Red Hat OpenShiftとNetApp

NetApp container solutions

NetApp
January 21, 2026

目次

Red Hat OpenShiftとNetApp

NVA-1160: NetAppを使用した Red Hat OpenShift

アラン・カウルズとニヒル・M・クルカルニ（NetApp）

このリファレンス ドキュメントでは、 NetAppによって検証された、インストーラー プロビジョニング インフラストラクチャ (IPI) を通じて複数の異なるデータセンター環境に導入された Red Hat OpenShift ソリューションの導入検証について説明します。また、永続ストレージの管理にTridentストレージ オーケストレーターを利用することで、NetAppストレージ システムとのストレージ統合についても詳しく説明します。最後に、いくつかのソリューション検証と実際の使用例が調査され、文書化されます。

ユースケース

Red Hat OpenShift with NetAppソリューションは、次のようなユースケースを持つ顧客に卓越した価値を提供するように設計されています。

- ベアメタル、Red Hat OpenStack Platform、Red Hat Virtualization、VMware vSphere に IPI (Installer Provisioned Infrastructure) を使用してデプロイされた Red Hat OpenShift を簡単にデプロイおよび管理できます。
- エンタープライズ コンテナと仮想化ワークロードのパワーを、OSP、RHV、vSphere 上に仮想的にデプロイされた Red Hat OpenShift、または OpenShift Virtualization によるベアメタル上にデプロイされた Red Hat OpenShift と組み合わせます。
- NetAppストレージおよび Kubernetes 用のオープンソース ストレージ オーケストレーターであるTrident と組み合わせて使用した場合の Red Hat OpenShift の機能を強調した実際の構成と使用例。

ビジネス価値

企業は、新製品の作成、リリース サイクルの短縮、新機能の迅速な追加のために DevOps プラクティスを採用するケースが増えています。コンテナとマイクロサービスは、その本質的なアジャイル性により、DevOps プラクティスのサポートにおいて重要な役割を果たします。ただし、エンタープライズ環境の運用規模で DevOps を実践するには、独自の課題があり、基盤となるインフラストラクチャに次のような特定の要件が課せられます。

- スタック内のすべてのレイヤーで高可用性を実現
- 導入手順の容易さ
- 中断のない運用とアップグレード
- マイクロサービスの俊敏性を維持するための API 駆動型でプログラム可能なインフラストラクチャ
- パフォーマンス保証付きのマルチテナンシー
- 仮想化されたワークロードとコンテナ化されたワークロードを同時に実行する機能
- ワークロードの需要に応じてインフラストラクチャを個別に拡張できる機能

NetAppを使用した Red Hat OpenShift はこれらの課題を認識し、顧客が選択したデータセンター環境で RedHat OpenShift IPI の完全自動導入を実装することにより、各懸念に対処するソリューションを提供しま

す。

技術概要

Red Hat OpenShift with NetAppソリューションは、次の主要コンポーネントで構成されています。

Red Hat OpenShift コンテナ プラットフォーム

Red Hat OpenShift Container Platform は、完全にサポートされているエンタープライズ Kubernetes プラットフォームです。 Red Hat はオープンソースの Kubernetes にいくつかの機能強化を加え、コンテナ化されたアプリケーションを構築、展開、管理するためのすべてのコンポーネントが完全に統合されたアプリケーション プラットフォームを提供します。

詳細については、OpenShift の Web サイトをご覧ください。 "ここをクリックしてください。" 。

NetApp ストレージ システム

NetApp には、エンタープライズ データ センターやハイブリッド クラウドの導入に最適なストレージ システムがいくつかあります。 NetAppポートフォリオには、 NetApp ONTAP、 NetApp Element、 NetApp e-Series ストレージ システムが含まれており、いずれもコンテナ化されたアプリケーションに永続的なストレージを提供できます。

詳細については、 NetAppのWebサイトをご覧ください。 "ここをクリックしてください。" 。

NetAppストレージ統合

Trident は、Red Hat OpenShift を含むコンテナおよび Kubernetes ディストリビューション向けのオープンソースで完全にサポートされているストレージ オーケストレーターです。

詳細については、Tridentのウェブサイトをご覧ください。 "ここをクリックしてください。" 。

高度な設定オプション

このセクションでは、専用のプライベート イメージ レジストリの作成やカスタム ロード バランサ インスタンスのデプロイなど、このソリューションを本番環境にデプロイするときに実際のユーザーが実行する必要があるカスタマイズについて説明します。

検証済みリリースの現在のサポート マトリックス

テクノロジ	目的	ソフトウェア バージョン
NetApp ONTAP	ストレージ	9.8、9.9.1、9.12.1
NetApp Element	ストレージ	12.3
NetAppTrident	ストレージオーケストレーション	22.01.0、23.04、23.07、23.10、24.02
レッドハット オープンシフト	コンテナオーケストレーション	4.6 EUS、4.7、4.8、4.10、4.11、4.12、4.13、4.14
VMware vSphere	データセンター仮想化	7.0、8.0.2

レッドハットオープンシフト

OpenShift の概要

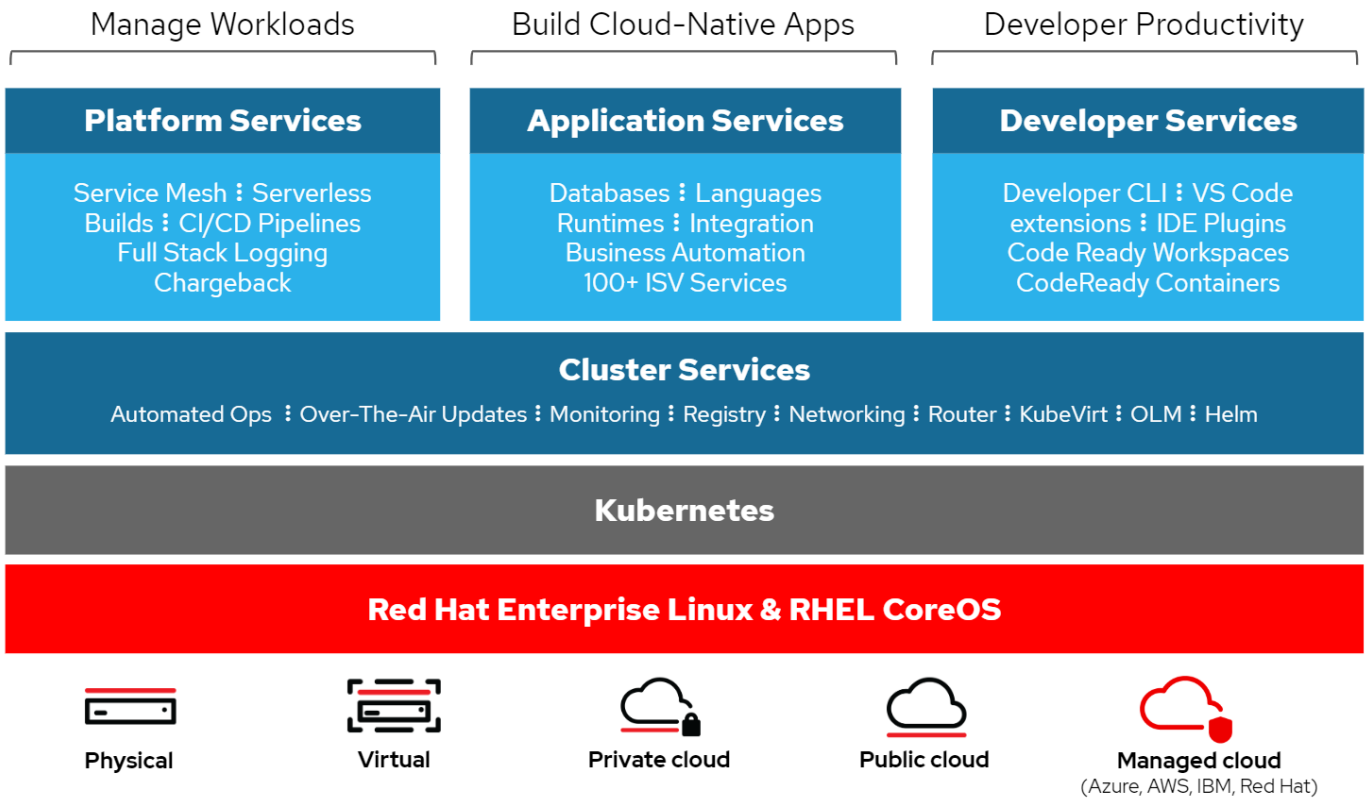
Red Hat OpenShift Container Platform は、開発と IT 運用を単一のプラットフォームに統合し、オンプレミスとハイブリッド クラウド インフラストラクチャ全体で一貫してアプリケーションを構築、展開、管理します。 Red Hat OpenShift は、Kubernetes や、コンテナベースのワークロード向けに設計された世界トップクラスのエンタープライズ Linux ディストリビューションである Red Hat Enterprise Linux CoreOS などのオープンソースのイノベーションと業界標準に基づいて構築されています。 OpenShift は、Cloud Native Computing Foundation (CNCF) 認定 Kubernetes プログラムの一部であり、コンテナー ワークロードの移植性と相互運用性を提供します。

Red Hat OpenShift は次の機能を提供します。

- セルフサービス プロビジョニング 開発者は、最もよく使用するツールからオンデマンドでアプリケーションを迅速かつ簡単に作成でき、運用担当者は環境全体を完全に制御できます。
- 永続ストレージ 永続ストレージのサポートを提供することで、OpenShift Container Platform ではステートフル アプリケーションとクラウド ネイティブのステートレス アプリケーションの両方を実行できます。
- 継続的インテグレーションと継続的開発 **(CI/CD)** このソースコード プラットフォームは、ビルドおよびデプロイメント イメージを大規模に管理します。
- オープンソース標準 これらの標準には、他のオープンソーステクノロジーに加えて、コンテナオーケストレーション用の Open Container Initiative (OCI) と Kubernetes が組み込まれています。特定のベンダーのテクノロジーやビジネス ロードマップに制限されることはありません。
- **CI/CD** パイプライン OpenShift は、すぐに使用できる CI/CD パイプラインのサポートを提供するため、開発チームはアプリケーション配信プロセスのすべてのステップを自動化し、アプリケーションのコードや構成に加えられたすべての変更に対して確実に実行することができます。
- ロールベースのアクセス制御 **(RBAC)** この機能は、チームとユーザーの追跡を提供し、大規模な開発者グループの編成に役立ちます。
- 自動ビルドとデプロイ OpenShift では、開発者はコンテナ化されたアプリケーションをビルドするか、プラットフォームでアプリケーションのソース コードやバイナリからコンテナをビルドするかを選択できます。次に、プラットフォームは、アプリケーションに対して定義された特性に基づいて、インフラストラクチャ全体にわたるこれらのアプリケーションの展開を自動化します。たとえば、サードパーティのライセンスに準拠するために、リソースをどのくらいの量割り当て、インフラストラクチャのどこに展開する必要があるかなどです。
- 一貫性のある環境 OpenShift は、一貫性のない環境から生じるリスクを排除するために、開発者向けにプロビジョニングされた環境とアプリケーションのライフサイクル全体にわたって、オペレーティング システムからライブラリ、ランタイム バージョン (Java ランタイムなど)、さらには使用中のアプリケーション ランタイム (Tomcat など) まで一貫性があることを確認します。
- 構成管理 構成と機密データの管理がプラットフォームに組み込まれているため、アプリケーションの構築にどのテクノロジが使用され、どの環境にデプロイされるかに関係なく、一貫性があり環境に依存しないアプリケーション構成がアプリケーションに提供されます。
- *アプリケーション ログとメトリック。*迅速なフィードバックはアプリケーション開発の重要な側面です。 OpenShift の統合モニタリングおよびログ管理では、開発者に即時のメトリックが提供され、開発者は変更後のアプリケーションの動作を調査し、アプリケーション ライフサイクルのできるだけ早い段階で

問題を修正できるようになります。

- セキュリティとコンテナ カタログ OpenShift は、Security-Enhanced Linux (SELinux)、CGroups、および Secure Computing Mode (seccomp) による確立されたセキュリティを使用してコンテナを分離および保護し、マルチテナント機能を提供し、有害なコード実行からユーザーを保護します。また、さまざまなサブシステムの TLS 証明書による暗号化や、セキュリティに特に重点を置いたスキャンとグレード付けが行われた Red Hat 認定コンテナ (access.redhat.com/containers) へのアクセスも提供し、認定された、信頼できる、安全なアプリケーション コンテナをエンド ユーザーに提供します。



Red Hat OpenShiftのデプロイメント方法

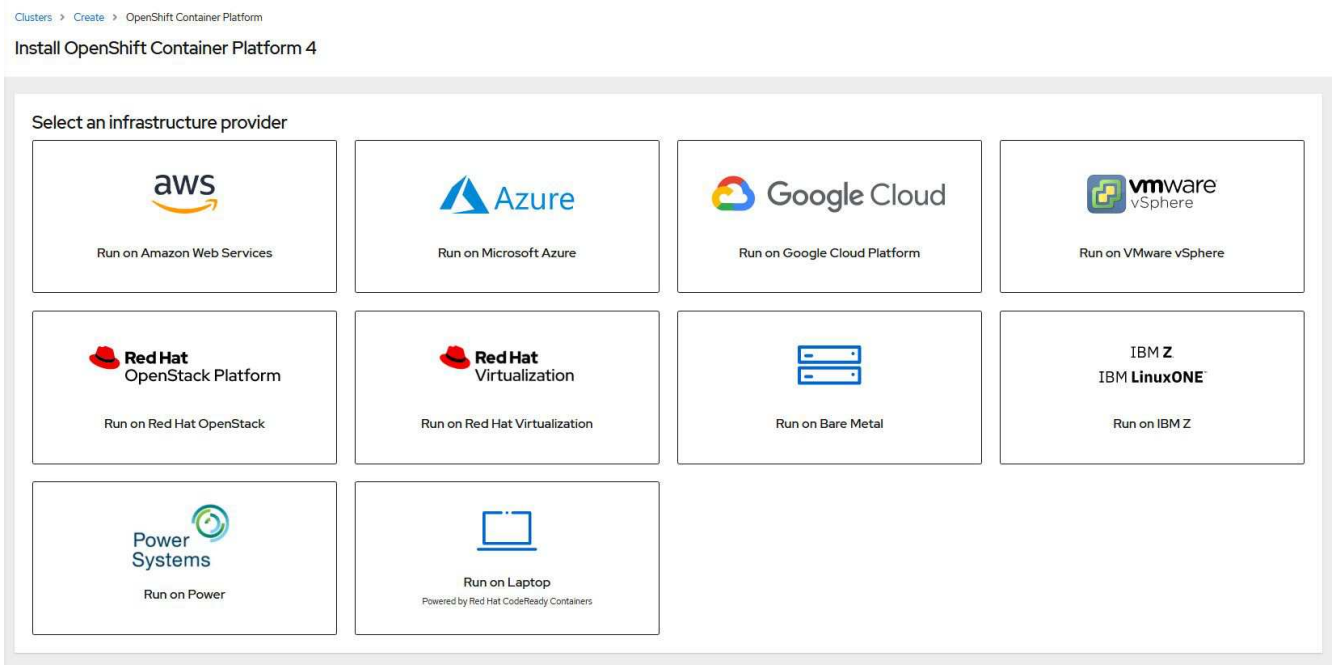
Red Hat OpenShift 4 以降、OpenShift のデプロイメント方法には、高度にカスタマイズされたデプロイメントのための User Provisioned Infrastructure (UPI) を使用した手動デプロイメントや、Installer Provisioned Infrastructure (IPI) を使用した完全に自動化されたデプロイメントが含まれます。

IPI インストール方法は、開発、テスト、実稼働環境向けの OpenShift クラスターの迅速な導入を可能にするため、ほとんどの場合に推奨される方法です。

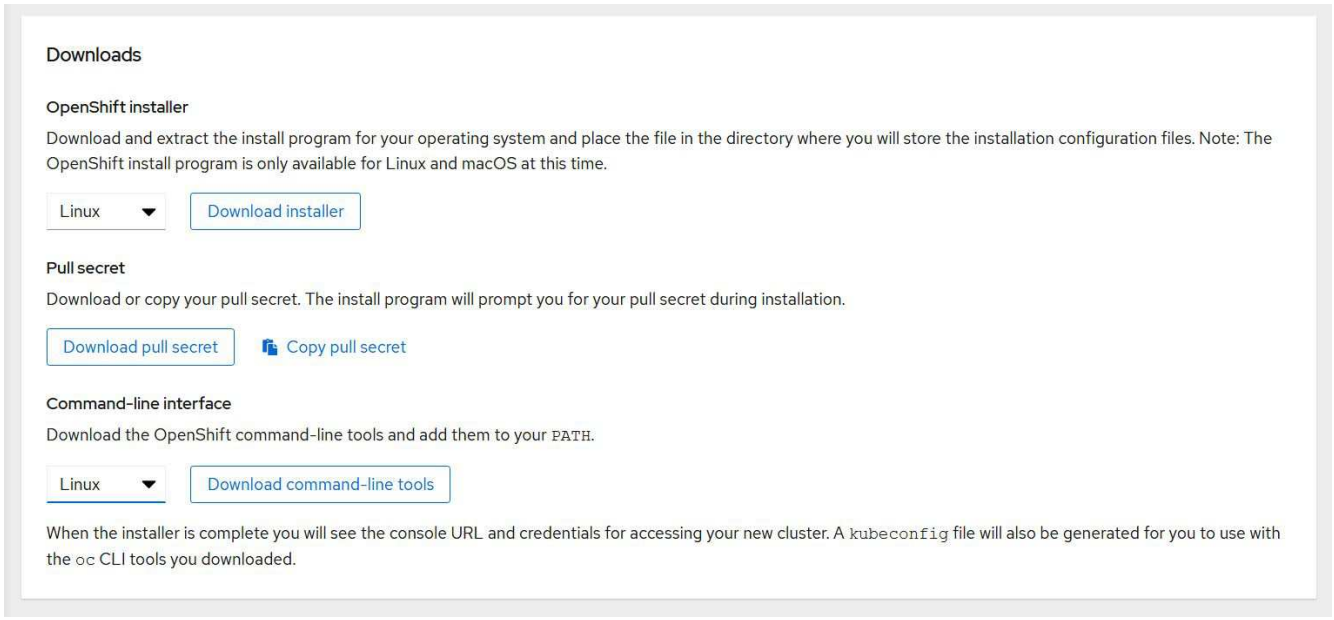
Red Hat OpenShiftのIPIインストール

OpenShift の Installer Provisioned Infrastructure (IPI) デプロイメントには、次の大まかな手順が含まれます。

1. Red Hat OpenShiftをご覧ください"Webサイト"SSO 資格情報を使用してログインします。
2. Red Hat OpenShift をデプロイする環境を選択します。



3. 次の画面で、インストーラー、一意のプル シークレット、および管理用の CLI ツールをダウンロードします。



4. フォロー"インストール手順"選択した環境にデプロイするために Red Hat によって提供されます。

NetApp検証済み **OpenShift** 導入

NetApp は、次の各データセンター環境で、インストーラー プロビジョニング インフラストラクチャ (IPI) 導入方法を使用して、ラボで Red Hat OpenShift の導入をテストおよび検証しました。

- "ベアメタル上の OpenShift"
- "Red Hat OpenStack プラットフォーム上の OpenShift"
- "Red Hat Virtualization 上の OpenShift"

- "VMware vSphere 上の OpenShift"

ベアメタル上の **OpenShift**

OpenShift on Bare Metal は、コモディティ サーバー上での OpenShift Container Platform の自動デプロイメントを提供します。

OpenShift on Bare Metal は OpenShift の仮想デプロイメントに似ており、コンテナ化の準備ができていないアプリケーションの仮想化ワークロードをサポートしながら、OpenShift クラスターのデプロイメント、迅速なプロビジョニング、スケーリングを容易にします。ベアメタルにデプロイすることで、OpenShift 環境に加えてホストハイパーバイザー環境を管理するために必要な余分なオーバーヘッドが不要になります。ベアメタル サーバーに直接デプロイすることで、ホストと OpenShift 環境間でリソースを共有する必要があることによる物理的なオーバーヘッドの制限も軽減できます。

OpenShift on Bare Metal は次の機能を提供します。

- **IPI** または支援インストーラーの導入 インストーラー プロビジョニング インフラストラクチャ (IPI) によってベアメタル サーバーに導入された OpenShift クラスターを使用すると、顧客はハイパーバイザー レイヤーを管理する必要なく、汎用性が高く、簡単に拡張できる OpenShift 環境をコモディティ サーバーに直接導入できます。
- コンパクトなクラスター設計 ハードウェア要件を最小限に抑えるために、ベアメタル上の OpenShift では、OpenShift コントロール プレーン ノードがワーカー ノードやホスト コンテナーとしても機能できるようにすることで、ユーザーは 3 ノードのみのクラスターを展開できます。
- **OpenShift** 仮想化 OpenShift は、OpenShift 仮想化を使用してコンテナー内で仮想マシンを実行できます。このコンテナネイティブ仮想化は、コンテナ内で KVM ハイパーバイザーを実行し、VM ストレージ用の永続ボリュームを接続します。
- **AI/ML** に最適化されたインフラストラクチャ GPU ベースのワーカー ノードを OpenShift 環境に組み込み、OpenShift Advanced Scheduling を活用することで、機械学習アプリケーション用の Kubeflow などのアプリケーションをデプロイします。

ネットワーク設計

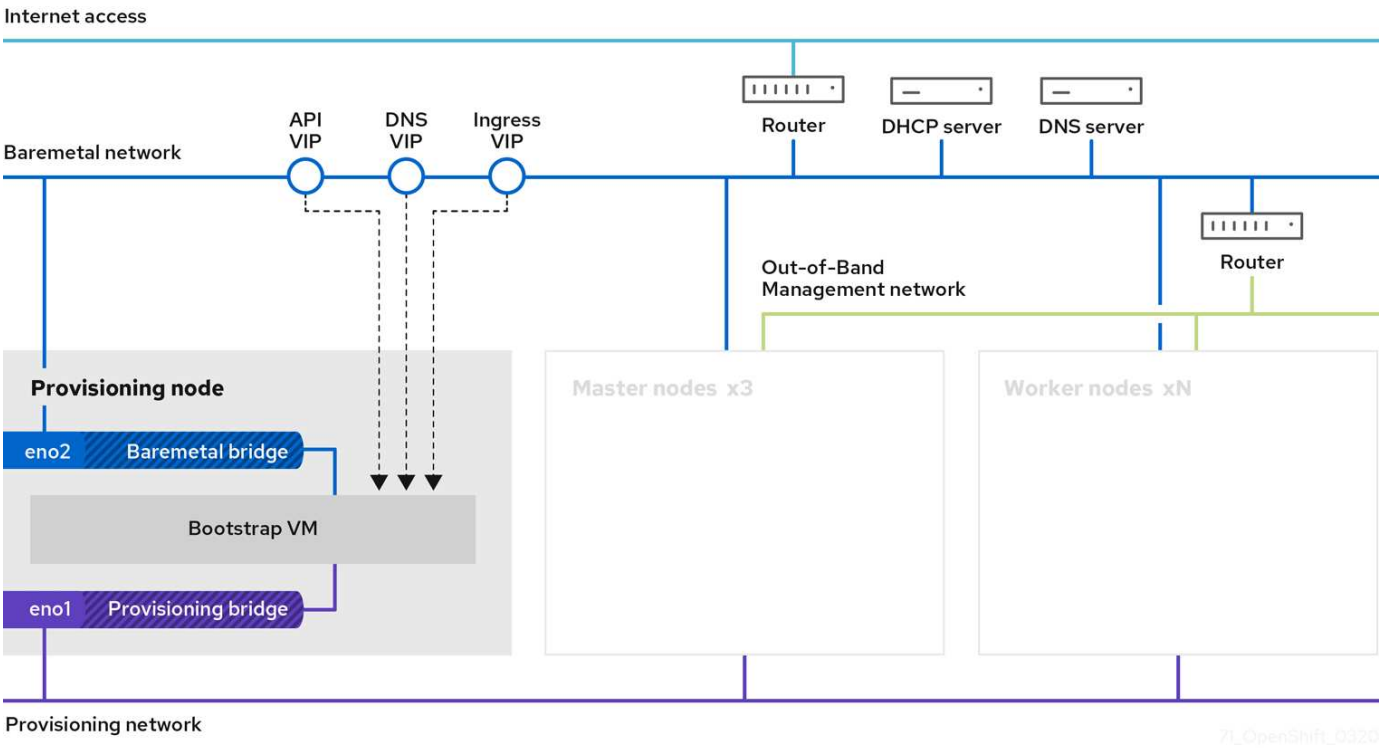
Red Hat OpenShift on NetAppソリューションは、2 つのデータ スイッチを使用して 25 Gbps でプライマリ データ接続を提供します。また、ストレージ ノードのインバンド管理と IPMI 機能のアウトオブバンド管理のために 1 Gbps の接続を提供する 2 つの管理スイッチも使用します。

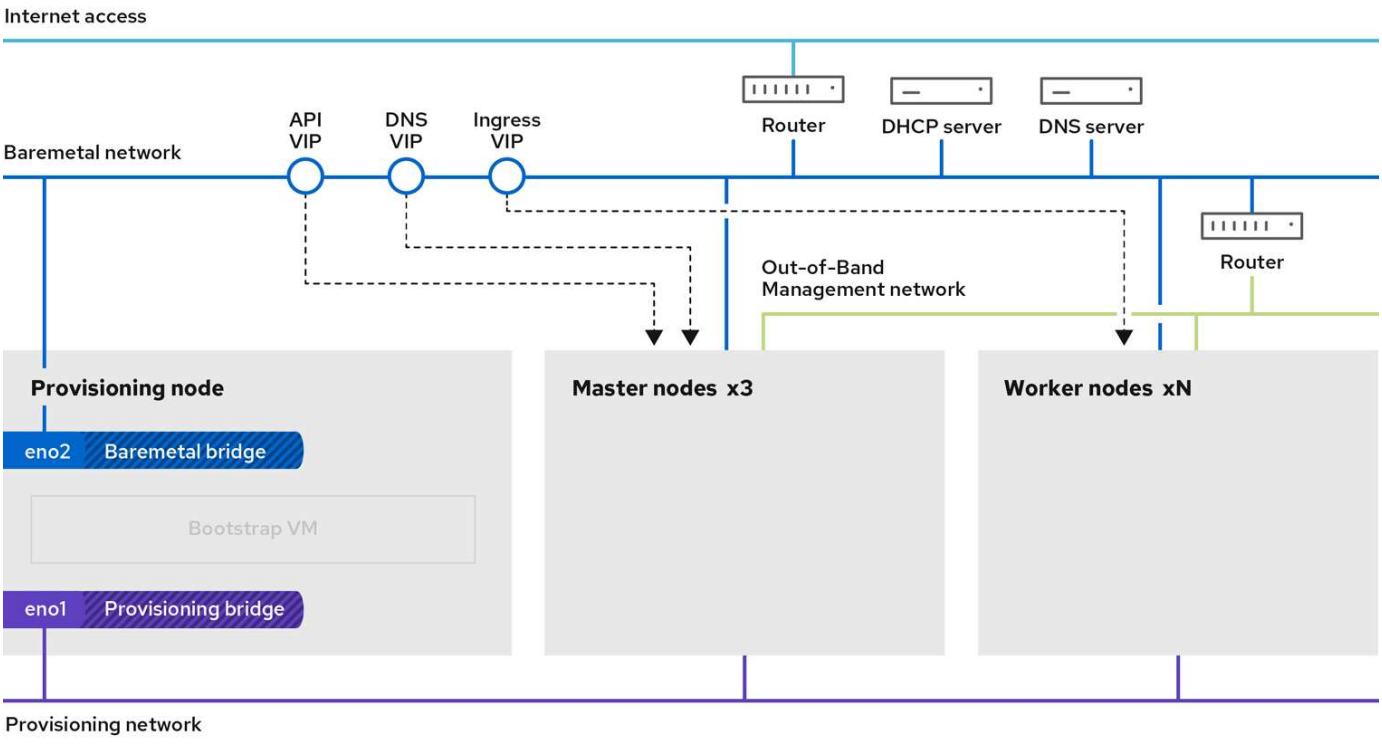
OpenShift ベアメタル IPI デプロイメントでは、ネットワーク インターフェースが別個のネットワークに接続されている必要がある Red Hat Enterprise Linux 8 マシンであるプロビジョナー ノードを作成する必要があります。

- プロビジョニング ネットワーク このネットワークは、ベアメタル ノードを起動し、OpenShift クラスターをデプロイするために必要なイメージとパッケージをインストールするために使用されます。
- ベアメタル ネットワーク このネットワークは、クラスターがデプロイされた後の公開通信に使用されます。

プロビジョナー ノードをセットアップするために、顧客は、ノード自体と、展開目的でプロビジョニングされたブートストラップ VM 上でトラフィックが適切にルーティングされるようにするブリッジ インターフェイスを作成します。クラスターがデプロイされた後、API および Ingress VIP アドレスはブートストラップ ノードから新しくデプロイされたクラスターに移行されます。

次の画像は、IPI の展開中と展開が完了した後の環境を示しています。





VLANの要件

NetAppソリューションを使用した Red Hat OpenShift は、仮想ローカル エリア ネットワーク (VLAN) を使用

して、さまざまな目的に合わせてネットワーク トラフィックを論理的に分離するように設計されています。

VLAN	目的	VLAN ID
帯域外管理ネットワーク	ベアメタルノードとIPMIの管理	16
ベアメタルネットワーク	クラスターが利用可能になった後の OpenShift サービス用のネットワーク	181
プロビジョニングネットワーク	IPI 経由の PXE ブートおよびベアメタルノードのインストール用のネットワーク	3485

これらの各ネットワークは VLAN によって仮想的に分離されていますが、PXE ブート シーケンス中に VLAN タグを渡す方法がないため、各物理ポートはプライマリ VLAN が割り当てられたアクセス モードで設定する必要があります。

ネットワークインフラストラクチャサポートリソース

OpenShift コンテナ プラットフォームを展開する前に、次のインフラストラクチャを準備しておく必要があります。

- インバンド管理ネットワークと VM ネットワークからアクセス可能な完全なホスト名解決を提供する少なくとも 1 つの DNS サーバー。
- インバンド管理ネットワークと VM ネットワークからアクセス可能な NTP サーバーが少なくとも 1 つ。
- (オプション) インバンド管理ネットワークと VM ネットワークの両方に対する送信インターネット接続。

Red Hat OpenStack プラットフォーム上の OpenShift

Red Hat OpenStack Platform は、安全で信頼性の高いプライベート OpenStack クラウドを作成、展開、拡張するための統合基盤を提供します。

OSP は、コンピューティング、ストレージ、およびネットワーク リソースを管理する制御サービスのコレクションによって実装される、サービスとしてのインフラストラクチャ (IaaS) クラウドです。環境は、管理者とユーザーが OpenStack リソースを制御、プロビジョニング、自動化できる Web ベースのインターフェースを使用して管理されます。さらに、OpenStack インフラストラクチャは、広範なコマンド ライン インターフェイスと API を通じて実現され、管理者とエンド ユーザー向けの完全な自動化機能を実現します。

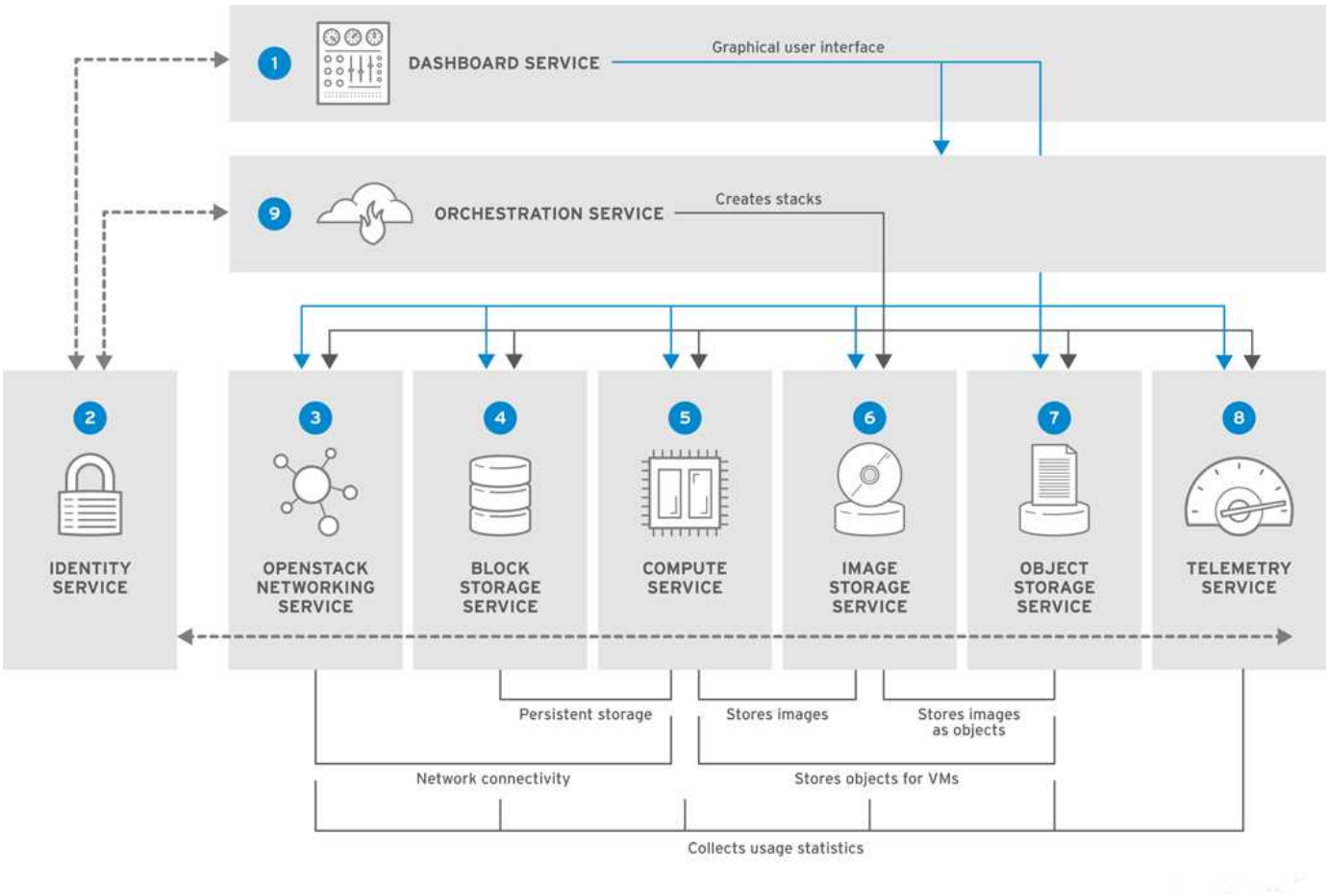
OpenStack プロジェクトは、6 か月ごとに更新リリースを提供する、急速に開発が進むコミュニティ プロジェクトです。当初、Red Hat OpenStack Platform は、アップストリーム リリースごとに新しいリリースを公開し、3 回目のリリースごとに長期サポートを提供することで、このリリース サイクルに対応していました。最近、OSP 16.0 リリース (OpenStack Train ベース) では、Red Hat はリリース番号に追従せず、代わりに新しい機能をサブリリースにバックポートすることを選択しました。最新リリースは Red Hat OpenStack Platform 16.1 で、アップストリームの Ussuri および Victoria リリースからバックポートされた高度な機能が含まれています。

OSPの詳細については、"Red Hat OpenStack Platform ウェブサイト" 。

OpenStackサービス

OpenStack Platform サービスはコンテナとしてデプロイされ、サービスが相互に分離され、簡単にアップグレードできるようになります。 OpenStack プラットフォームは、Kolla で構築および管理されるコンテナのセットを使用します。サービスのデプロイメントは、Red Hat カスタムポータルからコンテナイメージをプルす

ることによって実行されます。これらのサービス コンテナは Podman コマンドを使用して管理され、Red Hat OpenStack Director を使用してデプロイ、構成、および保守されます。



サービス	プロジェクト名	説明
ダッシュボード	地平線	OpenStack サービスを管理するために使用する Web ブラウザベースのダッシュボード。
身元	Keystone	OpenStack サービスの認証と承認、およびユーザー、プロジェクト、ロールの管理のための集中型サービス。
OpenStackネットワーク	中性子	OpenStack サービスのインターフェース間の接続を提供します。
ブロックストレージ	Cinder	仮想マシン (VM) の永続ブロック ストレージ ボリュームを管理します。
コンピューティング	ノヴァ	コンピューティング ノードで実行されている VM を管理およびプロビジョニングします。
イメージ	一目	VM イメージやボリューム スナップショットなどのリソースを保存するために使用されるレジストリ サービス。
オブジェクトストレージ	迅速	ユーザーがファイルや任意のデータを保存および取得できるようにします。
テレメトリー	天井計	クラウド リソースの使用量の測定を提供します。

サービス	プロジェクト名	説明
オーケストレーション	熱	リソース スタックの自動作成をサポートするテンプレート ベースのオーケストレーション エンジン。

ネットワーク設計

Red Hat OpenShift with NetAppソリューションは、2 つのデータ スイッチを使用して 25 Gbps でプライマリ データ接続を提供します。また、ストレージ ノードのインバンド管理と IPMI 機能のアウトオブバンド管理のために 1 Gbps の接続を提供する 2 つの追加管理スイッチも使用します。

Ironic ベアメタル プロビジョニング サービスを使用して Red Hat OpenStack Platform をデプロイするには、Red Hat OpenStack Director に IPMI 機能が必要です。

VLANの要件

NetAppを搭載した Red Hat OpenShift は、仮想ローカル エリア ネットワーク (VLAN) を使用して、さまざまな目的に合わせてネットワーク トラフィックを論理的に分離するように設計されています。この構成は、顧客の要求を満たすために、または特定のネットワーク サービスをさらに分離するために拡張できます。次の表は、 NetAppでソリューションを検証する際にソリューションを実装するために必要な VLAN を示しています。

VLAN	目的	VLAN ID
帯域外管理ネットワーク	Ironic の物理ノードと IPMI サービスの管理に使用されるネットワーク。	16
ストレージインフラストラクチャ	コントローラー ノードがボリュームを直接マップして Swift などのインフラストラクチャ サービスをサポートするために使用されるネットワーク。	201
貯蔵用灰	環境にデプロイされた仮想インスタンスにブロック ボリュームを直接マップおよび接続するために使用されるネットワーク。	202
内部API	API 通信、RPC メッセージ、およびデータベース通信を使用した OpenStack サービス間の通信に使用されるネットワーク。	301
テナント	Neutron は、VXLAN を介したトンネリングを通じて各テナントに独自のネットワークを提供します。ネットワーク トラフィックは各テナント ネットワーク内で分離されます。各テナント ネットワークには IP サブネットが関連付けられており、ネットワーク名前空間により、複数のテナント ネットワークが競合を起こすことなく同じアドレス範囲を使用できます。	302
ストレージ管理	OpenStack Object Storage (Swift) は、このネットワークを使用して、参加しているレプリカ ノード間でデータ オブジェクトを同期します。プロキシ サービスは、ユーザー要求と基盤となるストレージ層間の中間インターフェイスとして機能します。プロキシは着信要求を受け取り、要求されたデータを取得するために必要なレプリカを見つけます。	303
PXE	OpenStack Director は、OSP オーバークラウドのインストールを調整するために、Ironic ベアメタル プロビジョニング サービスの一部として PXE ブートを提供します。	3484
外部	グラフィカル管理用の OpenStack ダッシュボード (Horizon) をホストし、OpenStack サービスを管理するためのパブリック API 呼び出しを許可する、パブリックに利用可能なネットワーク。	3485

VLAN	目的	VLAN ID
インバンド管理ネットワーク	SSH アクセス、DNS トラフィック、ネットワーク タイム プロトコル (NTP) トラフィックなどのシステム管理機能へのアクセスを提供します。このネットワークは、コントローラー以外のノードのゲートウェイとしても機能します。	3486

ネットワークインフラストラクチャサポートリソース

OpenShift Container Platform をデプロイする前に、次のインフラストラクチャを準備しておく必要があります。

- 完全なホスト名解決を提供する少なくとも 1 つの DNS サーバー。
- ソリューション内のサーバーの時間を同期できる NTP サーバーが少なくとも 3 台。
- (オプション) OpenShift 環境の送信インターネット接続。

本番環境への導入に関するベストプラクティス

このセクションでは、このソリューションを本番環境に導入する前に組織が考慮する必要があるベスト プラクティスをいくつか示します。

少なくとも**3**つのコンピューティングノードを持つ**OSP**プライベートクラウドに**OpenShift**をデプロイする

このドキュメントで説明する検証済みのアーキテクチャは、3 つの OSP コントローラ ノードと 2 つの OSP コンピューティング ノードを展開することにより、HA 操作に適した最小限のハードウェア展開を示します。このアーキテクチャにより、両方のコンピューティング ノードが仮想インスタンスを起動でき、展開された VM が 2 つのハイパーバイザー間で移行できるフォールト トレラント構成が保証されます。

Red Hat OpenShift は最初に 3 つのマスターノードでデプロイされるため、2 ノード構成では少なくとも 2 つのマスターが同じノードを占有することになり、その特定のノードが使用できなくなった場合に OpenShift が停止する可能性があります。したがって、OpenShift マスターを均等に分散し、ソリューションのフォールト トレランスをさらに高めることができるように、少なくとも 3 つの OSP コンピューティング ノードをデプロイすることが Red Hat のベスト プラクティスです。

仮想マシン/ホストアフィニティを構成する

VM/ホスト アフィニティを有効にすると、OpenShift マスターを複数のハイパーバイザー ノードに分散できます。

アフィニティは、VM がグループ内の同じホストまたは複数のホスト上で一緒に実行されるか、または異なるホスト上で実行されるかを決定する VM および/またはホストのセットのルールを定義する方法です。これは、一連の同一のパラメータと条件を持つ VM やホストで構成されるアフィニティ グループを作成することによって VM に適用されます。アフィニティ グループ内の VM が同じホスト上で実行されるか、グループ内のホスト上で実行されるか、または異なるホスト上で個別に実行されるかに応じて、アフィニティ グループのパラメータは、正のアフィニティまたは負のアフィニティのいずれかを定義できます。 Red Hat OpenStack Platform では、サーバー グループを作成し、フィルターを構成することで、ホスト アフィニティ ルールと非アフィニティ ルールを作成して適用できます。これにより、サーバー グループ内の Nova によってデプロイされたインスタンスは、異なるコンピューティング ノードにデプロイされます。

サーバー グループでは、配置を管理できる仮想インスタンスのデフォルトの最大数は 10 です。これは、Nova のデフォルトのクォータを更新することで変更できます。

OSP サーバー グループには特定のハード アフィニティ/アンチ アフィニティ制限があり、個別のノードに展開するのに十分なリソースがない場合、またはノードの共有を許可するのに十分なリソースがない場合、VM は起動に失敗します。

アフィニティグループを構成するには、"OpenStack インスタンスのアフィニティとアンチアフィニティを設定するにはどうすればよいですか?" 。

OpenShift のデプロイメントにカスタム インストール ファイルを使用する

IPI は、このドキュメントで前述した対話型ウィザードを通じて OpenShift クラスターのデプロイメントを容易にします。ただし、クラスターの展開の一環として、いくつかのデフォルト値を変更する必要がある場合があります。

このような場合、すぐにクラスターを展開せずにウィザードを実行してタスクを実行できます。代わりに、後でクラスターを展開できる構成ファイルが作成されます。これは、IPI のデフォルトを変更する必要がある場合や、マルチテナントなどの他の用途のために環境内に複数の同一クラスターを展開する場合に非常に便利です。OpenShiftのカスタマイズされたインストール構成の作成の詳細については、以下を参照してください。"Red Hat OpenShift カスタマイズによる OpenStack へのクラスターのインストール"。

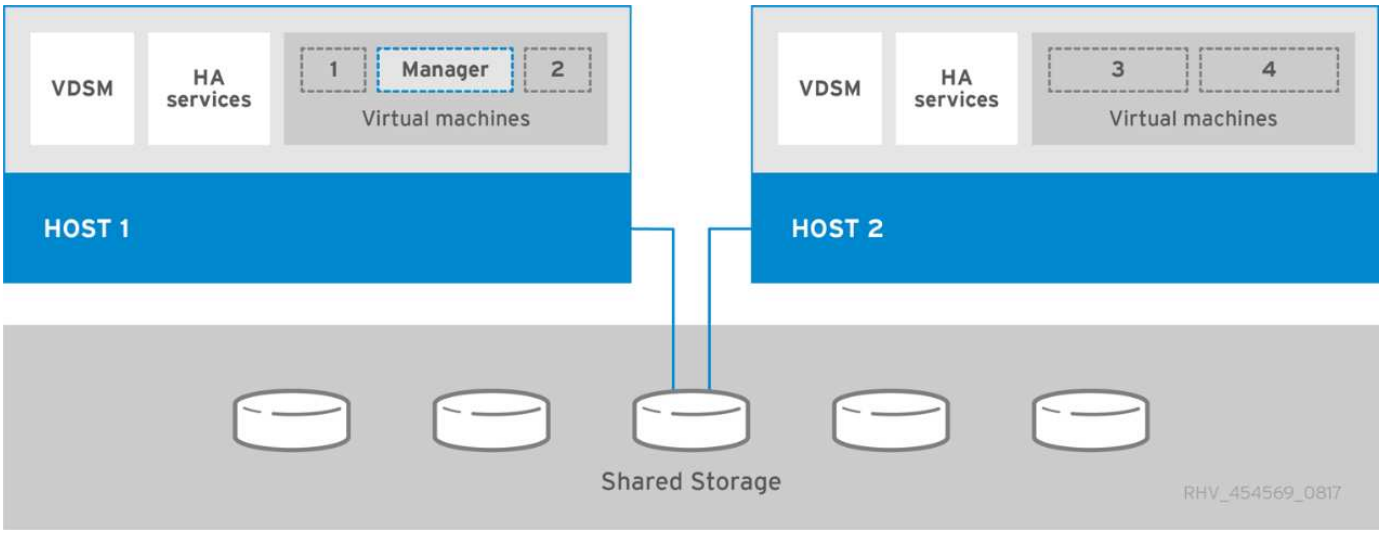
Red Hat Virtualization 上の OpenShift

Red Hat Virtualization (RHV) は、Red Hat Enterprise Linux (RHEL) 上で実行され、KVM ハイパーバイザーを使用するエンタープライズ仮想データセンター プラットフォームです。

RHVの詳細については、"Red Hat Virtualization のウェブサイト" 。

RHV は次の機能を提供します。

- **VM** とホストの集中管理 RHV マネージャーは、展開内で物理マシンまたは仮想マシン (VM) として実行され、中央インターフェースからソリューションを管理するための Web ベースの GUI を提供します。
- セルフホストエンジン ハードウェア要件を最小限に抑えるために、RHV では、ゲスト VM を実行する同じホスト上に RHV Manager (RHV-M) を VM として展開できます。
- 高可用性 ホスト障害が発生した場合の中断を回避するために、RHV では VM を高可用性向けに構成できます。高可用性 VM は、回復性ポリシーを使用してクラスター レベルで制御されます。
- 高いスケーラビリティ 単一の RHV クラスターには最大 200 個のハイパーバイザー ホストを含めることができるため、大量のリソースを必要とするエンタープライズ クラスのワークロードをホストする大規模な VM の要件をサポートできます。
- 強化されたセキュリティ RHV から継承された Secure Virtualization (sVirt) および Security Enhanced Linux (SELinux) テクノロジは、ホストと VM のセキュリティと強化を強化するために RHV によって採用されています。これらの機能の主な利点は、VM とその関連リソースの論理的な分離です。



ネットワーク設計

Red Hat OpenShift on NetAppソリューションは、2 つのデータ スイッチを使用して 25 Gbps でプライマリ データ接続を提供します。また、ストレージ ノードのインバンド管理と IPMI 機能のアウトオブバンド管理のために 1 Gbps の接続を提供する 2 つの追加管理スイッチも使用します。 OCP は、クラスター管理に RHV 上の仮想マシン論理ネットワークを使用します。このセクションでは、ソリューションで使用される各仮想ネットワーク セグメントの配置と目的について説明し、ソリューションを展開するための前提条件の概要を示します。

VLANの要件

RHV 上の Red Hat OpenShift は、仮想ローカル エリア ネットワーク (VLAN) を使用して、さまざまな目的に合わせてネットワーク トラフィックを論理的に分離するように設計されています。この構成は、顧客の要求を満たすために、または特定のネットワーク サービスをさらに分離するために拡張できます。次の表は、NetAppでソリューションを検証する際にソリューションを実装するために必要な VLAN を示しています。

VLAN	目的	VLAN ID
帯域外管理ネットワーク	物理ノードとIPMIの管理	16
VMネットワーク	仮想ゲストネットワークアクセス	1172
インバンド管理ネットワーク	RHV-Hノード、RHV-Manager、およびovirtmgmtネットワークの管理	3343
ストレージ ネットワーク	NetApp Element iSCSI のストレージ ネットワーク	3344
移住ネットワーク	仮想ゲスト移行用のネットワーク	3345

ネットワークインフラストラクチャサポートリソース

OpenShift Container Platform をデプロイする前に、次のインフラストラクチャを準備しておく必要があります。

- インバンド管理ネットワークおよび VM ネットワークからアクセス可能な完全なホスト名解決を提供する少なくとも 1 つの DNS サーバー。
- インバンド管理ネットワークと VM ネットワークからアクセス可能な NTP サーバーが少なくとも 1 つ。

- (オプション) インバンド管理ネットワークと VM ネットワークの両方に対する送信インターネット接続。

本番環境への導入に関するベストプラクティス

このセクションでは、このソリューションを本番環境に導入する前に組織が考慮する必要があるベスト プラクティスをいくつか示します。

少なくとも3ノードのRHVクラスタにOpenShiftをデプロイする

このドキュメントで説明する検証済みのアーキテクチャは、2 つの RHV-H ハイパーバイザー ノードを展開し、両方のホストがホスト エンジンを管理でき、展開された VM が 2 つのハイパーバイザー間で移行できるフォールト トレラント構成を確保することで、HA 操作に適した最小限のハードウェア展開を示します。

Red Hat OpenShift は最初に 3 つのマスターノードでデプロイされるため、2 ノード構成では少なくとも 2 つのマスターが同じノードを占有することが保証され、その特定のノードが使用できなくなった場合に OpenShift が停止する可能性があります。したがって、OpenShift マスターを均等に分散し、ソリューションのフォールト トレランスをさらに高めるために、ソリューションの一部として少なくとも 3 つの RHV-H ハイパーバイザー ノードを展開することが Red Hat のベスト プラクティスです。

仮想マシン/ホストアフィニティを構成する

VM/ホスト アフィニティを有効にすると、OpenShift マスターを複数のハイパーバイザー ノードに分散できます。

アフィニティは、VM がグループ内の同じホストまたは複数のホスト上で一緒に実行されるか、または異なるホスト上で実行されるかを決定する VM および/またはホストのセットのルールを定義する方法です。これは、一連の同一のパラメータと条件を持つ VM やホストで構成されるアフィニティ グループを作成することによって VM に適用されます。アフィニティ グループ内の VM が同じホスト上で実行されるか、グループ内のホスト上で実行されるか、または異なるホスト上で個別に実行されるかに応じて、アフィニティ グループのパラメータは、正のアフィニティまたは負のアフィニティのいずれかを定義できます。

パラメータに定義される条件は、ハード強制またはソフト強制のいずれかになります。ハード強制により、アフィニティ グループ内の VM は、外部条件に関係なく、常に正または負のアフィニティに厳密に従うようになります。ソフト強制により、アフィニティ グループ内の VM が可能な限り正または負のアフィニティに従うように高い優先順位が設定されます。このドキュメントで説明されている 2 つまたは 3 つのハイパーバイザー構成では、ソフト アフィニティが推奨される設定です。大規模なクラスターでは、ハード アフィニティによって OpenShift ノードを正しく分散できます。

アフィニティグループを構成するには、"レッドハット6.11。アフィニティグループのドキュメント" 。

OpenShift のデプロイメントにカスタム インストール ファイルを使用する

IPI は、このドキュメントで前述した対話型ウィザードを通じて OpenShift クラスターのデプロイメントを容易にします。ただし、クラスターの展開の一環として、いくつかのデフォルト値を変更する必要がある可能性があります。

このような場合、クラスターをすぐに展開せずにウィザードを実行してタスクを実行できます。代わりに、後でクラスターをデプロイできる構成ファイルが作成されます。これは、IPI のデフォルトを変更する場合や、マルチテナントなどの他の用途のために環境内に複数の同一クラスターを展開する場合に非常に便利です。OpenShiftのカスタマイズされたインストール構成の作成の詳細については、以下を参照してください。"Red Hat OpenShift カスタマイズによるRHVへのクラスターのインストール" 。

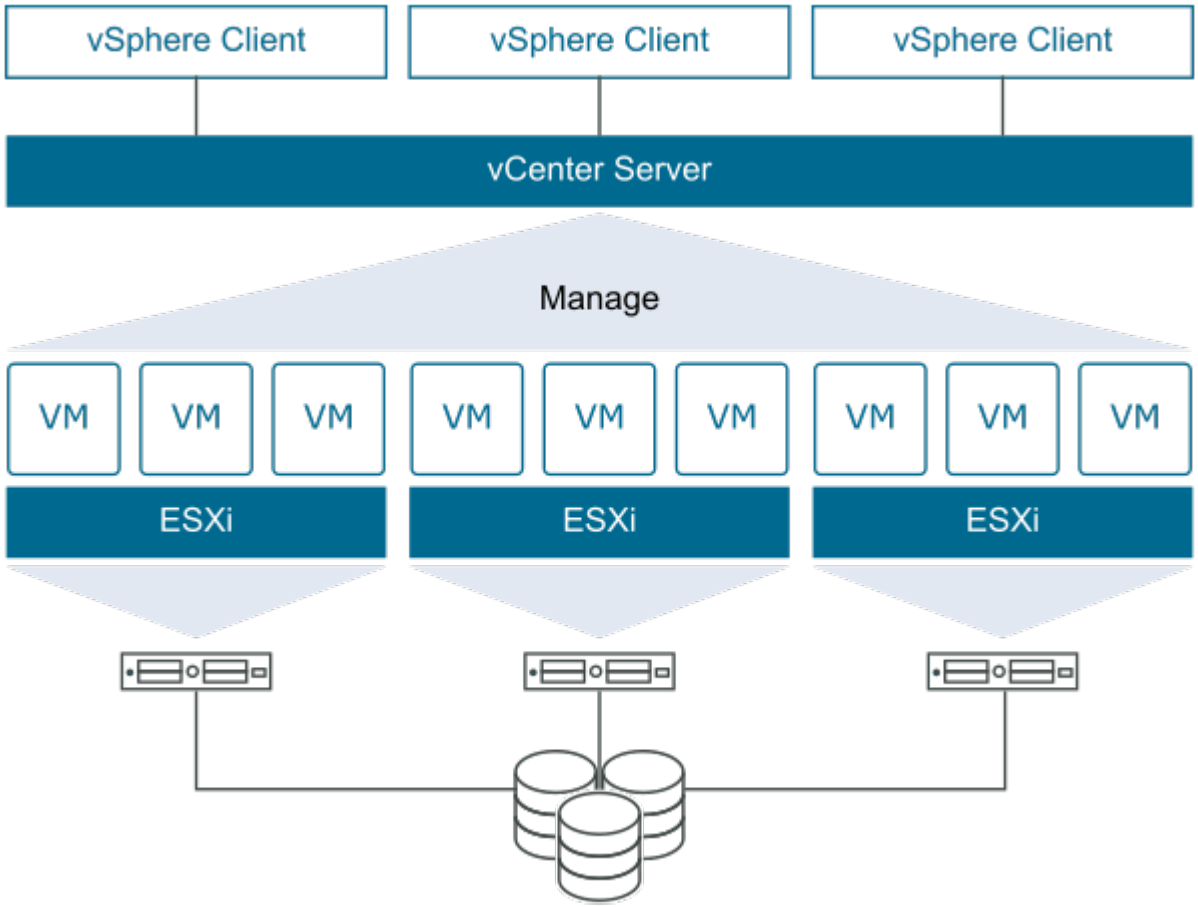
VMware vSphere 上の OpenShift

VMware vSphere は、ESXi ハイパーバイザー上で実行される多数の仮想化サーバーとネットワークを集中管理するための仮想化プラットフォームです。

VMware vSphereの詳細については、"VMware vSphere Web サイト" 。

VMware vSphere は次の機能を提供します。

- **VMware vCenter Server** VMware vCenter Server は、単一のコンソールからすべてのホストと VM を統合管理し、クラスター、ホスト、および VM のパフォーマンス監視を集約します。
- **VMware vSphere vMotion** VMware vCenter を使用すると、リクエストに応じて、中断することなくクラスター内のノード間で VM をホット マイグレーションできます。
- **vSphere High Availability** ホスト障害が発生した場合の中断を回避するために、VMware vSphere では、ホストをクラスタ化し、高可用性用に構成できます。ホスト障害によって中断された VM は、クラスター内の他のホスト上ですぐに再起動され、サービスが復元されます。
- 分散リソース スケジューラ **(DRS)** VMware vSphere クラスターは、ホストしている VM のリソース ニーズを負荷分散するように構成できます。リソース競合が発生している VM は、十分なリソースが利用可能であることを保証するため、クラスター内の他のノードにホット移行できます。



ネットワーク設計

Red Hat OpenShift on NetAppソリューションは、2 つのデータ スイッチを使用して 25 Gbps でプライマリ データ接続を提供します。また、ストレージ ノードのインバンド管理と IPMI 機能のアウトオブバンド管理の

ために 1 Gbps の接続を提供する 2 つの追加管理スイッチも使用します。 OCP は、クラスター管理に VMware vSphere 上の VM 論理ネットワークを使用します。このセクションでは、ソリューションで使用される各仮想ネットワーク セグメントの配置と目的について説明し、ソリューションの展開の前提条件の概要を示します。

VLANの要件

VMware vSphere 上の Red Hat OpenShift は、仮想ローカル エリア ネットワーク (VLAN) を使用して、さまざまな目的に合わせてネットワーク トラフィックを論理的に分離するように設計されています。この構成は、顧客の要求を満たすために、または特定のネットワーク サービスをさらに分離するために拡張できます。次の表は、 NetAppでソリューションを検証する際にソリューションを実装するために必要な VLAN を示しています。

VLAN	目的	VLAN ID
帯域外管理ネットワーク	物理ノードとIPMIの管理	16
VMネットワーク	仮想ゲストネットワークアクセス	181
ストレージ ネットワーク	ONTAP NFSのストレージネットワーク	184
ストレージ ネットワーク	ONTAP iSCSI のストレージ ネットワーク	185
インバンド管理ネットワーク	ESXiノード、VCenter Server、 ONTAP Selectの管理	3480
ストレージ ネットワーク	NetApp Element iSCSI のストレージ ネットワーク	3481
移住ネットワーク	仮想ゲスト移行用のネットワーク	3482

ネットワークインフラストラクチャサポートリソース

OpenShift Container Platform をデプロイする前に、次のインフラストラクチャを準備しておく必要があります。

- インバンド管理ネットワークおよび VM ネットワークからアクセス可能な完全なホスト名解決を提供する少なくとも 1 つの DNS サーバー。
- インバンド管理ネットワークと VM ネットワークからアクセス可能な NTP サーバーが少なくとも 1 つ。
- (オプション) インバンド管理ネットワークと VM ネットワークの両方に対する送信インターネット接続。

本番環境への導入に関するベストプラクティス

このセクションでは、このソリューションを本番環境に導入する前に組織が考慮する必要があるベスト プラクティスをいくつか示します。

少なくとも3つのノードのESXiクラスタにOpenShiftをデプロイする

このドキュメントで説明する検証済みのアーキテクチャは、2 つの ESXi ハイパーバイザー ノードを展開し、VMware vSphere HA と VMware vMotion を有効にしてフォールト トレラント構成を確保することで、HA 操作に適した最小限のハードウェア展開を示します。この構成により、展開された VM を 2 つのハイパーバイザー間で移行し、1 つのホストが使用できなくなった場合に再起動することができます。

Red Hat OpenShift は最初に 3 つのマスターノードでデプロイされるため、状況によっては 2 ノード構成内の少なくとも 2 つのマスターが同じノードを占有することがあり、その特定のノードが使用できなくなった場

合に OpenShift が停止する可能性があります。したがって、OpenShift マスターを均等に分散してフォールトトレランスをさらに高めるためには、少なくとも 3 つの ESXi ハイパーバイザー ノードをデプロイすることが Red Hat のベスト プラクティスです。

仮想マシンとホストのアフィニティを構成する

VM とホストのアフィニティを有効にすることで、OpenShift マスターを複数のハイパーバイザー ノードに分散させることができます。

アフィニティまたはアンチアフィニティは、VM がグループ内の同じホストまたは複数のホスト上で一緒に実行されるか、または異なるホスト上で実行されるかを決定する VM および/またはホストのセットのルールを定義する方法です。これは、一連の同一のパラメータと条件を持つ VM やホストで構成されるアフィニティ グループを作成することによって VM に適用されます。アフィニティ グループ内の VM が同じホスト上で実行されるか、グループ内のホスト上で実行されるか、または異なるホスト上で個別に実行されるかに応じて、アフィニティ グループのパラメータは、正のアフィニティまたは負のアフィニティのいずれかを定義できます。

アフィニティグループを構成するには、 "vSphere 9.0 ドキュメント: DRS アフィニティ ルールの使用"。

OpenShift のデプロイメントにカスタム インストール ファイルを使用する

IPI は、このドキュメントで前述した対話型ウィザードを通じて OpenShift クラスターのデプロイメントを容易にします。ただし、クラスターの展開の一環として、いくつかのデフォルト値を変更する必要がある場合があります。

このような場合、クラスターをすぐに展開せずにウィザードを実行してタスクを実行できますが、代わりにウィザードによって、後でクラスターを展開できる構成ファイルが作成されます。これは、IPI のデフォルトを変更する必要がある場合、またはマルチテナントなどの他の用途のために環境内に複数の同一クラスターを展開する場合に非常に便利です。 OpenShiftのカスタマイズされたインストール構成の作成の詳細については、以下を参照してください。"Red Hat OpenShift カスタマイズによる vSphere へのクラスタのインストール" 。

AWS 上の Red Hat OpenShift サービス

Red Hat OpenShift Service on AWS (ROSA) は、AWS 上の Red Hat OpenShift エンタープライズ Kubernetes プラットフォームを使用してコンテナ化されたアプリケーションを構築、スケーリング、およびデプロイするために使用できるマネージドサービスです。 ROSA は、オンプレミスの Red Hat OpenShift ワークロードの AWS への移行を効率化し、他の AWS サービスとの緊密な統合を実現します。

ROSA の詳細については、次のドキュメントを参照してください。"AWS 上の Red Hat OpenShift サービス (AWS ドキュメント)" 。"AWS 上の Red Hat OpenShift サービス (Red Hat ドキュメント)" 。

NetApp ストレージ システム

NetApp ONTAP

NetApp ONTAPは、直感的な GUI、自動化統合を備えた REST API、AI を活用した予測分析と修正アクション、中断のないハードウェア アップグレード、ストレージ間のインポートなどの機能を備えた強力なストレージ ソフトウェア ツールです。

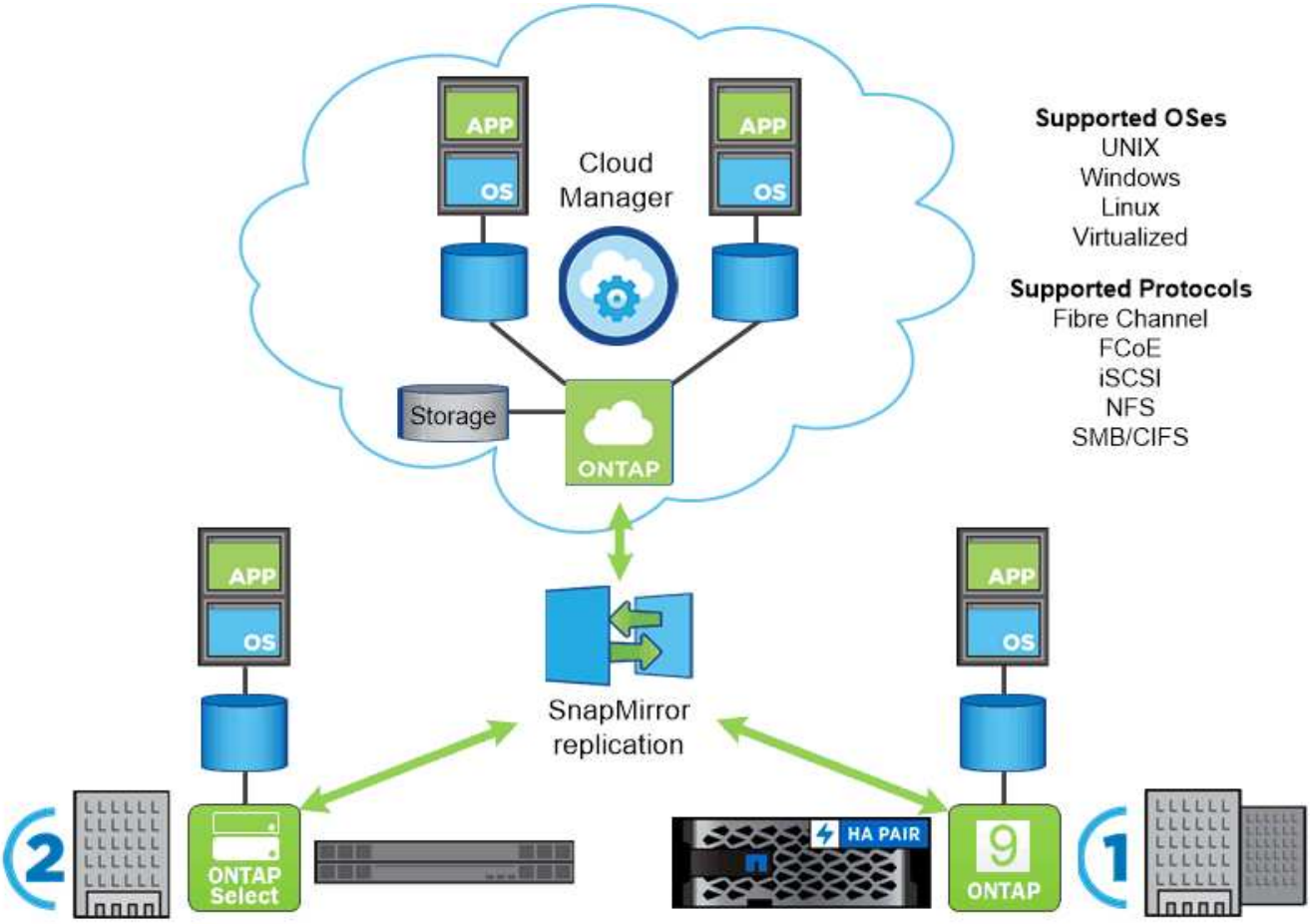
NetApp ONTAPストレージシステムの詳細については、 "NetApp ONTAPウェブサイト" 。

ONTAP は次の機能を提供します。

- NFS、CIFS、iSCSI、FC、FCoE、FC-NVMe プロトコルの同時データ アクセスと管理を備えた統合ストレージ システム。
- さまざまな導入モデルには、オールフラッシュ、ハイブリッド、オール HDD ハードウェア構成のオンプレミス、 ONTAP Selectなどのサポートされているハイパーバイザー上の VM ベースのストレージ プラットフォーム、およびCloud Volumes ONTAPとしてのクラウドが含まれます。
- 自動データ階層化、インライン データ圧縮、重複排除、および圧縮のサポートにより、 ONTAPシステムのデータ ストレージ効率が向上します。
- ワークロードベースの QoS 制御ストレージ。
- データの階層化と保護のためにパブリック クラウドとシームレスに統合します。 ONTAP は、あらゆる環境において際立った強力なデータ保護機能も提供します。
  - * NetAppスナップショットコピー。*追加のパフォーマンス オーバーヘッドなしで、最小限のディスク領域を使用して、データの高速なポイントインタイム バックアップを実行します。
  - * NetApp SnapMirror。*あるストレージ システムから別のストレージ システムにデータのスナップショット コピーをミラーリングします。 ONTAP は、他の物理プラットフォームやクラウドネイティブ サービスへのデータのミラーリングもサポートします。
  - * NetAppSnapLock。*指定された期間、上書きまたは消去できない特別なボリュームに書き込むことで、書き換え不可能なデータを効率的に管理します。
  - * NetApp SnapVault*複数のストレージ システムからのデータを、指定されたすべてのシステムのバックアップとして機能する中央のスナップショット コピーにバックアップします。
  - * NetApp SyncMirror。*同じコントローラに物理的に接続された 2 つの異なるディスク プレックスに、リアルタイムの RAID レベルのデータ ミラーリングを提供します。
  - * NetApp SnapRestore。*スナップショット コピーからオンデマンドでバックアップされたデータを高速に復元します。
  - * NetApp FlexClone。*スナップショット コピーに基づいて、 NetAppボリュームの完全に読み取りおよび書き込み可能なコピーを瞬時にプロビジョニングします。

ONTAPの詳細については、 "ONTAP 9ドキュメント センター" 。

NetApp ONTAP は、オンプレミス、仮想化、またはクラウドで利用できます。



NetAppプラットフォーム

NetApp AFF/ FAS

NetApp は、低遅延パフォーマンス、統合データ保護、およびマルチプロトコル サポートを備えたカスタマイズされた、堅牢なオールフラッシュ (AFF) およびスケールアウト ハイブリッド (FAS) ストレージ プラットフォームを提供します。

どちらのシステムも、業界最先端のデータ管理ソフトウェアであるNetApp ONTAPデータ管理ソフトウェアを搭載しており、高可用性、クラウド統合、簡素化されたストレージ管理を実現し、データ ファブリックに必要なエンタープライズ クラスの速度、効率、セキュリティを実現します。

NETAPP AFF/ FASプラットフォームの詳細については、こちらをクリックしてください。 "ここをクリックしてください。" 。

ONTAP Select

ONTAP Select は、環境内のハイパーバイザーに導入できるNetApp ONTAPのソフトウェア定義の導入です。VMware vSphere または KVM にインストールでき、ハードウェアベースのONTAPシステムの完全な機能とエクスペリエンスを提供します。

ONTAP Selectの詳細については、 "ここをクリックしてください。" 。

Cloud Volumes ONTAP

NetApp Cloud Volumes ONTAP は、Amazon AWS、Microsoft Azure、Google Cloud などのさまざまなパブリック クラウドに導入できるNetApp ONTAPのクラウド導入バージョンです。

Cloud Volumes ONTAPの詳細については、 "ここをクリックしてください。" 。

Amazon FSx ONTAP

Amazon FSx ONTAP は、 ONTAPの一般的なデータアクセスおよび管理機能を備えた、AWS クラウドで完全に管理された共有ストレージを提供します。 Amazon FSx ONTAPの詳細については、 "ここをクリックしてください。" 。

Azure NetApp Files

Azure NetApp Files は、Azure ネイティブ、ファーストパーティ、エンタープライズ クラスの高パフォーマンス ファイル ストレージ サービスです。 NetAppアカウント、容量プール、ボリュームを作成できるボリュームをサービスとして提供します。サービスとパフォーマンスのレベルを選択し、データ保護を管理することもできます。オンプレミスで使い慣れているものと同じプロトコルとツールを使用して、高パフォーマンス、高可用性、スケーラブルなファイル共有を作成および管理できます。 Azure NetApp Filesの詳細については、 "ここをクリックしてください。" 。

Google Cloud NetApp Volumes

Google Cloud NetApp Volumes は、高度なデータ管理機能と高度にスケーラブルなパフォーマンスを提供する、完全に管理されたクラウドベースのデータ ストレージ サービスです。ファイルベースのアプリケーションを Google Cloud に移動できます。ネットワーク ファイル システム (NFSv3 および NFSv4.1) とサーバー メッセージ ブロック (SMB) プロトコルのサポートが組み込まれているため、アプリケーションを再設計する必要がなく、アプリケーション用の永続的なストレージを継続的に利用できます。 Google Cloud NetApp VolumesPの詳細については、 "ここをクリックしてください。" 。

NetApp Element: NetAppを使用した Red Hat OpenShift

NetApp Elementソフトウェアは、モジュール式のスケーラブルなパフォーマンスを提供し、各ストレージ ノードが環境に保証された容量とスループットを提供します。 NetApp Elementシステムは、単一のクラスター内で 4 ノードから 100 ノードまで拡張でき、高度なストレージ管理機能を多数提供します。

NetApp Elementストレージシステムの詳細については、 "NetApp Solidfire ウェブサイト" 。

iSCSI ログインリダイレクトと自己修復機能

NetApp Elementソフトウェアは、従来の TCP/IP ネットワーク上で SCSI コマンドをカプセル化する標準的な方法である iSCSI ストレージ プロトコルを活用します。 SCSI 規格が変更されたり、イーサネット ネットワークのパフォーマンスが向上したりしても、iSCSI ストレージ プロトコルは変更を必要とせずにメリットを享受できます。

すべてのストレージ ノードには管理 IP とストレージ IP がありますが、 NetApp Elementソフトウェアは、クラスター内のすべてのストレージ トラフィックに対して単一のストレージ仮想 IP アドレス (SVIP アドレス) をアドバタイズします。 iSCSI ログイン プロセスの一部として、ストレージは、ターゲット ボリュームが別のアドレスに移動されたため、ネゴシエーション プロセスを続行できないと応答する場合があります。その後、ホストは、ホスト側の再構成を必要としないプロセスで、新しいアドレスへのログイン要求を再発行します。このプロセスは、iSCSI ログイン リダイレクトと呼ばれます。

iSCSI ログイン リダイレクトは、 NetApp Elementソフトウェア クラスターの重要な部分です。ホスト ログイン要求を受信すると、ノードは IOPS とボリュームの容量要件に基づいて、クラスターのどのメンバーがトラフィックを処理するかを決定します。ボリュームはNetApp Elementソフトウェア クラスタ全体に分散され、単一のノードがそのボリュームに対して過剰なトラフィックを処理している場合、または新しいノードが追加された場合は再分散されます。特定のボリュームの複数のコピーがアレイ全体に割り当てられます。

このように、ノード障害の後にボリュームの再配分が行われる場合、ログアウトして新しい場所にリダイレクトされるログイン以外には、ホストの接続には影響がありません。 iSCSI ログイン リダイレクトを使用すると、 NetApp Elementソフトウェア クラスターは、中断のないアップグレードと操作が可能な自己修復型のスケールアウト アーキテクチャになります。

NetApp Elementソフトウェア クラスタ QoS

NetApp Elementソフトウェア クラスターを使用すると、ボリュームごとに QoS を動的に構成できます。ボリュームごとの QoS 設定を使用して、定義した SLA に基づいてストレージ パフォーマンスを制御できます。次の 3 つの設定可能なパラメータによって QoS が定義されます。

- 最小 **IOPS** NetApp Elementソフトウェア クラスタがボリュームに提供する持続的な IOPS の最小数。ボリュームに設定された最小 IOPS は、ボリュームの保証されたパフォーマンス レベルです。ボリュームあたりのパフォーマンスはこのレベルを下回ることはありません。
- 最大 **IOPS** NetApp Elementソフトウェア クラスタが特定のボリュームに提供する持続 IOPS の最大数。
- *バースト IOPS*短いバーストシナリオで許可される IOPS の最大数。バースト期間の設定は構成可能で、デフォルトは 1 分です。ボリュームが最大 IOPS レベルを下回って実行されている場合、バースト クレジットが蓄積されます。パフォーマンス レベルが非常に高くなり、限界に達した場合、ボリューム上で最大 IOPS を超える IOPS の短時間のバーストが許可されます。

マルチテナンシー

安全なマルチテナントは次の機能によって実現されます。

- *安全な認証。*チャレンジ ハンドシェイク認証プロトコル (CHAP) は、安全なボリューム アクセスに使用されます。軽量ディレクトリ アクセス プロトコル (LDAP) は、管理およびレポート用のクラスターへの安全なアクセスに使用されます。
- *ボリューム アクセス グループ (VAG)*オプションで、認証の代わりに VAG を使用し、任意の数の iSCSI イニシエーター固有の iSCSI 修飾名 (IQN) を 1 つ以上のボリュームにマッピングできます。 VAG 内のボリュームにアクセスするには、イニシエーターの IQN がボリューム グループの許可された IQN リストに含まれている必要があります。

- *テナント仮想 LAN (VLAN)。*ネットワーク レベルでは、VLAN を使用することで、iSCSI イニシエーターとNetApp Elementソフトウェア クラスター間のエンドツーエンドのネットワーク セキュリティが実現されます。ワークロードまたはテナントを分離するために作成された VLAN に対して、 NetApp Element ソフトウェアは、特定の VLAN 経由でのみアクセス可能な個別の iSCSI ターゲット SVIP アドレスを作成します。
- *VRF 対応 VLAN。*データセンターのセキュリティとスケーラビリティをさらにサポートするために、NetApp Elementソフトウェアでは、任意のテナント VLAN で VRF のような機能を有効にできます。この機能により、次の 2 つの主要な機能が追加されます。
  - *テナント SVIP アドレスへの L3 ルーティング。*この機能を使用すると、iSCSI イニシエーターをNetApp Elementソフトウェア クラスターとは別のネットワークまたは VLAN に配置できます。
  - *重複または複製された IP サブネット。*この機能を使用すると、テナント環境にテンプレートを追加して、各テナント VLAN に同じ IP サブネットからの IP アドレスを割り当てることができます。この機能は、IPspace のスケールと保存が重要なインサービス プロバイダー環境に役立ちます。

エンタープライズストレージ効率

NetApp Elementソフトウェア クラスタは、全体的なストレージ効率とパフォーマンスを向上させます。次の機能はインラインで実行され、常にオンになっており、ユーザーによる手動構成は必要ありません。

- *重複排除。*システムは一意の 4K ブロックのみを保存します。重複した 4K ブロックは、すでに保存されているデータのバージョンに自動的に関連付けられます。データはブロック ドライブ上に保存され、NetApp Elementソフトウェア Helix データ保護を使用してミラーリングされます。このシステムにより、システム内の容量消費と書き込み操作が大幅に削減されます。
- *圧縮。*データがNVRAMに書き込まれる前に、インラインで圧縮が実行されます。データは圧縮され、4K ブロックに保存され、システム内で圧縮されたままになります。この圧縮により、クラスター全体の容量消費、書き込み操作、帯域幅消費が大幅に削減されます。
- *シンプロビジョニング*この機能により、必要なときに適切な量のストレージが提供され、過剰にプロビジョニングされたボリュームや十分に活用されていないボリュームによって発生する容量の消費が排除されます。
- *ヘリックス。*個々のボリュームのメタデータはメタデータ ドライブに保存され、冗長性のためにセカンダリ メタデータ ドライブに複製されます。

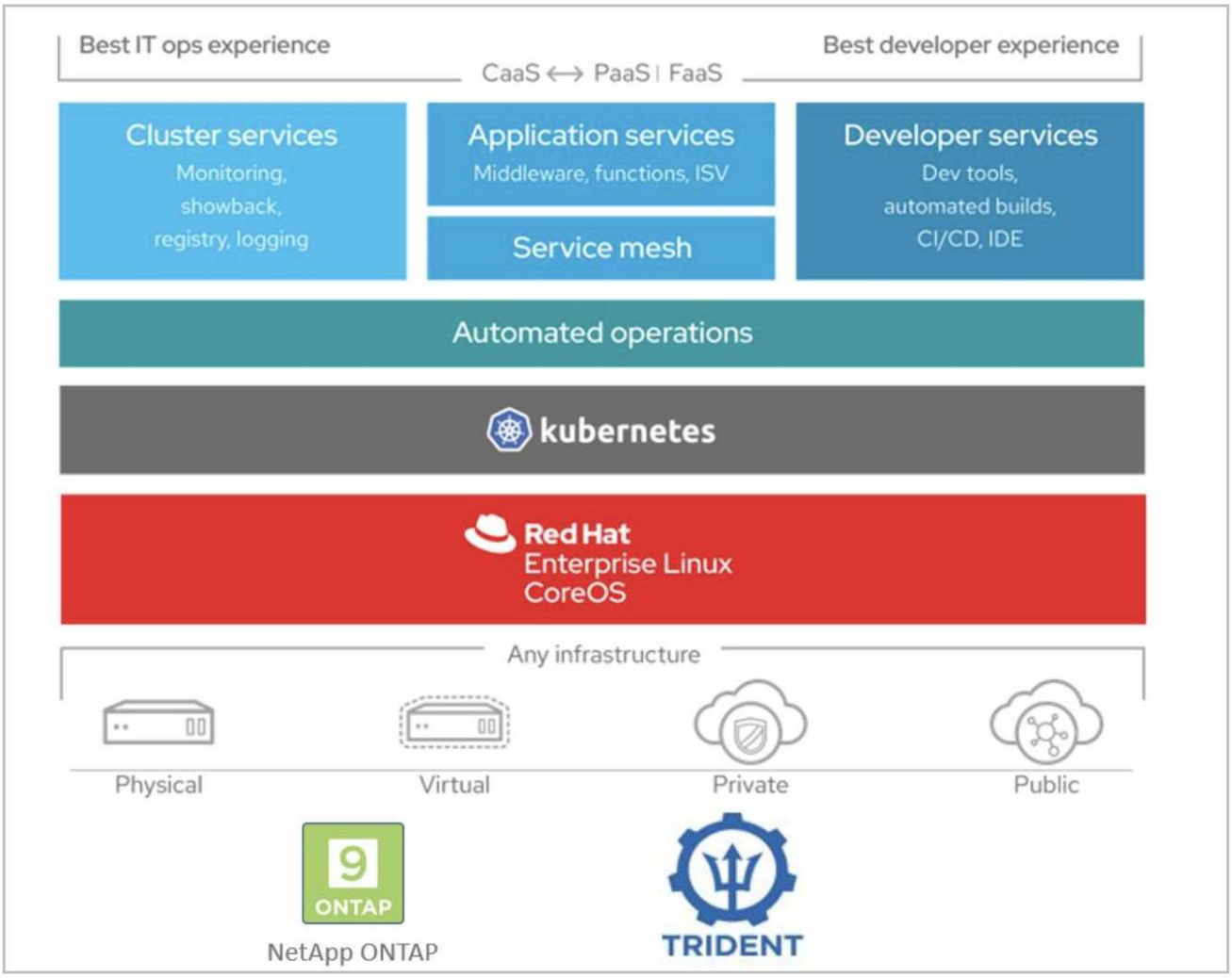
Element は自動化用に設計されました。すべてのストレージ機能は API を通じて利用できます。これらの API は、UI がシステムを制御するために使用する唯一の方法です。

NetAppストレージ統合

NetApp Tridentと Red Hat OpenShift の統合について学ぶ

OpenShift Virtualization ソリューションのアプリケーションおよび永続ストレージ管理用に検証されたNetApp Trident Protect について説明します。

Trident は、 NetAppが管理するオープンソースのストレージ プロビジョナーおよびオーケストレーターであり、 NetApp Trident Protect は、Red Hat OpenShift などのコンテナー ベースの環境での永続データのオーケストレーションと管理に役立ちます。



次のページには、Red Hat OpenShift with NetAppソリューションにおけるアプリケーションおよび永続ストレージ管理用に検証されたNetApp製品に関する追加情報が記載されています。

- "Tridentドキュメント"
- "Tridentプロテクトのドキュメント"

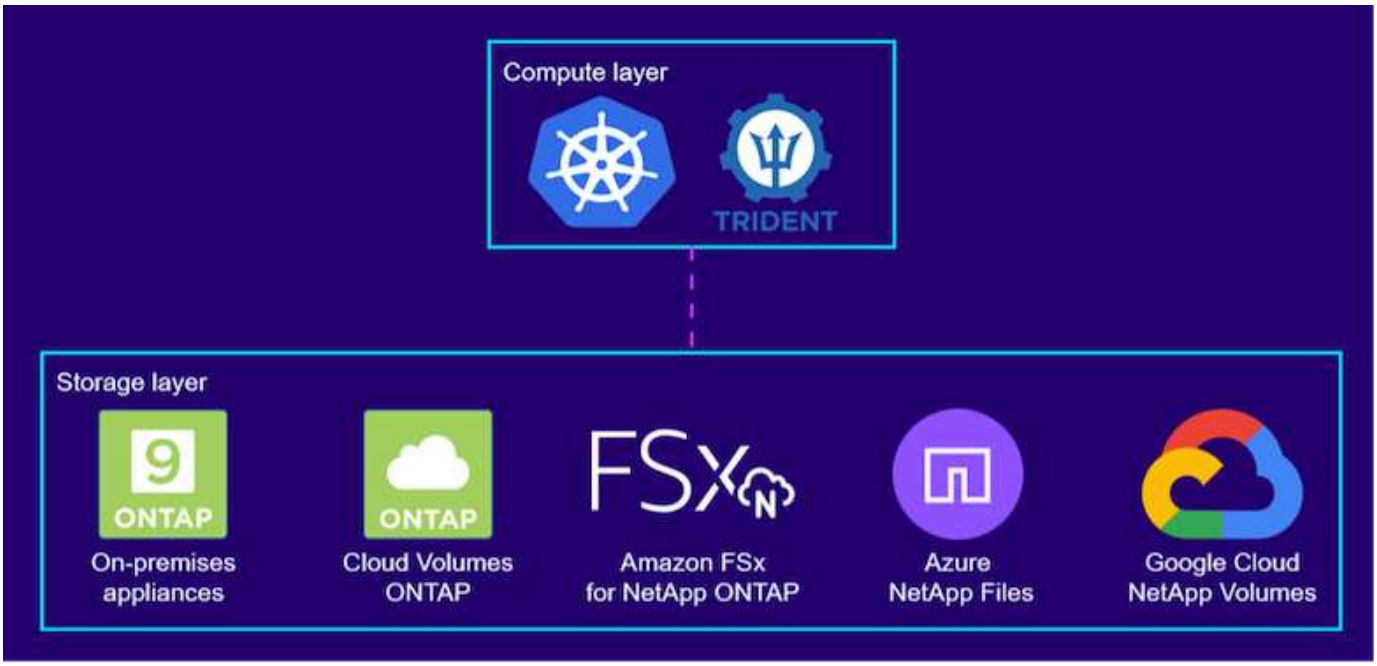
NetAppTrident

Tridentの概要

Trident は、Red Hat OpenShift を含むコンテナおよび Kubernetes ディストリビューション向けのオープンソースで完全にサポートされているストレージ オーケストレーターです。 Trident は、 NetApp ONTAPおよび Element ストレージ システムを含むNetAppストレージ ポートフォリオ全体と連携し、NFS および iSCSI 接続もサポートします。 Trident は、ストレージ管理者の介入を必要とせずに、エンドユーザーがNetAppストレージ システムからストレージをプロビジョニングおよび管理できるようにすることで、DevOps ワークフローを加速します。

管理者は、プロジェクトのニーズと、圧縮、特定のディスク タイプ、特定のレベルのパフォーマンスを保証する QoS レベルなどの高度なストレージ機能を有効にするストレージ システム モデルに基づいて、さまざ

まなストレージ バックエンドを構成できます。これらのバックエンドは定義されると、開発者がプロジェクト内で使用して永続ボリューム要求 (PVC) を作成し、オンデマンドで永続ストレージをコンテナーに接続できるようになります。



Tridentは開発サイクルが速く、Kubernetes と同様に年に 4 回リリースされます。

どのバージョンのTridentがどのKubernetesディストリビューションでテストされているかのサポートマトリックスは以下から参照できます。 "ここをクリックしてください。" 。

詳細は"Trident製品ドキュメント"インストールと構成の詳細については、こちらをご覧ください。

Tridentをダウンロード

デプロイされたユーザー クラスタにTridentをインストールし、永続ボリュームをプロビジョニングするには、次の手順を実行します。

1. インストール アーカイブを管理ワークステーションにダウンロードし、内容を抽出します。 Tridentの現在のバージョンはダウンロードできます "ここをクリックしてください。"。
2. ダウンロードしたバンドルからTridentインストールを抽出します。

```
[netapp-user@rhel7 ~]$ tar -xzf trident-installer-22.01.0.tar.gz
[netapp-user@rhel7 ~]$ cd trident-installer/
[netapp-user@rhel7 trident-installer]$
```

Helm で**Trident Operator** をインストールする

1. まずユーザークラスタの場所を設定します `kubeconfig`Trident にはこのファイルを渡すオプションがないため、このファイルを環境変数として設定して参照する必要がないようにする必要があります。

```
[netapp-user@rhel7 trident-installer]$ export KUBECONFIG=~/ocp-
install/auth/kubeconfig
```

2. Helm コマンドを実行して、ユーザー クラスタに trident 名前空間を作成しながら、helm ディレクトリの tarball からTridentオペレーターをインストールします。

```
[netapp-user@rhel7 trident-installer]$ helm install trident
helm/trident-operator-22.01.0.tgz --create-namespace --namespace trident
NAME: trident
LAST DEPLOYED: Fri May  7 12:54:25 2021
NAMESPACE: trident
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
Thank you for installing trident-operator, which will deploy and manage
NetApp's Trident CSI
storage provisioner for Kubernetes.

Your release is named 'trident' and is installed into the 'trident'
namespace.
Please note that there must be only one instance of Trident (and
trident-operator) in a Kubernetes cluster.

To configure Trident to manage storage resources, you will need a copy
of tridentctl, which is
available in pre-packaged Trident releases.  You may find all Trident
releases and source code
online at https://github.com/NetApp/trident.

To learn more about the release, try:

  $ helm status trident
  $ helm get all trident
```

3. 名前空間で実行されているポッドを確認するか、tridentctl バイナリを使用してインストールされているバージョンを確認することで、 Tridentが正常にインストールされていることを確認できます。

```
[netapp-user@rhel7 trident-installer]$ oc get pods -n trident
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-5z45l                   1/2     Running   2          30s
trident-csi-696b685cf8-htdb2        6/6     Running   0          30s
trident-csi-b74p2                   2/2     Running   0          30s
trident-csi-lrw4n                   2/2     Running   0          30s
trident-operator-7c748d957-gr2gw    1/1     Running   0          36s

[netapp-user@rhel7 trident-installer]$ ./tridentctl -n trident version
+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 22.01.0        | 22.01.0        |
+----------------+----------------+
```

場合によっては、顧客環境でTrident展開のカスタマイズが必要になることがあります。このような場合、 Tridentオペレーターを手動でインストールし、含まれるマニフェストを更新してデプロイメントをカスタマイズすることもできます。

Trident Operatorを手動でインストールする

1. まず、ユーザークラスタの場所を設定します。 `kubeconfig` Trident にはこのファイルを渡すオプションがないため、このファイルを環境変数として設定して参照する必要がないようにする必要があります。

```
[netapp-user@rhel7 trident-installer]$ export KUBECONFIG=~/ocp-install/auth/kubeconfig
```

2. その `trident-installer`ディレクトリには、必要なすべてのリソースを定義するためのマニフェストが含まれています。適切なマニフェストを使用して、 `TridentOrchestrator`カスタム リソース定義。

```
[netapp-user@rhel7 trident-installer]$ oc create -f deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.16.yaml
customresourcedefinition.apiextensions.k8s.io/tridentorchestrators.trident.netapp.io created
```

3. 存在しない場合は、提供されたマニフェストを使用してクラスターにTrident名前空間を作成します。

```
[netapp-user@rhel7 trident-installer]$ oc apply -f deploy/namespace.yaml
namespace/trident created
```

4. Tridentオペレーターの展開に必要なリソースを作成します。 `ServiceAccount`オペレーターにとって、 `ClusterRole`そして `ClusterRoleBinding`に `ServiceAccount`、専用の

PodSecurityPolicy、または演算子自体。

```
[netapp-user@rhel7 trident-installer]$ oc create -f deploy/bundle.yaml
serviceaccount/trident-operator created
clusterrole.rbac.authorization.k8s.io/trident-operator created
clusterrolebinding.rbac.authorization.k8s.io/trident-operator created
deployment.apps/trident-operator created
podsecuritypolicy.policy/tridentoperatorpods created
```

5. オペレーターをデプロイした後、次のコマンドを使用してオペレーターのステータスを確認できます。

```
[netapp-user@rhel7 trident-installer]$ oc get deployment -n trident
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
trident-operator   1/1     1            1           23s
[netapp-user@rhel7 trident-installer]$ oc get pods -n trident
NAME                                READY   STATUS    RESTARTS   AGE
trident-operator-66f48895cc-lzczk   1/1     Running   0          41s
```

6. オペレーターがデプロイされたので、これを使用してTridentをインストールできるようになりました。これには、 TridentOrchestrator 。

```
[netapp-user@rhel7 trident-installer]$ oc create -f
deploy/crds/tridentorchestrator_cr.yaml
tridentorchestrator.trident.netapp.io/trident created
[netapp-user@rhel7 trident-installer]$ oc describe torc trident
Name:         trident
Namespace:
Labels:       <none>
Annotations:  <none>
API Version:  trident.netapp.io/v1
Kind:         TridentOrchestrator
Metadata:
  Creation Timestamp:  2021-05-07T17:00:28Z
  Generation:          1
  Managed Fields:
    API Version:  trident.netapp.io/v1
    Fields Type:  FieldsV1
    fieldsV1:
      f:spec:
        .:
        f:debug:
        f:namespace:
    Manager:      kubectl-create
    Operation:    Update
```

```
    Time:         2021-05-07T17:00:28Z
    API Version:  trident.netapp.io/v1
    Fields Type:  FieldsV1
    fieldsV1:
      f:status:
        .:
        f:currentInstallationParams:
          .:
          f:IPv6:
          f:autosupportHostname:
          f:autosupportimage:
          f:autosupportProxy:
          f:autosupportSerialNumber:
          f:debug:
          f:enableNodePrep:
          f:imagePullSecrets:
          f:imageRegistry:
          f:k8sTimeout:
          f:kubeletDir:
          f:logFormat:
          f:silenceAutosupport:
          f:tridentimage:
        f:message:
        f:namespace:
        f:status:
        f:version:
    Manager:         trident-operator
    Operation:       Update
    Time:            2021-05-07T17:00:28Z
  Resource Version:  931421
  Self Link:
/apis/trident.netapp.io/v1/tridentorchestrators/trident
  UID:               8a26a7a6-dde8-4d55-9b66-a7126754d81f
Spec:
  Debug:      true
  Namespace:  trident
Status:
  Current Installation Params:
    IPv6:                      false
    Autosupport Hostname:
    Autosupport image:         netapp/trident-autosupport:21.01
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                     true
    Enable Node Prep:          false
    Image Pull Secrets:
```

```
    Image Registry:
    k8sTimeout:           30
    Kubelet Dir:          /var/lib/kubelet
    Log Format:           text
    Silence Autosupport:  false
    Trident image:        netapp/trident:22.01.0
  Message:                Trident installed
  Namespace:              trident
  Status:                 Installed
  Version:                v22.01.0
Events:
  Type    Reason      Age   From                           Message
  ----    ------      ----  ----                           -------
  Normal  Installing  80s   trident-operator.netapp.io  Installing
Trident
  Normal  Installed   68s   trident-operator.netapp.io  Trident
installed
```

7. 名前空間で実行されているポッドを確認するか、tridentctl バイナリを使用してインストールされているバージョンを確認することで、 Tridentが正常にインストールされていることを確認できます。

```
[netapp-user@rhel7 trident-installer]$ oc get pods -n trident
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-bb64c6cb4-lmd6h         6/6     Running   0          82s
trident-csi-gn59q                   2/2     Running   0          82s
trident-csi-m4szj                   2/2     Running   0          82s
trident-csi-sb9k9                   2/2     Running   0          82s
trident-operator-66f48895cc-lzczk   1/1     Running   0          2m39s

[netapp-user@rhel7 trident-installer]$ ./tridentctl -n trident version
+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 22.01.0        | 22.01.0        |
+----------------+----------------+
```

ストレージ用のワーカーノードを準備する

NFS

ほとんどの Kubernetes ディストリビューションには、Red Hat OpenShift を含め、NFS バックエンドをマウントするためのパッケージとユーティリティがデフォルトでインストールされています。

ただし、NFSv3 には、クライアントとサーバー間の同時実行をネゴシエートするメカニズムはありません。したがって、サーバーが接続のウィンドウ サイズを縮小することなく NFS 接続の最高のパフォーマンスを確保するには、クライアント側の sunrpc スロット テーブル エントリの最大数を、サーバーでサポートされて

いる値と手動で同期する必要があります。

ONTAPの場合、サポートされる sunrpc スロット テーブル エントリの最大数は 128 です。つまり、 ONTAP は一度に 128 の同時 NFS 要求を処理できます。ただし、デフォルトでは、Red Hat CoreOS/Red Hat Enterprise Linux では接続ごとに最大 65,536 個の sunrpc スロット テーブル エントリがあります。この値を 128 に設定する必要があり、これは OpenShift の Machine Config Operator (MCO) を使用して実行できます。

OpenShift ワーカー ノードの最大 sunrpc スロット テーブル エントリを変更するには、次の手順を実行します。

1. OCP Web コンソールにログインし、[Compute] > [Machine Configs] に移動します。 「マシン構成の作成」をクリックします。 YAML ファイルをコピーして貼り付け、「作成」をクリックします。

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: 98-worker-nfs-rpc-slot-tables
  labels:
    machineconfiguration.openshift.io/role: worker
spec:
  config:
    ignition:
      version: 3.2.0
    storage:
      files:
        - contents:
            source: data:text/plain;charset=utf-
8;base64,b3B0aW9ucyBzdW5ycGMgdGNwX21heF9zbG90X3RhYmxlX2VudHJpZXM9MTI4Cg=
=
          filesystem: root
          mode: 420
          path: /etc/modprobe.d/sunrpc.conf
```

2. MCO が作成された後、すべてのワーカー ノードに構成を適用し、1 つずつ再起動する必要があります。全体のプロセスには約20～30分かかります。マシン構成が適用されているかどうかを確認するには、`oc get mcp`ワーカーのマシン構成プールが更新されていることを確認します。

```
[netapp-user@rhel7 openshift-deploy]$ oc get mcp
NAME     CONFIG                                             UPDATED   UPDATING
DEGRADED
master   rendered-master-a520ae930e1d135e0dee7168   True      False
False
worker   rendered-worker-de321b36eeba62df41feb7bc   True      False
False
```

iSCSI

iSCSI プロトコルを介してブロック ストレージ ボリュームをマッピングできるようにワーカー ノードを準備するには、その機能をサポートするために必要なパッケージをインストールする必要があります。

Red Hat OpenShift では、クラスターのデプロイ後に MCO (Machine Config Operator) を適用することでこれが処理されます。

iSCSI サービスを実行するようにワーカーノードを構成するには、次の手順を実行します。

1. OCP Web コンソールにログインし、[Compute] > [Machine Configs] に移動します。 「マシン構成の作成」をクリックします。 YAML ファイルをコピーして貼り付け、「作成」をクリックします。

   マルチパスを使用しない場合:

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 99-worker-element-iscsi
spec:
  config:
    ignition:
      version: 3.2.0
    systemd:
      units:
        - name: iscsid.service
          enabled: true
          state: started
  osImageURL: ""
```

   マルチパスを使用する場合:

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: 99-worker-ontap-iscsi
  labels:
    machineconfiguration.openshift.io/role: worker
spec:
  config:
    ignition:
      version: 3.2.0
    storage:
      files:
      - contents:
          source: data:text/plain;charset=utf-
8;base64,ZGVmYXVsdHMgewogICAgICAgIHVzZXJfZnJpZW5kbHlfbmFtZXMgbm8KICAgICA
gICBmaW5kX211bHRpcGF0aHMgbm8KfQoKYmxhY2tsaXN0X2V4Y2VwdGlvbnMgewogICAgICA
gIHByb3BlcnR5ICIoU0NTSV9JREVOVF98SURfV1dOKSIKfQoKYmxhY2tsaXN0IHsKfQoK
          verification: {}
        filesystem: root
        mode: 400
        path: /etc/multipath.conf
    systemd:
      units:
        - name: iscsid.service
          enabled: true
          state: started
        - name: multipathd.service
          enabled: true
          state: started
  osImageURL: ""
```

2. 構成が作成された後、その構成をワーカーノードに適用して再ロードするまでに約 20 ～ 30 分かかります。マシン構成が適用されているかどうかを確認するには、 `oc get mcp`ワーカーのマシン構成プールが更新されていることを確認します。ワーカー ノードにログインして、iscsid サービスが実行されていること (マルチパスを使用している場合は multipathd サービスも実行されていること) を確認することもできます。

```
[netapp-user@rhel7 openshift-deploy]$ oc get mcp
NAME     CONFIG                                             UPDATED   UPDATING
DEGRADED
master   rendered-master-a520ae930e1d135e0dee7168   True      False
False
worker   rendered-worker-de321b36eeba62df41feb7bc   True      False
False

[netapp-user@rhel7 openshift-deploy]$ ssh core@10.61.181.22 sudo
systemctl status iscsid
● iscsid.service - Open-iSCSI
   Loaded: loaded (/usr/lib/systemd/system/iscsid.service; enabled;
vendor preset: disabled)
   Active: active (running) since Tue 2021-05-26 13:36:22 UTC; 3 min ago
     Docs: man:iscsid(8)
           man:iscsiadm(8)
 Main PID: 1242 (iscsid)
   Status: "Ready to process requests"
    Tasks: 1
   Memory: 4.9M
      CPU: 9ms
   CGroup: /system.slice/iscsid.service
           └─1242 /usr/sbin/iscsid -f

[netapp-user@rhel7 openshift-deploy]$ ssh core@10.61.181.22 sudo
systemctl status multipathd
 ● multipathd.service - Device-Mapper Multipath Device Controller
   Loaded: loaded (/usr/lib/systemd/system/multipathd.service; enabled;
vendor preset: enabled)
   Active: active (running) since Tue 2021-05-26 13:36:22 UTC; 3 min ago
  Main PID: 918 (multipathd)
    Status: "up"
    Tasks: 7
    Memory: 13.7M
    CPU: 57ms
    CGroup: /system.slice/multipathd.service
            └─918 /sbin/multipathd -d -s
```

また、MachineConfigが正常に適用され、サービスが期待どおりに開始されたことを確認するには、次のコマンドを実行します。 `oc debug`適切なフラグを指定したコマンド。

ストレージシステムのバックエンドを作成する

Trident Operator のインストールが完了したら、使用している特定のNetAppストレージ プラットフォームのバックエンドを構成する必要があります。 Tridentのセットアップと構成を続行するには、以下のリンクに従

ってください。

- "NetApp ONTAP NFS"
- "NetApp ONTAP iSCSI"
- "NetApp ElementiSCSI"

NetApp ONTAP NFS構成

TridentとNetApp ONTAPストレージ システムとの統合を有効にするには、ストレージ システムとの通信を可能にするバックエンドを作成する必要があります。

1. ダウンロードしたインストールアーカイブには、サンプルのバックエンドファイルが含まれています。 `sample-input`フォルダー階層。 NFSを提供するNetApp ONTAPシステムの場合は、 `backend-ontap-nas.json`ファイルを作業ディレクトリに移動し、編集します。

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/backends-
samples/ontap-nas/backend-ontap-nas.json ./
[netapp-user@rhel7 trident-installer]$ vi backend-ontap-nas.json
```

2. このファイル内の backendName、managementLIF、dataLIF、svm、username、および password の値を編集します。

```
{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "backendName": "ontap-nas+10.61.181.221",
    "managementLIF": "172.21.224.201",
    "dataLIF": "10.61.181.221",
    "svm": "trident_svm",
    "username": "cluster-admin",
    "password": "password"
}
```

> 簡単に識別できるように、カスタム backendName 値を storageDriverName と NFS を提供している dataLIF の組み合わせとして定義するのがベスト プラクティスです。

3. このバックエンド ファイルを配置したら、次のコマンドを実行して最初のバックエンドを作成します。

```
[netapp-user@rhel7 trident-installer]$ ./tridentctl -n trident create
backend -f backend-ontap-nas.json
+------------------------+----------------
+--------------------------------------+--------+---------+
|          NAME          | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+------------------------+----------------
+--------------------------------------+--------+---------+
| ontap-nas+10.61.181.221 | ontap-nas      | be7a619d-c81d-445c-b80c-
5c87a73c5b1e | online |       0 |
+------------------------+----------------
+--------------------------------------+--------+---------+
```

4. バックエンドを作成したら、次にストレージ クラスを作成する必要があります。バックエンドと同様に、sample-inputs フォルダーには、環境に合わせて編集できるサンプル ストレージ クラス ファイルがあります。それを作業ディレクトリにコピーし、作成されたバックエンドを反映するために必要な編集を行います。

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/storage-class-
samples/storage-class-csi.yaml.templ ./storage-class-basic.yaml
[netapp-user@rhel7 trident-installer]$ vi storage-class-basic.yaml
```

5. このファイルで編集する必要があるのは、 `backendType`新しく作成されたバックエンドのストレージ ドライバーの名前に値を設定します。また、後の手順で参照する必要がある名前フィールドの値にも注意してください。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: basic-csi
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
```

> オプションフィールドがあります `fsType`このファイルで定義されます。この行は NFS バックエンドでは削除できます。

6. 実行 `oc`ストレージ クラスを作成するコマンド。

```
[netapp-user@rhel7 trident-installer]$ oc create -f storage-class-
basic.yaml
storageclass.storage.k8s.io/basic-csi created
```

7. ストレージ クラスを作成したら、最初の永続ボリューム要求 (PVC) を作成する必要があります。サンプルがあります `pvc-basic.yaml`このアクションを実行するために使用できるファイルは、sample-inputs にもあります。

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/pvc-samples/pvc-
basic.yaml ./
[netapp-user@rhel7 trident-installer]$ vi pvc-basic.yaml
```

8. このファイルに対して行う必要がある唯一の編集は、 `storageClassName`フィールドは、先ほど作成したものと一致します。 PVC 定義は、プロビジョニングするワークロードの必要に応じてさらにカスタマイズできます。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: basic
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

9. PVCを作成するには、 `oc`指示。作成するバックアップボリュームのサイズによっては、作成に時間がかかる場合があります。そのため、プロセスが完了するまで監視することができます。

```
[netapp-user@rhel7 trident-installer]$ oc create -f pvc-basic.yaml
persistentvolumeclaim/basic created

[netapp-user@rhel7 trident-installer]$ oc get pvc
NAME    STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
basic   Bound    pvc-b4370d37-0fa4-4c17-bd86-94f96c94b42d   1Gi
RWO            basic-csi      7s
```

NetApp ONTAP iSCSI構成

TridentとNetApp ONTAPストレージ システムとの統合を有効にするには、ストレージ システムとの通信を可能にするバックエンドを作成する必要があります。

1. ダウンロードしたインストールアーカイブには、サンプルのバックエンドファイルが含まれています。 `sample-input`フォルダー階層。 iSCSIを提供するNetApp ONTAPシステムの場合は、 `backend-ontap-san.json`ファイルを作業ディレクトリに移動し、編集します。

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/backends-
samples/ontap-san/backend-ontap-san.json ./
[netapp-user@rhel7 trident-installer]$ vi backend-ontap-san.json
```

2. このファイル内の managementLIF、dataLIF、svm、ユーザー名、およびパスワードの値を編集します。

```
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "managementLIF": "172.21.224.201",
  "dataLIF": "10.61.181.240",
  "svm": "trident_svm",
  "username": "admin",
  "password": "password"
}
```

3. このバックエンド ファイルを配置したら、次のコマンドを実行して最初のバックエンドを作成します。

```
[netapp-user@rhel7 trident-installer]$ ./tridentctl -n trident create
backend -f backend-ontap-san.json
+------------------------+----------------
+--------------------------------------+--------+---------+
|          NAME          | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+------------------------+----------------
+--------------------------------------+--------+---------+
| ontapsan_10.61.181.241 | ontap-san      | 6788533c-7fea-4a35-b797-
fb9bb3322b91 | online |       0 |
+------------------------+----------------
+--------------------------------------+--------+---------+
```

4. バックエンドを作成したら、次にストレージ クラスを作成する必要があります。バックエンドと同様に、sample-inputs フォルダーには、環境に合わせて編集できるサンプル ストレージ クラス ファイルがあります。それを作業ディレクトリにコピーし、作成されたバックエンドを反映するために必要な編集を行います。

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/storage-class-
samples/storage-class-csi.yaml.templ ./storage-class-basic.yaml
[netapp-user@rhel7 trident-installer]$ vi storage-class-basic.yaml
```

5. このファイルで編集する必要があるのは、 `backendType`新しく作成されたバックエンドのストレージ ドライバーの名前に値を設定します。また、後の手順で参照する必要がある名前フィールドの値にも注意してください。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: basic-csi
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
```

オプションフィールドがあります `fsType`このファイルで定義されます。 iSCSI バックエンドでは、この値を特定の Linux ファイルシステム タイプ (XFS、ext4 など) に設定するか、削除して OpenShift が使用するファイルシステムを決定できるようにすることができます。

6. 実行 `oc`ストレージ クラスを作成するコマンド。

```
[netapp-user@rhel7 trident-installer]$ oc create -f storage-class-
basic.yaml
storageclass.storage.k8s.io/basic-csi created
```

7. ストレージ クラスを作成したら、最初の永続ボリューム要求 (PVC) を作成する必要があります。サンプルがあります `pvc-basic.yaml`このアクションを実行するために使用できるファイルは、sample-inputs にもあります。

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/pvc-samples/pvc-
basic.yaml ./
[netapp-user@rhel7 trident-installer]$ vi pvc-basic.yaml
```

8. このファイルに対して行う必要がある唯一の編集は、 `storageClassName`フィールドは、先ほど作成したものと一致します。 PVC 定義は、プロビジョニングするワークロードの必要に応じてさらにカスタマイズできます。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: basic
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

9. PVCを作成するには、 `oc`指示。作成するバックアップボリュームのサイズによっては、作成に時間がかかる場合があります。そのため、プロセスが完了するまで監視することができます。

```
[netapp-user@rhel7 trident-installer]$ oc create -f pvc-basic.yaml
persistentvolumeclaim/basic created

[netapp-user@rhel7 trident-installer]$ oc get pvc
NAME    STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
basic   Bound    pvc-7ceac1ba-0189-43c7-8f98-094719f7956c   1Gi
RWO            basic-csi      3s
```

NetApp Element iSCSI 構成

TridentとNetApp Elementストレージ システムとの統合を有効にするには、iSCSI プロトコルを使用してストレージ システムとの通信を可能にするバックエンドを作成する必要があります。

1. ダウンロードしたインストールアーカイブには、サンプルのバックエンドファイルが含まれています。 `sample-input`フォルダー階層。 iSCSIを提供するNetApp Elementシステムの場合は、 `backend-solidfire.json`ファイルを作業ディレクトリに移動し、編集します。

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/backends-
samples/solidfire/backend-solidfire.json ./
[netapp-user@rhel7 trident-installer]$ vi ./backend-solidfire.json
```

   a. ユーザー、パスワード、MVIP値を編集します。 `EndPoint`ライン。

   b. 編集する `SVIP`価値。

```
 {
    "version": 1,
    "storageDriverName": "solidfire-san",
    "Endpoint": "https://trident:password@172.21.224.150/json-
rpc/8.0",
    "SVIP": "10.61.180.200:3260",
    "TenantName": "trident",
    "Types": [{"Type": "Bronze", "Qos": {"minIOPS": 1000, "maxIOPS":
2000, "burstIOPS": 4000}},
              {"Type": "Silver", "Qos": {"minIOPS": 4000, "maxIOPS":
6000, "burstIOPS": 8000}},
              {"Type": "Gold", "Qos": {"minIOPS": 6000, "maxIOPS":
8000, "burstIOPS": 10000}}]
}
```

2. このバックエンド ファイルを配置したら、次のコマンドを実行して最初のバックエンドを作成します。

```
[netapp-user@rhel7 trident-installer]$ ./tridentctl -n trident create
backend -f backend-solidfire.json
+------------------------+----------------
+--------------------------------------+--------+---------+
|          NAME          | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+------------------------+----------------
+--------------------------------------+--------+---------+
| solidfire_10.61.180.200 | solidfire-san  | b90783ee-e0c9-49af-8d26-
3ea87ce2efdf | online |       0 |
+------------------------+----------------
+--------------------------------------+--------+---------+
```

3. バックエンドを作成したら、次にストレージ クラスを作成する必要があります。バックエンドと同様に、sample-inputs フォルダーには、環境に合わせて編集できるサンプル ストレージ クラス ファイルがあります。それを作業ディレクトリにコピーし、作成されたバックエンドを反映するために必要な編集を行います。

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/storage-class-
samples/storage-class-csi.yaml.templ ./storage-class-basic.yaml
[netapp-user@rhel7 trident-installer]$ vi storage-class-basic.yaml
```

4. このファイルで編集する必要があるのは、 `backendType`新しく作成されたバックエンドのストレージ ドライバーの名前に値を設定します。また、後の手順で参照する必要がある名前フィールドの値にも注意してください。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: basic-csi
provisioner: csi.trident.netapp.io
parameters:
  backendType: "solidfire-san"
```

> オプションフィールドがあります `fsType`このファイルで定義されます。 iSCSI バックエンドでは、この値を特定の Linux ファイルシステム タイプ (XFS、ext4 など) に設定することも、削除して OpenShift が使用するファイルシステムを決定できるようにすることもできます。

5. 実行 `oc`ストレージ クラスを作成するコマンド。

```
[netapp-user@rhel7 trident-installer]$ oc create -f storage-class-
basic.yaml
storageclass.storage.k8s.io/basic-csi created
```

6. ストレージ クラスを作成したら、最初の永続ボリューム要求 (PVC) を作成する必要があります。サンプルがあります `pvc-basic.yaml`このアクションを実行するために使用できるファイルは、sample-inputs にもあります。

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/pvc-samples/pvc-
basic.yaml ./
[netapp-user@rhel7 trident-installer]$ vi pvc-basic.yaml
```

7. このファイルに対して行う必要がある唯一の編集は、 `storageClassName`フィールドは、先ほど作成したものと一致します。 PVC 定義は、プロビジョニングするワークロードの必要に応じてさらにカスタマイズできます。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: basic
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

8. PVCを作成するには、 `oc`指示。作成するバックアップボリュームのサイズによっては、作成に時間がかかる場合があります。そのため、プロセスが完了するまで監視することができます。

```
[netapp-user@rhel7 trident-installer]$ oc create -f pvc-basic.yaml
persistentvolumeclaim/basic created

[netapp-user@rhel7 trident-installer]$ oc get pvc
NAME    STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
basic   Bound    pvc-3445b5cc-df24-453d-a1e6-b484e874349d   1Gi
RWO            basic-csi      5s
```

高度な設定オプション

ロードバランサーのオプションを調べる

ロードバランサーのオプションの検討: **Red Hat OpenShift**と**NetApp**

ほとんどの場合、Red Hat OpenShift はルートを通じてアプリケーションを外部に公開します。サービスは、外部からアクセス可能なホスト名を与えることによって公開されます。定義されたルートとそのサービスによって識別されるエンドポイントは、OpenShift ルーターによって使用され、この名前付き接続を外部クライアントに提供できます。

ただし、場合によっては、アプリケーションで適切なサービスを公開するために、カスタマイズされたロードバランサーの展開と構成が必要になります。その一例がNetApp Trident Protect です。このニーズを満たすために、いくつかのカスタム ロード バランサー オプションを評価しました。このセクションでは、それらのインストールと構成について説明します。

次のページには、Red Hat OpenShift with NetAppソリューションで検証されたロードバランサ オプションに関する追加情報が記載されています。

- "メタルLB"
- "F5 ビッグIP"

MetalLB ロードバランサーのインストール: **Red Hat OpenShift** と**NetApp**

このページでは、MetalLB ロード バランサーのインストールおよび構成の手順について説明します。

MetalLB は、OpenShift クラスターにインストールされるセルフホスト型ネットワーク ロード バランサーであり、クラウド プロバイダーで実行されないクラスターでロード バランサー タイプの OpenShift サービスの作成を可能にします。 LoadBalancer サービスをサポートするために連携して動作する MetalLB の 2 つの主な機能は、アドレス割り当てと外部アナウンスです。

MetalLB 構成オプション

MetalLB は、OpenShift クラスターの外部の LoadBalancer サービスに割り当てられた IP アドレスをアナウンスする方法に基づいて、次の 2 つのモードで動作します。

- *レイヤー2モード*このモードでは、OpenShift クラスター内の 1 つのノードがサービスの所有権を取得し、その IP に対する ARP 要求に応答して、OpenShift クラスターの外部からアクセスできるようにします。ノードのみが IP をアドバタイズするため、帯域幅のボトルネックとフェイルオーバー速度の制限が発生します。詳細については、ドキュメントを参照してください。"ここをクリックしてください。" 。
- *BGP モード*このモードでは、OpenShift クラスター内のすべてのノードがルーターとの BGP ピアリング セッションを確立し、トラフィックをサービス IP に転送するためのルートをアドバタイズします。このための前提条件は、MetalLB をそのネットワーク内のルーターに統合することです。 BGP のハッシュメカニズムにより、サービスの IP とノードのマッピングが変更される場合には、一定の制限があります。詳細については、ドキュメントを参照してください。"ここをクリックしてください。" 。

このドキュメントでは、MetalLB をレイヤー 2 モードで構成します。

MetalLBロードバランサのインストール

1. MetalLB リソースをダウンロードします。

```
[netapp-user@rhel7 ~]$ wget
https://raw.githubusercontent.com/metallb/metallb/v0.10.2/manifests/name
space.yaml
[netapp-user@rhel7 ~]$ wget
https://raw.githubusercontent.com/metallb/metallb/v0.10.2/manifests/meta
llb.yaml
```

2. ファイルを編集 `metallb.yaml`削除する `spec.template.spec.securityContext`コントローラーのデプロイメントとスピーカーの DaemonSet から。

   削除する行:

```
securityContext:
  runAsNonRoot: true
  runAsUser: 65534
```

3. 作成する `metallb-system`名前空間。

```
[netapp-user@rhel7 ~]$ oc create -f namespace.yaml
namespace/metallb-system created
```

4. MetalLB CR を作成します。

```
[netapp-user@rhel7 ~]$ oc create -f metallb.yaml
podsecuritypolicy.policy/controller created
podsecuritypolicy.policy/speaker created
serviceaccount/controller created
serviceaccount/speaker created
clusterrole.rbac.authorization.k8s.io/metallb-system:controller created
clusterrole.rbac.authorization.k8s.io/metallb-system:speaker created
role.rbac.authorization.k8s.io/config-watcher created
role.rbac.authorization.k8s.io/pod-lister created
role.rbac.authorization.k8s.io/controller created
clusterrolebinding.rbac.authorization.k8s.io/metallb-system:controller
created
clusterrolebinding.rbac.authorization.k8s.io/metallb-system:speaker
created
rolebinding.rbac.authorization.k8s.io/config-watcher created
rolebinding.rbac.authorization.k8s.io/pod-lister created
rolebinding.rbac.authorization.k8s.io/controller created
daemonset.apps/speaker created
deployment.apps/controller created
```

5. MetalLB スピーカーを構成する前に、スピーカーの DaemonSet に昇格した権限を付与して、ロード バランサーを動作させるために必要なネットワーク構成を実行できるようにします。

```
[netapp-user@rhel7 ~]$ oc adm policy add-scc-to-user privileged -n
metallb-system -z speaker
clusterrole.rbac.authorization.k8s.io/system:openshift:scc:privileged
added: "speaker"
```

6. MetalLBを構成するには、 `ConfigMap`の中で `metallb-system`名前空間。

```
[netapp-user@rhel7 ~]$ vim metallb-config.yaml

apiVersion: v1
kind: ConfigMap
metadata:
  namespace: metallb-system
  name: config
data:
  config: |
    address-pools:
    - name: default
      protocol: layer2
      addresses:
      - 10.63.17.10-10.63.17.200

[netapp-user@rhel7 ~]$ oc create -f metallb-config.yaml
configmap/config created
```

7. ロードバランサ サービスが作成されると、MetalLB はサービスに外部 IP を割り当て、ARP 要求に応答して IP アドレスをアドバタイズします。

MetalLBをBGPモードで設定する場合は、上記の手順6をスキップし、MetalLBドキュメントの手順に従ってください。"ここをクリックしてください。" 。

F5 BIG-IPロードバランサーのインストール

F5 BIG-IP は、L4-L7 ロード バランシング、SSL/TLS オフロード、DNS、ファイアウォールなど、幅広い高度な実稼働レベルのトラフィック管理およびセキュリティ サービスを提供するアプリケーション配信コントローラ (ADC) です。これらのサービスにより、アプリケーションの可用性、セキュリティ、パフォーマンスが大幅に向上します。

F5 BIG-IP は、専用ハードウェア、クラウド、オンプレミスの仮想アプライアンスなど、さまざまな方法で導入および使用できます。要件に応じて F5 BIG-IP を調査して展開するには、ここにあるドキュメントを参照してください。

F5 BIG-IP サービスと Red Hat OpenShift を効率的に統合するために、F5 は BIG-IP Container Ingress Service (CIS) を提供しています。 CIS は、特定のカスタム リソース定義 (CRD) の OpenShift API を監視し、F5 BIG-IP システム構成を管理するコントローラー ポッドとしてインストールされます。 F5 BIG-IP CIS は、OpenShift のサービス タイプ LoadBalancers および Routes を制御するように構成できます。

さらに、LoadBalancer タイプにサービスを提供する自動 IP アドレス割り当てには、F5 IPAM コントローラーを利用できます。 F5 IPAM コントローラーは、ipamLabel アノテーションを使用して LoadBalancer サービス用の OpenShift API を監視し、事前設定されたプールから IP アドレスを割り当てるコントローラー ポッドとしてインストールされます。

このページには、F5 BIG-IP CIS および IPAM コントローラのインストールおよび構成の手順が記載されています。前提条件として、F5 BIG-IP システムを導入し、ライセンスを取得する必要があります。また、BIG-IP

VE 基本ライセンスにデフォルトで含まれている SDN サービスのライセンスも取得する必要があります。

> F5 BIG-IP は、スタンドアロン モードまたはクラスター モードで導入できます。この検証のために、F5 BIG-IP はスタンドアロン モードで導入されましたが、実稼働環境では、単一障害点を回避するために BIG-IP のクラスターを使用することをお勧めします。

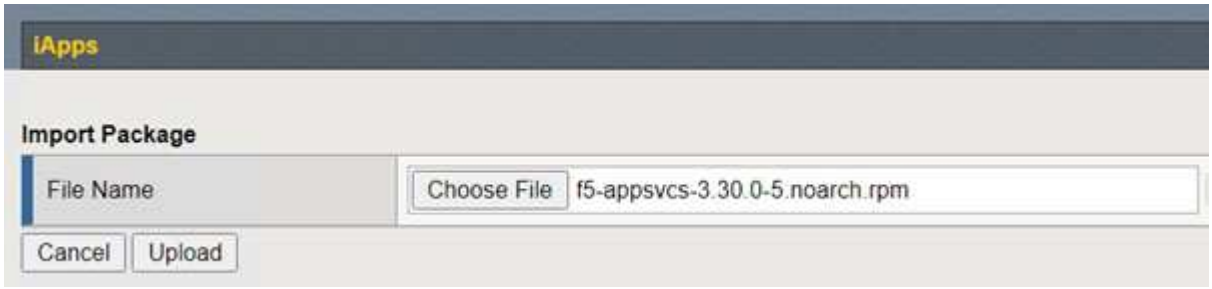
> F5 BIG-IP システムは、専用ハードウェア、クラウド、またはバージョン 12.x 以降のオンプレミスの仮想アプライアンスとして導入して、F5 CIS と統合できます。このドキュメントでは、F5 BIG-IP システムが、BIG-IP VE エディションなどを使用して仮想アプライアンスとして検証されました。

検証済みリリース

テクノロジ	ソフトウェア バージョン
レッドハット オープンシフト	4.6 EUS、4.7
F5 BIG-IP VEエディション	16.1.0
F5 コンテナ イングレス サービス	2.5.1
F5 IPAM コントローラー	0.1.4
F5 AS3	3.30.0

インストール

1. F5 Application Services 3 拡張機能をインストールすると、BIG-IP システムが命令型コマンドではなく JSON で構成を受け入れることができるようになります。へ移動 "F5 AS3 GitHubリポジトリ"最新の RPM ファイルをダウンロードします。
2. F5 BIG-IP システムにログインし、iApps > Package Management LX に移動して、インポートをクリックします。
3. [ファイルの選択] をクリックし、ダウンロードした AS3 RPM ファイルを選択し、[OK] をクリックして、[アップロード] をクリックします。

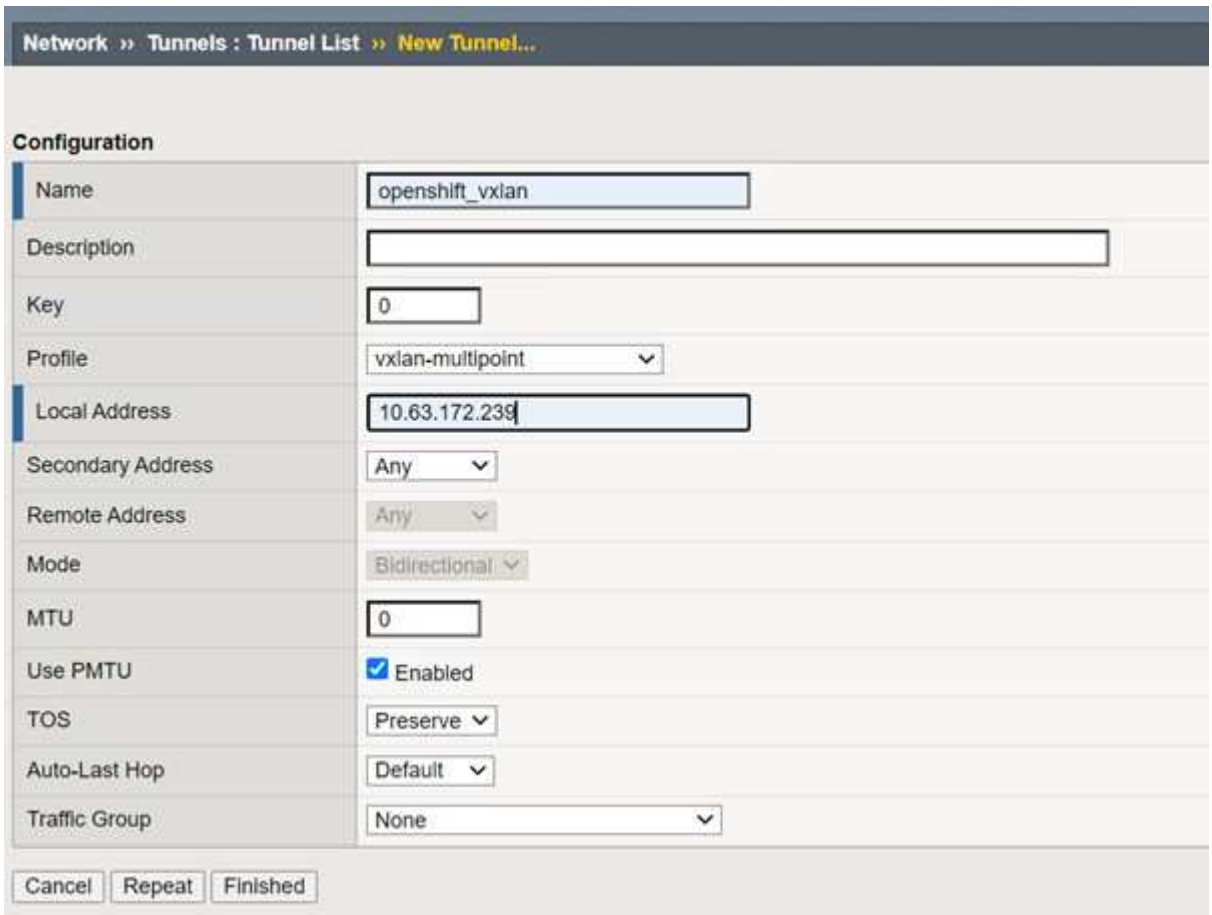


4. AS3 拡張機能が正常にインストールされていることを確認します。

5. 次に、OpenShift と BIG-IP システム間の通信に必要なリソースを構成します。まず、OpenShift SDN 用の

BIG-IP システム上に VXLAN トンネル インターフェイスを作成して、OpenShift と BIG-IP サーバーの間にトンネルを作成します。 [ネットワーク] > [トンネル] > [プロファイル] に移動し、[作成] をクリックして、親プロファイルを vxlan に、フラッディング タイプをマルチキャストに設定します。プロファイルの名前を入力し、「完了」をクリックします。

6. [ネットワーク] > [トンネル] > [トンネル リスト] に移動し、[作成] をクリックして、トンネルの名前とローカル IP アドレスを入力します。前の手順で作成したトンネル プロファイルを選択し、[完了] をクリックします。



7. cluster-admin 権限で Red Hat OpenShift クラスターにログインします。
8. F5 BIG-IP サーバー用のホストサブネットを OpenShift 上に作成します。これにより、サブネットが OpenShift クラスターから F5 BIG-IP サーバーに拡張されます。ホストサブネットの YAML 定義をダウンロードします。

```
wget https://github.com/F5Networks/k8s-bigip-
ctlr/blob/master/docs/config_examples/openshift/f5-kctlr-openshift-
hostsubnet.yaml
```

9. ホスト サブネット ファイルを編集し、OpenShift SDN の BIG-IP VTEP (VXLAN トンネル) IP を追加します。

```
apiVersion: v1
kind: HostSubnet
metadata:
  name: f5-server
  annotations:
    pod.network.openshift.io/fixed-vnid-host: "0"
    pod.network.openshift.io/assign-subnet: "true"
# provide a name for the node that will serve as BIG-IP's entry into the
cluster
host: f5-server
# The hostIP address will be the BIG-IP interface address routable to
the
# OpenShift Origin nodes.
# This address is the BIG-IP VTEP in the SDN's VXLAN.
hostIP: 10.63.172.239
```

環境に応じてホスト IP やその他の詳細を変更します。

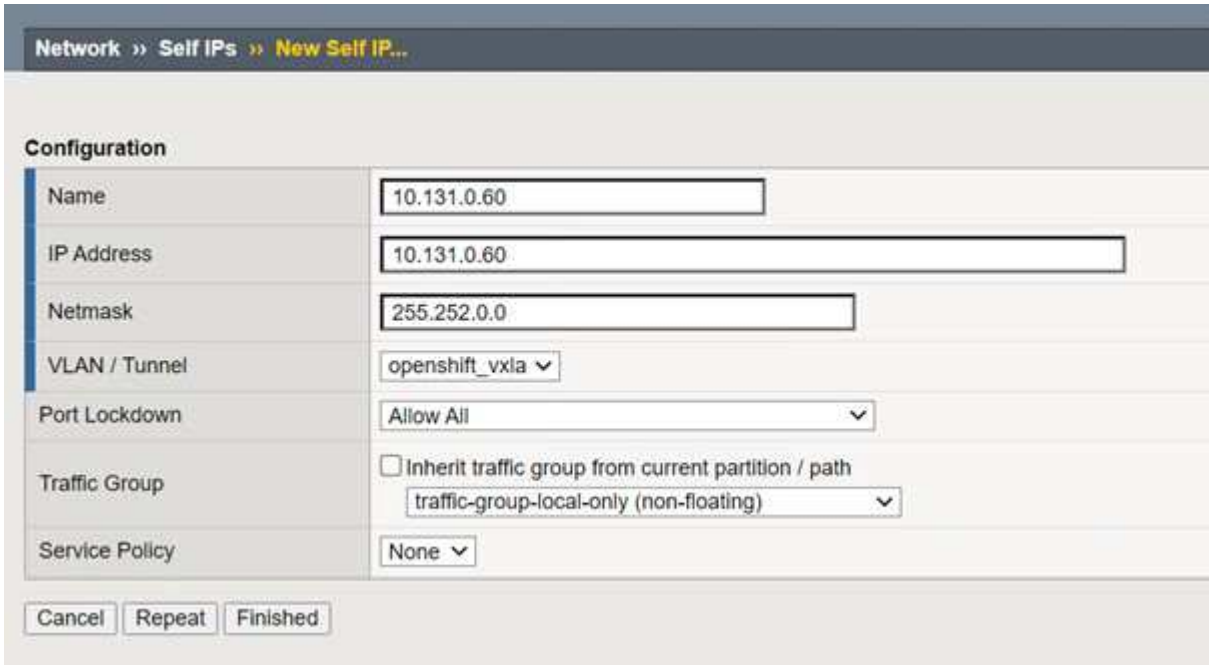
10. HostSubnet リソースを作成します。

```
[admin@rhel-7 ~]$ oc create -f f5-kctlr-openshift-hostsubnet.yaml

hostsubnet.network.openshift.io/f5-server created
```

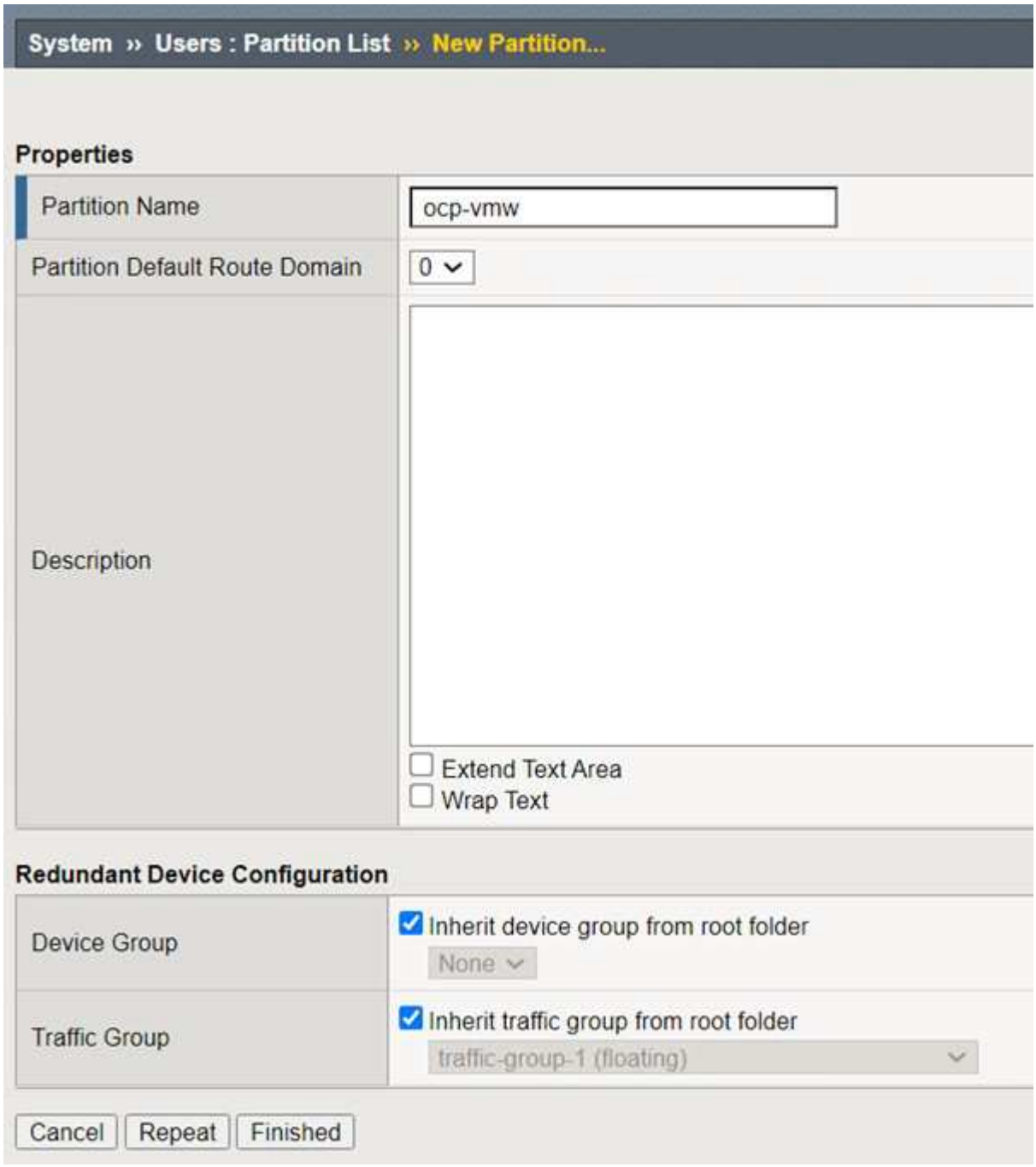
11. F5 BIG-IP サーバー用に作成されたホスト サブネットのクラスター IP サブネット範囲を取得します。

```
[admin@rhel-7 ~]$ oc get hostsubnet

NAME                           HOST                           HOST IP
SUBNET          EGRESS CIDRS   EGRESS IPS
f5-server                      f5-server                      10.63.172.239
10.131.0.0/23
ocp-vmw-nszws-master-0         ocp-vmw-nszws-master-0         10.63.172.44
10.128.0.0/23
ocp-vmw-nszws-master-1         ocp-vmw-nszws-master-1         10.63.172.47
10.130.0.0/23
ocp-vmw-nszws-master-2         ocp-vmw-nszws-master-2         10.63.172.48
10.129.0.0/23
ocp-vmw-nszws-worker-r8fh4     ocp-vmw-nszws-worker-r8fh4     10.63.172.7
10.130.2.0/23
ocp-vmw-nszws-worker-tvr46     ocp-vmw-nszws-worker-tvr46     10.63.172.11
10.129.2.0/23
ocp-vmw-nszws-worker-wdxhg     ocp-vmw-nszws-worker-wdxhg     10.63.172.24
10.128.2.0/23
ocp-vmw-nszws-worker-wg8r4     ocp-vmw-nszws-worker-wg8r4     10.63.172.15
10.131.2.0/23
ocp-vmw-nszws-worker-wtgfw     ocp-vmw-nszws-worker-wtgfw     10.63.172.17
10.128.4.0/23
```

12. F5 BIG-IP サーバーに対応する OpenShift のホスト サブネット範囲内の IP を使用して、OpenShift VXLAN 上にセルフ IP を作成します。 F5 BIG-IP システムにログインし、「ネットワーク」>「セルフ IP」に移動して「作成」をクリックします。 F5 BIG-IP ホスト サブネット用に作成されたクラスター IP サブネットから IP を入力し、VXLAN トンネルを選択して、その他の詳細を入力します。次に、「完了」をクリックします。



13. CIS で設定および使用されるパーティションを F5 BIG-IP システムに作成します。 [システム] > [ユーザー] > [パーティション リスト] に移動し、[作成] をクリックして詳細を入力します。次に、「完了」をクリックします。



F5 では、CIS によって管理されるパーティションでは手動構成を行わないことを推奨しています。

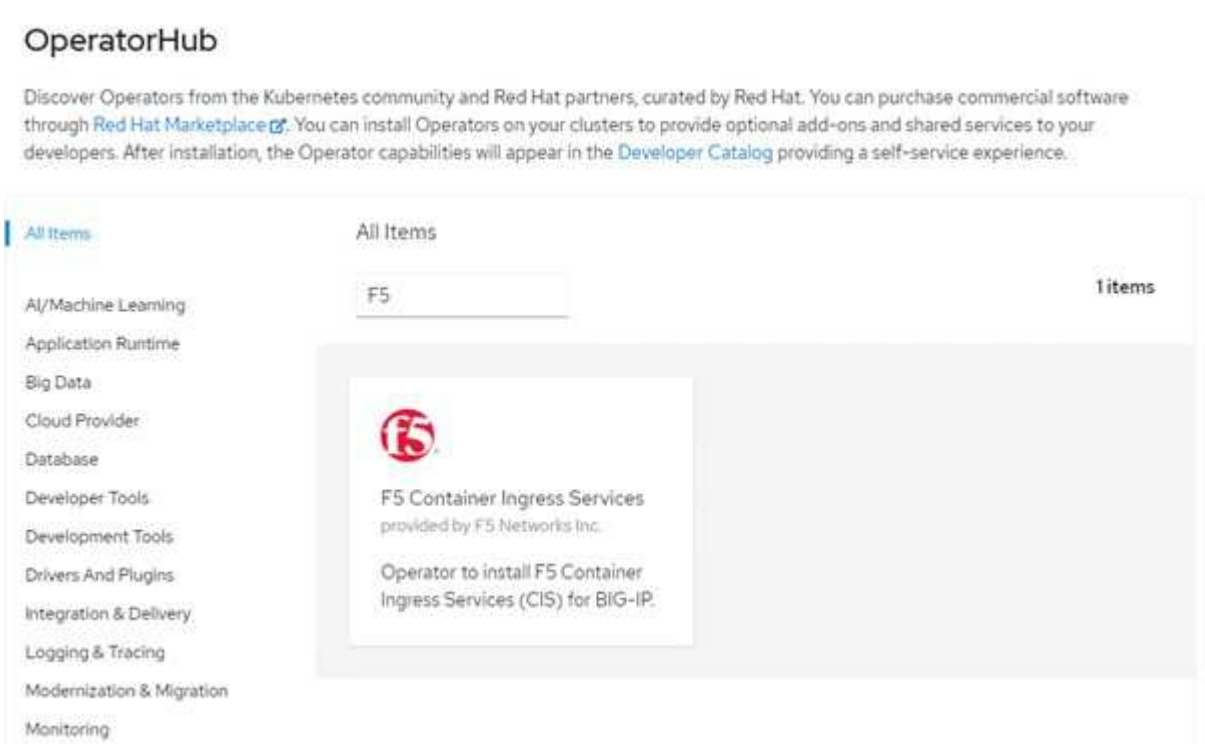
14. OperatorHub のオペレーターを使用して F5 BIG-IP CIS をインストールします。クラスター管理者権限で Red Hat OpenShift クラスターにログインし、オペレーターの前提条件である F5 BIG-IP システムのログイン資格情報を使用してシークレットを作成します。

```
[admin@rhel-7 ~]$ oc create secret generic bigip-login -n kube-system
--from-literal=username=admin --from-literal=password=admin

secret/bigip-login created
```

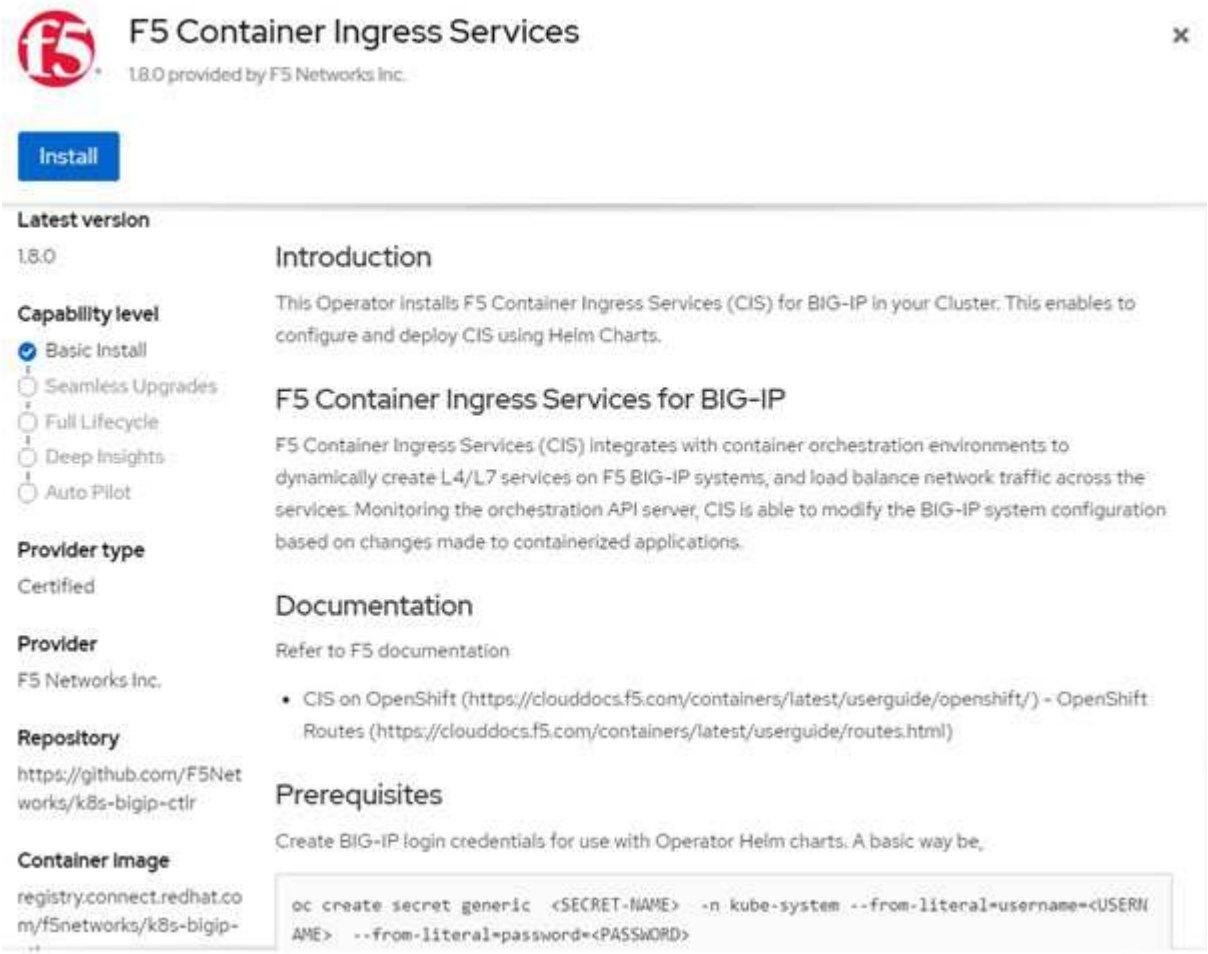
15. F5 CIS CRD をインストールします。

```
[admin@rhel-7 ~]$ oc apply -f
https://raw.githubusercontent.com/F5Networks/k8s-bigip-
ctlr/master/docs/config_examples/crd/Install/customresourcedefinitions.y
ml

customresourcedefinition.apiextensions.k8s.io/virtualservers.cis.f5.com
created
customresourcedefinition.apiextensions.k8s.io/tlsprofiles.cis.f5.com
created
customresourcedefinition.apiextensions.k8s.io/transportservers.cis.f5.co
m created
customresourcedefinition.apiextensions.k8s.io/externaldnss.cis.f5.com
created
customresourcedefinition.apiextensions.k8s.io/ingresslinks.cis.f5.com
created
```

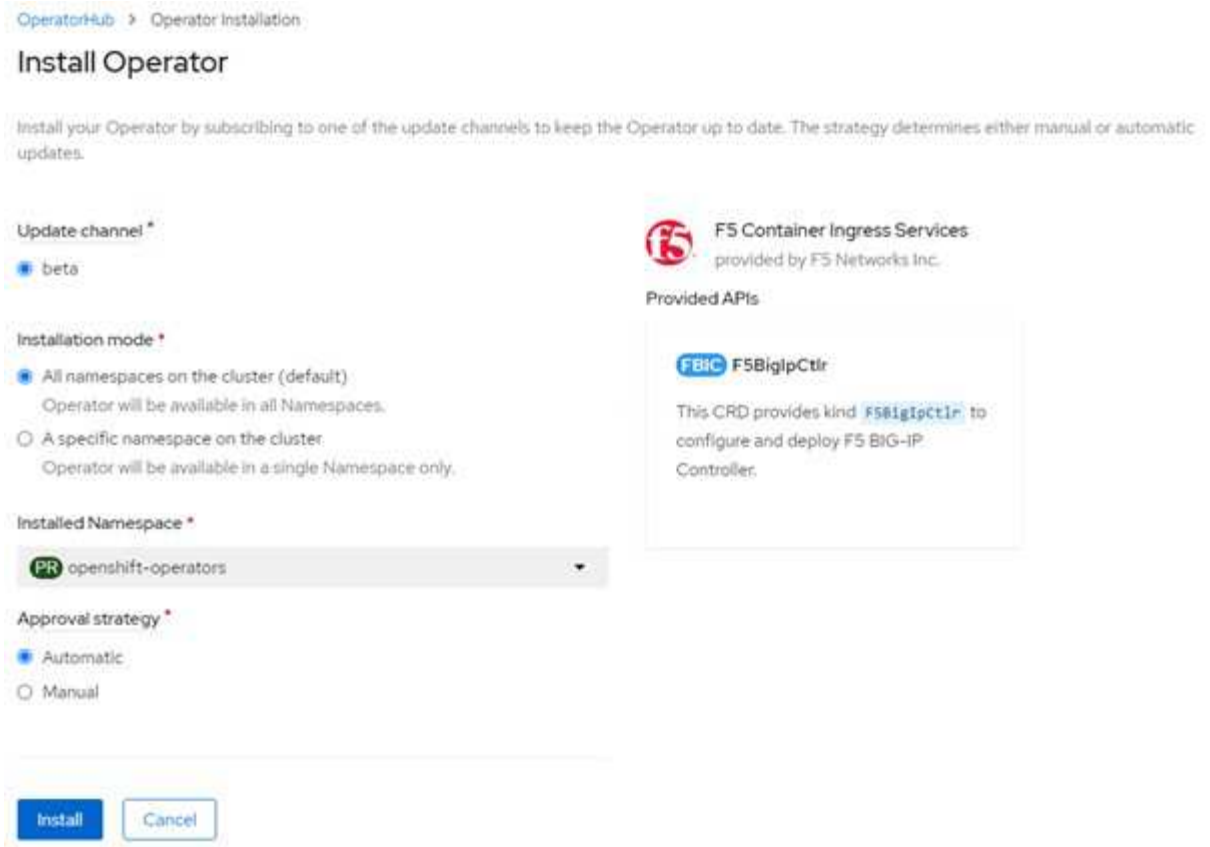
16. 「オペレーター」 > 「OperatorHub」に移動し、キーワード「F5」を検索して、「F5 コンテナー イングレス サービス」タイルをクリックします。



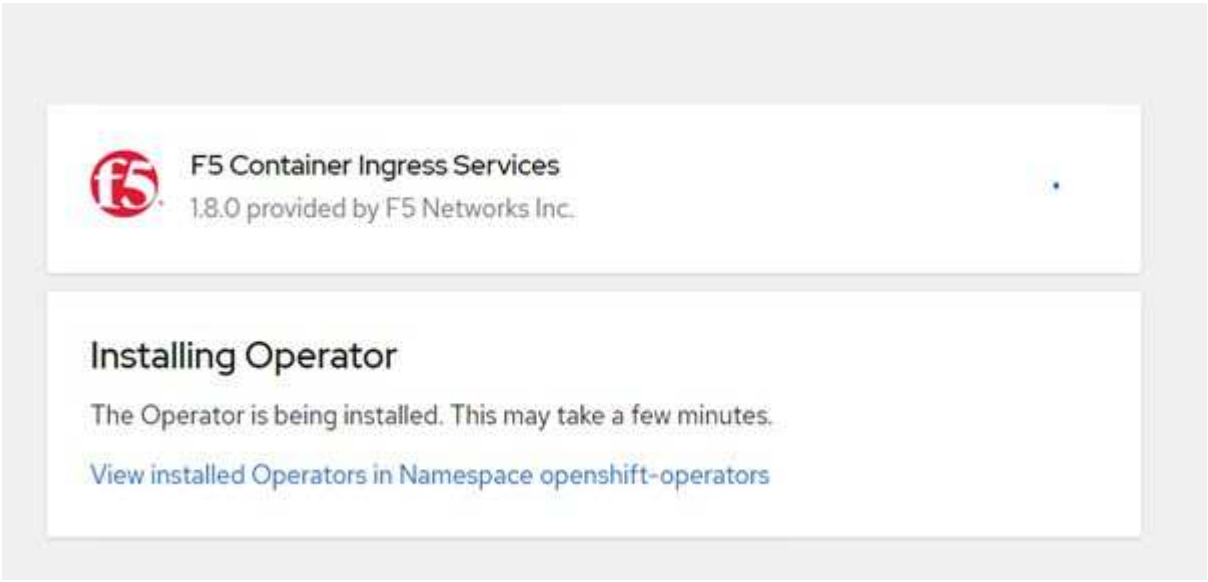
17. オペレーター情報を読み、「インストール」をクリックします。



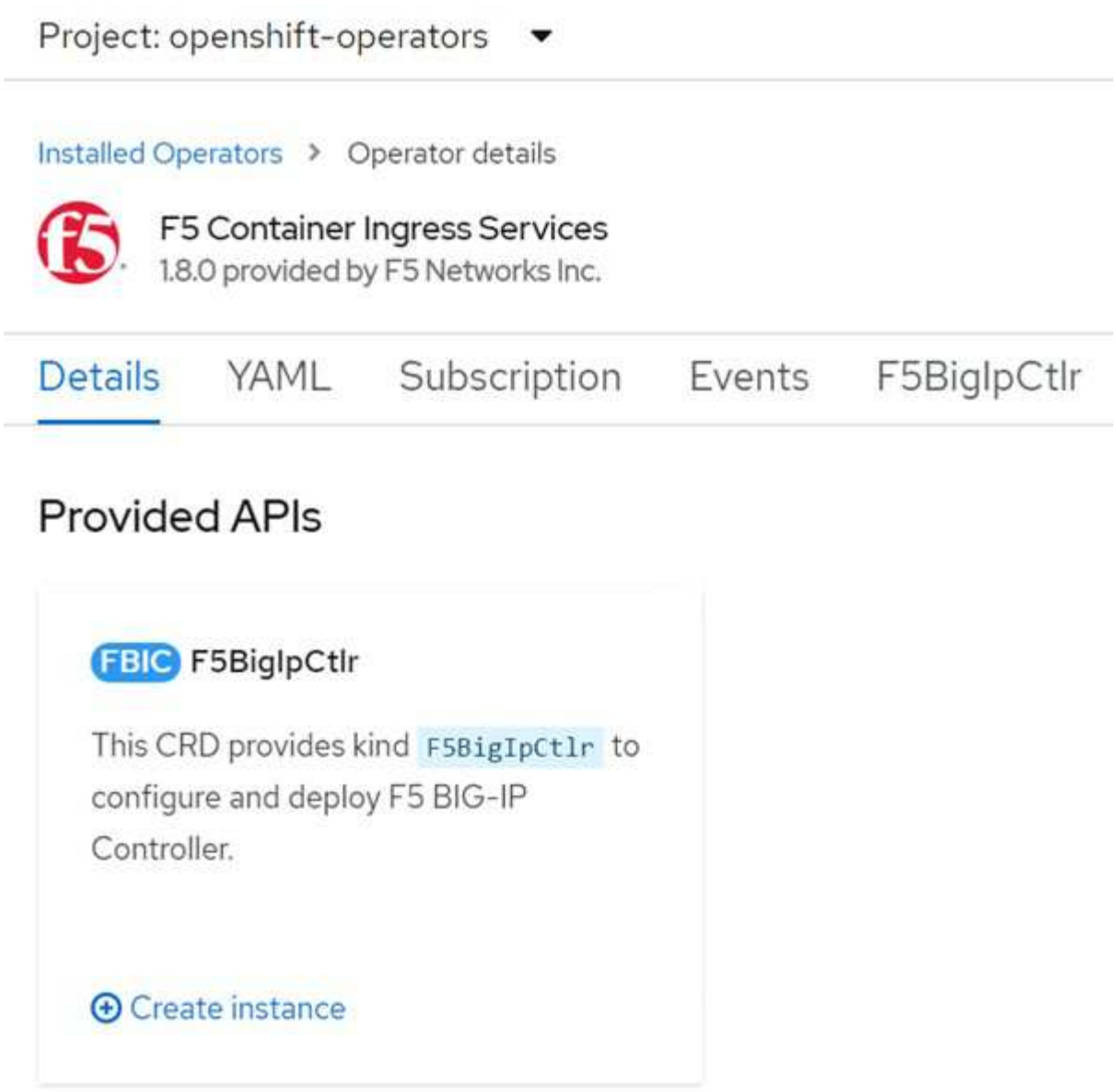
18. インストール オペレーター画面で、すべてのパラメーターをデフォルトのままにして、[インストール] をクリックします。



19. オペレータのインストールにはしばらく時間がかかります。



20. オペレータがインストールされると、「インストール成功」メッセージが表示されます。
21. [オペレーター] > [インストール済みオペレーター] に移動し、[F5 コンテナー イングレス サービス] をクリックして、F5BigIpCtlr タイルの下の [インスタンスの作成] をクリックします。



22. YAML ビューをクリックし、必要なパラメータを更新した後、次のコンテンツを貼り付けます。

パラメータを更新する `bigip_partition`, `openshift_sdn_name`, `bigip_url`そして `bigip_login_secret`コンテンツをコピーする前に、以下の設定値を反映させてください。

```
apiVersion: cis.f5.com/v1
kind: F5BigIpCtlr
metadata:
  name: f5-server
  namespace: openshift-operators
spec:
  args:
    log_as3_response: true
    agent: as3
    log_level: DEBUG
    bigip_partition: ocp-vmw
    openshift_sdn_name: /Common/openshift_vxlan
    bigip_url: 10.61.181.19
    insecure: true
    pool-member-type: cluster
    custom_resource_mode: true
    as3_validation: true
    ipam: true
    manage_configmaps: true
  bigip_login_secret: bigip-login
  image:
    pullPolicy: Always
    repo: f5networks/cntr-ingress-svcs
    user: registry.connect.redhat.com
  namespace: kube-system
  rbac:
    create: true
  resources: {}
  serviceAccount:
    create: true
  version: latest
```

23. このコンテンツを貼り付けた後、「作成」をクリックします。これにより、CIS ポッドが kube-system 名前空間にインストールされます。

Red Hat OpenShift は、デフォルトで、L7 ロード バランシング用のルートを介してサービスを公開する方法を提供します。組み込みの OpenShift ルーターは、これらのルートのトラフィックの通知と処理を担当します。ただし、補助ルーターとして、またはセルフホスト型 OpenShift ルーターの代わりとして実行できる外部 F5 BIG-IP システムを介してルートをサポートするように F5 CIS を構成することもできます。 CIS は、OpenShift ルートのルーターとして機能する仮想サーバーを BIG-IP システム内に作成し、BIG-IP がアドバタイズとトラフィック ルーティングを処理します。この機能を有効にするためのパラメータに関する情報については、ここにあるドキュメントを参照してください。これらのパラメーターは、apps/v1 API の OpenShift デプロイメント リソースに対して定義されていることに注意してください。したがって、これらを F5BigIpCtlr リソース cis.f5.com/v1 API で使用する場合は、パラメータ名のハイフン (-) をアンダースコア (_) に置き換えます。

24. CISリソースの作成に渡される引数には以下が含まれます。 `ipam: true`そして `custom_resource_mode: true`。これらのパラメータは、IPAM コントローラとの CIS 統合を有効にするために必要です。 F5 IPAM リソースを作成して、CIS が IPAM 統合を有効にしていることを確認します。

```
[admin@rhel-7 ~]$ oc get f5ipam -n kube-system

NAMESPACE   NAME                                AGE
kube-system   ipam.10.61.181.19.ocp-vmw      43s
```

25. F5 IPAM コントローラーに必要なサービス アカウント、ロール、およびロール バインディングを作成します。 YAML ファイルを作成し、次の内容を貼り付けます。

```
[admin@rhel-7 ~]$ vi f5-ipam-rbac.yaml

kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: ipam-ctlr-clusterrole
rules:
  - apiGroups: ["fic.f5.com"]
    resources: ["ipams","ipams/status"]
    verbs: ["get", "list", "watch", "update", "patch"]
---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: ipam-ctlr-clusterrole-binding
  namespace: kube-system
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: ipam-ctlr-clusterrole
subjects:
  - apiGroup: ""
    kind: ServiceAccount
    name: ipam-ctlr
    namespace: kube-system
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: ipam-ctlr
  namespace: kube-system
```

26. リソースを作成します。

```
[admin@rhel-7 ~]$ oc create -f f5-ipam-rbac.yaml

clusterrole.rbac.authorization.k8s.io/ipam-ctlr-clusterrole created
clusterrolebinding.rbac.authorization.k8s.io/ipam-ctlr-clusterrole-
binding created
serviceaccount/ipam-ctlr created
```

27. YAML ファイルを作成し、以下に示す F5 IPAM デプロイメント定義を貼り付けます。

以下のspec.template.spec.containers[0].argsのip-rangeパラメータを更新して、設定に対応するipamLabelsとIPアドレスの範囲を反映します。

ipamラベル[`range1`そして `range2`[以下の例] は、IPAM コントローラーが定義された範囲から IP アドレスを検出して割り当てるために、LoadBalancer タイプのサービスに対してアノテーションを付ける必要があることを示しています。

```
[admin@rhel-7 ~]$ vi f5-ipam-deployment.yaml

apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    name: f5-ipam-controller
  name: f5-ipam-controller
  namespace: kube-system
spec:
  replicas: 1
  selector:
    matchLabels:
      app: f5-ipam-controller
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: f5-ipam-controller
    spec:
      containers:
      - args:
        - --orchestration=openshift
        - --ip-range='{"range1":"10.63.172.242-10.63.172.249",
"range2":"10.63.170.111-10.63.170.129"}'
        - --log-level=DEBUG
        command:
        - /app/bin/f5-ipam-controller
        image: registry.connect.redhat.com/f5networks/f5-ipam-
controller:latest
        imagePullPolicy: IfNotPresent
        name: f5-ipam-controller
      dnsPolicy: ClusterFirst
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
      serviceAccount: ipam-ctlr
      serviceAccountName: ipam-ctlr
```

28. F5 IPAM コントローラーのデプロイメントを作成します。

```
[admin@rhel-7 ~]$ oc create -f f5-ipam-deployment.yaml

deployment/f5-ipam-controller created
```

29. F5 IPAM コントローラー ポッドが実行中であることを確認します。

```
[admin@rhel-7 ~]$ oc get pods -n kube-system

NAME                                      READY   STATUS    RESTARTS
AGE
f5-ipam-controller-5986cff5bd-2bvn6       1/1     Running   0
30s
f5-server-f5-bigip-ctlr-5d7578667d-qxdgj  1/1     Running   0
14m
```

30. F5 IPAM スキーマを作成します。

```
[admin@rhel-7 ~]$ oc create -f
https://raw.githubusercontent.com/F5Networks/f5-ipam-
controller/main/docs/_static/schemas/ipam_schema.yaml

customresourcedefinition.apiextensions.k8s.io/ipams.fic.f5.com
```

検証

1. LoadBalancerタイプのサービスを作成する

```
[admin@rhel-7 ~]$ vi example_svc.yaml

apiVersion: v1
kind: Service
metadata:
  annotations:
    cis.f5.com/ipamLabel: range1
  labels:
    app: f5-demo-test
  name: f5-demo-test
  namespace: default
spec:
  ports:
  - name: f5-demo-test
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: f5-demo-test
  sessionAffinity: None
  type: LoadBalancer
```

```
[admin@rhel-7 ~]$ oc create -f example_svc.yaml

service/f5-demo-test created
```

2. IPAM コントローラーが外部 IP を割り当てているかどうかを確認します。

```
[admin@rhel-7 ~]$ oc get svc

NAME           TYPE           CLUSTER-IP       EXTERNAL-IP
PORT(S)        AGE
f5-demo-test   LoadBalancer   172.30.210.108   10.63.172.242
80:32605/TCP   27s
```

3. デプロイメントを作成し、作成された LoadBalancer サービスを使用します。

```
[admin@rhel-7 ~]$ vi example_deployment.yaml

apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: f5-demo-test
  name: f5-demo-test
spec:
  replicas: 2
  selector:
    matchLabels:
      app: f5-demo-test
  template:
    metadata:
      labels:
        app: f5-demo-test
    spec:
      containers:
      - env:
        - name: service_name
          value: f5-demo-test
        image: nginx
        imagePullPolicy: Always
        name: f5-demo-test
        ports:
        - containerPort: 80
          protocol: TCP
```

```
[admin@rhel-7 ~]$ oc create -f example_deployment.yaml

deployment/f5-demo-test created
```

4. ポッドが実行中かどうかを確認します。

```
[admin@rhel-7 ~]$ oc get pods

NAME                           READY   STATUS    RESTARTS   AGE
f5-demo-test-57c46f6f98-47wwp   1/1     Running   0          27s
f5-demo-test-57c46f6f98-cl2m8   1/1     Running   0          27s
```

5. OpenShift の LoadBalancer タイプのサービスに対して、BIG-IP システム内に対応する仮想サーバーが作成されているかどうかを確認します。 [ローカル トラフィック] > [仮想サーバー] > [仮想サーバー リスト]

に移動します。

プライベートイメージレジストリの作成

Red Hat OpenShiftのほとんどのデプロイメントでは、次のようなパブリックレジストリを使用します。 "Quay.io"または "Dockerハブ"ほとんどの顧客のニーズを満たします。ただし、顧客が独自のプライベート画像やカスタマイズされた画像をホストしたい場合もあります。

この手順では、 TridentおよびNetApp ONTAPによって提供される永続ボリュームによってサポートされるプライベート イメージ レジストリを作成する方法について説明します。

> Trident Protect では、 Astraコンテナーに必要なイメージをホストするためのレジストリが必要です。次のセクションでは、Red Hat OpenShift クラスターにプライベート レジストリを設定し、 Trident Protect のインストールをサポートするために必要なイメージをプッシュする手順について説明します。

プライベートイメージレジストリの作成

1. 現在のデフォルトのストレージ クラスからデフォルトのアノテーションを削除し、Trident ベースのストレージ クラスを OpenShift クラスターのデフォルトとしてアノテーションします。

```
[netapp-user@rhel7 ~]$ oc patch storageclass thin -p '{"metadata":
{"annotations": {"storageclass.kubernetes.io/is-default-class":
"false"}}}'
storageclass.storage.k8s.io/thin patched

[netapp-user@rhel7 ~]$ oc patch storageclass ocp-trident -p
'{"metadata": {"annotations": {"storageclass.kubernetes.io/is-default-
class": "true"}}}'
storageclass.storage.k8s.io/ocp-trident patched
```

2. 以下のストレージパラメータを入力して、imageregistryオペレータを編集します。 `spec`セクション。

```
[netapp-user@rhel7 ~]$ oc edit
configs.imageregistry.operator.openshift.io

storage:
  pvc:
    claim:
```

3. 以下のパラメータを入力します。 `spec`カスタム ホスト名を使用して OpenShift ルートを作成するセクション。保存して終了します。

```
routes:
  - hostname: astra-registry.apps.ocp-vmw.cie.netapp.com
    name: netapp-astra-route
```

上記のルート設定は、ルートにカスタム ホスト名が必要な場合に使用されます。 OpenShiftでデフォルトのホスト名を持つルートを作成したい場合は、次のパラメータを `spec`セクション： `defaultRoute: true`。

カスタムTLS証明書

ルートにカスタム ホスト名を使用している場合、デフォルトでは、OpenShift Ingress オペレーターのデフォルトの TLS 構成が使用されます。ただし、ルートにカスタム TLS 構成を追加することはできます。そのためには、次の手順を実行します。

a. ルートの TLS 証明書とキーを使用してシークレットを作成します。

```
[netapp-user@rhel7 ~]$ oc create secret tls astra-route-tls -n
openshift-image-registry –cert/home/admin/netapp-astra/tls.crt
--key=/home/admin/netapp-astra/tls.key
```

b. imageregistryオペレータを編集し、次のパラメータを追加します。 `spec`セクション。

```
[netapp-user@rhel7 ~]$ oc edit
configs.imageregistry.operator.openshift.io

routes:
  - hostname: astra-registry.apps.ocp-vmw.cie.netapp.com
    name: netapp-astra-route
    secretName: astra-route-tls
```

4. イメージレジストリオペレータを再度編集し、オペレータの管理状態を次のように変更します。

`Managed`州。保存して終了します。

```
oc edit configs.imageregistry/cluster

managementState: Managed
```

5. すべての前提条件が満たされている場合は、プライベート イメージ レジストリの PVC、ポッド、およびサービスが作成されます。数分以内にレジストリが起動するはずです。

```
[netapp-user@rhel7 ~]$oc get all -n openshift-image-registry

NAME                                                READY   STATUS
RESTARTS   AGE
pod/cluster-image-registry-operator-74f6d954b6-rb7zr   1/1     Running
3          90d
pod/image-pruner-1627257600-f5cpj                   0/1     Completed
0          2d9h
pod/image-pruner-1627344000-swqx9                   0/1     Completed
0          33h
pod/image-pruner-1627430400-rv5nt                   0/1     Completed
0          9h
pod/image-registry-6758b547f-6pnj8                  1/1     Running
0          76m
pod/node-ca-bwb5r                                   1/1     Running
0          90d
pod/node-ca-f8w54                                   1/1     Running
0          90d
pod/node-ca-gjx7h                                   1/1     Running
0          90d
pod/node-ca-lcx4k                                   1/1     Running
0          33d
pod/node-ca-v7zmx                                   1/1     Running
0          7d21h
pod/node-ca-xpppp                                   1/1     Running
0          89d

NAME                              TYPE        CLUSTER-IP       EXTERNAL-
IP   PORT(S)     AGE
service/image-registry            ClusterIP   172.30.196.167   <none>
5000/TCP    15h
service/image-registry-operator   ClusterIP   None             <none>
60000/TCP   90d

NAME                      DESIRED   CURRENT   READY   UP-TO-DATE
AVAILABLE   NODE SELECTOR            AGE
```

```
daemonset.apps/node-ca   6         6         6       6            6
kubernetes.io/os=linux   90d

NAME                                                READY   UP-TO-DATE
AVAILABLE   AGE
deployment.apps/cluster-image-registry-operator   1/1     1            1
90d
deployment.apps/image-registry                    1/1     1            1
15h

NAME                                                                DESIRED
CURRENT   READY   AGE
replicaset.apps/cluster-image-registry-operator-74f6d954b6   1         1
1       90d
replicaset.apps/image-registry-6758b547f                    1         1
1       76m
replicaset.apps/image-registry-78bfbd7f59                   0         0
0       15h
replicaset.apps/image-registry-7fcc8d6cc8                   0         0
0       80m
replicaset.apps/image-registry-864f88f5b                    0         0
0       15h
replicaset.apps/image-registry-cb47fffb                     0         0
0       10h

NAME                                  COMPLETIONS   DURATION   AGE
job.batch/image-pruner-1627257600   1/1           10s        2d9h
job.batch/image-pruner-1627344000   1/1           6s         33h
job.batch/image-pruner-1627430400   1/1           5s         9h

NAME                           SCHEDULE    SUSPEND   ACTIVE   LAST
SCHEDULE   AGE
cronjob.batch/image-pruner   0 0 * * *   False     0        9h
90d

NAME                                        HOST/PORT
PATH   SERVICES         PORT    TERMINATION   WILDCARD
route.route.openshift.io/public-routes   astra-registry.apps.ocp-
vmw.cie.netapp.com          image-registry   <all>   reencrypt     None
```

6. Ingress オペレーターの OpenShift レジストリ ルートにデフォルトの TLS 証明書を使用している場合は、次のコマンドを使用して TLS 証明書を取得できます。

```
[netapp-user@rhel7 ~]$ oc extract secret/router-ca --keys=tls.crt -n
openshift-ingress-operator
```

7. OpenShift ノードがレジストリにアクセスしてイメージをプルできるようにするには、OpenShift ノード上の Docker クライアントに証明書を追加します。 configmapを作成します `openshift-config`TLS 証明書を使用して名前空間を作成し、それをクラスター イメージ構成にパッチして、証明書を信頼できるものにします。

```
[netapp-user@rhel7 ~]$ oc create configmap astra-ca -n openshift-config
--from-file=astra-registry.apps.ocp-vmw.cie.netapp.com=tls.crt

[netapp-user@rhel7 ~]$ oc patch image.config.openshift.io/cluster
--patch '{"spec":{"additionalTrustedCA":{"name":"astra-ca"}}}'
--type=merge
```

8. OpenShift 内部レジストリは認証によって制御されます。すべての OpenShift ユーザーは OpenShift レジストリにアクセスできますが、ログインしたユーザーが実行できる操作はユーザー権限によって異なります。
   a. ユーザーまたはユーザー グループがレジストリからイメージをプルできるようにするには、ユーザーに registry-viewer ロールが割り当てられている必要があります。

```
[netapp-user@rhel7 ~]$ oc policy add-role-to-user registry-viewer
ocp-user

[netapp-user@rhel7 ~]$ oc policy add-role-to-group registry-viewer
ocp-user-group
```

   b. ユーザーまたはユーザー グループがイメージを書き込んだりプッシュしたりできるようにするには、ユーザーにレジストリ エディターのロールが割り当てられている必要があります。

```
[netapp-user@rhel7 ~]$ oc policy add-role-to-user registry-editor
ocp-user

[netapp-user@rhel7 ~]$ oc policy add-role-to-group registry-editor
ocp-user-group
```

9. OpenShift ノードがレジストリにアクセスしてイメージをプッシュまたはプルするには、プル シークレットを構成する必要があります。

```
[netapp-user@rhel7 ~]$ oc create secret docker-registry astra-registry-
credentials --docker-server=astra-registry.apps.ocp-vmw.cie.netapp.com
--docker-username=ocp-user --docker-password=password
```

10. このプル シークレットは、サービス アカウントにパッチを適用したり、対応するポッド定義で参照したりできます。
    a. サービス アカウントにパッチを適用するには、次のコマンドを実行します。

```
[netapp-user@rhel7 ~]$ oc secrets link <service_account_name> astra-
registry-credentials --for=pull
```

b. ポッド定義でプルシークレットを参照するには、次のパラメータを `spec`セクション。

```
imagePullSecrets:
  - name: astra-registry-credentials
```

11. OpenShift ノード以外のワークステーションからイメージをプッシュまたはプルするには、次の手順を実行します。

a. TLS 証明書を docker クライアントに追加します。

```
[netapp-user@rhel7 ~]$ sudo mkdir /etc/docker/certs.d/astra-
registry.apps.ocp-vmw.cie.netapp.com

[netapp-user@rhel7 ~]$ sudo cp /path/to/tls.crt
/etc/docker/certs.d/astra-registry.apps.ocp-vmw.cie.netapp.com
```

b. oc login コマンドを使用して OpenShift にログインします。

```
[netapp-user@rhel7 ~]$ oc login --token=sha256~D49SpB_lesSrJYwrM0LIO
-VRcjWHu0a27vKa0 --server=https://api.ocp-vmw.cie.netapp.com:6443
```

c. podman/docker コマンドで OpenShift ユーザー資格情報を使用してレジストリにログインします。

ポッドマン

```
[netapp-user@rhel7 ~]$ podman login astra-registry.apps.ocp-
vmw.cie.netapp.com -u kubeadmin -p $(oc whoami -t) --tls
-verify=false
```

+ 注意: 使用している場合 `kubeadmin`ユーザーがプライベート レジストリにログインするには、パスワードの代わりにトークンを使用します。

ドッカー

```
[netapp-user@rhel7 ~]$ docker login astra-registry.apps.ocp-
vmw.cie.netapp.com -u kubeadmin -p $(oc whoami -t)
```

+ 注意: 使用している場合 `kubeadmin`ユーザーがプライベート レジストリにログインするには、パスワードの代わりにトークンを使用します。

d. 画像をプッシュまたはプルします。

ポッドマン

```
[netapp-user@rhel7 ~]$ podman push astra-registry.apps.ocp-
vmw.cie.netapp.com/netapp-astra/vault-controller:latest
[netapp-user@rhel7 ~]$ podman pull astra-registry.apps.ocp-
vmw.cie.netapp.com/netapp-astra/vault-controller:latest
```

ドッカー

```
[netapp-user@rhel7 ~]$ docker push astra-registry.apps.ocp-
vmw.cie.netapp.com/netapp-astra/vault-controller:latest
[netapp-user@rhel7 ~]$ docker pull astra-registry.apps.ocp-
vmw.cie.netapp.com/netapp-astra/vault-controller:latest
```

ソリューションの検証とユースケース

ソリューションの検証とユースケース: **Red Hat OpenShift**と**NetApp**

このページで提供されている例は、 NetAppを使用した Red Hat OpenShift のソリューション検証とユースケースです。

- "永続ストレージを使用した Jenkins CI/CD パイプラインのデプロイ"

- "NetAppを使用した Red Hat OpenShift でのマルチテナントの設定"
- "NetApp ONTAPを使用した Red Hat OpenShift 仮想化"
- "NetAppを使用した Red Hat OpenShift 上の Kubernetes 向け高度なクラスタ管理"

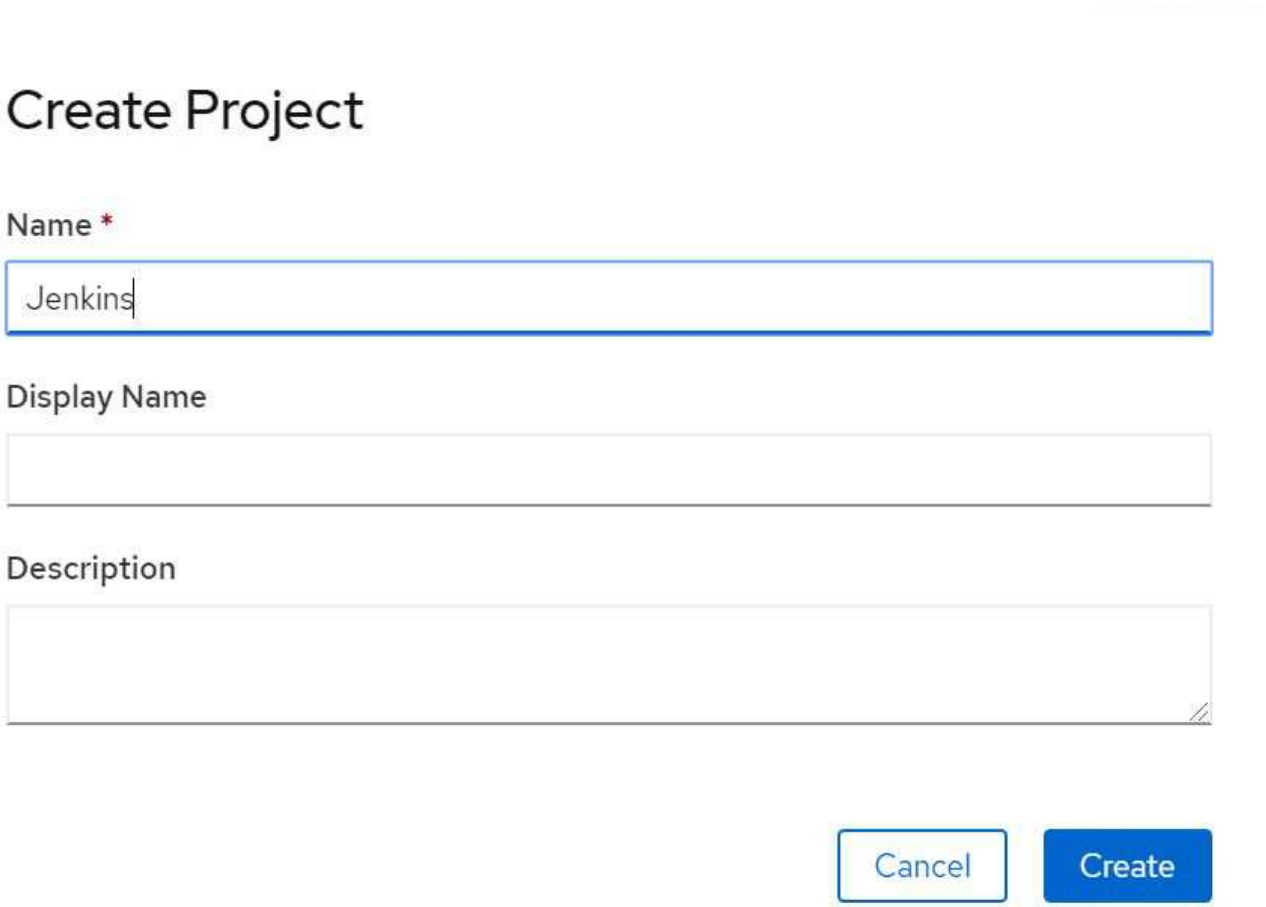
永続ストレージを使用した **Jenkins CI/CD** パイプラインの導入**: Red Hat OpenShift** と**NetApp**

このセクションでは、ソリューションの操作を検証するために、Jenkins を使用して継続的インテグレーション/継続的デリバリーまたはデプロイメント (CI/CD) パイプラインをデプロイする手順について説明します。

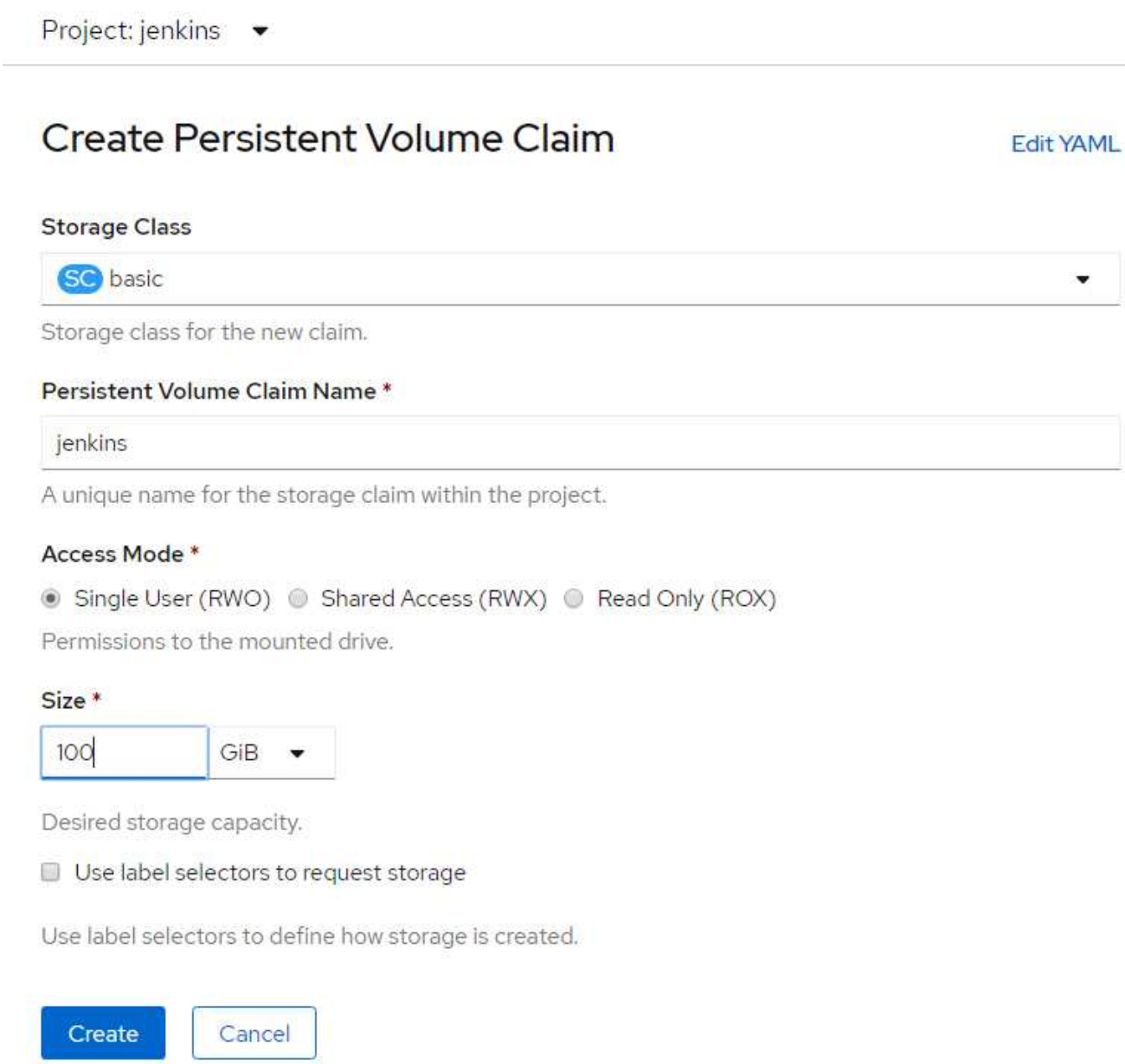
Jenkinsのデプロイメントに必要なリソースを作成する

Jenkins アプリケーションのデプロイに必要なリソースを作成するには、次の手順を実行します。

1. Jenkins という名前の新しいプロジェクトを作成します。



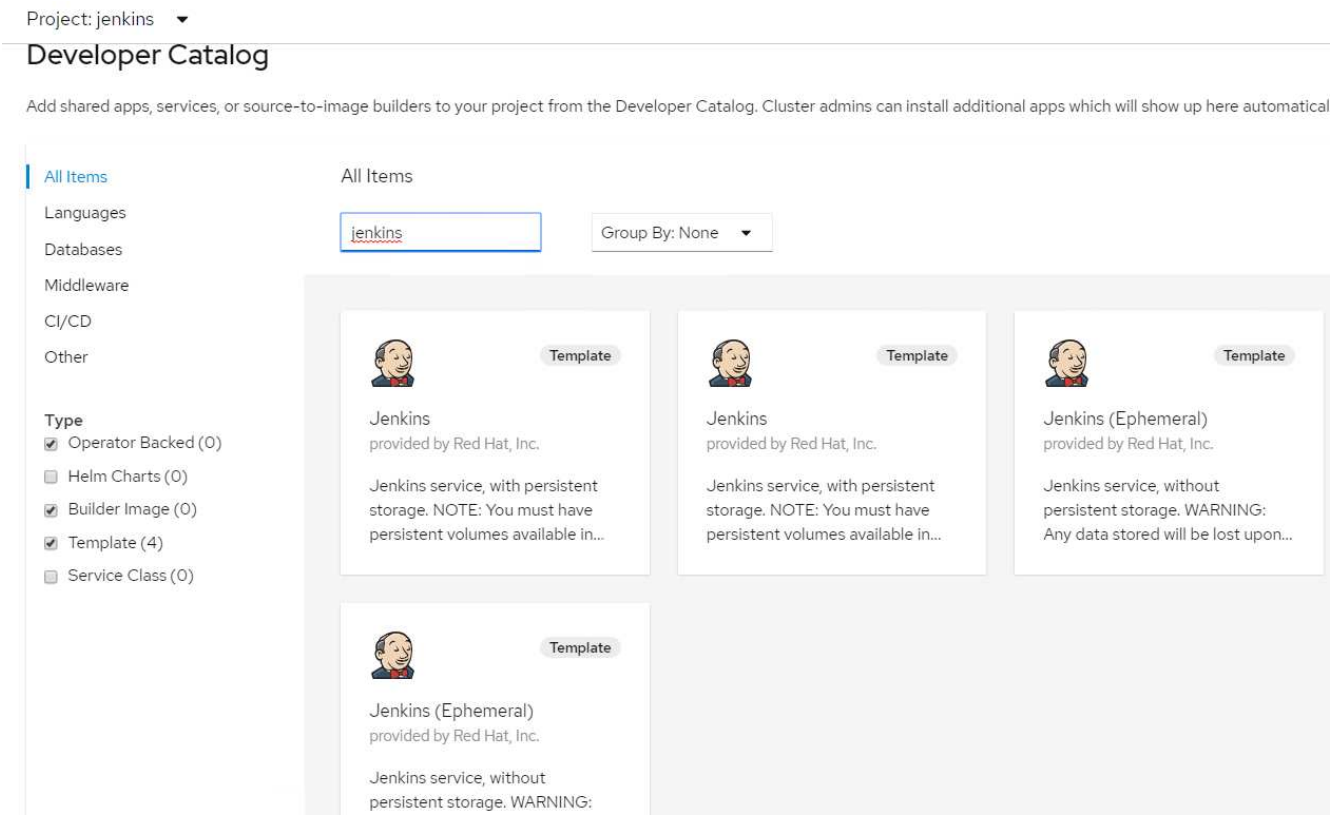
2. この例では、永続ストレージを備えた Jenkins をデプロイしました。 Jenkins ビルドをサポートするには、PVC を作成します。 [ストレージ] > [永続ボリューム要求] に移動し、[永続ボリューム要求の作成] をクリックします。作成されたストレージ クラスを選択し、永続ボリューム クレーム名が jenkins であることを確認して、適切なサイズとアクセス モードを選択し、[作成] をクリックします。



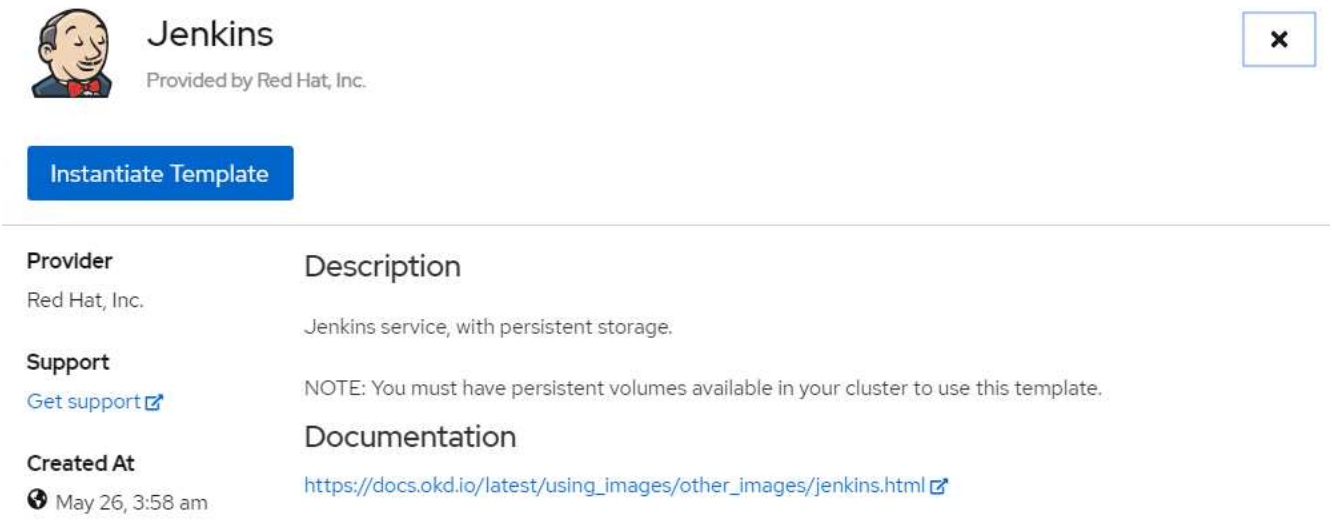
永続ストレージを使用した**Jenkins**のデプロイ

永続ストレージを使用して Jenkins をデプロイするには、次の手順を実行します。

1. 左上隅で、役割を管理者から開発者に変更します。 +追加をクリックし、カタログからを選択します。キーワードでフィルター バーで、jenkins を検索します。永続ストレージを備えた Jenkins サービスを選択します。

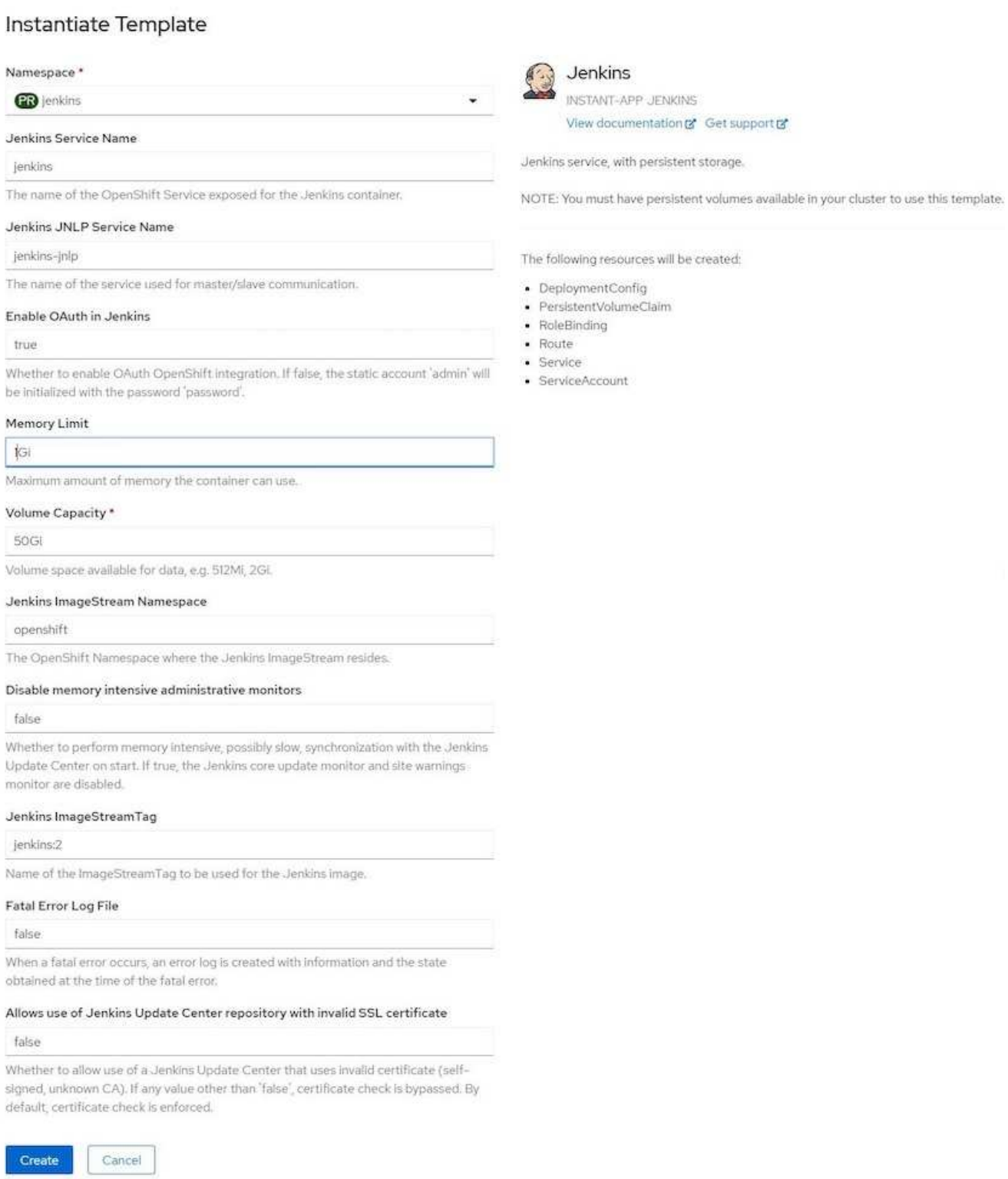


2. クリック `Instantiate Template`。

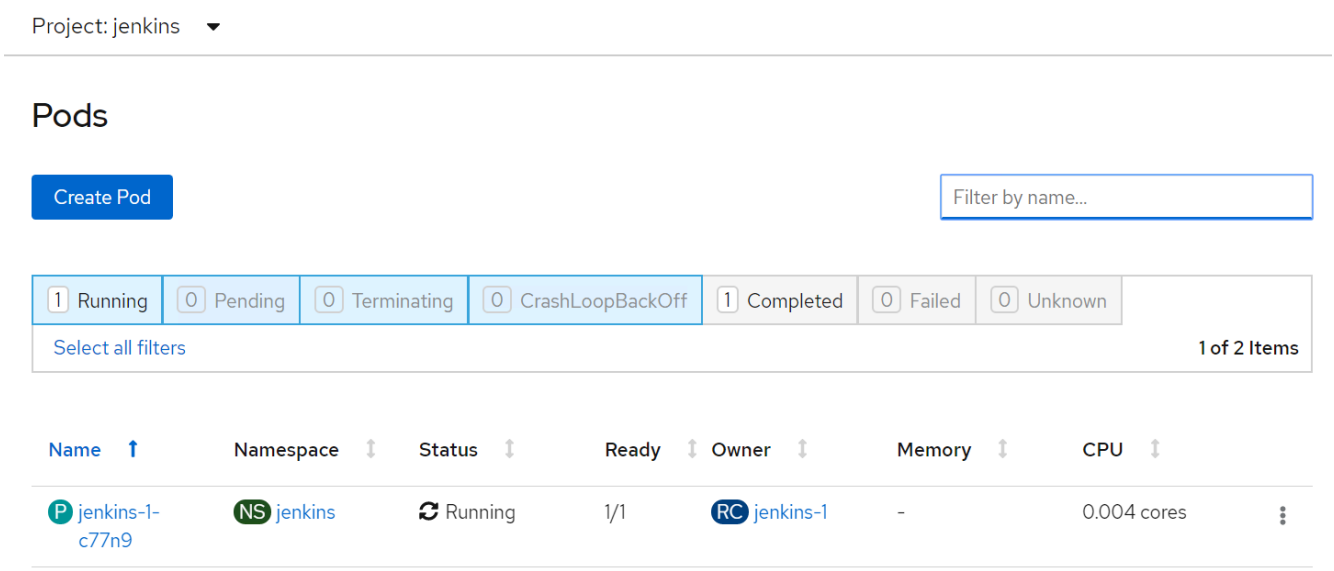


3. デフォルトでは、Jenkins アプリケーションの詳細が入力されます。要件に応じてパラメータを変更し、「作成」をクリックします。このプロセスでは、OpenShift 上で Jenkins をサポートするために必要

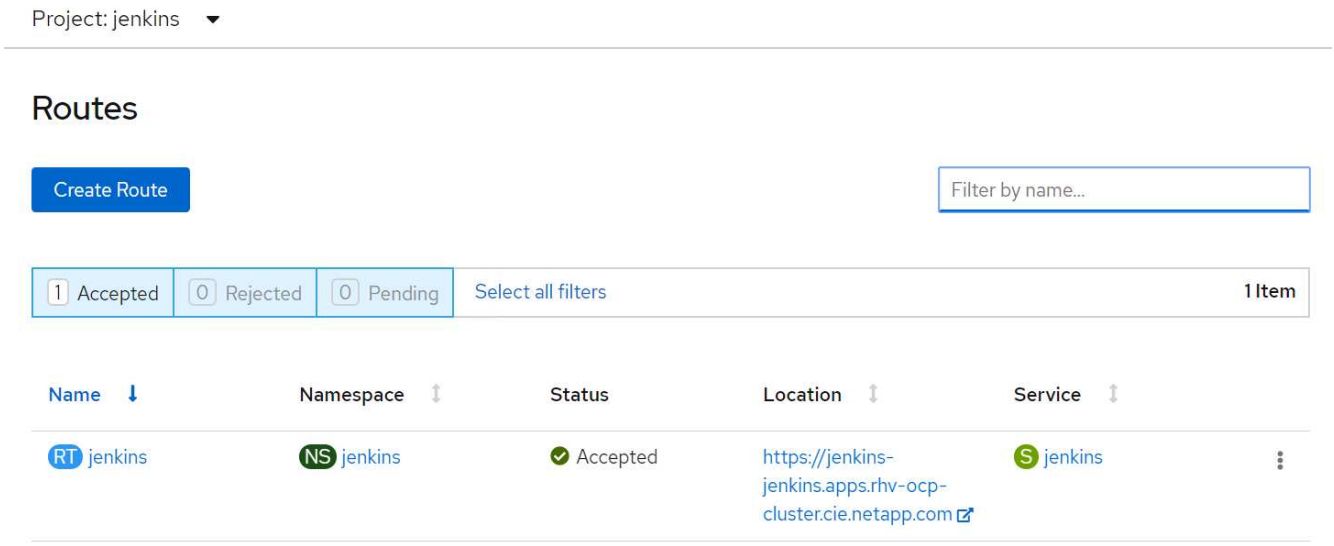
なすべてのリソースが作成されます。



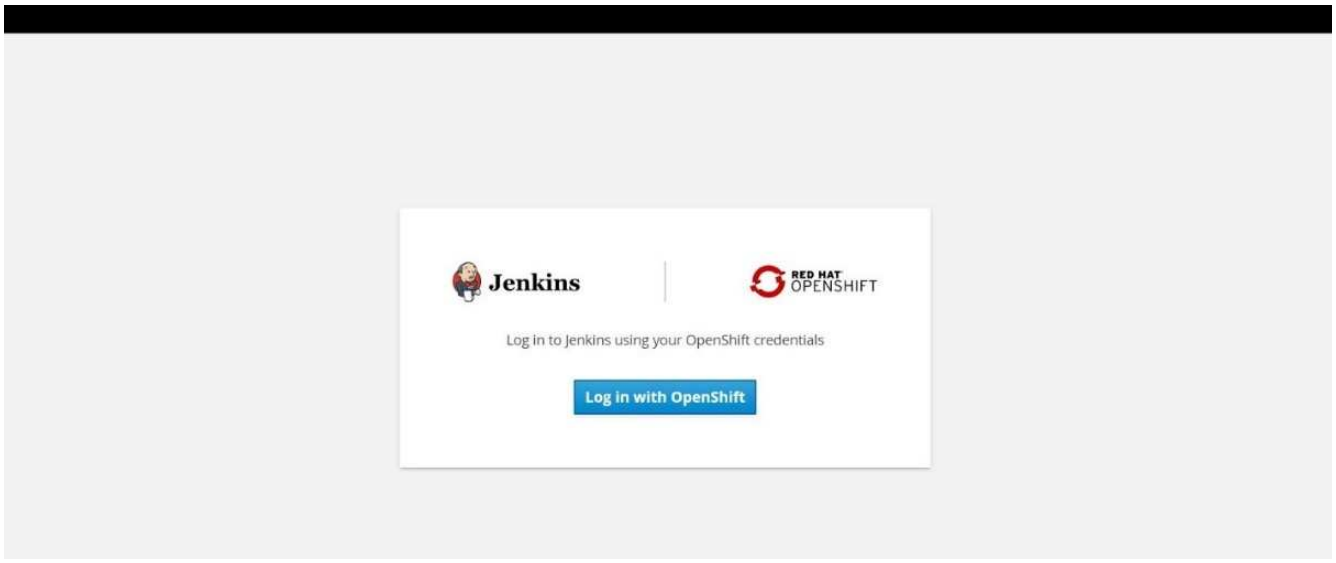
4. Jenkins ポッドが準備完了状態になるまでには約 10 ～ 12 分かかります。



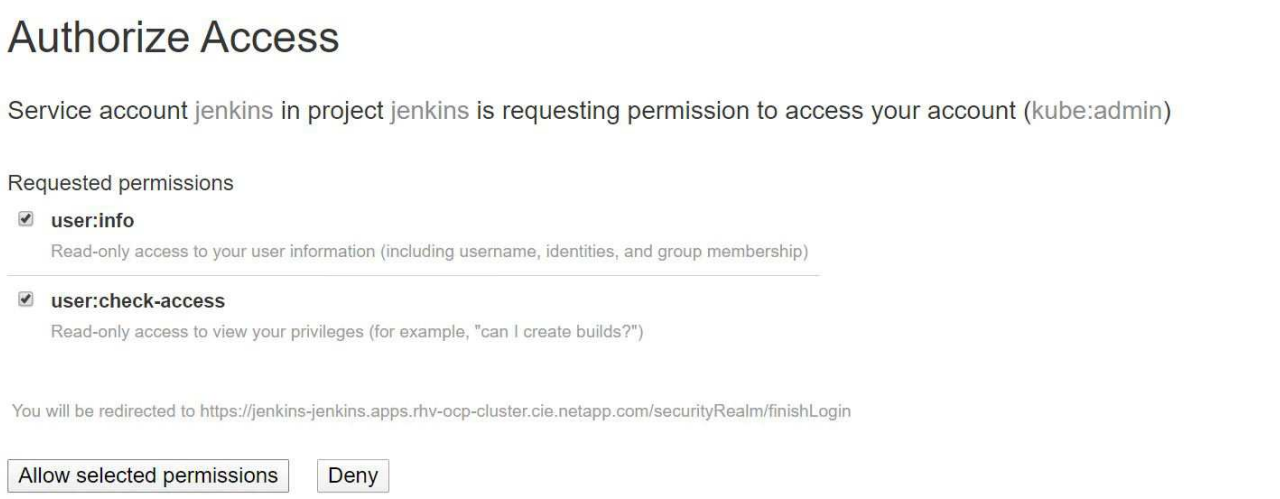
5. ポッドがインスタンス化されたら、「ネットワーク」>「ルート」に移動します。 Jenkins Web ページを開くには、jenkins ルートに提供されている URL をクリックします。



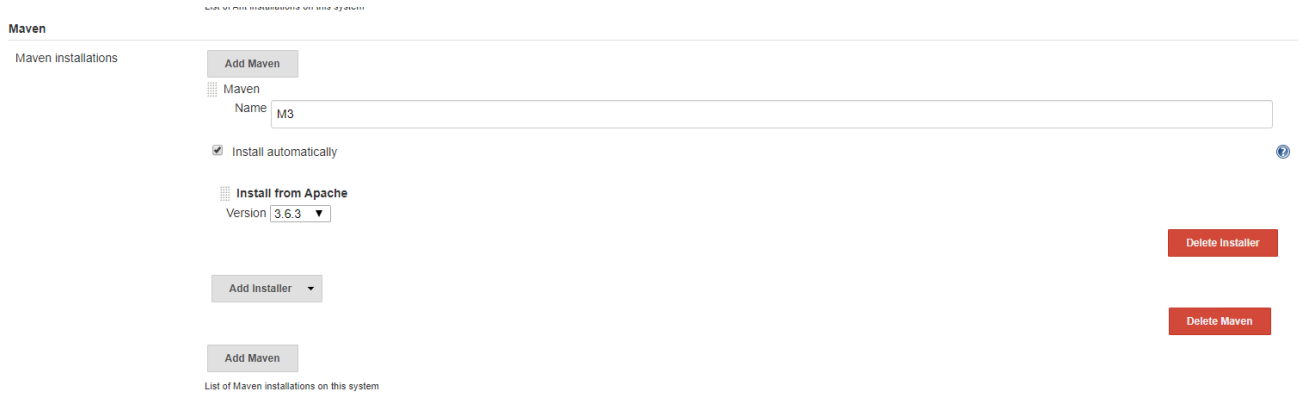
6. Jenkins アプリの作成中に OpenShift OAuth が使用されたため、「OpenShift でログイン」をクリックします。



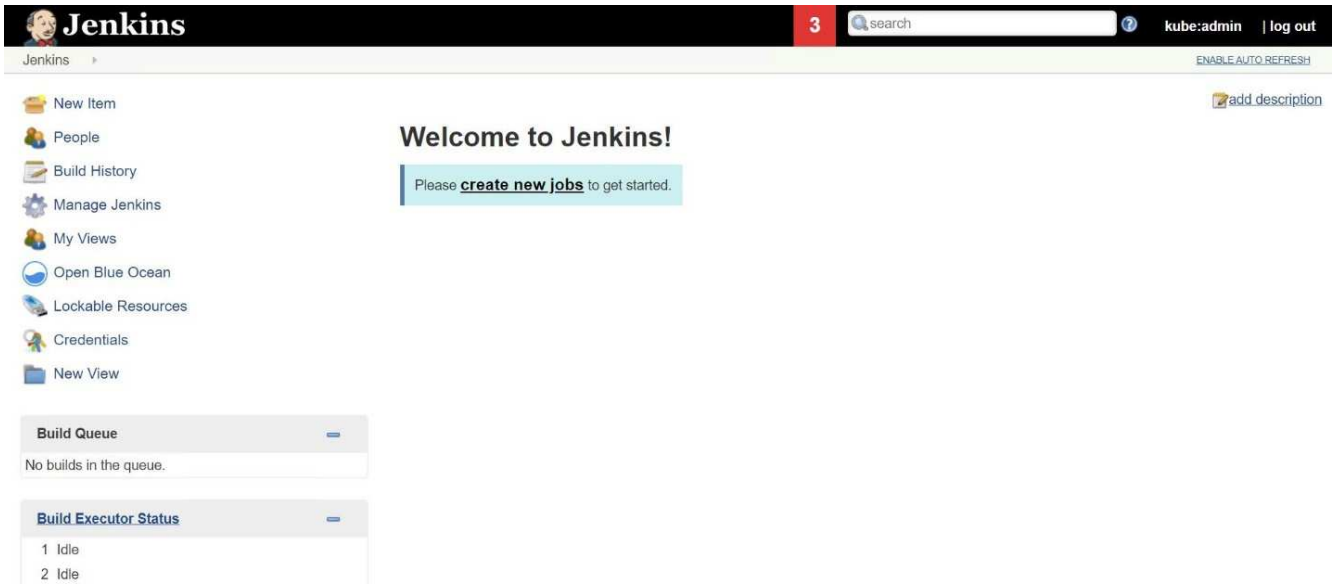
7. Jenkins サービス アカウントに OpenShift ユーザーへのアクセスを許可します。



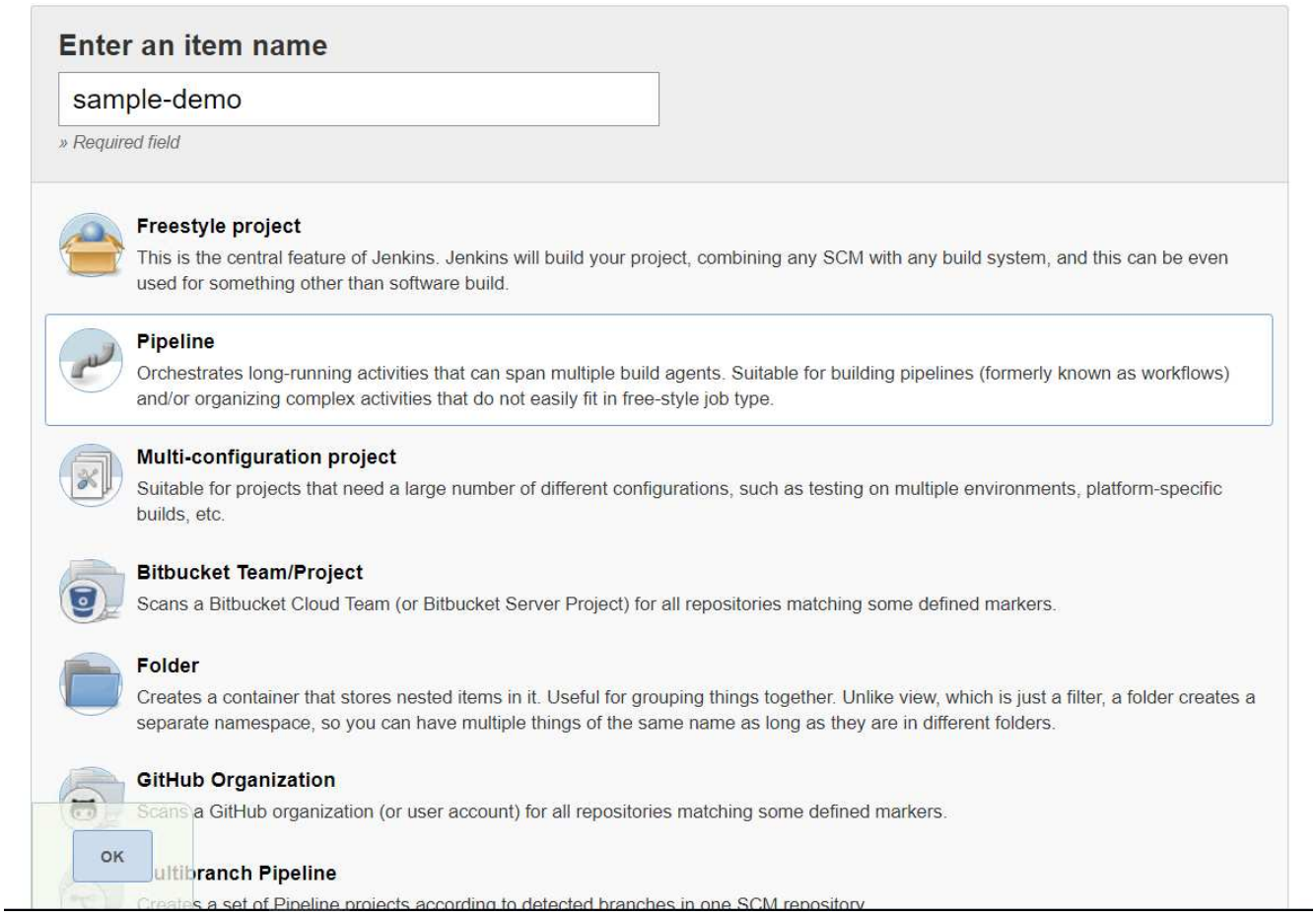
8. Jenkins のウェルカム ページが表示されます。 Maven ビルドを使用しているため、最初に Maven のインストールを完了します。 「Jenkins の管理」>「グローバル ツール構成」に移動し、Maven サブヘッドで「Maven の追加」をクリックします。選択した名前を入力し、「自動的にインストール」オプションが選択されていることを確認します。[Save]をクリックします。



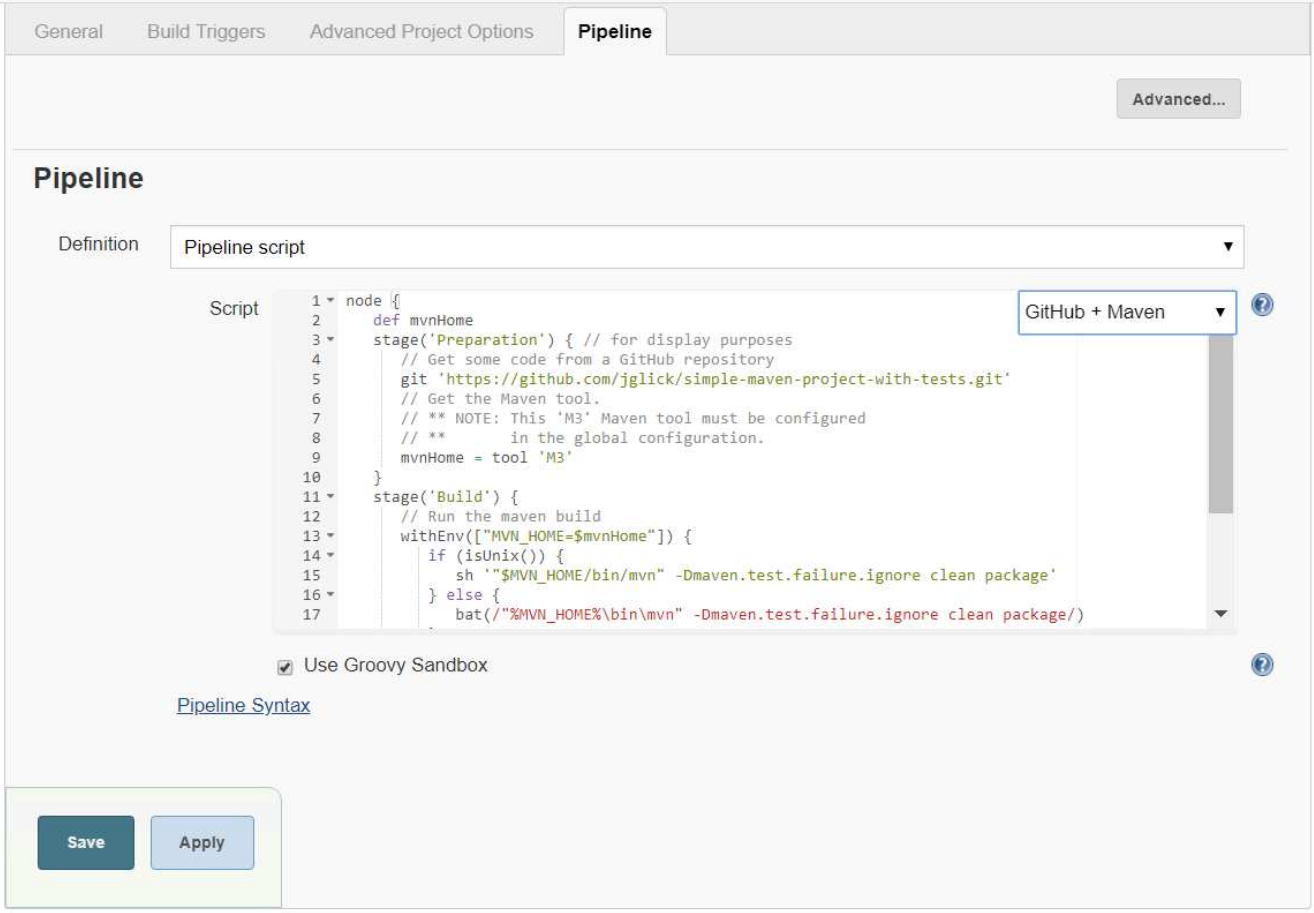
9. CI/CD ワークフローを実証するためのパイプラインを作成できるようになりました。ホームページで、左側のメニューから [新しいジョブの作成] または [新しいアイテム] をクリックします。



10. [アイテムの作成] ページで、任意の名前を入力し、[パイプライン] を選択して、[OK] をクリックします。

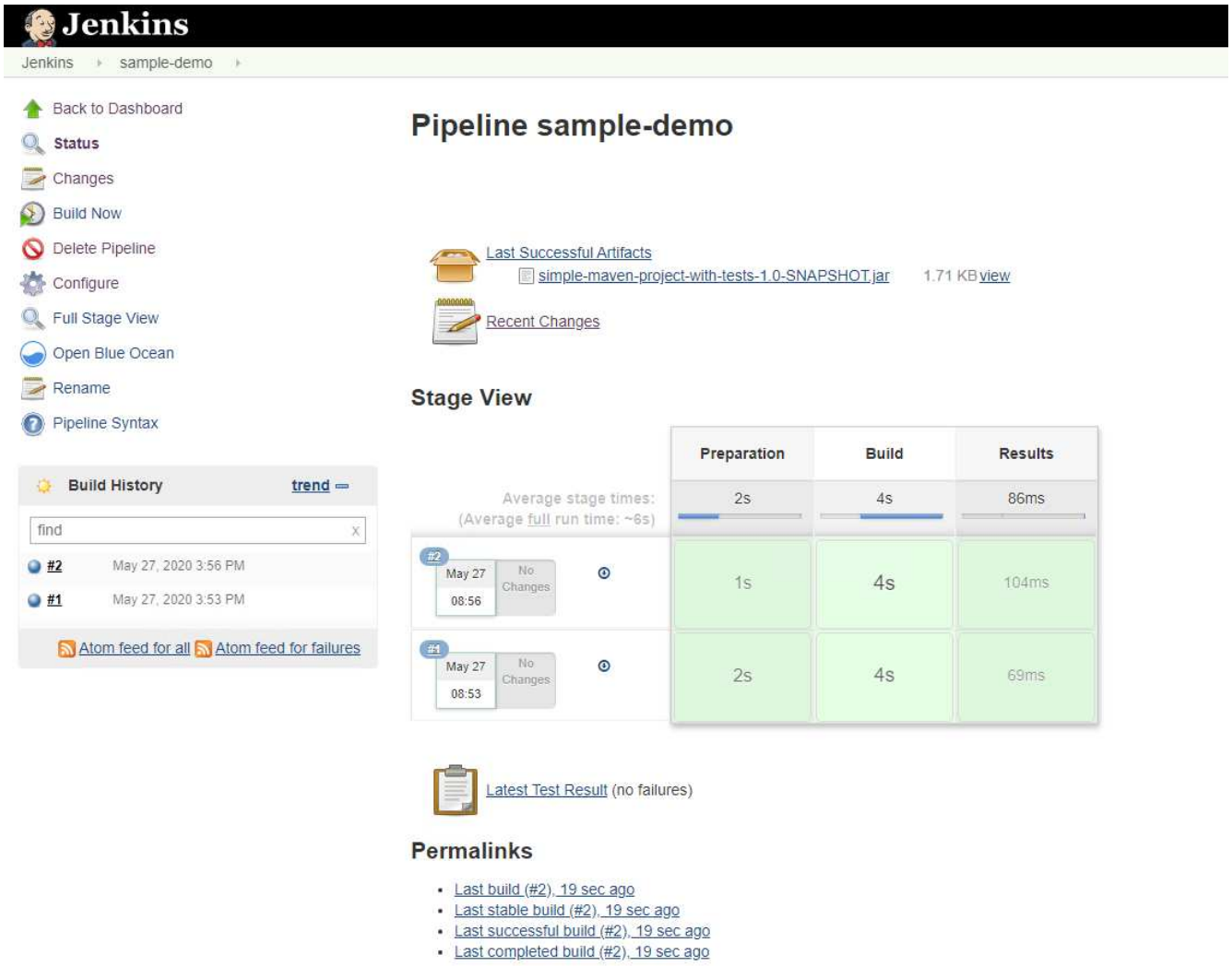


11. パイプラインタブを選択します。 「サンプル パイプラインを試す」ドロップダウン メニューから、「Github + Maven」を選択します。コードは自動的に入力されます。[Save]をクリックします。



12. 「今すぐビルド」をクリックして、準備、ビルド、テストのフェーズを通じて開発を開始します。ビルドプロセス全体を完了し、ビルドの結果が表示されるまでには数分かかる場合があります。

13. コードが変更されるたびに、パイプラインを再構築して新しいバージョンのソフトウェアにパッチを適用できるため、継続的な統合と継続的な配信が可能になります。以前のバージョンからの変更を追跡するには、「最近の変更」をクリックします。



マルチテナントを構成する

NetAppを使用した**Red Hat OpenShift**でのマルチテナントの構成

コンテナ上で複数のアプリケーションやワークロードを実行する多くの組織では、アプリケーションまたはワークロードごとに 1 つの Red Hat OpenShift クラスターをデプロイする傾向があります。これにより、アプリケーションまたはワークロードの厳密な分離を実装し、パフォーマンスを最適化し、セキュリティの脆弱性を軽減することができます。ただし、アプリケーションごとに個別の Red Hat OpenShift クラスターを展開すると、独自の問題が発生します。各クラスターを個別に監視および管理する必要があるため運用オーバーヘッドが増加し、さまざまなアプリケーションに専用のリソースが割り当てられるためコストが増加し、効率的なスケーラビリティが妨げられます。

これらの問題を克服するには、すべてのアプリケーションまたはワークロードを単一の Red Hat OpenShift クラスターで実行することを検討できます。しかし、このようなアーキテクチャでは、リソースの分離とアプリケーションのセキュリティの脆弱性が大きな課題の 1 つとなります。あるワークロードのセキュリティ上の脆弱性は、当然のことながら別のワークロードに波及し、影響範囲が拡大する可能性があります。さらに、デフォルトではリソース割り当てポリシーがないため、あるアプリケーションによる突然の制御されていないリソース使用は、別のアプリケーションのパフォーマンスに影響を及ぼす可能性があります。

したがって、組織は、すべてのワークロードを単一のクラスターで実行しながらも、各ワークロード専用のクラスターの利点を提供するなど、両方の長所を取り入れたソリューションを求めています。

そのような効果的なソリューションの 1 つは、Red Hat OpenShift でマルチテナントを構成することです。マルチテナンシーとは、リソースやセキュリティなどを適切に分離しながら、複数のテナントが同じクラスター上で共存できるようにするアーキテクチャです。このコンテキストでは、テナントは、特定のユーザー グループが排他的な目的で使用するように構成されたクラスター リソースのサブセットとして見ることができます。 Red Hat OpenShift クラスターでマルチテナントを構成すると、次の利点が得られます。

- クラスタリソースの共有を可能にすることで、設備投資と運用コストを削減
- 運用および管理のオーバーヘッドの削減
- セキュリティ侵害の相互汚染からワークロードを保護する
- リソース競合による予期せぬパフォーマンス低下からワークロードを保護する

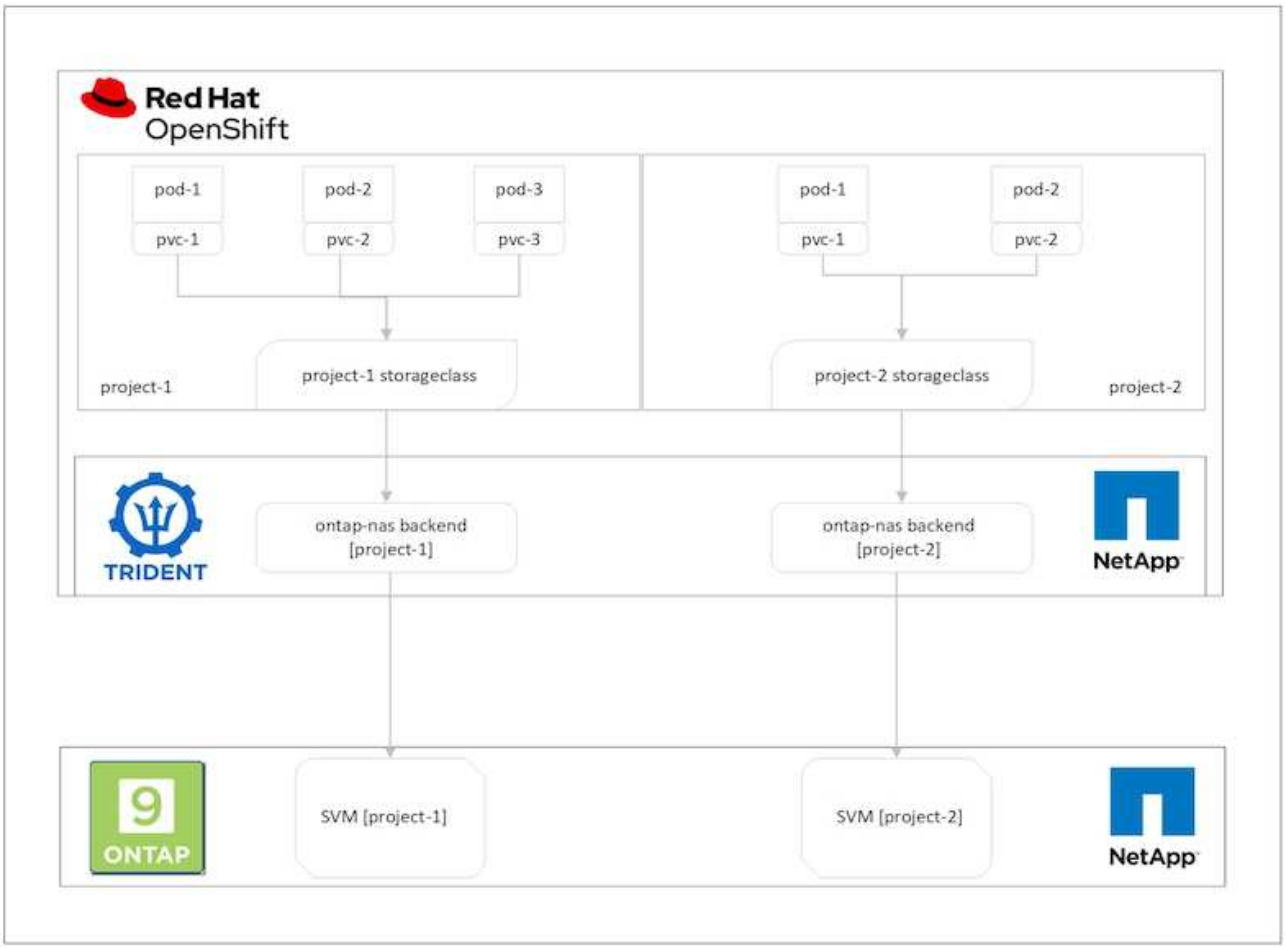
完全に実現されたマルチテナント OpenShift クラスターの場合、コンピューティング、ストレージ、ネットワーク、セキュリティなどのさまざまなリソース バケットに属するクラスター リソースに対してクォータと制限を構成する必要があります。このソリューションでは、すべてのリソース バケットの特定の側面を取り上げますが、NetApp ONTAP を基盤とするTridentによって動的に割り当てられるNetAppONTAPにマルチテナントを構成することで、同じ Red Hat OpenShift クラスター上の複数のワークロードによって提供または消費されるデータを分離して保護するためのベスト プラクティスに重点を置いています。

アーキテクチャ

NetApp ONTAPに裏打ちされた Red Hat OpenShift とTridentは、デフォルトではワークロード間の分離を提供しませんが、マルチテナンシーを構成するために使用できる幅広い機能を提供します。 NetApp ONTAPに裏打ちされたTridentを使用した Red Hat OpenShift クラスターでのマルチテナント ソリューションの設計をより深く理解するために、一連の要件を含む例を検討し、その構成の概要を説明します。

ある組織が、2 つの異なるチームが取り組んでいる 2 つのプロジェクトの一環として、Red Hat OpenShift クラスター上で 2 つのワークロードを実行しているとします。これらのワークロードのデータは、 NetApp ONTAP NAS バックエンドでTridentによって動的にプロビジョニングされる PVC 上に存在します。組織では、これら 2 つのワークロード用のマルチテナント ソリューションを設計し、これらのプロジェクトに使用されるリソースを分離して、主にこれらのアプリケーションに役立つデータに重点を置き、セキュリティとパフォーマンスが維持されるようにする必要があります。

次の図は、 NetApp ONTAPを搭載したTridentを搭載した Red Hat OpenShift クラスター上のマルチテナント ソリューションを示しています。



技術要件

1. NetApp ONTAPストレージ クラスタ
2. Red Hat OpenShift クラスター
3. Trident

Red Hat OpenShift – クラスターリソース

Red Hat OpenShift クラスターの観点から見ると、最初に開始する最上位のリソースはプロジェクトです。OpenShift プロジェクトは、OpenShift クラスター全体を複数の仮想クラスターに分割するクラスター リソースとして考えることができます。したがって、プロジェクト レベルでの分離は、マルチテナントを構成するための基盤となります。

次は、クラスターで RBAC を構成します。ベスト プラクティスは、単一のプロジェクトまたはワークロードで作業するすべての開発者を、アイデンティティ プロバイダー (IdP) 内の単一のユーザー グループに構成することです。 Red Hat OpenShift では、IdP の統合とユーザー グループの同期が可能になり、IdP からのユーザーとグループをクラスターにインポートできるようになります。これにより、クラスター管理者は、プロジェクト専用のクラスター リソースへのアクセスを、そのプロジェクトで作業しているユーザー グループに分離し、クラスター リソースへの不正アクセスを制限できるようになります。 Red Hat OpenShiftとのIdP統合の詳細については、ドキュメントを参照してください。 "ここをクリックしてください。" 。

NetApp ONTAP

Red Hat OpenShift クラスターの永続ストレージプロバイダーとして機能する共有ストレージを分離して、各プロジェクトのストレージ上に作成されたボリュームが、ホストに対して個別のストレージ上に作成されたかのように見えるようにすることが重要です。これを行うには、 NetApp ONTAP上にプロジェクトまたはワークロードと同じ数の SVM (ストレージ仮想マシン) を作成し、各 SVM をワークロード専用にします。

Trident

NetApp ONTAPで異なるプロジェクト用に異なる SVM を作成したら、各 SVM を異なるTridentバックエンドにマップする必要があります。 Tridentのバックエンド構成は、OpenShift クラスター リソースへの永続ストレージの割り当てを制御し、マップする SVM の詳細を必要とします。これは、少なくともバックエンドのプロトコル ドライバーである必要があります。オプションで、ストレージ上でボリュームをプロビジョニングする方法を定義したり、ボリュームのサイズやアグリゲートの使用法などの制限を設定したりすることもできます。 Tridentバックエンドの定義に関する詳細は、 "ここをクリックしてください。" 。

Red Hat OpenShift – ストレージリソース

Tridentバックエンドを構成した後、次のステップは StorageClasses を構成することです。バックエンドと同じ数のストレージ クラスを構成し、各ストレージ クラスに 1 つのバックエンドでのみボリュームを起動するアクセスを提供します。ストレージ クラスを定義するときに storagePools パラメータを使用することで、StorageClass を特定のTridentバックエンドにマップできます。ストレージクラスの定義の詳細については、 "ここをクリックしてください。" 。したがって、StorageClass から 1 つの SVM を指すTridentバックエンドへの 1 対 1 のマッピングがあります。これにより、そのプロジェクトに割り当てられた StorageClass 経由のすべてのストレージ要求が、そのプロジェクト専用の SVM によってのみ処理されるようになります。

ストレージ クラスは名前空間リソースではないため、別の名前空間またはプロジェクトのポッドによる 1 つのプロジェクトのストレージ クラスへのストレージ要求が拒否されることをどのように保証するのでしょうか。答えは、ResourceQuotas を使用することです。 ResourceQuotas は、プロジェクトごとのリソースの合計使用量を制御するオブジェクトです。プロジェクト内のオブジェクトが消費できるリソースの数と合計量を制限できます。 ResourceQuotas を使用すると、プロジェクトのほぼすべてのリソースを制限できます。これを効率的に使用することで、組織はリソースの過剰プロビジョニングや過剰消費によるコストと停止を削減できます。ドキュメントを参照してください "ここをクリックしてください。"詳細についてはこちらをご覧ください。

このユースケースでは、特定のプロジェクト内のポッドが、そのプロジェクト専用ではないストレージ クラスからストレージを要求することを制限する必要があります。そのためには、他のストレージクラスの永続ボリュームの要求を制限する必要があります。 `<storage-class-name>.storageclass.storage.k8s.io/persistentvolumeclaims` 0 にします。さらに、クラスター管理者は、プロジェクト内の開発者が ResourceQuotas を変更するアクセス権を持たないようにする必要があります。

構成

どのマルチテナント ソリューションでも、ユーザーは必要以上のクラスター リソースにアクセスすることはできません。したがって、マルチテナント構成の一部として構成されるリソースのセット全体は、クラスター管理者、ストレージ管理者、および各プロジェクトで作業する開発者の間で分割されます。

次の表は、さまざまなユーザーが実行するさまざまなタスクの概要を示しています。

ロール	タスク
クラスター管理者	さまざまなアプリケーションやワークロード用のプロジェクトを作成する
	ストレージ管理者用の ClusterRoles と RoleBindings を作成する
	開発者が特定のプロジェクトへのアクセスを割り当てるためのロールとロールバインディングを作成する
	[オプション] 特定のノードでポッドをスケジュールするようにプロジェクトを構成する
ストレージ管理者	NetApp ONTAP上に SVM を作成する
	Tridentバックエンドを作成する
	ストレージクラスを作成する
	ストレージリソースクォータを作成する
開発者	割り当てられたプロジェクトで PVC またはポッドを作成またはパッチ適用するためのアクセスを検証します
	別のプロジェクトで PVC またはポッドを作成またはパッチ適用するためのアクセスを検証する
	プロジェクト、リソースクォータ、ストレージクラスの表示または編集へのアクセスを検証します

構成

NetAppを使用して Red Hat OpenShift でマルチテナントを構成するための前提条件は次のとおりです。

前提条件

- NetApp ONTAPクラスタ
- Red Hat OpenShift クラスター
- クラスターにインストールされたTrident
- tridentctl および oc ツールがインストールされ、$PATH に追加された管理ワークステーション
- ONTAPへの管理者アクセス
- OpenShift クラスターへのクラスター管理者アクセス
- クラスタはアイデンティティプロバイダと統合されています
- アイデンティティプロバイダは、異なるチームのユーザーを効率的に区別するように構成されている

構成: クラスター管理タスク

Red Hat OpenShift クラスター管理者によって次のタスクが実行されます。

1. Red Hat OpenShift クラスターに cluster-admin としてログインします。

2. 異なるプロジェクトに対応する 2 つのプロジェクトを作成します。

```
oc create namespace project-1
oc create namespace project-2
```

3. project-1 の開発者ロールを作成します。

```
cat << EOF | oc create -f -
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: project-1
  name: developer-project-1
rules:
  - verbs:
      - '*'
    apiGroups:
      - apps
      - batch
      - autoscaling
      - extensions
      - networking.k8s.io
      - policy
      - apps.openshift.io
      - build.openshift.io
      - image.openshift.io
      - ingress.operator.openshift.io
      - route.openshift.io
      - snapshot.storage.k8s.io
      - template.openshift.io
    resources:
      - '*'
  - verbs:
      - '*'
    apiGroups:
      - ''
    resources:
      - bindings
      - configmaps
      - endpoints
      - events
      - persistentvolumeclaims
      - pods
      - pods/log
      - pods/attach
```

```
      - podtemplates
      - replicationcontrollers
      - services
      - limitranges
      - namespaces
      - componentstatuses
      - nodes
  - verbs:
      - '*'
    apiGroups:
      - trident.netapp.io
    resources:
      - tridentsnapshots
EOF
```

このセクションで提供されているロール定義は単なる例です。開発者の役割は、エンドユーザーの要件に基づいて定義する必要があります。

1. 同様に、プロジェクト 2 の開発者ロールを作成します。
2. すべての OpenShift およびNetAppストレージ リソースは通常、ストレージ管理者によって管理されます。ストレージ管理者のアクセスは、Tridentのインストール時に作成される Trident オペレーター ロールによって制御されます。これに加えて、ストレージ管理者は、ストレージの消費方法を制御するために ResourceQuotas へのアクセス権も必要とします。
3. クラスター内のすべてのプロジェクトで ResourceQuotas を管理するためのロールを作成し、それをストレージ管理者にアタッチします。

```
cat << EOF | oc create -f -
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: resource-quotas-role
rules:
  - verbs:
      - '*'
    apiGroups:
      - ''
    resources:
      - resourcequotas
  - verbs:
      - '*'
    apiGroups:
      - quota.openshift.io
    resources:
      - '*'
EOF
```

4. クラスターが組織の ID プロバイダーと統合され、ユーザー グループがクラスター グループと同期されていることを確認します。次の例は、アイデンティティ プロバイダーがクラスターと統合され、ユーザー グループと同期されていることを示しています。

```
$ oc get groups
NAME                          USERS
ocp-netapp-storage-admins     ocp-netapp-storage-admin
ocp-project-1                 ocp-project-1-user
ocp-project-2                 ocp-project-2-user
```

1. ストレージ管理者の ClusterRoleBindings を構成します。

```
cat << EOF | oc create -f -
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: netapp-storage-admin-trident-operator
subjects:
  - kind: Group
    apiGroup: rbac.authorization.k8s.io
    name: ocp-netapp-storage-admins
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: trident-operator
---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: netapp-storage-admin-resource-quotas-cr
subjects:
  - kind: Group
    apiGroup: rbac.authorization.k8s.io
    name: ocp-netapp-storage-admins
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: resource-quotas-role
EOF
```

ストレージ管理者の場合、 trident-operator と resource-quotas の 2 つのロールをバインドする必要があります。

1. 開発者用の RoleBinding を作成し、developer-project-1 ロールを project-1 内の対応するグループ (ocp-project-1) にバインドします。

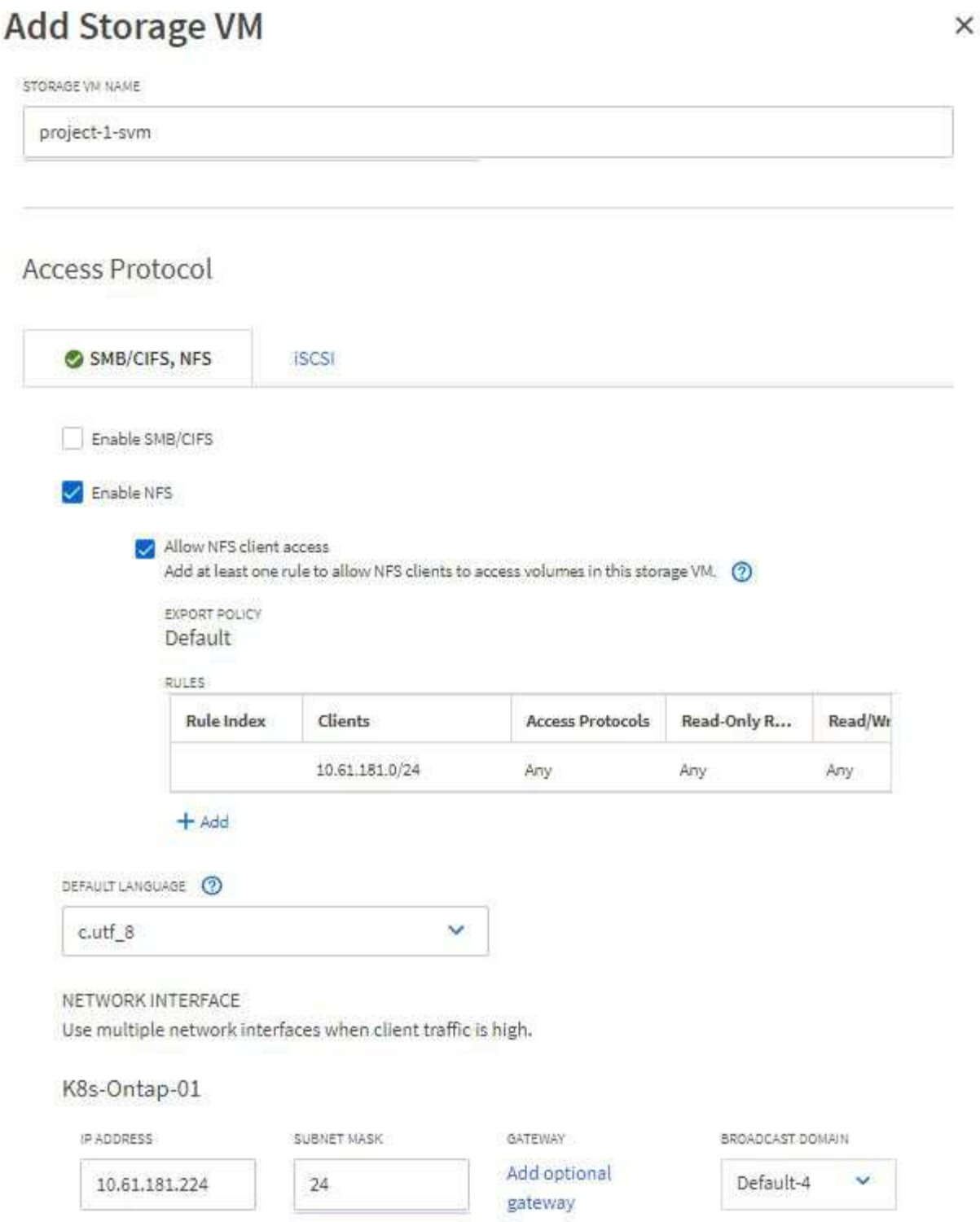
```
cat << EOF | oc create -f -
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: project-1-developer
  namespace: project-1
subjects:
  - kind: Group
    apiGroup: rbac.authorization.k8s.io
    name: ocp-project-1
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: developer-project-1
EOF
```

2. 同様に、開発者ロールをプロジェクト 2 の対応するユーザー グループにバインドする開発者用の RoleBinding を作成します。

構成: ストレージ管理タスク

次のリソースは、ストレージ管理者が構成する必要があります。

1. NetApp ONTAPクラスターに管理者としてログインします。
2. [ストレージ] > [ストレージ VM] に移動し、[追加] をクリックします。必要な詳細を指定して、プロジェクト 1 用とプロジェクト 2 用の 2 つの SVM を作成します。また、SVM とそのリソースを管理するための vsadmin アカウントも作成します。



1. ストレージ管理者として Red Hat OpenShift クラスターにログインします。
2. project-1 のバックエンドを作成し、それをプロジェクト専用の SVM にマップします。 NetApp、 ONTAP クラスタ管理者を使用する代わりに、SVM の vsadmin アカウントを使用してバックエンドを SVM に接続することをお勧めします。

```
cat << EOF | tridentctl -n trident create backend -f
{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "backendName": "nfs_project_1",
    "managementLIF": "172.21.224.210",
    "dataLIF": "10.61.181.224",
    "svm": "project-1-svm",
    "username": "vsadmin",
    "password": "NetApp123"
}
EOF
```

この例では、ontap-nas ドライバーを使用しています。ユースケースに基づいてバックエンドを作成するときは、適切なドライバーを使用します。

Trident がtrident プロジェクトにインストールされていることを前提としています。

1. 同様に、project-2 のTridentバックエンドを作成し、それを project-2 専用の SVM にマップします。
2. 次に、ストレージ クラスを作成します。 project-1 のストレージ クラスを作成し、storagePools パラメータを設定して、project-1 専用のバックエンドのストレージ プールを使用するように構成します。

```
cat << EOF | oc create -f -
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: project-1-sc
provisioner: csi.trident.netapp.io
parameters:
  backendType: ontap-nas
  storagePools: "nfs_project_1:.*"
EOF
```

3. 同様に、project-2 のストレージ クラスを作成し、project-2 専用のバックエンドのストレージ プールを使用するように構成します。
4. ResourceQuota を作成して、他のプロジェクト専用のストレージクラスからストレージを要求する project-1 のリソースを制限します。

```
cat << EOF | oc create -f -
kind: ResourceQuota
apiVersion: v1
metadata:
  name: project-1-sc-rq
  namespace: project-1
spec:
  hard:
    project-2-sc.storageclass.storage.k8s.io/persistentvolumeclaims: 0
EOF
```

5. 同様に、ResourceQuota を作成して、他のプロジェクト専用のストレージクラスからストレージを要求する project-2 のリソースを制限します。

検証

前の手順で構成されたマルチテナント アーキテクチャを検証するには、次の手順を実行します。

割り当てられたプロジェクトで **PVC** またはポッドを作成するためのアクセスを検証する

1. project-1 の開発者、ocp-project-1-user としてログインします。
2. 新しいプロジェクトを作成するにはアクセスを確認してください。

```
oc create ns sub-project-1
```

3. project-1 に割り当てられているストレージクラスを使用して、project-1 に PVC を作成します。

```
cat << EOF | oc create -f -
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: test-pvc-project-1
  namespace: project-1
  annotations:
    trident.netapp.io/reclaimPolicy: Retain
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: project-1-sc
EOF
```

4. PVC に関連付けられている PV を確認します。

```
oc get pv
```

5. NetApp ONTAP上の project-1 専用の SVM に PV とそのボリュームが作成されていることを確認します。

```
volume show -vserver project-1-svm
```

6. project-1 にポッドを作成し、前の手順で作成した PVC をマウントします。

```
cat << EOF | oc create -f -
kind: Pod
apiVersion: v1
metadata:
  name: test-pvc-pod
  namespace: project-1
spec:
  volumes:
    - name: test-pvc-project-1
      persistentVolumeClaim:
       claimName: test-pvc-project-1
  containers:
    - name: test-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/usr/share/nginx/html"
          name: test-pvc-project-1
EOF
```

7. ポッドが実行中かどうか、ボリュームがマウントされているかどうかを確認します。

```
oc describe pods test-pvc-pod -n project-1
```

別のプロジェクトで **PVC** またはポッドを作成したり、別のプロジェクト専用のリソースを使用したりするためのアクセスを検証します

1. project-1 の開発者、ocp-project-1-user としてログインします。
2. project-2 に割り当てられているストレージクラスを使用して、project-1 に PVC を作成します。

```
cat << EOF | oc create -f -
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: test-pvc-project-1-sc-2
  namespace: project-1
  annotations:
    trident.netapp.io/reclaimPolicy: Retain
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: project-2-sc
EOF
```

3. プロジェクト 2 に PVC を作成します。

```
cat << EOF | oc create -f -
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: test-pvc-project-2-sc-1
  namespace: project-2
  annotations:
    trident.netapp.io/reclaimPolicy: Retain
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: project-1-sc
EOF
```

4. PVCが `test-pvc-project-1-sc-2`そして `test-pvc-project-2-sc-1`作成されませんでした。

```
oc get pvc -n project-1
oc get pvc -n project-2
```

5. project-2 にポッドを作成します。

```
cat << EOF | oc create -f -
kind: Pod
apiVersion: v1
metadata:
  name: test-pvc-pod
  namespace: project-1
spec:
  containers:
    - name: test-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
EOF
```

プロジェクト、リソースクォータ、ストレージクラスの表示と編集へのアクセスを検証する

1. project-1 の開発者、ocp-project-1-user としてログインします。
2. 新しいプロジェクトを作成するためのアクセスを確認します。

```
oc create ns sub-project-1
```

3. プロジェクトを表示するためのアクセスを検証します。

```
oc get ns
```

4. ユーザーが project-1 の ResourceQuotas を表示または編集できるかどうかを確認します。

```
oc get resourcequotas -n project-1
oc edit resourcequotas project-1-sc-rq -n project-1
```

5. ユーザーがストレージクラスを表示するアクセス権を持っていることを確認します。

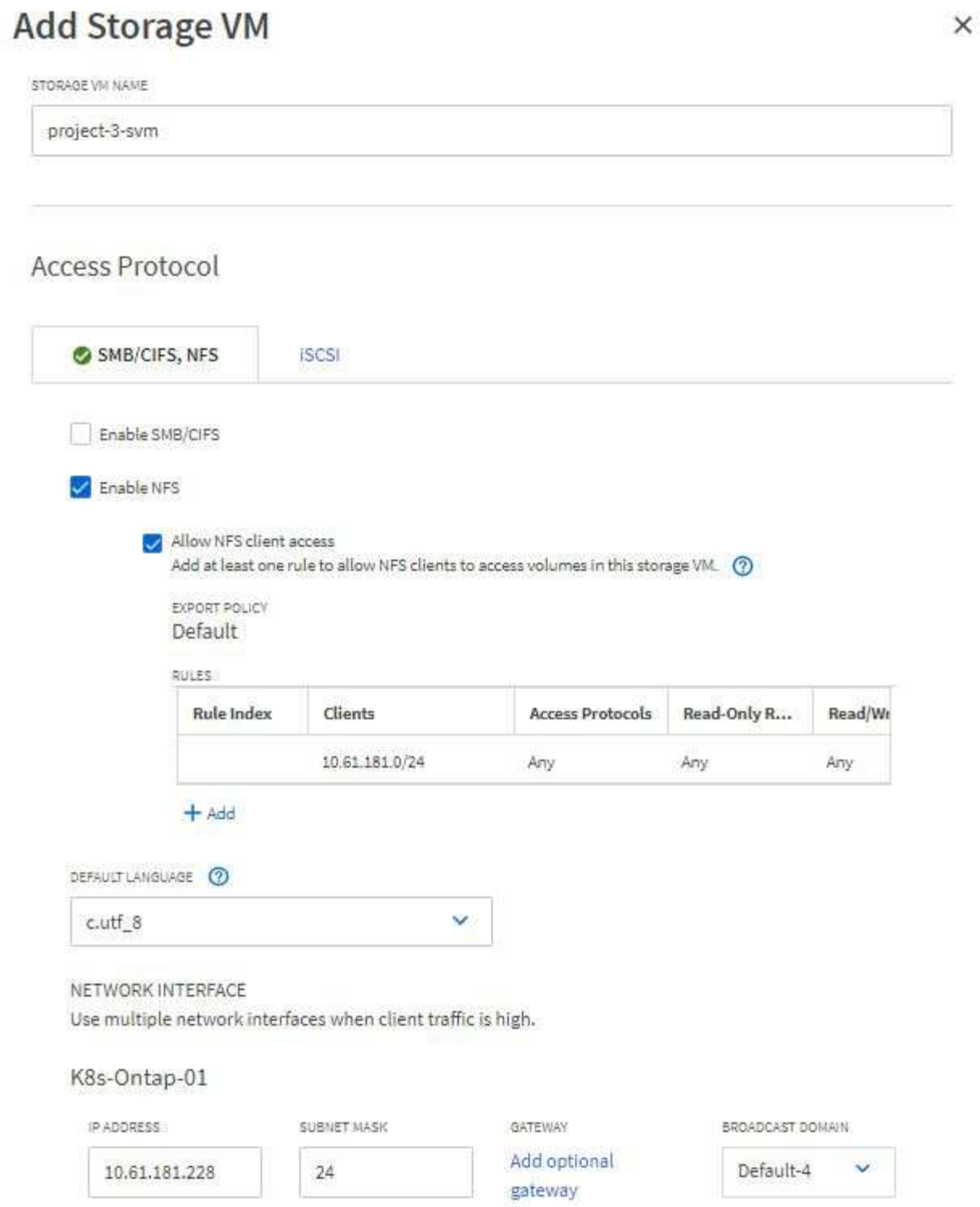
```
oc get sc
```

6. ストレージクラスを記述するためのアクセスを確認します。
7. ストレージクラスを編集するためのユーザーのアクセスを検証します。

```
oc edit sc project-1-sc
```

マルチテナント構成では、ストレージ リソースを含む新しいプロジェクトを追加する場合、マルチテナント性が侵害されないようにするための追加の構成が必要です。マルチテナント クラスターにプロジェクトを追加するには、次の手順を実行します。

1. ストレージ管理者としてNetApp ONTAPクラスターにログインします。
2. 移動先 `Storage → Storage VMs`クリック `Add`。プロジェクト 3 専用の新しい SVM を作成します。また、SVM とそのリソースを管理するための vsadmin アカウントも作成します。



1. クラスター管理者として Red Hat OpenShift クラスターにログインします。
2. 新しいプロジェクトを作成します。

```
oc create ns project-3
```

3. project-3 のユーザー グループが IdP 上に作成され、OpenShift クラスターと同期されていることを確認します。

```
oc get groups
```

4. project-3 の開発者ロールを作成します。

```
cat << EOF | oc create -f -
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: project-3
  name: developer-project-3
rules:
  - verbs:
      - '*'
    apiGroups:
      - apps
      - batch
      - autoscaling
      - extensions
      - networking.k8s.io
      - policy
      - apps.openshift.io
      - build.openshift.io
      - image.openshift.io
      - ingress.operator.openshift.io
      - route.openshift.io
      - snapshot.storage.k8s.io
      - template.openshift.io
    resources:
      - '*'
  - verbs:
      - '*'
    apiGroups:
      - ''
    resources:
      - bindings
      - configmaps
      - endpoints
      - events
      - persistentvolumeclaims
      - pods
      - pods/log
      - pods/attach
```

```
      - podtemplates
      - replicationcontrollers
      - services
      - limitranges
      - namespaces
      - componentstatuses
      - nodes
  - verbs:
      - '*'
    apiGroups:
      - trident.netapp.io
    resources:
      - tridentsnapshots
EOF
```

このセクションで提供されているロール定義は単なる例です。開発者ロールは、エンドユーザーの要件に基づいて定義する必要があります。

1. project-3 の開発者用の RoleBinding を作成し、developer-project-3 ロールを project-3 の対応するグループ (ocp-project-3) にバインドします。

```
cat << EOF | oc create -f -
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: project-3-developer
  namespace: project-3
subjects:
  - kind: Group
    apiGroup: rbac.authorization.k8s.io
    name: ocp-project-3
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: developer-project-3
EOF
```

2. Red Hat OpenShift クラスタにストレージ管理者としてログインします。
3. Tridentバックエンドを作成し、それを project-3 専用の SVM にマップします。 NetApp、 ONTAPクラスタ管理者を使用する代わりに、SVM の vsadmin アカウントを使用してバックエンドを SVM に接続することをお勧めします。

```
cat << EOF | tridentctl -n trident create backend -f
{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "backendName": "nfs_project_3",
    "managementLIF": "172.21.224.210",
    "dataLIF": "10.61.181.228",
    "svm": "project-3-svm",
    "username": "vsadmin",
    "password": "NetApp!23"
}
EOF
```

この例では、ontap-nas ドライバーを使用しています。ユースケースに基づいてバックエンドを作成するには、適切なドライバーを使用します。

Trident がtrident プロジェクトにインストールされていることを前提としています。

1. project-3 のストレージ クラスを作成し、project-3 専用のバックエンドのストレージ プールを使用するように構成します。

```
cat << EOF | oc create -f -
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: project-3-sc
provisioner: csi.trident.netapp.io
parameters:
  backendType: ontap-nas
  storagePools: "nfs_project_3:.*"
EOF
```

2. ResourceQuota を作成して、他のプロジェクト専用のストレージクラスからストレージを要求するプロジェクト 3 のリソースを制限します。

```
cat << EOF | oc create -f -
kind: ResourceQuota
apiVersion: v1
metadata:
  name: project-3-sc-rq
  namespace: project-3
spec:
  hard:
    project-1-sc.storageclass.storage.k8s.io/persistentvolumeclaims: 0
    project-2-sc.storageclass.storage.k8s.io/persistentvolumeclaims: 0
EOF
```

3. 他のプロジェクトの ResourceQuotas にパッチを適用して、それらのプロジェクトのリソースが project-3 専用のストレージクラスのストレージにアクセスすることを制限します。

```
oc patch resourcequotas project-1-sc-rq -n project-1 --patch
'{"spec":{"hard":{ "project-3-
sc.storageclass.storage.k8s.io/persistentvolumeclaims": 0}}}'
oc patch resourcequotas project-2-sc-rq -n project-2 --patch
'{"spec":{"hard":{ "project-3-
sc.storageclass.storage.k8s.io/persistentvolumeclaims": 0}}}'
```

Kubernetes の高度なクラスタ管理

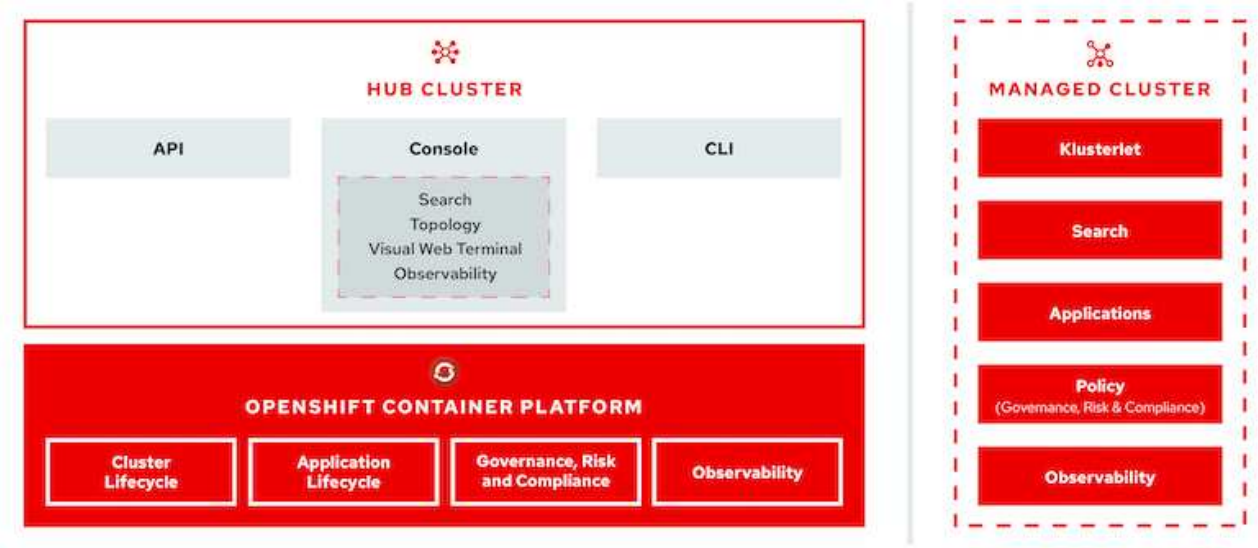
Kubernetes の高度なクラスタ管理: Red Hat OpenShift とNetApp - 概要

コンテナ化されたアプリケーションが開発から本番環境に移行すると、多くの組織では、そのアプリケーションのテストとデプロイメントをサポートするために複数の Red Hat OpenShift クラスターが必要になります。これに関連して、組織は通常、OpenShift クラスター上で複数のアプリケーションまたはワークロードをホストします。したがって、各組織は最終的に一連のクラスターを管理することになり、OpenShift 管理者は複数のオンプレミス データセンターとパブリック クラウドにまたがるさまざまな環境にわたって複数のクラスターを管理および維持するという追加の課題に直面することになります。これらの課題に対処するために、Red Hat は Kubernetes 向けの高度なクラスター管理を導入しました。

Red Hat Advanced Cluster Management for Kubernetes を使用すると、次のタスクを実行できます。

1. データセンターとパブリッククラウドにわたる複数のクラスタを作成、インポート、管理します
2. 単一のコンソールから複数のクラスタ上のアプリケーションまたはワークロードを展開および管理します
3. さまざまなクラスターリソースの健全性とステータスを監視および分析します
4. 複数のクラスタにわたってセキュリティコンプライアンスを監視および適用する

Red Hat Advanced Cluster Management for Kubernetes は、Red Hat OpenShift クラスターのアドオンとしてインストールされ、このクラスターをすべての操作の中央コントローラーとして使用します。このクラスターはハブ クラスターと呼ばれ、ユーザーが Advanced Cluster Management に接続するための管理プレーンを公開します。 Advanced Cluster Management コンソールを使用してインポートまたは作成された他のすべての OpenShift クラスターは、ハブ クラスターによって管理され、管理対象クラスターと呼ばれます。管理対象クラスターに Klusterlet と呼ばれるエージェントをインストールして、それらをハブ クラスターに接続し、クラスター ライフサイクル管理、アプリケーション ライフサイクル管理、可観測性、セキュリティ コンプライアンスに関連するさまざまなアクティビティの要求に応えます。



詳細については、ドキュメントを参照してください。 "ここをクリックしてください。" 。

Kubernetes 向け ACM をデプロイする

Kubernetes 向けの高度なクラスタ管理を導入する

このセクションでは、 NetAppを使用した Red Hat OpenShift 上の Kubernetes の高度なクラスター管理について説明します。

前提条件

1. ハブ クラスター用の Red Hat OpenShift クラスター (バージョン 4.5 以上)
2. マネージド クラスター用の Red Hat OpenShift クラスター (バージョン 4.4.3 以上)
3. Red Hat OpenShift クラスターへのクラスター管理者アクセス
4. Kubernetes向けAdvanced Cluster ManagementのRed Hatサブスクリプション

Advanced Cluster Management は OpenShift クラスターのアドオンであるため、ハブおよび管理対象クラスター全体で使用される機能に基づいて、ハードウェア リソースに特定の要件と制限があります。クラスターのサイズを決定する際には、これらの問題を考慮する必要があります。ドキュメントを参照 "ここをクリックしてください。"詳細についてはこちらをご覧ください。

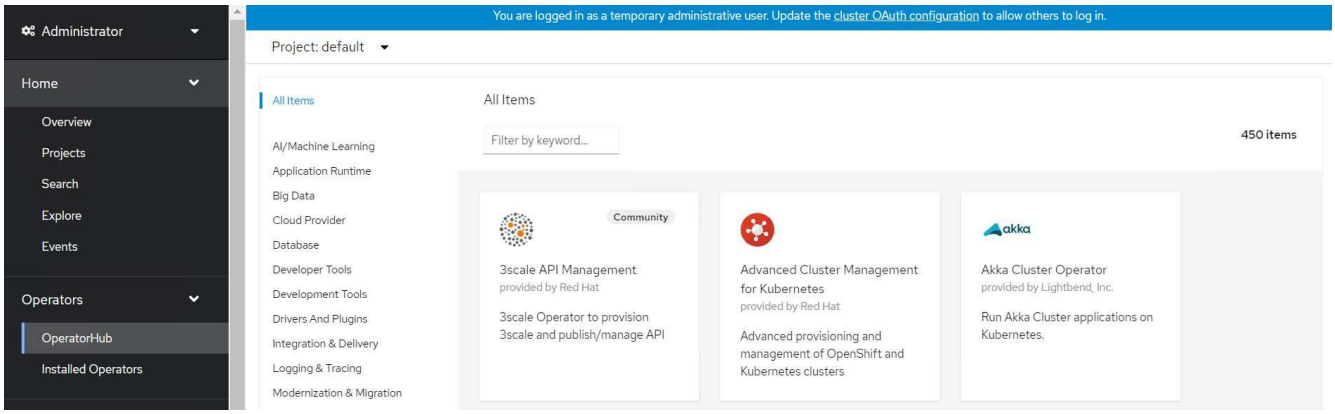
オプションとして、ハブ クラスターにインフラストラクチャ コンポーネントをホストするための専用ノード

があり、それらのノードにのみ Advanced Cluster Management リソースをインストールする場合は、それに応じてそれらのノードに許容値とセレクターを追加する必要があります。詳細については、ドキュメントを参照してください。"ここをクリックしてください。"。

Kubernetes 向けの高度なクラスタ管理を導入する

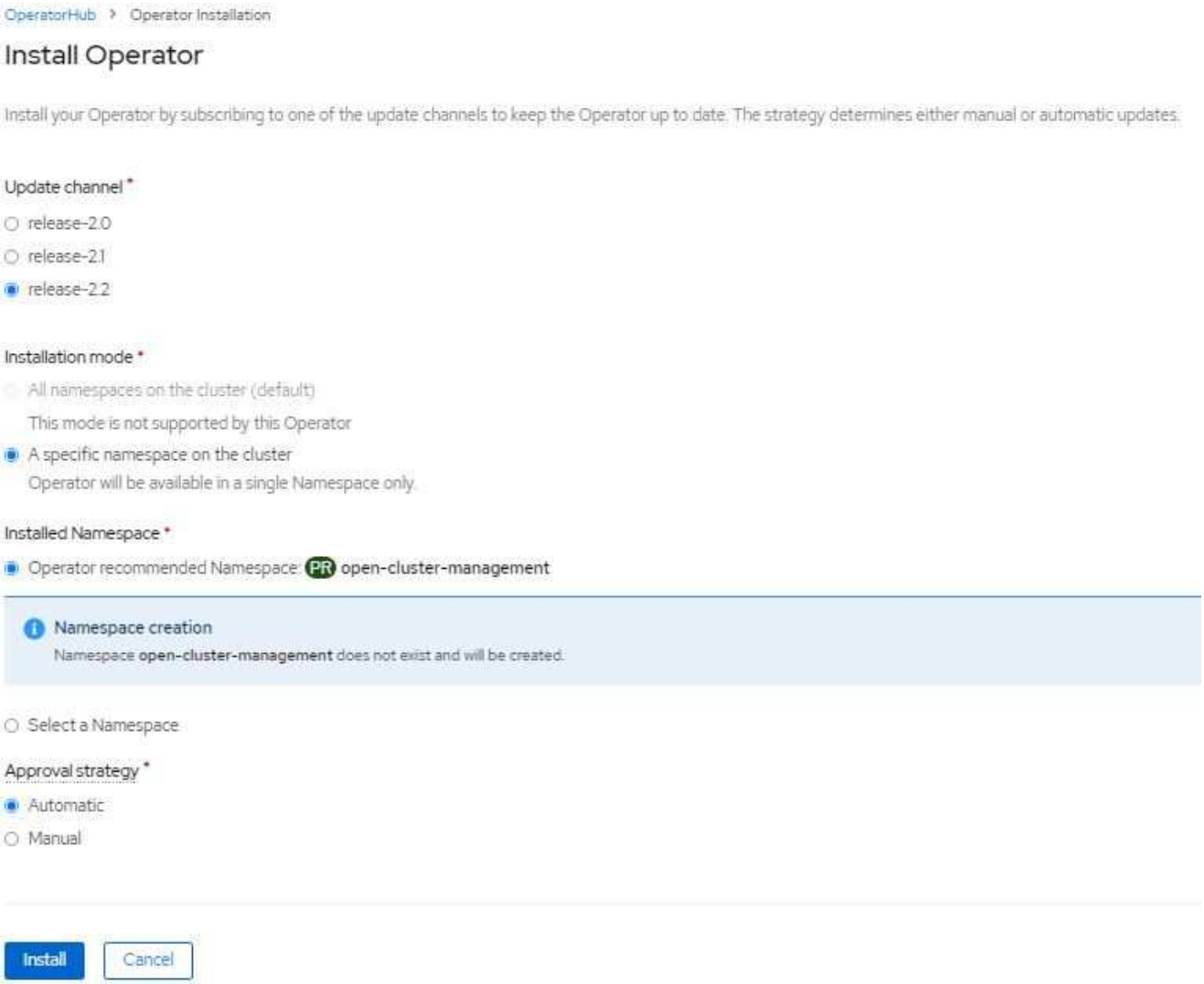
OpenShift クラスターに Advanced Cluster Management for Kubernetes をインストールするには、次の手順を実行します。

1. OpenShift クラスターをハブ クラスターとして選択し、cluster-admin 権限でログインします。
2. 「Operators」>「Operators Hub」に移動し、「Advanced Cluster Management for Kubernetes」を検索します。

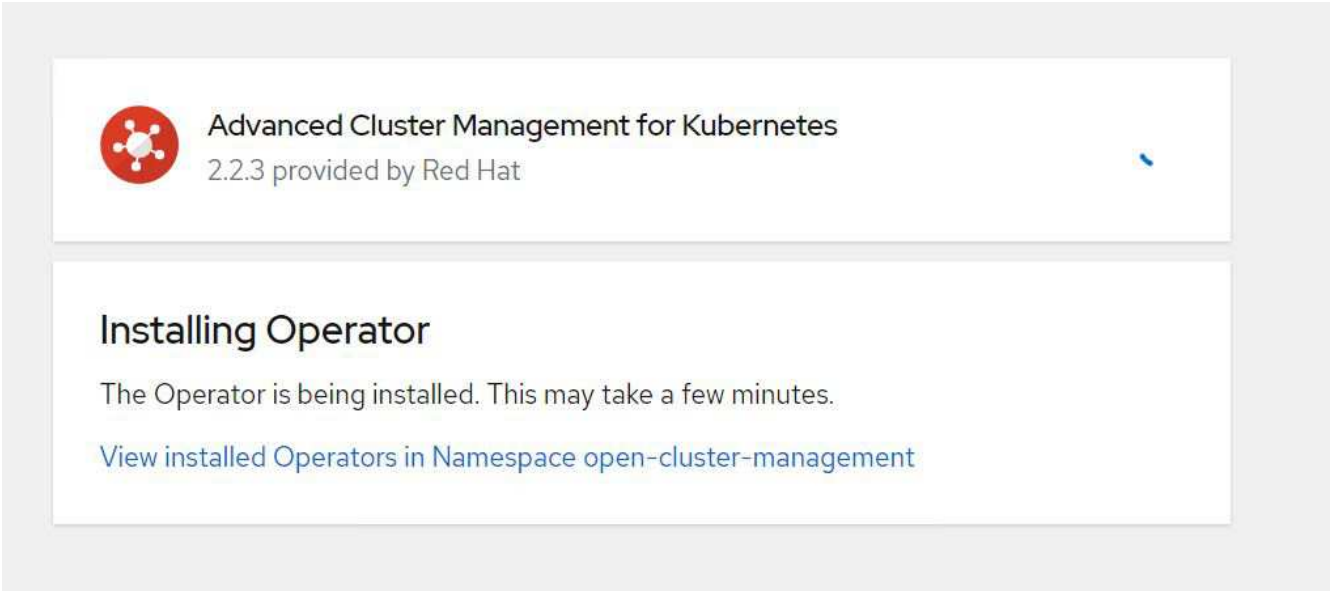


3. 「Advanced Cluster Management for Kubernetes」を選択し、「インストール」をクリックします。

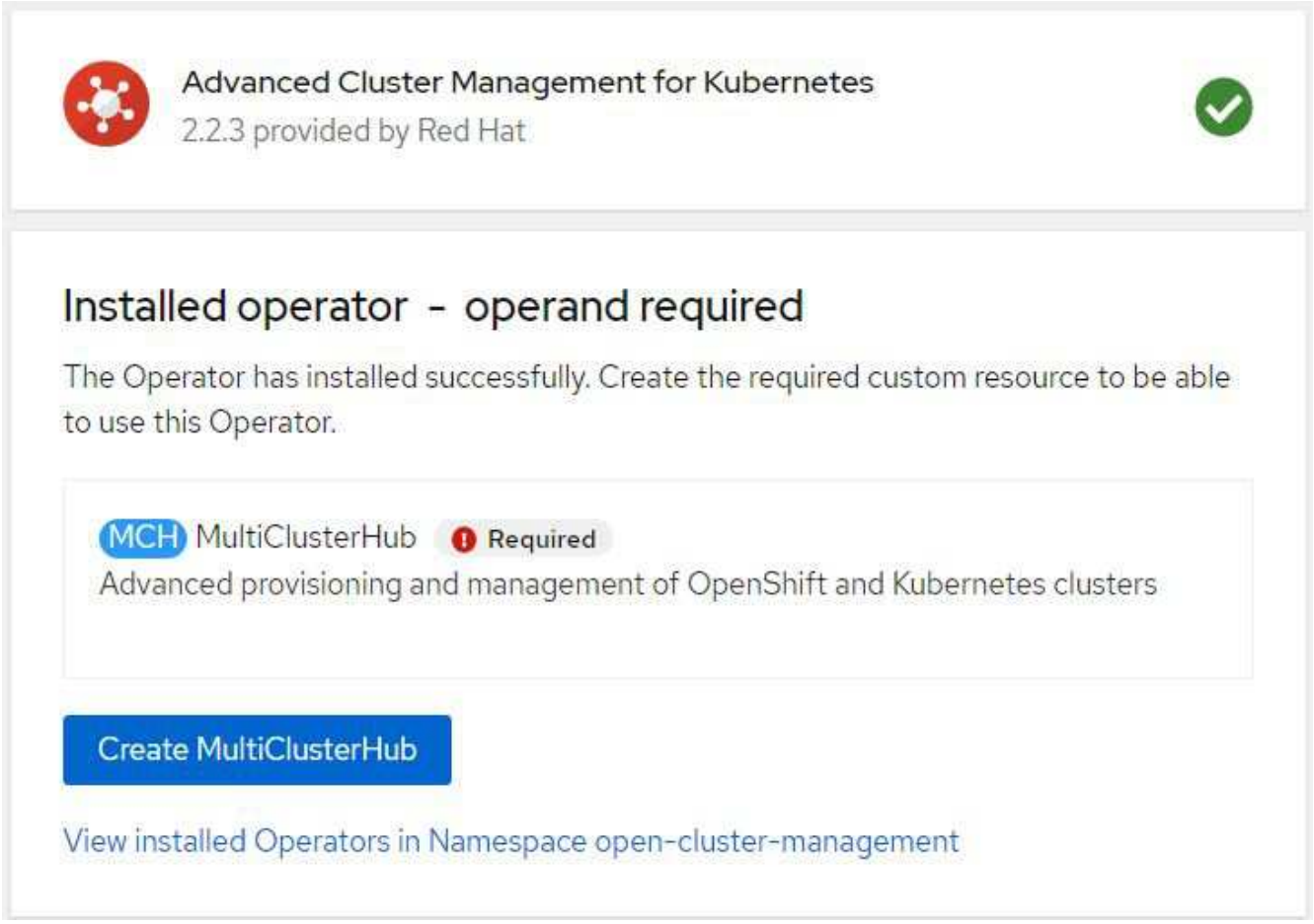
4. [Install Operator] 画面で、必要な詳細 (NetAppデフォルトのパラメータを保持することを推奨) を入力し、[Install] をクリックします。



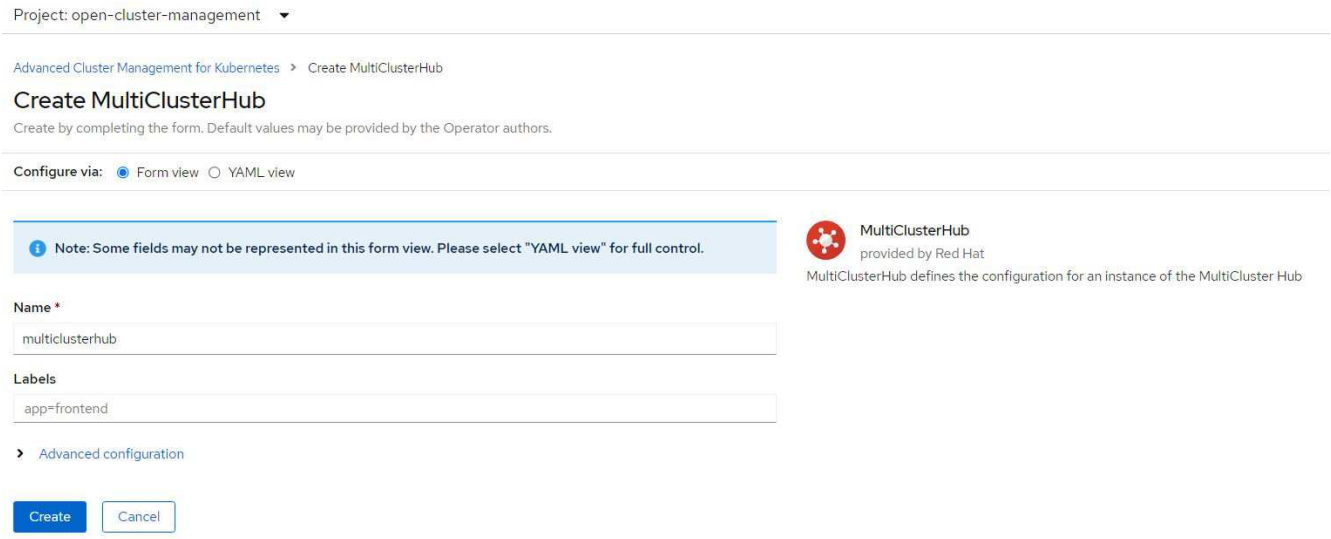
5. オペレータのインストールが完了するまで待ちます。



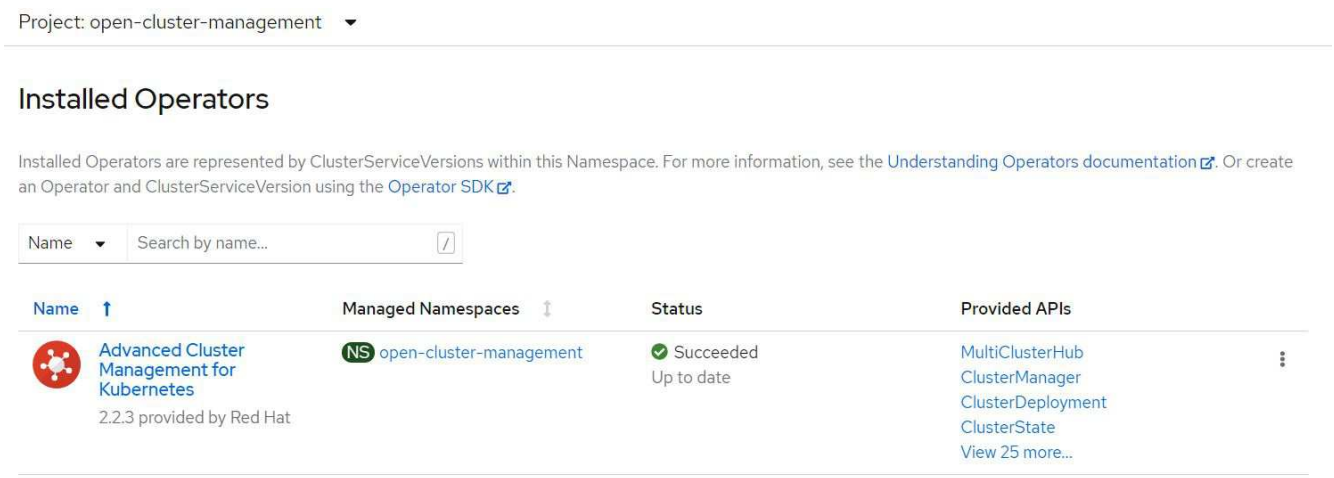
6. オペレーターがインストールされたら、「MultiClusterHub の作成」をクリックします。



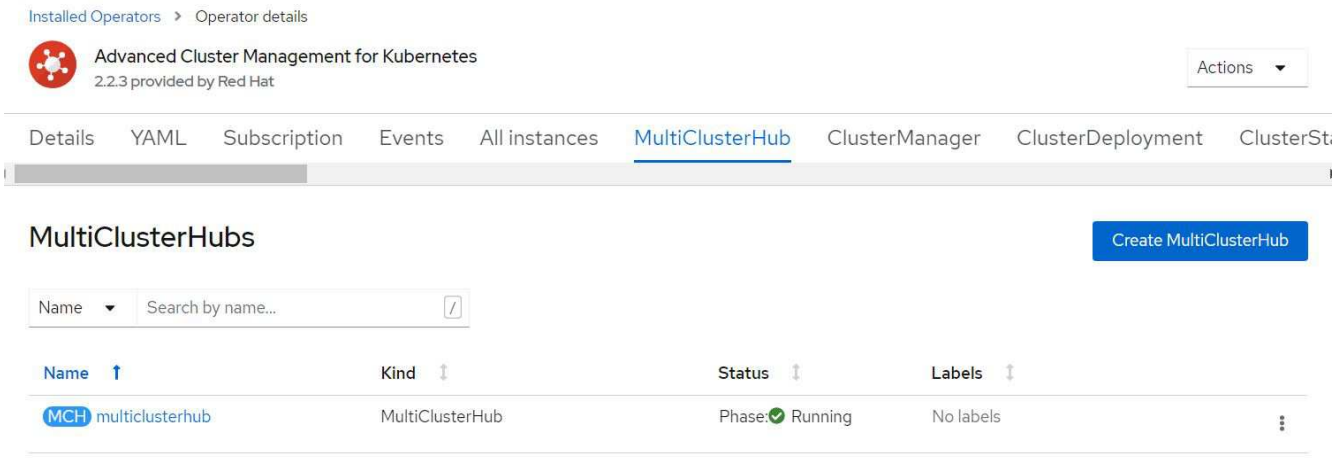
7. 「MultiClusterHub の作成」画面で、詳細を入力した後、「作成」をクリックします。これにより、マルチクラスター ハブのインストールが開始されます。



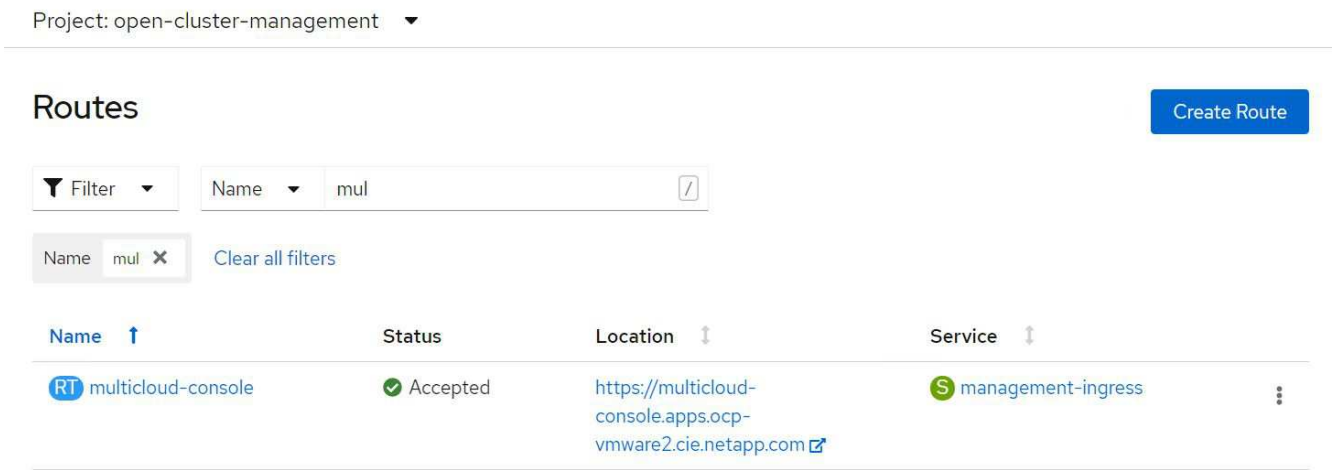
8. すべてのポッドが open-cluster-management 名前空間で実行状態に移行し、オペレーターが成功状態に移行すると、Advanced Cluster Management for Kubernetes がインストールされます。



9. ハブのインストールが完了するまでには少し時間がかかり、完了すると、MultiCluster ハブは実行状態に移行します。

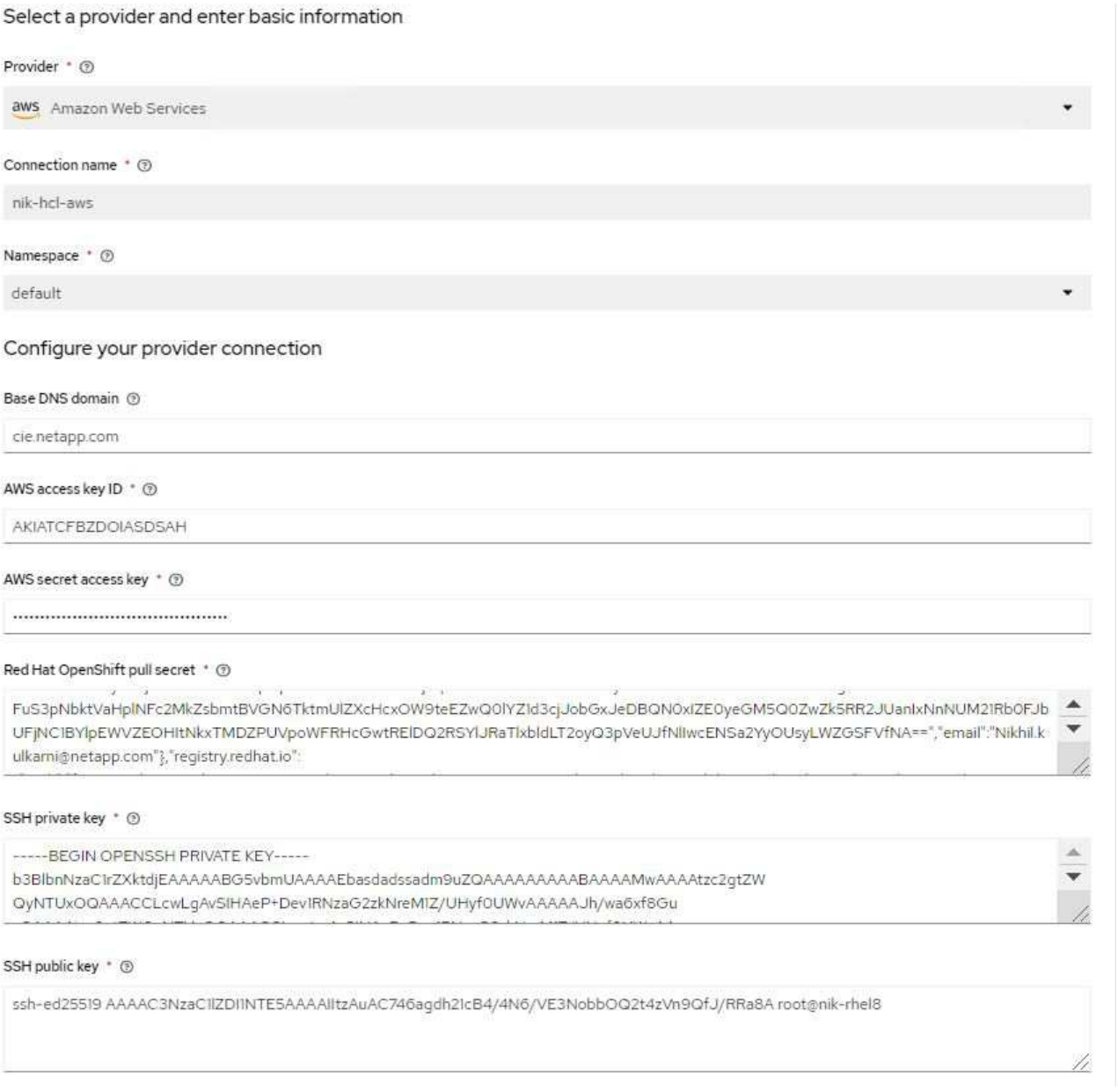


10. open-cluster-management 名前空間にルートを作成します。ルート内の URL に接続して、Advanced Cluster Management コンソールにアクセスします。

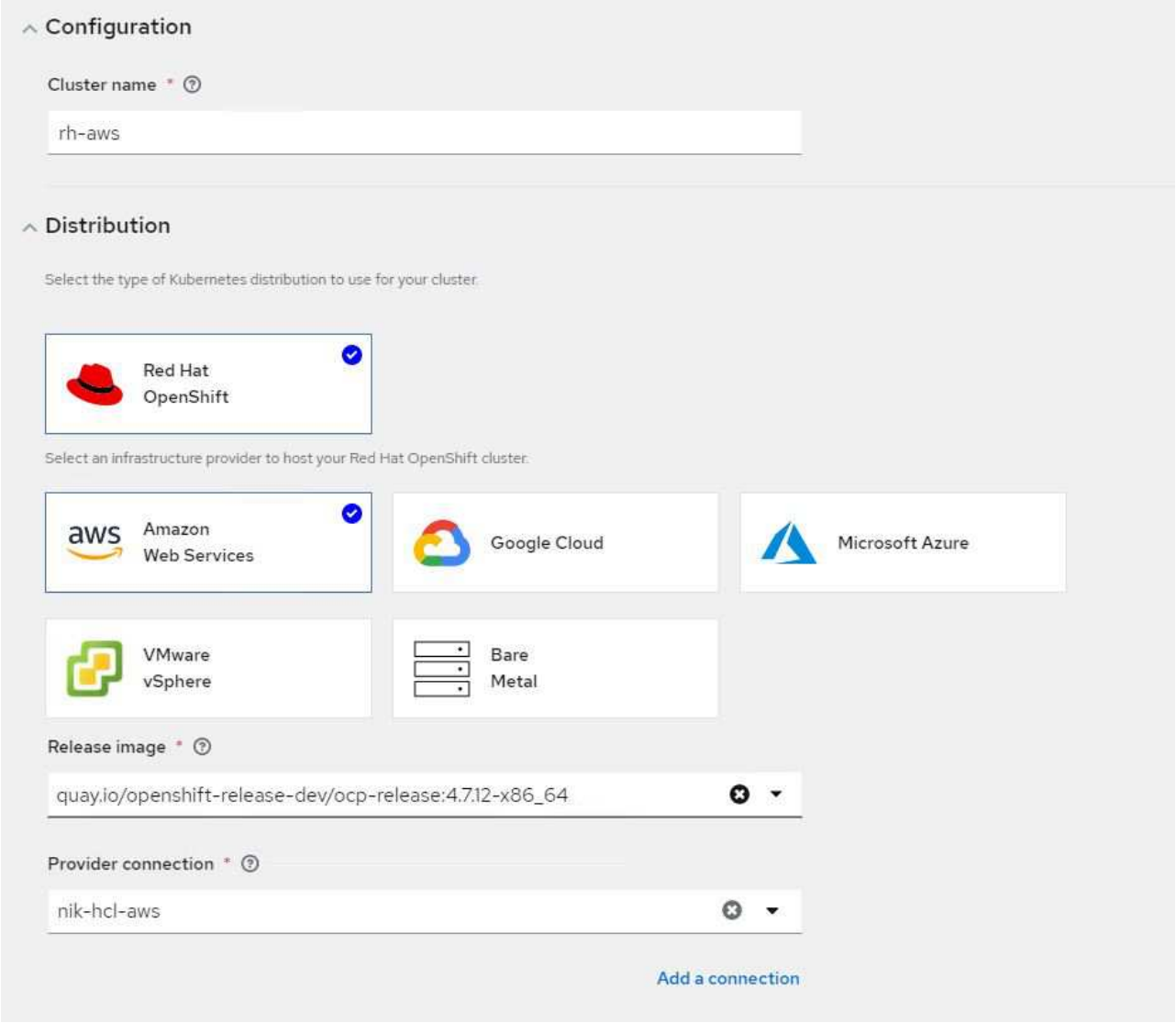


さまざまな OpenShift クラスターを管理するには、クラスターを作成するか、Advanced Cluster Management にインポートします。

1. まず、「インフラストラクチャの自動化」>「クラスター」に移動します。
2. 新しい OpenShift クラスターを作成するには、次の手順を実行します。
   a. プロバイダー接続を作成します。「プロバイダー接続」に移動して「接続の追加」をクリックし、選択したプロバイダー タイプに対応するすべての詳細を入力して「追加」をクリックします。



   b. 新しいクラスターを作成するには、「クラスター」に移動し、「クラスターの追加」>「クラスターの作成」をクリックします。クラスターと対応するプロバイダーの詳細を入力し、「作成」をクリックします。



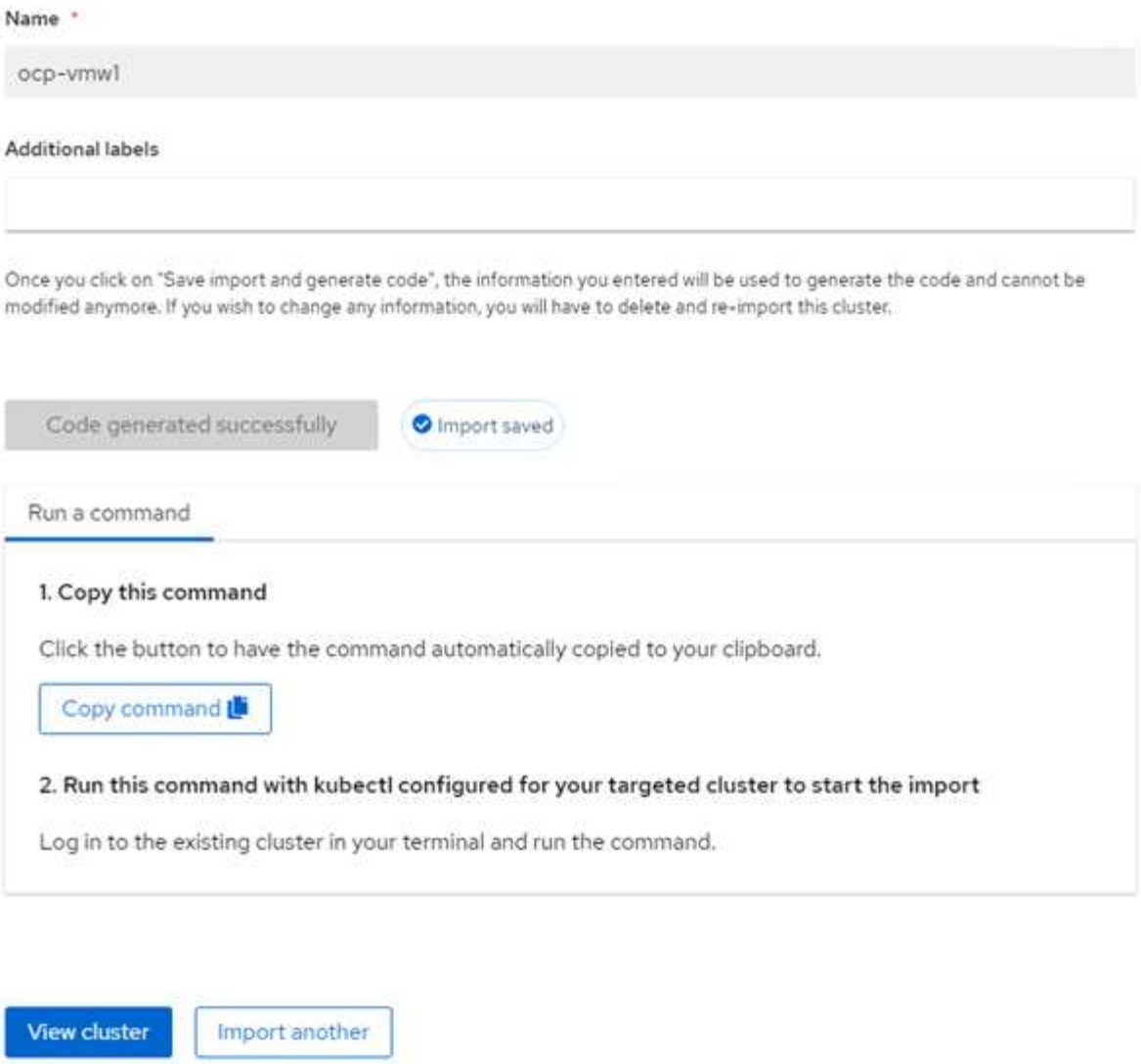
c. クラスターが作成されると、そのクラスターは「準備完了」ステータスでクラスター リストに表示されます。

3. 既存のクラスターをインポートするには、次の手順を実行します。

a. [クラスター] に移動し、[クラスターの追加] > [既存のクラスターのインポート] をクリックします。

b. クラスターの名前を入力し、「インポートを保存してコードの生成」をクリックします。既存のクラスターを追加するコマンドが表示されます。

c. [コマンドのコピー] をクリックし、ハブ クラスターに追加するクラスターでコマンドを実行します。これにより、クラスター上に必要なエージェントのインストールが開始され、このプロセスが完了すると、クラスターは「準備完了」のステータスでクラスター リストに表示されます。

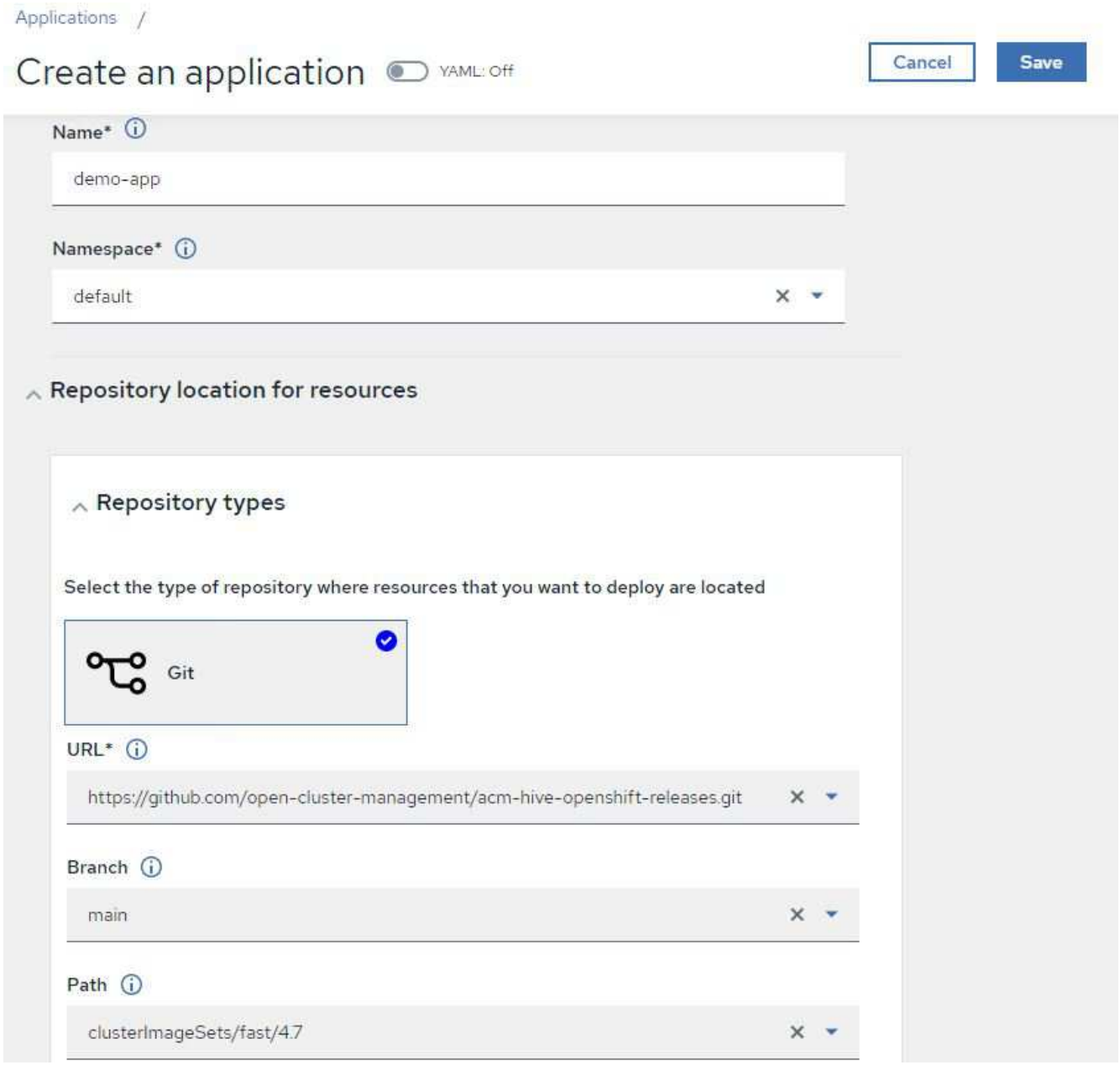


4. 複数のクラスターを作成してインポートすると、単一のコンソールからそれらを監視および管理できます。

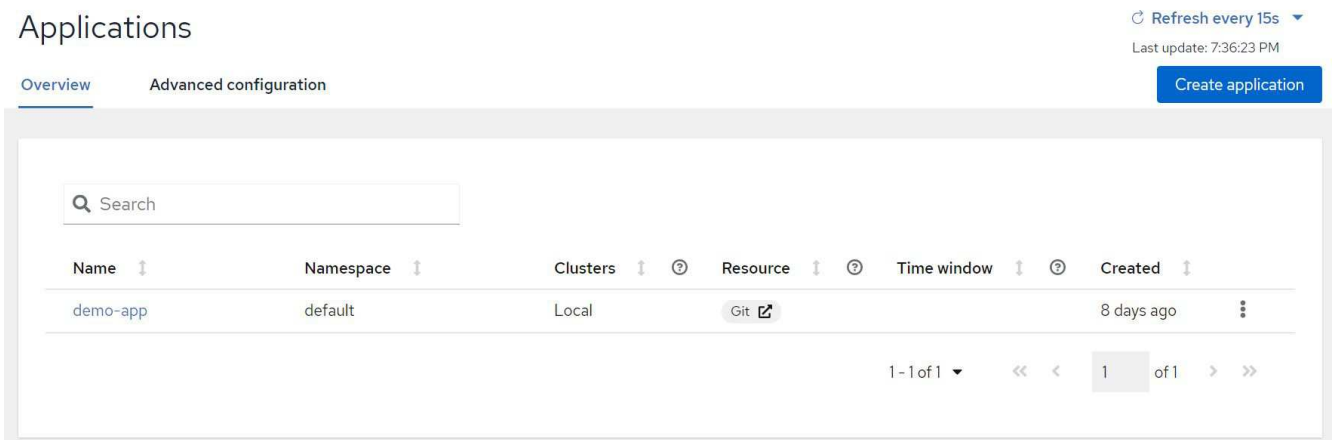
アプリケーションライフサイクル管理

アプリケーションを作成し、それを複数のクラスタにわたって管理するには、

1. サイドバーから「アプリケーションの管理」に移動し、「アプリケーションの作成」をクリックします。作成するアプリケーションの詳細を入力し、「保存」をクリックします。



2. アプリケーション コンポーネントがインストールされると、アプリケーションがリストに表示されます。

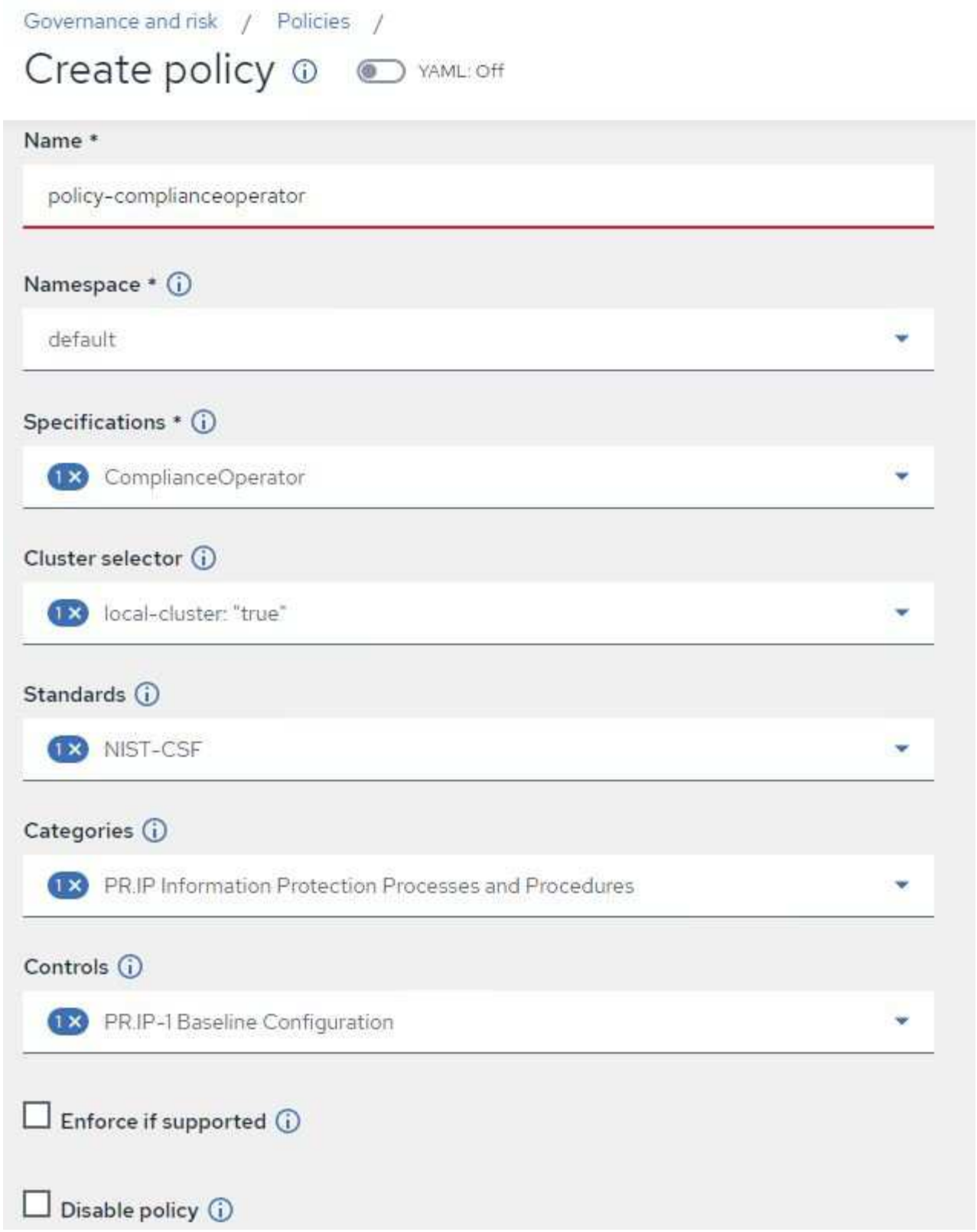


3. これで、コンソールからアプリケーションを監視および管理できるようになりました。

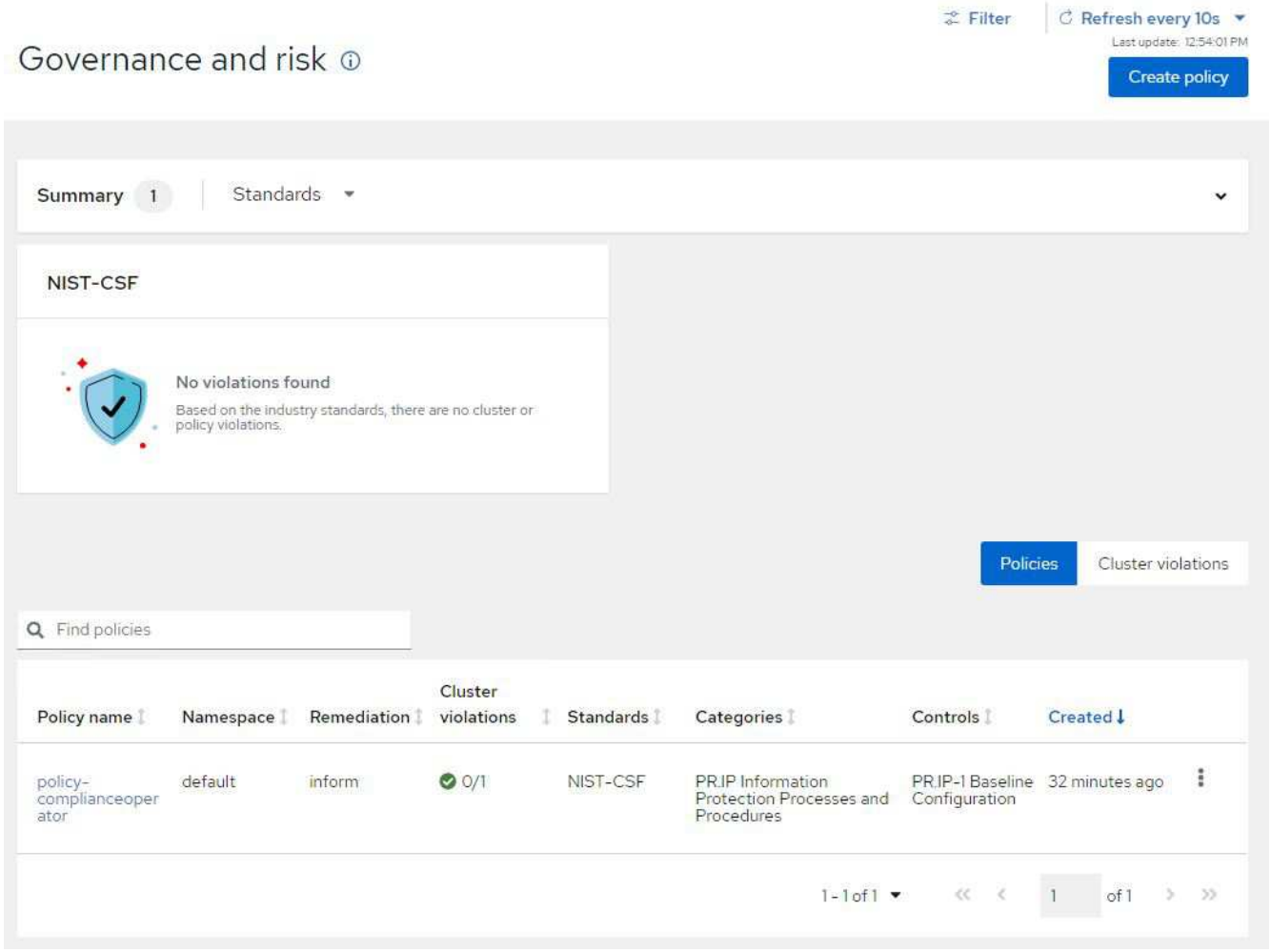
ガバナンスとリスク

この機能を使用すると、さまざまなクラスターのコンプライアンス ポリシーを定義し、クラスターがそれに準拠していることを確認できます。ルールからの逸脱や違反を通知したり、修正したりするようにポリシーを構成できます。

1. サイドバーからガバナンスとリスクに移動します。
2. コンプライアンス ポリシーを作成するには、[ポリシーの作成] をクリックし、ポリシー標準の詳細を入力して、このポリシーに準拠する必要があるクラスターを選択します。このポリシー違反を自動的に修正する場合は、「サポートされている場合は強制する」チェックボックスをオンにして、「作成」をクリックします。



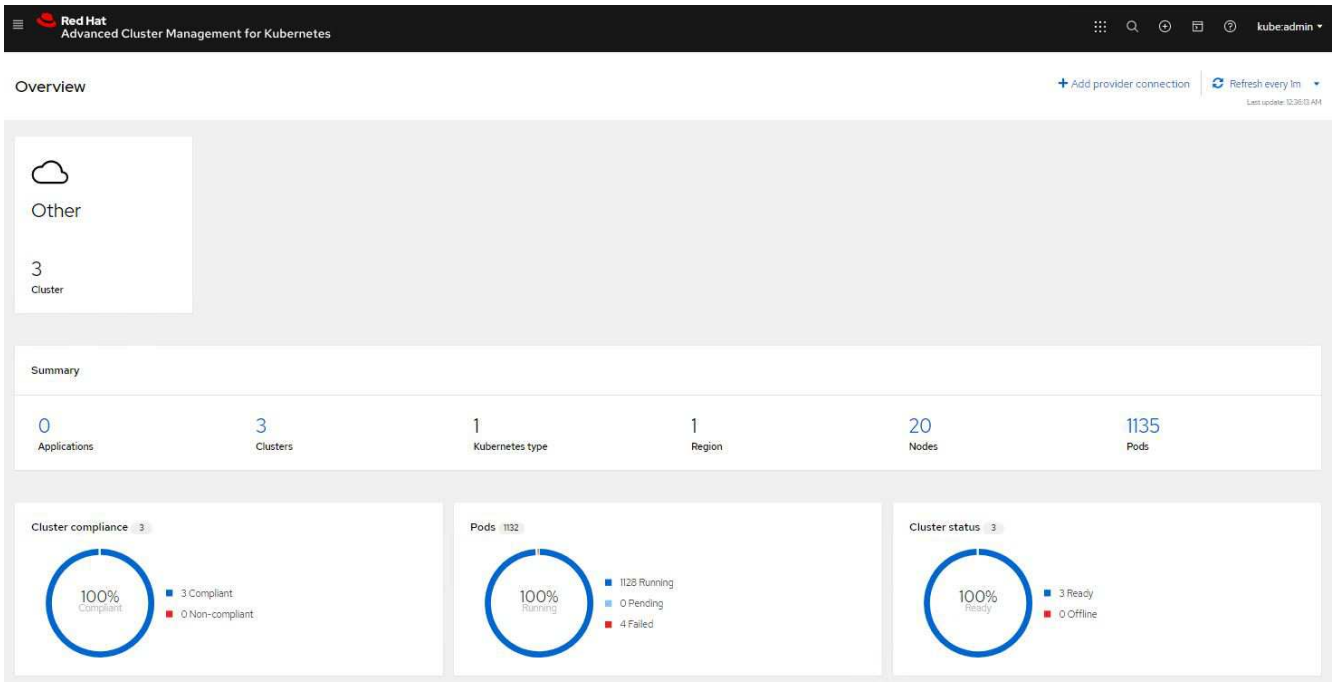
3. 必要なポリシーがすべて構成されたら、Advanced Cluster Management からポリシー違反またはクラスター違反を監視して修復できます。



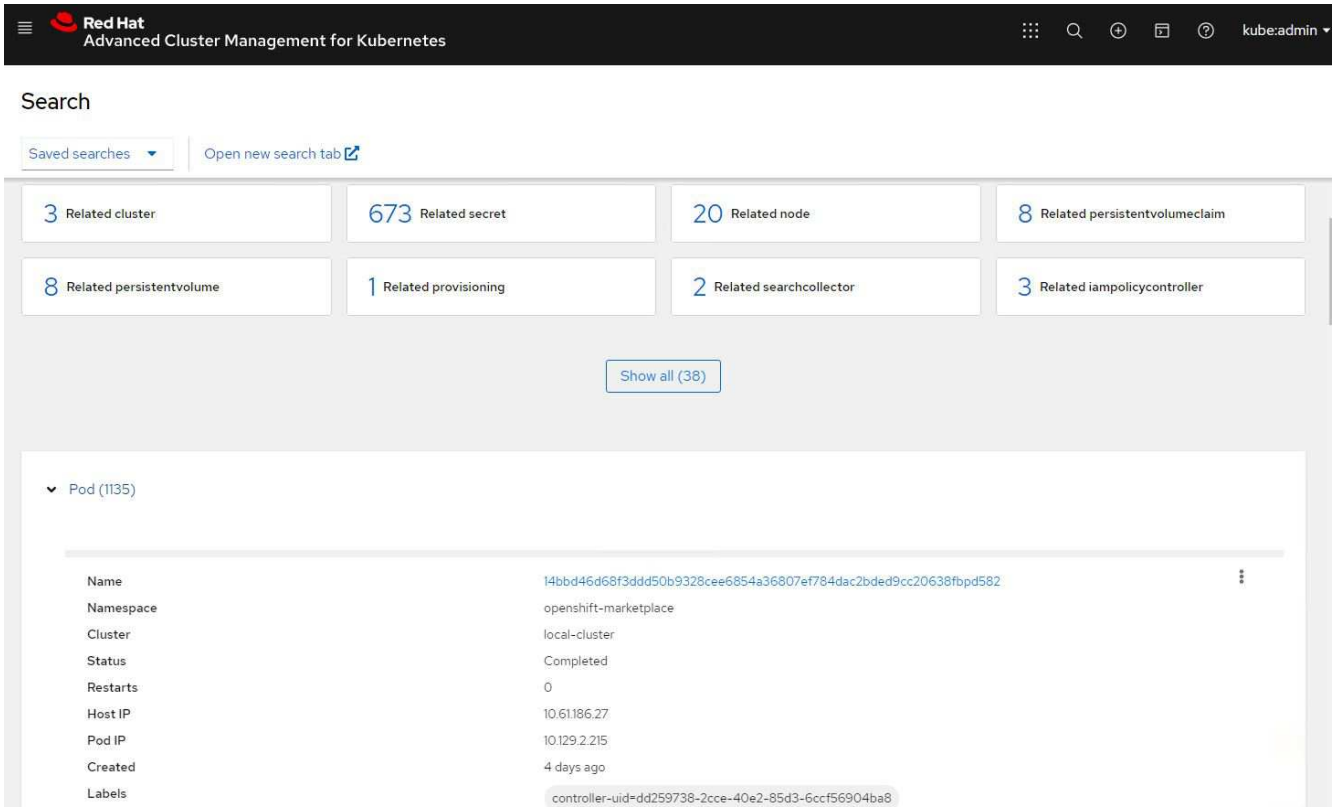
可観測性

Kubernetes の高度なクラスター管理は、すべてのクラスターにわたってノード、ポッド、アプリケーション、およびワークロードを監視する方法を提供します。

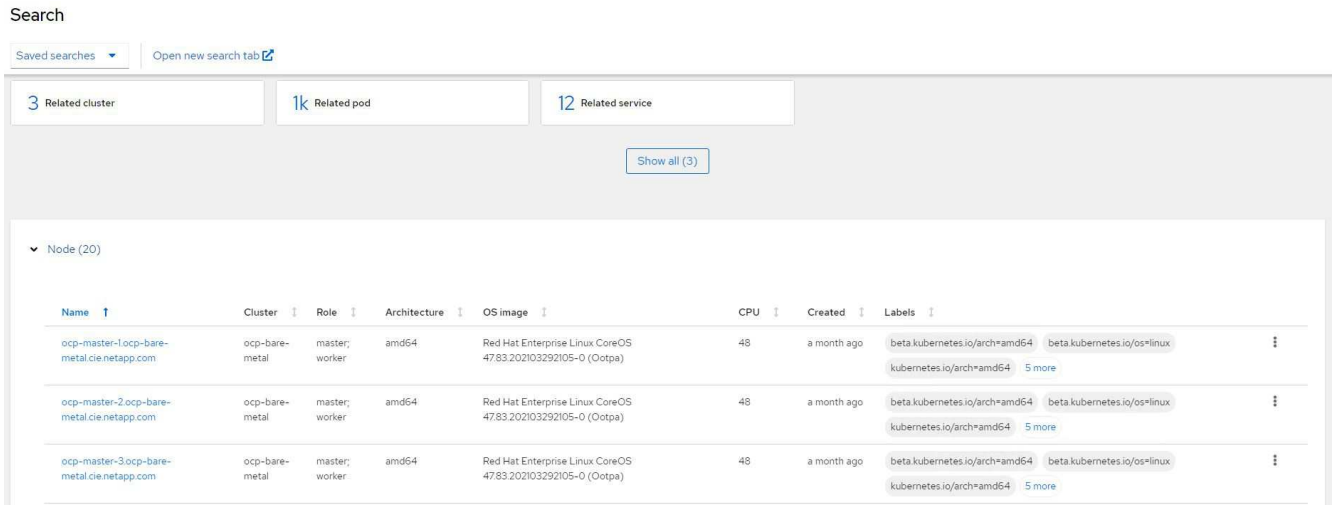
1. [環境の観察] > [概要] に移動します。



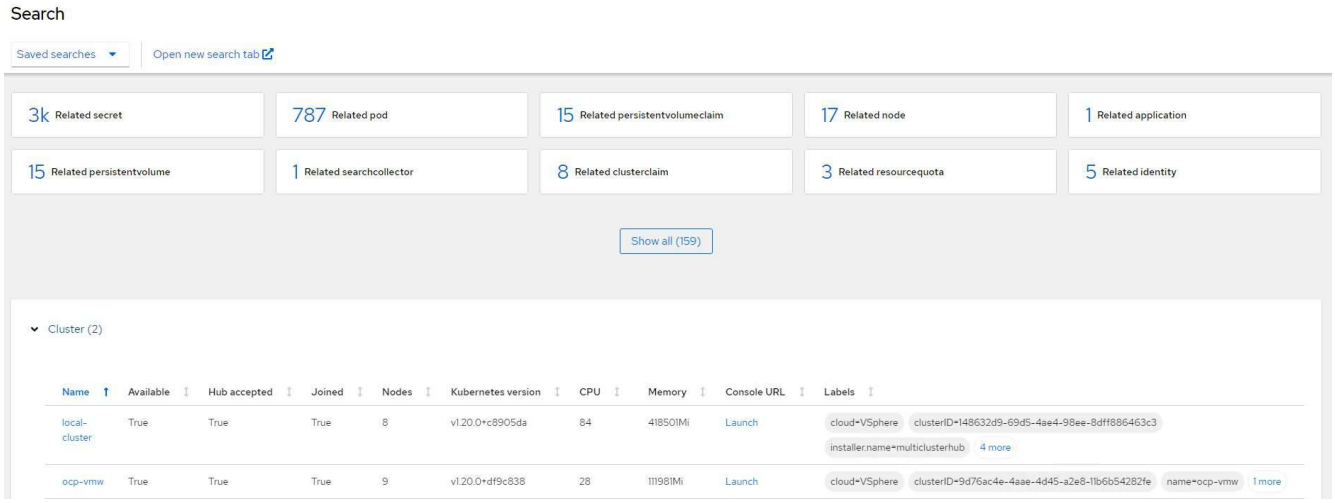
2. すべてのクラスターのすべてのポッドとワークロードは、さまざまなフィルターに基づいて監視および並べ替えられます。対応するデータを表示するには、ポッドをクリックします。



3. クラスター全体のすべてのノードは、さまざまなデータ ポイントに基づいて監視および分析されます。ノードをクリックすると、対応する詳細情報が表示されます。

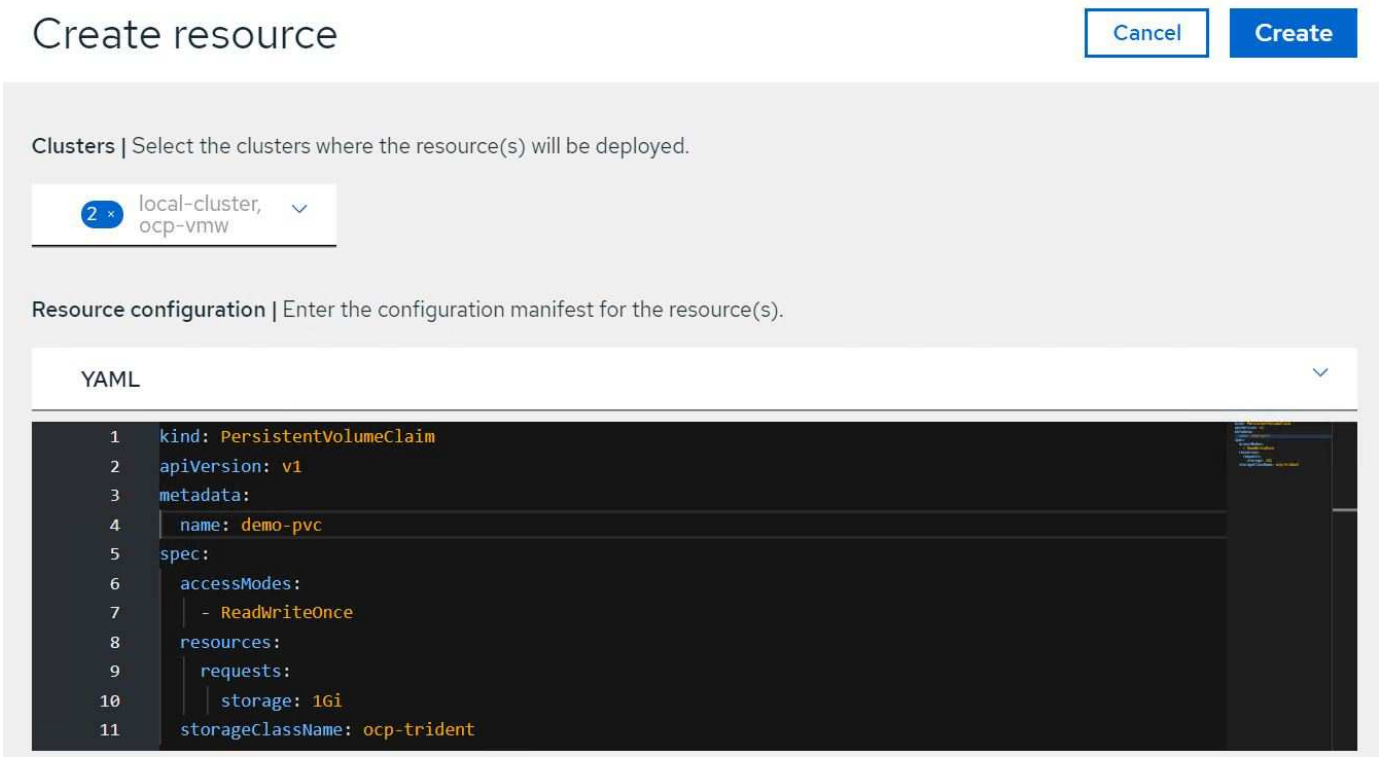


4. すべてのクラスターは、さまざまなクラスター リソースとパラメータに基づいて監視および整理されます。クラスターの詳細を表示するには、「クラスター」をクリックします。



複数のクラスターにリソースを作成する

Kubernetes の高度なクラスター管理を使用すると、ユーザーはコンソールから 1 つ以上の管理対象クラスター上に同時にリソースを作成できます。たとえば、異なるNetApp ONTAPクラスターでバックアップされた異なるサイトの OpenShift クラスターがあり、両方のサイトで PVC をプロビジョニングする場合は、上部のバーの (+) 記号をクリックします。次に、PVC を作成するクラスターを選択し、リソース YAML を貼り付けて、「作成」をクリックします。



Trident Protectを使用したコンテナアプリとVMのデータ保護

このソリューションでは、 Trident Protect を使用してコンテナーと VM のデータ保護操作を実行する方法を示します。

1. OpenShift Container Platform のコンテナアプリケーションのスナップショットとバックアップの作成と復元の詳細については、以下を参照してください。"ここをクリックしてください。" 。
2. OpenShift Container プラットフォームにデプロイされた OpenShift Virtualization の VM のバックアップの作成と復元の詳細については、以下を参照してください。"ここをクリックしてください。" 。

サードパーティツールを使用したコンテナアプリとVMのデータ保護

このソリューションでは、Red Hat OpenShift Container Platform の OADP オペレーターと統合された Velero を使用して、コンテナーと VM のデータ保護操作を実行する方法を示します。

1. OpenShift Container Platform のコンテナアプリケーションのバックアップの作成と復元の詳細については、以下を参照してください。"ここをクリックしてください。" 。
2. OpenShift Container プラットフォームにデプロイされた OpenShift Virtualization の VM のバックアップの作成と復元の詳細については、以下を参照してください。"ここをクリックしてください。" 。

Red Hat OpenShift VirtualizationとNetAppストレージの統合について学ぶための追加リソース

さまざまなプラットフォームとテクノロジーにわたるONTAPを使用した Red Hat OpenShift Virtualization の導入、管理、最適化のサポートに関する詳細情報を提供する追加リソースにアクセスします。

- NetAppのマニュアル

  "https://docs.netapp.com/"

- Tridentドキュメント

  "https://docs.netapp.com/us-en/trident/index.html"

- Red Hat OpenShift ドキュメント

  "https://access.redhat.com/documentation/en-us/openshift_container_platform/4.7/"

- Red Hat OpenStack Platform ドキュメント

  "https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/16.1/"

- Red Hat Virtualization ドキュメント

  "https://access.redhat.com/documentation/en-us/red_hat_virtualization/4.4/"

- VMware vSphereのドキュメント

  "https://docs.vmware.com/"

著作権に関する情報

商標に関する情報