Red Hat OpenStack Platform 11

アーキテクチャーガイド

製品、コンポーネント、アーキテクチャー例のご紹介

OpenStack Team

Red Hat OpenStack Platform 11 アーキテクチャーガイド

製品、コンポーネント、アーキテクチャー例のご紹介

OpenStack Team
rhos-docs@redhat.com

概要

本ガイドでは OpenStack のクラウドコンポーネントをご紹介し、OpenStack クラウドの設計に役立つ設計指針およびアーキテクチャーの例を記載しています。

目次

前書き

Red Hat OpenStack Platform は、Red Hat Enterprise Linux をベースとして、プライベートまたはパブリックの Infrastructure-as-a-Service (IaaS) クラウドを構築するための基盤を提供します。これにより、スケーラビリティーが高く、耐障害性に優れたプラットフォームをクラウド対応のワークロード開発にご利用いただくことができます。

Red Hat OpenStack Platform は、利用可能な物理ハードウェアをプライベート、パブリック、またはハイブリッドのクラウドプラットフォームに変換できるようにパッケージされています。これには以下のコンポーネントが含まれます。

- 完全に分散されたオブジェクトストレージ
- 永続的なブロックレベルのストレージ
- 仮想マシンのプロビジョニングエンジンおよびイメージストレージ
- 認証および承認メカニズム
- 統合されたネットワーク
- ユーザーおよび管理者がアクセス可能な Web ブラウザーベースのインターフェース

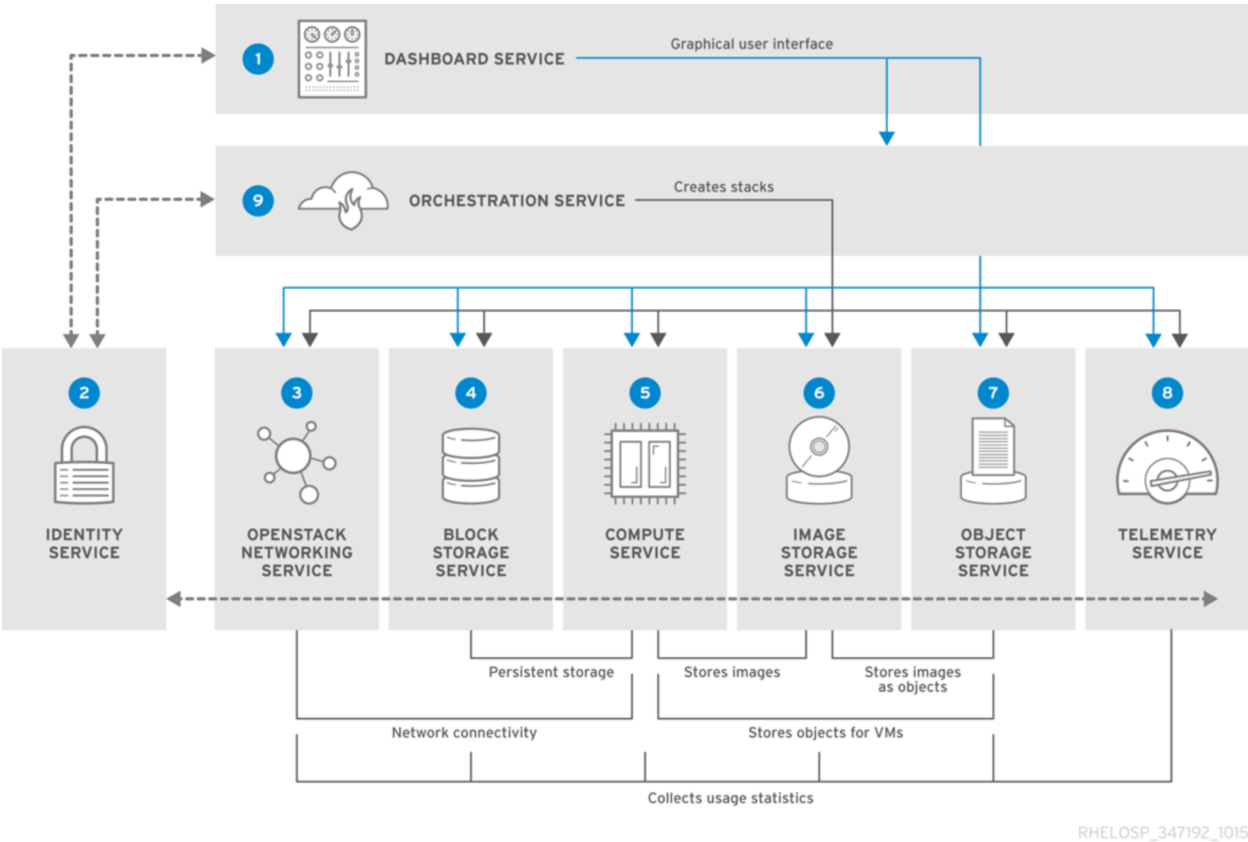
注記

- 本ガイドに記載するコンポーネントについての参考情報は、「5章 *デプロイメントの情報*」を参照してください。
- Red Hat OpenStack の全ドキュメントスイートは Red Hat OpenStack Platform の製品ドキュメント で参照してください。

第1章 コンポーネント

Red Hat OpenStack Platform IaaS クラウドは、コンピューティング、ストレージ、ネットワークのリソースを制御するために相互に対話するサービスのコレクションとして実装されます。クラウドは、Web ベースのダッシュボードまたはコマンドラインクライアントで管理されます。これにより、管理者は OpenStack リソースの制御、プロビジョニング、自動化を行うことができます。OpenStack は、クラウドの全ユーザーが利用できる豊富な API も提供しています。

以下の図は、OpenStack のコアサービスとそれらの相互関係の俯瞰的な概要を示しています。



以下の表には、上図に示した各コンポーネントについての簡単な説明と、それぞれのセクションへのリンクをまとめています。

表1.1 コアサービス

	サービス	コード	説明	リンク
1	Dashboard	horizon	OpenStack の各サービスの管理に使用する Web ブラウザーベースのダッシュボード	「OpenStack Dashboard (horizon)」
2	Identity	keystone	OpenStack サービスを認証および承認し、ユーザー/プロジェクト/ロールを管理する一元化されたサービス	「OpenStack Identity (keystone)」

	サービス	コード	説明	リンク
3	OpenStack Networking	neutron	OpenStack サービスのインターフェース間の接続性を提供します。	「OpenStack Networking (neutron)」
4	Block Storage	cinder	仮想マシン用の永続的な Block Storage ボリュームを管理します。	「OpenStack Block Storage (cinder)」
5	Compute	nova	ハイパーバイザーノードで実行されている仮想マシンの管理とプロビジョニングを行います。	「OpenStack Compute (nova)」
6	Image	glance	仮想マシンイメージやボリュームのスナップショットなどのリソースの保管に使用するレジストリーサービス	「OpenStack Image (glance)」
7	Object Storage	swift	ユーザーによるファイルおよび任意のデータの保管/取得を可能にします。	「OpenStack Object Storage (swift)」
8	Telemetry	ceilometer	クラウドリソースの計測値を提供します。	「OpenStack Telemetry (ceilometer)」
9	Orchestration	heat	リソーススタックの自動作成をサポートする、テンプレートベースのオーケストレーションエンジン	「OpenStack Orchestration (heat)」

各 OpenStack サービスには、Linux サービスおよびその他のコンポーネントの機能グループが含まれています。たとえば、**glance-api** および **glance-registry** Linux サービスは MariaDB データベースとともに Image サービスを実装します。OpenStack サービスに含まれるサードパーティーコンポーネントに関する詳しい情報は、「サードパーティーのコンポーネント」を参照してください。

その他のサービス:

- 「OpenStack Bare Metal Provisioning (ironic)」: さまざまなハードウェアベンダーの物理マシン (ベアメタル) のプロビジョニングを可能にします。
- 「OpenStack Data Processing (sahara)」: OpenStack で Hadoop クラスターのプロビジョニングと管理を行うことができるようになります。

1.1. ネットワーク

1.1.1. OpenStack Networking (neutron)

OpenStack Networking は、OpenStack クラウド内の仮想ネットワークインフラストラクチャーの作成と管理を処理します。インフラストラクチャー要素にはネットワーク、サブネット、ルーターなどが含まれます。また、ファイアウォールや仮想プライベートネットワーク (VPN) などの高度なサービスもデプロイすることができます。

OpenStack Networking は、クラウド管理者が、個々のサービスをどの物理システムで実行するかを決定する柔軟性を提供します。評価目的の場合には、全サービスデーモンを単一の物理ホストで実行することが可能です。また、各サービスに独自の物理ホストを使用したり、複数のホストにわたって複製して冗長化を行ったりすることもできます。

OpenStack Networking はソフトウェア定義型なので、新規 IP アドレスの作成や割り当てなど、変化するネットワークのニーズにリアルタイムで対応することができます。

OpenStack Networking には、次のような利点があります。

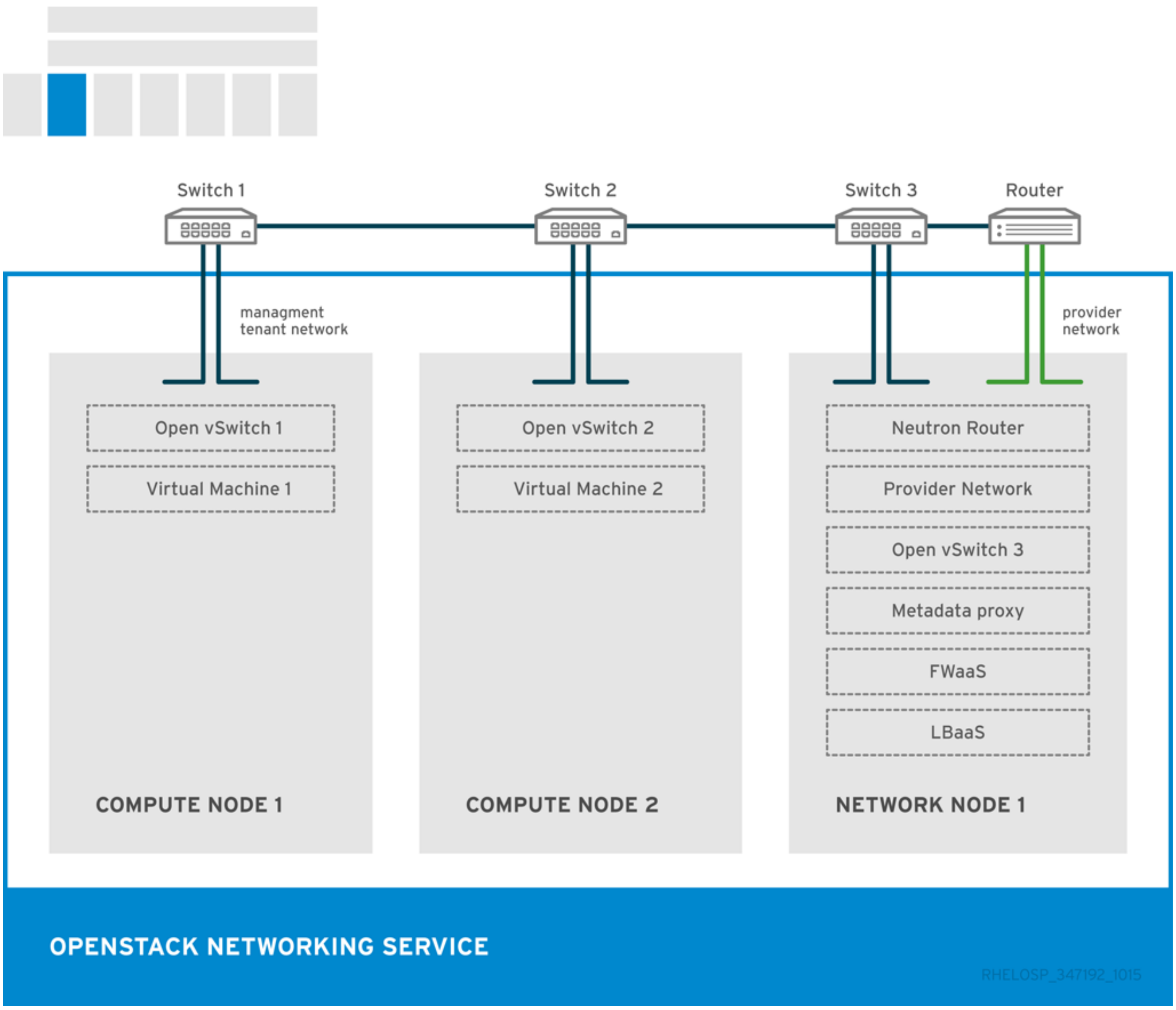
- ユーザーは、ネットワークの作成やトラフィックの制御を行ったり、サーバーおよびデバイスを単一または複数のネットワークに接続したりすることができます。
- ボリュームとテナンシーに適応可能な柔軟性の高いネットワークモデルを提供します。
- IP アドレスは、専用または Floating IP を使用することができます。Floating IP により、動的なトラフィックルーティングが可能となります。
- VLAN ネットワークを使用する場合には、最大で 4094 の VLAN (4094 のネットワーク) を使用することが可能です。この場合、4094 = 2^12 (この値から使用不可な 2 つを差し引いた) ネットワークアドレスです。これは、12 ビットのヘッダーで課せられる制限です。
- VXLAN トンネルベースのネットワークを使用する場合には、VNI (Virtual Network Identifier) は 24 ビットのヘッダーを使用することができます。これは、実質的には 1600 万の一意なアドレス/ネットワークを使用できることになります。

表1.2 OpenStack Networking のコンポーネント

コンポーネント	説明
ネットワークエージェント	各 OpenStack ノード上で稼働し、そのノードの仮想マシンとネットワークサービスのローカルネットワーク設定を行うサービス (例: Open vSwitch)
neutron-dhcp-agent	テナントネットワークに対して DHCP サービスを提供するエージェント
neutron-ml2	ネットワークドライバーを管理して、Open vSwitch や Ryu ネットワークなどのネットワークサービスのルーティングおよびスイッチングサービスを提供するプラグイン

コンポーネント	説明
neutron-server	ユーザー要求を管理し、Networking API を公開する Python デーモン。デフォルトのサーバー設定では、特定のネットワークメカニズムが装備されたプラグインを使用して Networking API を実装します。 **openvswitch** や **linuxbridge** などの特定のプラグインは、ネイティブの Linux ネットワークメカニズムを使用しますが、その他のプラグインは外部のデバイスや SDN コントローラーと連動します。
neutron	API にアクセスするためのコマンドラインクライアント

OpenStack Networking サービスおよびエージェントの配置は、ネットワーク要件によって異なります。以下の図には、コントローラーを使用しない、一般的なデプロイメントモデルを示しています。このモデルでは、専用の OpenStack Networking ノードとテナントネットワークを使用します。



上記の例は、以下のネットワークサービス構成を示しています。

- 2 つのコンピュートノードで Open vSwitch (ovs-agent) を実行し、1 つの OpenStack Networking ノードで、次のネットワーク機能を実行します。
  - L3 ルーティング

- DHCP
- FWaaS や LBaaS などのサービスを含む NAT

- コンピュートノードには、2 枚の物理ネットワークカードを装備します。一方はテナントトラフィックを処理し、もう一方は接続性を管理します。
- OpenStack Networking ノードには、プロバイダートラフィック専用に 3 枚目のネットワークカードを装備します。

1.2. ストレージ

「OpenStack Block Storage (cinder)」

「OpenStack Object Storage (swift)」

1.2.1. OpenStack Block Storage (cinder)

Block Storage サービスは、仮想ハードドライブの永続的なブロックストレージ管理機能を提供します。Block Storage により、ユーザーはブロックデバイスの作成/削除やサーバーへの Block Device の接続を管理することができます。

デバイスの実際の接続/切断は、Compute サービスとの統合により処理されます。分散ブロックストレージホストの処理には、リージョンとゾーンの両方を使用することができます。

Block Storage は、データベースストレージや、拡張可能なファイルシステムなど、パフォーマンスが要求されるシナリオに使用することができます。また、RAW ブロックレベルストレージにアクセス可能なサーバーとして使用することもできます。加えて、ボリュームのスナップショットを作成して、データの復元や新規ブロックストレージボリュームの作成も可能です。スナップショットは、ドライバーのサポートに依存します。

OpenStack Block Storage には、次のような利点があります。

- ボリュームとスナップショットの作成、一覧表示、削除
- 実行中の仮想マシンへのボリュームの接続/切断

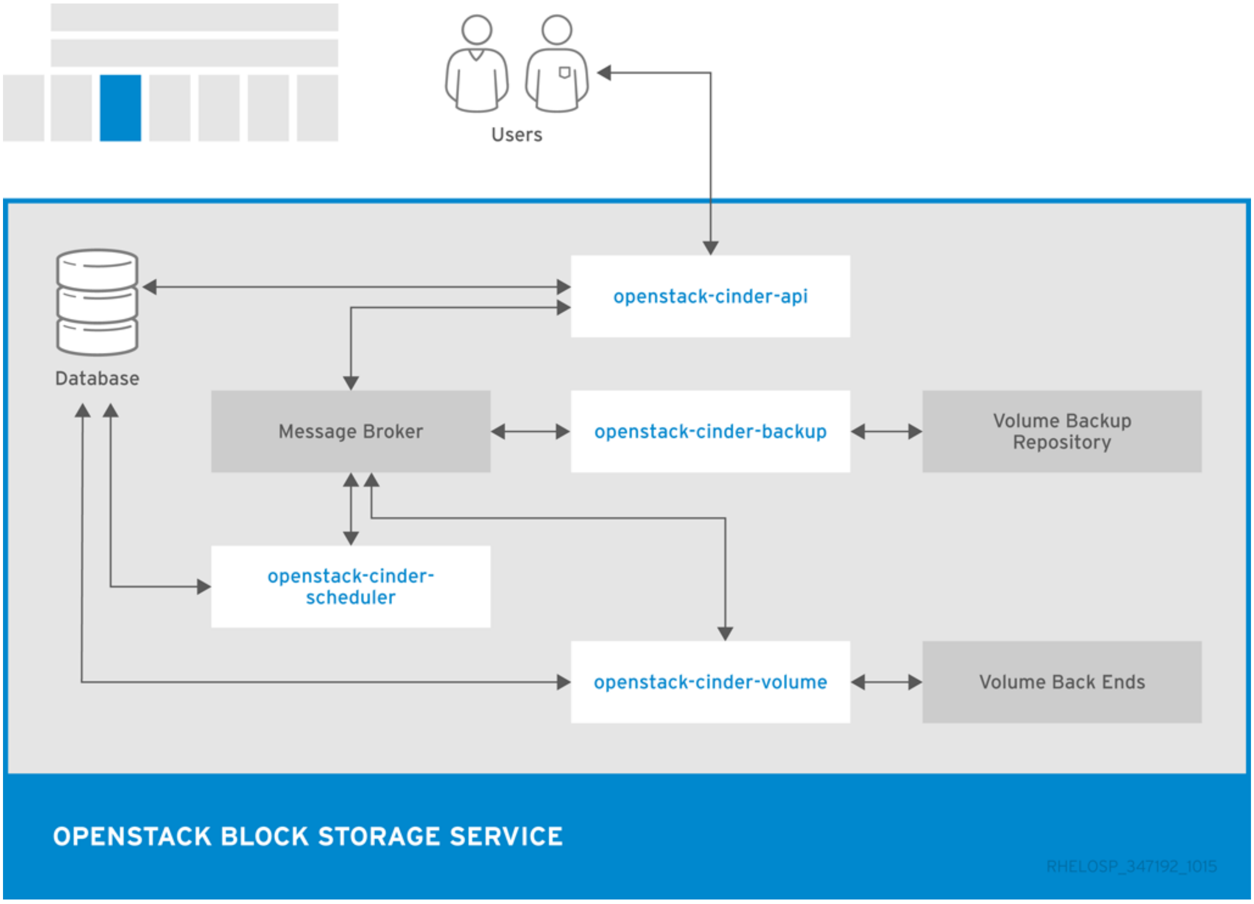
実稼働環境で Block Storage の主要なサービス (ボリューム、スケジューラー、API) を併置することは可能ですが、ボリュームサービスのインスタンスを複数デプロイして、API およびスケジューラーサービスのインスタンス (単一または複数) でそれらを管理する構成の方がより一般的です。

表1.3 Block Storage のコンポーネント

コンポーネント	説明
openstack-cinder-api	要求に応答し、メッセージキューに配置します。API サービスは Block Storage 要求のための HTTP エンドポイントを提供します。受信要求を受け取ると、API はアイデンティティー要件が満たされているかどうかを確認し、その要求を、必要とされる Block Storage のアクションを含むメッセージに変換します。このメッセージは、次にメッセージブローカーに送信され、他の Block Storage サービスによって処理されます。

コンポーネント	説明
openstack-cinder-backup	Block Storage ボリュームを外部のストレージリポジトリーにバックアップします。デフォルトでは、OpenStack は Object Storage サービスを使用してバックアップを保管します。Ceph または NFS のバックエンドをバックアップ用のストレージリポジトリーとして使用することも可能です。
openstack-cinder-scheduler	タスクをキューに割り当て、プロビジョニングするボリュームサーバーを決定します。スケジューラーサービスは、メッセージキューから要求を読み取り、要求されたアクションを実行する Block Storage ホストを判断します。スケジューラーは次に選択したホスト上の openstack-cinder-volume サービスと通信し、要求を処理します。
openstack-cinder-volume	仮想マシン用にストレージを指定します。ボリュームサービスは、ブロックストレージデバイスとの対話を管理します。スケジューラーからの要求を受け取ると、ボリュームサービスはボリュームの作成、変更、削除を行います。ボリュームサービスには、NFS、Red Hat Storage、Dell EqualLogic などのブロックストレージデバイスと対話するための複数のドライバーが同梱されています。
cinder	Block Storage API にアクセスするためのコマンドラインクライアント

以下の図には、Blocck Storage API、スケジューラー、ボリュームサービス、その他の OpenStack コンポーネントの関係を示しています。



1.2.2. OpenStack Object Storage (swift)

Object Storage サービスは、HTTP 経由でアクセス可能な、大量データ用のストレージシステムを提供します。ビデオ、イメージ、メールのメッセージ、ファイル、仮想マシンイメージなどの静的エンティティーをすべて保管することができます。オブジェクトは、各ファイルの拡張属性に保管されているメタデータとともに、下層のファイルシステムにバイナリーとして保管されます。

Object Storage の分散アーキテクチャーは、水平スケーリングに加えて、ソフトウェアベースのデータ複製を使用したフェイルオーバーのための冗長化をサポートします。このサービスは、非同期で結果整合性のある複製をサポートするので、複数のデータセンターで構成されるデプロイメントに使用することができます。

OpenStack Object Storage には、次のような利点があります。

- ストレージのレプリカにより、障害発生時にオブジェクトの状態が維持されます。最低でも 3 つ以上のレプリカが推奨されます。
- ストレージゾーンにより、レプリカがホストされます。ゾーンを使用することにより、任意のオブジェクトの各レプリカを個別に格納することができます。ゾーンは、個別のディスクドライブ、アレイ、サーバー、サーバーラック、あるいはデータセンター全体を指すこともあります。
- ストレージリージョンでゾーンをグループ化することができます。リージョンには、通常同じ地理的地域に配置されているサーバーやサーバーファームなどを含めることができます。リージョンには、Object Storage サービスをインストールしたシステムごとに個別の API エンドポイントがあり、サービスを分離することができます。

Object Storage は、データベースおよび設定ファイルとして機能するリング **.gz** ファイルを使用します。これらのファイルには、全ストレージデバイスの詳細情報と、保管されているエンティティーから各ファイルの物理的な場所へのマッピングが含まれているので、特定のデータの場所を決定するのに使用することができます。プロジェクト、アカウント、コンテナーサーバーにはそれ

ぞれ固有のリングファイルがあります。

Object Storage サービスは、他の OpenStack サービスおよびコンポーネントに依存してアクションを実行します。たとえば、Identity サービス (keystone)、rsync デーモン、ロードバランサーはすべて必要です。

表1.4 Object Storage のコンポーネント

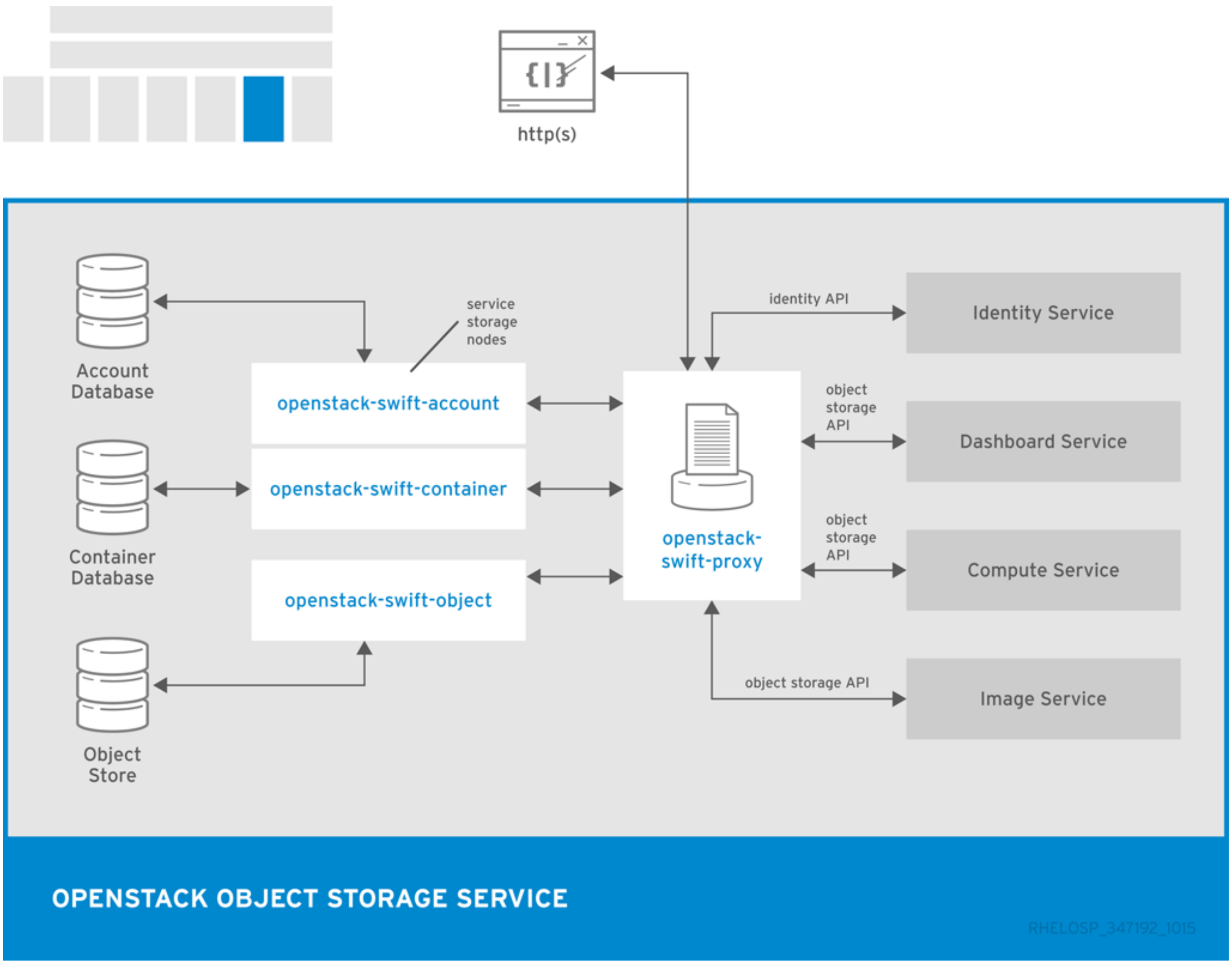
コンポーネント	説明
openstack-swift-account	アカウントデータベースを使用して、コンテナーのリストを処理します。
openstack-swift-container	コンテナーデータベースを使用して、特定のコンテナーに含まれているオブジェクトのリストを処理します。
openstack-swift-object	オブジェクトを保管、取得、削除します。
openstack-swift-proxy	パブリック API の公開、認証、要求のルーティングを行います。オブジェクトは、スプールせずに、プロキシーサーバーを介してユーザーにストリーミングされます。
swift	Object Storage API にアクセスするためのコマンドラインクライアント

表1.5 Object Storage のハウスキーピングコンポーネント

ハウスキーピング	コンポーネント	説明
監査	openstack-swift-account-auditor openstack-swift-container-auditor openstack-swift-object-auditor	Object Storage のアカウント、コンテナー、オブジェクトの整合性を検証し、データの破損から保護します。
レプリケーション	openstack-swift-account-replicator openstack-swift-container-replicator openstack-swift-object-replicator	Object Storage クラスター全体でレプリケーションの一貫性と可用性を確保します (ガベージコレクションを含む)。

ハウスキーピング	コンポーネント	説明
更新	openstack-swift-account-updater openstack-swift-container-updater openstack-swift-object-updater	失敗した更新を特定し、再試行します。

以下の図には、Object Storage が他の OpenStack サービス、データベース、ブローカーと対話するのに使用する主要なインターフェースを示しています。



1.3. 仮想マシン、イメージ、テンプレート

「OpenStack Compute (nova)」

「OpenStack Bare Metal Provisioning (ironic)」

「OpenStack Image (glance)」

「OpenStack Orchestration (heat)」

「OpenStack Data Processing (sahara)」

1.3.1. OpenStack Compute (nova)

OpenStack Compute サービスは、オンデマンドで仮想マシンを提供する、OpenStack クラウドの中核です。Compute は、下層の仮想化メカニズムと対話するドライバーを定義し、他の OpenStack コンポーネントに機能を公開することにより、仮想マシンが一式のノード上で実行されるようにスケジュールします。

Compute は、KVM をハイパーバイザーとして使用する libvirt ドライバー **libvirtd** をサポートしています。ハイパーバイザーは、仮想マシンを作成し、ノード間でのライブマイグレーションを可能にします。ベアメタルマシンのプロビジョニングには、「OpenStack Bare Metal Provisioning (ironic)」 を使用することもできます。

Compute は、インスタンスおよびデータベースへのアクセス認証には Identity サービス、イメージへのアクセスとインスタンスの起動には Image サービス、ユーザーおよび管理用のインターフェースの提供には Dashboard サービスと対話します。

イメージへのアクセスは、プロジェクトまたはユーザー別に制限することが可能です。また、プロジェクトとユーザーのクォータ (例: 単一のユーザーが作成可能なインスタンス数など) を指定することができます。

Red Hat OpenStack Platform クラウドをデプロイする際には、以下のような異なるカテゴリーでクラウドを分割することができます。

リージョン

Identity サービスでカタログ化されている各サービスは、サービスリージョン別に特定されます。サービスリージョンとは、通常地理的な場所およびサービスエンドポイントのことを示します。複数のコンピュートノードを使用するクラウドでは、リージョンによりサービスを別々に分けることができます。

また、リージョンを使用すると、コンピュートをインストールしたシステム間でインフラストラクチャーを共有しつつ、高度の耐障害性を維持することもできます。

セル (テクノロジープレビュー)

Compute ホストは、セルと呼ばれるグループに分割して、大型のデプロイメントや地理的に分離されたインストールを処理することができます。セルは、木構造に構成されています。最上位のセルは API セルと呼ばれ、nova-api サービスを実行しますが、nova-compute サービスは実行しません。

各子セルは、その他すべての標準的な nova-* サービスを実行しますが、nova-api サービスは実行しません。各セルには、個別のメッセージキューとデータベースサービスがあり、API と子セルの間の通信を管理する nova-cells サービスも実行します。

セルには以下のような利点があります。

- 単一の API サーバーを使用して、Compute がインストールされた複数のシステムへのアクセスを制御することができます。
- セルレベルで追加のスケジューリングレベルが利用できます。ホストのスケジューリングとは異なり、仮想マシンの実行時により高い柔軟性と制御を提供します。

注記

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

ホストアグリゲートとアベイラビリティーゾーン

単一のコンピュートのデプロイメントは、複数の論理グループに分割することが可能です。ストレージやネットワークなどの共通のリソースを共有する複数のグループやトラステッドコンピューティングハードウェアなどの特殊なプロパティーを共有するグループなどを複数作成することができます。

このグループは、管理者には、割り当て済みのコンピュートノードと関連メタデータとともにホストアグリゲートとして表示されます。ホストアグリゲートのメタデータは通常、ホストのサブネットに特定のフレーバーやイメージを制限するなどの openstack-nova-scheduler のアクションに向けた情報を提供するのに使用されます。

ユーザーに対しては、グループはアベイラビリティーゾーンとして表示されます。ユーザーはグループメタデータを表示したり、ゾーン内のホスト一覧を確認したりすることはできません。

アグリゲート/ゾーンには、以下のような利点があります。

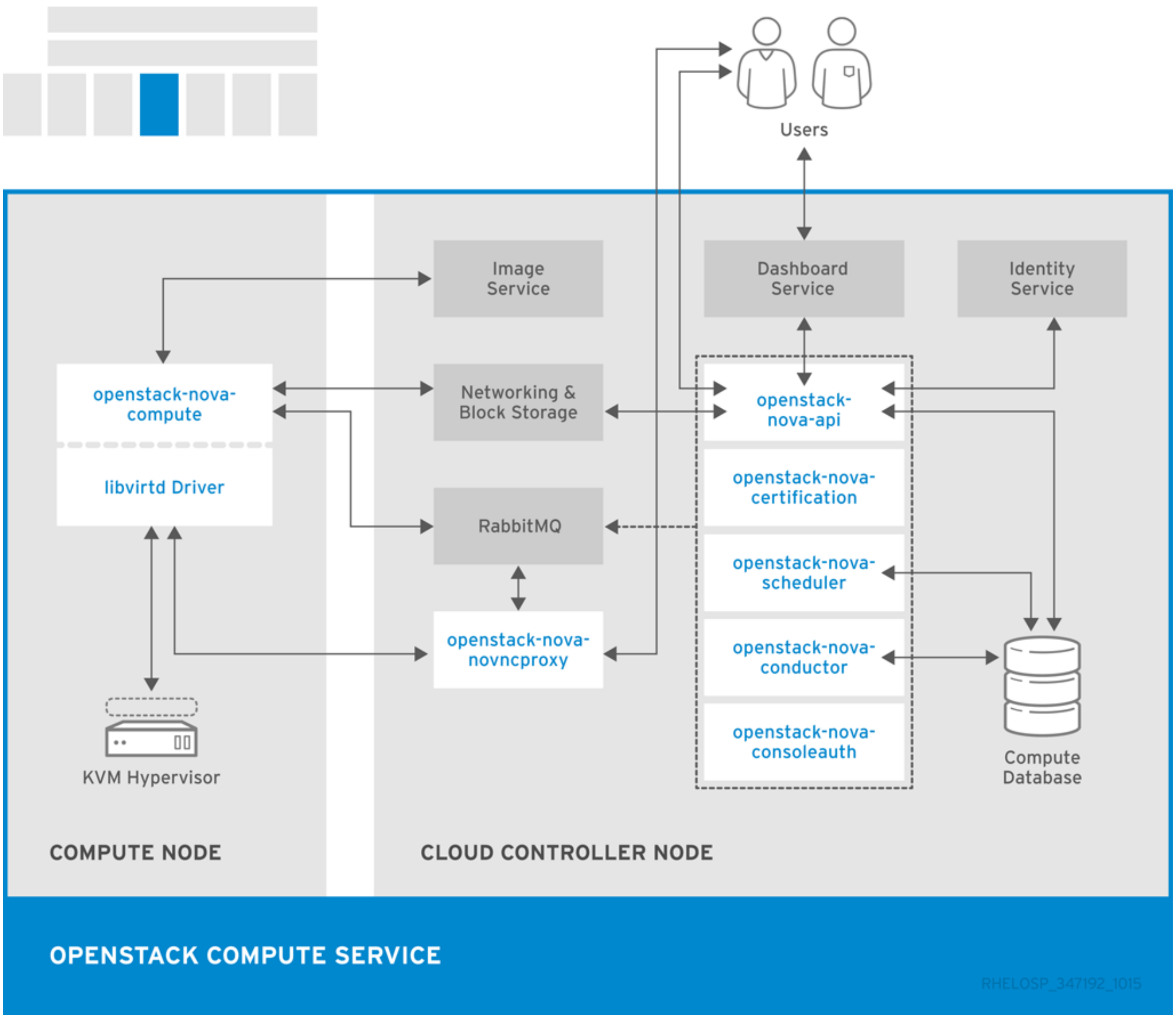
- ロードバランシングおよびインスタンスの分散
- 別々の電源とネットワーク機器を使用して実装される、ゾーン間における物理的な分離および冗長性
- 共通の属性を持つサーバーグループのラベル
- 異なるハードウェアクラスの分離

表1.6 Compute のコンポーネント

コンポーネント	説明
openstack-nova-api	要求を処理し、Compute サービス (例: インスタンスの起動など) へのアクセスを提供します。
openstack-nova-cert	証明書マネージャーを提供します。
openstack-nova-compute	仮想マシンの作成および削除を行うために、各ノード上で実行されます。Compute サービスはハイパーバイザーと対話して新規インスタンスを起動し、Compute のデータベースでインスタンスの状態が維持管理されるようにします。
openstack-nova-conductor	コンピュートノードのデータベースアクセスのサポートを提供し、セキュリティーリスクを軽減します。
openstack-nova-consoleauth	コンソールの認証を処理します。

コンポーネント	説明
openstack-nova-novncproxy	ブラウザー用の VNC プロキシーを提供して、VNC コンソールが仮想マシンにアクセスできるようにします。
openstack-nova-scheduler	設定済みの重みとフィルターに基づいて、 新規仮想マシンに対する要求を正しいノードに割り当てます。
nova	Compute API にアクセスするためのコマンドラインクライアント

以下の図は、Compute サービスとその他の OpenStack コンポーネントの間の関係を示しています。



1.3.2. OpenStack Bare Metal Provisioning (ironic)

OpenStack Bare Metal Provisioning により、ハードウェア固有のドライバーを使用するさまざまなハードウェアベンダーの製品で物理マシンまたはベアメタルマシンのプロビジョニングを行うこと

ができます。Bare Metal Provisioning は Compute サービスと統合して、仮想マシンのプロビジョニングと同じ方法で、ベアメタルマシンのプロビジョニングを行い、**bare-metal-to-trusted-tenant** ユースケースの解決策を提供します。

OpenStack Baremetal Provisioning には以下のような利点があります。

- Hadoop クラスターをベアメタルマシン上にデプロイすることができます。
- ハイパースケールおよびハイパフォーマンスコンピューティング (HPC) のクラスターをデプロイすることが可能です。
- 仮想マシンの影響を受けるアプリケーション用のデータベースホスティングを使用することが可能です。

Bare Metal Provisioning は、スケジューリングとクォータの管理に Compute サービスを使用します。また認証には Identity サービスを使用します。インスタンスのイメージは、KVM ではなく Bare Metal Provisioning をサポートするように設定する必要があります。

以下の図は、物理サーバーがプロビジョニングされた際に Ironic とその他の OpenStack サービスがどのように対話するかを示しています。

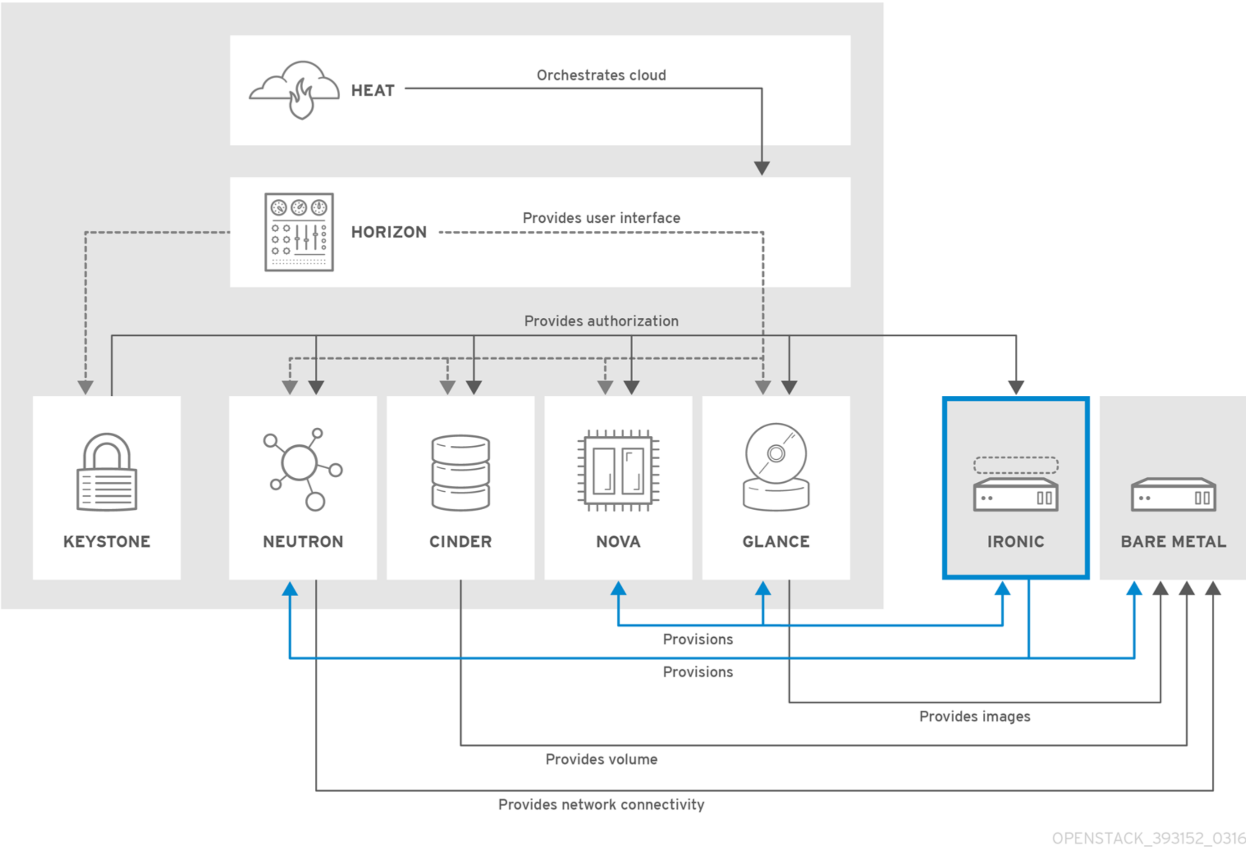
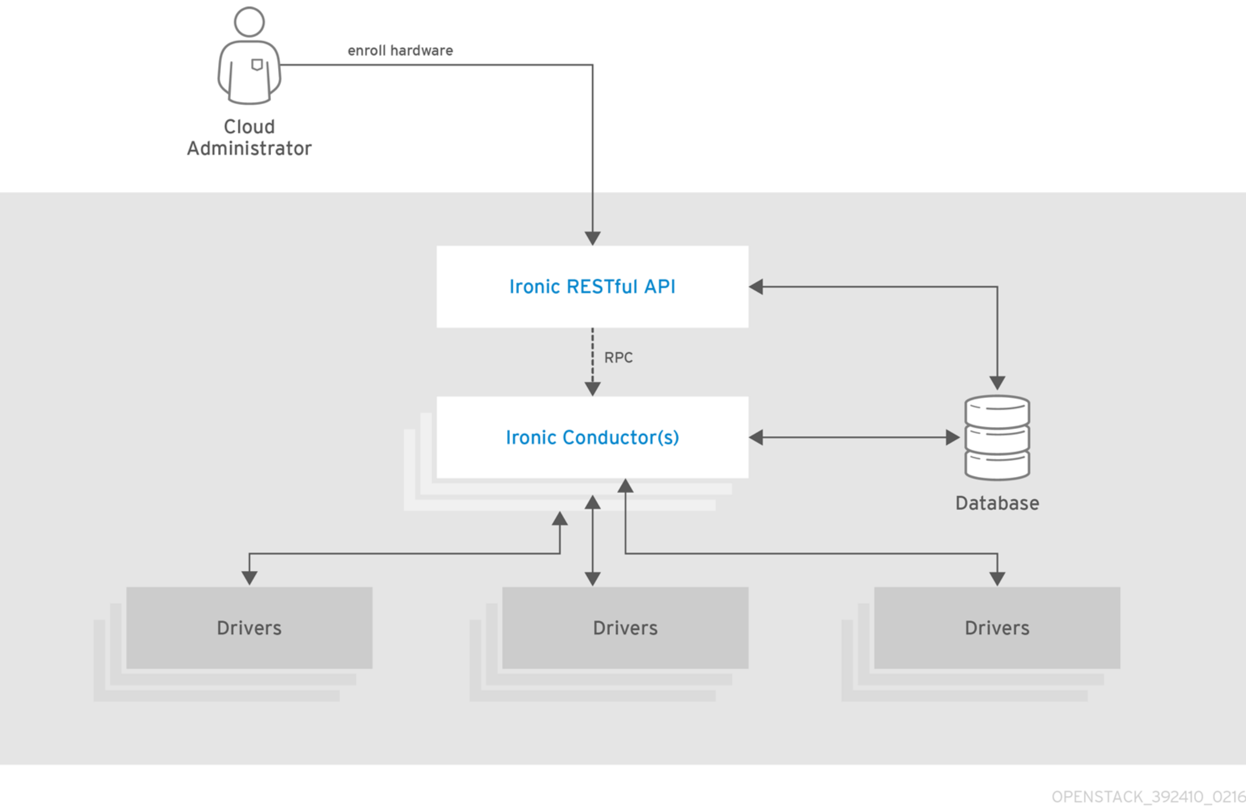


表1.7 Bare Metal Provisioning のコンポーネント

コンポーネント	説明
openstack-ironic-api	ベアメタルノード上のコンピュートリソースに対する要求を処理し、アクセスを提供します。

コンポーネント	説明
openstack-ironic-conductor	ハードウェアおよび ironic のデータベースと直接対話して、要求されたアクションおよび定期的なアクションを処理します。異なるハードウェアドライバーと対話するには、複数のコンダクターを作成してください。
ironic	Bare Metal Provisioning API にアクセスするためのコマンドラインクライアント

以下の図は、Ironic API、コンダクター、ドライバー、その他の OpenStack コンポーネントの間の関係を示しています。



1.3.3. OpenStack Image (glance)

OpenStack Image は、仮想ディスクイメージのレジストリーとして機能します。ユーザーは、新規イメージを追加したり、既存のサーバーのスナップショットを作成して直ちに保存したりすることができます。スナップショットはバックアップ用、またはサーバーを新規作成するためのテンプレートとして使用できます。

登録したイメージは、Object Storage サービスまたは別の場所 (例: シンプルなファイルシステムや外部の Web サーバー) に保管することができます。

以下のイメージディスク形式がサポートされています。

- aki/ami/ari (Amazon のカーネル、RAM ディスク、またはマシーンイメージ)
- iso (CD などの光学ディスクのアーカイブ形式)
- qcow2 (Copy On Write をサポートする Qemu/KVM)

- raw (非構造化の形式)
- vhd (VMware、Xen、Microsoft、VirtualBox などのベンダーの仮想マシンモニターで一般的な Hyper-V)
- vdi (Qemu/VirtualBox)
- vmdk (VMware)

コンテナーの形式は、Image サービスにより登録することも可能です。コンテナーの形式により、イメージに保管される仮想マシンのメタデータの種別と詳細レベルが決定されます。

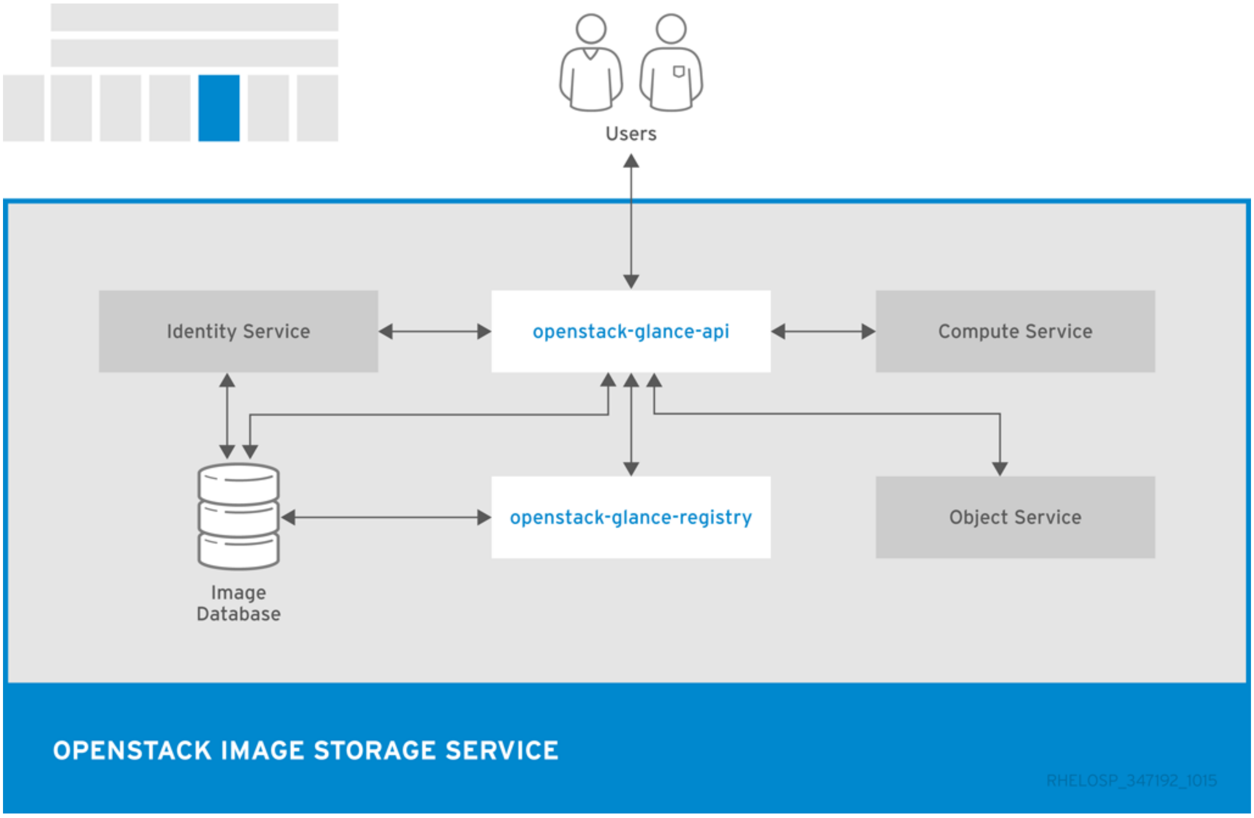
以下のコンテナー形式がサポートされています。

- bare (メタデータなし)
- ova (OVA tar アーカイブ)
- ovf (OVF 形式)
- aki/ami/ari (Amazon のカーネル、RAM ディスク、またはマシーンイメージ)

表1.8 Image のコンポーネント

コンポーネント	説明
openstack-glance-api	ストレージバックエンドと対話して、イメージの取得と保管の要求を処理します。API は openstack-glance-registry を使用してイメージの情報を取得します。レジストリーサービスには直接アクセスしてはなりません。
openstack-glance-registry	各イメージの全メタデータを管理します。
glance	Image API にアクセスするためのコマンドラインクライアント

以下の図には、Image データベースからイメージを登録/取得するのに Image サービスが使用する主要なインターフェースを示しています。



1.3.4. OpenStack Orchestration (heat)

OpenStack Orchestration は、ストレージ、ネットワーク、インスタンス、アプリケーションなどのクラウドリソースを作成および管理するためのテンプレートを提供します。テンプレートは、リソースのコレクションであるスタックの作成に使用されます。

たとえば、インスタンス、Floating IP、ボリューム、セキュリティーグループ、ユーザーのテンプレートを作成することができます。Orchestration は、単一のモジュール型テンプレートを使用した OpenStack の全コアサービスへのアクセスに加えて、自動スケーリングや高可用性などの機能を提供します。

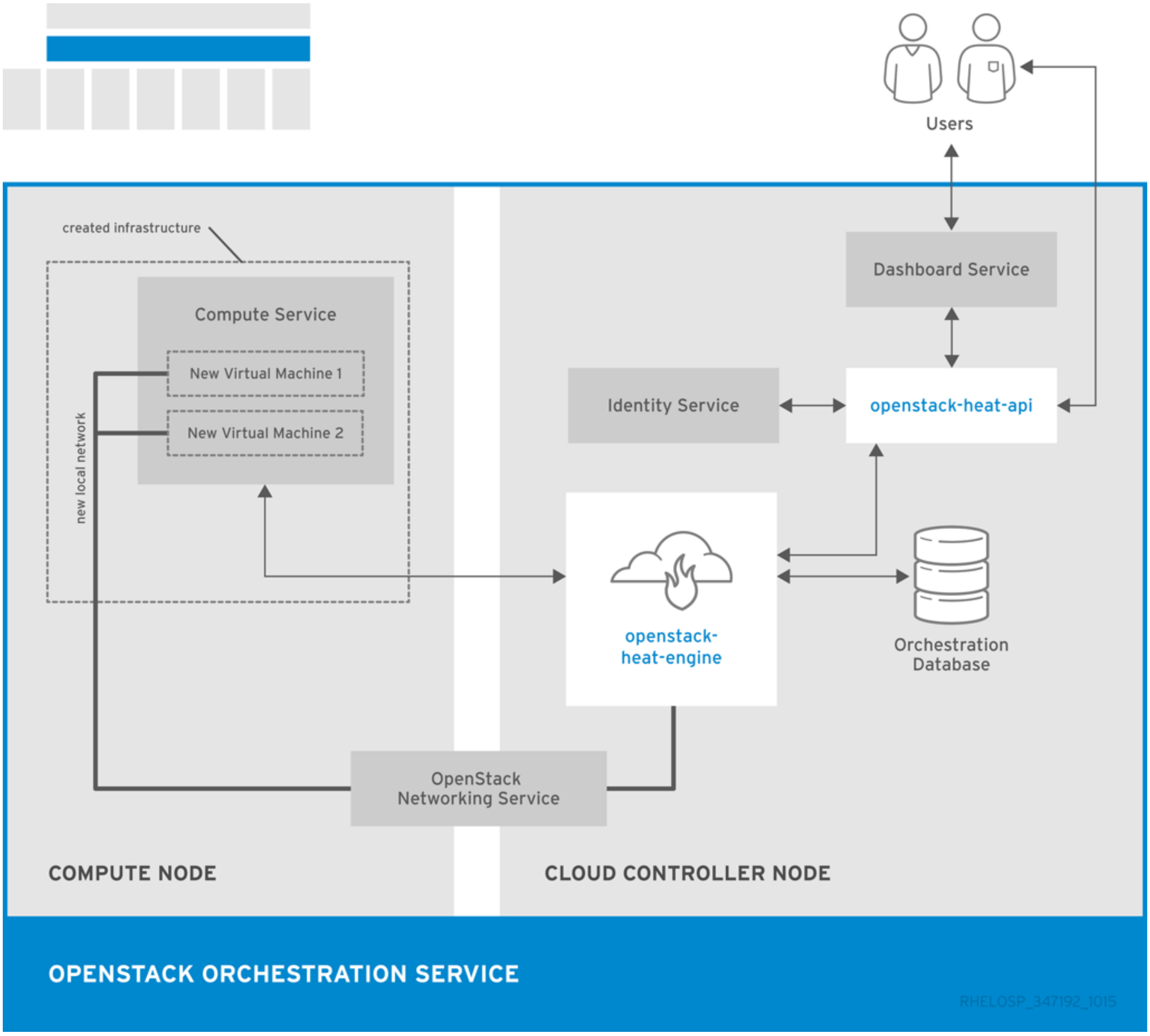
OpenStack Orchestration には以下のような利点があります。

- 単一のテンプレートで下層の全サービス API へのアクセスを提供します。
- テンプレートは、モジュール型でリソース指向です。
- テンプレートは、ネストされたスタックのように、再帰的に定義/再利用することができます。クラウドインフラストラクチャーはモジュール式に定義/再利用されます。
- リソースの実装はプラグ可能なので、カスタムリソースを利用することができます。
- リソースは自動スケーリングが可能なので、使用状況に応じてクラスターに追加/削除されます。
- 基本的な高可用性機能を利用することができます。

表1.9 Orchestration のコンポーネント

コンポーネント	説明
openstack-heat-api	OpenStack のネイティブの REST API。RPC を使用して openstack-heat-engine サービスへの要求を送信することにより、API 要求を処理します。
openstack-heat-api-cfn	AWS CloudFormation との互換性があるオプションの AWS-Query API。RPC を使用して openstack-heat-engine サービスへの要求を送信することにより、API 要求を処理します。
openstack-heat-engine	テンプレートの起動をオーケストレーションして、API コンシューマーに対してイベントを生成します。
openstack-heat-cfntools	ヘルパースクリプトのパッケージ (例: メタデータの更新を処理し、カスタムフックを実行する cfn-hup)
heat	Orchestration API と通信して AWS CloudFormation API を実行するコマンドラインツール

以下の図には、Orchestration サービスが 2 つの新規インスタンスと 1 つのローカルネットワークを使用して新規スタックを作成する場合の主要なインターフェースを示しています。



1.3.5. OpenStack Data Processing (sahara)

OpenStack Data Processing により、OpenStack 上の Hadoop クラスターのプロビジョニングと管理を行うことができます。Hadoop は、クラスター内の大量の構造化/非構造化データを保管および分析します。

Hadoop クラスターは、Hadoop Distributed File System (HDFS) を実行するストレージサーバー、Hadoop の MapReduce (MR) フレームワークを実行するコンピュートサーバー、またはそれらの両方として機能することができるサーバーグループです。

Hadoop クラスター内のサーバーは、同じネットワーク内に配置されている必要がありますが、メモリーやディスクを共有する必要はありません。このため、既存のサーバーとの互換性に影響を及ぼすことなく、サーバーとクラスターの追加/削除を行うことができます。

Hadoop のコンピュートサーバーとストレージサーバーは併置されるので、保管されているデータの分析を高速に行うことができます。タスクはすべてサーバー間で分散され、サーバーのローカルリソースを使用します。

OpenStack Data Processing には以下のような利点があります。

- Identity サービスがユーザーを認証して、Hadoop クラスター内におけるユーザーセキュリティーを提供することができます。
- Compute サービスがクラスターのインスタンスをプロビジョニングすることができます。

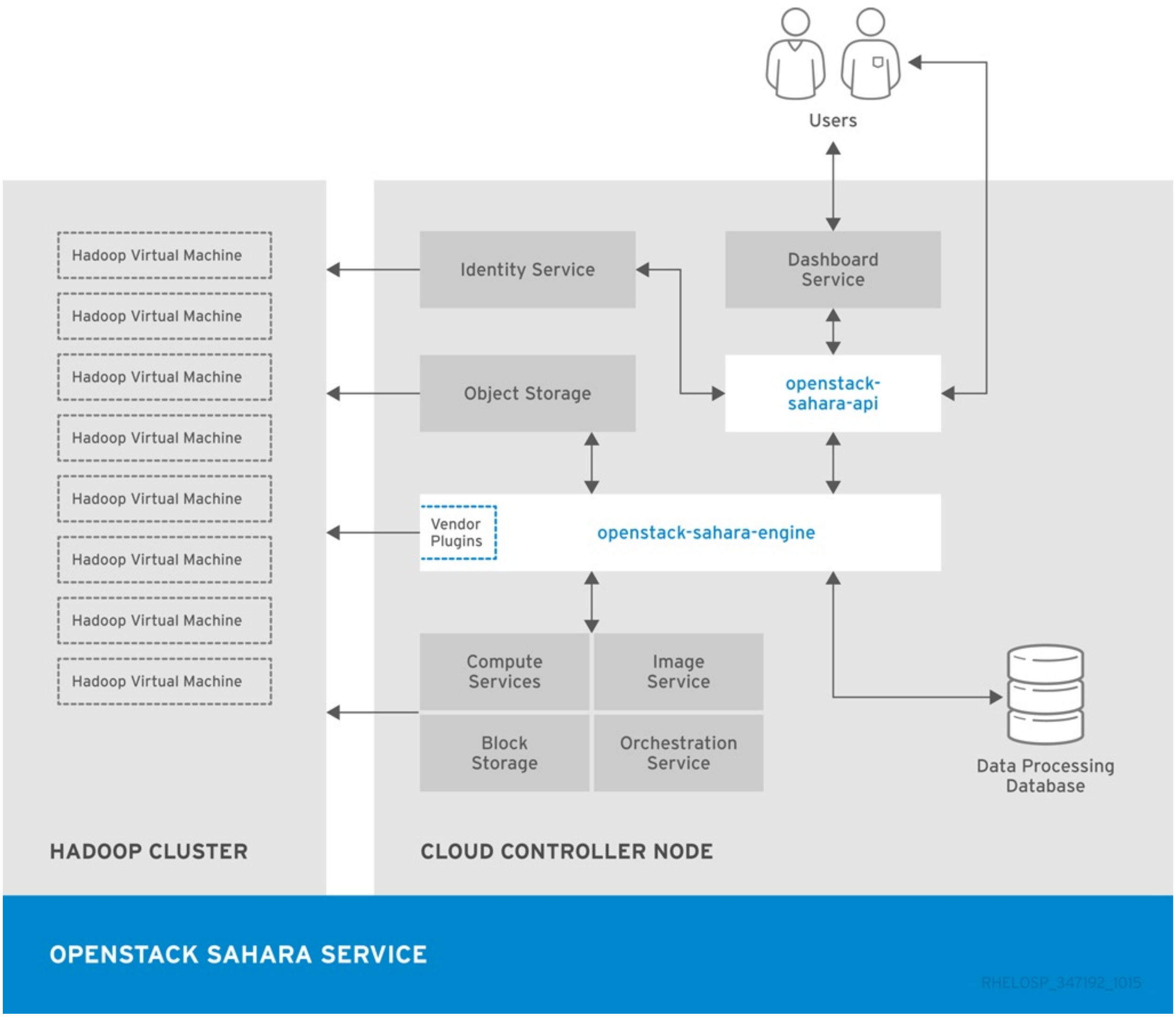
- Image サービスがクラスターのインスタンスを保管することができます。各インスタンスにはオペレーティングシステムとHDFS が含まれます。
- Object Storage サービスを使用して Hadoop ジョブプロセスのデータを格納することができます。
- クラスターの作成と設定にテンプレートを使用することができます。ユーザーは、カスタムテンプレートを作成して設定パラメーターを変更したり、クラスターの作成中にパラメーターを上書きすることができます。ノードはノードグループテンプレートを使用してグループ化され、ノードグループはクラスターテンプレートによりが統合されます。
- ジョブを使用して、Hadoop クラスター上のタスクを実行することができます。ジョブバイナリーには、実行可能コードが保管されます。また、データソースに入出力の場所や必要な認証情報が保管されます。

Data Processing は Cloudera (CDH) プラグインと、ベンダー固有の管理ツール (例: Apache Ambari) をサポートしています。OpenStack Dashboard またはコマンドラインツールを使用してクラスターのプロビジョニングと管理を行うことができます。

表1.10 Sahara のコンポーネント

コンポーネント	説明
openstack-sahara-all	API とエンジンサービスを処理するレガシーパッケージ
openstack-sahara-api	API 要求を処理して、Data Processing サービスへのアクセスを提供します。
openstack-sahara-engine	クラスターの要求とデータ配信を処理するプロビジョニングエンジン
sahara	Data Processing API にアクセスするためのコマンドラインクライアント

以下の図には、Data Processing サービスが Hadoop クラスターのプロビジョニングと管理に使用する主要なインターフェースを示しています。



1.4. アイデンティティー管理

1.4.1. OpenStack Identity (keystone)

OpenStack Identity は、全 OpenStack コンポーネントに対してユーザーの認証と承認を提供します。Identity は、ユーザー名/パスワード認証情報、トークンベースのシステム、AWS 式のログインなど複数の認証メカニズムをサポートしています。

デフォルトでは、Identity サービスはトークン、カタログ、ポリシー、アイデンティティー情報に MariaDB バックエンドを使用します。このバックエンドは、開発環境や小規模なユーザーセットを認証する場合に推奨されます。また、LDAP、SQL などの複数の Identity バックエンドの併用や、memcache や Redis によるトークンの永続化も可能です。

Identity は SAML によるフェデレーションをサポートしています。フェデレーション対応の Identity により、Identity Provider (IdP) とIdentity がエンドユーザーに提供するサービスとの間で信頼関係が確立されます。

注記

フェデレーション対応の Identity および複数のバックエンドの併用には、Eventlet デプロイメントの代わりに Identity API v3 と Apache HTTPD デプロイメントが必要です。

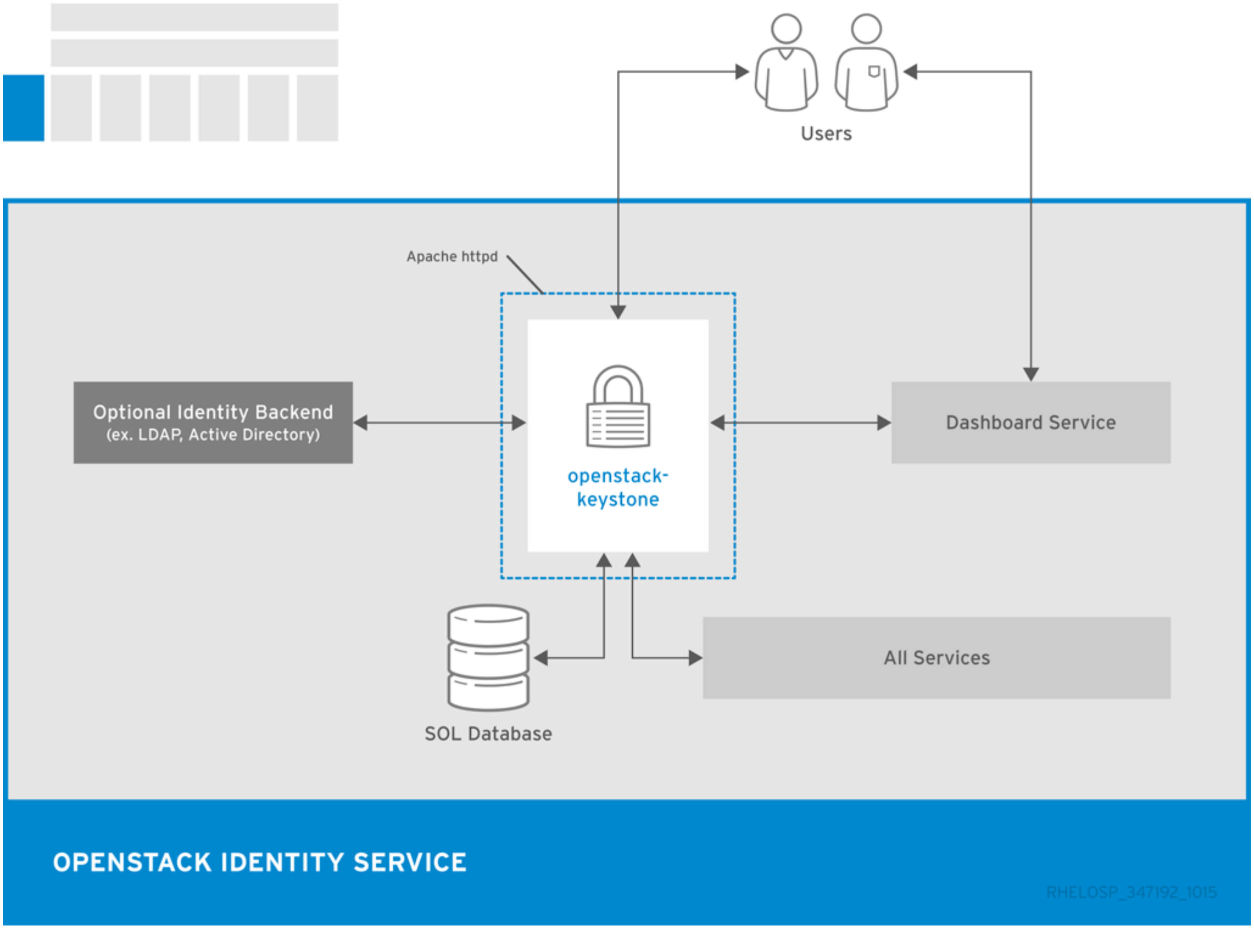
OpenStack Identity には以下のような利点があります。

- 名前やパスワードなどの関連情報を含むユーザーアカウント管理。カスタムユーザーに加えて、カタログ化された各サービス用にユーザーを定義する必要があります。たとえば、**glance** ユーザーは、Image サービス用に定義する必要があります。
- テナント/プロジェクトの管理。ユーザーグループ、プロジェクト、組織をテナントすることができます。
- ロール管理。ロールは、ユーザーのパーミッションを決定します。たとえば、ロールを使用して、営業担当者とマネージャーのパーミッションを区別することができます。
- ドメイン管理。ドメインは、Identity サービスエンティティーの管理上の境界線を決定し、1 つのドメインがユーザー、グループ、テナントのグループを示すマルチテナンシーをサポートします。1 つのドメインには複数のテナントを含めることができます。複数のアイデンティティープロバイダーを併用する場合には、プロバイダーにつき 1 ドメインとなります。

表1.11 Identity のコンポーネント

コンポーネント	説明
openstack-keystone	Identity のサービスおよび管理/パブリック用の API を提供します。Identity API v2 と API v3 の両方がサポートされています。
keystone	Identity API にアクセスするためのコマンドラインクライアント

以下の図には、Identity が他の OpenStack コンポーネントとのユーザー認証で使用する基本的な認証フローを示してます。



1.5. ユーザーインターフェース

「OpenStack Dashboard (horizon)」

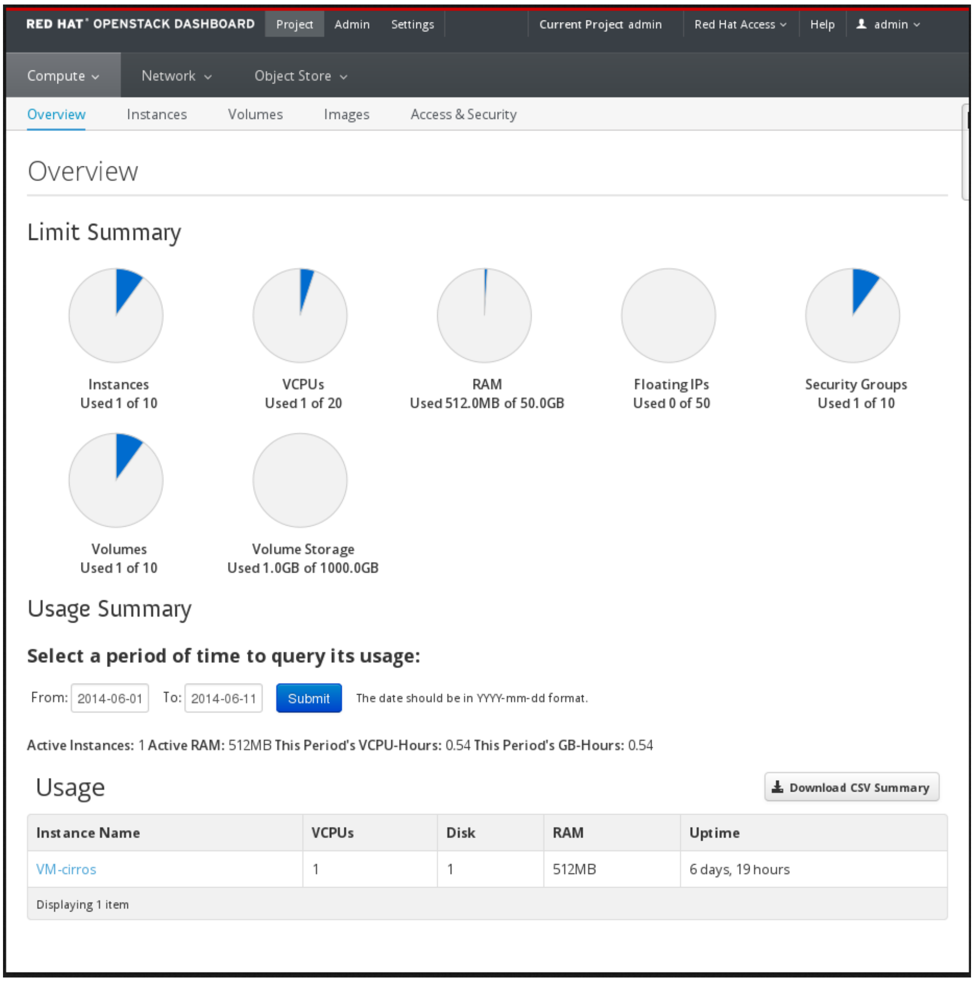
「OpenStack Telemetry (ceilometer)」

1.5.1. OpenStack Dashboard (horizon)

OpenStack Dashboard は、ユーザーおよび管理者がインスタンスの作成/起動やネットワークの管理、アクセス制御の設定などの操作を行うためのグラフィカルユーザーインターフェースを提供します。

Dashboard サービスは、プロジェクト、管理、設定のデフォルトダッシュボードを提供します。Dashboard は、モジュール型設計により、課金、モニタリング、追加の管理ツールなどの他の製品と連結することができます。

以下の画像は、管理ダッシュボードの Compute パネルの例を示しています。

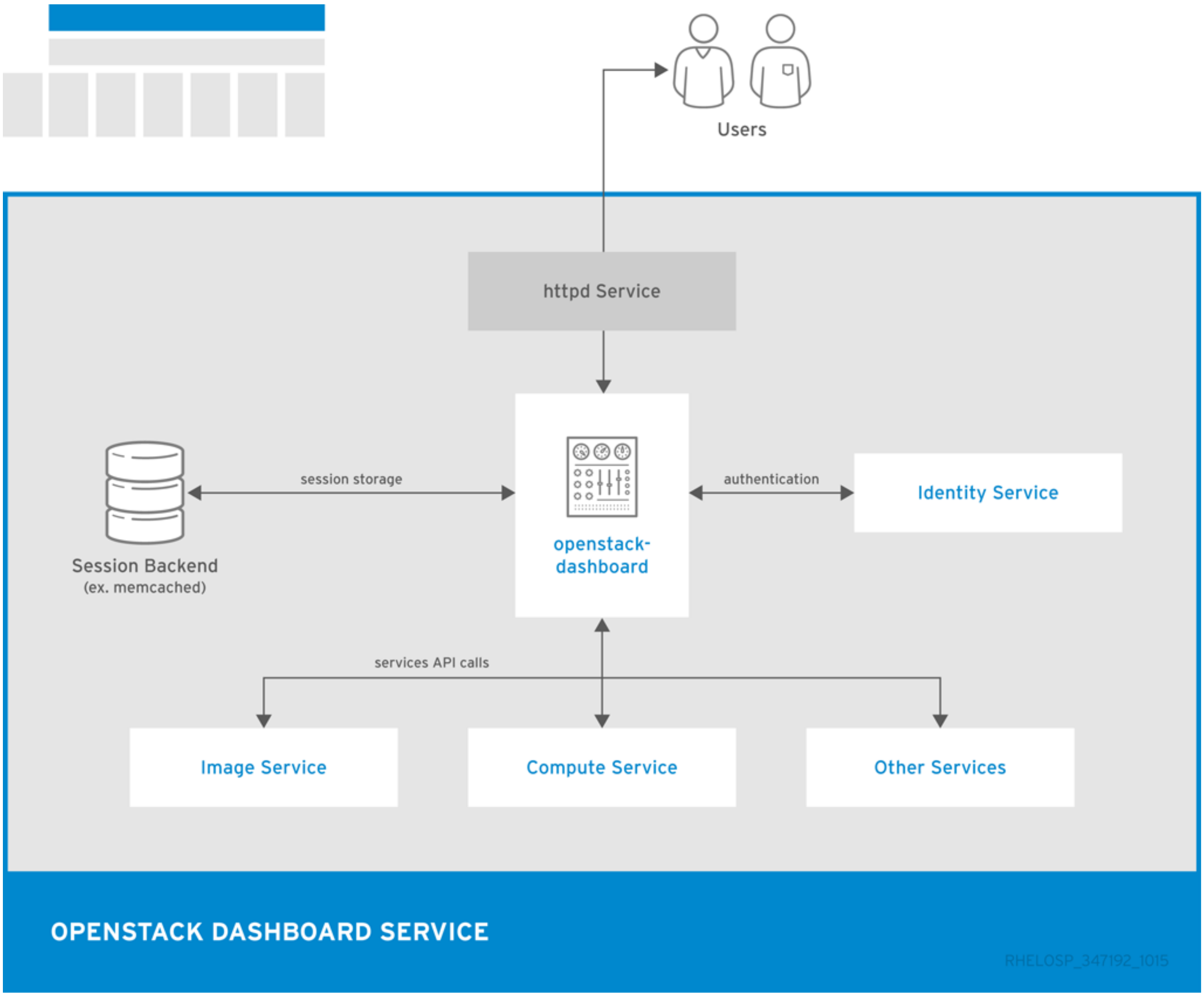


Dashboard にログインするユーザーのロールによって、表示されるダッシュボードとパネルが異なります。

表1.12 Dashboard のコンポーネント

コンポーネント	説明
openstack-dashboard	Web ブラウザーから Dashboard へのアクセスを提供する Django Web アプリケーション
Apache HTTP サーバー (httpd サービス)	アプリケーションをホストします。

以下の図には、Dashboard のアーキテクチャーの概観を示しています。



上記の例には、以下の対話を示しています。

- OpenStack Identity サービスがユーザーの認証および承認を行います。
- セッションバックエンドがデータベースサービスを提供します。
- httpd サービスは、Web アプリケーションおよび API コールのためのその他すべての OpenStack サービスをホストします。

1.5.2. OpenStack Telemetry (ceilometer)

OpenStack Telemetry は、OpenStack をベースとするクラウドのユーザーレベルの使用状況データを提供します。データは、顧客の課金、システムの監視、警告に使用することができます。Telemetry は既存の OpenStack コンポーネント (例: Compute の使用イベント) や libvirt などの OpenStack インフラストラクチャーリソースのポーリングにより送信される通知からデータを収集することができます。

Telemetry には、信頼済みのメッセージングシステムを介して認証済みのエージェントと通信し、データを収集/集計するストレージデーモンが含まれています。また、サービスは、新規モニターを追加するのに使用可能なプラグインシステムを使用します。API サーバー、中央エージェント、データストアサービス、コレクターエージェントを異なるホストにデプロイすることができます。

このサービスは、MongoDB データベースを使用して、収集したデータを格納します。データベースにアクセスできるのは、コレクターエージェントと API サーバーのみです。

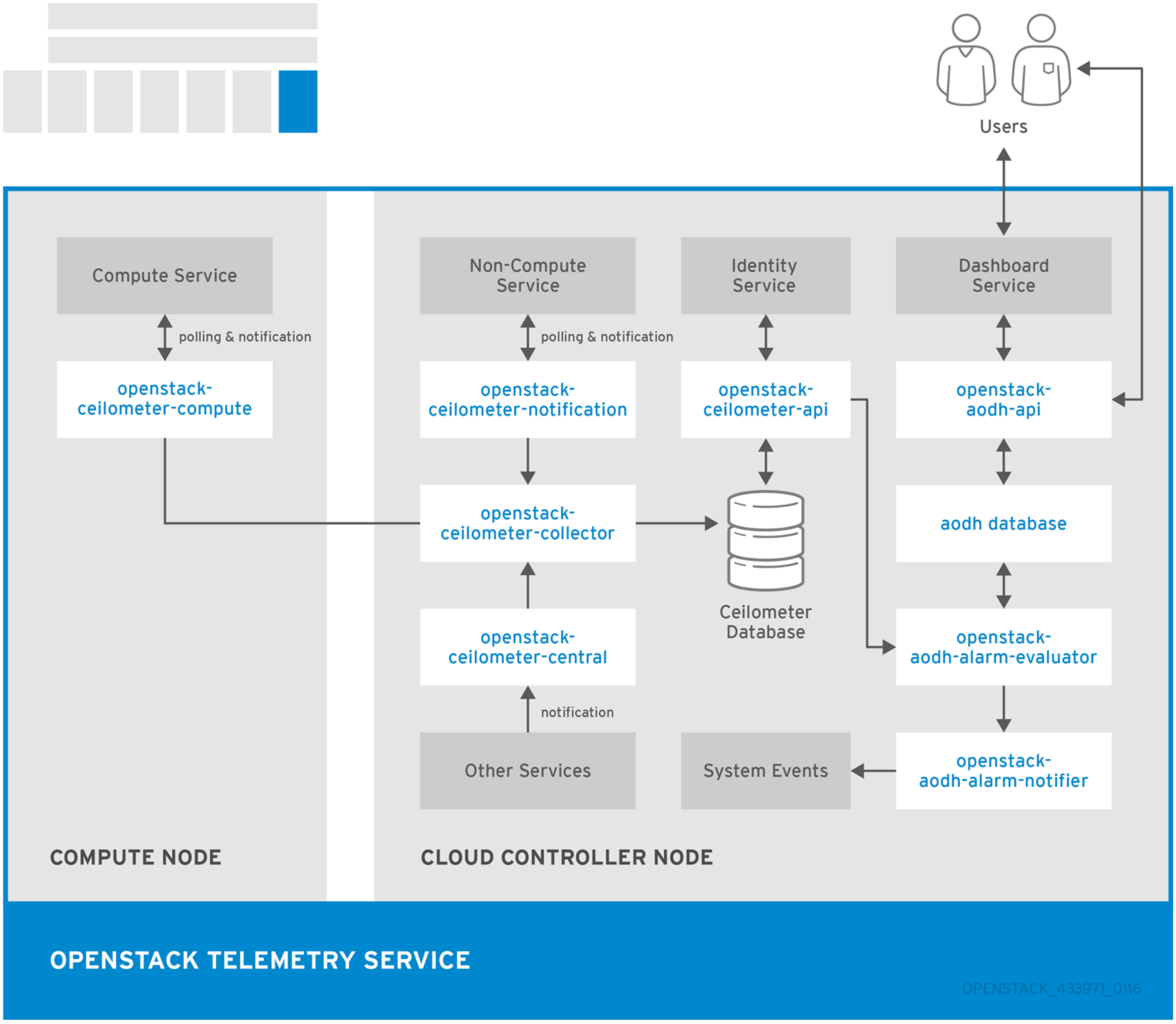
アラームと通知は、新たに **aodh** サービスによって処理/制御されます。

表1.13 Telemetry のコンポーネント

コンポーネント	説明
openstack-aodh-api	データストアに保管されているアラームの情報へのアクセスを提供します。
openstack-aodh-alarm-evaluator	スライディング時間枠の閾値超過に関連する統計の傾向に基づいて、どのような場合にアラームを生成するかを決定します。
openstack-aodh-alarm-notifier	アラームのトリガー時にアクションを実行します。
openstack-aodh-alarm-listener	事前に定義したイベントパターンが発生した場合にアラームを生成します。
openstack-ceilometer-api	中央管理サーバー (単一または複数) で実行して、データベース内のデータへのアクセスを提供します。
openstack-ceilometer-central	中央管理サーバーで実行して、インスタンスやコンピュートノードには依存しないリソースについての使用状況の統計をポーリングします。エージェントは水平スケーリングはできないので、このサービスのインスタンスは 1 回に 1 つしか実行できません。
openstack-ceilometer-collector	1 つまたは複数の中央管理サーバーを実行してメッセージキューをモニタリングします。各コレクターは処理を行い、通知メッセージを Telemetry メッセージに変換し、メッセージバスに対して適切なトピックでメッセージを送信します。 Telemetry メッセージは、変更なしでデータストアに書き込まれます。エージェント内の通信はすべて、ceilometer-alarm-evaluator サービスと同様の ceilometer-api サービスに対する AMQP/REST コールをベースとするので、これらのエージェントを実行する場所を選択することができます。
openstack-ceilometer-compute	各コンピュートノードで実行して、リソース使用状況の統計をポーリングします。各 nova-compute ノードでは ceilometer-compute エージェントがデプロイ済みで実行されている必要があります。
openstack-ceilometer-notification	さまざまな OpenStack サービスからメトリックをコレクターサービスにプッシュします。

コンポーネント	説明
ceilometer	Telemetry API にアクセスするためのコマンドラインクライアント

以下の図には、Telemetry サービスが使用するインターフェースを示しています。



1.6. サードパーティーのコンポーネント

1.6.1. サードパーティーのコンポーネント

一部の Red Hat OpenStack Platform コンポーネントは、サードパーティーのデータベース、サービス、ツールを使用します。

1.6.1.1. データベース

- MariaDB は、Red Hat Enterprise Linux に同梱されているデフォルトのデータベースです。MariaDB により、Red Hat はオープンソースコミュニティーで開発されたソフトウェアを完全にサポートすることができます。Telemetry を除く各 OpenStack コンポーネントは、MariaDB サービスを実行する必要があるため、完全な OpenStack クラウドサービスをデプロイする前

や、スタンドアロンの OpenStack コンポーネントをインストールする前には、MariaDB をデプロイしておく必要があります。

- Telemetry サービスは MongoDB データベースを使用して、コレクターエージェントから収集した使用状況のデータを保管します。データベースにアクセスできるのは、コレクターエージェントと API サーバーのみです。

1.6.1.2. メッセージング

RabbitMQ は、AMQP 基準に基づいた、オープンソースの頑強なメッセージングシステムです。RabbitMQ は、数多くのエンタープライズシステムで使用され、商用に幅広くサポートされているハイパフォーマンスなメッセージブローカーです。Red Hat OpenStack Platform では、RabbitMQ はデフォルトの推奨メッセージブローカーです。

RabbitMQ はキューイング、配信、セキュリティー、管理、クラスタリング、フェデレーションなどの OpenStack のトランザクションを管理します。また、高可用性とクラスタリングのシナリオで主要な役割を担います。

1.6.1.3. 外部キャッシュ

memcached や Redis などの外部のキャッシュ用アプリケーションは永続的な共有ストレージを提供し、データベースの負荷を軽減することにより動的な Web アプリケーションを迅速化します。外部キャッシュは、さまざまな OpenStack コンポーネントで使用されます。以下に例を示します。

- Object Storage サービスは、対話をするたびに各クライアントに再承認を要求する代わりに、memcached を使用して認証済みのクライアントをキャッシュします。
- デフォルトでは、Dashboard はセッションストレージに memcached を使用します。
- Identity サービスはトークンの永続化に Redis または memcached を使用します。

第2章 ネットワークに関する詳細

2.1. 基本的なネットワークの仕組み

ネットワークは、複数のコンピューター間で伝送される情報で構成されます。最も基本的なレベルでは、ネットワークインターフェースカード (NIC) が 1 枚ずつ搭載された 2 台のマシンをケーブルで接続することによって構成します。OSI ネットワークモデルでは、これはレイヤー 1 に相当します。

3 台以上のコンピューターを使用する場合には、スイッチというデバイスを追加してこの構成をスケールアウトする必要があります。スイッチとは、追加のマシンを結線するための複数のイーサネットポートが搭載された専用デバイスです。このような構成はローカルエリアネットワーク (LAN) と呼ばれています。

スイッチにより、OSI モデルのレイヤー 2 に上がり、下層のレイヤー 1 よりもインテリジェントな機能が適用されます。各 NIC には、ハードウェアに割り当てられる一意な MAC アドレス番号があり、この番号を使用することにより、同じスイッチに結線されたマシンが相互に認識できるようになります。

スイッチは、どの MAC アドレスがどのポートにプラグインされるかのリストを管理するので、コンピューター間でデータ送信を試みると、スイッチは各 NIC がどこに配置されているかを認識し、正しい宛先にネットワークトラフィックを伝送するための回路を調整します。

2.1.1. 複数の LAN の接続

2 つの別々のスイッチ上で 2 つの LAN を使用している場合には、以下の方法で、それらのスイッチを接続して相互に情報を共有することができます。

トランクケーブル

トランクケーブルと呼ばれる物理ケーブルを 1 本使用して、2 つのスイッチを直接接続することが可能です。この構成では、トランクケーブルを各スイッチのポートに差し込んでから、それらのポートをトランクポートとして定義します。これで 2 つのスイッチが 1 つの大きな論理スイッチとして機能し、接続されているコンピューター同士が相互に検出できるようになります。このオプションはあまりスケーラブルではないため、より多くのスイッチを直接リンクすると、オーバーヘッドが問題となります。

ルーター

ルーターと呼ばれるデバイスを使用して、各スイッチからケーブルを接続します。これにより、ルーターは両スイッチで設定したネットワークを認識するようになります。ルーターに結線した各スイッチはインターフェースとなり、ネットワークのデフォルトゲートウェイとして知られる IP アドレスが割り当てられます。デフォルトゲートウェイの「デフォルト」とは、宛先のマシンがデータの送信元と同じ LAN 上にないことが明らかな場合のトラフィックの送信先のことです。

各コンピューターにデフォルトゲートウェイを設定すると、トラフィック送信のために、他のネットワーク上にある全コンピューターを認識する必要がなくなります。これで、デフォルトゲートウェイのみにトラフィックが送信されるようになり、そこからの処理はルーターが行います。ルーターは、どのネットワークがどのインターフェースに存在するかを把握しているため、指定の宛先に、問題なくパケットを送信することができます。ルーティングは、IP アドレスやサブネットなどの一般的に知られている概念と同様に、OSI モデルのレイヤー 3 で機能します。

注記

この概念は、インターネット自体の仕組みと同じです。さまざまな組織で稼働する多数の個別ネットワークはすべて、スイッチやルーターを使用して相互に接続しています。トラフィックは適切なデフォルトゲートウェイを辿っていくと、最終的に適切な宛先に到着します。

2.1.2. VLAN

仮想ローカルエリアネットワーク (VLAN) により、同じスイッチ上で実行されるコンピューターのネットワークトラフィックを分割することができます。ポートが別のネットワークのメンバーとなるように設定することで、スイッチを論理的に分割することができます。この構成では、ポートをミニ LAN 化することによって、セキュリティーの目的別にトラフィックを分離することができます。

たとえば、スイッチにポートが 24 個ある場合に、ポート 1 から 6 までは VLAN200 に所属し、ポート 7 から 18 までは VLAN201 に所属するように定義することができます。VLAN200 に接続されているコンピューターは、VLAN201 のコンピューターと完全に分離され、直接通信できなくなります。2 つの VLAN 間のトラフィックはすべて、2 つの別個の物理スイッチであるかのようにルーターを通過する必要があります。また、ファイアウォールを使用して相互通信可能な VLAN を決定することにより、セキュリティーを強化することも可能です。

2.1.3. ファイアウォール

ファイアウォールは、IP ルーティングと同じ OSI レイヤーで動作します。多くの場合、ファイアウォールはルーターと同じネットワークセグメントに存在し、全ネットワーク間で移動するトラフィックを制御します。ファイアウォールは、トラフィックがネットワークに出入りできるかどうかを規定する事前定義済みのルールを使用します。これらのルールは粒度を高くすることが可能で、たとえば「VLAN 200 上のサーバーは、Web (HTTP) トラフィックを一方向に転送している場合にのみ、木曜の午後に限り、VLAN 201 上のコンピューターのみと通信できるものとする」というように定義することができます。

このようなルールを適用するには、ファイアウォールの一部でステートフルパケットインスペクション (SPI) も実行し、パケットのコンテンツを検証して、パケットの実際の内容がパケットが主張する内容と同じであることを確認します。ハッカーは、トラフィックを実際の内容とは別のものとして転送して、密かにデータを抜き出すことが知られており、SPI はこのような脅威を軽減する手段の 1 つです。

2.1.4. ブリッジ

ネットワークブリッジは、基本的には OSI モデルのレイヤー 2 で動作するスイッチですが、唯一の機能はルーターと同じように別々のネットワークに同時に接続することが可能なことです。

2.2. OPENSTACK のネットワーク

サービスおよび設定によって定義されるネットワークを除いた、OpenStack クラウドにおけるすべての基本的なネットワークの概念。これはソフトウェア定義ネットワーク (SDN) として知られています。仮想スイッチ (Open vSwitch) とルーター (L3 エージェント) により、インスタンスは相互に通信することができます。また、物理ネットワークを使用した外部との通信も許可することも可能です。Open vSwitch のブリッジは、仮想ポートをインスタンスに割り当て、送受信トラフィックを物理ネットワークに橋渡しすることができます。

2.3. 高度な OPENSTACK NETWORKING の概念

2.3.1. レイヤー 3 高可用性

OpenStack Networking は、仮想ネットワークコンポーネントをホストする機能に特化した物理サーバーである中央ネットワークノードで仮想ルーターをホストします。これらの仮想ルーターは、仮想マシンの送受信トラフィックを誘導するため、環境の接続性を維持するのに不可欠です。物理サーバーは多くの理由から停止する場合があるため、ネットワークノードが利用できなくなると仮想マシンがその影響を受けやすくなる可能性があります。

この問題を軽減するために、OpenStack Networking は、レイヤー 3 の高可用性 (L3 HA) を採用し、仮想ルーターと Floating IP アドレスを保護する業界標準の VRRP を実装しています。L3 HA を使用すると、テナントの仮想ルーターは複数の物理ネットワークノードに無作為に割り当てられます。その 1 つはアクティブなルーターとして指定され、それ以外のルーターは、アクティブなルーターをホストするネットワークノードがオフラインになった場合に引き継ぐ準備が整ったスタンバイの役割を果たします。

注記

「レイヤー 3」は、OSI モデルにおいてこの機能が動作する層で、ルーティングおよび IP アドレス指定の保護に役立ちます。

詳しい情報は、『**ネットワークガイド**』の「レイヤー 3 高可用性」の項を参照してください。

2.3.2. Load Balancing-as-a-Service (LBaaS)

Load Balancing-as-a-Service (LBaaS) により、OpenStack Networking は、指定のインスタンス間で受信ネットワーク要求を均等に分散できるようになります。この分散により、インスタンス間でワークロードが共有され、システムリソースをより効率的に使用できるようになります。受信要求は、以下の負荷分散メソッドのいずれかを使用して分散されます。

ラウンドロビン

複数のインスタンス間で要求を均等にローテーションします。

送信元 IP アドレス

同じ送信元 IP アドレスからの要求は常に同じインスタンスへ送信されます。

最小コネクション

アクティブな接続が最も少ないインスタンスに要求が割り当てられます。

詳しくは、『**ネットワークガイド**』の「Load Balancing-as-a-Service (LBaaS) の設定」の項を参照してください。

2.3.3. IPv6

OpenStack Networking は、テナントネットワークで IPv6 アドレスをサポートしているので、仮想マシンに IPv6 アドレスを動的に割り当てるのが可能です。OpenStack Networking は、既存の DHCP インフラストラクチャーから IPv6 アドレスを受け取ることができるように、物理ルーターで SLAAC と統合することも可能です。

詳しくは、『**ネットワークガイド**』の「IPv6」の項を参照してください。

第3章 設計

本項では、Red Hat OpenStack Platform デプロイメントの設計における技術および運用面の考慮事項を記載します。

注記

アーキテクチャー例はすべて、KVM ハイパーバイザーを使用する Red Hat Enterprise Linux 7.3 上に OpenStack Platform をデプロイすることを前提とします。

3.1. プランニングモデル

Red Hat OpenStack Platform デプロイメントを設計する際には、プロジェクトの期間がそのデプロイメントの設定とリソース割り当てに影響を及ぼす場合があります。各プランイングモデルは、異なる目標を達成することを目的としているため、異なる点を考慮する必要があります。

3.1.1. 短期モデル (3 カ月間)

短期間のキャパシティープランニングと予測を実行するには、以下のメトリックを取得することを検討してください。

- 仮想 CPU の合計数
- 割り当て済みの仮想メモリーの合計
- I/O の平均レイテンシー
- ネットワークトラフィック
- コンピュートの負荷
- ストレージの割り当て

仮想 CPU、仮想メモリー、およびレイテンシーのメトリックは、キャパシティープランニングで最も利用しやすい指標です。これらの詳細情報に対して、標準的な二次回帰を適用することで、向こう 3 カ月間のキャパシティー予測を得ることができます。この推定値を使用して、追加のハードウェアをデプロイする必要があるかどうかを判断してください。

3.1.2. 中期モデル (6 カ月間)

評価を繰り返し実施するモデルを採用し、トレンド予測と実際の使用率の偏差を推定する必要があります。この情報は、標準的な統計ツールや Nash-sutcliffe 係数などの特化した分析モデルを使用して分析することができます。トレンドは、先と同様に、二次回帰で計算することも可能です。

注記

複数のインスタンスフレーバーが混在する利用環境では、仮想 CPU と仮想メモリーのメトリックを 1 つのメトリックとして考察すると、仮想メモリーと仮想 CPU の使用率の相関をより簡単に把握することができます。

3.1.3. 長期モデル (1 年間)

1 年の期間においては、全体的なリソース使用量が変動する可能性があり、元の長期的なキャパシティー予測との間には通常偏差が発生します。そのため、特に使用率がクリティカルなリソースについては、二次回帰によるキャパシティー予測は、指標として不十分な場合があります。

長期的なデプロイメントのプランニングを行う際には、1 年間にわたるデータに基づくキャパシティープランニングのモデルでは、少なくとも一次導関数のフィッティングが必要です。利用パターンによっては、頻度分析が必要となる場合もあります。

3.2. コンピュートリソース

コンピュートリソースは、OpenStack クラウドの中核です。このため、Red Hat OpenStack Platform デプロイメントを設計する際には、物理/仮想リソースの割り当て、分散、フェイルオーバー、および追加デバイスを考慮することを推奨します。

3.2.1. 一般的な考慮事項

各ハイパーバイザーのプロセッサー数、メモリー、ストレージ

プロセッサーのコアおよびスレッドの数は、コンピュートノードで実行可能なワーカースレッドの数に直接影響を及ぼします。このため、サービスに基づいて、全サービスでバランスの取れたインフラストラクチャーをベースとする設計を決定する必要があります。

ワークロードのプロファイルによっては、追加のコンピュートリソースをクラウドに後で追加することが可能です。特定のインスタンスのフレーバーに対する需要は、ハードウェアを個別に設計するための正当な理由にはならない場合があり、代わりにコモディティー化したシステムが優先されます。

いずれの場合も、一般的なインスタンスの要求に対応することのできるハードウェアリソースを割り当てることから設計を開始します。追加のハードウェア設計をアーキテクチャー全体に加えるには、この手順は後で実行することができます。

プロセッサーの種別

プロセッサーの選択は、ハードウェアの設計における非常に重要な考慮事項です。特に、異なるプロセッサー間における機能とパフォーマンスの特性を比較する際に大切となります。

プロセッサーには、仮想化コンピュートホスト専用に、ハードウェア支援による仮想化、メモリーページング、EPT シャドウイングテクノロジーなどの機能を搭載することができます。これらの機能は、クラウド仮想マシンのパフォーマンスに大きな影響を及ぼします。

リソースノード

クラウド内のハイパーバイザー以外のリソースノードによるコンピュートに対する要求を考慮する必要があります。リソースノードには、コントローラーノードや Object Storage、Block Storage、Networking などのサービスを実行するノードが含まれます。

リソースプール

オンデマンドで提供するリソースの複数のプールを割り当てる Compute の設計を使用します。この設計により、クラウド内のリソースの使用率が最大限に高められます。各リソースのプールは特定のインスタンスフレーバーまたはフレーバーグループに対応する必要があります。

複数のリソースプールを設計することにより、Compute ハイパーバイザーに対してインスタンスがスケジュールされると必ず各ノードリソースセットが割り当てられて、利用可能なハードウェアの使用率を最大限に高めるのに役立ちます。これは一般的にはビンパッキングと呼ばれます。

リソースプール内のノード間で一貫したハードウェア設計を採用すると、ビンパッキングにも役立ちます。Compute リソースプールに選択されたハードウェアノードは、共通のプロセッサー、メモリー、およびストレージレイアウトを共有する必要があります。共通のハードウェア設計を選択することにより、デプロイメント、サポート、ノードのライフサイクル管理が容易になります。

オーバーコミット比

OpenStack では、コンピュートノードの CPU とメモリーをオーバーコミットすることができます。これは、クラウド内で実行するインスタンスの数を増やすのに役立ちます。ただし、オーバーコミットにより、インスタンスのパフォーマンスが低下する場合があります。

オーバーコミット比とは、利用可能な物理リソースと比較した利用可能な仮想リソースの比率です。

- デフォルトの CPU 割り当て比 16:1 は、スケジューラーが 1 つの物理コアに対して最大 16 の仮想コアを割り当てることを意味します。たとえば、物理ノードに 12 コアある場合には、スケジューラーは最大で 192 の仮想コアを割り当てることができます。標準的なフレーバーの定義では、1 インスタンスにつき 4 仮想コアなので、この比率では、1 物理ノードで 48 のインスタンスを提供することができます。

- デフォルトのメモリー割り当て比 1.5:1 とは、インスタンスに関連付けられているメモリーが物理ノードで利用可能なメモリー容量の 1.5 倍以下の場合に、スケジューラーがインスタンスを物理ノードに割り当てることを意味します。

CPU およびメモリーのオーバーコミット比は、コンピュートノードのハードウェアレイアウトに直接影響を及ぼすので、設計フェーズ中にチューニングしておくことが重要となります。Compute のリソースプールのようなインスタンスにサービスを提供するためのハードウェアノードを設計する際には、ノードで利用可能なプロセッサーコアの数とともに、フルキャパシティーで稼働するインスタンスにサービスを提供するのに必要なディスクとメモリーも考慮してください。

たとえば、**m1.small** のインスタンスは、1 仮想 CPU、20 GB の一時ストレージ、2048 MB のメモリーを使用します。10 コアの CPU が 2 つあり、ハイパースレッディングが有効化されたサーバーの場合には、次のようになります。

- デフォルトの CPU オーバーコミット比 16:1 で、合計 640 (2 × 10 × 2 × 16) の **m1.small** インスタンスを提供することが可能です。
- デフォルトのメモリーオーバーコミット 1.5:1 は、サーバーに最小で 853 GB (640 × 2048 MB / 1.5) のメモリーが必要なことを意味します。

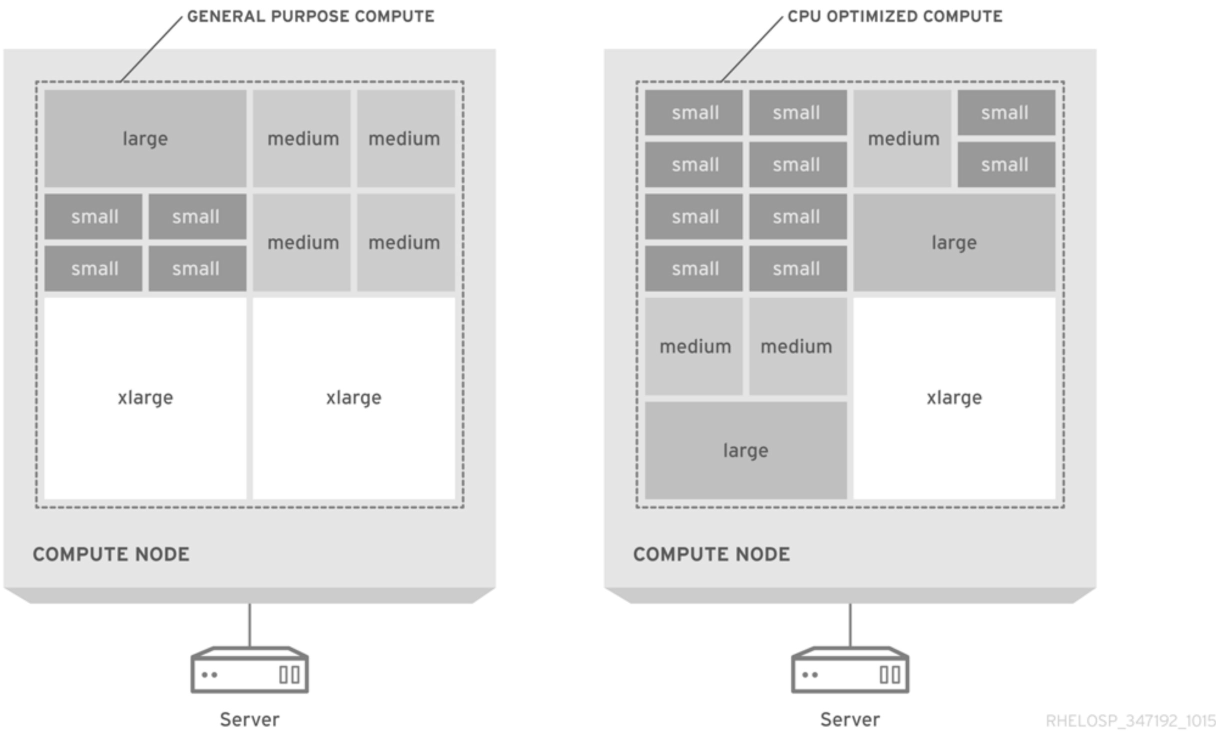
メモリーでノードのサイズ指定する際には、オペレーティングシステムとサービスのニーズに対応する必要な追加メモリーを考慮することも重要です。

3.2.2. フレーバー

作成される各インスタンスには、フレーバーまたはリソーステンプレートが適用され、それによってインスタンスのサイズと容量が決定します。フレーバーは、第 2 の一時ストレージ、スワップディスク、メタデータを指定して、使用量や特別なプロジェクトへのアクセスを制限することができます。デフォルトのフレーバーには、これらの追加の属性は定義されていません。インスタンスのフレーバーにより、一般的なユースケースは、予測可能なサイズに設定され、場当たり的に決定されるわけではないので、キャパシティー予測を立てることができます。

物理ホストへの仮想マシンのパッキングを円滑にするために、デフォルトでは、2 番目に大きなフレーバーが選択されます。このフレーバーは、すべてのサイズが最も大きなフレーバーの半分で、仮想 CPU の数、仮想メモリー容量、一時ディスクの容量がそれぞれ半分となります。それ以降の各サイズは、それよりも 1 サイズ大きなフレーバーの半分のサイズとなります。

以下の図には、汎用のコンピューティング設計と CPU を最適化した、パックされたサーバーでのフレーバーの割り当てを図解しています。



デフォルトのフレーバーは、コモディティーサーバーハードウェアの標準的な設定に推奨されます。使用率を最大限に高めるには、フレーバーをカスタマイズするか、新規フレーバーを作成して、インスタンスのサイズを利用可能なハードウェアに合わせる必要がある場合があります。

可能な場合には、1 フレーバーにつき 1 仮想 CPU の設定に制限してください。種別 1 のハイパーバイザーは、1 仮想 CPU で設定されている仮想マシンにはより簡単に CPU 時間をスケジュールで

きる点に注意することが重要です。たとえば、4 仮想 CPU で設定された仮想マシンに CPU 時間をスケジュールするハイパーバイザーは、実行するタスクに 1 仮想 CPU しか必要でない場合でも、4 つの物理コアが利用可能となるまで待つ必要があります。

ワークロードの特性がハードウェアの選択とフレーバーの設定に影響を及ぼす場合もあります。これは、特にタスクで CPU、メモリー、ハードディスクドライブの要件の比が異なる場合に著明です。フレーバーに関する情報は、**『Instances and Images Guide』**の「Managing Flavors」を参照してください。

3.2.3. 仮想 CPU コア対物理 CPU コアの比

Red Hat OpenStack Platform のデフォルトの割り当て比は 1 物理コアまたは 1 ハイパースレッドコアにつき 16 仮想 CPU です。

以下の表には、システム用に確保される 4 GB を含む使用可能なメモリーの合計に基づいて、1 物理ホストに適した実行可能な仮想マシンの最大数をまとめています。

メモリーの合計	仮想マシン	仮想 CPU の合計
64 GB	14	56
96 GB	20	80
128 GB	29	116

たとえば、60 インスタンスの初期展開をプランニングする場合には、4 つ (3+1) のコンピュートノードが必要になります。通常は、メモリーのボトルネックは CPU のボトルネックよりも一般的ですが、必要な場合には、割り当て比を 1 物理コアあたり 8 仮想 CPU に下げることができます。

3.2.4. メモリーオーバーヘッド

KVM ハイパーバイザーには少量の仮想マシンメモリーオーバーヘッドが必要です。これには、共有可能でないメモリーが含まれます。QEMU/KVM システムの共有可能なメモリーは、ハイパーバイザーにつき 200 MB に丸めることができます。

仮想メモリー	物理メモリーの使用量 (平均)
256	310
512	610
1024	1080

仮想メモリー	物理メモリーの使用量 (平均)
2048	2120
4096	4180

通常は、仮想マシン 1 つあたり 100 MB のハイパーバイザーオーバーヘッドを推定することができます。

3.2.5. オーバーサブスクリプション

メモリーは、ハイパーバイザーのデプロイメントを制限する一要素です。各物理ホスト上で実行できる仮想マシンは、そのホストがアクセス可能なメモリー容量によって制限されます。たとえば、クワッドコア CPU で 256 GB のメモリーを搭載したマシンで 200 を超える 1 GB のインスタンスを実行すると、パフォーマンスが低くなります。このため、CPU コアとメモリーの最適な比率は慎重に決定して、全インスタンス間で分散する必要があります。

3.2.6. 密度

インスタンスの密度

コンピュートを重視したアーキテクチャーでは、インスタンスの密度は低く、CPU とメモリーのオーバーサブスクリプション比も低くなります。特に設計にデュアルソケットハードウェアを使用している場合には、インスタンスの密度が低いと、予想されるスケールをサポートするホストがより多く必要になる可能性があります。

ホストの密度

クワッドソケットプラットフォームを使用することにより、ホスト数が多いデュアルソケット設計に対応できます。このプラットフォームは、ホストの密度を低くし、ラック数を増やします。この設定は、電源接続数などのネットワークの要件や冷却の要件にも影響を及ぼす場合があります。

電源と冷却の密度

電源と冷却の密度を低くすることは、古いインフラストラクチャーを使用するデータセンターでは重要な考慮事項です。たとえば、2U、3U、4U のサーバー設計の電源と冷却の密度の要件は、ホストの密度が低いため、ブレード、スレッド、IU のサーバー設計よりも低くなる場合があります。

3.2.7. コンピュートのハードウェア

ブレードサーバー

大半のブレードサーバーは、デュアルソケット、マルチコアの CPU をサポートしています。CPU の上限を超えないようにするには、**フルワイド** または **フルハイト** のブレードを選択してください。これらのブレードタイプは、サーバーの密度を低くすることもできます。たとえば、HP BladeSystem や Dell PowerEdge M1000e などの高密度のブレードサー

バーは、10 ラックユニットのみで 16 台のサーバーをサポートします。ハーフハイトのブレードは、フルハイトのブレードの 2 倍の密度で、10 ラックユニットにつき 8 台のみとなります。

1U サーバー

単一のラックユニットを占有する 1U ラックマウント型サーバーは、ブレードサーバーのソリューションよりもサーバーの密度が高い場合があります。1 ラックに 1U サーバーを 40 ユニット使用して、Top-of-Rack (ToR) スイッチ用のスペースを提供することができます。これに対して、フルワイドのブレードサーバーは、1 ラックで 32 台しか使用することができません。

ただし、大手ベンダーの 1U サーバーは、デュアルソケット、マルチコアの CPU 設定のみとなっており、1U ラックマウントのフォームファクターでそれよりも高い CPU 設定をサポートするには、ODM (相手先ブランド設計製造業者) または二次製造業者からシステムを購入してください。

2U サーバー

2U ラックマウント型サーバーは、クワッドソケット、マルチコアの CPU をサポートしますが、それに応じてサーバーの密度が低減します。2U ラックマウント型サーバーの密度は、1U ラックマウントサーバーが提供する密度の半分となります。

大型サーバー

4U サーバーのような大型のラックマウント型サーバーは、多くの場合、より高い CPU キャパシティーを提供し、通常は 4 または 8 もの CPU ソケットをサポートします。これらのサーバーは、拡張性は高いですが、サーバー密度がはるかに低く、大抵は高額です。

スレッドサーバー

スレッドサーバーは、ラックマウント型のサーバーで、複数の独立したサーバーを単一の 2U または 3U 筐体でサポートします。これらのサーバーは、通常の 1U または 2U のラックマウント型サーバーよりも高い密度を提供します。

たとえば、多くのスレッドサーバーは、2U で 4 つの独立したデュアルノードを提供し、合計 CPU ソケットは 8 となります。ただし、各ノードがデュアルソケットに制限されているため、追加のコストや設定の複雑さを補うには十分でない可能性があります。

3.2.8. 追加のデバイス

コンピュートノードに以下のデバイスを追加するかどうかを検討します。

- ハイパフォーマンスなコンピューティングジョブのためのグラフィックス処理装置 (GPU)
- 暗号化ルーチンでエントロピーのスターベーションが発生するのを回避するためのハードウェアベースの乱数生成器。乱数生成器は、インスタンスのイメージプロパティーを使用して、インスタンスに追加することができます。デフォルトのエントロピーソースは **`/dev/random`** です。
- データベース管理システムの読み取り/書き込み時間を最大限にするための一時ストレージ用 SSD
- ホストアグリゲートは、同じ特性 (例: ハードウェアの類似性など) を共有するホストをグループ化することによって機能します。クラウドデプロイメントに特化したハードウェアを追加すると、各ノードのコストが高くなる場合があるので、追加のカスタマイズが必要なのが全コンピュートノードか、サブセットのみかを検討してください。

3.3. ストレージのリソース

3.3. ストレージのリソース

クラウドを設計する際には、選択するストレージソリューションは、パフォーマンス、キャパシティー、可用性、相互運用性などのデプロイメントの重要な面に影響を及ぼします。

ストレージソリューションを選択する際には、以下の要素を考慮してください。

3.3.1. 一般的な考慮事項

アプリケーション

クラウドストレージソリューションを効果的に使用するには、アプリケーションが下層のストレージサブシステムを認識する必要があります。ネイティブで利用可能なレプリケーションが使用できない場合には、オペレーターがアプリケーションを設定してレプリケーションサービスを提供するようにできる必要があります。

下層のストレージシステムを検出できるアプリケーションは、多様なインフラストラクチャーで機能することが可能です。インフラストラクチャーの違いにかかわらず、基本的な操作は変わりません。

I/O

入出力パフォーマンスのベンチマークは、予想されるパフォーマンス水準のベースラインを提供します。ベンチマークの結果データは、異なる負荷がかかった状態の動作をモデリングして、適切なアーキテクチャーを設計するのに役立ちます。

アーキテクチャーのライフサイクル中には、異なる時点におけるシステムの健全性を記録するのに、より小さな、スクリプト化されたベンチマークが役立ちます。スクリプト化されたベンチマークから取得したデータは、スコープを定義し、組織のニーズをより深く理解するのに役立ちます。

相互運用性

ハードウェアまたはストレージ管理プラットフォームが OpenStack のコンポーネント (KVM ハイパーバイザーなど) と相互運用可能であることを確認してください。これは、インスタンスの短期ストレージに使用できるかどうかに影響を及ぼします。

セキュリティー

データセキュリティー設計は、SLA、法律上の要件、業界の規制、担当者またはシステムに必要な認定に基づいて異なる面に重点を置くことができます。データのタイプに応じて、HIPPA、ISO9000、SOX のコンプライアンスを検討してください。組織によっては、アクセス制御レベルも考慮する必要があります。

3.3.2. OpenStack Object Storage (swift)

可用性

対象となるオブジェクトデータに必要な可用性レベルを提供するオブジェクトストレージリソースプールを設計します。必要なレプリカ数に対応するラックレベルとゾーンレベルの設計を考慮してください。デフォルトのレプリカ数は 3 です。各データレプリカは、特定のゾーンに対応する電源、冷却、ネットワークリソースを備えた別々のアベイラビリティーゾーンに配置する必要があります。

OpenStack Object Storage サービスは、特定数のデータレプリカをオブジェクトとしてリソースノードに配置します。これらのレプリカは、クラスター内の全ノード上に存在するコンシステントハッシュリングに基づいてクラスター全体で分散されます。また、オブジェクトノードに保管されたデータへのアクセスを提供する Object Storage プロキシー

サーバーのプールは各アベイラビリティーゾーンにサービスを提供する必要があります。

レプリカ数に対し必要最小の正常な応答を提供するのに十分な数のゾーンで Object Storage システムを設計してください。たとえば、Swift クラスターに 3 つのレプリカを設定する場合、Object Storage クラスター内に設定するゾーンの推奨される数は 5 です。

より少ないゾーンを使用するソリューションをデプロイすることはできますが、一部のデータが利用できなくなる可能性があり、クラスター内に保管されている一部のオブジェクトに対する API 要求が失敗する場合があります。このため、Object Storage クラスター内のゾーン数を必ず考慮に入れるようにしてください。

各リージョンのオブジェクトプロキシーは、ローカルの読み取り/書き込みアフィニティーを活用して、可能な場合にはローカルストレージリソースがオブジェクトへのアクセスを円滑化するようにすべきです。アップストリームの負荷分散機能をデプロイして、プロキシーサービスが複数のゾーン間で分散されるようにするとよいでしょう。サービスの地理的分散を補助するサードパーティーのソリューションが必要な場合もあります。

Object Storage クラスター内のゾーンは、論理的な分割で、 単一のディスク、単一のノード、ノードのコレクション、複数のラック、複数の DC で構成することができます。Object Storage クラスターが利用可能な冗長ストレージシステムを提供中にスケーリングができるようにする必要があります。レプリカ、保有期間、および特定のゾーンのストレージの設計に影響を及ぼすその他の要素に応じて異なるストレージポリシーを設定する必要がある場合があります。

ノードストレージ

OpenStack Object Storage 用のハードウェアリソースを設計する際の主要な目的は、各リソースノードのストレージ容量を最大限に増やしつつ、テラバイトあたりのコストを最小限に抑えることです。これには多くの場合、 多数のスピニングディスクを保持することが可能なサーバーを活用する必要があります。ストレージがアタッチ済みの 2U サーバーフォームファクターや、多数のドライブを保持することが可能な外部シャーシを使用することができます。

OpenStack Object Storage の一貫性と分断耐性の特性により、データが最新の状態に維持され、ハードウェアの障害発生時にも特別なデータレプリケーションデバイスの必要なく存続します。

パフォーマンス

Object Storage のノードは、要求数によってクラスターのパフォーマンスが妨げられないように設計すべきです。Object Storage サービスは、通信の多いプロトコルです。このため、コア数の高い複数のプロセッサーを使用して、I/O 要求がサーバーに殺到しないようにします。

重み付けとコスト

OpenStack Object Storage は、Swift のリング内での重み付けによりドライブを組み合わせることができます。Swift のストレージクラスターを設計する際には、費用対効果が最も高いストレージソリューションを使用することができます。

多くのサーバーシャーシは、4U のラックスペースで 60 以上のドライブを収容することができます。このため、テラバイトあたりのコストを最低限に抑えて、ストレージの容量を最大化することができます。ただし、Object Storage ノードには、RAID コントローラーの使用はお勧めできません。

スケーリング

ストレージソリューションを設計する際には、Object Storage サービスが必要とする partition power を決定した上で作成可能なパーティション数を決定します。Object Storage は全ストレージクラスターデータを分散しますが、各パーティションは複数のディスクに

またがることはできないので、最大パーティション数は、ディスクの数より大きくすることはできません。

たとえば、最初のディスクが 1 つと partition power が 3 のシステムの場合には 8 パーティション (2^3) を設けることができます。2 番目のディスクを追加すると、各ディスクに 4 パーティションに分割できることになります。1 パーティションにつき 1 ディスクの制限により、このシステムでは、8 よりも多いディスクを使用できず、スケーラビリティーが制限されますが、最初のディスクが 1 つで partition power が 10 のシステムでは、最大で 1024 (2^{10}) パーティションを設けることができます。

システムバックエンドストレージの容量を増やす場合には必ず、パーティションマップが全ストレージノードにデータを再分散します。このレプリケーションは極めて大量のデータセットで構成されることがあり、その場合には、テナントからのデータへのアクセスと競合が発生しないバックエンドレプリケーションリンクを使用すべきです。

より多くのテナントがクラスター内でデータにアクセスし、データセットが拡大している場合には、フロントエンドの帯域幅を追加して、データアクセス要求に対応できるようにする必要があります。Object Storage クラスターに フロントエンドの帯域幅を追加するには、テナントがデータへのアクセスに使用できる Object Storage プロキシーとともに、プロキシー層のスケーリングを可能にする高可用性ソリューションを設計する必要があります。

テナントとコンシューマーがクラスター内に保管されているデータにアクセスするために使用するフロントエンドの負荷分散層を設計するべきです。この負荷分散層は、複数のゾーン/リージョンにわたって、あるいは地理的な境界を越えても分散することができます。

場合によっては、プロキシーサーバーとストレージノードの間の要求に対応するネットワークリソースに帯域幅と容量を追加する必要があります。このため、ストレージノードとプロキシーサーバーへのアクセスを提供するネットワークアーキテクチャーは、スケーラブルである必要があります。

3.3.3. OpenStack Block Storage (cinder)

可用性と冗長性

アプリケーションに必要な入出力毎秒 (IOPS) により、RAID コントローラーを使用すべきかどうかと、必要な RAID レベルが決定します。冗長性については、RAID の冗長設定 (RAID 5、RAID 6 など) を使用すべきです。ブロックストレージボリュームの自動レプリケーションのような一部の特殊な機能には、より高い要求を処理するのにサードパーティーのプラグインまたはエンタープライズブロックストレージソリューションが必要となる場合があります。

Block Storage に対する要求が極めて高い環境では、複数のストレージプールを使用すべきです。 各デバイスプール内の全ハードウェアノードに同様のハードウェア設計とディスク設定を適用すべきです。このような設計により、アプリケーションは多様な冗長性、可用性、パフォーマンス特性を提供する幅広い Block Storage プールにアクセスできるようになります。

ネットワークアーキテクチャーには、インスタンスが利用可能なストレージリソースを使用するための East-West (水平方向) の帯域幅も考慮すべきです。選択するネットワークデバイスは、大型のデータブロックを転送するためのジャンボフレームをサポートするべきです。場合によっては、インスタンスと Block Storage リソースの間の接続性を提供してネットワークリソースにかかる負荷を軽減するための追加の専用バックエンドストレージネットワークを作成する必要がある可能性があります。

複数のストレージプールをデプロイする場合には、複数のリソースノードにストレージを

プロビジョニングする Block Storage スケジューラーに対する影響を考慮する必要があります。アプリケーションが、特定のネットワーク、電源、冷却用のインフラストラクチャーを使用して、複数のリージョンのボリュームをスケジューリングできるようにします。この設計により、テナントは、複数のアベイラビリティーゾーンに分散する耐障害性のアプリケーションをビルドすることが可能となります。

Block Storage リソースノードに加えて、ストレージノードをプロビジョニングしてそのノードへのアクセスを提供する機能を果す API および関連サービスの高可用性と冗長性に対応する設計を行うことも重要となります。高可用性の REST API サービスを提供してサービスが中断されないようにするには、ハードウェアまたはソフトウェアのロードバランサーの層を設計すべきです。

Block Storage ボリュームにサービスを提供し、状態を保存する機能を果すバックエンドストレージサービスへのアクセスを提供する追加の負荷分散層をデプロイする必要がある場合があります。MariaDB や Galera などの Block Storage データベースを保管するには、高可用性ソリューションを設計すべきです。

アタッチするストレージ

Block Storage サービスは、ハードウェアベンダーによって開発されたプラグインドライバーを使用するエンタープライズストレージソリューションを活用することができます。エンタープライズプラグインの多くには、OpenStack Block Storage がそのまま使用できる状態で同梱されており、それ以外ではサードパーティーチャンネルから入手できるようになっています。

汎用クラウドは通常、大半の Block Storage ノードでダイレクトアタッチストレージを使用します。 このため、テナントに追加のサービスレベルを提供する必要がある場合があります。このようなレベルは、エンタープライズストレージソリューションしか提供できない可能性があります。

パフォーマンス

より高いパフォーマンスが要求される場合には、ハイパフォーマンス RAID ボリュームを使用することができます。極限のパフォーマンスを実現するには、高速のソリッドステートドライブ (SSD) ディスクを使用することができます。

プール

Block Storage プールでは、テナントがアプリケーション用に適切なストレージソリューションを選択できるようにすべきです。異なる種別のストレージプールを複数作成して、Block Storage サービス向けの高度なストレージスケジューラーを設定することにより、さまざまなパフォーマンスレベルと冗長性のオプションを備えたストレージサービスの大きなカタログをテナントに提供することができます。

スケーリング

Block Storage プールをアップグレードして、Block Storage サービス全体を中断せずにストレージ容量を追加することができます。適切なハードウェアとソフトウェアをインストール/設定して、プールにノードを追加します。この後に、新規ノードがメッセージバスを使用して適切なストレージプールにレポーティングするように設定することができます。

Block Storage ノードは、新規ノードがオンラインで利用可能な場合にスケジューラーサービスへノードが稼動中であることをレポーティングするので、テナントは新規ストレージリソースを即時に使用することができます。

場合によっては、インスタンスからの Block Storage への要求によって利用可能なネットワーク帯域幅がすべて使い果たされてしまうことがあります。このため、ネットワークインフラストラクチャーが Block Storage リソースに対応し、容量と帯域幅をシームレスに追加できるように設計すべきです。

これには大抵、動的なルーティングプロトコルまたはダウンストリームのデバイスに容量を追加するための高度なネットワーキングソリューションが必要となります。フロントエンドおよび バックエンドのストレージネットワーク設計には、容量と帯域幅を迅速かつ容易に追加するための機能を含めるべきです。

3.3.4. ストレージハードウェア

容量

ノードのハードウェアは、クラウドサービス用に十分なストレージに対応し、かつデプロイ後に容量を追加できるようにする必要があります。ハードウェアノードは、RAID コントローラーカードに依存せずに、安価なディスクを多数サポートするべきです。

ハードウェアベースのストレージパフォーマンスと冗長性を提供するには、ハードウェアノードが高速ストレージソリューションと RAID コントローラーカードもサポート可能であるべきです。破損したアレイを自動修正するハードウェア RAID コントローラーを選択すると、劣化または破損したストレージデバイスの取り替えや修復に役立ちます。

接続性

ストレージソリューションにイーサネット以外のストレージプロトコルを使用する場合は、 ハードウェアがそのプロトコルに対応可能であることを確認してください。中央集中型ストレージアレイを選択する場合には、ハイパーバイザーがそのイメージストレージ用のストレージアレイに接続可能であることを確認してください。

コスト

ストレージは、全システムコストの中で高い割合を占めます。ベンダーのサポートが必要な場合には、商用ストレージソリューションが推奨されますが、より高額な費用を伴います。初期投資額を最小限に抑える必要がある場合には、コモディティーハードウェアをベースとしたシステムを設計することができます。ただし、初期費用で節約しても、継続的なサポート費用と非互換性のリスクが高くなる可能性があります。

ダイレクトアタッチストレージ

ダイレクトアタッチストレージ (DAS) は、サーバーのハードウェア選択を左右し、ホストの密度、インスタンスの密度、電源の密度、ハイパーバイザー、および管理ツールに影響を及ぼします。

スケーラビリティー

スケーラビリティーは、どの OpenStack クラウドにおいても重要な考慮事項です。実装の最終的な想定サイズを予測するのが難しい場合があります。 将来の拡張およびユーザーの需要に対応するには、初期デプロイメントを大きく設計することを検討してください。

拡張性

拡張性は、ストレージソリューションの主要なアーキテクチャー要素です。50 PB にまで拡張するストレージソリューションは、10 PB までにしか拡張できないソリューションよりも拡張性が高いと見なされます。このメトリックは、ワークロードの増加に応じたソリューションのパフォーマンスの尺度であるスケーラビリティーとは異なります。

たとえば、開発プラットフォーム用クラウドのストレージアーキテクチャーには、商用のクラウドと同じ拡張性とスケーラビリティーは必要ない場合があります。

耐障害性

Object Storage リソースノードには、ハードウェアの耐障害性や RAID コントローラーは必要ありません。Object Storage サービスはゾーン間でのレプリケーションをデフォルトで提供するので、Object Storage のハードウェアには、耐障害性の計画は必要ありません。

ブロックストレージノード、コンピュートノード、およびクラウドコントローラーには、ハードウェア RAID コントローラーやさまざまなレベルの RAID 設定を使用してハードウェアレベルで耐障害性を組込むべきです。RAID レベルは、クラウドのパフォーマンスおよび可用性の要件と一貫している必要があります。

リンク

インスタンスおよびイメージのストレージの地理的な場所は、アーキテクチャー設計に影響を及ぼす場合があります。

パフォーマンス

Object Storage サービスを実行するディスクは、高速なパフォーマンスのディスクである必要はありません。このため、ストレージのテラバイトあたりの費用対効果を最大限に高めることができます。ただし、Block Storage サービスを実行するディスクには、パフォーマンスを強化する機能を使用すべきです。これには、 高パフォーマンスの Block Storage プールを提供するための SSD やフラッシュストレージが必要な場合があります。

インスタンス用に短期間使用するディスクのストレージパフォーマンスも考慮すべきです。コンピュートプールに高使用率の短期ストレージが必要な場合や、極めて高いパフォーマンスが要求される場合には、Block Storage 用にデプロイするのを同様のハードウェアソリューションをデプロイすべきです。

サーバーの種別

DAS を含むスケールアウト型のストレージアーキテクチャーは、サーバーのハードウェア選択を左右します。また、このアーキテクチャーは、ホストの密度、インスタンスの密度、電源の密度、ハイパーバイザー、管理ツールなどにも影響を及ぼします。

3.3.5. Ceph Storage

外部ストレージに Ceph を検討している場合には、Ceph クラスターバックエンドは、妥当なレイテンシーで、並行稼動に必要とされる仮想マシン数を処理できるようにサイジングする必要があります。I/O 操作の 99% を書き込み 20 ミリ秒以下、読み取り 10 ミリ秒以下に維持できることが、サービスレベルの許容範囲となっています。

Rados Block Device (RBD) あたりの最大帯域幅を設定するか、保証される最小コミットメントを指定することにより、I/O スパイクを他の仮想マシンから分離することが可能です。

3.4. ネットワークリソース

ネットワークの可用性は、クラウドデプロイメント内のハイパーバイザーにとって極めて重要です。たとえば、ハイパーバイザーが各ノードで数台の仮想マシンのみをサポートする場合には、 アプリケーションには高速のネットワークは必要なく、1 GB のイーサネットリンクを 1 つまたは 2 つ使用することができますが、アプリケーションが高速のネットワークを必要とする場合や、ハイパーバイザーが各ノードで多数の仮想マシンをサポートする場合には、10 GB のイーサネットリンクを 1 つまたは 2 つ使用することを推奨します。

標準的なクラウドのデプロイメントは、従来のコアネットワークトポロジーが通常必要とする量を上回る P2P 通信を使用します。仮想マシンはクラスター全体で無作為にプロビジョンされますが、同じネットワーク上にあるかのように相互通信する必要があります。この要件により、ネット

ワークの境界とコアの間のリンクがオーバーサブスクライブされるので、従来のコアネットワークトポロジーではネットワーク速度の低下やパケット損失が発生する可能性があります。

3.4.1. サービスの分離

OpenStack クラウドには、従来複数のネットワークセグメントがあります。各セグメントは、クラウド内のリソースへのアクセスをオペレーターやテナントに提供します。ネットワークサービスにも、他のネットワークから分離したネットワーク通信パスが必要です。サービスを分離してネットワークを分けると、機密データをセキュリティー保護して、サービスへの不正なアクセスから守るのに役立ちます。

推奨される最小限の分離を行うには、以下のネットワークセグメントが必要です。

- クラウドの REST API にアクセスするのにテナントとオペレーターが使用するパブリックネットワークセグメント。通常このネットワークセグメントに接続する必要があるのは、クラウド内のコントローラーノードと swift プロキシーのみです。場合によっては、このネットワークセグメントはハードウェアのロードバランサーやその他のネットワークデバイスによりサービスが提供されることもあります。

- クラウド管理者がハードウェアリソースを管理し、設定管理ツールが新しいハードウェアにソフトウェアとサービスをデプロイするために使用する管理ネットワークセグメント。場合によっては、このネットワークセグメントは、相互通信が必要なメッセージバスやデータベースサービスなどの内部サービスにも使用されることがあります。

  セキュリティー要件により、このネットワークセグメントは不正アクセスから保護することを推奨します。このネットワークセグメントは、通常クラウド内の全ハードウェアノードと通信する必要があります。

- アプリケーションおよびコンシューマーが物理ネットワークへのアクセスを提供し、ユーザーがクラウド内で稼働中のアプリケーションにアクセスするために使用するアプリケーションネットワークセグメント。このネットワークは、物理ネットワークから分離して、クラウド内のハードウェアリソースとは直接通信しないようにする必要があります。

  このネットワークセグメントは、アプリケーションデータをクラウド外部の物理ネットワークに転送するコンピュートのリソースノードとネットワークゲートウェイサービスが通信に使用することができます。

3.4.2. 一般的な考慮事項

セキュリティー

ネットワークサービスを分離し、トラフィックが不要な場所を経由せずに正しい送信先に流れるようにします。

以下に例示する要素を考慮してください。

- ファイアウォール

- 分離されたテナントネットワークを結合するためのオーバーレイの相互接続

- 特定のネットワークを経由または回避

ネットワークをハイパーバイザーにアタッチする方法により、セキュリティーの脆弱性にさらされる場合があります。ハイパーバイザーのセキュリティーホールが悪用されるリスクを軽減するには、他のシステムからのネットワークを分離して、 そのネットワークのインスタンスは専用のコンピュートノードにスケジューリングします。このような分離によ

り、セキュリティーが侵害されたインスタンスから攻撃者がネットワークに侵入するのを防ぎます。

キャパシティープランニング

クラウドのネットワークには、キャパシティーと拡張の管理が必要です。キャパシープランニングには、ネットワークサーキットとハードウェアの購入と、月/年単位で測定可能なリードタイムが含まれます。

複雑性

複雑なネットワーク設計は、メンテナンスとトラブルシューティングが難しくなる場合があります。デバイスレベルの設定でメンテナンスに関する懸念を軽減し、自動化ツールでオーバーレイネットワークの処理を行うことができますが、機能と特化したハードウェア間で従来とは異なる相互接続を避けるか文書化して、サービスの停止を防いでください。

設定エラー

誤った IP アドレス、VLAN、またはルーターを設定すると、ネットワークのエリアまたはクラウドインフラストラクチャー全体でもサービス停止の原因となる場合があります。ネットワーク設定を自動化して、ネットワークの可用性を中断させる可能性のあるオペレーターのエラーを最小限に抑えてください。

非標準の機能

クラウドネットワークを設定して、ベンダー固有の機能を利用するとリスクが高くなる可能性があります。

たとえば、マルチリンクアグリゲーション (MLAG) を使用して、ネットワークのアグリゲータースイッチレベルで冗長性を提供します。MLAG は標準のアグリゲーション形式ではなく、各ベンダーが機能の専用フレーバーを実装しています。MLAG アーキテクチャーは、全スイッチベンダー間で相互運用可能ではないため、ベンダーロックインに陥り、ネットワークコンポーネントのアップグレード時に遅延や問題が発生する場合があります。

単一障害点

アップストリームのリンクまたは電源が 1 つしかないために、ネットワークに単一障害点 (SPOF) がある場合には、障害の発生時にネットワークが停止する可能性があります。

チューニング

クラウドネットワークを設定して、リンク損失、パケット損失、パケットストーム、ブロードキャストストーム、ループを最小限に抑えます。

3.4.3. ネットワークハードウェア

OpenStack クラウドをサポートするネットワークハードウェアのアーキテクチャーには、すべての実装に適用可能な単一のベストプラクティスはありません。ネットワークハードウェアの選択における重要な考慮事項には以下の点が含まれます。

可用性

クラウドノードのアクセスが中断されないようにするには、ネットワークアーキテクチャーで単一障害点を特定し、適切な冗長性や耐障害性を提供する必要があります。

- ネットワークの冗長性は、冗長電源またはペアのスイッチを追加することによって提供することができます。

- ネットワークインフラストラクチャーには、LACP や VRRP などのネットワークプロトコルを使用して、高可用性のネットワーク接続を実現することができます。
- OpenStack API およびクラウド内のその他のサービスを高可用性で提供するには、ネットワークアーキテクチャー内に負荷分散ソリューションを設計するべきです。

接続性

OpenStack クラウド内の全ノードにはネットワーク接続が必要です。ノードが複数のネットワークセグメントにアクセスする必要がある場合もあります。クラウド内のすべての垂直/水平方向のトラフィックに十分なリソースを使用できるようにするには、設計に十分なネットワーク容量と帯域幅を組み入れる必要があります。

ポート

どのような設計にも、必要なポートを備えたネットワーク用ハードウェアが必要です。

- ポートを提供するのに必要な物理的なスペースがあることを確認します。Compute やストレージコンポーネント用のラックスペースが残るので、ポートの密度が高いほど望ましいですが、適切な可用性のポートを提供することにより、障害ドメインを防ぎ、電源密度に役立ちます。スイッチの密度が高いほど、費用が高額となり、必要がない場合にはネットワークを過剰設計しないことが重要となる点も考慮に入れるべきです。
- ネットワーク用のハードウェアは、予定されているネットワークスピードをサポートする必要があります。例: 1 GbE、10 GbE、40 GbE (または 100 GbE)

電源

選択したネットワークハードウェアに必要な電源を物理データセンターが提供するようにします。たとえば、リーフ/スパイン型ファブリック内のスパインスイッチまたは End of Row (EoR) スイッチは、十分な電源を供給しない可能性があります。

スケーラビリティー

ネットワーク設計には、スケーラブルな物理/論理ネットワークを組込むべきです。ネットワークハードウエアは、ハードウェアノードが必要とするインターフェースタイプとスピードを提供するべきです。

3.5. パフォーマンス

OpenStack デプロイメントのパフォーマンスは、インフラストラクチャーとコントローラーサービスに関連する複数の要素によって左右されます。ユーザーの要件は、全般的なネットワークパフォーマンス、Compute リソースのパフォーマンス、ストレージシステムのパフォーマンスに分けることができます

システムがスピードダウンせず、一貫して動作しているときでも、パフォーマンスのベースラインの履歴を保持するようにします。ベースラインの情報が利用できると、パフォーマンス問題の発生時に比較するためのデータが必要な場合に参照するのに役立ちます。

「OpenStack Telemetry (ceilometer)」 以外にも、外部のソフトウェアを使用してパフォーマンスをトラッキングすることができます。Red Hat OpenStack Platform の運用ツールリポジトリーには、次のツールが含まれています。

- collectd
- Graphite-web
- InfluxDB

- Grafana

3.5.1. ネットワークのパフォーマンス

ネットワーク要件はパフォーマンス能力を決定するのに役立ちます。たとえば、小型のデプロイメントでは、1 ギガビットイーサネット (GbE) のネットワークを採用し、複数の部署/多数のユーザーにサービスを提供する大型のデプロイメントでは 10 GbE のネットワークを使用するとよいでしょう。

インスタンスを実行するパフォーマンスは、これらのネットワークスピードによって左右されます。異なるネットワーク機能を実行する OpenStack 環境を設計することが可能です。異なるインターフェース速度を使用することにより、OpenStack 環境のユーザーは、自分の目的に適したネットワークを選択することができます。

たとえば、Web アプリケーションのインスタンスは、1 GbE 対応の OpenStack Networking を使用するパブリックネットワーク上で実行することができ、バックエンドデータベースには、10GbE 対応の OpenStack Networking を使用してデータのレプリケーションを行うことができます。場合によっては、設計にリンクアグリゲートを組み込んでスループットを拡大することが可能です。

ネットワークのパフォーマンスは、クラウド API にフロントエンドサービスを提供するハードウェアロードバランサーを実装することによって強化することができます。ハードウェアロードバランサーは、必要な場合に SSL 終了を実行することができます。SSL オフロードを実装する場合には、選択したデバイスの SSL オフロード機能を確認することが重要となります。

3.5.2. コンピュートノードのパフォーマンス

CPU、メモリー、ディスクの種別など、コンピュートノードに使用するハードウェア仕様は、インスタンスのパフォーマンスに直接影響します。OpenStack サービスの調整可能パラメーターもパフォーマンスに直接影響する場合があります。

たとえば、OpenStack Compute のデフォルトのオーバーコミット比は CPU が 16:1、メモリーが 1.5 です。これらの高い比によって、「noisy-neighbor」アクティビティーが増える可能性があります。Compute 環境のサイズは、慎重に設定して、このようなシナリオを回避し、使用率が高くなった場合には、必ず環境をモニタリングする必要があります。

3.5.3. Block Storage ホストのパフォーマンス

Block Storage では、NetApp や EMC などのエンタープライズバックエンドシステム、または Ceph などのスケールアウトストレージを使用したり、Block Storage ノード内でストレージを直接アタッチしたりする機能を活用することができます。

Block Storage は、トラフィックがホストネットワークを通過できるようにデプロイすることが可能です。このような構成は、フロントサイドの API トラフィックのパフォーマンスに影響を及ぼす可能性があります。またホストネットワークのトラフィックがフロントサイドの API トラフィックによって悪影響を受ける場合もあります。このため、コントローラーおよびコンピュートホストでは、専用のインターフェースを備えたデータストレージネットワークを使用すること検討してください。

3.5.4. Object Storage ホストのパフォーマンス

ユーザーは通常、ハードウェアロードバランサーの背後で動作するプロキシーサービスを使用して Object Storage にアクセスします。デフォルトでは、耐障害性の高いストレージシステムが保管されているデータを複製し、これによりシステム全体のパフォーマンスが影響を受けます。この場合、ストレージネットワークアーキテクチャー全体にわたってネットワーク容量を 10 GbE 以上に設定することを推奨します。

3.5.5. コントローラーノード

コントローラーノードは、エンドユーザーに管理サービスを提供し、クラウド運用のためのサービスを内部で提供します。コントローラーのインフラストラクチャーに使用するハードウェアの設計は慎重に行うことが重要となります。

コントローラーは、サービス間のシステムメッセージ用にメッセージキューサービスを実行します。インスタンスメッセージングにおけるパフォーマンス上の問題により、インスタンスのスピンアップ/削除、新規ストレージボリュームのプロビジョニング、ネットワークリソースの管理などの運用機能に遅延が発生する場合があります。このような遅延は、特に自動スケーリング機能を使用している場合に、一部の条件に対してアプリケーションが反応する能力に影響を及ぼす可能性があります。

コントローラーノードは、複数の同時ユーザーのワークロードを処理できるようにする必要もあります。API と Horizon のサービスのロードテストを予め行って、顧客のサービス信頼性を向上するようにしてください。

クラウド内で OpenStack およびエンドユーザーに全サービスの認証と承認を提供する OpenStack Identity サービス (keystone) について考慮することが重要です。このサービスのサイズを適切に設定していない場合には、全体的なパフォーマンスの低下をもたらす可能性があります。

モニタリングすべき極めて重要なメトリックには以下が含まれます。

- イメージディスクの使用率
- Compute API の応答時間

3.6. メンテナンスとサポート

環境のサポートとメンテナンスを行うために、OpenStack クラウドの管理には、運用スタッフがアーキテクチャー設計を理解している必要があります。運用スタッフとエンジニアリングスタッフのスキルレベルおよび役割分担はその環境のサイズと目的によって異なります。

- 大手のクラウドサービスプロバイダーまたは電気通信プロバイダーでは、特別に訓練を受けた専用の運用組織が管理を行う傾向があります。
- 小規模な実装は、エンジニアリング、設計、運用の機能を併せて担う必要のあるサポートスタッフに依存する傾向があります。

設計の運用オーバーヘッドを低減するための機能を組み込んだ場合には、一部の運用機能を自動化することができます。

設計は、サービスレベルアグリーメント (SLA) の条件によって直接の影響を受けます。SLA はサービスの可用性のレベルを定義し、契約上の義務を果たさない場合には、通常ペナルティーがあります。設計に影響を及ぼす SLA の条件には、以下が含まれます。

- 複数のインフラストラクチャーサービスと高可用性のロードバランサーを意味する、API の可用性の保証
- スイッチの設計に影響を及ぼし、冗長のスイッチまたは電源が必要な可能性のあるネットワークのアップタイムの保証
- ネットワークの分離または追加のセキュリティーメカニズムを暗示する、ネットワークセキュリティーポリシー要件

3.6.1. バックアップ

バックアップと復元のストラテジー、データの評価、階層型のストレージ管理、保持ストラテジー、データの配置、ワークフローの自動化によって設計が影響を受ける場合があります。

3.6.2. ダウンタイム

有効なクラウドアーキテクチャーは、以下をサポートする必要があります。

- 計画的なダウンタイム (メンテナンス)
- 計画外のダウンタイム (システムの障害)

たとえば、コンピュートホストで障害が発生した場合に、スナップショットからインスタンスを復元したり、インスタンスを再起動したりすることができます。ただし、高可用性には、共有ストレージなどの追加のサポートサービスをデプロイするか、信頼できるマイグレーションパスを設計する必要がある場合があります。

3.7. 可用性

OpenStack は、少なくとも 2 台のサーバーを使用している場合に高可用性のデプロイメントを提供することができます。サーバーは、RabbitMQ メッセージキューサービスおよび MariaDB データベースサービスからの全サービスを実行することが可能です。

クラウド内のサービスをスケーリングする際には、バックエンドサービスもスケーリングする必要があります。サーバーの使用率と応答時間のモニタリング/レポーティングに加えて、システムの付加テストを行うとスケーリングについての意思決定に役立てることができます。

- 単一障害点が生じないようにするには、OpenStack サービスを複数のサーバーにわたってデプロイするべきです。API の可用性は、複数の OpenStack サーバーをメンバーとする高可用性のロードバランサーの背後に配置することによって実現することができます。
- デプロイメントに適切なバックアップ機能を実装するようにします。たとえば、高可用性機能を使用する 2 つのインフラストラクチャーコントローラーノードで構成されるデプロイメントでは、1 つのコントローラーを失っても、もう 1 つのコントローラーからクラウドサービスを実行することが可能です。
- OpenStack のインフラストラクチャーは、サービスの提供に不可欠なので、常に稼働している必要があります。これは、特に SLA に基づいて運用している場合に重要となります。コアインフラストラクチャーに必要なスイッチ数、ルート、電源の冗長性に加え、それらに付随して、高可用性スイッチインフラストラクチャーに多様なルートを提供するためのネットワークボンディングも考慮してください。
- Compute ホストがライブマイグレーションに対応するように設定されていない場合には、1 台の Compute ホストでエラーが発生した際に Compute のインスタンスとそのインスタンスに保管されているデータが失われる可能性があります。Compute ホストのアップタイムを維持するには、エンタープライズストレージまたは OpenStack Block Storage 上で共有ファイルシステムを使用することができます。

サービスの可用性や閾値の制限をチェックして適切な警告を設定するのに外部のソフトウェアを使用することができます。Red Hat OpenStack Platform の運用ツールリポジトリーには、以下のツールが含まれています。

- Sensu
- Uchiwa ダッシュボード

注記

OpenStack で高可用性を使用するためのリファレンスアーキテクチャーは、「Deploying Highly Available Red Hat OpenStack Platform 6 with Ceph Storage」の記事を参照してください。

3.8. セキュリティー

セキュリティードメインには、単一のシステム内で共通する信頼の要件と予測を共有するユーザー、アプリケーション、サーバー、ネットワークが含まれます。セキュリティードメインは、通常同じ認証/承認要件およびユーザーを使用します。

標準的なセキュリティードメインのカテゴリーには、Public、Guest、Management、Data です。ドメインは、OpenStack デプロイメントに個別または統合してマッピングすることができます。たとえば、ゲストとデータのドメインを 1 つの物理ネットワーク内に統合するデプロイメントトポロジーもあれば、ネットワークを物理的に分離するデプロイメントトポロジーもあります。いずれの場合も、クラウドオペレーターは関連するセキュリティー問題について認識している必要があります。

セキュリティードメインは、特定の OpenStack デプロイメントトポロジーにマッピングする必要があります。ドメインおよびドメイン信頼要件は、インスタンスがパブリック、プライベート、ハイブリッドであるかによって異なります。

パブリックドメイン

クラウドインフラストラクチャーの完全に信頼されない領域です。パブリックドメインは、インターネット全体または権限のない複数のネットワークのことを指します。このドメインは、常に信頼できないものと見なす必要があります。

ゲストドメイン

通常は、インスタンス間のトラフィックに使用され、クラウド上でインスタンスによって生成された Compute のデータを処理しますが、API コールなどのクラウドの運用をサポートするサービスによって生成されるデータは対象ではありません。

インスタンスの使用を厳格に制御しない、またはインスタンスに制限なしのインターネットアクセスを許可しているパブリッククラウドプロバイダーおよびプライベートクラウドプロバイダーは、このドメインを信頼できないものと見なすべきです。プライベートクラウドのプロバイダーは、このネットワークを内部と見なす場合があり、インスタンスおよび全クラウドテナント内で信頼をアサートする制御を使用する場合のみ信頼されます。

ドメインの管理

サービスが相互に対話するドメイン。このドメインは、**コントロールプレーン**としても知られています。このドメイン内のネットワークは、設定パラメーター、ユーザー名、パスワードなどの機密データを転送します。大半のデプロイメントでは、このドメインは信頼されているものと見なされます。

データドメイン

ストレージサービスがデータを転送するドメイン。このドメインを通過するデータの大半は、整合性と機密性の要件が高く、デプロイメントの種別によっては、高可用性が要求される場合があります。このネットワークの信頼レベルは、デプロイメントの他の決定事項によって左右されます。

企業内のプライベートクラウドとして OpenStack をデプロイする場合には、デプロイメントは通常ファイアウォールの背後で、既存のシステムを使用する信頼済みのネットワーク内に設定します。クラウドのユーザーは、通常その企業が定義したセキュリティー要件に拘束される従業員です。このようなデプロイメントでは、大半のセキュリティードメインを信頼できます。

ただし、一般向けのロールに OpenStack をデプロイする場合には、ドメインの信頼レベルに関して何も想定することができず、攻撃ベクトルが大幅に増大します。たとえば、API エンドポイントと下層にあるソフトウェアは不正にアクセスしようとしたり、サービスへのアクセスを妨害しようとする悪意のある者に対して脆弱になります。このような攻撃により、データ、機能、評判を失う可能性があります。このようなサービスは、監査や適切なフィルタリングを使用して保護する必要があります。

パブリッククラウドとプライベートクラウドの両方のシステムのユーザーを管理する際にも注意を払う必要があります。Identity サービスは、LDAP などの外部のアイデンティティーバックエンドを使用することができます。これにより、OpenStack 内部でのユーザー管理の負担が軽減されます。ユーザー認証要求には、ユーザー名、パスワード、認証トークンなどの機密情報が含まれるので、API サービスは、SSL 終了を実行するハードウェアの背後に配置すべきです。

3.9. 追加のソフトウェア

標準的な OpenStack デプロイメントには OpenStack 固有のコンポーネントと「サードパーティーのコンポーネント」が含まれます。追加のソフトウェアにはクラスタリング、ロギング、監視、警告などのソフトウェアが含まれます。このため、デプロイメントの設計は、CPU、メモリー、ストレージ、ネットワーク帯域幅などの追加のリソース消費を考慮に入れる必要があります。

クラウドの設計時には、以下の要素を考慮してください。

データベースとメッセージング

下層にあるメッセージキュープロバイダーにより、コントローラーサービスの必要数や耐障害性の高いデータベース機能を提供する技術が影響を受ける場合があります。たとえば、MariaDB と Galera を併用する場合には、レプリケーションサービスは定足数によって左右されます。このため、下層のデータベースは、少なくとも 3 つのノードで構成して、障害が発生した Galera ノードのリカバリーに対応するべきです。

ソフトウェアの機能をサポートするためにノードの数を増やす場合には、ラックのスペースとスイッチポートの密度の両方を考慮してください。

外部のキャッシュ

Memcached は、分散型メモリーオブジェクトキャッシュシステムで、Redis はキーバリューストアです。いずれのシステムも、クラウド内にデプロイして、Identity サービスの負荷を軽減することができます。たとえば、memcached サービスは、トークンをキャッシュし、分散型キャッシュシステムを使用して下層の認証システムからの一部のボトルネックを軽減します。

memcached または Redis を使用しても、アーキテクチャー全体の設計には影響はありません。これらのサービスは通常、OpenStack サービスを提供するインフラストラクチャーノードにデプロイされるのが理由です。

ロードバランシング

多くの汎用デプロイメントは、ハードウェアロードバランサーを使用して高可用性の API アクセスと SSL 終了を提供しますが、HAProxy のようなソフトウェアソリューションも検討することができます。ソフトウェア定義のロードバランシング実装も高可用性を確保する必要があります。

Keepalived または Pacemaker と Corosync を併用するソリューションでソフトウェア定義

の高可用性を設定することができます。Pacemaker と Corosync は、OpenStack 環境内の特定のサービスをベースとする active-active または active-passive の高可用性構成を提供することができます。

これらのアプリケーションには少なくとも 2 コントローラーノードで構成されるデプロイメントで、そのうちの 1 つのコントローラーノードがスタンバイモードでサービスを実行可能である必要があるので、設計に影響を及ぼす可能性があります。

ロギングとモニタリング

ログは、容易に分析できるように一元的に保管するべきです。ログデータの解析エンジンは、一般的な問題を警告/修正するメカニズムを使用して、自動化と問題通知機能を提供することができます。

ツールがアーキテクチャー設計内の既存のソフトウェアおよびハードウェアがツールをサポートしている限りは、基本的な OpenStack のログに加えて外部のロギングまたはモニタリング用ソフトウェアを使用することができます。Red Hat OpenStack Platform の運用ツールリポジトリーには、以下のツールが含まれています。

- Fluentd
- ElasticSearch
- Kibana

3.10. プランニングツール

Cloud Resource Calculator は、キャパシティー要件の計算に役立つルールです。

このツールを使用するには、ハードウェアの詳細をスプレッドシートに入力します。ツールは次に利用可能なインスタンス数の推定値を算出して表示します。これには、フレーバー別の概算も含まれます。

重要

このツールは、お客様の便宜のためのみに提供されており、Red Hat では正式にサポートされていません。

第4章 アーキテクチャー例

本章では、Red Hat OpenStack Platform デプロイメントのアーキテクチャー例のリファレンスを記載します。

> **注記**
>
> アーキテクチャー例はすべて、KVM ハイパーバイザーを使用する Red Hat Enterprise Linux 7.3 上に OpenStack Platform をデプロイすることを前提とします。

4.1. 概要

通常、デプロイメントはパフォーマンスまたは機能性をベースにします。デプロイメントは、デプロイするインフラストラクチャーをベースにすることもできます。

表4.1 機能性またはパフォーマンスをベースとするデプロイメント

例	説明
「汎用アーキテクチャー」	特定の技術的または環境の要件が不明な場合に使用する一般的な高可用性クラウド。このアーキテクチャータイプは、柔軟性が高く、単一の OpenStack コンポーネントを重視しているわけではないので、特定の環境に制限されません。
「コンピュートを重視したアーキテクチャー」	コンピューティング集約型のワークロードを特にサポートする、コンピュートを重視したクラウド。コンピューティング集約型のワークロードは、データのコンピューティング、暗号化、暗号化解除などの CPU 集約型のワークロードを意味する場合もあります。また、インメモリーキャッシュ、データベースサーバーなどのメモリー集約型のワークロードや、CPU とメモリーの両方を集中的に使用するワークロードを意味する場合もあります。通常は、ストレージ集約型またはネットワーク集約型ではありません。このアーキテクチャータイプは、ハイパフォーマンスのコンピュトソースが必要な場合に選択することができます。
「データ分析アーキテクチャー」	Hadoop クラスターなどの大量のデータの管理/分析向けに設計された、パフォーマンス重視のストレージシステム 。このアーキテクチャータイプでは、OpenStack は Hadoop と統合し、Ceph をストレージバックエンドとして使用して Hadoop クラスターを管理します。
「ハイパフォーマンスデータベースアーキテクチャー」	データベースの IO 要件が高いことを想定し、ソリッドステートドライブ (SSD) を使用してデータを処理するハイパフォーマンスストレージシステム。このアーキテクチャータイプは、既存のストレージ環境に使用することができます。

例	説明
「クラウドのストレージとバックアップのアーキテクチャー」	OpenStack デプロイメントで一般的に使用されているクラウドベースのファイルストレージ/共有サービス。このアーキテクチャータイプは、クラウドトラフィックの受信データが送信データを上回る場合にクラウドバックアップアプリケーションを使用します。
「大規模な Web アプリケーション向けのアーキテクチャー」	大規模な Web アプリケーション向けのハードウェアベースのロードバランシングクラスター。このアーキテクチャータイプは、SSL オフロード機能を提供し、テナントネットワークに接続してアドレスの消費を低減し、Web アプリケーションを水平スケーリングします。

4.2. 汎用アーキテクチャー

具体的な技術/環境のニーズが不明な場合には、汎用の高可用性クラウドをデプロイすることができます。この柔軟なアーキテクチャータイプは、単一の OpenStack コンポーネントを重視していないので、特定の環境には制限されません。

このアーキテクチャータイプは、以下を含む潜在的なユースケースの 80% に対応します。

- シンプルなデータベース
- Web アプリケーションランタイム環境
- 共有アプリケーション開発環境
- テスト環境
- スケールアップ型ではなくスケールアウト型の拡張を必要とする環境

このアーキテクチャータイプは、高度なセキュリティーを必要とするクラウドドメインには推奨されません。

注記

インストールおよびデプロイメントに関する説明は、「5章*デプロイメントの情報*」を参照してください。

4.2.1. ユースケースの例

あるインターネット広告会社では、Apache Tomcat、Nginx、MariaDB を含む Web アプリケーションをプライベートクラウドで実行することを検討しています。ポリシー要件を満たすために、クラウドインフラストラクチャーはその会社のデータセンター内で稼働します。

同社は、予想可能な負荷の要件がありますが、毎晩の需要拡大に対応するためのスケーリングを必要としています。現在の環境には、オープンソースの API 環境を稼働するという同社の目標に対応するための柔軟性がありません。

現在の環境は以下のコンポーネントで構成されています。

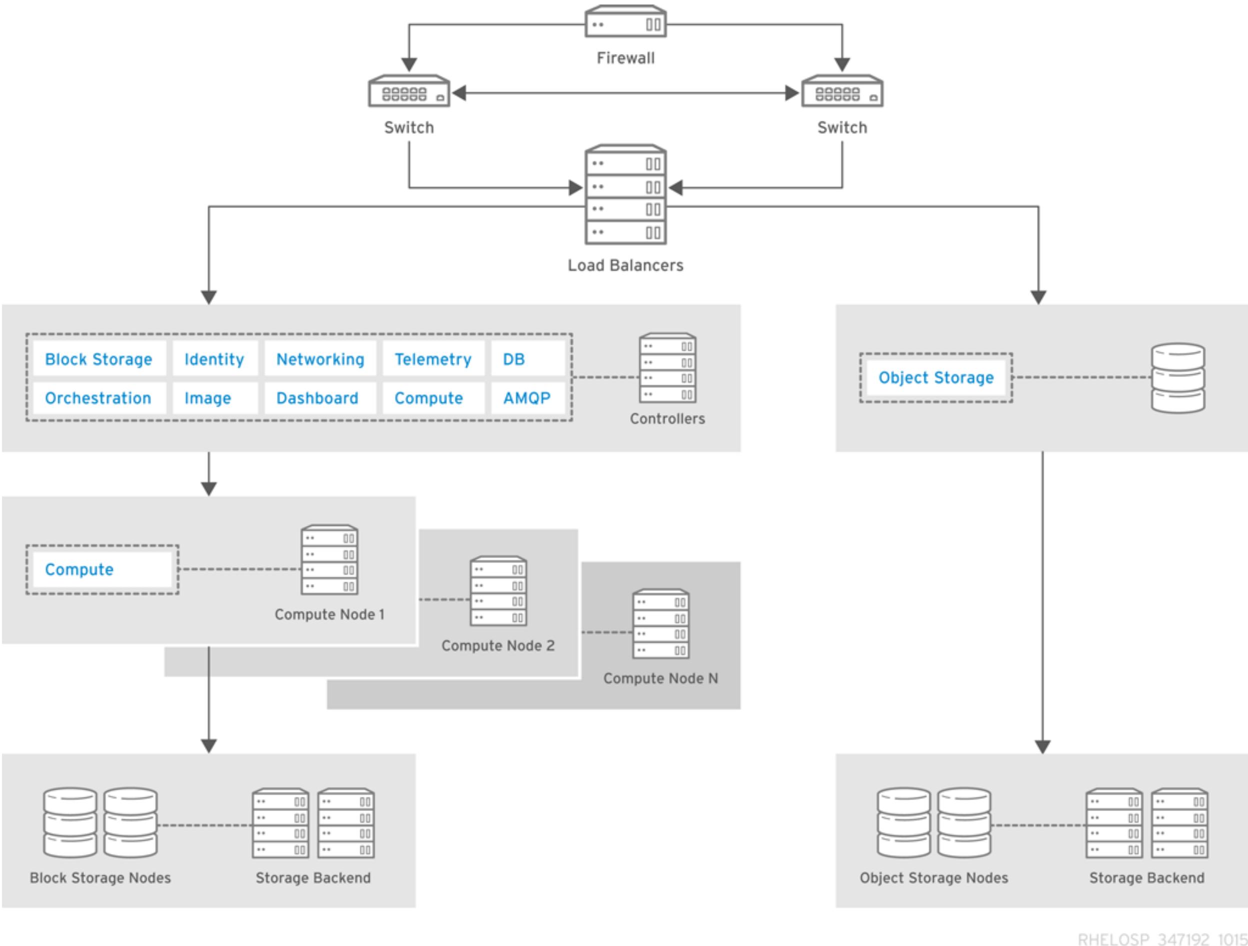
- Nginx および Tomcat をインストールしたシステムが 120 - 140。各システムには 2 つの仮想 CPU と 4 GB のメモリーを搭載。
- 3 ノードの MariaDB および Galera のクラスター。それぞれ 4 つの仮想 CPU と 8 GB のメモリーを搭載。Galera ノードの管理には Pacemaker を使用。

同社は、ハードウェアロードバランサーと、Web サイトにサービスを提供する複数の Web アプリケーションを実行しています。環境のオーケストレーションには、スクリプトと Puppet を併用しています。Web サイトはアーカイブが必要な大量のログデータを毎日生成します。

4.2.2. 設計について

この例のアーキテクチャーには、3 つのコントローラーノードと、少なくとも 8 つの Compute ノードが含まれます。静的オブジェクトには OpenStack Object Storage を、その他すべてのストレージニーズには OpenStack Block Storage を使用します。

OpenStack インフラストラクチャーの高可用性を確保するために、ノードは Red Hat Enterprise Linux 用の Pacemaker アドオンと HAProxy を併用します。



このアーキテクチャーには以下のコンポーネントが含まれます。

- 一般向けのネットワーク接続用のファイアウォール、スイッチ、ハードウェアロードバランサー
- Image、Identity、Networking を実行する OpenStack コントローラーサービスにサポートサービスとして MariaDB と RabbitMQ を組み合わせます。これらのサービスは、少なくとも 3 つのコントローラーノード上で高可用性に設定されます。
- クラウドノードは、Red Hat Enterprise Linux 用の Pacemaker アドオンとともに高可用性に設定されます。

- Compute ノードは、永続ストレージが必要なインスタンスに OpenStack Block Storage を使用します。
- OpenStack Object Storage はイメージなどの静的オブジェクトに対応します。

4.2.3. アーキテクチャーコンポーネント

コンポーネント	説明
Block Storage	インスタンス用の永続ストレージ
Compute コントローラーサービス	Compute の管理とコントローラー上で実行するサービスのスケジューリング
Dashboard	OpenStack の管理用 Web コンソール
Identity	ユーザーおよびテナントの基本的な認証と承認
Image	インスタンスの起動とスナップショットの管理に使用するイメージを保管します。
MariaDB	全 OpenStack コンポーネント用のデータベース。MariaDB サーバーインスタンスは、NetApp や Solidfire などの共有エンタープライズストレージ上のデータを保管します。MariaDB インスタンスに障害が発生した場合には、ストレージは再度他のインスタンスにアタッチして Galera クラスターに再び参加させる必要があります。
ネットワーク	プラグインと Networking API を使用してハードウェアバランサーを制御します。OpenStack Object Storage を拡張する場合には、ネットワーク帯域幅の要件を考慮する必要があります。OpenStack Object Storage は、ネットワーク接続スピードが 10 GbE 以上で実行することを推奨します。
Object Storage	Web アプリケーションサーバーからのログを処理し、アーカイブします。また、Object Storage を使用して静的な Web コンテンツを OpenStack Object Storage コンテナーから移動したり、または OpenStack Image で管理されているイメージをバックアップすることもできます。
Telemetry	その他の OpenStack サービスのモニタリングとレポーティング

4.2.4. Compute ノードの要件

Compute サービスは、各 Compute ノードにインストールされます。

この汎用アーキテクチャーでは、最大で 140 の Web インスタンスを実行可能で、少数の MariaDB インスタンスに 292 の仮想 CPU と 584 GB のメモリーが必要です。ハイパースレッディング対応で、デュアルソケット、ヘキサコア の Intel CPU を搭載した標準的な 1U サーバーで、2:1 CPU オーバーコミット比を前提とする場合、このアーキテクチャーには 8 Compute ノードが必要です。

Web アプリケーションインスタンスは、各 Compute ノードのローカルストレージから起動します。Web アプリケーションインスタンスはステートレスなので、いずれか 1 つのインスタンスにエラーが発生した場合でも、アプリケーションは実行を継続することができます。

4.2.5. ストレージ要件

ストレージには、サーバーにストレージを直接アタッチする、スケールアウト型ソリューションを使用します。たとえば、グリッドコンピューティングソリューションと同様の方法でコンピュートホストにストレージを追加したり、ブロックストレージのみを提供する専用のホストに追加することが可能です。

コンピュートホストにストレージをデプロイする場合には、ハードウェアがストレージとコンピュートのサービスを処理できることを確認してください。

4.3. コンピュートを重視したアーキテクチャー

注記

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

コンピュートを重視したクラウドは、データのコンピューティング、暗号化、暗号化解除などの CPU 集約型のワークロードと、インメモリーキャッシングやデータベースサーバーなどのメモリー集約型のワークロード、またはそれらの両方をサポートします。このアーキテクチャータイプは、通常はストレージやネットワークを集中的には使用せず、コンピュートリソースの能力を求めている顧客が対象です。

コンピュートを重視したワークロードには、以下のようなユースケースが含まれます。

- ハイパフォーマンスコンピューティング (HPC)
- Hadoop または他の分散型データストアを使用したビッグデータの分析
- 継続的インテグレーションまたは継続的デプロイ (CI/CD)
- Platform-as-a-Service (PaaS)
- ネットワーク機能仮想化 (NFV) のシグナル処理

コンピュートを重視した OpenStack クラウドは、ほとんどの場合、永続ブロックストレージを必要とするアプリケーションをホストしないので、通常 RAW ブロックストレージサービスは使用しません。インフラストラクチャーのコンポーネントは共有されないので、ワークロードは利用可能なリソースを必要なだけ多く消費することができます。インフラストラクチャーのコンポーネント

には、高可用性も必要です。この設計は、ロードバランサーを使用します。HAProxy を使用することも可能です。

注記

インストールおよびデプロイメントに関する説明は、「*5章デプロイメントの情報*」を参照してください。

4.3.1. ユースケースの例

ある組織では、研究プロジェクト用に HPC を提供しており、ヨーロッパにある 2 つの既存のコンピュートセンターに加えて、第 3 のコンピュートセンターを追加する必要があります。

以下の表には、各コンピュートセンターが追加すべき要件をまとめています。

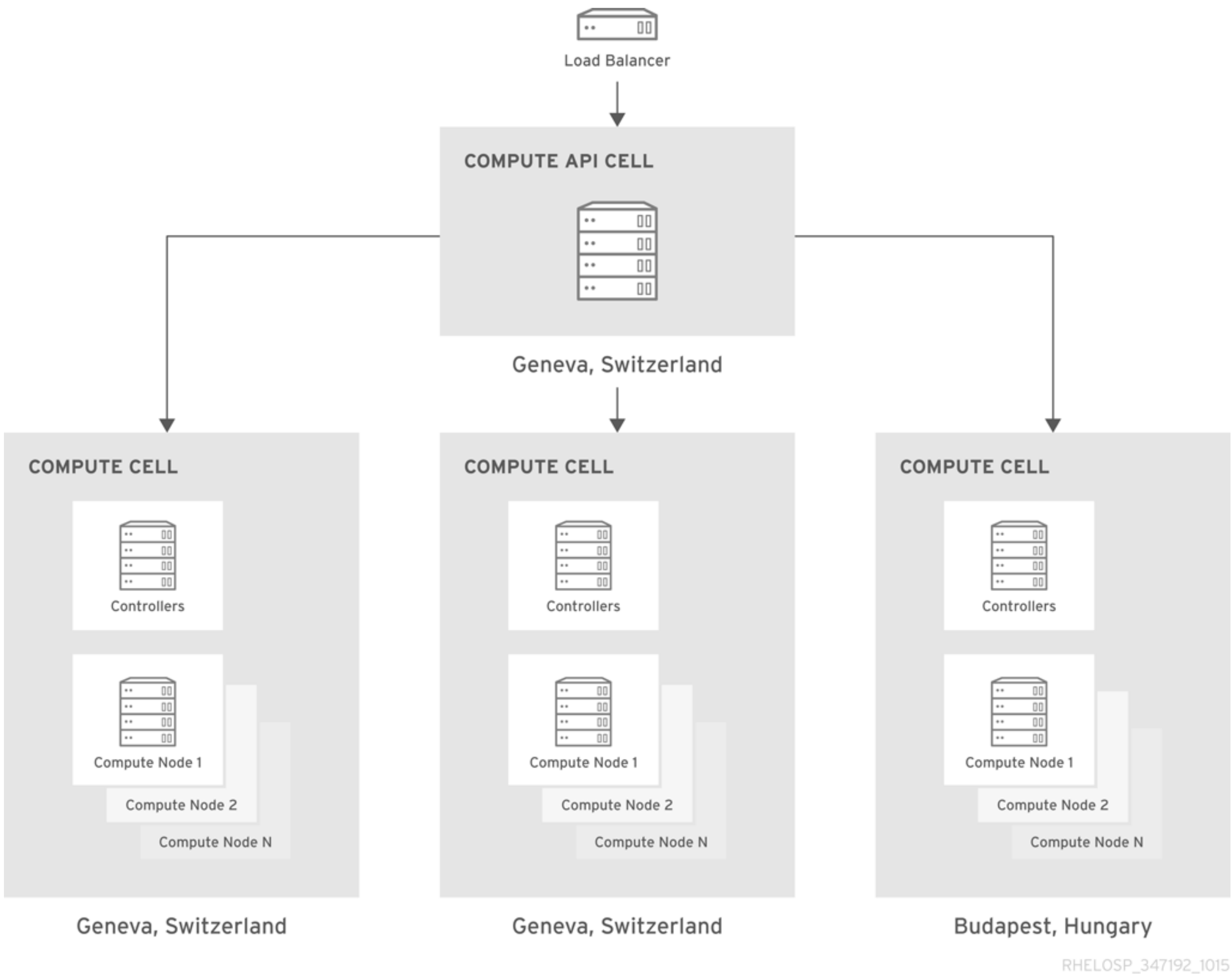
データセンター	キャパシティーの概算
スイス、ジェノバ	3.5 メガワット 91000 コア 120 PB HDD 100 PB テープ 310 TB メモリー
ハンガリー、ブダペスト	2.5 メガワット 20000 コア 6 PB HDD

4.3.2. 設計について

このアーキテクチャーでは、コンピュートリソースの分離と異なるデータセンター間での透過的なスケーリングにセルを使用します。この決定は、セキュリティーグループとライブマイグレーションのサポートに影響を及ぼす上、フレーバーなどの設定要素をセル間で手動で複製する必要がありますが、セルは単一のパブリック API エンドポイントをユーザーに公開すると同時に、必要なスケーリングを提供します。

クラウドは、元のデータセンター 2 つにそれぞれコンピュートセルを 1 つずつ使用し、新規データセンターが追加されると新なコンピュートセルを作成します。各セルには、コンピュートリソースと、高可用性のためにミラーリングされたキューを使用するクラスタリング用に設定された最小 3 つの RabbitMQ メッセージブローカーをさらに分離するためのアベイラビリティーゾーンが 3 つ含まれます。

HAProxy ロードバランサーの背後に常駐する API セルは、スイスのデータセンターにあります。この API セルは、カスタマイズされた種類のセルスケジューラーを使用して、API コールをコンピュートセルに転送します。カスタマイズにより、セルの使用可能なメモリーに応じて、特定のワークロードを特定のデータセンターまたは全データセンターにルーティングすることが可能となります。



フィルターを使用してセル内の配置を処理する Compute のスケジューラーをカスタマイズすることも可能です。たとえば、**ImagePropertiesFilter** は、ゲストが実行するのオペレーティングシステム (例: Linux、Windows など) に応じて、特別な処理を行います。

中央データベースチームは、NetApp ストレージバックエンドを使用して、セルごとに SQL データベースサーバーを active/passive 構成で管理します。バックアップは 6 時間ごとに実行されます。

4.3.3. アーキテクチャーコンポーネント

コンポーネント	説明
Compute	Compute の管理およびスケジューリングサービスは、コントローラー上で実行されます。Compute サービスは、各コンピュートノード上でも実行されます。
Dashboard サービス	OpenStack 管理のための GUI
Identity サービス	基本的な認証と承認

コンポーネント	説明
Image サービス	API セル内で実行され、オーケストレーションツールがアプリケーションを配置することができる少数の Linux イメージセットを維持管理します。
ネットワーク	ネットワークサービス。OpenStack Networking についての詳しい情報は、「*2章 ネットワークに関する詳細*」を参照してください。
モニタリング	Telemetry サービスは、シャード化および複製された MongoDB バックエンドでプロジェクトのクォータを調整するための計測を行います。API の負荷を分散するには、**openstack-nova-api** サービスのインスタンスを子セル内にデプロイして、Telemetry がそれに対してクエリーを実行できるようにします。このデプロイメントには、子セル内で Identity サービスや Image サービスなどの関連サービスの設定も必要です。キャプチャーすべき重要なメトリックには、Image のディスク使用率や Compute API への応答時間などがあります。
Orchestration サービス	新規インスタンスを自動的にデプロイ/テストします。
Telemetry サービス	オーケストレーションの自動スケーリング機能をサポートします。
Object Storage	3 PB の Ceph クラスターでオブジェクトを保管します。

4.3.4. 設計の考慮事項

コンピュート集約型のアーキテクチャーには、「3章*設計*」に記載した基本的な設計の考慮事項に加えて、「コンピュートリソース」に記載のコンピュートノードの設計の考慮事項も検討すべきです。

ワークロード

短期間のワークロードは、継続的インテグレーション/継続的デプロイ (CI-CD) のジョブを含めることができます。これにより、コンピュート集約型のタスクセットを実行するためのコンピュートインスタンスが多数同時に作成されます。次に環境は、インスタンスを削除する前に、結果とアーティファクトを各インスタンスから長期ストレージにコピーします。

Hadoop クラスターや HPC クラスターなどの長期間のワークロードは、通常大量のデータセットを受信し、それらのデータセットに対してコンピューティング作業を実行して、その結果を長期ストレージにプッシュします。コンピューティング作業が終了すると、インスタンスは別のジョブを受信するまでアイドル状態となります。長期間のワークロード用

の環境は、より大きく、複雑な場合が多いですが、ジョブとジョブの間にアクティブな状態を維持するように環境を構築することで、ビルドのコストを相殺することができます。

コンピュートを重視した OpenStack クラウドのワークロードは、ほとんどの場合、永続ブロックストレージを必要としません (ただし、HDFS で Hadoop を使用する一部のクラウドを除く)。共有ファイルシステムまたはオブジェクトストアが初期データセットを維持管理し、コンピューティング結果の保存先としての役割を果たします。入出力 (IO) オーバーヘッドを回避することにより、ワークロードパフォーマンスを大幅に強化することができます。データセットのサイズによっては、オブジェクトストアまたは共有ファイルシステムをスケーリングする必要がある場合があります。

拡張のプランニング

クラウドユーザーは、必要に応じて新規リソースに即時にアクセスできることを期待します。このため、使用状況が標準的な場合と、リソースの需要が急上昇した場合の計画を立てておく必要があります。計画が保守的過ぎると、クラウドが予想外に過剰にサブスクライブされる可能性があります。また、計画がアグレッシブ過ぎると、クラウドが予想外に過小使用の状態となり、不要な運用/保守コストを費すことになる場合があります。

拡張計画の重要な要因は、クラウドの使用状況トレンドの経時的な分析です。平均スピードやクラウドのキャパシティーではなく、サービス提供の一貫性を測定します。このような情報は、キャパシティーパフォーマンスをモデリングして、クラウドの現在および将来のキャパシティーをより正確に判断するのに役立てることができます。

ソフトウェアのモニタリングについての情報は、「追加のソフトウェア」 を参照してください。

CPU とメモリー

現在の標準的なサーバーオファリングは、最大 12 コアの CPU を搭載します。また、一部の Intel CPU はハイパースレッディング・テクノロジー (HTT) をサポートしており、その場合はコアのキャパシティーが倍になります。HTT 対応の複数の CPU をサポートするサーバーの場合は、利用可能なコア数が CPU 数を乗算した値となります。ハイパースレッディングは、インテル専有の同時マルチスレッディングの実装で、インテルの CPU 上での並列化を向上するために使用されます。マルチスレッドアプリケーションのパフォーマンスを向上させるには、HTT を有効化することを検討してください。

CPU での HTT 有効化の決定は、ユースケースによって異なります。たとえば、HTT を無効にすることによって、コンピューティング集約型の環境に役立つ場合があります。HTT を有効にした場合と無効にした場合のローカルワークロードのパフォーマンステストを実行すると、特定の例では、どちらのオプションがより適切かを判断するのに役立てることができます。

キャパシティープランニング

以下のストラテジーを 1 つまたは複数使用して、コンピュート環境にキャパシティーを追加することができます。

- クラウドにキャパシティーを追加して、水平方向にスケーリングします。追加のノードに同じまたは似た CPU を使用して、ライブマイグレーション機能で問題が発生する可能性を軽減します。 ハイパーバイザーホストのスケールアウトは、ネットワークおよびその他のデータセンターリソースにも影響を及ぼします。ラックの容量が上限に達した場合や、追加のネットワークスイッチが必要な場合には、このようなスケーリングを考慮してください。

注記

ノードを適切なアベイラビリティーゾーンおよびホストアグリゲートに配置する追加の作業が必要である点に注意してください。

- 使用量の増加に対応するために、内部のコンピュートホストコンポーネントのキャパシティーを拡大して、垂直方向にスケーリングします。たとえば、コア数のより高い CPU に交換したり、サーバーのメモリーを増設することができます。
- 平均のワークロードを査定し、必要な場合にはメモリーのオーバーコミット比を調節してコンピュート環境で実行可能なインスタンス数を増やします。

重要

コンピュートを重視した OpenStack の設計アーキテクチャーでは、CPU のオーバーコミット比は増やさないでください。CPU のオーバーコミット比を変更すると、CPU リソースを必要とする他のノードとの競合が発生する場合があります。

コンピュートのハードウェア

コンピュートを重視した OpenStack クラウドは、プロセッサーとメモリーリソースに対する需要が極めて高いため、CPU ソケット、CPU コア、メモリーをより多く提供することができる サーバーハードウェアを優先すべきです。

このアーキテクチャーには、ネットワーク接続とストレージ容量はそれほど重要ではありません。ハードウェアは、ユーザーの最小要件を満たすのに十分なネットワーク接続性とストレージキャパシティーを提供する必要がありますが、ストレージとネットワークのコンポーネントは主にデータセットをコンピュート用のクラスターにロードし、一定したパフォーマンスは必要ありません。

ストレージハードウェア

コモディティーハードウェアにオープンソースのソフトウェアを使用して、ストレージアレイをビルドすることができますが、デプロイには、専門知識が必要となります。サーバーで直接アタッチされたストレージを使用するスケールアウトストレージソリューションを使用することもできます。ただし、その場合には、サーバーのハードウェアがストレージソリューションをサポートする必要があります。

ストレージハードウェアの設計時には、以下の要素を考慮してください。

- 可用性。インスタンスが高可用性またはホスト間で移行可能である必要がある場合は、一時的なインスタンスのデータに共有ストレージファイルシステムを使用して、ノードに障害が発生しても、Compute サービスが中断せずに稼働を続けられるようにします。
- レイテンシー。ソリッドステートドライブ (SSD) のディスクを使用してインスタンスのストレージのレイテンシーを最小限に抑え、CPU の遅延を低減し、パフォーマンスを向上します。コンピュートホストに RAID コントローラーカードを使用して下層のディスクサブシステムのパフォーマンスを向上させることを検討してください。
- パフォーマンス。コンピュートを重視したクラウドは、通常ストレージとの間で大量のデータを入出力する必要はありませんが、ストレージのパフォーマンスは考慮すべき重要な要素です。ストレージの I/O 要求のレイテンシーを監視することによって、ストレージハードウェアのパフォーマンスを測定することができます。一部のコンピュート

集約型のワークロードでは、ストレージからのデータ取得中に CPU で遅延が発生するのを最小限に抑えることによって、アプリケーションの全体的なパフォーマンスを大幅に向上させるとができます。

- 拡張可能性。ストレージソリューションの最大容量を決定します。50 PB まで拡張するソリューションは、10 PB までしか拡張できないソリューションよりも拡張性が高いことになります。このメトリックは、スケーラビリティーに関連していますが、異なるメトリックです。スケーラビリティーとは、拡張に伴うソリューションのパフォーマンスの指標です。

- 接続性。接続性がストレージソリューションの要件を満たす必要があります。一元管理されたストレージアレイを選択する場合には、ハイパーバイザーがストレージアレイにどう接続するかを決定します。接続性は、レイテンシーおよびパフォーマンスに影響する場合があるので、ネットワークの特性がレイテンシーを最小限に抑え、環境の全体的なパフォーマンスを高めるようにしてください。

ネットワークのハードウェア

「*2章ネットワークに関する詳細*」に記載した基本的なネットワークの考慮事項に加えて、以下の要素も考慮してください。

- 必要なポート数は、ネットワーク設計に必要な物理スペースに影響を及ぼします。たとえば、1U サーバーでポートあたりのキャパシティーが 10 GbE のポートを 48 提供することができるスイッチは、2U サーバーでポートあたりのキャパシティーが 10 GbE のポートを 24 提供するスイッチよりもはるかにポートの密度が高くなります。ポートの密度を高くすると、コンピューティングやストレージコンポーネント用のラックスペースがより多く残ります。また、障害ドメインと電源の密度についても考慮する必要があります。より高額ですが、機能要件を超えるネットワークを設計すべきではないので、高密度のスイッチを考慮に入れることができます。

- ネットワークアーキテクチャーは、リーフ/スパイン型のように、容量と帯域幅を追加するのに役立つスケーラブルなネットワークモデルを使用して設計すべきです。このタイプのネットワーク設計では、帯域幅を追加したり、スケールアウトしたりして装置のラックを簡単に追加することができます。

- 必要なポート数、ポート速度、ポート密度をサポートするとともに、ワークロードの需要増加に伴う将来の拡張が可能なネットワークハードウェアを選択することが重要です。また、ネットワークアーキテクチャーのどこで冗長性を提供すると価値があるかを評価することも重要です。ネットワークの可用性と冗長性を高くするには高額の費用を伴う場合があるので、冗長ネットワークスイッチとホストレベルでボンディングされたインターフェースによってもたらされる利益と追加費用を比較検討すべきです。

4.4. ストレージを重視したアーキテクチャー

4.4.1. ストレージを重視したアーキテクチャーのタイプ

クラウドストレージモデルでは、物理ストレージデバイス上の論理プールにデータを保管します。このアーキテクチャーは、統合ストレージクラウドと呼ばれる場合もよくあります。

クラウドストレージは、一般的にはホストされたオブジェクトストレージサービスのことを指しますが、この用語には、サービスとして利用可能なその他のタイプのデータストレージも含まれる場合があります。OpenStack は Block Storage (cinder) と Object Storage (swift) の両方を提供しています。クラウドストレージは通常仮想化インフラストラクチャー上で実行され、インターフェースのアクセス可能性、弾力性、スケーラビリティー、マルチテナンシー、従量制課金リソースなどの面で、より広範なクラウドコンピューティングと似ています。

クラウドストレージサービス、オンプレミスまたはオフプレミスで使用することができます。クラウドストレージは冗長性とデータの分散によって耐障害性が高く、またバージョン付きコピーを使用することで高い持続性を提供し、一貫したデータレプリケーションを実行することが可能です。

クラウドストレージアプリケーションには以下のような例があります。

- アクティブなアーカイブ、バックアップ、階層型ストレージ管理
- 一般的なコンテンツストレージと同期 (例: プライベートのドロップボックスサービス)
- 並列ファイルシステムによるデータ分析
- サービス向けの非構造データストア (例: ソーシャルメディアのバックエンドストレージ)
- 永続的なブロックストレージ
- オペレーティングシステムとアプリケーションイメージストア
- メディアのストリーミング
- データベース
- コンテンツの配信
- クラウドストレージピアリング

OpenStack ストレージサービスについての詳しい情報は、「OpenStack Object Storage (swift)」および「OpenStack Block Storage (cinder)」の項を参照してください。

4.4.2. データ分析アーキテクチャー

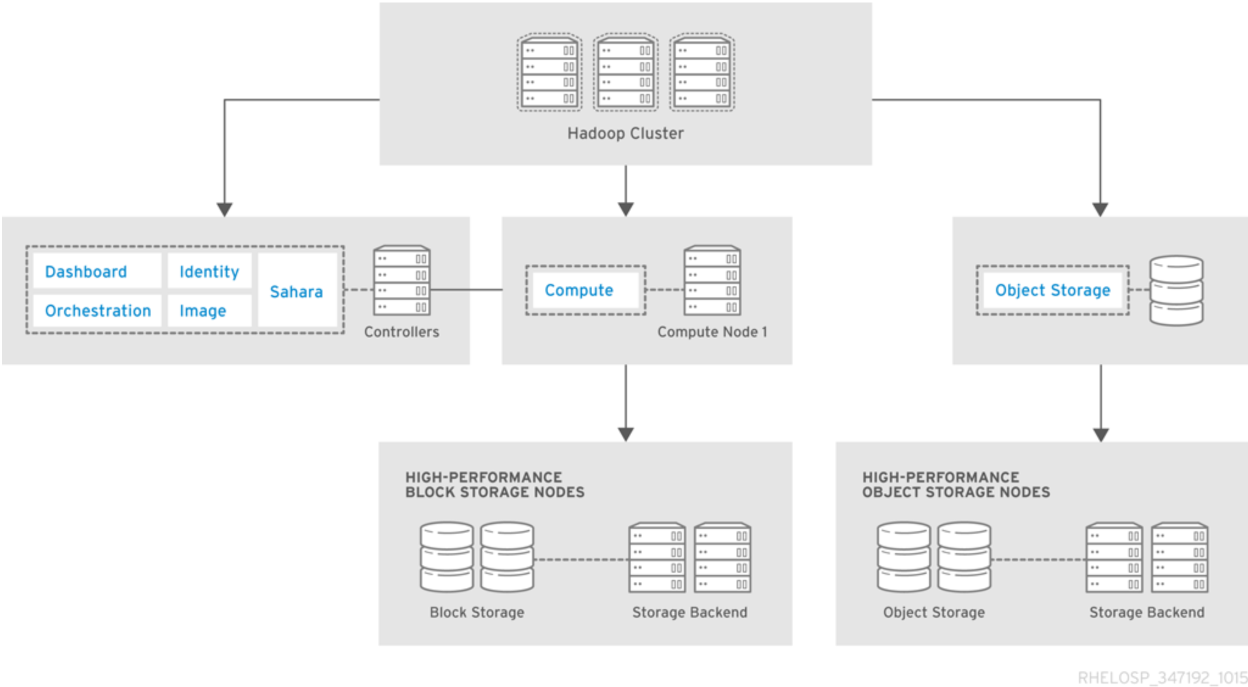
大量のデータセットの分析は、ストレージシステムのパフォーマンスに大きく左右されます。並列ファイルシステムは、ハイパフォーマンスのデータ処理を提供することができるので、パフォーマンスを重視した大規模なシステムに推奨されます。

注記

インストールおよびデプロイメントに関する説明は、「5章 *デプロイメントの情報*」を参照してください。

4.4.2.1. 設計について

OpenStack Data Processing (sahara) は Hadoop と統合してクラウド内の Hadoop クラスターを管理します。以下の図には、ハイパフォーマンス要件の OpenStack ストアを示しています。



ハードウェア要件と設定は、「ハイパフォーマンスデータベースアーキテクチャー」に記載のハイパフォーマンスアーキテクチャーと似ています。この例では、アーキテクチャーは、キャッシュプールに接続して利用可能なプールを加速することができる Ceph の Swift 対応 REST インターフェースを使用します。

4.4.2.2. アーキテクチャーコンポーネント

コンポーネント	説明
Compute	Compute の管理およびスケジューリングサービスは、コントローラー上で実行されます。Compute サービスは、各コンピュートノード上でも実行されます。
Dashboard	OpenStack の管理用 Web コンソール
Identity	基本的な認証と承認
Image	インスタンスの起動とスナップショットの管理に使用するイメージを保管します。このサービスは、コントローラーを実行して少量のイメージセットを提供します。
ネットワーク	ネットワークサービス。OpenStack Networking についての詳しい情報は、「*2章 ネットワークに関する詳細*」を参照してください。
Telemetry	その他の OpenStack サービスのモニタリングとレポーティング。このサービスは、インスタンスの使用状況のモニタリングとプロジェクトクォータの調整に使用します。

コンポーネント	説明
Object Storage	Hadoop バックエンドでデータを保管します。
Block Storage	Hadoop バックエンドでボリュームを保管します。
Orchestration	インスタンスおよびブロックストレージボリューム用のテンプレートを管理します。ストレージを集中的に使用する処理用に追加のインスタンスを起動し、自動的にスケーリングするには、このサービスを Telemetry と共に使用します。

4.4.2.3. クラウドの要件

要件	説明
パフォーマンス	パフォーマンスを強化するには、ディスクアクティビティーをキャッシュする特別なソリューションを選択することができます。
セキュリティー	伝送中のデータと保存されているデータの両方を保護する必要があります。
ストレージの近接性	ハイパフォーマンスまたは大容量のストレージスペースを提供するには、各ハイパーバイザーにストレージをアタッチするか、中央ストレージデバイスからサービスを提供する必要がある可能性があります。

4.4.2.4. 設計の考慮事項

「3章*設計*」に記載した基本的な設計の考慮事項に加えて、「ストレージを重視したアーキテクチャーの考慮事項」に記載の考慮事項にも従うべきです。

4.4.3. ハイパフォーマンスデータベースアーキテクチャー

データベースアーキテクチャーは、ハイパフォーマンスのストレージバックエンドのメリットを享受します。エンタープライズストレージは必須ではありませんが、多くの環境には、OpenStack クラウドでバックエンドとして使用可能なストレージが含まれます。

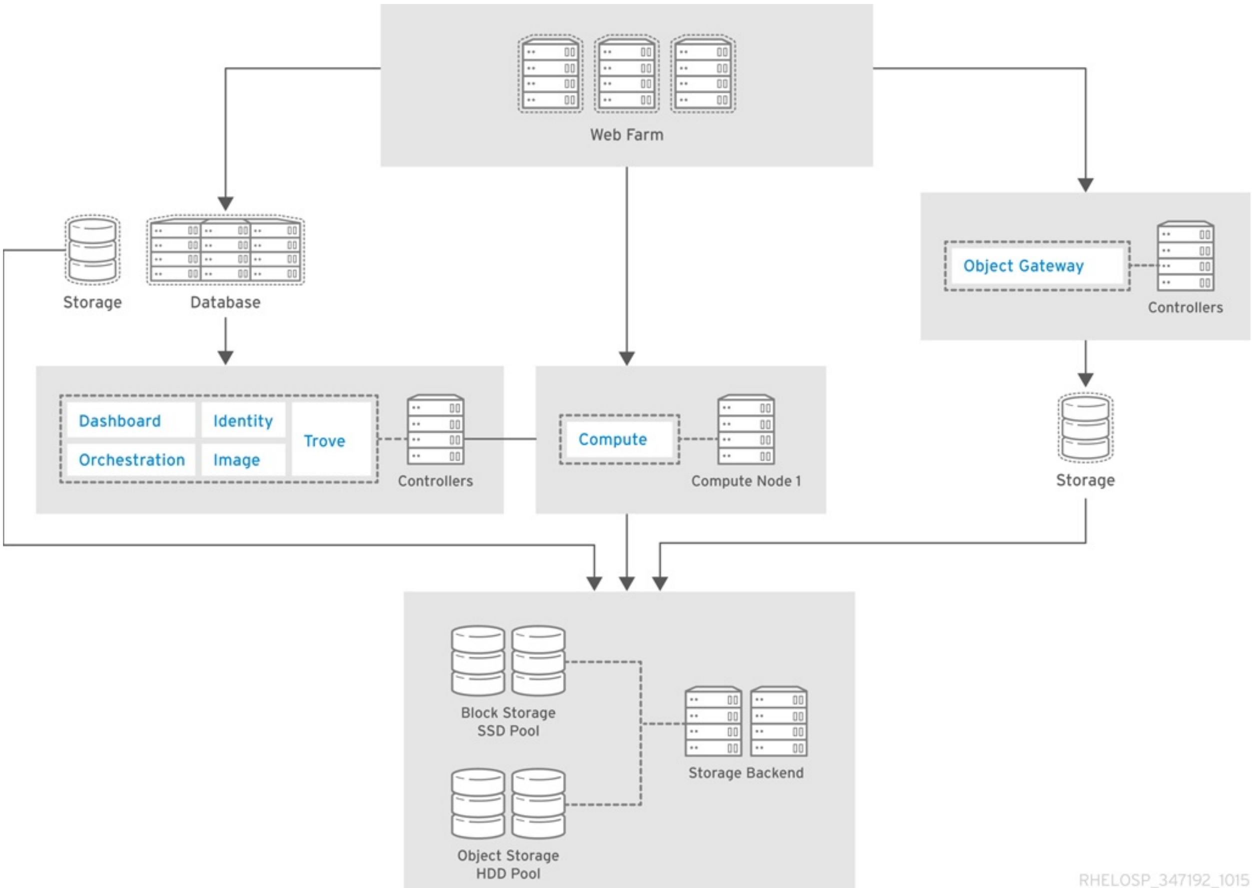
ストレージプールを作成して、OpenStack Block Storage でインスタンスおよびオブジェクトインターフェース用にブロックデバイスを提供することができます。このアーキテクチャーの例では、

データベースの入出力要件が高く、高速な SSD プールからのストレージが要求されます。

4.4.3.1. 設計について

このストレージシステムは、従来のストレージアレイで、SSD のセットによりバッキングされた LUN を使用し、OpenStack Block Storage の統合または Ceph などのストレージプラットフォームを採用します。

このシステムは、追加のパフォーマンス機能を提供することが可能です。データベースの例では、以下のデータベース例では、SSD プールの一部がデータベースサーバーに対するブロックデバイスとして機能します。ハイパフォーマンスの分析例では、インライン SSD キャッシュ層が REST インターフェースを加速化します。



この例では、Ceph が Swift 対応の REST インターフェースを提供するとともに、分散ストレージクラスターからのブロックレベルストレージも提供します。これは、柔軟性が非常に高い上、自己復旧や自動バランシングなどの機能により、運用コストを削減することができます。イレイジャーコーディング対応プールは、使用可能な容量を最大化するのに推奨されます。

注記

イレイジャーコーディング対応プールには、より高いコンピューティング要件やオブジェクトに許可される操作の制限などを特別に考慮する必要があります。イレイジャーコーディング対応プールは、部分的書き込みはサポートしません。

4.4.3.2. アーキテクチャーコンポーネント

コンポーネント	説明
Compute	Compute の管理およびスケジューリングサービスは、コントローラー上で実行されます。Compute サービスは、各コンピュートノード上でも実行されます。
Dashboard	OpenStack の管理用 Web コンソール
Identity	基本的な認証と承認
Image	インスタンスの起動とスナップショットの管理に使用するイメージを保管します。このサービスは、コントローラーを実行して少量のイメージセットを提供します。
ネットワーク	ネットワークサービス。OpenStack Networking についての詳しい情報は、「*2章 ネットワークに関する詳細*」を参照してください。
Telemetry	その他の OpenStack サービスのモニタリングとレポーティング。このサービスは、インスタンスの使用状況のモニタリングとプロジェクトクォータの調整に使用します。
モニタリング	Telemetry サービスプロジェクトクォータを調整する目的の計測を実行します。
Object Storage	Ceph バックエンドでデータを保管します。
Block Storage	Ceph バックエンドでボリュームを保管します。
Orchestration	インスタンスおよびブロックストレージボリューム用のテンプレートを管理します。ストレージを集中的に使用する処理用に追加のインスタンスを起動し、自動的にスケーリングするには、このサービスを Telemetry と共に使用します。

4.4.3.3. ハードウェア要件

SSD キャッシュ層を使用して、ブロックデバイスを直接ハイパーバイザーやインスタンスにリンクすることができます。REST インターフェースもインラインキャッシュとして SSD キャッシュシステムを使用することが可能です。

コンポーネント	要件	ネットワーク
10 GbE の水平スケーリングが可能なリーフ/スパイン型バックエンドストレージおよびフロントエンドネットワーク	ストレージハードウェア	* 24x1 TB SSD を搭載したストレージサーバー (キャッシュ層用) 5 台 * 1 台あたり 12x4 TB のディスクを搭載したストレージサーバー 10 台。合計の容量は 480 TB に相当。レプリカを 3 回作成後の使用可能な空き容量は約 160 TB。

4.4.3.4. 設計の考慮事項

「3章*設計*」に記載した基本的な設計の考慮事項に加えて、「ストレージを重視したアーキテクチャーの考慮事項」に記載の考慮事項にも従うべきです。

4.4.4. ストレージを重視したアーキテクチャーの考慮事項

ストレージ集約型のアーキテクチャーには、基本的な設計の考慮事項 (「3章*設計*」) とストレージノードの設計 (「ストレージのリソース」) に加えて、以下の項目を検討すべきです。

接続性

接続性がストレージソリューションの要件に対応していることを確認します。一元管理されたストレージアレイを選択する場合には、ハイパーバイザーがアレイにどう接続するかを決定します。接続性は、レイテンシーおよびパフォーマンスに影響する場合があるので、ネットワークの特性がレイテンシーを最小限に抑え、設計の全体的なパフォーマンスを高めるようにします。

密度

- インスタンスの密度。ストレージを重視したアーキテクチャーでは、インスタンスの密度と、CPU/メモリーのオーバーサブスクリプション比は低くなります。設計にデュアルソケットハードウェアを使用している場合には特に、予想されるスケールをサポートするホストがより多く必要になります。

- ホストの密度。クワッドソケットプラットフォームを使用することにより、ホスト数が多い構成に対応できます。このプラットフォームでは、ホストの密度を低くなり、ラック数が増えます。この構成は、電源接続数に加えて、ネットワークや冷却の要件にも影響を及ぼします。

- 電源と冷却。2U、3U、4U サーバーの電源および冷却の密度の要件は、ブレード、スレッド、1U サーバー設計よりも低い可能性があります。この構成は、より古いインフラストラクチャーを使用するデータセンターに推奨されます。

柔軟性

組織は、オフプレミスとオンプレミスのクラウドストレージオプションのいずれかを選択する柔軟性を必要とします。たとえば、運用の継続性、障害復旧、セキュリティー、記録の保管に関する法律、規制、ポリシーなどは、ストレージプロバイダーの費用対効果に影響を及ぼす場合があります。

レイテンシー

ソリッドステートドライブ (SSD) は、インスタンスのストレージのレイテンシーを最小限に抑え、ストレージのレイテンシーによって生じる場合のある CPU の遅延を低減することができます。下層のディスクシステムのパフォーマンスを向上させるためにコンピュートホストで RAID コントローラーカードを使用することによるメリットを評価します。

監視と警告

監視と警告のサービスは、ストレージリソースに対する需要の高いクラウド環境では極めて重要です。これらのサービスは、ストレージシステムの正常性とパフォーマンスのリアルタイムのビューを提供します。統合管理コンソール、または SNMP データを視覚化するその他のダッシュボードは、ストレージクラスター内で発生した問題の発見と解決に役立ちます。

ストレージを重視したクラウド設計には以下の要素が含まれるべきです。

- 物理ハードウェアと環境リソース 監視 (例: 温度、湿度)

- 使用可能なストレージ、メモリー、CPU などのストレージリソースの監視

- ストレージシステムが想定通りのパフォーマンスを達成していることを確認するための詳細なストレージパフォーマンスデータの監視

- ストレージへのアクセスに影響するサービス停止をチェックするためのネットワークリソースの監視

- 一元化されたログ収集とログ分析の機能

- 問題追跡のためのチケットシステムまたはチケットシステムとの統合

- 担当チームへの警告と通知、またはストレージに伴う問題が発生した際に解決することができる自動システム

- スタッフを配置し、問題解決に常時対応可能なネットワーク運用センター (NOC)

スケーリング

ストレージを重視した OpenStack アーキテクチャーは、スケールアウトではなく、スケールアップにフォーカスすべきです。コスト、電源、冷却、物理ラック、床面積、サポート/保証、管理容易性などの要素に基づいて、少数の大型ホストの構成にするか、多数の小型ホストの構成にするかを決定する必要があります。

4.5. ネットワークを重視したアーキテクチャー

「ネットワークを重視したアーキテクチャーのタイプ」

「クラウドのストレージとバックアップのアーキテクチャー」

「大規模な Web アプリケーション向けのアーキテクチャー」

「ネットワークを重視したアーキテクチャーの考慮事項」

4.5.1. ネットワークを重視したアーキテクチャーのタイプ

OpenStack デプロイメントすべて、サービスをベースとしているので、適切に機能するには、ネットワーク通信に依存します。ただし、場合によっては、ネットワーク設定はよりクリティカルで設計における追加の考慮事項が必要です。

以下の表では、ネットワークを重視した一般的なアーキテクチャーについての説明をまとめています。このようなアーキテクチャーは、ユーザーとアプリケーションの要件を満たす、信頼できるネットワークインフラストラクチャーとサービスに依存します。

アーキテクチャー	説明
ビッグデータ	ビッグデータの管理/収集に使用されるクラウドは、ネットワークリソースに対して多大な需要をもたらします。ビッグデータは、多くの場合、データの部分的なレプリカを使用して、大型の分散型クラウドの整合性を維持します。大量のネットワークリソースを必要とするビッグデータアプリケーションには Hadoop、Cassandra、NuoDB、Riak、その他の SQL 以外の分散データベースがあります。
コンテンツ配信ネットワーク (CDN)	CDN は、多数のエンドユーザーによるビデオのストリーミングや、写真の閲覧、Web コンファレンスのホスティング、分散されたクラウドベースのデータリポジトリーへのアクセスに使用することができます。ネットワーク設定は、レイテンシー、帯域幅、インスタンスの分散に影響を及ぼします。コンテンツの配信とパフォーマンスに影響するその他の要素には、バックエンドシステムのネットワークスループット、リソースの場所、WAN アーキテクチャー、キャッシュの方法などがあります。
高可用性 (HA)	高可用性の環境は、サイト間のデータレプリケーションを維持するネットワークのサイズに左右されます。1 つのサイトが利用不可になった場合に、そのサイトのサービスが復旧するまで、他のサイトが増えた分の負荷に対応することができます。追加の負荷を処理できるネットワーク容量にサイズを設定することが重要です。
ハイパフォーマンスコンピューティング (HPC)	HPC 環境には、クラウドクラスターのニーズに対応するためのトラフィックフローと使用パターンをさらに考慮する必要があります。HPC は、ネットワーク内の分散コンピューティングのための水平方向 (east-west) のトラフィックパターンが高いですが、アプリケーションによっては、ネットワークに出入りする垂直方向 (north-south) のトラフィックも相当な量となる場合があります。

アーキテクチャー	説明
高スピードまたは高容量のトランザクションシステム	このようなタイプのアプリケーションは、ネットワークジッターとレイテンシーの影響を受けます。環境の例としては、財務システム、クレジットカード取引用アプリケーション、商取引用のシステムなどがあります。これらのアーキテクチャーは、高ボリュームの水平方向と垂直方向のトラフィックのバランスを取って、データ配信の効率性を最大限に高める必要があります。このようなシステムの多くは、大型でハイパフォーマンスのデータベースバックエンドにアクセスしなければなりません。
ネットワーク管理機能	DNS、NTP、SNMP などのバックエンドネットワークサービスの配信をサポートする環境。これらのサービスは、内部ネットワークの管理に使用することができます。
ネットワークサービスオファリング	サービスをサポートするための顧客向けネットワークツールを実行する環境。VPN、MPLS プライベートネットワーク、GRE トンネルなどがその例です。
仮想デスクトップインフラストラクチャー (VDI)	VDI は、ネットワークの輻輳、レイテンシー、ジッターの影響を受けます。VDI にはアップストリームとダウンストリームの両方のトラフィックが必要で、キャッシュに依存してアプリケーションをエンドユーザーに提供することはできません。
Voice over IP (VoIP)	VoIP システムは、ネットワークの輻輳、レイテンシー、ジッターの影響を受けます。VoIP システムには、対称的なトラフィックパターンがあり、最適なパフォーマンスを提供するにはネットワーク QoS が必要です。また、アクティブなキュー管理を実装して、音声およびマルチメディアのコンテンツを配信することができます。ユーザーは、レイテンシーとジッターの変動の影響を受け、非常に低いレベルでそれらを検知することができます。
ビデオまたは Web コンファレンス	コンファレンスシステムは、ネットワークの輻輳、レイテンシー、ジッターの影響を受けます。ビデオコンファレンスシステムには、対称的なトラフィックパターンがありますが、ネットワークが MPLS プライベートネットワークでホストされていない場合には、システムはネットワーク QoS を使用してパフォーマンスを向上させることはできません。VoIP と同様に、このシステムのユーザーは低いレベルでもネットワークパフォーマンス問題を検知します。
Web ポータル/サービス	Web サーバーは、クラウドサービスにおける共通のアプリケーションなので、そのネットワーク要件を理解する必要があります。ネットワークは、ユーザーの需要を満たし、最小の待ち時間で Web ページを配信するためのスケールアウトが必要です。アーキテクチャーを計画する際には、ポータルのアーキテクチャー詳細に応じて、内部の水平/垂直方向のネットワーク帯域幅を検討すべきです。

4.5.2. クラウドのストレージとバックアップのアーキテクチャー

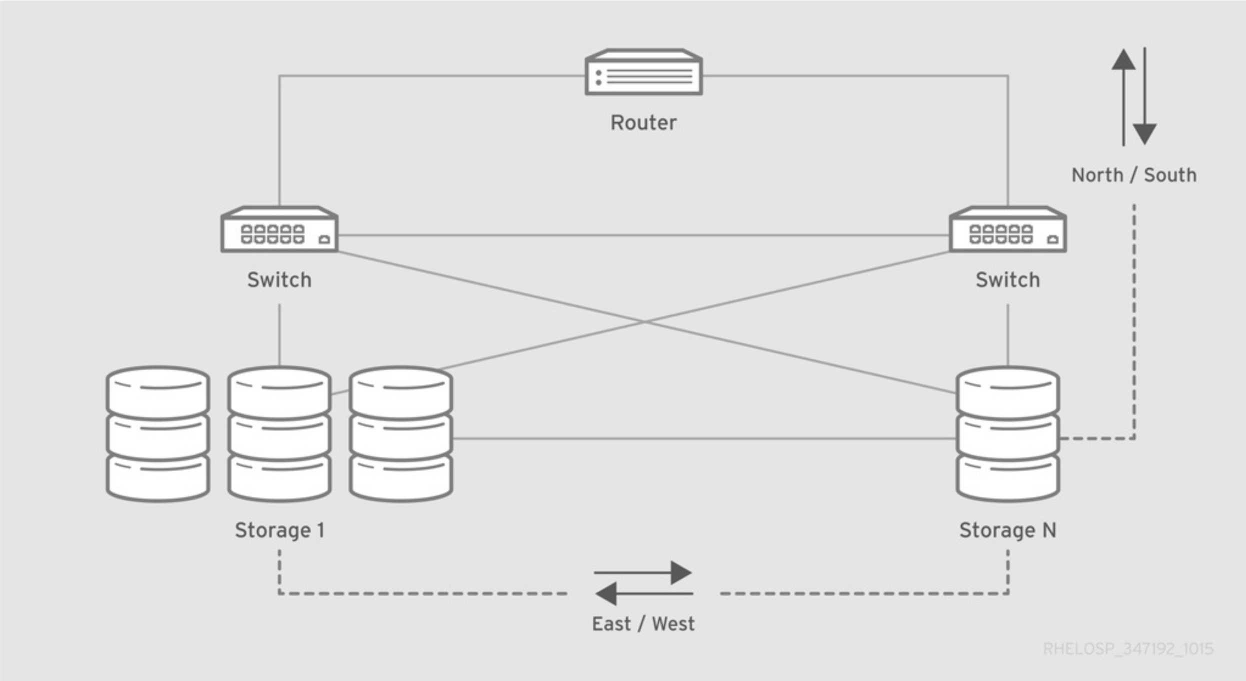
このアーキテクチャーは、ファイルストレージおよびファイル共有を提供するクラウドが対象です。これはストレージを重視したユースケースと見なされる場合がありますが、ネットワーク側の要件によりネットワークを重視したユースケースとなります。

注記

インストールおよびデプロイメントに関する説明は、「5章 *デプロイメントの情報*」を参照してください。

4.5.2.1. 設計について

以下のクラウドバックアップアプリケーションのワークロードには、ネットワークに影響を及ぼす 2 つの特定の動作があります。



このワークロードには、外部向けのサービスと内部でレプリケーションを行うアプリケーションが含まれ、どちらも垂直/水平方向のトラフィックを考慮する必要があります。

垂直方向のトラフィック

垂直方向のトラフィックは、クラウドに出入りするデータで構成されます。ユーザーがコンテンツをアップロードして保管すると、そのコンテンツは OpenStack 環境の中に入ります (下向き)。ユーザーがコンテンツをダウンロードすると、そのコンテンツは OpenStack 環境の外に移動します (上向き)。

このサービスは、主にバックアップサービスとして稼働するので、トラフィックの大半は環境の内部に移動します。このような状況では、OpenStack 環境の入ってくるトラフィックが環境から出て行くトラフィックよりも大きくなるため、ネットワークを非対称的なダウンストリームに設定すべきです。

水平方向のトラフィック

水平方向のトラフィックは、環境内を移動するデータで構成されます。レプリケーションは任意のノードで開始し、アルゴリズムによって複数の他のノードをターゲットとする場合があるので、このようなトラフィックは、完全に対称的となる傾向があります。ただし、このトラフィックは、垂直方向のトラフィックに干渉する場合があります。

4.5.2.2. アーキテクチャーコンポーネント

コンポーネント	説明
Compute	Compute の管理およびスケジューリングサービスは、コントローラー上で実行されます。Compute サービスは、各コンピュートノード上でも実行されます。
Dashboard	OpenStack の管理用 Web コンソール
Identity	基本的な認証と承認
Image	インスタンスの起動とスナップショットの管理に使用するイメージを保管します。このサービスは、コントローラーを実行して少量のイメージセットを提供します。
ネットワーク	ネットワークサービス。OpenStack Networking についての詳しい情報は、「*2章 ネットワークに関する詳細*」を参照してください。
Object Storage	バックアップコンテンツの保管
Telemetry	その他の OpenStack サービスのモニタリングとレポーティング

4.5.2.3. 設計の考慮事項

「*3章 設計*」に記載した基本的な設計の考慮事項に加えて、「ネットワークを重視したアーキテクチャーの考慮事項」に記載の考慮事項にも従うべきです。

4.5.3. 大規模な Web アプリケーション向けのアーキテクチャー

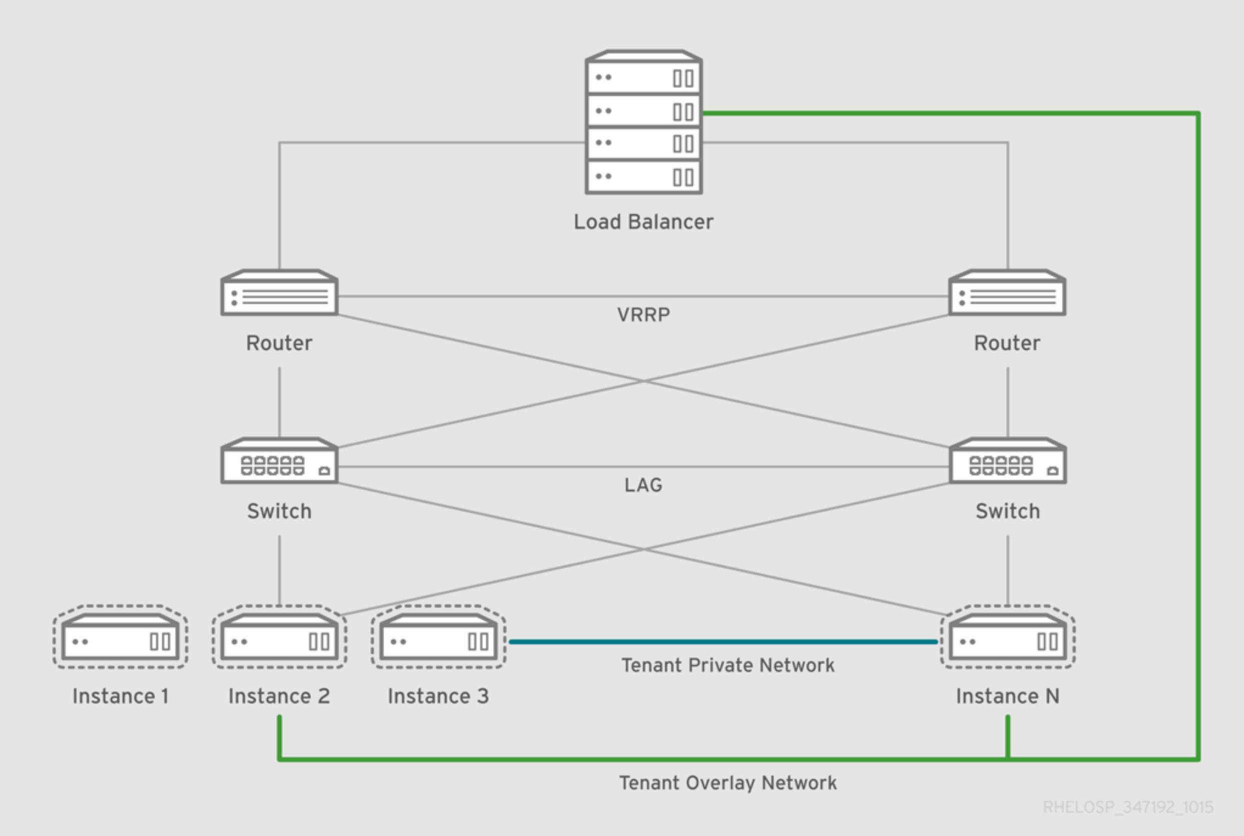
このアーキテクチャーは、需要の急激な増加に対応するように水平方向にスケーリングして、インスタンス数を高くする大規模な Web アプリケーション向けです。このアプリケーションは、データを保護する SSL 接続が必要で、個別のサーバーへの接続を失うことはできません。

注記

インストールおよびデプロイメントに関する説明は、「*5章 デプロイメントの情報*」を参照してください。

4.5.3.1. 設計について

以下の図は、このワークロード向けの設計例を示しています。



この設計には、以下のコンポーネントとワークフローが含まれます。

- ハードウェアロードバランサーは、SSL オフロード機能を提供し、アドレスの使用を削減するためにテナントネットワークに接続します。
- ロードバランサーは、アプリケーションの仮想 IP (VIP) にサービスを提供する際にルーティングアーキテクチャーにリンクします。
- ルーターとロードバランサーは、アプリケーションのテナントネットワークの GRE トンネル ID と、テナントサブネット内でアドレスプール外の IP アドレスを使用しますが、この設定により、パブリックの IP アドレスを使用せずにロードバランサーがアプリケーションの HTTP サーバーと通信することができます。

4.5.3.2. アーキテクチャーコンポーネント

Web サービスアーキテクチャーは、数多くのオプションとオプションコンポーネントで構成される場合があります。そのため、このアーキテクチャーは、複数の OpenStack 設計で使用することができます。ただし、大半の Web スケールワークロードを処理するには、いくつかの主要コンポーネントをデプロイする必要があります。

コンポーネント	説明
Compute	Compute の管理およびスケジューリングサービスは、コントローラー上で実行されます。Compute サービスは、各コンピュートノード上でも実行されます。

コンポーネント	説明
Dashboard	OpenStack の管理用 Web コンソール
Identity	基本的な認証と承認
Image	インスタンスの起動とスナップショットの管理に使用するイメージを保管します。このサービスは、コントローラーを実行して少量のイメージセットを提供します。
ネットワーク	ネットワークサービス。分離したネットワークの構成は、プライベートテナントネットワーク上にあるデータベースと互換性があります。そのようなデータベースは、大量のブロードキャストトラフィックを生成せず、コンテンツ用のデータベースと相互接続する必要がある場合があるためです。
Orchestration	スケールアウト時およびトラフィックバースト中に使用するインスタンスのテンプレートを管理します。
Telemetry	その他の OpenStack サービス用のモニタリングとレポーティング。このサービスは、インスタンスの使用状況のモニタリングと Orchestration サービスからのインスタンステンプレートの呼び出しに使用します。
Object Storage	バックアップコンテンツの保管

4.5.3.3. 設計の考慮事項

「3章*設計*」に記載した基本的な設計の考慮事項に加えて、「ネットワークを重視したアーキテクチャーの考慮事項」に記載の考慮事項にも従うべきです。

4.5.4. ネットワークを重視したアーキテクチャーの考慮事項

ネットワーク集約型のアーキテクチャーには、基本的な設計の考慮事項 (「3章*設計*」) とネットワークノードの設計 (2章*ネットワークに関する詳細*) に加えて、以下の点を検討すべきです。

外部の依存関係

以下のような外部ネットワークコンポーネントの使用を検討してください。

- ワークロードの分散または特定の機能の負荷を軽減するハードウェアロードバランサー
- 動的ルーティングを実装するための永続デバイス

OpenStack Networking は、トンネリング機能を提供しますが、ネットワーク管理された

リージョンのみに制限されています。OpenStack リージョンを超えて、他のリージョンまたは外部システムにトンネルを拡張するには、OpenStack の外部にトンネルを実装するか、トンネル管理システムを使用して、外部トンネルへのトンネルまたはオーバーレイをマッピングします。

最大転送単位 (MTU)

一部のワークロードは、大型のデータブロックを転送するために大きな MTU が必要です。ビデオストリーミングやストレージのレプリケーションなどのアプリケーション用のネットワークサービスを提供する場合には、OpenStack ハードウェアノードと、可能な場合にはジャンボフレーム用の補助ネットワーク機器を設定します。この構成は、利用可能な帯域幅の使用率を最大化します。

パケットが通過する全パスにわたってジャンボフレームを設定します。1 つのネットワークコンポーネントがジャンボフレームに対応できない場合には、全パスがデフォルトの MTU に戻ります。

NAT の使用

固定のパブリック IP ではなく、Floating IP が必要な場合は、NAT を使用する必要があります。たとえば、DHCP サーバーの IP にマッピングされている DHCP リレーを使用します。この場合には、各新規インスタンスにレガシーや外部のシステムを再設定する代わりに、インフラストラクチャーがターゲットの IP アドレスを新規インスタンスに適用するように自動化した方が簡単です。

OpenStack Networking によって管理される Floating IP 用の NAT は、ハイパーバイザー内に常駐しますが、その他のバージョンの NAT は他の場所で実行される場合があります。IPv4 アドレスが不足している場合には、以下の方法を用いて OpenStack 外の IPv4 アドレス不足を軽減することができます。

- ロードバランサーを OpenStack 内でインスタンスとして実行するか、外部でサービスとして実行します。OpenStack の Load-Balancer-as-a-Service (LBaaS) は、ロードバランシングソフトウェア (例: HAproxy) を内部で管理することができます。このサービスが仮想 IP (VIP) アドレスを管理する一方、HAproxy インスタンスからのデュアルホームコネクションが全コンテンツサーバーをホストするテナントプライベートネットワークにパブリックネットワークを接続します。

- ロードバランサーを使用して仮想 IP にサービスを提供するとともに、外部の方法やプライベートアドレスを使用してテナントオーバーレイネットワークに接続します。

場合によっては、インスタンスで IPv6 アドレスのみを使用し、NAT ベースの移行テクノロジー (例: NAT64、DNS64) を提供するインスタンスまたは外部サービスを稼働することもできます。この設定は、グローバルでルーティング可能な IPv6 アドレスを提供し、IPv4 アドレスは必要のある場合にしか使用しません。

サービス品質 (QoS)

QoS は、ネットワークパフォーマンスの低下により優先順位が高くなったパケットに対して即時にサービスを提供するので、ネットワーク集約型のワークロードに多大な影響を及ぼします。Voice over IP (VoIP) のようなアプリケーションでは、継続的な運用には、通常、差別化されたサービスコードポイントが必要です。

複数の混在するワークロードで QoS を使用して、優先順位が低く帯域幅の大きいアプリケーション (例: バックアップサービス、ビデオコンファレンス、ファイル共有など) が他のワークロードの継続的な運用に必要な帯域幅をブロックしてしまわないようにすることもできます。

ファイルとストレージ間のトラフィックに低いクラスのトラフィック (例: ベストエフォート、スカベンジャーなど) のタグを付けて、優先度の高いトラフィックがネットワークを通過できるようにすることが可能です。クラウド内のリージョンが地理的に分散されている場合、WAN 最適化を使用してレイテンシーやパケット損失を軽減することができます。

ワークロード

ルーティングとスイッチングのアーキテクチャーは、ネットワークレベルの冗長性を必要とするワークロードに対応すべきです。この構成は、選択したネットワークハードウェア、選択したハードウェアのパフォーマンス、ネットワークモデルによって異なります。例としては、リンクアグリゲーション (LAG)、ホットスタンバイルータープロトコル (HSRP) があります。

ワークロードは、オーバーレイネットワークの有効性に影響します。アプリケーションネットワークの接続が少量、短期間、またはバースト性の場合には、動的なオーバーレイを実行すると、ネットワークが伝送するパケットの分だけの帯域幅を生成することができます。オーバーレイは、ハイパーバイザーで問題が発生する原因となるのに十分な長さのレイテンシーを起し、パケット毎秒や接続数毎秒などでパフォーマンス低下をもたらす場合もあります。

デフォルトでは、オーバーレイには第 2 のフルメッシュオプションがあり、ワークロードによって異なります。たとえば、大半の Web サービスアプリケーションは、フルメッシュのオーバーレイネットワークで大きな問題はありませんが、一部のネットワーク監視ツールやストレージレプリケーションワークロードでは、スループットでパフォーマンスの問題が発生するか、ブロードキャストトラフィックが過剰となります。

第5章 デプロイメントの情報

以下の表には、本ガイドに記載したコンポーネントの参考情報をまとめています。

Red Hat OpenStack の他のガイドについては Red Hat OpenStack Platform の製品ドキュメント で参照してください。

コンポーネント	参考情報
Red Hat Enterprise Linux	Red Hat OpenStack Platform 10 は Red Hat Enterprise Linux 7.3 でサポートされています。Red Hat Enterprise Linux のインストールに関する情報は、Red Hat Enterprise Linux のドキュメント から対応するインストールガイドを参照してください。
OpenStack	OpenStack コンポーネントおよびそれらの依存関係のインストールについては、Red Hat OpenStack Platform director を使用してください。director では、基本的な OpenStack アンダークラウドを使用して、最終的なオーバークラウドの OpenStack ノードのプロビジョニングと管理を行います。 デプロイしたオーバークラウドに必要な環境に加えて、アンダークラウドのインストールには、追加のホストマシンが必要な点に注意してください。詳しい手順は、『Red Hat OpenStack Platform director のインストールと使用方法』を参照してください。
高可用性	追加の高可用性コンポーネント (例: HAProxy) の設定については、「Deploying Highly Available Red Hat OpenStack Platform 6 with Ceph Storage」の記事を参照してください。 ライブマイグレーションの設定については、『インスタンス＆イメージガイド』を参照してください。
LBasS	Load Balancing-as-a-Service を使用するには、『ネットワークガイド』のの「Load Balancing-as-a-Service (LBaaS) の設定」のセクションを参照してください。
Pacemaker	Pacemaker は Red Hat Enterprise Linux にアドオンとして統合されています。高可用性用の Red Hat Enterprise Linux を設定するには、『High Availability アドオンの概要』を参照してください。