して Horizon ダッシュボードにログインします。  
`http://system/horizon/`
2. デフォルト画面の右上隅にあるプロジェクト名を確認することにより、正しいプロジェクトにログインしていることを確認します。
3. 左パネルで、「プロジェクト」>「コンピューター」>「アクセスとセキュリティ」タブをクリックします。
4. 「キーペア」タブで、鍵ペアを作成するか、または鍵ペアをインポートするかを決定します。

#### ■ 鍵ペアを作成します。

- a. 「キーペアの作成」ボタンをクリックします。
- b. 「キーペア名」フィールドで名前を指定します。
- c. 「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。そうでない場合は、提供されるリンクをクリックして鍵ペアをダウンロードします。

新しい鍵ペアは、「アクセスとセキュリティ」パネルの「鍵ペア」タブに表示されます。

#### ■ 鍵ペアをインポートします。

- a. 「キーペアの取り込み」ボタンをクリックします。
- b. 「キーペア名」フィールドで名前を指定します。
- c. 端末ウィンドウから、root ユーザーの `.ssh/id_rsa.pub` ファイルの内容をコピーし、それを「公開鍵」フィールドに貼り付けます。

d. 「鍵ペアのインポート」ボタンをクリックします。

新しい鍵ペアは、「アクセスとセキュリティ」パネルの「鍵ペア」タブに表示されます。

## ▼ 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てください。71 ページの「SSH 鍵ペアを作成する方法」を参照してください。

内部ネットワークが定義されていることを確認してください。66 ページの「プロジェクト用の内部ネットワークの作成」を参照してください。

1. ブラウザで、次のアドレスのような URL を使用して Horizon ダッシュボードにログインします。

`http://system/horizon/`

2. デフォルト画面の右上隅にあるプロジェクト名を確認することにより、正しいプロジェクトにログインしていることを確認します。
3. 左パネルで、「プロジェクト」>「コンピューター」>「インスタンス」タブをクリックしてから、「インスタンスの起動」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。

### Launch Instance ×

**Details** | Access & Security | Networking

**Availability Zone**  
nova

**Instance Name**

**Flavor**  
Oracle Solaris kernel zone - tiny

**Instance Count**  
1

**Instance Boot Source**  
--- Select source ---

Specify the details for launching an instance.  
The chart below shows the resources used by this project

#### Flavor Details

Name	Oracle Sola...
VCPUs	1
Root Disk	10 GB
Ephemeral Disk	0 GB
Total Disk	10 GB
RAM	2,048 MB

#### Project Limits

**Number of Instances**

**Number of VCPUs**

**Total RAM**

4. 各タブで入力を要求されたら情報を指定します。

次のフィールドの情報を指定します。

- 「詳細」タブ
  - インスタンス名

- フレーバー - ドロップダウンリストから、適切な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。
- インスタンスのブートソース - ドロップダウンリストから、「イメージから起動」を選択します。次に、使用するイメージ名を選択します。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、非大域ゾーンのイメージを選択する必要があります。  
フレーバとイメージのタイプは一致している必要があります。たとえば、イメージのタイプが非大域ゾーンである場合は、フレーバのタイプも非大域ゾーンである必要があります。
- 「アクセスとセキュリティ」タブ - 使用する鍵ペアを選択します。
- 「ネットワーク処理」タブ - 使用可能なネットワークから、新しい VM の接続先ネットワークを選択します。

5. **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。

この手順は、完了するまでにしばらく時間がかかります。

6. **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」列のドロップダウンリストから、「Floating IP の割り当て」を選択します。  
「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。  
使用可能な IP アドレスがない場合は、「+」ボタンをクリックします。68 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. 作成した VM に対応するポートを選択します。
- 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、「ネットワークポリシー」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。次のコマンドでは、鍵ペアとフローティング IP アドレスを使用して root としてゾーンにログインします。

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

## ▼ VM インスタンスにユーザーを追加する方法

Oracle Solaris では、VM インスタンスは Oracle Solaris ゾーンテクノロジーを使用して、ユーザーがクラウド内の仮想マシンをプロビジョニングできるようにします。VM インスタンスにユーザーを追加するには、ゾーン管理者としてコマンドを発行する必要があります。この手順は、ダッシュボードではサポートされません。そのため、端末ウィンドウにアクセスする必要があります。

始める前に ダッシュボードの「管理」>「システム」>「インスタンス」タブをクリックすることによって、VM インスタンスが関連付けられている外部ネットワークのフローティング IP アドレスを取得します。

1. 端末ウィンドウで、VM インスタンスを一覧表示します。

```
# zoneadm list -cv
```

クラウド内の VM の名前には接頭辞 `instance` が含まれています。

2. 特定のゾーンにログインします。

```
# zlogin zonename
```

3. ユーザーのホームディレクトリを作成します。

```
root@zone# mkdir -p /export/home/username
```

4. ユーザーを作成します。

```
root@zone# useradd -d home-dir options
```

ここで、`home-dir` はそのユーザー用に作成したディレクトリです。`useradd` コマンドで使用できるその他のオプションについては、[useradd\(1M\)](#) のマニュアルページを参照してください。

5. ユーザーパスワードを作成するには、次のコマンドを発行し、プロンプトに従います。

```
root@zone# passwd username
```

6. (オプション) パスワードが作成されていることを確認します。

```
root@zone# grep username /etc/passwd
```

7. root パスワードを作成するには、次のコマンドを発行し、プロンプトに従います。

```
zone# passwd root
```

8. ゾーンを終了したあと、ログアウトします。
9. Secure Shell を使用して仮想マシンにログインします。

```
# ssh username@floating-IP
```

ここで、floating-IP はその VM の関連付けられたフローティング IP アドレスです。

#### 例 4 VM インスタンスにユーザーを追加

この例では、ユーザー名 jsmith が VM1 のユーザーとして追加されます。

```
# zoneadm list -cv
ID NAME STATUS PATH BRAND IP
0 global running / solaris shared
6 instance-00000006 running /system/zones/instance-00000006 solaris excl
- myzone installed /system/zones/myzone solaris excl
```

```
# zlogin instance-00000006
[Connected to zone 'instance-00000006' pts/3]
Last login: Wed Jan 6 14:31:18 2016 on pts/2
Oracle Corporation SunOS 5.11 11.3 September 2015
```

```
root@VM1# mkdir -p /export/home/jsmith
root@VM1# useradd -d /export/home/jsmith -m -s /usr/bin/bash jsmith
```

ユーザー jsmith が、デフォルトシェルとして bash を使用して作成されます。

```
root@VM1# passwd jsmith
New Password: password
Re-enter new Password: password
passwd: password successfully changed for jsmith
```

```
root@VM1# passwd root
New Password: password
Re-enter new Password: password
passwd: password successfully changed for root
```

```
root@VM1# exit
logout
```

```
[Connection to zone 'instance-00000006' pts/3 closed]
```

```
# ssh jsmith@10.132.10.9
```

## 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

ンリソースを指定します。Oracle Solaris では、ベース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

クラウド管理者としてダッシュボードにログインしている場合は、「管理」>「システム」>「フレーバ」タブから使用可能なフレーバを表示できます。

図 5 Oracle Solaris 内の Oracle OpenStack 用フレーバ

<input type="checkbox"/>	Flavor Name	VCPUs	RAM	Root Disk	Ephemeral Disk	Swap Disk	ID	Public	Metadata	Actions
<input type="checkbox"/>	Oracle Solaris kernel zone - tiny	1	2048MB	10GB	0GB	0MB	1	Yes	Yes	Edit Flavor ▾
<input type="checkbox"/>	Oracle Solaris non-global zone - tiny	1	2048MB	10GB	0GB	0MB	6	Yes	Yes	Edit Flavor ▾
<input type="checkbox"/>	Oracle Solaris non-global zone - small	4	3072MB	20GB	0GB	0MB	7	Yes	Yes	Edit Flavor ▾
<input type="checkbox"/>	Oracle Solaris kernel zone - small	4	4096MB	20GB	0GB	0MB	2	Yes	Yes	Edit Flavor ▾
<input type="checkbox"/>	Oracle Solaris non-global zone - medium	8	4096MB	40GB	0GB	0MB	8	Yes	Yes	Edit Flavor ▾
<input type="checkbox"/>	Oracle Solaris kernel zone - medium	8	8192MB	40GB	0GB	0MB	3	Yes	Yes	Edit Flavor ▾
<input type="checkbox"/>	Oracle Solaris non-global zone - large	16	8192MB	40GB	0GB	0MB	9	Yes	Yes	Edit Flavor ▾
<input type="checkbox"/>	Oracle Solaris kernel zone - large	16	16384MB	40GB	0GB	0MB	4	Yes	Yes	Edit Flavor ▾
<input type="checkbox"/>	Oracle Solaris non-global zone - xlarge	32	16384MB	80GB	0GB	0MB	10	Yes	Yes	Edit Flavor ▾
<input type="checkbox"/>	Oracle Solaris kernel zone - xlarge	32	32768MB	80GB	0GB	0MB	5	Yes	Yes	Edit Flavor ▾

Displaying 10 items

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。

## フレーバ仕様の変更

各フレーバの「アクション」列で最初の 3 つのオプションのいずれかをクリックすると、そのフレーバのプロパティが表示されるとともに、それらのプロパティを変更できるようになります。次の 3 つのアクションを使用できます。

- 「フレーバの編集」では、そのフレーバに関する情報および変更できるプロパティが表示されます。「フレーバアクセス権」タブでは、そのフレーバにアクセスできるプロジェクトを制

限できます。デフォルト設定はなしです。つまり、そのフレーバはパブリックであり、すべてのプロジェクトがアクセスできます。

- 「アクセス権の変更」では、フレーバの「フレーバアクセス」タブが直接開き、アクセス設定を変更できます。
- 「メタデータの更新」では、フレーバのメタデータを変更できます。

すべてのフレーバ変更をダッシュボードで実行できるわけではありません。たとえば、`extra_specs` プロパティのキーはコマンド行でのみ改訂できます。プロパティのキーは、通常 `zonecfg` コマンドで構成され、OpenStack でサポートされる一連のゾーンプロパティを示します。

次のキーは、カーネルゾーンと非大域ゾーンの両方のフレーバでサポートされます。

- `zonecfg:bootargs`
- `zonecfg:brand`
- `zonecfg:hostid`
- `zonecfg:cpu-arch`

次のキーは、非大域ゾーンのフレーバでのみサポートされます。

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

これらのゾーン構成プロパティについては、[zonecfg\(1M\)](#) のマニュアルページを参照してください。

---

**注記** - すべてのゾーン構成プロパティが OpenStack でサポートされているわけではありません。

---

`sc_profile` キーもまた、コマンド行からのみ変更できます。このキーは、そのフレーバのシステム構成プロファイルを指定するために使用します。

コマンド行からフレーバを変更するには、次の構文を使用します。

```
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。

たとえば、フレーバリスト内の 8 番目のフレーバ (Oracle Solaris kernel zone - large) の特定のシステム構成ファイルを設定するには、次のコマンドを発行します。

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

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。

## ▼ フレーバの extra\_specs プロパティを変更する方法

1. Neutron のグローバルシェル変数を設定します。

```
controller# export OS_USERNAME=nova
controller# export OS_PASSWORD=service-password
controller# export OS_TENANT_NAME=service
controller# export OS_AUTH_URL=http://$CONTROLLER_ADMIN_NODE:5000/v2.0
```

2. 使用可能なフレーバを表示します。

```
controller# nova flavor-list
```

3. 変更しているフレーバの ID を書きとめます。

4. そのフレーバの extra\_specs キーを変更します。

```
controller# nova flavor-key flavor action es-key=value
```

ここで、*es-key* は extra\_specs プロパティの特定のキーを指します。

5. (オプション) フレーバのプロパティを表示します。

### 例 5 zonecfg:bootargs キーの変更

この例では、Oracle Solaris non-global zone - medium フレーバ (ID は 8) の zonecfg:bootargs キーを変更する方法を示します。

領域を節約するために、RTX\_Factor および Is\_Public 列は次の nova flavor-list サンプル出力から省略されています。

```
controller# 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    |
```

```

+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
controller# nova flavor-key 8 set zonecfg:bootargs=-v
controller# nova flavor-show 8

+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Property                | Value                | |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| OS-FLV-DISABLED:disabled | False                | |
| OS-FLV-EXT-DATA:ephemeral | 0                    | |
| disk                     | 40                   | |
| extra_specs              | {"zonecfg:brand": "solaris"} | |
|                           | u'zonecfg:bootargs': u'-v'} | | bootargs modified
| id                       | 8                    | |
| name                     | Oracle Solaris non-global zone - medium | |
| os-flavor-access:is_public | True                 | |
| ram                      | 4096                 | |
| rxtx_factor              | 1.0                  | |
| swap                     |                      | |
| vcpus                    | 8                    | |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

## VM インスタンスの管理

このセクションでは、クラウド上で作成した VM インスタンスを変更する方法について説明します。内容は次のとおりです。

- [80 ページの「VM インスタンスの移行および退避」](#)。
- [82 ページの「VM インスタンスのサイズ変更」](#)。

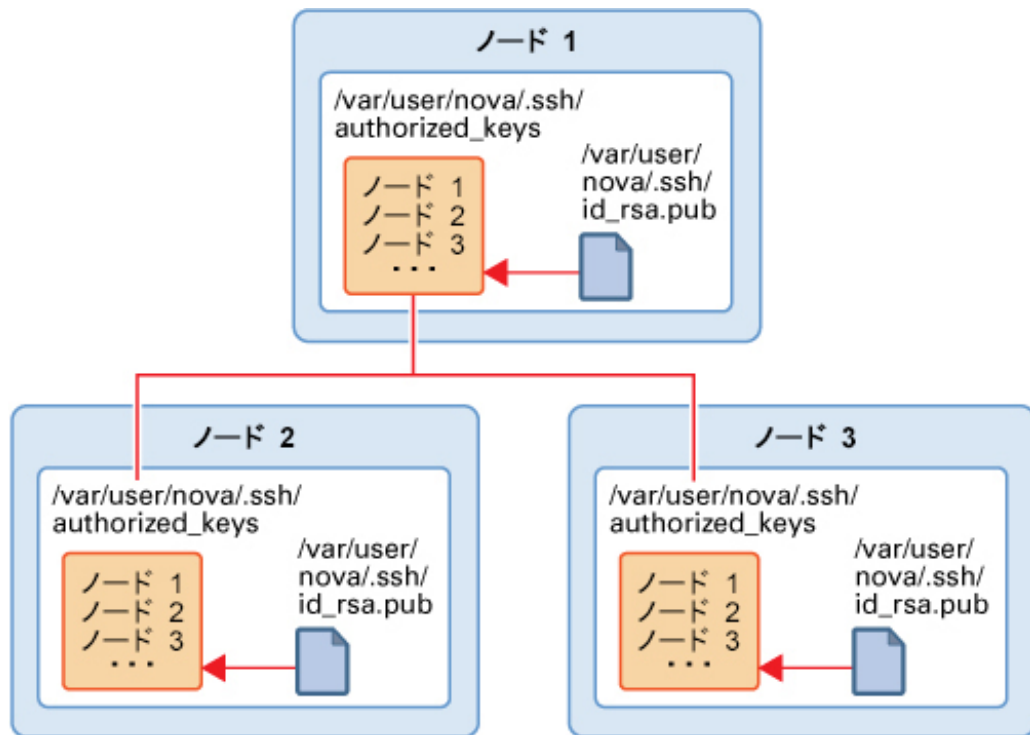
## VM インスタンスの移行および退避

Oracle OpenStack for Oracle Solaris では、VM インスタンスは、Oracle Solaris のスケラブルな高密度仮想環境であるカーネルゾーンまたは非大域ゾーンです。

ゾーンのライブ移行の詳細は、『*Oracle Solaris カーネルゾーンの作成と使用*』のカーネルゾーンの移行に関する説明を参照してください。この本は、[オペレーティングシステムのドキュメント](#)内の使用している Oracle Solaris バージョンのライブラリにあります。[zoneadm\(1M\)](#) および [solaris-kz\(5\)](#) のマニュアルページも参照してください。

ノードの移行を成功させるには、各コンピュータノードの SSH 鍵が各ノードの承認された鍵のファイルに追加されていることを確認する必要があります。それにより、次の図に示すように、これらのノードは承認された鍵の同じファイルを共有します。





次の手順を完了することによって、VM インスタンスの移行を準備します。

1. 各ノードで、SSH 鍵を作成します。

```
# su - nova -c "ssh-keygen -N '' -f /var/user/nova/.ssh/id_rsa -t rsa"
```

2. 各ノードのすべての鍵ファイルを、いずれかのノード内の共通の場所に移動します。
3. すべての鍵を authorized\_keys ファイルに結合します。

例:

```
# cat nova(1)/id_rsa.pub nova(n)/id_rsa.pub >> /var/user/nova/.ssh/authorized_keys
```

ここで、*nova(1)* から *nova(n)* は、参加しているノードの SSH 鍵を表します。

4. authorized\_keys ファイルを、その他のすべての参加しているノードの /var/user/nova/.ssh ディレクトリに配布します。
5. オプションとして、各コンピュータノードの /etc/nova/nova.conf ファイル内の live\_migration\_cipher パラメータに、移行中に使用する暗号化方式を指定します。  
ただし、適切な暗号化方式がこのプロセスで自動的に選択されるようにする場合は、このパラメータを設定解除されたままにします。

インスタンスを移行するには、グローバルシェル変数を設定したあとに次の構文を使用します。

実行中のサーバーの別のマシンへのライブ移行を実行するには、グローバルシェル変数を設定したあとに次の構文を使用します。

```
# nova live-migration server [host]
```

ここで、*server* にはそのサーバーの名前または ID を指定でき、オプションの *host* は宛先サーバーの名前です。

現在のインスタンスのノードに障害が発生した場合や、Nova サービス自体がある期間無効になった場合は、`nova evacuate` コマンドを使用して、そのインスタンスを移動し、それを別のノード上で再構築できます。それにより、このノードを回復できます。

カーネルブーンだけを退避できます。退避は、ルートデバイスが共有ストレージ上に存在する構成でサポートされます。

あるホストから別のホストへのすべての VM インスタンスのライブ移行を実行するには、グローバルシェル変数を設定したあとに次の構文を使用します。

```
# nova host-evacuate-live [--target-host target] server
```

## VM インスタンスのサイズ変更

VM のサイズは、その VM が起動される基盤となるフレーバによって示されます。VM インスタンスを作成する手順については、[72 ページの「VM インスタンスを作成する方法」](#)を参照してください。次の図は、Horizon ダッシュボードに表示されているサンプル VM `hr_vm1` に関する詳細を示しています。

図 6 VM インスタンスのサイズ

### Instances

#### Instances

<input type="checkbox"/>	Instance Name	Image Name	IP Address	Size
<input type="checkbox"/>	<a href="#">hr_vm1</a>	Solaris Non-global Zone	10.132.20.5 10.132.10.10	Oracle Solaris non-global zone - tiny

Displaying 1 item

この図は、`hr_vm1` のサイズが、`Oracle Solaris non-global zone - tiny` フレーバに定義されているサイズであることを示しています。インスタンス名をクリックすることにより、特定のフレーバに関する詳細を確認できます。

- RAM = 2G バイト
- VCPU 数 = 1 VCPU
- ディスク = 10G バイト

インスタンスのサイズを変更することは、インスタンスに別のフレーバを使用することを意味します。

デフォルトでは、サイズ変更プロセスは、新しいサイズのインスタンスを新しいノード上に作成します。ただし、非大域ゾーンのフレーバにサイズ変更している場合は、同じノード内で完了するようにサイズ変更プロセスを設定する必要があります。非大域ゾーンは、その元の大域ゾーンと同じバージョンのものである必要があります。非大域ゾーンのフレーバにサイズ変更し、そのインスタンスが別のノードに作成された場合は、そのインスタンスが大域ゾーンのバージョンが異なるノードに配置されるリスクがあります。このリスクを回避し、サイズ変更されたインスタンスを同じホスト上に作成するには、`/etc/nova/nova.conf` ファイル内の `allow_resize_to_same_host` を `true` に設定します。

ただし、カーネルゾーンには非大域ゾーンのこの制限はありません。そのため、カーネルゾーンを別のノードにサイズ変更できます。

## ▼ VM インスタンスのサイズを変更する方法

始める前に 変更する VM インスタンスの現在のサイズを知ってください。この情報はダッシュボードから取得できます。例については、[図6](#)を参照してください。

1. グローバルシェル変数を設定します。

```
# export OS_USERNAME=nova
# export OS_PASSWORD=service-password
# export OS_TENANT_NAME=service
# export OS_AUTH_URL=http://$CONTROLLER_ADMIN_NODE:5000/v2.0"
```

2. 非大域ゾーンのフレーバにサイズ変更している場合は、`/etc/nova/nova.conf` ファイル内の次のパラメータを次のように編集します。

```
allow_resize_to_same_host=true
```

3. フレーバのリストを表示します。

```
# nova flavor-list
```

4. VM インスタンスで使用する新しいフレーバの ID 番号を書きとめます。

5. フレーバの ID 番号を参照することによって、新しいフレーバを使用する VM インスタンスを変更します。

```
# nova resize instance-name flavor-id
```

たとえば、現在、インスタンス `hr_vm1` のサイズが `Oracle Solaris non-global zone - tiny` フレーバ (ID 番号は 6) に設定されているとします。このインスタンスのサイズを `Oracle`

Solaris non-global zone - medium フレーバ (ID 番号は 8) に変更します。その場合、次のように入力します。

```
# nova resize --poll hr_vm1 8
```

--poll オプションを使用すると、サイズ変更プロセスが完了するまでその進行状況を確認できます。

6. (オプション) ダッシュボードで、インスタンスのサイズが新しいフレーバに変更されたことを確認します。

あるいは、次のコマンドを使用して同じ確認を行うことができます。

```
# nova show instance-name
```

flavor フィールドの値が、前の手順の新しいフレーバを指定していることを確認します。

次の例は、展開された出力です。インスタンス hr\_vm1 が TenantA という名前のプロジェクト内に作成され、Oracle Solaris non-global zone - medium にサイズ変更されています。

```
# export OS_USERNAME=admin
# export OS_PASSWORD=admin-password
# export OS_TENANT_NAME=TenantA
# export OS_AUTH_URL=http://$CONTROLLER_ADMIN_NODE:5000/v2.0"
```

```
# nova show hr_vm1
```

```
+-----+-----+
| Property      | Value                                     |
+-----+-----+
| ...           |                                           |
| created       | 2016-01-26T12:38:47Z                     |
| flavor        | Oracle Solaris non-global zone - medium (8) |
| ...           |                                           |
+-----+-----+
```

## Cinder の構成と配備のためのオプション

---

この章では、OpenStack 設定の Cinder やストレージコンポーネントを構成できる代替の方法について説明します。

---

**注記** - ブロックストレージを構成するための 46 ページの「[ストレージノードの構成](#)」の手順は引き続き、ストレージノードを設定するためのデフォルトの方法です。この章のオプションは、ストレージコンポーネントを配備するための代替の方法を提供します。ただし、現時点では、クラウドフレームワーク上の Cinder を設定するためにデフォルトのストレージ構成方法と代替の構成方法の両方を同時に使用することは推奨されていません。

---

この章で扱う内容は、次のとおりです。

- [85 ページの「ストレージのためのリモートシステムの配備」](#)
- [90 ページの「Oracle ZFS Storage Appliance での OpenStack の使用」](#)

### ストレージのためのリモートシステムの配備

このセクションで説明されている Cinder 配備は、Oracle Solaris 11.3 SRU 2 以降のリリースを実行している Juno 設定に適用されます。

SAN のサポートのない以前の OpenStack リリースでは、Cinder ボリュームサービスは、ZFS iSCSI ドライバが使用されている場合はターゲットホスト上で動作するように構成する必要がありました。

Cinder でのストレージエリアネットワーク (SAN) のサポートにより、SSH を使用して複数のストレージホストバックエンドを構成したり、これらのホストのボリュームタイプを定義してそれらがどのように使用されるかを決定したりできます。このタイプの配備では、Cinder パッケージやすべての Cinder サービスは、イニシエータホスト (多くの場合、コンピュータノードとしても機能するホスト) にのみインストールされます。

リモートのターゲットホストに OpenStack パッケージをインストールする必要はありません。これらのホストは、COMSTAR に基づいてイニシエータホストに LUN ディスクを提供するだけです。

ストレージのためのリモートシステムの正しい配備には、次の要件があります。

- /etc/cinder/cinder.conf ファイルの構成。
- 指定されたユーザーへの適切な権利プロファイルの付与。
- 追加パッケージの手動インストール。

以降のセクションでは、これらの要件について詳細に説明します。

## cinder.conf ファイルの構成

イニシエータホストに Cinder パッケージをインストールしたあと、/etc/cinder/cinder.conf ファイルの [DEFAULT] セクションを編集します。

構成ファイル内の定義を理解するためのガイドとして、次の例を参照してください。

```
[DEFAULT]
my_ip = localhost
osapi_volume_workers = 1
auth_strategy = keystone
#num_volume_device_scan_tries = 6
os-volume-api-version = 2
scheduler_driver=cinder.scheduler.filter_scheduler.FilterScheduler

enabled_backends=zfsdriver-1, zfsdriver-2, zfsdriver-3, zfsdriver-4, zfsdriver-5

[zfsdriver-1]
volume_group=zvolumes-1
volume_driver=cinder.volume.drivers.solaris.zfs.ZFSVolumeDriver
volume_backend_name=ZFS_LOCAL
zfs_volume_base = rpool/cinder
san_is_local = True
debug=true
verbose=true
[zfsdriver-2]
volume_group=zvolumes-2
volume_driver=cinder.volume.drivers.solaris.zfs.ZFISISCSIDriver
volume_backend_name=ZFS_REMOTE
zfs_volume_base = rpool/cinder
san_ip = 10.134.13.38
san_login = jdoe
san_password = password
debug=true
verbose=true
[zfsdriver-3]
volume_group=zvolumes-3
volume_driver=cinder.volume.drivers.solaris.zfs.ZFSFCDriver
volume_backend_name=ZFS_REMOTE_FC
zfs_volume_base = rpool/fc
san_ip = 10.134.13.38
san_login = jdoe
```

```

san_password = password
debug=true
verbose=true
[zfsdriver-4]
volume_group=zvolumes-4
volume_driver=cinder.volume.drivers.solaris.zfs.ZFSISCSIDriver
volume_backend_name=ZFS_REMOTE
zfs_volume_base = rpool/cinder/zq
san_ip = 10.134.13.38
san_login = jdoe
san_password = password
debug=true
verbose=true
[zfsdriver-5]
volume_group=zvolumes-5
volume_driver=cinder.volume.drivers.solaris.zfs.ZFSISCSIDriver
volume_backend_name=ZFS_REMOTE
zfs_volume_base = rpool/zq
san_ip = 10.134.63.182
san_login = jdoe
san_password = password
debug=true
verbose=true

```

enabled\_backends= 有効になっているバックエンドホストをリストします。この例では、5 つのバックエンドホストが定義されています。

zfsdriver-1, これらのバックエンドホストは、3 つのホスト上で 3 つの ZFS ドライバ (ZFSVolumeDriver、ZFSISCSIDriver、ZFSFCDriver) を使用します。これらのホストのうち、1 つはローカル (localhost) であり、その他はリモート (10.134.13.38 および 10.134.63.182) です。

zfsdriver-2,  
zfsdriver-3,  
zfsdriver-4,  
zfsdriver-5

volume\_backend\_name= ボリュームタイプを指定された名前前で定義します。次のサンプルコマンドは、それぞれ次のアクションを実行します。

ZFS\_REMOTE

```

# cinder type-create type-remote
# cinder type-key type-remote set volume_backend_name=ZFS_REMOTE
# cinder create --display-name t1 --volume-type type-remote

```

前の例で、各コマンドは次のアクションを実行します。

- 新しいボリュームタイプを作成します。
- 新しいボリュームタイプにバックエンド名を割り当てます。
- 新しいボリュームタイプに基づいて新しいボリュームを作成します。

最後のコマンドを実行すると、フィルタリング規則に基づいて、ZFS\_REMOTE という名前を持つバックエンドの 1 つに新しいボリューム t1 が作成されます。

zfs\_volume\_base 各ボリュームバックエンド上の新しい ZFS ボリュームの基本データセットを指定します。

= rpool/zq

san\_is\_local ZFS ドライバはすべて SAN ドライバに基づいているため、san\_\* パラメータを設定する必要があります。  
 san\_ip san\_login  
 san\_password 各コマンドは、バックエンドホスト上でローカルに、または SSH を使用してリモートで実行されます。そのため、バックエンドごとに san\_is\_local = True を設定するか、または san\_ip\* パラメータを構成しますが、この両方はいりません。

debug=true デバッグのためのオプションパラメータ。これらのパラメータの構成は省略できます。  
 verbose=true

Cinder を正しく構成したあとにサービスを一覧表示すると、各サービスの状態が表示されません。

```
# cinder service-list
```

Binary	Host	Zone	Status	State	Updated_at	Disabled Reason
cinder-backup	host-2	nova	enabled	up	2015-10-13 T19:22:45.000000	None
cinder-scheduler	host-2	nova	enabled	up	2015-10-13 T19:22:43.000000	None
cinder-volume	host-2	nova	enabled	down	2015-10-13 T18:31:41.000000	None
cinder-volume	@zfsdriver-1	nova	enabled	up	2015-10-13 T19:22:46.000000	None
cinder-volume	host-2 @zfsdriver-2	nova	enabled	up	2015-10-13 T19:22:47.000000	None
cinder-volume	host-2 @zfsdriver-3	nova	enabled	up	2015-10-13 T19:22:48.000000	None
cinder-volume	host-2 @zfsdriver-4	nova	enabled	up	2015-10-13 T19:22:47.000000	None
cinder-volume	host-2 @zfsdriver-5	nova	enabled	up	2015-10-13 T19:22:48.000000	None
cinder-volume	host-2 @zfsdriver-6	nova	enabled	down	2015-10-13 T18:32:55.000000	None



## 指定されたユーザーへの権利の付与

san\_login によって定義されたユーザーがリモートのターゲットを使用できるようにするには、そのユーザーに適切な権利プロファイルを付与する必要があります。次の例は、ユーザーの権利プロファイルを作成する方法を示しています。

```
# useradd -s /usr/bin/pfbash -m jdoe
# passwd jdoe password
# profiles -p "Cinder Storage management"
profiles:Cinder Storage management> set desc="Cinder Storage management on target host"
profiles:Cinder Storage management> add profiles="File System Management"
profiles:Cinder Storage management> add auths="solaris.smf.modify.stmf"
profiles:Cinder Storage management> add cmd=/usr/sbin/itadm
profiles:Cinder Storage management:itadm> set euid=0
profiles:Cinder Storage management:itadm> end
profiles:Cinder Storage management> add cmd=/usr/sbin/stmfadm
profiles:Cinder Storage management:stmfadm> set euid=0
profiles:Cinder Storage management:stmfadm> end
profiles:Cinder Storage management> add cmd=/usr/sbin/zfs
profiles:Cinder Storage management:zfs> set priv=sys_config,sys_mount
profiles:Cinder Storage management:zfs> end
profiles:Cinder Storage management> exit

# usermod -P "Cinder Storage management" jdoe
```

プロファイルと権利の詳細は、『[Oracle Solaris 11.3](#) でのユーザーとプロセスのセキュリティ保護』を参照してください。

## ターゲットとしてのリモートホストの有効化

リモートホストをターゲットとして有効にするには、リモートホスト上で次の手順を実行します。

1. リモートホストにパッケージ group/feature/storage-server をインストールします。

```
# pkg install storage-server
```

2. ホスト上で次のサービスを有効にします。

- svc:/system/stmf:default
- svc:/network/iscsi/target:default

例:

```
remote-host# svcadm enable stmf
remote-host# svcadm enable -r svc:/network/iscsi/target:default
```

3. 定義された zfs\_volume\_base のアクセス制御リスト (ACL) を初期化および設定します。  
たとえば、構成ファイル内に zfs\_volume\_base=rpool/fc という定義が含まれているとします。その場合は、次のコマンドを実行する必要があります。

```
# chmod A+user:cinder:add_subdirectory:allow /rpool/fc
# zfs allow cinder clone,create,destroy,mount,snapshot rpool/fc
```

## Oracle ZFS Storage Appliance での OpenStack の使用

このセクションでは、Oracle ZFS Storage Appliance (ZFSSA) をバックエンドストレージとして使用する OpenStack 構成について説明します。

### Oracle ZFS Storage Appliance について

Oracle ZFS Storage Appliance 製品ファミリは、ネットワーククライアントに対して効果的なファイルサービスおよびブロックデータサービスを提供し、システムに保存されたデータに適用できる一連のデータサービスを提供します。これらのサービスには、次の技術が含まれます。

- Analytics - システムの動作をリアルタイムで動的に観察し、データをグラフィカルに表示できます。
- ZFS ハイブリッドストレージプール - オプションのフラッシュメモリーデバイス、低消費電力で高容量のディスク、DRAM メモリーなどのさまざまなデバイスを単一のデータ階層として管理できます。
- さまざまなハードウェアのサポート

Oracle OpenStack for Oracle Solaris の `cloud/openstack/cinder` パッケージには、Oracle ZFSSA iSCSI Cinder ドライバが含まれています。このドライバにより、アプライアンスを Cinder コンポーネント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。具体的には、このドライバを使用すると、Cinder サーバーが Nova サービスによってインスタンス化された仮想マシンに割り当てることができる iSCSI ボリュームを作成できます。Oracle ZFSSA をストレージとして使用するには、まずアプライアンスが ZFSSA ソフトウェアリリースの少なくともバージョン 2013.1.2.0 を実行していることを確認してください。

### Oracle ZFSSA を使用した OpenStack の構成

このセクションには、Oracle ZFSSA を設定するための詳細な手順は含まれていません。構成手順を含むアプライアンスの詳細は、次のソースを参照してください。

- [Oracle Help Center ストレージのドキュメントページ](#)にある Oracle ZFSSA 製品ドキュメント
- [Oracle ZFS Storage Appliance を OpenStack Cinder のストレージバックエンドとして使用する \(http://www.oracle.com/technetwork/server-storage/sun-unified-storage/documentation/openstack-cinder-zfssa-120915-2813178.pdf\)](http://www.oracle.com/technetwork/server-storage/sun-unified-storage/documentation/openstack-cinder-zfssa-120915-2813178.pdf)。このホワイト

ペーパーでは、クラウド内で使用するためにストレージアプライアンスを設定する方法について詳細に説明しています。

OpenStack を Oracle ZFSSA とともに設定する手順の前に、データストレージトラフィックをホストするサブネットをすでに作成していることを確認してください。この章のシナリオでは、この開始点のための基盤として [図2](#) を使用しますが、少し違いがあります。ここでは、ストレージデータトラフィックは 2 つのサブネットにホストされ、そのトラフィックのインタフェースとして net2 と net3 を使用します。この 2 つのサブネットにより、データストレージサービスの効率的なパフォーマンスと可用性が保証されます。

## ZFSSA 側

Oracle ZFSSA は、ネットワーク経由で SCSI コマンドを交換するために iSCSI を使用します。iSCSI ノード間の通信を設定するには、次の情報が入手されている必要があります。

- SCSI ノードの DNS 名または IP アドレス。SCSI ノードは、イニシエータノードとターゲットノードの両方で構成されます。
- ターゲットノード上の TCP iSCSI インターネットポート。デフォルトでは、このポート番号は 3260 です。
- イニシエータノードとターゲットノードの両方の iSCSI 修飾名 (IQN)。
- チャレンジハンドシェイク認証プロトコル (CHAP 認証) を使用したオプションの認証情報

この準備手順に関連する iSCSI 情報を取得するには、Oracle Solaris `iscsiadm` コマンドを使用します。

このドキュメントでは、アプライアンスを構成する手順については説明していません。運用で使用するためにアプライアンスを準備する手順については、Oracle の [ストレージのドキュメント](#) にある該当するリリースドキュメントを参照してください。

## OpenStack 側

OpenStack ZFSSA Cinder ドライバは、アプライアンス上に関連する LUN を作成し、これらの LUN の適切なプロパティを設定し、さらにボリュームが使用されている OpenStack コンピュートノードやゲストインスタンスへの LUN の可視性を管理することによってボリュームを作成および管理します。

Oracle ZFSSA を設定したら、ZFSSA iSCSI Cinder ドライバを構成できます。

## ▼ OpenStack 用に Oracle ZFSSA を構成する方法

この手順では、次のタスクを実行するワークフロー `cinder.akwf` を使用します。

- 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。

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. 次のいずれかの方法を使用して、ワークフロー `cinder.akwf` を実行します。

### ■ 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-cinder.akwf-file"
    url = "url-to-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
zfssa:maintenance workflow-001 execute (uncommitted)>

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

user と password の場合、これらの値は cinder.conf ファイル内の san_login および
san_password パラメータで定義されます。
```

### ■ 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 ドライブ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` ファイル内の次の必須プロパティを指定します。

- `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、セキュリティなど) を設定するために追加の ZFSSA プロパティ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   ADDR          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
```

## Heat の操作

---

この章では、Oracle Solaris で実装およびサポートされる Heat について説明します。内容は次のとおりです。

- 95 ページの「Heat コンポーネントについて」
- 96 ページの「Heat のインストール」
- 96 ページの「HOT テンプレートについて」

### Heat コンポーネントについて

Heat は OpenStack のオーケストレーションエンジンであり、作成した Heat オーケストレーション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ます。これらのテンプレートは、HOT テンプレートとも呼ばれます。

HOT テンプレートを使用すると、インスタンス、フローティング IP、ボリューム、ユーザーなどのさまざまな OpenStack リソースタイプを作成できます。テンプレートを使用すると、インスタンスの高可用性、インスタンスの自動拡大縮小、入れ子のスタックなどの高度な機能を配備することもできます。したがって、Heat を使用することで、すべての OpenStack コアプロジェクトで受け入れるユーザーベースが大きくなる可能性があります。テンプレートを使用すると、デフォルトのリソース実装をオーバーライドする方法が提供されます。それにはテンプレートからエンジンに渡されるパラメータを使用します。

Heat のサービスは、RESTful Web サービス API 経由で提供されます。すべての OpenStack アプリケーションと同様に、Python WSGI インタフェースが使用され、ペーストを使用してアプリケーションが同時に構成されます。アプリケーションの HTTP エンドポイントは、Web Server Gateway Interface (WSGI) ミドルウェアのパイプラインで構成されています。Heat は具体的には、Heat API 用のポート 8004 と Heat CloudFormation 用のポート 8000 の 2 つのエンドポイントを使用します。

Heat 独自の構成は、`/etc/heat/heat.conf` 構成ファイルで制御されます。この時点で、プライマリ Heat 構成ファイルには Solaris 固有の構成パラメータが存在しません。

Heat コンポーネントの詳細については、OpenStack コミュニティーの [Heat ドキュメント](#) を参照してください。

## Heat のインストール

一般的な構成では、Keystone と同じノード上に Heat サービスをインストールします。次のコマンドを使用してノード上に OpenStack をインストールした場合は、Heat パッケージが自動的に含まれています。

```
# pkg install openstack
```

### ▼ 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. `/etc/heat/heat.conf` 内のパラメータをコメント解除または設定することによって Heat を構成します。

```
[database]
connection = mysql://heat:heat-service-password@$CONTROLLER_ADMIN_NODE/heat

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

2. 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
```

## HOT テンプレートについて

Heat を使用して OpenStack 構成で複数の複合クラウドアプリケーションのオーケストレーションを行うには、Heat オーケストレーションテンプレート (HOT) を定義する必要があります。HOT テンプレートには、ユーザーが入力する必要がある指定が含まれています。指定したパラメータは、リソースタイプやその他の高度な機能を作成するプロセスの実行時に読み取られません。

HOT テンプレートの指定とその説明については、[http://docs.openstack.org/developer/heat/template\\_guide/hot\\_spec.html#hot-spec](http://docs.openstack.org/developer/heat/template_guide/hot_spec.html#hot-spec) を参照してください。

HOT テンプレートを作成する方法については、[http://docs.openstack.org/developer/heat/template\\_guide/hot\\_guide.html#hot-guide](http://docs.openstack.org/developer/heat/template_guide/hot_guide.html#hot-guide) を参照してください。



HOT テンプレートを配備するには、次のコマンドを使用します。

```
# heat stack-create -f template
```

*template* にはテンプレートの場所のフルパスが含まれます。

次の例は、サブネットを持つ 3 つのプライベートネットワークを作成する `stack3` という名前の HOT テンプレートの内容を示しています。

---

**注記** - 行 `heat_template_version: 2013-05-23` は、テンプレートの最上部にある必要があります。

---

```
heat_template_version: 2013-05-23
description: Create a few networks.

resources:
  heat_net1:
    type:OS::Neutron::Net
    properties:
      name:heat_net1

  heat_subnet1:
    type:OS::Neutron::Subnet
    properties:
      name:heat_subnet1
      network_id: { get_resource: heat_net1 }
      cidr: 192.168.50.0/24

  heat_net2:
    type:OS::Neutron::Net
    properties:
      name: heat_net2

  heat_subnet2:
    type:OS::Neutron::Subnet
    properties:
      name:heat_subnet2
      network_id: { get_resource: heat_net2 }
      cidr: 192.168.51.0/24

  heat_net3:
    type:OS::Neutron::Net
    properties:
      name: heat_net3

  heat_subnet3:
    type:OS::Neutron::Subnet
    properties:
      name:heat_subnet3
      network_id: { get_resource: heat_net3 }
      cidr: 192.168.52.0/24
```



## Ironic の操作

---

この章では、Oracle Solaris で実装およびサポートされる Ironic について説明します。内容は次のとおりです。

- 99 ページの「Ironic コンポーネントについて」
- 100 ページの「Ironic のインストールおよび構成」
- 104 ページの「概要: Ironic でのベアメタルの配備」
- 106 ページの「Ironic を使用したベアメタルの配備」

### Ironic コンポーネントについて

前の章では、クラウドを作成するためのコア OpenStack コンポーネントについて説明しました。追加のコンポーネントにより、クラウドの管理に関連したその他のサービスが提供されます。この章では、Juno リリースで使用可能なコンポーネント Ironic について説明します。

OpenStack のコアコンポーネントが仮想マシンまたは VM インスタンスのプロビジョニングを可能にするのに対して、Ironic は、ベアメタルインスタンスまたはノードを登録、プロビジョニング、および廃棄するためのサービスを提供します。Ironic は、PXE ブートや IPMI などの一般的なテクノロジーを使用して、プロビジョニング可能な幅広いハードウェアをサポートします。さらに、プラグイン可能なドライバメカニズムにより、Ironic はベンダー固有のハードウェアを管理およびサポートできます。

Ironic およびそれにより提供される利点の詳細は、OpenStack コミュニティ Web サイトにある Ironic の [開発者ドキュメント](#) を参照してください。

Ironic は、3 つの主要なコンポーネントで構成されています。Oracle Solaris では、これらのコンポーネントが SMF サービスとして提供されます。次の表に、これらのコンポーネントの一覧と説明を示します。

コンポーネント	説明	SMF サービス
OpenStack Ironic API サービス	オペレータやその他のサービスが管理対象のベアメタルノードと対話するために使用できる RESTful API を提供するサービス。	svc:/application/openstack/ironic/ironic-api
OpenStack Ironic コンダクタ サービス	参照およびベンダー固有のドライバを使用してベアメタルノードの実際のプロビジョニング	svc:/application/openstack/ironic/ironic-conductor

コンポーネント	説明	SMF サービス
	グを実行するメインコントローラ。コンダクタサービスと API サービスは、RPC を使用して通信します。	
OpenStack Ironic データベースサービス	Ironic のバックエンドデータベースを作成して同期するための一時的な SMF サービス。	svc:/application/openstack/ironic/ironic-db

## Ironic のインストールおよび構成

Ironic は、スタンドアロンのコンポーネントとして、ほかの OpenStack サービスなしで使用できます。または、ほかの OpenStack コンポーネントとともに (通常はコンピュータノード上に) 配備できます。その構成は、Ironic を配備する方法によって異なります。

Oracle Solaris ベアメタルインスタンスをプロビジョニングするには、Automated Installer (AI) が必要です。AI は、Ironic サービスと同じノード上に配置できます。または、AI をリモートサーバー上で使用して Ironic と連携させることができます。

AI の構成および使用の詳細は、[オペレーティングシステムのドキュメント](#)内の該当するライブラリにあるインストールの本を参照してください。

### ▼ Ironic をインストールおよび構成する方法

**始める前に** Ironic を残りの OpenStack コンポーネントとともに配備している場合は、最初にそれらのコアコンポーネントを構成するようにしてください。少なくとも、Ironic に関して作業する前に Keystone と Glance の構成を完了する必要があります。

同様に、Ironic 用のデータベースが設定されていることを確認してください。この手順は、データベースが残りの OpenStack データベースとともにコントローラノード上に設定される、[図2](#)の 3 ノードリファレンスアーキテクチャーに従っています。

#### 1. Ironic データベースを作成します。

##### a. コントローラ管理者ノードのグローバルシェル変数を設定します。

```
controller# export $CONTROLLER_ADMIN_NODE=controller-node
```

ここで、*controller-node* はそのコントローラの IP アドレスまたはホスト名のどちらかです。変数の設定の詳細は、[26 ページ](#)の「[ホスト名、変数、およびパスワードの準備](#)」を参照してください。

##### b. 次のコマンドを使用してデータベースを作成します。

```
controller# mysql -u root -p
Enter password: MySQL-root-password
```

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

## 2. OpenStack Ironic パッケージをインストールします。

```
# pkg install ironic ironicclient rabbitmq
# pkg update stevedore stevedore-27
```

## 3. AI サーバーおよび管理ツールをインストールします。

AI サーバーパッケージを Ironic とともにローカルにインストールするか、または別のホストにリモートにインストールするかにかかわらず、同じコマンドを使用します。

```
# pkg install pkg:/install/installadm
```

## 4. AI がリモートに配置されている場合は、Ironic ユーザーをそのサーバー上に構成します。

```
remote-AI# useradd -d /var/lib/ironic -m -g 88 -u 88 \
  -P "Install Service Management" ironic
remote-AI# password ironic
New Password: password
Re-enter new Password: password
```

---

**注記** - AI が Ironic とともにローカルにインストールされる場合は、Ironic インストールプロセスによって ironic ユーザーがそのシステム上に自動的に作成されます。

---

## 5. Ironic ユーザーの SSH 鍵を作成および管理します。

- AI が Ironic とともにローカルに存在する場合は、そのシステム上で次のコマンドを発行します。

```
ironic-localhost# su - ironic
ironic-localhost# mkdir /var/lib/ironic/.ssh
ironic-localhost# ssh-keygen -N '' -t rsa \
  -f /var/lib/ironic/.ssh/id_rsa
ironic-localhost# cat /var/lib/ironic/.ssh/id_rsa.pub > \
  /var/lib/ironic/.ssh/authorized_keys
```

- AI がリモートに配置されている場合は、次の手順を実行します。

### a. リモート AI で、次のコマンドを発行します。

```
remote-AI# su - ironic
remote-AI# mkdir /var/lib/ironic/.ssh
remote-AI# ssh-keygen -N '' -t rsa \
  -f /var/lib/ironic/.ssh/id_rsa
remote-AI# cat /var/lib/ironic/.ssh/id_rsa.pub > \
  /var/lib/ironic/.ssh/authorized_keys
```

## b. Ironic ホスト上で、次のコマンドを発行します。

```

ironic-localhost# mkdir /var/lib/ironic/.ssh
ironic-localhost# scp ironic@AI-server:~/.ssh/id_rsa /var/lib/ironic/.ssh
ironic-localhost# scp ironic@AI-server:~/.ssh/id_rsa.pub /var/lib/ironic/.ssh
ironic-localhost# cat /var/lib/ironic/.ssh/id_rsa.pub > \
/var/lib/ironic/.ssh/authorized_keys
ironic-localhost# chown -R ironic:ironic /var/lib/ironic/.ssh

```

ここで、*AI-server* には AI サーバーの IP アドレスまたはホスト名を指定できます。

6. `/etc/ironic/ironic.conf` ファイルを編集します。

このファイル内の特定のパラメータの構成に関する次の考慮事項に注意してください。

<code>auth_strategy</code>	DEFAULT セクションで、Ironic をスタンドアロンのコンポーネントとして使用している場合は、 <code>noauth</code> を指定します。Ironic をほかの OpenStack コンポーネントとともに配備している場合は、 <code>keystone</code> を指定します。
<code>server</code>	<code>ai</code> セクションで、Automated Installer (AI) が Ironic でローカルの場合は <code>localhost</code> を指定します。AI がリモートに存在する場合は、そのサーバーの IP アドレスまたはホスト名を指定します。
<code>connection</code>	<code>[database]</code> セクションで、MySQL データベースへの接続を指定します。
<code>glance_host</code>	<code>glance</code> セクションで、 <code>auth_strategy</code> パラメータが <code>keystone</code> に設定されている場合は Glance サーバーの IP アドレスまたはホスト名を指定します。
<code>glance_api_servers</code>	また、 <code>glance</code> セクションで、ホスト名または IP アドレスのどちらかをそれに対応するポート番号 (192.168.0.150:9292 など) とともに指定します。 <code>localhost</code> を指定しないでください。

Ironic が機能するために必要なすべてのパラメータの構成に役立つように、次のサンプルファイルを使用してください。このサンプルファイルで、変数 `glance-serverIP` は Glance サーバーの IP アドレスまたはホスト名を表しています。

```

[DEFAULT]
enabled_drivers=solaris
auth_strategy= setting depends on whether Ironic is stand alone or not
pybasedir = /usr/lib/python2.6/vendor-packages/ironic
bindir = /usr/lib/ironic
host = ironic

[ai]
server= setting depends on whether AI is local or remote
username=ironic

```

```

port=22
timeout=10
ssh_key_file=/var/lib/ironic/.ssh/id_rsa
deploy_interval=30

[api]
port=6385

[conductor]
api_url=http://localhost:6385/
heartbeat_timeout=60
heartbeat_interval=60
sync_power_state_interval=300
check_provision_state_interval=120

[database]
connection= mysql://ironic:ironic-service-password@$CONTROLLER_ADMIN_NODE/ironic

[solaris_ipmi]
imagecache_dirname = /var/lib/ironic/images
imagecache_lock_timeout = 60

[glance]
glance_host = glance-serverIP
glance_port = 9292
glance_protocol = http
glance_api_servers = glance-serverIP:port
auth_strategy = for Ironic to use Keystone to interact with Glance, specify keystone

[keystone_authtoken] configure if under DEFAULT, auth_strategy = keystone
auth_host = localhost
auth_uri = http://$CONTROLLER_ADMIN_NODE:5000/v2.0
identity_uri = http://$CONTROLLER_ADMIN_NODE:35357
admin_user = ironic
admin_password = ironic-service-password
admin_tenant_name = tenant
signing_dir = /var/lib/ironic/keystone-signing

[neutron]
auth_strategy = for Ironic to use Keystone to interact with Neutron, specify keystone

```

## 7. Ironic 用の SMF サービスが実行されていることを確認します。

```

ironic-localhost# svcs -a | grep rabbitmq
ironic-localhost# svcs -a | grep ironic

```

Ironic がスタンドアロンモードにある場合は、これらのサービスの手動での有効化が必要になることがあります。

```

ironic-localhost# svcadm enable rabbitmq
ironic-localhost# svcadm enable ironic-db
ironic-localhost# svcadm enable ironic-api ironic-conductor

```

## 8. (オプション) コマンド行ユーティリティーをテストします。

a. Ironic のグローバルシェル変数を設定します。

- `auth_strategy` が `noauth` に設定されている場合は、シェル変数を次のように設定します。

```
ironic-localhost# export OS_AUTH_TOKEN=fake-token
ironic-localhost# export IRONIC_URL=http://localhost:6385
```

- `auth_strategy` が `keystone` に設定されている場合は、シェル変数を次のように設定します。

Keystone サービスが Ironic とは異なるノード内に存在する場合、`OS_AUTH_URL` には、Keystone がインストールされている場所のホスト名または IP アドレスを指定します。

```
ironic-localhost# export OS_AUTH_URL=http://$CONTROLLER_ADMIN_NODE:5000/v2.0"
ironic-localhost# export OS_TENANT_NAME=service
ironic-localhost# export OS_USERNAME=ironic
ironic-localhost# export OS_PASSWORD=ironic-service-password
ironic-localhost# export IRONIC_URL=http://localhost:6385/
```

b. Ironic コマンド行を発行します。

```
# ironic driver-list
+-----+-----+
| Supported driver(s) | Active host(s) |
+-----+-----+
| solaris             | ironic         |
+-----+-----+
```

現在、`solaris` は Ironic で有効で、かつテストされている唯一のドライバです。ただし、`/etc/ironic/ironic.conf` ファイル内の `[enabled_drivers]` セクションの下に名前を含めることによって、このリストにドライバを追加できます。ドライバを追加したあと、`ironic-api` および `ironic-conductor` SMF サービスを再起動する必要があります。

## 概要: Ironic でのベアメタルの配備

Ironic `solaris` ドライバでは、Oracle Solaris の統合アーカイブ (UAR) 機能またはその Image Packaging System (IPS) を使用してベアメタルノードをプロビジョニングできます。ノードを作成するときは、構成可能なノード要素を使用して `solaris` ドライバに情報を渡します。

次の表に、ノードを作成するための要素の一覧を示します。`driver_info/archive_uri` 要素は、UAR ファイルを使用したノードのプロビジョニングに適用されます。この表内の残りの要素は、IPS を使用したノードのプロビジョニングに適用されます。



要素	説明	例
driver_info/ archive_uri	ベアメタルのプロビジョニングに使用される統合アーカイブの URI。	http://host.example.com/sol-11_3-x86.uar
driver_info/ ai_service	使用する AI サービスの名前	default-x86
driver_info/ publishers	プラス (+) 記号で区切られ、 <i>publisher-name@origin</i> という命名形式を使用している IPS パブリッシャーのリスト。	solaris@http://ipkg.us.oracle.com/ solarisN/dev+userland@http:// my.example.repo
driver_info/fmri	プラス (+) 記号で区切られた、インストールする PKG FMRI のリスト。	pkg:/group/system/solaris-small-server+pkg:/cloud/openstack/nova
driver_info/ install_profiles	インストール環境に適用する構成プロファイルの URI のリスト。これらの URI はプラス (+) 記号で区切られます。	http://host.example.com/profile1.xml +glance://glance-image
driver_info/ sc_profiles	インストールされたシステムに適用するシステム構成プロファイルの URI のリスト。これらの URI はプラス (+) 記号で区切られます。	http://host.example.com/profile1.xml +glance://glance-image
driver_info/ manifest	ベアメタルをプロビジョニングするために使用する AI マニフェストの URI。	http://host.example.com/my-manifest.xml
driver_info/ ipmi_address	プロビジョニングされるノードの ILOM に接続するシリアルコンソールの IP アドレスまたはホスト名。	192.168.2.200
driver_info/ ipmi_username	IPMI 接続のためのユーザー名。	root
driver_info/ ipmi_password	IPMI 接続のためのパスワード。	<i>password</i>

UAR ファイルを使用してノードをプロビジョニングする場合は、`driver_info/archive_uri` の情報を指定するだけで済みます。アーカイブ URI は、次のリスト内のいずれかのスキーマを使用できます。IPS を使用してノードをプロビジョニングすることを選択した場合は、プロファイルの指定に対して同じスキーマオプションが適用されます。

- file://
- http://
- https://
- glance://

**注記** - Glance は通常、インストールイメージの格納に使用されますが、ストレージへのプロファイルの追加も受け入れ可能です。

IPS を使用してノードをプロビジョニングする場合は、次の例に示すように、まず少なくともノードのアーキテクチャー用のデフォルトの AI サービスが存在することを確認してください。

```
# installadm list
Service Name          Status Arch  Type Alias Aliases Clients Profiles Manifests
```

```

-----
default-i386          on   i386  pkg  yes  0    0    0    1
default-sparc        on   i386  pkg  yes  0    0    0    1
ironic-x86           on   i386  pkg  no   0    0    0    1
ironic-sparc         on   i386  pkg  no   0    0    0    1

```

```

# installadm list -vcn ironic-x86
There are no clients configured for service 'ironic-x86'.

```

```

# installadm list -vmn ironic-x86
Service Name      Manifest Name Status  Criteria
-----
ironic-x86        orig_default  default none
ironic-sparc      orig_default  default none

```

IPS を使用してノードをプロビジョニングする場合、`driver_info/ai_service` の指定はオプションです。AI サービスの名前を省略した場合は、そのノードのアーキテクチャー用のデフォルトの AI サービスが使用されます。

`driver_info/fmri` にカスタムパッケージを指定する場合は、`driver_info/publishers` 要素にパブリッシャーも指定する必要があります。

## Ironic を使用したベアメタルの配備

次のタスクでは、ノードの作成から実際の配備までの基本的な手順が提供されます。

Oracle Solaris の統合アーカイブ機能の使用の詳細は、『*Oracle Solaris でのシステム復旧とクローン*』を参照してください。この本は、[オペレーティングシステムのドキュメント](#)内の使用している Oracle Solaris バージョンのライブラリにあります。

### ▼ UAR ファイルからベアメタルを配備する方法

次の手順を使用する具体的な例については、[例6「UAR ファイルを使用したノードの配備」](#)を参照してください。

始める前に 使用する UAR ファイルがすでに存在することを確認してください。

#### 1. Ironic コマンド行ユーティリティの使用を容易にするために、次の変数を作成します。

- IP - ノードの ILOM に接続するために使用される IP アドレス。
- USER - 通常、ユーザーは root です。
- PASS - root のパスワードです。

- HOST\_MAC - システムの MAC アドレスです。

## 2. Ironic ノードを作成します。

```
# ironic node-create options
```

ノードの UUID を含む、新しいノードのプロパティが表示されます。

## 3. ノードの UUID を容易に参照するための変数を作成します。

## 4. インストールする UAR の場所を指定することによってノードを更新します。

```
# ironic node-update options
```

## 5. このノードに関連付けられたポートを作成します。

```
# ironic port-create options
```

## 6. (オプション) ノードに対して指定したフィールドを検証します。

```
# ironic node-validate options
```

## 7. ノードをプロビジョニングします。

```
# ironic node-set-provision-state options
```

## 8. (オプション) 配備のステータスを表示します。

```
# ironic node-show options
```

---

注記 - このコマンドをプロビジョニングの完了後ではなく、このプロセスの進行中に実行した場合は出力が異なります。

---

### 例 6 UAR ファイルを使用したノードの配備

この例では、次のように仮定しています。

#### ■ ノードをホストするマシンの基本的な事項

- ホスト名: mynewnode.example.com
- アーキテクチャー: x86
- IP アドレス: 1.1.1.1
- MAC アドレス: 01:02:03:04:05:06

#### ■ ILOM ホストの基本的な事項

- ホスト名: mynewnode-aa.example.com
- IP アドレス: 2.2.2.2

- ユーザー: root
- パスワード: password
- UAR ファイルの名前: myuar.server/sol11-3-x86.uar

```
# export ILOM_IP=2.2.2.2
# export ILOM_USER=root
# export ILOM_PASS=password
# export HOST_MAC=01:02:03:04:05:06
```

```
# ironic node-create -d solaris -i ipmi_address=$ILOM_IP \
-i ipmi_username=$ILOM_USER -i ipmi_password=$ILOM_PASS
```

```
+-----+
| Property      | Value                                     |
+-----+-----+
| uuid          | 4eacbfde-4977-4d8c-8043-8cbe8f876187    |
| driver_info   | {u'ipmi_address': u'2.2.2.2', u'ipmi_username': u'root', |
|               | u'ipmi_password': u'password'}         |
| extra        | {}                                       |
| driver       | solaris                                 |
| chassis_uuid | None                                    |
| properties   | {}                                       |
+-----+-----+
```

ノードが作成されま

```
# export NODE=4eacbfde-4977-4d8c-8043-8cbe8f876187
```

UUID が保存されま

```
# ironic node-update $NODE \
add driver_info/archive_uri=http://myuar.server/sol11-3-x86.uar
```

```
+-----+-----+
| Property      | Value                                     |
+-----+-----+
| instance_uuid | None                                    |
| target_power_state | None                                    |
| properties   | {}                                       |
| maintenance   | False                                  |
| driver_info   | {u'archive_uri': u'http://myuar.server/sol11-3-x86.uar', |
|               | u'ipmi_address': u'2.2.2.2', u'ipmi_username': u'root', |
|               | u'ipmi_password': u'password'}         |
| extra        | {}                                       |
| last_error    | None                                    |
| created_at    | 2014-10-03T15:38:43+00:00              |
| target_provision_state | None                                    |
| driver       | solaris                                 |
| updated_at    | 2014-10-03T15:53:04+00:00              |
| instance_info |                                           |
| chassis_uuid | None                                    |
| provision_state | None                                    |
| reservation   | None                                    |
| power_state   | None                                    |
| console_enabled | False                                  |
| uuid         | 4eacbfde-4977-4d8c-8043-8cbe8f876187    |
+-----+-----+
```

```
# ironic port-create -n $NODE -a $HOST_MAC
+-----+
| Property | Value |
+-----+
| node_uuid | 4eacbfde-4977-4d8c-8043-8cbe8f876187 |
| extra      | {} |
| uuid       | 4c765ab0-2529-4463-a51b-e5544dd15a32 |
| address    | 01:02:03:04:05:06 |
+-----+

# ironic node-validate $NODE
+-----+
| Interface | Result | Reason |
+-----+
| console   | None   | not supported |
| deploy    | True   | |
| management | True  | |
| power     | True   | |
+-----+

# ironic node-set-provision-state $NODE active ノードがプロビジョニングされます

# ironic node-show $NODE
+-----+
| Property          | Value |
+-----+
| instance_uuid     | None |
| target_power_state | None |
| properties         | {} |
| maintenance       | False |
| driver_info        | {u'archive_uri': u'http://myuar.server/sol11-3-x86.uar', |
|                    | u'ipmi_address': u'2.2.2.2', u'ipmi_username': u'root', |
|                    | u'ipmi_password': u'password'} |
| extra             | {} |
| last_error        | None |
| created_at        | 2014-10-03T15:38:43+00:00 |
| target_provision_state | deploy_complete |
| driver            | solaris |
| updated_at        | 2014-10-03T15:53:04+00:00 |
| instance_info     | |
| chassis_uuid      | None |
| provision_state    | active |
| reservation       | None |
| power_state       | power on |
| console_enabled    | False |
| uuid              | 4eacbfde-4977-4d8c-8043-8cbe8f876187 |
+-----+
```

ironic node-show コマンドをプロビジョニングの進行中に発行した場合、provision\_state は active ではなく、別のステータスを示します。

## ▼ ノードを廃棄する方法

この手順は、プロビジョニング操作が失敗した場合にも使用します。

1. ノードを削除された状態に設定します。

```
# ironic node-set-provision-state $NODE deleted
```

2. ノードの情報を表示します。

次の例で、アスタリスク (\*) が付いたプロパティはノードの削除された状態を示しています。

```
# ironic node-show $NODE
+-----+-----+-----+
| Property          | Value                                     |
+-----+-----+-----+
| instance_uuid     | None                                     |
| target_power_state | None                                     |
| properties         | {}                                       |
| maintenance       | True                                    | *
| driver_info       | {u'archive_uri': u'http://myuar.server/sol11-3-x86.uar', |
|                   | u'ipmi_address': u'2.2.2.2', u'ipmi_username': u'root', |
|                   | u'ipmi_password': u'password'}        |
| extra             | {}                                       |
| last_error        | None                                     |
| created_at        | 2014-10-03T15:38:43+00:00              |
| target_provision_state | None                                    | *
| driver            | solaris                                 |
| updated_at        | 2014-10-03T15:53:04+00:00              |
| instance_info     | {}                                       |
| chassis_uuid      | None                                    | *
| provision_state   | None                                    | *
| reservation       | None                                    |
| power_state       | power off                               | *
| console_enabled   | False                                   |
| uuid              | 4eacbfde-4977-4d8c-8043-8cbe8f876187   |
+-----+-----+-----+
```

3. 失敗したプロビジョニングプロセスをトラブルシューティングしている場合は、ノードのフィールドを検証します。

ノードに対して指定された driver\_info 要素のいずれかに関する問題は、Reason 列の下で識別されます。

```
# ironic node-validate $NODE
+-----+-----+-----+
| Interface | Result | Reason          |
+-----+-----+-----+
| console   | None   | not supported   |
| deploy    | True   |                 |
| management | True  |                 |
| power     | True   |                 |
+-----+-----+-----+
```

+-----+-----+-----+





## OpenStack のトラブルシューティング

---

この章では、現在のリリースに関連する OpenStack の問題について説明します。また、構成中に発生する可能性のある基本的な問題を解決するためのトラブルシューティングのヒントや回避方法も提供します。この章の内容は次のとおりです。

- 113 ページの「コマンド行ヘルプの取得」
- 114 ページの「既知の制限事項」
- 115 ページの「ログファイルの調査」
- 117 ページの「問題の調査および解決」
- 124 ページの「デバッグに関する一般的なヒントとコツ」
- 124 ページの「役立つサイト」

### コマンド行ヘルプの取得

OpenStack では、コマンドは OpenStack コンポーネントに対応します。たとえば、`nova` コマンドは計算操作に、`cinder` はストレージに、`neutron` はネットワークにそれぞれ適用されます。

これらのコマンドの使用に関するヘルプ (正しい構文、サポートされるサブコマンド、可能なオプションなど) を取得するには、`command-component help` コマンド (`nova help` や `neutron help` など) を使用します。`grep` コマンドを使用して、`command-component` コマンドとともに使用する可能なサブコマンドのリストをフィルタできます。たとえば、ルーターに関連した `neutron` サブコマンドを一覧表示するには、次のコマンドを入力します。

```
# neutron help | grep router
l3-agent-list-hosting-router List L3 agents hosting a router.
l3-agent-router-add        Add a router to a L3 agent.
l3-agent-router-remove     Remove a router from a L3 agent.
net-gateway-connect        Add an internal network interface to a router.
router-create              Create a router for a given tenant.
router-delete              Delete a given router.
router-gateway-clear       Remove an external network gateway from a router.
router-gateway-set         Set the external network gateway for a router.
router-interface-add       Add an internal network interface to a router.
router-interface-delete    Remove an internal network interface from a router.
router-list                List routers that belong to a given tenant.
```

<code>router-list-on-l3-agent</code>	List the routers on a L3 agent.
<code>router-port-list</code>	List ports that belong to a given tenant, with specified router.
<code>router-show</code>	Show information of a given router.
<code>router-update</code>	Update router's information.

次に、あるサブコマンド (クラウド内のルーターを識別する `router-list` など) に関する具体的な詳細を取得するには、次のコマンドを入力します。

```
# neutron help router-list
```

## 既知の制限事項

次に、Oracle Solaris 11.3 の OpenStack (Juno) に関する既知の問題を示します。

- 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 Oracle 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 11.3 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-backup サービスは、cinder パッケージをインストールするときにインストールされます。ただし、46 ページの「ストレージノードの構成」で説明されているようなデフォルトの Cinder 配備では、このサービスは現在 backup\_volume に対して機能しません。
- 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、UploadToImage はサポートされていません。
- /etc/neutron/l3\_agent.ini ファイル内の external\_network\_dataink 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\_dataink 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
```

## ログファイルの調査

SMF サービスとさまざまな Oracle 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。

OpenStack は通常は複数のシステムにインストールされるため、確認が必要なログファイルもさまざまな場所にある必要があります。より体系的なトラブルシューティングについては、ノードごとにログを検査します。

SMF サービスに関する問題を解決する場合の一般的なヘルプについては、使用しているオペレーティングシステムのリリースに該当する『*Oracle Solaris*でのシステムサービスの管理』にあるトラブルシューティングのセクションを参照してください。<http://docs.oracle.com/en/operating-systems/> 内の該当するライブラリに移動します。

サービスを表示するには、適切な権限があることを確認してください。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

トラブルシューティングするには、次の一般的なコマンドを使用します。

- 特定のノードで実行中の OpenStack サービスを見つけるには:

```
# svcs -a | grep openstack
```

- 保守モードである可能性のあるサービスを一覧表示するには:

```
# svcs -x
svc:/application/openstack/swift/swift-replicator-rsync:
    default (OpenStack Swift Replication Service)
State: maintenance since Fri May 22 04:06:11 2015
Reason: Start method exited with $SMF_EXIT_ERR_FATAL.
    See: http://support.oracle.com/msg/SMF-8000-KS
    See: rsync(1)
    See: rsyncd.conf(5)
    See: /var/svc/log/application-openstack-swift-swift-replicator-rsync:default.log
Impact: This service is not running.
```

サービスが保守モードにある場合は、サービスのログファイルを確認してください。

- 特定の OpenStack サービスのログを特定するには:

```
# svcs -L openstack-service
```

例:

```
# svcs -L neutron-server
/var/svc/log/application-openstack-neutron-neutron-server:default.log
```

適切な権限があれば、`-Lv` のようにオプションを組み合わせることで、サービスのログを一覧表示して確認できます。

- 特定のログに記録されているエラーのインスタンスをただちに識別するには、`grep` などの一般的な UNIX コマンドを使用できます。

```
# grep keyword `svcs -L openstack-service`
```

`error`、`warning`、およびその他の重要なキーワードの出現箇所を検索して、エラーメッセージを直接読み取ることができます。

- ネットワーキングの問題をトラブルシューティングするときに `EVS` プロパティを確認するには、`evsadm show-prop` などのさまざまな `evsadm` サブコマンドを使用します。
- OpenStack の `list` コマンド行は、コンポーネントのステータスに関する情報を提供できます。たとえば `nova list` はコンピュータノードのアクティブなインスタンスを表示

し、`nova image-list` は使用可能なイメージとそのステータスを表示します。サンプル出力は、59 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。

多くの場合、次のログにはトラブルシューティングに役立つ情報が含まれています。

- `nova-compute`
- `nova-scheduler`
- `cinder-scheduler`
- `neutron-server`

SMF サービスログファイルだけでなく、`/var/log` ディレクトリ内のログも確認できます。その他の Oracle Solaris プロセスと同様に、OpenStack サービスも専用のログファイルを `/var/log/openstack-service` ディレクトリ内に生成します。

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の問題は、`/var/log/zones` ディレクトリに記録されることがあります。メッセンジングのログは、`/var/log/rabbitmq/rabbit@hostname.log` として格納されます。

ほとんどの OpenStack 構成ファイルは、`/etc` ディレクトリの 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。Nova のものは `/etc/nova` にあります。ほかも同様です。サービスの構成ファイルで次のパラメータを設定またはコメント解除することで、これらのファイルを特定のサービスのトラブルシューティングに使用できます。

- `debug=true`
- `verbose=true`

これらのパラメータによって、その構成ファイルによって影響を受ける操作からさらに多くの出力を表示できます。<http://www.oracle.com/technetwork/articles/servers-storage-admin/getting-started-openstack-os11-2-2195380.html>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。

---

**注記** - 個々の OpenStack サービスコマンドは、`--debug` オプションも使用できます。このオプションは、構成ファイルで `debug=true` を設定することと等価です。

---

## 問題の調査および解決

このセクションでは、OpenStack をインストールして構成するときに発生する可能性のあるいくつかの問題について説明します。

次の例は、ダッシュボードに関するエラーを示しています。

```
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.
```

これらのメッセージは、RSA ホスト鍵が変更されてすべてのコンポーネントに伝播していないことを示している可能性があります。RSA 鍵の構成の詳細は、「[Neutron をインストールし、構成する方法](#)」、および「[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。

次のエラーレポートが nova-scheduler ログに含まれている可能性があります。

```
controller# grep error `svcs -L nova-scheduler`
2014-12-03 12:49:19.271 3475 TRACE
nova.openstack.common.rpc.common error: [Errno 32] Broken pipe
```

パイプ切断エラーは、通常、1 つの OpenStack サービスをリフレッシュし、その他はリフレッシュしないときに報告されます。ノードで構成ファイルに変更を加えた場合は、ノードのすべてのサービスをリフレッシュします。次のコマンドは、オンラインサービスであるもののリフレッシュが必要なサービスを再起動します。

```
controller# svcs `*openstack` | grep online \
| awk -e '{print $3}' | xargs svcadm restart
```

リソース不足が原因のエラーである可能性もあります。VM インスタンスの作成時に、nova-compute ログは次のようなメッセージを表示することがあります。

```
[abc-123-def-456] Build of instance
abc-123-def-456 aborted: Image
xyz-987-uvw-654 is unacceptable: Image query failed.
Possibly invalid or corrupt. Log file location: controller:/tmp/archive_log.4249
```

さらに、ログに out of space/storage が示されます。システムのリソースを表示するには、top コマンドを使用します。システムのメモリーが 1G バイト未満のときは、さらに追加が必要ながあります。

## 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、[57 ページの「Glance リポジトリ用のイメージの準備」](#)を参照してください。

## VM インスタンスのプロパティ値がゾーンのプロパティ値と一致しない

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 Oracle 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 Oracle 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 <code>nova list</code> 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)。 <code>zoneadm list</code> 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、 <code>nova show</code> コマンドを使用します。 <code>nova show</code> の出力の中で、OS-EXT-SRV-ATTR:instance_name プロパティの値はゾーンの名前、name プロパティの値は VM インスタンスの名前です。
UUID	Horizon または <code>nova show</code> コマンドで表示される VM インスタンスの UUID は、 <code>zoneadm list -p</code> コマンドで表示される同じゾーンの UUID と一致しません。 <code>zoneadm</code> 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。 <code>psrinfo</code> 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 <code>prtconf</code> 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プレーバが指定しているメモリー量を表示します。 <code>prtconf</code> 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。

## 資格に関する問題

場合によっては、サービスコマンドの発行を妨げる、正しくない資格に関連したエラーメッセージが表示されることがあります。たとえば、glance コマンドを発行すると、次のエラーメッセージが生成されることがあります。

```
Invalid OpenStack Identity credentials.
```

このメッセージの根本原因は、そのつど異なる可能性があります。そのため、ログを検査して考えられる原因を限定する必要があります。例として glance サービスの場合は、Glance SMF サービスログの内容を参照してください。/var/log/glance/api.log で、次が報告されている可能性があります。

```
WARNING keystonemiddleware.auth_token [-] Authorization failed for token
```

glance 構成ファイルで Debug = True および Verbose = True を設定している場合は、/var/svc/log/application-openstack-glance-glance-api:default.log で、次のような、より詳細な情報が提供されます。

```
DEBUG keystonemiddleware.auth_token [-] Received request from user:
  user_id None, project_id None, roles None service: user_id None,
  project_id None, roles None
__call__ /usr/lib/python2.7/vendor-packages/keystonemiddleware/auth_token.py:821
```

問題を解決するために、次の領域を調査できます。

- サービス構成ファイルをチェックして、関連するパラメータが正しく定義されていることを確認します。
- サービスのグローバルシェル変数が正しいことを確認します。たとえば、Glance サービスの場合は、次の変数が設定されているべきです。
  - OS\_USERNAME=glance
  - OS\_PASSWORD=service-password
  - OS\_TENANT\_NAME=service
  - OS\_AUTH\_URL=http://\$CONTROLLER\_ADMIN\_NODE:5000/v2.0

報告されたエラーメッセージでコマンドが引き続き失敗する場合は、新しい資格を生成するためにサービスユーザーを作成し直すことが必要になる可能性があります。次の例について調べてみます。

```
# export OS_USERNAME=keystone
# export OS_PASSWORD=service-password
# export OS_TENANT_NAME=service
# export OS_AUTH_URL=http://$CONTROLLER_ADMIN_NODE:5000/v2.0

# keystone user-list
```

このコマンドの出力から、破損したサービスユーザーの ID 番号を書きとめます。そのユーザーを削除し、正しい資格を使用して作成し直します。



```
# keystone user-delete user-ID

# keystone user-create --name glance --password service-password
# keystone user-role-add --user=glance --tenant=service --role=admin
```

## Horizon 関連の問題

VM インスタンスを起動したあと、Horizon ダッシュボードにアクセスできなくなることがあり、「404 Not Found」エラーメッセージが表示されます。Apache サービスのログには、次のエントリが含まれます。

```
Oct 13 16:13:53 Executing start method (" /lib/svc/method/http-apache24 start"). ]
Apache version is 2.4
(125) Address already in use: AH000/2: make_sock: could not bind to address [::]:80
Oct 13 16:13:55 Method "start" exited with status 0. ]
```

このログは、ポート 80 がビジー状態にあるため、アドレスをこのポートにバインドできないことを示しています。

Juno バージョンから、OpenStack では、以前の OpenStack バージョンの Apache 2.2 の代わりに Apache 2.4 を使用しています。正しい Apache バージョンが有効になり、そのポートで待機していることを確認してください。

ポートを解放するには、次の手順に従います。

1. 現在そのポート上で待機しているプロセス ID を取得します。

```
# netstat -uan -f inet | grep "*.*80"
```

この手順では、設定で IPv4 アドレスが使用され、ポート 80 を保持しているプロセスがそれらのすべてのアドレス上で待機していることを前提にしています。プロセスが IPv6 トラフィックを待機している場合、このコマンドでは何も結果が得られない可能性があります。

2. プロセス ID に基づいて、実際のプロセスまたはサービスを識別します。次のいずれかのコマンドを使用できます。

```
# svcs -p | egrep "online|pid http" | ggrep -B1 pid
```

または

```
# ps -lf -p pid
```

3. 間違った Apache バージョンがポートを使用している場合は、そのサービスを無効にします。
4. Juno バージョンの正しい Apache バージョンを有効にします。  
正しい Apache バージョンが保守モードにある場合は、それを有効にする前に、まずサービスをクリアします。

次の例は、ポート 80 を解放し、正しい Apache バージョンに切り替える方法を示しています。

```
# netstat -uan -f inet | grep "*.*80"
*.80      *.*      root      5560 httpd    0      0 128000    0 LISTEN
*.8080    *.*      webservd  1124 java    0      0 128000    0 LISTEN
*.8009    *.*      webservd  1124 java    0      0 128000    0 LISTEN

# svcs -p | egrep "online|5560 http" | ggrep -B1 5560
online    Aug_31   svc:/network/http:apache22
          Sep_09   5560 httpd

# svcadm disable apache22
# svcadm clear apache24
# svcadm enable apache24
```

## Heat 関連の問題

**注記** - この問題は、Oracle Solaris 11.3 SRU 2 リリースまたは以前のリリースを使用している場合にのみ適用されます。

Heat スタックを作成するとき、次の例のような通常の `heat stack-create` コマンドを使用すると失敗することがあります。

```
# heat stack-create -f heat-template -P parameters stack-name
```

ここで、`heat-template` は、有効に定義された HOT テンプレートです。

`neutron-server` ログには、次の例のようなエントリが含まれます。この例では、問題に関連する情報のみを含む編集されたログが示されています。

```
stack_name          | stack2          |
stack_owner         | admin           |
stack_status        | CREATE_FAILED   |
stack_status_reason | Resource CREATE failed: InternalServerError: Request |
                   | Failed: internal server error while processing your |
                   | request         |
```

この問題は、ネットワーク用の DHCP ポートと通常のポートが同時に作成されたため、タイムアウト条件が発生したことにあります。回避方法として、サブネットの DHCP を無効にできます。より適切な解決方法は、新しい Oracle Solaris 11.3 SRU リリースにアップグレードすることです。

## スケーラビリティの問題

デフォルトでは、RabbitMQ には 255 というファイル記述子制限があります。この制限により、コンピュータノードをいくつか作成したあと、クラウドをそれ以上拡張することが簡単に妨げられ

てしまいます。このブロックを回避するには、`/etc/rabbitmq/rabbitmq-env.conf` ファイル内の制限値を増やします。

```
# Increase soft limit on file descriptors for RabbitMQ
ulimit -n 8192
```

## 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、[124 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 remove-vport vport
```

## デバッグに関する一般的なヒントとコツ

次の一般的なヒントが、OpenStack での問題のトラブルシューティングを開始するのに役立つことがあります。

- 問題の診断に役立つように、各種の構成ファイルで `debug = true` および `verbose = true` を設定します。一部の構成ファイルでは、コメントアウトされており、詳細ロギングを有効にするために切り替えることができるプレースホルダ値を探することができます。
- コンポーネントの構成ファイルを変更した場合は、変更を有効にするために、そのコンポーネントのサービスを再起動してください。
- SMF ログの情報を提供する `tail -30 `svcs -L service-name`` コマンドを使用します。デバッグだけでなく詳細ロギングを有効にした場合は、`tail` コマンドで指定する行数を増やすことが必要になる可能性があります。
- Horizon プロセスは Apache を使用します。そのため、Horizon を診断するときは、`/etc/openstack_dashboard/local_settings.py` ファイル内の `debug = True` を有効にしてください。すると、Web ページ上で Django エラーが生成されます。
- Nova コンポーネントは、Oracle Solaris ゾーンの上に構築されています。そのため、Nova の問題をトラブルシューティングするために `/var/log/zones` 内のログを参照することもできます。

## 役立つサイト

OpenStack のさまざまな問題を解決するためのトラブルシューティングのヒントについては、次のサイトを参照してください。

- <https://blogs.oracle.com/openstack/>

- <https://ask.openstack.org/en/questions/>
- <https://raymi.org/s/tags/openstack.html>





## 一般的な OpenStack 構成ファイルおよびサービス

---

この付録には、コア OpenStack コンポーネントの標準的な構成ファイル、および OpenStack SMF サービスを一覧表示します。

### 構成ファイル

#### Cinder ファイル

- /etc/cinder/api\_paste.ini
- /etc/cinder/cinder.conf

#### Glance ファイル

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

#### Keystone ファイル

- /etc/keystone/keystone.conf
- /etc/keystone/keystone-paste.ini

## Neutron ファイル

- /etc/neutron/neutron.conf
- /etc/neutron/dhcp\_agent.ini
- /etc/neutron/l3\_agent.ini
- /etc/neutron/api-paste.ini
- /etc/neutron/metadata\_agent.ini
- /etc/neutron/plugins/evs/evs\_plugin.ini

## Nova ファイル

- /etc/nova/nova.conf
- /etc/nova/api\_paste.ini

## Horizon ファイル

- /etc/openstack\_dashboard/local\_settings.py
  - /etc/apache2/2.4/httpd.conf
  - /etc/apache2/2.4/conf.d/openstack-dashboard-http.conf
- または
- /etc/apache2/2.4/conf.d/openstack-dashboard-tls.conf

## Swift ファイル

- /etc/swift/swift.conf
- /etc/swift/account-server.conf
- /etc/swift/container-server.conf
- /etc/swift/object-server.conf
- /etc/swift/proxy-server.conf
- /etc/swift/rsyncd.conf



## OpenStack SMF サービス

### Cinder

```
svc:/application/openstack/cinder/cinder-db:default
svc:/application/openstack/cinder/cinder-backup:default
svc:/application/openstack/cinder/cinder-scheduler:default
svc:/application/openstack/cinder/cinder-api:default
svc:/application/openstack/cinder/cinder-volume:setup
svc:/application/openstack/cinder/cinder-volume:default
```

### Glance

```
svc:/application/openstack/glance/glance-db:default
svc:/application/openstack/glance/glance-registry:default
svc:/application/openstack/glance/glance-scrubber:default
svc:/application/openstack/glance/glance-api:default
```

### Keystone

```
svc:/application/openstack/keystone:default
```

### Neutron

```
svc:/application/openstack/neutron/neutron-server:default
svc:/application/openstack/neutron/neutron-dhcp-agent:default
svc:/application/openstack/neutron/neutron-metadata-agent:default
svc:/application/openstack/neutron/neutron-l3-agent:default
```

### Nova \*

```
svc:/application/openstack/nova/nova-objectstore:default
svc:/application/openstack/nova/nova-consoleauth:default
svc:/application/openstack/nova/nova-novncproxy:default
svc:/application/openstack/nova/nova-api-metadata:default
svc:/application/openstack/nova/nova-api-ec2:default
svc:/application/openstack/nova/nova-api-osapi-compute:default
svc:/application/openstack/nova/nova-conductor:default
svc:/application/openstack/nova/nova-cert:default
svc:/application/openstack/nova/nova-compute:default
svc:/application/openstack/nova/nova-scheduler:default
```

\* コンピュートノードの設定によっては、ほかの Nova サービスも一覧表示される可能性があります。

## Swift

```
svc:/application/openstack/swift/swift-object-expirer:default
svc:/application/openstack/swift/swift-account-reaper:default
svc:/application/openstack/swift/swift-container-replicator:default
svc:/application/openstack/swift/swift-account-replicator:default
svc:/application/openstack/swift/swift-object-auditor:default
svc:/application/openstack/swift/swift-container-updater:default
svc:/application/openstack/swift/swift-container-sync:default
svc:/application/openstack/swift/swift-object-updater:default
svc:/application/openstack/swift/swift-account-auditor:default
svc:/application/openstack/swift/swift-replicator-rsync:default
svc:/application/openstack/swift/swift-container-auditor:default
svc:/application/openstack/swift/swift-object-replicator:default
svc:/application/openstack/swift/swift-container-reconciler:default
svc:/application/openstack/swift/swift-container-server:default
svc:/application/openstack/swift/swift-object-server:default
svc:/application/openstack/swift/swift-proxy-server:default
svc:/application/openstack/swift/swift-account-server:default
```

# 索引

---

## あ

イメージ, 22, 57  
参照 VM インスタンス  
イメージキャッシュ, 57  
情報の表示, 59  
スナップショット, 58  
バックアップ, 58  
レジストリサーバー, 57  
レスキュー, 58  
イメージを作成およびアップロードするためのスクリプト, 61  
インスタンス 参照 VM インスタンス  
インスタンステンプレート 参照 フレーバ  
エラスティック仮想スイッチ  
evsadm コマンド, 52  
エラスティック仮想スイッチ (EVS)  
SSH 鍵, 43  
構成, 42

## か

仮想マシン (VM) 参照 VM インスタンス  
クラウド仮想マシン 参照 VM インスタンス  
コンソールアクセス, 44  
コントローラノード, 23  
構成, 30  
コンピュータノード, 23, 23  
参照 Nova  
回復, 82  
構成, 41

## さ

サービス管理機能 (SMF)  
ターゲットホストに必要なサービス, 89  
システム構成プロファイル 参照 SC プロファイル

シングルノード OpenStack インストール  
OVM Server for SPARC の使用, 24  
統合アーカイブの使用, 15  
スケラビリティ, 122  
ストレージノード, 23  
構成, 46  
複数のバックエンドホスト, 85  
スナップショット, 58  
セキュアネットワークアドレス変換 (SNAT), 55  
無効化, 55  
ゾーンコンソールログイン, 19  
ゾーンフレームワーク, 41

## た

ダッシュボード  
VM インスタンスの作成, 71  
イメージの表示, 22  
内部ネットワークの作成, 66  
フレーバの表示, 22, 77  
プロジェクトの表示, 22  
ログイン, 20  
データベース, 31  
テナント, 22 参照 プロジェクト  
デバッグオプション, 117  
テンプレート  
ハードウェア 参照 フレーバ  
統合アーカイブ  
OpenStack イメージの作成, 58  
シングルノード OpenStack インストール, 15, 18  
ベアメタル AI サービス, 17  
ダウンロード, 16

## な

ネットワーク

外部ネットワークの作成, 51, 54  
サブネット, 66  
内部, 66  
内部ネットワークを外部ネットワークに接続, 69  
フローティング IP アドレスの関連付け, 68  
ルーターの作成, 52  
ネットワークアドレス変換, 51

## は

ハードウェアテンプレート 参照 フレーバ  
フラットなネットワーク, 54  
フレーバ, 22, 76  
参照 VM インスタンス  
extra-specs プロパティ, 77  
プロパティの変更, 77  
フローティング IP アドレス, 68  
参照 ネットワーク  
プロジェクト, 22  
作成, 63  
ユーザーの追加, 63, 64  
ユーザー役割の変更, 64  
プロバイダルーター 参照 Neutron  
ベアメタル, 104  
参照 Ironic  
Ironic での配備, 104  
UAR と IPS の使用, 105  
UAR を使用した配備, 106  
構成可能なノード要素, 105  
廃棄, 110  
ベアメタルの配備 参照 ベアメタル

## ま

マルチノード OpenStack インストール  
3 ノードリファレンスアーキテクチャー, 23  
準備, 26  
メモリーの最適化, 28

## ら

ライブ移行, 80  
リモートストレージシステム, 85  
参照 Cinder

アクセス制御リスト (ACL), 89  
ターゲットとして有効にする, 89  
必要な SMF サービス, 89  
ルーター  
外部ネットワーク用の作成, 52  
レジストリサーバー, 57  
レスキューイメージ, 58

## A

Advanced Message Queuing Protocol (AMQP),  
31  
archiveadm コマンド, 58  
Automated Installer (AI), 17

## C

Cinder  
SAN のサポート, 85  
ZFSSA 用の iSCSI Cinder ドライバ, 90  
インストール, 37  
構成ファイル, 37, 46, 86, 93  
ターゲットストレージホスト, 85  
配備オプション, 85  
バックアップ, 114  
バックエンドストレージとしての ZFSSA の使用, 91  
ユーザーの権利プロファイル, 89  
リモートストレージシステムの構成, 85  
リモートホストでのアクセス制御リストの設定, 89  
Cinder での SAN のサポート, 85  
Cinder ユーザーの権利プロファイル, 89

## E

EVS  
evsadm コマンド, 54

## G

Glance, 57  
イメージ情報の表示, 59  
インストール, 33  
構成ファイル, 33  
作成およびアップロードスクリプト, 61

**H**

## Heat

- インストール, 96

## Horizon

- SSL アクセスの構成, 36

- フレーバの表示, 77

**I**

## Ironic

- /etc/ironic/ironic.conf ファイル, 102

- AI 構成, 100

- UAR からのベアメタルの配備, 106

- UAR と IPS の使用, 105

- インストールおよび構成, 100

- コンポーネント, 99

- サービス, 99

- ノードの廃棄, 110

- プロビジョニング中の障害, 110

- ベアメタルの配備, 104

**K**

## Keystone

- sample\_data.sh スクリプト, 27

- インストール, 32

- 構成ファイル, 33

**L**

- L3 エージェント, 51

- LDom, 24

**M**

- MySQL, 31

**N**

- NAT 参照 ネットワークアドレス変換

- neutron-l3-agent, 70

- neutron-l3-agent SMF サービス, 52

- Neutron

- L3 エージェント, 51, 70

- インストール, 38

- 外部ネットワーク, 53

- 構成ファイル, 38

- ルーター, 51

- ルーターの作成, 52

## Nova

- VM インスタンスの作成, 71

- イメージ情報の表示, 59

- インスタンスの移行, 80

- 構成, 41

- 構成ファイル, 35, 41

- コントローラノードへのインストール, 35

- コンピュータノードの回復, 82

- 退避, 82

## NTP (Network Time Protocol)

- クライアント, 30

- クライアント構成ファイル, 31

- サーバー, 29

- サーバー構成ファイル, 29

**O**

- OpenStack 統合アーカイブ 参照 統合アーカイブ

- OpenStack のインストール

- シングルノード構成, 15

- 統合アーカイブを使用した, 15

- 評価構成, 15

- マルチノード構成, 23

- OVN Server for SPARC, 24

**R**

- RabbitMQ, 31

- RAD, 41

- Remote Access Daemon (RAD), 41

**S**

- sample\_data.sh スクリプト, 27

- SC プロファイル, 58, 78

- SPARC, 24

- SQLite, 31

- SSH 鍵

EVS 構成, 43  
インスタンスの移行, 81  
Swift, 25  
構成ファイル, 48

set サブコマンド, 19

## T

TLS  
構成ファイル, 37

## U

user\_reserve\_hint\_pct, 28

## V

VM インスタンス, 14, 71  
参照 Nova  
移行, 80  
イメージ, 22, 57  
鍵ペア, 71  
作成, 71  
スナップショット, 58  
バックアップ, 58  
フレーバ, 21, 22, 76  
ユーザーの追加, 75  
レスキュー, 58  
ログイン, 75

## Z

ZFS Storage Appliance (ZFSSA), 90  
参照 Cinder  
OpenStack 用のバックエンドストレージ, 90  
ワークフローユーティリティ, 92  
zlogin コマンド, 19  
zoneadm コマンド  
boot サブコマンド, 19  
zonecfg:bootargs, 79  
zonecfg コマンド  
add サブコマンド, 19  
create サブコマンド, 19  
select サブコマンド, 19