Red Hat OpenStack Platform 17.0

セキュリティーおよび強化ガイド

グッドプラクティス、コンプライアンス、およびセキュリティーの強化

Last Updated: 2023-09-30

Red Hat OpenStack Platform 17.0 セキュリティーおよび強化ガイド

グッドプラクティス、コンプライアンス、およびセキュリティーの強化

OpenStack Team
rhos-docs@redhat.com

法律上の通知

概要

このガイドでは、Red Hat OpenStack Platform 環境のセキュリティーを強化するためのグッドプラクティスのアドバイスと概念的な情報を提供します。

目次

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、Red Hat CTO である Chris Wright のメッセージ を参照してください。

RED HAT ドキュメントへのフィードバック (英語のみ)

Red Hat ドキュメントに対するご意見をお聞かせください。ドキュメントの改善点があればお知らせください。

ドキュメントへのダイレクトフィードバック (DDF) 機能の使用 (英語版のみ)

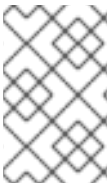
特定の文章、段落、またはコードブロックに対して直接コメントを送付するには、DDF の **Add Feedback** 機能を使用してください。なお、この機能は英語版のドキュメントでのみご利用いただけます。

1. **Multi-page HTML** 形式でドキュメントを表示します。
2. ドキュメントの右上隅に **Feedback** ボタンが表示されていることを確認してください。
3. コメントするテキスト部分をハイライト表示します。
4. **Add Feedback** をクリックします。
5. **Add Feedback** フィールドにコメントを入力します。
6. オプション:ドキュメントチームが問題の詳細を確認する際に使用できるメールアドレスを記入してください。
7. **Submit** をクリックします。

第1章 セキュリティーの概要

Red Hat Openstack Platform (RHOSP) で提供されるツールを使用して、計画や操作においてセキュリティーの優先順位付けを行い、ユーザーのプライバシーとデータのセキュリティーを強化します。セキュリティー標準の実装に失敗すると、ダウンタイムやデータ違反が発生します。ユースケースは、監査プロセスおよびコンプライアンスプロセスを通過する必要のある法律が適用される場合があります。

- Ceph を強化する方法は、データセキュリティーおよび強化ガイド を参照してください。



注記

本章の手順に従って、お使いの環境のセキュリティーを強化します。ただし、これらの推奨事項はセキュリティーやコンプライアンスを保証しません。環境が独自の要件に基づいてセキュリティーを評価する必要があります。

1.1. RED HAT OPENSTACK PLATFORM のセキュリティー

デフォルトでは、Red Hat OpenStack Platform (RHOSP) director は、以下のツールを使用してオーバークラウドを作成し、セキュリティーに対するアクセス制御を行います。

SElinux

SELinux は、各プロセスに必要なアクセス制御を提供して、すべてのアクションに明示的なパーミッションを提供することで、RHOSP のセキュリティー拡張機能を提供します。

Podman

コンテナーツールとしての Podman は、root アクセスを持つプロセスが機能する必要のあるクライアント/サーバーモデルを使用しないため、RHOSP のセキュアなオプションです。

システムアクセスの制限

オーバークラウドのデプロイメント時に director が tripleo-admin に作成する SSH 鍵またはオーバークラウドで作成した SSH キーのいずれかを使用して、オーバークラウドノードにログインすることができます。オーバークラウドノードにログインしてオーバークラウドノードにパスワードで SSH を使用したり、root を使用してオーバークラウドノードにログインしたりすることはできません。

組織のニーズおよび信頼レベルに基づいて、以下に示す新たなセキュリティー機能で director を設定することができます。

- パブリック TLS および TLS-everywhere
- OpenStack Key Manager (barbican) とハードウェアセキュリティーモジュールの統合
- 署名付きイメージおよび暗号化されたボリューム
- ワークフロー実行を使用したパスワードおよび fernet 鍵のローテーション

1.2. RED HAT OPENSTACK PLATFORM 管理者ロールを理解する

ユーザーに **admin** のロールを割り当てると、このユーザーには、任意のプロジェクトの任意のリソースを表示、変更、作成、または削除する権限があります。このユーザーは、公開されている Glance イメージやプロバイダーネットワークなど、プロジェクト間でアクセスできる共有リソースを作成できます。さらに、**admin** ロールを持つユーザーは、ユーザーを作成または削除し、ロールを管理できます。

ユーザーに **admin** ロールを割り当てるプロジェクトは、**openstack** コマンドが実行されるデフォルトのプロジェクトです。たとえば、**development** という名前のプロジェクトの **admin** ユーザーが次のコ

マンドを実行すると、**internal-network** という名前のネットワークが **development** プロジェクトに作成されます。

```
openstack network create internal-network
```

admin ユーザーは、**--project** パラメーターを使用して、任意のプロジェクトに **internal-network** を作成できます。

```
openstack network create internal-network --project testing
```

1.3. RED HAT OPENSTACK PLATFORM におけるセキュリティーゾーンの識別

セキュリティーゾーンは、共通のセキュリティー上の懸念を共有するリソース、アプリケーション、ネットワーク、およびサーバーです。認証認可要件とユーザーを共有するようにセキュリティーゾーンを設計します。クラウドのアーキテクチャー、環境内で許可される信頼レベル、および標準化された要件に基づいて、独自のセキュリティーゾーンを必要な粒度で定義します。ゾーンとその信頼要件は、クラウドインスタンスがパブリック、プライベート、またはハイブリッドであるかによって異なります。

たとえば、Red Hat OpenStack Platform のデフォルトのインストールを以下のゾーンにセグメント化することができます。

表1.1 セキュリティーゾーン

ゾーン	Networks	詳細
公開	external	パブリックゾーンは、外部ネットワーク、パブリック API、およびインスタンスの外部接続用のフローティング IP アドレスをホストします。このゾーンは、管理者のコントロール外のネットワークからのアクセスを可能にし、クラウドインフラストラクチャーの信頼されていない領域となります。
Guest	tenant	ゲストゾーンは、プロジェクトのネットワークをホストします。インスタンスへの無制限のアクセスを許可しているパブリッククラウドやプライベートクラウドのプロバイダーには信用されません。
Storage access	storage、storage_mgmt	ストレージアクセスゾーンは、ストレージの管理、監視とクラスタリング、およびストレージのトラフィック用です。

ゾーン	Networks	詳細
Control	ctlplane、internal_api、ipmi	コントロールゾーンには、アンダークラウド、ホストオペレーティングシステム、サーバーハードウェア、物理ネットワーク、および Red Hat OpenStack Platform director コントロールプレーンも含まれます。

1.4. RED HAT OPENSTACK PLATFORM におけるセキュリティーゾーンの位置確認

以下のコマンドを実行して、Red Hat OpenStack Platform デプロイメントの物理的な設定に関する情報を収集します。

前提条件

- Red Hat OpenStack Platform 環境がインストールされている。
- スタックとしてディレクターにログインしています。

手順

1. **stackrc** 読み込みます。

```
$ source /home/stack/stackrc
```

2. **openstack subnet list** を実行して、割り当てられた IP ネットワークと関連するゾーンを照合します。

```
openstack subnet list -c Name -c Subnet
+--------------------+-----------------+
| Name               | Subnet          |
+--------------------+-----------------+
| ctlplane-subnet    | 192.168.101.0/24 |
| storage_mgmt_subnet | 172.16.105.0/24  |
| tenant_subnet      | 172.16.102.0/24  |
| external_subnet    | 10.94.81.0/24    |
| internal_api_subnet | 172.16.103.0/24  |
| storage_subnet     | 172.16.104.0/24  |
+--------------------+-----------------+
```

3. **openstack server list** を実行して、インフラストラクチャーの物理的なサーバーをリストアップします。

```
openstack server list -c Name -c Networks
+------------------------+------------------------+
| Name                   | Networks               |
+------------------------+------------------------+
```

```
| overcloud-controller-0  | ctlplane=192.168.101.15 |
| overcloud-controller-1  | ctlplane=192.168.101.19 |
| overcloud-controller-2  | ctlplane=192.168.101.14 |
| overcloud-novacompute-0 | ctlplane=192.168.101.18 |
| overcloud-novacompute-2 | ctlplane=192.168.101.17 |
| overcloud-novacompute-1 | ctlplane=192.168.101.11 |
+-------------------------+-------------------------+
```

4. 物理ノードの設定を問い合わせるには、**openstack server list** コマンドの **ctlplane** アドレスを使用します。

```
ssh tripleo-admin@192.168.101.15 ip addr
```

1.5. セキュリティーゾーンの接続

信頼レベルや認証要件が異なる複数のセキュリティーゾーンにまたがるコンポーネントは、慎重に設定する必要があります。これらの接続は、ネットワークアーキテクチャーの弱点になることが多々あります。これらの接続は、接続されるいずれかのゾーンの最高の信頼レベルのセキュリティー要件を満たすように設定されていることを確認してください。多くの場合、攻撃の可能性があるため、接続されたゾーンのセキュリティー制御が主要な懸念事項になります。ゾーンが交わるポイントは、新たな潜在的な攻撃サービスを提供し、攻撃者がデプロイメントのより繊細な部分に攻撃を移行する機会を増やします。

場合によっては、OpenStack 運用者は、統合ポイントのセキュリティーを、それが存在するどのゾーンよりも高い基準で確保することを検討する必要があるかもしれません。上記の API エンドポイントの例では、管理ゾーンが完全に分離されていない場合、敵対者はパブリックゾーンからパブリック API エンドポイントを標的とし、これを足掛かりにして管理ゾーン内の内部または管理者 API を侵害したりアクセスしたりする可能性があります。

OpenStack の設計は、セキュリティーゾーンの分離が難しいようになっています。コアサービスは通常 2 つのゾーンにまたがるので、セキュリティーコントロールを適用する際には、特別な考慮が必要になります。

1.6. 脅威の軽減策

クラウドデプロイメント、パブリック、プライベート、またはハイブリッドの多くのタイプは、一部のセキュリティーの脅威に公開されます。以下のプラクティスを使用すると、セキュリティーの脅威を軽減するのに役立ちます。

- 最小限の特権の原則を適用します。
- 内部および外部のインターフェイスに暗号化を使用します。
- 一元化された ID 管理を使用します。
- Red Hat OpenStack Platform を常に更新します。

Compute サービスは、悪意のあるアクターに DDoS 攻撃やブルートフォース攻撃のツールを提供できます。防止策としては、出口のセキュリティーグループ、トラフィック検査、侵入検知システム、お客様への教育啓発などがあります。インターネットなどのパブリックネットワークからアクセスできる、またはパブリックネットワークにアクセスできる環境では、理想的にアウトバウンドの不正使用を検出し、またそれに対処するためのプロセスとインフラストラクチャーを整備する必要があります。

関連情報

- Ansible を使用した TLS-e の実装
- OpenStack Identity (keystone) と Red Hat Identity Manager (IdM) の統合
- Red Hat OpenStack Platform の最新状態の維持

第2章 セキュリティーの強化

以下の項では、オーバークラウドのセキュリティーを強化するための推奨事項について説明します。

2.1. セキュアな ROOT ユーザーアクセスの使用

オーバークラウドのイメージでは、**root** ユーザーのセキュリティー強化機能が自動的に含まれます。たとえば、デプロイされる各オーバークラウドノードでは、**root** ユーザーへの 直接の SSH アクセスを自動的に無効化されます。ただし、オーバークラウドノードの **root** ユーザーにアクセスすることはできます。各オーバークラウドノードには **tripleo-admin** ユーザーアカウントがあります。このユーザーアカウントにはアンダークラウドのパブリック SSH 鍵が含まれており、アンダークラウドからオーバークラウドノードへのパスワード無しの SSH アクセスが可能です。

前提条件

- Red Hat OpenStack Platform director 環境がインストールされている。
- スタックとしてディレクターにログインしています。

手順

1. アンダークラウドノードで、**tripleo-admin** ユーザーとして SSH 経由でオーバークラウドノードにログインします。
2. **sudo -i** で **root** ユーザーに切り替えます。

2.2. オーバークラウドファイアウォールへのサービスの追加

Red Hat OpenStack Platform をデプロイすると、各コアサービスがデフォルトのファイアウォールルールセットとともに各オーバークラウドノードにデプロイされます。**ExtraFirewallRules** パラメーターを使用して、追加サービスのポートを開くルールを作成したり、サービスを制限するルールを作成したりできます。

各ルール名はそれぞれの **iptables** ルールのコメントになります。各ルール名は、3 桁の接頭辞で始まる点に注意してください。この接頭辞は、Puppet が最終の **iptables** ファイルに記載されているルールを順序付けるのに役立ちます。デフォルトの Red Hat OpenStack Platform ルールでは、000 から 200 までの範囲の接頭辞を使用します。新しいサービスのルールを作成するときは、名前の前に 200 より大きい 3 桁の数字を付けます。

手順

1. 文字列を使用して、**ExtraFireWallRules** パラメーターの下に各ルール名を定義します。ルール名の下に次のパラメーターを使用して、ルールを定義できます。
   - dport::ルールに関連付けられた宛先ポート
   - proto::ルールに関連付けられたプロトコル。デフォルトは **tcp** です。
   - action::ルールに関連付けられたアクションポリシー。デフォルトは **accept** です。
   - source::ルールに関連付けられた送信元の IP アドレス
     次の例は、規則を使用して、カスタムアプリケーション用に追加のポートを開く方法を示しています。

```
cat > ~/templates/firewall.yaml <<EOF
parameter_defaults:
  ExtraFirewallRules:
    '300 allow custom application 1':
      dport: 999
      proto: udp
    '301 allow custom application 2':
      dport: 8081
      proto: tcp
EOF
```

注記

action パラメーターを設定しない場合、結果は **accept** になります。**action** パラメーターは、**drop**、**insert**、または **append** のみに設定できます。

2. ~/**templates**/**firewall.yaml** ファイルを **openstack overcloud deloy** コマンドに含めます。デプロイメントに必要なすべてのテンプレートを含めます。

```
openstack overcloud deploy --templates /
...
-e /home/stack/templates/firewall.yaml /
....
```

2.3. オーバークラウドファイアウォールからのサービスの削除

ルールを使用してサービスを制限できます。ルール名に使用する番号によって、ルールが挿入される **iptables** の場所が決まります。次の手順は、**rabbitmq** サービスを InternalAPI ネットワークに制限する方法を示しています。

手順

1. コントローラーノードで、**rabbitmq** のデフォルトの **iptables** ルールの番号を見つけます。

```
[tripleo-admin@overcloud-controller-2 ~]$ sudo iptables -L | grep rabbitmq
ACCEPT     tcp  --  anywhere             anywhere             multiport dports vtr-
emulator,epmd,amqp,25672,25673:25683 state NEW /* 109 rabbitmq-bundle ipv4 */
```

2. 環境ファイルの **parameter_defaults** で、**ExtraFirewallRules** パラメーターを使用して、**rabbitmq** を InternalApi ネットワークに限定します。ルールには、デフォルトの **rabbitmq** ルール番号または 109 よりも小さい番号が与えられます。

```
cat > ~/templates/firewall.yaml <<EOF
parameter_defaults:
  ExtraFirewallRules:
    '098 allow rabbit from internalapi network':
      dport:
      - 4369
      - 5672
      - 25672
      proto: tcp
      source: 10.0.0.0/24
    '099 drop other rabbit access':
```

```
    dport:
    - 4369
    - 5672
    - 25672
    proto: tcp
    action: drop
EOF
```

注記

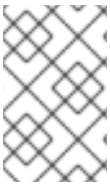
action パラメーターを設定しない場合、結果は **accept** になります。**action** パラメーターは、**drop**、**insert**、または **append** のみに設定できます。

3. **~/templates/firewall.yaml** ファイルを **openstack overcloud deloy** コマンドに含めます。デプロイメントに必要なすべてのテンプレートを含めます。

```
openstack overcloud deploy --templates /
...
-e /home/stack/templates/firewall.yaml /
....
```

2.4. SIMPLE NETWORK MANAGEMENT PROTOCOL (SNMP) 文字列の変更

director は、オーバークラウド向けのデフォルトの読み取り専用 SNMP 設定を提供します。SNMP の文字列を変更して、権限のないユーザーがネットワークデバイスに関する情報を取得するリスクを軽減することを推奨します。



注記

文字列パラメーターを使用して **ExtraConfig** インターフェイスを設定する場合には、heat および Hiera が文字列をブール値と解釈しないように、**'"<VALUE>"'** の構文を使用する必要があります。

オーバークラウドの環境ファイルで **ExtraConfig** フックを使用して、以下の hieradata を設定します。

SNMP の従来のアクセス制御設定

snmp::ro_community

IPv4 の読み取り専用 SNMP コミュニティー文字列。デフォルト値は **public** です。

snmp::ro_community6

IPv6 の読み取り専用 SNMP コミュニティー文字列。デフォルト値は **public** です。

snmp::ro_network

デーモンへの **RO クエリー** が許可されるネットワーク。この値は文字列または配列のいずれかです。デフォルト値は **127.0.0.1** です。

snmp::ro_network6

デーモンへの IPv6 **RO クエリー** が許可されるネットワーク。この値は文字列または配列のいずれかです。デフォルト値は **::1/128** です。

tripleo::profile::base::snmp::snmpd_config

安全弁として **snmpd.conf** ファイルに追加する行の配列。デフォルト値は **[]** です。利用できるすべてのオプションについては、SNMP 設定ファイル に関する Web ページを参照してください。

以下に例を示します。

```
parameter_defaults:
  ExtraConfig:
    snmp::ro_community: mysecurestring
    snmp::ro_community6: myv6securestring
```

これにより、全ノードで、読み取り専用の SNMP コミュニティー文字列が変更されます。

SNMP のビューベースのアクセス制御設定 (VACM)

snmp::com2sec

VACM com2sec マッピングの配列。SECNAME、SOURCE、COMMUNITY を指定する必要があります。

snmp::com2sec6

VACM com2sec6 マッピングの配列。SECNAME、SOURCE、COMMUNITY を指定する必要があります。

以下に例を示します。

```
parameter_defaults:
  ExtraConfig:
    snmp::com2sec: ["notConfigUser default mysecurestring"]
    snmp::com2sec6: ["notConfigUser default myv6securestring"]
```

これにより、全ノードで、読み取り専用の SNMP コミュニティー文字列が変更されます。

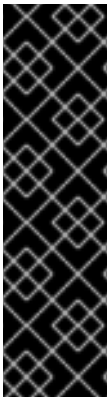
詳細は man ページの **snmpd.conf** を参照してください。

2.5. OPEN VSWITCH ファイアウォールの使用

Red Hat OpenStack Platform director の Open vSwitch (OVS) ファイアウォールドライバーを使用するように、セキュリティーグループを設定することができます。**NeutronOVSFirewallDriver** パラメーターを使用して、使用するファイアウォールドライバーを指定します。

- **iptables_hybrid** - Networking サービス (neutron) が iptables/ハイブリッドベースの実装を使用するように設定します。
- **openvswitch** - Networking サービスが OVS ファイアウォールのフローベースのドライバーを使用するように設定します。

openvswitch ファイアウォールドライバーはパフォーマンスがより高く、ゲストをプロジェクトネットワークに接続するためのインターフェイスとブリッジの数を削減します。



重要

Open vSwitch (OVS) ファイアウォールドライバーによるマルチキャストトラフィックの処理は、iptables ファイアウォールドライバーの場合とは異なります。iptables の場合、デフォルトでは VRRP トラフィックは拒否されます。したがって、VRRP トラフィックがエンドポイントに到達できるようにするには、セキュリティーグループルールで VRRP を有効にする必要があります。OVS の場合、すべてのポートが同じ OpenFlow コンテキストを共有し、ポートごとに個別にマルチキャストトラフィックを処理することはできません。セキュリティーグループはすべてのポート (ルーター上のポートなど) には適用されないため、OVS は RFC 4541 の定義に従って **NORMAL** アクションを使用してマルチキャストトラフィックを全ポートに転送します。

注記

iptables_hybrid オプションは、OVS-DPDK との互換性はありません。**openvswitch** オプションには、OVS ハードウェアオフロードとの互換性はありません。

network-environment.yaml ファイルで **NeutronOVSFirewallDriver** パラメーターを設定します。

```
NeutronOVSFirewallDriver: openvswitch
```

- **NeutronOVSFirewallDriver**:セキュリティーグループを実装する時に使用するファイアウォールドライバーの名前を設定します。設定可能な値は、お使いのシステム設定により異なります。たとえば、**noop**、**openvswitch**、および **iptables_hybrid** です。デフォルト値である空の文字列を指定すると、サポートされている設定となります。

第3章 RHOSP 環境のドキュメント化

システムのコンポーネント、ネットワーク、サービス、およびソフトウェアを文書化することは、セキュリティー上の懸念事項、攻撃ベクトル、およびセキュリティーゾーンの橋渡しとなりうるポイントを特定する上で重要です。Red Hat OpenStack Platform (RHOSP) デプロイメントに関する文書には、以下の情報が含まれている必要があります。

- RHOSP の本番環境、開発環境、テスト環境におけるシステムコンポーネント、ネットワーク、サービス、ソフトウェアの説明。
- 仮想マシンや仮想ディスクボリュームなどの一時リソースのインベントリー。

3.1. システムロールの文書化

Red Hat OpenStack Platform (RHOSP) デプロイメントの各ノードは、クラウドのインフラストラクチャーに貢献したり、クラウドのリソースを提供したりと、特定ロールを果たしています。

インフラストラクチャーに貢献するノードは、メッセージキューイングサービス、ストレージ管理、モニタリング、ネットワークなど、クラウドの運用やプロビジョニングをサポートするために必要なクラウド関連サービスを実行します。インフラストラクチャーロールの例としては、以下のようなものがあります。

- Controller
- Networker
- データベース
- Telemetry

クラウドリソースを提供するノードは、お客様のクラウド上で動作するインスタンスのために、コンピュートやストレージの容量を提供します。リソースロールの例としては、以下のようなものがあります。

- CephStorage
- コンピュート
- ComputeOvsDpdk
- ObjectStorage

お客様の環境で使用されているシステムロールを文書化します。これらのロールは、RHOSP のデプロイに使用されるテンプレート内で識別できます。たとえば、お客様の環境で使用されている各ロールの NIC 設定ファイルがあります。

手順

1. デプロイメント用の既存のテンプレートに、現在使用されているロールを指定するファイルがあるかどうかを確認します。お客様の環境で使用されている各ロールの NIC 設定ファイルがあります。次の例では、RHOSP 環境に **ComputeHCI** ロール、**Compute** ロール、および **Controller** ロールが含まれています。

```
$ cd ~/templates
$ tree
.
```

```
├── environments
│   └── network-environment.yaml
├── hci.yaml
├── network
│   └── config
│       └── multiple-nics
│           ├── computehci.yaml
│           ├── compute.yaml
│           └── controller.yaml
├── network_data.yaml
├── plan-environment.yaml
└── roles_data_hci.yaml
```

2. RHOSP 環境の各ロールは、相互に関連する多くのサービスを実行します。各ロールが使用するサービスを記録するには、**roles** ファイルを調べる必要があります。

   a. テンプレート用に **roles** ファイルが生成されている場合は、~/**templates** ディレクトリーにあります。

   ```
   $ cd ~/templates
   $ find . -name *role*
   > ./templates/roles_data_hci.yaml
   ```

   b. お使いのテンプレートに **roles** ファイルが生成されていない場合は、現在使用しているロールに生成し、ドキュメント作成のために検査することができます。

   ```
   $ openstack overcloud roles generate \
   > --roles-path /usr/share/openstack-tripleo-heat-templates/roles \
   > -o roles_data.yaml Controller Compute
   ```

3.2. ハードウェアインベントリーの作成

イントロスペクションで収集されたデータを表示することで、Red Hat OpenStack Platform のデプロイメントに関するハードウェア情報を取得できます。Introspection では、CPU、メモリー、ディスクなどのハードウェア情報をノードから収集します。

前提条件

- Red Hat OpenStack Platform director 環境がインストールされている。
- Red Hat OpenStack Platform デプロイメントのノードをイントロスペクトしました。
- スタックとしてディレクターにログインしています。

手順

1. アンダークラウドから、**stackrc** ファイルのソースを作成します。

   ```
   $ source ~/stackrc
   ```

2. 環境内のノードをリスト表示します。

   ```
   $ openstack baremetal node list -c Name
   +--------------+
   ```

```
| Name         |
+--------------+
| controller-0 |
| controller-1 |
| controller-2 |
| compute-0    |
| compute-1    |
| compute-2    |
+--------------+
```

3. 情報を収集する各ベアメタルノードについて、以下のコマンドを実行してイントロスペクションデータを取得します。

```
$ openstack baremetal introspection data save <node> | jq
```

<node> を手順 1 で取得したリストのノードの名前に置き換えます。

4. オプション:特定の種類のハードウェアに出力を限定するには、インベントリーキーのリストを取得し、特定のキーのイントロスペクションデータを表示することができます。

   a. イントロスペクションデータからトップレベルのキーのリストを得るために、次のコマンドを実行します。

```
$ openstack baremetal introspection data save controller-0 | jq '.inventory | keys'

[
  "bmc_address",
  "bmc_v6address",
  "boot",
  "cpu",
  "disks",
  "hostname",
  "interfaces",
  "memory",
  "system_vendor"
]
```

   b. **disks** などのキーを選択し、次のように実行すると、詳細な情報が得られます。

```
$ openstack baremetal introspection data save controller-1 | jq '.inventory.disks'
[
  {
    "name": "/dev/sda",
    "model": "QEMU HARDDISK",
    "size": 85899345920,
    "rotational": true,
    "wwn": null,
    "serial": "QM00001",
    "vendor": "ATA",
    "wwn_with_extension": null,
    "wwn_vendor_extension": null,
    "hctl": "0:0:0:0",
    "by_path": "/dev/disk/by-path/pci-0000:00:01.1-ata-1"
  }
]
```

3.3. ソフトウェアインベントリーの作成

Red Hat OpenStack Platform (RHOSP) インフラストラクチャーにデプロイしたノードで使用されているソフトウェアコンポーネントを文書化します。システムデータベース、RHOSP ソフトウェアサービス、およびロードバランサー、DNS、または DHCP サービスなどのサポートコンポーネントは、ライブラリー、アプリケーション、またはソフトウェアのクラスにおける漏洩や脆弱性の影響を評価する際に重要です。

- Red Hat OpenStack Platform 環境がインストールされている。
- スタックとしてディレクターにログインしています。

手順

1. 悪意ある活動の対象となりうるシステムやサービスのエントリーポイントを確実に把握します。アンダークラウド上で以下のコマンドを実行します。

```
$ cat /etc/hosts
$ source stackrc ; openstack endpoint list
$ source overcloudrc ; openstack endpoint list
```

2. RHOSP はコンテナー化されたサービスにデプロイされているため、オーバークラウドノード上で稼働しているコンテナーを確認することで、そのノード上のソフトウェアコンポーネントを確認することができます。**ssh** を使用してオーバークラウドノードに接続し、実行中のコンテナーをリスト表示します。たとえば、**compute-0** のオーバークラウドサービスを表示するには、以下のようなコマンドを実行します。

```
$ ssh tripleo-admin@compute-0 podman ps
```

第4章 アイデンティティーおよびアクセス管理

Identity サービス (keystone) は、Red Hat OpenStack Platform 環境でクラウドユーザーの認証と認可を提供します。エンドユーザーの直接認証に Identity サービスを使用するか、セキュリティー要件を満たすため、または現在の認証インフラストラクチャーに一致するように外部認証方法を使用するように設定できます。

4.1. RED HAT OPENSTACK PLATFORM FERNET トークン

Fernet は、**UUID** トークンプロバイダーに代わるデフォルトのトークンプロバイダーです。デフォルトでは、各 fernet トークンは最大 1 時間有効です。これにより、ユーザーは再認証を必要とせずに一連のタスクを実行できます。

認証後、Identity サービス (keystone) は次のことを行います。

- fernet トークンと呼ばれる暗号化されたベアラートークンを発行します。このトークンは ID を表します。
- ロールに基づいて操作を実行する権限を付与します。

関連情報

- オーバークラウドでの暗号化に Fernet キーの使用

4.2. OPENSTACK IDENTITY サービスエンティティー

Red Hat OpenStack Identity サービス (keystone) は、以下のエンティティーを認識します。

ユーザー

OpenStack Identity サービス (keystone) ユーザーは、認証の最小単位です。ユーザーを認証するには、プロジェクトでロールを割り当てる必要があります。

グループ

OpenStack Identity サービスグループは、ユーザーの論理グループです。グループでは、特定のロールのプロジェクトにアクセスできるようにします。ユーザーではなくグループを管理すると、ロールの管理が簡素化されます。

ロール

OpenStack Identity サービスロールは、それらのロールが割り当てられたユーザーまたはグループがアクセスできる OpenStack API を定義します。

プロジェクト

OpenStack Identity サービスプロジェクトは、ユーザーの分離グループで、物理リソースの共有クォータとそれらの物理リソースから構築された仮想インフラストラクチャーに対する共通のアクセス権が割り当てられます。

ドメイン

OpenStack Identity サービスドメインは、プロジェクト、ユーザー、およびグループの高レベルのセキュリティー境界です。OpenStack Identity ドメインを使用して、keystone ベースのすべての ID コンポーネントを集中管理できます。Red Hat OpenStack Platform は、複数のドメインをサポートしています。別々の認証バックエンドを使用して、異なるドメインのユーザーを表すことができます。

4.3. キーストーンによる認証

OpenStack Identity サービス (keystone) に必要な認証セキュリティー要件を調整できます。

Red Hat OpenStack Platform (RHOSP) をデプロイすると、サービス用に生成されるデフォルトのパスワードよりも複雑なパスワード要件を指定できます。これが発生すると、サービスは認証できず、デプロイは失敗します。

最初に、パスワードの複雑さの要件なしで RHOSP をデプロイする必要があります。デプロイが完了したら、**KeystonePasswordRegex** パラメーターをテンプレートに追加し、デプロイを再実行します。

環境を強化するには、組織の基準を満たすパスワードの複雑性の要件を実装します。NIST が推奨するパスワードの複雑性の要件については、Publication 88-63B、Appendix A を参照してください。

表4.1 ID サービスの認証パラメーター

パラメーター	説明
KeystoneChangePasswordUponFirstUse	このオプションを有効にすると、ユーザーの作成時や、管理者がパスワードをリセットした場合に、ユーザーによるパスワードの変更が必要となります。
KeystoneDisableUserAccountDaysInactive	ユーザーが認証なしでアカウントを使用し続けることのできる最大日数。この期間が過ぎるとアカウントは非アクティブと見なされて自動的に無効 (ロック状態) になります。
KeystoneLockoutDuration	認証の最大試行回数 (**KeystoneLockoutFailureAttempts** で指定) を超過した場合にユーザーアカウントがロックされる秒数
KeystoneLockoutFailureAttempts	ユーザーの認証失敗がこの最大回数を超えると、**KeystoneLockoutDuration** で指定された秒数の期間ユーザーアカウントがロックされます。
KeystoneMinimumPasswordAge	ユーザーがパスワードを変更できるようになるまで、そのパスワードを使用する必要のある日数。これは、パスワードの履歴を消去して古いパスワードを再利用するためにユーザーがパスワードを直ちに変更するのを防ぎます。
KeystonePasswordExpiresDays	ユーザーにパスワードの変更を要求する前に、パスワードが有効と見なされる日数。
KeystonePasswordRegex	パスワードの強度要件を検証するために使用される正規表現。
KeystonePasswordRegexDescription	人が判読できる言語でパスワードの正規表現を記述してください。

KeystoneUniqueLastPasswordCount	これにより、新たに作成されたパスワードが一意であることを強制するために、履歴に保管する以前のユーザーパスワードのイテレーション数が制御されます。

関連情報

- Identity (keystone) パラメーター

4.3.1. Identity サービスの heat パラメーターを使用した無効なログイン試行阻止

ログイン試行の失敗が繰り返される場合は、ブルートフォース攻撃が試みられている可能性があります。ログインに何度も失敗した場合、Identity Service を使用してアカウントへのアクセスを制限できます。

前提条件

- Red Hat OpenStack Platform director 環境がインストールされている。
- スタックとしてディレクターにログインしています。

手順

1. ユーザーアカウントがロックされる前にユーザーが認証に失敗できる最大回数を設定するには、環境ファイルで **KeystoneLockoutFailureAttempts** および **KeystoneLockoutDuration** heat パラメーターの値を設定します。次の例では、**KeystoneLockoutDuration** が 1 時間に設定されています。

```
parameter_defaults
    KeystoneLockoutDuration: 3600
    KeystoneLockoutFailureAttempts: 3
```

2. 環境ファイルをデプロイスクリプトに追加します。以前にデプロイされた環境でデプロイスクリプトを実行すると、追加のパラメーターで更新されます。

```
openstack overcloud deploy --templates \
...
-e keystone_config.yaml
...
```

4.4. 外部 ID プロバイダーによる認証

外部認証プロバイダー (IdP) を使用して OpenStack サービスプロバイダー (SP) に対して認証することができます。SP は、OpenStack クラウドによって提供されるサービスです。

別の IdP を使用する場合、外部認証のクレデンシャルは、他の OpenStack サービスによって使用されるデータベースからは分離されます。このように分離することで、保存された認証情報が侵害されるリスクが軽減されます。

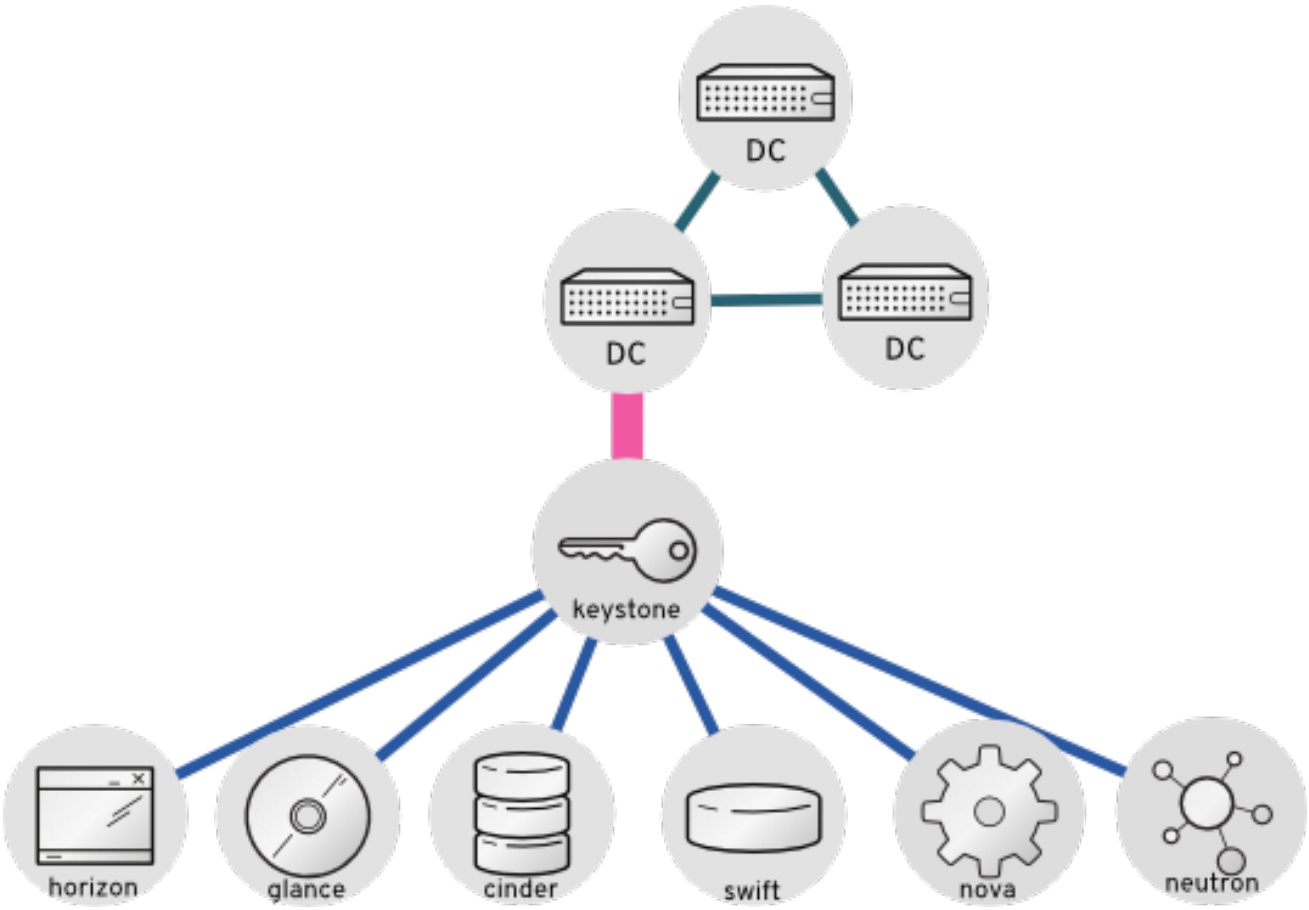
各外部 IdP には、OpenStack Identity サービス (keystone) ドメインへの 1 対 1 のマッピングがあります。Red Hat OpenStack Platform では複数のドメインを共存させることができます。

外部認証は、追加の ID を作成することなく、既存の認証情報を使用して Red Hat OpenStack Platform のリソースにアクセスする方法を提供します。認証情報は、ユーザーの IdP によって維持されます。

ID 管理には、Red Hat Identity Management (IdM) や Microsoft Active Directory Domain Services (AD DS) などの IdP を使用できます。この設定では、OpenStack Identity サービスには LDAP ユーザーデータベースへの読み取り専用アクセス権があります。ユーザーまたはグループのロールに基づく API アクセスの管理は、keystone によって実行されます。ロールは、OpenStack Identity サービスを使用して LDAP アカウントに割り当てられます。

4.4.1. LDAP 統合の仕組み

以下の図では、keystone は暗号化された LDAPS 接続を使用して Active Directory Domain Controller に接続します。ユーザーが horizon にログインすると、keystone は指定したユーザー認証情報を受け取り、Active Directory に渡します。



関連情報

- OpenStack Identity (keystone) と Active Directory の統合
- OpenStack Identity (keystone) と Red Hat Identity Manager (IdM) の統合
- ドメイン固有の LDAP バックエンドを使用する director の設定

第5章 TLS と PKI を使用して RED HAT OPENSTACK デプロイメントを保護する

Red Hat OpenStack Platform は、保護可能な機密データを処理する多くのネットワークとエンドポイントで設定されています。Transport Layer Security (TLS) を使用する場合は、対称キー暗号化でトラフィックを保護します。キーと暗号は TLS ハンドシェイクでネゴシエートされます。これには、認証局 (CA) と呼ばれる仲介者の共有信頼によるサーバーの ID の検証が必要です。

Public Key Infrastructure (PKI) は、認証局を通じてエンティティーを検証するためのフレームワークです。

5.1. 公開鍵基盤 (PKI) のコンポーネント

PKI のコアコンポーネントを次の表に示します。

表5.1 主要な用語

用語	定義
エンドエンティティー	デジタル証明書を使用して自身を検証するユーザー、プロセス、またはシステム。
認証局 (CA)	CA は、エンドエンティティーと、エンドエンティティーを検証する依拠当事者の両方によって信頼されるエンティティーです。
依拠当事者	依拠当事者は、エンドエンティティーの検証としてデジタル証明書を受け取り、デジタル証明書を検証する機能を備えています。
デジタル証明書	署名付き公開鍵証明書には、検証可能なエンティティーと公開鍵があり、CA によって発行されます。CA が証明書に署名すると、秘密鍵で暗号化された証明書からメッセージダイジェストが作成されます。CA に関連付けられた公開鍵を使用して、署名を検証できます。X.509 標準は、証明書の定義に使用されます。
Registration Authority (RA)	RA は、CA によって証明書が発行される前に、エンドエンティティーの認証などの管理機能を実行できるオプションの専用機関です。RA がない場合、CA はエンドエンティティーを認証します。
証明書失効リスト (CRL)	CRL は、失効した証明書のシリアル番号のリストです。シリアル番号が取り消された証明書を提示するエンドエンティティーは、PKI モデルでは信頼されません。
CRL 発行者	CA が証明書失効リストの公開を委譲するオプションのシステム。

用語	定義
証明書リポジトリー	エンドエンティティー証明書と証明書失効リストが格納され、クエリーが実行される場所。

5.2. 認証局の要件と推奨事項

公開されている Red Hat OpenStack Platform ダッシュボードまたは公開されている API については、広く認知されている認証局 (CA) によって署名された証明書を取得する必要があります。

TLS で保護する各エンドポイントに DNS ドメインまたはサブドメインを付与する必要があります。指定したドメインは、CA が発行する証明書を作成するために使用されます。CA がエンドポイントを検証できるように、顧客は DNS 名を使用してダッシュボードまたは API にアクセスします。

Red Hat は、内部トラフィックを保護するために、別の内部管理 CA を使用することを推奨します。これにより、クラウドデプロイヤーは秘密鍵インフラストラクチャー (PKI) 実装の制御を維持でき、内部システムの証明書の要求、署名、デプロイが容易になります。

オーバークラウドのエンドポイントで SSL/TLS を有効にすることができます。どこでも TLS (TLS-e) を設定するために必要な証明書の数が多いため、director は Red Hat Identity Management (IdM) サーバーと統合して認証局として機能し、オーバークラウド証明書を管理します。TLS-e の設定の詳細については、Ansible を使用した TLS-e の実装 を参照してください。

OpenStack 全コンポーネントの TLS サポートのステータスを確認するには、TLS Enablement status matrix を参照してください。

独自の認証局で SSL 証明書を使用する場合は、オーバークラウドのパブリックエンドポイントでの SSL/TLS の有効化 を参照してください。

注記

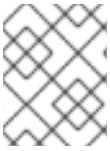
これにより、パブリックにアクセス可能なエンドポイントのみで SSL/TLS を使用する Red Hat OpenStack Platform が設定されます。

5.3. 環境内の TLS バージョンの識別

重要

TLS バージョン 1.0 は、Red Hat OpenStack Platform では非推奨です。さらに、NIST 承認には少なくとも TLS 1.2 を使用する必要があります。詳細は、Transport Layer Security (TLS) 実装の選択、設定、および使用に関するガイドライン を参照してください。

cipherscan を使用して、デプロイで提示されている TLS のバージョンを確認できます。暗号スキャンは、https://github.com/mozilla/cipherscan からクローンできます。この出力例は、**horizon** から受信した結果を示しています。



注記

実稼働以外のシステムから **cipherscan** を実行すると、初回実行時に追加の依存関係をインストールする場合があるためです。

手順

- ダッシュボードサービスのアクセス可能な **URL** に対して暗号スキャンを実行します。

```
$ ./cipherscan https://openstack.lab.local
...............................
Target: openstack.lab.local:443

prio  ciphersuite                protocols  pfs                curves
1     ECDHE-RSA-AES128-GCM-SHA256  TLSv1.2    ECDH,P-256,256bits  prime256v1
2     ECDHE-RSA-AES256-GCM-SHA384  TLSv1.2    ECDH,P-256,256bits  prime256v1
3     DHE-RSA-AES128-GCM-SHA256    TLSv1.2    DH,1024bits         None
4     DHE-RSA-AES256-GCM-SHA384    TLSv1.2    DH,1024bits         None
5     ECDHE-RSA-AES128-SHA256      TLSv1.2    ECDH,P-256,256bits  prime256v1
6     ECDHE-RSA-AES256-SHA384      TLSv1.2    ECDH,P-256,256bits  prime256v1
7     ECDHE-RSA-AES128-SHA         TLSv1.2    ECDH,P-256,256bits  prime256v1
8     ECDHE-RSA-AES256-SHA         TLSv1.2    ECDH,P-256,256bits  prime256v1
9     DHE-RSA-AES128-SHA256        TLSv1.2    DH,1024bits         None
10    DHE-RSA-AES128-SHA           TLSv1.2    DH,1024bits         None
11    DHE-RSA-AES256-SHA256        TLSv1.2    DH,1024bits         None
12    DHE-RSA-AES256-SHA           TLSv1.2    DH,1024bits         None
13    ECDHE-RSA-DES-CBC3-SHA       TLSv1.2    ECDH,P-256,256bits  prime256v1
14    EDH-RSA-DES-CBC3-SHA         TLSv1.2    DH,1024bits         None
15    AES128-GCM-SHA256            TLSv1.2    None                None
16    AES256-GCM-SHA384            TLSv1.2    None                None
17    AES128-SHA256                TLSv1.2    None                None
18    AES256-SHA256                TLSv1.2    None                None
19    AES128-SHA                   TLSv1.2    None                None
20    AES256-SHA                   TLSv1.2    None                None
21    DES-CBC3-SHA                 TLSv1.2    None                None

Certificate: trusted, 2048 bits, sha256WithRSAEncryption signature
TLS ticket lifetime hint: None
NPN protocols: None
OCSP stapling: not supported
Cipher ordering: server
Curves ordering: server - fallback: no
Server supports secure renegotiation
Server supported compression methods: NONE
TLS Tolerance: yes

Intolerance to:
 SSL 3.254          : absent
 TLS 1.0            : PRESENT
 TLS 1.1            : PRESENT
 TLS 1.2            : absent
 TLS 1.3            : absent
 TLS 1.4            : absent
```

サーバーのスキャン時に、Cipherscan インターバリーが、ネゴシエートする最大の TLS バージョンである特定の TLS バージョンのサポートをアドバータイズします。ターゲットサーバーが TLS プロトコ

ルを正しく従うと、相互に対応している最新バージョンで応答します。これは、最初にアドバータイズされた Cipherscan よりも低くなる可能性があります。サーバーが特定のバージョンを使用するクライアントとの接続を確立し続けると、そのプロトコルバージョンに対する耐性があるとみなされません。(指定されたバージョンまたは下位バージョンを使用する) 接続を確立しない場合は、そのバージョンのプロトコルの耐性があることが考慮されます。以下に例を示します。

```
Intolerance to:
 SSL 3.254          : absent
 TLS 1.0            : PRESENT
 TLS 1.1            : PRESENT
 TLS 1.2            : absent
 TLS 1.3            : absent
 TLS 1.4            : absent
```

この出力では、**TLS 1.0** および **TLS 1.1** のイントレランスは **PRESENT** として報告されます。これは、接続を確立できず、その Cipherscan がこれらの TLS バージョンのアドバタイズメントサポート中に接続できなかったことを意味します。そのため、スキャンされたサーバーでプロトコルの (およびそれより低い) バージョンが有効ではないことを確認する必要があります。

5.4. OPENSTACK 向けの IDENTITY MANAGEMENT (IDM) サーバーの推奨事項

Red Hat では IdM サーバーと OpenStack 環境の統合が円滑に進むように、以下の情報を提供しています。

IdM インストール用に Red Hat Enterprise Linux を準備する方法は、Identity Management のインストール を参照してください。

ipa-server-installコマンドを実行して、IdM をインストールおよび設定します。コマンドパラメーターを使用すると対話型プロンプトをスキップできます。IdM サーバーを Red Hat Open Stack Platform 環境と統合できるように、以下の推奨事項を使用してください。

表5.2 パラメーターの推奨事項

オプション	推奨事項
--admin-password	指定した値をメモしておいてください。Red Hat Open Stack Platform を IdM と連携するように設定するときに、このパスワードが必要になります。
--ip-address	指定した値をメモしておいてください。アンダークラウドノードとオーバークラウドノードには、この IP アドレスへのネットワークアクセスが必要です。
--setup-dns	このオプションを使用して、IdM サーバーに統合 DNS サービスをインストールします。アンダークラウドノードとオーバークラウドノードは、ドメイン名の解決に IdM サーバーを使用します。
--auto-forwarders	このオプションを使用して、**/etc/resolv.conf**のアドレスを DNS フォワーダーとして使用します。

オプション	推奨事項
--auto-reverse	このオプションを使用して、IdM サーバーの IP アドレスのリバースレコードとゾーンを解決します。リバースレコードもゾーンも解決できない場合には、IdM はリバースゾーンを作成します。こうすることで IdM のデプロイメントが簡素化されます。
--ntp-server, **--ntp-pool**	これらのオプションの両方またはいずれかを使用して、NTP ソースを設定できます。Id M サーバーと Open Stack 環境の両方で、時間が正しく同期されている必要があります。

Red Hat Open Stack Platform ノードとの通信を有効にするには、IdM に必要なファイアウォールポートを開く必要があります。詳細は、IdM に必要なポートの解放 を参照してください。

関連情報

- Identity Management の設定および管理
- Red Hat Identity Management のドキュメント

5.5. ANSIBLE を使用した TLS-E の実装

新しい **tripleo-ipa** メソッドを使用して、どこでも TLS (TLS-e) と呼ばれるオーバークラウドエンドポイントで SSL/TLS を有効にすることができます。必要な証明書の数が多いため、Red Hat OpenStack Platform は Red Hat Identity Management (IdM) と統合されています。**tripleo-ipa** を使用して TLS-e を設定する場合、IdM が認証局です。

前提条件

- stack ユーザーの作成など、アンダークラウドの設定手順がすべて完了していること。詳細は、director のインストールと使用方法 を参照してください。
- DNS サーバーの IP アドレスは、アンダークラウド上で IdM サーバーの IP アドレスに設定されます。以下のパラメーターのいずれかを **undercloud.conf** ファイルに設定する必要があります。
  - **DEFAULT/undercloud_nameservers**
  - **%SUBNET_SECTION%/dns_nameservers**

手順

次の手順で、Red Hat OpenStack Platform の新規インストール、または TLS-e で設定する既存のデプロイメントに TLS-e を実装します。事前にプロビジョニングされたノードに TLS-e を設定した Red Hat OpenStack Platform をデプロイする場合は、この方式を使用する必要があります。

注記

既存の環境に TLS-e を実装している場合は、**openstack undercloud install** や **openstack overcloud deploy** などのコマンドを実行する必要があります。これらの手順はべき等性を持ち、更新されたテンプレートおよび設定ファイルと一致するように既存のデプロイメント設定を調整するだけです。

1. **/etc/resolv.conf** ファイルを設定します。
   アンダークラウドの **/etc/resolv.conf** に、適切な検索ドメインおよびネームサーバーを設定します。たとえば、デプロイメントドメインが **example.com** で FreeIPA サーバーのドメインが **bigcorp.com** の場合、以下の行を /etc/resolv.conf に追加します。

```
search example.com bigcorp.com
nameserver $IDM_SERVER_IP_ADDR
```

2. 必要なソフトウェアをインストールします。

```
sudo dnf install -y python3-ipalib python3-ipaclient krb5-devel
```

3. ご自分の環境に固有の値で環境変数をエクスポートします。

```
export IPA_DOMAIN=bigcorp.com
export IPA_REALM=BIGCORP.COM
export IPA_ADMIN_USER=$IPA_USER (1)
export IPA_ADMIN_PASSWORD=$IPA_PASSWORD (2)
export IPA_SERVER_HOSTNAME=ipa.bigcorp.com
export UNDERCLOUD_FQDN=undercloud.example.com (3)
export USER=stack
export CLOUD_DOMAIN=example.com
```

(1)(2) IdM のユーザー認証情報は、新しいホストおよびサービスを追加できる管理ユーザーでなければなりません。

(3) **UNDERCLOUD_FQDN** パラメーターの値は、**/etc/hosts** の最初のホスト名から IP へのマッピングと一致します。

4. アンダークラウドで **undercloud-ipa-install.yaml** Ansible Playbook を実行します。

```
ansible-playbook \
--ssh-extra-args "-o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null" \
/usr/share/ansible/tripleo-playbooks/undercloud-ipa-install.yaml
```

5. undercloud.conf に以下のパラメーターを追加します。

```
undercloud_nameservers = $IDM_SERVER_IP_ADDR
overcloud_domain_name = example.com
```

6. [オプション] IPA レルムが IPA ドメインと一致しない場合は、**certmonger_krb_realm** パラメーターの値を設定します。

   a. **/home/stack/hiera_override.yaml** で **certmonger_krb_realm** の値を設定します。

```
parameter_defaults:
  certmonger_krb_realm = EXAMPLE.COMPANY.COM
```

b. **undercloud.conf** で **custom_env_files** パラメーターの値を **/home/stack/hiera_override.yaml** に設定します。

```
custom_env_files = /home/stack/hiera_override.yaml
```

7. アンダークラウドをデプロイします。

```
openstack undercloud install
```

検証

以下の手順を実施して、アンダークラウドが正しく登録されたことを確認します。

1. IdM のホストをリスト表示します。

```
$ kinit admin
$ ipa host-find
```

2. アンダークラウドに **/etc/novajoin/krb5.keytab** が存在することを確認します。

```
ls /etc/novajoin/krb5.keytab
```

注記

novajoin というディレクトリー名は、従来の方式に対応させる目的でのみ使用されています。

オーバークラウドでの TLS-e の設定

TLS everywhere (TLS-e) を設定したオーバークラウドをデプロイする場合、アンダークラウドおよびオーバークラウドの IP アドレスは自動的に IdM に登録されます。

1. オーバークラウドをデプロイする前に、以下のような内容で YAML ファイル **tls-parameters.yaml** を作成します。お使いの環境に固有の値を選択してください。

```
parameter_defaults:
    DnsSearchDomains: ["example.com"]
    CloudDomain: example.com
    CloudName: overcloud.example.com
    CloudNameInternal: overcloud.internalapi.example.com
    CloudNameStorage: overcloud.storage.example.com
    CloudNameStorageManagement: overcloud.storagemgmt.example.com
    CloudNameCtlplane: overcloud.ctlplane.example.com
    IdMServer: freeipa-0.redhat.local
    IdMDomain: redhat.local
    IdMInstallClientPackages: False

resource_registry:
      OS::TripleO::Services::IpaClient: /usr/share/openstack-tripleo-heat-
templates/deployment/ipa/ipaservices-baremetal-ansible.yaml
```

- **OS::TripleO::Services::IpaClient** パラメーターに示す値は、**enable-internal-tls.yaml** ファイルのデフォルト設定を上書きします。**openstack overcloud deploy** コマンドで、**enable-internal-tls.yaml** の後に **tls-parameters.yaml** ファイルを指定するようにします。

- TLS-e の実装に使用するパラメーターの詳細は、tripleo-ipa のパラメーター を参照してください。

2. オーバークラウドをデプロイする。デプロイメントコマンドに tls-parameters.yaml を追加する必要があります。

```
DEFAULT_TEMPLATES=/usr/share/openstack-tripleo-heat-templates/
CUSTOM_TEMPLATES=/home/stack/templates

openstack overcloud deploy \
-e ${DEFAULT_TEMPLATES}/environments/ssl/tls-everywhere-endpoints-dns.yaml \
-e ${DEFAULT_TEMPLATES}/environments/services/haproxy-public-tls-certmonger.yaml \
-e ${DEFAULT_TEMPLATES}/environments/ssl/enable-internal-tls.yaml \
-e ${CUSTOM_TEMPLATES}/tls-parameters.yaml \
...
```

3. keystone にエンドポイントリストのクエリーを行い、各エンドポイントが HTTPS を使用していることを確認します。

```
openstack endpoint list
```

5.6. TRIPO-IPA のパラメーター

クラウドの完全修飾ドメイン名 (FQDN) を使用して、**tripleo-ipa** に必要なクラウド名とクラウドドメインパラメーターを定義します。たとえば、FQDN が **overcloud.example.com** の場合は、次の値を使用します。

- CloudDomain: example.com
- CloudName: overcloud.example.com
- CloudNameCtlplane: overcloud.ctlplane.example.com
- CloudNameInternal: overcloud.internalapi.example.com
- CloudNameStorage: overcloud.storage.example.com
- CloudNameStorageManagement: overcloud.storagemgmt.example.com

環境の要件に基づいて、次の追加パラメーターを設定します。

CertmongerKerberosRealm

CertmongerKerberosRealm パラメーターを IPA レルムの値に設定します。これは、IPA レルムが IPA ドメインと一致しない場合に必要です。

DnsSearchDomains

DnsSearchDomains パラメーターは、コンマ区切りのリストです。IdM サーバーのドメインがクラウドドメインと異なる場合は、**DnsSearchDomains** パラメーターに IdM サーバーのドメインを含めます。

EnableEtcdInternalTLS

TLS を分散コンピュートノード (DCN) アーキテクチャーにデプロイする場合は、値 **True** を指定して **EnableEtcdInternalTLS** パラメーターを追加する必要があります。

IDMInstallClientPackages

コンピュートノードを事前にプロビジョニングした場合は、**IDMInstallClientPackages** パラメーターの値を **True** に設定します。それ以外の場合は、値を**False**に設定します。

IDMModifyDNS

IDMModifyDNS パラメーターを **false** に設定して、Red Hat Identity Server 上のオーバークラウドノードの自動 IP 登録を無効にします。

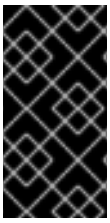
IdmDomain

IdmDomain パラメーターを Red Hat Identity サーバーの FQDN のドメイン部分に設定します。指定した値は、IdM レルムの値としても使用されます。IdM ドメインと IdM レルムが異なる場合は、**CertmongerKerberosRealm** パラメーターを使用してレルムを明示的に設定します。

IdmServer

IdmServer パラメーターを Red Hat Identity サーバーの FQDN に設定します。複製された IdM 環境を使用する場合は、コンマ区切りのリストを使用して複数の値を設定します。IdM レプリカの詳細については、IdM レプリカのインストール を参照してください。

5.7. TLS EVERYWHERE (TLS-E) による MEMCACHED トラフィックの暗号化



重要

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、テスト用途にのみご利用いただく機能です。実稼働環境にはデプロイしないでください。テクノロジープレビュー機能についての詳細は、対象範囲の詳細 を参照してください。

memcached トラフィックを TLS-e で暗号化できるようになりました。この機能は、novajoin と tripleo-ipa の両方で機能します。

1. 以下の内容で **memcached.yaml** という名前の環境ファイルを作成し、memcached の TLS サポートを追加します。

```
parameter_defaults:
  MemcachedTLS: true
  MemcachedPort: 11212
```

2. オーバークラウドのデプロイプロセスに **memcached.yaml** 環境ファイルを追加します。

```
openstack overcloud deploy --templates \
-e /usr/share/openstack-tripleo-heat-templates/environments/ssl/enable-internal-tls.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/ssl/tls-everywhere-endpoints-dns.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/services/haproxy-public-tls-certmonger.yaml \
-e /home/stack/memcached.yaml
...
```

関連情報

- tripleo-ipa を使用した TLSe のデプロイに関する詳細は、Ansible を使用した TLS-e の実装 を参照してください。

5.8. 秘密鍵のサイズの拡大

暗号化されたサービストラフィックの証明書の作成に使用される秘密鍵のサイズを大きくすることで、セキュリティーを向上させることができます。デフォルトの RHOSP 秘密鍵のサイズである 2048 ビットは、米国国立標準技術研究所 (NIST) が推奨する最小値と同じです。

- 秘密鍵のサイズをグローバルに変更するには、**CertificateKeySize** パラメーターを使用します。
- **RedisCertificateKeySize** などのサービス固有のパラメーターを使用して、特定の秘密鍵を変更したり、グローバルの **CertificateKeySize** パラメーターをオーバーライドしたりします。

環境 Heat テンプレートでこれらのパラメーターを使用し、そのテンプレートをオーバークラウドのデプロイメントコマンドに含めます。オーバークラウドをすでにデプロイしている場合は、最初に使用したものと同じテンプレートを使用して同じ **openstack overcloud deploy** コマンドを再実行して、新しいパラメーターを追加し、変更を有効にする必要があります。

次の例では、秘密鍵のグローバル値は **4096** です。**RedisCertificateKeySize** がグローバルパラメーターをオーバーライドするため、**redis** の秘密鍵は **2048** になります。

例

```
parameter_defaults:
  CertificateKeySize: '4096'
  RedisCertificateKeySize: '2048'
```

5.9. RED HAT OPENSTACK PLATFORM の IDM サーバーをレプリカに置き換える

既存の IPA サーバーをレプリカに置き換える場合は、必要なパラメーターを更新する必要があります。そうしないと、クラスターの設定を更新する際、オーバークラウドのデプロイメントが失敗します。

手順

1. 各オーバークラウドノードで **/etc/ipa/default.conf** 設定ファイルを編集して、**server** および **xmlrpc_uri** パラメーターに IdM サーバーの完全修飾ドメイン名 (FQDN) が含まれるようにします。

```
#File modified by ipa-client-install

[global]
basedn = dc=redhat,dc=local
realm = REDHAT.LOCAL
domain = redhat.local
server = freeipa-0.redhat.local
host = undercloud-0.redhat.local
xmlrpc_uri = https://freeipa-0.redhat.local/ipa/xml
enable_ra = True
```

2. アンダークラウドで **/home/stack/templates/tls-parameters.yaml** 環境ファイルを編集し、**IPA_SERVER_HOSTNAME** パラメーターが **/etc/ipa/default.conf** の **xmlrpc_uri** および

server パラメーターに示されている FQDN に一致するようにします。すべてのパラメーターが環境に一致していることを確認します。

```
export IPA_DOMAIN=bigcorp.com
export IPA_REALM=BIGCORP.COM
export IPA_ADMIN_USER=$IPA_USER
export IPA_ADMIN_PASSWORD=$IPA_PASSWORD
export IPA_SERVER_HOSTNAME=ipa.bigcorp.com
export UNDERCLOUD_FQDN=undercloud.example.com
export USER=stack
export CLOUD_DOMAIN=example.com
```

第6章 カスタム SSL/TLS 証明書の設定

アンダークラウドがパブリックエンドポイントの通信に SSL/TLS を使用するように手動で設定できます。SSL/TLS を使用してアンダークラウドエンドポイントを手動で設定すると、概念実証としてセキュアなエンドポイントが作成されます。Red Hat は、認証局ソリューションを使用することを推奨します。

認証局 (CA) ソリューションを使用すると、証明書の更新、証明書失効リスト (CRL)、業界で受け入れられている暗号化など、運用に対応したソリューションが得られます。Red Hat Identity Manager (IdM) を CA として使用する方法は、Ansible を使用した TLS-e の実装 を参照してください。

独自の認証局で発行した SSL 証明書を使用する場合は、以下の設定手順を実施する必要があります。

6.1. 署名ホストの初期化

署名ホストとは、認証局を使用して新規証明書を生成し署名するホストです。選択した署名ホスト上で SSL 証明書を作成したことがない場合には、ホストを初期化して新規証明書に署名できるようにする必要がある可能性があります。

手順

1. すべての署名済み証明書の記録は、**/etc/pki/CA/index.txt** ファイルに含まれます。ファイルシステムパスと **index.txt** ファイルが存在することを確認します。

```
$ sudo mkdir -p /etc/pki/CA
$ sudo touch /etc/pki/CA/index.txt
```

2. **/etc/pki/CA/serial** ファイルは、次に署名する証明書に使用する次のシリアル番号を特定します。このファイルが存在しているかどうかを確認してください。ファイルが存在しない場合には、新規ファイルを作成して新しい開始値を指定します。

```
$ echo '1000' | sudo tee /etc/pki/CA/serial
```

6.2. 認証局の作成

通常、SSL/TLS 証明書の署名には、外部の認証局を使用します。場合によっては、独自の認証局を使用する場合もあります。たとえば、内部のみの認証局を使用するように設定する場合などです。

手順

1. 鍵と証明書のペアを生成して、認証局として機能するようにします。

```
$ openssl genrsa -out ca.key.pem 4096
$ openssl req  -key ca.key.pem -new -x509 -days 7300 -extensions v3_ca -out ca.crt.pem
```

2. **openssl req** コマンドは、認証局に関する特定の情報を要求します。要求されたら、それらの情報を入力してください。これらのコマンドにより、**ca.crt.pem** という名前の認証局ファイルが作成されます。

3. 証明書の場所を **enable-tls.yaml** ファイルの **PublicTLSCAFile** パラメーターの値として設定します。証明書の場所を **PublicTLSCAFile** パラメーターの値として設定する場合、CA 証明書パスが **clouds.yaml** 認証ファイルに追加されていることを確認してください。

```
parameter_defaults:
  PublicTLSCAFile: /etc/pki/ca-trust/source/anchors/cacert.pem
```

6.3. クライアントへの認証局の追加

SSL/TLS を使用して通信する外部クライアントについては、Red Hat OpenStack Platform 環境にアクセスする必要のある各クライアントに認証局ファイルをコピーします。

手順

1. 認証局をクライアントシステムにコピーします。

```
$ sudo cp ca.crt.pem /etc/pki/ca-trust/source/anchors/
```

2. 各クライアントに認証局ファイルをコピーしたら、それぞれのクライアントで以下のコマンドを実行し、証明書を認証局のトラストバンドルに追加します。

```
$ sudo update-ca-trust extract
```

6.4. SSL/TLS 鍵の作成

OpenStack 環境で SSL/TLS を有効にするには、証明書を生成するための SSL/TLS 鍵が必要です。

手順

1. 以下のコマンドを実行し、SSL/TLS 鍵 (**server.key.pem**) を生成します。

```
$ openssl genrsa -out server.key.pem 2048
```

6.5. SSL/TLS 証明書署名要求の作成

証明書署名要求を作成するには、以下の手順を実施します。

手順

1. デフォルトの OpenSSL 設定ファイルをコピーします。

```
$ cp /etc/pki/tls/openssl.cnf .
```

2. 新しい **openssl.cnf** ファイルを編集して、director に使用する SSL パラメーターを設定します。変更するパラメーターの種別には以下のような例が含まれます。

```
[req]
distinguished_name = req_distinguished_name
req_extensions = v3_req

[req_distinguished_name]
countryName = Country Name (2 letter code)
countryName_default = AU
stateOrProvinceName = State or Province Name (full name)
stateOrProvinceName_default = Queensland
```

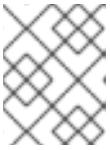
```
localityName = Locality Name (eg, city)
localityName_default = Brisbane
organizationalUnitName = Organizational Unit Name (eg, section)
organizationalUnitName_default = Red Hat
commonName = Common Name
commonName_default = 192.168.0.1
commonName_max = 64

[ v3_req ]
# Extensions to add to a certificate request
basicConstraints = CA:FALSE
keyUsage = nonRepudiation, digitalSignature, keyEncipherment
subjectAltName = @alt_names

[alt_names]
IP.1 = 192.168.0.1
DNS.1 = instack.localdomain
DNS.2 = vip.localdomain
DNS.3 = 192.168.0.1
```

commonName_default を、以下のエントリーのいずれかに設定します。

- IP アドレスを使用して SSL/TLS 経由で director にアクセスする場合には、**undercloud.conf** ファイルの **undercloud_public_host** パラメーターを使用します。
- 完全修飾ドメイン名を使用して SSL/TLS 経由で director にアクセスする場合には、ドメイン名を使用します。

alt_names セクションを編集して、以下のエントリーを追加します。

- **IP**: SSL 経由で director にアクセスするためにクライアントが使用する IP アドレスリスト
- **DNS**: SSL 経由で director にアクセスするためにクライアントが使用するドメイン名リスト。**alt_names** セクションの最後に DNS エントリーとしてパブリック API の IP アドレスも追加します。



注記

openssl.cnf に関する詳しい情報については、**man openssl.cnf** コマンドを実行してください。

3. 以下のコマンドを実行し、証明書署名要求 (**server.csr.pem**) を生成します。

```
$ openssl req -config openssl.cnf -key server.key.pem -new -out server.csr.pem
```

-key オプションを使用して、OpenStack SSL/TLS 鍵を指定するようにしてください。

このコマンドにより、証明書署名要求として **server.csr.pem** ファイルが生成されます。このファイルを使用して OpenStack SSL/TLS 証明書を作成します。

6.6. SSL/TLS 証明書の作成

OpenStack 環境の SSL/TLS 証明書を生成するには、以下のファイルが必要です。

openssl.cnf

v3 拡張機能を指定するカスタム設定ファイル

server.csr.pem

証明書を生成して認証局を使用して署名するための証明書署名要求

ca.crt.pem

証明書への署名を行う認証局

ca.key.pem

認証局の秘密鍵

手順

1. **newcerts** ディレクトリーが存在しない場合は作成します。

```
sudo mkdir -p /etc/pki/CA/newcerts
```

2. 以下のコマンドを実行し、アンダークラウドまたはオーバークラウドの証明書を作成します。

```
$ sudo openssl ca -config openssl.cnf -extensions v3_req -days 3650 -in server.csr.pem -out
server.crt.pem -cert ca.crt.pem -keyfile ca.key.pem
```

コマンドは、以下のオプションを使用します。

-config

カスタム設定ファイルを使用します (ここでは、v3 拡張機能を指定した **openssl.cnf** ファイル)。

-extensions v3_req

v3 拡張機能を有効にします。

-days

証明書の有効期限が切れるまでの日数を定義します。

-in

証明書署名要求

-out

作成される署名済み証明書

-cert

認証局ファイル

-keyfile

認証局の秘密鍵

上記のコマンドにより、**server.crt.pem** という名前の新規証明書が作成されます。OpenStack SSL/TLS 鍵と共にこの証明書を使用します。

6.7. アンダークラウドへの証明書の追加

OpenStack SSL/TLS 証明書をアンダークラウドのトラストバンドルに追加するには、以下の手順を実施します。

手順

1. 以下のコマンドを実行して、証明書と鍵を統合します。

```
$ cat server.crt.pem server.key.pem > undercloud.pem
```

このコマンドにより、**undercloud.pem** ファイルが作成されます。

2. **undercloud.pem** ファイルを **/etc/pki** ディレクトリー内の場所にコピーし、HAProxy が読み取ることができるように必要な SELinux コンテキストを設定します。

```
$ sudo mkdir /etc/pki/undercloud-certs
$ sudo cp ~/undercloud.pem /etc/pki/undercloud-certs/.
$ sudo semanage fcontext -a -t etc_t "/etc/pki/undercloud-certs(/.*)?"
$ sudo restorecon -R /etc/pki/undercloud-certs
```

3. **undercloud.conf** ファイルの **undercloud_service_certificate** オプションに **undercloud.pem** ファイルの場所を追加します。

```
undercloud_service_certificate = /etc/pki/undercloud-certs/undercloud.pem
```

generate_service_certificate および **certificate_generation_ca** パラメーターを設定または有効にしないでください。director は、手動で作成した **undercloud.pem** 証明書を使用する代わりに、これらのパラメーターを使用して証明書を自動的に生成します。

4. アンダークラウド内の別のサービスが認証局にアクセスできるように、証明書に署名した認証局をアンダークラウドの信頼済み認証局のリストに追加します。

```
$ sudo cp ca.crt.pem /etc/pki/ca-trust/source/anchors/
$ sudo update-ca-trust extract
```

認証局がアンダークラウドに追加されたことを確認するには、**openssl** を使用してトラストバンドルを確認します。

```
$ openssl crl2pkcs7 -nocrl -certfile /etc/pki/tls/certs/ca-bundle.crt | openssl pkcs7 -print_certs -text | grep <CN of the CA issuer> -A 10 -B 10
```

<CN of the CA issuer> を CA の発行者の一般名に置き換えます。このコマンドにより、有効期間を含むメインの証明書の詳細が出力されます。

第7章 オーバークラウドのパブリックエンドポイントでの SSL/TLS の有効化

デフォルトでは、オーバークラウドはオーバークラウドサービスに暗号化されていないエンドポイントを使用します。オーバークラウドで SSL/TLS を有効にするには、Red Hat は認証局 (CA) ソリューションを使用することを推奨します。

CA ソリューションを使用すると、証明書の更新、証明書失効リスト (CRL)、業界で受け入れられている暗号化など、運用に対応したソリューションが得られます。Red Hat Identity Manager (IdM) を CA として使用する方法は、Ansible を使用した TLS-e の実装 を参照してください。

次の手動プロセスを使用して、パブリック API エンドポイントに対してのみ SSL/TLS を有効にすることができます。内部 API と管理 API は暗号化されません。CA を使用しない場合は、SSL/TLS 証明書を手動で更新する必要があります。詳細は、SSL/TLS 証明書の手動更新 を参照してください。

前提条件

- パブリック API のエンドポイントを定義するためのネットワーク分離
- **openssl-perl** パッケージがインストールされている。
- SSL/TLS 証明書があります。詳細は、カスタム SSL/TLS 証明書の設定 を参照してください。

7.1. SSL/TLS の有効化

オーバークラウドで SSL/TLS を有効にするには、SSL/TLS 証明書と秘密鍵のパラメーターを含む環境ファイルを作成する必要があります。

手順

1. heat テンプレートコレクションから **enable-tls.yaml** の環境ファイルをコピーします。

```
$ cp -r /usr/share/openstack-tripleo-heat-templates/environments/ssl/enable-tls.yaml
~/templates/.
```

2. このファイルを編集して、下記のパラメーターに以下の変更を加えます。

SSLCertificate

証明書ファイル (**server.crt.pem**) のコンテンツを **SSLCertificate** パラメーターにコピーします。

```
parameter_defaults:
  SSLCertificate: |
    -----BEGIN CERTIFICATE-----
    MIIDgzCCAmugAwIBAgIJAKk46qw6ncJaMA0GCSqGS
    ...
    sFW3S2roS4X0Af/kSSD8mlBBTFTCMBAj6rtLBKLaQ
    -----END CERTIFICATE-----
```

重要

この証明書の内容に新しく追加する行は、すべて同じレベルにインデントする必要があります。

SSLIntermediateCertificate

中間証明書がある場合、中間証明書のコンテンツを **SSLIntermediateCertificate** パラメーターにコピーします。

```
parameter_defaults:
  SSLIntermediateCertificate: |
    -----BEGIN CERTIFICATE-----
    sFW3S2roS4X0Af/kSSD8mlBBTFTCMBAj6rtLBKLaQbIxEpIzrgvpBCwUAMFgxCzAJB
    ...
    MIIDgzCCAmugAwIBAgIJAKk46qw6ncJaMA0GCSqGSIb3DQE
    -----END CERTIFICATE-----
```

重要

この証明書の内容に新しく追加する行は、すべて同じレベルにインデントする必要があります。

SSLKey

秘密鍵 (**server.key.pem**) の内容を **SSLKey** パラメーターにコピーします。

```
parameter_defaults:
  ...
  SSLKey: |
    -----BEGIN RSA PRIVATE KEY-----
    MIIEowIBAAKCAQEAqVw8lnQ9RbeI1EdLN5PJP0IVO
    ...
    ctlKn3rAAdyumi4JDjESAXHIKFjJNOLrBmpQyES4X
    -----END RSA PRIVATE KEY-----
```

重要

この秘密鍵の内容に新しく追加する行は、すべて同じレベルにインデントする必要があります。

7.2. ルート証明書の注入

証明書の署名者がオーバークラウドのイメージにあるデフォルトのトラストストアに含まれない場合には、オーバークラウドのイメージに認証局を注入する必要があります。

手順

1. Heat テンプレートコレクションから **inject-trust-anchor-hiera.yaml** 環境ファイルをコピーします。

```
$ cp -r /usr/share/openstack-tripleo-heat-templates/environments/ssl/inject-trust-anchor-hiera.yaml ~/templates/.
```

このファイルを編集して、下記のパラメーターに以下の変更を加えます。

CAMap

オーバークラウドに注入する各認証局 (CA) の内容をリストにして定義します。オーバークラウドには、アンダークラウドとオーバークラウド両方の証明書を署名するのに使用する CA ファイルが必要です。ルート認証局ファイル (**ca.crt.pem**) の内容をエントリーにコピーします。**CAMap** パラメーターの例を以下に示します。

```
parameter_defaults:
  CAMap:
    ...
    undercloud-ca:
      content: |
        -----BEGIN CERTIFICATE-----
        MIIDITCCAn2gAwIBAgIJAOnPtx2hHEhrMA0GCS
        BAYTAlVTMQswCQYDVQQIDAJOQzEQMA4GA1UEBw
        UmVkIEhhdDELMAkGA1UECwwCUUUxFDASBgNVBA
        -----END CERTIFICATE-----
    overcloud-ca:
      content: |
        -----BEGIN CERTIFICATE-----
        MIIDBzCCAe+gAwIBAgIJAIc75A7FD++DMA0GCS
        BAMMD3d3dy5leGFtcGxlLmNvbTAeFw0xOTAxMz
        Um54yGCARyp3LpkxvyfMXX1DokpS1uKi7s6CkF
        -----END CERTIFICATE-----
```

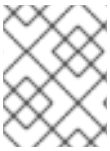
重要

この認証局の内容に新しく追加する行は、すべて同じレベルにインデントする必要があります。

CAMap パラメーターを使用して、別の CA を注入することもできます。

7.3. DNS エンドポイントの設定

DNS ホスト名を使用して SSL/TLS でオーバークラウドにアクセスする場合には、**/usr/share/openstack-tripleo-heat-templates/environments/predictable-placement/custom-domain.yaml** ファイルを **/home/stack/templates** ディレクトリーにコピーします。



注記

この環境ファイルが初回のデプロイメントに含まれていない場合は、TLS-everywhere のアーキテクチャーで再デプロイすることはできません。

すべてのフィールドのホスト名およびドメイン名を設定し、必要に応じてカスタムネットワークのパラメーターを追加します。

CloudDomain

ホストの DNS ドメイン

CloudName

オーバークラウドエンドポイントの DNS ホスト名

CloudNameCtlplane

プロビジョニングネットワークエンドポイントの DNS 名

CloudNameInternal

Internal API エンドポイントの DNS 名

CloudNameStorage

ストレージエンドポイントの DNS 名

CloudNameStorageManagement

ストレージ管理エンドポイントの DNS 名

手順

- 次のパラメーターのいずれかを使用して、使用する DNS サーバーを追加します。
  - **DEFAULT/undercloud_nameservers**
  - **%SUBNET_SECTION%/dns_nameservers**

7.4. オーバークラウド作成時の環境ファイルの追加

デプロイメントコマンド **openstack overcloud deploy** で **-e** オプションを使用して、デプロイメントプロセスに環境ファイルを追加します。本項の環境ファイルは、以下の順序で追加します。

- SSL/TLS を有効化する環境ファイル (**enable-tls.yaml**)
- DNS ホスト名を設定する環境ファイル (**custom-domain.yaml**)
- ルート認証局を注入する環境ファイル (**inject-trust-anchor-hiera.yaml**)
- パブリックエンドポイントのマッピングを設定するための環境ファイル:
  - パブリックエンドポイントへのアクセスに DNS 名を使用する場合には、**/usr/share/openstack-tripleo-heat-templates/environments/ssl/tls-endpoints-public-dns.yaml** を使用します。
  - パブリックエンドポイントへのアクセスに IP アドレスを使用する場合には、**/usr/share/openstack-tripleo-heat-templates/environments/ssl/tls-endpoints-public-ip.yaml** を使用します。

手順

- SSL/TLS 環境ファイルを含める方法の例として、次のデプロイコマンドスニペットを使用します。

```
$ openstack overcloud deploy --templates \
[...]
-e /home/stack/templates/enable-tls.yaml \
-e ~/templates/custom-domain.yaml \
-e ~/templates/inject-trust-anchor-hiera.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/ssl/tls-endpoints-public-dns.yaml
```

7.5. SSL/TLS 証明書の手動更新

TLS everywhere (TLS-e) プロセスで自動生成されない専用の SSL/TLS 証明書を使用する場合は、以下の手順を実施します。

手順

1. 以下のように heat テンプレートを編集します。

   - **enable-tls.yaml** ファイルを編集して、**SSLCertificate**、**SSLKey**、**SSLIntermediateCertificate** のパラメーターを更新してください。

   - 認証局が変更された場合には、**inject-trust-anchor-hiera.yaml** ファイルを編集して、**CAMap** パラメーターを更新してください。

2. プロイメントコマンドを再度実行します。

```
$ openstack overcloud deploy --templates \
[...]
-e /home/stack/templates/enable-tls.yaml \
-e ~/templates/custom-domain.yaml \
-e ~/templates/inject-trust-anchor-hiera.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/ssl/tls-endpoints-public-dns.yaml
```

3. Director で、コントローラーごとに次のコマンドを実行します。

```
ssh tripleo-admin@<controller> sudo podman \
restart $(podman ps --format="{{.Names}}" | grep -w -E 'haproxy(-bundle-.*-[0-9]+)?')
```

第8章 オーバークラウドでの暗号化に FERNET キーの使用

Fernet はデフォルトのトークンプロバイダーであり、**uuid** に置き換わるものです。Fernet デプロイメントを確認して、Fernet キーをローテーションできます。Fernet は、**/var/lib/config-data/puppet-generated/keystone/etc/keystone/fernet-keys** に保存されている 3 種類のキーを使用します。最も数値の高いディレクトリーには、新しいトークンを生成し、既存のトークンを復号するプライマリーキーが含まれます。

Fernet キーのローテーションでは、以下のプロセスを使用します。

1. プライマリーキーはセカンダリーキーになります。
2. <system> は新しいプライマリーキーを発行します。送信プライマリーキーは無効になりました。セカンダリーキーを使用して、以前のプライマリーキーに関連付けられたトークンを復号できますが、新しいトークンを発行することはできません。

Fernet 鍵のローテーションサイクルの長さを決定する際には、組織のセキュリティー体制に従います。組織にガイダンスがない場合は、セキュリティー上の理由から、毎月のローテーションサイクルが推奨されます。

8.1. FERNET デプロイメントの確認

Fernet トークンが適切に機能していることを確認するには、コントローラーノードの IP アドレスを取得し、コントローラーノードに対して SSH を実行し、トークンドライバーおよびプロバイダーの設定を確認します。

手順

1. コントローラーノードの IP アドレスを取得します。

```
[stack@director ~]$ source ~/stackrc
[stack@director ~]$ openstack server list
+--------------------------------------+------------------------+--------+-------------------------+
| ID                                   | Name                   | Status | Networks                |
+--------------------------------------+------------------------+--------+-------------------------+
| 756fbd73-e47b-46e6-959c-e24d7fb71328 | overcloud-controller-0 | ACTIVE | ctlplane=192.0.2.16 |
| 62b869df-1203-4d58-8e45-fac6cd4cfbee | overcloud-novacompute-0 | ACTIVE | ctlplane=192.0.2.8  |
+--------------------------------------+------------------------+--------+-------------------------+
```

2. コントローラーノードに対して SSH を実行します。

```
[tripleo-admin@overcloud-controller-0 ~]$ ssh tripleo-admin@192.0.2.16
```

3. トークンドライバーおよびプロバイダー設定の値を取得します。

```
[tripleo-admin@overcloud-controller-0 ~]$ sudo crudini --get /var/lib/config-data/puppet-generated/keystone/etc/keystone/keystone.conf token driver
sql
[tripleo-admin@overcloud-controller-0 ~]$ sudo crudini --get /var/lib/config-data/puppet-generated/keystone/etc/keystone/keystone.conf token provider
fernet
```

4. Fernet プロバイダーをテストします。

```
[tripleo-admin@overcloud-controller-0 ~]$ exit
[stack@director ~]$ source ~/overcloudrc
[stack@director ~]$ openstack token issue
------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------+
| Field | Value |
------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------+
| expires | 2016-09-20 05:26:17+00:00 |
| id | gAAAAABX4LppE8vaiFZ992eah2i3edpO1aDFxlKZq6a_RJzxUx56QVKORrmW0-oZK3-
Xuu2wcnpYq_eek2SGLz250eLpZOzxKBR0GsoMfxJU8mEFF8NzfLNcbuS-iz7SV-
N1re3XEywSDG90JcgwjQfXW-8jtCm-n3LL5IaZexAYIw059T_-cd8 |
| project_id | 26156621d0d54fc39bf3adb98e63b63d |
| user_id | 397daf32cadd490a8f3ac23a626ac06c |
------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------+
```

結果には長い Fernet トークンが含まれます。

8.2. WORKFLOW サービスの使用による FERNET キーのローテーション

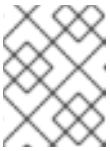
スタックの更新後も Fernet キーが維持されるようにするには、Workflow サービス (mistral) でキーをローテーションします。デフォルトでは、director は **ManageKeystoneFernetKeys** パラメーターを使用して、環境ファイルのオーバークラウドの Fernet キーを管理します。Fernet キーは、**KeystoneFernetKeys** セクションのワークフローサービスに保存されます。

手順

1. 既存の Fernet キーを確認します。

   a. Fernet キーの場所を特定します。heat-admin ユーザーとしてコントローラーノードにログインし、**crudini** コマンドを使用して Fernet キーをクエリーします。

```
[stack@<undercloud_host> ~]$ ssh tripleo-admin@overcloud-controller-o
[tripleo-admin@overcloud-controller-0 ~]$ sudo crudini --get /var/lib/config-data/puppet-
generated/keystone/etc/keystone/keystone.conf fernet_tokens key_repository
/etc/keystone/fernet-keys
```



注記

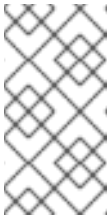
/etc/keystone/ ディレクトリーは、コンテナーのファイルシステムのパスを参照します。

   b. 現在の Fernet キーディレクトリーを検査します。

```
[tripleo-admin@overcloud-controller-0 ~]$ sudo ls /var/lib/config-data/puppet-
generated/keystone/etc/keystone/fernet-keys
0  1  2
```

   - **0** - 次のプライマリーキーになるステージングされたキーが含まれ、番号は常に **0** になります。

- **1** - セカンダリーキーが含まれます。
- **2** - プライマリーキーが含まれます。この数は、キーのローテーションが行われるたびに増えます。最大の数字が、常にプライマリーキーとして機能します。



注記

- キーの最大数は **max_active_keys** プロパティーで設定されます。デフォルトは 5 つのキーです。
- 鍵は、すべてのコントローラーノードに伝播します。

2. **workflow** コマンドを使用して Fernet キーをローテーションします。

```
[stack@director ~]$ source ~/stackrc
[stack@director ~]$ openstack workflow execution create
tripleo.fernet_keys.v1.rotate_fernet_keys {"container": "overcloud"}
+-------------------+--------------------------------------+
| Field             | Value                                |
+-------------------+--------------------------------------+
| ID                | 58c9c664-b966-4f82-b368-af5ed8de5b47 |
| Workflow ID       | 78f0990a-3d34-4bf2-a127-10c149bb275c |
| Workflow name     | tripleo.fernet_keys.v1.rotate_fernet_keys |
| Description       |                                      |
| Task Execution ID | <none>                               |
| State             | RUNNING                              |
| State info        | None                                 |
| Created at        | 2017-12-20 11:13:50                  |
| Updated at        | 2017-12-20 11:13:50                  |
+-------------------+--------------------------------------+
```

検証

1. ID を取得し、ワークフローが成功したことを確認します。

```
[stack@director ~]$ openstack workflow execution show 58c9c664-b966-4f82-b368-
af5ed8de5b47
+-------------------+--------------------------------------+
| Field             | Value                                |
+-------------------+--------------------------------------+
| ID                | 58c9c664-b966-4f82-b368-af5ed8de5b47 |
| Workflow ID       | 78f0990a-3d34-4bf2-a127-10c149bb275c |
| Workflow name     | tripleo.fernet_keys.v1.rotate_fernet_keys |
| Description       |                                      |
| Task Execution ID | <none>                               |
| State             | SUCCESS                              |
| State info        | None                                 |
| Created at        | 2017-12-20 11:13:50                  |
| Updated at        | 2017-12-20 11:15:00                  |
+-------------------+--------------------------------------+
```

2. コントローラーノードで Fernet キーの数を確認し、前の結果と比較します。

```
[tripleo-admin@overcloud-controller-0 ~]$ sudo ls /var/lib/config-data/puppet-
generated/keystone/etc/keystone/fernet-keys
0  1  2  3
```

- **0** - ステージングされたキーが含まれ、番号は常に **0** になります。このキーは、次回のローテーション時にプライマリーキーになります。
- **1 & 2** - セカンダリーキーが含まれます。
- **3** - プライマリーキーが含まれます。この数は、キーのローテーションが行われるたびに増えます。最大の数字が、常にプライマリーキーとして機能します。

注記

- キーの最大数は **max_active_keys** プロパティーで設定されます。デフォルトは 5 つのキーです。
- 鍵は、すべてのコントローラーノードに伝播します。

第9章 RED HAT OPENSTACK PLATFORM の連邦情報処理標準

重要

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、テスト用途にのみご利用いただく機能です。実稼働環境にはデプロイしないでください。テクノロジープレビュー機能についての詳細は、対象範囲の詳細 を参照してください。

連邦情報処理標準 (FIPS) は、米国国立標準技術研究所 (NIST) によって開発された一連のセキュリティー要件です。Red Hat Enterprise Linux 9 でサポートされている標準は、FIPS パブリケーション 140-3 です。**暗号化モジュールのセキュリティー要件**。サポートされている標準の詳細は、Federal Information Processing Standards Publication 140-3 を参照してください。

これらのセキュリティー要件は、受け入れ可能な暗号化アルゴリズムと、セキュリティーモジュールを含むそれらの暗号化アルゴリズムの使用を定義します。

- FIPS 140-3 検証は、FIPS を通じて承認された暗号アルゴリズムのみを、規定された方法で、検証済みモジュールを通じて使用することによって実現されます。
- FIPS 140-3 との互換性は、FIPS を通じて承認された暗号化アルゴリズムのみを使用することによって実現されます。

Red Hat OpenStack Platform 17 は **FIPS 140-3 と互換性があります**。Red Hat が提供するイメージを使用してオーバークラウドをデプロイすることで、FIPS 互換性を利用できます。

注記

OpenStack 17.1 は Red Hat Enterprise Linux (RHEL) 9.2 に基づいています。RHEL 9.2 はまだ FIPS 認定のために提出されていません。ただし、特定の期限は確約できませんが、Red Hat は RHEL 9.0 および RHEL 9.2 モジュール、その後は RHEL 9.x のマイナーリリースについても、FIPS 認定を取得することを想定しています。更新は Compliance Activities and Government Standards から入手できる予定です。

9.1. FIPS を有効にする

FIPS を有効にする場合は、アンダークラウドとオーバークラウドのインストール中に一連の手順を完了する必要があります。

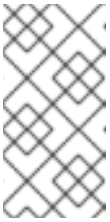
前提条件

- Red Hat Enterprise Linux のインストールが完了し、Red Hat OpenStack Platform director のインストールを開始する準備が整いました。

手順

1. アンダークラウドで FIPS を有効にします。
   a. アンダークラウドをインストールする予定のシステムで FIPS を有効にします。

```
fips-mode-setup --enable
```



注記

この手順により、**fips=1** カーネルパラメーターが GRUB 設定ファイルに追加されます。その結果、Red Hat Enterprise Linux で使用される暗号化アルゴリズムモジュールのみが FIPS モードになり、標準によって承認された暗号化アルゴリズムのみが使用されます。

b. システムを再起動します。

c. FIPS が有効になっていることを確認します。

```
fips-mode-setup --check
```

d. Red Hat OpenStack Platform director をインストールして設定します。詳細は、アンダークラウドへの director のインストール

2. オーバークラウド用に FIPS 対応のイメージを準備します。

a. オーバークラウドのイメージをインストールします。

```
sudo dnf -y install rhosp-director-images-uefi-fips-x86_64
```

b. **stack** ユーザーのホームディレクトリーに **images** ディレクトリーを作成します。

```
$ mkdir /home/stack/images
$ cd /home/stack/images
```

c. イメージをホームディレクトリーにデプロイします。

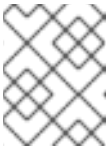
```
for i in /usr/share/rhosp-director-images/*fips*.tar; do tar -xvf $i; done
```

d. イメージをアップロードする前にシンボリックリンクを作成する必要があります。

```
ln -s ironic-python-agent-fips.initramfs       ironic-python-agent.initramfs
ln -s ironic-python-agent-fips.kernel          ironic-python-agent.kernel
ln -s overcloud-hardened-uefi-full-fips.qcow2  overcloud-hardened-uefi-full.qcow2
```

e. FIPS 対応のオーバークラウドイメージを Image サービスにアップロードします。

```
 openstack overcloud image upload --update-existing --whole-disk
```



注記

現在 OpenStack Image サービスにイメージがない場合でも、**--update-existing** フラグを使用する必要があります。

3. オーバークラウドで FIPS を有効にします。
環境に固有のオーバークラウドデプロイメント用のテンプレートを設定します。fips.yaml を含むすべての設定テンプレートをデプロイコマンドに含めます。

```
openstack overcloud deploy
...
-e /usr/share/openstack-tripleo-heat-templates/environents/fips.yaml
```

第10章 ユーザーアクセスセキュリティーの向上

重要

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、テスト用途にのみご利用いただく機能です。実稼働環境にはデプロイしないでください。テクノロジープレビュー機能についての詳細は、対象範囲の詳細 を参照してください。

Red Hat OpenStack Platform 17 でセキュアなロールベースのアクセス制御 (SRBAC) を有効にすることができます。SRBAC モデルには、プロジェクトスコープ内に存在する 3 つのロールに基づいて、3 つのペルソナがあります。

10.1. SRBAC のペルソナ

ペルソナは、ロールとロールが属するスコープの組み合わせです。Red Hat OpenStack Platform 17 をデプロイすると、プロジェクトスコープから任意のペルソナを割り当てることができます。

10.1.1. Red Hat OpenStack Platform SRBAC ロール

現在、プロジェクトの範囲内で 3 つの異なるロールを利用できます。

admin

admin ロールには、リソースまたは API に対するすべての作成、読み取り、更新、または削除操作が含まれます。

member

member ロールは、メンバーであるスコープによって所有されるリソースを作成、読み取り、更新、および削除することができます。

reader

reader ロールは、適用されるスコープに関係なく、読み取り専用操作用です。このロールは、それが適用されるスコープ全体にわたってリソースを表示できます。

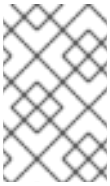
10.1.2. Red Hat OpenStack Platform SRBAC スコープ

スコープは、操作が実行されるコンテキストです。Red Hat OpenStack Platform 17 では、**project** スコープのみが利用可能です。**project** スコープは、OpenStack 内の分離されたセルフサービスリソース用の API のサブセットです。

10.1.3. Red Hat OpenStack Platform SRBAC ペルソナ

プロジェクト管理者

プロジェクト管理者ペルソナは使用可能な唯一の管理者ペルソナであるため、Red Hat OpenStack Platform 17 には、プロジェクト管理者ペルソナに最高レベルの権限を付与する修正されたポリシーが含まれています。このペルソナには、プロジェクト全体のリソースに対する作成、読み取り、更新、および削除操作が含まれます。これには、ユーザーや他のプロジェクトの追加と削除が含まれます。



注記

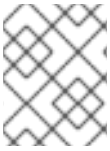
このペルソナは、将来の開発によって範囲が変更されることが予想されます。このロールは、プロジェクトメンバーおよびプロジェクトリーダーに付与されるすべての権限を意味します。

プロジェクトメンバー

プロジェクトメンバーペルソナは、プロジェクトスコープ内のリソースを消費する権限を付与されたユーザー用です。このペルソナは、割り当てられているプロジェクト内のリソースを作成、一覧表示、更新、および削除できます。このペルソナは、プロジェクトリーダーに付与されるすべての権限を意味します。

プロジェクトリーダー

プロジェクトリーダーペルソナは、プロジェクト内の機密性の低いリソースを表示する権限を付与されたユーザー用です。プロジェクトでは、リソースを検査または表示する必要があるエンドユーザー、または監査目的で 1 つのプロジェクト内のプロジェクト固有のリソースのみを表示する必要がある監査者にリーダーのロールを割り当てます。プロジェクトリーダーペルソナは、すべての監査ユースケースに対応できるわけではありません。



注記

system または **domain** スコープに基づく追加のペルソナは開発中であり、使用できません。

10.2. 安全なロールベースのアクセス制御の有効化

セキュアなロールベースの認証をアクティブ化すると、Red Hat OpenStack Platform 環境でユーザーに割り当てられるパーミッションの範囲を定義する一連の新しいポリシーファイルがアクティブ化されます。

前提条件

- Red Hat OpenStack Platform director 環境がインストールされている。

手順

- Red Hat OpenStack Platform をデプロイするときに、デプロイメントスクリプトに **enable-secure-rbac.yaml** 環境ファイルを含めます。

```
openstack overcloud deploy --templates
…
-e /usr/share/openstack-tripleo-heat-templates/environments/enable-secure-rbac.yaml
```

10.3. SRBAC 環境でのロールの割り当て

Red Hat OpenStack Platform で SRBAC を使用すると、ユーザーを **project-admin**、**project-member**、または **project-reader** のロールに割り当てることができます。

前提条件

- 安全なロールベース認証 (SRBAC) を使用して Red Hat OpenStack Platform をデプロイしました。

手順

- 次の構文を使用して **openstack role add** コマンドを使用します。
  - **admin** のロールを割り当てます。

```
$ openstack role add --user <user> --user-domain <domain> --project <project> --
project-domain <project-domain> admin
```

  - **member** のロールを割り当てます。

```
$ openstack role add --user <user> --user-domain <domain> --project <project>  --
project-domain <project-domain> member
```

  - **reader** のロールを割り当てます。

```
$ openstack role add --user <user> --user-domain <domain> --project <project> --
project-domain <project-domain> reader
```

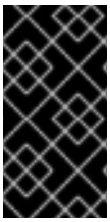
- **<user>** を既存のユーザーに置き換えて、ロールを適用します。
- **<domain>** を、ロールが適用されるドメインに置き換えます。
- **<project>** を、ユーザーがロールを付与されているプロジェクトに置き換えます。
- **<project-domain>** をプロジェクトが属するドメインに置き換えます。

第11章 ポリシー

各 OpenStack サービスには、アクセスポリシーで管理されるリソースが含まれています。たとえば、リソースには以下の関数が含まれる場合があります。

- インスタンスを作成して起動するパーミッション
- インスタンスにボリュームを接続する機能

Red Hat OpenStack Platform (RHOSP) 管理者は、カスタムポリシーを作成して、さまざまなアクセスレベルで新しいロールを導入することや、既存のロールのデフォルト動作を変更することができます。



重要

Red Hat では、カスタマイズされたロールまたはポリシーはサポートしていません。構文エラーや、承認が誤って適用されると、セキュリティーやユーザービリティーに悪影響を及ぼす可能性があります。実稼働環境でカスタマイズされたロールまたはポリシーが必要な場合は、サポート例外について Red Hat サポートにお問い合わせください。

11.1. 既存のポリシーの確認

サービスのポリシーファイルはこれまで **/etc/$service** ディレクトリーに存在していました。たとえば、Compute (nova) の **policy.json** ファイルの完全パスは **/etc/nova/policy.json** でした。

既存のポリシーを見つける方法に影響を与える重要なアーキテクチャーの変更には、以下の 2 つがあります。

- Red Hat OpenStack Platform はコンテナー化されています。
  - サービスコンテナーからポリシーファイルを確認する場合は、ポリシーファイル (ある場合) は従来のパスにあります。
    /etc/$service/policy.json
  - サービスコンテナーの外部からそれらを表示する場合は、以下のパスに配置されます。
    /var/lib/config-data/puppet-generated/$service/etc/$service/policy.json
- 各サービスには、コードで提供されるデフォルトのポリシーがあります。これは、手動で作成した場合にのみ利用できるファイル、または **oslopolicy** ツールで生成された場合に限ります。ポリシーファイルを生成するには、以下の例のように、コンテナーから **oslopolicy-policy-generator** を使用します。

```
podman exec -it keystone oslopolicy-policy-generator --namespace keystone
```

デフォルトでは、生成されたポリシーは osly.policy CLI ツールにより stdout にプッシュされます。

11.2. サービスポリシーについて

サービスポリシーファイルステートメントは、エイリアス定義またはルールです。エイリアスの定義はファイルの先頭に存在します。以下のリストには、Compute (nova) 用に生成された **policy.json** ファイルからのエイリアス定義の説明が記載されています。

- "context_is_admin": "role:admin"
  ターゲットの後に **rule:context_is_admin** が表示されたら、そのアクションを許可する前にユーザーが管理コンテキストで動作していることを確認します。

- "admin_or_owner": "is_admin:True or project_id:%(project_id)s"
  ターゲットの後に **admin_or_owner** が表示されると、ポリシーはそのユーザーが admin であるか、プロジェクト ID がターゲットオブジェクトの独自のプロジェクト ID と一致することを確認してからそのアクションを許可します。

- "admin_api": "is_admin:True
  ターゲットの後に **admin_api** が表示されると、ポリシーはそのアクションを許可する前にそのユーザーが admin であることをチェックします。

11.3. ポリシー構文

Policy.json ファイルは特定のオペレーターをサポートするので、これらの設定のターゲットスコープを制御できます。たとえば、以下の keystone の設定には、ユーザー作成の管理ユーザーだけがルールが含まれます。

```
"identity:create_user": "rule:admin_required"
```

: 文字の左側にあるセクションでは、特権について説明し、右側のセクションは、特権を使用できるユーザーを定義します。右側にオペレーターを使用して、スコープをさらに制御することもできます。

- **!**: このアクションを実行するユーザー (admin を含む) はありません。

- **@** および **""**: 任意のユーザーがこのアクションを実行できます。

- **not**、**and**、**or**: 標準の Operator 機能を利用できます。

たとえば、以下の設定は、新規ユーザーを作成するパーミッションを持たないことを意味します。

```
"identity:create_user": "!"
```

11.4. アクセス制御でのポリシーファイルの使用

重要

Red Hat では、カスタマイズされたロールまたはポリシーはサポートしていません。構文エラーや、承認が誤って適用されると、セキュリティーやユーザービリティーに悪影響を及ぼす可能性があります。実稼働環境でカスタマイズされたロールまたはポリシーが必要な場合は、サポート例外について Red Hat サポートにお問い合わせください。

デフォルトのルールを上書きするには、適切な OpenStack サービスの **policy.json** ファイルを編集します。たとえば、Compute サービスには nova ディレクトリーに **policy.json** があります。これは、コンテナーから表示する際にコンテナー化されたサービスの正しい場所となります。

注記

- ステージング環境でポリシーファイルへの変更をよくテストしてから、それらを実稼働環境で実装する必要があります。

- アクセス制御ポリシーへの変更が、リソースのセキュリティーを意図せずに脆弱化しないことを確認する必要があります。また、**policy.json** ファイルへの変更はすぐに有効であり、サービスの再起動は必要ありません。

以下に例を示します。パワーユーザーロールの作成

keystone ロールのパーミッションをカスタマイズするには、サービスの **policy.json** ファイルを更新します。これは、ユーザーのクラスに割り当てるパーミッションをより詳細に定義できることを意味します。以下の例では、以下の特権を使用してデプロイメントの **power user** ロールを作成します。

- インスタンスを起動するには、以下のコマンドを実行します。
- インスタンスを停止します。
- インスタンスに割り当てられているボリュームを管理します。

このロールの意図は、**admin** アクセスを付与せずに、特定のユーザーに追加のパーミッションを付与することです。これらの特権を使用するには、以下のパーミッションをカスタムロールに付与する必要があります。

- インスタンスを起動する: **"os_compute_api:servers:start": "role:PowerUsers"**
- インスタンスを停止する: **"os_compute_api:servers:stop": "role:PowerUsers"**
- 特定のボリュームを使用するようにインスタンスを設定する: **"os_compute_api:servers:create:attach_volume": "role:PowerUsers"**
- インスタンスに割り当てられているボリュームをリスト表示する: **"os_compute_api:os-volumes-attachments:index": "role:PowerUsers"**
- ボリュームを割り当てる: **"os_compute_api:os-volumes-attachments:create": "role:PowerUsers"**
- 割り当てられたボリュームの詳細を表示する: **"os_compute_api:os-volumes-attachments:show": "role:PowerUsers"**
- インスタンスに割り当てられているボリュームを変更する: **"os_compute_api:os-volumes-attachments:update": "role:PowerUsers"**
- インスタンスに割り当てられているボリュームを削除する: **"os_compute_api:os-volumes-attachments:delete": "role:PowerUsers"**

注記

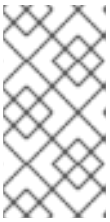
policy.json ファイルを変更すると、デフォルトのポリシーを上書きします。その結果、**PowerUsers** のメンバーは、これらのアクションを実行できる唯一のユーザーになります。**admin** ユーザーがこれらのパーミッションを保持できるようにするには、admin_or_power_user. のルールを作成できます。また、基本的な条件ロジックを使用して **role:PowerUsers or role:Admin** を定義することもできます。

1. カスタム keystone ロールを作成します。

```
$ openstack role create PowerUsers
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | None                             |
| id        | 7061a395af43455e9057ab631ad49449 |
| name      | PowerUsers                       |
+-----------+----------------------------------+
```

2. 既存のユーザーをロールに追加し、ロールをプロジェクトに割り当てます。

```
$ openstack role add --project [PROJECT_NAME] --user [USER_ID] [PowerUsers-ROLE_ID]
```



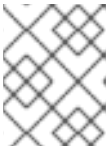
注記

ロールの割り当ては、1つのプロジェクトのみとペアになります。つまり、ロールをユーザーに割り当てると、ターゲットプロジェクトも同時に定義します。ユーザーが同じロールを受信し、別のプロジェクトを対象にする場合は、ロールを別々に割り当てる必要がありますが、別のプロジェクトが対象となります。

3. デフォルトの nova ポリシー設定を表示します。

```
$ oslopolicy-policy-generator --namespace nova
```

4. 以下のエントリーを **/var/lib/config-data/puppet-generated/nova/etc/nova/policy.json** に追加して、新しい **PowerUsers** ロールのカスタムパーミッションを作成します。



注記

デプロイメント前にポリシーの変更をテストし、想定通りに機能していることを確認します。

```
{
"os_compute_api:servers:start": "role:PowerUsers",
"os_compute_api:servers:stop": "role:PowerUsers",
"os_compute_api:servers:create:attach_volume": "role:PowerUsers",
"os_compute_api:os-volumes-attachments:index": "role:PowerUsers",
"os_compute_api:os-volumes-attachments:create": "role:PowerUsers",
"os_compute_api:os-volumes-attachments:show": "role:PowerUsers",
"os_compute_api:os-volumes-attachments:update": "role:PowerUsers",
"os_compute_api:os-volumes-attachments:delete": "role:PowerUsers"
}
```

このファイルを保存して nova コンテナーを再起動する際に、変更を実装する。**PowerUsers** keystone ロールに追加したユーザーは、これらの権限を受信します。

11.5. 以下に例を示します。属性に基づくアクセスの制限

重要

Red Hat では、カスタマイズされたロールまたはポリシーはサポートしていません。構文エラーや、承認が誤って適用されると、セキュリティーやユーザービリティーに悪影響を及ぼす可能性があります。実稼働環境でカスタマイズされたロールまたはポリシーが必要な場合は、サポート例外について Red Hat サポートにお問い合わせください。

API 呼び出しを実行するユーザーの属性に基づいて、API 呼び出しへのアクセスを制限するポリシーを作成できます。たとえば、以下のデフォルトのルールは、管理者コンテキストから実行された場合のキーペアの削除が許可されるか、トークンのユーザー ID がターゲットに関連付けられたユーザー ID と一致することを示しています。

```
"os_compute_api:os-keypairs:delete": "rule:admin_api or user_id:%(user_id)s"
```

注記: *新しい実装機能は、各リリースと共に各サービスに保証される訳ではありません。したがって、ターゲットサービスの既存のポリシーの規則を使用してルールを作成することが重要です。これらのポリシー表示の詳細については、既存のポリシーの確認を参照してください。* すべてのポリシーは、リリース間の互換性の保証が保証されないため、デプロイするすべてのバージョンの非実稼働環境でテストする必要があります。

上記の例では、API ルールを作成し、リソースを所有するかどうかに基づいてユーザーへのアクセスを拡張または制限できます。また、属性と他の制限を組み合わせて、以下の例で示すようにルールを形成できます。

```
"admin_or_owner": "is_admin:True or project_id:%(project_id)s"
```

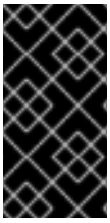
上記の例を考慮すると、管理者およびユーザーに限定した一意のルールを作成し、そのルールを使用してさらにアクションを制限することができます。

```
"admin_or_user": "is_admin:True or user_id:%(user_id)s"
"os_compute_api:os-instance-actions": "rule:admin_or_user"
```

関連情報

- ポリシー構文。

11.6. HEAT でのポリシーの変更



重要

Red Hat では、カスタマイズされたロールまたはポリシーはサポートしていません。構文エラーや、承認が誤って適用されると、セキュリティーやユーザービリティーに悪影響を及ぼす可能性があります。実稼働環境でカスタマイズされたロールまたはポリシーが必要な場合は、サポート例外について Red Hat サポートにお問い合わせください。

heat を使用して、オーバークラウド内の特定のサービスのアクセスポリシーを設定できます。次のパラメーターを使用して、それぞれのサービスにポリシーを設定します。

表11.1 ポリシーパラメーター

パラメーター	説明
KeystonePolicies	OpenStack Identity (keystone) 向けに設定するポリシーのハッシュ
IronicApiPolicies	OpenStack Bare Metal (ironic) API 向けに設定するポリシーのハッシュ
BarbicanPolicies	OpenStack Key Manager (barbican) 向けに設定するポリシーのハッシュ
NeutronApiPolicies	OpenStack Networking (neutron) API 向けに設定するポリシーのハッシュ

パラメーター	説明
SaharaApiPolicies	OpenStack Clustering (sahara) API 向けに設定するポリシーのハッシュ
NovaApiPolicies	OpenStack Compute (nova) API 向けに設定するポリシーのハッシュ
CinderApiPolicies	OpenStack Block Storage (cinder) API 向けに設定するポリシーのハッシュ
GlanceApiPolicies	OpenStack Image Storage (glance) API 向けに設定するポリシーのハッシュ
HeatApiPolicies	OpenStack Orchestration (heat) API 向けに設定するポリシーのハッシュ

サービスのポリシーを設定するには、サービスのポリシーを含むハッシュ値をポリシーパラメーターに指定します。次に例を示します。

- OpenStack Identity (keystone) には **KeystonePolicies** パラメーターを使用します。このパラメーターを環境ファイルの **parameter_defaults** セクションで設定します。

```
parameter_defaults:
  KeystonePolicies: { keystone-context_is_admin: { key: context_is_admin, value: 'role:admin' } }
```

- OpenStack Compute (nova) には **NovaApiPolicies** パラメーターを使用します。このパラメーターを環境ファイルの **parameter_defaults** セクションで設定します。

```
parameter_defaults:
  NovaApiPolicies: { nova-context_is_admin: { key: 'compute:get_all', value: '@' } }
```

11.7. ユーザーおよびロールの監査

Red Hat OpenStack Platform で利用可能なツールを使用して、ユーザーおよび関連付けられた権限ごとのロール割り当てのレポートをビルドできます。

前提条件

- Red Hat OpenStack Platform 環境がインストールされている。
- スタックとしてディレクターにログインしています。

手順

1. **openstack role list** コマンドを実行して、環境内の現在ロールを表示します。

```
openstack role list -c Name -f value

swiftoperator
```

```
ResellerAdmin
admin
_member_
heat_stack_user
```

2. **openstack role assignment list** コマンドを実行して、特定のロールのメンバーであるすべてのユーザーをリスト表示します。たとえば、admin ロールを持つユーザーを確認するには、以下を実行します。

```
$ openstack role assignment list --names --role admin
+-------+-----------------------------------+-------+----------------+------------+--------+-----------+
| Role  | User                              | Group | Project        | Domain     | System | Inherited |
+-------+-----------------------------------+-------+----------------+------------+--------+-----------+
| admin | heat-cfn@Default                  |       | service@Default |            |        | False     |
| admin | placement@Default                 |       | service@Default |            |        | False     |
| admin | neutron@Default                   |       | service@Default |            |        | False     |
| admin | zaqar@Default                     |       | service@Default |            |        | False     |
| admin | swift@Default                     |       | service@Default |            |        | False     |
| admin | admin@Default                     |       | admin@Default   |            |        | False     |
| admin | zaqar-websocket@Default           |       | service@Default |            |        | False     |
| admin | heat@Default                      |       | service@Default |            |        | False     |
| admin | ironic-inspector@Default          |       | service@Default |            |        | False     |
| admin | nova@Default                      |       | service@Default |            |        | False     |
| admin | ironic@Default                    |       | service@Default |            |        | False     |
| admin | glance@Default                    |       | service@Default |            |        | False     |
| admin | mistral@Default                   |       | service@Default |            |        | False     |
| admin | heat_stack_domain_admin@heat_stack |       |                | heat_stack |        | False
|
| admin | admin@Default                     |       |                | all        | False  |
+-------+-----------------------------------+-------+----------------+------------+--------+-----------+
```

注記

-f {csv,json,table,value,yaml} パラメーターを使用して、これらの結果をエクスポートできます。

11.8. API アクセスの監査

所定のロールがアクセスできる API 呼び出しを監査できます。各ロールに対してこのプロセスを繰り返すと、各ロールのアクセス可能な API に包括的なレポートが実行されます。

前提条件

- ターゲットロールのユーザーとしてソースとする認証ファイル。
- JSON 形式のアクセストークン
- 監査する各サービスの API のポリシーファイル。

手順

1. 次に、必要なロールでユーザーの認証ファイルを一時停止します。
2. Keystone 生成されたトークンを取得し、ファイルに保存します。そのためには、openstack-cli

コマンドを実行し、--debug オプションを指定して、指定したトークンを stdout に出力します。このトークンをコピーして、アクセスファイルに保存できます。このコマンドを 1 つのステップとして実行します。

```
openstack token issue --debug 2>&1 | egrep ^'{\"token\":' > access.file.json
```

3. ポリシーファイルを作成します。これは、コンテナー化されたサービスをホストするオーバークラウドノード上で実行できます。以下の例では、cinder サービスのポリシーファイルを作成します。

```
ssh tripleo-admin@CONTROLLER-1 sudo podman exec cinder_api \
oslopolicy-policy-generator \
--config-file /etc/cinder/cinder.conf \
--namespace cinder > cinderpolicy.json
```

4. これらのファイルを使用して、cinder の API にアクセスするための対象のロールを監査できるようになりました。

```
oslopolicy-checker --policy cinderpolicy.json --access access.file.json
```

第12章 ネットワークタイムプロトコル

Red Hat OpenStack Platform クラスター内のシステムが、システム間で正確かつ一貫性のあるタイムスタンプを持つことを確認する必要があります。

Red Hat Enterprise Linux 8 上の Red Hat OpenStack Platform は、時間管理に Chrony をサポートしています。詳細は、Using the chrony suite to configure NTP を参照してください。

12.1. 一貫した時刻が重要な理由

運用とセキュリティー上のニーズの両方で、組織全体で時刻が一貫していることが重要です。

セキュリティーイベントの特定

一貫した時刻を維持することにより、影響を受けるシステムのイベントのタイムスタンプを関連付けることができ、イベントのシーケンスを理解することができます。

認証システムおよびセキュリティーシステム

セキュリティーシステムは、以下の例のように、時刻のずれの影響を受ける場合があります。

- Kerberos ベースの認証システムは、時刻のずれの秒数により影響を受けるクライアントの認証を拒否する可能性があります。
- トランスポート層セキュリティー (TLS) 証明書は、有効な時刻ソースに依存します。クライアントとサーバーシステム間の差異が **Valid From** の日付範囲を超えると、クライアントからサーバーへの TLS 接続が失敗します。

Red Hat OpenStack Platform のサービス

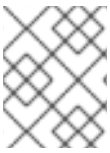
高可用性 (HA) や Ceph など、一部の OpenStack のコアサービスは、特に正確な時刻管理に依存しています。

12.2. NTP 設計

Network time protocol (NTP) は、階層的な設定で整理されています。各レイヤーは stratum と呼ばれます。階層の最上位は、原子時計などの stratum 0 デバイスです。NTP 階層では、Stratum 0 デバイスは、一般に利用可能な stratum 1 および stratum 2 の NTP 時刻サーバーに参照を提供します。

データセンタークライアントは、一般に利用可能な NTP stratum 1 または 2 サーバーに直接接続しないでください。直接接続の数は、パブリック NTP リソースに不必要な負荷をかけます。代わりに、データセンターで専用の時刻サーバーを割り当て、クライアントをその専用サーバーに接続します。

インスタンスがホストではなく専用の時刻サーバーから時刻を受信するように設定します。



注記

Red Hat OpenStack Platform 環境内で実行中のサービスコンテナーは、それらのサービスが存在するホストから時刻を受信します。

第13章 インフラストラクチャーおよび仮想化の強化

物理環境と仮想環境を強化して、内部および外部の脅威に対する保護を強化できます。

13.1. RED HAT OPENSTACK PLATFORM のハードウェア

クラウド環境で使用するハードウェアを追加する場合は、ハードウェア仮想化がサポートされていることを確認してください。使用しないハードウェア機能を無効にします。

手順

1. Red Hat OpenStack の認定ハードウェアを チェックして、ハードウェアがサポートされていることを確認します。
2. ハードウェア仮想化が利用可能で有効になっていることを確認します。

```
cat /proc/cpuinfo | egrep "vmx|svm"
```

3. ハードウェアプラットフォームのすべてのファームウェアが最新であることを確認します。詳細については、ハードウェアベンダーのドキュメントを参照してください。

13.2. クラウド環境でのソフトウェア更新

セキュリティー、パフォーマンス、およびサポート性のために、Red Hat OpenStack Platform (RHOSP) を最新の状態に保ちます。

- 更新時にカーネルの更新が含まれている場合は、更新した物理システムまたはインスタンスを再起動する必要があります。
- OpenStack Image (glance) イメージを更新して、新しく作成されたインスタンスに最新の更新が適用されるようにします。
- RHOSP でパッケージを選択的に更新する場合は、すべてのセキュリティー更新プログラムが含まれていることを確認してください。最新の脆弱性とセキュリティー更新プログラムの詳細については、次を参照してください。
  - RHSA-announce
  - エラータ通知
  - セキュリティーアドバイザリー

13.3. ハードウェアとソフトウェアの機能を制限する

使用するハードウェアおよびソフトウェア機能のみを有効にして、攻撃の可能性にさらされるコードを少なくします。信頼できる環境でのみ有効にする必要がある機能がいくつかあります。

PCI パススルー

PCI パススルーにより、インスタンスはノード上の PCI デバイスに直接アクセスできます。PCI デバイスにアクセスできるインスタンスでは、悪意のあるアクターがファームウェアを変更できる可能性があります。さらに、一部の PCI デバイスにはダイレクトメモリーアクセス (DMA) があります。インスタンスが DMA を使用してデバイスを制御できるようにすると、任意の物理メモリーアクセスが可能になります。

Network Functions Virtualization (NFV) などの特定のユースケースでは、PCI パススルーを有効にする必要があります。デプロイメントに必要でない限り、PCI パススルーを有効にしないでください。

カーネルの同一ページマージ

カーネルの同一ページマージ (KSM) は、メモリーページの重複排除および共有によってメモリーの使用を削減する機能です。2 つ以上の仮想マシンがメモリー内に同一のページを持っている場合、それらのページを共有して密度を高めることができます。メモリーの重複排除戦略は、サイドチャネル攻撃に対して脆弱であり、信頼できる環境でのみ使用する必要があります。Red Hat OpenStack Platform では、KSM はデフォルトで無効になっています。

13.4. RED HAT OPENSTACK PLATFORM 上の SELINUX

SELinux (Security-Enhanced Linux) は、強制アクセス制御 (MAC) の実装です。MAC は、プロセスまたはアプリケーションがシステム上で実行できることを制限することにより、攻撃の影響を制限します。SELinux の詳細は、SELinux とは を参照してください。.

Red Hat OpenStack Platform (RHOSP) サービス用に SELinux ポリシーが事前設定されています。RHOSP では、SELinux は、個別のセキュリティーコンテキストで各 QEMU プロセスを実行するように設定されます。RHOSP では、SELinux ポリシーはハイパーバイザーのホストとインスタンスを次の脅威から保護するのに役立ちます。

ハイパーバイザーの脅威

インスタンス内で実行している侵害されたアプリケーションは、ハイパーバイザーを攻撃して、基盤となるリソースにアクセスします。インスタンスがハイパーバイザー OS にアクセスできる場合は、物理デバイスやその他のアプリケーションがターゲットになる可能性があります。この脅威は、かなりのリスクを表しています。ハイパーバイザーが侵害されると、ファームウェア、他のインスタンス、およびネットワークリソースも侵害される可能性があります。

インスタンスの脅威

インスタンス内で実行されている侵害されたアプリケーションは、ハイパーバイザーを攻撃して、別のインスタンスとそのリソース、またはインスタンスファイルイメージにアクセスまたは制御します。実際のネットワークを保護するための管理戦略は、仮想環境には直接適用されません。すべてのインスタンスは SELinux によってラベル付けされたプロセスであるため、Linux カーネルによって適用される各インスタンスの周りにセキュリティー境界があります。

RHOSP では、ディスク上のインスタンスイメージファイルは SELinux データ型 **svirt_image_t** でラベル付けされます。インスタンスの電源がオンになると、SELinux はランダムな数値識別子をイメージに追加します。ランダムな数値識別子により、侵害された OpenStack インスタンスが他のコンテナーへの不正アクセスを取得するのを防ぐことができます。SELinux は、各ハイパーバイザーノードに最大 524,288 個の数値識別子を割り当てることができます。

13.5. コンテナー化されたサービスの調査

Red Hat OpenStack Platform に付属する OpenStack サービスは、コンテナー内で実行されます。コンテナー化により、依存関係に関連する競合なしでサービスの開発とアップグレードが可能になります。サービスがコンテナー内で実行される場合、そのサービスに対する潜在的な脆弱性も含まれます。

次の手順を使用して、環境で実行されているサービスに関する情報を取得できます。

手順

- `podman inspect を使用して、マウントされたホストディレクトリーのバインドなどの情報を取得します。
  以下に例を示します。
  -

```
$ sudo podman inspect <container_name> | less
```

<container_name> をコンテナーの名前に置き換えます。たとえば、**nova compute** です。

- **/var/log/containers** にあるサービスのログを確認します。
以下に例を示します。

```
sudo less /var/log/containers/nova/nova-compute.log
```

- コンテナー内で対話型 CLI セッションを実行します。
以下に例を示します。

```
podman exec -it nova_compute /bin/bash
```

注記

コンテナー内で直接、テスト目的でサービスに変更を加えることができます。コンテナーを再起動すると、すべての変更が失われます。

13.6. コンテナー化されたサービスに一時的な変更を加える

コンテナーが再起動されても維持されるが、Red Hat OpenStack Platform (RHOSP) クラスターの永続的な設定には影響しない、コンテナー化されたサービスに変更を加えることができます。これは、設定の変更をテストしたり、トラブルシューティング時にデバッグレベルのログを有効にしたりするのに役立ちます。変更を手動で元に戻すことができます。または、RHOSP クラスターで再デプロイを実行すると、すべてのパラメーターが永続的な設定にリセットされます。

/var/lib/config-data/puppet-generated/service にある設定ファイルを使用して、サービスに一時的な変更を加えます。次の例では、nova サービスでデバッグを有効にします。

手順

1. **nova_compute** コンテナーにバインドマウントされている **nova.conf** 設定ファイルを編集します。**debug** パラメーターの値を **True** に設定します。

```
$ sudo sed -i 's/^debug=.*/debug=True' \
/var/lib/config-data/puppet-generated/nova/etc/nova/nova.conf
```

警告

OpenStack ファイルの設定ファイルは、**DEFAULT** や **[database]** などの複数のセクションを持つ **ini** ファイルです。各セクションに固有のパラメーターは、ファイル全体で固有ではない場合があります。**sed** は慎重に使用してください。**egrep -v "^$|^#" [configuration_file] | grep [parameter]** を実行して、**parameter** が設定ファイルに複数回出現するかどうかを確認できます。

2. nova コンテナーを再起動します。

```
sudo podman restart nova_compute
```

13.7. コンテナー化されたサービスに永続的な変更を加える

Heat を使用して、Red Hat OpenStack Platform (RHOSP) サービスのコンテナー化されたサービスに永続的な変更を加えることができます。最初に RHOSP をデプロイしたときに使用した既存のテンプレートを使用するか、新しいテンプレートを作成してデプロイスクリプトに追加します。次の例では、libvirt の秘密鍵のサイズを **4096** に増やしています。

手順

1. **libvirt-keysize.yaml** という新しい **yaml** テンプレートを作成し、**LibvirtCertificateKeySize** パラメーターを使用してデフォルト値を **2048** から **4096** に増やします。

```
cat > /home/stack/templates/libvirt-keysize.yaml
parameter_defaults:
      LibvirtCertificateKeySize: 4096
EOF
```

2. **libvirt-keysize.yaml** 設定ファイルをデプロイスクリプトに追加します。

```
openstack overcloud deploy --templates \
...
-e /home/stack/templates/libvirt-keysize.yaml
...
```

3. デプロイスクリプトを再実行します。

```
./deploy.sh
```

13.8. ファームウェアの更新

物理サーバーは複雑なファームウェアを使用して、サーバーのハードウェアと軽量な管理カードを有効にして操作します。このカードでは、独自のセキュリティー脆弱性を持つことができ、システムアクセスと中断が可能になります。これに対処するために、ハードウェアベンダーは、オペレーティングシステムの更新とは別にインストールされるファームウェアの更新を発行します。この更新を取得、テスト、および実装する運用セキュリティープロセスが必要になります。ファームウェアの更新では多くの場合で、物理ホストの再起動が効果的に行われるためです。

13.9. SSH バナーテキストの使用

SSH 経由で接続する全ユーザーにコンソールメッセージを表示するバナーを設定できます。環境ファイルの以下のパラメーターを使用して、**/etc/issue** にバナーテキストを追加できます。実際の要件に合わせて、このサンプルテキストをカスタマイズすることを検討してください。

```
resource_registry:
  OS::TripleO::Services::Sshd:
    /usr/share/openstack-tripleo-heat-templates/deployment/sshd/sshd-baremetal-puppet.yaml

parameter_defaults:
  BannerText: |
    ******************************************************************
```

```
* This system is for the use of authorized users only. Usage of  *
* this system may be monitored and recorded by system personnel. *
* Anyone using this system expressly consents to such monitoring *
* and is advised that if such monitoring reveals possible         *
* evidence of criminal activity, system personnel may provide     *
* the evidence from such monitoring to law enforcement officials.*
*****************************************************************
```

この変更をデプロイメントに適用するには、設定を **ssh_banner.yaml** という名前のファイルで保存し、以下のように **overcloud deploy** コマンドに渡します。**<full environment>** は、元のデプロイメントパラメーターをすべて含める必要があることを示します。以下に例を示します。

```
openstack overcloud deploy --templates \
  -e <full environment> -e  ssh_banner.yaml
```

13.10. システムイベントの監査

すべての監査イベントの記録を維持すると、システムベースラインの確立、トラブルシューティングの実行、特定の結果の原因となったイベントシーケンスの分析に役立ちます。監査システムは、システム時刻の変更、Mandatory/Discretionary アクセス制御の変更、ユーザーおよびグループの作成や削除など、多数のタイプのイベントをログに記録できます。

環境ファイルを使用してルールを作成し、director がそれを **/etc/audit/audit.rules** に挿入します。以下に例を示します。

```
  resource_registry:
    OS::TripleO::Services::AuditD: /usr/share/openstack-tripleo-heat-
templates/deployment/auditd/auditd-baremetal-puppet.yaml
  parameter_defaults:
    AuditdRules:
      'Record Events that Modify User/Group Information':
        content: '-w /etc/group -p wa -k audit_rules_usergroup_modification'
        order  : 1
      'Collects System Administrator Actions':
        content: '-w /etc/sudoers -p wa -k actions'
        order  : 2
      'Record Events that Modify the Systems Mandatory Access Controls':
        content: '-w /etc/selinux/ -p wa -k MAC-policy'
        order  : 3
```

13.11. ファイアウォールルールの管理

ファイアウォールルールは、デプロイメント時にオーバークラウドノードに自動的に適用され、OpenStack の運用に必要なポートのみを公開することを目的としています。必要に応じて、追加のファイアウォールルールを指定できます。たとえば、Zabbix モニタリングシステムのルールを追加するには、以下のようにします。

```
parameter_defaults:
  ControllerExtraConfig:
    ExtraFirewallRules:
      '301 allow zabbix':
```

```
    dport: 10050
    proto: tcp
    source: 10.0.0.8
```

注記

action パラメーターを設定しない場合、結果は **accept** になります。**action** パラメーターは、**drop**、**insert**、または **append** のみに設定できます。

アクセスを制限するルールを追加することもできます。ルールの定義時に使用される数字は、ルールの優先順位を決定します。たとえば、RabbitMQ のルール番号は、デフォルトでは **109** です。これを抑制するには、小さい数字を使用するように切り替えます。

```
parameter_defaults:
  ControllerParameters
    ExtraFirewallRules:
      '098 allow rabbit from internalapi network':
        dport: [4369,5672,25672]
        proto: tcp
        source: 10.0.0.0/24
      '099 drop other rabbit access:
        dport: [4369,5672,25672]
        proto: tcp
        action: drop
```

この例では、**098** と **099** は、RabbitMQ のルール番号 **109** よりも小さい任意に選んだ番号です。ルールの番号を確認するには、適切なノードで iptables ルールを検査できます。RabbitMQ の場合は、コントローラーをチェックします。

```
iptables-save
[...]
-A INPUT -p tcp -m multiport --dports 4369,5672,25672 -m comment --comment "109 rabbitmq" -m
state --state NEW -j ACCEPT
```

または、puppet 定義からポート要件を抽出することもできます。たとえば、RabbitMQ のルールは **puppet/services/rabbitmq.yaml** に保存されます。

```
    ExtraFirewallRules:
      '109 rabbitmq':
        dport:
          - 4369
          - 5672
          - 25672
```

ルールには、以下のパラメーターを設定できます。

- **dport**:ルールに関連付けられた宛先ポート
- **sport**:ルールに関連付けられた送信元ポート
- **proto**:ルールに関連付けられたプロトコル。デフォルトは tcp です。
- **action**:ルールに関連付けられたアクションポリシー。デフォルトは INSERT であり、ジャンプを ACCEPTS に設定します。

- **state**:ルールに関連付けられた状態の配列。デフォルトは [**NEW**] です。
- **source**:ルールに関連付けられた送信元の IP アドレス
- **interface**:ルールに関連付けられたネットワークインターフェイス
- **chain**:ルールに関連付けられたチェーン。デフォルトは **INPUT** です。
- **destination**:ルールに関連付けられた宛先の cidr

13.12. AIDE を使用した侵入検知

AIDE (Advanced Intrusion Detection Environment) は、ファイルとディレクトリーの整合性チェッカーです。これは、承認されていないファイルの改ざんまたは変更のインシデントを検出するために使用されます。たとえば、AIDE は、システムパスワードファイルが変更された場合に警告を出すことができます。

AIDE は、システムファイルを分析し、ファイルハッシュの整合性データベースをまとめることで機能します。次に、データベースは、ファイルとディレクトリーの整合性を検証し、変更を検出する際の比較ポイントとなります。

director には AIDE サービスが含まれており、AIDE 設定にエントリーを追加でき、AIDE サービスはこれを使用して整合性データベースを作成できます。以下に例を示します。

```
resource_registry:
  OS::TripleO::Services::Aide:
    /usr/share/openstack-tripleo-heat-templates/deployment/aide/aide-baremetal-ansible.yaml

parameter_defaults:
  AideRules:
    'TripleORules':
      content: 'TripleORules = p+sha256'
      order: 1
    'etc':
      content: '/etc/ TripleORules'
      order: 2
    'boot':
      content: '/boot/ TripleORules'
      order: 3
    'sbin':
      content: '/sbin/ TripleORules'
      order: 4
    'var':
      content: '/var/ TripleORules'
      order: 5
    'not var/log':
      content: '!/var/log.*'
      order: 6
    'not var/spool':
      content: '!/var/spool.*'
      order: 7
    'not nova instances':
      content: '!/var/lib/nova/instances.*'
      order: 8
```

注記

上記の例は、積極的には保守やベンチマーク設定されないので、要件に合わせて AIDE の値を選択する必要があります。

1. **TripleORules** という名前のエイリアスを宣言することで、毎回同じ属性を繰り返し除外する必要がなくなります。

2. エイリアスは **p+sha256** の属性を受け取ります。AIDE では、これは次の命令として解釈されます。**sha256** の整合性チェックサムを使用してすべてのファイルパーミッション **p** を監視する。

AIDE の設定ファイルで利用可能な属性の完全リストは、AIDE MAN ページ (https://aide.github.io/) を参照してください。

以下の手順を実行して、変更をデプロイメントに適用します。

1. **/home/stack/templates/** ディレクトリーに **aide.yaml** というファイルとして設定を保存します。

2. **aide.yaml** 環境ファイルを編集して、お使いの環境に適したパラメーターおよび値を指定します。

3. ご自分の環境に固有のその他すべての heat テンプレートおよび環境ファイルと共に、**/home/stack/templates/aide.yaml** 環境ファイルを **openstack overcloud deploy** コマンドに追加します。

```
openstack overcloud deploy --templates
...
-e /home/stack/templates/aide.yaml
```

13.12.1. 複雑な AIDE ルールの使用

前述の形式を使用して、複雑なルールを作成できます。以下に例を示します。

```
MyAlias = p+i+n+u+g+s+b+m+c+sha512
```

上記は、次の命令として解釈されます。チェックサムの生成に sha256 を使用して、パーミッション、inode、リンクの数、ユーザー、グループ、サイズ、ブロック数、mtime、ctime をモニターする。

エイリアスの順番の位置は常に **1** であることに注意してください。つまり、AIDE ルールの先頭に配置され、それ以下のすべての値に再帰的に適用されます。

エイリアスの後は、監視するディレクトリーになります。正規表現を使用できます。たとえば、**var** ディレクトリーの監視を設定しますが、**!** を使用して not 句で上書きします (**'!/var/log.*'** および **'!/var/spool.*'**)。

13.12.2. その他の AIDE 値

以下の AIDE 値も使用できます。

AideConfPath:aide 設定ファイルへの完全な POSIX パス。デフォルトは **/etc/aide.conf** です。ファイルの場所を変更する要件がない場合は、デフォルトのパスのままにすることが推奨されます。

AideDBPath:AIDE 整合性データベースへの完全な POSIX パス。この値は設定が可能で、オペレーターが独自のフルパスを宣言できます。多くの場合、AIDE データベースファイルはノード外に保管されるためです (読み取り専用のファイルマウント)。

AideDBTempPath:AIDE 整合性一時データベースへの完全な POSIX パス。この一時ファイルは、AIDE が新規データベースを初期化する際に作成されます。

AideHour:この値は、AIDE cron 設定の一部として hour 属性を設定します。

Aide Minute:この値は、AIDE cron 設定の一部として minute 属性を設定します。

AideCronUser:この値は、Linux ユーザーを AIDE cron 設定の一部として設定するためのものです。

AideEmail:この値は、cron が実行されるたびに AIDE レポートを受信するメールアドレスを設定します。

AideMuaPath:この値は、**AideEmail** で設定したメールアドレスに AIDE レポートを送信するために使用される Mail User Agent へのパスを設定します。

13.12.3. AIDE の cron 設定

AIDE director サービスにより、cron ジョブを設定できます。デフォルトでは、レポートを **/var/log/audit/** に送信します。メールアラートを使用する場合は、**AideEmail** パラメーターを有効にして、設定されたメールアドレスにアラートを送信します。重大なアラートをメールに依存することは、システム停止や意図しないメッセージフィルタリングに対して脆弱である可能性があることに注意してください。

13.12.4. システムアップグレードの影響に関する考慮

アップグレードが実行されると、AIDE サービスが新しい整合性データベースを自動的に再生成し、アップグレードしたすべてのファイルが正しく再計算され、更新されたチェックサムが生成されるようにします。

openstack overcloud deploy が初期デプロイメントに対して後続の実行として呼び出され、AIDE 設定ルールが変更されると、director AIDE サービスはデータベースを再構築して、整合性データベースに新規設定属性が取り込まれるようにします。

13.13. SECURETTY の確認

securetty を使用すると、コンソールデバイス (tty) に対する root アクセスを無効にできます。この動作は、**/etc/securetty** ファイルのエントリーで管理されます。以下に例を示します。

```
resource_registry:
  OS::TripleO::Services::Securetty: ../puppet/services/securetty.yaml

parameter_defaults:
  TtyValues:
    - console
    - tty1
    - tty2
    - tty3
    - tty4
    - tty5
    - tty6
```

13.14. IDENTITY サービスの CADF 監査

詳細な監査プロセスは、OpenStack デプロイメントの現在の状態を確認するのに役立ちます。これは、セキュリティーモデルにおけるそのロールにより、特に keystone で重要です。

Red Hat OpenStack Platform では、Identity およびトークン操作の CADF イベントを生成する keystone サービスにより、監査イベントのデータ形式として Cloud Auditing Data Federation (CADF) が採用されています。**KeystoneNotificationFormat** を使用して、keystone の CADF 監査を有効にできます。

```
parameter_defaults:
  KeystoneNotificationFormat: cadf
```

13.15. LOGIN.DEFS 値の確認

新規システムユーザー (keystone 以外) のパスワード要件を強化するために、以下のこれらのパラメーターの例に従って、director はエントリーを **/etc/login.defs** に追加できます。

```
resource_registry:
  OS::TripleO::Services::LoginDefs: ../puppet/services/login-defs.yaml

parameter_defaults:
  PasswordMaxDays: 60
  PasswordMinDays: 1
  PasswordMinLen: 5
  PasswordWarnAge: 7
  FailDelay: 4
```

第14章 DASHBOARD サービスの強化

Dashboard サービス (horizon) は、管理者が設定した制限内で独自のリソースをプロビジョニングするためのセルフサービスポータルをユーザーに提供します。OpenStack API と同じ機密性で Dashboard サービスのセキュリティーを管理します。

14.1. DASHBOARD サービスのデバッグ

DEBUG パラメーターのデフォルト値は **False** です。実稼働環境では、デフォルト値を保持してください。この設定は、調査中のみ変更してください。**DEBUG** パラメーターの値を **True** に変更すると、Django は Web サーバーの状態に関する機密情報を含むスタックトレースをブラウザーユーザーに出力できます。

DEBUG パラメーターの値が **True** の場合は、**ALLOWED_HOSTS** 設定も無効になります。**ALLOWED_HOSTS** の設定の詳細については、ALLOWED_HOSTS の設定 を参照してください。

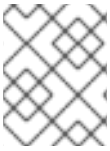
14.2. ドメイン名の選択

Dashboard サービス (horizon) は、任意のレベルの共有ドメインではなく、2 次レベルのドメインにデプロイすることを推奨します。それぞれの例を以下に示します。

- 2 次レベルのドメイン: **https://example.com**
- 共有サブドメイン: **https://example.public-url.com**

Dashboard サービスを専用の 2 次レベルのドメインにデプロイすると、ブラウザーの **same-origin** ポリシーに基づいて、Cookie とセキュリティートークンが他のドメインから分離されます。サブドメインにデプロイされた場合、ダッシュボードサービスのセキュリティーは、同じ 2 次レベルのドメインにデプロイされた最も安全性の低いアプリケーションと同等になります。

クッキーでサポートされるセッションストアを回避し、HTTP Strict Transport Security (HSTS) (本書で説明されている) を設定することにより、このリスクをさらに軽減できます。



注記

https://example/ などのベアドメインへの Dashboard サービスのデプロイはサポートされていません。

14.3. ALLOWED_HOSTS の設定

Horizon は python Django ウェブフレームワークでビルドされており、誤解を招く HTTP Host ヘッダーに関連するセキュリティ脅威からの保護が必要です。この保護を適用するには、OpenStack ダッシュボードによって提供される FQDN を使用するように **ALLOWED_HOSTS** を設定します。

ALLOWED_HOSTS 設定を設定すると、このリストの値と一致しない Host ヘッダーを持つ HTTP 要求は拒否され、エラーが発生します。

手順

1. テンプレートの **parameter_defaults** の下で、**HorizonAllowedHosts** パラメーターの値を設定します。

```
parameter_defaults:
  HorizonAllowedHosts: <value>
```

-

<value> を OpenStack ダッシュボードによって提供される FQDN に置き換えます。

2. 変更したテンプレートを使用してオーバークラウドをデプロイし、環境に必要な他のすべてのテンプレートをデプロイします。

14.4. クロスサイトスクリプティング (XSS)

OpenStack ダッシュボードは、ほとんどのフィールドで Unicode 文字セット全体を受け入れます。悪意のある攻撃者は、この拡張性を利用して、クロスサイトスクリプティング (XSS) の脆弱性をテストしようとする可能性があります。OpenStack Dashboard サービス (horizon) には、XSS の脆弱性を強化するツールがあります。カスタムダッシュボードでこれらのツールを正しく使用することが重要です。カスタムダッシュボードに対して監査を実行する場合は、次の点に注意してください。

- **mark_safe** 関数
- **is_safe**: カスタムテンプレートタグと共に使用する場合
- **safe** テンプレートタグ
- 自動エスケープがオフで、不適切にエスケープされたデータを評価する可能性のある JavaScript の場合

14.5. クロスサイトリクエストフォージェリー (CSRF)

複数の JavaScript インスタンスを使用する Dashboard は、**@csrf_exempt** デコレーターの不適切な使用など、脆弱性について監査する必要があります。CORS (Cross Origin Resource Sharing) の制限を下げる前に、推奨されるセキュリティー設定に従っていないダッシュボードを評価してください。各応答で制限付き CORS ヘッダーを送信するように Web サーバーを設定します。ダッシュボードのドメインとプロトコルのみを許可します (例: **Access-Control-Allow-Origin: https://example.com**/)。ワイルドカードオリジンを許可しないでください。

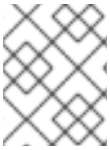
14.6. IFRAME 埋め込みの許可

DISALLOW_IFRAME_EMBED 設定は、Dashboard が iframe 内に埋め込まれるのを防ぎます。従来のブラウザーは、クロスフレームスクリプティング (XFS) に対して脆弱性があります。したがって、このオプションを使用すると、iframes を要求しないデプロイメントにさらにセキュリティー強化機能が追加されます。この設定は、デフォルトで **True** に設定されていますが、必要な場合は環境ファイルを使用して無効にできます。

手順

- 以下のパラメーターを使用して、iframe 埋め込みを許可できます。

```
parameter_defaults:
  ControllerExtraConfig:
    horizon::disallow_iframe_embed: false
```



注記

これらの設定は、潜在的なセキュリティーの影響を完全に理解した後にのみ、**False** に設定するべきです。

14.7. DASHBOARD トラフィックの HTTPS 暗号化の使用

HTTPS を使用して Dashboard トラフィックを暗号化することが推奨されます。これは、認識されている認証局 (CA) からの有効な信頼される証明書を使用するように設定することで実行できます。プライベート組織が発行した証明書は、信頼の root がすべてのユーザーブラウザーで事前インストールされている場合にのみ適切になります。

完全修飾 HTTPS URL にリダイレクトするように、Dashboard ドメインへの HTTP 要求を設定します。

詳細は、*7章オーバークラウドのパブリックエンドポイントでの SSL/TLS の有効化* を参照してください。

14.8. HTTP STRICT TRANSPORT SECURITY (HSTS)

HTTP Strict Transport Security(HSTS) は、最初にセキュアな接続を行った後に、ブラウザーが後続の非セキュアな接続を確立できないようにします。パブリックまたは信頼できないゾーンに HTTP サービスをデプロイした場合、HSTS が特に重要になります。

director ベースのデプロイメントの場合、この設定は **/usr/share/openstack-tripleo-heat-templates/deployment/horizon/horizon-container-puppet.yaml** ファイルでデフォルトで有効になっています。

```
horizon::enable_secure_proxy_ssl_header: true
```

検証

オーバークラウドがデプロイされたら、検証のために Red Hat OpenStack ダッシュボード (horizon) の **local_settings** ファイルを確認します。

1. **ssh** を使用してコントローラーに接続します。

```
$ ssh tripleo-admin@controller-0
```

2. **SECURE_PROXY_SSL_HEADER** パラメーターの値が **('HTTP_X_FORWARDED_PROTO', 'https')** であることを確認します。

```
sudo egrep ^SECURE_PROXY_SSL_HEADER /var/lib/config-data/puppet-generated/horizon/etc/openstack-dashboard/local_settings
SECURE_PROXY_SSL_HEADER = ('HTTP_X_FORWARDED_PROTO', 'https')
```

14.9. フロントエンドキャッシング

OpenStack API 要求から直接生成される動的コンテンツをレンダリングするため、Dashboard でフロントエンドキャッシュツールを使用することは推奨されません。そのため、**varnish** などのフロントエンドキャッシュレイヤーにより正しいコンテンツが表示されなくなります。Dashboard は Django を使用します。これは Web サービスから直接提供される静的メディアに対応し、Web ホストのキャッシュの利点をすでに活用します。

14.10. セッションバックエンド

director ベースのデプロイメントの場合、horizon のデフォルトのセッションバックエンドは **django.contrib.sessions.backends.cache** で、memcached と組み合わされます。パフォーマンス上の理由から、このアプローチはローカルメモリーキャッシュに対して推奨されます。高可用性や負荷分

散のインストールの場合により安全であり、単一キャッシュとして扱いながら複数のサーバーでキャッシュを共有することができます。

これらの設定は、director の **horizon.yaml** ファイルで確認することができます。

```
horizon::cache_backend: django.core.cache.backends.memcached.MemcachedCache
horizon::django_session_engine: 'django.contrib.sessions.backends.cache'
```

14.11. シークレットキーの確認

Dashboard は、一部のセキュリティー機能について共有の **SECRET_KEY** 設定に依存します。秘密鍵は、少なくとも 64 文字の長さで無作為に生成される文字列である必要があります。これは、すべてのアクティブな Dashboard インスタンスで共有する必要があります。この鍵を危険にさらすと、リモートの攻撃者は任意コードを実行できる可能性があります。このキーをローテーションすると、既存のユーザーセッションおよびキャッシュが無効になります。このキーをパブリックリポジトリーにコミットしないでください。

director のデプロイメントでは、この設定は **HorizonSecret** の値として管理されます。

14.12. セッションクッキーの設定

Dashboard セッションクッキーは、JavaScript などのブラウザーテクノロジーによる対話のために開くことができます。TLS everywhere を使用した director デプロイメントの場合、**HorizonSecureCookies** 設定を使用してこの動作を強化することができます。

注記

CSRF またはセッションクッキーを、先頭のドットでワイルドカードドメインを使用するように設定しないでください。

14.13. 静的メディア

Dashboard の静的メディアは、Dashboard ドメインのサブドメインにデプロイして、Web サーバーによって提供されます。また、外部のコンテンツ配信ネットワーク (CDN) の使用も受け入れ可能です。このサブドメインに Cookie を設定したり、ユーザーによって提供されるコンテンツを提供したりしないでください。メディアは HTTPS で提供される必要もあります。

Dashboard のデフォルト設定では、**django_compressor** を使用して CSS および JavaScript コンテンツを圧縮および最小化してから提供します。このプロセスは、デフォルトのリクエスト内動的圧縮を使用し、デプロイされたコードと共に得られたファイルをコピーまたは CDN サーバーに送付するのではなく、Dashboard をデプロイする前に静的に行う必要があります。圧縮は実稼働以外のビルド環境で実行する必要があります。これができない場合は、リソース圧縮を完全に無効にすることを検討してください。オンライン圧縮の依存関係 (less, Node.js) は、実稼働マシンにインストールしないでください。

14.14. パスワードの複雑性の検証

OpenStack Dashboard (horizon) は、パスワード検証チェックを使用してパスワードの複雑さを強制的に適用することができます。

手順

1. パスワードの検証に正規表現を指定することや、テストに失敗した場合に表示されるヘルプテキストを指定します。以下の例では、8 文字から 18 文字までのパスワードを作成することをユーザーに要求します。

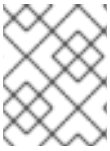
```
parameter_defaults:
  HorizonPasswordValidator: '^.{8,18}$'
  HorizonPasswordValidatorHelp: 'Password must be between 8 and 18 characters.'
```

1. この変更をデプロイメントに適用します。設定を **horizon_password.yaml** というファイルとして保存し、以下のように **overcloud deploy** コマンドに渡します。**<full environment>** は、元のデプロイメントパラメーターをすべて含める必要があることを示します。以下に例を示します。

```
openstack overcloud deploy --templates \
  -e <full environment> -e  horizon_password.yaml
```

14.15. 管理者パスワードチェックの強制

以下の設定は、デフォルトで **True** に設定されていますが、必要な場合は環境ファイルを使用して無効にできます。



注記

これらの設定は、潜在的なセキュリティーの影響を完全に理解した後にのみ、**False** に設定するべきです。

手順

Dashboard の **local_settings.py** ファイルの **ENFORCE_PASSWORD_CHECK** 設定により、**Change Password** フォームに **Admin Password** フィールドが表示されます。これは、管理者がパスワード変更を開始していることを検証するのに役立ちます。

- 環境ファイルを使用して **ENFORCE_PASSWORD_CHECK** を無効にすることができます。

```
parameter_defaults:
  ControllerExtraConfig:
    horizon::enforce_password_check: false
```

14.16. パスワード表示の無効化

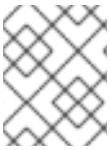
disable_password_reveal パラメーターはデフォルトで **True** に設定されていますが、必要に応じて環境ファイルを使用して無効にすることができます。パスワード表示ボタンは、ダッシュボードのユーザーが入力するパスワードを表示できるようにします。

手順

- **ControllerExtraConfig** パラメーターの下に、**Horizon::disable_password_reveal: false** を含めます。これを heat 環境ファイルに保存し、デプロイコマンドに含めます。

例

```
parameter_defaults:
  ControllerExtraConfig:
    horizon::disable_password_reveal: false
```



注記

これらの設定は、潜在的なセキュリティーの影響を完全に理解した後にのみ、**False** に設定するべきです。

14.17. ダッシュボードのログオンバナーの表示

HIPAA、PCI-DSS、および米国政府などの規制では、ユーザーログオンバナーを表示する必要があります。Red Hat OpenStack Platform (RHOSP) Dashboard (horizon) では、horizon コンテナー内に保管されるデフォルトのテーマ (RCUE) が使用されます。

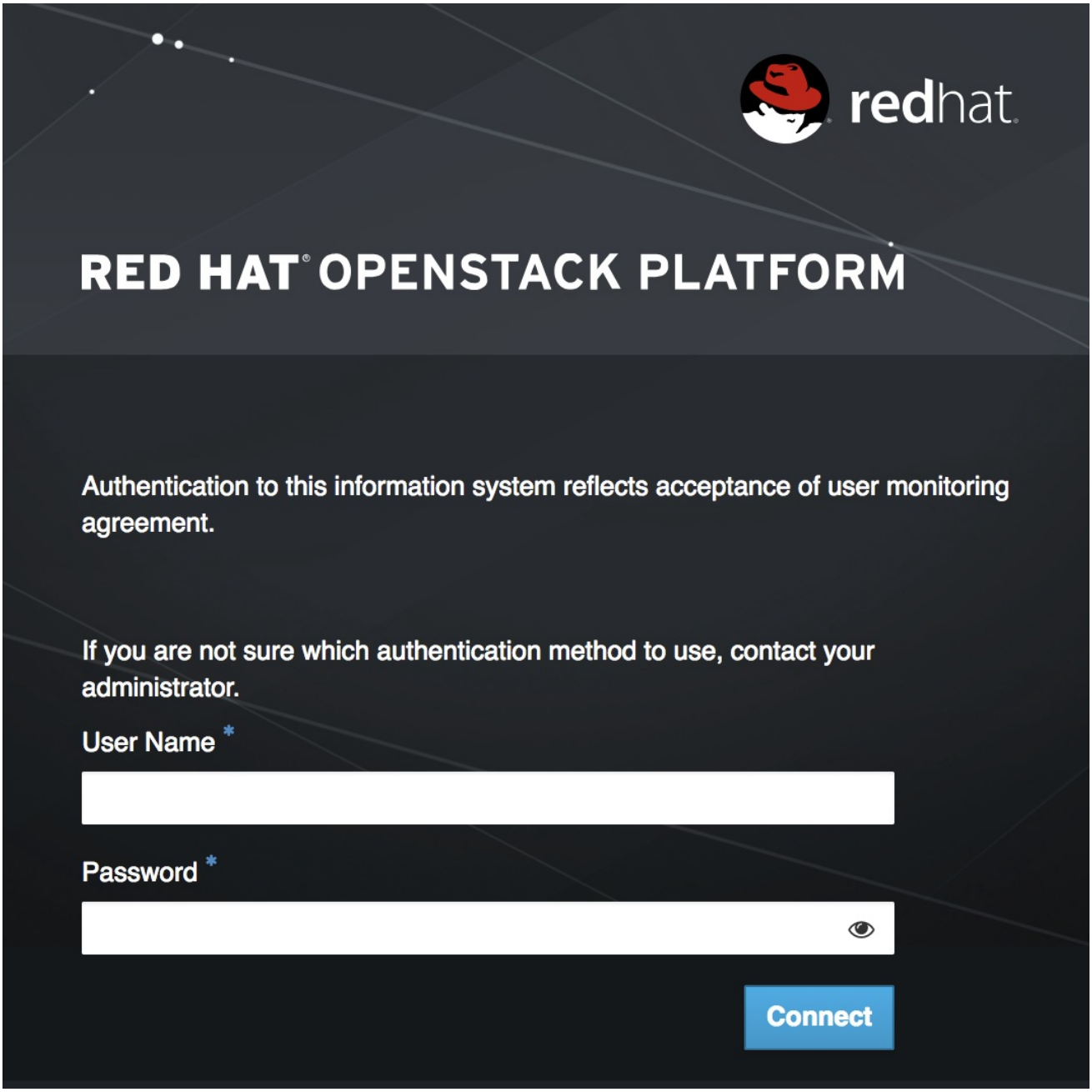
カスタムの Dashboard コンテナーでは、**/usr/share/openstack-dashboard/openstack_dashboard/themes/rcue/templates/auth/login.html** ファイルを手動で編集して、ログオンバナーを作成できます。

手順

1. **{% include 'auth/_login.html' %}** セクションの直前に、必要なログオンバナーを入力します。HTML タグが許可されます。

```
<snip>
<div class="container">
  <div class="row-fluid">
    <div class="span12">
      <div id="brand">
        <img src="../../static/themes/rcue/images/RHOSP-Login-Logo.svg">
      </div><!--/#brand-->
    </div><!--/.span*-->

    <!-- Start of Logon Banner -->
    <p>Authentication to this information system reflects acceptance of user monitoring
agreement.</p>
    <!-- End of Logon Banner -->

    {% include 'auth/_login.html' %}
  </div><!--/.row-fluid→
</div><!--/.container-->

{% block js %}
  {% include "horizon/_scripts.html" %}
{% endblock %}

  </body>
</html>
```

上の例では、次のようなダッシュボードが作成されます。



関連情報

- Dashboard のカスタマイズ

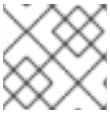
14.18. ファイルのアップロードサイズの制限

オプションとして、ファイルのアップロードのサイズを制限するように Dashboard を設定できます。この設定は、さまざまなセキュリティー強化ポリシーで必要になる場合があります。

LimitRequestBody: この値 (バイト単位) は、Dashboard を使用して転送できるファイル (イメージや他の大きなファイルなど) の最大サイズを制限します。

重要

この設定は、Red Hat では正式にテストされていません。この設定の影響を十分にテストしてから、実稼働環境にデプロイすることが推奨されます。



注記

値が小さすぎると、ファイルのアップロードは失敗します。

たとえば、この設定では、各アップロードファイルが最大 10 GB (**10737418240**) に制限されます。実際のデプロイメントに合わせて、この値を修正する必要があります。

- **/var/lib/config-data/puppet-generated/horizon/etc/httpd/conf/httpd.conf**

```
<Directory />
  LimitRequestBody 10737418240
</Directory>
```

- **/var/lib/config-data/puppet-generated/horizon/etc/httpd/conf.d/10-horizon_vhost.conf**

```
<Directory "/var/www">
  LimitRequestBody 10737418240
</Directory>
```

- **/var/lib/config-data/puppet-generated/horizon/etc/httpd/conf.d/15-horizon_ssl_vhost.conf**

```
<Directory "/var/www">
  LimitRequestBody 10737418240
</Directory>
```

注記

これらの設定ファイルは Puppet によって管理されるため、**openstack overcloud deploy** プロセスを実行するたびに、マネージド外の変更が上書きされます。

第15章 NETWORKING サービスのハードニング

Networking サービス (neutron) は、Red Hat OpenStack Platform (RHOSP) のソフトウェア定義ネットワーク (SDN) のコンポーネントです。RHOSP Networking サービスは、仮想マシンインスタンスとの間の内部および外部トラフィックを管理し、ルーティング、セグメンテーション、DHCP、メタデータなどのコアサービスを提供します。仮想ネットワーク機能とスイッチ、ルーター、ポート、ファイアウォールの管理のための API を提供します。

Red Hat OpenStack Platfrom Networking サービスの詳細は、ネットワークガイド を参照してください。

本項では、OpenStack デプロイメント内のプロジェクトネットワークセキュリティーに適用される、OpenStack Networking の設定に関する適切なプラクティスについて説明します。

15.1. API サーバーのバインドアドレスの制限: NEUTRON-SERVER

OpenStack Networking API サービスが受信クライアント接続用にネットワークソケットをバインドするインターフェイスまたは IP アドレスを制限するには、**/var/lib/config-data/puppet-generated/neutron/etc/neutron/neutron.conf** ファイルで **bind_host** および **bind_port** を指定します。

```
# Address to bind the API server
bind_host = IP ADDRESS OF SERVER

# Port the bind the API server to
bind_port = 9696
```

15.2. プロジェクトネットワークサービスのワークフロー

OpenStack Networking は、ユーザーがネットワークリソースのセルフサービス設定を提供します。クラウドアーキテクトとオペレーターは、ユーザーが利用可能なネットワークリソースを作成、更新、破棄できる設計のユースケースを評価することが重要です。

15.3. ネットワークリソースポリシーエンジン

OpenStack Networking 内のポリシーエンジンと設定ファイル (**policy.json**) は、プロジェクトネットワークのメソッドおよびオブジェクトに対するユーザーの詳細な認可を提供する方法を提供します。OpenStack Networking ポリシーの定義は、ネットワークの可用性、ネットワークセキュリティー、および OpenStack セキュリティー全体に影響します。クラウドアーキテクトとオペレーターは、ネットワークリソースの管理に対するユーザーおよびプロジェクトアクセスに対して、ポリシーを慎重に評価する必要があります。

注記

このポリシーはセキュリティーの問題に応じて変更できるため、デフォルトのネットワークリソースポリシーを確認することが重要です。

OpenStack が複数の外部アクセスポイントを提供する場合は、複数の仮想 NIC を複数の外部アクセスポイントにアタッチするプロジェクトの機能を制限することが重要です。よりこれらのセキュリティーゾーンがブリッジされ、セキュリティーへの不正アクセスの原因となる可能性があります。Compute が提供するホスト集約関数を使用するか、プロジェクトインスタンスを異なる仮想ネットワーク設定を持つ複数のプロジェクトに分割することで、このリスクを軽減することができます。ホストアグリゲートの詳細は、Creating and managing host aggregates を参照してください。

15.4. セキュリティーグループ

セキュリティーグループとは、セキュリティーグループルールのコレクションです。セキュリティーグループとそれらのグループにより、管理者およびプロジェクトがトラフィックの種別を指定することができ、かつ方向 (ingress/egress) が仮想インターフェイスポートを通過できる方向 (ingress/egress) を指定することができます。OpenStack Networking で仮想インターフェイスのポートが作成されると、セキュリティーグループが関連付けられます。デプロイメントごとに動作を変更するには、デフォルトのセキュリティーグループにルールを追加できます。

Compute API を使用してセキュリティーグループを変更する場合、更新されたセキュリティーグループはインスタンスのすべての仮想インターフェイスポートに適用されます。これは、neutron にあるように、コンピュートセキュリティーグループ API がポートベースではなくインスタンスベースであるためです。

15.5. ARP スプーフィングの緩和策

OpenStack Networking には、インスタンスの ARP スプーフィングの脅威を緩和するための組み込み機能が含まれています。これは、結果となるリスクに注意を払う場合を除き無効にしないでください。

15.6. 認証にセキュアプロトコルを使用

/var/lib/config-data/puppet-generated/neutron/etc/neutron/neutron.conf で、**[keystone_authtoken]** セクションの **auth_uri** の値が https: で始まる Identity API エンドポイントに設定されていることを確認します。

第16章 RED HAT OPENSTACK PLATFORM でのブロックストレージの強化

OpenStack Block Storage (cinder) は、ソフトウェア (サービスおよびライブラリー) を提供するサービスで、永続的なブロックレベルのストレージデバイスをセルフサービスで管理するサービスです。これにより、Compute (nova) インスタンスで使用する Block Storage リソースへのオンデマンドアクセスが作成されます。これにより、ブロックストレージのプールをさまざまなバックエンドストレージデバイスに仮想化することで、ソフトウェア定義のストレージが作成され、ソフトウェア実装または従来のハードウェアストレージ製品のいずれかになります。これの主な機能は、ブロックデバイスの作成、割り当て、切断を管理することです。コンシューマーは、バックエンドストレージ機器の種類や、そのタイプの配置先に関する知識は必要ありません。

コンピュートインスタンスは、iSCSI、ATA over Ethernet、または Fibre-Channel などの業界標準のストレージプロトコルを使用してブロックストレージを保存し、取得します。これらのリソースは、OpenStack ネイティブ HTTP RESTful API を使用して管理および設定されます。

16.1. 要求のボディーの最大サイズの設定

リクエストごとの最大ボディーサイズが定義されていない場合、攻撃者は大きなサイズの任意の OSAPI リクエストを作成します。すると、その結果サービスがクラッシュし、最終的にサービス拒否攻撃につながることになります。最大値を割り当てると、悪意のあるリクエストはすべてブロックされ、引き続きサービスの可用性が確保されます。

cinder.conf の **[oslo_middleware]** セクションの **max_request_body_size** が **114688** に設定されているかどうかを確認します。

16.2. ボリュームの暗号化の有効化

暗号化されていないボリュームデータにより、攻撃者が多数の仮想マシンのデータを読み取ることができるため、ボリュームホスティングプラットフォームは特に攻撃者にとってハイバリューターゲットとなります。さらに、物理ストレージメディアは、盗まれたり、再マウントされたりする可能性ががあり、別のマシンからアクセスされることもあり得ます。ボリュームのデータとボリュームのバックアップを暗号化すると、これらのリスクを軽減し、ボリュームをホストしているプラットフォームに厚い防御を提供することができます。Block Storage (cinder) は、ディスクに書き込まれる前にボリュームデータを暗号化できるため、ボリュームの暗号化を有効にし、秘密鍵ストレージに Barbican を使用することを検討してください。

16.3. ボリュームの接続

ブロックストレージデバイスを消去する方法は複数あります。従来の方法として、**lvm_type** を thin に設定し、続いて **volume_clear** パラメーターを使用します。または、ボリュームの暗号化機能があれば、ボリュームの暗号化キーが削除される場合にボリュームのワイプは必要ありません。

注記

以前のバージョンでは、**lvm_type=default** は wipe を示すために使用されていました。この手法は依然として機能しますが、secure delete の設定に **lvm_type=default** は推奨されません。

volume_clear パラメーターは、引数として **zero** または **shred** のいずれかを利用できます。**zero** は、ゼロのシングルパスをデバイスに書き込みます。**shred** 操作は、事前に決定したビットパターンの 3 つを書き込みます。

第17章 SHARED FILE SYSTEMS (MANILA) のセキュリティー強化

Shared File Systems サービス (manila) は、マルチプロジェクトのクラウド環境で共有ファイルシステムを管理するためのサービスセットを提供します。manila を使用すると、共有ファイルシステムを作成し、可視性、アクセシビリティー、クォータなどの属性を管理できます。

manila についての詳しい情報は、Storage Guide(https://access.redhat.com/documentation/ja-jp/red_hat_openstack_platform/17.0/html-single/storage_guide/) を参照してください。

17.1. MANILA のセキュリティーに関する考慮事項

Manila は keystone に登録されており、**manila endpoints** コマンドを使用して API を特定することができます。以下に例を示します。

```
$ manila endpoints
+-------------+-----------------------------------------+
| manila      | Value                                   |
+-------------+-----------------------------------------+
| adminURL    | http://172.18.198.55:8786/v1/20787a7b...|
| region      | RegionOne                               |
| publicURL   | http://172.18.198.55:8786/v1/20787a7b...|
| internalURL | http://172.18.198.55:8786/v1/20787a7b...|
| id          | 82cc5535aa444632b64585f138cb9b61        |
+-------------+-----------------------------------------+

+-------------+-----------------------------------------+
| manilav2    | Value                                   |
+-------------+-----------------------------------------+
| adminURL    | http://172.18.198.55:8786/v2/20787a7b...|
| region      | RegionOne                               |
| publicURL   | http://172.18.198.55:8786/v2/20787a7b...|
| internalURL | http://172.18.198.55:8786/v2/20787a7b...|
| id          | 2e8591bfcac4405fa7e5dc3fd61a2b85        |
+-------------+-----------------------------------------+
```

デフォルトでは、manila API サービスは、**tcp6** のポート **8786** でのみリッスンします。これは、IPv4 と IPv6 の両方をサポートします。

manila は複数の設定ファイルを使用します。これらは **/var/lib/config-data/puppet-generated/manila/** に保存されます。

```
api-paste.ini
manila.conf
policy.json
rootwrap.conf
rootwrap.d

./rootwrap.d:
share.filters
```

manila を root 以外のサービスアカウントで実行するように設定し、システム管理者のみが変更できるようにファイルのパーミッションを変更することを推奨します。manila では、管理者のみが設定ファイルに書き込みを行うことができ、サービスは **manila** グループのグループメンバーシップを介して読み

取りのみを行うことを想定します。サービスアカウントパスワードが含まれるため、他のユーザーがこれらのファイルを読み取り可能であってはいけません。

注記

root ユーザーのみが、**rootwrap.conf** の **manila-rootwrap** の設定および **rootwrap.d/share.filters** の共有ノードの **manila-rootwrap** コマンドフィルターに書き込みできる必要があります。

17.2. MANILA のネットワークおよびセキュリティーモデル

manila の共有ドライバーは、共有操作を管理するためにバックエンドに設定可能な Python クラスです。これらの一部はベンダー固有のものになります。バックエンドは、**manila-share** サービスのインスタンスです。manila は、多くの異なるストレージシステム用の共有ドライバーを持ち、商用ベンダーおよびオープンソースソリューションの両方をサポートします。各共有ドライバーは、1 つまたは複数のバックエンドモード (**共有用サーバー** および **共有用サーバーなし**) をサポートします。管理者は、**driver_handles_share_servers** を使用して **manila.conf** でモードを指定して選択します。

共有サーバーは、共有ファイルシステムをエクスポートする論理 Network Attached Storage (NAS) サーバーです。今日のバックエンドストレージシステムは高機能で、異なる OpenStack プロジェクト間でデータパスとネットワークパスを分離することができます。

manila 共有ドライバーによってプロビジョニングされる共有サーバーは、作成するプロジェクトユーザーに属する分離ネットワーク上に作成されます。**共有用サーバー** モードは、ネットワークプロバイダーに応じて、フラットネットワークまたはセグメント化されたネットワークのいずれかで設定できます。

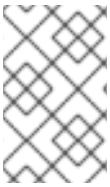
異なるモードのそれぞれのドライバーが、同じハードウェアを使用することができます。選択したモードによっては、設定ファイルでより多くの設定情報を提供する必要がある場合があります。

17.3. ファイル共有バックエンドモード

各共有ドライバーは、少なくとも 1 つの利用可能なドライバーモードをサポートします。

- 共有用サーバー (**driver_handles_share_servers = True**): 共有ドライバーは共有サーバーを作成し、共有サーバーのライフサイクルを管理します。
- 共有用サーバーなし (**driver_handles_share_servers = False**): ファイル共有サーバーの存在に依存するのではなく、管理者 (共有ドライバーではなく) がネットワークインターフェイスでベアメタルストレージを管理します。

共有用サーバーなしモード: このモードでは、ドライバーは共有サーバーを設定しないため、新しいネットワークインターフェイスを設定する必要はありません。ドライバーが管理するストレージコントローラーには必要なネットワークインターフェイスがすべて含まれていることを前提とします。ドライバーは、以前共有サーバーを作成せずに直接共有を作成します。このモードで稼働するドライバーを使用してファイル共有を作成するには、manila では、いずれかのプライベート共有ネットワークを作成する必要はありません。



注記

共有用サーバーなしモード では、manila は、ファイル共有をエクスポートするネットワークインターフェイスがすべてのプロジェクトからすでに到達可能であることを想定します。

共有用サーバーなし モードでは、共有ドライバーは共有サーバーのライフサイクルを処理しません。管理者は、プロジェクトの分離を提供するのに必要なストレージ、ネットワーク、およびその他のホスト側の設定を処理することが想定されます。このモードでは、管理者はファイル共有をエクスポートするホストとしてストレージを設定できます。OpenStack クラウド内のすべてのプロジェクトは、共通のネットワークパイプを共有します。分離がない場合、セキュリティーおよび QoS (Quality of Service) に影響を及ぼす可能性があります。共有用サーバーを処理しない共有ドライバーを使用する場合には、クラウドユーザーは、ファイルシステムの最上位ディレクトリー上のツリーウォークにより、信頼できないユーザーが自分のファイル共有にアクセスできないことを確認することはできません。パブリッククラウドでは、1つのクライアントですべてのネットワーク帯域幅が使用される可能性があるため、管理者はそうならないように注意する必要があります。ネットワークのバランシングは、必ずしも OpenStack ツールだけを使用しなければならない訳ではなく、どの方法ででも実行できます。

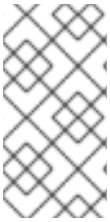
共有用サーバーモード: このモードでは、ドライバーは共有用サーバーを作成して、既存の OpenStack ネットワークにプラグインすることができます。manila は、新規共有用サーバーが必要であるかどうかを判別し、共有ドライバーが必須共有用サーバーを作成するのに必要なすべてのネットワーク情報を提供します。

共有用サーバーを処理するドライバーモードでファイル共有を作成する場合、ユーザーはファイル共有をエクスポートすることを期待する共有ネットワークを提供する必要があります。manila は、このネットワークを使用して、このネットワーク上の共有用サーバーのネットワークポートを作成します。

共有用サーバー および **共有用サーバーなし** バックエンドモードの両方で、**セキュリティーサービス** を設定することができます。ただし、**共有用サーバーなし** バックエンドモードでは、管理者はホストで必要な認証サービスを手動で設定する必要があります。**共有用サーバー** モードでは、manila は、生成する共有用サーバーでユーザーが識別するセキュリティーサービスを設定することができます。

17.4. MANILA のネットワーク要件

manila は、**flat**、**GRE**、**VLAN**、**VXLAN** など、さまざまなネットワーク種別と統合できます。



注記

manila はネットワーク情報をデータベースに格納するだけで、実際のネットワークはネットワークプロバイダーによって提供されます。Manila は、OpenStack Networking サービス (neutron) およびスタンドアロンの事前設定されたネットワークの使用をサポートしています。

共有用サーバー バックエンドモードでは、共有ドライバーにより、ファイル共有ネットワークごとに共有用サーバーを作成および管理します。このモードは、2 つのバリエーションに分割できます。

- **共有用サーバー** バックエンドモードのフラットネットワーク
- **共有用サーバー** バックエンドモードのセグメント化されたネットワーク

ユーザーは、OpenStack Networking (neutron) サービスからのネットワークおよびサブネットを使用して、ファイル共有ネットワークを作成することができます。管理者が **StandAloneNetworkPlugin** の使用を決定した場合、ユーザーはネットワーク情報を提供する必要がありません。管理者が設定ファイルでこれを事前設定するためです。

注記

一部の共有ドライバーにより起動するファイル共有用サーバーは、Compute サービスで作成した Compute サーバーです。これらのドライバーの一部は、ネットワークプラグインに対応していません。

ファイル共有ネットワークが作成されると、manila はネットワークプロバイダーによって決定されたネットワーク情報 (ネットワーク種別、セグメンテーション ID (ネットワークがセグメンテーションを使用する場合)、およびネットワークの割り当て元の IP ブロック (CIDR 表記)) を取得します。

ユーザーは、AD、LDAP ドメイン、Kerberos レルムなどのセキュリティー要件を指定するセキュリティーサービスを作成できます。manila は、セキュリティーサービスで呼び出されるホストが、共有用サーバーが作成されるサブネットから到達可能であることを前提としています。これにより、このモードが使用されるケースが制限されます。

注記

共有ドライバーによっては、すべてのタイプのセグメンテーションに対応していない場合があります。詳細は、使用しているドライバーの仕様を参照してください。

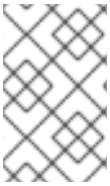
17.5. MANILA を使用したセキュリティーサービス

manila は、ネットワーク認証プロトコルと統合することで、ファイル共有へのアクセスを制限できます。各プロジェクトには、クラウドの keystone 認証ドメインとは別に機能する独自の認証ドメインを持つことができます。このプロジェクトドメインを使用して、manila を含む OpenStack クラウド内で実行されるアプリケーションに承認 (AuthZ) サービスを提供することができます。利用可能な認証プロトコルには、LDAP、Kerberos、および Microsoft Active Directory 認証サービスが含まれます。

17.6. セキュリティーサービスの概要

ファイル共有を作成してエクスポート場所を取得した後、ユーザーにはファイル共有をマウントし、ファイルを操作するパーミッションがありません。ユーザーは、新規共有へのアクセスを明示的に取得する必要があります。

クライアントの認証および承認 (authN/authZ) は、セキュリティーサービスと共に実行できます。LDAP、Kerberos、または Microsoft Active Directory が共有ドライバーおよびバックエンドでサポートされている場合、manila はこれらを使用できます。



注記

場合によっては、NetApp, EMC などのセキュリティーサービスの 1 つを明示的に指定する必要があります。Windows ドライバーには、CIFS プロトコルを使用したファイル共有の作成に Active Directory が必要です。

17.7. セキュリティーサービスの管理

セキュリティーサービス は、Active Directory ドメインや Kerberos ドメインなど、特定の共有ファイルシステムプロトコルのセキュリティーゾーンを定義するオプションのセットを抽象化する manila エンティティーです。セキュリティーサービスには、manila が指定のドメインに参加させるサーバーを作成するために必要な全情報が含まれます。

ユーザーは、API を使用してセキュリティーサービスの作成、更新、表示、および削除を行うことができます。セキュリティーサービスは、以下の前提条件に基づいて設計されています。

- プロジェクトが、セキュリティーサービスの詳細を提供する。
- 管理者がセキュリティーサービスに対応する (このようなセキュリティーサービスのサーバー側を設定する)。
- manila API 内で、**security_service** は **share_networks** に関連付けられる。

- 共有ドライバーは、セキュリティーサービスのデータを使用して、新規に作成された共有用サーバーを設定する。

セキュリティーサービスの作成時に、以下のいずれかの認証サービスを選択できます。

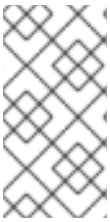
- **LDAP**: Lightweight Directory Access Protocol。IP ネットワークを通じて分散ディレクトリー情報サービスにアクセスし、維持するためのアプリケーションプロトコル。
- **Kerberos**: セキュアでないネットワーク上で通信するノードが、安全な方法で互いに ID を証明できるように、チケットベースで機能するコンピューターネットワーク認証プロトコル
- **Active Directory**: Windows ドメインネットワーク向けに Microsoft が開発したディレクトリーサービス。LDAP、Microsoft バージョンの Kerberos、および DNS を使用します。

manila では、以下のオプションを使用してセキュリティーサービスを設定することができます。

- プロジェクトネットワーク内で使用される DNS IP アドレス
- セキュリティーサービスの IP アドレスまたはホスト名
- セキュリティーサービスのドメイン
- プロジェクトが使用するユーザーまたはグループ名
- ユーザー名を指定した場合は、ユーザーのパスワード

既存のセキュリティーサービスエンティティーには、ファイル共有ネットワークエンティティーを関連付けることができます。このエンティティーは、manila に、ファイル共有グループのセキュリティーおよびネットワーク設定を通知します。指定したファイル共有ネットワークに対するセキュリティーサービスのリストを確認し、ファイル共有ネットワークから関連付けを解除することもできます。

ファイル共有の所有者として、管理者およびユーザーは、IP アドレス、ユーザー、グループ、または TLS 証明書を使用した認証でアクセスルールを作成することにより、ファイル共有へのアクセスを管理できます。認証方法は、設定して使用する共有ドライバーやセキュリティーサービスによって異なります。その後、特定の認証サービスを使用するようにバックエンドを設定することができます。これにより、manila および keystone を使用せずにクライアントと機能することができます。

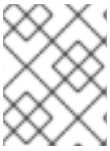


注記

さまざまな認証サービスが、さまざまな共有ドライバーによりサポートされます。さまざまなドライバーによりサポートされる機能に関する詳細は、https://docs.openstack.org/manila/latest/admin/share_back_ends_feature_support_m を参照してください。

あるドライバーが特定の認証サービスをサポートするからと言って、任意の共有ファイルシステムプロトコルで設定できるという訳ではありません。サポート対象の共有ファイルシステムプロトコルは、NFS、CEPHFS、CIFS、GlusterFS、および HDFS です。特定のドライバーおよびそのセキュリティーサービスの設定に関する情報は、ドライバーベンダーのドキュメントを参照してください。

一部のドライバーは、セキュリティーサービスをサポートしますが、上記のセキュリティーサービスをサポートしないドライバーもあります。たとえば、NFS または CIFS 共有ファイルシステムプロトコルを使用する汎用ドライバーは、IP アドレスによる認証方法のみをサポートします。



注記

ほとんどの場合、CIFS 共有ファイルシステムプロトコルをサポートするドライバーは、Active Directory を使用して、ユーザー認証でアクセスを管理するように設定できます。

- GlusterFS プロトコルをサポートするドライバーは、TLS 証明書を使用する認証で使用できます。
- IP アドレスを使用した NFS プロトコル認証をサポートするドライバーは、唯一のサポートされているオプションです。
- HDFS 共有ファイルシステムプロトコルは、NFS アクセスを使用するので、IP アドレスを使用して認証するように設定することもできます。

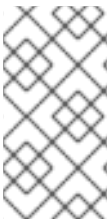
実稼働環境用の manila デプロイメントの推奨設定は、CIFS 共有プロトコルを使用してファイル共有を作成し、それを Microsoft Active Directory ディレクトリーサービスに追加することです。この設定では、集中データベースと、Kerberos および LDAP アプローチを統合するサービスが提供されます。

17.8. ファイル共有へのアクセスの制御

ユーザーは、作成するファイル共有にアクセスできる特定のクライアントを指定できます。keystone サービスにより、個別ユーザーが作成したファイル共有は、作成したユーザーおよび同じプロジェクト内のユーザーだけに表示されます。Manila により、ユーザーはパブリックに表示されるファイル共有を作成することができます。所有者がアクセス権を付与した場合、これらのファイル共有は、他の OpenStack プロジェクトに属するユーザーのダッシュボードに表示されます。ネットワーク上でアクセス可能な場合、これらの共有をマウントできる場合もあります。

ファイル共有の作成時に、キー **--public** を使用して、他のプロジェクトに対してファイル共有をパブリックにし、ファイル共有のリストにその共有を表示し、詳細情報を表示します。

policy.json ファイルに従って、ファイル共有の所有者として、管理者およびユーザーは、アクセスルールを作成することにより、ファイル共有へのアクセスを管理できます。**manila access-allow**、**manila access-deny**、および **manila access-list** コマンドを使用すると、指定されたファイル共有へのアクセスを許可、拒否、およびリスト表示することができます。



注記

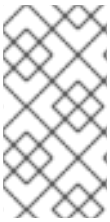
manila は、ストレージシステムのエンドツーエンド管理を提供しません。したがって、別途バックエンドシステムを承認されていないアクセスから保護する必要があります。その結果、アウトオブバンドアクセスを取得することでバックエンドストレージデバイスが侵害された場合、manila API が提供する保護は損なわれる可能性があります。

ファイル共有が作成されたままの状態では、それに関連付けられたデフォルトのアクセスルールとマウントパーミッションはありません。これは、使用中のエクスポートプロトコルのマウント設定で確認できます。たとえば、NFS コマンド **exportfs** またはストレージの **/etc/exports** ファイルがあり、各リモートファイル共有を制御し、アクセスできるホストを定義します。誰もファイル共有をマウントできない場合、これは空です。リモート CIFS サーバーでは、設定を表示する **net conf list** コマンドがあります。**hosts deny** パラメーターは、共有ドライバーにより **0.0.0.0/0** に設定する必要があります。これは、いずれのホストも、ファイル共有のマウントを拒されることを意味します。

manila を使用して、サポートされるファイル共有へのアクセスレベルのいずれかを指定して、ファイル共有へのアクセスを許可または拒否できます。

- **rw**: 読み取りおよび書き込み (RW) アクセス。これはデフォルト値です。

- **ro**: 読み取り専用 (RO) アクセス

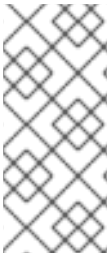


注記

RO アクセスレベルは、パブリックなファイル共有で役立ちます。この場合、管理者が特定のエディターまたはコントリビューターに対して読み取りおよび書き込み (RW) アクセスを提供し、残りのユーザー (ビューアー) には読み取り専用 (RO) アクセスを提供します。

サポートされる認証方法のいずれかも指定する必要があります。

- **IP**: インスタンスの認証に IP アドレスを使用します。IP アクセスは、適切に形成された IPv4 アドレスまたは IPv6 アドレスで指定可能なクライアント、または CIDR 表記で指定されるサブネットに提供できます。
- **cert**: インスタンスの認証に TLS 証明書を使用します。TLS アイデンティティーを **IDENTKEY** として指定します。有効な値は、証明書の共通名 (CN) の 64 文字までの任意の文字列です。
- **user**: 指定したユーザーまたはグループ名で認証します。有効な値は、特殊文字を含む 4 から 32 文字の長さの英数字の文字列です。



注記

サポートされる認証方法は、使用する共有ドライバー、セキュリティーサービス、および共有ファイルシステムプロトコルにより異なります。サポート対象の共有ファイルシステムプロトコルは、MapRFS、CEPHFS、NFS、CIFS、および HDFS です。サポートされるセキュリティーサービスは、LDAP、Kerberos プロトコル、または Microsoft Active Directory サービスです。

アクセスルール (ACL) がファイル共有に対して正しく設定されていることを確認するには、そのパーミッションをリスト表示できます。

注記

ファイル共有にセキュリティーサービスを選択する場合は、共有ドライバーが利用可能な認証方法を使用してアクセスルールを作成できるかどうかを考慮する必要があります。サポートされるセキュリティーサービスは、LDAP、Kerberos、および Microsoft Active Directory です。

17.9. 共有種別のアクセス制御

共有種別は、管理者の定義する **サービスの種別** で、プロジェクトに表示される説明および **追加仕様** と呼ばれるプロジェクトに表示されないキー/値ペアのリストで設定されます。**manila-scheduler** は追加仕様を使用してスケジューリングの決定を行い、ドライバーはファイル共有の作成を制御します。

管理者は共有種別の作成や削除が可能で、さらに、共有種別に manila 内での意味を持たせる追加仕様を管理することもできます。プロジェクトでは共有種別をリスト表示でき、それらを使用して新規ファイル共有を作成することができます。共有種別は **public** および **private** として作成できます。これは共有種別の可視性のレベルで、他のプロジェクトがリストに共有種別を表示し、新規ファイル共有を作成するのに使用できるかどうかを定義するものです。

デフォルトでは、共有種別は public として作成されます。共有種別を作成する時に **--is_public** パラメーターを使用して **False** に設定し、共有種別をプライベートに設定します。これにより、他のプロジェクトがリストに共有種別を表示したり、それを使用して新たなファイル共有を作成するのを防ぐこ

とができます。一方、**public** の共有種別は、クラウド内のすべてのプロジェクトで利用できます。

manila により、管理者はプロジェクトの **private** の共有種別へのアクセスを許可または拒否することができます。指定したプライベート共有種別のアクセスに関する情報を取得することもできます。

注記

追加仕様により、共有種別はユーザーがファイル共有を作成する前にバックエンドをフィルターまたは選択するのに役立つため、共有種別へのアクセスを使用すると、特定のバックエンドにクライアントを限定することができます。

たとえば、**admin** プロジェクトの管理者ユーザーは、**my_type** という名前のプライベート共有種別を作成し、これをリストで確認できます。以下のコンソールの例では、ログインとログアウトが省略され、現在ログインしているユーザーを示す環境変数が提供されます。

```
$ env | grep OS_
...
OS_USERNAME=admin
OS_TENANT_NAME=admin
...
$ manila type-list --all
+----+--------+-----------+-----------+---------------------------------+----------------------+
| ID | Name   | Visibility| is_default| required_extra_specs            | optional_extra_specs  |
+----+--------+-----------+-----------+---------------------------------+----------------------+
| 4..| my_type| private   | -         | driver_handles_share_servers:False| snapshot_support:True |
| 5..| default| public    | YES       | driver_handles_share_servers:True | snapshot_support:True |
+----+--------+-----------+-----------+---------------------------------+----------------------+
```

demo プロジェクトの **demo** ユーザーは種別をリスト表示できますが、**my_type** という名前のプライベート共有種別は表示されません。

```
$ env | grep OS_
...
OS_USERNAME=demo
OS_TENANT_NAME=demo
...
$ manila type-list --all
+----+--------+-----------+-----------+---------------------------------+----------------------+
| ID | Name   | Visibility| is_default| required_extra_specs            | optional_extra_specs |
+----+--------+-----------+-----------+---------------------------------+----------------------+
| 5..| default| public    | YES       | driver_handles_share_servers:True| snapshot_support:True|
+----+--------+-----------+-----------+---------------------------------+----------------------+
```

管理者は、プロジェクト ID が **df29a37db5ae48d19b349fe947fada46** の demo プロジェクトのプライベート共有種別へのアクセスを付与できます。

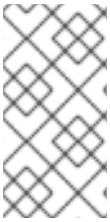
```
$ env | grep OS_
...
OS_USERNAME=admin
OS_TENANT_NAME=admin
...
$ openstack project list
+----------------------------------+------------------+
| ID                               | Name             |
```

```
+----------------------------------+------------------+
| ...                              | ...              |
| df29a37db5ae48d19b349fe947fada46 | demo             |
+----------------------------------+------------------+
$ manila type-access-add my_type df29a37db5ae48d19b349fe947fada46
```

その結果、**demo** プロジェクトのユーザーは、プライベートの共有種別を表示し、ファイル共有の作成で使用できます。

```
$ env | grep OS_
...
OS_USERNAME=demo
OS_TENANT_NAME=demo
...
$ manila type-list --all
+----+--------+-----------+-----------+---------------------------------+----------------------+
| ID | Name   | Visibility| is_default| required_extra_specs            | optional_extra_specs |
+----+--------+-----------+-----------+---------------------------------+----------------------+
| 4..| my_type| private   | -         | driver_handles_share_servers:False| snapshot_support:True |
| 5..| default| public    | YES       | driver_handles_share_servers:True | snapshot_support:True |
+----+--------+-----------+-----------+---------------------------------+----------------------+
```

指定されたプロジェクトのアクセスを拒否するには、**manila type-access-remove <share_type> <project_id>** を使用します。



注記

共有タイプの目的を示す例として、2 つのバックエンドがある状況を考えてみます。パブリックストレージとしての LVM と、プライベートストレージとしての Ceph です。このような場合には、特定のプロジェクトにアクセスを許可し、**user/group** 認証方法によりアクセスを制御できます。

17.10. ポリシー

Shared File Systems サービス API は、ロールベースのアクセス制御ポリシーで制御されます。これらのポリシーは、どのユーザーが特定の API にどのようにアクセスできるかを決定し、サービスの **policy.json** ファイルで定義されます。

注記

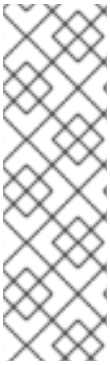
設定ファイル **policy.json** はどこにも配置できます。パス **/var/lib/config-data/puppet-generated/manila/etc/manila/policy.json** はデフォルトで想定されています。

manila への API 呼び出しを行うたびに、ポリシーエンジンは適切なポリシー定義を使用して、呼び出しを受け入れることができるかどうかを判断します。ポリシールールは、API 呼び出しを許可する状況を決定します。ルールが空の文字列 **""** の場合、**/var/lib/config-data/puppet-generated/manila/etc/manila/policy.json** ファイルは常にアクションが許可されるルールを持ちます。ユーザーロールまたはルールに基づくルール、ブール値式のルール。以下は、manila の **policy.json** ファイルのスニペットです。OpenStack リリース間で変更することが想定されます。

```
{
    "context_is_admin": "role:admin",
    "admin_or_owner": "is_admin:True or project_id:%(project_id)s",
    "default": "rule:admin_or_owner",
```

```
    "share_extension:quotas:show": "",
    "share_extension:quotas:update": "rule:admin_api",
    "share_extension:quotas:delete": "rule:admin_api",
    "share_extension:quota_classes": "",
}
```

ユーザーは、ポリシーで参照するグループおよびロールに割り当てられる必要があります。これは、ユーザー管理コマンドが使用されている場合にサービスによって自動的に行われます。



注記

/var/lib/config-data/puppet-generated/manila/etc/manila/policy.json への変更はすぐに有効になります。したがって、manila の稼働中に新規ポリシーを実装できます。ポリシーを手動で変更することは、予期しない副作用が発生する可能性があり、推奨されません。Manila ではデフォルトのポリシーファイルが提供されず、すべてのデフォルトポリシーがコードベース内にあります。**oslopolicy-sample-generator --config-file=var/lib/config-data/puppet-generated/manila/etc/manila/manila-policy-generator.conf** を実行することで、manila コードからデフォルトのポリシーを生成することができます。

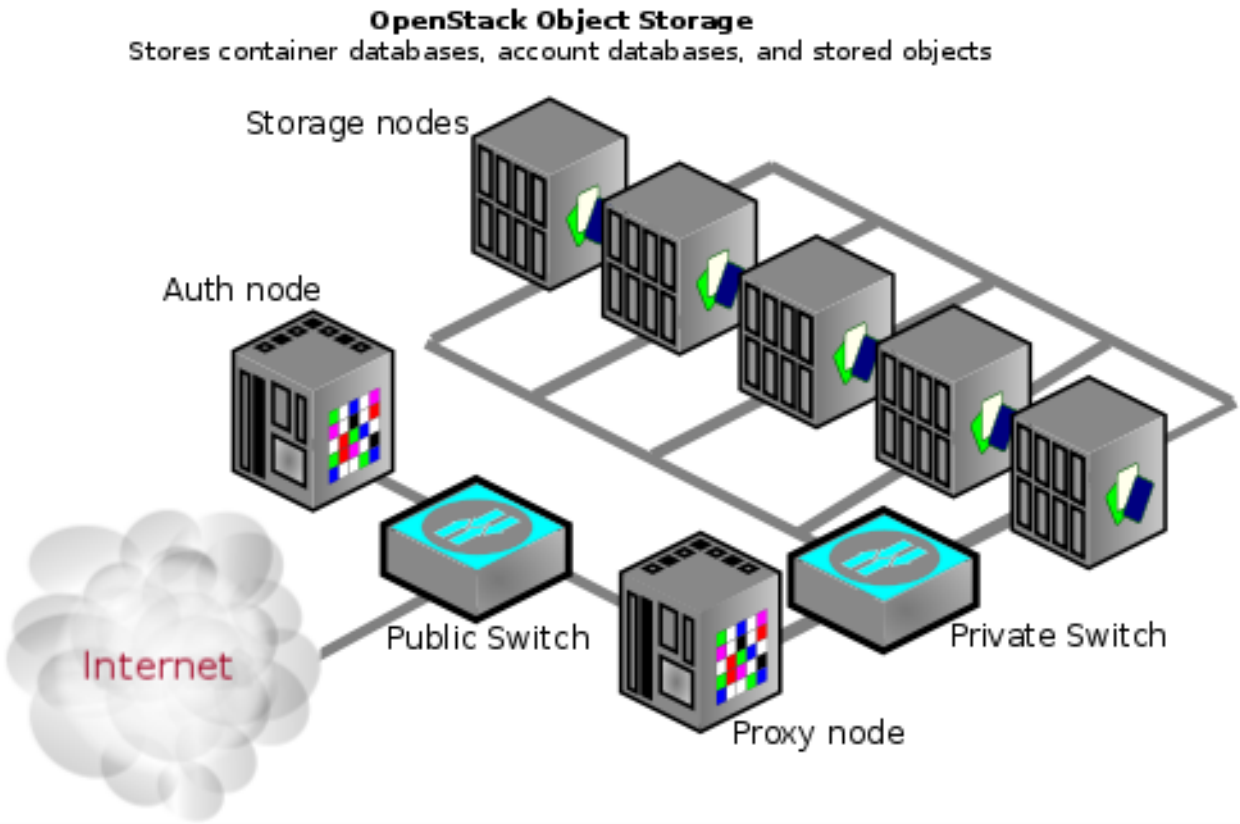
第18章 オブジェクトストレージ

Object Storage (swift) サービスは、HTTP 経由でデータを保存および取得します。オブジェクト (データの塊) は、組織的な階層に保存され、匿名の読み取り専用アクセス、ACL 定義のアクセス、または一時的なアクセスを提供するように設定できます。swift は、ミドルウェアを使用して実装される複数のトークンベースの認証メカニズムをサポートします。

アプリケーションは、業界標準の HTTP RESTful API を使用して、オブジェクトストレージにデータを保存し、取得します。バックエンドの swift コンポーネントは同じ RESTful モデルに従いますが、一部の API (耐久性を管理する API など) はクラスターに対してプライベートなままになります。

swift のコンポーネントは以下のプライマリーグループに分類されます。

- プロキシーサービス
- 認証サービス
- ストレージサービス
  - アカウントサービス
  - コンテナーサービス
  - オブジェクトサービス



注記

オブジェクトストレージのインストールは、インターネットに接続する必要はなく、パブリックスイッチ (組織の内部ネットワークインフラストラクチャーの一部) を持つプライベートクラウドにすることもできます。

18.1. ネットワークセキュリティー

swift のセキュリティー強化は、ネットワークコンポーネントのセキュリティー保護から始まります。詳細は、Networking の章を参照してください。

高可用性を確保する場合、rsync プロトコルがストレージサービスノード間でデータをレプリケートするために使用されます。さらに、プロキシーサービスは、クライアントエンドポイントとクラウド環境間でデータをリレーする際にストレージサービスと通信します。

注記

swift は、ノード間通信に暗号化または認証を使用しません。これは、swift がパフォーマンス上の理由でネイティブ rsync プロトコルを使用し、rsync 通信に SSH を使用しないためです。これが、アーキテクチャー図にプライベートスイッチまたはプライベートネットワーク ([V]LAN) が表示される理由です。このデータゾーンは、他の OpenStack データネットワークからも分離する必要があります。

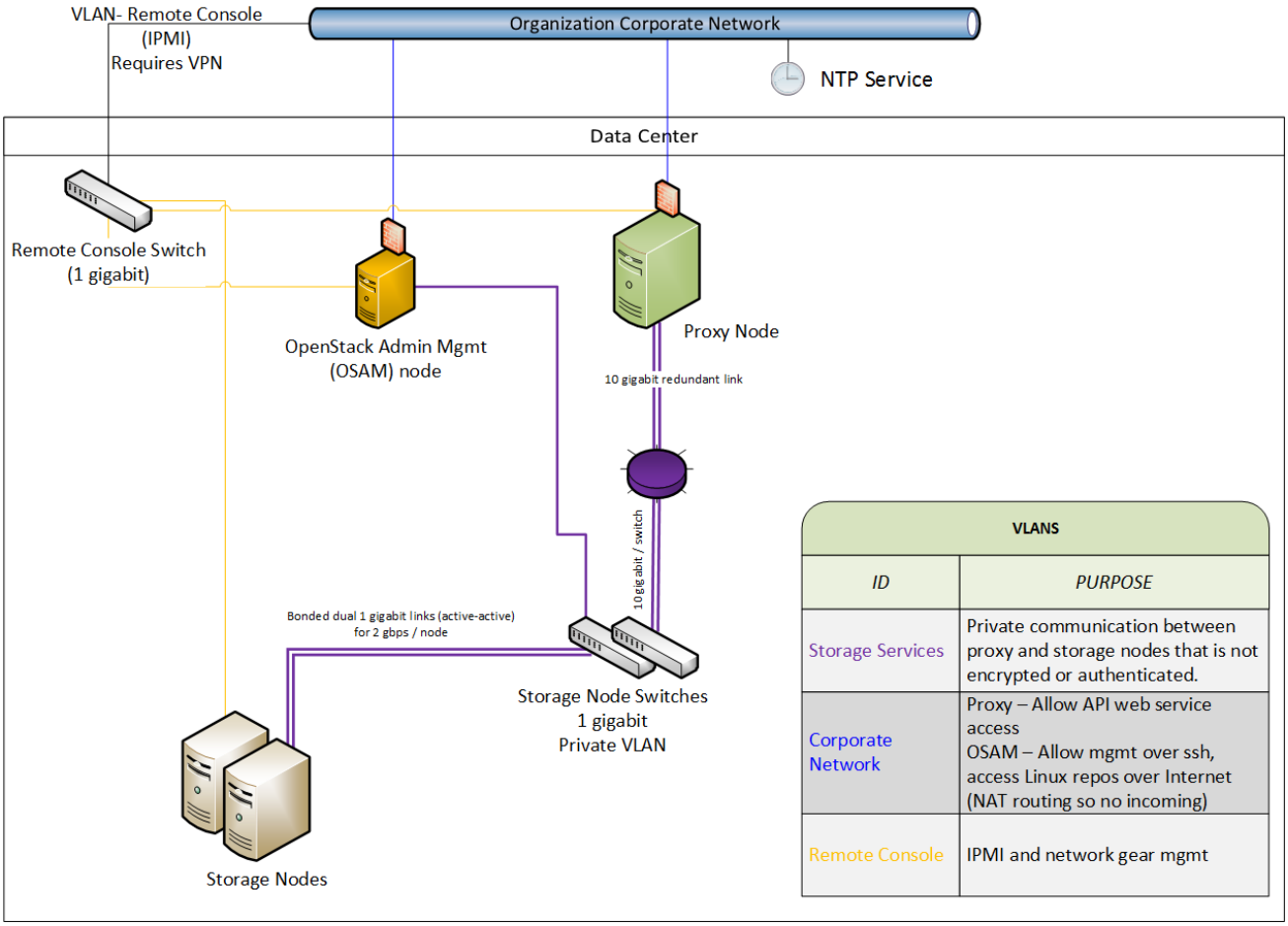
注記

データゾーン内のストレージノードには、プライベート (V)LAN ネットワークセグメントを使用します。

これには、プロキシーノードにデュアルインターフェイス (物理または仮想) が必要です。

- コンシューマーが到達できるパブリックインターフェイスとして 1 つのインターフェイス。
- ストレージノードにアクセスできるプライベートインターフェイスとしての別インターフェイス。

以下の図は、Object Storage ネットワークアーキテクチャーと管理ノード (OSAM) を使用する、可能なネットワークアーキテクチャーの 1 つを示しています。



18.2. 非 ROOT ユーザーとしてのサービスの実行

root (**UID 0**) 以外のサービスアカウントで実行するように swift を設定することを推奨します。1 つの推奨事項として、director によりデプロイされるように、ユーザー名を **swift** に、プライマリーグループを **swift** にすることです。Object Storage サービスには、 **proxy-server**、**container-server**、**account-server** が含まれます。

18.3. ファイル権限

/var/lib/config-data/puppet-generated/swift/etc/swift/ ディレクトリーには、リングトポロジーと環境設定に関する情報が含まれます。以下のパーミッションが推奨されます。

```
chown -R root:swift /var/lib/config-data/puppet-generated/swift/etc/swift/*
find /var/lib/config-data/puppet-generated/swift/etc/swift/ -type f -exec chmod 640 {} \;
find /var/lib/config-data/puppet-generated/swift/etc/swift/ -type d -exec chmod 750 {} \;
```

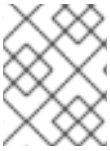
この制限は、root だけに設定ファイルの変更を許可しますが、**swift** グループのメンバーシップにより、サービスにそれらの読み取りを許可します。

18.4. ストレージサービスのセキュリティー保護

以下は、さまざまなストレージサービスのデフォルトリスニングポートです。

- アカウントサービス: **TCP/6002**
- コンテナーサービス: **TCP/6001**

- オブジェクトサービス: **TCP/6000**
- rsync: **TCP/873**



注記

rsync ではなく ssync が使用される場合、耐久性を維持するためにオブジェクトサービスポートが使用されます。

注記

認証はストレージノードで発生することはありません。これらのポートのいずれかでストレージノードに接続できる場合は、認証なしでデータにアクセスしたり変更したりすることができます。この問題を軽減するには、プライベートストレージネットワークの使用に関する前述の推奨事項に従ってください。

18.5. OBJECT STORAGE アカウントの用語

swift アカウントは、ユーザーアカウントまたは認証情報ではありません。定義は以下のとおりです。

- Swift アカウント: コンテナーのコレクションです (ユーザーアカウントや認証ではない)。使用する認証システムは、アカウントに関連付けられているユーザー、およびそのアクセス方法を決定します。
- swift コンテナー: オブジェクトのコレクション。コンテナーのメタデータは ACL で利用できます。ACL の使用は、使用される認証システムによって異なります。
- swift オブジェクト: 実際のデータオブジェクト。オブジェクトレベルの ACL もメタデータで利用できますが、使用する認証システムに依存します。

各レベルでは、ユーザーアクセスを制御する ACL があり、ACL は使用中の認証システムを基に解釈されます。最も一般的な認証プロバイダーの種別は Identity サービス (keystone) です。カスタム認証プロバイダーも利用できます。

18.6. プロキシーサービスのセキュリティー保護

プロキシーノードには、少なくとも 2 つのインターフェイス (物理または仮想) が必要です。1 つはパブリック、および 1 つはプライベート。パブリックインターフェイスを保護するには、ファイアウォールまたはサービスのバインディングを使用できます。パブリック向けサービスは、エンドポイントのクライアント要求を処理し、認証を行い、適切なアクションを実行する HTTP Web サーバーです。プライベートインターフェイスにはリスニングサービスは必要ありませんが、代わりにプライベートストレージネットワーク上のストレージノードへの発信接続を確立するために使用されます。

18.7. HTTP リッスンポート

director は、root 以外のユーザー (UID 0 以外) で実行するように Web サービスを設定します。**1024** より大きいポート番号を使用すると、Web コンテナーのいずれの部分も root として実行されなくなります。通常、HTTP REST API を使用する (そして自動認証を実行する) クライアントは、認証応答から必要な完全な REST API URL を取得します。OpenStack REST API により、クライアントはある URL に対して認証を行い、実際のサービス用に完全に異なる URL を使用するためにリダイレクトできます。たとえば、クライアントは **https://identity.cloud.example.org:55443/v1/auth** に対して認証を行い、認証キーおよび、**https://swift.cloud.example.org:44443/v1/AUTH_8980** のストレージ URL (プロキシーノードまたはロードバランサーの URL) を持つ応答を取得できます。

18.8. ロードバランサー

Apache を使用するオプションが実行できない場合や、パフォーマンス上の理由で TLS 作業をオフロードする場合は、専用のネットワークデバイスのロードバランサーを使用する場合があります。これは、複数のプロキシーノードを使用する場合に冗長性および負荷分散を提供する一般的な方法です。

TLS のオフロードを選択する場合は、ロードバランサーとプロキシーノード間のネットワークリンクがプライベート (V)LAN セグメント上にありるようにします。これにより、ネットワーク上の他のノード (セキュリティー侵害されている可能性がある) による非暗号化トラフィックの盗聴 (スニフィング) を防ぐことができます。このような侵害が発生した場合、攻撃者はエンドポイントクライアントまたはクラウド管理者の認証情報にアクセスし、クラウドデータにアクセスできるようになります。

使用する認証サービスは、エンドポイントクライアントへの応答で異なる URL を設定する方法を決定し、個別のプロキシーノードではなくロードバランサーを使用できるようにします。

18.9. OBJECT STORAGE の認証

Object Storage (swift) は WSGI モデルを使用して、一般的な機能拡張を提供するだけではなく、エンドポイントクライアントの認証にも使用されるミドルウェア機能を提供します。認証プロバイダーは、どのロールおよびユーザー種別が存在するかを定義します。従来のユーザー名およびパスワードの認証情報を使用するものもあれば、API キートークンやクライアント側の x.509 証明書を利用するものもあります。カスタムプロバイダーは、カスタムミドルウェアを使用して統合できます。

オブジェクトストレージには、デフォルトで 2 つの認証ミドルウェアモジュールが含まれています。これらのモジュールのいずれかを、カスタム認証ミドルウェアを開発するためのサンプルコードとして使用できます。

18.10. 保存されている SWIFT オブジェクトの暗号化

swift は、Barbican を統合して、保管されている (at-rest) オブジェクトを透過的に暗号化/復号化できます。at-rest 暗号化は、in-transit 暗号化とは異なり、ディスクに保管されている間にオブジェクトが暗号化されることを指します。

Swift はこれらの暗号化タスクを透過的に実行し、オブジェクトは swift にアップロードされる際には自動的に暗号化され、ユーザーに提供される際には自動的に復号化されます。この暗号化と復号化は、Barbican に保管されている同じ (対称) キーを使用して処理されます。

関連情報

- OpenStack Key Manager でシークレットを管理する

18.11. その他の項目

すべてのノードの **/var/lib/config-data/puppet-generated/swift/etc/swift/swift.conf** に、**swift_hash_path_prefix** 設定および **swift_hash_path_suffix** 設定があります。これらは、保存されるオブジェクトのハッシュ競合の可能性を低減し、あるユーザーが別のユーザーのデータを上書きするのを防ぐために提供されます。

この値は、最初に暗号的に安全な乱数ジェネレーターで設定し、すべてのノードで一貫性を持たせる必要があります。適切な ACL で保護し、データの喪失を回避するために、バックアップコピーを作成するようにします。

第19章 監視およびロギング

ログ管理は、OpenStack デプロイメントのセキュリティーステータスを監視する重要なコンポーネントです。ログは、OpenStack デプロイメントを設定するコンポーネントのアクティビティーに加えて、管理者、プロジェクト、およびインスタンスの BAU アクションに関する洞察を提供します。

ログは、予防的なセキュリティー活動および継続的なコンプライアンスアクティビティーのみならず、調査およびインシデントへの対応に関する貴重な情報ソースでもあります。たとえば、keystone アクセスログを分析すると、その他の関連情報に加えて、ログインの失敗、その頻度、送信元 IP、およびイベントが特定のアカウントに限定されるかどうかなどについて、警告が得られます。

director には、AIDE を使用した侵入検知機能、および keystone の CADF 監査が含まれています。詳細は、インフラストラクチャーおよび仮想化の強化 を参照してください。

19.1. モニタリングインフラストラクチャーの強化

集中ロギングシステムは、侵入者にとって価値のあるターゲットです。侵入に成功すると、イベントの記録を消去または改ざんできるからです。この点を考慮に入れて、モニタリングプラットフォームを強化することが推奨されます。さらに、機能停止またはサービス拒否 (DoS) 攻撃に備えて、フェイルオーバーの計画を作成すると共に、このシステムのバックアップを定期的に作成することを検討してください。

19.2. 監視するイベントの例

イベントの監視は、環境を保護し、リアルタイムの検出および応答を提供するより予防的な手段です。監視に役立つツールが複数存在します。OpenStack のデプロイメントでは、ハードウェア、OpenStack サービス、およびクラウドリソースの使用状況をモニターする必要があります。

本セクションでは、認識する必要のあるイベントの例について説明します。

重要

このリストがすべてを網羅している訳ではありません。実際のネットワークに適用される可能性のある追加のユースケースを考慮する必要があります。また、異常な動作を考慮する必要があります。

- ログが生成されないことを検出するのは、価値の高いイベントです。このようなギャップは、サービス障害や、攻撃を隠すために侵入者が一時的にログを停止したり、ログレベルを変更したりしたことを示す場合があります。
- スケジュールされていないイベントの開始や停止などのアプリケーションイベントは、セキュリティーと何らかの関係がある可能性があります。
- ユーザーのログインや再起動などの OpenStack ノードでのオペレーティングシステムイベントこれにより、システムの適切な使用方法と不適切な使用方法を区別するための、貴重な洞察が得られます。
- ネットワークブリッジがダウンする。これは、サービスが停止するリスクがあるため、対処を要するイベントです。
- コンピュートノードでの IPtable のフラッシュイベント、およびそれに伴うインスタンスへのアクセス不能

Identity サービス内のユーザー、プロジェクト、またはドメインの削除により使用されなくなったイン

スタンスからのセキュリティーリスクを軽減するには、システム内で通知を生成し、OpenStack コンポーネントに適切にこれらのイベントに応答させます (例: インスタンスの切断、攻撃を受けたボリュームの接続解除、CPU およびストレージリソースの再確保など)。

侵入検知ソフトウェア、ウイルス対策ソフトウェア、およびスウェアウェアの検出や削除ユーティリティーなどのセキュリティー監視は、攻撃または侵入がいつどのように発生したかを示すログを生成することができます。これらのツールは、OpenStack ノードにデプロイすると、保護のレイヤーを提供します。プロジェクトユーザーも、このようなツールをインスタンスで実行する必要があります。

第20章 プロジェクトのデータのプライバシー

OpenStack は、異なるデータ要件を持つプロジェクト間のマルチテナンシーをサポートするように設計されています。クラウドオペレーターは、適用可能なデータのプライバシーに関する懸念点と規制を考慮する必要があります。本章では、OpenStack デプロイメントにおけるデータ保持および破棄の要素に対応しています。

20.1. データ保持

データのプライバシーと分離は、過去数年にクラウド導入に対する主要なバリアとして、一貫して引用されています。クラウド内で誰がデータを所有するのか、クラウドオペレーターが究極的にこのデータの管理者として信頼できるかどうかの懸念が、これまで重大な問題となっていました。

特定の OpenStack サービスは、プロジェクトに属するデータおよびメタデータ、または参考プロジェクト情報にアクセスできます。たとえば、OpenStack クラウドに保存されるプロジェクトデータには、以下の項目が含まれます。

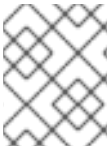
- Object Storage オブジェクト
- コンピュートインスタンスの一時ファイルシステムのストレージ
- コンピュートインスタンスのメモリー
- Block Storage ボリュームデータ
- Compute アクセス用の公開鍵
- Image サービスの仮想マシンイメージ
- インスタンスのスナップショット
- Compute の configuration-drive 拡張に渡されるデータ

OpenStack クラウドにより保存されるメタデータには、以下の項目が含まれます (このリストがすべてを網羅している訳ではありません)。

- 組織名
- ユーザーの本名
- 実行中のインスタンス、バケット、オブジェクト、ボリューム、およびその他のクォータ関連項目の数またはサイズ
- インスタンスの実行時間またはデータの保存時間
- ユーザーの IP アドレス
- コンピュートイメージのバンドル用に内部的に生成された秘密鍵

20.2. データの破棄

オペレーターは、廃棄の前に、組織の管理から外す前に、または再使用のために解放する前に、クラウドシステムメディア (デジタルおよび非デジタル) をサニタイズする必要があります。サニタイズ方法は、その情報の具体的なセキュリティードメインおよび機密性に対し、適切な強度および整合性のレベルを実装する必要があります。



注記

NIST Special Publication 800-53 Revision 4 に、本トピックが具体的に説明されています。

> The sanitization process removes information from the media such that the information cannot be retrieved or reconstructed. Sanitization techniques, including clearing, purging, cryptographic erase, and destruction, prevent the disclosure of information to unauthorized individuals when such media is reused or released for disposal.

クラウドオペレーターは、データ破棄およびサニタイズに関する一般的なガイドラインを開発する場合は、以下を考慮してください (NIST の推奨セキュリティー制御に基づく)。

- メディアサニタイズおよび破棄アクションを追跡、文書化、および検証する。
- サニタイズ機器と手順をテストして、適切なパフォーマンスを確認する。
- ポータブル、リムーバブルストレージデバイスデバイスをクラウドインフラストラクチャーに接続する前に、これらのデバイスをサニタイズする。
- サニタイズができないクラウドシステムメディアを破棄する。

その結果、OpenStack のデプロイメントで以下のプラクティスを対処する必要があります (一例)。

- セキュアなデータイレイジャー
- インスタンスメモリーのスクラブ
- Block Storage ボリュームデータ
- コンピュートインスタンスの一時ストレージ
- ベアメタルサーバーのサニタイズ

20.2.1. 安全に消去されないデータ

OpenStack の一部のデータは削除される可能性がありますが、上記の NIST 規格のコンテキストでは安全に消去されません。通常、これは、データベースに保存されている上記のメタデータと情報の多くまたはすべてに適用できます。これは、自動バキュームおよび定期的な空き領域の消去のためのデータベースやシステム設定で修正される可能性があります。

20.2.2. インスタンスメモリーのスクラブ

さまざまなハイパーバイザーに固有のことは、インスタンスメモリーの取り扱いです。一般に、インスタンスの削除時もしくは作成時に、またはその両方で、ハイパーバイザーはベストエフォートでメモリーのスクラブを行うと考えられますが、この動作は Compute では定義されません。

20.3. CINDER ボリュームデータの暗号化

OpenStack のボリューム暗号化機能の使用を強く推奨します。詳細は、下記データの暗号化セクションのボリュームの暗号化で説明されています。この機能が使用される場合、暗号鍵を安全に削除することでデータの破棄が実行されます。エンドユーザーは、ボリュームの作成時にこの機能を選択できますが、管理者はまず 1 度限りのボリューム暗号化機能の設定を実行する必要があることに注意してください。

OpenStack のボリューム暗号化機能が使用されていない場合、他のアプローチは通常有効にすることが困難になります。バックエンドプラグインが使用されている場合は、独立した暗号化方法や標準以外の上書きソリューションある可能性があります。OpenStack Block Storage へのプラグインでは、さまざまな方法でデータを保管します。多くのプラグインはベンダーや技術に固有のものですが、その他はファイルシステム (LVM または ZFS など) に関するより DIY 的なソリューションです。データを安全に破棄する方法は、プラグイン、ベンダー、およびファイルシステムによって異なります。

一部のバックエンド (ZFS など) は、データの公開を防ぐためにコピーオンライトをサポートします。このような場合、書き込まれていないブロックから読み取ると、常にゼロを返します。他のバックエンド (LVM など) はこれをネイティブにサポートしていない可能性があるため、cinder プラグインは、ブロックをユーザーに渡す前に以前に書き込まれたブロックを上書きします。選択したボリュームバックエンドが提供する保証を確認することや、提供されない保証についてどのような修復が利用できるかを確認することが重要です。

20.4. IMAGE サービスの削除遅延機能

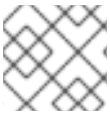
Image サービスには削除遅延機能があり、定義された期間イメージの削除を保留します。この動作がセキュリティー上の問題である場合は、この機能を無効にすることを検討してください。**glance-api.conf** ファイルを編集して **delayed_delete** オプションを **False** に設定すると、この機能を無効にすることができます。

20.5. COMPUTE のソフト削除機能

Compute にはソフト削除機能があり、定義した期間、削除されるインスタンスをソフト削除状態にすることができます。この期間は、インスタンスを復元することができます。ソフト削除機能を無効にするには、**/var/lib/config-data/puppet-generated/nova_libvirt/etc/nova/nova.conf** ファイルを編集し、**reclaim_instance_interval** オプションを空のままにします。

20.6. ベアメタルプロビジョニングのセキュリティー強化

ベアメタルプロビジョニングインフラストラクチャーの場合、一般的にはベースボード管理コントローラー (BMC)、特に IPMI のセキュリティー強化を検討する必要があります。たとえば、プロビジョニングネットワーク内のこれらのシステムを分離し、デフォルト以外の強力なパスワードを設定し、不要な管理機能を無効にします。詳細は、これらのコンポーネントのセキュリティーの強化に関するベンダーのガイダンスを参照してください。



注記

可能な場合は、レガシーに代えて Redfish ベースの BMC の評価を検討してください。

20.7. ハードウェアの特定

サーバーをデプロイする場合、攻撃者のサーバーと区別するための信頼できる方法が常にあるとは限りません。この機能は、ある程度ハードウェア/BMC に依存する可能性がありますが、通常は立証可能な手段はサーバーに組み込まれていないようです。

20.8. データの暗号化

プロジェクトデータがディスクのどこに保存されていても、あるいはネットワークを通じて転送されていても、実装者にはデータを暗号化するためのオプションがあります。たとえば、以下で説明されている OpenStack ボリューム暗号化機能です。これは、プロバイダーに送信する前にユーザーが自分のデータを暗号化する一般的な推奨事項を上回るものです。

プロジェクトの代わりにデータを暗号化することの重要性は、攻撃者がプロジェクトデータにアクセスできるというプロバイダーによって想定されるリスクに非常に関係します。政府、ポリシー、私的な契約、またはパブリッククラウドプロバイダーの私的な契約に関する法的な要件があります。プロジェクトの暗号化ポリシーを選択する前に、リスク評価と法的アドバイスを取得することを検討してください。

インスタンスまたはオブジェクトごとの暗号化が、降順、プロジェクト、ホスト、およびクラウドアグリゲーションごとよりも優先されます。この推奨事項は、実装の複雑性と困難度とは逆です。現時点で、一部のプロジェクトでは、プロジェクトごとの粗い粒度での暗号化を実装することが困難、または不可能な場合があります。実装者は、プロジェクトデータの暗号化に深刻な考慮事項を与えます。

多くの場合、データの暗号化は、単にキーを廃棄することで、プロジェクトやインスタンスごとのデータを確実に破棄する機能に貢献します。このようにすると、信頼できる安全な方法でこれらのキーを破棄することが非常に重要になります。

ユーザーのデータを暗号化する機会が存在します。

- Object Storage オブジェクト
- ネットワークデータ

20.8.1. ボリュームの暗号化

OpenStack のボリューム暗号化機能は、プロジェクトごとにプライバシーをサポートします。以下の機能がサポートされます。

- Dashboard またはコマンドラインインターフェイスから開始された暗号化されたボリューム種別の作成および使用
- 暗号化を有効にし、暗号化アルゴリズムやキーサイズなどのパラメーターを選択します。
- iSCSI パケットに含まれるボリュームデータの暗号化
- 元のボリュームが暗号化されている場合、暗号化されたバックアップをサポートします。
- Dashboard には、ボリューム暗号化のステータスが表示されます。ボリュームが暗号化されていること、およびアルゴリズムやキーサイズなどの暗号化パラメーターが含まれます。
- 鍵管理サービスとのインターフェイス

20.8.2. Object Storage オブジェクト

Object Storage (swift) は、ストレージノードに保管されているオブジェクトデータのオプション暗号化をサポートします。オブジェクトデータの暗号化は、承認されていないユーザーがディスクへの物理アクセスを取得している場合に、ユーザーのデータが読み取られるリスクを軽減することを目的としています。

保管されているデータの暗号化は、プロキシーサーバー WSGI パイプラインに含まれる可能性のあるミドルウェアにより実装されます。この機能は swift クラスターに内部にあり、API 経由で公開されません。クライアントは、swift サービス内のこの機能によりデータが暗号化されていることを認識しません。内部で暗号化されたデータは swift API 経由でクライアントに返されません。

swift で保管されている間、以下のデータは暗号化されます。

- オブジェクトコンテンツ (例: オブジェクト **PUT** 要求ボディーのコンテンツ)。
- ゼロ以外のコンテンツを持つオブジェクトのエンティティータグ (**ETag**)。

- すべてのカスタムユーザーオブジェクトのメタデータ値。たとえば、 **PUT** または **POST** リクエストで、**X-Object-Meta-** の接頭辞が付けられたヘッダーを使用して送信されるメタデータなどです。

上記のリストに含まれていないデータまたはメタデータは暗号化されません。以下に例を示します。

- アカウント、コンテナー、およびオブジェクト名
- アカウントおよびコンテナーのカスタムユーザーメタデータの値
- すべてのカスタムユーザーメタデータ名
- オブジェクトコンテンツ種別の値
- オブジェクトサイズ
- システムメタデータ

20.8.3. Block Storage のパフォーマンスおよびバックエンド

オペレーティングシステムを有効にすると、Intel と AMD の両方のプロセッサーで利用可能なハードウェアアクセラレーション機能を使用して、OpenStack Volume Encryption のパフォーマンスを向上させることができます。

OpenStack のボリューム暗号化機能は、ホスト上の **dm-crypt** またはネイティブの **QEMU** 暗号化サポートを使用してボリュームデータを保護します。Red Hat は、暗号化ボリュームを作成する際に **LUKS** ボリューム暗号化タイプを使用することを推奨します。

20.8.4. ネットワークデータ

コンピュートノードのプロジェクトデータは IPsec または他のトンネルで暗号化される可能性があります。これは、OpenStack では一般的または標準ではありませんが、興味のある実装者が利用可能なオプションです。同様に、ネットワーク上で転送されるため、暗号化されたデータは暗号化された状態のままです。

20.9. キー管理

プロジェクトデータのプライバシーに関する頻出する懸念に対処するために、OpenStack コミュニティーでは、データの暗号化をより広範囲で行うことに大きな関心が寄せられています。クラウドに保存する前にエンドユーザーがデータを暗号化するのは比較的簡単です。メディアファイルやデータベースアーカイブなどのプロジェクトオブジェクトに関して、実行可能なパスがあります。一部では、クライアント側の暗号化は、今後の使用のためにデータを復号化するために、鍵の提示などのクライアントの連携を必要とする仮想化技術で保持されるデータの暗号化に使用されます。

barbican を使用すると、プロジェクトがデータをシームレスに暗号化し、キー管理でユーザーに負担をかけることなくデータにアクセスすることができます。OpenStack の一部として暗号化およびキー管理サービスを提供することにより、保管データのセキュリティー確保の採用が促進され、データのプライバシーや誤用に関するお客様の懸念に対処するのに役立ちます。

ボリューム暗号化機能は、Key Manager サービス (バービカン) などのキー管理サービスを使用しており、キーの作成とセキュリティー保護された保存を行います。

第21章 インスタンスのセキュリティーの管理

仮想化環境でインスタンスを実行する利点の1つは、通常ベアメタルへのデプロイ時に利用できないセキュリティー制御の新しい機会です。OpenStack デプロイメントの情報セキュリティーを向上させる特定のテクノロジーを仮想化スタックに適用できます。セキュリティー要件が強固な運用者は、これらの技術のデプロイを検討する必要がある場合がありますが、すべての状況に該当する訳ではありません。場合によっては、規定されたビジネス要件により、クラウドでの使用にテクノロジーを除外できる場合があります。同様に、一部の技術は、動作状態などのインスタンスデータを検査しますが、システムのユーザーにとって望ましくない可能性があります。

本章では、これらのテクノロジーと、インスタンスまたは基盤のノードのセキュリティーを強化するために使用できる状況について説明します。プライバシーの懸念事項も詳しく説明します。これには、データパススルー、イントロスペクション、またはエントロピーソースが含まれます。

21.1. インスタンスへのエントロピーの提供

エントロピーとは、インスタンスで利用可能なランダムデータの質とソースのことです。暗号技術は一般的にランダム性に依存しており、エントロピーのプールから抽出する必要があります。エントロピーの枯渇は、暗号技術に必要なランダム性をサポートするのに十分なエントロピーをインスタンスが得られない場合に起こります。エントロピーの枯渇は、一見すると関係のないものとして現れることがあります。たとえば、ブート時間が遅くなるのは、インスタンスが SSH 鍵の生成を待機することが原因である可能性があります。また、エントロピーが枯渇すると、クラウドのユーザーがインスタンス内の質の悪いエントロピーソースを使用することになり、クラウド上で動作するアプリケーションの安全性が低下します。

高品質のエントロピー源をインスタンスに提供するためには、インスタンスをサポートするのに十分な数のハードウェア乱数生成器 (HRNG) がクラウドに必要です。日常的な運用であれば、最新の HRNG は 50 ~ 100 台の Compute ノードをサポートするのに十分なエントロピーを生成することができます。高帯域の HRNG は、より多くのノードを扱うことができます。十分なエントロピーを確保するためには、クラウドのアプリケーション要件を特定する必要があります。

VirtIO RNG は、デフォルトで **/dev/urandom** をエントロピーソースとして使用する乱数生成器で、起動時にインスタンスでエントロピーが枯渇しないようにします。また、デプロイメント全体でエントロピーを配布する方法を提供するために、HRNG またはエントロピー収集デーモン (EGD) などのツールを使用するように設定することもできます。Virtio RNG デバイスは、インスタンスのデフォルトで有効になっています。インスタンスで Virtio RNG デバイスを無効にするには、インスタンスフレーバーで **hw_rng:allowed** を **False** に設定する必要があります。

21.2. ノードへのインスタンスのスケジューリング

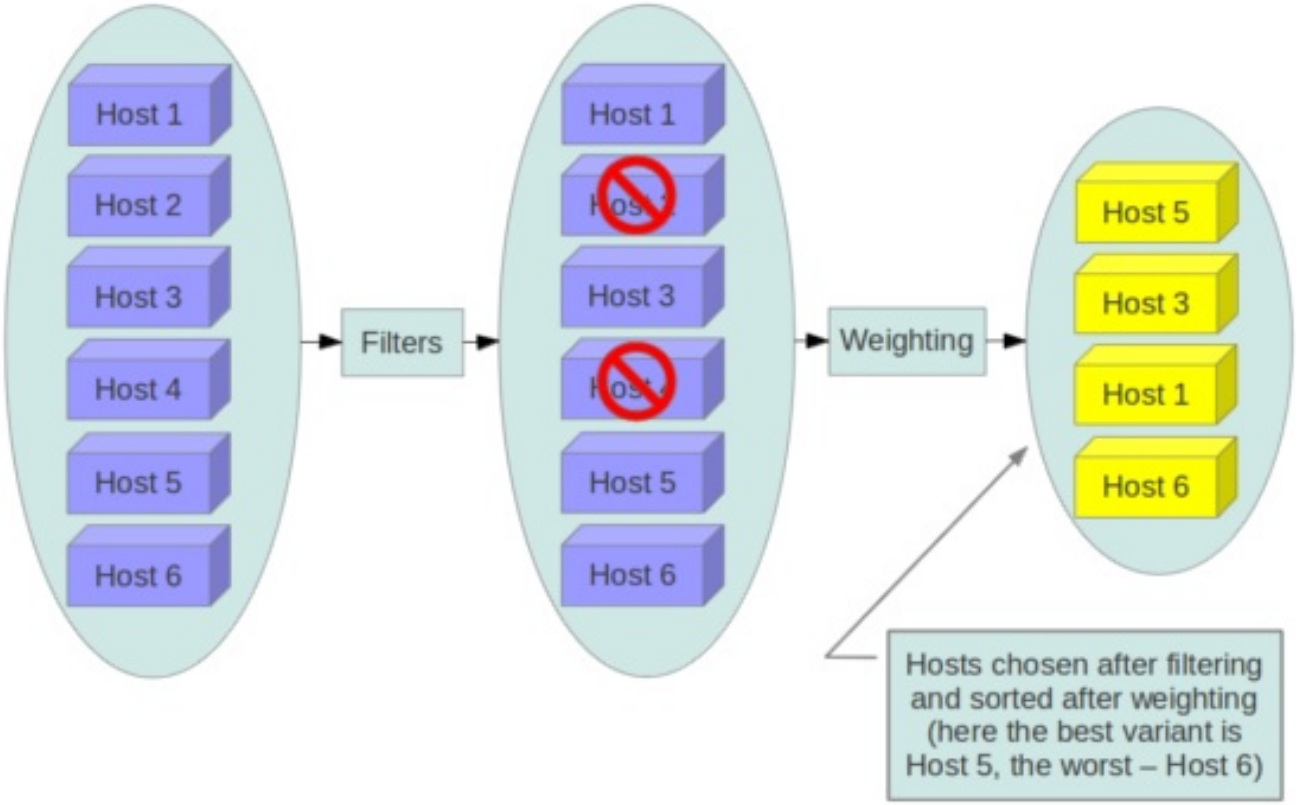
インスタンスを作成する前に、イメージのインスタンス化のホストを選択する必要があります。この選択は、コンピュートおよびボリューム要求のディスパッチ方法を決定する **nova-scheduler** により行われます。

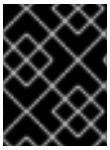
FilterScheduler は Compute のデフォルトスケジューラーですが、他のスケジューラーが存在します。この機能は、**フィルターヒント** と協力して、インスタンスの開始場所を決定します。ホスト選択のこのプロセスにより、管理者はさまざまなセキュリティーおよびコンプライアンス要件を満たすことができます。データの分離が主な懸念である場合は、可能な場合は同じホストにプロジェクトインスタンスを配置することができます。逆に、インスタンスの可用性やフォールトトレランスの理由で、できるだけ異なるホストにインスタンスを配置することができます。

フィルタースケジューラーは、以下の主要なカテゴリーに分類されます。

- **リソースベースのフィルター**: ハイパーバイザーホストセットのシステムリソース使用状況に基づいてインスタンスの配置を決定し、RAM、IO、CPU 使用率などの空きまたは使用されるプロパティーで起動することができます。
- **イメージベースのフィルター**: 仮想マシンのオペレーティングシステムや使用するイメージの種別など、使用されるイメージメタデータに基づいて、インスタンスの作成を委譲します。
- **環境ベースのフィルター**: 特定の IP 範囲内、アベイラビリティーゾーン間、または別のインスタンスと同じホストなど、外部の詳細に基づいてインスタンスの配置を決定します。
- **カスタム基準**: 信頼やメタデータの解析などの、ユーザーまたは管理者が提供する基準に基づいて、インスタンスの作成を委譲します。

複数のフィルターを一度に適用できます。たとえば、**ServerGroupAffinity** フィルターは、特定のホストセットのメンバーにインスタンスが作成されていることを確認し、**ServerGroupAntiAffinity** フィルターは同じインスタンスが別の特定ホストセットで作成されないことを確認します。これら 2 つのフィルターは通常同時に有効になり、互いに競合しないことに注意してください。特定のプロパティーの値をチェックし、いずれも同時に true にできないためです。





重要

ユーザーが提供するオブジェクトまたは操作可能なオブジェクト (メタデータなど) を解析するフィルターを無効にすることを検討してください。

21.3. 信頼できるイメージの使用

クラウド環境では、ユーザーは事前にインストール済みのイメージまたは自身がアップロードするイメージと連携します。いずれの場合も、使用するイメージが改ざんされていないことを確認できる必要

があります。イメージを検証する機能は、セキュリティーに関する基本的な必須要件です。イメージのソースから使用される宛先まで、信頼チェーンが必要です。これは、信頼できるソースから取得したイメージを署名し、使用する前に署名を確認して実行できます。検証されたイメージを取得し、作成する各種の方法について以下に説明し、続いてイメージ署名の検証機能について説明します。

21.4. イメージの作成

Red Hat OpenStack Image Service (glance)にイメージを作成し、アップロードする方法は、イメージの 作成および管理 を参照してください。セキュリティーを強化するために信頼できるイメージを使用し、組織の強化ガイドラインを使用してさらに保護します。環境用のイメージは、以下のいずれかの方法で取得できます。

インスタンスメディアのダウンロード

信頼できるソースからブートメディアを取得するには、公式の Red Hat ソースからイメージをダウンロードし、検証に SHA256SUM を使用します。

ISO からのイメージの作成

インストールプロセスでイメージを作成する方法は、Red Hat Enterprise Linux 9 イメージの作成 を参照してください。

Image Builder の使用

disk-image-builder を使用して、OpenStack 内で目的に必要なコンポーネントのみを持つ最小限のシステムを生成することができます。**disk-image-builder** でカスタムイメージを作成する方法は、RHEL システムイメージのカスタマイズ を 参照してください。

21.5. イメージの署名の確認

イメージの署名検証を有効にして、Compute サービス (nova) がインスタンスを開始する前に、Image Service (glance) のイメージに不正な変更が含まれていないことを確認できます。この機能を有効にすると、マルウェアやセキュリティーの脆弱性が含まれる可能性のある新しいインスタンスが起動できなくなります。

前提条件

- Red Hat OpenStack Platform director 環境がインストールされている。
- スタックとしてディレクターにログインしています。

手順

1. Heat テンプレートで、**VerifyGlanceSignatures** パラメーターに **True** の値を設定して、インスタンスの署名検証を有効にします。

```
parameter_defaults:
  VerifyGlanceSignatures: True
```

2. **VerifyGlanceSignatures** パラメーターの変更に使用するテンプレートが **openstack overcloud デプロイ** スクリプトに含まれていることを確認し、デプロイスクリプトを再実行します。

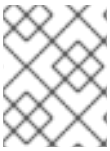
注記

署名していないイメージでインスタンスを作成すると、イメージは検証に失敗し、インスタンスは起動しません。イメージへの署名の詳細は、Image Service のイメージへの署名 を参照してください。

21.6. インスタンスの移行

OpenStack と基礎となる仮想化層は、OpenStack ノード間のイメージのライブマイグレーションを提供するため、インスタンスのダウンタイムなしに、コンピュートノードのローリングアップグレードをシームレスに実行できます。ただし、ライブマイグレーションでは、大きなリスクも伴います。リスクを理解するため、ライブマイグレーション中に実行される手順の概要を以下に示します。

1. 移行先ホストでインスタンスを起動する
2. メモリーを転送する
3. ゲストを停止してディスクの同期する
4. 状態を遷移する
5. ゲストを起動する



注記

コールドマイグレーション、サイズ変更、退避などの特定の操作により、インスタンスのデータをネットワークを通じて他のサービスに転送することになります。

21.6.1. ライブマイグレーションのリスク

ライブマイグレーションプロセスのさまざまな段階では、インスタンスのランタイムメモリーとディスクの内容は、ネットワークを通じてプレーンテキストで送信されます。したがって、ライブマイグレーションの使用時には、対応する必要があるリスクが複数含まれています。以下は、これらのリスクについて詳しく説明します (すべてを網羅している訳ではありません。)。

- サービス拒否 (DoS):移行プロセス中に何かが失敗した場合は、インスタンスが失われる可能性があります。
- データ公開:メモリーまたはディスクの転送は安全に処理する必要があります。
- データ操作:メモリーまたはディスク転送が安全に処理されない場合、攻撃者は移行中にユーザーデータを操作できます。
- コードインジェクション:メモリーまたはディスク転送が安全に処理されない場合、攻撃者は移行中にディスクまたはメモリーいずれかの実行ファイルを操作できます。

21.6.2. ライブマイグレーションの無効化

現時点で、OpenStack ではデフォルトでライブマイグレーションが有効になっています。ライブマイグレーションは、デフォルトでは管理者専用タスクであるため、ユーザーはこの操作を開始できず、管理者 (信頼されると考えられる) しか開始できません。**policy.json** ファイルに以下の行を追加して、ライブマイグレーションを無効にすることができます。

```
"compute_extension:admin_actions:migrate": "!",
"compute_extension:admin_actions:migrateLive": "!",
```

あるいは、TCP ポート **49152** から **49261** までブロックする、または、nova ユーザーにコンピュートホスト間でのパスワードなしの SSH アクセスが含まれないようにすると、ライブマイグレーションに失敗することが期待されます。

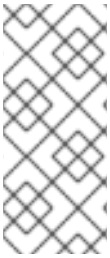
ライブマイグレーションの SSH 設定は大幅にロックダウンされていることに注意してください。新しいユーザーが作成され (nova_migration)、SSH キーはそのユーザーに制限され、許可されたネットワークでのみ使用できます。次に、ラッパースクリプトは実行可能なコマンド (例: libvirt ソケット上の netcat) を制限します。

21.6.3. 暗号化されたライブマイグレーション

ライブマイグレーションのトラフィックは、実行中のインスタンスのディスクおよびメモリーのコンテンツをプレーンテキストで転送し、デフォルトでは内部 API ネットワークでホストされています。

ライブマイグレーションを有効にするのに十分な要件 (アップグレードなど) がある場合、libvirtd はライブマイグレーション用に暗号化されたトンネルを提供できます。ただし、この機能は OpenStack Dashboard または nova-client コマンドで公開されず、libvirtd の手動設定でのみアクセスできます。ライブマイグレーションプロセスは、以下の手順概要に変わります。

1. インスタンスデータがハイパーバイザーから libvirtd にコピーされます。
2. 移行元ホストと移行先ホストの両方で libvirtd プロセス間で暗号化されたトンネルが作成されます。
3. 移行先の libvirtd ホストは、インスタンスを基礎となるハイパーバイザーにコピーして戻します。



注記

Red Hat OpenStack Platform 13 では、推奨されるアプローチはトンネル化された移行を使用することで、ceph をバックエンドとして使用する場合にデフォルトで有効化されます。詳細は、https://docs.openstack.org/nova/queens/configuration/config.html#libvirt.live_migrat を参照してください。

21.7. モニタリング、アラート、およびレポート

インスタンスは、ホスト間で複製可能なサーバーイメージです。したがって、物理ホストと仮想ホスト間と同様にロギングを適用することが推奨されます。ホストやデータへのアクセスイベント、ユーザーの追加および削除、権限の変更、要件で規定されるその他のイベントなどのオペレーティングシステムやアプリケーションイベントをログに記録する必要があります。結果をログアグリゲーターにエクスポートすることを検討してください。ログアグリゲーターは、ログイベントを収集し、解析用に関連付け、参照または今後のアクション用に保管します。これを実行するための一般的なツールの 1 つは、ELK スタックまたは Elasticsearch、Logstash、および Kibana です。

注記

これらのログは、定期的に確認したり、ネットワークオペレーションセンター (NOC) によって実行されるライブビュー内で監視する必要もあります。

さらに、どのイベントがその後にアクションのレスポンダーに送信されるアラートのトリガーになるかを把握する必要があります。

関連情報

- 監視ツール設定ガイド

21.8. 更新およびパッチ

ハイパーバイザーは、独立した仮想マシンを実行します。このハイパーバイザーは、オペレーティングシステム内または直接 (ベアメタルと呼ばれる) ハードウェア上で実行できます。ハイパーバイザーの更新は、仮想マシンに伝播されません。たとえば、デプロイメントが KVM を使用し、CentOS 仮想マシンのセットがある場合、KVM への更新では CentOS 仮想マシンで実行されているものは更新されません。

仮想マシンの明確な所有権を所有者に割り当て、その所有者が仮想マシンの強化、デプロイメント、および継続する機能を実施することを検討してください。また、更新を定期的にデプロイする計画を作成し、最初に実稼働環境と類似する環境でテストする必要があります。

21.9. ファイアウォールおよびインスタンスプロファイル

最も一般的なオペレーティングシステムには、追加のセキュリティー層用のホストベースのファイアウォールが含まれています。インスタンスはできるだけ少ないアプリケーションを実行すべきで (可能な場合、単一目的のインスタンスの観点で)、インスタンス上で実行されているすべてのアプリケーションは、アプリケーションがアクセスする必要のあるシステムリソース、それの実行に必要な権限の最低レベル、および仮想マシンから送受信されると予想されるネットワークトラフィックを決定するために、プロファイリングする必要があります。この予想されるトラフィックは、SSH または RDP などの必要なロギングおよび管理通信と共に、許可されるトラフィックとしてホストベースのファイアウォールに追加する必要があります。その他のトラフィックは、すべてファイアウォール設定で明示的に拒否する必要があります。

Linux インスタンスでは、上記のアプリケーションプロファイルを、**audit2allow** などのツールと併用して、ほとんどの Linux ディストリビューションで機密システム情報をさらに保護する SELinux ポリシーを構築することができます。SELinux は、ユーザー、ポリシー、およびセキュリティーコンテキストの組み合わせを使用して、アプリケーションの実行に必要なリソースを区分し、必要のない他のシステムリソースから分離します。

注記

Red Hat OpenStack Platform では、OpenStack サービス用にカスタマイズされるポリシーと共に、デフォルトで SELinux が有効化されます。必要に応じて、これらのポリシーを定期的に確認することを検討してください。

21.10. セキュリティーグループ

OpenStack は、特定のプロジェクトのインスタンスに厚い防御を追加するために、ホストとネットワークの両方にセキュリティーグループを提供します。これらは、ポート、プロトコル、およびアドレスに基づいて着信トラフィックを許可または拒否するため、ホストベースのファイアウォールと似ています。ただし、ホストベースのファイアウォールルールは、着信トラフィックと送信トラフィックの両方に適用できますが、セキュリティーグループルールは、着信トラフィックにのみ適用されます。ホストおよびネットワークセキュリティーグループルールが競合して、正当なトラフィックを拒否することもあります。使用されているネットワークに対して、セキュリティーグループが正しく設定されていることを確認することを検討してください。詳細は、本ガイドのセキュリティーグループを参照してください。

注記

特に無効にする必要がある場合を除き、セキュリティーグループおよびポートセキュリティーを有効なままにしておくべきです。厚い防御のアプローチを構築するには、インスタンスに細かな粒度のルールを適用することが推奨されます。

21.11. インスタンスのコンソールへのアクセス

デフォルトでは、インスタンスのコンソールには仮想コンソールからリモートでアクセスできます。これは、トラブルシューティングに役立ちます。Red Hat OpenStack Platform は、リモートコンソールアクセスに VNC を使用します。

- ファイアウォールルールを使用して VNC ポートをロックダウンすることを検討してください。デフォルトでは、**nova_vnc_proxy** は **6080** と **13080** を使用します。
- VNC トラフィックが TLS で暗号化されていることを確認します。director ベースのデプロイメントの場合は、**UseTLSTransportForVnc** で開始します。

21.12. 証明書の挿入

インスタンスに SSH 接続する必要がある場合は、作成時に必要な SSH 鍵をインスタンスに自動的に挿入するよう Compute を設定することができます。

関連情報

- イメージの作成。

第22章 メッセージキュー

メッセージキューサービスは、OpenStack でのプロセス間通信を容易にします。これは、以下のメッセージキューサービスのバックエンドを使用して行われます。

- RabbitMQ: Red Hat OpenStack Platform はデフォルトで RabbitMQ を使用します。
- Qpid

RabbitMQ と Qpid はいずれも Advanced Message Queuing Protocol (AMQP) フレームワークで、ピアツーピア通信用のメッセージキューを提供します。キューの実装は、通常、キューサーバーの集中的または分散プールとしてデプロイされます。

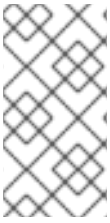
メッセージキューは、OpenStack デプロイメント全体にわたるコマンドおよび制御機能を効果的に容易化します。キューへのアクセスが許可されたら、それ以上の承認チェックは実行されません。キュー経由でアクセス可能なサービスは、実際のメッセージペイロード内のコンテキストおよびトークンを検証します。ただし、トークンは再使用でき、インフラストラクチャー内の他のサービスを承認できるため、トークンの有効期限に注意してください。

OpenStack は、メッセージ署名などのメッセージレベルの機密性をサポートしません。したがって、メッセージトランスポート自体をセキュア化し、認証する必要があります。高可用性 (HA) 設定には、キューツーキュー認証および暗号化を実行する必要があります。

22.1. メッセージングトランスポートのセキュリティー

AMQP ベースのソリューション (Qpid および RabbitMQ) は、TLS を使用したトランスポートレベルのセキュリティーをサポートします。

メッセージキューのトランスポートレベルの暗号化を有効にすることを検討してください。メッセージングクライアント接続に TLS を使用すると、メッセージングサーバーへの改ざんおよび盗聴から通信を保護することができます。以下のガイダンスは、一般的な 2 つのメッセージングサーバー用に TLS を通常どのように設定するかに関するものです。Qpid および RabbitMQ。メッセージングサーバーがクライアント接続を検証するために使用する信頼できる認証局 (CA) バンドルを設定する場合は、これをノードに使用される CA のみに制限することが推奨されます (内部管理 CA が推奨されます)。信頼された CA のバンドルは、承認されるクライアント証明書を決定し、TLS 接続の設定のクライアント-サーバー検証手順をスキップします。



注記

証明書と鍵ファイルをインストールする場合は、ファイルパーミッションが制限されており (たとえば、**chmod 0600** を使用)、メッセージングサーバーの他のプロセスやユーザーによる不正アクセスを阻止するために、その所有権がメッセージングサーバーのデーモンユーザーに制限されるようにしてください。

22.1.1. RabbitMQ サーバーの SSL 設定

以下の行をシステム全体の RabbitMQ 設定ファイル (通常は **/etc/rabbitmq/rabbitmq.config**) に追加する必要があります。

```
[
  {rabbit, [
    {tcp_listeners, [] },
    {ssl_listeners, [{"<IP address or hostname of management network interface>", 5671}] },
    {ssl_options, [{cacertfile,"/etc/ssl/cacert.pem"},
                   {certfile,"/etc/ssl/rabbit-server-cert.pem"},
```

```
                {keyfile,"/etc/ssl/rabbit-server-key.pem"},
                {verify,verify_peer},
                {fail_if_no_peer_cert,true}]}
  ]}
].
```

注記

SSL 以外のポートでリッスンしないように、**tcp_listeners** オプションは **[]** に設定します。サービスの管理ネットワークでのみリッスンするように、**ssl_listeners** オプションは制限する必要があります。

22.2. キュー認証およびアクセス制御

RabbitMQ および Qpid は、キューへのアクセスを制御する認証およびアクセス制御メカニズムを提供します。

Simple Authentication and Security Layer (SASL) は、インターネットプロトコルにおける認証およびデータセキュリティーのフレームワークです。RabbitMQ と Qpid はどちらも、簡単なユーザー名とパスワードより優れた SASL およびその他のプラグ可能な認証メカニズムを提供し、認証セキュリティーが強化されます。RabbitMQ は SASL をサポートしますが、現在 OpenStack のサポートでは、特定の SASL 認証メカニズムを要求することは許可されません。OpenStack の RabbitMQ サポートでは、暗号化されていない接続を使用したユーザー名およびパスワード認証、またはユーザー名/パスワードとセキュアな TLS 接続を確立するための X.509 クライアント証明書の組み合わせのどちらか可能です。

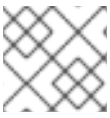
メッセージングキューへのクライアント接続用に、すべての OpenStack サービスノードで X.509 クライアント証明書を設定することを検討してください。また、可能であれば X.509 クライアント証明書を使用した認証を実施します (現在は Qpid のみ)。ユーザー名とパスワードを使用する場合は、キューへのアクセスをより細かく監査できるように、サービスおよびノードごとにアカウントを作成する必要があります。

デプロイメントの前に、キューサーバーが使用する TLS ライブラリーを考慮してください。Qpid は Mozilla の NSS ライブラリーを使用しますが、RabbitMQ は OpenSSL を使用する Erlang の TLS モジュールを使用します。

22.3. RABBITMQ 用 OPENSTACK サービスの設定

本項では、OpenStack サービス用の標準的な RabbitMQ 設定について説明します。

```
[DEFAULT]
rpc_backend = nova.openstack.common.rpc.impl_kombu
rabbit_use_ssl = True
rabbit_host = RABBIT_HOST
rabbit_port = 5671
rabbit_user = compute01
rabbit_password = RABBIT_PASS
kombu_ssl_keyfile = /etc/ssl/node-key.pem
kombu_ssl_certfile = /etc/ssl/node-cert.pem
kombu_ssl_ca_certs = /etc/ssl/cacert.pem
```



注記

RABBIT_PASS を適切なパスワードに置き換えます。

22.4. QPID 用 OPENSTACK サービスの設定

本項では、OpenStack サービス用の標準的な Qpid 設定について説明します。

```
[DEFAULT]
rpc_backend = nova.openstack.common.rpc.impl_qpid
qpid_protocol = ssl
qpid_hostname = <IP or hostname of management network interface of messaging server>
qpid_port = 5671
qpid_username = compute01
qpid_password = QPID_PASS
```

注記

QPID_PASS を適切なパスワードに置き換えます。

(オプション) Qpid で SASL を使用している場合は、以下を追加して、使用する SASL メカニズムを指定します。

```
qpid_sasl_mechanisms = <space separated list of SASL mechanisms to use for auth>
```

22.5. メッセージキュープロセスの分離およびポリシー

各プロジェクトは、メッセージを送信および消費するサービスを多数提供します。メッセージを送信する各バイナリーは、返信のみの場合、キューからのメッセージを消費します。

メッセージキューサービスのプロセスは、互いに、およびマシン上の他のプロセスから分離する必要があります。

22.6. 名前空間

Linux では、名前空間を使用してプロセスを独立したドメインに割り当てます。本ガイドのその他の部分では、システムの区分について詳細に説明します。

第23章 RED HAT OPENSTACK PLATFORM でのエンドポイントの保護

OpenStack クラウドと連携するプロセスは、API エンドポイントをクエリーすることで開始します。パブリックおよびプライベートのエンドポイントにはさまざまな課題がありますが、危険にさらされた場合に重大なリスクが生じる可能性のある価値の高いアセットがあります。

本章では、パブリックおよびプライベート向け API エンドポイント両方のセキュリティーを強化することを推奨しています。

23.1. 内部 API 通信

OpenStack は、パブリック向け内部管理エンドポイントとプライベート API エンドポイントの両方を提供します。デフォルトでは、OpenStack コンポーネントはパブリック向けに定義されたエンドポイントを使用します。適切なセキュリティードメイン内の API エンドポイントを使用するようにこれらのコンポーネントを設定することが推奨されます。内部管理エンドポイントにより keystone へのアクセスをさらに昇格できるため、これをさらに分離することが望ましい場合があります。

サービスは、OpenStack サービスカタログに基づいてそれぞれの API エンドポイントを選択します。これらのサービスは、リスト表示されるパブリックまたは内部 API エンドポイントの値に準拠していない可能性があります。これにより、内部管理トラフィックが外部 API エンドポイントにルーティングされる可能性があります。

23.2. IDENTITY サービスカタログでの内部 URL の設定

Identity サービスカタログは内部 URL を認識している必要があります。この機能はデフォルトでは使用されていませんが、設定を介して利用できます。さらに、この動作がデフォルトになると、予想される変更と前方互換性があるはずです。

異なるアクセスレベルがある場合に、設定したエンドポイントをネットワークレベルから分離することを検討します。管理エンドポイントは、クラウド管理者によるアクセスを目的としています。内部またはパブリックエンドポイントでは利用できない keystone 操作へのアクセスを昇格できるためです。内部エンドポイントは、クラウド内部 (例: OpenStack サービス) の使用を目的としており、通常はデプロイメントネットワーク外からはアクセスできません。パブリックエンドポイントは TLS 対応の必要があり、クラウドユーザーが操作するデプロイメント外からアクセスできる唯一の API エンドポイントです。

エンドポイントの内部 URL の登録は、director により自動化されます。詳細は、https://github.com/openstack/tripleo-heat-templates/blob/a7857d6dfcc875eb2bc611dd9334104c18fe8ac6/network/endpoints/build_endpoint_map を参照してください。

23.3. 内部 URL のアプリケーションの設定

特定の API エンドポイントを使用するように一部のサービスを強制することができます。したがって、別のサービスの API に接続する OpenStack サービスは、適切な内部 API エンドポイントにアクセスするように明示的に設定する必要があります。

各プロジェクトには、ターゲット API エンドポイントを定義する方法に一貫性がなくなる場合があります。OpenStack の今後のリリースでは、Identity サービスカタログの一貫した使用でこの不整合を解決します。

23.4. PASTE およびミドルウェア

OpenStack の API エンドポイントおよびその他の HTTP サービスの多くは、Python Paste Deploy ライブラリーを使用します。このライブラリーは、セキュリティーの観点からは、アプリケーションの設定を介して要求フィルターパイプラインの操作を可能にします。このチェーンの各要素はミドルウェアを指します。パイプラインでフィルターの順序を変更したり、追加のミドルウェアを追加したりすると、予測不可能なセキュリティー上の影響を及ぼす可能性があります。

一般的に、実装者は OpenStack のベース機能を拡張するためにミドルウェアを追加します。標準以外のソフトウェアコンポーネントを HTTP 要求パイプラインに追加することによって導入される潜在的な漏えいを慎重に考慮することを検討します。

23.5. API エンドポイントプロセスの分離およびポリシー

API エンドポイントプロセスを分離して、セキュリティーを高めることができます。公開されたセキュリティードメイン内の API エンドポイントを別のホストにデプロイし、隔離性を高めることを検討してください。

23.5.1. metadef API を制限するためのポリシーの設定

Red Hat OpenStack Platform (RHOSP) では、ユーザーはメタデータ定義 (metadef) API を使用してキー/値のペアおよびタグメタデータを定義することができます。現時点では、ユーザーが作成することのできる metadef 名前空間、オブジェクト、属性、リソース、またはタグの数に制限はありません。

metadef API により、情報が権限のないユーザーに漏えいする可能性があります。悪意のあるユーザーは制約がないことを悪用し、Image サービス (glance) のデータベースを無制限のリソースで埋め尽くすことができます。これにより、サービス拒否 (DoS) 型の攻撃を行うことができます。

Image サービスのポリシーは metadef API を制御します。ただし、metadef API のデフォルトのポリシー設定では、すべてのユーザーが metadef 情報を作成または読み取ることができます。metadef リソースへのアクセスは所有者だけに制限されている訳ではないため、内部インフラストラクチャーの詳細や顧客名などの秘匿すべき名前を持つ metadef リソースの情報が、悪意のあるユーザーに漏えいする可能性があります。

Image サービス (glance) をよりセキュアにするには、Red Hat OpenStack Platform (RHOSP) デプロイメントのデフォルトでは metadef 変更 API へのアクセスを管理者だけに制限します。

手順

1. クラウド管理者として新たな heat テンプレートの環境ファイルを作成し (例: **lock-down-glance-metadef-api.yaml**)、Image サービス metadef API のポリシーオーバーライドを含めます。

```
...
parameter_defaults:
  GlanceApiPolicies: {
        glance-metadef_default: { key: 'metadef_default', value: '' },
        glance-metadef_admin: { key: 'metadef_admin', value: 'role:admin' },
        glance-get_metadef_namespace: { key: 'get_metadef_namespace', value:
'rule:metadef_default' },
        glance-get_metadef_namespaces: { key: 'get_metadef_namespaces', value:
'rule:metadef_default' },
        glance-modify_metadef_namespace: { key: 'modify_metadef_namespace', value:
'rule:metadef_admin' },
        glance-add_metadef_namespace: { key: 'add_metadef_namespace', value:
'rule:metadef_admin' },
        glance-delete_metadef_namespace: { key: 'delete_metadef_namespace', value:
```

```
'rule:metadef_admin' },
        glance-get_metadef_object: { key: 'get_metadef_object', value: 'rule:metadef_default' },
        glance-get_metadef_objects: { key: 'get_metadef_objects', value: 'rule:metadef_default'
},
        glance-modify_metadef_object: { key: 'modify_metadef_object', value:
'rule:metadef_admin' },
        glance-add_metadef_object: { key: 'add_metadef_object', value: 'rule:metadef_admin' },
        glance-delete_metadef_object: { key: 'delete_metadef_object', value:
'rule:metadef_admin' },
        glance-list_metadef_resource_types: { key: 'list_metadef_resource_types', value:
'rule:metadef_default' },
        glance-get_metadef_resource_type: { key: 'get_metadef_resource_type', value:
'rule:metadef_default' },
        glance-add_metadef_resource_type_association: { key:
'add_metadef_resource_type_association', value: 'rule:metadef_admin' },
        glance-remove_metadef_resource_type_association: { key:
'remove_metadef_resource_type_association', value: 'rule:metadef_admin' },
        glance-get_metadef_property: { key: 'get_metadef_property', value:
'rule:metadef_default' },
        glance-get_metadef_properties: { key: 'get_metadef_properties', value:
'rule:metadef_default' },
        glance-modify_metadef_property: { key: 'modify_metadef_property', value:
'rule:metadef_admin' },
        glance-add_metadef_property: { key: 'add_metadef_property', value:
'rule:metadef_admin' },
        glance-remove_metadef_property: { key: 'remove_metadef_property', value:
'rule:metadef_admin' },
        glance-get_metadef_tag: { key: 'get_metadef_tag', value: 'rule:metadef_default' },
        glance-get_metadef_tags: { key: 'get_metadef_tags', value: 'rule:metadef_default' },
        glance-modify_metadef_tag: { key: 'modify_metadef_tag', value: 'rule:metadef_admin' },
        glance-add_metadef_tag: { key: 'add_metadef_tag', value: 'rule:metadef_admin' },
        glance-add_metadef_tags: { key: 'add_metadef_tags', value: 'rule:metadef_admin' },
        glance-delete_metadef_tag: { key: 'delete_metadef_tag', value: 'rule:metadef_admin' },
        glance-delete_metadef_tags: { key: 'delete_metadef_tags', value: 'rule:metadef_admin' }
 }

…
```

2. オーバークラウドのデプロイ時に **-e** オプションを使用して、ポリシーオーバーライドが含まれる環境ファイルをデプロイメントコマンドに追加します。

```
$ openstack overcloud deploy -e lock-down-glance-metadef-api.yaml
```

23.5.2. metadef API の有効化

以前にメタデータ定義 (metadef) API を制限している場合や、新規のデフォルトを緩和する場合は、metadef 変更ポリシーをオーバーライドして、ユーザーがそれぞれのリソースを更新できるようにすることが可能です。

重要

metadef API への書き込みアクセスに依存するユーザーを管理するクラウド管理者は、すべてのユーザーがこれらの API にアクセスできるようにすることが可能です。ただし、この種の設定では、顧客名や内部プロジェクト等の秘匿すべきリソース名が意図せず漏えいする可能性があります。すべてのユーザーに読み取りアクセスしか付与していない場合であっても、管理者はシステムを監査し、過去に作成したセキュリティー的に脆弱なリソースを識別する必要があります。

手順

1. クラウド管理者としてアンダークラウドにログインし、ポリシーオーバーライド用のファイルを作成します。以下に例を示します。

```
$ cat open-up-glance-api-metadef.yaml
```

2. すべてのユーザーが metadef API を読み取り/書き込みできるように、ポリシーオーバーライドファイルを設定します。

```
GlanceApiPolicies: {
    glance-metadef_default: { key: 'metadef_default', value: '' },
    glance-get_metadef_namespace: { key: 'get_metadef_namespace', value:
'rule:metadef_default' },
    glance-get_metadef_namespaces: { key: 'get_metadef_namespaces', value:
'rule:metadef_default' },
    glance-modify_metadef_namespace: { key: 'modify_metadef_namespace', value:
'rule:metadef_default' },
    glance-add_metadef_namespace: { key: 'add_metadef_namespace', value:
'rule:metadef_default' },
    glance-delete_metadef_namespace: { key: 'delete_metadef_namespace', value:
'rule:metadef_default' },
    glance-get_metadef_object: { key: 'get_metadef_object', value: 'rule:metadef_default' },
    glance-get_metadef_objects: { key: 'get_metadef_objects', value: 'rule:metadef_default' },
    glance-modify_metadef_object: { key: 'modify_metadef_object', value:
'rule:metadef_default' },
    glance-add_metadef_object: { key: 'add_metadef_object', value: 'rule:metadef_default' },
    glance-delete_metadef_object: { key: 'delete_metadef_object', value: 'rule:metadef_default'
},
    glance-list_metadef_resource_types: { key: 'list_metadef_resource_types', value:
'rule:metadef_default' },
    glance-get_metadef_resource_type: { key: 'get_metadef_resource_type', value:
'rule:metadef_default' },
    glance-add_metadef_resource_type_association: { key:
'add_metadef_resource_type_association', value: 'rule:metadef_default' },
    glance-remove_metadef_resource_type_association: { key:
'remove_metadef_resource_type_association', value: 'rule:metadef_default' },
    glance-get_metadef_property: { key: 'get_metadef_property', value: 'rule:metadef_default'
},
    glance-get_metadef_properties: { key: 'get_metadef_properties', value:
'rule:metadef_default' },
    glance-modify_metadef_property: { key: 'modify_metadef_property', value:
'rule:metadef_default' },
    glance-add_metadef_property: { key: 'add_metadef_property', value: 'rule:metadef_default'
},
    glance-remove_metadef_property: { key: 'remove_metadef_property', value:
```

```
'rule:metadef_default' },
    glance-get_metadef_tag: { key: 'get_metadef_tag', value: 'rule:metadef_default' },
    glance-get_metadef_tags: { key: 'get_metadef_tags', value: 'rule:metadef_default' },
    glance-modify_metadef_tag: { key: 'modify_metadef_tag', value: 'rule:metadef_default' },
    glance-add_metadef_tag: { key: 'add_metadef_tag', value: 'rule:metadef_default' },
    glance-add_metadef_tags: { key: 'add_metadef_tags', value: 'rule:metadef_default' },
    glance-delete_metadef_tag: { key: 'delete_metadef_tag', value: 'rule:metadef_default' },
    glance-delete_metadef_tags: { key: 'delete_metadef_tags', value: 'rule:metadef_default' }
  }
```

注記

すべての metadef ポリシーを設定する際に、**rule:metadeta_default** を使用する必要があります。

3. オーバークラウドのデプロイ時に **-e** オプションを使用して、デプロイメントコマンドに新しいポリシーファイルを追加します。

```
$ openstack overcloud deploy -e open-up-glance-api-metadef.yaml
```

23.6. HAPROXY の SSL/TLS の暗号およびルールの変更

オーバークラウドで SSL/TLS を有効にした場合は、HAProxy 設定で使用される SSL/TLS 暗号とルールを強化することを検討してください。SSL/TLS 暗号を強化することで、POODLE 脆弱性 などの SSL/TLS 脆弱性を回避できます。

1. **tls-ciphers.yaml** という名前の Heat テンプレート環境ファイルを作成します。

```
touch ~/templates/tls-ciphers.yaml
```

2. 環境ファイルで **ExtraConfig** フックを使用して、値を **tripleo::haproxy::ssl_cipher_suite** および **tripleo::haproxy::ssl_options** hieradata に適用します。

```
parameter_defaults:
  ExtraConfig:
    tripleo::haproxy::ssl_cipher_suite:: `DHE-RSA-AES128-CCM:DHE-RSA-AES256-
CCM:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-
AES128-CCM:ECDHE-ECDSA-AES256-CCM:ECDHE-ECDSA-AES128-GCM-
SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-
POLY1305:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-AES128-GCM-
SHA256:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-CHACHA20-POLY1305`

    tripleo::haproxy::ssl_options:: no-sslv3 no-tls-tickets
```

注記

暗号のコレクションは、改行せずに 1 行に記述します。

3. オーバークラウドをデプロイする際に、overcloud deploy コマンドで **tls-ciphers.yaml** 環境ファイルを含めます。

```
openstack overcloud deploy --templates \
```

```
...
-e /home/stack/templates/tls-ciphers.yaml
...
```

23.7. ネットワークポリシー

API エンドポイントは通常複数のセキュリティーゾーンにまたがるので、API プロセスの分離に特に注意を払う必要があります。たとえば、ネットワーク設計レベルでは、指定したシステムにのみアクセスを限定することを検討してください。詳細は、セキュリティーゾーンに関するガイダンスを参照してください。

慎重にモデル化することで、ネットワーク ACL および IDS テクノロジーを使用して、ネットワークサービス間の明示的なポイントツーポイント通信を適用することができます。重要なクロスドメインサービスとして、このタイプの明示的な適用が OpenStack のメッセージキューサービスで機能します。

ポリシーを適用するには、サービス、ホストベースのファイアウォール (iptables など)、ローカルポリシー (SELinux)、およびオプションでグローバルネットワークポリシーを設定できます。

23.8. 必須のアクセス制御

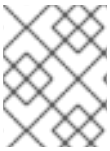
API エンドポイントプロセスは、互いに、およびマシン上の他のプロセスから分離する必要があります。これらのプロセスの設定は、Discretionary Access Controls (DAC) および Mandatory Access Controls (MAC) によってこれらのプロセスに制限される必要があります。これらの強化されたアクセス制御の目的は、API エンドポイントセキュリティー違反の封じ込めを支援することにあります。

23.9. API エンドポイントのレート制限

レート制限は、ネットワークベースのアプリケーションによって受信されるイベントの頻度を制御する手段です。堅牢なレート制限が存在しない場合、アプリケーションはさまざまなサービス拒否攻撃を受けやすくなる場合があります。これは特に、その性質上、同様の要求タイプや操作を高い頻度で受け入れるように設計されている API が該当します。

すべてのエンドポイント (特にパブリック) には、物理ネットワーク設計、レート制限プロキシー、または Web アプリケーションのファイアウォールを使用するなど、追加の保護レイヤーを設定することが推奨されます。

レート制限機能を設定して実装する際に、運用者は OpenStack クラウド内のユーザーとサービスの個々のパフォーマンスニーズについて慎重に計画して考慮することが重要です。



注記

Red Hat OpenStack Platform デプロイメントの場合、全サービスは負荷分散プロキシーの背後に置かれます。

第24章 フェデレーションの実装

警告

現時点では、Red Hat はフェデレーションをサポートしていません。これはテスト目的にのみ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。

24.1. RED HAT SINGLE SIGN-ON を使用した IDM でのフェデレーション

Red Hat Single Sign-On (RH-SSO) を使用して、OpenStack 認証 (authN) 用の IdM ユーザーをフェデレーションすることができます。フェデレーションにより、IdM ユーザーは OpenStack サービスに認証情報を公開せずに OpenStack Dashboard にログインすることができます。代わりに、Dashboard がユーザーの認証情報が必要な場合には、ユーザーを Red Hat Single Sign-On (RH-SSO) に転送し、そこで IdM 認証情報を入力できるようにします。これにより、RH-SSO はユーザーが正常に認証されたとして Dashboard に戻し、Dashboard はユーザーがプロジェクトにアクセスするのを許可します。

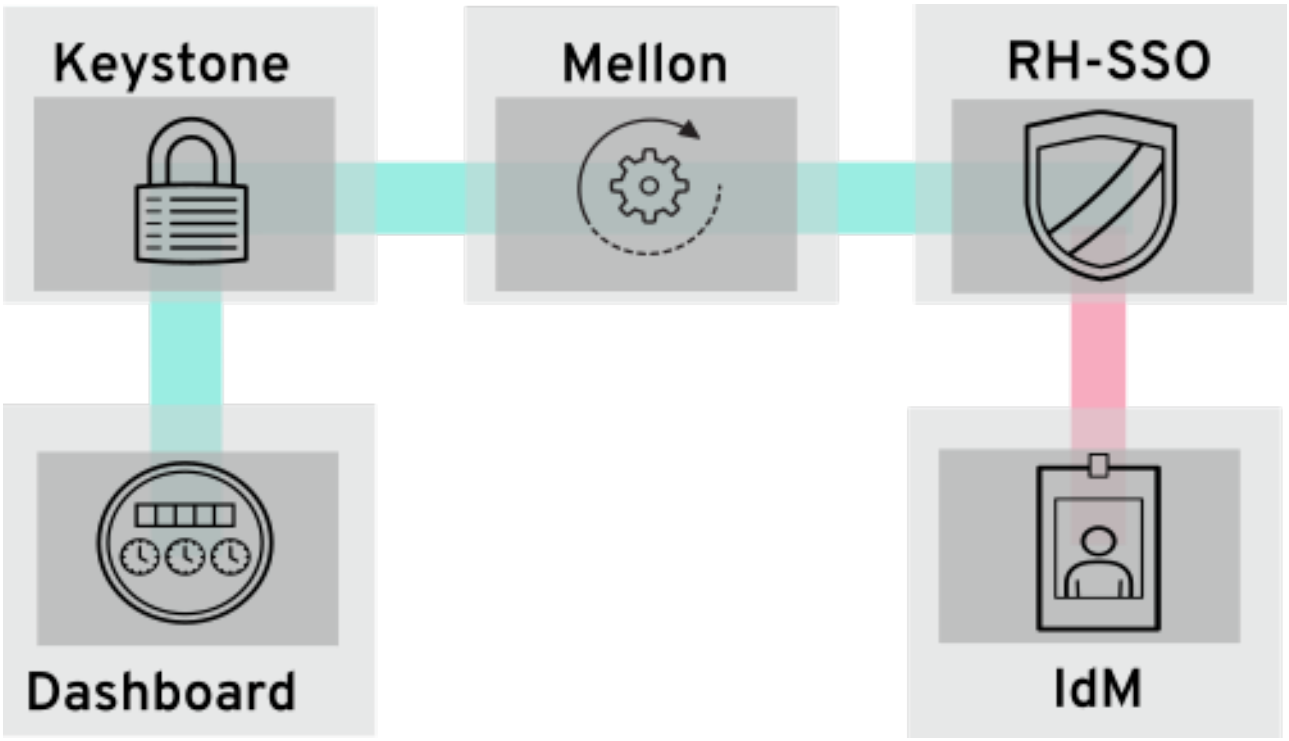
以下の図では、keystone (SP) は RH-SSO[id=**the-federation-workflow_implementing-federation**] = フェデレーションワークフローと通信します。

ここでは、Identity サービス (keystone)、RH-SSO、IdM の相互作用について説明します。OpenStack におけるフェデレーションは、認証プロバイダーとサービスプロバイダーの概念を使用します。

認証プロバイダー (IdP): ユーザーアカウントを保存するサービス。この場合、IdM に保持されるユーザーアカウントは、RH-SSO を使用して keystone に表示されます。

サービスプロバイダー (SP): IdP のユーザーからの認証を必要とするサービス。この場合、keystone は IdM ユーザーに Dashboard へのアクセスを付与するサービスプロバイダーです。

以下の図では、keystone (SP) は RH-SSO (IdP) と通信します。これは、他の IdP の汎用アダプターとしても機能することができます。この設定では、RH-SSO に keystone をポイントすることができ、RH-SSO は要求をサポートする認証プロバイダー (認証モジュールと呼ばれます) に転送します。現在、これらには IdM および Active Directory が含まれます。これは、サービスプロバイダー (SP) および認証プロバイダー (IdP) がメタデータを交換し、各システム管理者が信頼する決定を行うことで行われます。その結果、IdP は確実に決定を行い、SP はこれらの決定を受け取ることができます。



関連情報

- Identity サービスとのフェデレーション