



Red Hat OpenStack Platform 13

ストレージガイド

OpenStack での永続ストレージの理解、使用、管理

Red Hat OpenStack Platform 13 ストレージガイド

OpenStack での永続ストレージの理解、使用、管理

OpenStack Team
rhos-docs@redhat.com

法律上の通知

Copyright © 2018 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

概要

本ガイドでは、Red Hat OpenStack Platform 環境における永続ストレージの使用/管理手順について詳しく説明します。また、各永続ストレージの種別に対応する OpenStack サービスの設定および管理の手順も記載しています。

目次

前書き	4
第1章 OPENSTACK での永続ストレージの概要	5
1.1. スケーラビリティおよびバックエンド	6
1.2. アクセシビリティと管理	6
1.3. セキュリティ	6
1.4. 冗長性および災害復旧	7
第2章 BLOCK STORAGE とボリューム	8
2.1. バックエンド	8
2.2. BLOCK STORAGE サービスの管理	8
2.2.1. ボリューム種別へのボリューム設定の関連付け	8
2.2.1.1. ホストドライバーの機能一覧	9
2.2.1.2. ボリューム種別の作成と設定	10
2.2.1.3. ボリューム種別の編集	10
2.2.1.4. ボリューム種別の削除	11
2.2.1.5. プライベートのボリューム種別の作成と設定	11
2.2.2. Block Storage サービスの内部テナントの作成および設定	12
2.2.3. Image-Volume キャッシュの設定および有効化	13
2.2.4. QoS スペックの使用	14
2.2.4.1. QoS スペックの作成と設定	14
2.2.4.2. 容量ベースの QoS 上限の設定	15
2.2.4.3. QoS スペックとボリューム種別の関連付け	16
2.2.4.4. ボリューム種別からの QoS スペックの関連付け解除	16
2.2.5. ボリュームの暗号化設定	17
2.2.5.1. ボリューム種別の暗号化設定	17
2.2.6. ボリュームを複数のバックエンドに割り当てる方法の設定	17
2.2.7. 整合性グループの設定と使用	19
2.2.7.1. 整合性グループの設定	20
2.2.7.2. 整合性グループの作成および管理	21
2.2.7.3. 整合性グループのスナップショットの作成および管理	22
2.2.7.4. 整合性グループのクローン作成	22
2.2.8. バックアップの管理	23
2.2.8.1. テナントのバックアップクォータの表示と変更	23
2.2.8.2. Dashboard を使用したボリュームバックアップ管理の有効化	23
2.2.8.3. バックアップリポジトリとしての NFS 共有の設定	24
2.2.8.3.1. 異なるバックアップファイルサイズの設定	25
2.3. ボリュームの基本的な使用方法と設定	25
2.3.1. ボリュームの作成	25
2.3.2. ボリュームを作成するバックエンドの指定	27
2.3.3. ボリュームの名前と説明の編集	27
2.3.4. ボリュームの削除	27
2.3.5. インスタンスへのボリュームの接続と切断	28
2.3.5.1. インスタンスへのボリュームの接続	28
2.3.5.2. インスタンスからのボリュームの切断	28
2.3.6. ボリュームの読み取り専用設定	28
2.3.7. ボリュームの所有者の変更	28
2.3.7.1. コマンドラインを使用したボリュームの譲渡	28
2.3.7.2. ダッシュボードを使用したボリュームの譲渡	29
2.3.8. ボリュームスナップショットの作成、使用、削除	30
2.3.8.1. Red Hat Ceph バックエンドにおけるスナップショットの保護と保護解除	31
2.3.9. Image サービスにボリュームをアップロードする手順	31

2.3.10. ボリューム種別の変更	31
2.4. ボリュームの高度な設定	32
2.4.1. ボリュームのバックアップと復元	32
2.4.1.1. ボリュームの完全バックアップの作成	33
2.4.1.1.1. admin としてのボリュームのバックアップ作成	34
2.4.1.2. ボリュームの増分バックアップの作成	34
2.4.1.3. Block Storage データベースでデータ損失が発生した後の復元	35
2.4.1.4. バックアップからのボリュームの復元	36
2.4.2. ボリュームの移行	36
2.4.2.1. ホスト間の移行	36
2.4.2.2. バックエンド間での移行	37
第3章 OBJECT STORAGE	38
3.1. OBJECT STORAGE サービスの管理	38
3.1.1. Erasure Coding の設定	38
3.1.1.1. Erasure Coding ポリシーの設定	38
3.1.1.2. オブジェクトストレージリングの設定	40
3.1.1.3. Erasure Coding の使用	41
3.1.2. Fast-POST の設定	41
3.1.3. Image サービスのバックエンドとしてのオブジェクトストレージの設定	42
3.1.4. 保存データの暗号化の有効化	43
3.1.5. スタンドアロンの Object Storage クラスターのデプロイ	43
3.1.5.1. roles_data.yaml ファイルの作成	43
3.1.5.2. 新規ロールのデプロイ	46
3.2. 基本的なコンテナ管理	46
3.2.1. コンテナの作成	46
3.2.2. コンテナ用の擬似フォルダーの作成	47
3.2.3. コンテナの削除	47
3.2.4. オブジェクトのアップロード	47
3.2.5. オブジェクトのコピー	47
3.2.6. オブジェクトの削除	48
第4章 ファイル共有	49
4.1. バックエンド	49
4.2. 共有種別の作成と管理	49
4.3. ファイル共有の作成	50
4.4. ファイル共有とエクスポートの情報の一覧表示	51
4.5. ファイル共有に対するアクセス権の付与	51
4.6. ファイル共有へのアクセスの取り消し	52
4.7. インスタンスへのファイル共有のマウント	52
4.8. ファイル共有の削除	53
4.9. DASHBOARD を使用したパブリックのファイル共有および共有種別の作成の無効化	53

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、スケーラビリティが極めて高く、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。

本ガイドは、永続ストレージの作成と管理の手順を説明します。OpenStack では、永続ストレージは主に 3 つのサービスで提供されます。

- Block Storage (**openstack-cinder**)
- Object Storage (**openstack-swift**)
- Shared File Systems ストレージ (**openstack-manila**)

これらのサービスは、異なる種別の永続ストレージを提供します。それぞれのストレージは、異なるユースケースで独自の利点があります。本ガイドでは、一般的なエンタープライズストレージ要件に対する各ストレージの適合性について説明します。

OpenStack Dashboard またはコマンドラインクライアントを使用してクラウドストレージの管理を行うことができます。大半の手順は、これらのいずれかの方法を使用することができますが、一部の高度な手順はコマンドラインのみで実行可能となっています。本ガイドでは、可能な場合には Dashboard を使用する手順を記載しています。



注記

Red Hat OpenStack Platform 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。



重要

本ガイドでは、**crudini** を使用してカスタムのサービス設定を適用する方法について説明します。そのため、**crudini** パッケージを最初にインストールする必要があります。

```
# yum install crudini -y
```


第1章 OPENSTACK での永続ストレージの概要

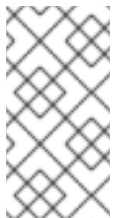
OpenStack は、一時ストレージと永続ストレージの2種類を認識します。一時ストレージは、特定の Compute インスタンスにのみ関連付けられるストレージです。インスタンスが終了されると、一時ストレージも終了します。この種別のストレージは、インスタンスのオペレーティングシステムの保存など、ランタイム時の基本的要件に対応する際に役立ちます。

一方、永続ストレージは、実行中のインスタンスからは独立して存続 (永続) するように設計されています。このストレージは、別のインスタンスまたは有効期間を超えた特定のインスタンスが再利用する必要のあるデータに使用されます。OpenStack は以下の種別の永続ストレージを使用します。

ボリューム

OpenStack Block Storage サービス (**openstack-cinder**) は、**ボリューム** を使用してブロックストレージデバイスにユーザーがアクセスできるようにします。一時ストレージを汎用の永続ストレージで拡張するために、インスタンスにボリュームを接続することができます。ボリュームは、任意でインスタンスからデタッチすることも、再度接続することもできます。接続したインスタンス経由でないと、ボリュームにはアクセスできません。

ボリュームには、バックアップやスナップショットを使用することで冗長性と災害復旧機能も備わっています。さらに、ボリュームを暗号化できるため、セキュリティが強化されます。ボリュームに関する情報は、「[2章Block Storage とボリューム](#)」を参照してください。



注記

一時ストレージは絶対に使用しないように、インスタンスを設定することも可能です。このような場合は、Block Storage サービスによりボリュームにイメージが書き込まれ、そのボリュームはインスタンスのブータブル root ボリュームとして利用することができます。

コンテナ

OpenStack Object Storage サービス (**openstack-swift**) は、メディアファイル、大容量のデータセット、ディスクイメージなど、静的データやバイナリー **オブジェクト** を保存するために使用する完全に分散されたストレージソリューションを提供します。Object Storage サービスは、**コンテナ** を使用してこれらのオブジェクトを整理します。

ボリュームのコンテンツにはインスタンス経由でしかアクセスできませんが、コンテナの中のオブジェクトは Object Storage REST API 経由でアクセスすることができます。そのため、クラウド内にあるほぼすべてのサービスが、Object Storage サービスをリポジトリとして使用することができます。たとえば、Data Processing サービス (**openstack-sahara**) は、Object Storage サービスから直接、そのバイナリー、データ入力、データ出力、テンプレートをすべて管理できます。

ファイル共有

OpenStack Shared File Systems サービス (**openstack-manila**) は、リモートにある共有可能なファイルシステムまたは **ファイル共有** を簡単にプロビジョニングする手段を提供します。ファイル共有により、クラウド内のテナントはストレージをオープンに共有できます。また、ファイル共有は、複数のインスタンスが同時に消費することが可能です。

各ストレージの種別は、特定のストレージ要件に対応するために設計されています。コンテナは、幅広いアクセスに対応できるように設計されているため、全ストレージ種別において最高レベルのスループット、アクセス、フォールトトレランスが備えられています。コンテナは主にサービスへの使用を対象としています。

一方で、ボリュームは主にインスタンスの消費に使用されます。ボリュームは、コンテナと同じレベルのアクセスやパフォーマンスには対応しにくくなっていますが、コンテナに比べ、機能セットが幅広く、ネイティブのセキュリティ機能も多くなっています。この点では、ファイル共有はボリューム

とよく似ていますが、複数のインスタンスにより消費可能である点が異なります。

以下のセクションでは、具体的なストレージ基準との関連において、各ストレージ種別のアーキテクチャーおよび機能セットについて考察します。

1.1. スケーラビリティおよびバックエンド

一般的に、クラスターストレージソリューションは、バックエンドのスケーラビリティが高くなっています。Red Hat Ceph を Block Storage のバックエンドとして使用する場合は、Ceph OSD (Object Storage Daemon) ノードをさらに追加することで、ストレージの容量および冗長性をスケーリングできます。Block Storage も Object Storage サービスも、バックエンドとして使用する Red Hat Ceph をサポートします。

Block Storage サービスは、個別のバックエンドとして複数のストレージソリューションを使用することができます。バックエンドレベルでは、バックエンドを追加してサービスを再起動することで、容量をスケーリングすることができます。Block Storage サービスは、数多くのサポートバックエンドソリューションをサポートしており、その一部には追加のスケーラビリティ機能が備えられています。

デフォルトでは、Object Storage サービスは設定済みの **ストレージノード** を使用しており、空き容量がある分だけ使用することができます。Object Storage サービスは、XFS および ext4 ファイルシステムをサポートし、いずれのサービスもスケーリングして、下層にあるブロックストレージで利用可能な容量を消費することができます。また、ストレージデバイスをストレージノードに追加して、容量をスケーリングすることも可能です。

Shared File Systems サービスは、別の **ストレージプール** からのストレージでバックアップされるファイル共有をプロビジョニングします。通常はサードパーティーのバックエンドサービスによって管理されるこのプールは、ファイルシステムレベルでストレージをファイル共有に提供します。Shared File Systems サービスでは NetApp および CephFS の両方をサポートしており、事前作成済みのボリューム (プロビジョニングしたファイル共有でストレージとして使用可能) のストレージプールを使用するように設定できます。いずれのデプロイメントにおいても、プールにボリュームを追加してスケーリングを行います。

1.2. アクセシビリティと管理

ボリュームは、インスタンスによってのみ消費され、1 回に 1 つのインスタンスにしか接続できず、またそのインスタンス内にしかマウントできません。ボリュームのスナップショットを作成して、クローンを作成する際や以前の状態にボリュームを復元する際に使用することができます (「[冗長性および災害復旧](#)」を参照)。Block Storage サービスでは、新規ボリュームを作成する際にこれらの設定を簡単に呼び出すことができるようにボリュームの各種設定 (例: サイズおよびバックエンド) をまとめた **ボリューム種別** を作成することも可能です。これらの種別はさらに **QoS** スペックに関連付けて、ユーザー向けに異なるストレージ階層を作成することができます。

ファイル共有は、ボリュームと同様にインスタンスにより消費されますが、ファイル共有の場合はインスタンス内に直接マウントすることができるので、ダッシュボードまたは CLI 経由で接続する必要がありません。ファイル共有は、同時に複数のインスタンスによりマウントすることができます。Shared File Systems サービスは、ファイル共有のスナップショットやクローン作成もサポートしており、(ボリューム種別と同様に) 設定をまとめた **共有種別** を作成することも可能です。

コンテナ内のオブジェクトは、API 経由でアクセスすることができ、クラウド内のインスタンスやサービスからもアクセスできるようになるため、サービスのオブジェクトリポジトリとして理想的です。たとえば、Image サービス (**openstack-glance**) は Object Storage サービスで管理するコンテナにイメージを保存することができます。

1.3. セキュリティ

Block Storage サービスは、**ボリュームの暗号化** を使用して基本的なデータセキュリティを確保します。これにより、静的キーでボリューム種別を暗号化するように設定できます。このキーは設定したボリュームの種別から作成したボリュームすべてを暗号化する際に使用されます。詳細は「[ボリュームの暗号化設定](#)」を参照してください。

一方で、オブジェクトとコンテナのセキュリティは、サービスおよびノードレベルで設定されます。Object Storage サービスでは、コンテナとオブジェクトに対するネイティブの暗号化はなく、Object Storage サービスによりクラウド内のアクセス性の優先順位が付けられるため、オブジェクトデータの保護はクラウドのネットワークセキュリティにのみ依存します。

Shared File Systems サービスでは、インスタンスの IP アドレス、ユーザー/グループ、または TLS 証明書別にアクセス制限することでファイル共有のセキュリティを確保することができます。さらに、一部の Shared File Systems サービスのデプロイメントは、別の **ファイル共有サービス** が備えられているため、ファイル共有ネットワークとファイル共有間の関係を管理することができます。ファイル共有サーバーによっては追加のネットワークセキュリティをサポートする (または必要とする) 場合があります。たとえば、CIFS ファイル共有サーバーでは LDAP、Active Directory または Kerberos 認証サービスのデプロイメントが必要です。

1.4. 冗長性および災害復旧

Block Storage サービスには、ボリュームのバックアップと復旧機能があり、ユーザーストレージの基本的な災害復旧を行います。バックアップで、ボリュームのコンテンツを保護することができます。それに加え、サービスはクローン作成以外にスナップショットもサポートしており、以前の状態にボリュームを復元する際に役立ちます。

マルチバックエンドの環境では、バックエンド間でボリュームを移行することも可能です。この機能は、メンテナンスでバックエンドをオフラインにする必要がある場合に役立ちます。バックアップは通常、データが保護できるように、ソースのボリュームとは別のストレージバックエンドに保存されます。ただし、スナップショットはソースのボリュームに依存するため、この方法を用いることはできません。

Block Storage サービスは、**整合性グループ** の作成もサポートしており、ボリュームをグループ化して同時にスナップショットの作成ができます。これにより、複数のボリューム間のデータの整合性レベルを向上することができます。詳しい情報は「[整合性グループの設定と使用](#)」を参照してください。

最後に、Block Storage サービスには **ボリュームの複製** 機能も備えられているため、ボリューム間でコンテンツを複製するように設定して、基本的な冗長性を提供することができます。



注記

ボリュームの複製は、特定のサードパーティーのバックエンド、およびそれに適したドライバを使用することでのみ利用可能です。

Object Storage サービスには、バックアップ機能がないため、すべてのバックアップはファイルシステムまたはノードレベルで行う必要があります。ただし、このサービスにはより強力な冗長機能とフォールトトレランスが備えられており、Object Storage サービスの最も基本的なデプロイメントでさえ、複数回オブジェクトを複製します。**dm-multipath** などのフェイルオーバー機能を使用して冗長性を強化することができます。

Shared File Systems サービスには、ファイル共有向けのバックアップ機能は組み込まれていませんが、スナップショットを作成してクローンを作成したり、復元したりすることができます。

第2章 BLOCK STORAGE とボリューム

Block Storage サービス (**openstack-cinder**) は、全ボリュームの管理タスク、セキュリティー、スケジューリング、全体を管理します。Compute インスタンスでは、永続ストレージとしてボリュームが主に使用されます。

2.1. バックエンド

デフォルトでは、Block Storage サービスは、ボリュームのリポジトリとして LVM バックエンドを使用します。このバックエンドは、テスト環境に適していますが、Enterprise 環境にはより堅牢なバックエンドをデプロイすることを推奨します。

この環境に Red Hat OpenStack Platform をデプロイする際は director を使用することを推奨します。director を使用することで、Block Storage サービス (拡張でバックエンドも含む) など、各サービスが正しく設定されるようにします。director には、複数のバックエンド設定が統合されています。

Red Hat OpenStack Platform は、Block Storage バックエンドとして [Red Hat Ceph Storage](#) および NFS をサポートします。OpenStack を使用した Ceph のデプロイメントの方法は、『[Deploying an Overcloud with Containerized Red Hat Ceph](#)』を参照してください。

オーバークラウドで NFS ストレージを設定する方法は、『[NFS ストレージの設定](#)』 (『[オーバークラウドの高度なカスタマイズ](#)』 guide) を参照してください。

サードパーティーのストレージプロバイダー

Block Storage サービスをサポート対象のサードパーティー製ストレージアプライアンスを使用するように設定することも可能です。director には、以下が簡単にデプロイされるように、必要なコンポーネントが含まれています。

- [Dell EMC PS Series](#)
- [Dell Storage Center](#)
- [NetApp](#) (サポートされているアプライアンス)
- [Fujitsu Eternus](#) もバックエンドとしてサポートされていますが、director にはまだ統合されていません。

サポートされるバックエンドアプライアンスとドライバーの完全な一覧は『[Component, Plug-In, and Driver Support in RHEL OpenStack Platform](#)』を参照してください。

2.2. BLOCK STORAGE サービスの管理

以下の手順では、Block Storage サービスをニーズに合わせて設定する方法を説明します。以下の手順はすべて管理者権限が必要です。

2.2.1. ボリューム種別へのボリューム設定の関連付け

OpenStack では、ボリューム種別を作成することができ、種別に関連付けられた設定を適用することができます。そのため、ボリュームの作成時 (『[ボリュームの作成](#)』) や作成後にも (『[ボリューム種別の変更](#)』) これらの設定を適用することができます。たとえば、以下のような関連付けが可能です。

- ボリュームの暗号化/非暗号化 (『[ボリューム種別の暗号化設定](#)』)

- ボリュームが使用するバックエンド (「[ボリュームを作成するバックエンドの指定](#)」 および「[バックエンド間での移行](#)」)
- Quality-of-Service (QoS) スペック

設定は、「追加スペック」と呼ばれるキーと値のペアを使用してボリューム種別に関連付けられます。ボリュームの作成時にボリューム種別を指定する際には、Block Storage のスケジューラーがこれらのキーと値のペアを設定として適用します。また、複数のキーと値のペアを同じボリューム種別に関連付けることができます。

ボリューム種別は、異なるユーザーにストレージ階層を使用できるようにする機能をします。特定のパフォーマンス、耐障害性、およびその他の設定をキーと値のペアとしてボリューム種別に関連付けることにより、階層固有の設定を異なるボリューム種別にマップすることができます。マップされた階層固有の設定は、ボリュームの作成時に対応するボリューム種別を指定することによって適用が可能です。



注記

利用可能な追加スペックやサポートされている追加スペックは、ボリュームのドライバーにより異なります。有効な追加スペックの一覧については、ボリュームドライバーのマニュアルを参照してください。

2.2.1.1. ホストドライバーの機能一覧

利用可能な追加スペックやサポートされている追加スペックは、バックエンドのドライバーにより異なります。有効な追加スペックの一覧については、ボリュームドライバーのマニュアルを参照してください。

または、Block Storage ホストに直接クエリーを送信して、そのドライバーがサポートしている、明確に定義された標準の追加スペックを確認することができます。コマンドラインから、Block Storage サービスをホストするノードにログインしてから、以下のコマンドを実行します。

```
# cinder service-list
```

このコマンドは、各 Block Storage サービス (**cinder-backup**、**cinder-scheduler**、**cinder-volume**) のホストが含まれる一覧を返します。たとえば以下のとおりです。

```
+-----+-----+-----+-----+
| Binary | Host | Zone | Status ... |
+-----+-----+-----+-----+
| cinder-backup | localhost.localdomain | nova | enabled ... |
| cinder-scheduler | localhost.localdomain | nova | enabled ... |
| cinder-volume | localhost.localdomain@lvm | nova | enabled ... |
+-----+-----+-----+-----+
```

Block Storage サービスのドライバーの機能を表示するには (これによりサポートされる追加スペックを判断)、以下のコマンドを実行します。

```
# cinder get-capabilities VOLSVCHOST
```

VOLSVCHOST は **cinder-volume** のホストの完全な名前に置き換えます。以下に例を示します。

```
# cinder get-capabilities localhost.localdomain@lvm
+-----+-----+-----+-----+
| Volume stats | Value |
```

+-----+-----+	
description	None
display_name	None
driver_version	3.0.0
namespace	OS::Storage::Capabilities::localhost.loc...
pool_name	None
storage_protocol	iSCSI
vendor_name	Open Source
visibility	None
volume_backend_name	lvm
+-----+-----+	
+-----+-----+	
Backend properties	Value
+-----+-----+	
compression	{u'type': u'boolean', u'description'...
qos	{u'type': u'boolean', u'des ...
replication	{u'type': u'boolean', u'description'...
thin_provisioning	{u'type': u'boolean', u'description': u'S...
+-----+-----+	

Backend properties の列には設定可能な追加スペックキーの一覧が、**Value** の列には、backend properties に対する有効な値が表示されます。

2.2.1.2. ボリューム種別の作成と設定

1. Dashboard に管理ユーザーとしてログインして **管理 > ボリューム > ボリューム種別** を選択します。
2. **ボリューム種別の作成** をクリックします。
3. **名前** フィールドにボリューム種別の名前を入力します。
4. **ボリューム種別の作成** をクリックします。**ボリューム種別** の表に新しい種別が表示されます。
5. ボリューム種別の **追加スペックの表示** のアクションを選択します。
6. **作成** をクリックして **キー** と **値** を指定します。キーと値のペアは有効である必要があります。有効でない場合には、ボリュームの作成時にそのボリューム種別を指定するとエラーが発生してしまいます。
7. **作成** をクリックします。関連付けられた設定 (キー/値のペア) が **追加スペック** の表に表示されます。

デフォルトでは、すべてのボリューム種別が OpenStack の全テナントにアクセス可能です。アクセスが制限されたボリューム種別を作成する必要がある場合は、CLI から作成する必要があります。手順については「[プライベートのボリューム種別の作成と設定](#)」を参照してください。



注記

QoS スペックをボリューム種別に関連付けることも可能です。詳しい説明は、「[QoS スペックとボリューム種別の関連付け](#)」を参照してください。

2.2.1.3. ボリューム種別の編集

1. Dashboard に管理ユーザーとしてログインして **管理 > ボリューム > ボリューム種別** を選択します。

2. ボリューム種別の表で、ボリューム種別の **追加スペックの表示** のアクションを選択します。

3. このページの **追加スペック** 表では、以下のような操作を行うことができます。

- ボリューム種別への新規設定の追加。これには、**作成** をクリックして、ボリューム種別に対して、関連付ける新規設定のキー/値ペアを指定します。
- ボリューム種別に関連付けられている既存の設定の編集。これには、設定の **編集** アクションを選択します。
- ボリューム種別に関連付けられている既存の設定の削除。これには、追加スペックのチェックボックスを選択して、このダイアログ画面と次の画面で **追加スペックの削除** をクリックします。

2.2.1.4. ボリューム種別の削除

ボリューム種別を削除するには、**ボリューム種別** の表でそのボリューム種別のチェックボックスを選択して **ボリューム種別の削除** をクリックします。

2.2.1.5. プライベートのボリューム種別の作成と設定

デフォルトでは、すべてのボリューム種別が全テナントに対して表示されます。ボリューム種別の作成中に、**プライベート** に指定すると、この設定をオーバーライドすることができます。そのためには、その種別の **Is_Public** フラグを **False** に設定する必要があります。

プライベートのボリューム種別は、特定のボリューム設定に対するアクセスを制限するのに役立ちます。これは通常は、特定のテナントのみが使用可能とする必要のある設定です。たとえば、テスト中の新規バックエンドや超ハイパフォーマンスの設定などが例としてあげられます。

プライベートのボリューム種別を作成するには、以下のコマンドを実行します。

```
# cinder --os-volume-api-version 2 type-create --is-public false VTYPE
```

+ **VTYPE** は、プライベートのボリューム種別の名前に置き換えます。

デフォルトでは、プライベートのボリューム種別は、作成者のみがアクセス可能ですが、admin ユーザーは、以下のコマンドを使用するとプライベートのボリューム種別を特定/表示することができます。

```
# cinder --os-volume-api-version 2 type-list --all
```

このコマンドは、パブリックとプライベートの両方のボリューム種別を一覧表示します。一覧には、各ボリューム種別の名前と ID も表示されます。ボリューム種別にアクセスするには、そのボリューム種別の ID が必要となります。

プライベートのボリューム種別へのアクセスは、テナントレベルで許可されます。テナントがプライベートのボリューム種別にアクセスできるようにするには、以下のコマンドを実行します。

```
# cinder --os-volume-api-version 2 type-access-add --volume-type VTYPEID -project-id TENANTID
```

上記の設定で、

- **VTYPEID** は、プライベートのボリューム種別の ID に置き換えます。

- **TENANTID** は、**VTYPERID** へのアクセスを許可するプロジェクト/テナントの ID に置き換えます。

プライベートのボリューム種別にアクセス可能なテナントを確認するには、以下のコマンドを実行します。

```
# cinder --os-volume-api-version 2 type-access-list --volume-type VTYPE
```

プライベートのボリューム種別のアクセスリストからテナントを削除するには、以下のコマンドを実行します。

```
# cinder --os-volume-api-version 2 type-access-remove --volume-type VTYPE
--project-id TENANTID
```



注記

プライベートのボリューム種別へのアクセスは、デフォルトでは管理権限のあるユーザーのみが作成、表示、設定することが可能です。

2.2.2. Block Storage サービスの内部テナントの作成および設定

Block Storage 機能 (例: Image-Volume キャッシュ) の一部では、**内部テナント** の設定を必要とします。Block Storage サービスは、このテナント/プロジェクトを使用して、通常のユーザーに公開する必要のないブロックストレージアイテムを管理します。このようなアイテムの例として、頻繁にボリュームをクローン作成したり、ボリュームの一時コピーを移行したりするためにキャッシュされたイメージなどが挙げられます。

内部プロジェクトを設定するには、まず **cinder-internal** という名前の一般プロジェクトとユーザーを作成します。そのためには、コントローラーノードにログインして以下のコマンドを実行します。

```
# openstack project create --enable --description "Block Storage Internal
Tenant" cinder-internal
+-----+-----+
| Property | Value |
+-----+-----+
| description | Block Storage Internal Tenant |
| enabled | True |
| id | cb91e1fe446a45628bb2b139d7dccaef |
| name | cinder-internal |
+-----+-----+
# openstack user create --project cinder-internal cinder-internal
+-----+-----+
| Property | Value |
+-----+-----+
| email | None |
| enabled | True |
| id | 84e9672c64f041d6bfa7a930f558d946 |
| name | cinder-internal |
| project_id | cb91e1fe446a45628bb2b139d7dccaef |
| username | cinder-internal |
+-----+-----+
```

プロジェクトおよびユーザーを作成すると、それぞれの ID が表示される点に注意してください。Block Storage サービスがプロジェクトとユーザーの両方を内部プロジェクトとして使用するよう、それら

の ID を指定して設定します。この操作を行うには、各 Block Storage ノードで以下のコマンドを実行します。

```
# crudini --set /etc/cinder/cinder.conf DEFAULT
cinder_internal_tenant_project_id TENANTID
# crudini --set /etc/cinder/cinder.conf DEFAULT
cinder_internal_tenant_user_id USERID
```

TENANTID と **USERID** は、前のステップで作成した **cinder-internal** プロジェクトとユーザーのそれぞれの ID に置き換えます。たとえば、上記のステップで提供された ID を使用すると、以下のようなコマンドになります。

```
# crudini --set /etc/cinder/cinder.conf DEFAULT
cinder_internal_tenant_project_id cb91e1fe446a45628bb2b139d7dccaef
# crudini --set /etc/cinder/cinder.conf DEFAULT
cinder_internal_tenant_user_id 84e9672c64f041d6bfa7a930f558d946
```



注記

この手順では、director を使用せずにサービスを設定する必要があります。そのため、オーバークラウドを再デプロイまたは更新する際には、同じ手順を繰り返す必要がある場合があります。

2.2.3. Image-Volume キャッシュの設定および有効化

Block Storage サービスには、任意の **Image-Volume** キャッシュが含まれており、イメージからボリュームを作成する際にこのキャッシュを使用できます。このキャッシュは、頻繁に使用するイメージからボリュームを作成する際の時間を短縮するように設計されています。イメージからボリュームを作成する方法は「[ボリュームの作成](#)」を参照してください。

Image-Volume のキャッシュを有効化すると、ボリュームの初回作成時にベースとなったイメージのコピーが保存されます。この保存されたイメージは、Block Storage バックエンドのローカルにキャッシュされ、次回このイメージを使用してボリュームを作成する際のパフォーマンス向上に役立ちます。Image-Volume キャッシュは、サイズ (GB)、イメージ数、または両方を指定して上限を設定することができます。

Image-Volume キャッシュは、複数のバックエンドでサポートされます。サードパーティーのバックエンドを使用する場合は、Image-Volume キャッシュサポートに関する情報については、サードパーティーのドキュメントを参照してください。



注記

Image-Volume キャッシュでは、**内部テナント** を Block Storage サービスに設定する必要があります。方法は、「[Block Storage サービスの内部テナントの作成および設定](#)」を参照してください。

バックエンド (**BACKEND**) で Image-Volume キャッシュを有効化して設定するには、以下のコマンドを実行します。

```
# crudini --set /etc/cinder/cinder.conf BACKEND image_volume_cache_enabled
True
```

BACKEND は、ターゲットのバックエンドの名前に置き換えてください (例: **volume_backend_name** の値)。

デフォルトでは、Image-Volume キャッシュサイズはバックエンドによってのみ制限されます。最大サイズ (**MAXSIZE**、GB 単位) を設定するには、以下のコマンドを実行します。

```
# crudini --set /etc/cinder/cinder.conf BACKEND
image_volume_cache_max_size_gb MAXSIZE
```

または、イメージの最大数を設定することもできます (**MAXNUMBER**)。設定するには、以下のコマンドを実行します。

```
# crudini --set /etc/cinder/cinder.conf BACKEND
image_volume_cache_max_count MAXNUMBER
```

Block Storage サービスのデータベースは、タイムスタンプを使用して、キャッシュされた各イメージの最終使用日時をトラッキングします。**MAXSIZE** と **MAXNUMBER** のいずれか一方または両方が設定されている場合は、Block Storage サービスは必要に応じてキャッシュされたイメージを削除し、新たにイメージをキャッシュするためのスペースを解放します。Image-Volume キャッシュが上限に達すると、最も古いタイムスタンプが付いたキャッシュイメージが最初に削除されます。

Image-Volume キャッシュを設定した後は、Block Storage サービスを再起動します。そのためには、任意のコントローラーノードに **heat-admin** ユーザーとしてログインして、以下のコマンドを実行します。

```
# pcs resource restart openstack-cinder-volume
```



注記

この手順では、director を使用せずにサービスを設定する必要があります。そのため、オーバークラウドを再デプロイまたは更新する際には、同じ手順を繰り返す必要がある場合があります。

2.2.4. QoS スペックの使用

複数のパフォーマンス設定を単一の Quality-of-Service の仕様 (QoS スペック) にマップすることができます。これにより、ユーザータイプ別のパフォーマンス階層を提供することができます。

パフォーマンス設定はキーと値のペアとして QoS スペックにマップされます。これは、ボリュームの設定がボリューム種別に関連付けられる方法と似ていますが、QoS スペックの場合は以下の面でボリューム種別の場合とは異なります。

- QoS スペックは、ディスクの読み取り/書き込み操作を制限するなどのパフォーマンス設定を適用するのに使用されます。利用可能かつサポートされているパフォーマンス設定はストレージドライバーによって異なります。
バックエンドがサポートしている QoS スペックを確認するには、そのバックエンドデバイスのボリュームドライバーのマニュアルを参照してください。
- ボリューム種別はボリュームに直接適用されるのに対して QoS スペックは直接適用されるのではなく、ボリューム種別に関連付けられます。また、ボリュームの作成時にボリューム種別を指定すると、そのボリューム種別に関連付けられた QoS スペックにマップされたパフォーマンス設定も適用されます。

2.2.4.1. QoS スペックの作成と設定

管理者は、「QoS スペック」の表で QoS スペックの作成および設定を行うことができます。同じ QoS スペックには、複数のキー/値のペアを関連付けることができます。

1. Dashboard に管理ユーザーとしてログインして **管理 > ボリューム > ボリューム種別** を選択します。
2. **QoS スペック** の表で **QoS スペックの作成** をクリックします。
3. **QoS スペック** の名前を入力します。
4. **使用者** フィールドで、QoS ポリシーを適用する先を指定します。

表2.1 使用者のタイプ

タイプ	説明
back-end	QoS ポリシーが Block Storage バックエンドに適用されます。
front-end	QoS ポリシーが Compute に適用されます。
両方	QoS ポリシーが Block Storage と Compute の両方に適用されます。

5. **作成** をクリックします。新規 QoS スペックが **QoS スペック** の表に表示されるはずです。
6. **QoS スペック** の表で、新規スペックの **スペックの管理** アクションを選択します。
7. **作成** をクリックして **キー** と **値** を指定します。キーと値のペアは有効である必要があります。有効でない場合には、ボリュームの作成時に、この QoS スペックに関連付けられたボリューム種別を指定するとエラーが発生してしまいます。
たとえば、容量ベースの読み取りの上限を **500 IOPS** に設定するには、以下のキー/値のペアを使用します。

```
read_iops_sec_per_gb=500
```



注記

容量ベースの QoS 上限についての詳しい情報は、[「容量ベースの QoS 上限の設定」](#)を参照してください。

8. **作成** をクリックします。関連付けられた設定 (キー/値のペア) が **キーと値のペア** の表に表示されます。

2.2.4.2. 容量ベースの QoS 上限の設定

ボリュームの種別を使用して、容量ベースの Quality of Service (QoS) をボリュームに実装することができます。これにより、プロビジョニングされるボリュームのサイズに基づいて、確定的な IOPS スループットを設定することができます。このように設定すると、ストレージリソースがユーザーにプロビジョニングされる方法が簡素化され、プロビジョニングされるボリュームのサイズに基づいて、事前に決定された (最終的には高度に予測可能な) スループット速度が提供されます。

特に、Block Storage サービスでは、実際にプロビジョニングされるサイズに基づいてボリュームに割り当てる IOPS を設定することができます。このスループットは、以下の QoS キーを使用して、1 GB あたりの IOPS で設定されます。

```
read_iops_sec_per_gb
write_iops_sec_per_gb
total_iops_sec_per_gb
```

これらのキーにより、プロビジョニングされるボリュームのサイズに応じてスケーリングするための読み取り、書き込み、IOPS 合計を設定することができます。たとえば、ボリューム種別に **read_iops_sec_per_gb=500** を指定した場合には、プロビジョニングされる 3 GB のボリュームには、読み取り IOPS が 1500 に自動設定されます。

容量ベースの QoS 上限はボリューム種別ごとに設定され、通常の QoS スペックと同様に構成されます。また、これらの上限は配下の Block Storage サービスにより直接サポートされており、特定のドライバーに依存しません。

ボリューム種別に関する詳しい情報は、「[ボリューム種別へのボリューム設定の関連付け](#)」および「[ボリューム種別の作成と設定](#)」を参照してください。QoS スペックの設定方法については、「[QoS スペックの使用](#)」を参照してください。



警告

容量ベースの QoS を使用して、接続済みのボリュームにボリューム種別を適用した場合 (またはボリューム種別を変更した場合) には、上限は適用されません。この上限は、そのボリュームをインスタンスから接続解除した後にのみ適用されます。

ボリューム種別の変更に関する情報は、「[ボリューム種別の変更](#)」を参照してください。

2.2.4.3. QoS スペックとボリューム種別の関連付け

管理者は、**ボリューム種別** の表で QoS スペックを既存のボリューム種別に関連付ける事ができます。

1. Dashboard に管理者としてログインして **管理 > ボリューム > ボリューム種別** を選択します。
2. **ボリューム種別** の表で、その種別の **QoS スペックの関連付けの管理** のアクションを選択します。
3. 関連付ける **QoS スペック** のリストから QoS スペックを選択します。
4. **割り当て** をクリックします。選択した QoS スペックが、編集したボリューム種別の **QoS スペックの関連付け** のコラムに表示されるようになります。

2.2.4.4. ボリューム種別からの QoS スペックの関連付け解除

1. Dashboard に管理者としてログインして **管理 > ボリューム > ボリューム種別** を選択します。
2. **ボリューム種別** の表で、その種別の **QoS スペックの関連付けの管理** のアクションを選択します。
3. 「関連付ける QoS スペック」のリストから **なし** を選択します。
4. **割り当て** をクリックします。選択した QoS スペックは、編集したボリューム種別の **QoS スペックの関連付け** のコラムに表示されなくなります。

2.2.5. ボリュームの暗号化設定

ボリュームの暗号化は、ボリュームのバックエンドのセキュリティを侵害されたり、完全に盗難されたりした場合に、基本的なデータ保護を提供します。Compute および Block Storage サービスを両方統合して、インスタンスがアクセスを読み込み、暗号化されたボリュームを使用できるようにします。



重要

現在、ボリュームの暗号化は、ファイルベースのボリューム (例: NFS) ではサポートされていません。

ボリュームの暗号化は、ボリューム種別を使用して適用されます。暗号化されたボリューム種別に関する情報は「[ボリューム種別の暗号化設定](#)」を参照してください。

2.2.5.1. ボリューム種別の暗号化設定

暗号化されたボリュームを作成するには、まず **暗号化されたボリューム種別** が必要です。ボリューム種別を暗号化するには、使用すべきプロバイダークラス、暗号、キーサイズを設定する必要があります。

1. Dashboard に管理ユーザーとしてログインして **管理 > ボリューム > ボリューム種別** を選択します。
2. 暗号化するボリュームの **アクション** コラムで、**暗号化設定の作成** を選択すると、**ボリューム種別の暗号化設定の作成** ウィザードが開きます。
3. このウィザードで、ボリューム種別の暗号化の **プロバイダー**、**制御場所**、**暗号**、**キーサイズ** を設定します。**説明** のコラムで各設定について説明されています。



重要

現在、唯一サポートされている **プロバイダー** は **LuksEncryptor** です。また、**暗号** で唯一サポートされているのは、**aes-xts-plain64** となっています。

4. **ボリューム種別の暗号化設定の作成** をクリックします。

ボリューム種別の暗号化設定が完了したら、その設定を使用して、暗号化されたボリュームを自動的に作成することができます。ボリューム種別の作成に関する詳しい情報は、「[ボリューム種別の作成と設定](#)」を参照してください。具体的には、**ボリュームの作成** ウィンドウの種別のドロップダウンから「ボリューム種別の暗号化設定」を選択します（「[ボリュームの基本的な使用方法と設定](#)」を参照）。

暗号化されたボリューム種別の設定を再構成することも可能です。ボリューム種別の **アクション** コラムから **暗号化設定の更新** を選択して、**ボリューム種別の暗号化設定の更新** ウィザードを開きます。

プロジェクト > コンピュート > ボリューム にある **ボリューム** テーブルの **暗号化** コラムでは、ボリュームが暗号化されているかが分かります。暗号化されている場合には、暗号化のコラムの **はい** をクリックすると暗号化設定が表示されます。

2.2.6. ボリュームを複数のバックエンドに割り当てる方法の設定

Block Storage サービスが複数のバックエンドを使用するように設定されている場合には、設定済みのボリューム種別を使用して、ボリュームの作成先を指定することができます。詳しくは、「[ボリュームを作成するバックエンドの指定](#)」を参照してください。

ボリュームの作成時にバックエンドを指定していない場合には、Block Storage サービスにより、自動

的に選択されます。Block Storage は、最初に定義したバックエンドをデフォルトとして設定します。このバックエンドは、容量がなくなるまで使用されます。容量がなくなった時点で、Block Storage は 2 番目に定義されたバックエンドをデフォルトに設定し、その容量がなくなるとさらに次のバックエンドがデフォルトに設定されるようになっています。

このメカニズムが必要な条件を満たさない場合には、フィルタースケジューラーを使用して、Block Storage がバックエンドを選択する方法を制御することができます。このスケジューラーは、以下の例に示したような異なるフィルターを使用して適切なバックエンドをトリアージすることができます。

AvailabilityZoneFilter

要求されたボリュームのアベイラビリティゾーン要件を満たさないバックエンドを除外します。

CapacityFilter

ボリュームを収容するのに十分な容量のあるバックエンドのみを選択します。

CapabilitiesFilter

ボリュームで指定した設定に対応可能なバックエンドのみを選択します。

InstanceLocality

クラスターが、同じノードに対してローカルのボリュームを使用するように設定します (OpenStack Data Processing サービスが有効化されている場合)。

フィルタースケジューラーを設定するには、以下の内容を記載した環境ファイルをデプロイメントに追加します。

```
parameter_defaults:
  ControllerExtraConfig: # 1
    cinder::config::cinder_config:
      DEFAULT/scheduler_default_filters:
        value:
'AvailabilityZoneFilter,CapacityFilter,CapabilitiesFilter,InstanceLocality'
```

- 1 ControllerExtraConfig:** フックとそのネストされているセクションを、既存の環境ファイルの **parameter_defaults:** セクションに追加することもできます。

もしくは、手動でフィルタースケジューラーを設定することもできます。そのためには、以下のステップを実行します。

1. Block Storage サービスをホストしているノードに **heat-admin** としてログインします。
2. **FilterScheduler** スケジューラードライバーを有効化します。

```
$ sudo crudini --set /etc/cinder/cinder.conf DEFAULT
scheduler_driver cinder.scheduler.filter_scheduler.FilterScheduler
```

3. アクティブにする必要のあるフィルターを設定します。

```
$ sudo crudini --set /etc/cinder/cinder.conf DEFAULT
scheduler_default_filters
AvailabilityZoneFilter,CapacityFilter,CapabilitiesFilter
```

4. スケジューラーが適切なバックエンドを選択する方法を設定します。

- スケジューラーが、空き容量の最も大きなバックエンドを常に変更するようにするには、以下のコマンドを実行します。

```
$ sudo crudini --set /etc/cinder/cinder.conf DEFAULT
scheduler_default_weighters AllocatedCapacityWeigher
$ sudo crudini --set /etc/cinder/cinder.conf DEFAULT
allocated_capacity_weight_multiplier -1.0
```

- スケジューラーが、すべての適切なバックエンドの中から無作為に変更するようにするには、以下のコマンドを実行します。

```
$ crudini --set /etc/cinder/cinder.conf DEFAULT
scheduler_default_weighters ChanceWeigher
```

5. Block Storage スケジューラーを再起動して、設定を適用します。そのためには、任意のコントローラーノードに **heat-admin** ユーザーとしてログインして、以下のコマンドを実行します。

```
$ pcs resource restart openstack-cinder-scheduler
```



注記

この手順では、director を使用せずにサービスを設定する必要があります。そのため、オーバークラウドを再デプロイまたは更新する際には、同じ手順を繰り返す必要がある場合があります。

2.2.7. 整合性グループの設定と使用

Block Storage サービスで **整合性グループ** を設定できます。この機能により、複数のボリュームを単一のエンティティとしてグループ化することができるので、操作はボリュームごとではなく、複数のボリュームに対して 1 度に実行することが可能となります。具体的には、このリリースでは整合性グループを使用して複数のボリュームのスナップショットを同時に作成することができます。その延長で、これらのボリュームを同時にリストアまたはクローンすることも可能です。

ボリュームは、複数の整合性グループのメンバーですが、ボリュームを整合性グループに追加すると、そのボリュームの削除、種別変更、移行はできません。

今回のリリースでは、整合性グループは、以下のストレージバックエンドのドライバーでのみサポートされます。

- EMC VMAX
- EMC VNX
- EMC ScaleIO
- EMC ExtremIO
- HP 3Par StorServ
- IBM DS8000
- IBM StorwizeSVC
- IBM XIV

- NetApp Data ONTAP
- NetApp ESERIES
- NetApp SolidFire

2.2.7.1. 整合性グループの設定

デフォルトでは、整合性グループ API は Block Storage のセキュリティポリシーにより無効化されます。この機能を使用するには、事前に有効化しておく必要があります。そのためには、Block Storage API ストレージ (**openstack-cinder-api**) をホストするノードの **/etc/cinder/policy.json** で関連の整合性グループのエントリーを編集します。このエントリーは以下のように表示されます。

```
"consistencygroup:create" : "group:nobody",
"consistencygroup:delete": "group:nobody",
"consistencygroup:update": "group:nobody",
"consistencygroup:get": "group:nobody",
"consistencygroup:get_all": "group:nobody",
"consistencygroup:create_cgsnapshot" : "group:nobody",
"consistencygroup:delete_cgsnapshot": "group:nobody",
"consistencygroup:get_cgsnapshot": "group:nobody",
"consistencygroup:get_all_cgsnapshots": "group:nobody",
```

セキュリティを強化するには、整合性グループの API およびボリューム種別管理の API のパーミッションをいずれも同じに設定します。デフォルトでは、ボリューム種別管理の API は (同じ **/etc/cinder/policy.json** ファイルで) **"rule:admin_or_owner"** に設定されています。

```
"volume_extension:types_manage": "rule:admin_or_owner",
```

推奨されているように、整合性グループの API を有効化するには、以下のようにエントリーを編集します。

```
"consistencygroup:create" : "rule:admin_api",
"consistencygroup:delete": "rule:admin_api",
"consistencygroup:update": "rule:admin_api",
"consistencygroup:get": "rule:admin_api",
"consistencygroup:get_all": "rule:admin_api",
"consistencygroup:create_cgsnapshot" : "rule:admin_api",
"consistencygroup:delete_cgsnapshot": "rule:admin_api",
"consistencygroup:get_cgsnapshot": "rule:admin_api",
"consistencygroup:get_all_cgsnapshots": "rule:admin_api",
```


注記

整合性グループ機能をすべてのユーザーが利用できるようにすることも可能です。これには、**rule:admin_or_owner** を使用して、ユーザーが独自の整合性グループを作成、使用、管理できるように API ポリシーエントリを設定します。

```
"consistencygroup:create" : "rule:admin_or_owner",
"consistencygroup:delete": "rule:admin_or_owner",
"consistencygroup:update": "rule:admin_or_owner",
"consistencygroup:get": "rule:admin_or_owner",
"consistencygroup:get_all": "rule:admin_or_owner",
"consistencygroup:create_cgsnapshot" : "rule:admin_or_owner",
"consistencygroup:delete_cgsnapshot": "rule:admin_or_owner",
"consistencygroup:get_cgsnapshot": "rule:admin_or_owner",
"consistencygroup:get_all_cgsnapshots": "rule:admin_or_owner",
```

整合性グループ API を有効にした後には、Block Storage API サービスを再起動します。そのためには、任意のコントローラーノードに **heat-admin** ユーザーとしてログインして、以下のコマンドを実行します。

```
# pcs resource restart openstack-cinder-api
```

注記

この手順では、director を使用せずにサービスを設定する必要があります。そのため、オーバークラウドを再デプロイまたは更新する際には、同じ手順を繰り返す必要がある場合があります。

2.2.7.2. 整合性グループの作成および管理

整合性グループの API を有効化した後には、以下の手順で、整合性グループの作成を開始することができます。

1. Dashboard で管理者ユーザーとして **プロジェクト > コンピュート > ボリューム > ボリュームの整合性グループ** を選択します。
2. **整合性グループの作成** をクリックします。
3. ウィザードの **整合性グループの情報** タブで、整合性グループの名前と説明を入力します。次に **アベイラビリティゾーン** を指定します。
4. 整合性グループにボリューム種別を追加することもできます。整合性グループにボリュームを作成する際には、Block Storage サービスにより、これらのボリューム種別から互換性のある設定が適用されます。ボリューム種別を追加するには、**利用可能な全ボリューム種別** 一覧から追加するボリューム種別の **+** ボタンをクリックします。
5. **整合性グループの作成** をクリックすると、**ボリュームの整合性グループ** テーブルに表示されずです。

アクション コラムから **整合性グループの編集** を選択して、整合性グループの名前または説明を変更することができます。

さらに、以下の手順に従って、直接整合性グループにボリュームを追加または削除することもできます。

1. Dashboard で管理者ユーザーとして **プロジェクト > コンピュート > ボリューム > ボリュームの整合性グループ** を選択します。
2. 設定する整合性グループを特定します。その整合性グループの **アクション** コラムで、**ボリュームの管理** を選択して、**整合性グループボリュームの追加/削除** ウィザードを起動します。
 - a. 整合性グループにボリュームを追加するには、**利用可能な全ボリューム** 一覧から追加するボリュームの **+** ボタンをクリックします。
 - b. 整合性グループからボリュームを削除するには、**選択済みのボリューム** 一覧から削除するボリュームの **-** ボタンをクリックします。
3. **整合性グループの編集** をクリックします。

2.2.7.3. 整合性グループのスナップショットの作成および管理

整合性グループにボリュームを追加した後に、そこからスナップショットを作成することができます。その前に、まず **openstack-cinder-api** をホストするノード上のコマンドラインから **admin** としてログインして、以下を実行します。

```
# export OS_VOLUME_API_VERSION=2
```

このコマンドを実行すると、クライアントが **openstack-cinder-api** のバージョン **2** を使用するよう設定されます。

利用可能な整合性グループ (およびそれらに対応する ID。後で必要となります) をすべて表示するには以下を実行します。

```
# cinder consisgroup-list
```

整合性グループを使用してスナップショットを作成するには以下を実行します。

```
# cinder cgsnapshot-create --name CGSNAPNAME --description "DESCRIPTION"
CGNAMEID
```

上記の設定で、

- **CGSNAPNAME** はスナップショットの名前に置き換えてください (任意)。
- **DESCRIPTION** はスナップショットの説明に置き換えてください (任意)。
- **CGNAMEID** は整合性グループの名前または ID に置き換えてください。

利用可能な整合性グループのスナップショットの全一覧を表示するには、以下を実行します。

```
# cinder cgsnapshot-list
```

2.2.7.4. 整合性グループのクローン作成

整合性グループを使用して、事前に設定されたボリューム群を一括で同時に作成することもできます。この操作は、既存の整合性グループをクローンするか、整合性グループのスナップショットを復元することによって実行できます。いずれのプロセスも同じコマンドを使用します。

既存の整合性グループのクローンを作成します。

-

```
# cinder consisgroup-create-from-src --source-cg CGNAMEID --name CGNAME --  
description "DESCRIPTION"
```

上記のコマンドで、**CGNAMEID** はクローン作成する整合性グループの名前または ID (任意) に、**CGNAME** は整合性グループの名前 (任意) に、**DESCRIPTION** は整合性グループの説明 (任意) に置き換えてください。

整合性グループのスナップショットから整合性グループを作成します。

```
# cinder consisgroup-create-from-src --cgsnapshot CGSNAPNAME --name CGNAME  
--description "DESCRIPTION"
```

CGSNAPNAME は、整合性グループの作成に使用するスナップショットの名前または ID に置き換えてください。

2.2.8. バックアップの管理

以下のセクションでは、Block Storage サービスでのボリュームのバックアップ設定をカスタマイズする方法を説明します。

2.2.8.1. テナントのバックアップクォータの表示と変更

大半のテナントストレージクォータ (ボリューム、ボリュームのストレージ、スナップショットの数) とは異なり、バックアップクォータは現在のところ、Dashboard では編集できません。

バックアップクォータは、コマンドラインインターフェース (**cinder quota-update** コマンド) でのみ編集することができます。

特定のテナント(**TENANTNAME**) のストレージクォータを確認するには、以下のコマンドを実行します。

```
# cinder quota-show TENANTNAME
```

特定のテナントで作成可能なバックアップの最大数 (**MAXNUM**) を更新するには、以下のコマンドを実行します。

```
# cinder quota-update --backups MAXNUM TENANTNAME
```

特定のテナント内の全バックアップの最大合計容量 (**MAXGB**) を更新するには、以下のコマンドを実行します。

```
# cinder quota-update --backup-gigabytes MAXGB TENANTNAME
```

特定のテナントのストレージクォータの使用状況を確認するには、以下のコマンドを実行します。

```
# cinder quota-usage TENANTNAME
```

2.2.8.2. Dashboard を使用したボリュームバックアップ管理の有効化

Dashboard を使用してボリュームの作成、表示、削除、復元ができるようになりました。これらの操作を実行するには、プロジェクト > コンピュート > ボリューム > ボリュームバックアップタブを開いてください。

ただし、**ボリュームバックアップ** タブは、デフォルトでは有効化されません。有効にするには、Dashboard を適切に設定する必要があります。

1. `/etc/openstack-dashboard/local_settings` を開きます。
2. 以下の設定を検索します。

```
OPENSTACK_CINDER_FEATURES = {
    'enable_backup': False,
}
```

この設定を以下のように変更します。

```
OPENSTACK_CINDER_FEATURES = {
    'enable_backup': True,
}
```

3. `httpd` サービスを再実行して、Dashboard を再起動します。

```
# systemctl restart httpd.service
```



注記

この手順では、director を使用せずにサービスを設定する必要があります。そのため、オーバークラウドを再デプロイまたは更新する際には、同じ手順を繰り返す必要がある場合があります。

2.2.8.3. バックアップリポジトリとしての NFS 共有の設定

デフォルトでは、Block Storage サービスは Object Storage サービスをバックアップリポジトリとして使用しますが、Block Storage サービスが Object Storage サービスの代わりに既存の NFS 共有をバックアップリポジトリとして使用するよう設定することが可能です。この設定は、以下の手順に従って行います。

1. バックアップサービス (**openstack-cinder-backup**) をホストしているノードに、管理権限のあるユーザーとしてログインします。
2. Block Storage サービスが NFS バックアップドライバー (`cinder.backup.drivers.nfs`) を使用するよう設定します。

```
# crudini --set /etc/cinder/cinder.conf DEFAULT backup_driver
cinder.backup.drivers.nfs
```

3. バックアップリポジトリとして使用する NFS 共有の詳細を設定します。

```
# crudini --set /etc/cinder/cinder.conf DEFAULT backup_share
NFSHOST:PATH
```

上記の設定で、

- **NFSHOST** は、NFS サーバーの IP アドレスまたはホスト名に置き換えます。
- **PATH** は、**NFSHOST** 上の NFS 共有の絶対パスに置き換えます。

4. NFS 共有のマウントオプション設定を指定するには、以下のコマンドを実行します。

```
# crudini --set /etc/cinder/cinder.conf DEFAULT backup_mount_options
NFSMOUNTOPTS
```

NFSMOUNTOPTS には、NFS マウントオプションのコンマ区切りの一覧を指定します (例: `rw, sync`)。サポートされているマウントオプションについての詳しい情報は、**nfs** および **mount** の **man** ページを参照してください。

5. Block Storage バックアップサービスを再起動して、変更を適用します。

```
# systemctl restart openstack-cinder-backup.service
```



注記

この手順では、director を使用せずにサービスを設定する必要があります。そのため、オーバークラウドを再デプロイまたは更新する際には、同じ手順を繰り返す必要がある場合があります。

2.2.8.3.1. 異なるバックアップファイルサイズの設定

バックアップサービスのバックアップするファイルのサイズは、最大 **バックアップファイルサイズ** を上限としています。ボリュームのバックアップがこの値を超える場合には、作成されるバックアップは複数のチャンクに分割されます。デフォルトのバックアップファイルサイズは 1.8 GB です。

異なるバックアップファイルサイズを設定するには、以下のコマンドを実行します。

```
# crudini --set /etc/cinder/cinder.conf DEFAULT backup_file_size SIZE
```

SIZE は指定するファイルサイズ (バイト単位) に置き換えます。Block Storage バックアップサービスを再起動して、変更を適用します。

```
# systemctl restart openstack-cinder-backup.service
```



注記

この手順では、director を使用せずにサービスを設定する必要があります。そのため、オーバークラウドを再デプロイまたは更新する際には、同じ手順を繰り返す必要がある場合があります。

2.3. ボリュームの基本的な使用方法と設定

以下の手順では、基本的なエンドユーザー向けのボリューム管理方法について説明します。これらの手順には管理者の権限は必要ありません。

2.3.1. ボリュームの作成

1. Dashboard で **プロジェクト > コンピュート > ボリューム** を選択します。
2. **ボリュームの作成** をクリックして、以下のフィールドを編集します。

フィールド	説明
ボリューム名	ボリュームの名前
説明	ボリュームの簡単な説明 (オプション)
タイプ	<p>オプションのボリューム種別 (「ボリューム種別へのボリューム設定の関連付け」 を参照)</p> <p>複数の Block Storage バックエンドがある場合には、このフィールドを使用して特定のバックエンドを選択します。詳しくは、 「ボリュームを作成するバックエンドの指定」 を参照してください。</p>
容量 (GB)	ボリュームの容量 (ギガバイト単位)
アベイラビリティゾーン	<p>アベイラビリティゾーン (論理サーバーグループ) は、ホストアグリゲートと併せて、OpenStack 内のリソースを分離する一般的な方法です。アベイラビリティゾーンは、インストール中に定義されます。アベイラビリティゾーンとホストについてのさらに詳しい説明は、 「Manage Host Aggregates」 を参照してください。</p>

3. ボリュームソース を指定します。

ソース	説明
ソースの指定なし (空のボリューム)	ボリュームは空となり、ファイルシステムやパーティションテーブルは含まれません。
スナップショット	<p>既存のスナップショットをボリュームソースとして使用します。このオプションを選択すると、「スナップショットをソースとして使用する」の一覧が新たに表示され、スナップショットを選択できるようになります。ボリュームのスナップショットについての詳しい情報は、 「ボリュームスナップショットの作成、使用、削除」 を参照してください。</p>
イメージ	<p>既存のイメージをボリュームソースとして使用します。このオプションを選択すると、「イメージをソースとして使用する」の一覧が新たに表示され、イメージを選択できるようになります。</p>
ボリューム	<p>既存のボリュームをボリュームソースとして使用します。このオプションを選択すると、「ボリュームをソースとして使用する」の一覧が新たに表示され、ボリュームを選択できるようになります。</p>

4. **ボリュームの作成** をクリックします。ボリュームが作成されると、**ボリューム** の表に名前が表示されます。

後ほどボリュームの種別を変更することも可能です。詳しくは「[ボリューム種別の変更](#)」を参照してください。

2.3.2. ボリュームを作成するバックエンドの指定

複数の Block Storage バックエンドが設定された場合には、必ず、バックエンドごとにボリューム種別を作成する必要があります。その種別を使用して、作成したボリュームに、どのバックエンドを使用すべきかを指定することができます。ボリューム種別の詳しい情報は、「[ボリューム種別へのボリューム設定の関連付け](#)」を参照してください。

ボリュームの作成時にバックエンドを指定するには「種別」のドロップダウンリストから適切なボリューム種別を選択します（「[ボリュームの作成](#)」を参照）。

ボリュームの作成時にバックエンドを指定しない場合には、Block Storage サービスにより自動的に選択されます。デフォルトでは、このサービスは、最も空き容量の多いバックエンドを選択します。また、Block Storage サービスが利用可能な全バックエンドから無作為に選択するように設定することも可能です。詳しくは「[ボリュームを複数のバックエンドに割り当てる方法の設定](#)」を参照してください。

2.3.3. ボリュームの名前と説明の編集

1. Dashboard で **プロジェクト > コンピュート > ボリューム** を選択します。
2. 対象のボリュームの **ボリュームの編集** ボタンをクリックします。
3. 必要に応じて、ボリュームの名前または説明を編集します。
4. **ボリュームの編集** をクリックして、変更を保存します。



注記

暗号化ボリュームを作成するには、最初にボリュームの暗号化専用設定されたボリューム種別を使用する必要があります。また、Compute サービスと Block Storage サービスの両方で、同じ静的キーを使用するように設定しておく必要があります。ボリュームの暗号化に必要な設定の方法についての説明は、「[ボリュームの暗号化設定](#)」を参照してください。

2.3.4. ボリュームの削除

1. Dashboard で **プロジェクト > コンピュート > ボリューム** を選択します。
2. **ボリューム** の表で、削除するボリュームを選択します。
3. **ボリュームの削除** をクリックします。



注記

スナップショットが存在する場合には、ボリュームは削除できません。スナップショットの削除手順については、「[ボリュームスナップショットの作成、使用、削除](#)」を参照してください。

2.3.5. インスタンスへのボリュームの接続と切断

インスタンスでは永続ストレージにボリュームを使用することができます。ボリュームは、1 度に 1 つのインスタンスにしか接続できません。インスタンスに関する詳しい情報は、[Red Hat OpenStack Platform](#) から、『インスタンスおよびイメージガイド』の「インスタンスの管理」を参照してください。

2.3.5.1. インスタンスへのボリュームの接続

1. Dashboard で **プロジェクト > コンピュート > ボリューム** を選択します。
2. 対象のボリュームの **接続の編集** アクションを選択します。ボリュームがインスタンスに接続されていない場合には、**インスタンスへの接続** のドロップダウンリストが表示されます。
3. **インスタンスへの接続** の一覧から、ボリュームの接続先となるインスタンスを選択します。
4. **ボリュームの接続** をクリックします。

2.3.5.2. インスタンスからのボリュームの切断

1. Dashboard で **プロジェクト > コンピュート > ボリューム** を選択します。
2. 対象のボリュームの **接続の管理** アクションを選択します。ボリュームがインスタンスに接続されている場合には、そのインスタンスの名前が **接続状況** の表に表示されます。
3. このダイアログ画面と次の画面で **ボリュームの切断** をクリックします。

2.3.6. ボリュームの読み取り専用設定

1 つのボリュームでコンテンツを編集できないようにして、複数のユーザーに共有アクセスを許可することができます。そのためには、以下のコマンドを実行して、ボリュームを **read-only** に設定します。

```
# cinder readonly-mode-update VOLUME true
```

VOLUME は、ターゲットボリュームの **ID** に置き換えます。

読み取り専用ボリュームを読み取り/書き込み可能に戻すには、以下のコマンドを実行します。

```
# cinder readonly-mode-update VOLUME false
```

2.3.7. ボリュームの所有者の変更

ボリュームの所有者を変更するには、ボリュームの譲渡を行います。ボリュームの譲渡は、ボリュームの所有者が開始し、ボリュームの新しい所有者が譲渡を承認すると、そのボリュームの所有権の変更が完了します。

2.3.7.1. コマンドラインを使用したボリュームの譲渡

1. コマンドラインから、ボリュームの現在の所有者としてログインします。
2. 利用可能なボリュームを一覧表示します。

```
# cinder list
```


3. 以下のコマンドを実行して、ボリュームの譲渡を開始します。

```
# cinder transfer-create VOLUME
```

VOLUME は譲渡するボリュームの名前または **ID** に置き換えます。

```
+-----+-----+
| Property | Value |
+-----+-----+
| auth_key | f03bf51ce7ead189 |
| created_at | 2014-12-08T03:46:31.884066 |
| id | 3f5dc551-c675-4205-a13a-d30f88527490 |
| name | None |
| volume_id | bcf7d015-4843-464c-880d-7376851ca728 |
+-----+-----+
```

cinder transfer-create コマンドはボリュームの所有権を消去し、譲渡用の **id** と **auth_key** を作成します。この値は別のユーザーに渡すことができます。受け取ったユーザーは、その値を使用して譲渡を承認し、ボリュームの新しい所有者となります。

4. 新規ユーザーがボリュームの所有権を宣言できる状態となりました。所有権を宣言するには、ユーザーは最初にコマンドラインからログインして以下のコマンドを実行する必要があります。

```
# cinder transfer-accept TRANSFERID TRANSFERKEY
```

TRANSFERID と **TRANSFERKEY** はそれぞれ、**cinder transfer-create** で返された **id** と **auth_key** の値に置き換えます。以下に例を示します。

```
# cinder transfer-accept 3f5dc551-c675-4205-a13a-d30f88527490
f03bf51ce7ead189
```



注記

利用可能なボリュームの譲渡をすべて表示するには、以下のコマンドを実行します。

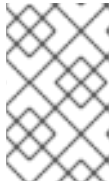
```
# cinder transfer-list
```

2.3.7.2. ダッシュボードを使用したボリュームの譲渡

Dashboard を使用したボリューム譲渡の作成

1. Dashboard にボリュームの所有者としてログインして **プロジェクト > ボリューム** を選択します。
2. 譲渡するボリュームの **アクション** のコラムで、**譲渡の作成** を選択します。
3. **ボリュームの譲渡の作成** ダイアログボックスで、譲渡名を入力して **ボリュームの譲渡の作成** をクリックします。
ボリュームの譲渡が作成され、**ボリュームの譲渡** の画面で **譲渡 ID** と **認証キー** を取得して譲渡先のプロジェクトに送信することができます。

譲渡認証情報のダウンロード ボタンをクリックして **transfer name**、**transfer ID**、**authorization key** が記載されている **.txt** ファイルをダウンロードします。



注記

認証キーは **ボリュームの譲渡** の画面にしか表示されません。この認証キーをなくした場合には、譲渡をキャンセルし、別の譲渡を作成して新たな認証キーを生成する必要があります。

4. **ボリュームの譲渡** の画面を閉じて、ボリュームの一覧に戻ります。
譲渡先のプロジェクトが譲渡を受理するまで、ボリュームのステータスは **awaiting-transfer** と表示されます。

Dashboard を使用したボリューム譲渡の受理

1. Dashboard にボリュームの譲渡先としてログインして **プロジェクト > ボリューム** を選択します。
2. **譲渡の受理** をクリックします。
3. **ボリュームの譲渡の受理** のダイアログボックスで、ボリュームの所有者から受け取った **譲渡 ID** と **認証キー** を入力して、**ボリュームの譲渡の受理** をクリックします。
譲渡先のプロジェクトのボリューム一覧に、そのボリュームが表示されるようになります。

2.3.8. ボリュームスナップショットの作成、使用、削除

ボリュームのスナップショットを作成することによって、ある特定の時点のボリュームの状態を保持することができます。そのスナップショットを使用して、新規ボリュームをクローン作成することが可能です。



注記

ボリュームのバックアップはスナップショットとは異なります。バックアップはボリューム内のデータを保持するのに対して、スナップショットはある特定の時点におけるボリュームの状態を保持します。また、スナップショットが存在している場合にはボリュームを削除することはできません。ボリュームのバックアップはデータ損失を防ぐ目的で使用されるのに対してスナップショットはクローン作成を円滑に行う目的で使用されます。

このため、スナップショットのバックエンドは、クローン作成中のレイテンシーを最小限に抑えるように、通常ボリュームのバックエンドと同じ場所に配置されます。一方、バックアップのリポジトリは通常、一般的なエンタープライズデプロイメント内の別の場所に配置されます (例: 異なるノード、物理ストレージ、あるいは別の地理的ロケーションの場合もあり)。これは、ボリュームのバックエンドが一切ダメージを受けないように保護することを目的とします。

ボリュームのバックアップについての詳しい情報は、[「ボリュームのバックアップと復元」](#)を参照してください。

ボリュームのスナップショットの作成手順

1. Dashboard で **プロジェクト > コンピュート > ボリューム** を選択します。
2. 対象のボリュームの **スナップショットの作成** アクションを選択します。

3. 作成するスナップショットの **スナップショット名** を指定して **ボリュームのスナップショットの作成** をクリックします。ボリュームのスナップショット タブに全スナップショットが表示されます。

ボリュームのスナップショット の表にスナップショットが表示されたら、そのスナップショットから新規ボリュームをクローン作成することができます。この操作を行うには、対象のボリュームの **ボリュームの作成** アクションを選択します。ボリュームの作成に関する詳しい説明は、[「ボリュームの作成」](#) を参照してください。

スナップショットを削除するには、**ボリュームスナップショットの削除** アクションを選択します。

OpenStack デプロイメントで Red Hat Ceph バックエンドを使用している場合には、[「Red Hat Ceph バックエンドにおけるスナップショットの保護と保護解除」](#) でスナップショットのセキュリティとトラブルシューティングについての詳しい情報を参照してください。

2.3.8.1. Red Hat Ceph バックエンドにおけるスナップショットの保護と保護解除

Red Hat Ceph を OpenStack デプロイメントのバックエンドとして使用する場合には、そのバックエンドでスナップショットの **保護** を設定することができます。OpenStack で (Dashboard または **cinder snapshot-delete** コマンドを使用して) 保護されているスナップショットの削除を試みると、操作は失敗します。

このようなエラーが発生した場合には、最初に Red Hat Ceph バックエンドでスナップショットを **保護解除** に設定すると、OpenStack で通常通りに削除することができるようになりますはずです。

関連する手順については、[「Protecting a Snapshot」](#) および [「Unprotecting a Snapshot」](#) を参照してください。

2.3.9. Image サービスにボリュームをアップロードする手順

イメージとして既存のボリュームを Image サービスに直接アップロードすることができます。これには、以下の手順を実行してください。

1. Dashboard で **プロジェクト > コンピュート > ボリューム** を選択します。
2. 対象のボリュームの **イメージにアップロード** アクションを選択します。
3. ボリュームの **イメージ名** を指定して、一覧から **ディスク形式** を選択します。
4. **アップロード** をクリックします。QEMU ディスクイメージのユーティリティにより、指定した名前を使用して、選択した形式で新規イメージがアップロードされます。

アップロードしたイメージを表示するには、**プロジェクト > コンピュート > イメージ** を選択します。新しいイメージが **イメージ** の表に表示されます。イメージの使用方法や設定方法に関する情報は、[Red Hat OpenStack Platform](#) から『**インスタンスおよびイメージガイド**』の「**イメージの管理**」を参照してください。

2.3.10. ボリューム種別の変更

ボリューム種別の変更 は、ボリューム種別 (およびその設定) を既存のボリュームに適用するプロセスです。ボリューム種別に関する情報は [「ボリューム種別へのボリューム設定の関連付け」](#) を参照してください。

既存のボリューム種別の有無に拘らず、ボリューム種別は変更できます。いずれの場合も、ボリューム種別の追加スペックがボリュームに適用できる場合のみ、ボリューム種別の変更は成功します。これは、事前定義の設定を適用する際や、ストレージの属性を既存のボリュームに適用する際に役立ちま

す。以下に例を示します。

- 異なるのバックエンドへボリュームを移行する場合 (「[バックエンド間での移行](#)」)
- ボリュームのストレージクラス/層を変更する場合

管理者権限のないユーザーは、自分が所有するボリューム種別しか変更できません。ボリューム種別を変更するには、以下のステップを実行します。

1. Dashboard で **プロジェクト > コンピュート > ボリューム** を選択します。
2. 移行するボリュームの **アクション** のコラムで、**ボリューム種別の変更** を選択します。
3. **ボリューム種別の変更** ダイアログで、**種別** のドロップダウンリストから新しいバックエンドを定義する、移行先のボリューム種別を選択します。



注記

別のバックエンドにボリュームを移行する場合は、**移行ポリシー** のドロップダウンリストから **要求時** を選択します。詳しい情報は「[バックエンド間での移行](#)」を参照してください。

4. **ボリューム種別の変更** をクリックして移行を開始します。

2.4. ボリュームの高度な設定

以下の手順では、ボリュームの高度な管理方法について説明します。

2.4.1. ボリュームのバックアップと復元

ボリュームのバックアップは、ボリュームのコンテンツの永続的なコピーです。ボリュームのバックアップは通常オブジェクトストアとして作成されるため、デフォルトでは、Object Storage サービスで管理されますが、バックアップに異なるリポジトリを設定することができます。OpenStack は、バックアップ用のバックエンドのオプションとして Red Hat Ceph、NFS をサポートしています。

ボリュームのバックアップの作成時には、バックアップのメタデータはすべて Block Storage サービスのデータベースに保管されます。**cinder** ユーティリティーはこのメタデータを使用して、バックアップからボリュームを復元します。このため、データベースの致命的なデータ損失から回復する際には、バックアップからボリュームを復元する前に Block Storage サービスのデータベースを復元する必要があります。これは、元のボリュームのバックアップのメタデータがすべて完全な状態で Block Storage サービスのデータベースが復元されることも前提としています。

データベースの致命的なデータ損失が発生した際にボリュームのバックアップのサブセットのみを保護するように設定する場合は、バックアップのメタデータをエクスポートすることも可能です。メタデータをエクスポートすると、エクスポートしたデータを後で Block Storage のデータベースに再度インポートしてボリュームのバックアップを通常のように復元することができます。

注記

ボリュームのバックアップはスナップショットとは異なります。バックアップはボリューム内のデータを保持するのに対して、スナップショットはある特定の時点におけるボリュームの状態を保持します。また、スナップショットが存在している場合にはボリュームを削除することはできません。ボリュームのバックアップはデータ損失を防ぐ目的で使用されるのに対してスナップショットはクローン作成を円滑に行う目的で使用されます。

このため、スナップショットのバックエンドは、クローン作成中のレイテンシーを最小限に抑えるように、通常ボリュームのバックエンドと同じ場所に配置されます。一方、バックアップのリポジトリは通常、一般的なエンタープライズデプロイメント内の別の場所に配置されます (例: 異なるノード、物理ストレージ、あるいは別の地理的ロケーションの場合もあり)。これは、ボリュームのバックエンドが一切ダメージを受けないように保護することを目的とします。

ボリュームのスナップショットに関する詳しい情報は、[「ボリュームスナップショットの作成、使用、削除」](#)を参照してください。

2.4.1.1. ボリュームの完全バックアップの作成

ボリュームをバックアップするには、**cinder backup-create** コマンドを使用します。デフォルトでは、このコマンドは、ボリュームの完全バックアップを作成します。ボリュームに既存のバックアップがある場合には、完全バックアップの代わりに、**増分** バックアップの作成を選択することができます (詳しくは [「ボリュームの増分バックアップの作成」](#) を参照)。

ボリュームのバックアップを作成できるのは、そのボリュームにアクセス可能なユーザーなので、管理権限があるユーザーは、ボリュームを誰が所有しているかに拘らず、任意のボリュームのバックアップを作成できることになります。詳しい説明は、[「admin としてのボリュームのバックアップ作成」](#)を参照してください。

1. バックアップするボリュームの **ID** または **表示名** を確認します。

```
# cinder list
```

2. 以下のコマンドを実行して、ボリュームをバックアップします。

```
# cinder backup-create VOLUME
```

VOLUME の箇所は、バックアップするボリュームの **ID** または **表示名** に置き換えます。以下に例を示します。

```
+-----+-----+
| Property | Value |
+-----+-----+
| id       | e9d15fc7-eeae-4ca4-aa72-d52536dc551d |
| name     | None |
| volume_id | 5f75430a-abff-4cc7-b74e-f808234fa6c5 |
+-----+-----+
```

注記

作成されるバックアップの **volume_id** は、ソースボリュームの **ID** と全く同じになります。

3. 以下のコマンドを実行して、ボリュームのバックアップ作成が完了したことを確認します。

```
# cinder backup-list
```

バックアップエントリーのステータスが **available** に変わったら、ボリュームのバックアップ作成は完了です。

この時点で、ボリュームのバックアップのメタデータをエクスポートして保管することもできます。これにより、Block Storage データベースで致命的なデータ損失が発生した場合でも、ボリュームのバックアップを復元することができます。以下のコマンドを実行します。

```
# cinder --os-volume-api-version 2 backup-export BACKUPID
```

BACKUPID はボリュームのバックアップの ID または名前に置き換えます。

```
+-----+-----+
| Property | Value |
+-----+-----+
| backup_service | cinder.backup.drivers.swift |
| backup_url | eyJzdGF0dXMiOiAiYXZhaWxhYmxlIiwgIm9iam... |
| | ...4NS0ZmY4MzBhZWYwNWUiLCAlc2l6ZSI6IDF9 |
+-----+-----+
```

ボリュームのバックアップのメタデータは **backup_service** と **backup_url** の値で構成されます。

2.4.1.1.1. admin としてのボリュームのバックアップ作成

管理者権限のあるユーザー (例: デフォルトの **admin** アカウントなど) は、OpenStack で管理されている任意のボリュームをバックアップすることができます。admin ユーザーが、admin 以外のユーザーの所有するボリュームをバックアップする場合には、そのボリュームの所有者からはバックアップはデフォルトでは表示されないように設定されます。

また、admin ユーザーとしてバックアップしたボリュームを、特定のテナントが利用できるようにすることも可能です。そのためには、以下のコマンドを実行します。

```
# cinder --os-auth-url KEYSTONEURL --os-tenant-name TENANTNAME --os-username USERNAME --os-password PASSWD backup-create VOLUME
```

上記の設定で、

- **TENANTNAME** は、バックアップを利用できるテナントの名前に置き換えます。
- **USERNAME** と **PASSWD** は、**TENANTNAME** 内のユーザーのユーザー名とパスワードに置き換えます。
- **VOLUME** は、バックアップするボリュームの名前または ID に置き換えます。
- **KEYSTONEURL** は、Identity サービスの URL エンドポイントに置き換えます (通常は `http://IP:5000/v2` の形式。IP は Identity サービスホストの IP アドレス)。

この操作を実行する場合は、作成されるバックアップの容量は、admin テナントではなく、**TENANTNAME** のクォータに対して加算されます。

2.4.1.2. ボリュームの増分バックアップの作成

デフォルトでは、**cinder backup-create** コマンドを実行すると、ボリュームの完全バックアップが作成されますが、ボリュームに既存のバックアップがある場合には、**増分** バックアップを作成することが可能です。

増分バックアップは、前回のバックアップ (完全または増分バックアップ) 以降に加えられた変更をキャプチャーします。ボリュームの完全バックアップを何度も定期的に行うと、時間の経過とともにボリュームの容量が拡大されて、リソースを集中的に使用することになります。この点に関しては、増分バックアップの場合には、一定期間の変更のみをキャプチャーして、リソースの使用を最小限に抑えることができます。

増分バックアップを作成するには、**--incremental** オプションを使用します。

```
# cinder backup-create VOLUME --incremental
```

VOLUME の箇所は、バックアップするボリュームの **ID** または **表示名** に置き換えます。増分バックアップは、NFS および Object Storage のバックアップリポジトリで完全にサポートされています。



注記

増分バックアップがすでに作成されている場合には、ベースとなっている完全バックアップを削除することはできません。また、完全バックアップに複数の増分バックアップがある場合、削除できるのは、最新の増分バックアップのみとなります。



警告

Red Hat Ceph Storage を Block Storage (cinder) ボリュームとバックアップの両方のバックエンドとして使用する場合には、増分バックアップを試みると、代わりに完全なバックアップが実行され、警告は表示されません。これは、既知の問題です ([BZ#1463058](#))。

2.4.1.3. Block Storage データベースでデータ損失が発生した後の復元

通常、Block Storage のデータベースのデータ損失が発生すると、ボリュームのバックアップを復元できなくなります。これは、Block Storage データベースにボリュームのバックアップサービス (openstack-cinder-backup) で必要とされるメタデータが含まれているためです。このメタデータは **backup_service** と **backup_url** の値で構成され、ボリュームのバックアップ作成後に ([「ボリュームの完全バックアップの作成」](#)に記載した手順に従って) エクスポートすることができます。

メタデータをエクスポートして保管した場合には、新しい Block Storage データベースにインポートすることができます (これにより、ボリュームのバックアップを復元することができます)。

1. 管理者権限を持つユーザーとして以下のコマンドを実行します。

```
# cinder --os-volume-api-version 2 backup-import backup_service backup_url
```

backup_service および **backup_url** は、エクスポートしたメタデータに置き換えます。たとえば、[「ボリュームの完全バックアップの作成」](#)でエクスポートしたメタデータを使用します。

```
# cinder --os-volume-api-version 2 backup-import
```

```
cinder.backup.drivers.swift eyJzdGF0dXMi...c2l6ZSI6IDF9
+-----+-----+
| Property | Value |
+-----+-----+
| id       | 77951e2f-4aff-4365-8c64-f833802eaa43 |
| name     | None |
+-----+-----+
```

2. メタデータが Block Storage サービスのデータベースにインポートされたら、通常のようにボリュームを復元することができます (「[バックアップからのボリュームの復元](#)」を参照)。

2.4.1.4. バックアップからのボリュームの復元

1. 使用するボリュームバックエンドの **ID** を確認します。

```
# cinder backup-list
```

Volume ID は、復元するボリュームの ID と一致する必要があります。

2. ボリュームのバックアップを復元します。

```
# cinder backup-restore BACKUP_ID
```

BACKUP_ID は使用するボリュームのバックアップ ID に置き換えます。

3. バックアップが必要なくなった場合には、削除します。

```
# cinder backup-delete BACKUP_ID
```

2.4.2. ボリュームの移行

Block Storage サービスでは、ホストまたはバックエンド間のボリューム移行ができます。ボリュームの移行は一部制限があります。

- 使用中のボリュームの移行は、ドライバーがサポートしているかどうかによります。
- スナップショットのあるボリュームは移行できません。
- 異なるバックエンドで (ドライバーが異なる) ボリューム間の移行は最適化されません。
- 使用中のボリュームの移行のターゲットには、iSCSI のブロックバックエンドデバイスが必要で、Ceph RADOS Block Device (RBD) などの非ブロックデバイスは使用できません。

移行のスピードは、ホストの設定と構成によって異なります。ドライバーを使用した移行の場合は、データの移動は OpenStack Block Storage サービス内ではなく、ストレージバックプレーンで実行されます。ドライバーを使用する最適なコピー操作は、ドライバーがその機能をサポートしている場合に、使用中でない RBD ボリュームで利用することが可能で、ボリューム種別の変更は必要ありません。そうでない場合には、データは Block Storage サービスを使用してホスト間でコピーされます。

2.4.2.1. ホスト間の移行

ホスト間でボリュームを移行する場合には、両方のホストが同じバックエンド上にある必要があります。ダッシュボードを使用してホスト間のボリュームの移行を行います。

1. Dashboard で **管理 > ボリューム** を選択します。
2. 移行するボリュームの **アクション** のコラムで、**ボリュームの管理** を選択します。
3. **ボリュームの管理** ダイアログでは、**移行先ホスト** ドロップダウンリストからボリュームを移行する先のホストを選択します。



注記

ホストの移行でドライバーの最適化をスキップするには、**強制ホストコピー** のチェックボックスを選択します。

4. **移行** をクリックして移行を開始します。

2.4.2.2. バックエンド間での移行

バックエンド間でボリュームを移行するには **ボリューム種別の変更** を行う必要があります。新規バックエンドに移行するためには、以下の条件を満たす必要があります。

1. 新規バックエンドは、別の **移行先のボリューム種別** の **追加スペック** として指定すること。
2. 移行先のボリューム種別に定義されるその他の追加スペックはすべて、ボリュームの移行元のボリューム種別と互換性があること。

詳細は、「[ボリューム種別へのボリューム設定の関連付け](#)」と「[ボリュームを作成するバックエンドの指定](#)」を参照してください。

追加スペックとしてバックエンドを定義する場合は、**キー** として **volume_backend_name** を使用します。これに対応する値は、Block Storage の設定ファイル (/etc/cinder/cinder.conf) に定義されているバックエンド名になります。このファイルでは、各バックエンドは独自のセクションに定義され、対応する名前は **volume_backend_name** のパラメーターに設定されています。

移行先のボリューム種別にバックエンドを定義すると、**ボリューム種別の変更** でバックエンドにボリュームを移行することができます。この際には、移行先のボリューム種別をボリュームに再適用し、それにより新しいバックエンドの設定が適用されます。方法は「[ボリューム種別の変更](#)」を参照してください。

以下の手順に従って設定します。

1. Dashboard で **プロジェクト > コンピュート > ボリューム** を選択します。
2. 移行するボリュームの **アクション** のコラムで、**ボリューム種別の変更** を選択します。
3. **ボリューム種別の変更** ダイアログで、**種別** のドロップダウンリストから新しいバックエンドを定義する、移行先のボリューム種別を選択します。
4. **移行ポリシー** から **要求時** を選択します。
5. **ボリューム種別の変更** をクリックして移行を開始します。

第3章 OBJECT STORAGE

OpenStack Object Storage (**openstack-swift**) は、コンテナ内にオブジェクト (データ) を格納します。コンテナとは、ファイルシステムにおけるディレクトリーと似ています。ただし、入れ子にはできません。コンテナは、あらゆるタイプの非構造化データを格納する簡単な方法をユーザーに提供します。たとえば、オブジェクトには写真、テキストファイル、イメージなどが含まれます。格納されるオブジェクトは圧縮されません。

3.1. OBJECT STORAGE サービスの管理

以下の手順では、Object Storage サービスをニーズに合わせてさらにカスタマイズする方法について説明します。

3.1.1. Erasure Coding の設定

Erasure coding (EC) は、データの断片化、拡張、冗長データによる暗号化、別の場所やストレージメディアセットでの保存の際にデータを保護する方法です。EC は、従来のレプリケーションより小さいストレージボリュームを使用して、必要とされる耐久性を取得します。レプリケーションファクターが 3 のものと比較すると、注意深くデプロイメントを行うことで、50% の節約が実現できます。ただし、ワークロードによっては、Erasure Coding がパフォーマンスに悪影響を与える可能性があります。

Erasure Coding は、Object Storage Service 向けにストレージポリシーとしてサポートされます。ストレージポリシーにより複数のオブジェクトリングを作成することでさまざまな目的のクラスターをセグメント化できます。Red Hat は、Erasure Coding およびレプリケーションストレージポリシーで使用するデバイスを分割することを推奨します。これにより、クラスターの動きをより簡単に分析することができます。

選択する方向性は、Erasure Coding ポリシーのデプロイの理由により異なります。主な検討事項は以下のとおりです。

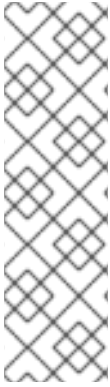
- 既存のインフラストラクチャーのレイアウト
- 専用の Erasure Coding ノード (または Erasure Coding デバイス) の追加コスト
- 目的とする使用モデル

3.1.1.1. Erasure Coding ポリシーの設定

Erasure Coding を使用するには、最初に `/etc/swift/swift.conf` で EC の追加のポリシーを定義します。以下の「[Erasure Coding ポリシーの例](#)」では、一般的な例を紹介しています。

Erasure Coding ポリシーの例

```
[storage-policy:1]
default = no
name = ec104
alias = myec,erasure_coding
policy_type = erasure_coding
ec_type = jerasure_rs_vand
ec_num_data_fragments = 10
ec_num_parity_fragments = 4
ec_object_segment_size = 1048576
```



注記

Object Storage ポリシーヘッダー (例: **[storage-policy:1]**) にはインデックスの番号が含まれます。今回の場合は **1** です。Object Storage のインデックス数は **0** から開始するので、上記の「[Erasure Coding ポリシーの例](#)」では、ポリシーインデックスが **0** の別のポリシーがすでに存在することを想定しています。以下に例を示します。

```
[storage-policy:0]
name = default
default = yes
```

Erasure Coding ポリシーの定義後に、関連のオブジェクトストレージリングを作成、設定する必要があります。説明は「[オブジェクトストレージリングの設定](#)」を参照してください。

以下の表では、「[Erasure Coding ポリシーの例](#)」で使用する各種パラメーターについて説明しています。

表3.1 ストレージポリシーのパラメーター

用語	説明
default	このポリシーがデフォルトのものかどうかを設定します (yes/no)。これは、 <code>/etc/swift/swift.conf</code> に複数のポリシーが定義されている場合に使用します。
name	標準のストレージポリシーのパラメーター
alias	ポリシーの別名をコンマ区切りにしたリスト
policy_type	この値を erasure_coding に設定して、Erasure Coding ポリシーであることを指定します。
ec_type	使用予定の Erasure Coding スキームを指定します。サポートされる値の一覧については表3.2「 サポートされる Erasure Coding スキーム 」を参照してください。
ec_num_data_fragments	データを構成するフラグメントの合計数
ec_num_parity_fragments	パリティを構成するフラグメントの合計数
ec_object_segment_size	セグメントをエンコーダー/デコーダーにフィードする前に増やすデータのバッファ量。デフォルト値は 1048576 です。

表3.2 サポートされる Erasure Coding スキーム

スキーム	説明/参照
liberasurecode_rs_vand	Vandermonde Reed-Solomon エンコーディング。 liberasurecode により実装されるソフトウェアのみのバックエンド

スキーム	説明/参照
<code>jerasure_rs_vand</code>	Jerasure をベースにする Vandermonde Reed-Solomon エンコーディング
<code>jerasure_rs_cauchy</code>	Jerasure をベースにする Cauchy Reed-Solomon エンコーディング (Jerasure の派生)
<code>flat_xor_hd_3</code> , <code>flat_xor_hd_4</code>	Flat-XOR ベースの HD 組み合わせコード (liberasurecode)
<code>isa_l_rs_vand</code>	Intel Storage Acceleration Library (ISA-L): SIMD accelerated Erasure Coding バックエンド

Object Storage サービスがオブジェクトを暗号化する際には、N 個のフラグメントに分割します。設定時に、フラグメントのうち何個がデータで、何個がパリティであるかを知っておくことが重要です。[Erasure Coding ポリシーの例](#)では、PyECLib はオブジェクトを 14 個のフラグメントに分割して、そのうち、実際のオブジェクトデータは 10 個で、パリティデータは 4 個 (計算は `ec_type` に左右される) です。このような設定では、システムは、ディスクの障害は 4 回分までであればデータの損失なしで済みます。その他に一般的に使用される設定は 4+2 (データフラグメント 4 個とパリティフラグメント 2 個) または 8+3 (データフラグメント 8 個とパリティフラグメント 3 個) です。



注記

この手順では、director を使用せずにサービスを設定する必要があります。そのため、オーバークラウドを再デプロイまたは更新する際には、同じ手順を繰り返す必要がある場合があります。



重要

ポリシーをデプロイして、そのポリシーでオブジェクトを作成した後はこれらの設定オプションの変更はできません。設定の変更が望ましい場合には、新規ポリシーを作成して、データを新しいコンテナに移行します。ただし、定義が済むと、ポリシーのインデックスは破棄できません。ポリシーを終了する場合には、ポリシーを無効にすることはできますが、削除はできません。基本的に、以前のポリシーが残っていてもパフォーマンスへの影響はありませんが、若干、管理の負担が出てきます。

3.1.1.2. オブジェクトストレージリングの設定

オブジェクトストレージは、**リング**と呼ばれるデータ構造を使用して、パーティション領域をクラスターに分散します。このパーティション領域は、Object Storage サービスのデータ永続性エンジン (data durability engine) の中核となります。これにより、Object Storage サービスが迅速かつ簡単にクラスター内の各パーティションを同期できるようになります。Swift のコンポーネントがデータと対話する必要がある場合には、ローカルでリング内を素早く検索して、各レプリカに使用可能なパーティションを見つけます。

Object Storage サービスにはすでに 3 つのリングがあり、各種データが格納されています。リングは、アカウント情報に 1 つ、(アカウント内のオブジェクトを整理するのに役立つ) コンテナに 1 つ、オブジェクトのレプリカに 1 つあります。Erasure Code をサポートするために、Erasure Code のチャンクを格納するために作成された追加のリングがあります。

たとえば、典型的なレプリケーションリングを作成するには、以下のコマンドを使用します。

```
# swift-ring-builder object-1.builder create 10 3 1
```

3 は、レプリカの数です。

以下のように、Erasure Coding のオブジェクトリングを作成するには、レプリカの数の部分にフラグメントの数を指定する必要があります。

```
# swift-ring-builder object-1.builder create 10 14 1
```

14 は、データフラグメント 10 とパリティフラグメント 4 (10+4) を合わせた設定です。

Erasure Coding ポリシーのオブジェクトリングで使用するデバイスを決定する際には、パフォーマンスの影響を考慮します。デプロイメントの前の設定に対して、テスト環境でパフォーマンスのベンチマーキングを実行することを推奨します。`/etc/swift/swift.conf` で Erasure Coding のポリシーを設定してオブジェクトリングを作成した後は、指定のポリシー名でコンテナを作成して通常通りに対話することで、アプリケーションでの Erasure Coding の使用準備が整います。

3.1.1.3. Erasure Coding の使用

新規の Erasure Coding ポリシーを定義して、そのオブジェクトストレージリングを設定した後は、初めて新しいコンテナを作成する際に EC ポリシーを使用することができます。複数のポリシーを定義した場合には、デフォルトのストレージポリシーが割り当てられたコンテナが作成されます。

新しいコンテナでデフォルト以外のストレージポリシーを使用するには、コンテナの作成時に特別なメタデータヘッダー送信する必要があります。たとえば、新規コンテナ (**CONTAINERNAME**) で「[Erasure Coding ポリシーの設定](#)」に定義した「[Erasure Coding ポリシーの例](#)」で提示したポリシーを使用するには、以下のコマンドを実行します。

```
# swift post -H "X-Storage-Policy:ec104" CONTAINERNAME
```

3.1.2. Fast-POST の設定

デフォルトでは、Object Storage サービスは、メタデータの一部でも変更があると必ず、オブジェクト全体をコピーします。**fast-post** 機能を使用することでこれを回避できます。これは、複数の大型オブジェクトのコンテンツタイプを変更する際に時間を節約する際に役立ちます。

Fast-Post 機能を有効化するには、Object Storage プロキシサービスの **object_post_as_copy** オプションを無効にします。これには、以下を実行します。

1. Object Storage プロキシサービスをホストするノードにログインします。
2. `/etc/swift/proxy-server.conf` を開きます。
3. 以下のように、**[app:proxy-server]** のセクションで、**object_post_as_copy** を **false** に設定します。

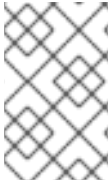
```
[app:proxy-server]
use = egg:swift#proxy
set log_name = proxy-server
...
object_post_as_copy = false
...
```

4. ノードで Object Storage プロキシサービスを再起動します。

```
# systemctl restart openstack-swift-proxy.service
# systemctl restart openstack-swift-object-expirer.service
```

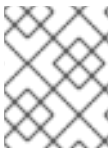
policy-0 以外のストレージポリシーが **/etc/swift/swift.conf** に記載されている場合には、以下も実行してください。

```
# systemctl restart openstack-swift-container-reconciler.service
```



注記

この手順では、director を使用せずにサービスを設定する必要があります。そのため、オーバークラウドを再デプロイまたは更新する際には、同じ手順を繰り返す必要がある場合があります。



注記

一般的なオーバークラウドのデプロイメントでは、Object Storage サービスはコントローラーノードにインストールされています。

3.1.3. Image サービスのバックエンドとしてのオブジェクトストレージの設定

デフォルトでは、OpenStack の Image サービスは、イメージおよびインスタンスのスナップショットを **/var/lib/glance/images/** のローカルのファイルシステムに保存します。または、(可能な場合には) Image サービスが Object Storage サービスにイメージとスナップショットを保存するように設定することも可能です。

この設定には、以下の手順を実行します。

1. root として Image サービスを実行中のノード (Identity も実行するコントローラーノード) にログインし、OpenStack の認証情報 (これは通常 **openrc** という名前のファイル) を読み込みます。

```
# source ~/openrc
```

2. Image サービスが **admin** ロールを持つ **services** テナントの一部であることを確認します。

```
# openstack user role list --project services glance
```

返されたロールの 1 つは、**admin** でなければなりません。

3. **/etc/glance/glance.conf** ファイルを開き、以下の行をコメントアウトします。

```
##### DEFAULT OPTIONS #####
#default_store = file
#filesystem_store_datadir = /var/lib/glance/images/
```

4. 同じファイル内で **DEFAULT OPTIONS** のセクションに以下の行を追加します。

```
default_store = swift
swift_store_auth_address = http://KEYSTONEIP:35357/v2.0/
swift_store_user = service:glance
swift_store_key = ADMINPW
swift_store_create_container_on_put = True
```


■

上記の設定で、

- **KEYSTONEIP** は、Identity サービスの IP アドレスに置き換えてください。
- **ADMINPW** は、`/etc/glance/glance-api.conf` ファイルの管理者パスワードの値に置き換えてください。

5. Image サービスを再起動して変更を適用します。

```
# systemctl restart openstack-glance-api
# systemctl restart openstack-glance-registry
```

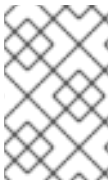
この時点から、(Dashboard または **glance** 経由) Image サービスにアップロードされたイメージは **glance** という名前の Object Storage コンテナに保存されるはずです。このコンテナは、サービスアカウントに存在します。

新規作成イメージが Image サービスに保存されたことを確認するには、以下のコマンドを実行します。

```
# ls /var/lib/glance/images
```

Dashboard または **glance image-list** により、イメージがアクティブな状態であることが報告されたら、以下のコマンドを実行してイメージがオブジェクトストレージにあるかどうかを確認できます。

```
# swift --os-auth-url http://KEYSTONEIP:5000/v2.0 --os-tenant-name service
--os-username glance --os-password ADMINPW list glance
```



注記

この手順では、director を使用せずにサービスを設定する必要があります。そのため、オーバークラウドを再デプロイまたは更新する際には、同じ手順を繰り返す必要がある場合があります。

3.1.4. 保存データの暗号化の有効化

デフォルトでは、Object Storage にアップロードされるオブジェクトは暗号化されずに保管されます。このため、ファイルシステムからオブジェクトに直接アクセスすることが可能です。このため、ディスクを破棄する前に適切に消去しなかった場合には、セキュリティリスクとなってしまいます。

OpenStack Key Manager (barbican) を使用して、保存されている swift オブジェクトを暗号化することができます。詳しい情報は、https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/13/html-single/manage_secrets_with_openstack_key_manager/#encrypt_at_rest_swift_objects を参照してください。

3.1.5. スタンドアロンの Object Storage クラスターのデプロイ

コンポーザブルロールの概念を採用して、最小限の追加のサービス (例: Keystone、HAProxy) を実装したスタンドアロンの Object Storage (openstack-swift) クラスターをデプロイすることができます。「[roles_data ファイルの作成](#)」の項には、ロールについての情報が記載されています。

3.1.5.1. roles_data.yaml ファイルの作成

1. `/usr/share/openstack-tripleo-heat-templates` から `roles_data.yaml` をコピーします。
2. 新規ファイルを編集します。
3. 不要なコントローラーロールを削除します (例: Aodh*, Ceilometer*, Ceph*, Cinder*, Glance*, Heat*, Ironic*, Manila*, Mistral*, Nova*, Octavia*, Swift*).
4. `roles_data.yaml` 内で `ObjectStorage` を見つけます。
5. このロールを、同じファイル内の新しいロールにコピーして、**ObjectProxy** という名前を付けます。
6. このロールの **SwiftStorage** は **SwiftProxy** に置き換えます。

以下の `roles_data.yaml` ファイルの例には、サンプルのロールを記載しています。

```
- name: Controller
  description: |
    Controller role that has all the controller services loaded and handles
    Database, Messaging and Network functions.
  CountDefault: 1
  tags:
  - primary
  - controller
  networks:
  - External
  - InternalApi
  - Storage
  - StorageMgmt
  - Tenant
  HostnameFormatDefault: '%stackname%-controller-%index%'
  ServicesDefault:
  - OS::TripleO::Services::AuditD
  - OS::TripleO::Services::CACerts
  - OS::TripleO::Services::CertmongerUser
  - OS::TripleO::Services::Clustercheck
  - OS::TripleO::Services::Docker
  - OS::TripleO::Services::Ec2Api
  - OS::TripleO::Services::Etc
  - OS::TripleO::Services::HAproxy
  - OS::TripleO::Services::Keepalived
  - OS::TripleO::Services::Kernel
  - OS::TripleO::Services::Keystone
  - OS::TripleO::Services::Memcached
  - OS::TripleO::Services::MySQL
  - OS::TripleO::Services::MySQLClient
  - OS::TripleO::Services::Ntp
  - OS::TripleO::Services::Pacemaker
  - OS::TripleO::Services::RabbitMQ
  - OS::TripleO::Services::Securetty
  - OS::TripleO::Services::Snmp
  - OS::TripleO::Services::Sshd
  - OS::TripleO::Services::Timezone
  - OS::TripleO::Services::TripleoFirewall
  - OS::TripleO::Services::TripleoPackages
```



```

- OS::Triple0::Services::Vpp

- name: ObjectStorage
  CountDefault: 1
  description: |
Swift Object Storage node role
  networks:
- InternalApi
- Storage
- StorageMgmt
  disable_upgrade_deployment: True
  ServicesDefault:
- OS::Triple0::Services::AuditD
- OS::Triple0::Services::CACerts
- OS::Triple0::Services::CertmongerUser
- OS::Triple0::Services::Collectd
- OS::Triple0::Services::Docker
- OS::Triple0::Services::FluentdClient
- OS::Triple0::Services::Kernel
- OS::Triple0::Services::MySQLClient
- OS::Triple0::Services::Ntp
- OS::Triple0::Services::Securetty
- OS::Triple0::Services::SensuClient
- OS::Triple0::Services::Snmp
- OS::Triple0::Services::Sshd
- OS::Triple0::Services::SwiftRingBuilder
- OS::Triple0::Services::SwiftStorage
- OS::Triple0::Services::Timezone
- OS::Triple0::Services::TripleoFirewall
- OS::Triple0::Services::TripleoPackages

- name: ObjectProxy
  CountDefault: 1
  description: |
Swift Object proxy node role
  networks:
- InternalApi
- Storage
- StorageMgmt
  disable_upgrade_deployment: True
  ServicesDefault:
- OS::Triple0::Services::AuditD
- OS::Triple0::Services::CACerts
- OS::Triple0::Services::CertmongerUser
- OS::Triple0::Services::Collectd
- OS::Triple0::Services::Docker
- OS::Triple0::Services::FluentdClient
- OS::Triple0::Services::Kernel
- OS::Triple0::Services::MySQLClient
- OS::Triple0::Services::Ntp
- OS::Triple0::Services::Securetty
- OS::Triple0::Services::SensuClient
- OS::Triple0::Services::Snmp
- OS::Triple0::Services::Sshd
- OS::Triple0::Services::SwiftRingBuilder
- OS::Triple0::Services::SwiftProxy

```

```
- OS::TripleO::Services::Timezone
- OS::TripleO::Services::TripleoFirewall
- OS::TripleO::Services::TripleoPackages
```

3.1.5.2. 新規ロールのデプロイ

通常の **openstack deploy** コマンドで、新規ロールを指定して、オーバークラウドをデプロイします。

```
openstack overcloud deploy --templates -r roles_data.yaml -e [...]
```

3.2. 基本的なコンテナ管理

擬似フォルダーは、オブジェクトを格納することができる論理デバイスで、入れ子が可能となっていますので、整理がしやすくなります。たとえば、画像を保管する **Images** フォルダーや、ビデオを保管する **Media** フォルダーなどを作成することができます。

各プロジェクトに 1 つまたは複数のコンテナを作成することができます。また、各コンテナには、1 つまたは複数の擬似フォルダーを作成することができます。

3.2.1. コンテナの作成

1. Dashboard で **プロジェクト > オブジェクトストア > コンテナ** を選択します。
2. **コンテナの作成** をクリックします。
3. **コンテナ名** を指定して、**コンテナアクセス** フィールドで以下のいずれかのオプションを選択します。

タイプ	説明
プライベート	現在のプロジェクトでユーザーに対してアクセスを制限します。
パブリック	パブリックの URL を使用して API アクセスを全員に許可します。ただし、Dashboard では、プロジェクトユーザーには、他のプロジェクトのパブリックコンテナおよびデータは表示されません。

4. **コンテナの作成** をクリックします。

新しいコンテナはデフォルトのストレージポリシーを使用します。複数のストレージポリシーが定義されている場合には (たとえば、デフォルトのポリシーと、Erasure Coding を有効にするポリシーなど)、コマンドラインからデフォルト以外のストレージポリシーを使用するようにコンテナを設定することができます。これには以下を実行してください。

```
# swift post -H "X-Storage-Policy:POLICY" CONTAINERNAME
```

上記の設定で、

- **POLICY** は、コンテナで使用するポリシーの名前またはエイリアスに置き換えます。Erasure Coding を使用するサンプルのポリシーについては、「[Erasure Coding ポリシーの設定](#)」を参照してください。
- **CONTAINERNAME** は、コンテナの名前に置き換えてください。

3.2.2. コンテナ用の擬似フォルダーの作成

1. Dashboard で **プロジェクト > オブジェクトストア > コンテナ** を選択します。
2. 擬似フォルダーを追加するコンテナの名前をクリックします。
3. **擬似フォルダーの作成** をクリックします。
4. **擬似フォルダー名** フィールドに名前を指定し、**作成** をクリックします。

3.2.3. コンテナの削除

1. Dashboard で **プロジェクト > オブジェクトストア > コンテナ** を選択します。
2. **コンテナ** のセクションの一覧を参照して全オブジェクトが削除済みであることを確認します ([「オブジェクトの削除」](#)を参照)。
3. 対象のコンテナの矢印メニューで **コンテナの削除** を選択します。
4. **コンテナの削除** をクリックして、コンテナを削除する操作を確定します。

3.2.4. オブジェクトのアップロード

実際のファイルをアップロードしない場合でも、オブジェクトは (プレースホルダーとして) 作成され、後でファイルをアップロードする際に使用することができます。

1. Dashboard で **プロジェクト > オブジェクトストア > コンテナ** を選択します。
2. アップロードしたオブジェクトの配置先となるコンテナの名前をクリックします。そのコンテナに擬似フォルダーがすでに存在している場合には、擬似フォルダーの名前をクリックすることもできます。
3. ファイルをブラウズして**オブジェクトのアップロード** をクリックします。
4. **オブジェクト名** フィールドに名前を指定します。
 - 擬似フォルダーはスラッシュ (/) の記号を使用して指定することができます (例: **Images/myImage.jpg**)。指定したフォルダーがまだ存在していない場合には、オブジェクトのアップロード時に作成されます。
 - その場所 (オブジェクトがすでに存在している場所) に一意ではない名前は、そのオブジェクトの以前のコンテンツを上書きします。
5. **オブジェクトのアップロード** をクリックします。

3.2.5. オブジェクトのコピー

1. Dashboard で **プロジェクト > オブジェクトストア > コンテナ** を選択します。

- オブジェクトのコンテナまたはフォルダーの名前をクリックします (オブジェクトを表示します)。
- オブジェクトのアップロード** をクリックします。
- コピーするファイルを参照し、矢印メニューで **コピー** を選択します。
- 以下の項目を設定します。

フィールド	説明
宛先コンテナ	新規プロジェクトの宛先コンテナ
パス	宛先コンテナの擬似フォルダー。フォルダーが存在しない場合は、作成されます。
宛先オブジェクト名	新規オブジェクト名。その場所に一意ではない名前を使用した場合 (そのオブジェクトがすでに存在している場合) には、そのオブジェクトの以前のコンテンツが上書きされます。

- オブジェクトのコピー** をクリックします。

3.2.6. オブジェクトの削除

- Dashboard で **プロジェクト > オブジェクトストア > コンテナ** を選択します。
- 一覧を参照して対象のオブジェクトを特定し、矢印メニューで **オブジェクトの削除** を選択します。
- オブジェクトの削除** をクリックして、オブジェクトを削除する操作を確定します。

第4章 ファイル共有

OpenStack Shared File Systems サービス (**openstack-manila**) は、複数のインスタンスで消費可能な共有ファイルシステムを簡単にプロビジョニングするための手段を提供します。以前は、OpenStack のユーザーは、インスタンスにマウントする前に共有ファイルシステムを手動でデプロイする必要がありました。一方で、Shared File Systems サービスでは、事前設定済みのストレージプールからファイル共有を簡単にプロビジョニングして、セキュアにマウントする準備を整えることができます。次に、ニーズを満たすためこのプールを個別に管理、スケーリングすることができます。

4.1. バックエンド

この環境に Red Hat OpenStack Platform をデプロイする際は **director** を使用することを推奨します。**director** を使用することで、Shared File Systems サービス (拡張でバックエンドも含む) など、各サービスが正しく設定されるようにします。**director** には、複数のバックエンド設定が統合されています。

- [NetApp](#)
- [CephFS](#) (テクノロジープレビュー)

サポート対象のバックエンドアプライアンスおよびドライバーの完全な一覧は、「[Red Hat OpenStack Platform におけるコンポーネント、プラグイン、およびドライバーのサポート](#)」を参照してください。

4.2. 共有種別の作成と管理

デフォルト以外のバックエンドでファイル共有を作成する場合は、使用するバックエンドを明示的に指定する必要があります。プロセスがユーザーにとってシームレスになるように、**共有種別**を作成して、バックエンドの **share_backend_name** の値と関連付けます。**SHARETYPE** という名前の共有種別を作成するには、OpenStack の admin として、以下を実行します。

```
# manila type-create --is_public false SHARETYPE DHSS
```

DHSS (または **driver handles share servers**) は、共有種別がファイル共有サーバーを処理するドライバーを使用するかどうかを指定します。これは、バックエンドの定義の **driver_handles_share_servers** の値と一致する必要があります (**true** または **false**)。

--is_public false オプションは、共有種別がパブリックにアクセス可能であってはならないことを示します。共有種別をパブリックにする場合には、このオプションは削除してください。

たとえば、**driver_handles_share_servers=false** で設定されたバックエンドをベースにして **general** という名前の非パブリック共有種別を作成するには、以下のコマンドを実行します。

```
# manila type-create --is_public false general false
```

共有種別の作成後に、バックエンドにマッピングすることができます。これには、以下のコマンドを実行します。

```
# manila type-key SHARETYPE set share_backend_name='SHAREBACKEND'
```

SHAREBACKEND はバックエンドの名前に置き換えます (例: **share_backend_name** の値)。これで、Shared File System サービスは (「[ファイル共有の作成](#)」のように) ファイル共有の作成時に **SHARETYPE** を呼び出すたびに、**SHAREBACKEND** のバックエンドを使用します。たとえば、共有種別 **general** をバックエンド **cdotSingleSVM** にマッピングするには、以下を実行します。

```
# manila type-key general set share_backend_name='cdotSingleSVM'
```

ヒント

OpenStack の管理者は、ユーザーが Dashboard でパブリックの共有種別を作成しないようにすることができます。関連情報は、[「Dashboard を使用したパブリックのファイル共有および共有種別の作成の無効化」](#)を参照してください。

今回のリリースでは、Shared File Systems サービスは、バックエンドとして NetApp ストレージコントローラーのみをサポートしています。NetApp がサポートする [追加スペック](#) に関する情報は、[「Creating and Defining Manila Share Types」](#)を参照してください。

4.3. ファイル共有の作成

ファイル共有を作成するには、Shared File System サービスのホストにログインして、以下のコマンドを実行します。

```
# manila create --share-type SHARETYPE --name SHARENAME PROTO GB
```

上記の設定で、

- **SHARETYPE** は、指定の共有種別に関連する設定を適用します。
- **SHARENAME** は、ファイル共有名に置き換えます。
- **PROTO** は、使用する共有プロトコルに置き換えます。
- **GB** は、ファイル共有のサイズ (GB) に置き換えます。

たとえば、[「共有種別の作成と管理」](#)で、**general** という名前の共有種別を作成し、バックエンド **cdotSingleSVM** にマッピングしました。**cdotSingleSVM** バックエンドに **share-01** という名前の 1 GB NFS 共有を作成するには、以下のコマンドを実行します。

```
# manila create --share-type general --name share-01 nfs 10
```

Property	Value
status	creating
description	None
availability_zone	nova
share_network_id	None
export_locations	[]
share_server_id	None
host	None
snapshot_id	None
is_public	False
id	d760eee8-1d91-48c4-8f9a-ad07072e17a2
size	10
name	share-01
share_type	8245657b-ab9e-4db1-8224-451c32d6b5ea
created_at	2015-09-29T16:27:54.092272
export_location	None
share_proto	NFS

```
| project_id          | a19dc7ec562c4ed48cea58d22eb0d3c7 |
| metadata            | {}                                |
+-----+-----+
```

4.4. ファイル共有とエクスポートの情報の一覧表示

ファイル共有が正常に作成されたことを確認するには、以下のコマンドを実行します。

```
# manila list
+-----+-----+-----+-----+
| ID                  | Name      | ... | Status      ...
+-----+-----+-----+-----+
| d760eee8-1d91-48c4-8f9a-ad07072e17a2 | share-01 | ... | available ...
+-----+-----+-----+-----+
```

manila list コマンドは、共有の **export location** も表示します。

```
+-----+-----+
| Export location                                     ...
+-----+-----+
| 10.70.37.46:/manila-nfs-volume-01/share-d760eee8-1d91-...
+-----+-----+
```

この情報は後ほど、ファイル共有をマウントする際に使用します (「[インスタンスへのファイル共有のマウント](#)」)。

4.5. ファイル共有に対するアクセス権の付与

インスタンスにファイル共有をマウントする前に、インスタンスがファイル共有にアクセスできるようにする必要があります。

```
# manila access-allow SHAREID IDENT IDENTKEY
```

上記の設定で、

- **SHAREID** は、「[ファイル共有の作成](#)」で作成したファイル共有の ID に置き換えます。
- **IDENT** は、Shared File Systems サービスがユーザーまたはインスタンスの認証に使用する必要のあるメソッドに置き換えます。
- **IDENTKEY** は、**IDENT** にどの認証の手法を選択するかにより異なります。
 - **cert**: この手法は、TLS 証明書でインスタンスを認証するのに使用します。
 - **user**: これはユーザーまたはグループ名で認証するのに使用します。
 - **ip**: IP アドレスでインスタンスを認証するのに使用します。

たとえば、インスタンス (IP **10.70.36.85** として識別される) に読み取り/書き込みアクセス権を付与するには、以下のコマンドを実行します。

```
# manila access-allow d760eee8-1d91-48c4-8f9a-ad07072e17a2 ip 10.70.36.85
+-----+-----+
| Property      | Value                                |
+-----+-----+
```

```
+-----+-----+
| share_id      | d760eee8-1d91-48c4-8f9a-ad07072e17a2 |
| deleted       | False                                |
| created_at    | 2015-09-29T16:35:33.862114          |
| updated_at    | None                                |
| access_type   | ip                                  |
| access_to     | 10.70.36.85                          |
| access_level  | rw                                  |
| state        | new                                  |
| deleted_at    | None                                |
| id            | b4e990d7-e9d1-4801-bcbe-a860fc1401d1 |
+-----+-----+
```

ファイル共有へのアクセスには、**b4e990d7-e9d1-4801-bcbe-a860fc1401d1** という独自の ID (**ACCESSID**) が指定されていることに注目してください。

正常にアクセス設定が行われたことを確認するには、以下のコマンドを実行します。

```
# manila access-list d760eee8-1d91-48c4-8f9a-ad07072e17a2
+-----+-----+-----+-----+
| id            | access type | access to  | access level ... |
+-----+-----+-----+-----+
| b4e990d7-e9d1-4801-bcbe-... | ip          | 10.70.36.85 | rw              ... |
+-----+-----+-----+-----+
```

4.6. ファイル共有へのアクセスの取り消し

以前に付与したアクセス権を取り消すには、ファイル共有へのアクセス権を削除する必要があります。

```
# manila access-deny SHAREID ACCESSID
```

たとえば、先ほど「[ファイル共有に対するアクセス権の付与](#)」で付与したアクセス権を取り消すには、以下のコマンドを実行します。

```
# manila access-list d760eee8-1d91-48c4-8f9a-ad07072e17a2
+-----+-----+-----+-----+
| id            | access type | access to  | access level ... |
+-----+-----+-----+-----+
| b4e990d7-e9d1-4801-bcbe-... | ip          | 10.70.36.85 | rw              ... |
+-----+-----+-----+-----+
# manila access-deny d760eee8-1d91-48c4-8f9a-ad07072e17a2 b4e990d7-e9d1-4801-bcbe-a860fc1401d1
```

この時点で、インスタンスはマウントしたファイル共有を使用することができなくなります。

4.7. インスタンスへのファイル共有のマウント

インスタンスを認証するようにファイル共有を設定した後に、その共有をマウントすることができま。たとえば、「[ファイル共有の作成](#)」からのファイル共有を、「[ファイル共有に対するアクセス権の付与](#)」からのインスタンスの /mnt にマウントするには、通常通りにインスタンスにログインしてマウントします。


```
# ssh root@10.70.36.85
# mount -t nfs -o vers=3 10.70.37.46:/manila-nfs-volume-01/share-d760eee8-1d91-48c4-8f9a-ad07072e17a2 /mnt
```

ファイル共有のエクスポート情報を表示する方法については、[「ファイル共有とエクスポートの情報の一覧表示」](#) を参照してください。

4.8. ファイル共有の削除

ファイル共有を削除するには以下のコマンドを実行します。

```
# manila delete SHAREID
```

例:

```
# manila delete d760eee8-1d91-48c4-8f9a-ad07072e17a2
```

4.9. DASHBOARD を使用したパブリックのファイル共有および共有種別の作成の無効化

Dashboard では、ユーザーがファイル共有または共有種別の作成時にチェックボックスで **パブリック** に指定することが可能です。ファイル共有の作成時には **全ユーザーに公開する** チェックボックスでファイル共有をパブリックに設定します。

Create Share

Share Name *

Description

Share Protocol *

Size (GiB) *

↑
↓

Share Type *

Availability Zone

Metadata

☐ Make visible for all ?

共有種別の作成インターフェースには、**パブリック** のチェックボックスがあり、同じ機能を果たします。

Create Share Type

Name *

Driver handles share servers *

Extra specs

☒ Public ?

このチェックボックスを無効にして、ユーザーが作成するパブリックのファイル共有および共有種別の数を制限することができます。そのためには、コントローラーノードにログインして、以下の内容を `/etc/openstack-dashboard/local_settings` に追加します。

```
OPENSTACK_MANILA_FEATURES = {
    'enable_public_shares': False
    'enable_public_share_type_creation': False, }
```

`'enable_public_shares':False` の設定により、**全ユーザーに公開する** のチェックボックスが無効になります。また同様に、`'enable_public_share_type_creation':False` の設定により、共有種別の作成インターフェースの **パブリック** チェックボックスが無効になります。

高可用性の構成で OpenStack をデプロイした場合には、同じ変更を全コントローラーノードに適用してから、任意のコントローラーノードで **httpd** サービスを再起動します。

```
# pcs resource restart httpd-clone
```

そうでない場合には、以下のコマンドを使用して **httpd** サービスを再起動します。

```
# systemctl restart httpd
```

変更を確認するには、ダッシュボードを再読み込みします。対応するインターフェースには、チェックボックスが表示されなくなったはずです。



重要

これらのチェックボックスを無効しても、ユーザーまたは OpenStack の管理者がファイル共有または共有種別を **作成後に** パブリックに設定するのを防ぐことはできません。