ネットアップを利用した **Red Hat OpenShift**

NetApp Solutions

NetApp
April 10, 2024

目次

NVA-1160 ：ネットアップでの Red Hat OpenShift

ネットアップ、 Alan Cowles 氏と Nikhil M Kulkarni 氏

このリファレンスドキュメントでは、ネットアップによって検証済みの複数の異なるデータセンター環境に Installer Provisioned Infrastructure （ IPI ）を通じて導入された Red Hat OpenShift 解決策の導入を検証します。また、ネットアップストレージシステムとの統合についても、 Astra Trident ストレージオーケストレーションツールを使用して永続的ストレージを管理することで詳しく説明します。最後に、解決策検証と実際の使用事例をいくつか確認して文書化します。

ユースケース

ネットアップ解決策を使用した Red Hat OpenShift は、次のユースケースでお客様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 （インストーラでプロビジョニングされたインフラ）を使用して導入した Red Hat OpenShift の導入と管理が容易です。
- OSP 、 RHV 、 vSphere 、または OpenShift Virtualization を使用したベアメタルに導入された Red Hat OpenShift を使用して、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組み合わせたもの。
- ネットアップストレージと Kubernetes 向けオープンソースストレージオーケストレーションツールである Astra Trident とともに使用される Red Hat OpenShift の機能を紹介する実際の構成とユースケース。

ビジネスバリュー

企業は、新しい製品の作成、リリースサイクルの短縮、新機能の迅速な追加を目的として、 DevOps の手法を採用する傾向に迫られています。即応性に優れた本来の性質から、コンテナやマイクロサービスは、 DevOps の実践を支援するうえで重要な役割を果たします。しかし、エンタープライズ環境で本番環境規模で DevOps を実践する場合、独自の課題が生じ、基盤となるインフラに次のような一定の要件が課せられます。

- スタック内のすべてのレイヤで高可用性を実現します
- 導入手順の簡易化
- ノンストップオペレーションとアップグレード
- API ベースのプログラム可能なインフラで、マイクロサービスの即応性を維持します
- パフォーマンスを保証するマルチテナンシー
- 仮想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できます
- ワークロードのニーズに応じてインフラを個別に拡張できる

ネットアップとともに Red Hat OpenShift を導入することで、これらの課題に対応し、お客様が選択したデータセンター環境に RedHat OpenShift IPI を完全に自動で導入できるようになり、それぞれの問題に対処できる解決策が提供されます。

テクノロジの概要

NetApp 解決策を使用した Red Hat OpenShift は、次の主要コンポーネントで構成されています。

Red Hat OpenShift Container Platform

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向け Kubernetes プラットフォームです。Red Hat は、オープンソースの Kubernetes をいくつか強化して、コンテナ化されたアプリケーションの構築、導入、管理を完全に統合したすべてのコンポーネントを備え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 サイトを参照してください "こちらをご覧ください"。

ネットアップストレージシステム

ネットアップには、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環境に最適なストレージシステムが複数あります。ネットアップのポートフォリオには、コンテナ化されたアプリケーションに永続的ストレージを提供できる NetApp ONTAP 、 NetApp Element 、および NetApp E シリーズストレージシステムが含まれています。

詳細については、ネットアップの Web サイトをご覧ください "こちらをご覧ください"。

ネットアップとストレージの統合

NetApp Astra Control Center は、信頼性の高いネットアップのデータ保護テクノロジを基盤とするオンプレミス環境に導入された、ステートフル Kubernetes ワークロード向けの充実したストレージおよびアプリケーション対応のデータ管理サービスを提供します。

詳細については、 NetApp Astra の Web サイトをご覧ください "こちらをご覧ください"。

Astra Trident は、コンテナや Kubernetes ディストリビューション向けの、 Red Hat OpenShift など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ションツールです。

詳細については、 Astra Trident の Web サイトをご覧ください "こちらをご覧ください"。

Advanced Configuration Options （詳細設定オプション）

このセクションは、実環境のユーザがこの解決策を本番環境に導入するときに実行する必要があるカスタマイズ（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導入など）に特化したものです。

検証済みリリースの現在のサポートマトリックスです

テクノロジ	目的	ソフトウェアのバージョン
NetApp ONTAP	ストレージ	9.8 、 9.9.1
NetApp Element	ストレージ	12.3
ネットアップアストラコントロールセンター	アプリケーション対応データ管理	21.12.60
ネットアップアストラト Trident	ストレージオーケストレーション	22.01.0
Red Hat OpenShift のサービスです	コンテナオーケストレーション	4.6 EUS 、 4.7 、 4.8
Red Hat OpenStack Platform	プライベートクラウドインフラ	16.1

Red Hat 仮想化	データセンターの仮想化	4 月 4 日
VMware vSphere の場合	データセンターの仮想化	6.7U3

OpenShift の概要

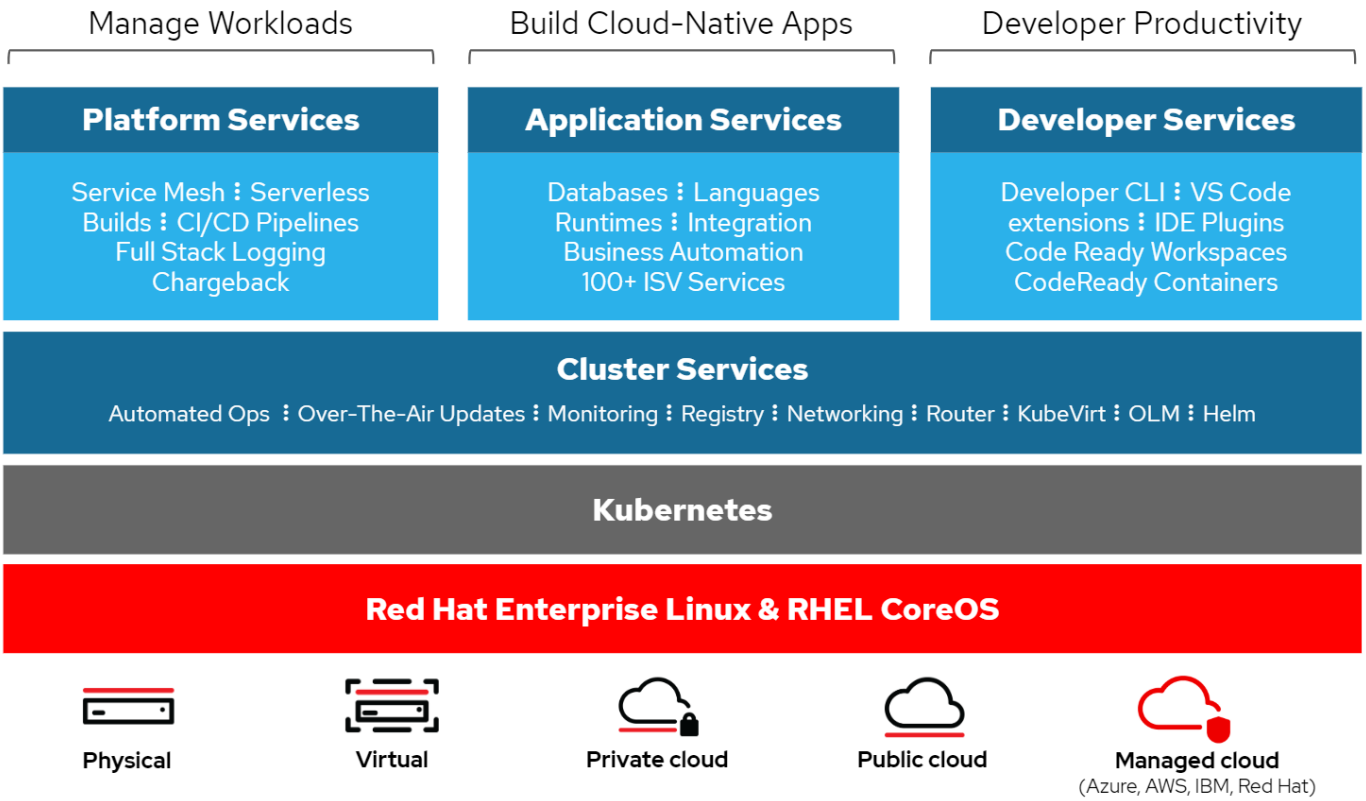
Red Hat OpenShift Container Platform は、開発と IT の運用を単一のプラットフォーム上に統合し、オンプレミスとハイブリッドクラウドのインフラ全体でアプリケーションを一貫して構築、導入、管理します。Red Hat OpenShift は、コンテナベースのワークロード向けに設計された、世界をリードするエンタープライズ Linux ディストリビューションである Kubernetes や Red Hat Enterprise Linux CoreOS など、オープンソースのイノベーションと業界標準に基づいて構築されています。OpenShift は Cloud Native 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 には次の機能があります。

- *セルフサービスプロビジョニング*開発者は、最も頻繁に使用するツールを使用して、必要に応じてアプリケーションをすばやく簡単に作成できます。また、運用担当者は、環境全体を完全に制御できます。
- 永続的ストレージ OpenShift Container Platformは、永続的ストレージをサポートすることで、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。
- *継続的統合および継続的開発（CI/CD）*このソースコードプラットフォームは、大規模なビルドおよび展開イメージを管理します。
- *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に加えて、コンテナオーケストレーションのためのOpen Container Initiative（OCI）とKubernetesが組み込まれています。お客様は、特定のベンダーのテクノロジ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、CI/CDパイプラインを即座にサポートします。これにより、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または構成に加えられたすべての変更に対して確実に実行できるようになります。
- *ロールベースアクセス制御(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を編成するのに役立ちます。
- ビルドとデプロイの自動化 OpenShiftを使用すると、開発者は、コンテナ化されたアプリケーションをビルドしたり、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プラットフォームにビルドさせることができます。プラットフォームは、アプリケーションに定義された特性に基づいて、これらのアプリケーションのインフラストラクチャへの導入を自動化します。たとえば、割り当てられるリソースの量や、サードパーティのライセンスに準拠するために導入するインフラストラクチャ上の場所などです。
- 一貫した環境 OpenShiftにより、開発者向けにプロビジョニングされた環境が、オペレーティングシステム、ライブラリ、ランタイムバージョン（Javaランタイムなど）、アプリケーションのライフサイクル全体にわたって一貫していることが保証されます。 また、一貫性のない環境に起因するリスクを排除するために使用されているアプリケーションランタイム（Tomcatなど）でも使用されています。
- *構成管理*構成と機密データ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使用されるテクノロジーや環境に依存しない一貫した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。
  導入済み：
- *アプリケーションのログとメトリック。*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 に統合された監視機能とログ管理機能により、開発者はアプリケーションがどのように変化しても動作しているかを調査し、アプリケーションのライフサイクルの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はマルチテナンシーを提供し、Security-Enhanced Linux（SELinux）、cgroups、Secure Computing Mode（seccomp）で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することで、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用の TLS 証明書による暗号化、およびエンドユーザーに認証済みの信頼できるセキュアなアプリケーションコンテナを提供するためにセキュリティを重視してスキャンおよび採点される Red Hat 認定コンテナ（ access.redhat.com/containers ）へのアクセスも提供します。



Red Hat OpenShift の導入方法

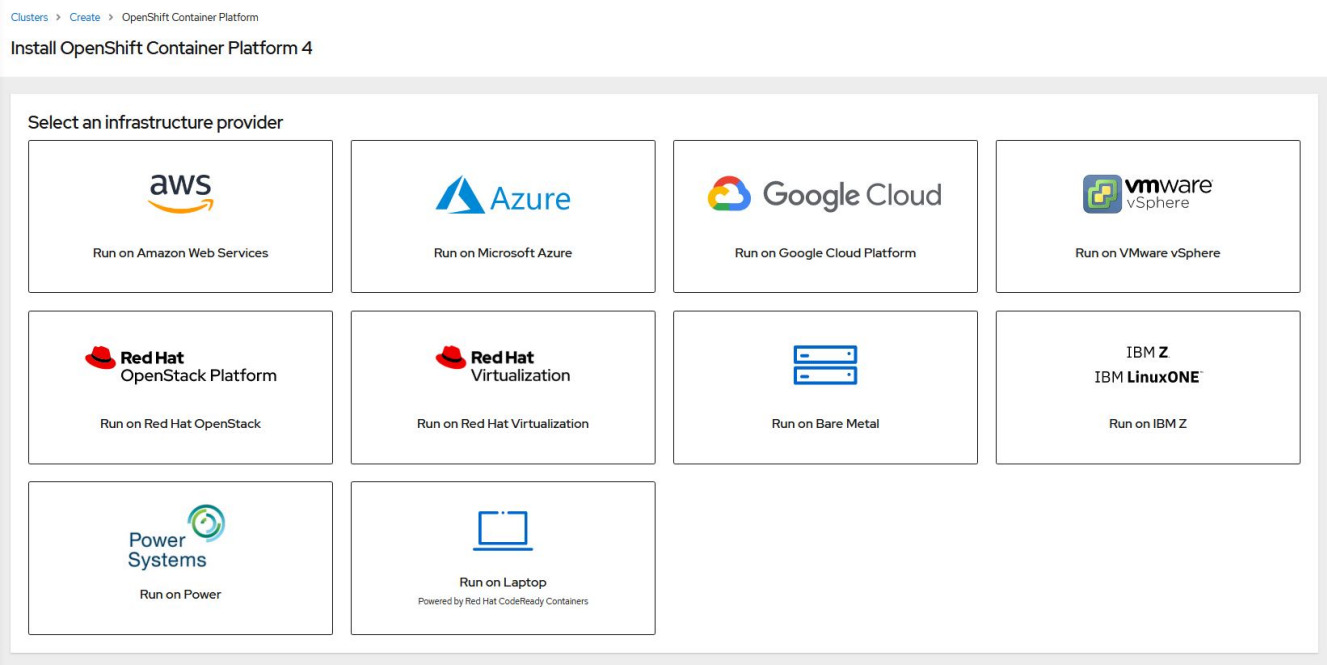
Red Hat OpenShift 4 以降、 OpenShift の導入方法には、高度にカスタマイズされた導入に User Provisioned Infrastructure （ UPI ；ユーザプロビジョニングインフラ）を使用する手動導入、または Installer Provisioned Infrastructure （ IPI ）を使用した完全に自動化された導入が含まれます。

開発、テスト、本番環境向けにOpenShiftクラスタを迅速に導入できるため、ほとんどの場合、IPIのインストール方法が推奨されます。

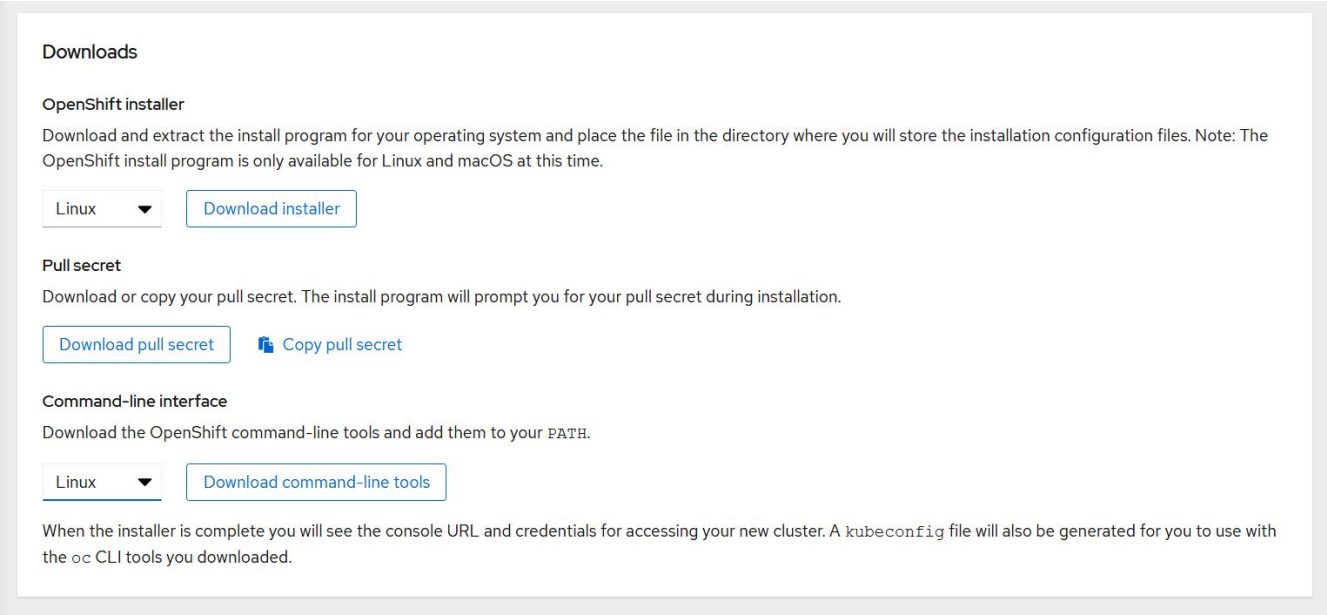
Red Hat OpenShift の IPI インストール

OpenShift の Installer Provisioned Infrastructure （ IPI ）導入には、次の高度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 クレデンシャルでログインします。
2. Red Hat OpenShift を導入する環境を選択します。



3. 次の画面で、インストーラ、独自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. に従ってください "インストール手順" Red Hat が提供する、お客様の環境への導入サービスです。

ネットアップが検証済みの **OpenShift** 環境

ネットアップでは、以下の各データセンター環境で Installer Provisioned Infrastructure （ IPI ）導入方法を使用して、 Red Hat OpenShift のラボへの導入をテストし、検証しています。

- "ベアメタルで実装された OpenShift"
- "Red Hat OpenStack Platform 上の OpenShift"
- "Red Hat 仮想化を基盤とした OpenShift"

- "VMware vSphere 上の OpenShift"

ベアメタルで実装された **OpenShift**

ベアメタル上の OpenShift では、コモディティサーバ上に OpenShift Container Platform を自動で導入できます。

ベアメタル上の OpenShift は、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ワークロードをサポートしながら、 OpenShift クラスタの導入、迅速なプロビジョニング、拡張を容易にする OpenShift の仮想導入に似ています。ベアメタルに導入することで、 OpenShift 環境に加えてホストハイパーバイザー環境の管理に必要な追加のオーバーヘッドを必要としません。ベアメタルサーバに直接導入することで、ホストと OpenShift 環境間でリソースを共有する必要がある物理的なオーバーヘッドの制限を軽減できます。

ベアメタル上の **OpenShift** には次の機能があります。

- * IPIまたはAssisted Installer Deployment * Installer Provisioned Infrastructure（IPI）によってベアメタルサーバにOpenShiftクラスタを導入すると、ハイパーバイザーレイヤを管理することなく、汎用性が高く拡張性の高いOpenShift環境を汎用サーバに直接導入できます。
- *コンパクトなクラスタ設計*ハードウェア要件を最小限に抑えるために、OpenShiftをベアメタルで使用すると、OpenShiftコントロールプレーンノードがワーカーノードおよびホストコンテナとしても機能するため、わずか3ノードのクラスタを導入できます。
- * OpenShift仮想化* OpenShiftは、OpenShift仮想化を使用して、コンテナ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の仮想化では、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* 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、Kubeflowなどのアプリケーションを機械学習アプリケーションに導入できます。

ネットワーク設計

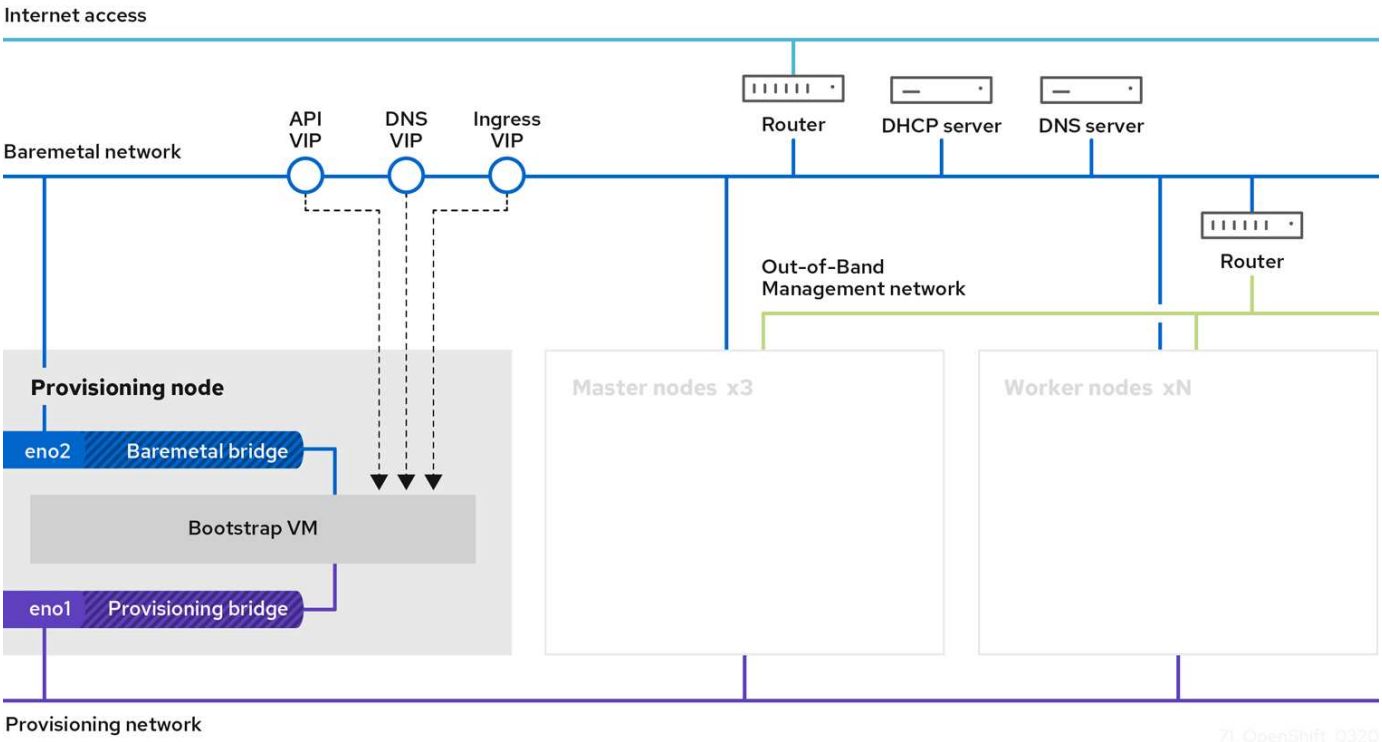
NetApp 解決策上の Red Hat OpenShift では、 2 つのデータスイッチを使用して 25Gbps でプライマリデータ接続を提供します。また、ストレージノードのインバンド管理用に 1Gbps で接続を提供する管理スイッチを 2 台使用し、 IPMI 機能のアウトオブバンド管理も使用します。

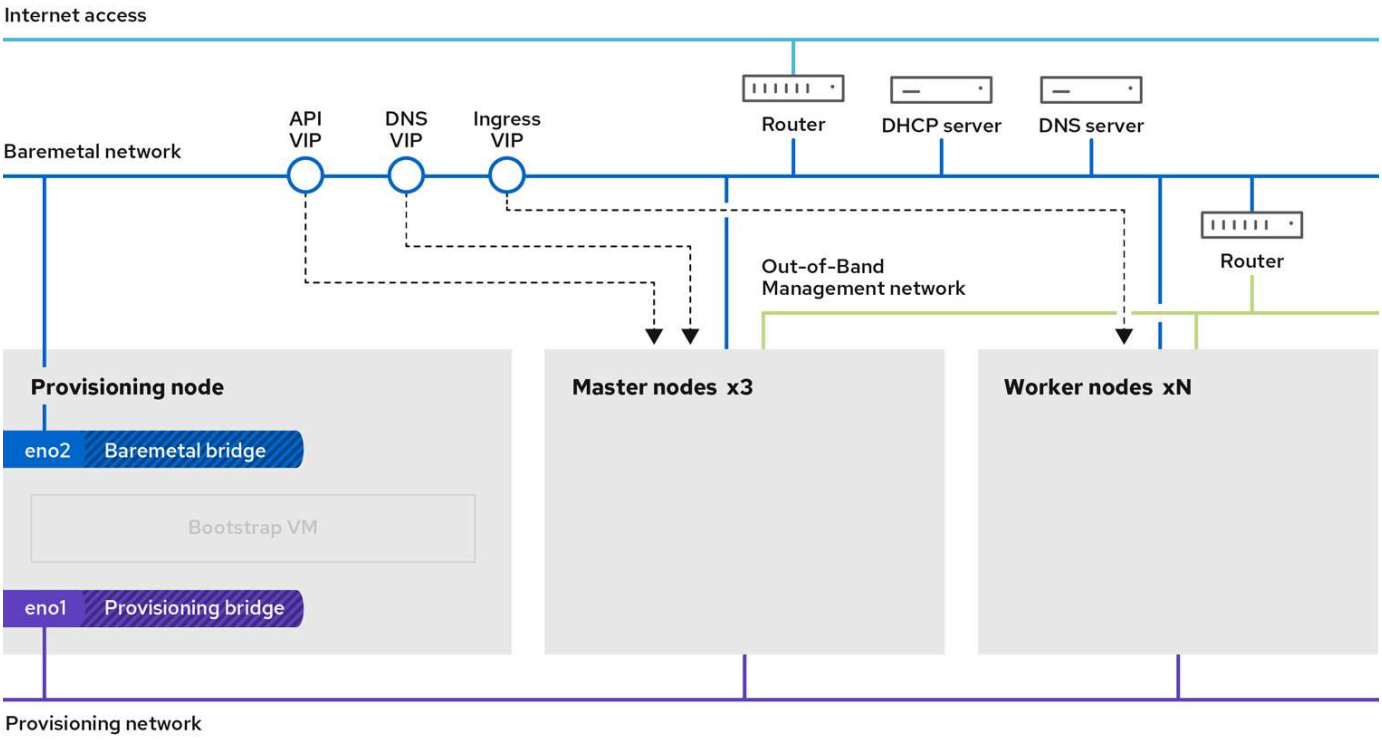
OpenShift ベアメタル IPI 環境では、プロビジョニングノード、つまりネットワークインターフェイスが別々のネットワークに接続されている Red Hat Enterprise Linux 8 マシンを作成する必要があります。

- *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ブートし、OpenShiftクラスタの導入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、クラスタ導入後のパブリック側通信に使用されます。

プロビジョニングノードをセットアップするために、お客様は、トラフィックをノード自体と、導入用にプロビジョニングされたブートストラップ VM に適切にルーティングできるようにするブリッジインターフェイスを作成します。クラスタが導入されると、 API および入力 VIP アドレスがブートストラップノードから新しく導入されたクラスタに移行されます。

次の図は、 IPI の導入時と導入の完了後の環境を示しています。





VLAN の要件

ネットアップ解決策を使用した 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、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
アウトオブバンド管理ネットワーク	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	クラスタが使用可能になると、 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	Network for PXE boot and installation of bare metal nodes （ベアメタルノードの PXE ブートおよびインストール用ネットワーク IPI を使用	3485

これらの各ネットワークは仮想的に VLAN で分離されますが、 PXE ブートシーケンス中に VLAN タグを渡す方法がないため、各物理ポートをプライマリ 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 Container Platform を導入する前に、次のインフラを用意する必要があります。

- 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 DNS サーバが少なくとも 1 台必要です。
- 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できる NTP サーバが少なくとも 1 台必要です。
- （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のアウトバウンドインターネット接続。

Red Hat OpenStack Platform 上の OpenShift

Red Hat OpenStack Platform は、セキュアで信頼性の高いプライベート OpenStack クラウドの構築、導入、拡張を行う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、ネットワークリソースを管理する一連の制御サービスによって実装される IaaS （インフラサービス）クラウドです。この環境の管理には Web ベースのインターフェイスを使用します。このインターフェイスを使用すると、管理者とユーザは OpenStack リソースの制御、プロビジョニング、自動化を行うことができます。さらに、 OpenStack インフラは、広範なコマンドラインインターフェイスと API を通じて管理者とエンドユーザにフルオートメーション機能を提供します。

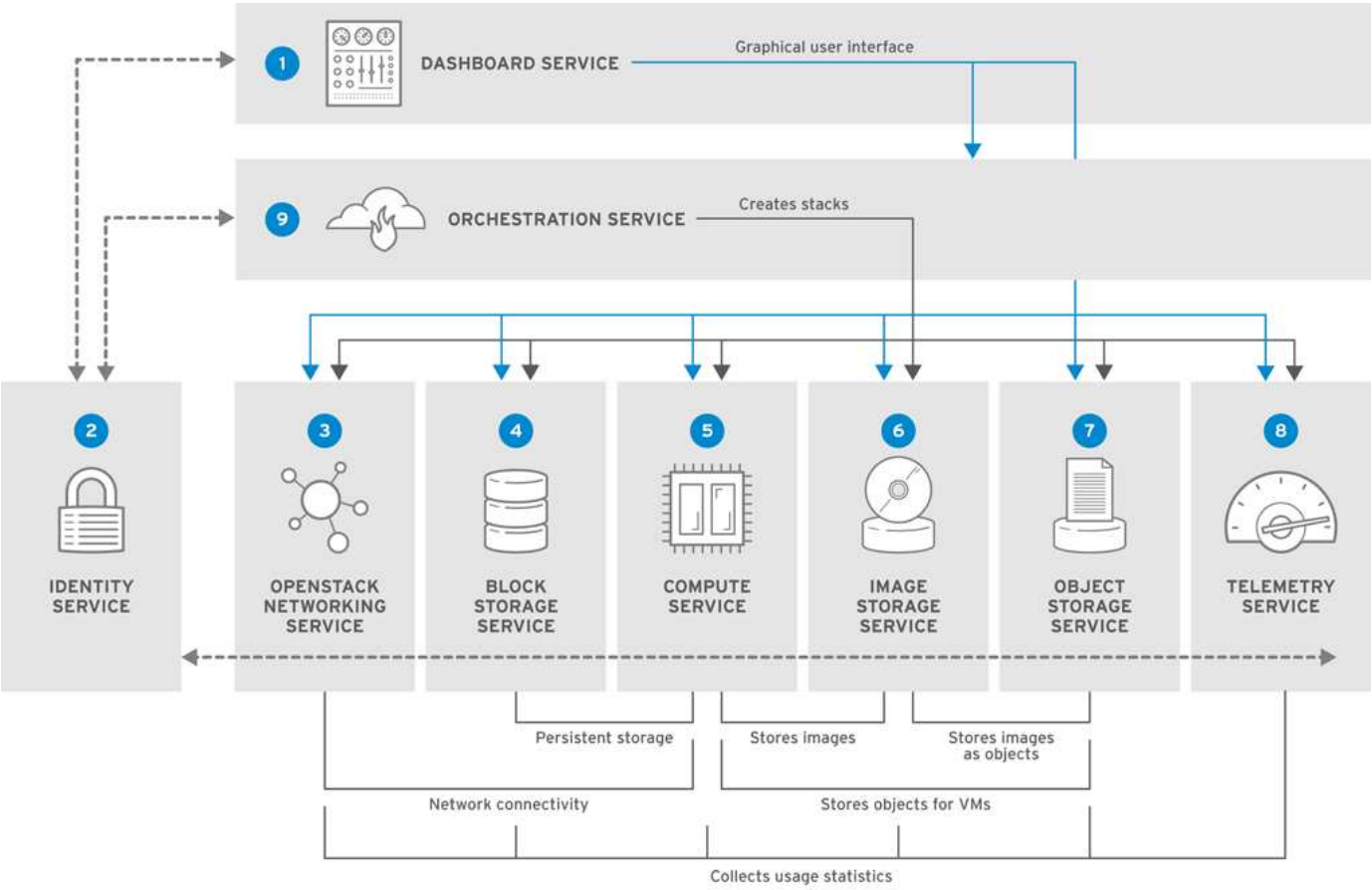
OpenStack プロジェクトは、短期間で開発されたコミュニティプロジェクトで、 6 カ月ごとに更新リリースを提供します。最初の Red Hat OpenStack Platform は、すべてのアップストリームリリースに加えて新しいリリースを公開することで、このリリースサイクルのペースを維持していました。また、 3 回目のリリースごとに長期的なサポートを提供します。最近、 OpenStack Train をベースとした OSP リリース 16.0 ではリリース番号に対応しないことが選択されましたが、新しい機能はサブリリースにバックポートされています。最新のリリースは Red Hat OpenStack Platform 16.1 です。これには、アップストリームの Usuri 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、を参照してください "Red Hat OpenStack Platform の Web サイト"。

OpenStack サービス

OpenStack Platform サービスはコンテナとして導入されます。コンテナはサービスを分離するため、アップグレードも簡単です。OpenStack Platform は、 Kolla によって構築、管理された一連のコンテナを使用しま

す。サービスの導入は、 Red Hat Custom Portal からコンテナイメージを取得することによって行われます。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 で導入、設定、および管理されます。



サービス	プロジェクト名	説明
ダッシュボード	地平線	OpenStack サービスの管理に使用する Web ブラウザベースのダッシュボード。
ID	Keystone	OpenStack サービスの認証と許可、およびユーザ、プロジェクト、ロールの管理を一元化するサービスです。
OpenStack ネットワーク	中性子	OpenStack サービスのインターフェイス間の接続を提供します。
ブロックストレージ	Cinder の場合	仮想マシン（ VM ）の永続的な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	ノバ	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（ Image ）	Glance	VM イメージやボリューム Snapshot などのリソースを格納するためのレジストリサービス。
オブジェクトストレージ	Swift	ユーザにファイルおよび任意のデータの格納および取得を許可します。
テレメータ	Ceilometer	クラウドリソースの使用状況を測定できます。

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

NetApp 解決策を使用した Red Hat OpenShift では、 2 つのデータスイッチを使用して 25Gbps 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用に 1Gbps で接続を提供する管理スイッチをさらに 2 台使用し、 IPMI 機能のアウトオブバンド管理も行います。

Red Hat OpenStack Director では、皮肉な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 を導入するために、 IPMI 機能が必要です。

VLAN の要件

ネットアップとともに Red Hat OpenShift を実装することで、仮想ローカルエリアネットワーク（ VLAN ）を使用し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、お客様のニーズに合わせて拡張することも、特定のネットワークサービスをさらに分離することもできます。次の表に、ネットアップで解決策を検証する際に解決策を実装するために必要な VLAN を示します。

VLAN	目的	VLAN ID
アウトオブバンド管理ネットワーク	物理ノードの管理に使用するネットワークと、皮肉なことに IPMI サービス。	16
ストレージインフラ	Swift などのインフラサービスをサポートするためにボリュームを直接マッピングするためのコントローラノードのネットワーク。	201
ストレージ Cinder	環境に導入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ピングして接続するためのネットワーク。	202.
内部 API	API 通信、 RPC メッセージ、データベース通信を使用する OpenStack サービス間の通信に使用するネットワーク。	301
テナント	Neutron は、 VXLAN を介したトンネリングによって、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、ネットワークネームスペースとは、複数のテナントネットワークで同じアドレス範囲を使用しても競合が発生することを意味します	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、対象の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、ユーザ要求と基盤となるストレージレイヤの中間インターフェイスとして機能します。プロキシは受信要求を受信し、要求されたデータを取得するために必要なレプリカを検索します。	303
PXE	OpenStack Director は、 OSP Overcloud のインストールをオーケストレーションするための、皮肉な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	OpenStack Dashboard （ Horizon ）をグラフィカルに管理するためにホストする、公開されているネットワーク。 OpenStack サービスを管理するためのパブリック API 呼び出しが可能です。	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、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 を導入する前に、次のインフラを用意する必要があります。

- ホスト名の完全な解決を可能にする DNS サーバが少なくとも 1 つ必要です。
- 解決策内のサーバの時刻を同期できる NTP サーバが 3 台以上ある。
- （オプション） OpenShift 環境でのアウトバウンドのインターネット接続。

本番環境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、この解決策を本番環境に導入する前に考慮する必要があるベストプラクティスをいくつか紹介します。

少なくとも **3** つのコンピューティングノードで構成された **OSP** プライベートクラウドに **OpenShift** を導入します。

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では、 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導入して、 HA 運用に適した最小限のハードウェアを導入します。このアーキテクチャにより、耐障害性を備えた構成が実現し、両方のコンピューティングノードで仮想インスタンスを起動し、導入した VM を 2 つのハイパーバイザー間で移行できます。

Red Hat OpenShift 原因では最初に 3 つのマスターノードを導入するため、 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可能性があり、その特定のノードが使用できなくなった場合には OpenShift が停止する可能性があります。そのため、 Red Hat では、少なくとも 3 つの OSP コンピューティングノードを導入して、 OpenShift マスターを均等に分散させ、解決策にフォールトトレランスを強化することをベストプラクティスとして推奨します。

仮想マシンとホストのアフィニティを設定します

仮想マシンとホストのアフィニティを有効にすると、複数のハイパーバイザーノードに OpenShift マスターを分散できます。

アフィニティとは、 VM やホストのセットに対してルールを定義する方法で、グループ内の同じホストで複数の VM が実行されるか、別々のホストで実行されるかを決定します。VM と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で、 VM に適用されます。このアフィニティグループには同じパラメータと条件が設定されます。アフィニティグループ内の VM がグループ内の同じホストで実行されているのか、または別々のホストで実行されているのかに応じて、アフィニティグループのパラメータでは正のアフィニティまたは負のアフィニティを定義できます。Red Hat OpenStack Platform では、サーバグループを作成し、 Nova で導入されたインスタンスが異なるコンピューティングノードに導入されるようにフィルタを設定することで、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することができます。

サーバグループには、配置を管理できる最大 10 個の仮想インスタンスがデフォルトで存在します。Nova のデフォルトクォータを更新することで変更できます。

> OSP サーバグループには、特定のハードアフィニティや非アフィニティの制限があります。ノードを共有するために十分なリソースが別々のノードに導入できない場合や、リソースが不足している場合は、 VM をブートできません。

アフィニティグループを設定するには、を参照してください "OpenStack インスタンス用にアフィニティおよび非アフィニティを設定するにはどうすればよいですか？"。

OpenShift 環境にカスタムインストールファイルを使用します

IPI を使用すると、このドキュメントで前述した対話型ウィザードを使用して、 OpenShift クラスタを簡単に導入できます。ただし、クラスタ導入の一環として、一部のデフォルト値の変更が必要になる場合があります。

このような場合は、クラスタをすぐに導入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、あとでクラスタを導入できる構成ファイルを作成します。これは、 IPI のデフォルト値を変更する必要がある場合や、マルチテナンシーなどの他の用途のために環境内に同一のクラスタを複数導入する必要がある場合に非常に便利です。OpenShift 用に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、を参照してください "Red Hat OpenShift カスタマイズを使用した OpenStack へのクラスタのインストール"。

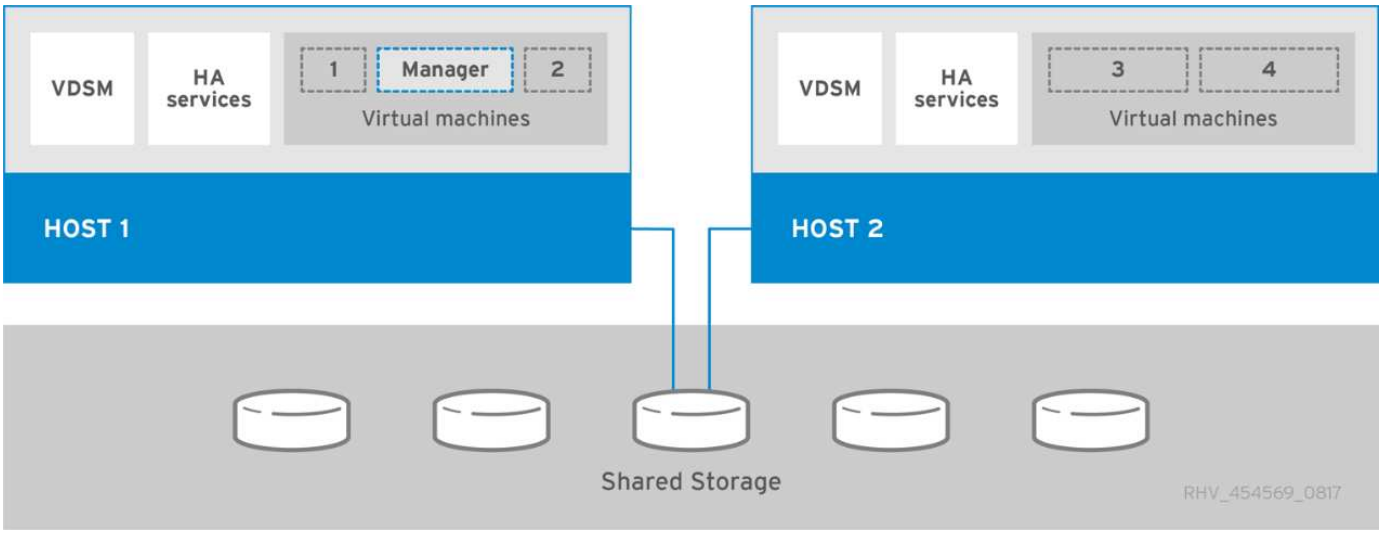
Red Hat 仮想化を基盤とした OpenShift

Red Hat Virtualization （ RHV ）は、 Red Hat Enterprise Linux （ RHEL ）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、を参照してください "Red Hat Virtualization の Web サイト"。

RHV は以下の機能を提供します。

- 仮想マシンとホストの一元管理 RHVマネージャは、導入環境内で物理マシンまたは仮想マシン（VM）として実行され、中央インターフェイスから解決策を管理するためのWebベースのGUIを提供します。
- *自己ホスト型エンジン*ハードウェア要件を最小限に抑えるために、RHV Manager（RHV-M）をゲストVMを実行するホスト上にVMとして導入できます。
- *高可用性*ホストで障害が発生した場合の中断を回避するために、RHVではVMを高可用性用に構成できます。高可用性 VM は、耐障害性ポリシーを使用してクラスタレベルで制御されます。
- 高い拡張性 1つのRHVクラスタに最大200のハイパーバイザホストを配置できるため、IT部門は大規模なVMの要件をサポートし、リソースを大量に消費するエンタープライズクラスのワークロードをホストできます。
- 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）テクノロジーは、ホストおよびVMの高度なセキュリティと強化を目的として、RHVに採用されています。これらの機能の主なメリットは、 VM とそれに関連するリソースを論理的に分離できることです。



ネットワーク設計

NetApp 解決策上の Red Hat OpenShift では、 2 つのデータスイッチを使用して 25Gbps 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用に 1Gbps で接続を提供する管理スイッチを 2 台追加し、 IPMI 機能用にアウトオブバンド管理を使用します。OCP は、クラスタ管理に 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、解決策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、解決策を導入するための前提条件について説明します。

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、さまざまな目的で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、お客様のニーズに合わせて拡張することも、特定のネットワークサービスをさらに分離することもできます。次の表に、ネットアップで解決策を検証する際に解決策を実装するために必要な VLAN を示します。

VLAN	目的	VLAN ID
アウトオブバンド管理ネットワーク	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 用のストレージネットワーク	344
移行用ネットワーク	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 を導入する前に、次のインフラを用意する必要があります。

- 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 DNS サーバが少なくとも 1 台必要です。
- 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できる NTP サーバが少なくとも 1 台必要です。
- （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のアウトバウンドインターネット

接続。

本番環境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、この解決策を本番環境に導入する前に考慮する必要があるベストプラクティスをいくつか紹介します。

少なくとも **3** つの **RHV** クラスタに **OpenShift** を導入します ノード

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導入し、ホスト型エンジンと導入済み VM を両方のホストで管理して 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な構成を確保することによって、 HA 処理に適した最小限のハードウェア導入を示しています。

Red Hat OpenShift は最初に 3 つのマスターノードで導入するため、 2 ノード構成で少なくとも 2 つのマスターが同じノードを占有します。そのため、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。そのため、解決策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導入して、 OpenShift マスターを均等に分散できるようにし、解決策にさらにフォールトトレランスを追加することが 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設定します

VM とホストの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。

アフィニティとは、 VM やホストのセットに対してルールを定義する方法で、グループ内の同じホストで複数の VM が実行されるか、別々のホストで実行されるかを決定します。VM と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で、 VM に適用されます。このアフィニティグループには同じパラメータと条件が設定されます。アフィニティグループ内の VM がグループ内の同じホストで実行されているのか、または別々のホストで実行されているのかに応じて、アフィニティグループのパラメータでは正のアフィニティまたは負のアフィニティを定義できます。

パラメータに定義された条件は、強制またはソフト強制のいずれかです。強制をハードに行うことで、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従って配置されます。ソフトな適用では、可能なかぎり、アフィニティグループ内の VM に対して肯定的または否定的なアフィニティに従って高い優先度が設定されます。このドキュメントで説明する 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。大規模なクラスタでは、ハードアフィニティによって OpenShift ノードを適切に分散できます。

アフィニティグループを設定するには、を参照してください "Red Hat 6.11アフィニティグループのドキュメント"。

OpenShift 環境にカスタムインストールファイルを使用します

IPI を使用すると、このドキュメントで前述した対話型ウィザードを使用して、 OpenShift クラスタを簡単に導入できます。ただし、一部のデフォルト値については、クラスタの導入時に変更が必要になる場合があります。

このような場合は、クラスタをすぐに導入せずにウィザードを実行してタスクを実行できます。クラスタの導入に使用する構成ファイルが作成されます。これは、 IPI のデフォルト値を変更する場合や、マルチテナンシーなどの他の用途のために環境内に同一のクラスタを複数導入する場合に非常に便利です。OpenShift 用に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、を参照してください "Red Hat OpenShift カスタマイズを使用した RHV へのクラスタのインストール"。

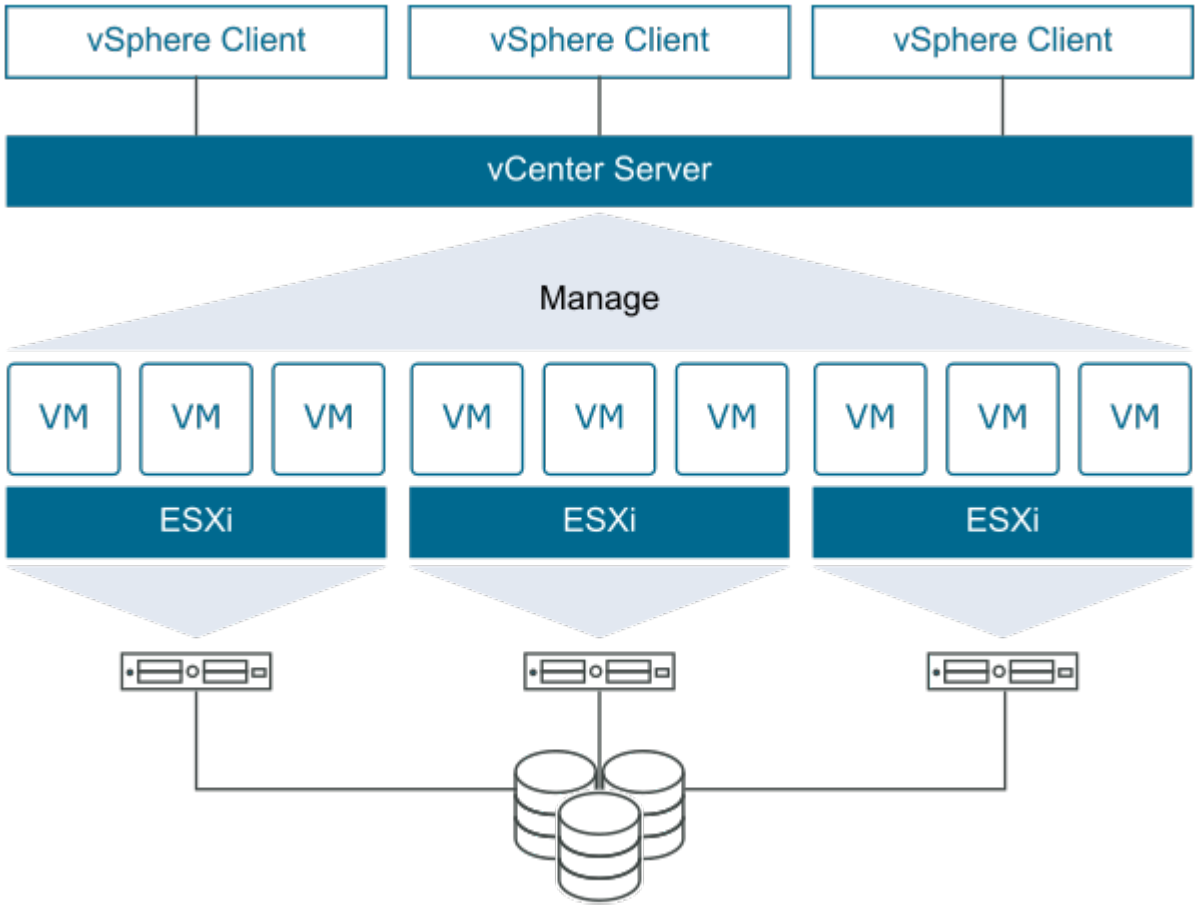
VMware vSphere 上の OpenShift

VMware vSphere は、 ESXi ハイパーバイザー上で実行される多数の仮想サーバとネットワークを一元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、を参照してください "VMware vSphere の Web サイト"。

VMware vSphere には次の機能があります。

- * VMware vCenter Server* VMware vCenter Server は、 1 つ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、ホスト、 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、要求に応じて、システムを停止せずにクラスタ内のノード間でVMをホット移行できます。
- * vSphere High Availability *ホスト障害時のシステム停止を回避するために、VMware vSphereではホストをクラスタ化し、高可用性を実現できるように構成できます。ホストの障害によってシステムが停止した VM は、クラスタ内の他のホストでまもなくリブートされ、サービスがリストアされます。
- * Distributed Resource Scheduler（DRS）* VMware vSphereクラスタは、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設定できます。リソース競合のある VM は、十分なリソースを使用できるように、クラスタ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

NetApp 解決策上の Red Hat OpenShift では、 2 つのデータスイッチを使用して 25Gbps 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用に 1Gbps で接続を提供する管理スイッチを

さらに 2 台使用し、 IPMI 機能のアウトオブバンド管理も行います。OCP のクラスタ管理には、 VMware vSphere 上の VM 論理ネットワークが使用されます。このセクションでは、解決策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、解決策を導入するための前提条件について説明します。

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、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、お客様のニーズに合わせて拡張することも、特定のネットワークサービスをさらに分離することもできます。次の表に、ネットアップで解決策を検証する際に解決策を実装するために必要な VLAN を示します。

VLAN	目的	VLAN ID
アウトオブバンド管理ネットワーク	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 用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 用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 用のストレージネットワーク	3481
移行用ネットワーク	仮想ゲスト移行用のネットワーク	3487

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 を導入する前に、次のインフラを用意する必要があります。

- 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 DNS サーバが少なくとも 1 台必要です。
- 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できる NTP サーバが少なくとも 1 台必要です。
- （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のアウトバウンドインターネット接続。

本番環境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、この解決策を本番環境に導入する前に考慮する必要があるベストプラクティスをいくつか紹介します。

少なくとも 3 つのボリュームからなる ESXi クラスタに OpenShift を導入します ノード

本ドキュメントで説明する検証済みのアーキテクチャには、 VMware vSphere HA と VMware vMotion を有効にして、 2 つの ESXi ハイパーバイザーノードを導入し、フォールトトレラント構成を確保することで、 HA 処理に適した最小限のハードウェア環境が示されています。この構成では、導入した 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リブートすることができます。

Red Hat OpenShift では最初に 3 つのマスターノードを導入するため、 2 ノード構成の少なくとも 2 つのマスターが同じノードを占有することがあります。その場合、特定のノードが使用できなくなったときに

OpenShift が停止する可能性があります。そのため、 Red Hat のベストプラクティスでは、 OpenShift マスターを均等に分散してフォールトトレランスを高めるために、少なくとも 3 つの ESXi ハイパーバイザーノードを導入する必要があります。

仮想マシンとホストのアフィニティを設定します

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

アフィニティまたは非アフィニティは、 VM やホストのセットに対してルールを定義する方法で、グループ内の同じホストまたはホスト上で VM を一緒に実行するか、別のホスト上で実行するかを決定します。VM と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で、 VM に適用されます。このアフィニティグループには同じパラメータと条件が設定されます。アフィニティグループ内の VM がグループ内の同じホストで実行されているのか、または別々のホストで実行されているのかに応じて、アフィニティグループのパラメータでは正のアフィニティまたは負のアフィニティを定義できます。

アフィニティグループを設定するには、を参照してください "vSphere 6.7 ドキュメント：「 DRS アフィニティルールの使用"。

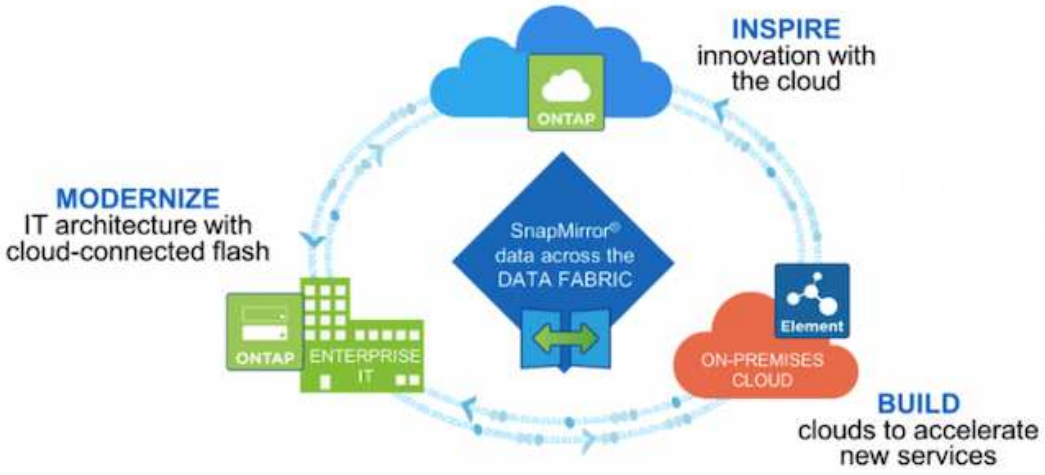
OpenShift 環境にカスタムインストールファイルを使用します

IPI を使用すると、このドキュメントで前述した対話型ウィザードを使用して、 OpenShift クラスタを簡単に導入できます。ただし、クラスタ導入の一環として、一部のデフォルト値の変更が必要になる場合があります。

このような場合は、クラスタをすぐに導入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、あとでクラスタを導入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、マルチテナンシーなどの他の用途のために環境内に同一のクラスタを複数導入する場合に非常に便利です。OpenShift 用に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、を参照してください "Red Hat OpenShift カスタマイズを使用して vSphere にクラスタをインストールします"。

ネットアップストレージの概要

ネットアップには、 Red Hat OpenShift に導入されたアプリケーション用のストレージをプロビジョニングするための、ネットアップの Astra Trident ストレージオーケストレーションツールで認定されているストレージプラットフォームが複数あります。



- AFF システムと FAS システムは、 NetApp ONTAP を実行し、ファイルベース（ NFS ）とブロックベース（ iSCSI ）の両方のユースケースにストレージを提供します。
- Cloud Volumes ONTAP と ONTAP Select は、それぞれクラウドと仮想スペースに同じメリットをもたらします。
- NetApp Cloud Volumes Service （ AWS / GCP ）と Azure NetApp Files は、クラウドでファイルベースのストレージを提供します。
- NetApp Element ストレージシステムは、拡張性に優れた環境でブロックベース（ iSCSI ）のユースケースに対応します。

> ネットアップのポートフォリオに含まれる各ストレージシステムでは、オンプレミスサイトとクラウド間でのデータ管理と移動の両方を容易に行えるため、データがアプリケーションの配置場所にあることを保証できます。

以下のページでは、 Red Hat OpenShift with NetApp 解決策で検証されたネットアップストレージシステムに関する追加情報について説明します。

- "NetApp ONTAP"
- "NetApp Element"

NetApp ONTAP

NetApp ONTAP は、わかりやすい GUI 、自動化統合機能を備えた REST API 、 AI に基づく予測分析と修正措置、無停止のハードウェアアップグレード、ストレージ間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。

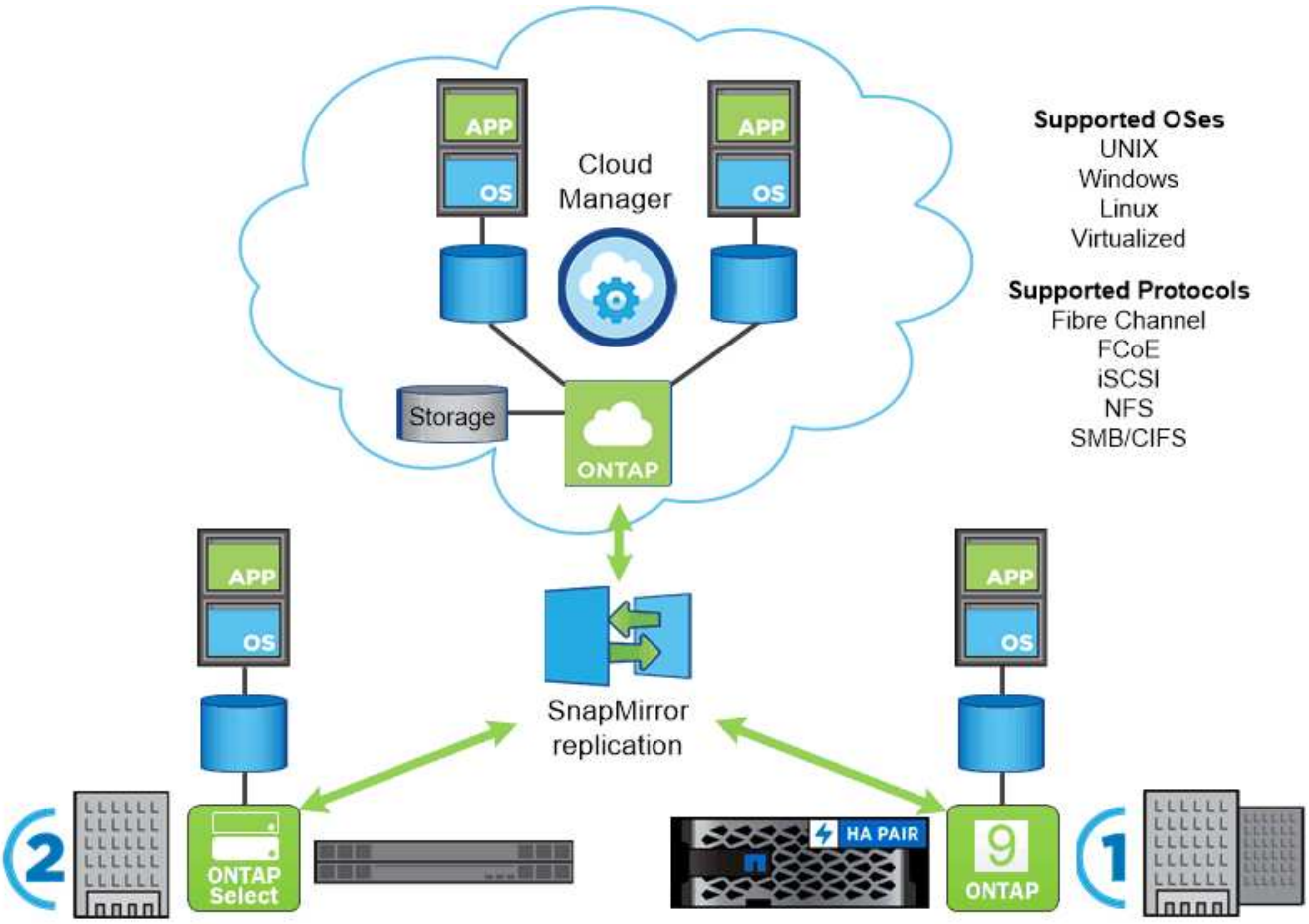
NetApp ONTAP ストレージシステムの詳細については、を参照してください "ネットアップの ONTAP Web サイト"。

ONTAP は以下の機能を提供します。

- NFS 、 CIFS 、 iSCSI 、 FC 、 FCoE を同時にデータアクセスと管理できるユニファイドストレージシステム FC-NVMe プロトコルが必要です。
- 導入モデルには、オンプレミスのオールフラッシュ、ハイブリッド、オール HDD のハードウェア構成、ONTAP Select などのサポートされるハイパーバイザーを使用する VM ベースのストレージプラットフォーム、 Cloud Volumes ONTAP などのクラウドがあります。
- ONTAP システムでは、データの自動階層化、インラインデータ圧縮、重複排除、コンパクションがサポートされ、データストレージ効率が向上しています。
- ワークロードベースの QoS 制御ストレージ：
- パブリッククラウドとのシームレスな統合により、データの階層化と保護を実現ONTAP は、あらゆる環境に対応する堅牢なデータ保護機能も備えています。
  - * NetApp Snapshot コピー。 * 最小限のディスク・スペースでデータをポイント・イン・タイムで高速バックアップし、パフォーマンス・オーバーヘッドを追加する必要はありません。
  - * NetApp SnapMirror 。 * 1 つのストレージ・システムから別のストレージ・システムへデータの Snapshot コピーをミラーリングします。ONTAP では、他の物理プラットフォームやクラウドネイティブのサービスへのデータのミラーリングもサポートされています。
  - * NetApp SnapLock 。 * 指定された期間にわたって上書きまたは消去できない特殊なボリュームに書き込むことにより、書き換え不可能なデータを効率的に管理します。
  - * NetApp SnapVault 。 * は、複数のストレージ・システムのデータを、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 Snapshot 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対して、データをリアルタイムで RAID レベルでミラーリングします。
  - * NetApp SnapRestore * を使用すると、 Snapshot 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迅速にリストアできます。
  - * NetApp FlexClone 。 * Snapshot コピーに基づいて、ネットアップボリュームの読み書き可能なフルコピーを瞬時にプロビジョニングできます。

ONTAP の詳細については、を参照してください "ONTAP 9 ドキュメンテーション・センター"。

NetApp ONTAP は、オンプレミス、仮想環境、クラウド環境で利用できます。



ネットアップのプラットフォーム

NetApp AFF/FAS

ネットアップは、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、低レイテンシのパフォーマンス、統合データプロテクション、マルチプロトコルのサポートのそれぞれに合わせてカスタマイズします。

どちらのシステムも、 NetApp ONTAP データ管理ソフトウェアを搭載しています。 NetApp は、可用性が高く、クラウドと統合されたシンプルなストレージ管理を実現する業界最先端のデータ管理ソフトウェアで、データファブリックのニーズに応じたエンタープライズクラスのスピード、効率性、セキュリティを提供します。

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

ONTAP Select の場合

ONTAP Select は、お客様の環境のハイパーバイザーに導入できる、ソフトウェアで定義された NetApp ONTAP 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 システムの全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、をクリックしてください "こちらをご覧ください"。

Cloud Volumes ONTAP

NetApp Cloud Volumes ONTAP は、クラウドで導入される NetApp ONTAP のバージョンで、 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ます。

Cloud Volumes ONTAP の詳細については、をクリックしてください "こちらをご覧ください"。

NetApp Element ：ネットアップを使用した Red Hat OpenShift

NetApp Element ソフトウェアは、拡張性に優れたモジュラ型のパフォーマンスを提供し、ストレージノードごとに容量とスループットを保証します。NetApp Element システムは、 1 つのクラスタで 4~100 ノードまで拡張でき、高度なストレージ管理機能も多数備えています。

NetApp Element ストレージ・システムの詳細については、を参照してください "ネットアップの SolidFire Web サイト"。

iSCSI ログインのリダイレクト機能と自己回復機能

NetApp Element ソフトウェアは、 iSCSI ストレージプロトコルを利用します。これは、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す。SCSI 標準が変更された場合や、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場合、 iSCSI ストレージプロトコルには変更は必要ありません。

すべてのストレージノードには管理 IP とストレージ IP が設定されますが、 NetApp Element ソフトウェアは、クラスタ内のすべてのストレージトラフィックについて、ストレージ仮想 IP アドレス（ SVIP アドレス）を 1 つアドバタイズします。iSCSI のログインプロセスでは、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ことを応答するため、ネゴシエーションプロセスを続行できません。その後、ホスト側の再設定を必要としないプロセスで、ホストはログイン要求を新しいアドレスに再発行します。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の重要な要素です。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、トラフィックを処理するクラスタのメンバーを決定します。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、単一のノードがボリュームのトラフィックを大量に処理している場合や新しいノードが追加された場合に再配置されます。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。

この方法では、ノード障害のあとにボリュームの再配分が発生しても、ログアウトして新しい場所にリダイレクトしてログインした場合を超えてホスト接続には影響はありません。iSCSI ログインリダイレクションを使用する NetApp Element ソフトウェアクラスタは、無停止のアップグレードと運用が可能な自己回復型のスケールアウトアーキテクチャです。

NetApp Element ソフトウェアクラスタの **QoS**

NetApp Element ソフトウェアクラスタでは、 QoS をボリューム単位で動的に設定できます。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。QoS は、次の 3 つの設定可能なパラメータで定義されます。

- * 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平常時の最小 IOPS 。ボリュームに設定された最小 IOPS は、そのボリュームに対して最低限保証されるパフォーマンスレベルです。ボリュームごとのパフォーマンスが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平常時の最大 IOPS 。
- * Burst IOPS 。 * 短時間のバースト時に許容される最大 IOPS 。バースト期間の設定は、デフォルトの 1 分に設定できます。ボリュームが最大 IOPS レベル未満で動作しているときは、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ってプッシュされると、ボリュームで IOPS が最大 IOPS を超えた短時間のバーストが許容されます。

マルチテナンシー

セキュアマルチテナンシーには、次の機能があります。

- * セキュアな認証。 * Challenge Handshake Authentication Protocol （ CHAP ；チャレンジハンドシェイク認証プロトコル）は、ボリュームへのセキュアなアクセスに使用されます。Lightweight Directory Access Protocol （ LDAP ）は、管理とレポートのためのクラスタへのセキュア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 ）。 * 必要に応じて、任意の数の iSCSI イニシエータ固有の iSCSI Qualified Name （ IQN ）を 1 つ以上のボリュームにマッピングし、認証の代わりに VAG を使用できます。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、 iSCSI イニシエータと NetApp Element ソフトウェアクラスタ間のエンドツーエンドのネットワークセキュリティは、 VLAN を使用することで容易になります。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 については、 NetApp Element ソフトウェアが、特定の VLAN 経由でのみアクセス可能な iSCSI ターゲット SVIP アドレスを別途作成します。
- * VRF 対応 VLAN 。 * データセンターのセキュリティと拡張性をさらにサポートするため、 NetApp Element ソフトウェアを使用すると、 VRF に似た機能を持つテナント VLAN を有効にできます。この機能には、次の 2 つの主要機能が追加されて
  - * 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 イニシエータを、NetApp Element ソフトウェアクラスタとは別のネットワークまたは VLAN に配置できます。
  - * IP サブネットの重複または重複 * 。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、各テナント VLAN に同じ IP サブネットから IP アドレスを割り当てることができます。この機能は、 IPspace の拡張と保持が重要なサービスプロバイダ環境に役立ちます。

エンタープライズクラスのストレージ効率化

NetApp Element ソフトウェアクラスタを使用すると、全体的なストレージ効率とパフォーマンスが向上します。次の機能はインラインで実行されます。常時有効であり、ユーザによる手動設定は必要ありません。

- * 重複排除。 * システムには、一意の 4K ブロックのみが保存されます。重複する 4K ブロックは格納済みのデータ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に格納され、 NetApp Element ソフトウェアの Helix データ保護を使用してミラーリングされます。このシステムは、システム

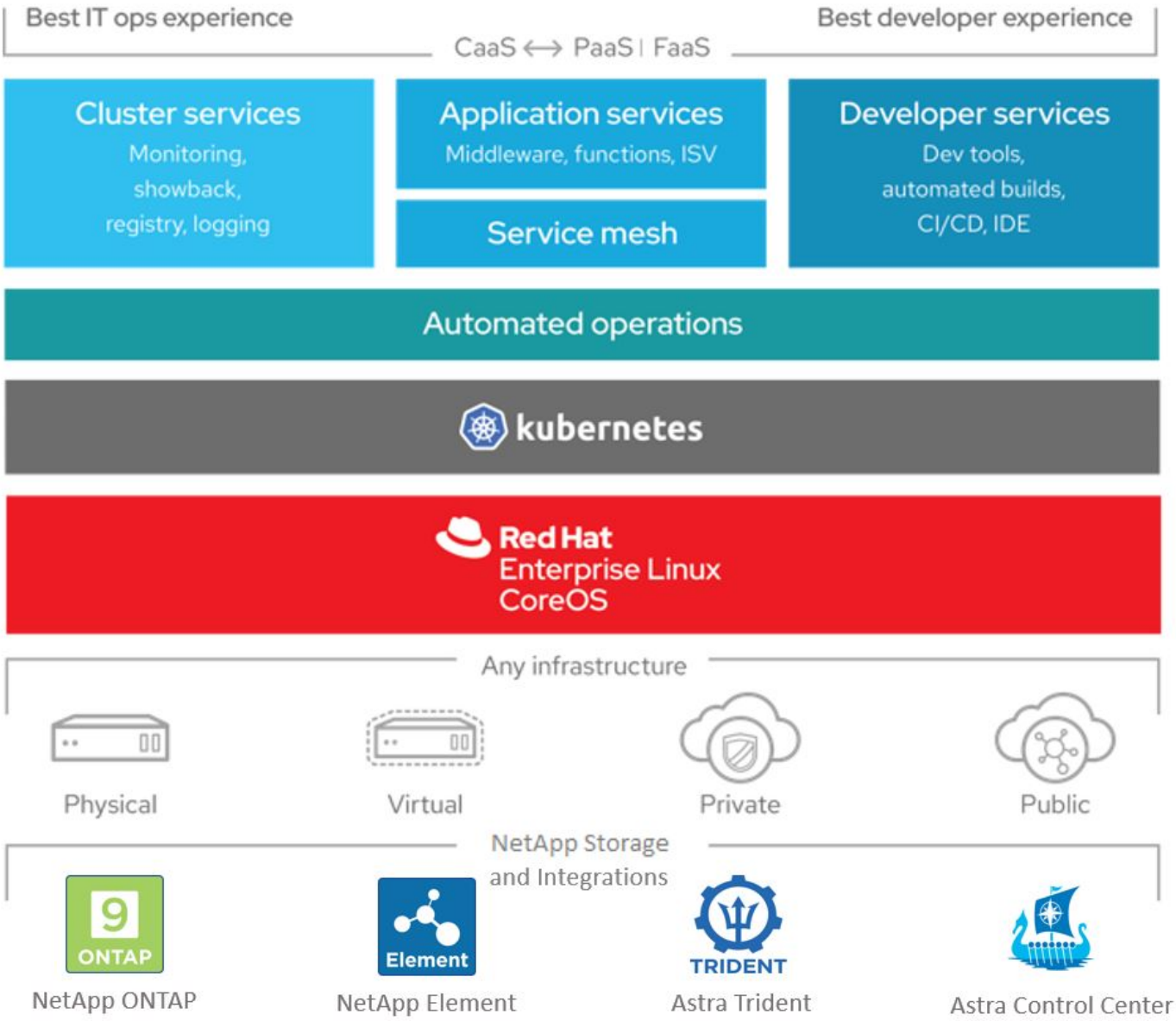
内の容量消費と書き込み処理数を大幅に削減します。

- * 圧縮。 * 圧縮は、データが NVRAM に書き込まれる前にインラインで実行されます。データは 4K ブロック単位で圧縮され、システム内で圧縮されたままとなります。この圧縮により、クラスタ全体での容量消費、書き込み処理数、および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。 * この機能は、必要なときに必要な量のストレージを提供し、オーバープロビジョニングされたボリュームや利用率の低いボリュームによる容量消費を排除します。
- * Helix 。 * 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格納され、セカンダリメタデータドライブにレプリケートされて冗長性が確保されます。

> Element は自動化を目的として設計されました。ストレージ機能はすべて API を使用して利用できます。これらの API は、システムの制御に UI で使用される唯一のメソッドです。

ネットアップストレージ統合の概要

ネットアップは、 Red Hat OpenShift などのコンテナベースの環境における永続的データのオーケストレーションと管理に役立つさまざまな製品を提供します。



NetApp Astra Control は、ネットアップのデータ保護テクノロジを基盤とするステートフル Kubernetes ワークロード向けの充実したストレージサービスとアプリケーション対応データ管理サービスを提供します。Astra Control Service は、クラウドネイティブの Kubernetes 環境でステートフルワークロードをサポートするために利用できます。Astra Control Center は、 Red Hat OpenShift などのオンプレミス環境でステートフルワークロードをサポートするために使用できます。詳細については、 NetApp Astra Control の Web サイトをご覧ください "こちらをご覧ください"。

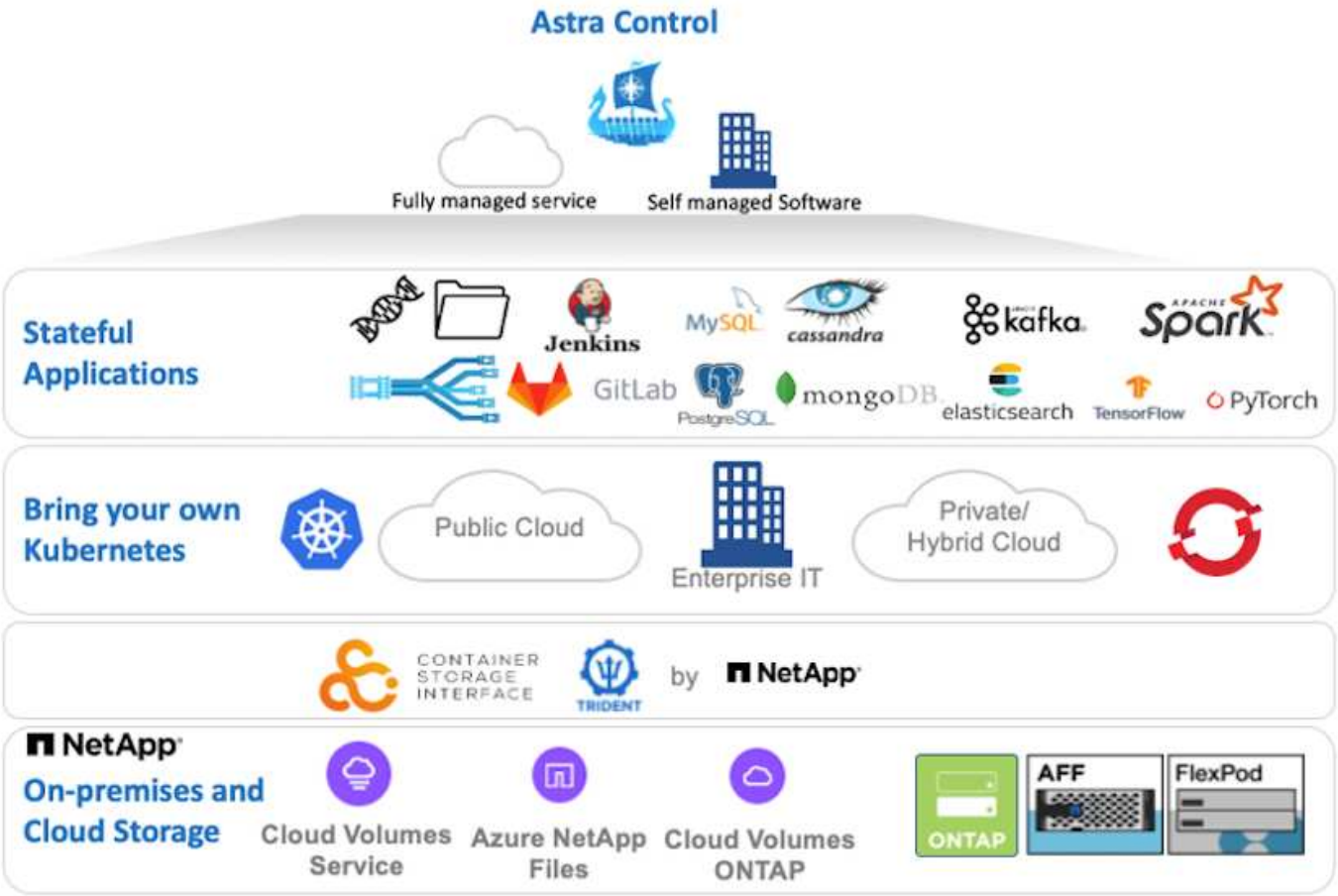
NetApp Astra Trident は、コンテナや Kubernetes ディストリビューション向けの、 Red Hat OpenShift などのオープンソースで完全にサポートされているストレージオーケストレーションツールです。詳細については、 Astra Trident の Web サイトをご覧ください "こちらをご覧ください"。

以下のページには、解決策追加情報に実装された Red Hat OpenShift でアプリケーションおよび永続的ストレージ管理のために検証されたネットアップ製品に関する があります。

- "ネットアップアストラコントロールセンター"
- "ネットアップアストラト Trident"

NetApp Astra Control Center の概要

NetApp Astra Control Center は、オンプレミス環境に導入され、ネットアップのデータ保護テクノロジを基盤とするステートフル Kubernetes ワークロード向けの充実したストレージサービスとアプリケーション対応データ管理サービスを提供します。



NetApp Astra Control Center は、 Astra Trident ストレージオーケストレーションツールを導入し、 NetApp ONTAP ストレージシステムにストレージクラスとストレージバックエンドを使用して構成した Red Hat OpenShift クラスタにインストールできます。

Astra Trident のインストールと設定を行い、 Astra Control Center をサポートするには、を参照してください "このドキュメントはこちら"。

クラウド接続環境では、 Cloud Insights を使用して高度なモニタリングとテレメトリを提供します。Cloud Insights 接続がない場合は、限定的な監視と計測（ 7 日間相当の指標）を使用でき、オープン指標エンドポイントを介して Kubernetes の標準の監視ツール（ Prometheus および Grafana ）にエクスポートされます。

Astra Control Center は、ネットアップの AutoSupport と Active IQ のエコシステムに完全に統合されており、ユーザをサポートし、トラブルシューティングを支援し、使用状況の統計を表示します。

Astra Control Center の有料版に加え、 90 日間の評価ライセンスも提供されています。評価版は、 E メールとコミュニティ（ Slack チャンネル）を通じてサポートされています。お客様は、これらの記事やその他のナレッジベース記事、および製品サポートダッシュボードから入手可能なドキュメントにアクセスできます。

ネットアップアストラコントロールセンターの利用を開始するには、にアクセスしてください "Astra の Web サイト"。

Astra Control Center のインストールの前提条件

1. 1 つ以上の Red Hat OpenShift クラスタ。バージョン 4.6 EUS および 4.7 が現在サポートされています。
2. 各 Red Hat OpenShift クラスタに Astra Trident をインストールして設定しておく必要があります。
3. ONTAP 9.5 以降を実行している NetApp ONTAP ストレージシステムが 1 つ以上必要です。

> サイトに各 OpenShift インストールを実装し、永続的ストレージ専用の SVM を用意することがベストプラクティスです。マルチサイト環境では、追加のストレージシステムが必要です。

4. Trident ストレージバックエンドは、 ONTAP クラスタがサポートする SVM を含む各 OpenShift クラスタで設定する必要があります。
5. ストレージプロビジョニングツールとして Astra Trident を使用し、各 OpenShift クラスタに設定されたデフォルトのストレージクラス。
6. ロードバランシングや OpenShift Services の公開のために、各 OpenShift クラスタにロードバランサをインストールして構成する必要があります。

> リンクを参照してください "こちらをご覧ください" この目的で検証済みのロードバランサに関する情報。

7. NetApp アストラ Control Center イメージをホストするには、プライベートイメージのレジストリを設定する必要があります。

> リンクを参照してください "こちらをご覧ください" この目的のために OpenShift プライベートレジストリをインストールして構成します。

8. Red Hat OpenShift クラスタにクラスタ管理者アクセス権限が必要です。
9. NetApp ONTAP クラスタへの管理者アクセスが必要です。
10. Docker または podman 、 tridentctl 、 OC または kubectl 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。

> Docker をインストールする場合は、 20.10 よりも前のバージョンの Docker 、 Podman をインストールする場合は、バージョン 3.0 よりも前の podman が必要です。

Astra Control Center をインストールします

OperatorHub を使用する

1. ネットアップサポートサイトにログインし、最新バージョンの NetApp Astra Control Center をダウンロードします。そのためには、ネットアップアカウントにライセンスを関連付ける必要があります。tarball をダウンロードしたら、 admin ワークステーションに転送します。

   > Astra Control の試用版ライセンスの使用を開始するには、にアクセスしてください "Astra 登録サイト"。

2. tar ボールを開梱し、作業ディレクトリを作成されたフォルダに変更します。

```
[netapp-user@rhel7 ~]$ tar -vxzf astra-control-center-
21.12.60.tar.gz
[netapp-user@rhel7 ~]$ cd astra-control-center-21.12.60
```

3. インストールを開始する前に、 Astra Control Center イメージをイメージレジストリにプッシュします。この手順では、 Docker または Podman のいずれかを使用して実行します。両方の手順については、この手順で説明します。

ポドマン

a. レジストリ FQDN を、組織 / 名前空間 / プロジェクト名とともに環境変数「管理」としてエクスポートします。

```
[netapp-user@rhel7 ~]$ export REGISTRY=astra-
registry.apps.ocp-vmw.cie.netapp.com/netapp-astra
```

b. レジストリにログインします。

```
[netapp-user@rhel7 ~]$ podman login -u ocp-user -p password
--tls-verify=false astra-registry.apps.ocp-vmw.cie.netapp.com
```

> 「 kubeadmin 」ユーザを使用してプライベートレジストリにログインしている場合は、「 podman login -u OCP -user -p token --tls-verify=false astra-registry.apps.ocp-vmw.cie.netapp.com` 」の代わりにトークンを使用します。

> または、サービスアカウントのトークンを使用して、サービスアカウントを作成し、（プッシュアクセスまたはプルアクセスが必要かどうかに応じて）レジストリエディタまたはレジストリビューアロールを割り当て、レジストリにログインすることもできます。

c. シェルスクリプトファイルを作成し、次の内容を貼り付けます。

```
[netapp-user@rhel7 ~]$ vi push-images-to-registry.sh

for astraImageFile in $(ls images/*.tar) ; do
  # Load to local cache. And store the name of the loaded
image trimming the 'Loaded images: '
  astraImage=$(podman load --input ${astraImageFile} | sed
's/Loaded image(s): //')
  astraImage=$(echo ${astraImage} | sed 's!localhost/!!')
  # Tag with local image repo.
  podman tag ${astraImage} ${REGISTRY}/${astraImage}
  # Push to the local repo.
  podman push ${REGISTRY}/${astraImage}
done
```

> レジストリに信頼されていない証明書を使用している場合は、シェルスクリプトを編集し、 podman push コマンドに「 --tls-verify=false 」を使用します。「 podman push $registry/ $ 」（ echo $astraaIlmage | sed’s /[\\/]\\/'/')--tls-verify=false 」）。

d. ファイルを実行可能にします

```
[netapp-user@rhel7 ~]$ chmod +x push-images-to-registry.sh
```

e. シェルスクリプトを実行します。

```
[netapp-user@rhel7 ~]$ ./push-images-to-registry.sh
```

Docker です

a. レジストリ FQDN を、組織 / 名前空間 / プロジェクト名とともに環境変数「管理」としてエクスポートします。

```
[netapp-user@rhel7 ~]$ export REGISTRY=astra-
registry.apps.ocp-vmw.cie.netapp.com/netapp-astra
```

b. レジストリにログインします。

```
[netapp-user@rhel7 ~]$ docker login -u ocp-user -p password
astra-registry.apps.ocp-vmw.cie.netapp.com
```

> 「 kubeadmin 」ユーザを使用してプライベートレジストリにログインする場合は、「 password - `d Occker login -u OCP-user-p token astra-registry.apps.ocp-vmw.cie.netapp.com` 」の代わりにトークンを使用します。

> または、サービスアカウントのトークンを使用して、サービスアカウントを作成し、（プッシュアクセスまたはプルアクセスが必要かどうかに応じて）レジストリエディタまたはレジストリビューアロールを割り当て、レジストリにログインすることもできます。

c. シェルスクリプトファイルを作成し、次の内容を貼り付けます。

```
[netapp-user@rhel7 ~]$ vi push-images-to-registry.sh

for astraImageFile in $(ls images/*.tar) ; do
  # Load to local cache. And store the name of the loaded
image trimming the 'Loaded images: '
  astraImage=$(docker load --input ${astraImageFile} | sed
's/Loaded image: //')
  astraImage=$(echo ${astraImage} | sed 's!localhost/!!')
  # Tag with local image repo.
  docker tag ${astraImage} ${REGISTRY}/${astraImage}
  # Push to the local repo.
  docker push ${REGISTRY}/${astraImage}
done
```

d. ファイルを実行可能にします

```
[netapp-user@rhel7 ~]$ chmod +x push-images-to-registry.sh
```

e. シェルスクリプトを実行します。

```
[netapp-user@rhel7 ~]$ ./push-images-to-registry.sh
```

4. 公開されていないプライベートイメージレジストリを使用する場合は、イメージレジストリ TLS 証明書を OpenShift ノードにアップロードします。そのためには、 TLS 証明書を使用して OpenShift -config ネームスペースに ConfigMap を作成し、クラスタイメージ構成にパッチを適用して証明書を信頼できるようにします。

```
[netapp-user@rhel7 ~]$ oc create configmap default-ingress-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":"default-ingress-
ca"}}}' --type=merge
```

> ルートとともに入力オペレータからのデフォルト TLS 証明書を含む OpenShift 内部レジストリを使用している場合は、前の手順に従って、ルートホスト名に証明書をパッチする必要があります。入力オペレータから証明書を抽出するには、コマンド「 oc extract secret/router-ca --keys=tls.crt-n OpenShift ingress-operator 」を使用します。

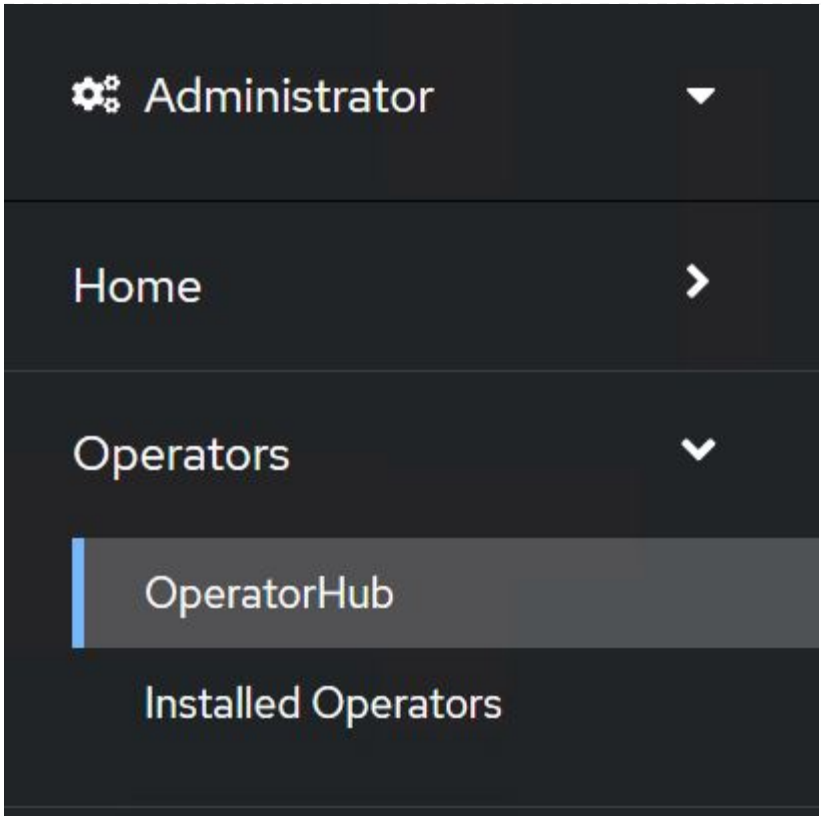
5. Astra Control Center 用の名前空間 NetApp-acc-operator' を作成します

```
[netapp-user@rhel7 ~]$ oc create ns netapp-acc-operator

namespace/netapp-acc-operator created
```

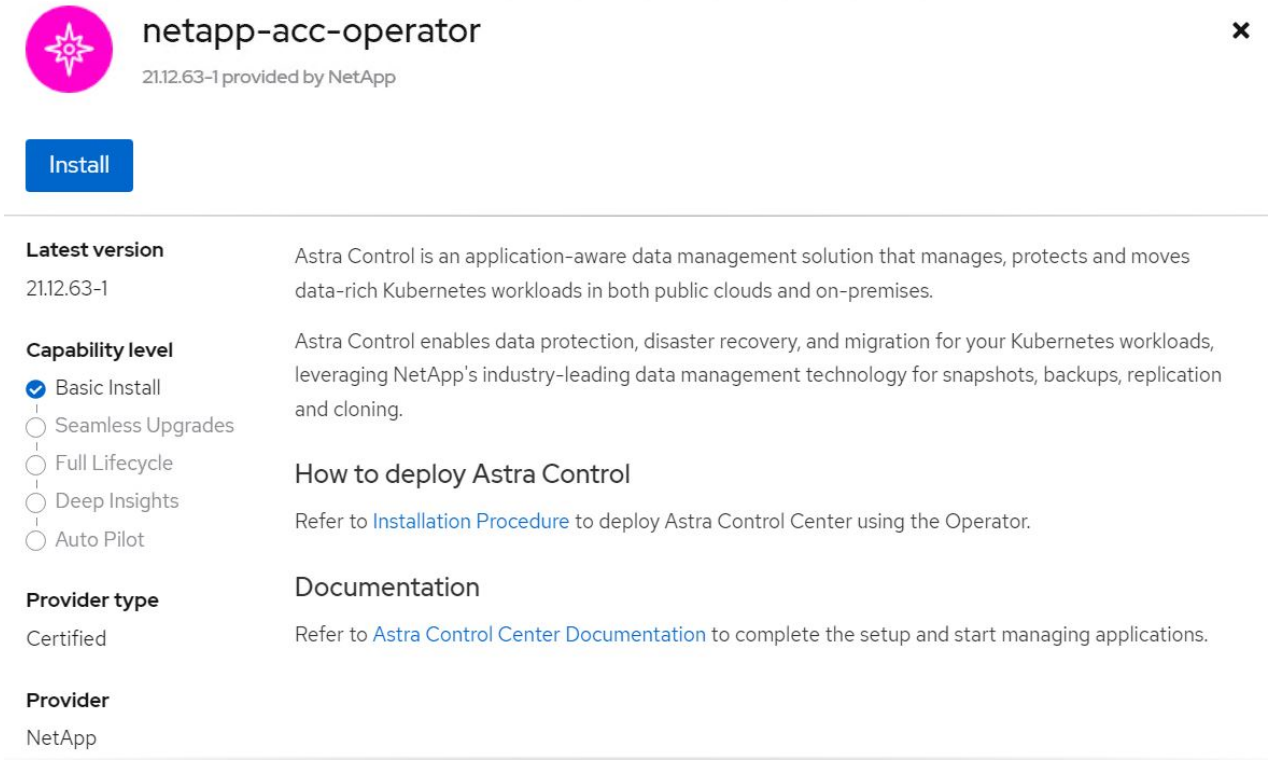
6. NetApp-acc-operator ネームスペースのイメージレジストリにログインするためのクレデンシャルを含むシークレットを作成します。

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

secret/astra-registry-cred created
```

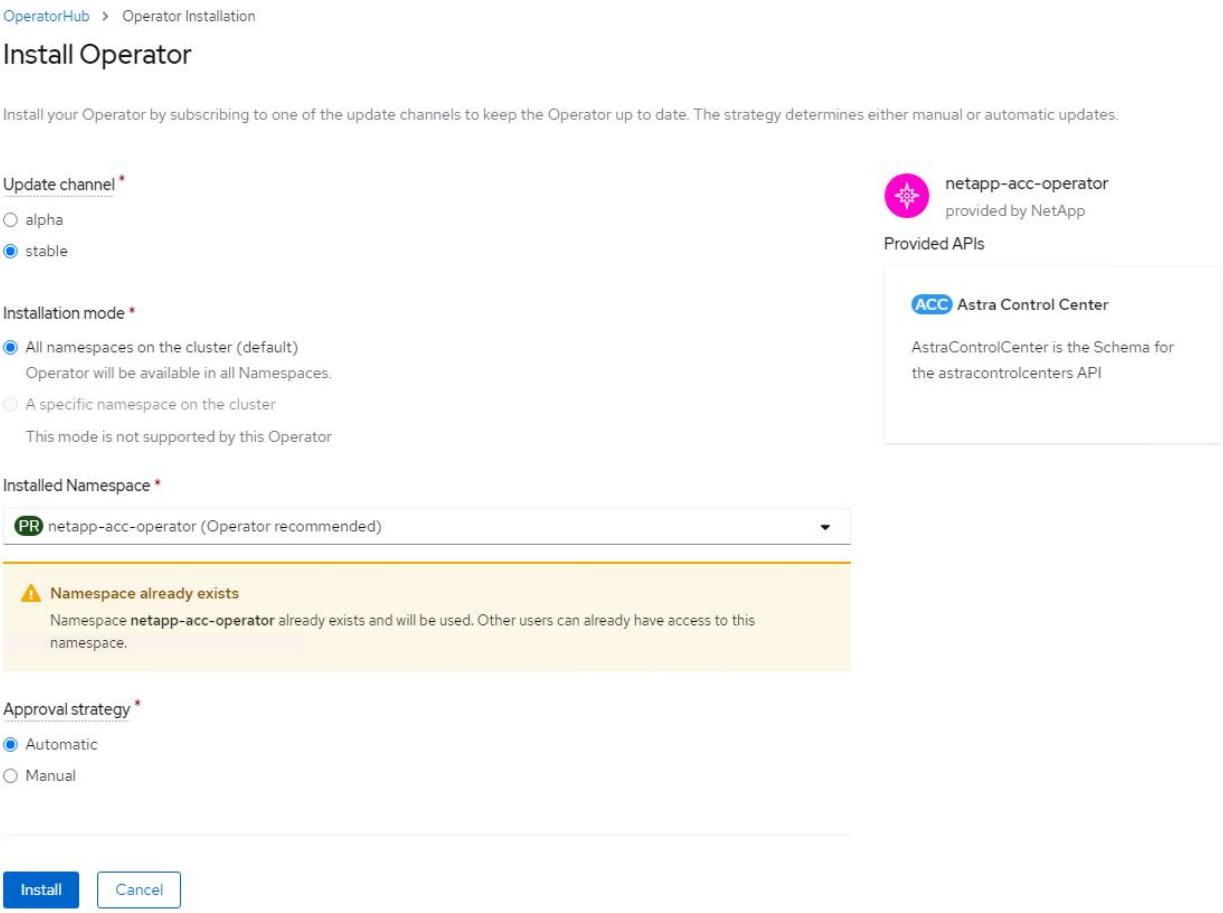
7. クラスタ管理者アクセスで Red Hat OpenShift GUI コンソールにログインします。
8. Perspective ドロップダウンから Administrator を選択します。
9. [演算子]>[演算子ハブ] の順に移動し、 Astra を検索します。



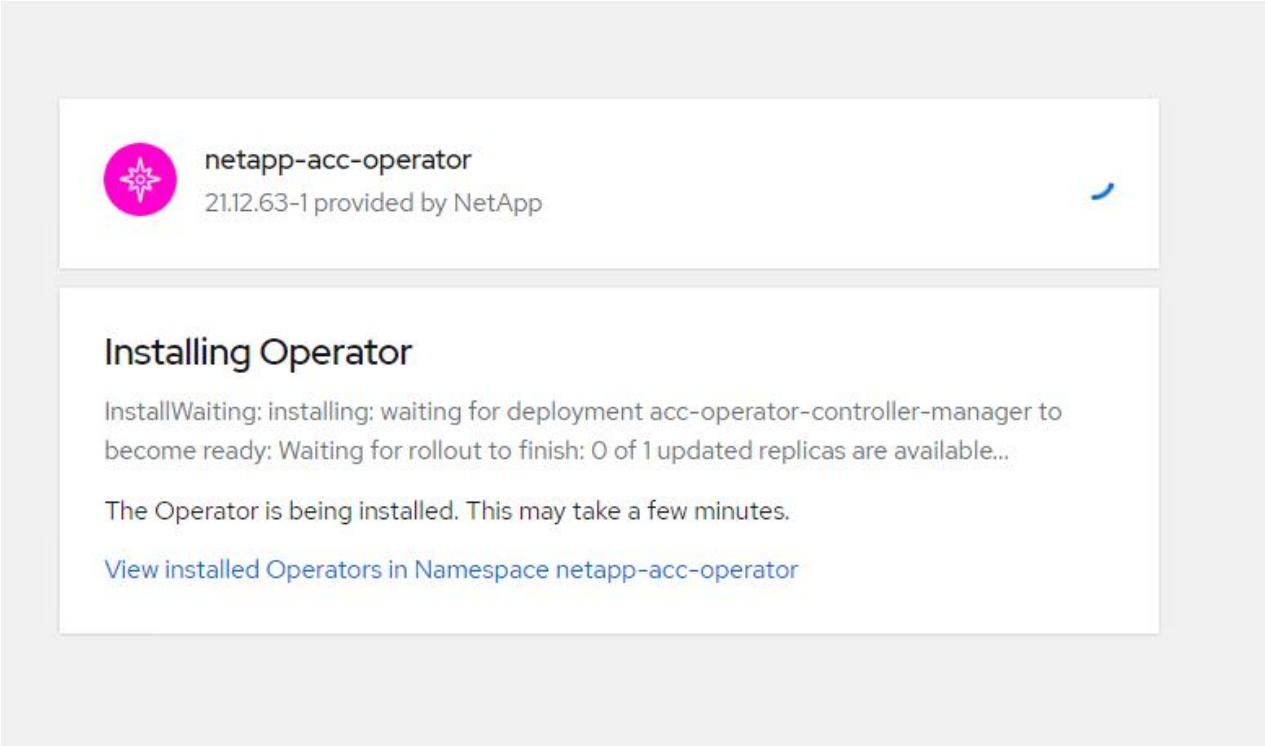
10. NetApp-acc-operator' タイルを選択し、 [インストール] をクリックします。



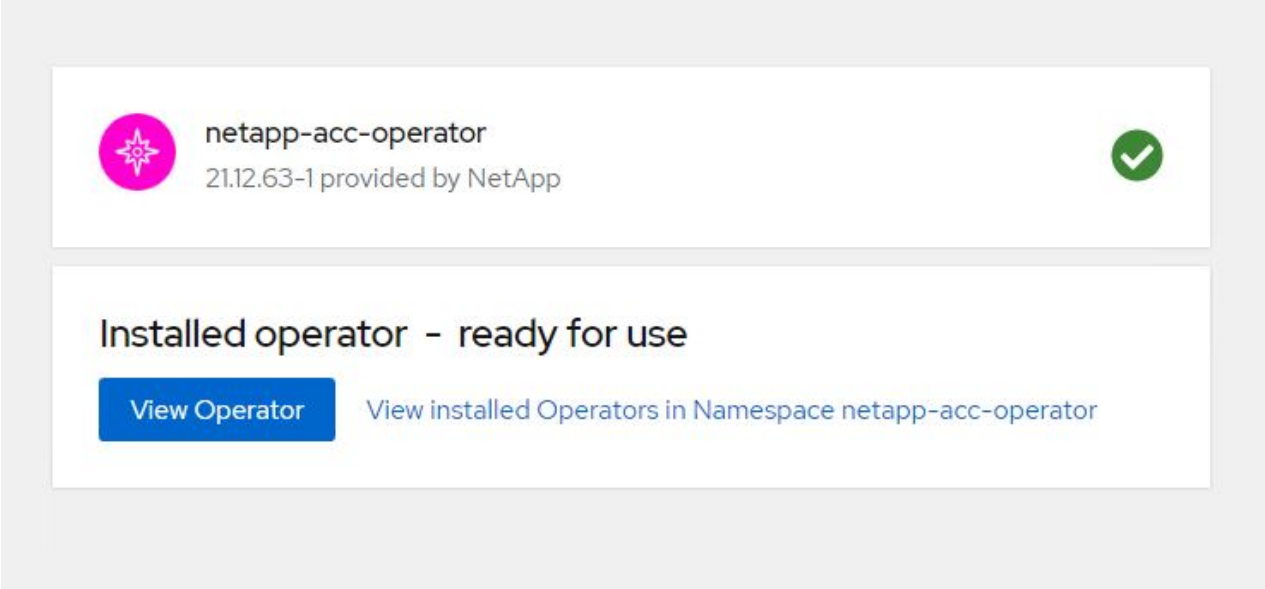
11. インストールオペレータ画面で、デフォルトのパラメータをすべて受け入れて、「インストール」をクリックします。



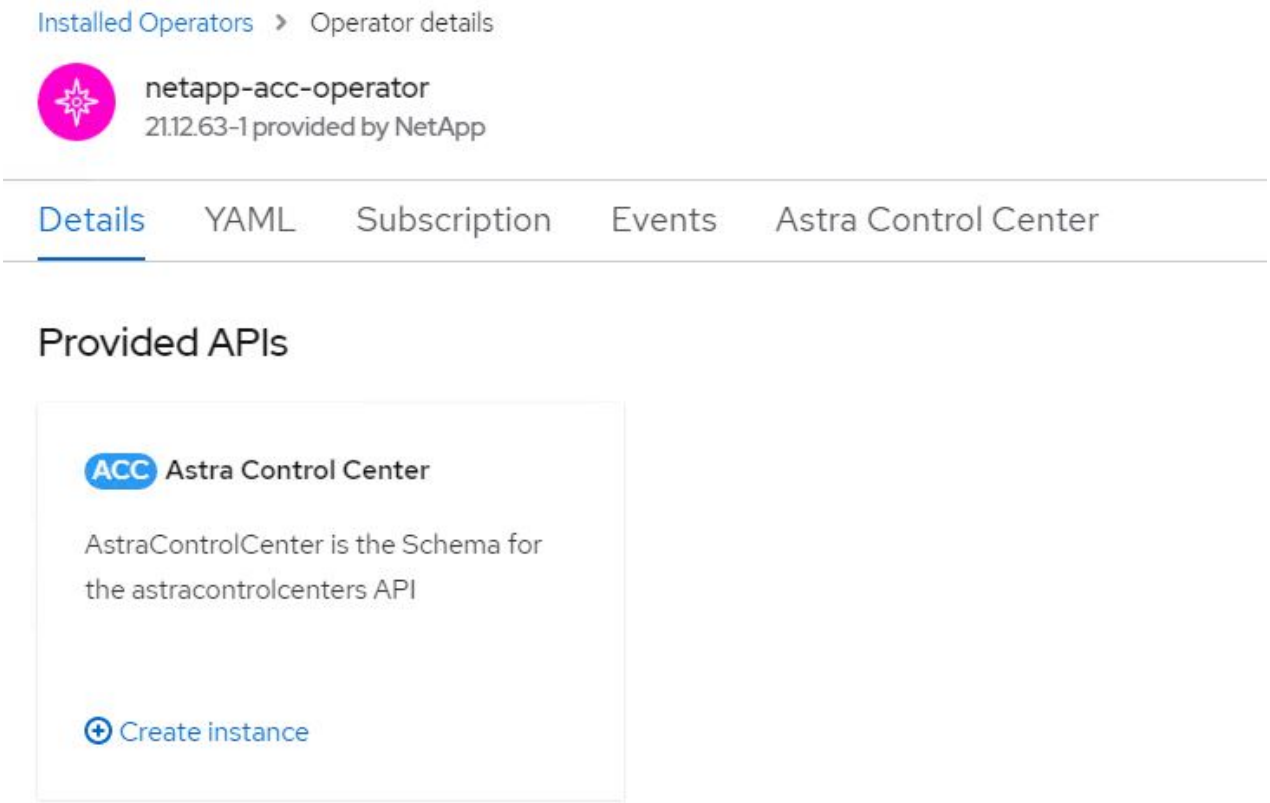
12. オペレータによるインストールが完了するまで待ちます。



13. オペレータのインストールが完了したら、 [View Operator] をクリックします。



14. 次に、オペレーターの Astra Control Center タイルで [Create Instance] をクリックします。



15. [Create AstraeControl] フォームフィールドに入力し '[Create] をクリックします

    a. 必要に応じて、 Astra Control Center インスタンス名を編集します。
    b. 必要に応じて、 AutoSupport を有効または無効にします。Auto Support 機能の保持を推奨します。
    c. Astra Control Center の FQDN を入力します。
    d. Astra Control Center のバージョンを入力します。デフォルトで最新のバージョンが表示されま

す。

e. Astra Control Center のアカウント名を入力し、管理者の詳細（名、姓、メールアドレスなど）を入力します。

f. ボリューム再利用ポリシーを入力します。デフォルトは Retain です。

g. Image Registry に、レジストリの FQDN と、イメージをレジストリにプッシュする際に指定した組織名を入力します（この例では「 astra-registry.apps.ocp-vmw.cie.netapp.com/netapp-astra`」）。

h. 認証が必要なレジストリを使用する場合は、 [イメージレジストリ] セクションにシークレット名を入力します。

i. Astra Control Center のリソース制限のスケーリングオプションを設定します。

j. デフォルト以外のストレージクラスに PVC を配置する場合は、ストレージクラス名を入力します。

k. CRD 処理の環境設定を定義します。

Project: netapp-acc-operator

Name *

astra

Labels

app=frontend

Account Name *

HCG Solutions Engineering

Astra Control Center account name

Astra Address *

astra-control-center.cie.netapp.com

AstraAddress defines how Astra will be found in the data center. This IP address and/or DNS A record must be created prior to provisioning Astra Control Center. Example - "astra.example.com" The A record and its IP address must be allocated prior to provisioning Astra Control Center

Astra Version *

21.12.60

Version of AstraControlCenter to deploy. You are provided a Helm repository with a corresponding version. Example - 1.5.2, 1.4.2-patch

Email *

solutions_tme@netapp.com

EmailAddress will be notified by Astra as events warrant.

Auto Support *

AutoSupport indicates willingness to participate in NetApp's proactive support application, NetApp Active IQ. The default election is true and indicates support data will be sent to NetApp. An empty or blank election is the same as a default election. Air gapped installations should enter false.

First Name

HCG

The first name of the SRE supporting Astra.

Last Name

Admin

The last name of the SRE supporting Astra.

Image Registry

The container image registry that is hosting the Astra application images, ACC Operator and ACC Helm Repository.

Name

astra-registry.apps.ocp-vmw.cie.netapp.com/netapp-astra

The name of the image registry. For example "example.registry/astra". Do not prefix with protocol.

Secret

astra-registry-cred

The name of the Kubernetes secret that will authenticate with the image registry.

Volume Reclaim Policy

Retain

Reclaim policy to be set for persistent volumes

Astra Resources Scaler

Default

Scaling options for AstraControlCenter Resource limits.

Storage Class

The storage class to be used for PVCs. If not set, default storage class will be used.

Crds

Options for how ACC should handle CRDs.

Create Cancel

自動化された **[Ansible]**

1. Ansibleプレイブックを使用してAstra Control Centerを導入するには、Ansibleがインストールされたユbuntu / RHELマシンが必要です。手順に従います "こちらをご覧ください" UbuntuおよびRHELの場合。
2. Ansible コンテンツをホストする GitHub リポジトリをクローニングします。

```
git clone https://github.com/NetApp-
Automation/na_astra_control_suite.git
```

3. ネットアップサポートサイトにログインし、最新バージョンのNetApp Astra Control Centerをダウンロードします。そのためには、ネットアップアカウントにライセンスを関連付ける必要があります。tar ファイルをダウンロードしたら、ワークステーションに転送します。

> Astra Control の試用版ライセンスの使用を開始するには、にアクセスしてください "Astra 登録サイト"。

4. Astra Control CenterをインストールするOpenShiftクラスタにadminとしてアクセスし、kubeconfig

ファイルを作成または取得します。

5. ディレクトリを na_Astra_control_site に変更します。

```
cd na_astra_control_suite
```

6. 「vars/vars.yml」ファイルを編集し、必要な情報を変数に入力します。

```
#Define whether or not to push the Astra Control Center images to
your private registry [Allowed values: yes, no]
push_images: yes

#The directory hosting the Astra Control Center installer
installer_directory: /home/admin/

#Specify the ingress type. Allowed values - "AccTraefik" or
"Generic"
#"AccTraefik" if you want the installer to create a LoadBalancer
type service to access ACC, requires MetalLB or similar.
#"Generic" if you want to create or configure ingress controller
yourself, installer just creates a ClusterIP service for traefik.
ingress_type: "AccTraefik"

#Name of the Astra Control Center installer (Do not include the
extension, just the name)
astra_tar_ball_name: astra-control-center-22.04.0

#The complete path to the kubeconfig file of the
kubernetes/openshift cluster Astra Control Center needs to be
installed to.
hosting_k8s_cluster_kubeconfig_path: /home/admin/cluster-
kubeconfig.yml

#Namespace in which Astra Control Center is to be installed
astra_namespace: netapp-astra-cc

#Astra Control Center Resources Scaler. Leave it blank if you want
to accept the Default setting.
astra_resources_scaler: Default

#Storageclass to be used for Astra Control Center PVCs, it must be
created before running the playbook [Leave it blank if you want the
PVCs to use default storageclass]
astra_trident_storageclass: basic

#Reclaim Policy for Astra Control Center Persistent Volumes [Allowed
```

```
values: Retain, Delete]
storageclass_reclaim_policy: Retain

#Private Registry Details
astra_registry_name: "docker.io"

#Whether the private registry requires credentials [Allowed values:
yes, no]
require_reg_creds: yes

#If require_reg_creds is yes, then define the container image
registry credentials
#Usually, the registry namespace and usernames are same for
individual users
astra_registry_namespace: "registry-user"
astra_registry_username: "registry-user"
astra_registry_password: "password"

#Kuberenets/OpenShift secret name for Astra Control Center
#This name will be assigned to the K8s secret created by the
playbook
astra_registry_secret_name: "astra-registry-credentials"

#Astra Control Center FQDN
acc_fqdn_address: astra-control-center.cie.netapp.com

#Name of the Astra Control Center instance
acc_account_name: ACC Account Name

#Administrator details for Astra Control Center
admin_email_address: admin@example.com
admin_first_name: Admin
admin_last_name: Admin
```

7. プレイブックを実行して Astra Control Center を導入します。Playbookには、特定の構成用のroot権限が必要です。

   このプレイブックを実行しているユーザがrootである場合、またはパスワードを使用しないsudoが設定されている場合は、次のコマンドを実行してプレイブックを実行します。

   ```
   ansible-playbook install_acc_playbook.yml
   ```

   ユーザにパスワードベースのsudoアクセスが設定されている場合は、次のコマンドを実行してこのPlaybookを実行し、sudoパスワードを入力します。

```
ansible-playbook install_acc_playbook.yml -K
```

インストール後の手順

1. インストールが完了するまでに数分かかることがあります。NetApp-AstrA-cc' ネームスペース内のすべてのポッドとサービスが稼働していることを確認します

```
[netapp-user@rhel7 ~]$ oc get all -n netapp-astra-cc
```

2. 「 acc-operator-controller-manager 」ログをチェックし、インストールが完了したことを確認します。

```
[netapp-user@rhel7 ~]$ oc logs deploy/acc-operator-controller-manager -n
netapp-acc-operator -c manager -f
```

> 次のメッセージは、 Astra Control Center のインストールが正常に完了したことを示します。

```
{"level":"info","ts":1624054318.029971,"logger":"controllers.AstraContro
lCenter","msg":"Successfully Reconciled AstraControlCenter in
[seconds]s","AstraControlCenter":"netapp-astra-
cc/astra","ae.Version":"[21.12.60]"}
```

3. Astra Control Center にログインするためのユーザ名は、 CRD ファイルに提供された管理者の電子メールアドレスで、パスワードは Astra Control Center UUID に付加された文字列「 ACC-` 」です。次のコマンドを実行します。

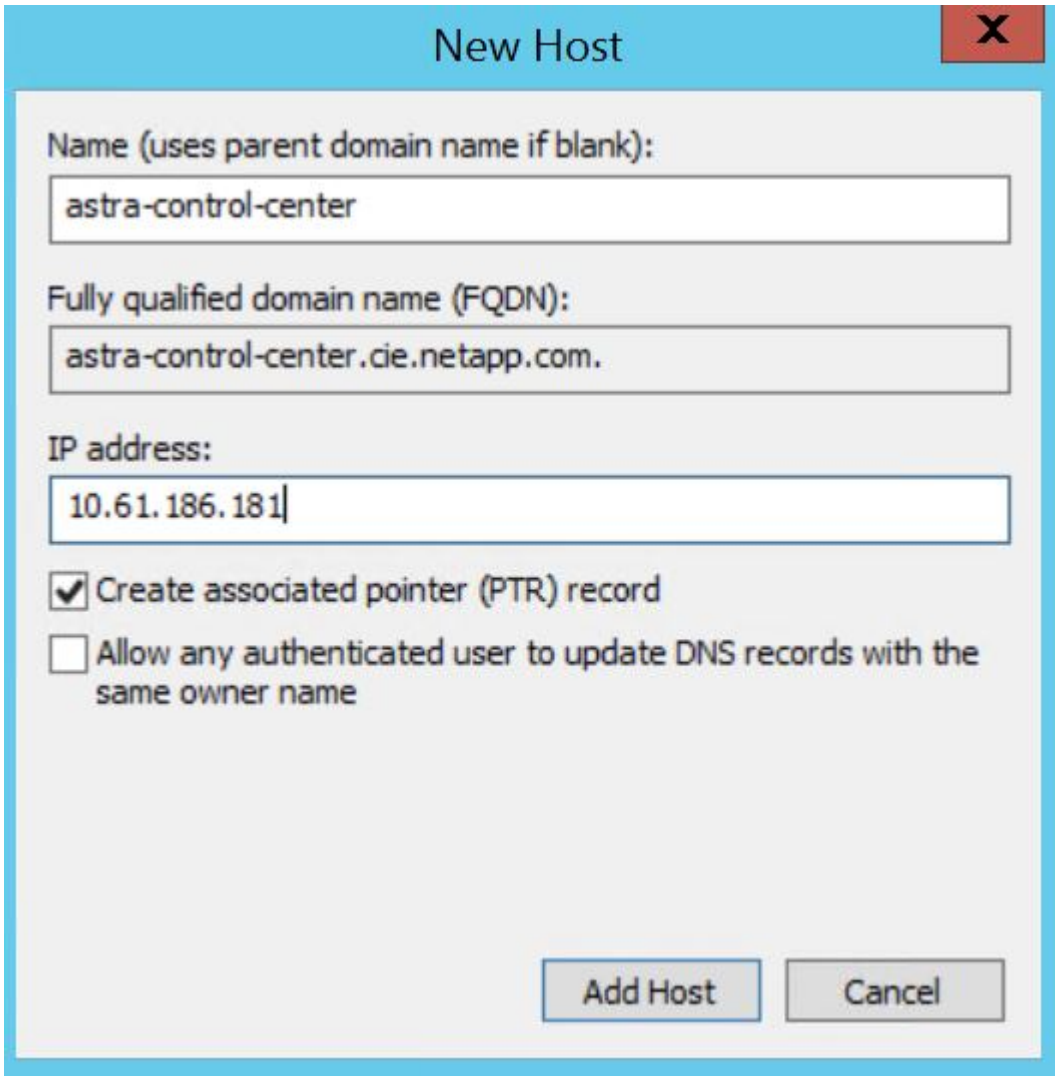
```
[netapp-user@rhel7 ~]$ oc get astracontrolcenters -n netapp-astra-cc
NAME    UUID
astra   345c55a5-bf2e-21f0-84b8-b6f2bce5e95f
```

> この例では、パスワードは「 ACC-345c55a5 -bf2e-21f0 -84b8 -b6f2bce5e95f 」です。

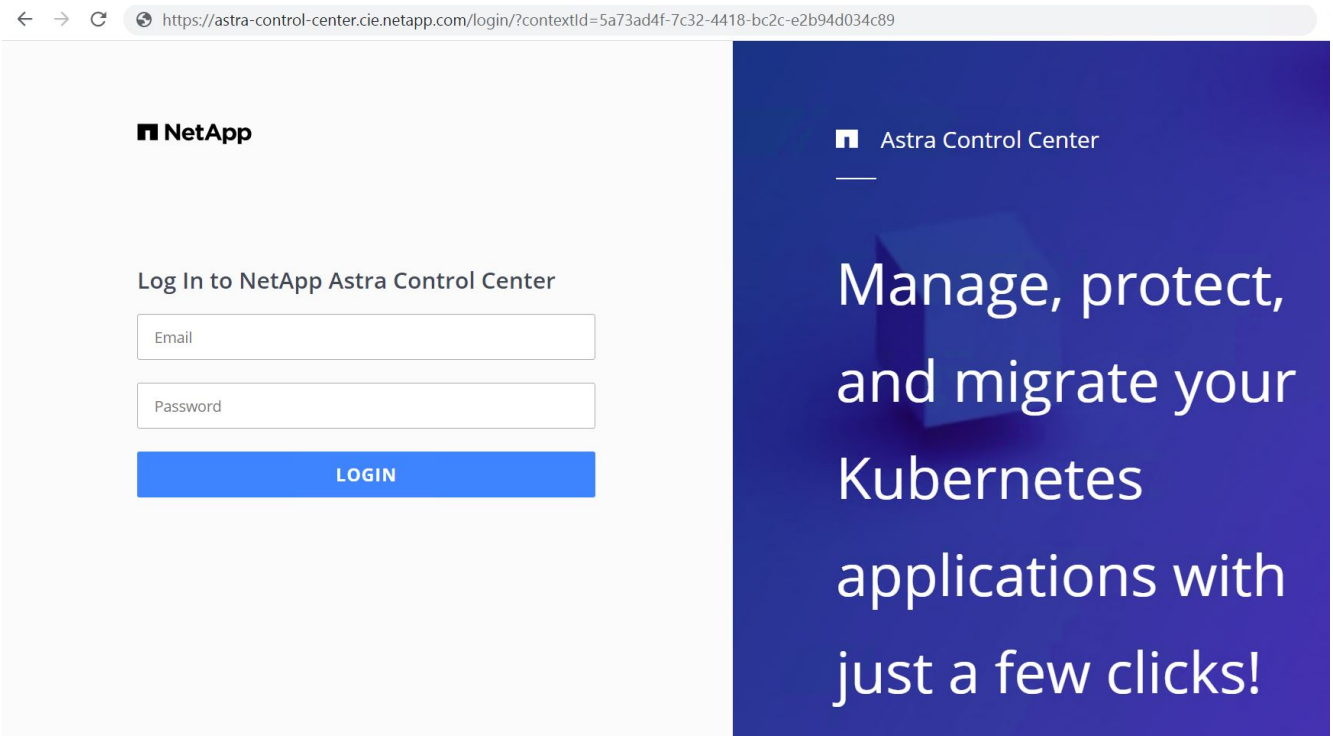
4. traefik サービスのロードバランサ IP を取得します。

```
[netapp-user@rhel7 ~]$ oc get svc -n netapp-astra-cc | egrep
'EXTERNAL|traefik'

NAME                                          TYPE           CLUSTER-IP
EXTERNAL-IP     PORT(S)
AGE
traefik                                       LoadBalancer   172.30.99.142
10.61.186.181   80:30343/TCP,443:30060/TCP
16m
```

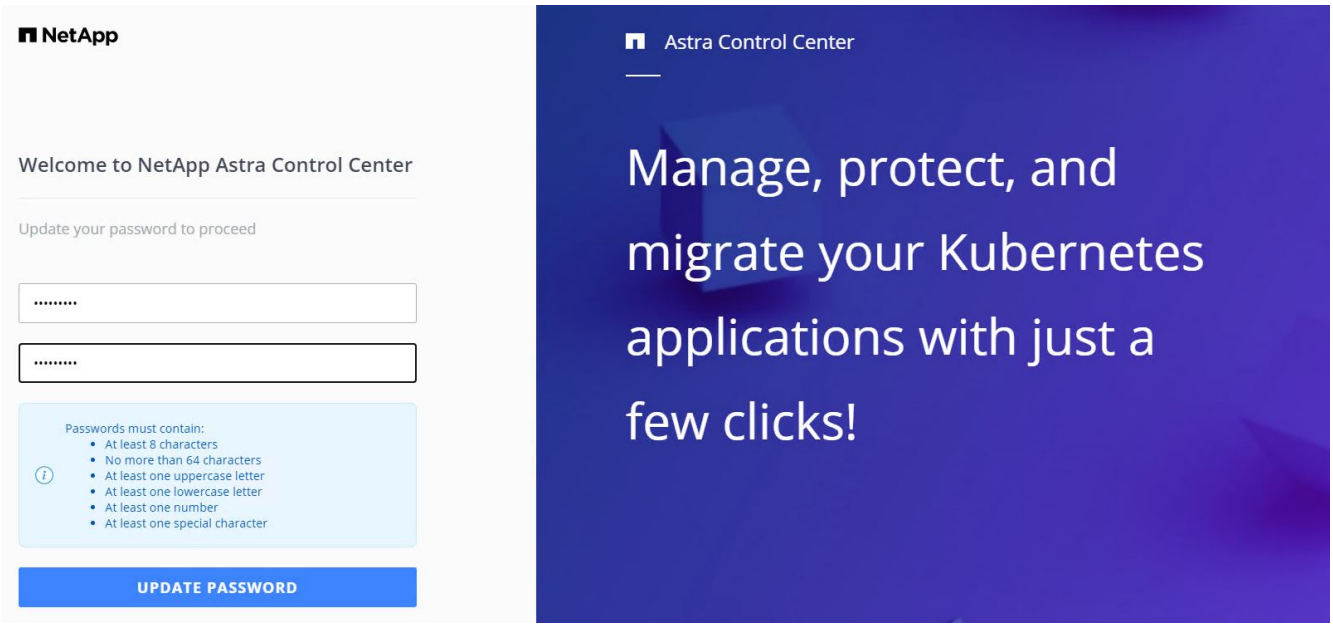
5. Astra Control Center CRD ファイルに指定された FQDN を指す DNS サーバーのエントリを、 traefik サービスの「 external-IP 」に追加します。



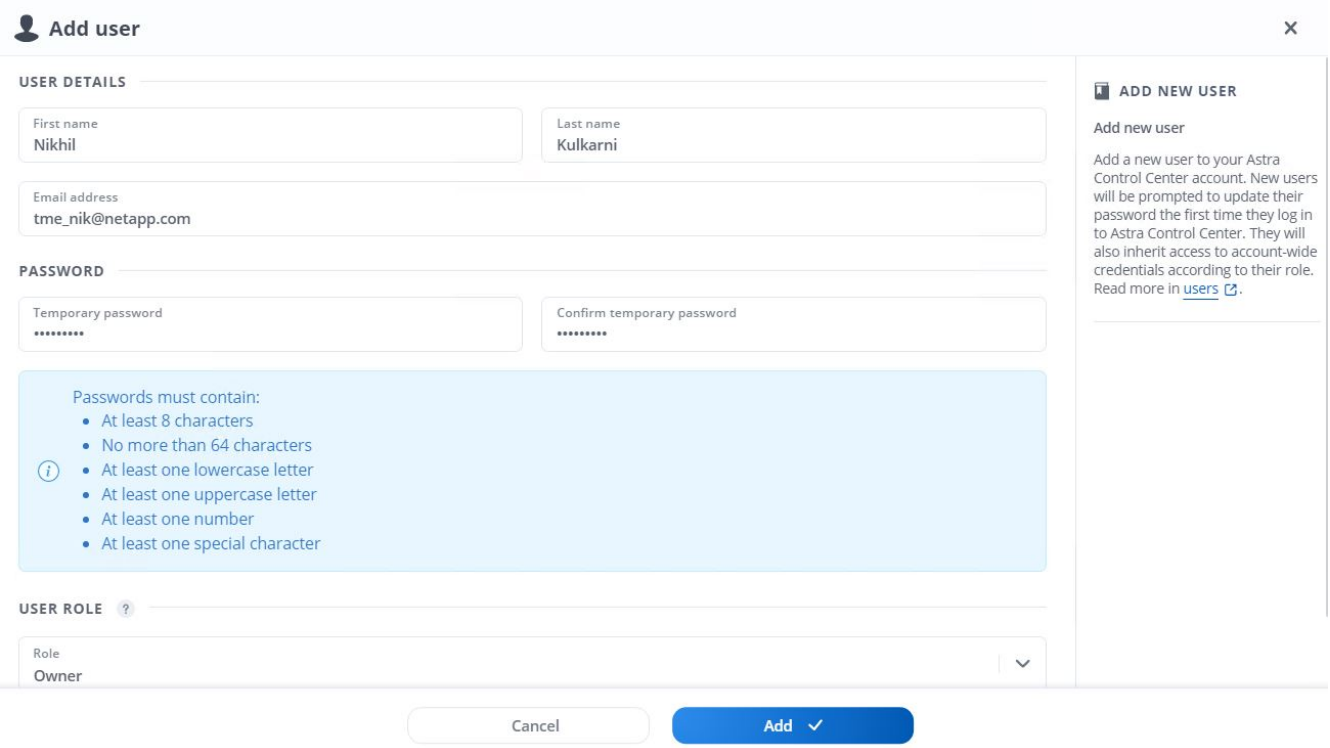
6. Astra Control Center GUI に、 FQDN を参照してログインします。



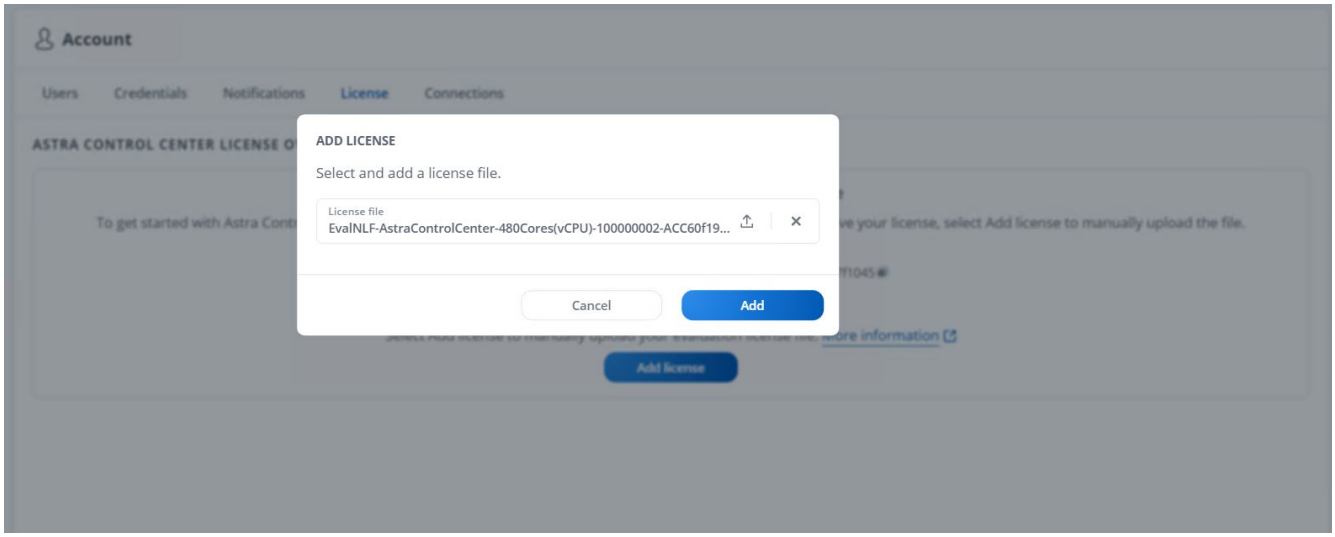
7. CRD で提供された管理者メールアドレスを使用して初めて Astra Control Center GUI にログインする場合は、パスワードを変更する必要があります。



8. ユーザーを Astra Control Center に追加する場合は、 [アカウント]>[ユーザー] の順に選択し、 [追加] をクリックしてユーザーの詳細を入力し、 [追加] をクリックします。



9. Astra Control Center では、すべての機能が動作するためにライセンスが必要です。ライセンスを追加するには、 ［アカウント］ > ［ライセンス］ の順に選択し、 ［ライセンスの追加］ をクリックして、ライセンスファイルをアップロードします。



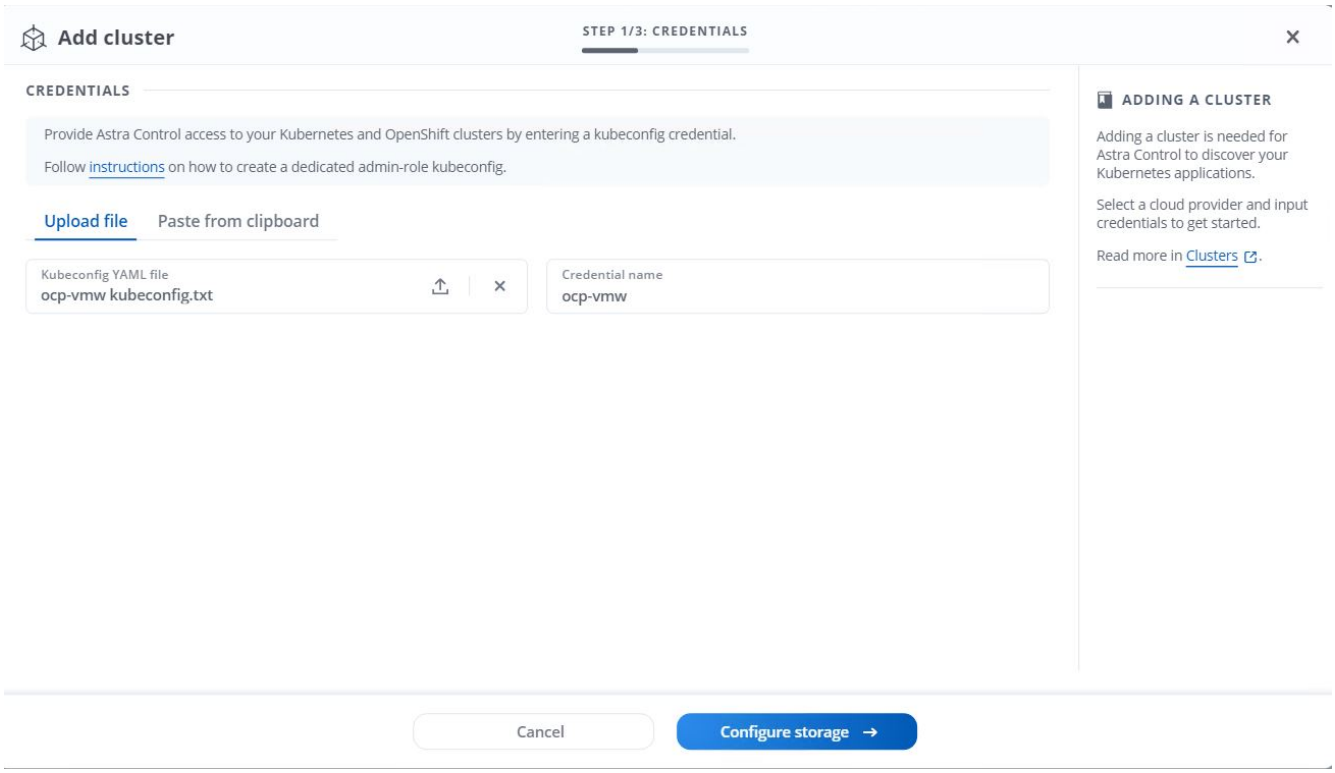
NetApp Astra Control Center のインストールまたは設定で問題が発生した場合は、既知の問題のナレッジベースを利用できます "こちらをご覧ください"。

Red Hat OpenShift クラスタを Astra Control Center に登録します

Astra Control Center でワークロードを管理できるようにするには、まず Red Hat OpenShift クラスタを登録する必要があります。

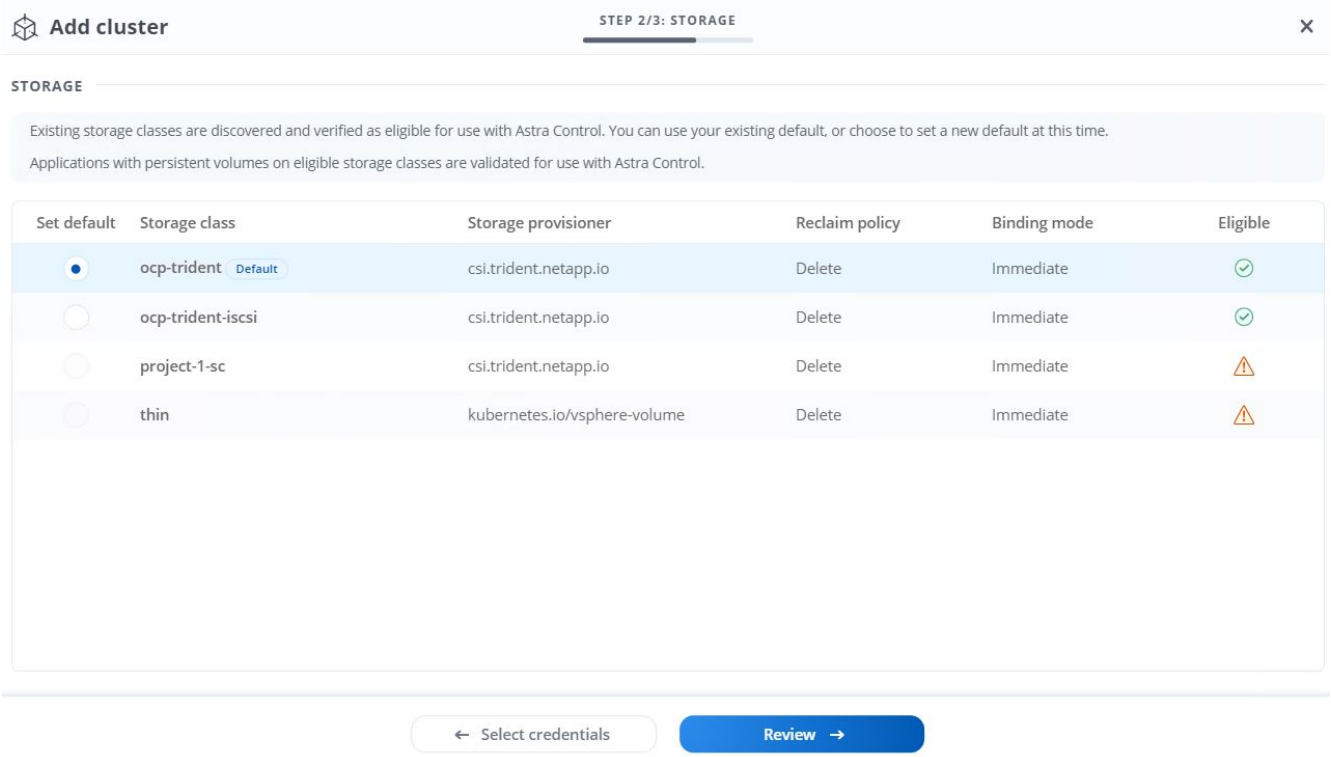
Red Hat OpenShift クラスタを登録します

1. 最初のステップでは、 OpenShift クラスタを Astra Control Center に追加して管理します。クラスタに移動してクラスタの追加をクリックし、 OpenShift クラスタの kubeconfig ファイルをアップロードして、ストレージの選択をクリックします。

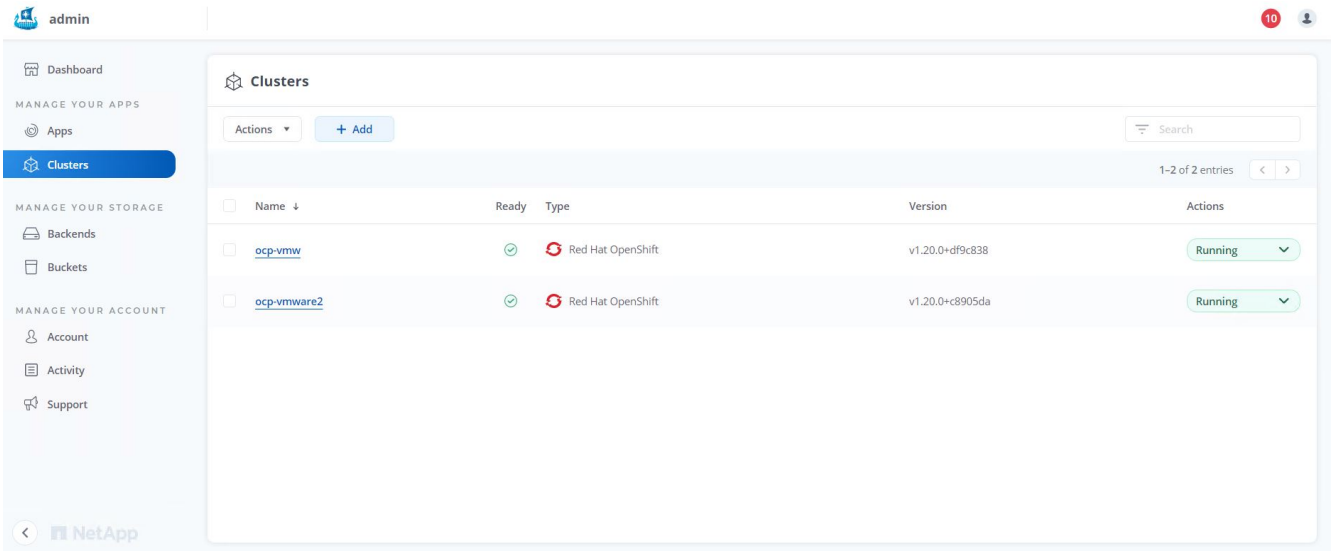


> ユーザ名とパスワードまたはトークンを使用して認証するために kubeconfig ファイルを生成できます。トークンが期限切れになるまでの時間は制限されており、登録されたクラスタに到達できなくなる可能性があります。ネットアップでは、 OpenShift クラスタを Astra Control Center に登録するために、ユーザ名とパスワードを付けた kubeconfig ファイルを使用することを推奨します。

2. Astra Control Center で、対象となるストレージクラスが検出される。次に、ストレージクラスが NetApp ONTAP 上の SVM がサポートする Trident を使用してボリュームをプロビジョニングする方法を選択し、Review （確認）をクリックします。次のペインで詳細を確認し、 Add Cluster をクリックします。

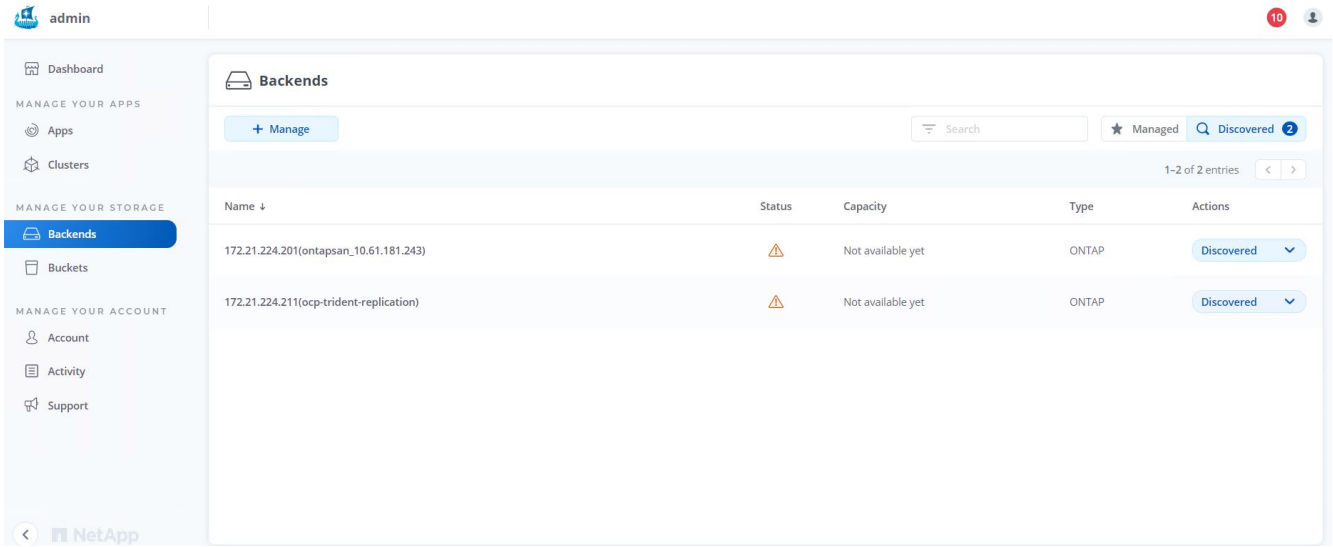


3. 手順 1 の説明に従って、両方の OpenShift クラスタを登録します。追加すると、 Astra Control Center がクラスタを検査して必要なエージェントをインストールしながら、クラスタは Discovering ステータスに移行します。クラスタが登録されると、クラスタのステータスが「 Running 」に変わります。

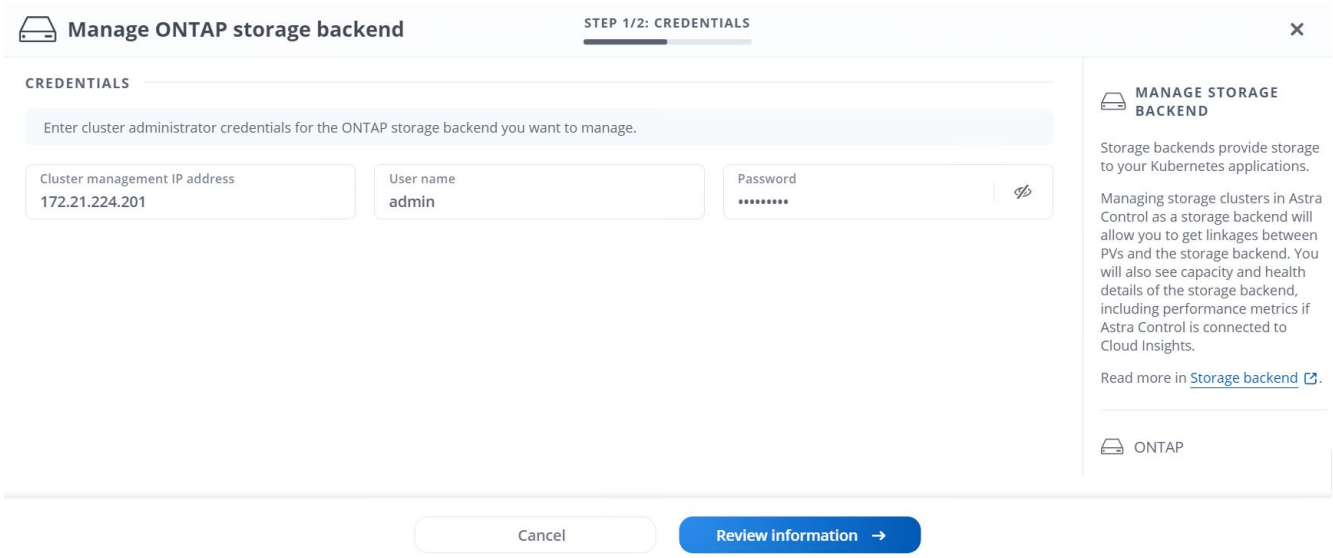


> Astra Control Center で管理するすべての Red Hat OpenShift クラスタは、管理対象クラスタにインストールされたエージェントとしてインストールに使用されたイメージレジストリにアクセスできる必要があります。このレジストリからイメージがプルされます。

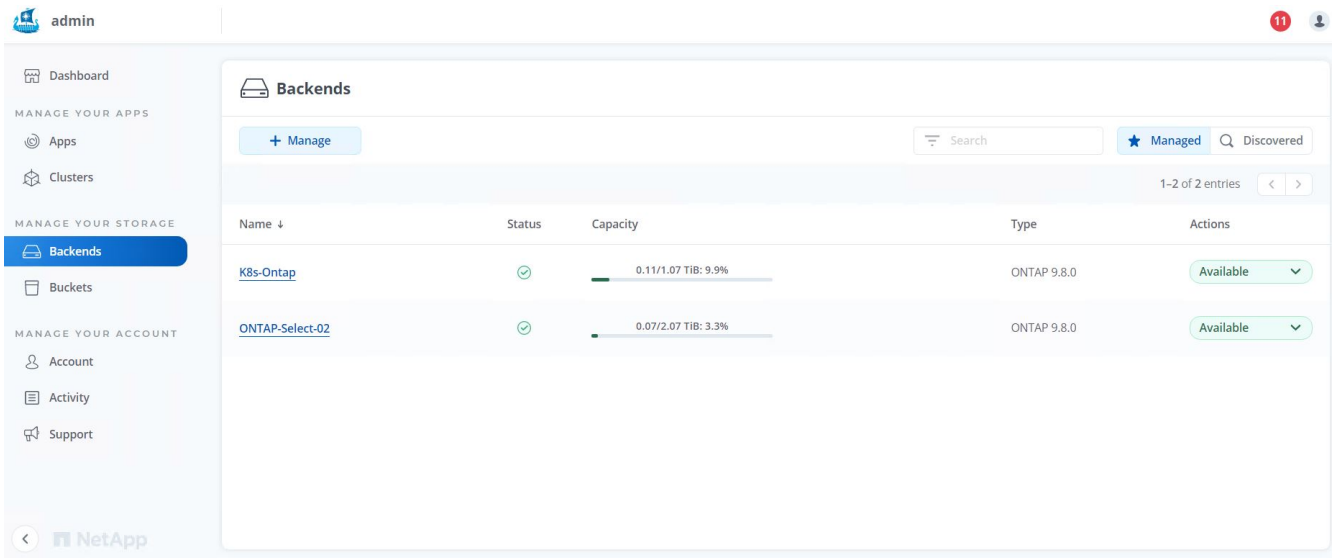
4. ONTAP クラスタをストレージリソースとして Astra Control Center でバックエンドとして管理するようにインポートします。ストレージクラスが設定されている Astra に OpenShift クラスタが追加されると、ストレージクラスをサポートする ONTAP クラスタが自動的に検出されて検査されますが、 Astra コントロールセンターにインポートされて管理されません。



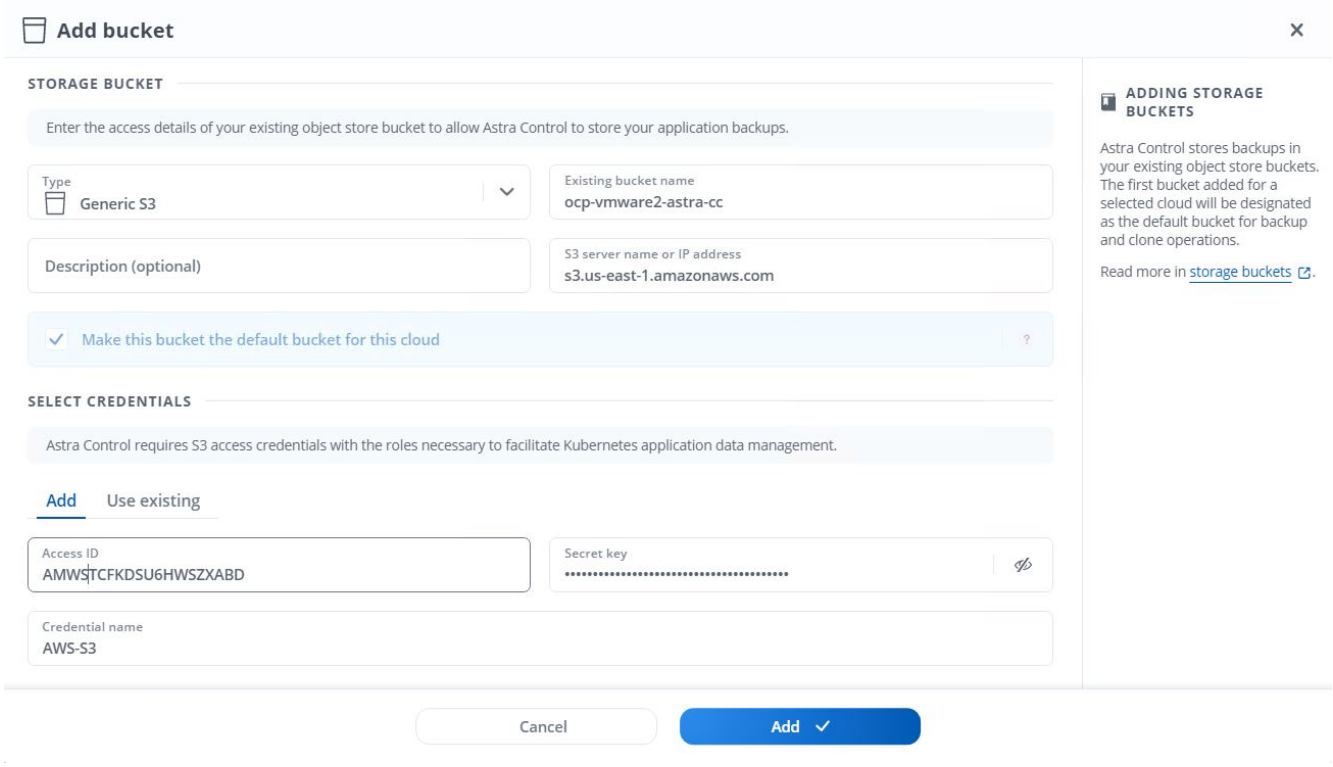
5. ONTAP クラスタをインポートするには、バックエンドに移動し、ドロップダウンをクリックして、管理対象の ONTAP クラスタの横にある Manage を選択します。ONTAP クラスタの資格情報を入力し、 [情報の確認] をクリックして、 [ストレージバックエンドのインポート] をクリックします。



6. バックエンドを追加すると、ステータスが Available に変わります。このバックエンドには、 OpenShift クラスタ内の永続ボリュームと ONTAP システム上の対応するボリュームに関する情報が含まれます。



7. Astra Control Center を使用して OpenShift クラスタ間でバックアップとリストアを行うには、 S3 プロトコルをサポートするオブジェクトストレージバケットをプロビジョニングする必要があります。現在サポートされているオプションは、 ONTAP S3 、 StorageGRID 、および AWS S3 です。このインストールのために、 AWS S3 バケットを設定します。バケットに移動し、バケットの追加をクリックして、汎用 S3 を選択します。S3 バケットの詳細とアクセスするためのクレデンシャルを入力し、「 Make this bucket the default bucket for the cloud 」のチェックボックスをオンにして、 Add をクリックします。

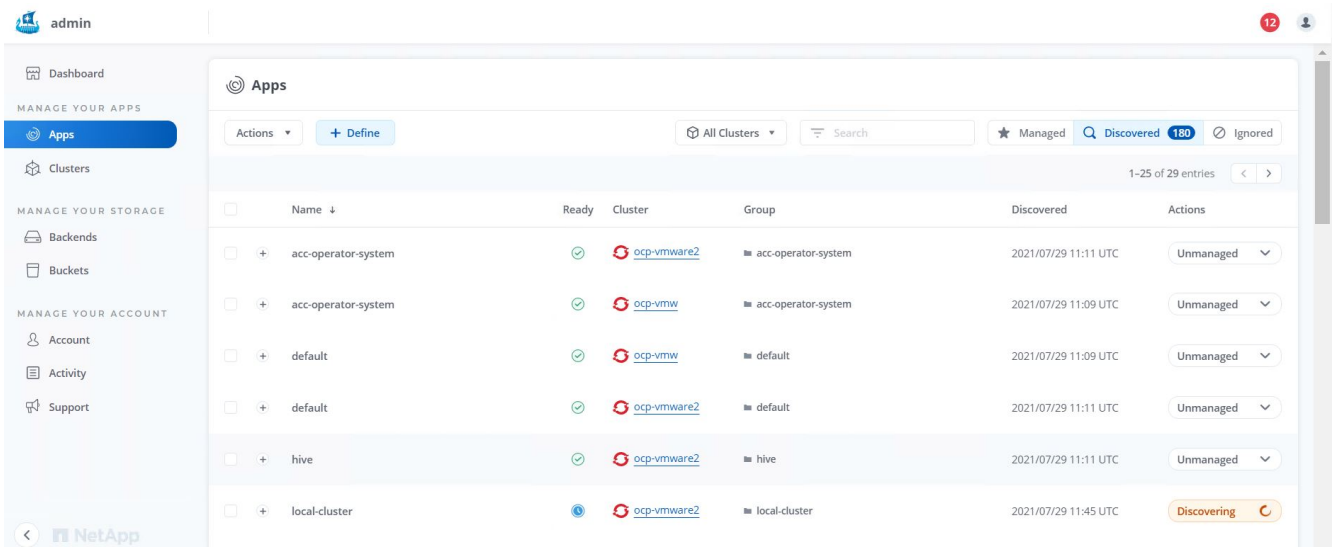


保護するアプリケーションを選択します

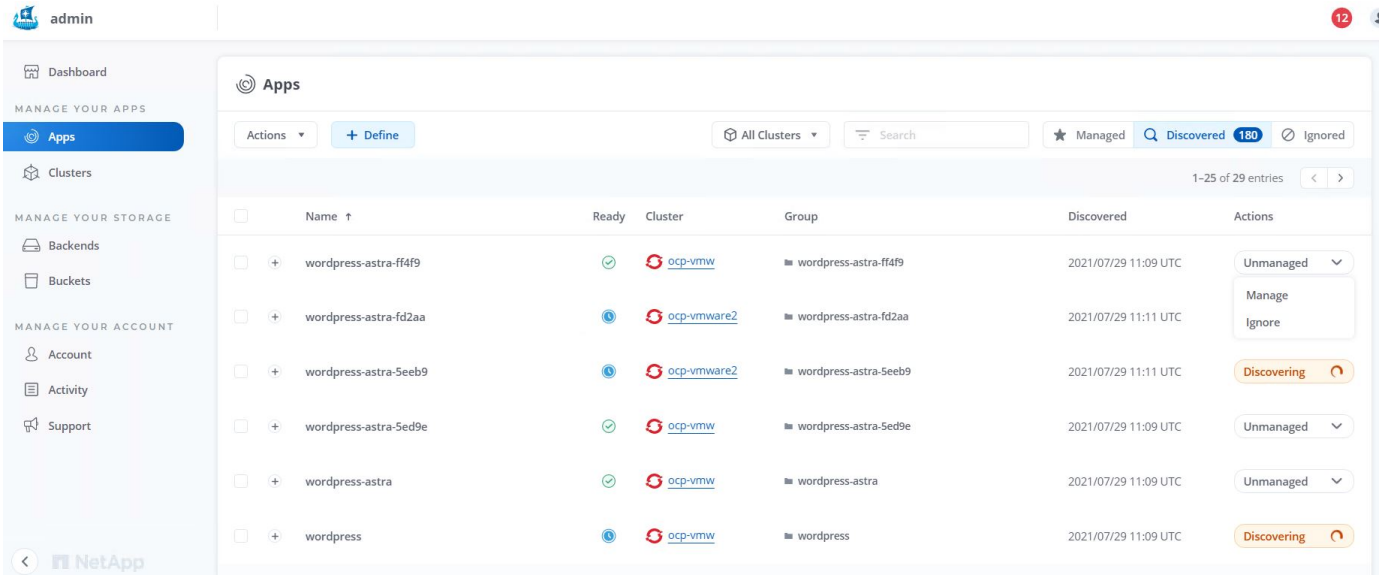
Red Hat OpenShift クラスタを登録したら、 Astra Control Center を使用して導入および管理するアプリケーションを検出できます。

アプリケーションを管理します

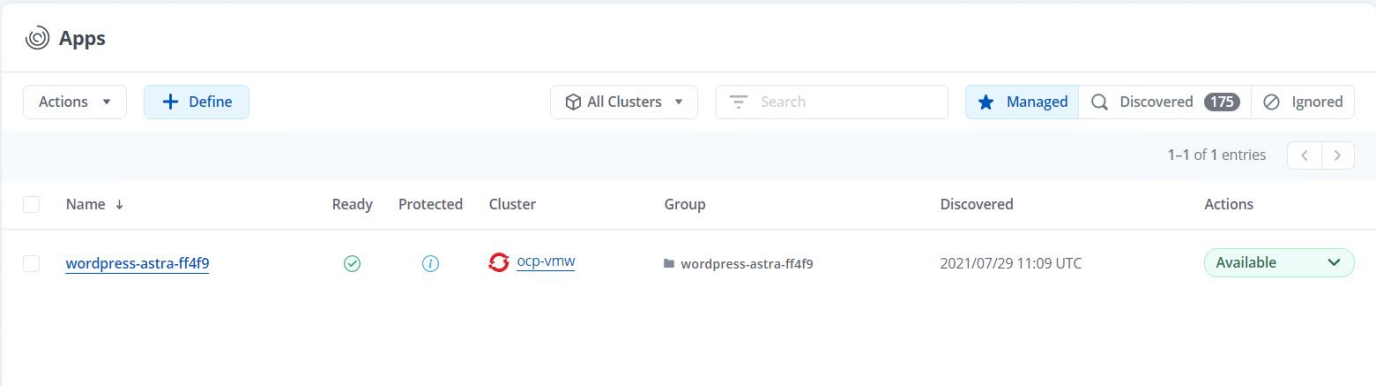
1. OpenShift クラスタと ONTAP バックエンドが Astra Control Center に登録されると、コントロールセンターは、指定した ONTAP バックエンドで構成されたストレージクラスを使用するすべてのネームスペース内のアプリケーションを自動的に検出します。



2. [アプリケーション]>[検出済み] の順に移動し、 Astra を使用して管理するアプリケーションの横にあるドロップダウンメニューをクリックします。[管理] をクリックします。



1. アプリケーションが [使用可能（ Available ）] 状態になり、 [アプリケーション（ Apps ）] セクションの [管理（ Managed ）] タブで表示できます。



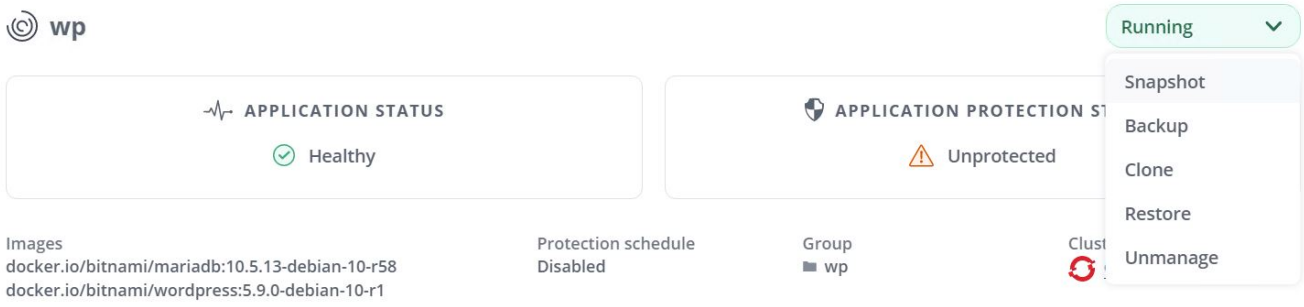
アプリケーションを保護

アプリケーションワークロードを Astra Control Center で管理した後、それらのワークロードの保護設定を構成できます。

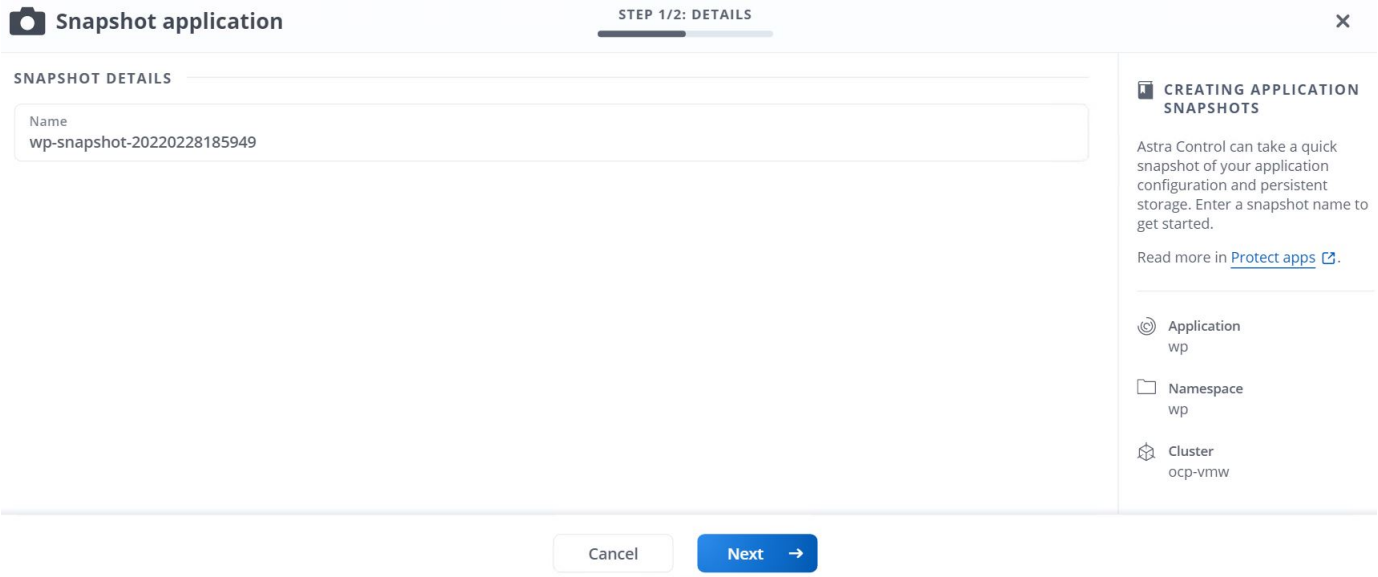
アプリケーションスナップショットを作成しています

アプリケーションの Snapshot コピーを作成すると、 ONTAP Snapshot コピーが作成されます。 Snapshot コピーに基づいて、アプリケーションを特定の時点にリストアまたはクローニングできます。

1. アプリケーションのスナップショットを作成するには、 [アプリ] > [管理] タブに移動し、 Snapshot コピーを作成するアプリケーションをクリックします。アプリケーション名の横にあるドロップダウンメニューをクリックし、 Snapshot をクリックします。



2. スナップショットの詳細を入力し、 [次へ] をクリックして、 [スナップショット] をクリックします。Snapshot の作成には約 1 分かかり、作成が完了するとステータスを確認できるようになります。



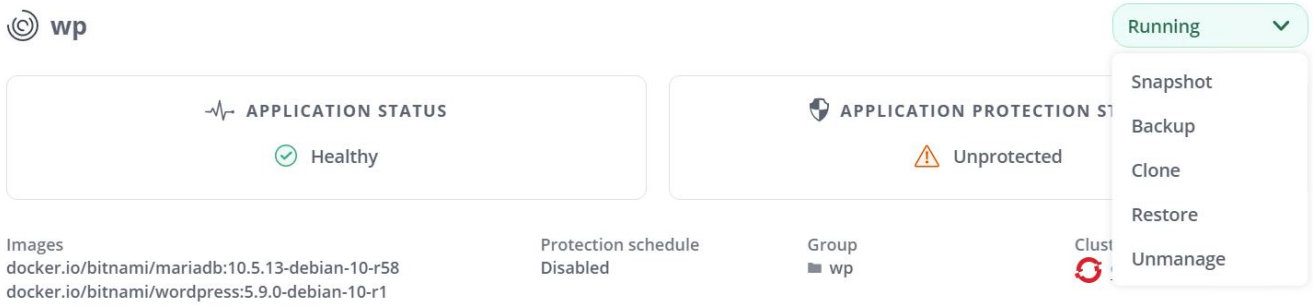
アプリケーションのバックアップを作成しています

アプリケーションのバックアップは、アプリケーションのアクティブな状態とそのリソースの設定をキャプチャしてファイルに変換し、リモートのオブジェクトストレージバケットに格納します。

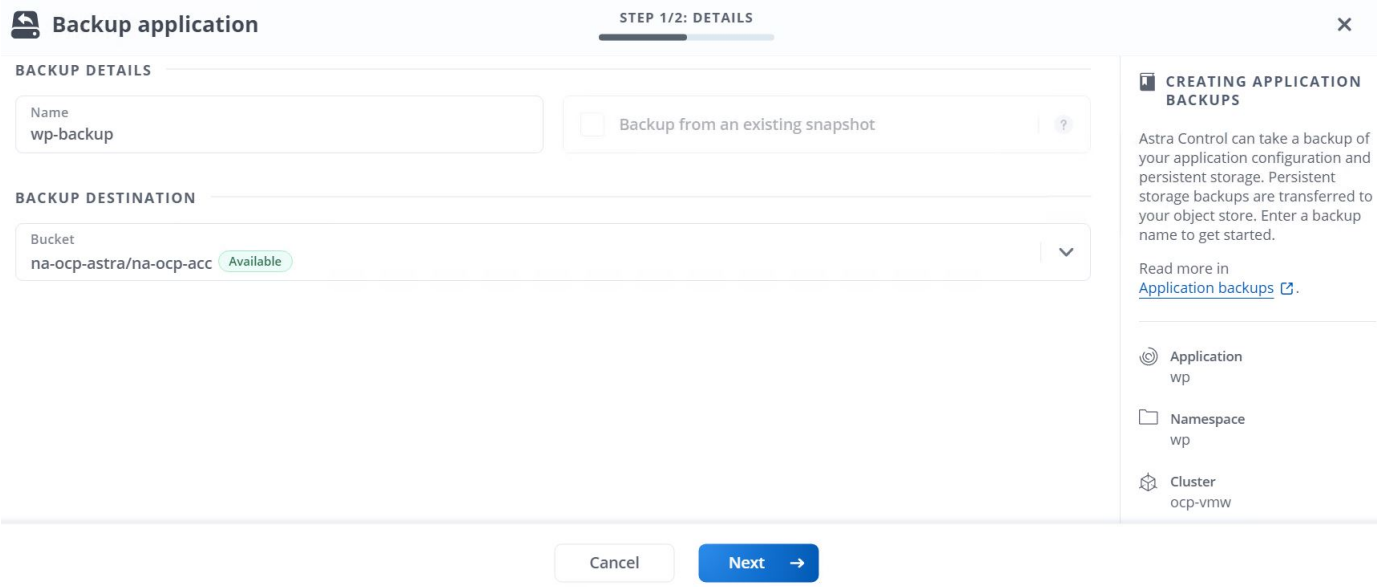
Astra Control Center で管理対象アプリケーションのバックアップとリストアを行うには、バッキング ONTAP システムのスーパーユーザ設定を前提条件として設定する必要があります。そのためには、次のコマンドを入力します。

```
ONTAP::> export-policy rule modify -vserver ocp-trident -policyname
default -ruleindex 1 -superuser sys
ONTAP::> export-policy rule modify -policyname default -ruleindex 1 -anon
65534 -vserver ocp-trident
```

1. Astra Control Center で管理対象アプリケーションのバックアップを作成するには、 [アプリ] > [管理] タブに移動し、バックアップを作成するアプリケーションをクリックします。アプリケーション名の横にあるドロップダウンメニューをクリックし、 [バックアップ] をクリックします。



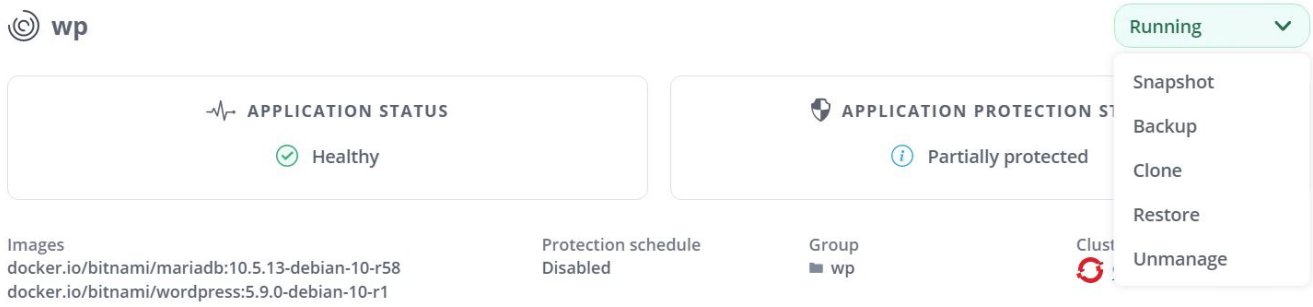
2. バックアップの詳細を入力し、バックアップファイルを保存するオブジェクトストレージバケットを選択して次へをクリックします。詳細を確認したら、バックアップをクリックします。アプリケーションのサイズとデータによっては、バックアップに数分かかることがあり、バックアップが正常に完了したあとでバックアップのステータスを確認できるようになります。



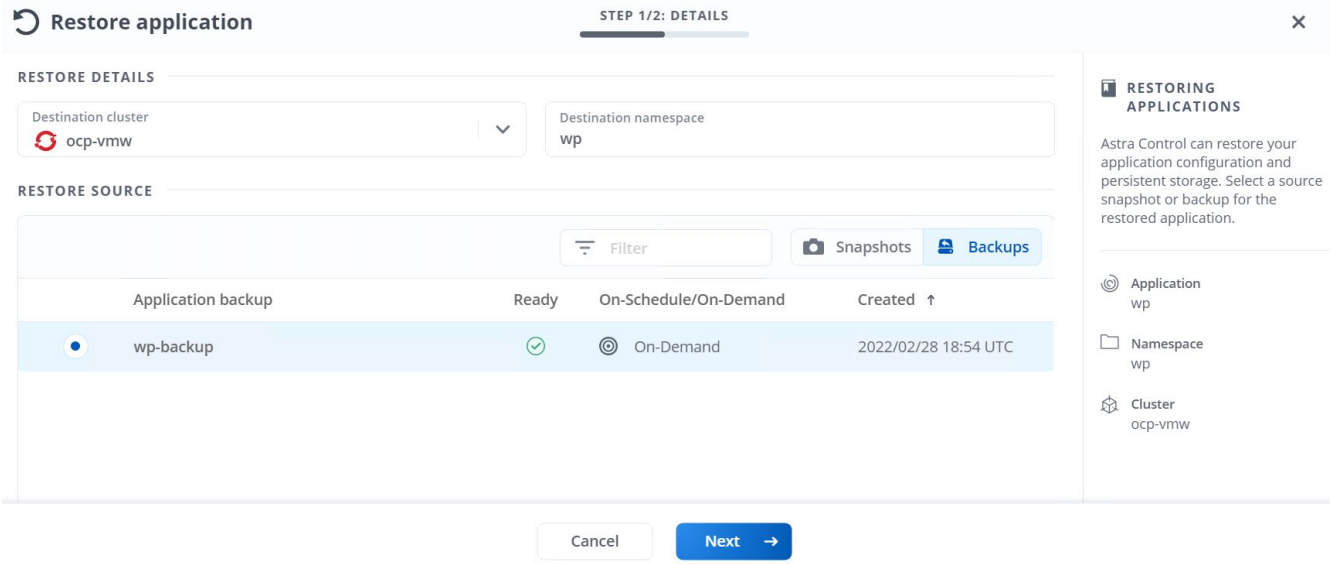
アプリケーションのリストア

ボタンを押すだけで、アプリケーションを同じクラスタ内の元のネームスペースまたはリモートクラスタにリストアし、アプリケーションを保護してディザスタリカバリに使用できます。

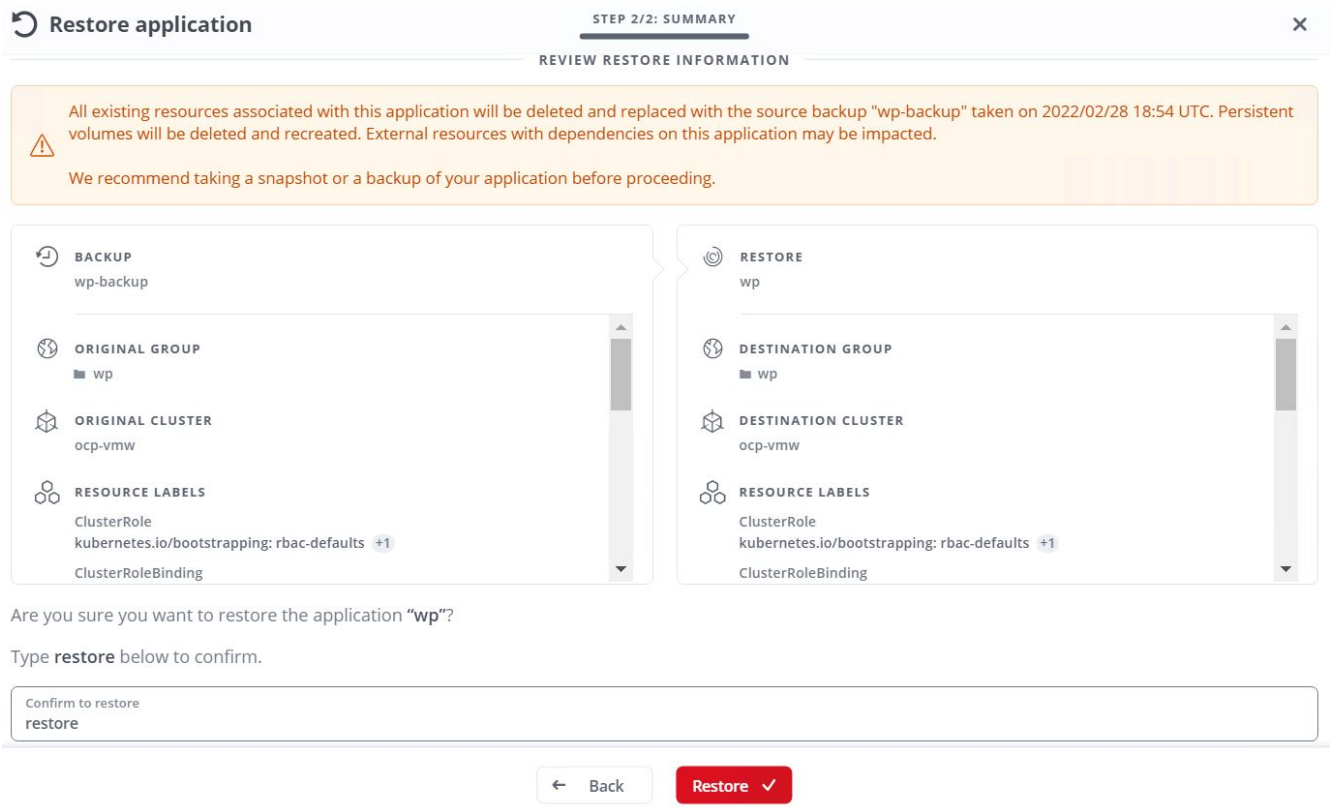
1. アプリケーションを復元するには、 [アプリ] > [管理] タブに移動し、該当するアプリをクリックします。アプリケーション名の横にあるドロップダウン・メニューをクリックし '[リストア] をクリックします



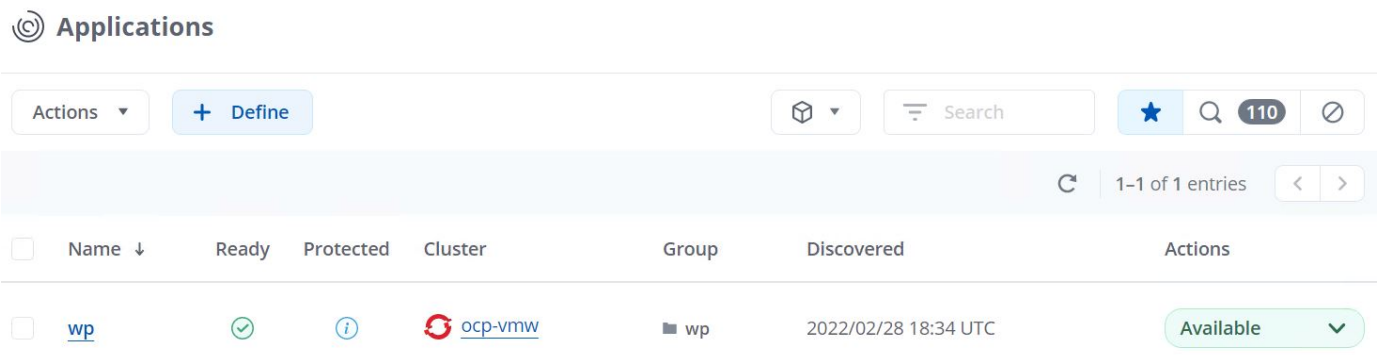
2. リストアネームスペースの名前を入力し、リストア先のクラスタを選択して、既存の Snapshot からリストアするかアプリケーションのバックアップからリストアするかを選択します。次へをクリックします。



3. レビューペインで「 restore 」と入力し、詳細を確認した後で「 Restore 」をクリックします。



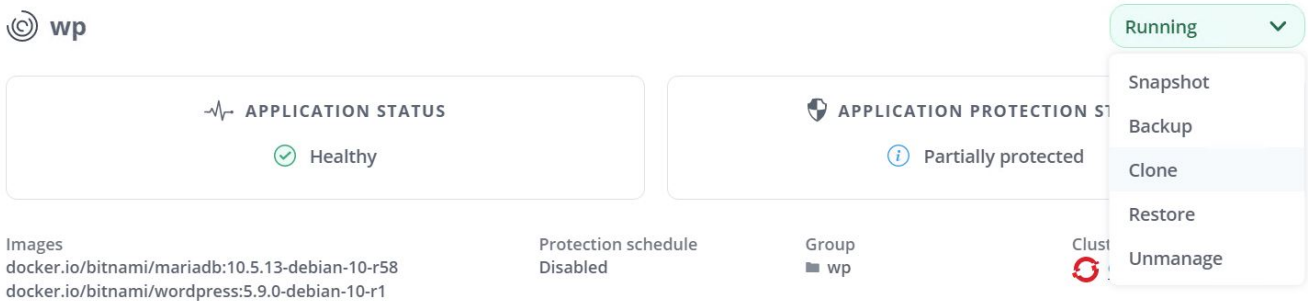
4. 新しいアプリケーションは、 Astra Control Center が選択したクラスタ上のアプリケーションを復元している間、 Restoring 状態になります。アプリケーションのすべてのリソースが Astra によってインストールおよび検出されると、アプリケーションは Available 状態になります。



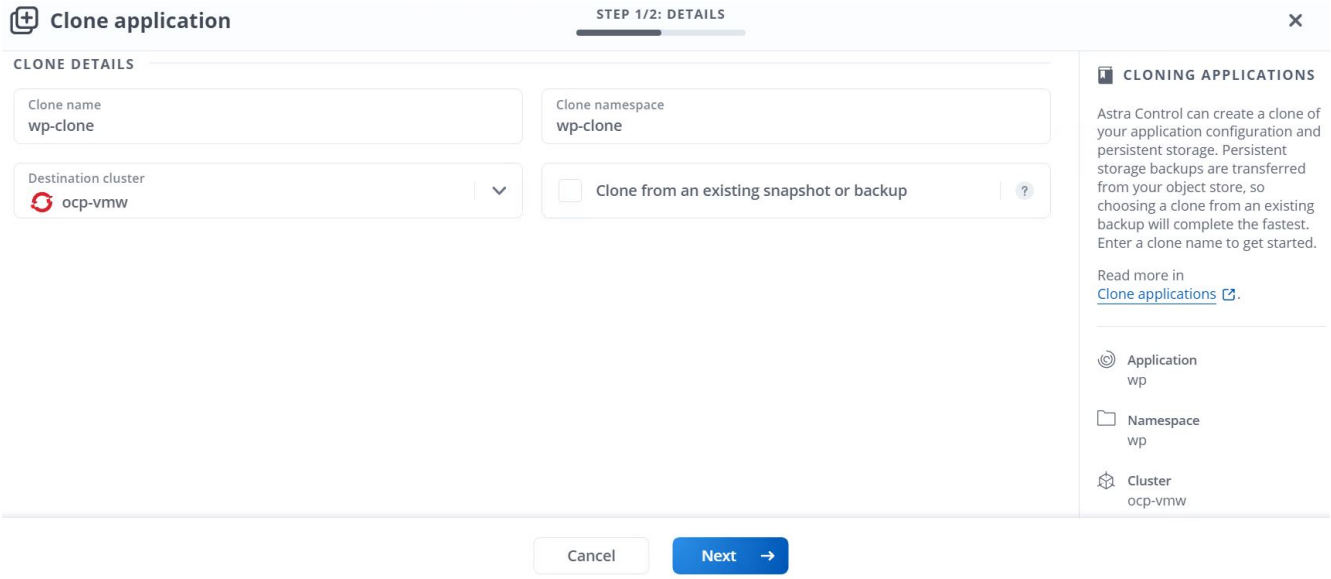
アプリケーションのクローニング

アプリケーションは、開発 / テストやアプリケーションの保護およびディザスタリカバリ目的で、元のクラスタまたはリモートクラスタにクローニングできます。同じストレージバックエンドで同じクラスタ内にあるアプリケーションをクローニングする場合、 NetApp FlexClone テクノロジを使用します。 FlexClone テクノロジを使用すると、 PVC のクローンを瞬時に作成し、ストレージスペースを節約できます。

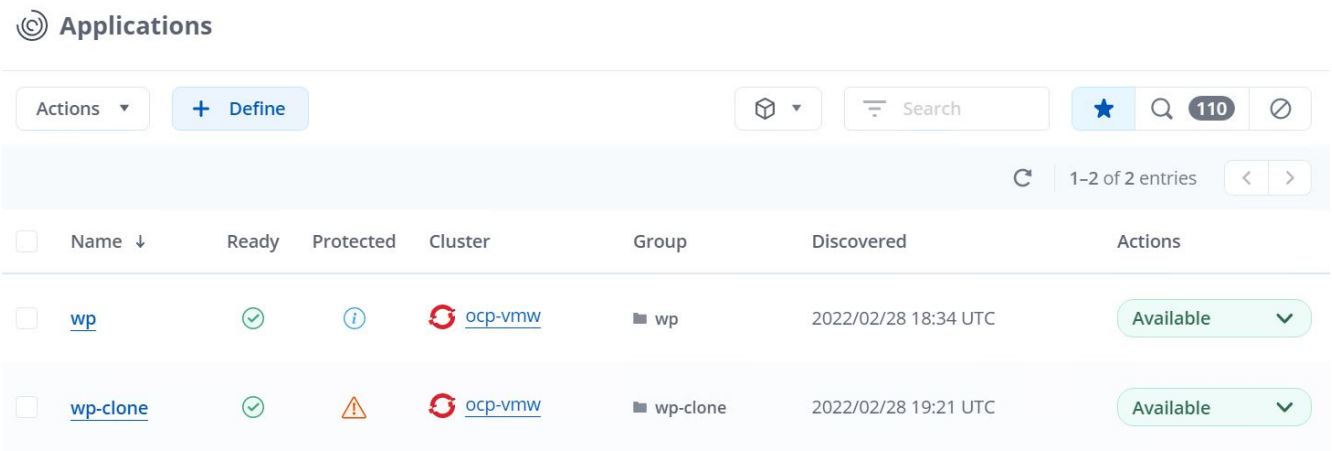
1. アプリケーションをクローンするには、 [アプリケーション（ Apps ）] > [管理（ Managed ）] タブに移動し、該当するアプリケーションをクリックします。アプリケーション名の横にあるドロップダウンメニューをクリックし、 Clone をクリックします。



2. 新しいネームスペースの詳細を入力し、クローニング先のクラスタを選択します。クローンを既存の Snapshot 、バックアップ、またはアプリケーションの現在の状態から作成するかどうかを選択します。詳細を確認したら、 [次へ] をクリックして、 [レビューペインに複製] をクリックします。



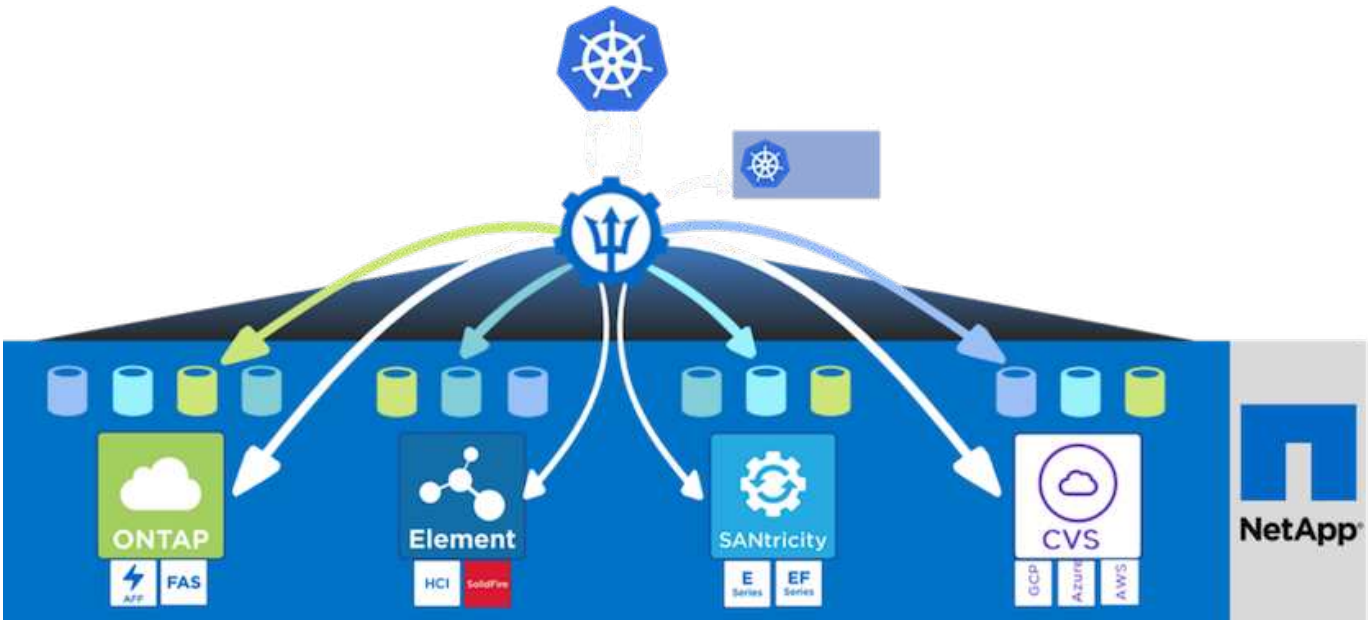
3. 新しいアプリケーションは Discovering 状態になり、 Astra Control Center は選択したクラスタにアプリケーションを作成します。アプリケーションのすべてのリソースが Astra によってインストールおよび検出されると、アプリケーションは Available 状態になります。



Astra Trident の概要

Astra Trident は、コンテナや Kubernetes ディストリビューション向けの、 Red Hat OpenShift など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ションツールです。Trident は、 NetApp ONTAP や Element ストレージシステムを含むネットアップストレージポートフォリオ全体と連携し、 NFS 接続と iSCSI 接続もサポートします。Trident を使用すると、ストレージ管理者の手を煩わせることなく、エンドユーザがネットアップ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して管理できるため、DevOps ワークフローが高速化されます。

管理者は、プロジェクトのニーズやストレージシステムモデルに基づいて複数のストレージバックエンドを構成し、圧縮、特定のディスクタイプ、 QoS レベルなどの高度なストレージ機能を有効にして一定のレベルのパフォーマンスを保証できます。定義されたバックエンドは、プロジェクトの開発者が永続的ボリューム要求（ PVC ）を作成し、永続的ストレージをオンデマンドでコンテナに接続するために使用できます。



Astra Trident は、 Kubernetes と同様、迅速な開発サイクルを 1 年に 4 回リリースしています。

2022 年 1 月にリリースされた最新バージョンの Astra Trident は 22.01 です。Trident のどのバージョンがサポートされているかを確認できます Kubernetes ディストリビューションのテストに使用 "こちらをご覧ください"。

20.04 リリース以降、 Trident のセットアップは Trident オペレータによって実行されます。オペレータが大規模な導入を容易にし、 Trident インストールの一部として導入されたポッドの自己修復などの追加サポートを提供します。

21.01 リリースでは、 Trident Operator のインストールを容易にするために Helm チャートを使用できるようになりました。

Astra Trident をダウンロード

導入したユーザクラスタに 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、内容を展開します。Trident の最新バージョンは 22.01 で、ダウンロードできます "こちらをご覧ください"。

```
[netapp-user@rhel7 ~]$ wget
https://github.com/NetApp/trident/releases/download/v22.01.0/trident-
installer-22.01.0.tar.gz
--2021-05-06 15:17:30--
https://github.com/NetApp/trident/releases/download/v22.01.0/trident-
installer-22.01.0.tar.gz
Resolving github.com (github.com)... 140.82.114.3
Connecting to github.com (github.com)|140.82.114.3|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://github-
releases.githubusercontent.com/77179634/a4fa9f00-a9f2-11eb-9053-
98e8e573d4ae?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-
Credential=AKIAIWNJYAX4CSVEH53A%2F20210506%2Fus-east-
1%2Fs3%2Faws4_request&X-Amz-Date=20210506T191643Z&X-Amz-Expires=300&X-
Amz-
Signature=8a49a2a1e08c147d1ddd8149ce45a5714f9853fee19bb1c507989b9543eb36
30&X-Amz-
SignedHeaders=host&actor_id=0&key_id=0&repo_id=77179634&response-
content-disposition=attachment%3B%20filename%3Dtrident-installer-
22.01.0.tar.gz&response-content-type=application%2Foctet-stream
[following]
--2021-05-06 15:17:30--  https://github-
releases.githubusercontent.com/77179634/a4fa9f00-a9f2-11eb-9053-
98e8e573d4ae?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-
Credential=AKIAIWNJYAX4CSVEH53A%2F20210506%2Fus-east-
1%2Fs3%2Faws4_request&X-Amz-Date=20210506T191643Z&X-Amz-Expires=300&X-
Amz-
Signature=8a49a2a1e08c147d1ddd8149ce45a5714f9853fee19bb1c507989b9543eb36
30&X-Amz-
```

```
SignedHeaders=host&actor_id=0&key_id=0&repo_id=77179634&response-
content-disposition=attachment%3B%20filename%3Dtrident-installer-
22.01.0.tar.gz&response-content-type=application%2Foctet-stream
Resolving github-releases.githubusercontent.com (github-
releases.githubusercontent.com)... 185.199.108.154, 185.199.109.154,
185.199.110.154, ...
Connecting to github-releases.githubusercontent.com (github-
releases.githubusercontent.com)|185.199.108.154|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 38349341 (37M) [application/octet-stream]
Saving to: 'trident-installer-22.01.0.tar.gz'

100%[=================================================================
===============================================>] 38,349,341  88.5MB/s
in 0.4s

2021-05-06 15:17:30 (88.5 MB/s) - 'trident-installer-22.01.0.tar.gz'
saved [38349341/38349341]
```

2. ダウンロードしたバンドルから Trident のインストールを解凍します。

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

Helm を使用して **Trident Operator** をインストールします

1. Trident にはこのファイルを渡すオプションがないため、まず、ユーザクラスタの「 kubeconfig 」ファイルの場所を環境変数として設定します。

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

2. Helm コマンドを実行し、ユーザークラスタに trident 名前空間を作成しながら、 Helball ディレクトリ内の tarball から Trident 演算子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. Trident が正しくインストールされていることを確認するには、ネームスペース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します。

```
[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、お客様の環境で Trident の導入のカスタマイズが必要になることもあります。このような場合は、 Trident のオペレータを手動でインストールし、含まれているマニフェストを更新して配置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します

1. Trident にはこのファイルを渡すオプションがないため、まず、ユーザクラスタの「 kubeconfig 」ファイルの場所を環境変数として設定します。

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

2. '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を作成します。

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

3. 存在しない場合は、指定されたマニフェストを使用して、クラスタ内に Trident ネームスペースを作成します。

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

4. トライデントオペレータの配備に必要なリソースを作成しますたとえば ' オペレータ用のサービスアカウ

ント 'ClusterRole' および 'ClusterRoleBind' を 'ServiceAccount' 専用の 'PodSecurityPolicy' またはオペレータ自体に割り当てます

```
[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. 次のコマンドを使用すると、展開後にオペレータ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. オペレータが導入したら、 Trident をインストールできます。これには 'TridentOrchestrator を作成する必要があります

```
[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. Trident が正しくインストールされていることを確認するには、ネームスペース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します。

```
[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          |
+----------------+----------------+
```

ワーカーノードをストレージ用に準備する

NFS

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

ただし NFSv3 については、クライアントとサーバ間の同時処理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 ' サーバが接続のウィンドウ・サイズを小さくしなくても NFS 接続の最適なパフォーマンス

を確保できるように ' クライアント側 sunrpc スロット・テーブル・エントリーの最大数は ' サーバ上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 でサポートされる sunrpcslot table エントリの最大数は 128 です。つまり、 ONTAP は、一度に 128 個の NFS 要求を同時に処理できます。ただし、 Red Hat CoreOS / Red Hat Enterprise Linux では、接続ごとに最大 65 、 536 の sunrpc スロットテーブルエントリがデフォルトでサポートされます。この値を 128 に設定する必要があります。これは OpenShift のマシン構成オペレータ（ MCO ）を使用して実行できます。

OpenShift ワーカーノードで最大 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、 [計算]>[マシン構成] 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 が作成されたら、すべてのワーカーノードに設定を適用し、 1 つずつ再起動する必要があります。プロセス全体には約 20~30 分かかります。「 OC GET MCP 」を使用してマシン構成が適用されているかどうかを確認し、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 プロトコルによるブロックストレージボリュームのマッピングを許可す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、 MCO （マシン構成オペレータ）を展開後にクラスタに適用することによって処理されます。

ワーカーノードで iSCSI サービスを実行するように設定するには、次の手順を実行します。

1. OCP Web コンソールにログインし、 [計算]>[マシン構成] 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. 構成の作成後、約 20~30 分で設定がワーカーノードに適用され、再ロードされます。「 OC GET MCP 」を使用してマシン構成が適用されているかどうかを確認し、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（マルチパスを使用している場合、 multipathd サービスが実行されている）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、適切なフラグを指定して「 OC debug 」コマンドを実行することにより、MachineConfig が正常に適用され、サービスが正常に開始されたことを確認することもできます。

ストレージシステムバックエンドを作成

Astra Trident Operator のインストールが完了したら、使用するネットアップストレージプラットフォームに合わせてバックエンドを設定する必要があります。Astra Trident のセットアップと設定を続行するには、次のリンクを参照してください。

- "NetApp ONTAP NFS"
- "NetApp ONTAP iSCSI の略"
- "NetApp Element iSCSI の略"

NetApp ONTAP の NFS 構成

Trident を NetApp ONTAP ストレージシステムと統合するには、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のサンプルバックエンドファイルは、「 sample -input 」フォルダ階層にあります。NFS を提供している 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コピーし、ファイルを編集します。

```
[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. backendName 、 managementLIF 、 dataLIF 、 SVM 、ユーザ名を編集します。 パスワードの値を入力します。

```
{
    "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"
}
```

カスタムの backendName 値は、簡単に識別できるように NFS を提供するストレージ DriverName とデータ LIF を組み合わせて定義することを推奨します。

3. このバックエンドファイルを設定した状態で、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 フォルダにある環境用に編集可能なサンプルのストレージクラスファイルがあります。作業ディレクトリにコピーし、作成し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. このファイルに対して行う必要がある唯一の編集は ' バックエンドタイプの値を ' 新しく作成されたバックエンドのストレージドライバの名前に定義することですまた、名前フィールドの値もメモしておきます。この値は、以降の手順で参照する必要があります。

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

> このファイルに定義されているオプションのフィールド「 fsType 」があります。この行は NFS バックエンドで削除できます。

6. oc` コマンドを実行して ' ストレージ・クラスを作成します

```
[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、最初の永続的ボリューム要求（ PVC ）を作成する必要があります。sample_inputs にもあるこのアクションを実行するために使用できるサンプルの 'pvc-basicy.yaml ファイルがあります

```
[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 ''torageClassName` フィールドが作成したものと一致することを確認することですプロビジョニングするワークロードによって必要に応じて、 PVC 定義を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. 「 OC 」コマンドを発行して、 PVC を作成します。作成中の元のボリュームのサイズによっては作成にしばらく時間がかかることがあるため、作成が完了した時点でこのプロセスを監視できます。

```
[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 を NetApp ONTAP ストレージシステムと統合するには、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のサンプルバックエンドファイルは、「 sample -input 」フォルダ階層にあります。iSCSI を提供している 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コピーし、ファイルを編集します。

```
[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. このファイルで管理 LIF 、データ LIF 、 SVM 、ユーザ名、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設定した状態で、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 フォルダにある環境用に編集可能なサンプルのストレージクラスファイルがあります。作業ディレクトリにコピーし、作成し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. このファイルに対して行う必要がある唯一の編集は ' バックエンドタイプの値を ' 新しく作成されたバックエンドのストレージドライバの名前に定義することですまた、名前フィールドの値もメモしておきます。この値は、以降の手順で参照する必要があります。

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

このファイルに定義されているオプションのフィールド「 fsType 」があります。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、 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ために削除することもできます。

6. oc` コマンドを実行して ' ストレージ・クラスを作成します

```
[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、最初の永続的ボリューム要求（ PVC ）を作成する必要があります。sample_inputs にもあるこのアクションを実行するために使用できるサンプルの 'pvc-basicy.yaml ファイルがあります

```
[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 ''torageClassName` フィールドが作成したものと一致することを確認することですプロビジョニングするワークロードによって必要に応じて、 PVC 定義を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. 「 OC 」コマンドを発行して、 PVC を作成します。作成中の元のボリュームのサイズによっては作成にしばらく時間がかかることがあるため、作成が完了した時点でこのプロセスを監視できます。

```
[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 を NetApp Element ストレージシステムと統合するには、 iSCSI プロトコルを使用してストレージシステムと通信でき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のサンプルバックエンドファイルは、「 sample -input 」フォルダ階層にあります。iSCSI を提供している NetApp Element システムの場合、「 backend-solidfire.json 」ファイルを作業ディレクトリにコピーし、ファイルを編集します。

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

   a. エンドポイント行のユーザ、パスワード、および MVIP 値を編集します。
   b. 「仮想 IP 」の値を編集します。

```
 {
    "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設定したら、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 フォルダにある環境用に編集可能なサンプルのストレージクラスファイルがあります。作業ディレクトリにコピーし、作成し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. このファイルに対して行う必要がある唯一の編集は ' バックエンドタイプの値を ' 新しく作成されたバックエンドのストレージドライバの名前に定義することですまた、名前フィールドの値もメモしておきます。この値は、以降の手順で参照する必要があります。

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

> このファイルに定義されているオプションのフィールド「 fsType 」があります。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、 OpenShift で使用するファイルシステムを決定できるようにするためにこの値を削除できます。

5. oc` コマンドを実行して ' ストレージ・クラスを作成します

```
[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、最初の永続的ボリューム要求（ PVC ）を作成する必要があります。sample_inputs にもあるこのアクションを実行するために使用できるサンプルの 'pvc-basicy.yaml ファイルがあります

```
[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 ''torageClassName` フィールドが作成したものと一致することを確認することですプロビジョニングするワークロードによって必要に応じて、 PVC 定義を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. 「 OC 」コマンドを発行して、 PVC を作成します。作成中の元のボリュームのサイズによっては作成にしばらく時間がかかることがあるため、作成が完了した時点でこのプロセスを監視できます。

```
[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
```

Advanced Configuration Options （詳細設定オプション）

ロードバランサのオプションの確認：ネットアップを使用した Red Hat OpenShift

ほとんどの場合、 Red Hat OpenShift は、ルートを介してアプリケーションを外部で利用できるようにします。サービスは、外部からアクセス可能なホスト名を付与することで公開されます。定義されたルートおよびサービスによって識別されるエンドポイントは、 OpenShift ルータによって使用され、外部クライアントにこの名前付き接続を提供できます。

ただし、アプリケーションでは、適切なサービスを公開するために、カスタマイズしたロードバランサの導入と設定が必要になる場合があります。その一例が、ネットアップアストラコントロールセンターです。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オプションを評価しました。このセクションでは、これらのインストールと設定について説明します。

以下のページでは、解決策追加情報を搭載した Red Hat OpenShift で検証済みのロードバランサオプションについて説明します。

- "MetalLB"
- "F5 BIG-IP"

MetalLB ロードバランサのインストール：ネットアップでの Red Hat OpenShift

このページでは、 MetalLB ロードバランサのインストールおよび設定手順を示します。

MetalLB は、 OpenShift クラスタにインストールされた自己ホスト型ネットワークロードバランサであり、クラウドプロバイダで実行されないクラスタでタイプロードバランサの OpenShift サービスを作成できます。LoadBalancer サービスをサポートするために連携する 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メントです。

MetalLB 設定オプション

MetalLB が OpenShift クラスタの外部でロードバランササービスに割り当てられた IP アドレスをどのようにアナウンスするかに基づいて、次の 2 つのモードで動作します。

- * レイヤ 2 モード。 * このモードでは、 OpenShift クラスタ内の 1 つのノードがサービスの所有権を取得し、その IP の ARP 要求に応答して、 OpenShift クラスタ外で到達可能にします。IP をアドバタイズするのはノードだけなので、帯域幅のボトルネックと低速フェールオーバーの制限があります。詳細については、のドキュメントを参照してください "こちらをご覧ください"。
- * このモードでは、 OpenShift クラスタ内のすべてのノードがルータとの 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 に転送するために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と統合することです。BGP のハッシュメカニズムにより、サービスの IP-to-Node マッピングが変更されることがあります。詳細については、のドキュメントを参照してください "こちらをご覧ください"。

> このマニュアルでは、レイヤ 2 モードで MetalLB を設定します。

MetalLB ロードバランサをインストールします

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「 metallb.yaml 」を編集し、「 pec.template.spec.securityContext` 」をコントローラ展開とスピーカー DemonSet から削除します。

   - 削除する行数： *

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

3. 「 metallb-system' 」ネームスペースを作成します。

```
[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 スピーカを設定する前に、スピーカ DemonSet の昇格特権を与えて、ロードバランサを動作させるために必要なネットワーク設定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 - システム」ネームスペースに「 ConfigMap 」を作成して、 MetalLB を設定します。

```
[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. これで、ロードバランササービスが作成されると、 MetalLB は外部 IP をサービスに割り当て、 ARP 要求に応答して IP アドレスをアドバタイズします。

> BGP モードで MetalLB を設定する場合は、上記の手順 6 を省略し、 MetalLB マニュアルの手順に従います "こちらをご覧ください"。

F5 BIG-IP ロードバランサのインストール

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、高度な運用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幅広く提供する Application Delivery Controller （ADC; アプリケーションデリバリコントローラ）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、またはオンプレミスの仮想アプライアンスに、さまざまな方法で導入、使用できます。要件に応じて F5 BIG-IP を調査し、導入するには、ここで説明しているドキュメントを参照してください。

F5 BIG-IP サービスを Red Hat OpenShift と効率的に統合するために、 F5 は BIG-IP Container Ingress

Service （ CIS ）を提供します。CI 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ポッドとしてインストールされます。F5 BIG-IP CIS は、OpenShift でサービスタイプ Loadancers とルートを制御するように構成できます。

さらに、タイプ LoadBalancer にサービスを提供するための自動 IP アドレス割り当てには、 F5 IPAM コントローラを使用できます。F5 IPAM コントローラは、 LoadBalancer サービスの OpenShift API を ipamLabel 注釈で監視し、事前構成済みプールから IP アドレスを割り当てるコントローラ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設定手順がリストされています。前提条件として、 F5 BIG-IP システムを導入し、ライセンスを取得しておく必要があります。また、デフォルトでは BIG-IP VE 基本ライセンスに含まれている SDN サービスのライセンスも必要です。

> F5 BIG-IP は、スタンドアロンモードまたはクラスタモードで導入できます。この検証の目的上、 F5 BIG-IP はスタンドアロンモードで導入されましたが、本番環境では、単一点障害を避けるために、大量の IP で構成されたクラスタを使用することを推奨します。

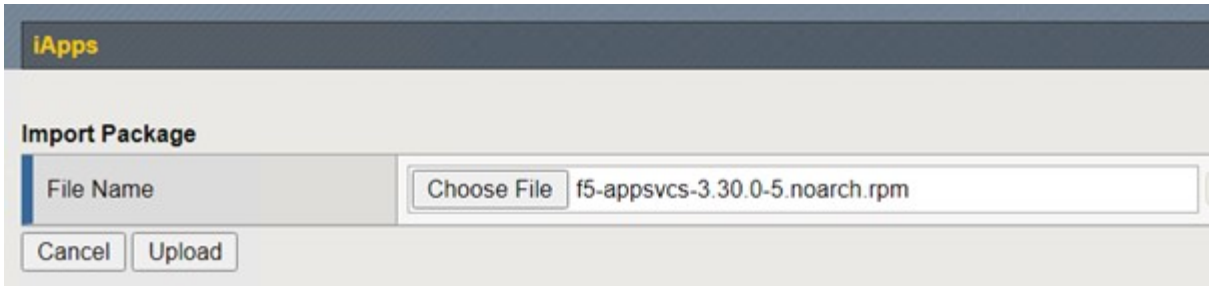
> F5 BIG-IP システムは、専用のハードウェア、クラウド、またはオンプレミスの仮想アプライアンスとして、バージョンが 12.x よりも大きいオンプレミスに導入でき、 F5 CIS と統合できます。このドキュメントでは、 BIG-IP VE エディションなどを使用して、 F5 BIG-IP システムを仮想アプライアンスとして検証しました。

検証済みのリリース

テクノロジ	ソフトウェアのバージョン
Red Hat OpenShift のサービスです	4.6 EUS 、 4.7
F5 BIG-IP VE エディション	16.1.0
F5 Container Ingress Service の略	2.5.1
F5 IPAM コントローラ	0.1.4
F5 AS3	3.30.0

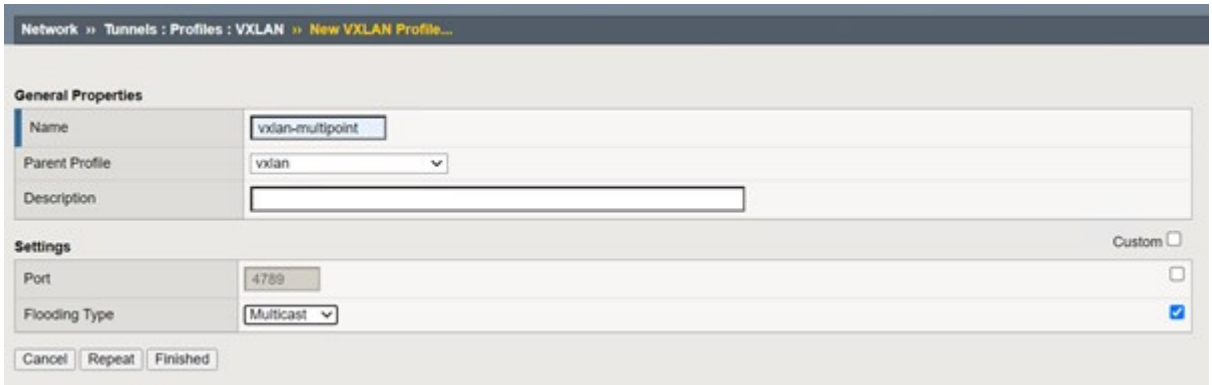
インストール

1. F5 Application Services 3 拡張機能をインストールして、 big-IP システムが命令コマンドではなく JSON で構成を受け入れるようにします。に進みます "F5 AS3 GitHub リポジトリ"をクリックし、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、 Import （インポート）をクリックします。
3. [ファイルの選択] をクリックして、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。

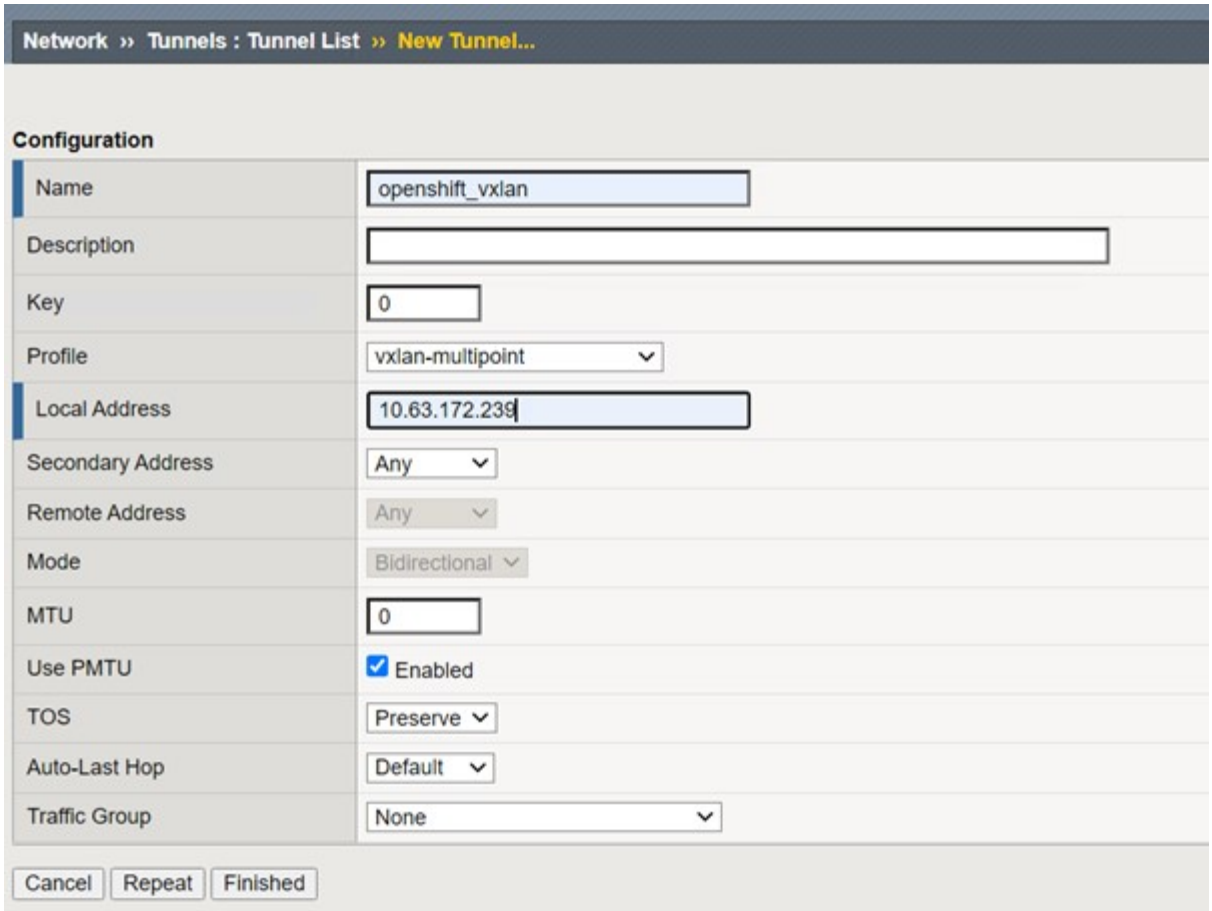


4. AS3 拡張機能が正常にインストールされたことを確認します。

5. 次に、 OpenShift システムと 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 のための BIG-IP システムに VXLAN トンネルインターフェイスを作成し、 OpenShift と BIG-IP サーバ間にトンネルを作成します。Network > Tunnels > Profiles と進み、 Create をクリックして Parent Profile を VXLAN に設定し、フラッディング Type を Multicast 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. Network > Tunnels > Tunnel List と進み、 Create 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. クラスタ管理者権限で Red Hat OpenShift クラスタにログインします。
8. F5 BIG-IP サーバの OpenShift にホストサブネットを作成します。このサブネットは、 OpenShift クラスタから F5 BIG-IP サーバに拡張します。ホストサブネット 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
```

ご使用の環境に応じて、 hostIP などの詳細情報を変更します。

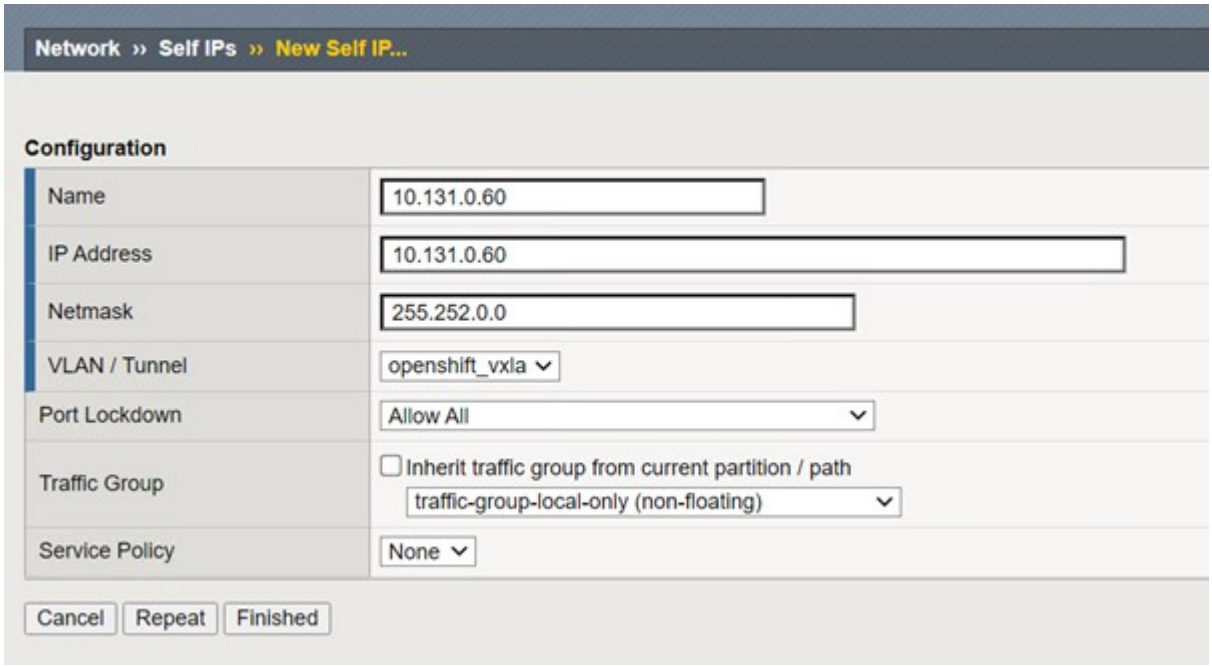
10. HostSubnet リソースを作成します。

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

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

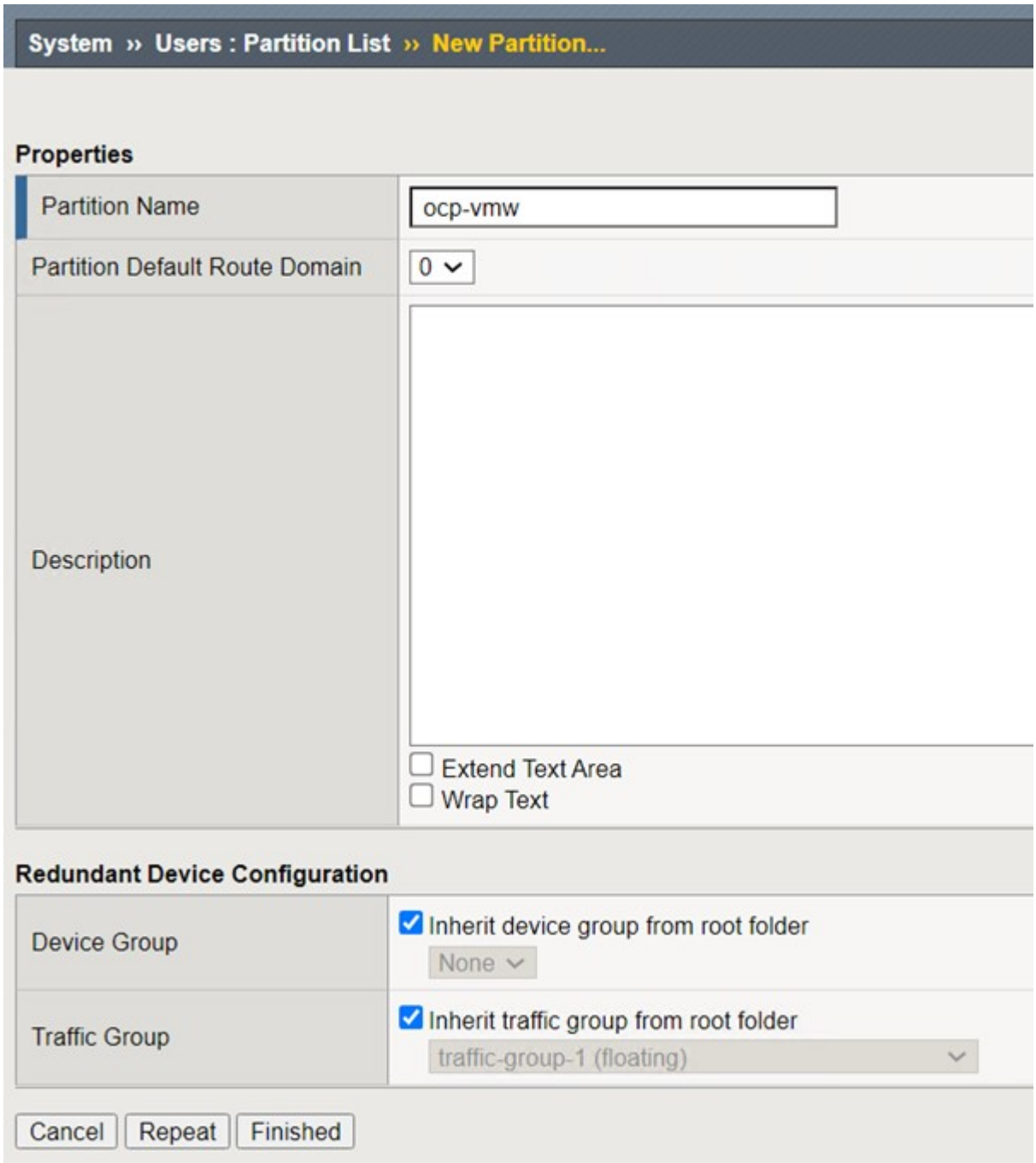
11. F5 BIG-IP サーバ用に作成されたホストサブネットのクラスタ IP サブネット範囲を取得します。

```
[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 サーバに対応する OpenShift のホストサブネット範囲の IP を使用して、 VXLAN OpenShift 上に自己 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]>[自己 IP] の順に選択し、 [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 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。[完了] をクリックします。



13. CIS で設定および使用する F5 BIG-IP システムにパーティションを作成します。[システム]>[ユーザ]>[パーティションリスト] の順に選択し、 [作成] をクリックして詳細を入力します。[完了] をクリックします。



CIS で管理されるパーティションでは手動で設定しないことをお勧めします。

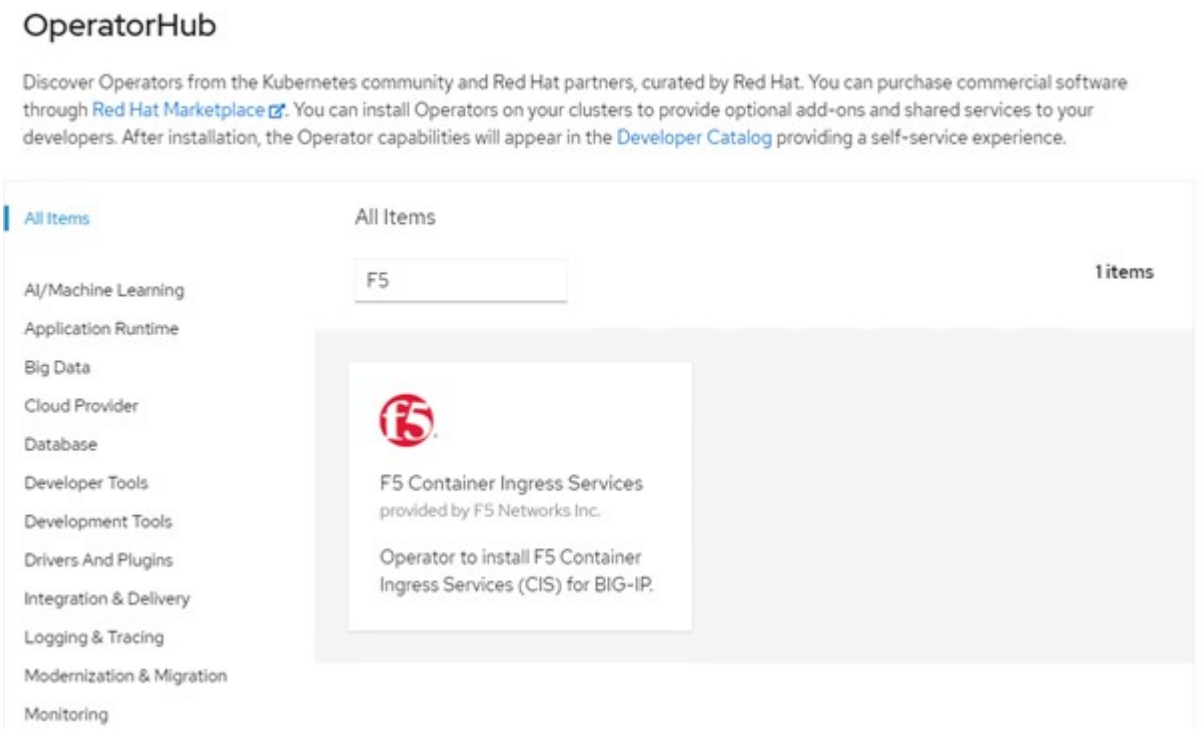
14. OperatorHub のオペレータを使用して F5 BIG-IP CIS をインストールします。cluster-admin 権限を持つ Red Hat OpenShift クラスタにログインし、 F5 BIG-IP システムログインクレデンシャルを使用してシークレットを作成します。これはオペレータの前提条件です。

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

secret/bigip-login created
```

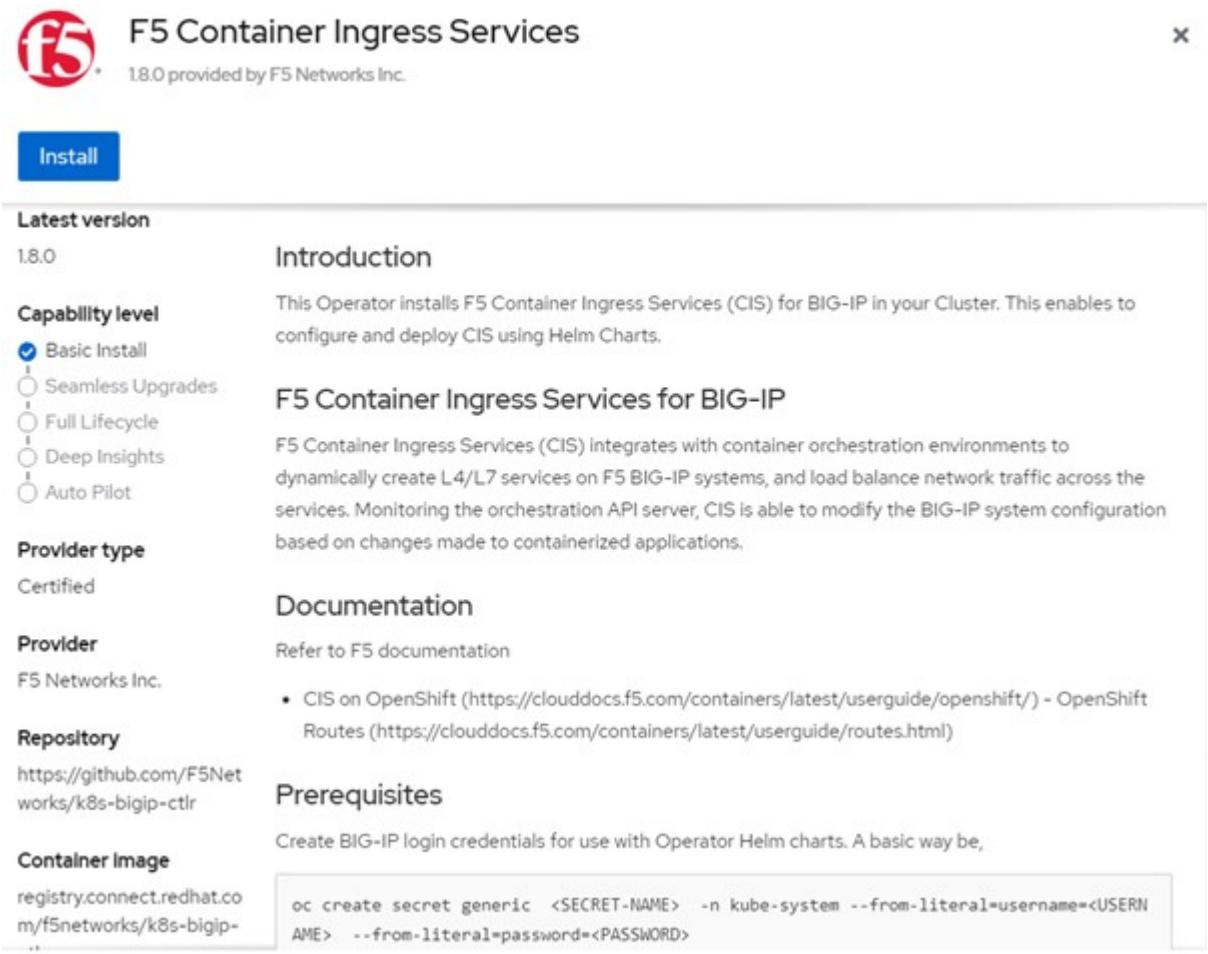
15. F5 CIS CRD をインストールします。

```
[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
```

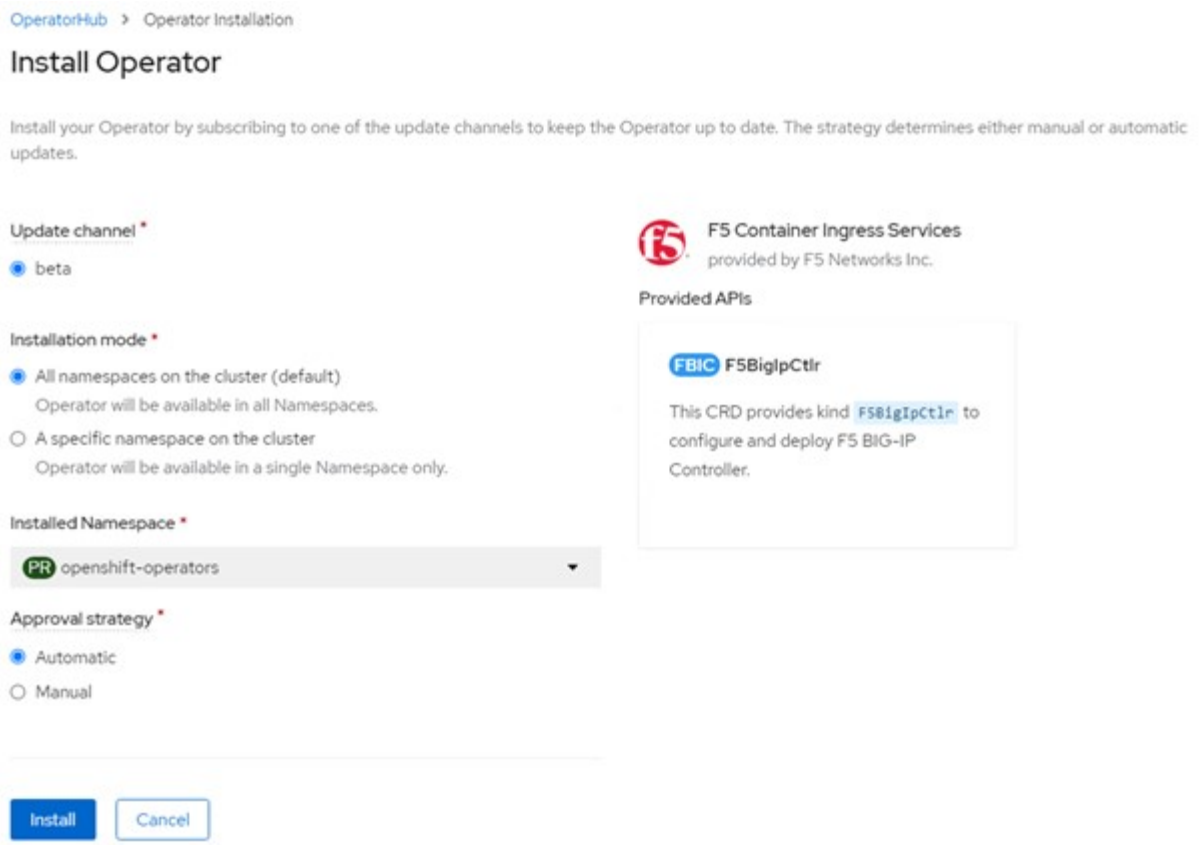
16. [演算子]>[演算子ハブ] に移動し、キーワード F5 を検索して、 F5 Container Ingress Service タイルをクリックします。



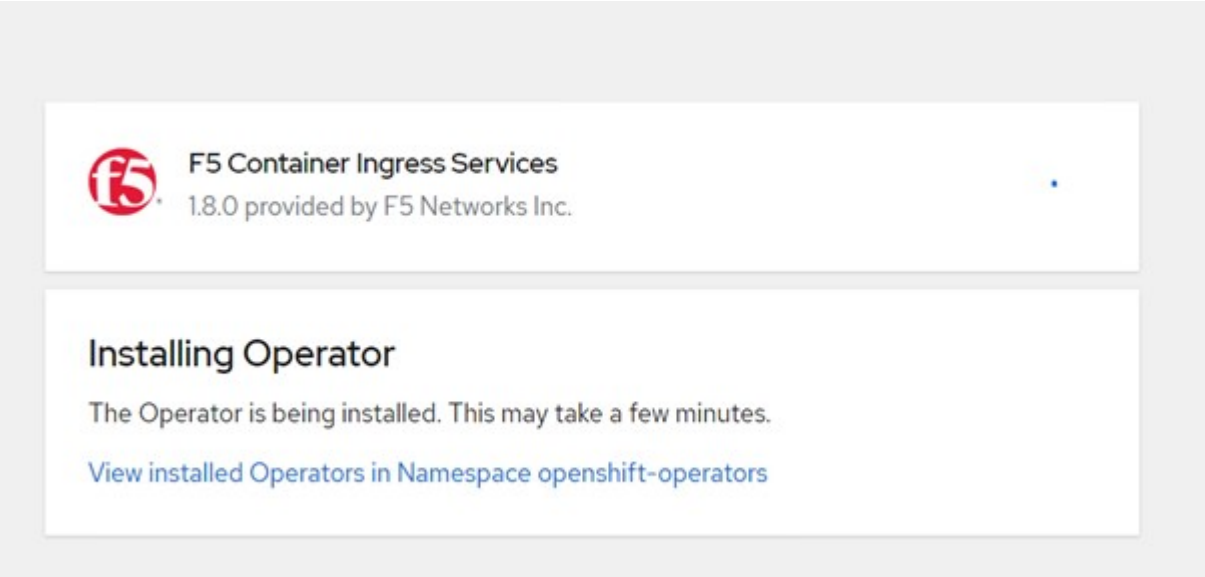
17. オペレータ情報を読み、 [インストール] をクリックします。



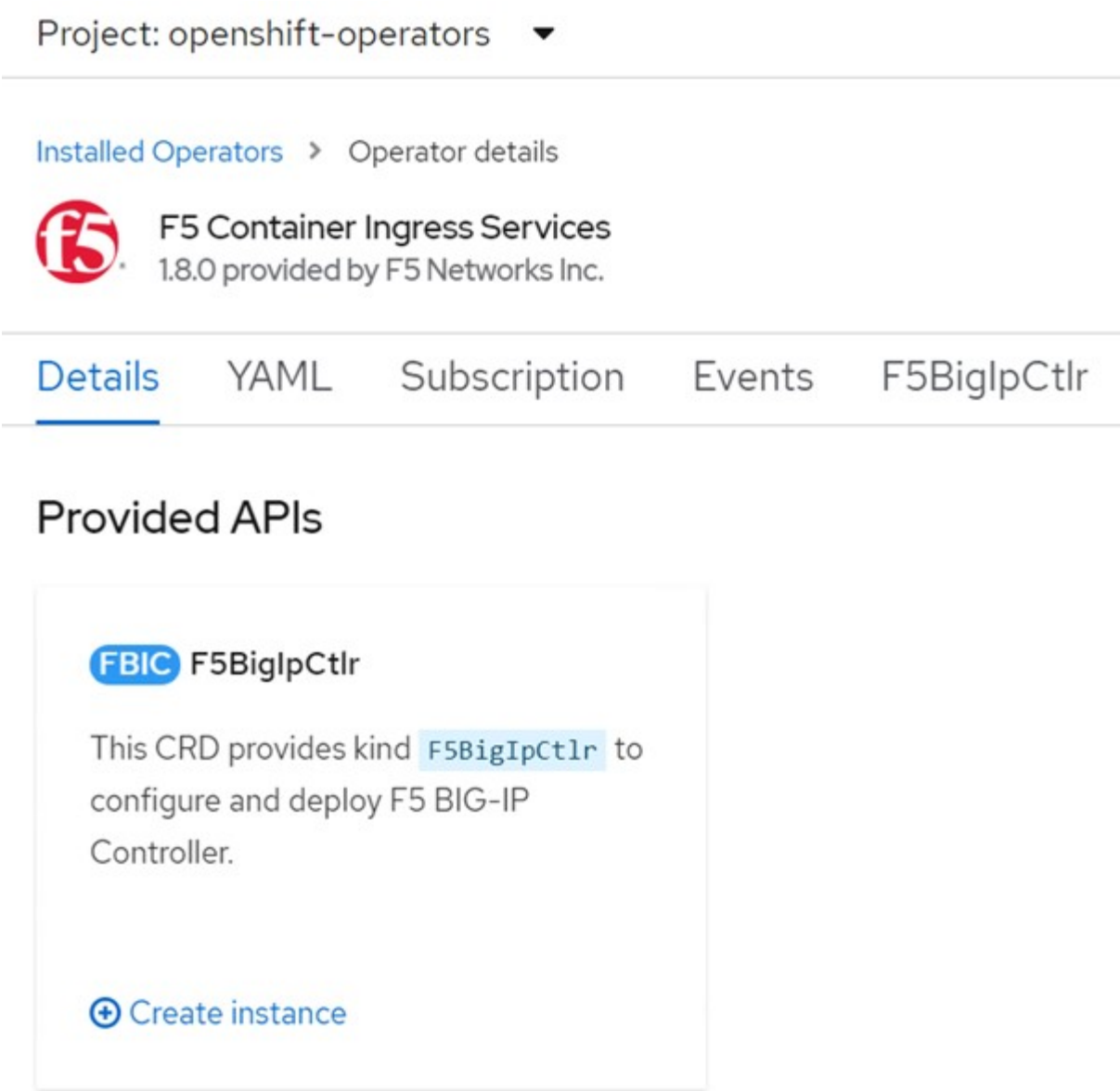
18. Install Operator （オペレータのインストール）画面で、デフォルトのパラメータをすべてそのままにして、 Install （インストール）をクリックします。



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、「 Installation Successful 」というメッセージが表示されます。
21. [演算子]>[インストールされている演算子] に移動し、 [F5BigIpCtlr] タイルの下にある [F5 Container Ingress Service] をクリックして、 [インスタンスの作成] をクリックします。

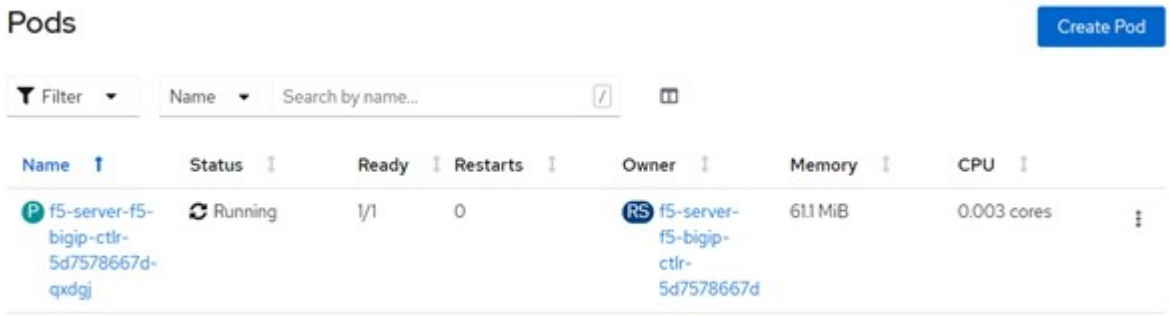


22. YAML View をクリックし、必要なパラメータを更新した後で次の内容を貼り付けます。

> 以下のパラメータ「 bigip_dpartition 」、「 OpenShift 」 SDN_NAME 」、「 bigip_url 」、「 bigip_login_secret 」を更新して、内容をコピーする前にセットアップの値を反映させます。

```
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ら、 [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 ルータは、これらのルートのトラフィックのアドバタイズと処理を行います。ただし、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。このシステムは、補助ルータとして実行することも、自己ホスト型 OpenShift ルータに代わるものでもあります。CIS は、 OpenShift ルートのルータとして機能する BIG-IP システムに仮想サーバを作成し、BIG-IP はアドバタイズメント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ついては、次のドキュメントを参照してください。これらのパラメータは、 APPS/v1 API の OpenShift Deployment リソースに対して定義されています。したがって、 F5BigIpCtlr リソース cis.f5.com/v1 API でこれらを使用する場合は、パラメータ名にハイフン (-) 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 'IPAM:true' と 'custom_resource_mode:true' があります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 で IPAM 統合が有効になっ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ドを作成します。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、下記の F5 IPAM 展開定義を貼り付けます。

以下の spec.template.spec.containers [0] の ip-range パラメータを更新して、設定に対応する ipamLabel と IP アドレス範囲を反映させます。

IPAM コントローラ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には 'ipamLabels[`range1`' および `range2` を以下の例に示します] が 'LoadBalancer 型のサービスに注釈を付ける必要があります

```
[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 コントローラ配置を作成します。

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

deployment/f5-ipam-controller created
```

29. F5 IPAM コントローラポッドが実行されてい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します

```
[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 コントローラが外部 IP を割り当て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. 導入環境を作成し、作成した LoadBalancer サービスを使用します。

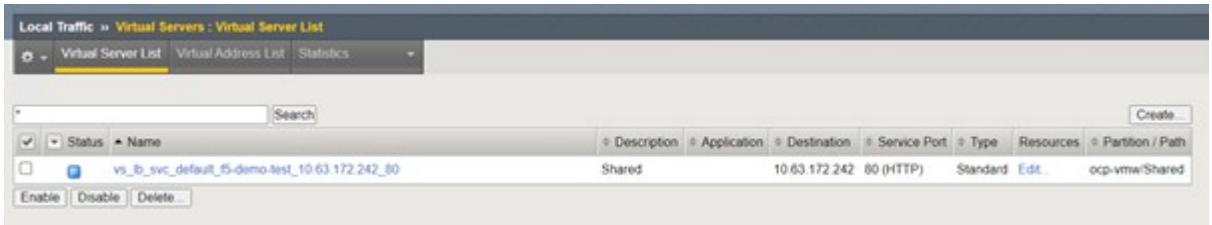
```
[admin@rhel-7 ~]$ vi example_deployment.yaml

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. ポッドが実行されている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. 対応する仮想サーバが、 OpenShift の LoadBalancer タイプのサービス用に BIG-IP システムに作成されているかどうかを確認します。Local Traffic > Virtual Servers > Virtual Server List の順に選択します。



プライベートイメージレジストリを作成しています

Red Hat OpenShift の導入では、のようなパブリックレジストリを使用します "キー・ IO" または "DockerHub" お客様のほとんどのニーズに対応ただし、お客様が独自のプライベートイメージまたはカスタマイズされたイメージをホストしたい場合があります。

この手順ドキュメントでは、 Astra Trident と NetApp ONTAP が提供する永続的ボリュームを使用して作成された、プライベートイメージレジストリを作成しています。

> Astra Control Center では、 Astra コンテナに必要なイメージをホストするためにレジストリが必要です。次のセクションでは、 Red Hat OpenShift クラスタにプライベートレジストリをセットアップし、 Astra Control Center 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を作成しています

1. 現在のデフォルト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、 OpenShift クラスタの Trident バック対象ストレージクラスをデフォルトとしてアノテートします。

```
[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. 「 PEC 」セクションに以下の保管パラメータを入力して、 imagegeistry のオペレータを編集します。

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

storage:
  pvc:
    claim:
```

3. カスタムホスト名を使用して OpenShift ルートを作成するには、「 PEC 」セクションに次のパラメータ

を入力します。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、ルートの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するには、「 PEC 」セクションに「 defaultRoute ： true 」というパラメータを追加します。

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 入力オペレータのデフォルトの TLS 設定が使用されます。ただし、カスタム TLS 設定をルートに追加することはできます。これには、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秘密を作成します。

```
[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 演算子を編集して 'PEC' セクションに次のパラメータを追加します

```
[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. このような場合は、すべての管理者をもう一度編集し、管理状態を「管理状態」に変更してください。保存して終了します。

```
oc edit configs.imageregistry/cluster

managementState: Managed
```

5. すべての前提条件を満たしている場合は、プライベートイメージレジストリに PVC 、ポッド、およびサービスが作成されます。数分後にレジストリが起動します。

```
[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. 入力オペレータ 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。TLS 証明書を使用して「 OpenShift -config 」ネームスペースに ConfigMap を作成し、証明書を信頼できるようにクラスタイメージ設定にパッチします。

```
[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 の内部レジストリは認証によって制御されます。OpenShift ユーザはすべて OpenShift レジストリにアクセスできますが、ログインユーザが実行できる操作はユーザ権限によって異なります。
   a. ユーザーまたはユーザーのグループがレジストリから画像をプルできるようにするには、ユーザーにレジストリビューアの役割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にイメージの書き込みまたはプッシュを許可するには、ユーザーにレジストリエディタの役割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、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設定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するか、対応するポッド定義で参照できます。
    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、「 PEC 」セクションに次のパラメータを追加します。

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

11. OpenShift ノードとは別に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 ログイン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 ユーザクレデンシャル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 : 「 kubeadmin 」ユーザを使用してプライベートレジストリにログインする場合は、パスワードの代わりにトークンを使用します。

Docker です

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

+ 注 : 「 kubeadmin 」ユーザを使用してプライベートレジストリにログインする場合は、パスワードの代わりにトークンを使用します。

d. 画像を押したり引いたりします。

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
```

Docker です

```
[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
```

解決策の検証とユースケース：ネットアップを使用した **Red Hat OpenShift**

このページに記載する例は、ネットアップでの Red Hat OpenShift の解決策の検証と使用事例です。

- "永続的ストレージを使用した Jenkins CI/CD パイプラインの導入"
- "ネットアップを使用して Red Hat OpenShift でマルチテナンシーを構成します"
- "NetApp ONTAP を使用した Red Hat OpenShift Virtualization"
- "ネットアップを使用した Red Hat OpenShift での Kubernetes 向けの高度なクラスタ管理"

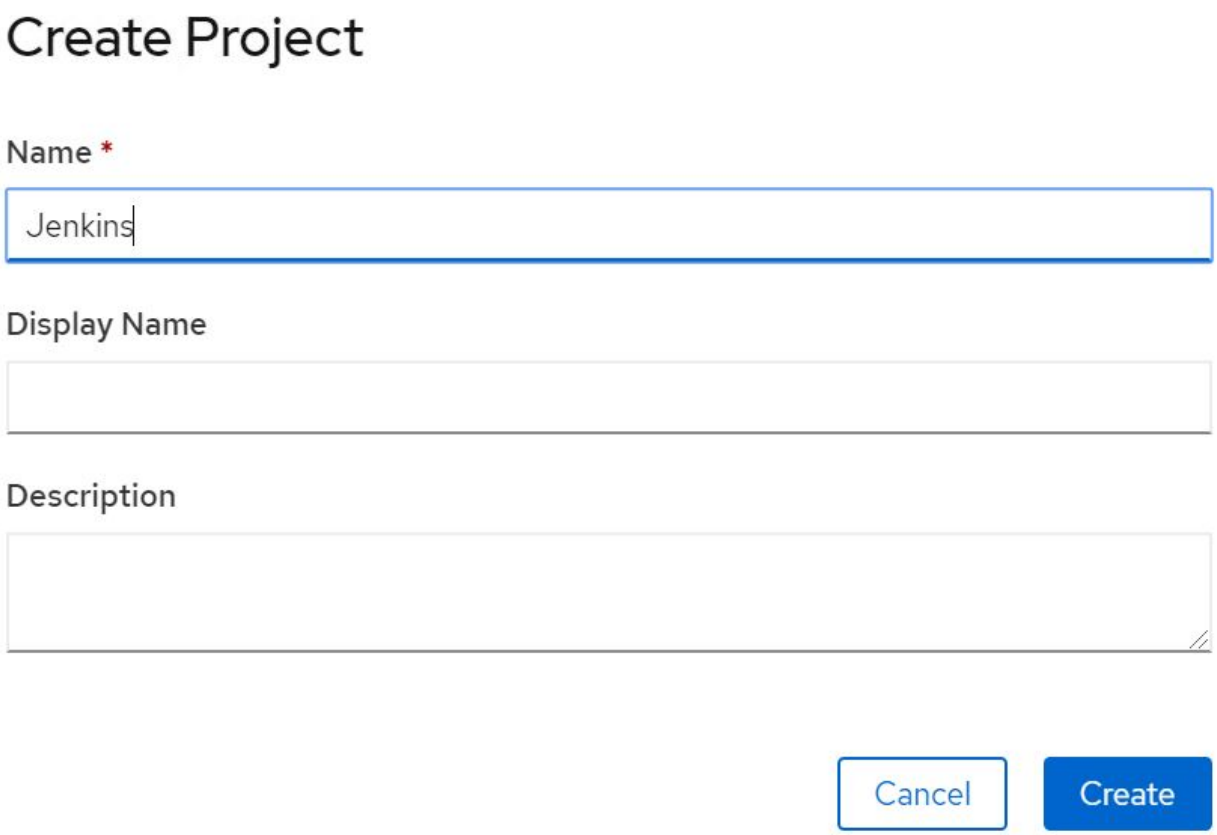
永続的ストレージを使用した **Jenkins CI / CD** パイプラインの導入：ネットアップでの **Red Hat OpenShift**

このセクションでは、 Jenkins との継続的統合 / 継続的配信または導入（ CI / CD ）パイプラインを導入して解決策の動作を検証する手順について説明します。

Jenkins の導入に必要なリソースを作成します

Jenkins アプリケーションの導入に必要なリソースを作成するには、次の手順に従います。

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、永続的ストレージを使用して Jenkins を導入しています。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] > [永続的ボリューム要求] の順に選択し、 [永続的ボリューム要求の作成] をクリックします。作成したストレージクラスを選択し、永続ボリューム要求名が Jenkins であることを確認し、適切なサイズとアクセスモードを選択して、作成をクリックします。

Project: jenkins

Create Persistent Volume Claim

Edit YAML

Storage Class

SC basic

Storage class for the new claim.

Persistent Volume Claim Name *

jenkins

A unique name for the storage claim within the project.

Access Mode *

Single User (RWO) Shared Access (RWX) Read Only (ROX)

Permissions to the mounted drive.

Size *

100 GiB

Desired storage capacity.

Use label selectors to request storage

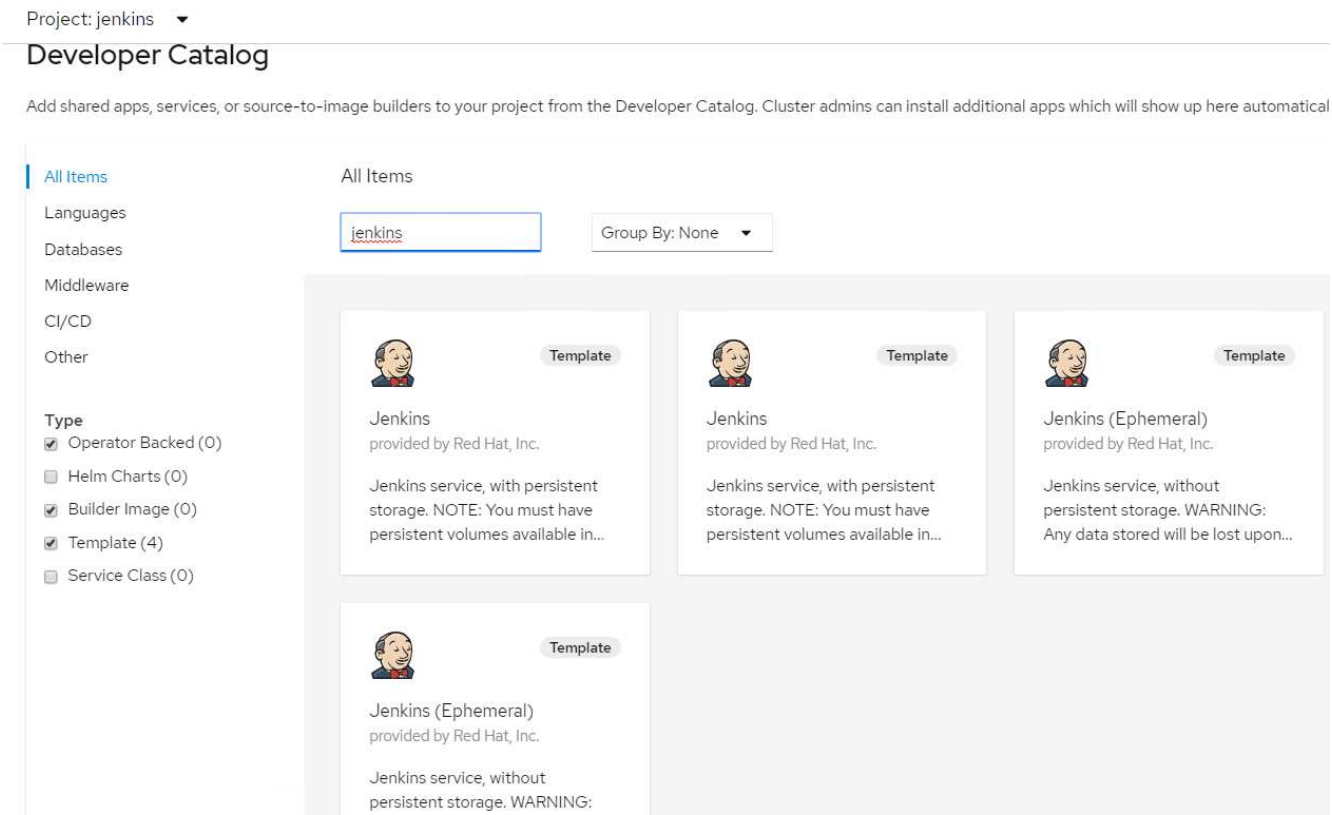
Use label selectors to define how storage is created.

Create Cancel

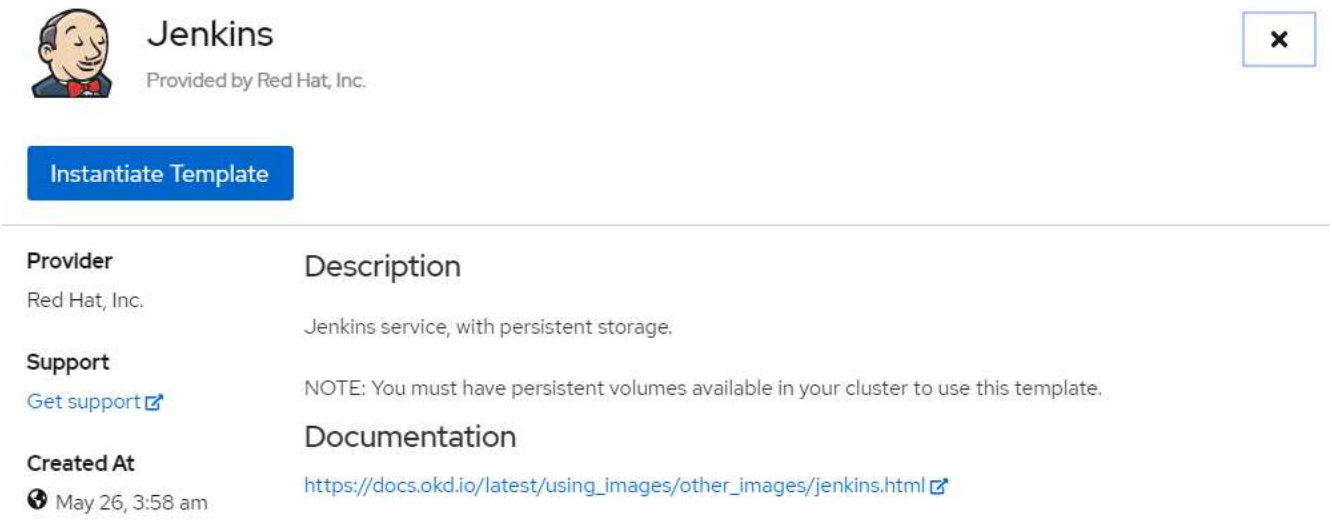
永続的ストレージを使用して **Jenkins** を導入する

永続ストレージを使用して Jenkins を導入するには、次の手順を実行します。

1. 左上隅で、ロールを Administrator から Developer 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。永続的ストレージを使用する Jenkins Service を選択します。

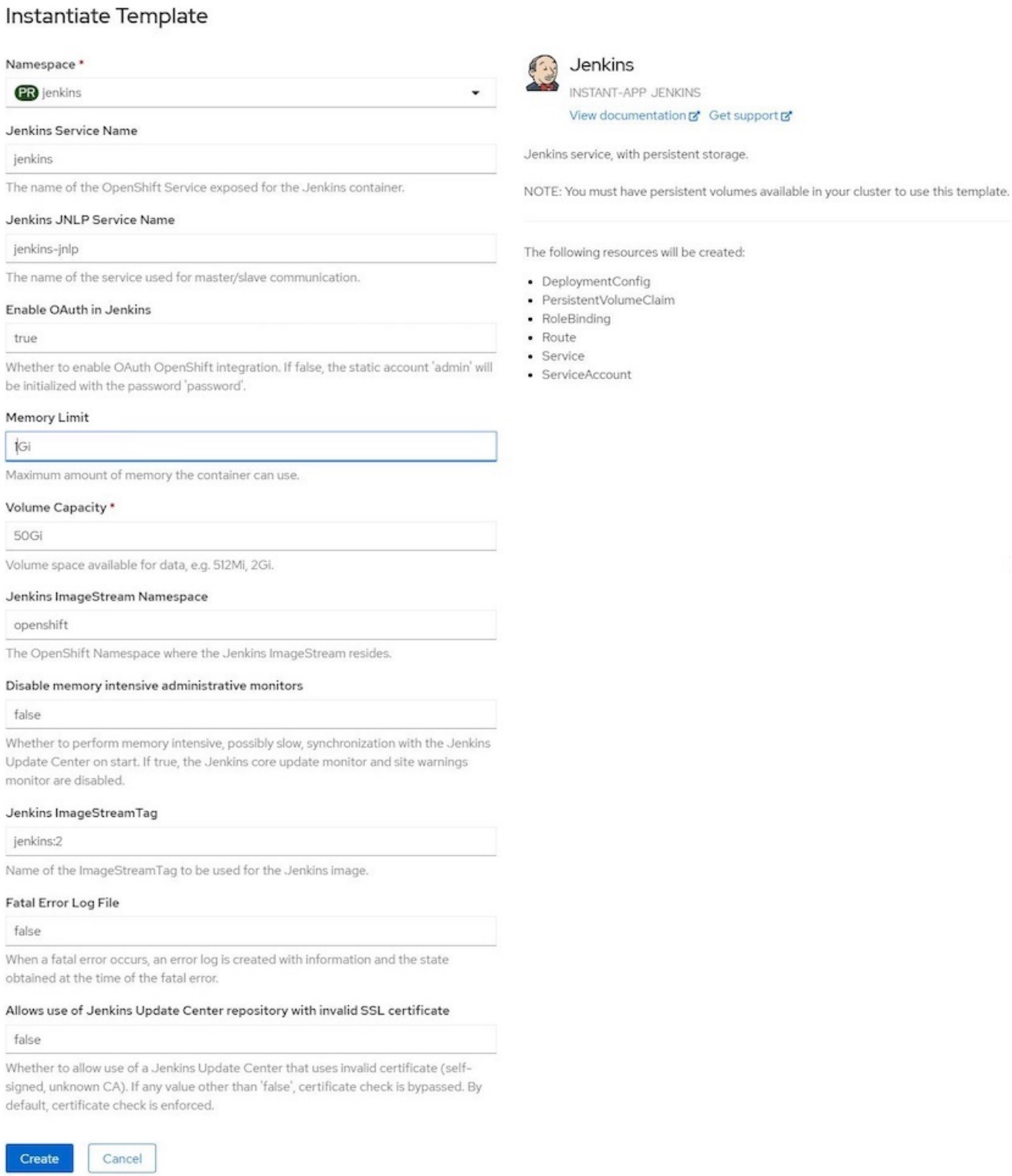


2. 「テンプレートをインスタンス化」をクリックします。



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。要件に基づいてパラメータを変更し、 [作成（ Create ）] をクリックします。このプロセスでは、 OpenShift で Jenkins をサポートするた

めに必要なリソースがすべて作成されます。



4. Jenkins ポッドが「 Ready 」状態になるまでに約 10 〜 12 分かかります。

Project: jenkins

Pods

Create Pod

Filter by name...

1 Running | 0 Pending | 0 Terminating | 0 CrashLoopBackOff | 1 Completed | 0 Failed | 0 Unknown

Select all filters

1 of 2 Items

Name	Namespace	Status	Ready	Owner	Memory	CPU
P jenkins-1-c77n9	NS jenkins	Running	1/1	RC jenkins-1	-	0.004 cores

5. ポッドがインスタンス化されたら、ネットワーキング > ルートと進みます。Jenkins の Web ページを開くには、 Jenkins ルート用の URL をクリックします。

Project: jenkins

Routes

Create Route

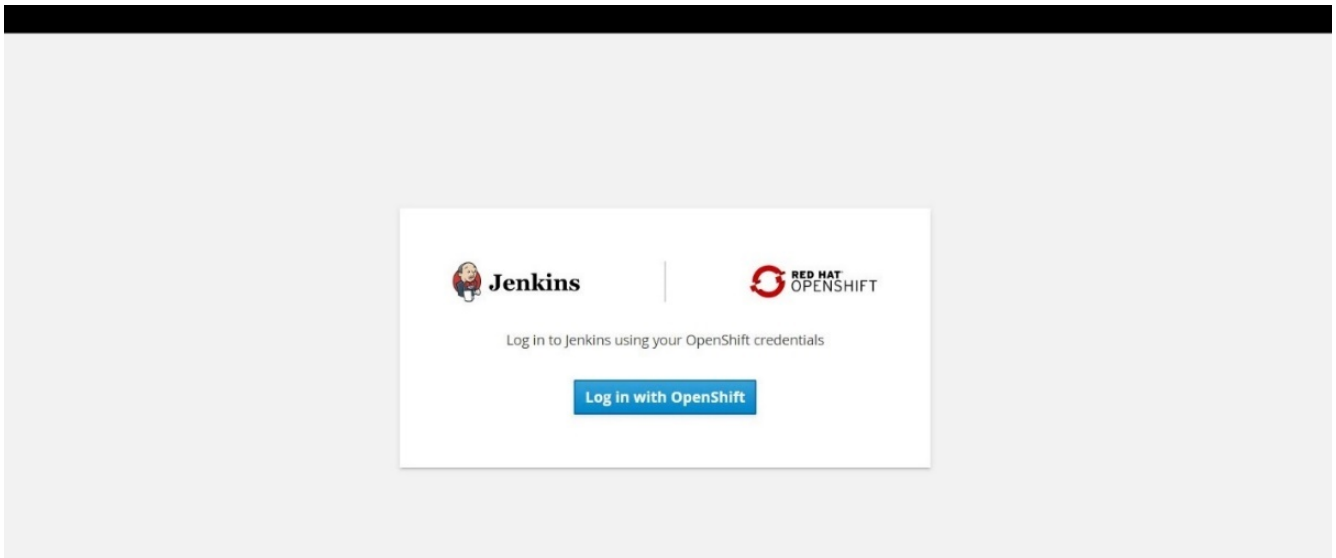
Filter by name...

1 Accepted | 0 Rejected | 0 Pending | Select all filters

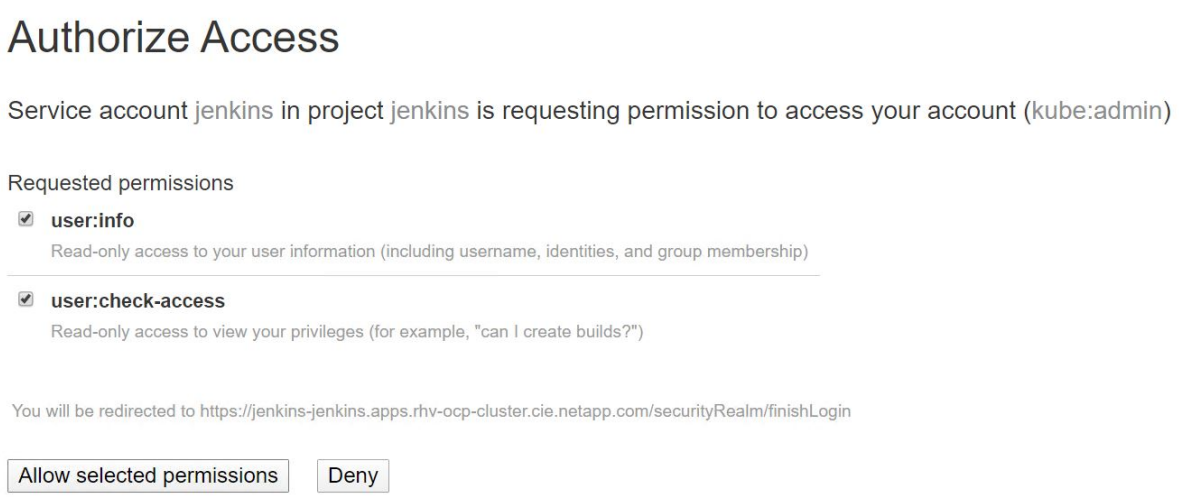
1 Item

Name	Namespace	Status	Location	Service
RT jenkins	NS jenkins	Accepted	https://jenkins-jenkins.apps.rhv-ocp-cluster.cie.netapp.com	S jenkins

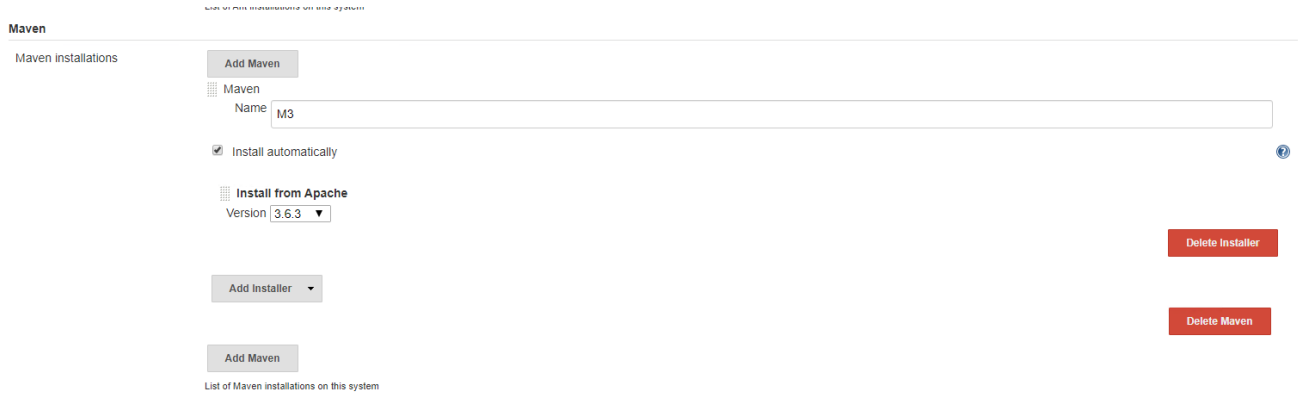
6. Jenkins アプリケーションの作成時に OpenShift OAuth が使用されていたため、「 OpenShift でログイン」をクリックします。



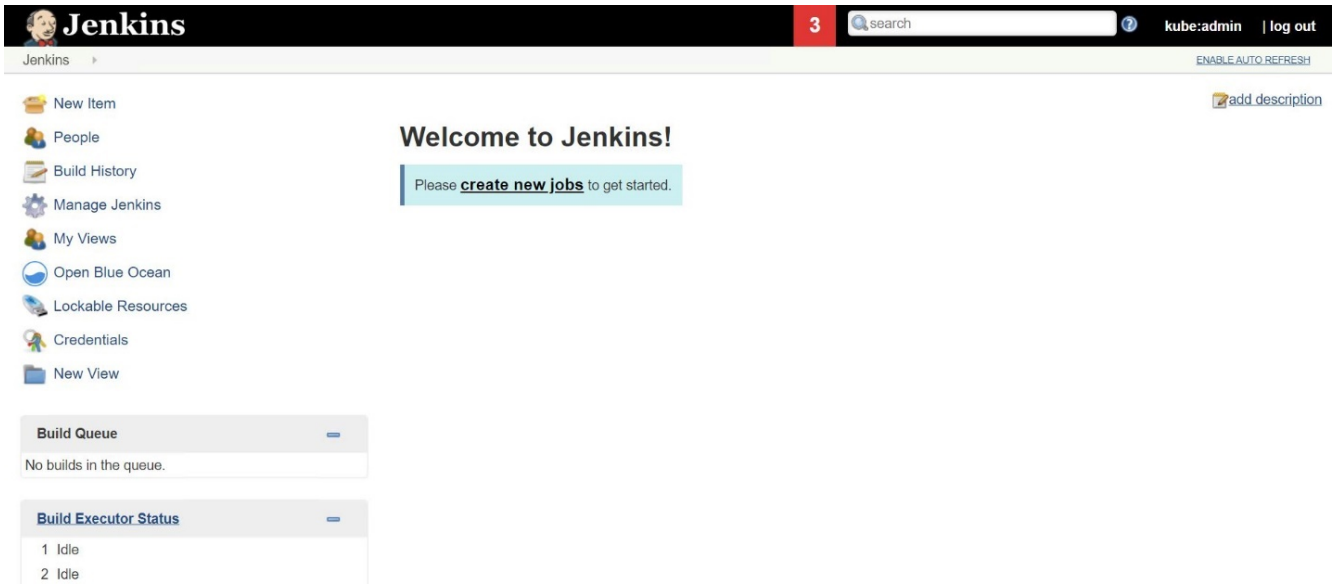
7. Jenkins サービスアカウントに OpenShift ユーザへのアクセスを許可します。



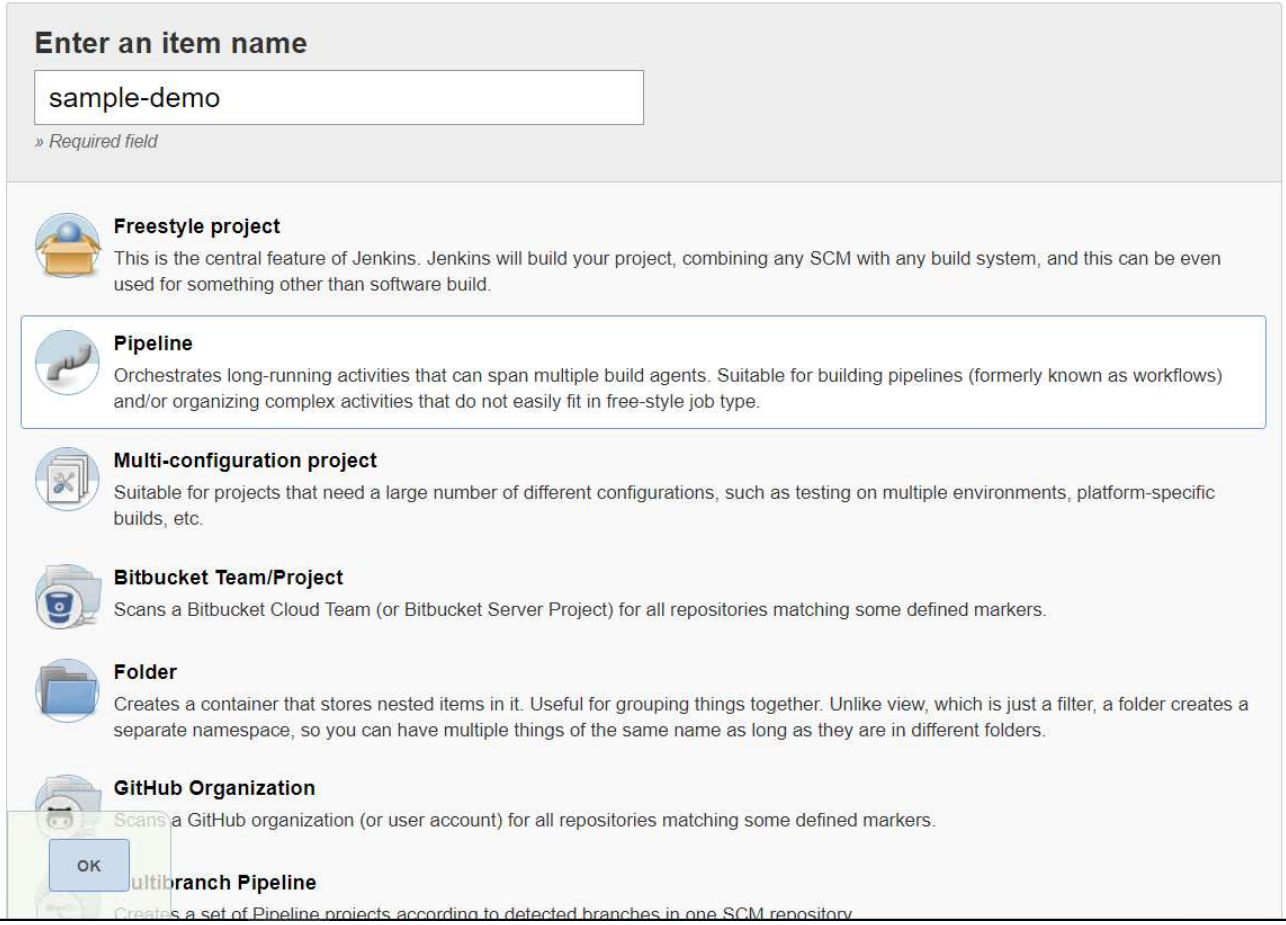
8. Jenkins のようこそページが表示されます。Maven ビルドを使用しているので、まず Maven のインストールを完了します。Manage Jenkins > Global Tool Configuration に移動し、 Maven サブヘッドで Add Maven をクリックします。任意の名前を入力し、 [自動的にインストール] オプションが選択されていることを確認します。[保存] をクリックします .



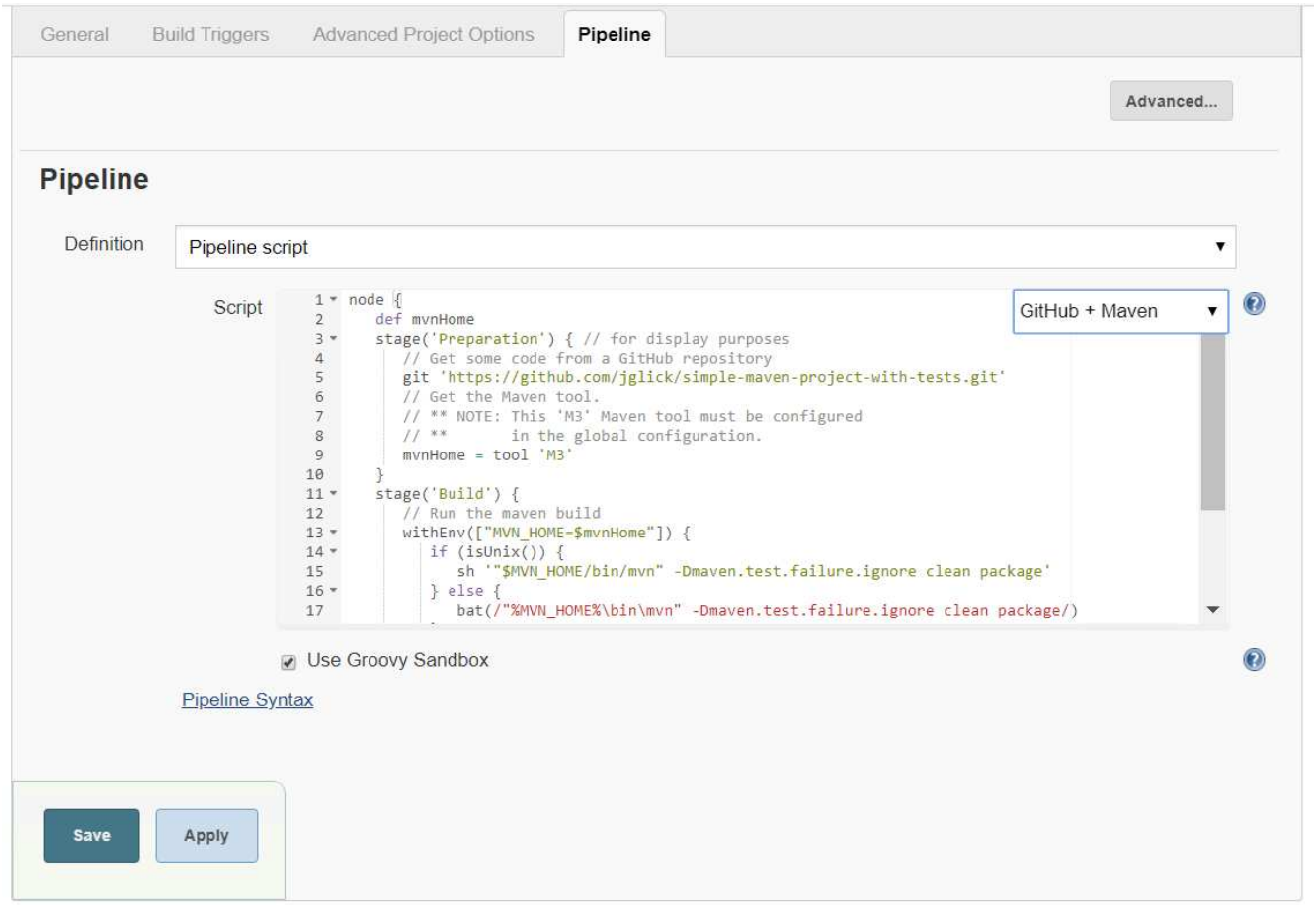
9. CI / CD のワークフローを示すパイプラインを作成できるようになりました。ホームページで、左側のメニューから [新規ジョブの作成] または [新規アイテム] をクリックします。



10. [項目の作成] 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 (Pipeline) 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。コードが自動的に入力されます。[保存] 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、テストの各フェーズで開発を開始します。ビルドプロセス全体が完了してビルドの結果が表示されるまでに数分かかることがあります。

13. コードが変更された場合は、必ず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することで、継続的な統合と継続的な提供を実現できます。[最近の変更] をクリックして、前のバージョンからの変更を追跡します。

NetApp ONTAP を使用して Red Hat OpenShift にマルチテナンシーを設定します

ネットアップを使用した **Red Hat OpenShift** でのマルチテナンシーの構成

コンテナ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は、アプリケーションやワークロードごとに 1 つの Red Hat OpenShift クラスタを導入する傾向にあります。これにより、アプリケーションやワークロードを厳密に分離し、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できます。ただし、アプリケーションごとに独立した Red Hat OpenShift クラスタを導入するには、独自の問題が発生します。これにより、各クラスタを個別に監視および管理する必要がある運用上のオーバーヘッドが増大し、さまざまなアプリケーションに専用リソースを使用することでコストが増大し、効率的な拡張性が妨げられます。

この問題を解決するには、すべてのアプリケーションまたはワークロードを 1 つの Red Hat OpenShift クラスタで実行することを検討します。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セキュリティの脆弱性が大きな課題の 1 つとなっています。あるワークロードのセキュリティの脆弱性は、自然に別のワークロードにオーバーフローする可能性があるため、影響ゾーンが増加します。また、あるアプリケーションによる突然の制御されないリソース使用率は、デフォルトではリソース割り当てポリシーがないため、別のアプリケーションのパフォーマンスに影響を与える可能性があります。

そのため、組織は、たとえば、すべてのワークロードを単一のクラスタで実行しながら、各ワークロードに専用のクラスタのメリットを提供することで、両方の世界で最も優れたソリューションを見つけることができます。

このように効果的な解決策の 1 つは、 Red Hat OpenShift でマルチテナンシーを構成することです。マルチテナンシーは、複数のテナントを同じクラスタ上に共存させ、リソースやセキュリティなどを適切に分離できるアーキテクチャです。この場合、テナントは、特定のユーザグループが専用として使用するように設定されたクラスタリソースのサブセットとみなすことができます。Red Hat OpenShift クラスタでマルチテナンシーを設定する利点は次のとおりです。

- クラスタリソースを許可することで設備投資と運用コストを削減 を共有します
- 運用と管理のオーバーヘッドを軽減
- セキュリティ侵害のクロスコンタミネーションからワークロードを保護
- リソースの競合による予期しないパフォーマンスの低下からワークロードを保護

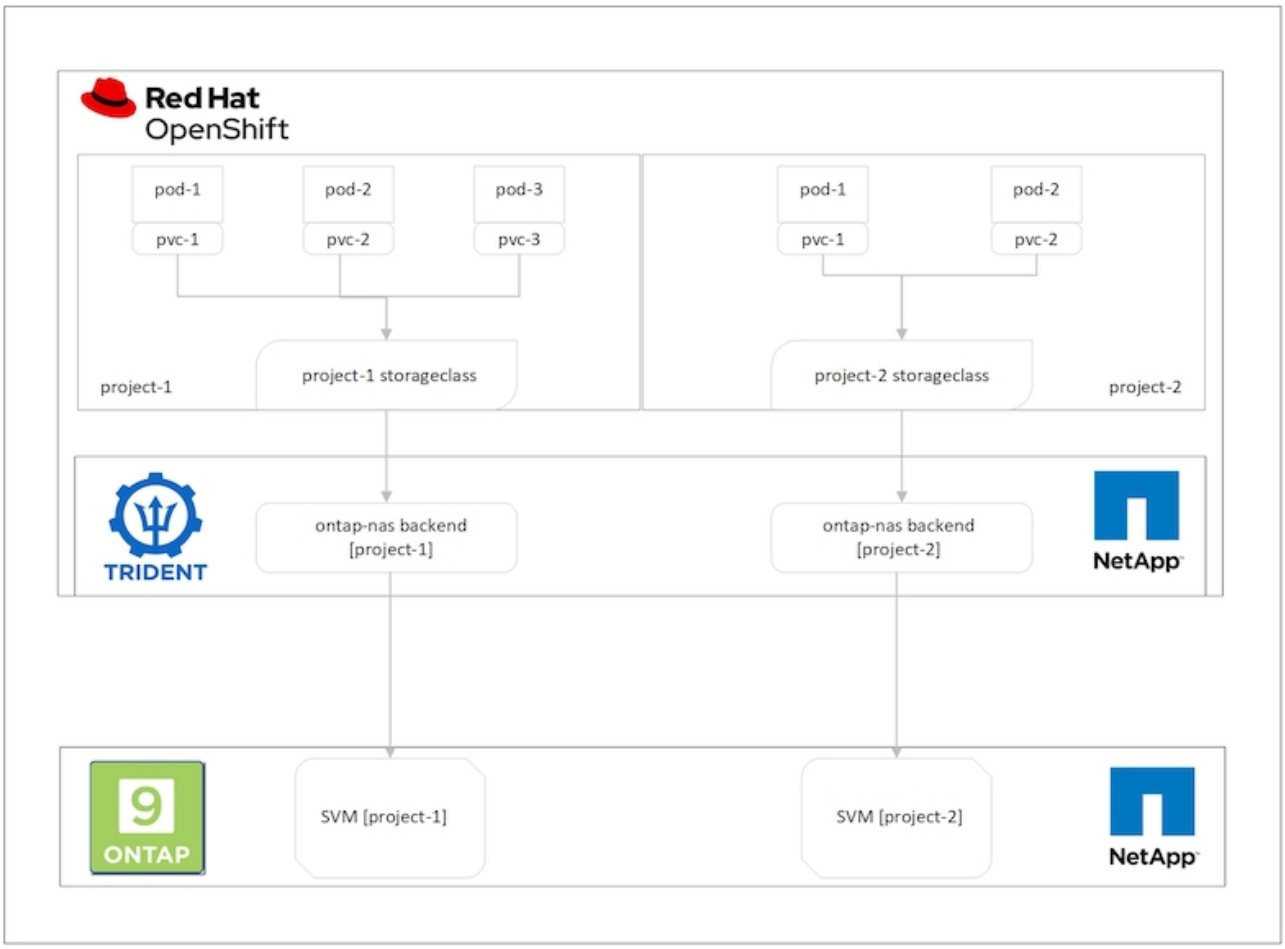
マルチテナント OpenShift クラスタを完全に実現するには、コンピューティング、ストレージ、ネットワーク、セキュリティなど、異なるリソースバケットに属するクラスタリソースにクォータと制限を設定する必要があります。この解決策のすべてのリソースバケットの特定の側面について説明しますが、 ネットアップでは、 NetApp ONTAP を基盤とする Astra Trident によって動的に割り当てられるストレージリソースにマルチテナンシーを設定することで、複数のワークロードで提供または消費されるデータを分離し、保護するためのベストプラクティスに焦点を当てています。

アーキテクチャ

ネットアップ ONTAP を基盤とする Red Hat OpenShift と Astra Trident は、デフォルトでワークロードを分離する機能を提供していませんが、マルチテナンシーの設定に使用できる幅広い機能を備えています。ネットアップ ONTAP を基盤とする Astra Trident を使用した Red Hat OpenShift クラスタでのマルチテナント解決策の設計について理解を深めるために、一連の要件を含む例を検討し、その構成について概説します。

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、組織が Red Hat OpenShift クラスタ上で 2 つのワークロードを実行するとします。こうしたワークロードのデータは、 NetApp ONTAP NAS バックエンドの Astra 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、この 2 つのワークロードに対応するマルチテナント解決策を設計し、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、セキュリティとパフォーマンスを維持することが求められています。主に、これらのアプリケーションを提供するデータに重点が置かれています。

次の図は、ネットアップ ONTAP を基盤とする Astra Trident を使用した Red Hat OpenShift クラスタ上のマルチテナント解決策を示しています。



テクノロジ要件

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

Red Hat OpenShift –クラスタリソース

Red Hat OpenShift クラスタの観点からは、最初に最上位のリソースがプロジェクトです。OpenShift プロジェクトは、 OpenShift クラスタ全体を複数の仮想クラスタに分割するクラスタリソースと見なすことができます。したがって、プロジェクトレベルでの分離によって、マルチテナンシーの設定の基盤が提供されます。

次に、クラスタで RBAC を設定します。ベストプラクティスとして、すべての開発者が 1 つのプロジェクトまたはワークロードを担当し、アイデンティティプロバイダ（ IdP ）内の単一のユーザグループに設定することを推奨します。Red Hat OpenShift では、 IdP の統合とユーザグループの同期が可能なため、 IdP のユーザとグループをクラスタにインポートできるようになります。これにより、クラスタ管理者は、プロジェクト専用のクラスタリソースへのアクセスをそのプロジェクトに使用するユーザグループまたはグループに分離して、クラスタリソースへの不正アクセスを制限できます。Red Hat OpenShift への IdP の統合の詳細については、のドキュメントを参照してください "こちらをご覧ください"。

NetApp ONTAP

Red Hat OpenShift クラスタの永続的ストレージプロバイダとして機能している共有ストレージを分離し、各プロジェクト用にストレージ上に作成されたボリュームが、別々のストレージ上に作成されたものと同じようにホストに表示されるようにすることが重要です。そのためには、プロジェクトやワークロードに応じて Storage Virtual Machine （ SVM ）を NetApp ONTAP 上に作成し、各 SVM をワークロード専用にします。

Astra Trident

NetApp ONTAP で作成されたプロジェクトごとに異なる SVM が作成されたら、各 SVM を異なる Trident バックエンドにマッピングする必要があります。Trident のバックエンド構成は、 OpenShift クラスタリソースへの永続的ストレージの割り当てを促進します。また、マッピング先の SVM の詳細が必要です。これは、バックエンドのプロトコルドライバである必要があります。必要に応じて、ストレージでのボリュームのプロビジョニング方法を定義したり、ボリュームのサイズやアグリゲートの使用などを制限したりできます。Trident バックエンドの定義に関する詳細はこちらをご覧ください "こちらをご覧ください"。

Red Hat OpenShift –ストレージリソース

Trident バックエンドを設定したら、次の手順として StorageClasses を設定します。バックエンドと同じ数のストレージクラスを構成して、各ストレージクラスが 1 つのバックエンドにしかボリュームをスピンアップできない。ストレージクラスを定義する際に StoragePools パラメータを使用して、ストレージクラスを特定の Trident バックエンドにマッピングできます。ストレージクラスを定義する詳細については、を参照してください "こちらをご覧ください"。そのため、 StorageClass から Trident バックエンドへの 1 対 1 のマッピングで、 1 つの SVM をポイントします。これにより、そのプロジェクトに割り当てられた StorageClass を経由するすべてのストレージ要求が、そのプロジェクト専用の SVM によって処理されます。

ストレージクラスにネームスペースリソースが含まれていないため、あるプロジェクトのストレージクラスに対するストレージ要求を別のネームスペースまたはプロジェクトのポッドで拒否するにはどうすればよいですか？回答では、 ResourceQuotas を使用します。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で消費できるリソースの合計量だけでなく、リソースの数も制限できます。ほとんどの場合、 ResourceQuotas を使用してプロジェクトのリソースを制限することができます。この機能を効率的に使用することで、リソースのオーバープロビジョニングや過剰消費によるコストやシステム停止を削減できます。のドキュメントを参照してください "こちらをご覧ください" を参照してください。

このユースケースでは、特定のプロジェクトのポッドが、プロジェクト専用ではないストレージクラスのストレージを要求しないように制限する必要があります。これを行うには '<storage-class-name>.storageeclass.storage0.k8sio/persistentvolumeclaims'0 を設定して ' 他のストレージ・クラスに対する永続的ボリューム要求を制限する必要がありますさらに、クラスタ管理者は、プロジェクト内の開発者が ResourceQuotas を変更するためのアクセス権を持っていないことを確認する必要があります。

設定

マルチテナント解決策では、必要以上に多くのクラスタリソースにアクセスすることはできません。つまり、マルチテナンシー構成の一部として構成するリソースセット全体が、クラスタ管理者、ストレージ管理者、および各プロジェクトに取り組む開発者に分けられます。

次の表に、各ユーザが実行する各タスクを示します。

ロール	タスク
* Cluster-admin*	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できます
	Storage Admin 用の ClusterRoles および RoleBindings を作成します
	ロールとロールの作成特定のアクセス権を割り当てる開発者のためのバインド プロジェクト
	[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設定します
* ストレージ管理者 *	NetApp ONTAP に SVM を作成する
	Trident バックエンドを作成
	ストレージクラスを作成します
	ストレージリソースクォータを作成します
* 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するためのアクセスを検証します
	アクセスを検証して、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します
	アクセス権を検証して、プロジェクト、リソースクォータ、ストレージクラスを表示または編集します

設定

前提条件

- NetApp ONTAP クラスタ：
- Red Hat OpenShift クラスタ
- Trident がクラスタにインストールされている。
- 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 への管理アクセス。
- OpenShift クラスタへのクラスタ管理者アクセス。
- クラスタがアイデンティティプロバイダに統合されました。
- アイデンティティプロバイダは、異なるチームのユーザを効率的に区別するように設定されています。

Configuration ：クラスタ管理者のタスク

Red Hat OpenShift cluster-admin によって次のタスクが実行されます。

1. Red Hat OpenShift クラスタに 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
```

ここで説明するロール定義は単なる例です。エンドユーザの要件に基づいて開発者の役割を定義する必要があります。

1. 同様に、 project-2 の開発者ロールを作成します。
2. すべての OpenShift およびネットアップストレージリソースは、通常はストレージ管理者が管理します。ストレージ管理者向けのアクセスは、 Trident のインストール時に作成された Trident オペレータ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 ResourceQuotas にアクセスして、ストレージの消費方法を制御する必要があります。
3. クラスタ内のすべてのプロジェクトの ResourceQuotas を管理する役割を作成して、ストレージ管理者に割り当てます。

```
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. クラスタが組織のアイデンティティプロバイダと統合され、ユーザグループがクラスタ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がクラスタに統合され、ユーザ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. ストレージ管理者用の ClusterRoleBindings を設定します。

```
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は、 Trident オペレータとリソースクォータの 2 つのロールにバインドする必要があります。

1. ロールの作成 - developer-project-1 のロールを project-1 の対応するグループ (OCP-project-1) にバインド

する開発者のバインディング。

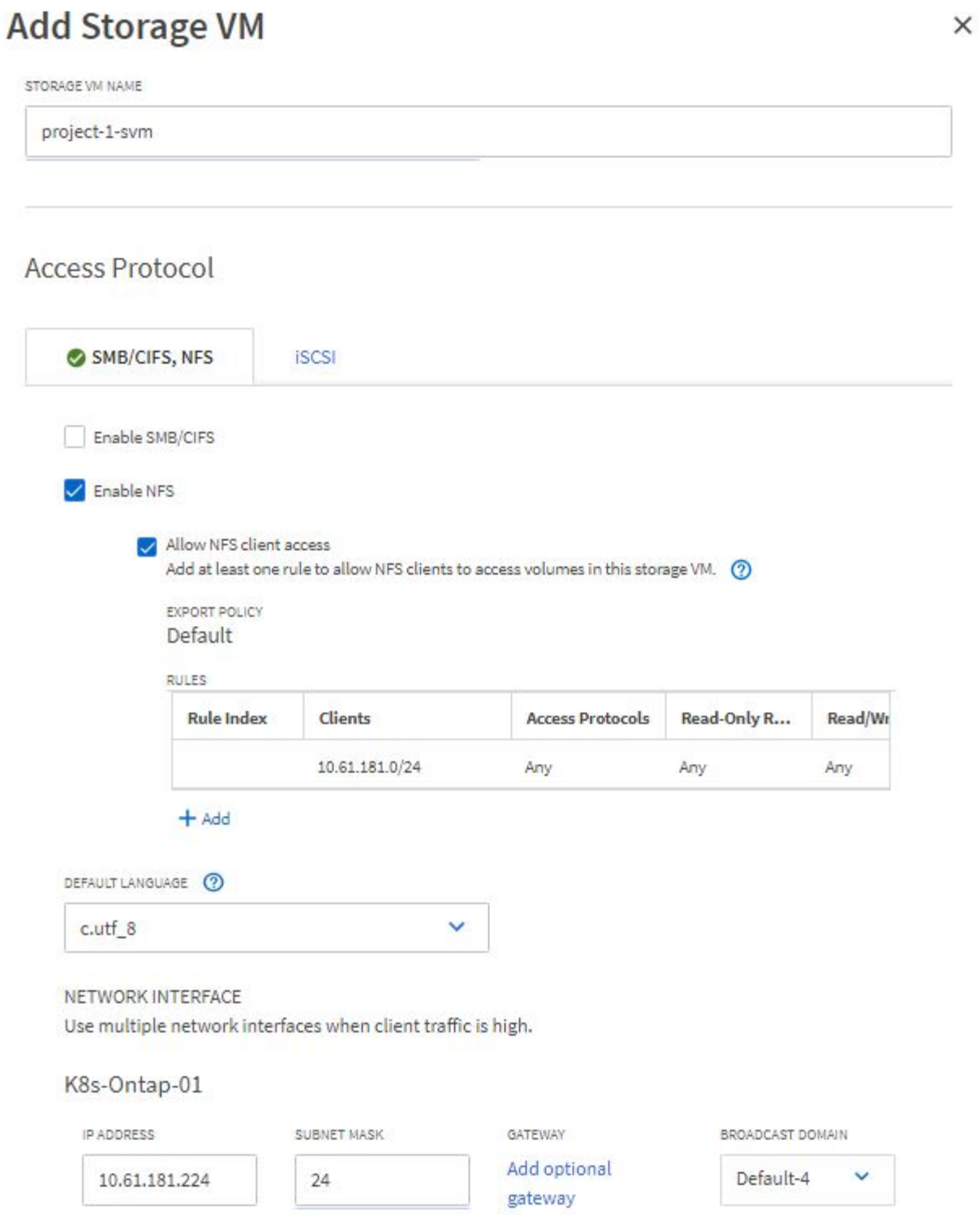
```
cat << EOF | oc create -f -
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: project-1-developer
  namespace: project-1
subjects:
  - kind: Group
    apiGroup: rbac.authorization.k8s.io
    name: ocp-project-1
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role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

2. 同様に、開発者の役割を project-2 の対応するユーザーグループにバインドする開発者の RoleBindings を作成します。

設定：ストレージ管理者のタスク

ストレージ管理者が次のリソースを設定する必要があります。

1. NetApp ONTAP クラスタに admin としてログインします。
2. Storage > Storage VMs と進み、 Add 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 用に 1 つずつ、 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には vsadmin アカウントを作成します。



1. ストレージ管理者として Red Hat OpenShift クラスタにログインします。
2. project-1 のバックエンドを作成し、プロジェクト専用の SVM にマッピングします。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 に接続することを推奨します。

```
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 と NAS のドライバを使用しています。ユースケースに基づいてバックエンドを作成する場合は、適切なドライバを使用します。

Trident が Trident プロジェクトにインストールされているとします。

1. 同様に、 project-2 の Trident バックエンドを作成し、 project-2 に専用の SVM にマッピングします。
2. 次に、ストレージクラスを作成します。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 project-1 のストレージクラスを作成し、これを設定します。

```
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 に対してストレージクラスを作成し、 project-2 に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設定します。
4. ResourceQuota を作成して ' プロジェクト 1 内のリソースを制限し ' 他のプロジェクト専用のストレージを要求します

```
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 'project-2 内のリソースを制限し ' 他のプロジェクト専用のストレージを要求します

検証

前の手順で設定し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します

1. OCP-project-1-user として、 project-1 の開発者としてログインします。
2. アクセス権をチェックして新しいプロジェクトを作成してください

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

3. project-1 に割り当てられた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. PV とそのボリュームが、 NetApp ONTAP 上のプロジェクト 1 専用の SVM に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、および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
```

アクセスを検証して別のプロジェクトに **PVC** またはポッドを作成するか、別のプロジェクト専用のリソースを使用します

1. OCP-project-1-user として、 project-1 の開発者としてログインします。
2. project-2 に割り当てられた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. PROJECT -2 で 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 「 test-pvc-project-1-sc-2 」および「 test-pvc-project-2-ssc-1 」が作成されていないことを確認します。

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

5. プロジェクト 2 で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
```

アクセス権を検証して、プロジェクト、リソースクォータ、ストレージクラスを表示および編集します

1. OCP-project-1-user として、 project-1 の開発者としてログインします。
2. アクセス権をチェックして新しいプロジェクトを作成してください。

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

3. アクセスを検証してプロジェクトを表示します

```
oc get ns
```

4. ユーザーがで ResourceQuotas を表示または編集できるかどうかを確認します プロジェクト 1

```
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ためのアクセス権を持っていることを確認します

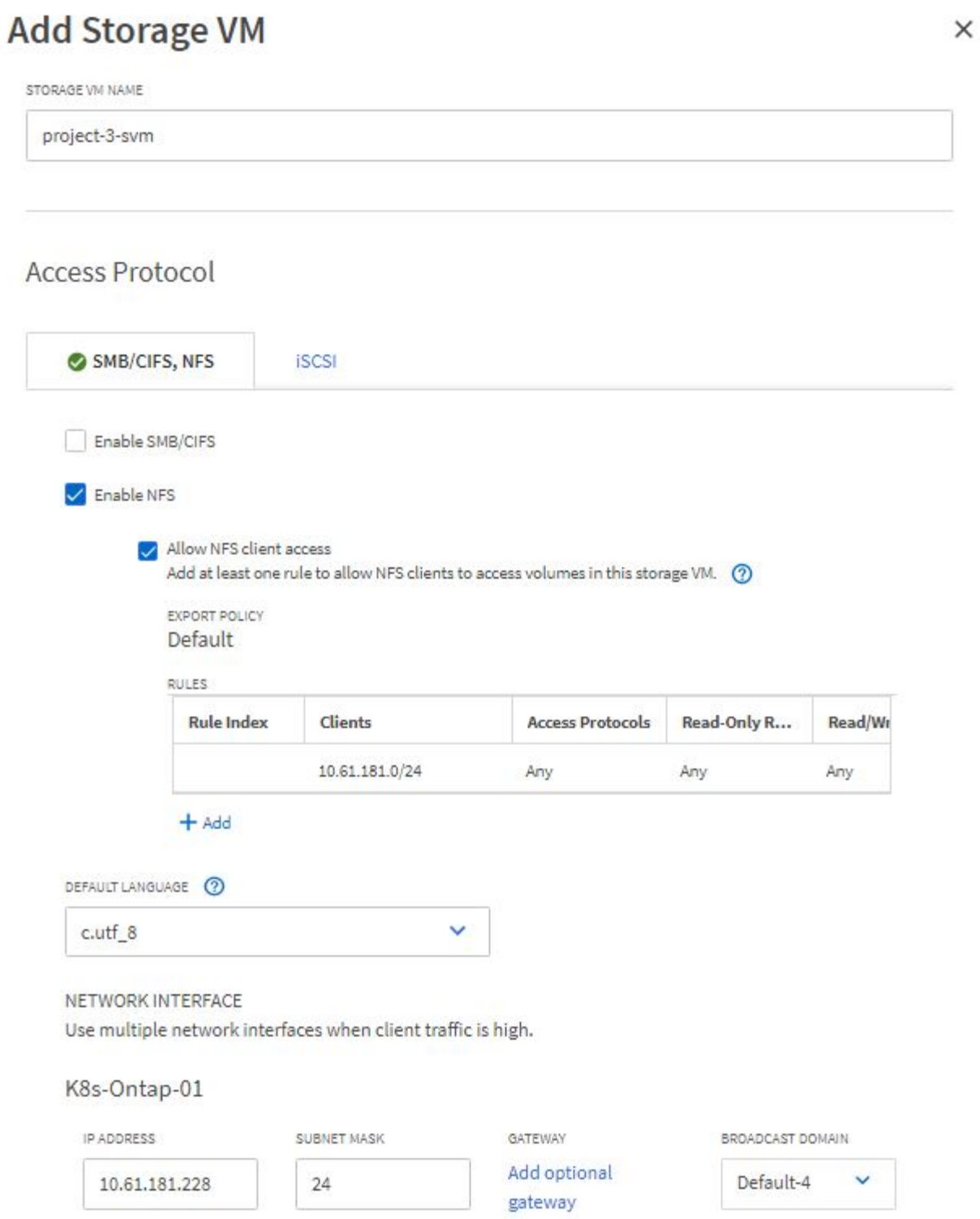
```
oc get sc
```

6. ストレージクラスについては ' アクセスを確認してください
7. ストレージクラスを編集するためにユーザーのアクセス権を検証します

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

マルチテナント構成でストレージリソースを使用する新しいプロジェクトを追加する場合、マルチテナンシーを違反しないように追加の設定が必要になります。マルチテナントクラスタでプロジェクトを追加するには、次の手順を実行します。

1. NetApp ONTAP クラスタにストレージ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。project-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には vsadmin アカウントを作成します。



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

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

3. IdP に project-3 のユーザグループが作成され、 OpenShift クラスタと同期されていることを確認してください。

```
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
```

ここで説明するロール定義は単なる例です。開発者ロールは、エンドユーザの要件に基づいて定義する必要があります。

1. プロジェクト 3 の開発者用に RoleBinding を作成します。これは、 developer-project-3 の役割を、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、 project-3 専用の SVM にマッピングします。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 に接続することを推奨します。

```
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 と NAS のドライバを使用しています。ユースケースに基づいてバックエンドを作成するための適切なドライバを使用します。

Trident が Trident プロジェクトにインストールされているとします。

1. project-3 用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設定します。

```
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 ' プロジェクト 3 のリソースを制限しますストレージを要求するストレージは ' 他のプロジェクト専用のストレージになります

```
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 ' プロジェクト内のリソースがプロジェクト 3 専用のストレージからストレージにアクセスするのを制限します

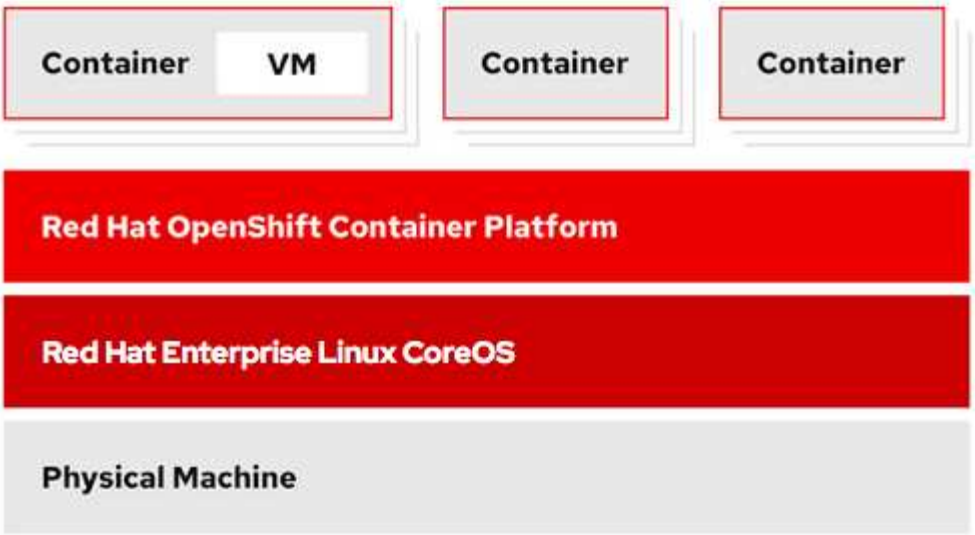
```
oc patch resourcequotas project-1-sc-rq -n project-1 --patch
'{"spec":{"hard":{ "project-3-
sc.storageclass.storage.k8s.io/persistentvolumeclaims": 0}}}'
oc patch resourcequotas project-2-sc-rq -n project-2 --patch
'{"spec":{"hard":{ "project-3-
sc.storageclass.storage.k8s.io/persistentvolumeclaims":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

NetApp ONTAP を使用した Red Hat OpenShift Virtualization

NetApp ONTAP を使用した Red Hat OpenShift Virtualization

それぞれのユースケースに応じて、コンテナと仮想マシン（ VM ）はどちらも、さまざまなタイプのアプリケーションに最適なプラットフォームとして機能します。そのため、多くの組織では、ワークロードの一部をコンテナで実行し、一部を VM で実行しています。そのため多くの場合、 VM 用のハイパーバイザーとアプリケーション用のコンテナオーケストレーションツールという別々のプラットフォームを管理する必要があり、組織はさらに多くの課題に直面します。

この課題に対処するために、 Red Hat は OpenShift バージョン 4.6 から始まる OpenShift Virtualization （以前のコンテナネイティブ仮想化）を導入しました。OpenShift Virtualization 機能を使用すると、同じ OpenShift Container Platform インストール上でコンテナとともに仮想マシンを実行および管理できるため、オペレータを介して VM の導入と管理を自動化するハイブリッド管理機能が提供されます。OpenShift Virtualization では、 OpenShift で VM を作成するだけでなく、 VMware vSphere 、 Red Hat Virtualization 、 Red Hat OpenStack Platform の各環境からの VM のインポートもサポートします。



NetApp ONTAP をベースにした OpenShift Virtualization では、ライブ VM 移行、 VM ディスククローニング、 VM スナップショットなどの一部の機能がサポートされており、 Astra Trident の支援を受けています。それぞれのセクションで、このドキュメントの後半で各ワークフローの例について説明します。

Red Hat OpenShift Virtualization の詳細については、のドキュメントを参照してください "こちらをご覧ください"。

OpenShift仮想化の導入

NetApp ONTAP を使用して Red Hat OpenShift Virtualization を導入します

前提条件

- Red Hat OpenShift クラスタ（バージョン 4.6 以降） RHCOS ワーカーノードを使用するベアメタルインフラストラクチャにインストールします
- OpenShift クラスタは、インストーラでプロビジョニングされたインフラを介してインストールする必要があります （ IPI ）
- VM の HA を維持するには、マシンの健全性チェックを導入します
- NetApp ONTAP クラスタ
- OpenShift クラスタに Trident の Astra をインストール
- ONTAP クラスタの SVM で設定された Trident バックエンド
- OpenShift クラスタ上でストレージクラスを構成し、 Astra Trident をプロビジョニングツールとして提供
- Red Hat OpenShift クラスタへのクラスタ管理者アクセス
- NetApp ONTAP クラスタへの管理者アクセス
- tridentctl および OC ツールがインストールされている管理ワークステーション $PATH に追加されました

OpenShift Virtualization は、 OpenShift クラスタにインストールされたオペレータによって管理されるため、メモリ、 CPU 、およびストレージに追加のオーバーヘッドが発生します。このオーバーヘッドは、クラスタのハードウェア要件を計画する際に考慮する必要があります。のドキュメントを参照してください "こちらをご覧ください" 詳細：

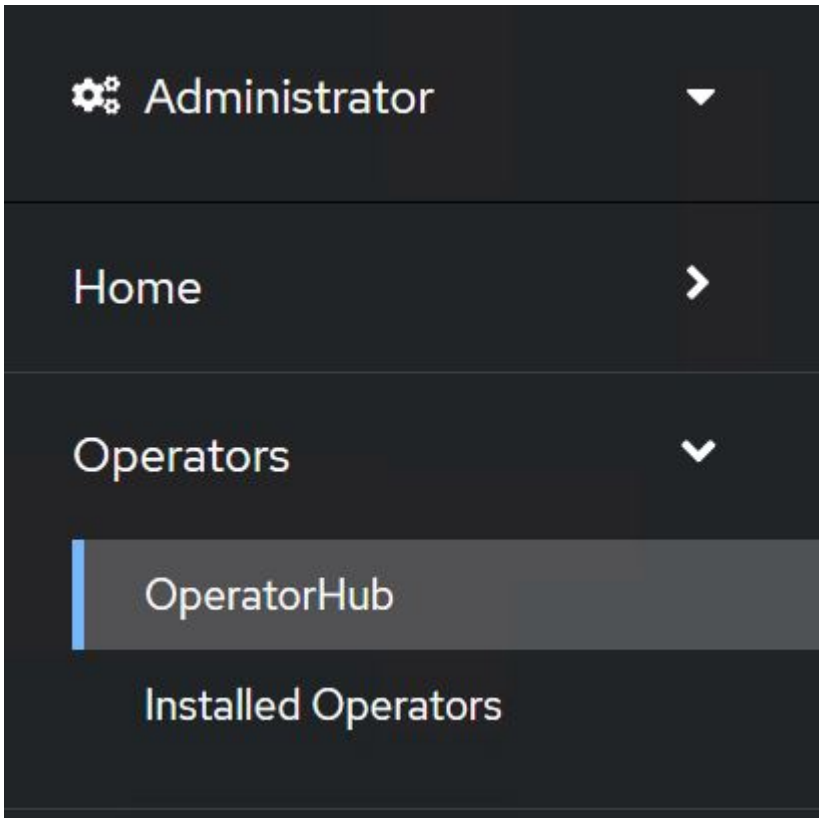
ノード配置ルールを設定して、 OpenShift Virtualization オペレータ、コントローラ、 VM をホストする OpenShift クラスタノードのサブセットを指定することもできます。OpenShift Virtualization のノード配置ルールを設定するには、のドキュメントに従ってください "こちらをご覧ください"。

OpenShift Virtualization を基盤とするストレージについては、特定の Trident バックエンドからストレージを要求する専用のストレージクラスを用意し、そのストレージクラスを専用の SVM でバックアップすることを推奨します。これにより、 OpenShift クラスタ上で VM ベースのワークロードに提供されるデータに関して、レベルのマルチテナンシーが維持されます。

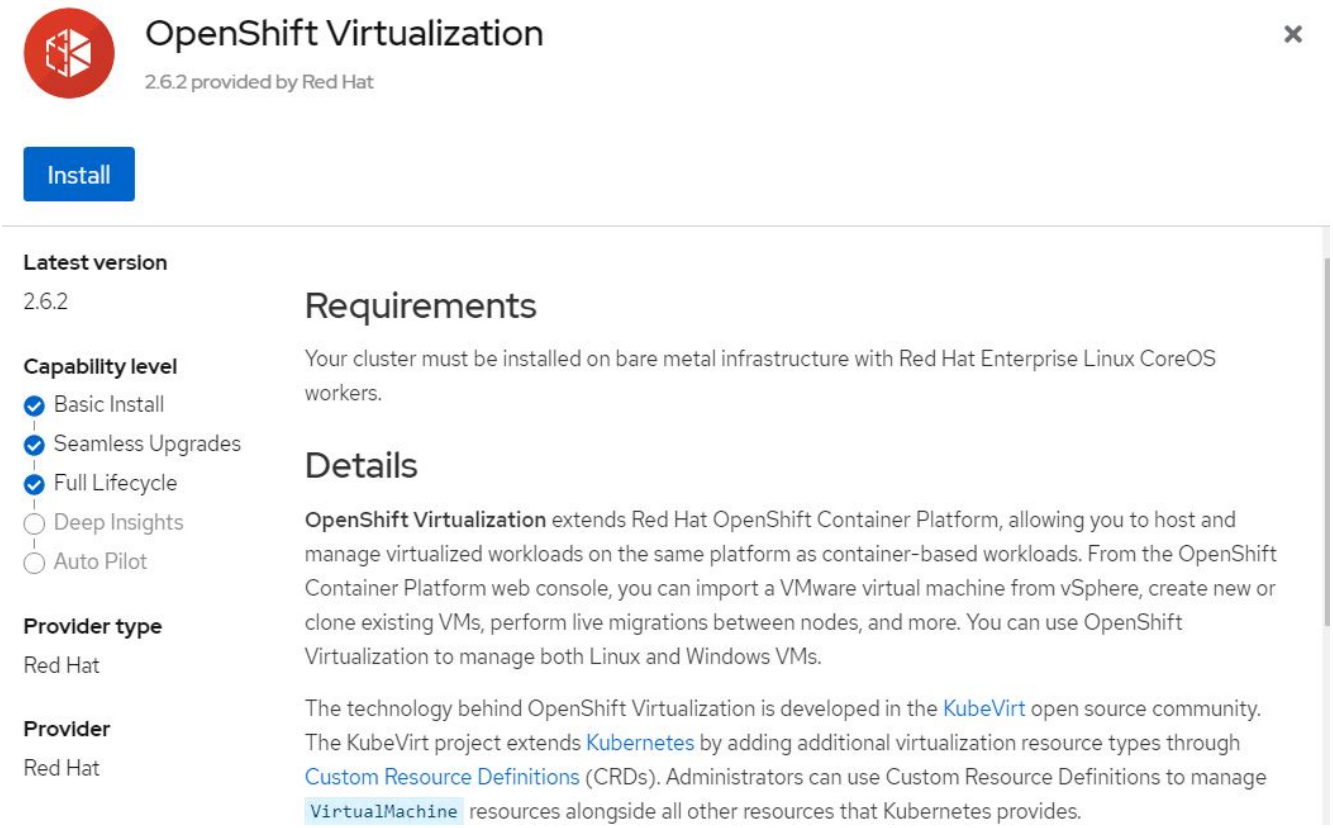
NetApp ONTAP を使用して Red Hat OpenShift Virtualization を導入します

OpenShift Virtualization をインストールするには、次の手順を実行します。

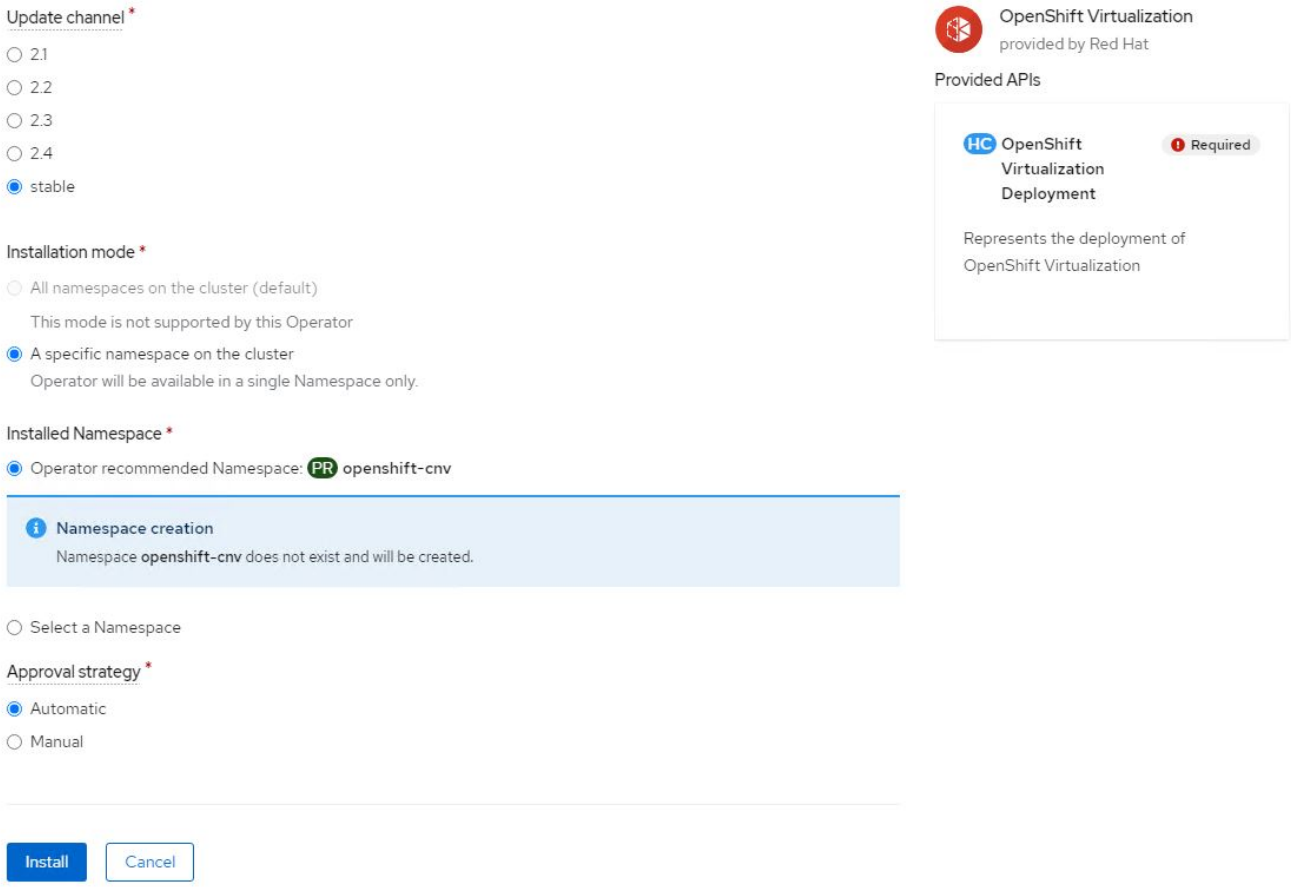
1. クラスタ管理者アクセス権を持つ Red Hat OpenShift ベアメタルクラスタにログインします。
2. Perspective ドロップダウンから Administrator を選択します。
3. Operators > OperatorHub に移動して、 OpenShift Virtualization を検索します。



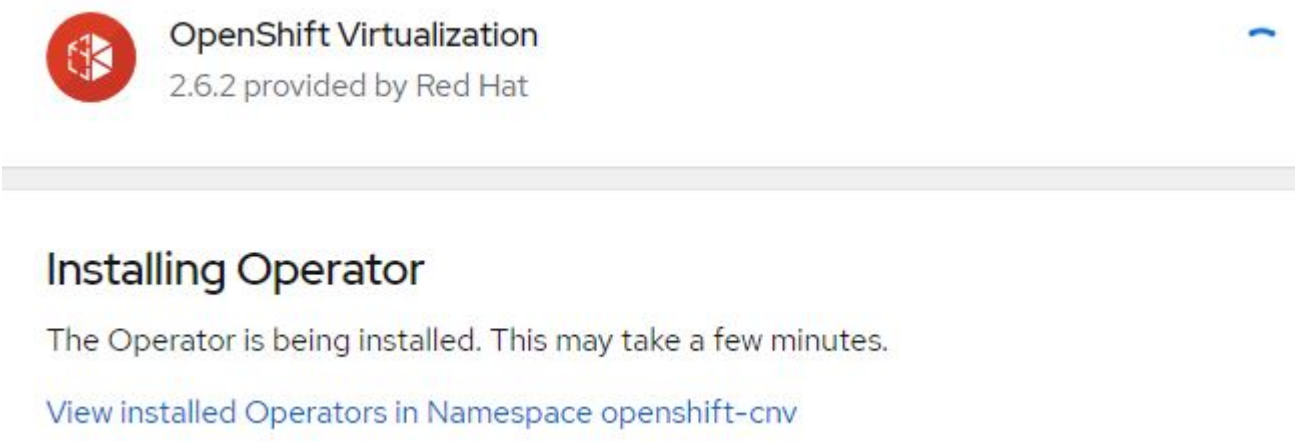
4. OpenShift Virtualization タイルを選択し、 Install をクリックします。



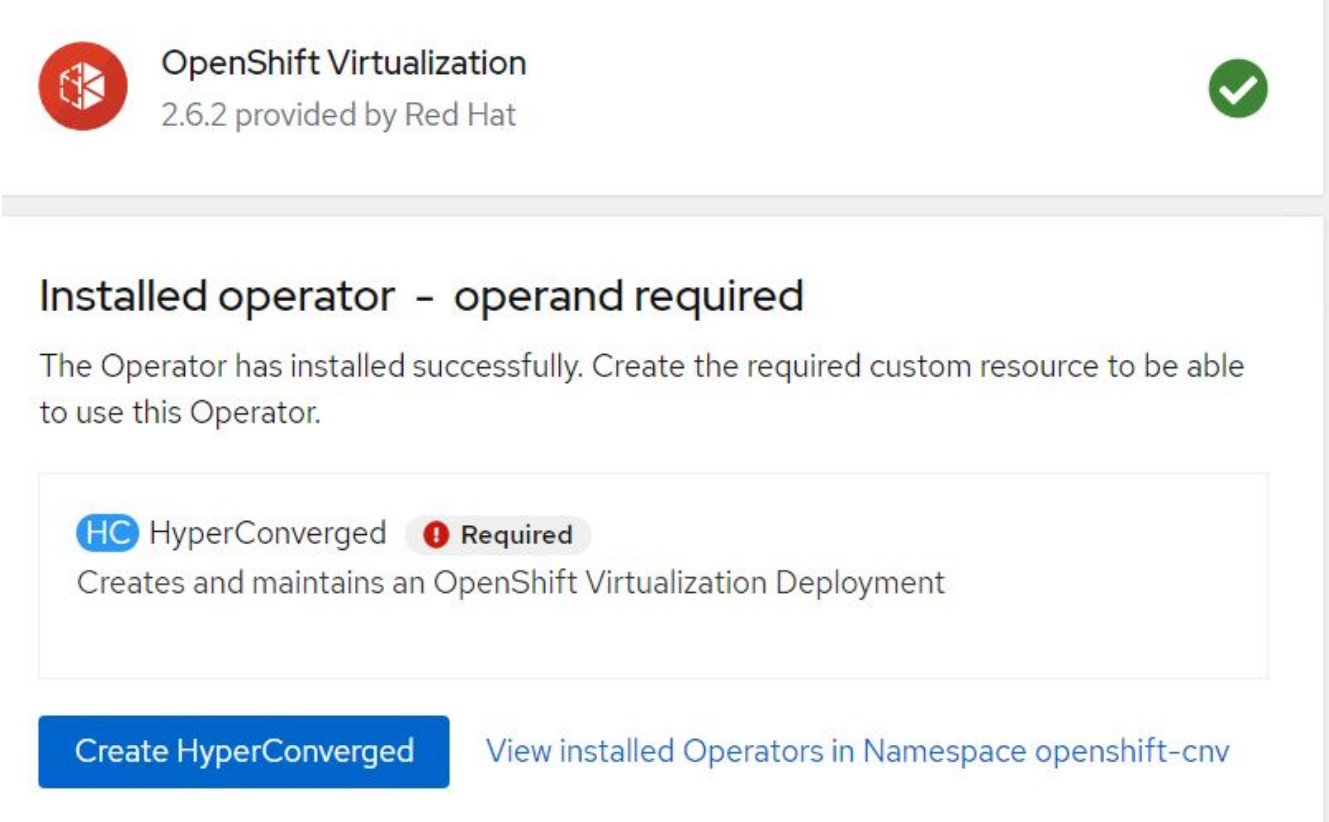
5. Install Operator （オペレータのインストール）画面で、デフォルトのパラメータをすべてそのままにして、 Install （インストール）をクリックします。



6. オペレータによるインストールが完了するまで待ちます。



7. オペレータがインストールされたら、 Create HyperConverged をクリックします。



8. [Create HyperConverged (ハイパーコンバージドの作成)] 画面で、 [Create (作成)] をクリックし、すべてのデフォルトパラメータを受け入れます。このステップでは、 OpenShift Virtualization のインストールを開始します。

Name *

kubevirt-hyperconverged

Labels

app=frontend

Infra

infra HyperConvergedConfig influences the pod configuration (currently only placement) for all the infra components needed on the virtualization enabled cluster but not necessarely directly on each node running VMs/VMIs.

Workloads

workloads HyperConvergedConfig influences the pod configuration (currently only placement) of components which need to be running on a node where virtualization workloads should be able to run. Changes to Workloads HyperConvergedConfig can be applied only without existing workload.

Bare Metal Platform

true

BareMetalPlatform indicates whether the infrastructure is baremetal.

Feature Gates

featureGates is a map of feature gate flags. Setting a flag to `true` will enable the feature. Setting `false` or removing the feature gate, disables the feature.

Local Storage Class Name

LocalStorageClassName the name of the local storage class.

Create Cancel

9. OpenShift CNV ネームスペースですべてのポッドが running 状態に移行し、 OpenShift Virtualization オペレータが Succeeded 状態になると、オペレータは使用可能な状態になります。これで、 OpenShift クラスタで VM を作成できるようになります。

Project: openshift-cnv

Installed Operators

Installed Operators are represented by ClusterServiceVersions within this Namespace. For more information, see the Understanding Operators documentation. Or create an Operator and ClusterServiceVersion using the Operator SDK.

Name Search by name...

Name	Managed Namespaces	Status	Last updated	Provided APIs
OpenShift Virtualization 2.6.2 provided by Red Hat	NS openshift-cnv	Succeeded Up to date	May 18, 8:02 pm	OpenShift Virtualization Deployment HostPathProvisioner deployment

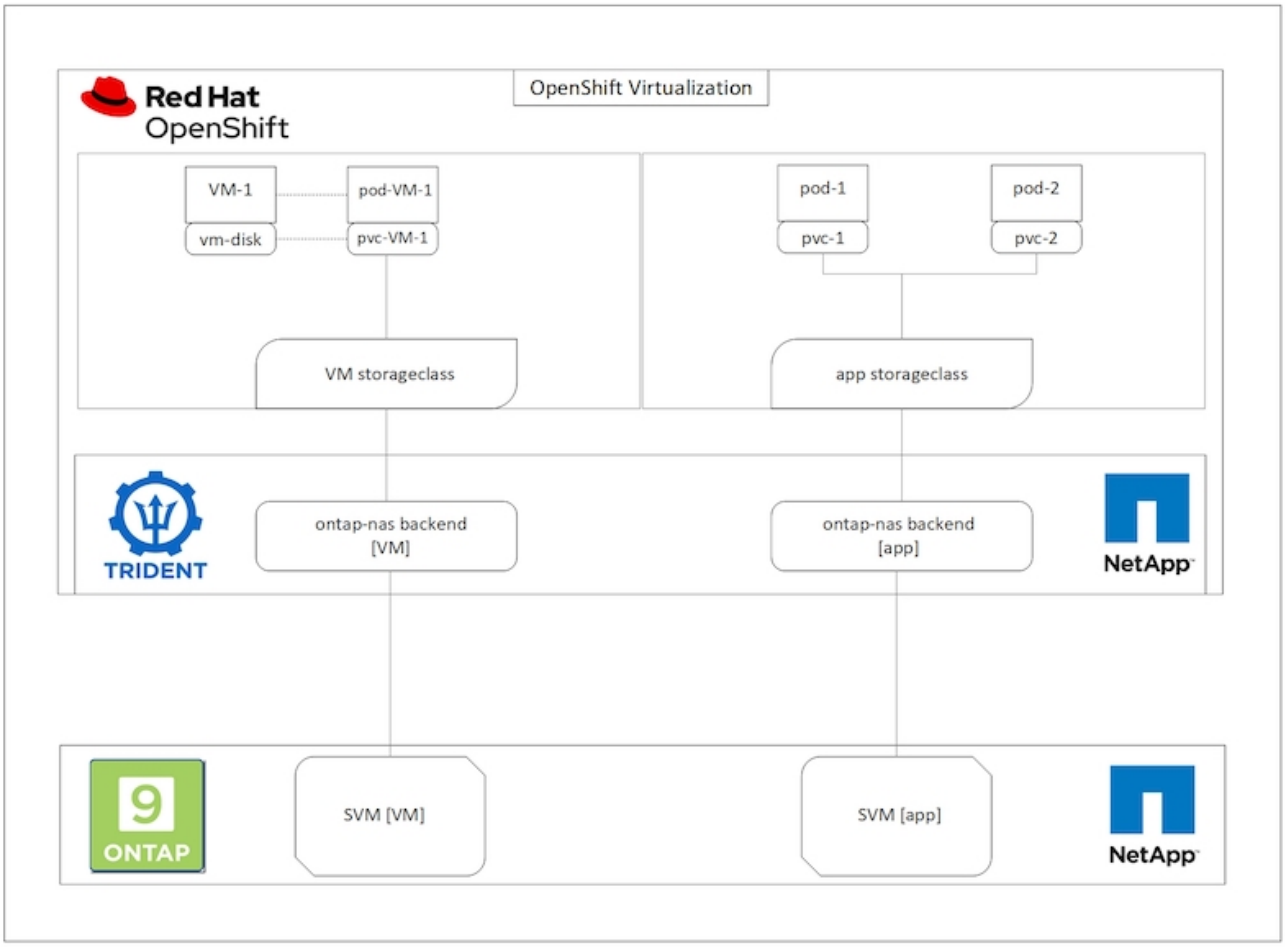
ワークフロー

ワークフロー： **NetApp ONTAP** を使用した **Red Hat OpenShift Virtualization**

VM を作成します

VM は、オペレーティングシステムとデータをホストするボリュームを必要とするステートフルな導入で

す。CNV では、 VM がポッドとして実行されるため、 VM は Trident 経由で NetApp ONTAP にホストされた PVS によってバックアップされます。これらのボリュームはディスクとして接続され、 VM のブートソースを含むファイルシステム全体が格納されます。



OpenShift クラスタ上に仮想マシンを作成するには、次の手順を実行します。

1. ワークロード > 仮想化 > 仮想マシンと進み、作成 > ウィザードを使用してをクリックします。
2. 目的の OS を選択し、 Next （次へ）をクリックします。
3. 選択したオペレーティングシステムにブートソースが設定されていない場合は、設定する必要があります。Boot Source （起動ソース）で、 URL またはレジストリから OS イメージをインポートするかどうかを選択し、対応する詳細を指定します。Advanced を展開し、 Trident から作成されたストレージクラスを選択します。[次へ] をクリックします。

Boot source

This template does not have a boot source. Provide a custom boot source for this **CentOS 8.0+ VM** virtual machine.

Boot source type *

Import via URL (creates PVC)

Import URL *

https://access.cdn.redhat.com/content/origin/files/sha256/58/588167f828001e57688ec4b9b31c11a59d532489f527488ebc89ac5e952...

Example: For RHEL, visit the **RHEL download page** (requires login) and copy the download link URL of the KVM guest image

☑ Mount this as a CD-ROM boot source

Persistent Volume Claim size *

5 GiB

Ensure your PVC size covers the requirements of the uncompressed image and any other space requirements. More storage can be added later.

Advanced

Storage class *

basic (default)

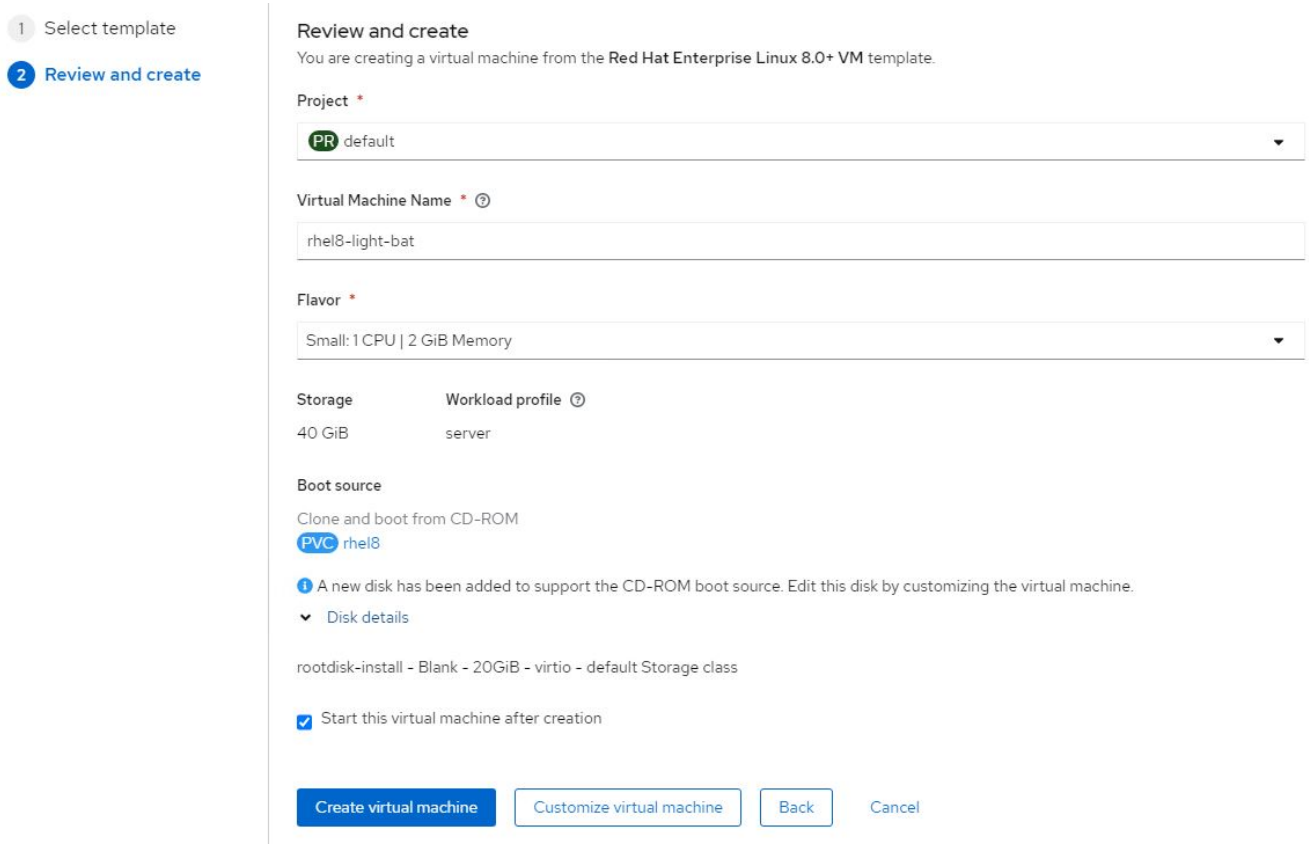
Access mode *

Single User (RWO)

Volume mode *

Filesystem

4. 選択したオペレーティングシステムにすでにブートソースが設定されている場合は、前の手順を省略できます。
5. [Review and Create] ペインで、 VM を作成して VM の詳細を提供するプロジェクトを選択します。ブートソースがクローンとして選択されていることを確認し、選択した OS に適切な PVC が割り当てられた CD-ROM から起動します。



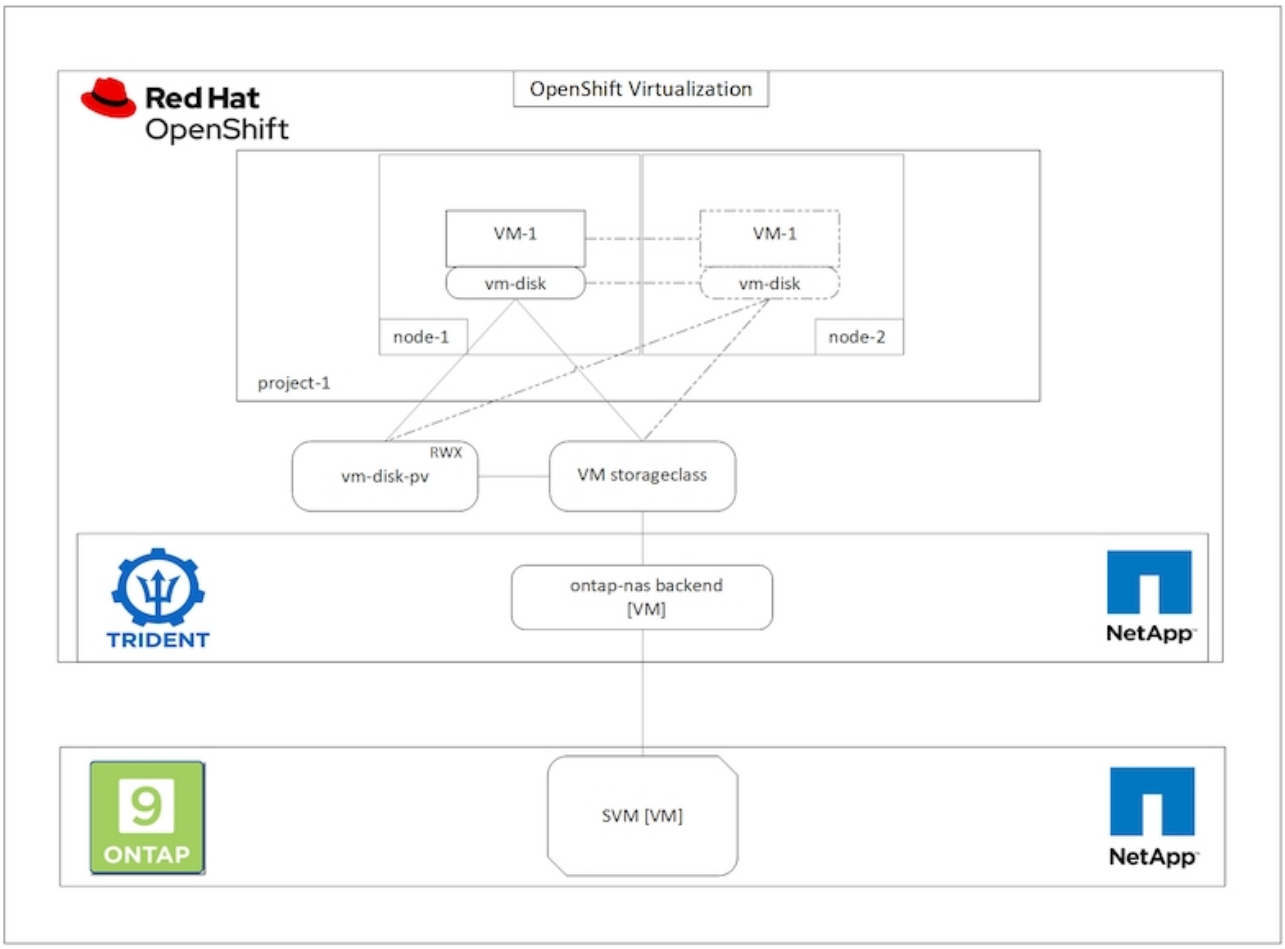
6. 仮想マシンをカスタマイズする場合は、 [仮想マシンのカスタマイズ] をクリックし、必要なパラメータを変更します。
7. [仮想マシンの作成] をクリックして仮想マシンを作成します。これにより、対応するポッドがバックグラウンドでスピンアップされます。

ブート・ソースが URL またはレジストリからテンプレートまたはオペレーティング・システム用に構成されている場合 'OpenShift Virtualization-os-images' プロジェクトに PVC を作成し 'KVM ゲスト・イメージを PVC にダウンロードしますテンプレート PVC に、対応する OS の KVM ゲストイメージを格納できるだけの十分なプロビジョニングスペースがあることを確認する必要があります。これらの PVC は、任意のプロジェクトでそれぞれのテンプレートを使用して作成されると、クローン作成され、ルートディスクとして仮想マシンに接続されます。

ワークフロー： **NetApp ONTAP** を使用した **Red Hat OpenShift Virtualization**

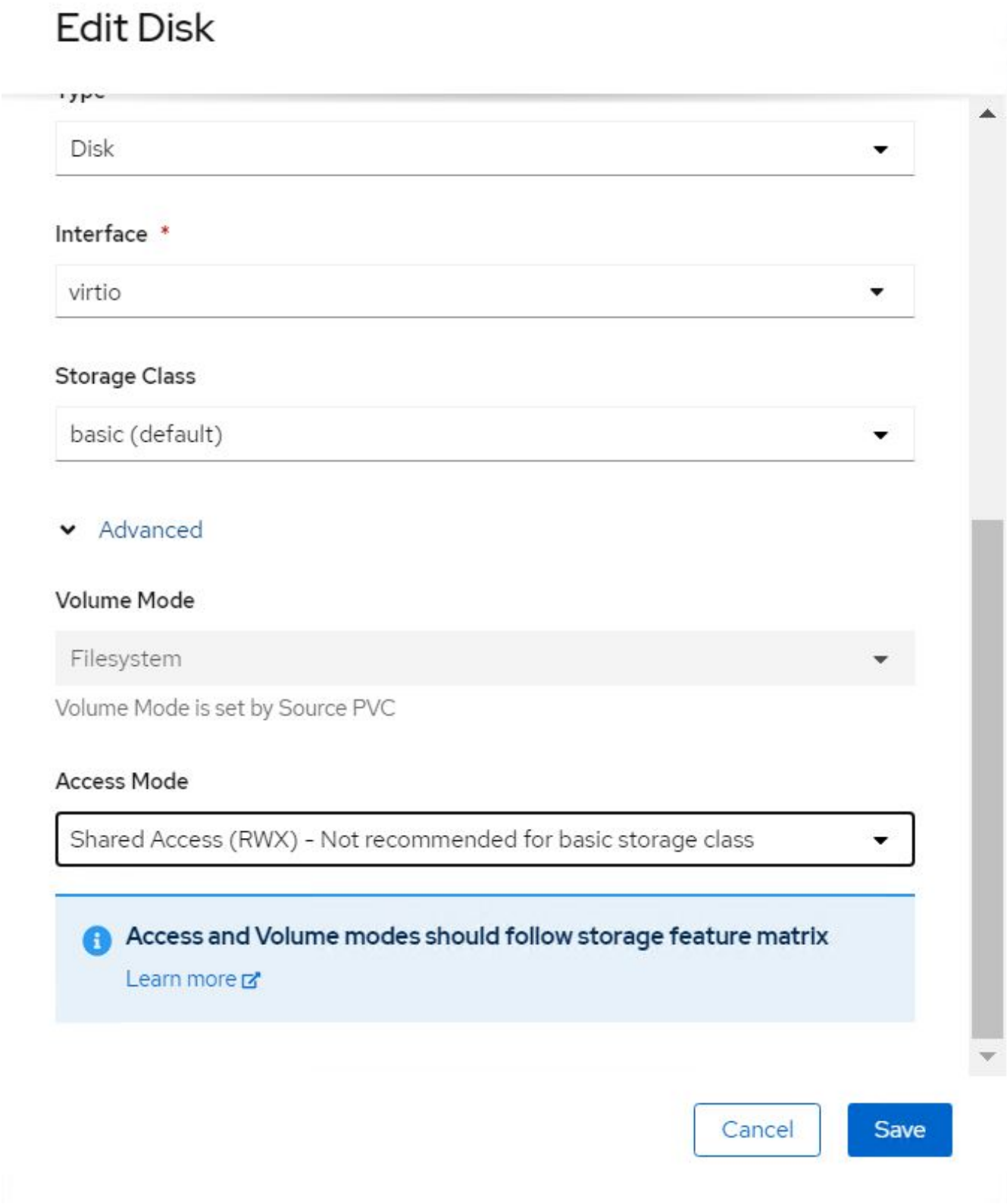
VM ライブマイグレーション

ライブマイグレーションは、 OpenShift クラスタ内の 1 つのノードから別のノードに VM インスタンスをダウンタイムなしで移行するプロセスです。OpenShift クラスタでライブマイグレーションを実行するには、 VM を共有 ReadWriteAny アクセスモードの PVC にバインドする必要があります。NFS プロトコルが有効になっている NetApp ONTAP クラスタで SVM を使用して設定された Astra Trident バックエンドは、 PVC に対する共有 ReadWriteAny アクセスをサポートします。そのため、 NFS 対応の SVM から Trident によってプロビジョニングされた StorageClasses から要求された PVC を持つ VM をダウンタイムなしで移行できます。



共有 ReadWriteAny アクセス権を持つ PVC にバインドされた VM を作成するには、次の手順を実行します。

1. ワークロード > 仮想化 > 仮想マシンと進み、作成 > ウィザードを使用してをクリックします。
2. 目的の OS を選択し、 Next （次へ）をクリックします。選択した OS には、すでに起動ソースが設定されているとしましょう。
3. [Review and Create] ペインで、 VM を作成して VM の詳細を提供するプロジェクトを選択します。ブートソースがクローンとして選択されていることを確認し、選択した OS に適切な PVC が割り当てられた CD-ROM から起動します。
4. [仮想マシンのカスタマイズ] をクリックし、 [ストレージ] をクリックします。
5. rootdisk の横にある省略記号をクリックし、 Trident を使用してプロビジョニングされたストレージクラスが選択されていることを確認します。[詳細設定] を展開し、 [アクセスモード] で [共有アクセス (RWX)] を選択します。[保存] をクリックします。



6. [確認] をクリックして確定し、 [仮想マシンの作成] をクリックします。

OpenShift クラスタ内の別のノードに VM を手動で移行するには、次の手順を実行します。

1. ワークロード > 仮想化 > 仮想マシンと進みます。

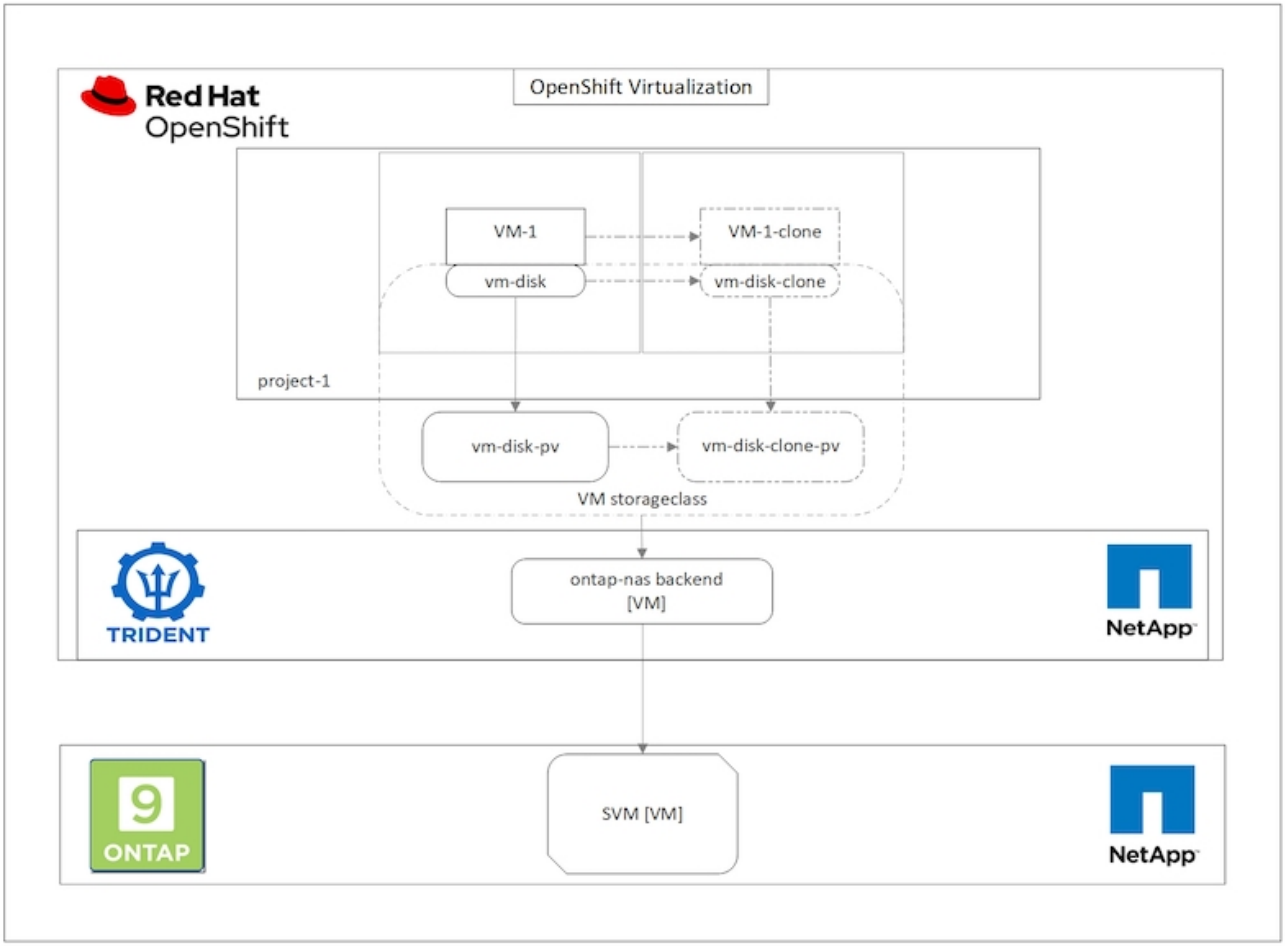
2. 移行する VM の場合は、省略記号をクリックし、 Migrate the Virtual Machine （仮想マシンの移行）をクリックします。
3. メッセージが表示されたら、 [移行] をクリックして確認します。

> OpenShift クラスタ内の VM インスタンスは、 evictionStrategy が LiveMigrate に設定されている場合、元のノードがメンテナンスモードになると、自動的に別のノードに移行します。

ワークフロー： **NetApp ONTAP** を使用した **Red Hat OpenShift Virtualization**

VM のクローニング

OpenShift で既存の VM をクローニングするには、 Astra Trident の Volume CSI クローニング機能をサポートします。CSI ボリュームクローニングでは、 PV を複製することによって、既存の PVC をデータソースとして使用して新しい PVC を作成できます。新しい PVC が作成されると、その PVC は独立したエンティティとして機能し、送信元 PVC へのリンクや依存関係はありません。



CSI ボリュームクローニングには、次のような一定の制限事項があります。

1. 送信元 PVC と宛先 PVC は同じプロジェクト内に存在する必要があります。
2. クローニングは、同じストレージクラス内でサポートされます。
3. クローニングを実行できるのは、ソースボリュームとデスティネーションボリュームで同じボリュームモ

ード設定を使用している場合のみです。たとえば、ブロックボリュームは別のブロックボリュームにしかクローニングできません。

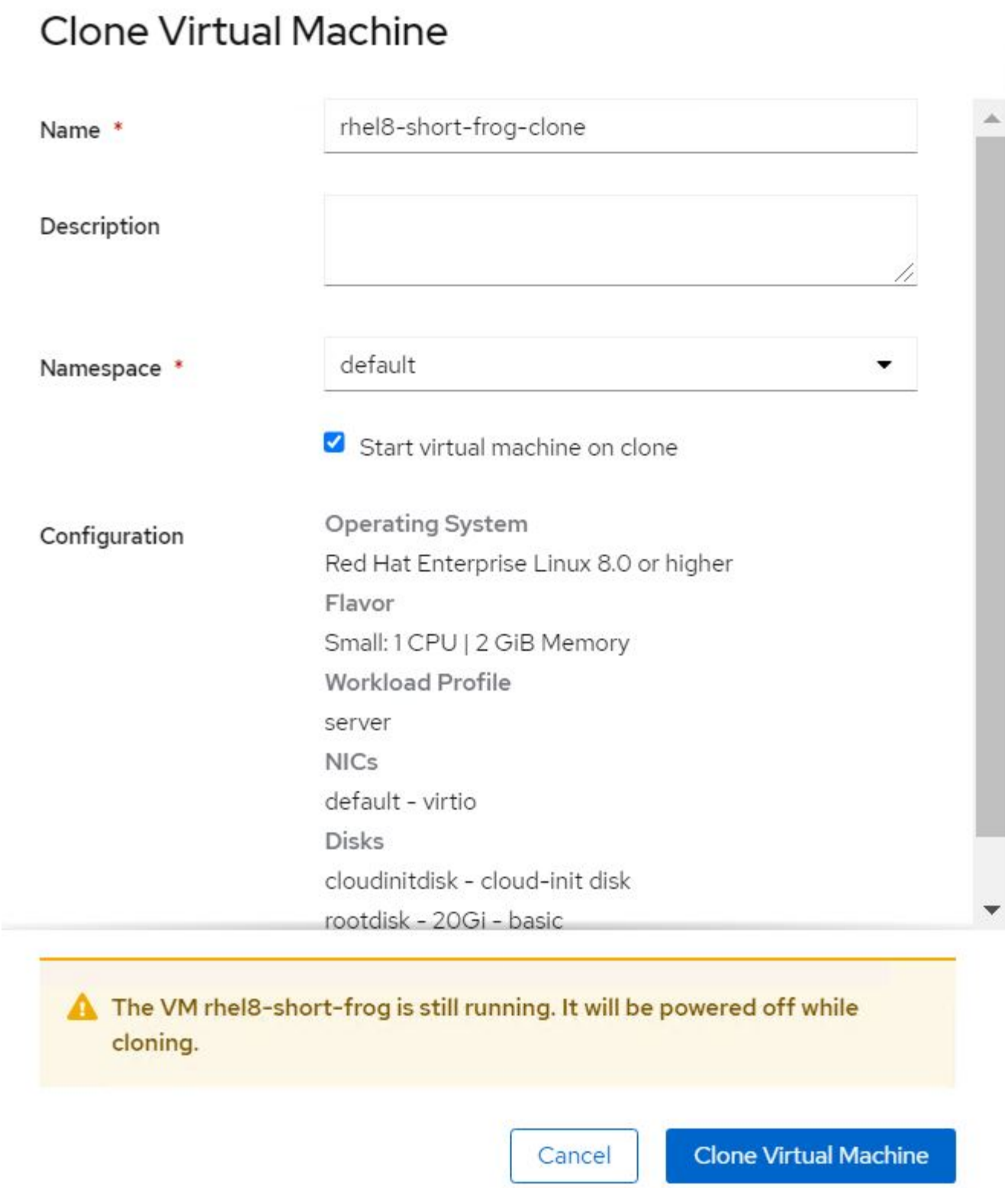
OpenShift クラスタ内の VM は、次の 2 つの方法でクローニングできます。

1. ソース VM をシャットダウンします
2. ソース VM を稼働させます

ソース **VM** をシャットダウンします

VM をシャットダウンして既存の VM をクローニングすることは、 Astra Trident のサポートとともに実装されるネイティブの OpenShift 機能です。VM をクローニングするには、次の手順を実行します。

1. [Workloads （ワークロード）] > [Virtualization （仮想化）] > [Virtual Machines （仮想マシン）] に移動し、クローンを作成する仮想マシンの横にある省略記号をクリックします。
2. Clone Virtual Machine をクリックして、新しい VM の詳細を指定します。

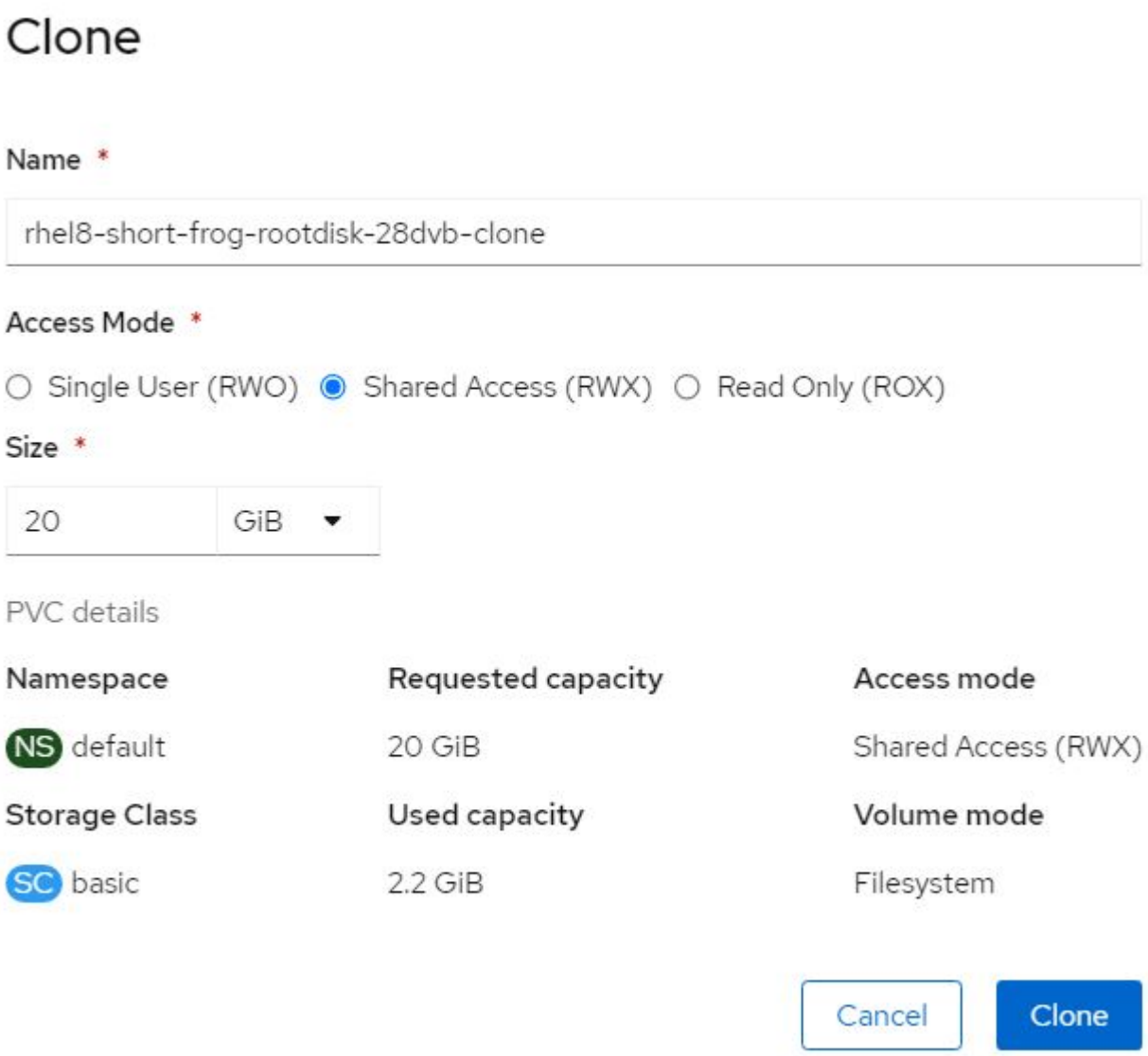


3. Clone Virtual Machine をクリックします。これにより、ソース VM がシャットダウンされ、クローン VM の作成が開始されます。
4. この手順が完了すると、クローニングした VM のコンテンツにアクセスして確認できるようになります。

ソース **VM** を稼働させます

既存の VM は、ソース VM の既存の PVC をクローニングしてから、クローン PVC を使用して新しい VM を作成することによってもクローニングできます。この方法では、ソース VM をシャットダウンする必要はありません。シャットダウンせずに VM をクローニングするには、次の手順を実行します。

1. Storage > PersistentVolume要求 と進み、ソース VM に接続されている PVC の横にある省略記号をクリックします。
2. Clone PVC をクリックして、新しい PVC の詳細を提供します。



3. 次に、 Clone をクリックします。これにより、新しい VM の PVC が作成されます。
4. [Workloads （ワークロード）] > [Virtualization （仮想化）] > [Virtual Machines （仮想マシン）] に移動し、 [Create （作成）
5. spec> template> spec> volumes セクションで、コンテナディスクではなく、クローン PVC を接続します。新しい VM について、要件に応じてその他の詳細をすべて指定します。

```
- name: rootdisk
  persistentVolumeClaim:
    claimName: rhel8-short-frog-rootdisk-28dvb-clone
```

6. Create をクリックして、新しい VM を作成します。
7. VM が作成されたら、にアクセスし、新しい VM がソース VM のクローンであることを確認します。

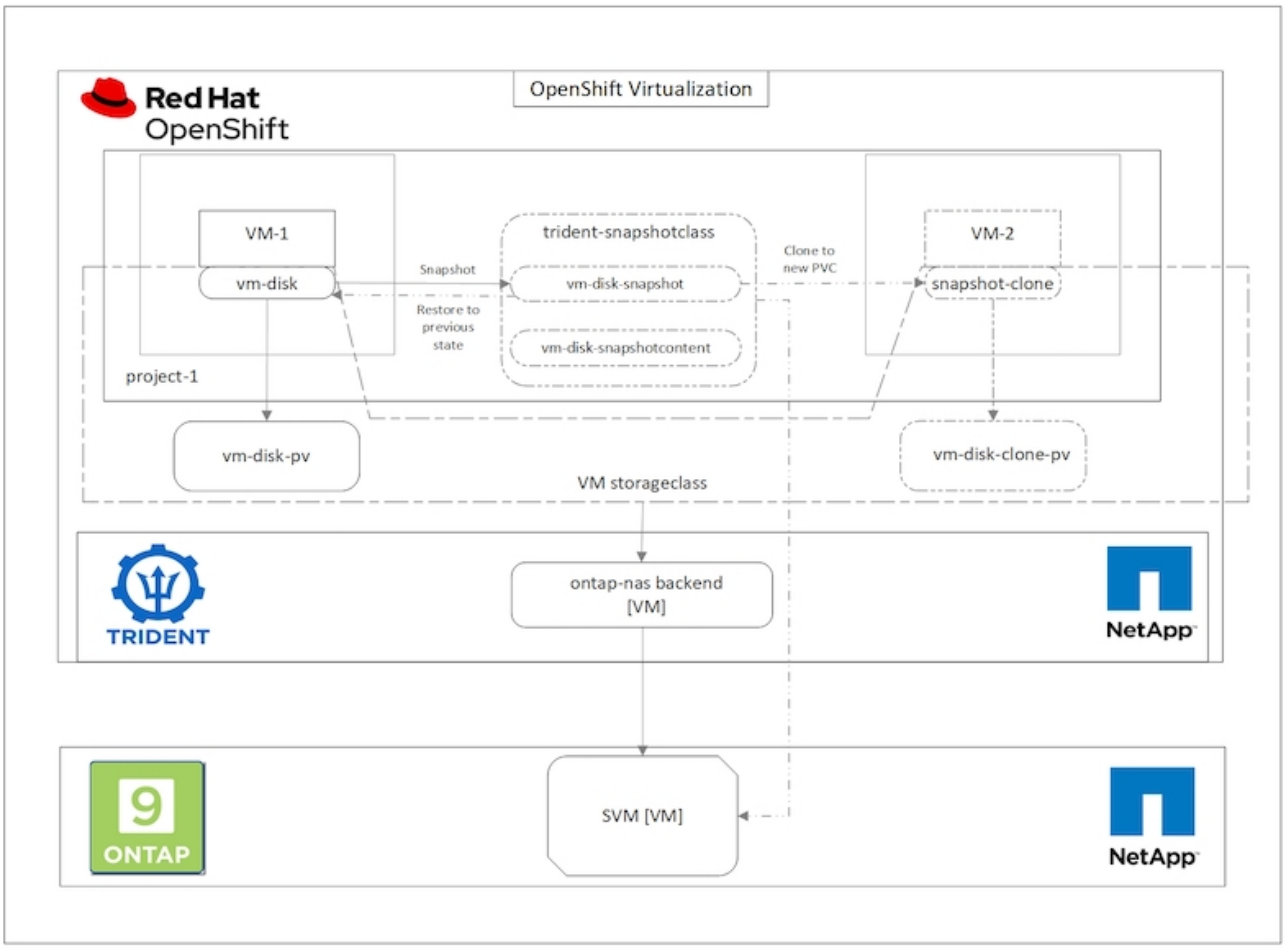
ワークフロー： **NetApp ONTAP** を使用した **Red Hat OpenShift Virtualization**

Snapshot から **VM** を作成します

Astra Trident と Red Hat OpenShift を使用すると、ユーザは、プロビジョニングされたストレージクラス上の永続的ボリュームのスナップショットを作成できます。この機能を使用すると、ボリュームのポイントインタイムコピーを作成して、そのコピーを使用して新しいボリュームを作成したり、同じボリュームを以前の状態にリストアしたりできます。これにより、ロールバックからクローン、データリストアまで、さまざまなユースケースを実現またはサポートできます。

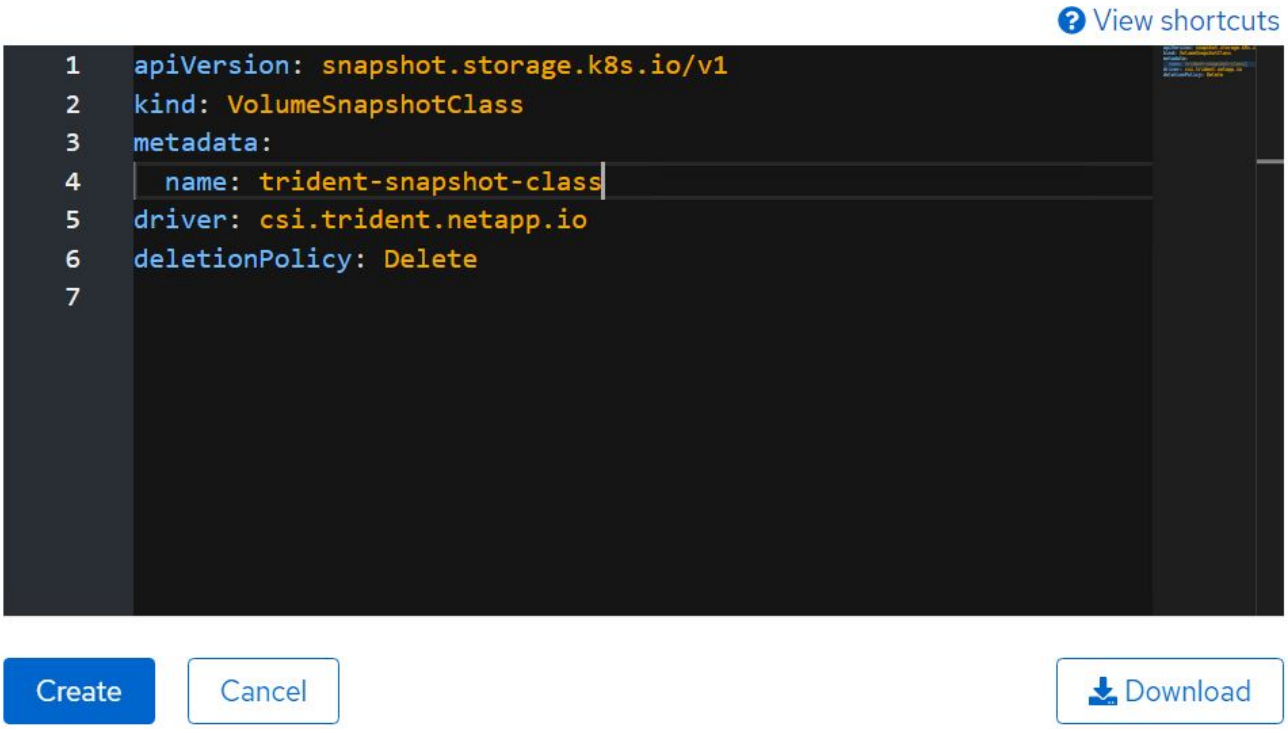
OpenShift で Snapshot 処理を実行するには、リソース VolumeSnapshotClass 、 VolumeSnapshot 、および VolumeSnapshotContent を定義する必要があります。

- VolumeSnapshotContent は、クラスタ内のボリュームから作成された実際の Snapshot です。このデータストアは、 Storage 用の PersistentVolume に似た、クラスタ全体のリソースです。
- ボリューム Snapshot は、ボリュームの Snapshot 作成要求です。これは、 PersistentVolumeClaim に似ています。
- VolumeSnapshotClass を使用すると、管理者はボリューム Snapshot のさまざまな属性を指定できます。これにより、同じボリュームから作成された異なる Snapshot に対して異なる属性を設定できます。

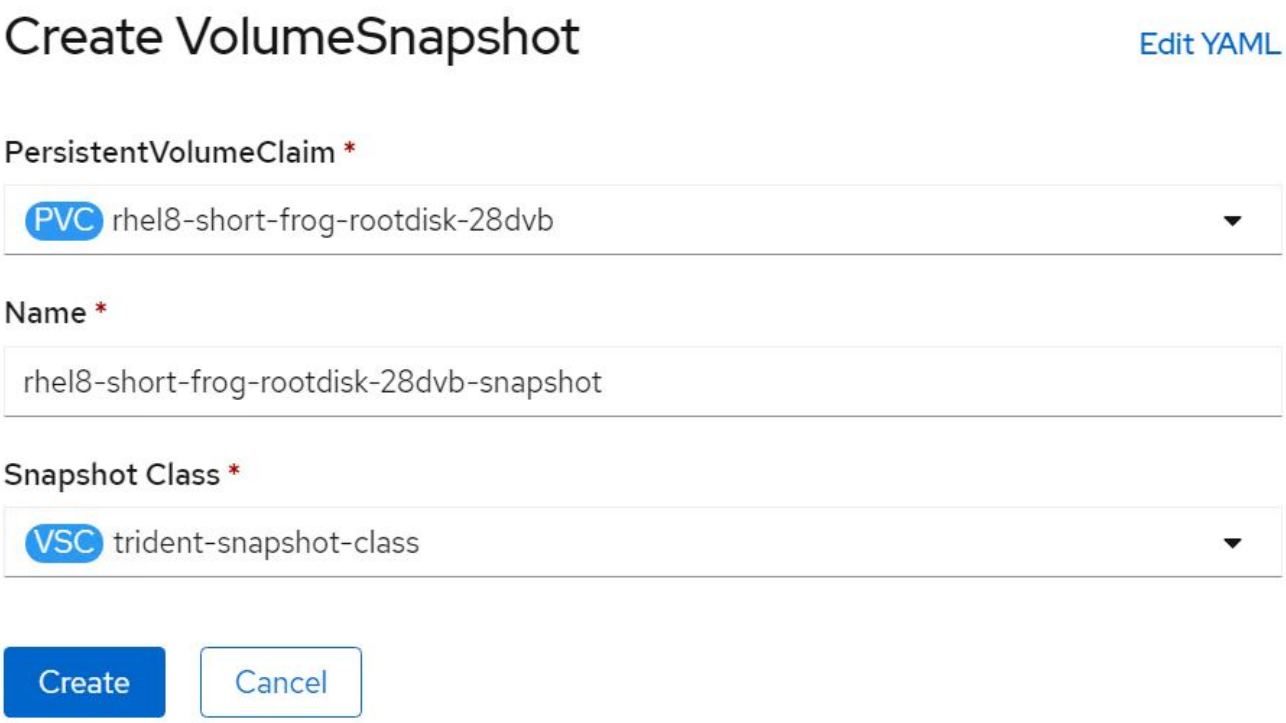


VM の Snapshot を作成するには、次の手順を実行します。

1. ボリューム Snapshot を作成するために使用できるボリューム Snapshot クラスを作成します。Storage > VolumeSnapshotClasses の順に移動し、 Create VolumeSnapshotClass をクリックします。
2. スナップショットクラスの名前を入力し、ドライバの csi.trident.netapp.io を入力して、 Create をクリックします。



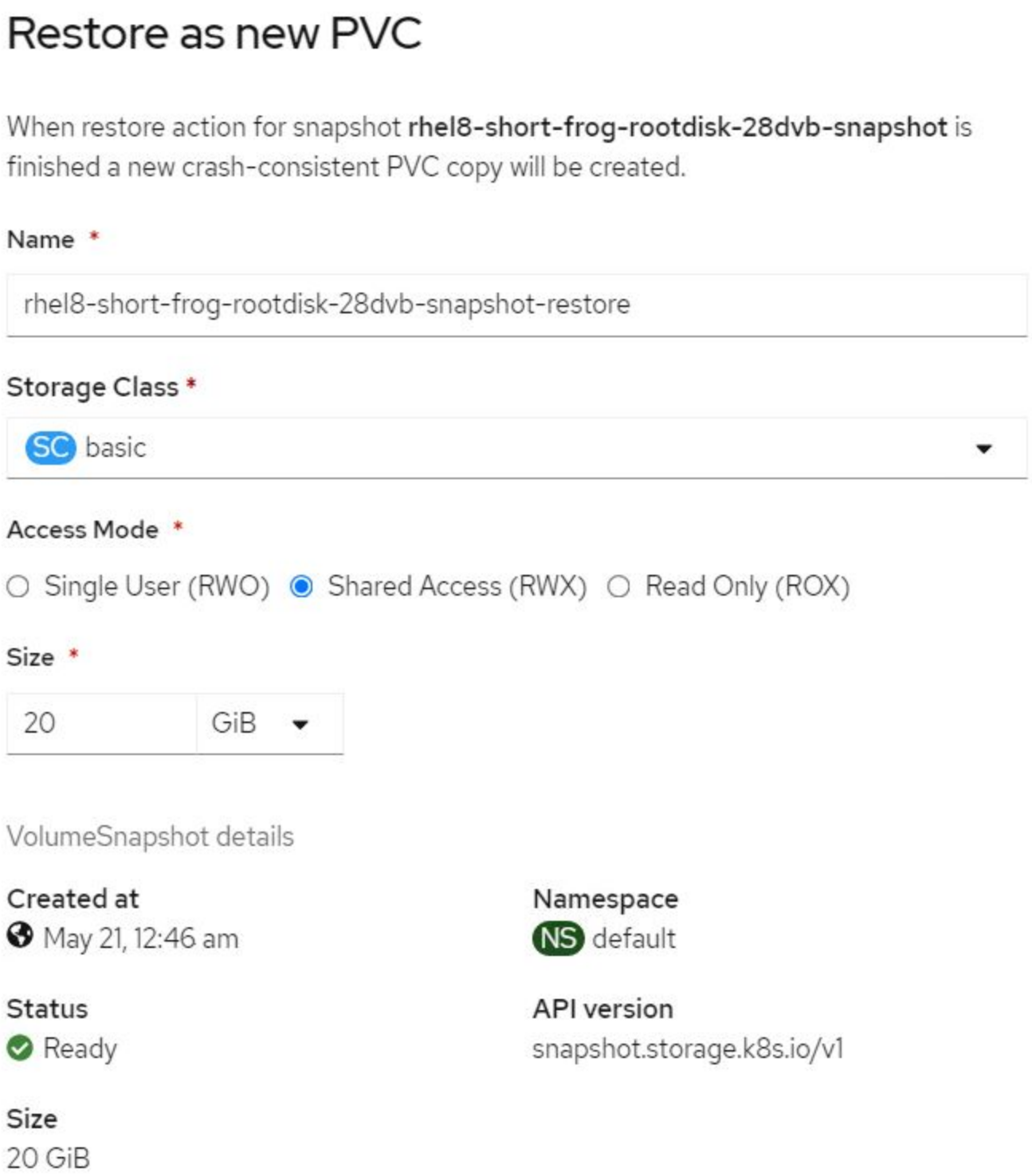
3. ソース VM に接続されている PVC を特定し、その PVC の Snapshot を作成します。「ストレージ」＞「ボリュームスナップショット」と選択し、「ボリュームスナップショットの作成」をクリックします。
4. Snapshot を作成する PVC を選択し、 Snapshot の名前を入力するか、デフォルトを受け入れて、適切な VolumeSnapshotClass を選択します。[作成] をクリックします。



5. これにより、その時点で PVC のスナップショットが作成されます。

スナップショットから新しい **VM** を作成します

1. 最初に、スナップショットを新しい PVC に復元します。Storage > VolumeSnapshots と進み、リストアする Snapshot の横にある省略記号をクリックして、 Restore as new PVC （新しい PVC として復元）をクリックします。
2. 新しい PVC の詳細を入力し、 Restore をクリックします。これにより、新しい PVC が作成されます。



3. 次に、この PVC から新しい VM を作成します。[Workloads （ワークロード）] > [Virtualization （仮想化）] > [Virtual Machines （仮想マシン）] に移動し、 [Create （作成）

4. spec>template>spec>volumes セクションで、コンテナディスクからではなく、スナップショットから作成された新しい PVC を指定します。新しい VM について、要件に応じてその他の詳細をすべて指定します。

```
- name: rootdisk
  persistentVolumeClaim:
    claimName: rhel8-short-frog-rootdisk-28dvb-snapshot-restore
```

5. Create をクリックして、新しい VM を作成します。
6. VM が正常に作成されたら、にアクセスして、新しい VM の状態が、スナップショット作成時に PVC を使用してスナップショットを作成した VM の状態と同じであることを確認します。

ワークフロー： **NetApp ONTAP** を使用した **Red Hat OpenShift Virtualization**

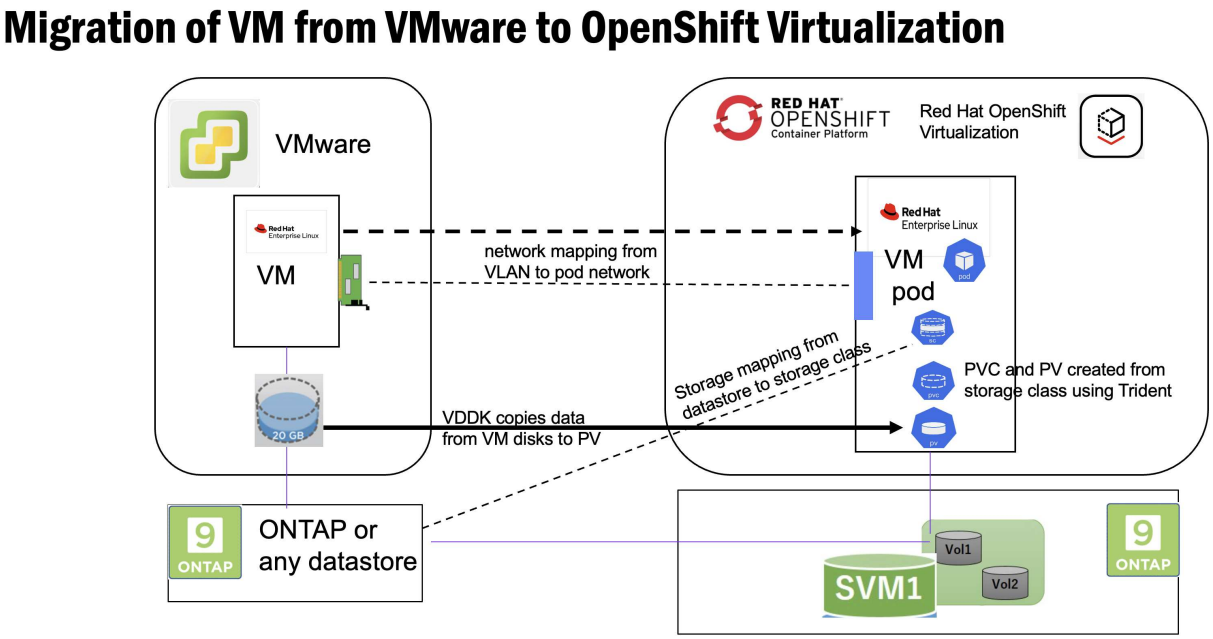
仮想化向け移行ツールキットを使用した**VMware**から**OpenShift**による仮想化への**VM**の移行

このセクションでは、仮想化向け移行ツールキット（MTV）を使用して、VMwareからOpenShift Containerプラットフォームで実行されるOpenShift仮想化に仮想マシンを移行し、Astra Tridentを使用してNetApp ONTAPストレージと統合する方法を説明します。

次のビデオでは、永続的ストレージ用にONTAP-SANを使用して、VMwareからOpenShiftによる仮想化にRHEL VMを移行するデモを示します。

Red Hat MTVを使用したNetApp ONTAPストレージによるOpenShift仮想化へのVMの移行

次の図は、VMwareからRed Hat OpenShift VirtualizationへのVMの移行の概要を示しています。



サンプル移行の前提条件

VMware環境

- 次の構成のRHEL 9.3を使用するRHEL 9 VMをインストールしました。
  - CPU：2、メモリ：20 GB、ハードディスク：20 GB
  - ユーザクレデンシャル：rootユーザとadminユーザのクレデンシャル
- VMの準備が完了したら、PostgreSQLサーバがインストールされました。
  - PostgreSQLサーバが起動され、起動時に起動できるようになりました

```
systemctl start postgresql.service`
systemctl enable postgresql.service
The above command ensures that the server can start in the VM in
OpenShift Virtualization after migration
```

  - 2つのデータベース、1つのテーブル、および1つの行が追加されました。を参照してください "こちらをご覧ください" RHELにPostgreSQLサーバをインストールし、データベースエントリとテーブルエントリを作成する手順については、を参照してください。

> PostgreSQLサーバを起動し、起動時にサービスを開始できるようにしてください。

OpenShiftクラスタ上

MTVをインストールする前に、次のインストールが完了しました。

- OpenShiftクラスタ4.13.34
- "Astra Trident 23.10"
- クラスタノードのマルチパスがiSCSIに対して有効になっている（ONTAP-SANストレージクラス用）。クラスタ内の各ノードでiSCSIを有効にするデーモンセットを作成するには、提供されているYAMLを参照してください。
- iSCSIを使用するONTAP SAN向けのTridentバックエンドおよびストレージクラス。Tridentバックエンドとストレージクラス用に提供されているYAMLファイルを参照してください。
- "OpenShift 仮想化"

OpenShiftクラスタノードにiSCSIとマルチパスをインストールするには、以下のYAMLファイルを使用します。
クラスタノードを**iSCSI**用に準備しています

```yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  namespace: trident
  name: trident-iscsi-init
  labels:
    name: trident-iscsi-init
spec:
```

```
  selector:
    matchLabels:
      name: trident-iscsi-init
  template:
    metadata:
      labels:
        name: trident-iscsi-init
    spec:
      hostNetwork: true
      serviceAccount: trident-node-linux
      initContainers:
      - name: init-node
        command:
          - nsenter
          - --mount=/proc/1/ns/mnt
          - --
          - sh
          - -c
        args: ["$(STARTUP_SCRIPT)"]
        image: alpine:3.7
        env:
        - name: STARTUP_SCRIPT
          value: |
            #! /bin/bash
            sudo yum install -y lsscsi iscsi-initiator-utils sg3_utils
device-mapper-multipath
            rpm -q iscsi-initiator-utils
            sudo sed -i 's/^\(node.session.scan\).*/\1 = manual/'
/etc/iscsi/iscsid.conf
            cat /etc/iscsi/initiatorname.iscsi
            sudo mpathconf --enable --with_multipathd y --find_multipaths
n
            sudo systemctl enable --now iscsid multipathd
            sudo systemctl enable --now iscsi
        securityContext:
          privileged: true
      hostPID: true
      containers:
      - name: wait
        image: k8s.gcr.io/pause:3.1
      hostPID: true
      hostNetwork: true
      tolerations:
      - effect: NoSchedule
        key: node-role.kubernetes.io/master
  updateStrategy:
```

```
    type: RollingUpdate
```

次のYAMLファイルを使用して、ONTAP SANストレージを使用するためのTridentバックエンド構成を作成
iSCSI向けTridentバックエンド

```
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-ontap-san-secret
type: Opaque
stringData:
  username: <username>
  password: <password>
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: ontap-san
spec:
  version: 1
  storageDriverName: ontap-san
  managementLIF: <management LIF>
  backendName: ontap-san
  svm: <SVM name>
  credentials:
    name: backend-tbc-ontap-san-secret
```

次のYAMLファイルを使用して、ONTAP SANストレージを使用するためのTridentストレージクラス構成を作成
iSCSI用のTridentストレージクラス

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-san
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  media: "ssd"
  provisioningType: "thin"
  snapshots: "true"
allowVolumeExpansion: true
```

*** MTVのインストール***

これで、Migration Toolkit for Virtualization（MTV）をインストールできます。付属の説明書を参照してください。 "こちらをご覧ください" を参照してください。

Migration Toolkit for Virtualization（MTV）ユーザーインターフェイスは、OpenShift Webコンソールに統合されています。
次を参照できます。 "こちらをご覧ください" さまざまなタスクのユーザーインターフェイスの使用を開始します。

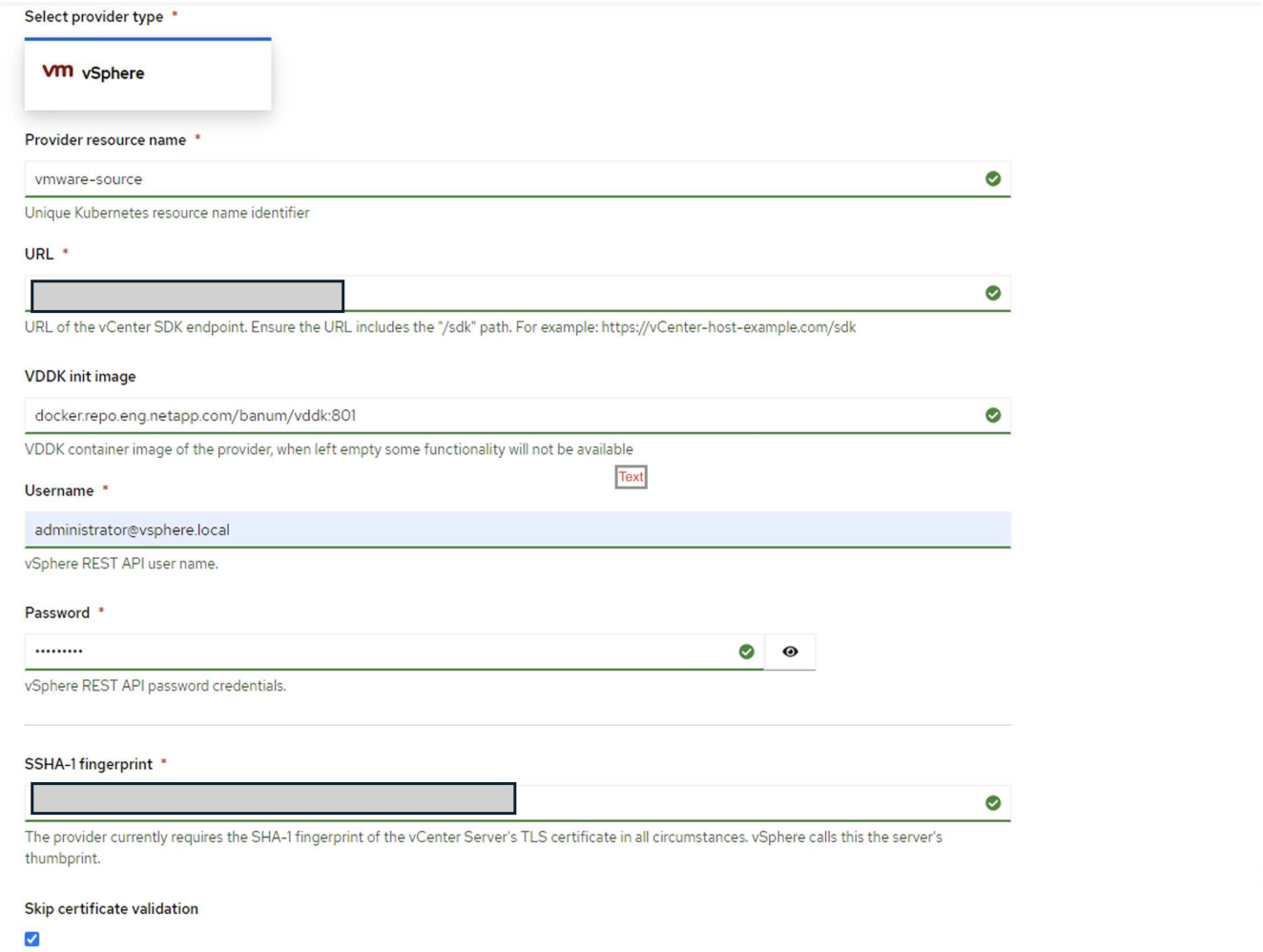
ソースプロバイダの作成

RHEL VMをVMwareからOpenShift Virtualizationに移行するには、まずVMwareのソースプロバイダを作成する必要があります。手順を参照してください "こちらをご覧ください" ソースプロバイダを作成します。

VMwareソースプロバイダを作成するには、次のものが必要です。

- vCenter URL
- vCenterクレデンシャル
- vCenter Serverサムプリント
- リポジトリ内のVDDKイメージ

ソースプロバイダの作成例：



> Migration Toolkit for Virtualization（MTV）では、VMware Virtual Disk Development Kit（VDDK）SDKを使用して、VMware vSphereからの仮想ディスクの転送を高速化します。そのため、VDDKイメージはオプションですが作成することを強くお勧めします。
> この機能を使用するには、VMware Virtual Disk Development Kit（VDDK）をダウンロードし、VDDKイメージをビルドして、VDDKイメージをイメージレジストリにプッシュします。

表示される指示に従います。 "こちらをご覧ください" VDDKイメージを作成して、OpenShiftクラスタからアクセス可能なレジストリにプッシュします。

送信先プロバイダの作成

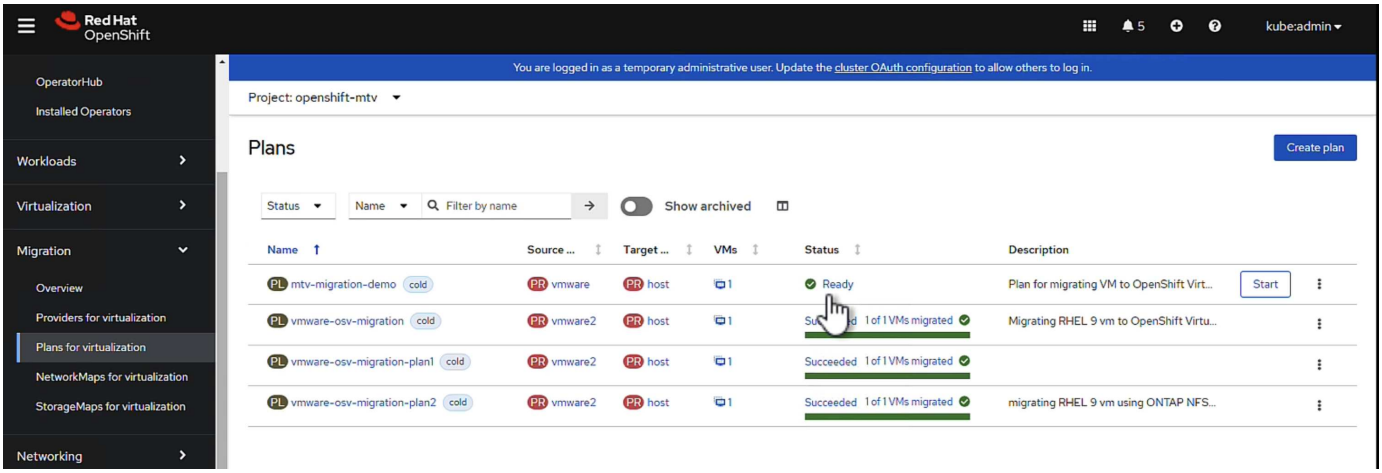
OpenShift仮想化プロバイダがソースプロバイダであるため、ホストクラスタが自動的に追加されます。

移行計画の作成

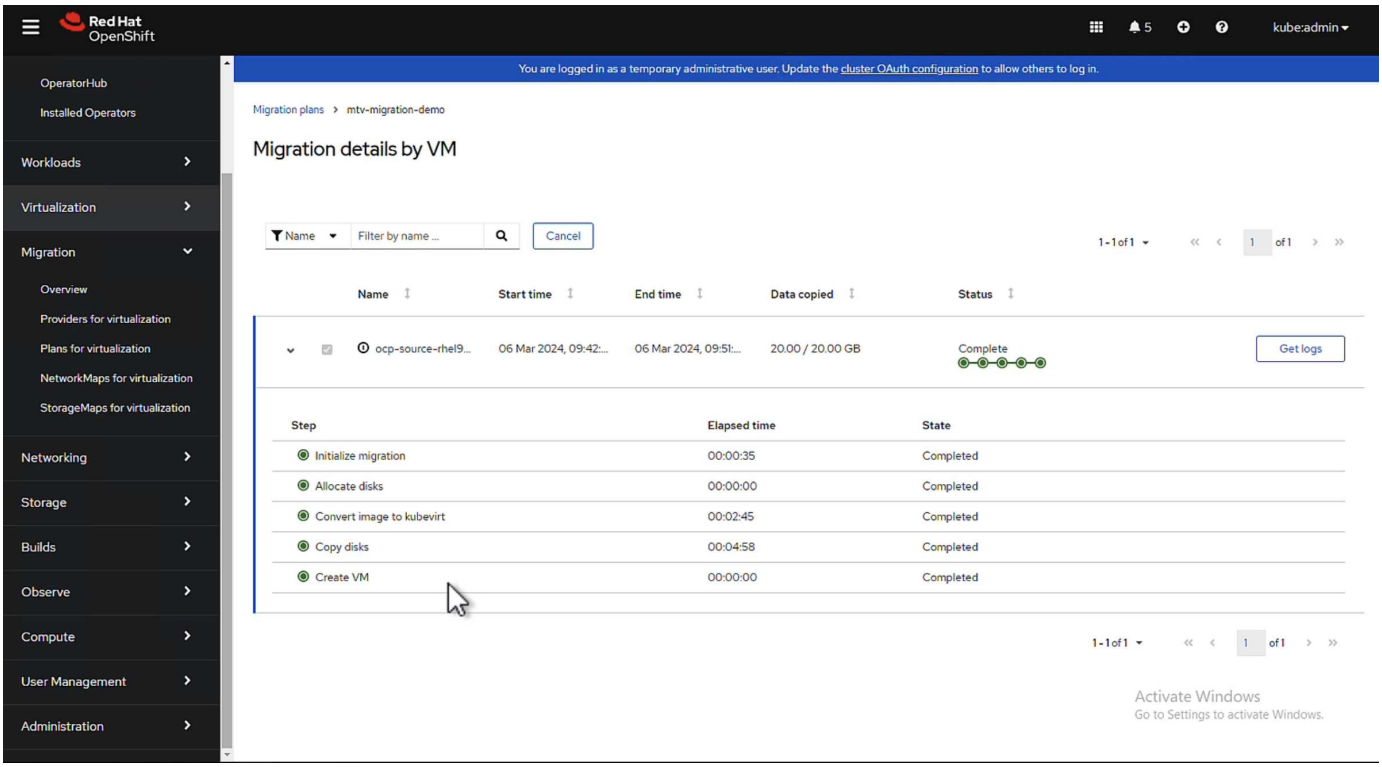
表示される指示に従います。 "こちらをご覧ください" をクリックして移行計画を作成します。

まだ計画を作成していない場合は、計画の作成時に次のものを作成する必要があります。

- ソースネットワークをターゲットネットワークにマッピングするネットワークマッピング。
- ソースデータストアをターゲットストレージクラスにマッピングするストレージマッピング。このためには、ONTAP-SANストレージクラスを選択できます。
移行計画が作成されると、計画のステータスが*準備完了*と表示され、計画を*開始*できるようになります。



[Start]*をクリックすると、VMの移行が完了するまでの一連の手順が実行されます。



すべての手順が完了したら、左側のナビゲーションメニューの*の[仮想マシン]*をクリックすると、移行されたVMが表示されます。
仮想マシンへのアクセス手順が記載されています。 "こちらをご覧ください"。

仮想マシンにログインして、posgresqlデータベースの内容を検証できます。データベース、テーブル、およびテーブル内のエントリは、ソースVMで作成されたものと同じである必要があります。

ネットアップを使用した **Red Hat OpenShift** での **Kubernetes** 向けの高度なクラスタ管理

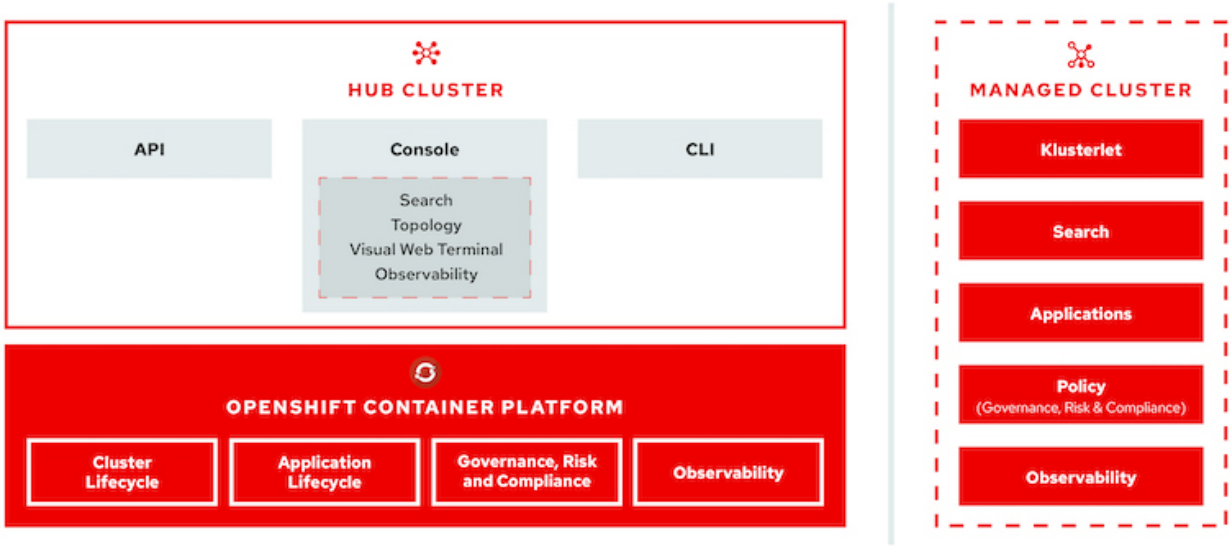
Kubernetes 向けの高度なクラスタ管理：ネットアップを使用した **Red Hat OpenShift**

コンテナ化されたアプリケーションを開発環境から本番環境に移行する際、多くの組織では、そのアプリケーションのテストと導入をサポートするために複数の Red Hat OpenShift クラスタが必要になります。この機能を利用することで、多くの組織は、 OpenShift クラスタ上で複数のアプリケーションやワークロードをホストしています。そのため、組織ごとにクラスタのセットを管理する必要があり、 OpenShift の管理者は、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で複数のクラスタを管理および管理するという新たな課題に直面する必要があります。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管理機能を導入しました。

Kubernetes 向けの Red Hat Advanced Cluster Management では、次のタスクを実行できます。

1. 複数のデータセンターとパブリッククラウドにわたって、複数のクラスタを作成、インポート、管理できます。
2. 1 つのコンソールから複数のクラスタにアプリケーションやワークロードを導入して管理
3. さまざまなクラスタリソースの健常性とステータスを監視および分析できます
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し、実施できます。

Red Hat OpenShift クラスタに Red Hat Advanced Cluster Management for Kubernetes をアドオンとしてインストールし、このクラスタをすべての処理の中央コントローラとして使用します。このクラスタはハブクラスタと呼ばれ、ユーザが 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 コンソールからインポートまたは作成されたその他のすべての OpenShift クラスタは、ハブクラスタによって管理され、管理対象クラスタと呼ばれます。Klusterlet というエージェントを管理対象クラスタにインストールし、ハブクラスタに接続し、クラスタ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、オブザーバビリティ、および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を処理します。



詳細については、のドキュメントを参照してください "こちらをご覧ください"。

導入

Kubernetes 向けの高度なクラスタ管理機能を導入

前提条件

1. ハブクラスタには Red Hat OpenShift クラスタ（バージョン 4.5 以降）が必要です
2. 管理対象クラスタの Red Hat OpenShift クラスタ（バージョン 4.4.4 よりも大きい）
3. Red Hat OpenShift クラスタへのクラスタ管理者アクセス
4. Kubernetes 向けの Advanced Cluster Management 向けの Red Hat サブスクリプション

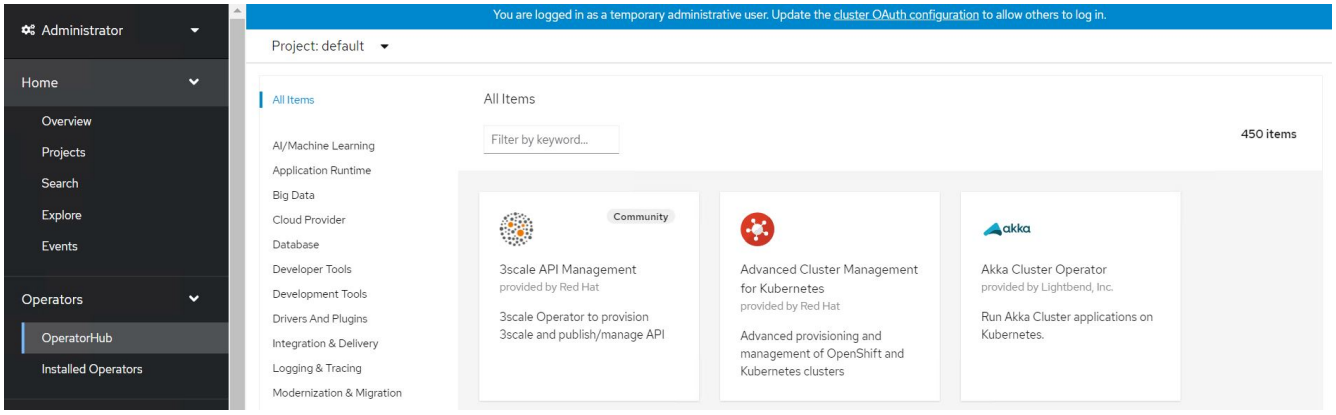
高度なクラスタ管理は OpenShift クラスタのアドオンであるため、ハブクラスタと管理対象クラスタで使用される機能に基づいて、ハードウェアリソースには一定の要件と制限があります。クラスタのサイジングを行う際は、これらの問題について考慮する必要があります。のドキュメントを参照してください "こちらをご覧ください" 詳細：

オプションで、ハブクラスタにインフラストラクチャコンポーネントをホストする専用ノードがあり、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公差とセレクタを追加する必要があります。詳細については、のドキュメントを参照してください "こちらをご覧ください"。

Kubernetes 向けの高度なクラスタ管理機能を導入

OpenShift クラスタに Kubernetes 向けの高度なクラスタ管理をインストールするには、次の手順を実行します。

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



3. Kubernetes の高度なクラスタ管理を選択し、インストールをクリックします。

4. Install Operator 画面で、必要な詳細情報を入力し（デフォルトのパラメータをそのまま使用することを推奨）、 Install をクリックします。

OperatorHub > Operator Installation

Install Operator

Install your Operator by subscribing to one of the update channels to keep the Operator up to date. The strategy determines either manual or automatic updates.

Update channel *

- release-2.0
- release-2.1
- release-2.2

Installation mode *

- All namespaces on the cluster (default)
  This mode is not supported by this Operator
- A specific namespace on the cluster
  Operator will be available in a single Namespace only.

Installed Namespace *

- Operator recommended Namespace: PR open-cluster-management

Namespace creation
Namespace **open-cluster-management** does not exist and will be created.

- Select a Namespace

Approval strategy *

- Automatic
- Manual

Install Cancel

5. オペレータによるインストールが完了するまで待ちます。

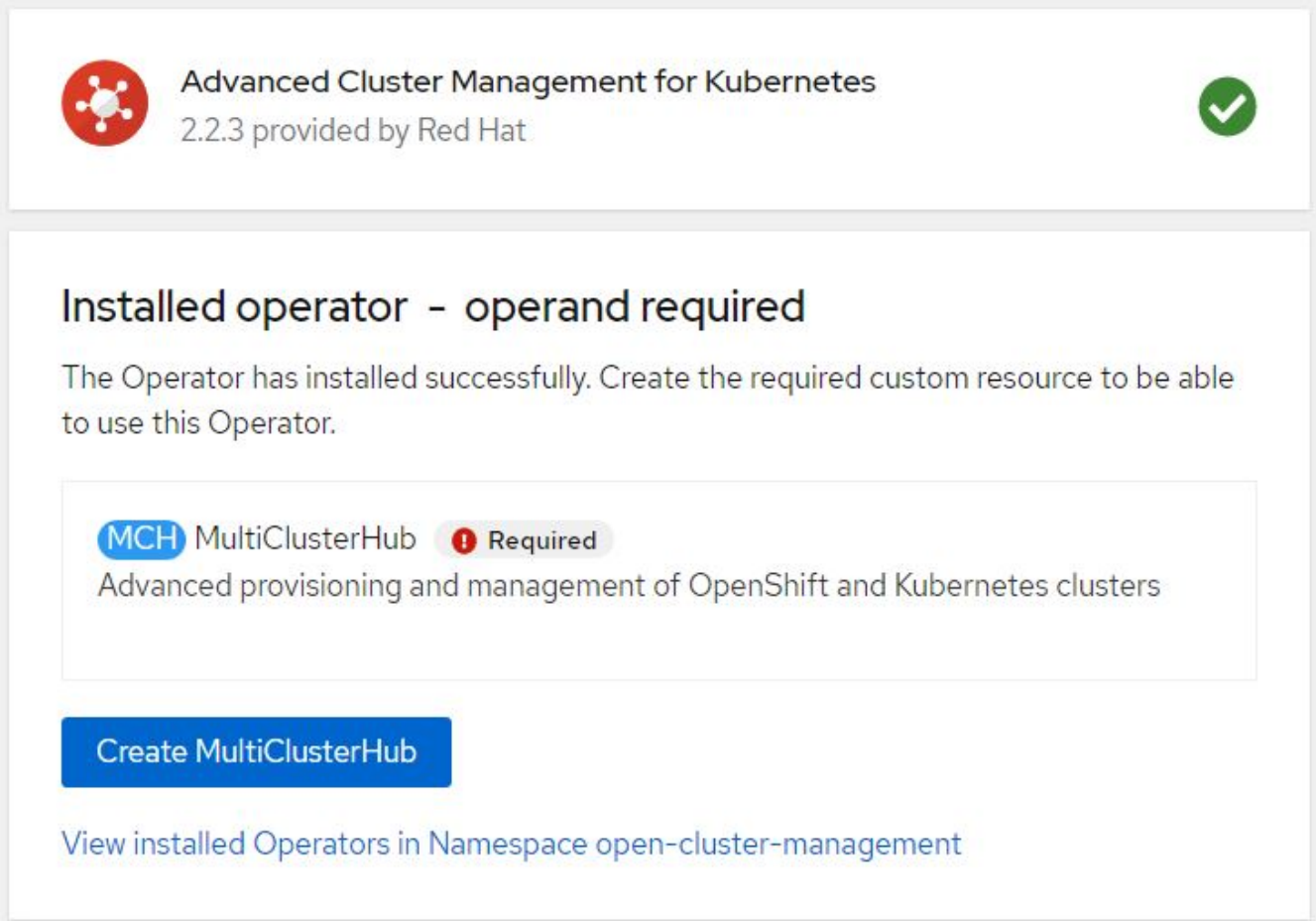
Advanced Cluster Management for Kubernetes
2.2.3 provided by Red Hat

Installing Operator

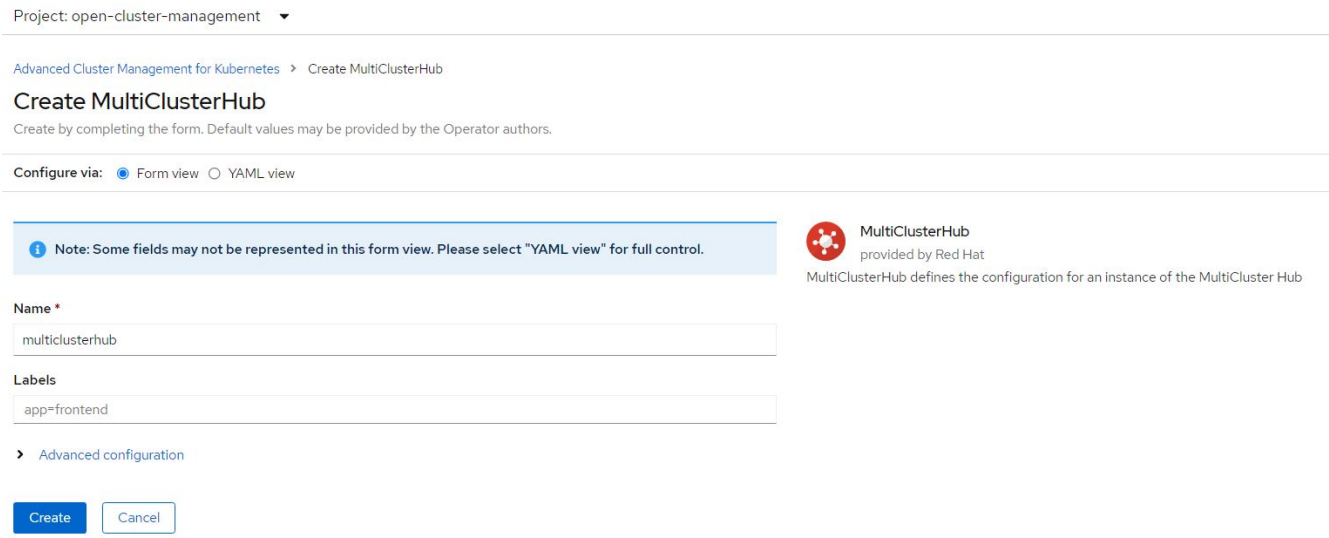
The Operator is being installed. This may take a few minutes.

View installed Operators in Namespace open-cluster-management

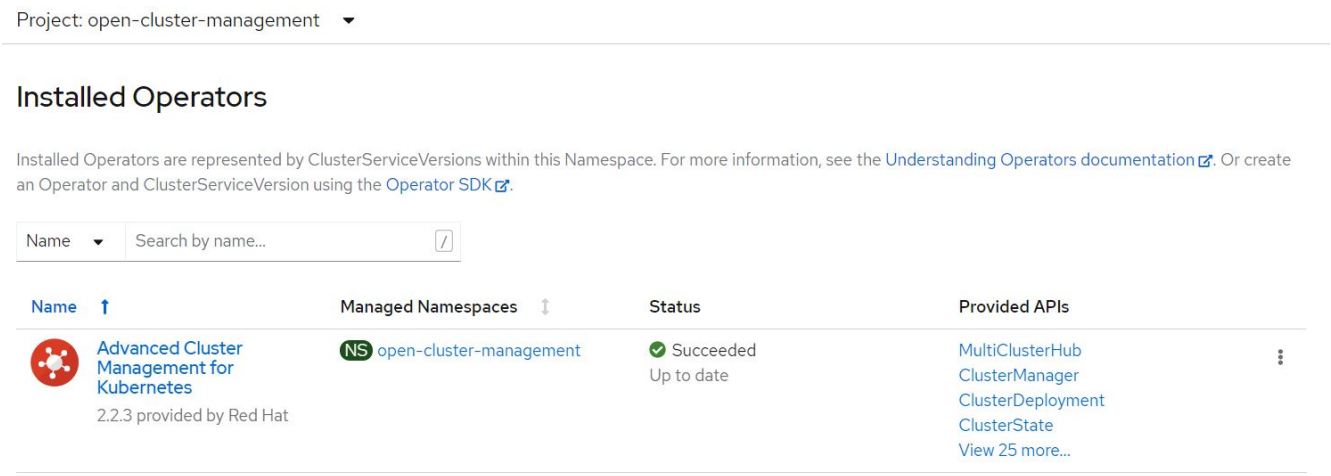
6. オペレータがインストールされたら、 Create MultiClusterHub （ MultiClusterHub の作成）をクリックします。



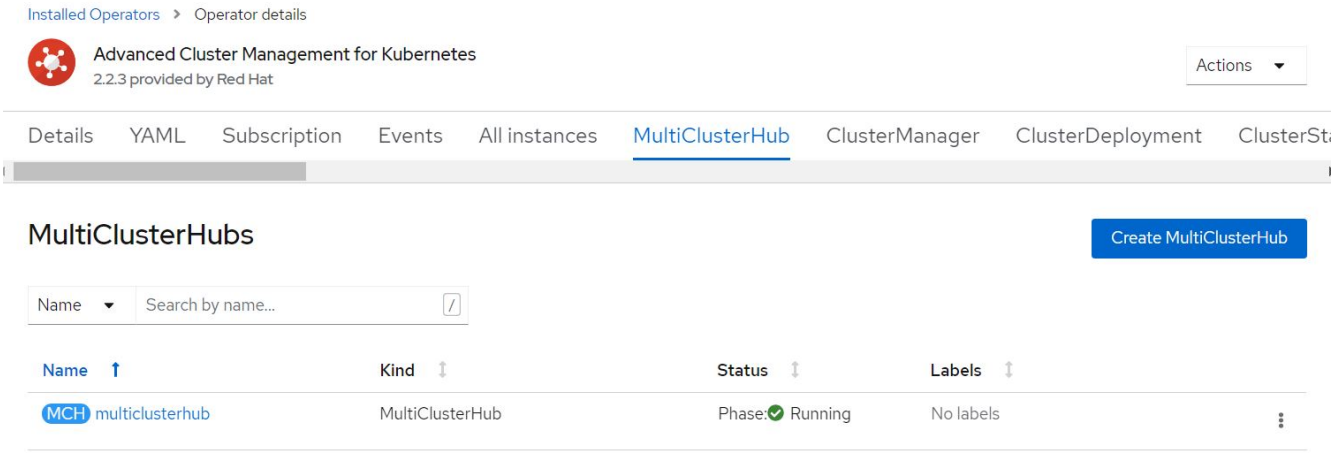
7. Create MultiClusterHub （マルチクラスタハブの作成）画面で、詳細を提供した後に Create （作成）をクリックします。これにより、マルチクラスタハブのインストールが開始されます。



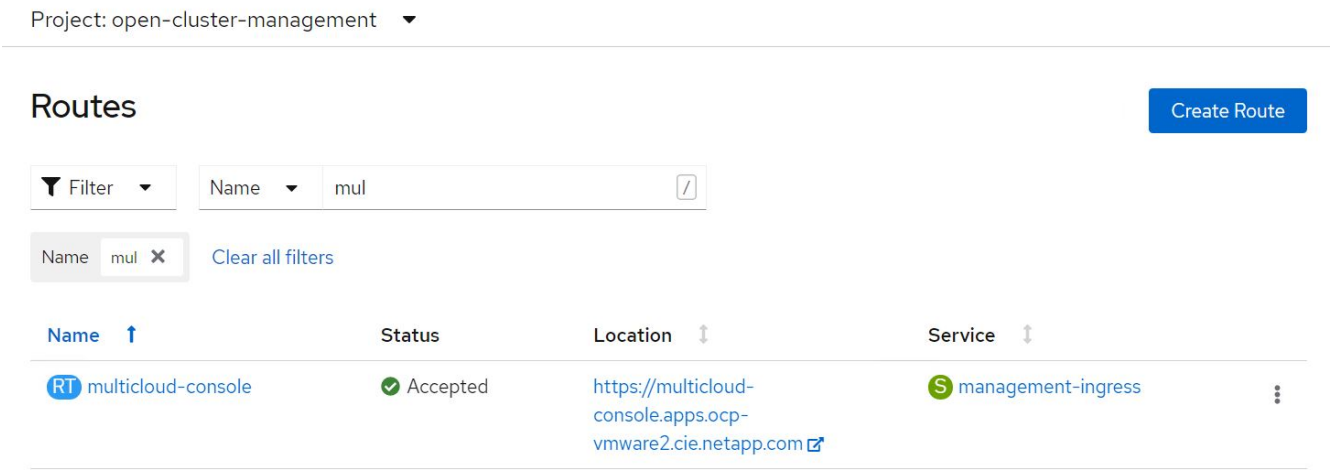
8. すべてのポッドがオープンクラスタ管理ネームスペースの running 状態に移行し、オペレータが Succeeded 状態に移行すると、 Kubernetes の Advanced Cluster Management がインストールされます。



9. ハブのインストールが完了するまでにはしばらく時間がかかり、完了すると、マルチクラスタハブは running 状態に移行します。



10. オープンクラスタ管理ネームスペースにルートが作成されます。ルートの URL に接続して、 Advanced Cluster Management コンソールにアクセスします。



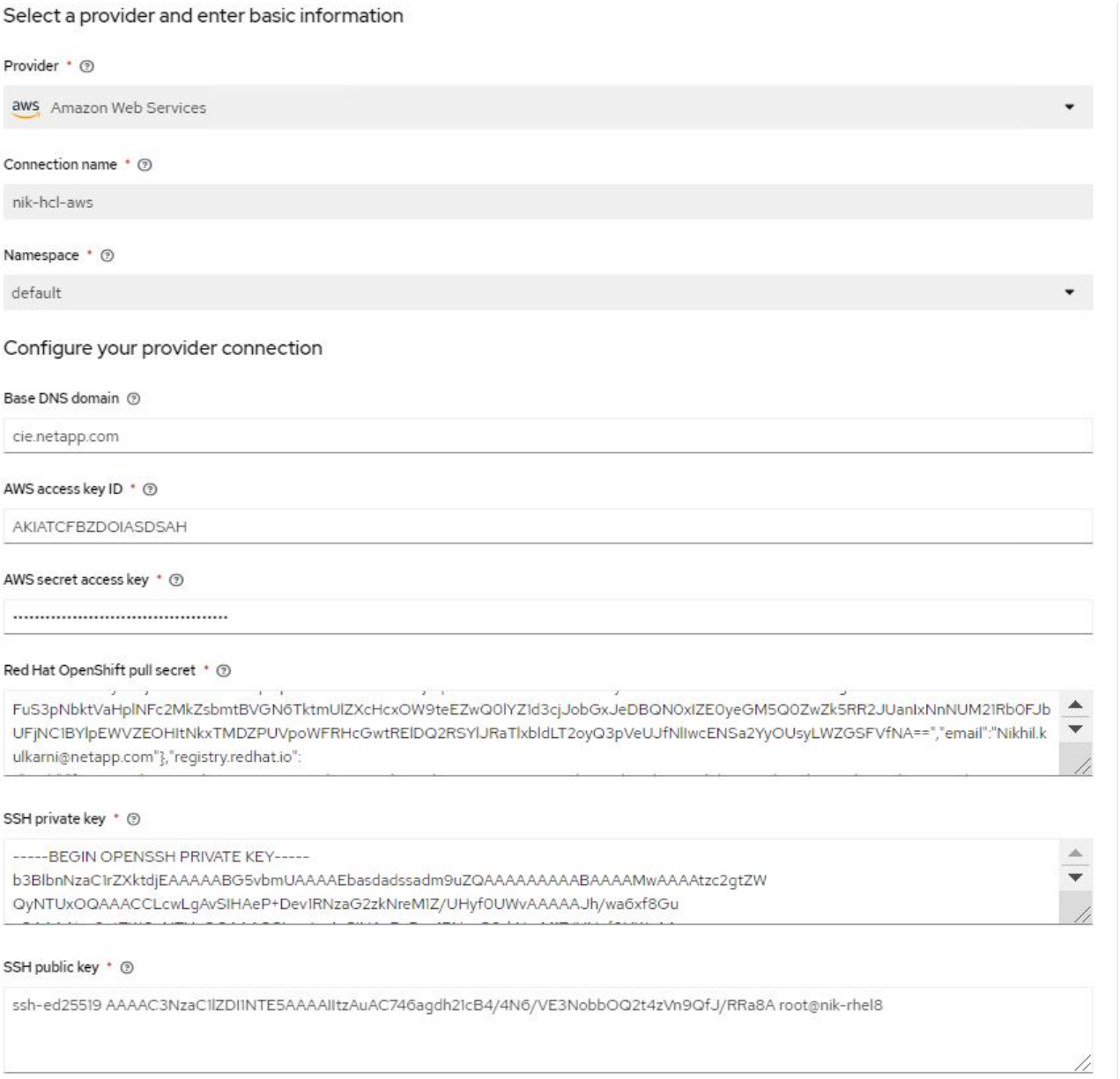
の機能

機能：ネットアップを使用した **Red Hat OpenShift** での **Kubernetes** 向けの高度なクラスタ管理

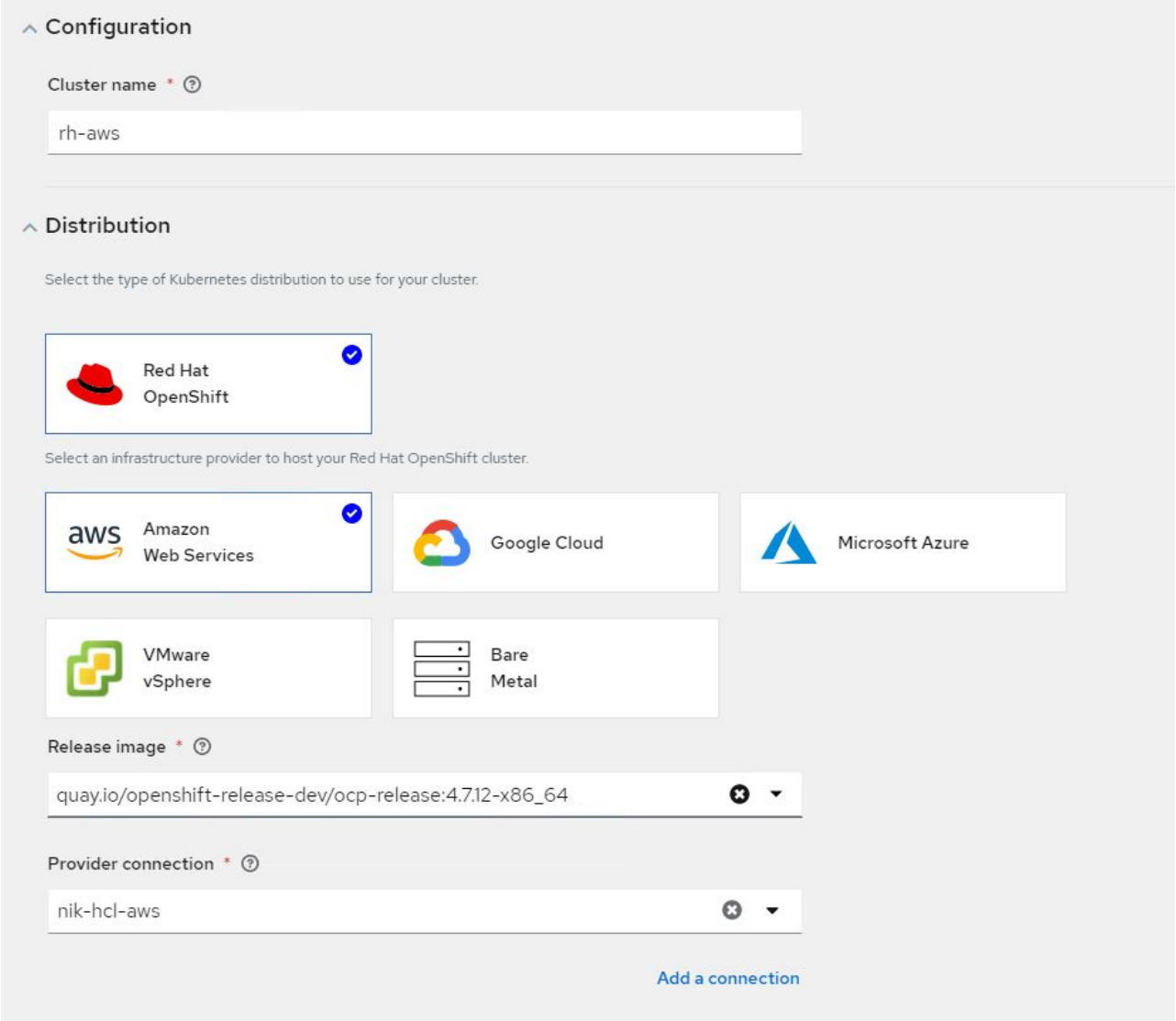
クラスタのライフサイクル管理

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

1. 最初に、 [インフラストラクチャの自動化] 、 [クラスタ] の順に移動
2. 新しい OpenShift クラスタを作成するには、次の手順を実行します。
   a. プロバイダ接続の作成： [プロバイダ接続] 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. 新しいクラスタを作成するには、クラスタに移動し、クラスタの追加 > クラスタの作成をクリックします。クラスタと対応するプロバイダの詳細を指定し、 Create をクリックします。



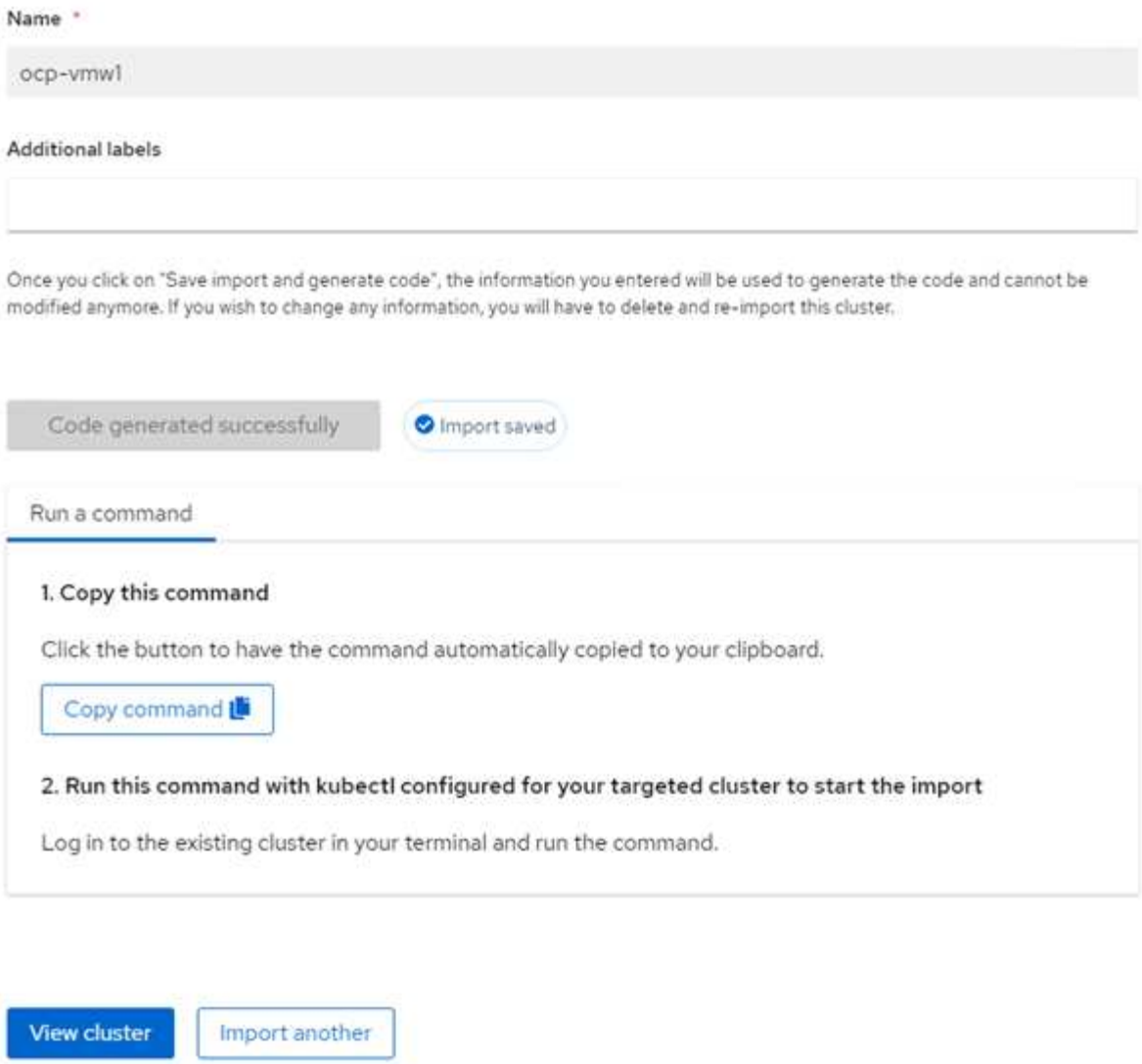
c. 作成されたクラスタは、クラスタのリストに Ready ステータスで表示されます。

3. 既存のクラスタをインポートするには、次の手順を実行します。

   a. クラスタに移動し、クラスタの追加 > 既存クラスタのインポートをクリックします。

   b. クラスタの名前を入力し、 [インポートしてコードを生成して保存] をクリックします。既存のクラスタを追加するコマンドが表示されます。

   c. Copy コマンドをクリックし、ハブクラスタに追加するクラスタ上でコマンドを実行します。これにより、必要なエージェントのクラスタへのインストールが開始され、このプロセスが完了すると、クラスタがクラスタリストに「 Ready 」と表示されます。



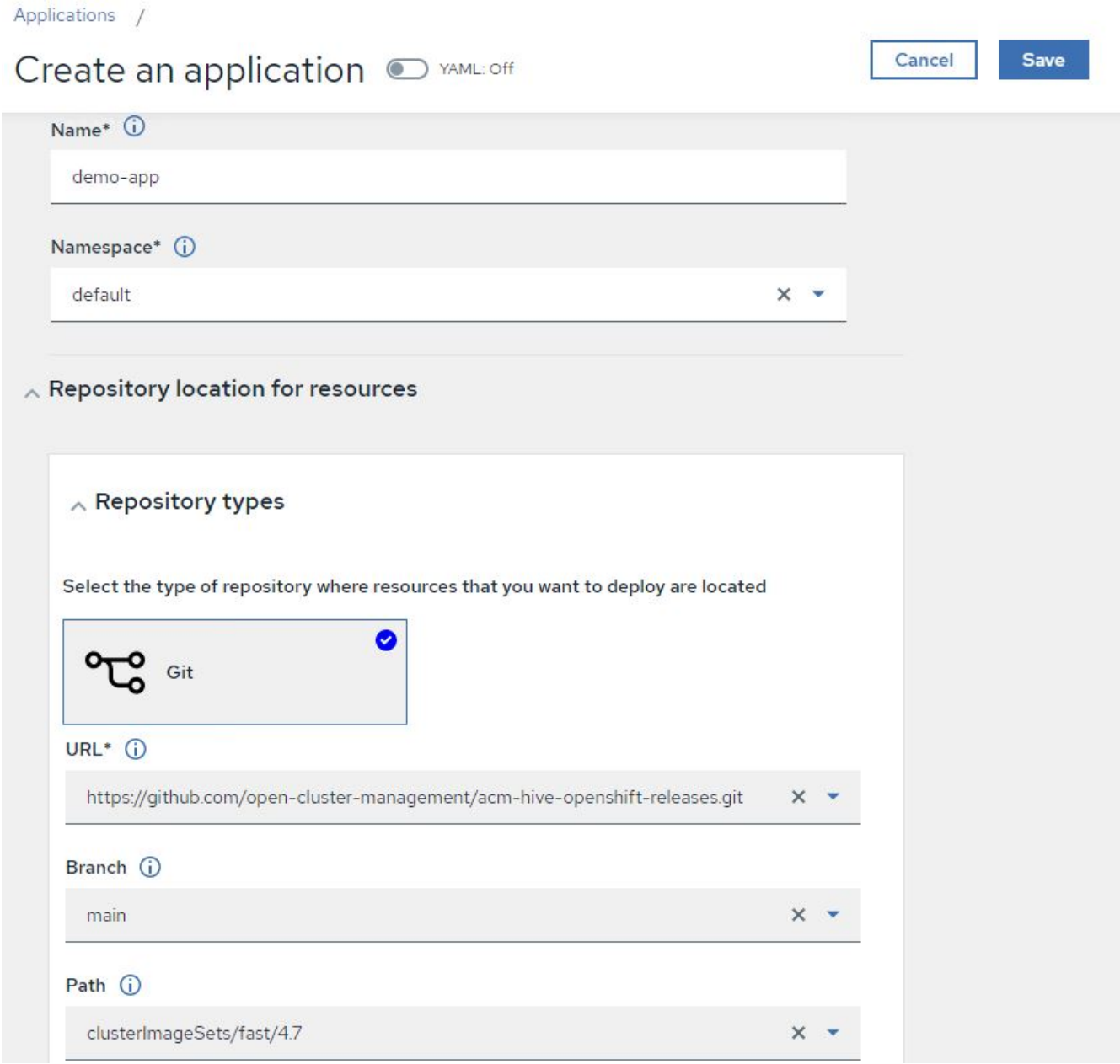
4. 複数のクラスタを作成してインポートしたら、 1 つのコンソールからクラスタを監視および管理できます。

機能：ネットアップを使用した **Red Hat OpenShift** での **Kubernetes** 向けの高度なクラスタ管理

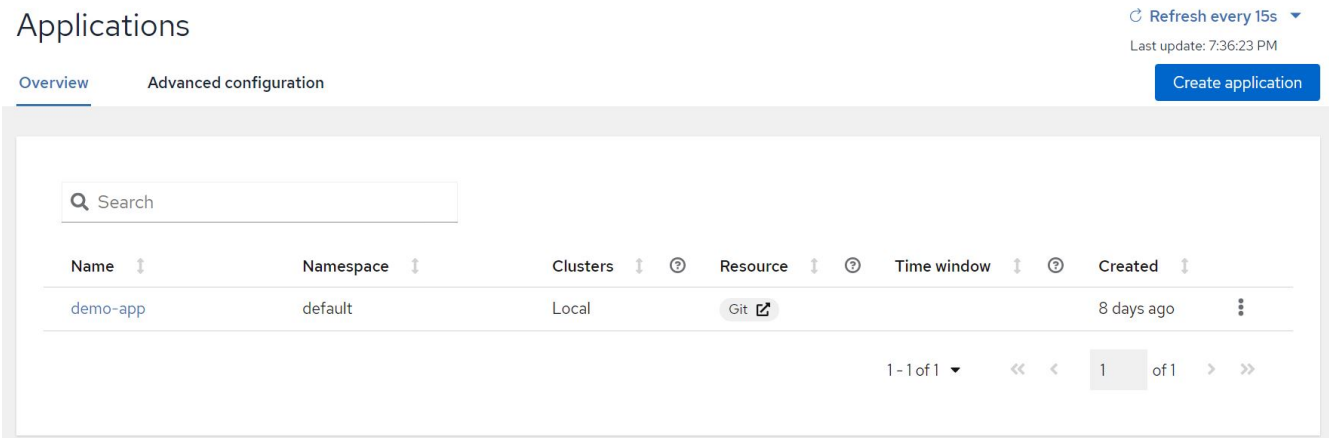
アプリケーションのライフサイクル管理

アプリケーションを作成して一連のクラスタ全体で管理するには、

1. サイドバーから Manage Applications に移動し、 Create Application 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。

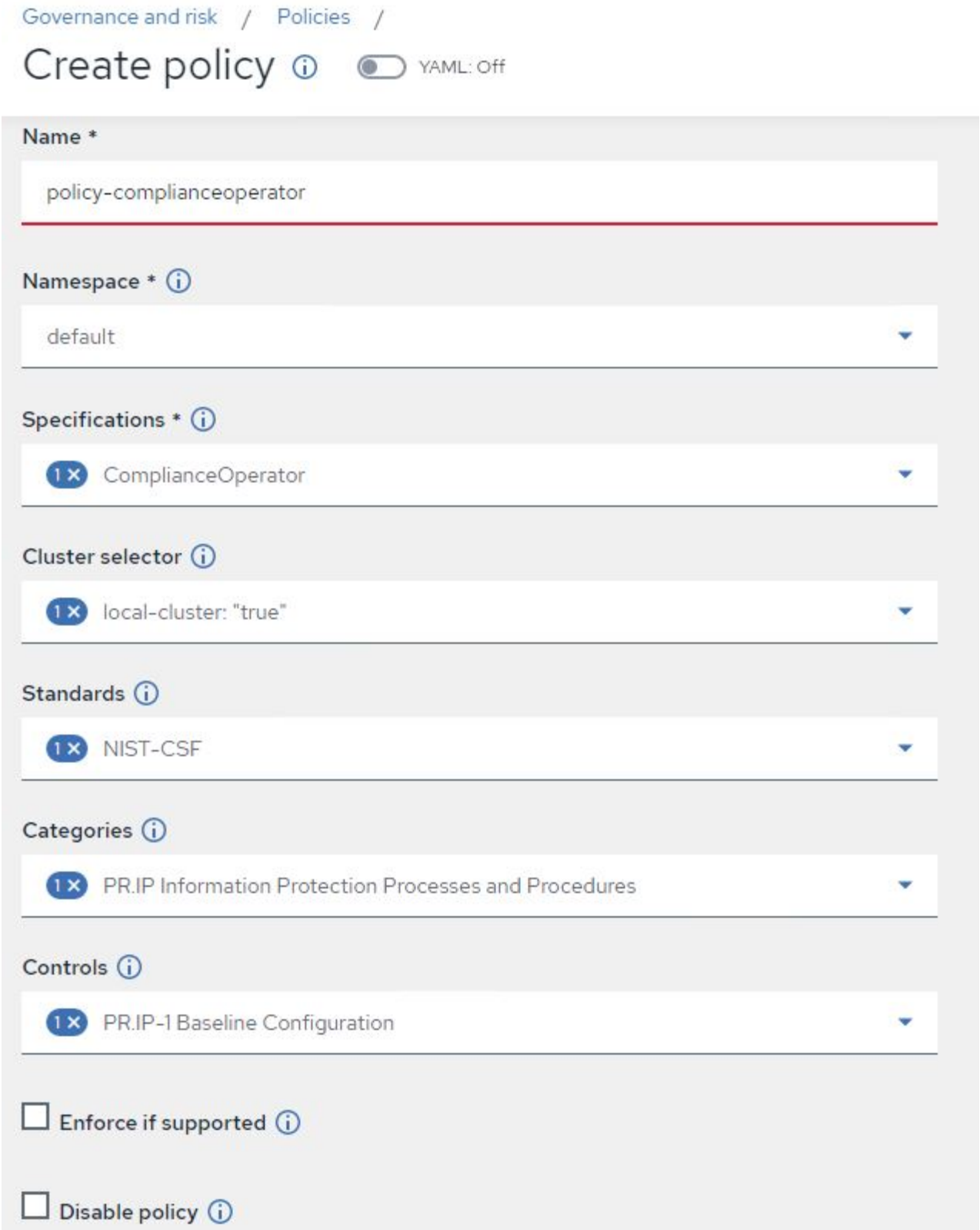


3. これで、アプリケーションをコンソールから監視および管理できるようになります。

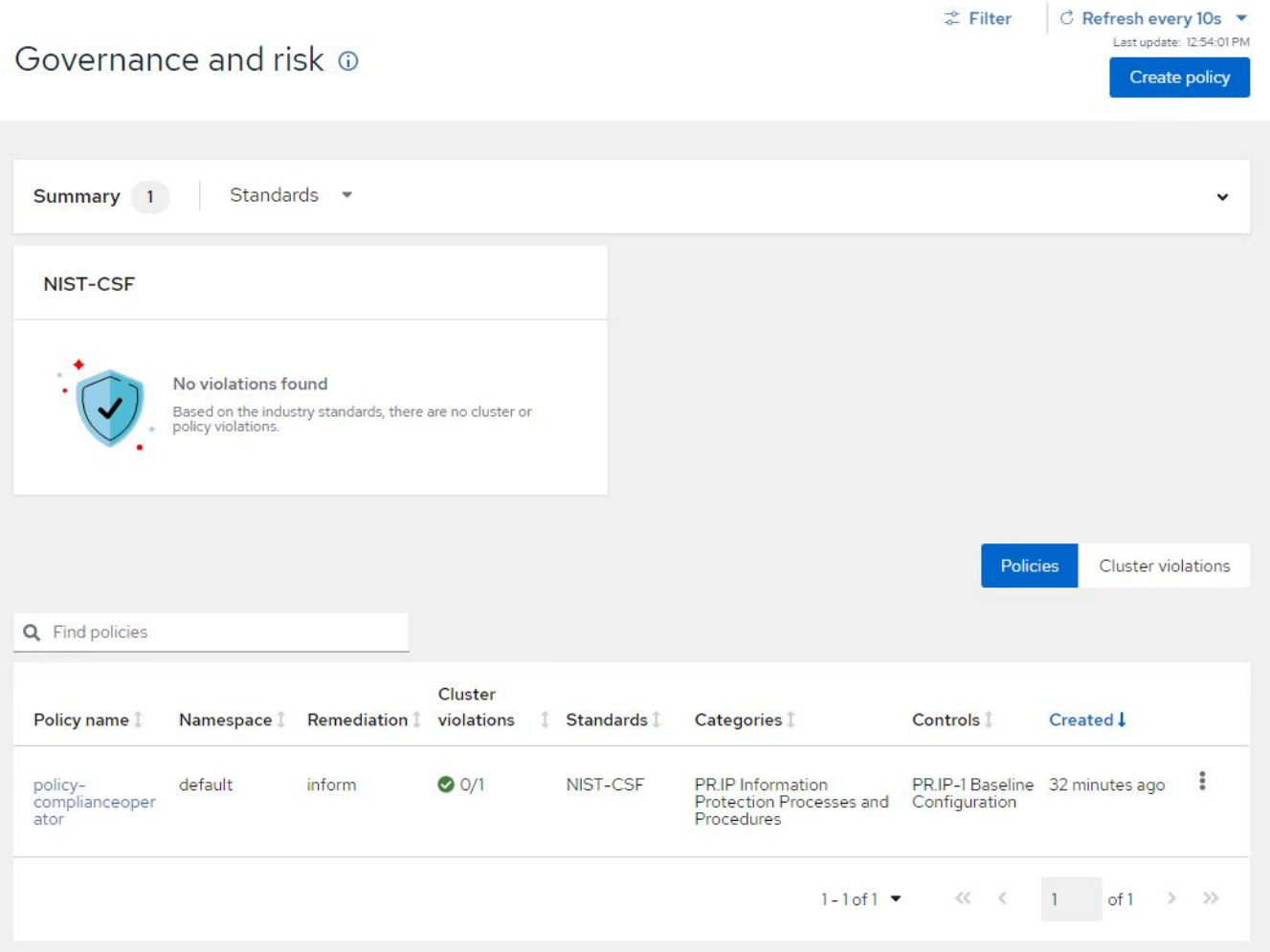
ガバナンスとリスク

この機能を使用すると、異なるクラスタのコンプライアンスポリシーを定義し、それらのクラスタが準拠していることを確認できます。ポリシーを設定して、ルールの逸脱や違反について通知したり修正したりできます。

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、 Create Policy （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クラスタを選択します。このポリシーの違反を自動的に修正するには、 [サポートされている場合に適用] 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. 必要なポリシーをすべて設定したら、 Advanced Cluster Management でポリシーやクラスタの違反を監視して修正できます。

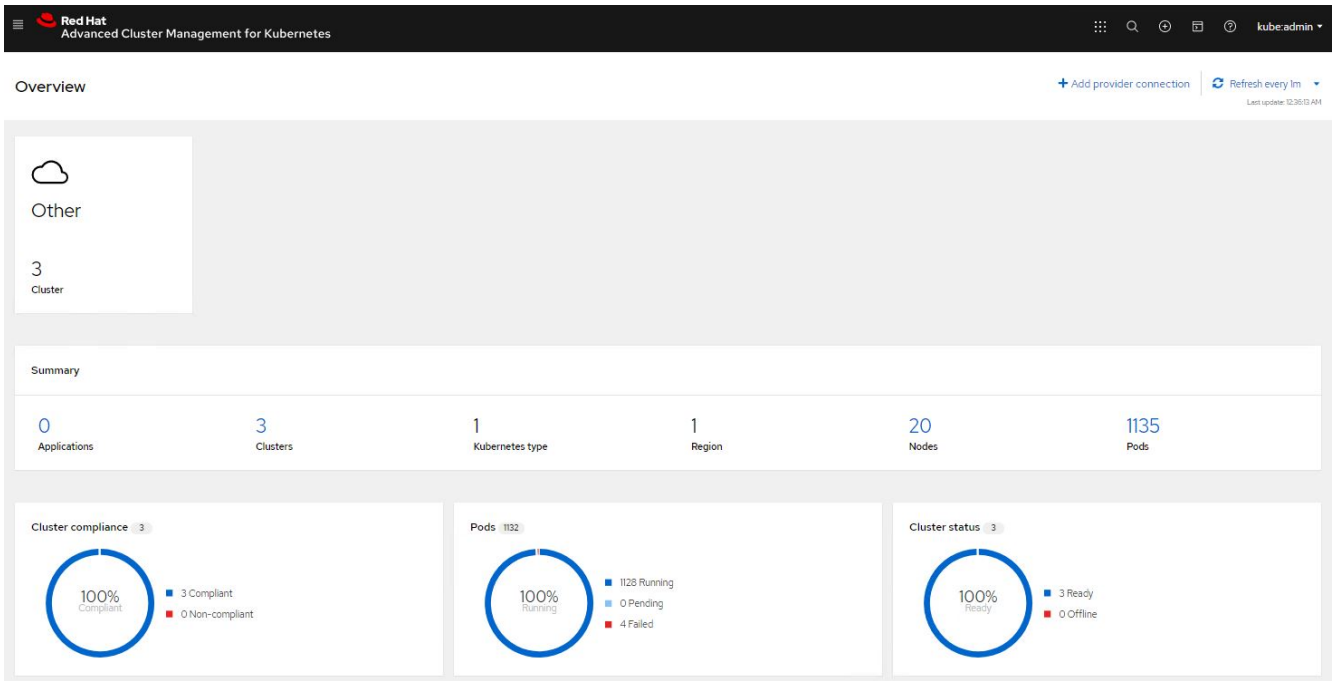


機能：ネットアップを使用した **Red Hat OpenShift** での **Kubernetes** 向けの高度なクラスタ管理

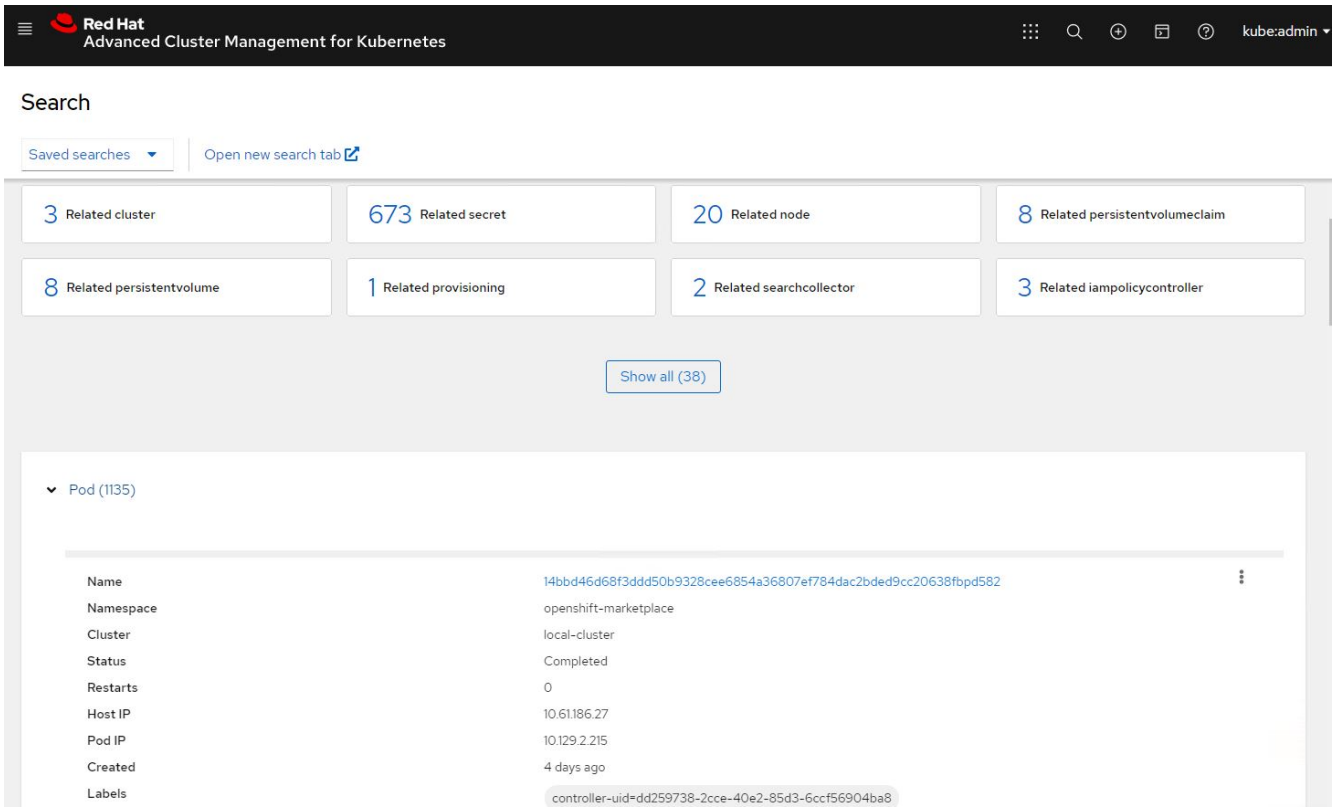
オブザーバビリティ

Kubernetes 向けの高度なクラスタ管理機能を使用すると、ノード、ポッド、およびすべてのクラスタのアプリケーションとワークロードを監視できます。

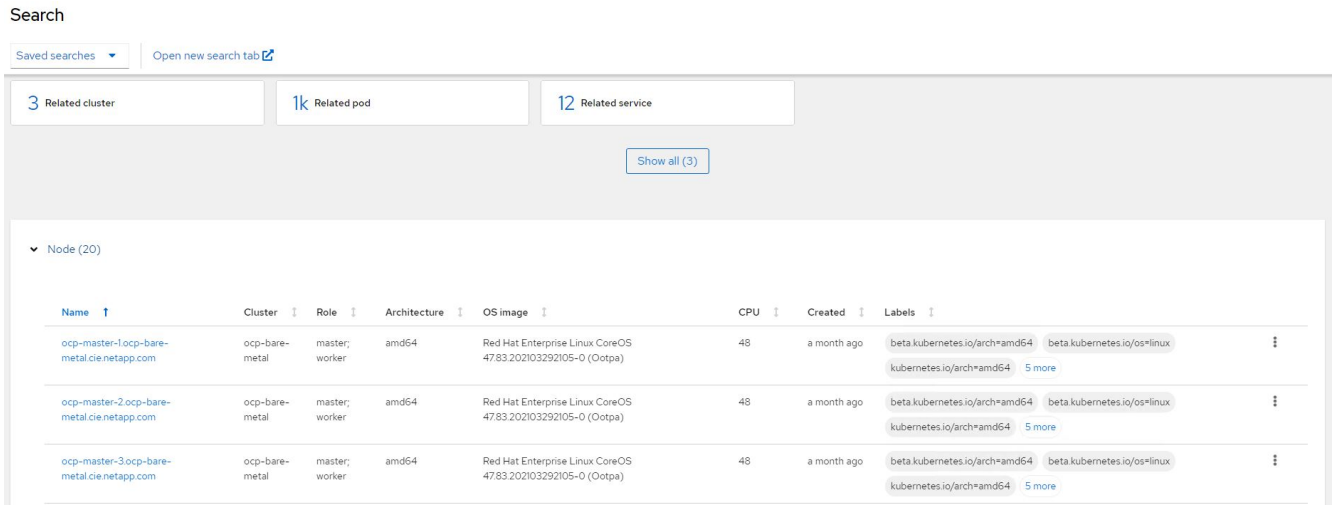
1. [環境の監視]>[概要] に移動します。



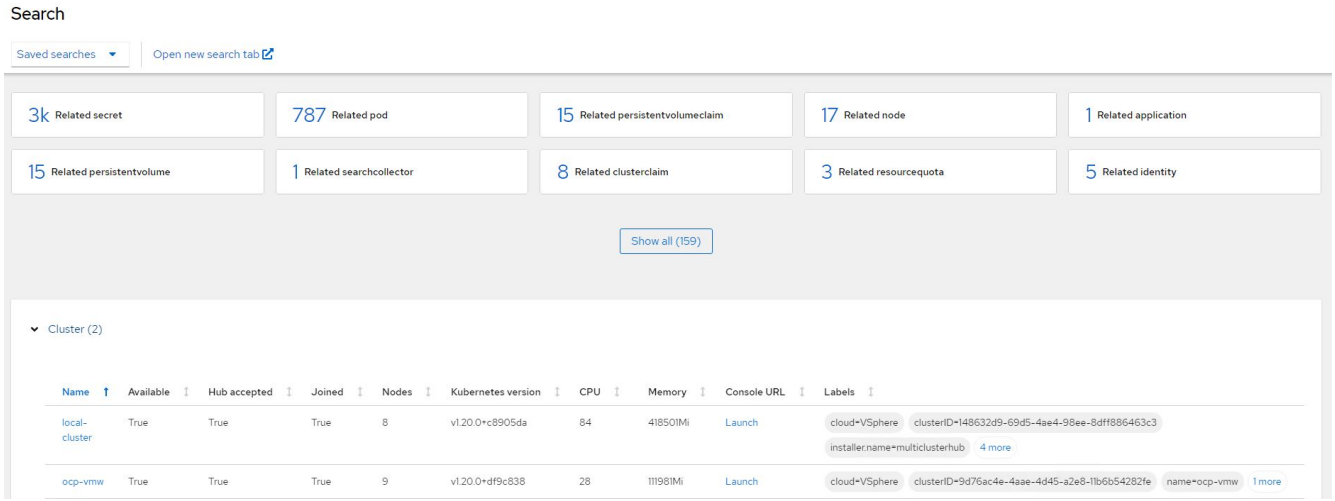
2. すべてのクラスタのすべてのポッドとワークロードが監視され、さまざまなフィルタに基づいてソートされます。ポッドをクリックすると、対応するデータが表示されます。



3. クラスタ内のすべてのノードが、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、対応する詳細が表示されます。



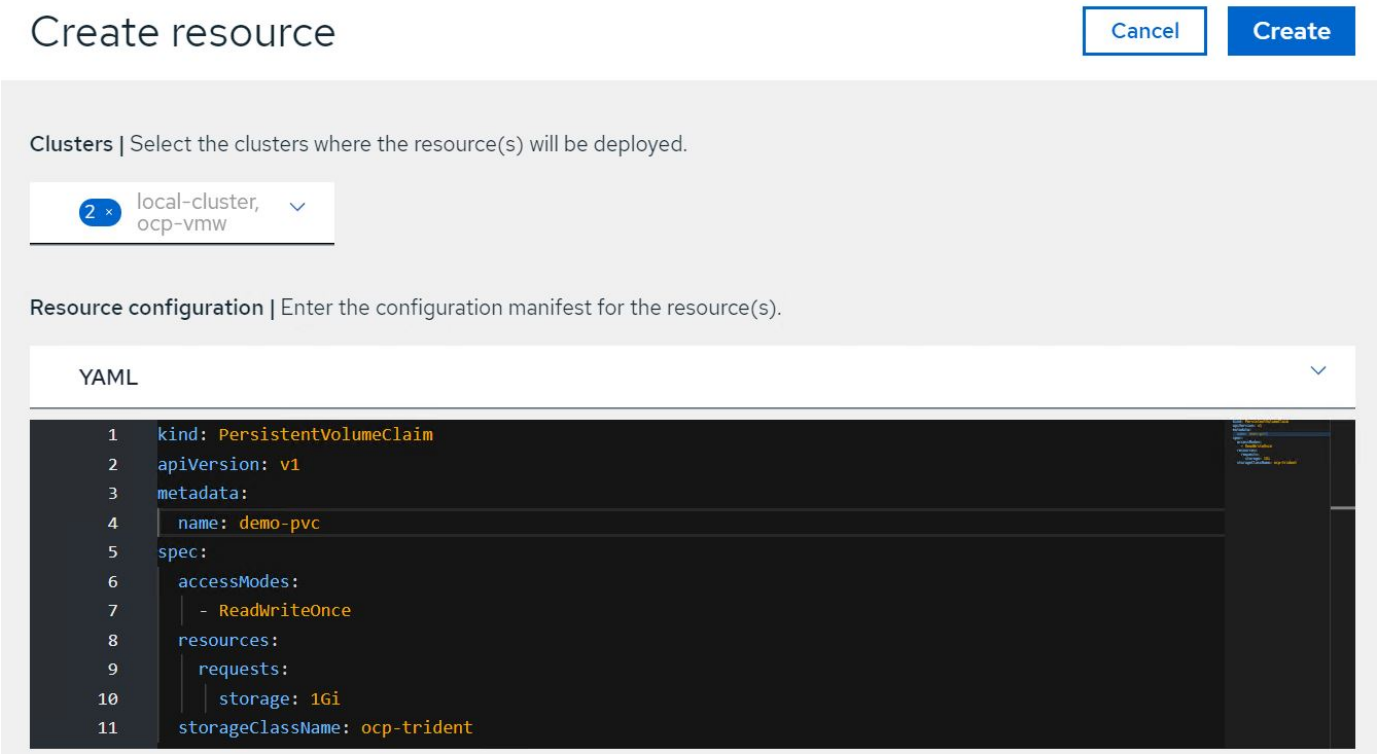
4. クラスタはすべて、クラスタのリソースとパラメータに基づいて監視および整理されます。クラスタをクリックしてクラスタの詳細を表示します。



機能：ネットアップを使用した **Red Hat OpenShift** での **Kubernetes** 向けの高度なクラスタ管理

複数のクラスタにリソースを作成する

Kubernetes 向けの高度なクラスタ管理機能を使用すると、ユーザはコンソールから 1 つ以上の管理対象クラスタ上にリソースを同時に作成できます。たとえば、異なる NetApp ONTAP クラスタでサポートされている異なるサイトに OpenShift クラスタがあり、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を選択し、リソース YAML を貼り付けて、Create をクリックします。



ビデオとデモ：ネットアップを使用した Red Hat OpenShift

次のビデオでは、このドキュメントで説明している機能の一部を紹介します。

Red Hat MTVを使用したNetApp ONTAPストレージによるOpenShift仮想化へのVMの移行

Astra ControlとNetApp FlexCloneテクノロジでソフトウェア開発を高速化- Red Hat OpenShift with NetApp

NetApp Astra Control を活用して、事後分析とアプリケーションのリストアを実行

Astra Control Centerを使用したCI / CDパイプラインのデータ保護

Astra Control Center - Red Hat OpenShiftとNetAppを使用したワークロードの移行

ワークロードの移行 - ネットアップを使用した Red Hat OpenShift

OpenShift Virtualizationのインストール-ネットアップでRed Hat OpenShiftを実装します

OpenShift仮想化を使用した仮想マシンの導入-ネットアップでRed Hat OpenShiftを実装します

Red Hat 仮想化での NetApp HCI for Red Hat OpenShift

追加情報：ネットアップを使用した Red Hat OpenShift

このドキュメントに記載されている情報の詳細については、次の Web サイトを参照してください。

- NetApp のドキュメント

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

- Astra Trident のドキュメント

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

- NetApp Astra Control Center のドキュメント

  "https://docs.netapp.com/us-en/astra-control-center/"

- Red Hat OpenShift のドキュメント

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

- Red Hat OpenStack Platform のドキュメント

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

- Red Hat Virtualization のドキュメント

  "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/"