vSphere with Tanzu の構成と管理

Update 3
VMware vSphere 7.0
vCenter Server 7.0
VMware ESXi 7.0

vmware®

最新の技術ドキュメントは、 VMware の Web サイト （https://docs.vmware.com/jp/）

VMware, Inc.
3401 Hillview Ave.
Palo Alto, CA 94304
www.vmware.com

ヴイエムウェア株式会社
〒108-0023 東京都港区芝浦 3-1-1
田町ステーションタワー N 18 階
www.vmware.com/jp

目次

vSphere with Tanzu の設定と管理

『vSphere with Tanzu の構成と管理』には、vSphere Client を使用して vSphere with Tanzu を構成および管理する方法が記載されています。また、kubectl を使用して vSphere with Tanzu で実行されている名前空間に接続し、指定した名前空間で Kubernetes ワークロードを実行する方法についても説明します。

『vSphere with Tanzu の構成と管理』では、プラットフォーム アーキテクチャの概要や、vSphere with Tanzu に固有の要件を満たすストレージ、コンピューティング、およびネットワークの設定に関する考慮事項とベスト プラクティスを示します。また、既存の vSphere クラスタで vSphere with Tanzu を有効にする手順、名前空間を作成および管理する手順、VMware Tanzu™ Kubernetes Grid™ サービス を使用して作成された Tanzu Kubernetes クラスタを監視する手順を示します。

この情報には、kubectl を使用した vSphere with Tanzu Kubernetes 制御プレーンとのセッションの確立、サンプル アプリケーションの実行、VMware Tanzu™ Kubernetes Grid™ サービス を使用した Tanzu Kubernetes クラスタの作成に関するガイドラインも示されます。

VMware では、多様性の受け入れを尊重しています。お客様、パートナー企業、社内コミュニティとともにこの原則を推進することを目的として、多様性に配慮した言葉遣いでコンテンツを作成します。

対象読者

『vSphere with Tanzu の構成と管理』は、vSphere で vSphere with Tanzu を有効にし、名前空間を設定して DevOps チームに提供するほか、vSphere で Kubernetes ワークロードを管理および監視する vSphere 管理者を対象としています。vSphere with Tanzu を使用する vSphere 管理者には、コンテナおよび Kubernetes に関する基本的な知識が必要です。

この情報は、vSphere with Tanzu 制御プレーンとのセッションを確立し、Kubernetes ワークロードを実行するほか、VMware Tanzu™ Kubernetes Grid™ サービス を使用して Kubernetes クラスタをデプロイする DevOps エンジニアも対象としています。また、プラットフォームにアプリケーションをデプロイする開発者は、ガイダンスとして例を参照することができます。

更新情報

1

『vSphere with Tanzu の構成と管理』は、必要に応じて、新しい情報や修正によって定期的に更新されます。

『vSphere with Tanzu の構成と管理』の更新履歴については、次の表をご確認ください。

リビジョン	説明
2022 年 12 月 08 日	ネットワーク セキュリティ ポリシーに関する情報を追加。NSX スーパーバイザー クラスタ 名前空間へのセキュリティ ポリシーの追加を参照してください。
2022 年 10 月 17 日	スペル ミスを修正。
2022 年 10 月 12 日	■ TKG 1.6 パッケージをインストールするためのリンクを追加。Tanzu Kubernetes クラスタへの TKG パッケージのデプロイを参照してください。 ■ nodeDrainTimeout の正しいデータ型 (string) を使用して TKGS v1alpha2 API 仕様を更新。Tanzu Kubernetes クラスタをプロビジョニングするための TKGS v1alpha2 API を参照してください。 ■ Guestbook アプリケーションのソースの YAML ファイルを更新。Guestbook のサンプル YAML ファイルを参照してください。
2022 年 9 月 15 日	軽微なバグ修正。
2022 年 7 月 29 日	vSphere with Tanzu セキュリティでシークレットの暗号化に関する記述を明確にしました。
2022 年 7 月 07 日	vCenter Server と NSX-T の互換性を確認するための VMware の相互運用性マトリックスへのリンクを追加しました。vSphere with Tanzu の更新についてを参照してください。
2022 年 6 月 28 日	vSphere with Tanzu のシャットダウンと起動についてのトピックを更新し、最新手順へのリンクを追加しました。vSphere with Tanzu ワークロード ドメインのシャットダウンと起動を参照してください。
2022 年 6 月 24 日	vSphere 名前空間の作成手順を更新しました。vSphere 名前空間 の作成と設定を参照してください。
2022 年 6 月 03 日	■ 外部 DNS 拡張機能のドキュメントを更新しました。外部 DNS サービス検出のための TKG 拡張機能のデプロイと管理を参照してください。 ■ タイプミスを修正しました。
2022 年 5 月 24 日	vSphere with Tanzu での ReadWriteMany パーシステント ボリュームの作成を更新し、Tanzu Kubernetes クラスタによる ReadWriteMany のサポートを表明しました。
2022 年 5 月 20 日	redis-leader デプロイに redis:v6.0.5 を指定して Guestbook のサンプル YAML を更新しました。Guestbook のサンプル YAML ファイルを参照してください。
2022 年 5 月 13 日	■ Velero Plugin for vSphere のインストールに関するドキュメントを更新します。スーパーバイザー クラスタ での Velero Plugin for vSphere のインストールと構成を参照してください。 ■ マイナー タイプミスを修正しました。

リビジョン	説明
2022 年 5 月 06 日	■ TKGS クラスタ ワーカー ノード ボリュームをスケーリングすると、既存のボリューム データが削除されることについての注記を追加しました。ノード ボリュームのスケーリングを参照してください。 ■ SVC-TMC 統合のトピックを更新しました。スーパーバイザー クラスタ の Tanzu Kubernetes Grid サービス と Tanzu Mission Control の統合を参照してください。 ■ TKGS クラスタのプロビジョニング ワークフローを更新しました。TKGS v1alpha2 API を使用して Tanzu Kubernetes クラスタをプロビジョニングするためのワークフローを参照してください。 ■ TKGS ネットワークの概念を更新しました。Tanzu Kubernetes Grid サービス クラスタ ネットワーク を参照してください。 ■ タイプミスを修正し、軽微な編集を行いました。
2022 年 4 月 21 日	マイナー改訂。
2022 年 4 月 18 日	■ リソースの競合やクラスタのダウンタイムの可能性を回避するために、保証型の仮想マシン クラスを、本番クラスタに使用することを推奨する記述を強調しました。Tanzu Kubernetes クラスタの仮想マシンのクラスと vSphere with Tanzu の名前空間で使用可能な仮想マシン リソースの表示を参照してください。 ■ ベスト エフォート型仮想マシン クラスで Tanzu Kubernetes クラスタがプロビジョニングされる vSphere 名前空間 における制限の設定に関する注意事項を追加しました。vSphere 名前空間 での Kubernetes オブジェクトの制限の構成を参照してください。
2022 年 4 月 15 日	■ TKGS プロビジョニング ドキュメントに、0 個のワーカー ノード/ノード プールでプロビジョニングしたクラスタには、ロード バランサ サービスが割り当てられないことを記載した注意事項を追加しました。 ■ ローカル コンテンツ ライブラリのドキュメントを更新しました。Tanzu Kubernetes リリース 用のローカル コンテンツ ライブラリの作成、セキュリティ保護、同期を参照してください。 ■ Sysadmin ログイン認証キーに関する詳細情報を追加しました。コントローラのデプロイを参照してください。 ■ サブジェクト代替名（SAN）に関する詳細情報を追加しました。コントローラへの証明書の割り当てを参照してください。 ■ vsan-direct ストレージ クラスと vsan-sna ストレージ クラスを使用できるのは、スーパーバイザー クラスタ 上のアプリケーションのみであって、Tanzu Kubernetes クラスタ内では使用できないことを明記しました。vSphere with Tanzu でのステートフル サービスの有効化を参照してください。 ■ スーパーバイザー クラスタ 制御プレーン ログのリモート ストリーミングの構成方法に関する情報を追加しました。リモート rsyslog に対する スーパーバイザー クラスタ 制御プレーンのログ ストリーミング を参照してください。
2022 年 3 月 28 日	vSphere with Tanzu 環境内の HTTP プロキシの構成に関する情報を追加しました。vSphere with Tanzu での HTTP プロキシ設定の構成を参照してください。
2022 年 3 月 18 日	マイナー タイプミスを修正しました。
2022 年 3 月 04 日	ローカル コンテンツ ライブラリのトラブルシューティング トピックを追加しました。ローカル コンテンツ ライブラリのエラーのトラブルシューティングを参照してください。
2022 年 2 月 28 日	■ TKG 1.5 パッケージをインストールするためのリンクを追加しました。Tanzu Kubernetes クラスタへの TKG パッケージのデプロイを参照してください。 ■ 図の制御プレーン仮想マシン名を修正しました。NSX-T Data Center を使用した スーパーバイザー クラスタ のトポロジを参照してください。 ■ マイナー タイプミスを修正しました。

リビジョン	説明
2022 年 2 月 18 日	■ コントローラの構成を更新し、DHCP が有効になっていないときの構成オプションの手順を追加しました。 ■ コントローラへの証明書の割り当てを更新し、事前に作成した有効な証明書をアップロードする手順を追加しました。 ■ システム要件を更新し、IPv6 がサポートされていないことを明記しました。4 章 vSphere with Tanzu のネットワークを参照してください。 ■ クラスタ仕様で `pods.cidrBlocks` 設定をカスタマイズする場合の TKGS クラスタのネットワーク要件を更新しました。TKGS v1alpha2 API を使用するための要件と Tanzu Kubernetes クラスタをプロビジョニングするための TKGS v1alpha2 API を参照してください。 ■ TKGS クラスタのネットワーク コマンドの例のリストを更新しました。Tanzu Kubernetes クラスタのネットワーク コマンドの使用を参照してください。
2022 年 2 月 11 日	■ vSphere ネットワークを使用したワークロード管理の有効化を更新し、有効化後の次の手順として vSphere 名前空間 の作成と構成を行う手順までのリンクを追加しました。 ■ vSphere 名前空間 の作成と設定を更新し、次の手順として スーパーバイザー クラスタ にログインするためのリンクを追加しました。
2022 年 2 月 8 日	■ HA プロキシ ロード バランサをインストールするためのシステム要件を更新し、ワークロード ネットワークは管理ネットワークとは別のサブネット上に配置する必要があることを記した注記を加えました。vSphere ネットワークおよび HAProxy ロード バランサを使用して vSphere with Tanzu をセットアップするためのシステム要件を参照してください。
2022 年 2 月 4 日	■ vSphere with Tanzu での ReadWriteMany パーシステント ボリュームの作成を更新し、ネットワークの古いガイドラインを削除しました。 ■ TKGS v1alpha2 API を使用して Tanzu Kubernetes クラスタをプロビジョニングするためのワークフローを更新し、コンテキストと例を追加しました。 ■ スーパーバイザー クラスタ ネットワークトピックを更新し、タイプミスを修正しました。 ■ パッチ方式を使用した Tanzu Kubernetes クラスタの更新トピックを更新し、v1alpah2 TKGS API に準拠する目的では、`kubectl patch` メソッドを使用してクラスタ仕様を更新できないことを記した警告を追加しました。 ■ クラスタ仕様を TKGS v1alpha2 API に変換した後の Tanzu Kubernetes リリースの更新を更新し、この操作を実行するために `kubectl edit` メソッドを使用する必要があることを明記しました。
2022 年 1 月 28 日	■ Kubernetes CLI のダウンロードに関するトピックを更新し、ユーザー ナビゲーションに役立つスクリーンショットを追加しました。vSphere 向け Kubernetes CLI Tools のダウンロードとインストールを参照してください。 ■ Tanzu Kubernetes クラスタのスケーリングに関するトピックを更新し、クラスタ ノード ボリュームの追加または変更に関する情報を追加しました。TKGS v1alpha2 API を使用した Tanzu Kubernetes クラスタのスケーリングを参照してください。 ■ 「TKG 拡張機能 1.3.1 のダウンロード」トピックを更新し、ダウンロードする拡張機能マニフェストを表示するには 1.3.1 バージョンを選択する必要があることを示しました。TKG 拡張機能 v1.3.1 バンドルのダウンロードを参照してください。 ■ vSphere with Tanzu の vSphere ネットワークと NSX Advanced Load Balancer を構成する際の、vSphere ポッドの CIDR 範囲の要件を更新しました。vSphere ネットワークと NSX Advanced Load Balancer を使用して vSphere with Tanzu をセットアップするためのシステム要件を参照してください。
2021 年 12 月 17 日	TKG 1.3.1 拡張機能の前提条件トピックを更新し、Kapp Controller 構成を編集してプロキシ サーバを追加する方法を説明しました。TKG 拡張機能の前提条件のインストールを参照してください。

リビジョン	説明
2021 年 12 月 10 日	■ Tanzu Kubernetes Grid サービス TKGS v1alpha2 API を使用した Tanzu Kubernetes クラスタのプロビジョニング をサポートするために、Tanzu Kubernetes クラスタの更新 のトピックを更新しました。クラスタ仕様を TKGS v1alpha2 API に変換した後の Tanzu Kubernetes リリースの更新も参照してください。 ■ お客様からのフィードバックに対応するため、TKG 拡張機能のドキュメントを更新しました。Tanzu Kubernetes クラスタへの TKG パッケージのデプロイを参照してください。 ■ TKGS プロキシ構成のスクリーンショットと要件を更新しました。TKGS v1alpha2 API の構成パラメータを参照してください。
2021 年 11 月 24 日	vSphere with Tanzu の vSphere ネットワークと NSX Advanced Load Balancer の構成に関するドキュメントを更新しました。vSphere ネットワークと vSphere with Tanzu 用 NSX Advanced Load Balancer の構成を参照してください
2021 年 11 月 05 日	Tanzu Kubernetes Grid サービス によってプロビジョニングされた Tanzu Kubernetes クラスタに TKG 1.4 パッケージをインストールするためのリンクを追加しました。Tanzu Kubernetes クラスタへの TKG パッケージのデプロイを参照してください。
2021 年 10 月 29 日	■ TKGS クラスタに vGPU ワークロードをデプロイするためのドキュメントを更新しました。Tanzu Kubernetes クラスタへの AI/ML ワークロードのデプロイを参照してください。 ■ Velero Plugin for vSphere のインストールに関するドキュメントを更新しました。スーパーバイザー クラスタ での Velero Plugin for vSphere のインストールと構成を参照してください。 ■ RBAC の例を更新しました。ポッド セキュリティ ポリシーのロール バインドの例を参照してください。 ■ スーパーバイザー クラスタ ネットワークを更新しました。スーパーバイザー クラスタ ネットワークを参照してください。 ■ ネットワークに関する注意事項と、ワークロード管理の有効化での前提条件に関するトピックを追加しました。これには、スーパーバイザー クラスタ での DRS の無効化の禁止と、DRS を無効にするとクラスタが破損することが記載されています。
2021 年 10 月 21 日	■ vGPU 対応 TKGS クラスタに AI/ML ワークロードをデプロイするためのドキュメントを追加しました。Tanzu Kubernetes クラスタへの AI/ML ワークロードのデプロイを参照してください。 ■ パススルー モードでの PCI デバイスのサポートに関する情報を追加して vSphere with Tanzu での仮想マシン クラスへの PCI デバイスの追加 を更新しました。 ■ Tanzu Kubernetes リリース のリストは専用のリリース ノートに移動しました。Tanzu Kubernetes リリース のすべての詳細については、これらのリリース ノートを参照してください。 ■ 誤字とドキュメントのマイナー バグを修正しました。
2021 年 10 月 08 日	■ Tanzu Kubernetes リリース バージョンを更新しました。更新のための Tanzu Kubernetes クラスタ互換性の確認と TKGS v1alpha2 API を使用した Tanzu Kubernetes クラスタのプロビジョニングを参照してください。 ■ エアギャップ環境に Velero Plugin for vSphere をインストールする手順を追加しました。詳細については、『スーパーバイザー クラスタ での Velero Plugin for vSphere のインストールと構成』を参照してください。 ■ スーパーバイザー クラスタのバックアップとリストアに関する情報を更新しました。詳細については、『vSphere with Tanzu のバックアップとリストアに関する考慮事項』を参照してください。 ■ タイプミスを修正しました。
2021 年 10 月 05 日	初期リリース。

vSphere with Tanzu の概念

2

vSphere with Tanzu を使用すると、vSphere クラスタを専用のリソース プールで Kubernetes ワークロードを実行するためのプラットフォームに変換できます。vSphere with Tanzu を vSphere クラスタで有効にすると、Kubernetes 制御プレーンがハイパーバイザー レイヤーに直接作成されます。vSphere ポッド をデプロイして Kubernetes コンテナを実行することができます。また、VMware Tanzu™ Kubernetes Grid™ サービス を使用してアップストリームの Kubernetes クラスタを作成し、これらのクラスタ内でアプリケーションを実行することもできます。

この章には、次のトピックが含まれています。

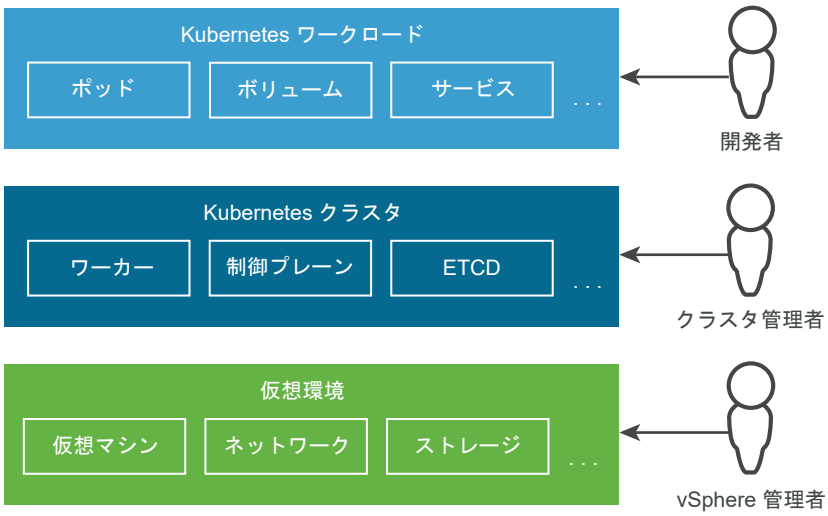
vSphere with Tanzu について

vSphere with Tanzu を使用すると、vSphere を、Kubernetes ワークロードをハイパーバイザー レイヤーでネイティブに実行するためのプラットフォームに変換できます。vSphere クラスタで vSphere with Tanzu を有効にすると、Kubernetes ワークロードを ESXi ホストで直接実行し、専用リソース プール内にアップストリーム Kubernetes クラスタを作成する機能が提供されます。

現在のアプリケーション スタックについての課題

現在の分散システムは、一般に多数の Kubernetes ポッドと仮想マシンを実行する複数のマイクロサービスから構成されています。vSphere with Tanzu に基づかない典型的なスタックは、各仮想マシン内に Kubernetes インフラストラクチャがデプロイされた基盤となる仮想環境と、これらの仮想マシンでそれぞれ実行される Kubernetes ポッドで構成されます。スタックの各部分は、アプリケーション開発者、Kubernetes クラスタ管理者、および vSphere 管理者の 3 種類のロールによって操作されます。

図 2-1. 現在のアプリケーション スタック



各ロールは、互いの環境を可視化または制御できません。

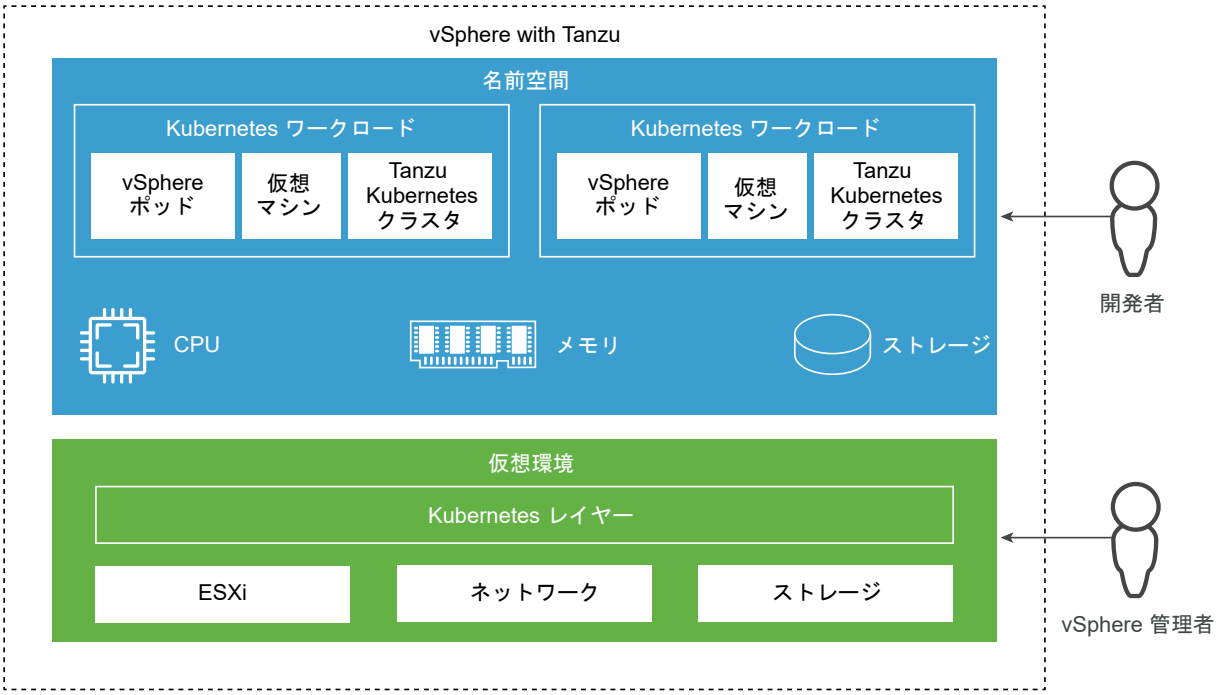
- アプリケーション開発者は、Kubernetes ポッドの実行と、Kubernetes ベースのアプリケーションのデプロイおよび管理を行うことができます。数百のアプリケーションを実行しているスタック全体は可視化できません。
- DevOps エンジニアは Kubernetes インフラストラクチャのみを制御することができます。仮想環境を管理または監視することや、リソース関連の問題やその他の問題を解決することはできません。
- vSphere 管理者は、基盤となる仮想環境を完全に制御できますが、Kubernetes インフラストラクチャ、仮想環境内でのさまざまな Kubernetes オブジェクトの配置、およびそれらのオブジェクトによるリソースの使用を可視化することはできません。

3 つのすべてのロール間の通信が必要になるため、スタック全体の操作は困難になる可能性があります。また、スタックの異なるレイヤーが統合されていないことが問題となる場合もあります。たとえば、Kubernetes スケジューラには vCenter Server インベントリに対する可視性がないため、ポッドをインテリジェントに配置することができません。

vSphere with Tanzu を使用するメリット

vSphere with Tanzu は、Kubernetes 制御プレーンをハイパーバイザー レイヤーに直接作成します。vSphere 管理者は、既存の vSphere クラスタで [ワークロード管理] を有効にして、クラスタに含まれている ESXi ホスト内に Kubernetes レイヤーを作成します。[ワークロード管理] が有効なクラスタを、スーパーバイザー クラスタ と言います。

図 2-2. vSphere with Tanzu



ハイパーバイザー レイヤーに Kubernetes 制御プレーンがあると、vSphere で次の機能が実現します。

- vSphere 管理者は、vSphere 名前空間 という スーパーバイザー クラスタ 上の名前空間を作成し、指定された容量のメモリ、CPU、ストレージを使用して名前空間を構成することができます。構成した vSphere 名前空間は、DevOps エンジニアに提供します。
- DevOps エンジニアは、同じプラットフォームの Kubernetes コンテナで構成されるワークロードを、vSphere 名前空間 内の共有リソース プールで実行できます。vSphere with Tanzu では、コンテナは vSphere ポッド と呼ばれる特別なタイプの仮想マシン内で実行されます。通常の仮想マシンをデプロイすることもできます。
- DevOps エンジニアは、名前空間内の複数の Kubernetes クラスタを作成および管理し、Tanzu Kubernetes Grid サービス を使用してライフサイクルを管理できます。Tanzu Kubernetes Grid サービス を使用して作成された Kubernetes クラスタは、Tanzu Kubernetes クラスタと呼ばれます。
- vSphere 管理者は、vSphere Client を使用して、vSphere ポッド、仮想マシン、および Tanzu Kubernetes クラスタを管理および監視できます。
- vSphere 管理者は、異なる名前空間内で実行されている vSphere ポッド、仮想マシン、Tanzu Kubernetes クラスタ、環境内でのそれらの配置、およびそれらのオブジェクトによるリソースの使用方法を完全に可視化できます。

ハイパーバイザー レイヤーで Kubernetes を実行していると、vSphere 管理者と DevOps チームの両方のロールが同じオブジェクトを操作するため、共同作業も容易になります。

ワークロードについて

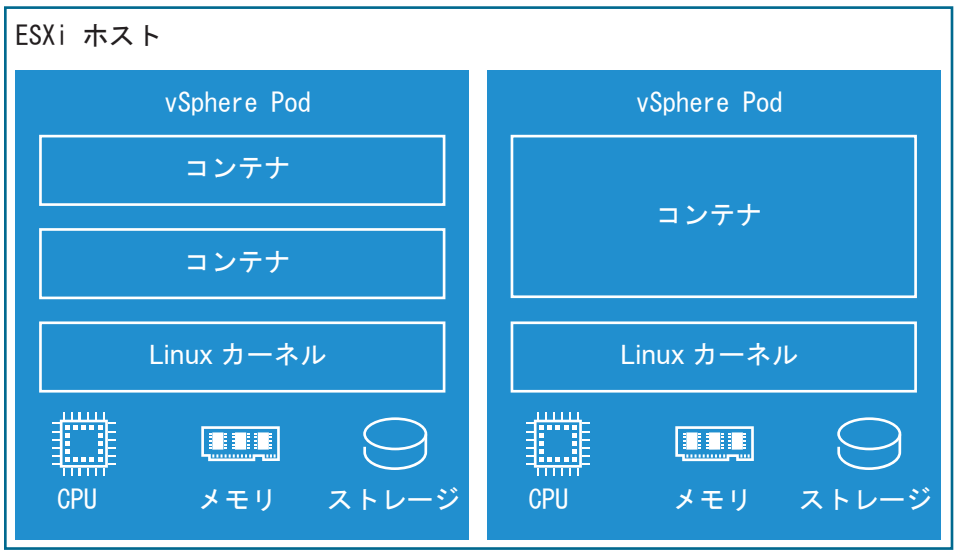
vSphere with Tanzu では、ワークロードとは次のいずれかの方法でデプロイされたアプリケーションを指します。

- vSphere ポッド、通常の仮想マシン、またはその両方で実行されているコンテナで構成されるアプリケーション。
- VMware Tanzu™ Kubernetes Grid™ サービス を使用してデプロイされた Tanzu Kubernetes クラスタ。
- VMware Tanzu™ Kubernetes Grid™ サービス を使用してデプロイされた Tanzu Kubernetes クラスタ内で実行されるアプリケーション。

vSphere ポッド について

vSphere with Tanzu では、Kubernetes ポッドに相当する vSphere ポッド と呼ばれる新しい構造が導入されています。vSphere ポッド は、1 つ以上の Linux コンテナを実行する占有量の小さい仮想マシンです。各 vSphere ポッド は、格納するワークロードに応じて正確にサイズ調整され、そのワークロードに対して明示的なリソース予約を保持します。これにより、ワークロードの実行に必要な量のストレージ、メモリ、および CPU リソースが正確に割り当てられます。vSphere ポッド は、ネットワーク スタックとして NSX-T Data Center を使用して構成された スーパーバイザー クラスタ でのみサポートされます。

図 2-3. vSphere ポッド



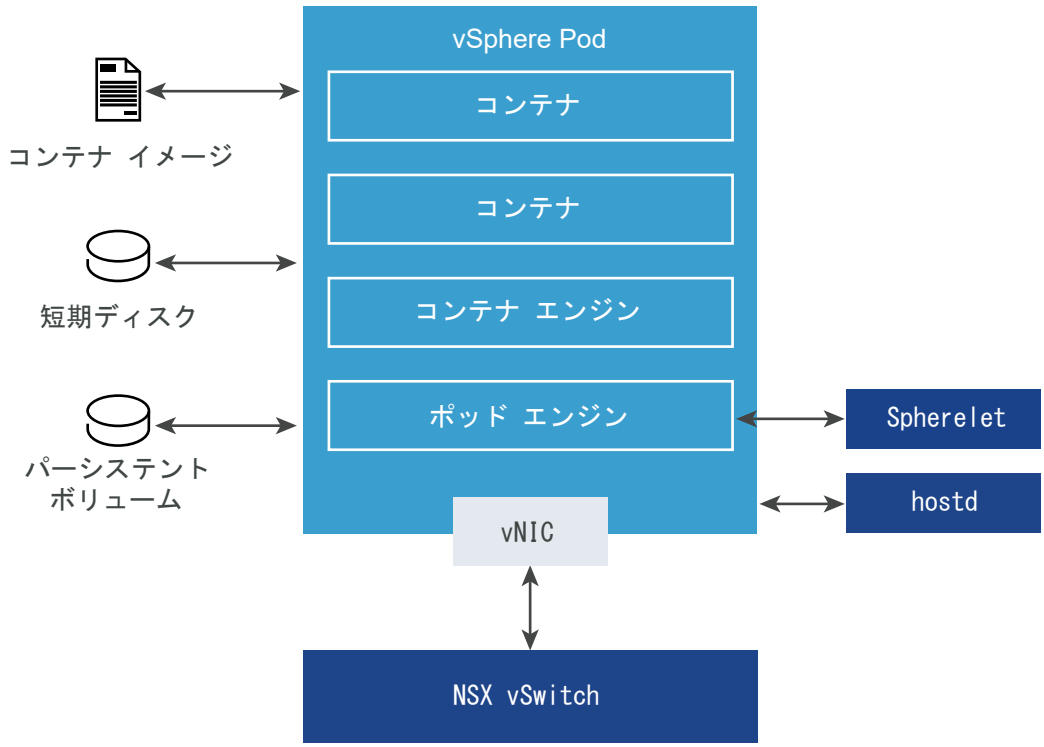
vSphere ポッド は vCenter Server 内のオブジェクトであるため、ワークロードに対して次の機能を実現します。

- 高レベルの隔離。vSphere ポッド は、仮想マシンと同じ方法で隔離されます。各 vSphere ポッド には、Photon OS で使用されるカーネルに基づく独自の Linux カーネルがあります。vSphere ポッド では、ベアメタル構成と異なり、多くのコンテナでカーネルを共有することはありません。コンテナごとに一意の Linux カーネルがあります
- リソース管理。vSphere DRS により、スーパーバイザー クラスタ 上の vSphere ポッド の配置が処理されます。

- 高パフォーマンス。vSphere ポッド では、仮想マシンと同じレベルのリソース隔離が実現するため、高速な起動時間と、コンテナの低オーバーヘッドを維持しながら、ノイジー ネイバー問題を回避できます。
- 診断。vSphere 管理者は、ワークロード上の vSphere で使用可能なすべての監視ツールおよびイントロスペクション ツールを使用できます。

vSphere ポッド は Open Container Initiative (OCI) と互換性があり、任意のオペレーティング システムのコンテナを実行できます（コンテナも OCI 互換の場合に限る）。

図 2-4. vSphere ポッド ネットワークおよびストレージ



vSphere ポッド では、格納するオブジェクトに応じて、短期 VMDK、パーシステント ボリューム VMDK、およびコンテナ イメージ VMDK の 3 種類のストレージを使用します。vSphere 管理者は、スーパーバイザー クラスタ レベルで、コンテナ イメージ キャッシュ、短期 VMDK、および制御プレーン仮想マシンを配置するためのストレージ ポリシーを設定します。vSphere 名前空間 レベルでは、パーシステント ボリュームの配置用と Tanzu Kubernetes クラスタの仮想マシンの配置用のストレージ ポリシーを構成します。vSphere with Tanzu におけるストレージの要件と概念の詳細については、10 章 vSphere with Tanzu でのパーシステント ストレージの使用を参照してください。

ネットワークについては、vSphere ポッド と、Tanzu Kubernetes Grid サービス を使用して作成された Tanzu Kubernetes クラスタの仮想マシンは、NSX-T Data Center によって提供されるトポロジを使用します。詳細については、スーパーバイザー クラスタ ネットワークを参照してください。

vSphere ポッド は、ネットワーク スタックとして NSX-T Data Center を使用する スーパーバイザー クラスタでのみサポートされます。vSphere ネットワーク スタックを使用して構成されたクラスタではサポートされません。

Tanzu Kubernetes クラスタについて

Tanzu Kubernetes クラスタは、VMware によってビルド、署名、およびサポートされているオープンソースの Kubernetes コンテナ オーケストレーション プラットフォームの完全なディストリビューションです。Tanzu Kubernetes Grid サービス を使用することで、Tanzu Kubernetes クラスタを スーパーバイザー クラスタ 上にプロビジョニングして運用することができます。スーパーバイザー クラスタ は、vSphere with Tanzu で有効にされる vSphere クラスタです。

Tanzu Kubernetes Grid サービス によって作成される Tanzu Kubernetes クラスタの主な特性

Tanzu Kubernetes Grid サービス によってプロビジョニングされる Tanzu Kubernetes クラスタには、次の特性があります。

- Kubernetes の個人用インストール
- vSphere インフラストラクチャとの統合
- 本番環境に対応
- VMware による完全サポート
- Kubernetes による管理

注： VMware は、Kubernetes 対応のスイート製品を Tanzu ブランド名で提供しています。Tanzu Kubernetes Grid サービス を使用して作成する Tanzu Kubernetes クラスタは、Tanzu エディション ライセンスの対象になります。Tanzu ブランドで VMware が提供している他の Kubernetes 対応製品の詳細については、VMware Tanzu ドキュメントを参照してください。vSphere with Tanzu のライセンスの詳細については、vSphere with Tanzu のライセンスを参照してください。

Kubernetes の個人用インストール

Tanzu Kubernetes クラスタは、Kubernetes の個人用インストールです。

Tanzu Kubernetes Grid サービス では、vSphere で Tanzu Kubernetes クラスタをプロビジョニングするために十分に考慮され、最適化されたデフォルト値が提供されています。Tanzu Kubernetes Grid サービス を使用することで、通常エンタープライズ クラスの Kubernetes クラスタを展開および実行する際に必要になる時間と労力の削減が可能になります。

詳細については、『Tanzu Kubernetes Grid サービス アーキテクチャ』を参照してください。

vSphere インフラストラクチャとの統合

Tanzu Kubernetes クラスタは、基盤となる vSphere インフラストラクチャと統合されており、Kubernetes を実行するために最適化されています。

Tanzu Kubernetes クラスタは、ストレージ、ネットワーク、認証などを含めて、vSphere Software-Defined Data Center (SDDC) スタックと緊密に統合されています。また、Tanzu Kubernetes クラスタは、vCenter Server クラスタにマッピングされる スーパーバイザー クラスタ 上に構築されます。緊密に統合されているため、他の製品と同様な操作方法で Tanzu Kubernetes クラスタを実行することができます。

詳細については、vSphere with Tanzu アーキテクチャを参照してください。

本番環境に対応

Tanzu Kubernetes クラスタは、本番ワークロードの実行に合わせて調整されます。

Tanzu Kubernetes Grid サービス は、本番環境に対応した Tanzu Kubernetes クラスタをプロビジョニングします。追加の設定を行う必要なく、本番環境のワークロードを実行できます。また、可用性を確保し、Kubernetes ソフトウェアのローリング アップグレードを可能にし、異なるバージョンの Kubernetes を別々のクラスタで実行することもできます。

詳細については、『13 章 TKGS クラスタのプロビジョニングと操作』を参照してください。

VMware による完全サポート

Tanzu Kubernetes クラスタは、VMware によってサポートされています。

Tanzu Kubernetes クラスタは、VMware が提供するオープンソースの Linux ベース Photon OS を使用し、vSphere インフラストラクチャにデプロイされ、ESXi ホストで実行されます。ハイパーバイザーから Kubernetes クラスタに至るまで、スタックのどのレイヤーでも問題が発生した場合は、VMware がお客様のお問い合わせ先となります。

詳細については、VMware のサポートにお問い合わせください。

Kubernetes による管理

Tanzu Kubernetes クラスタは Kubernetes によって管理されます。

Tanzu Kubernetes クラスタは、Kubernetes クラスタである スーパーバイザー クラスタ 上に構築されています。Tanzu Kubernetes は、カスタム リソースを使用して vSphere 名前空間 で定義されます。Tanzu Kubernetes クラスタをセルフサービスでプロビジョニングするには、使い慣れた kubectl コマンドを使用します。ツールチェーン全体に整合性があるため、クラスタをプロビジョニングする場合でも、またはワークロードをデプロイする場合でも、同じコマンド、使い慣れた YAML、共通のワークフローを使用できます。

詳細については、『Tanzu Kubernetes クラスタのテナント モデル』を参照してください。

vSphere ポッド と Tanzu Kubernetes クラスタの使い分け

vSphere ポッド と Tanzu Kubernetes Grid サービス によってプロビジョニングされた Tanzu Kubernetes クラスタのどちらを使用するかは、スーパーバイザー クラスタ での Kubernetes ワークロードのデプロイと管理に関連する目的によって決まります。

vSphere 管理者または DevOps エンジニアが次の操作を行う場合は、vSphere ポッド を使用します。

- コンテナを実行する（Kubernetes クラスタをカスタマイズする必要はない）。
- リソースとセキュリティの隔離レベルが高いコンテナ化されたアプリケーションを作成する。
- ESXi ホストに vSphere ポッド を直接デプロイする。

DevOps エンジニアまたは開発者が次の操作を行う場合は、Tanzu Kubernetes Grid サービス によってプロビジョニングされた Tanzu Kubernetes クラスタを使用します。

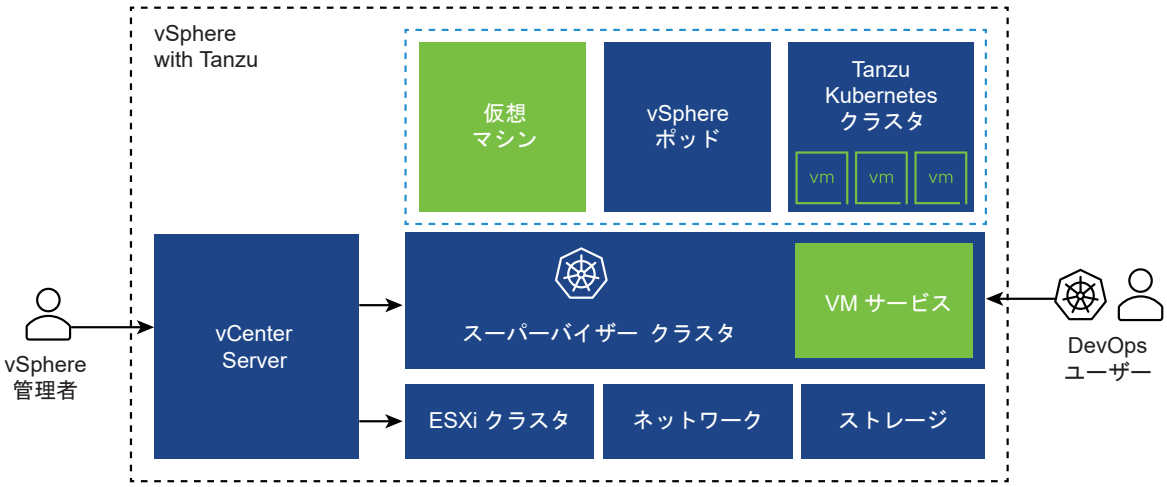
- コンテナ化されたアプリケーションをコミュニティに合わせたオープンソースの Kubernetes ソフトウェアで実行する。
- 制御プレーン ノードとワーカー ノードへの root レベルのアクセスを含めて、Kubernetes クラスタを制御する。
- Kubernetes のバージョンを最新の状態に保つ（インフラストラクチャのアップグレードは不要）。
- CI/CD パイプラインを使用して、有効期間の短い Kubernetes クラスタをプロビジョニングする。
- Kubernetes クラスタをカスタマイズする（たとえば、カスタム リソース定義、演算子、helm チャートをインストールする）。
- kubectl CLI を使用して Kubernetes 名前空間を作成する。
- クラスタレベルのアクセス コントロールを管理し、`PodSecurityPolicies.` を設定する
- `NodePort.` タイプのサービスを作成する。
- `HostPath` ボリュームを使用する。
- 特権ポッドを実行する。

vSphere with Tanzu での仮想マシンの使用

vSphere with Tanzu で提供されている仮想マシン サービス機能を使用すると、DevOps エンジニアは一般的な共有 Kubernetes 環境でコンテナに加え、仮想マシンをデプロイして実行することができます。コンテナと仮想マシンの両方が同じ vSphere 名前空間 リソースを共有するため、単一の vSphere with Tanzu インターフェイスを通して管理することができます。

仮想マシン サービスは、Kubernetes を使用する DevOps チームのニーズに対応しますが、既存の仮想マシン ベースのワークロードには容易にコンテナ化できないものがあります。仮想マシン サービスを使用することで、コンテナ プラットフォームと一緒に Kubernetes 以外のプラットフォームを管理する場合のオーバーヘッドを削減することもできます。Kubernetes プラットフォーム上でコンテナと仮想マシンを実行する場合、DevOps チームはワークロードの占有量を 1 つのプラットフォームに統合できます。

注： 仮想マシン サービスは、スタンドアローン仮想マシンのほか、Tanzu Kubernetes クラスタを構成する仮想マシンも管理します。クラスタの詳細については、Tanzu Kubernetes Grid サービス アーキテクチャおよび 13 章 TKGS クラスタのプロビジョニングと操作を参照してください。



仮想マシン サービスを使用してデプロイされた各仮想マシンは、vSphere with Tanzu インフラストラクチャ上で、独自のオペレーティング システムを含むすべてのコンポーネントを実行する完全なマシンとして機能します。仮想マシンは、スーパーバイザー クラスタ が提供するネットワークおよびストレージにアクセスすることができ、標準の Kubernetes コマンド `kubectl` を使用して管理されます。仮想マシンは完全に隔離されたシステムとして動作し、Kubernetes 環境内の他の仮想マシンまたはワークロードからの干渉を受けません。

Kubernetes プラットフォームで仮想マシンを使用する場合

一般に、コンテナと仮想マシンのどちらでワークロードを実行するかは、ビジネス ニーズおよび目標に応じて決まります。仮想マシンを使用する理由には、次のようなものがあります。

- アプリケーションのコンテナ化ができません。
- プロジェクトに固有のハードウェア要件があります。
- アプリケーションが、カスタム カーネルまたはカスタム オペレーティング システム用に設計されています。
- アプリケーションが、仮想マシンで実行するのに適しています。
- 一貫性のある Kubernetes 環境により、オーバーヘッドを回避することを検討しています。Kubernetes 以外のプラットフォームとコンテナ プラットフォームにインフラストラクチャ セットを個別に実行するのではなく、これらのスタックを統合し、使い慣れた `kubectl` コマンドを使用して管理することができます。

仮想マシンのデプロイおよび管理の詳細については、12 章 vSphere with Tanzu での仮想マシンのデプロイと管理を参照してください。

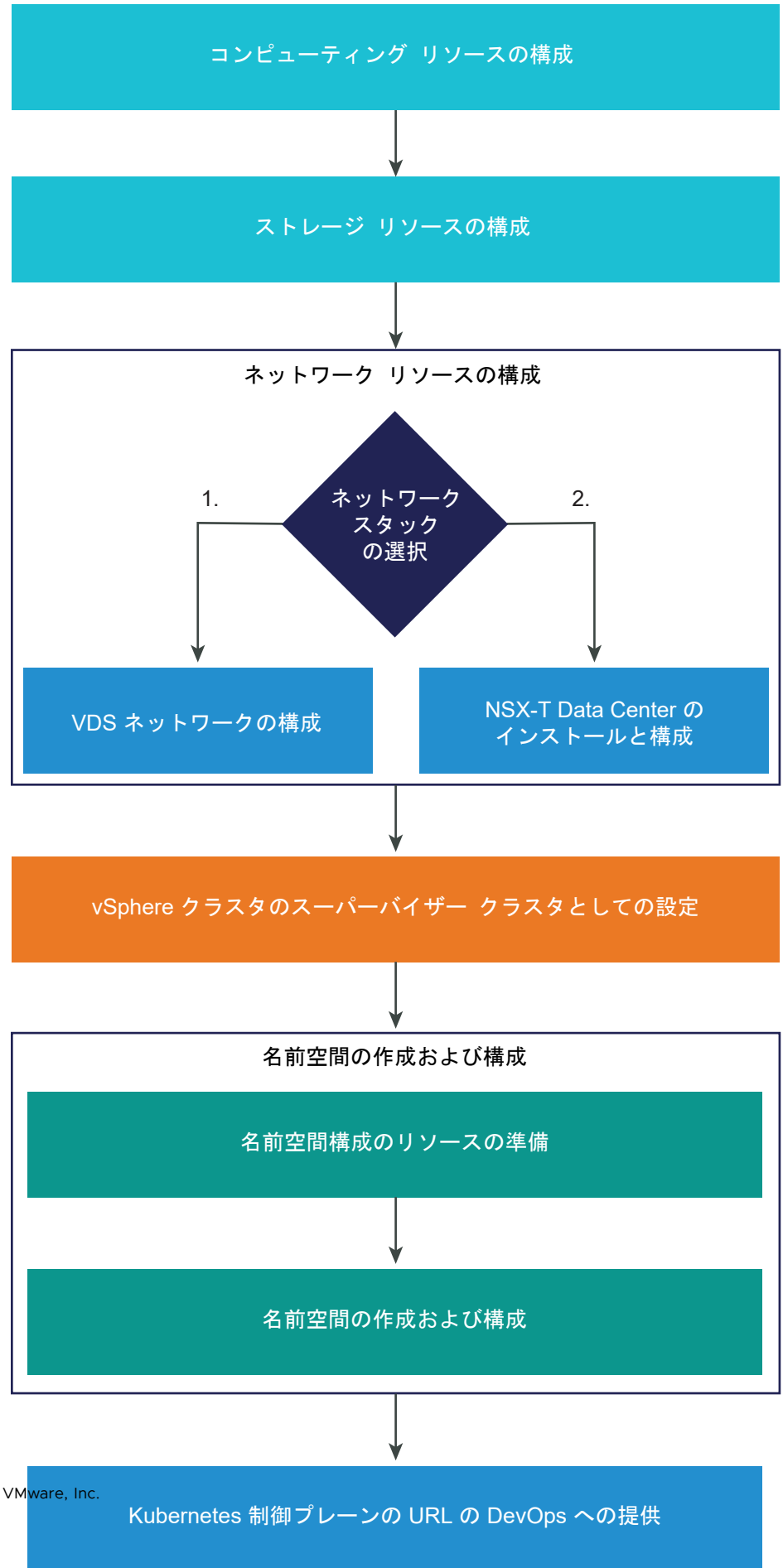
vSphere with Tanzu のユーザー ロールとワークフロー

vSphere with Tanzu プラットフォームには、vSphere 管理者と DevOps エンジニアの 2 つのロールがあります。これらのロールは、それぞれ異なるインターフェイスを通じてプラットフォームと通信します。vCenter Single Sign-On でこれらのロールのユーザーまたはユーザー グループを定義し、権限を関連付けることができます。vSphere 管理者ロールと DevOps エンジニア ロールのワークフローは異なり、これらのロールで必要となる専門知識の特定の領域によって決定されます。

ユーザー ロールとワークフロー

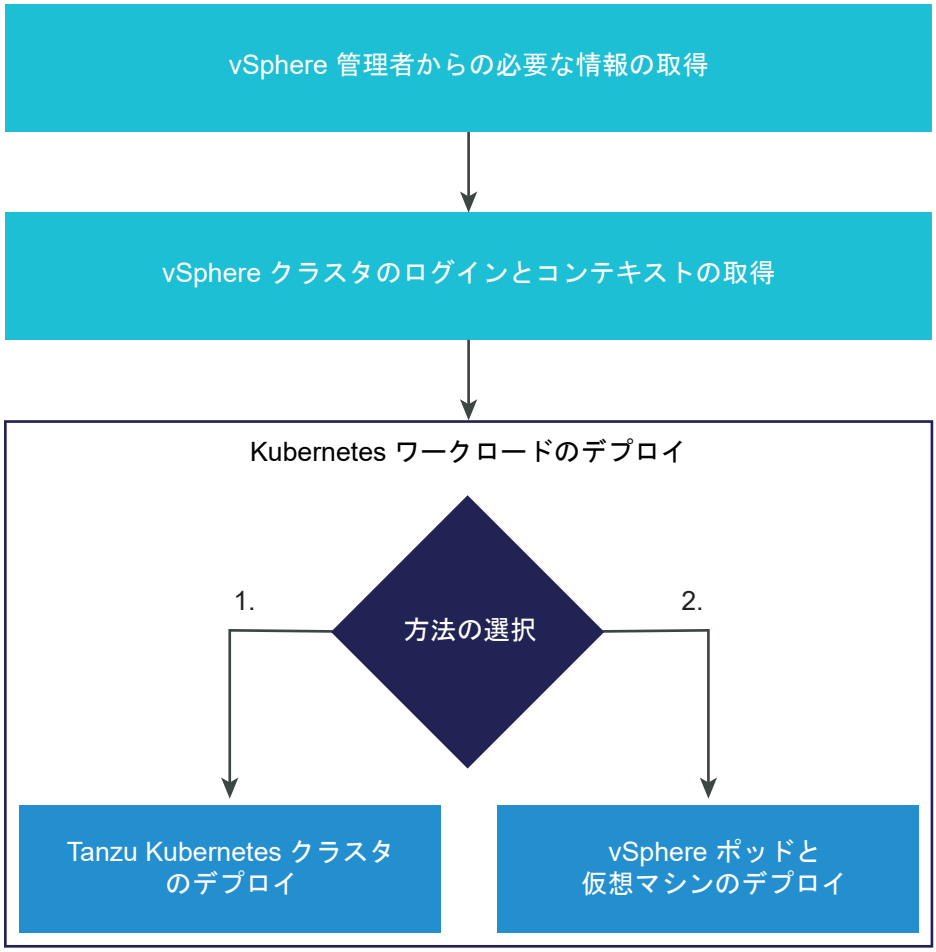
vSphere 管理者が vSphere with Tanzu プラットフォームと通信するための主要インターフェイスは vSphere Client です。大まかに、vSphere 管理者の責務には、DevOps エンジニアが Kubernetes ワークロードをデプロイできるように スーパーバイザー クラスタ と名前空間を構成することが含まれます。vSphere と NSX-T テクノロジーについて優れた知識を持ち、Kubernetes の基本を理解している必要があります。

図 2-5. vSphere 管理者の大まかなワークフロー



DevOps エンジニアには、Kubernetes 開発者およびアプリケーション所有者と、Kubernetes 管理者、またはその両方を組み合わせた役割があります。DevOps エンジニアは、kubectl コマンドを使用して、スーパーバイザー クラスタ 上の既存の名前空間に vSphere ポッド、仮想マシン、および Tanzu Kubernetes のクラスタをデプロイします。通常、DevOps エンジニアは vSphere と NSX-T のエキスパートである必要はありませんが、これらのテクノロジーと vSphere with Tanzu プラットフォームに関する基礎知識があれば vSphere 管理者とより効率的にやり取りすることができます。

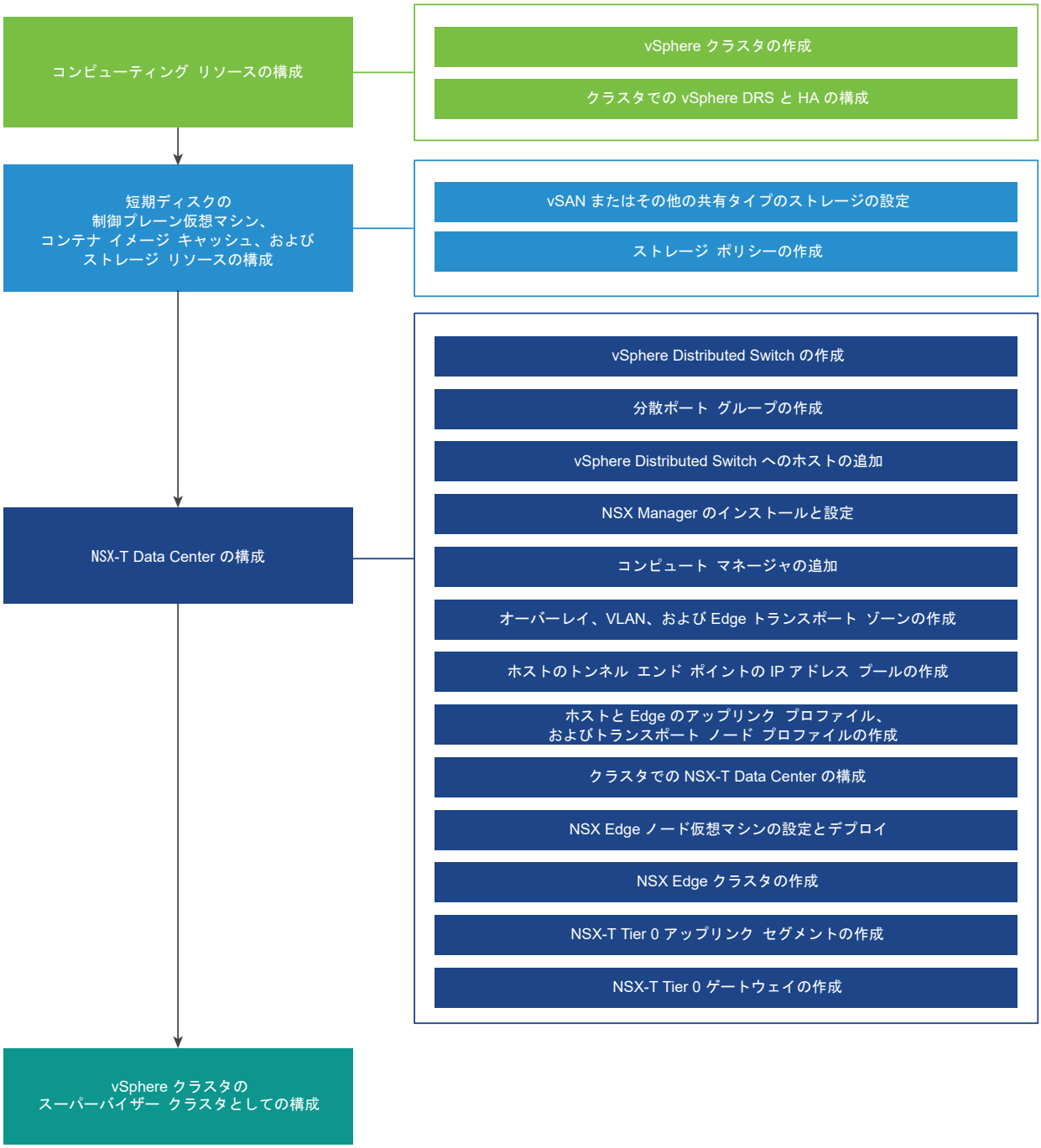
図 2-6. DevOps エンジニアの大まかなワークフロー



NSX-T Data Center を含む スーパーバイザー クラスタ のワークフロー

vSphere 管理者は、必要なコンピューティング、ストレージ、およびネットワーク コンポーネントを備える vSphere with Tanzu プラットフォームを構成します。NSX-T Data Center は、スーパーバイザー クラスタ のネットワーク スタックとして使用できます。システム要件の詳細については、NSX-T Data Center を使用して vSphere with Tanzu をセットアップするためのシステム要件を参照してください。

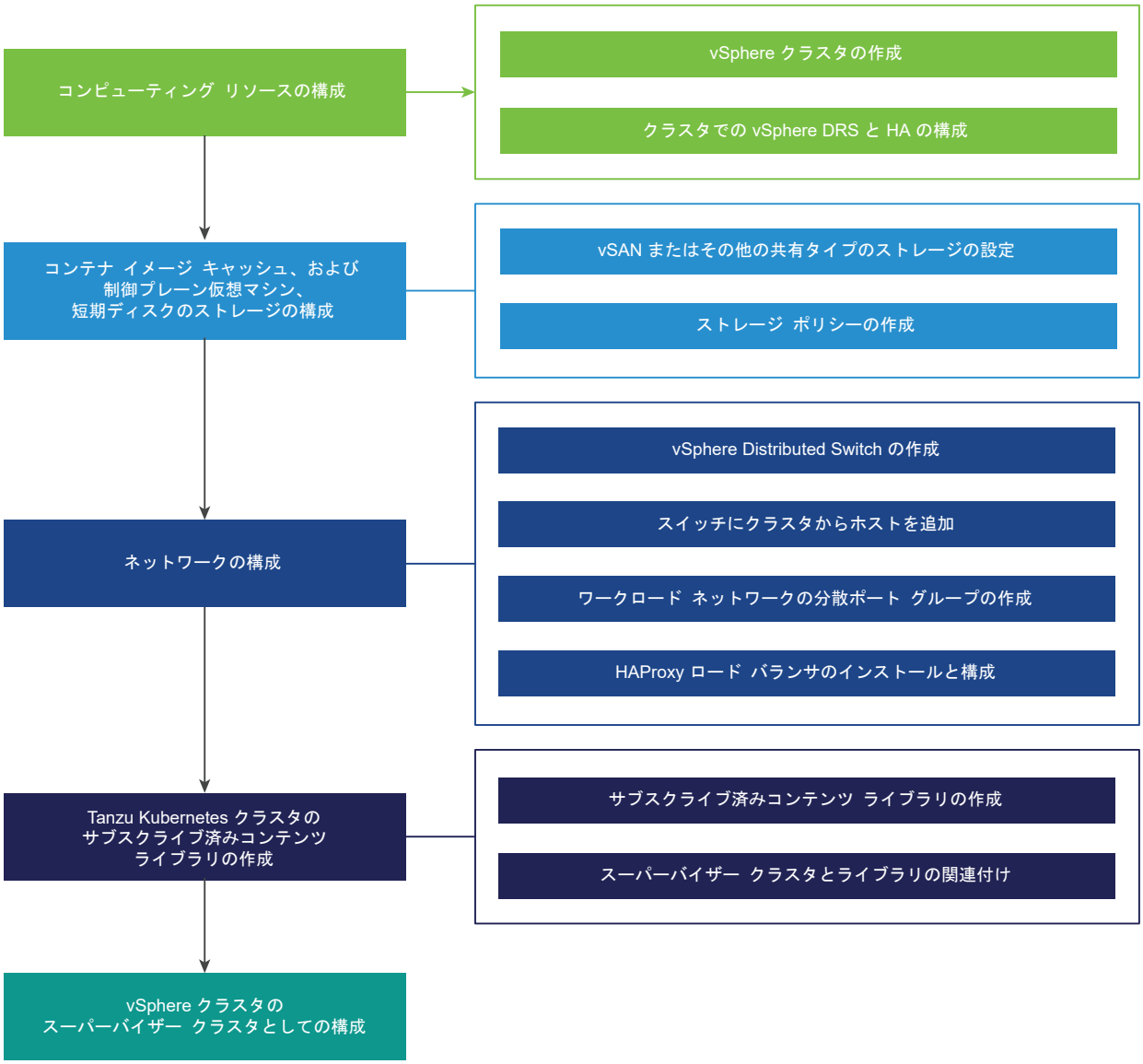
図 2-7. NSX-T Data Center ネットワークを使用する スーパーバイザー クラスタ のワークフロー



vSphere ネットワーク スタックを使用する スーパーバイザー クラスタ のワークフロー

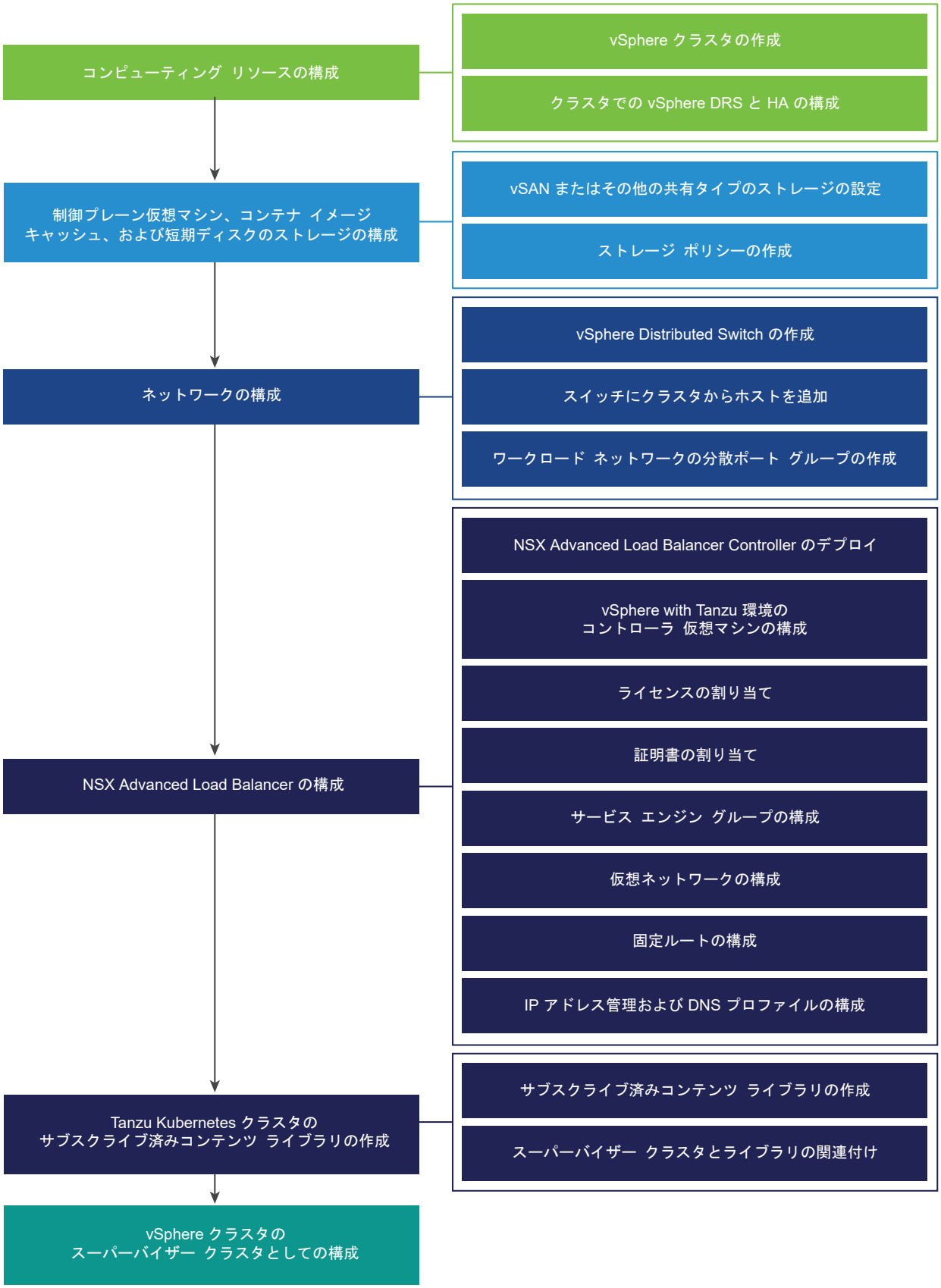
vSphere 管理者は、vSphere クラスタを、vSphere ネットワーク スタックを使用する スーパーバイザー クラスタ として構成できます。システム要件の詳細については、vSphere ネットワークおよび HAProxy ロード バランサを使用して vSphere with Tanzu をセットアップするためのシステム要件を参照してください。

図 2-8. vSphere ネットワーク スタックを使用する スーパーバイザー クラスタ の構成ワークフロー



vSphere ネットワークと NSX Advanced Load Balancer を使用する スーパーバイザー クラスタ のワークフロー

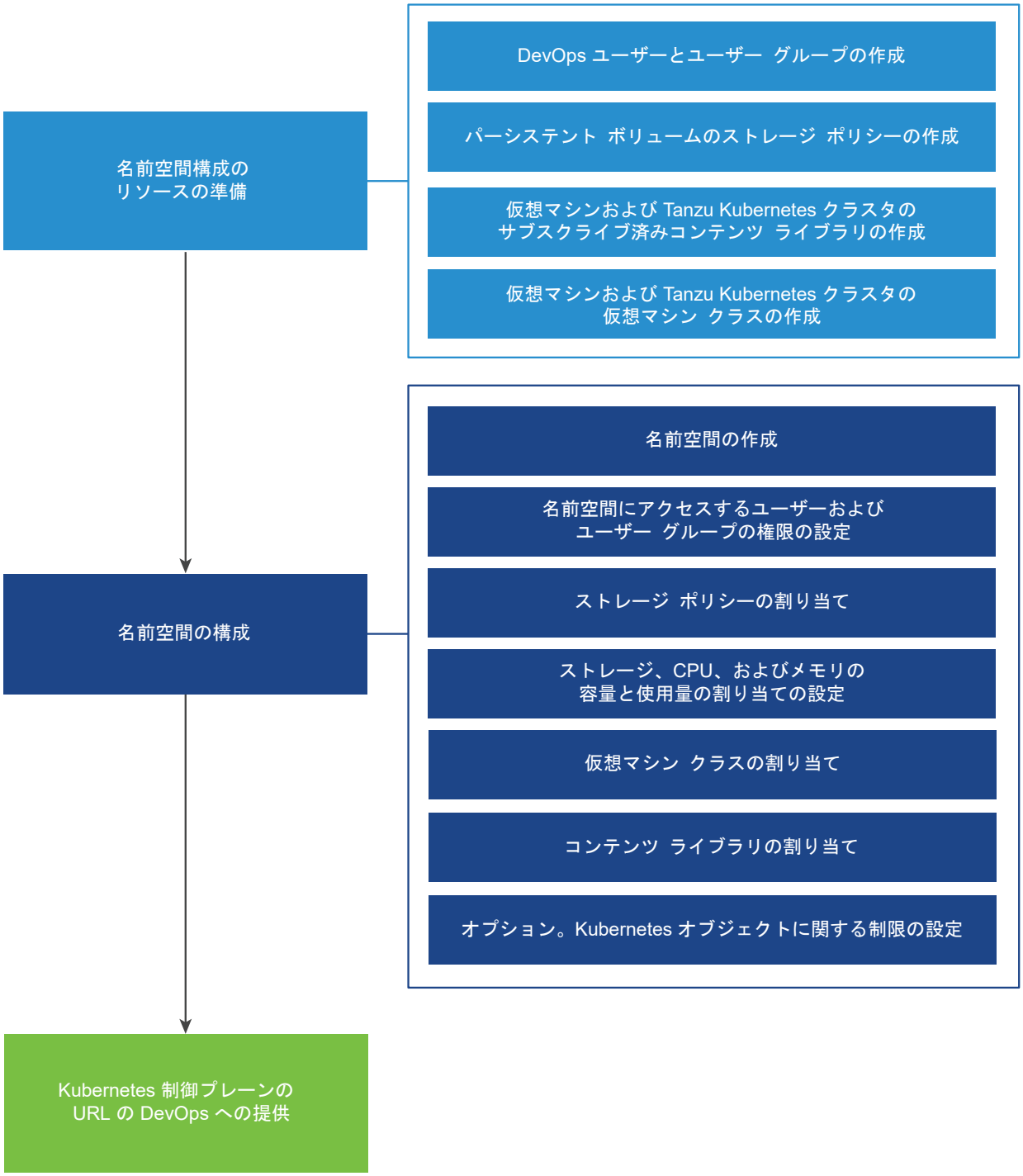
この図では、vSphere ネットワークと vSphere with Tanzu 用 NSX Advanced Load Balancer を構成するためのワークフローを示します。詳細については、『NSX Advanced Load Balancer のインストールと構成』を参照してください。


コンピューティング リソースの構成
vSphere クラスタの作成
クラスタでの vSphere DRS と HA の構成
制御プレーン仮想マシン、コンテナ イメージ キャッシュ、および短期ディスクのストレージの構成
vSAN またはその他の共有タイプのストレージの設定
ストレージ ポリシーの作成
ネットワークの構成
vSphere Distributed Switch の作成
スイッチにクラスタからホストを追加
ワークロード ネットワークの分散ポート グループの作成
NSX Advanced Load Balancer の構成
NSX Advanced Load Balancer Controller のデプロイ
vSphere with Tanzu 環境の コントローラ 仮想マシンの構成
ライセンスの割り当て
証明書の割り当て
サービス エンジン グループの構成
仮想ネットワークの構成
固定ルートの構成
IP アドレス管理および DNS プロファイルの構成
Tanzu Kubernetes クラスタの サブスクライブ済みコンテンツ ライブラリの作成
サブスクライブ済みコンテンツ ライブラリの作成
スーパーバイザー クラスタとライブラリの関連付け
vSphere クラスタの スーパーバイザー クラスタとしての構成

名前空間の作成と構成のワークフロー

vSphere 管理者は、スーパーバイザー クラスタ 上に名前空間を作成して構成します。実行するアプリケーションとワークロードについて DevOps エンジニアから特定のリソース要件を収集し、それに応じて名前空間を構成する必要があります。詳細については、7 章 vSphere 名前空間の構成と管理を参照してください。

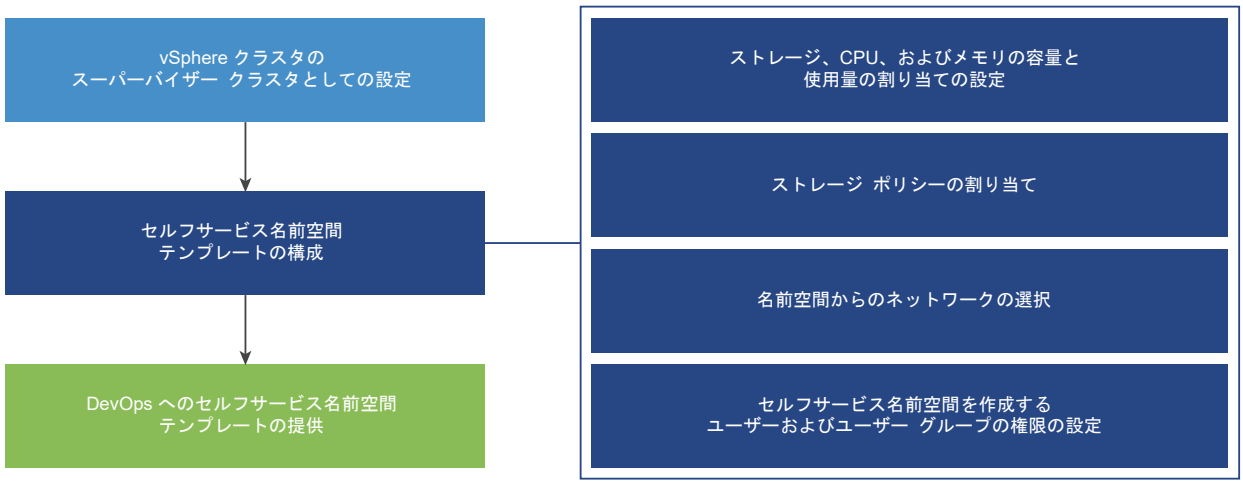
図 2-9. 名前空間の構成のワークフロー



セルフサービス名前空間の作成と構成のワークフロー

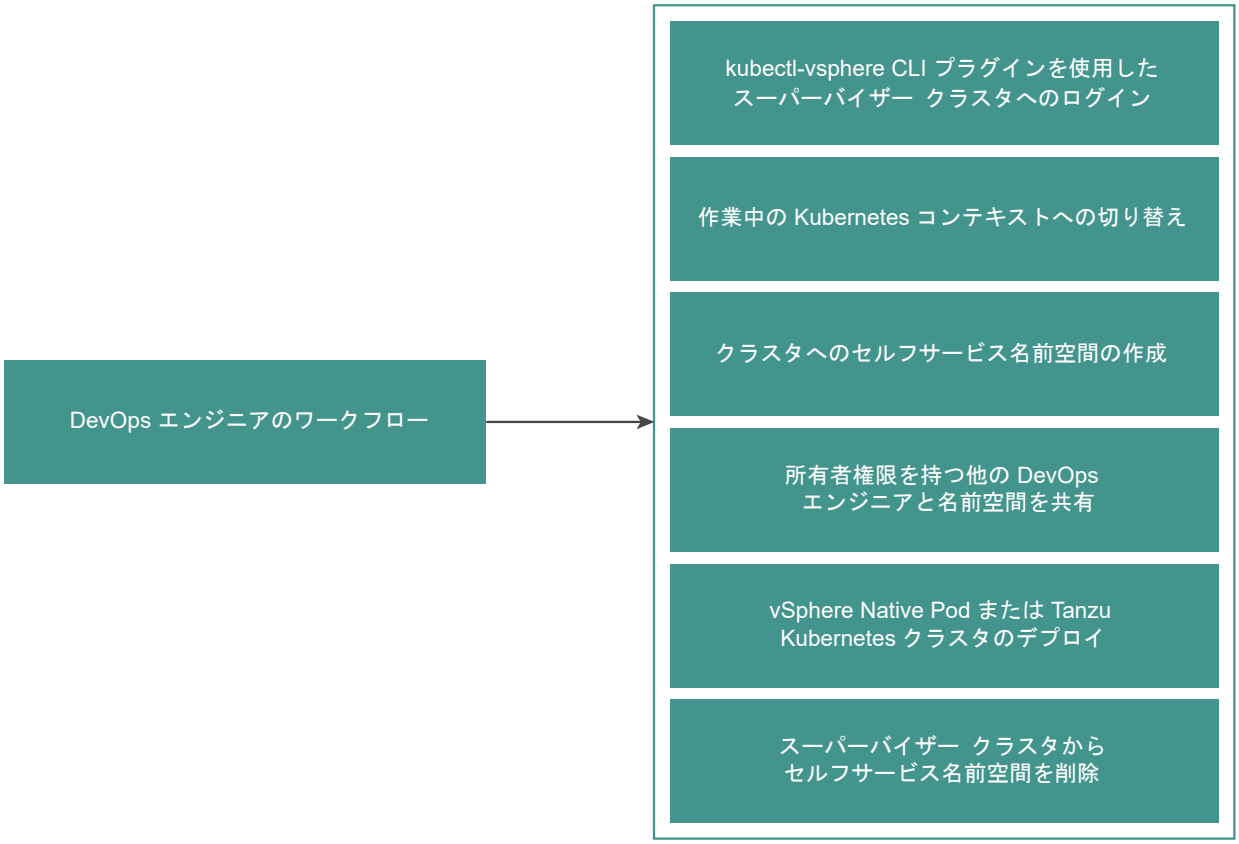
vSphere 管理者は、スーパーバイザー名前空間の作成、名前空間に対する CPU、メモリ、ストレージの制限の設定、権限の割り当て、およびクラスタの名前空間サービスのテンプレートとしてのプロビジョニングと有効化を行うことができます。

図 2-10. セルフサービス名前空間テンプレートのプロビジョニングのワークフロー



DevOps エンジニアは、セルフ サービス方式でスーパーバイザー名前空間を作成し、その中にワークロードをデプロイすることができます。また、他の DevOps エンジニアと名前空間を共有したり、不要になった名前空間を削除したりできます。

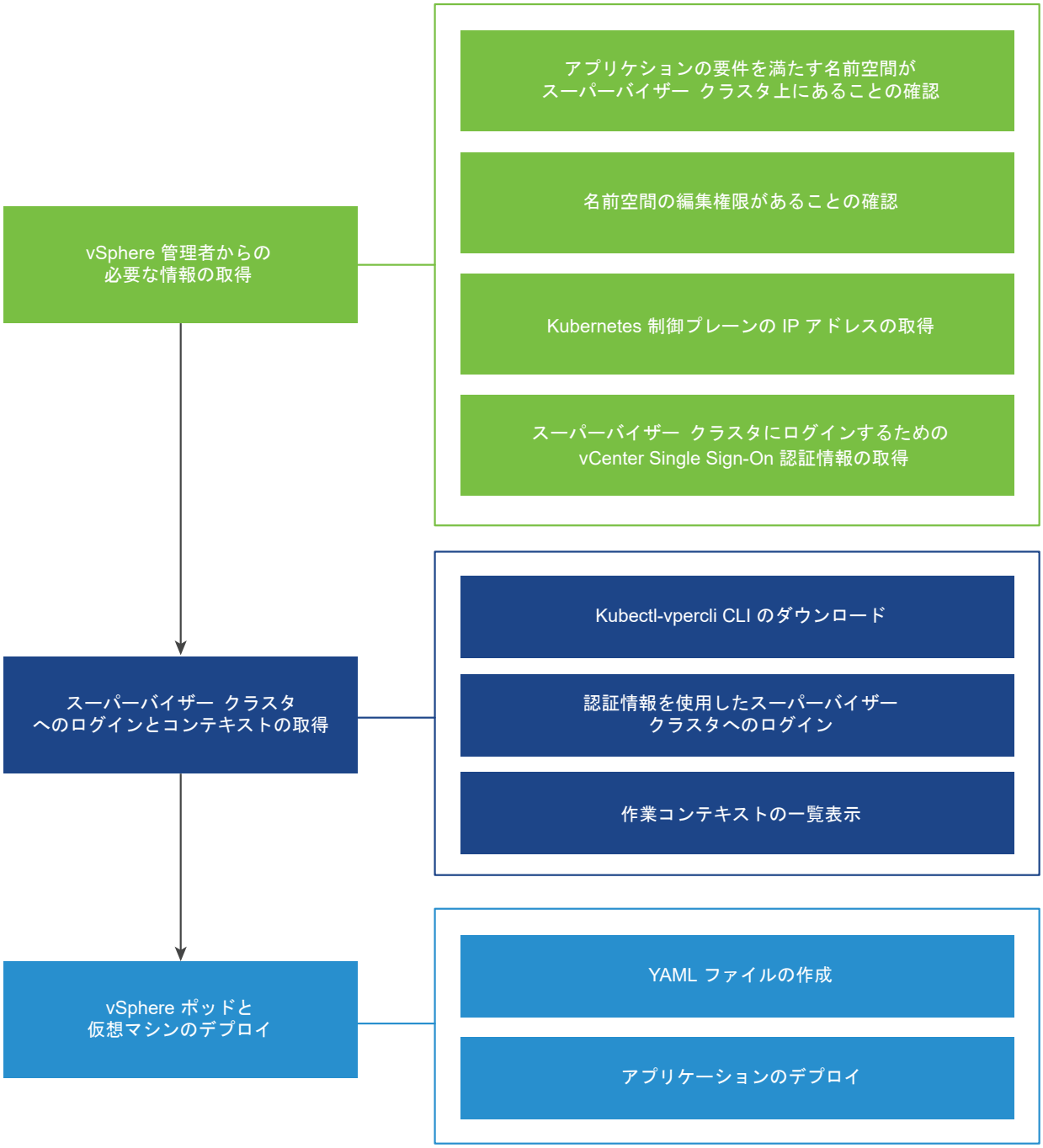
図 2-11. セルフサービス名前空間の作成のワークフロー



vSphere ポッド と仮想マシンのプロビジョニング ワークフロー

DevOps エンジニアは、スーパーバイザー クラスタ で実行されている名前空間のリソースの境界内で vSphere ポッド および仮想マシンをデプロイできます。詳細については、『11 章 vSphere ポッド へのワークロードのデプロイ』および『12 章 vSphere with Tanzu での仮想マシンのデプロイと管理』を参照してください。

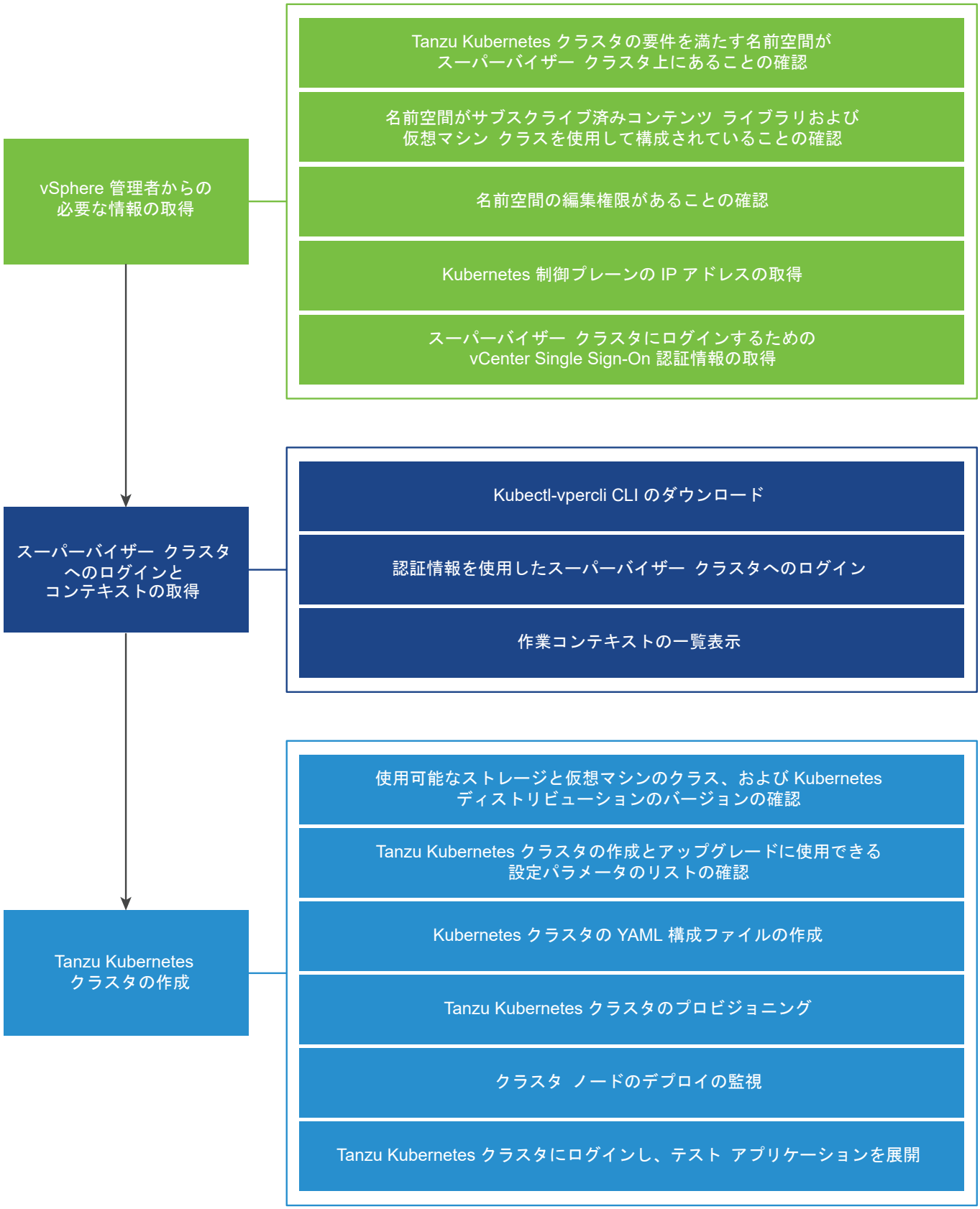
図 2-12. vSphere ポッド と仮想マシンのプロビジョニング ワークフロー



Tanzu Kubernetes クラスタのプロビジョニング ワークフロー

DevOps エンジニアは、vSphere 管理者によって作成および構成された名前空間に Tanzu Kubernetes を作成および構成します。詳細については、『TKGS v1alpha2 API を使用して Tanzu Kubernetes クラスタをプロビジョニングするためのワークフロー』を参照してください。

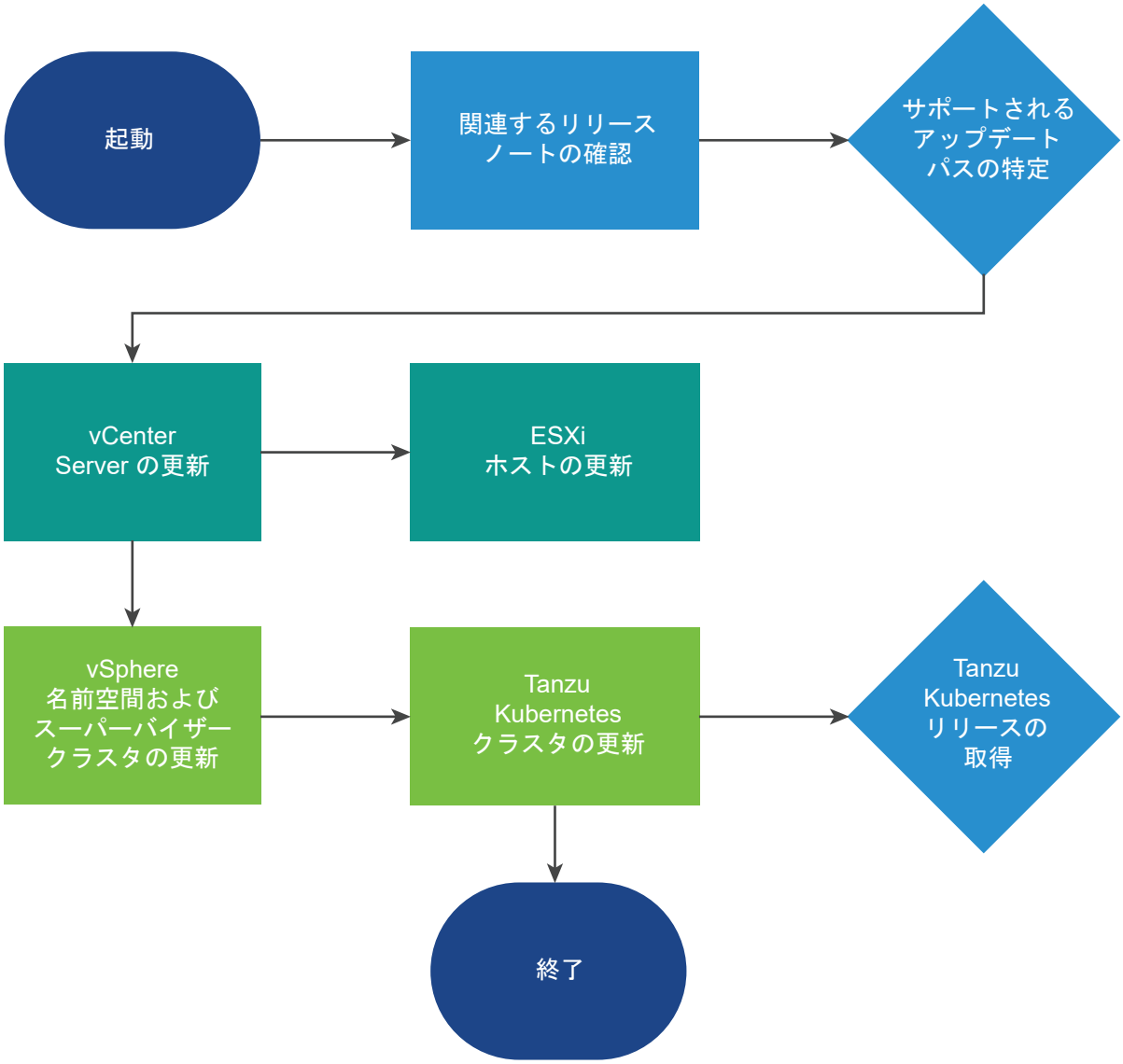
図 2-13. Tanzu Kubernetes クラスタのプロビジョニング ワークフロー



vSphere with Tanzu の更新ワークフロー

次の図に、スーパーバイザー クラスタ や Tanzu Kubernetes クラスタなどの vSphere with Tanzu 環境を更新するためのワークフローを示します。詳細については、『17 章 vSphere with Tanzu 環境の更新』を参照してください。

図 2-14. vSphere with Tanzu の更新ワークフロー



vSphere with Tanzu による vSphere 環境の変革

vSphere クラスタは、Kubernetes ワークロード用に構成されている場合、スーパーバイザー クラスタ になります。これにより、Tanzu Kubernetes Grid サービス を使用してプロビジョニングされた名前空間、vSphere ポッド、Tanzu Kubernetes クラスタなどのオブジェクトが vCenter Server インベントリに追加されます。

各 スーパーバイザー クラスタ で、次のものを表示できます。

- クラスタで実行されている論理アプリケーションを表す名前空間。

- スーパーバイザー クラスタ の各名前空間のリソース プール。

すべての名前空間内で、次のものを表示できます。

- vSphere ポッド。
- Tanzu Kubernetes Grid サービス を介して作成された Kubernetes クラスタ。
- Kubernetes 制御プレーン仮想マシン。
- ネットワークおよびストレージ リソース。
- その名前空間のユーザー権限。

vSphere with Tanzu のライセンス

vSphere with Tanzu 用に構成した vSphere クラスタが スーパーバイザー クラスタ になった場合は、60 日の評価期間が終了する前に、クラスタに Tanzu エディション ライセンスを割り当てる必要があります。

Tanzu ライセンスについて

Tanzu ライセンスを使用すると、vSphere 7.0 Update 1 以降のワークロード管理機能が有効になります。これは、vSphere ネットワーク スタックまたは NSX-T Data Center が構成されている スーパーバイザー クラスタに適用されます。スーパーバイザー クラスタ が vSphere 7.0 で実行されている場合、スーパーバイザー クラスタから各ホストに割り当てられている Kubernetes ライセンスのアドオンを含む VMware vSphere 7 Enterprise Plus が必要です。

vSphere 管理者として、スーパーバイザー クラスタ に Tanzu ライセンスを割り当てる場合は、名前空間を作成および構成し、DevOps エンジニアにこれらの名前空間へのアクセス権を与えます。DevOps エンジニアは、アクセス権を持つ名前空間内に Tanzu Kubernetes クラスタおよび vSphere ポッド をデプロイできます。スーパーバイザー クラスタ が vSphere ネットワーク スタックを使用して構成されている場合は、Tanzu Kubernetes クラスタをデプロイできるのはこのスタック内に限られます。

スーパーバイザー クラスタ のライセンス

スーパーバイザー クラスタ をデプロイする vSphere クラスタで [ワークロード管理] を有効にすると、60 日の評価期間の間、クラスタのすべての機能を使用できるようになります。60 日の評価期間が終了する前に、Tanzu ライセンスを スーパーバイザー クラスタ に割り当てる必要があります。

スーパーバイザー クラスタ のネットワーク スタックとして NSX-T Data Center を構成する場合は、NSX Manager に NSX-T Data Center Advanced 以降のライセンスを割り当てる必要があります。スーパーバイザー クラスタ を NSX Advanced Load Balancer を含む vSphere ネットワーク スタックで構成する場合、Tanzu ライセンス エディションに応じてロードバランサーに対し適切なライセンスが必要です。

環境が vSphere 7.0 上で実行され、スーパーバイザー クラスタ を vSphere 7.0 Update 1 以降にアップグレードする場合、アップグレードが完了するとクラスタは評価モードになります。VMware vSphere 7 Enterprise Plus と、ホストに割り当てられている Kubernetes ライセンスのアドオンの組み合わせは、通常の vSphere Enterprise 7 Plus ライセンスとして機能するため、vSphere with Tanzu 機能は有効になりません。この場合は、60 日の評価期間が終了する前に、スーパーバイザー クラスタ に Tanzu エディション ライセンスを割り当てる必要があります。

Tanzu ライセンスの有効期限

- vSphere 7.0 Update 3。vSphere 7.0 Update 3 以降では、Tanzu エディション ライセンスの有効期限が切れても、新しいライセンスが入手されるまで vSphere with Tanzu のすべての機能セットを引き続き使用できます。ただし、期限切れのライセンスを新しい スーパーバイザー クラスタ に割り当てることはできません。60 日の評価期間が終了する前に、有効な Tanzu エディション ライセンスを新しく作成された スーパーバイザー クラスタ に割り当てる必要があります。
- vSphere 7.0 Update 2 および Update 1。vSphere Update 2 または Update 1 で実行される環境で Tanzu エディション ライセンスの有効期限が切れると、vSphere 管理者は新しい名前空間を作成することも、スーパーバイザー クラスタ の Kubernetes バージョンをアップデートすることもできません。DevOps エンジニアとして、新しいワークロードをデプロイすることはできません。新しいノードの追加など、既存の Tanzu Kubernetes クラスタの構成を変更することはできません。

  引き続き Tanzu Kubernetes クラスタにワークロードをデプロイして、既存のすべてのワークロードに予期した動作を継続させることはできます。すでにデプロイされているすべての Kubernetes ワークロードは、通常の動作を継続します。

Tanzu ライセンスのコンプライアンス

Tanzu ライセンス キーには、ESXi ホスト ライセンスと同様に、CPU あたり最大 32 コアの CPU キャパシティがあります。Tanzu ライセンスを スーパーバイザー クラスタ に割り当てる場合、使用されるキャパシティの量は、クラスタからのホスト上の CPU の数と各 CPU のコア数によって決まります。Tanzu エディション ライセンス キーは一度に複数の スーパーバイザー クラスタ に割り当てることができますが、複数のライセンス キーを 1 つのクラスタに割り当てることはできません。

- vSphere 7.0 Update 3。vSphere 7.0 Update 3 以降では、たとえば、新しいホストを追加して スーパーバイザー クラスタ を拡張し、クラスタに割り当てられたライセンス キーがキャパシティ不足になった場合、同じライセンス キーを引き続き使用できます。ただし、EULA への準拠を維持するには、スーパーバイザー クラスタ のすべての CPU とコアに十分対応できるキャパシティのある新しいライセンス キーを取得する必要があります。
- vSphere 7.0 Update 2 および Update 1。vSphere with Tanzu 環境が vSphere 7.0 Update 2 および Update 1 で実行される場合、スーパーバイザー クラスタ の CPU の合計数はクラスタに割り当てられている Tanzu エディション ライセンスの CPU キャパシティの量を超えてはなりません。

評価期間の有効期限

スーパーバイザー クラスタ の評価期間が終了した場合、vSphere 管理者は新しい名前空間を作成することも、スーパーバイザー クラスタ の Kubernetes バージョンをアップデートすることもできません。DevOps エンジニアが新しいワークロードをデプロイすることはできません。また、新しいノードの追加など、既存の Tanzu Kubernetes クラスタの構成を変更することもできません。

引き続き Tanzu Kubernetes クラスタにワークロードをデプロイして、既存のすべてのワークロードに予期した動作を継続させることはできます。すでにデプロイされているすべての Kubernetes ワークロードは、通常の動作を継続します。

評価期間の有効期限における動作は、vSphere 7.0 Update 2 と Update 3 の両方で有効です。

vSphere with Tanzu アーキテクチャおよびコンポーネント

3

vSphere with Tanzu が有効なクラスタを、スーパーバイザー クラスタ と言います。クラスタは、vSphere ポッド、仮想マシン、および Tanzu Kubernetes クラスタが含まれているワークロードの実行に必要なコンポーネントとリソースを提供する vSphere with Tanzu の基盤に配置されます。

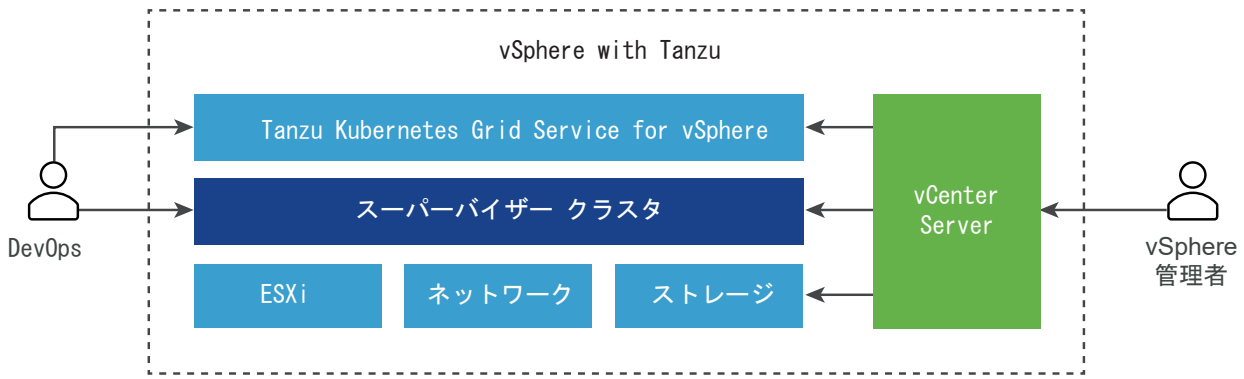
この章には、次のトピックが含まれています。

- vSphere with Tanzu アーキテクチャ
- Tanzu Kubernetes Grid サービス アーキテクチャ
- Tanzu Kubernetes クラスタのテナント モデル
- vSphere with Tanzu 認証
- vSphere with Tanzu のネットワーク
- vSphere with Tanzu セキュリティ
- vSphere with Tanzu ストレージ

vSphere with Tanzu アーキテクチャ

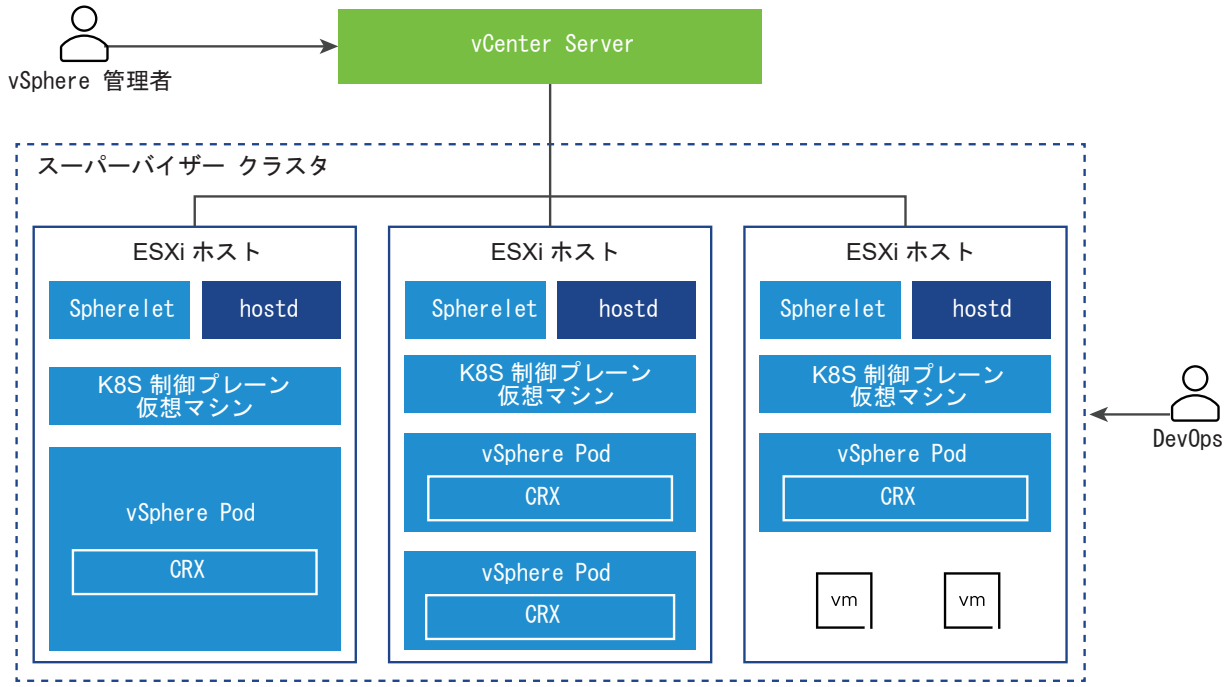
vSphere with Tanzu が vSphere クラスタで有効な場合は、ハイパーバイザー レイヤー内に Kubernetes 制御プレーンが作成されます。このレイヤーには、ESXi 内で Kubernetes ワークロードを実行する機能を有効にする特定のオブジェクトが含まれています。

図 3-1. スーパーバイザー クラスタ の一般的なアーキテクチャ



vSphere with Tanzu が有効なクラスタは、スーパーバイザー クラスタ と呼ばれます。これは、コンピューティング用の ESXi、NSX-T Data Center または vSphere ネットワーク、および vSAN または他の共有ストレージ ソリューションで構成される Software-Defined Data Center (SDDC) レイヤー上で実行されます。共有ストレージは、vSphere ポッド のパーシステント ボリューム、スーパーバイザー クラスタ 内で実行される仮想マシン、および Tanzu Kubernetes クラスタ内のポッドに使用されます。スーパーバイザー クラスタ を作成したら、vSphere 管理者は、スーパーバイザー クラスタ 内に vSphere 名前空間 と呼ばれる名前空間を作成できます。DevOps エンジニアは、vSphere ポッド 内で実行されているコンテナで構成されるワークロードを実行し、Tanzu Kubernetes クラスタを作成することができます。

図 3-2. スーパーバイザー クラスタのアーキテクチャ



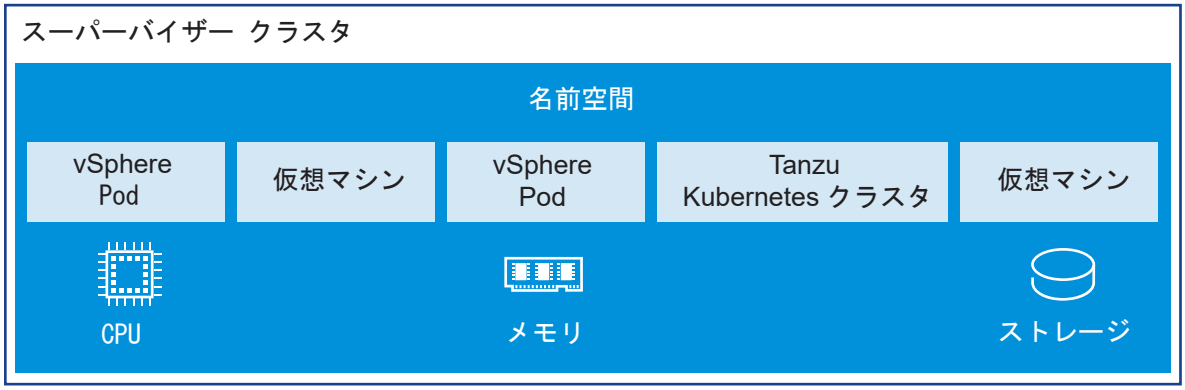
- Kubernetes 制御プレーン仮想マシン。スーパーバイザー クラスタ に含まれているホストに合計で 3 台の Kubernetes 制御プレーン仮想マシンが作成されます。3 つの制御プレーン仮想マシンにはそれぞれ独自の IP アドレスがあるため、ロード バランシングが行われます。また、仮想マシンの 1 つにフローティング IP アドレスが割り当てられます。vSphere DRS は、ESXi ホスト上の制御プレーン仮想マシンの正確な配置を決定し、必要に応じて移行します。vSphere DRS は、制御プレーン仮想マシンの Kubernetes スケジューラとも統合されているため、DRS によって vSphere ポッド の配置が決まります。DevOps エンジニアとして vSphere ポッド をスケジュール設定すると、その要求は通常の Kubernetes ワークフローを経由して DRS に送信され、そこで最終的な配置が決定されます。
- Spherelet。各ホストに Spherelet と呼ばれる追加のプロセスが作成されます。このプロセスは、ESXi に対してネイティブに移植された kubelet であり、このプロセスによって ESXi ホストは Kubernetes クラスタのメンバーになることができます。

- Container Runtime Executive (CRX)。hostd と vCenter Server の観点から見ると、CRX は仮想マシンと似ています。CRX には、ハイパーバイザーと連携する準仮想化 Linux カーネルが含まれています。CRX は仮想マシンと同じハードウェア仮想化技術を使用しており、仮想マシンの境界で囲まれています。直接起動の技法が使用されるため、CRX の Linux ゲストは、カーネルの初期化を経由することなくメインの init プロセスを開始できます。これにより、vSphere ポッド がコンテナとほぼ同じ速度で起動できるようになります。
- クラスタ API および VMware Tanzu™ Kubernetes Grid™ サービス は、スーパーバイザー クラスタ で実行されるモジュールであり、Tanzu Kubernetes クラスタのプロビジョニングと管理を可能にします。仮想マシン サービス モジュールは、スタンドアローン仮想マシンと Tanzu Kubernetes クラスタを構成する仮想マシンをデプロイして、実行します。

vSphere 名前空間

vSphere 名前空間 は、Tanzu Kubernetes Grid サービス を使用して作成された vSphere ポッド と Tanzu Kubernetes クラスタを実行できるリソースの境界を設定します。最初に作成された名前空間には、スーパーバイザー クラスタ 内の無制限のリソースがあります。vSphere 管理者は、CPU、メモリ、ストレージのほか、名前空間内で実行できる Kubernetes オブジェクトの数に制限を設定できます。vSphere 内の名前空間ごとにリソース プールが作成されます。ストレージ制限は、Kubernetes ではストレージ割り当てと表されます。

図 3-3. vSphere 名前空間



DevOps エンジニアが名前空間にアクセスできるようにするために、vSphere 管理者は、vCenter Single Sign-On に関連付けられている ID ソース内で使用可能なユーザーまたはユーザー グループに権限を割り当てます。

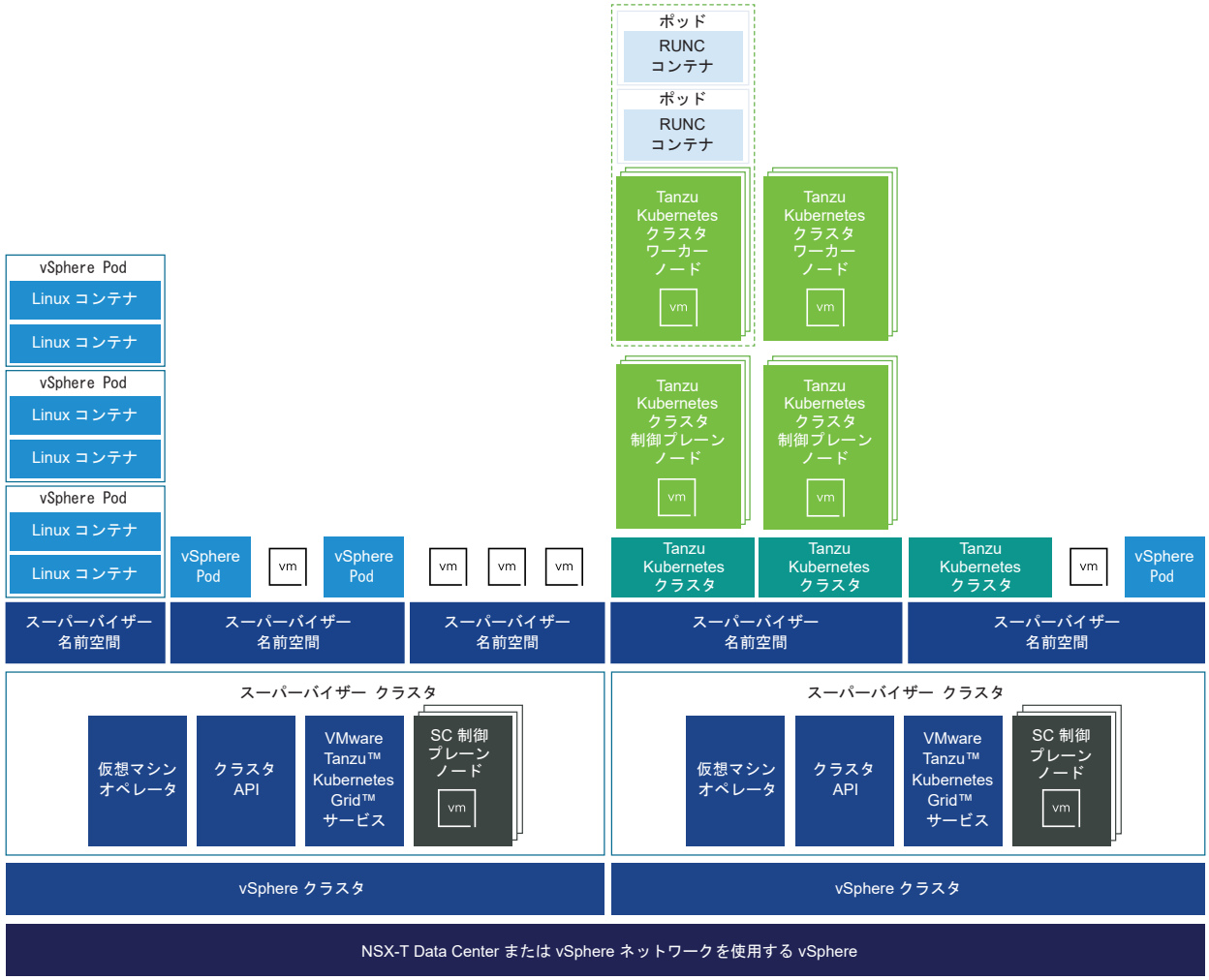
名前空間が作成され、リソースとオブジェクトの制限、権限、およびストレージ ポリシーが設定されたら、DevOps エンジニアは名前空間にアクセスして Kubernetes ワークロードを実行し、Tanzu Kubernetes Grid サービス を使用して Tanzu Kubernetes クラスタを作成することができます。

Tanzu Kubernetes クラスタ

Tanzu Kubernetes クラスタは、VMware によってパッケージ、署名、およびサポートされているオープンソースの Kubernetes ソフトウェアの完全なディストリビューションです。vSphere with Tanzu のコンテキストでは、Tanzu Kubernetes Grid サービス を使用して スーパーバイザー クラスタ 上の Tanzu Kubernetes クラスタをプロビジョニングできます。kubectl と YAML の定義を使用して、Tanzu Kubernetes Grid サービス API を宣言によって呼び出すことができます。

Tanzu Kubernetes クラスタは vSphere 名前空間 に配置されます。ユーザーは、標準の Kubernetes クラスタの場合と同じツールを使用して、同じ方法でワークロードとサービスを Tanzu Kubernetes クラスタにデプロイできます。

図 3-4. Tanzu Kubernetes クラスタの vSphere with Tanzu アーキテクチャ



vSphere ネットワーク スタックを使用して構成された スーパーバイザー クラスタ

vSphere ネットワーク スタックを使用して構成された スーパーバイザー クラスタ は、Tanzu Kubernetes Grid サービス を使用して作成された Tanzu Kubernetes クラスタの実行のみをサポートします。このクラスタでは vSphere ネットワーク サービス およびストレージ サービスもサポートされます。

vSphere ネットワーク スタックが構成された スーパーバイザー クラスタ では、vSphere ポッド はサポートされません。したがって、Spherelet コンポーネントは、Tanzu Kubernetes クラスタ内でのみ実行される スーパーバイザー クラスタ ポッドおよび Kubernetes ポッドでは使用できません。レジストリ サービスは vSphere ポッド でのみ使用されるため、vSphere ネットワーク スタックを使用して構成された スーパーバイザー クラスタ は Harbor レジストリ もサポートしません。

vSphere ネットワーク スタックを使用して構成されたクラスタで作成された vSphere 名前空間 も、vSphere ポッド の実行をサポートしません。Tanzu Kubernetes クラスタのみがサポートされます。

Tanzu Kubernetes Grid サービス アーキテクチャ

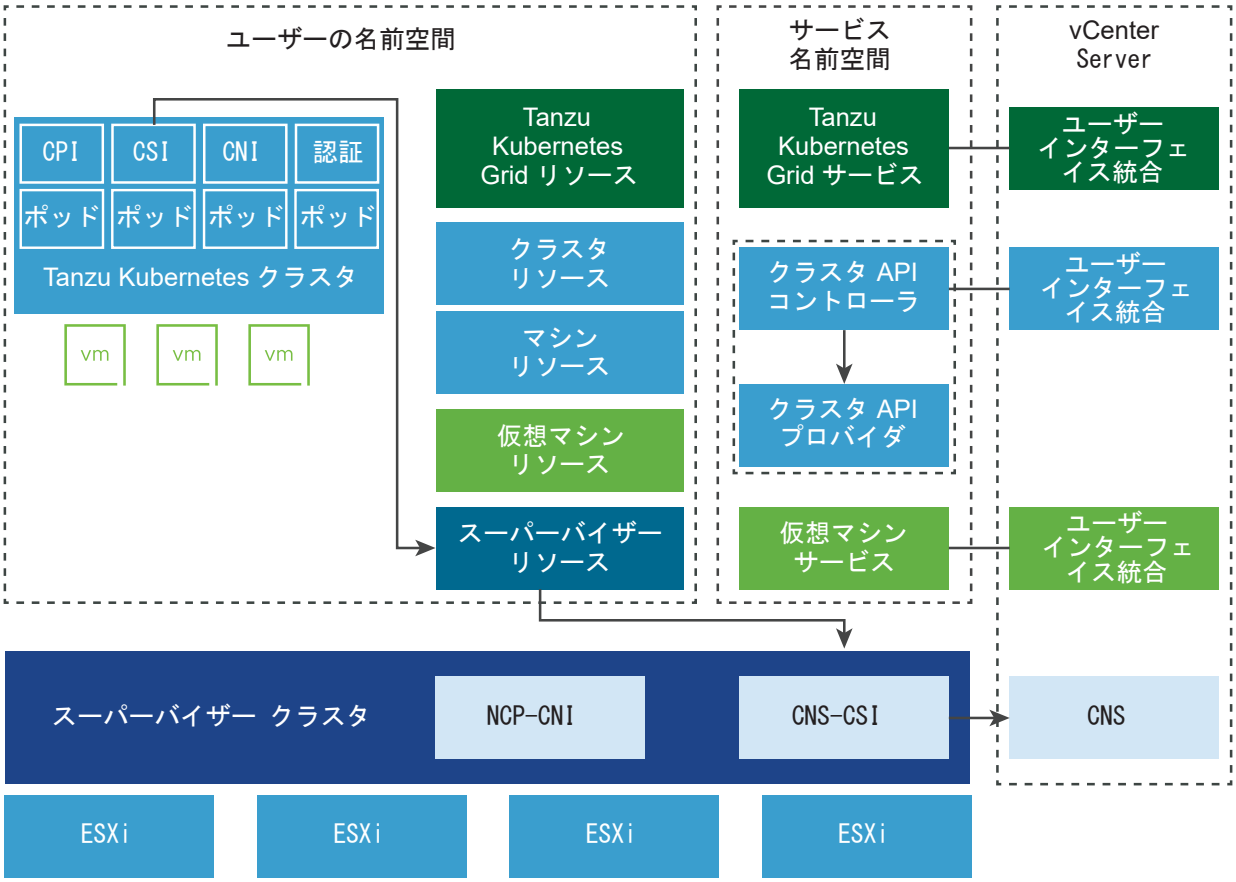
Tanzu Kubernetes Grid サービス により、Tanzu Kubernetes クラスタのライフサイクルをセルフサービスで管理できるようになります。Tanzu Kubernetes Grid サービス を使用して Tanzu Kubernetes クラスタを作成して管理する際は、Kubernetes のオペレータや開発者になじみのある、宣言による方法を使用します。

Tanzu Kubernetes Grid サービス コンポーネント

Tanzu Kubernetes Grid サービス は、Tanzu Kubernetes クラスタのライフサイクルを管理するためにコントローラの 3 つのレイヤーを公開します。

- Tanzu Kubernetes Grid サービス は、基盤となる vSphere 名前空間 リソースと統合するために必要なコンポーネントを含むクラスタをプロビジョニングします。これらのコンポーネントには、スーパーバイザー クラスタ と連携するクラウド プロバイダ プラグインが含まれています。さらに、Tanzu Kubernetes クラスタは、VMware クラウド ネイティブ ストレージ (CNS) と統合されている スーパーバイザー クラスタ にパーシステント ボリュームの要求を渡します。10 章 vSphere with Tanzu でのパーシステント ストレージの使用を参照してください。
- クラスタ API は、クラスタを作成、設定、および管理するための、宣言型の Kubernetes 形式 API を提供します。クラスタ API への入力には、クラスタを記述するリソース、クラスタを構成する仮想マシンを記述するリソースのセット、クラスタのアドオンを記述するリソースのセットなどがあります。
- 仮想マシン サービス は、仮想マシンとそれに関連する vSphere リソースを管理するための、宣言型の Kubernetes 形式 API を提供します。仮想マシン サービス により、抽象的な再利用可能ハードウェア構成を表す仮想マシン クラスの概念が導入されます。仮想マシン サービス の機能を使用すると、Tanzu Kubernetes クラスタをホストする制御プレーンおよびワーカー ノード仮想マシンのライフサイクルを管理できます。

図 3-5. Tanzu Kubernetes Grid サービス アーキテクチャおよびコンポーネント



Tanzu Kubernetes クラスタ コンポーネント

Tanzu Kubernetes クラスタで実行されるコンポーネントは、認証と認可、ストレージ統合、ポッド ネットワーク、ロード バランシングの 4 つの領域にわたっています。

- 認証 Webhook：クラスタ内のポッドとして動作し、ユーザー認証トークンを検証する Webhook。
- コンテナ ストレージ インターフェイス プラグイン:スーパーバイザー クラスタ を介して CNS と統合される準仮想化 CSI プラグイン。
- コンテナ ネットワーク インターフェイス プラグイン：ポッド ネットワークを提供する CNI プラグイン。
- クラウド プロバイダの実装：Kubernetes ロード バランサ サービスの作成をサポートします。

Tanzu Kubernetes Grid サービス API

Tanzu Kubernetes クラスタをプロビジョニングおよび管理するには、Tanzu Kubernetes Grid サービス API を使用します。これは、kubectl と YAML を使用して呼び出す宣言型 API です。

宣言型 API では、システムに対して命令型のコマンドを実行するのではなく、Tanzu Kubernetes クラスタが達成する目的の状態を指定します。具体的には、ノードの数、使用可能なストレージ、仮想マシンのサイズ、Kubernetes ソフトウェアのバージョンを指定します。Tanzu Kubernetes Grid サービス は、目的の状態を満たすクラスタをプロビジョニングするための作業を実行します。

Tanzu Kubernetes Grid サービス API を呼び出すには、YAML ファイルを使用して kubectl を呼び出します。これによって API が呼び出されます。クラスタが作成されたら、YAML を更新してクラスタを更新します。

Tanzu Kubernetes Grid サービス のインターフェイス

vSphere 管理者は、vSphere Client を使用して vSphere 名前空間 を設定し、権限を付与します。また、クラスタ コンポーネントが使用するリソースを監視し、これらのリソースの関連情報を vSphere インベントリで確認することもできます。

DevOps エンジニアは、kubectl 向けの vSphere プラグイン を使用して、vCenter Single Sign-On の認証情報によって vSphere 名前空間 に接続します。接続後、DevOps エンジニアは kubectl を使用して Tanzu Kubernetes クラスタをプロビジョニングします。

開発者は、プロビジョニングされたクラスタに、kubectl 向けの vSphere プラグイン と自分の vCenter Single Sign-On 認証情報を使用して接続できます。または、サポートされている Kubernetes 認証プロバイダをクラスタ管理者が構成している場合、開発者は kubectl を使用して接続できます。開発者が Kubernetes にワークロードをデプロイし、クラスタ環境を操作する際は、kubectl を使用します。

Tanzu Kubernetes Grid サービス のデモ

Tanzu Kubernetes Grid サービス を使用して Tanzu Kubernetes クラスタを作成および運用する方法については、ビデオ vSphere 7 with Kubernetes - Tanzu Kubernetes cluster - Technical Overview をご覧ください。

Tanzu Kubernetes クラスタのテナント モデル

スーパーバイザー クラスタ は、Tanzu Kubernetes Grid サービス によってプロビジョニングされた Tanzu Kubernetes クラスタの管理プレーンです。テナント モデルは、Tanzu Kubernetes クラスタが配置されている vSphere 名前空間 を使用して適用されます。

スーパーバイザー クラスタ

スーパーバイザー クラスタ は、Tanzu Kubernetes クラスタが構築される管理レイヤーを提供します。Tanzu Kubernetes Grid サービス は、スーパーバイザー クラスタ の一部であるコントローラのセットを備えたカスタム コントローラ マネージャです。Tanzu Kubernetes Grid サービス の目的は Tanzu Kubernetes クラスタをプロビジョニングすることです。

スーパーバイザー クラスタ と vSphere クラスタの関係は 1 対 1 ですが、スーパーバイザー クラスタ と Tanzu Kubernetes クラスタの関係は 1 対多です。1 つの スーパーバイザー クラスタ 内で複数の Tanzu Kubernetes クラスタをプロビジョニングできます。スーパーバイザー クラスタ のワークロード管理機能により、クラスタ構成とライフサイクルを簡単に制御できるとともに、アップストリーム Kubernetes での同時実行を維持できます。

詳細については、『5 章 スーパーバイザー クラスタ の構成と管理』を参照してください。

vSphere 名前空間

1 つ以上の Tanzu Kubernetes クラスタを vSphere 名前空間 にデプロイします。リソースの割り当てとストレージ ポリシーは、vSphere 名前空間 に適用され、そこにデプロイされている Tanzu Kubernetes クラスタによって継承されます。

Tanzu Kubernetes クラスタをプロビジョニングすると、vSphere 名前空間 内にリソース プールと仮想マシン フォルダが作成されます。Tanzu Kubernetes クラスタの制御プレーンとワーカー ノード仮想マシンは、このリソース プールと仮想マシン フォルダ内に配置されます。vSphere Client を使用して、この階層を表示するには、[ホストおよびクラスタ] パースペクティブを選択し、[仮想マシンおよびテンプレート] ビューも選択します。

詳細については、『7 章 vSphere 名前空間の構成と管理』を参照してください。

コンテンツ ライブラリ

vSphere コンテンツ ライブラリには、Tanzu Kubernetes クラスタノードを作成するために使用される仮想マシン テンプレートがあります。Tanzu Kubernetes クラスタをデプロイする スーパーバイザー クラスタ ごとに、Tanzu Kubernetes Grid サービス で使用される OVA のソースとなるサブスクライブ済みコンテンツ ライブラリ オブジェクトを定義して、クラスタ ノードを構築する必要があります。複数の スーパーバイザー クラスタ に同じサブスクライブ済みコンテンツ ライブラリを構成できます。サブスクライブ済みコンテンツ ライブラリと vSphere 名前空間 の間に関係はありません。サブスクライブ済みコンテンツ ライブラリによって、VMware から最新のテンプレートが直接ダウンロードされます。使用する OVA テンプレートをローカル コンテンツ ライブラリにアップロードします。

詳細については、『Tanzu Kubernetes リリース のコンテンツ ライブラリの作成と管理』を参照してください。

vSphere with Tanzu 認証

vSphere 管理者には、スーパーバイザー クラスタ を設定し、名前空間を管理するための権限が必要です。名前空間の権限を定義して、どの DevOps エンジニアが名前空間にアクセスできるかを決定します。DevOps エンジニアは、vCenter Single Sign-On の認証情報を使用して スーパーバイザー クラスタ で認証を行うと、権限が付与されている名前空間にのみアクセスできます。

vSphere 管理者の権限

vSphere 管理者には、vSphere クラスタを スーパーバイザー クラスタ として構成したり、名前空間を作成および管理したりするために vSphere クラスタに対する権限が必要です。vSphere 管理者には、vSphere クラスタ上のユーザー アカウントに関連付けられた次の権限が少なくとも 1 つ必要です。

- 名前空間構成の変更。スーパーバイザー クラスタ で名前空間を作成および設定できます。
- クラスタ全体の構成の変更。vSphere クラスタを スーパーバイザー クラスタ として設定できます。

DevOps エンジニアの権限の設定

vSphere 管理者は、名前空間レベルでユーザー アカウントに表示、編集、または所有者権限を付与します。それらのユーザー アカウントは、vCenter Single Sign-On に接続されている ID ソースに含まれている必要があります。1 つのユーザー アカウントで複数の名前空間にアクセスできます。管理者グループのメンバーであるユーザーは、スーパーバイザー クラスタ 上のすべての名前空間にアクセスできます。

権限、リソースの割り当て、およびストレージを使用して名前空間を設定したら、Kubernetes 制御プレーンの URL を DevOps エンジニアに提供します。DevOps エンジニアはその URL を使用して制御プレーンにログインできます。ログインすると、DevOps エンジニアは、vCenter Server システムに属するすべての スーパーバイザー クラスタ 内の、権限を持つすべての名前空間にアクセスできます。vCenter Server システムが拡張リンク モードに

なっている場合、DevOps エンジニアは、リンク モード グループで使用可能なすべての スーパーバイザー クラスタ 内の、権限を持つすべての名前空間にアクセスできます。Kubernetes 制御プレーンの IP アドレスは、NSX-T または Distributed Switch ネットワーク スタックで使用するロード バランサによって生成される仮想 IP アドレスで、Kubernetes 制御プレーンへのアクセス ポイントとして機能します。

所有者権限を持つ DevOps エンジニアは、ワークロードをデプロイできます。また、他の DevOps エンジニアまたはグループと名前空間を共有したり、不要になった名前空間を削除したりできます。DevOps エンジニアは名前空間を共有する場合、表示、編集、または所有権限を他の DevOps エンジニアおよびグループに割り当てることができます。

スーパーバイザー クラスタ での認証

DevOps エンジニアは、vSphere 向け Kubernetes CLI Tools により、vCenter Single Sign-On の認証情報と Kubernetes 制御プレーンの IP アドレスを使用して スーパーバイザー クラスタ で認証します。詳細については、『vCenter Single Sign-On ユーザーとして スーパーバイザー クラスタ に接続する』を参照してください。

DevOps エンジニアが スーパーバイザー クラスタ にログインすると、認証プロキシによって要求が vCenter Single Sign-On にリダイレクトされます。vSphere kubectl プラグインは、vCenter Server とのセッションを確立し、vCenter Single Sign-On から認証トークンを取得します。また、DevOps エンジニアがアクセスできる名前空間のリストを取得し、これらの名前空間を構成に取り込みます。ユーザー アカウントの権限に変更があった場合は、次回ログイン時に名前空間のリストが更新されます。

スーパーバイザー クラスタ にログインする際に使用するアカウントでアクセスできるのは、自分に割り当てられている名前空間のみです。このアカウントを使用して vCenter Server にログインすることはできません。vCenter Server にログインするには、明示的な権限が必要です。

注： kubectl へのセッションは 10 時間継続します。セッションの期限が切れると、スーパーバイザー クラスタ で再度認証する必要があります。ログアウト時にトークンはユーザー アカウントの構成ファイルから削除されますが、セッションが終了するまで有効なままです。

Tanzu Kubernetes クラスタを使用した認証

DevOps エンジニア、開発者、管理者などの Tanzu Kubernetes クラスタ ユーザーは、さまざまな方法でクラスタに対して認証できます。詳細については、『Tanzu Kubernetes クラスタを使用した認証』を参照してください。

注： Tanzu Kubernetes クラスタにポッドとリソースをデプロイするには、ユーザーおよびシステムのアカウントにポッド セキュリティ ポリシーが必要です。詳細については、『Tanzu Kubernetes クラスタでのポッド セキュリティ ポリシーの使用』を参照してください。

vSphere with Tanzu のネットワーク

スーパーバイザー クラスタ は、vSphere のネットワーク スタックまたは VMware NSX-T™ Data Center を使用して、Kubernetes 制御プレーンの仮想マシン、サービス、およびワークロードへの接続を提供できます。Tanzu Kubernetes Grid サービス によってプロビジョニングされた Tanzu Kubernetes クラスタで使用されるネットワークは、vSphere with Tanzu インフラストラクチャの基盤であるファブリックと、クラスタのポッド、サービス、および入力方向のネットワークを提供するオープンソース ソフトウェアの組み合わせです。

詳細については、4 章 vSphere with Tanzu のネットワーク

vSphere with Tanzu セキュリティ

vSphere with Tanzu は vSphere のセキュリティ機能を利用して、デフォルトで安全な Tanzu Kubernetes クラスタをプロビジョニングします。

vSphere with Tanzu は、vCenter Server および ESXi に組み込まれたセキュリティ機能を利用できる、vSphere に対するアドオン モジュールです。詳細については、vSphere Security ドキュメントを参照してください。

スーパーバイザー クラスタ は、データベース (etcd) に保存されているすべてのシークレットを暗号化します。シークレットは、起動時に vCenter Server によって提供されるローカル復号キー ファイルを介して暗号化されます。復号キーは スーパーバイザー クラスタ ノードのメモリ (tempfs) に格納されるほかに、vCenter Server データベース内のディスクに暗号化形式で格納されます。各システムの root ユーザーは、復号キーをクリア テキストで入手できます。各ワークロード クラスタのデータベース内に保持されているシークレットは、クリア テキスト形式で保存されます。すべての etcd 接続は、インストール時に生成され、アップグレード中にローテーションされる証明書を使用して認証されます。現在、証明書を手動でローテーションまたは更新することはできません。

vSphere 7.0 Update 2 以降では、AMD システムの スーパーバイザー クラスタ で機密性の高い vSphere ポッド を実行することができます。機密性の高い vSphere ポッド を作成するには、セキュリティ拡張機能として Secure Encrypted Virtualization-Encrypted State (SEV-ES) を追加します。詳細については、『機密性の確保された vSphere ポッド のデプロイ』を参照してください。

Tanzu Kubernetes クラスタはデフォルトでセキュリティ保護されています。Tanzu Kubernetes Grid サービス によってプロビジョニングされたすべての Tanzu Kubernetes クラスタで、制限付き PodSecurityPolicy (PSP) を使用できます。開発者が特権ポッドまたはルート コンテナを実行する必要がある場合、クラスタ管理者は、最低でも、デフォルトの特権 PSP へのユーザー アクセスを許可するロールバインドを作成する必要があります。詳細については、『Tanzu Kubernetes クラスタでのポッド セキュリティ ポリシーの使用』を参照してください。

Tanzu Kubernetes クラスタにはインフラストラクチャ認証情報がありません。Tanzu Kubernetes クラスタ内に保存される認証情報では、Tanzu Kubernetes クラスタにテナントのある vSphere 名前空間 にのみアクセスが可能です。そのため、クラスタ オペレータまたはユーザーの権限のエスカレーションが行われることはありません。

Tanzu Kubernetes クラスタへのアクセスに使用される認証トークンは、スーパーバイザー クラスタ へのアクセスに使用できないように範囲が設定されます。これにより、クラスタ オペレータ、またはクラスタを侵害する可能性があるユーザーは、Tanzu Kubernetes クラスタにログインするときに、root レベルのアクセス権を使用して vSphere 管理者のトークンをキャプチャできなくなります。

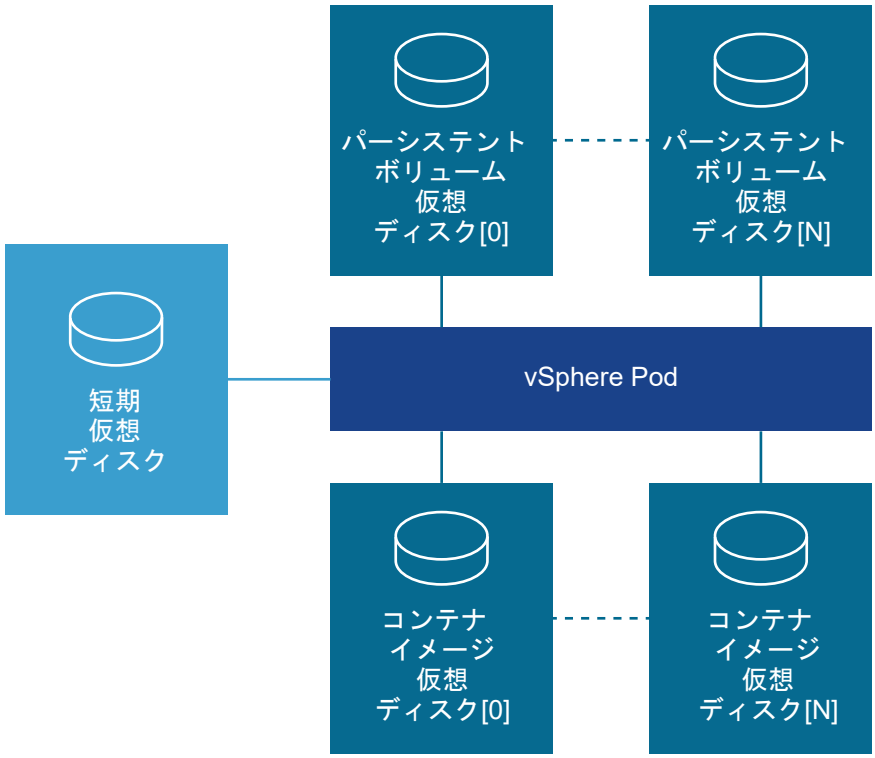
vSphere with Tanzu ストレージ

vSphere with Tanzu は、ストレージ ポリシーを使用して、VMFS、NFS、vSAN、vVols データストアなど、環境内で使用可能な共有データストアと統合します。このポリシーは、制御プレーン仮想マシン、ポッドの短期ディスク、コンテナ イメージ、パーシステント ストレージ ボリュームなどのオブジェクトのストレージ配置を制御します。Tanzu Kubernetes クラスタを使用する場合は、ストレージ ポリシーによって、Tanzu Kubernetes クラスタ ノードのデプロイ方法も決定されます。

vSphere with Tanzu を有効にする前に、スーパーバイザー クラスタ および名前空間で使用するストレージ ポリシーを作成します。

vSphere ストレージ環境と DevOps のニーズに応じて、複数のストレージ ポリシーを作成してさまざまなストレージ クラスを表すことができます。

たとえば、vSphere ポッド に 3 つのタイプの仮想ディスクがすべてマウントされていて、vSphere ストレージ環境にブロンズ、シルバー、ゴールドの 3 つのクラスのデータストアがある場合、すべてのデータストアに対してストレージ ポリシーを作成できます。その後、短期仮想ディスクとコンテナ イメージ仮想ディスクにブロンズ データストアを使用し、パーシステント ボリューム仮想ディスクにシルバーおよびゴールド データストアを使用することができます。



ストレージ ポリシーの一般的な詳細については、vSphere のストレージドキュメントの「ストレージ ポリシー ベースの管理」の章を参照してください。ストレージ ポリシーの作成の詳細については、vSphere with Tanzu のストレージ ポリシーの作成 を参照してください。

短期仮想ディスク

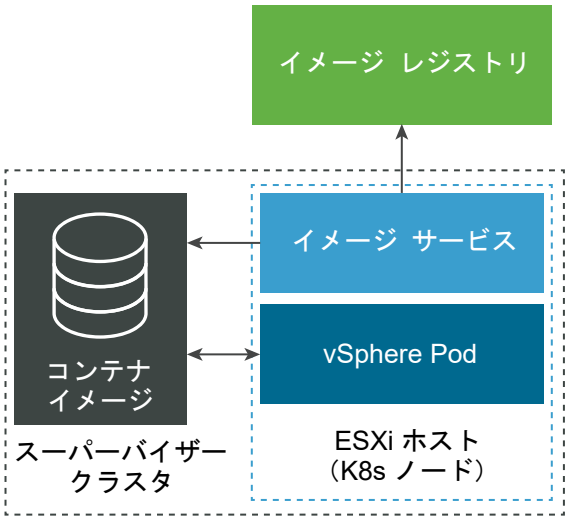
vSphere ポッド および Tanzu Kubernetes クラスタで実行されるポッドには、操作中にログ、`emptyDir` ボリューム、および `ConfigMaps` などの Kubernetes オブジェクトを保存するための短期ストレージが必要です。この短期（一時）ストレージは、ポッドが存続する限り保持されます。短期データはコンテナの再起動後も維持されますが、ポッドのライフタイムが終了すると、短期仮想ディスクは破棄されます。

各ポッドには 1 つの短期仮想ディスクがあります。vSphere 管理者は、スーパーバイザー クラスタ のストレージを設定するときに、ストレージ ポリシーを使用してすべての短期仮想ディスクのデータストアの場所を定義します。

コンテナ イメージ仮想ディスク

ポッド内のコンテナでは、実行するソフトウェアを含むイメージを使用します。ポッドは、コンテナで使用されるイメージをイメージ仮想ディスクとしてマウントします。ポッドのライフサイクルが完了すると、イメージ仮想ディスクはポッドから接続を解除されます。

ESXi コンポーネントのイメージ サービスは、イメージ レジストリからコンテナ イメージをプルし、そのイメージをポッド内で実行される仮想ディスクに変換します。



ESXi は、ポッドで実行されているコンテナ用にダウンロードされたイメージをキャッシュできます。これ以降、同じイメージを使用するポッドは、外部コンテナ レジストリではなくローカル キャッシュからプルします。

短期ストレージの場合と同様に、vSphere 管理者はイメージ キャッシュのデータストアの場所を スーパーバイザー クラスタ レベルで指定します。5 章 スーパーバイザー クラスタ の構成と管理とスーパーバイザー クラスタ のストレージ設定の変更を参照してください。

コンテナ イメージの操作については、15 章 vSphere with Tanzu ワークロードに対するコンテナ レジストリの使用を参照してください。

パーシステント ストレージ仮想ディスク

特定の Kubernetes ワークロードでデータを永続的に保存するには、パーシステント ストレージが必要です。Kubernetes ワークロード用にパーシステント ストレージをプロビジョニングするため、vSphere with Tanzu は、パーシステント ボリュームを管理する vCenter Server コンポーネントであるクラウド ネイティブ ストレージ (CNS) と統合します。

パーシステント ストレージは、vSphere ポッド、Tanzu Kubernetes クラスタ、仮想マシンで使用できます。DevOps チームがパーシステント ストレージを使用できるようにするために、vSphere 管理者は、さまざまなストレージ要件とサービス クラスを記述する仮想マシン ストレージ ポリシーを作成します。その後、ストレージ ポリシーを vSphere 名前空間 に割り当てることができます。vSphere 名前空間 の作成と設定と名前空間のストレージ設定の変更を参照してください。

スーパーバイザー クラスタ および Tanzu Kubernetes クラスタでのパーシステント ストレージの使用方法の詳細については、10 章 vSphere with Tanzu でのパーシステント ストレージの使用と 13 章 TKGS クラスタのプロビジョニングと操作を参照してください。

vSphere with Tanzu のネットワーク

4

スーパーバイザー クラスタ は、vSphere のネットワーク スタックまたは VMware NSX-T™ Data Center を使用して、Kubernetes 制御プレーンの仮想マシン、サービス、およびワークロードへの接続を提供できます。Tanzu Kubernetes Grid サービス によってプロビジョニングされた Tanzu Kubernetes クラスタで使用されるネットワークは、vSphere with Tanzu インフラストラクチャの基盤であるファブリックと、クラスタのポッド、サービス、および入力方向のネットワークを提供するオープンソース ソフトウェアの組み合わせです。

この章には、次のトピックが含まれています。

スーパーバイザー クラスタ ネットワーク

vSphere with Tanzu 環境の場合、スーパーバイザー クラスタ は vSphere のネットワーク スタックまたは VMware NSX-T Data Center™ を使用して、Kubernetes 制御プレーンの仮想マシン、サービス、およびワークロードへの接続を提供できます。スーパーバイザー クラスタ に vSphere ネットワーク スタックが構成されている場合、クラスタのすべてのホストは、Kubernetes ワークロードと制御プレーン仮想マシンへの接続を提供する vSphere Distributed Switch に接続されます。vSphere ネットワーク スタックを使用する スーパーバイザー クラスタ には、DevOps ユーザーおよび外部サービスへの接続を提供するために vCenter Server 管理ネットワーク上にロード バランサが必要です。VMware NSX-T Data Center™ を使用して構成される スーパーバイザー クラスタ は、ソリューションのソフトウェアベースのネットワークと NSX Edge ロード バランサを使用して、外部サービスと DevOps ユーザーへの接続を提供します。

NSX-T Data Center を使用する スーパーバイザー クラスタ ネットワーク

VMware NSX-T Data Center™ は、スーパーバイザー クラスタ 内のオブジェクトおよび外部ネットワークへのネットワーク接続を提供します。クラスタを構成する ESXi ホストへの接続は、標準の vSphere ネットワークによって処理されます。

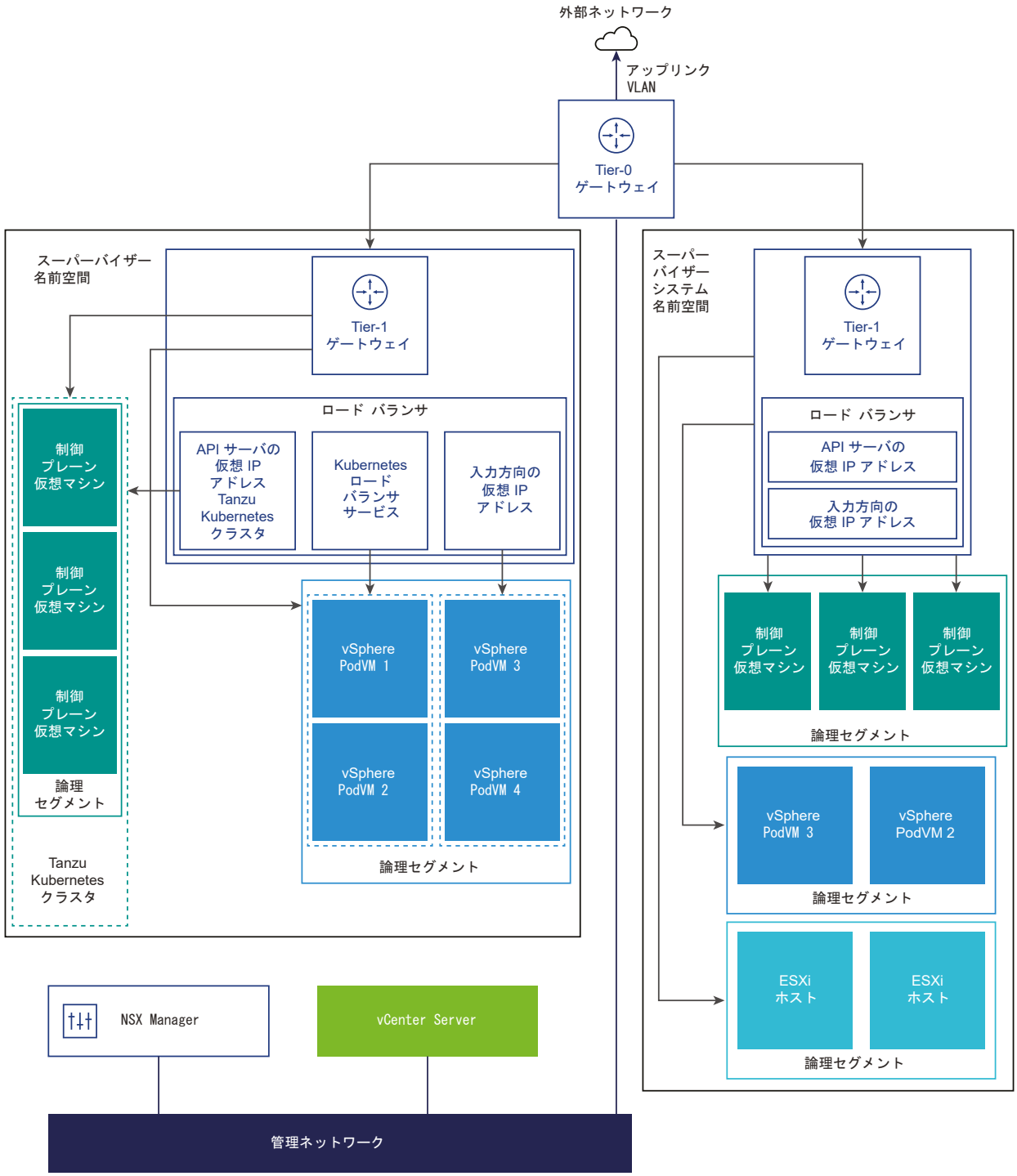
既存の NSX-T Data Center デプロイを使用するか、NSX-T Data Center の新しいインスタンスをデプロイすることによって、スーパーバイザー クラスタ ネットワークを手動で構成することもできます。

次の表に、サポートされている NSX-T Data Center のバージョンを示します。

vSphere with Tanzu	NSX-T Data Center
Version 7.0 Update 3	バージョン 3.0、3.0.x、3.1、3.1.1、3.1.2、および 3.1.3
Version 7.0 Update 2	バージョン 3.0、3.0.x、3.1、3.1.1、および 3.1.2
Version 7.0 Update 1c	バージョン 3.0、3.0.x、3.1、および 3.1.1
Version 7.0 Update 1	バージョン 3.0、3.0.1、3.0.1.1、および 3.0.2
バージョン 7.0	バージョン 3.0

このセクションでは、vSphere with Tanzu バージョン 7.0 Update 2 をインストールして構成する場合のネットワーク トポロジについて説明します。vSphere with Tanzu バージョン 7.0 Update 1 からバージョン 7.0 Update 2 にアップグレードする場合のアップグレード手順については、ネットワーク トポロジのアップグレードを参照してください。

図 4-1. スーパーバイザー クラスタ ネットワーク



- NSX Container Plug-in (NCP) は NSX-T Data Center と Kubernetes を統合します。NCP のメインコンポーネントはコンテナで実行され、NSX Manager および Kubernetes 制御プレーンと通信します。NCP は、コンテナおよびその他のリソースへの変更を監視し、NSX API を呼び出して、コンテナの論理ポート、セグメント、ルーター、セキュリティ グループなどのネットワーク リソースを管理します。

NCP はデフォルトで、システム名前空間用の共有 Tier-1 ゲートウェイを 1 つ作成し、名前空間ごとに Tier-1 ゲートウェイとロード バランサを 1 つずつ作成します。Tier-1 ゲートウェイは、Tier-0 ゲートウェイとデフォルトのセグメントに接続されています。

システム名前空間は、スーパーバイザー クラスタと Tanzu Kubernetes の機能に不可欠な主要コンポーネントで使用される名前空間です。Tier-1 ゲートウェイ、ロード バランサ、SNAT IP を含む共有ネットワーク リソースは、システム名前空間内でグループ化されます。

- NSX Edge は、外部ネットワークから スーパーバイザー クラスタ オブジェクトへの接続を提供します。NSX Edge クラスタには、制御プレーン仮想マシン上にある Kubernetes API サーバの冗長性を確保するロード バランサや、スーパーバイザー クラスタ の外部から公開およびアクセスする必要があるアプリケーションがあります。
- NSX Edge クラスタには Tier-0 ゲートウェイが関連付けられており、外部ネットワークへのルーティングを提供します。アップリンク インターフェイスでは、動的ルーティング プロトコルの BGP またはスタティック ルーティングのいずれかが使用されます。
- 各 vSphere 名前空間 には、個別のネットワークのほかに、Tier-1 ゲートウェイ、ロード バランサ サービス、SNAT IP アドレスなど、名前空間内のアプリケーションで共有される一連のネットワーク リソースが含まれています。
- 同じ名前空間内にある vSphere ポッド、通常の仮想マシン、または Tanzu Kubernetes クラスタで実行されるワークロードは、North-South 接続に対して同じ SNAT IP アドレスを共有します。
- vSphere ポッドまたは Tanzu Kubernetes クラスタで実行されるワークロードには、デフォルトのファイアウォールによって実装される共通の隔離ルールが適用されます。
- Kubernetes 名前空間ごとに個別の SNAT IP アドレスが必要になることはありません。名前空間の間の East-West 接続は、SNAT ではありません。
- 各名前空間のセグメントは、NSX Edge クラスタに関連付けられている、標準モードで機能する vSphere Distributed Switch (VDS) に配置されます。このセグメントは、スーパーバイザー クラスタ にオーバーレイ ネットワークを提供します。
- スーパーバイザー クラスタの共有 Tier-1 ゲートウェイ内に、個別のセグメントがあります。各 Tanzu Kubernetes クラスタのセグメントは、名前空間の Tier-1 ゲートウェイ内で定義されています。
- 各 ESXi ホストの Spherelet プロセスは、管理ネットワーク上のインターフェイスを介して vCenter Server と通信します。

スーパーバイザー クラスタ のネットワークの詳細については、ビデオ vSphere 7 with Kubernetes Network Service - Part 1 - The Supervisor Cluster を参照してください。

NSX-T Data Center を使用するネットワークの構成方法

スーパーバイザー クラスタ は、固定型ネットワーク構成を使用します。NSX-T Data Center を使用する スーパーバイザー クラスタ ネットワークを構成するには次の 2 つの方法があり、いずれの場合でも同じネットワーク モデルがデプロイされます。

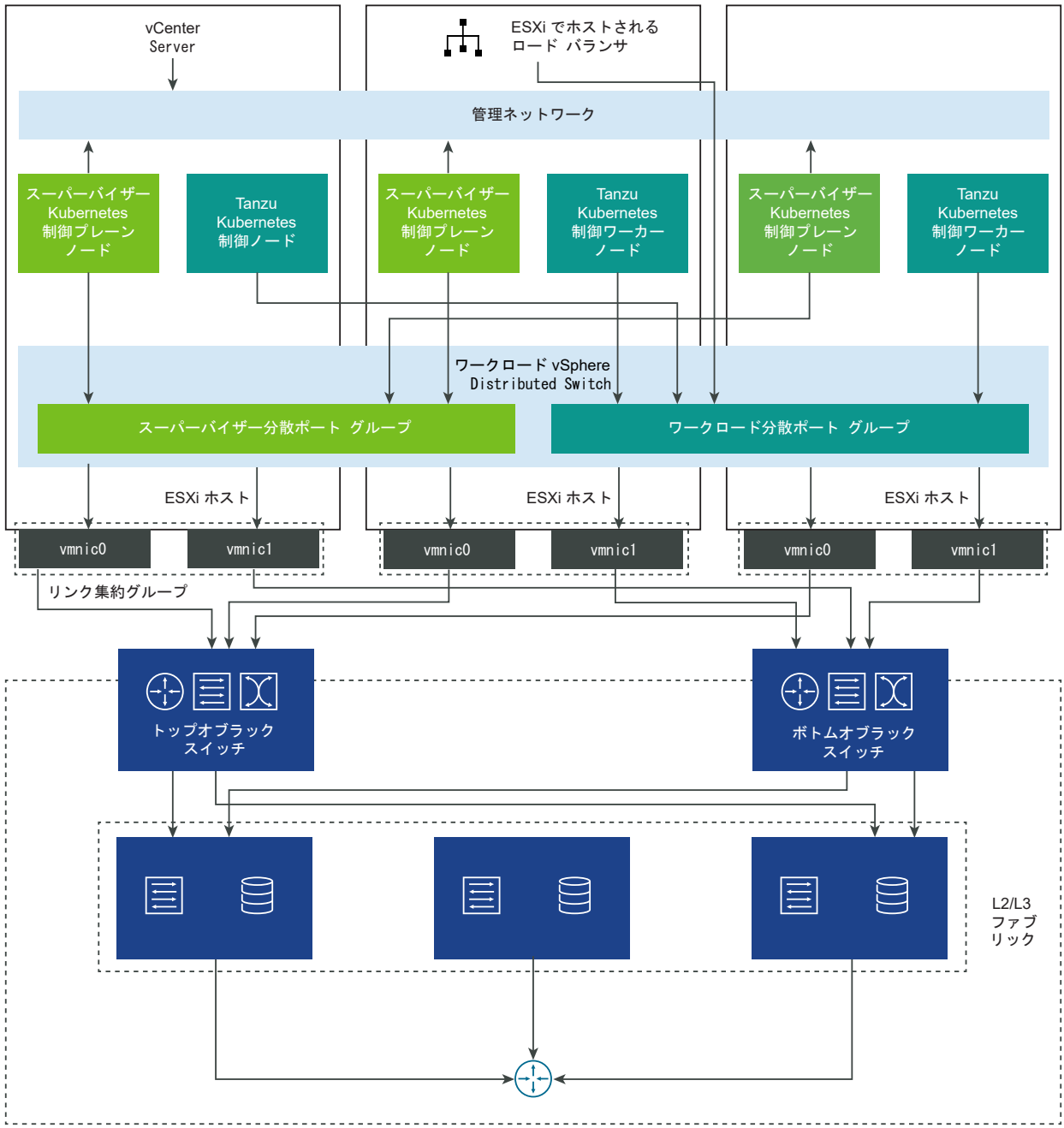
- スーパーバイザー クラスタ ネットワークを構成する最も簡単な方法は、VMware Cloud Foundation SDDC Manager を使用することです。詳細については、VMware Cloud Foundation SDDC Manager のドキュメントを参照してください。詳細については、ワークロード管理の操作を参照してください。

- 既存の NSX-T Data Center デプロイを使用するか、NSX-T Data Center の新しいインスタンスをデプロイすることによって、スーパーバイザー クラスタ ネットワークを手動で構成することもできます。詳細については vSphere with Tanzu で使用する NSX-T Data Center のインストールと構成を参照してください。

vSphere Distributed Switch を使用した スーパーバイザー クラスタ ネットワーク

vSphere Distributed Switch によってバッキングされる スーパーバイザー クラスタ は、名前空間のワークロード ネットワークとして分散ポート グループを使用します。

図 4-2. vSphere Distributed Switch を使用した名前空間ネットワーク



スーパーバイザー クラスタ に実装するトポロジによっては、1 つ以上の分散ポート グループをワークロード ネットワークとして使用できます。Kubernetes 制御プレーン仮想マシンへの接続を提供するネットワークは、プライマリ ワークロード ネットワークと呼ばれます。スーパーバイザー クラスタ 上のすべての名前空間にこのネットワークを割り当てることも、名前空間ごとに異なるネットワークを使用することもできます。クラスタが配置されている名前空間に割り当てられたワークロード ネットワークに、Tanzu Kubernetes クラスタが接続されます。

vSphere Distributed Switch によってバッキングされる スーパーバイザー クラスタ は、DevOps ユーザーと外部サービスへの接続を提供するためにロード バランサを使用します。NSX Advanced Load Balancer または HAProxy ロード バランサを使用できます。

詳細については、『vSphere ネットワークと vSphere with Tanzu 用 NSX Advanced Load Balancer の構成』および『HAProxy ロード バランサのインストールと構成』を参照してください。

Tanzu Kubernetes Grid サービス クラスタ ネットワーク

Tanzu Kubernetes Grid サービス によってプロビジョニングされた Tanzu Kubernetes Grid サービス クラスタでは、Antrea（デフォルト）と Calico の 2 つの CNI オプションがサポートされています。いずれも、クラスタのポッド、サービス、および入力用のネットワークを提供するオープンソース ソフトウェアです。

Tanzu Kubernetes Grid サービス によってプロビジョニングされた Tanzu Kubernetes Grid サービス クラスタでは、次の コンテナ ネットワーク インターフェイス (CNI) オプションがサポートされます。

- Antrea
- Calico

Antrea は、新しい Tanzu Kubernetes Grid サービス クラスタのデフォルトの CNI です。Antrea を使用する場合、クラスタのプロビジョニング中に CNI として指定する必要はありません。Calico を CNI として使用するには、次の 2 つの方法があります。

- クラスタの YAML で直接 CNI を指定します。Tanzu Kubernetes Grid サービス v1alpha1 API を使用した Tanzu Kubernetes クラスタのプロビジョニングの例を参照してください。
- デフォルトの CNI を変更します。Tanzu Kubernetes Grid サービス v1alpha1 API の構成の例を参照してください。

注： Antrea をデフォルトの CNI として使用するには、Tanzu Kubernetes Grid サービス クラスタ用の OVA ファイルの最小バージョンが必要です。更新のための Tanzu Kubernetes クラスタ互換性の確認を参照してください。

次の表に、Tanzu Kubernetes Grid サービス クラスタのネットワーク機能とその実装の概要を示します。

表 4-1. Tanzu Kubernetes Grid サービス クラスタ ネットワーク

エンドポイント	プロバイダ	説明
ポッドの接続	Antrea または Calico	ポッドのコンテナ ネットワーク インターフェイス。Antrea は Open vSwitch を使用します。Calico は BGP を利用する Linux ブリッジを使用します。
サービス タイプ：ClusterIP	Antrea または Calico	クラスタ内からのみアクセス可能なデフォルトの Kubernetes サービス タイプ。

表 4-1. Tanzu Kubernetes Grid サービス クラスタ ネットワーク （続き）

エンドポイント	プロバイダ	説明
サービス タイプ：NodePort	Antrea または Calico	Kubernetes ネットワーク プロキシによって各ワーカー ノードで開かれているポートを介して外部からアクセスできるようにします。
サービス タイプ：LoadBalancer	NSX-T ロード バランサ、NSX Advanced Load Balancer、HAProxy	NSX-T では、サービス タイプの定義ごとに 1 台の仮想サーバ。NSX Advanced Load Balancer については、このドキュメントの該当するセクションを参照してください。 **注：** 固定 IP アドレスのサポートなどの一部のロード バランシング機能は、HAProxy で使用できない場合があります。
クラスタの入力方向	サードパーティ製の入力方向コントローラ	受信ポッド トラフィックのルーティング。Contour などの任意のサードパーティ製入力方向コントローラを使用できます。
ネットワーク ポリシー	Antrea または Calico	選択したポッドとネットワーク エンドポイントの間で送受信されるトラフィックを制御します。Antrea は Open vSwitch を使用します。Calico は Linux IP テーブルを使用します。

vSphere with Tanzu 用 NSX-T Data Center の構成

vSphere with Tanzu では、スーパーバイザー クラスタ、vSphere 名前空間、名前空間内で実行されるすべてのオブジェクト（vSphere ポッド、仮想マシン、Tanzu Kubernetes クラスタなど）への接続を有効にするために、特定のネットワーク構成が必要です。vSphere 管理者は、vSphere with Tanzu で使用する NSX-T Data Center をインストールして構成します。

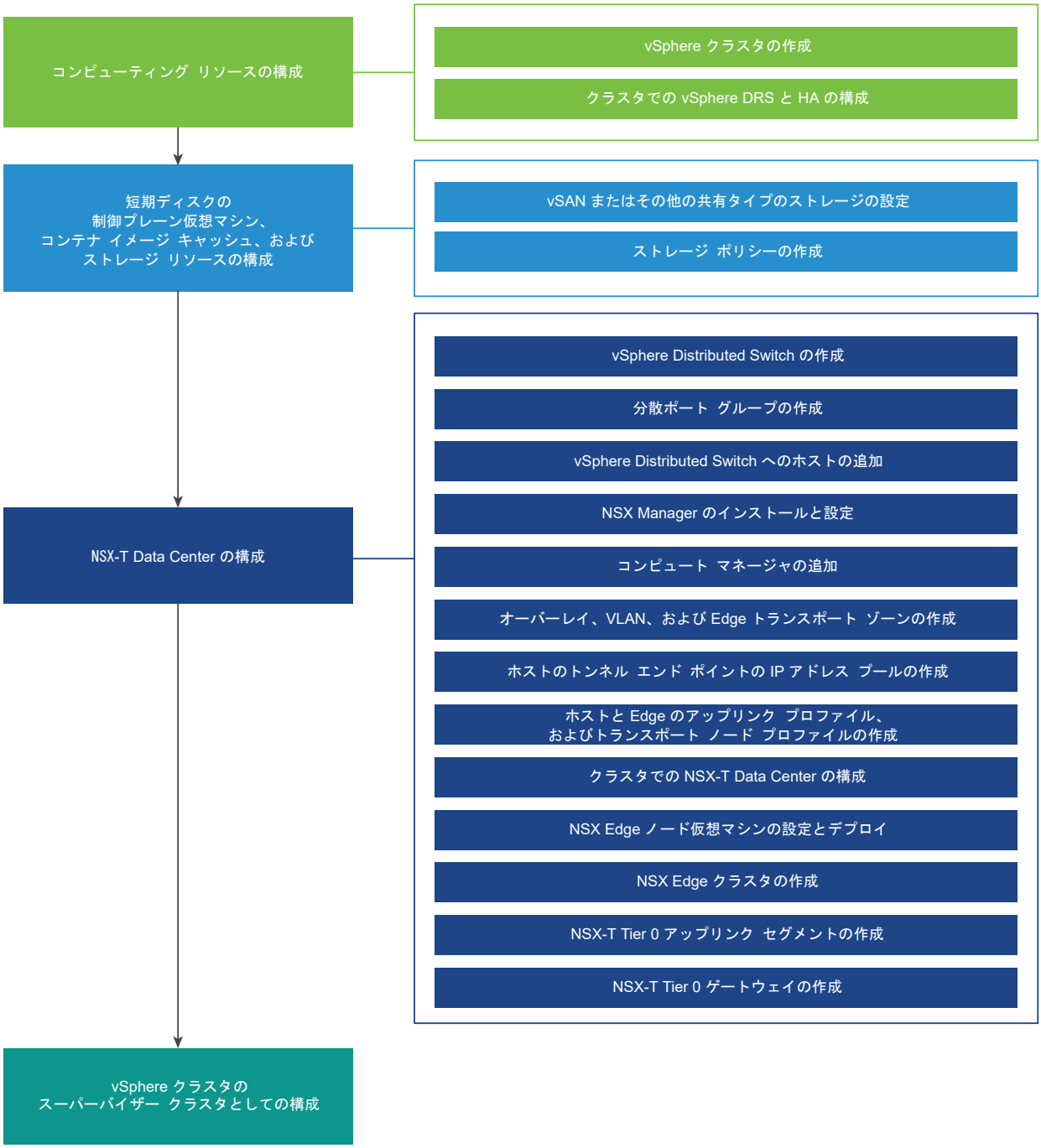
スーパーバイザー クラスタ は、固定型ネットワーク構成を使用します。同じネットワーク モデルをデプロイする スーパーバイザー クラスタ ネットワークを構成するには、次の 2 つの方法があります。

- スーパーバイザー クラスタ ネットワークを構成する最も簡単な方法は、VMware Cloud Foundation SDDC Manager を使用することです。詳細については、VMware Cloud Foundation SDDC Manager のドキュメントを参照してください。詳細については、ワークロード管理の操作を参照してください。
- 既存の NSX-T Data Center デプロイを使用するか、NSX-T Data Center の新しいインスタンスをデプロイすることによって、スーパーバイザー クラスタ ネットワークを手動で構成することもできます。

NSX-T Data Center を含む スーパーバイザー クラスタ のワークフロー

vSphere 管理者は、vSphere クラスタを、vSphere ネットワーク スタックを使用する スーパーバイザー クラスタ として構成できます。

図 4-3. NSX-T Data Center ネットワークを使用する スーパーバイザー クラスタ のワークフロー



手順

1 NSX-T Data Center を使用して vSphere with Tanzu をセットアップするためのシステム要件

NSX-T Data Center ネットワーク スタックを使用して、vSphere クラスタに vSphere with Tanzu を構成するためのシステム要件を確認します。

2 NSX-T Data Center を使用した スーパーバイザー クラスタ のトポロジ

Kubernetes ワークロードのニーズや基盤となるネットワーク インフラストラクチャに応じて、クラスタにさまざまなトポロジを適用できます。

3 スーパーバイザー クラスタ で NSX-T Data Center を構成する際のベスト プラクティスに関する考慮事項

NSX-T Data Center を使用して vSphere クラスタを スーパーバイザー クラスタ として構成する場合は、次に示すベスト プラクティスを考慮してください。

4 vSphere with Tanzu で使用する NSX-T Data Center のインストールと構成

vSphere with Tanzu では、スーパーバイザー クラスタ、vSphere 名前空間、名前空間内で実行されるすべてのオブジェクト（vSphere ポッド、仮想マシン、Tanzu Kubernetes クラスタなど）への接続を有効にするために、特定のネットワーク構成が必要です。vSphere 管理者は、vSphere with Tanzu で使用する NSX-T Data Center をインストールして構成します。

NSX-T Data Center を使用して vSphere with Tanzu をセットアップするためのシステム要件

NSX-T Data Center ネットワーク スタックを使用して、vSphere クラスタに vSphere with Tanzu を構成するためのシステム要件を確認します。

vSphere with Tanzu の構成の上限

VMware は、VMware 構成の上限ツールで構成上の制限についての情報を提供しています。

スーパーバイザー クラスタ や Tanzu Kubernetes クラスタなどの vSphere with Tanzu に固有の構成制限については、[vSphere] - [vSphere 7.0] - [vSphere with Kubernetes] - [VMware Tanzu Kubernetes Grid Service for vSphere] の順に選択し、[制限の表示] をクリックするか、このリンクを参照してください。

管理、Edge、およびワークロード ドメイン クラスタの要件

管理、Edge、およびワークロード管理の機能を 1 つの vSphere クラスタに統合した vSphere with Tanzu をデプロイできます。

表 4-2. 管理、Edge、およびワークロード管理クラスタの最小限のコンピューティング要件

システム	最小デプロイ サイズ	CPU	メモリ	ストレージ
vCenter Server 7.0	小	2	16 GB	290 GB
ESXi ホスト 7.0	3 台の ESXi ホストと、ホストあたり 1 つの固定 IP アドレス。 vSAN を使用している場合は、それぞれに物理 NIC を 2 つ以上搭載した ESXi ホストが 3 台以上必要です。ただし、パッチ適用時やアップグレード時は、回復性のために、ESXi ホストを 4 台使用することを推奨します。 ホストは、vSphere DRS と HA が有効になっているクラスタに参加している必要があります。vSphere DRS は、完全自動化モードまたは一部自動化モードになっている必要があります。 **注意：** スーパーバイザー クラスタ を構成した後に vSphere DRS を無効にしないでください。DRS を常に有効にすることは、スーパーバイザー クラスタ でワークロードを実行するための必須の前提条件です。Tanzu Kubernetes DRS を無効にすると、クラスタが破損します。	8	ホストあたり 64 GB	該当なし
NSX Manager	中	6	24 GB	300 GB
NSX Edge 1	大	8	32 GB	200 GB
NSX Edge 2	大	8	32 GB	200 GB
Kubernetes 制御プレーンの仮想マシン	3	4	16 GB	16 GB

管理および Edge クラスタと、ワークロード管理クラスタを分離したトポロジ

vSphere with Tanzu は、2 つのクラスタにデプロイできます。1 つは管理機能と Edge 機能用、もう 1 つはワークロード管理専用です。

表 4-3. 管理および Edge クラスタの最小限のコンピューティング要件

システム	最小デプロイ サイズ	CPU	メモリ	ストレージ
vCenter Server 7.0	小	2	16 GB	290 GB
ESXi ホスト 7.0	2 台の ESXi ホスト	8	ホストあたり 64 GB	該当なし
NSX Manager	中	6	24 GB	300 GB
NSX Edge 1	大	8	32 GB	200 GB
NSX Edge 2	大	8	32 GB	200 GB

表 4-4. ワークロード管理クラスタの最小限のコンピューティング要件

システム	最小デプロイ サイズ	CPU	メモリ	ストレージ
ESXi ホスト 7.0	3 台の ESXi ホストと、ホストあたり 1 つの固定 IP アドレス。 vSAN を使用している場合は、ホストあたりの物理 NIC が 2 つ以上である ESXi ホストが最低でも 3 台必要です。ただし、パッチ適用やアップグレード時の回復性のために、4 台の ESXi ホストが推奨されます。 ホストは、vSphere DRS と HA が有効になっているクラスタに参加している必要があります。vSphere DRS は、完全自動化モードになっている必要があります。 **注意：** スーパーバイザー クラスタ を構成した後に vSphere DRS を無効にしないでください。DRS を常に有効にすることは、スーパーバイザー クラスタ でワークロードを実行するための必須の前提条件です。Tanzu Kubernetes DRS を無効にすると、クラスタが破損します。	8	ホストあたり 64 GB	該当なし
Kubernetes 制御プレーンの仮想マシン	3	4	16 GB	16 GB

ネットワーク要件

vSphere の Kubernetes ワークロード管理用に実装するトポロジに関係なく、以下の表のネットワーク要件に従ってデプロイしてください。

注： vSphere 7 スーパーバイザー クラスタ による IPv6 クラスタの作成や、Tanzu Mission Control による IPv6 クラスタの登録はできません。

コンポーネント	最小数	必要な構成
Kubernetes 制御プレーン仮想マシンの固定 IP アドレス	5 つのアドレスのブロック	スーパーバイザー クラスタ 内の Kubernetes 制御プレーン仮想マシンに割り当てられる、連続する 5 つの固定 IP アドレスのブロック。
管理トラフィック ネットワーク	1	ESXi ホスト、vCenter Server、および DHCP サーバにルーティング可能な管理ネットワーク。コンテナ レジストリが外部ネットワーク上にある場合、ネットワークはコンテナ レジストリにアクセス可能で、インターネットに接続できる必要があります。コンテナ レジストリは DNS 経由で解決できる必要があり、以下で説明する出力方向設定でアクセスできる必要があります。
NTP サーバおよび DNS サーバ	1	vCenter Server に使用できる DNS サーバおよび NTP サーバ。 **注：** すべての ESXi ホスト、vCenter Server システム、および NSX Manager インスタンスで NTP を構成します。
DHCP サーバ	1	オプション。管理用の IP アドレスを自動的に取得するように DHCP サーバを構成します。DHCP サーバはクライアント識別子をサポートし、互換性のある DNS サーバ、DNS 検索ドメイン、および NTP サーバを提供する必要があります。
イメージ レジストリ	1	サービスのレジストリへのアクセス。
管理ネットワークのサブネット	1	ESXi ホストと、vCenter Server、NSX アプライアンス、および Kubernetes 制御プレーンとの間の管理トラフィックに使用されるサブネット。サブネットのサイズは次のようにする必要があります。 ■ ホストの VMkernel アダプタごとに 1 つの IP アドレス。 ■ vCenter Server アプライアンスに 1 つの IP アドレス。 ■ NSX Manager に 1 つまたは 4 つの IP アドレス。3 台のノードと 1 つの仮想 IP アドレス (VIP) の NSX Manager クラスタリングを実行する場合は 4 つです。 ■ Kubernetes 制御プレーンに 5 つの IP アドレス。3 台のノードそれぞれに 1 つずつ、仮想 IP アドレス用に 1 つ、クラスタのローリング アップグレード用に 1 つ。 **注：** 管理ネットワークとワークロード ネットワークは異なるサブネット上に配置する必要があります。管理ネットワークとワークロード ネットワークに同じサブネットを割り当てることはできないため、システム エラーや問題が発生することがあります。
管理ネットワークの VLAN	1	管理ネットワークのサブネットの VLAN ID。

コンポーネント	最小数	必要な構成
VLAN	3	VLAN IP アドレスはトンネル エンドポイント (TEP) の IP アドレスです。ESXi ホスト TEP と Edge TEP はルーティング可能である必要があります。 VLAN IP アドレスが必要となるものは次のとおりです。 ■ ESXi ホスト VTEP ■ 固定 IP アドレスを使用する Edge VTEP ■ トランスポート ノードの Tier-0 ゲートウェイとアップリンク。 **注：** ESXi ホスト VTEP および Edge VTEP では、MTU サイズを 1,600 よりも大きくする必要があります。 ESXi ホストおよび NSX-T Edge ノードはトンネル エンドポイントとして機能し、各ホストおよび Edge ノードにトンネル エンドポイント (TEP) IP アドレスが割り当てられます。 ESXi ホストの TEP IP アドレスは Edge ノードの TEP IP アドレスとのオーバーレイ トンネルを確立するため、VLAN IP アドレスはルーティング可能である必要があります。 Tier-0 ゲートウェイへの North-South 接続を提供するには、追加の VLAN が必要です。 IP アドレス プールはクラスタ間で共有できます。ただし、ホスト オーバーレイの IP アドレス プール/VLAN を Edge オーバーレイの IP アドレス プール/VLAN と共有することはできません。 **注：** ホスト TEP と Edge TEP が異なる物理 NIC を使用している場合、同じ VLAN を使用できます。
Tier-0 アップリンクの IP アドレス	/24 プライベート IP アドレス	Tier 0 アップリンクに使用される IP サブネット。Tier 0 アップリンクの IP アドレスの要件は次のとおりです。 ■ 1 つの IP アドレス：Edge の冗長性を使用しない場合。 ■ 4 つの IP アドレス：BGP と Edge の冗長性を使用する場合。Edge ごとに 2 つの IP アドレス。 ■ 3 つの IP アドレス：スタティック ルートと Edge の冗長性を使用する場合。 Edge 管理の IP アドレス、サブネット、ゲートウェイ、アップリンクの IP アドレス、サブネット、ゲートウェイは一意である必要があります。
物理ネットワークの MTU	1600	オーバーレイ トラフィックを伝送するネットワークでは、MTU サイズを 1,600 以上にする必要があります。
vSphere ポッド CIDR 範囲	/23 プライベート IP アドレス	vSphere ポッド の IP アドレスを提供するプライベート CIDR 範囲。これらのアドレスは、Tanzu Kubernetes クラスタ ノードにも使用されます。 クラスタごとに一意の vSphere ポッド CIDR 範囲を指定する必要があります。 **注：** vSphere ポッド の CIDR 範囲と、Kubernetes サービス アドレスの CIDR 範囲が重複しないようにする必要があります。

コンポーネント	最小数	必要な構成
Kubernetes サービスの CIDR 範囲	/16 プライベート IP アドレス	Kubernetes サービスに IP アドレスを割り当てるためのプライベート CIDR 範囲。スーパーバイザー クラスタ ごとに一意の Kubernetes サービス CIDR 範囲を指定する必要があります。
出力方向 CIDR の範囲	/27 固定 IP アドレス	Kubernetes サービスの出力方向 IP アドレスを決定するプライベート CIDR 注釈。スーパーバイザー クラスタ 内の名前空間ごとに 1 つの出力方向 IP アドレスのみが割り当てられます。出力方向 IP アドレスは、外部エンティティが名前空間内のサービスとの通信に使用するアドレスです。出力方向 IP アドレスの数によって、スーパーバイザー クラスタ で保持できる出力方向ポリシーの数が制限されます。 最小値は /27 以上の CIDR です。たとえば、`10.174.4.96/27`。 **注：** 出力方向 IP アドレスと入力方向 IP アドレスは重複できません。
入力方向 CIDR	/27 固定 IP アドレス	入力方向の IP アドレスに使用されるプライベート CIDR 範囲。入力方向を使用すると、外部ネットワークから スーパーバイザー クラスタ に送信される要求にトラフィック ポリシーを適用できます。入力方向 IP アドレスの数によって、クラスタで保持できる入力の数が制限されます。 最小値は /27 以上の CIDR です。 **注：** 出力方向 IP アドレスと入力方向 IP アドレスは重複できません。

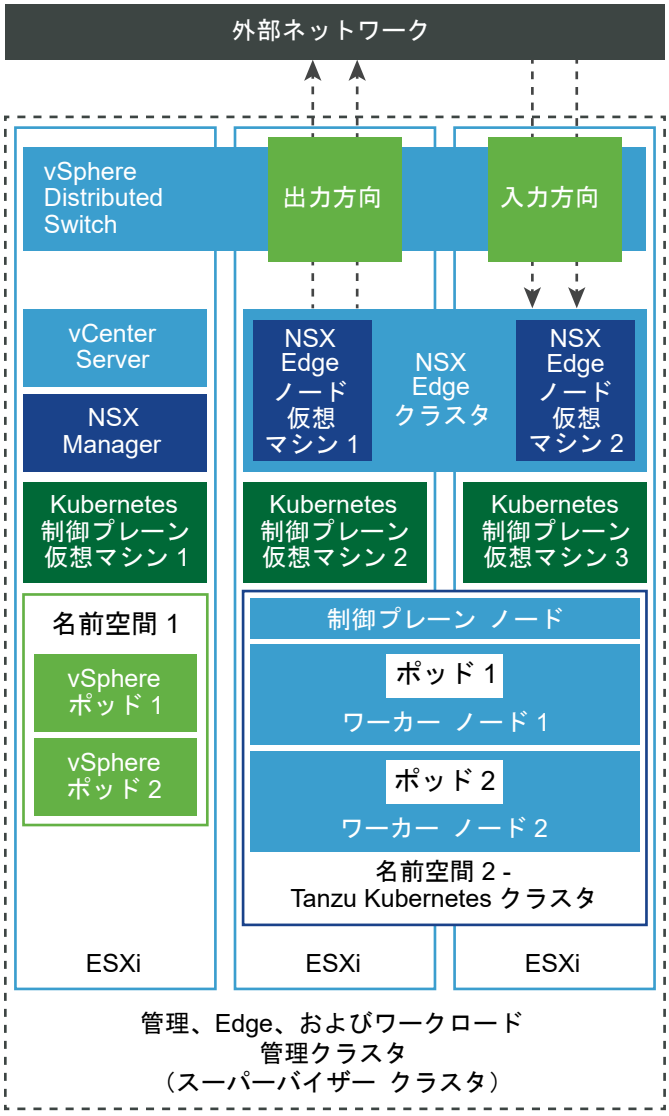
NSX-T Data Center を使用した スーパーバイザー クラスタ のトポロジ

Kubernetes ワークロードのニーズや基盤となるネットワーク インフラストラクチャに応じて、クラスタにさまざまなトポロジを適用できます。

管理、Edge、およびワークロード ドメイン クラスタのトポロジ

管理、Edge、およびワークロード管理の機能を 1 つの vSphere クラスタに統合した vSphere with Tanzu をデプロイできます。

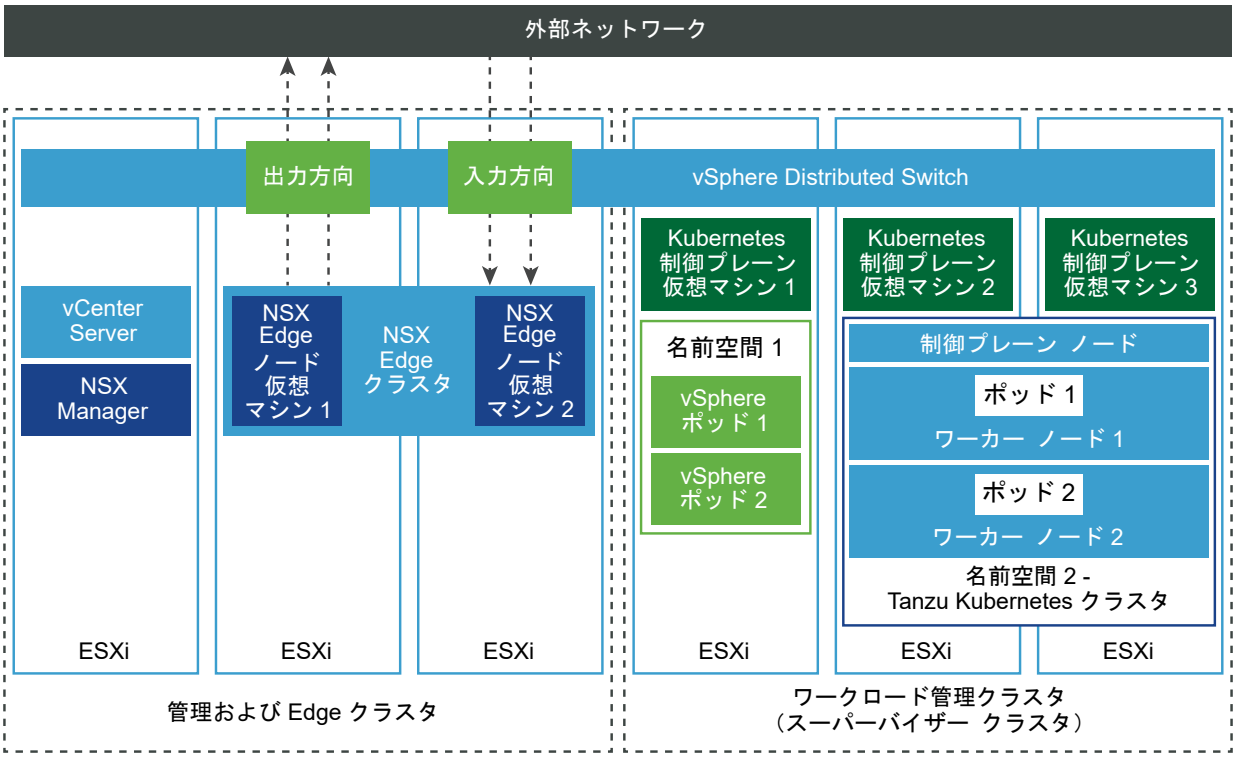
図 4-4. 管理、Edge、およびワークロード管理クラスタ



管理および Edge クラスタと、ワークロード管理クラスタを分離したトポロジ

vSphere with Tanzu は、2 つのクラスタにデプロイできます。1 つは管理機能と Edge 機能用、もう 1 つはワークロード管理専用です。

図 4-5. 管理および Edge クラスタとワークロード管理クラスタ



スーパーバイザー クラスタ で NSX-T Data Center を構成する際のベスト プラクティスに関する考慮事項

NSX-T Data Center を使用して vSphere クラスタを スーパーバイザー クラスタ として構成する場合は、次に示すベスト プラクティスを考慮してください。

- NSX Edge の vSAN データストアを使用します。
- vSAN データストアを使用する場合は、vSAN 環境がワークロードに合わせて適切にサイズ調整されていることを確認します。vSAN でカーネル メモリが使用されるため、vSAN のセットアップにはより多くのメモリが必要になります。これにより、NSX Edge 仮想マシンで使用できるメモリが減少します。適切にサイズ調整するには、vSAN の計算ツールを使用します。詳細については、vSAN ReadyNode Sizer を参照してください。
- NFS データストアを使用する場合は、クラスタ内のすべてのホストで共有されていることを確認します。NSX Edge ノードごとに一意の NFS データストアを作成します。
- 各 NSX Edge クラスタに専用のリソース プールを設定します。リソース プールを他の仮想マシンと共有しないでください。
- ESXi ホスト オーバーレイを設定する場合は、1 ～ 4,094 の範囲の VLAN を使用します。
- Edge オーバーレイを設定する場合は、1 ～ 4,094 の範囲の VLAN を使用します。

vSphere with Tanzu で使用する NSX-T Data Center のインストールと構成

vSphere with Tanzu では、スーパーバイザー クラスタ、vSphere 名前空間、名前空間内で実行されるすべてのオブジェクト（vSphere ポッド、仮想マシン、Tanzu Kubernetes クラスタなど）への接続を有効にするために、特定のネットワーク構成が必要です。vSphere 管理者は、vSphere with Tanzu で使用する NSX-T Data Center をインストールして構成します。

このセクションでは、新しい NSX-T Data Center インスタンスをデプロイして スーパーバイザー クラスタ ネットワークを構成する方法について説明しますが、手順は既存の NSX-T Data Center デプロイにも適用できます。また、このセクションでは、スーパーバイザー クラスタ ワークロード ドメインを設定するときに VMware Cloud Foundation SDDC Manager で行われている処理を理解するための背景も説明します。

前提条件

- 環境が vSphere クラスタを スーパーバイザー クラスタ として設定するためのシステム要件を満たしていることを確認します。要件の詳細については、NSX-T Data Center を使用して vSphere with Tanzu をセットアップするためのシステム要件を参照してください。
- Kubernetes のアドオンを含む VMware vSphere 7 Enterprise Plus のライセンスを、スーパーバイザー クラスタ の一部となるすべての ESXi ホストに割り当てます。
- 制御プレーン仮想マシン、ポッドの短期ディスク、コンテナ イメージの配置用のストレージ ポリシーを作成します。
- クラスタの共有ストレージを構成します。vSphere DRS と HA、およびコンテナのパーシステント ボリュームの保存には、共有ストレージが必要です。
- vSphere クラスタで DRS と HA が有効であり、DRS が完全自動化モードに設定されていることを確認します。
- クラスタに関するクラスタ全体の構成の変更権限があることを確認します。

vSphere Distributed Switch の作成

スーパーバイザー クラスタ 内のすべてのホストのネットワーク構成を処理するには、vSphere Distributed Switch を作成します。

手順

1. vSphere Client で、データセンターに移動します。
2. ナビゲータでデータセンターを右クリックし、[Distributed Switch] - [新しい Distributed Switch] の順に選択します。
3. 新しい Distributed Switch の名前を入力します。

   例えば、`DSwitch` です。
4. [バージョンの選択] に Distributed Switch のバージョンを入力します。

   [7.0.0 - ESXi 7.0 以降] を選択します。

5 [設定] にアップリンク ポートの数を入力します。

2 の値を入力します。

6 設定内容を確認して、[終了] をクリックします。

7 作成した Distributed Switch を右クリックし、[設定] - [設定の編集] の順に選択します。

8 [詳細] タブで、MTU（バイト）値として 1600 を超える値を入力し、[OK] をクリックします。

オーバーレイ トラフィックを伝送するネットワークでは、MTU サイズを 1,600 以上にする必要があります。

たとえば、9000 です。

次のステップ

分散ポート グループを追加します。分散ポート グループの作成を参照してください。

分散ポート グループの作成

NSX Edge ノードのアップリンク、Edge ノード TEP、管理ネットワーク、および共有ストレージごとに、分散ポートグループを作成します。

前提条件

vSphere Distributed Switch が作成されていることを確認します。

手順

1 vSphere Client で、データセンターに移動します。

2 ナビゲータで Distributed Switch を右クリックして、[分散ポート グループ] - [新規分散ポート グループ] の順に選択します。

3 NSX Edge アップリンクのポート グループを作成します。

例えば、DPortGroup-EDGE-UPLINK です。

4 [VLAN タイプ] を VLAN トランクに設定します。

5 Distributed Switch を右クリックして、[アクション] メニューから [分散ポート グループ] - [分散ポート グループの管理] の順に選択します。

6 [チーミングおよびフェイルオーバー] を選択して、[次へ] をクリックします。

7 アクティブ アップリンクとスタンバイ アップリンクを構成します。

例えば、アクティブ アップリンクは Uplink1、スタンバイ アップリンクは Uplink2 です。

8 Edge ノード TEP、管理ネットワーク、および共有ストレージについて、手順 4 ～ 7 を繰り返します。

たとえば、次のポート グループを作成します。

ポート グループ	名前	VLAN タイプ
Edge ノード TEP	DPortGroup-EDGE-TEP	[VLAN タイプ] を VLAN トランクに設定します。 アクティブ アップリンクを Uplink2、スタンバイ アップリンクを Uplink1 と設定します。 **注：** Edge ノード TEP に使用される VLAN は、ESXi TEP に使用される VLAN とは異なる必要があります。
マネージメント ツール	DPortGroup-MGMT	[VLAN タイプ] を [VLAN] に設定し、管理ネットワークの VLAN ID を入力します。たとえば、1060 です。
共有ストレージまたは vSAN	DPortGroup-VSAN	[VLAN タイプ] を [VLAN] に設定し、VLAN ID を入力します。たとえば、3082 です。

9 （オプション） 次のコンポーネントのポート グループを作成します。

- vSphere vMotion
- 仮想マシンのトラフィック

次のステップ

vSphere Distributed Switch にホストを追加します。vSphere Distributed Switch へのホストの追加を参照してください。

vSphere Distributed Switch へのホストの追加

vSphere Distributed Switch を使用して環境のネットワークを管理するには、スイッチにホストを関連付ける必要があります。Distributed Switch にホストの物理 NIC、VMkernel アダプタ、および仮想マシン ネットワーク アダプタを接続してください。

前提条件

- スイッチに接続する物理 NIC に割り当てるための十分なアップリンクが Distributed Switch で使用可能であることを確認します。
- Distributed Switch で使用できる分散ポート グループが 1 つ以上あることを確認します。
- 分散ポート グループのチーミングおよびフェイルオーバー ポリシーで、アクティブなアップリンクが構成されていることを確認します。

手順

1 vSphere Client で、[ネットワーク] を選択して Distributed Switch に移動します。

2 [アクション] メニューから、[ホストの追加と管理] を選択します。

3 [タスクを選択] 画面で、[ホストの追加] を選択し、[次へ] をクリックします。

4 [ホストの選択] 画面で、[新規ホスト] をクリックし、データセンター内のホストを選択して [OK] をクリックします。次に [次へ] をクリックします。

5 [物理アダプタの管理] 画面で、Distributed Switch の物理 NIC を構成します。

a [他のスイッチ上/未要求] リストから、物理 NIC を選択します。

他のスイッチにすでに接続されている物理 NIC を選択した場合、その物理 NIC は現在の Distributed Switch に移行されます。

b [アップリンクの割り当て] をクリックします。

c アップリンクを選択します。

d このアップリンクをクラスタ内のすべてのホストに割り当てるには、[このアップリンクの割り当てを残りのホストに適用します] を選択します。

e [OK] をクリックします。

たとえば、`Uplink 1` を `vmnic0` に割り当て、`Uplink 2` を `vmnic1` に割り当てます。

6 [次へ] をクリックします。

7 [VMkernel アダプタの管理] 画面で、VMkernel アダプタを構成します。

a VMkernel アダプタを選択し、[ポート グループの割り当て] をクリックします。

b 分散ポート グループを選択します。

たとえば、[DPortGroup] です。

c このポート グループをクラスタ内のすべてのホストに適用するには、[このポート グループの割り当てを残りのホストに適用します] を選択します。

d [OK] をクリックします。

8 [次へ] をクリックします。

9 （オプション）[仮想マシン ネットワークの移行] 画面で [仮想マシン ネットワークの移行] チェック ボックスをオンにして、仮想マシン ネットワークを構成します。

a 仮想マシンのすべてのネットワーク アダプタを分散ポート グループに接続するには、仮想マシンを選択するか、個々のネットワーク アダプタを選択して、そのアダプタのみに接続します。

b [ポート グループの割り当て] をクリックします。

c リストから分散ポート グループを選択し、[OK] をクリックします。

d [次へ] をクリックします。

次のステップ

NSX Manager をデプロイし、構成します。NSX Manager のデプロイと構成を参照してください。

NSX Manager のデプロイと構成

vSphere Client を使用して NSX Manager を vSphere クラスタにデプロイし、vSphere with Tanzu で使用することができます。

OVA ファイルを使用して NSX Manager をデプロイするには、この手順を実行します。

ユーザー インターフェイスまたは CLI を使用した NSX Manager のデプロイの詳細については、『NSX-T Data Center インストールガイド』を参照してください。

前提条件

- 環境がネットワークの要件を満たしていることを確認します。詳細については、NSX-T Data Center を使用して vSphere with Tanzu をセットアップするためのシステム要件を参照してください。
- 必要なポートが開いていることを確認します。ポートとプロトコルの詳細については、『NSX-T Data Center インストール ガイド』を参照してください。

手順

1. VMware ダウンロード ポータルで NSX-T Data Center OVA ファイルを見つけます。

   ダウンロード URL をコピーするか、OVA ファイルをダウンロードします。
2. 右クリックして [OVF テンプレートのデプロイ] を選択し、インストール ウィザードを開始します。
3. [OVF テンプレートの選択] タブで、OVA のダウンロード URL を入力するか、OVA ファイルに移動します。
4. [名前とフォルダの選択] タブで、NSX Manager 仮想マシン (VM) の名前を入力します。
5. [コンピューティング リソースの選択] タブで、NSX Manager をデプロイする vSphere クラスタを選択します。
6. [次へ] をクリックして、詳細を確認します。
7. [構成] タブで、NSX-T のデプロイ サイズを選択します。

   推奨される最小デプロイ サイズは [中] です。
8. [ストレージの選択] タブで、デプロイ用の共有ストレージを選択します。
9. [仮想ディスク フォーマットの選択] で [シン プロビジョニング] を選択して、シン プロビジョニングを有効にします。

   デフォルトでは、仮想ディスクはシック プロビジョニングされます。
10. [ネットワークの選択] タブの [ターゲット ネットワーク] で、NSX Manager の管理ポート グループまたはターゲット ネットワークを選択します。

    たとえば、`DPortGroup-MGMT` です。
11. [テンプレートのカスタマイズ] タブで、NSX Manager のシステム root、CLI 管理者、および監査パスワードを入力します。パスワードの強度の制限に従ってパスワードを入力する必要があります。
    - 12 文字以上。
    - 小文字が 1 文字以上。
    - 大文字が 1 文字以上。
    - 数字が 1 文字以上。
    - 特殊文字が 1 文字以上。
    - 異なる文字が 5 文字以上。

- デフォルトのパスワードの複雑性ルールが Linux PAM モジュールによって適用されます。

12 デフォルトの IPv4 ゲートウェイ、管理ネットワークの IPv4、管理ネットワークのネットマスク、DNS サーバ、ドメイン検索リスト、および NTP IP アドレスを入力します。

13 SSH を有効にして、NSX Manager コマンド ラインに root による SSH ログインを許可します。

デフォルトでは、SSH オプションはセキュリティ上の理由から無効になっています。

14 カスタム OVF テンプレートの仕様が正確であることを確認し、[終了] をクリックしてインストールを開始します。

15 NSX Manager が起動したら、admin として CLI にログインし、`get interface eth0` コマンドを実行して、IP アドレスが想定どおりに適用されていることを確認します。

16 `get services` コマンドを入力して、すべてのサービスが実行されていることを確認します。

NSX Manager ノードのデプロイによるクラスタの形成

NSX Manager クラスタは高可用性を提供します。vCenter Server によって管理されている ESXi ホストにのみ、ユーザー インターフェイスを使用して NSX Manager ノードをデプロイできます。NSX Manager クラスタを作成するには、2 台の追加ノードをデプロイして、合計 3 台のノードによるクラスタを形成します。ユーザー インターフェイスから新しいノードをデプロイすると、そのノードは最初にデプロイされたノードに接続してクラスタを形成します。最初にデプロイされたノードのリポジトリのすべての詳細とパスワードは、新しくデプロイされたノードと同期されます。

前提条件

- NSX Manager ノードがインストールされていることを確認します。
- コンピュート マネージャが構成されていることを確認します。
- 必要なポートが開いていることを確認します。
- ESXi ホストにデータストアが構成されていることを確認します。
- IP アドレスとゲートウェイ、DNS サーバの IP アドレス、ドメイン検索リスト、および NTP サーバの IP アドレスが、NSX Manager で使用できるようになっていることを確認します。
- ターゲット仮想マシンのポート グループ ネットワークがあることを確認します。管理仮想マシン ネットワークに NSX-T Data Center アプライアンスを配置します。

手順

1 ブラウザから管理者権限で https://<manager-ip-address> の NSX Manager にログインします。

2 アプライアンスをデプロイするために、[システム] - [アプライアンス] - [NSX アプライアンスの追加] の順に選択します。

3 アプライアンスの詳細を入力します。

オプション	説明
ホスト名	ノードに使用するホスト名または FQDN を入力します。
管理 IP アドレス/ネットマスク	ノードに割り当てる IP アドレスを入力します。

オプション	説明
管理ゲートウェイ	ノードで使用するゲートウェイ IP アドレスを入力します。
DNS サーバ	ノードで使用する DNS サーバの IP アドレスのリストを入力します。
NTP サーバ	NTP サーバの IP アドレスのリストを入力します
ノード サイズ	オプションから [中（6 個の vCPU、24 GB の RAM、300 GB のストレージ）] のフォーム ファクタを選択します。

4 アプライアンスの構成の詳細を入力します

オプション	説明
コンピュート マネージャ	コンピュート マネージャとして構成した vCenter Server を選択します。
コンピューティング クラスタ	ノードが参加するクラスタを選択します。
データストア	ノードのファイル用のデータストアを選択します。
仮想ディスクのフォーマット	[シン プロビジョニング] フォーマットを選択します。
ネットワーク	[ネットワークの選択] をクリックして、ノードの管理ネットワークを選択します。

5 アクセスと認証情報の詳細を入力します。

オプション	説明
SSH の有効化	ボタンを切り替えて、新しいノードへの SSH ログインを許可します。
root アクセスの有効化	ボタンを切り替えて、新しいノードへの root アクセスを許可します。
システムの root 認証情報	新しいノードの root パスワードを設定して確認します。 パスワードの強度の制限に従ってパスワードを入力する必要があります。 ■ 12 文字以上。 ■ 小文字が 1 文字以上。 ■ 大文字が 1 文字以上。 ■ 数字が 1 文字以上。 ■ 特殊文字が 1 文字以上。 ■ 異なる文字が 5 文字以上。 ■ デフォルトのパスワードの複雑性ルールが Linux PAM モジュールによって適用されます。
管理者 CLI 認証情報と監査 CLI 認証情報	root に設定したものと同じパスワードを使用する場合は、[root パスワードと同じ] チェックボックスをオンにします。または、チェックボックスをオフにして、別のパスワードを設定します。

6 [アプライアンスのインストール] をクリックします。

新しいノードがデプロイされます。デプロイ プロセスは [システム] - [アプライアンス] 画面で追跡できます。インストールが完了してクラスタが安定するまでは、ノードを追加しないでください。

7 デプロイ、クラスタの形成、およびリポジトリの同期が完了するまで待機します。

ノードの参加とクラスタの安定化には、10 ～ 15 分程度かかる場合があります。クラスタに他の変更を加える前に、すべてのクラスタ サービス グループの状態が接続中であることを確認します。

8 ノードが起動したら、管理者として CLI にログインし、`get interface eth0` コマンドを実行して、IP アドレスが想定どおりに適用されていることを確認します。

9 クラスタのノードが 2 台のみの場合は、別のアプライアンスを追加します。[システム] - [アプライアンス] - [NSX アプライアンスの追加] の順に選択して、構成手順を繰り返します。

ライセンスの追加

NSX Manager を使用してライセンスを追加します。

前提条件

NSX-T Data CenterAdvanced 以上のライセンスを取得します。

手順

1 NSX Manager にログインします。

2 [システム] - [ライセンス] - [追加] の順に選択します。

3 ライセンス キーを入力します。

4 [追加] をクリックします。

コンピュート マネージャの追加

コンピュート マネージャは、ホストや仮想マシンなどのリソースを管理するアプリケーションです。NSX-T Data Center に関連付けられている vCenter Server を NSX Manager のコンピュート マネージャとして構成します。

手順

1 NSX Manager にログインします。

2 [システム] - [ファブリック] - [コンピュート マネージャ] - [追加] の順に選択します。

3 コンピュート マネージャの詳細を入力します。

オプション	説明
名前と説明	vCenter Server の名前および説明を入力します。
FQDN または IP アドレス	vCenter Server の FQDN または IP アドレスを入力します。
ユーザー名とパスワード	vCenter Server のログイン認証情報を入力します。

4 [信頼の有効化] を選択して、vCenter Server が NSX-T Data Center と通信できるようにします。

5 NSX Manager のサムプリント値を指定しなかった場合、システムはサムプリントを識別して表示します。

6 [追加] をクリックして、サムプリントを受け入れます。

結果

しばらくすると、コンピュート マネージャが vCenter Server に登録され、接続ステータスが 接続中に変わります。vCenter Server の FQDN または PNID が変更された場合は、NSX Manager に再登録する必要があります。詳細については、『NSX Manager による vCenter Server の登録』を参照してください。

注： vCenter Server が正常に登録されたら、コンピュート マネージャを削除する前に NSX Manager 仮想マシンをパワーオフして削除しないでください。そのようにすると、新しい NSX Manager をデプロイするときに、同じ vCenter Server を再度登録することができなくなります。vCenter Server がすでに別の NSX Manager に登録されていることを示すエラーが表示されます。

コンピュート マネージャ名をクリックすると、詳細の表示、コンピュート マネージャの編集、またはコンピュート マネージャに適用されるタグの管理を行うことができます。

トランスポート ゾーンの作成

トランスポート ゾーンは、特定のネットワークを使用できるホストおよび仮想マシンを示します。トランスポート ゾーンは、1 つ以上のホスト クラスタにまたがることができます。

vSphere 管理者は、デフォルトのトランスポート ゾーンを使用するか、次のトランスポート ゾーンを作成します。

- スーパーバイザー クラスタ 制御プレーンの仮想マシンによって使用されるオーバーレイ トランスポート ゾーン。
- 物理ネットワークへのアップリンクに使用する NSX Edge ノードの VLAN トランスポート ゾーン。

手順

1. NSX Manager にログインします。
2. [システム] - [ファブリック] - [トランスポート ゾーン] - [追加] の順に選択します。
3. トランスポート ゾーンの名前と、必要に応じて説明を入力します。
4. トラフィック タイプを選択します。

   [オーバーレイ] または [VLAN] を選択できます。

   デフォルトでは、次のトランスポート ゾーンがあります。

   - 名前が `nsx-vlan-transportzone` の VLAN トランスポート ゾーン。
   - 名前が `nsx-overlay-transportzone` のオーバーレイ トランスポート ゾーン。
5. （オプション） 1 つ以上のアップリンク チーミング ポリシー名を入力します。

   トランスポート ゾーンに接続されたセグメントは、これらの名前付きチーミング ポリシーを使用します。一致する名前付きチーミング ポリシーをセグメントが見つけられない場合は、デフォルトのアップリンク チーミング ポリシーが使用されます。

結果

[トランスポート ゾーン] 画面に新しいトランスポート ゾーンが表示されます。

ホストのトンネル エンドポイント IP アドレス用の IP アドレス プールの作成

ESXi ホストのトンネル エンドポイント (TEP) と Edge ノードの IP アドレス プールを作成します。TEP は、オーバーレイ フレームの NSX-T のカプセル化を開始および終了する ESXi ホストを識別するために、外部 IP ヘッダーで使用されるソース IP アドレスとターゲット IP アドレスです。TEP の IP アドレスには、DHCP を使用するか、手動で設定した IP アドレス プールを使用できます。

手順

1 NSX Manager にログインします。

2 [ネットワーク] - [IP アドレス プール] - [IP アドレス プールの追加] の順に選択します。

3 次の IP アドレス プールの詳細を入力します。

オプション	説明
名前と説明	IP アドレス プール名と、必要に応じて説明を入力します。 例えば、`ESXI-TEP-IP-POOL` です。
IP アドレス範囲	IP アドレスの割り当ての範囲を入力します。 たとえば、`10.197.79.158 - 10.197.79.160`。
ゲートウェイ	ゲートウェイ IP アドレス を入力します。 たとえば、`10.197.79.253` です。
CIDR	ネットワーク アドレスを CIDR 表記で入力します。 たとえば、`10.197.79.0/24` です。

4 [追加] および [適用] クリックします。

5 手順 2 ～ 4 を繰り返して、Edge ノードの IP アドレス プールを作成します。

たとえば、`EDGE-TEP-IP-POOL` です。

6 作成した TEP の IP アドレス プールが [IP アドレス プール] 画面に表示されていることを確認します。

ホスト アップリンク プロファイルの作成

ホスト アップリンク プロファイルは、ESXi ホストから NSX-T Data Center セグメントへのアップリンクのポリシーを定義します。

手順

1 NSX Manager にログインします。

2 [システム] - [ファブリック] - [プロファイル] - [アップリンク プロファイル] - [追加] の順に選択します。

3 アップリンク プロファイルの名前と、オプションでアップリンク プロファイルの説明を入力します。

例えば、`ESXI-UPLINK-PROFILE` です。

4 [チーミング] セクションで、[追加] をクリックして名前付けチーミング ポリシーを追加し、[フェイルオーバーの順序] ポリシーを構成します。

アクティブ アップリンクのリストが指定され、トランスポート ノードの各インターフェイスが 1 つのアクティブ アップリンクに固定されます。この設定により、複数のアクティブ アップリンクを同時に使用できます。

5 アクティブ アップリンクとスタンバイ アップリンクを構成します。

たとえば、`uplink-1` をアクティブ アップリンクとして、`uplink-2` をスタンバイ アップリンクとして構成します。

6 トランスポート VLAN の値を入力します。

アップリンク プロファイルで設定されたトランスポート VLAN がオーバーレイ トラフィックをタグ付けし、VLAN ID はトンネル エンドポイント (TEP) によって使用されます。

たとえば、`1060` です。

7 MTU 値を入力します。

アップリンク プロファイル MTU のデフォルト値は 1,600 です。

注： 値は 1,600 以上にする必要がありますが、物理スイッチと vSphere Distributed Switch の MTU 値よりも大きくすることはできません。

Edge アップリンク プロファイルの作成

アップリンク プロファイルを作成し、フェイルオーバーの順序チーミング ポリシーを追加して、Edge 仮想マシンのオーバーレイ トラフィックに対して 1 つのアップリンクが使用されるようにします。

手順

1 NSX Manager にログインします。

2 [システム] - [ファブリック] - [プロファイル] - [アップリンク プロファイル] - [追加] の順に選択します。

3 アップリンク プロファイルの名前を入力し、オプションでアップリンク プロファイルの説明を追加します。

例えば、`EDGE-UPLINK-PROFILE` です。

4 [チーミング] セクションで、[追加] をクリックして名前付けチーミング ポリシーを追加し、[フェイルオーバー] ポリシーを構成します。

アクティブ アップリンクのリストが表示され、トランスポート ノードの各インターフェイスが 1 つのアクティブ アップリンクに固定されます。この設定により、複数のアクティブ アップリンクを同時に使用できます。

5 アクティブ アップリンクを構成します。

たとえば、`uplink-1` をアクティブ アップリンクとして構成します。

6 アップリンクを [アップリンク プロファイル] 画面で確認します。

トランスポート ノード プロファイルの作成

トランスポート ノード プロファイルは、プロファイルが添付されている特定のクラスタ内のホストに NSX-T Data Center をどのようにインストールして構成するかを定義します。

前提条件

オーバーレイ トランスポート ゾーンを作成してあることを確認します。

手順

1 NSX Manager にログインします。

2 [システム] - [ファブリック] - [プロファイル] - [トランスポート ノード プロファイル] - [追加] の順に選択します。

3 トランスポート ノード プロファイルの名前と、必要に応じて説明を入力します。

例えば、`HOST-TRANSPORT-NODE-PROFILE` です。

4 [新しいノード スイッチ] セクションで、[タイプ] として `VDS` を選択します。

5 [モード] として `Standard` を選択します。

6 リストから vCenter Server と Distributed Switch の名前を選択します。

たとえば、`DSwitch`。

7 以前に作成したオーバーレイ トランスポート ゾーンを選択します。

例えば、`NSX-OVERLAY-TRANSPORTZONE` です。

8 以前に作成したホスト アップリンク プロファイルを選択します。

例えば、`ESXI-UPLINK-PROFILE` です。

9 [IP アドレスの割り当て] リストから [IP アドレス プールを使用] を選択します。

10 以前に作成したホスト TEP プールを選択します。

例えば、`ESXI-TEP-IP-POOL` です。

11 [チーミング ポリシー スイッチ マッピング] で編集アイコンをクリックし、NSX-T アップリンク プロファイルで定義されているアップリンクを vSphere Distributed Switch アップリンクにマッピングします。

たとえば、`uplink-1 (active)` を `Uplink 1` にマッピングし、`uplink-2 (standby)` を `Uplink 2` にマッピングします。

12 [追加] をクリックします。

13 作成したプロファイルが [トランスポート ノード プロファイル] 画面に一覧表示されていることを確認します。

クラスタ上の NSX-T Data Center の構成

NSX-T Data Center をインストールしてオーバーレイ TEP を準備するには、トランスポート ノード プロファイルを vSphere クラスタに適用します。

前提条件

トランスポート ノード プロファイルが作成されていることを確認します。

手順

1 NSX Manager にログインします。

2 [システム] - [ファブリック] - [ノード] - [ホスト トランスポート ノード] の順に選択します。

3 [管理元] ドロップダウン メニューから、既存の vCenter Server を選択します。

画面には、使用可能な vSphere クラスタが一覧表示されます。

4 NSX-T Data Center を構成するコンピューティング クラスタを選択します。

5 [NSX の構成] をクリックします。

6 以前に作成したトランスポート ノード プロファイルを選択し、[適用] をクリックします。

例えば、`HOST-TRANSPORT-NODE-PROFILE` です。

7 [ホスト トランスポート ノード] 画面で、NSX-T Data Center の構成状態が `Success` であること、およびクラスタ内にあるホストの NSX Manager の接続ステータスが `Up` であることを確認します。

結果

NSX-T Data Center のインストールとオーバーレイ TEP の準備のために、以前に作成したトランスポート ノード プロファイルが vSphere クラスタに適用されます。

NSX Edge トランスポートノードの設定とデプロイ

NSX Edge 仮想マシン (VM) を NSX-T Data Center ファブリックに追加して、NSX Edge トランスポート ノード仮想マシンとして構成することができます。

前提条件

トランスポート ゾーン、Edge アップリンク プロファイル、Edge TEP IP アドレス プールを作成済みであることを確認します。

手順

1 NSX Manager にログインします。

2 [システム] - [ファブリック] - [ノード] - [Edge トランスポート ノード] - [Edge 仮想マシンの追加] の順に選択します。

3 [名前と説明] に NSX Edge の名前を入力します。

たとえば、`nsx-edge-1`。

4 vCenter Server のホスト名または FQDN を入力します。

たとえば、`nsx-edge-1.lab.com` です。

5 `Large` フォーム ファクタを選択します。

6 [認証情報] に NSX Edge の CLI および root パスワードを入力します。パスワードの強度の制限に従ってパスワードを入力する必要があります。

- 12 文字以上。
- 小文字が 1 文字以上。
- 大文字が 1 文字以上。
- 数字が 1 文字以上。

- 特殊文字が 1 文字以上。
- 異なる文字が 5 文字以上。
- デフォルトのパスワードの複雑性ルールが Linux PAM モジュールによって適用されます。

7 CLI および Root 認証情報に対して [SSH ログインを許可] を有効にします。

8 [デプロイの構成] で、次のプロパティを構成します。

オプション	説明
コンピュート マネージャ	ドロップダウン メニューからコンピュート マネージャを選択します。 たとえば、`vCenter` を選択します。
クラスタ	ドロップダウン メニューからクラスタを選択します。 たとえば、`Compute-Cluster` を選択します。
データストア	リストから共有データストアを選択します。 たとえば、`vsanDatastore` です。

9 ノードを設定します。

オプション	説明
IP アドレスの割り当て	[固定] を選択します。 以下の値を入力します。 ■ [管理 IP アドレス] : vCenter Server 管理ネットワークと同じ VLAN 上の IP アドレスを入力します。 たとえば、`10.197.79.146/24` です。 ■ [デフォルト ゲートウェイ] : 管理ネットワークのデフォルト ゲートウェイ。 たとえば、`10.197.79.253` です。
管理インターフェイス	[インターフェイスの選択] をクリックし、以前に作成したドロップダウン メニューから、管理ネットワークと同じ VLAN 上の vSphere Distributed Switch ポート グループを選択します。 たとえば、`DPortGroup-MGMT` です。

10 [NSX の構成] で [スイッチの追加] をクリックして、スイッチのプロパティを構成します。

11 [Edge スイッチ名] のデフォルト名を使用します。

たとえば、`nvds1` です。

12 トランスポート ノードが属するトランスポート ゾーンを選択します。

以前に作成したオーバーレイ トランスポート ゾーンを選択します。

たとえば、`nsx-overlay-transportzone` です。

13 以前に作成した Edge アップリンク プロファイルを選択します。

例えば、`EDGE-UPLINK-PROFILE` です。

14 [IP アドレスの割り当て] の [IP アドレス プールを使用] を選択します。

15 以前に作成した Edge TEP の IP アドレス プールを選択します。

例えば、EDGE-TEP-IP-POOL です。

16 [チーミング ポリシー スイッチ マッピング] セクションで、以前に作成した Edge アップリンク プロファイルにアップリンクをマッピングします。

たとえば、Uplink1 の場合は、DPortGroup-EDGE-TEP を選択します。

17 手順 10 ～ 16 を繰り返して、新しいスイッチを追加します。

たとえば、次の値を構成します。

プロパティ	値
Edge スイッチ名	nvds2
トランスポート ゾーン	nsx-vlan-transportzone
Edge アップリンク プロファイル	EDGE-UPLINK-PROFILE
チーミング ポリシー スイッチ マッピング	DPortGroup-EDGE-UPLINK

18 [終了] をクリックします。

19 2 台目の NSX Edge 仮想マシンについて、手順 2 ～ 18 を繰り返します。

20 [Edge トランスポート ノード] 画面で接続状態を確認します。

NSX Edge クラスタの作成

1 つ以上の NSX Edge が常に使用可能になるようにするには、NSX Edge クラスタを作成します。

手順

1 NSX Manager にログインします。

2 [システム] - [ファブリック] - [ノード] - [Edge クラスタ] - [追加] の順に選択します。

3 NSX Edge クラスタ名を入力します。

例えば、EDGE-CLUSTER です。

4 ドロップダウン メニューからデフォルトの NSX Edge クラスタ プロファイルを選択します。

[nsx-default-edge-high-availability-profile] を選択します。

5 [メンバーのタイプ] ドロップダウン メニューで、[Edge ノード] を選択します。

6 [使用可能] 列で、以前に作成した NSX Edge 仮想マシンを選択し、右矢印をクリックして [選択済み] 列に移動します。

7 たとえば、nsx-edge-1 および nsx-edge-2 などです。

8 [保存] をクリックします。

Tier-0 アップリンク セグメントの作成

Tier-0 アップリンク セグメントは、NSX-T Data Center から物理インフラストラクチャへの North-South 接続を提供します。

前提条件

Tier-0 ゲートウェイが作成されていることを確認します。

手順

1 NSX Manager にログインします。

2 [ネットワーク] - [セグメント] - [セグメントの追加] の順に選択します。

3 セグメントの名前を入力します。

例えば、`TIER-0-LS-UPLINK` です。

4 以前に作成したトランスポート ゾーンを選択します。

たとえば、`nsx-vlan-transportzone` を選択します。

5 [管理者ステータス] を切り替えて有効にします。

6 Tier-0 ゲートウェイの VLAN ID を入力します。

たとえば、`1089` です。

7 [保存] をクリックします。

Tier-0 ゲートウェイの作成

Tier-0 ゲートウェイは、物理インフラストラクチャへの NSX-T Data Center 論理ネットワークの North-South 接続を提供する NSX-T Data Center 論理ルーターです。vSphere with Tanzu は、同じトランスポート ゾーン内の複数の NSX Edge クラスタで複数の Tier-0 ゲートウェイをサポートします。

前提条件

NSX Edge クラスタを作成済みであることを確認します。

手順

1 NSX Manager にログインします。

2 [ネットワーク] - [Tier-0 ゲートウェイ] の順に選択します。

3 [Tier-0 ゲートウェイの追加] をクリックします。

4 Tier-0 ゲートウェイの名前を入力します。

例えば、`Tier-0_VWT` です。

5 アクティブ/スタンバイ HA モードを選択します。

アクティブ/スタンバイ モードでは、選択されたアクティブなメンバーがすべてのトラフィックを処理します。アクティブなメンバーに障害が発生した場合は、新しいメンバーが選択されてアクティブになります。

6 以前に作成した NSX Edge クラスタを選択します。

たとえば、EDGE-CLUSTER を選択します。

7 [保存] をクリックします。

Tier-0 ゲートウェイが作成されます。

8 [はい] を選択して、構成を続行します。

9 インターフェイスを設定します。

a [インターフェイス] を展開して、[設定] をクリックします。

b [インターフェイスの追加] をクリックします。

c 名前を入力します。

たとえば、TIER-0_VWT-UPLINK1 という名前を入力します。

d [タイプ] で [外部] を選択します。

e Edge 論理ルーター – アップリンク VLAN から IP アドレスを入力します。IP アドレスは、以前に作成した NSX Edge 仮想マシン用に設定した管理 IP アドレスとは異なる必要があります。

たとえば、10.197.154.1/24 です。

f [接続先] で、以前に作成した Tier-0 アップリンク セグメントを選択します。

たとえば、TIER-0-LS-UPLINK。

g リストから NSX Edge ノードを選択します。

たとえば、nsx-edge-1 です。

h [保存] をクリックします。

i 2 つ目のインターフェイスについて、手順 a ～ h を繰り返します。

たとえば、IP アドレス 10.197.154.2/24 で nsx-edge-2 Edge ノードに接続される 2 つ目のアップリンク TIER-0_VWT-UPLINK2 を作成します。

j [閉じる] をクリックします。

10 高可用性を構成するには、[HA VIP 構成] で [設定] をクリックします。

a [HA VIP 構成の追加] をクリックします。

b IP アドレスを入力します。

たとえば、10.197.154.3/24。

c インターフェイスを選択します。

たとえば、TIER-0_WVT-UPLINK1 や TIER-0_WVT-UPLINK2 を選択します。

d [追加] および [適用] クリックします。

11 ルーティングを構成するには、[ルーティング] をクリックします。

a スタティック ルートで [設定] をクリックします。

b [スタティック ルートの追加] をクリックします。

c 名前を入力します。

たとえば、`DEFAULT-STATIC-ROUTE` です。

d ネットワーク IP アドレスとして `0.0.0.0/0` を入力します。

e ネクスト ホップを設定するには、[ネクスト ホップの設定] をクリックし、次に [ネクスト ホップの追加] をクリックします。

f ネクスト ホップ ルーターの IP アドレスを入力します。通常、これは NSX Edge 論理ルーター アップリンク VLAN からの管理ネットワーク VLAN のデフォルト ゲートウェイです。

たとえば、`10.197.154.253` です。

g [追加]、[適用]、[保存] の順にクリックします。

h [閉じる] をクリックします。

12 接続を確認するために、物理アーキテクチャの外部デバイスが、構成したアップリンクに対して ping を実行できることを確認します。

次のステップ

スーパーバイザー クラスタ を構成します。『NSX-T Data Center ネットワークを使用したワークロード管理の有効化』を参照してください

vSphere ネットワークと vSphere with Tanzu 用 NSX Advanced Load Balancer の構成

vSphere with Tanzu は、NSX Advanced Load Balancer（別名、Avi Load Balancer Essentials エディションおよび Enterprise エディション）をサポートしています。vSphere with Tanzu 環境に NSX Advanced Load Balancer 20.1.7 をインストールして構成できるのは、スーパーバイザー クラスタ に vSphere Distributed Switch (VDS) ネットワークを使用している場合のみです。

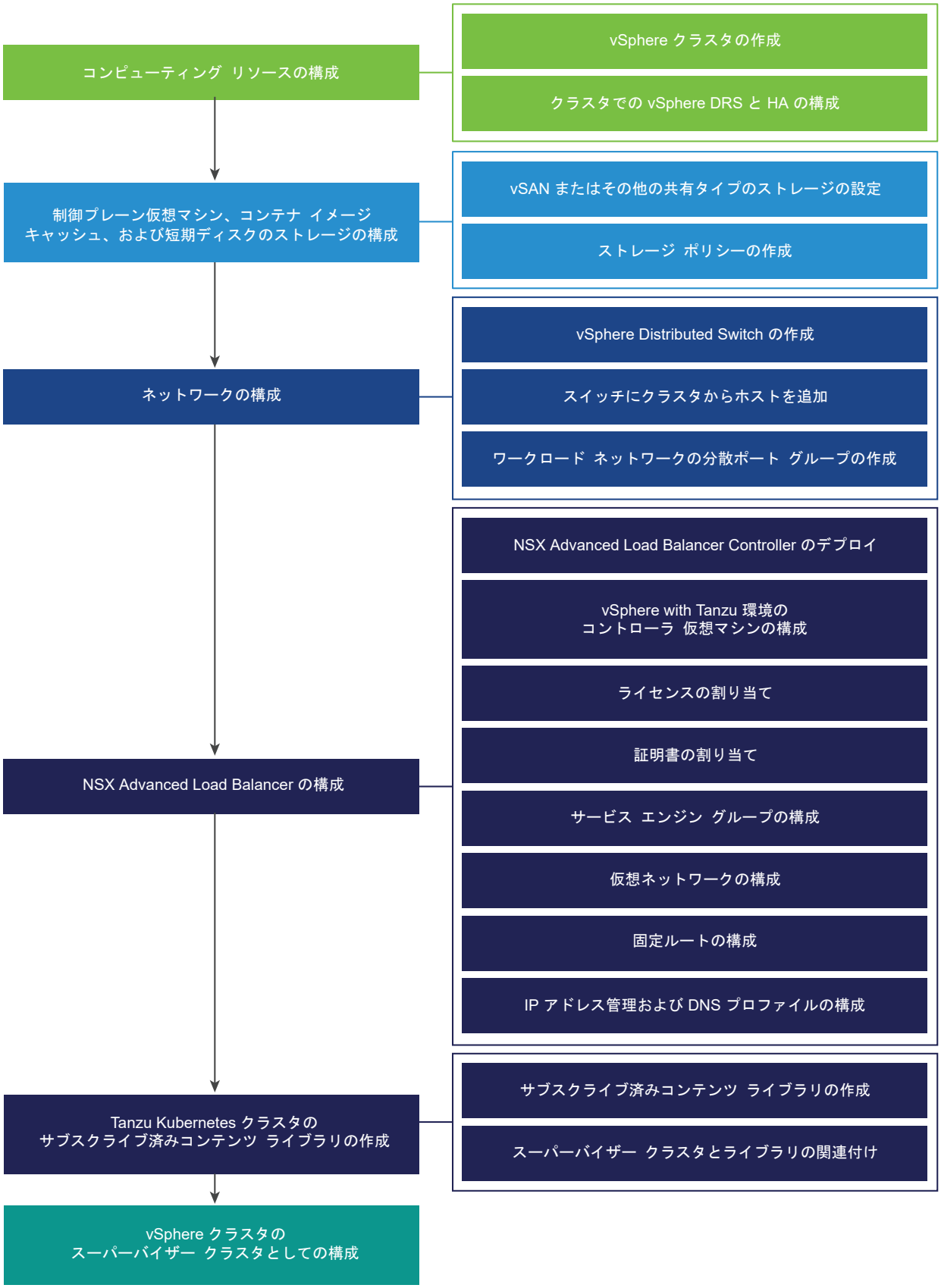
ロード バランサと Tanzu Kubernetes クラスタの連携方法

NSX Advanced Load Balancer は、Tanzu Kubernetes Grid サービス によってプロビジョニングされた Tanzu Kubernetes クラスタに対して、動的スケーリング ロード バランシング エンドポイントを提供します。Avi Controller 仮想マシンをインストールして構成します。コントローラを構成すると、ロード バランシング エンドポイントが自動的にプロビジョニングされます。たとえば、Tanzu Kubernetes クラスタをプロビジョニングすると、コントローラは仮想サービスを作成し、サービスをホストするサービス エンジン仮想マシンをデプロイします。この仮想サービスは、Kubernetes 制御プレーンにロード バランシングを提供します。

そのクラスタにタイプがロード バランサの Kubernetes サービスを作成すると、コントローラは自動的に仮想サービスを作成し、サービス エンジンにデプロイします。最初のサービス エンジンは、最初の仮想サービスが構成された後にのみ作成されます。以降に構成された仮想サービスは、既存のサービス エンジンを使用します。サービス エンジン仮想マシンに複数の仮想サービスをデプロイできます。

vSphere ネットワークと NSX Advanced Load Balancer を使用する スーパーバイザー クラスタ のワークフロー

この図では、vSphere ネットワークと vSphere with Tanzu 用 NSX Advanced Load Balancer を構成するためのワークフローを示します。


コンピューティング リソースの構成
vSphere クラスタの作成
クラスタでの vSphere DRS と HA の構成
制御プレーン仮想マシン、コンテナ イメージ キャッシュ、および短期ディスクのストレージの構成
vSAN またはその他の共有タイプのストレージの設定
ストレージ ポリシーの作成
ネットワークの構成
vSphere Distributed Switch の作成
スイッチにクラスタからホストを追加
ワークロード ネットワークの分散ポート グループの作成
NSX Advanced Load Balancer の構成
NSX Advanced Load Balancer Controller のデプロイ
vSphere with Tanzu 環境の コントローラ 仮想マシンの構成
ライセンスの割り当て
証明書の割り当て
サービス エンジン グループの構成
仮想ネットワークの構成
固定ルートの構成
IP アドレス管理および DNS プロファイルの構成
Tanzu Kubernetes クラスタの サブスクライブ済みコンテンツ ライブラリの作成
サブスクライブ済みコンテンツ ライブラリの作成
スーパーバイザー クラスタとライブラリの関連付け
vSphere クラスタの スーパーバイザー クラスタとしての構成

手順

1 NSX Advanced Load Balancer コンポーネント

NSX Advanced Load Balancer（別名、Avi Load Balancer）のコンポーネントには、Avi Controller クラスタ、サービス エンジン（データ プレーン）仮想マシン、および Avi Kubernetes Operator (AKO) などがあります。

2 vSphere ネットワークと NSX Advanced Load Balancer を使用して vSphere with Tanzu をセットアップするためのシステム要件

Avi Load Balancer とも呼ばれる NSX Advanced Load Balancer を使用して vSphere with Tanzu を構成するには、使用環境が特定の要件を満たしている必要があります。vSphere with Tanzu では Avi ネットワークに対して複数のトポロジ（Avi サービス エンジンおよびロード バランサ サービス用の単一の Distributed Switch ネットワーク、Avi 管理プレーン用の Distributed Switch および NSX Advanced Load Balancer 用の別の Distributed Switch）をサポートしています。

3 vSphere ネットワークと NSX Advanced Load Balancer を使用した スーパーバイザー クラスタ のトポロジ

Avi Controller は常に管理ネットワークにデプロイされ、vCenter Server、ESXi ホスト、および スーパーバイザー クラスタ 制御プレーン ノードとのインターフェイスを行うことができます。サービス エンジンは、管理ネットワークおよびデータ ネットワークへのインターフェイスを使用してデプロイされます。

4 NSX Advanced Load Balancer のインストールと構成

vSphere Distributed Switch (VDS) ネットワークを使用している場合は、vSphere with Tanzu 環境に NSX Advanced Load Balancer 20.1.7 をインストールして、構成することができます。

NSX Advanced Load Balancer コンポーネント

NSX Advanced Load Balancer（別名、Avi Load Balancer）のコンポーネントには、Avi Controller クラスタ、サービス エンジン（データ プレーン）仮想マシン、および Avi Kubernetes Operator (AKO) などがあります。

コントローラ

Avi Controller（別名、コントローラ）は vCenter Server と連携して Tanzu Kubernetes クラスタのロード バランシングを自動実行します。コントローラは、サービス エンジンのプロビジョニング、サービス エンジン間でのリソースの調整、サービス エンジンのメトリックとログの集計を行います。また、ユーザー操作およびプログラムによる連携のための Web インターフェイス、コマンドライン インターフェイス、および API を提供します。

vSphere でコントローラ仮想マシンをデプロイして構成した後、HA 用の制御プレーン クラスタを設定する方法については、コントローラ クラスタのデプロイを参照してください。

サービス エンジン

サービス エンジンとも呼ばれる Avi サービス エンジンは、データ プレーン仮想マシンです。サービス エンジンは 1 つ以上の仮想サービスを実行します。サービス エンジンはコントローラによって管理されます。コントローラは、仮想サービスをホストするようにサービス エンジンをプロビジョニングします。

サービス エンジンには、次の 2 種類のネットワーク インターフェイスがあります。

- 仮想マシンの最初のネットワーク インターフェイス `vnic0` は管理ネットワークに接続され、そこから Avi Controller に接続することができます。
- もう一方のインターフェイス `vnic1 - 8` は、仮想サービスが実行されるワークロード ネットワークに接続されます。

サービス エンジン インターフェイスは、適切な Distributed Switch ポート グループに自動的に接続します。未使用のインターフェイスは自動的に作成されるポート グループ `Avi Internal` に接続され、今後使用するために予約されます。各サービス エンジンは、最大 1,000 個の仮想サービスをサポートできます。

仮想サービスは、Tanzu Kubernetes クラスタ ワークロード用のレイヤー 4 およびレイヤー 7 ロード バランシング サービスを提供します。仮想サービスは、1 つの仮想 IP アドレスと複数のポートで構成されます。仮想サービスをデプロイすると、コントローラによって ESX サーバが自動的に選択され、サービス エンジンが起動して適切なネットワーク（ポート グループ）に接続します。

最初のサービス エンジンは、最初の仮想サービスが構成された後にのみ作成されます。以降に構成された仮想サービスは、既存のサービス エンジンを使用します。

各仮想サーバは、Tanzu Kubernetes クラスタのロード バランサ タイプの異なる IP アドレスを持つレイヤー 4 ロード バランサを公開します。各仮想サーバに割り当てられる IP アドレスは、構成時にコントローラに指定する IP アドレス ブロックから選択されます。

Avi には、ネイティブ IP アドレス管理と外部 IP アドレス管理プロバイダのサポートが付属しています。vSphere では、Avi のネイティブ IP アドレス管理が利用されます。

Avi Kubernetes オペレータ

Avi Kubernetes Operator (AKO) は Kubernetes リソースを監視し、コントローラと通信して、対応するロード バランシング リソースを要求します。

Avi Kubernetes Operator は、有効化のプロセスでスーパーバイザー クラスタにインストールされます。

vSphere ネットワークと NSX Advanced Load Balancer を使用して vSphere with Tanzu をセットアップするためのシステム要件

Avi Load Balancer とも呼ばれる NSX Advanced Load Balancer を使用して vSphere with Tanzu を構成するには、使用環境が特定の要件を満たしている必要があります。vSphere with Tanzu では Avi ネットワークに対して複数のトポロジ（Avi サービス エンジンおよびロード バランサ サービス用の単一の Distributed Switch ネットワーク、Avi 管理プレーン用の Distributed Switch および NSX Advanced Load Balancer 用の別の Distributed Switch）をサポートしています。

ワークロード ネットワーク

vSphere ネットワーク スタックを使用した スーパーバイザー クラスタ を構成するには、クラスタ内のすべてのホストを vSphere Distributed Switch に接続する必要があります。スーパーバイザー クラスタ 用に実装するトポロジに応じて、1 つ以上の分散ポート グループを作成します。作成したポート グループを、ワークロード ネットワークとして、vSphere 名前空間 に指定します。

ホストを スーパーバイザー クラスタ に追加する前に、クラスタに含まれるすべての vSphere Distributed Switch にホストを追加する必要があります。

ワークロード ネットワークは、Tanzu Kubernetes クラスタのノードと、スーパーバイザー クラスタ 制御プレーン仮想マシンへの接続を提供します。Kubernetes 制御プレーン仮想マシンに接続を提供するワークロード ネットワークは、プライマリ ワークロード ネットワークと呼ばれます。各 スーパーバイザー クラスタ にそれぞれ 1 つのプライマリ ワークロード ネットワークが必要です。スーパーバイザー クラスタ に対して、分散ポート グループの 1 つをプライマリ ワークロード ネットワークとして指定する必要があります。

スーパーバイザー クラスタ 上の Kubernetes 制御プレーン仮想マシンは、プライマリ ワークロード ネットワークに割り当てられた IP アドレス範囲から 3 つの IP アドレスを使用します。Tanzu Kubernetes クラスタの各ノードには、この Tanzu Kubernetes クラスタが実行されている名前空間で構成されたワークロード ネットワークのアドレス範囲から割り当てられた、それぞれ別の IP アドレスが割り当てられます。

ネットワーク要件

NSX Advanced Load Balancer には、次の 2 つのルーティング可能なサブネットが必要です。

- 管理ネットワーク。管理ネットワークには、Avi Controller（別名、コントローラ）が配置されています。管理ネットワークは、コントローラに vCenter Server、ESXi ホスト、および スーパーバイザー クラスタ 制御プレーン ノードへの接続を提供します。このネットワークには、Avi サービス エンジンの管理インターフェイスが配置されます。このネットワークには、vSphere Distributed Switch (VDS) と分散ポート グループが必要です。
- データ ネットワーク。Avi サービス エンジン（別名、サービス エンジン）のデータ インターフェイスはこのネットワークに接続されます。ロード バランサの仮想 IP アドレス (VIP) は、このネットワークから割り当てられます。このネットワークには、vSphere Distributed Switch (VDS) と分散ポート グループが必要です。ロード バランサをインストールする前に、Distributed Switch とポート グループを構成する必要があります。

IP アドレスの割り当て

コントローラとサービス エンジンは管理ネットワークに接続されます。NSX Advanced Load Balancer をインストールして構成する際に、各コントローラ仮想マシンのルーティング可能な固定 IP アドレスを指定します。

サービス エンジンでは DHCP を使用できます。DHCP を使用できない場合は、サービス エンジンの IP アドレス プールを構成できます。

詳細については、『デフォルト ゲートウェイの構成』を参照してください。

コンピューティングの最小要件

次の表に、NSX Advanced Load Balancer を使用した vSphere ネットワークのコンピューティングの最小要件を示します。

注意： スーパーバイザー クラスタ を構成した後に vSphere DRS を無効にしないでください。DRS を常に有効にすることは、スーパーバイザー クラスタ でワークロードを実行するための必須の前提条件です。Tanzu Kubernetes DRS を無効にすると、クラスタが破損します。

表 4-5. コンピューティングの最小要件

システム	最小デプロイ サイズ	CPU	メモリ	ストレージ
vCenter Server 7.0、7.0.2、7.0.3	小	2	16 GB	290 GB
ESXi ホスト 7.0	3 台の ESXi ホストと、ホストあたり 1 つの固定 IP アドレス。 vSAN を使用している場合は、物理 NIC が 2 つ以上ある ESXi ホストが最低でも 3 台必要です。ただし、パッチ適用やアップグレード時の回復性のために、4 台の ESXi ホストが推奨されます。 ホストは、vSphere DRS と HA が有効になっているクラスタに参加している必要があります。vSphere DRS は、完全自動化モードまたは一部自動化モードになっている必要があります。 **注：** クラスタに参加するホストの名前に小文字が使用されていることを確認します。この条件に該当しない場合は、ワークロード管理のためのクラスタの有効化が失敗する場合があります。	8	ホストあたり 64 GB	該当なし
Kubernetes 制御プレーンの仮想マシン	3	4	16 GB	16 GB

表 4-5. コンピューティングの最小要件 （続き）

システム	最小デプロイ サイズ	CPU	メモリ	ストレージ
Avi Controller	Essentials	4	12 GB	128 GB
	Enterprise	8	24 GB	128 GB
	注： 小規模環境では、Essentials サイズのコントローラを単一コントローラ ノードとしてデプロイできます。Avi Controller クラスタを作成することはできますが、パフォーマンス上のメリットはなく、リソース使用率を低くするという目的を達成することはできません。この場合のディザスタ リカバリには、リモート バックアップを使用します。このサイズは Avi Essentials ライセンス モードの場合にのみ使用する必要があり、仮想サービス数は 50、サービス エンジン数は 10 に制限されます。			
	本番環境では、3 台の Avi Controller 仮想マシンによるクラスタをインストールすることを推奨します。HA 構成にするには、2 台以上のサービス エンジン仮想マシンが必要です。			
サービス エンジン	HA 構成にするには、2 台以上のサービス エンジン仮想マシンが必要です。	1	2 GB	15 GB

ネットワークの最小要件

次の表に、NSX Advanced Load Balancer を使用した vSphere ネットワークの最小ネットワーク要件を示します。

注： vSphere 7 スーパーバイザー クラスタ による IPv6 クラスタの作成や、Tanzu Mission Control による IPv6 クラスタの登録はできません。NSX Advanced Load Balancer サービスは現在 IPv6 をサポートしていません。

表 4-6. ネットワークの最小要件

コンポーネント	最小数	必要な構成
Kubernetes 制御プレーン仮想マシンの固定 IP アドレス	5 つのアドレスのブロック	管理ネットワークから スーパーバイザー クラスタ 内の Kubernetes 制御プレーン仮想マシンに割り当てられる、連続する 5 つの固定 IP アドレスのブロック。
管理トラフィック ネットワーク	1	ESXi ホスト、vCenter Server、スーパーバイザー クラスタ、およびロード バランサにルーティング可能な管理ネットワーク。イメージ レジストリが外部ネットワーク上にある場合、ネットワークはイメージ レジストリにアクセス可能で、インターネットに接続できる必要があります。イメージ レジストリは、DNS で解決できる必要があります。

表 4-6. ネットワークの最小要件 （続き）

コンポーネント	最小数	必要な構成
vSphere Distributed Switch 7.0 以降	1	クラスタのすべてのホストが、vSphere Distributed Switch に接続されている必要があります。
ワークロード ネットワーク	1	プライマリ ワークロード ネットワークとして構成する vSphere Distributed Switch には、1 つ以上の分散ポート グループを作成する必要があります。選択したトポロジによっては、名前空間のワークロード ネットワークと同じ分散ポート グループを使用することや、追加のポート グループを作成してワークロード ネットワークとして構成することができます。ワークロード ネットワークは次の要件を満たす必要があります。 ■ Tanzu Kubernetes クラスタのトラフィックに使用されるワークロード ネットワークが、相互に、および スーパーバイザー クラスタ プライマリ ワークロード ネットワークとの間でルーティングできること。 ■ 任意のワークロード ネットワークと、NSX Advanced Load Balancer が仮想 IP アドレスの割り当てに使用するネットワークとの間でルーティングできること。 ■ スーパーバイザー クラスタ 内のすべてのワークロード ネットワークで IP アドレス範囲の重複がないこと。
NTP サーバおよび DNS サーバ	1	vCenter Server で使用できる DNS サーバおよび NTP サーバ。 **注：** すべての ESXi ホストおよび vCenter Server で NTP を構成します。
DHCP サーバ	1	オプション。管理ネットワークとワークロード ネットワークの IP アドレスおよびフローティング IP アドレスを自動的に取得するように DHCP サーバを構成します。DHCP サーバはクライアント識別子をサポートし、互換性のある DNS サーバ、DNS 検索ドメイン、および NTP サーバを提供する必要があります。 管理ネットワークの場合、制御プレーン仮想マシンの IP アドレス、フローティング IP アドレス、DNS サーバ、DNS、検索ドメイン、NTP サーバなどのすべての IP アドレスは、DHCP サーバから自動的に取得されます。 DHCP 構成は、スーパーバイザー クラスタ で使用されます。ロード バランサで管理を行うには、固定 IP アドレスが必要になる場合があります。DHCP スコープは、これらの固定 IP アドレスと重複しないようにしてください。DHCP は仮想 IP アドレスには使用されません。(VIP)
管理ネットワークのサブネット	1	管理ネットワークには、Avi Controller（別名、コントローラ）が配置されています。 また、サービス エンジン管理インターフェイスも接続されます。Avi Controller は、このネットワーク内の vCenter Server および ESXi 管理 IP アドレスに接続する必要があります **注：** 管理ネットワークとワークロード ネットワークは異なるサブネット上に配置する必要があります。管理ネットワークとワークロード ネットワークに同じサブネットを割り当てることはできないため、システム エラーや問題が発生することがあります。

表 4-6. ネットワークの最小要件 （続き）

コンポーネント	最小数	必要な構成
データ ネットワークのサブネット	1	Avi サービス エンジン（別名、サービス エンジン）のデータ インターフェイスはこのネットワークに接続されます。サービス エンジンの IP アドレス プールを構成します。ロード バランサの仮想 IP アドレス (VIP) は、このネットワークから割り当てられます。
物理ネットワークの MTU	1500	vSphere Distributed Switch ポート グループの MTU サイズは 1500 以上にする必要があります。
vSphere ポッド CIDR 範囲	/24 プライベート IP アドレス	vSphere ポッド の IP アドレスを提供するプライベート CIDR 範囲。
Avi Controller の IP アドレス	1 または 4	Avi Controller を単一ノードとしてデプロイする場合、その管理インターフェイス用に 1 つの固定 IP アドレスが必要です。 3 ノード クラスタの場合は、4 つの IP アドレスが必要です。各 Avi Controller 仮想マシンに 1 つ、クラスタ仮想 IP アドレスに 1 つです。これらの IP アドレスは、管理ネットワーク サブネットから取得する必要があります。
VIP IP アドレス管理の範囲	-	Kubernetes サービスに IP アドレスを割り当てるためのプライベート CIDR 範囲。IP アドレスは、データ ネットワーク サブネットから取得する必要があります。スーパーバイザー クラスタごとに一意の Kubernetes サービス CIDR 範囲を指定する必要があります。
NTP サーバおよび DNS サーバ	1	Avi Controller で vCenter Server と ESXi のホスト名が正しく解決されるようにするには、DNS サーバの IP アドレスが必要です。 パブリック NTP サーバはデフォルトで使用されるため、NTP は省略可能です。

ポートとプロトコル

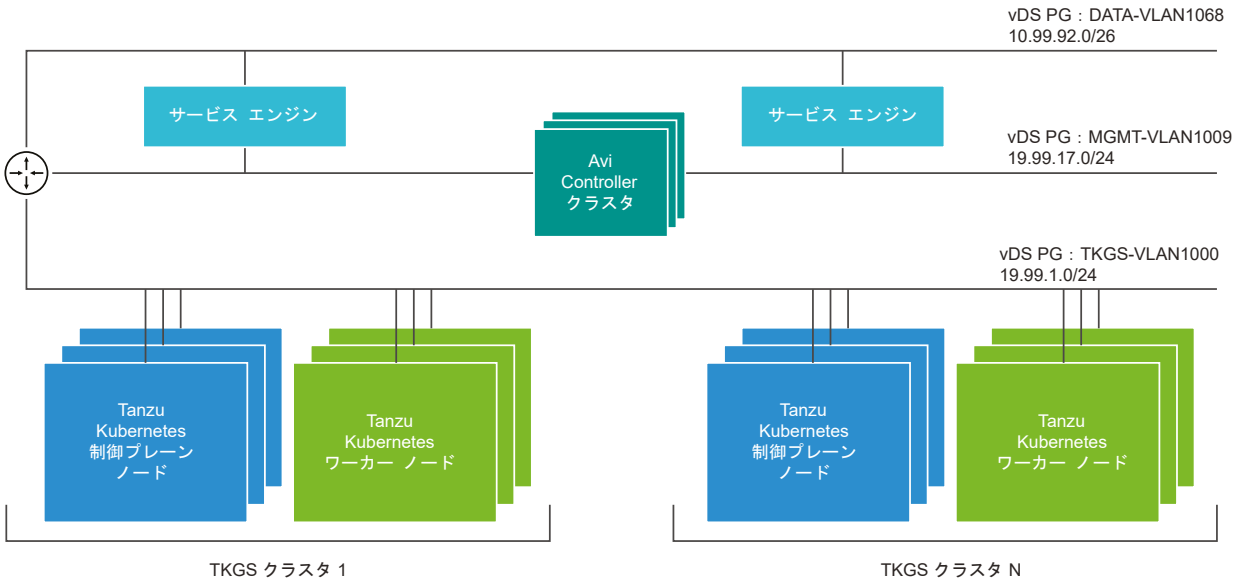
次の表に、NSX Advanced Load Balancer、vCenter Server とその他の vSphere with Tanzu コンポーネント間の IP 接続を管理するために必要なプロトコルとポートを示します。

ソース	ターゲット	プロトコルとポート
Avi Controller	Avi Controller（クラスタ内）	TCP 22 (SSH) TCP 443 (HTTPS) TCP 8443 (HTTPS)
サービス エンジン	HA のサービス エンジン	TCP 9001（VMware、LSC、NSX-T クラウド用）
サービス エンジン	Avi Controller	TCP 22 (SSH) TCP 8443 (HTTPS) UDP 123 (NTP)
Avi Controller	vCenter Server、ESXi、NSX-T Manager	TCP 443 (HTTPS)
スーパーバイザー制御プレーンノード (AKO)	Avi Controller	TCP 443 (HTTPS)

NSX Advanced Load Balancer のポートとプロトコルの詳細については、https://ports.esp.vmware.com/home/NSX-Advanced-Load-Balancer を参照してください。

vSphere ネットワークと NSX Advanced Load Balancer を使用した スーパーバイザー クラスタ のトポロジ

Avi Controller は常に管理ネットワークにデプロイされ、vCenter Server、ESXi ホスト、および スーパーバイザー クラスタ 制御プレーン ノードとのインターフェイスを行うことができます。サービス エンジンは、管理ネットワークおよびデータ ネットワークへのインターフェイスを使用してデプロイされます。



`MGMT-VLAN1009` などの管理ネットワークには Avi Controller が配置され、サービスエンジンの管理インターフェイスが接続されます。

`DATA-VLAN1068` などのデータ ネットワークには、仮想 IP アドレスを配置するためのサービス エンジン インターフェイスが接続されます。クライアント トラフィックが VIP に到達すると、サービス エンジンはこのネットワークを介してワークロード ネットワーク IP アドレスへのトラフィックをロード バランシングします。

`TKGS-VLAN1000` などのワークロード ネットワークでは、Tanzu Kubernetes クラスタが実行されます。サービス エンジンは、ワークロード ネットワークへのインターフェイスを必要としません。

サービス エンジンはワンアーム モードで実行されます。ロード バランシングされたトラフィックは、サービス エンジンによってルーター経由でワークロード ネットワークにルーティングされます。サービス エンジンは、データ ネットワークの DHCP からデフォルト ゲートウェイの IP アドレスを取得しません。サービス エンジンがワークロード ネットワークおよびクライアント IP アドレスにトラフィックを適切にルーティングできるように、スタティック ルートを構成する必要があります。スタティック ルートの構成の詳細については、デフォルト ゲートウェイの構成を参照してください。

このトポロジでは、サービス エンジンを単一のネットワークに配置できます。複数のワークロード ネットワークがある場合は、ロード バランシング サービスを提供できます。サービス エンジンの作成とネットワーク接続は、Avi Controller によって自動実行されます。

NSX Advanced Load Balancer のインストールと構成

vSphere Distributed Switch (VDS) ネットワークを使用している場合は、vSphere with Tanzu 環境に NSX Advanced Load Balancer 20.1.7 をインストールして、構成することができます。

- 環境が、NSX Advanced Load Balancer を使用して vSphere with Tanzu を構成するための要件を満たしていることを確認します。vSphere ネットワークと NSX Advanced Load Balancer を使用して vSphere with Tanzu をセットアップするためのシステム要件を参照してください。
- NSX Advanced Load Balancer OVA をダウンロードします。VMware は、ワークロード管理を有効にする vSphere 環境にデプロイされる NSX Advanced Load Balancer OVA ファイルを提供しています。製品のダウンロード ページから、vSphere with Tanzu でサポートされている最新バージョンの OVA ファイルをダウンロードします。

NSX Advanced Load Balancer で使用する スーパーバイザー クラスタ の vSphere Distributed Switch の作成

vSphere ネットワーク スタックおよび NSX Advanced Load Balancer を使用するスーパーバイザー クラスタとして vSphere クラスタを構成するには、vSphere Distributed Switch を作成する必要があります。Distributed Switch 上で スーパーバイザー クラスタ のワークロード ネットワークとして構成できるポート グループを作成します。NSX Advanced Load Balancer に Avi サービス エンジン データ インターフェイスを接続するには、分散ポート グループが必要です。分散ポート グループは、アプリケーションの仮想 IP アドレス (VIP) をサービス エンジンに配置するために使用されます。

前提条件

NSX Advanced Load Balancer で スーパーバイザー クラスタ の vSphere ネットワーク使用するためのシステム要件とネットワーク トポロジを確認します。vSphere ネットワークと NSX Advanced Load Balancer を使用して vSphere with Tanzu をセットアップするためのシステム要件を参照してください。

手順

1. vSphere Client で、データセンターに移動します。
2. データセンターを右クリックして、[Distributed Switch] - [新しい Distributed Switch] の順に選択します。
3. スイッチの名前（**ワークロード Distributed Switch** など）を入力して、[次へ] をクリックします。
4. スイッチのバージョン 7.0 を選択して、[次へ] をクリックします。
5. [ポート グループ名] に **プライマリ ワークロード ネットワーク** と入力し、[次へ] をクリックして [終了] をクリックします。
6. ワークロード ネットワークの分散ポート グループを作成します。
   a. 新しく作成した Distributed Switch に移動します。
   b. スイッチを右クリックして、[分散ポート グループ] - [新規分散ポート グループ] の順に選択します。
   c. ポート グループの名前（**ワークロード ネットワーク** など）を入力して、[次へ] をクリックします。
   d. デフォルトのままにして、[次へ] をクリックし、[終了] をクリックします。

7 データ ネットワークのポート グループを作成します。

a Distributed Switch を右クリックして、[分散ポート グループ] - [新規分散ポート グループ] を選択します。

b ポート グループの名前（**データ ネットワーク** など）を入力して、[次へ] をクリックします。

c [設定の構成] 画面で新規分散ポート グループの全般プロパティを入力し、[次へ] をクリックします。

プロパティ	説明
ポート バインド	この分散ポート グループに接続された仮想マシンにポートを割り当てるときに選択します。 [静的バインド] を選択すると、仮想マシンが分散ポート グループに接続されるときに仮想マシンにポートを割り当てます。
ポートの割り当て	ポート割り当てとして [弾性] を選択します。 デフォルトのポート数は 8 個です。すべてのポートが割り当てられたら、新しい 8 組のポートが作成されます。
ポート数	デフォルト値を保持します。
ネットワーク リソース プール	ドロップダウン メニューで、新しい分散ポート グループをユーザー定義のネットワーク リソース プールに割り当てます。ネットワーク リソース プールを作成していない場合、このメニューは空です。
VLAN	ドロップダウン メニューで VLAN トラフィックのフィルタリングおよびマーキングのタイプを選択します。 ■ [なし]： VLAN を使用しません。外部スイッチ タギングを使用している場合は、このオプションを選択します。 ■ [VLAN]：[VLAN ID] テキスト ボックスに、仮想スイッチ タギング用の値を 1 ～ 4,094 の範囲で入力します。 ■ [VLAN トランク]：仮想ゲスト タギングを行って、VLAN トラフィックに ID を設定してゲスト OS に送信するには、このオプションを使用します。VLAN トランク範囲を入力します。コンマ区切りリストを使用して複数の範囲や個々の VLAN を設定できます。たとえば、`1702-1705、1848-1849` です。 ■ [プライベート VLAN]：トラフィックと、Distributed Switch で作成されたプライベート VLAN を関連付けます。プライベート VLAN を作成していない場合、このメニューは空です。
詳細	このオプションは選択解除したままにします。

8 [設定の確認] 画面で構成を確認し、[完了] をクリックします。

結果

Distributed Switch が作成され、Distributed Switch の下に分散ポート グループが表示されます。以上で、作成したポート グループを NSX Advanced Load Balancer のデータ ネットワークとして使用できます。

コントローラのデプロイ

コントローラ仮想マシンを、vSphere with Tanzu 環境内の管理ネットワークにデプロイします。

前提条件

- NSX Advanced Load Balancer をデプロイする管理ネットワークがあることを確認します。これは、vSphere Distributed Switch (vDS) または vSphere 標準スイッチ (vSS) のいずれかです。
- データ ネットワーク用の Distributed Switch およびポートグループが作成されていることを確認します。NSX Advanced Load Balancer で使用する スーパーバイザー クラスタ の vSphere Distributed Switch の作成を参照してください。
- すべての前提条件を満たしていることを確認します。vSphere ネットワークと NSX Advanced Load Balancer を使用して vSphere with Tanzu をセットアップするためのシステム要件を参照してください。

手順

1 vSphere Client を使用して、vCenter Server にログインします。

2 管理コンポーネント用に指定されている vSphere クラスタを選択します。

3 **AVI-LB** という名前のリソース プールを作成します。

4 リソース プールを右クリックして、[OVF テンプレートのデプロイ] を選択します。

5 [ローカル ファイル] を選択し [ファイルのアップロード] をクリックします。

6 前提条件としてダウンロードした `controller-VERSION.ova` ファイルを参照して選択します。

7 名前を入力し、コントローラのフォルダを選択します。

オプション	説明
仮想マシン名	**avi-controller-1**
仮想マシンの場所	[Datacenter]

8 コンピューティング リソースとして **AVI-LB** リソース プール選択します。

9 構成の詳細を確認し、[次へ] をクリックします。

10 [仮想マシン ストレージ ポリシー] を選択します（**vsanDatastore** など）。

11 管理ネットワークを選択します（**MGMT-VLAN1009** など）。

12 次のように構成をカスタマイズして、完了したら [次へ] をクリックします。

オプション	説明
管理インターフェイスの IP アドレス	コントローラ仮想マシンの IP アドレスを入力します（**10.999.17.51** など）。
管理インターフェイスのサブネット マスク	サブネット マスクを入力します（**255.255.255.0** など）。
デフォルト ゲートウェイ	管理ネットワークのデフォルト ゲートウェイを入力します（**10.199.17.235** など）。
sysadmin ログイン認証キー	プライベート キーの内容を貼り付けます（オプション）。 これは、仮想マシンに SSH 接続するために必要なプライベート SSH キーです。OpenSSH または PuTTY を使用して作成できます。

13 展開設定を確認します。

14 [完了] をクリックして構成を完了します。

15 vSphere Client を使用して、[タスク] パネルでコントローラ仮想マシンのプロビジョニングを監視します。

16 コントローラ仮想マシンがデプロイされたら、vSphere Client を使用してパワーオンします。

コントローラのパワーオン

コントローラ仮想マシンをデプロイしたら、その仮想マシンをパワーオンできます。起動中に、デプロイ時に指定した IP アドレスが仮想マシンに割り当てられます。

パワーオン後、コントローラ仮想マシンの最初の起動プロセスが実行されるまで、最大 10 分かかる場合があります。

前提条件

コントローラをデプロイします。

手順

1 vCenter Server で、デプロイした `avi-controller-1` 仮想マシンを右クリックします。

2 [電源] - [パワーオン] の順に選択します。

仮想マシンに、デプロイ時に指定した IP アドレスが割り当てられます。

3 仮想マシンがパワーオン状態かどうかを確認するには、ブラウザで IP アドレスにアクセスします。

仮想マシンがオンラインになると、TLS 証明書と接続に関する警告が表示されます。

4 [接続はプライベートではありません] という警告で、[詳細を表示] をクリックします。

5 表示されるウィンドウで [この Web サイトを閲覧] をクリックします。

ユーザー認証情報を入力するように求められます。

コントローラの構成

コントローラ仮想マシンを vSphere with Tanzu 環境に構成します。

ロード バランサ制御プレーンを vCenter Server 環境に接続するには、コントローラにデプロイ後の構成パラメータをいくつか指定する必要があります。

前提条件

- 環境が NSX Advanced Load Balancer を構成するためのシステム要件を満たしていることを確認します。vSphere ネットワークと NSX Advanced Load Balancer を使用して vSphere with Tanzu をセットアップするためのシステム要件を参照してください。
- コントローラをデプロイします。『コントローラのデプロイ』を参照してください

手順

1 ブラウザを使用して、コントローラのデプロイ時に指定した IP アドレスに移動します。

2 [管理者アカウント] を作成します。

オプション	説明
ユーザー名	初期構成に使用する管理者のユーザー名。このフィールドは編集できません。
パスワード	コントローラ仮想マシンの管理者パスワードを入力します。 パスワードには、数字、特殊文字、大文字、小文字の組み合わせを含む 8 文字以上を指定する必要があります。
パスワードの確認	管理者パスワードを再度入力します。
メール アドレス（オプション）	管理者のメール アドレスを入力します。 本番環境でのパスワード回復用のメール アドレスを指定することを推奨します。

3 [システム設定] を構成します。

オプション	説明
パスフレーズ	コントローラ バックアップのパスフレーズを入力します。コントローラ構成は、定期的にローカル ディスクに自動的にバックアップされます。詳細については、バックアップとリストアを参照してください。 パスフレーズには、数字、特殊文字、大文字、小文字の組み合わせを含む 8 文字以上を指定する必要があります。
パスフレーズの確認	バックアップ パスフレーズを再度入力します。
DNS リゾルバ	Tanzu 環境の vSphere で使用している DNS サーバの IP アドレスを入力します。たとえば、`10.14.7.12` など。
DNS 検索ドメイン	ドメイン文字列を入力します。

4 （オプション） [E メール/SMTP] を構成します

オプション	説明
SMTP ソース	なし、ローカル ホスト、SMTP サーバ、または匿名サーバ
送信元アドレス	メール アドレス

5 マルチテナントを構成します。

a デフォルトのテナント アクセスを維持します。

b [以下の後にクラウドをセットアップ] を選択して、[保存] をクリックします。

> **注：** 保存する前に [以下の後にクラウドをセットアップ] オプションを選択しなかった場合は、初期構成ウィザードが終了します。クラウド構成ウィンドウは自動的に起動せず、コントローラのダッシュボード ビューに移動します。この場合は、[インフラストラクチャ] - [クラウド] の順に選択して、[Default-Cloud] を編集して、次の手順に進みます。

6 [Default-Cloud] を構成します。

a [クラウド] を選択します。

b インフラストラクチャ タイプとして [VMware vCenter/vSphere ESX] を選択します。

7 [インフラストラクチャ] を構成します。

[vCenter/vSphere ログイン] 情報を入力します。

オプション	説明
ユーザー名	vCenter Server 管理者ユーザー名を入力します（administrator@vsphere.local など）。 より少ない権限を使用するには、専用ロールを作成します。詳細については、VMware ユーザー ロールを参照してください。
パスワード	ユーザー パスワードを入力します。
vCenter Server アドレス	vSphere with Tanzu 環境の vCenter Server のホスト名または IP アドレスを入力します。
アクセス権	[読み取り]：サービス エンジン仮想マシンを作成および管理します。 [書き込み]：コントローラでサービス エンジン仮想マシンが作成および管理されます。 [書き込み] を選択する必要があります。

IP アドレス管理プロファイルと DNS プロファイルは空のままにすることができます。

8 [データセンター] を設定します。

a [ワークロード管理] を有効にする vSphere [データセンター] を選択します。

b [デフォルトのネットワーク IP アドレス管理] モードを選択します。

- vSphere ポート グループで DHCP が使用可能な場合は、[DHCP 有効] を選択します。
- サービス エンジン インターフェイスで固定 IP アドレスのみを使用する場合は、このオプションを選択解除したままにします。これは、ネットワークごとに個別に構成できます。

  詳細については、『仮想 IP ネットワークの構成』を参照してください。

c [仮想サービス配置設定] を構成します。

オプション	説明
仮想サービス配置用の直接接続ネットワークより固定ルートを優先	サービス エンジン仮想マシンからサーバ ネットワークにアクセスする際にデフォルト ゲートウェイを経由するように設定するには、このオプションを選択します。 デフォルトでは、コントローラは NIC をサーバ ネットワークに直接接続します。ユーザーはサービス エンジンをデータ ネットワークにのみ強制的に接続し、ワークロード ネットワークにルーティングする必要があります。
仮想サービス配置用の VIP のネットワーク解決にスタティック ルートを使用	このオプションは選択解除したままにします。

9 [ネットワーク] 設定を構成し、[保存] をクリックします。

オプション	説明
管理ネットワーク	管理ネットワークを選択します。このネットワーク インターフェイスは、サービス エンジンがコントローラに接続するために使用されます。たとえば、`Primary Workload Network` など。
サービス エンジン	[テンプレート サービス エンジン グループ] は空のままにします。
管理ネットワーク IP アドレス管理	[DHCP 有効] を選択します。

10 （オプション） [DHCP 有効] を選択しなかった場合のみ、次のネットワーク設定を構成します。

オプション	説明
IP サブネット	管理ネットワークの IP サブネットを入力します。たとえば、`192.168.110.0/24` です。 **注：** DHCP を使用できない場合にのみ、IP サブネットを入力します。
固定 IP アドレス プールの追加	1 つ以上の IP アドレスまたは IP アドレス範囲を入力します。たとえば、`192.168.110.66-192.168.110.90` です。 **注：** DHCP を使用できない場合にのみ、IP サブネットを入力します。
デフォルト ゲートウェイ	管理ネットワークのデフォルト ゲートウェイを入力します（`192.168.110.1` など）。 **注：** DHCP を使用できない場合にのみ、IP サブネットを入力します。

11 （オプション） 内部 NTP サーバを使用する場合は、NTP 設定を構成します。

a [管理] - [設定] - [DNS/NTP] の順に選択します。

b 既存の NTP サーバがある場合は削除し、使用している DNS サーバの IP アドレスを入力します。たとえば、`192.168.100.1` のように入力します。

結果

構成が完了すると、コントローラ [ダッシュボード] が表示されます。[インフラストラクチャ] - [クラウド] の順に選択し、[Default-Cloud] のコントローラのステータスが緑であることを確認します。Avi Controller が vCenter Server 環境内のすべてのポート グループを検出してステータスが緑になるまで、しばらくの間、ステータスが黄色になることがあります。

ライセンスの追加

NSX Advanced Load Balancer を構成したら、ライセンスを追加する必要があります。コントローラは評価モードで起動します。評価モードでは、Enterprise Edition ライセンスに相当するすべての機能を使用できます。評価期間が終了する前に、有効な Enterprise ライセンスをコントローラに割り当てる必要があります。

前提条件

Enterprise ライセンスを持っていることを確認します。

手順

1 [Avi Controller] ダッシュボードで、左上隅のメニューをクリックし、[管理] を選択します。

2 [設定] - [ライセンス] の順に選択します。

3 ライセンスを追加するには、[コンピュータからアップロード] を選択します。

ライセンス ファイルをアップロードすると、コントローラのライセンス リストに表示されます。開始日や有効期限などのライセンスに関する情報が表示されます。

4 （オプション） Enterprise ライセンスを持っていない場合は、Essentials エディションを使用できます。

注： Enterprise モードまたは評価モードから Essentials エディションに切り替える場合は、Avi Controller を構成する前に切り替える必要があります。Enterprise エディションの機能が構成されている場合は、Essentials エディションに切り替える前に、構成された設定を削除する必要があります。

a [設定] - [ライセンス] の順に選択します。

b [ライセンス] の横にある歯車アイコンをクリックします。

c [Essential ライセンス] を選択し、[保存] をクリックします。

d ポップアップ ウィンドウで [はい] を選択して、エディションを確認します。

構成を保存して、Avi Controller が Enterprise 機能を無効にするまでに時間がかかることがあります。

コントローラ クラスタのデプロイ

必要に応じて、3 台のコントローラ ノードからなるクラスタをデプロイできます。HA およびディザスタ リカバリ用に、本番環境内にクラスタを構成することを推奨します。単一ノードの Avi Contoller を実行している場合は、バックアップとリストア機能を使用する必要があります。

3 ノード クラスタを実行するには、最初のコントローラ仮想マシンをデプロイした後、さらに 2 台のコントローラ仮想マシンをデプロイしてパワーオンします。初期構成ウィザードの実行や、これらのコントローラの管理者パスワードの変更が不要になります。最初のコントローラ仮想マシンの構成が、新しい 2 台のコントローラ仮想マシンに割り当てられます。

手順

1 [管理] - [コントローラ] の順に選択します。

2 [ノード] を選択します。

3 [編集] アイコンをクリックします。

4 [コントローラ クラスタ IP アドレス] に固定 IP アドレスを追加します。

この IP アドレスは、管理ネットワークから取得する必要があります。

5 [クラスタ ノード] で、新しい 2 台のクラスタ ノードを構成します。

オプション	説明
IP	コントローラ ノードの IP アドレス。
名前	ノードの名前。名前に IP アドレスを指定できます。
パスワード	コントローラ ノードのパスワード。パスワードは空のままにします。
パブリック IP アドレス	コントローラ ノードのパブリック IP アドレス。空のままにします。

6 [保存] をクリックします。

注： クラスタをデプロイした後の構成には、コントローラ ノードの IP アドレスではなく、コントローラ クラスタの IP アドレスを使用する必要があります。

コントローラへの証明書の割り当て

コントローラは、安全な通信を確立するために証明書をクライアントに送信する必要があります。この証明書には、Avi Controller クラスタのホスト名または IP アドレスと一致する Subject Alternative Name (SAN) が必要です。

コントローラにはデフォルトの自己署名の証明書があります。ただし、この証明書には正しい SAN がありません。正しい SAN を持つ有効な証明書または自己署名証明書に置き換える必要があります。自己署名証明書を作成するか、外部証明書をアップロードします。

証明書の詳細については、Avi のドキュメントを参照してください。

手順

1 Avi Controller ダッシュボードで、左上隅にあるメニューをクリックして、[テンプレート] - [セキュリティ] の順に選択します。

2 [SSL/TLS 証明書] を選択します。

3 証明書を作成するには、[作成] をクリックし、[コントローラ証明書] を選択します。

[新しい証明書 (SSL/TLS)] ウィンドウが表示されます。

4 証明書の名前を入力します。

5 事前に作成された有効な証明書がない場合は、[タイプ] に `Self Signed` を選択して自己署名証明書を追加します。

a 次の詳細を入力します。

オプション	説明
共通名	サイトの完全修飾名を指定します。サイトが信頼されていると見なされるには、このエントリがクライアントのブラウザに入力されたホスト名と一致する必要があります。
サブジェクトの代替名 (SAN)	Avi Controller が単一ノードとしてデプロイされている場合は、クラスタの IP アドレスまたは FQDN、あるいは両方を入力します。 IP アドレスまたは FQDN のみが使用されている場合は、これがデプロイ時に指定したコントローラ仮想マシンの IP アドレスと一致する必要があります。コントローラのデプロイを参照してください。 Avi Controller クラスタが 3 ノードのクラスタとしてデプロイされている場合は、クラスタの IP アドレスまたは FQDN を入力します。3 コントローラ ノードで構成するクラスタのデプロイの詳細については、コントローラ クラスタのデプロイを参照してください。
アルゴリズム	EC（楕円曲線暗号）または RSA を選択します。EC が推奨です。
キーのサイズ	ハンドシェイクに使用する暗号化のレベルを選択します。 ■ `SECP256R1` は EC 証明書に使用されます。 ■ RSA 証明書には 2048 ビットが推奨です。

b [保存] をクリックします。

この証明書は、スーパーバイザー クラスタでワークロード管理機能を有効にするように構成する場合に必要になります。

6 作成した自己署名証明書をダウンロードします。

a [セキュリティ] - [SSL/TLS 証明書] の順に選択します。

証明書が表示されない場合は、ページを更新します。

b 作成した証明書を選択し、ダウンロード アイコンをクリックします。

c 表示された [証明書のエクスポート] 画面で、証明書に対して [クリップボードにコピー] をクリックします。キーをコピーしないでください。

d 後でワークロード管理の有効化の際に使用するためにコピーした証明書を保存します。

7 事前に作成された有効な証明書がある場合は、[タイプ] に `Import` を選択してアップロードします。

a [証明書] で [ファイルのアップロード] をクリックして、証明書をインポートします。

アップロードする証明書の SAN フィールドには、コントローラのクラスタ IP アドレスまたは FQDN が必要です。

注： 証明書の内容をアップロードまたは貼り付けるのは、必ず 1 回だけにしてください。

b [キー (PEM) または PKCS12] で [ファイルのアップロード] をクリックして、キーをインポートします。

c [検証] をクリックして、証明書とキーを検証します。

d [保存] をクリックします。

8 ポータル証明書を変更するには、次の手順を実行します。

a Avi Controller ダッシュボードで、左上隅にあるメニューをクリックして、[管理] - [設定] の順に選択します。

b [アクセス設定] を選択します。

c [編集] アイコンをクリックします。

d [SSL/TLS 証明書] で、既存のデフォルト ポータル証明書を削除します。

e ドロップダウンで、新しく作成した証明書またはアップロードした証明書を選択します。

f [保存] をクリックします。

サービス エンジン グループの構成

vSphere with Tanzu では、サービス エンジン グループとして [Default-Group] が使用されます。必要に応じて、グループ内に [Default-Group] サービス エンジンを構成して、vCenter Server 内のサービス エンジン仮想マシンの配置と数を定義することができます。Avi Controller が Enterprise モードになっている場合は、高可用性を構成することもできます。

フェイルオーバーの場合に十分なキャパシティをプロビジョニングする方法については、『Avi ドキュメント』を参照してください。

手順

1 Avi コントローラ ダッシュボードの左上隅にあるメニューをクリックし、最初に [インフラストラクチャ] を選択してから、[Cloud リソース] を選択します。

2 設定ページで、[サービス エンジン グループ] をクリックします。

3 [サービス エンジン グループ] ページで、[Default-Group] の編集アイコンをクリックします。

[基本設定] ページが表示されます。

4 [高可用性 & 配置設定] セクションで、[高可用性モード] を選択します。

Essentials ライセンスを使用したデフォルトのオプションは、`Active/Standby` です。Enterprise ライセンスを使用する場合、`Elastic HA N + M Mode` または `Elastic HA Active/Active Mode` モードを構成することもできます。

5 [詳細] タブをクリックします。

6 （オプション）[ホスト & データ ストア範囲] セクションで、以下の設定を構成します。

a [含む] をクリックして、[クラスタ] のリストから vSphere クラスタを選択します。

b [含む] をクリックして、[ホスト] のリストから vSphere クラスタを選択します。

7 （オプション）[詳細な HA と配置] セクションでサービス エンジン グループに十分なキャパシティを構成できるのは、Enterprise ライセンスを使用する場合のみです。

十分なキャパシティを構成するには、[バッファ サービス エンジン] で値を指定します。指定する値は、フェイルオーバーの際に十分なキャパシティを確保するためにデプロイされる仮想マシンの数です。

たとえば、値を `0` に設定します。

8 [保存] をクリックします。

仮想 IP ネットワークの構成

データ ネットワークの仮想 IP アドレス (VIP) サブネットを構成します。仮想サービスが特定の VIP ネットワークに配置されている場合に使用する VIP 範囲を構成できます。サービス エンジンに対して DHCP を構成できます。DHCP が使用できない場合は、必要に応じて、該当するネットワーク上のサービス エンジン インターフェイスに割り当てられる IP アドレス プールを構成できます。

手順

1 Avi Controller ダッシュボードで、左上隅にあるメニューをクリックして、[インフラストラクチャ] を選択します。

2 [ネットワーク] をクリックして、vCenter Server 上のネットワークのリストを表示します。

3 仮想 IP アドレスを提供するデータ ネットワークを特定し、編集アイコンをクリックしてネットワーク設定を編集します。

たとえば、`Data Nework` など。

4 データ ネットワークで DHCP が使用可能な場合は、[DHCP 有効] を選択したままにします。

DHCP を使用できない場合は、このオプションを選択解除します。

5 [仮想サービス配置に対し検出されたサブネットの除外] を選択解除します。このオプションを選択解除すると、仮想 IP アドレス配置に構成されているサブネットを使用できます。

仮想マシンがネットワーク上で実行されていて、タイプが [検出済み] と表示される場合、Avi Controller はネットワークの CIDR を自動的に検出します。

6 Avi Controller が IP サブネットを自動的に検出する場合は、サブネットの IP アドレス範囲を構成します。

a 検出されたネットワークの [編集] アイコンをクリックします。

b [固定 IP アドレス プールの追加] を選択します。

c 1 つ以上の IP アドレスまたは IP アドレス範囲を入力します。

たとえば、`10.202.35.1-10.202.35.254` など。

注： `0` で終わる IP アドレスを入力できます。たとえば、`192.168.0.0` です。表示される警告は無視してください。

d サービス エンジンの IP アドレスに DHCP を使用できる場合は、[VIP と SE に固定 IP アドレスを使用] を選択解除し、[VIP に使用] を選択します。

e [保存] をクリックします。

7 コントローラで IP サブネットとそのタイプが検出されない場合は、次の手順を実行します。

a [サブネットの追加] をクリックします。

b [IP サブネット] で、仮想 IP アドレスを提供するネットワークの CIDR を入力します。

たとえば、`10.202.35.0/22`。

c [固定 IP アドレス プールの追加] を選択します。

d 1 つ以上の IP アドレスまたは IP アドレス範囲を入力します。

範囲は [IP サブネット] のネットワーク CIDR のサブネットである必要があります。たとえば、`10.202.35.1-10.202.35.254` など。

注： `0` で終わる IP アドレスを入力できます。たとえば、`192.168.0.0` です。表示される警告は無視してください。

e サービス エンジンの IP アドレスに DHCP を使用できる場合は、[VIP と SE に固定 IP アドレスを使用] を選択解除し、[VIP に使用] を選択します。

f [保存] をクリックして、サブネットの構成を保存します。

[ネットワーク設定の編集] 画面に、タイプが [構成済み] の IP サブネットと IP アドレス プールが一覧表示されます。

8 [保存] をクリックして、ネットワーク設定を保存します。

結果

[ネットワーク] 画面には、構成済みのネットワークが一覧表示されます。

例

`Primary Workload Network` ネットワークには、検出されたネットワークが `10.202.32.0/22` として、構成されたサブネットが `10.202.32.0/22 [254/254]` としてそれぞれ表示されます。これは、254 の仮想 IP アドレスが `10.202.32.0/22` からのものであることを示します。サマリ ビューには、IP アドレス範囲 `10.202.35.1-10.202.35.254` は一覧表示されません。

デフォルト ゲートウェイの構成

デフォルト ゲートウェイを構成すると、サービス エンジンはワークロード ネットワーク上のプール サーバにトラフィックをルーティングできるようになります。データ ネットワーク ゲートウェイの IP アドレスをデフォルト ゲートウェイとして構成する必要があります。

手順

1. Avi Controller ダッシュボードで、左上隅にあるメニューをクリックして、[インフラストラクチャ] を選択します。
2. [ルーティング] をクリックします。
3. [固定ルート] セクションで、[作成] をクリックします。
4. [ゲートウェイ サブネット] に `0.0.0.0/0` と入力します。
5. [ネクスト ホップ] に、データ ネットワークのゲートウェイ IP アドレスを入力します。

   たとえば、`192.168.0.1` です。
6. [保存] をクリックします。

IP アドレス管理の構成

IP アドレス管理をコントローラに対して構成し、それを Default-Cloud 構成に割り当てます。現在、Default-Cloud 構成のみがサポートされています。

仮想サービスの作成時に仮想 IP アドレスを割り当てるには、IP アドレス管理が必要です。

手順

1. Avi Controller ダッシュボードで、[テンプレート] - [プロファイル] - [IP アドレス管理/DNS プロファイル] の順に選択します。
2. [作成] をクリックし、ドロップダウン メニューで [IP アドレス管理プロファイル] を選択します。
3. [IP アドレス管理プロファイル] を構成します。

オプション	説明
名前	ユーザー定義の文字列（**ipam-profile** など）
タイプ	[AVI Vantage の IP アドレス管理] を選択します
VRF での IP アドレスの割り当て	このオプションを選択解除します。

4 [使用可能なネットワークの追加] をクリックして、構成します。

オプション	説明
使用可能なネットワーク用クラウド	[Default-Cloud]
使用可能なネットワーク	構成した仮想 IP ネットワークを選択します。

5 [保存] をクリックします。

ipam-profile が [IP アドレス管理/DNS プロファイル] 画面に一覧表示されます。

6 IP アドレス管理を [Default-Cloud] 構成に割り当てます。

a [インフラストラクチャ] - [クラウド] の順に移動します。

b [Default-Cloud] 構成を編集します。

[IP アドレス管理プロファイル]：**ipam-profile**

c その他すべての値はデフォルトのままにします。

d [保存] をクリックします。

NSX Advanced Load Balancer のテスト

NSX Advanced Load Balancer 制御プレーンをデプロイして構成したら、その機能を確認します。

手順

1 Avi Controller ダッシュボードで、[インフラストラクチャ] - [クラウド] の順に移動します。

2 [デフォルト クラウド] のコントローラのステータスが緑色であることを確認します。

発生する可能性のある問題のトラブルシューティングについては、NSX Advanced Load Balancer のトラブルシューティングを参照してください。

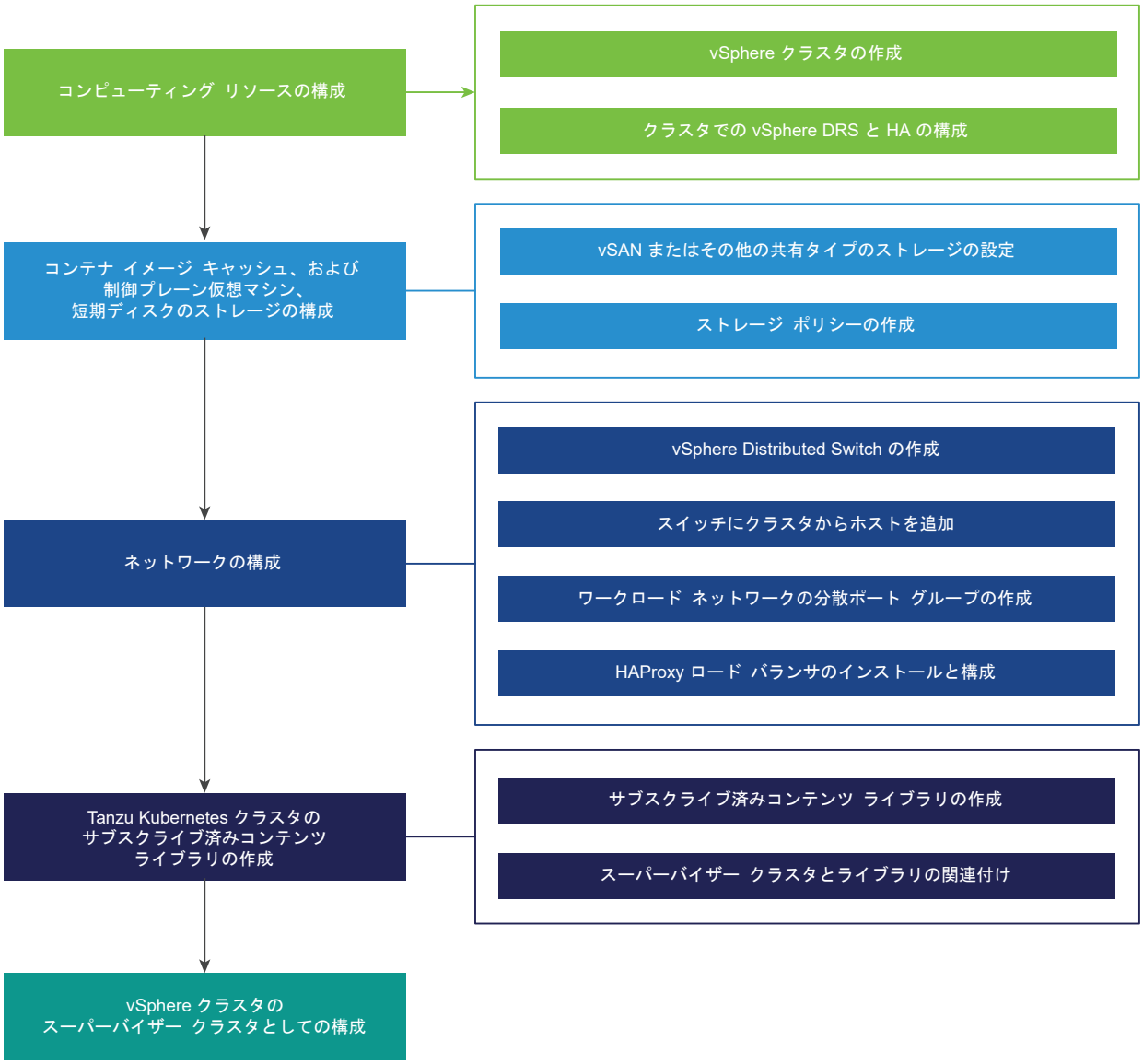
vSphere ネットワークと vSphere with Tanzu 用 HAProxy ロード バランサの構成

vSphere with Tanzu 環境で vSphere Distributed Switch ネットワークを使用している場合は、オープン ソース HAProxy ロード バランサをインストールして構成できます。VMware は、OVA ファイルからデプロイできる HAProxy の実装を提供しています。

vSphere ネットワークと HAProxy ロード バランサを使用するスーパーバイザー クラスタのワークフロー

この図では、vSphere ネットワークと vSphere with Tanzu 用 HAProxy ロード バランサを構成するためのワークフローを示します。

図 4-6. HAProxy を使用した Distributed Switch ネットワークの構成ワークフロー



vSphere ネットワークおよび HAProxy ロード バランサを使用して vSphere with Tanzu をセットアップするためのシステム要件

vSphere ネットワーク スタックと HAProxy ロード バランサを使用して vSphere クラスタをスーパーバイザー クラスタとしてセットアップするためのシステム要件を確認します。

コンピューティングの最小要件

システム	最小デプロイ サイズ	CPU	メモリ	ストレージ
vCenter Server 7.0	小	2	16 GB	290 GB
ESXi ホスト 7.0	vSAN を使用しない場合：3 台の ESXi ホストと、ホストあたり 1 つの固定 IP アドレス。 vSAN を使用する場合：2 つ以上の物理 NIC を持つ 4 台の ESXi ホスト ホストは、vSphere DRS と HA が有効になっているクラスタに参加している必要があります。vSphere DRS は、完全自動化モードまたは一部自動化モードになっている必要があります。 **注：** クラスタに参加するホストの名前に小文字が使用されていることを確認します。この条件に該当しない場合は、ワークロード管理のためのクラスタの有効化が失敗する場合があります。	8	ホストあたり 64 GB	該当なし
Kubernetes 制御プレーンの仮想マシン	3	4	16 GB	16 GB

ネットワークの最小要件

注： vSphere 7 スーパーバイザー クラスタ による IPv6 クラスタの作成や、Tanzu Mission Control による IPv6 クラスタの登録はできません。

コンポーネント	最小数	必要な構成
Kubernetes 制御プレーン仮想マシンの固定 IP アドレス	5 つのアドレスのブロック	スーパーバイザー クラスタ 内の Kubernetes 制御プレーン仮想マシンに割り当てられる、連続する 5 つの固定 IP アドレスのブロック。
管理トラフィック ネットワーク	1	ESXi ホスト、vCenter Server、スーパーバイザー クラスタ、およびロード バランサにルーティング可能な管理ネットワーク。イメージ レジストリが外部ネットワーク上にある場合、ネットワークはイメージ レジストリにアクセス可能で、インターネットに接続できる必要があります。イメージ レジストリは、DNS で解決できる必要があります。
vSphere Distributed Switch	1	クラスタのすべてのホストが、vSphere Distributed Switch に接続されている必要があります。

コンポーネント	最小数	必要な構成
HAProxy ロード バランサ	1	vCenter Server インスタンスで構成された HAProxy ロード バランサのインスタンス。 ■ 1 つの HAProxy インスタンスが複数の スーパーバイザー クラスタ を処理する場合、そのインスタンスはすべての スーパーバイザー クラスタ のすべてのワークロード ネットワークとの間でトラフィックをルーティングできることが必要です。この HAProxy が処理するすべての スーパーバイザー クラスタ のワークロード ネットワークで、IP アドレス範囲が重複しないようにする必要があります。 ■ 仮想 IP アドレスの専用 IP アドレス範囲。HAProxy 仮想マシンは、この仮想 IP アドレス範囲の唯一の所有者である必要があります。この範囲は、どの スーパーバイザー クラスタ が所有するワークロード ネットワークに割り当てられた IP アドレス範囲とも重複が許されません。 ■ HAProxy が仮想 IP アドレスを割り当てるために使用するネットワークは、この HAProxy が接続されているすべての スーパーバイザー クラスタ で使用されるワークロード ネットワークにルーティング可能である必要があります。
ワークロード ネットワーク	1	プライマリ ワークロード ネットワークとして構成する vSphere Distributed Switch には、1 つ以上の分散ポート グループを作成する必要があります。選択したトポロジによっては、名前空間のワークロード ネットワークと同じ分散ポート グループを使用することや、追加のポート グループを作成してワークロード ネットワークとして構成することができます。ワークロード ネットワークは次の要件を満たす必要があります。 ■ Tanzu Kubernetes クラスタのトラフィックに使用されるワークロード ネットワークが、相互に、および スーパーバイザー クラスタ プライマリ ワークロード ネットワークとの間でルーティングできること。 ■ 任意のワークロード ネットワークと、HAProxy が仮想 IP アドレスの割り当てに使用するネットワークとの間でルーティングできること。 ■ スーパーバイザー クラスタ 内のすべてのワークロード ネットワークで IP アドレス範囲の重複がないこと。 **重要：** ワークロード ネットワークは、管理ネットワークとは別のサブネットに配置してください。
NTP サーバおよび DNS サーバ	1	vCenter Server で使用できる DNS サーバおよび NTP サーバ。 **注：** すべての ESXi ホストおよび vCenter Server で NTP を構成します。

コンポーネント	最小数	必要な構成
DHCP サーバ	1	オプション。管理ネットワークとワークロード ネットワークの IP アドレスおよびフローティング IP アドレスを自動的に取得するように DHCP サーバを構成します。DHCP サーバはクライアント識別子をサポートし、互換性のある DNS サーバ、DNS 検索ドメイン、および NTP サーバを提供する必要があります。 DHCP 構成は、スーパーバイザー クラスタ で使用されます。ロード バランサで管理を行うには、固定 IP アドレスが必要になる場合があります。DHCP スコープは、これらの固定 IP アドレスと重複しないようにしてください。DHCP は仮想 IP アドレスには使用されません。(VIP)
管理ネットワークのサブネット	1	ESXi ホストおよび vCenter Server と Kubernetes 制御プレーンとの間の管理トラフィックに使用されるサブネット。サブネットのサイズは次のようにする必要があります。 ■ ホストの VMkernel アダプタごとに 1 つの IP アドレス。 ■ vCenter Server アプライアンスに 1 つの IP アドレス。 ■ Kubernetes 制御プレーンに 5 つの IP アドレス。3 台のノードそれぞれに 1 つずつ、仮想 IP アドレス用に 1 つ、クラスタのローリング アップグレード用に 1 つ。 **注：** 管理ネットワークとワークロード ネットワークは異なるサブネット上に配置する必要があります。管理ネットワークとワークロード ネットワークに同じサブネットを割り当てることはできないため、システム エラーや問題が発生することがあります。
管理ネットワークの VLAN	1	管理ネットワークのサブネットの VLAN ID。
物理ネットワークの MTU	1600	オーバーレイ トラフィックを伝送するネットワークでは、MTU サイズを 1,600 以上にする必要があります。
Kubernetes サービスの CIDR 範囲	/16 プライベート IP アドレス	Kubernetes サービスに IP アドレスを割り当てるためのプライベート CIDR 範囲。スーパーバイザー クラスタ ごとに一意の Kubernetes サービス CIDR 範囲を指定する必要があります。

HAProxy ロード バランサをデプロイするトポロジ

vSphere with Tanzu と Distributed Switch ネットワークを併用している場合、HAProxy は、Tanzu Kubernetes 制御プレーンにアクセスする開発者と、ロード バランサ タイプの Kubernetes サービスに対応したロード バランシングを提供します。HAProxy ロード バランサに実装可能なトポロジを確認してください。

スーパーバイザー クラスタ のワークロード ネットワーク

vSphere ネットワーク スタックを使用した スーパーバイザー クラスタ を構成するには、クラスタ内のすべてのホストを vSphere Distributed Switch に接続する必要があります。スーパーバイザー クラスタ ワークロード ネットワーク用に実装するトポロジに応じて、1 つ以上の分散ポート グループを作成します。作成したポート グループを、ワークロード ネットワークとして、vSphere 名前空間 に指定します。

ホストを スーパーバイザー クラスタ に追加する前に、クラスタに含まれるすべての vSphere Distributed Switch にホストを追加する必要があります。

ワークロード ネットワークは、Tanzu Kubernetes クラスタのノードと、スーパーバイザー クラスタ 制御プレーン仮想マシンへの接続を提供します。Kubernetes 制御プレーン仮想マシンに接続を提供するワークロード ネットワークは、プライマリ ワークロード ネットワークと呼ばれます。各 スーパーバイザー クラスタ にそれぞれ 1 つのプライマリ ワークロード ネットワークが必要です。スーパーバイザー クラスタ に対して、分散ポート グループの 1 つをプライマリ ワークロード ネットワークとして指定する必要があります。

注： ワークロード ネットワークが追加されるのは スーパーバイザー クラスタ を有効にするときだけで、後で追加することはできません。

スーパーバイザー クラスタ 上の Kubernetes 制御プレーン仮想マシンは、プライマリ ワークロード ネットワークに割り当てられた IP アドレス範囲から 3 つの IP アドレスを使用します。Tanzu Kubernetes クラスタの各ノードには、この Tanzu Kubernetes クラスタが実行されている名前空間で構成されたワークロード ネットワークのアドレス範囲から割り当てられた、それぞれ別の IP アドレスが割り当てられます。

IP アドレス範囲の割り当て

HA プロキシ ロード バランサを使用する スーパーバイザー クラスタ のネットワーク トポロジを計画するときは、次の 2 種類の IP アドレス範囲を設定するように計画します。

- HAProxy に対する仮想 IP アドレスの割り当ての範囲。HAProxy の仮想サーバ用に構成した IP アドレス範囲は、ロード バランサ アプライアンスによって予約されます。たとえば、仮想 IP アドレス範囲が `192.168.1.0/24` の場合、この範囲に含まれるすべてのホストは、仮想 IP トラフィック以外のトラフィックではアクセスできません。

  注： HAProxy 仮想 IP アドレス範囲内にゲートウェイを構成するとゲートウェイへのすべてのルートが失敗するため、構成しないようにします。

- スーパーバイザー クラスタ および Tanzu Kubernetes クラスタのノードの IP アドレス範囲。スーパーバイザー クラスタ 内の Kubernetes 制御プレーン仮想マシンにはそれぞれ IP アドレスが割り当てられていて、合計で 3 つの IP アドレスがあります。Tanzu Kubernetes クラスタの各ノードには、別の IP アドレスが割り当てられます。名前空間に構成する スーパーバイザー クラスタ の各ワークロード ネットワークに、一意の IP アドレス範囲を割り当てる必要があります。

1 つの /24 ネットワークによる構成の例：

- ネットワーク：192.168.120.0/24
- HAProxy 仮想 IP アドレス：192.168.120.128/25
- HAProxy ワークロード インターフェイスの 1 つの IP アドレス：192.168.120.5

最初の 128 アドレスに含まれる制限のない IP アドレスに応じて、スーパーバイザー クラスタ のワークロード ネットワークの IP アドレス範囲を定義できます。たとえば、次のようにします。

- 192.168.120.31 ～ 192.168.120.40：プライマリ ワークロード ネットワーク
- 192.168.120.51 ～ 192.168.120.60：別のワークロード ネットワーク

注： ワークロード ネットワークに定義する範囲は、HAProxy 仮想 IP アドレスの範囲と重複しないようにする必要があります。

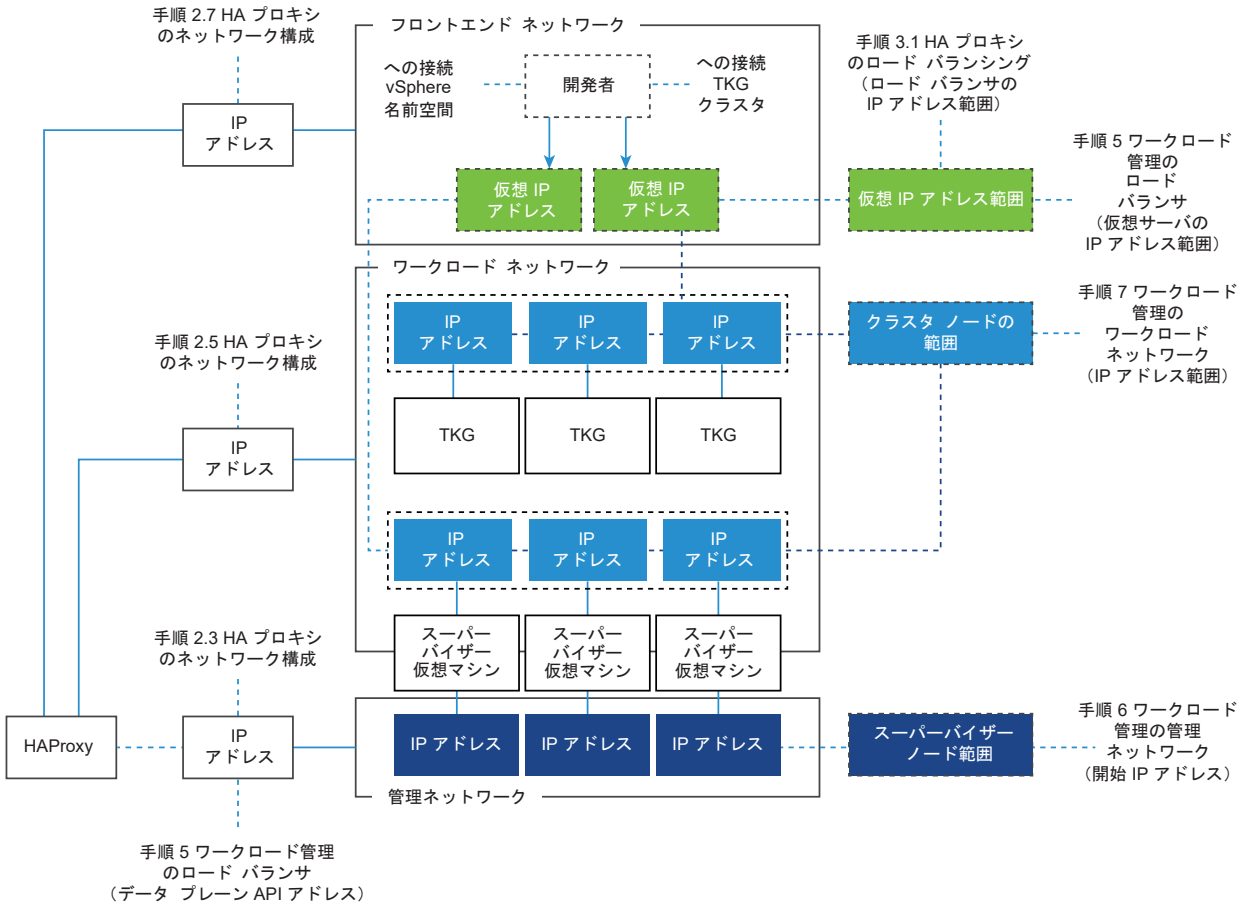
HAProxy ネットワーク トポロジ

ネットワーク構成には、[デフォルト] と [フロントエンド] という、HAProxy をデプロイする 2 つのオプションがあります。デフォルト ネットワークには、管理ネットワーク用とワークロード ネットワーク用の 2 つの NIC があります。フロントエンド ネットワークには、管理ネットワーク、ワークロード ネットワーク、クライアント用のフロントエンド ネットワークの 3 つの NIC があります。次の表に、各ネットワークの特性の一覧と説明を示します。

本番環境インストールでは、[フロントエンド ネットワーク] 構成を使用して HAProxy ロード バランサをデプロイすることが推奨されます。[デフォルト] 構成を使用して HAProxy ロード バランサをデプロイする場合は、ワークロード ネットワークに /24 の IP アドレス ブロック サイズを割り当てることをお勧めします。いずれの構成オプションでも、DHCP は推奨されません。

ネットワーク	特性
[管理]	スーパーバイザー クラスタは管理ネットワークを使用して HAProxy ロード バランサに接続し、このバランサをプログラムします。 ■ HAProxy データ プレーン API エンドポイントは、管理ネットワークに接続されているネットワーク インターフェイスにバインドされています。 ■ スーパーバイザー クラスタがロード バランサ API に確実に接続できるようにするには、HAProxy 制御プレーン仮想マシンに割り当てられた管理 IP アドレスに、管理ネットワーク上の固定 IP アドレスを指定する必要があります。 ■ HAProxy 仮想マシンのデフォルト ゲートウェイは、このネットワーク上に存在する必要があります。 ■ DNS クエリはこのネットワーク上で実行する必要があります。
[ワークロード]	HAProxy 制御プレーン仮想マシンは、ワークロード ネットワークを使用して、スーパーバイザー クラスタおよび Tanzu Kubernetes クラスタ ノードのサービスにアクセスします。 ■ HAProxy 制御プレーン仮想マシンは、このネットワーク上のスーパーバイザーおよび Tanzu Kubernetes クラスタ ノードにトラフィックを転送します。 ■ HAProxy 制御プレーン仮想マシンがデフォルト モード（2 つの NIC）でデプロイされている場合、ロード バランサ サービスへのアクセスに使用される論理ネットワークはワークロード ネットワークが提供する必要があります。 ■ [デフォルト] 構成では、ロードバランサの仮想 IP アドレスと Kubernetes クラスタ ノードの IP アドレスはこのネットワークから取得されます。これらのアドレスは、ネットワーク内の、重複しない、独立した範囲として定義されます。 **注：** ワークロード ネットワークは、管理ネットワークとは別のサブネットに配置してください。vSphere ネットワークおよび HAProxy ロード バランサを使用して vSphere with Tanzu をセットアップするためのシステム要件を参照してください。
[フロントエンド]（オプション）	クラスタ ワークロードにアクセスする外部クライアント（ユーザーやアプリケーションなど）は、フロントエンド ネットワークを使用して、仮想 IP アドレスによってバックエンド ロード バランシング サービスにアクセスします。 ■ フロントエンド ネットワークが使用されるのは、HAProxy 制御プレーン仮想マシンが 3 つの NIC を使用してデプロイされている場合のみです。 ■ 本番環境インストールに推奨されます。 ■ フロントエンド ネットワークで、仮想 IP アドレス (VIP) が公開されます。HAProxy は、トラフィックのバランスを調整し、トラフィックを適切なバックエンドに転送します。

次の図に、[フロントエンド ネットワーク] トポロジを使用した HAProxy デプロイを示します。この図は、インストールおよび構成プロセスで想定される構成フィールドの場所を示しています。



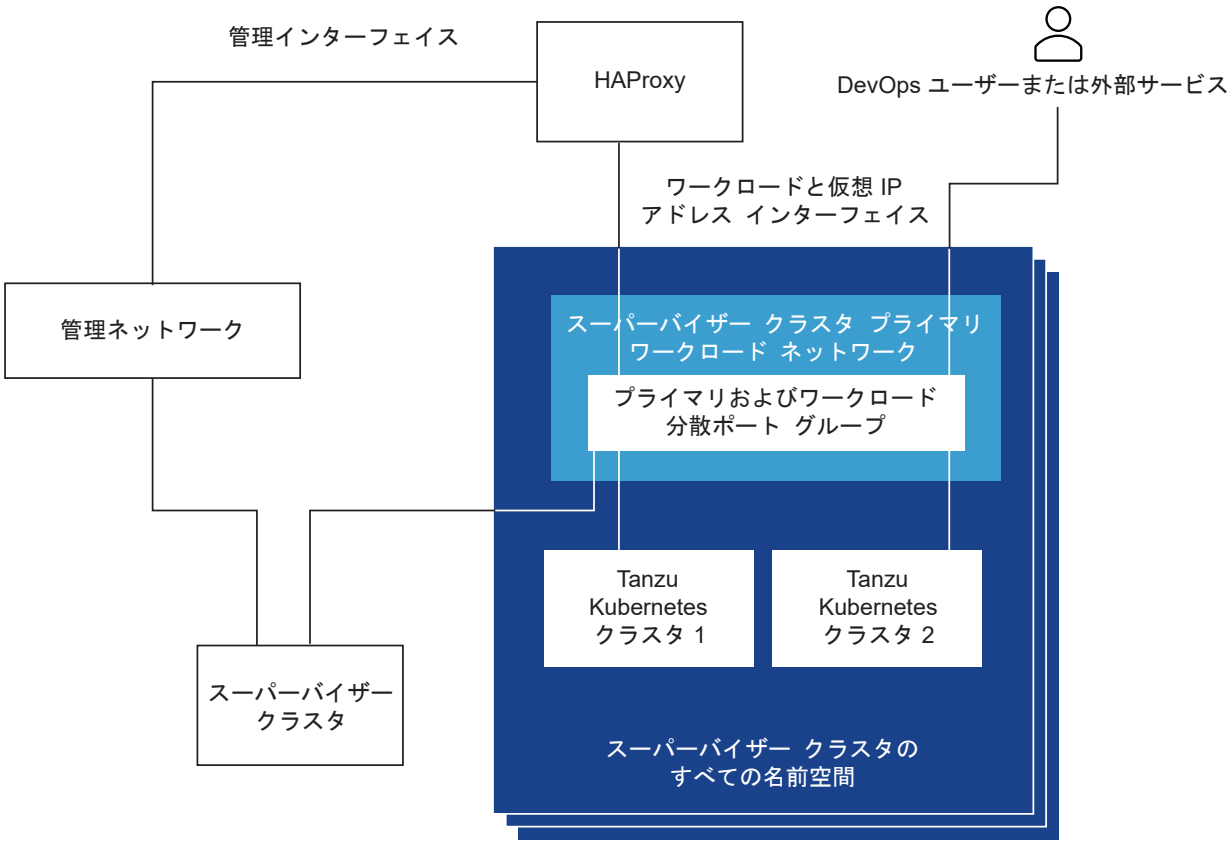
1 つのワークロード ネットワークを使用する スーパーバイザー クラスタ トポロジと 2 つの仮想 NIC を使用する HA プロキシ

このトポロジでは、次のコンポーネントに対する 1 つのワークロード ネットワークを持つ スーパーバイザー クラスタ を構成します。

- Kubernetes 制御プレーンの仮想マシン
- Tanzu Kubernetes クラスタのノード。
- 外部サービスと DevOps ユーザーが接続する HAProxy の仮想 IP アドレス範囲。この構成では、2 つの仮想 NIC を使用する HAProxy がデプロイされます（[デフォルト] 構成）。1 つは管理ネットワークに接続され、もう 1 つはプライマリ ワークロード ネットワークに接続されます。仮想 IP アドレスは、プライマリ ワークロード ネットワークとは別のサブネットに割り当てる必要があります。

スーパーバイザー クラスタ に対して 1 つのポート グループをプライマリ ワークロード ネットワークとして指定してから、同じポート グループを vSphere 名前空間 のワークロード ネットワークとして使用します。スーパーバイザー クラスタ、Tanzu Kubernetes クラスタ、HAProxy、DevOps ユーザー、および外部サービスはすべて、プライマリ ワークロード ネットワークとして設定されている同じ分散ポート グループに接続されます。

図 4-7. 1 つのネットワークによってバッキングされる スーパーバイザー クラスタ



DevOps ユーザーまたは外部アプリケーションのトラフィック パスは次のとおりです。

1 DevOps ユーザーまたは外部サービスは、分散ポート グループのワークロード ネットワーク サブネット上の仮想 IP アドレスにトラフィックを送信します。

2 HAProxy は、仮想 IP トラフィックを Tanzu Kubernetes ノードの IP アドレス、または制御プレーン仮想マシンの IP アドレスのいずれかにロード バランシングします。HAProxy は、仮想 IP アドレスを要求して、その IP アドレスで受信されるトラフィックのロード バランシングを行うことができるようにします。

3 制御プレーン仮想マシンまたは Tanzu Kubernetes クラスタ ノードは、スーパーバイザー クラスタ または Tanzu Kubernetes クラスタ内でそれぞれ実行されているターゲット ポッドにトラフィックを配信します。

隔離されたワークロード ネットワークを使用する スーパーバイザー クラスタ トポロジと 2 つの仮想 NIC を使用する HA プロキシ

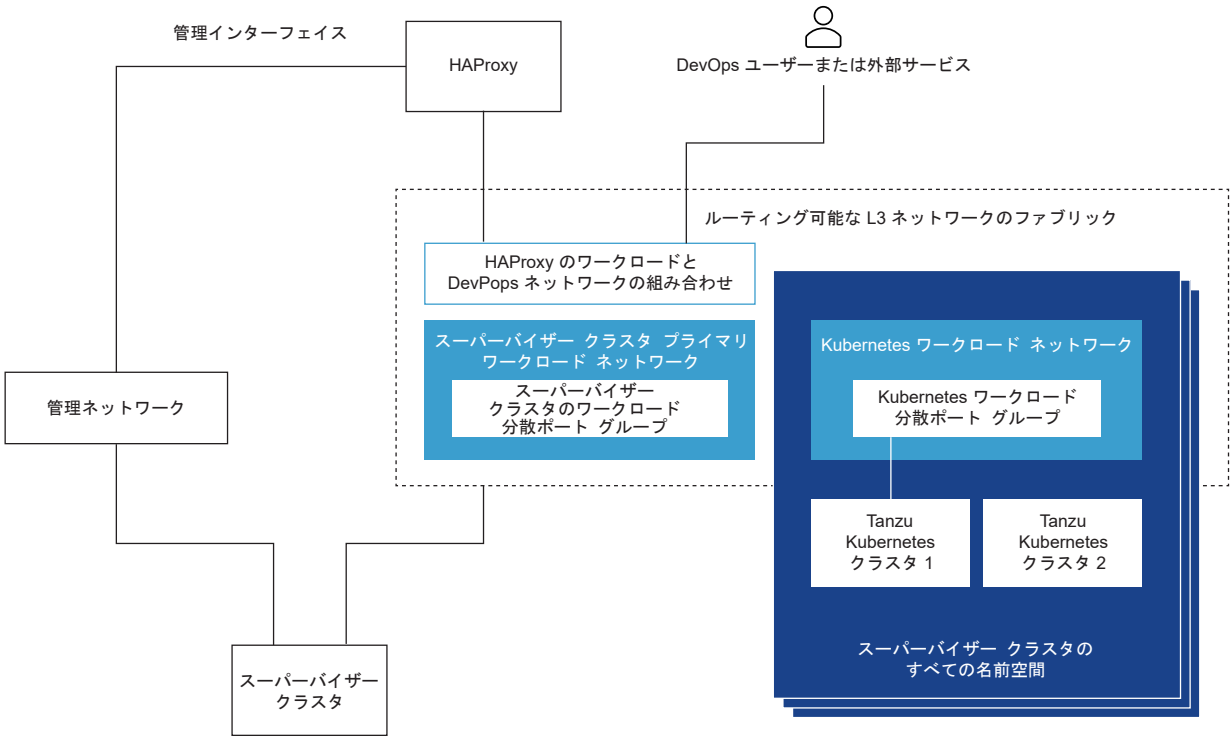
このトポロジでは、次のコンポーネントに対するネットワークを構成します。

- Kubernetes 制御プレーン仮想マシン。Kubernetes 制御プレーン仮想マシンのトラフィックを処理するプライマリ ワークロード ネットワーク。
- Tanzu Kubernetes クラスタ ノード。ワークロード ネットワーク。これは、スーパーバイザー クラスタ 上のすべての名前空間に割り当てられます。このネットワークは Tanzu Kubernetes クラスタ ノードに接続します。

- HAProxy の仮想 IP アドレス。この構成では、2 つの仮想 NIC を使用する HAProxy 仮想マシンがデプロイされます（[デフォルト] 構成）。HAProxy 仮想マシンは、プライマリ ワークロード ネットワークか、名前空間に使用するワークロード ネットワークのいずれかに接続できます。また、HAProxy は、vSphere に既存の、プライマリ ネットワークおよびワークロード ネットワークにルーティング可能な仮想マシン ネットワークに接続することもできます。

スーパーバイザー クラスタ は、プライマリ ワークロード ネットワークをバッキングする分散ポート グループに接続され、Tanzu Kubernetes クラスタは、ワークロード ネットワークをバッキングする分散ポート グループに接続されます。2 つのポート グループは、レイヤー 3 でルーティング可能である必要があります。VLAN を使用してレイヤー 2 の隔離を実装できます。レイヤー 3 トラフィックのフィルタリングは、IP ファイアウォールとゲートウェイを介して実現できます。

図 4-8. 隔離されたワークロード ネットワークを使用する スーパーバイザー クラスタ



DevOps ユーザーまたは外部サービスのトラフィック パスは次のとおりです。

1 DevOps ユーザーまたは外部サービスは、仮想 IP アドレスにトラフィックを送信します。トラフィックは、HAProxy が接続されているネットワークにルーティングされます。

2 HAProxy は、仮想 IP トラフィックを Tanzu Kubernetes ノードの IP アドレス、または制御プレーン仮想マシンのいずれかにロード バランシングします。HAProxy は、仮想 IP アドレスを要求して、その IP アドレスで受信されるトラフィックのロードバランシングを行うことができるようにします。

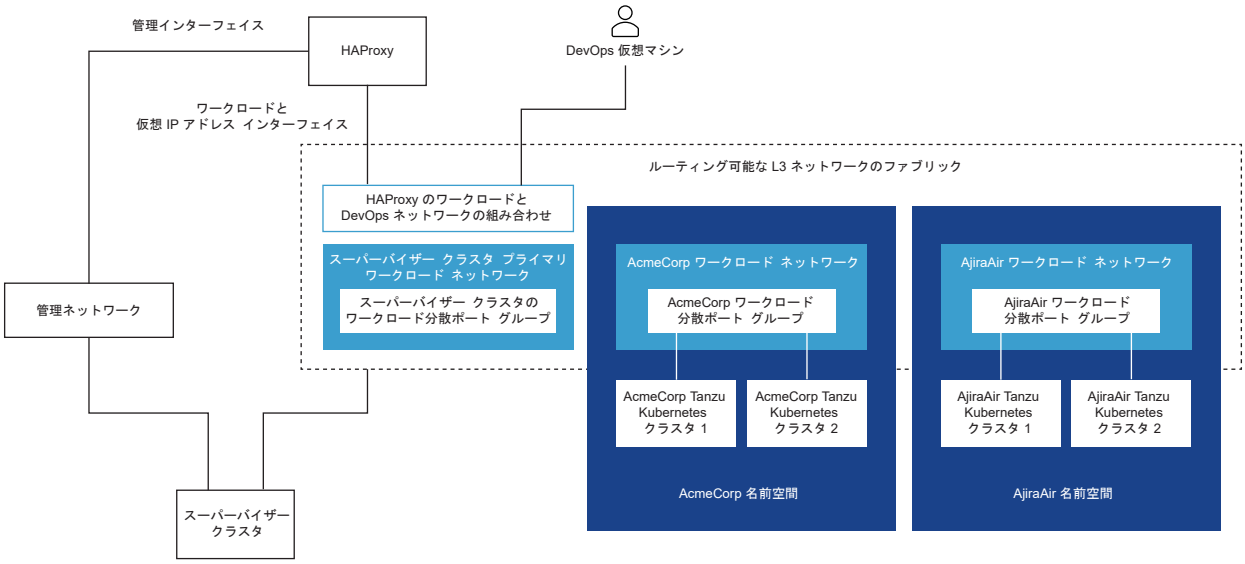
3 制御プレーン仮想マシンまたは Tanzu Kubernetes クラスタ ノードは、Tanzu Kubernetes クラスタ内で実行されているターゲット ポッドにトラフィックを配信します。

複数のワークロード ネットワークを使用する スーパーバイザー クラスタ トポロジと 2 つの仮想 NIC を使用する HA プロキシ

このトポロジでは、1 つのポート グループがプライマリ ワークロード ネットワークとして動作するように、また、1 つの専用ポート グループが各名前空間に対するワークロード ネットワークとして機能するように構成できます。HAProxy は 2 つの仮想 NIC とともにデプロイされ（[デフォルト] 構成）、プライマリ ワークロード ネットワークか、いずれかのワークロード ネットワークに接続することができます。プライマリおよびワークロード ネットワークにルーティング可能な、既存の仮想マシン ネットワークを使用することもできます。

このトポロジの DevOps ユーザーと外部サービスのトラフィック パスは、隔離されたワークロード ネットワーク トポロジの場合と同じです。

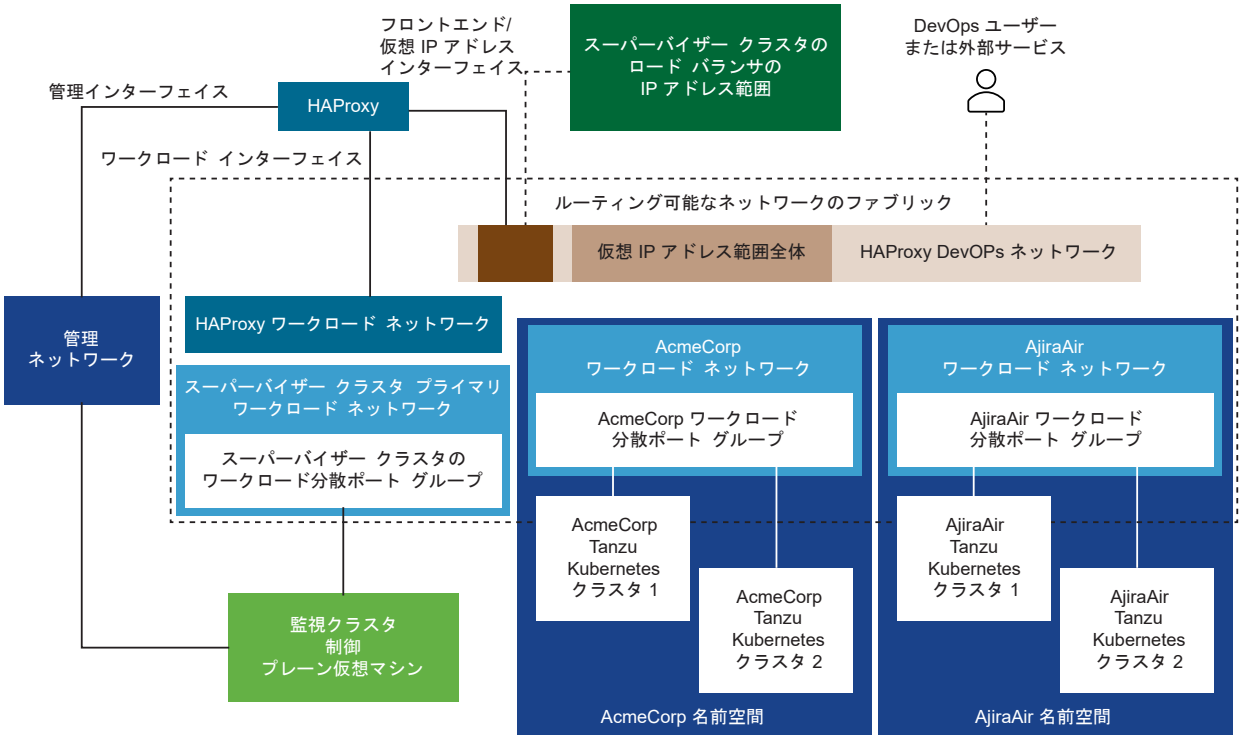
図 4-9. 複数の隔離されたワークロード ネットワークによってバッキングされる スーパーバイザー クラスタ



複数のワークロード ネットワークを使用する スーパーバイザー クラスタ トポロジと 3 つの仮想 NIC を使用する HA プロキシ

この構成では、3 つの仮想 NIC を使用する HAProxy 仮想マシンをデプロイします。これにより、HAProxy がフロントエンド ネットワークに接続されます。DevOps ユーザーと外部サービスは、フロントエンド ネットワーク上の仮想 IP アドレスを介して HAProxy にアクセスできます。本番環境では、3 つの仮想 NIC を使用する HA プロキシをデプロイすることを推奨します。

図 4-10. 3 つの仮想 NIC を使用する HAProxy をデプロイ



使用できるトポロジからの選択

使用できるトポロジから選択する前に、次のように環境のニーズを検討します。

1 スーパーバイザー クラスタ と Tanzu Kubernetes クラスタの間でレイヤー 2 の隔離が必要ですか。

   a いいえ：すべてのコンポーネントで使用される 1 つのワークロード ネットワークを持つ最もシンプルなトポロジ。

   b はい：分離されたプライマリ ネットワークとワークロード ネットワークを持つ、隔離ワークロード ネットワーク トポロジ。

2 Tanzu Kubernetes クラスタの間でさらにレイヤー 2 の隔離が必要ですか。

   a いいえ：分離されたプライマリ ネットワークとワークロード ネットワークを持つ、隔離ワークロード ネットワーク トポロジ。

   b はい：名前空間ごとに分離されたワークロード ネットワークと専用のプライマリ ワークロード ネットワークを持つ、複数のワークロード ネットワーク トポロジ。

3 DevOps ユーザーと外部サービスが Kubernetes 制御プレーン仮想マシンと Tanzu Kubernetes クラスタ ノードに直接ルーティングされないようにする必要がありますか?

   a いいえ：2 つの NIC を使用する HAProxy 構成。

   b はい：3 つの NIC を使用する HAProxy 構成。この構成は、本番環境に推奨されます。

HAProxy ロード バランサで使用する スーパーバイザー クラスタ の vSphere Distributed Switch の作成

vSphere ネットワーク スタックおよび HA プロキシ ロード バランサを使用する スーパーバイザー クラスタ として vSphere クラスタを構成するには、vSphere Distributed Switch にホストを追加する必要があります。Distributed Switch 上で スーパーバイザー クラスタ のワークロード ネットワークとして構成するポート グループを作成する必要があります。

クラスタで実行される Kubernetes ワークロードに指定した隔離のレベルに応じて、スーパーバイザー クラスタ のさまざまなトポロジから選択できます。

前提条件

- HAProxy ロード バランサで スーパーバイザー クラスタ の vSphere ネットワークを使用するためのシステム要件を確認します。vSphere ネットワークおよび HAProxy ロード バランサを使用して vSphere with Tanzu をセットアップするためのシステム要件を参照してください。
- スーパーバイザー クラスタ で HA プロキシを使用するワークロード ネットワークを設定するトポロジを判別します。HAProxy ロード バランサをデプロイするトポロジを参照してください。

手順

1 vSphere Client で、データセンターに移動します。

2 データセンターを右クリックして、[Distributed Switch] - [新しい Distributed Switch] の順に選択します。

3 スイッチの名前（**ワークロード Distributed Switch** など）を入力して、[次へ] をクリックします。

4 スイッチのバージョン 7.0 を選択して、[次へ] をクリックします。

5 [ポート グループ名] に **プライマリ ワークロード ネットワーク** と入力し、[次へ] をクリックして [終了] をクリックします。

1 つのポート グループを持つ新しい Distributed Switch がデータセンターに作成されます。このポート グループは、作成する スーパーバイザー クラスタ のプライマリ ワークロード ネットワークとして使用できます。プライマリ ワークロード ネットワークは、Kubernetes 制御プレーン仮想マシンのトラフィックを処理します。

6 ワークロード ネットワークの分散ポート グループを作成します。

作成するポート グループの数は、スーパーバイザー クラスタ 用に実装するトポロジによって異なります。隔離されたワークロード ネットワークが 1 つ含まれるトポロジの場合は、スーパーバイザー クラスタ のすべての名前空間用のネットワークとして使用する分散ポート グループを 1 つ作成します。ネットワークが名前空間ごとに隔離されているトポロジの場合は、作成する名前空間と同じ数のポート グループを作成します。

a 新しく作成した Distributed Switch に移動します。

b スイッチを右クリックして、[分散ポート グループ] - [新規分散ポート グループ] の順に選択します。

c ポート グループの名前（**ワークロード ネットワーク** など）を入力して、[次へ] をクリックします。

d デフォルトのままにして、[次へ] をクリックし、[終了] をクリックします。

7 スーパーバイザー クラスタ として構成する vSphere クラスタのホストを Distributed Switch に追加します。

a Distributed Switch を右クリックして、[ホストの追加と管理] を選択します。

b [ホストの追加] を選択します。

c [新規ホスト] をクリックし、スーパーバイザー クラスタ として構成する vSphere クラスタからホストを選択して [次へ] をクリックします。

d 各ホストから物理 NIC を選択して、Distributed Switch 上のアップリンクに割り当てます。

e ウィザードの残りの各画面で [次へ] をクリックし、最後に [終了] をクリックします。

結果

ホストが Distributed Switch に追加されます。以上で、スイッチ上に作成したポート グループを スーパーバイザー クラスタ のワークロード ネットワークとして使用できます。

HAProxy ロード バランサのインストールと構成

VMware では、vSphere with Tanzu 環境で使用できるオープン ソースの HAProxy ロード バランサの実装を提供しています。[ワークロード管理] に vSphere Distributed Switch (vDS) ネットワークを使用している場合は、HAProxy ロード バランサをインストールして、構成することができます。

HAProxy ロード バランサ制御プレーン仮想マシンのデプロイ

Kubernetes ワークロードに vSphere ネットワーク スタックを使用する場合は、HAProxy 制御プレーン仮想マシンをインストールして、Tanzu Kubernetes クラスタにロード バランシング サービスを提供します。

前提条件

- 環境が HA プロキシをデプロイするためのコンピューティング要件とネットワーク要件を満たしていることを確認します。vSphere ネットワークおよび HAProxy ロード バランサを使用して vSphere with Tanzu をセットアップするためのシステム要件を参照してください。
- HAProxy ロード バランサをデプロイする vSphere 標準スイッチまたは Distributed Switch 上に管理ネットワークが配置されていることを確認します。スーパーバイザー クラスタ は管理ネットワーク上の HAProxy ロード バランサと通信します。
- ワークロード ネットワーク用の vSphere Distributed Switch とポート グループを作成します。HAProxy ロード バランサは、ワークロード ネットワーク経由で スーパーバイザー クラスタ および Tanzu Kubernetes クラスタのノードと通信します。HAProxy ロード バランサで使用する スーパーバイザー クラスタ の vSphere Distributed Switch の作成を参照してください。ワークロード ネットワークの詳細については、スーパーバイザー クラスタ のワークロード ネットワークを参照してください。
- VMware-HAProxy サイトから最新バージョンの VMware HAProxy OVA ファイルをダウンロードします。
- スーパーバイザー クラスタ に HAProxy ロード バランサとワークロード ネットワークをデプロイするためのトポロジを選択します。HAProxy ロード バランサをデプロイするトポロジを参照してください

Distributed Switch ネットワークおよび HAProxy と vSphere with Tanzu の併用方法を示すデモが役立つ場合があります。「Getting Started Using vSphere with Tanzu」のビデオを確認してください。

手順

1 vSphere Client を使用して、vCenter Server にログインします。

2 HAProxy OVA ファイルから新しい仮想マシンを作成します。

オプション	説明
コンテンツ ライブラリ	OVA をローカル コンテンツ ライブラリにインポートした場合： ■ [メニュー] - [コンテンツ ライブラリ] の順に選択します。 ■ OVA をインポートしたライブラリを選択します。 ■ `vmware-haproxy-vX.X.X` テンプレートを選択します。 ■ 右クリックして、[このテンプレートから仮想マシンを新規作成] を選択します。
ローカル ファイル	OVA ファイルをローカル ホストにダウンロードした場合： ■ [ワークロード管理] を有効にする vCenter Server クラスタを選択します。 ■ 右クリックして [OVF テンプレートのデプロイ] を選択します。 ■ [ローカル ファイル] を選択し [ファイルのアップロード] をクリックします。 ■ `vmware-haproxy-vX.X.X.ova` ファイルを参照して選択します。

3 [仮想マシン名] に **haproxy** などの名前を入力します。

4 HAProxy をデプロイする [データセンター] 選択して、[次へ] をクリックします。

5 [ワークロード管理] を有効にする vCenter Server クラスタを選択して、[次へ] をクリックします。

6 デプロイの詳細を確認して、[次へ] をクリックします。

7 使用許諾契約書を承諾して、[次へ] をクリックします。

8 デプロイの設定を選択してください。詳細については、HAProxy ネットワーク トポロジを参照してください。

構成	説明
デフォルト	管理ネットワーク用と単一ワークロード ネットワーク用の 2 つの NIC を持つアプライアンスをデプロイする場合は、このオプションを選択します。
フロントエンド ネットワーク	3 つの NIC を持つアプライアンスをデプロイする場合は、このオプションを選択します。フロントエンド サブネットは、開発者がクラスタ制御プレーンにアクセスするために使用するネットワークからクラスタ ノードを隔離する場合に使用されます。

9 仮想マシンに使用するストレージ ポリシーを選択し、[次へ] をクリックします。

10 ロード バランサに使用するネットワーク インターフェイスを選択し、[次へ] をクリックします。

ソース ネットワーク	ターゲット ネットワーク
管理	仮想マシン ネットワークなどの管理ネットワークを選択します。
ワークロード	[ワークロード管理] 用に構成された Distributed Switch ポートグループを選択します。
フロントエンド	フロントエンド サブネットに構成された Distributed Switch ポートグループを選択します。フロントエンド構成を選択しなかった場合、この設定はインストール中に無視されるため、デフォルトのままにすることができます。

注： ワークロード ネットワークは、管理ネットワークとは別のサブネットに配置してください。vSphere ネットワークおよび HAProxy ロード バランサを使用して vSphere with Tanzu をセットアップするためのシステム要件を参照してください。

11 アプリケーション構成設定をカスタマイズします。アプライアンスの設定を参照してください。

12 ネットワーク構成の詳細を指定します。ネットワークの構成を参照してください。

13 ロード バランシングを構成します。ロード バランシングの設定を参照してください。

14 [次へ] をクリックして、OVA の構成を完了します。

15 デプロイ構成の詳細を確認し、[完了] をクリックして OVA をデプロイします。

16 [タスク] パネルを使用して、仮想マシンのデプロイを監視します。

17 仮想マシンのデプロイが完了したら、パワーオンします。

次のステップ

HAProxy ロード バランサが正常に展開されて、パワーオンされたら、[ワークロード管理] の有効化を続行します。5 章 スーパーバイザー クラスタ の構成と管理を参照してください。

HAProxy ロード バランサのカスタマイズ

構成設定、ネットワーク設定、ロード バランシング設定などを行って、HAProxy 制御プレーン仮想マシンをカスタマイズします。

アプライアンスの設定

次の表に、HAProxy アプライアンス構成のパラメータの一覧と説明を示します。

パラメータ	説明	注釈または例
root パスワード	root ユーザーの初期パスワード（6 ～ 128 文字）。	以降のパスワード変更は、オペレーティング システムで実行する必要があります。
root ログインの許可	root ユーザーに、SSH を使用して仮想マシンにリモートでログインすることを許可するオプション。	root ログインはトラブルシューティングのために必要になることがありますが、root ログインを許可した場合のセキュリティ面の影響を考慮する必要があります。

パラメータ	説明	注釈または例
TLS 認証局 (ca.crt)	自己署名 CA 証明書を使用するには、このフィールドを空白のままにします。 独自の CA 証明書 (ca.crt) を使用するには、証明書の内容をこのフィールドに貼り付けます。 Base64 によるコンテンツのエンコードが必要になる場合があります。https://www.base64encode.org/	自己署名 CA 証明書を使用している場合は、証明書からパブリック キーとプライベート キーが生成されます。
キー (ca.key)	自己署名証明書を使用している場合は、このフィールドを空白のままにします。 CA 証明書を指定した場合は、このフィールドに証明書のプライベート キーの内容を貼り付けます。	

ネットワークの構成

次の表に、HAProxy ネットワーク構成のパラメータの一覧と説明を示します。

パラメータ	説明	注釈または例
ホスト名	HAProxy 制御プレーン仮想マシンに割り当てるホスト名（または FQDN）	デフォルト値：`haproxy.local`
DNS	DNS サーバの IP アドレスのカンマ区切りリスト。	デフォルト値：`1.1.1.1, 1.0.0.1` 値の例：`10.8.8.8`
管理 IP	管理ネットワーク上の HAProxy 制御プレーン仮想マシンの固定 IP アドレス。	ネットワークのプリフィックス長を含む有効な IPv4 アドレス（例：`192.168.0.2/24`）。
管理ゲートウェイ	管理ネットワークのゲートウェイの IP アドレス。	例： `192.168.0.1`
ワークロード IP アドレス	ワークロード ネットワーク上の HAProxy 制御プレーン仮想マシンの固定 IP アドレス。 この IP アドレスには、ロード バランサの IP アドレス範囲外のアドレスを指定する必要があります。	ネットワークのプリフィックス長を含む有効な IPv4 アドレス（例：`192.168.10.2/24`）。
ワークロード ゲートウェイ	ワークロード ネットワークのゲートウェイの IP アドレス。	例： `192.168.10.1` フロントエンド構成を選択した場合は、ゲートウェイを入力する必要があります。フロントエンドが選択されていても、ゲートウェイが指定されていない場合、デプロイは正常に実行されません。
フロントエンド IP アドレス	フロントエンド ネットワーク上の HAProxy アプライアンスの固定 IP アドレス。 この値は、フロントエンド デプロイ モデルが選択されている場合のみ使用されます。	ネットワークのプリフィックス長を含む有効な IPv4 アドレス（例：`192.168.100.2/24`）。
フロントエンド ゲートウェイ	フロントエンド ネットワークのゲートウェイの IP アドレス。 この値は、フロントエンド デプロイ モデルが選択されている場合のみ使用されます。	例： `192.168.100.1`

ロード バランシングの設定

次の表に、HAProxy ロード バランサ構成のパラメータの一覧と説明を示します。

パラメータ	説明	例または注釈
ロード バランサの IP アドレス範囲	このフィールドには、CIDR 形式を使用する IPv4 アドレスの範囲を指定します。値には有効な CIDR 範囲を指定する必要があります。指定しなかった場合、インストールは失敗します。 HAProxy は、仮想 IP アドレス (VIP) 用の IP アドレスを予約します。割り当てられていた各 VIP アドレスが割り当てられると、HAProxy はこのアドレスで要求に応答します。 vSphere Client を使用して vCenter Server で [ワークロード管理] を有効にした場合、ここで指定する CIDR 範囲と仮想サーバに割り当てる IP アドレスが重複することはできません。	たとえば、ネットワーク CIDR `192.168.100.0/24` は、ロード バランサに 256 個の仮想 IP アドレスを提供します（各アドレスの範囲は `192.168.100.0 - 192.168.100.255`）。 たとえば、ネットワーク CIDR `192.168.100.0/25` は、ロード バランサに 128 個の仮想 IP アドレスを提供します (各アドレスの範囲は `192.168.100.0 - 192.168.100.127`)。
データプレーン API の管理ポート	ロード バランサの API サービスが待機する HAProxy 仮想マシンのポート。	有効なポート。ポート 22 は SSH 用に予約されています。デフォルト値は `5556` です。
HAProxy のユーザー ID	ロード バランサ API のユーザー名	クライアントがロード バランサの API サービスの認証に使用するユーザー名です。 **注：** このユーザー名は、スーパーバイザー クラスタ を有効にするときに必要になります。
HAProxy のパスワード	ロード バランサ API のパスワード	クライアントがロード バランサの API サービスの認証に使用するパスワードです。 **注：** このパスワードは、スーパーバイザー クラスタ を有効にするときに必要になります。

スーパーバイザー クラスタ の構成と管理

5

vSphere 管理者は、スーパーバイザー クラスタ を作成することで、ワークロード管理で vSphere クラスタを有効にします。スーパーバイザー クラスタ を作成する際に、ネットワーク ソリューションとして vSphere ネットワーク スタックを使用するのか、あるいは NSX-T Data Center を使用するのかを選択できます。NSX-T Data center が構成されたクラスタでは、vSphere ポッド と、VMware Tanzu™ Kubernetes Grid™ サービス を介して作成された Tanzu Kubernetes クラスタの実行がサポートされます。vSphere ネットワーク スタックが構成された スーパーバイザー クラスタ では、Tanzu Kubernetes クラスタのみがサポートされます。

スーパーバイザー クラスタ を有効にした後で、vSphere Client を使用したクラスタの管理と監視が可能になります。

この章には、次のトピックが含まれています。

- vSphere クラスタで vSphere with Tanzu を構成するための前提条件
- vSphere ネットワークを使用したワークロード管理の有効化
- NSX-T Data Center ネットワークを使用したワークロード管理の有効化
- スーパーバイザー クラスタ への Tanzu Edition ライセンスの割り当て
- スーパーバイザー クラスタ API エンドポイントに安全に接続するための VIP 証明書の置き換え
- スーパーバイザー クラスタ の Tanzu Kubernetes Grid サービス と Tanzu Mission Control の統合
- Tanzu Kubernetes クラスタのデフォルト CNI の設定
- VDS ネットワークが構成されている スーパーバイザー クラスタ へのワークロード ネットワークの追加
- スーパーバイザー クラスタ の制御プレーン サイズの変更
- スーパーバイザー クラスタ の管理ネットワーク設定の変更
- VDS ネットワークが構成されている スーパーバイザー クラスタ のワークロード ネットワーク設定の変更
- NSX-T Data Center が構成されている スーパーバイザー クラスタ のワークロード ネットワーク設定の変更
- 初期構成またはアップグレード中の スーパーバイザー クラスタ の健全性ステータス エラーの解決
- vSphere with Tanzu での HTTP プロキシ設定の構成
- リモート rsyslog に対する スーパーバイザー クラスタ 制御プレーンのログ ストリーミング

vSphere クラスタで vSphere with Tanzu を構成するための前提条件

vSphere 環境で vSphere with Tanzu を有効にするための前提条件を確認してください。vSphere でコンテナ ベースのワークロードをネイティブに実行するには、vSphere 管理者として vSphere クラスタで [ワークロード管理] を有効にします。これにより、vSphere ポッド を実行し Tanzu Kubernetes クラスタおよび仮想マシンをプロビジョニングする、スーパーバイザー クラスタ と呼ばれる Kubernetes 管理クラスタが使用可能になります。

vSphere クラスタの作成と構成

vSphere クラスタは、vCenter Server システムによって管理される ESXi ホストの集合です。スーパーバイザー クラスタ は vSphere クラスタで実行されます。[ワークロード管理] を有効にできるように、次の要件を満たす vSphere クラスタを作成します。

- 3 台以上の ESXi ホストが含まれる vSphere クラスタを作成して、構成します。vSAN を使用している場合は、4 台の ESXi ホストを使用することが推奨されます（必須ではありません）。クラスタの作成と構成を参照してください。
- クラスタに vSAN などの共有ストレージを構成します。vSphere HA や DRS を使用するため、およびパーシステント コンテナ ボリュームを格納するためには、共有ストレージが必要です。vSAN クラスタの作成を参照してください。
- ReadWriteMany モードでパーシステント ボリュームを使用する場合は、vSAN クラスタでファイル サービスを有効にします。vSphere with Tanzu での ReadWriteMany パーシステント ボリュームの作成を参照してください。
- vSphere HA のクラスタを有効にします。vSphere HA クラスタの作成と使用を参照してください。
- 完全自動化モードで vSphere DRS のクラスタを有効にします。DRS クラスタの作成を参照してください。
- ユーザー アカウントに vSphere クラスタに対するクラスタ全体の構成の変更権限があって、ユーザーが[ワークロード管理] 機能を有効にできることを確認します。

注意： スーパーバイザー クラスタ を構成した後に vSphere DRS を無効にしないでください。DRS を常に有効にすることは、スーパーバイザー クラスタ でワークロードを実行するための必須の前提条件です。Tanzu Kubernetes DRS を無効にすると、クラスタが破損します。

ネットワーク スタックの選択と構成

vSphere クラスタで [ワークロード管理] を有効にするには、スーパーバイザー クラスタ で使用するネットワーク スタックを構成する必要があります。NSX-T Data Center ネットワーク、またはロード バランサを使用する vSphere Distributed Switch (vDS) ネットワークという 2 つのオプションがあります。NSX Advanced Load Balancer または HAProxy ロード バランサを構成できます。

次の表に、サポートされている 2 つのネットワーク スタックの違いについての概要を示します。アーキテクチャ上の違いについては、vSphere ネットワーク スタックを使用して構成された スーパーバイザー クラスタを参照してください。

機能	NSX-T ネットワーク	Distributed Switch ネットワーク
vSphere ポッド	はい	いいえ
Tanzu Kubernetes クラスタ	はい	はい
組み込み Harbor レジストリ	はい	いいえ
ロード バランシング	はい	あり。有効にするには、NSX Advanced Load Balancer または HAProxy ロード バランサをインストールして構成します。

スーパーバイザー クラスタ に NSX-T Data Center ネットワークを使用するには、次の操作を実行します。

- NSX-T ネットワークのシステム要件とトポロジを確認します。NSX-T Data Center を使用して vSphere with Tanzu をセットアップするためのシステム要件を参照してください。
- vSphere with Tanzu 用の NSX-T Data Center をインストールして、構成します。vSphere with Tanzu で使用する NSX-T Data Center のインストールと構成を参照してください。

スーパーバイザー クラスタ に、NSX Advanced Load Balancer を使用する vSphere Distributed Switch ネットワークを使用するには、次の操作を実行します。

- NSX Advanced Load Balancer の要件を確認します。vSphere ネットワークと NSX Advanced Load Balancer を使用して vSphere with Tanzu をセットアップするためのシステム要件を参照してください。
- vSphere Distributed Switch (vDS) を作成し、クラスタ内のすべての ESXi ホストを Distributed Switch に追加して、ワークロード ネットワーク用のポート グループを作成します。NSX Advanced Load Balancer で使用する スーパーバイザー クラスタ の vSphere Distributed Switch の作成を参照してください。
- NSX Advanced Load Balancer をデプロイして構成します。コントローラのデプロイを参照してください。

注： vSphere with Tanzu は、vSphere 7 U2 以降で NSX Advanced Load Balancer をサポートします。

スーパーバイザー クラスタ に、HAProxy ロード バランシングを使用する vSphere Distributed Switch ネットワークを使用するには、次の操作を実行します。

- 外部ロード バランサを使用する vSphere ネットワークのシステム要件とネットワーク トポロジを確認します。vSphere ネットワークおよび HAProxy ロード バランサを使用して vSphere with Tanzu をセットアップするためのシステム要件と HAProxy ロード バランサをデプロイするトポロジを参照してください。
- vSphere Distributed Switch (vDS) を作成し、クラスタ内のすべての ESXi ホストを Distributed Switch に追加して、ワークロード ネットワーク用のポート グループを作成します。HAProxy ロード バランサで使用する スーパーバイザー クラスタ の vSphere Distributed Switch の作成を参照してください。
- vSphere クラスタからホストに接続されている vSphere Distributed Switch にルーティング可能な HAProxy ロード バランサ インスタンスをインストールして、構成します。HAProxy ロード バランサは、クライアント ネットワークからワークロードへのネットワーク接続をサポートし、Tanzu Kubernetes クラスタ間のトラフィックのロード バランシングを行います。HAProxy ロード バランサのインストールと構成を参照してください。

注： vSphere with Tanzu は、vSphere 7 U1 以降で HAProxy ロード バランサをサポートします。

ストレージ ポリシーの作成

Kubernetes 制御プレーンの仮想マシン、コンテナ、イメージのデータストア配置を決定するストレージ ポリシーを作成する必要があります。さまざまなストレージ クラスに関連付けられたストレージ ポリシーを作成できます。

vSphere クラスタで [ワークロード管理] を有効にする前に、Kubernetes 制御プレーン仮想マシンを配置するためのストレージ ポリシーを作成します。vSphere with Tanzu のストレージ ポリシーの作成 を参照してください。

コンテンツライブラリの作成

Tanzu Kubernetes クラスタと仮想マシンをプロビジョニングするには、スーパーバイザー クラスタ が実行されている vSphere クラスタを管理する vCenter Server で、[コンテンツ ライブラリ] が作成されている必要があります。

[コンテンツ ライブラリ] は、OVA テンプレート形式の Tanzu Kubernetes リリース のディストリビューションをシステムに提供します。Tanzu Kubernetes クラスタをプロビジョニングするときには、選択したバージョンの OVA テンプレートを使用して、Kubernetes クラスタ ノードが作成されます。

[サブスクライブ済みコンテンツ ライブラリ] を作成して、リリースされた最新イメージを自動的にプルしたり、[ローカル コンテンツ ライブラリ] を作成して、イメージを手動でアップロードしたりできます。手動によるアップロードは、エアギャップ環境で Tanzu Kubernetes クラスタのプロビジョニングを行う場合に必要になることがあります。

Tanzu Kubernetes リリース のコンテンツ ライブラリの作成と管理を参照してください。

vSphere with Tanzu のデモの確認

厳しい要件ではありませんが、開始する前に、スーパーバイザー クラスタ のデプロイ準備における vSphere 環境の設定、[ワークロード管理] の有効化、Tanzu Kubernetes クラスタのプロビジョニングなどに関する vSphere with Tanzu のデモが役立つことがあります。役立つと思われる場合は、VMware vSphere チャンネルで、一連の vSphere with Tanzu の詳細情報のビデオを確認してください。また、vSphere Tanzu Quick Bytes という短いビデオのシリーズで、Distributed Switch ネットワークと HAProxy ロード バランサを使用して [ワークロード管理] を構成する方法を確認することもできます。

vSphere ネットワークを使用したワークロード管理の有効化

vSphere 管理者が vSphere クラスタ上の [ワークロード管理] プラットフォームを有効にするには、ワークロードに接続できるように vSphere ネットワーク スタックを構成する必要があります。vSphere ネットワークを使用するように構成された スーパーバイザー クラスタ では、Tanzu Kubernetes Grid サービス を使用して作成された Tanzu Kubernetes クラスタのデプロイがサポートされます。vSphere ポッド を実行したり、組み込みの Harbor レジストリ を使用したりすることはできません。

注意： スーパーバイザー クラスタ を構成した後に vSphere DRS を無効にしないでください。DRS を常に有効にすることは、スーパーバイザー クラスタ でワークロードを実行するための必須の前提条件です。Tanzu Kubernetes DRS を無効にすると、クラスタが破損します。

前提条件

- vSphere クラスタを スーパーバイザー クラスタ として構成するための前提条件を満たすこと。vSphere クラスタで vSphere with Tanzu を構成するための前提条件を参照してください。

手順

1 ホーム メニューから、[ワークロード管理] を選択します。

2 スーパーバイザー クラスタ のライセンス オプションを選択します。

- 有効な Tanzu エディション ライセンスを所有している場合は、[ライセンスの追加] をクリックして、vSphere のライセンス インベントリにライセンス キーを追加します。
- Tanzu エディション ライセンスをまだ所有していない場合は、VMware からの連絡を受信できるように、連絡先の詳細を入力してから、[開始する] をクリックします。

スーパーバイザー クラスタ の評価期間は、60 日間です。この期間内に、有効な Tanzu エディション ライセンスをクラスタに割り当てる必要があります。Tanzu エディション ライセンス キーを追加した場合は、スーパーバイザー クラスタ の設定を完了した後、60 日の評価期間内にそのキーを割り当てることができます。

3 [ワークロード管理] 画面で、[開始する] を再度クリックします。

4 vCenter Server システムを選択し、[vCenter Server ネットワーク] を選択して、[次へ] をクリックします。

5 互換性のあるクラスタのリストからクラスタを選択します。

6 [制御プレーンのサイズ] 画面で、クラスタの各ホスト上で作成される Kubernetes 制御プレーン仮想マシンのサイズを選択します。

スーパーバイザー クラスタ で管理できる Kubernetes ワークロードの数は、制御プレーン仮想マシンに割り当てるリソースの量によって決まります。

7 [ロード バランサ] 画面で、使用するロード バランサを選択します。NSX Advanced Load Balancer または HAProxy を選択できます。

- NSX Advanced Load Balancer の場合は次の設定を入力します。

オプション	説明
名前	NSX Advanced Load Balancer の名前を入力します。
Avi Controller の IP アドレス	NSX Advanced Load Balancer Controller の IP アドレス。 デフォルトのポートは 443 です。
ユーザー名	NSX Advanced Load Balancer を使用して構成されたユーザー名。このユーザー名は、コントローラへのアクセスに使用します。
パスワード	ユーザー名のパスワード。
サーバ認証局	コントローラによって使用される証明書。 構成時に割り当てた証明書を指定できます。 詳細については、コントローラへの証明書の割り当てを参照してください。

- HAProxy の場合は次の設定を入力します。

オプション	説明
名前	ロード バランサのわかりやすい名前。
データ プレーン API のアドレス	HAProxy データ プレーン API の IP アドレスとポート。このコンポーネントは、HAProxy サーバを制御し、HAProxy 仮想マシン内で実行されます。これは、HAProxy アプライアンスの管理ネットワーク IP アドレスです。
ユーザー名	HAProxy OVA ファイルを使用して構成されたユーザー名。この名前は、HAProxy データ プレーン API での認証に使用します。
パスワード	ユーザー名のパスワード。

オプション	説明
仮想サーバの IP アドレス範囲	Tanzu Kubernetes クラスタによってワークロード ネットワークで使用される IP アドレスの範囲。この IP アドレス範囲は、HAProxy アプライアンスのデプロイ時に構成した CIDR で定義された IP アドレスのリストから取得されます。通常、この範囲は、HAProxy のデプロイ時に指定された範囲全体になりますが、この CIDR のサブセットにすることもできます。複数のスーパーバイザー クラスタを作成し、CIDR 範囲内の複数の IP アドレスを使用することがあるためです。この範囲は、このウィザードでワークロード ネットワーク用に定義されている IP アドレス範囲と重複することはできません。また、この範囲は、このワークロード ネットワークの DHCP 範囲とも重複することができません。

サーバ認証局	署名済みの PEM 形式の証明書、またはデータ プレーン API によって提供されるサーバ証明書の信頼できるルートである PEM 形式の証明書。
	■ オプション 1：root アクセスが有効な場合に、HAProxy 仮想マシンに root として SSH 接続し、[サーバ認証局] に `/etc/haproxy/ca.crt` をコピーします。`\n` 形式のエスケープ行を使用しないでください。
	■ オプション 2：HAProxy 仮想マシンを右クリックし、[設定の編集] を選択します。適切なフィールドから CA 証明書をコピーし、https://www.base64decode.org/ などの変換ツールを使用して Base64 から変換します。
	■ オプション 3：次の PowerCLI スクリプトを実行します。変数 `$vc`、`$vc_user`、`$vc_password` の値を適切な値に置き換えます。
	`$vc = "10.21.32.43"`
	`$vc_user = "administrator@vsphere.local"`
	`$vc_password = "PASSWORD"`
	`Connect-VIServer -User $vc_user -Password $vc_password -Server $vc`
	`$VMname = "haproxy-demo"`
	`$AdvancedSettingName = "guestinfo.dataplaneapi.cacert"`
	`$Base64cert = get-vm $VMname |Get-AdvancedSetting -Name $AdvancedSettingName`
	`while ([string]::IsNullOrEmpty($Base64cert.Value)) {`
	`Write-Host "Waiting for CA Cert Generation... This may take a under 5-10 minutes as the VM needs to boot and generate the CA Cert (if you haven't provided one already)."`
	`$Base64cert = get-vm $VMname | Get-AdvancedSetting -Name $AdvancedSettingName`
	`Start-sleep -seconds 2`
	`}`
	`Write-Host "CA Cert Found... Converting from BASE64"`

オプション	説明
	`$cert = [Text.Encoding]::Utf8.GetString([Convert]::FromBase64String($Base64cert.Value))` `Write-Host $cert`

8 [管理ネットワーク] 画面で、Kubernetes 制御プレーン仮想マシンに使用されるネットワークのパラメータを構成します。

a [ネットワーク モード] を選択します。

- [DHCP ネットワーク]。このモードでは、制御プレーン仮想マシンの IP アドレス、フローティング IP アドレス、DNS サーバ、DNS、検索ドメイン、NTP サーバなど、管理ネットワークのすべての IP アドレスが DHCP サーバから自動的に取得されます。フローティング IP アドレスを取得するためには、クライアント ID をサポートするように DHCP サーバを構成する必要があります。DHCP モードでは、すべての制御プレーン仮想マシンが安定した DHCP クライアント ID を使用して IP アドレスを取得します。これらのクライアント ID を使用すると、DHCP サーバ上の制御プレーン仮想マシンの IP アドレスに対して固定 IP アドレス割り当てを設定して、IP アドレスが変更されないようにすることができます。制御プレーン仮想マシンの IP アドレスとフローティング IP アドレスの変更はサポートされていません。
- [固定]。管理ネットワークのすべてのネットワーク設定を手動で入力します。

b 管理ネットワークの設定を構成します。

DHCP ネットワーク モードを選択した場合に、DHCP から取得した設定をオーバーライドするには、[追加設定] をクリックして新しい値を入力します。固定ネットワーク モードを選択した場合は、管理ネットワーク設定の値を手動で入力します。

オプション	説明
ネットワーク	VMkernel アダプタが管理トラフィック用に構成されているネットワークを選択します。
開始制御 IP アドレス	Kubernetes 制御プレーン仮想マシンに連続する 5 つの IP アドレスを予約するための開始点を決定する IP アドレスを、次のように入力します。 ■ Kubernetes 制御プレーン仮想マシンそれぞれの IP アドレス。 ■ 管理ネットワークへのインターフェイスとして機能するいずれかの Kubernetes 制御プレーン仮想マシンのフローティング IP アドレス。フローティング IP アドレスが割り当てられた制御プレーン仮想マシンは、3 台すべての Kubernetes 制御プレーン仮想マシンの中で主要な仮想マシンとして機能します。フローティング IP アドレスは、Kubernetes クラスタ内の ectd リーダーである制御プレーン ノードに移動されます。これにより、ネットワーク パーティション イベントが発生した場合に、可用性が向上します。 ■ Kubernetes 制御プレーン仮想マシンで障害が発生し、新しい制御プレーン仮想マシンが引き継ぐため起動しているときにバッファとして機能する IP アドレス。
サブネット マスク	固定 IP 構成にのみ適用されます。管理ネットワークのサブネット マスクを入力します。たとえば、`255.255.255.0`。
DNS サーバ	環境内で使用する DNS サーバのアドレスを入力します。vCenter Server システムが FQDN で登録されている場合は、vSphere 環境で使用する DNS サーバの IP アドレスを入力して、スーパーバイザー クラスタ で FQDN を解決できるようにする必要があります。
DNS 検索ドメイン	DNS が Kubernetes 制御プレーン ノード内で検索するドメイン名（`corp.local` など）を入力して、DNS サーバで解決できるようにします。
NTP	環境内で使用する NTP サーバがある場合は、そのアドレスを入力します。

9 [ワークロード ネットワーク] 画面で、スーパーバイザー クラスタ 上で実行されている Kubernetes ワークロードのネットワーク トラフィックを処理するネットワークの設定を入力します。

注： ワークロード ネットワークのネットワーク設定に DHCP サーバを使用する場合、スーパーバイザー クラスタ の構成を完了した後に新しいワークロード ネットワークを作成することはできません。

a ネットワーク モードを選択します。

- [DHCP ネットワーク]。このネットワーク モードでは、ワークロード ネットワークのすべてのネットワーク設定が DHCP を介して取得されます。
- [固定]。ワークロード ネットワーク設定を手動で構成します。

b スーパーバイザー クラスタ に対してプライマリ ワークロード ネットワークとして機能するポート グループを選択します。

プライマリ ネットワークは、Kubernetes 制御プレーン仮想マシンのトラフィックと Kubernetes ワークロード トラフィックを処理します。

ネットワーク トポロジによっては、ネットワークとして機能する別のポート グループを後で各名前空間に割り当てることができます。これにより、スーパーバイザー クラスタ の名前空間の間でレイヤー 2 の隔離が可能になります。名前空間のネットワークとして別のポート グループが割り当てられていない場合は、プライマリ ネットワークが使用されます。Tanzu Kubernetes クラスタで使用されるネットワークは、そのクラスタがデプロイされた名前空間に割り当てられているネットワークのみです。その名前空間にネットワークが明示的に割り当てられていない場合は、プライマリ ネットワークが使用されます。

c ワークロード ネットワークの設定を構成します。

DHCP ネットワーク モードを選択した場合、[その他の設定] セクションのすべての値は DHCP サーバから自動的に入力されます。これらの値をオーバーライドする場合は、[その他の設定] をクリックして新しい値を入力します。[固定] ネットワーク モードを選択した場合は、すべての設定を手動で入力します。

オプション	説明
Kubernetes サービスの内部ネットワーク	Tanzu Kubernetes クラスタおよびクラスタ内で実行されるサービスの IP アドレスの範囲を決定する CIDR 表記を入力します。
ネットワーク名	ネットワーク名を入力します。
DNS サーバ	環境内で使用する DNS サーバがある場合は、その IP アドレスを入力します。 たとえば、**`10.142.7.1`** のように入力します。 DNS サーバの IP アドレスを入力すると、各制御プレーン仮想マシンにスタティック ルートが追加されます。これは、DNS サーバへのトラフィックがワークロード ネットワークを通過することを意味します。 指定した DNS サーバが管理ネットワークとワークロード ネットワークの間で共有されている場合、制御プレーン仮想マシンの DNS ルックアップは、初期セットアップ後にワークロード ネットワークを介してルーティングされます。
ゲートウェイ	プライマリ ネットワークのゲートウェイを入力します。

オプション	説明
サブネット マスク IP	サブネット マスク IP アドレスを入力します。
IP アドレス範囲	Kubernetes 制御プレーン仮想マシンおよびワークロードの IP アドレスを割り当てるために IP アドレス範囲を入力します。 このアドレス範囲は スーパーバイザー クラスタ ノードに接続します。単一のワークロード ネットワークを使用している場合も、Tanzu Kubernetes クラスタ ノードに接続します。HAProxy に [デフォルト] 構成を使用している場合、この IP アドレス範囲がロード バランサの VIP 範囲と重複することはできません。

10 [ストレージ] 画面で、ストレージおよびファイル ボリュームのサポートを構成します。

a スーパーバイザー クラスタ のストレージ ポリシーを選択します。

次のオブジェクトごとに選択したストレージ ポリシーによって、そのオブジェクトがストレージ ポリシーで参照されるデータストアに配置されます。各オブジェクトには、同じストレージ ポリシーを使用することも異なるストレージ ポリシーを使用することもできます。

オプション	説明
制御プレーン ノード	制御プレーン仮想マシンを配置するためのストレージ ポリシーを選択します。
ポッドの短期ディスク	vSphere ポッド を配置するためのストレージ ポリシーを選択します。
コンテナ イメージ キャッシュ	コンテナ イメージのキャッシュを配置するためのストレージ ポリシーを選択します。

b （オプション） ファイル ボリュームのサポートを有効にします。

クラスタに ReadWriteMany パーシステント ボリュームをデプロイする予定がある場合、このオプションは必須です。vSphere with Tanzu での ReadWriteMany パーシステント ボリュームの作成を参照してください。

11 [Tanzu Kubernetes Grid] 画面で [追加] をクリックし、Tanzu Kubernetes クラスタのノードをデプロイするための仮想マシン イメージを含む、サブスクライブ済みコンテンツ ライブラリを選択します。

12 設定を確認して、[終了] をクリックします。

結果

タスクは、スーパーバイザー クラスタ を作成する vCenter Server で実行されます。このタスクが完了すると、vSphere クラスタ内のホスト上に 3 台の Kubernetes 制御プレーン仮想マシンが作成されます。

次のステップ

スーパーバイザー クラスタ で vSphere 名前空間 を作成して構成します。vSphere 名前空間 の作成と設定を参照してください。

NSX-T Data Center ネットワークを使用したワークロード管理の有効化

vSphere 管理者は、NSX-T Data Center ネットワーク スタックを使用して Kubernetes ワークロードに接続する スーパーバイザー クラスタ として vSphere クラスタを構成できます。

前提条件

- 環境が vSphere クラスタを スーパーバイザー クラスタ として構成するための前提条件を満たしていることを確認します。要件の詳細については、vSphere クラスタで vSphere with Tanzu を構成するための前提条件を参照してください。

注意： スーパーバイザー クラスタ を構成した後に vSphere DRS を無効にしないでください。DRS を常に有効にすることは、スーパーバイザー クラスタ でワークロードを実行するための必須の前提条件です。Tanzu Kubernetes DRS を無効にすると、クラスタが破損します。

手順

1. vSphere Client ホーム メニューから、[ワークロード管理] を選択します。
2. [開始する] をクリックします。
3. 構成する vCenter Server システムを選択します。
4. [NSX] ネットワーク スタックを選択します。
5. [次へ] をクリックします。
6. [クラスタを選択] - [データセンター] の順に選択します。
7. 互換性のあるクラスタのリストからクラスタを選択し、[次へ] をクリックします。
8. [制御プレーンのサイズ] ページで、制御プレーン仮想マシンのサイズを選択します。

   制御プレーン仮想マシンのサイズによって、スーパーバイザー クラスタ で実行できるワークロードの数が決まります。

   ガイダンスについては、VMware 構成の上限サイトを参照してください。
9. [次へ] をクリックします。

10 [管理ネットワーク] 画面で、Kubernetes 制御プレーン仮想マシンに使用されるネットワークのパラメータを構成します。

a [ネットワーク モード] を選択します。

- [DHCP ネットワーク]。このモードでは、制御プレーン仮想マシンの IP アドレス、DNS サーバ、DNS、検索ドメイン、NTP サーバなど、管理ネットワークのすべての IP アドレスが DHCP から自動的に取得されます。
- [固定]。管理ネットワークのすべてのネットワーク設定を手動で入力します。

b 管理ネットワークの設定を構成します。

DHCP ネットワーク モードを選択した場合に、DHCP から取得した設定をオーバーライドするには、[追加設定] をクリックして新しい値を入力します。固定ネットワーク モードを選択した場合は、管理ネットワーク設定の値を手動で入力します。

オプション	説明
ネットワーク	VMkernel アダプタが管理トラフィック用に構成されているネットワークを選択します。
開始制御 IP アドレス	Kubernetes 制御プレーン仮想マシンに連続する 5 つの IP アドレスを予約するための開始点を決定する IP アドレスを、次のように入力します。 ■ Kubernetes 制御プレーン仮想マシンそれぞれの IP アドレス。 ■ 管理ネットワークへのインターフェイスとして機能するいずれかの Kubernetes 制御プレーン仮想マシンのフローティング IP アドレス。フローティング IP アドレスが割り当てられた制御プレーン仮想マシンは、3 台すべての Kubernetes 制御プレーン仮想マシンの中で主要な仮想マシンとして機能します。フローティング IP アドレスは、この Kubernetes クラスタ内の ectd リーダーであり スーパーバイザー クラスタ でもある制御プレーン ノードに移動されます。これにより、ネットワーク パーティション イベントが発生した場合に、可用性が向上します。 ■ Kubernetes 制御プレーン仮想マシンで障害が発生し、新しい制御プレーン仮想マシンが引き継ぐため起動しているときにバッファとして機能する IP アドレス。
サブネット マスク	固定 IP 構成にのみ適用されます。管理ネットワークのサブネット マスクを入力します。たとえば、`255.255.255.0`。
DNS サーバ	環境内で使用する DNS サーバのアドレスを入力します。vCenter Server システムが FQDN で登録されている場合は、vSphere 環境で使用する DNS サーバの IP アドレスを入力して、スーパーバイザー クラスタ で FQDN を解決できるようにする必要があります。
DNS 検索ドメイン	DNS が Kubernetes 制御プレーン ノード内で検索するドメイン名（`corp.local` など）を入力して、DNS サーバで解決できるようにします。
NTP	環境内で使用する NTP サーバがある場合は、そのアドレスを入力します。

11 [ワークロード ネットワーク] ペインで、名前空間のネットワークの設定を構成します。

名前空間ネットワークの設定により、スーパーバイザー クラスタ で実行されている vSphere ポッド および名前空間への接続が確立されます。デフォルトでは、名前空間はクラスタ レベルのネットワーク構成を使用します。

オプション	説明
vSphere Distributed Switch	スーパーバイザー クラスタ のオーバーレイ ネットワークを処理する vSphere Distributed Switch を選択します。 たとえば、`DSwitch` を選択します。
DNS サーバ	環境内で使用する DNS サーバがある場合は、その IP アドレスを入力します。 たとえば、`10.142.7.1` です。
API サーバ エンドポイントの FQDN	必要に応じて、API サーバ エンドポイントの FQDN を入力します。
Edge クラスタ	名前空間ネットワークに使用する Tier-0 ゲートウェイを持つ NSX Edge クラスタを選択します。 たとえば、`EDGE-CLUSTER` を選択します。
Tier-0 ゲートウェイ	クラスタの Tier-1 ゲートウェイに関連付ける Tier-0 ゲートウェイを選択します。
NAT モード	NAT モードは、デフォルトで選択されています。 このオプションを選択解除すると、vSphere ポッド、仮想マシン、Tanzu Kubernetes クラスタ ノードの IP アドレスなどのワークロードがいずれも Tier-0 ゲートウェイの外から直接アクセスできるようになります。出力方向の CIDR を構成する必要はありません。 **注：** NAT モードを選択解除すると、ファイル ボリューム ストレージはサポートされません。
名前空間ネットワーク	1 つ以上の IP CIDR を入力してサブネット/セグメントを作成し、ワークロードに IP アドレスを割り当てます。
名前空間サブネット プリフィックス	名前空間セグメント用に予約されるサブネットのサイズを指定する、サブネット プリフィックスを入力します。デフォルトは 28 です。
ポッド CIDR	vSphere Native Pod の IP アドレス範囲を決定する CIDR 注釈を入力します。デフォルト値を使用できます。
サービス CIDR	Kubernetes サービスの IP アドレス範囲を決定する CIDR 注釈を入力します。デフォルト値を使用できます。
入力方向 CIDR	Kubernetes サービスの入力方向 IP アドレス範囲を決定する CIDR 注釈を入力します。この範囲は、タイプがロード バランサで入力方向のサービスに使用されます。
出力方向 CIDR	Kubernetes サービスの出力方向 IP アドレスを決定する CIDR 注釈を入力します。スーパーバイザー クラスタ 内の名前空間ごとに 1 つの出力方向 IP アドレスのみが割り当てられます。出力方向 IP アドレスは、特定の名前空間内の vSphere ポッド が NSX-T Data Center の外部と通信するために使用する IP アドレスです。

12 [次へ] をクリックします。

13 [ストレージ] 画面で、ストレージおよびファイル ボリュームのサポートを構成します。

a スーパーバイザー クラスタ のストレージ ポリシーを選択します。

次のオブジェクトごとに選択したストレージ ポリシーによって、そのオブジェクトがストレージ ポリシーで参照されるデータストアに配置されます。各オブジェクトには、同じストレージ ポリシーを使用することも異なるストレージ ポリシーを使用することもできます。

オプション	説明
制御プレーン ノード	制御プレーン仮想マシンを配置するためのストレージ ポリシーを選択します。
ポッドの短期ディスク	vSphere ポッド を配置するためのストレージ ポリシーを選択します。
コンテナ イメージ キャッシュ	コンテナ イメージのキャッシュを配置するためのストレージ ポリシーを選択します。

b （オプション） ファイル ボリュームのサポートを有効にします。

クラスタに ReadWriteMany パーシステント ボリュームをデプロイする予定がある場合、このオプションは必須です。vSphere with Tanzu での ReadWriteMany パーシステント ボリュームの作成を参照してください。

14 [設定の確認] セクションで設定を確認し、[完了] をクリックします。

クラスタで vSphere with Tanzu が有効になり、DevOps エンジニアに提供する vSphere 名前空間 を作成できるようになります。クラスタに含まれているホストに Kubernetes 制御プレーン ノードが作成され、Spherelet プロセスも作成されます。

次のステップ

スーパーバイザー クラスタ で vSphere 名前空間 を作成して構成します。『vSphere 名前空間 の作成と設定』を参照してください

スーパーバイザー クラスタ への Tanzu Edition ライセンスの割り当て

スーパーバイザー クラスタ を評価モードで使用している場合は、60 日の評価期間が終了する前に、クラスタに Tanzu エディション ライセンスを割り当てる必要があります。

Tanzu ライセンスの仕組みについては、vSphere with Tanzu のライセンスを確認してください。

手順

1 vSphere Client で、スーパーバイザー クラスタ に移動します。

2 [設定] を選択し、[ライセンス] で [スーパーバイザー クラスタ] を選択します。

3 [ライセンスの割り当て] を選択します。

4 [ライセンスの割り当て] ダイアログで、[新規ライセンス] をクリックします。

5 有効なライセンス キーを入力し、[OK] をクリックします。

スーパーバイザー クラスタ API エンドポイントに安全に接続するための VIP 証明書の置き換え

vSphere 管理者は、仮想 IP アドレス (VIP) の証明書を置き換えて、ホストがすでに信頼している CA で署名された証明書を使用して、スーパーバイザー クラスタ API エンドポイントに安全に接続することができます。証明書は、ログイン時と以降の スーパーバイザー クラスタ の操作の両方で、DevOps エンジニアに対して Kubernetes 制御プレーンを認証します。

前提条件

CSR に署名できる認証局 (CA) へのアクセス権があることを確認します。DevOps エンジニアの場合、認証局 (CA) を信頼できるルートとしてシステムにインストールする必要があります。

手順

1. vSphere Client で、スーパーバイザー クラスタ に移動します。
2. [構成] をクリックし、[名前空間] で [証明書] を選択します。
3. [ワークロード プラットフォーム MTG] ペインで、[アクション] - [CSR の生成] の順に選択します。
4. 証明書の詳細を入力します。
5. CSR が生成されたら、[コピー] をクリックします。
6. 認証局 (CA) を使用して証明書に署名します。
7. [ワークロード プラットフォーム MTG] ペインで、[アクション] - [証明書の置き換え] の順に選択します。
8. 署名付き証明書ファイルをアップロードし、[証明書の置き換え] をクリックします。
9. Kubernetes 制御プレーンの IP アドレスで証明書を検証します。

   たとえば、vSphere 向け Kubernetes CLI Tools のダウンロード画面を開き、ブラウザを使用して証明書が正常に置き換えられたことを確認できます。Linux または UNIX システムでは、`echo | openssl s_client -connect https://ip:6443` も使用できます。

スーパーバイザー クラスタ の Tanzu Kubernetes Grid サービスと Tanzu Mission Control の統合

スーパーバイザー クラスタ で実行される Tanzu Kubernetes Grid サービス と Tanzu Mission Control を統合できます。統合により、Tanzu Mission Control を使用した Tanzu Kubernetes クラスタのプロビジョニングと管理が可能になります。

Tanzu Mission Control の詳細については、Tanzu Kubernetes クラスタのライフサイクルの管理を参照してください。デモを見るには、Tanzu Mission Control Integrated with Tanzu Kubernetes Grid Service のビデオを確認してください。

スーパーバイザー クラスタ の Tanzu Mission Control 名前空間の表示

vSphere with Tanzu v7.0.1 U1 以降には Tanzu Mission Control 用の vSphere 名前空間 が付属しています。この名前空間は、Tanzu Mission Control エージェントをインストールする スーパーバイザー クラスタ 上に存在します。エージェントをインストールすると、Tanzu Mission Control の Web インターフェイスを使用して Tanzu Kubernetes クラスタをプロビジョニングおよび管理できるようになります。

1 kubectl 向けの vSphere プラグイン を使用して、スーパーバイザー クラスタ での認証を行います。vCenter Single Sign-On ユーザーとして スーパーバイザー クラスタ に接続するを参照してください。

2 次のように、コンテキストを スーパーバイザー クラスタ に切り替えます。

```
kubectl config use-context 10.199.95.59
```

3 次のコマンドを実行して、名前空間を一覧表示します。

```
kubectl get ns
```

4 Tanzu Mission Control に提供される vSphere 名前空間 は `svc-tmc-cXX` として識別されます（XX は数字）。

5 この名前空間に Tanzu Mission Control エージェントをインストールします。スーパーバイザー クラスタ への Tanzu Mission Control エージェントのインストールを参照してください。

スーパーバイザー クラスタ への Tanzu Mission Control エージェントのインストール

Tanzu Kubernetes Grid サービス を Tanzu Mission Control と統合するには、スーパーバイザー クラスタ にエージェントをインストールします。

注： 次の手順では、スーパーバイザー クラスタ バージョン 1.21.0 以降を使用する vSphere 7.0 U3 が必要です。

1 Tanzu Mission Control Web インターフェイスを使用して、スーパーバイザー クラスタ を Tanzu Mission Control に登録します。管理クラスタの Tanzu Mission Control への登録を参照してください。

2 Tanzu Mission Control Web インターフェイスを使用して、[管理] - [管理クラスタ] に移動して、登録 URL を取得します。

3 vSphere with Tanzu 環境で、Tanzu Mission Control で必要となるポート（通常は `443`）用のファイアウォール ポートを開きます。クラスタ エージェント拡張機能によって確立される送信接続を参照してください。

4 vSphere Client を使用して、vSphere with Tanzu 環境にログインします。

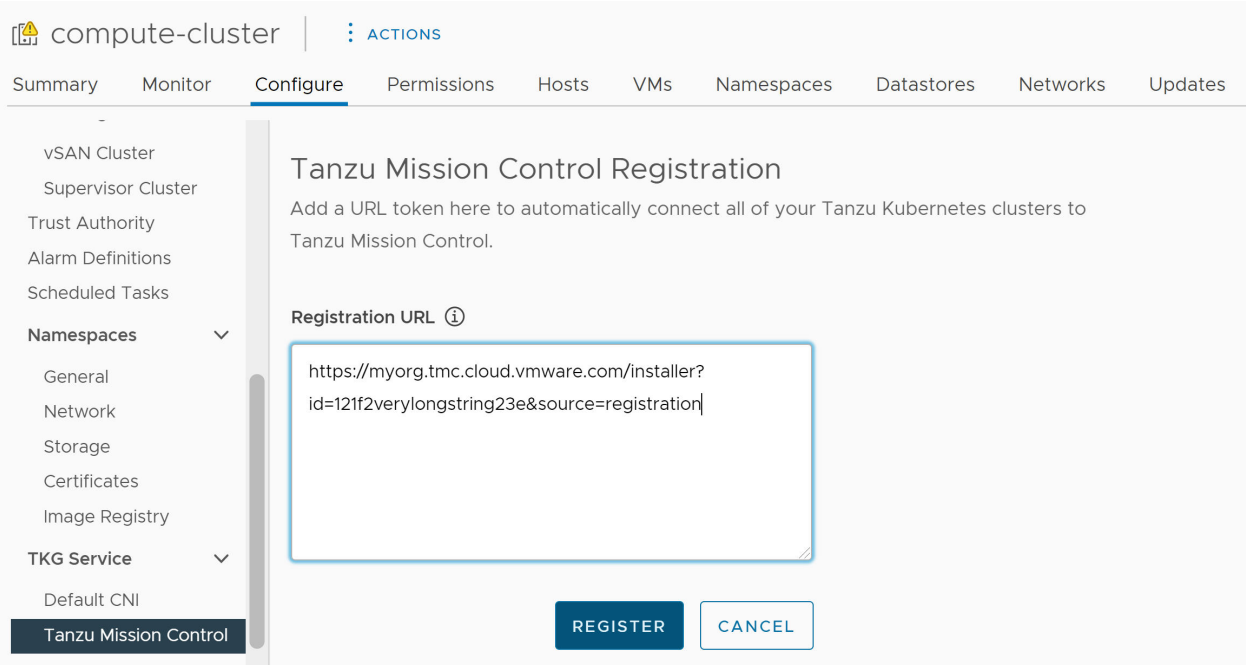
5 [ワークロード管理] が有効になっている vCenter Server クラスタを指定します。

6 [構成] タブを選択します。

7 [TKG サービス] - [Tanzu Mission Control] の順に選択します。

8 [登録 URL] フィールドで登録 URL を指定します。

9 [登録] をクリックします。



Tanzu Mission Control エージェントのアンインストール

スーパーバイザー クラスタ から Tanzu Mission Control エージェントをアンインストールするには、vSphere with Tanzu におけるスーパーバイザー クラスタからのクラスタ エージェントの手動削除を参照してください。

Tanzu Kubernetes クラスタのデフォルト CNI の設定

vSphere 管理者は、Tanzu Kubernetes クラスタのデフォルトのコンテナ ネットワーク インターフェイス (CNI) を設定できます。

デフォルトの CNI

Tanzu Kubernetes Grid サービス は、Tanzu Kubernetes クラスタについて、Antrea と Calico の 2 つの CNI オプションをサポートしています。

システムで定義されているデフォルトの CNI は Antrea です。デフォルトの CNI 設定の詳細については、Tanzu Kubernetes Grid サービス v1alpha1 API の構成パラメータを参照してください。

デフォルトの CNI は、vSphere Client を使用して変更できます。デフォルトの CNI を設定するには、次の手順を実行します。

注意： デフォルトの CNI の変更は、グローバルに行われます。新しく設定されたデフォルトは、サービスによって作成されたすべての新規クラスタに適用されます。既存のクラスタは変更されません。

---

1 vSphere Client を使用して、vSphere with Tanzu 環境にログインします。

2 ワークロード管理が有効になっている vCenter Server クラスタを選択します。

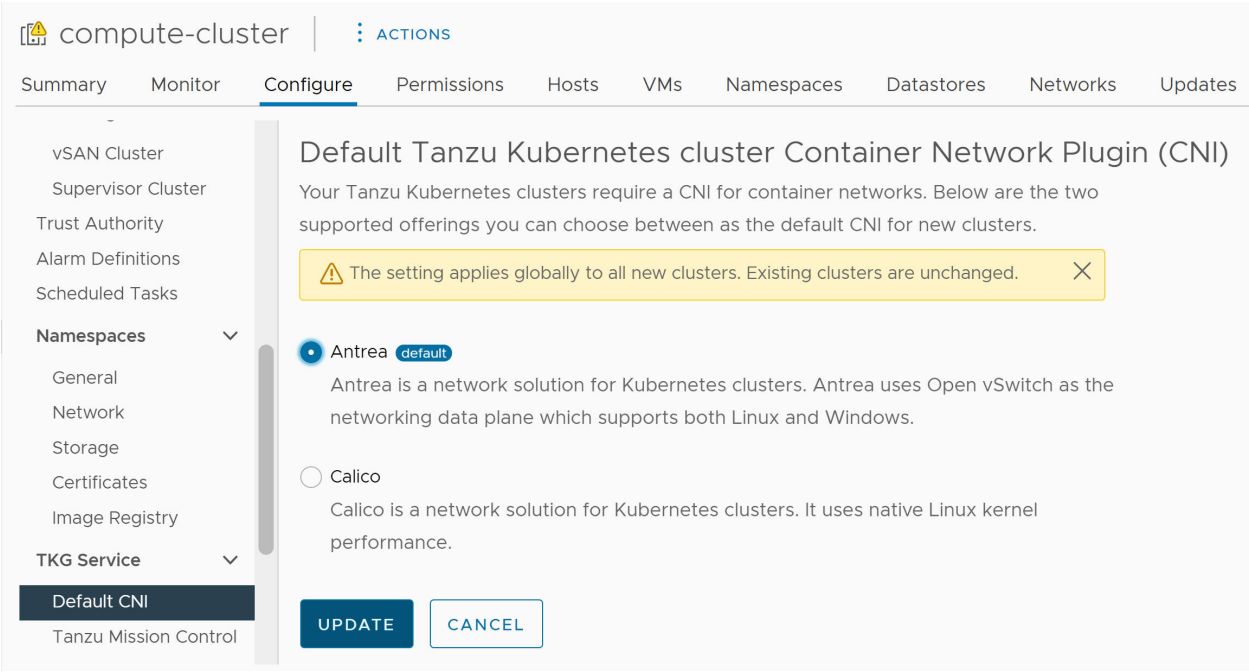
3 [構成] タブを選択します。

4 [TKG サービス] - [デフォルトの CNI] の順に選択します。

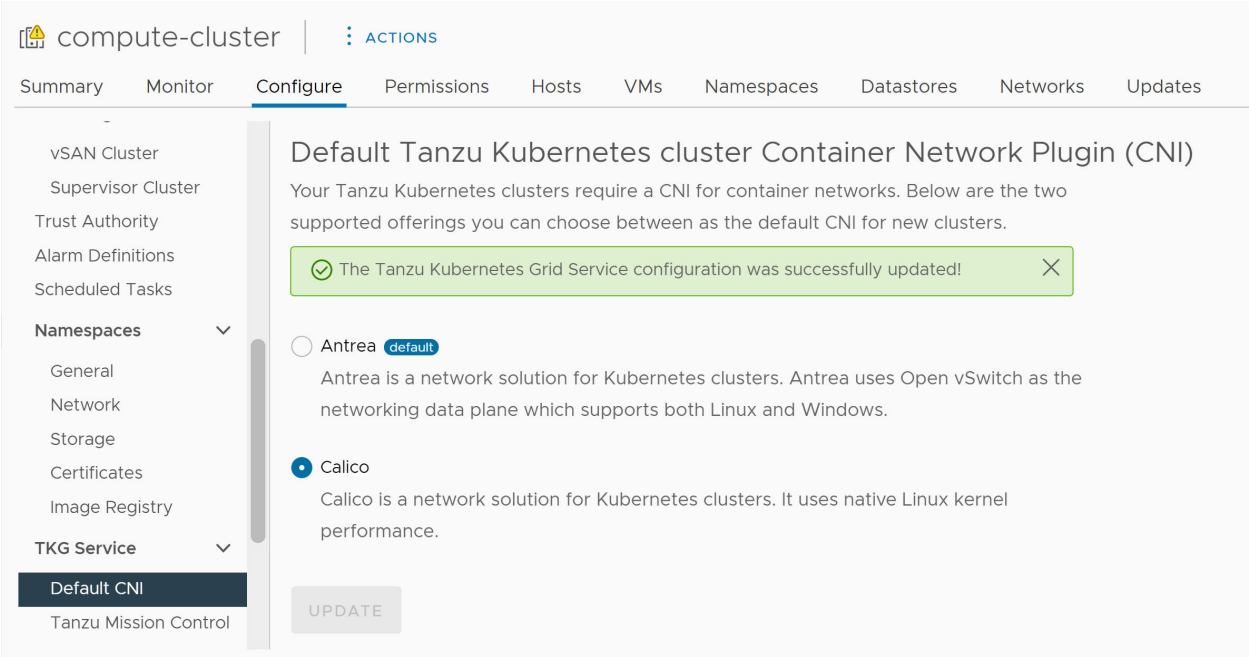
5　新規のクラスタに対してデフォルトの CNI を選択します。

6　[更新] をクリックします。

次の図は、デフォルトの CNI が選択されている様子を示しています。



次の図は、CNI の選択が Antrea から Calico に変更された様子を示しています。



VDS ネットワークが構成されている スーパーバイザー クラスタ へのワークロード ネットワークの追加

スーパーバイザー クラスタ に vSphere ネットワーク スタックが構成されている場合は、ワークロード ネットワークを作成して名前空間に割り当てることにより、Kubernetes ワークロードに対するレイヤー 2 の隔離を実現できます。ワークロード ネットワークでは、名前空間内の Tanzu Kubernetes クラスタに接続できます。また、ワークロード ネットワークは、スーパーバイザー クラスタ 内のホストに接続されているスイッチの分散ポート グループによってバッキングされます。

スーパーバイザー クラスタ に実装できるトポロジの詳細については、vSphere ネットワークと NSX Advanced Load Balancer を使用した スーパーバイザー クラスタ のトポロジ または HAProxy ロード バランサをデプロイするトポロジ。

注： ワークロード ネットワークのネットワーク設定を割り当てる DHCP サーバを スーパーバイザー クラスタ に構成している場合には、スーパーバイザー クラスタ の構成後に新しいワークロード ネットワークを作成することはできません。

前提条件

- ワークロード ネットワークをバッキングする分散ポート グループを作成します。
- ワークロード ネットワークに割り当てる IP アドレス範囲が、環境内で使用可能なすべての スーパーバイザー クラスタ 内で一意であることを確認します。

手順

1. vSphere Client で、スーパーバイザー クラスタ に移動します。
2. [構成] を選択します。
3. [スーパーバイザー クラスタ] で [ネットワーク] を選択します。
4. [ワークロード ネットワーク] を選択して、[追加] をクリックします。

オプション	説明
ポート グループ	このワークロード ネットワークに関連付ける分散ポート グループを選択します。[スーパーバイザー クラスタ] ネットワーク用に構成された vSphere Distributed Switch (VDS) には、選択元となるポート グループが含まれています。
ネットワーク名	名前空間に割り当てられるときにワークロード ネットワークを識別するネットワーク名。この値は、選択したポート グループの名前から自動的に入力されますが、必要に応じて変更できます。
IP アドレス範囲	Tanzu Kubernetes クラスタ ノードに割り当てる IP アドレス範囲を入力します。IP アドレス範囲は、サブネット マスクによって示されるサブネットに含まれている必要があります。 **注：** ワークロード ネットワークごとに一意の IP アドレス範囲を使用する必要があります。複数のネットワークに同じ IP アドレス範囲を構成しないでください。

オプション	説明
サブネット マスク	ポート グループのネットワークのサブネット マスクの IP アドレスを入力します。
ゲートウェイ	ポート グループ上のネットワークのデフォルト ゲートウェイを入力します。ゲートウェイは、サブネット マスクによって示されるサブネットに含まれている必要があります。 注： HAProxy ロードバランサーに割り当てられたゲートウェイは使用しないでください。

5 [追加] をクリックします。

次のステップ

新しく作成したワークロード ネットワークを vSphere 名前空間 に割り当てます。

スーパーバイザー クラスタ の制御プレーン サイズの変更

vSphere with Tanzu 環境にある スーパーバイザー クラスタ の Kubernetes 制御プレーン仮想マシンのサイズを変更する方法を確認します。

前提条件

- クラスタに関するクラスタ全体の構成の変更権限があることを確認します。

手順

1 vSphere Client で、スーパーバイザー クラスタ に移動します。

2 [構成] を選択して、[全般] をクリックします。

3 [制御プレーンのサイズ] を展開して [編集] をクリックし、ドロップダウン メニューから新しい制御プレーンのサイズを選択します。

4 [保存] をクリックします。

制御プレーンのサイズは、スケール アップのみが可能です。

スーパーバイザー クラスタ の管理ネットワーク設定の変更

vSphere with Tanzu 環境で スーパーバイザー クラスタ 管理ネットワークの DNS および NTP 設定を更新する方法について説明します。

前提条件

- クラスタに関するクラスタ全体の構成の変更権限があることを確認します。

手順

1 vSphere Client で、スーパーバイザー クラスタ に移動します。

2 [構成] を選択します。

3 [スーパーバイザー クラスタ] で [ネットワーク] を選択します。

4 [管理ネットワーク] をクリックします。

5 DNS および NTP 設定を編集します。

オプション	説明
DNS サーバ	環境内で使用する DNS サーバのアドレスを入力します。vCenter Server システムが FQDN で登録されている場合は、vSphere 環境で使用する DNS サーバの IP アドレスを入力して、スーパーバイザー クラスタ で FQDN を解決できるようにする必要があります。
DNS 検索ドメイン	DNS が Kubernetes 制御プレーン ノード内で検索するドメイン名（`corp.local` など）を入力して、DNS サーバで解決できるようにします。
NTP サーバ	環境内で使用する NTP サーバがある場合は、そのアドレスを入力します。

VDS ネットワークが構成されている スーパーバイザー クラスタ の ワークロード ネットワーク設定の変更

VDS ネットワーク スタックが構成されている スーパーバイザー クラスタ のワークロード ネットワークについて、その NTP および DNS サーバ設定を変更する方法を確認します。ワークロード ネットワーク用に構成する DNS サーバは、Kubernetes ワークロードに公開される外部 DNS サーバであり、スーパーバイザー クラスタ の外部でホストされるデフォルトのドメイン名を解決します。

前提条件

- クラスタに関するクラスタ全体の構成の変更権限があることを確認します。

手順

1 vSphere Client で、スーパーバイザー クラスタ に移動します。

2 [構成] を選択します。

3 [スーパーバイザー クラスタ] で [ネットワーク] を選択します。

4 [ワークロード ネットワーク] を選択します。

5 DNS サーバ設定を編集します。

vCenter Server などの vSphere 管理コンポーネントのドメイン名を解決できる DNS サーバのアドレスを入力します。

たとえば、**`10.142.7.1`** のように入力します。

DNS サーバの IP アドレスを入力すると、各制御プレーン仮想マシンにスタティック ルートが追加されます。これは、DNS サーバへのトラフィックがワークロード ネットワークを通過することを意味します。

指定した DNS サーバが管理ネットワークとワークロード ネットワークの間で共有されている場合、制御プレーン仮想マシンの DNS ルックアップは、初期セットアップ後にワークロード ネットワークを介してルーティングされます。

6 必要に応じて NTP 設定を編集します。

NSX-T Data Center が構成されている スーパーバイザー クラスタのワークロード ネットワーク設定の変更

ネットワーク スタックとして NSX-T Data Center 用に構成された スーパーバイザー クラスタ の DNS サーバ、名前空間ネットワーク、入力方向と出力方向のネットワーク設定を変更する方法について説明します。

前提条件

- クラスタに関するクラスタ全体の構成の変更権限があることを確認します。

手順

1. vSphere Client で、スーパーバイザー クラスタ に移動します。
2. [構成] を選択します。
3. [スーパーバイザー クラスタ] で [ネットワーク] を選択します。
4. [ワークロード ネットワーク] を選択します。
5. 必要に応じてネットワーク設定を変更します。

オプション	説明
DNS サーバ	vCenter Server などの vSphere 管理コンポーネントのドメイン名を解決できる DNS サーバのアドレスを入力します。 たとえば、`10.142.7.1` です。 DNS サーバの IP アドレスを入力すると、各制御プレーン仮想マシンにスタティック ルートが追加されます。これは、DNS サーバへのトラフィックがワークロード ネットワークを通過することを意味します。 指定した DNS サーバが管理ネットワークとワークロード ネットワークの間で共有されている場合、制御プレーン仮想マシンの DNS ルックアップは、初期セットアップ後にワークロード ネットワークを介してルーティングされます。
名前空間ネットワーク	スーパーバイザー クラスタ の名前空間セグメントに接続された Kubernetes ワークロードの IP アドレス範囲を変更する CIDR 注釈を入力します。NAT モードが構成されていない場合、この IP アドレス CIDR 範囲は、ルーティング可能な IP アドレスでなければなりません。
入力方向	Kubernetes サービスの入力方向 IP アドレス範囲を変更する CIDR 注釈を入力します。この範囲は、タイプがロード バランサで入力方向のサービスに使用されます。Tanzu Kubernetes クラスタの場合、ServiceType ロードバランサを介してサービスを公開すると、この IP CIDR ブロックから IP アドレスも取得されます。 **注:** 入力方向とワークロードのネットワーク フィールドに CIDR を追加することのみが可能です。既存のフィールドを編集または削除することはできません。
出力方向	スーパーバイザー クラスタ から出て外部サービスにアクセスするトラフィック用の SNAT（送信元ネットワーク アドレス変換）の IP アドレスを割り当てるための CIDR 注釈を入力します。スーパーバイザー クラスタ内のそれぞれの名前空間には、出力方向 IP アドレスが 1 つのみ割り当てられます。出力方向 IP アドレスは、特定の名前空間内の vSphere ポッド が NSX-T Data Center の外部と通信するために使用する IP アドレスです。

初期構成またはアップグレード中の スーパーバイザー クラスタ の健全性ステータス エラーの解決

最初に vSphere クラスタを スーパーバイザー クラスタ として構成するか、既存の スーパーバイザー クラスタ の設定をアップグレードまたは編集すると、指定したすべての設定が検証され、構成が完了するまでクラスタに適用されます。健全性チェックは、入力したパラメータに対して実行されます。これにより、構成内のエラーが検出されて、スーパーバイザー クラスタ の健全性ステータスがエラーになることがあります。これらの健全性ステータス エラーを解決して、スーパーバイザー クラスタ の構成またはアップグレードを再開できるようにする必要があります。

vSphere Client の スーパーバイザー クラスタ の健全性ステータスは、[ワークロード管理] - [スーパーバイザー クラスタ] に表示されます。クラスタ構成のステータスは [構成ステータス] 列に表示されます。

表 5-1. vCenter Server 接続エラー

エラー メッセージ	原因	解決方法
制御プレーン仮想マシン <*仮想マシン名*> に構成された管理 DNS サーバで vCenter Server プライマリ ネットワーク識別子 <*FQDN*> を解決できません。管理 DNS サーバ <サーバ名> で <*ネットワーク名*> が解決できることを検証してください。	■ 1 台以上の管理 DNS サーバにアクセスできます。 ■ 1 つの管理 DNS が静的に提供されています。 ■ 管理 DNS サーバには、vCenter Server PNID のホスト名ルックアップ機能がありません。 ■ vCenter Server PNID が固定 IP アドレスではなく、ドメイン名です。	■ vCenter Server PNID のホスト エントリを管理 DNS サーバに追加します。 ■ 構成された DNS サーバが正しいことを確認します。
制御プレーン仮想マシン <*仮想マシン名*> の管理ネットワーク上の DHCP を介して取得した DNS サーバで vCenter Server プライマリ ネットワーク識別子 <*ネットワーク名*> を解決できません。管理 DNS サーバで <*ネットワーク名*> を解決できることを検証してください。	■ DHCP サーバによって提供される管理 DNS サーバ（1 台以上）にアクセスできます。 ■ 管理 DNS サーバが静的に提供されています。 ■ 管理 DNS サーバには、vCenter Server PNID のホスト名ルックアップ機能がありません。 ■ 管理 DNS サーバには、vCenter Server PNID のホスト名ルックアップ機能がありません。 ■ vCenter Server PNID が固定 IP アドレスではなく、ドメイン名です。	■ 構成された DHCP サーバによって提供される管理 DNS サーバに、vCenter Server PNID のホスト エントリを追加します。 ■ DHCP サーバによって提供される DNS サーバが正しいことを確認します。
管理 DNS サーバが構成されていないため、制御プレーン仮想マシン <*仮想マシン名*> でホスト <*ホスト名*> を解決できません。	■ vCenter Server PNID が固定 IP アドレスではなく、ドメイン名です。 ■ DNS サーバが構成されていません。	管理 DNS サーバを構成します。
制御プレーン仮想マシン <*仮想マシン名*> でホスト <*ホスト名*> を解決できません。ホスト名の末尾のトップ レベル ドメインが「.local」であるため、管理 DNS 検索ドメインに「local」を含める必要があります。	vCenter Server PNID にはトップ レベル ドメイン (TLD) として **.local** が含まれていますが、構成された検索ドメインには **local** が含まれていません。	管理 DNS 検索ドメインに **local** を追加します。

表 5-1. vCenter Server 接続エラー （続き）

エラー メッセージ	原因	解決方法
制御プレーン仮想マシン <*仮想マシン名*> から管理 DNS サーバ <*サーバ名*> に接続できません。接続は、ワークロード ネットワークを使用して試行されました。	■ 管理 DNS サーバを vCenter Server に接続できません。 ■ 指定した `worker_dns` の値には、指定した管理 DNS の値が完全に含まれています。つまり、スーパーバイザー クラスタは固定トラフィックをこれらの IP アドレスに送信するためのネットワーク インターフェイスを 1 つ選択する必要があるため、トラフィックはワークロード ネットワークを介してルーティングされます。	■ ワークロード ネットワークを調べて、構成された管理 DNS サーバにルーティングできることを確認します。 ■ 競合する IP アドレスがないため、ワークロード ネットワーク上の DNS サーバと他のサーバ間で代替ルーティングがトリガされる可能性がないことを確認します。 ■ 構成された DNS サーバが実際には DNS サーバであり、その DNS ポートがポート 53 でホストされていることを確認します。 ■ ワークロード DNS サーバが、制御プレーン仮想マシンの IP アドレス（ワークロード ネットワークによって提供される IP アドレス）からの接続を許可するように構成されていることを確認します。 ■ 管理 DNS サーバのアドレスに誤りがないことを確認します。 ■ 検索ドメインに、不要な「~」が含まれていないことを確認します。これは、ホスト名の誤った解決の原因になる可能性があります。
制御プレーン仮想マシン <*仮想マシン名*> から管理 DNS サーバ <*サーバ名*> に接続できません。	DNS サーバに接続できません。	■ 管理ネットワークを調べて、管理 DNS サーバのルートが存在することを確認します。 ■ 競合する IP アドレスがないため、DNS サーバと他のサーバ間で代替ルーティングがトリガされる可能性がないことを確認します。 ■ 構成された DNS サーバが実際には DNS サーバであり、その DNS ポートがポート 53 でホストされていることを確認します。 ■ 管理 DNS サーバが、制御プレーン仮想マシンの IP アドレスからの接続を許可するように構成されていることを確認します。 ■ 管理 DNS サーバのアドレスに誤りがないことを確認します。 ■ 検索ドメインに、不要な「~」が含まれていないことを確認します。これは、ホスト名の誤った解決の原因になる可能性があります。

表 5-1. vCenter Server 接続エラー （続き）

エラー メッセージ	原因	解決方法
制御プレーン仮想マシン <仮想マシン名> から <*コンポーネント名*> <*コンポーネント アドレス*> に接続できません。エラー: *エラー メッセージ テキスト*	■ 一般的なネットワーク障害が発生しました。 ■ vCenter Server への実際の接続中にエラーが発生しました。	■ vCenter Server、HAProxy、NSX Manager、NSX Advanced Load Balancer などの構成済みコンポーネントのホスト名または IP アドレスが正しいことを検証します。 ■ 管理ネットワーク上の競合する IP アドレス、ファイアウォール ルールなどの外部ネットワーク設定を検証します。
制御プレーン仮想マシン <*仮想マシン名*> が vCenter Server <*vCenter Server 名*> 証明書を検証できません。vCenter Server 証明書は無効です。	vCenter Server から提供された証明書が無効な形式であるため、信頼できません。	■ `wcpsvc` を再起動して、制御プレーン仮想マシンの信頼されたルートのバンドルが最新の vCenter Server ルート証明書を含む最新バンドルであることを確認します。 ■ vCenter Server 証明書が実際に有効な証明書であることを確認します。
制御プレーン仮想マシン <*仮想マシン名*> は vCenter Server <*vCenter Server 名*> 証明書を信頼していません。	■ vCenter Server によって提示される `vmca.pem` 証明書は、制御プレーン仮想マシンに構成されているものとは異なります。 ■ vCenter Server アプライアンスの信頼されたルート証明書が置き換えられましたが、`wcpsvc` は再起動しませんでした。	■ `wcpsvc` を再起動して、制御プレーン仮想マシンの信頼されたルートのバンドルが最新の vCenter Server 証明書のルートを含む最新バンドルであることを確認します。

表 5-2. NSX Manager 接続エラー

制御プレーン仮想マシン <*仮想マシン名*> が NSX サーバ <*NSX サーバ名*> の証明書を検証できませんでした。サーバ <*NSX-T アドレス*> から返されたサムプリントが、vCenter Server <*vCenter Server 名*> に登録された、想定されるクライアント証明書サムプリントと一致しません。	スーパーバイザー クラスタ に登録された SSL サムプリントが、NSX Manager によって提示された証明書の SHA-1 ハッシュと一致しません。	■ NSX と vCenter Server インスタンス間にある NSX Manager で信頼を再度有効にします。 ■ vCenter Server で `wcpsvc` を再起動します。
制御プレーン仮想マシン <*仮想マシン名*> から <*コンポーネント名*> <*コンポーネント アドレス*> に接続できません。エラー: *エラー メッセージ テキスト*	一般的なネットワーク障害が発生しました。	■ NSX Manager の管理ネットワークで、競合する IP アドレス、ファイアウォール ルールなどの外部ネットワーク設定を検証します。 ■ NSX 拡張機能の NSX Manager の IP アドレスが正しいことを確認します。 ■ NSX Manager が実行されていることを確認します。

表 5-3. ロード バランサ エラー

制御プレーン仮想マシン <*仮想マシン名*> は、ロード バランサの (<*ロード バランサ*>- <*ロード バランサ エンドポイント*>) 証明書を信頼していません。	ロード バランサが提示する証明書は、制御プレーン仮想マシンに構成されている証明書とは異なります。	ロード バランサに正しい管理 TLS 証明書が構成されていることを確認します。
制御プレーン仮想マシン <*仮想マシン名*> がロード バランサの (<*ロード バランサ*>- <*ロード バランサ エンドポイント*>) 証明書を検証できませんでした。証明書が無効です。	ロード バランサが提示する証明書の形式が無効であるか、有効期限が切れています。	構成されたロード バランサのサーバ証明書を修正します。
制御プレーン仮想マシン <*仮想マシン名*> が、ユーザー名 <*ユーザー名*> と指定のパスワードを使用してロード バランサ (<*ロード バランサ*>- <*ロード バランサ エンドポイント*>) に対して認証できませんでした。	ロード バランサのユーザー名またはパスワードが正しくありません。	ロード バランサに構成されたユーザー名とパスワードが正しいかどうかを確認します。
制御プレーン仮想マシン <*仮想マシン名*> からロード バランサ (<*ロード バランサ*>- <*ロード バランサ エンドポイント*>) に接続する際に HTTP エラーが発生しました。	制御プレーン仮想マシンはロード バランサ エンドポイントに接続できますが、エンドポイントが正常な (200) HTTP 応答を返しません	ロード バランサが健全な状態であり、要求を受け入れていることを確認します。
制御プレーン仮想マシン <*仮想マシン名*> から <*ロード バランサ*> (<*ロード バランサ エンドポイント*>) に接続できません。エラー: <*エラー テキスト*>	■ 一般的なネットワーク障害が発生しました。 ■ 通常は、ロード バランサが機能していないか、ファイアウォールの一部によって接続がブロックされています。	■ ロード バランサ エンドポイントにアクセスできることを検証します ■ ファイアウォールがロード バランサへの接続をブロックしていないことを検証します。

vSphere with Tanzu での HTTP プロキシ設定の構成

スーパーバイザー クラスタ および Tanzu Kubernetes クラスタの HTTP プロキシ設定の構成方法を確認します。スーパーバイザー クラスタ および Tanzu Kubernetes クラスタを Tanzu Mission Control に登録する際に、これらのクラスタに HTTP プロキシを構成するためのワークフローについて説明します。Tanzu Mission Control に管理クラスタとして登録するオンプレミス スーパーバイザー クラスタ のイメージ プルおよびコンテナ トラフィックには、HTTP プロキシを使用します。

Tanzu Mission Control で使用する スーパーバイザー クラスタ および Tanzu Kubernetes クラスタで HTTP プロキシ設定を構成するためのワークフロー

Tanzu Mission Control に管理クラスタとして登録する スーパーバイザー クラスタ で HTTP プロキシを構成するには、次の手順を実行します。

1 vSphere で、vCenter Server から HTTP プロキシ設定を継承するか、名前空間管理クラスタの API または DCLI コマンド ラインを使用して個々の スーパーバイザー クラスタ のプロキシ設定を構成して、スーパーバイザー クラスタ で HTTP プロキシを構成します。

2 Tanzu Mission Control では、vSphere with Tanzu で スーパーバイザー クラスタ に構成したプロキシ設定を使用して、プロキシ構成オブジェクトを作成します。vSphere with Tanzu で実行されている Tanzu Kubernetes Grid サービス クラスタのプロキシ構成オブジェクトの作成を参照してください。

3 Tanzu Mission Control で、スーパーバイザー クラスタ を管理クラスタとして登録する場合は、このプロキシ構成オブジェクトを使用します。Tanzu Mission Control への管理クラスタの登録および vSphere with Tanzu でのスーパーバイザー クラスタの登録の完了を参照してください。

Tanzu Mission Control でワークロード クラスタとしてプロビジョニングまたは追加する Tanzu Kubernetes クラスタに HTTP プロキシを構成するには、次の手順を実行します。

1 Tanzu Kubernetes クラスタで使用するプロキシ設定を使用して、プロキシ構成オブジェクトを作成します。vSphere with Tanzu で実行されている Tanzu Kubernetes Grid サービス クラスタのプロキシ構成オブジェクトの作成を参照してください。

2 Tanzu Kubernetes クラスタをワークロード クラスタとしてプロビジョニングまたは追加する場合は、このプロキシ構成オブジェクトを使用します。vSphere with Tanzu でのクラスタのプロビジョニングおよび Tanzu Mission Control 管理へのワークロード クラスタの追加を参照してください。

vSphere with Tanzu での Tanzu Kubernetes クラスタへの HTTP プロキシの構成

次のいずれかの方法を使用して、vSphere with Tanzu で Tanzu Kubernetes クラスタにプロキシを構成します。

- 個々の Tanzu Kubernetes クラスタにプロキシ設定を構成します。Tanzu Kubernetes Grid サービス v1alpha2 API を使用して Tanzu Kubernetes クラスタをプロビジョニングするための構成パラメータを参照してください。構成 YAML の例については、Tanzu Kubernetes Grid サービス v1alpha2 API を使用してカスタム Tanzu Kubernetes クラスタをプロビジョニングするためのサンプル YAML を参照してください。
- すべての Tanzu Kubernetes クラスタに適用されるグローバル プロキシ構成を作成します。Tanzu Kubernetes Grid サービス v1alpha2 API の構成パラメータを参照してください。

注： Tanzu Mission Control を使用して Tanzu Kubernetes クラスタを管理する場合は、vSphere with Tanzu のクラスタ YAML ファイルを使用してプロキシ設定を構成する必要はありません。Tanzu Kubernetes クラスタをワークロード クラスタとして Tanzu Mission Control に追加するときにプロキシ設定を構成できます。

新規作成された vSphere 7.0 Update 3 スーパーバイザー クラスタ でのプロキシ設定の構成

vSphere 7.0 Update 3 環境で新規作成された スーパーバイザー クラスタ の場合、HTTP プロキシ設定は vCenter Server から継承されます。vCenter Server で HTTP プロキシ設定を構成する前または後に スーパーバイザー クラスタ を作成したかに関係なく、設定はクラスタによって継承されます。

vCenter Server で HTTP プロキシ設定を構成する方法については、DNS、IP アドレス、およびプロキシの設定を参照してください。

また、クラスタ管理 API または DCLI を使用して、個々の スーパーバイザー クラスタ で継承された HTTP プロキシの構成をオーバーライドすることもできます。

vCenter Server プロキシ設定の継承は、新しく作成された vSphere 7.0.3 スーパーバイザー クラスタ のデフォルト構成であるため、スーパーバイザー クラスタ がプロキシを必要とせず、vCenter Server が引き続きプロキシを必要とする場合は、クラスタ管理 API または DCLI を使用して HTTP プロキシ設定を継承しないことも可能です。

vSphere 7.0 Update 3 にアップグレードされた スーパーバイザー クラスタ でのプロキシ設定の構成

スーパーバイザー クラスタ を vSphere 7.0 Update 3 にアップグレードした場合、vCenter Server の HTTP プロキシ設定は自動的に継承されません。この場合は、`vcenter/namespace-management/clusters` API または DCLI コマンド ラインを使用して、スーパーバイザー クラスタ のプロキシ設定を構成します。

クラスタ管理 API を使用した スーパーバイザー クラスタ での HTTP プロキシの構成

`vcenter/namespace-management/clusters` API を使用して スーパーバイザー クラスタ プロキシ設定を構成します。この API には、スーパーバイザー クラスタ でプロキシを構成するオプションが 3 つあります。

API 設定	新規作成された vSphere 7.0.3 スーパーバイザー クラスタ	vSphere 7.0.3 にアップグレードされた スーパーバイザー クラスタ
`VC_INHERITED`	これは新しい スーパーバイザー クラスタ のデフォルト設定です。API を使用して スーパーバイザー クラスタ のプロキシ設定を構成する必要はありません。vCenter Server の管理インターフェイスを使用してプロキシ設定を構成するだけで済みます。	この設定を使用して、HTTP プロキシの構成を vSphere 7.0.3 にアップグレードされた スーパーバイザー クラスタ にプッシュします。
`CLUSTER_CONFIGURED`	この設定を使用して、次のいずれかの場合に、vCenter Server から継承された HTTP プロキシ構成をオーバーライドします。 ■ スーパーバイザー クラスタ は vCenter Server と異なるサブネット上にあり、別のプロキシ サーバが必要になる。 ■ プロキシ サーバはカスタム CA バンドルを使用している。	この設定を使用して、次のいずれかの場合に、vSphere 7.0.3 にアップグレードされた個々の スーパーバイザー クラスタ に HTTP プロキシを構成します。 ■ スーパーバイザー クラスタ は vCenter Server と異なるサブネット上にあり、別のプロキシ サーバが必要になるため、vCenter Server プロキシは使用できない。 ■ プロキシ サーバはカスタム CA バンドルを使用している。
`NONE`	スーパーバイザー クラスタ がインターネットに直接接続されている一方で、vCenter Server がプロキシを必要としている場合は、この設定を使用します。NONE の設定を使用すると、vCenter Server のプロキシ設定が スーパーバイザー クラスタ によって継承されなくなります。	

HTTP プロキシを スーパーバイザー クラスタ に設定するか、既存の設定を変更するには、vCenter Server との SSH セッションで次のコマンドを使用します。

```
vc_address=<IP address>
cluster_id=domain-c<number>
session_id=$(curl -ksX POST --user '<SSO user name>:<password>' https://$vc_address/api/
session | xargs -t)
curl -k -X PATCH -H "vmware-api-session-id: $session_id" -H "Content-Type: application/json"
```

```
-d '{ "cluster_proxy_config": { "proxy_settings_source": "CLUSTER_CONFIGURED",
"http_proxy_config":"<proxy_url>" } }' https://$vc_address/api/vcenter/namespace-management/
clusters/$cluster_id
```

クラスタ ID 全体の中で渡す必要があるのは domain_c<number> だけです。たとえば、クラスタ ID が ClusterComputeResource:domain-c50:5bbb510f-759f-4e43-96bd-97fd703b4edb の場合は、そこから domain-c50 を取得します。

VC_INHERITED または NONE の設定を使用する場合は、コマンド内で "http_proxy_config:<proxy_url>" を省略します。

カスタム CA バンドルを使用するには、TSL CA 証明書をプレーン テキストで指定して、コマンドに "tlsRootCaBundle": "<TLS_certificate>" を追加します。

DCLI を使用した スーパーバイザー クラスタ での HTTP プロキシ設定の構成

次の DCLI コマンドを使用すると、CLUSTER_CONFIGURED 設定を使用して スーパーバイザー クラスタ に HTTP プロキシ設定を構成することができます。

```
<dcli> namespacemanagement clusters update --cluster domain-c57 --cluster-proxy-config-http-
proxy-config <proxy URL> --cluster-proxy-config-https-proxy-config <proxy URL> --cluster-
proxy-config-proxy-settings-source CLUSTER_CONFIGURED
```

リモート rsyslog に対する スーパーバイザー クラスタ 制御プレーンのログ ストリーミング

重要なログ データの損失を防ぐため、スーパーバイザー クラスタ 制御プレーン仮想マシンからリモート rsyslog レシーバに対するログ ストリーミングの構成方法をチェックします。

スーパーバイザー クラスタ 制御プレーン仮想マシンのコンポーネントで生成されたログは、仮想マシンのファイル システムにローカルに保存されます。大量のログが溜まると、ログが急速にローテーションされ、さまざまな問題の根本原因の特定に役立つ貴重なメッセージが失われます。vCenter Server と スーパーバイザー クラスタ 制御プレーン仮想マシンでは、リモート rsyslog レシーバに対するローカル ログのストリーミングをサポートしています。この機能は、次のサービスとコンポーネントのログのキャプチャに便利です。

- vCenter Server：ワークロード制御プレーン サービス、ESX Agent Manager サービス、認証局サービス、vCenter Server で実行しているその他のサービス。
- スーパーバイザー クラスタ 制御プレーン コンポーネントと スーパーバイザー クラスタ の組み込みサービス（仮想マシン サービスおよび Tanzu Kubernetes Grid サービス など）。

ローカル ログ データを収集して、それがリモートの rsyslog レシーバにストリーミングされるように、vCenter Server アプライアンスを構成できます。この構成を vCenter Server に適用すると、vCenter Server 内で実行している rsyslog センダーは、その vCenter Server システム内のサービスで生成されたログの送信を開始します。

スーパーバイザー クラスタ は、vCenter Server と同じメカニズムでローカル ログをオフロードして構成管理のオーバーヘッドを軽減します。ワークロード制御プレーン サービスは、定期的にログをポーリングして、vCenter Server rsyslog 構成を監視します。ワークロード制御プレーン サービスは、リモート vCenter Server rsyslog 構成が空でないことを検出すると、すべての スーパーバイザー クラスタ の各制御プレーン仮想マシンにこの構成を伝達します。その場合、大量の rsyslog メッセージ トラフィックが生成される場合があり、リモート rsyslog レシーバが過剰負荷になるおそれがあります。そのため、レシーバ マシンには、大量の rsyslog メッセージを保存できる十分なストレージ容量が必要です。

vCenter Server から rsyslog 構成を削除すると、vCenter Server からの rsyslog メッセージが停止します。ワークロード制御プレーン サービスは変更を検出すると、それをすべての スーパーバイザー クラスタ の各制御プレーン仮想マシンに伝達し、最終的に、制御プレーン仮想マシンのストリームも停止します。

構成手順

スーパーバイザー クラスタ 制御プレーン仮想マシンの rsyslog ストリーミングは、次の手順に従って構成してください。

1 次のようなマシンをプロビジョニングして、rsyslog レシーバを構成します。

- rsyslog サービスをレシーバ モードで実行する。rsyslog ドキュメントのハイ パフォーマンスで大量のメッセージを受信するの例を参照してください。
- 大量のログ データを保存できる十分なストレージ容量がある。
- vCenter Server と スーパーバイザー クラスタ 制御プレーン仮想マシンからデータを受信するためのネットワーク接続がある。

2 https://<vcenter server address>:5480 の vCenter Server アプライアンス管理インターフェイスに root としてログインします。

3 vCenter Server アプライアンス管理インターフェイスから rsyslog レシーバにストリーミングするように vCenter Server を構成します。vCenter Server のログ ファイルをリモート Syslog サーバへ転送を参照してください。

vCenter Server の rsyslog 構成が スーパーバイザー クラスタ 制御プレーン仮想マシンに適用されるまでに数分かかる場合があります。vCenter Server アプライアンス上のワークロード制御プレーン サービスは、5 分ごとにアプライアンス構成をポーリングして、使用可能なすべての スーパーバイザー クラスタ にそれを伝達します。伝達の完了に必要な時間は、環境内の スーパーバイザー クラスタ の数によって異なります。スーパーバイザー クラスタ 上の一部の制御プレーン仮想マシンが健全でない場合、または他の操作を実行している場合、ワークロード制御プレーン サービスは rsyslog 構成が適用されるまでこの動作を繰り返します。

制御プレーン仮想マシン コンポーネントのログの検査

スーパーバイザー クラスタ 制御プレーン仮想マシンの rsyslog は、ログ メッセージのソース コンポーネントを示すタグをログ メッセージに組み込みます。

ログ タグ	説明
vns-control-plane-pods <pod_name>/<instance_number>.log	制御プレーン仮想マシンの Kubernetes ポッドから送信されたログ。 例： vns-control-plane-pods etcd/0.log または vns-control-plane-pods nsx-ncp/573.log
vns-control-plane-imc	制御プレーン仮想マシンから得られた初期構成ログ。
vns-control-plane-boostrap	Kubernetes ノードの制御プレーンのデプロイから得られたブートストラップ ログ。
vns-control-plane-upgrade-logs	制御プレーン ノードのパッチとマイナー バージョンのアップグレードから得られたログ。
vns-control-plane-svchost-logs	制御プレーン仮想マシンのシステム レベルのサービス ホストまたはエージェントのログ。
vns-control-plane-update-controller	制御プレーンの目的の状態のシンクロナイザとリアライザのログ。
vns-control-plane-compact-etcd-logs	制御プレーンの etcd サービス ストレージ圧縮を維持するためのログ。

vSphere with Tanzu でのコンテンツ ライブラリの作成と管理

6

スタンドアローン仮想マシンや Tanzu Kubernetes クラスタなどの vSphere with Tanzu オブジェクトは、コンテンツ ライブラリをテンプレート、イメージ、ディストリビューション、およびデプロイに関連するその他のファイルの統合リポジトリとして使用します。

この章には、次のトピックが含まれています。

- Tanzu Kubernetes リリース のコンテンツ ライブラリの作成と管理
- vSphere with Tanzu におけるスタンドアローン仮想マシン向けのコンテンツ ライブラリの作成と管理

Tanzu Kubernetes リリース のコンテンツ ライブラリの作成と管理

VMware Tanzu は、Kubernetes ソフトウェアのバージョンを Tanzu Kubernetes リリース として配布します。これらのリリースを使用するには、vSphere コンテンツ ライブラリを構成し、使用可能なリリースを同期します。この操作を行うには、サブスクリプションベース モデルを使用するか、オンデマンドで実行します。インターネットが制限された環境で Tanzu Kubernetes をプロビジョニングする場合は、ローカル ライブラリを作成し、リリースを手動でインポートすることができます。

Tanzu Kubernetes リリース の配布について

Tanzu Kubernetes リリース は、Tanzu Kubernetes クラスタで使用するために VMware が署名した、サポート対象の Kubernetes ソフトウェアを配布します。Tanzu Kubernetes リリース を取得して管理するには、vSphere コンテンツ ライブラリを使用します。

各 Tanzu Kubernetes リリース は OVA テンプレートとして配布されます。Tanzu Kubernetes Grid サービス は OVA テンプレートを使用して、Tanzu Kubernetes クラスタ内の仮想マシン ノードを構築します。

Tanzu Kubernetes リリースのリストおよび スーパーバイザー クラスタ との互換性については、Tanzu Kubernetes リリースのリリース ノートを参照してください。

Tanzu Kubernetes リリース は、Photon OS と Ubuntu でサポートされています。OVA テンプレートから構築された仮想マシンのディスク サイズは固定されています。Tanzu Kubernetes クラスタをプロビジョニングするときに、CPU リソースと RAM リソースを指定します。Tanzu Kubernetes クラスタの仮想マシンのクラスを参照してください。

Tanzu Kubernetes Grid サービス は、vSphere コンテンツ ライブラリから Tanzu Kubernetes リリース OVA テンプレートを取得します。サブスクライブされたコンテンツ ライブラリを使用すると、プロセスを自動化したり、インターネットが制限された環境でローカル コンテンツ ライブラリを使用したりできます。Tanzu Kubernetes リリース のサブスクライブ済みコンテンツ ライブラリの作成、セキュリティ保護、同期と Tanzu Kubernetes リリース 用のローカル コンテンツ ライブラリの作成、セキュリティ保護、同期を参照してください。

新しい Tanzu Kubernetes リリース が配布されるにつれて、コンテンツ ライブラリのサイズは大きくなる可能性があります。基盤となるストレージの容量が不足している場合は、新しいコンテンツ ライブラリに移行できます。Tanzu Kubernetes クラスタの新しいコンテンツ ライブラリへの移行を参照してください。

コンテンツ ライブラリを作成して同期したら、Tanzu Kubernetes クラスタをプロビジョニングする各 vSphere 名前空間 を構成します。構成する際に、コンテンツ ライブラリおよび仮想マシン クラスと vSphere 名前空間 を関連付けます。この時点で、vSphere 名前空間 にログインし、Tanzu Kubernetes リリース が使用可能であることを確認できます。Tanzu Kubernetes リリース 用の vSphere 名前空間 の構成を参照してください。

Tanzu Kubernetes リリース のサブスクライブ済みコンテンツ ライブラリの作成、セキュリティ保護、同期

Tanzu Kubernetes クラスタで使用するために Tanzu Kubernetes リリース を保存するには、vSphere with Tanzu が有効な vCenter Server 上にサブスクライブ済みコンテンツ ライブラリを作成します。

サブスクライブ済みコンテンツ ライブラリは、公開コンテンツ ライブラリを基にしています。サブスクリプションが作成されると、システムによってサブスクリプションは公開ライブラリと同期されます。同期モードは、即時に、またはオンデマンドで選択できます。詳細については、サブスクライブ済みライブラリの管理を参照してください。

前提条件

Tanzu Kubernetes リリース の配布について を確認します。

コンテンツ ライブラリを作成するには、vSphere に関する次の権限が必要です。

- ライブラリを作成する vCenter Server インスタンス上の コンテンツ ライブラリ.ローカル ライブラリの作成 または コンテンツ ライブラリ.購読済みライブラリの作成。
- データストア.容量の割り当て（展開先のデータストアが対象）。

手順

1. vSphere Client を使用して、vCenter Server にログインします。
2. [メニュー] - [コンテンツ ライブラリ] の順に選択します。
3. [作成] をクリックします。

   [新しいコンテンツ ライブラリ] ウィザードが開きます。

4 コンテンツ ライブラリの [名前と場所] を指定し、完了したら [次へ] をクリックします。

フィールド	説明
名前	`TanzuKubernetesRelease-subscriber` などのわかりやすい名前を入力します。
メモ	`Tanzu Kubernetes リリース用のオンデマンド サブスクリプション ライブラリ`などの説明を入力します。
vCenter Server	vSphere with Tanzu が有効になっている vCenter Server インスタンスを選択します。

5 [コンテンツ ライブラリの設定] 画面でコンテンツ ライブラリのサブスクリプションを構成し、完了したら [次へ] をクリックします。

a [サブスクライブ済みコンテンツ ライブラリ] オプションを選択します。

注： ローカル コンテンツ ライブラリを使用するには、Tanzu Kubernetes リリース 用のローカル コンテンツ ライブラリの作成、セキュリティ保護、同期を参照してください。

b 発行者の [サブスクリプション URL] アドレスを入力します。

https://wp-content.vmware.com/v2/latest/lib.json

c [コンテンツのダウンロード] オプションで、次のいずれかを選択します。

オプション	説明
ただちに	サブスクリプション プロセスによって、ライブラリ メタデータとイメージの両方が同期されます。公開ライブラリからアイテムが削除されても、そのコンテンツはサブスクライブ済みライブラリのストレージに残るため、手動で削除する必要があります。
必要に応じて	サブスクリプション プロセスでは、ライブラリ メタデータのみが同期されます。イメージが公開されると、Tanzu Kubernetes Grid サービスによってダウンロードされます。アイテムが不要になったら、アイテムのコンテンツを削除してストレージの容量を解放できます。ストレージを保存するには、このオプションを推奨します。

6 プロンプトが表示されたら、SSL 証明書のサムプリントを受け入れます。

SSL 証明書のサムプリントは、インベントリから購読済みコンテンツ ライブラリを削除するまで、システムに格納されます。

7 [セキュリティ ポリシーの適用] 画面で OVF セキュリティ ポリシーを構成し、完了したら [次へ] をクリックします。

a [セキュリティ ポリシーの適用] を選択します。

b [OVF デフォルト ポリシー] を選択します。

このオプションを選択すると、同期プロセス中に OVF 署名証明書が検証されます。証明書の検証に合格しなかった OVF テンプレートは、[検証失敗] タグでマークされます。テンプレートのメタデータは保持されますが、OVF ファイルは同期できません。

注： 現在サポートされているセキュリティ ポリシーは、[OVF デフォルト ポリシー] のみです。

8 [ストレージの追加] 画面で、コンテンツ ライブラリのコンテンツのストレージ場所としてデータストアを選択し、[次へ] をクリックします。

9 [設定の確認] 画面で詳細を確認し、[完了] をクリックします。

10 [コンテンツ ライブラリ] 画面で、作成した新しいコンテンツ ライブラリを選択します。

11 ライブラリ コンテンツの同期を確認または完了します。

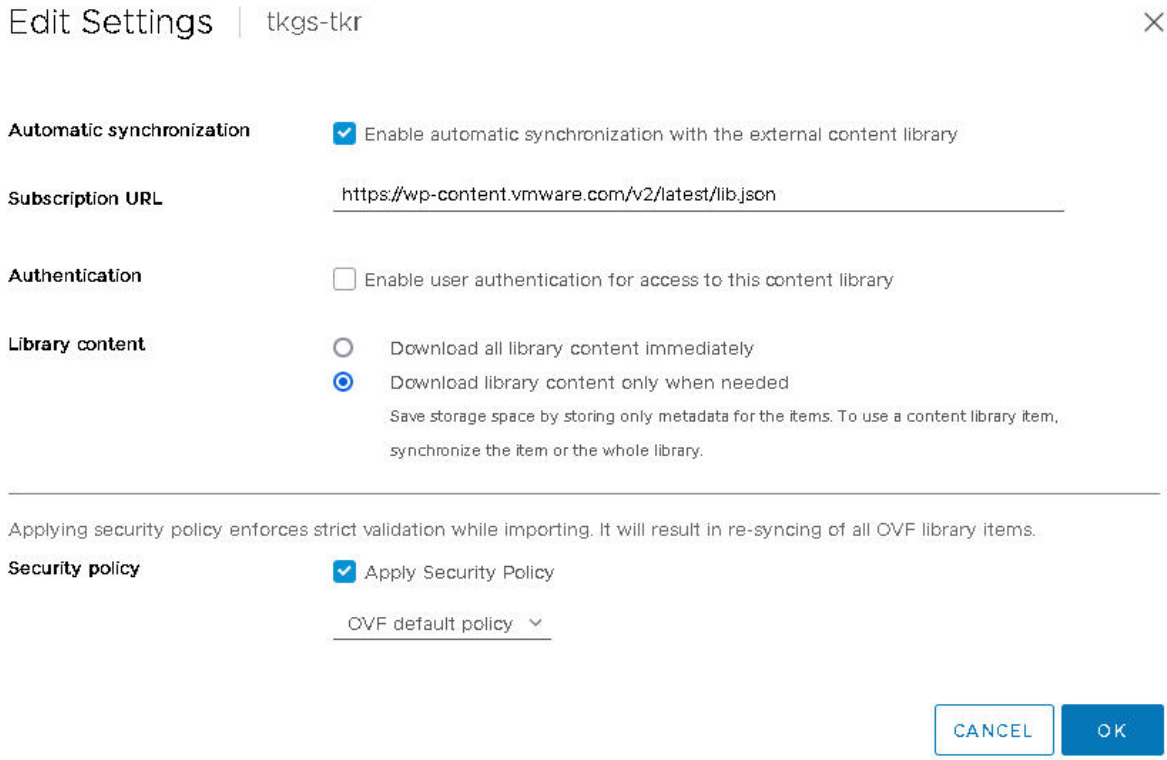
同期オプション	説明
ただちに	すべてのコンテンツをすぐにダウンロードする場合は、ライブラリが同期されていることを確認します。 同期されたライブラリ コンテンツを表示するには、[テンプレート] - [OVF & OVA テンプレート] の順に選択します。
必要に応じて	必要に応じてライブラリを同期する場合は、次の 2 つの方法があります。 ■ [アクション] > [同期] を使用してライブラリ全体を同期する ■ アイテムを右クリックし、[同期] を選択して、このアイテムのみを同期します。 同期されたライブラリ コンテンツを表示するには、[テンプレート] - [OVF & OVA テンプレート] の順に選択します。

12 [必要に応じて] オプションを選択した場合は、使用する OVF テンプレートをダウンロードします。

[必要に応じて] オプションを選択した場合、イメージ ファイルはローカルに保存されず、メタデータのみが保存されます。テンプレート ファイルをダウンロードするには、アイテムを選択して右クリックし、[アイテムの同期] を選択します。

13 サブスクライブ済みコンテンツ ライブラリの設定を更新するには、[アクション] - [設定の編集] の順に選択します。

設定	値
サブスクリプション URL	https://wp-content.vmware.com/v2/latest/lib.json
認証	有効になっていません
ライブラリ コンテンツ	必要に応じてダウンロード
セキュリティ ポリシー	OVF デフォルト ポリシー



次のステップ

コンテンツ ライブラリおよび仮想マシン クラスを名前空間に関連付けて、Tanzu Kubernetes クラスタをプロビジョニングする各 vSphere 名前空間 を構成します。Tanzu Kubernetes リリース 用の vSphere 名前空間 の構成を参照してください。

Tanzu Kubernetes リリース 用のローカル コンテンツ ライブラリの作成、セキュリティ保護、同期

インターネットが制限された（「エアギャップ」）環境で Tanzu Kubernetes クラスタをプロビジョニングするには、ローカル コンテンツ ライブラリを作成し、各 Tanzu Kubernetes リリース を手動でインポートします。

ローカル コンテンツ ライブラリを作成するには、ライブラリの構成、OVA ファイルのダウンロード、OVA ファイルのローカル コンテンツ ライブラリへのインポートを行います。

前提条件

Tanzu Kubernetes リリース の配布について を確認します。

サブスクライブ済みコンテンツ ライブラリを作成するには、次の権限が必要です。

- ライブラリを作成する vCenter Server インスタンス上の コンテンツ ライブラリ.ローカル ライブラリの作成 または コンテンツ ライブラリ.購読済みライブラリの作成。
- データストア.容量の割り当て（展開先のデータストアが対象）。

手順

1 vSphere Client を使用して、vCenter Server にログインします。

2 [メニュー] をクリックします。

3 [コンテンツ ライブラリ] をクリックします。

4 [作成] をクリックします。

[新しいコンテンツ ライブラリ] ウィザードが表示されます。

5 コンテンツ ライブラリの [名前と場所] を指定し、完了したら [次へ] をクリックします。

フィールド	説明
名前	`TanzuKubernetesRelease-local` などのわかりやすい名前を入力します。
メモ	`Tanzu Kubernetes リリース用のローカル ライブラリ`などの説明を入力します。
vCenter Server	vSphere with Tanzu が有効になっている vCenter Server インスタンスを選択します。

6 [コンテンツ ライブラリの設定] 画面で、[ローカル コンテンツ ライブラリ] オプションを選択し、[次へ] をクリックします。

以下で説明するように、ローカル コンテンツ ライブラリの場合、使用する OVF テンプレートを手動でインポートします。

注： サブスクライブ済みコンテンツ ライブラリを使用するには、Tanzu Kubernetes リリース のサブスクライブ済みコンテンツ ライブラリの作成、セキュリティ保護、同期を参照してください。

7 [セキュリティ ポリシーの適用] 画面で OVF セキュリティ ポリシーを構成し、完了したら [次へ] をクリックします。

a [セキュリティ ポリシーの適用] を選択します。

b [OVF デフォルト ポリシー] を選択します。

このオプションを選択すると、同期プロセス中に OVF 署名証明書が検証されます。証明書の検証に合格しなかった OVF テンプレートは、[検証失敗] タグでマークされます。テンプレートのメタデータは保持されますが、OVF ファイルは同期できません。

注： 現在サポートされているセキュリティ ポリシーは、[OVF デフォルト ポリシー] のみです。

8 [ストレージの追加] 画面で、コンテンツ ライブラリのコンテンツのストレージ場所としてデータストアを選択し、[次へ] をクリックします。

9 [設定の確認] 画面で詳細を確認し、[完了] をクリックします。

10 [コンテンツ ライブラリ] 画面で、作成した新しいコンテンツ ライブラリを選択します。

11 ローカル コンテンツ ライブラリにインポートする各 Tanzu Kubernetes リリース の OVA ファイルをダウンロードします。

a ブラウザを使用して、次の URL に移動します。

https://wp-content.vmware.com/v2/latest/

b 目的のイメージのディレクトリをクリックします。通常、このディレクトリは、Kubernetes ディストリビューションの最新または直近のバージョンです。

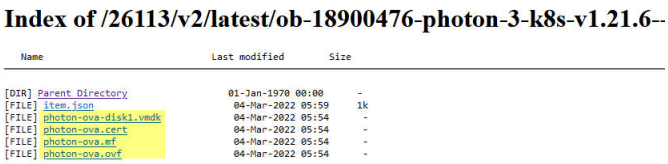
例：

```
ob-18186591-photon-3-k8s-v1.20.7---vmware.1-tkg.1.7fb9067
```

注： ディストリビューション名は、ローカル コンテンツ ライブラリにファイルをインポートするときに必要になります。そのため、ファイルにコピーするか、手順を完了するまでブラウザを開いたままにすることを推奨します。

c 次の各ファイルに対して、右クリックして [リンクに名前を付けて保存] を選択します。

- photon-ova-disk1.vmdk
- photon-ova.cert
- photon-ova.mf
- photon-ova.ovf



d 各ファイルがローカル ファイル システムに正常にダウンロードされていることを確認します。

注： インポート プロセスでソース ディレクトリ内の証明書またはマニフェスト ファイルを利用できない場合、インポートされたライブラリ アイテムは使用できません。つまり、セキュリティ ポリシーを使用して構成されたローカル コンテンツ ライブラリの場合、必要な 4 つのファイルはすべて、ovf と vmdk のインポート元のローカル ディレクトリに配置する必要があります。ovf および vmdk ファイルに加えて、証明書ファイルとマニフェスト ファイルもダウンロードし、4 つのすべてのファイルを同じソース ディレクトリに配置する必要があります。

12 OVA ファイルをローカル コンテンツ ライブラリにインポートします。

a [メニュー] - [コンテンツ ライブラリ] - [ ] の順に選択します。

b [コンテンツ ライブラリ] のリストから、作成したローカル コンテンツ ライブラリの名前のリンクをクリックします。

c [アクション] をクリックします。

d [アイテムのインポート] を選択します。

e [ライブラリ アイテムのインポート] ウィンドウで、[ローカル ファイル] を選択します。

f [ファイルのアップロード] をクリックします。

g `photon-ova.ovf` ファイルと `photon-ova-disk1.vmdk` ファイルの両方を選択します。

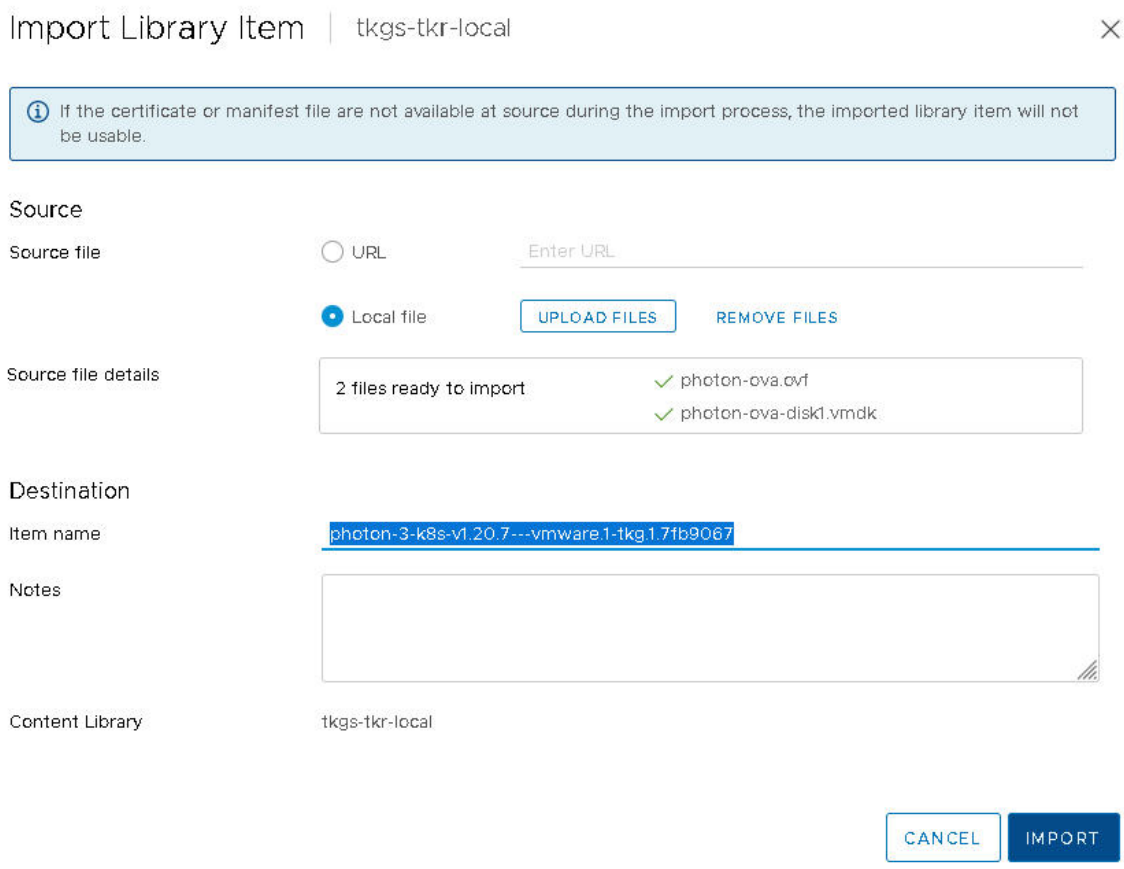
`2 files ready to import` というメッセージが表示されます。各ファイルには、名前の横に緑色のチェック マークが表示されます。

h ファイルをダウンロードしたディレクトリにある Photon イメージのバージョンと Kubernetes バージョンを合わせたものになるように、[ターゲット アイテム] 名を変更します。

例：

```
photon-3-k8s-v1.20.7---vmware.1-tkg.1.7fb9067
```

i [インポート] をクリックします。



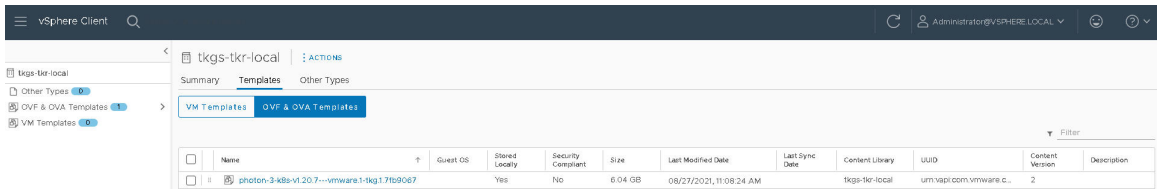
13 ローカル コンテンツ ライブラリに Tanzu Kubernetes リリース がポピュレートされることを確認します。

a 画面の下部にある [最近のタスク] ペインを表示します。

b [ライブラリ アイテムのコンテンツの取得] タスクを監視し、正常に [完了] になっていることを確認します。

c ローカル コンテンツ ライブラリで、[テンプレート] - [OVF & OVA テンプレート] の順に選択します。

d Tanzu Kubernetes リリース メタデータが一覧表示されて、そのコンテンツがローカルに保存されていることを確認します。



次のステップ

コンテンツ ライブラリおよび仮想マシン クラスを名前空間に関連付けて、Tanzu Kubernetes クラスタをプロビジョニングする各 vSphere 名前空間 を構成します。Tanzu Kubernetes リリース 用の vSphere 名前空間 の構成を参照してください。

Tanzu Kubernetes クラスタの新しいコンテンツ ライブラリへの移行

サブスクライブ済みコンテンツ ライブラリが容量に達した場合は、Tanzu Kubernetes クラスタを移行して、追加のストレージ容量を持つ新しいライブラリを使用することができます。

vSphere 管理者がサブスクライブ済みコンテンツ ライブラリを作成する場合、管理者はライブラリ コンテンツを格納するためのデータストア（この場合は OVA ファイル）を指定します。時間の経過とともに配布される Kubernetes バージョンが増え、更新のたびに OVA ファイルが追加されるため、サブスクライブ済みコンテンツ ライブラリのサイズは拡大します。サブスクライブ済みコンテンツ ライブラリには容量が明示的に設定されていませんが、データストアの容量によって制限されています。

サブスクライブ済みコンテンツ ライブラリが容量に達すると、`Internal error occurred: get library items failed for.` というメッセージが表示されることがあります。この場合は、Tanzu Kubernetes クラスタを新しいサブスクライブ済みコンテンツ ライブラリに移行してストレージ容量を増やすことができます。移行は、vSphere 管理者が vSphere Client を使用して実行します。

手順

1 ターゲット クラスタ用に十分な容量がある新しいサブスクライブ済みコンテンツ ライブラリを作成します。Tanzu Kubernetes リリース のサブスクライブ済みコンテンツ ライブラリの作成、セキュリティ保護、同期を参照してください。

2 vSphere Client を使用して、vCenter Server にログインします。

3 [メニュー] - [ホストおよびクラスタ] の順に選択します。

4 Tanzu Kubernetes クラスタを含む スーパーバイザー クラスタ がプロビジョニングされている vSphere クラスタ オブジェクトを選択します。

5 [構成] タブを選択します。

6 ナビゲーション パネルの [名前空間] - [全般] - [] オプションを選択します。

7 メイン パネルの [コンテンツ ライブラリ] セクションの横の [編集] をクリックします。

8 作成した新しいコンテンツ ライブラリを選択し、[OK] をクリックします。

この操作により、クラスタ構成の更新がトリガされます。

注： コンテンツ ライブラリを変更した後、Tanzu Kubernetes クラスタがコンテンツ ソースから変更を取得できるまでに最大 10 分かかることがあります。

ローカル コンテンツ ライブラリへの HAProxy OVA のインポート

Distributed Switch ネットワークを使用している場合は、便宜的な方法として、HAProxy OVA ファイルをコンテンツ ライブラリにインポートできます。エアギャップ環境では、HAProxy OVA をローカル コンテンツ ライブラリにインポートできます。

前提条件

ローカル コンテンツ ライブラリを作成します。Tanzu Kubernetes リリース 用のローカル コンテンツ ライブラリの作成、セキュリティ保護、同期を参照してください。

VMware-HAProxy サイトから最新バージョンの VMware HAProxy OVA ファイルをダウンロードします。

手順

1 vSphere Client を使用して、vCenter Server にログインします。

2 [メニュー] - [コンテンツ ライブラリ] - [ ] の順に選択します。

3 [コンテンツ ライブラリ] のリストから、作成したローカル コンテンツ ライブラリの名前（[HAProxy] など）のリンクをクリックします。

4 [アクション] をクリックします。

5 [アイテムのインポート] を選択します。

6 [ライブラリ アイテムのインポート] ウィンドウで、[ローカル ファイル] を選択します。

7 [ファイルのアップロード] をクリックします。

8 `vmware-haproxy-vX.X.X.ova` ファイルを選択します。

`1 file ready to import` というメッセージが表示されます。ファイルの名前の横に、緑色のチェック マークが表示されます。

9 [インポート] をクリックします。

10 画面の下部にある [最近のタスク] ペインを表示します。

11 [ライブラリ アイテムのコンテンツの取得] タスクを監視し、正常に [完了] になっていることを確認します。

次のステップ

HAProxy 制御プレーン仮想マシンをデプロイします。HAProxy ロード バランサ制御プレーン仮想マシンのデプロイを参照してください。

vSphere with Tanzu におけるスタンドアローン仮想マシン向けのコンテンツ ライブラリの作成と管理

vSphere with Tanzu 環境内のスタンドアローン仮想マシンから、オペレーティング システム、アプリケーション、データなどのソフトウェア構成を含む仮想マシン イメージ（テンプレート）にアクセスできる必要があります。イメージへのアクセスを可能にするには、仮想マシン コンテンツ ライブラリを構成し、仮想マシンがデプロイされている名前空間に関連付ける必要があります。

手順

1 vSphere with Tanzu におけるスタンドアローン仮想マシン向けのコンテンツ ライブラリの作成

vSphere with Tanzu 環境に仮想マシンをデプロイするには、DevOps ユーザーに仮想マシン テンプレートおよびイメージへのアクセス権が必要です。vSphere 管理者として、仮想マシン テンプレートを保存および管理するためのコンテンツ ライブラリを作成します。

2 コンテンツ ライブラリに対する vSphere with Tanzu のスタンドアローン仮想マシンの仮想マシン イメージのポピュレート

vSphere 管理者として、コンテンツ ライブラリに仮想マシン テンプレートを OVA または OVF 形式でポピュレートします。DevOps エンジニアは、このテンプレートを使用して、vSphere with Tanzu 環境に新しいスタンドアローン仮想マシンをプロビジョニングできます。

3 vSphere with Tanzu での仮想マシン コンテンツ ライブラリと名前空間の関連付け

vSphere 管理者は、DevOps エンジニアがテンプレートを使用して vSphere with Tanzu 環境内で新しいスタンドアローン仮想マシンをプロビジョニングできるように、DevOps ユーザーに仮想マシン テンプレートのソースへのアクセス権を付与する必要があります。アクセス権を付与するには、仮想マシン テンプレートを含むコンテンツ ライブラリを名前空間に追加します。

4 vSphere with Tanzu における名前空間での仮想マシン コンテンツ ライブラリの管理

vSphere 管理者は仮想マシン テンプレートを含むコンテンツ ライブラリを名前空間に関連付けて、DevOps エンジニアがテンプレートを使用して vSphere with Tanzu 環境内でスタンドアローン仮想マシンをプロビジョニングできるようにします。コンテンツ ライブラリを名前空間に関連付けた後に、コンテンツ ライブラリを削除して Kubernetes 名前空間から公開解除することができます。コンテンツ ライブラリを追加することもできます。

vSphere with Tanzu におけるスタンドアローン仮想マシン向けのコンテンツ ライブラリの作成

vSphere with Tanzu 環境に仮想マシンをデプロイするには、DevOps ユーザーに仮想マシン テンプレートおよびイメージへのアクセス権が必要です。vSphere 管理者として、仮想マシン テンプレートを保存および管理するためのコンテンツ ライブラリを作成します。

ローカル コンテンツ ライブラリを作成し、テンプレートやその他のタイプのファイルをポピュレートできます。

既存の公開済みローカル ライブラリのコンテンツを自由に使用するために、購読済みライブラリを作成することもできます。

vSphere 7.0 Update 3 以降では、OVF セキュリティ ポリシーを適用することでコンテンツ ライブラリのアイテムを保護できます。OVF セキュリティ ポリシーは、コンテンツ ライブラリのデプロイまたは更新、コンテンツ ライブラリへのアイテムのインポート、またはテンプレートの同期を行うときに、厳密な検証を実施します。テンプレートが信頼されている証明書によって署名されていることを確認するには、信頼された CA からコンテンツ ライブラリへの OVF 署名証明書を追加します。

vSphere のコンテンツ ライブラリおよび仮想マシン テンプレートの詳細については、「コンテンツ ライブラリの使用」を参照してください。

前提条件

必要な権限：

- ライブラリを作成する vCenter Server インスタンス上の コンテンツ ライブラリ.ローカル ライブラリの作成 または コンテンツ ライブラリ.購読済みライブラリの作成。
- データストア.容量の割り当て（展開先のデータストアが対象）。

手順

1 [VM サービス] 画面に移動します。

   a vSphere Client ホーム メニューから、[ワークロード管理] を選択します。

   b [サービス] タブをクリックし、[VM サービス] ペインで [管理] をクリックします。

2 [VM サービス] 画面で [コンテンツ ライブラリ] - [コンテンツ ライブラリの作成] をクリックします。

   このアクションを実行すると、vSphere Client のコンテンツ ライブラリのセクションに移動します。

3 [作成] をクリックします。

   [新しいコンテンツ ライブラリ] ウィザードが開きます。

4 [名前と場所] 画面で、名前を入力し、コンテンツ ライブラリの vCenter Server インスタンスを選択し、[次へ] をクリックします。

   DevOps チームがライブラリのアイテムを簡単に検索してアクセスできるように、コンテンツ ライブラリにはわかりやすい名前を使用してください。

5 [コンテンツ ライブラリの設定] 画面で、作成するコンテンツ ライブラリのタイプを選択し、[次へ] をクリックします。

オプション	説明
ローカル コンテンツ ライブラリ	デフォルトで、ローカル コンテンツ ライブラリには、これを作成した vCenter Server インスタンスからのみアクセスできます。 a （オプション）ライブラリのコンテンツを他の vCenter Server インスタンスで使用できるようにするには、[公開の有効化] を選択します。 b （オプション） コンテンツ ライブラリにアクセスする際にパスワードを要求する場合は、[認証の有効化] を選択してパスワードを設定します。
サブスクライブ済みコンテンツ ライブラリ	サブスクライブ済みコンテンツ ライブラリは、公開コンテンツ ライブラリを基にしています。既存のコンテンツ ライブラリを利用するには、このオプションを使用します。 購読済みライブラリを公開ライブラリと同期して最新のコンテンツを表示することはできますが、購読済みライブラリのコンテンツの追加や削除はできません。公開ライブラリのコンテンツを追加、修正、削除できるのは、公開ライブラリの管理者だけです。 ライブラリを購読するには、次の情報を入力します。 a [サブスクリプション URL] テキスト ボックスで、公開ライブラリの URL アドレスを入力します。 b 公開ライブラリで認証が有効である場合は、[認証の有効化] を選択して、発行元のパスワードを入力します。 c 購読済みライブラリのコンテンツについて、ダウンロード方法を選択します。 ■ 購読した直後に公開ライブラリ内のすべてのアイテムのローカル コピーをダウンロードする場合は、[ただちに] を選択します。 ■ ストレージ容量を節約する場合は、[必要な場合] を選択します。公開ライブラリ内のアイテムのメタデータのみがダウンロードされます。 アイテムを使用する必要がある場合は、アイテムまたはライブラリ全体を同期して、そのコンテンツをダウンロードします。 d プロンプトが表示されたら、SSL 証明書のサムプリントを受け入れます。 SSL 証明書のサムプリントは、インベントリから購読済みコンテンツ ライブラリを削除するまで、システムに格納されます。

6 （オプション） [セキュリティ ポリシーの適用] ページで、[セキュリティ ポリシーの適用]、[OVF デフォルト ポリシー] の順に選択します。

購読済みライブラリでは、このオプションはライブラリでセキュリティ ポリシーがサポートされている場合にのみ表示されます。

このオプションを選択すると、ローカル ホストからライブラリに OVF アイテムをインポートするとき、またはアイテムを同期するときに、システムによって厳密な OVF 証明書検証が実行されます。証明書の検証に合格しなかった OVF アイテムはインポートできません。

同期の際に検証に合格しなかったアイテムは、[検証失敗] タグでマークされます。アイテムとメタデータのみが保持され、アイテム内のファイルは保持されません。

7 [ストレージの追加] 画面で、コンテンツ ライブラリのコンテンツのストレージ場所としてデータストアを選択し、[次へ] をクリックします。

8 [設定の確認] 画面で詳細を確認し、[完了] をクリックします。

次のステップ

コンテンツ ライブラリを作成したら、ライブラリに仮想マシン テンプレートをポピュレートし、DevOps エンジニアがテンプレートを使用して新しい仮想マシンをプロビジョニングできるようにします。コンテンツ ライブラリに対する vSphere with Tanzu のスタンドアローン仮想マシンの仮想マシン イメージのポピュレートを参照してください。

コンテンツ ライブラリに対する vSphere with Tanzu のスタンドアローン仮想マシンの仮想マシン イメージのポピュレート

vSphere 管理者として、コンテンツ ライブラリに仮想マシン テンプレートを OVA または OVF 形式でポピュレートします。DevOps エンジニアは、このテンプレートを使用して、vSphere with Tanzu 環境に新しいスタンドアローン仮想マシンをプロビジョニングできます。

コンテンツ ライブラリを作成した後に、いくつかの方法でアイテムをポピュレートできます。このトピックには、ローカル マシンまたは Web サーバからファイルをインポートして、ローカル コンテンツ ライブラリにアイテムを追加する方法が示されています。コンテンツ ライブラリにデータをポピュレートする他の方法については、ライブラリへのコンテンツのポピュレートを参照してください。

前提条件

- 仮想マシンをプロビジョニングするためのコンテンツ ライブラリを作成します。vSphere with Tanzu におけるスタンドアローン仮想マシン向けのコンテンツ ライブラリの作成。
- VMware Cloud Marketplace に OVF として表示される、互換性のある仮想マシン イメージのみを使用します。互換性のあるイメージを見つけるには、 VMware Cloud Marketplace Web サイトで **仮想マシン サービス イメージ**を検索します。CentOS の仮想マシン サービス イメージの例については、「CentOS の仮想マシン サービス イメージ」を参照してください。
- ライブラリがセキュリティ ポリシーによって保護されている場合は、すべてのライブラリ アイテムが準拠していることを確認します。保護されたライブラリ内に準拠しているアイテムと準拠していないアイテムが混在している場合、`kubectl get virtualmachineimages` は DevOps エンジニアに仮想マシン イメージを表示できません。
- 必要な権限：コンテンツ ライブラリ.ライブラリ アイテムの追加 および コンテンツ ライブラリ.ファイルの更新（ライブラリが対象）

手順

1. vSphere Client ホーム メニューから、[コンテンツ ライブラリ] を選択します。
2. ローカル コンテンツ ライブラリを右クリックし、[アイテムのインポート] を選択します。

   [ライブラリ アイテムのインポート] ダイアログ ボックスが開きます。

3 [ソース] セクションで、アイテムのソースを選択します。

オプション	説明
URL	アイテムが格納されている Web サーバへのパスを入力します。 **注：** .ovf ファイルまたは .ova ファイルをインポートできます。結果のコンテンツ ライブラリ アイテムは OVF テンプレート タイプになります。
ローカル ファイル	[ファイルのアップロード] をクリックして、ローカル システムからインポートするファイルに移動します。ドロップダウン メニューでは、ローカル システムのファイルをフィルタリングできます。 **注：** .ovf ファイルまたは .ova ファイルをインポートできます。OVF テンプレートをインポートする場合は、最初に OVF 記述子ファイル (.ovf) を選択します。次に、.vmdk ファイルなど、OVF テンプレートの他のファイルの選択を求められます。結果のコンテンツ ライブラリ アイテムは OVF テンプレート タイプになります。

vCenter Server は、インポート時に、OVF パッケージ内のマニフェストと証明書ファイルを読み取り、検証します。証明書の問題（vCenter Server が期限切れの証明書を検出した場合など）があると、[ライブラリ アイテムのインポート] ウィザードに警告が表示されます。

注： OVF パッケージがローカル マシンの .ovf ファイルからインポートされた場合、vCenter Server は署名済みコンテンツを読み取りません。

4 [ターゲット] セクションで、アイテムの名前と説明を入力します。

5 [インポート] をクリックします。

結果

アイテムが [テンプレート] タブまたは [その他のアイテム] タブに表示されます。

次のステップ

コンテンツ ライブラリを作成して仮想マシン テンプレートをポピュレートしたら、名前空間にライブラリを追加して、DevOps ユーザーにコンテンツ ライブラリへのアクセス権を付与します。vSphere with Tanzu での仮想マシン コンテンツ ライブラリと名前空間の関連付けを参照してください。

vSphere with Tanzu での仮想マシン コンテンツ ライブラリと名前空間の関連付け

vSphere 管理者は、DevOps エンジニアがテンプレートを使用して vSphere with Tanzu 環境内で新しいスタンドアローン仮想マシンをプロビジョニングできるように、DevOps ユーザーに仮想マシン テンプレートのソースへのアクセス権を付与する必要があります。アクセス権を付与するには、仮想マシン テンプレートを含むコンテンツ ライブラリを名前空間に追加します。

1 つの名前空間に複数のコンテンツ ライブラリを追加できます。また、同じコンテンツ ライブラリを複数の名前空間に追加できます。

注： この手順は、仮想マシン サービスのコンテンツ ライブラリにのみ適用されます。Tanzu Kubernetes Grid サービス コンテンツ ライブラリは、Tanzu Kubernetes Grid サービス ペインから管理する必要があります。

前提条件

- コンテンツ ライブラリを作成します。vSphere with Tanzu におけるスタンドアローン仮想マシン向けのコンテンツ ライブラリの作成を参照してください。
- ライブラリに仮想マシン テンプレートをポピュレートします。コンテンツ ライブラリに対する vSphere with Tanzu のスタンドアローン仮想マシンの仮想マシン イメージのポピュレートを参照してください。
- 必要な権限：
  - 名前空間.クラスタ全体の構成の変更
  - 名前空間.名前空間構成の変更

手順

1 vSphere Client で、名前空間に移動します。

   a vSphere Client ホーム メニューから、[ワークロード管理] を選択します。

   b [名前空間] タブをクリックして、名前空間をクリックします。

2 コンテンツ ライブラリを追加します。

   a [VM サービス] ペインで [コンテンツ ライブラリの追加] をクリックします。

   b 1 つ以上のコンテンツ ライブラリを選択して、[OK] をクリックします。

結果

追加したライブラリのコンテンツは Kubernetes 名前空間で仮想マシン イメージとして使用できるようになり、DevOps は仮想マシンをセルフサービスする際に、このコンテンツを使用することができます。vSphere with Tanzu への仮想マシンのデプロイを参照してください。

次のステップ

コンテンツ ライブラリを名前空間に関連付けた後に、コンテンツ ライブラリを追加するか、ライブラリを削除して Kubernetes 名前空間から公開解除することができます。vSphere with Tanzu における名前空間での仮想マシン コンテンツ ライブラリの管理を参照してください。

vSphere with Tanzu における名前空間での仮想マシン コンテンツ ライブラリの管理

vSphere 管理者は仮想マシン テンプレートを含むコンテンツ ライブラリを名前空間に関連付けて、DevOps エンジニアがテンプレートを使用して vSphere with Tanzu 環境内でスタンドアローン仮想マシンをプロビジョニングできるようにします。コンテンツ ライブラリを名前空間に関連付けた後に、コンテンツ ライブラリを削除して Kubernetes 名前空間から公開解除することができます。コンテンツ ライブラリを追加することもできます。

名前空間からコンテンツ ライブラリを削除しても、ライブラリ イメージを使用して以前にデプロイされた仮想マシンは影響を受けません。

注： この手順は、仮想マシン サービスのコンテンツ ライブラリにのみ適用されます。Tanzu Kubernetes Grid サービス コンテンツ ライブラリは、Tanzu Kubernetes Grid サービス ペインから管理する必要があります。

前提条件

- 名前空間にコンテンツ ライブラリを追加します。vSphere with Tanzu での仮想マシン コンテンツ ライブラリと名前空間の関連付け。
- 必要な権限：
  - 名前空間.クラスタ全体の構成の変更
  - 名前空間.名前空間構成の変更

手順

1 vSphere Client で、名前空間に移動します。

   a vSphere Client ホーム メニューから、[ワークロード管理] を選択します。

   b [名前空間] タブをクリックして、名前空間をクリックします。

2 コンテンツ ライブラリを追加または削除します。

   a [VM サービス] ペインで [コンテンツ ライブラリの管理] をクリックします。

   b 次のいずれかの操作を実行します。

オプション	説明
コンテンツ ライブラリの削除	コンテンツ ライブラリを選択解除し、[OK] をクリックします。
コンテンツ ライブラリの追加	1 つ以上のコンテンツ ライブラリを選択して、[OK] をクリックします。

次のステップ

ライブラリのコンテンツは Kubernetes 名前空間で仮想マシン イメージとして使用できるようになり、DevOps は仮想マシンをセルフサービスする際に、このコンテンツを使用することができます。vSphere with Tanzu への仮想マシンのデプロイを参照してください。

vSphere 名前空間の構成と管理 7

vSphere ポッド、仮想マシン、Tanzu Kubernetes クラスタなどの vSphere with Tanzu ワークロードは、vSphere 名前空間 にデプロイされます。スーパーバイザー クラスタ に vSphere 名前空間 を定義して、リソース割り当ておよびユーザー権限を使用して構成します。DevOps に関するニーズと実行予定のワークロードに応じて、ストレージ ポリシー、仮想マシン クラス、およびコンテンツ ライブラリを割り当てて、最新の Tanzu Kubernetes リリース および仮想マシン イメージを取得することもできます。

この章には、次のトピックが含まれています。

vSphere 名前空間 の作成と設定

vSphere 管理者は、スーパーバイザー クラスタ に vSphere 名前空間 を作成します。この名前空間にリソース制限を設定し、DevOps エンジニアがアクセスできるように権限を設定します。DevOps エンジニアに Kubernetes 制御プレーンの URL を提供し、権限が付与されている名前空間で Kubernetes ワークロードを実行できるようにします。

vSphere ネットワーク スタックが構成されている スーパーバイザー クラスタ の名前空間と、NSX-T Data Center が構成されているクラスタの名前空間では、ネットワークの構成と機能が異なります。

NSX-T Data Center で構成された スーパーバイザー クラスタ に作成された名前空間は、ワークロード管理プラットフォームのすべての機能をサポートします。サポートされるのは、vSphere ポッド、仮想マシン、および Tanzu Kubernetes クラスタです。これらの名前空間に対するワークロード ネットワークのサポートは、NSX-T Data Center によって提供されます。詳細については、『NSX-T Data Center を使用して vSphere with Tanzu をセットアップするためのシステム要件』を参照してください。

vSphere ネットワーク スタックが構成された スーパーバイザー クラスタ に作成された名前空間は、Tanzu Kubernetes クラスタと仮想マシンのみをサポートし、vSphere ポッド をサポートしないため、ユーザーはこれらと Harbor レジストリ を併用できません。これらの名前空間に対するワークロード ネットワークのサポートは、スーパーバイザー クラスタ のホスト部分に接続されている vSphere Distributed Switch によって提供されます。詳細については、『vSphere ネットワークおよび HAProxy ロード バランサを使用して vSphere with Tanzu をセットアップするためのシステム要件』を参照してください。

また、名前空間へのリソース制限の設定、権限の割り当てや、名前空間サービスをテンプレートとしてクラスタ上でプロビジョニングしたり、有効にしたりできます。したがって、DevOps エンジニアはセルフ サービス方式でスーパーバイザー名前空間を作成し、その中にワークロードをデプロイすることができます。詳細については、『セルフサービス名前空間テンプレートのプロビジョニング』を参照してください。

前提条件

- vSphere with Tanzu でクラスタを設定します。
- 名前空間にアクセスするすべての DevOps エンジニアのユーザーまたはグループを作成します。
- パーシステント ストレージのストレージ ポリシーを作成します。ストレージ ポリシーでは、ゴールド、シルバー、ブロンズなど、さまざまなタイプやクラスのストレージを定義できます。
- スタンドアローン仮想マシンの仮想マシン クラスおよびコンテンツ ライブラリを作成します。
- Tanzu Kubernetes クラスタで使用する Tanzu Kubernetes リリース のコンテンツ ライブラリを作成します。Tanzu Kubernetes リリース のコンテンツ ライブラリの作成と管理を参照してください。
- 必要な権限：
  - 名前空間.クラスタ全体の構成の変更
  - 名前空間.名前空間構成の変更

手順

1. vSphere Client ホーム メニューから、[ワークロード管理] を選択します。
2. [名前空間] タブを選択します。
3. [名前空間の作成] をクリックします。
4. 名前空間を置く スーパーバイザー クラスタ を選択します。
5. 名前空間の名前を入力します。

   名前は DNS 準拠の形式にする必要があります。
6. [ネットワーク] ドロップダウン メニューから、名前空間のワークロード ネットワークを選択します。

注： この手順を使用できるのは、vSphere ネットワーク スタックが構成されたクラスタに名前空間を作成する場合のみです。

7 クラスタの NSX-T Data Center ネットワーク スタックが構成済みの場合は、[クラスタ ネットワーク設定のオーバーライド] を選択してクラスタ ネットワーク設定をオーバーライドし、名前空間のネットワーク設定を構成できます。

名前空間の次のネットワーク設定を構成します。

オプション	説明
NAT モード	NAT モードは、デフォルトで選択されています。 このオプションを選択解除すると、vSphere ポッド、仮想マシン、Tanzu Kubernetes クラスタ ノードの IP アドレスなどのワークロードがいずれも Tier-0 ゲートウェイの外から直接アクセスできるようになります。出力方向の CIDR を構成する必要はありません。 **注：** 名前空間モードは、一度有効にすると変更できません。
Tier-0 ゲートウェイ	名前空間の Tier-1 ゲートウェイに関連付ける Tier-0 ゲートウェイを選択します。 Tier-0 ゲートウェイを選択すると、クラスタを有効にするときに構成した Tier-0 ゲートウェイがオーバーライドされるため、再度 CIDR 範囲を構成する必要があります。 Tier-0 ゲートウェイにリンクされている VRF ゲートウェイを選択すると、ネットワークとサブネットが自動的に構成されます。 NAT モードを選択済みの場合は、サブネット、入力方向、および出力方向の CIDR を構成する必要があります。 NAT モードを選択解除した場合は、サブネットと入力方向 CIDR のみを構成する必要があります。 **注：** Tier-0 ゲートウェイは、一度選択すると変更できません。
名前空間ネットワーク CIDR	1 つ以上の IP CIDR を入力してサブネット/セグメントを作成し、名前空間に接続されているワークロードに IP アドレスを割り当てます。 **注：** クラスタに CIDR 範囲を構成していない場合は、CIDR 範囲を入力します。名前空間を作成した後、名前空間ネットワーク設定を編集して追加の CIDR を構成できます。
名前空間サブネット プリフィックス	名前空間セグメント用に予約されるサブネットのサイズを指定する、サブネット プリフィックスを入力します。デフォルトは 28 です。 **注：** サブネット プリフィックスは、一度指定すると変更できません。
入力方向 CIDR	vSphere ポッド または Tanzu Kubernetes クラスタのロード バランサ サービスによって公開される仮想 IP アドレスの入力方向 IP アドレス範囲を決定する CIDR 注釈を入力します。 名前空間を作成した後、名前空間ネットワーク設定を編集して追加の CIDR を構成できます。
出力方向 CIDR	SNAT IP アドレスの出力方向 IP アドレス範囲を決定する CIDR 注釈を入力します。 名前空間を作成した後、名前空間ネットワーク設定を編集して追加の CIDR を構成できます。
ロード バランサのサイズ	名前空間の Tier-1 ゲートウェイ上のロード バランサ インスタンスのサイズを選択します。

8 説明を入力し、[作成] をクリックします。

名前空間が スーパーバイザー クラスタ に作成されます。

9 DevOps エンジニアが名前空間にアクセスできるように権限を設定します。

a [権限] ペインで [権限の追加] を選択します。

b ID ソース、ユーザーまたはグループ、およびロールを選択し、[OK] をクリックします。

10 名前空間にパーシステント ストレージを設定します。

名前空間に割り当てるストレージ ポリシーは、パーシステント ボリュームと Tanzu Kubernetes クラスタ ノードを vSphere ストレージ環境のデータストア内に配置する方法を制御します。パーシステント ボリュームに対応するパーシステント ボリュームの要求は、vSphere ポッド、仮想マシン、または Tanzu Kubernetes クラスタから行われることがあります。詳細については、『10 章 vSphere with Tanzu でのパーシステント ストレージの使用』を参照してください。

a [ストレージ] ペインで [ストレージの追加] を選択します。

b パーシステント ボリュームのデータストアの配置を制御するストレージ ポリシーを選択して、[OK] をクリックします。

ストレージ ポリシーを割り当てると、vSphere with Tanzu によって、一致する Kubernetes ストレージ クラスが vSphere 名前空間 に作成されます。VMware Tanzu™ Kubernetes Grid™ サービス を使用する場合、ストレージ クラスは名前空間から Kubernetes クラスタに自動的に複製されます。名前空間に複数のストレージ ポリシーを割り当てると、ストレージ ポリシーごとに個別のストレージ クラスが作成されます。

11 [容量と使用量] ペインで [制限の編集] を選択し、名前空間に対するリソース制限を構成します。

オプション	説明
CPU	名前空間に予約する CPU リソースの量。
メモリ	名前空間に予約するメモリの量。
ストレージ	名前空間に予約するストレージ容量の合計。
ストレージ ポリシーの制限	名前空間に関連付けた各ストレージ ポリシーに専用のストレージ容量を個別に設定します。

名前空間のリソース プールが vCenter Server に作成されます。名前空間で使用可能なストレージの総容量は、ストレージ制限によって決まる一方で、関連付けられたストレージ クラスにおける vSphere ポッド のパーシステント ボリュームの配置は、ストレージ ポリシーによって決まります。

12 スタンドアローン仮想マシン用の仮想マシン サービスをセットアップします。

詳細については、『12 章 vSphere with Tanzu での仮想マシンのデプロイと管理』を参照してください。

13 Tanzu Kubernetes クラスタの名前空間を構成します。次に例を示します。

- Tanzu Kubernetes リリース コンテンツ ライブラリに名前空間を関連付けます。
- 名前空間にデフォルトの仮想マシン クラスを追加します。

詳細については、『Tanzu Kubernetes リリース 用の vSphere 名前空間 の構成』を参照してください。

次のステップ

vSphere 向け Kubernetes CLI Tools を介して スーパーバイザー クラスタ にログインするためのユーザー名と Kubernetes 制御プレーン URL を DevOps エンジニアと共有します。DevOps エンジニアには、複数の名前空間へのアクセス権を付与できます。9 章 vSphere with Tanzu クラスタへの接続を参照してください。

vSphere ポッド コンテナに対するデフォルトのメモリおよび CPU の予約と制限の設定

vSphere Client を使用して、名前空間のコンテナに対してデフォルトのメモリおよび CPU の予約と制限を設定できます。DevOps エンジニアは、定義するポッド仕様でこれらの値を後でオーバーライドできます。コンテナ要求は vSphere ポッド のリソース予約に変換されます。

前提条件

- スーパーバイザー クラスタ に関する名前空間の構成の変更権限があることを確認します。

手順

1. vSphere Client ホーム メニューから、[ワークロード管理] を選択します。
2. 名前空間を選択し、[設定] を選択して、[リソースの制限] をクリックします。
3. 名前空間のコンテナに対してデフォルトのメモリおよび CPU の予約と制限を設定します。

オプション	説明
CPU 要求	名前空間のコンテナに対してデフォルトの CPU 予約を設定します。
CPU リミット	名前空間のコンテナに対して CPU 使用量のデフォルトの制限を設定します。
メモリ要求	名前空間のコンテナに対してデフォルトのメモリ予約を設定します。
メモリの制限	名前空間のコンテナに対してメモリ使用量のデフォルトの制限を設定します。

vSphere 名前空間 での Kubernetes オブジェクトの制限の構成

vSphere 名前空間 内で実行されているポッドの制限や、さまざまな Kubernetes オブジェクトの制限を構成できます。オブジェクトに構成する制限は、アプリケーションの仕様と、vSphere 名前空間 内のリソースの使用方法によって異なります。

前提条件

- スーパーバイザー クラスタ に関する名前空間の構成の変更権限があることを確認します。

手順

1. vSphere Client ホーム メニューから、[ワークロード管理] を選択します。
2. オブジェクトまたはコンテナの制限を適用する名前空間を選択します。
3. コンテナの制限を設定するには、[リソースの制限] を選択し、[編集] をクリックします。

オプション	説明
CPU 要求	コンテナの CPU 要求の量を設定します。
CPU リミット	コンテナで使用可能な CPU の量を設定します。

オプション	説明
メモリ要求	コンテナのメモリ要求の量を設定します。
メモリ制限	コンテナで使用可能なメモリの量を設定します。

注： Tanzu Kubernetes クラスタがプロビジョニングされる vSphere 名前空間 にリソース制限を設定した場合の影響は、クラスタ ノードに使用される仮想マシン クラスのタイプによって異なります。リソース制限を設定する前に、ベスト エフォート型と保証型の違いを理解しておく必要があります。Tanzu Kubernetes クラスタの仮想マシンのクラスを参照してください。

4 名前空間内に存在できる Kubernetes オブジェクトに関する制限を設定するには、[オブジェクト制限] を選択し、[編集] をクリックします。

オプション	説明
ポッド	名前空間内で実行できる vSphere ポッド の数。
展開	名前空間内で実行できるデプロイの数。
ジョブ	名前空間内で実行できるジョブの数。
DaemonSets	名前空間内で実行できるデーモン セットの数。
ReplicaSets	名前空間内のレプリカセットの数。
ReplicationControllers	名前空間内で実行できるレプリケーション コントローラの数。
StatefulSets	名前空間で実行できる StatefulSet の数。
ConfigMaps	名前空間で実行できる ConfigMap の数。
シークレット	名前空間内で実行できるシークレットの数。
パーシステント ボリュームの要求	名前空間内に存在できるパーシステント ボリュームの要求。
サービス	名前空間内に存在できるサービス。

vSphere 名前空間 でのリソースの監視と管理

名前空間のリソース使用量や、名前空間内に存在するさまざまな Kubernetes オブジェクトの数やその状態など、vSphere 名前空間 のさまざまな側面を監視および管理できます。

前提条件

vSphere 名前空間 の作成と設定。

手順

1 vSphere Client を使用して、vCenter Server にログインします。

2 [メニュー] - [ホストおよびクラスタ] ビューの順に移動します。

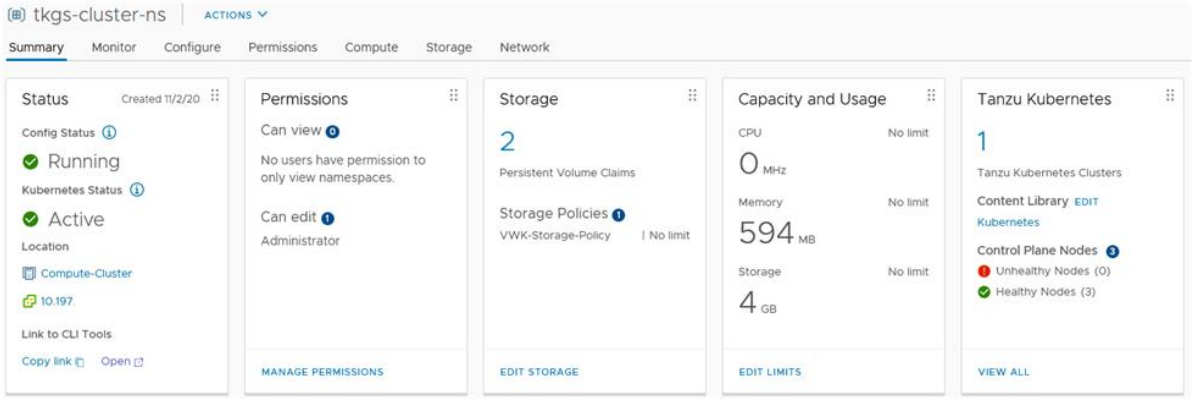
3 [ワークロード管理] が有効な vCenter Server クラスタを選択します。

4 [名前空間] リソース プールを選択して、その内容を展開します。

スーパーバイザー クラスタ 制御プレーン ノードは、名前空間リソース プール内にあります。この スーパーバイザー クラスタ 用に作成された各 vSphere 名前空間 も、[名前空間] リソース プール内にあります。

5 ウィンドウ アイコンとして表示されている vSphere 名前空間 オブジェクトを選択します。

[サマリ] タブには、[ステータス]、[権限]、[ストレージ]、[容量と使用量]、[Tanzu Kubernetes] など、vSphere 名前空間 のさまざまな構成セクションがあります。この画面で、これらの設定を管理できます。



Tanzu Kubernetes リリース 用の vSphere 名前空間 の構成

Tanzu Kubernetes クラスタをプロビジョニングする vSphere 名前空間 は、名前空間を Tanzu Kubernetes リリース のコンテンツ ライブラリおよび使用する仮想マシン クラスに関連付けて構成します。

前提条件

vSphere 名前空間 を作成します。vSphere 名前空間 の作成と設定を参照してください。

Tanzu Kubernetes リリース をホストするコンテンツ ライブラリを作成します。Tanzu Kubernetes リリース のサブスクライブ済みコンテンツ ライブラリの作成、セキュリティ保護、同期または Tanzu Kubernetes リリース用のローカル コンテンツ ライブラリの作成、セキュリティ保護、同期を参照してください。

コンテンツ ライブラリの vSphere 名前空間 への関連付け

Tanzu Kubernetes リリース 用に作成されたコンテンツ ライブラリを vSphere 名前空間 に関連付けるには、vSphere Client を使用して vCenter Server にログインし、次のいずれかの手順を実行します。

vSphere インベントリ パスを使用した関連付け	ワークロード管理パスを使用した関連付け
1 [メニュー] - [ホストおよびクラスタ] の順に選択します。 2 [ワークロード管理] が有効になっている vSphere クラスタを指定します。 3 [構成] タブを選択します。 4 [名前空間] - [全般] の順に選択します。 5 [Tanzu Kubernetes Grid サービス構成] を選択します。 6 [コンテンツ ライブラリ] ラベルの横にある [編集] をクリックします。 7 Tanzu Kubernetes リリース のコンテンツ ライブラリを選択します。 8 [OK] をクリックします。	1 [メニュー] - [ワークロー管理] の順に選択します。 2 [名前空間] タブを選択します。 3 ターゲット vSphere 名前空間 を選択します。 4 [Tanzu Kubernetes Grid サービス] タイルを特定します。 5 [コンテンツ ライブラリ] ラベルの横にある [編集] をクリックします。 6 Tanzu Kubernetes リリース のコンテンツ ライブラリを選択します。 7 [OK] をクリックします。

注： vSphere 名前空間 にコンテンツ ライブラリを関連付けた後、仮想マシン テンプレートを Tanzu Kubernetes クラスタのプロビジョニングに使用できるようになるまでに数分かかることがあります。vSphere 名前空間 の構成の確認を参照してください。

仮想マシン クラスの vSphere 名前空間 への関連付け

vSphere with Tanzu にはデフォルトの仮想マシン クラスがいくつかあり、ユーザーは独自のクラスを作成できます。Tanzu Kubernetes クラスタの仮想マシンのクラスを参照してください。

Tanzu Kubernetes クラスタをプロビジョニングするには、使用する仮想マシン クラスを、Tanzu Kubernetes クラスタをプロビジョニングする各 vSphere 名前空間 に関連付ける必要があります。

デフォルトの仮想マシン クラスを vSphere 名前空間 に関連付けるには、vSphere Client を使用して vCenter Server にログインし、次の手順を実行します。

1 [メニュー] - [ワークロー管理] の順に選択します。

2 [名前空間] タブを選択します。

3 Tanzu Kubernetes クラスタをプロビジョニングするターゲット vSphere 名前空間 を選択します。

4 [VM サービス] タイルを特定します。

5 [VM クラスの追加] リンクをクリックします。

6 追加する仮想マシン クラスを選択します。

   a デフォルトの仮想マシン クラスを追加するには、リストの 1 ページ目の表ヘッダー内のチェックボックスを選択し、2 ページ目に移動して、そのページの表ヘッダーのチェックボックスをオンにします。すべてのクラスが選択されていることを確認します。

   b カスタム クラスを作成するには、[新しい VM クラスの作成] をクリックします。vSphere with Tanzu での仮想マシン クラスの作成を参照してください。

7 [OK] をクリックして操作を完了します。

8　クラスが追加されていることを確認します。[VM サービス] タイルに [VM クラスの管理] が表示されます。

注： [VM サービス] タイルで参照されているコンテンツ ライブラリは、Tanzu Kubernetes リリース ではなく、スタンドアローン仮想マシンで使用するためのものです。vSphere with Tanzu におけるスタンドアローン仮想マシン向けのコンテンツ ライブラリの作成と管理を参照してください。

vSphere 名前空間 の構成の確認

コンテンツ ライブラリおよび仮想マシン クラスを vSphere 名前空間 に関連付けた後、スーパーバイザー クラスタにログインし、同期された各 Tanzu Kubernetes リリース が使用可能であること、および選択した各仮想マシン クラスが使用可能であることを確認します。

1　vSphere 向け Kubernetes CLI Tools をインストールします。vSphere 向け Kubernetes CLI Tools のダウンロードとインストールを参照してください。

2　スーパーバイザー クラスタ にログインします。

```
kubectl vsphere login --server IP-ADDRESS-SUPERVISOR-CLUSTER --vsphere-username VCENTER-SSO-USERNAME
```

3　コンテキストをターゲット vSphere 名前空間 に切り替えます。

```
kubectl config use-context SUPERVISOR-NAMESPACE
```

4　使用可能な Tanzu Kubernetes リリース を一覧表示して、説明を記述します。

```
kubectl get tanzukubernetesreleases
```

```
kubectl describe tanzukubernetesreleases
```

5　使用可能な仮想マシン クラスを一覧表示します。

```
kubectl get virtualmachineclassbindings
```

名前空間を構成したら、Tanzu Kubernetes クラスタをプロビジョニングできます。TKGS v1alpha2 API を使用して Tanzu Kubernetes クラスタをプロビジョニングするためのワークフローを参照してください。ローカル コンテンツ ライブラリを使用している場合は、ライブラリにアップロードした OVA を指定する必要があります。Tanzu Kubernetes Grid サービス v1alpha1 API を使用した Tanzu Kubernetes クラスタのプロビジョニングの例を参照してください。

NSX スーパーバイザー クラスタ 名前空間へのセキュリティ ポリシーの追加

NSX ネットワークを使用する スーパーバイザー クラスタ は、セキュリティ ポリシー CRD を介して構成されたネットワーク セキュリティ ポリシーをサポートします。

セキュリティ ポリシーの作成

DevOps は、セキュリティ ポリシー CRD を構成し、NSX ベースのセキュリティ ポリシーが スーパーバイザー クラスタ 名前空間に適用されるようにすることができます。セキュリティ ポリシーは、vSphere ポッド および仮想マシンのトラフィックを保護します。仮想マシンには、TKG クラスタ ノードと、スーパーバイザー クラスタ にデプロイされた他の仮想マシンが含まれます。

前提条件

NSX バージョン 3.2 以降を使用します。

手順

1 セキュリティ ポリシー CRD を作成します。

使用するフィールドと CRD の例については、GitHub の NSX Operator Security Policy CRD ドキュメントを参照してください。

2 Kubernetes 環境の名前空間にアクセスします。

スーパーバイザー クラスタ コンテキストの取得と使用を参照してください。

3 セキュリティ ポリシーを名前空間に適用します。

```
kubectl apply -f policy-name.yaml
```

4 セキュリティ ポリシーを表示します。

a セキュリティ ポリシーの詳細を表示します。

```
kubectl get securitypolicy policy-name
```

b セキュリティ ポリシーの説明を表示します。

```
kubectl describe securitypolicy policy-name
```

結果

NSX ユーザー インターフェイスを使用して、ポリシーの詳細を表示することもできます。詳細については、VMware NSX のドキュメントページを参照してください。

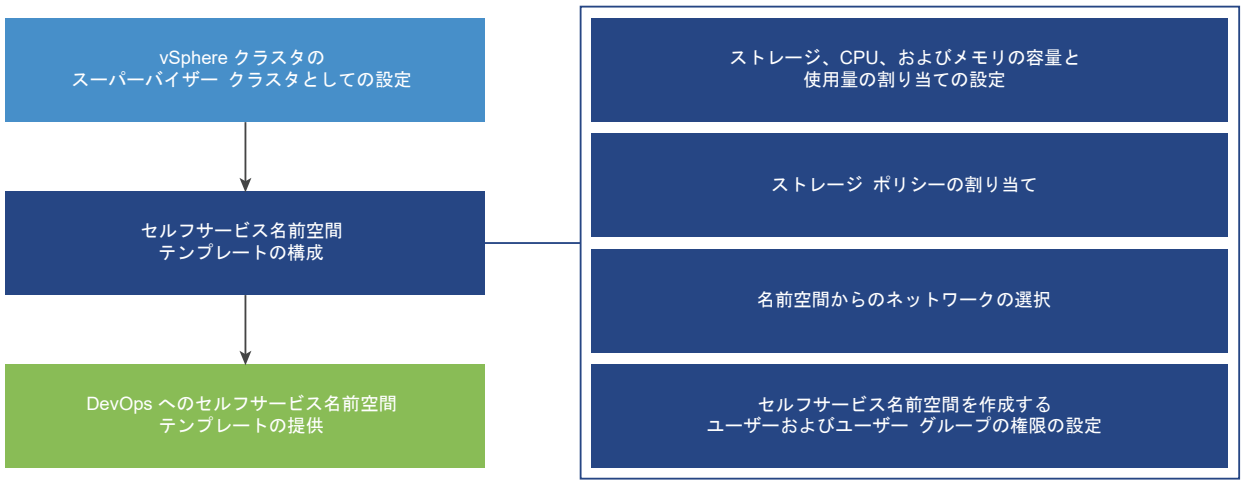
セルフサービス名前空間テンプレートのプロビジョニング

vSphere 管理者は、スーパーバイザー名前空間の作成、名前空間に対する CPU、メモリ、ストレージの制限の設定、権限の割り当て、およびクラスタの名前空間サービスのテンプレートとしての有効化を行うことができます。したがって、DevOps エンジニアはセルフ サービス方式でスーパーバイザー名前空間を作成し、その中にワークロードをデプロイすることができます。

セルフサービス名前空間の作成と構成のワークフロー

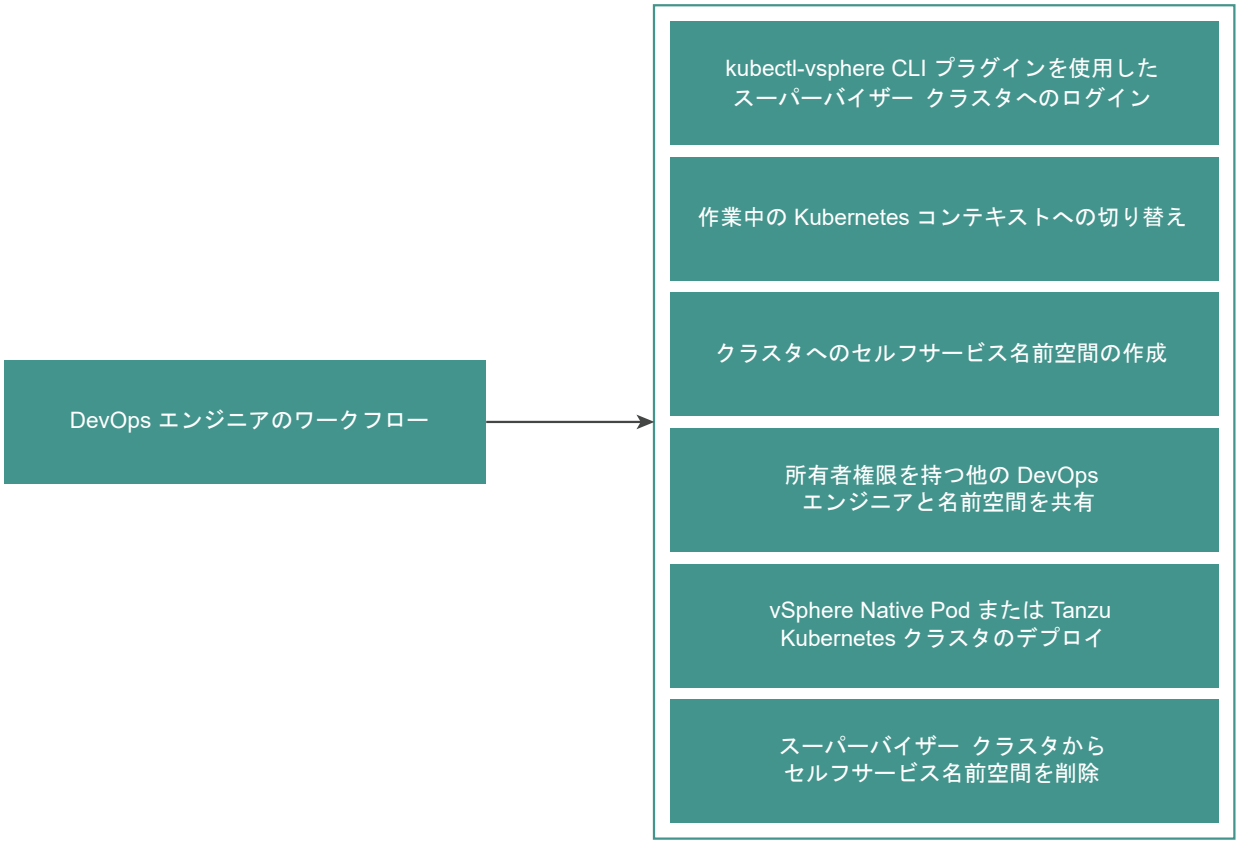
vSphere 管理者は、スーパーバイザー名前空間の作成、名前空間に対する CPU、メモリ、ストレージの制限の設定、権限の割り当て、およびクラスタの名前空間サービスのテンプレートとしてのプロビジョニングと有効化を行うことができます。

図 7-1. セルフサービス名前空間テンプレートのプロビジョニングのワークフロー



DevOps エンジニアは、セルフ サービス方式でスーパーバイザー名前空間を作成し、その中にワークロードをデプロイすることができます。また、他の DevOps エンジニアと名前空間を共有したり、不要になった名前空間を削除したりできます。名前空間を他の DevOps エンジニアと共有するには、vSphere 管理者にお問い合わせください。

図 7-2. セルフサービス名前空間の作成のワークフロー



セルフサービス名前空間テンプレートの作成と構成

vSphere 管理者は、スーパーバイザー名前空間をセルフサービス名前空間テンプレートとして作成および構成することができます。DevOps エンジニアは、`kubectl` コマンド ラインを使用してスーパーバイザー名前空間を作成および削除できます。

前提条件

vSphere with Tanzu でクラスタを設定します。

手順

1 vSphere Client で、スーパーバイザー クラスタ が有効になっている vSphere クラスタを選択します。

2 [構成] タブで、[名前空間] - [全般] の順に選択します。

3 [名前空間サービス] を選択します。

4 [ステータス] スイッチを切り替えて、この機能を有効にします。

[名前空間テンプレートの作成] 画面が表示されます。

5 [構成] ペインで、名前空間に対するリソース制限を構成します。

オプション	説明
CPU	名前空間に予約する CPU リソースの量。
メモリ	名前空間に予約するメモリの量。
ストレージ	名前空間に予約するストレージ容量の合計。
ストレージ ポリシー	名前空間に関連付けた各ストレージ ポリシーに専用のストレージ容量を個別に設定します。
ネットワーク	[ネットワーク] ドロップダウン メニューから、名前空間のネットワークを選択します。

6 [次へ] をクリックします。

7 [権限] ペインで DevOps エンジニアとグループを追加して、テンプレートを使用して名前空間を作成できるようにします。

ID ソースとユーザーまたはグループを選択し、[次へ] をクリックします。

8 [確認] ペインに、構成するプロパティが表示されます。

プロパティを確認して、[完了] をクリックします。

結果

名前空間テンプレートが構成され、アクティブな状態になっています。vSphere 管理者としてテンプレートを編集できます。DevOps エンジニアは、テンプレートを使用して名前空間を作成できます。

セルフサービス名前空間の無効化

vSphere 管理者は、クラスタのセルフサービス名前空間を無効にできます。

セルフサービス名前空間テンプレートを無効にすると、DevOps エンジニアはテンプレートを使用してクラスタに新しい名前空間を作成できなくなります。自分が作成した名前空間を削除することはできます。

手順

1 vSphere Client でスーパーバイザー クラスタに移動します。

2 [構成] タブの [名前空間] で [全般] を選択します。

3 [名前空間のセルフサービス] ペインで、[ステータス] スイッチを切り替えてテンプレートを無効にします。

4 テンプレートを再度有効にするには、[ステータス] スイッチを切り替えます。

別のセルフサービス名前空間を作成するか、既存の名前空間を使用することができます。

セルフサービス名前空間の作成

DevOps エンジニアはセルフサービス名前空間を作成し、その中でワークロードを実行することができます。名前空間を作成した後で、他の DevOps エンジニアと共有したり、不要になった場合に削除したりできます。

前提条件

- vSphere 管理者がクラスタにセルフサービス名前空間テンプレートを作成し、有効にしたことを確認します。セルフサービス名前空間テンプレートの作成と構成 を参照してください。
- セルフサービス名前空間テンプレートの権限リストに、自分が個別に、またはグループのメンバーとして追加されていることを確認します。
- スーパーバイザー クラスタ 制御プレーンの IP アドレスを取得します。

手順

1 kubectl 向けの vSphere プラグイン を使用して、スーパーバイザー クラスタ での認証を行います。vCenter Single Sign-On ユーザーとして スーパーバイザー クラスタ に接続するを参照してください。

```
kubectl vsphere login --server=IP-ADDRESS --vsphere-username USERNAME
```

2 コンテキストを スーパーバイザー クラスタ に切り替えます。

```
kubectl config use-context SUPERVISOR-CLUSTER-IP
```

3 クラスタにセルフサービス名前空間を作成します。

```
kubectl create namespace NAMESPACE NAME
```

例

```
kubectl create namespace test-ns
```

注： vSphere with Tanzu を有効にし、クラスタをアップグレードすると、DevOps エンジニアは所有者権限を使用できるようになります。クラスタではなく vCenter Server のみをアップグレードした場合、Devops エンジニアに付与されるのは名前空間に対する編集権限のみです。

作成した名前空間がクラスタに表示されます。名前空間を他の DevOps エンジニアと共有するには、vSphere 管理者にお問い合わせください。

注釈とラベルを含むセルフサービス名前空間の作成

DevOps エンジニアは、kubectl コマンド ラインを使用して、注釈とラベルを含むセルフサービス名前空間を作成できます。

DevOps エンジニアは、ユーザー定義の注釈とラベルを含む YAML マニフェストを使用できます。

手順

1 スーパーバイザー クラスタにログインします。

```
kubectl vsphere login --server IP-ADDRESS-SUPERVISOR-CLUSTER --vsphere-username VCENTER-SSO-USERNAME
```

2 注釈とラベルを含む名前空間 YAML マニフェスト ファイルを作成します。

```
kubectl create -f ns-create.yaml
```

たとえば、次の ns-create.yaml ファイルを作成します。

```
apiVersion: v1
kind: Namespace
metadata:
  name: test-ns-yaml
  labels:
    my-label: "my-label-val-yaml"
  annotations:
    my-ann-yaml: "my-ann-val-yaml"
```

3 YAML マニフェストを適用します。

```
kubectl create -f ns-create.yaml
```

または

```
kubectl apply -f ns-create.yaml
```

4 名前空間を指定した describe コマンドを実行し、変更を確認します。

```
root@localhost [ /tmp ]# kubectl describe ns test-ns-yaml
Name:         test-ns-yaml
Labels:       my-label=my-label-val-yaml
              vSphereClusterID=domain-c50
Annotations:  my-ann-yaml: my-ann-val-yaml
              vmware-system-namespace-owner-count: 1
              vmware-system-resource-pool: resgroup-171
              vmware-system-resource-pool-cpu-limit: 0.4770
              vmware-system-resource-pool-memory-limit: 2000Mi
              vmware-system-self-service-namespace: true
              vmware-system-vm-folder: group-v172
Status:       Active

Resource Quotas
 Name:             test-ns-yaml
 Resource          Used  Hard
 --------          ---   ---
 requests.storage  0     5000Mi

 Name:                                                                          test-ns-
yaml-storagequota
 Resource                                                                       Used  Hard
 --------                                                                       ---   ---
 namespace-service-storage-profile.storageclass.storage.k8s.io/requests.storage  0
9223372036854775807

No LimitRange resource.
```

kubectl annotate および kubectl label を使用したセルフサービス名前空間の更新

DevOps エンジニアは、kubectl annotate コマンドと kubectl label コマンドを使用して、セルフサービス名前空間の注釈とラベルを更新または削除できます。

前提条件

更新する名前空間に対する所有者権限があることを確認します。

手順

1 スーパーバイザー クラスタにログインします。

`kubectl vsphere login --server IP-ADDRESS-SUPERVISOR-CLUSTER --vsphere-username VCENTER-SSO-USERNAME`

2 更新する名前空間を定義します。

```
root@localhost [ /tmp ]# kubectl describe ns testns
Name:         testns
Labels:       my-label=test-label-2
              vSphereClusterID=domain-c50
Annotations:  my-ann: test-ann-2
              vmware-system-namespace-owner-count: 2
              vmware-system-resource-pool: resgroup-153
              vmware-system-resource-pool-cpu-limit: 0.4770
              vmware-system-resource-pool-memory-limit: 2000Mi
              vmware-system-self-service-namespace: true
              vmware-system-vm-folder: group-v154
Status:       Active


Resource Quotas
 Name:             testns
 Resource          Used  Hard
 --------          ---   ---
 requests.storage  0     5000Mi


 Name:                                                                          testns-
storagequota
 Resource                                                                       Used  Hard
 --------                                                                       ---   ---
 namespace-service-storage-profile.storageclass.storage.k8s.io/requests.storage  0
9223372036854775807
```

3 `kubectl annotate` コマンドを使用して注釈を更新します。

たとえば、`kubectl annotate --overwrite ns testns my-ann="test-ann-3"`。

注釈を削除するには、`kubectl annotate --overwrite ns testns my-ann-` コマンドを実行します。

4 `kubectl label` コマンドを使用してラベルを更新します。

たとえば、`kubectl label --overwrite ns testns my-label="test-label-3"`。

ラベルを削除するには、`kubectl label --overwrite ns testns my-label-` コマンドを実行します。

5 更新を表示する名前空間を定義します。

```
root@localhost [ /tmp ]# kubectl describe ns testns
Name:         testns
Labels:       my-label=test-label-3
              vSphereClusterID=domain-c50
Annotations:  my-ann: test-ann-3
              vmware-system-namespace-owner-count: 2
              vmware-system-resource-pool: resgroup-153
              vmware-system-resource-pool-cpu-limit: 0.4770
              vmware-system-resource-pool-memory-limit: 2000Mi
              vmware-system-self-service-namespace: true
              vmware-system-vm-folder: group-v154
Status:       Active


Resource Quotas
 Name:             testns
 Resource          Used  Hard
 --------          ---   ---
 requests.storage  0     5000Mi


 Name:                                                                                testns-
storagequota
 Resource                                                                             Used  Hard
 --------                                                                             ---   ---
 namespace-service-storage-profile.storageclass.storage.k8s.io/requests.storage  0
9223372036854775807


No LimitRange resource.
```

kubectl edit を使用したセルフサービス名前空間の更新

DevOps エンジニアは、`kubectl edit` コマンドを使用してセルフサービス名前空間を更新できます。

前提条件

更新する名前空間に対する所有者権限があることを確認します。

手順

1 スーパーバイザー クラスタにログインします。

```
kubectl vsphere login --server IP-ADDRESS-SUPERVISOR-CLUSTER --vsphere-
username VCENTER-SSO-USERNAME
```

2 更新する名前空間を定義します。

```
kubectl describe ns testns-1
Name:         testns
Labels:       vSphereClusterID=domain-c50
Annotations:  my-ann: test-ann-2
              vmware-system-namespace-owner-count: 2
```

```
              vmware-system-resource-pool: resgroup-153
              vmware-system-resource-pool-cpu-limit: 0.4770
              vmware-system-resource-pool-memory-limit: 2000Mi
              vmware-system-self-service-namespace: true
              vmware-system-vm-folder: group-v154
Status:       Active


Resource Quotas
 Name:             testns-1
 Resource          Used  Hard
 --------          ---   ---
 requests.storage  0     5000Mi


 Name:                                                                                                                testns-1-
storagequota
 Resource                                                                                                             Used  Hard
 --------                                                                                                             ---   ---
 namespace-service-storage-profile.storageclass.storage.k8s.io/requests.storage  0
9223372036854775807
```

3 `kubectl edit` コマンドを使用して名前空間を編集します。

たとえば、`kubectl edit ns testns-1`。

`kubectl edit` コマンドを実行すると、KUBE_EDITOR または EDITOR の環境変数によって定義されたテキスト エディタ内に名前空間のマニフェストが開きます。

4 ラベルを更新します。

たとえば、`my-label=test-label`。

5 注釈を更新します。

たとえば、`my-ann: test-ann`。

6 更新を表示する名前空間を定義します。

```
root@localhost [ /tmp ]# kubectl describe ns testns-1
Name:         testns-1
Labels:       my-label=test-label
              vSphereClusterID=domain-c50
Annotations:  my-ann: test-ann
              vmware-system-namespace-owner-count: 1
              vmware-system-resource-pool: resgroup-173
              vmware-system-resource-pool-cpu-limit: 0.4770
              vmware-system-resource-pool-memory-limit: 2000Mi
              vmware-system-self-service-namespace: true
              vmware-system-vm-folder: group-v174
Status:       Active


Resource Quotas
 Name:             testns-1
 Resource          Used  Hard
```

```
 --------           ---   ---
 requests.storage   0     5000Mi


 Name:                                                                                 testns-1-
storagequota
 Resource                                                                              Used  Hard
 --------                                                                              ---   ---
 namespace-service-storage-profile.storageclass.storage.k8s.io/requests.storage  0
9223372036854775807


No LimitRange resource.
```

セルフサービス名前空間の削除

DevOps エンジニアは、自分が作成したセルフサービス名前空間を削除できます。

前提条件

kubectl 向けの vSphere プラグイン を使用して、セルフサービス名前空間が作成されていることを確認します。

手順

1 kubectl 向けの vSphere プラグイン を使用して、スーパーバイザー クラスタ での認証を行います。vCenter Single Sign-On ユーザーとして スーパーバイザー クラスタ に接続するを参照してください。

2 クラスタからセルフサービス名前空間を削除します。

```
kubectl delete namespace NAMESPACE NAME
```

例：

```
kubectl delete namespace test-ns
```

vSphere with Tanzu を使用した スーパーバイザー サービス の管理

8

スーパーバイザー サービス は、Infrastructure-as-a-Service コンポーネントと緊密に統合された独立系ソフトウェア ベンダー サービスを開発者に提供する vSphere 認定の Kubernetes オペレータです。vSphere with Tanzu 環境に スーパーバイザー サービス をインストールして管理し、Kubernetes ワークロードで使用可能にすることができます。スーパーバイザー サービス が スーパーバイザー クラスタ にインストールされている場合、DevOps エンジニアはサービス API を使用して、ユーザー名前空間内の スーパーバイザー クラスタ にインスタンスを作成できます。これらのインスタンスは、vSphere ポッド および Tanzu Kubernetes クラスタで使用できます。

サポートされている スーパーバイザー サービス の詳細と、サービス YAML ファイルのダウンロード方法については、http://vmware.com/go/supervisor-service を参照してください。

vSphere Services プラットフォームの スーパーバイザー サービス の管理は、vSphere Client から実行します。プラットフォームを使用することで、スーパーバイザー サービス のライフサイクル管理、スーパーバイザー クラスタ へのインストール、およびバージョン管理を行うことができます。1 つの スーパーバイザー クラスタ で一度に実行できる スーパーバイザー サービス のバージョンは 1 つのみであるため、複数のバージョンを使用するには、複数の スーパーバイザー クラスタ にインストールします。

表 8-1. スーパーバイザー サービス の状態

都道府県	サービス バージョン	サービス全体
アクティブ	サービス バージョンを スーパーバイザー クラスタ バージョンにインストールする準備ができました。	1 つ以上のサービス バージョンがアクティブな状態です。
Deactivated（非アクティブ）	サービス バージョンを スーパーバイザー クラスタ にインストールできません。サービス バージョンがインストールされているすべての スーパーバイザー クラスタ で実行を継続できますが、非アクティブなバージョンを新しい スーパーバイザー クラスタ にインストールすることはできません。	スーパーバイザー サービス 全体が非アクティブになっている場合は、そのすべてのバージョンも非アクティブになっています。サービスを再アクティブにするまでは、スーパーバイザー クラスタ にサービスをインストールしたり、新しいサービス バージョンを追加したりできません。

スーパーバイザー サービス ライフ サイクル管理の操作

スーパーバイザー サービス のライフ サイクルの管理には、次の操作が含まれます。

- vCenter Server への新しい スーパーバイザー サービス の追加。vCenter Server に新しいサービスを追加すると、サービスとそのサービスに関するすべての情報が vCenter Server に登録されます。サービスはどのスーパーバイザー クラスタ にもインストールされていません。サービスが vCenter Server に登録されると、サービスの状態はアクティブになり、このサービスを スーパーバイザー クラスタ にインストールできるようになります。
- vCenter Server への新しい スーパーバイザー サービス バージョンの追加。vCenter Server に スーパーバイザー サービス を追加すると、そのサービスの新しいバージョンを追加できるようになります。新しいサービス バージョンを vCenter Server に登録すると、このバージョンがアクティブな状態になり、スーパーバイザー クラスタ にこのバージョンをインストールできるようになります。
- スーパーバイザー クラスタ への スーパーバイザー サービス のインストール。サービスを スーパーバイザー クラスタ にインストールすると、サービス YAML ファイルがクラスタに適用されて、すべてのポッドとサービスが動作するために必要なリソースが作成されます。スーパーバイザー クラスタ にインストールしたサービスごとに、サービス リソースを管理するための専用の名前空間があります。スーパーバイザー サービス に、vCenter Server 用のユーザー インターフェイス プラグインが用意されていて、サービス構成を管理できる場合もあります。
- スーパーバイザー サービス へのアップグレード。スーパーバイザー クラスタ にインストールされているサービスをアップグレードするには、まず新しいサービス バージョンを vCenter Server に追加してから、スーパーバイザー クラスタ に新しいバージョンをインストールします。サービスのアップグレード中に、新しいバージョンの YAML ファイルが スーパーバイザー クラスタ に適用されます。以前のサービス バージョンで指定されているリソースのうち、新しいバージョンで不要なものは、削除されます。たとえば、バージョン 1 がポッド A を指定し、バージョン 2 がポッド B を指定している場合に、バージョン 2 にアップグレードすると、新しいポッド B が作成され、ポッド A は削除されます。現在実行中のワークロードは、処理中に影響を受けません。
- スーパーバイザー サービス バージョンのアンインストール。スーパーバイザー クラスタ からサービス バージョンをアンインストールすると、サービスの名前空間を含むクラスタからすべてのサービス リソースが削除されます。Kubernetes ワークロード内のサービスのアプリケーション インスタンスは引き続き実行されます。
- スーパーバイザー サービス バージョンの削除。サービス バージョンを削除するには、まずそのバージョンを非アクティブにして、そのバージョンが実行されている スーパーバイザー クラスタ からアンインストールする必要があります。その後、vCenter Server からサービス バージョンを削除できます。
- スーパーバイザー サービス 全体の削除。サービス全体を削除するには、そのすべてのバージョンを非アクティブにし、その後これらのバージョンを スーパーバイザー クラスタ からアンインストールして、最後にすべてのサービス バージョンを削除する必要があります。

この章には、次のトピックが含まれています。

- スーパーバイザー クラスタ にインストールされている スーパーバイザー サービス の表示
- スーパーバイザー サービス またはバージョンの無効化
- vCenter Server における任意のバージョンの スーパーバイザー サービス の有効化
- スーパーバイザー クラスタ からの スーパーバイザー サービス のアンインストール
- 特定のバージョンの スーパーバイザー サービス の削除
- スーパーバイザー サービス の削除

vCenter Server への スーパーバイザー サービス の追加

スーパーバイザー サービス は、vSphere with Tanzu が稼動している vCenter Server システムに追加できます。vCenter Server にサービスを追加した後、スーパーバイザー クラスタ に スーパーバイザー サービス をインストールします。これにより、DevOps エンジニアがサービスを Kubernetes ワークロードで使用できるようになります。

- サポートされている スーパーバイザー サービス の詳細と、サービス YAML ファイルのダウンロード方法については、http://vmware.com/go/supervisor-service を参照してください。

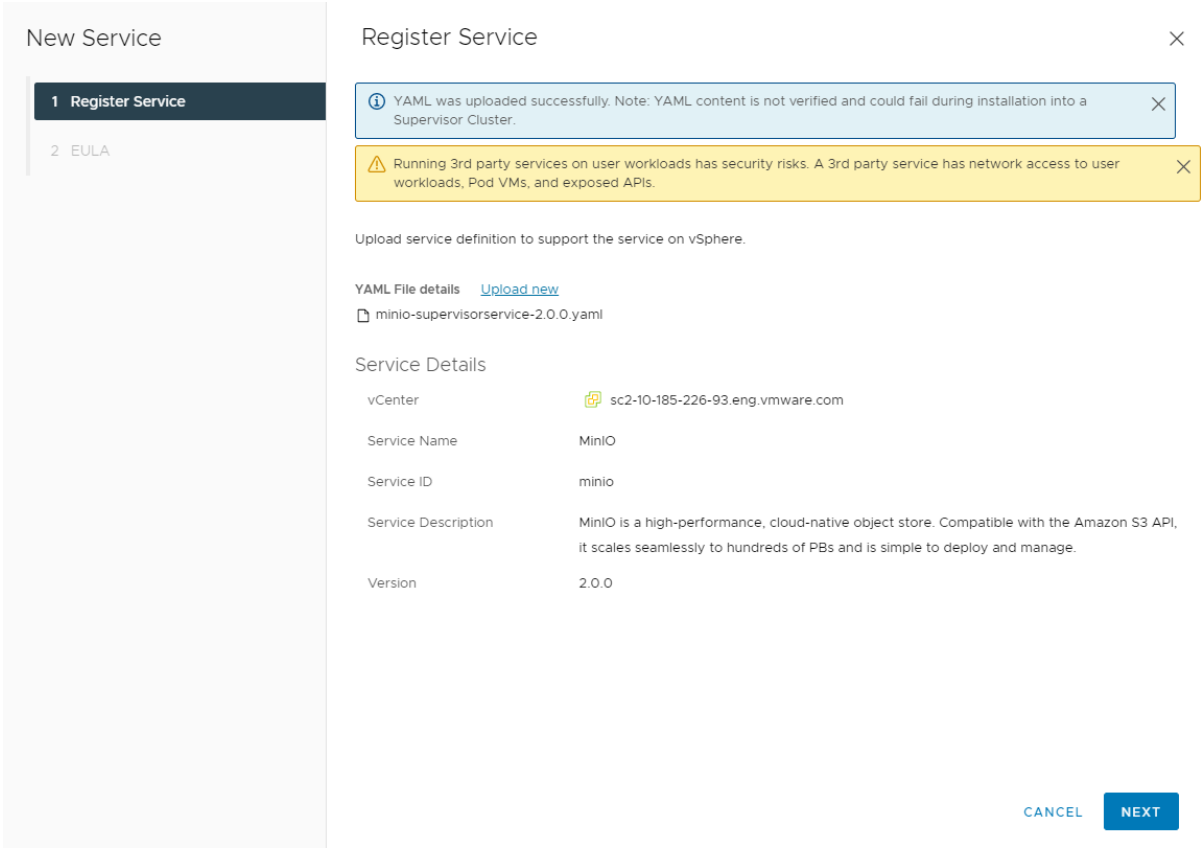
前提条件

- サービスを追加する vCenter Server システム上でスーパーバイザー サービスの管理権限があることを確認します。

手順

1. vSphere Client ホーム メニューから、[ワークロード管理] を選択します。
2. [[サービス]] を選択します。
3. 上部のドロップダウン メニューから vCenter Server システムを選択します。

4 [サービスの新規追加] カードで、サービス YAML ファイルをドラッグ アンド ドロップします。

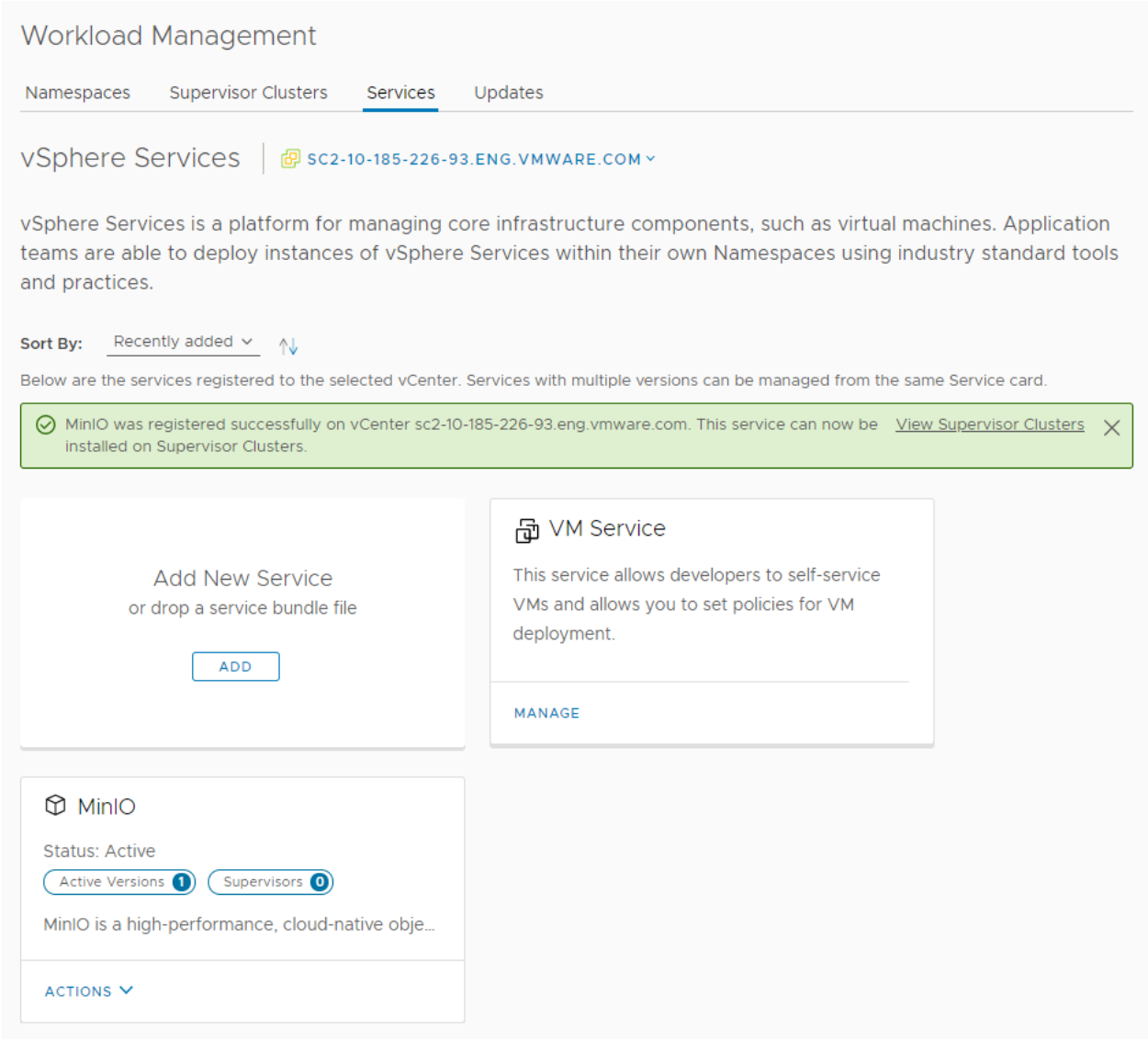


5 [次へ] をクリックします。EULA が表示された場合は、同意します。

6 [終了] をクリックします。

結果

スーパーバイザー サービス とそのすべての情報が vCenter Server システムに登録されます。サービスは [アクティブ] 状態になります。



次のステップ

スーパーバイザー クラスタ に スーパーバイザー サービス をインストールして、DevOps エンジニアが Kubernetes ワークロードで使用できるようにします。スーパーバイザー クラスタ への スーパーバイザー サービス のインストールを参照してください。

スーパーバイザー クラスタ への スーパーバイザー サービス のインストール

vCenter Server に追加した スーパーバイザー サービス は、vSphere with Tanzu 環境内の スーパーバイザー クラスタ にインストールできます。スーパーバイザー サービス の新しいバージョンをインストールすると、その スー

パーバイザー クラスタ に古いバージョンのサービスがあった場合に、オーバーライドされます。スーパーバイザー クラスタ では、一度に 1 つのバージョンの スーパーバイザー クラスタ のみが実行可能です。

- サポートされている スーパーバイザー サービス の詳細と、サービス YAML ファイルのダウンロード方法については、http://vmware.com/go/supervisor-service を参照してください。

前提条件

- 新しい スーパーバイザー サービス、または既存のサービスの新しいバージョンを vCenter Server に追加します。vCenter Server への スーパーバイザー サービス の追加またはスーパーバイザー サービス への新しいバージョンの追加を参照してください。
- サービスのインストール先である スーパーバイザー クラスタ をホストする vCenter Server システムで、スーパーバイザー サービスの管理 権限があることを確認します。
- スーパーバイザー サービス がパーシステント ストレージを必要とする場合は、vSAN データ パーシステンス プラットフォームを構成します。最新のステートフル サービスでの vSAN データ パーシステンス プラットフォームの使用を参照してください。

手順

1 vSphere Client ホーム メニューから、[ワークロード管理] を選択します。

2 [[サービス]] を選択します。

3 インストールする スーパーバイザー サービス のバージョンを選択します。

> **注：** 無効になっているバージョンの スーパーバイザー サービス はインストールできません。

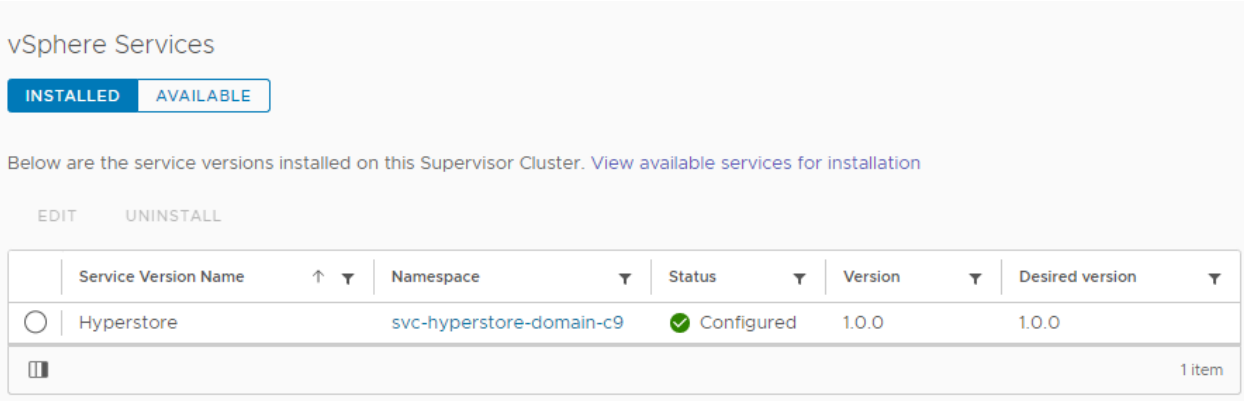
4 サービスをインストールする スーパーバイザー クラスタ を選択します。

5 イメージをプライベート レジストリでホストしている場合は、レジストリ エンドポイント、ユーザー名、パスワードを入力します。

   サービスがキーと値のペアを必要とする場合、他のペアが存在する可能性があります。詳細については、サービスのドキュメントを参照してください。

結果

スーパーバイザー サービス は、[設定中] 状態です。これは、必要なすべてのリソースが スーパーバイザー クラスタ で作成中であり、サービス YAML がクラスタに適用されていることを意味します。YAML が スーパーバイザー クラスタ に正常に適用されて、YAML で指定されているすべてのリソースと名前空間が作成または更新されると、サービスの状態が [設定済み] に変わります。サービスは、そのクラスタ内のすべての名前空間で使用でき、DevOps エンジニアは Kubernetes ワークロードで使用できます。



注： スーパーバイザー クラスタ は、スーパーバイザー サービス が実際に動作しているかどうかを監視していません。たとえば、誤ったレジストリ ユーザー名またはパスワードを指定した場合、サービスは実行に必要なイメージを取得できない可能性があります。そのため、サービスで [設定済み] と表示されても実際には機能していない可能性があります。スーパーバイザー サービス が動作しているかどうかを確認するには、サービスの名前空間と実行中のポッドを確認します。

次のステップ

インターフェイスを使用して、スーパーバイザー サービス を構成します。確認する場所については、スーパーバイザー クラスタ の スーパーバイザー サービス の管理インターフェイスへのアクセスを参照してください。

スーパーバイザー クラスタ の スーパーバイザー サービス の管理インターフェイスへのアクセス

スーパーバイザー サービス の管理ユーザー インターフェイスを スーパーバイザー クラスタ にインストールした後、このインターフェイスが表示される場所を確認します。スーパーバイザー サービス には、vSphere Client の スーパーバイザー クラスタ ビューにサービス インターフェイスを追加する、vCenter Server 独自のユーザー インターフェイス プラグインが提供されています。スーパーバイザー サービス の仕様によっては、このインターフェイスを使用してサービスを構成および管理し、このサービスのサービス インスタンスをデプロイすることもできます。

手順

1. vSphere Client で、スーパーバイザー クラスタ に移動します。
2. [構成] を選択して、サービス インターフェイスまでスクロールします。このインターフェイスには、通常、サービスに基づいた名前が付いています（[MinIO] など）。

スーパーバイザー サービス への新しいバージョンの追加

vSphere with Tanzu 環境がある vCenter Server に スーパーバイザー サービス を追加すると、そのサービスに新しいバージョンを追加できるようになります。スーパーバイザー クラスタ には、さまざまなバージョンのサービスをインストールできます。

- サポートされている スーパーバイザー サービス の詳細と、サービス YAML ファイルのダウンロード方法については、http://vmware.com/go/supervisor-service を参照してください。

前提条件

- サービスを vCenter Server に追加します。vCenter Server への スーパーバイザー サービス の追加を参照してください。
- 新しいバージョンのサービスを追加する vCenter Server システム上でスーパーバイザー サービスの管理権限があることを確認します。

手順

1. vSphere Client ホーム メニューから、[ワークロード管理] を選択します。
2. [[サービス]] を選択します。
3. 新しいバージョンを追加するサービスのカードで、[アクション] - [新規バージョンの追加] の順に選択します。
4. 新しいバージョンのサービスの YAML ファイルをアップロードして、[次へ] をクリックします。
5. EULA が表示された場合は同意して、[完了] をクリックします。

結果

新しいバージョンのサービスが追加され、[アクティブ] 状態になります。

次のステップ

スーパーバイザー クラスタ に新しいバージョンのサービスをインストールします。スーパーバイザー クラスタ への スーパーバイザー サービス のインストールを参照してください。

スーパーバイザー クラスタ にインストールされている スーパーバイザー サービス の表示

vSphere with Tanzu 環境の スーパーバイザー クラスタ にインストールされている vSphere サービスを表示します。スーパーバイザー クラスタ にインストールされている スーパーバイザー サービス は、クラスタの各名前空間で使用できます。

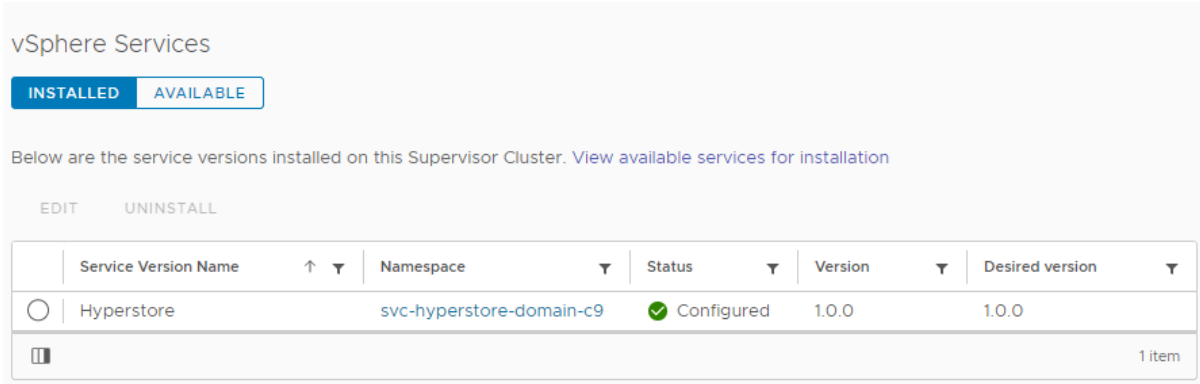
前提条件

- vCenter Server に スーパーバイザー サービス を追加します。vCenter Server への スーパーバイザー サービス の追加を参照してください。
- スーパーバイザー クラスタ に スーパーバイザー サービス をインストールします。スーパーバイザー クラスタ への スーパーバイザー サービス のインストールを参照してください。

手順

1. vSphere Client ホーム メニューから、[ワークロード管理] を選択します。
2. [スーパーバイザー クラスタ] タブを選択します。

3 スーパーバイザー クラスタ の [サービス] 列で [表示] をクリックします。



- [インストール済み] タブで、スーパーバイザー クラスタ に現在インストールされている スーパーバイザー サービス を確認します。
- [使用可能] タブで、インストールに使用できる スーパーバイザー サービス を確認します。

次のステップ

この スーパーバイザー クラスタ の スーパーバイザー サービス の管理や、サービスのアンインストールを実行したり、[使用可能] タブのサービスの中から新しいサービスをインストールしたりできます。

スーパーバイザー サービス またはバージョンの無効化

特定のバージョンの スーパーバイザー サービス が vSphere with Tanzu 環境内の Kubernetes ワークロードで使用する必要がなくなった場合には、無効にします。無効にされたサービス バージョンは、インストールされている スーパーバイザー クラスタ では引き続き実行されますが、他の スーパーバイザー クラスタ にインストールすることはできません。サービス全体を無効にすると、すべてのサービス バージョンが無効になり、サービスを再度有効にするまで、新しいサービス バージョンを追加したり、スーパーバイザー クラスタ にインストールしたりすることはできません。

前提条件

- vCenter Server レベルで スーパーバイザー サービスの管理 権限があることを確認します。

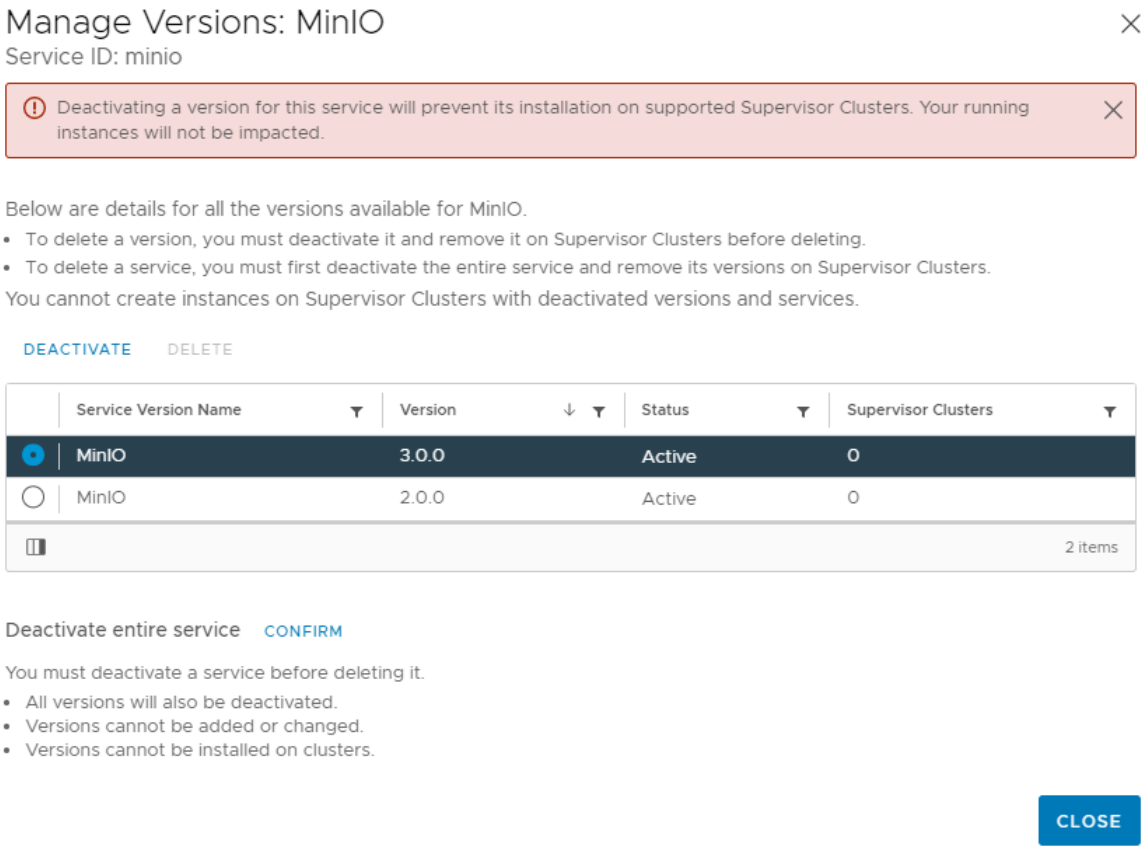
手順

1 vSphere Client ホーム メニューから、[ワークロード管理] を選択します。

2 [[サービス]] を選択します。

3 サービス カードで、[アクション] - [バージョンの管理] の順に選択します。

- 特定のバージョンの スーパーバイザー サービス を無効にするには、バージョンを選択し、[アクティベーションの解除] をクリックします。
- サービス全体を無効にするには、[サービス全体を無効にします] の横にある [確認] をクリックします。



結果

サービスのバージョンが無効になり、スーパーバイザー クラスタ にインストールできません。

vCenter Server における任意のバージョンの スーパーバイザー サービス の有効化

いずれかのバージョンの スーパーバイザー サービス を無効にした後で、vSphere with Tanzu 上で実行している Kubernetes ワークロードで同じバージョンのサービスを使用する必要が生じた場合には、再度有効にすることができます。

- サービスが登録されている vCenter Server システム上でスーパーバイザー サービスの管理権限があることを確認します。

手順

1. vSphere Client ホーム メニューから、[ワークロード管理] を選択します。
2. [[サービス]] を選択します。
3. スーパーバイザー サービス カードで、[アクティブなバージョン] をクリックします。
4. [バージョンの管理] を選択します。

5 非アクティブになっているバージョンの スーパーバイザー サービス を選択して、[再有効化] をクリックします。

スーパーバイザー クラスタ からの スーパーバイザー サービス のアンインストール

DevOps チームが vSphere with Tanzu 環境で実行されている Kubernetes ワークロードに スーパーバイザー サービス を使用する必要がなくなった場合は、スーパーバイザー クラスタ からこのサービスをアンインストールします。

前提条件

- サービスがインストールされている スーパーバイザー クラスタ をホストする vCenter Server システムに対して、スーパーバイザー サービスの管理権限があることを確認します。

手順

1 vSphere Client ホーム メニューから、[ワークロード管理] を選択します。

2 [スーパーバイザー クラスタ] を選択します。

3 スーパーバイザー クラスタ の [サービス] 列で [表示] をクリックします。

4 アンインストールする スーパーバイザー サービス を選択して、[アンインストール] をクリックします。

結果

スーパーバイザー クラスタ から スーパーバイザー サービス がアンインストールされます。すべてのサービス リソースとサービスの名前空間が スーパーバイザー クラスタ から削除されます。vSAN データ パーシステンス プラットフォームを使用するサービスのすべての管理対象インスタンスが スーパーバイザー クラスタ から削除されます。

特定のバージョンの スーパーバイザー サービス の削除

vCenter Server にある スーパーバイザー サービス のバージョンが古くなり、vSphere with Tanzu 環境で実行されている Kubernetes ワークロードで DevOps チームにとって不要になったときは、このバージョンを削除します。

前提条件

- 削除する スーパーバイザー サービス バージョンが スーパーバイザー クラスタ にインストールされていないことを確認します。スーパーバイザー クラスタ からの スーパーバイザー サービス のアンインストールを参照してください。
- vCenter Server レベルで スーパーバイザー サービスの管理 権限があることを確認します。

手順

1 vSphere Client ホーム メニューから、[ワークロード管理] を選択します。

2 [[サービス]] を選択します。

3 スーパーバイザー サービス カードで、[アクション] - [バージョンの管理] の順に選択します。

4 削除するバージョンを選択し、[無効化] をクリックします。

5 無効になったバージョンを選択し、[削除] をクリックします。

スーパーバイザー サービス の削除

vSphere with Tanzu 環境内の スーパーバイザー サービス が Kubernetes ワークロードで DevOps エンジニアにとって不要になったときは、削除します。

前提条件

- サービスが登録されている vCenter Server システム上でスーパーバイザー サービスの管理権限があることを確認します。

手順

1 vSphere Client ホーム メニューから、[ワークロード管理] を選択します。

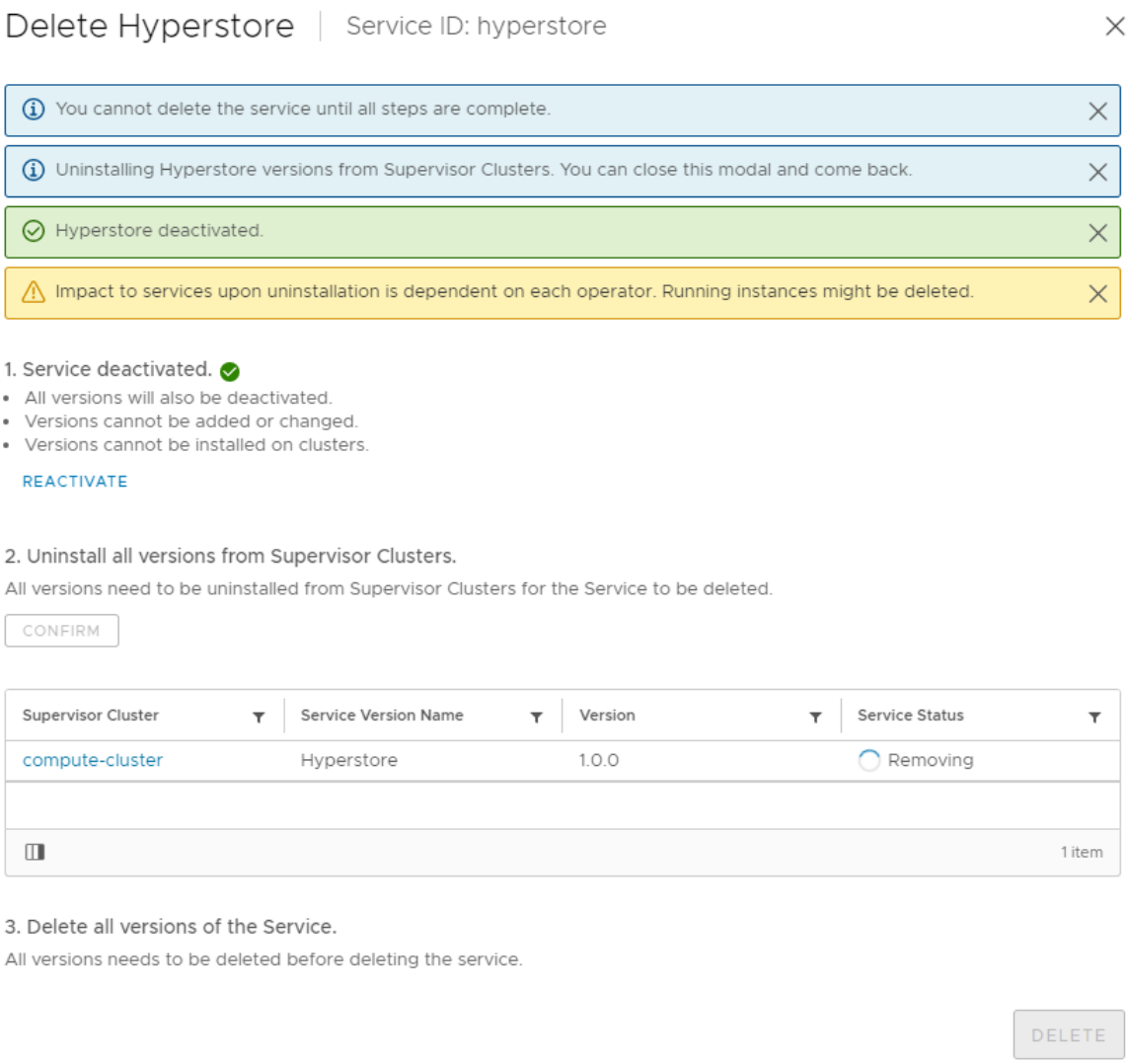
2 [[サービス]] を選択します。

3 削除する スーパーバイザー サービス のカードで、[アクション] - [削除] の順に選択します。

4 現在使用可能なすべてのサービス バージョンを無効にすることを確認します。

5 スーパーバイザー クラスタ からサービスをアンインストールすることを確認します。

スーパーバイザー サービス を、それが実行されている スーパーバイザー クラスタ からアンインストールするまでには、時間がかかることがあります。プロセスが完了するまでの間は、ダイアログを閉じることができます。次のステージに進むときには、ダイアログを再度開きます。



6 使用可能なすべてのバージョンのサービスを削除することを確認します。

7 [削除] をクリックします。

vSphere with Tanzu クラスタへの接続

9

スーパーバイザー クラスタ に接続して、Tanzu Kubernetes クラスタをプロビジョニングします。プロビジョニングが完了すると、さまざまな方法を使用して Tanzu Kubernetes クラスタに接続し、ロールと目的に基づいて認証を行うことができます。

この章には、次のトピックが含まれています。

- vSphere 向け Kubernetes CLI Tools のダウンロードとインストール
- vSphere with Tanzu クラスタでのセキュア ログインの構成
- vCenter Single Sign-On ユーザーとして スーパーバイザー クラスタ に接続する
- Tanzu Kubernetes クラスタを使用した認証
- vCenter Single Sign-On ユーザーとして Tanzu Kubernetes クラスタに接続
- 管理者としての Tanzu Kubernetes クラスタ制御プレーンへの接続
- プライベート キーを使用した、システム ユーザーとしての Tanzu Kubernetes クラスタ ノードへの SSH 接続
- パスワードを使用した、システム ユーザーとしての Tanzu Kubernetes クラスタ ノードへの SSH 接続
- 開発者に対する Tanzu Kubernetes クラスタへのアクセス権の付与

vSphere 向け Kubernetes CLI Tools のダウンロードとインストール

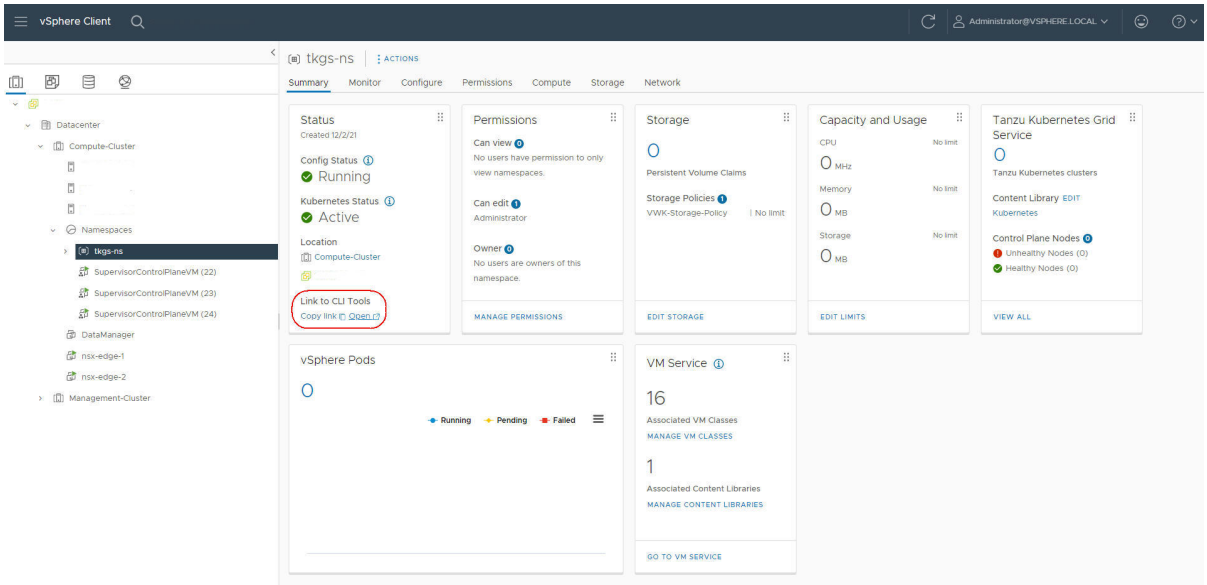
vSphere 向け Kubernetes CLI Tools を使用して、vSphere with Tanzu の名前空間とクラスタを表示および制御できます。

Kubernetes CLI Tools のダウンロード パッケージには、標準のオープンソース kubectl と kubectl 向けの vSphere プラグイン の 2 つの実行可能ファイルが含まれています。kubectl CLI は、プラグイン可能なアーキテクチャを備えています。kubectl 向けの vSphere プラグイン では、kubectl で使用できるコマンドが拡張されているため、vCenter Single Sign-On の認証情報を使用して スーパーバイザー クラスタ および Tanzu Kubernetes クラスタに接続できます。

注： ベスト プラクティスとして、vSphere 名前空間 のアップデートを実行し、スーパーバイザー クラスタ をアップグレードした後で、kubectl 向けの vSphere プラグイン をアップデートしてください。kubectl 向けの vSphere プラグイン の更新を参照してください。

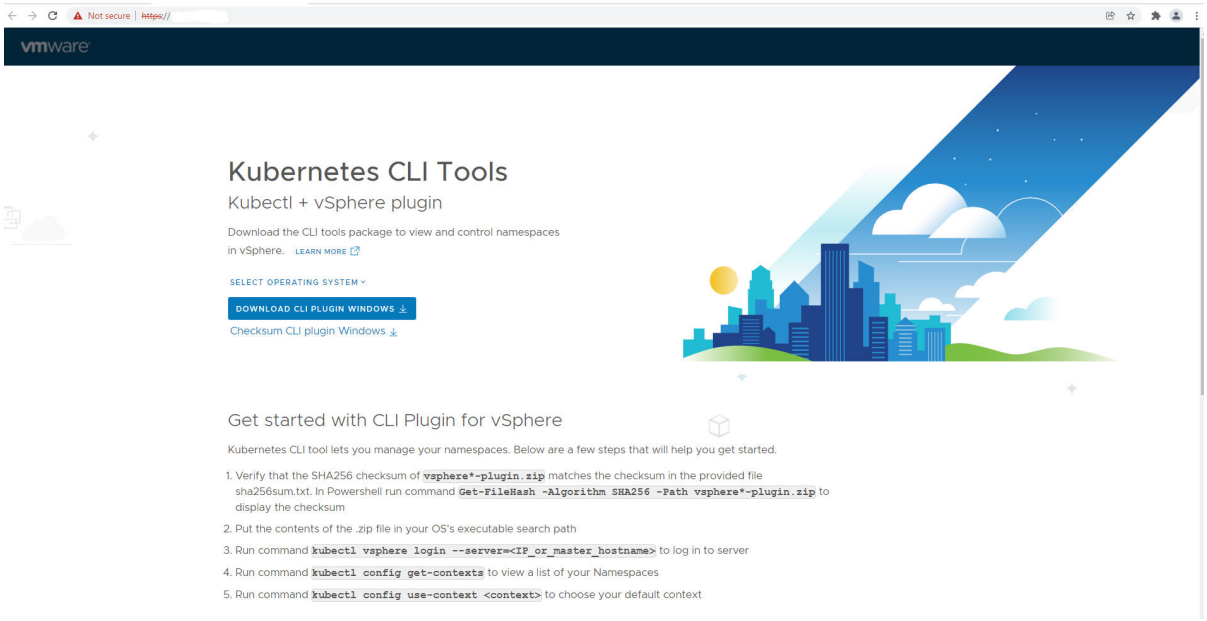
前提条件

- vSphere 管理者から Kubernetes CLI Tools ダウンロード ページのリンクを取得します。
- または、vCenter Server にアクセスできる場合は、次のようにしてリンクを取得します。
  - vSphere Client を使用して、vCenter Server にログインします。
  - [vSphere クラスタ] - [名前空間] に移動し、作業している vSphere 名前空間 を選択します。次の例では、これは、Tanzu Kubernetes クラスタ用に作成した「tkgs-ns」名前空間です。
  - [サマリ] タブを選択し、この画面の [ステータス] 領域を見つけます。
  - [CLI ツールへのリンク] という見出しの下にある [開く] を選択してダウンロード ページを開きます。または、リンクを [コピー] することもできます。



手順

1 ブラウザを使用して、環境に応じた [Kubernetes CLI Tools] ダウンロード URL に移動します。ダウンロード URL の検索方法については、上記の前提条件のセクションを参照してください。



2 オペレーティング システムを選択します。

3 `vsphere-plugin.zip` ファイルをダウンロードします。

4 この ZIP ファイルのコンテンツを作業ディレクトリに解凍します。

`vsphere-plugin.zip` パッケージには、kubectl と kubectl 向けの vSphere プラグイン の 2 つの実行可能ファイルが含まれています。`kubectl` は標準の Kubernetes CLI です。`kubectl-vsphere` は、vCenter Single Sign-On の認証情報を使用して スーパーバイザー クラスタ および Tanzu Kubernetes クラスタで認証を行う際に役立つ kubectl 向けの vSphere プラグイン です。

5 両方の実行ファイルの場所をシステムの PATH 変数に追加します。

6 kubectl CLI のインストールを確認するには、シェル、ターミナル、またはコマンド プロンプトのセッションを開始し、`kubectl` コマンドを実行します。

kubectl のバナー メッセージと、CLI のコマンドライン オプションのリストが表示されます。

7 kubectl 向けの vSphere プラグイン のインストールを確認するには、`kubectl vsphere` コマンドを実行します。

kubectl 向けの vSphere プラグイン のバナー メッセージと、プラグインのコマンドライン オプションのリストが表示されます。

次のステップ

vSphere with Tanzu クラスタでのセキュア ログインの構成。

vSphere with Tanzu クラスタでのセキュア ログインの構成

スーパーバイザー クラスタ や Tanzu Kubernetes クラスタなどの vSphere with Tanzu クラスタに安全にログインするには、適切な TLS 証明書を使用して kubectl 向けの vSphere プラグイン を構成し、プラグインの最新バージョンが実行されていることを確認します。

スーパーバイザー クラスタ CA 証明書

vSphere with Tanzu は、kubectl 向けの vSphere プラグイン コマンド `kubectl vsphere login …` を使用してクラスタ アクセスのための vCenter Single Sign-On をサポートします。このユーティリティをインストールして使用するには、vSphere 向け Kubernetes CLI Tools のダウンロードとインストール を参照してください。

kubectl 向けの vSphere プラグイン では、デフォルトで安全なログインが行われ、信頼されている証明書が必要とされます。デフォルトは、vCenter Server ルート CA によって署名された証明書です。プラグインは `--insecure-skip-tls-verify` フラグをサポートしていますが、これはセキュリティ上の理由から推奨されません。

kubectl 向けの vSphere プラグイン を使用して スーパーバイザー クラスタ および Tanzu Kubernetes クラスタに安全にログインするには、次の 2 つのオプションがあります。

オプション	方法
各クライアント マシンに vCenter Server ルート CA 証明書をダウンロードしてインストールします。	VMware ナレッジベースの記事 Web ブラウザで証明書に関する警告表示を出さないようにするために vCenter Server のルート証明書をダウンロードしてインストールする方法を参照してください。
スーパーバイザー クラスタ で使用される VIP 証明書を、各クライアント マシンが信頼する CA によって署名された証明書に置き換えます。	スーパーバイザー クラスタ API エンドポイントに安全に接続するための VIP 証明書の置き換えを参照してください

注： vCenter Single Sign-On、vCenter Server 証明書の管理とローテーション、認証のトラブルシューティングなど、vSphere 認証の詳細については、vSphere Authentication のドキュメントを参照してください。

Tanzu Kubernetes クラスタ CA 証明書

`kubectl` CLI を使用して Tanzu Kubernetes クラスタ API サーバと安全に接続するには、Tanzu Kubernetes クラスタ CA 証明書をダウンロードする必要があります。

kubectl 向けの vSphere プラグイン の最新バージョンを使用している場合、Tanzu Kubernetes クラスタに初めてログインすると、プラグインによって kubeconfig ファイルに Tanzu Kubernetes クラスタ CA 証明書が登録されます。この証明書は *`TANZU-KUBERNETES-NAME`*`-ca` という名前の Kubernetes シークレットに格納されます。プラグインは、この証明書を使用して、対応するクラスタの認証局データストアの CA 情報を入力します。

vSphere with Tanzu をアップデートする場合は、プラグインの最新バージョンに更新してください。kubectl 向けの vSphere プラグイン の更新を参照してください。

vCenter Single Sign-On ユーザーとして スーパーバイザー クラスタ に接続する

Tanzu Kubernetes Grid サービス を使用して vSphere ポッド または Tanzu Kubernetes クラスタをプロビジョニングするには、kubectl 向けの vSphere プラグイン を使用して スーパーバイザー クラスタ に接続し、vCenter Single Sign-On の認証情報によって認証します。

スーパーバイザー クラスタ にログインすると、kubectl 向けの vSphere プラグイン によってクラスタのコンテキストが生成されます。Kubernetes では、設定コンテキストにクラスタ、名前空間、およびユーザーが含まれます。クラスタのコンテキストは `.kube/config` ファイルで確認できます。このファイルは、通常、`kubeconfig` ファイルと呼ばれます。

注： 既存の `kubeconfig` ファイルがある場合は、そのファイルに各クラスタ コンテキストが追加されます。kubectl 向けの vSphere プラグイン は、kubectl 自体が使用する KUBECONFIG 環境変数に従います。必須ではありませんが、`kubectl vsphere login ...` を実行する前にこの変数を設定することで、（情報が現在の `kubeconfig` ファイルに追加されるのではなく）新しいファイルに書き込まれるようにすることができます。

前提条件

- vCenter Single Sign-On の認証情報を取得します。
- スーパーバイザー クラスタ 制御プレーンの IP アドレスを取得します。
- vSphere 名前空間 の名前を取得します。
- vSphere 名前空間 での編集権限があることを確認します。
- vSphere 向け Kubernetes CLI Tools のダウンロードとインストール。
- 署名を付与する認証局 (CA) を Trust Root としてインストールするか、または証明書を Trust Root として直接追加することにより、Kubernetes 制御プレーンによって提供される証明書がシステムで信頼されることを確認します。vSphere with Tanzu クラスタでのセキュア ログインの構成を参照してください。

手順

1 ログインのコマンド構文とオプションを表示するには、次のコマンドを実行します。

```
kubectl vsphere login --help
```

2 スーパーバイザー クラスタ に接続するには、次のコマンドを実行します。

```
kubectl vsphere login --server=<KUBERNETES-CONTROL-PLANE-IP-ADDRESS> --vsphere-username
<VCENTER-SSO-USER>
```

例：

```
kubectl vsphere login --server=10.92.42.13 --vsphere-username administrator@example.com
```

この操作により、Kubernetes API への認証に使用する JSON Web トークン (JWT) を含む設定ファイルが作成されます。

3 認証するには、ユーザーのパスワードを入力します。

スーパーバイザー クラスタ に接続すると、アクセス可能な設定コンテキストが表示されます。例：

```
You have access to the following contexts:
tanzu-ns-1
tkg-cluster-1
tkg-cluster-2
```

4 アクセスする権限のある設定コンテキストの詳細を表示するには、次の kubectl コマンドを実行します。

```
kubectl config get-contexts
```

CLI に、使用可能な各コンテキストの詳細が表示されます。

5 コンテキストを切り替えるには、次のコマンドを使用します。

```
kubectl config use-context <example-context-name>
```

次のステップ

vCenter Single Sign-On ユーザーとして Tanzu Kubernetes クラスタに接続。

Tanzu Kubernetes クラスタを使用した認証

ロールと目的に応じてさまざまな方法により、Tanzu Kubernetes クラスタ環境で認証を行うことができます。

DevOps エンジニアは、Tanzu Kubernetes クラスタをプロビジョニングして運用します。開発者は、Tanzu Kubernetes クラスタにワークロードをデプロイします。管理者は、Tanzu Kubernetes クラスタのトラブルシューティングが必要になる場合があります。vSphere with Tanzu では、各ロールまたは目的に応じた認証方法を提供しています。

- DevOps エンジニアは、スーパーバイザー クラスタ に接続して Tanzu Kubernetes クラスタをプロビジョニングおよび更新します。認証は、kubectl 向けの vSphere プラグイン および vCenter Single Sign-On の認証情報を使用して行われます。vCenter Single Sign-On ユーザーとして スーパーバイザー クラスタ に接続するを参照してください。
- クラスタ管理者は、プロビジョニングされた Tanzu Kubernetes クラスタに接続して運用と管理を行います。
  - クラスタがデプロイされている vSphere 名前空間 で編集権限を付与されているユーザーには、cluster-admin ロールが割り当てられます。クラスタ管理者は、kubectl 向けの vSphere プラグイン および自分の vCenter Single Sign-On 認証情報を使用して認証します。vCenter Single Sign-On ユーザーとして Tanzu Kubernetes クラスタに接続を参照してください。
  - または、クラスタ管理者は kubernetes-admin ユーザーとして Tanzu Kubernetes クラスタに接続できます。この方法は、vCenter Single Sign-On 認証が使用できない場合に適していることがあります。管理者としての Tanzu Kubernetes クラスタ制御プレーンへの接続を参照してください。
- クラスタ ユーザーまたは開発者は、Tanzu Kubernetes クラスタに接続してポッド、サービス、ロード バランサ、その他のリソースなどのワークロードをデプロイします。
  - クラスタ管理者は、ユーザーまたはグループをデフォルトまたはカスタムのポッド セキュリティ ポリシーにバインドすることにより、開発者にクラスタ アクセス権を付与します。詳細については、『開発者に対する Tanzu Kubernetes クラスタへのアクセス権の付与』を参照してください。
  - バインドされた開発者は、kubectl 向けの vSphere プラグイン および自分の vCenter Single Sign-On 認証情報を使用して Tanzu Kubernetes クラスタで認証を行います。vCenter Single Sign-On ユーザーとして Tanzu Kubernetes クラスタに接続を参照してください。

- トラブルシューティングを行う場合、システム管理者は SSH とプライベート キーを使用することにより、`vmware-system-user` として Tanzu Kubernetes クラスタに接続できます。プライベート キーを使用した、システム ユーザーとしての Tanzu Kubernetes クラスタ ノードへの SSH 接続を参照してください。

vCenter Single Sign-On ユーザーとして Tanzu Kubernetes クラスタに接続

kubectl 向けの vSphere プラグイン を使用して Tanzu Kubernetes クラスタに接続し、vCenter Single Sign-On の認証情報を使用して認証することができます。

Tanzu Kubernetes クラスタにログインすると、kubectl 向けの vSphere プラグイン によってクラスタのコンテキストが生成されます。Kubernetes では、設定コンテキストにクラスタ、名前空間、およびユーザーが含まれます。クラスタのコンテキストは `.kube/config` ファイルで確認できます。このファイルは、通常、`kubeconfig` ファイルと呼ばれます。

注： 既存の `kubeconfig` ファイルがある場合は、そのファイルに各クラスタ コンテキストが追加されます。kubectl 向けの vSphere プラグイン は、kubectl 自体が使用する KUBECONFIG 環境変数に従います。必須ではありませんが、`kubectl vsphere login ...` を実行する前にこの変数を設定することで、（情報が現在の `kubeconfig` ファイルに追加されるのではなく）新しいファイルに書き込まれるようにすることができます。

前提条件

vSphere 管理者から次の情報を取得します。

- vCenter Single Sign-On の認証情報を取得します。
- スーパーバイザー クラスタ 制御プレーンの IP アドレスを取得します。
- vSphere 名前空間 の名前を取得します。
- vSphere 向け Kubernetes CLI Tools のダウンロードとインストール。

手順

1 ログインのコマンド構文とオプションを表示するには、次のコマンドを実行します。

```
kubectl vsphere login --help
```

2 Tanzu Kubernetes クラスタに接続するには、次のコマンドを実行します。

```
kubectl vsphere login --server=SUPERVISOR-CLUSTER-CONTROL-PLANE-IP
--tanzu-kubernetes-cluster-name TANZU-KUBERNETES-CLUSTER-NAME
--tanzu-kubernetes-cluster-namespace SUPERVISOR-NAMESPACE-WHERE-THE-CLUSTER-IS-DEPLOYED
--vsphere-username VCENTER-SSO-USER-NAME
```

例：

```
kubectl vsphere login --server=10.92.42.137
--tanzu-kubernetes-cluster-name tanzu-kubernetes-cluster-01
--tanzu-kubernetes-cluster-namespace tanzu-ns-1
--vsphere-username administrator@example.com
```

この操作により、Kubernetes API への認証に使用する JSON Web トークン (JWT) を含む設定ファイルが作成されます。

3 認証するには、vCenter Single Sign-On パスワードを入力します。

操作が成功すると、「`Logged in successfully`」というメッセージが表示され、クラスタに対して kubectl コマンドを実行できます。このコマンドによって `Error from server (Forbidden)` が返された場合、通常、このエラーは必要な権限がないことを意味します。詳細については、『vCenter Single Sign-On 接続エラーのトラブルシューティング』を参照してください。

4 使用可能なコンテキストのリストを取得するには、次のコマンドを実行します。

```
kubectl config get-contexts
```

このコマンドでは、アクセスできる設定コンテキストがリスト表示されます。`tkg-cluster-01` などのターゲット クラスタの設定コンテキストが表示されます。

5 ターゲット クラスタのコンテキストを使用するには、次のコマンドを実行します。

```
kubectl config use-context CLUSTER-NAME
```

6 クラスタ ノードをリスト表示するには、次のコマンドを実行します。

```
kubectl get nodes
```

このクラスタ内の制御プレーンとワーカー ノードが表示されます。

7 すべてのクラスタ ポッドをリスト表示するには、次のコマンドを実行します。

```
kubectl get pods -A
```

アクセス権のあるすべての Kubernetes 名前空間について、このクラスタ内のすべてのポッドが表示されます。ワークロードをデプロイしていない場合は、デフォルトの名前空間にポッドは表示されません。

管理者としての Tanzu Kubernetes クラスタ制御プレーンへの接続

`kubernetes-admin` ユーザーとして Tanzu Kubernetes クラスタ制御プレーンに接続して、管理者のタスクおよびクラスタの問題のトラブルシューティングを行うことができます。

プロビジョニングされた Tanzu Kubernetes クラスタで有効な `kubeconfig` ファイルは、*`TKGS-CLUSTER-NAME`*`-kubeconfig` という名前のシークレット オブジェクトとして スーパーバイザー クラスタ から入手できます。このシークレットを使用することで、クラスタ制御プレーンに `kubernetes-admin` ユーザーとして接続できます。詳細については、『Tanzu Kubernetes クラスタのシークレットの取得』を参照してください。

手順

1 スーパーバイザー クラスタ に接続します。vCenter Single Sign-On ユーザーとして スーパーバイザー クラスタ に接続するを参照してください。

2 ターゲット Tanzu Kubernetes クラスタがプロビジョニングされている vSphere 名前空間 にコンテキストを切り替えます。

```
kubectl config use-context VSPHERE-NAMESPACE
```

3 名前空間内のシークレット オブジェクトを表示します。

```
kubectl get secrets
```

シークレットには、*TKGS-CLUSTER-NAME*-kubeconfig という名前が付いています。

```
kubectl config use-context tkgs-cluster-ns
Switched to context "tkgs-cluster-ns".
ubuntu@ubuntu:~$ kubectl get secrets
NAME                                TYPE                                  DATA   AGE
...
tkgs-cluster-1-kubeconfig           Opaque                                1      23h
...
```

4 次のコマンドを実行してシークレットをデコードします。

シークレットは Base64 でエンコードされています。デコードするには、Linux の場合は `base64 --decode`（または `base64 -d`）、MacOS の場合は `base64 --Decode`（または `base64 -D`）、Windows の場合はオンライン ツールを使用します。

```
kubectl get secret TKGS-CLUSTER-NAME-kubeconfig -o jsonpath='{.data.value}' | base64 -d >
tkgs-cluster-kubeconfig-admin
```

このコマンドはシークレットをデコードし、`tkgs-cluster-kubeconfig-admin` という名前のローカルファイルに書き込みます。`cat` コマンドを使用して、ファイルの内容を確認します。

5 デコードされた `tkgs-cluster-kubeconfig-admin` ファイルを使用し、Kuberentes 管理者として Tanzu Kubernetes クラスタに接続します。

これには、以下の 2 つの方法があります。

オプション	説明
--kubeconfig <path\to\kubeconfig>	`--kubeconfig` フラグとローカルの kubeconfig ファイルへのパスを使用します。たとえば、kubeconfig ファイルが同じディレクトリにある場合は次のコマンドを実行します：`kubectl --kubeconfig tkgs-cluster-kubeconfig-admin get nodes`
KUBECONFIG	デコードされた kubeconfig ファイルを参照するように KUBECONFIG 環境変数を設定し、`kubectl get nodes` などの kubectl を実行します。

クラスタ内にノードが表示されます。例：

```
kubectl --kubeconfig tkgs-cluster-kubeconfig-admin get nodes
NAME                                          STATUS   ROLES    AGE   VERSION
tkgs-cluster-1-control-plane-4ncm4            Ready    master   23h   v1.18.5+vmware.1
tkgs-cluster-1-control-plane-jj9gq            Ready    master   23h   v1.18.5+vmware.1
tkgs-cluster-1-control-plane-r4hm6            Ready    master   23h   v1.18.5+vmware.1
tkgs-cluster-1-workers-6njk7-84dd7f48c6-nz2n8   Ready    <none>   23h   v1.18.5+vmware.1
tkgs-cluster-1-workers-6njk7-84dd7f48c6-rk9pk   Ready    <none>   23h   v1.18.5+vmware.1
tkgs-cluster-1-workers-6njk7-84dd7f48c6-zzngh   Ready    <none>   23h   v1.18.5+vmware.1
```

プライベート キーを使用した、システム ユーザーとしての Tanzu Kubernetes クラスタ ノードへの SSH 接続

プライベート キーを使用して、`vmware-system-user` として Tanzu Kubernetes クラスタ ノードに SSH で接続できます。

`vmware-system-user` ユーザーとして任意の Tanzu Kubernetes クラスタ ノードに SSH で接続できます。SSH プライベート キーを含むシークレットには、`CLUSTER-NAME-ssh` という名前が付いています。詳細については、『Tanzu Kubernetes クラスタのシークレットの取得』を参照してください。

プライベート キーを使用して SSH 経由で Tanzu Kubernetes クラスタ ノードに接続するには、スーパーバイザー クラスタ にジャンプ ボックス vSphere ポッド を作成します。

前提条件

このタスクでは、SSH 接続のジャンプ ホストとして vSphere ポッド をプロビジョニングします。vSphere ポッド では、スーパーバイザー クラスタ 用に NSX-T ネットワークが必要です。スーパーバイザー クラスタ 用に Distributed Switch ネットワークを使用している場合は、パスワードを使用した、システム ユーザーとしての Tanzu Kubernetes クラスタ ノードへの SSH 接続に記載された方法を代わりに使用します。

手順

1 スーパーバイザー クラスタ に接続します。

vCenter Single Sign-On ユーザーとして スーパーバイザー クラスタ に接続するを参照してください。

2 **NAMESPACE** という名前の環境変数を作成し、値に、ターゲット Tanzu Kubernetes クラスタがプロビジョニングされている vSphere 名前空間 の名前を指定します。

```
export NAMESPACE=VSPHERE-NAMESPACE
```

3 Tanzu Kubernetes クラスタがプロビジョニングされている vSphere 名前空間 にコンテキストを切り替えます。

```
kubectl config use-context $NAMESPACE
```

4 *TKGS-CLUSTER-NAME*-ssh シークレット オブジェクトを表示します。

```
kubectl get secrets
```

5 次の jumpbox.yaml を使用して、vSphere ポッド を作成します。

namespace 値の YOUR-NAMESPACE を、ターゲット クラスタがプロビジョニングされている vSphere 名前空間 に置き換えます。secretName 値の YOUR-CLUSTER-NAME-ssh をターゲット クラスタの名前に置き換えます。

```
apiVersion: v1
kind: Pod
metadata:
  name: jumpbox
  namespace: YOUR-NAMESPACE     #REPLACE
spec:
  containers:
  - image: "photon:3.0"
    name: jumpbox
    command: [ "/bin/bash", "-c", "--" ]
    args: [ "yum install -y openssh-server; mkdir /root/.ssh; cp /root/ssh/ssh-privatekey /
root/.ssh/id_rsa; chmod 600 /root/.ssh/id_rsa; while true; do sleep 30; done;" ]
    volumeMounts:
      - mountPath: "/root/ssh"
        name: ssh-key
        readOnly: true
    resources:
      requests:
        memory: 2Gi
  volumes:
    - name: ssh-key
      secret:
        secretName: YOUR-CLUSTER-NAME-ssh     #REPLACE
```

6 jumpbox.yaml の仕様を適用して、ポッドをデプロイします。

```
kubectl apply -f jumpbox.yaml
```

```
pod/jumpbox created
```

7 ポッドが実行されていることを確認します。

```
kubectl get pods
```

```
NAME      READY   STATUS    RESTARTS   AGE
jumpbox   1/1     Running   0          3h9m
```

注： vSphere 名前空間 内の vCenter Server のジャンプボックス ポッドも確認する必要があります。

8 次の一連のコマンドを実行して、ターゲット クラスタ ノードの IP アドレスを持つ環境変数を作成します。

a ターゲット仮想マシンの名前を取得します。

```
kubectl get virtualmachines
```

b 値がターゲット ノードの名前である VMNAME 環境変数を作成します。

```
export VMNAME=NAME-OF-THE-VIRTUAL-MACHINE
```

c 値がターゲット ノード仮想マシンの IP アドレスである VMIP 環境変数を作成します。

```
export VMIP=$(kubectl -n $NAMESPACE get virtualmachine/$VMNAME -o
jsonpath='{.status.vmIp}')
```

9 次のコマンドを実行して、ジャンプ ボックス ポッドを使用してクラスタ ノードに SSH 接続します。

```
kubectl exec -it jumpbox  /usr/bin/ssh vmware-system-user@$VMIP
```

重要： コンテナを作成してソフトウェアをインストールするには、約 60 秒かかります。「error executing command in container: container_linux.go:370: starting container process caused: exec: "/usr/bin/ssh": stat /usr/bin/ssh: no such file or directory」というエラーが表示された場合は、数秒後に再実行してください。

10 **yes** を入力して、ホストの信頼性を確認します。

```
The authenticity of host '10.249.0.999 (10.249.0.999)' can't be established.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.249.0.999' (ECDSA) to the list of known hosts.
Welcome to Photon 3.0
```

11 ターゲット ノードに vmware-system-user としてログインしていることを確認します。

たとえば、次の出力は、システム ユーザーとして制御プレーン ノードにログインしていることを示しています。

```
vmware-system-user@tkgs-cluster-1-control-plane-66tbr [ ~ ]$
```

12 ノードで必要な操作を実行します。

注目： kubelet の再起動などの特定の処理をノードで実行するには、sudo または sudo su を使用しなければならない場合があります。

13 操作完了後、**exit** を入力し、vSphere ポッド の SSH セッションからログアウトします。

14 ポッドを削除するには、kubectl delete pod jumpbox コマンドを実行します。

注意： セキュリティを確保するには、作業の完了後にジャンプボックス ポッドを削除することを検討してください。必要に応じて、後でポッドを再作成できます。

パスワードを使用した、システム ユーザーとしての Tanzu Kubernetes クラスタ ノードへの SSH 接続

パスワードを使用して、`vmware-system-user` として Tanzu Kubernetes クラスタ ノードに SSH 接続できます。

パスワードを使用して、`vmware-system-user` ユーザーとしてクラスタ ノードに接続できます。パスワードは、`CLUSTER-NAME-ssh-password` という名前のシークレットとして保存されます。パスワードは `.data.ssh-passwordkey` に base64 でエンコードされています。SSH セッションを介してパスワードを指定できます。このシークレットの詳細については、Tanzu Kubernetes クラスタのシークレットの取得を参照してください。

前提条件

SSH 接続を適切なワークロード ネットワークにルーティングするには、[ワークロード管理] が有効な vSphere 環境に Linux ジャンプ ホスト仮想マシンをデプロイします。Linux ジャンプ ホスト仮想マシンの作成を参照してください。

注： クラスタ ノードへの接続に SSH の使用を検討中で、vSphere ポッド がサポートされない Distributed Switch ネットワークを使用している場合、これは難しい要件になります。プライベート キーの代わりにパスワードを使用して SSH 接続すると、NSX-T ネットワークでもこの方法を使用できます。

手順

1. ジャンプ ホスト仮想マシンの IP アドレス、ユーザー名、およびパスワードを取得します。Linux ジャンプ ホスト仮想マシンの作成を参照してください。
2. スーパーバイザー クラスタ に接続します。

   vCenter Single Sign-On ユーザーとして スーパーバイザー クラスタ に接続する。
3. ターゲット Tanzu Kubernetes クラスタがプロビジョニングされている vSphere 名前空間 にコンテキストを切り替えます。

   ```
   kubectl config use-context VSPHERE-NAMESPACE
   ```
4. ターゲット クラスタ ノードの IP アドレスを取得します。

   ノードを一覧表示します。

   ```
   kubectl get virtualmachines
   ```

   ターゲット ノードの IP アドレスを取得するノードを記述します。

   ```
   kubectl describe virtualmachines
   ```
5. `TKGS-CLUSTER-NAME-ssh-password` シークレットを表示します。

   ```
   kubectl get secrets
   ```

6 ターゲット クラスタの ssh-passwordkey を取得します。

```
kubectl get secrets TKGS-CLUSTER-NAME-ssh-password -o yaml
```

たとえば、ssh-passwordkey が返されます。

```
apiVersion: v1
data:
  ssh-passwordkey: RU1pQllLTC9TRjVFV0RBcCtmd1zwOTROeURYSWNGeXNReXJhaXRBUllYaz0=
```

7 ssh-passwordkey をデコードします。

シークレットは Base64 でエンコードされています。デコードするには、Linux の場合は `base64 --decode`（または `base64 -d`）、MacOS の場合は `base64 --Decode`（または `base64 -D`）、Windows の場合はオンライン ツールを使用します。

```
echo <ssh-passwordkey> | base64 --decode
```

8 ターゲット クラスタ ノードに `vmware-system-user` として SSH 接続します。

```
ssh vmware-system-user@TKGS-CLUSTER-NODE-IP-ADDRESS
```

9 デコードしたパスワードを使用してログインします。

Linux ジャンプ ホスト仮想マシンの作成

パスワードを使用して Tanzu Kubernetes クラスタ ノードに SSH 接続するには、まず、SSH トンネル接続のためにワークワークロード ネットワークと管理またはフロントエンド ネットワークに接続するジャンプ ボックス仮想マシンを作成します。

Linux ジャンプ ホスト仮想マシンの作成

Linux ジャンプ ボックス仮想マシンを作成するには、次の手順に従います。これを実現するには、さまざまな方法があります。ここに示すのは、そのうちの 1 つです。この手順では、https://github.com/vmware/photon/wiki/Downloading-Photon-OS からダウンロードできる PhotonOS を使用します。

1 vSphere Client を使用して vCenter Server にログインします。

2 新しい仮想マシンを作成します。

3 Linux ゲスト OS を選択します。この例では、VMware Photon OS（64 ビット）を選択します。

4 OS をインストールします。これを行うには、ISO ファイルをダウンロードして仮想マシンに適用し、起動します。

5 ワークロード ネットワーク上の IP アドレスを使用して仮想マシンを構成します。

6 2 つ目の仮想 NIC を仮想マシンに追加し、フロントエンド ネットワークに割り当てます。

7 OS の構成を完了し、再起動後に仮想マシンをパワーオンします。

8 仮想マシンの vSphere コンソールに root ユーザーとしてログインします。

9 新しい NIC のネットワーク インターフェイスを作成して、フロントエンド ネットワーク上の IP アドレスを割り当てます。

```
ifconfig eth1 IP-ADDRESS netmask NETMASK up
```

注： この方法による効果は、再起動すると無効になります。

10 このインターフェイスからゲートウェイと DNS サーバに ping を実行できることを確認します。

11 仮想マシンの vSphere コンソールで、証明書を使用して SSH ユーザーを設定します。ネストされたシェルを作成して、機能することを確認します。

12 SSH ユーザーとしてフロントエンド ネットワークからジャンプ ボックスに SSH 接続して、正常に動作することを確認します。

13 sshpass を仮想マシンにインストールします（これにより、パスワードを使用して SSH 接続からログインできるようになります）。PhotonOS の場合、コマンドは次のとおりです。

```
tdnf install -y sshpass
```

14 ssh がパスワードなしで動作するように、`~/.ssh/authorized_keys` ファイルにクライアントのパブリック キーを追加し、`sshd` プロセスを再起動します。

- パブリック キーを取得します（例：`cat ~/.ssh/id_rsa.pub`）。
- ジャンプホスト仮想マシンにアクセスします。
- SSH ディレクトリがない場合は、次のコマンドで作成します：`mkdir -p ~/.ssh`。
- `authorized_keys` ファイルにパブリック キーを追加します。`echo ssh-rsa AAAA.... >> ~/.ssh/authorized_keysssh-rsa AAAA....` を、`cat ~/.ssh/id_rsa.pub` コマンドで出力されたパブリック キー文字列全体に置き換えます。
- `~/.ssh` ディレクトリと `authorized_keys` ファイルに適切な権限が設定されていることを確認します（例：`chmod -R go= ~/.ssh`）。

開発者に対する Tanzu Kubernetes クラスタへのアクセス権の付与

開発者は、Kubernetes が対象としているユーザーです。Tanzu Kubernetes クラスタをプロビジョニングすると、vCenter Single Sign-On 認証を使用して開発者にアクセス権を付与することができます。

開発者向けの認証

クラスタ管理者は、開発者などの他のユーザーにクラスタへのアクセス権を付与できます。開発者は、ユーザー アカウントを使用して直接、またはサービス アカウントを使用して間接的に、クラスタにポッドをデプロイできます。詳細については、『Tanzu Kubernetes クラスタでのポッド セキュリティ ポリシーの使用』を参照してください。

- ユーザー アカウント認証の場合、Tanzu Kubernetes クラスタは vCenter Single Sign-On のユーザーとグループをサポートします。ユーザーまたはグループは、vCenter Server のローカルであるか、サポートされているディレクトリ サーバから同期されます。

- サービス アカウント認証の場合は、サービス トークンを使用できます。詳細については、Kubernetes のドキュメントを参照してください。

クラスタへの開発者ユーザーの追加

開発者にクラスタ アクセスを許可するには：

1 ユーザーまたはグループの Role または ClusterRole を定義し、クラスタに適用します。詳細については、Kubernetes のドキュメントを参照してください。

2 ユーザーまたはグループの RoleBinding または ClusterRoleBinding を作成し、クラスタに適用します。次の例を参照してください。

RoleBinding の例

vCenter Single Sign-On のユーザーまたはグループにアクセス権を付与するには、RoleBinding のサブジェクトに `name` パラメータの値として次のいずれかを含める必要があります。

表 9-1. サポートされているユーザーおよびグループのフィールド

フィールド	説明
`sso:USER-NAME@DOMAIN`	たとえば、`sso:joe@vsphere.local` などのローカル ユーザー名です。
`sso:GROUP-NAME@DOMAIN`	たとえば、`sso:devs@ldap.example.com` などの vCenter Server と統合されたディレクトリ サーバのグループ名です。

次の RoleBinding の例では、Joe という vCenter Single Sign-On ローカル ユーザーが、`edit` というデフォルトの ClusterRole にバインドされます。このロールにより、名前空間（この例では `default` 名前空間）内のほとんどのオブジェクトに対する読み取り/書き込みアクセスが許可されます。

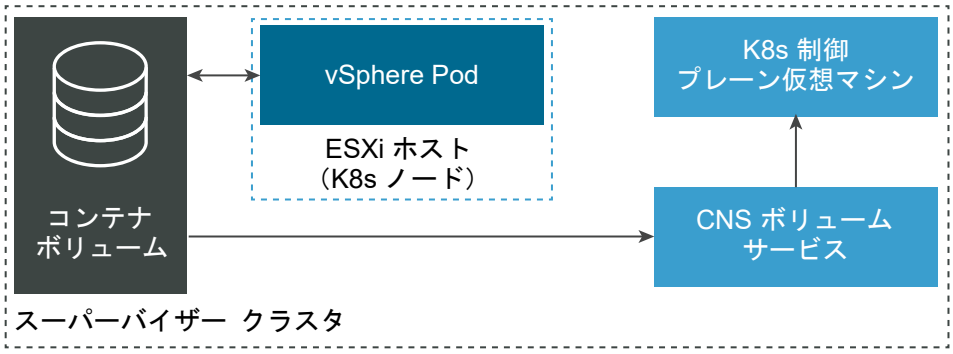
```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: rolebinding-cluster-user-joe
  namespace: default
roleRef:
  kind: ClusterRole
  name: edit                              #Default ClusterRole
  apiGroup: rbac.authorization.k8s.io
subjects:
- kind: User
  name: sso:joe@vsphere.local             #sso:<username>@<domain>
  apiGroup: rbac.authorization.k8s.io
```

vSphere with Tanzu でのパーシステント ストレージの使用

10

特定の Kubernetes ワークロードでデータを永続的に保存するには、パーシステント ストレージが必要です。Kubernetes ワークロード用にパーシステント ストレージをプロビジョニングするため、vSphere with Tanzu は、パーシステント ボリュームを管理する vCenter Server コンポーネントであるクラウド ネイティブ ストレージ (CNS) と統合します。

パーシステント ストレージは、vSphere ポッド、Tanzu Kubernetes クラスタ、および仮想マシンで使用します。次の例では、vSphere ポッド でのパーシステント ストレージの使用方法を示します。



vSphere with Tanzu とパーシステント ストレージがどのように連携するかを理解するには、以下の重要な概念を把握しておく必要があります。

パーシステント ボリューム

パーシステント ストレージを提供するために、Kubernetes では、その状態とデータを保持できるパーシステント ボリュームを使用します。パーシステント ボリュームがポッドによってマウントされている場合は、ポッドを削除または再構成しても、パーシステント ボリュームは引き続き存在します。vSphere with Tanzu 環境では、パーシステント ボリューム オブジェクトは、データストアの最初のクラス ディスクによってバッキングされます。

vSphere with Tanzu は、ボリュームをポッド単体にマウント可能な ReadWriteOnce モードでボリュームの動的/静的プロビジョニングをサポートします。

vSphere 7.0 Update 3 リリース以降、vSphere with Tanzu は Tanzu Kubernetes クラスタ内のパーシステント ボリュームに対しても ReadWriteMany モードをサポートしています。ReadWriteMany サポートにより、1 つのクラスタ内で実行されている複数のポッドまたはアプリケーションによって 1 つのボリュームを同時にマウントできます。vSphere with Tanzu は、ReadWriteMany タイプのパーシステント ボリュームに vSAN ファイル共有を使用します。詳細については、vSphere with Tanzu での ReadWriteMany パーシステント ボリュームの作成を参照してください。

動的および静的プロビジョニング

動的ボリューム プロビジョニングが有効な場合は、ストレージを事前にプロビジョニングしておく必要はありません。パーシステント ボリュームは必要に応じて作成できます。DevOps エンジニアは、名前空間で使用可能なストレージ クラスを参照するパーシステント ボリュームの要求を発行します。vSphere with Tanzu は、対応するパーシステント ボリュームとバッキング仮想ディスクを自動的にプロビジョニングします。

スーパーバイザー クラスタ と Tanzu Kubernetes クラスタは両方とも、動的プロビジョニングをサポートしています。

パーシステント ボリュームの動的作成例については、ステートフル アプリケーションの動的パーシステント ボリュームのプロビジョニングを参照してください。

静的プロビジョニングが有効な場合は、既存のストレージ オブジェクトを使用して、クラスタで使用できるようにすることができます。

通常、DevOps エンジニアは既存のストレージ オブジェクトの詳細、サポートされている構成、およびこのオブジェクトを再利用するためのマウント オプションについて把握している必要があります。

静的パーシステント ボリュームのプロビジョニング例については、Tanzu Kubernetes クラスタでの静的パーシステント ボリュームのプロビジョニングを参照してください。

最初のクラス ディスク

vSphere with Tanzu は、最初のクラス ディスク (FCD) タイプの仮想ディスクを使用して、パーシステント ボリュームをバッキングします。改善した仮想ディスクとも呼ばれる最初のクラス ディスクは、仮想マシンに関連付けられていない名前付き仮想ディスクです。

最初のクラス ディスクは UUID で識別されます。この UUID は、グローバルで一意な FCD のプライマリ識別子です。UUID は、FCD が再配置された場合やそのスナップショットが作成された場合でも有効なままです。

パーシステント ボリュームの要求

DevOps エンジニアは、パーシステント ストレージ リソースを要求するためにパーシステント ボリュームの要求を作成します。この要求により、パーシステント ボリューム オブジェクトとそれに対応する仮想ディスクがプロビジョニングされます。vSphere Client では、パーシステント ボリュームの要求は、vSphere 管理者が監視できる FCD 仮想ディスクとして表されます。

要求はパーシステント ボリュームにバインドされています。ワークロードは、この要求を使用してパーシステント ボリュームをマウントし、ストレージにアクセスできます。

DevOps エンジニアがこの要求を削除すると、対応するパーシステント ボリューム オブジェクトおよびプロビジョニングされた仮想ディスクも削除されます。

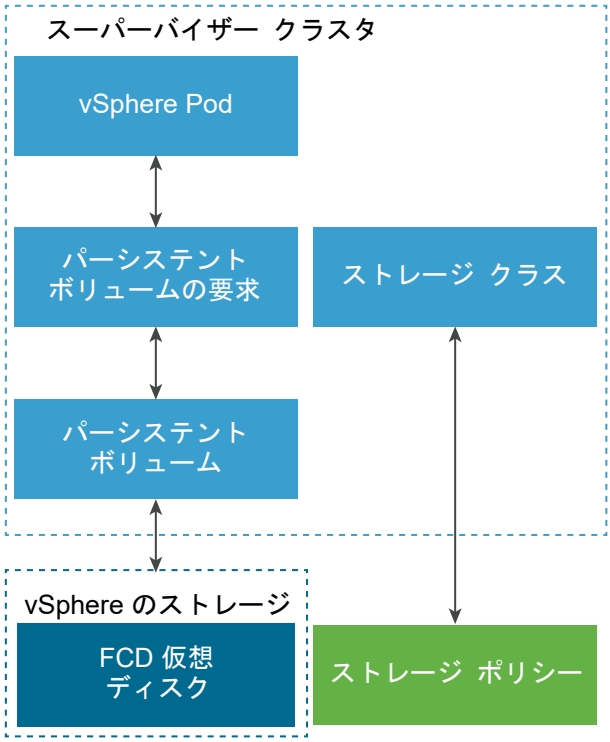
ストレージ クラス

Kubernetes では、ストレージ クラスを使用して、パーシステント ボリュームをバッキングするストレージの要件を記述します。DevOps エンジニアは、パーシステント ボリュームの要求指定に特定のストレージ クラスを含めて、そのクラスが示すタイプのストレージを要求できます。

パーシステント ストレージのワークフロー

vSphere with Tanzu にパーシステント ストレージをプロビジョニングするためのワークフローには、一般に、次の一連のアクションが含まれます。

手順	操作	説明
1	vSphere 管理者が DevOps チームにパーシステント ストレージ リソースを提供します。	vSphere 管理者は、さまざまなストレージ要件およびサービス クラスを記述する仮想マシン ストレージ ポリシーを作成します。その後、ストレージ ポリシーを vSphere 名前空間 に割り当てることができます。
2	vSphere with Tanzu により、vSphere 名前空間 に割り当てられたストレージ ポリシーと一致するストレージ クラスが作成されます。	ストレージ クラスは Kubernetes 環境に自動的に表示され、DevOps チームが使用できるようになります。vSphere 管理者が vSphere 名前空間 に複数のストレージ ポリシーを割り当てると、ストレージ ポリシーごとに個別のストレージ クラスが作成されます。 Tanzu Kubernetes Grid サービス を使用して Tanzu Kubernetes クラスタをプロビジョニングした場合、各 Tanzu Kubernetes クラスタには、このクラスタがプロビジョニングされている vSphere 名前空間 からストレージ クラスが継承されます。
3	DevOps エンジニアがストレージ クラスを使用して、ワークロードのパーシステント ストレージ リソースを要求します。	この要求は、特定のストレージ クラスを参照するパーシステント ボリュームの要求の形式で行います。
4	vSphere with Tanzu により、ワークロードに対して、パーシステント ボリューム オブジェクトとそれに対応するパーシステント仮想ディスクが作成されます。	vSphere with Tanzu は、元のストレージ ポリシーとそれに一致するストレージ クラスで指定された要件を満たす仮想ディスクをデータストアに配置します。仮想ディスクはワークロードを使用してマウントできます。
5	vSphere 管理者が vSphere with Tanzu 環境のパーシステント ボリュームを監視します。	vSphere 管理者は、vSphere Client を使用して、パーシステント ボリュームとそのバッキング仮想ディスクを監視します。また、パーシステント ボリュームのストレージ コンプライアンスと健全性ステータスを監視することもできます。



vSphere with Tanzu のパーシステント ストレージについては、こちらのビデオをご覧ください。

(vSphere with Kubernetes のパーシステント ストレージ)

vSphere 管理者のストレージ管理タスク

通常、vSphere with Tanzu のパーシステント ストレージ管理タスクには次のようなものがあります。vSphere 管理者は、vSphere Client を使用してこれらのタスクを実行します。

- 仮想マシン ストレージ ポリシーのライフサイクル操作を実行します。

  スーパーバイザー クラスタ を有効にして名前空間を構成する前に、パーシステント ストレージのストレージ ポリシーを作成します。ストレージ ポリシーは、DevOps エンジニアから伝えられたストレージ要件に基づきます。vSphere with Tanzu のストレージ ポリシーの作成 を参照してください。

  注： 対応するストレージ クラスを含むパーシステント ボリュームの要求が名前空間で実行されている場合は、vCenter Server または vSphere 名前空間 からストレージ ポリシーを削除しないでください。この推奨事項は、Tanzu Kubernetes クラスタにも適用されます。

- ストレージ ポリシーを名前空間に割り当てて、ストレージ制限を設定することで、ストレージ リソースを DevOps エンジニアに提供します。ストレージ ポリシー割り当ての変更については、名前空間のストレージ設定の変更を参照してください。制限の設定については、vSphere 名前空間 での Kubernetes オブジェクトの制限の構成を参照してください。

- vSphere Client で、Kubernetes オブジェクトとそのストレージ ポリシーのコンプライアンスを監視します。vSphere Client のパーシステント ボリュームの監視を参照してください。

DevOps エンジニアのストレージ管理タスク

通常、DevOps エンジニアは `kubectl` を使用して次のストレージ タスクを実行します。

- ストレージ クラスを管理する。vSphere 名前空間 または Tanzu Kubernetes クラスタでのストレージ クラスの表示を参照してください。
- ステートフル アプリケーションをデプロイして管理する。ステートフル アプリケーションの動的パーシステント ボリュームのプロビジョニングを参照してください。
- パーシステント ボリュームのライフサイクル操作を実行する。Tanzu Kubernetes パーシステント ボリュームの要求例を参照してください。

この章には、次のトピックが含まれています。

- vSphere with Tanzu と vSphere ストレージの統合方法
- vSphere with Tanzu で vSphere CNS-CSI、準仮想化 CSI によってサポートされる機能
- vSphere with Tanzu のストレージ権限
- vSphere with Tanzu のストレージ ポリシーの作成
- スーパーバイザー クラスタ のストレージ設定の変更
- 名前空間のストレージ設定の変更
- vSphere 名前空間 または Tanzu Kubernetes クラスタでのストレージ クラスの表示
- ステートフル アプリケーションの動的パーシステント ボリュームのプロビジョニング
- Tanzu Kubernetes クラスタでの静的パーシステント ボリュームのプロビジョニング
- vSphere with Tanzu での ReadWriteMany パーシステント ボリュームの作成
- vSphere with Tanzu でのボリュームの拡張
- vSphere Client のパーシステント ボリュームの監視
- vSphere 名前空間 または Tanzu Kubernetes クラスタでのボリュームの健全性の監視
- 最新のステートフル サービスでの vSAN データ パーシステンス プラットフォームの使用

vSphere with Tanzu と vSphere ストレージの統合方法

vSphere with Tanzu はいくつかのコンポーネントを使用して、vSphere ストレージと統合します。

vCenter Server 上のクラウド ネイティブ ストレージ (CNS)

CNS コンポーネントは、vCenter Server に配置されます。これは、パーシステント ボリュームのプロビジョニングとライフサイクルの操作を実装する vCenter Server 管理の拡張機能です。

コンテナ ボリュームをプロビジョニングするときに、コンポーネントは vSphere の最初のクラス ディスク機能と通信して、ボリュームをバッキングする仮想ディスクを作成します。また、CNS サーバ コンポーネントは、ストレージ ポリシーベースの管理と通信して、ディスクに必要なサービス レベルを確保します。

CNS は、vSphere 管理者が vCenter Server を介してパーシステント ボリュームとそのバッキング ストレージ オブジェクトを管理および監視するクエリ処理も実行します。

最初のクラス ディスク (FCD)

強化された仮想ディスクとも呼ばれます。これは、仮想マシンと関連付けられていない名前付き仮想ディスクです。これらのディスクは、VMFS、NFS、または vSAN データストア、およびバック ReadWriteOnce パーシステント ボリュームに配置されます。

FCD テクノロジーは、仮想マシンまたはポッドのライフサイクル外でパーシステント ボリュームに関連するライフサイクル操作を実行します。

FCD を使用する場合は、次の点に注意してください。

- FCD は NFS 4.x プロトコルをサポートしません。代わりに、NFS 3 を使用してください。
- vCenter Server は、同じ FCD での操作をシリアル化しません。したがって、アプリケーションは同じ FCD で複数の操作を同時に実行できません。複数のスレッドからクローン作成、再配置、削除、取得などの操作を同時に実行すると、予期しない結果になります。問題を回避するには、アプリケーションが同じ FCD で操作を実行するときに、順番に実行する必要があります。
- FCD は管理対象オブジェクトではなく、単一 FCD への複数の書き込みから保護するグローバル ロックをサポートしていません。したがって、複数の vCenter Server インスタンスで同じ FCD を管理することはできません。FCD で複数の vCenter Server インスタンスを使用する必要がある場合は、次の方法を使用できます。
  - 複数の vCenter Server インスタンスで複数のデータストアを管理できます。
  - 複数の vCenter Server インスタンスが同じ FCD で動作することはありません。

ストレージ ポリシー ベースの管理

ストレージ ポリシー ベースの管理は、ストレージ ポリシーに記述されているストレージ要件に従って、パーシステント ボリュームとそのバッキング仮想ディスクのプロビジョニングをサポートする、vCenter Server サービスです。プロビジョニング後、サービスは、ストレージ ポリシー特性に対するボリュームのコンプライアンスを監視します。ストレージ ポリシー ベースの管理の詳細については、「ストレージ ポリシー ベースの管理」を参照してください。

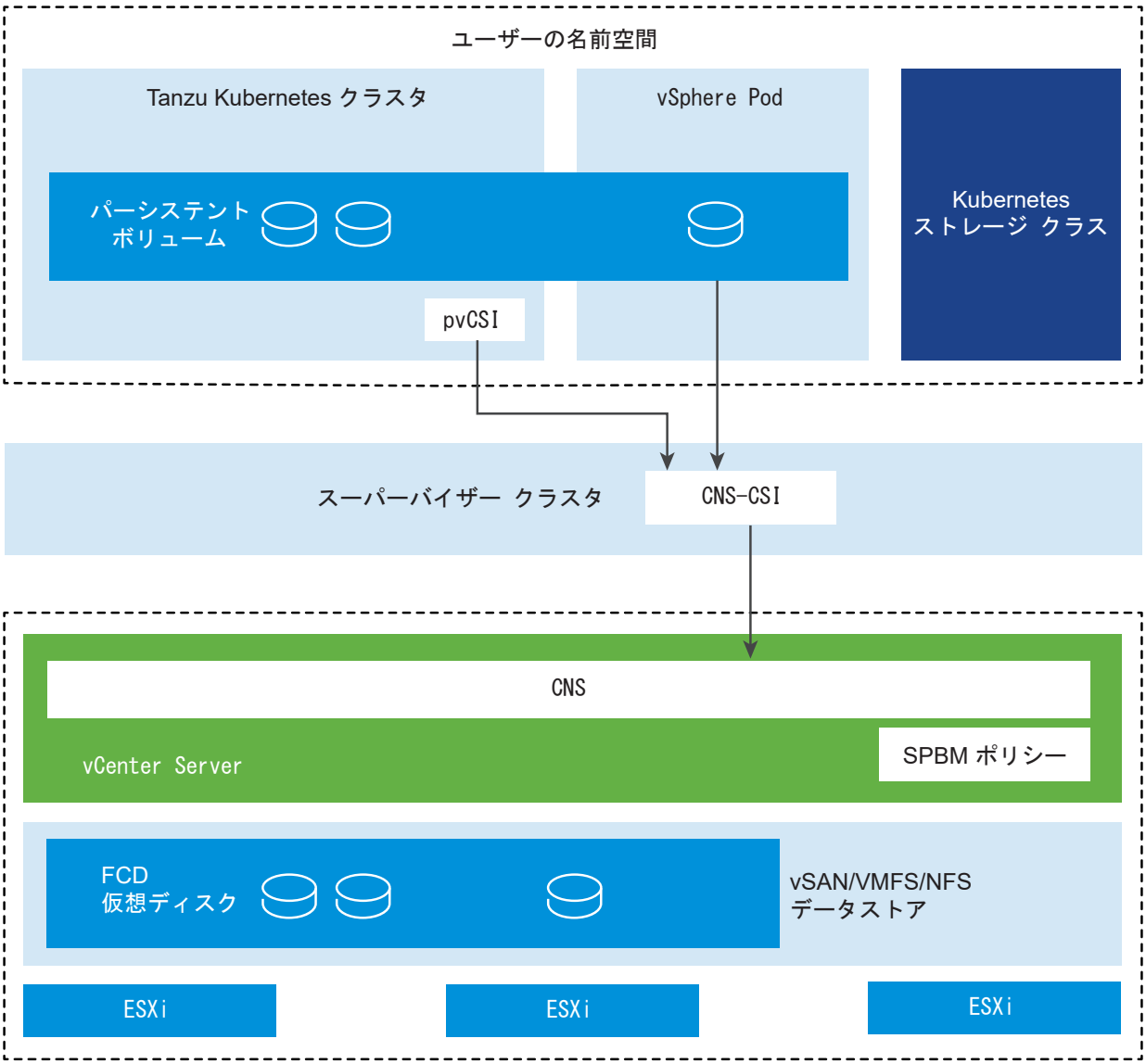
vSphere CNS-CSI

vSphere CNS-CSI コンポーネントは、コンテナ ストレージ インターフェイス (CSI) 仕様に準拠しています。これは、Kubernetes のようなコンテナ オーケストレータによってパーシステント ストレージのプロビジョニングに使用されるインターフェイスを提供するために設計された業界標準です。CNS-CSI は スーパーバイザー クラスタ で実行され、vSphere ストレージを vSphere 名前空間 の Kubernetes 環境に接続します。vSphere CNS-CSI は、vSphere ポッド と名前空間の Tanzu Kubernetes クラスタで実行されているポッドから送信される、すべてのストレージ プロビジョニング要求について、CNS 制御プレーンと直接通信します。

準仮想化 CSI (pvCSI)

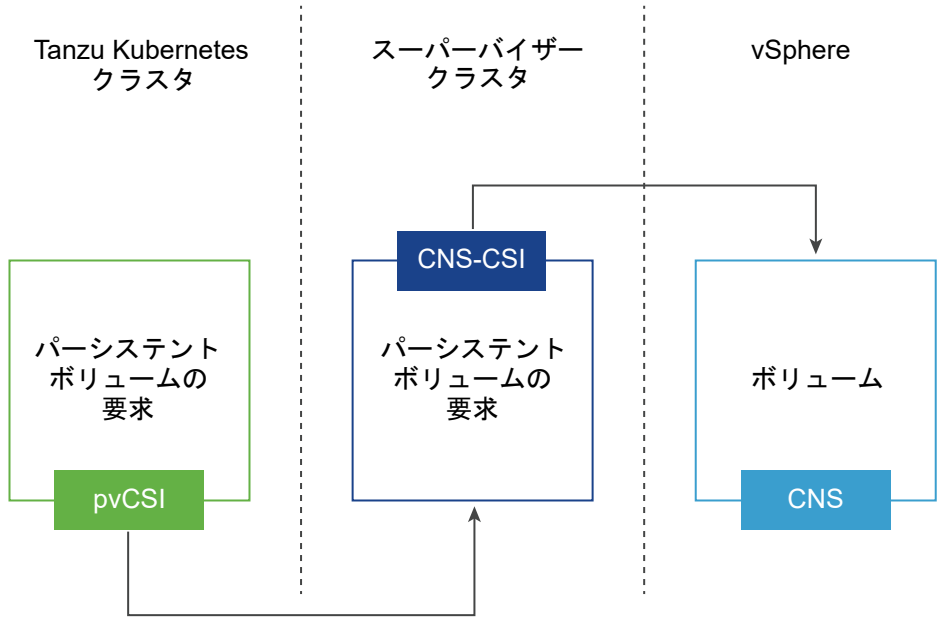
pvCSI は Tanzu Kubernetes クラスタ用に変更された vSphere CNS-CSI ドライバのバージョンです。pvCSI は Tanzu Kubernetes クラスタに配置され、Tanzu Kubernetes クラスタから送信されるすべてのストレージ関連の要求に対処します。要求は CNS-CSI に配信され、vCenter Server の CNS に伝達されます。その結果、pvCSI は CNS コンポーネントとの直接通信は行わず、すべてのストレージ プロビジョニング操作に CNS-CSI を使用します。

CNS-CSI とは異なり、pvCSI はインフラストラクチャの認証情報を必要としません。vSphere 名前空間 のサービス アカウントを使用して構成されます。



次の図は、DevOps エンジニアが Tanzu Kubernetes クラスタ内でストレージ関連の操作（たとえば、パーシステント ボリューム要求 (PVC) の作成）を実行するときに、さまざまなコンポーネントがどのように相互作用するかを示しています。

DevOps エンジニアは、Tanzu Kubernetes クラスタでコマンド ラインを使用して PVC を作成します。このアクションにより、対応する PVC がスーパーバイザー クラスタに生成され、CNS-CSI がトリガされます。CNS-CSI は、CNS ボリューム作成 API を呼び出します。



ボリュームの作成が正常に完了すると、操作はスーパーバイザー クラスタを介して元の Tanzu Kubernetes クラスタに伝達されます。この伝達の結果、ユーザーは、スーパーバイザー クラスタでバインド状態のパーシステント ボリュームとパーシステント ボリュームの要求を確認できます。また、バインド状態のパーシステント ボリュームとパーシステント ボリュームの要求は Tanzu Kubernetes クラスタでも確認できます。

vSphere with Tanzu で vSphere CNS-CSI、準仮想化 CSI によってサポートされる機能

スーパーバイザー クラスタ で実行される vSphere CNS-CSI ドライバと、Tanzu Kubernetes クラスタ用に変更された CSI バージョンである pvCSI ドライバは、複数の vSphere と Kubernetes ストレージ機能をサポートします。ただし、特定の制限が適用されます。

サポートされている機能	vSphere CNS-CSI と スーパーバイザー クラスタ の併用	pvCSI と Tanzu Kubernetes クラスタの併用
vSphere Client での CNS のサポート	はい	はい
vSphere Client でのオブジェクト健全性の向上	はい（vSAN のみ）	はい（vSAN のみ）
動的ブロック パーシステント ボリューム（ReadWriteOnce アクセス モード）	はい	はい
動的ファイル パーシステント ボリューム（ReadWriteMany アクセス モード）	なし	はい（vSAN ファイル サービスを使用）
vSphere データストア	VMFS/NFS/vSAN/vVols	VMFS/NFS/vSAN/vVols
静的パーシステント ボリューム	はい	はい
暗号化	なし	なし
オフライン ボリュームの拡張	はい	はい
オンライン ボリュームの拡張	はい	はい

サポートされている機能	vSphere CNS-CSI と スーパーバイザー クラスタ の併用	pvCSI と Tanzu Kubernetes クラスタの併用
ボリューム トポロジとゾーン	なし	なし
Kubernetes の複数の制御プレーン インスタンス	はい	はい
WaitForFirstConsumer	なし	なし
VolumeHealth	はい	はい

vSphere with Tanzu のストレージ権限

vSphere with Tanzu には、ストレージ操作の一連の権限を含むワークロード ストレージ マネージャというサンプル ロールがあります。このロールのクローンを作成して、同様のロールを作成することができます。

権限名	説明	必要とするオブジェクト
Cns.検索	ストレージ管理者がクラウド ネイティブ ストレージのユーザー インターフェイスを表示できるようにします。	root vCenter Server
データストア.容量の割り当て データストア.低レベルのファイル操作	仮想マシン、スナップショット、クローン、または仮想ディスク用に、データストアの領域を割り当てられるようにします。 データストア ブラウザ内で、読み取り、書き込み、削除、および名前変更操作を実行できるようにします。	パーシステント ボリュームが配置されている共有データストア
ESX Agent Manager.変更	仮想マシンのパワーオフや削除など、エージェント仮想マシンへの変更を可能にします。	vSphere ポッド
リソース.仮想マシンのリソース プールへの割り当て	リソース プールに仮想マシンを割り当てられるようにします。	リソース プール
プロファイルベースのストレージ.プロファイルベースのストレージの表示	定義済みストレージ ポリシーを表示できるようにします。	root vCenter Server
仮想マシン.構成の変更.既存ディスクの追加 仮想マシン.構成の変更.新規ディスクの追加 仮想マシン.構成の変更.デバイスの追加または削除 仮想マシン.構成の変更.設定の変更 仮想マシン.構成の変更.ディスクの削除 仮想マシン.インベントリの編集.新規作成 仮想マシン.インベントリの編集.削除	仮想マシンの作成と削除を許可します。仮想マシンのオプションとデバイスの構成を可能にします。	vSphere ポッド

vSphere with Tanzu のストレージ ポリシーの作成

vSphere with Tanzu を有効にする前に、スーパーバイザー クラスタ および名前空間で使用するストレージ ポリシーを作成します。このポリシーは vSphere 環境で使用可能なデータストアを表します。これらは、制御プレーン仮想マシン、ポッドの短期ディスク、コンテナ イメージ、パーシステント ストレージ ボリュームなどのオブジェクトのストレージの配置を制御します。Tanzu Kubernetes クラスタを使用する場合は、ストレージ ポリシーによって、Tanzu Kubernetes クラスタ ノードのデプロイ方法も決定されます。

vSphere ストレージ環境と DevOps のニーズに応じて、複数のストレージ ポリシーを作成してさまざまなストレージ クラスを表すことができます。たとえば、vSphere ストレージ環境にブロンズ、シルバー、ゴールドの 3 つのクラスのデータストアがある場合、すべてのデータストアに対してストレージ ポリシーを作成できます。その後、短期仮想ディスクとコンテナ イメージ仮想ディスクにブロンズ データストアを使用し、パーシステント ボリューム仮想ディスクにシルバーおよびゴールド データストアを使用することができます。ストレージ ポリシーの詳細については、『vSphere のストレージ』ドキュメントの「ストレージ ポリシー ベースの管理」の章を参照してください。

次の例では、ゴールドとタグ付けされたデータストアのストレージ ポリシーを作成します。

vSAN データ パーシステンス プラットフォームを使用する場合、ストレージ ポリシーを vSAN Direct または vSAN SNA データストアに対して作成できます。詳細については、vSAN Direct ストレージ ポリシーの作成 と vSAN SNA ストレージ ポリシーの作成 を参照してください。

前提条件

- ストレージ ポリシーで参照するデータストアが、クラスタ内のすべての ESXi ホスト間で共有されることを確認します。
- 必要な権限：仮想マシン ストレージ ポリシー.更新および仮想マシン ストレージ ポリシー.表示。

手順

1 データストアにタグを追加します。

a タグ付けするデータストアを右クリックし、[タグとカスタム属性] - [タグの割り当て] の順に選択します。

b [タグの追加] をクリックして、タグのプロパティを指定します。

プロパティ	説明
名前	データストア タグの名前を指定します（**Gold** など）。
説明	タグの説明を追加します（**Datastore for Kubernetes objects** など）。
カテゴリ	既存のカテゴリを選択するか、新しいカテゴリを作成します（**Storage for Kubernetes** など）。

2 vSphere Client で、[仮想マシン ストレージ ポリシーの作成] ウィザードを開きます。

a [メニュー] - [ポリシーおよびプロファイル] の順にクリックします。

b [ポリシーおよびプロファイル] で、[仮想マシン ストレージ ポリシー] をクリックします。

c [仮想マシン ストレージ ポリシーの作成] をクリックします。

3 ポリシーの名前と説明を入力します。

オプション	操作
vCenter Server	vCenter Server インスタンスを選択します。
名前	ストレージ ポリシーの名前（**goldsp** など）を入力します。 **注：** vSphere with Tanzu によって、名前空間に割り当てたストレージ ポリシーが Kubernetes ストレージ クラスに変換されると、すべての大文字は小文字に変更され、スペースがダッシュ (-) に置き換えられます。混乱を避けるため、仮想マシン ストレージ ポリシー名には小文字を使用し、スペースは含めないでください。
説明	ストレージ ポリシーの説明を入力します。

4 [データストア固有のルール] の [ポリシー構造] 画面で、タグベースの配置ルールを有効にします。

5 [タグベースの配置] ページで、タグ ルールを作成します。

次の例を使用してオプションを選択します。

オプション	説明
タグ カテゴリ	ドロップダウン メニューから、タグのカテゴリ（[Kubernetes のストレージ] など）を選択します。
使用量オプション	[以下のタグ付けをされたストレージを使用:] を選択します。
Tags	[タグを参照] をクリックし、データストア タグ（[Gold] など）を選択します。

6 [ストレージ互換性] ページでこのポリシーに適合するデータストアのリストを確認します。

この例では、ゴールドとタグ付けされたデータストアのみが表示されます。

7 [確認して完了] ページでポリシーの設定を確認し、[完了] をクリックします。

結果

既存のストレージ ポリシーのリストに、ゴールドとタグ付けされたデータストアの新しいストレージ ポリシーが表示されます。

次のステップ

ストレージ ポリシーを作成した後、vSphere 管理者は次のタスクを実行できます。

- ストレージ ポリシーを スーパーバイザー クラスタ に割り当てます。スーパーバイザー クラスタ に設定したストレージ ポリシーにより、制御プレーン仮想マシン、ポッドの短期ディスク、およびコンテナ イメージが、ポリシーによって表されるデータストアに配置されます。NSX-T Data Center ネットワークを使用したワークロード管理の有効化を参照してください。
- ストレージ ポリシーを vSphere 名前空間 に割り当てます。名前空間で認識されるストレージ ポリシーにより、名前空間がどのデータストアにアクセスしてパーシステント ボリュームに使用できるかが決まります。ストレージ ポリシーは、名前空間では一致する Kubernetes ストレージ クラスとして表示されます。このストレージ クラスは、この名前空間の Tanzu Kubernetes クラスタにも伝達されます。DevOps エンジニアは、ストレージ クラスをパーシステント ボリュームの要求指定で使用できます。vSphere 名前空間 の作成と設定を参照してください。

スーパーバイザー クラスタ のストレージ設定の変更

スーパーバイザー クラスタ に割り当てられたストレージ ポリシーに基づいて、制御プレーンの仮想マシン、vSphere ポッド、コンテナ イメージ キャッシュなどのオブジェクトが vSphere ストレージ環境のデータストア内に配置されます。vSphere 管理者は、通常、スーパーバイザー クラスタ を有効にするときにストレージ ポリシーを設定します。最初の スーパーバイザー クラスタ の設定後にストレージ ポリシーの割り当てを変更する必要がある場合は、このタスクを実行します。このタスクを使用して、ReadWriteMany パーシステント ボリュームのファイル ボリューム サポートを有効または無効にすることもできます。

ストレージ設定に加えた変更は、新しいオブジェクトにのみ適用されます。

前提条件

ReadWriteMany モードのパーシステント ボリュームに対してファイル ボリューム サポートを有効にする場合は、vSphere with Tanzu での ReadWriteMany パーシステント ボリュームの作成に記載されている前提条件に従ってください。

手順

1 vSphere Client で、vSphere with Tanzu が有効になっているホスト クラスタに移動します。

2 [設定] タブをクリックし、[名前空間] の [ストレージ] をクリックします。

3 次の項目のストレージ ポリシーの割り当てを変更します。

オプション	説明
制御プレーン ノード	制御プレーン仮想マシンを配置するためのストレージ ポリシーを選択します。
ポッドの短期ディスク	vSphere ポッド を配置するためのストレージ ポリシーを選択します。
コンテナ イメージ キャッシュ	コンテナ イメージのキャッシュを配置するためのストレージ ポリシーを選択します。

4 ReadWriteMany パーシステント ボリュームをデプロイするために、ファイル ボリューム サポートを有効にします。

名前空間のストレージ設定の変更

vSphere 管理者が スーパーバイザー クラスタ の名前空間に割り当てるストレージ ポリシーでは、パーシステント ボリュームと Tanzu Kubernetes クラスタ ノードを vSphere データストア内に配置する方法を制御します。パーシステント ボリュームに対応するパーシステント ボリュームの要求は、vSphere ポッド または Tanzu Kubernetes クラスタから行われることがあります。元のストレージ ポリシーの割り当てを変更できます。

前提条件

vCenter Server または vSphere 名前空間 からストレージ ポリシーを削除する前、またはストレージ ポリシーの割り当てを変更する前に、対応するストレージ クラスのパーシステント ボリュームの要求が名前空間で実行されていないことを確認します。また、そのストレージ クラスを使用している Tanzu Kubernetes クラスタがないことも確認します。

手順

1 vSphere Client で、名前空間に移動します。

   a vSphere Client ホーム メニューから、[ワークロード管理] を選択します。

   b [名前空間] タブをクリックして、名前空間をクリックします。

2 [ストレージ] タブをクリックして、[ストレージ ポリシー] をクリックします。

3 [編集] アイコンをクリックして、ストレージ ポリシーの割り当てを変更します。

vSphere 名前空間 または Tanzu Kubernetes クラスタでのストレージ クラスの表示

vSphere 管理者がストレージ ポリシーを作成して vSphere 名前空間 に割り当てると、そのストレージ ポリシーは、この名前空間および使用可能な Tanzu Kubernetes クラスタでは、一致する Kubernetes ストレージ クラスとして表示されます。DevOps エンジニアは、ストレージ クラスが利用可能であることを確認できます。

コマンドを実行できるかどうかは、ユーザーの権限によって異なります。

前提条件

vSphere 管理者が適切なストレージ ポリシーを作成し、そのポリシーを vSphere 名前空間 に割り当てていることを確認します。

手順

1 次のいずれかのコマンドを使用して、ストレージ クラスが使用可能であることを確認します。

- **`kubectl get storageclass`**

  注： このコマンドは、管理者権限を持つユーザーのみが使用できます。

  次のような出力が表示されます。ストレージ クラスの名前は、vSphere 側のストレージ ポリシーの名前と一致します。

```
NAME     PROVISIONER              AGE
silver   csi.vsphere.vmware.com   2d
gold     csi.vsphere.vmware.com   1d
```

- **`kubectl describe namespace`** ***`namespace_name`***

  出力では、ストレージ クラスの名前は、***`storageclass_name`*****`.storageclass.storage.k8s.io/requests.storage`** パラメータの一部として表示されます。例：

```
------------------
Name:                                                          namespace_name
 Resource                                                       Used  Hard
 --------                                                       ---   ---
```

```
 silver.storageclass.storage.k8s.io/requests.storage              1Gi
9223372036854775807
 gold.storageclass.storage.k8s.io/requests.storage                0
9223372036854775807
```

2 名前空間で使用可能なストレージ容量を確認するには、次のコマンドを実行します。

kubectl describe resourcequotas -namespace *namespace*

次のような出力が表示されます。

```
Name:             ns-my-namespace
Namespace:        ns-my-namespace
Resource          Used  Hard
--------          ----  ----
requests.storage  0     200Gi
```

ステートフル アプリケーションの動的パーシステント ボリュームのプロビジョニング

データベースなどのステートフル アプリケーションでは、セッション間でデータが保存されるため、データを格納するためのパーシステント ストレージが必要です。保持されたデータは、アプリケーションの状態と呼ばれます。後でデータを取得し、次のセッションで使用することができます。Kubernetes は、状態とデータを保持できるオブジェクトとしてパーシステント ボリュームを提供します。

vSphere 環境では、パーシステント ボリューム オブジェクトは、データストアにある仮想ディスクによってバッキングされます。データストアはストレージ ポリシーによって表されます。vSphere 管理者が **gold** などのストレージ ポリシーを作成して、スーパーバイザー クラスタ 内の名前空間に割り当てると、そのストレージ ポリシーは、vSphere 名前空間 および使用可能な Tanzu Kubernetes クラスタでは、一致する Kubernetes ストレージ クラスとして表示されます。

DevOps エンジニアは、このストレージ クラスをパーシステント ボリュームの要求指定で使用できます。その後、パーシステント ボリュームの要求から得たストレージを使用するアプリケーションをデプロイできます。この例では、アプリケーションのパーシステント ボリュームが動的に作成されています。

前提条件

vSphere 管理者が適切なストレージ ポリシーを作成し、そのポリシーを名前空間に割り当てていることを確認します。

手順

1 vSphere Kubernetes 環境内の名前空間にアクセスします。

2 ストレージ クラスを使用できることを確認します。

vSphere 名前空間 または Tanzu Kubernetes クラスタでのストレージ クラスの表示を参照してください。

3 パーシステント ボリュームの要求を作成します。

a パーシステント ボリュームの要求設定を含む YAML ファイルを作成します。

この例では、ファイルは **gold** ストレージ クラスを参照しています。

ReadWriteMany パーシステント ボリュームをプロビジョニングするには、`accessModes` を `ReadWriteMany` に設定します。vSphere with Tanzu での ReadWriteMany パーシステント ボリュームの作成を参照してください。

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: gold
  resources:
    requests:
        storage: 3Gi
```

b Kubernetes クラスタにパーシステント ボリュームの要求を適用します。

```
kubectl apply -f pvc_name.yaml
```

このコマンドでは、要求のストレージ要件を満たすバッキング仮想ディスクを持つ Kubernetes パーシステント ボリュームと vSphere ボリュームが動的に作成されます。

c パーシステント ボリュームの要求のステータスを確認します。

```
kubectl get pvc my-pvc
```

出力には、ボリュームがパーシステント ボリュームの要求にバインドされていることが示されます。

```
NAME     STATUS    VOLUME   CAPACITY   ACCESSMODES   STORAGECLASS   AGE
my-pvc   Bound     my-pvc   2Gi        RWO           gold           30s
```

4 パーシステント ボリュームをマウントするポッドを作成します。

a パーシステント ボリュームを含む YAML ファイルを作成します。

このファイルには次のパラメータが含まれます。

```
...
volumes:
    - name: my-pvc
      persistentVolumeClaim:
        claimName: my-pvc
```

b YAML ファイルからポッドをデプロイします。

```
kubectl create -f pv_pod_name.yaml
```

c ポッドが作成されたことを確認します。

```
kubectl get pod
```

```
NAME        READY   STATUS    RESTARTS   AGE
pod_name    1/1     Ready     0          40s
```

結果

設定したポッドでは、パーシステント ボリュームの要求で指定されているパーシステント ストレージが使用されます。

次のステップ

パーシステント ボリュームの健全性ステータスを監視するには、vSphere 名前空間 または Tanzu Kubernetes クラスタでのボリュームの健全性の監視を参照してください。vSphere Client のパーシステント ボリュームを確認して監視するには、vSphere Client のパーシステント ボリュームの監視を参照してください。

Tanzu Kubernetes クラスタでの静的パーシステント ボリュームのプロビジョニング

スーパーバイザー クラスタ から未使用のパーシステント ボリューム要求 (PVC) を使用して、Tanzu Kubernetes クラスタ内にブロック ボリュームを静的に作成することができます。

PVC は次の条件を満たす必要があります。

- PVC は Tanzu Kubernetes クラスタが配置されている名前空間内に存在します。
- PVC は スーパーバイザー クラスタ の vSphere ポッド または Tanzu Kubernetes クラスタ内のポッドに接続されていません。

静的プロビジョニングを使用すると、別の Tanzu Kubernetes クラスタで不要になった PVC を新しい Tanzu Kubernetes クラスタで再利用することもできます。再利用するには、元の Tanzu Kubernetes クラスタ内のパーシステント ボリューム (PV) の `Reclaim policy` を `Retain` に変更して、対応する PV を削除します。

次の手順に従って、残りの基盤となるボリュームの情報を使用して新しい Tanzu Kubernetes クラスタ内に PVC を静的に作成します。

手順

1 スーパーバイザー クラスタ 内の元の PVC の名前を書き留めます。

古い Tanzu Kubernetes クラスタの PVC を再利用する場合は、Tanzu Kubernetes クラスタ内の古い PV オブジェクトの `volumeHandle` から PVC 名を取得できます。

2 PV を作成します。

YAML ファイルで、次のアイテムの値を指定します。

- `storageClassName` には、スーパーバイザー クラスタ で PVC が使用しているストレージ クラスの名前を入力できます。
- `volumeHandle` には、手順 1 で取得した PVC 名を入力します。

別の Tanzu Kubernetes クラスタのボリュームを再利用する場合は、新しい Tanzu Kubernetes クラスタで PV を作成する前に、古い Tanzu Kubernetes クラスタから PVC と PV オブジェクトを削除します。

例として、次の YAML マニフェストを使用します。

```
apiVersion: v1
   kind: PersistentVolume
   metadata:
     name: static-tkg-block-pv
     annotations:
       pv.kubernetes.io/provisioned-by: csi.vsphere.vmware.com
   spec:
     storageClassName: gc-storage-profile
     capacity:
       storage: 2Gi
     accessModes:
       - ReadWriteOnce
     persistentVolumeReclaimPolicy: Delete
     claimRef:
       namespace: default
       name: static-tkg-block-pvc
     csi:
       driver: "csi.vsphere.vmware.com"
       volumeAttributes:
         type: "vSphere CNS Block Volume"
       volumeHandle: "supervisor-block-pvc-name"  # Enter the PVC name from the Supervisor
cluster.
```

3 手順 手順 2 で作成した PV オブジェクトと一致する PV を作成します。

`storageClassName` を PV と同じ値に設定します。

```
kind: PersistentVolumeClaim
   apiVersion: v1
   metadata:
     name: static-tkg-block-pvc
   spec:
```

```
    accessModes:
      - ReadWriteOnce
    resources:
      requests:
        storage: 2Gi
    storageClassName: gc-storage-profile
    volumeName: static-tkg-block-pv
```

4 作成した PV に PVC がバインドされていることを確認します。

```
$ kubectl get pv,pvc
 NAME                                     CAPACITY   ACCESS MODES   RECLAIM POLICY
STATUS   CLAIM                           STORAGECLASS         REASON   AGE
 persistentvolume/static-tkg-block-pv    2Gi        RWO            Delete
Bound    default/static-tkg-block-pvc    gc-storage-profile            10s

 NAME                                            STATUS   VOLUME                CAPACITY
ACCESS MODES   STORAGECLASS         AGE
 persistentvolumeclaim/static-tkg-block-pvc   Bound    static-tkg-block-pv   2Gi
RWO            gc-storage-profile   10s
```

vSphere with Tanzu での ReadWriteMany パーシステント ボリュームの作成

vSphere 7.0 Update 3 リリース以降、vSphere with Tanzu では ReadWriteMany モードのパーシステント ボリュームがサポートされます。ReadWriteMany サポートにより、1 つのクラスタ内で実行されている複数のポッドまたはアプリケーションによって 1 つのボリュームを同時にマウントできます。vSphere with Tanzu では、ReadWriteMany パーシステント ボリュームにファイル共有を提供するために vSAN ファイル サービスが使用されます。

ReadWriteMany パーシステント ボリュームに関する考慮事項

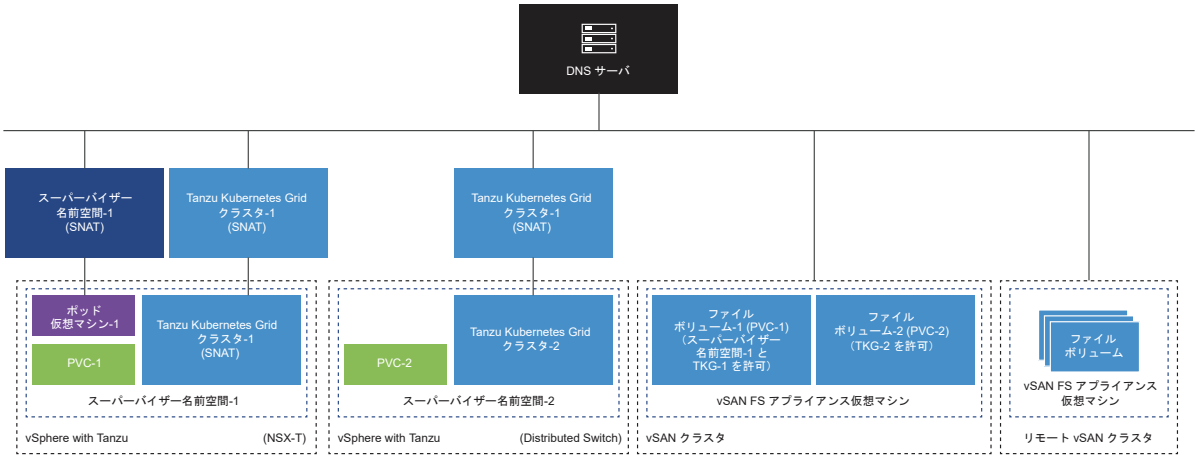
vSphere with Tanzu でパーシステント ボリュームの ReadWriteMany サポートを有効にする場合は、次の点に注意してください。

- Tanzu Kubernetes クラスタでは、TKr バージョン 1.22 を使用します。

  注： ReadWriteMany 機能を使用できるのは、次回の TKr バージョン 1.22 がリリースされている場合のみです。TKr バージョンの詳細については、VMware Tanzu Kubernetes リリース ノートを参照してください。

- vSphere with Tanzu に対してファイル ボリュームのサポートを有効にすると、セキュリティが脆弱になる可能性がありますのでご注意ください。
  - ボリュームは暗号化なしでマウントされます。暗号化されていないデータは、データがネットワークを通過する間にアクセスされる可能性があります。
  - スーパーバイザー名前空間内でファイル共有アクセスを隔離するために、アクセス コントロール リスト (ACL) がファイル共有に対して使用されます。これにより、IP スプーフィングのリスクが発生する可能性があります。

- ネットワークについては、次のガイドラインに従ってください。
  - vSphere 名前空間 が NAT モードであることを確認します。vSphere 名前空間 の作成と設定を参照してください。
  - vSAN ファイル サービスがワークロード ネットワークからルーティング可能であることと、ワークロード ネットワークと vSAN ファイル サービスの IP アドレスの間に NAT がないことを確認します。
  - vSAN ファイル サービスと vSphere クラスタに、共通の DNS サーバを使用します。



- ファイル ボリュームのサポートを有効にし、後から無効にしても、クラスタにプロビジョニングした既存の ReadWriteMany パーシステント ボリュームは影響を受けず、使用可能なままになります。新しい ReadWriteMany パーシステント ボリュームを作成することはできなくなります。

パーシステント ボリュームの ReadWriteMany サポートを有効にするためのワークフロー

このプロセスに従って、パーシステント ボリュームに対する ReadWriteMany サポートを有効にします。

1 vSphere 管理者が、vSAN ファイル サービスが構成された vSAN クラスタを設定します。ファイル サービスの構成を参照してください。

2 vSphere 管理者は、スーパーバイザー クラスタ でファイル ボリュームのサポートを有効にします。

アクション	説明
ワークロード管理プラットフォームを有効にする場合は、ファイル ボリュームのサポートを有効にします。	■ vSphere ネットワークを使用したワークロード管理の有効化 ■ NSX-T Data Center ネットワークを使用したワークロード管理の有効化
vSphere with Tanzu のアップグレード後などに、既存のクラスタでファイル ボリュームのサポートを有効にします。	スーパーバイザー クラスタ のストレージ設定の変更

3 DevOps エンジニアがパーシステント ボリュームをプロビジョニングし、PVC の `accessMode` を `ReadWriteMany` と設定します。

同じ PVC で複数のポッドをプロビジョニングできます。

ステートフル アプリケーションの動的パーシステント ボリュームのプロビジョニングを参照してください。

vSphere with Tanzu でのボリュームの拡張

DevOps エンジニアは、Kubernetes ボリューム拡張機能を使用して、パーシステント ブロック ボリュームを作成後に拡張できます。オフライン ボリュームとオンライン ボリュームの拡張は、スーパーバイザー クラスタ と Tanzu Kubernetes の両方のタイプのクラスタでサポートされます。

デフォルトでは、vSphere with Tanzu 環境に表示されるストレージ クラスについて、`allowVolumeExpansion` が `true` に設定されています。このパラメータを使用して、オフライン ボリュームまたはオンライン ボリュームのサイズを変更できます。

ボリュームは、ノードまたはポッドに接続されていない場合はオフラインと見なされます。オンライン ボリュームは、ノードまたはポッドで使用可能なボリュームです。

ボリューム拡張機能のサポート レベルは、vSphere のバージョンによって異なります。拡張をサポートする適切なバージョンに vSphere 環境をアップグレードすると、以前のバージョンの vSphere で作成されたボリュームを拡張できます。

Tanzu Kubernetes クラスタを使用する場合は、Tanzu Kubernetes クラスタと スーパーバイザー クラスタ の両方を、機能がサポートされる適切なバージョンにアップグレードしてください。Tanzu Kubernetes クラスタの機能は、スーパーバイザー クラスタ でその機能が有効かどうかで決まります。

たとえば、Tanzu Kubernetes クラスタを vSphere 7.0 Update 2 にアップグレードし、スーパーバイザー クラスタ を 7.0 Update 1 のままにすると、Tanzu Kubernetes クラスタでオンライン ボリュームの拡張が機能しません。

	スーパーバイザー クラスタ 7.0	スーパーバイザー クラスタ 7.0 Update 1	スーパーバイザー クラスタ 7.0 Update 2
Tanzu Kubernetes クラスタ 7.0	Tanzu Kubernetes クラスタまたは スーパーバイザー クラスタにおけるオフラインおよびオンライン拡張：`not supported`	Tanzu Kubernetes クラスタまたは スーパーバイザー クラスタにおけるオフラインおよびオンライン拡張：`not supported`	■ Tanzu Kubernetes クラスタにおけるオフラインおよびオンライン拡張：`not supported` ■ スーパーバイザー クラスタ におけるオフラインおよびオンライン拡張：`supported`

Tanzu Kubernetes クラスタ 7.0 Update 1	Tanzu Kubernetes クラスタまたは スーパーバイザー クラスタにおけるオフラインおよびオンライン拡張：not supported	■ Tanzu Kubernetes クラスタにおけるオフライン拡張：supported ■ スーパーバイザー クラスタにおけるオフライン拡張：not supported ■ Tanzu Kubernetes クラスタまたは スーパーバイザー クラスタ におけるオンライン拡張：not supported	■ Tanzu Kubernetes クラスタにおけるオフライン拡張：supported ■ Tanzu Kubernetes クラスタにおけるオンライン拡張：not supported ■ スーパーバイザー クラスタ におけるオフラインおよびオンライン拡張：supported
Tanzu Kubernetes クラスタ 7.0 Update 2	Tanzu Kubernetes クラスタまたは スーパーバイザー クラスタにおけるオフラインおよびオンライン拡張：not supported	■ Tanzu Kubernetes クラスタにおけるオフライン拡張：supported ■ スーパーバイザー クラスタにおけるオフライン拡張：not supported ■ Tanzu Kubernetes クラスタまたは スーパーバイザー クラスタ におけるオンライン拡張：not supported	Tanzu Kubernetes クラスタまたは スーパーバイザー クラスタ におけるオフラインおよびオンライン拡張：supported

ボリュームを拡張する場合は、次の点に注意してください。

- ボリュームは、ストレージ割り当てで指定された制限まで拡張できます。vSphere with Tanzu は、パーシステント ボリュームの要求オブジェクトに対する連続したサイズ変更要求をサポートします。
- VMFS、vSAN、vSAN Direct、vVols、NFS など、すべてのタイプのデータストアでボリューム拡張がサポートされます。
- デプロイまたはスタンドアローン ポッドのボリューム拡張を実行できます。
- 静的にプロビジョニングされたボリュームにストレージ クラスが関連付けられている場合は、スーパーバイザー クラスタ および Tanzu Kubernetes クラスタでそのボリュームのサイズを変更できます。
- StatefulSet の一部として作成されたボリュームは拡張できません。
- ボリュームをバッキングする仮想ディスクにスナップショットがある場合、そのサイズは変更できません。
- vSphere with Tanzu では、ツリー内または移行後のボリュームの拡張はサポートされません。

オフライン モードでのパーシステント ボリュームの拡張

ボリュームは、ノードまたはポッドに接続されていない場合はオフラインと見なされます。オフライン ボリュームの拡張は、スーパーバイザー クラスタ と Tanzu Kubernetes の両方のタイプのクラスタでサポートされます。

前提条件

vSphere 環境がオフライン ボリュームの拡張をサポートする適切なバージョンにアップグレードされていることを確認します。vSphere with Tanzu でのボリュームの拡張を参照してください。

手順

1 ストレージ クラスを使用して、パーシステント ボリュームの要求 (PVC) を作成します。

a 例として、次の YAML マニフェストを使用して PVC を定義します。

この例では、要求されるストレージのサイズは 1 Gi です。

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: example-block-pvc
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: example-block-sc
```

b Kubernetes クラスタに PVC を適用します。

```
kubectl apply -f example-block-pvc.yaml
```

2 PVC にパッチを適用してサイズを増やします。

PVC がノードに接続されていない場合、またはポッドで使用されていない場合は、次のコマンドを使用して PVC にパッチを適用します。この例では、要求されるストレージの増加は 2 Gi です。

```
kubectl patch pvc example-block-pvc -p '{"spec": {"resources": {"requests": {"storage":
"2Gi"}}}}'
```

この操作により、PVC に関連付けられているボリュームで拡張がトリガされます。

3 ボリュームのサイズが増加したことを確認します。

```
kubectl get pv
NAME                                        CAPACITY ACCESS MODES RECLAIM POLICY STATUS
CLAIM                          STORAGECLASS           REASON AGE
pvc-9e9a325d-ee1c-11e9-a223-005056ad1fc1   2Gi           RWO         Delete     Bound
default/example-block-pvc   example-block-sc              6m44s
```

注： PVC がポッドによって使用されるまで、PVC のサイズは変更されません。

次の例は、PVC のサイズが変更されていないことを示しています。PVC を記述すると、PVC に適用されている `FilesystemResizePending` 条件が表示されます。

```
kubectl get pvc
NAME                STATUS VOLUME                                     CAPACITY ACCESS
MODES   STORAGECLASS       AGE
example-block-pvc   Bound  pvc-9e9a325d-ee1c-11e9-a223-005056ad1fc1   1Gi
RWO       example-block-sc   6m57s
```

4 PVC を使用するポッドを作成します。

PVC がポッドで使用されると、ファイルシステムが拡張されます。

5 PVC のサイズが変更されたことを確認します。

```
kubectl get pvc
NAME                STATUS VOLUME                                     CAPACITY ACCESS MODES
STORAGECLASS     AGE
example-block-pvc   Bound  pvc-24114458-9753-428e-9c90-9f568cb25788   2Gi         RWO
example-block-sc 2m12s
```

`FilesystemResizePending` 条件は PVC から削除されています。ボリュームの拡張が完了しました。

次のステップ

vSphere 管理者は、vSphere Client で新しいボリューム サイズを確認できます。vSphere Client のパーシステント ボリュームの監視を参照してください。

オンライン モードでのパーシステント ボリュームの拡張

オンライン ボリュームは、ノードまたはポッドで使用可能なボリュームです。DevOps エンジニアは、オンラインのパーシステント ブロック ボリュームを拡張できます。オンライン ボリュームの拡張は、スーパーバイザー クラスタ と Tanzu Kubernetes の両方のタイプのクラスタでサポートされます。

前提条件

vSphere 環境がオンライン ボリュームの拡張をサポートする適切なバージョンにアップグレードされていることを確認します。vSphere with Tanzu でのボリュームの拡張を参照してください。

手順

1 サイズを変更するパーシステント ボリュームの要求を見つけます。

```
$ kubectl get pv,pvc,pod
NAME                                                        CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS      CLAIM               STORAGECLASS   REASON   AGE
persistentvolume/pvc-5cd51b05-245a-4610-8af4-f07e77fdc984   1Gi        RWO
Delete           Bound       default/block-pvc   block-sc                 4m56s

NAME                                STATUS   VOLUME
CAPACITY   ACCESS MODES   STORAGECLASS   AGE
persistentvolumeclaim/block-pvc   Bound    pvc-5cd51b05-245a-4610-8af4-f07e77fdc984
1Gi        RWO            block-sc       5m3s

NAME            READY   STATUS    RESTARTS   AGE
pod/block-pod   1/1     Running   0          26s
```

ボリュームで使用されているストレージのサイズは 1 Gi です。

2 PVC にパッチを適用してサイズを増やします。

たとえば、サイズを 2 Gi に増やします。

```
$ kubectl patch pvc block-pvc -p '{"spec": {"resources": {"requests": {"storage":
"2Gi"}}}}'
persistentvolumeclaim/block-pvc edited
```

この操作により、PVC に関連付けられているボリュームで拡張がトリガされます。

3 PVC と PV の両方のサイズが増加したことを確認します。

```
$ kubectl get pvc,pv,pod
NAME                                STATUS   VOLUME
CAPACITY   ACCESS MODES   STORAGECLASS   AGE
persistentvolumeclaim/block-pvc   Bound    pvc-5cd51b05-245a-4610-8af4-f07e77fdc984
2Gi        RWO            block-sc       6m18s

NAME                                                        CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS      CLAIM               STORAGECLASS   REASON   AGE
persistentvolume/pvc-5cd51b05-245a-4610-8af4-f07e77fdc984   2Gi        RWO
Delete           Bound       default/block-pvc   block-sc                6m11s

NAME            READY   STATUS    RESTARTS   AGE
pod/block-pod   1/1     Running   0          101s
```

次のステップ

vSphere 管理者は、vSphere Client で新しいボリューム サイズを確認できます。vSphere Client のパーシステント ボリュームの監視を参照してください。

vSphere Client のパーシステント ボリュームの監視

DevOps エンジニアがパーシステント ボリュームの要求を使用してステートフル アプリケーションをデプロイすると、vSphere with Tanzu により、パーシステント ボリューム オブジェクトとそれに対応するパーシステント仮想ディスクが作成されます。vSphere 管理者は、vSphere Client でパーシステント ボリュームの詳細を確認できます。また、そのストレージ コンプライアンスと健全性ステータスを監視することもできます。

手順

1 vSphere Client で、パーシステント ボリュームのある名前空間に移動します。

a vSphere Client ホーム メニューから、[ワークロード管理] を選択します。

b 名前空間をクリックします。

2 [ストレージ] タブをクリックし、[パーシステント ボリュームの要求] をクリックします。

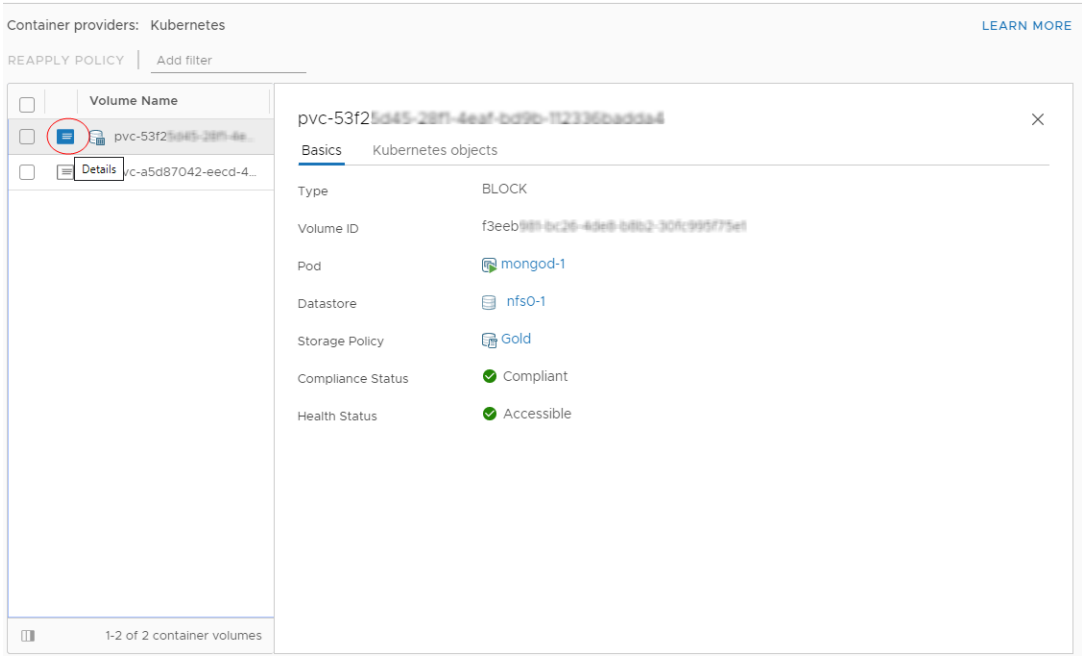
vSphere Client に、すべてのパーシステント ボリュームの要求オブジェクトと、名前空間内で使用可能な対応するボリュームが一覧表示されます。

3 選択したパーシステント ボリュームの要求の詳細を表示するには、[パーシステント ボリュームの名前] 列でボリュームの名前をクリックします。

4 [コンテナ ボリューム] 画面で、ボリュームの健全性ステータスとストレージ ポリシーのコンプライアンスを確認します。

a [詳細] アイコンをクリックし、[基本] タブと [Kubernetes オブジェクト] タブを切り替えて、Kubernetes パーシステント ボリュームの追加情報を表示します。

`kubectl` コマンドを使用してボリュームの健全性ステータスを監視するには、vSphere 名前空間 または Tanzu Kubernetes クラスタでのボリュームの健全性の監視を参照してください。



b ボリュームの健全性ステータスを確認します。

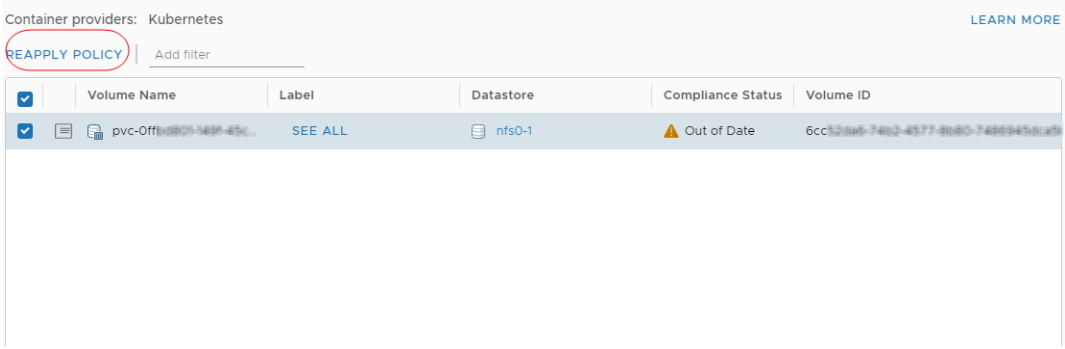
健全性ステータス	説明
アクセスの可能性	パーシステント ボリュームにアクセスして、使用できます。
アクセス不可	パーシステント ボリュームにアクセスしたり、使用したりできません。データストアに接続しているホストからボリュームが格納されているデータストアにアクセスできない場合、パーシステント ボリュームにはアクセスできなくなります。

c ストレージ コンプライアンスの状態を確認します。

[コンプライアンスの状態] 列に次のいずれかが表示されます。

コンプライアンス ステータス	説明
準拠	ボリュームをバッキングする仮想ディスクがあるデータストアには、ポリシーで必要とされるストレージ機能があります。
旧バージョン	このステータスは、ポリシーが編集されており、新しいストレージ要件がデータストアに伝送されていないことを示しています。変更を伝送するには、ポリシーを期限切れのボリュームに再適用します。
コンプライアンスに非準拠	データストアは特定のストレージ要件をサポートしますが、現在はストレージ ポリシーを満たすことができません。たとえば、データストアの物理リソースが使用不可の場合に、ステータスが「非準拠」になることがあります。ホストやディスクをクラスタに追加する方法などで、ホスト クラスタの物理構成を変更するとデータストアを準拠させることができます。その他のリソースがストレージ ポリシーを満たす場合は、ステータスが「準拠」に変わります。
該当なし	ストレージ ポリシーは、データストアでサポートされていないデータストア機能を参照しています。

d コンプライアンスの状態が「期限切れ」になっている場合は、ボリュームを選択して [ポリシーの再適用] をクリックします。



ステータスが「準拠」になります。

vSphere 名前空間 または Tanzu Kubernetes クラスタでのボリュームの健全性の監視

DevOps エンジニアは、バインド状態のパーシステント ボリュームの健全性ステータスを確認できます。

バインド状態の各パーシステント ボリュームの健全性ステータスは、パーシステント ボリュームにバインドされているパーシステント ボリューム要求の `Annotations: volumehealth.storage.kubernetes.io/messages:` フィールドに表示されます。健全性ステータスには、2 つの値があります。

健全性ステータス	説明
アクセスの可能性	パーシステント ボリュームにアクセスして、使用できます。
アクセス不可	パーシステント ボリュームにアクセスしたり、使用したりできません。データストアに接続しているホストからボリュームが格納されているデータストアにアクセスできない場合、パーシステント ボリュームにはアクセスできなくなります。

vSphere Client のボリュームの健全性ステータスを監視するには、vSphere Client のパーシステント ボリュームの監視を参照してください。

手順

1 vSphere Kubernetes 環境内の名前空間にアクセスします。

2 パーシステント ボリュームの要求を作成します。

a パーシステント ボリュームの要求設定を含む YAML ファイルを作成します。

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: gold
  resources:
    requests:
        storage: 2Gi
```

b Kubernetes クラスタにパーシステント ボリュームの要求を適用します。

```
kubectl apply -f pvc_name.yaml
```

このコマンドでは、要求のストレージ要件を満たすバッキング仮想ディスクを持つ Kubernetes パーシステント ボリュームと vSphere ボリュームが作成されます。

c パーシステント ボリュームの要求がボリュームにバインドされているかどうかを確認します。

```
kubectl get pvc my-pvc
```

出力には、パーシステント ボリュームの要求と、ボリュームがバインド状態であることが示されます。

```
NAME     STATUS    VOLUME   CAPACITY   ACCESSMODES   STORAGECLASS   AGE
my-pvc   Bound     my-pvc   2Gi        RWO           gold           30s
```

3 ボリュームの健全性ステータスを確認します。

次のコマンドを実行して、パーシステント ボリュームにバインドされたパーシステント ボリューム要求のボリューム健全性に関する注釈を確認します。

```
kubectl describe pvc my-pvc
```

次のサンプル出力では、`volumehealth.storage.kubernetes.io/messages` フィールドに健全性ステータスがアクセス可能と表示されています。

```
Name:          my-pvc
Namespace:     test-ns
StorageClass:  gold
Status:        Bound
Volume:        my-pvc
Labels:        <none>
Annotations:   pv.kubernetes.io/bind-completed: yes
```

```
                pv.kubernetes.io/bound-by-controller: yes
                volume.beta.kubernetes.io/storage-provisioner: csi.vsphere.vmware.com
                volumehealth.storage.kubernetes.io/messages: accessible
Finalizers:     [kubernetes.io/pvc-protection]
Capacity:       2Gi
Access Modes:   RWO
VolumeMode:     Filesystem
```

最新のステートフル サービスでの vSAN データ パーシステンス プラットフォームの使用

パーシステント ストレージを必要とする最新のステートフル サービスに vSAN データ パーシステンス プラットフォームを使用することができます。このプラットフォームでは、サードパーティが自社のサービス アプリケーションを基盤となる vSphere インフラストラクチャと統合できるようにするフレームワークが提供されているため、サードパーティ製のソフトウェアは vSphere with Tanzu で最適に実行可能です。

vSAN データ パーシステンス を使用するメリットは、次のとおりです。

サービスの自動デプロイと拡張

管理者は vSphere Client を使用して、スーパーバイザー クラスタ に最新のステートフル サービスをインストールおよびデプロイすることで、DevOps エンジニアにサービス名前空間へのアクセス権を付与できます。DevOps エンジニアは、Kubernetes API を使用して、ステートフル サービスのインスタンスのプロビジョニングおよびスケーリングをセルフサービス方式で動的に行うことができます。

vCenter Server と統合されたサービス監視

パートナーは、vCenter Server と統合するダッシュボード プラグインを構築できます。vSphere 管理者は、ユーザー インターフェイス プラグインを使用して、ステートフル サービスを管理および監視できます。また、vSAN では、これらの統合サードパーティ サービスの健全性機能とキャパシティ監視機能も提供されています。

vSAN Direct を使用する最適化されたストレージ構成

vSAN Direct を使用すると、最新のステートフル サービスは基盤となる直接接続型ストレージと直接通信して、I/O とストレージの効率を最適化できます。

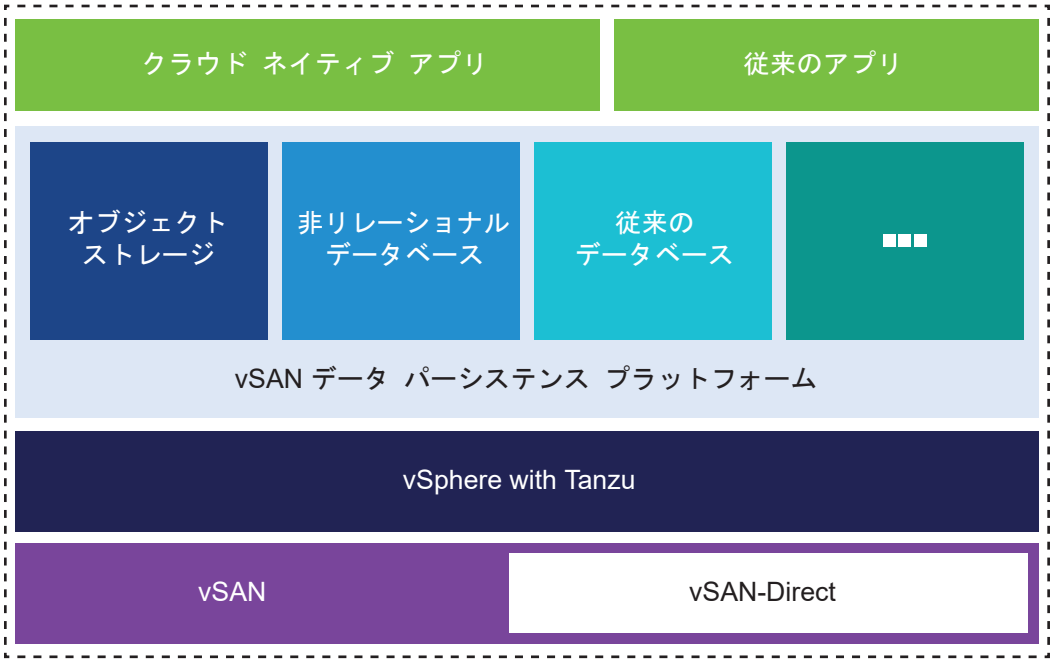
プラットフォームは、次のタイプのサービスをサポートしています。

- オブジェクト ストレージ（MinIO など）。
- NoSQL データベース（別名、非リレーショナル データベース）。
- 従来のデータベース。

vSphere Shared Nothing Storage

通常、ほとんどの最新ステートフル サービスは Shared Nothing Architecture (SNA) を採用しています。これは、複製されていないローカル ストレージを使用し、独自のストレージ レプリケーション、圧縮、その他のデータ操作を提供します。そのため、同じ操作を基盤となるストレージが実行してもサービスにとってはメリットがありません。

vSAN データ パーシステンス プラットフォームでは、操作の重複を避けるために、データ パスを最適化した 2 つの vSAN ソリューションが提供されます。パーシステント サービスは、SNA ストレージ ポリシーを使用する vSAN 上、または基本的に Raw ローカル ストレージである vSAN Direct 上で実行できます。



SNA ストレージ ポリシーを使用する vSAN

このテクノロジーの採用により、vSAN ホストのローカル SNA ポリシーとレプリケートされた分散 vSAN データストアを併用することができます。その結果、SNA サービス アプリケーションは配置を制御し、データ可用性を維持する作業を引き継ぐことができます。このテクノロジーにより、パーシステント サービスは、コンピューティング インスタンスとストレージ オブジェクトを同じ物理 ESXi ホストに共存させることが容易になります。ホストをローカルに配置すると、これらの操作を、ストレージ レイヤーではなく、サービス レイヤーで、レプリケーションとして実行できるようになります。

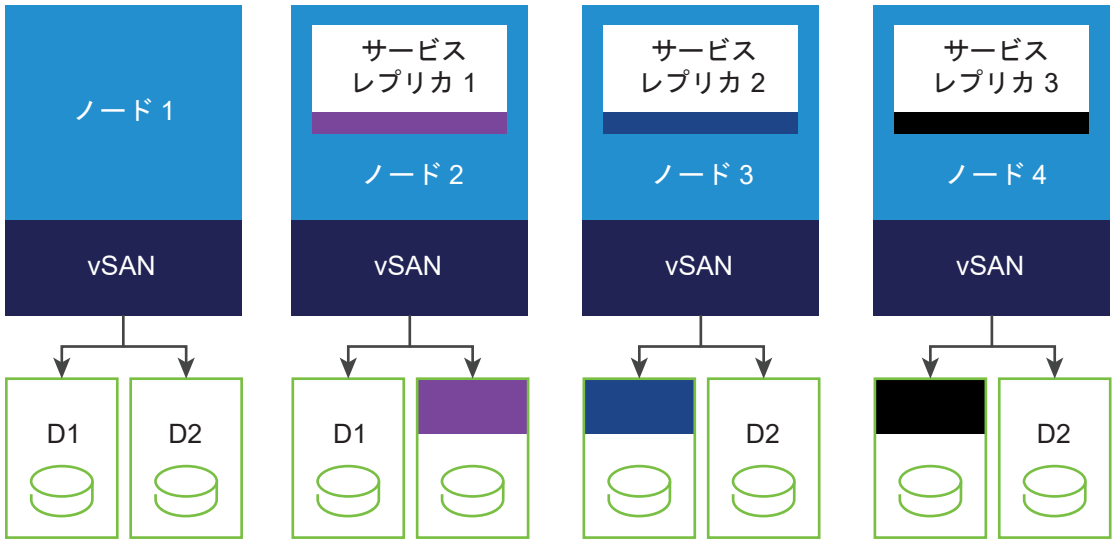
ポッドなどのコンピューティング インスタンスは、最初は、vSAN クラスタ内のいずれかのノードに配置されます。その後、vSAN SNA ポリシーによって作成された vSAN オブジェクトのすべてのデータは、ポッドが実行されているノードに自動的に配置されます。

以下に、SNA ストレージ クラスを使用するアプリケーションのストレージをパーシステント ボリュームに配置する例を示します。vSAN は、ノード上のディスク グループの中から、パーシステント ボリュームに配置するものを任意に選択できます。

データの合計コピー数 = 3

予想されるフォールト トレランス = 2

許容される実際の障害数 = 2

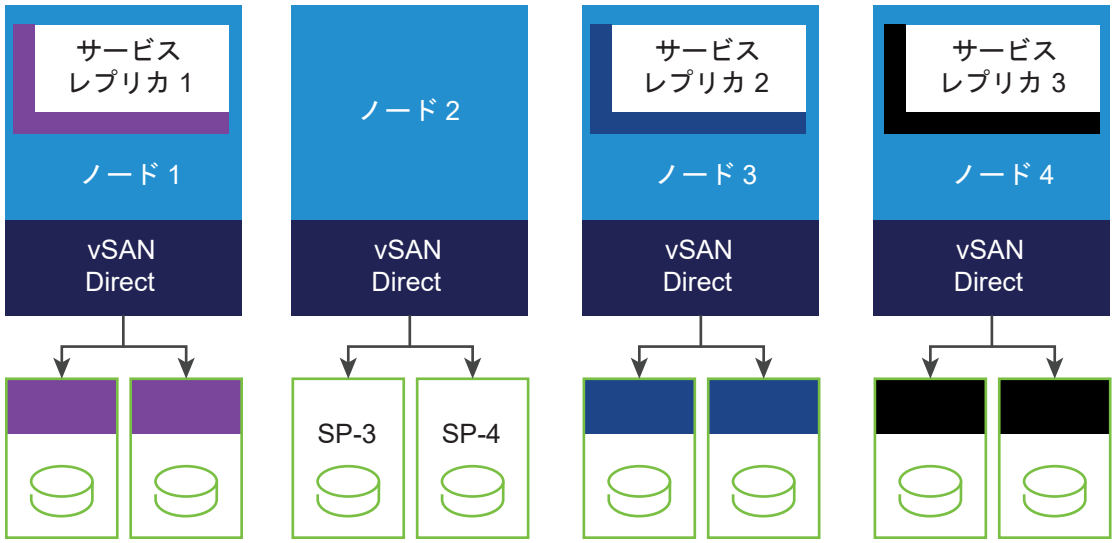


vSAN Direct

SNA ストレージ ポリシーを使用する vSAN によってデータをコンピューティング インスタンスにローカルに配置できますが、アプリケーションと物理ストレージ デバイス間の分散 vSAN データ パスにオーバーヘッドが発生します。vSAN Direct が有効な場合、ステートフル サービス アプリケーションはより直接的なデータ パスを使用して vSAN 以外のほとんどの Raw ローカル ストレージにアクセスできるため、ソリューションのパフォーマンスが最適化されます。

vSphere 管理者は vSAN Direct を使用することで、ホストのローカル デバイスを要求し、これらのデバイスを管理および監視することができます。vSAN Direct は、デバイスの健全性、パフォーマンス、キャパシティに関するインサイトを提供します。vSAN Direct は、要求されたすべてのローカル デバイスに独立した VMFS データストアを作成し、アプリケーションの配置先の候補として使用できるようにします。vSAN Direct によって管理される VMFS データストアは、Kubernetes ではストレージ プールとして公開されます。vSphere Client では vSAN Direct データストアとして表示されます。

次の図に、vSAN Direct ディスクにローカルに配置されたパーシステント ボリュームを示します。



SNA を使用する vSAN と vSAN Direct の使い分け

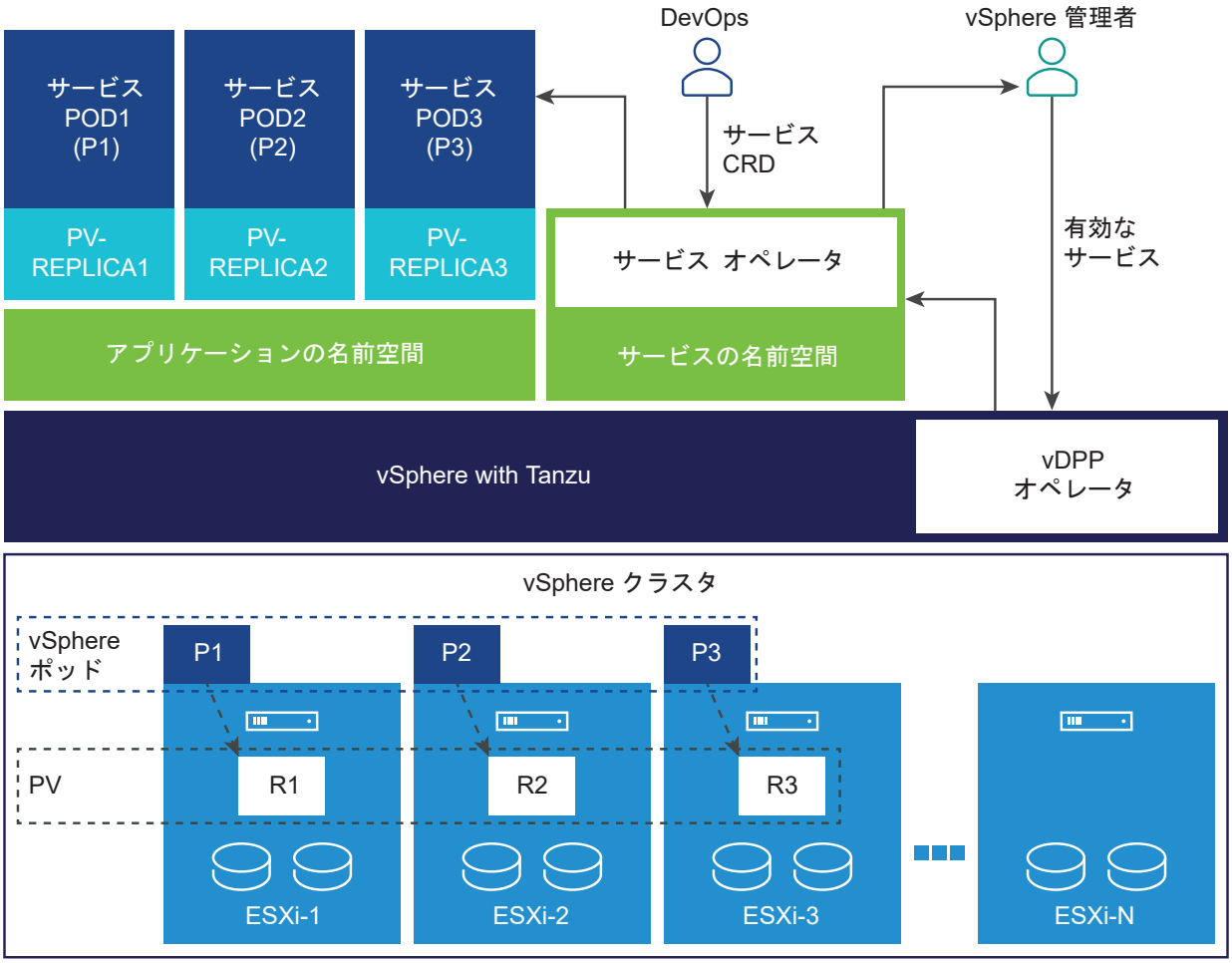
使用する vSAN のタイプを決定する場合は、次の一般的な推奨事項に従ってください。

- クラウド ネイティブのステートフル アプリケーションで物理インフラストラクチャを他の通常の仮想マシンまたは Kubernetes ワークロードと共有する場合は、SNA を使用する vSAN を使用します。各ワークロードでは、独自のストレージ ポリシーを定義できます。また、1 つのクラスタで両方の環境のメリットを受け取ることができます。
- Shared Nothing クラウド ネイティブ サービス用に専用のハードウェア クラスタを作成する場合は、vSAN Direct を使用します。

vSAN データ パーシステンス プラットフォーム オペレータ

vSAN データ パーシステンス プラットフォーム (vDPP) オペレータは、vSphere と統合されたパートナー ステートフル サービスを実行および管理するコンポーネントです。vDPP オペレータは、使用可能なステートフル サービスを vSphere 管理者に公開します。vSphere 管理者が、MinIO などのパーシステント サービスを有効にすると、vDPP オペレータはスーパーバイザー クラスタにこのサービスのアプリケーション固有のオペレータをデプロイします。

アプリケーション固有のオペレータはサードパーティから提供され、vDPP に準拠している必要があります。通常、オペレータが提供する CRD によって、Kubernetes ユーザーがインスタンスをインスタンス化するためのセルフサービス インターフェイスを利用できます。vSphere with Tanzu は、このオペレータと CRD を使用して新しいサービス インスタンスをプロビジョニングし、ステートフル サービス レイヤーを通して管理および監視します。これらのオペレータの多くは、ステートフル セットを使用してインスタンスをデプロイします。



vSphere 管理者がサービスを有効にすると、次の処理が行われます。

- vDPP オペレータにより、サービス固有のオペレータが有効になります。
- サービス固有のオペレータにより、ユーザー インターフェイス プラグインが登録されます。
- ストレージ最適化のストレージ ポリシーが作成されます。

vSAN データ パーシステンス プラットフォームの構成の上限

VMware は、VMware 構成の上限ツールで構成上の制限についての情報を提供しています。

vSAN データ パーシステンス の最大値	制限
vSAN データ パーシステンス プラットフォームあたりのパーシステント ボリュームの最大数	1000
vSAN データ パーシステンス プラットフォームのサービス インスタンスあたりのパーシステント ボリュームの最大数	60～80

vSAN Direct のストレージ デバイスへのタグ付け

VMware Cloud Foundation デプロイ環境では、vSAN は ESXi ホスト上のすべてのローカル ストレージ デバイスを自動的に要求します。デバイスを通常の vSAN の対象外にしたり、vSAN Direct に使用できるようにしたりできます。

このトピックでは、`esxcli` コマンドを使用してデバイスを vSAN Direct としてマークする方法について説明します。スクリプトを使用することもできます。スクリプトを使用した vSAN Direct のストレージ デバイスのタグ付けを参照してください。

手順

1 vSAN Direct のローカル ストレージ デバイスにタグ付けします。

```
esxcli vsan storage tag add -d diskName -t vsanDirect
```

次に例を示します。

```
esxcli vsan storage tag add -d mpx.vmhba0:C0:T1:L0 -t vsanDirect
```

デバイスは、通常の vSAN の対象外になります。

2 デバイスから vSAN Direct タグを削除します。

```
esxcli vsan storage tag remove -d diskName -t vsanDirect
```

次に例を示します。

```
esxcli vsan storage tag remove -d mpx.vmhba0:C0:T1:L0 -t vsanDirect
```

スクリプトを使用した vSAN Direct のストレージ デバイスのタグ付け

VMware Cloud Foundation 環境では、スクリプトを使用して、ESXi ホストに接続されている HDD デバイスにタグを付けることができます。スクリプトを実行すると、デバイスは通常の vSAN の対象外になり、vSAN Direct で使用できるようになります。

```
#!/usr/bin/env python3

# Copyright 2020 VMware, Inc. All rights reserved.

# Abstract
#
#    This script helps manage tagging of Direct Attached HDD disks
#    on ESXi systems for vSAN Direct in preparation for a VCF deployment.
#
#    It is expected to be used with ESX systems of version 7.0.1 or later.
#

import argparse
from enum import Enum
import logging
import sys
import os
import paramiko
import subprocess
```

```
import traceback
import ast
import getpass
from six.moves import input
from distutils.util import strtobool
from argparse import ArgumentParser

class ParseState(Enum):
        OPEN = 0
        DEVICE = 1

class RemoteOperationError(Exception):
    pass

class EsxVersion:

    def __init__(self, major, minor, release):
        self.major = major
        self.minor = minor
        self.release = release

    def __str__(self):
        return '{}.{}.{}'.format(self.major, self.minor, self.release)

    @staticmethod
    def build(str):
        tokens = str.split(b'.',3)
        return EsxVersion(int(tokens[0]), int(tokens[1]), int(tokens[2]))

class StorageDevice:

    def __init__(self, deviceId, isSSD, isVsanDirectEnabled):
        self.deviceId = str(deviceId.decode())
        self.isSSD = isSSD
        self.isVsanDirectCapable = True
        self.isVsanDirectEnabled = isVsanDirectEnabled

    def __str__(self):
        return '{}:\n\tIs SSD: {}\n\tvsanDirect enabled:{}'.format(
                    self.deviceId,
                    self.isSSD,
                    self.isVsanDirectEnabled)

    @staticmethod
    def strToBool(v):
        return bool(strtobool(str(v.decode())))

    @staticmethod
    def build(deviceId, props):
        vsanDirectEnabled = False
        isLocal = StorageDevice.strToBool(props[b'Is Local'])
        status = props[b'Status']
        isOffline = StorageDevice.strToBool(props[b'Is Offline'])
        isSSD = StorageDevice.strToBool(props[b'Is SSD'])
        isBootDevice = StorageDevice.strToBool(props[b'Is Boot Device'])
```

```
        deviceType = props[b'Device Type']
        if deviceType == b'Direct-Access' and isLocal and (not isOffline) and (not
isBootDevice) and status == b'on':
            return StorageDevice(deviceId, isSSD, vsanDirectEnabled)
        else:
            print("Skipping device {}".format(deviceId))
            return None

def parse_arguments():
    """
    Parses the command line arguments to the function
    """
    parser = argparse.ArgumentParser()
    parser.add_argument('--hostname', dest='hostname',
            help='specify hostname for the ESX Server', required=True)
    parser.add_argument('--username', dest='username',
            help='specify username to connect to the ESX Server', required=True)
    parser.add_argument('--password', dest='password',
            help='specify password to connect to the ESX Server', required=False)
    return parser.parse_args()

def get_esx_version(sshClient):
    global logger
    stdin_, stdout_, stderr_ = sshClient.exec_command('vmware -v')
    exit_status = stdout_.channel.recv_exit_status()
    if exit_status != 0:
        logger.error('Command exited with non-zero status: %s' % exit_status)
        logger.error('Error message: %s' % stderr_.read())
        raise RemoteOperationError('Failed to determine ESX version')
    output = stdout_.read()
    tokens = output.split()
    if len(tokens) < 3:
        raise RemoteOperationError('Invalid ESX Version - %s', output)
    return EsxVersion.build(tokens[2])

def check_esx_version(esxVersion):
    return esxVersion.major >= 7 and esxVersion.minor >= 0 and esxVersion.release >= 1

def query_devices(sshClient):
    global logger
    stdin_, stdout_, stderr_ = sshClient.exec_command('esxcli storage core device list')
    exit_status = stdout_.channel.recv_exit_status()
    if exit_status != 0:
        logger.error('Command exited with non-zero status: %s' % exit_status)
        logger.error('Error message: %s' % stderr_.read())
        raise RemoteOperationError('Failed to query core storage device list')
    output = stdout_.read()
    # Build the device list from the output
    return create_device_list(output)

def create_device_list(str):
    devices = []

    deviceId=""
    deviceProps={}
```

```
    parseState = ParseState.OPEN
    for line in str.splitlines():
        if parseState == ParseState.OPEN:
            if line.strip():
                deviceId=line.strip()
                parseState = ParseState.DEVICE
        elif parseState == ParseState.DEVICE:
            if line.strip():
                props = line.strip().split(b':',1)
                deviceProps[props[0]] = props[1].strip()
            else:
                if deviceId:
                    device = StorageDevice.build(deviceId, deviceProps)
                    if device:
                        devices.append(device)
                    else:
                        logger.debug("Skipping device {}".format(deviceId))
                deviceId=""
                deviceProps={}
                parseState = ParseState.OPEN
    if deviceId:
        device = StorageDevice.build(deviceId, deviceProps)
        if device:
            devices.append(device)
    return devices

def tag_device_for_vsan_direct(sshClient, deviceId):
    global logger
    logger.info("Tagging device [{}] for vSAN Direct".format(deviceId))
    command = "esxcli vsan storage tag add -d " + deviceId + " -t vsanDirect"
    stdin_, stdout_, stderr_ = sshClient.exec_command(command)
    exit_status = stdout_.channel.recv_exit_status()
    if exit_status != 0:
        logger.error('Command exited with non-zero status: %s' % exit_status)
        logger.error('Error message: %s' % stderr_.read())
        raise RemoteOperationError('Failed to tag device [{}] for vSAN
Direct'.format(deviceId))
    logger.info('Successfully tagged device [{}] for vSAN Direct'.format(deviceId))

def untag_device_for_vsan_direct(sshClient, deviceId):
    global logger
    logger.info("Untagging device [{}] for vSAN Direct".format(deviceId))
    command = "esxcli vsan storage tag remove -d " + deviceId + " -t vsanDirect"
    stdin_, stdout_, stderr_ = sshClient.exec_command(command)
    exit_status = stdout_.channel.recv_exit_status()
    if exit_status != 0:
        logger.error('Command exited with non-zero status: %s' % exit_status)
        logger.error('Error message: %s' % stderr_.read())
        raise RemoteOperationError('Failed to untag device [{}] for vSAN
Direct'.format(deviceId))
    logger.info('Successfully untagged device [{}] for vSAN Direct'.format(deviceId))

def get_vsan_info_for_device(sshClient, deviceId):
    global logger
```

```
    command = "vdq -q -d {}".format(deviceId)
    stdin_, stdout_, stderr_ = sshClient.exec_command(command)
    exit_status = stdout_.channel.recv_exit_status()
    if exit_status != 0:
        logger.error('Command exited with non-zero status: %s' % exit_status)
        logger.error('Error message: %s' % stderr_.read())
        raise RemoteOperationError('Failed to query vsan direct status on device [%s]' %
deviceId)
    output = stdout_.read()
    return ast.literal_eval(str(output.decode()))

def update_vsan_direct_status(sshClient, devices):
        for device in devices:
            vsanInfo = get_vsan_info_for_device(sshClient, device.deviceId)
            device.isVsanDirectEnabled = vsanInfo[0]['IsVsanDirectDisk'].strip() == "1"
            device.isVsanDirectCapable = vsanInfo[0]['State'].strip() == 'Eligible for use by
VSAN'

def getVsanDirectCapableDevices(devices):
    selectDevices = []
    # Cull devices incapable of vSAN Direct
    for device in devices:
        if device.isVsanDirectCapable:
            selectDevices.append(device)
    return selectDevices

def print_devices(devices):
    print("Direct-Attach Devices:")
    print("======================")
    iDevice = 0
    for device in devices:
        iDevice = iDevice + 1
        print ("{}. {}".format(iDevice, device))
    print("======================")

def tag_devices(sshClient, devices):
    for device in devices:
        tag_device_for_vsan_direct(sshClient, device.deviceId)

def untag_devices(sshClient, devices):
    for device in devices:
        untag_device_for_vsan_direct(sshClient, device.deviceId)

def tag_all_hdd_devices(sshClient, devices):
    hddDevices = []
    for device in devices:
        if not device.isSSD:
            hddDevices.append(device)
    if len(hddDevices) > 0:
        tag_devices(sshClient, hddDevices)

def show_usage():
    print ("====================================")
    print ("commands: {tag-all-hdd, tag, untag}")
    print ("\ttag <comma separated serial numbers of devices>")
```

```
    print ("\tuntag <comma separated serial numbers of devices>")
    print ("\ttag-all-hdd")
    print ("=============")

def main():
    global logger
    logger.info('Tag disks for vSAN Direct')

    try:
        # Parse arguments
        args = parse_arguments()

        # 1. Setup SSH connection to ESX system
        sshClient = paramiko.SSHClient()
        sshClient.load_system_host_keys()
        sshClient.set_missing_host_key_policy(paramiko.AutoAddPolicy())
        passwd = args.password
        if passwd == None:
            passwd = getpass.getpass(prompt='Password: ')
        logger.info('Connecting to ESX System (IP: %s)' % args.hostname)
        sshClient.connect(args.hostname, username=args.username, password=passwd)
        # version check
        esxVersion = get_esx_version(sshClient)
        print('ESX Version on {} is {}'.format(args.hostname, esxVersion))
        logger.info('Checking ESX Version...')
        if not check_esx_version(esxVersion):
            raise Exception('ESX Version must be 7.0.1 or greater')

        print ('This script helps tag direct-attached disks for vSAN Direct on ESX')
        print ('Note: Only disks of type HDD are supported at this time.')
        print ()
        print ("For help, type help")
        show_usage()

        while True:
            # get device list
            print("Querying devices...")
            devices = query_devices(sshClient)
            # update devices with vSAN Direct status
            update_vsan_direct_status(sshClient, devices)
            # cull device list
            selectDevices = getVsanDirectCapableDevices(devices)
            # List the devices for the user to see
            print_devices(selectDevices)
            # find out what the user wants to do to these devices
            args = input('Command> ').split()
            if len(args) == 0:
                break
            cmd = args[0]
            if cmd == 'q' or cmd == 'quit' or cmd == 'exit':
                break
            elif cmd == 'help':
                show_usage()
            elif cmd == 'tag-all-hdd':
                print("Tagging all HDD devices...")
```

```
                tag_all_hdd_devices(sshClient, selectDevices)
            elif cmd == 'tag' or cmd == 'untag':
                chosenDevices = []
                if len(args) > 1:
                    serials = args[1].split(',')
                    for serialStr in serials:
                        serial = int(serialStr)
                        if serial < 1 or serial > len(selectDevices):
                            raise Exception("Error: Serial {} is out of range".format(serial))
                        chosenDevices.append(selectDevices[serial-1])
                if len(chosenDevices) == 0:
                    print("No devices specified")
                    continue
                if cmd == 'tag':
                    print("Tagging devices...")
                    tag_devices(sshClient, chosenDevices)
                else:
                    print("Untagging devices...")
                    untag_devices(sshClient, chosenDevices)
            else:
                print ("Error: Unrecognized command - %s" % cmd)
    except paramiko.ssh_exception.AuthenticationException as e:
        logger.error(e)
        sys.exit(5)
    except Exception as e:
        logger.error('Disk tagging failed with error: %s' % e)
        logger.error(traceback.format_exc())
        sys.exit(1)
    finally:
        # Close SSH client
        try:
            sshClient.close()
        except:
            pass

# Set up logging
logging.basicConfig()
logger = logging.getLogger('tag-disks-for-vsan-direct')

if __name__ == "__main__":
    main()
```

vSphere with Tanzu 用の vSAN Direct の設定

vSphere 管理者として、vSAN データ パーシステンス プラットフォームで使用する vSAN Direct を設定します。ESXi ホストに対してローカルである未要求ストレージ デバイスを使用します。

前提条件

vSAN が環境内のすべてのローカル ストレージ デバイスを自動的に要求する場合は、`esxcli` コマンドを使用して、vSAN Direct に使用できるデバイスにタグを付けます。vSAN Direct のストレージ デバイスへのタグ付けを参照してください。スクリプトを使用することもできます。詳細については、『スクリプトを使用した vSAN Direct のストレージ デバイスのタグ付け』を参照してください。

手順

1 vSphere Client で、vSAN クラスタに移動します。

2 [設定] タブをクリックします。

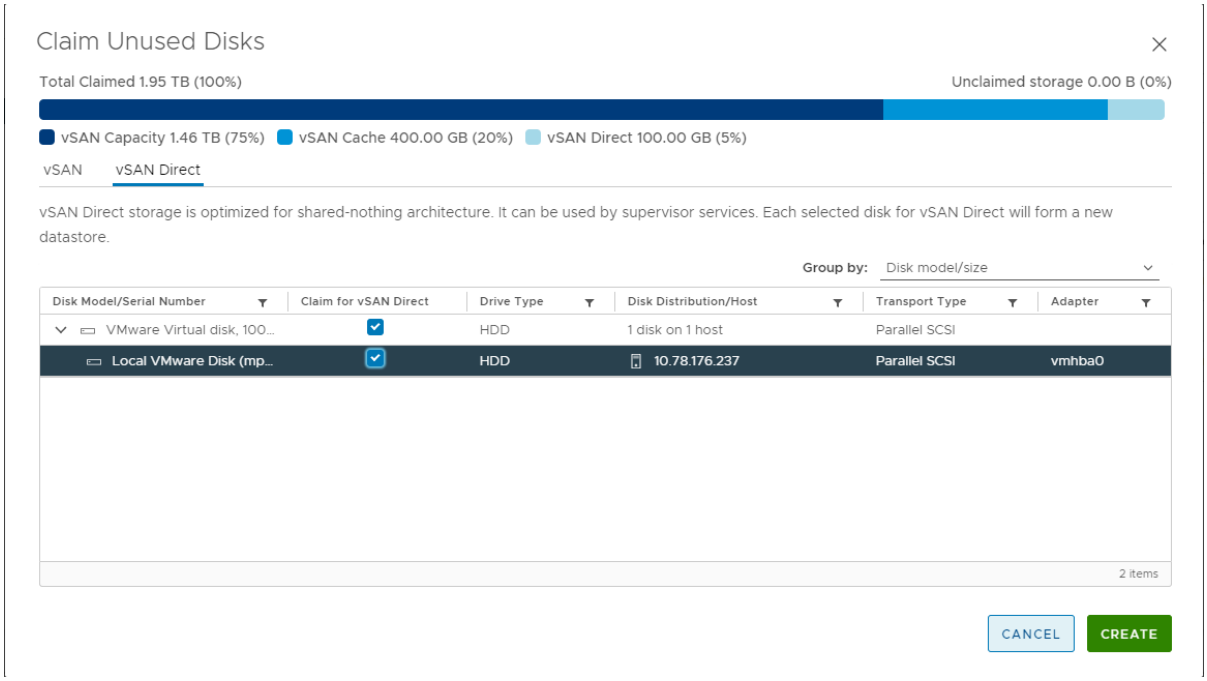
3 [vSAN] の下で、[ディスク管理] をクリックします。

4 [未使用のディスクの要求] をクリックします。

5 [未使用のディスクの要求] ダイアログ ボックスで、[vSAN Direct] タブをクリックします。

6 要求するデバイスを選択し、[vSAN Direct の要求] 列のチェック ボックスを選択します。

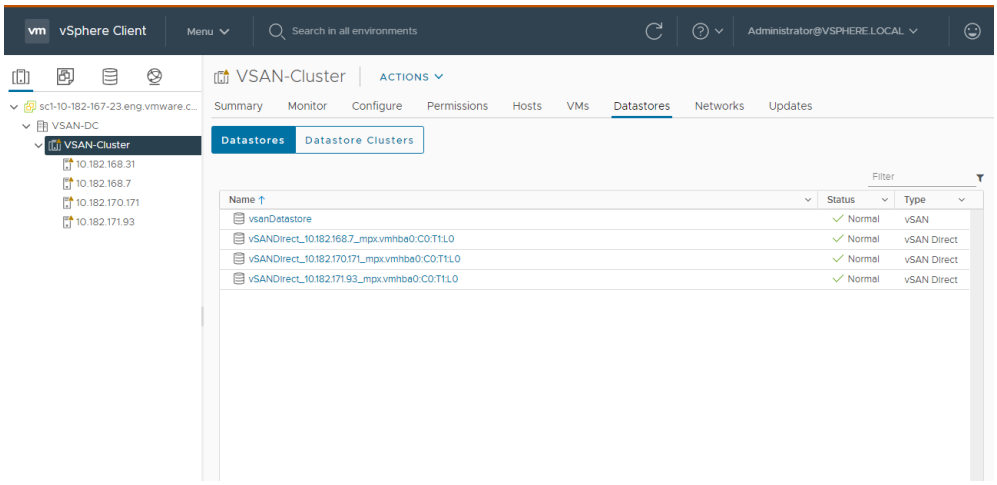
注： 通常の vSAN データストアのデバイスを要求した場合、これらのデバイスは [vSAN Direct] タブには表示されません。



7 [作成] をクリックします。

要求されるデバイスごとに、vSAN Direct は新しいデータストアを作成します。

8 [データストア] タブをクリックして、クラスタ内のすべての vSAN Direct データストアを表示します。



次のステップ

vSAN Direct と外部ストレージを併用できます。詳細については、『vSAN Direct と外部ストレージの併用』を参照してください。

vSphere with Tanzu でのステートフル サービスの有効化

vSphere with Tanzu は、パーシステンス ストレージのニーズを満たすために vSAN データ パーシステンス プラットフォームを使用するいくつかのサードパーティのサービスと連携します。vSphere 管理者は、vCenter Server でサービスを有効にします。

vSphere with Tanzu 7.0 Update 3 リリース以降では、使用可能なサードパーティ サービスを VMware のサポート リポジトリからダウンロードできます。

ステートフル サービスを有効にするには、まず、サービスを記述しているダウンロード済みの YAML ファイルを使用して、vCenter Server にサービスを登録します。次に、DevOps エンジニアがそのサービスを Kubernetes ワークロードで使用できるように、サービスを スーパーバイザー クラスタ にインストールします。

前提条件

- 必要な権限：Supervisor Services.Manage Supervisor Services
- スーパーバイザー クラスタ で NSX-T Data Center ネットワーク スタックを使用していることを確認します。vSAN データ パーシステンス プラットフォームでは、vSphere Distributed Switch (vDS) ネットワークはサポートされていません。

  NSX-T の設定については、vSphere with Tanzu 用 NSX-T Data Center の構成を参照してください。
- VMware によって管理されているリポジトリから、パートナー サービスの YAML ファイルをダウンロードします。

  サービスの YAML ファイルをダウンロードするときは、使用している vSphere のバージョンと互換性のある正しいサービス バージョンを使用してください。

パートナー サービスの以前のバージョンである MinIO および Cloudian Hyperstore をインストールしてある場合は、vSphere をバージョン 7.0 Update 3 にアップグレードした後、互換性のあるバージョンにアップグレードします。パートナー オペレータの新しいバージョンでは、いくつかの問題が修正され、新しいプラットフォーム機能が使用されています。詳細については、パートナーのドキュメントを参照してください。

表 10-1. vSphere とパートナー サービスの互換性マトリックス

vSphere のバージョン	パートナー サービス	サービス バージョン	Kubernetes バージョン
vSphere 7.0 Update 3	MinIO	2.0.0	1.19、1.20、1.21
	Cloudian	1.2.0	1.19、1.20、1.21

次のいずれかの方法で、YAML ファイルをダウンロードします。

- https://vmwaresaas.jfrog.io/ リポジトリで、[Artifacts] - [vDPP-Partner-YAML] 内の適切なパートナー フォルダに移動し、ダウンロードする YAML ファイルを選択します。

  パートナー YAML の最新バージョンは、パートナー ディレクトリのトップ レベルにあります。

- `wget` または `curl` コマンドを使用して YAML ファイルをダウンロードします。

  例：

```
wget https://vmwaresaas.jfrog.io/artifactory/vDPP-Partner-YAML/Cloudian/Hyperstore/
SupervisorService/hyperstore-supervisorservice.yaml
```

手順

1 vSAN または vSAN Direct ストレージを構成します。

vSAN ストレージの設定の詳細については、VMware vSAN の管理を参照してください。vSAN Direct を設定するには、vSphere with Tanzu 用の vSAN Direct の設定を参照してください。

vSAN Direct データストアは、Kubernetes ではストレージ プールとして表示されます。

2 ステートフル サービスを vCenter Server システムに追加します。

VMware で管理されているリポジトリからダウンロードしたパートナー サービス YAML ファイルを使用します。

vCenter Server への スーパーバイザー サービス の追加を参照してください。

3 サービスを スーパーバイザー クラスタ にインストールします。

スーパーバイザー クラスタ への スーパーバイザー サービス のインストールを参照してください。

サービスを有効にすると、vSAN データ パーシステンス プラットフォームは次のアクションを実行して、サービスに必要なリソースを作成します。

- スーパーバイザー クラスタ内にこのサービスの名前空間を作成します。

- vSAN Shared-Nothing-Architecture (SNA) データストアおよび vSAN Direct データストアで使用されるデフォルトのストレージ ポリシーおよび対応するストレージ クラスを作成します。

  注： vSphere 管理者がステートフル サービスを有効にすると、vSAN データ パーシステンス プラットフォームは名前空間に vsan-direct および vsan-sna ストレージ クラスを自動的に作成します。スーパーバイザー クラスタ で実行されているアプリケーションのみが、vsan-direct および vsan-sna ストレージ クラスを使用できます。これらのストレージ クラスは、Tanzu Kubernetes クラスタ内では使用できません。

  vSphere 7.0 Update 2 以降では、vSAN Direct ストレージ ポリシーは機能ベースです。vSphere 7.0 Update 1 でタグベースのポリシーを作成した場合は、vSphere 7.0 Update 2 以降にアップグレードした後に、自動的に機能ベースに変換されます。

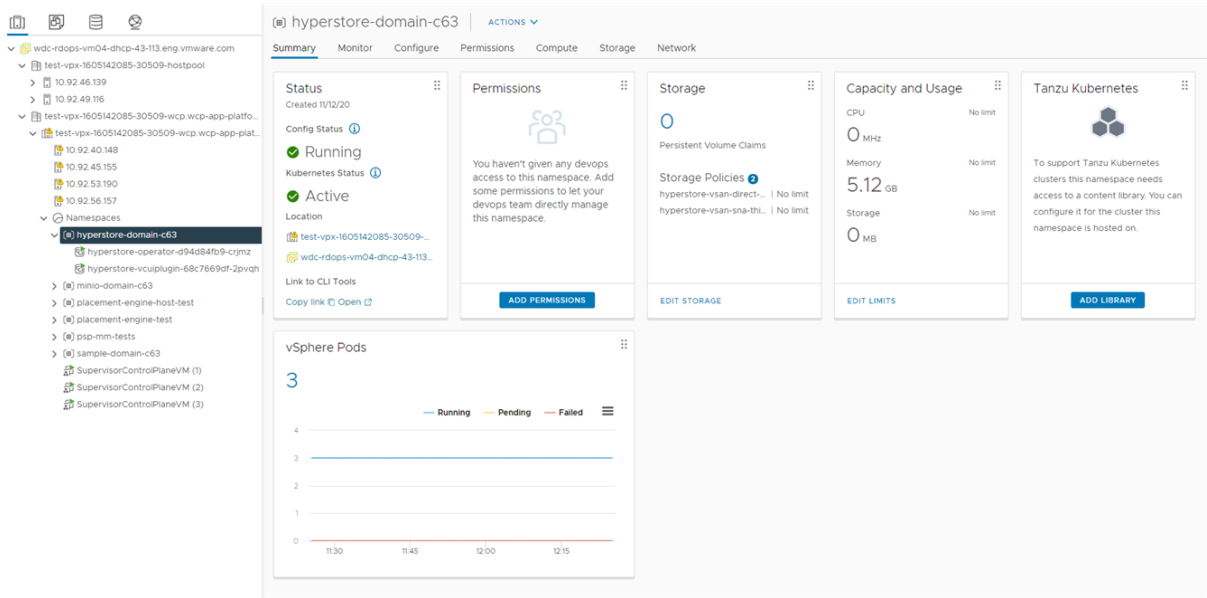
  デフォルトを使用せずに、新しいストレージ ポリシーを作成してサービス名前空間に割り当てる場合は、vSAN Direct ストレージ ポリシーの作成および vSAN SNA ストレージ ポリシーの作成を参照してください。

- 編集権限と表示権限を持つロールなどの DevOps ロールを作成します。

  サービス オペレータがデプロイされている場合は、そのカスタム CRD が スーパーバイザー クラスタ にインストールされています。編集権限を持つユーザーは、名前空間内でこれらの CRD のリソースに対して CRUD を実行することができます。表示権限を持つユーザーのみが、この CRD のリソースを表示できます。

- サードパーティからカスタム ユーザー インターフェイス プラグインが提供されている場合は、そのプラグインが vSphere Client に表示されます。vSphere 管理者は、プラグインを使用してサービスを管理できます。

4 サービス用に作成された名前空間を選択し、[サマリ] タブをクリックして、サービスに該当するすべてのリソースが作成されていることを確認します。



次のステップ

- DevOps エンジニアは、`kubectl` コマンドを使用してサービスの名前空間にアクセスし、サードパーティの CRD を使用して、サードパーティ アプリケーション サービスのインスタンスをデプロイします。詳細については、サードパーティのドキュメントを参照してください。

  ステートフル サービスに使用する名前空間に適切なストレージ クラスがあることを確認するには、ステートフル サービスで使用可能なストレージ ポリシーの確認を参照してください。

- サードパーティからカスタム ユーザー インターフェイス プラグインが提供されている場合、vSphere 管理者はこれらのプラグインを使用してサービスを管理および監視できます。

  詳細については、サードパーティ ユーザー インターフェイス プラグインのドキュメントを参照してください。また、vSphere 管理者は Skyline Health チェックを使用して、サービスを監視できます。vSphere with Tanzu でのステートフル サービスの監視を参照してください

vSphere with Tanzu でのステートフル サービスの監視

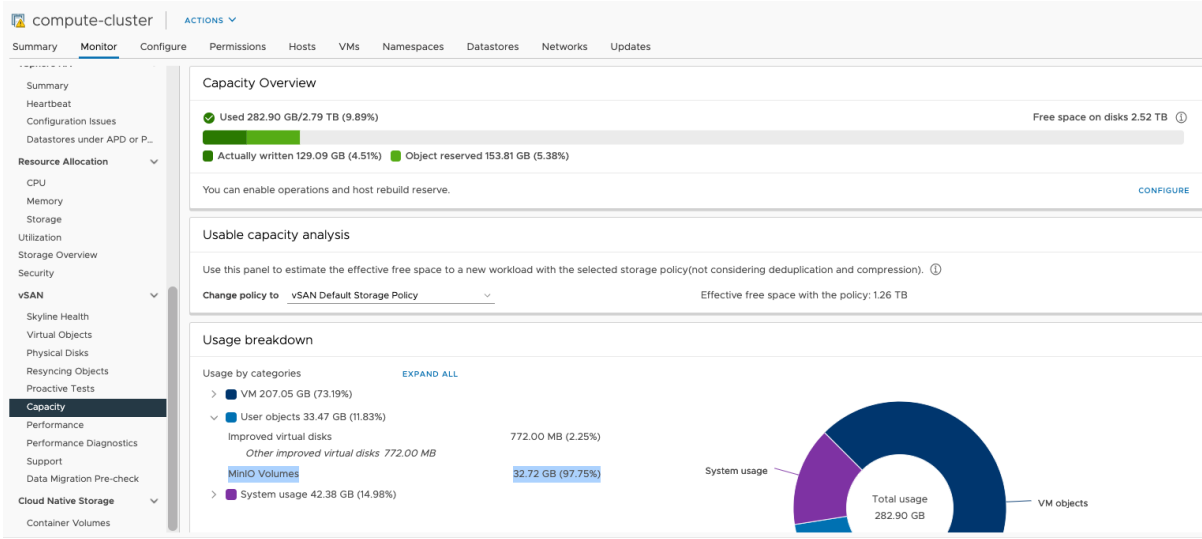
統合されたサードパーティ ステートフル サービスを有効にした後、vSAN の健全性機能とキャパシティ監視機能を使用して、サービス オブジェクトのステータスを表示し、容量の使用率を分析します。

手順

1. vSphere Client で、スーパーバイザー クラスタ に移動します。
2. [監視] タブをクリックします。
3. 有効なサービスに対応する名前空間内で実行される仮想オブジェクトを監視します。
   a. [vSAN] で [仮想オブジェクト] をクリックします。

      仮想オブジェクト（MinIO オペレータ オブジェクトなど）を参照して、状態を確認することができます。

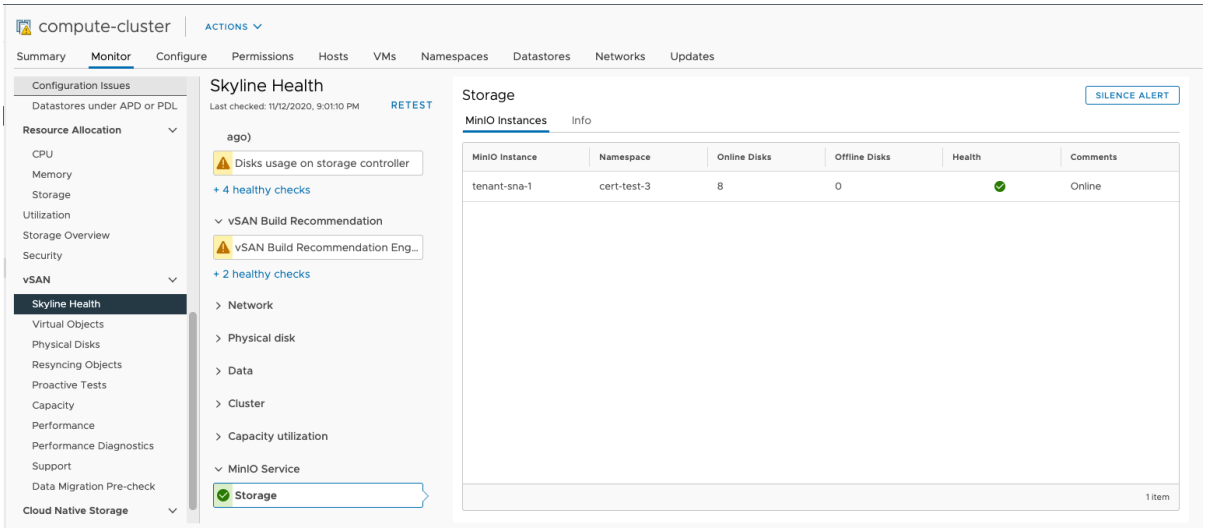
   b. 物理インフラストラクチャ全体のオブジェクト配置を確認するには、特定のオブジェクトを選択し、[配置の詳細の表示] をクリックします。
4. サービス オブジェクトで使用される容量を監視します。
   a. [vSAN] で [容量] をクリックします。
   b. [使用量の内訳] ペインの[ユーザー オブジェクト] にサービス オブジェクトを表示します。



5 サービス インスタンスの健全性を監視します。

a [vSAN] で [Skyline Health] を選択します。

b サービス健全性チェックを個別に選択すると、詳細情報が表示されます。



ステートフル サービスで使用可能なストレージ ポリシーの確認

DevOps エンジニアとして、vSphere with Tanzu 環境内のステートフル サービスで使用する名前空間に適切なストレージ クラスがあることを確認します。ストレージ クラスには、vSAN Shared-Nothing-Architecture (SNA) と vSAN Direct を指定できます。

vSphere 管理者がステートフル サービスを有効にすると、vSAN データ パーシステンス プラットフォームは名前空間にこれらのストレージ クラスを自動的に作成します。vSphere with Tanzu でのステートフル サービスの有効化を参照してください。

注： スーパーバイザー クラスタ で実行されているアプリケーションのみが、vsan-direct および vsan-sna ストレージ クラスを使用できます。これらのストレージ クラスは、Tanzu Kubernetes クラスタ内では使用できません。

デフォルトのストレージ クラスのほかに、vSphere 管理者はカスタム ストレージ ポリシーを作成して、名前空間に割り当てることもできます。vSAN Direct ストレージ ポリシーの作成と vSAN SNA ストレージ ポリシーの作成を参照してください。

手順

- vSAN SNA と vSAN Direct で使用するストレージ ポリシーが名前空間で使用できることを確認します。

```
# kubectl get sc
NAME                        PROVISIONER              RECLAIMPOLICY   VOLUMEBINDINGMODE
ALLOWVOLUMEEXPANSION   AGE
sample-vsan-direct-thick    csi.vsphere.vmware.com   Delete          WaitForFirstConsumer
true                   3m36s
sample-vsan-sna-thick       csi.vsphere.vmware.com   Delete          WaitForFirstConsumer
true                   13m
```

vSAN SNA ストレージ ポリシーの作成

vSAN を vSAN データ パーシステンス プラットフォームで使用する場合、ステートフル サービスが実行される名前空間と使用される vSAN Shared Nothing Architecture (SNA) ストレージ ポリシーを作成できます。

手順

1. vSphere Client で、[仮想マシン ストレージ ポリシーの作成] ウィザードを開きます。
   a. [ホーム] メニューで、[ポリシーおよびプロファイル] をクリックします。
   b. [ポリシーおよびプロファイル] で、[仮想マシン ストレージ ポリシー] をクリックします。
   c. [作成] をクリックします。
2. ポリシーの名前と説明を入力します。

オプション	操作
vCenter Server	vCenter Server インスタンスを選択します。
名前	ストレージ ポリシーの名前（**Sample SNA Thick** など）を入力します。
説明	ストレージ ポリシーの説明を入力します。

3. [データストア固有のルール] の [ポリシー構造] ページで、vSAN ストレージ配置のルールを有効にします。

4 [vSAN] ページで、[可用性] タブをクリックし、以下の値を選択します。これらの値は、vSAN データ パーシステンス プラットフォームの SNA ワークロードにのみ適用可能です。仮想マシン ワークロードのプロビジョニングには使用できません。

オプション	説明
オプション	値
サイトの耐障害性	[なし - 標準クラスタ] **注：** vSAN データ パーシステンス プラットフォームでは、標準クラスタのみがサポートされます。
許容される障害の数	[ホスト アフィニティを使用したデータの冗長性なし]

シック プロビジョニングは SNA ワークロードに適用され、[詳細なポリシー ルール] タブでオブジェクト容量の予約の値として選択されます。この値は変更できません。

5 [ストレージ互換性] ページでこのポリシーに適合する vSAN データストアのリストを確認します。

6 [確認して完了] ページでポリシーの設定を確認し、[完了] をクリックします。

設定を変更するには、[戻る] をクリックして関連するページに移動します。

次のステップ

作成したポリシーは、ステートフル サービスが実行される名前空間に割り当てることができます。名前空間のストレージ設定の変更を参照してください。

vSAN Direct ストレージ ポリシーの作成

vSAN Direct を vSAN データ パーシステンス プラットフォームで使用する場合、ステートフル サービスが実行される名前空間と使用される機能ベースのストレージ ポリシーを作成できます。

手順

1 vSphere Client で、[仮想マシン ストレージ ポリシーの作成] ウィザードを開きます。

a [ホーム] メニューで、[ポリシーおよびプロファイル] をクリックします。

b [ポリシーおよびプロファイル] で、[仮想マシン ストレージ ポリシー] をクリックします。

c [作成] をクリックします。

2 ポリシーの名前と説明を入力します。

オプション	操作
vCenter Server	vCenter Server インスタンスを選択します。
名前	ストレージ ポリシーの名前（**`Sample vSAN Direct Thick`** など）を入力します。
説明	ストレージ ポリシーの説明を入力します。

3 [データストア固有のルール] の [ポリシー構造] ページで、vSAN Direct ストレージ配置のルールを有効にします。

4 [vSAN Direct ルール] ページで、ストレージ配置タイプとして vSAN Direct を指定します。

5 [ストレージ互換性] ページでこのポリシーに適合する vSAN Direct データストアのリストを確認します。

6 [確認して完了] ページでポリシーの設定を確認し、[完了] をクリックします。

設定を変更するには、[戻る] をクリックして関連するページに移動します。

次のステップ

作成したポリシーは、ステートフル サービスが実行される名前空間に割り当てることができます。名前空間のストレージ設定の変更を参照してください。

vSphere ポッド へのワークロードのデプロイ

11

DevOps エンジニアは、スーパーバイザー クラスタ で実行されている名前空間のリソースの境界内で vSphere ポッド をデプロイし、そのライフサイクルを管理できます。名前空間に vSphere ポッド をデプロイするには、その名前空間に対する書き込み権限が必要です。

この章には、次のトピックが含まれています。

- スーパーバイザー クラスタ コンテキストの取得と使用
- vSphere 名前空間 での vSphere ポッド へのアプリケーションのデプロイ
- 組み込みの Harbor レジストリ を使用した vSphere ポッド へのアプリケーションのデプロイ
- vSphere ポッド アプリケーションのスケーリング
- 機密性の確保された vSphere ポッド のデプロイ

スーパーバイザー クラスタ コンテキストの取得と使用

vSphere 管理者から スーパーバイザー クラスタ 上の Kubernetes 制御プレーンの IP アドレスが提供されたら、スーパーバイザー クラスタ にログインして、アクセス権のあるコンテキストを取得できます。コンテキストは スーパーバイザー クラスタ の名前空間に対応します。

前提条件

- vSphere 管理者から スーパーバイザー クラスタ の Kubernetes 制御プレーンの IP アドレスを入手します。
- vCenter Single Sign-On でユーザー アカウントを取得します。
- 自分が必要なコンテキストにアクセスする権限を持っていることを vSphere 管理者に確認します。
- 署名を付与する認証局 (CA) を Trust Root としてインストールするか、または証明書を Trust Root として直接追加することにより、Kubernetes 制御プレーンによって提供される証明書がシステムで信頼されることを確認します。

手順

1. ブラウザ ウィンドウで、Kubernetes 制御プレーンの URL を開きます。
2. `vsphere-plugin.zip` の SHA256 チェックサムが `sha256sum.txt` ファイルのチェックサムと一致することを確認します。

3 マシンに vsphere-plugin.zip ファイルをダウンロードし、OS の実行可能ファイルの検索パスに設定します。

4 コマンド プロンプト ウィンドウで、次のコマンドを実行して vCenter Server にログインします。

```
kubectl vsphere login --server=https://<server_adress> --vsphere-username <your user account name>
```

5 アクセスする権限のある設定コンテキストの詳細を表示するには、次の kubectl コマンドを実行します。

```
kubectl config get-contexts
```

CLI に、使用可能な各コンテキストの詳細が表示されます。

6 コンテキストを切り替えるには、次のコマンドを使用します。

```
kubectl config use-context <example-context-name>
```

結果

Kubernetes 制御プレーンのログイン API が起動します。認証プロキシによって認証要求が vCenter Single Sign-On にリダイレクトされます。vCenter Server から JSON Web トークンが返され、kubeconfig ファイルに追加されます。このトークンは、新しい kubectl コマンドでユーザーを認証するたびに Kubernetes 制御プレーンに送信されます。

vSphere 名前空間 での vSphere ポッド へのアプリケーションのデプロイ

スーパーバイザー クラスタ の名前空間にアプリケーションをデプロイできます。アプリケーションがデプロイされると、スーパーバイザー クラスタ の名前空間内に、それぞれ特定の数の vSphere ポッド が作成されます。

また、Harbor イメージ レジストリに保存されているイメージからアプリケーションをデプロイすることもできます。組み込みの Harbor レジストリ を使用した vSphere ポッド へのアプリケーションのデプロイを参照してください。

前提条件

- vSphere 管理者から スーパーバイザー クラスタ の Kubernetes 制御プレーンの IP アドレスを入手します。
- vCenter Single Sign-On でユーザー アカウントを取得します。
- 自分が必要なコンテキストにアクセスする権限を持っていることを vSphere 管理者に確認します。

手順

1 スーパーバイザー クラスタ で認証します。

vCenter Single Sign-On ユーザーとして スーパーバイザー クラスタ に接続するを参照してください。

2 アプリケーションをデプロイするコンテキストに切り替えます。

```
kubectl config use-context <namespace>
```

3 アプリケーションをデプロイします。

```
kubectl apply -f <application name>.yaml
```

組み込みの Harbor レジストリ を使用した vSphere ポッド へのアプリケーションのデプロイ

Harbor レジストリ に保存されているイメージを使用して、スーパーバイザー クラスタ の名前空間に vSphere ポッド をデプロイできます。

前提条件

- アプリケーションをデプロイする名前空間と同じ名前を持つ Harbor レジストリ 内のプロジェクトにイメージをプッシュします。組み込みの Harbor レジストリ へのイメージのプッシュ を参照してください。
- `vspere-plugin.zip` の内容を使用環境の実行ファイル パスに追加します。

手順

1 次のパラメータを含む YAML ファイルを作成します。

```
...
namespace: <namespace-name>
...
spec:
...
image: <image registry URL>/<namespace name>/<image name>
```

2 スーパーバイザー クラスタ にログインします。

```
kubectl vsphere login --server=https://<server_adress> --vsphere-username <your user
account name>
```

3 アプリケーションをデプロイする名前空間に切り替えます。

```
kubectl config use-context <namespace>
```

4 その YAML ファイルから vSphere ポッド をデプロイします。

```
kubectl apply -f <yaml file name>.yaml
```

5 次のコマンドを実行して、イメージが Harbor レジストリ からプルされており、vSphere ポッド が実行中の状態であることを確認します。

```
kubectl describe pod/<yaml name>
```

結果

作成した YAML ファイルは、名前空間に基づいて名前が付けられた Harbor レジストリ 内のプロジェクトのイメージを使用して、指定した名前空間にデプロイされます。

例：

次の YAML ファイルを作成し、Harbor レジストリ 内の demoapp1 プロジェクトの busybox イメージを使用して、その YAML ファイルを名前空間 demoapp1 にデプロイします。

```
apiVersion: v1
kind: Pod
metadata:
  name: busybox
  namespace: demoapp1
spec:
  containers:
  - name: busybox
    image: <harbor_IP>/demoapp1/busybox:latest
    command:
      - sleep
      - "3600"
    imagePullPolicy: IfNotPresent
  restartPolicy: Always
```

vSphere ポッド アプリケーションのスケーリング

スーパーバイザー クラスタ で実行されている各アプリケーションのレプリカの数をスケール アップおよびスケール ダウンできます。

前提条件

- vSphere 管理者から スーパーバイザー クラスタ の Kubernetes 制御プレーンの IP アドレスを入手します。
- vCenter Single Sign-On でユーザー アカウントを取得します。
- 自分が必要なコンテキストにアクセスする権限を持っていることを vSphere 管理者に確認します。

手順

1 スーパーバイザー クラスタ で認証します。

```
kubectl vsphere login --server <control plane load balancer IP address> --vsphere-username
<vSphere user account name>
```

2 アプリケーションをスケール アップまたはスケール ダウンします。

```
kubectl get deployments
kubectl scale deployment <deployment-name> --replicas=<number-of-replicas>
```

機密性の確保された vSphere ポッド のデプロイ

vSphere with Tanzu を使用すると、スーパーバイザー クラスタ 上で機密性の確保された vSphere ポッド を実行できます。機密性の確保された vSphere ポッド では、ゲスト OS のメモリを常に暗号化してハイパーバイザーからのアクセスから保護するハードウェア テクノロジーが使用されます。

vSphere 7.0 Update 2 以降の場合、機密性の確保された vSphere ポッド を作成するには、セキュリティ拡張として Secure Encrypted Virtualization-Encrypted State (SEV-ES) を追加する必要があります。SEV-ES により、CPU レジスタ内の情報がレジスタからハイパーバイザーなどのコンポーネントに漏洩することを防止できます。SEV-ES は、CPU レジスタの状態に対する悪意のある変更を検出することもできます。vSphere 環境での SEV-ES テクノロジーの使用の詳細については、「AMD の Secure Encrypted Virtualization -Encrypted State による仮想マシンの保護」を参照してください。

前提条件

ESXi ホストで SEV-ES を有効にするには、vSphere 管理者が次のガイドラインに従う必要があります。

- SEV-ES 機能をサポートするホストを使用します。現在、SEV-ES でサポートされているのは AMD EPYC 7xx2 CPU（コード名は Rome）およびそれ以降の CPU のみです。
- ESXi バージョン 7.0 Update 2 以降を使用します。
- ESXi システムの BIOS 構成で SEV-ES を有効にします。BIOS 構成へのアクセスの詳細については、システムのドキュメントを参照してください。
- BIOS で SEV-ES を有効にするときに、ホスト上の SEV-ES 仮想マシンと機密性の確保された vSphere ポッド の数に 1 を加えた値を [Minimum SEV non-ES ASID] の設定に入力します。たとえば、100 台の SEV-ES 仮想マシンと 128 個の vSphere ポッド を実行する計画がある場合は、229 以上の値を入力します。入力できる設定の最大値は 500 です。

手順

1 次のパラメータを含む YAML ファイルを作成します。

a 注釈で、機密性の確保された vSphere ポッド 機能を有効にします。

```
...
annotations:
        vmware/confidential-pod: enabled
...
```

b コンテナのメモリ リソースを指定します。

この例のように、メモリ要求とメモリ制限を同じ値に設定してください。

```
resources:
      requests:
        memory: "512Mi"
      limits:
        memory: "512Mi"
```

例として、次の YAML ファイルを使用します。

```
apiVersion: v1
kind: Pod
metadata:
  name: photon-pod
  namespace: my-podvm-ns
  annotations:
    vmware/confidential-pod: enabled
spec:  # specification of the pod's contents
  restartPolicy: Never
  containers:
  - name: photon
    image: wcp-docker-ci.artifactory.eng.vmware.com/vmware/photon:1.0
    command: ["/bin/sh"]
    args:    ["-c", "while true; do echo hello, world!; sleep 1; done"]
    resources:
      requests:
        memory: "512Mi"
      limits:
        memory: "512Mi"
```

2 スーパーバイザー クラスタ にログインします。

```
kubectl vsphere login --server=https://<server_adress> --vsphere-username <your user
account name>
```

3 アプリケーションをデプロイする名前空間に切り替えます。

```
kubectl config use-context <namespace>
```

4 YAML ファイルから、機密性の確保された vSphere ポッド をデプロイします。

```
kubectl apply -f <yaml file name>.yaml
```

注： vSphere ポッド がデプロイされている場合、DRS は SEV-ES をサポートする ESXi ノードに配置されます。このようなノードを使用できない場合、vSphere ポッド は失敗とマークされます。

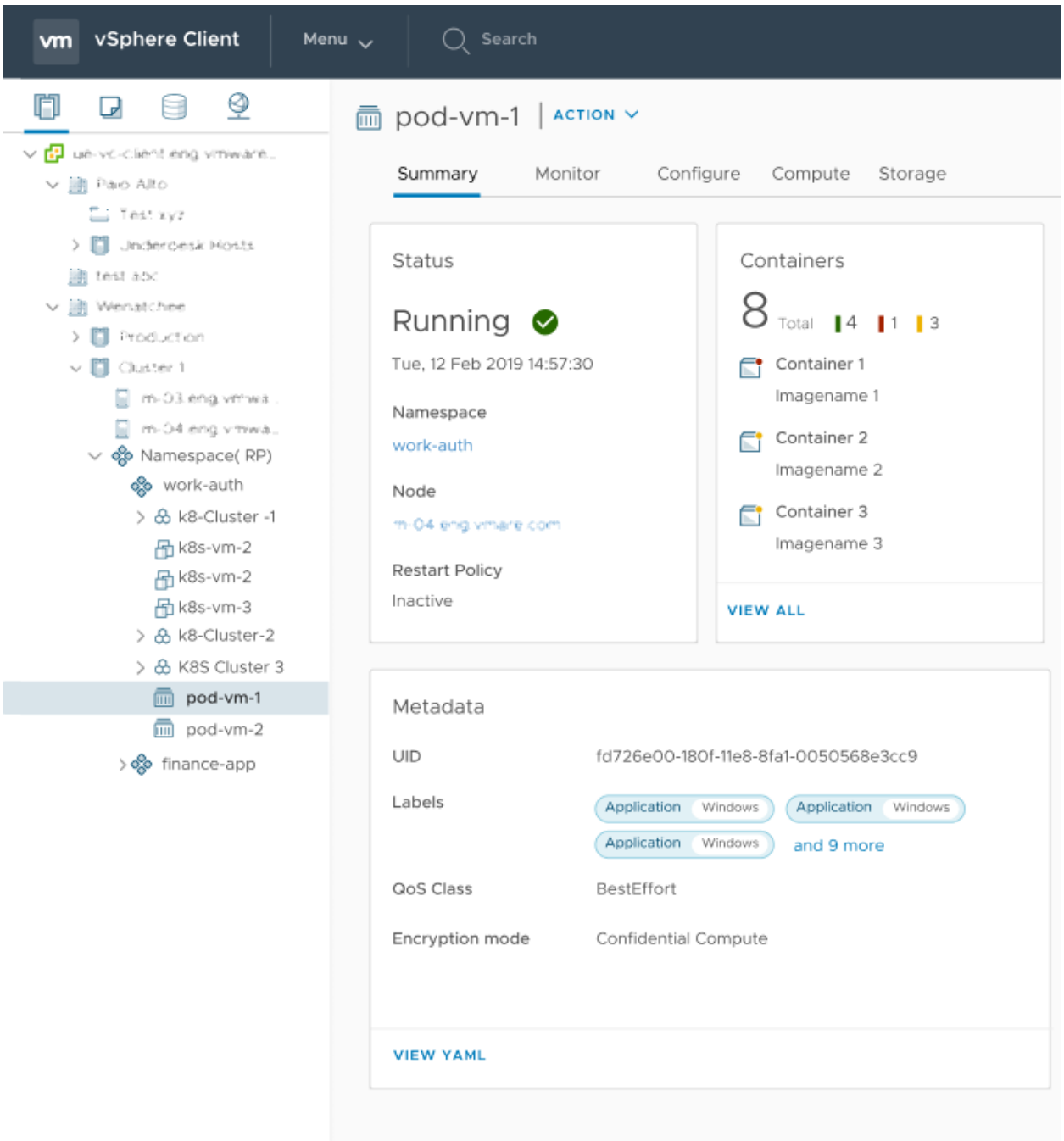
機密性の確保された vSphere ポッド を起動すると、このポッドで実行されているすべてのワークロードに対して、ハードウェア メモリ暗号化がサポートされます。

5 次のコマンドを実行して、機密性が確保された vSphere ポッド が作成されていることを確認します。

```
kubectl describe pod/<yaml name>
```

次のステップ

vSphere 管理者には、機密性の確保された vSphere ポッド が表示されます。vSphere Client では、この仮想マシンに [暗号化モード: 機密性の高いコンピューティング] というタグが付加されて表示されます。


vSphere Client
Menu
Search
pod-vm-1
ACTION
Summary
Monitor
Configure
Compute
Storage
Status
Running
Tue, 12 Feb 2019 14:57:30
Namespace
work-auth
Node
Restart Policy
Inactive
Containers
8 Total 4 1 3
Container 1
Imagename 1
Container 2
Imagename 2
Container 3
Imagename 3
VIEW ALL
Metadata
UID
fd726e00-180f-11e8-8fa1-0050568e3cc9
Labels
Application Windows
Application Windows
Application Windows
and 9 more
QoS Class
BestEffort
Encryption mode
Confidential Compute
VIEW YAML
Namespace(RP)
work-auth
k8-Cluster -1
k8s-vm-2
k8s-vm-2
k8s-vm-3
k8-Cluster-2
K8S Cluster 3
pod-vm-1
pod-vm-2
finance-app

vSphere with Tanzu での仮想マシンのデプロイと管理

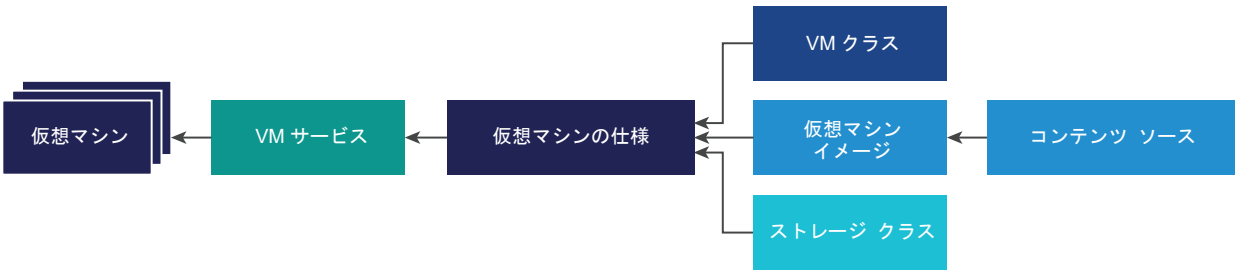
12

vSphere with Tanzu で提供されている仮想マシン サービス機能を使用すると、DevOps エンジニアは一般的な共有 Kubernetes 環境でコンテナに加え、仮想マシンをデプロイして実行することができます。この仮想マシン サービスを使用して、vSphere 名前空間 内の仮想マシンのライフサイクルを管理できます。仮想マシン サービスは、スタンドアローン仮想マシンおよび Tanzu Kubernetes クラスタを構成する仮想マシンを管理します。

一般的に、コンテナではなく仮想マシンを使用したワークロードの実行は、ビジネスのニーズや目標に応じて決定されます。仮想マシンを実行する場合の詳細については、vSphere with Tanzu での仮想マシンの使用を参照してください。

仮想マシン サービスの概念

vSphere 名前空間 にデプロイする仮想マシンの状態を記述するには、仮想マシン クラス、仮想マシン イメージ、ストレージ クラスなどのパラメータを使用します。仮想マシン サービスは、これらの仕様を組み合わせて、スタンドアローン仮想マシンまたは Tanzu Kubernetes クラスタをサポートする仮想マシンを作成します。



VM サービス

仮想マシン サービスは、仮想マシンおよび関連する vSphere リソースを管理するための Kubernetes スタイルの宣言型 API を提供する vSphere with Tanzu コンポーネントです。仮想マシン サービスを使用すると、vSphere 管理者はリソースを提供し、仮想マシン クラスや仮想マシン イメージなどのテンプレートを Kubernetes に提供できるようになります。DevOps エンジニアは、これらのリソースを使用して、仮想マシンに目標とする状態を記述できます。DevOps エンジニアが仮想マシンの状態を指定すると、仮想マシン サービスは目標とする状態を、バッキング インフラストラクチャ リソースに対して実現された状態に変換します。

仮想マシン サービスを介して作成された仮想マシンを管理するには、`kubectl` コマンドを使用して Kubernetes 名前空間から管理する必要があります。vSphere 管理者は、仮想マシンを vSphere Client から管理することはできませんが、詳細を表示し、仮想マシンで使用されるリソースを監視することはできます。詳細については、『vSphere with Tanzu で利用可能な仮想マシンの監視』を参照してください。

VM クラス

仮想マシン クラスは、仮想マシンの一連のリソースの要求に使用可能な仮想マシンの仕様です。仮想マシン クラスは vSphere 管理者によって制御および管理され、仮想 CPU の数、メモリ容量、予約設定などのパラメータを定義します。定義されたパラメータは、スーパーバイザー クラスタの基盤となるインフラストラクチャ リソースによってバッキングされて、保証されます。

vSphere 管理者はカスタムの仮想マシン クラスを作成できます。

また、ワークロード管理には、いくつかのデフォルトの仮想マシン クラスがあります。デフォルトのクラス タイプには、一般に、保証型とベスト エフォート型の 2 つのエディションがあります。保証型のエディションは、仮想マシンの仕様によって要求されるリソースを完全に予約します。ベスト エフォート型のクラス エディションは、リソースの予約を行いません。つまり、リソースのオーバーコミットが可能です。通常、保証型のタイプは本番環境で使用されます。

デフォルトの仮想マシン クラスの例を次に示します。

クラス	CPU	メモリ (GB)	予約済みの CPU とメモリ
`guaranteed-large`	4	16	はい
`best-effort-large`	4	16	なし
`guaranteed-small`	2	4	はい
`best-effort-small`	2	4	なし

vSphere 管理者は既存の仮想マシン クラスを任意の数だけ割り当てて、特定の名前空間内で DevOps エンジニアがこれらを使用できるようにします。

仮想マシン クラスを使用すると、DevOps エンジニアの操作環境は簡素化されます。DevOps は、エンジニアが作成する各仮想マシンの構成全体を認識する必要はありません。代わりに、使用可能なオプションから仮想マシン クラスを選択することができ、仮想マシン サービスは仮想マシン構成を管理するようになります。

Kubernetes 側では、仮想マシン クラスが `VirtualMachineClass` および `VirtualMachineClassBinding` のリソースとして表示されます。

仮想マシン イメージ

仮想マシン イメージは、オペレーティング システム、アプリケーション、データなどのソフトウェア構成が含まれているテンプレートです。

DevOps エンジニアは仮想マシンを作成するときに、名前空間に関連付けられているコンテンツ ライブラリからイメージを選択できます。イメージは DevOps に `VirtualMachineImage` オブジェクトとして公開されます。

仮想マシン サービスは一部の仮想マシン イメージとゲスト OS をサポートします。互換性のある仮想マシン イメージは VMware Marketplace に OVF として表示されます。仮想マシン サービスでサポートされている仮想マシン イメージのみを使用してください。互換性のあるイメージを見つけるには、 VMware Cloud Marketplace Web サイトで**仮想マシン サービス イメージ**を検索します。CentOS の仮想マシン サービス イメージの例については、「CentOS の仮想マシン サービス イメージ」を参照してください。

コンテンツ ソース

DevOps エンジニアは、仮想マシンを作成するためのイメージ ソースとしてコンテンツ ライブラリを使用します。vSphere 管理者は、仮想マシン クラスと同様に、DevOps エンジニアがこれらを使用できるように、既存のコンテンツ ライブラリを名前空間に割り当てることができます。

ストレージ クラス

仮想マシン サービスはストレージ クラスを使用して仮想ディスクを配置し、パーシステント ボリュームを動的に接続します。ストレージ クラスの詳細については、10 章 vSphere with Tanzu でのパーシステント ストレージの使用を参照してください。

仮想マシンの仕様

DevOps エンジニアは、仮想マシン イメージ、仮想マシン クラス、およびストレージ クラスをまとめて記述する YAML ファイルで、仮想マシンの目標とする状態を記述します。

ネットワーク

仮想マシン サービスには特定の要件が設定されていないため、vSphere with Tanzu で使用可能なネットワーク構成を利用します。仮想マシン サービスは、vSphere ネットワークと NSX-T というネットワーク タイプを両方サポートしています。仮想マシンをデプロイすると、使用可能なネットワーク プロバイダによって仮想マシンに固定 IP アドレスが割り当てられます。詳細については、『4 章 vSphere with Tanzu のネットワーク』を参照してください。

仮想マシンをプロビジョニングするための vSphere 管理者のワークフロー

vSphere 管理者は仮想マシンのポリシーとガバナンスの保護を設定し、仮想マシン クラスや仮想マシン テンプレートなどの仮想マシン リソースを DevOps エンジニアに提供します。仮想マシンのデプロイ後は、vSphere Client を使用して仮想マシンを監視できます。

手順	説明	方法
1	仮想マシン クラスを作成および管理します。	■ vSphere with Tanzu での仮想マシン クラスの作成 ■ NVIDIA vGPU を使用するには、仮想マシン クラスで PCI デバイスを構成します。vSphere with Tanzu での仮想マシン クラスへの PCI デバイスの追加を参照してください。 ■ vSphere with Tanzu での仮想マシン クラスの編集または削除
2	一連の仮想マシン クラスを名前空間に関連付けます。	仮想マシン クラスと vSphere with Tanzu の名前空間の関連付け
3	コンテンツ ライブラリを作成および管理します。	■ スタンドアローン仮想マシンの場合は、vSphere with Tanzu におけるスタンドアローン仮想マシン向けのコンテンツ ライブラリの作成と管理を参照してください。 ■ Tanzu Kubernetes クラスタの場合は、Tanzu Kubernetes リリース のコンテンツ ライブラリの作成と管理を参照してください。
4	コンテンツ ライブラリを名前空間に関連付けます。	■ スタンドアローン仮想マシンの場合は、vSphere with Tanzu での仮想マシン コンテンツ ライブラリと名前空間の関連付けを参照してください。 ■ Tanzu Kubernetes クラスタの場合は、Tanzu Kubernetes リリース 用の vSphere 名前空間 の構成を参照してください。
5	ストレージ クラスを名前空間に関連付けます。	vSphere 名前空間 の作成と設定
6	デプロイされた仮想マシンを監視します。	vSphere with Tanzu で利用可能な仮想マシンの監視

仮想マシンをプロビジョニングするための DevOps エンジニアのワークフロー

権限を持つ DevOps エンジニアは、使用可能な仮想マシン リソースを確認し、名前空間に仮想マシンをデプロイできます。これらのエンジニアは `kubectl` コマンドを使用して、以下のタスクを実行できます。

手順	説明	方法
1	名前空間に関連付けられている仮想マシン クラス、イメージ、その他のリソースを一覧表示します。	vSphere with Tanzu の名前空間で使用可能な仮想マシン リソースの表示
2	仮想マシンを作成します。	■ スタンドアローン仮想マシンの場合は、vSphere with Tanzu への仮想マシンのデプロイを参照してください。 ■ Tanzu Kubernetes クラスタの仮想マシンの場合は、TKGS v1alpha2 API を使用して Tanzu Kubernetes クラスタをプロビジョニングするためのワークフローを参照してください。

この章には、次のトピックが含まれています。

- vSphere with Tanzu での仮想マシン クラスの作成
- vSphere with Tanzu での仮想マシン クラスへの PCI デバイスの追加
- vSphere with Tanzu での仮想マシン クラスの編集または削除
- 仮想マシン クラスと vSphere with Tanzu の名前空間の関連付け
- vSphere with Tanzu の名前空間での仮想マシン クラスの管理
- vSphere with Tanzu の名前空間で使用可能な仮想マシン リソースの表示
- vSphere with Tanzu への仮想マシンのデプロイ
- vSphere with Tanzu で利用可能な仮想マシンの監視

vSphere with Tanzu での仮想マシン クラスの作成

vSphere 管理者として、vSphere with Tanzu 内の名前空間に仮想マシンをデプロイする際に使用するカスタムの仮想マシン クラスを作成します。カスタム仮想マシン クラスは、名前空間で実行されるスタンドアローン仮想マシンと、Tanzu Kubernetes クラスタをホストしている仮想マシンで使用できます。

仮想マシン クラスは、仮想マシンの CPU、メモリ、予約を定義するテンプレートです。開発のニーズを予想し、リソース可用性および制約を考慮することで、仮想マシン クラスにより、仮想マシンのポリシーおよびガバナンスに対する保護を設定できます。vSphere with Tanzu には、いくつかのデフォルトの仮想マシン クラスがあります。これらのクラスは、そのまま使用したり、編集または削除したりできます。

カスタムの仮想マシン クラスを作成することもできます。新しいクラスを作成する場合は、次の点を考慮してください。

- vCenter Server インスタンスで作成した仮想マシン クラスは、すべての vCenter Server クラスタおよびこれらのクラスタ内のすべての名前空間で使用できます。
- 仮想マシン クラスは、vCenter Server 内の仮想マシン内のすべての名前空間で使用できます。ただし、DevOps エンジニアは、特定の名前空間に関連付けられた仮想マシン クラスのみを使用できます。

前提条件

必要な権限：

- 名前空間.クラスタ全体の構成の変更
- 名前空間.名前空間構成の変更
- 仮想マシン クラス.仮想マシン クラスの管理

手順

1 [VM サービス] 画面に移動します。

a vSphere Client ホーム メニューから、[ワークロード管理] を選択します。

b [サービス] タブをクリックし、[VM サービス] ペインで [管理] をクリックします。

2 [VM サービス] 画面で [VM クラス] をクリックして、[VM クラスの作成] をクリックします。

3 [構成] 画面で、仮想マシン クラスの全般的な属性を指定します。

仮想マシン クラスの属性	説明
名前	仮想マシン クラスを識別します。次の要件を満たす一意の DNS 準拠名を入力します。 ■ 環境内のデフォルトまたはカスタム仮想マシン クラスの名前と重複しない、一意の名前を使用します。 ■ 63 文字以下の英数字の文字列を使用します。 ■ 大文字や空白は使用しないでください。 ■ ダッシュは最初または最後の文字以外の場所で使用します。たとえば、**`vm-class1`** のようにします。 仮想マシン クラスを作成した後に、名前を変更することはできません。
vCPU の数	仮想マシンの仮想 CPU (vCPU) の数を定義します。これは仮想マシンのハードウェア構成を示します。DevOps ユーザーが仮想マシン クラスを仮想マシンに割り当てている場合、この数は仮想マシンに構成された vCPU の数になります。
CPU リソース予約	オプションのパラメータです。仮想マシンに保証される CPU リソースの最小割り当てを指定します。値はパーセント (%) で表します。値が 0 % の場合は、CPU 予約なしと定義されます。 入力した割合に、すべてのクラスタ ノードで使用可能な最小 CPU が乗算されます。結果の値 (MHz) が、vSphere が仮想マシンに保証する CPU リソース量になります。
メモリ	仮想マシン用に構成されたメモリを MB、GB、または TB 単位で定義します。これは仮想マシンのハードウェア構成を示します。DevOps ユーザーが仮想マシン クラス ポリシーを仮想マシンに割り当てている場合、仮想マシンはこの属性で定義された容量のメモリを受け取ります。 値は 4 MB ～ 24 TB の範囲内の、4 MB の倍数で指定する必要があります。
メモリ リソース予約	オプションのパラメータです。仮想マシンに構成されるメモリの予約容量を定義します。この属性の値の範囲は 0 ～ 100% です。 仮想マシン クラス構成に PCI デバイスを追加する場合は、このパラメータを 100% に設定します。

4 （オプション） PCI デバイスを追加するには、[構成] 画面の [PCI デバイス] ドロップダウン メニューから [はい] を選択して、[次へ] をクリックします。

このオプションを選択すると、メモリ リソース予約の値が自動的に 100% に変更されます。

要件およびその他の詳細については、vSphere with Tanzu での仮想マシン クラスへの PCI デバイスの追加を参照してください。

5 [確認] 画面で詳細を確認して、[完了] をクリックします。

次のステップ

仮想マシン クラスの作成後、パラメータの編集や、環境からの削除が可能になります。vSphere with Tanzu での仮想マシン クラスの編集または削除を参照してください。

仮想マシン クラスを DevOps エンジニアが使用できるようにするには、そのクラスを名前空間に関連付けます。仮想マシン クラスの関連付けは、名前空間単位で行われます。仮想マシン クラスと vSphere with Tanzu の名前空間の関連付けを参照してください。

vSphere with Tanzu での仮想マシン クラスの属性

vSphere 管理者は、vSphere 名前空間 内の仮想マシンで使用される仮想マシン (VM) クラスを作成したり、編集したりできます。仮想マシン クラスごとに、使用可能な属性のサブセットを指定します。

次の表に、仮想マシン クラス内で定義できるすべての属性を示します。

仮想マシン クラスの属性	説明
名前	仮想マシン クラスを識別します。次の要件を満たす一意の DNS 準拠名を入力します。 ■ 環境内のデフォルトまたはカスタム仮想マシン クラスの名前と重複しない、一意の名前を使用します。 ■ 63 文字以下の英数字の文字列を使用します。 ■ 大文字や空白は使用しないでください。 ■ ダッシュは最初または最後の文字以外の場所で使用します。たとえば、**`vm-class1`** のようにします。 仮想マシン クラスを作成した後に、名前を変更することはできません。
vCPU の数	仮想マシンの仮想 CPU (vCPU) の数を定義します。これは仮想マシンのハードウェア構成を示します。DevOps ユーザーが仮想マシン クラスを仮想マシンに割り当てている場合、この数は仮想マシンに構成された vCPU の数になります。
CPU リソース予約	オプションのパラメータです。仮想マシンに保証される CPU リソースの最小割り当てを指定します。値はパーセント (%) で表します。値が 0 % の場合は、CPU 予約なしと定義されます。 入力した割合に、すべてのクラスタ ノードで使用可能な最小 CPU が乗算されます。結果の値 (MHz) が、vSphere が仮想マシンに保証する CPU リソース量になります。

仮想マシン クラスの属性	説明
メモリ	仮想マシン用に構成されたメモリを MB、GB、または TB 単位で定義します。これは仮想マシンのハードウェア構成を示します。DevOps ユーザーが仮想マシン クラス ポリシーを仮想マシンに割り当てている場合、仮想マシンはこの属性で定義された容量のメモリを受け取ります。 値は 4 MB ～ 24 TB の範囲内の、4 MB の倍数で指定する必要があります。
メモリ リソース予約	オプションのパラメータです。仮想マシンに構成されるメモリの予約容量を定義します。この属性の値の範囲は 0 ～ 100% です。 仮想マシン クラス構成に PCI デバイスを追加する場合は、このパラメータを 100% に設定します。

vSphere with Tanzu での仮想マシン クラスへの PCI デバイスの追加

vSphere with Tanzu 環境内の ESXi ホストに NVIDIA GRID GPU グラフィック デバイスが 1 つ以上搭載されている場合は、NVIDIA GRID 仮想 GPU (vGPU) テクノロジーを使用するように仮想マシンを構成できます。パススルー モードで仮想マシンを使用できるように ESXi ホスト上の他の PCI デバイスを構成することもできます。

NVIDIA GRID GPU

NVIDIA GRID GPU グラフィック デバイスは、複雑なグラフィック操作を最適化し、CPU に過大な負荷をかけずに高パフォーマンスで動作するように設計されています。NVIDIA GRID vGPU は、単一の物理 GPU を個別の vGPU 対応パススルー デバイスとして複数の仮想マシンで共有することにより、比類のないグラフィック パフォーマンス、費用対効果、およびスケーラビリティを提供します。

仮想マシンに NVIDIA vGPU を構成するときは、vGPU 用の PCI デバイスを仮想マシン クラスに追加します。

NVIDIA vGPU を使用するときは次の考慮事項が適用されます。

- 仮想マシン サービスで管理される vGPU デバイスを備えた仮想マシンは、ESXi ホストがメンテナンス モードになると自動的にパワーオフされます。このことが、仮想マシンで実行されているワークロードに一時的に影響する可能性があります。ホストがメンテナンス モードを終了すると、仮想マシンは自動的にパワーオンされます。

動的 DirectPath I/O

動的 DirectPath I/O を使用すると、仮想マシンはホストに接続されている物理 PCI および PCIe デバイスに直接アクセスできます。

動的 DirectPath I/O を使用して、複数の PCI パススルー デバイスを仮想マシンに割り当てることができます。各パススルー デバイスは、PCI ベンダーおよびデバイス識別子で指定できます。

前提条件

- ホスト マシンがサポートされていることを VMware 互換性ガイドで確認した後、ホストが電源および構成の要件を満たしていることをベンダーに問い合わせて確認します。ESXi ホストに PCI デバイスをインストールします。

- NVIDIA vGPU を構成するには、次の前提条件に従います。
  - vSphere バージョン 7.0 Update 3 以降を使用します。
  - [直接共有] モードの 1 つ以上のデバイスを使用するように ESXi ホストのグラフィック設定を構成します。ホスト グラフィックの設定を参照してください。
  - NVIDIA vGPU ソフトウェアをインストールします。NVIDIA は、次のコンポーネントを含む vGPU ソフトウェア パッケージを提供します。

    詳細については、該当する NVIDIA 仮想 GPU ソフトウェア ドキュメントを参照してください。
    - vSphere 管理者が ESXi ホストにインストールする vGPU マネージャ。VMware のナレッジベースの記事 KB2033434 を参照してください。
    - 仮想マシンのデプロイおよび起動後に DevOps エンジニアが仮想マシンにインストールするゲスト仮想マシン ドライバ。vSphere with Tanzu の仮想マシンへの NVIDIA ゲスト ドライバのインストールを参照してください。
- PCI パススルー デバイスに動的 DirectPath I/O を構成するには、次の前提条件に従います。
  - vSphere バージョン 7.0 Update 3 MP01 を使用します。
  - PCI デバイスをホストに接続して、パススルーで使用可能とマークします。PCI デバイスにパススルーとしてマークを参照してください。
- 必要な権限：
  - 名前空間.クラスタ全体の構成の変更
  - 名前空間.名前空間構成の変更
  - 仮想マシン クラス.仮想マシン クラスの管理

手順

1 既存の仮想マシン クラスを作成または編集するときは、仮想マシン クラスに PCI デバイスを追加します。

オプション	操作
新しい VM クラスの作成	a vSphere Client ホーム メニューから、[ワークロード管理] を選択します。 b [サービス] タブをクリックし、[VM サービス] ペインで [管理] をクリックします。 c [VM サービス] 画面で [VM クラス] をクリックして、[VM クラスの作成] をクリックします。 d [構成] 画面で、仮想マシン クラスの全般的な属性を指定します。vSphere with Tanzu での仮想マシン クラスの属性を参照してください。 メモリ リソース予約の値が 100% に設定されていることを確認します。 e PCI デバイスを追加するには、[構成] 画面の [PCI デバイス] ドロップダウン メニューから [はい] を選択して、[次へ] をクリックします。
仮想マシン クラスの編集	a vSphere Client ホーム メニューから、[ワークロード管理] を選択します。 b [サービス] タブをクリックし、[VM サービス] ペインで [管理] をクリックします。 c [VM サービス] 画面で [VM クラス] をクリックします。 d 既存の仮想マシン クラスのペインで、[管理] をクリックして [編集] をクリックします。 メモリ リソース予約の値が 100% に設定されていることを確認します。 e PCI デバイスを追加するには、[構成] 画面の [PCI デバイス] ドロップダウン メニューから [はい] を選択して、[次へ] をクリックします。

2 [PCI デバイス] 画面で [PCI デバイスの追加] メニューを展開し、アクセス タイプと他の適切なオプションを選択して、[次へ] をクリックします。

オプション	操作
NVIDIA GRID vGPU	次のオプションを指定します。 ■ [モデル]。物理デバイスの名前。ホストで使用可能なデバイスのリストからデバイスを選択します。 ■ [GPU 共有]。仮想マシン間での物理 GPU の共有方法を示します。たとえば、[時刻の共有] などがあります。 ■ [GPU モード]。仮想マシンの GPU モード。たとえば、[コンピューティング] は、高パフォーマンス コンピューティング アプリケーション用に最適化された構成です。[Workstation] は、グラフィックを多用するワークロードに使用されます。 ■ [GPU メモリ]。仮想マシンあたりの最小 GPU メモリ (GB)。 ■ [vGPU の数]。仮想マシンあたりの vGPU デバイス数。
動的 DirectPath I/O	[PCI デバイス] リストで、ベンダー、モデル名、またはハードウェア ラベルを使用して PCI パススルー デバイスを選択します。

3 [確認] 画面で詳細を確認して、[完了] をクリックします。

結果

仮想マシン クラスのペインに [GPU] タグが表示されている場合は、仮想マシン クラスが GPU 対応となっています。

vSphere with Tanzu での仮想マシン クラスの編集または削除

vSphere 管理者は、vSphere 名前空間 で仮想マシンのカスタム仮想マシン クラスを作成できます。仮想マシン クラスを作成した後に、パラメータを編集できます。vSphere with Tanzu が提供するデフォルトの仮想マシン クラスを編集することもできます。既存の仮想マシン クラスが不要になった場合は、環境から削除できます。

仮想マシン クラスを編集しても、このクラスから以前にデプロイされた仮想マシンは自動的に再構成されません。たとえば、DevOps ユーザーが仮想マシン クラスを持つ Tanzu Kubernetes クラスタを作成し、その後で仮想マシン クラス定義を変更した場合、既存の Tanzu Kubernetes 仮想マシンは影響を受けません。新しい Tanzu Kubernetes 仮想マシンは、変更されたクラス定義を使用します。

注意： Tanzu Kubernetes クラスタで使用される仮想マシン クラスを編集した後に、このクラスタをスケール アウトした場合、新しいクラスタ ノードは更新されたクラス定義を使用しますが、既存のクラスタ ノードは初期のクラス定義を引き続き使用するため、不一致が生じます。制御プレーン ノードとワーカー ノードを両方とも拡張できます。スケールの詳細については、Tanzu Kubernetes Grid サービス v1alpha1 API を使用した Tanzu Kubernetes クラスタのスケーリングを参照してください。

仮想マシン クラスを削除すると、関連付けられているすべての名前空間から削除されます。DevOps ユーザーは、この仮想マシン クラスを使用して仮想マシンをセルフサービスできなくなります。この仮想マシン クラスですでに作成されている仮想マシンは、影響を受けません。

前提条件

- 仮想マシン クラスが 1 つ以上あることを確認します。vSphere with Tanzu での仮想マシン クラスの作成を参照してください。
- 必要な権限：
  - 名前空間.クラスタ全体の構成の変更
  - 名前空間.名前空間構成の変更
  - 仮想マシン クラス.仮想マシン クラスの管理

手順

1. vSphere Client に、使用可能な仮想マシン クラスが表示されます。
   a. vSphere Client ホーム メニューから、[ワークロード管理] を選択します。
   b. [サービス] タブをクリックし、[VM サービス] ペインをクリックします。
   c. [VM サービス] 画面で [VM クラス] をクリックします。

      すべてのデフォルトの仮想マシン クラスまたはユーザー作成の仮想マシン クラスは、[使用可能な VM クラス] の下に表示されます。

2 既存の仮想マシン クラスを編集または削除します。

オプション	説明
仮想マシン クラスの編集	a 選択した仮想マシン クラスのペインで、[管理] をクリックし、[編集] をクリックします。 b 仮想マシン クラスのパラメータを変更します。vSphere with Tanzu での仮想マシン クラスの属性を参照してください。 **注：** 仮想マシン クラスの名前を変更することはできません。
仮想マシン クラスの削除	a 選択した仮想マシン クラスのペインで、[管理] をクリックし、[削除] をクリックします。 b 仮想マシン クラスを削除することを確定します。

次のステップ

仮想マシン クラスを DevOps エンジニアが使用できるようにするには、その仮想マシン クラスを名前空間に関連付けます。仮想マシン クラスの関連付けは、名前空間単位で行われます。仮想マシン クラスと vSphere with Tanzu の名前空間の関連付けを参照してください。

仮想マシン クラスと vSphere with Tanzu の名前空間の関連付け

vSphere 管理者は、仮想マシン クラスを スーパーバイザー クラスタ の 1 つ以上の名前空間に追加できます。名前空間に仮想マシン クラスを追加する場合は、Kubernetes 名前空間環境で仮想マシンのセルフサービスを開始できるように DevOps ユーザーがこのクラスを使用できるようにします。名前空間に割り当てる仮想マシン クラスは、Tanzu Kubernetes クラスタを構成する仮想マシンでも使用されます。

1 つの名前空間に複数の仮想マシン クラスを追加できます。さまざまな仮想マシン クラスが、さまざまなレベルのサービスの指標として機能します。複数の仮想マシン クラスを公開している場合、DevOps ユーザーは名前空間で仮想マシンを作成および管理する際に、すべてのカスタム クラスとデフォルト クラスの中から選択できます。

注： 新しく作成した名前空間に Tanzu Kubernetes クラスタをデプロイできるようにするには、DevOps エンジニアに仮想マシン クラスへのアクセス権が必要になります。vSphere 管理者は、デフォルトまたはカスタムの仮想マシン クラスを、Tanzu Kubernetes クラスタがデプロイされている新しい名前空間に明示的に関連付ける必要があります。既存の（Tanzu Kubernetes クラスタがすでにプロビジョニングされている）名前空間は、引き続きデフォルトの仮想マシン クラスに自動的にアクセスできます。ただし、カスタム仮想マシン クラスを既存の名前空間に関連付けることもできます。

前提条件

VMware から提供されるデフォルトの仮想マシン クラスを使用するか、新しいクラスを作成します。vSphere with Tanzu での仮想マシン クラスの作成を参照してください。

必要な権限：

- 名前空間.クラスタ全体の構成の変更
- 名前空間.名前空間構成の変更
- 仮想マシン クラス.仮想マシン クラスの管理

手順

1 vSphere Client で、名前空間に移動します。

a vSphere Client ホーム メニューから、[ワークロード管理] を選択します。

b [名前空間] タブをクリックして、名前空間をクリックします。

2 仮想マシン クラスを追加します。

a [VM サービス] ペインで [VM クラスの追加] をクリックします。

b 1 つ以上の仮想マシン クラスを選択して、[OK] をクリックします。

結果

追加した仮想マシン クラスは、DevOps が仮想マシンをセルフサービスする際に、名前空間で使用できるようになります。これらのクラスは、Tanzu Kubernetes クラスタを構成する仮想マシンで使用することもできます。

次のステップ

仮想マシン クラスを名前空間に関連付けた後に、仮想マシン クラスを追加するか、クラスを削除して名前空間から公開を解除することができます。vSphere with Tanzu の名前空間での仮想マシン クラスの管理を参照してください。

vSphere with Tanzu の名前空間での仮想マシン クラスの管理

vSphere 管理者は、スーパーバイザー クラスタ で仮想マシン クラスを 1 つ以上の名前空間に関連付けることができます。仮想マシン クラスを名前空間に関連付けた後に、仮想マシン クラスを追加するか、クラスを削除して Kubernetes 名前空間から公開解除することができます。

前提条件

- 1 つ以上の仮想マシン クラスが名前空間に関連付けられていることを確認します。仮想マシン クラスと vSphere with Tanzu の名前空間の関連付けを参照してください。
- 名前空間から仮想マシン クラスを削除する場合は、仮想マシン クラスが Tanzu Kubernetes Grid サービスで使用されていないことを確認します。仮想マシンを削除すると、Tanzu Kubernetes Grid サービス の操作が影響を受けることがあります。
- 必要な権限：
  - 名前空間.クラスタ全体の構成の変更
  - 名前空間.名前空間構成の変更
  - 仮想マシン クラス.仮想マシン クラスの管理

手順

1 vSphere Client で、名前空間に移動します。

a vSphere Client ホーム メニューから、[ワークロード管理] を選択します。

b [名前空間] タブをクリックして、名前空間をクリックします。

2 仮想マシン クラスを追加または削除します。

a [VM サービス] ペインで [VM クラスの管理] をクリックします。

b 次のいずれかの操作を実行します。

オプション	説明
仮想マシン クラスの削除	仮想マシン クラスを選択解除し、[OK] をクリックします。
仮想マシン クラスの追加	1 つ以上の仮想マシン クラスを選択して、[OK] をクリックします。

vSphere with Tanzu の名前空間で使用可能な仮想マシン リソースの表示

スタンドアローン仮想マシンを vSphere with Tanzu にデプロイできるようにするには、DevOps エンジニアに特定の仮想マシン リソースに対するアクセス権が必要です。DevOps エンジニアは、これらのリソースにアクセスできること、および環境内で使用可能な仮想マシン クラスと仮想マシン テンプレートを表示できることを確認します。また、仮想マシンのセルフサービスに必要になることがあるストレージ クラスやその他のアイテムを一覧表示することもできます。

このタスクでは、スタンドアローン仮想マシンのデプロイで使用可能なリソースにアクセスするためのコマンドについて説明します。Tanzu Kubernetes クラスタおよびこのクラスタを構成する仮想マシンをデプロイするために必要なリソースについては、Tanzu Kubernetes クラスタの仮想マシンのクラスを参照してください。

前提条件

vSphere 管理者が次の手順を実行しています。

- 名前空間を作成し、ストレージ ポリシーに関連付けました。vSphere 名前空間 の作成と設定を参照してください。
- デフォルトまたはカスタムの仮想マシン クラスを名前空間に関連付けました。仮想マシン クラスと vSphere with Tanzu の名前空間の関連付けを参照してください。
- コンテンツ ライブラリを作成し、名前空間に関連付けました。vSphere with Tanzu での仮想マシン コンテンツ ライブラリと名前空間の関連付けを参照してください。

注： コンテンツ ライブラリがセキュリティ ポリシーによって保護されている場合は、すべてのライブラリ アイテムが準拠している必要があります。保護されたライブラリ内に準拠しているアイテムと準拠していないアイテムが混在している場合、`kubectl get virtualmachineimages` コマンドで DevOps エンジニアに仮想マシン イメージを表示できません。

手順

1 Kubernetes 環境の名前空間にアクセスします。

スーパーバイザー クラスタ コンテキストの取得と使用を参照してください。

2 名前空間で使用可能な仮想マシン クラスを表示するには、次のコマンドを実行します。

```
kubectl get virtualmachineclassbindings
```

次の出力が表示されます。

注： ベスト エフォート型の仮想マシン クラス タイプではリソースをオーバーコミットできるため、仮想マシンをプロビジョニングする名前空間に制限を設定していると、リソースを使い果たすおそれがあります。そのため、本番環境では保証型の仮想マシン クラス タイプを使用します。

```
NAME                        VIRTUALMACHINECLASS        AGE

best-effort-large           best-effort-large          44m
best-effort-medium          best-effort-medium         44m
best-effort-small           best-effort-small          44m
best-effort-xsmall          best-effort-xsmall         44m
custom                      custom                     44m
```

3 特定の仮想マシン クラスの詳細を表示するには、次のコマンドを実行します。

- `kubectl describe virtualmachineclasses` *name_vm_class*

  仮想マシン クラスに vGPU デバイスが含まれている場合、そのプロファイルは `spec: hardware: devices: vgpuDevices` に表示されます。

```
......
spec:
  hardware:
    cpus: 4
    devices:
      vgpuDevices:
      - profileName: grid_v100-q4
......
```

- `kubectl get virtualmachineclasses -o wide`

  仮想マシン クラスに vGPU デバイスまたはパススルー デバイスが含まれている場合は、出力の `VGPUDevicesProfileNames` または `PassthroughDeviceIDs` 列に表示されます。

4 仮想マシン イメージを表示します。

```
kubectl get virtualmachineimages
```

表示される出力は次のようになります。

```
NAME                                              VERSION  OSTYPE                FORMAT
IMAGESUPPORTED  AGE
centos-stream-8-vmservice-v1alpha1-xxxxxxxxxxxxx           centos8_64Guest       ovf
true            4d3h
```

5 特定のイメージを記述するには、次のコマンドを使用します。

```
kubectl describe virtualmachineimage/centos-stream-8-vmservice-v1alpha1-
xxxxxxxxxxxxx
```

vGPU デバイスを備えた仮想マシンには、起動モードが EFI に設定された CentOS などのイメージが必要です。これらのイメージに確実にアクセスできるようにします。サポートされているイメージについては、VMware Cloud Marketplace Web サイトで**仮想マシン サービス イメージ**を検索します。

6 ストレージ クラスにアクセスできることを確認します。

```
kubectl get resourcequotas
```

詳細については、『vSphere 名前空間 または Tanzu Kubernetes クラスタでのストレージ クラスの表示』を参照してください。

```
NAME                          AGE
REQUEST
     LIMIT
my-ns-ubuntu-storagequota   24h   wcpglobal-storage-profile.storageclass.storage.k8s.io/
requests.storage: 0/9223372036854775807
```

7 ワークロード ネットワークに vSphere Distributed Switch を使用している場合は、ネットワークの名前を取得します。

注： この情報は、`networkType` が **`vsphere-distributed`** の場合に、仮想マシンの YAML ファイルで `networkName` パラメータを指定する場合に使用します。VMware NSX-T を使用する場合は、ネットワーク名を取得して指定する必要はありません。

```
kubectl get network
```

```
NAME      AGE
primary   7d2h
```

次のステップ

仮想マシンをデプロイできるようになりました。vSphere with Tanzu への仮想マシンのデプロイを参照してください。

vSphere with Tanzu への仮想マシンのデプロイ

DevOps エンジニアは、Kubernetes YAML ファイルに仮想マシンのデプロイ仕様を記述することで、宣言による方法で仮想マシンとそのゲスト OS をプロビジョニングできます。

前提条件

- 名前空間への仮想マシンのデプロイに使用できるリソースがあることを確認します。vSphere with Tanzu の名前空間で使用可能な仮想マシン リソースの表示を参照してください。
- 仮想マシンに NVIDIA vGPU または他の PCI デバイスを使用する場合は、次の考慮事項が適用されます。
  - PCI 構成の適切な仮想マシン クラスを使用してください。vSphere with Tanzu での仮想マシン クラスへの PCI デバイスの追加を参照してください。

- vGPU デバイスを備えた仮想マシンには、起動モードが EFI に設定された CentOS などのイメージが必要です。これらのイメージに確実にアクセスできるようにします。サポートされているイメージについては、VMware Cloud Marketplace Web サイトで**仮想マシン サービス イメージ**を検索します。
- 仮想マシン サービスで管理される vGPU デバイスを備えた仮想マシンは、ESXi ホストがメンテナンス モードになると自動的にパワーオフされます。このことが、仮想マシンで実行されているワークロードに一時的に影響する可能性があります。ホストがメンテナンス モードを終了すると、仮想マシンは自動的にパワーオンされます。

手順

1 仮想マシン YAML ファイルを準備します。

ファイルで、次のパラメータを指定します。

オプション	説明
apiVersion	仮想マシン サービス API のバージョンを指定します。例：**vmoperator.vmware.com/v1alpha1**。
kind	作成する Kubernetes リソースのタイプを指定します。使用可能な値は、**VirtualMachine** です。
spec.imageName	仮想マシンが使用するコンテンツ ライブラリ イメージを指定します。たとえば、**centos-stream-8-vmservice-v1alpha1-xxxxxxxxxxxxx** と入力します。
spec.storageClass	パーシステント ボリュームのストレージに使用するストレージ クラスを識別します。例：**wcpglobal-storage-profile**。
spec.className	使用する仮想ハードウェア設定を記述する仮想マシン クラスの名前を指定します。例：**custom**。
spec.networkInterfaces	仮想マシンのネットワーク関連の設定を指定します。 ■ networkType。このキーの値には **nsx-t** または **vsphere-distributed** を指定できます。 ■ networkName。networkType が **vsphere-distributed** の場合のみ、名前を指定します。この情報は、kubectl get network コマンドを使用して取得できます。 networkType が **nsx-t** の場合、networkName を指定する必要はありません。
spec.vmMetadata	仮想マシンに渡す追加のメタデータが含まれます。このキーを使用すると、ゲスト OS イメージをカスタマイズし、仮想マシンの hostname などのアイテムや、パスワード、ssh キーなどを含む user-data を設定できます。次の YAML の例では、メタデータを格納するために ConfigMap を使用します。

YAML ファイル vmsvc-centos-vm.yaml の例として、以下を使用します。

```
apiVersion: vmoperator.vmware.com/v1alpha1
kind: VirtualMachine
metadata:
  name: vmsvc-centos-vm
  namespace: my-ns-centos
spec:
  imageName: centos-stream-8-vmservice-v1alpha1-xxxxxxxxxxxxx
  className: custom
  powerState: poweredOn
```

```
  storageClass: wcpglobal-storage-profile
  networkInterfaces:
  - networkName: primary
    networkType: vsphere-distributed
  vmMetadata:
      configMapName: vmsvc-centos-nginx-cm
      transport: OvfEnv
---
apiVersion: v1
kind: ConfigMap
metadata:
    name: vmsvc-centos-nginx-cm
    namespace: my-ns-centos
data:
  user-data: >-

I2Nsb3VkLWNvbmZpZwoKcGFzc3dvcmQ6IFZNV0FSRQpzc2hfcHdhdXRoOiB0cnVlCgp1c2VyczoKICAtIG5hbWU6IHZ
td2FyZQogICAgc3VkbzogQUxMPShBTEwpIE5PUEFTU1dEOkFMTAogICAgbG9ja19wYXNzd2Q6IGZhbHNlCiAgICAjIF
Bhc3N3b3JkIHNldCB0byBBZG1pbiEyMwogICAgcGFzc3dkOiAnJDEkc2FsdCRTT0MzM2ZWYkEvWnhlSXdENXl3MXUxJ
wogICAgc2hlbGw6IC9iaW4vYmFzaAoKd3JpdGVfZmlsZXM6CiAgLSBjb250ZW50OiB8CiAgICAgVk1TVkMgU2F5cyBIZW
SGVsbG8gV29ybGQKICAgIHBhdGg6IC9oZWxsb3dvcmxkCg==
```

`ConfigMap` には、ゲスト OS のユーザー名とパスワードを指定する cloud-config Blob が含まれています。この例で、vmsvc-centos-nginx-cm ConfigMap 内の `user-data` は、base64 形式の次のスニペットを表します。

```
#cloud-config
password: VMWARE
ssh_pwauth: true
users:
  - name: vmware
    sudo: ALL=(ALL) NOPASSWD:ALL
    lock_passwd: false
    passwd: '$1$salt$SOC33fVbA/ZxeIwD5yw1u1'
    shell: /bin/bash
write_files:
  - content: |
      VMSVC Says Hello World
    path: /helloworld
```

cloud-config の仕様の詳細については、https://cloudinit.readthedocs.io/en/latest/topics/examples.html を参照してください。

2 仮想マシンをデプロイします。

```
kubectl apply -f vmsvc-centos-vm.yaml
```

3 仮想マシンが作成されたことを確認します。

```
kubectl get vm -n my-ns-centos
NAME              AGE
vmsvc-centos-vm   28s
```

4 仮想マシンのステータスと関連イベントを確認します。

```
kubectl describe virtualmachine vmsvc-centos-vm
```

出力は次のようになります。出力から、仮想マシンの IP アドレスを取得することもできます。仮想マシンの IP アドレスは `Vm Ip` フィールドに表示されます。

```
Name:         vmsvc-centos-vm
Namespace:    my-ns-centos
Annotations:  vmoperator.vmware.com/image-supported-check: disabled
API Version:  vmoperator.vmware.com/v1alpha1
Kind:         VirtualMachine
Metadata:
  Creation Timestamp:  2021-03-23T19:07:36Z
  Finalizers:
    virtualmachine.vmoperator.vmware.com
  Generation:  1
  Managed Fields:
 ...
 ...
Spec:
  Class Name:  custom
  Image Name:  vmservice-centos-20-10-server-cloudimg-amd64
  Network Interfaces:
    Network Name:  primary
    Network Type:  vsphere-distributed
  Power State:     poweredOn
  Storage Class:   wcpglobal-storage-profile
  Vm Metadata:
    Config Map Name:  vmsvc-centos-nginx-cm
    Transport:        OvfEnv
Status:
  Bios UUID:              4218ec42-aeb3-9491-fe22-19b6f954ce38
  Change Block Tracking:  false
  Conditions:
    Last Transition Time:  2021-03-23T19:08:59Z
    Status:                True
    Type:                  VirtualMachinePrereqReady
  Host:                    10.185.240.10
  Instance UUID:           50180b3a-86ee-870a-c3da-90ddbaffc950
  Phase:                   Created
  Power State:             poweredOn
  Unique ID:               vm-73
  Vm Ip:                   10.161.75.162
Events:                    <none>
...
```

5 仮想マシンの IP アドレスにアクセスできることを確認します。

```
ping 10.161.75.162
PING 10.161.75.162 (10.161.75.162): 56 data bytes
64 bytes from 10.161.75.162: icmp_seq=0 ttl=59 time=43.528 ms
64 bytes from 10.161.75.162: icmp_seq=1 ttl=59 time=53.885 ms
64 bytes from 10.161.75.162: icmp_seq=2 ttl=59 time=31.581 ms
```

結果

仮想マシン サービスを介して作成された仮想マシンを管理できるのは、Kubernetes 名前空間の DevOps のみです。vSphere Client からライフサイクルを管理することはできませんが、vSphere 管理者は仮想マシンとそのリソースを監視できます。詳細については、『vSphere with Tanzu で利用可能な仮想マシンの監視』を参照してください。

次のステップ

詳細については、ブログ記事 Introducing Virtual Machine Provisioning を参照してください。

vGPU 用に構成された PCI デバイスが仮想マシンに含まれる場合は、NVIDIA ディスプレイ ドライバをインストールします。vSphere with Tanzu の仮想マシンへの NVIDIA ゲスト ドライバのインストールを参照してください。

vSphere with Tanzu の仮想マシンへの NVIDIA ゲスト ドライバのインストール

vSphere with Tanzu 環境内に仮想マシンを作成して起動したら、この仮想マシンに NVIDIA vGPU グラフィックス ドライバをインストールして、GPU 操作を完全に有効にします。

前提条件

- NVIDIA vGPU デバイスを使用する仮想マシンを作成します。仮想マシンは、vGPU 定義を含む仮想マシン クラスを参照する必要があります。vSphere with Tanzu での仮想マシン クラスへの PCI デバイスの追加を参照してください。
- NVIDIA ダウンロード サイトから vGPU ソフトウェア パッケージをダウンロードし、パッケージを解凍し、ゲスト ドライブ コンポーネントの準備ができていることを確認します。詳細については、該当する NVIDIA 仮想 GPU ソフトウェアのドキュメントを参照してください。

  注： ドライバ コンポーネントのバージョンは、vSphere 管理者が ESXi ホストにインストールした vGPU Manager のバージョンに対応している必要があります。vSphere with Tanzu での仮想マシン クラスへの PCI デバイスの追加を参照してください。

手順

1 NVIDIA vGPU ソフトウェアの Linux ドライバ パッケージ（`NVIDIA-Linux-x86_64-version-grid.run` など）をゲスト仮想マシンにコピーします。

2 ドライバ インストーラを実行する前に、すべてのアプリケーションを終了します。

3 NVIDIA vGPU ドライバ インストーラを起動します。

```
sudo ./NVIDIA-Linux-x86_64-version-grid.run
```

4 NVIDIA ソフトウェア使用許諾契約書に同意し、[はい] を選択して設定を自動的に更新します。

5 ドライバがインストールされていることを確認します。

次に例を示します。

```
~$ nvidia-smi
Wed May 19 22:15:04 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 460.63       Driver Version: 460.63       CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  GRID V100-4Q        On   | 00000000:02:00.0 Off |                  N/A|
| N/AN/AP0    N/A/  N/A|    304MiB /  4096MiB |      0%      Default |
|                               |                      |                  N/A|
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

vSphere with Tanzu で利用可能な仮想マシンの監視

vSphere 管理者として、Kubernetes 環境で DevOps によってデプロイされた仮想マシンを監視するには、vSphere Client を使用します。

vSphere Client から仮想マシンのライフサイクルを管理することはできません。

前提条件

DevOps エンジニアが仮想マシンをデプロイしています。vSphere with Tanzu への仮想マシンのデプロイを参照してください。

手順

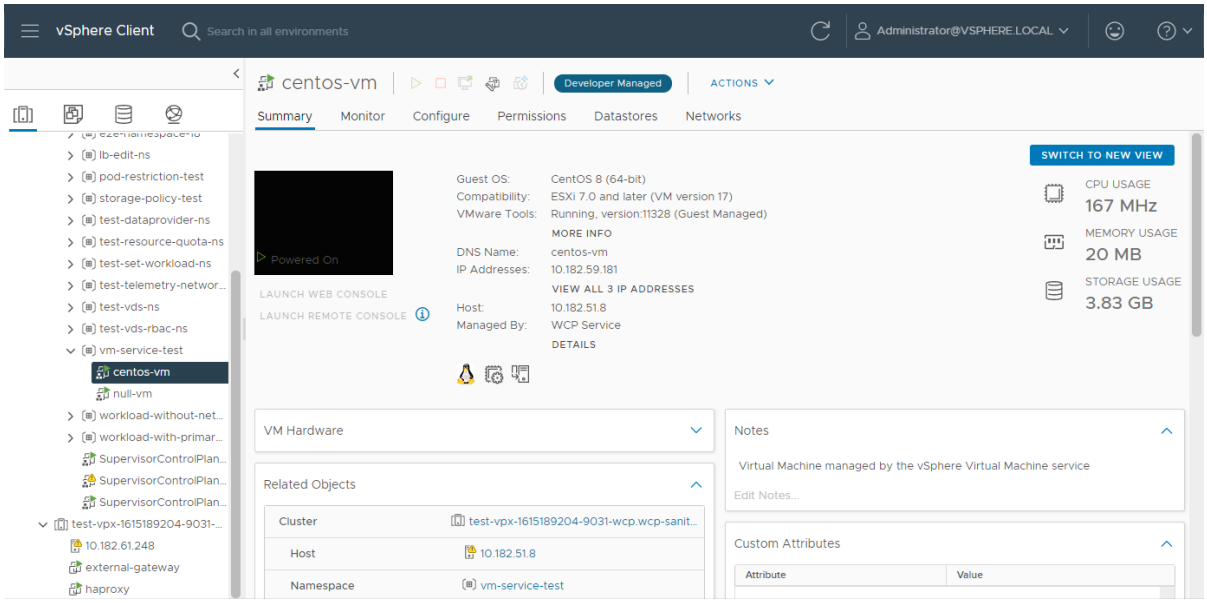
1 vSphere Client で、vSphere with Tanzu が有効になっているホスト クラスタに移動します。

2 [名前空間] で、仮想マシンがデプロイされた名前空間を展開します。

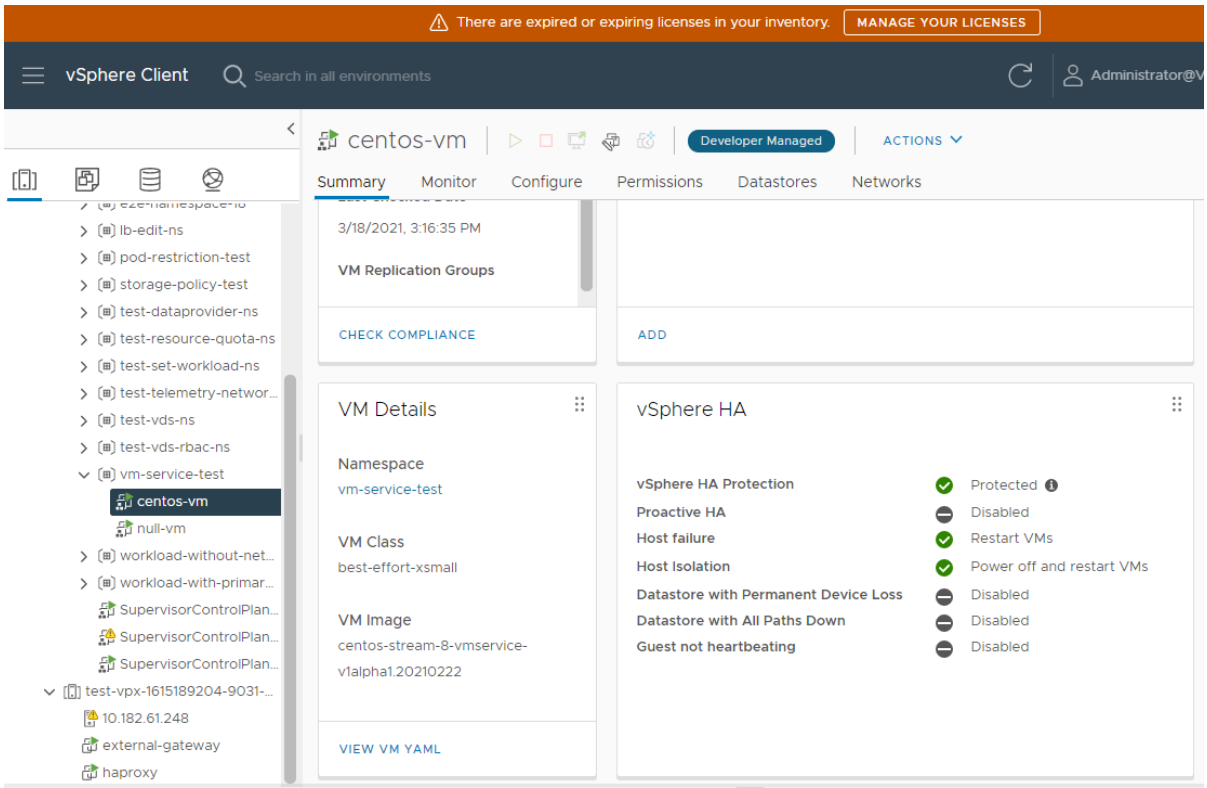
3 表示する仮想マシンを選択し、[サマリ] タブをクリックします。

[サマリ] 画面の上部にある [管理対象の開発者] タグを確認してください。

この画面には、ゲスト OS や IP アドレスなど、仮想マシンに関する情報が表示されます。



4 画面の右上隅にある [新しい表示に切り替える] をクリックして、仮想マシン クラスや仮想マシン イメージ、および仮想マシンが実行されている名前空間などの追加の詳細を表示します。



TKGS クラスタのプロビジョニングと操作

13

vSphere with Tanzu には、Tanzu Kubernetes クラスタのプロビジョニングと運用に便利なツールと簡単なワークフローが用意されています。ニーズに合わせてさまざまな構成を使用してクラスタを作成し、カスタマイズする手順および例を参照してください。

この章には、次のトピックが含まれています。

TKGS v1alpha2 API を使用して Tanzu Kubernetes クラスタをプロビジョニングするためのワークフロー

Tanzu Kubernetes クラスタをプロビジョニングするには、kubectl と、YAML で定義したクラスタ仕様を使用して、Tanzu Kubernetes Grid サービス 宣言型 API を呼び出します。クラスタをプロビジョニングしたら、クラスタを操作し、kubectl を使用してワークロードをクラスタにデプロイします。

このワークフローは、Tanzu Kubernetes Grid サービス Tanzu Kubernetes クラスタをプロビジョニングするための TKGS v1alpha2 API をサポートしています。Tanzu Kubernetes Grid サービス v1alpha1 API を使用する Tanzu Kubernetes クラスタの構成パラメータを使用している場合は、Tanzu Kubernetes Grid サービス v1alpha1 API を使用して Tanzu Kubernetes クラスタをプロビジョニングするためのワークフローを参照してください。

前提条件

ワークフローの手順を開始する前に、次の前提条件が完了していることを確認します。

- Tanzu Kubernetes Grid サービス Tanzu Kubernetes クラスタをプロビジョニングするための TKGS v1alpha2 API をサポートするように環境をインストールまたは更新します。詳細については、TKGS v1alpha2 API を使用するための要件を参照してください。Tanzu Kubernetes クラスタをプロビジョニングするための TKGS v1alpha2 API をサポートする最小の Tanzu Kubernetes リリースは `v1.21.2` です。詳細については、VMware Tanzu Kubernetes リリース ノートを参照してください。
- Tanzu Kubernetes クラスタをホストするための vSphere 名前空間 を構成します。名前空間には、DevOps エンジニアと共有ストレージのための編集権限が必要です。vSphere 名前空間 の作成と設定を参照してください。
- Tanzu Kubernetes リリース 用のコンテンツ ライブラリを作成して、使用するリリースを同期します。Tanzu Kubernetes リリース のコンテンツ ライブラリの作成と管理を参照してください。
- 使用するデフォルトの仮想マシン クラスと、カスタム仮想マシン クラスが必要かどうかを決定します。Tanzu Kubernetes クラスタの仮想マシンのクラスを参照してください。
- コンテンツ ライブラリおよび仮想マシン クラスに vSphere 名前空間 を関連付けている。Tanzu Kubernetes リリース 用の vSphere 名前空間 の構成を参照してください。

手順

1 vSphere 向け Kubernetes CLI Tools をダウンロードしてインストールします。

ガイダンスについては、vSphere 向け Kubernetes CLI Tools のダウンロードとインストールを参照してください。

2 スーパーバイザー クラスタ を使用して、kubectl 向けの vSphere プラグイン で認証します。

```
kubectl vsphere login --server=IP-ADDRESS --vsphere-username USERNAME
```

ガイダンスについては、vCenter Single Sign-On ユーザーとして スーパーバイザー クラスタ に接続するを参照してください。

3 スーパーバイザー クラスタ にログインできたことを確認します。

次のようなメッセージが表示されます。

```
Logged in successfully.

You have access to the following contexts:
   192.197.2.65
   tkgs-ns
```

ここで、`192.197.2.65` は スーパーバイザー クラスタ コンテキストで、`tkgs-ns` は vSphere 名前空間 クラスタをプロビジョニングする Tanzu Kubernetes のコンテキストです。

4 ターゲット vSphere 名前空間 が現在のコンテキストであることを確認します。

```
kubectl config get-contexts
```

```
CURRENT   NAME            CLUSTER         AUTHINFO                                NAMESPACE
          192.197.2.65    192.197.2.65    wcp:192.197.2.65:user@vsphere.local
*         tkgs-ns         192.197.2.65    wcp:192.197.2.65:user@vsphere.local   tkgs-ns
```

ターゲット vSphere 名前空間 が現在のコンテキストでない場合は、そのコンテキストに切り替えます。

```
kubectl config use-context tkgs-ns
```

5 ターゲット vSphere 名前空間 で使用可能な仮想マシン クラスのバインドを一覧表示します。

```
kubectl get virtualmachineclassbindings
```

使用できるのは、ターゲット名前空間にバインドされている仮想マシン クラスのみです。仮想マシン クラスが表示されない場合は、vSphere 名前空間 にデフォルトの仮想マシン クラスが追加されていることを確認します。

6 使用可能なパーシステント ボリューム ストレージ クラスを取得します。

```
kubectl describe storageclasses
```

7 使用可能な Tanzu Kubernetes リリース を一覧表示します。

この操作を実行には、次のいずれかのコマンドを使用します。

```
kubectl get tkr
```

```
kubectl get tanzukubernetesreleases
```

このコマンドで返されたリリースのみを使用できます。いずれのリリースも表示されない場合、または必要なリリースが表示されない場合は、目的の OVA ファイルがコンテンツ ライブラリと同期されていることを確認します。

8 Tanzu Kubernetes クラスタをプロビジョニングするための YAML ファイルを作成します。

a Tanzu Kubernetes クラスタをプロビジョニングするための TKGS v1alpha2 API の仕様を確認します。

b 要件に応じて、デフォルトの Tanzu Kubernetes クラスタをプロビジョニングするためのサンプル YAML またはカスタムの Tanzu Kubernetes クラスタをプロビジョニングするためのサンプル YAML のいずれかの TKGS v1alpha2 API を使用して Tanzu Kubernetes クラスタをプロビジョニングする YAML の例を最初に使用してクラスタをプロビジョニングします。

c YAML ファイルを `tkgs-cluster-1.yaml` などの名前で保存します。

d 前述のコマンドの出力から収集された情報を使用し、要件に基づいて、次に示す情報を YAML ファイルにポピュレートします。

- `tkgs-cluster-1` などのクラスタの名前
- `tkgs-ns` などのターゲット vSphere 名前空間
- `guaranteed-medium` や `guaranteed-small` などのバインドされた仮想マシン クラス
- `vwt-storage-policy` などのクラスタ ノードおよびワークロードのストレージ クラス
- 制御プレーンおよびワーカー ノード（レプリカ）の数
- `v1.21.6---vmware.1-tkg.1.b3d708a` などの TKR NAME 文字列で指定された Tanzu Kubernetes リリース

e 必要に応じて YAML ファイルをカスタマイズします。例：

- `etcd` や `containerd` などの変更の多いコンポーネント用のボリュームを個別に追加します
- クラスタ ノードのデフォルトのパーシステント ストレージ クラスを指定します
- CNI、ポッド、サービス CIDR などのクラスタ ネットワークをカスタマイズします

この手順を行うと、TKGS クラスタをプロビジョニングするための有効な YAML が得られます。例：

```
apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TanzuKubernetesCluster
metadata:
  name: tkgs-cluster-1
  namespace: tkgs-ns
spec:
  topology:
    controlPlane:
      replicas: 3
      vmClass: guaranteed-medium
      storageClass: vwt-storage-policy
      volumes:
        - name: etcd
          mountPath: /var/lib/etcd
          capacity:
            storage: 4Gi
      tkr:
        reference:
          name: v1.21.6---vmware.1-tkg.1.b3d708a
    nodePools:
    - name: worker-nodepool-a1
      replicas: 3
      vmClass: guaranteed-medium
      storageClass: vwt-storage-policy
      volumes:
        - name: containerd
          mountPath: /var/lib/containerd
          capacity:
            storage: 16Gi
      tkr:
        reference:
```

```
          name: v1.21.6---vmware.1-tkg.1.b3d708a
    - name: worker-nodepool-a2
      replicas: 2
      vmClass: guaranteed-small
      storageClass: vwt-storage-policy
      tkr:
        reference:
          name: v1.21.6---vmware.1-tkg.1.b3d708a
  settings:
    storage:
      defaultClass: vwt-storage-policy
```

注： 上記の例では、デフォルトのクラスタ ネットワークである Antrea CNI と、クラスタ ポッドとサービスのデフォルトの CIDR 範囲を使用しています。

9 次の kubectl コマンドを実行して、クラスタをプロビジョニングします。

```
kubectl apply -f tkgs-cluster-1.yaml
```

予期される結果：

```
tanzukubernetescluster.run.tanzu.vmware.com/tkgs-cluster-1 created
```

10 kubectl を使用して、クラスタ ノードのデプロイを監視します。

```
kubectl get tanzukubernetesclusters
```

最初は、クラスタはプロビジョニング中なので、使用準備はできていません。

```
NAME             CONTROL PLANE   WORKER   TKR NAME                           AGE
READY   TKR COMPATIBLE   UPDATES AVAILABLE
tkgs-cluster-1   3               5        v1.21.6---vmware.1-tkg.1.b3d708a   2m4s
False   True
```

数分後、READY ステータスが True になります。

```
NAME             CONTROL PLANE   WORKER   TKR NAME                           AGE   READY
TKR COMPATIBLE   UPDATES AVAILABLE
tkgs-cluster-1   3               5        v1.21.6---vmware.1-tkg.1.b3d708a   13m   True
True
```

その他のガイダンスについては、kubectl を使用した Tanzu Kubernetes クラスタのステータスの監視 を参照してください。

11 クラスタ ノードのデプロイを vSphere Client で監視します。

vSphere の [ホストおよびクラスタ] のインベントリに、ターゲット vSphere 名前空間 にデプロイ中の仮想マシン ノードが表示されます。

その他のガイダンスについては、vSphere Client を使用した Tanzu Kubernetes クラスタのステータスの監視を参照してください。

12 追加の `kubectl` コマンドを実行して、クラスタのプロビジョニングを確認します。

```
kubectl get tanzukubernetescluster,cluster-
api,virtualmachinesetresourcepolicy,virtualmachineservice,virtualmachine
```

その他のガイダンスについては、Tanzu Kubernetes クラスタの操作コマンドの使用を参照してください。

トラブルシューティングについては、Tanzu Kubernetes クラスタのトラブルシューティングを参照してください。

13 kubectl 向けの vSphere プラグイン を使用して、クラスタにログインします。

```
kubectl vsphere login --server=IP-ADDRESS --vsphere-username USERNAME \
--tanzu-kubernetes-cluster-name CLUSTER-NAME \
--tanzu-kubernetes-cluster-namespace NAMESPACE-NAME
```

例：

```
kubectl vsphere login --server=192.197.2.65 --vsphere-username user@vsphere.local \
--tanzu-kubernetes-cluster-name tkgs-cluster-1 --tanzu-kubernetes-cluster-namespace tkgs-ns
```

その他のガイダンスについては、vCenter Single Sign-On ユーザーとして Tanzu Kubernetes クラスタに接続を参照してください。

14 Tanzu Kubernetes クラスタにログインできたことを確認します。

次のようなメッセージが表示されます。

```
Logged in successfully.

You have access to the following contexts:
   192.197.2.65
   tkgs-cluster-1
   tkgs-ns
```

ここで、`192.197.2.65` は スーパーバイザー クラスタ コンテキスト、`tkgs-ns` は vSphere 名前空間 コンテキスト、`tkgs-cluster-1` は Tanzu Kubernetes クラスタ コンテキストです。

15 `kubectl` で、使用可能なクラスタ コンテキストを一覧表示します。

```
kubectl config get-contexts
```

例：

```
CURRENT   NAME             CLUSTER        AUTHINFO
NAMESPACE
          192.197.2.65     192.197.2.65   wcp:192.197.2.65:administrator@vsphere.local
*         tkgs-cluster-1   192.197.2.67   wcp:192.197.2.67:administrator@vsphere.local
          tkgs-ns          192.197.2.65   wcp:192.197.2.65:administrator@vsphere.local
tkgs-ns
```

必要に応じて、`kubect config use-context tkgs-cluster-1` で Tanzu Kubernetes クラスタに切り替えて現在のコンテキストにします。

16 次の kubectl コマンドで、クラスタのプロビジョニングを確認します。

```
kubectl get nodes
```

例：

```
NAME                                                STATUS   ROLES
AGE   VERSION
tkgs-cluster-1-control-plane-6ln2h                  Ready    control-plane,master
30m   v1.21.6+vmware.1
tkgs-cluster-1-control-plane-6q67n                  Ready    control-plane,master
33m   v1.21.6+vmware.1
tkgs-cluster-1-control-plane-jw964                  Ready    control-plane,master
37m   v1.21.6+vmware.1
tkgs-cluster-1-worker-nodepool-a1-4vvkb-65494d66d8-h5fp8   Ready    <none>
32m   v1.21.6+vmware.1
tkgs-cluster-1-worker-nodepool-a1-4vvkb-65494d66d8-q4g24   Ready    <none>
33m   v1.21.6+vmware.1
tkgs-cluster-1-worker-nodepool-a1-4vvkb-65494d66d8-vdcn4   Ready    <none>
33m   v1.21.6+vmware.1
tkgs-cluster-1-worker-nodepool-a2-2n22f-bd59d7b96-nh4dg    Ready    <none>
34m   v1.21.6+vmware.1
tkgs-cluster-1-worker-nodepool-a2-2n22f-bd59d7b96-vvfmf    Ready    <none>
33m   v1.21.6+vmware.1
```

17 追加の kubectl コマンドで、クラスタのプロビジョニングを確認します。

```
kubectl get namespaces
```

```
kubectl get pods -A
```

```
kubectl cluster-info
```

```
kubectl api-resources
```

18 適切なポッド セキュリティ ポリシーを定義します。

Tanzu Kubernetes クラスタでは、PodSecurityPolicy アドミッション コントローラがデフォルトで有効になります。ガイダンスについては、Tanzu Kubernetes クラスタでのポッド セキュリティ ポリシーの使用を参照してください。

ワークロードとユーザーによっては、システム提供の PodSecurityPolicy のバインドを作成するか、カスタム PodSecurityPolicy を作成します。ポッド セキュリティ ポリシーのロール バインドの例を参照してください。

19 サンプル ワークロードをデプロイし、クラスタの作成を確認します。

ガイダンスについては、Tanzu Kubernetes クラスタへのワークロードのデプロイを参照してください。

20 TKG Extensions をデプロイして、クラスタを操作できるようにします。

ガイダンスについては、Tanzu Kubernetes クラスタへの TKG パッケージのデプロイを参照してください。

Tanzu Kubernetes クラスタの仮想マシンのクラス

Tanzu Kubernetes クラスタ ノードのサイズを設定するには、仮想マシン クラスを指定します。vSphere with Tanzu にはデフォルトのクラスが用意されていて、ユーザーは独自のクラスを作成できます。クラスを使用するには、そのクラスをターゲット vSphere 名前空間 に関連付けて、マニフェストでクラスを参照します。

仮想マシン クラスについて

仮想マシン クラスは、CPU やメモリ (RAM) などの仮想マシン (VM) の処理能力を確保するためのリソース予約の要求です。たとえば、「guaranteed-large」という名前の仮想マシン クラス タイプは、4 個の CPU と 16 GB の RAM を予約します。デフォルトの仮想マシン クラスのリストと、その対応 CPU と RAM 予約については、デフォルトの仮想マシン クラスを参照してください。

注： 仮想マシンのディスク サイズは、仮想マシン クラスの定義ではなく、OVA テンプレートによって設定されます。Tanzu Kubernetes リリース の場合、ディスク サイズは 16 GB になります。Tanzu Kubernetes リリースの配布についてを参照してください。

仮想マシン クラスには、保証型とベスト エフォート型という 2 つの予約タイプがあります。保証型のクラスは、構成したリソースをすべて予約します。つまり、指定したクラスタで `spec.policies.resources.requests` と `spec.hardware` の設定が一致します。ベスト エフォート型のクラスでは、リソースをオーバーコミットできます。本番環境のワークロードでは、保証型の仮想マシン クラス タイプの使用をお勧めします。

注意: ベスト エフォート型の仮想マシン クラス タイプではリソースをオーバーコミットできるため、Tanzu Kubernetes クラスタをプロビジョニングする vSphere 名前空間 に制限を設定していると、リソースを使い果たすおそれがあります。競合が発生し、制御プレーンが影響を受けると、クラスタの実行が停止する場合があります。そのため、本番クラスタには、必ず保証型の仮想マシン クラス タイプを使用してください。すべての本番ノードに保証型の仮想マシン クラス タイプを使用できない場合は、少なくとも制御プレーン ノードに使用してください。

仮想マシン クラスの使用

Tanzu Kubernetes クラスタで仮想マシン クラスを使用するには、このクラスタがプロビジョニングされている vSphere 名前空間 に仮想マシン クラスをバインドする必要があります。この操作を行うには、仮想マシン クラスをターゲット名前空間に関連付けます。Tanzu Kubernetes リリース 用の vSphere 名前空間 の構成を参照してください。

ターゲット vSphere 名前空間 で使用可能な仮想マシン クラスを一覧表示するには、`kubectl get virtualmachineclassbinding` コマンドを使用します。スーパーバイザー クラスタ にあるすべての仮想マシン クラスを表示するには、`kubectl describe virtualmachineclasses` コマンドを実行します。ただし、バインドされたクラスのみを使用してクラスタをプロビジョニングできるため、後者のコマンドは単なる情報用であることに注意してください。TKGS v1alpha2 API を使用して Tanzu Kubernetes クラスタをプロビジョニングするためのワークフローを参照してください。

注： 仮想マシン クラスを vSphere 名前空間 に関連付ける要件は、新しいクラスタにのみ適用されます。デフォルトの仮想マシン クラスを使用する既存の Tanzu Kubernetes クラスタは、名前空間を関連付けなくても、機能し続けます。

デフォルトの仮想マシン クラス

表 13-1. デフォルトの仮想マシン クラスには、Tanzu Kubernetes クラスタ ノードのデプロイ サイズとして使用される、デフォルトの仮想マシンのクラス タイプが一覧表示されています。

リソースのオーバーコミットを回避するには、本番ワークロードで保証されたクラス タイプを使用する必要があります。環境（開発、テスト、本番）にワークロードをデプロイしている場合にメモリ不足を回避するには、ワーカー ノードに小さなクラス サイズまたは極端に小さなクラス サイズを使用しないでください。

表 13-1. デフォルトの仮想マシン クラス

クラス	CPU	メモリ (GB)	予約済みの CPU とメモリ
guaranteed-8xlarge	32	128	はい
best-effort-8xlarge	32	128	なし
guaranteed-4xlarge	16	128	はい
best-effort-4xlarge	16	128	なし
guaranteed-2xlarge	8	64	はい
best-effort-2xlarge	8	64	なし
guaranteed-xlarge	4	32	はい
best-effort-xlarge	4	32	なし
guaranteed-large	4	16	はい
best-effort-large	4	16	なし
guaranteed-medium	2	8	はい
best-effort-medium	2	8	なし
guaranteed-small	2	4	はい
best-effort-small	2	4	なし
guaranteed-xsmall	2	2	はい
best-effort-xsmall	2	2	なし

カスタム仮想マシン クラス

vSphere with Tanzu は、Tanzu Kubernetes クラスタで使用するカスタム仮想マシン クラスをサポートします。カスタム仮想マシン クラスを定義したら、クラスタで使用する前に、ターゲット vSphere 名前空間 に関連付ける必要があります。vSphere with Tanzu での仮想マシン クラスの作成を参照してください。

仮想マシン クラスの編集

仮想マシン クラスの定義は変更できません。すべての仮想マシンは、Tanzu Kubernetes クラスタの仮想マシンのクラスを含めて、vSphere with Tanzu での仮想マシン クラスの編集または削除することができます。仮想マシン クラスを編集した場合、既存の Tanzu Kubernetes クラスタ ノードは影響を受けません。新しい Tanzu Kubernetes クラスタは、変更されたクラス定義を使用します。

注意： Tanzu Kubernetes クラスタで使用されている仮想マシン クラスを編集し、そのクラスタをスケール アウトした場合、新しいノードは編集されたクラス定義を使用しますが、既存のノードは最初のクラス定義を使用するため、クラスの不一致が発生します。

TKGS v1alpha2 API を使用した Tanzu Kubernetes クラスタのプロビジョニング

このセクションでは、TKGS v1alpha2 API を使用して Tanzu Kubernetes クラスタをプロビジョニングする方法について説明します。

TKGS v1alpha2 API を使用するための要件

Tanzu Kubernetes Grid サービス v1alpha2 API を使用して Tanzu Kubernetes クラスタをプロビジョニングするには、要件のリスト全体に従います。

TKGS v1alpha2 API を使用するための要件

Tanzu Kubernetes Grid サービス v1alpha2 API では、Tanzu Kubernetes クラスタをプロビジョニングするための堅牢な機能強化セットが提供されています。詳細については、『Tanzu Kubernetes クラスタをプロビジョニングするための TKGS v1alpha2 API』を参照してください。

Tanzu Kubernetes Grid サービス v1alpha2 API によって提供される新機能を利用するには、使用環境が次の各要件を満たしている必要があります。

要件	リファレンス
NSX-T Data Center またはネイティブ vSphere Distributed Switch のいずれかのサポート対象ネットワークで、[ワークロード管理] が有効になっています。 **注：** 機能によっては、特定のタイプのネットワークが必要になる場合があります。その場合は、この機能のトピックを参照してください。	vSphere クラスタで vSphere with Tanzu を構成するための前提条件を参照してください。 NSX-T Data Center ネットワークを使用したワークロード管理の有効化を参照してください。 vSphere ネットワークを使用したワークロード管理の有効化を参照してください。
[ワークロード管理] をホストする vCenter Server は、バージョン 7 Update 3 以降にアップデートされています。	[vCenter Server のアップデート リリースとパッチ リリース]に関するリリース ノートを参照してください。 更新手順については、vCenter Server Appliance のアップグレードを参照してください。
[ワークロード管理]が有効になっている vCenter Server クラスタをサポートするすべての ESXi ホストが、バージョン 7 Update 3 以降にアップデートされています。	[ESXi のアップデート リリースとパッチ リリースに関するリリース ノート]を参照してください。 更新手順については、ESXi ホストのアップグレードを参照してください。

要件	リファレンス
vSphere 名前空間 は、`v0.0.11` 以降に更新されています。	リリースの詳細については、vSphere with Tanzu リリース ノートを参照してください。 更新手順については、17 章 vSphere with Tanzu 環境の更新を参照してください。
スーパーバイザー クラスタ は、`v1.21.0+vmware.wcp.2` 以降に更新されています。	リリースの詳細については、vSphere with Tanzu リリース ノートを参照してください。 更新手順については、vSphere 名前空間の更新の実行による スーパーバイザー クラスタ の更新を参照してください。
Tanzu Kubernetes リリース `v1.21.2---vmware.1-tkg.1.ee25d55` 以降を使用する必要があります。	リリースの詳細については、更新のための Tanzu Kubernetes クラスタ互換性の確認を参照してください。 新しいクラスタをプロビジョニングする手順については、TKGS v1alpha2 API を使用して Tanzu Kubernetes クラスタをプロビジョニングする YAML の例を参照してください。 既存のクラスタを更新する手順については、クラスタ仕様を TKGS v1alpha2 API に変換した後の Tanzu Kubernetes リリースの更新を参照してください。
ノード制限に関する CNI の考慮事項	クラスタ仕様の設定 `spec.settings.network.pods.cidrBlocks` のデフォルトは 192.168.0.0/16 です。Tanzu Kubernetes クラスタをプロビジョニングするための TKGS v1alpha2 API を参照してください。 カスタマイズする場合、ポッドの CIDR ブロックの最小サイズは /24 です。ただし、/16 を超える `pods.cidrBlocks` サブネット マスクを制限する場合は注意が必要です。 TKGS は、`pods.cidrBlocks` から取得された /24 サブネットを各クラスタ ノードに割り当てます。この割り当てを指定するには、Kubernetes Controller Manager を選択し、NodeCIDRMaskSize という名前の NodeIPAMController パラメータを指定して、クラスタ内のノード CIDR のサブネット マスク サイズを設定します。IPv4 のデフォルトのノード サブネット マスクは /24 です。 クラスタ内の各ノードは `pods.cidrBlocks` から /24 サブネットを取得するため、プロビジョニングするクラスタに対して制限が厳しすぎるサブネット マスク サイズを使用すると、ノードの IP アドレスが不足する可能性があります。 Antrea または Calico CNI でプロビジョニングされた Tanzu Kubernetes クラスタには、次のノード制限が適用されます。 /16 == 最大 150 ノード（ConfigMax あたり） /17 == 最大 128 ノード /18 == 最大 64 ノード /19 == 最大 32 ノード /20 == 最大 16 ノード /21 == 最大 8 ノード /22 == 最大 4 ノード /23 == 最大 2 ノード /24 == 最大 1 ノード

Tanzu Kubernetes クラスタをプロビジョニングするための TKGS v1alpha2 API

Tanzu Kubernetes Grid サービス v1alpha2 API を使用すると、Tanzu Kubernetes クラスタを宣言によってプロビジョニングすることができます。クラスタを作成およびカスタマイズするには、すべてのパラメータのリストと説明、および使用に関するガイドラインを参照してください。

Tanzu Kubernetes クラスタをプロビジョニングするための Tanzu Kubernetes Grid サービス v1alpha2 API の仕様

この YAML 仕様には、Tanzu Kubernetes Grid サービス v1alpha2 API を使用して Tanzu Kubernetes クラスタをプロビジョニングする際に使用できるすべてのパラメータがリストされています。

```
apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TanzuKubernetesCluster
metadata:
  name: string
  namespace: string
spec:
  topology:
    controlPlane:
      replicas: int32
      vmClass: string
      storageClass: string
      volumes:
        - name: string
          mountPath: string
          capacity:
            storage: size in GiB
      tkr:
        reference:
          name: string
      nodeDrainTimeout: string
    nodePools:
    - name: string
      labels: map[string]string
      taints:
        -  key: string
           value: string
           effect: string
           timeAdded: time
      replicas: int32
      vmClass: string
      storageClass: string
      volumes:
        - name: string
          mountPath: string
          capacity:
            storage: size in GiB
      tkr:
        reference:
          name: string
      nodeDrainTimeout: string
  settings:
```

```
    storage:
      classes: [string]
      defaultClass: string
    network:
      cni:
        name: string
      pods:
        cidrBlocks: [string]
      services:
        cidrBlocks: [string]
      serviceDomain: string
      proxy:
        httpProxy: string
        httpsProxy: string
        noProxy: [string]
      trust:
        additionalTrustedCAs:
          - name: string
            data: string
```

Tanzu Kubernetes クラスタをプロビジョニングするための注釈付き Tanzu Kubernetes Grid サービス v1alpha2 API の仕様

この注釈付きの YAML 仕様には、Tanzu Kubernetes Grid サービス v1alpha2 API を使用して Tanzu Kubernetes クラスタをプロビジョニングする際に使用できるすべてのパラメータと、各フィールドのドキュメントがリストされています。

注： 現在は、すべての `tkr.reference.name` フィールドが一致している必要があります。今後、ノード プールを使用する別の Tanzu Kubernetes のリリースがサポートされる可能性があります。

```
apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TanzuKubernetesCluster
#metadata defines cluster information
metadata:
  #name for this Tanzu Kubernetes cluster
  name: string
  #namespace vSphere Namespace where to provision this cluster
  namespace: string
#spec defines cluster configuration
spec:
  #topology describes the number, purpose, organization
  #of nodes and the resources allocated for each
  #nodes are grouped into pools based on their purpose
  #`controlPlane` is special kind of a node pool
  #`nodePools` is for groups of worker nodes
  #each node pool is homogeneous: its nodes have the same
  #resource allocation and use the same storage
  topology:
    #controlPlane defines the topology of the cluster
    #controller, including the number of nodes and
    #the resources allocated for each
    #control plane must have an odd number of nodes
    controlPlane:
```

```
      #replicas is the number of nodes in the pool
      #the control plane can have 1 or 3 nodes
      #defaults to 1 if `nil`
      replicas: int32
      #vmClass is the name of the VirtualMachineClass
      #which describes the virtual hardware settings
      #to be used for each node in the node pool
      #vmClass controls the CPU and memory available
      #to the node and the requests and limits on
      #those resources; to list available vm classes run
      #`kubectl describe virtualmachineclasses`
      vmClass: string
      #storageClass to be used for storage of the disks
      #which store the root filesystems of the nodes
      #to list available storage classes run
      #`kubectl describe storageclasses`
      storageClass: string
      #volumes is the optional set of PVCs to create
      #and attach to each node; use for high-churn
      #control plane components such as etcd
      volumes:
        #name of the PVC to be used as the suffix (node.name)
        - name: string
          #mountPath is the directory where the volume
          #device is mounted; takes the form /dir/path
          mountPath: string
          #capacity is the PVC capacity
          capacity:
            #storage to be used for the disk
            #volume; if not specified defaults to
            #`spec.controlPlane.storageClass`
            storage: size in GiB
      #tkr.reference.name is the TKR NAME
      #to be used by control plane nodes; supported
      #format is `v1.21.2---vmware.1-tkg.1.ee25d55`
      #currently all `tkr.reference.name` fields must match
      tkr:
        reference:
          name: string
      #nodeDrainTimeout is the total amount of time
      #the controller will spend draining a node
      #the default value is 0 which means the node is
      #drained without any time limit
      nodeDrainTimeout: string
    #nodePools is an array that describes a group of
    #worker nodes in the cluster with the same configuration
    nodePools:
    #name of the worker node pool
    #must be unique in the cluster
    - name: string
      #labels are an optional map of string keys and values
      #to organize and categorize objects
      #propagated to the created nodes
      labels: map[string]string
      #taints specifies optional taints to register the
```

```
      #Node API object with; user-defined taints are
      #propagated to the created nodes
      taints:
        #key is the taint key to be applied to a node
        -  key: string
        #value is the taint value corresponding to the key
           value: string
        #effect is the effect of the taint on pods
        #that do not tolerate the taint; valid effects are
        #`NoSchedule`, `PreferNoSchedule`, `NoExecute`
           effect: string
        #timeAdded is the time when the taint was added
        #only written by the system for `NoExecute` taints
           timeAdded: time
      #replicas is the number of nodes in the pool
      #worker nodePool can have from 0 to 150 nodes
      #value of `nil` means the field is not reconciled,
      #allowing external services like autoscalers
      #to choose the number of nodes for the nodePool
      #by default CAPI's `MachineDeployment` will pick 1
      #NOTE: a cluster provisioned with 0 worker nodes/nodepools
      #is not assigned any load balancer services
      replicas: int32
      #vmClass is the name of the VirtualMachineClass
      #which describes the virtual hardware settings
      #to be used for each node in the pool
      #vmClass controls the CPU and memory available
      #to the node and the requests and limits on
      #those resources; to list available vm classes run
      #`kubectl describe virtualmachineclasses`
      vmClass: string
      #storageClass to be used for storage of the disks
      #which store the root filesystems of the nodes
      #to list available storage classes run
      #`kubectl describe ns`
      storageClass: string
      #volumes is the optional set of PVCs to create
      #and attach to each node for high-churn worker node
      #components such as the container runtime
      volumes:
        #name of this PVC to be used as the suffix (node.name)
        - name: string
          #mountPath is the directory where the volume
          #device is mounted; takes the form /dir/path
          mountPath: string
          #capacity is the PVC capacity
          capacity:
            #storage to be used for the disk
            #volume; if not specified defaults to
            #`topology.nodePools[*].storageClass`
            storage: size in GiB
      #tkr.reference.name points to the TKR NAME
      #to be used by `spec.topology.nodePools[*]` nodes; supported
      #format is `v1.21.2---vmware.1-tkg.1.ee25d55`
      #currently all `tkr.reference.name` fields must match
```

```
      tkr:
        reference:
          name: string
      #nodeDrainTimeout is the total amount of time
      #the controller will spend draining a node
      #the default value is 0 which means the node is
      #drained without any time limit
      nodeDrainTimeout: string
  #settings are optional runtime configurations
  #for the cluster, including persistent storage
  #for pods and node network customizations
  settings:
    #storage defines persistent volume (PV) storage entries
    #for container workloads; note that the storage used for
    #node disks is defined by `topology.controlPlane.storageClass`
    #and by `spec.topology.nodePools[*].storageClass`
    storage:
      #classes is a list of persistent volume (PV) storage
      #classes to expose for container workloads on the cluster
      #any class specified must be associated with the
      #vSphere Namespace where the cluster is provisioned
      #if omitted, all storage classes associated with the
      #namespace will be exposed in the cluster
      classes: [string]
      #defaultClass treats the named storage class as the default
      #for the cluster; because all namespaced storage classes
      #are exposed if specific `classes` are not named,
      #classes is not required to specify a defaultClass
      #many workloads, including TKG Extensions and Helm,
      #require a default storage class
      #if omitted, no default storage class is set
      defaultClass: string
    #netowrk defines custom networking for cluster workloads
    network:
      #cni identifies the CNI plugin for the cluster
      #use to override the default CNI set in the
      #tkgservicesonfiguration spec, or when customizing
      #network settings for the default CNI
      cni:
        #name is the name of the CNI plugin to use; supported
        #values are `antrea`, `calico`, `antrea-nsx-routed`
        name: string
      #pods configures custom networks for pods
      #defaults to 192.168.0.0/16 if CNI is `antrea` or `calico`
      #defaults to empty if CNI is `antrea-nsx-routed`
      #custom subnet size must equal or exceed /24
      #use caution before seeting CIDR range other than /16
      #cannot overlap with Supervisor Cluster workload network
      pods:
        #cidrBlocks is an array of network ranges; supplying
        #multiple ranges may not be supported by all CNI plugins
        cidrBlocks: [string]
      #services configures custom network for services
      #defaults to 10.96.0.0/12
      #cannot overlap with Supervisor Cluster workload network
```

```
    services:
      #cidrBlocks is an array of network ranges; supplying
      #multiple ranges many not be supported by all CNI plugins
      cidrBlocks: [string]
    #serviceDomain specifies the service domain for the cluster
    #defaults to `cluster.local`
    serviceDomain: string
    #proxy configures proxy server to be used inside the cluster
    #if omitted no proxy is configured
    proxy:
      #httpProxy is the proxy URI for HTTP connections
      #to endpoints outside the cluster
      #takes form `http://<user>:<pwd>@<ip>:<port>`
      httpProxy: string
      #httpsProxy is the proxy URL for HTTPS connections
      #to endpoints outside the cluster
      #takes the frorm `http://<user>:<pwd>@<ip>:<port>`
      httpsProxy: string
      #noProxy is the list of destination domain names, domains,
      #IP addresses, and other network CIDRs to exclude from proxying
      #must include Supervisor Cluster Pod, Egress, Ingress CIDRs
      noProxy: [string]
    #trust configures additional certificates for the cluster
    #if omitted no additional certificate is configured
    trust:
      #additionalTrustedCAs are additional trusted certificates
      #can be additional CAs or end certificates
      additionalTrustedCAs:
        #name is the name of the additional trusted certificate
        #must match the name used in the filename
        - name: string
          #data holds the contents of the additional trusted cert
          #PEM Public Certificate data encoded as base64 string
          #such as `LS0tLS1C...LS0tCg==` where "..." is the
          #middle section of the long base64 string
          data: string
```

TKGS v1alpha2 API を使用して Tanzu Kubernetes クラスタをプロビジョニングする YAML の例

TKGS API は、インテリジェントなデフォルト設定と、Tanzu Kubernetes クラスタをカスタマイズする一連のオプションを提供します。ニーズに合ったさまざまな構成とカスタマイズを使用して、さまざまのタイプのクラスタをプロビジョニングする例を参照してください。

デフォルトの Tanzu Kubernetes クラスタをプロビジョニングするためのサンプル YAML

次のサンプル YAML は、TKGSTanzu Kubernetes クラスタをプロビジョニングするための TKGS v1alpha2 API を使用してデフォルトの Tanzu Kubernetes クラスタをプロビジョニングします。このサンプル YAML では使用可能なすべてのデフォルト設定が使用されるため、クラスタのプロビジョニングに必要な最小構成が表されます。

```
apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TanzuKubernetesCluster
metadata:
  name: tkgs-v2-cluster-default
  namespace: tkgs-cluster-ns
spec:
  topology:
    controlPlane:
      replicas: 3
      vmClass: guaranteed-medium
      storageClass: vwt-storage-policy
      tkr:
        reference:
          name: v1.21.2---vmware.1-tkg.1.ee25d55
    nodePools:
    - name: worker-nodepool-a1
      replicas: 3
      vmClass: guaranteed-large
      storageClass: vwt-storage-policy
      tkr:
        reference:
          name: v1.21.2---vmware.1-tkg.1.ee25d55
```

カスタムの Tanzu Kubernetes クラスタをプロビジョニングするためのサンプル YAML

次のサンプル YAML は、Tanzu Kubernetes Grid サービス Tanzu Kubernetes クラスタをプロビジョニングするための TKGS v1alpha2 API を使用してカスタムの Tanzu Kubernetes クラスタをプロビジョニングします。このサンプル YAML では、変更の多いノード コンポーネントにボリュームを個別に指定し、特定のストレージおよびネットワークの設定をカスタマイズします。

```
apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TanzuKubernetesCluster
metadata:
  name: tkgs-v2-cluster-custom
  namespace: tkgs-cluster-ns
spec:
  topology:
    controlPlane:
      replicas: 3
      vmClass: guaranteed-medium
      storageClass: vwt-storage-policy
      volumes:
        - name: etcd
          mountPath: /var/lib/etcd
          capacity:
            storage: 4Gi
```

```
      tkr:
        reference:
          name: v1.21.2---vmware.1-tkg.1.ee25d55
    nodePools:
    - name: worker-nodepool-a1
      replicas: 3
      vmClass: guaranteed-large
      storageClass: vwt-storage-policy
      volumes:
        - name: containerd
          mountPath: /var/lib/containerd
          capacity:
            storage: 16Gi
      tkr:
        reference:
          name: v1.21.2---vmware.1-tkg.1.ee25d55
    - name: worker-nodepool-a2
      replicas: 2
      vmClass: guaranteed-medium
      storageClass: vwt-storage-policy
      tkr:
        reference:
          name: v1.21.2---vmware.1-tkg.1.ee25d55
    - name: worker-nodepool-a3
      replicas: 1
      vmClass: guaranteed-small
      storageClass: vwt-storage-policy
      tkr:
        reference:
          name: v1.21.2---vmware.1-tkg.1.ee25d55
  settings:
    storage:
      defaultClass: vwt-storage-policy
    network:
      cni:
        name: antrea
      services:
        cidrBlocks: ["198.53.100.0/16"]
      pods:
        cidrBlocks: ["192.0.5.0/16"]
      serviceDomain: cluster.local
      proxy:
        httpProxy: http://<user>:<pwd>@<ip>:<port>
        httpsProxy: http://<user>:<pwd>@<ip>:<port>
        noProxy: [10.246.0.0/16,192.168.144.0/20,192.168.128.0/20]
      trust:
        additionalTrustedCAs:
          - name: CompanyInternalCA-1
            data: LS0tLS1C...LS0tCg==
          - name: CompanyInternalCA-2
            data: MTLtMT1C...MT0tPg==
```

クラスタ仕様を TKGS v1alpha2 API に変換した後の Tanzu Kubernetes リリースの更新

Tanzu Kubernetes クラスタ仕様が v1alpha2 API 形式に自動変換した後、通常は、Tanzu Kubernetes リリース バージョンを変更して行う Tanzu Kubernetes クラスタのローリング アップデートを実行する際に、エラーを回避するために、クラスタ仕様の事前処理が必要な場合があります。

クラスタ仕様の自動変換

vSphere with Tanzu 環境を Tanzu Kubernetes Grid サービス v1alpha2 API に更新するには、サービスが実行されている スーパーバイザー クラスタ を更新します。

Tanzu Kubernetes Grid サービス で v1alpha2 API が実行されている場合は、既存のすべての Tanzu Kubernetes クラスタ仕様が v1alpha1 形式から v1alpha2 形式に自動的に変換されます。自動変換処理中に、システムは各クラスタ マニフェストに想定されるフィールドを作成して、ポピュレートします。API の廃止と追加 に、v1alpha2 API で新しく導入されたクラスタ仕様フィールドと廃止されたクラスタ仕様フィールドを示します。

マニフェストが v1alpha2 形式に自動的に変換されたクラスタの Tanzu Kubernetes リリース バージョンを更新するには、エラーを回避するために手動での事前処理が必要になる場合があります。クラスタ更新の例 に、さまざまなオプションを示します。

API の廃止と追加

次の表に、v1alpha2 API で廃止され、新しい設定に置き換えられたクラスタ仕様の設定を示します。

廃止された設定	新しい設定	コメント
`spec.distribution.version` `spec.distribution.fullVersion`	`spec.topology.controlPlane.tkr.refernece.name` `spec.topology.nodePools[*].tkr.reference.name`	TKR NAME 形式を使用する必要があります。例を参照してください。
`spec.topology.workers`	`spec.topology.nodePools[*]`	変換されたクラスタでは、ブロック `spec.topology.workers` が `spec.topology.nodePools[0]` になります。 `nodePools` リストの最初のエントリは `name: workers` です。
`spec.topology.controlPlane.count` `spec.topology.workers.count`	`spec.topology.controlPlane.replicas` `spec.topology.nodePools[*].replicas`	`count` は `replicas` に置き換わります
`spec.topology.controlPlane.class` `spec.topology.workers.class`	`spec.topology.controlPlane.vmClass` `spec.topology.nodePools[*].vmClass`	`class` は `vmClass` に置き換わります

廃止された設定	新しい設定	コメント
該当なし	spec.topology.nodePools[*].labels	オブジェクトを整理および分類するためのオプションのキーペア値。ラベルは、作成されたノードに伝達されます
該当なし	spec.topology.nodePools[*].taints	ノードを登録するオプションのテイント。ユーザー定義のテイントは、作成されたノードに伝達されます

TKR NAME 形式が必須

廃止される spec.distribution.version フィールドだけでなく、Tanzu Kubernetes リリース バージョンを指定する DISTRIBUTION 形式もサポートされていません。つまり、1.21.2+vmware.1-tkg.1.ee25d55、1.21.2、および 1.21 の文字列形式を使用してターゲット リリースを参照することはできません。

v1alpha2 API クラスタ仕様で Tanzu Kubernetes リリース バージョンを参照する場合は、廃止された DISTRIBUTION 形式ではなく、TKR NAME 形式を使用する必要があります。廃止された形式は UPDATES AVAILABLE 列に表示されますが、サポート対象の形式は TKR NAME 列に一覧表示されている形式のみです。

```
kubectl get tanzukubernetescluster
NAMESPACE        NAME             CONTROL PLANE    WORKER    TKR
NAME                              AGE     READY    TKR COMPATIBLE    UPDATES AVAILABLE
tkgs-cluster-1   test-cluster     3                 3         v1.21.2---vmware.1-
tkg.1.ee25d55     38h     True     True              [1.21.2+vmware.1-tkg.1.ee25d55]
```

kubectl edit を使用したクラスタ仕様の更新

TKGS v1alpha2 API に準拠させるためのクラスタ仕様の編集には、kubectl edit メソッドを使用します。この種の更新に、kubectl patch メソッドは使用しないでください。クラスタ マニフェストの編集方法を参照してください。エディタによる kubectl の構成方法については、Kubectl のデフォルトのテキスト エディタの指定を参照してください。

クラスタ更新の例

クラスタのローリング アップデートをトリガする最も一般的な方法は spec.distribution.version を変更すること（Tanzu Kubernetes クラスタの更新 を参照）ですが、このフィールドは v1alpha2 API で廃止されています。そのため、クラスタの更新に関する潜在的な問題を回避するには、いくつかの点に注意して、事前処理に関するいくつかの推奨事項に従う必要があります。

次の例は、v1alpha1 API を使用してプロビジョニングされた Tanzu Kubernetes クラスタのバージョンを v1alpha2 API を実行しているシステムに更新する例を示します。

クラスタ アップグレードの例 1：制御プレーンで単一の TKR NAME リファレンスを使用する

推奨方法は、変換された仕様からすべての nodePools[*].tkr.reference.name ブロックを削除し、ターゲット リリースの TKR NAME を使用して controlPlane.tkr.reference.name を更新することです。この場合、同じ Tanzu Kubernetes リリースがすべての nodePools[*] ノードに伝達されます。

これ以降、controlPlane と nodePools[*] で Tanzu Kubernetes リリースのバージョンが異なる場合があります。ただし、現在はクラスタ内のすべてのリリースが一致している必要があるため、単一の TKR NAME リファレンスを controlPlane に配置するだけで十分です。

例：

```
apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TanzuKubernetesCluster
metadata:
  name: tkgs-cluster-update-example1
  namespace: tkgs-cluster-ns
spec:
  settings:
    network:
      cni:
        name: antrea
      pods:
        cidrBlocks:
        - 192.0.2.0/16
      serviceDomain: cluster.local
      services:
        cidrBlocks:
        - 198.51.100.0/12
  topology:
    controlPlane:
      replicas: 3
      storageClass: vwt-storage-policy
      tkr:
        reference:
          name: v1.21.2---vmware.1-tkg.1.ee25d55
      vmClass: best-effort-medium
    nodePools:
    - name: workers
      replicas: 3
      storageClass: vwt-storage-policy
      vmClass: best-effort-medium
```

クラスタ アップグレードの例 2：各ノード プールに TKR NAME リファレンスを使用する

2 番目の例では、`controlPlane` トポロジと `nodePools[*]` トポロジの両方の `tkr.reference.name` ブロックに TKR NAME を配置します。

この方法には、Tanzu Kubernetes リリースがノード プール間で異なる可能性がある場合に、将来のリリースに対応できるという利点があります。現在、これらのリリースは一致する必要があります。

例：

```
apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TanzuKubernetesCluster
metadata:
  name: tkgs-cluster-update-example2
  namespace: tkgs-cluster-ns
spec:
  settings:
    network:
      cni:
        name: antrea
      pods:
```

```
        cidrBlocks:
        - 192.0.2.0/16
      serviceDomain: cluster.local
      services:
        cidrBlocks:
        - 198.51.100.0/12
  topology:
    controlPlane:
      replicas: 3
      storageClass: vwt-storage-policy
      vmClass: best-effort-medium
      tkr:
        reference:
          name: v1.21.2---vmware.1-tkg.1.ee25d55
    nodePools:
    - name: workers
      replicas: 3
      storageClass: vwt-storage-policy
      vmClass: best-effort-medium
      tkr:
        reference:
          name: v1.21.2---vmware.1-tkg.1.ee25d55
```

クラスタ アップグレードの例 3：廃止されたディストリビューションのフィールドを使用する

最後のオプションは、廃止されたフィールド `spec.distribution.fullVersion` と `spec.distribution.version` を使用し、すべての `tkr.reference.name` ブロックを手動で削除することです。両方のフィールドを含める必要があります。1 つのフィールドには TKR NAME 形式を使用し、もう 1 つのフィールドは null に設定します。`v1.21.2` や `v1.21` などのバージョンのショートカットはサポートされていません。

注： Ubuntu の Tanzu Kubernetes リリースでは、`spec.distribution.version` の使用はサポートされません。

次の例では、`version` フィールドに TKR NAME と null（空）値を持つ `fullVersion` を使用しています。すべての `tkr.reference.name` エントリが削除されます。

```
apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TanzuKubernetesCluster
metadata:
  name: tkgs-cluster-update-example3a
  namespace: tkgs-cluster-ns
spec:
  distribution:
    fullVersion: v1.21.2---vmware.1-tkg.1.ee25d55
    version: ""
  settings:
    network:
      cni:
        name: antrea
      pods:
        cidrBlocks:
        - 192.0.2.0/16
      serviceDomain: cluster.local
```

```
      services:
        cidrBlocks:
        - 198.51.100.0/12
  topology:
    controlPlane:
      replicas: 3
      storageClass: vwt-storage-policy
      vmClass: best-effort-medium
    nodePools:
    - name: workers
      replicas: 3
      storageClass: vwt-storage-policy
      vmClass: best-effort-medium
```

また、`fullVersion` フィールドに TKR NAME と null（空）値を持つ `version` を使用することもできます。`version` フィールドを使用している場合でも、バージョンのショートカットはサポートされません。すべての `tkr.reference.name` エントリが削除されます。

```
apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TanzuKubernetesCluster
metadata:
  name: tkgs-cluster-update-example3b
  namespace: tkgs-cluster-ns
spec:
  distribution:
    fullVersion: ""
    version: v1.21.2---vmware.1-tkg.1.ee25d55
  settings:
    network:
      cni:
        name: antrea
      pods:
        cidrBlocks:
        - 192.0.2.0/16
      serviceDomain: cluster.local
      services:
        cidrBlocks:
        - 198.51.100.0/12
  topology:
    controlPlane:
      replicas: 3
      storageClass: vwt-storage-policy
      vmClass: best-effort-medium
    nodePools:
    - name: workers
      replicas: 3
      storageClass: vwt-storage-policy
      vmClass: best-effort-medium
```

v1alpha2 API を使用して、Tanzu Kubernetes クラスタにルーティング可能なポッド ネットワークを構成する

ルーティング可能なポッド ネットワークを使用するように Tanzu Kubernetes クラスタを構成するには、`antrea-nsx-routed` をクラスタの CNI として指定します。

ルーティング可能なポッド ネットワークの概要

Kubernetes ネットワーク モデルでは、クラスタのノード ネットワーク内のポッドが、同じクラスタ内のすべてのノード上のすべてのポッドとネットワーク アドレス変換 (NAT) を使用することなく通信できる必要があります。この要件を満たすために、各 Kubernetes ポッドには、専用ポッド ネットワークから IP アドレスが割り当てられます。

`antrea` または `calico` CNI プラグインを使用して Tanzu Kubernetes クラスタをプロビジョニングすると、デフォルトのポッド ネットワーク `192.168.0.0/16` が作成されます。このサブネットは、クラスタ内でのみ一意のプライベート アドレス空間であり、インターネットではルーティングできません。`network.pods.cidrBlocks` はカスタマイズできますが、これらの CNI プラグインを使用してポッド ネットワークにルーティングすることはできません。詳細については、『Tanzu Kubernetes クラスタをプロビジョニングするための TKGS v1alpha2 API』を参照してください。

Tanzu Kubernetes Grid サービス v1alpha2 API は、`antrea-nsx-routed` CNI プラグインを使用してルーティング可能なポッド ネットワークをサポートします。このネットワーク インターフェイスは、カスタマイズされた Antrea プラグインであり、Tanzu Kubernetes クラスタでルーティング可能なポッド ネットワークをサポートするように構成されています。クラスタの仕様では、IP アドレス管理 (IPAM) が スーパーバイザー クラスタ によって処理されるようにするために、ポッドの CIDR ブロック フィールドを明示的に null にする必要があります。

ルーティング可能なポッド ネットワークを有効にすると、クラスタ外のクライアントから直接ポッドのアドレスを指定できます。また、外部のネットワーク サービスおよびサーバがソース ポッドを識別し、IP アドレスに基づいてポリシーを適用できるように、ポッドの IP アドレスが保存されます。次のようにトラフィック パターンがサポートされます。

- 同じ vSphere 名前空間 にある Tanzu Kubernetes クラスタ ポッドと vSphere ポッド の間では、トラフィックが許可されます。
- 異なる vSphere 名前空間 にある Tanzu Kubernetes クラスタ ポッドと vSphere ポッド の間では、トラフィックがドロップされます。
- スーパーバイザー クラスタ 制御プレーン ノードは、Tanzu Kubernetes クラスタ ポッドにアクセスできます。
- Tanzu Kubernetes クラスタ ポッドは、外部ネットワークにアクセスできます。
- 外部ネットワークは、Tanzu Kubernetes クラスタ ポッドにアクセスできません。Tanzu Kubernetes クラスタ ノードでは、分散ファイアウォール (DFW) の隔離ルールによってトラフィックがドロップされます。

ルーティング可能なポッドのシステム要件

ルーティング可能なポッド ネットワークを使用するには、NSX-T Data Center を使用して スーパーバイザー クラスタ を構成する必要があります。ネイティブ vSphere vDS ネットワークでは、ルーティング可能なポッドを使用することはできません。

ルーティング可能なポッドには、Tanzu Kubernetes Grid サービス v1alpha2 API が必要です。TKGS v1alpha2 API を使用するための要件を参照してください。

ルーティング可能なポッドに関する NSX-T 構成の要件

基本要件以外に、Tanzu Kubernetes クラスタでルーティング可能なポッド ネットワークを使用するための特別な NSX-T 構成は必要ありません。NSX-T を使用して vSphere U3 を実行している vSphere with Tanzu 環境には、ルーティング可能なポッド ネットワークをサポートするバージョンの NCP が含まれています。追加の NSX-T 構成は不要です。

NCP は、ルーティング可能なポッド ネットワークの IP アドレス プールを次の 2 つのソースのいずれかから作成します。

- ワークロード ネットワークが名前空間ネットワークを使用して構成されている場合、NCP は、この名前空間ネットワークに指定されている IP アドレス ブロックから 1 つ以上の IP プールを作成します。
- ワークロード ネットワークに対して名前空間ネットワークが指定されていない場合、NCP は、スーパーバイザー クラスタ ポッド CIDR から 1 つ以上の IP プールを作成します。

詳細については、『VDS ネットワークが構成されている スーパーバイザー クラスタ へのワークロード ネットワークの追加』および『NSX-T Data Center が構成されている スーパーバイザー クラスタ のワークロード ネットワーク設定の変更』を参照してください。

ルーティング可能なポッドに関する スーパーバイザー クラスタ の構成要件

Tanzu Kubernetes クラスタでルーティング可能なポッド ネットワークを使用するうえでは、基本要件以外に特別な スーパーバイザー クラスタ 構成は必要ありません。

ルーティング可能なポッド ネットワークが以下の説明に従って有効にされている場合、Tanzu Kubernetes クラスタ ポッドの CIDR は、名前空間ネットワークから作成された IP アドレス プールから、またはこれが作成されていない場合は スーパーバイザー クラスタ ポッド CIDR から割り当てられます。

クラスタ ノードに IP アドレスを割り当てる スーパーバイザー クラスタ サービス CIDR が、名前空間ネットワーク CIDR または スーパーバイザー クラスタ ポッド CIDR と重複していないことを確認する必要があります。

ルーティング可能なポッドのクラスタ構成の例

次のサンプル YAML は、ルーティング可能なポッド ネットワークを使用してクラスタを構成する方法を示します。ここで示されているのは Tanzu Kubernetes Grid サービス を呼び出すためのカスタム構成であり、v1alpha2 API を使用して Tanzu Kubernetes クラスタをプロビジョニングします。

クラスタの仕様では、ルーティング可能なポッド ネットワークを有効にするために `antrea-nsx-routed` を CNI として宣言しています。CNI が `antrea-nsx-routed` と指定されている場合、`pods.cidrBlock` フィールドは空にする必要があります。`antrea-nsx-routed` が指定されている場合、NSX-T ネットワークが使用されていないとクラスタのプロビジョニングは失敗します。

```
apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TanzuKubernetesCluster
metadata:
  name: tkgs-v2-cluster-routable-pods
  namespace: tkgs-cluster-ns
spec:
```

```
  topology:
    controlPlane:
      replicas: 3
      vmClass: guaranteed-medium
      storageClass: vwt-storage-policy
      tkr:
        reference:
          name: v1.21.2---vmware.1-tkg.1.ee25d55
    nodePools:
    - name: worker-nodepool-a1
      replicas: 3
      vmClass: guaranteed-large
      storageClass: vwt-storage-policy
      tkr:
        reference:
          name: v1.21.2---vmware.1-tkg.1.ee25d55
  settings:
    storage:
      defaultClass: vwt-storage-policy
    network:
      #`antrea-nsx-routed` is the required CNI
      #for routable pods
      cni:
        name: antrea-nsx-routed
      services:
        cidrBlocks: ["10.97.0.0/24"]
      serviceDomain: tanzukubernetescluster.local
      #`pods.cidrBlocks` value must be empty
      #when `antrea-nsx-routed` is the CNI
      pods:
        cidrBlocks:
```

TKGS v1alpha2 API の構成パラメータ

コンテナ ネットワーク インターフェイス (CNI)、プロキシ サーバ、TLS 証明書などの主要な機能のグローバル設定を使用して、Tanzu Kubernetes Grid サービス をカスタマイズできます。グローバル機能とクラスタ単位の機能を実装する場合は、トレードオフと考慮事項に注意してください。

TkgServiceConfiguration v1alpha2 仕様

`TkgServiceConfiguration` 仕様は、Tanzu Kubernetes Grid サービス インスタンスを構成するためのフィールドを提供します。

```
apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TkgServiceConfiguration
metadata:
  name: tkg-service-configuration-spec
spec:
  defaultCNI: string
  proxy:
    httpProxy: string
    httpsProxy: string
    noProxy: [string]
  trust:
```

```
    additionalTrustedCAs:
      - name: string
        data: string
  defaultNodeDrainTimeout: time
```

注意： Tanzu Kubernetes Grid サービス の構成は、グローバルに行われます。`TkgServiceConfiguration` 仕様に対して行った変更は、そのサービスによってプロビジョニングされたすべての Tanzu Kubernetes クラスタに適用されます。ローリング アップデートが手動またはアップグレードによって開始された場合は、変更されたサービス仕様によってクラスタが更新されます。

注釈付きの TkgServiceConfiguration v1alpha2 仕様

次の YAML は、`TkgServiceConfiguration` 仕様の各パラメータについて、構成可能なフィールドを示し、説明したものです。例については、Tanzu Kubernetes Grid サービス v1alpha1 API の構成の例を参照してください。

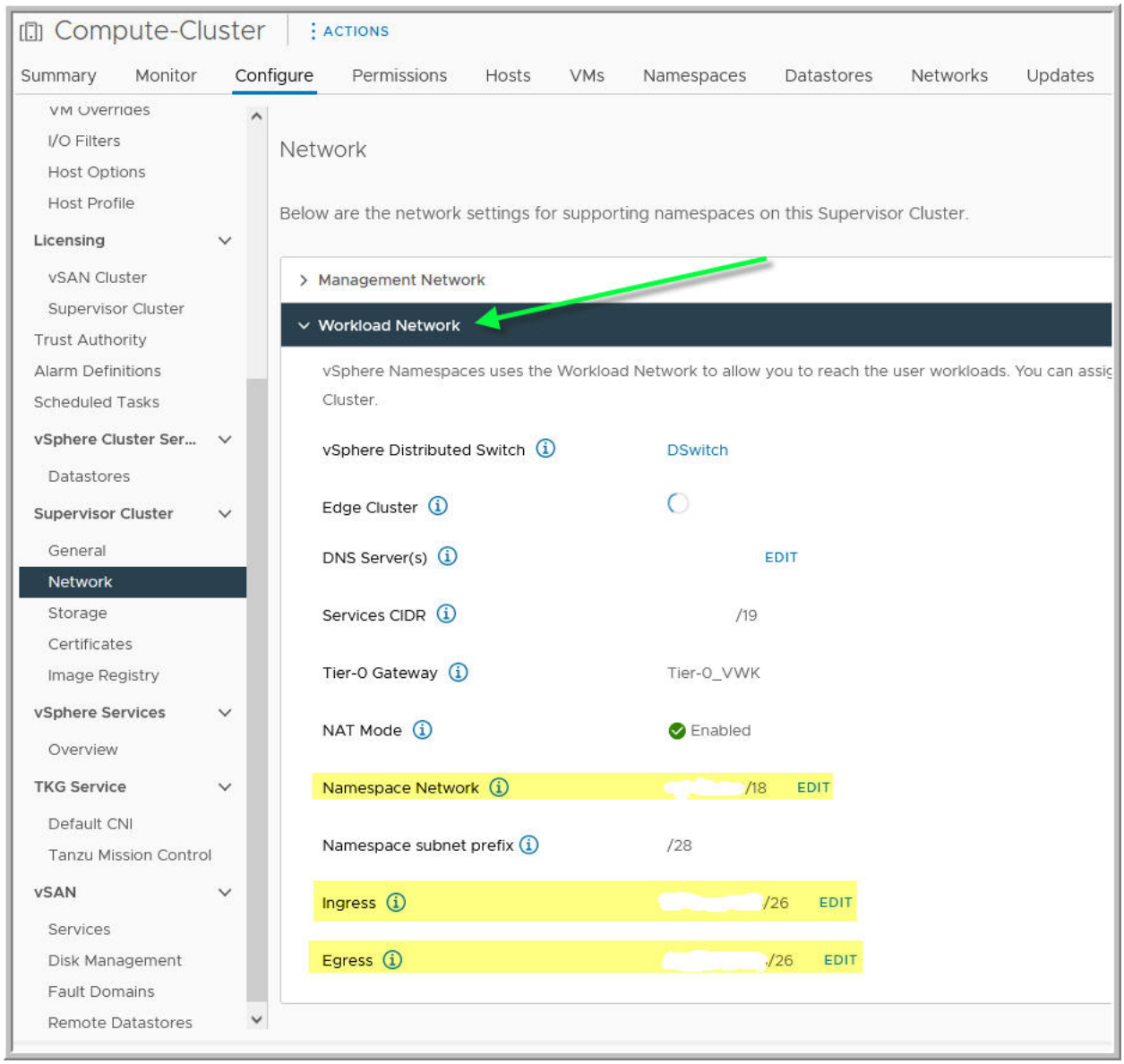
```
apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TkgServiceConfiguration
metadata:
  name: tkg-service-configuration-spec
spec:
  #defaultCNI is the default CNI for all Tanzu Kubernetes
  #clusters to use unless overridden on a per-cluster basis
  #supported values are antrea, calico, antrea-nsx-routed
  #defaults to antrea
  defaultCNI: string
  #proxy configures a proxy server to be used inside all
  #clusters provisioned by this TKGS instance
  #if implemented all fields are required
  #if omitted no proxy is configured
  proxy:
    #httpProxy is the proxy URI for HTTP connections
    #to endpionts outside the clusters
    #takes the form http://<user>:<pwd>@<ip>:<port>
    httpProxy: string
    #httpsProxy is the proxy URI for HTTPS connections
    #to endpoints outside the clusters
    #takes the frorm http://<user>:<pwd>@<ip>:<port>
    httpsProxy: string
    #noProxy is the list of destination domain names, domains,
    #IP addresses, and other network CIDRs to exclude from proxying
    #must include Supervisor Cluster Pod, Egress, Ingress CIDRs
    noProxy: [string]
  #trust configures additional trusted certificates
  #for the clusters provisioned by this TKGS instance
  #if omitted no additional certificate is configured
  trust:
    #additionalTrustedCAs are additional trusted certificates
    #can be additional CAs or end certificates
    additionalTrustedCAs:
      #name is the name of the additional trusted certificate
      #must match the name used in the filename
      - name: string
        #data holds the contents of the additional trusted cert
```

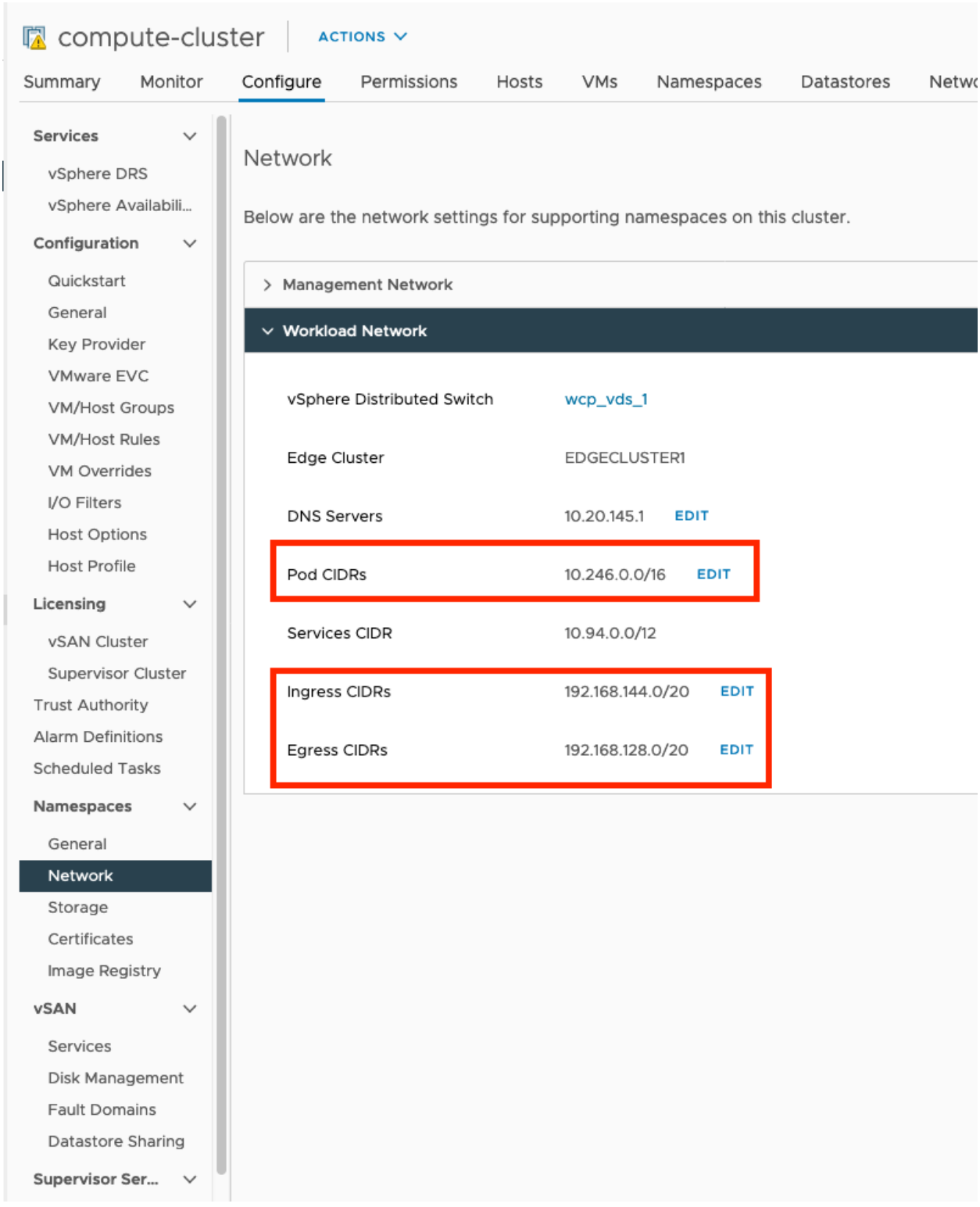
```
        #PEM Public Certificate data encoded as a base64 string
        data: string
  #defaultNodeDrainTimeout is the total amount of time the
  #controller spends draining a node; default is undefined
  #which is the value of 0, meaning the node is drained
  #without any time limitations; note that `nodeDrainTimeout`
  #is different from `kubectl drain --timeout`
  defaultNodeDrainTimeout: time
```

プロキシ サーバの構成要件

構成されているプロキシ サーバは、クラスタによって送信 HTTP および HTTPS トラフィックに使用されます。Tanzu Kubernetes クラスタのプロキシ サーバを構成するには、次の要件に留意してください。

- 必要な `proxy` パラメータは `httpProxy`、`httpsProxy`、および `noProxy` です。`proxy` スタンザを追加する場合、3 つのフィールドはすべて必須です。
- HTTP を使用してプロキシ サーバに接続できます。HTTPS 接続はサポートされていません。
- 必要な `spec.proxy.noProxy` 値は、[ワークロード ネットワーク] から取得します。[名前空間ネットワーク]（旧称はポッドの CIDR）、[Ingress]（旧称は入力方向 CIDR）、および [Egress]（旧称は出力方向 CIDR）を `noProxy` フィールドに含めてプロキシにすることはできません。次のサンプル イメージを参照してください。
- `noProxy` フィールドにサービスの CIDR を含める必要はありません。Tanzu Kubernetes クラスタは、このサブネットと連携しません。
- `noProxy` フィールドに Tanzu Kubernetes クラスタ仕様の `network.services.cidrBlocks` 値および `network.pods.cidrBlocks` 値を含める必要はありません。これらのサブネットは自動的にプロキシされません。
- `noProxy` フィールドに `localhost` および `127.0.0.1` を含める必要はありません。エンドポイントは自動的にプロキシされません。


Compute-Cluster
ACTIONS
Summary
Monitor
Configure
Permissions
Hosts
VMs
Namespaces
Datastores
Networks
Updates
VM Overrides
I/O Filters
Host Options
Host Profile
Licensing
vSAN Cluster
Supervisor Cluster
Trust Authority
Alarm Definitions
Scheduled Tasks
vSphere Cluster Ser...
Datastores
Supervisor Cluster
General
Network
Storage
Certificates
Image Registry
vSphere Services
Overview
TKG Service
Default CNI
Tanzu Mission Control
vSAN
Services
Disk Management
Fault Domains
Remote Datastores
Network
Below are the network settings for supporting namespaces on this Supervisor Cluster.
Management Network
Workload Network
vSphere Namespaces uses the Workload Network to allow you to reach the user workloads. You can assig
Cluster.
vSphere Distributed Switch
DSwitch
Edge Cluster
DNS Server(s)
EDIT
Services CIDR
/19
Tier-0 Gateway
Tier-0_VWK
NAT Mode
Enabled
Namespace Network
/18
EDIT
Namespace subnet prefix
/28
Ingress
/26
EDIT
Egress
/26
EDIT



グローバル構成オプションとクラスタ単位の構成オプションの使い分け

`TkgServiceConfiguration` は、Tanzu Kubernetes Grid サービス インスタンスによってプロビジョニングされているすべての Tanzu Kubernetes クラスタに影響するグローバルな仕様です。

TkgServiceConfiguration 仕様を編集する前に、グローバル構成よりも、クラスタ単位の代替方法のほうが使用事例に適合する可能性について確認してください。

表 13-2. グローバル構成オプションとクラスタ単位の構成オプション

設定	グローバル オプション	クラスタ単位のオプション
デフォルトの CNI	TkgServiceConfiguration 仕様を編集します。Tanzu Kubernetes Grid サービス v1alpha1 API の構成の例を参照してください。	クラスタ仕様で CNI を指定します。たとえば、デフォルトの CNI は Antrea です。Calico を使用するには、クラスタの YAML 内で指定します。Tanzu Kubernetes Grid サービス v1alpha1 API を使用した Tanzu Kubernetes クラスタのプロビジョニングの例を参照してください
プロキシ サーバ	TkgServiceConfiguration 仕様を編集します。Tanzu Kubernetes Grid サービス v1alpha1 API の構成の例を参照してください。	クラスタ仕様にプロキシ サーバ構成パラメータを含めます。Tanzu Kubernetes Grid サービス v1alpha1 API を使用した Tanzu Kubernetes クラスタのプロビジョニングの例を参照してください。
証明書の信頼	TkgServiceConfiguration 仕様を編集します。外部コンテナ レジストリと証明書ベース プロキシ構成の 2 つの使用事例があります。Tanzu Kubernetes Grid サービス v1alpha1 API の構成の例を参照してください	はい。カスタム証明書をクラスタ単位で含めることも、クラスタ仕様でグローバルに設定された trust 設定をオーバーライドすることもできます。Tanzu Kubernetes Grid サービス v1alpha1 API を使用した Tanzu Kubernetes クラスタのプロビジョニングの例を参照してください。

注： TkgServiceConfiguration でグローバル プロキシが構成されている場合、そのプロキシ情報は、クラスタの初期デプロイ後にクラスタ マニフェストに伝達されます。グローバル プロキシ構成は、クラスタの作成時にプロキシ構成フィールドがない場合にのみ、クラスタ マニフェストに追加されます。つまり、クラスタごとの構成が優先されるため、グローバル プロキシ構成は上書きされます。詳細については、『Tanzu Kubernetes Grid サービス v1alpha1 API の構成パラメータ』を参照してください。

TkgServiceConfiguration 仕様を編集する前に、設定をグローバル レベルで適用する場合の影響を確認してください。

フィールド	適用	追加/変更した場合の既存のクラスタへの影響	クラスタ作成時のクラスタ単位のオーバーライド	クラスタ更新時のクラスタ単位のオーバーライド
defaultCNI	グローバル	なし	可能。クラスタ作成時にグローバル設定をオーバーライドできます	不可能。既存のクラスタの CNI は変更できません。クラスタ作成時にグローバルに設定されたデフォルトの CNI を使用した場合は、変更できません
proxy	グローバル	なし	可能。クラスタ作成時にグローバル設定をオーバーライドできます	可能。U2+ を使用している場合は、クラスタ更新時にグローバル設定をオーバーライドできます
trust	グローバル	なし	可能。クラスタ作成時にグローバル設定をオーバーライドできます	可能。U2+ を使用している場合は、クラスタ更新時にグローバル設定をオーバーライドできます

既存のクラスタへのグローバル構成変更の伝達

TkgServiceConfiguration でグローバル レベルの設定を行っても、既存のクラスタにはその設定が自動的に伝達されません。たとえば、TkgServiceConfiguration で proxy や trust の設定を変更しても、すでにプロビジョニングされているクラスタには影響がありません。

既存のクラスタにグローバルな変更を伝達するには、Tanzu Kubernetes クラスタにパッチを適用して、TkgServiceConfiguration に対する変更をクラスタに継承させる必要があります。

例：

```
kubectl patch tkc <CLUSTER_NAME> -n <NAMESPACE> --type merge -p "{\"spec\":{\"settings\":
{\"network\":{\"proxy\": null}}}}"
```

```
kubectl patch tkc <CLUSTER_NAME> -n <NAMESPACE> --type merge -p "{\"spec\":{\"settings\":
{\"network\":{\"trust\": null}}}}"
```

v1alpha2 API を使用した TKGS インスタンスの構成の例

コンテナ ネットワーク インターフェイス、プロキシ サーバ、および TLS 証明書のグローバル設定を使用して、Tanzu Kubernetes Grid サービス v1alpha2 API をカスタマイズする例を参照してください。

v1alpha2 API を使用した Tanzu Kubernetes Grid サービス の構成

Tanzu Kubernetes Grid サービス をカスタマイズするには、デフォルトの CNI を変更し、グローバル プロキシ サーバを追加し、信頼できる証明書を追加します。Tanzu Kubernetes Grid サービス v1alpha1 API の構成パラメータを参照してください。

```
apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TkgServiceConfiguration
metadata:
  name: tkg-service-v2-configuration-example
spec:
  defaultCNI: antrea
```

```
proxy:
  #supported format is `http://<user>:<pwd>@<ip>:<port>`
  httpProxy: http://admin:PaSsWoRd@10.66.100.22:80
  httpsProxy: http://admin:PaSsWoRd@10.66.100.22:80
  noProxy: [10.246.0.0/16,192.168.144.0/20,192.168.128.0/20]
trust:
  additionalTrustedCAs:
    - name: CompanyInternalCA-1
      data: LS0tLS1C...LS0tCg==
      #where "..." is the middle section of the long base64 string
    - name: CompanyInternalCA-2
      data: MTLtMT1C...MT0tPg==
defaultNodeDrainTimeout: 0
```

注意： Tanzu Kubernetes Grid サービス 仕様を編集すると、新しいクラスタや、手動または自動でアップグレードされる既存のクラスタなど、このサービスによってプロビジョニングされた全クラスタがグローバルに変更されます。

前提条件：Kubectl 編集の構成

Tanzu Kubernetes クラスタを拡張するには、`kubectl edit tanzukubernetescluster/CLUSTER-NAME` コマンドを使用してクラスタ マニフェストを更新します。kubectl edit コマンドを実行すると、KUBE_EDITOR または EDITOR の環境変数によって定義されたテキスト エディタ内にクラスタ マニフェストが開きます。環境変数を設定する手順については、Kubectl のデフォルトのテキスト エディタの指定を参照してください。

仕様の変更を保存すると、`kubectl` から、編集内容が正常に記録されたことが報告されます。キャンセルするには、保存せずにエディタを閉じます。

デフォルト CNI の構成

Tanzu Kubernetes Grid サービス は、Tanzu Kubernetes クラスタにデフォルトのコンテナ ネットワーク インターフェイス (CNI) を提供します。デフォルトの構成では、CNI を指定せずにクラスタを作成できます。デフォルトの CNI を変更するには、サービスの仕様を編集します。

Tanzu Kubernetes Grid サービス は、Antrea と Calico という 2 つの CNI をサポートしています。デフォルトは Antrea です。詳細については、『Tanzu Kubernetes Grid サービス クラスタ ネットワーク 』を参照してください。

デフォルトの CNI をオーバーライドするには、使用する CNI を明示的に指定します。また、CNI の TKG サービス コントローラを編集して、デフォルトの CNI を変更することもできます。

1 スーパーバイザー クラスタ で認証します。

```
kubectl vsphere login --server=SVC-IP-ADDRESS --vsphere-username USERNAME
```

2 コンテキストをターゲット vSphere 名前空間 に切り替えます。

```
kubectl config use-context tkgs-cluster-ns
```

3 デフォルトの CNI を一覧表示します。

```
kubectl get tkgserviceconfigurations
```

結果の例：

```
NAME                                DEFAULT CNI
tkg-service-configuration           antrea
```

4 Tanzu Kubernetes Grid サービス 仕様の編集内容をロードします。

```
kubectl edit tkgserviceconfigurations tkg-service-configuration
```

`tkg-service-configuration` 仕様が、KUBE_EDITOR または EDITOR 環境変数によって定義されたデフォルトのテキスト エディタで開かれます。

5 `spec.defaultCNI` の値を編集します。

たとえば、次の値からの変更を考えます。

```
spec:
  defaultCNI: antrea
```

この値を次のように変更します。

```
spec:
  defaultCNI: calico
```

6 変更を適用するには、テキスト エディタでこのファイルを保存します。キャンセルするには、保存せずにエディタを閉じます。

テキスト エディタで変更を保存すると、kubectl によって `tkg-service-configuration` サービス仕様が更新されます。

7 デフォルトの CNI が更新されていることを確認します。

```
kubectl get tkgserviceconfigurations
```

デフォルトの CNI が更新されます。デフォルトのネットワーク設定によってプロビジョニングされたクラスタでは、デフォルトの CNI が使用されます。

```
NAME                                DEFAULT CNI
tkg-service-configuration           calico
```

グローバル プロキシ サーバの構成

グローバル プロキシ サーバを有効にするには、TkgServiceConfiguration にプロキシ サーバのパラメータを追加します。必須フィールドの説明については、Tanzu Kubernetes Grid サービス v1alpha1 API の構成パラメータを参照してください。

1 スーパーバイザー クラスタ で認証します。

```
kubectl vsphere login --server=SVC-IP-ADDRESS --vsphere-username USERNAME
```

2 コンテキストをターゲット vSphere 名前空間 に切り替えます。

```
kubectl config use-context tkgs-cluster-ns
```

3 現在の構成を取得します。

```
kubectl get tkgserviceconfigurations
```

結果の例：

```
NAME                          DEFAULT CNI
tkg-service-configuration     antrea
```

4 Tanzu Kubernetes Grid サービス 仕様の編集内容をロードします。

```
kubectl edit tkgserviceconfigurations tkg-service-configuration
```

tkg-service-configuration 仕様が、KUBE_EDITOR または EDITOR 環境変数によって定義されたデフォルトのテキスト エディタで開かれます。

5 httpProxy、httpsProxy、noProxy などの必須フィールドをそれぞれ含む spec.proxy サブセクションを追加します。

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TkgServiceConfiguration
metadata:
  ...
  name: tkg-service-configuration-example
  resourceVersion: "44170525"
  selfLink: /apis/run.tanzu.vmware.com/v1alpha1/tkgserviceconfigurations/tkg-service-
configuration
  uid: 10347195-5f0f-490e-8ae1-a758a724c0bc
spec:
  defaultCNI: antrea
  proxy:
    httpProxy: http://<user>:<pwd>@<ip>:<port>
    httpsProxy: http://<user>:<pwd>@<ip>:<port>
    noProxy: [SVC-POD-CIDRs, SVC-EGRESS-CIDRs, SVC-INGRESS-CIDRs]
```

6 各プロキシ フィールドに適切な値をポピュレートします。各フィールドの説明については、Tanzu Kubernetes Grid サービス v1alpha1 API の構成パラメータを参照してください。

noProxy フィールドに必要な値は、スーパーバイザー クラスタ の [ワークロード ネットワーク] から取得されます。これらの値を取得する場所については、上記のトピックの図を参照してください。

例：

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TkgServiceConfiguration
metadata:
  ...
  name: tkg-service-configuration-example
  resourceVersion: "44170525"
  selfLink: /apis/run.tanzu.vmware.com/v1alpha1/tkgserviceconfigurations/tkg-service-
configuration
  uid: 10347195-5f0f-490e-8ae1-a758a724c0bc
spec:
  defaultCNI: antrea
  proxy:
    httpProxy: http://user:password@10.186.102.224:3128
    httpsProxy: http://user:password@10.186.102.224:3128
    noProxy: [10.246.0.0/16,192.168.144.0/20,192.168.128.0/20]
```

7 変更を適用するには、テキスト エディタでこのファイルを保存します。キャンセルするには、保存せずにエディタを閉じます。

テキスト エディタで変更を保存すると、tkg-service-configuration は kubectl サービス仕様で定義された構成を使用して Tanzu Kubernetes Grid サービス を更新します。

8 Tanzu Kubernetes Grid サービス がプロキシ設定を使用して更新されていることを確認します。

```
kubectl get tkgserviceconfigurations -o yaml
```

9 確認するには、Tanzu Kubernetes クラスタをプロビジョニングします。TKGS v1alpha2 API を使用して Tanzu Kubernetes クラスタをプロビジョニングするためのワークフローを参照してください。

次のコマンドを使用して、クラスタがプロキシを使用していることを確認します。

```
kubectl get tkc CLUSTER-NAME -n NAMESPACE -o yaml
```

証明書ベースのプロキシ構成

環境によっては、プロキシ サーバを使用してインターネット トラフィックをルーティングすることが困難な場合があります。たとえば、金融機関などの規制の厳しい業界の企業では、すべてのインターネット トラフィックが企業のプロキシを経由する必要があります。

Tanzu Kubernetes Grid サービス を構成すると、送信 HTTP/S トラフィックにプロキシ サーバを使用するように Tanzu Kubernetes クラスタをプロビジョニングできます。詳細については、『Tanzu Kubernetes Grid サービス v1alpha1 API の構成パラメータ』を参照してください。

この例に示すように、プロキシ サーバの信頼できる証明書を TkgServiceConfiguration 仕様に追加できます。

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TkgServiceConfiguration
metadata:
  name: tkg-service-configuration-example
spec:
  defaultCNI: antrea
  proxy:
    httpProxy: http://user:password@10.186.102.224:3128
    httpsProxy: http://user:password@10.186.102.224:3128
    noProxy: [10.246.0.0/16,192.168.144.0/20,192.168.128.0/20]
  trust:
    additionalTrustedCAs:
      - name: first-cert-name
        data: base64-encoded string of a PEM encoded public cert 1
      - name: second-cert-name
        data: base64-encoded string of a PEM encoded public cert 2
  defaultNodeDrainTimeout: 0
```

外部プライベート レジストリの構成

外部プライベート レジストリを Tanzu Kubernetes クラスタに接続するためのカスタム証明書を使用して、Tanzu Kubernetes Grid サービス を構成できます。詳細については、『Tanzu Kubernetes クラスタでの外部コンテナ レジストリの使用』を参照してください。

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TkgServiceConfiguration
metadata:
  name: tkg-service-configuration-example
spec:
  defaultCNI: antrea
  trust:
    additionalTrustedCAs:
      - name: harbor-vm-cert
        data: <<<base64-encoded string of a PEM encoded public cert>>>>
```

TKGS v1alpha2 API を使用した Tanzu Kubernetes クラスタのスケーリング

ノード数を変更するか、ノードをホストする仮想マシン クラスを垂直方向に変更することで、Tanzu Kubernetes クラスタを水平方向にスケーリングできます。

サポートされているスケーリング操作

次の表に、Tanzu Kubernetes クラスタでサポートされているスケーリング操作を示します。

表 13-3. Tanzu Kubernetes クラスタでサポートされているスケーリング操作

ノード	水平方向のスケール アウト	水平方向のスケール イン	垂直方向のスケーリング	ボリュームのスケーリング
制御プレーン	はい	なし	はい	なし
ワーカー	はい	はい	はい	はい

次の考慮事項に留意してください。

- クラスタ ノードを垂直方向にスケーリングするときに、使用可能なリソースが不足していると、ノード上でワークロードを実行できなくなる可能性があります。このため、水平方向にスケーリングする方法が推奨されることがあります。
- 仮想マシン クラスは変更できません。Tanzu Kubernetes クラスタで使用される仮想マシン クラスを編集した後に、このクラスタをスケール アウトした場合、新しいクラスタ ノードは更新されたクラス定義を使用しますが、既存のクラスタ ノードは初期のクラス定義を引き続き使用するため、不一致が生じます。Tanzu Kubernetes クラスタの仮想マシンのクラスを参照してください。
- ワーカー ノードのボリュームは、プロビジョニング後に変更できますが、制御プレーン ノードのボリュームは変更できません。

スケーリングの前提条件：Kubectl 編集の構成

Tanzu Kubernetes クラスタを拡張するには、`kubectl edit tanzukubernetescluster/CLUSTER-NAME` コマンドを使用してクラスタ マニフェストを更新します。kubectl edit コマンドを実行すると、KUBE_EDITOR または EDITOR の環境変数によって定義されたテキスト エディタ内にクラスタ マニフェストが開きます。環境変数を設定する手順については、Kubectl のデフォルトのテキスト エディタの指定を参照してください。

マニフェストの変更を保存すると、`kubectl` から編集内容が正常に記録されたことが報告され、この変更内容を使用してクラスタが更新されます。

```
kubectl edit tanzukubernetescluster/tkgs-cluster-1
tanzukubernetescluster.run.tanzu.vmware.com/tkgs-cluster-1 edited
```

キャンセルするには、保存せずにエディタを閉じます。

```
kubectl edit tanzukubernetescluster/tkgs-cluster-1
Edit cancelled, no changes made.
```

制御プレーンのスケール アウト

制御プレーン ノードの数を 1 から 3 に増やすことで、Tanzu Kubernetes クラスタをスケール アウトできます。制御プレーン ノードの数は奇数にする必要があります。制御プレーンではスケーリングできません。

1 スーパーバイザー クラスタ で認証します。

```
kubectl vsphere login --server=SVC-IP-ADDRESS --vsphere-username USERNAME
```

2 Tanzu Kubernetes クラスタが実行されている vSphere 名前空間 にコンテキストを切り替えます。

```
kubectl config use-context tkgs-cluster-ns
```

3 この名前空間で実行されている Kubernetes クラスタを一覧表示します。

```
kubectl get tanzukubernetescluster -n tkgs-cluster-ns
```

4 ターゲット クラスタで実行されているノードの数を取得します。

```
kubectl get tanzukubernetescluster tkgs-cluster-1
```

たとえば、次のクラスタには、1 台の制御プレーン ノードと、3 台のワーカー ノードが含まれています。

```
NAMESPACE             NAME             CONTROL PLANE   WORKER   TKR
NAME                            AGE     READY
tkgs-cluster-ns       test-cluster     1               3        v1.21.2---vmware.1-
tkg.1.13da849   5d12h   True
```

5 `kubectl edit` コマンドを使用して、編集するクラスタ マニフェストをロードします。

```
kubectl edit tanzukubernetescluster/tkgs-cluster-1
```

クラスタ マニフェストは、KUBE_EDITOR または EDITOR 環境変数によって定義されたテキスト エディタで開かれます。

6 `spec.topology.controlPlane.count` パラメータを特定し、ノードの数を 1 から 3 に増やします。

```
...
controlPlane:
    replicas: 1
...
```

```
...
ControlPlane:
    replicas: 3
...
```

7 変更を適用するには、テキスト エディタでこのファイルを保存します。キャンセルするには、保存せずにエディタを閉じます。

ファイルを保存すると、kubectl によって変更がクラスタに適用されます。バックグラウンドでは、スーパーバイザー クラスタ の 仮想マシン サービス によって新しいワーカー ノードがプロビジョニングされます。

8 新しいノードが追加されていることを確認します。

```
kubectl get tanzukubernetescluster tkgs-cluster-1
```

スケール アウト制御プレーンのノード数が 3 になりました。

```
NAMESPACE             NAME             CONTROL PLANE   WORKER   TKR
NAME                            AGE     READY
tkgs-cluster-ns       test-cluster     3               3        v1.21.2---vmware.1-
tkg.1.13da849   5d12h   True
```

ワーカー ノードのスケール アウト

kubectl を使用してワーカー ノードの数を増やすことで、Tanzu Kubernetes クラスタをスケール アウトできます。

1 スーパーバイザー クラスタ で認証します。

```
kubectl vsphere login --server=SVC-IP-ADDRESS --vsphere-username USERNAME
```

2 Tanzu Kubernetes クラスタが実行されている vSphere 名前空間 にコンテキストを切り替えます。

```
kubectl config use-context tkgs-cluster-ns
```

3 この名前空間で実行されている Kubernetes クラスタを一覧表示します。

```
kubectl get tanzukubernetescluster -n tkgs-cluster-ns
```

4 ターゲット クラスタで実行されているノードの数を取得します。

```
kubectl get tanzukubernetescluster tkgs-cluster-1
```

たとえば、次のクラスタには、3 台の制御プレーン ノードと、3 台のワーカー ノードが含まれています。

```
NAMESPACE             NAME             CONTROL PLANE   WORKER   TKR
NAME                            AGE     READY
tkgs-cluster-ns       test-cluster     3               3        v1.21.2---vmware.1-
tkg.1.13da849   5d12h   True
```

5 `kubectl edit` コマンドを使用して、編集するクラスタ マニフェストをロードします。

```
kubectl edit tanzukubernetescluster/tkgs-cluster-1
```

クラスタ マニフェストは、KUBE_EDITOR または EDITOR 環境変数によって定義されたテキスト エディタで開かれます。

6 `spec.topology.workers.count` パラメータを特定し、ノードの数を増やします。

```
...
workers:
    replicas: 3
...
```

```
...
workers:
    replicas: 4
...
```

7 変更を適用するには、テキスト エディタでこのファイルを保存します。キャンセルするには、保存せずにエディタを閉じます。

ファイルを保存すると、kubectl によって変更がクラスタに適用されます。バックグラウンドでは、スーパーバイザー クラスタ の 仮想マシン サービス によって新しいワーカー ノードがプロビジョニングされます。

8 新しいワーカー ノードが追加されていることを確認します。

```
kubectl get tanzukubernetescluster tkgs-cluster-1
```

スケール アウト後、クラスタ内のワーカー ノード数は 4 になります。

```
NAMESPACE              NAME             CONTROL PLANE   WORKER   TKR
NAME                             AGE     READY
tkgs-cluster-ns        test-cluster     3               4        v1.21.2---vmware.1-
tkg.1.13da849   5d12h   True
```

ワーカー ノードのスケール イン

ワーカー ノードの数を減らすことで、Tanzu Kubernetes クラスタをスケール インできます。制御プレーンのスケーリングはサポートされていません。

1 スーパーバイザー クラスタ で認証します。

```
kubectl vsphere login --server=SVC-IP-ADDRESS --vsphere-username USERNAME
```

2 Tanzu Kubernetes クラスタが実行されている vSphere 名前空間 にコンテキストを切り替えます。

```
kubectl config use-context tkgs-cluster-ns
```

3 この名前空間で実行されている Kubernetes クラスタを一覧表示します。

```
kubectl get tanzukubernetescluster -n tkgs-cluster-ns
```

4 ターゲット クラスタで実行されているノードの数を取得します。

```
kubectl get tanzukubernetescluster tkgs-cluster-1
```

たとえば、次のクラスタには、3 台の制御プレーン ノードと、4 台のワーカー ノードが含まれています。

```
NAMESPACE             NAME             CONTROL PLANE   WORKER   TKR
NAME                            AGE     READY
tkgs-cluster-ns       test-cluster     3               4        v1.21.2---vmware.1-
tkg.1.13da849   5d12h   True
```

5 `kubectl edit` コマンドを使用して、編集するクラスタ マニフェストをロードします。

```
kubectl edit tanzukubernetescluster/tkgs-cluster-1
```

クラスタ マニフェストは、KUBE_EDITOR または EDITOR 環境変数によって定義されたテキスト エディタで開かれます。

6 `spec.topology.workers.count` パラメータを特定し、ノードの数を減らします。

```
...
workers:
    replicas: 4
...
```

```
...
workers:
    replicas: 2
...
```

7 変更を適用するには、テキスト エディタでこのファイルを保存します。キャンセルするには、保存せずにエディタを閉じます。

ファイルを保存すると、kubectl によって変更がクラスタに適用されます。バックグラウンドでは、スーパーバイザー クラスタ の 仮想マシン サービス によって新しいワーカー ノードがプロビジョニングされます。

8 ワーカー ノードが削除されたことを確認します。

```
kubectl get tanzukubernetescluster tkgs-cluster-1
```

スケール イン後、クラスタ内のワーカー ノード数は 2 になります。

```
NAMESPACE             NAME             CONTROL PLANE   WORKER   TKR
NAME                            AGE     READY
tkgs-cluster-ns       test-cluster     3               2        v1.21.2---vmware.1-
tkg.1.13da849   5d12h   True
```

クラスタの垂直方向のスケーリング

クラスタ ノードのホストに使用されている仮想マシンのクラスを変更することで、Tanzu Kubernetes クラスタを垂直方向にスケーリングできます。垂直方向のスケーリングは、制御プレーンノードとワーカー ノードの両方でサポートされています。

Tanzu Kubernetes Grid サービス は、サービスに組み込まれているローリング アップデート メカニズムを使用して、クラスタ ノードの垂直方向のスケーリングをサポートします。`VirtualMachineClass` の定義を変更すると、この新しいクラスを使用して新しいノードがロール アウトされて、古いノードがスピン ダウンされます。Tanzu Kubernetes クラスタの更新を参照してください。

1 スーパーバイザー クラスタ で認証します。

```
kubectl vsphere login --server=SVC-IP-ADDRESS --vsphere-username USERNAME
```

2 Tanzu Kubernetes クラスタが実行されている vSphere 名前空間 にコンテキストを切り替えます。

```
kubectl config use-context tkgs-cluster-ns
```

3 この名前空間で実行されている Kubernetes クラスタを一覧表示します。

```
kubectl get tanzukubernetescluster -n tkgs-cluster-ns
```

4 ターゲット Tanzu Kubernetes クラスタを記述し、仮想マシンのクラスを確認します。

```
kubectl describe tanzukubernetescluster tkgs-cluster-2
```

たとえば、次のクラスタでは best-effort-medium 型の仮想マシン クラスが使用されています。

```
Spec:
  ...
  Topology:
    Control Plane:
      Class:          best-effort-medium
      ...
    nodePool-a1:
      Class:          best-effort-medium
      ...
```

5 使用可能な仮想マシン クラスを一覧表示して、記述します。

```
kubectl get virtualmachineclassbinding
```

```
kubectl describe virtualmachineclassbinding
```

注： 使用する仮想マシン クラスを、vSphere 名前空間 にバインドする必要があります。Tanzu Kubernetes クラスタの仮想マシンのクラスを参照してください。

6 ターゲット クラスタのマニフェスト開いて編集します。

```
kubectl edit tanzukubernetescluster/tkgs-cluster-2
```

クラスタ マニフェストは、KUBE_EDITOR または EDITOR 環境変数によって定義されたテキスト エディタで開かれます。

7 仮想マシン クラスを変更して、マニフェストを編集します。

たとえば、制御プレーン ノードとワーカー ノードに guaranteed-large 仮想マシン クラスを使用するように、クラスタ マニフェストを編集します。

```
spec:
  topology:
    controlPlane:
      class: guaranteed-large
      ...
    nodePool-a1:
      class: guaranteed-large
      ...
```

8 変更を適用するには、テキスト エディタでこのファイルを保存します。キャンセルするには、保存せずにエディタを閉じます。

ファイルを保存すると、kubectl によって変更がクラスタに適用されます。Tanzu Kubernetes Grid サービス はバックグラウンドで新しいノードをプロビジョニングし、古いノードを削除します。ローリング アップデート プロセスの説明については、Tanzu Kubernetes Grid サービス クラスタの更新についてを参照してください。

9 クラスタが更新されたことを確認します。

```
kubectl get tanzukubernetescluster
NAMESPACE             NAME             CONTROL PLANE   WORKER   TKR
NAME                              AGE     READY
tkgs-cluster-ns       test-cluster     3               3        v1.21.2---vmware.1-
tkg.1.13da849   5d12h   True
```

ノード ボリュームのスケーリング

ノードの Tanzu Kubernetes クラスタ仕様には、1 つ以上のパーシステント ボリュームを宣言するオプションがあります。ノード ボリュームの宣言は、制御プレーンの ectd データベースやワーカー ノードのコンテナ ランタイムなど、変更の多いコンポーネントに役立ちます。リファレンスのために、これらの両方のノード ボリュームが宣言されたクラスタ仕様の抜粋を以下に示します。(クラスタ仕様の完全な例は、TKGS v1alpha2 API を使用して Tanzu Kubernetes クラスタをプロビジョニングする YAML の例にあります。)

クラスタの作成後に 1 つ以上のノード ボリュームを追加または変更する場合は、次の考慮事項に留意してください。

ボリューム ノード	説明
ワーカー ノードのボリュームは変更できます。	Tanzu Kubernetes クラスタがプロビジョニングされた後に、ワーカー ノード ボリュームを追加または更新できます。ローリング アップデートを開始すると、クラスタは新しいボリュームまたは変更されたボリュームを使用して更新されます。 **注意:** 新しいボリュームまたは変更されたボリュームを含むワーカー ノードをスケーリングすると、ローリング アップデート中に現在のボリュームのデータが削除されます。
制御プレーン ノードのボリュームは変更できません。	Tanzu Kubernetes クラスタがプロビジョニングされた後に、制御プレーン ノードのボリュームを追加または更新することはできません。Kubernetes クラスタ API (CAPI) では、クラスタの作成後に `spec.toplogy.controlPlane.volumes` を変更することはできません。クラスタの作成後に制御プレーン ボリュームを追加または変更しようとすると、要求は拒否され、「変更できないフィールドの更新は許可されていません」というエラー メッセージが表示されます。

```
spec:
  topology:
    controlPlane:
      replicas: 3
      vmClass: guaranteed-medium
      storageClass: vwt-storage-policy
      volumes:
        - name: etcd
          mountPath: /var/lib/etcd
          capacity:
            storage: 4Gi
      tkr:
        reference:
          name: v1.21.2---vmware.1-tkg.1.ee25d55
    nodePools:
    - name: worker-nodepool-a1
      replicas: 3
      vmClass: guaranteed-large
      storageClass: vwt-storage-policy
      volumes:
        - name: containerd
          mountPath: /var/lib/containerd
          capacity:
            storage: 16Gi
```

Tanzu Kubernetes Grid サービス v1alpha1 API を使用した Tanzu Kubernetes クラスタのプロビジョニング

このセクションでは、Tanzu Kubernetes Grid サービス v1alpha1 API を使用して Tanzu Kubernetes クラスタをプロビジョニングする方法について説明します。

Tanzu Kubernetes Grid サービス v1alpha1 API を使用して Tanzu Kubernetes クラスタをプロビジョニングするためのワークフロー

Tanzu Kubernetes クラスタをプロビジョニングするには、kubectl と、YAML で定義したクラスタ仕様を使用して、Tanzu Kubernetes Grid サービス 宣言型 API を呼び出します。クラスタをプロビジョニングしたら、クラスタを操作し、kubectl を使用してワークロードをクラスタにデプロイします。

このワークフローは、クラスタのプロビジョニング プロセスに対するエンドツーエンドの手順を示します。各ステップには、特定のタスクの詳細についてのリンクが含まれています。

前提条件

次の前提条件を満たすようにします。

- スーパーバイザー クラスタ インスタンスを構成している。5 章 スーパーバイザー クラスタ の構成と管理を参照してください。
- Tanzu Kubernetes クラスタをプロビジョニングする vSphere 名前空間 を作成している。vSphere 名前空間 の作成と設定を参照してください。

  名前空間の初期構成には、以下が必要です。

  - 1 人以上の DevOps エンジニアが vCenter Single Sign-On 認証情報を使用して名前空間にアクセスできるように編集権限が追加されている。
  - 名前空間のタグベースの共有ストレージ ポリシー。
  - 名前空間の容量と使用量の割り当てが検証され、必要に応じて調整されている。
- 共有データストアに Tanzu Kubernetes リリース 用のコンテンツ ライブラリを作成し、使用するリリースを同期している。Tanzu Kubernetes リリース のコンテンツ ライブラリの作成と管理を参照してください。
- コンテンツ ライブラリおよび仮想マシン クラスに vSphere 名前空間 を関連付けている。Tanzu Kubernetes リリース 用の vSphere 名前空間 の構成を参照してください。

手順

1. vSphere 向け Kubernetes CLI Tools をダウンロードしてインストールします。vSphere 向け Kubernetes CLI Tools のダウンロードとインストールを参照してください。
2. kubectl 向けの vSphere プラグイン を使用して、スーパーバイザー クラスタ での認証を行います。vCenter Single Sign-On ユーザーとして スーパーバイザー クラスタ に接続するを参照してください。

   ```
   kubectl vsphere login --server=IP-ADDRESS --vsphere-username USERNAME
   ```
3. kubectl を使用して、Tanzu Kubernetes クラスタをプロビジョニングする vSphere 名前空間 にコンテキストを切り替えます。

   ```
   kubectl config get-contexts
   ```

   ```
   kubectl config use-context SUPERVISOR-NAMESPACE
   ```

   例：

   ```
   kubectl config use-context tkgs-cluster-ns
   ```

4 使用可能な仮想マシン クラスのバインドを一覧表示します。Tanzu Kubernetes クラスタの仮想マシンのクラスを参照してください。

次のコマンドを使用して、クラスタをデプロイする vSphere 名前空間 で使用可能なすべての仮想マシン クラスのバインドを一覧表示します。

```
kubectl get virtualmachineclassbindings
```

注： `kubectl get virtualmachineclasses` コマンドは、スーパーバイザー クラスタ にあるすべての仮想マシン クラスを一覧表示します。仮想マシン クラスと vSphere 名前空間 を関連付ける必要がある場合は、ターゲット名前空間にバインドされているこれらの仮想マシン クラスのみを使用します。

5 名前空間を記述して、使用可能なデフォルト ストレージ クラスを取得します。

```
kubectl describe namespace SUPERVISOR-NAMESPACE
```

6 使用可能な Tanzu Kubernetes リリース を一覧表示します。

注： 互換性については、Tanzu Kubernetes リリース のリストを参照してください。更新のための Tanzu Kubernetes クラスタ互換性の確認を参照してください。

```
kubectl get tanzukubernetesreleases
```

注： `kubectl get virtualmachineimages` コマンドは、仮想マシンに関する一般情報を返します。

7 Tanzu Kubernetes クラスタをプロビジョニングするための YAML ファイルを作成します。

a いずれかのサンプル YAML ファイルを使用して開始します。Tanzu Kubernetes Grid サービス v1alpha1 API を使用した Tanzu Kubernetes クラスタのプロビジョニングの例を参照してください。

たとえば、次の YAML ファイルは、使用可能なすべてのデフォルト設定を使用して、最小のクラスタをプロビジョニングします。

```
apiVersion: run.tanzu.vmware.com/v1alpha1  #TKGS API endpoint
kind: TanzuKubernetesCluster               #required parameter
metadata:
  name: tkgs-cluster-1                     #cluster name, user defined
  namespace: tgks-cluster-ns               #vsphere namespace
spec:
  distribution:
    version: v1.19                         #Resolves to latest TKR 1.19 version
  topology:
    controlPlane:
      count: 1                             #number of control plane nodes
      class: best-effort-medium            #vmclass for control plane nodes
      storageClass: vwt-storage-policy     #storageclass for control plane
    workers:
      count: 3                             #number of worker nodes
      class: best-effort-medium            #vmclass for worker nodes
      storageClass: vwt-storage-policy     #storageclass for worker nodes
```

b 前述のコマンドの出力から取得した情報を使用して、名前空間、ストレージ クラス、仮想マシン クラスを含むクラスタ YAML にポピュレートします。

c クラスタ構成パラメータの完全なリストを参照して、必要に応じてクラスタをカスタマイズします。Tanzu Kubernetes Grid サービス v1alpha1 API を使用する Tanzu Kubernetes クラスタの構成パラメータを参照してください。

d ファイルを `tkgs-cluster-1.yaml` などの名前で保存します。

8 次の kubectl コマンドを実行して、クラスタをプロビジョニングします。

```
kubectl apply -f CLUSTER-NAME.yaml
```

例：

```
kubectl apply -f tkgs-cluster-1.yaml
```

予期される結果：

```
tanzukubernetescluster.run.tanzu.vmware.com/tkgs-cluster-1 created
```

9 kubectl を使用して、クラスタ ノードのデプロイを監視します。kubectl を使用した Tanzu Kubernetes クラスタのステータスの監視 を参照してください。

```
kubectl get tanzukubernetesclusters
```

結果の例：

```
NAME              CONTROL PLANE   WORKER   DISTRIBUTION                       AGE     PHASE
tkgs-cluster-2    1               3        v1.19.7+vmware.1-tkg.1.c40d30d     7m59s   running
```

10 vSphere Client を使用して、クラスタ ノードのデプロイを監視します。vSphere Client を使用した Tanzu Kubernetes クラスタのステータスの監視を参照してください。

たとえば、vSphere インベントリに、名前空間にデプロイされている仮想マシン ノードが表示されます。

11 追加のコマンドを実行して、クラスタのプロビジョニングを確認します。Tanzu Kubernetes Cluster の操作コマンドの使用を参照してください。

例：

```
kubectl get tanzukubernetescluster,cluster-
api,virtualmachinesetresourcepolicy,virtualmachineservice,virtualmachine
```

注： その他のトラブルシューティングについては、Tanzu Kubernetes クラスタのトラブルシューティングを参照してください。

12 kubectl 向けの vSphere プラグイン を使用して、クラスタにログインします。vCenter Single Sign-On ユーザーとして Tanzu Kubernetes クラスタに接続を参照してください。

```
kubectl vsphere login --server=IP-ADDRESS --vsphere-username USERNAME \
--tanzu-kubernetes-cluster-name CLUSTER-NAME --tanzu-kubernetes-cluster-namespace
NAMESPACE-NAME
```

13 次の kubectl コマンドを使用して、クラスタのプロビジョニングを確認します。

```
kubectl cluster-info
```

```
kubectl get nodes
```

```
kubectl get namespaces
```

```
kubectl api-resources
```

14 サンプル ワークロードをデプロイし、クラスタの作成を確認します。Tanzu Kubernetes クラスタへのワークロードのデプロイを参照してください。

注： Tanzu Kubernetes クラスタではポッド セキュリティ ポリシーが有効になっています。ワークロードとユーザーによっては、適切な RoleBinding またはカスタムの PodSecurityPolicy を作成する必要がある場合があります。Tanzu Kubernetes クラスタでのポッド セキュリティ ポリシーの使用を参照してください。

15 TKG Extensions をデプロイして、クラスタを操作できるようにします。Tanzu Kubernetes クラスタへの TKG パッケージのデプロイを参照してください。

Tanzu Kubernetes Grid サービス v1alpha1 API を使用する Tanzu Kubernetes クラスタの構成パラメータ

Tanzu Kubernetes Grid サービス 宣言型 API は、Tanzu Kubernetes クラスタを構成するためのパラメータをいくつか公開しています。クラスタをプロビジョニングおよびカスタマイズするには、すべてのパラメータのリストと説明、および使用に関するガイドラインを参照してください。

Tanzu Kubernetes クラスタをプロビジョニングするための注釈付き YAML

注釈付き YAML には、Tanzu Kubernetes クラスタをプロビジョニングする際に使用可能なすべてのパラメータと、各フィールドの概要コメントが示されています。

注： クラスタをプロビジョニングする場合は、注釈付き YAML は検証されません。このガイダンスの例については、Tanzu Kubernetes Grid サービス v1alpha1 API を使用した Tanzu Kubernetes クラスタのプロビジョニングの例を参照してください。

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TanzuKubernetesCluster
metadata:
  name: <tanzu kubernetes cluster name>
  namespace: <vsphere namespace where the cluster will be provisioned>
spec:
  distribution:
    version: <tanzu kubernetes release version string: full, point, short>
  topology:
    controlPlane:
      count: <integer either 1 or 3>
      class: <vm class bound to the target vsphere namespace>
      storageClass: <vsphere storage policy bound to the target vsphere namespace>
      volumes: #optional setting for high-churn control plane component (such as etcd)
        - name: <user-defined string>
          mountPath: </dir/path>
          capacity:
            storage: <size in GiB>
    workers:
      count: <integer from 0 to 150>
      class: <vm class bound to the target vsphere namespace>
      storageClass: <vsphere storage policy bound to the target vsphere namespace>
      volumes: #optional setting for high-churn worker node component (such as containerd)
        - name: <user-defined string>
          mountPath: </dir/path>
          capacity:
            storage: <size in GiB>
  settings: #all spec.settings are optional
    storage: #optional storage settings
      classes: [<array of kubernetes storage classes for dynamic pvc provisioning>]
      defaultClass: <default kubernetes storage class>
    network: #optional network settings
      cni: #override default cni set in the tkgservicesonfiguration spec
        name: <antrea or calico>
      pods: #custom pod network
        cidrBlocks: [<array of pod cidr blocks>]
```

```
    services: #custom service network
      cidrBlocks: [<array of service cidr blocks>]
    serviceDomain: <custom service domain>
    proxy: #proxy server for outbound connections
      httpProxy: http://<IP:PORT>
      httpsProxy: http://<IP:PORT>
      noProxy: [<array of CIDRs to not proxy>]
    trust: #trust fields for custom public certs for tls
      additionalTrustedCAs:
        - name: <first-cert-name>
          data: <base64-encoded string of PEM encoded public cert 1>
        - name: <second-cert-name>
          data: <base64-encoded string of PEM encoded public cert 2>
```

Tanzu Kubernetes クラスタをプロビジョニングするためのパラメータ

次の表に、Tanzu Kubernetes クラスタのプロビジョニングに使用できるすべてのパラメータおよび使用できる値のリストと説明を示します。例については、Tanzu Kubernetes Grid サービス v1alpha1 API の構成の例を参照してください。

表 13-4. Tanzu Kubernetes クラスタをプロビジョニングするためのパラメータ

名前	値	説明
apiVersion	run.tanzu.vmware.com/v1alpha1	Tanzu Kubernetes Grid サービス API のバージョンを指定します。
kind	TanzuKubernetesCluster	作成する Kubernetes リソースのタイプを指定します。使用できる値は TanzuKubernetesCluster のみです（大文字と小文字を区別）。
metadata	クラスタ メタデータのセクション	name や namespace などのクラスタ メタデータが含まれています。これは標準の Kubernetes メタデータであるため、name の代わりに generateName を使用したり、ラベルや注釈を追加したりすることができます。
name	英数字とダッシュを使用するユーザー定義の文字列（例：my-tkg-cluster-1）	作成するクラスタの名前を指定します。現在のクラスタの名前付けの制約： ■ 名前の長さは 41 文字以内にする必要があります。 ■ 名前は文字で始まる必要があります。 ■ 名前には、文字、数字、ハイフンを含めることができます。 ■ 名前は文字または数字で終わる必要があります。
namespace	英数字とダッシュを使用するユーザー定義の文字列（例：my-sns-1）	クラスタがデプロイされるスーパーバイザー ネームスペースの名前を識別します。これは、スーパーバイザー クラスタ内にあるスーパーバイザー ネームスペースへの参照です。
spec	クラスタの技術仕様のセクション	ノードの topology や Kubernetes ソフトウェアの distribution など、クラスタのエンド状態の仕様を宣言型形式で示します。

表 13-4. Tanzu Kubernetes クラスタをプロビジョニングするためのパラメータ （続き）

名前	値	説明
distribution	Tanzu Kubernetes リリースのバージョンを指定するセクション	クラスタのディストリビューションを指定します。これは、制御プレーンとワーカー ノードにインストールされている Tanzu Kubernetes クラスタ ソフトウェアで、Kubernetes 自体も含みます。
version	Kubernetes のバージョンを表す、ダッシュを含む英数字の文字列（例：v1.20.2+vmware.1-tkg.1、v1.20.2、v1.20）	セマンティックなバージョン表記を使用して、クラスタ ノードにインストールする Kubernetes ディストリビューションのソフトウェア バージョンを指定します。完全修飾バージョンを指定するか、ショートカットを使用することができます。「version: v1.20.2」は、そのパッチ バージョンに一致する最新のイメージに解決され、「version: v1.20」は該当する最新のパッチ バージョンに解決されます。クラスタの作成後、その説明に解決後のバージョンが「fullVersion」として表示されます。
topology	クラスタ ノード トポロジのセクション	クラスタ ノードの数、目的、および編成と、それぞれに割り当てられたリソースを示すフィールドを含んでいます。クラスタ ノードは、目的に応じて control-plane または worker のプールにグループ化されます。各プールは同種で、同じリソース割り当てを持ち、同じストレージを使用します。
controlPlane	制御プレーン設定のセクション	ノードの数 (count)、仮想マシンのタイプ (class)、各ノードに割り当てられるストレージ リソース (storageClass) など、クラスタ制御プレーンのトポロジを指定します。
count	1 または 3 のいずれかの整数	制御プレーン ノードの数を指定します。制御プレーンには奇数個のノードが必要です。
class	列挙セットからの文字列形式のシステム定義要素（例：guaranteed-small、best-effort-large）	プール内の各ノードで使用される仮想ハードウェア設定を示す VirtualMachineClass の名前を指定します。これにより、ノード (CPU とメモリ) で使用できるハードウェア、およびこれらのリソースの要求と制限が制御されます。Tanzu Kubernetes クラスタの仮想マシンのクラスを参照してください。

表 13-4. Tanzu Kubernetes クラスタをプロビジョニングするためのパラメータ （続き）

名前	値	説明
`storageClass`	`node-storage`（例）	制御プレーン ノードのルート ファイル システムを格納するディスクのストレージに使用されるストレージ クラスを識別します。使用可能なストレージ クラスを表示するには、名前空間で `kubectl describe ns` を実行します。名前空間で使用可能なストレージ クラスは、vSphere 管理者が設定するストレージによって異なります。スーパーバイザー ネームスペースに関連付けられているストレージ クラスがクラスタに複製されます。つまり、ストレージ クラスがこのフィールドで有効な値になるためには、スーパーバイザー ネームスペースで使用できる必要があります。7 章 vSphere 名前空間の構成と管理を参照してください。
`volumes`	オプションのストレージ設定 ■ volumes： 　■ name：*string* 　■ mountPath：*/dir/path* 　■ キャパシティ 　　■ storage：GiB サイズ	制御プレーン ノードの etcd のディスクとストレージのパラメータを個別に指定できます。ディスク パラメータとストレージ パラメータを個別に使用するクラスタの例を参照してください。
`workers`	ワーカー ノード設定のセクション	ノードの数 (`count`)、仮想マシンのタイプ (`class`)、各ノードに割り当てられるストレージ リソース (`storageClass`) など、クラスタ ワーカー ノードのトポロジを指定します。
`count`	0 ～ 150 の整数（例：`1`、`2`、`7`）	クラスタ内のワーカー ノードの台数を指定します。ワーカー ノード数がゼロのクラスタを作成することで、制御プレーン ノードのみを持つクラスタを使用できます。ワーカー ノードの数には、厳密な上限はありませんが、妥当な制限は 150 です。 **注：** 0 台のワーカー ノードを使用してプロビジョニングされたクラスタには、ロード バランサ サービスが割り当てられません。
`class`	列挙セットからの文字列形式のシステム定義要素（例：`guaranteed-small`、`best-effort-large`）	プール内の各ノードで使用される仮想ハードウェア設定を示す VirtualMachineClass の名前を指定します。これにより、ノード (CPU とメモリ) で使用できるハードウェア、およびこれらのリソースの要求と制限が制御されます。Tanzu Kubernetes クラスタの仮想マシンのクラスを参照してください。

表 13-4. Tanzu Kubernetes クラスタをプロビジョニングするためのパラメータ （続き）

名前	値	説明
`storageClass`	`node-storage`（例）	ワーカー ノードのルート ファイル システムを格納するディスクのストレージに使用されるストレージ クラスを識別します。使用可能なストレージ クラスを一覧表示するには、名前空間で `kubectl describe ns` を実行します。名前空間で使用可能なストレージ クラスは、vSphere 管理者が設定するストレージによって異なります。スーパーバイザー ネームスペースに関連付けられているストレージ クラスがクラスタに複製されます。つまり、有効にするためには、スーパーバイザー ネームスペースでストレージ クラスを使用できる必要があります。7 章 vSphere 名前空間の構成と管理を参照してください。
`volumes`	オプションのストレージ設定 ■ volumes： ■ name：*string* ■ mountPath：*/dir/path* ■ キャパシティ ■ storage：GiB サイズ	ワーカー ノードのコンテナ イメージのディスクとストレージのパラメータを個別に指定できます。ディスク パラメータとストレージ パラメータを個別に使用するクラスタの例を参照してください。
`settings`	クラスタ固有の設定に関するセクション（すべての `spec.settings` はオプション）	クラスタのランタイム構成情報（オプション）を識別します。これには、ノードの `network` の詳細とポッドのパーシステント `storage` が含まれます。
`storage`	ストレージを指定するセクション	コンテナ ワークロードのパーシステント ボリューム (PV) ストレージ エントリを識別します。
`classes`	1 つ以上のユーザー定義文字列の配列（例：`["gold", "silver"]`）	コンテナ ワークロードの名前付きパーシステント ボリューム (PV) ストレージ クラスを指定します。スーパーバイザー ネームスペースに関連付けられているストレージ クラスがクラスタに複製されます。つまり、有効な値であるためには、スーパーバイザー ネームスペースでストレージ クラスを使用できる必要があります。ストレージ クラスとパーシステント ボリュームのデフォルト クラスを使用するクラスタの例を参照してください。
`defaultClass`	`silver`（例）	クラスタ内でデフォルトとして注釈が付けられる名前付きストレージ クラスを指定します。これを指定しないと、デフォルトはありません。`defaultClass` を指定するために 1 つまたは複数の `classes` を指定する必要はありません。一部のワークロードでは、Helm などのデフォルト クラスが必要な場合があります。ストレージ クラスとパーシステント ボリュームのデフォルト クラスを使用するクラスタの例を参照してください。

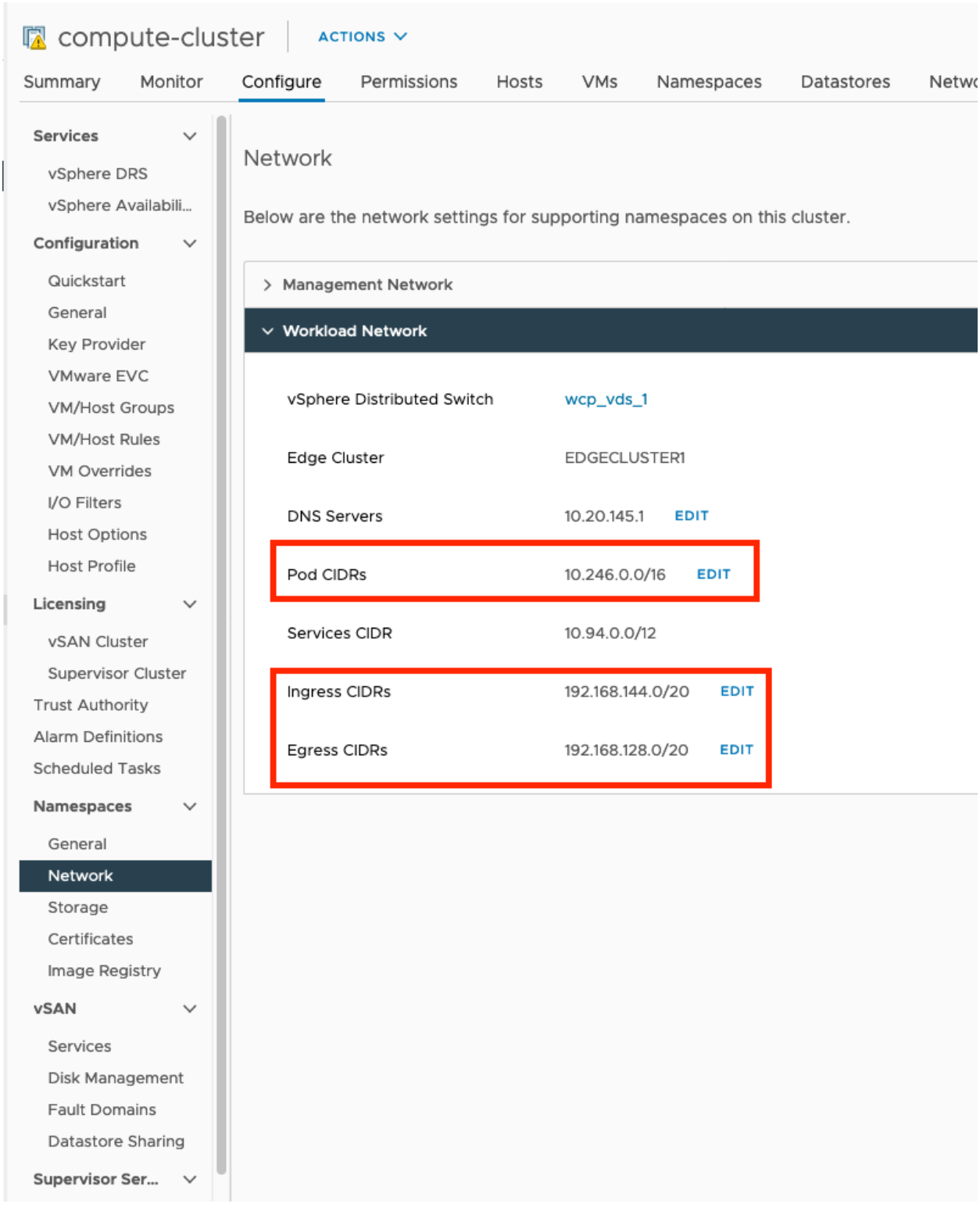
表 13-4. Tanzu Kubernetes クラスタをプロビジョニングするためのパラメータ （続き）

名前	値	説明
network	ネットワーク設定のセクション マーカー	クラスタのネットワーク関連の設定を指定します。
cni	CNI を指定するためのセクション マーカー	クラスタのコンテナ ネットワーク インターフェイス (CNI) プラグインを識別します。デフォルトは Antrea です。新しいクラスタに指定する必要はありません。
name	文字列 antrea または calico	使用する CNI を指定します。Antrea と Calico がサポートされています。システム構成により、Antrea がデフォルトの CNI として設定されます。デフォルトの CNI は変更できます。デフォルトを使用する場合、このフィールドを指定する必要はありません。
services	Kubernetes サービス サブネットを指定するセクション マーカー	Kubernetes サービスのネットワーク設定を識別します。デフォルトは 10.96.0.0/12 です。
cidrBlocks	配列 ["198.51.100.0/12"]（例）	Kubernetes サービスで使用する IP アドレスの範囲を指定します。デフォルトは 10.96.0.0/12 です。スーパーバイザー クラスタ用に選択された設定と重複しないようにしてください。このフィールドは複数の範囲を指定できる配列ですが、現在は IP アドレス範囲を 1 つのみ使用できます。Tanzu Kubernetes Grid サービス v1alpha1 API を使用した Tanzu Kubernetes クラスタのプロビジョニングの例のネットワークの例を参照してください。
pods	Kubernetes ポッドのサブネットを指定するセクション マーカー	ポッドのネットワーク設定を指定します。デフォルトは 192.168.0.0/16 です。最小ブロック サイズは /24 です。
cidrBlocks	配列 ["192.0.2.0/16"]（例）	Kubernetes ポッドで使用する IP アドレスの範囲を指定します。デフォルトは 192.168.0.0/16 です。スーパーバイザー クラスタ用に選択された設定と重複しないようにしてください。ポッドのサブネット サイズは /24 以上にする必要があります。このフィールドは複数の範囲を指定できる配列ですが、現在は IP アドレス範囲を 1 つのみ使用できます。Tanzu Kubernetes Grid サービス v1alpha1 API を使用した Tanzu Kubernetes クラスタのプロビジョニングの例のネットワークの例を参照してください。
serviceDomain	"cluster.local"	クラスタのサービス ドメインを指定します。デフォルトは cluster.local です。

表 13-4. Tanzu Kubernetes クラスタをプロビジョニングするためのパラメータ （続き）

名前	値	説明
proxy	クラスタの HTTP(S) プロキシ設定を指定するセクション。実装されている場合は、すべてのフィールドが必須です。	指定したプロキシ設定のフィールドに入力します。グローバル プロキシが設定され、クラスタ プロキシが個別に設定されていない場合は、自動的にポピュレートされます。プロキシ サーバを使用するクラスタの例を参照してください。
httpProxy	http://<user>:<pwd>@<ip>:<port>	クラスタ外の HTTP 接続の作成に使用するプロキシ URL を指定します。
httpsProxy	http://<user>:<pwd>@<ip>:<port>	クラスタ外の HTTPS 接続の作成に使用するプロキシ URL を指定します。
noProxy	プロキシにしない CIDR ブロックの配列（[10.246.0.0/16,192.168.144.0/20, 192.168.128.0/20] など）。 スーパーバイザー クラスタのワークロード ネットワークから必要な値を取得します（Pod CIDRs、Ingress CIDRs、および Egress CIDRs）。 noProxy 配列フィールドに含める値については、次の図を参照してください。	スーパーバイザー クラスタ上のワークロード ネットワークで使用されるサブネット（ポッド、Ingress、および Egress）をプロキシにすることはできません。 noProxy フィールドにスーパーバイザー クラスタのサービス CIDR を含める必要はありません。Tanzu Kubernetes クラスタは、これらのサービスと連携しません。 エンドポイント localhost と 127.0.0.1 は自動的にプロキシされません。これらを noProxy フィールドに追加する必要はありません。 Tanzu Kubernetes クラスタのポッド CIDR およびサービス CIDR は、自動的にプロキシされません。これらを noProxy フィールドに追加する必要はありません。 プロキシ サーバを使用するクラスタの例を参照してください。
trust	trust パラメータのセクション マーカー。	データを受け入れない。
additionalTrustedCAs	name と data の証明書の配列をそれぞれ受け入れます。	データを受け入れない。
name	文字列	TLS 証明書の名前。
data	文字列	PEM でエンコードされたパブリック証明書の base64 エンコード文字列。

図に示すように、スーパーバイザー クラスタ の [ワークロード ネットワーク] から必要な noProxy 値を取得します。


compute-cluster
ACTIONS
Summary
Monitor
Configure
Permissions
Hosts
VMs
Namespaces
Datastores
Netwo
Services
vSphere DRS
vSphere Availabili...
Configuration
Quickstart
General
Key Provider
VMware EVC
VM/Host Groups
VM/Host Rules
VM Overrides
I/O Filters
Host Options
Host Profile
Licensing
vSAN Cluster
Supervisor Cluster
Trust Authority
Alarm Definitions
Scheduled Tasks
Namespaces
General
Network
Storage
Certificates
Image Registry
vSAN
Services
Disk Management
Fault Domains
Datastore Sharing
Supervisor Ser...
Network
Below are the network settings for supporting namespaces on this cluster.
Management Network
Workload Network
vSphere Distributed Switch
wcp_vds_1
Edge Cluster
EDGECLUSTER1
DNS Servers
10.20.145.1
EDIT
Pod CIDRs
10.246.0.0/16
EDIT
Services CIDR
10.94.0.0/12
Ingress CIDRs
192.168.144.0/20
EDIT
Egress CIDRs
192.168.128.0/20
EDIT

Tanzu Kubernetes Grid サービス v1alpha1 API を使用した Tanzu Kubernetes クラスタのプロビジョニングの例

Tanzu Kubernetes Grid サービス API は、インテリジェントなデフォルト設定と、Tanzu Kubernetes クラスタをカスタマイズする一連のオプションを提供します。ニーズに合ったさまざまな構成とカスタマイズを使用して、さまざまのタイプのクラスタをプロビジョニングする例を参照してください。

Tanzu Kubernetes クラスタをプロビジョニングするための最小限の YAML

次のサンプル YAML は、Tanzu Kubernetes Grid サービス を呼び出して、すべてのデフォルト設定を使用する Tanzu Kubernetes クラスタをプロビジョニングするために必要な最小限の構成です。

最小限必要なサンプル YAML の特性は、次のとおりです。

- 「v1.19」と示されている Tanzu Kubernetes リリース のバージョンは、`v1.19.7+vmware.1-tkg.1.xxxxxx` などのマイナー バージョンと一致する最新のディストリビューションに解決されます。更新のための Tanzu Kubernetes クラスタ互換性の確認を参照してください。
- 仮想マシン クラス `best-effort-<size>` に予約がありません。詳細については、『Tanzu Kubernetes クラスタの仮想マシンのクラス』を参照してください。
- クラスタにコンテナのパーシステント ストレージは含まれません。必要に応じて、`spec.settings.storage` で設定します。次のストレージの例を参照してください。
- Helm などの一部のワークロードでは、`spec.settings.storage.defaultClass` が必要になることがあります。次のストレージの例を参照してください。
- `spec.settings.network` セクションは指定されていません。つまり、クラスタは次のデフォルトのネットワーク設定を使用します。
  - デフォルトの CNI：`antrea`
  - デフォルトのポッド CIDR：`192.168.0.0/16`
  - デフォルトのサービス CIDR：`10.96.0.0/12`
  - デフォルトのサービス ドメイン：`cluster.local`

注： ポッドのデフォルトの IP アドレス範囲は、`192.168.0.0/16` です。このサブネットがすでに使用されている場合は、別の CIDR 範囲を指定する必要があります。以下のカスタム ネットワークの例を参照してください。

```
apiVersion: run.tanzu.vmware.com/v1alpha1      #TKGS API endpoint
kind: TanzuKubernetesCluster                   #required parameter
metadata:
  name: tkgs-cluster-1                         #cluster name, user defined
  namespace: tgks-cluster-ns                   #vsphere namespace
spec:
  distribution:
    version: v1.20                             #Resolves to latest TKR 1.20
  topology:
    controlPlane:
```

```
      count: 1                                  #number of control plane nodes
      class: best-effort-medium                 #vmclass for control plane nodes
      storageClass: vwt-storage-policy          #storageclass for control plane
    workers:
      count: 3                                  #number of worker nodes
      class: best-effort-medium                 #vmclass for worker nodes
      storageClass: vwt-storage-policy          #storageclass for worker nodes
```

ディスク パラメータとストレージ パラメータを個別に使用するクラスタ

次のサンプル YAML は、クラスタ制御プレーン ノードとワーカー ノードにディスク パラメータとストレージ パラメータを個別に使用してクラスタをプロビジョニングする方法を示しています。

変更の多いデータのディスク パラメータとストレージ パラメータを分離すると、特に、リンク クローンの使用に関連する読み取り/書き込みオーバーヘッドを最小限にするのに有効です。主に 2 つのユースケースがあります。

- 制御プレーン ノードで etcd データベースのストレージ パフォーマンスをカスタマイズする
- ワーカー ノードでコンテナ イメージのディスク サイズをカスタマイズする

この例の特性は次のとおりです。

- `spec.topology.controlPlane.volumes` 設定で、etcd データベースに個別のボリュームを指定します。
- `spec.topology.workers.volumes` 設定で、コンテナ イメージに個別のボリュームを指定します。
- コンテナ イメージの `mountPath: /var/lib/containerd` は、Tanzu Kubernetes リリース 1.17 以降でサポートされています。

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TanzuKubernetesCluster
metadata:
  name: tkgs-cluster-5
  namespace: tgks-cluster-ns
spec:
  distribution:
    version: v1.20
  topology:
    controlPlane:
      count: 3
      class: best-effort-medium
      storageClass: vwt-storage-policy
      volumes:
        - name: etcd
          mountPath: /var/lib/etcd
          capacity:
            storage: 4Gi
    workers:
      count: 3
      class: best-effort-medium
      storageClass: vwt-storage-policy
      volumes:
```

```
        - name: containerd
          mountPath: /var/lib/containerd
          capacity:
            storage: 16Gi
```

カスタム Antrea ネットワークを使用するクラスタ

次の YAML は、Antrea CNI のカスタム ネットワーク範囲を使用して Tanzu Kubernetes クラスタをプロビジョニングする方法を示します。

- カスタム ネットワーク設定が適用されるため、デフォルトの Antrea CNI が使用されている場合でも、`cni.name` パラメータが必要になります。
  - CNI 名：`antrea`
  - カスタム ポッドの CIDR：`193.0.2.0/16`
  - カスタムのサービス CIDR：`195.51.100.0/12`
  - カスタムのサービス ドメイン：`managedcluster.local`
- カスタム CIDR ブロックを スーパーバイザー クラスタ と重複させることはできません。詳細については、『Tanzu Kubernetes Grid サービス v1alpha1 API を使用する Tanzu Kubernetes クラスタの構成パラメータ』を参照してください。

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TanzuKubernetesCluster
metadata:
  name: tkg-cluster-3-antrea
  namespace: tkgs-cluster-ns
spec:
  distribution:
    version: v1.20
  topology:
    controlPlane:
      class: guaranteed-medium
      count: 3
      storageClass: vwt-storage-policy
    workers:
      class: guaranteed-medium
      count: 5
      storageClass: vwt-storage-policy
  settings:
    network:
      cni:
        name: antrea                        #Use Antrea CNI
      pods:
        cidrBlocks:
        - 193.0.2.0/16                      #Must not overlap with SVC
      services:
        cidrBlocks:
        - 195.51.100.0/12                   #Must not overlap with SVC
      serviceDomain: managedcluster.local
```

カスタム Calico ネットワークを使用するクラスタ

次の YAML は、カスタム Calico ネットワークを使用して Tanzu Kubernetes クラスタをプロビジョニングする方法を示しています。

- Calico はデフォルトの CNI ではないため、マニフェストで明示的に指定されます。サービス レベルでデフォルトの CNI を変更する方法については、Tanzu Kubernetes Grid サービス v1alpha1 API の構成の例を参照してください。
  - CNI 名：`calico`
  - カスタム ポッドの CIDR：`198.51.100.0/12`
  - カスタムのサービス CIDR：`192.0.2.0/16`
  - カスタムのサービス ドメイン：`managedcluster.local`
- ネットワークは、デフォルトではなく、カスタムの CIDR 範囲を使用します。これらの範囲が スーパーバイザー クラスタ と重複することはできません。Tanzu Kubernetes Grid サービス v1alpha1 API を使用する Tanzu Kubernetes クラスタの構成パラメータを参照してください。

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TanzuKubernetesCluster
metadata:
  name: tkgs-cluster-2
  namespace: tkgs-cluster-ns
spec:
  distribution:
    version: v1.20
  topology:
    controlPlane:
      count: 3
      class: guaranteed-large
      storageClass: vwt-storage-policy
    workers:
      count: 5
      class: guaranteed-xlarge
      storageClass: vwt-storage-policy
  settings:
    network:
      cni:
        name: calico                          #Use Calico CNI for this cluster
      services:
        cidrBlocks: ["198.51.100.0/12"]       #Must not overlap with SVC
      pods:
        cidrBlocks: ["192.0.2.0/16"]          #Must not overlap with SVC
      serviceDomain: managedcluster.local
```

ストレージ クラスとパーシステント ボリュームのデフォルト クラスを使用するクラスタ

次のサンプル YAML は、動的な PVC プロビジョニング用のストレージ クラスとデフォルトのストレージ クラスを使用してクラスタをプロビジョニングする方法を示しています。

- `spec.settings.storage.classes` の設定は、クラスタ内のコンテナのパーシステント ストレージに 2 つのストレージ クラスを指定します。
- `spec.settings.storage.defaultClass` が指定されています。アプリケーションによっては、デフォルト クラスが必要になることがあります。たとえば、多くのチャートで参照されている `defaultClass` として Helm または Kubeapps を使用する場合などです。

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TanzuKubernetesCluster
metadata:
  name: default-storage-spec
  namespace: tkgs-cluster-ns
spec:
  topology:
    controlPlane:
      count: 3
      class: best-effort-medium
      storageClass: vwt-storage-policy
    workers:
      count: 3
      class: best-effort-medium
      storageClass: vwt-storage-policy
  distribution:
    version: v1.20
  settings:
    network:
      cni:
        name: antrea
      services:
        cidrBlocks: ["198.51.100.0/12"]
      pods:
        cidrBlocks: ["192.0.2.0/16"]
      serviceDomain: "tanzukubernetescluster.local"
    storage:
      classes: ["gold", "silver"]              #Array of named PVC storage classes
      defaultClass: silver                     #Default PVC storage class
```

プロキシ サーバを使用するクラスタ

プロキシ サーバの構成をクラスタ マニフェストに適用することにより、個々の Tanzu Kubernetes クラスタでプロキシ サーバを使用できるようになります。

次の特性に注意してください。

- `spec.settings.network.proxy` セクションは、この Tanzu Kubernetes クラスタの HTTP プロキシ構成を指定します。
- いずれの `proxy` サーバ値の構文も `http://<user>:<pwd>@<ip>:<port>` です。

- `localhost` と `127.0.0.1`、および Tanzu Kubernetes クラスタのポッドとサービスの CIDR などの特定のエンドポイントは、自動的にプロキシされません。`noProxy` フィールドにこれらを含める必要はありません。
- `noProxy` フィールドには、プロキシにしない CIDR の配列を使用できます。スーパーバイザー クラスタ のワークロード ネットワークから必要な値を取得します。Tanzu Kubernetes Grid サービス v1alpha1 API を使用する Tanzu Kubernetes クラスタの構成パラメータ の図を参照してください。
- `TkgServiceConfiguration` でグローバル プロキシが構成されている場合、そのプロキシ情報は、クラスタの初期デプロイ後にクラスタ マニフェストに伝達されます。グローバル プロキシ構成は、クラスタの作成時にプロキシ構成フィールドがない場合にのみ、クラスタ マニフェストに追加されます。つまり、クラスタごとの構成が優先されるため、グローバル プロキシ構成は上書きされます。

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TanzuKubernetesCluster
metadata:
  name: tkgs-cluster-with-proxy
  namespace: tkgs-cluster-ns
spec:
  distribution:
    version: v1.20
  topology:
    controlPlane:
      count: 3
      class: guaranteed-medium
      storageClass: vwt-storage-policy
    workers:
      count: 5
      class: guaranteed-xlarge
      storageClass: vwt-storage-policy
  settings:
    storage:
      classes: ["gold", "silver"]
      defaultClass: silver
    network:
      cni:
        name: antrea
      pods:
        cidrBlocks:
        - 193.0.2.0/16
      services:
        cidrBlocks:
        - 195.51.100.0/12
      serviceDomain: managedcluster.local
      proxy:
        httpProxy: http://10.186.102.224:3128  #Proxy URL for HTTP connections
        httpsProxy: http://10.186.102.224:3128 #Proxy URL for HTTPS connections
        noProxy: [10.246.0.0/16,192.168.144.0/20,192.168.128.0/20] #SVC Pod, Egress, Ingress
CIDRs
```

TLS 用のカスタム証明書を使用するクラスタ

TkgServiceConfiguration で trust.additionalTrustedCAs を指定する場合と同様の方法（Tanzu Kubernetes Grid サービス v1alpha1 API の構成パラメータを参照）により、TanzuKubernetesCluster の仕様で trust.additionalTrustedCAs を spec.settings.network に含めることができます。例：

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TanzuKubernetesCluster
metadata:
  name: tkgs-cluster-with-custom-certs-tls
  namespace: tkgs-cluster-ns
spec:
  topology:
    controlPlane:
      count: 3
      class: guaranteed-medium
      storageClass: vwt-storage-profile
    workers:
      count: 3
      class: guaranteed-large
      storageClass: vwt-storage-profile
  distribution:
    version: 1.20.2
  settings:
    network:
      cni:
        name: antrea
      services:
        cidrBlocks: ["198.51.100.0/12"]
      pods:
        cidrBlocks: ["192.0.2.0/16"]
      serviceDomain: "managedcluster.local"
      trust:
        additionalTrustedCAs:
          - name: custom-selfsigned-cert-for-tkc
            data: |
              LS0aaaaaaaaaaaaaaabase64...
```

TkgServiceConfiguration の仕様からグローバル設定を継承するクラスタと継承しないクラスタ

注： 次のクラスタ構成例では、vCenter Server バージョン 7.0 Update 2a 以降と、スーパーバイザー クラスタ バージョン 1.18.10 以降が必要です。

TkgServiceConfiguration からグローバル設定を継承する Tanzu Kubernetes クラスタをプロビジョニングするには、グローバル設定を指定しないか null に設定してクラスタを構成します。

たとえば、proxy 設定を継承するクラスタを構成する場合は、次のいずれかの方法を使用します。

オプション 1：クラスタ仕様に proxy 設定を含めません。

```
...
settings:
  network:
    cni:
      name: antrea
    services:
      cidrBlocks: ["198.51.100.0/12"]
    pods:
      cidrBlocks: ["192.0.2.0/16"]
    serviceDomain: "tanzukubernetescluster.local"
```

オプション 2：仕様に proxy 設定を含め、その値を明示的に null に設定します。

```
settings:
  network:
    proxy: null
```

TkgServiceConfiguration からデフォルト値を継承しない Tanzu Kubernetes クラスタをプロビジョニングするには、空の値を除くすべての要素を含むクラスタ仕様を構成します。

たとえば、TkgServiceConfiguration にグローバル proxy が構成されているときに、グローバル proxy 設定を継承しないクラスタをプロビジョニングするには、クラスタ仕様に次の構文を含めます。

```
...
settings:
  network:
    proxy:
      httpProxy: ""
      httpsProxy: ""
      noProxy: null
```

ローカル コンテンツ ライブラリを使用するクラスタ

エアギャップ環境で Tanzu Kubernetes クラスタをプロビジョニングするには、ローカル コンテンツ ライブラリから同期した仮想マシン イメージを使用してクラスタを作成します。

ローカル コンテンツ ライブラリ イメージを使用してクラスタをプロビジョニングするには、そのイメージをクラスタ仕様で指定する必要があります。`distribution.version` 値には、フル イメージの名前を入力できます。また、イメージ ディレクトリから名前の形式を保持している場合は、Kubernetes のバージョンに短縮することができます。完全修飾バージョン番号を使用する場合は、`-----` を `+` に置き換えます。たとえば、OVA イメージに `photon-3-k8s-v1.20.2---vmware.1-tkg.1.1d4f79a` という名前が付けられている場合、次の形式を使用できます。

```
spec:
  distribution:
    version: v1.20
```

```
spec:
  distribution:
    version: v1.20.2
```

```
spec:
  distribution:
    version: v1.20.2+vmware.1-tkg.1
```

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TanzuKubernetesCluster
metadata:
    name: tgks-cluster-9
    namespace: tkgs-cluster-ns
spec:
   topology:
       controlPlane:
           count: 3
           class: best-effort-medium
           storageClass: vwt-storage-policy
       workers:
           count: 3
           class: best-effort-medium
           storageClass: vwt-storage-policy
   distribution:
        version: v1.20.2
   settings:
        network:
          cni:
              name: antrea
          services:
             cidrBlocks: ["198.51.100.0/12"]
          pods:
             cidrBlocks: ["192.0.2.0/16"]
```

Tanzu Kubernetes Grid サービス v1alpha1 API の構成パラメータ

コンテナ ネットワーク インターフェイス (CNI)、プロキシ サーバ、TLS 証明書などの主要な機能のグローバル設定を使用して、Tanzu Kubernetes Grid サービス をカスタマイズできます。グローバル機能とクラスタ単位の機能を実装する場合は、トレードオフと考慮事項に注意してください。

必要に応じて、グローバル パラメータを使用して Tanzu Kubernetes Grid サービス を構成できます。

注意： Tanzu Kubernetes Grid サービス の構成は、グローバルに行われます。つまり、`TkgServiceConfiguration` の仕様に対して行った変更は、そのサービスによってプロビジョニングされたすべての Tanzu Kubernetes クラスタに適用されます。ローリング アップデートが手動またはアップグレードによって開始された場合は、変更されたサービス仕様によってクラスタが更新されます。

TkgServiceConfiguration の仕様

`TkgServiceConfiguration` 仕様は、Tanzu Kubernetes Grid サービス インスタンスを構成するためのフィールドを提供します。

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TkgServiceConfiguration
metadata:
  name: tkg-service-configuration-example
spec:
  defaultCNI: <antrea or calico>
  proxy:
    httpProxy: http://<user>:<pwd>@<ip>:<port>
    httpsProxy: http://<user>:<pwd>@<ip>:<port>
    noProxy: [<array of CIDRs to not proxy>]
  trust:
    additionalTrustedCAs:
      - name: <first-cert-name>
        data: <base64-encoded string of a PEM encoded public cert 1>
      - name: <second-cert-name>
        data: <base64-encoded string of a PEM encoded public cert 2>
```

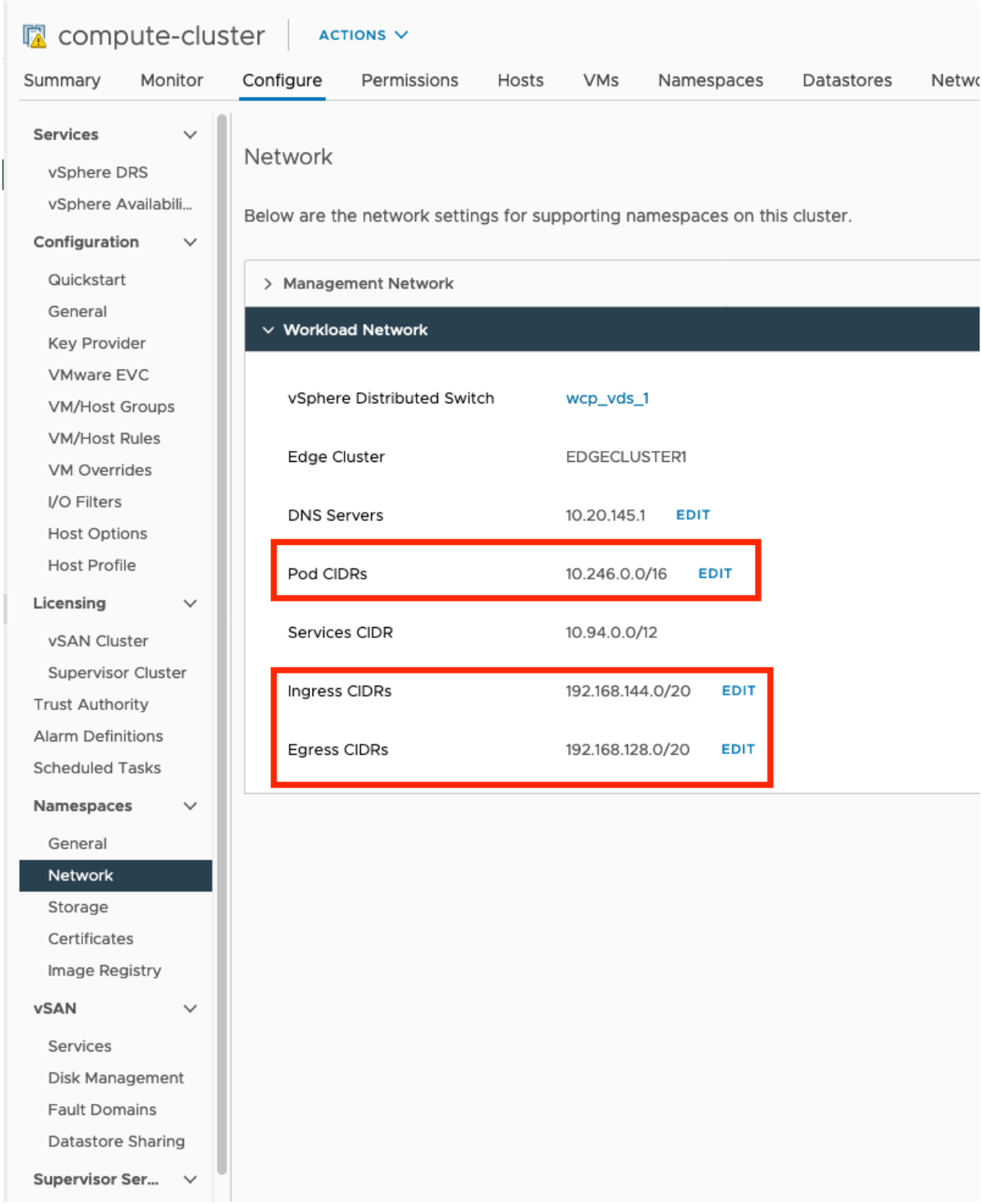
TkgServiceConfiguration 仕様のパラメータ

次の表に、`TkgServiceConfiguration` 仕様の各パラメータとその説明を示します。例については、Tanzu Kubernetes Grid サービス v1alpha1 API の構成の例を参照してください。

フィールド	値	説明
`defaultCNI`	`antrea or calico`	クラスタで使用するデフォルトの CNI。デフォルトは `antrea` です。サポートされているその他の CNI は `calico` です。
`proxy`	`proxy` パラメータのセクション マーカー。	`proxy` パラメータは `httpProxy`、`httpsProxy`、および `noProxy` です。すべてのパラメータが必要です。`proxy` パラメータが指定されていない場合は、Tanzu Kubernetes クラスタを作成できません。

フィールド	値	説明
httpProxy	http://<user>:<pwd>@<ip>:<port> 形式の URI	https プロトコルを許可しません。https が使用されている場合は、Tanzu Kubernetes クラスタを作成できません。
httpsProxy	http://<user>:<pwd>@<ip>:<port> 形式の URI	https プロトコルを許可しません。https が使用されている場合は、Tanzu Kubernetes クラスタを作成できません。
noProxy	プロキシにしない CIDR ブロックの配列（[10.246.0.0/16,192.168.144.0/20, 192.168.128.0/20] など）。 スーパーバイザー クラスタのワークロードネットワークから必要な値を取得します（Pod CIDRs、Ingress CIDRs、および Egress CIDRs）。 noProxy 配列フィールドに含める値については、次の図を参照してください。	スーパーバイザー クラスタ上のワークロードネットワークで使用されるサブネット（ポッド、Ingress、および Egress）をプロキシにすることはできません。 noProxy フィールドにスーパーバイザー クラスタのサービス CIDR を含める必要はありません。Tanzu Kubernetes クラスタは、これらのサービスと連携しません。 エンドポイント localhost と 127.0.0.1 は自動的にプロキシされません。これらを noProxy フィールドに追加する必要はありません。 Tanzu Kubernetes クラスタのポッド CIDR およびサービス CIDR は、自動的にプロキシされません。これらを noProxy フィールドに追加する必要はありません。
trust	trust パラメータのセクション マーカー。	データを受け入れない。
additionalTrustedCAs	name と data の証明書の配列をそれぞれ受け入れます。	データを受け入れない。
name	文字列	TLS 証明書の名前。
data	文字列	PEM でエンコードされたパブリック証明書の base64 エンコード文字列。

図に示すように、スーパーバイザー クラスタ の [ワークロード ネットワーク] から必要な noProxy 値を取得します。



グローバル構成オプションとクラスタ単位の構成オプションの使い分け

`TkgServiceConfiguration` は、Tanzu Kubernetes Grid サービス インスタンスによってプロビジョニングされているすべての Tanzu Kubernetes クラスタに影響するグローバルな仕様です。

TkgServiceConfiguration 仕様を編集する前に、グローバル構成よりも、クラスタ単位の代替方法のほうが使用事例に適合する可能性について確認してください。

表 13-5. グローバル構成オプションとクラスタ単位の構成オプション

設定	グローバル オプション	クラスタ単位のオプション
デフォルトの CNI	TkgServiceConfiguration 仕様を編集します。Tanzu Kubernetes Grid サービス v1alpha1 API の構成の例を参照してください。	クラスタ仕様で CNI を指定します。たとえば、デフォルトの CNI は Antrea です。Calico を使用するには、クラスタの YAML 内で指定します。Tanzu Kubernetes Grid サービス v1alpha1 API を使用した Tanzu Kubernetes クラスタのプロビジョニングの例を参照してください
プロキシ サーバ	TkgServiceConfiguration 仕様を編集します。Tanzu Kubernetes Grid サービス v1alpha1 API の構成の例を参照してください。	クラスタ仕様にプロキシ サーバ構成パラメータを含めます。Tanzu Kubernetes Grid サービス v1alpha1 API を使用した Tanzu Kubernetes クラスタのプロビジョニングの例を参照してください。
証明書の信頼	TkgServiceConfiguration 仕様を編集します。外部コンテナ レジストリと証明書ベース プロキシ構成の 2 つの使用事例があります。Tanzu Kubernetes Grid サービス v1alpha1 API の構成の例を参照してください	はい。カスタム証明書をクラスタ単位で含めることも、クラスタ仕様でグローバルに設定された trust 設定をオーバーライドすることもできます。Tanzu Kubernetes Grid サービス v1alpha1 API を使用した Tanzu Kubernetes クラスタのプロビジョニングの例を参照してください。

注： TkgServiceConfiguration でグローバル プロキシが構成されている場合、そのプロキシ情報は、クラスタの初期デプロイ後にクラスタ マニフェストに伝達されます。グローバル プロキシ構成は、クラスタの作成時にプロキシ構成フィールドがない場合にのみ、クラスタ マニフェストに追加されます。つまり、クラスタごとの構成が優先されるため、グローバル プロキシ構成は上書きされます。詳細については、『Tanzu Kubernetes Grid サービス v1alpha1 API の構成パラメータ』を参照してください。

TkgServiceConfiguration 仕様を編集する前に、設定をグローバル レベルで適用する場合の影響を確認してください。

フィールド	適用	追加/変更した場合の既存のクラスタへの影響	クラスタ作成時のクラスタ単位のオーバーライド	クラスタ更新時のクラスタ単位のオーバーライド
defaultCNI	グローバル	なし	可能。クラスタ作成時にグローバル設定をオーバーライドできます	不可能。既存のクラスタの CNI は変更できません。クラスタ作成時にグローバルに設定されたデフォルトの CNI を使用した場合は、変更できません
proxy	グローバル	なし	可能。クラスタ作成時にグローバル設定をオーバーライドできます	可能。U2+ を使用している場合は、クラスタ更新時にグローバル設定をオーバーライドできます
trust	グローバル	なし	可能。クラスタ作成時にグローバル設定をオーバーライドできます	可能。U2+ を使用している場合は、クラスタ更新時にグローバル設定をオーバーライドできます

既存のクラスタへのグローバル構成変更の伝達

TkgServiceConfiguration でグローバル レベルの設定を行っても、既存のクラスタにはその設定が自動的に伝達されません。たとえば、TkgServiceConfiguration で proxy や trust の設定を変更しても、すでにプロビジョニングされているクラスタには影響がありません。

既存のクラスタにグローバルな変更を伝達するには、Tanzu Kubernetes クラスタにパッチを適用して、TkgServiceConfiguration に対する変更をクラスタに継承させる必要があります。

例：

```
kubectl patch tkc <CLUSTER_NAME> -n <NAMESPACE> --type merge -p "{\"spec\":{\"settings\":
{\"network\":{\"proxy\": null}}}}"
```

```
kubectl patch tkc <CLUSTER_NAME> -n <NAMESPACE> --type merge -p "{\"spec\":{\"settings\":
{\"network\":{\"trust\": null}}}}"
```

Tanzu Kubernetes Grid サービス v1alpha1 API の構成の例

コンテナ ネットワーク インターフェイス、プロキシ サーバ、および TLS 証明書のグローバル設定を使用して、Tanzu Kubernetes Grid サービス をカスタマイズする例を参照してください。

Tanzu Kubernetes Grid サービス の構成について

Tanzu Kubernetes Grid サービス をカスタマイズするには、デフォルトの CNI を変更し、グローバル プロキシ サーバを追加し、信頼できる証明書を追加します。Tanzu Kubernetes Grid サービス v1alpha1 API の構成パラメータを参照してください。

注意： Tanzu Kubernetes Grid サービス 仕様を編集すると、新しいクラスタや、手動または自動でアップグレードされる既存のクラスタなど、このサービスによってプロビジョニングされた全クラスタがグローバルに変更されます。

前提条件：Kubectl 編集の構成

Tanzu Kubernetes クラスタを拡張するには、`kubectl edit tanzukubernetescluster/CLUSTER-NAME` コマンドを使用してクラスタ マニフェストを更新します。kubectl edit コマンドを実行すると、KUBE_EDITOR または EDITOR の環境変数によって定義されたテキスト エディタ内にクラスタ マニフェストが開きます。環境変数を設定する手順については、Kubectl のデフォルトのテキスト エディタの指定を参照してください。

仕様の変更を保存すると、`kubectl` から、編集内容が正常に記録されたことが報告されます。キャンセルするには、保存せずにエディタを閉じます。

デフォルト CNI の構成

Tanzu Kubernetes Grid サービス は、Tanzu Kubernetes クラスタにデフォルトのコンテナ ネットワーク インターフェイス (CNI) を提供します。デフォルトの構成では、CNI を指定せずにクラスタを作成できます。デフォルトの CNI を変更するには、サービスの仕様を編集します。

Tanzu Kubernetes Grid サービス は、Antrea と Calico という 2 つの CNI をサポートしています。デフォルトは Antrea です。詳細については、『Tanzu Kubernetes Grid サービス クラスタ ネットワーク』を参照してください。

デフォルトの CNI をオーバーライドするには、使用する CNI を明示的に指定します。また、CNI の TKG サービス コントローラを編集して、デフォルトの CNI を変更することもできます。

1 スーパーバイザー クラスタ で認証します。

```
kubectl vsphere login --server=SVC-IP-ADDRESS --vsphere-username USERNAME
```

2 コンテキストをターゲット vSphere 名前空間 に切り替えます。

```
kubectl config use-context tkgs-cluster-ns
```

3 デフォルトの CNI を一覧表示します。

```
kubectl get tkgserviceconfigurations
```

結果の例：

```
NAME                                DEFAULT CNI
tkg-service-configuration           antrea
```

4 Tanzu Kubernetes Grid サービス 仕様の編集内容をロードします。

```
kubectl edit tkgserviceconfigurations tkg-service-configuration
```

`tkg-service-configuration` 仕様が、KUBE_EDITOR または EDITOR 環境変数によって定義されたデフォルトのテキスト エディタで開かれます。

5 `spec.defaultCNI` の値を編集します。

たとえば、次の値からの変更を考えます。

```
spec:
  defaultCNI: antrea
```

この値を次のように変更します。

```
spec:
  defaultCNI: calico
```

6 変更を適用するには、テキスト エディタでこのファイルを保存します。キャンセルするには、保存せずにエディタを閉じます。

テキスト エディタで変更を保存すると、kubectl によって `tkg-service-configuration` サービス仕様が更新されます。

7 デフォルトの CNI が更新されていることを確認します。

```
kubectl get tkgserviceconfigurations
```

デフォルトの CNI が更新されます。デフォルトのネットワーク設定によってプロビジョニングされたクラスタでは、デフォルトの CNI が使用されます。

```
NAME                                DEFAULT CNI
tkg-service-configuration           calico
```

グローバル プロキシ サーバの構成

グローバル プロキシ サーバを有効にするには、`TkgServiceConfiguration` にプロキシ サーバのパラメータを追加します。必須フィールドの説明については、Tanzu Kubernetes Grid サービス v1alpha1 API の構成パラメータを参照してください。

1 スーパーバイザー クラスタ で認証します。

```
kubectl vsphere login --server=SVC-IP-ADDRESS --vsphere-username USERNAME
```

2 コンテキストをターゲット vSphere 名前空間 に切り替えます。

```
kubectl config use-context tkgs-cluster-ns
```

3 現在の構成を取得します。

```
kubectl get tkgserviceconfigurations
```

結果の例：

```
NAME                         DEFAULT CNI
tkg-service-configuration    antrea
```

4 Tanzu Kubernetes Grid サービス 仕様の編集内容をロードします。

```
kubectl edit tkgserviceconfigurations tkg-service-configuration
```

tkg-service-configuration 仕様が、KUBE_EDITOR または EDITOR 環境変数によって定義されたデフォルトのテキスト エディタで開かれます。

5 httpProxy、httpsProxy、noProxy などの必須フィールドをそれぞれ含む spec.proxy サブセクションを追加します。

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TkgServiceConfiguration
metadata:
  ...
  name: tkg-service-configuration-example
  resourceVersion: "44170525"
  selfLink: /apis/run.tanzu.vmware.com/v1alpha1/tkgserviceconfigurations/tkg-service-
configuration
  uid: 10347195-5f0f-490e-8ae1-a758a724c0bc
spec:
  defaultCNI: antrea
  proxy:
    httpProxy: http://<user>:<pwd>@<ip>:<port>
    httpsProxy: http://<user>:<pwd>@<ip>:<port>
    noProxy: [SVC-POD-CIDRs, SVC-EGRESS-CIDRs, SVC-INGRESS-CIDRs]
```

6 各プロキシ フィールドに適切な値をポピュレートします。各フィールドの説明については、Tanzu Kubernetes Grid サービス v1alpha1 API の構成パラメータを参照してください。

noProxy フィールドに必要な値は、スーパーバイザー クラスタ の [ワークロード ネットワーク] から取得されます。これらの値を取得する場所については、上記のトピックの図を参照してください。

例：

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TkgServiceConfiguration
metadata:
  ...
  name: tkg-service-configuration-example
  resourceVersion: "44170525"
  selfLink: /apis/run.tanzu.vmware.com/v1alpha1/tkgserviceconfigurations/tkg-service-
configuration
  uid: 10347195-5f0f-490e-8ae1-a758a724c0bc
spec:
  defaultCNI: antrea
  proxy:
    httpProxy: http://user:password@10.186.102.224:3128
    httpsProxy: http://user:password@10.186.102.224:3128
    noProxy: [10.246.0.0/16,192.168.144.0/20,192.168.128.0/20]
```

7 変更を適用するには、テキスト エディタでこのファイルを保存します。キャンセルするには、保存せずにエディタを閉じます。

テキスト エディタで変更を保存すると、`tkg-service-configuration` は kubectl サービス仕様で定義された構成を使用して Tanzu Kubernetes Grid サービス を更新します。

8 Tanzu Kubernetes Grid サービス がプロキシ設定を使用して更新されていることを確認します。

```
kubectl get tkgserviceconfigurations -o yaml
```

9 確認するには、Tanzu Kubernetes クラスタをプロビジョニングします。TKGS v1alpha2 API を使用して Tanzu Kubernetes クラスタをプロビジョニングするためのワークフローを参照してください。

次のコマンドを使用して、クラスタがプロキシを使用していることを確認します。

```
kubectl get tkc CLUSTER-NAME -n NAMESPACE -o yaml
```

証明書ベースのプロキシ構成

環境によっては、プロキシ サーバを使用してインターネット トラフィックをルーティングすることが困難な場合があります。たとえば、金融機関などの規制の厳しい業界の企業では、すべてのインターネット トラフィックが企業のプロキシを経由する必要があります。

Tanzu Kubernetes Grid サービス を構成すると、送信 HTTP/S トラフィックにプロキシ サーバを使用するように Tanzu Kubernetes クラスタをプロビジョニングできます。詳細については、『Tanzu Kubernetes Grid サービス v1alpha1 API の構成パラメータ』を参照してください。

この例に示すように、プロキシ サーバの信頼できる証明書を `TkgServiceConfiguration` 仕様に追加できます。

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TkgServiceConfiguration
metadata:
  name: tkg-service-configuration-example
spec:
  defaultCNI: antrea
  proxy:
    httpProxy: http://user:password@10.186.102.224:3128
    httpsProxy: http://user:password@10.186.102.224:3128
    noProxy: [10.246.0.0/16,192.168.144.0/20,192.168.128.0/20]
  trust:
    additionalTrustedCAs:
      - name: first-cert-name
        data: base64-encoded string of a PEM encoded public cert 1
      - name: second-cert-name
        data: base64-encoded string of a PEM encoded public cert 2
```

外部プライベート レジストリの構成

外部プライベート レジストリを Tanzu Kubernetes クラスタに接続するためのカスタム証明書を使用して、Tanzu Kubernetes Grid サービス を構成できます。詳細については、『Tanzu Kubernetes クラスタでの外部コンテナ レジストリの使用』を参照してください。

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TkgServiceConfiguration
metadata:
  name: tkg-service-configuration-example
spec:
  defaultCNI: antrea
  trust:
    additionalTrustedCAs:
      - name: harbor-vm-cert
        data: <<<base64-encoded string of a PEM encoded public cert>>>>
```

Tanzu Kubernetes Grid サービス v1alpha1 API を使用した Tanzu Kubernetes クラスタのスケーリング

ノード数を変更するか、ノードをホストする仮想マシン クラスを垂直方向に変更することで、Tanzu Kubernetes クラスタを水平方向にスケーリングできます。

サポートされているスケーリング操作

次の表に、Tanzu Kubernetes クラスタでサポートされているスケーリング操作を示します。

表 13-6. Tanzu Kubernetes クラスタでサポートされているスケーリング操作

ノード	水平方向のスケール アウト	水平方向のスケール イン	垂直方向のスケーリング
制御プレーン	はい	なし	はい
ワーカー	はい	はい	はい

次の考慮事項に留意してください。

- クラスタ ノードを垂直方向にスケーリングするときに、使用可能なリソースが不足していると、ノード上でワークロードを実行できなくなる可能性があります。このため、水平方向にスケーリングする方法が推奨されることがあります。
- 仮想マシン クラスは変更できません。Tanzu Kubernetes クラスタで使用される仮想マシン クラスを編集した後に、このクラスタをスケール アウトした場合、新しいクラスタ ノードは更新されたクラス定義を使用しますが、既存のクラスタ ノードは初期のクラス定義を引き続き使用するため、不一致が生じます。Tanzu Kubernetes クラスタの仮想マシンのクラスを参照してください。

スケーリングの前提条件：Kubectl 編集の構成

Tanzu Kubernetes クラスタを拡張するには、`kubectl edit tanzukubernetescluster/CLUSTER-NAME` コマンドを使用してクラスタ マニフェストを更新します。kubectl edit コマンドを実行すると、KUBE_EDITOR または EDITOR の環境変数によって定義されたテキスト エディタ内にクラスタ マニフェストが開きます。環境変数を設定する手順については、Kubectl のデフォルトのテキスト エディタの指定を参照してください。

マニフェストの変更を保存すると、`kubectl` から編集内容が正常に記録されたことが報告され、この変更内容を使用してクラスタが更新されます。

```
kubectl edit tanzukubernetescluster/tkgs-cluster-1
tanzukubernetescluster.run.tanzu.vmware.com/tkgs-cluster-1 edited
```

キャンセルするには、保存せずにエディタを閉じます。

```
kubectl edit tanzukubernetescluster/tkgs-cluster-1
Edit cancelled, no changes made.
```

制御プレーンのスケール アウト

制御プレーン ノードの数を 1 から 3 に増やすことで、Tanzu Kubernetes クラスタをスケール アウトできます。制御プレーン ノードの数は奇数にする必要があります。制御プレーンではスケーリングできません。

1 スーパーバイザー クラスタ で認証します。

```
kubectl vsphere login --server=SVC-IP-ADDRESS --vsphere-username USERNAME
```

2 Tanzu Kubernetes クラスタが実行されている vSphere 名前空間 にコンテキストを切り替えます。

```
kubectl config use-context tkgs-cluster-ns
```

3 この名前空間で実行されている Kubernetes クラスタを一覧表示します。

```
kubectl get tanzukubernetescluster -n tkgs-cluster-ns
```

4 ターゲット クラスタで実行されているノードの数を取得します。

```
kubectl get tanzukubernetescluster tkgs-cluster-1
```

たとえば、次のクラスタには、1 台の制御プレーン ノードと、3 台のワーカー ノードが含まれています。

```
NAME                CONTROL PLANE   WORKER   DISTRIBUTION                      AGE   PHASE
tkgs-cluster-1      1               3        v1.18.5+vmware.1-tkg.1.886c781    1d    running
```

5 `kubectl edit` コマンドを使用して、編集するクラスタ マニフェストをロードします。

```
kubectl edit tanzukubernetescluster/tkgs-cluster-1
```

クラスタ マニフェストは、KUBE_EDITOR または EDITOR 環境変数によって定義されたテキスト エディタで開かれます。

6 `spec.topology.controlPlane.count` パラメータを特定し、ノードの数を 1 から 3 に増やします。

```
...
controlPlane:
    count: 1
...
```

```
...
ControlPlane:
    count: 3
...
```

7 変更を適用するには、テキスト エディタでこのファイルを保存します。キャンセルするには、保存せずにエディタを閉じます。

ファイルを保存すると、kubectl によって変更がクラスタに適用されます。バックグラウンドでは、スーパーバイザー クラスタ の 仮想マシン サービス によって新しいワーカー ノードがプロビジョニングされます。

8 新しいノードが追加されていることを確認します。

```
kubectl get tanzukubernetescluster tkgs-cluster-1
```

スケール アウト制御プレーンのノード数が 3 になりました。

```

NAME                CONTROL PLANE   WORKER   DISTRIBUTION                      AGE   PHASE
tkgs-cluster-1      3               3        v1.18.5+vmware.1-tkg.1.886c781   1d    running
```

ワーカー ノードのスケール アウト

kubectl を使用してワーカー ノードの数を増やすことで、Tanzu Kubernetes クラスタをスケール アウトできます。

1 スーパーバイザー クラスタ で認証します。

```
kubectl vsphere login --server=SVC-IP-ADDRESS --vsphere-username USERNAME
```

2 Tanzu Kubernetes クラスタが実行されている vSphere 名前空間 にコンテキストを切り替えます。

```
kubectl config use-context tkgs-cluster-ns
```

3 この名前空間で実行されている Kubernetes クラスタを一覧表示します。

```
kubectl get tanzukubernetescluster -n tkgs-cluster-ns
```

4 ターゲット クラスタで実行されているノードの数を取得します。

```
kubectl get tanzukubernetescluster tkgs-cluster-1
```

たとえば、次のクラスタには、3 台の制御プレーン ノードと、3 台のワーカー ノードが含まれています。

```
NAME                CONTROL PLANE   WORKER   DISTRIBUTION                      AGE   PHASE
tkgs-cluster-1      3               3        v1.18.5+vmware.1-tkg.1.886c781   1d    running
```

5 `kubectl edit` コマンドを使用して、編集するクラスタ マニフェストをロードします。

```
kubectl edit tanzukubernetescluster/tkgs-cluster-1
```

クラスタ マニフェストは、KUBE_EDITOR または EDITOR 環境変数によって定義されたテキスト エディタで開かれます。

6 `spec.topology.workers.count` パラメータを特定し、ノードの数を増やします。

```
...
workers:
    count: 3
...
```

```
...
workers:
    count: 4
...
```

7 変更を適用するには、テキスト エディタでこのファイルを保存します。キャンセルするには、保存せずにエディタを閉じます。

ファイルを保存すると、kubectl によって変更がクラスタに適用されます。バックグラウンドでは、スーパーバイザー クラスタ の 仮想マシン サービス によって新しいワーカー ノードがプロビジョニングされます。

8 新しいワーカー ノードが追加されていることを確認します。

```
kubectl get tanzukubernetescluster tkgs-cluster-1
```

スケール アウト後、クラスタ内のワーカー ノード数は 4 になります。

```
NAME                CONTROL PLANE   WORKER   DISTRIBUTION                      AGE   PHASE
tkgs-cluster-1      3               4        v1.18.5+vmware.1-tkg.1.886c781   1d    running
```

ワーカー ノードのスケール イン

ワーカー ノードの数を減らすことで、Tanzu Kubernetes クラスタをスケール インできます。制御プレーンのスケーリングはサポートされていません。

1 スーパーバイザー クラスタ で認証します。

```
kubectl vsphere login --server=SVC-IP-ADDRESS --vsphere-username USERNAME
```

2 Tanzu Kubernetes クラスタが実行されている vSphere 名前空間 にコンテキストを切り替えます。

```
kubectl config use-context tkgs-cluster-ns
```

3 この名前空間で実行されている Kubernetes クラスタを一覧表示します。

```
kubectl get tanzukubernetescluster -n tkgs-cluster-ns
```

4 ターゲット クラスタで実行されているノードの数を取得します。

```
kubectl get tanzukubernetescluster tkgs-cluster-1
```

たとえば、次のクラスタには、3 台の制御プレーン ノードと、3 台のワーカー ノードが含まれています。

```
NAME                CONTROL PLANE   WORKER   DISTRIBUTION                      AGE   PHASE
tkgs-cluster-1      3               4        v1.18.5+vmware.1-tkg.1.886c781   1d    running
```

5 `kubectl edit` コマンドを使用して、編集するクラスタ マニフェストをロードします。

```
kubectl edit tanzukubernetescluster/tkgs-cluster-1
```

クラスタ マニフェストは、KUBE_EDITOR または EDITOR 環境変数によって定義されたテキスト エディタで開かれます。

6 `spec.topology.workers.count` パラメータを特定し、ノードの数を増やします。

```
...
workers:
    count: 4
...
```

```
...
workers:
    count: 2
...
```

7 変更を適用するには、テキスト エディタでこのファイルを保存します。キャンセルするには、保存せずにエディタを閉じます。

ファイルを保存すると、kubectl によって変更がクラスタに適用されます。バックグラウンドでは、スーパーバイザー クラスタ の 仮想マシン サービス によって新しいワーカー ノードがプロビジョニングされます。

8 ワーカー ノードが削除されたことを確認します。

```
kubectl get tanzukubernetescluster tkgs-cluster-1
```

スケール イン後、クラスタ内のワーカー ノード数は 2 になります。

```
NAME                CONTROL PLANE   WORKER   DISTRIBUTION                      AGE   PHASE
tkgs-cluster-1      3               2        v1.18.5+vmware.1-tkg.1.886c781   1d    running
```

クラスタの垂直方向のスケーリング

クラスタ ノードのホストに使用されている仮想マシンのクラスを変更することで、Tanzu Kubernetes クラスタを垂直方向にスケーリングできます。垂直方向のスケーリングは、制御プレーンノードとワーカー ノードの両方でサポートされています。

Tanzu Kubernetes Grid サービス は、サービスに組み込まれているローリング アップデート メカニズムを使用して、クラスタ ノードの垂直方向のスケーリングをサポートします。`VirtualMachineClass` の定義を変更すると、この新しいクラスを使用して新しいノードがロール アウトされて、古いノードがスピン ダウンされます。Tanzu Kubernetes クラスタの更新を参照してください。

1 スーパーバイザー クラスタ で認証します。

```
kubectl vsphere login --server=SVC-IP-ADDRESS --vsphere-username USERNAME
```

2 Tanzu Kubernetes クラスタが実行されている vSphere 名前空間 にコンテキストを切り替えます。

```
kubectl config use-context tkgs-cluster-ns
```

3 この名前空間で実行されている Kubernetes クラスタを一覧表示します。

```
kubectl get tanzukubernetescluster -n tkgs-cluster-ns
```

4 ターゲット Tanzu Kubernetes クラスタを記述し、仮想マシンのクラスを確認します。

```
kubectl describe tanzukubernetescluster tkgs-cluster-2
```

たとえば、次のクラスタでは best-effort-small 型の仮想マシン クラスが使用されています。

```
Spec:
  ...
  Topology:
    Control Plane:
      Class:          best-effort-small
      ...
    Workers:
      Class:          best-effort-small
      ...
```

5 使用可能な仮想マシン クラスを一覧表示して、記述します。

```
kubectl get virtualmachineclassbinding
```

```
kubectl describe virtualmachineclassbinding
```

注： 使用する仮想マシン クラスを、vSphere 名前空間 にバインドする必要があります。Tanzu Kubernetes クラスタの仮想マシンのクラスを参照してください。

6 ターゲット クラスタのマニフェスト開いて編集します。

```
kubectl edit tanzukubernetescluster/tkgs-cluster-2
```

クラスタ マニフェストは、KUBE_EDITOR または EDITOR 環境変数によって定義されたテキスト エディタで開かれます。

7 仮想マシン クラスを変更して、マニフェストを編集します。

たとえば、制御プレーン ノードとワーカー ノードに guaranteed-xlarge 仮想マシン クラスを使用するように、クラスタ マニフェストを編集します。

```
spec:
  topology:
    controlPlane:
      class: guaranteed-xlarge
      ...
    workers:
      class: guaranteed-xlarge
      ...
```

8 変更を適用するには、テキスト エディタでこのファイルを保存します。キャンセルするには、保存せずにエディタを閉じます。

ファイルを保存すると、kubectl によって変更がクラスタに適用されます。Tanzu Kubernetes Grid サービス はバックグラウンドで新しいノードをプロビジョニングし、古いノードを削除します。ローリング アップデート プロセスの説明については、Tanzu Kubernetes Grid サービス クラスタの更新についてを参照してください。

9 クラスタが更新されていることを確認します。

```
kubectl get tanzukubernetescluster
NAME             CONTROL PLANE   WORKER   DISTRIBUTION                      AGE   PHASE
tkgs-cluster-1   3               3        v1.18.5+vmware.1-tkg.1.c40d30d   21h   updating
```

Tanzu Kubernetes クラスタの削除

Tanzu Kubernetes Grid サービス によってプロビジョニングされた Tanzu Kubernetes クラスタを削除するには、kubectl を使用します。

kubectl を使用して Tanzu Kubernetes クラスタを削除すると、Kubernetes ガベージ コレクションによってすべての依存リソースが削除されます。

注： vSphere Client または vCenter Server CLI を使用して Tanzu Kubernetes クラスタの削除を試みないでください。

手順

1 スーパーバイザー クラスタ で認証します。

vCenter Single Sign-On ユーザーとして スーパーバイザー クラスタ に接続するを参照してください。

2 削除する Tanzu Kubernetes がプロビジョニングされている vSphere 名前空間 にコンテキストを変更します。

```
kubectl config use-context CLUSTER-NAMESPACE
```

例：

```
kubectl config use-context tkgs-ns-1
```

3 名前空間内にある Tanzu Kubernetes クラスタを一覧表示します。

```
kubectl get clusters
```

例：

```
kubectl get clusters
NAME             PHASE
tkgs-cluster-1   provisioned
```

4 次の構文を使用して Tanzu Kubernetes クラスタを削除します。

```
kubectl delete tanzukubernetescluster --namespace CLUSTER-NAMESPACE CLUSTER-NAME
```

例：

```
kubectl delete tanzukubernetescluster --namespace tkgs-ns-1 tkgs-cluster-1
```

予期される結果：

```
tanzukubernetescluster.run.tanzu.vmware.com "tkgs-cluster-1" deleted
```

5 クラスタが削除されたことを確認します。

```
kubectl get clusters
```

例：

```
kubectl get clusters
No resources found in tkgs-ns-1 namespace.
```

6 kubeconfig ファイルからクラスタのコンテキストを削除します。

```
kubectl config delete-context CONTEXT
```

例：

```
kubectl config get-contexts
CURRENT   NAME              CLUSTER         AUTHINFO
NAMESPACE
          192.0.2.1         192.0.2.1       wcp:192.0.2.1:administrator@vsphere.local
          tkgs-cluster-1    192.0.2.6       wcp:192.0.2.6:administrator@vsphere.local
*         tkgs-ns-1         192.0.2.7       wcp:192.0.2.7:administrator@vsphere.local
tkgs-ns-1
```

```
kubectl config delete-context tkgs-cluster-1
deleted context tkgs-cluster-1 from $HOME/.kube/config
```

```
kubectl config get-contexts
CURRENT   NAME              CLUSTER         AUTHINFO
NAMESPACE
          192.0.2.1         192.0.2.1       wcp:192.0.2.1:administrator@vsphere.local
*         tkgs-ns-1         192.0.2.7       wcp:192.0.2.7:administrator@vsphere.local
tkgs-ns-1
```

Kubectl のデフォルトのテキスト エディタの指定

Tanzu Kubernetes クラスタをプロビジョニング、操作、および維持できるようにするには、kubectl のデフォルトのテキスト エディタを指定します。

目的

Tanzu Kubernetes クラスタをプロビジョニングした後も、クラスタを維持する必要があります。一般的なメンテナンス タスクでは、Kubernetes バージョンのアップグレードとクラスタ ノードのスケーリングを行います。このようなタスクを実行するには、クラスタ マニフェストを更新します。

プロビジョニングされたクラスタのマニフェストを更新する最も便利な方法は、kubectl edit コマンドを使用することです。このコマンドを実行すると、選択したテキスト エディタで Kubernetes マニフェストが開きます。変更を保存すると、Kubernetes によって変更が自動的に適用されて、クラスタが更新されます。

`kubectl edit` コマンドを使用するには、KUBE_EDITOR 環境変数を作成し、変数の値として優先するテキスト エディタを指定します。また、行った変更をコミット（保存）したときに kubectl が認識できるように、監視フラグ (`-w`) を値に追加します。

Windows

Windows の場合は、`KUBE_EDITOR` という名前のシステム環境変数を作成して、値を優先テキスト エディタのパスに設定します。この値に監視フラグ (`-w`) を追加します。

たとえば、次の環境変数は、kubectl のデフォルトのテキスト エディタとして Visual Studio Code を設定し、変更を保存したときに Kubernetes が認識できるように監視フラグを追加します。

```
KUBE_EDITOR=code -w
```

Mac OS

Mac OS の場合は、KUBE_EDITOR という名前の環境変数を作成して、値を優先テキスト エディタのパスに設定します。行った変更をコミット（保存）したときに kubectl が認識できるように、監視フラグ (-w) を値に追加します。

たとえば、次のように .bash_profile に追加すると、kubectl のデフォルトのテキスト エディタとして Sublime が設定され、変更を保存したときに kubectl が認識できるように監視フラグが追加されます。

```
export KUBE_EDITOR="/Applications/Sublime.app/Contents/SharedSupport/bin/subl -w"
```

Linux

Linux (Ubuntu など) では、通常、デフォルトのコマンドライン EDITOR は Vim です。この場合、kubectl edit コマンドを使用するための追加のアクションは不要です。別のエディタを使用する場合は、KUBE_EDITOR という名前の環境変数を作成して、値を優先テキスト エディタのパスに設定します。

Tanzu Kubernetes クラスタの運用

Tanzu Kubernetes Grid サービス には、Tanzu Kubernetes クラスタを運用するために使用するカスタム リソースが含まれています。また、vSphere インフラストラクチャとの緊密な連携により、使い慣れた vSphere ツールを使用して、Tanzu Kubernetes クラスタの管理と維持に役立てることができます。

kubectl を使用した Tanzu Kubernetes クラスタのステータスの監視

kubectl を使用して、プロビジョニングされている Tanzu Kubernetes クラスタのステータスを監視できます。

手順

1 スーパーバイザー クラスタ で認証します。vCenter Single Sign-On ユーザーとして スーパーバイザー クラスタ に接続するを参照してください。

2 クラスタが実行されている vSphere 名前空間 に切り替えます。

```
kubectl config use-context SUPERVISOR-NAMESPACE
```

3 名前空間で実行されている Tanzu Kubernetes クラスタのリストを表示します。

```
kubectl get tanzukubernetesclusters
```

このコマンドは、クラスタのステータスを返します。ステータス フィールドの説明については、kubectl での Tanzu Kubernetes クラスタ ライフサイクル ステータスを参照してください。

4 クラスタの詳細を表示します。

```
kubectl describe tanzukubernetescluster <cluster-name>
```

このコマンドを実行すると、クラスタの詳細が返されます。コマンド出力の [ステータス] セクションに、クラスタに関する詳細情報が表示されます。

```
...
Status:
  Addons:
    Cni:
      Name:    calico
      Status:  applied
    Csi:
      Name:    pvcsi
      Status:  applied
    Psp:
      Name:    defaultpsp
      Status:  applied
  Cloudprovider:
    Name:  vmware-guest-cluster
  Cluster API Status:
    API Endpoints:
      Host:  10.161.90.22
      Port:  6443
    Phase:   provisioned
  Node Status:
    test-tanzu-cluster-control-plane-0:                 ready
    test-tanzu-cluster-workers-0-749458f97c-97ljv:      ready
  Phase:                                                running
  Vm Status:
    test-tanzu-cluster-control-plane-0:                 ready
    test-tanzu-cluster-workers-0-749458f97c-97ljv:      ready
Events:                                                 <none>
```

5 追加の `kubectl` コマンドを実行して、クラスタの詳細を表示します。Tanzu Kubernetes クラスタの操作コマンドの使用を参照してください。

Tanzu Kubernetes クラスタの準備の確認

Tanzu Kubernetes Grid サービス が Tanzu Kubernetes クラスタをプロビジョニングすると、いくつかのステータス状態が報告されるようになります。これらを使用することでマシンの健全性に関する重要な情報を直接把握できます。

TanzuKubernetesCluster の準備の確認

TanzuKubernetesCluster の準備状態を使用して、準備ができていないフェーズまたはコンポーネントがある場合はそれを判別できます。ControlPlaneReady の状態と理由を参照してください。

クラスタの準備を確認したら、capwcluster とマシンの状態を使用し、障害を詳細に調べることでさらに診断をすることができます。『Tanzu Kubernetes マシンの健全性の確認 』と『Tanzu Kubernetes クラスタの健全性の確認』を参照してください。

Tanzu Kubernetes クラスタの準備を確認するには、次の手順を実行します。

1 スーパーバイザー クラスタ にログインします。

2 ターゲット クラスタがプロビジョニングされている名前空間にコンテキストを切り替えます。例：

```
kubectl config use-context tkgs-cluster-ns
```

3 `kubectl get tkc -o yaml` コマンドを実行します。クラスタの準備状態が表示されます。例：

```
status:
  addons:
    authsvc:
      conditions:
      - lastTransitionTime: "2021-01-30T19:53:54Z"
        status: "True"
        type: AuthServiceProvisioned
      name: authsvc
      status: applied
      version: 0.1-66-g8b8f07f
    cloudprovider:
      conditions:
      - lastTransitionTime: "2021-01-30T19:53:53Z"
        status: "True"
        type: CPIProvisioned
      name: vmware-guest-cluster
      status: applied
      version: 0.1-77-g5875817
    cni:
      conditions:
      - lastTransitionTime: "2021-01-30T19:53:53Z"
        status: "True"
        type: CNIProvisioned
      name: calico
      status: applied
      version: 1.16.14+vmware.1-tkg.1.ada4837
    csi:
      conditions:
      - lastTransitionTime: "2021-01-30T19:53:54Z"
        status: "True"
        type: CSIProvisioned
      name: pvcsi
      status: applied
      version: v0.0.1.alpha+vmware.79-7ecdcb1
    dns:
      conditions:
      - lastTransitionTime: "2021-01-30T19:53:48Z"
        status: "True"
        type: CoreDNSProvisioned
      name: CoreDNS
      status: applied
      version: v1.6.2_vmware.10
    proxy:
      conditions:
      - lastTransitionTime: "2021-01-30T19:53:48Z"
        status: "True"
        type: KubeProxyProvisioned
```

```
      name: kube-proxy
      status: applied
      version: 1.16.14+vmware.1
    psp:
      conditions:
      - lastTransitionTime: "2021-01-30T19:53:47Z"
        status: "True"
        type: PSPProvisioned
      name: defaultpsp
      status: applied
      version: v1.16.14+vmware.1-tkg.1.ada4837
  clusterApiStatus:
    apiEndpoints:
    - host: 192.168.1.2
      port: 6443
    phase: Provisioned
  conditions:
  - lastTransitionTime: "2021-01-30T19:53:54Z"
    status: "True"
    type: AddonsReady
  - lastTransitionTime: "2021-01-30T19:51:11Z"
    status: "True"
    type: ControlPlaneReady
  - lastTransitionTime: "2021-01-30T19:51:04Z"
    message: 3/3 Control Plane Node(s) healthy. 1/1 Worker Node(s) healthy
    status: "True"
    type: NodesHealthy
  - lastTransitionTime: "2021-01-31T21:22:45Z"
    status: "True"
    type: ProviderServiceAccountsReady
  - lastTransitionTime: "2021-01-30T19:53:50Z"
    status: "True"
    type: RoleBindingSynced
  - lastTransitionTime: "2021-01-30T19:53:58Z"
    status: "True"
    type: ServiceDiscoveryReady
  - lastTransitionTime: "2021-01-30T19:53:59Z"
    status: "True"
    type: StorageClassSynced
  - lastTransitionTime: "2021-01-27T11:34:53Z"
    status: "True"
    type: TanzuKubernetesReleaseCompatible
  - lastTransitionTime: "2021-01-27T11:34:54Z"
    message: '[1.17.13+vmware.1-tkg.2.2c133ed]'
    severity: Info
    status: "True"
    type: UpdatesAvailable
```

ControlPlaneReady の状態と理由

次の表に、`ControlPlaneReady` の状態の一覧と説明を示します。

表 13-7. ControlPlaneReady の状態

状態のタイプ	説明
ControlPlaneReady	制御プレーン ノードがクラスタに対して準備完了し、機能しているかどうかを報告します。

次の表に、ControlPlaneReady の状態が false になる理由の一覧と説明を示します。

表 13-8. ControlPlaneReady が False になる理由

理由	重要度	説明
WaitingForClusterInfrastructure		クラスタが、ロード バランサなどのマシンの実行に必要な前提条件を待機していることを示します。この理由は、InfrastructureCluster が独自の準備状態を報告していない場合にのみ使用されます。
WaitingForControlPlaneInitialized		最初の制御プレーン ノードが初期化中であることを示します。
WaitingForControlPlane		KubeadmControlPlane の条件を反映します。この理由は、KubeadmControlPlane が独自の準備状態を報告していない場合に使用されます。
クラスタ インフラストラクチャの準備が完了するのを待機している	メッセージ	クラスタが、ネットワークやロード バランサなどのマシンの実行に必要な前提条件を待機していることを示します。

NodesHealthy の状態と理由

次の表に、NodesHealthy の状態の一覧と説明を示します。

表 13-9. NodesHealthy の状態

状態のタイプ	説明
NodesHealthy	TankuKubernetesCluster ノードのステータスを報告します。

次の表に、NodesHealthy の状態が true にならない理由の一覧と説明を示します。

表 13-10. NodesHealthy が False になる理由

理由	重要度	説明
WaitingForNodesHealthy		すべてのノードが健全ではないドキュメント。

アドオンの状態と理由

次の表に、クラスタ アドオン コンポーネントに関連する状態の一覧と説明を示します。

表 13-11. アドオンの状態

状態のタイプ	説明
AddonsReady	TanzuKubernetesCluster アドオン（CoreDNS、KubeProxy、CSP、CPI、CNI、AuthSvc）の状態のサマリ。
CNIProvisioned	TanzuKubernetesCluster コンテナ ネットワーク インターフェイス (CNI) アドオンのステータスを文書化します。
CSIProvisioned	TanzuKubernetesCluster コンテナ ストレージ インターフェイス (CSI) アドオンのステータスを文書化します。
CPIProvisioned	TanzuKubernetesCluster クラウド プロバイダ インターフェイス (CPI) アドオンのステータスを文書化します。
KubeProxyProvisioned	TanzuKubernetesCluster KubeProxy アドオンのステータスを文書化します。
CoreDNSProvisioned	TanzuKubernetesCluster CoreDNS アドオンのステータスを文書化します。
AuthServiceProvisioned	TanzuKubernetesCluster AuthService アドオンのステータスを文書化します。
PSPProvisioned	PodSecurityPolicy のステータスを文書化します。

次の表に、アドオンの状態が True にならない理由の一覧と説明を示します。

表 13-12. アドオンが False になる理由

理由	重要度	説明
AddonsReconciliationFailed		すべてのアドオンの調整に失敗した理由の概要。
CNIProvisioningFailed	警告	CNI アドオンが作成または更新に失敗したことを文書化します。
CSIProvisioningFailed	警告	CSI アドオンが作成または更新に失敗したことを文書化します。
CPIProvisioningFailed	警告	CPI アドオンが作成または更新に失敗したことを文書化します。
KubeProxyProvisioningFailed	警告	KubeProxy アドオンが作成または更新に失敗したことを文書化します。
CoreDNSProvisioningFailed	警告	CoreDNS アドオンが作成または更新に失敗したことを文書化します。
AuthServiceProvisioningFailed	警告	AuthService アドオンが作成または更新に失敗したことを文書化します。
AuthServiceUnManaged		AuthService がコントローラによって管理されていないことを文書化します。
PSPProvisioningFailed	警告	PodSecurityPolicy アドオンが作成または更新に失敗したことを文書化します。

その他の状態と理由

次の表に、StorageClass と RoleBinding の同期、ProviderServiceAccount リソースの調整、ServiceDiscovery、および TanzuKubernetesCluster の互換性の状態の一覧と説明を示します。

表 13-13. その他の状態

条件	説明
StorageClassSynced	スーパーバイザー クラスタからワークロード クラスタへの StorageClass 同期のステータスを文書化します。
RoleBindingSynced	スーパーバイザー クラスタからワークロード クラスタへの RoleBinding 同期のステータスを文書化します。
ProviderServiceAccountsReady	プロバイダ サービスのアカウントと、関連するロール、ロールバインド、およびシークレットの作成のステータスを文書化します。
ServiceDiscoveryReady	サービス検出のステータスを文書化します。
TanzuKubernetesReleaseCompatible	TanzuKubernetesCluster が TanzuKubernetesRelease と互換性があるかどうかを示します。

次の表に、その他の状態が True にならない理由の一覧と説明を示します。

表 13-14. その他の理由

理由	重要度	説明
StorageClassSyncFailed		StorageClass の同期に失敗したことを報告します。
RoleBindingSyncFailed		ロールバインドの同期に失敗したことを報告します。
ProviderServiceAccountsReconciliationFailed		プロバイダ サービス アカウントに関連するリソースの調整に失敗したことを報告します。
SupervisorHeadlessServiceSetupFailed		スーパーバイザー クラスタ API サーバのヘッドレス サービス設定に失敗したことを文書化します。

Tanzu Kubernetes クラスタの完全なリソース階層の表示

kubectl を使用して、Tanzu Kubernetes クラスタの完全なリソース階層を表示できます。クラスタ リソースの完全なリストを表示すると、問題の原因となっている可能性のあるリソースを特定するのに役立ちます。

前提条件

スーパーバイザー クラスタ に接続します。vCenter Single Sign-On ユーザーとして スーパーバイザー クラスタに接続するを参照してください。

手順

1 ターゲット クラスタのコンテキストを使用するようにコンテキストを切り替えます。

```
kubectl config use-context CLUSTER-NAME
```

2 次のコマンドを実行して、クラスタ API クラスタ リソースを表示します。

```
kubectl describe clusters.cluster.x-k8s.io CLUSTER-NAME
```

このコマンドは、名前空間、API のバージョン、リソースのバージョンなど、指定したクラスタのリソース階層を返します。

Tanzu Kubernetes クラスタのライフサイクル ステータスの表示

kubectl を使用して、vSphere インベントリ内の Tanzu Kubernetes クラスタのライフサイクル ステータスを表示できます。

vSphere での Tanzu Kubernetes クラスタのライフサイクル ステータス

以下の表に、vSphere インベントリに表示される Tanzu Kubernetes クラスタのステータス情報を示します。この情報を表示するには、vSphere Client を使用した Tanzu Kubernetes クラスタのステータスの監視を参照してください。

表 13-15. vSphere インベントリ内の Tanzu Kubernetes クラスタのステータス

フィールド	説明	例
名前	クラスタのユーザー定義名。	`tkg-cluster-01`
作成時間	クラスタの作成日時。	`Mar 17, 2020, 11:42:46 PM`
段階	クラスタのライフサイクル ステータス。表 13-17. クラスタのライフサイクル フェーズのステータスを参照してください。	`creating`
ワーカー数	クラスタ内のワーカー ノードの台数。	1、2、または 5
配布バージョン	クラスタが実行している Kubernetes ソフトウェアのバージョン。	`v1.16.6+vmware.1-tkg.1.7144628`
制御プレーンのアドレス	クラスタ制御プレーンのロード バランサの IP アドレス。	`192.168.123.2`

kubectl での Tanzu Kubernetes クラスタ ライフサイクル ステータス

この表には、kubectl に表示される Tanzu Kubernetes クラスタのステータス情報が示されています。この情報を表示するには、kubectl を使用した Tanzu Kubernetes クラスタのステータスの監視 を参照してください。

表 13-16. kubectl での Tanzu Kubernetes クラスタのステータス

フィールド	説明	例
NAME	クラスタの名前。	`tkg-cluster-01`
CONTROL PLANE	クラスタ内の制御プレーン ノードの台数。	`3`
WORKER	クラスタ内のワーカー ノードの台数。	`5`
DISTRIBUTION	クラスタが実行されている Kubernetes バージョン。	`v1.16.6+vmware.1-tkg.1.5b5608b`

表 13-16. kubectl での Tanzu Kubernetes クラスタのステータス （続き）

フィールド	説明	例
AGE	クラスタが実行されている日数。	`13d`
PHASE	クラスタのライフサイクル ステータス。表 13-17. クラスタのライフサイクル フェーズのステータスを参照してください。	`running`

クラスタのライフサイクル フェーズのステータス

この表には、クラスタ ライフサイクルの各フェーズのステータスが示されています。『表 13-17. クラスタのライフサイクル フェーズのステータス』を参照してください

表 13-17. クラスタのライフサイクル フェーズのステータス

段階	説明
`creating`	クラスタのプロビジョニングを開始可能か、制御プレーンが作成中か、制御プレーンが作成中で未初期化のいずれかの段階です。
`deleting`	クラスタの削除中です。
`failed`	クラスタ制御プレーンの作成に失敗し、ユーザーの操作が必要な場合があります。
`running`	インフラストラクチャが作成および構成されていて、制御プレーンが完全に初期化されています。
`updating`	クラスタの更新中です。

Tanzu Kubernetes クラスタの操作コマンドの使用

カスタム kubectl コマンドを使用して Tanzu Kubernetes クラスタを管理できます。これらのコマンドは、Tanzu Kubernetes Grid サービス によって作成されるカスタム リソースによって提供されます。

Tanzu Kubernetes クラスタを管理するためのカスタム コマンド

次の表に、Tanzu Kubernetes クラスタを管理するための kubectl コマンドの一覧と説明を示します。

表 13-18. Tanzu Kubernetes クラスタを管理するためのカスタム コマンド

コマンド	説明
`kubectl get tanzukubernetescluster`	現在の名前空間内にあるクラスタのリストを表示します。
`kubectl get tkc`	前のコマンドの短縮形バージョンです。
`kubectl describe tanzukubernetescluster CLUSTER-NAME`	指定されたクラスタを説明し、示されている状態、ステータス、およびイベントを表示します。プロビジョニングが完了すると、このコマンドは、Kubernetes API エンドポイントに対応するロード バランサ用に作成された仮想 IP アドレスを示します。

表 13-18. Tanzu Kubernetes クラスタを管理するためのカスタム コマンド （続き）

コマンド	説明
`kubectl get cluster-api`	現在の名前空間でクラスタをサポートしているクラスタ API リソースのリストを表示します。これには、クラスタ API プロジェクトのリソース、および Tanzu Kubernetes Grid サービスによって使用されるクラスタ API プロバイダのリソースも含まれます。
`kubectl get tanzukubernetesreleases`	利用可能な Tanzu Kubernetes リリースのリストを表示します。
`kubectl get tkr`	前のコマンドの短縮形バージョンです。
`kubectl get tkr v1.17.8---vmware.1-tkg.1.5417466 -o yaml`	指定した Tanzu Kubernetes リリースの詳細を提供します。
`kubectl get virtualmachine`	現在の名前空間内のクラスタ ノードをサポートしている仮想マシン リソースのリストを表示します。
`kubectl get vm`	前のコマンドの短縮形バージョンです。
`kubectl describe virtualmachine VIRTUAL-MACHINE-NAME`	指定した仮想マシンを説明し、状態、現在のステータス、およびイベントを表示します。
`kubectl describe virtualmachinesetresourcepolicy`	現在の名前空間内のクラスタをサポートしている仮想マシン セット リソース ポリシー リソースのリストを表示します。このリソースは、クラスタに使用される vSphere オブジェクトのリソース プールとフォルダを表します。
`kubectl get virtualmachineservice`	現在の名前空間内のクラスタ ノードをサポートしている仮想マシン サービス リソースのリストを表示します。これらのリソースは、サービスに似ていますが、ポッドではなく仮想マシン用です。仮想マシン サービスは、クラスタの制御プレーン ノードのロード バランサを提供するために使用され、クラスタ内で LoadBalancer タイプの Kubernetes Service をサポートするために準仮想化クラウド プロバイダにも使用されます。`kubectl loadbalancer` コマンドも参照してください。
`kubectl get vmservice`	前のコマンドの短縮形バージョンです。
`kubectl describe virtualmachineservice VIRTUAL-MACHINE-SERVICE-NAME`	指定した仮想マシン サービスを説明し、クラスタの示された状態、現在のステータス、およびイベントを表示します。
`kubectl get virtualmachineimage`	利用可能な Tanzu Kubernetes リリースのリストを表示します。
`kubectl get vmimage`	前のコマンドのショートカット バージョンです。
`kubectl describe vmimage VM_IMAGE_NAME`	名前付き仮想マシン イメージの詳細情報を表示します。
`kubectl get loadbalancer`	現在の名前空間内のロード バランサ リソース（クラスタのために使用されているものも含む）のリストを表示します。仮想マシン サービスのためにロード バランサが作成されます。

表 13-18. Tanzu Kubernetes クラスタを管理するためのカスタム コマンド （続き）

コマンド	説明
`kubectl get virtualnetwork`	現在の名前空間内の仮想ネットワーク リソース（クラスタに使用されているものも含む）のリストを表示します。仮想ネットワークは、クラスタがプロビジョニングされる名前空間ごと、および各クラスタに対して作成されます。
`kubectl get persistentvolumeclaim`	現在の名前空間内のパーシステント ボリューム要求リソース（クラスタに使用されているものも含む）のリストを表示します。10 章 vSphere with Tanzu でのパーシステント ストレージの使用を参照してください。
`kubectl get cnsnodevmattachment`	現在の名前空間内の CNS ノード仮想マシン接続リソースのリストを表示します。これらのリソースは、クラスタのノードとして機能する仮想マシンへの、CNS によって管理されるパーシステント ボリュームの接続を表します。10 章 vSphere with Tanzu でのパーシステント ストレージの使用を参照してください。
`kubectl get configmap`	現在の名前空間内の構成マップ（クラスタ ノードの作成に使用されるものも含む）のリストを表示します。構成マップはユーザーが変更できるものではなく、すべての変更が上書きされます。
`kubectl get secret`	現在の名前空間内のシークレット（クラスタ ノードの作成および管理に使用されるシークレットも含む）のリストを表示します。Tanzu Kubernetes クラスタのシークレットの取得を参照してください。

Tanzu Kubernetes クラスタのネットワーク コマンドの使用

Tanzu Kubernetes Grid サービス は、ノード、ポッド、サービスのためのデフォルトのネットワークを使用して、Tanzu Kubernetes クラスタをプロビジョニングします。カスタム kubectl コマンドを使用してクラスタ ネットワークを確認できます。

Tanzu Kubernetes クラスタ ネットワークを確認するためのカスタム コマンド

クラスタ ネットワークの検証には、次のコマンドを参照してください。

表 13-19. クラスタ ネットワークを確認するためのカスタム kubectl コマンド

コマンド	説明

コンテキストを vSphere 名前空間に切り替えます。例：
`kubectl config use-context tkgs-ns`
次のコマンドを実行します。
`kubectl get tkgserviceconfigurations`
結果の例：
`NAME DEFAULT CNI`
`tkg-service-configuration antrea`
	デフォルトの CNI（変更されていなければ `antrea`）を返します。
	クラスタの YAML で明示的にオーバーライドされていないかぎり、クラスタの作成にはデフォルトの CNI が使用されます。
	デフォルトの CNI を変更するには、Tanzu Kubernetes Grid サービス v1alpha1 API の構成の例を参照してください。

コンテキストを vSphere 名前空間に切り替えます。例：
`kubectl config use-context tkgs-ns`
次のコマンドを実行します。
`kubectl get virtualnetwork`
結果の例：
`NAME SNAT`
`READY AGE`
`tkgs-cluster-12-vnet 10.191.152.133`
`True 4h3m`
	クラスタ ノードの仮想ネットワークを返します。
	ネットワーク アドレス変換 (SNAT) IP アドレスが割り当てられていることを確認するために使用します。

表 13-19. クラスタ ネットワークを確認するためのカスタム kubectl コマンド （続き）

コマンド	説明

コンテキストを vSphere 名前空間に切り替えます。例：
`kubectl config use-context tkgs-ns`
次のコマンドを実行します。
`kubectl get virtualmachines -o wide`
結果の例：
`NAME POWERSTATE CLASS IMAGE PRIMARY-IP AGE tkgs-cluster-12-control-plane-... poweredOn guaranteed-medium ob-...- v1.21.6---vmware.1-tkg.1.b3d708a 10.244.0.66 4h6m tkgs-cluster-12-worker-... poweredOn guaranteed-medium ob-...- v1.21.6---vmware.1-tkg.1.b3d708a 10.244.0.68 4h3m tkgs-cluster-12-worker-... poweredOn guaranteed-medium ob-...- v1.21.6---vmware.1-tkg.1.b3d708a 10.244.0.67 4h3m`
	クラスタ ノードの仮想ネットワーク インターフェイスを返します。
	各クラスタ ノードの仮想マシンに IP アドレスが割り当てられていることを確認するために使用します。

コンテキストを vSphere 名前空間に切り替えます。例：
`kubectl config use-context tkgs-ns`
次のコマンドを実行します。
`kubectl get virtualmachineservices`
結果の例：
`NAME TYPE AGE tkgs-cluster-12-control-plane-service LoadBalancer 3h53m`
	各クラスタ ノードの仮想マシン サービスを返します。
	状態が更新されており、ロード バランサの仮想 IP (VIP) アドレスが含まれていることを確認するために使用します。

表 13-19. クラスタ ネットワークを確認するためのカスタム kubectl コマンド （続き）

コマンド	説明

コンテキストを TKGS クラスタ名前空間に切り替えます。例：
`kubectl config use-context tkgs-cluster-10`
次のコマンドを実行します。
`kubectl get services -n NAMESPACE`
次の内容を確認します。
`curl -k https://EXTERNAL-IP:PORT/healthz`
	クラスタ API のアクセス用に作成された Kubernetes サービス ロード バランサを返します。外部 IP アドレスが割り当てられていることを確認するために使用します。
	ロード バランサ サービスの外部 IP アドレスおよびポートを使用した API へのアクセスを確認するには、`curl` を使用します。

コンテキストを vSphere 名前空間に切り替えます。例：
`kubectl config use-context tkgs-ns`
次のコマンドを実行します。
`kubectl get endpoints`
結果の例：
`NAME ENDPOINTS AGE tkgs-cluster-12-control-plane-service 10.244.0.66:6443 3h44m`
	クラスタの制御プレーン ノード（エンドポイント）を返します。各エンドポイントが作成され、エンドポイント プールに含まれていることを確認するために使用します。

Tanzu Kubernetes クラスタのシークレットの取得

Tanzu Kubernetes クラスタはシークレットを使用して、動作中の Tanzu Kubernetes クラスタのトークン、キー、およびパスワードを格納します。

Tanzu Kubernetes クラスタのシークレットのリスト

Kubernetes シークレットは、パスワード、トークン、SSH キーなどの少量の秘密データを格納するオブジェクトです。Tanzu Kubernetes クラスタの管理者は、クラスタの運用中に複数のシークレットを使用することがあります。この表には、クラスタ管理者が使用する可能性のある主なシークレットとその説明が示されています。

注： このリストは、完全なものではありません。このリストには、手動でのローテーションが必要になるシークレットや、トラブルシューティングのためにクラスタ ノードにアクセスする際に使用する必要があるシークレットのみが含まれています。

シークレット	説明
TANZU-KUBERNETES-CLUSTER-NAME-ccm-token-RANDOM	vSphere 名前空間 に接続するために準仮想化クラウド プロバイダのクラウド コントローラ マネージャによって使用されるサービス アカウント トークン。この認証情報のローテーションをトリガするには、シークレットを削除します。
TANZU-KUBERNETES-CLUSTER-NAME-pvcsi-token-RANDOM	vSphere 名前空間 に接続するために準仮想化 CSI プラグインによって使用されるサービス アカウント トークン。この認証情報のローテーションをトリガするには、シークレットを削除します。vSphere with Tanzu と vSphere ストレージの統合方法を参照してください。
TANZU-KUBERNETES-CLUSTER-NAME-kubeconfig	kubernetes-admin ユーザーとしてクラスタの制御プレーンに接続する際に使用可能な kubeconfig ファイル。vCenter Single Sign-On 認証が使用できない場合に、このシークレットを使用してクラスタにアクセスして、トラブルシューティングを行うことができます。管理者としての Tanzu Kubernetes クラスタ制御プレーンへの接続を参照してください。
TANZU-KUBERNETES-CLUSTER-NAME-ssh	vmware-system-user として任意のクラスタ ノードに接続する際に使用可能な SSH プライベート キー。このシークレットを使用して、任意のクラスタ ノードに SSH 接続し、トラブルシューティングを行うことができます。プライベート キーを使用した、システム ユーザーとしての Tanzu Kubernetes クラスタ ノードへの SSH 接続を参照してください。
TANZU-KUBERNETES-CLUSTER-NAME-ssh-password	vmware-system-user として任意のクラスタ ノードに接続する際に使用可能なパスワード。プライベート キーを使用した、システム ユーザーとしての Tanzu Kubernetes クラスタ ノードへの SSH 接続を参照してください。
TANZU-KUBERNETES-CLUSTER-NAME-ca	kubectl で Kubernetes API サーバに安全に接続するために使用する Tanzu Kubernetes クラスタ制御プレーンのルート CA 証明書。

Tanzu Kubernetes マシンの健全性の確認

Tanzu Kubernetes Grid サービス が Tanzu Kubernetes クラスタをプロビジョニングすると、いくつかのステータス状態が報告されるようになります。これらを使用することでマシンの健全性に関する重要な情報を直接把握できます。

マシンの健全性状態について

Tanzu Kubernetes Grid サービス によってプロビジョニングされた Tanzu Kubernetes クラスタは、いくつかの可動部分で構成されます。これらのすべての可動部分は独立した関連性のあるコントローラによって操作され、一連の Kubernetes ノードを構築および維持するために連携します。TanzuKubernetesCluster オブジェクトには、マシンの健全性について詳細な情報を提供するステータス状態があります。

マシンの健全性の確認

Tanzu Kubernetes マシンの健全性を確認するには、次の手順を実行します。

1 kubectl describe machine コマンドを実行します。

ステータスが準備完了になっている場合、マシンは健全です。ただし、マシンの状態が false になっている場合は（InfrastructureReady など）、マシンの準備ができていません。

2　マシンの準備ができていない場合は、次のコマンドを実行して、インフラストラクチャの問題を特定します。

```
kubectl describe wcpmachine
```

マシンの健全性状態のリスト

次の表に、Tanzu Kubernetes クラスタで使用可能なマシンの健全性状態の一覧と定義を示します。

条件	説明
ResourcePolicyReady	リソース ポリシーの作成が成功したことを報告します。
ResourcePolicyCreationFailed	リソース ポリシーの作成中にエラーが発生した場合に報告されます。
ClusterNetworkReady	クラスタ ネットワークのプロビジョニングが成功したことを報告します。
ClusterNetworkProvisionStarted	クラスタ ネットワークの準備が完了するまで待機している場合に報告されます。
ClusterNetworkProvisionFailed	ネットワークのプロビジョニング中にエラーが発生した場合に報告されます。
LoadBalancerReady	固定制御プレーン エンドポイントの調整に成功したことを報告します。
LoadBalancerCreationFailed	ロードバランサ関連のリソースの作成が失敗した場合に報告されます。
WaitingForLoadBalancerIP	ロード バランサの IP アドレスが存在するようになるまで待機している場合に報告されます。
VMProvisioned	仮想マシンが作成され、パワーオンされて、IP アドレスが割り当てられていることを報告します。
WaitingForBootstrapData	vSphere マシンが、プロビジョニング プロセスを開始する前に、ブートストラップ スクリプトの準備ができるまで待機している場合に報告されます。
VMCreationFailed	仮想マシンの CRD または対応するブートストラップ ConfigMap の作成が失敗したことを報告します。
VMProvisionStarted	仮想マシンが現在、作成プロセスにある場合に報告されます。
PoweringOn	仮想マシンが現在パワーオン シーケンスを実行している場合に報告されます。
WaitingForNetworkAddress	マシンのネットワーク設定がアクティブになるまで待機している場合に報告されます。
WaitingForBIOSUUID	マシンに BIOS UUID が設定されるまで待機している場合に報告されます

状態フィールド

各状態に複数のフィールドを含めることができます。

Type	状態のタイプの説明です。たとえば、ResourcePolicyReady など。Ready 状態の場合は、他のすべての状態のサマリになります。
Status	タイプのステータスを示します。 状態には、True、False、Unknown などがあります。

Severity	Reason の分類です。 Info は、調整の実行中を意味します。 Warning は、問題が発生している可能性があるため、再試行されることを意味します。 Error は、エラーの発生により、解決のために手動による対処が必要なことを意味します。
Reason	ステータスが False である理由を示します。準備ができるまで待機していることを示す場合や、障害の理由を示す場合があります。通常は、ステータスが False の場合にスローされます。
Message	Reason について説明する、人間が解読可能な情報です。

Tanzu Kubernetes クラスタの健全性の確認

Tanzu Kubernetes Grid サービス が Tanzu Kubernetes クラスタをプロビジョニングすると、いくつかのステータス状態が報告されるようになります。これらを使用することでクラスタの健全性に関する重要な情報を直接把握できます。

クラスタの健全性状態について

Tanzu Kubernetes Grid サービス によってプロビジョニングされた Tanzu Kubernetes クラスタは、いくつかの可動部分で構成されます。これらのすべての可動部分は独立した関連性のあるコントローラによって操作され、一連の Kubernetes ノードを構築および維持するために連携します。TanzuKubernetesCluster オブジェクトには、クラスタとマシンの健全性について詳細な情報を提供するステータス状態があります。

クラスタの健全性の確認

Tanzu Kubernetes クラスタの健全性を確認するには、次の手順を実行します。

1 kubectl describe cluster コマンドを実行します。

ステータスが準備完了になっている場合、クラスタ インフラストラクチャとクラスタ制御プレーンは両方とも準備ができています。例：

```
Status:
  Conditions:
    Last Transition Time:     2020-11-24T21:37:32Z
    Status:                   True
    Type:                     Ready
    Last Transition Time:     2020-11-24T21:37:32Z
    Status:                   True
    Type:                     ControlPlaneReady
    Last Transition Time:     2020-11-24T21:31:34Z
    Status:                   True
    Type:                     InfrastructureReady
```

ただし、クラスタの状態が false の場合は、クラスタの準備ができておらず、メッセージ フィールドに問題の内容が示されます。たとえば、次の例では、ステータスは False であり、インフラストラクチャの準備はできていません。

```
Status:
  Conditions:
    Last Transition Time:     2020-11-24T21:37:32Z
    Status:                   False
    Type:                     Ready
    Last Transition Time:     2020-11-24T21:37:32Z
    Status:                   True
    Type:                     ControlPlaneReady
    Last Transition Time:     2020-11-24T21:31:34Z
    Status:                   False
    Type:                     InfrastructureReady
```

2 クラスタの準備ができていない場合は、次のコマンドを実行して、クラスタ インフラストラクチャの問題を特定します。

```
kubectl describe wcpcluster
```

クラスタの健全性状態のリスト

次の表に、Tanzu Kubernetes クラスタで使用可能な健全性状態の一覧と定義を示します。

条件	説明
Ready	クラスタ API オブジェクトの動作状態の概要を示します。
Deleting	基盤となるオブジェクトが現在削除されているため、ステータスは True ではありません。
DeletionFailed	基盤となるオブジェクトの削除中に問題が発生したため、ステータスは True ではありません。調整機能によって削除が再試行されるため、これは警告になります。
Deleted	基盤となるオブジェクトが削除されたため、ステータスは True ではありません。
InfrastructureReady	このクラスタに対して定義されているインフラストラクチャ オブジェクトの現在のステータスの概要を報告します。
WaitingForInfrastructure	基盤となるインフラストラクチャが利用可能になるまでクラスタが待機している場合に報告されます。注 : この状態は、インフラストラクチャが準備完了状態であることを報告していない場合に、フォールバックとして使用されます。
ControlPlaneReady	クラスタ制御プレーンの準備ができている場合に報告されます。
WaitingForControlPlane	制御プレーンが利用可能になるまでクラスタが待機している場合に報告されます。注 : この状態は、制御プレーンが準備完了状態であることを報告していない場合に、フォールバックとして使用されます。

状態フィールド

各状態に複数のフィールドを含めることができます。

Type	状態のタイプの説明です。たとえば、ControlPlaneReady など。Ready 状態の場合は、他のすべての状態のサマリになります。
Status	タイプのステータスを示します。 状態には、True、False、Unknown などがあります。
Severity	Reason の分類です。 Info は、調整の実行中を意味します。 Warning は、問題が発生している可能性があるため、再試行されることを意味します。 Error は、エラーの発生により、解決のために手動による対処が必要なことを意味します。
Reason	ステータスが False である理由を示します。準備ができるまで待機していることを示す場合や、障害の理由を示す場合があります。通常は、ステータスが False の場合にスローされます。
Message	Reason について説明する、人間が解読可能な情報です。

vSphere Client を使用した Tanzu Kubernetes クラスタのステータスの監視

vSphere Client を使用して、Tanzu Kubernetes クラスタのステータスを監視できます。

手順

1 vSphere Client を使用して、vCenter Server にログインします。

2 [メニュー] の [ホストおよびクラスタ] ビューを選択します。

3 スーパーバイザー クラスタ が作成されている [データセンター] - [クラスタ] オブジェクトを展開します。

4 [名前空間] リソース プールを展開します。

5 Tanzu Kubernetes クラスタがデプロイされている vSphere 名前空間 を選択します。

各 Tanzu Kubernetes クラスタは、名前空間リソース プール内のフォルダとして一覧表示されます。各 Tanzu Kubernetes は、名前の横にある 3 つの六角形のアイコンでグラフィカルに表されます。

6 [メニュー] - [仮想マシンおよびテンプレート] ビューに切り替えます。

クラスタ フォルダ内に、クラスタ ノードを構成する仮想マシンが表示されます。

7 vSphere 名前空間 を選択して、[コンピューティング] タブを選択します。

8 [VMware リソース] で [Tanzu Kubernetes] を選択します。

この vSphere 名前空間 にデプロイされている各 Tanzu Kubernetes クラスタが一覧表示されます。各ステータス フィールドの説明については、vSphere での Tanzu Kubernetes クラスタのライフサイクル ステータスを参照してください。

TKGS クラスタへのワークロードとパッケージのデプロイ

14

ワークロードとパッケージを Tanzu Kubernetes クラスタにインストールするには、このセクションの内容を参照してください。

この章には、次のトピックが含まれています。

- Tanzu Kubernetes クラスタへのワークロードのデプロイ
- Tanzu Kubernetes クラスタへの TKG パッケージのデプロイ
- Tanzu Kubernetes クラスタへの AI/ML ワークロードのデプロイ

Tanzu Kubernetes クラスタへのワークロードのデプロイ

アプリケーション ワークロードは、ポッド、サービス、パーシステント ボリュームのほか、デプロイやレプリカのセットなどの上位レベルのリソースを使用して Tanzu Kubernetes クラスタにデプロイできます。

Tanzu Kubernetes クラスタへのテスト ワークロードのデプロイ

Tanzu Kubernetes クラスタをプロビジョニングしたら、テスト ワークロードをデプロイして、クラスタの機能を確認することをお勧めします。

kuard デモ アプリケーションを使用して、Tanzu Kubernetes クラスタが実行中であることを確認します。

前提条件

- Tanzu Kubernetes クラスタをプロビジョニングします。TKGS v1alpha2 API を使用して Tanzu Kubernetes クラスタをプロビジョニングするためのワークフローを参照してください。
- vCenter Single Sign-On ユーザーとして Tanzu Kubernetes クラスタに接続

手順

1 構成コンテキストをターゲット Tanzu Kubernetes クラスタに切り替えます。

```
kubectl config use-context TANZU-KUBERNETES-CLUSTER-NAME
```

例：

```
kubectl config use-context tkgs-cluster-1
Switched to context "tkgs-cluster-1".
```

2 `kuard` デモ アプリケーションをデプロイします。

```
kubectl run --restart=Never --image=gcr.io/kuar-demo/kuard-amd64:blue kuard
```

予期される結果：

```
pod/kuard created
```

3 ポッドが実行されていることを確認します。

```
kubectl get pods
```

予期される結果：

```

NAME                     READY   STATUS    RESTARTS   AGE
kuard                    1/1     Running   0          10d
```

4 コンテナ ポート 8080 をローカル ホストのポート 8080 に転送します。

```
kubectl port-forward kuard 8080:8080
```

予期される結果：

```
Forwarding from 127.0.0.1:8080 -> 8080
Forwarding from [::1]:8080 -> 8080
Handling connection for 8080
```

5 ブラウザを使用して http://localhost:8080 に移動します。

kuard デモ アプリケーションの Web ページが表示され、クラスタの各要素を操作し、確認できるようになります。たとえば、稼動状態と準備状態のプローブを実行します。

6 kubectl セッションで `Ctrl+C` を押して、ポート転送を停止します。

7 `kuard` ポッドを削除します。

```
kubectl delete pod kuard
```

予期される結果：

```
pod "kuard" deleted
```

8 ポッドが削除されたことを確認します。

```
kubectl get pods
```

Octant のインストールと実行

Octant Web インターフェイスをインストールして、Tanzu Kubernetes クラスタのワークロード、名前空間、メタデータなどを可視化することができます。

Octant について

Octant は、Kubernetes クラスタとそのアプリケーションを表示するためのオープン ソース Web インターフェイスです。

`kubectl` を実行するクライアントに Octant をインストールして、実行します。以下に、共通プラットフォームのインストール手順を示します。詳細については、Octant のサイトを参照してください。

Octant のインストール後、`kubectl` を使用して Tanzu Kubernetes クラスタにログインし、`octant` コマンドを実行して使用を開始します。

Windows への Octant のインストール

Windows PowerShell 用の Chocolatey パッケージ マネージャをインストールします。

PowerShell セッションを管理者として実行します。

次のコマンドを使用して、Octant をインストールします。

```
choco install octant --confirm
```

Mac への Octant のインストール

Homebrew パッケージ マネージャをインストールします。

次のコマンドを使用して Octant をインストールします。

```
brew install octant
```

Ubuntu への Octant のインストール

リリース ページから `.deb` をダウンロードします。

`dpkg -i` コマンドを使用してインストールします。

Tanzu Kubernetes サービス ロード バランサの例

Tanzu Kubernetes クラスタで外部ロード バランサをプロビジョニングするには、LoadBalancer タイプのサービスを作成します。ロード バランサ サービスは、パブリック IP アドレスを公開します。外部ロード バランサからのトラフィックは、クラスタ ポッドで転送できます。

サービスとして公開される Kubernetes ポッドに外部ロード バランサをプロビジョニングできます。たとえば、Nginx コンテナをデプロイして、LoadBalancer タイプの Kubernetes サービスとして公開できます。

前提条件

- Kubernetes のドキュメントで LoadBalancer サービス タイプについて確認します。
- Tanzu Kubernetes クラスタをプロビジョニングします。TKGS v1alpha2 API を使用して Tanzu Kubernetes クラスタをプロビジョニングするためのワークフローを参照してください。
- ターゲットの Tanzu Kubernetes クラスタに接続します。vCenter Single Sign-On ユーザーとして Tanzu Kubernetes クラスタに接続を参照してください。

手順

1 デフォルトの特権 PSP に適したロール バインドを作成します。ポッド セキュリティ ポリシーのロール バインドの例を参照してください。

2 次の `nginx-lbsvc.yaml` YAML ファイルを作成します。

この YAML ファイルは、LoadBalancer タイプの Kubernetes サービスを定義し、このサービスの外部ロード バランサとして Nginx コンテナをデプロイします。

```
kind: Service
apiVersion: v1
metadata:
  name: srvclb-ngnx
spec:
  selector:
    app: hello
    tier: frontend
  ports:
  - protocol: "TCP"
    port: 80
    targetPort: 80
  type: LoadBalancer
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: loadbalancer
spec:
  replicas: 2
  selector:
    matchLabels:
      app: hello
  template:
    metadata:
      labels:
        app: hello
        tier: frontend
    spec:
      containers:
      - name: nginx
        image: "nginxdemos/hello"
```

3 YAML を適用します。

```
kubectl apply -f nginx-lbsvc.yaml
```

4 Nginx サービスが展開されていることを確認します。

```
kubectl get services
```

srvclb-ngnx は、外部および内部 IP アドレスで稼動しています。

```
NAME          TYPE           CLUSTER-IP       EXTERNAL-IP     PORT(S)        AGE
srvclb-ngnx   LoadBalancer   10.11.12.19      10.19.15.89     80:30818/TCP   18m
```

5 ブラウザを使用して、Nginx LoadBalancer サービスの外部 IP アドレスを入力します。

NGINX のメッセージ バナーとロード バランサの詳細が表示されます。

固定 IP アドレスを使用する Tanzu Kubernetes サービス ロード バランサの例

固定 IP アドレスを使用するように LoadBalancer タイプの Kubernetes サービスを構成できます。この機能を実装する場合は、コンポーネントの最小要件、セキュリティに関する重要な考慮事項、およびクラスタのセキュリティ強化に関するガイダンスに注意してください。

最小要件

次の要件を満たす Tanzu Kubernetes クラスタでは、LoadBalancer タイプの Kubernetes サービスに固定 IP アドレスを指定することができます。

コンポーネント	最小要件	詳細情報
vCenter Server と ESXi	vSphere 7.0 Update 2	リリース ノートを参照してください。
スーパーバイザー クラスタ	v1.19.1+vmware.2-vsc0.0.8-17610687	vSphere 名前空間の更新の実行による スーパーバイザー クラスタ の更新を参照してください。
ロード バランサ	NSX-T Data Center v3.1 または NSX Advanced 20.1.x	リリース ノートを参照してください。
Tanzu Kubernetes リリース	最新の Tanzu Kubernetes リリースの 1 つ。	更新のための Tanzu Kubernetes クラスタ互換性の確認を参照してください。

LoadBalancer タイプのサービスにおける固定 IP アドレスの使用

通常、LoadBalancer タイプの Kubernetes サービスを定義すると、ロード バランサによって IP アドレスが一時的に割り当てられます。Tanzu Kubernetes サービス ロード バランサの例を参照してください。

ロード バランサの固定 IP アドレスを指定することもできます。サービスの作成時に、割り当てられた固定 IP アドレスを使用してロード バランサのインスタンスがプロビジョニングされます。

次のサービスの例では、固定 IP アドレスを使用してサポート対象のロード バランサを構成する方法を示します。サービス仕様に loadBalancerIP パラメータと IP アドレス値を含めます。この例では、IP アドレスは 10.11.12.49 です。

```
kind: Service
apiVersion: v1
metadata:
  name: load-balancer-service-with-static-ip
```

```
spec:
  selector:
    app: hello-world
    tier: frontend
  ports:
  - protocol: "TCP"
    port: 80
    targetPort: 80
  type: LoadBalancer
  loadBalancerIP: 10.11.12.49
```

NSX Advanced Load Balancer の場合は、ロード バランサのインストール時に構成された IP アドレス管理プール内の IP アドレスを使用します。サービスが作成され、固定 IP アドレスが割り当てられると、ロード バランサは割り当て済みとマークし、一時的な IP アドレスと同じように IP アドレスのライフサイクルを管理します。つまり、サービスが削除されると、IP アドレスは割り当て解除され、再割り当てが可能になります。

NSX-T ロード バランサの場合は、2 つのオプションがあります。デフォルトのメカニズムは、NSX Advanced Load Balancer と同じように、ロード バランサのインストール時に構成された IP アドレス プールから取得された IP アドレスを使用します。固定 IP アドレスが割り当てられると、ロード バランサは自動的に割り当て済みとマークし、そのライフサイクルを管理します。

NSX-T の 2 番目のオプションでは、固定 IP アドレスを手動で事前に割り当てます。この場合、ロード バランサに割り当てられた外部ロード バランサの IP アドレス プールに含まれない、フローティング IP アドレス プールから取得された IP アドレスを使用します。この場合は、NSX Manager を使用して、IP アドレスの割り当てとライフサイクルを手動で管理します。

セキュリティに関する重要な検討事項とセキュリティ強化要件

この機能を使用する場合は、セキュリティ上の問題が発生する可能性を認識しておく必要があります。開発者が `Service.status.loadBalancerIP` 値にパッチを適用できる場合、その開発者は、パッチが適用された IP アドレス宛てのクラスタ内のトラフィックをハイジャックできる可能性があります。特に、この機能が実装されているクラスタで、`patch` 権限を持つ Role または ClusterRole がサービスまたはユーザー アカウントにバインドされている場合は、そのアカウント所有者が自身の認証情報を使用して `kubectl` コマンドを発行し、ロード バランサに割り当てられた固定 IP アドレスを変更できます。

ロード バランサ サービスに固定 IP アドレスを割り当てた場合に生じる可能性のあるセキュリティ面の影響を回避するには、この機能を実装している各クラスタのセキュリティを強化する必要があります。これを行うには、開発者に対して定義した Role または ClusterRole で、`apiGroups:""` および `resources: services/status` の verb として `patch` を禁止する必要があります。ロール スニペットの例は、この機能を実装する際の禁止項目を示しています。

パッチ適用の禁止

```
- apiGroups:
  - ""
  resources:
  - services/status
  verbs:
  - patch
```

開発者にパッチ権限が付与されているかどうかを確認するには、次のコマンドをそのユーザーとして実行します。

```
kubectl --kubeconfig <KUBECONFIG> auth can-i patch service/status
```

このコマンドによって yes が返された場合、そのユーザーにはパッチ権限が付与されています。詳細については、Kubernetes のドキュメントの Checking API Access を参照してください。

開発者にクラスタへのアクセス権を付与する方法については、開発者に対する Tanzu Kubernetes クラスタへのアクセス権の付与を参照してください。カスタマイズ可能なサンプルのロール テンプレートの例については、ポッド セキュリティ ポリシーのロールの例を参照してください。クラスタへのアクセスを制限する方法の例については、https://kubernetes.io/docs/reference/access-authn-authz/rbac/#role-example を参照してください。

Tanzu Kubernetes サービス ロード バランサのローカル トラフィック ポリシーと送信元 IP アドレス範囲に関する例

LoadBalancer タイプの Kubernetes サービスを構成して、受信要求の送信元 IP アドレスに基づいてロード バランサ トラフィックを許可し、ローカル ポッド トラフィックのみを許可することができます。

最小要件

次の最小要件を満たす Tanzu Kubernetes クラスタでは、LoadBalancer タイプの Kubernetes サービスと一緒に externalTrafficPolicy および LoadBalancerSourceRanges 機能を使用できます。

コンポーネント	最小要件	詳細情報
vCenter Server と ESXi	vSphere 7.0 Update 2	リリース ノートを参照してください。
スーパーバイザー クラスタ	v1.19.1+vmware.2-vsc0.0.8-17610687	vSphere 名前空間の更新の実行による スーパーバイザー クラスタ の更新を参照してください。
ロード バランサ	NSX-T Data Center v3.1	4 章 vSphere with Tanzu のネットワークを参照してください。
Tanzu Kubernetes リリース	最新の Tanzu Kubernetes リリースの 1 つ。	更新のための Tanzu Kubernetes クラスタ互換性の確認を参照してください。

ローカル トラフィック ポリシーと送信元 IP アドレス範囲のサポートについて

NSX-T Data Center ネットワークを使用している場合は、LoadBalancer タイプの Kubernetes サービスを構成して、外部トラフィック ポリシーとロード バランサの送信元 IP アドレス範囲を許可できます。externalTrafficPolicy 機能を使用すると、ポッド トラフィックをローカル ノードに制限できます。LoadBalancerSourceRange を使用すると、許可またはブロックする送信元 IP アドレスを指定できます。

ローカル トラフィック専用のサービスの例

次のロード バランサ サービス仕様では、externalTrafficPolicy パラメータを Local に設定してロード バランサ インスタンスを構成します。その結果、ポッド トラフィックはローカル ポッドが実行されているノードにのみルーティングされます。

```
apiVersion: v1
kind: Service
metadata:
  name: local-only
spec:
  selector:
    app: testApp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
  externalTrafficPolicy: Local
  type: LoadBalancer
```

この機能は、新しい NSX-T 健全性チェック モニタを使用して動作します。NSX-T 管理の観点から、この機能の内部動作を認識することが重要です。

NSX-T 健全性チェック モニタは、LoadBalancer タイプのサービスに対応するサーバ プールに対して kube-proxy によって割り当てられた Kubernetes 健全性チェック NodePort を監視します。この NSX-T 健全性チェック モニタは、ターゲットの健全性チェック NodePort に HTTP GET 要求を送信します。実行中のローカル ポッドがない場合、ノード上の kube-proxy は HTTP ステータス コード 500 を返します。ローカル ポッドがないノードは、NSX-T によって「切断」とマークされ、NSX Manager にそのように表示されます。トラフィックは、ローカル ポッドが実行されているノードにのみルーティングされます。

送信元 IP アドレス範囲に基づいてトラフィックを許可するサービスの例

次のロード バランサ サービス仕様では、許可された送信元 IP アドレスの CIDR の配列を使用して loadBalancerSourceRanges パラメータを構成します。これらの送信元 IP アドレス範囲から受信した要求のみが許可されます。その他の受信トラフィックはすべてドロップされます。

```
apiVersion: v1
kind: Service
metadata:
  name: allow-based-on-source-IPs
spec:
  selector:
    app: testApp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

```
  loadBalancerSourceRanges:
  - 10.0.0.0/24
  - 10.1.0.0/24
  type: LoadBalancer
```

NGINX を使用した Tanzu Kubernetes Ingress の例

Kubernetes Ingress のリソースは、クラスタ外からクラスタ内の 1 つ以上のサービスに HTTP または HTTPS ルーティングを提供します。Tanzu Kubernetes クラスタは、NGINX などのサードパーティ製コントローラを介した入力をサポートします。

このチュートリアルは、Tanzu Kubernetes クラスタ内のサービスに外部トラフィックをルーティングする際に、NGINX に基づく Kubernetes Ingress サービスをデプロイする方法を示しています。Ingress サービスには Ingress コントローラが必要です。NGINX Ingress Controller は、Helm を使用してインストールします。Helm は Kubernetes のパッケージ マネージャです。

注： このタスクを実行する方法はいくつかあります。ここに記載されている手順では、1 つの方法を示します。環境によっては、他の方法が適している場合があります。

前提条件

- Kubernetes のドキュメントで、Ingress のリソースについて確認します。
- NGINX Ingress Controller のドキュメントを確認してください。
- Tanzu Kubernetes クラスタをプロビジョニングします。TKGS v1alpha2 API を使用して Tanzu Kubernetes クラスタをプロビジョニングするためのワークフローを参照してください。
- ポッド セキュリティ ポリシーを有効にします。ポッド セキュリティ ポリシーのロールの例を参照してください。
- Tanzu Kubernetes クラスタに接続します。vCenter Single Sign-On ユーザーとして Tanzu Kubernetes クラスタに接続を参照してください。

手順

1. Helm ドキュメントを参照して、Helm をインストールします。
2. Helm を使用して NGINX Ingress Controller をインストールします。

```
helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
helm install ingress-nginx ingress-nginx/ingress-nginx
```

3 NGINX Ingress Controller が LoadBalancer タイプのサービスとしてデプロイされていることを確認します。

```
kubectl get services
```

```
NAME                                 TYPE           CLUSTER-IP     EXTERNAL-IP
PORT(S)                      AGE
ingress-nginx-controller             LoadBalancer   10.16.18.20    10.19.14.76   80:30635/
TCP,443:30873/TCP   59m
ingress-nginx-controller-admission   ClusterIP      10.87.41.25    <none>
443/TCP                      59m
```

4 外部 IP アドレスを使用してロード バランサを ping します。

```
ping 10.19.14.76
```

```
Pinging 10.19.14.76 with 32 bytes of data:
Reply from 10.19.14.76: bytes=32 time<1ms TTL=62
Reply from 10.19.14.76: bytes=32 time=1ms TTL=62
```

5 NGINX Ingress Controller が実行されていることを確認します。

```
kubectl get pods
```

```
NAME                                        READY   STATUS    RESTARTS   AGE
ingress-nginx-controller-7c6c46898c-v6blt   1/1     Running   0          76m
```

6 Ingress ルールと `ingress-hello.yaml` という名前のパスを含む Ingress リソースを作成します。

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-hello
spec:
  rules:
  - http:
      paths:
      - path: /hello
        backend:
          serviceName: hello
          servicePort: 80
```

7 `ingress-hello` リソースをデプロイします。

```
kubectl apply -f ingress-hello.yaml
```

```
ingress.networking.k8s.io/ingress-hello created
```

8 Ingress リソースがデプロイされたことを確認します。

IP アドレスは Ingress コントローラの外部 IP アドレスにマッピングされることに注意してください。

```
kubectl get ingress
```

```
NAME            CLASS    HOSTS   ADDRESS         PORTS   AGE
ingress-hello   <none>   *       10.19.14.76     80      51m
```

9 hello テスト アプリケーションおよび `ingress-hello-test.yaml` という名前のサービスを作成します。

```
kind: Service
apiVersion: v1
metadata:
  name: hello
spec:
  selector:
    app: hello
    tier: backend
  ports:
  - protocol: TCP
    port: 80
    targetPort: http
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello
spec:
  replicas: 3
  selector:
    matchLabels:
      app: hello
      tier: backend
      track: stable
  template:
    metadata:
      labels:
        app: hello
        tier: backend
        track: stable
    spec:
      containers:
        - name: hello
          image: "gcr.io/google-samples/hello-go-gke:1.0"
          ports:
            - name: http
              containerPort: 80
```

10 `ingress-hello-test` リソースをデプロイします。

```
kubectl apply -f ingress-hello-test.yaml
```

```
service/hello created
deployment.apps/hello created
```

11 `hello` デプロイを使用できることを確認します。

```
kubectl get deployments
```

```
NAME                         READY   UP-TO-DATE   AVAILABLE   AGE
hello                        3/3     3            3           4m59s
ingress-nginx-controller     1/1     1            1           3h39m
```

12 NGINX Ingress Controller によって使用されるロード バランサのパブリック IP アドレスを取得します。

```
kubectl get ingress
```

```
NAME            CLASS    HOSTS   ADDRESS         PORTS   AGE
ingress-hello   <none>   *       10.19.14.76     80      13m
```

13 ブラウザを使用してパブリック IP アドレスに移動し、Ingress のパスを含めます。

```
http://10.19.14.76/hello
```

「hello」というメッセージが返されます。

```
{"message":"Hello"}
```

結果

クラスタ内で実行されているサービスの背後にあるバックエンド アプリケーションに外部からアクセスするには、ロード バランサの外部 IP アドレスを使用して、Ingress コントローラを介してブラウザからアクセスします。

Tanzu Kubernetes ストレージ クラスの例

永続性を必要とするワークロードの場合は、デフォルトのストレージ クラスを使用するか、パーシステント ボリュームで使用する独自のストレージ クラスを定義します。Tanzu Kubernetes クラスタは、コンテナ ストレージ インターフェイス (CSI) のプロビジョナをサポートしています。

コンテナ ストレージ インターフェイス (CSI) はサポート対象

Tanzu Kubernetes クラスタは、コンテナ ストレージ インターフェイス (CSI) をサポートしています。`StorageClass` の定義では、このタイプのプロビジョナは `csi.vsphere.vware.com` と識別されます。

次の YAML 定義をテンプレートとして使用して、Tanzu Kubernetes クラスタのストレージ クラスを定義できます。ストレージ クラスをデフォルト（「true」）にするかどうかを指定し、ストレージ環境のデータストア URL を指定します。

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: tkgs-storage-class
  annotations:
    storageclass.kubernetes.io/is-default-class: "true" or "false"
provisioner: csi.vsphere.vmware.com
parameters:
  datastoreurl: "ds:///vmfs/volumes/vsan:52d8eb4842dbf493-41523be9cd4ff7b7/"
```

次のストレージ クラスを作成します。

```
kubectl apply -f tkgs-storage-class.yaml

storageclass.storage.k8s.io/tkgs-storage-class created
```

ストレージ クラスが作成されたことを確認します。

```
kubectl get storageclass
```

または、次のショートカットを使用します。

```
kubectl get sc
```

VMware Cloud Provider (vCP) は非サポート対象

Tanzu Kubernetes クラスタは、次に示すようにレガシー VMware Cloud Provider (vCP) `StorageClass` をサポートしていません。vCP プロビジョナを使用して `StorageClass` を作成しようとしても、`StorageClass` は作成されません。

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: demo-sts-sc
provisioner: kubernetes.io/vsphere-volume
parameters:
  diskformat: thin
```

Tanzu Kubernetes パーシステント ボリュームの要求例

Tanzu Kubernetes クラスタでステートフル ワークロードを実行するために、パーシステント ボリューム要求 (PVC) を作成して、基盤となるストレージ インフラストラクチャの詳細を把握していなくてもパーシステント ストレージ リソースを要求することができます。PVC に使用されるストレージは、vSphere 名前空間 用のストレージ割り当てから割り当てられます。

コンテナはデフォルトで短期、かつステートレスです。ステートフル ワークロードの場合、一般的な方法は、パーシステント ボリューム要求 (PVC) を作成することです。PVC を使用してパーシステント ボリュームをマウントし、ストレージにアクセスできます。この要求により、パーシステント ボリューム オブジェクトとそれに対応する仮想ディスクが動的にプロビジョニングされます。要求はパーシステント ボリュームにバインドされています。この要求を削除すると、対応するパーシステント ボリューム オブジェクトおよびプロビジョニングされた仮想ディスクも削除されます。

手順

1 ターゲットの Tanzu Kubernetes クラスタにログインします。vCenter Single Sign-On ユーザーとして Tanzu Kubernetes クラスタに接続を参照してください。

2 クラスタが実行されている名前空間に切り替えます。

```
kubectl config use-context NAMESPACE
```

3 ストレージ クラスを検証するか、作成します。

既存のストレージ クラスを検証するには、次の手順を実行します。

```
kubectl get storageclass
```

ストレージ クラスを作成する手順については、Tanzu Kubernetes ストレージ クラスの例を参照してください。

4 名前空間を作成します。

```
kubectl create namespace guestbook
```

5 Guestbook PVC YAML ファイルを作成します。

- Redis リーダーの PVC
- Redis フォロワーの PVC

6 クラスタに Guestbook PVC を適用します。

```
kubectl apply -f redis-leader-pvc.yaml -n guestbook
```

```
kubectl apply -f redis-follower-pvc.yaml -n guestbook
```

7 PVC のステータスを確認します。

```
kubectl get pvc,pv -n guestbook
```

PVC とパーシステント ボリューム (PV) が一覧表示され、使用できるようになります。

```
NAME                                        STATUS
VOLUME                                      CAPACITY   ACCESS MODES   STORAGECLASS
AGE
persistentvolumeclaim/redis-follower-pvc    Bound    pvc-37b72f35-3de2-4f84-
be7d-50d5dd968f62   2Gi        RWO            tkgs-storage-class   66s
```

```
persistentvolumeclaim/redis-leader-pvc     Bound    pvc-2ef51f31-dd4b-4fe2-bf4c-
f0149cb4f3da   2Gi        RWO            tkgs-storage-class   66s

NAME                                                    CAPACITY   ACCESS MODES   RECLAIM
POLICY STATUS   CLAIM                  STORAGECLASS
persistentvolume/pvc-2ef51f31-dd4b-4fe2-bf4c   2Gi   RWO            Delete
Bound    guestbook/redis-leader-pvc     tkgs-storage-class
persistentvolume/pvc-37b72f35-3de2-4f84-be7d   2Gi   RWO            Delete
Bound    guestbook/redis-follower-pvc   tkgs-storage-class
```

Tanzu Kubernetes Guestbook のチュートリアル

Guestbook アプリケーションを Tanzu Kubernetes クラスタにデプロイして、サービス アカウントのポッド セキュリティ ポリシー、およびデプロイとサービスの作成について確認します。

Guestbook アプリケーションのデプロイは、Kubernetes を確認する際の一般的な手段です。Tanzu Kubernetes Grid サービス によってプロビジョニングされた Tanzu Kubernetes クラスタにすべてのゲストブック YAML ファイルをデプロイすると、アプリケーション ポッドは正常に作成されません。`kubectl describe pod` コマンドを実行すると、次のエラー メッセージが表示されます。

```
“Error: container has runAsNonRoot and image will run as root”
```

Guestbook アプリケーションは、`deployment` と `replicaset` の両方のリソースを使用して特権コンテナをデフォルトの名前空間にデプロイします。Tanzu Kubernetes クラスタでは PodSecurityPolicy コントローラが有効になっているため、クラスタ ユーザーが Guestbook アプリケーション ポッドを作成しようとすると、これらのコントローラのサービス アカウントが PodSecurityPolicy に対してチェックされます。適切な PSP がこれらのサービス アカウントにバインドされていない場合、アプリケーションはデプロイされません。

デフォルトでは、Tanzu Kubernetes 管理者はそれぞれのユーザー アカウントを使用して、任意の名前空間に特権ポッドを直接作成できます。ただし、Guestbook アプリケーションはサービス アカウントを使用して特権コンテナをデプロイします。クラスタ管理者は、`kube-system` 名前空間に Deployments、StatefulSets、および DaemonSet を作成できます。ただし、Guestbook アプリケーションはこれらのリソースをデフォルトの名前空間にデプロイします。また、管理者以外のユーザーは、適切な PSP やバインドがなければ特権ポッドも権限のないポッドも一切作成できません。

1 つの解決策は、デフォルトの特権 PSP へのバインドを作成して、Guestbook アプリケーションをデプロイできるようにすることです。特権 PodSecurityPolicy により、バインドされているアカウントに対して、root として実行されるポッドと特権コンテナが許可されます。`vmware-system-privileged` クラスタ全体に適用される ClusterRoleBinding を作成できますが、それによって必要以上の権限を付与することになり、最小限の権限の原則に違反する可能性があります。より適切な方法は、システム サービス アカウントにデフォルトの名前空間で特権 PodSecurityPolicy の使用を許可するよう RoleBinding を作成することです。詳細については、『ポッド セキュリティ ポリシーのロール バインドの例』を参照してください。

前提条件

次のトピックを参照してください。

- Kubernetes ドキュメントの「Guestbook アプリケーションのチュートリアル」
- Tanzu Kubernetes クラスタでのポッド セキュリティ ポリシーの使用

- ポッド セキュリティ ポリシーのロール バインドの例

手順

1 Tanzu Kubernetes クラスタにログインします。vCenter Single Sign-On ユーザーとして Tanzu Kubernetes クラスタに接続を参照してください。

2 Guestbook 名前空間を作成します。

```
kubectl create namespace guestbook
```

以下を確認します。

```
kubectl get ns
```

3 デフォルトの特権 PSP を使用して、ロールベースのアクセス制御を作成します。

```
kubectl create clusterrolebinding default-tkg-admin-privileged-binding --
clusterrole=psp:vmware-system-privileged --group=system:authenticated
```

注： セキュリティを強化する必要がある場合は、Guestbook 名前空間に RoleBinding を適用します。『ポッド セキュリティ ポリシーのロール バインドの例』を参照してください

4 ストレージ クラスを検証するか、作成します。

既存のストレージ クラスを検証するには、次の手順を実行します。

```
kubectl get storageclass
```

ストレージ クラスを作成する手順については、Tanzu Kubernetes ストレージ クラスの例を参照してください。

5 ストレージ クラスを使用するパーシステント ボリュームの要求 (PVC) を作成します。

次の YAML ファイルを使用します。

- Redis リーダーの PVC
- Redis フォロワーの PVC

PVC を作成する手順については、Tanzu Kubernetes パーシステント ボリュームの要求例を参照してください。

6 Guestbook YAML ファイルを作成します。

次の YAML ファイルを使用します。

- Redis リーダーのデプロイ
- Redis リーダーのサービス
- Redis フォロワーのデプロイ
- Redis フォロワー サービス
- Guestbook フロントエンドのデプロイ

- Guestbook フロントエンド サービス

7 名前空間に Guestbook アプリケーションをデプロイします。

```
kubectl apply -f . --namespace guestbook
```

8 Guestbook リソースの作成を確認します。

```
kubectl get all -n guestbook
```

```
NAME                                                READY   STATUS
RESTARTS   AGE
pod/guestbook-frontend-deployment-56fc5b6b47-cd58r   1/1     Running
0          65s
pod/guestbook-frontend-deployment-56fc5b6b47-fh6dp   1/1     Running
0          65s
pod/guestbook-frontend-deployment-56fc5b6b47-hgd2b   1/1     Running
0          65s
pod/redis-follower-deployment-6fc9cf5759-99fgw       1/1     Running
0          65s
pod/redis-follower-deployment-6fc9cf5759-rhxf7       1/1     Running
0          65s
pod/redis-leader-deployment-7d89bbdbcf-flt4q         1/1     Running
0          65s

NAME                                 TYPE           CLUSTER-IP       EXTERNAL-IP
PORT(S)        AGE
service/guestbook-frontend           LoadBalancer   10.10.89.59      10.19.15.99
80:31513/TCP   65s
service/redis-follower               ClusterIP      10.111.163.189   <none>
6379/TCP       65s
service/redis-leader                 ClusterIP      10.111.70.189    <none>
6379/TCP       65s

NAME                                          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/guestbook-frontend-deployment   3/3     3            3           65s
deployment.apps/redis-follower-deployment       1/2     2            1           65s
deployment.apps/redis-leader-deployment         1/1     1            1           65s

NAME                                                DESIRED   CURRENT   READY   AGE
replicaset.apps/guestbook-frontend-deployment-56fc5b6b47   3         3         3       65s
replicaset.apps/redis-follower-deployment-6fc9cf5759       2         2         1       65s
replicaset.apps/redis-leader-deployment-7d89bbdbcf         1         1         1       65s
```

9 `service/guestbook-frontend` ロード バランサの `External-IP` アドレス（この例では `10.19.15.99`）を使用して Guestbook Web ページにアクセスします。

Guestbook Web インターフェイスが表示され、Guestbook データベースに値を入力できます。アプリケーションを再起動すると、データは保持されます。

Guestbook のサンプル YAML ファイル

サンプル YAML ファイルを使用して、パーシステント データとともに Guestbook アプリケーションをデプロイします。

Redis リーダーの PVC

ファイル `redis-leader-pvc.yaml` は、指定したストレージ クラスを参照するパーシステント ボリュームの要求の例です。この例を使用するには、ストレージ クラスの名前を入力します。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: redis-leader-pvc
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: tkgs-storage-class-name
  resources:
    requests:
      storage: 2Gi
```

Redis フォロワーの PVC

ファイル `redis-follower-pvc.yaml` は、指定したストレージ クラスを参照するパーシステント ボリュームの要求の例です。この例を使用するには、ストレージ クラスの名前を入力します。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: redis-follower-pvc
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: tkgs-storage-class-name
  resources:
    requests:
      storage: 2Gi
```

Redis リーダーのデプロイ

ファイル `redis-leader-deployment.yaml` は、パーシステント ボリュームを使用した Redis リーダーのデプロイ例です。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: redis-leader-deployment
spec:
  selector:
    matchLabels:
      app: redis
      role: leader
```

```
      tier: backend
  replicas: 1
  template:
    metadata:
      labels:
        app: redis
        role: leader
        tier: backend
    spec:
      containers:
      - name: leader
        image: redis:6.0.5
        resources:
          requests:
            cpu: 100m
            memory: 100Mi
        ports:
        - containerPort: 6379
        volumeMounts:
        - name: redis-leader-data
          mountPath: /data
      volumes:
      - name: redis-leader-data
        persistentVolumeClaim:
          claimName: redis-leader-pvc
```

Redis フォロワーのデプロイ

ファイル `redis-follower-deployment.yaml` は、パーシステント ボリュームを使用した Redis フォロワーのデプロイ例です。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: redis-follower-deployment
  labels:
    app: redis
spec:
  selector:
    matchLabels:
      app: redis
      role: follower
      tier: backend
  replicas: 1
  template:
    metadata:
      labels:
        app: redis
        role: follower
        tier: backend
    spec:
      containers:
      - name: follower
        image: gcr.io/google_samples/gb-redis-follower:v2
```

```
        resources:
          requests:
            cpu: 100m
            memory: 100Mi
        env:
        - name: GET_HOSTS_FROM
          value: dns
        ports:
        - containerPort: 6379
        volumeMounts:
        - name: redis-follower-data
          mountPath: /data
      volumes:
      - name: redis-follower-data
        persistentVolumeClaim:
          claimName: redis-follower-pvc
```

Redis リーダーのサービス

ファイル `redis-leader-service.yaml` は、Redis リーダーのサービス例です。

```
apiVersion: v1
kind: Service
metadata:
  name: redis-leader
  labels:
    app: redis
    role: leader
    tier: backend
spec:
  ports:
  - port: 6379
    targetPort: 6379
  selector:
    app: redis
    role: leader
    tier: backend
```

Redis フォロワー サービス

ファイル `redis-follower-service.yaml` は、Redis フォロワーのサービス例です。

```
apiVersion: v1
kind: Service
metadata:
  name: redis-follower
  labels:
    app: redis
    role: follower
    tier: backend
spec:
  ports:
  - port: 6379
```

```
  selector:
    app: redis
    role: follower
    tier: backend
```

Guestbook フロントエンドのデプロイ

ファイル `guestbook-frontend-deployment.yaml` は、Guestbook フロントエンドのデプロイ例です。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: guestbook-frontend-deployment
spec:
  selector:
    matchLabels:
      app: guestbook
      tier: frontend
  replicas: 3
  template:
    metadata:
      labels:
        app: guestbook
        tier: frontend
    spec:
      containers:
      - name: php-redis
        image: gcr.io/google_samples/gb-frontend:v5
        resources:
          requests:
            cpu: 100m
            memory: 100Mi
        env:
        - name: GET_HOSTS_FROM
          value: dns
        ports:
        - containerPort: 80
```

Guestbook フロントエンド サービス

ファイル `guestbook-frontend-service.yaml` は、Guestbook フロントエンド ロード バランサ サービスの例です。

```
apiVersion: v1
kind: Service
metadata:
  name: guestbook-frontend
  labels:
    app: guestbook
    tier: frontend
spec:
  type: LoadBalancer
  ports:
```

```
  - port: 80
  selector:
    app: guestbook
    tier: frontend
```

Tanzu Kubernetes クラスタでのポッド セキュリティ ポリシーの使用

Tanzu Kubernetes Grid サービス は、PodSecurityPolicy アドミッション コントローラが有効な状態で Tanzu Kubernetes クラスタをプロビジョニングします。これは、ワークロードをデプロイするためにはポッド セキュリティ ポリシーが必要であることを意味します。クラスタ管理者は、ユーザー アカウントから任意の名前空間に、およびサービス アカウントから kube-system 名前空間にポッドをデプロイできます。その他のすべてのユースケースでは、PodSecurityPolicy オブジェクトに明示的にバインドする必要があります。クラスタには、バインド可能なデフォルトのポッド セキュリティ ポリシーがあります。また、独自のポッド セキュリティ ポリシーをクラスタに作成することもできます。

Kubernetes のポッド セキュリティ ポリシーについて

Kubernetes のポッド セキュリティ ポリシー (PSP) は、ポッドのセキュリティを制御するクラスタレベルのリソースです。PSP を使用すると、デプロイできるポッドのタイプと、それらをデプロイできるアカウントのタイプを制御できます。

PodSecurityPolicy リソースは、デプロイのためにポッドが満たす必要のある必要な一連の条件を定義しています。条件が満たされていない場合は、ポッドをデプロイできません。1 つの PodSecurityPolicy で 1 つのポッド全体を検証する必要があります。ポッドのルールを複数のポリシーに分けて定義することはできません。

Kubernetes では、さまざまな方法でポッド セキュリティ ポリシーを使用できます。一般的なアプローチは、ロールベースのアクセス コントロール (RBAC) オブジェクトを使用するものです。ClusterRole および ClusterRoleBinding は、クラスタ全体に適用されます。Role および RoleBinding は、特定の名前空間に適用されます。RoleBinding を使用すると、ポッドはこのバインドと同じ名前空間内でのみ実行されます。

Kubernetes ポッドの作成には、直接的な方法と間接的な方法の 2 つがあります。ユーザー アカウントを使用してポッドの仕様をデプロイすることにより、ポッドを直接作成します。Deployment や DaemonSet などの上位レベルのリソースを定義することにより、ポッドを間接的に作成します。この場合、基盤となるポッドがサービス アカウントによって作成されます。

PSP を効果的に使用するには、両方のポッド作成ワークフローを考慮する必要があります。ユーザーがポッドを直接作成すると、そのユーザー アカウントにバインドされた PSP によって操作が制御されます。ユーザーがサービス アカウントを使用してポッドを作成する場合、PSP は、ポッドの作成に使用するサービス アカウントにバインドする必要があります。ポッド仕様でサービス アカウントが指定されていない場合は、名前空間のデフォルトのサービス アカウントが使用されます。

詳細については、Kubernetes のドキュメントでポッド セキュリティ ポリシー、RBAC、およびサービス アカウントについて参照してください。

Tanzu Kubernetes クラスタのデフォルトの PodSecurityPolicy

次の表に、Tanzu Kubernetes クラスタについて権限のある、および制限されたデフォルトのポッド セキュリティ ポリシーと、各ポリシーに関連付けられているデフォルトの ClusterRole の一覧と説明を示します。

表 14-1. デフォルトの PodSecurityPolicy および関連付けられている ClusterRole

デフォルトの PSP	権限	説明	関連付けられているデフォルトの ClusterRole
`vmware-system-privileged`	任意のユーザーとして実行	許可的な PSP。PSP アドミッション コントローラが有効になっていないクラスタの実行と同等です。	`psp:vmware-system-privileged` がこの PSP を使用可能
`vmware-system-restricted`	非 root として実行	制限的な PSP。ポッド コンテナへの特権アクセスを許可せず、root へのエスカレーションをブロックし、いくつかのセキュリティ メカニズムを使用する必要があります。	`psp:vmware-system-restricted` がこの PSP を使用可能

Tanzu Kubernetes クラスタのデフォルト バインドはない

Tanzu Kubernetes Grid サービス には、Tanzu Kubernetes クラスタのデフォルトの RoleBinding と ClusterRoleBinding はありません。

vSphere 名前空間 に対する編集権限が付与されている vCenter Single Sign-On ユーザーは、その名前空間でデプロイされるすべての Tanzu Kubernetes クラスタで cluster-admin ロールに割り当てられます。認証されたクラスタ管理者は、暗黙的に `vmware-system-privileged` PSP を使用できます。厳密には ClusterRoleBinding とは異なりますが、実質的には同じです。

クラスタ管理者は、ユーザーがクラスタにデプロイできるポッドのタイプを許可または制限するために、何らかのバインドを定義する必要があります。RoleBinding を使用すると、バインドと同じ名前空間でのみポッドの実行が許可されます。これをシステム グループと組み合わせると、名前空間で実行されているすべてのポッドへのアクセスを許可することができます。クラスタに対して認証を行う管理者以外のユーザーは、`authenticated` ロールに割り当てられ、デフォルトの PSP にバインドできます。開発者に対する Tanzu Kubernetes クラスタへのアクセス権の付与を参照してください。

Tanzu Kubernetes クラスタでのデフォルトの PodSecurityPolicy の効果

すべての Tanzu Kubernetes クラスタで次の動作が適用されます。

- クラスタ管理者はそれぞれのユーザー アカウントを使用して、任意の名前空間に特権ポッドを直接作成できます。
- クラスタ管理者は、kube-system 名前空間に Deployments、StatefulSets、および DaemonSet（それぞれが特権ポッドを作成）を作成できます。別の名前空間を使用する場合は、Tanzu Kubernetes Guestbook のチュートリアルを参照してください。
- クラスタ管理者は、独自の PSP を（2 つのデフォルトの PSP に加えて）作成して任意のユーザーにバインドすることができます。独自の PSP を定義する場合は、Kubernetes のドキュメントでポリシーの順序について参照してください。
- クラスタ管理者が PSP を認証されたユーザーにバインドするまで、認証されたユーザーは、特権の有無にかかわらずポッドを作成することはできません。Tanzu Kubernetes Guestbook のチュートリアルを参照してください。

ポッド セキュリティ ポリシーのロール バインドの例

Tanzu Kubernetes クラスタには、権限のある、および制限されたワークロードのデプロイのためにバインドできるデフォルトの PodSecurityPolicy が含まれています。

デフォルトのポッド セキュリティ ポリシーについて

このセクションでは、ClusterRoleBinding や RoleBinding などのデフォルトのポッド セキュリティ ポリシーにロール バインド オブジェクトを作成するための YAML および CLI コマンドについて説明します。詳細については、『Tanzu Kubernetes クラスタでのポッド セキュリティ ポリシーの使用』を参照してください。

RoleBinding は特定の名前空間内の権限を付与しますが、ClusterRoleBinding はクラスタ全体に対する権限を付与します。RoleBindings または ClusterRoleBinding を使用するかどうかは、使用方法によって異なります。たとえば、ClusterRoleBinding を使用して、`system:serviceaccounts:<namespace>` を使用するようにサブジェクトを構成する場合は、名前空間を作成する前に PSP にバインドできます。詳細については、Kubernetes ドキュメントの RoleBinding および ClusterRoleBinding を参照してください。

例 1：権限のあるワークロードのセットを実行するための ClusterRoleBinding

次の kubectl コマンドにより、ClusterRoleBinding が作成されます。これは、デフォルトの PSP `vmware-system-privileged` を使用して権限のあるワークロードのセットを実行する認証済みユーザーに、アクセス権を付与します。

注意: 例 1 を宣言または命令によって適用すると、クラスタ全体で権限を付与されたワークロードのデプロイが可能になります。実際、例 1 では、ネイティブのセキュリティ制御が無効になるため、注意を払い、影響を十分に認識して使用する必要があります。セキュリティを強化するには、例 2、3、および 4 を検討してください。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: psp:privileged
rules:
- apiGroups: ['policy']
  resources: ['podsecuritypolicies']
  verbs:     ['use']
  resourceNames:
  - vmware-system-privileged
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: all:psp:privileged
roleRef:
  kind: ClusterRole
  name: psp:privileged
  apiGroup: rbac.authorization.k8s.io
subjects:
- kind: Group
  name: system:serviceaccounts
  apiGroup: rbac.authorization.k8s.io
```

YAML を適用する代わりに、次の `kubectl` コマンドを実行することもできます。

```
kubectl create clusterrolebinding default-tkg-admin-privileged-binding --
clusterrole=psp:vmware-system-privileged --group=system:authenticated
```

例 2：権限のあるワークロードのセットを実行するための RoleBinding

次の kubectl コマンドは、デフォルトの PSP `vmware-system-privileged` を使用して、権限のあるワークロードのセットを実行するためにデフォルトの名前空間内のすべてのサービス アカウントへのアクセスを許可する RoleBinding を作成します。

```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: rolebinding-default-privileged-sa-ns_default
  namespace: default
roleRef:
  kind: ClusterRole
  name: psp:vmware-system-privileged
  apiGroup: rbac.authorization.k8s.io
subjects:
- kind: Group
  apiGroup: rbac.authorization.k8s.io
  name: system:serviceaccounts
```

YAML を適用する代わりに、次の kubectl コマンドを実行することもできます。

```
kubectl create rolebinding rolebinding-default-privileged-sa-ns_default --namespace=default --
clusterrole=psp:vmware-system-privileged --group=system:serviceaccounts
```

例 3：制限されたワークロードのセットを実行するための ClusterRoleBinding

次の YAML は、デフォルトの PSP `vmware-system-restricted` を使用して、制限されたワークロードのセットを実行するためにクラスタ全体へのアクセスを認証ユーザーに許可する ClusterRoleBinding を作成します。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: psp:authenticated
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: system:authenticated
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: psp:vmware-system-restricted
```

YAML を適用する代わりに、次の kubectl コマンドを実行することもできます。

```
kubectl create clusterrolebinding psp:authenticated --clusterrole=psp:vmware-system-
restricted --group=system:authenticated
```

例 4：制限されたワークロードのセットを実行するための RoleBinding

次の YAML は、デフォルトの PSP `vmware-system-restricted` を使用して、制限されたワークロードのセットを実行するために特定の名前空間内のすべてのサービス アカウントへのアクセスを許可する RoleBinding を作成します。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: psp:serviceaccounts
  namespace: some-namespace
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: system:serviceaccounts
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: psp:vmware-system-restricted
```

YAML を適用する代わりに、次の kubectl コマンドを実行することもできます。

```
kubectl create rolebinding psp:serviceaccounts --clusterrole=psp:vmware-system-restricted --
group=system:serviceaccounts
```

ポッド セキュリティ ポリシーのロールの例

Tanzu Kubernetes クラスタでは、ワークロードをデプロイするためにポッド セキュリティ ポリシー (PSP) が必要です。独自の PSP を定義する場合は、PSP を参照する Role または ClusterRole を作成する必要があります。

PodSecurityPolicy のロールの例

次の例は、PodSecurityPolicy にバインドされたロールを示しています。ロールの定義で `use` を verb として使用して、定義するカスタム PSP リソースに `example-role` を付与します。または、デフォルトの PSP の 1 つを使用します。次に、Tanzu Kubernetes Guestbook のチュートリアルを作成します。

```
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: Role
metadata:
  name: example-role
  namespace: tkgs-cluster-ns
rules:
- apiGroups:
```

```
  - ""
  resources:
  - configmaps
  verbs:
  - create
  - get
  - list
  - watch
  - update
- apiGroups:
  - ""
  resources:
  - events
  verbs:
  - create
  - update
  - patch
- apiGroups:
  - extensions
  resourceNames:
  - CUSTOM-OR-DEFAULT-PSP
  resources:
  - podsecuritypolicies
  verbs:
  - use
```

Tanzu Kubernetes クラスタへの TKG パッケージのデプロイ

TKGS によってプロビジョニングされた Tanzu Kubernetes クラスタに TKG パッケージをデプロイするには、ここに示された一連の手順を参照してください。

TKG 1.6 パッケージ

TKGS によってプロビジョニングされた Tanzu Kubernetes クラスタに TKG 1.6 パッケージをインストールするには、TKG 1.6 パッケージのインストールと構成を参照してください。

TKG 1.4 パッケージ

TKGS によってプロビジョニングされた Tanzu Kubernetes クラスタに TKG 1.4 パッケージをインストールするには、TKG 1.4 パッケージのインストールと構成を参照してください。

TKG 1.3.1 拡張機能

TKGS によってプロビジョニングされた Tanzu Kubernetes クラスタに TKG 1.3.1 拡張機能をインストールするには、このセクションに記載されたドキュメントを参照してください。

注： TKG パッケージは TKG 拡張機能に代わるものです。拡張機能の代わりにパッケージを使用することを検討してください。

TKG 拡張機能 v1.3.1 バンドルのダウンロード

1 つ以上の TKG 拡張機能をデプロイする準備として、VMware から TKG 拡張機能 v1.3.1 バンドルをダウンロードします。

TKG 拡張機能は、VMware からダウンロードされた個別のバンドルとしてパッケージ化されています。

注： vSphere with Tanzu は vSphere 7 U2 以降で TKG 拡張機能 v1.3.1 をサポートしています。

手順

1 https://www.vmware.com/go/get-tkg にアクセスします。

2 My VMware の認証情報を使用してログインします。

3 [製品のダウンロード] - [ダウンロードに移動] の順に選択します。

4 ドロップダウン メニューからバージョン [1.3.1] を選択します。

5 [VMware Tanzu Kubernetes Grid Extensions Manifest 1.3.1] までスクロールして特定します。

6 [今すぐダウンロード] をクリックして、ファイル `tkg-extensions-manifests-v1.3.1-vmware.1.tar.gz` をローカル システムにダウンロードします。

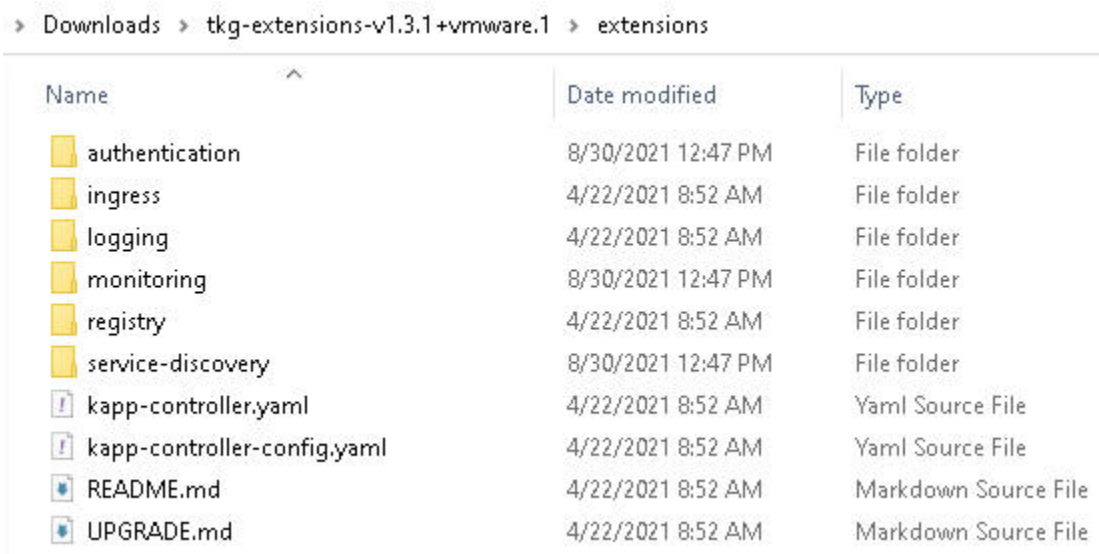
7 `kubectl` コマンドを実行するコンピュータに、ファイル `tkg-extensions-manifests-v1.3.1-vmware.1.tar.gz` をコピーします。

8 `tar` コマンドまたは任意の抽出ツールを使用して、TKG 拡張機能ファイルを抽出します。

```
tar -xzf tkg-extensions-manifests-v1.3.1-vmware.1.tar.gz
```

9 次のファイル パスに移動して、TKG 拡張機能ファイルを確認します。

```
cd /tkg-extensions-v1.3.1+vmware.1/extensions
```



これは、TKG 拡張機能のホーム ディレクトリです。

TKG 拡張機能の前提条件のインストール

1 つ以上の TKG 拡張機能 v1.3.1 をインストールする各 Tanzu Kubernetes クラスタに、前提条件となるアプリケーションをインストールします。

TKG 拡張機能 v1.3.1 には、Kapp Controller と Cert Manager という、前提条件となる 2 つのコンポーネントが必要です。

注： Cert Manager の代わりに、独自の TLS 証明書を使用することもできます。TKG 拡張機能における独自の TLS 証明書の使用を参照してください。

前提条件

- Tanzu Kubernetes クラスタをプロビジョニングします。TKGS v1alpha2 API を使用して Tanzu Kubernetes クラスタをプロビジョニングするためのワークフローを参照してください。
- Tanzu Kubernetes クラスタに接続します。vCenter Single Sign-On ユーザーとして Tanzu Kubernetes クラスタに接続を参照してください。

手順

1 TKG 拡張機能 v1.3.1 バンドルのダウンロード。

2 クラスタに Cert Manager をインストールします。

ダウンロードして展開した TKG 拡張機能バンドルのルート ディレクトリに移動します。

```
cd /tkg-extensions-v1.3.1+vmware.1
```

Cert Manager にはいくつかのコンポーネントが含まれています。`/cert-manager` ディレクトリに 3 つの YAML ファイルがあります。`ls` を使用して、このディレクトリがあることを確認します。

次の単一コマンドを実行して、すべての Cert Manager コンポーネントをインストールします。

```
kubectl apply -f cert-manager/
```

この操作により、`cert-manager` 名前空間、コンポーネント、証明書、および関連するオブジェクトが作成されます。

3 クラスタに Kapp Controller をインストールします。

`kapp-controller.yaml` を使用して Kapp Controller をインストールします。必要に応じて、`kapp-controller-config.yaml` を使用して Kapp Controller の構成をカスタマイズできます。

TKG 拡張機能のホーム ディレクトリに移動します。

```
cd /tkg-extensions-v1.3.1+vmware.1/extensions
```

`ls` を使用して、Kapp Controller の `kapp-controller.yaml` ファイルと `kapp-controller-config.yaml` ファイルが両方あることを確認します。

Kapp Controller のコンテナは、デフォルトの構成パラメータを使用してインストールされます。通常は、構成をカスタマイズしなくても、Kapp Controller をインストールできます。Kapp Controller をカスタマイズする必要がある場合は、`kapp-controller-config.yaml` を編集します。たとえば、プロキシの背後に Kapp Controller をデプロイする場合は、このファイルを編集する必要があります。

必要に応じて、`kapp-controller-config.yaml` ファイルを編集します。構成ファイルを編集する場合は、次のコマンドを使用してファイルを保存し、変更を適用します。

```
kubectl apply -f kapp-controller-config.yaml
```

Kapp Controller コンテナは、`kapp-controller.yaml` を使用してインストールされます。この YAML ファイル内の `spec.containers.image` パスは、パブリック VMware レジストリを参照します。エアギャップ環境の場合は、プライベート レジストリを参照するようにこのパスを更新します。

次のコマンドを実行して、Kapp Controller をインストールします。

```
kubectl apply -f kapp-controller.yaml
```

この操作により、`tkg-system` 名前空間、kapp-controller アプリケーション、およびロール オブジェクトが作成されます。

4 Cert Manager と Kapp Controller のインストールを確認します。

`kubectl get pods -A` コマンドを実行します。それぞれが実行されているのが確認されます。

```
cert-manager       cert-manager-cainjector-...    1/1     Running    0      7h54m
cert-manager       cert-manager-...               1/1     Running    0      7h54m
cert-manager       cert-manager-webhook-...       1/1     Running    0      7h54m
tkg-system         kapp-controller-...            1/1     Running    0      16m
```

TKG 拡張機能のパーシステント ストレージの要件の確認

Prometheus と Grafana の両方の監視拡張機能で、パーシステント ストレージが必要になります。これらの拡張機能のデプロイを準備するには、ターゲット Tanzu Kubernetes クラスタにデフォルトのストレージ クラスが構成されていて、vSphere 名前空間 に十分なストレージがあることを確認します。

TKG 拡張機能のパーシステント ストレージの要件

Prometheus または Grafana 拡張機能をデプロイする Tanzu Kubernetes クラスタは、デフォルトのストレージ クラスを使用してプロビジョニングする必要があります。Tanzu Kubernetes Grid サービス v1alpha1 API を使用した Tanzu Kubernetes クラスタのプロビジョニングの例を参照してください。

拡張機能をデプロイするときに、パーシステント ボリュームの要求を使用するように構成することもできます。この方法で構成する場合は、各拡張機能のデプロイ手順に従って行います。

拡張機能をインストールするクラスタがプロビジョニングされている vSphere 名前空間 のストレージ制限は、パーシステント ボリュームの要求の合計サイズよりも大きくする必要があります。

表 14-2. TKG 拡張機能のデフォルト ストレージの要件

コンポーネント	TKG 拡張機能	デフォルトのストレージ サイズ
Grafana	Grafana	8 Gi
Prometheus サーバ	Prometheus	8 Gi
アラート マネージャ	Prometheus	8 Gi
Harbor	Harbor レジストリ	PVC ごとに異なる

vSphere 名前空間 のストレージ制限の調整

Tanzu Kubernetes クラスタがプロビジョニングされている vSphere 名前空間 のストレージ制限を調整するには、次の手順を実行します。

1 vSphere Client を使用して、vSphere with Tanzu が有効になっている vCenter Server にログインします。

2 ターゲット Tanzu Kubernetes クラスタがプロビジョニングされている vSphere 名前空間 を選択します。

3 [構成] - [リソースの制限] を選択します。

4 [編集] をクリックします。

5 [ストレージ] 制限を調整して、Prometheus および Grafana 拡張機能に必要なパーシステント ボリューム要求の合計サイズよりも大きくします。

TKG 拡張機能における独自の TLS 証明書の使用

TKG 拡張機能には TLS 証明書が必要です。この前提条件を満たすために cert-manager をインストールするか、独自の自己署名証明書を使用できます。CA からの証明書もサポートされます。

TKG 拡張機能における独自の TLS 証明書の使用について

cert-manager のインストールについては、各 TKG 拡張機能のデプロイ プロセスの一環としてドキュメントに記載されています。または、独自の TLS 証明書を使用することもできます。

注： このタスクでは、OpenSSL がインストールされた Linux ホストを使用することを前提にしています。

認証局証明書の生成

本番環境では、CA から証明書を取得する必要があります。開発環境またはテスト環境では、独自の自己署名証明書を生成できます。CA 証明書を生成するには、次の手順を実行します。

1 CA 証明書のプライベート キーを生成します。

```
openssl genrsa -out ca.key 4096
```

2 CA 証明書を生成します。

テンプレートとして、次のコマンドを使用します。環境に基づいて `-subj` オプションの値を更新します。FQDN を使用して TKG 拡張機能ホストを接続する場合は、この FQDN を共通名 (CN) 属性として指定する必要があります。

```
openssl req -x509 -new -nodes -sha512 -days 3650 \
 -subj "/C=US/ST=PA/L=PA/O=example/OU=Personal/CN=tkg-extensions.system.tanzu" \
 -key ca.key \
 -out ca.crt
```

サーバ証明書の生成

証明書には通常、.crt ファイルと .key ファイル（`tls.crt` と `tls.key` など）が含まれます。

1 プライベート キーを生成します。

```
openssl genrsa -out tls.key 4096
```

2 証明書署名リクエスト (CSR) を生成します。

テンプレートとして、次のコマンドを使用します。環境に基づいて `-subj` オプションの値を更新します。FQDN を使用して TKG 拡張機能ホストを接続する場合は、この FQDN を共通名 (CN) 属性として指定し、キーと CSR のファイル名で FQDN を使用する必要があります。

```
openssl req -sha512 -new \
    -subj "/C=US/ST=PA/L=PA/O=example/OU=Personal/CN=tkg-extensions.system.tanzu" \
    -key tls.key \
    -out tls.csr
```

3 x509 v3 拡張ファイルを生成します。

テンプレートとして、次のコマンドを使用します。このファイルを作成して、Subject Alternative Names (SAN) および x509 v3 拡張の要件に準拠する TKG 拡張機能ホストの証明書を生成できるようにします。

```
cat > v3.ext <<-EOF
authorityKeyIdentifier=keyid,issuer
basicConstraints=CA:FALSE
keyUsage = digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment
extendedKeyUsage = serverAuth
subjectAltName = @alt_names

[alt_names]
DNS.1=tkg-extensions.system.tanzu
EOF
```

4 x509 v3 拡張ファイルを使用して、TKG 拡張機能ホストの証明書を生成します。

```
openssl x509 -req -sha512 -days 3650 \
-extfile v3.ext \
-CA ca.crt -CAkey ca.key -CAcreateserial \
-in tls.csr \
-out tls.crt
```

5 次の形式を使用して、ファイル `ca.crt`、`tls.crt`、`tls.key` の内容を `TKG-EXTENSION-data-values.yaml` ファイルにコピーします。

```
ingress:
  tlsCertificate:
    tls.crt: |
        -----BEGIN ...
```

6 ドキュメントに沿って、サポートされている TKG 拡張機能のデプロイを続行します。

Fluent Bit ログ作成用 TKG 拡張機能のデプロイと管理

このトピックでは、Fluent Bit 用 TKG 拡張機能 v1.3.1 をデプロイする方法について説明します。Fluent Bit は高速で軽量なログ プロセッサであり、さまざまなソースからアプリケーション データとログを収集し、それらを統一して複数の宛先に送信できます。Tanzu Kubernetes クラスタ ログを収集して、選択した転送先に転送するには、Fluent Bit の TKG 拡張機能をデプロイします。

拡張機能の前提条件

Fluent Bit 用 TKG 拡張機能 v1.3.1 をデプロイする前に、次の要件を満たす必要があります。

- クラスタをプロビジョニングします。TKGS v1alpha2 API を使用して Tanzu Kubernetes クラスタをプロビジョニングするためのワークフローを参照してください。
- クラスタに接続します。vCenter Single Sign-On ユーザーとして Tanzu Kubernetes クラスタに接続を参照してください。
- kubectl を実行するクライアント ホストへの TKG 拡張機能 v1.3.1 バンドルのダウンロードを実行します。
- ターゲット クラスタへの TKG 拡張機能の前提条件のインストールを実行します。

Fluent Bit 拡張機能のデプロイ

Fluent Bit の TKG 拡張機能は、Fluent Bit コンテナをクラスタにインストールします。このコンテナの詳細については、https://fluentbit.io/を参照してください。

コンテナ	リソースの種類	レプリカ	説明
Fluent Bit	DaemonSet	6	ログ コレクタ、アグリゲータ、フォワーダ

この拡張機能は、https://projects.registry.vmware.com/ にある VMware パブリック レジストリからコンテナをプルするように構成されています。プライベート レジストリを使用する場合は、データ値と拡張機能構成ファイルに含まれるエンドポイント URL を、対応するものに変更します。フィールドとオプションの説明については、Fluent Bit 拡張機能の構成を参照してください。

1 拡張機能のすべての前提条件を満たしていることを確認します。拡張機能の前提条件を参照してください。

2 Fluent Bit 拡張機能のあるディレクトリに移動します。

```
cd /tkg-extensions-v1.3.1+vmware.1/extensions/logging/fluent-bit
```

3 `tanzu-system-logging` 名前空間と、Fluent Bit サービスのアカウントおよびロール オブジェクトを作成します。

```
kubectl apply -f namespace-role.yaml
```

4 Fluent Bit に使用するログの出力先を決定します。サポートされる出力は、Elasticsearch、HTTP、Kafka、Splunk、Syslog などです。詳細については https://docs.fluentbit.io/manual/pipeline/outputs を参照してください。

5 いずれかの `<LOG_BACKEND>/fluent-bit-data-values.example.yaml` をコピーして、選択したログ出力先に Fluent Bit データ値ファイルを作成します。

サポートされているログ出力先ごとに、サンプル データ値ファイルがあります。このサンプルでは、該当するログ出力先の最小構成を示します。

```
cp elasticsearch/fluent-bit-data-values.yaml.example elasticsearch/fluent-bit-data-values.yaml
```

```
cp http/fluent-bit-data-values.yaml.example http/fluent-bit-data-values.yaml
```

```
cp kafka/fluent-bit-data-values.yaml.example kafka/fluent-bit-data-values.yaml
```

```
cp splunk/fluent-bit-data-values.yaml.example splunk/fluent-bit-data-values.yaml
```

```
cp syslog/fluent-bit-data-values.yaml.example syslog/fluent-bit-data-values.yaml
```

6 `<LOG_BACKEND>/fluent-bit-data-values.yaml` にデータを入力して、Fluent Bit 拡張機能を構成します。フィールドとオプションの説明については、Fluent Bit 拡張機能の構成を参照してください。

たとえば、Fluent Bit Syslog 構成には次の値が必要です。

```
logging:
  image:
    repository: projects.registry.vmware.com/tkg # Public registry
tkg:
  instance_name: "<TKG_INSTANCE_NAME>" #mandatory but arbitrary; appears in logs
  cluster_name: "<CLUSTER_NAME>" #name of the target tkgs cluster
fluent_bit:
  output_plugin: "syslog"
```

```
  syslog:
    host: "<SYSLOG_HOST>"
    port: "<SYSLOG_PORT>"
    mode: "<SYSLOG_MODE>"
    format: "<SYSLOG_FORMAT>"
```

Fluent Bit Syslog でポピュレートされるデータ値ファイルの構成は、次のようになります。

```
logging:
  image:
    repository: projects.registry.vmware.com/tkg
tkg:
  instance_name: "tkgs-cluster-1"
  cluster_name: "tkgs-cluster-1"
fluent_bit:
  output_plugin: "syslog"
  syslog:
    host: "10.192.175.59"
    port: "514"
    mode: "tcp"
    format: "rfc5424"
```

7 ログ出力先のデータ値を含む Fluent Bit シークレットを作成します。

```
kubectl create secret generic fluent-bit-data-values --from-file=values.yaml=elasticsearch/
fluent-bit-data-values.yaml -n tanzu-system-logging
```

```
kubectl create secret generic fluent-bit-data-values --from-file=values.yaml=kafka/fluent-
bit-data-values.yaml -n tanzu-system-logging
```

```
kubectl create secret generic fluent-bit-data-values --from-file=values.yaml=splunk/fluent-
bit-data-values.yaml -n tanzu-system-logging
```

```
kubectl create secret generic fluent-bit-data-values --from-file=values.yaml=http/fluent-
bit-data-values.yaml -n tanzu-system-logging
```

```
kubectl create secret generic fluent-bit-data-values --from-file=values.yaml=syslog/fluent-
bit-data-values.yaml -n tanzu-system-logging
```

`secret/fluent-bit-data-values` が `tanzu-system-logging` 名前空間に作成されます。次のコマンドを使用して確認します。

```
kubectl get secrets -n tanzu-system-logging
```

8 Fluent Bit アプリケーションをデプロイします。

```
kubectl apply -f fluent-bit-extension.yaml
```

成功すると、`app.kappctrl.k14s.io/fluent-bit created` が表示されます。

9 Fluent Bit アプリケーションのステータスを確認します。

```
kubectl get app fluent-bit -n tanzu-system-logging
```

成功すると、ステータスが `Reconciling` から `Reconcile succeeded` に変更されます。ステータスが `Reconcile failed` の場合は、Fluent Bit のデプロイのトラブルシューティングを参照してください。

10 アプリケーションの詳細なステータスを確認します。

```
kubectl get app fluent-bit -n tanzu-system-logging -o yaml
```

11 Fluent Bit デーモンセットを確認します。

```
kubectl get daemonsets -n tanzu-system-logging
```

成功すると、次の情報が表示されます。

```
NAME         DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
fluent-bit   6         6         6       6            6           <none>          105s
```

Fluent Bit のデプロイのトラブルシューティング

デプロイまたは調整が失敗した場合は、`kubectl get pods -A` を実行してポッドのステータスを確認します。`fluent-bit` ポッドは、`Running` である必要があります。ポッドのステータスが `ImagePullBackOff` または `ImageCrashLoopBackOff` の場合は、コンテナ イメージをプルできていません。データ値と拡張機能 YAML ファイルに記述されているリポジトリ URL を確認し、これらが正確であることを確認します。

コンテナ ログを確認します。`name-XXXX` は、`kubectl get pods -A` を実行すると表示される一意のポッド名です。

```
kubectl logs pod/fluent-bit-XXXXX -c fluent-bit -n tanzu-system-logging
```

Fluent Bit 拡張機能の更新

Tanzu Kubernetes クラスタにデプロイされている Fluent Bit 拡張機能を更新します。

1 シークレットから Fluent Bit データ値を取得します。

```
kubectl get secret fluent-bit-data-values -n tanzu-system-logging -o 'go-
template={{ index .data "values.yaml" }}' | base64 -d > fluent-bit-data-values.yaml
```

2 `fluent-bit-data-values.yaml` 内の Fluent Bit データ値を更新します。Fluent Bit 拡張機能の構成を参照してください。

3 Fluent Bit データ値のシークレットを更新します。

```
kubectl create secret generic fluent-bit-data-values --from-file=values.yaml=fluent-bit-
data-values.yaml -n tanzu-system-logging -o yaml --dry-run | kubectl replace -f-
```

Fluent Bit 拡張機能が上記のデータ値で再調整されます。

注： デフォルトでは、kapp-controller によってアプリケーションが 5 分ごとに同期されます。更新は 5 分以内に有効になります。更新をすぐに有効にするには、`fluent-bit-extension.yaml` 内の `syncPeriod` の値を小さくし、`kubectl apply -f fluent-bit-extension.yaml` を使用して Fluent Bit 拡張機能を適用します。

4 拡張機能のステータスを確認します。

```
kubectl get app fluent-bit -n tanzu-system-logging
```

5 詳細なステータスを確認して、トラブルシューティングを行います。

```
kubectl get app fluent-bit -n tanzu-system-logging -o yaml
```

6 必要に応じてトラブルシューティングを実行します。Fluent Bit のデプロイのトラブルシューティングを参照してください。

Fluent Bit 拡張機能の削除

Tanzu Kubernetes クラスタから Fluent Bit 拡張機能を削除します。

注： 手順を順番に実行します。Fluent Bit アプリケーションが完全に削除される前に、名前空間、サービス アカウント、ロール オブジェクトを削除しないでください。これを行うと、システム エラーが発生する可能性があります。

1 Fluent Bit 拡張機能のあるディレクトリに移動します。

```
cd extensions/logging/fluent-bit/
```

2 Fluent Bit アプリケーションを削除します。

```
kubectl delete app fluent-bit -n tanzu-system-logging
```

予期される結果：`app.kappctrl.k14s.io "fluent-bit" deleted`。

3 Fluent Bit アプリケーションが削除されていることを確認します。

```
kubectl get app fluent-bit -n tanzu-system-logging
```

予期される結果：`apps.kappctrl.k14s.io "fluent-bit" not found`。

4 `tanzu-system-logging` 名前空間、Fluent Bit 拡張機能サービス アカウント、およびロール オブジェクトを削除します。

```
kubectl delete -f namespace-role.yaml
```

Fluent Bit 拡張機能のアップグレード

既存の Fluent Bit 拡張機能がデプロイされている場合は、最新バージョンにアップグレードできます。

1 Fluent Bit の構成マップをエクスポートします。

```
kubectl get configmap fluent-bit -n tanzu-system-logging -o 'go-template={{ index .data
"fluent-bit.yaml" }}' > fluent-bit-configmap.yaml
```

2 既存の Fluent Bit 環境を削除します。Fluent Bit 拡張機能の削除を参照してください。

3 最新の Fluent Bit 拡張機能をデプロイします。Fluent Bit 拡張機能のデプロイを参照してください。

Fluent Bit 拡張機能の構成

構成値は `extensions/logging/fluent-bit/<LOG_BACKEND>/fluent-bit-data-values.yaml` で設定されています。

表 14-3. Fluent Bit 拡張機能の構成

パラメータ	説明	タイプ	デフォルト
logging.namespace	Fluent Bit がデプロイされる名前空間	文字列	tanzu-system-logging
logging.service_account_name	Fluent Bit サービス アカウントの名前	文字列	fluent-bit
logging.cluster_role_name	fluent bit に get、watch、および list の権限を付与するクラスタロールの名前	文字列	fluent-bit-read
logging.image.name	Fluent Bit イメージの名前	文字列	fluent-bit
logging.image.tag	Fluent Bit イメージ タグ。バージョンをアップグレードしている場合は、この値の更新が必要になることがあります。	文字列	v1.6.9_vmware.1
logging.image.repository	Fluent Bit イメージを含むリポジトリの場所。デフォルトは、パブリック VMware レジストリです。プライベート リポジトリを使用している場合は（エアギャップ環境など）、この値を変更します。	文字列	projects.registry.vmware.com/tkg
logging.image.pullPolicy	Fluent bit イメージ プル ポリシー	文字列	IfNotPresent
logging.update_strategy	DaemonSet の更新時に使用される更新方法	文字列	RollingUpdate
tkg.cluster_name	Tanzu Kubernetes クラスタの名前	文字列	Null（必須パラメータ）

表 14-3. Fluent Bit 拡張機能の構成 （続き）

パラメータ	説明	タイプ	デフォルト
tkg.instance_name	スーパーバイザー クラスタ、および 1 つのデプロイ内のすべての Tanzu Kubernetes クラスタで共有される TKG インスタンスのユーザー定義による名前。インストールに関連する任意の名前を使用できます。	文字列	Null（必須パラメータ） **注：** このフィールドは必須ですが、任意に指定できます。ログに表示される名前です。
fluent_bit.log_level	Fluent Bit に使用するログ レベル	文字列	info
fluent_bit.output_plugin	Fluent-Bit が収集する情報をフラッシュするバックエンドを設定します	文字列	Null（必須パラメータ）
fluent_bit.elasticsearch.host	ターゲット Elasticsearch インスタンスの IP アドレスまたはホスト名	文字列	Null（output_plugin が弾性検索の場合の必須パラメータ）
fluent_bit.elasticsearch.port	ターゲット Elasticsearch インスタンスの TCP ポート	整数型	Null（output_plugin が弾性検索の場合の必須パラメータ）
fluent_bit.elasticsearch.buffer_size	Elasticsearch サービスからの応答の読み取りに使用するバッファ サイズを指定します。False の場合は無制限に設定されます	文字列	False
fluent_bit.elasticsearch.tls	Elasticsearch の TLS のデフォルト設定を指定します	文字列	オフ
fluent_bit.kafka.broker_service_name	Kafka Broker の単一または複数リスト（例：192.168.1.3:9092）	文字列	Null（output_plugin が kafka の場合の必須パラメータ）
fluent_bit.kafka.topic_name	Fluent Bit が Kafka へのメッセージ送信に使用する単一のエントリまたはトピックのカンマ区切りリスト	文字列	Null（output_plugin が kafka の場合の必須パラメータ）
fluent_bit.splunk.host	ターゲット Splunk サーバの IP アドレスまたはホスト名	文字列	Null（output_plugin が splunk の場合の必須パラメータ）
fluent_bit.splunk.port	ターゲット Splunk サーバの TCP ポート	整数型	Null（output_plugin が splunk の場合の必須パラメータ）
fluent_bit.splunk.token	HTTP イベント コレクタ インターフェイスの認証トークンを指定します	文字列	Null（output_plugin が splunk の場合の必須パラメータ）
fluent_bit.http.host	ターゲット HTTP サーバの IP アドレスまたはホスト名	文字列	Null（output_plugin が http の場合の必須パラメータ）
fluent_bit.http.port	ターゲット HTTP サーバの TCP ポート	整数型	Null（output_plugin が http の場合の必須パラメータ）

表 14-3. Fluent Bit 拡張機能の構成 （続き）

パラメータ	説明	タイプ	デフォルト
fluent_bit.http.mode	ターゲット Web サーバの HTTP URI を指定します	文字列	Null（output_plugin が http の場合の必須パラメータ）
fluent_bit.http.header_key_value	HTTP ヘッダー キー/値ペア。複数のヘッダーを設定できます	文字列	Null（output_plugin が http の場合の必須パラメータ）
fluent_bit.http.format	HTTP 要求本文で使用されるデータ形式を指定します	文字列	Null（output_plugin が http の場合の必須パラメータ）
fluent_bit.syslog.host	リモート Syslog サーバのドメインまたは IP アドレス	文字列	Null（output_plugin が syslog の場合の必須パラメータ）
fluent_bit.syslog.port	リモート Syslog サーバの TCP または UDP ポート	整数型	Null（output_plugin が syslog の場合の必須パラメータ）
fluent_bit.syslog.mode	TCP、UDP、TLS からの転送タイプを指定します	文字列	Null（output_plugin が syslog の場合の必須パラメータ）
fluent_bit.syslog.format	HTTP 要求本文で使用されるデータ形式を指定します	文字列	Null（output_plugin が syslog の場合の必須パラメータ）
host_path.volume_1	ホスト ノードのファイル システムからポッドへのディレクトリ パス（ボリューム 1 の場合）	文字列	/var/log
host_path.volume_2	ホスト ノードのファイル システムからポッドへのディレクトリ パス（ボリューム 2 の場合）	文字列	/var/lib/docker/containers
host_path.volume_3	ホスト ノードのファイル システムからポッドへのディレクトリ パス（ボリューム 3 の場合）	文字列	/run/log
systemd.path	Systemd ジャーナル ディレクトリへのパス	文字列	/var/log/journal

Contour Ingress 用 TKG 拡張機能のデプロイと管理

このトピックでは、Contour Ingress 用 TKG 拡張機能 v1.3.1 をデプロイする方法について説明します。Contour は、Envoy リバース プロキシを使用する Kubernetes Ingress コントローラです。Tanzu Kubernetes クラスタで実行されているサービスにルートを公開するには、Contour Ingress 用 TKG 拡張機能をデプロイします。

拡張機能の前提条件

Contour Ingress 用 TKG 拡張機能 v1.3.1 をデプロイするには、次の要件を満たす必要があります。

- クラスタをプロビジョニングします。TKGS v1alpha2 API を使用して Tanzu Kubernetes クラスタをプロビジョニングするためのワークフローを参照してください。
- クラスタに接続します。vCenter Single Sign-On ユーザーとして Tanzu Kubernetes クラスタに接続を参照してください。
- kubectl を実行するクライアント ホストへの TKG 拡張機能 v1.3.1 バンドルのダウンロードを実行します。
- ターゲット クラスタへの TKG 拡張機能の前提条件のインストールを実行します。

Contour 拡張機能のデプロイ

Contour Ingress 用 TKG 拡張機能は、クラスタに Envoy と Contour の 2 つのコンテナをインストールします。詳細については、『https://projectcontour.io/』を参照してください。

コンテナ	リソースの種類	レプリカ	説明
Envoy	DaemonSet	3	高パフォーマンスのリバース プロキシ
Contour	デプロイ	2	Envoy 用の管理および構成サーバ

この拡張機能は、https://projects.registry.vmware.com/ にある VMware パブリック レジストリからコンテナをプルするように構成されています。プライベート レジストリを使用する場合は、データ値と拡張機能構成に含まれるエンドポイント URL を、対応するものに変更します。Contour 拡張機能の構成を参照してください。

1 拡張機能のすべての前提条件を満たしていることを確認します。拡張機能の前提条件を参照してください。

2 Contour の拡張機能ファイルをダウンロードしたディレクトリに移動します。

```
cd /tkg-extensions-v1.3.1+vmware.1/extensions/ingress/contour
```

3 次のコマンドを実行して、`tanzu-system-ingress` 名前空間と、Contour サービスのアカウントおよびロール オブジェクトを作成します。

```
kubectl apply -f namespace-role.yaml
```

4 vSphere 用の Contour データ値ファイルを作成します。

```
cp vsphere/contour-data-values-lb.yaml.example vsphere/contour-data-values.yaml
```

5 ファイル `vsphere/contour-data-values.yaml` を更新して Contour を構成します。

サンプル データ値ファイルには、最低限必要な構成が含まれています。すべての構成フィールドとオプションの説明については、Contour 拡張機能の構成を参照してください。

たとえば、次に示す vSphere 用の Contour 構成には、LoadBalancer タイプのサービスが使用されます。

```
infrastructure_provider: "vsphere"
contour:
  image:
    repository: projects.registry.vmware.com/tkg
envoy:
  image:
    repository: projects.registry.vmware.com/tkg
    tag: v1.17.3_vmware.1
  service:
    type: "LoadBalancer"
```

注： CVE を含む Envoy イメージ バージョン `v1.16.2_vmware.1` を使用しないようにするために、Envoy イメージ バージョン `v1.17.3_vmware.1` を指定することをお勧めします。詳細については、リリース ノートを参照してください。

6 データ値を使用してシークレットを作成します。

```
kubectl create secret generic contour-data-values --from-file=values.yaml=vsphere/contour-data-values.yaml -n tanzu-system-ingress
```

`secret/contour-data-values` が `tanzu-system-ingress` 名前空間に作成されます。次のコマンドを使用して確認します。

```
kubectl get secrets -n tanzu-system-ingress
```

7 Contour Ingress コントローラ アプリケーションをデプロイします。

```
kubectl apply -f contour-extension.yaml
```

成功すると、`app.kappctrl.k14s.io/contour created` が表示されます。

8 Contour Ingress コントローラ アプリケーションのステータスを確認します。

```
kubectl get app contour -n tanzu-system-ingress
```

成功すると、ステータスが `Reconciling` から `Reconcile succeeded` に変わります。ステータスが `Reconcile failed` の場合は、Contour Ingress のデプロイのトラブルシューティングを参照してください。

9 Contour Ingress コントローラ アプリケーションの詳細情報を表示します。

```
kubectl get app contour -n tanzu-system-ingress -o yaml
```

10 LoadBalancer タイプの Envoy サービスを表示します。

```
kubectl get service envoy -n tanzu-system-ingress -o wide
```

成功すると、Envoy LoadBalancer の詳細が表示されます。

```
NAME    TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)                      AGE
SELECTOR
envoy   LoadBalancer   10.79.65.110   10.178.147.73   80:30437/TCP,443:30589/TCP   2m42s
app=envoy,kapp.k14s.io/app=1629916985840017976
```

11 Envoy DaemonSet を確認します。

```
kubectl get daemonsets -n tanzu-system-ingress
```

成功すると、3 ポッドの Envoy DaemonSet が表示されます。

```
NAME    DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
envoy   3         3         3       3            3           <none>          6m10s
```

12 Contour のデプロイを確認します。

```
kubectl get deployments -n tanzu-system-ingress
```

成功すると、2 ポッドの Contour のデプロイが表示されます。

```
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
contour   2/2     2            2           8m7s
```

13 Contour Ingress コントローラが正しくインストールされ、使用可能な状態であることを確認します。

```
kubectl get pod,svc -n tanzu-system-ingress
```

Contour ポッドと Envoy ポッドのステータスは `Running` である必要があり、Envoy サービスの LoadBalancer には `EXTERNAL-IP` が割り当てられます。

```
NAME                            READY   STATUS    RESTARTS   AGE
pod/contour-84bb5475cf-7h4cx    1/1     Running   0          9m52s
pod/contour-84bb5475cf-v8k9r    1/1     Running   0          9m52s
pod/envoy-4828j                 2/2     Running   0          9m52s
pod/envoy-c54dw                 2/2     Running   0          9m52s
pod/envoy-qpjqp                 2/2     Running   0          9m52s

NAME              TYPE           CLUSTER-IP     EXTERNAL-IP
PORT(S)                      AGE
service/contour   ClusterIP      10.105.6.207   <none>
8001/TCP                     9m52s
service/envoy     LoadBalancer   10.79.65.110   10.178.147.73   80:30437/
TCP,443:30589/TCP   9m52s
```

Contour Ingress のデプロイのトラブルシューティング

デプロイまたは調整が失敗した場合は、`kubectl get pods -n tanzu-system-ingress` を実行してポッドのステータスを確認します。`contour` および `envoy` ポッドは、`Running` である必要があります。ポッドのステータスが `ImagePullBackOff` または `ImageCrashLoopBackOff` の場合は、コンテナ イメージをプルできていません。データ値と拡張機能 YAML ファイルに記述されているリポジトリ URL を確認し、これらが正確であることを確認します。

コンテナログを確認します。`name-XXXX` は、`kubectl get pods -A` を実行したときの一意のポッド名です。

```
kubectl logs pod/envoy-XXXXX -c envoy -n tanzu-system-ingress
```

```
kubectl logs pod/contour-XXXXX -c contour -n tanzu-system-ingress
```

Contour ポッドが `ContainerCreating` 状態で停止し、上記のいずれかのイメージ エラーでは失敗しておらず、処理が進行していない（「contour-xxxxx has timed out progressing」）場合は、IP アドレスの競合が発生している可能性があります。ワークロード ネットワークを構成したときに指定したノード CIDR 範囲が、クラスタ仕様のポッド CIDR 範囲（デフォルトでは `192.168.0.0/16`）と競合していないことを確認します。競合がある場合は、別のポッド サブネットを使用してクラスタを更新するか、ノード ネットワークを変更します。

Contour 拡張機能の更新

Tanzu Kubernetes クラスタにデプロイされている Contour 拡張機能を更新します。

1 シークレットから Contour データ値を取得します。

```
kubectl get secret contour-data-values -n tanzu-system-ingress -o 'go-
template={{ index .data "values.yaml" }}' | base64 -d > contour-data-values.yaml
```

2 `ingress/contour/values.yaml` 内の Contour Ingress データ値を更新します。Contour 拡張機能の構成を参照してください。

たとえば、次に示す vSphere 用の Contour 構成には、LoadBalancer タイプのサービスが使用されます。

```
infrastructure_provider: "vsphere"
contour:
  image:
    repository: projects.registry.vmware.com/tkg
envoy:
  image:
    repository: projects.registry.vmware.com/tkg
    tag: v1.17.3_vmware.1
  service:
    type: "LoadBalancer"
```

注： CVE を含む Envoy イメージ バージョン `v1.16.2_vmware.1` を使用しないようにするために、Envoy イメージ バージョン `v1.17.3_vmware.1` を指定することをお勧めします。詳細については、リリース ノートを参照してください。

3 Contour データ値のシークレットを更新します。

```
kubectl create secret generic contour-data-values --from-file=values.yaml=contour-data-
values.yaml -n tanzu-system-ingress -o yaml --dry-run | kubectl replace -f-
```

Contour 拡張機能が新しいデータ値で調整されます。

注： デフォルトでは、kapp-controller によってアプリケーションが 5 分ごとに同期されます。更新は 5 分以内に有効になります。更新をすぐに有効にするには、`contour-extension.yaml` 内の `syncPeriod` の値を小さくし、`kubectl apply -f contour-extension.yaml` を使用して Grafana 拡張機能を再デプロイします。

4 アプリケーションのステータスを確認します。

```
kubectl get app contour -n tanzu-system-ingress
```

Contour が更新されると、ステータスが `Reconcile Succeeded` に変更されます。

5 詳細なステータスを確認します。

```
kubectl get app contour -n tanzu-system-ingress -o yaml
```

6 必要に応じてトラブルシューティングを実行します。Contour Ingress のデプロイのトラブルシューティングを参照してください。

Contour 拡張機能の削除

Tanzu Kubernetes クラスタから Contour 拡張機能を削除します。

注： 手順を順番に実行します。Contour Ingress コントローラ アプリケーションが完全に削除される前に、名前空間、サービス アカウント、ロール オブジェクトを削除しないでください。これを行うと、システム エラーが発生する可能性があります。

1 Contour 拡張機能のディレクトリに移動します。

```
cd extensions/ingress/contour/
```

2 Contour Ingress コントローラ アプリケーションを削除します。

```
kubectl delete app contour -n tanzu-system-ingress
```

予期される結果：`app.kappctrl.k14s.io "contour" deleted`。

3 Contour Ingress アプリケーションが削除されていることを確認します。

```
kubectl get app contour -n tanzu-system-ingress
```

予期される結果：`apps.kappctrl.k14s.io "contour" not found`。

4 `tanzu-system-ingress` 名前空間と、Contour 拡張機能サービスのアカウントおよびロール オブジェクトを削除します。

```
kubectl delete -f namespace-role.yaml
```

Contour 拡張機能のアップグレード

既存の Contour 拡張機能がデプロイされている場合は、最新バージョンにアップグレードできます。

1 Contour の構成マップをエクスポートし、バックアップとして保存します。

```
kubectl get configmap contour -n tanzu-system-ingress -o 'go-template={{ index .data
"contour.yaml" }}' > contour-configmap.yaml
```

2 既存の Contour 環境を削除します。Contour 拡張機能の削除を参照してください。

3 最新の Contour 拡張機能をデプロイします。Contour 拡張機能のデプロイを参照してください。

Contour 拡張機能の構成

Contour Ingress コントローラの構成値は、`/extensions/ingress/contour/vsphere/contour-data-values.yaml` で設定されます。

表 14-4. Contour Ingress の構成パラメータ

パラメータ	説明	タイプ	デフォルト
infrastructure_provider	インフラストラクチャ プロバイダ。サポートされている値：vsphere、aws、azure	文字列	必須パラメータ
contour.namespace	contour がデプロイされる名前空間	文字列	tanzu-system-ingress
contour.config.requestTimeout	Envoy に渡されるクライアント要求タイムアウト	time.Duration	0s ファイル ダウンロードのルート タイムアウトを参照してください。
contour.config.server.xdsServerType	使用する XDS サーバ タイプ：サポートされている値：contour または envoy	文字列	Null
contour.config.tls.minimumProtocolVersion	Contour がネゴシエートする最小 TLS バージョン	文字列	1.1
contour.config.tls.fallbackCertificate.name	vhost に定義されている SNI と一致しない要求に対するフォールバック証明書が含まれているシークレットの名前	文字列	Null
contour.config.tls.fallbackCertificate.namespace	フォールバック証明書が含まれているシークレットの名前空間	文字列	Null
contour.config.tls.envoyClientCertificate.name	クライアント証明書として使用するシークレットの名前、バックエンド サービスへの TLS 接続用プライベート キー	文字列	Null
contour.config.tls.envoyClientCertificate.namespace	クライアント証明書として使用するシークレットの名前空間、バックエンド サービスへの TLS 接続用プライベート キー	文字列	Null
contour.config.leaderelection.configmapName	contour leaderelection に使用される configmap の名前	文字列	leader-elect
contour.config.leaderelection.configmapNamespace	contour leaderelection configmap の名前空間	文字列	tanzu-system-ingress
contour.config.disablePermitInsecure	ingressroute permitInsecure フィールドを無効にします	boolean	false
contour.config.accesslogFormat	アクセス ログ形式	文字列	envoy
contour.config.jsonFields	ログに記録されるフィールド	文字列の配列	https://godoc.org/github.com/projectcontour/contour/internal/envoy#JSONFields
contour.config.useProxyProtocol	https://projectcontour.io/guides/proxy-proto/	boolean	false

表 14-4. Contour Ingress の構成パラメータ （続き）

パラメータ	説明	タイプ	デフォルト
contour.config.defaultHTTPVersions	Envoy がサービスを提供するためにプログラムする必要がある HTTP バージョン	文字列の配列	"HTTP/1.1 HTTP2"
contour.config.timeouts.requestTimeout	要求全体のタイムアウト	time.Duration	Null（タイムアウトが無効）
contour.config.timeouts.connectionIdleTimeout	アイドル接続を終了するまでの待機時間	time.Duration	60s
contour.config.timeouts.streamIdleTimeout	アクティビティがない要求またはストリームを終了するまでの待機時間	time.Duration	5m
contour.config.timeouts.maxConnectionDuration	アクティビティに関係なく接続を終了するまでの待機時間	time.Duration	Null（タイムアウトが無効）
contour.config.timeouts.ConnectionShutdownGracePeriod	最初と最後の GOAWAY を送信するまでの待機時間	time.Duration	5s
contour.config.cluster.dnsLookupFamily	HTTPProxy ルートから externalName タイプ サービスへのアップストリーム要求に使用する dns-lookup-family	文字列	Null（サポートされている値：auto、v4、v6）
contour.config.debug	デバッグ機能をオンにします	boolean	false
contour.config.ingressStatusAddress	すべての Ingress リソースのステータスで設定するアドレス	文字列	Null
contour.certificate.duration	contour 証明書の有効期間	time.Duration	8760h
contour.certificate.renewBefore	contour 証明書の更新が必要になるまでの期間	time.Duration	360h
contour.deployment.replicas	contour レプリカの数	整数型	2
contour.image.repository	Contour イメージを含むリポジトリの場所。デフォルトは、パブリック VMware レジストリです。プライベート リポジトリを使用している場合は（エアギャップ環境など）、この値を変更します。	文字列	projects.registry.vmware.com/tkg
contour.image.name	contour イメージの名前	文字列	contour
contour.image.tag	Contour イメージ タグ。Contour バージョンをアップグレードしている場合は、この値の更新が必要になることがあります。	文字列	v1.11.0_vmware.1
contour.image.pullPolicy	Contour イメージ プル ポリシー	文字列	IfNotPresent

表 14-4. Contour Ingress の構成パラメータ （続き）

パラメータ	説明	タイプ	デフォルト
envoy.image.repository	Envoy イメージを含むリポジトリの場所。デフォルトは、パブリック VMware レジストリです。プライベート リポジトリを使用している場合は（エアギャップ環境など）、この値を変更します。	文字列	projects.registry.vmware.com/tkg
envoy.image.name	envoy イメージの名前	文字列	envoy
envoy.image.tag	イメージ タグを使用します。Envoy バージョンをアップグレードしている場合は、この値の更新が必要になることがあります。	文字列	`v1.17.3_vmware.1` **注：** CVE が含まれているため、Envoy イメージ `v1.16.2_vmware.1` を使用しないでください。詳細については、リリース ノートを参照してください。
envoy.image.pullPolicy	Envoy イメージ プル ポリシー	文字列	IfNotPresent
envoy.hostPort.enable	ホスト上の envoy ポートを公開するフラグ	boolean	true
envoy.hostPort.http	Envoy HTTP ホスト ポート	整数型	80
envoy.hostPort.https	Envoy HTTPS ホスト ポート	整数型	443
envoy.service.type	envoy を公開するサービスのタイプ。サポートされている値：ClusterIP、NodePort、LoadBalancer	文字列	vSphere の必須パラメータ：NodePort または LoadBalancer、AWS：LoadBalancer、Azure：LoadBalancer
envoy.service.annotations	Envoy サービスの注釈	マップ（キー値）	空のマップ
envoy.service.externalTrafficPolicy	envoy サービスの外部トラフィック ポリシー。サポートされている値：Local、Cluster	文字列	Cluster
envoy.service.nodePort.http	http 要求に使用されるタイプ NodePort のサービスに必要な nodePort	整数型	Null - Kubernetes は動的ノード ポートを割り当てます
envoy.service.nodePort.https	HTTPS 要求に使用されるタイプ NodePort のサービスに必要な nodePort	整数型	Null - Kubernetes は動的ノード ポートを割り当てます
envoy.deployment.hostNetwork	hostNetwork で envoy を実行します	boolean	false
envoy.service.aws.LBType	envoy サービスの公開に使用される AWS LB タイプ。サポートされている値：classic、nlb	文字列	classic
envoy.loglevel	envoy に使用するログ レベル	文字列	info

ファイル ダウンロードのルート タイムアウト

パラメータ `contour.config.requestTimeout` は、Contour ルート タイムアウト期間を定義します。デフォルト値は `0s` です。Contour をファイル転送に使用している場合は、この値の調整が必要になる場合があります。

Contour のドキュメントによると、タイムアウト値が `0s` の場合、Contour は Envoy タイムアウトを使用するように設定されます。Envoy のドキュメントによると、Envoy のデフォルトのタイムアウトは 15 秒です。また、Envoy では、要求から応答までの操作全体がタイムアウト間隔内で完了することが想定されます。

つまり、デフォルトの Contour タイムアウト設定が `0s` の場合、ファイル転送は 15 秒以内に完了する必要があります。これは、サイズが大きいファイルの転送では時間が足りない可能性があります。デフォルトの Envoy のタイムアウトを無効にするには、`contour.config.requestTimeout` の値を `0` に設定します。

Prometheus Monitoring 用 TKG 拡張機能のデプロイと管理

このトピックでは、Prometheus 用 TKG 拡張機能 v1.3.1 をデプロイする方法について説明します。Prometheus は、システムおよびサービス監視システムです。構成されたターゲットから指定間隔でメトリックを収集し、ルール式を評価し、結果を表示します。何らかの条件が満たされている場合は、アラートをトリガすることがあります。Alertmanager は Prometheus によって生成されたアラートを処理して、受信側のエンドポイントにルーティングします。Tanzu Kubernetes クラスタのメトリックを収集して表示するには、Prometheus 用 TKG 拡張機能をデプロイします。

拡張機能の前提条件

クラスタ監視のための Alertmanager とともに Prometheus 用 TKG 拡張機能 v1.3.1 をデプロイする前に、次の要件を満たす必要があります。

- クラスタをプロビジョニングします。TKGS v1alpha2 API を使用して Tanzu Kubernetes クラスタをプロビジョニングするためのワークフローを参照してください。

  注： Prometheus 拡張機能をインストールするには、デフォルトの serviceDomain (`cluster.local`) を使用するクラスタをデプロイする必要があります。

- クラスタに接続します。vCenter Single Sign-On ユーザーとして Tanzu Kubernetes クラスタに接続を参照してください。
- kubectl を実行するクライアント ホストへの TKG 拡張機能 v1.3.1 バンドルのダウンロードを実行します。
- ターゲット Tanzu Kubernetes クラスタに TKG 拡張機能の前提条件のインストールします。

Prometheus の監視には、一般的な要件に加えてデフォルトのパーシステント ストレージ クラスが必要です。デフォルトのパーシステント ストレージ クラスを使用してクラスタを作成するか、拡張機能をデプロイするときに Prometheus 構成ファイルでクラスタを指定することができます。TKG 拡張機能のパーシステント ストレージの要件の確認を参照してください。

Prometheus 拡張機能のデプロイ

Prometheus 用 TKG 拡張機能は、いくつかのコンテナをインストールします。詳細については、『https://prometheus.io/』を参照してください。

コンテナ	リソースの種類	レプリカ	説明
prometheus-alertmanager	デプロイ	1	Prometheus サーバなどのクライアント アプリケーションによって送信されたアラートを処理します。
prometheus-cadvisor	DaemonSet	5	実行中のコンテナのリソース使用量とパフォーマンス データを分析して公開します。
prometheus-kube-state-metrics	デプロイ	1	ノードのステータスとキャパシティ、レプリカセットのコンプライアンス、ポッド、ジョブ、cron ジョブのステータス、リソース要求、制限を監視します。
prometheus-node-exporter	DaemonSet	5	カーネルによって公開されるハードウェアおよび OS メトリックのエクスポータ。
prometheus-pushgateway	デプロイ	1	スクレイピングできないジョブからメトリックをプッシュできるサービス。
prometheus-server	デプロイ	1	スクレイピング、ルール処理、アラートなどの中核的な機能を提供します。

この拡張機能は、https://projects.registry.vmware.com/ にある VMware パブリック レジストリからコンテナをプルするように構成されています。プライベート レジストリを使用する場合は、データ値と拡張機能構成に含まれるエンドポイント URL を、対応するものに変更します。Prometheus 拡張機能の構成を参照してください。

1 拡張機能のすべての前提条件を満たしていることを確認します。拡張機能の前提条件を参照してください。

2 Prometheus 拡張機能のディレクトリに移動します。

```
cd /tkg-extensions-v1.3.1+vmware.1/extensions/monitoring/prometheus
```

3 tanzu-system-monitoring 名前空間、Prometheus サービス アカウント、およびロール オブジェクトを作成します。

```
kubectl apply -f namespace-role.yaml
```

4 Prometheus データ値ファイルを作成します。

サンプル データ値ファイルには、最小構成が含まれています。

```
cp prometheus-data-values.yaml.example prometheus-data-values.yaml
```

5 prometheus-data-values.yaml を更新して Prometheus 拡張機能を構成します。フィールドとオプションの説明については、Prometheus 拡張機能の構成を参照してください。

クラスタがデフォルトのパーシステント ストレージ クラスでプロビジョニングされていない場合は、データ値ファイル内で指定できます。また、名前空間にパーシステント ボリュームの要求を格納するのに必要なストレージが確保されている必要があります。

```
monitoring:
  prometheus_server:
    image:
      repository: projects.registry.vmware.com/tkg/prometheus
    pvc:
      storage_class: vwt-storage-policy
      storage: "8Gi"
```

```
  alertmanager:
    image:
      repository: projects.registry.vmware.com/tkg/prometheus
    pvc:
      storage_class: vwt-storage-policy
      storage: "8Gi"
...
```

6 prometheus-data-values ファイルを使用して Prometheus シークレットを作成します。

```
kubectl create secret generic prometheus-data-values --from-file=values.yaml=prometheus-data-values.yaml -n tanzu-system-monitoring
```

prometheus-data-values シークレットが tanzu-system-monitoring 名前空間に作成されている。kubectl get secrets -n tanzu-system-monitoring を使用して確認します。

7 Prometheus 拡張機能をデプロイします。

```
kubectl apply -f prometheus-extension.yaml
```

成功すると、Prometheus アプリケーションが作成されます：app.kappctrl.k14s.io/prometheus created。

8 Prometheus アプリケーションのステータスを確認します。

```
kubectl get app prometheus -n tanzu-system-monitoring
```

ステータスが Reconciling から Reconcile succeeded に変わります。ステータスが Reconcile failed の場合は、トラブルシューティングを参照してください。

9 Prometheus アプリケーションの詳細情報を確認します。

```
kubectl get app prometheus -n tanzu-system-monitoring -o yaml
```

10 Prometheus DaemonSets を確認します。

```
kubectl get daemonsets -n tanzu-system-monitoring
```

11 Prometheus のデプロイを確認します。

```
kubectl get deployments -n tanzu-system-monitoring
```

Prometheus のデプロイのトラブルシューティング

デプロイまたは調整が失敗した場合は、kubectl get pods -A を実行してポッドのステータスを確認します。通常の状態では、ポッドは Running です。ステータスが ImagePullBackOff または ImageCrashLoopBackOff の場合、コンテナ イメージをリポジトリからプルすることができていません。データ値と拡張機能 YAML ファイルに記述されている URL を確認し、これらが正確であることを確認します。

コンテナログを確認します。name-XXXX は、kubectl get pods -A を実行したときの一意のポッド名です。

```
kubectl logs pod/prometheus-alertmanager-XXXXX -c prometheus-alertmanager -n tanzu-system-monitoring
```

```
kubectl logs pod/prometheus-server-XXXXX -c prometheus-server -n tanzu-system-monitoring
```

Prometheus 拡張機能の更新

Tanzu Kubernetes クラスタにデプロイされている Prometheus 拡張機能の構成を更新します。

1 シークレットから Prometheus データ値を取得します。

```
kubectl get secret prometheus-data-values -n tanzu-system-monitoring -o 'go-template={{ index .data "values.yaml" }}' | base64 -d > prometheus-data-values.yaml
```

2 Prometheus データ値のシークレットを更新します。

```
kubectl create secret generic prometheus-data-values --from-file=values.yaml=prometheus-data-values.yaml -n tanzu-system-monitoring -o yaml --dry-run | kubectl replace -f-
```

Prometheus 拡張機能が更新されたデータ値で調整されます。

注： デフォルトでは、kapp-controller によってアプリケーションが 5 分ごとに同期されます。更新は 5 分以内に有効になります。更新をすぐに有効にするには、prometheus-extension.yaml 内の syncPeriod の値を小さくし、kubectl apply -f prometheus-extension.yaml を使用して Fluent Bit 拡張機能を適用します。

3 拡張機能のステータスを確認します。

```
kubectl get app prometheus -n tanzu-system-monitoring
```

Prometheus が更新されると、ステータスが Reconcile Succeeded に変更されます。

4 詳細なステータスを確認して、トラブルシューティングを行います。

```
kubectl get app prometheus -n tanzu-system-monitoring -o yaml
```

Prometheus 拡張機能の削除

Tanzu Kubernetes クラスタから Prometheus 拡張機能を削除します。

注： 手順を順番に実行します。Prometheus アプリケーションが完全に削除される前に、名前空間、サービス アカウント、ロール オブジェクトを削除しないでください。これを行うと、システム エラーが発生する可能性があります。

注意： Prometheus と Grafana は同じ名前空間を使用します。名前空間を削除すると、そこにデプロイされている拡張機能があった場合、その拡張機能が損なわれます。Grafana がデプロイされている場合は、Grafana を削除する前に名前空間を削除しないでください。

1 Prometheus 拡張機能のディレクトリに移動します。

```
cd /extensions/monitoring/prometheus/
```

2 Prometheus アプリケーションを削除します。

```
kubectl delete app prometheus -n tanzu-system-monitoring
```

予期される結果：`app.kappctrl.k14s.io "prometheus" deleted`。

3 Prometheus アプリケーションが削除されていることを確認します。

```
kubectl get app prometheus -n tanzu-system-monitoring
```

予期される結果：`apps.kappctrl.k14s.io "prometheus" not found`。

4 `tanzu-system-monitoring` 名前空間と、Prometheus サービスのアカウントおよびロール オブジェクトを削除します。

注意: Grafana がデプロイされている場合は、この手順を実行しません。

```
kubectl delete -f namespace-role.yaml
```

5 Prometheus を再デプロイする場合は、シークレット `prometheus-data-values` を削除します。

```
kubectl delete secret prometheus-data-values -n tanzu-system-monitoring
```

予期される結果：`secret "prometheus-data-values" deleted`。

Prometheus 拡張機能のアップグレード

既存の Prometheus 拡張機能がデプロイされている場合は、最新バージョンにアップグレードできます。

1 Prometheus の構成マップをエクスポートし、バックアップとして保存します。

```
kubectl get configmap prometheus -n tanzu-system-monitoring -o 'go-template={{ index .data
"prometheus.yaml" }}' > prometheus-configmap.yaml
```

2 既存の Prometheus 環境を削除します。Prometheus 拡張機能の削除を参照してください。

3　Prometheus 拡張機能をデプロイします。Prometheus 拡張機能のデプロイを参照してください。

Prometheus 拡張機能の構成

Prometheus の構成は、`/extensions/monitoring/prometheus/prometheus-data-values.yaml` で設定されます。

表 14-5. Prometheus 構成パラメータ

パラメータ	説明	タイプ	デフォルト
monitoring.namespace	Prometheus がデプロイされる名前空間	文字列	tanzu-system-monitoring
monitoring.create_namespace	フラグは、monitoring.namespace で指定された名前空間を作成するかどうかを示します	boolean	false
monitoring.prometheus_server.config.prometheus_yaml	Prometheus に渡される Kubernetes クラスタ監視構成の詳細	yaml ファイル	prometheus.yaml
monitoring.prometheus_server.config.alerting_rules_yaml	Prometheus で定義された詳細なアラート ルール	yaml ファイル	alerting_rules.yaml
monitoring.prometheus_server.config.recording_rules_yaml	Prometheus で定義された詳細なレコード ルール	yaml ファイル	recording_rules.yaml
monitoring.prometheus_server.service.type	Prometheus を公開するサービスのタイプ。サポートされている値：ClusterIP	文字列	ClusterIP
monitoring.prometheus_server.enable_alerts.kubernetes_api	Prometheus で Kubernetes API の SLO アラートを有効にする	boolean	true
monitoring.prometheus_server.sc.aws_type	AWS 上の storageclass に定義された AWS タイプ	文字列	gp2
monitoring.prometheus_server.sc.aws_fsType	AWS 上の storageclass に定義された AWS ファイル システム タイプ	文字列	ext4
monitoring.prometheus_server.sc.allowVolumeExpansion	ボリュームの拡張が AWS 上の storageclass に許可されているかどうかを定義します	boolean	true
monitoring.prometheus_server.pvc.annotations	ストレージ クラスの注釈	マップ	{}
monitoring.prometheus_server.pvc.storage_class	パーシステント ボリュームの要求に使用するストレージ クラス。これはデフォルトで null で、デフォルトのプロビジョナが使用されます	文字列	null

表 14-5. Prometheus 構成パラメータ （続き）

パラメータ	説明	タイプ	デフォルト
monitoring.prometheus_server.pvc.accessMode	パーシステント ボリューム要求に対しアクセス モードを定義します。サポートされている値：ReadWriteOnce、ReadOnlyMany、ReadWriteMany	文字列	ReadWriteOnce
monitoring.prometheus_server.pvc.storage	パーシステント ボリューム要求に対しストレージ サイズを定義します	文字列	8Gi
monitoring.prometheus_server.deployment.replicas	prometheus レプリカの数	整数型	1
monitoring.prometheus_server.image.repository	Prometheus イメージを含むリポジトリの場所。デフォルトは、パブリック VMware レジストリです。プライベート リポジトリを使用している場合は（エアギャップ環境など）、この値を変更します。	文字列	projects.registry.vmware.com/tkg/prometheus
monitoring.prometheus_server.image.name	Prometheus イメージの名前	文字列	prometheus
monitoring.prometheus_server.image.tag	Prometheus イメージ タグ。バージョンをアップグレードしている場合は、この値の更新が必要になることがあります。	文字列	v2.17.1_vmware.1
monitoring.prometheus_server.image.pullPolicy	Prometheus イメージ プル ポリシー	文字列	IfNotPresent
monitoring.alertmanager.config.slack_demo	Alertmanager のスラック通知構成	文字列	`slack_demo:   name: slack_demo   slack_configs:   - api_url: https:// hooks.slack.com   channel: '#alertmanager- test'`

表 14-5. Prometheus 構成パラメータ （続き）

パラメータ	説明	タイプ	デフォルト
monitoring.alertmanager.config.email_receiver	Alertmanager の E メール通知構成	文字列	`email_receiver:   name: email-receiver   email_configs:   - to: demo@tanzu.com   send_resolved: false   from: from-email@tanzu.com   smarthost: smtp.eample.com:25   require_tls: false`
monitoring.alertmanager.service.type	Alertmanager を公開するサービスのタイプ。サポートされている値：ClusterIP	文字列	ClusterIP
monitoring.alertmanager.image.repository	Alertmanager イメージを含むリポジトリの場所。デフォルトは、パブリック VMware レジストリです。プライベート リポジトリを使用している場合は（エアギャップ環境など）、この値を変更します。	文字列	projects.registry.vmware.com/tkg/prometheus
monitoring.alertmanager.image.name	Alertmanager イメージの名前	文字列	alertmanager
monitoring.alertmanager.image.tag	Alertmanager イメージ タグ。バージョンをアップグレードしている場合は、この値の更新が必要になることがあります。	文字列	v0.20.0_vmware.1
monitoring.alertmanager.image.pullPolicy	Alertmanager イメージ プル ポリシー	文字列	IfNotPresent
monitoring.alertmanager.pvc.annotations	ストレージ クラスの注釈	マップ	{}
monitoring.alertmanager.pvc.storage_class	パーシステント ボリュームの要求に使用するストレージ クラス。これはデフォルトで null で、デフォルトのプロビジョナが使用されます。	文字列	null
monitoring.alertmanager.pvc.accessMode	パーシステント ボリューム要求に対しアクセス モードを定義します。サポートされている値：ReadWriteOnce、ReadOnlyMany、ReadWriteMany	文字列	ReadWriteOnce

表 14-5. Prometheus 構成パラメータ （続き）

パラメータ	説明	タイプ	デフォルト
monitoring.alertmanager.pvc.storage	パーシステント ボリューム要求に対しストレージ サイズを定義します	文字列	2Gi
monitoring.alertmanager.deployment.replicas	alertmanager レプリカの数	整数型	1
monitoring.kube_state_metrics.image.repository	kube-state-metircs イメージを含むリポジトリ。デフォルトは、パブリック VMware レジストリです。プライベート リポジトリを使用している場合は（エアギャップ環境など）、この値を変更します。	文字列	projects.registry.vmware.com/tkg/prometheus
monitoring.kube_state_metrics.image.name	kube-state-metircs イメージの名前	文字列	kube-state-metrics
monitoring.kube_state_metrics.image.tag	kube-state-metircs イメージ タグ。バージョンをアップグレードしている場合は、この値の更新が必要になることがあります。	文字列	v1.9.5_vmware.1
monitoring.kube_state_metrics.image.pullPolicy	kube-state-metircs イメージ プル ポリシー	文字列	IfNotPresent
monitoring.kube_state_metrics.deployment.replicas	kube-state-metrics レプリカの数	整数型	1
monitoring.node_exporter.image.repository	node-exporter イメージを含むリポジトリ。デフォルトは、パブリック VMware レジストリです。プライベート リポジトリを使用している場合は（エアギャップ環境など）、この値を変更します。	文字列	projects.registry.vmware.com/tkg/prometheus
monitoring.node_exporter.image.name	node-exporter イメージの名前	文字列	node-exporter
monitoring.node_exporter.image.tag	node-exporter イメージ タグ。バージョンをアップグレードしている場合は、この値の更新が必要になることがあります。	文字列	v0.18.1_vmware.1
monitoring.node_exporter.image.pullPolicy	node-exporter イメージ プル ポリシー	文字列	IfNotPresent
monitoring.node_exporter.hostNetwork	ポッドが `hostNetwork: true` に設定されている場合は、そのポッドで、ネットワークの名前空間とノードのネットワーク リソースを使用できます。	boolean	false
monitoring.node_exporter.deployment.replicas	node-exporter レプリカの数	整数型	1

表 14-5. Prometheus 構成パラメータ （続き）

パラメータ	説明	タイプ	デフォルト
monitoring.pushgateway.image.repository	pushgateway イメージを含むリポジトリ。デフォルトは、パブリック VMware レジストリです。プライベート リポジトリを使用している場合は（エアギャップ環境など）、この値を変更します。	文字列	projects.registry.vmware.com/tkg/prometheus
monitoring.pushgateway.image.name	pushgateway イメージの名前	文字列	pushgateway
monitoring.pushgateway.image.tag	pushgateway イメージ タグ。バージョンをアップグレードしている場合は、この値の更新が必要になることがあります。	文字列	v1.2.0_vmware.1
monitoring.pushgateway.image.pullPolicy	pushgateway イメージ プル ポリシー	文字列	IfNotPresent
monitoring.pushgateway.deployment.replicas	pushgateway レプリカの数	整数型	1
monitoring.cadvisor.image.repository	cadvisor イメージを含むリポジトリ。デフォルトは、パブリック VMware レジストリです。プライベート リポジトリを使用している場合は（エアギャップ環境など）、この値を変更します。	文字列	projects.registry.vmware.com/tkg/prometheus
monitoring.cadvisor.image.name	cadvisor イメージの名前	文字列	cadvisor
monitoring.cadvisor.image.tag	cadvisor イメージ タグ。バージョンをアップグレードしている場合は、この値の更新が必要になることがあります。	文字列	v0.36.0_vmware.1
monitoring.cadvisor.image.pullPolicy	cadvisor イメージ プル ポリシー	文字列	IfNotPresent
monitoring.cadvisor.deployment.replicas	cadvisor レプリカの数	整数型	1
monitoring.ingress.enabled	prometheus および alertmanager に対し ingress を有効または無効にします	boolean	false ingress を使用するには、このフィールドを `true` に設定して、Contour Ingress 用 TKG 拡張機能のデプロイと管理をデプロイします。Prometheus にアクセスするには、`/etc/hosts` をワーカー ノードの IP アドレスにマッピングするエントリを使用して、ローカル `prometheus.system.tanzu` を更新します。

表 14-5. Prometheus 構成パラメータ （続き）

パラメータ	説明	タイプ	デフォルト
monitoring.ingress.virtual_host_fqdn	Prometheus および Alertmanager にアクセスするホスト名	文字列	prometheus.system.tanzu
monitoring.ingress.prometheus_prefix	prometheus のパス プリフィックス	文字列	/
monitoring.ingress.alertmanager_prefix	alertmanager のパス プリフィックス	文字列	/alertmanager/
monitoring.ingress.tlsCertificate.tls.crt	独自の TLS 証明書を使用する場合は、ingress のオプション証明書。自己署名証明書はデフォルトで生成されます	文字列	Generated cert
monitoring.ingress.tlsCertificate.tls.key	独自の TLS 証明書を使用する場合は、ingress のオプション証明書プライベート キー。	文字列	Generated cert key

表 14-6. Prometheus_Server Configmap の構成可能フィールド

パラメータ	説明	タイプ	デフォルト
evaluation_interval	ルールを評価する頻度	duration	1m
scrape_interval	ターゲットを取得する頻度	duration	1m
scrape_timeout	取得要求がタイムアウトになるまでの期間	duration	10s
rule_files	ルール ファイルは glob のリストを指定します。ルールとアラートは、一致するすべてのファイルから読み取られます	yaml ファイル	
scrape_configs	取得構成のリスト。	リスト	
job_name	デフォルトで取得されたメトリックに割り当てられたジョブ名	文字列	
kubernetes_sd_configs	Kubernetes サービス検出構成のリスト。	リスト	
relabel_configs	ターゲット再ラベル付け構成のリスト。	リスト	
action	正規表現の一致に基づいて実行するアクション。	文字列	
regex	抽出された値が一致する正規表現。	文字列	
source_labels	ソース ラベルは、既存のラベルから値を選択します。	文字列	
scheme	要求に使用されるプロトコル スキームを構成します。	文字列	

表 14-6. Prometheus_Server Configmap の構成可能フィールド （続き）

パラメータ	説明	タイプ	デフォルト
tls_config	取得要求の TLS 設定を構成します。	文字列	
ca_file	API サーバ証明書を検証する CA 証明書。	ファイル名	
insecure_skip_verify	サーバ証明書の検証を無効にします。	boolean	
bearer_token_file	オプションのベアラー トークン ファイル認証情報。	ファイル名	
replacement	正規表現が一致する場合に正規表現の置き換えが実行される置き換え値。	文字列	
target_label	結果値が置換アクションで書き込まれるラベル。	文字列	

表 14-7. Alertmanager Configmap の構成可能フィールド

パラメータ	説明	タイプ	デフォルト
resolve_timeout	ResolveTimeout は、アラートに EndsAt が含まれていない場合に alertmanager によって使用されるデフォルト値です	duration	5m
smtp_smarthost	E メールが送信される SMTP ホスト。	duration	1m
slack_api_url	スラック Webhook URL。	文字列	global.slack_api_url
pagerduty_url	API 要求の送信先の pagerduty URL。	文字列	global.pagerduty_url
テンプレート	カスタム通知テンプレート定義を読み取るためのファイル	ファイルのパス	
group_by	アラートをラベルでグループ化します	文字列	
group_interval	グループに追加された新しいアラートに関する通知を送信するまでの待機時間を設定	duration	5m
group_wait	アラートのグループに関する通知を最初に送信するまでの待機時間	duration	30s
repeat_interval	通知がアラートに対してすでに正常に送信されている場合に再送信するまでの待機時間	duration	4h
receivers	通知受信者のリスト。	リスト	
severity	インシデントの重要度。	文字列	

表 14-7. Alertmanager Configmap の構成可能フィールド （続き）

パラメータ	説明	タイプ	デフォルト
channel	通知の送信先となるチャネルまたはユーザー。	文字列	
html	E メール通知の HTML 本文。	文字列	
text	E メール通知のテキスト本文。	文字列	
send_resolved	解決済みアラートについて通知するかどうか。	ファイル名	
email_configs	E メール統合の構成	boolean	

ポッドの注釈により、取得プロセスの適切な管理が可能になります。これらの注釈は、ポッド メタデータの一部である必要があります。Services、DaemonSets などのその他のオブジェクトで設定されている場合は無効になります。

表 14-8. Prometheus ポッドの注釈

ポッドの注釈	説明
`prometheus.io/scrape`	デフォルトの構成ではすべてのポッドが取得されます。false に設定した場合、この注釈はポッドを取得プロセスから除外します。
`prometheus.io/path`	メトリック パスが /metrics ではない場合、この注釈を使用して定義します。
`prometheus.io/port`	ポッドをポッドの宣言されたポートではなく、指示されたポートで取得します（宣言がない場合、デフォルトはポートなしターゲットです）。

以下の DaemonSet マニフェストは、ポート 9102 ですべてのポッドを取得するように Prometheus に指示します。

```
apiVersion: apps/v1beta2 # for versions before 1.8.0 use extensions/v1beta1
kind: DaemonSet
metadata:
  name: fluentd-elasticsearch
  namespace: weave
  labels:
    app: fluentd-logging
spec:
  selector:
    matchLabels:
      name: fluentd-elasticsearch
  template:
    metadata:
      labels:
        name: fluentd-elasticsearch
      annotations:
        prometheus.io/scrape: 'true'
        prometheus.io/port: '9102'
```

```
    spec:
      containers:
      - name: fluentd-elasticsearch
        image: gcr.io/google-containers/fluentd-elasticsearch:1.20
```

Grafana Monitoring 用 TKG 拡張機能のデプロイと管理

このトピックでは、Grafana 用 TKG 拡張機能 v1.3.1 をデプロイおよび管理する方法について説明します。Grafana を使用すると、保存納場所に関係なく、メトリックの照会、可視化、アラート、および確認を行うことができます。Grafana は、アプリケーション データからグラフを作成し、可視化するためのツールを提供します。Tanzu Kubernetes クラスタのメトリックを生成して表示するには、Grafana 用 TKG 拡張機能をデプロイします。

Grafana 拡張機能の前提条件

拡張機能をデプロイするには、次の要件を満たす必要があります。

- クラスタをプロビジョニングします。TKGS v1alpha2 API を使用して Tanzu Kubernetes クラスタをプロビジョニングするためのワークフローを参照してください。

  注： デフォルトの serviceDomain (`cluster.local`) を使用するクラスタをデプロイする必要があります。

- クラスタに接続します。vCenter Single Sign-On ユーザーとして Tanzu Kubernetes クラスタに接続を参照してください。
- kubectl コマンドを実行するクライアント ホストへの TKG 拡張機能 v1.3.1 バンドルのダウンロードを実行します。
- ターゲット クラスタへの TKG 拡張機能の前提条件のインストールを実行します。

Grafana 拡張機能の追加要件

Grafana 監視用 TKG 拡張機能 v1.3.1 には、インストール前およびインストール後の追加要件があります。

- Grafana の監視拡張機能にはデフォルトのパーシステント ストレージ クラスが必要です。デフォルトのパーシステント ストレージ クラスを使用してクラスタを作成するか、拡張機能をデプロイするときに Grafana 構成ファイルでクラスタを指定することができます。TKG 拡張機能のパーシステント ストレージの要件の確認を参照してください。
- Grafana 拡張機能がデプロイされたら、ClusterIP（デフォルト）、NodePort、または LoadBalancer のいずれかの Kubernetes サービス タイプによって公開される IP アドレスを使用して、HTTP/S 経由で Grafana ダッシュボードにアクセスします。クラスタの外部から Grafana ダッシュボードにアクセスするには、Grafana をデプロイする前に Contour 拡張機能をデプロイします。Contour 拡張機能をデプロイするには、Grafana Monitoring 用 TKG 拡張機能のデプロイと管理を参照してください。

Grafana は次の Kubernetes サービス タイプをサポートします。

サービス タイプ	説明	アクセス可能性
`ClusterIP`	クラスタ内の IP アドレスでサービスを公開します。	サービスには、クラスタ内からのみアクセスできます。
`NodePort`	静的ポートの各ノードの IP アドレスでサービスを公開します。	サービスには、クラスタの外部からアクセスできます。
`LoadBalancer`	ロード バランサを使用してサービスを外部で公開します。	サービスには、クラスタの外部からアクセスできます。

`ClusterIP` がデフォルトですが、アクセスできるのはクラスタ内からのみです。スーパーバイザー クラスタ に対して NSX-T ネットワークを使用している場合は、LoadBalancer タイプの Contour Envoy サービスを作成します。スーパーバイザー クラスタ に対して vSphere Distributed Switch ネットワークを使用している場合は、要件に応じて LoadBalancer タイプまたは NodePort タイプの Contour Envoy サービスを作成します。

可視化と分析のための Grafana 拡張機能のデプロイ

Grafana 用 TKG 拡張機能は、単一のコンテナをデプロイします。詳細については、『https://grafana.com/』を参照してください。

コンテナ	リソースの種類	レプリカ	説明
Grafana	デプロイ	2	データの可視化

この拡張機能は、https://projects.registry.vmware.com/ にある VMware パブリック レジストリからコンテナをプルするように構成されています。プライベート レジストリを使用する場合は、データ値と拡張機能構成に含まれるエンドポイント URL を、対応するものに変更します。Grafana 拡張機能の構成を参照してください。

1 Grafana 拡張機能のすべての前提条件を満たしていることを確認します。Grafana 拡張機能の前提条件を参照してください。

2 Grafana 拡張機能のディレクトリに移動します。

```
cd /tkg-extensions-v1.3.1+vmware.1/extensions/monitoring/grafana
```

3 `tanzu-system-monitoring` 名前空間と、Grafana サービスのアカウントおよびロール オブジェクトを作成します。

```
kubectl apply -f namespace-role.yaml
```

4 Grafana データ値ファイルを作成します。

サンプル データ値ファイルには、最低限必要な構成が含まれています。

```
cp grafana-data-values.yaml.example grafana-data-values.yaml
```

5 `grafana-data-values.yaml` を更新して Grafana 拡張機能を構成します。

必要に応じて構成をカスタマイズします。Grafana 拡張機能の構成を参照してください。

admin_password は base64 でエンコードする必要がありますが、base64 でエンコードされていない場合でも、拡張機能のデプロイはブロックされません。次の例では、パスワード「admin」は base64 でエンコードされています。Grafana のパスワードを独自にエンコードするには、次のサイトにアクセスします：https://www.base64encode.org/。

クラスタがデフォルトのストレージ クラスでプロビジョニングされていない場合は、データ値ファイル内で指定できます。また、名前空間にパーシステント ボリュームの要求を格納するのに必要なストレージが確保されている必要があります。

```
monitoring:
  grafana:
    image:
      repository: "projects.registry.vmware.com/tkg/grafana"
    pvc:
      storage_class: vwt-storage-policy
      storage: "8Gi"
    secret:
      admin_password: "YWRtaW4="
  grafana_init_container:
    image:
      repository: "projects.registry.vmware.com/tkg/grafana"
  grafana_sc_dashboard:
    image:
      repository: "projects.registry.vmware.com/tkg/grafana"
```

LoadBalancer タイプまたは NodePort タイプの Envoy サービスを使用して Contour をデプロイした場合は、これを次のように構成ファイルで指定します。詳細については Grafana 拡張機能の構成を参照してください。

```
monitoring:
  grafana:
    service:
      type: LoadBalancer OR NodePort
```

デフォルトでは、Grafana 拡張機能は Grafana ダッシュボードにアクセスするための完全修飾ドメイン名 (FQDN) grafana.system.tanzu を作成します。この FQDN をカスタマイズするには、monitoring.grafana.ingress.virtual_host_fqdn の構成ファイルで目的のホスト名を指定します。詳細については Grafana 拡張機能の構成を参照してください。

6 grafana-data-values ファイルを使用して Grafana シークレットを作成します。

```
kubectl create secret generic grafana-data-values --from-file=values.yaml=grafana-data-values.yaml -n tanzu-system-monitoring
```

grafana-data-values シークレットが tanzu-system-monitoring 名前空間に作成されている。kubectl get secrets -n tanzu-system-monitoring を使用して確認します。

7 Grafana 拡張機能をデプロイします。

```
kubectl apply -f grafana-extension.yaml
```

成功すると、Grafana アプリケーションが作成されます：`app.kappctrl.k14s.io/grafana created`。

8 Grafana アプリケーションのステータスを確認します。

```
kubectl get app grafana -n tanzu-system-monitoring
```

ステータスが `Reconciling` から `Reconcile succeeded` に変わります。ステータスが `Reconcile failed` の場合は、トラブルシューティングを参照してください。

9 Grafana アプリケーションの詳細なステータスを確認します。

```
kubectl get app grafana -n tanzu-system-monitoring -o yaml
```

10 Grafana のデプロイを確認します。

```
kubectl get deployments -n tanzu-system-monitoring
```

LoadBalancer タイプの Contour Envoy サービスを使用した Grafana ダッシュボードへのアクセス

LoadBalancer タイプの前提条件である Contour Envoy サービスがデプロイされていて、Grafana 構成ファイル内で指定されている場合は、ロード バランサの外部 IP アドレスを取得し、Grafana FQDN の DNS レコードを作成します。

1 LoadBalancer タイプの Envoy サービスの `External-IP` アドレスを取得します。

```
kubectl get service envoy -n tanzu-system-ingress
```

返された `External-IP` アドレスが次の例のように表示されます。

```
NAME    TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)                      AGE
envoy   LoadBalancer   10.99.25.220   10.195.141.17   80:30437/TCP,443:30589/TCP   3h27m
```

または、次のコマンドを使用して `External-IP` アドレスを取得することもできます。

```
kubectl get svc envoy -n tanzu-system-ingress -o
jsonpath='{.status.loadBalancer.ingress[0]}'
```

2 Grafana 拡張機能のインストールを確認するには、次の例のように、ロード バランサの `External-IP` アドレスにマッピングされる Grafana の FQDN を指定してローカルの `/etc/hosts` ファイルを更新します。

```
127.0.0.1 localhost
127.0.1.1 ubuntu
# TKG Grafana Extension with Envoy Load Balancer
10.195.141.17 grafana.system.tanzu
```

3 https://grafana.system.tanzu に移動して、Grafana ダッシュボードにアクセスします。

サイトでは自己署名証明書が使用されるため、ダッシュボードにアクセスする前にブラウザ固有のセキュリティ警告を確認しなければならない場合があります。

4 本番環境にアクセスする場合は、DNS サーバで 2 つの CNAME レコードを作成し、Envoy サービス ロード バランサの `External-IP` アドレスを Grafana ダッシュボードにマッピングします。

NodePort タイプの Contour Envoy サービスを使用した Grafana ダッシュボードへのアクセス

NodePort タイプの前提条件である Contour Envoy サービスがデプロイされていて、Grafana 構成ファイル内で指定されている場合は、ワーカー ノードの仮想マシン IP アドレスを取得し、Grafana FQDN の DNS レコードを作成します。

1 クラスタがプロビジョニングされている vSphere 名前空間 にコンテキストを切り替えます。

```
kubectl config use-context VSPHERE-NAMESPACE
```

2 クラスタ内のノードを一覧表示します。

```
kubectl get virtualmachines
```

クラスタ ノードが次のように表示されます。

```
NAME                                          POWERSTATE   AGE
tkgs-cluster-X-control-plane-6dgln            poweredOn    6h7m
tkgs-cluster-X-control-plane-j6hq6            poweredOn    6h10m
tkgs-cluster-X-control-plane-xc25f            poweredOn    6h14m
tkgs-cluster-X-workers-9twdr-59bc54dc97-kt4cm poweredOn    6h12m
tkgs-cluster-X-workers-9twdr-59bc54dc97-pjptr poweredOn    6h12m
tkgs-cluster-X-workers-9twdr-59bc54dc97-t45mn poweredOn    6h12m
```

3 ワーカー ノードの 1 つを選択し、次のコマンドを使用して記述します。

```
kubectl describe virtualmachines tkgs-cluster-X-workers-9twdr-59bc54dc97-kt4cm
```

4 `Vm Ip: 10.115.22.43` などの仮想マシンの IP アドレスを見つけます。

5 Grafana 拡張機能のインストールを確認するには、次の例のように、ワーカー ノードの IP アドレスにマッピングされる Grafana の FQDN を指定してローカルの `/etc/hosts` ファイルを更新します。

```
127.0.0.1 localhost
127.0.1.1 ubuntu
# TKGS Grafana with Envoy NodePort
10.115.22.43 grafana.system.tanzu
```

6 https://grafana.system.tanzu に移動して、Grafana ダッシュボードにアクセスします。

サイトでは自己署名証明書が使用されるため、ダッシュボードにアクセスする前にブラウザ固有のセキュリティ警告を確認しなければならない場合があります。

Grafana のデプロイのトラブルシューティング

デプロイまたは調整が失敗した場合は、`kubectl get pods -A` を実行してポッドのステータスを確認します。`contour` および `envoy` ポッドは、`Running` である必要があります。ポッドのステータスが `ImagePullBackOff` または `ImageCrashLoopBackOff` の場合は、コンテナ イメージをプルできていません。データ値と拡張機能 YAML ファイルに記述されているリポジトリ URL を確認し、これらが正確であることを確認します。

コンテナログを確認します。`name-XXXX` は、`kubectl get pods -A` を実行したときの一意のポッド名です。

```
kubectl logs pod/grafana-XXXX -c grafana -n tanzu-system-monitoring
```

Grafana 拡張機能の更新

Tanzu Kubernetes クラスタにデプロイされている Grafana 拡張機能を更新します。

1 `grafana-data-values` シークレットから現在の Grafana データ値を取得します。

```
kubectl get secret grafana-data-values -n tanzu-system-monitoring -o 'go-
template={{ index .data "values.yaml" }}' | base64 -d > grafana-data-values.yaml
```

2 `grafana-data-values.yaml` 内の Grafana データ値を更新します。Grafana 拡張機能の構成を参照してください。

3 Grafana データ値のシークレットを更新します。

```
kubectl create secret generic grafana-data-values --from-file=values.yaml=grafana-data-
values.yaml -n tanzu-system-monitoring -o yaml --dry-run | kubectl replace -f-
```

Grafana 拡張機能が更新されたデータ値で調整されます。

注： デフォルトでは、kapp-controller によってアプリケーションが 5 分ごとに同期されます。更新は 5 分以内に有効になります。更新をすぐに有効にするには、`grafana-extension.yaml` 内の `syncPeriod` の値を小さくし、`kubectl apply -f grafana-extension.yaml` を使用して Grafana 拡張機能を適用します。

4 拡張機能のステータスを確認します。

```
kubectl get app grafana -n tanzu-system-monitoring
```

Grafana が更新されると、ステータスが `Reconcile Succeeded` に変更されます。

5 詳細なステータスを表示し、必要に応じてトラブルシューティングを行います。

```
kubectl get app grafana -n tanzu-system-monitoring -o yaml
```

Grafana 拡張機能の削除

Tanzu Kubernetes クラスタから Grafana 拡張機能を削除します。

注： 手順を順番に実行します。Grafana アプリケーションが完全に削除される前に、名前空間、サービス アカウント、ロール オブジェクトを削除しないでください。これを行うと、システム エラーが発生する可能性があります。

注： Prometheus および Grafana 拡張機能は、同じ名前空間 (`tanzu-system-monitoring`) にデプロイされます。両方の拡張機能を同じクラスタにデプロイしている場合は、名前空間を削除する前に各拡張機能を削除する必要があります。

1 Grafana 拡張機能のディレクトリに移動します。

```
cd /tkg-extensions-v1.3.1+vmware.1/extensions/monitoring/grafana
```

2 Grafana アプリケーションを削除します。

```
kubectl delete app grafana -n tanzu-system-monitoring
```

予期される結果：`app.kappctrl.k14s.io "grafana" deleted`。

3 Grafana アプリケーションが削除されていることを確認します。

```
kubectl get app grafana -n tanzu-system-monintoring
```

予期される結果：`apps.kappctrl.k14s.io "grafana" not found`。

4 `tanzu-system-monitoring` 名前空間と、Grafana サービスのアカウントおよびロール オブジェクトを削除します。

注意: Prometheus がデプロイされている場合は、この手順は実行しません。

```
kubectl delete -f namespace-role.yaml
```

5 Grafana を再デプロイする場合は、シークレット `grafana-data-values` を削除します。

```
kubectl delete secret grafana-data-values -n tanzu-system-monitoring
```

予期される結果：`secret "grafana-data-values" deleted`。

Grafana 拡張機能のアップグレード

既存の Grafana 拡張機能がデプロイされている場合は、最新バージョンを使用するようにアップグレードできます。

1 Grafana の構成マップをエクスポートし、バックアップとして保存します。

```
kubectl get configmap grafana -n tanzu-system-monitoring -o 'go-template={{ index .data
"grafana.yaml" }}' > grafana-configmap.yaml
```

2 既存の Grafana 拡張機能を削除します。Grafana 拡張機能の削除を参照してください。

3 Grafana 拡張機能をデプロイします。可視化と分析のための Grafana 拡張機能のデプロイを参照してください。

Grafana 拡張機能の構成

Grafana の構成は、`/tkg-extensions-v1.3.1+vmware.1/extensions/monitoring/grafana/grafana-data-values.yaml` で設定されます。

表 14-9. Grafana 構成パラメータ

パラメータ	説明	タイプ	デフォルト
monitoring.namespace	Prometheus がデプロイされる名前空間	文字列	tanzu-system-monitoring
monitoring.create_namespace	フラグは、monitoring.namespace で指定された名前空間を作成するかどうかを示します	boolean	false
monitoring.grafana.cluster_role.apiGroups	grafana clusterrole に対して定義された api グループ	リスト	[""]
monitoring.grafana.cluster_role.resources	grafana clusterrole に対して定義されたリソース	リスト	["configmaps", "secrets"]
monitoring.grafana.cluster_role.verbs	clusterrole に対して定義されたアクセス権限	リスト	["get", "watch", "list"]
monitoring.grafana.config.grafana_ini	Grafana 構成ファイルの詳細	構成ファイル	grafana.ini このファイルでは、`grafana_net` URL を使用して Grafana と統合し、たとえばダッシュボードを Grafana.com から直接インポートします。
monitoring.grafana.config.datasource.type	Grafana データソース タイプ	文字列	prometheus
monitoring.grafana.config.datasource.access	アクセス モード。proxy または direct (ユーザー インターフェイスのサーバまたはブラウザ)	文字列	proxy
monitoring.grafana.config.datasource.isDefault	デフォルトの Grafana データソースとしてマーク	boolean	true
monitoring.grafana.config.provider_yaml	grafana ダッシュボード プロバイダを定義する構成ファイル	yaml ファイル	provider.yaml
monitoring.grafana.service.type	Grafana を公開するサービスのタイプ。サポートされている値：ClusterIP、NodePort、LoadBalancer	文字列	vSphere: NodePort、aws/azure: LoadBalancer
monitoring.grafana.pvc.storage_class	パーシステント ボリューム要求に対しアクセス モードを定義します。サポートされている値：ReadWriteOnce、ReadOnlyMany、ReadWriteMany	文字列	ReadWriteOnce

表 14-9. Grafana 構成パラメータ （続き）

パラメータ	説明	タイプ	デフォルト
monitoring.grafana.pvc.storage	パーシステント ボリューム要求に対しストレージ サイズを定義します	文字列	2Gi
monitoring.grafana.deployment.replicas	grafana レプリカの数	整数型	1
monitoring.grafana.image.repository	Grafana イメージを含むリポジトリの場所。デフォルトは、パブリック VMware レジストリです。プライベート リポジトリを使用している場合は（エアギャップ環境など）、この値を変更します。	文字列	projects.registry.vmware.com/tkg/grafana
monitoring.grafana.image.name	Grafana イメージの名前	文字列	grafana
monitoring.grafana.image.tag	Grafana イメージ タグ。バージョンをアップグレードしている場合は、この値の更新が必要になることがあります。	文字列	v7.3.5_vmware.1
monitoring.grafana.image.pullPolicy	Grafana イメージ プル ポリシー	文字列	IfNotPresent
monitoring.grafana.secret.type	Grafana ダッシュボードに定義されたシークレット タイプ	文字列	不透明
monitoring.grafana.secret.admin_user	Grafana ダッシュボードにアクセスするユーザー名	文字列	YWRtaW4= 値は base64 でエンコードされています。デコードには以下を実行します：`echo "xxxxxx" | base64 --decode`
monitoring.grafana.secret.admin_password	Grafana ダッシュボードにアクセスするパスワード	文字列	null
monitoring.grafana.secret.ldap_toml	ldap 認証を使用している場合、ldap 構成ファイルのパス	文字列	""
monitoring.grafana_init_container.image.repository	grafana init コンテナ イメージを含むリポジトリ。デフォルトは、パブリック VMware レジストリです。プライベート リポジトリを使用している場合は（エアギャップ環境など）、この値を変更します。	文字列	projects.registry.vmware.com/tkg/grafana
monitoring.grafana_init_container.image.name	grafana init コンテナ イメージの名前	文字列	k8s-sidecar
monitoring.grafana_init_container.image.tag	grafana init コンテナ イメージ タグ。バージョンをアップグレードしている場合は、この値の更新が必要になることがあります。	文字列	0.1.99

表 14-9. Grafana 構成パラメータ （続き）

パラメータ	説明	タイプ	デフォルト
monitoring.grafana_init_container.image.pullPolicy	grafana init コンテナ イメージ プル ポリシー	文字列	IfNotPresent
monitoring.grafana_sc_dashboard.image.repository	Grafana ダッシュボード イメージを含むリポジトリ。デフォルトは、パブリック VMware レジストリです。プライベート リポジトリを使用している場合は（エアギャップ環境など）、この値を変更します。	文字列	projects.registry.vmware.com/tkg/grafana
monitoring.grafana_sc_dashboard.image.name	grafana ダッシュボード イメージの名前	文字列	k8s-sidecar
monitoring.grafana_sc_dashboard.image.tag	grafana ダッシュボード イメージ タグ。バージョンをアップグレードしている場合は、この値の更新が必要になることがあります。	文字列	0.1.99
monitoring.grafana_sc_dashboard.image.pullPolicy	grafana ダッシュボード イメージ プル ポリシー	文字列	IfNotPresent
monitoring.grafana.ingress.enabled	Grafana に対し ingress を有効または無効にします	boolean	true
monitoring.grafana.ingress.virtual_host_fqdn	grafana にアクセスするホスト名	文字列	grafana.system.tanzu
monitoring.grafana.ingress.prefix	grafana のパス プリフィックス	文字列	/
monitoring.grafana.ingress.tlsCertificate.tls.crt	独自の TLS 証明書を使用する場合は、ingress のオプション証明書。自己署名証明書はデフォルトで生成されます	文字列	Generated cert
monitoring.grafana.ingress.tlsCertificate.tls.key	独自の TLS 証明書を使用する場合は、ingress のオプション証明書プライベート キー。	文字列	Generated cert key

Harbor レジストリ用 TKG 拡張機能のデプロイと管理

Harbor はオープンソースのコンテナ レジストリです。Tanzu Kubernetes クラスタにデプロイするコンテナ イメージのプライベート レジストリ ストアとして、Harbor レジストリ用 TKG 拡張機能をデプロイできます。

Harbor 拡張機能のバージョン依存関係

Tanzu Kubernetes Grid サービス によってプロビジョニングされる Tanzu Kubernetes クラスタに Harbor レジストリ用 TKG 拡張機能をインストールするには、次の最小要件を満たす必要があります。

コンポーネント	最小バージョン
vCenter Server	7.0.2.00400
vSphere 名前空間	0.0.10-18245956
スーパーバイザー クラスタ	v1.20.2+vmware.1-vsc0.0.10-18245956
Tanzu Kubernetes リリース	v1.20.7+vmware.1-tkg.1.7fb9067

Harbor 拡張機能の前提条件

Harbor レジストリ用 TKG 拡張機能 v1.3.1 をデプロイする前に、次の前提条件を満たす必要があります。

- クラスタをプロビジョニングします。TKGS v1alpha2 API を使用して Tanzu Kubernetes クラスタをプロビジョニングするためのワークフローを参照してください。
- クラスタに接続します。vCenter Single Sign-On ユーザーとして Tanzu Kubernetes クラスタに接続を参照してください。
- kubectl コマンドを実行するクライアント ホストへの TKG 拡張機能 v1.3.1 バンドルのダウンロードを実行します。
- ターゲット クラスタへの TKG 拡張機能の前提条件のインストールを実行します。

Harbor 拡張機能の追加要件

Harbor レジストリ用 TKG 拡張機能 v1.3.1 には、インストール前およびインストール後の追加要件があります。

- Harbor 拡張機能には、デフォルトの PVC ストレージ クラスが必要です。TKG 拡張機能のパーシステント ストレージの要件の確認を参照してください。
- Harbor 拡張機能には、HTTP/S 入力が必要です。具体的には、Harbor サービスは Contour 拡張機能の Envoy サービスを介して公開されます。前提条件として、Contour 拡張機能をデプロイします。Contour Ingress 用 TKG 拡張機能のデプロイと管理を参照してください。
  - スーパーバイザー クラスタ に対して NSX-T ネットワークを使用している場合は、LoadBalancer タイプの Envoy サービスを作成します。
  - スーパーバイザー クラスタ に対して vSphere vDS ネットワークを使用している場合は、環境と要件に応じて LoadBalancer タイプまたは NodePort タイプの Envoy サービスを作成します。
- Harbor 拡張機能には、DNS が必要です。Harbor 拡張機能のインストールが終わったら、DNS を構成する必要があります。
  - テストと検証のために、Harbor および Notary の FQDN をローカル `/etc/hosts` ファイルに追加します。以下の手順では、これを行う方法を説明します。
  - 本番環境の Harbor では、BIND などのローカル DNS サーバ、または AWS Route53、Azure DNS、Google CloudDNS などのパブリック クラウドのいずれかの DNS ゾーンが必要です。DNS を設定した後、Harbor の FQDN を DNS サーバに自動的に登録するには、外部 DNS 拡張機能をインストールします。外部 DNS サービス検出のための TKG 拡張機能のデプロイと管理を参照してください。

Harbor 拡張機能のデプロイ

Harbor レジストリ用 TKG 拡張機能は、いくつかのコンテナをクラスタにインストールします。詳細については、『https://goharbor.io/』を参照してください。

コンテナ	リソースの種類	レプリカ	説明
harbor-core	デプロイ	1	Envoy 用の管理および構成サーバ
harbor-database	ポッド	1	Postgres データベース
harbor-jobservice	デプロイ	1	Harbor ジョブ サービス
harbor-notary-server	デプロイ	1	Harbor Notary サービス
harbor-notary-signer	デプロイ	1	Harbor Notary
harbor-portal	デプロイ	1	Harbor Web インターフェイス
harbor-redis	ポッド	1	Harbor Redis インスタンス
harbor-registry	デプロイ	2	Harbor コンテナ レジストリ インスタンス
harbor-trivy	ポッド	1	Harbor イメージ脆弱性スキャナ

TKG 拡張機能を使用して Harbor レジストリをインストールするには、次の手順を実行します。

1 拡張機能のすべての前提条件を満たしていることを確認します。Harbor 拡張機能の前提条件と Harbor 拡張機能の追加要件を参照してください。

2 Harbor 拡張機能のディレクトリに移動します。

```
cd /tkg-extensions-v1.3.1+vmware.1/extensions/registry/harbor
```

3 `tanzu-system-registry` 名前空間、Harbor サービス アカウント、およびロールを作成します。

```
kubectl apply -f namespace-role.yaml
```

4 Harbor データ値ファイルを作成します。

```
cp harbor-data-values.yaml.example harbor-data-values.yaml
```

5 必須のパスワードとシークレットを `harbor-data-values.yaml` に指定します。

Harbor レジストリには、この表に記載されたいくつかのパスワードとシークレットが必要です。

パスワードまたはシークレット	説明
harborAdminPassword	Harbor 管理者の初期パスワード。
secretKey	暗号化に使用されるプライベート キー。16 文字の文字列にする必要があります。
database.password	Postgres データベースの初期パスワード。
core.secret	コア サーバが他のコンポーネントと通信するときに使用されるシークレット。
core.xsrfKey	XSRF キー。32 文字の文字列にする必要があります。

パスワードまたはシークレット	説明
jobservice.secret	ジョブ サービスが他のコンポーネントと通信するときに使用されるシークレット。
registry.secret	クライアントおよびレジストリ ストレージ バックエンドからアップロード状態を保護するために使用されるシークレット。

ランダムなパスワードとシークレットを自動的に生成して harbor-data-values.yaml ファイルにポピュレートされるようにするには、次のコマンドを実行します。

```
bash generate-passwords.sh harbor-data-values.yaml
```

成功すると、次のメッセージが表示されます。

```
Successfully generated random passwords and secrets in harbor-data-values.yaml
```

harbor-data-values.yaml ファイルを開き、必須のパスワードとシークレットを確認します。

6 必要に応じて、harbor-data-values.yaml にその他の Harbor 構成値を指定します。一般的に更新される値は、次のとおりです。

構成フィールド	説明
hostname	デフォルトの Harbor ホスト名は、core.harbor.domain です。 必要に応じて、要件に合わせてこの値を変更します。
port.https	デフォルトは 443 です。 スーパーバイザー クラスタ に NSX-T ネットワークを使用しているために Envoy Ingress サービスが LoadBalancer タイプである場合は、この設定をデフォルトの 443 のままにします。 スーパーバイザー クラスタ に vDS ネットワークを使用しているために Envoy Ingress サービスが NodePort タイプの場合は、この値を Envoy ノード ポートと一致するように設定します。
clair.enabled	Clair イメージ スキャナは廃止され、Trivy が使用されます。構成ファイルでは両方とも有効になっています。 Clair の値を false に設定して無効にします。
persistence.persistentVolumeClaim. <component>.accessMode	この設定には、いくつかのインスタンスがあります。 デフォルトは ReadWriteOnce です。 ReadWriteMany は、今後のリリースでサポートされる予定です。
imageChartStorage.type	デフォルトは filesystem です。 必要に応じて変更し、使用しているストレージを構成します。
proxy	必要に応じて、Harbor 用にプロキシを構成します。プロキシを構成する場合は、デフォルトの noProxy 値が必要です。

7 データ値を使用してシークレットを作成します。

```
kubectl create secret generic harbor-data-values --from-file=values.yaml=harbor-data-values.yaml -n tanzu-system-registry
```

`secret/harbor-data-values` が `tanzu-system-registry` 名前空間に作成されます。これを確認するには、次のコマンドを実行します。

```
kubectl get secrets -n tanzu-system-registry
```

8 Harbor 拡張機能をデプロイします。

```
kubectl apply -f harbor-extension.yaml
```

成功すると、`app.kappctrl.k14s.io/harbor created` が表示されます。

9 Harbor アプリケーションのステータスを確認します。

```
kubectl get app harbor -n tanzu-system-registry
```

成功すると、ステータスが `Reconciling` から `Reconcile succeeded` に変わります。

```
NAME     DESCRIPTION   SINCE-DEPLOY   AGE
harbor   Reconciling   96s            98s
```

```
NAME     DESCRIPTION           SINCE-DEPLOY   AGE
harbor   Reconcile succeeded   39s            2m29s
```

ステータスが `Reconcile failed` の場合は、Harbor レジストリのデプロイのトラブルシューティングを参照してください。

10 Harbor 拡張機能の詳細情報を確認します。

```
kubectl get app harbor -n tanzu-system-registry -o yaml
```

11 Harbor デプロイ オブジェクトのステータスを確認します。

```
kubectl get deployments -n tanzu-system-registry
```

成功すると、次のデプロイが表示されます。

```
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
harbor-core           1/1     1            1           5m16s
harbor-jobservice     1/1     1            1           5m16s
harbor-notary-server  1/1     1            1           5m16s
harbor-notary-signer  1/1     1            1           5m16s
harbor-portal         1/1     1            1           5m16s
harbor-registry       1/1     1            1           5m16s
```

12 Harbor ポッドのステータスを確認します。

```
kubectl get pods -n tanzu-system-registry
```

```
NAME                                    READY   STATUS    RESTARTS   AGE
harbor-core-9cbf4b79d-gxvgx             1/1     Running   0          7m11s
harbor-database-0                       1/1     Running   0          7m11s
harbor-jobservice-6b656ccb95-lm47d      1/1     Running   0          7m11s
harbor-notary-server-8494c684db-gm7jf   1/1     Running   0          7m11s
harbor-notary-signer-6f96b549d4-dzcnm   1/1     Running   0          7m11s
harbor-portal-5b8f4ddbd-qdnp2           1/1     Running   0          7m11s
harbor-redis-0                          1/1     Running   0          7m11s
harbor-registry-688894c58d-72txm        2/2     Running   0          7m11s
harbor-trivy-0                          1/1     Running   0          7m11s
```

13 必要に応じて、Harbor のインストールのトラブルシューティングを行います。Harbor レジストリのデプロイのトラブルシューティングを参照してください。

LoadBalancer タイプの Envoy サービスを使用する Harbor 向けの DNS の構成（NSX-T ネットワーク）

前提条件の Envoy サービスが LoadBalancer を介して公開される場合は、ロード バランサの外部 IP アドレスを取得し、Harbor の FQDN の DNS レコードを作成します。

1 LoadBalancer タイプの Envoy サービスの `External-IP` アドレスを取得します。

```
kubectl get service envoy -n tanzu-system-ingress
```

返された `External-IP` アドレスが次の例のように表示されます。

```
NAME    TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)                      AGE
envoy   LoadBalancer   10.99.25.220   10.195.141.17   80:30437/TCP,443:30589/TCP   3h27m
```

または、次のコマンドを使用して `External-IP` アドレスを取得することもできます。

```
kubectl get svc envoy -n tanzu-system-ingress -o
jsonpath='{.status.loadBalancer.ingress[0]}'
```

2 Harbor 拡張機能のインストールを確認するには、次の例のように、ロード バランサの `External-IP` アドレスにマッピングされる Harbor および Notary の FQDN を指定してローカルの `/etc/hosts` ファイルを更新します。

```
127.0.0.1 localhost
127.0.1.1 ubuntu
# TKGS Harbor with Envoy Load Balancer IP
10.195.141.17 core.harbor.domain
10.195.141.17 core.notary.harbor.domain
```

3 Harbor 拡張機能のインストールを確認するために、Harbor にログインします。Harbor Web インターフェイスへのログインを参照してください。

4 Envoy サービスのロード バランサの `External-IP` アドレスを Harbor の FQDN と Notary の FQDN にマッピングする 2 つの CNAME レコードを DNS サーバに作成します。

5 外部 DNS 拡張機能をインストールします。外部 DNS サービス検出のための TKG 拡張機能のデプロイと管理を参照してください。

NodePort タイプの Envoy サービスを使用する Harbor 向けの DNS の構成（vDS ネットワーク）

前提条件の Envoy サービスが NodePort を介して公開される場合は、ワーカー ノードの仮想マシンの IP アドレスを取得し、Harbor の FQDN の DNS レコードを作成します。

注： NodePort を使用するには、`harbor-data-values.yaml` ファイルに正しい `port.https` 値を指定しておく必要があります。

1 クラスタがプロビジョニングされている vSphere 名前空間 にコンテキストを切り替えます。

```
kubectl config use-context VSPHERE-NAMESPACE
```

2 クラスタ内のノードを一覧表示します。

```
kubectl get virtualmachines
```

クラスタ ノードが次のように表示されます。

```
NAME                                          POWERSTATE    AGE
tkgs-cluster-X-control-plane-6dgln            poweredOn     6h7m
tkgs-cluster-X-control-plane-j6hq6            poweredOn     6h10m
tkgs-cluster-X-control-plane-xc25f            poweredOn     6h14m
tkgs-cluster-X-workers-9twdr-59bc54dc97-kt4cm poweredOn     6h12m
tkgs-cluster-X-workers-9twdr-59bc54dc97-pjptr poweredOn     6h12m
tkgs-cluster-X-workers-9twdr-59bc54dc97-t45mn poweredOn     6h12m
```

3 ワーカー ノードの 1 つを選択し、次のコマンドを使用して記述します。

```
kubectl describe virtualmachines tkgs-cluster-X-workers-9twdr-59bc54dc97-kt4cm
```

4 `Vm Ip: 10.115.22.43` などの仮想マシンの IP アドレスを見つけます。

5 Harbor 拡張機能のインストールを確認するには、次の例のように、ワーカー ノードの IP アドレスにマッピングされる Harbor および Notary の FQDN を指定してローカルの `/etc/hosts` ファイルを更新します。

```
127.0.0.1 localhost
127.0.1.1 ubuntu
# TKGS Harbor with Envoy NodePort
10.115.22.43 core.harbor.domain
10.115.22.43 core.notary.harbor.domain
```

6 Harbor 拡張機能のインストールを確認するために、Harbor にログインします。Harbor Web インターフェイスへのログインを参照してください。

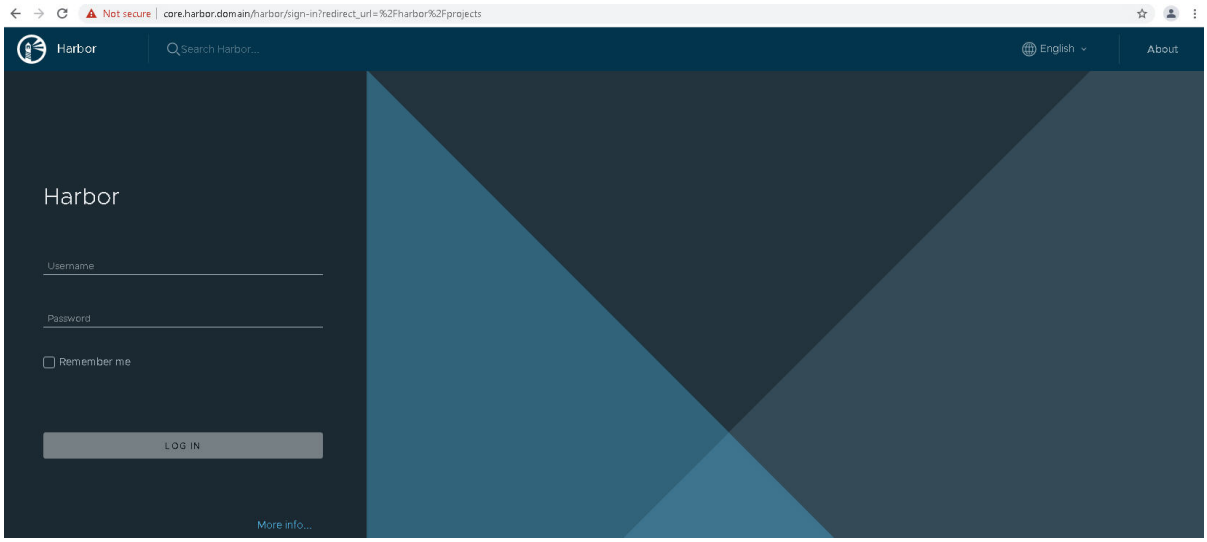
7 ワーカー ノードの IP アドレスを Harbor の FQDN と Notary の FQDN にマッピングする 2 つの CNAME レコードを DNS サーバに作成します。

8 外部 DNS 拡張機能をインストールします。外部 DNS サービス検出のための TKG 拡張機能のデプロイと管理を参照してください。

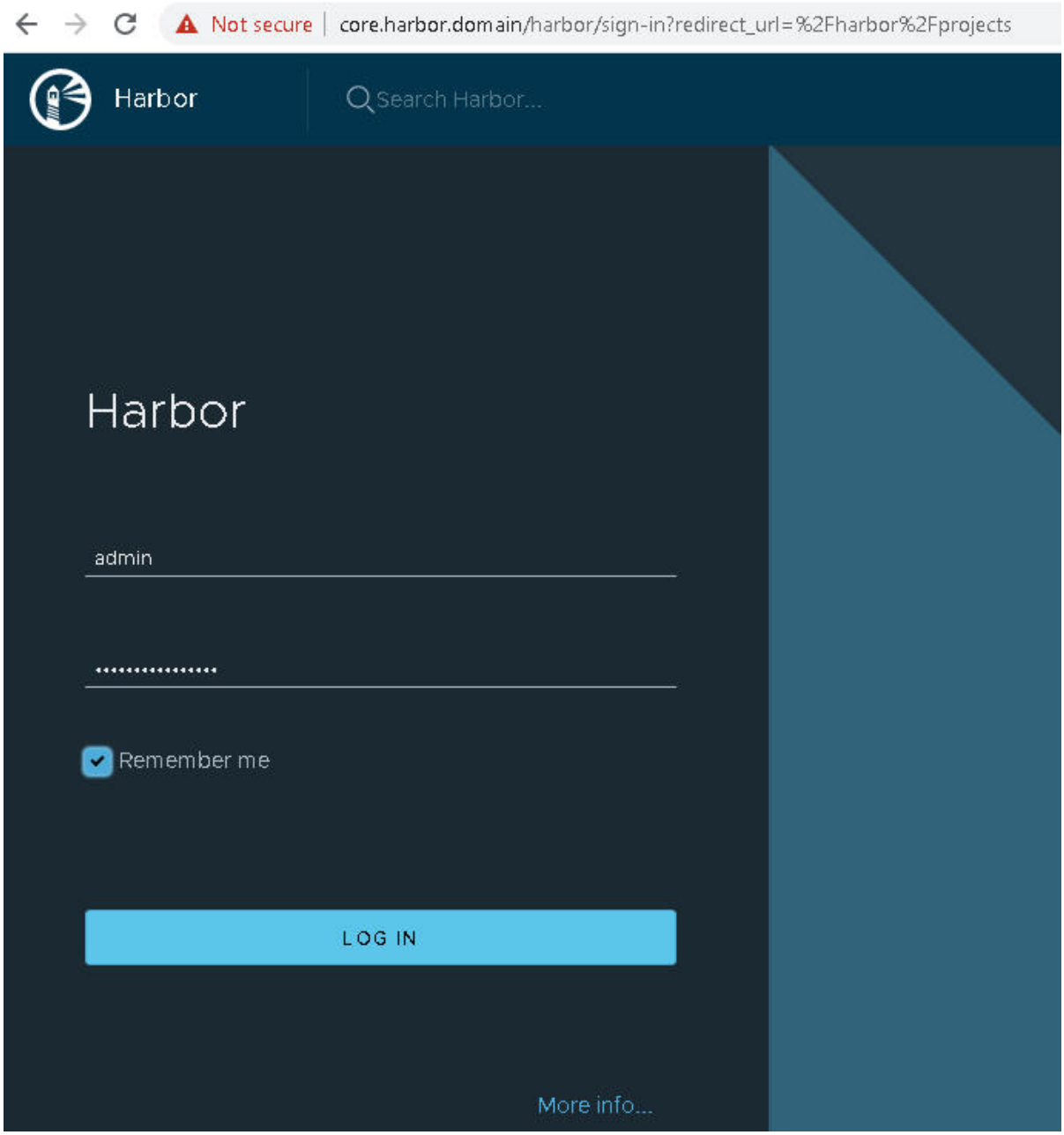
Harbor Web インターフェイスへのログイン

Harbor をインストールして構成したら、ログインして使用を開始します。

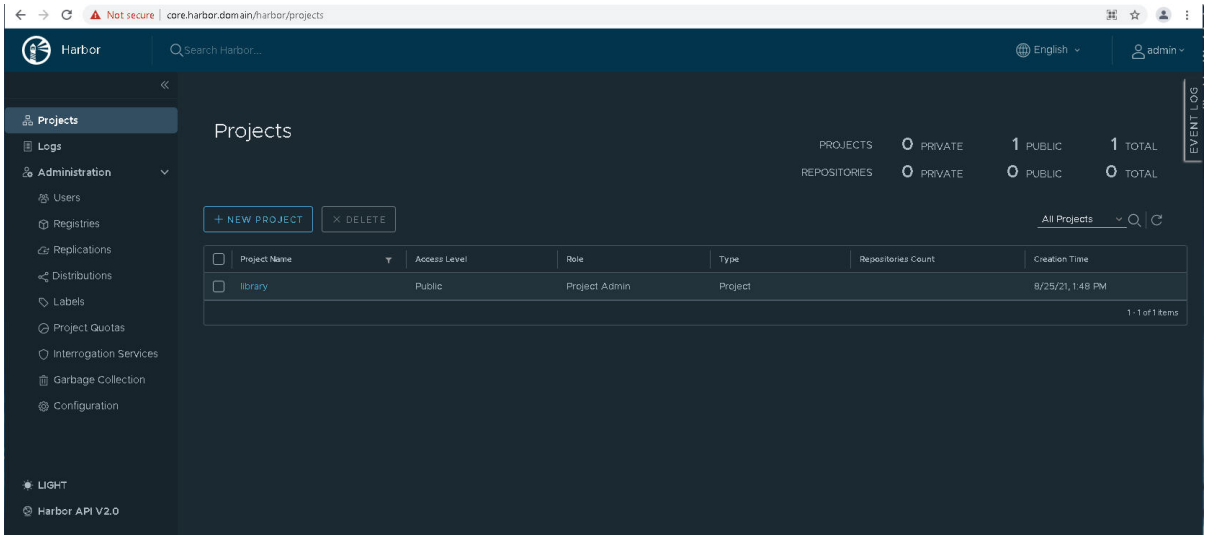
1 Harbor レジストリの Web インターフェイス (https://core.harbor.domain)、または使用したホスト名にアクセスします。



2 ユーザー名 **admin** と、生成され、`harbor-data-values.yaml` ファイルに入力したたパスワードを使用して Harbor にログインします。



3 Harbor ユーザー インターフェイスにアクセスできることを確認します。



4 Harbor CA 証明書を取得します。

Harbor インターフェイスで、[プロジェクト] - [ライブラリ] の順に選択するか、[新しいプロジェクト] を作成します。

[レジストリ証明書] をクリックして、Harbor CA 証明書 (`ca.crt`) をダウンロードします。

5 Harbor CA 証明書を Docker クライアントのトラスト ストアに追加して、Harbor レジストリとの間でコンテナ イメージをプッシュおよびプルできるようにします。組み込みの Harbor レジストリ 証明書を使用した Docker クライアントの構成を参照してください。

6 Harbor の使用方法の詳細については、Harbor のドキュメントを参照してください。

Harbor レジストリのデプロイのトラブルシューティング

デプロイまたは調整が失敗した場合は、`kubectl get pods -n tanzu-system-registry` を実行してポッドのステータスを確認します。`harbor` ポッドは、`Running` である必要があります。ポッドのステータスが `ImagePullBackOff` または `ImageCrashLoopBackOff` の場合は、コンテナ イメージをプルできていません。データ値と拡張機能 YAML ファイルに記述されているリポジトリ URL を確認し、これらが正確であることを確認します。

コンテナログを確認します。`name-XXXX` は、`kubectl get pods -A` を実行したときの一意のポッド名です。

```
kubectl logs pod/harbor-XXXXX -c harbor -n tanzu-system-registry
```

Harbor 拡張機能の更新

Tanzu Kubernetes クラスタにデプロイされている Contour 拡張機能を更新します。

1 シークレットから Harbor データ値を取得します。

```
kubectl get secret harbor-data-values -n tanzu-system-registry -o 'go-
template={{ index .data "values.yaml" }}' | base64 -d > harbor-data-values.yaml
```

2 `harbor-data-values.yaml` 内の Harbor データ値を更新します。

3 Harbor データ値のシークレットを更新します。

```
kubectl create secret generic harbor-data-values --from-file=values.yaml=harbor-data-values.yaml -n tanzu-system-registry -o yaml --dry-run | kubectl replace -f-
```

Harbor 拡張機能が新しいデータ値で調整されます。

注： デフォルトでは、kapp-controller によってアプリケーションが 5 分ごとに同期されます。更新は 5 分以内に有効になります。更新をすぐに有効にするには、`harbor-extension.yaml` 内の `syncPeriod` の値を小さくし、`kubectl apply -f harbor-extension.yaml` を使用して Contour 拡張機能を適用します。

4 拡張機能のステータスを確認します。

```
kubectl get app harbor -n tanzu-system-registry
```

Contour が更新されると、Contour アプリケーションのステータスが `Reconcile Succeeded` に変更されます。

5 詳細なステータスを確認して、トラブルシューティングを行います。

```
kubectl get app harbor -n tanzu-system-registry -o yaml
```

Harbor 拡張機能の削除

Tanzu Kubernetes クラスタから Harbor 拡張機能を削除します。

注： 手順を順番に実行します。Contour 拡張機能とアプリケーションを削除する前に、Contour 名前空間とロール オブジェクトを削除しないでください。Contour 名前空間とロール オブジェクトを削除すると、kapp-controller で使用されるサービス アカウントが削除されます。アプリケーションと拡張機能が削除される前にこのサービス アカウントを削除すると、システム エラーが発生する可能性があります。

1 Harbor の拡張機能ファイルをダウンロードしたディレクトリに移動します。

```
cd /extensions/registry/harbor/
```

2 Harbor アプリケーションを削除します。

```
kubectl delete app harbor -n tanzu-system-registry
```

予期される結果：

```
app.kappctrl.k14s.io "harbor" deleted
```

3 Harbor アプリケーションが削除されていることを確認します。

```
kubectl get app Harbor -n tanzu-system-registry
```

予期される結果：アプリケーションが `Not Found` になります。

```
apps.kappctrl.k14s.io "harbor" not found
```

4 レジストリ名前空間を削除します。

Harbor 拡張機能とアプリケーションが完全に削除されていることを確認した後でないと、名前空間とロール オブジェクトを安全に削除できません。

```
kubectl delete -f namespace-role.yaml
```

予期される結果：Harbor がデプロイされている名前空間と、関連付けられたロール ベースのアクセス コントロール オブジェクトが削除されます。

```
namespace "tanzu-system-registry" deleted
serviceaccount "harbor-extension-sa" deleted
role.rbac.authorization.k8s.io "harbor-extension-role" deleted
rolebinding.rbac.authorization.k8s.io "harbor-extension-rolebinding" deleted
clusterrole.rbac.authorization.k8s.io "harbor-extension-cluster-role" deleted
clusterrolebinding.rbac.authorization.k8s.io "harbor-extension-cluster-rolebinding" deleted
```

Harbor 拡張機能のアップグレード

既存の Harbor 拡張機能がデプロイされている場合は、最新バージョンにアップグレードできます。

1 Harbor の構成マップを取得します。

```
kubectl get configmap harbor -n tanzu-system-harbor -o 'go-template={{ index .data
"harbor.yaml" }}' > harbor-configmap.yaml
```

2 既存の Harbor 環境を削除します。Harbor 拡張機能の削除を参照してください。

3 Harbor 拡張機能をデプロイします。Harbor 拡張機能のデプロイを参照してください。

外部 DNS サービス検出のための TKG 拡張機能のデプロイと管理

外部 DNS を使用すると、Kubernetes ロード バランサ サービスに基づいて DNS レコードを動的に構成できます。外部 DNS 用 TKG 拡張機能をデプロイして、動的サービス検出をクラスタに提供できます。

拡張機能の前提条件

外部 DNS 用 TKG 拡張機能 v1.3.1 をデプロイする前に、次の要件を満たす必要があります。

- Tanzu Kubernetes クラスタをプロビジョニングします。TKGS v1alpha2 API を使用して Tanzu Kubernetes クラスタをプロビジョニングするためのワークフローを参照してください。
- Tanzu Kubernetes クラスタに接続します。vCenter Single Sign-On ユーザーとして Tanzu Kubernetes クラスタに接続を参照してください。
- kubectl を実行するクライアント ホストへの TKG 拡張機能 v1.3.1 バンドルのダウンロードを実行します。
- ターゲット Tanzu Kubernetes クラスタに TKG 拡張機能の前提条件のインストールします。

その他の要件

外部 DNS 拡張機能で提供されるサンプル構成には、Contour Ingress コントローラを使用する例および使用しない例が含まれています。Contour を使用する場合は、外部 DNS 拡張機能をインストールする前に Contour をインストールします。Contour Ingress 用 TKG 拡張機能のデプロイと管理を参照してください。

外部 DNS 拡張機能を使用すると、動的サービス検出が使用可能になります。一般的なユースケースは、Harbor レジストリで使用する場合です。Harbor を使用するには、AWS Route53、Azure DNS、Google Cloud DNS などの RFC 2136 準拠の動的 DNS プロバイダ、または BIND などのローカル DNS サーバを設定する必要があります。Harbor レジストリ用 TKG 拡張機能のデプロイと管理を参照してください。

外部 DNS 拡張機能のデプロイ

外部 DNS 用 TKG 拡張機能 v1.3.1 をインストールする前に、次の手順を実行します。

1 外部 DNS 拡張機能ファイルをダウンロードしたディレクトリに移動します。

```
cd /tkg-extensions-v1.3.1+vmware.1/extensions/service-discovery/external-dns
```

2 名前空間といくつかのロールベースのアクセス コントロール オブジェクトを作成し、外部 DNS 拡張機能で使用できるようにします。

```
kubectl apply -f namespace-role.yaml
```

このコマンドにより、名前空間 `tanzu-system-service-discovery` および関連する RBAC オブジェクトが作成されます。`kubect get ns` を実行して確認します。

3 データ値ファイルを作成します。サンプル データ値ファイルには、最低限必要な構成が含まれています。

AWS、Azure、および RFC 2136 準拠の動的 DNS プロバイダ用に、Contour Ingress がある場合とない場合のそれぞれについてデータ値ファイルのサンプルが用意されています。適切なサンプル ファイルを選んでコピーします。

たとえば、Contour ありで AWS Route 53 を使用している場合は、次のコマンドを実行します。

```
cp external-dns-data-values-aws-with-contour.yaml.example external-dns-data-values-aws-
with-contour.yaml
```

一方、Contour ありで Azure を使用している場合は、次のコマンドを実行します。

```
cp external-dns-data-values-azure-with-contour.yaml.example external-dns-data-values-azure-
with-contour.yaml
```

4 外部 DNS データ値を構成します。

たとえば、Azure DNS の構成は次のとおりです。`domain-filter` と `azure-resource-group` の値を指定します。

```
#@data/values
#@overlay/match-child-defaults missing_ok=True
---
externalDns:
  image:
    repository: projects.registry.vmware.com/tkg
  deployment:
    #@overlay/replace
    args:
    - --provider=azure
```

```
    - --source=service
    - --source=ingress
    - --domain-filter=my-zone.example.org #! zone where services are deployed
    - --azure-resource-group=my-resource-group #! Azure resource group
    #@overlay/replace
    volumeMounts:
    - name: azure-config-file
      mountPath: /etc/kubernetes
      readOnly: true
    #@overlay/replace
    volumes:
    - name: azure-config-file
      secret:
        secretName: azure-config-file
```

5 入力済みのデータ値ファイルを使用して汎用シークレットを作成します。

たとえば、次のコマンドは、Azure DNS データ値ファイルを使用してシークレットを作成します。

```
kubectl create secret generic external-dns-data-values --from-file=values.yaml=external-dns-data-values-azure-with-contour.yaml -n tanzu-system-service-discovery
```

`tanzu-system-service-discovery` 名前空間に `secret/external-dns-data-values created` が作成されていることを確認できます。これを確認するには、`kubectl get secrets -n tanzu-system-service-discovery` コマンドを使用します。

6 外部 DNS 拡張機能をデプロイします。

```
kubectl apply -f external-dns-extension.yaml
```

成功すると、`app.kappctrl.k14s.io/external-dns created` が表示されます。

7 拡張機能のデプロイのステータスを確認します。

```
kubectl get app external-dns -n tanzu-system-service-discovery
```

外部 DNS が正常にデプロイされると、アプリケーションのステータスが `Reconciling` から `Reconcile succeeded` に変わります。ステータスが `Reconcile failed` の場合は、デプロイのトラブルシューティングを参照してください。

8 詳細なステータスを確認します。

```
kubectl get app external-dns -n tanzu-system-service-discovery -o yaml
```

デプロイのトラブルシューティング

調整に失敗した場合は、コマンド `kubectl get pods -A` を実行してポッドのステータスを確認します。通常の条件下では、`external-dns-XXXXX` ポッドが `Running` であることを確認できます。調整に失敗するか、ポッドのステータスが `ImagePullBackOff` または `ImagecRashLoopBackOff` の場合、コンテナ イメージをリポジトリからプルできなかったことを意味します。データ値と拡張機能 YAML ファイルのリポジトリ URL を調べて、これらが正確であることを確認します。

コンテナ ログを確認するには、次のコマンドを実行します。`name-XXXX` は、`kubectl get pods -A` を実行すると表示される一意のポッド名です。

```
kubectl logs pod/external-dns-XXXXX -c external-dns -n tanzu-system-service-discovery
```

外部 DNS 拡張機能の更新

Tanzu Kubernetes クラスタにデプロイされている外部 DNS 拡張機能を更新します。

1 シークレットから Contour データ値を取得します。

```
kubectl get secret external-dns-data-values -n tanzu-system-service-discovery -o 'go-
template={{ index .data "values.yaml" }}' | base64 -d > external-dns-data-values.yaml
```

2 `external-dns-data-values.yaml` 内の外部 DNS データ値を更新します。外部 DNS 拡張機能の構成を参照してください。

3 Contour データ値のシークレットを更新します。

```
kubectl create secret generic external-dns-data-values --from-file=values.yaml=external-
dns-data-values.yaml -n tanzu-system-service-discovery -o yaml --dry-run | kubectl replace
-f-
```

外部 DSN 拡張機能が新しいデータ値で調整されます。

注： デフォルトでは、kapp-controller によってアプリケーションが 5 分ごとに同期されます。更新は 5 分以内に有効になります。更新をすぐに有効にするには、`external-dns-extension` 内の `syncPeriod` の値を小さくし、`kubectl apply -f external-dns-extension` を使用して Contour 拡張機能を適用します。

4 拡張機能のステータスを確認します。

```
kubectl get app external-dns -n tanzu-system-service-disovery
```

更新されると、アプリケーションのステータスが `Reconcile Succeeded` に変更されます。

5 詳細なステータスを確認して、トラブルシューティングを行います。

```
kubectl get app external-dns -n tanzu-system-service-disovery -o yaml
```

外部 DNS 拡張機能の削除

Tanzu Kubernetes クラスタから外部 DNS 拡張機を削除します。

注： 手順を順番に実行します。対象となる拡張機能とアプリケーションを削除する前に、その名前空間とロール オブジェクトを削除しないでください。名前空間とロール オブジェクトを削除すると、kapp-controller で使用されるサービス アカウントが削除されます。アプリケーションと拡張機能が削除される前にこのサービス アカウントを削除すると、システム エラーが発生する可能性があります。

1 拡張機能ファイルをダウンロードしたディレクトリに移動します。

```
cd /tkg-extensions-v1.3.1+vmware.1/extensions/service-discovery/external-dns
```

2 外部 DNS 拡張機能を削除します。

```
kubectl delete -f external-dns-extension.yaml
```

3 拡張機能が削除されていることを確認します。

```
kubectl get app contour -n tanzu-system-ingress
```

予期される結果：アプリケーションが `Not Found` になります。

4 名前空間を削除します。

Contour 拡張機能とアプリケーションが完全に削除されていることを確認した後でないと、名前空間とロール オブジェクトを安全に削除できません。

```
kubectl delete -f namespace-role.yaml
```

予期される結果：拡張機能がデプロイされている名前空間と、関連付けられたロール ベースのアクセス コントロール オブジェクトが削除されます。

外部 DNS 拡張機能の構成

外部 DSN 拡張機能をカスタム設定で構成できます。

外部 DNS プロバイダのデプロイ パラメータを構成します。その他のガイダンスについては、Kubernetes サイト https://github.com/kubernetes-sigs/external-dns#running-externaldns を参照してください。

表 14-10. Harbor 拡張機能構成パラメータ

パラメータ	説明	タイプ	デフォルト
externalDns.namespace	external-dns がデプロイされる名前空間	文字列	tanzu-system-service-discovery
externalDns.image.repository	external-dns image	文字列	projects.registry.vmware.com/tkg
externalDns.image.name	external-dns の名前	文字列	external-dns
externalDns.image.tag	ExternalDNS のイメージ タグ	文字列	v0.7.4_vmware.1
externalDns.image.pullPolicy	ExternalDNS のイメージ プル ポリシー	文字列	IfNotPresent
externalDns.deployment.annotations	external-dns のデプロイの注釈	map<string,string>	{}
externalDns.deployment.args	コマンドラインを介して external-dns に渡される引数	list<string>	`[]`（必須パラメータ）
externalDns.deployment.env	external-dns に渡す環境変数	list<string>	[]
externalDns.deployment.securityContext	external-dns コンテナのセキュリティ コンテキスト	セキュリティ コンテキスト	{}

表 14-10. Harbor 拡張機能構成パラメータ （続き）

パラメータ	説明	タイプ	デフォルト
externalDns.deployment.volumeMounts	external-dns コンテナのボリューム マウント	list<VolumeMount>	[]
externalDns.deployment.volumes	external-dns ポッドのボリューム	list<Volume>	[]

Tanzu Kubernetes クラスタへの AI/ML ワークロードのデプロイ

Tanzu Kubernetes Grid サービス によってプロビジョニングされた Tanzu Kubernetes クラスタに AI/ML ワークロードをデプロイできます。AI/ML ワークロードをデプロイするには、vSphere 管理者がいくつかの初期設定を行い、クラスタ オペレータがいくつかの構成を行う必要があります。vSphere with Tanzu 環境が vGPU に対応している場合、開発者は Kubernetes ワークロードと同じように AI/ML ワークロードを TKGS クラスタにデプロイできます。

TKGS クラスタへの AI/ML ワークロードのデプロイについて

vSphere with Tanzu および NVIDIA vGPU テクノロジーを使用して、TKGS クラスタに AI/ML ワークロードをデプロイできます。

AI/ML ワークロードの TGKS サポートについてのお知らせ

vSphere with Tanzu バージョン 7 Update 3 Monthly Patch 1 以降のリリースでは、計算を多用するワークロードを、Tanzu Kubernetes Grid サービス によってプロビジョニングされた Tanzu Kubernetes クラスタにデプロイできます。このコンテキストでは、計算を多用するワークロードとは、GPU アクセラレータ デバイスを使用する必要がある人工知能 (AI) または機械学習 (ML) アプリケーションを意味します。

Kubernetes 環境で AI/ML ワークロードの実行を容易にするために、VMware は NVIDIA と連携して、vSphere with Tanzu で NVIDIA GPU Cloud プラットフォームをサポートしています。つまり、NGC カタログ内のコンテナ イメージを、Tanzu Kubernetes Grid サービス によってプロビジョニングされた Tanzu Kubernetes クラスタにデプロイできます。

AI-Ready Enterprise 向けの NVIDIA と VMware の共通アーキテクチャの詳細については、Accelerating Workloads on vSphere 7 with Tanzu - A Technical Preview of Kubernetes Clusters with GPU を参照してください。

サポート対象の vGPU モード

TKGS に AI/ML ワークロードをデプロイするには、vSphere with Tanzu コンテンツ配信ネットワーク経由で入手可能な Ubuntu OVA を使用する必要があります。TKGS では、2 つの GPU 動作モードがサポートされています。1 つは vGPU で、もう 1 つは NIC パススルー機能を持つ vGPU です。次の表では、2 つのモードについて詳しく説明します。

モード	構成	説明
NVIDIA + TKGS + Ubuntu + vGPU	NVIDIA vGPU	GPU デバイスは、各 ESXi ホストにインストールされている NVIDIA ホスト マネージャ ドライバによって仮想化されます。仮想化された GPU デバイスは、複数の NVIDIA 仮想 GPU (vGPU) 間で共有されます。 各 NVIDIA vGPU は、GPU デバイスのメモリ容量によって定義されます。たとえば、GPU デバイスの RAM の合計容量が 32 GB の場合は、それぞれ約 4 GB のメモリを持つ 8 つの vGPU を作成できます。
NVIDIA + TKGS + Ubuntu + vGPU + NIC パススルー	NVIDIA vGPU [および] 動的 DirectPath I/O	NVIDIA vGPU プロファイルを構成した仮想マシン クラスに、動的 DirectPath IO を使用するパススルー ネットワーク デバイスのサポートを含めます。この場合、仮想マシンの配置は vSphere DRS によって決まります。

はじめに

TKGS 用に NVIDIA vGPU を構成するには、次のトピックを参照してください。

- TKGS クラスタ (vGPU) への AI/ML ワークロードのデプロイに関する vSphere 管理者ワークフロー
- TKGS クラスタへの AI/ML ワークロードのデプロイに関するクラスタ オペレータのワークフロー

NIC パススルー機能を持つ vGPU を使用している場合は、TKGS クラスタへの AI/ML ワークロードのデプロイに関する vSphere 管理者向けの補足（vGPU および動的 DirectPath I/O）のトピックも参照してください。

NVAIE アカウントに NVIDIA Delegated Licensing Server (DLS) を使用している場合は、TKGS クラスタ (DLS) への AI/ML ワークロードのデプロイに関するクラスタ オペレータについての補足のトピックも参照してください。

TKGS クラスタ (vGPU) への AI/ML ワークロードのデプロイに関する vSphere 管理者ワークフロー

開発者が TKGS クラスタに AI/ML ワークロードをデプロイできるようにするには、vSphere 管理者が NVIDIA vGPU ハードウェアをサポートするように vSphere with Tanzu 環境を設定します。

TKGS クラスタへの AI/ML ワークロードのデプロイに関する vSphere 管理者ワークフロー

次の表に、vSphere 管理者が TKGS クラスタに AI/ML ワークロードを展開できるようにするための大まかなワークフローを示します。その後で、各手順の詳細な手順を示します。

手順	操作	リンク
0	システム要件を確認します。	管理者の手順 0：システム要件を確認するを参照してください。
1	サポートされている NVIDIA GPU デバイスを ESXi ホストにインストールします。	管理者の手順 1：サポートされている NVIDIA GPU デバイスを ESXi ホストにインストールするを参照してください。
2	vGPU を操作するための ESXi デバイス グラフィック設定を構成します。	管理者の手順 2：各 ESXi ホストを vGPU 操作用に構成するを参照してください。
3	NVIDIA vGPU Manager (VIB) を各 ESXi ホストにインストールします。	管理者の手順 3：各 ESXi ホストに NVIDIA ホスト マネージャ ドライバをインストールするを参照してください。
4	NVIDIA ドライバの動作と GPU 仮想化モードを確認します。	管理者の手順 4：ESXi ホストで NVIDIA vGPU 操作の準備ができていることを確認するを参照してください。

手順	操作	リンク
5	GPU が構成されたクラスタでワークロード管理を有効にします。これにより、vGPU 対応の ESXi ホストでスーパーバイザー クラスタが実行されるようになります。	管理者の手順 5：vGPU が構成された vCenter Server クラスタでワークロード管理を有効にするを参照してください。
6	Tanzu Kubernetes リリース用のコンテンツ ライブラリを作成*または更新し、vGPU ワークロードに必要なサポート対象の Ubuntu OVA をライブラリにポピュレートします。	管理者の手順 6：Tanzu Kubernetes Ubuntu リリースを含むコンテンツ ライブラリを作成または更新するを参照してください。 **注：** *必要に応じて行います。TKGS クラスタの Photon イメージ用のコンテンツ ライブラリがすでにある場合は、Ubuntu イメージ用の新しいコンテンツ ライブラリを作成しないでください。
7	特定の vGPU プロファイルが選択されたカスタムの仮想マシン クラスを作成します。	管理者の手順 7：vGPU プロファイルを使用するカスタム仮想マシン クラスを作成するを参照してください
8	TKGS GPU クラスタ用の vSphere 名前空間を作成して構成します。パーシステント ボリュームの編集権限とストレージを持つユーザーを追加します。	管理者の手順 8：TKGS GPU クラスタの vSphere 名前空間を作成して構成するを参照してください
9	Ubuntu OVA および vGPU 用のカスタム仮想マシン クラスが含まれているコンテンツ ライブラリを、TGKS 用に作成した vSphere 名前空間に関連付けます。	管理者の手順 9：コンテンツ ライブラリと仮想マシン クラスを vSphere 名前空間に関連付けるを参照してください
10	スーパーバイザー クラスタがプロビジョニングされ、クラスタ オペレータがアクセスできることを確認します。	管理者の手順 10：スーパーバイザー クラスタにアクセスできることを確認するを参照してください

管理者の手順 0：システム要件を確認する

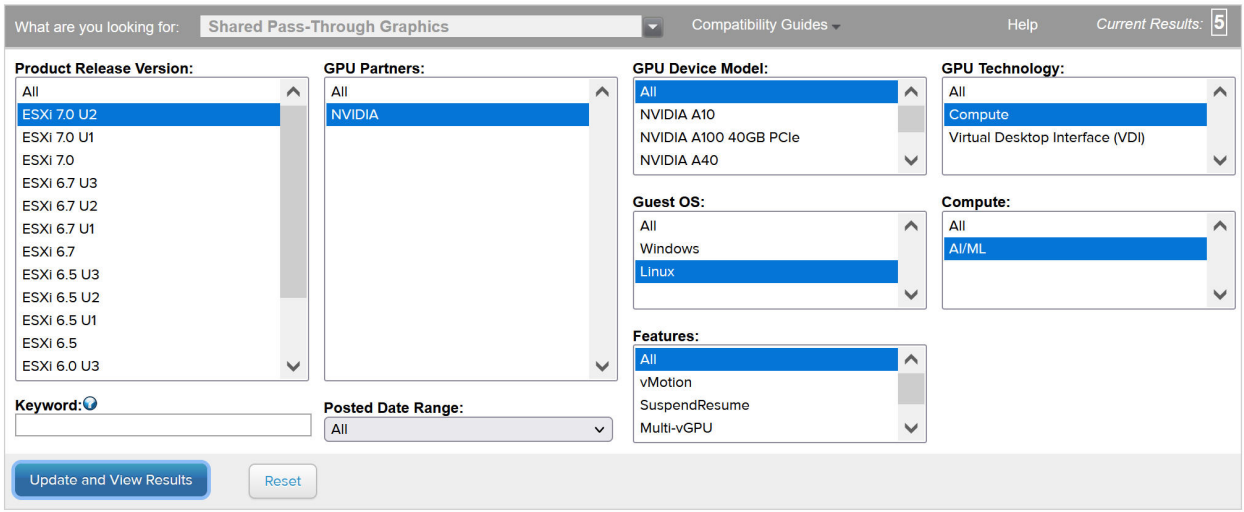
次のシステム要件を参照して、TKGS クラスタに AI/ML ワークロードをデプロイできるように環境を設定します。

要件	説明
vSphere インフラストラクチャ	vSphere 7 Update3 Monthly Patch 1 ESXi ビルド `18778458` 以降 vCenter Server ビルド `18644231` 以降
ワークロード管理	vSphere 名前空間のバージョン `0.0.11-18610518` 以降
スーパーバイザー クラスタ	スーパーバイザー クラスタ のバージョン `v1.21.0+vmware.1-vsc0.0.11-18610518` 以降
TKR Ubuntu OVA	Tanzu Kubernetes リリース Ubuntu `ob-18691651-tkgs-ova-ubuntu-2004-v1.20.8---vmware.1-tkg.2`
NVIDIA vGPU ホスト ドライバ	NGC Web サイトから VIB をダウンロードします。詳細については、vGPU ソフトウェア ドライバのドキュメントを参照してください。例： `NVIDIA-AIE_ESXi_7.0.2_Driver_470.51-1OEM.702.0.0.17630552.vib`
vGPU の NVIDIA ライセンス サーバ	組織から提供された FQDN

管理者の手順 1：サポートされている NVIDIA GPU デバイスを ESXi ホストにインストールする

TKGS に AI/ML ワークロードをデプロイするには、[ワークロード管理] を有効にする vCenter Server クラスタ内の各 ESXi ホストに、サポートされている NVIDIA GPU デバイスを 1 つ以上インストールします。

互換性のある NVIDIA GPU デバイスを表示するには、VMware 互換性ガイドを参照してください。



Click here to **Read** Important Support Information

Click on the 'GPU Device Model' to view more details and to subscribe to RSS feeds.

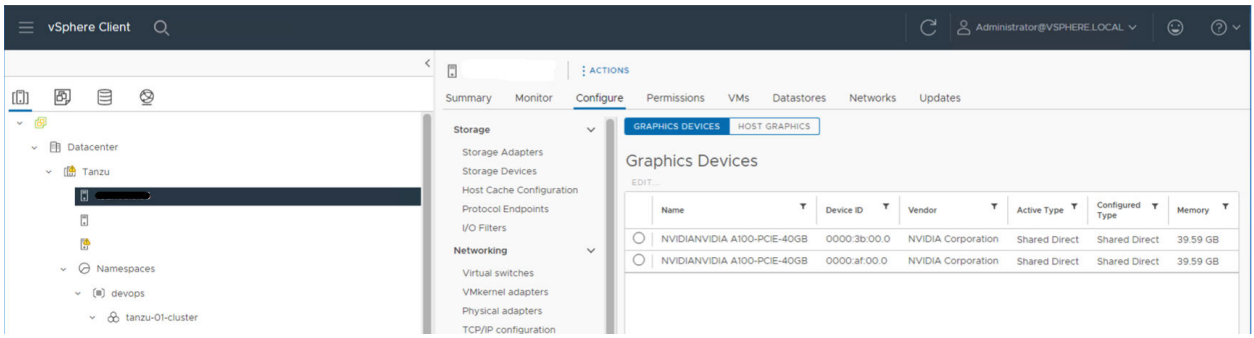
Bookmark | Print | Export to CSV

Search Results: Your search for" Shared Pass-Through Graphics " returned **5 results.** Back to Top Turn Off Auto Scroll Display: 10

GPU Partner	GPU Device Model	ESX Version	Compute
NVIDIA	NVIDIA A10	ESXi 7.0 U2	AI/ML
NVIDIA	NVIDIA A100 40GB PCIe	ESXi 7.0 U2	AI/ML
NVIDIA	NVIDIA A100 80GB PCIe	ESXi 7.0 U2	AI/ML
NVIDIA	NVIDIA A30	ESXi 7.0 U2	AI/ML
NVIDIA	NVIDIA A40	ESXi 7.0 U2	AI/ML

NVIDA GPU デバイスは、最新の NVIDIA AI Enterprise (NVAIE) vGPU プロファイルをサポートしている必要があります。ガイダンスについては、NVIDIA Virtual GPU Software Supported GPUs ドキュメントを参照してください。

たとえば、次の ESXi ホストには、2 つの NVIDIA GPU A100 デバイスがインストールされています。



管理者の手順 2：各 ESXi ホストを vGPU 操作用に構成する

[直接共有] および [SR-IOV] を有効にして、各 ESXi ホストを vGPU 用に構成します。

[各 ESXi ホストでの直接共有の有効化]

NVIDIA vGPU 機能をロック解除するには、[ワークロード管理] を有効にする vCenter Server クラスタ内の各 ESXi ホストで [直接共有] モードを有効にします。

[直接共有] を有効にするには、次の手順を実行します。その他のガイダンスについては、vSphere ドキュメントのグラフィック デバイスの構成を参照してください。

1 vSphere Client を使用して、vCenter Server にログインします。

2 vCenter Server クラスタで ESXi ホストを選択します。

3 [構成] - [ハードウェア] - [グラフィック] の順に選択します。

4 NVIDIA GPU アクセラレータ デバイスを選択します。

5 グラフィック デバイスの設定を [編集] します。

6 [直接共有] を選択します。

7 [[X.Org Server を再起動]] を選択します。

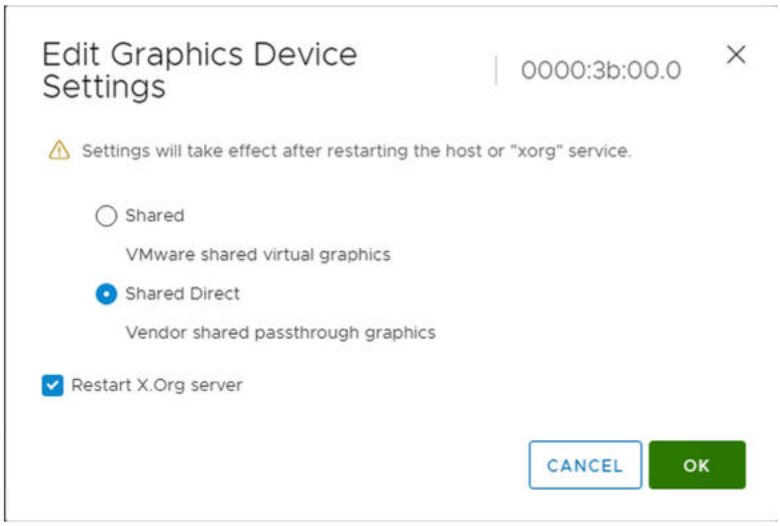
8 [OK] をクリックして構成を保存します。

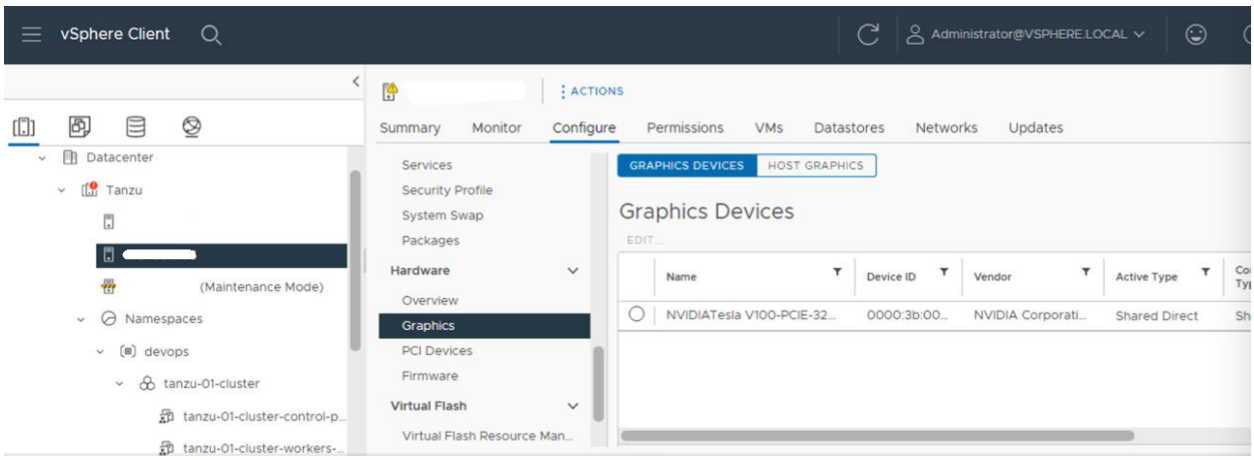
9 ESXi ホストを右クリックして、メンテナンス モードにします。

10 ホストを再起動します。

11 ホストが再実行されているときに、メンテナンス モードを終了します。

12 [ワークロード管理] を有効にする vCenter Server クラスタ内の ESXi ホストごとにこのプロセスを繰り返します。





[NVIDIA GPU A30 デバイスおよび A100 デバイスの SR-IOV BIOS の有効化]

マルチインスタンス GPU (MIG モード) で必要となる NVIDIA A30 デバイスまたは A100 GPU デバイスを使用している場合は、ESXi ホストで SR-IOV を有効にする必要があります。SR-IOV が有効になっていない場合は、Tanzu Kubernetes クラスタ ノード仮想マシンを起動できません。この問題が発生すると、[ワークロード管理] が有効になっている vCenter Server の [最近のタスク] ペインに次のエラー メッセージが表示されます。

```
Could not initialize plugin libnvidia-vgx.so for vGPU nvidia_aXXX-xx. Failed to start the virtual machine. Module DevicePowerOn power on failed.
```

SR-IOV を有効にするには、Web コンソールを使用して ESXi ホストにログインします。[管理] - [ハードウェア] の順に選択します。NVIDIA GPU デバイスを選択して、[SR-IOV の構成] をクリックします。ここで SR-IOV をオンにできます。その他のガイダンスについては、vSphere ドキュメントの Single Root I/O Virtualization (SR-IOV)を参照してください。

注： NIC パススルー機能を持つ vGPU を使用している場合、その他の ESXi 構成手順については、TKGS クラスタへの AI/ML ワークロードのデプロイに関する vSphere 管理者向けの補足（vGPU および動的 DirectPath I/O）のトピックを参照してください。

管理者の手順 3：各 ESXi ホストに NVIDIA ホスト マネージャ ドライバをインストールする

NVIDIA vGPU グラフィック アクセラレーションを使用して Tanzu Kubernetes クラスタ ノード仮想マシンを実行するには、[ワークロード管理] を有効にする vCenter Server クラスタ内の各 ESXi ホストに NVIDIA ホスト マネージャ ドライバをインストールします。

NVIDIA vGPU ホスト マネージャ ドライバのコンポーネントは、vSphere インストール バンドル (VIB) に含まれています。NVAIE VIB は、NVIDIA GRID ライセンス プログラムを通じて組織から提供されます。VMware は NVAIE VIB を提供することも、ダウンロード可能にすることもしません。NVIDIA ライセンス プログラムの一環として、ユーザーの組織がライセンス サーバを設定します。詳細については、NVIDIA 仮想 GPU ソフトウェア クイック スタート ガイドを参照してください。

NVIDIA 環境が設定されたら、各 ESXi ホストで次のコマンドを実行して、NVIDIA ライセンス サーバのアドレスと NVAIE VIB のバージョンを環境に適した値に置き換えます。その他のガイダンスについては、VMware サポートのナレッジベースの記事 Installing and configuring the NVIDIA VIB on ESXi を参照してください。

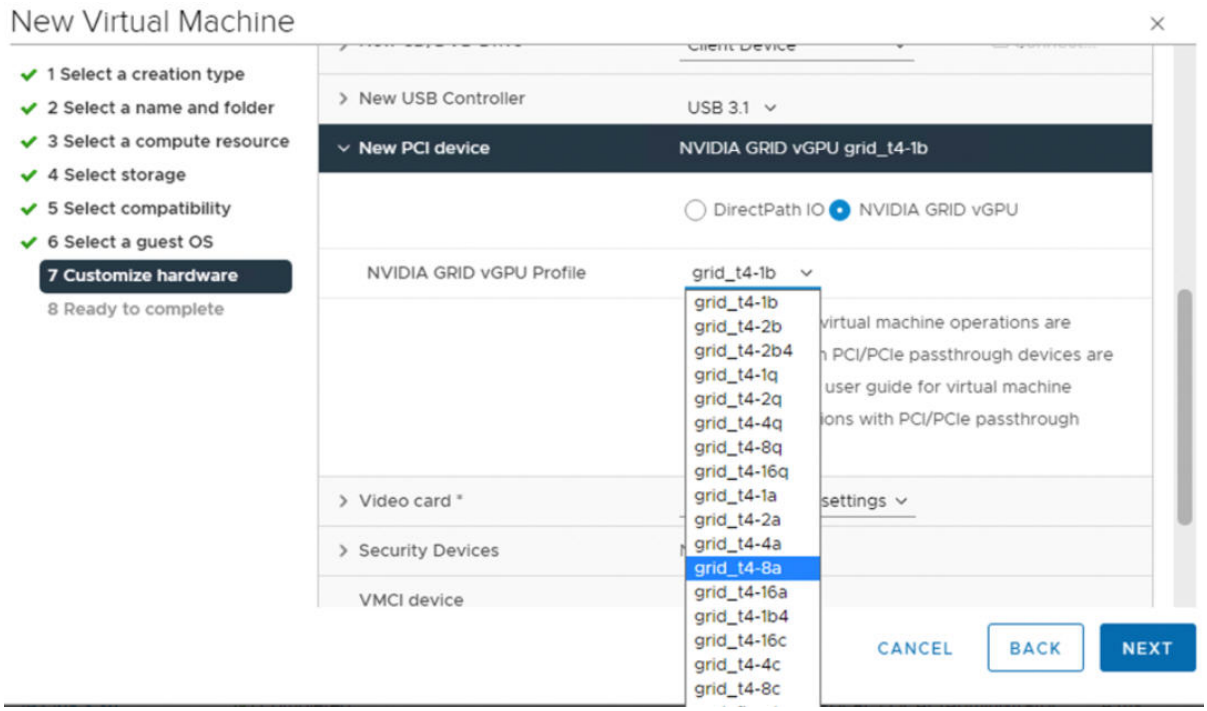
注： ESXi ホストにインストールされている NVAIE VIB のバージョンは、ノードの仮想マシンにインストールされている vGPU ソフトウェアのバージョンと一致する必要があります。以下のバージョンは単なる例です。

```
esxcli system maintenanceMode set --enable true
esxcli software vib install -v ftp://server.domain.example.com/nvidia/signed/
NVIDIA_bootbank_NVIDIA-VMware_ESXi_7.0_Host_Driver_460.73.02-1OEM.700.0.0.15525992.vib
esxcli system maintenanceMode set --enable false
/etc/init.d/xorg restart
```

管理者の手順 4：ESXi ホストで NVIDIA vGPU 操作の準備ができていることを確認する

各 ESXi ホストで NVIDIA vGPU 操作を行う準備ができていることを確認するには、[ワークロード管理] を有効にする vCenter Server クラスタ内の各 ESXi ホストで次のチェックを実行します

- ESXi ホストに SSH 接続を行い、シェル モードに切り替えて、コマンド `nvidia-smi` を実行します。NVIDIA システム管理インターフェイスは、NVIDA vGPU ホスト マネージャから提供されるコマンド ライン ユーティリティです。このコマンドを実行すると、ホスト上の GPU とドライバが返されます。
- 次のコマンドを実行して、NVIDIA ドライバが適切にインストールされていることを確認します。`esxcli software vib list | grep NVIDA`
- ホストに GPU の直接共有が構成され、SR-IOV がオンになっていることを確認します（NVIDIA A30 デバイスまたは A100 デバイスを使用している場合）。
- vSphere Client を使用して、GPU 用に構成されている ESXi ホストに PCI デバイスを含む新しい仮想マシンを作成します。NVIDIA vGPU プロファイルが表示されて、選択可能になります。



管理者の手順 5：vGPU が構成された vCenter Server クラスタでワークロード管理を有効にする

NVIDIA vGPU をサポートするように ESXi ホストを構成しました。これらのホストで構成される vCenter Server クラスタを作成します。[ワークロード管理] をサポートするには、vCenter Server クラスタが共有ストレージ、高可用性、完全自動化 DRS などの特定の要件を満たしている必要があります。

[ワークロード管理] を有効にするには、ネイティブ vSphere vDS ネットワークまたは NSX-T Data Center ネットワークのいずれかのネットワーク スタックを選択する必要もあります。Distributed Switch ネットワークを使用する場合は、ロード バランサとして NSX Advanced または HAProxy NSX のいずれかをインストールする必要があります。

[ワークロード管理] を有効にすると、スーパーバイザー クラスタ が vGPU 対応の ESXi ホストで実行されます。[ワークロード管理] を有効にする手順については、次のタスクとドキュメントを参照してください。

注： [ワークロード管理] が有効になっている vCenter Server クラスタがすでに存在する場合は、このクラスタが vGPU 用に構成された ESXi ホストを使用していると想定して、この手順をスキップします。

タスク	方法
[ワークロード管理] を有効にするための要件を満たす vCenter Server クラスタを作成します	vSphere クラスタで vSphere with Tanzu を構成するための前提条件
スーパーバイザー クラスタ のネットワーク（NSX-T または Distributed Switch）にロード バランサを構成します。	vSphere with Tanzu 用 NSX-T Data Center の構成。 vSphere ネットワークと vSphere with Tanzu 用 NSX Advanced Load Balancer の構成。 vSphere ネットワークと vSphere with Tanzu 用 HAProxy ロード バランサの構成。
[ワークロード管理] を有効にします。	NSX-T Data Center ネットワークを使用したワークロード管理の有効化。 vSphere ネットワークを使用したワークロード管理の有効化。

管理者の手順 6：Tanzu Kubernetes Ubuntu リリースを含むコンテンツ ライブラリを作成または更新する

GPU が構成された vCenter Server クラスタで [ワークロード管理] が有効になったら、次に、Tanzu Kubernetes リリースの OVA イメージを格納するコンテンツ ライブラリを作成します。

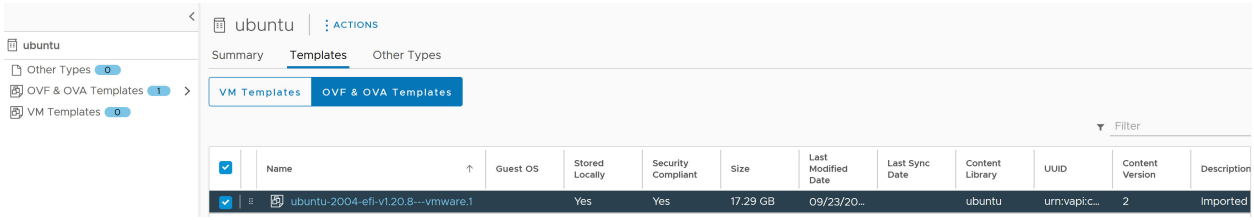
注意: Photon イメージで構成される Tanzu Kubernetes リリースを含むコンテンツ ライブラリがすでにある場合は、既存のコンテンツ ライブラリと必要な Ubuntu イメージの同期の実行のみを実行します。TKGS クラスタ用に 2 つ目のコンテンツ ライブラリを作成しないでください。作成すると、システムが不安定になる可能性があります。

NVIDIA vGPU には Ubuntu オペレーティング システムが必要です。VMware は、このような目的に使用する Ubuntu OVA を提供しています。vGPU クラスタに PhotonOS Tanzu Kubernetes リリースを使用することはできません。

このイメージを vSphere with Tanzu 環境にインポートするには、表に記載されているいずれかの方法を選択し、対応する手順に従います。

コンテンツ ライブラリ タイプ	説明
[サブスクライブ済みコンテンツ ライブラリ] を作成して、Ubuntu OVA を使用環境と自動的に同期します。	Tanzu Kubernetes リリース のサブスクライブ済みコンテンツ ライブラリの作成、セキュリティ保護、同期
[ローカル コンテンツ ライブラリ] を作成して、Ubuntu OVA を使用環境に手動でアップロードします。	Tanzu Kubernetes リリース 用のローカル コンテンツ ライブラリの作成、セキュリティ保護、同期

このタスクを完了すると、コンテンツ ライブラリで使用可能な Ubuntu OVA が表示されます。



管理者の手順 7：vGPU プロファイルを使用するカスタム仮想マシン クラスを作成する

次の手順では、vGPU プロファイルを使用するカスタム仮想マシン クラスを作成します。Tanzu Kubernetes クラスタ ノードが作成されるときに、このクラス定義が使用されます。

次の手順に従って、vGPU プロファイルを使用するカスタム仮想マシン クラスを作成します。その他のガイダンスについては、vSphere with Tanzu での仮想マシン クラスへの PCI デバイスの追加を参照してください。

注： NIC パススルー機能を持つ vGPU を使用している場合、その他の手順については、TKGS クラスタへの AI/ML ワークロードのデプロイに関する vSphere 管理者向けの補足（vGPU および動的 DirectPath I/O）のトピックを参照してください。

1 vSphere Client を使用して、vCenter Server にログインします。

2 [ワークロード管理] を選択します。

3 [[サービス]] を選択します。

4 [仮想マシン クラス] を選択します。

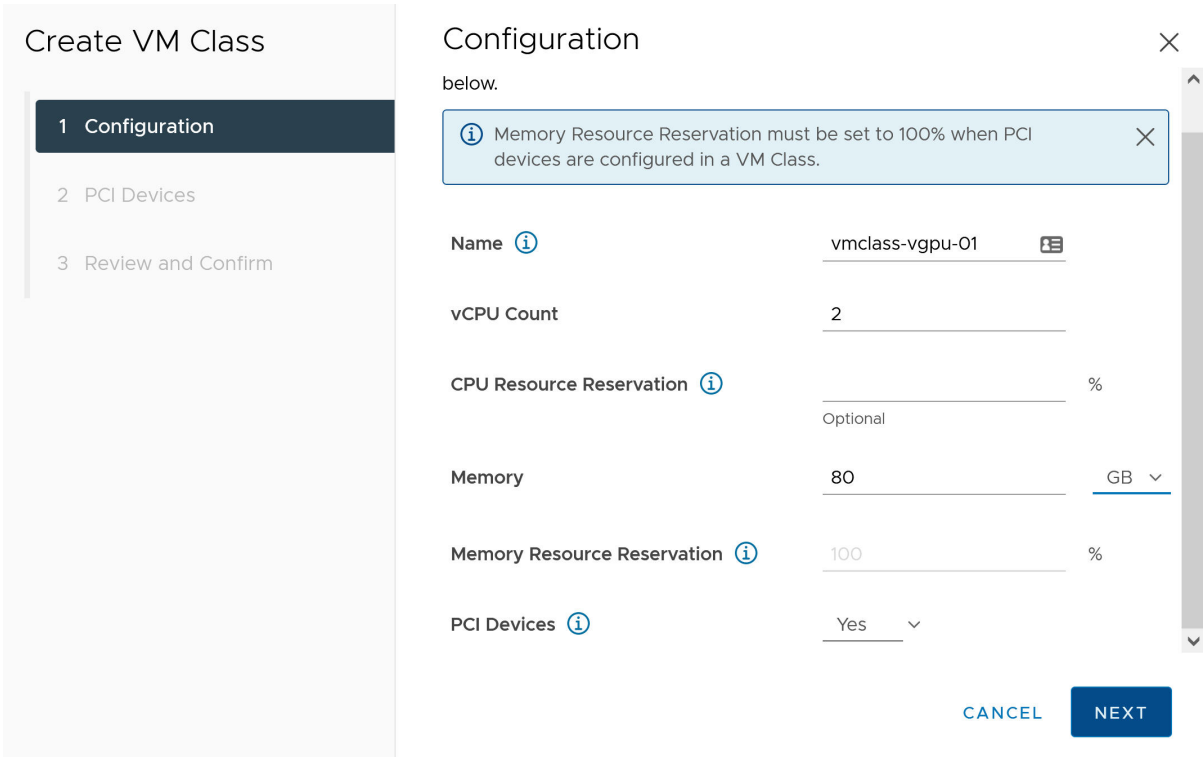
5 [仮想マシン クラスの作成] をクリックします。

6 [構成] タブで、カスタム仮想マシン クラスを構成します。

構成フィールド	説明
[名前]	カスタム仮想マシン クラスのわかりやすい名前を入力します（**`vmclass-vgpu-1`** など）。
[vCPU の数]	**2**
[CPU リソース予約]	オプション。空白のままにする場合は [OK] をクリックします。
[メモリ]	**80** [GB] など

構成フィールド	説明
[メモリ リソース予約]	[100%] (仮想マシン クラスで PCI デバイスが構成されている場合は必須)
[PCI デバイス]	[はい] **注：** PCI デバイスに対して [はい] を選択すると、GPU デバイスを使用していることがシステムに通知され、vGPU 構成をサポートするように仮想マシン クラスの構成が変更されます。

例：



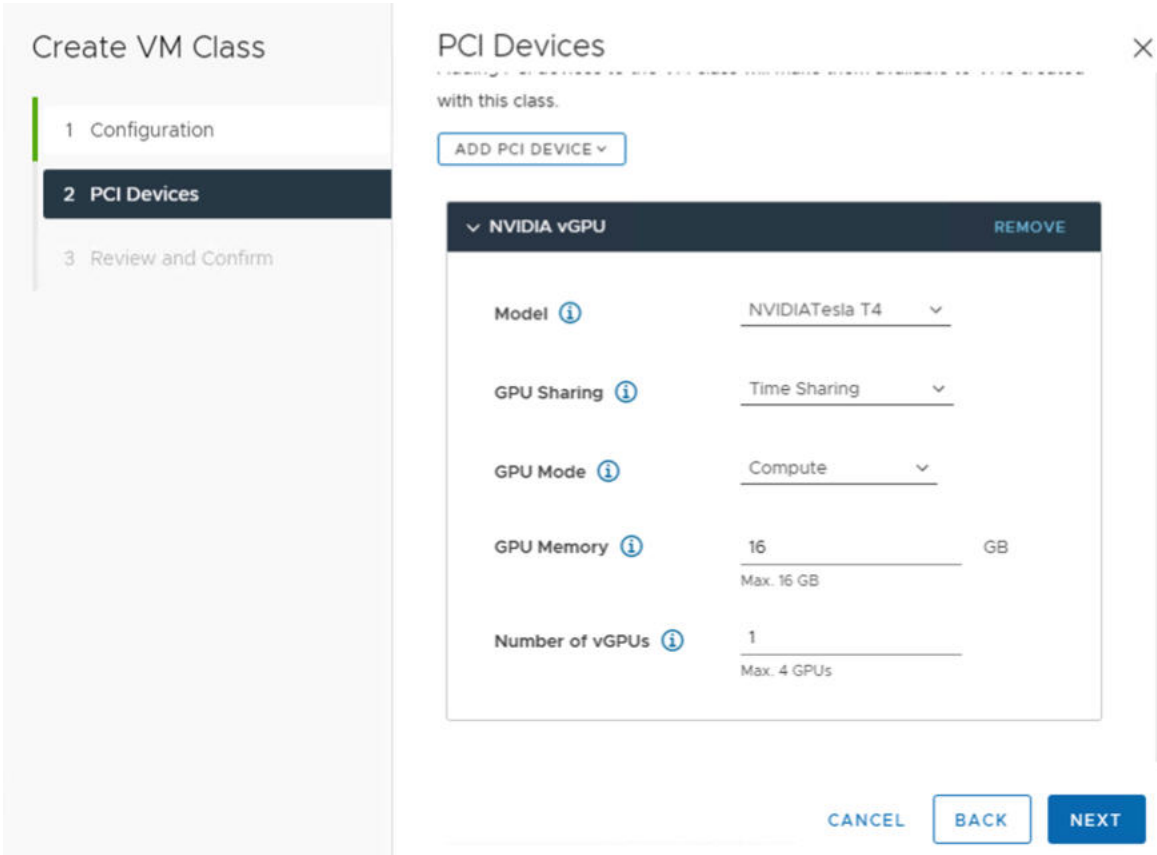
7 [次へ] をクリックします。

8 [PCI デバイス] タブで [PCI デバイスの追加] - [NVIDIA vGPU] オプションを選択します。

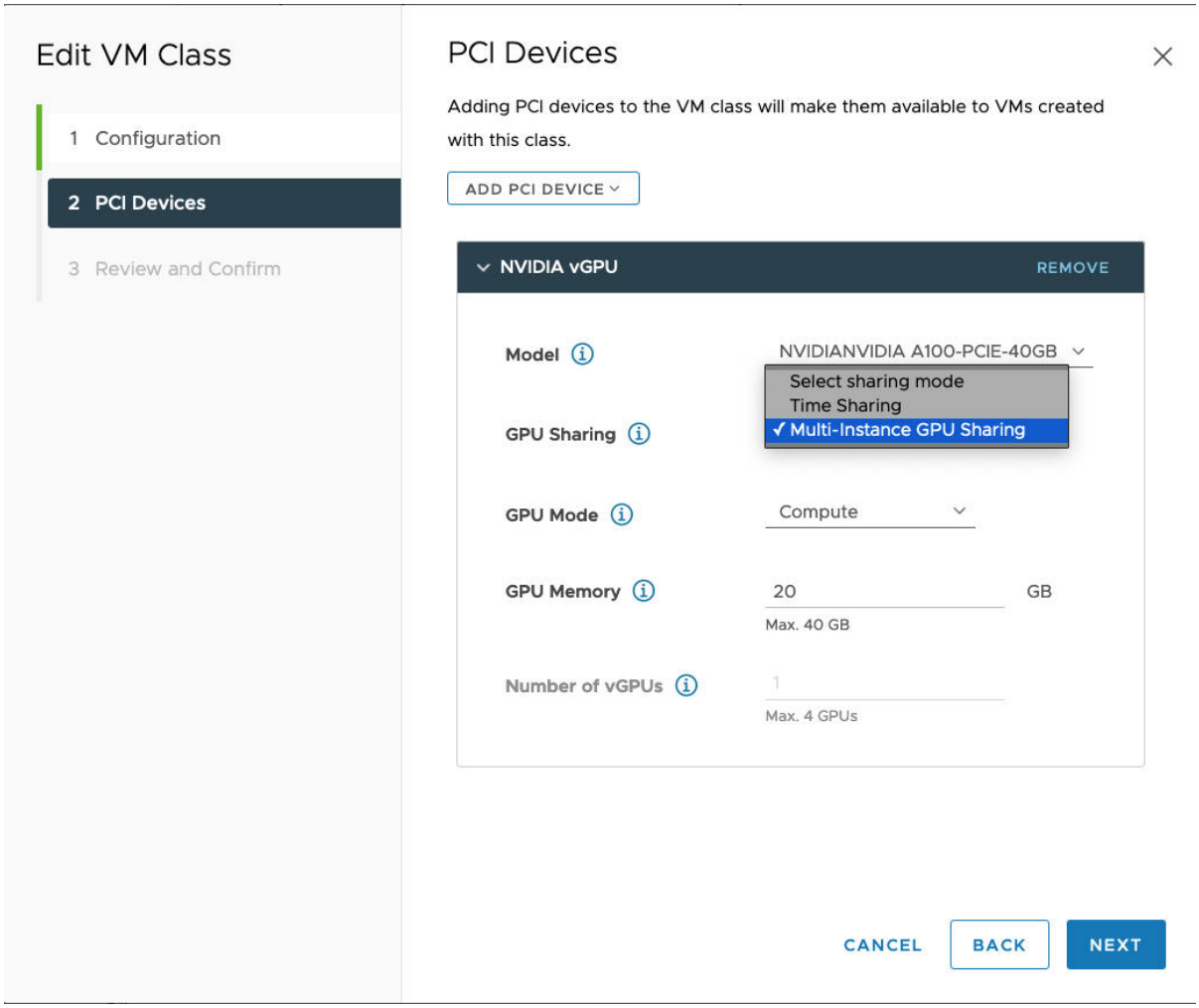
9 NVIDIA vGPU モデルを構成します。

NVIDIA vGPU フィールド	説明
[モデル]	[NVIDIA vGPU] - [モデル] メニューで使用可能なモデルの中から、NVIDIA GPU ハードウェア デバイス モデルを選択します。プロファイルが表示されない場合は、クラスタ内のどのホストも PCI デバイスをサポートしていません。
[GPU 共有]	この設定は、GPU 対応の仮想マシン間における GPU デバイスの共有方法を定義します。vGPU の実装には、[時刻の共有] と [マルチインスタンス GPU 共有]の 2 種類があります。 時刻の共有モードの場合、vGPU スケジューラは vGPU 間でパフォーマンスを調整するというベスト エフォート型の目標に従い、vGPU 対応の各仮想マシンの作業を一定期間、順番に実行するように GPU に指示します。 MIG モードの場合は、単一の GPU デバイスで複数の vGPU 対応仮想マシンを並列に実行できます。MIG モードは新しい GPU アーキテクチャに基づいていて、NVIDIA A100 デバイスおよび A30 デバイスでのみサポートされます。MIG オプションが表示されない場合、選択した PCI デバイスではサポートされていません。
[GPU モード]	[コンピューティング]
[GPU メモリ]	8 [GB] など
[vGPU の数]	[1] など

たとえば、時刻の共有モードで構成された NVIDIA vGPU プロファイルを次に示します。



たとえば、サポートされている GPU デバイスを使用して MIG モードで構成された NVIDIA vGPU プロファイルを次に示します。



10 [次へ] をクリックします。

11 選択内容を確認します。

12 [終了] をクリックします。

13 新しいカスタム仮想マシン クラスが仮想マシン クラスのリストで使用可能になっていることを確認します。

管理者の手順 8：TKGS GPU クラスタの vSphere 名前空間を作成して構成する

プロビジョニングする TKGS GPU クラスタごとに、vSphere 名前空間を作成します。編集権限を持つ vSphere SSO ユーザーを追加して名前空間を構成し、パーシステント ボリュームにストレージ ポリシーを適用します。

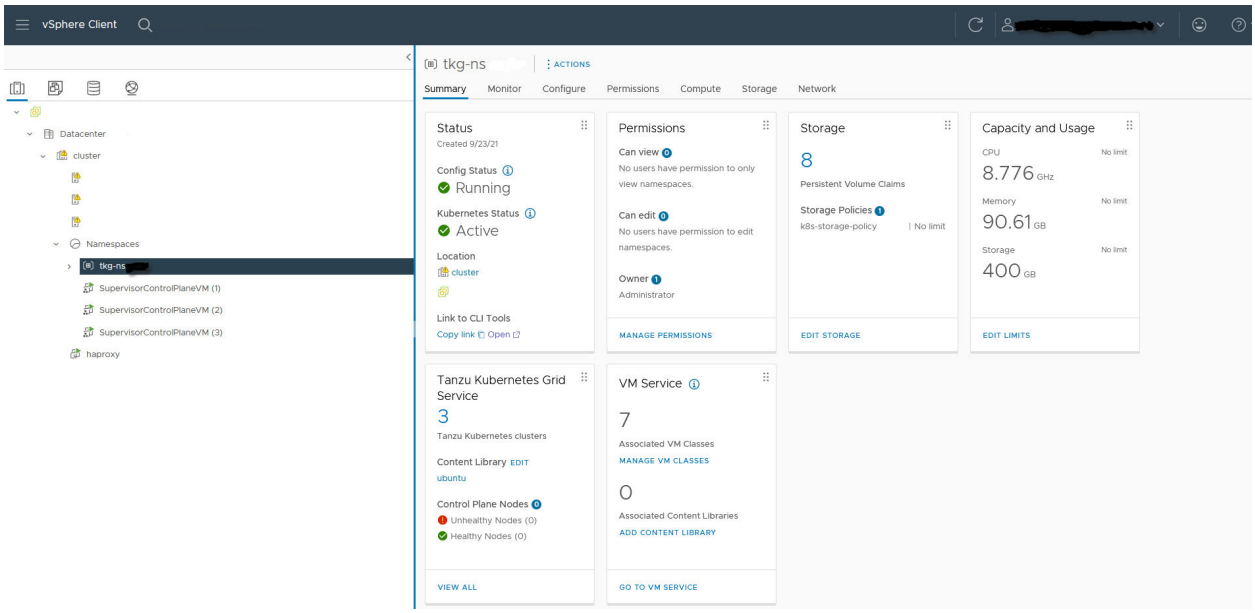
この操作を行うには、vSphere 名前空間 の作成と設定を参照してください。

管理者の手順 9：コンテンツ ライブラリと仮想マシン クラスを vSphere 名前空間に関連付ける

vSphere 名前空間を作成して構成したら、Ubuntu OVA を含むコンテンツ ライブラリを vSphere 名前空間に関連付けて、vGPU プロファイルを使用するカスタム仮想マシン クラスを同じ vSphere 名前空間に関連付けます。

タスク	説明
vGPU 用の Ubuntu OVA が含まれているコンテンツ ライブラリを、TKGS クラスタをプロビジョニングする vSphere 名前空間に関連付けます。	Tanzu Kubernetes リリース 用の vSphere 名前空間 の構成を参照してください。
vGPU プロファイルを使用するカスタム仮想マシン クラスを、TKGS クラスタをプロビジョニングする vSphere 名前空間に関連付けます。	仮想マシン クラスと vSphere with Tanzu の名前空間の関連付けを参照してください。

次の例では、vGPU クラスタで使用するためにコンテンツ ライブラリとカスタム仮想マシン クラスが関連付けられた、構成済みの vSphere 名前空間を示します。



管理者の手順 10：スーパーバイザー クラスタにアクセスできることを確認する

最後の管理タスクでは、スーパーバイザー クラスタ がプロビジョニングされ、クラスタ オペレータが AI/ML ワークロード用の TKGS クラスタをプロビジョニングする際に使用できることを確認します。

1 vSphere 向け Kubernetes CLI Tools をダウンロードしてインストールします。

vSphere 向け Kubernetes CLI Tools のダウンロードとインストールを参照してください。

2 スーパーバイザー クラスタ に接続します。

vCenter Single Sign-On ユーザーとして スーパーバイザー クラスタ に接続するを参照してください。

3 vSphere 向け Kubernetes CLI Tools をダウンロードするためのリンクと vSphere 名前空間の名前をクラスタ オペレータに提供します。

TKGS クラスタへの AI/ML ワークロードのデプロイに関するクラスタ オペレータのワークフローを参照してください。

TKGS クラスタへの AI/ML ワークロードのデプロイに関するクラスタ オペレータのワークフロー

開発者が TKGS クラスタに AI/ML ワークロードをデプロイできるようにするには、クラスタ オペレータとして、NVIDIA vGPU の動作をサポートするように Kubernetes 環境を構成します。

TKGS クラスタへの AI/ML ワークロードのデプロイに関するクラスタ オペレータのワークフロー

TKGS クラスタに AI/ML ワークロードをデプロイする大まかな手順は、次のとおりです。

手順	操作	リンク
0	システム要件を確認します。	Operator の手順 0：システム要件を確認するを参照してください。
1	kubectl と kubectl 向け vSphere プラグインをローカル ワークステーションにダウンロードします。	Operator の手順 1：ワークステーションに vSphere 向け Kubernetes CLI Tools をインストールするを参照してください。
2	kubectl を使用してスーパーバイザー クラスタにログインします。これにより、.kube/config に新しいスーパーバイザー クラスタのコンテキストが入力されます。	Operator の手順 2：スーパーバイザー クラスタにログインするを参照してください。
3	kubectl を使用して、コンテキストを vSphere 名前空間に切り替えます。	Operator の手順 3：コンテキストを vSphere 名前空間に切り替えるを参照してください。
4	kubectl を使用して仮想マシン クラスを一覧表示し、NVIDIA vGPU 対応クラスが含まれていることを確認します。	Operator の手順 4：vGPU ワークロード用のカスタム仮想マシン クラスを取得するを参照してください。
5	kubectl を使用して、使用可能な Tanzu Kubernetes リリースを一覧表示し、Ubuntu イメージが含まれていることを確認します。	Operator の手順 5：GPU ノード用の Ubuntu Tanzu Kubernetes リリースを取得するを参照してください。
6	GPU 対応 TKGS クラスタをプロビジョニングするための YAML 仕様を作成し、TKR のバージョンと仮想マシン クラスを指定します。	Operator の手順 6：vGPU 対応 TKGS クラスタをプロビジョニングするための YAML を作成するを参照してください。
7	TKGS クラスタをプロビジョニングします。	Operator の手順 7：TKGS クラスタをプロビジョニングするを参照してください。
8	クラスタにログインして、プロビジョニングを確認します。	Operator の手順 8：TKGS クラスタにログインしてプロビジョニングを確認するを参照してください。
9	名前空間、ロール バインド、イメージ シークレット、ライセンス構成マップなど、前提条件となるオブジェクトを TKGS クラスタにいくつか作成して、NVAIE GPU Operator をインストールする準備を整えます。	Operator の手順 9：NVAIE GPU Operator のインストール準備を行うを参照してください。
10	NVAIE GPU Operator をクラスタにインストールします。	Operator の手順 10：NVIDIA GPU Operator をクラスタにインストールするを参照してください。
11	AI/ML ワークロードを vGPU 対応 TKGS クラスタにデプロイします。	Operator の手順 11：AI/ML ワークロードをデプロイするを参照してください。

Operator の手順 0：システム要件を確認する

次のシステム要件を参照して、TKGS クラスタに AI/ML ワークロードをデプロイできるように環境を設定します。

要件	説明
vSphere 管理者が NVIDIA vGPU の環境を設定している	TKGS クラスタ (vGPU) への AI/ML ワークロードのデプロイに関する vSphere 管理者ワークフローを参照してください
TKR Ubuntu OVA	Tanzu Kubernetes リリース Ubuntu `ob-18691651-tkgs-ova-ubuntu-2004-v1.20.8---vmware.1-tkg.2`

要件	説明
TKG クラスタ プロビジョナ	Tanzu Kubernetes Grid サービス API のバージョン：`run.tanzu.vmware.com/v1alpha2`
NVIDIA GPU Operator	GPU Operator v1.8.0
NVIDIA GPU ドライバ コンテナ	`nvcr.io/nvstating/cnt-ea/driver:470.51-ubuntu20.04`

Operator の手順 1：ワークステーションに vSphere 向け Kubernetes CLI Tools をインストールする

vSphere 向け Kubernetes CLI Tools をダウンロードしてインストールします。

Linux を使用している場合は、次のコマンドを実行してツールをダウンロードできます。

```
curl -LOk https://${SC_IP}/wcp/plugin/linux-amd64/vsphere-plugin.zip
unzip vsphere-plugin.zip
mv -v bin/* /usr/local/bin/
```

その他のガイダンスについては、vSphere 向け Kubernetes CLI Tools のダウンロードとインストールを参照してください。

Operator の手順 2：スーパーバイザー クラスタにログインする

kubectl 向けの vSphere プラグイン を使用して、スーパーバイザー クラスタ での認証を行います。

```
kubectl vsphere login --server=IP-ADDRESS --vsphere-username USERNAME
```

Operator の手順 3：コンテキストを vSphere 名前空間に切り替える

kubectl を使用して、vSphere 管理者が TKGS GPU クラスタ用に作成した vSphere 名前空間 にコンテキストを切り替えます。

```
kubectl config get-contexts
```

```
kubectl config use-context TKGS-GPU-CLUSTER-NAMESPACE
```

Operator の手順 4：vGPU ワークロード用のカスタム仮想マシン クラスを取得する

vSphere 管理者が作成した vGPU プロファイルを使用するカスタム仮想マシン クラスがターゲットの vSphere 名前空間 で使用可能なことを確認します。

```
kubectl get virtualmachineclassbindings
```

注： 仮想マシン クラスをターゲット vSphere 名前空間 にバインドする必要があります。vGPU ワークロードのカスタム仮想マシン クラスが表示されない場合は、vSphere 管理者に確認してください。

Operator の手順 5：GPU ノード用の Ubuntu Tanzu Kubernetes リリースを取得する

vSphere 管理者がコンテンツ ライブラリから同期した必須の Ubuntu Tanzu Kubernetes リリースが vSphere 名前空間 で使用可能なことを確認します。

```
kubectl get tanzukubernetesreleases
```

または、次のショートカットを使用します。

```
kubectl get tkr
```

Operator の手順 6：vGPU 対応 TKGS クラスタをプロビジョニングするための YAML を作成する

Tanzu Kubernetes クラスタをプロビジョニングするための YAML ファイルを作成します。

次のいずれかの例から開始します。前述のコマンドの出力から収集した情報を使用して、クラスタの仕様をカスタマイズします。Tanzu Kubernetes クラスタをプロビジョニングするための TKGS v1alpha2 API で構成パラメータの完全なリストを参照してください。

例 1 では、2 つのワーカー ノード プールを指定します。

```
apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TanzuKubernetesCluster
metadata:
   #cluster name
   name: tkgs-cluster-gpu-a100
   #target vsphere namespace
   namespace: tkgs-gpu-operator
spec:
   topology:
     controlPlane:
       replicas: 3
       #storage class for control plane nodes
       #use `kubectl describe storageclasses`
       #to get available pvcs
       storageClass: vwt-storage-policy
       vmClass: guaranteed-medium
       #TKR NAME for Ubuntu ova supporting GPU
       tkr:
         reference:
           name: 1.20.8---vmware.1-tkg.1
     nodePools:
     - name: nodepool-a100-primary
       replicas: 3
       storageClass: vwt-storage-policy
       #custom VM class for vGPU
       vmClass: class-vgpu-a100
       #TKR NAME for Ubuntu ova supporting GPU
       tkr:
         reference:
           name: 1.20.8---vmware.1-tkg.1
     - name: nodepool-a100-secondary
       replicas: 3
```

```
        vmClass: class-vgpu-a100
        storageClass: vwt-storage-policy
        #TKR NAME for Ubuntu ova supporting GPU
        tkr:
          reference:
            name: 1.20.8---vmware.1-tkg.1
    settings:
      storage:
        defaultClass: vwt-storage-policy
      network:
        cni:
         name: antrea
        services:
         cidrBlocks: ["198.51.100.0/12"]
        pods:
         cidrBlocks: ["192.0.2.0/16"]
        serviceDomain: managedcluster.local
```

例 2 では、ワーカー ノード上にある容量が 50 GiB の別のボリュームをコンテナ ランタイム用に指定します。この設定は構成可能です。コンテナベースの AI/ML ワークロードには、適切なサイズの別のボリュームを使用することを推奨します。

```
apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TanzuKubernetesCluster
metadata:
  name: tkc
  namespace: tkg-ns-auto
spec:
  distribution:
    fullVersion: v1.20.8+vmware.1-tkg.1
  topology:
    controlPlane:
      replicas: 3
      storageClass: vwt-storage-policy
      tkr:
        reference:
          name: v1.20.8---vmware.1-tkg.1
      vmClass: best-effort-medium
    nodePools:
    - name: workers
      replicas: 3
      storageClass: k8s-storage-policy
      tkr:
        reference:
          name: v1.20.8---vmware.1-tkg.1
      vmClass: vmclass-vgpu
      volumes:
      - capacity:
          storage: 50Gi
        mountPath: /var/lib/containerd
        name: containerd
      - capacity:
          storage: 50Gi
        mountPath: /var/lib/kubelet
```

```
        name: kubelet
    - name: nodepool-1
      replicas: 1
      storageClass: vwt-storage-policy
      vmClass: best-effort-medium
```

例 3 には、ラベルなどの他のクラスタ メタデータが含まれています。

```
apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TanzuKubernetesCluster
metadata:
  annotations:
  labels:
    run.tanzu.vmware.com/tkr: v1.20.8---vmware.1-tkg.1
  name: tkgs-gpu-direct-rdma
  namespace: tkgs-ns
spec:
  settings:
    network:
      cni:
        name: antrea
      pods:
        cidrBlocks:
        - 192.168.0.0/16
      serviceDomain: cluster.local
      services:
        cidrBlocks:
        - 10.96.0.0/12
  topology:
    controlPlane:
      replicas: 3
      storageClass: tkgs-storage-policy
      vmClass: guaranteed-medium
      tkr:
        reference:
          name: v1.20.8---vmware.1-tkg.1
    nodePools:
    - name: workers
      replicas: 5
      storageClass: tkgs-storage-policy
      vmClass: claire-gpu-direct-rdma
      volumes:
      - capacity:
          storage: 50Gi
        mountPath: /var/lib/containerd
        name: containerd
      - capacity:
          storage: 50Gi
        mountPath: /var/lib/kubelet
        name: kubelet
      tkr:
        reference:
          name: v1.20.8---vmware.1-tkg.1
```

Operator の手順 7：TKGS クラスタをプロビジョニングする

次の kubectl コマンドを実行して、クラスタをプロビジョニングします。

```
kubectl apply -f CLUSTER-NAME.yaml
```

例：

```
kubectl apply -f tkgs-gpu-cluster-1.yaml
```

kubectl を使用して、クラスタ ノードのデプロイを監視します。

```
kubectl get tanzukubernetesclusters -n NAMESPACE
```

Operator の手順 8：TKGS クラスタにログインしてプロビジョニングを確認する

kubectl 向けの vSphere プラグイン を使用して、TKGS クラスタにログインします。

```
kubectl vsphere login --server=IP-ADDRESS --vsphere-username USERNAME \
--tanzu-kubernetes-cluster-name CLUSTER-NAME --tanzu-kubernetes-cluster-namespace NAMESPACE-
NAME
```

次のコマンドを使用して、クラスタを確認します。

```
kubectl cluster-info
```

```
kubectl get nodes
```

```
kubectl get namespaces
```

```
kubectl api-resources
```

Operator の手順 9：NVAIE GPU Operator のインストール準備を行う

GPU Operator と NVIDIA AI Enterprise をインストールする前に、プロビジョニングした TKGS クラスタに次のタスクを実行します。その他のガイダンスについては、NVAIE ドキュメントの「Prerequisite Tasks」を参照してください。

注： NVIDIA Delegated Licensing Server (DLS) を使用している場合の手順については、TKGS クラスタ (DLS) への AI/ML ワークロードのデプロイに関するクラスタ オペレータについての補足のトピックを参照してください。

1 Kubernetes 名前空間 `gpu-operator-resources` を作成します。ベスト プラクティスとして、常にこの名前空間にすべてをデプロイするようにします。

```
kubectl create ns gpu-operator-resources
```

2 ロール バインドを作成します。

Tanzu Kubernetes クラスタではポッド セキュリティ ポリシーが有効になっています。

rolebidings.yaml を作成します。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: psp:vmware-system-privileged:default
  namespace: default
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: psp:vmware-system-privileged
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: system:nodes
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: system:serviceaccounts
```

このロール バインドを適用します。

```
kubectl apply -f rolebindings.yaml
```

post-rolebindings.yaml を作成します。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: psp:vmware-system-privileged:gpu-operator-resources
  namespace: gpu-operator-resources
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: psp:vmware-system-privileged
subjects:
- kind: Group
  apiGroup: rbac.authorization.k8s.io
  name: system:serviceaccounts
```

このロール バインドを適用します。

```
kubectl apply -f post-rolebindings.yaml
```

3 Docker が NVIDIA GPU Cloud カタログからコンテナ イメージをプルする場合に使用できる NGC 認証情報を使用して、イメージ シークレットを作成します。

```
kubectl create secret docker-registry registry-secret \
     --docker-server=server-name --docker-username='$oauthtoken' \
     --docker-password=<place_holder> \
     --docker-email=email-name -n gpu-operator-resources
```

4 NVIDIA ライセンス サーバの構成マップを作成します。

```
kubectl create configmap licensing-config -n gpu-operator-resources --from-file=gridd.conf
```

`gridd.conf` は、NVIDIA ライセンス サーバのアドレスを参照します。次に例を示します。

```
# Description: Set License Server Address
# Data type: string
# Format:  "<address>"
ServerAddress=<place_holder>
```

Operator の手順 10：NVIDIA GPU Operator をクラスタにインストールする

NVAIE GPU Operator バージョン 1.8.0 を TKGS クラスタにインストールします。その他のガイダンスについては、GPU Operator のドキュメントを参照してください。

注： NVIDIA Delegated Licensing Server (DLS) を使用している場合の手順については、TKGS クラスタ (DLS) への AI/ML ワークロードのデプロイに関するクラスタ オペレータについての補足のトピックを参照してください。

1 Helm ドキュメントを参照して、Helm をインストールします。

2 `gpu-operator` Helm リポジトリを追加します。

```
helm repo add nvidia https://nvidia.github.io/gpu-operator
```

3 次のコマンドを実行して、NVAIE GPU Operator をインストールします。

必要に応じて、環境変数の値を環境に対応する値に置き換えます。

```
export PRIVATE_REGISTRY="private/registry/path"
export OS_TAG=ubuntu20.04
export VERSION=460.73.01
export VGPU_DRIVER_VERSION=460.73.01-grid
export NGC_API_KEY=ZmJjMHZya...LWExNTRi
export REGISTRY_SECRET_NAME=registry-secret

helm install nvidia/gpu-operator \
   --set driver.repository=$PRIVATE_REGISTRY \
   --set driver.version=$VERSION \
   --set driver.imagePullSecrets={$REGISTRY_SECRET_NAME} \
   --set operator.defaultRuntime=containerd \
   --set driver.licensingConfig.configMapName=licensing-config
```

Operator の手順 11：AI/ML ワークロードをデプロイする

NVIDIA GPU Cloud カタログには、vGPU 対応 Tanzu Kubernetes クラスタで AI/ML ワークロードを実行する場合に使用できる既製のコンテナ イメージがいくつかあります。使用可能なイメージの詳細については、NGC のドキュメントを参照してください。

TKGS クラスタへの AI/ML ワークロードのデプロイに関する vSphere 管理者向けの補足（vGPU および動的 DirectPath I/O）

vGPU および動的 DirectPath I/O を使用して AI/ML ワークロードをサポートするように TKGS を構成する場合は、この差分トピックを参照してください。

動的 DirectPath I/O 機能を持つ vGPU のための vSphere 管理者ワークフローの調整

vGPU と 動的 DirectPath I/O を使用するには、同じ TKGS クラスタ (vGPU) への AI/ML ワークロードのデプロイに関する vSphere 管理者ワークフローに従いますが、次に示す変更点があります。

管理者の手順 2：PCI デバイスのパススルーを有効にする

vGPU と 動的 DirectPath I/O を使用するには、各 ESXi ホストを vGPU 用に構成します（管理者の手順 2：各 ESXi ホストを vGPU 操作用に構成するを参照）。

さらに、次のようにして、GPU デバイスを構成します。

1 vSphere Client を使用して、vCenter Server にログインします。

2 vCenter Server クラスタでターゲット ESXi ホストを選択します。

3 [構成] - [ハードウェア] - [PCI デバイス] の順に選択します。

4 [すべての PCI デバイス] タブを選択します。

5 ターゲット NVIDIA GPU アクセラレータ デバイスを選択します。

6 [パススルーの切り替え] をクリックします。

7 ESXi ホストを右クリックして、メンテナンス モードにします。

8 ホストを再起動します。

9 ホストが再実行されているときに、メンテナンス モードを終了します。

管理者の手順 7：vGPU と 動的 DirectPath I/O を使用したカスタム仮想マシン クラスの作成

vGPU と動的 DirectPath I/O を使用するには、[NVIDIA vGPU] プロファイルを使用するカスタム仮想マシン クラスを構成します（管理者の手順 7：vGPU プロファイルを使用するカスタム仮想マシン クラスを作成するを参照）。

この仮想マシン クラスに、[動的 DirectPath I/O] が指定され、サポート対象の PCI デバイスが選択された、2 番目の PCI デバイス構成を追加します。このタイプの仮想マシン クラスがインスタンス化されている場合、仮想マシンの配置は vSphere Distributed Resource Scheduler (DRS) によって決まります。

vGPU と動的 DirectPath I/O をサポートするカスタム仮想マシン クラスを作成するには、次の手順を参照してください。その他のガイダンスについては、vSphere with Tanzu での仮想マシン クラスへの PCI デバイスの追加を参照してください。

1 vSphere Client を使用して、vCenter Server にログインします。

2 [ワークロード管理] を選択します。

3 [[サービス]] を選択します。

4 [仮想マシン クラス] を選択します。

5 [NVIDIA vGPU] プロファイルを使用してすでに構成されているカスタム仮想マシン クラスを編集します。

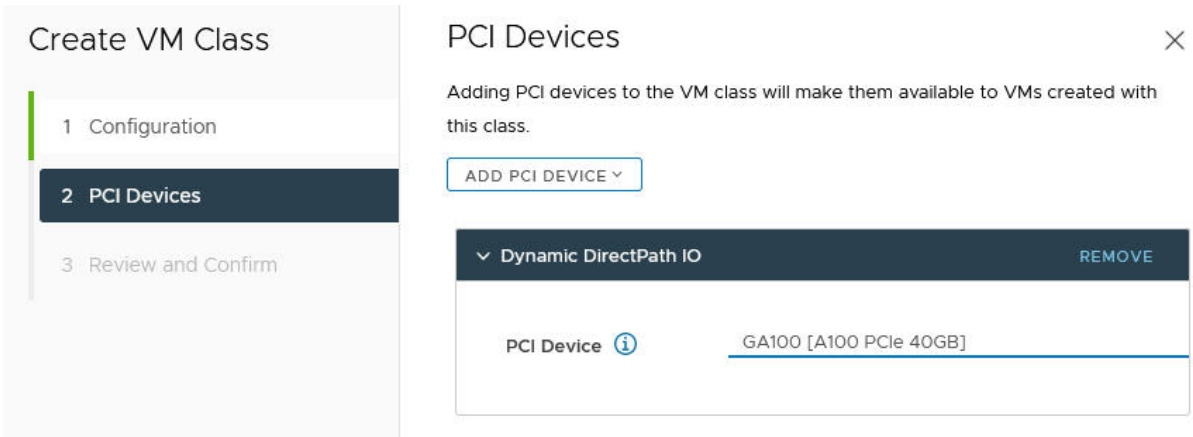
6 [PCI デバイス] タブを選択します。

7 [PCI デバイスの追加] をクリックします。

8 [動的 DirectPath I/O] オプションを選択します。

9 [PCI デバイス] を選択します。

例：



10 [次へ] をクリックします。

11 選択内容を確認します。

12 [終了] をクリックします。

13 新しいカスタム仮想マシン クラスが仮想マシン クラスのリストで使用可能になっていることを確認します。

TKGS クラスタ (DLS) への AI/ML ワークロードのデプロイに関するクラスタ オペレータについての補足

NVIDIA AI Enterprise アカウントで NVIDIA Delegated Licensing Server (DLS) を使用している場合は、この差分トピックを参照してください。

TKGS クラスタへの AI/ML ワークロードのデプロイに関するクラスタ オペレータについての補足

NVIDIA は、DLS (Delegated Licensing Serve) と呼ばれる新しい NVIDIA ライセンス サーバ (NLS) システムを提供しています。詳細については、NVIDIA のドキュメントを参照してください。

NVAIE アカウントで DLS を使用している場合、NVAIE GPU Operator の準備とデプロイの手順は TKGS クラスタへの AI/ML ワークロードのデプロイに関するクラスタ オペレータのワークフローに記載されている内容と異なります。具体的には、手順 9 および 10 は次のように変更されています。

Operator の手順 9：NVAIE GPU Operator のインストール準備を行う

次の手順を実行し、DLS を使用して GPU Operator をインストールする準備を行います。

1 シークレットを作成します。

```
kubectl create secret docker-registry registry-secret \
  --docker-server=<users private NGC registry name>
  --docker-username='$oauthtoken' \
  --docker-password=ZmJj……Ri \
  --docker-email=<user-email-address> -n gpu-operator-resources
```

注： パスワードは、以前に NVIDIA GPU Cloud (NGC) ポータルで作成されたユーザー API キーです。

2 DLS サーバからクライアント トークンを取得します。

vGPU ライセンスを使用するユーザーは、DLS ライセンス サーバから「クライアント トークン」と呼ばれるトークンを取得する必要があります。この操作のメカニズムは、NVIDIA のドキュメントに記載されています。

3 クライアント トークンを使用して、TKGS クラスタ内に構成マップ オブジェクトを作成します。

クライアント トークン ファイルを `<path>/client_configuration_token.tok` に配置します。

次のコマンドを実行します。

```
kubectl delete configmap licensing-config -n gpu-operator-resources; > gridd.conf
kubectl create configmap licensing-config \
  -n gpu-operator-resources --from-file=./gridd.conf --from-file=./
client_configuration_token.tok
```

注： DLS で使用される grid.conf ファイルは空です。ただし、両方の「--from-file」パラメータは必須です。

Operator の手順 10：NVAIE GPU Operator をインストールする

次の手順を実行して、DLS を使用して NVAIE GPU Operator をインストールします。その他のガイダンスについては、GPU Operator のドキュメントを参照してください。

1 TKGS クラスタに NVAIE GPU Operator をインストールします。

- Helm ドキュメントを参照して、Helm をインストールします。

- `gpu-operator` Helm リポジトリを追加します。

```
helm repo add nvidia https://nvidia.github.io/gpu-operator
```

- Helm を使用して GPU Operator をインストールします。

```
export PRIVATE_REGISTRY="<user's private registry name>"
export OS_TAG=ubuntu20.04
export VERSION=470.63.01
export VGPU_DRIVER_VERSION=470.63.01-grid
export NGC_API_KEY=Zm……Ri  <- The user's NGC AP Key
export REGISTRY_SECRET_NAME=registry-secret

helm show chart .
kubectl delete crd clusterpolicies.nvidia.com
helm install gpu-operator . -n gpu-operator-resources \
  --set psp.enabled=true \
  --set driver.licensingConfig.configMapName=licensing-config \
  --set operator.defaultRuntime=containerd \
  --set driver.imagePullSecrets={$REGISTRY_SECRET_NAME} \
  --set driver.version=$VERSION \
  --set driver.repository=$PRIVATE_REGISTRY \
  --set driver.licensingConfig.nlsEnabled=true
```

2 DLS が機能していることを確認します。

GPU Operator によってデプロイされた NVIDIA ドライバの DaemonSet ポッド内で `nvidia-smi` コマンドを実行して、DLS が機能していることを確認します。

まず、次のコマンドを実行してポッドに移動し、シェル セッションを起動します。

```
kubectl exec -it nvidia-driver-daemonset-cvxx6 nvidia-driver-ctr -n gpu-operator-resources
- bash
```

これで、コマンドを実行して、DLS の設定を確認できるようになりました。

```
nvidia-smi
```

DLS が正しく設定されている場合は、このコマンドの出力で「Licensed」が返されます。

vSphere with Tanzu ワークロードに対するコンテナ レジストリの使用

15

コンテナ レジストリは、Kubernetes オペレータにコンテナ イメージの保存と共有に便利なリポジトリを提供します。vSphere with Tanzu には、スーパーバイザー クラスタ で有効にできる組み込みの Harbor レジストリ が含まれています。また、Tanzu Kubernetes クラスタで外部のプライベート コンテナ レジストリを使用することもできます。

スーパーバイザー クラスタ で組み込み Harbor レジストリ を有効にして、vSphere ポッド および Tanzu Kubernetes クラスタ ワークロードをデプロイするためのプライベート コンテナ レジストリとして機能させることができます。vSphere Docker 認証情報ヘルパー を使用してレジストリに安全にアクセスし、コンテナ イメージをプッシュおよびプルできる開発者に、レジストリの URL を提供します。

注： 組み込み Harbor レジストリ を使用するには、NSX-T ネットワークを使用するように スーパーバイザー クラスタ が構成されている必要があります。

組み込み Harbor レジストリ の代わりに、またはこれに加えて、外部のプライベート コンテナ レジストリを使用するように Tanzu Kubernetes クラスタを構成することもできます。

この章には、次のトピックが含まれています。

- スーパーバイザー クラスタ の組み込み Harbor レジストリ の有効化
- 組み込みの Harbor レジストリ コンソールへのログイン
- 組み込みの Harbor レジストリ 証明書のダウンロードとインストール
- 組み込みの Harbor レジストリ 証明書を使用した Docker クライアントの構成
- vSphere Docker 認証情報ヘルパー のインストールとレジストリへの接続
- 組み込みの Harbor レジストリ へのイメージのプッシュ
- 組み込みの Harbor レジストリ からのイメージのパージ
- Tanzu Kubernetes クラスタでの組み込みの Harbor レジストリ の使用
- Tanzu Kubernetes クラスタでの外部コンテナ レジストリの使用

スーパーバイザー クラスタ の組み込み Harbor レジストリ の有効化

vSphere 管理者は、vSphere with Tanzu に組み込まれている Harbor レジストリを有効にできます。レジストリに対してコンテナ イメージのプッシュとプルを行ったり、これらのイメージを使用して をデプロイしたりできます。

Harbor レジストリ を有効にすると、スーパーバイザー クラスタ のすべての名前空間に、プライベート イメージ レジストリと同じ名前のプロジェクトが対応するようになります。名前空間に対する編集権限または表示権限を持つすべてのユーザーまたはグループは、プライベート イメージ レジストリで同じ名前の対応するプロジェクトで、対応するロールのメンバーになります。プライベート イメージ レジストリのプロジェクトおよびメンバーのライフサイクルは自動的に管理され、名前空間および名前空間内のユーザーまたはグループの権限のライフサイクルにリンクされます。

前提条件

組み込みの Harbor レジストリを有効にするには、[ワークロード管理] を有効にして、スーパーバイザー クラスタをデプロイしておく必要があります。さらに、コンテナ イメージの配置用のストレージ ポリシーを作成します。このストレージ ポリシーは、レジストリ内のコンテナ イメージのバッキング ストアとして使用するパーシステント ボリュームをプロビジョニングするために使用されます。

注： 組み込み Harbor レジストリ を使用するには、ネットワーク ソリューションとして NSX-T Data Center とともに スーパーバイザー クラスタ をデプロイする必要があります。vSphere with Tanzu 用 NSX-T Data Center の構成を参照してください。

手順

1 vSphere Client で、[ワークロード管理] が有効な vCenter Server クラスタを参照します。

2 [構成] を選択します。

3 [スーパーバイザー クラスタ] を選択します。

4 [イメージ レジストリ] を選択します。

5 [Harbor を有効化] をクリックします。

6 コンテナ イメージの配置用の [ストレージ ポリシー] を選択します。

7 [OK] をクリックしてプロセスを完了します。

結果

プライベート イメージ レジストリは、数分後に有効になります。プライベート イメージ レジストリのインスタンスに対して、特殊な名前空間が作成されます。この名前空間に対して操作を実行することはできません。vSphere ユーザーにとっては、読み取り専用です。

次のステップ

組み込みの Harbor レジストリ コンソールへのログイン。

組み込みの Harbor レジストリ コンソールへのログイン

組み込みの Harbor レジストリ 管理コンソールを使用して、プライベート レジストリを管理および操作します。

vSphere 管理者は、組み込みの Harbor レジストリ 管理コンソールを使用して、プロジェクトの作成と管理、ログの表示、Harbor API の確認を行うことができます。

前提条件

スーパーバイザー クラスタ の組み込み Harbor レジストリ の有効化

手順

1 vSphere Client で、[ワークロード管理] が有効な vCenter Server クラスタを参照します。

2 [構成] を選択します。

3 [名前空間] で、[イメージ レジストリ] を選択します。

4 [Harbor ユーザー インターフェイスへのリンク] をクリックします。

組み込みの Harbor レジストリ コンソールのログイン画面が表示されます。

5 vSphere 管理者の認証情報を使用してログインします。

組み込みの Harbor レジストリ 証明書のダウンロードとインストール

組み込みの Harbor レジストリ ルート CA 証明書をダウンロードし、それを使用してクライアントをレジストリに接続できるようにします。

Docker クライアントを使用して組み込みの Harbor レジストリ にログインするには、このクライアントにルート CA 証明書をインストールする必要があります。

前提条件

このタスクでは、クライアント ホスト マシンに Docker をインストールして、構成していることを前提としています。また、組み込みの Harbor レジストリ が有効になっている必要があります。スーパーバイザー クラスタ の組み込み Harbor レジストリ の有効化を参照してください。

手順

1 組み込みの Harbor レジストリ 証明書をダウンロードします。これには、以下の 2 つの方法があります。

- 組み込みの Harbor レジストリ コンソール インターフェイスを使用します。
  - URL を使用して、組み込みの Harbor レジストリ コンソールにログインします。組み込みの Harbor レジストリ コンソールへのログインを参照してください。
  - [プロジェクト] - [プロジェクト名] 画面にあるプロジェクトのリンクをクリックします。
  - [リポジトリ] タブを選択します。
  - [レジストリ証明書] をクリックします。
  - `ca.crt` 証明書ファイルをローカル マシンに保存します。
- vSphere Client を使用する場合。
  - [ワークロード管理] および組み込みの Harbor レジストリが有効な vCenter Server クラスタを選択します。
  - [構成] - [名前空間] - [イメージ レジストリ] の順に選択します。

- [ルート証明書] フィールドで、[SSL ルート証明書のダウンロード] リンクをクリックします。
- `root-certificate.txt` ファイルをローカル マシンに保存します。
- ファイルの名前を `ca.crt` に変更します。

2 ダウンロードした組み込みの Harbor レジストリ の `ca.crt` ファイルを、Docker がインストールされているホスト上の適切なディレクトリにコピーします。証明書のデフォルトの場所は、Docker クライアントで実行されている OS のタイプによって異なります。

- Linux

```
/etc/docker/certs.d/ca.crt
```

- Mac OS

```
security add-trusted-cert -d -r trustRoot -k ~/Library/Keychains/login.keychain ca.crt
```

注： `ca.crt` をデフォルトの場所にインストールしていない場合は、vSphere Docker 認証情報ヘルパー を使用してログインするときに `--tlscacert /path/to/ca.crt` フラグを渡すことができます。

3 インポートが完了したら、Docker デーモンを再起動します。

- Linux

```
sudo systemctl restart docker.service
```

- Mac

  Docker デスクトップ メニュー オプションの [Docker の再起動]、または `command R` キーボード ショートカットを使用します。

次のステップ

vSphere Docker 認証情報ヘルパー のインストールとレジストリへの接続

組み込みの Harbor レジストリ 証明書を使用した Docker クライアントの構成

Docker を使用して組み込みの Harbor レジストリのコンテナ イメージを操作するには、Docker クライアントにレジストリ証明書を追加する必要があります。この証明書は、ログイン時に Docker に対して認証する場合に使用されます。

組み込みの Harbor レジストリと通信するように Docker クライアントを構成します。このタスクは、組み込みの Harbor レジストリへの接続と操作の際に、vSphere が提供する Docker 認証情報ヘルパーの使用準備で必要になります。

前提条件

このタスクでは、組み込みの Harbor レジストリが有効なことと、ログイン可能なことを前提としています。

- スーパーバイザー クラスタ の組み込み Harbor レジストリ の有効化

- 組み込みの Harbor レジストリ コンソールへのログイン

また、手順では、Docker デーモンがインストールされている Linux ホスト (Ubuntu) を使用していることを前提としています。Docker がインストールされ、Docker ハブからイメージをプルできることを確認するには、次のコマンドを実行します。

```
docker run hello-world
```

予期される結果：

```
Hello from Docker!
This message shows that your installation appears to be working correctly.
```

注： これらの手順は、Ubuntu 20.04 および Docker 19.03 を使用して検証されています。

手順

1. 組み込みの Harbor レジストリ証明書 `root-certificate.txt` をダウンロードします。組み込みの Harbor レジストリ 証明書のダウンロードとインストールを参照してください。
2. 証明書の名前を `ca.crt` に変更します。
3. `ca.crt` ファイルを Docker ホストに安全にコピーします。
4. Docker ホストで、Harbor の IP アドレスを使用してプライベート レジストリのディレクトリ パスを作成します。

   ```
   /etc/docker/certs.d/IP-address-of-harbor/
   ```

   例：

   ```
   mkdir /etc/docker/certs.d/10.179.145.77
   ```

5. `ca.crt` をこのディレクトリに移動します。

   例：

   ```
    mv ca.crt /etc/docker/certs.d/10.179.145.77/ca.crt
   ```

6. Docker デーモンを再起動します。

   ```
   sudo systemctl restart docker.service
   ```

7. Docker クライアントを使用して、組み込みの Harbor レジストリにログインします。

   ```
   docker login https://10.179.145.77
   ```

次のメッセージが表示されます。

```
WARNING! Your password will be stored unencrypted in /home/ubuntu/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

次のステップ

メッセージの指示に沿って、セキュリティのために vSphere Docker 認証情報ヘルパー をダウンロードしてインストールします。vSphere Docker 認証情報ヘルパー のインストールとレジストリへの接続を参照してください。

vSphere Docker 認証情報ヘルパー のインストールとレジストリへの接続

コンテナ イメージをに安全にプッシュし、組み込みの Harbor レジストリからコンテナ イメージをプルするには、vSphere Docker 認証情報ヘルパー CLI を使用します。

Kubernetes CLI Tools ダウンロード ページには、vSphere Docker 認証情報ヘルパー をダウンロードするためのリンクが掲載されています。vSphere Docker 認証情報ヘルパー を使用して、Docker クライアントを組み込みの Harbor レジストリに安全に接続します。

前提条件

- スーパーバイザー クラスタ の組み込み Harbor レジストリ の有効化
-
- vSphere 管理者から vSphere 向け Kubernetes CLI Tools ダウンロード ページのリンクを取得します。
- または、vCenter Server にアクセスできる場合は、次のようにしてリンクを取得します。
  - vSphere Client を使用して、vCenter Server にログインします。
  - [vSphere クラスタ] - [名前空間] に移動し、作業している vSphere 名前空間 を選択します。
  - [サマリ] タブを選択し、[ステータス] ファイルを特定します。
  - [CLI ツールへのリンク] という見出しの下にある [開く] を選択してダウンロード ページを開きます。または、リンクを [コピー] することもできます。
- Docker クライアントを構成します。組み込みの Harbor レジストリ 証明書を使用した Docker クライアントの構成を参照してください。

手順

1. ブラウザを使用して、環境に応じた [Kubernetes CLI Tools] ダウンロード URL に移動します。
2. vSphere Docker 認証情報ヘルパー セクションまでスクロールします。
3. オペレーティング システムを選択します。
4. `vsphere-docker-credential-helper.zip` ファイルをダウンロードします。

5 この ZIP ファイルのコンテンツを作業ディレクトリに解凍します。

`docker-credential-vsphere` バイナリ実行ファイルを使用できます。

6 `docker-credential-vsphere` バイナリを Docker クライアント ホストにコピーします。

7 バイナリの場所をシステム パスに追加します。

たとえば、Linux では次のようになります。

```
mv docker-credential-vsphere /usr/local/bin/docker-credential-vsphere
```

8 シェルまたはターミナル セッションで `docker-credential-vsphere` コマンドを実行して、vSphere Docker 認証情報ヘルパー のインストールを確認します。

バナー メッセージと、CLI のコマンドライン オプションのリストが表示されます。

```
vSphere login manager is responsible for vSphere authentication.
It allows vSphere users to securely login and logout to access Harbor images.

Usage:
  docker-credential-vsphere [command]

Available Commands:
  help        Help about any command
  login       Login into specific harbor server and get authentication
  logout      Logout from Harbor server and erase user token

Flags:
  -h, --help   help for docker-credential-vsphere

Use "docker-credential-vsphere [command] --help" for more information about a command.
```

9 組み込みの Harbor レジストリにログインします。

まず、使用量を確認します。

```
docker-credential-vsphere login -help
Usage:
  docker-credential-vsphere login [harbor-registry] [flags]

Flags:
  -h, --help               help for login
  -s, --service string     credential store service
      --tlscacert string   location to CA certificate (default "/etc/docker/certs.d/*.crt")
  -u, --user string        vSphere username and password
```

次のコマンドを使用してログインします。

```
docker-credential-vsphere login <container-registry-IP>
```

認証トークンが取得され、保存されて、ログイン状態になります。

```
docker-credential-vsphere login 10.179.145.77
Username: administrator@vsphere.local
Password: INFO[0017] Fetched username and password
INFO[0017] Fetched auth token
INFO[0017] Saved auth token
```

10 組み込みの Harbor レジストリからログアウトします。

```
docker-credential-vsphere logout 10.179.145.77
```

次のステップ

組み込みの Harbor レジストリ へのイメージのプッシュ 。

組み込みの Harbor レジストリ へのイメージのプッシュ

Docker から組み込みの Harbor レジストリ のプロジェクトにイメージをプッシュできます。組み込みの Harbor レジストリ 内のプロジェクトは、スーパーバイザー クラスタ の vSphere 名前空間に対応しています。

前提条件

次のタスクが完了していることが前提となります。

- スーパーバイザー クラスタ の組み込み Harbor レジストリ の有効化
- vSphere Docker 認証情報ヘルパー のインストールとレジストリへの接続

さらに、イメージをプッシュする Harbor レジストリ のプロジェクトに対応する名前空間に対する書き込み権限を持つユーザー アカウントを取得します。

最後に、レジストリにプッシュできるローカル イメージが必要になります。次のコマンドを実行すると、Docker ハブから hello-world イメージがプルされます。イメージをプルするには、アカウントが必要になります。

```
docker run hello-world
```

予期される結果：

```
Hello from Docker!
This message shows that your installation appears to be working correctly.
```

`docker images` コマンドを使用してイメージを確認します。

```
docker images
REPOSITORY     TAG                 IMAGE ID            CREATED             SIZE
hello-world    latest              bf756fb1ae65        10 months ago       13.3kB
```

手順

1 vSphere Docker 認証情報ヘルパーを使用して Harbor レジストリ にログインします。

```
docker-credential-vsphere login <container-registry-IP> --user username@domain.com
```

注： ログイン時に `--user username` を指定することは可能ですが、UserPrincipalName (UPN) 構文 (`--user username@domain.com`) を使用してログインし、`docker push` コマンドを使用する必要があります。

2 Harbor レジストリ のプロジェクトにプッシュするイメージに、そのイメージを使用する名前空間と同じ名前をタグ付けします。

```
docker tag <image-name>[:TAG] <container-registry-IP>/<project-name>/<image-name>[:TAG]
```

例：

```
docker tag hello-world:latest 10.179.145.77/tkgs-cluster-ns/hello-world:latest
```

```
docker images
REPOSITORY                                      TAG                 IMAGE ID
CREATED             SIZE
10.179.145.77/tkgs-cluster-ns/hello-world     latest              bf756fb1ae65        10
months ago       13.3kB
hello-world                                     latest              bf756fb1ae65        10
months ago       13.3kB
```

3 イメージを Harbor のプロジェクトにプッシュするには、次のコマンドを実行します。

構文:

```
docker push <container-registry-IP>/<namespace-name>/<image_name>
```

例：

```
docker push 10.179.145.77/tkgs-cluster-ns/hello-world:latest
```

予期される結果：

```
The push refers to repository [10.179.145.77/tkgs-cluster-ns/hello-world]
9c27e219663c: Pushed
latest: digest: sha256:90659bf80b44ce6be8234e6ff90a1ac34acbeb826903b02cfa0da11c82cbc042
size: 525
```

4 組み込みの Harbor レジストリでイメージが使用可能になったことを確認します。

- 組み込みの Harbor レジストリ コンソールへのログイン
- [プロジェクト] - [プロジェクト名] にあるプロジェクトのリンクをクリックします。
- [リポジトリ] タブを選択します。

- レジストリにプッシュしたイメージが、`namespace/image-name` の形式（`tkgs-cluster-ns/hello-world` など）で表示されます。
- このイメージを選択すると、`latest` タグおよびその他のメタデータが表示されます。

5 [リポジトリ] タブに戻ります。

6 [イメージ Docker コマンドのプッシュ] ドロップダウン メニューを選択します。このリポジトリにタグを付けてイメージをプッシュするコマンドが表示されます。

例

次の図に、組み込みの Harbor レジストリにイメージをプッシュする例を示します。

```
docker tag busybox:latest <container-registry-IP>/<namespace-name>/busybox:latest
docker push <container-registry-IP>/busybox/busybox:latest
```

次のステップ

Harbor レジストリのイメージを使用して vSphere ポッド をデプロイします。組み込みの Harbor レジストリを使用した vSphere ポッド へのアプリケーションのデプロイを参照してください。

組み込みの Harbor レジストリ からのイメージのパージ

vSphere 管理者は、DevOps エンジニアからの要求により、プライベート イメージ レジストリ内のプロジェクトのイメージをパージできます。プライベート イメージ レジストリからイメージをパージすると、ポッドによって作成されたイメージへの参照がすべて削除されますが、イメージ レジストリからイメージが削除されることはありません。

DevOps エンジニアからプロジェクト用に保存されているイメージが多すぎると報告を受けた場合、vSphere 管理者はプライベート イメージ レジストリ内のそのプロジェクトのイメージをパージできます。プロジェクトからイメージをパージすると、これらのイメージへのすべての参照が削除されますが、これらのイメージは vSphere データストアからは削除されません。毎週土曜日の午前 2 時に、プライベート イメージ レジストリ全体から、アプリケーションによって参照されていないすべてのイメージがガベージ コレクタ サービスによって削除されます。

手順

1 vSphere Client で、名前空間に移動します。

2 [構成]、[全般] の順に選択します。

3 [組み込みレジストリ] の横にある [パージ] をクリックします。

Tanzu Kubernetes クラスタでの組み込みの Harbor レジストリ の使用

組み込みの Harbor レジストリ を、Tanzu Kubernetes Grid サービス によってプロビジョニングされた Tanzu Kubernetes クラスタにデプロイするイメージのプライベート コンテナ レジストリとして使用することができます。

vSphere with Tanzu には、スーパーバイザー クラスタ で有効にできる Harbor レジストリ インスタンスが組み込まれていて、コンテナベースのワークロードを Tanzu Kubernetes クラスタにデプロイする場合に使用できます。

スーパーバイザー クラスタ で組み込みの Harbor レジストリ が有効になっている場合は、Tanzu Kubernetes Grid サービス によって、レジストリ インスタンスのルート CA 証明書が Tanzu Kubernetes クラスタ ノードにインストールされます。この証明書は、(調整ループを使用して) 新しいクラスタと既存のクラスタの両方にインストールされます。クラスタ内のイメージを実行するには、ワークロード YAML 内でプライベート レジストリを指定します。

ワークフロー

次のワークフローを使用して、Tanzu Kubernetes クラスタ ノードからプライベート レジストリに安全にアクセスし、コンテナ イメージをプルします。

手順	操作	方法
0	Tanzu Kubernetes クラスタで組み込みの Harbor レジストリを使用する際のワークフローを確認します。	15 章 vSphere with Tanzu ワークロードに対するコンテナ レジストリの使用を参照してください。
1	スーパーバイザー クラスタで組み込みの Harbor レジストリを有効にします。	スーパーバイザー クラスタ の組み込み Harbor レジストリ の有効化を参照してください。
2	レジストリ サービスのシークレットを使用して、各クラスタの kubeconfig を構成します。	以下の手順を参照してください：組み込みの Harbor レジストリ のイメージ プル シークレットを使用した Tanzu Kubernetes クラスタの構成。
3	プライベート コンテナ レジストリを指定するようにワークロード YAML を構成します。	以下の手順を参照してください：組み込みの Harbor レジストリ のイメージ プル シークレットを使用した Tanzu Kubernetes クラスタの構成。
4	組み込みの Harbor レジストリにイメージをプッシュするには、Docker クライアントを構成し、vSphere Docker 認証情報ヘルパーをインストールします。	組み込みの Harbor レジストリ 証明書を使用した Docker クライアントの構成と組み込みの Harbor レジストリ へのイメージのプッシュ を参照してください。

組み込みの Harbor レジストリ のイメージ プル シークレットを使用した Tanzu Kubernetes クラスタの構成

イメージ プル シークレットで kubeconfig を構成し、Tanzu Kubernetes クラスタをプライベート コンテナ レジストリ、組み込みの Harbor レジストリ または外部プライベート レジストリに接続します。

1 スーパーバイザー クラスタ に接続します。vCenter Single Sign-On ユーザーとして スーパーバイザー クラスタ に接続するを参照してください。

2 ターゲット Tanzu Kubernetes クラスタがプロビジョニングされている vSphere 名前空間 にコンテキストを切り替えます。

```
kubectl config use-context tkgs-cluster-ns
```

3 vSphere 名前空間 のイメージ プル シークレットを取得して、ファイルに保存します。

```
kubectl get secret -n <vsphere-namespace> <vsphere-namespace>-default-image-pull-secret -o
yaml > <path>/image-pull-secret.yaml
```

例：

```
kubectl get secret -n tkgs-cluster-ns tkgs-cluster-ns-default-image-pull-secret -o yaml >
tanzu/image-pull-secret.yaml
```

4 テキスト エディタで image-pull-secret.yaml を開きます。最小限必要な変更を行い、完了したらファイルを保存します。

```
apiVersion: v1
data:
  .dockerconfigjson: ewoJCQkJImF1dGhzJUV2s1ZVZwWVFuWmp...
kind: Secret
metadata:
  creationTimestamp: "2020-11-12T02:41:08Z"
  managedFields:
  - apiVersion: v1
    ...
  name: harbor-registry-secret   #OPTIONAL: Change if desired
  namespace: default           #REQUIRED: Enter the Kuberentes namespace
  ownerReferences:
  - apiVersion: registryagent.vmware.com/v1alpha1
    ...
  resourceVersion: "675868"
  selfLink: /api/v1/namespaces/tkgs-cluster-ns/secrets/tkgs-cluster-ns-default-image-pull-
secret
  uid: 66606b41-7363-4b74-a3f2-4436f83f
type: kubernetes.io/dockerconfigjson
```

- 必須：namespace の値を、**default** など、クラスタ内の適切な Kubernetes 名前空間と一致するように変更します。

  注： イメージ プル シークレットを構成するには、Kubernetes 名前空間を指定します。Tanzu Kubernetes クラスタがすでに存在する場合は、コンテキストをそのクラスタに切り替えて kubectl get namespaces を実行し、使用可能な Kubernetes 名前空間を一覧表示します。必要な場合は、先にターゲット名前空間を作成してから続行します。Tanzu Kubernetes クラスタが存在しない場合は、default 名前空間を使用できます。

- オプション：name の値を **harbor-registry-secret**、**private-registry-secret** などの意味のある値に変更します。

5 Tanzu Kubernetes クラスタへのアクセスに使用可能な kubeconfig ファイルを作成します。

```
kubectl get secret -n <vsphere-namespace> <cluster-name>-kubeconfig -o
jsonpath='{.data.value}' | base64 -d > <path>/cluster-kubeconfig
```

<vsphere-namespace> を、ターゲット Tanzu Kubernetes クラスタがプロビジョニングされている vSphere 名前空間 の名前に置き換えます。*<cluster-name>* を Tanzu Kubernetes クラスタの名前に置き換えます。

```
kubectl get secret -n tkgs-cluster-ns tkgs-cluster-5-kubeconfig -o
jsonpath='{.data.value}' | base64 -d > tanzu/cluster-kubeconfig
```

6 Tanzu Kubernetes クラスタ内にレジストリ サービスのシークレットを作成します。ローカルで保存および更新していたイメージ プル シークレット ファイルを参照します。

```
kubectl --kubeconfig=<path>/cluster-kubeconfig apply -f <path>/image-pull-secret.yaml
```

例：

```
kubectl --kubeconfig=tanzu/cluster-kubeconfig apply -f tanzu/image-pull-secret.yaml
```

レジストリ サービスのシークレットが正常に作成されたことを確認します。

```
secret/harbor-registry-secret created
```

プライベート コンテナ レジストリからイメージをプルするための Tanzu Kubernetes ワークロードの構成

Tanzu Kubernetes クラスタのワークロードのためにプライベート コンテナ レジストリからイメージをプルするには、プライベート レジストリの詳細を使用してワークロード YAML を構成します。

この手順は、プライベート コンテナ レジストリまたは組み込みの Harbor レジストリ からイメージをプルする場合に使用できます。この例では、組み込みの Harbor レジストリ に保存されているイメージを使用するポッド仕様を作成し、以前に構成したイメージ プル シークレットを利用します。

1 プライベート レジストリの詳細を含むサンプルのポッド仕様を作成します。

```
apiVersion: v1
kind: Pod
metadata:
  name: <workload-name>
  namespace: <kubernetes-namespace>
spec:
  containers:
  - name: private-reg-container
    image: <Registry-IP-Address>/<vsphere-namespace>/<image-name>:<version>
  imagePullSecrets:
  - name: <registry-secret-name>
```

- <workload-name> をポッド ワークロードの名前に置き換えます。
- <kubernetes-namespace> を、ポッドが作成されるクラスタ内の Kubernetes 名前空間に置き換えます。この名前空間は、レジストリ サービスのイメージ プル シークレットが格納されている、Tanzu Kubernetes クラスタ内の Kubernetes 名前空間（デフォルトの名前空間など）と同じにする必要があります。
- <Registry-IP-Address> を、スーパーバイザー クラスタ で実行されている組み込みの Harbor レジストリ インスタンスの IP アドレスに置き換えます。
- *<vsphere-namespace>* を、ターゲット Tanzu Kubernetes がプロビジョニングされている vSphere 名前空間 に置き換えます。
- <image-name> を、任意のイメージ名に置き換えます。

- <version> を、イメージの適切なバージョン（「最新」バージョンなど）に置き換えます。
- <registry-secret-name> を、以前に作成したレジストリ サービスのイメージ プル シークレットの名前に置き換えます。

2 定義したポッド仕様に基づいて、Tanzu Kubernetes クラスタ内にワークロードを作成します。

```
kubectl --kubeconfig=<path>/cluster-kubeconfig apply -f <pod.yaml>
```

ポッドは、レジストリからプルされたイメージに基づいて作成する必要があります。

Tanzu Kubernetes クラスタでの外部コンテナ レジストリの使用

Tanzu Kubernetes クラスタ ポッドで外部コンテナ レジストリを使用できます。これは組み込みの Harbor レジストリ の代わりに使用できます。

外部プライベート レジストリの使用事例

コンテナ レジストリは、Kubernetes 環境に重要な機能を提供し、コンテナ イメージを保存して共有するための一元的なリポジトリとして機能します。最もよく使用されるパブリック コンテナ レジストリは DockerHub です。プライベート コンテナ レジストリのサービスは多数あります。VMware Harbor は、オープンソースのクラウドネイティブなプライベート コンテナ レジストリです。vSphere with Tanzu には、vSphere ポッド および Tanzu Kubernetes クラスタで実行されているポッドのプライベート コンテナ レジストリとして使用できる Harbor のインスタンスが組み込まれています。詳細については、『スーパーバイザー クラスタ の組み込み Harbor レジストリ の有効化』を参照してください。

vSphere with Tanzu に付属する組み込みの Harbor レジストリ を使用するには、NSX-T ネットワークが必要です。vSphere ネットワークを使用している場合、このレジストリは使用できません。また、Tanzu Kubernetes クラスタと統合する独自のプライベート コンテナ レジストリをすでに実行している場合があります。この場合、自己署名証明書を使用してプライベート レジストリを信頼するように Tanzu Kubernetes Grid サービス を構成できます。これにより、Tanzu Kubernetes クラスタで実行されている Kubernetes ポッドで外部レジストリを使用できます。

外部プライベート レジストリの要件

Tanzu Kubernetes クラスタで外部プライベート レジストリを使用するには、vSphere with Tanzu バージョン 7 U2 以降を使用する必要があります。

独自のプライベート レジストリは、Tanzu Kubernetes クラスタおよび Tanzu Kubernetes リリース ノード仮想マシンで実行される Kubernetes ポッドでのみ使用できます。ESXi ホスト上でネイティブに実行される vSphere ポッド では独自のプライベート レジストリは使用できません。vSphere ポッド でサポートされるレジストリは、vSphere with Tanzu プラットフォームに組み込まれている Harbor レジストリ です。

Tanzu Kubernetes Grid サービス をプライベート レジストリ用に構成すると、プロビジョニングされる新しいクラスタでプライベート レジストリがサポートされます。既存のクラスタでプライベート レジストリをサポートするには、TkgServiceConfiguration を適用するためのローリング アップデートが必要です。Tanzu Kubernetes クラスタの更新を参照してください。また、カスタム TkgServiceConfiguration を初めて作成した場合は、ローリング アップデートが開始されます。

外部プライベート レジストリの構成

Tanzu Kubernetes クラスタで独自のプライベート レジストリを使用するには、1 つ以上の自己署名証明書を使用して、HTTPS 経由でプライベート レジストリ コンテンツを提供するように Tanzu Kubernetes Grid サービスを構成します。

プライベート レジストリの自己署名証明書をサポートするように `TkgServiceConfiguration` が更新されます。具体的には、`additionalTrustedCAs` フィールドを含む新しい `trust` セクションが追加され、Tanzu Kubernetes クラスタが信頼する任意の数の自己署名証明書を定義できます。この機能を使用すると、証明書のリストを簡単に定義でき、ローテーションが必要な場合にそれらの証明書を更新できます。

`TkgServiceConfiguration` を更新して適用すると、次にクラスタが作成されるときに TLS 証明書が新しいクラスタに適用されます。つまり、`TkgServiceConfiguration.trust.additionalTrustedCAs` に更新を適用しても、Tanzu Kubernetes クラスタの自動ローリング アップデートはトリガされません。

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TkgServiceConfiguration
metadata:
  name: tkg-service-configuration
spec:
  defaultCNI: antrea
  trust:
    additionalTrustedCAs:
      - name: first-cert-name
        data: base64-encoded string of a PEM encoded public cert 1
      - name: second-cert-name
        data: base64-encoded string of a PEM encoded public cert 2
```

更新を適用するには、次のコマンドを実行します。

```
kubectl apply -f tkgserviceconfiguration.yaml
```

Tanzu Kubernetes Grid サービス 仕様をプライベート レジストリ証明書で更新しているため、Tanzu Kubernetes クラスタで組み込みの Harbor レジストリ を使用する場合と同様に、パブリック キーを Tanzu Kubernetes クラスタの kubeconfig に追加する必要はありません。

プライベート コンテナ レジストリからイメージをプルするための Tanzu Kubernetes ワークロードの構成

Tanzu Kubernetes クラスタのワークロードのためにプライベート コンテナ レジストリからイメージをプルするには、プライベート レジストリの詳細を使用してワークロード YAML を構成します。

この手順は、プライベート コンテナ レジストリまたは組み込みの Harbor レジストリ からイメージをプルする場合に使用できます。この例では、組み込みの Harbor レジストリ に保存されているイメージを使用するポッド仕様を作成し、以前に構成したイメージ プル シークレットを利用します。

1 プライベート レジストリの詳細を含むサンプルのポッド仕様を作成します。

```
apiVersion: v1
kind: Pod
metadata:
  name: <workload-name>
  namespace: <kubernetes-namespace>
spec:
  containers:
  - name: private-reg-container
    image: <Registry-IP-Address>/<vsphere-namespace>/<image-name>:<version>
  imagePullSecrets:
  - name: <registry-secret-name>
```

- `<workload-name>` をポッド ワークロードの名前に置き換えます。
- `<kubernetes-namespace>` を、ポッドが作成されるクラスタ内の Kubernetes 名前空間に置き換えます。この名前空間は、レジストリ サービスのイメージ プル シークレットが格納されている、Tanzu Kubernetes クラスタ内の Kubernetes 名前空間（デフォルトの名前空間など）と同じにする必要があります。
- `<Registry-IP-Address>` を、スーパーバイザー クラスタ で実行されている組み込みの Harbor レジストリ インスタンスの IP アドレスに置き換えます。
- *<vsphere-namespace>* を、ターゲット Tanzu Kubernetes がプロビジョニングされている vSphere 名前空間 に置き換えます。
- `<image-name>` を、任意のイメージ名に置き換えます。
- `<version>` を、イメージの適切なバージョン（「最新」バージョンなど）に置き換えます。
- `<registry-secret-name>` を、以前に作成したレジストリ サービスのイメージ プル シークレットの名前に置き換えます。

2 定義したポッド仕様に基づいて、Tanzu Kubernetes クラスタ内にワークロードを作成します。

```
kubectl --kubeconfig=<path>/cluster-kubeconfig apply -f <pod.yaml>
```

ポッドは、レジストリからプルされたイメージに基づいて作成する必要があります。

外部プライベート レジストリの信頼フィールド

証明書エントリ（PEM でエンコードされたパブリック証明書の base64 エンコード文字列）を `TkgServiceConfiguration` の `additionalTrustedCAs` セクションに追加します。データは、TkgServiceConfiguration にプレーン テキストで保存されるパブリック証明書です。

表 15-1. プライベート レジストリの信頼フィールド

フィールド	説明
trust	セクション マーカー。データを受け入れない。
additionalTrustedCAs	セクション マーカー。それぞれの名前とデータを示す証明書の配列が含まれます。
name	TLS 証明書の名前。
data	PEM でエンコードされたパブリック証明書の base64 エンコード文字列。

外部プライベート レジストリ証明書の削除

証明書を TkgServiceConfiguration の additionalTrustedCAs セクションの証明書リストから削除し、TkgServiceConfiguration を Tanzu Kubernetes Grid サービス に適用します。

外部プライベート レジストリ証明書のローテーション

証明書をローテーションするには、仮想インフラストラクチャ管理者または DevOps エンジニアが TkgServiceConfiguration または Tanzu Kubernetes クラスタ仕様の証明書の内容を変更し、その構成を適用して、その TKC のローリング アップデートをトリガします。

外部プライベート レジストリ証明書のトラブルシューティング

信頼する証明書を使用して Tanzu Kubernetes Grid サービス を構成し、自己署名証明書をクラスタの kubeconfig に追加すると、その自己署名証明書を使用するプライベート レジストリからコンテナ イメージを正常にプルできるはずです。

次のコマンドは、コンテナ イメージがポッド ワークロードに対して正常にプルされたかどうかを判断するのに役立ちます。

```
kubectl describe pod PODNAME
```

このコマンドを実行すると、特定のポッドの詳細なステータスとエラー メッセージが表示されます。カスタム証明書をクラスタに追加する前にイメージをプルする例を次に示します。

```
Events:
  Type     Reason                          Age               From               Message
  ----     ------                          ----              ----               -------
  Normal   Scheduled                       33s               default-scheduler  ...
  Normal   Image                           32s               image-controller   ...
  Normal   Image                           15s               image-controller   ...
  Normal   SuccessfulRealizeNSXResource  7s (x4 over 31s)  nsx-container-ncp  ...
  Normal   Pulling                         7s                kubelet            Waiting test-gc-
e2e-demo-ns/testimage-8862e32f68d66f727d1baf13f7eddef5a5e64bbd-v10612
  Warning  Failed                          4s                kubelet            failed to get
images: ... Error: ... x509: certificate signed by unknown authority
```

次のコマンドを実行します。

```
kubectl get pods
```

ポッド全体のステータス ビューに ErrImagePull エラーも表示されます。

```
NAME                                          READY   STATUS         RESTARTS   AGE
testimage-nginx-deployment-89d4fcff8-2d9pz    0/1     Pending        0          17s
testimage-nginx-deployment-89d4fcff8-7kp9d    0/1     ErrImagePull   0          79s
testimage-nginx-deployment-89d4fcff8-7mpkj    0/1     Pending        0          21s
testimage-nginx-deployment-89d4fcff8-fszth    0/1     ErrImagePull   0          50s
testimage-nginx-deployment-89d4fcff8-sjnjw    0/1     ErrImagePull   0          48s
testimage-nginx-deployment-89d4fcff8-xr5kg    0/1     ErrImagePull   0          79s
```

エラー「x509: certificate signed by unknown authority」と「ErrImagePull」は、プライベート コンテナ レジストリに接続するための正しい証明書を使用してクラスタが構成されていないことを示しています。証明書が見つからないか、構成が正しくありません。

証明書の構成後にプライベート レジストリに接続するとエラーが発生する場合は、構成に適用された証明書がクラスタに適用されているかどうかを確認できます。構成に適用された証明書が正しく適用されているかどうかを確認するには、SSH を使用します。

SSH を使用してワーカー ノードに接続することで、2 つの調査手順を実行できます。

1 フォルダ /etc/ssl/certs/ に tkg-<cert_name>.pem という名前のファイルがあるかを確認します。<cert_name> は、TkgServiceConfiguration に追加された証明書の「name」プロパティです。証明書が TkgServiceConfiguration 内の証明書と一致していてもプライベート レジストリの使用が機能しない場合は、次の手順を実行してさらに診断します。

2 コマンド openssl s_client -connect hostname:port_num を実行して、自己署名証明書を使用したターゲット サーバへの openssl 接続テストを実行します。hostname は自己署名証明書を使用しているプライベート レジストリのホスト名/DNS 名で、port_num はサービスが実行されているポート番号です（HTTPS の場合は通常 443）。自己署名証明書を使用しているエンドポイントに接続するときに openssl によって返されるエラーを正確に確認し、適切な証明書を TkgServiceConfiguration に追加するなどしてその状況を解決できます。Tanzu Kubernetes クラスタに誤った証明書が組み込まれている場合は、正しい証明書を使用して Tanzu Kubernetes Grid サービス 構成を更新し、Tanzu Kubernetes クラスタを削除してから、正しい証明書を含む構成を使用してクラスタを再作成する必要があります。

vSphere Lifecycle Manager の操作 16

vSphere 管理者は、単一の VMware vSphere Lifecycle Manager イメージを使用して管理している vSphere クラスタで vSphere with Tanzu を有効にできます。その後、スーパーバイザー クラスタ が vSphere Lifecycle Manager によって管理されている間、このクラスタを使用できます。

vSphere Lifecycle Manager には、環境内の ESXi ホストおよびクラスタを管理する機能があります。スーパーバイザー クラスタ を最新バージョンの vSphere with Tanzu にアップグレードできます。スーパーバイザー クラスタ 内のホストの ESXi バージョンをアップグレードすることもできます。

vSphere Lifecycle Manager は vCenter Server で実行されるサービスです。vCenter Server をデプロイすると、HML5 ベースの vSphere Client で vSphere Lifecycle Manager ユーザー インターフェイスが有効になります。

vSphere Lifecycle Manager の詳細については、ホストとクラスタのライフサイクルの管理ドキュメントを参照してください。

この章には、次のトピックが含まれています。

要件

vSphere Lifecycle Manager によって管理されている vSphere クラスタで vSphere with Tanzu を構成するには、環境が特定の要件を満たしている必要があります。

システム要件

vSphere with Tanzu を有効にするには、vSphere クラスタのコンポーネントが次の要件を満たしていることを確認します。

- NSX-T Data Center を使用する場合は、vCenter Server と ESXi がバージョン 7.0 Update 2 であることを確認します。

- vSphere ネットワークを使用する場合は、vCenter Server と ESXi がバージョン 7.0 Update 1 以降であることを確認します。
- スーパーバイザー クラスタ の一部として使用するすべての ESXi ホストに、Kubernetes のアドオンを含む VMware vSphere 7 Enterprise Plus のライセンスが割り当てられていることを確認します。
- vSphere クラスタで HA と DRS が有効であることを確認します。
- vSphere Distributed Switch バージョン 7.0 Update 2 が構成されていることを確認します。
- クラスタで vSphere ネットワークまたは NSX-T Data Center 3.1 以降のバージョンが構成されていることを確認します。それより前のバージョンの NSX-T Data Center で構成されているクラスタは、vSphere Lifecycle Manager イメージを使用して管理することはできません。

vSphere Lifecycle Manager によって管理されているクラスタでの vSphere with Tanzu の有効化

単一の vSphere Lifecycle Manager イメージを使用して管理しているクラスタで vSphere with Tanzu を有効にすることで、Kubernetes ワークロードを実行できます。このクラスタを有効にすると、vSphere Lifecycle Manager を使用して スーパーバイザー クラスタ を管理できるようになります。

NSX-T Data Center を使用する vSphere with Tanzu でクラスタを有効にすると、vSphere Lifecycle Manager によってクラスタ内のすべての ESXi ホストに Spherelet vSphere インストール バンドル (VIB) がインストールされます。クラスタを有効にすると、vCenter Server に付属するバージョンの Kubernetes が割り当てられます。インストールが完了すると、WCP サービスは、Spherelet の起動や構成などのインストール後のタスクを実行します。

クラスタを有効にする手順については、vSphere ネットワークを使用したワークロード管理の有効化を参照してください。

スーパーバイザー クラスタ のアップグレード

vSphere with Tanzu クラスタを支えている vSphere インフラストラクチャ、Kubernetes のバージョン、単一の vSphere Lifecycle Manager イメージを使用するクラスタ上の vSphere 向け Kubernetes CLI Tools などを含め、vSphere with Tanzu を最新バージョンに更新できます。

スーパーバイザー クラスタ 内のホストの ESXi バージョンをアップグレードします。アップグレード中に、すべての ESXi ホスト上の Spherelet VIB がアップグレードされます。

vSphere Lifecycle Manager は DRS を使用して、ホストをメンテナンス モードにしてから修正します。DRS は、修正が正常に行われるようにするために、vCenter Server が実行されている仮想マシンを別のホストに移行するよう試みます (ホストにアフィニティ化された仮想マシンやホストのローカル ストレージ上で実行されている仮想マシンなど)。また、vSphere ポッド を含むワークロードを他のホストに移行します。

注： vSphere Lifecycle Manager を使用すると、単一の vSphere Lifecycle Manager イメージを使用するクラスタでのみ スーパーバイザー クラスタ をアップグレードできます。

手順

1 vSphere Client メニューから、[ワークロード管理] を選択します。

2 [更新] タブを選択します。

3 更新する [使用可能なバージョン] を選択します。

たとえば、バージョン `v1.17.4-vsc0.0.2-16293900` を選択します。

4 アップデートを適用する対象の スーパーバイザー クラスタ を選択します。

5 アップデートを開始するには [更新の適用] をクリックします。

6 更新の状態を監視するには、[最近のタスク] ペインを使用します。

スーパーバイザー クラスタ へのホストの追加

実行するワークロードを増やすには、vSphere 管理者が スーパーバイザー クラスタ をスケールアウトする必要があります。クラスタに容量を追加するには、単一の vSphere Lifecycle Manager イメージを使用するクラスタに ESXi ホストを追加します。

NSX-T Data Center で構成されている スーパーバイザー クラスタ にホストを追加すると、vSphere Lifecycle Manager によって Spherelet VIB とイメージがホストにインストールされます。インストール後、vSphere with Tanzu は、新たに追加されたホスト上で Spherelet プロセスを構成します。これにより、コンテナを ESXi 上でネイティブに実行できます。

前提条件

- ホストの root ユーザー アカウントのユーザー名およびパスワードを取得します。
- ファイアウォールの背後にあるホストが vCenter Server と通信できることを確認します。

手順

1 vSphere Client メニューから、[ワークロード管理] を選択します。

2 データセンター、クラスタ、またはフォルダを右クリックし、[ホストの追加] を選択します。

3 ホストの IP アドレスまたは名前を入力し、[次へ] をクリックします。

4 管理者の認証情報を入力し、[次へ] をクリックします。

5 ホストの概要を確認し、[次へ] をクリックします。

6 ホストにライセンスを割り当て、[終了] をクリックします。

7 [ホストの追加] ウィザードで [次へ] をクリックします。

8 概要を確認し、[終了] をクリックします。

注： ホストが同じデータセンター内にある場合は、スーパーバイザー クラスタ 内に移動することができます。ホストを移動するには、ホストをメンテナンス モードにしてから、クラスタにドラッグしてください。

スーパーバイザー クラスタ からのホストの削除

vSphere 管理者として、コストを節約するために スーパーバイザー クラスタ のスケール インが必要になる場合があります。スーパーバイザー クラスタ のキャパシティを削減するために、単一の vSphere Lifecycle Manager イメージを使用するクラスタから ESXi ホストを削除できます。

NSX-T Data Center を使用して構成される スーパーバイザー クラスタ からホストを削除すると、vSphere with Tanzu によって Spherelet の構成がクリアされ、ESXi ホストの Spherelet プロセスが停止します。その後、vSphere Lifecycle Manager によって Spherelet VIB およびイメージがホストからアンインストールされ、vSphere with Tanzu によってクラスタ制御プレーンからホスト メタデータが削除されます。

前提条件

クラスタからホストを削除する前に、そのホストで実行されているすべての仮想マシンをパワーオフするか、新しいホストに仮想マシンを移行する必要があります。

手順

1 vSphere Client で、ホストを削除するクラスタに移動します。

2 ホストを右クリックし、ポップアップ メニューで [メンテナンス モードへの切り替え] を選択します。

3 表示される確認のダイアログ ボックスで、[はい] をクリックします。

確認のダイアログ ボックスでは、パワーオフ状態の仮想マシンを他のホストに移動するかどうかも尋ねられます。それらの仮想マシンをクラスタ内のホストに登録したままにする場合は、このオプションをオンにします。

ホストのアイコンが変わり、名前に 「メンテナンス モード」 と括弧書きで追加されます。

4 インベントリからホストのアイコンを選択し、新しい場所にドラッグします。

ホストは、別のクラスタまたは別のデータセンターに移動できます。

vCenter Server により、ホストが新しい場所に移動されます。

5 ホストを右クリックし、ポップアップ メニューから [メンテナンス モードの終了] を選択します。

6 （オプション） 必要に応じて仮想マシンを再起動します。

スーパーバイザー クラスタの無効化

単一の vSphere Lifecycle Manager イメージを使用する vSphere クラスタから vSphere with Tanzu を無効にして、従来のワークロードに使用できるようにすることができます。

クラスタで vSphere with Tanzu を無効にすると、vSphere Lifecycle Manager によって各 ESXi ホストから Spherelet の VIB とイメージがアンインストールされるとともに、`WCP` サービスによってクラスタ内のすべてのワークロードが停止し、削除されます。

手順

1 vSphere Client メニューから、[ワークロード管理] を選択します。

2 [クラスタ] タブを選択します。

3 vSphere with Tanzu を無効にするクラスタを選択します。

4 [無効化] をクリックします。

[クラスタの無効化] ダイアログ ボックスが開き、クラスタですべての Kubernetes ワークロードと NSX-T Data Center 構成が無効になることを示すメッセージが表示されます。

5 [無効化] をクリックします。

vSphere with Tanzu 環境の更新

17

vSphere with Tanzu の最新バージョンに更新できます。これには、vSphere with Tanzu クラスタをサポートする vSphere インフラストラクチャ、Kubernetes のバージョン、および vSphere 向け Kubernetes CLI Tools が含まれます。

この章には、次のトピックが含まれています。

vSphere with Tanzu の更新について

vSphere with Tanzu は、スーパーバイザー クラスタ と Tanzu Kubernetes Grid サービス クラスタ、およびこれらのクラスタをサポートするインフラストラクチャのローリング アップデートに対応しています。

スーパーバイザー クラスタ および Tanzu Kubernetes Grid サービス クラスタの更新方法

vSphere with Tanzu は、スーパーバイザー クラスタ および Tanzu Kubernetes Grid サービス クラスタのローリング アップデート モデルを使用します。ローリング アップデート モデルを使用すると、更新プロセスでのクラスタのワークロードのダウンタイムが最小限に抑えられます。ローリング アップデートには、Kubernetes ソフトウェアのバージョンのアップグレードに加えて、仮想マシンの構成とリソース、vSphere サービスと名前空間、カスタム リソースなどの、Kubernetes クラスタをサポートするインフラストラクチャおよびサービスの更新が含まれます。

アップデートが正常に実行されるためには、構成がいくつかの互換性要件を満たしている必要があります。そのため、システムは再チェック条件を適用して、クラスタの更新の準備ができていることを確認し、クラスタのアップグレードが失敗した場合のロールバックをサポートします。

注： vSphere with Tanzu のアップデートに含まれるのは、Kubernetes ソフトウェア バージョンのアップグレードだけではありません。このプロセスの説明には、ソフトウェア バージョンの更新に限定される「アップグレード」という言葉ではなく、「更新」という言葉を使用しています。

スーパーバイザー クラスタ の更新と Tanzu Kubernetes Grid サービス クラスタの更新の依存関係

スーパーバイザー クラスタ クラスタと Tanzu Kubernetes Grid サービス クラスタを個別に更新します。ただし、両者の間には依存関係があることに注意してください。

スーパーバイザー クラスタ を更新すると、そこにデプロイされている Tanzu Kubernetes Grid サービス クラスタのローリング アップデートがトリガされることがあります。vSphere 名前空間の更新の実行による スーパーバイザー クラスタ の更新を参照してください。

Tanzu Kubernetes Grid サービス クラスタがターゲット スーパーバイザー クラスタ のバージョンに準拠していない場合は、スーパーバイザー クラスタ を更新する前に、1 つ以上の Tanzu Kubernetes Grid サービス クラスタを更新する必要が生じることがあります。更新のための Tanzu Kubernetes クラスタ互換性の確認を参照してください。

スーパーバイザー クラスタ の更新について

スーパーバイザー クラスタ の更新を開始すると、システムは新しい制御プレーン ノードを作成し、既存の制御プレーンに追加します。更新のこの段階では、システムは新たに更新されたノードを追加して、古いノードを削除するため、vSphere インベントリには 4 台の制御プレーン ノードが表示されます。オブジェクトは、古い制御プレーン ノードのいずれかから新しいノードに移行され、古い制御プレーン ノードは削除されます。すべての制御プレーン ノードが更新されるまで、このプロセスが 1 回ずつ繰り返されます。制御プレーンが更新されると、同様のローリング アップデート方式でワーカー ノードが更新されます。ワーカー ノードは ESXi ホストであり、各 ESXi ホストの各 spherelet プロセスが 1 つずつ更新されます。

次の更新の中から選択できます。

- vSphere 名前空間 を更新します。
- VMware バージョンや Kubernetes バージョンなどを含め、すべてを更新します。

スーパーバイザー クラスタ が実行される Kubernetes バージョンの更新（Kubernetes 1.16.7 から Kubernetes 1.17.4 など）、および スーパーバイザー クラスタ と Tanzu Kubernetes Grid サービス クラスタをサポートするインフラストラクチャの更新には、vSphere 名前空間 の更新ワークフローを使用します。このタイプの更新は頻繁に行われ、Kubernetes のリリース頻度に対応するために使用されます。vSphere 名前空間 の更新手順は次のとおりです。

1 vCenter Server をアップグレードします。

2 vSphere 名前空間 の更新（Kubernetes のアップグレードを含む）を実行します。

vSphere 名前空間 を更新するには、vSphere 名前空間の更新の実行による スーパーバイザー クラスタ の更新を参照してください。

すべての vSphere with Tanzu コンポーネントを更新するには、すべてを更新するワークフローを使用します。このタイプの更新は、NSX-T 3.X から 4、vSphere 7.X から 8 など、メジャー リリースをアップデートする場合に必要です。この更新ワークフローは、新しい VMware 製品リリースの公開時期に応じて低い頻度で行われます。すべてを更新する手順は次のとおりです。

1 互換性を判断するには、VMware 製品の相互運用性マトリックス (https://interopmatrix.vmware.com/Interoperability) で vCenter Server と NSX-T Data Center を確認してください。vSphere with Tanzu の機能は、vCenter Server に付属するワークロード制御プレーン (WCP) ソフトウェアによって提供されます。

2 NSX-T Data Center に互換性がある場合は、アップグレードします。

3 vCenter Server をアップグレードします。

4 vSphere Distributed Switch をアップグレードします。

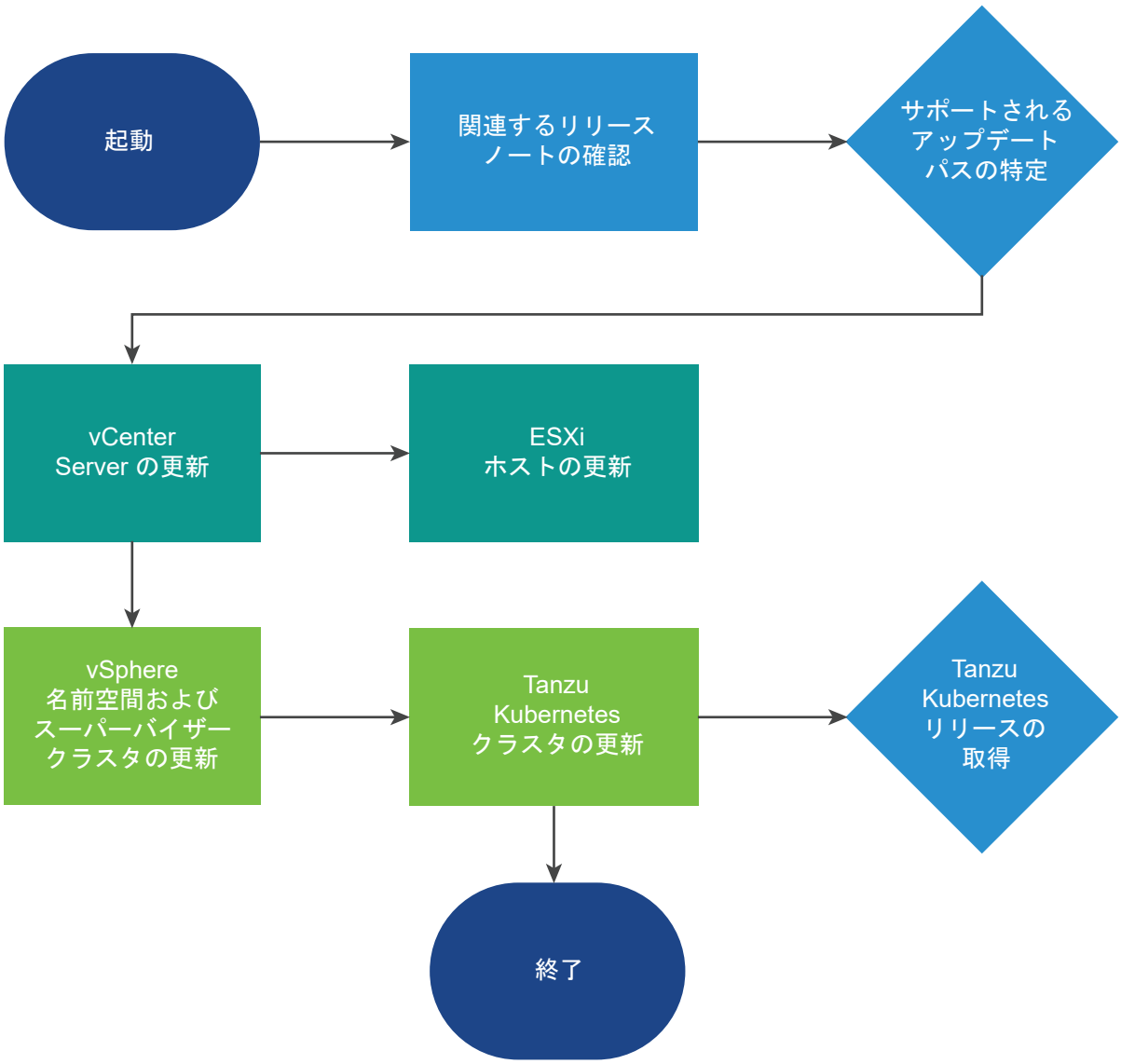
5 ESXi ホストをアップグレードします。

6 プロビジョニングされた Tanzu Kubernetes Grid サービス クラスタとターゲット スーパーバイザー クラスタ のバージョンの互換性を確認します。

7 vSphere 名前空間 を更新します（スーパーバイザー クラスタ Kubernetes バージョンを含む）。

8 Tanzu Kubernetes Grid サービス クラスタを更新します。

次の図に、vSphere with Tanzu を更新する一般的なワークフローを示します。



Tanzu Kubernetes Grid サービス クラスタの更新について

スーパーバイザー クラスタ を更新すると、スーパーバイザー クラスタ に展開された Tanzu Kubernetes Grid サービス などの Tanzu Kubernetes Grid サービス クラスタをサポートするインフラストラクチャ コンポーネントも同様に更新されます。各インフラストラクチャの更新には、Tanzu Kubernetes Grid サービス (CNI、CSI、CPI) をサポートするサービスの更新、および既存の Tanzu Kubernetes Grid サービス クラスタに適用可能な、制御プレーンおよびワーカー ノード用の更新された設定を含めることができます。構成が互換性の要件を満たすようにするために、vSphere with Tanzu はローリング アップデート中に事前チェックを実行し、コンプライアンスを適用します。

Tanzu Kubernetes Grid サービス クラスタのローリング アップデートを実行するには、クラスタのマニフェストを更新します。Tanzu Kubernetes クラスタの更新を参照してください。ただし、vSphere 名前空間 の更新を実行すると、システムは更新された構成を直ちにすべての Tanzu Kubernetes Grid サービス クラスタに伝達することに注意してください。これらの更新により、Tanzu Kubernetes Grid サービス 制御プレーンおよびワーカー ノードのローリング アップデートが自動的にトリガされます。

クラスタ ノードを置き換えるためのローリング アップデート プロセスは、Kubernetes 環境でのポッドのローリング アップデートと同様です。Tanzu Kubernetes Grid サービス クラスタのローリング アップデートを実行するコントローラは 2 つあります。アドオン コントローラと TanzuKubernetesCluster コントローラです。これらの 2 つのコントローラでは、ローリング アップデートに、アドオンの更新、制御プレーンの更新、およびワーカー ノードの更新の 3 つの主要なステージがあります。これらのステージは順番に実行されますが、前の手順が十分に進行するまで次のステップの開始を防ぐ事前チェックが実行されます。不要と判断された場合、ステップはスキップされることがあります。たとえば、更新がワーカー ノードのみに影響する場合、アドオンや制御プレーンの更新は必要なくなります。

更新プロセスでは、システムは新しいクラスタ ノードを追加し、ノードがターゲットの Kubernetes バージョンでオンラインになるまで待機します。その後、古いノードを削除対象としてマークし、次のノードに移動して、プロセスを繰り返します。すべてのポッドが削除されるまで、古いノードは削除されません。たとえば、ノードの完全なドレーンを妨げる PodDisruptionBudgets でポッドが定義されている場合、ノードは遮断されますが、それらのポッドが消去できるようになるまで削除されません。システムは、最初にすべての制御プレーン ノード、次にワーカー ノードをアップグレードします。更新中に、Tanzu Kubernetes Grid サービス クラスタのステータスが「更新中」に変わります。ローリング アップデート プロセスが完了すると、Tanzu Kubernetes Grid サービス クラスタのステータスが「実行中」に変わります。

レプリケーション コントローラによって管理されていない Tanzu Kubernetes Grid サービス クラスタで実行されているポッドは、Tanzu Kubernetes Grid サービス クラスタの更新中にワーカー ノードのドレインの一環として、Kubernetes バージョンのアップグレードで削除されます。クラスタの更新が手動でトリガされた場合や、vSphere 名前空間 の更新によって自動実行された場合がこれに該当します。レプリケーション コントローラによって管理されていないポッドには、Deployment または ReplicaSet 仕様の一部として作成されていないポッドなどがあります。詳細については、Kubernetes ドキュメントの Pod Lifecycle: Pod lifetime のトピックを参照してください。

ネットワーク トポロジのアップグレード

vSphere with Tanzu バージョン 7.0 Update 1c をインストールするか、スーパーバイザー クラスタ をバージョン 7.0 Update 1 からバージョン 7.0 Update 1c にアップグレードする場合は、NSX Container Plug-in (NCP) をアップグレードします。この操作を行うと、スーパーバイザー クラスタ、名前空間、および Tanzu Kubernetes クラスタのネットワーク トポロジが移行されます。アップグレード後、ネットワーク トポロジは単一の Tier-1 ゲートウェイ トポロジから、スーパーバイザー クラスタ 内の名前空間ごとに Tier-1 ゲートウェイが 1 つあるトポロジにアップグレードされます。

アップグレード中に、NCP は新しいトポロジをサポートするように NSX-T Data Center リソースを構成します。NCP は、レイヤー 4 およびレイヤー 7 ロード バランシング サービスが少ない名前空間向けに、共有ネットワーク インフラストラクチャを提供します。これにより、NSX のリソースが削減され、Tanzu Kubernetes クラスタで使用できるリソースが増えます。

システム名前空間は、スーパーバイザー クラスタ クラスタと Tanzu Kubernetes クラスタの機能に不可欠な主要コンポーネントで使用される名前空間です。Tier-1 ゲートウェイ、ロード バランサ、SNAT IP を含む共有ネットワーク リソースは、システム名前空間内でグループ化されます。

NCP はデフォルトで、システム名前空間用の共有 Tier-1 ゲートウェイを 1 つ作成し、名前空間ごとに Tier-1 ゲートウェイとロード バランサを 1 つずつ作成します。Tier-1 ゲートウェイは、Tier-0 ゲートウェイとデフォルトのセグメントに接続されています。

NSX-T ロード バランサは、仮想サーバの形式でロード バランシング サービスを提供します。

移行後のネットワーク トポロジーの特性は、次のとおりです。

- 各 vSphere 名前空間 には、個別のネットワークのほかに、Tier-1 ゲートウェイ、ロード バランサ サービス、SNAT IP アドレスなど、名前空間内のアプリケーションで共有される一連のネットワーク リソースが含まれています。
- 同じ名前空間内にある vSphere ポッド、通常の仮想マシン、または Tanzu Kubernetes クラスタで実行されるワークロードは、North-South 接続に対して同じ SNAT IP アドレスを共有します。
- vSphere ポッド または Tanzu Kubernetes クラスタで実行されるワークロードには、デフォルトのファイアウォールによって実装される共通の隔離ルールが適用されます。
- Kubernetes 名前空間ごとに個別の SNAT IP アドレスが必要になることはありません。名前空間の間の East-West 接続は、SNAT ではありません。

実行できる名前空間の最大数は、Edge ノードのサイズ（中、大、または特大）と、NSX Edge クラスタ内の Edge ノードの数によって決まります。実行可能な名前空間の数は、Edge ノードの数に 20 を掛けた値より小さくなります。たとえば、NSX Edge クラスタ内に「大」サイズの Edge ノードが 10 台含まれている場合、作成可能なスーパーバイザー ネームスペースの最大数は 199 です。

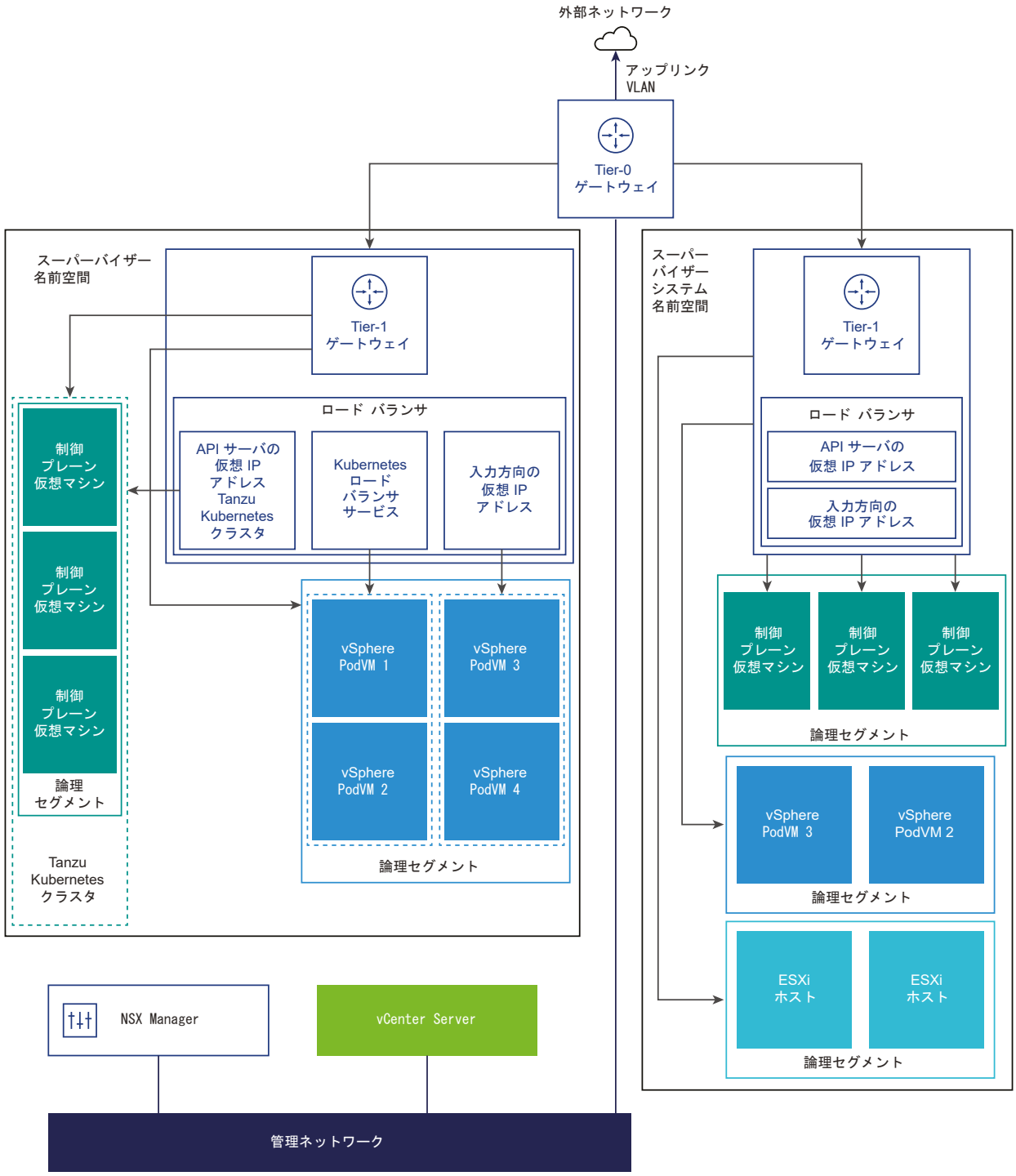
Edge ノードのサイズの詳細については、『NSX-T Data Center インストール ガイド』を参照してください。

スーパーバイザー クラスタのネットワーク

スーパーバイザー クラスタの共有 Tier-1 ゲートウェイ内に、個別のセグメントがあります。各 Tanzu Kubernetes クラスタのセグメントは、名前空間の Tier-1 ゲートウェイ内で定義されています。

同じ名前空間内にある、vSphere ポッド および Tanzu Kubernetes クラスタを含むワークロードは、North-South 接続に対して SNAT IP アドレスを共有します。名前空間の間の East-West 接続は、SNAT ではありません。

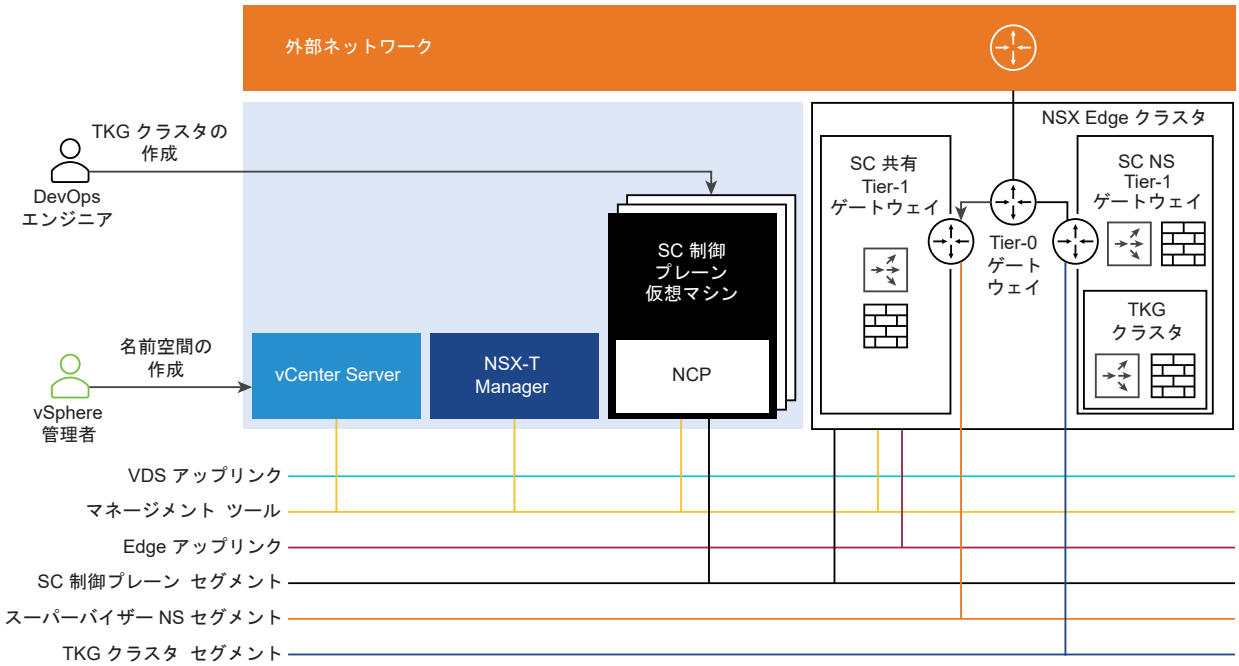
図 17-1. スーパーバイザー クラスタのネットワーク



Tanzu Kubernetes クラスタのネットワーク

スーパーバイザー クラスタ のアップグレード後、DevOps エンジニアがスーパーバイザー名前空間内の最初の Tanzu Kubernetes クラスタをプロビジョニングすると、クラスタは名前空間と同じ Tier-1 ゲートウェイおよびロード バランサを共有するようになります。この名前空間で Tanzu Kubernetes クラスタがプロビジョニングされるたびに、このクラスタのセグメントが作成され、そのスパーバイザー名前空間内の共有 Tier-1 ゲートウェイに接続されます。

Tanzu Kubernetes Grid サービス によって Tanzu Kubernetes クラスタがプロビジョニングされると、1 台の仮想サーバが作成され、Kubernetes API 用のレイヤー 4 ロード バランシングが可能になります。この仮想サーバは、名前空間を使用する共有ロード バランサ上にホストされ、制御プレーンに kubectl トラフィックをルーティングします。さらに、クラスタでリソースが提供される Kubernetes サービス ロード バランサごとに、このサービスのレイヤー 4 のロード バランシングを提供する仮想サーバが作成されます。



NSX-T ネットワーク トポロジのアップグレード

ネットワーク トポロジをアップグレードするには、NSX-T Data Center、vCenter Server、およびすべての vSphere with Tanzu コンポーネントをアップグレードします。

NSX-T Data Center のバージョンをアップグレードできます。アップグレードすると、システムのダウンタイムを最小限に抑えながら、データ プレーン、制御プレーン、管理プレーンを含む NSX-T Data Center のコンポーネントがアップグレードされます。

NSX-T Data Center コンポーネントのアップグレード手順の詳細については、『NSX-T Data Center アップグレード ガイド』を参照してください。

注： サポートされているバージョンについては、「vSphere with Tanzu リリース ノート」を参照してください。

前提条件

- 環境がシステムの要件を満たしていることを確認します。要件の詳細については、NSX-T Data Center を使用して vSphere with Tanzu をセットアップするためのシステム要件を参照してください。
- vSphere with Tanzu に固有の構成制限については、VMware 構成の上限ツールの vSphere 構成の制限を参照してください。

手順

1. NSX-T Data Center を NSX-T Data Center 3.0.x から NSX-T Data Center 3.1 にアップグレードします。
2. vCenter Server をアップグレードします。
3. ESXi ホストをアップグレードします。
4. スーパーバイザー名前空間を更新します。

   更新するには、vSphere 名前空間の更新の実行による スーパーバイザー クラスタ の更新を参照してください。

vSphere Distributed Switch のアップグレード

バージョン 7.0 の vSphere Distributed Switch を以降のバージョンにアップグレードできます。Distributed Switch のアップグレードにより、スイッチに接続されているホストおよび仮想マシンで短いダウンタイムが発生することがあります。

前提条件

- vCenter Server をバージョン 7.0 にアップグレードします。
- Distributed Switch に接続されたすべてのホストを ESXi 7.0 にアップグレードします。

手順

1. vSphere Client のホーム メニューで、[ホストおよびクラスタ] を選択します。
2. [ネットワーク] を選択して、Distributed Switch に移動します。

   たとえば、[DSwitch] などです。

   vSphere Client のユーザー インターフェイスには、アップグレード先として使用可能なバージョンが表示されます。
3. [アクション] メニューで [アップグレード] - [Distributed Switch のアップグレード] を選択します。
4. スイッチをアップグレードする vSphere Distributed Switch のバージョンを選択し、[次へ] をクリックします。
5. ホストの互換性を確認し、[次へ] をクリックします。
6. アップグレードの構成を行い、[終了] をクリックします。

vSphere 名前空間の更新の実行による スーパーバイザー クラスタの更新

スーパーバイザー クラスタ が実行されている Kubernetes バージョンを含む、1 つ以上の スーパーバイザー クラスタ と、Tanzu Kubernetes Grid サービス を含む、Tanzu Kubernetes クラスタをサポートするインフラストラクチャを更新するには、vSphere 名前空間を更新します。

vSphere with Tanzu のバージョン エンティティがあります。バージョン エンティティは `v1.19.1+vmware.2-vsc0.0.8-17610687` 形式のセマンティック バージョン文字列です。プリフィックスは Kubernetes バージョン (`v1.19.1`)、サフィックスは vSphere 名前空間のバージョン (`vsc0.0.8-17610687`) です。

表に、使用可能な スーパーバイザー クラスタ リリースを示します。

バージョン	説明
`v1.19.1+vmware.2-vsc0.0.8-17610687`	最新の スーパーバイザー クラスタ リリース。vSphere 7.0 Update 2 をサポートします。
`v1.18.2-vsc0.0.5-16762486`	vSphere 7.0 Update 1 をサポートします。
`v1.17.4-vsc0.0.5-16762486`	Antrea CNI をサポートする Tanzu Kubernetes Grid サービス を含む最小の スーパーバイザー クラスタ リリース。
`v1.16.7-vsc0.0.5-16762486`	このリリースを実行している場合は、1.17 以降にアップデートする必要があります。

前提条件

vSphere 名前空間の更新を実行する前に、リリース ノートを参照してください。

必要なバイナリをインストールするには、vCenter Server アプライアンスと ESXi ホストをサポート対象バージョンにアップグレードします。vCenter Server のドキュメントの Upgrading the vCenter Server Appliance を参照してください。

注： vSphere 名前空間の更新を実行する際、プロビジョニングされたすべての Tanzu Kubernetes クラスタが実行中の状態である必要があります。Tanzu Kubernetes が作成または削除状態である場合は、プロセスが完了するまで待機してから スーパーバイザー クラスタ を更新してください。そうしないと、成功しない場合があります。

注： スーパーバイザー クラスタ を更新すると、そこにデプロイされている Tanzu Kubernetes クラスタのローリング アップデートがトリガされることがあります。Tanzu Kubernetes クラスタの更新を参照してください。

手順

1 vCenter Server に vSphere 管理者としてログインします。

2 [メニュー] - [ワークロー管理] の順に選択します。

3 [名前空間] - [更新] タブの順に選択します。

4 更新する [使用可能なバージョン] を選択します。

たとえば、バージョン `v1.18.2-vsc0.0.5-16762486` を選択します。

注： アップデートは段階的に行う必要があります。1.16 から 1.18 へのアップデートのように、アップデートをスキップしないでください。このアップデート パスは 1.16、1.17、1.18 の順にする必要があります。

5 更新を適用する 1 つ以上の スーパーバイザー クラスタ を選択します。

6 アップデートを開始するには [更新の適用] をクリックします。

7 更新の状態を監視するには、[最近のタスク] ペインを使用します。

スーパーバイザー クラスタ の自動アップグレード

vCenter Server アプライアンスをアップグレードすると、スーパーバイザー クラスタ を自動的にアップグレードできます。

vSphere with Tanzu コンポーネントには、vCenter Server 内のコンポーネント、Kubernetes コンポーネント、および ESXi コンポーネントが含まれています。vCenter Server をアップグレードすると、vCenter Server の vSphere with Tanzu コンポーネントのみがアップグレードされます。Kubernetes コンポーネントと ESXi コンポーネントは手動でアップグレードする必要があります。

自動アップグレード機能を使用すると、vCenter Server のアップグレード時に スーパーバイザー クラスタ を自動的にアップグレードすることができます。vCenter Server リリースのリストについては、ナレッジベースの記事 https://kb.vmware.com/s/article/2143838 を参照してください。

vCenter Server をアップグレードすると、スーパーバイザー クラスタ と vCenter Server のバージョン間の互換性を確認するための事前チェックが実行されます。次のシナリオでは、警告が表示されます。

- ターゲットの vCenter Server に Kubernetes のバージョンがあり、スーパーバイザー クラスタ が 1 つ前のバージョンになってしまう可能性がある。このシナリオでは、vCenter Server のアップグレードを続行すると、クラスタが自動的にアップグレードされます。

  たとえば、スーパーバイザー クラスタ のバージョンは 1.16 で、ターゲットの vCenter Server のバージョンは 1.17、1.18、または 1.19 になります。

- ターゲットの vCenter Server に Kubernetes のバージョンがあり、スーパーバイザー クラスタ が 2 つ以上前のバージョンになってしまう可能性がある。このシナリオでは、vCenter Server はアップグレードされません。

  たとえば、スーパーバイザー クラスタ のバージョンは 1.15 で、ターゲットの vCenter Server のバージョンは 1.17、1.18、または 1.19 になります。

- スーパーバイザー クラスタ のライセンスの期限が切れている。このシナリオでは、vCenter Server のアップグレードを続行すると、ライセンスの期限の切れた スーパーバイザー クラスタ が使用できなくなり、リカバリ不能になります。

kubectl 向けの vSphere プラグイン の更新

vSphere 名前空間 のアップデートを実行し、スーパーバイザー クラスタ をアップグレードしたら、kubectl 向けの vSphere プラグイン をアップデートします。

最新バージョンの kubectl 向けの vSphere プラグイン は、Tanzu Kubernetes クラスタのルート CA 証明書を *TANZU-KUBERNETES-NAME*-ca という名前の Kubernetes シークレットにダウンロードしてインストールします。プラグインは、この証明書を使用して、対応するクラスタの認証局データストアの CA 情報を入力します。

kubectl 向けの vSphere プラグイン をダウンロードしてインストールするには、vSphere 向け Kubernetes CLI Tools のダウンロードとインストールを参照してください。*TANZU-KUBERNETES-CLUSTER-NAME*-ca シークレットの詳細については、Tanzu Kubernetes クラスタのシークレットの取得を参照してください。

更新のための Tanzu Kubernetes クラスタ互換性の確認

Tanzu Kubernetes リリース は、Tanzu Kubernetes Grid サービス によってプロビジョニングされている Tanzu Kubernetes クラスタのために VMware が署名した、サポート対象の Kubernetes ソフトウェアを配布します。

Tanzu Kubernetes リリース リリース ノート

すべての Tanzu Kubernetes リリース のリストについては、リリース ノートを参照してください。

更新のための Tanzu Kubernetes クラスタ互換性の確認

vCenter Server と スーパーバイザー クラスタ を含む vSphere with Tanzu インフラストラクチャをアップグレードする前に、Tanzu Kubernetes クラスタがアップグレードに対応していないかどうかを確認する方法については、ナレッジベースの記事 https://kb.vmware.com/s/article/82592 を参照してください。Tanzu Kubernetes クラスタがターゲット インフラストラクチャと互換性がない場合は、クラスタをアップグレードしてから、システムをアップグレードします。

Kubectl を使用した使用可能な Tanzu Kubernetes リリース の一覧表示

次のコマンドを使用して Tanzu Kubernetes リリース を一覧表示し、各リリースの互換性と更新可能性を確認できます。

```
kubectl get tanzukubernetesreleases
```

COMPATIBLE 列は、その Tanzu Kubernetes リリース が現在の スーパーバイザー クラスタ と互換性があるかどうかを示します。UPDATES AVAILABLE 列は、Kubernetes の利用可能なアップグレードがあるかどうかと、使用する次の推奨 Tanzu Kubernetes リリース を示します。

```
NAME                                VERSION                          READY   COMPATIBLE
CREATED   UPDATES AVAILABLE
v1.18.15---vmware.1-tkg.1.600e412   1.18.15+vmware.1-tkg.1.600e412   True    True
```

```
21h       [1.19.7+vmware.1-tkg.1.fc82c41]
v1.19.7---vmware.1-tkg.1.fc82c41    1.19.7+vmware.1-tkg.1.fc82c41    True    True
21h       [1.20.2+vmware.1-tkg.1.1d4f79a]
v1.20.2---vmware.1-tkg.1.1d4f79a    1.20.2+vmware.1-tkg.1.1d4f79a    True    True         21h
```

これと同じ情報は、`kubectl get tkc <tkgs-cluster-name>` を使用して入手することもできます。

Tanzu Kubernetes クラスタの更新

Kubernetes バージョンなど、Tanzu Kubernetes クラスタのローリング アップデートを開始するには、Tanzu Kubernetes リリース、仮想マシン クラス、またはストレージ クラスを更新します。

Tanzu Kubernetes クラスタの更新準備のチェックリスト

Tanzu Kubernetes クラスタの更新を実行する前に、前提条件となる次のタスクを完了します。

手順	操作
1	vSphere with Tanzu リリース ノートを参照します。
2	Tanzu Kubernetes リリース リリース ノートを参照します。
3	vSphere with Tanzu vSphere with Tanzu の更新についてを確認します。
4	ターゲット アップグレード バージョンとの Tanzu Kubernetes 更新のための Tanzu Kubernetes クラスタ互換性の確認を確認します。
5	Tanzu Kubernetes Grid サービス API ターゲット バージョン（TKGS v1alpha2 API を使用した Tanzu Kubernetes クラスタのプロビジョニング など）の機能と、現行バージョン（Tanzu Kubernetes Grid サービス v1alpha1 API を使用した Tanzu Kubernetes クラスタのプロビジョニング など）の機能を確認します。（以下の重要な注記を参照してください。）
6	プロビジョニングしたすべての Tanzu Kubernetes クラスタが Tanzu Kubernetes クラスタのライフサイクル ステータスの表示状態であることを確認します。
7	スーパーバイザー クラスタ と Tanzu Kubernetes Grid サービス をアップグレードする vSphere 名前空間 vSphere 名前空間の更新の実行による スーパーバイザー クラスタ の更新を実行します。
8	Tanzu Kubernetes クラスタの Tanzu Kubernetes クラスタのローリング アップデートの開始オプションを確認します。
9	クラスタ マニフェストを更新するためにクラスタ マニフェストの編集方法を確認します。

重要： vSphere with Tanzu バージョン 7 Update 3、具体的には スーパーバイザー クラスタ バージョン `v1.21.0+vmware.wcp.2` には、Tanzu Kubernetes Grid サービス v1alpha2 API への自動アップグレードが含まれています。Tanzu Kubernetes クラスタ仕様の一部のフィールドは推奨されておらず、Kubernetes バージョンをアップグレードする前にクラスタ マニフェストを手動で編集しなければならないことがあります。クラスタ仕様を TKGS v1alpha2 API に変換した後の Tanzu Kubernetes リリースの更新を参照してください。

Tanzu Kubernetes クラスタのローリング アップデートの開始

ローリング アップデートを開始するには、`TanzuKubernetesCluster` の仕様に対して次の 1 つまたは複数の変更を加えます。

- Tanzu Kubernetes リリース バージョンのアップグレードによる Tanzu Kubernetes クラスタのアップデート
- VirtualMachineClass の変更による Tanzu Kubernetes クラスタの更新
- ストレージ クラスの変更による Tanzu Kubernetes クラスタの更新

注： これらは、ローリング アップデートを開始するための最も一般的な方法ですが、唯一のものではありません。いずれかの構成要素を変更して、ローリング アップデートを開始することもできます。たとえば、配布バージョンに対応する `VirtualMachineImage` を名前変更または置換すると、システムが新しいイメージで実行されているすべてのノードの取得を試行するため、ローリング アップデートが開始されます。スーパーバイザー クラスタ を更新した場合も、そこにデプロイされている Tanzu Kubernetes クラスタのローリング アップデートがトリガされることがあります。たとえば、`vmware-system-tkg-controller-manager` が更新された場合、システムは新しい値をマニフェスト ジェネレータに導入し、コントローラはこれらの値をデプロイするローリング アップデートを開始します。

クラスタ マニフェストの編集方法

クラスタを更新するには、クラスタのマニフェストを更新する必要があります。これを行う方法は、次のようにいくつかあります。

- `kubectl edit tanzukubernetescluster/CLUSTER-NAME` コマンドを使用する方法。このコマンドを実行すると、KUBE_EDITOR または EDITOR 環境変数によって定義されたテキスト エディタで、クラスタ マニフェスト全体が開きます。ファイルを保存すると、変更が反映されてクラスタが更新されます。`kubectl edit` コマンドの詳細については、Kubernetes のドキュメントの edit command を参照してください。`kubectl edit` 方式を使用するには、次のトピックを参照してください。
  - Tanzu Kubernetes リリース バージョンのアップグレードによる Tanzu Kubernetes クラスタのアップデート
  - VirtualMachineClass の変更による Tanzu Kubernetes クラスタの更新
  - ストレージ クラスの変更による Tanzu Kubernetes クラスタの更新
- `kubectl patch` コマンドを使用する方法。このコマンドは、クラスタの「インプレース」更新を実行します。このコマンドの目的は、Kubernetes のバージョンを更新する方法を提供することです。ここでは、その方法について説明します。`kubectl patch` コマンドの詳細については、Kubernetes のドキュメントの Update API Objects in Place Using kubectl patch を参照してください。`kubectl patch` 方式を使用するには、次のトピックを参照してください。
  - パッチ方式を使用した Tanzu Kubernetes クラスタの更新
- 手動で更新したローカル YAML ファイルを `kubectl apply` コマンドから使用する方法。この方法には TKGS v1alpha2 API を使用して Tanzu Kubernetes クラスタをプロビジョニングするためのワークフローと似ているという利点がありますが、現在のクラスタの YAML にアクセスできない場合は問題が発生するため、推奨されません。

Tanzu Kubernetes リリース バージョンのアップグレードによる Tanzu Kubernetes クラスタのアップデート

Tanzu Kubernetes リリース バージョンをアップグレードして、Tanzu Kubernetes クラスタをアップデートします。

Tanzu Kubernetes リリース バージョンをアップグレードすると、Tanzu Kubernetes クラスタのローリング アップデートを開始できます。実行方法は、使用中の Tanzu Kubernetes Grid サービス API のバージョンによって異なります。

TKGS API バージョン	バージョンのアップデート方法
v1alpha2 API	クラスタ マニフェストの `spec.topology.controlPlane.tkr.refernece.name` プロパティと `spec.topology.nodePools[*].tkr.reference.name` プロパティの TKR NAME 文字列をアップデートします。クラスタ仕様を TKGS v1alpha2 API に変換した後の Tanzu Kubernetes リリースの更新を参照してください。
v1alpha1 API	クラスタ マニフェストの `spec.distribution.version` プロパティと `spec.distribution.fullVersion` プロパティの DISTRIBUTION バージョンをアップデートします。下記を参照してください。

前提条件

Tanzu Kubernetes クラスタをアップデートするための前提条件が完了していることを確認します。Tanzu Kubernetes クラスタの更新を参照してください。

このタスクでは、`kubectl edit tanzukubernetescluster/CLUSTER-NAME` コマンドを使用してクラスタのマニフェストを更新します。kubectl edit コマンドを実行すると、KUBE_EDITOR または EDITOR の環境変数によって定義されたテキスト エディタ内にクラスタ マニフェストが開きます。ファイルを保存すると、変更が反映されてクラスタが更新されます。Kubectl のデフォルトのテキスト エディタの指定を参照してください。

手順

1 スーパーバイザー クラスタ で認証します。vCenter Single Sign-On ユーザーとして スーパーバイザー クラスタ に接続するを参照してください。

```
kubectl vsphere login --server=IP-ADDRESS --vsphere-username USERNAME
```

2 ターゲット Tanzu Kubernetes クラスタがプロビジョニングされている vSphere 名前空間 にコンテキストを切り替えます。

```
kubectl config use-context SUPERVISOR-NAMESPACE
```

3 ターゲット Tanzu Kubernetes クラスタおよびバージョンを取得します。

```
kubectl get tanzukubernetescluster
```

たとえば、TKGS v1alpha2 API を使用した結果は次のようになります。

```
kubectl get tanzukubernetescluster
NAMESPACE        NAME            CONTROL PLANE   WORKER   TKR
NAME                             AGE    READY   TKR COMPATIBLE   UPDATES AVAILABLE
tkgs-cluster-1   test-cluster    3               3        v1.21.2---vmware.1-
tkg.1.ee25d55     38h    True    True             [1.21.2+vmware.1-tkg.1.ee25d55]
```

たとえば、TKGS v1alpha1 API を使用した結果は次のようになります。

```
kubectl get tanzukubernetescluster
NAME             CONTROL PLANE   WORKER   DISTRIBUTION                      AGE   PHASE
tkgs-cluster-1   3               3        v1.19.16+vmware.1-tkg.1.df910e2   19h   running
```

4 使用できる Tanzu Kubernetes リリースをリストします。

```
kubectl get tanzukubernetesreleases
```

5 次のコマンドを実行して、クラスタのマニフェストを編集します。

```
kubectl edit tanzukubernetescluster/CLUSTER-NAME
```

6 Tanzu Kubernetes リリース をアップデートしてマニフェストを編集します。実行方法は、使用している TKGS API のバージョンによって異なります。

TKGS v1alpha2 API を使用している場合は、TKR NAME 文字列を更新します。クラスタ仕様を TKGS v1alpha2 API に変換した後の Tanzu Kubernetes リリースの更新を参照してください。

TKGS v1alpha1 API を使用している場合は、たとえば次のようにマニフェストを更新します。更新前：

```
spec:
  distribution:
    fullVersion: v1.19.16+vmware.1-tkg.1.df910e2
    version: v1.19.16
```

更新後：

```
spec:
  distribution:
    fullVersion: null
    version: v1.20.12
```

注： TKGS v1alpha1 API を使用している場合は、完全修飾バージョンを指定するか、バージョンのショートカットを指定できます。たとえば、`version: v1.20.12` と指定すると、そのパッチ バージョンに一致する最新のイメージに解決され、`version: v1.20` と指定すると、一致する最新のパッチ バージョンに解決されます。解決されたバージョンは、クラスタ マニフェストに `fullVersion` として表示されます。ショートカットでバージョンをアップグレードするには、検出中にバージョンの不一致が生じないよう、`fullVersion` を設定解除するか、null にする必要があります。

7 マニフェスト ファイルに行った変更内容を保存します。

ファイルを保存すると、kubectl によって変更がクラスタに適用されます。バックグラウンドで、スーパーバイザー クラスタの仮想マシン サービスによって新しいワーカー ノードがプロビジョニングされます。

8 kubectl から、マニフェストの編集が正常に記録されたことが報告されているかを確認します。

```
kubectl edit tanzukubernetescluster/tkgs-cluster-1
tanzukubernetescluster.run.tanzu.vmware.com/tkgs-cluster-1 edited
```

注： エラーが表示された場合、またはクラスタ マニフェストが正常に編集されたことが kubectl から報告されない場合は、KUBE_EDITOR 環境変数を使用して、デフォルトのテキスト エディタが適切に構成されていることを確認してください。Kubectl のデフォルトのテキスト エディタの指定を参照してください。

9 クラスタが更新されていることを確認します。

```
kubectl get tanzukubernetescluster
NAME             CONTROL PLANE   WORKER   DISTRIBUTION                      AGE   PHASE
tkgs-cluster-1   3               3        v1.20.12+vmware.1-tkg.1.b9a42f3   21h   updating
```

10 クラスタが更新されたことを確認します。

```
kubectl get tanzukubernetescluster
NAME             CONTROL PLANE   WORKER   DISTRIBUTION                      AGE   PHASE
tkgs-cluster-1   3               3        v1.20.12+vmware.1-tkg.1.b9a42f3   22h   running
```

VirtualMachineClass の変更による Tanzu Kubernetes クラスタの更新

クラスタ ノードのホストに使用されている仮想マシンのクラスを変更することで、Tanzu Kubernetes クラスタを更新できます。

Tanzu Kubernetes Grid サービス では、`VirtualMachineClass` の定義の変更によるクラスタの更新がサポートされています。この変更を行うと、この新しいクラスを使用して新しいノードがロールアウトされて、古いノードがスピンダウンされます。Tanzu Kubernetes Grid サービス クラスタの更新についてを参照してください。

注： Tanzu Kubernetes クラス他がプロビジョニングされている vSphere 名前空間 に `VirtualMachineClass` をバインドする必要があります。Tanzu Kubernetes クラスタの仮想マシンのクラスを参照してください。

前提条件

このタスクでは、`kubectl edit tanzukubernetescluster/CLUSTER-NAME` コマンドを使用してクラスタのマニフェストを更新します。kubectl edit コマンドを実行すると、KUBE_EDITOR または EDITOR の環境変数によって定義されたテキスト エディタ内にクラスタ マニフェストが開きます。ファイルを保存すると、変更が反映されてクラスタが更新されます。Kubectl のデフォルトのテキスト エディタの指定を参照してください。

手順

1 スーパーバイザー クラスタ で認証します。vCenter Single Sign-On ユーザーとして スーパーバイザー クラスタ に接続するを参照してください。

```
kubectl vsphere login --server=IP-ADDRESS --vsphere-username USERNAME
```

2 ターゲット Tanzu Kubernetes クラスタがプロビジョニングされている vSphere 名前空間 にコンテキストを切り替えます。

```
kubectl config use-context SUPERVISOR-NAMESPACE
```

3 ターゲット Tanzu Kubernetes クラスタを記述し、仮想マシンのクラスを確認します。

```
kubectl describe tanzukubernetescluster CLUSTER-NAME
```

たとえば、このクラスタでは best-effort-medium 型の仮想マシン クラスが使用されています。

```
Spec:
  ...
  Topology:
    Control Plane:
      Class:          best-effort-medium
      ...
    Workers:
      Class:          best-effort-medium
      ...
```

4 名前空間で使用可能な仮想マシン クラスを一覧表示して、記述します。

```
kubectl get virtualmachineclassbindings
```

注： `kubectl get virtualmachineclasses` コマンドは、スーパーバイザー クラスタ にあるすべての仮想マシン クラスを一覧表示します。仮想マシン クラスと vSphere 名前空間 を関連付ける必要がある場合は、ターゲット名前空間にバインドされているこれらの仮想マシン クラスのみを使用します。

5 次のコマンドを実行して、クラスタのマニフェストを編集します。

```
kubectl edit tanzukubernetescluster/CLUSTER-NAME
```

6 検出時にバージョンの不一致が発生しないようにするには、`version` 文字列を変更し、`fullVersion` を設定解除するか null にして、マニフェストを編集します。

たとえば、制御プレーンおよびワーカー ノードに `best-effort-medium` 仮想マシン クラスを使用しないよう、クラスタ マニフェストを変更します。

```
spec:
  topology:
    controlPlane:
      class: best-effort-medium
```

```
      ...
    workers:
      class: best-effort-medium
      ...
```

制御プレーンおよびワーカーノードに guaranteed-large 仮想マシン クラスを使用するには、次の手順を実行します。

```
spec:
  topology:
    controlPlane:
      class: guaranteed-large
      ...
    workers:
      class: guaranteed-large
      ...
```

7 マニフェスト ファイルに行った変更内容を保存します。

ファイルを保存すると、kubectl によって変更がクラスタに適用されます。バックグラウンドで、Tanzu Kubernetes Grid サービス は新しいノード仮想マシンをプロビジョニングし、古い仮想マシンをスピンダウンします。

8 kubectl から、マニフェストの編集が正常に記録されたことが報告されているかを確認します。

```
kubectl edit tanzukubernetescluster/tkgs-cluster-1
tanzukubernetescluster.run.tanzu.vmware.com/tkgs-cluster-1 edited
```

注： エラーが表示された場合、またはクラスタ マニフェストが正常に編集されたことが kubectl から報告されない場合は、KUBE_EDITOR 環境変数を使用して、デフォルトのテキスト エディタが適切に構成されていることを確認してください。Kubectl のデフォルトのテキスト エディタの指定を参照してください。

9 クラスタが更新されていることを確認します。

```
kubectl get tanzukubernetescluster
NAME             CONTROL PLANE   WORKER   DISTRIBUTION                      AGE   PHASE
tkgs-cluster-1   3               3        v1.18.5+vmware.1-tkg.1.c40d30d   21h   updating
```

10 クラスタが更新されたことを確認します。

```
kubectl get tanzukubernetescluster
NAME             CONTROL PLANE   WORKER   DISTRIBUTION                      AGE   PHASE
tkgs-cluster-1   3               3        v1.18.5+vmware.1-tkg.1.c40d30d   22h   running
```

ストレージ クラスの変更による Tanzu Kubernetes クラスタの更新

クラスタ ノードで使用されるストレージ クラスを変更して、Tanzu Kubernetes クラスタを更新できます。

Tanzu Kubernetes Grid サービス では、ノード プールの `StorageClass` の変更、たとえば、`.spec.topology.controlPlane.storageClass` プロパティや `.spec.topology.workers.storageClass` プロパティの変更によって、クラスタを更新することができます。Tanzu Kubernetes Grid サービス クラスタの更新についてを参照してください。

前提条件

このタスクでは、`kubectl edit tanzukubernetescluster/CLUSTER-NAME` コマンドを使用してクラスタのマニフェストを更新します。kubectl edit コマンドを実行すると、KUBE_EDITOR または EDITOR の環境変数によって定義されたテキスト エディタ内にクラスタ マニフェストが開きます。ファイルを保存すると、変更が反映されてクラスタが更新されます。Kubectl のデフォルトのテキスト エディタの指定を参照してください。

手順

1 スーパーバイザー クラスタ で認証します。vCenter Single Sign-On ユーザーとして スーパーバイザー クラスタ に接続するを参照してください。

```
kubectl vsphere login --server=IP-ADDRESS --vsphere-username USERNAME
```

2 ターゲット Tanzu Kubernetes クラスタがプロビジョニングされている vSphere 名前空間 にコンテキストを切り替えます。

```
kubectl config use-context SUPERVISOR-NAMESPACE
```

3 使用可能なストレージ クラスを判別して、使用するクラスを決定するには、次のコマンドを実行します。

```
kubectl describe tanzukubernetescluster CLUSTER-NAME
```

4 次のコマンドを実行して、クラスタのマニフェストを編集します。

```
kubectl edit tanzukubernetescluster/CLUSTER-NAME
```

5 `storageClass` の値を変更して、マニフェストを編集します。

たとえば、制御プレーンおよびワーカー ノードに `silver-storage-class` クラスを使用しないようにクラスタのマニフェストを変更します。

```
spec:
  topology:
    controlPlane:
      ...
      storageClass: silver-storage-class
    workers:
      ...
      storageClass: silver-storage-class
```

制御プレーンおよびワーカー ノードに `gold-storage-class` クラスを使用するには、次の手順を実行します。

```
spec:
  topology:
```

```
    controlPlane:
      ...
      storageClass: gold-storage-class
    workers:
      ...
      storageClass: gold-storage-class
```

6 マニフェスト ファイルに行った変更内容を保存します。

ファイルを保存すると、kubectl によって変更がクラスタに適用されます。バックグラウンドで、Tanzu Kubernetes Grid サービス は新しいノード仮想マシンをプロビジョニングし、古い仮想マシンをスピンダウンします。

7 kubectl から、マニフェストの編集が正常に記録されたことが報告されているかを確認します。

```
kubectl edit tanzukubernetescluster/tkgs-cluster-1
tanzukubernetescluster.run.tanzu.vmware.com/tkgs-cluster-1 edited
```

注： エラーが表示された場合、またはクラスタ マニフェストが正常に編集されたことが kubectl から報告されない場合は、KUBE_EDITOR 環境変数を使用して、デフォルトのテキスト エディタが適切に構成されていることを確認してください。Kubectl のデフォルトのテキスト エディタの指定を参照してください。

8 クラスタが更新されていることを確認します。

```
kubectl get tanzukubernetescluster
NAME             CONTROL PLANE   WORKER   DISTRIBUTION                      AGE   PHASE
tkgs-cluster-1   3               3        v1.18.5+vmware.1-tkg.1.c40d30d   21h   updating
```

9 クラスタが更新されたことを確認します。

```
kubectl get tanzukubernetescluster
NAME             CONTROL PLANE   WORKER   DISTRIBUTION                      AGE   PHASE
tkgs-cluster-1   3               3        v1.18.5+vmware.1-tkg.1.c40d30d   22h   running
```

パッチ方式を使用した Tanzu Kubernetes クラスタの更新

`kubectl patch` 方式を使用して、Tanzu Kubernetes クラスタの「インプレース」更新を実行できます。`kubectl edit` コマンドを使用する代わりに `kubectl patch` 方式を使用することで、サポートされているクラスタの更新操作のいずれかを実行できます。

Kubectl Patch コマンドについて

制限: TKGS v1alpha2 API に準拠させる目的で、このトピックの説明にあるように `kubectl patch` を使用してクラスタの仕様を更新しないでください。この種の更新では、`kubectl edit` を使用してください。クラスタ仕様を TKGS v1alpha2 API に変換した後の Tanzu Kubernetes リリースの更新を参照してください。

kubectl patch コマンドは、クラスタの「インプレース」更新を実行します。このコマンドの目的は、Kubernetes のバージョンを更新する方法を提供することです。ここでは、その方法について説明します。kubectl patch コマンドの詳細については、Kubernetes のドキュメントの Update API Objects in Place Using kubectl patch を参照してください。

ここに示すアプローチでは、UNIX シェル コマンド read を使用して、キーボードから入力を行い、$PATCH という変数に割り当てます。kubectl patch コマンドは Kubernetes API を呼び出して、クラスタ マニフェストを更新します。--type=merge フラグは、既存のマニフェストとは異なるプロパティのみがデータに含まれていることを示しています。

パッチ方式を使用した Kubernetes バージョンのアップグレード

ローリング アップデートをトリガする最も一般的な方法は、.spec.distribution.version および .spec.distribution.fullVersion プロパティを使用して、クラスタの Kubernetes 配布バージョンを変更することです。version ヒントを更新し、検出時にバージョンの不一致が発生しないように、fullVersion を設定解除するか、null にします。

```
$ read -r -d '' PATCH <<'EOF'
spec:
  distribution:
    fullVersion: null    # NOTE: Must set to null when updating just the version field
    version: v1.18.5
EOF
```

kubectl patch コマンドを使用して、更新を適用します。クラスタ マニフェストで改行文字を保持するには、"$PATCH" 変数を引用符で囲む必要があります。*TKG-CLUSTER-NAME* の値を、クラスタの実際の名前に置き換えます。

```
kubectl patch --type=merge tanzukubernetescluster TKG-CLUSTER-NAME --patch "$PATCH"
```

予期される結果：

```
tanzukubernetescluster.run.tanzu.vmware.com/TKG-CLUSTER-NAME patched
```

パッチ方式を使用した、ノードの VirtualMachineClass の変更によるクラスタの更新

Tanzu Kubernetes クラスタのローリング アップデートをトリガする別の方法として、ノード プールの VirtualMachineClass を変更することもできます。これは、.spec.topology.controlPlane.class プロパティまたは .spec.topology.workers.class プロパティを変更することによって行われます。

```
read -r -d '' PATCH <<'EOF'
spec:
  topology:
    controlPlane:
```

```
      class: best-effort-xlarge
    workers:
      class: best-effort-xlarge
EOF
```

kubectl patch コマンドを使用して更新を適用し、変数をクラスタ名に置き換えます。

```
kubectl patch --type=merge tanzukubernetescluster TKG-CLUSTER-NAME --patch "$PATCH"
```

予期される結果：

```
tanzukubernetescluster.run.tanzu.vmware.com/TKG-CLUSTER-NAME patched
```

パッチ方式を使用した、ノードの StorageClass の変更によるクラスタの更新

Tanzu Kubernetes クラスタのローリング アップデートをトリガする別の方法として、ノード プールの StorageClass を変更することもできます。これは、.spec.topology.controlPlane.storageClass プロパティまたは .spec.topology.workers.storageClass プロパティを変更することによって行われます。

```
$ read -r -d '' PATCH <<'EOF'
spec:
  topology:
    controlPlane:
      storageClass: gc-storage-profile
    workers:
      storageClass: gc-storage-profile
EOF
```

kubectl patch コマンドを使用して更新を適用し、変数をクラスタ名に置き換えます。

```
kubectl patch --type=merge tanzukubernetescluster TKG-CLUSTER-NAME --patch "$PATCH"
```

予期される結果：

```
tanzukubernetescluster.run.tanzu.vmware.com/TKG-CLUSTER-NAME patched
```

vSphere with Tanzu のバックアップとリストア

18

vSphere ポッド および Tanzu Kubernetes クラスタで実行中のワークロードをバックアップおよびリストアできます。また、vSphere with Tanzu のインストールをサポートする vCenter Server および NSX-T インフラストラクチャで実行中のワークロードをバックアップおよびリストアすることもできます。

この章には、次のトピックが含まれています。

vSphere with Tanzu のバックアップとリストアに関する考慮事項

このトピックでは、vSphere with Tanzu のバックアップおよびリストア プロセスの概要と、vSphere with Tanzu のバックアップおよびリストアの戦略を実践する際の大局的な考慮事項について説明します。

vSphere with Tanzu のバックアップとリストアは、さまざまなレイヤーとツールで構成されています。

次の表に、これらのレイヤーとツールの概要について、ワークロードからインフラストラクチャまでトップダウンで示します。各レイヤーでバックアップとリストアを実行する詳細については、個々のセクションを参照してください。

使用例	Tools	コメント
vSphere ポッドのバックアップおよびリストア	Velero Plugin for vSphere	スーパーバイザー クラスタにプラグインをインストールして構成します。 **注：** プラグインは、スーパーバイザー クラスタの状態をバックアップしません。 スーパーバイザー クラスタ での Velero Plugin for vSphere のインストールと構成を参照してください。 Velero Plugin for vSphere を使用した vSphere ポッド のバックアップとリストアを参照してください。
Tanzu Kubernetes クラスタのステートレス ワークロードとステートフル ワークロードをバックアップし、Tanzu Kubernetes Grid サービスによってプロビジョニングされたクラスタにリストアする	Velero Plugin for vSphere	Kubernetes メタデータとパーシステント ボリュームの両方をバックアップおよびリストアできます。 パーシステント ボリュームには、Velero スナップショット作成（Restic ではない）が使用されます。 Tanzu Kubernetes クラスタでの Velero Plugin for vSphere のインストールと構成を参照してください。 Velero Plugin for vSphere を使用した Tanzu Kubernetes クラスタ ワークロードのバックアップとリストアを参照してください。
Tanzu Kubernetes クラスタのステートレス ワークロードとステートフル ワークロードをバックアップし、Tanzu Kubernetes Grid サービスによってプロビジョニングされた、準拠する Kubernetes クラスタにリストアする	スタンドアローン Velero と Restic	移植性が必要な場合は、スタンドアローン Velero を使用します。ステートフル アプリケーションの場合は、Restic を含める必要があります。 Tanzu Kubernetes クラスタへのスタンドアローン Velero と Restic のインストールと構成を参照してください。 スタンドアローンの Velero と Restic を使用した Tanzu Kubernetes クラスタ ワークロードのバックアップとリストアを参照してください。
スーパーバイザー クラスタ スーパーバイザー クラスタをアップグレードした後に、バックアップを新規に実行する必要があります。古いバージョンのスーパーバイザー クラスタが使用されるバックアップに vCenter Server をリストアすることはできません。	vCenter Server Velero Plugin for vSphere スタンドアローン Velero と Restic	vCenter Server をバックアップからリストアします。vCenter Server は、3 台すべてのスーパーバイザー クラスタ制御プレーン仮想マシンを再作成します。 プラグインまたはスタンドアローン Velero と Restic を使用して、クラスタのワークロードをバックアップからリストアします。

使用例	Tools	コメント
vCenter Server の構成	vCenter Server	vCenter Server が失われた場合は、vCenter Server を使用して vCenter Server オブジェクトをバックアップおよびリストアします。 vCenter Server のバックアップとリストアを参照してください。
NSX-T Data Center	NSX-T Manager	ロード バランサと入力方向サービスは、NSX-T のバックアップに依存します。 NSX-T Manager を使用して、NSX-T データベースをバックアップおよびリストアします。 NSX-T Data Center のバックアップとリストアを参照してください。

スーパーバイザー クラスタ での Velero Plugin for vSphere のインストールと構成

Velero Plugin for vSphere を使用して vSphere ポッド で実行されているワークロードをバックアップおよびリストアするには、スーパーバイザー クラスタ に Velero Plugin for vSphere をインストールします。

概要

Velero Plugin for vSphere は、vSphere with Tanzu ワークロードをバックアップおよびリストアするためのソリューションを提供します。このソリューションを使用するには、いくつかのコンポーネントをインストールして構成する必要があります。スーパーバイザー クラスタ に Velero Plugin for vSphere をインストールして構成すると、vSphere ポッド をバックアップおよびリストアできるようになります。パーシステント ワークロードの場合は、Velero Plugin for vSphere でパーシステント ボリュームのスナップショットを作成できます。

Velero Plugin for vSphere を使用して Tanzu Kubernetes クラスタ ワークロードをバックアップおよびリストアする場合も、スーパーバイザー クラスタ に Velero Plugin for vSphere をインストールすることが前提条件となります。

注： Velero Plugin for vSphere を使用して スーパーバイザー クラスタ の状態をバックアップおよびリストアすることはできません。vSphere with Tanzu のバックアップとリストアに関する考慮事項を参照してください。

注： Velero Plugin for vSphere は、単独では増分バックアップを実行しません。Dell EMC PowerProtect は増分バックアップをサポートし、Velero と Velero Plugin for vSphere を利用します。

前提条件

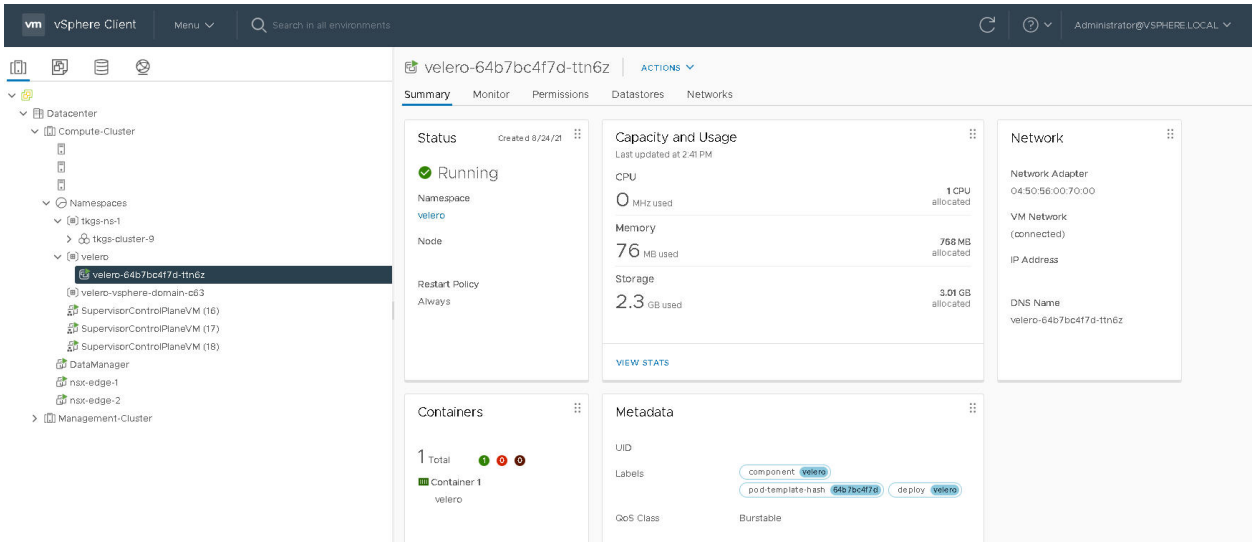
Velero Plugin for vSphere をインストールする前に、次の前提条件を満たす必要があります。

- NSX-T Data Center ネットワークで、ワークロード管理が有効になっている。NSX-T Data Center ネットワークを使用したワークロード管理の有効化を参照してください
- スーパーバイザー クラスタ がバージョン 1.21.1 以降である。
- vSphere 名前空間 が作成され、構成されている。

- vSphere 管理者ロールのメンバーであるか、次の vSphere 権限を持っている必要がある。
  - SupervisorServices.Manage
  - Namespaces.Manage
  - Namespaces.Configure

このスクリーンショットは、Velero Plugin for vSphere インストールの終了状態を示しています。

- NSX-T ネットワークは、vSphere ポッド のデプロイをサポートするために使用されている
- Data Manager 仮想マシンがデプロイされている
- Velero Operator が有効で、`velero-vsphere-domain-cXX` 名前空間で実行されている
- `velero` という名前空間が構成されている
- Velero Plugin for vSphere が `velero` 名前空間内で vSphere ポッド として実行されている



アップグレード

ここに示す手順では、vSphere 7 U3 を実行していることが前提となります。Velero Plugin for vSphere を vSphere 7 U2 P3 環境にすでにインストールしている場合は、アップグレード時に Data Manager 仮想マシンと [Velero vSphere Operator] が新しいフレームワークに移行されます。[Velero vSphere Operator] は、新しい vSphere サービス形式に変換されます。アクションは不要です。

バックアップおよびリストア トラフィック専用ネットワークの作成（オプション）

本番環境でバックアップ トラフィックとリストア トラフィックを vSphere with Tanzu 管理ネットワーク トラフィックから分離することは、必須ではありませんが、推奨です。この操作には、2 つの側面があります。

- ネットワーク ファイル コピー (NFC) をサポートする ESXi ホストにタグを付ける
- NSX-T Data Center を使用してバックアップおよびリストア ネットワークを構成する

専用ネットワーク ブロック デバイス (NBD) の転送をサポートするように vSphere 7.x ESXi ホストを構成するには、ワークロード管理が有効になっている vCenter Server クラスタ内の各 ESXi ホストに VMkernel NIC を追加して、その NIC に `vSphereBackupNFC` を設定します。タグ `vSphereBackupNFC` を VMkernel アダプタの NIC タイプに適用すると、バックアップ トラフィックとリストア トラフィックは選択した仮想 NIC を経由するようになります。

この構成を実行するには、Virtual Disk Development Kit を使用します。NBD のドキュメントを参照してください。

注： `vSphereBackupNFC` が VMkernel NIC で有効でない場合は、バックアップとリストアのネットワークが構成されているときでも、バックアップとリストアのトラフィックはこれらのネットワークを経由しません。`vSphereBackupNFC` が有効でない場合、トラフィックは vSphere 管理ネットワークを経由します。

`vSphereBackupNFC` タグを有効にしたら、クラスタの既存の vSphere Distributed Switch (vDS) を次のように更新して、NSX-T を使用してバックアップおよびリストア ネットワークを構成します。

- vSphere Client で [メニュー] - [ネットワーク] の順に選択します。
- クラスタの既存の Distributed Switch を選択します。
- Distributed Switch を右クリックして、[分散ポート グループ] - [新規分散ポート グループ] の順に選択します。
- [BackupRestoreNetwork] という名前の新しい分散ポート グループを作成します。
- [BackupRestoreNetwork] 分散ポート グループに VMkernel アダプタを追加します。
- ワークロード管理が有効になっている vCenter Server クラスタ内のすべての ESXi ホストを [BackupRestoreNetwork] 分散ポート グループに接続します。
- `vSphereBackupNFC` タグを有効にします。

既存の Distributed Switch で NSX-T ネットワークを作成する手順については、vSphere with Tanzu で使用する NSX-T Data Center のインストールと構成を参照してください。

S3 互換オブジェクト ストアの作成

パーシステント ボリュームのバックアップおよびリストアを行うには、S3 互換オブジェクト ストアを指定する必要があります。Velero は、複数のオブジェクト ストア プロバイダをサポートしています。

Velero Plugin for vSphere をインストールするには、S3 互換オブジェクト ストアに関する次の情報を指定する必要があります。

データ項目	値の例
s3Url	http://my-s3-store.example.com
aws_access_key_id	ACCESS-KEY-ID-STRING
aws_secret_access_key	SECRET-ACCESS-KEY-STRING

次の情報を使用して、`s3-credentials` という名前のシークレット ファイルを作成します。このファイルは、Velero Plugin for vSphere をインストールするときに参照します。

```
[default]
aws_access_key_id = ACCESS-KEY-ID-STRING
aws_secret_access_key = SECRET-ACCESS-KEY-STRING
```

MinIO は、インストールと使用が容易な S3 互換オブジェクト ストアです。vSphere with Tanzu には、ユーザーが有効にできる MinIO スーパーバイザー サービスが付属しています。詳細については、『vSphere with Tanzu でのステートフル サービスの有効化』を参照してください。

Linux 仮想マシンに MinIO サーバを手動でインストールすることもできます。手順については、Tanzu Kubernetes クラスタへのスタンドアローン Velero と Restic のインストールと構成を参照してください。

Data Manager のインストールと構成

Velero Plugin for vSphere を使用してバックアップとリストアを容易にするには、1 台以上の Data Manager 仮想マシンをデプロイして、S3 互換オブジェクト ストレージとの間でパーシステント ボリューム バックアップ データを移動します。Data Manager は、バックアップ時には vSphere ボリュームから耐久性のあるリモート S3 互換ストレージにボリューム スナップショット データを移動し、リストア時にはリモート S3 互換ストレージから vSphere ボリュームに移動します。

vSphere with Tanzu 環境で、Data Manager を仮想マシンとしてインストールします。

注： Velero vSphere Operator を有効にするまで、Data Manager 仮想マシンをパワーオンしないでください。

1 Data Manager OVA をダウンロードします。

   https://vsphere-velero-datamgr.s3-us-west-1.amazonaws.com/datamgr-ob-17253392-photon-3-release-1.1.ova

2 vSphere Client を使用して、ワークロード管理が有効になっている [データセンター] を右クリックし、[OVF テンプレートのデプロイ] を選択します。

3 ダウンロードした Data Manager OVA ファイルを選択して、vCenter Server にアップロードします。

4 たとえば、仮想マシンに **`DataManager`** という名前を付けます。

5 コンピューティング リソースとして、スーパーバイザー クラスタ が構成されている vCenter Server クラスタを選択します。

6 仮想マシンのデプロイの詳細を確認し、[次へ] をクリックします。

7 使用許諾契約書を承諾して、[次へ] をクリックします。

8 ストレージを選択して、[次へ] をクリックします。

9 Data Manager 仮想マシンのターゲット ネットワークを選択します。

   - 専用のバックアップおよびリストア ネットワークが構成されている場合は、[BackupRestoreNetwork] を選択します。バックアップおよびリストア トラフィック専用ネットワークの作成（オプション）を参照してください。

- 専用のバックアップおよびリストア ネットワークが構成されていない場合は、[管理ネットワーク] を選択します。

10 選択内容を確認し、[終了] をクリックしてプロセスを完了します。

11 [最近のタスク] パネルを使用して、デプロイの進行状況を監視します。

注： 「OVF 記述子が利用できません。」というエラーが表示される場合は、Chrome ブラウザを使用します。

12 Data Manager 仮想マシンがデプロイされたら、仮想マシンの入力パラメータを構成します。

13 仮想マシンを右クリックして、[設定の編集] を選択します。

14 [仮想ハードウェア] タブで、[CD/DVD ドライブ] を [ホスト デバイス] から [クライアント デバイス] に変更します。

注： この操作を行わないと、必要な詳細設定を保存できません。

15 [設定の編集] - [仮想マシン オプション] タブで [詳細] - [構成パラメータの編集] の順に選択します。

16 次の各設定の入力パラメータを構成します。

パラメータ	値
`guestinfo.cnsdp.vcUser`	仮想マシンのデプロイに必要な権限を持っている vCenter Server ユーザーの名前を入力します。
`guestinfo.cnsdp.vcAddress`	vCenter Server IP アドレスまたは FQDN を入力します。
`guestinfo.cnsdp.vcPasswd`	vCenter Server ユーザー パスワードを入力します。
`guestinfo.cnsdp.vcPort`	デフォルトは **`443`** です。この値は変更しないでください。
`guestinfo.cnsdp.wcpControlPlaneIP`	スーパーバイザー クラスタの IP アドレスを入力します。 この値を取得するには、ワークロード管理 が有効になっている vCenter Server クラスタに移動して、[構成] - [名前空間] - [ネットワーク] - [管理ネットワーク] - [開始 IP アドレス] の順に選択します。
`guestinfo.cnsdp.updateKubectl`	デフォルトは **`false`** です。この値は変更しないでください。
`guestinfo.cnsdp.veleroNamespace`	デフォルトは **`velero`** です。変更する理由が特にない限り、そのままにします。プロセスの後半で、スーパーバイザー クラスタに `velero` という名前の vSphere 名前空間を作成します。これらの名前が一致する必要があります。
`guestinfo.cnsdp.datamgrImage`	構成されていない（未設定の）システムでは、デフォルトで、`vsphereveleroplugin/data-manager-for-plugin:1.1.0` にある Docker ハブからコンテナ イメージが取得されます。

17 [OK] をクリックして構成を保存し、もう一度 [OK] をクリックして仮想マシンの設定を保存します。

注： [CD/DVD ドライブ] を [ホスト デバイス] から [クライアント デバイス] に変更しなかった場合は、設定を保存できません。この場合は、操作をキャンセルしてドライブを変更し、詳細設定を繰り返します。

18 Velero vSphere Operator を有効にするまで（次のセクションまで）、Data Manager 仮想マシンをパワーオンしないでください。

スーパーバイザー クラスタ への Velero vSphere Operator サービスのインストール

vSphere with Tanzu は、Velero vSphere Operator を vSphere サービスとして提供します。Velero vSphere Operator サービスは Velero Plugin for vSphere と連携して、パーシステント ボリュームのスナップショット作成など、Kubernetes ワークロードのバックアップとリストアをサポートします。vSphere サービスの詳細については、8 章 vSphere with Tanzu を使用した スーパーバイザー サービス の管理を参照してください。

次の操作を実行して、[Velero vSphere Operator] の仕様を [ワークロード管理] が有効になっている vCenter Server に登録し、[Velero vSphere Operator] を スーパーバイザー クラスタ にサービスとしてインストールします。

注： [Velero vSphere Operator] は vSphere ポッド として実行されるため、NSX-T ネットワークが必要です。

1 次の場所から [Velero vSphere Operator] の YAML をダウンロードします。

   http://vmware.com/go/supervisor-service

   サービス仕様ファイルの名前は、`velero-supervisorservice-1.0.0.yaml` です。

2 vSphere Client ホーム メニューから、[ワークロード管理] を選択します。

3 [サービス] タブを選択します。

4 上部のドロップダウン メニューからターゲット vCenter Server を選択します。

5 ダウンロードしたサービス仕様ファイル `velero-supervisorservice-1.0.0.yaml` を [サービスの新規追加] カードにドラッグ アンド ドロップします。

   [追加] をクリックし、`velero-supervisorservice-1.0.0.yaml` ファイルを参照して選択することもできます。

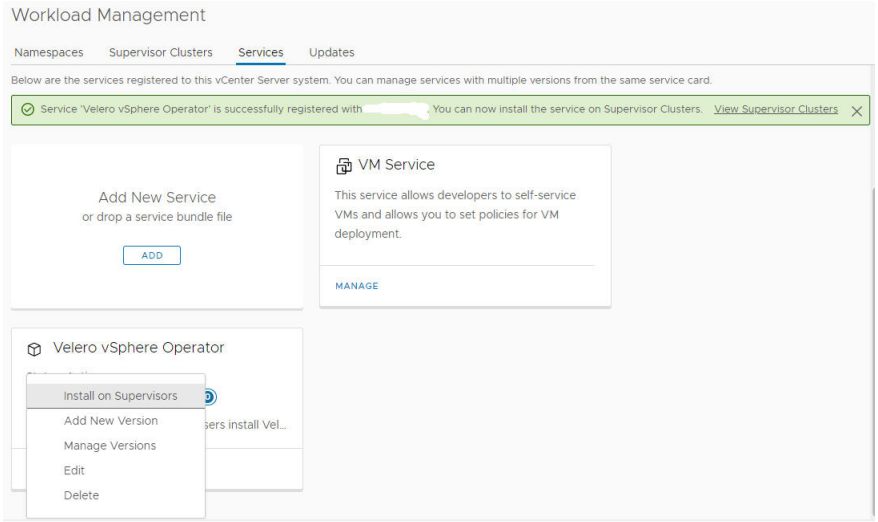
6 [次へ] をクリックして、使用許諾契約書に同意します。

7 [終了] をクリックします。

   [Velero vSphere Operator] が vCenter Server に登録されています。サービスが [有効] な状態になっていることを確認します。サービスが非アクティブな状態になっている場合は、インストールできません。

8 [サービス] タブで [Velero vSphere Operator] の仕様を見つけます。

9 [アクション] - [スーパーバイザーへのインストール] の順にクリックします。



10 サービスをインストールするターゲット スーパーバイザー クラスタ を選択します。

注： スーパーバイザー クラスタ が表示されない場合は、NSX-T ネットワークを使用していることを確認します。

11 次のように、[Velero vSphere Operator] サービスのインストールを構成します。

a ドロップダウンからバージョン [1.1.0] を選択します。

b [リポジトリ エンドポイント] を指定しないでください。

c ユーザー名またはパスワードを入力しないでください。

d [次へ] をクリックします。

12 [完了] をクリックして、サービスのインストールを完了します。

スーパーバイザー クラスタ で [Velero vSphere Operator] サービスを確認し、Data Manager 仮想マシンを起動します。

1 vSphere Client ホーム メニューから、[インベントリ] を選択します。

2 [ワークロード管理] が有効になっている vCenter Server クラスタを指定します。

3 [構成] - [vSphere サービス] - [概要] の順に選択します。

4 [Velero vSphere Operator] がインストールされていて、ステータスが [設定済み] になっていることを確認します。

5 [名前空間] リソース プールに、`svc-velero-vsphere-domain-xxx` という名前の新しい名前空間が表示されていることを確認します。`xxx` は一意の英数字トークンです。これは、[Velero vSphere Operator] 用にシステムによって作成された名前空間です。

注： この名前空間を構成する必要はありません。編集しないでください。

6 [ホストおよびクラスタ] ビューで [Data Manager] 仮想マシンを選択します。

7 [Data Manager] 仮想マシンを右クリックして、パワーオンします。

Velero Plugin for vSphere の vSphere 名前空間の作成

vSphere Client を使用して、スーパーバイザー クラスタ で vSphere 名前空間 を手動で作成します。Velero Plugin for vSphere を使用するには、この名前空間が必要です。ガイダンスについては、vSphere 名前空間 の作成と設定を参照してください。

- 名前空間に **velero** という名前を付けます。
- [velero] 名前空間を選択して、構成します。
- [velero] 名前空間のストレージを指定します。
- 適切な権限を持つユーザーに、[velero] 名前空間に対する編集権限を付与します。

Velero Plugin for vSphere のインストール

これで、Velero Plugin for vSphere をインストールする準備ができました。インストールするには、[velero-vsphere] CLI をダウンロードして実行します。

注： この手順には Linux 仮想マシンが必要です。`kubectl-vsphere` と `kubectl` CLI を実行する Linux ジャンプ ホストに [velero-vsphere] をダウンロードして、実行する必要があります。

1 この CLI を実行できる Linux 仮想マシンを作成します。または、既存の Linux ジャンプ ホストを使用して スーパーバイザー クラスタ にアクセスします。

2 次の場所から Velero Plugin for vSphere CLI をダウンロードします。

https://github.com/vmware-tanzu/velero-plugin-for-vsphere/releases/download/v1.1.0/velero-vsphere-1.1.0-linux-amd64.tar.gz

3 CLI を Linux ジャンプ ホストに安全にコピーします。例：

```
pscp -P 22 C:\temp\velero-vsphere-1.1.0-linux-amd64.tar.gz ubuntu@10.117.29.131:/home/ubuntu/tanzu
```

4 `velero-vsphere` CLI を抽出して、書き込み可能にします。

```
tar -xf velero-vsphere-1.1.0-linux-amd64.tar.gz
chmod +x velero-vsphere
```

5 次の内容の `s3-credentials` ファイルを作成します。

```
aws_access_key_id = ACCESS-KEY-ID-STRING
aws_secret_access_key = SECRET-ACCESS-KEY-STRING
```

6 S3 互換オブジェクト ストアのリージョン、URL、バケット名を取得します。

7 kubectl 向けの vSphere プラグイン を使用して、スーパーバイザー クラスタ にログインします。

8 コンテキストを スーパーバイザー クラスタ に切り替えます。

```
kubectl config use-context SUPERVISOR-CLUSTER-IP-ADDRESS
```

9 次の `velero-vsphere` CLI コマンドを実行して、作成した [velero] 名前空間に Velero Plugin for vSphere をインストールします。

BUCKET-NAME、**REGION**（2 つのインスタンス）、および **s3Url** フィールドのプレースホルダ値を適切な値に置き換えます。前述のいずれかの手順に従わなかった場合は、シークレット ファイルの名前や場所、手動で作成した `velero` 名前空間の名前などの値を調整します。

```
./velero-vsphere install \
    --namespace velero \
    --image velero/velero:v1.5.1 \
    --provider aws \
    --plugins velero/velero-plugin-for-aws:v1.1.0,vsphereveleroplugin/velero-plugin-for-
vsphere:1.1.0 \
    --bucket BUCKET-NAME \
    --secret-file s3-credentials \
    --snapshot-location-config region=REGION \
    --backup-location-config region=REGION,s3ForcePathStyle="true",s3Url=http://my-s3-
store.example.com
```

注： たとえば、`vsphereveleroplugin/velero-plugin-for-vsphere:v1.1.1` や `vsphereveleroplugin/velero-plugin-for-vsphere:v1.2.0` のように、スーパーバイザー クラスタ で Velero Plugin for vSphere v1.1.0 以降を使用することができます。Velero のバージョンは、v1.5.1 (`velero/velero:v1.5.1`) である必要があります。

10 Velero Plugin for vSphere が正常にインストールされたことを確認します。

インストールが正常に完了すると、次のメッセージが表示されます。

```
Send the request to the operator about installing Velero in namespace velero
```

次のコマンドを実行して、さらに確認します。「完了」と表示されて、バージョンが表示されます。

```
kubectl -n velero get veleroservice default -o json | jq '.status'
```

予期される結果：

```
{
  "enabled": true,
  "installphase": "Completed",
  "version": "v1.5.1"
}
```

注： 上記のコマンドでは、ターミナルに送信される JSON 出力をフォーマットする `jq` ユーティリティがインストールされていることが前提となります。`jq` がインストールされていない場合は、インストールするか、コマンドのこの部分（`json` 以降のすべて）を削除します。

11 必要に応じてトラブルシューティングを行います。

インストールに失敗した場合は、インストールを削除して、再試行してください。インストールを削除するには、次のセクションに記載されている手順を記載されている順序で実行します。

エアギャップ環境への Velero プラグインのインストール

エアギャップ環境に Velero Plugin for vSphere をインストールする場合は、カスタマイズされたイメージとともにインストールする必要があります。カスタマイズされたイメージのうち `backup-driver` および `data-manager-for-plugin` の対応するイメージが想定されるレジストリ内にあり、Kubernetes クラスタからアクセスできることを確認する必要があります。エアギャップ環境では、Docker ハブ内のリリース済みイメージにアクセスできないため、プライベート レジストリ内にあるカスタマイズ済みイメージが使用されます。

プラグインをインストールするには、次の手順を実行します。

1 `velero-plugin-for-vsphere`、`backup-driver`、および `data-manager-for-plugin` のリリース済みイメージをダウンロードします。

2 イメージの名前を変更します。つまり、対応する `<Registry endpoint and path>` および `<Version tag>` をイメージにタグ付けして、カスタマイズされたリポジトリにアップロードします。

3 カスタマイズした `velero-plugin-for-vsphere` イメージを使用してプラグインをインストールします。

vanilla クラスタに Velero Plugin for vSphere をインストールすると、`backup-driver` 環境と `data-manager-for-plugin` デーモンセットという 2 つの追加コンポーネントがバックグラウンドでデプロイされます。スーパーバイザー クラスタ および Tanzu Kubernetes クラスタでは、`backup-driver` 環境のみがデプロイされます。

`velero-plugin-for-vsphere` のコンテナ イメージを指定した場合は、イメージ解析メカニズムを使用して対応する `backup-driver` および `data-manager-for-plugin` イメージが解析されます。

コンテナ イメージは、次のパターンの形式になります。

```
<Registry endpoint and path>/<Container name>:<Version tag>
```

`velero-plugin-for-vsphere` コンテナ イメージを指定した場合は、`backup-driver` および `data-manager-for-plugin` の対応するイメージのうち、`<Registry endpoint and path>` および `<Version tag>` が一致しているものが解析されます。

たとえば、次の velero-plugin-for-vsphere コンテナ イメージについて考えます。

```
abc.io:8989/x/y/.../z/velero-plugin-for-vsphere:vX.Y.Z
```

次に示す backup-driver と data-manager-for-plugin の対応するイメージが取得されると想定されます。

```
abc.io:8989/x/y/.../z/backup-driver:vX.Y.Z
abc.io:8989/x/y/.../z/data-manager-for-plugin:vX.Y.Z
```

4 インストールのトラブルシューティングを行います。

backup-driver と data-manager-for-plugin の対応するイメージの解析中に問題やエラーが発生した場合、インストールは Docker ハブの公式 velerovspherepluginリポジトリ内の対応するイメージにフォールバックされます。次の問題が発生すると、フォールバック メカニズムがトリガされます。

a ユーザー入力内のカスタマイズされた velero-plugin-for-vsphere イメージに、予期しないコンテナ名が使用されています。

たとえば、x/y/velero-velero-plugin-for-vsphere:v1.1.1 が使用されています。

b Velero のデプロイ名が、velero 以外の名前にカスタマイズされています。たとえば、Velero をデプロイする前に、Velero manifests ファイルで Velero のデプロイ名が velero-server に更新された場合は、問題がトリガされます。

velero-plugin-for-vsphere の既存のイメージ解析メカニズムで認識できるのは、velero という固定された名前を持つ Velero 環境のみです。

Velero Plugin for vSphere のアンインストール

次の手順を実行して、Velero Plugin for vSphere をアンインストールします。これらの手順は、記載されている順序で実行します。

1 velero-vsphere CLI を実行して、Velero Plugin for vSphere をアンインストールします。

```
./velero-vsphere uninstall -n velero
```

2 velero という名前の vSphere ポッド が削除されたことを確認します。

```
kubectl get pods -n velero
```

ポッドが「終了しています」と表示されている場合は、ポッドが削除されるまで待機してから続行します。

3 vSphere Client を使用して、手動で作成した velero という名前の vSphere 名前空間 を削除します。

注： 名前空間の削除が完了するまで、次の手順に進まないでください。kubectl を使用して、velero 名前空間が削除されていることを確認できます（ただし、kubectl を使用して velero 名前空間を削除しないでください）。

4 vSphere Client を使用して、スーパーバイザー クラスタ から [Velero vSphere Operator] をアンインストールします。

a [ワークロード管理] が有効になっている vCenter Server クラスタを指定します。

b [構成] - [vSphere サービス] - [概要] の順に選択します。

c [Velero vSphere Operator] を選択します。

d [アンインストール] をクリックします。

このアクションを行うと、スーパーバイザー クラスタ から [Velero vSphere Operator] がアンインストールされます。Operator は、[ワークロード管理] - [サービス] ページで引き続き再インストールできます。サービスを完全に削除するには、[アクション] - [削除] の順に選択します。

Velero Plugin for vSphere を使用した vSphere ポッド のバックアップとリストア

Velero Plugin for vSphere を使用して、vSphere ポッド で実行されているワークロードをバックアップおよびリストアできます。

概要

Velero Plugin for vSphere を使用すると、スーパーバイザー クラスタ 内の vSphere ポッド で実行されているワークロードをバックアップおよびリストアできます。vSphere ポッド で実行されているステートレス アプリケーションとステートフル アプリケーションの両方をバックアップおよびリストアできます。ステートフル アプリケーションの場合は、Velero Plugin for vSphere を使用して、パーシステント ボリューム (PV) のスナップショットを作成します。

注： スタンドアローンの Velero と Restic を使用して vSphere ポッド をバックアップおよびリストアすることはできません。スーパーバイザー クラスタ にインストールされている Velero Plugin for vSphere を使用する必要があります。

前提条件

vSphere ポッド をバックアップおよびリストアする前に、Velero Plugin for vSphere をインストールして構成する必要があります。スーパーバイザー クラスタ での Velero Plugin for vSphere のインストールと構成を参照してください。

注： Velero Plugin for vSphere では、スーパーバイザー クラスタ の状態のバックアップとリストアは行われません。

vSphere ポッド のバックアップ

ステートレス vSphere ポッド をバックアップするには、次のコマンドを実行します。

```
velero backup create <backup name> --include-namespaces=my-namespace
```

バックアップは、すべてのローカル スナップショットが作成され、Kubernetes メタデータがオブジェクト ストアにアップロードされた後に `Completed` としてマークされます。ただし、ボリューム スナップショットのバックアップは非同期で実行されるため、バックグラウンドでまだ実行されている可能性があり、完了に時間がかかることがあります。

ボリューム スナップショットのステータスは、スナップショットおよびアップロード カスタム リソースを監視することにより確認できます。

[スナップショット CRD]

スナップショット カスタム リソースは、スナップショットが作成された PVC と同じ名前空間に、ボリューム スナップショットごとに作成されます。PVC 名前空間のすべてのスナップショットを取得するには、次のコマンドを実行します。

```
kubectl get -n <pvc namespace> snapshot
```

スナップショット CRD には、`status.phase` フィールドのフェーズがいくつかあります。次に例を示します。

都道府県	説明
新規	まだ処理されていません
Snapshotted	ローカル スナップショットが作成されました
SnapshotFailed	ローカル スナップショットの作成に失敗しました
アップロードしています	スナップショットをアップロードしています
Uploaded	スナップショットがアップロードされました
UploadFailed	スナップショットのアップロードに失敗しました
Canceling	スナップショットのアップロードをキャンセルしています
キャンセルされました	スナップショットのアップロードがキャンセルされました
CleanupAfterUploadFailed	スナップショットのアップロード後のローカル スナップショットのクリーンアップに失敗しました

[アップロード CRD]

アップロード CR は、オブジェクト ストアにアップロードされるボリューム スナップショットごとに、Velero と同じ名前空間に作成されます。Velero 名前空間のすべてのアップロードを取得するには、次のコマンドを実行します。

```
kubectl get -n <velero namespace> upload
```

アップロード CRD には、`status.phase` フィールドのフェーズがいくつかあります。次に例を示します。

都道府県	説明
新規	まだ処理されていません
InProgress	アップロードが進行中です
UploadError	アップロードに失敗しました
CleanupFailed	アップロード後のローカル スナップショットの削除に失敗しました 再試行されます

都道府県	説明
Canceling	アップロードをキャンセルしています スナップショットのアップロードの進行中に `velero backup delete` が呼び出されると発生する場合があります
キャンセルされました	アップロードがキャンセルされました

[UploadError] では、アップロードが定期的に再試行されます。その時点でフェーズは [InProgress] に戻ります。アップロードが正常に完了すると、そのレコードは、一定期間保持され、最終的には削除されます。

vSphere ポッド のリストア

Velero Plugin for vSphere を使用してバックアップされた vSphere ポッド ワークロードをリストアするには、次の手順を実行します。

1 リストアするワークロードの vSphere 名前空間 を作成します。

2 名前空間のストレージ ポリシーを構成します。

3 ワークロードをリストアするために、次の Velero コマンドを実行します。

```
velero restore create --from-backup backup-name
```

Velero リストアが `Completed` としてマークされるのは、ボリューム スナップショットと他の Kubernetes メタデータが現在のクラスタに正常にリストアされたときです。この時点で、このリストアに関連する vSphere プラグインのすべてのタスクも完了します。Velero バックアップの場合とは異なり、バックグラウンドの非同期のデータ移動タスクはありません。

Velero リストアが `Completed` になる前は、次のように CloneFromSnapshot/Download CR を監視することにより、ボリューム リストアのステータスを確認できます。

[CloneFromSnapshots CRD]

各ボリューム スナップショットからリストアする場合、CloneFromSnapshot CR は、最初にスナップショットが作成された PVC と同じ名前空間に作成されます。PVC 名前空間のすべての CloneFromSnapshot を取得するには、次のコマンドを実行します。

```
kubectl -n <pvc namespace> get clonefromsnapshot
```

CloneFromSnapshot CRD には、`status.phase` フィールドのフェーズがいくつかあります。次に例を示します。

都道府県	説明
新規	スナップショットからのクローン作成が完了していません
完了	スナップショットからのクローン作成が完了しました
Failed	スナップショットからのクローン作成に失敗しました

[ダウンロード CRD]

ダウンロード CR は、オブジェクト ストアからダウンロードされるボリューム スナップショットの各リストアから、Velero と同じ名前空間に作成されます。Velero 名前空間のすべてのダウンロードを取得するには、次のコマンドを実行します。

```
kubectl -n <velero namespace> get download
```

ダウンロード CRD には、`status.phase` フィールドのフェーズがいくつかあります。次に例を示します。

ステータス	説明
新規	まだ処理されていません
InProgress	ダウンロードが進行中です
完了	ダウンロードが完了しました
再試行	ダウンロードを再試行します。 バックアップ データのダウンロード中にエラーが発生した場合は、ダウンロードが再試行されます
Failed	ダウンロードに失敗しました

Tanzu Kubernetes クラスタでの Velero Plugin for vSphere のインストールと構成

Velero Plugin for vSphere を使用して Tanzu Kubernetes クラスタで実行されているワークロードをバックアップおよびリストアするには、このクラスタに Velero Plugin for vSphere をインストールします。

概要

Velero Plugin for vSphere は、Tanzu Kubernetes Grid サービス によってプロビジョニングされたクラスタの Tanzu Kubernetes クラスタ ワークロードをバックアップおよびリストアするためのソリューションを提供します。パーシステント ワークロードの場合は、Velero Plugin for vSphere でパーシステント ボリュームのスナップショットを作成できます。

注： バックアップおよびリストアする Tanzu Kubernetes クラスタ ワークロードに可搬性が必要な場合は、Velero Plugin for vSphere を使用しないでください。Kubernetes クラスタ間で可搬性を確保するには、スタンドアローン Velero と Restic を併用します。Tanzu Kubernetes クラスタへのスタンドアローン Velero と Restic のインストールと構成を参照してください。

前提条件：スーパーバイザー クラスタ への Velero Plugin for vSphere のインストール

Tanzu Kubernetes クラスタに Velero Plugin for vSphere をインストールするには、スーパーバイザー クラスタ に Velero Plugin for vSphere がインストールされている必要があります。また、スーパーバイザー クラスタに NSX-T ネットワークが構成されている必要があります。

Tanzu Kubernetes クラスタに Velero Plugin for vSphere をインストールする前に、まず スーパーバイザー クラスタ に Velero Plugin for vSphere をインストールする必要があります。スーパーバイザー クラスタ での Velero Plugin for vSphere のインストールと構成を参照してください。

Linux Workstation への Velero CLI のインストール

Velero CLI は、Velero とのインターフェイスを提供する標準ツールです。Velero CLI は Velero Plugin for vSphere CLI (`velero-vsphere`) よりも高機能で、Tanzu Kubernetes クラスタ ワークロードのバックアップとリストアに必要です。

Linux ワークステーションに Velero CLI をインストールします。理想的なのは、このワークステーションが、vSphere with Tanzu 環境に関連付けられた CLI（`kubectl`、`kubectl-vsphere`、`velero-vsphere` など）を実行するジャンプ ホストと同じであることです。

次の手順を実行して、Velero CLI をインストールします。

1 VMware 製品ダウンロード ページからサポート対象バージョンの Velero CLI をダウンロードします。サポート対象の Velero バージョンの詳細については、リリース ノートを参照してください。

2 コマンド ラインを開き、Velero CLI をダウンロードしたディレクトリに移動します。

```
gunzip velero-linux-v1.x.x_vmware.1.gz
```

3 Velero バイナリを確認します。

```
ls -l

-rw-r--r-- 1 root root 7142128 Aug 14 14:14 velero-linux-v1.x.x_vmware.1
```

4 Velero CLI に実行権限を付与します。

```
chmod +x velero-linux-v1.x.x_vmware.1
```

5 Velero CLI をシステム パスに移動して、グローバルに使用可能にします。

```
cp velero-linux-v1.x.x_vmware.1 /usr/local/bin/velero
```

6 Velero CLI がインストールされていることを確認します。

```
velero version

Client:
    Version: v1.x.x
```

S3 互換バケットの詳細の取得

便宜上、この手順では、スーパーバイザー クラスタ に Velero Plugin for vSphere をインストールするときに構成したのと同じ S3 互換オブジェクト ストアを使用していると仮定します。本番環境では、別のオブジェクト ストアを作成できます。

Velero Plugin for vSphere をインストールするには、S3 互換オブジェクト ストアに関する次の情報を指定する必要があります。

データ項目	値の例
s3Url	http://my-s3-store.example.com
aws_access_key_id	ACCESS-KEY-ID-STRING
aws_secret_access_key	SECRET-ACCESS-KEY-STRING

次の情報を使用して、`s3-credentials` という名前のシークレット ファイルを作成します。このファイルは、Velero Plugin for vSphere をインストールするときに参照します。

```
aws_access_key_id = ACCESS-KEY-ID-STRING
aws_secret_access_key = SECRET-ACCESS-KEY-STRING
```

Tanzu Kubernetes クラスタでの Velero Plugin for vSphere のインストール

Velero CLI を使用して、バックアップおよびリストアするターゲット Tanzu Kubernetes クラスタに Velero Plugin for vSphere をインストールします。

Velero CLI コンテキストは、`kubectl` コンテキストに自動的に従います。Velero CLI コマンドを実行してターゲット クラスタに Velero と Velero Plugin for vSphere をインストールする前に、`kubectl` コンテキストをターゲット クラスタに設定してください。

1 kubectl 向けの vSphere プラグイン を使用して、スーパーバイザー クラスタ での認証を行います。vCenter Single Sign-On ユーザーとして スーパーバイザー クラスタ に接続するを参照してください。

2 `kubectl` コンテキストをターゲット Tanzu Kubernetes クラスタに設定します。

```
kubectl config use-context TARGET-TANZU-KUBERNETES-CLUSTER
```

3 次の Velero CLI コマンドを実行して、ターゲット クラスタに Velero をインストールします。

BUCKET-NAME、**REGION**（2 つのインスタンス）、および **s3Url** フィールドのプレースホルダ値を適切な値に置き換えます。前述のいずれかの手順に従わなかった場合は、シークレット ファイルの名前や場所、手動で作成した `velero` 名前空間の名前などの値を調整します。

```
./velero install --provider aws \
--bucket BUCKET-NAME \
--secret-file ./s3-credentials \
--features=EnableVSphereItemActionPlugin \
--plugins velero/velero-plugin-for-aws:v1.1.0 \
--snapshot-location-config region=REGION \
--backup-location-config region=REGION,s3ForcePathStyle="true",s3Url=http://my-s3-
store.example.com
```

4 ターゲット クラスタに Velero Plugin for vSphere をインストールします。インストールされている Velero は Kubernetes API サーバと通信して、プラグインをインストールします。

```
velero plugin add vsphereveleroplugin/velero-plugin-for-vsphere:1.1.0
```

クラスタからの Velero Plugin for vSphere のアンインストール

次の手順を実行して、Velero Plugin for vSphere をアンインストールします。

1 `kubectl` コンテキストをターゲット Tanzu Kubernetes クラスタに設定します。

```
kubectl config use-context TARGET-TANZU-KUBERNETES-CLUSTER
```

2 プラグインをアンインストールするには、次のコマンドを実行して、Velero のデプロイから velero-plugin-for-vsphere の InitContainer を削除します。

```
velero plugin remove vsphereveleroplugin/velero-plugin-for-vsphere:1.1.0
```

3 アンインストールを完了するには、バックアップ ドライバのデプロイと関連する CRD を削除します。

```
kubectl -n velero delete deployment.apps/backup-driver
```

```
kubectl delete crds \
backuprepositories.backupdriver.cnsdp.vmware.com \
backuprepositoryclaims.backupdriver.cnsdp.vmware.com \
clonefromsnapshots.backupdriver.cnsdp.vmware.com \
deletesnapshots.backupdriver.cnsdp.vmware.com \
snapshots.backupdriver.cnsdp.vmware.com
```

```
kubectl delete crds uploads.datamover.cnsdp.vmware.com downloads.datamover.cnsdp.vmware.com
```

Velero Plugin for vSphere を使用した Tanzu Kubernetes クラスタ ワークロードのバックアップとリストア

Tanzu Kubernetes クラスタ ワークロードをバックアップおよびリストアするには、Velero Plugin for vSphere を使用します。ただし、移植性が必要な場合は、スタンドアローンの Velero を使用してください。

前提条件

Velero Plugin for vSphere を使用して Tanzu Kubernetes クラスタ ワークロードをバックアップおよびリストアするには、最初に Velero と Velero Plugin for vSphere をターゲット クラスタにインストールする必要があります。Tanzu Kubernetes クラスタでの Velero Plugin for vSphere のインストールと構成を参照してください。

ワークロードのバックアップ

Velero バックアップを作成するコマンドの例を次に示します。

```
velero backup create <backup name> --include-namespaces=my-namespace
```

Velero バックアップは、すべてのローカル スナップショットが作成され、ボリューム スナップショットを除く Kubernetes メタデータがオブジェクト ストアにアップロードされた後に、`Completed` としてマークされます。この時点でも、非同期のデータ移動タスク、つまりボリューム スナップショットのアップロードはバックグラウンドで実行されており、完了に時間がかかることがあります。ボリューム スナップショットのステータスは、スナップショット カスタム リソース (CR) を監視することにより確認できます。

スナップショット

スナップショットは、パーシステント ボリュームをバックアップするために使用されます。スナップショット CR は、スナップショットが作成されたパーシステント ボリューム要求 (PVC) と同じ名前空間に、ボリューム スナップショットごとに作成されます。

PVC 名前空間のすべてのスナップショットを取得するには、次のコマンドを実行します。

```
kubectl get -n <pvc namespace> snapshot
```

スナップショットのカスタム リソース定義 (CRD) には、`.status.phase` フィールドのフェーズが多数あります。次に例を示します。

スナップショット フェーズ	説明
新規	まだ処理されていません
Snapshotted	ローカル スナップショットが作成されました
SnapshotFailed	ローカル スナップショットの作成に失敗しました
アップロードしています	スナップショットをアップロードしています
Uploaded	スナップショットがアップロードされました
UploadFailed	スナップショットのアップロードに失敗しました
Canceling	スナップショットのアップロードをキャンセルしています
キャンセルされました	スナップショットのアップロードがキャンセルされました
CleanupAfterUploadFailed	スナップショットのアップロード後のローカル スナップショットのクリーンアップに失敗しました

ワークロードのリストア

Velero リストアのコマンドの例を次に示します。

```
velero restore create --from-backup <velero-backup-name>
```

Velero リストアが `Completed` としてマークされるのは、ボリューム スナップショットと他の Kubernetes メタデータが現在のクラスタに正常にリストアされたときです。この時点で、このリストアに関連する vSphere プラグインのすべてのタスクも完了します。Velero バックアップの場合とは異なり、バックグラウンドの非同期のデータ移動タスクはありません。

CloneFromSnapshot

各ボリューム スナップショットからリストアするために、最初にスナップショットが作成された PVC と同じ名前空間に `CloneFromSnapshot` カスタム リソース (CR) が作成されます。PVC 名前空間のすべての CloneFromSnapshot を取得するには、次のコマンドを実行します。

```
kubectl -n <pvc namespace> get clonefromsnapshot
```

CloneFromSnapshot CRD には、`.status.phase` フィールドの主要なフェーズがいくつかあります。

スナップショット フェーズ	説明
新規	スナップショットからのクローン作成が完了していません
InProgress	vSphere ボリュームのスナップショットがリモート リポジトリからダウンロードされています
完了	スナップショットからのクローン作成が完了しました
Failed	スナップショットからのクローン作成に失敗しました

Tanzu Kubernetes クラスタへのスタンドアローン Velero と Restic のインストールと構成

Tanzu Kubernetes でワークロードをバックアップおよびリストアするには、データストアを作成し、Kubernetes クラスタに Velero と Restic をインストールします。

概要

Tanzu Kubernetes クラスタは仮想マシン ノードで実行されます。Tanzu Kubernetes クラスタをバックアップおよびリストアするには、そのクラスタに Velero と Restic をインストールします。

前提条件

Tanzu Kubernetes クラスタで実行されているワークロードをバックアップおよびリストアするために Velero と Restic をインストールするための次の前提条件を環境が満たしていることを確認します。

- いくつかのワークロード バックアップを保存するために十分なストレージを持つ Linux 仮想マシン。この仮想マシンに MinIO をインストールします。
- kubectl 向けの vSphere プラグイン と kubectl を含む vSphere 向け Kubernetes CLI Tools がインストールされる Linux 仮想マシン。このクライアント仮想マシンに Velero CLI をインストールします。このような仮想マシンがない場合は Velero CLI をローカルにインストールできますが、条件に合わせてインストール手順を調整する必要があります。
- Kubernetes 環境がインターネットにアクセスでき、クライアント仮想マシンからアクセスできること。

MinIO オブジェクト ストアのインストールと構成

Velero は、Kubernetes ワークロードのバックアップ先として S3 互換のオブジェクト ストアを必要とします。Velero は、そのようなオブジェクト ストア プロバイダをいくつかサポートしています。簡単にするために、これらの手順では、オブジェクト ストア仮想マシンでローカルに実行される S3 互換のストレージ サービスである MinIO を使用します。

1 MinIO をインストールします。

```
wget https://dl.min.io/server/minio/release/linux-amd64/minio
```

2 MinIO に実行権限を付与します。

```
chmod +x minio
```

3 ファイル システムに MinIO 用のディレクトリを作成します。

```
mkdir /DATA-MINIO
```

4 MinIO サーバを起動します。

```
./minio server /DATA-MINIO
```

5 MinIO サーバが起動すると、エンドポイント URL、AccessKey、SecretKey などの重要なデータストア インスタンスの詳細が得られます。表に示されているエンドポイント URL、AccessKey、SecretKey を記録します。

データストア メタデータ	値
エンドポイント URL	
AccessKey	
SecretKey	

6 ブラウザで MinIO サーバのエンドポイント URL を指定して、MinIO データストアを参照します。

7 MinIO サーバにログインし、AccessKey と SecretKey を指定します。

8 MinIO をサービスとして有効にするには、`minio.service` スクリプトをダウンロードして、MinIO に自動起動を構成します。

```
curl -O https://raw.githubusercontent.com/minio/minio-service/master/linux-systemd/
minio.service
```

9 `minio.service` スクリプトを編集し、`ExecStart` に次の値を追加します。

```
ExecStart=/usr/local/bin/minio server /DATA-MINIO path
```

10 変更したスクリプトを保存します。

11 次のコマンドを実行して MinIO サービスを構成します。

```
cp minio.service /etc/systemd/system
cp minio /usr/local/bin/
systemctl daemon-reload
systemctl start minio
systemctl status minio
systemctl enable minio
```

12 MinIO ブラウザを起動し、オブジェクト ストアにログインして、バックアップとリストア用の MinIO バケットを作成します。

13 バケット作成のアイコンをクリックします。

14 たとえば `my-cluster-backups` のように、バケット名を入力します。

15 バケットが作成されたことを確認します。

16 デフォルトでは、新しい MinIO バケットは読み取り専用です。Velero スタンドアローンのバックアップとリストアの場合、MinIO バケットは読み取り/書き込みである必要があります。バケットを読み取り/書き込みに設定するには、バケットを選択し、省略記号（点が並んだ記号）のリンクをクリックします。

17 [ポリシーの編集] を選択します。

18 ポリシーを [読み取りおよび書き込み] に変更します。

19 [追加] をクリックします。

20 ダイアログ ボックスを閉じるために、X をクリックします。

Velero CLI のインストール

仮想マシン クライアントまたはローカル マシンに Velero CLI をインストールします。

1 VMware 製品のダウンロード ページから、vSphere with Tanzu 用署名付き Velero バイナリのサポート対象バージョンをダウンロードします。

注： VMware からサポートを受けるには、VMware によって署名された Velero バイナリを使用する必要があります。

2 コマンド ラインを開き、Velero CLI をダウンロードしたディレクトリに移動します。

3 ダウンロード ファイルを解凍します。例：

```
gunzip velero-linux-vX.X.X_vmware.1.gz
```

4 Velero バイナリを確認します。

```
ls -l
```

5 Velero CLI に実行権限を付与します。

```
chmod +x velero-linux-vX.X.X_vmware.1
```

6 Velero CLI をシステム パスに移動して、グローバルに使用可能にします。

```
cp velero-linux-vX.X.X_vmware.1 /usr/local/bin/velero
```

7 インストールを確認します。

```
velero version
```

Tanzu Kubernetes クラスタへの Velero と Restic のインストール

Velero CLI のコンテキストは、kubectl のコンテキストに自動的に従います。Velero CLI コマンドを実行して Velero と Restic をターゲット クラスタにインストールする前に、kubectl のコンテキストを設定します。

1 MinIO バケットの名前を取得します。たとえば、`my-cluster-backups` です。

2 MinIO バケットの AccessKey と SecretKey を取得します。

3 Velero CLI の動作対象となるクラスタが認識されるように、コンテキストをターゲットの Kubernetes クラスタに設定します。

```
kubectl config use-context tkgs-cluster-name
```

4 `credentials-minio` という名前のシークレット ファイルを作成します。収集した MinIO サーバ アクセス認証情報を使用して、ファイルを更新します。例：

```
aws_access_key_id = 0XXNO8JCCGV41QZBV0RQ
aws_secret_access_key = clZ1bf8Ljkvkmq7fHucrKCkxV39BRbcycGeXQDfx
```

注： 「バックアップ ストアの取得中にエラーが発生しました」というエラー メッセージと「NoCredentialProviders: チェーン内に有効なプロバイダがありません」という説明が表示された場合は、認証情報ファイルの先頭に `[default]` 行を追加します。例：

```
[default]
aws_access_key_id = 0XXNO8JCCGV41QZBV0RQ
aws_secret_access_key = clZ1bf8Ljkvkmq7fHucrKCkxV39BRbcycGeXQDfx
```

5 ファイルを保存し、ファイルが正しい場所に置かれたことを確認します。

```
ls
```

6 次のコマンドを実行して、ターゲット Kubernetes クラスタに Velero と Restic をインストールします。両方の URL を MinIO インスタンスの URL に置き換えます。

```
velero install \
--provider aws \
--plugins velero/velero-plugin-for-aws:v1.0.0 \
--bucket tkgs-velero \
--secret-file ./credentials-minio \
--use-volume-snapshots=false \
```

```
--use-restic \
--backup-location-config \
region=minio,s3ForcePathStyle="true",s3Url=http://10.199.17.63:9000,publicUrl=http://
10.199.17.63:9000
```

7 Velero と Restic のインストールを確認します。

```
kubectl logs deployment/velero -n velero
```

8 `velero` 名前空間を確認します。

```
kubectl get ns
```

9 `velero` ポッドと `restic` ポッドを確認します。

```
kubectl get all -n velero
```

Restic DaemonSet のトラブルシューティング（必要な場合）

Kubernetes クラスタで 3 ポッドの Restic DaemonSet を実行するには、Restic DaemonSet の仕様を更新して hostPath を変更することが必要な場合があります。この問題の詳細については、Velero のドキュメントで Restic Integration を参照してください。

1 3 ポッドの Restic DaemonSet を確認します。

```
kubectl get pod -n velero
```

ポッドのステータスが CrashLoopBackOff の場合は、次のように編集します。

2 `edit` コマンドを実行します。

```
kubectl edit daemonset restic -n velero
```

3 hostPath を `/var/lib/kubelet/pods` から `/var/vcap/data/kubelet/pods` に変更します。

```
- hostPath:
    path: /var/vcap/data/kubelet/pods
```

4 ファイルを保存します。

5 3 ポッドの Restic DaemonSet を確認します。

```
kubectl get pod -n velero
```

```
NAME                      READY   STATUS    RESTARTS   AGE
restic-5jln8              1/1     Running   0          73s
restic-bpvtq              1/1     Running   0          73s
restic-vg8j7              1/1     Running   0          73s
velero-72c84322d9-le7bd   1/1     Running   0          10m
```

Velero のメモリ制限の調整（必要な場合）

Velero のバックアップが数時間に渡って `status=InProgress` を返す場合は、メモリ設定の limits と requests の値を増やします。

1 次のコマンドを実行します。

```
kubectl edit deployment/velero -n velero
```

2 メモリ設定の limits と requests をデフォルトの `256Mi` および `128Mi` から `512Mi` および `256Mi` に変更します。

```
ports:
- containerPort: 8085
  name: metrics
  protocol: TCP
resources:
  limits:
    cpu: "1"
    memory: 512Mi
  requests:
    cpu: 500m
    memory: 256Mi
terminationMessagePath: /dev/termination-log
terminationMessagePolicy: File
```

スタンドアローンの Velero と Restic を使用した Tanzu Kubernetes クラスタ ワークロードのバックアップとリストア

スタンドアローンの Velero と Restic を使用して、Tanzu Kubernetes クラスタ ワークロードをバックアップおよびリストアできます。これは Velero Plugin for vSphere を使用する代わりの方法です。Velero Plugin for vSphere の代わりにスタンドアローンの Velero を使用する一番の理由は、移植性が必要であるということです。ステートフル ワークロードには、Restic が必要です。

前提条件

スタンドアローンの Velero と Restic を使用して Tanzu Kubernetes クラスタ ワークロードをバックアップおよびリストアするには、ターゲット クラスタにスタンドアローン バージョンの Velero と Restic をインストールする必要があります。Tanzu Kubernetes クラスタへのスタンドアローン Velero と Restic のインストールと構成を参照してください。

注： Kubernetes クラスタをバックアップおよびリストアするときに、スタンドアローンの Velero と Restic を使用すると、移植性が得られます。つまり、Tanzu Kubernetes Grid サービス でプロビジョニングされていない Kubernetes クラスタにクラスタ ワークロードをリストアできるようにする場合は、スタンドアローンの Velero を使用する必要があります。

Tanzu Kubernetes クラスタで実行されているステートレス アプリケーションのバックアップ

Tanzu Kubernetes クラスタで実行されているステートレス アプリケーションをバックアップするには、Velero を使用する必要があります。

この例では、サンプル ステートレス アプリケーションの名前空間にすべてのアプリケーション コンポーネントがある場合に、`--include namespaces` タグを使用して、そのサンプル ステートレス アプリケーションをバックアップおよびリストアする方法を示します。

```
velero backup create example-backup --include-namespaces example-backup
```

次の内容が表示されます。

```
Backup request "example-backup" submitted successfully.
Run `velero backup describe example-backup` or `velero backup logs example-backup` for more
details.
```

作成されたバックアップを確認します。

```
velero backup get
```

```
velero backup describe example-backup
```

MinIO サーバなどの S3 互換オブジェクト ストアの Velero バケットを確認します。

Velero は Kubernetes カスタム リソース定義 (CRD) にメタデータを書き込みます。

```
kubectl get crd
```

Velero CRD では、次のような特定のコマンドを実行できます。

```
kubectl get backups.velero.io -n velero
```

```
kubectl describe backups.velero.io guestbook-backup -n velero
```

Tanzu Kubernetes クラスタで実行されているステートレス アプリケーションのリストア

Tanzu Kubernetes クラスタで実行されているステートレス アプリケーションをリストアするには、Velero を使用する必要があります。

サンプル アプリケーションのリストアをテストするために、サンプル アプリケーションを削除します。

名前空間を削除します。

```
kubectl delete ns guestbook
namespace "guestbook" deleted
```

アプリケーションをリストアします。

```
velero restore create --from-backup example-backup
```

次の内容が表示されます。

```
Restore request "example-backup-20200721145620" submitted successfully.
Run `velero restore describe example-backup-20200721145620` or `velero restore logs example-
backup-20200721145620` for more details.
```

アプリケーションがリストアされたことを確認します。

```
velero restore describe example-backup-20200721145620
```

検証のために、次のコマンドを実行します。

```
velero restore get
```

```
kubectl get ns
```

```
kubectl get pod -n example
```

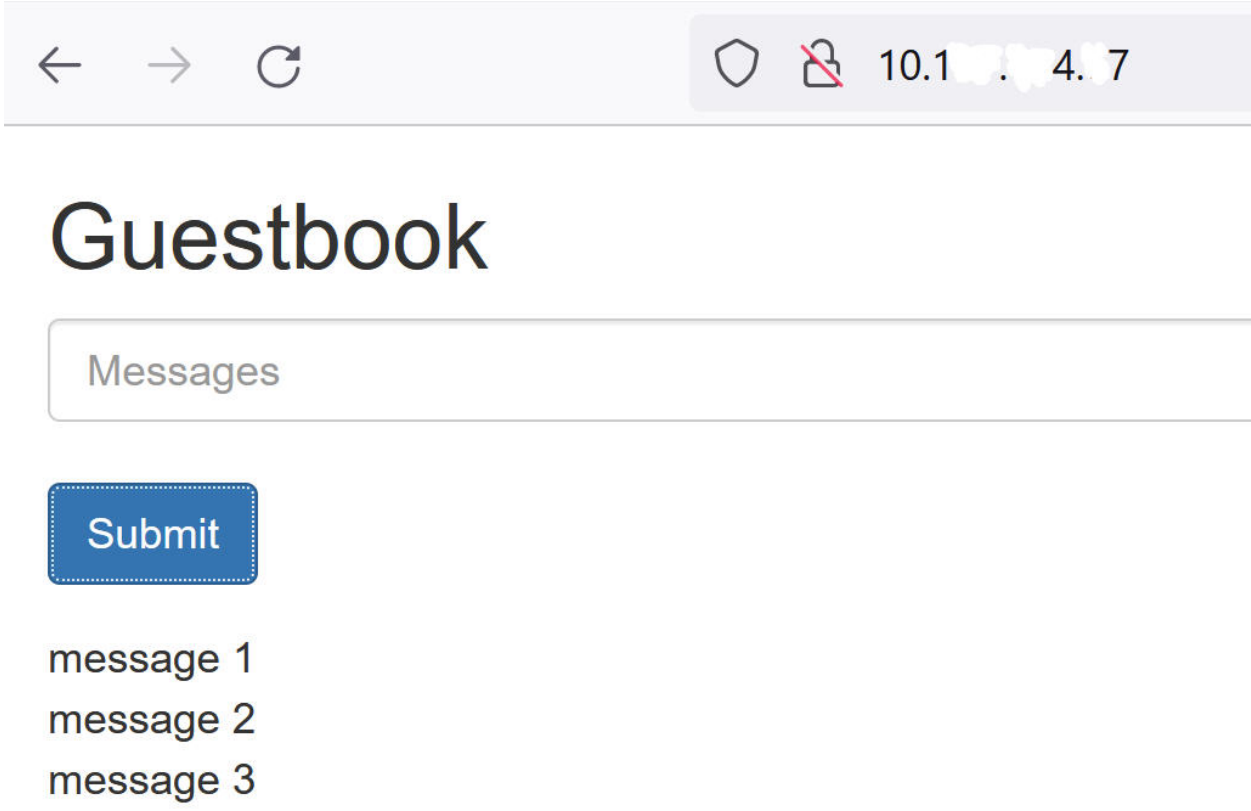
```
kubectl get svc -n example
```

Tanzu Kubernetes クラスタで実行されているステートフル アプリケーションのバックアップ

Tanzu Kubernetes クラスタで実行されているステートフル アプリケーションをバックアップする場合は、保存されているアプリケーション メタデータとアプリケーション データの両方をパーシステント ボリュームにバックアップする必要があります。これを行うには、Velero と Restic の両方が必要です。

この例では、Guestbook アプリケーションを使用します。Guestbook アプリケーションは、Tanzu Kubernetes クラスタにデプロイされているものと想定します。詳細については、Tanzu Kubernetes Guestbook のチュートリアルを参照してください。

ステートフルなバックアップとリストアを説明するために、フロントエンド Web ページを使用してメッセージを Guestbook アプリケーションに送信し、そのメッセージが保存されるようにしてみましょう。例：



この例では、`--include namespace` タグやポッドの注釈を使用して、Guestbook アプリケーションをバックアップおよびリストアする方法を示します。

注： この例では、注釈を使用します。ただし、Velero バージョン 1.5 以降では、注釈は不要になりました。注釈を使用しないようにするには、バックアップの作成時に `--default-volumes-to-restic` オプションを使用してください。これにより、すべての PV が Restic を使用して自動的にバックアップされます。詳細については https://velero.io/docs/v1.5/restic/を参照してください。

バックアップ手順を開始するために、ポッドの名前を取得します。

```
kubectl get pod -n guestbook
```

例：

```
kubectl get pod -n guestbook

NAME                                        READY   STATUS    RESTARTS   AGE
guestbook-frontend-deployment-85595f5bf9-h8cff  1/1     Running             0          55m
guestbook-frontend-deployment-85595f5bf9-lw6tg  1/1     Running             0          55m
guestbook-frontend-deployment-85595f5bf9-wpqc8  1/1     Running             0          55m
redis-leader-deployment-64fb8775bf-kbs6s        1/1     Running             0          55m
redis-follower-deployment-84cd76b975-jrn8v      1/1     Running             0          55m
redis-follower-deployment-69df9b5688-zml4f      1/1     Running             0          55m
```

パーシステント ボリュームが Redis ポッドに接続されます。これらのステートフル ポッドは、Restic を使用してバックアップします。このため、volumeMount の名前を使用して、ステートフル ポッドに注釈を追加する必要があります。

ステートフル ポッドに注釈を付けるには、volumeMount を知る必要があります。mountName を取得するには、次のコマンドを実行します。

```
kubectl describe pod redis-leader-deployment-64fb8775bf-kbs6s -n guestbook
```

結果には、redis-leader-data からの Containers.leader.Mounts: /data が表示されます。この最後のトークンが、リーダー ポッドの注釈に使用する volumeMount 名です。フォロワーの場合は、redis-follower-data になります。volumeMount 名は、ソース YAML から取得することもできます。

各 Redis ポッドに注釈を付けます。次に例を示します。

```
kubectl -n guestbook annotate pod redis-leader-64fb8775bf-kbs6s backup.velero.io/backup-
volumes=redis-leader-data
```

次の内容が表示されます。

```
pod/redis-leader-64fb8775bf-kbs6s annotated
```

注釈を確認します。

```
kubectl -n guestbook describe pod redis-leader-64fb8775bf-kbs6s | grep Annotations
Annotations:  backup.velero.io/backup-volumes: redis-leader-data
```

```
kubectl -n guestbook describe pod redis-follower-779b6d8f79-5dphr | grep Annotations
Annotations:  backup.velero.io/backup-volumes: redis-follower-data
```

Velero バックアップを実行します。

```
velero backup create guestbook-backup --include-namespaces guestbook
```

次の内容が表示されます。

```
Backup request "guestbook-backup" submitted successfully.
Run `velero backup describe guestbook-pv-backup` or `velero backup logs guestbook-pv-backup`
for more details.
```

作成されたバックアップを確認します。

```
velero backup get

NAME                   STATUS      ERRORS   WARNINGS   CREATED
EXPIRES   STORAGE LOCATION   SELECTOR
guestbook-backup       Completed   0        0          2020-07-23 16:13:46 -0700 PDT
29d       default            <none>
```

バックアップの詳細を確認します。

```
velero backup describe guestbook-backup --details
```

Velero では、他にも次のようなコマンドを実行できます。

```
kubectl get backups.velero.io -n velero

NAME               AGE
guestbook-backup   4m58s
```

次のコマンドも実行できます。

```
kubectl describe backups.velero.io guestbook-backup -n velero
```

Tanzu Kubernetes クラスタで実行されているステートフル アプリケーションのリストア

Tanzu Kubernetes クラスタで実行されるステートフル アプリケーションをリストアする場合は、保存されているアプリケーション メタデータとアプリケーション データの両方をパーシステント ボリュームにリストアする必要があります。これを行うには、Velero と Restic の両方が必要です。

この例では、前のセクションの説明に従ってステートフル Guestbook アプリケーションをバックアップしたものと想定しています。

ステートフル アプリケーションのリストアをテストするために、その名前空間を削除します。

```
kubectl delete ns guestbook
namespace "guestbook" deleted
```

アプリケーションが削除されたことを確認します。

```
kubectl get ns
kubectl get pvc,pv --all-namespaces
```

アプリケーションをリストアします。

```
Restore request "guestbook-backup-20200723161841" submitted successfully.
Run `velero restore describe guestbook-backup-20200723161841` or `velero restore logs
guestbook-backup-20200723161841` for more details.
```

ステートフル Guestbook アプリケーションがリストアされたことを確認します。

```
velero restore describe guestbook-backup-20200723161841

Name:         guestbook-backup-20200723161841
Namespace:    velero
Labels:       <none>
Annotations:  <none>

Phase:  Completed

Backup:  guestbook-backup

Namespaces:
  Included:  all namespaces found in the backup
  Excluded:  <none>

Resources:
  Included:        *
  Excluded:        nodes, events, events.events.k8s.io, backups.velero.io,
restores.velero.io, resticrepositories.velero.io
  Cluster-scoped:  auto

Namespace mappings:  <none>

Label selector:  <none>

Restore PVs:  auto

Restic Restores (specify --details for more information):
  Completed:  3
```

さらに次のコマンドを実行して、リストアを確認します。

```
velero restore get

NAME                                BACKUP                STATUS      ERRORS   WARNINGS
CREATED                         SELECTOR
guestbook-backup-20200723161841      guestbook-backup      Completed   0        0
2021-08-11 16:18:41 -0700 PDT   <none>
```

名前空間がリストアされたことを確認します。

```
kubectl get ns

NAME              STATUS   AGE
default           Active   16d
guestbook         Active   76s
...
velero            Active   2d2h
```

アプリケーションがリストアされたことを確認します。

```
vkubectl get all -n guestbook

NAME                                 READY   STATUS    RESTARTS   AGE
pod/frontend-6cb7f8bd65-h2pnb        1/1     Running   0          6m27s
pod/frontend-6cb7f8bd65-kwlpr        1/1     Running   0          6m27s
pod/frontend-6cb7f8bd65-snwl4        1/1     Running   0          6m27s
pod/redis-leader-64fb8775bf-kbs6s    1/1     Running   0          6m28s
pod/redis-follower-779b6d8f79-5dphr 1/1     Running   0          6m28s
pod/redis-follower-899c7e2z65-8apnk 1/1     Running   0          6m28s

NAME                                  TYPE           CLUSTER-IP       EXTERNAL-IP
PORT(S)        AGE
service/guestbook-frontend            LoadBalancer   10.10.89.59      10.19.15.99
80:31513/TCP   65s
service/redis-follower                ClusterIP      10.111.163.189   <none>
6379/TCP       65s
service/redis-leader                  ClusterIP      10.111.70.189    <none>
6379/TCP       65s

NAME                                          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/guestbook-frontend-deployment   3/3     3            3           65s
deployment.apps/redis-follower-deployment       1/2     2            1           65s
deployment.apps/redis-leader-deployment         1/1     1            1           65s

NAME                                                DESIRED   CURRENT   READY   AGE
replicaset.apps/guestbook-frontend-deployment-56fc5b6b47   3         3         3       65s
replicaset.apps/redis-follower-deployment-6fc9cf5759       2         2         1       65s
replicaset.apps/redis-leader-deployment-7d89bbdbcf         1         1         1       65s
```

パーシステント ボリュームがリストアされたことを確認します。

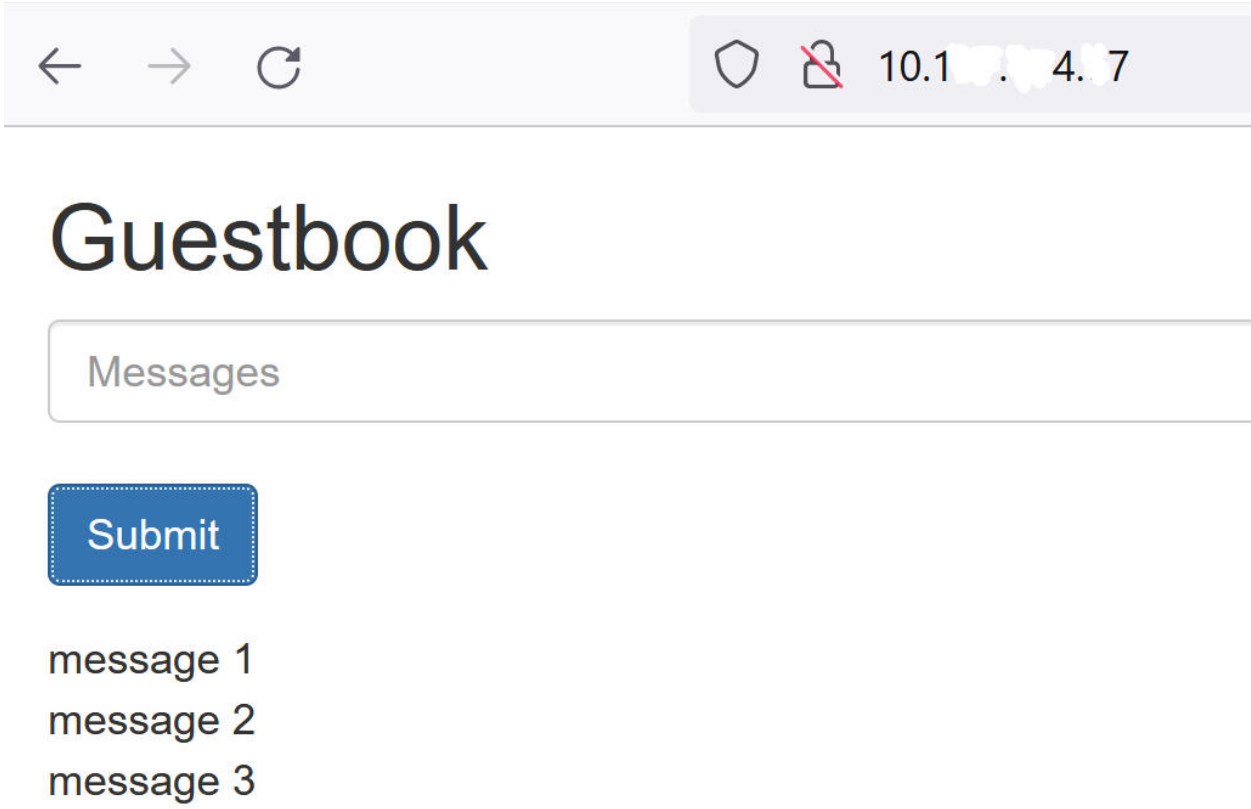
```
kubectl get pvc,pv -n guestbook

NAME                                       STATUS
VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
persistentvolumeclaim/redis-leader-claim   Bound    pvc-a2f6e6d4-42db-4fb8-
a198-5379a2552509   2Gi        RWO            thin-disk      2m40s
persistentvolumeclaim/redis-follower-claim Bound    pvc-55591938-921f-452a-
b418-2cc680c0560b   2Gi        RWO            thin-disk      2m40s

NAME                                                     CAPACITY   ACCESS MODES   RECLAIM
```

```
POLICY   STATUS   CLAIM                                STORAGECLASS   REASON   AGE
persistentvolume/pvc-55591938-921f-452a-b418-2cc680c0560b   2Gi        RWO
Delete           Bound    guestbook/redis-follower-claim    thin-disk               2m40s
persistentvolume/pvc-a2f6e6d4-42db-4fb8-a198-5379a2552509   2Gi        RWO
Delete           Bound    guestbook/redis-leader-claim      thin-disk               2m40s
```

最後に、guestbook-frontend サービスの外部 IP アドレスを使用して Guestbook フロントエンドにアクセスし、チュートリアルの最初に送信したメッセージがリストアされたことを確認します。例：



vCenter Server のバックアップとリストア

このトピックでは、vSphere with Tanzu 環境のコンテキストで vCenter Server をバックアップおよびリストアする方法について説明します。

vCenter Server クラスタ HA

高可用性の スーパーバイザー クラスタ をサポートするために、vSphere with Tanzu では、vSphere with Tanzu が有効な vCenter Server クラスタに高可用性を構成する必要があります。詳細は、VMware vSphere のドキュメントの vSphere の可用性を参照してください。

vCenter Server のバックアップとリカバリ

vCenter Server は、ネットワーク接続ストレージへのファイル バックアップをサポートしています。vCenter Server をバックアップおよびリストアするには、プライマリ vCenter Server のバックアップを作成します。詳細については、vCenter Server のドキュメントの File-Based Backup and Restore of vCenter Server を参照してください。

NSX-T Data Center のバックアップとリストア

vSphere with Tanzu ワークロードが停止した場合にネットワーク機能をサポートするために、NSX-T Data Center のバックアップとリストアを行います。

要件

NSX-T Data Center のバックアップとリストアを行うには、前提として、3 台の NSX Manager ノードがデプロイされていること、および NSX 管理プレーンにアクセスするように HA VIP が構成されていることが必要です。また、2 台以上の Edge ノードが専用の HA VIP とともにデプロイされていることも必要です。詳細については、『vSphere with Tanzu 用 NSX-T Data Center の構成』を参照してください。

NSX-T のバックアップとリストア

NSX-T Data Center が製品内に備えているバックアップとリカバリでは、NSX Manager ノードおよびオブジェクトのバックアップとリストアがサポートされています。詳細については、NSX-T ドキュメントの Backing Up and Restoring NSX Manager を参照してください。

vSphere with Tanzu のトラブルシューティング

19

vSphere with Tanzu のインフラストラクチャでは、以下のトラブルシューティング手法とベスト プラクティスを使用します。

この章には、次のトピックが含まれています。

ストレージのベスト プラクティスとトラブルシューティング

vSphere with Tanzu のストレージ環境で、ベスト プラクティスとトラブルシューティング手法を使用できます。

vSAN 以外のデータストアで制御プレーン仮想マシンの非アフィニティ ルールを使用する

vSphere with Tanzu でクラスタ内の vSAN 以外のデータストアを使用する場合は、可用性を確保するために、3 つの制御プレーン仮想マシンを異なるデータストアに配置します。

制御プレーン仮想マシンはシステムで管理されているため、手動で移行することはできません。データストア クラスタと Storage DRS を組み合わせて使用して制御プレーン仮想マシンを再分散し、それらを別々のデータストアに配置します。

手順

1 vSphere Client で、データストア クラスタを作成します。

   a データセンターに移動します。

   b データセンター オブジェクトを右クリックし、[新規データストア クラスタ] を選択します。

   c データストア クラスタに名前を付け、[Storage DRS をオンにする] が有効であることを確認します。

   d クラスタの自動化レベルを [自動化なし (手動モード)] に設定します。

e Storage DRS ランタイム設定はデフォルトのままにします。

f vSphere with Tanzu で有効になっている ESXi クラスタを選択します。

g データストア クラスタに追加するすべての共有データストアを選択します。

h [終了] をクリックします。

2 制御プレーン仮想マシンの Storage DRS ルールを定義します。

a データストア クラスタに移動します。

b [構成] タブをクリックし、[設定] の [ルール] をクリックします。

c [追加] アイコンをクリックして、ルールの名前を入力します。

d [ルールの有効化] が有効であることを確認します。

e [ルール タイプ] を [仮想マシンの非アフィニティ] に設定します。

f [追加] アイコンをクリックして、3 つのスーパーバイザー制御プレーン仮想マシンを選択します。

g [OK] をクリックして設定を終了します。

3 仮想マシン オーバーライドを作成します。

a データストア クラスタに移動します。

b [構成] タブをクリックし、[設定] の [仮想マシンのオーバーライド] をクリックします。

c [追加] アイコンをクリックして、3 つの制御プレーン仮想マシンを選択します。

d Storage DRS の自動化レベルを有効にするには、[オーバーライド] チェック ボックスを選択し、値を [完全自動化] に設定します。

e [終了] をクリックします。

結果

このタスクにより、制御プレーン仮想マシンの Storage DRS のみが有効になり、仮想マシンが再分散されて異なるデータストアに配置されます。

Storage vMotion が実行されたら、SDRS ルールおよびオーバーライドを削除し、Storage DRS を無効にして、データストア クラスタを削除することができます。

vSphere から削除されたストレージ ポリシーが引き続き Kubernetes ストレージ クラスとして表示される

vSphere Client を使用して vCenter Server または スーパーバイザー クラスタ 内の名前空間からストレージ ポリシーを削除すると、それに一致するストレージ クラスは Kubernetes 環境に残りますが、使用できません。

問題

`kubectl get sc` コマンドを実行すると、出力には名前空間で使用できるストレージ クラスとして引き続き表示されます。ただし、そのストレージ クラスは使用できません。たとえば、新しいパーシステント ボリュームの要求に対してそのストレージ クラスを使用すると失敗します。

ストレージ クラスがすでに Kubernetes 環境で使用されている場合は、その環境で予期しない動作が発生する可能性があります。

解決方法

1 名前空間にあるストレージ クラスを確認するには、
`kubectl describe namespace` ***`namespace_name`*** コマンドを実行します。

一致するストレージ ポリシーが削除されている場合、このコマンドの出力にストレージ クラスは表示されません。

2 ストレージ クラスがすでにデプロイ環境で使用されている場合は、そのストレージ クラスをリストアします。

a vSphere Client を使用して、削除したポリシーと同じ名前の新しいストレージ ポリシーを作成します。

たとえば、*Gold* ポリシーを削除した場合は、新しいポリシーに *Gold* という名前を付けます。vSphere with Tanzu のストレージ ポリシーの作成 を参照してください。

b ポリシーを名前空間に割り当てます。

名前空間のストレージ設定の変更を参照してください。

ポリシーを名前空間に割り当てると、vSphere with Tanzu によって古いストレージ クラスが削除され、一致するストレージ クラスが同じ名前で作成されます。

vSAN Direct と外部ストレージの併用

vSphere with Tanzu 環境で vSAN Direct を使用している場合に、外部共有ストレージを使用して管理内部仮想マシンおよびその他のメタデータを格納できます。

問題

同種の vSAN Direct クラスタをデプロイする場合は、vSphere with Tanzu 管理仮想マシンおよびその他のメタデータを格納するために、クラスタ内の各 ESXi ホストにレプリケートされた vSAN データストアを作成する必要があります。vSAN データストアは容量を使用するほか、各ホストで追加の I/O コントローラが必要となります。また、vSAN Direct をサポートできるハードウェア構成が制限されます。

vSAN データストアを構成する代わりに、外部共有ストレージを使用して管理内部仮想マシンおよびその他のメタデータを格納できます。

解決方法

1 クラスタ内の ESXi ホストに vSAN または vSAN Direct がデプロイされている場合は、構成からホストをクリアします。

a vSAN または vSAN Direct に割り当てられているディスクを削除します。vSAN からのディスク グループまたはデバイスの削除を参照してください。

b （オプション） スクリプトを使用して、ホスト上のディスクに vSAN Direct 用のタグを付けます。スクリプトを使用した vSAN Direct のストレージ デバイスのタグ付けを参照してください。

2 VMware Cloud Foundation を使用して、外部ストレージを含むワークロード ドメインを作成します。

NFS、vVols、FC などのストレージ オプションのいずれかを選択してください。選択できるオプションは 1 つだけです。

詳細については、VMware Cloud Foundation 運用および管理ガイドの「Working with Workload Domains」を参照してください。

この手順により、vCenter Server と指定した ESXi ホストを含むワークロード ドメインがデプロイされます。外部ストレージがすべてのホストにマウントされ、デフォルトのクラスタに追加されます。

3 vSAN を有効にします。

vSAN 用のディスクが要求されないことを確認します。

詳細については、既存のクラスタで vSAN を有効にするを参照してください。

この手順により、vSAN ネットワークを使用する 0 バイトの vSAN データストアが作成されます。ローカル ディスクは vSAN に使用されません。

4 ホスト上のローカル ディスクを vSAN Direct 用に要求します。

詳細については、『vSphere with Tanzu 用の vSAN Direct の設定』を参照してください。

要求されるデバイスごとに、vSAN Direct は個別のデータストアを作成します。

5 vSAN Direct のストレージ ポリシーを作成します。

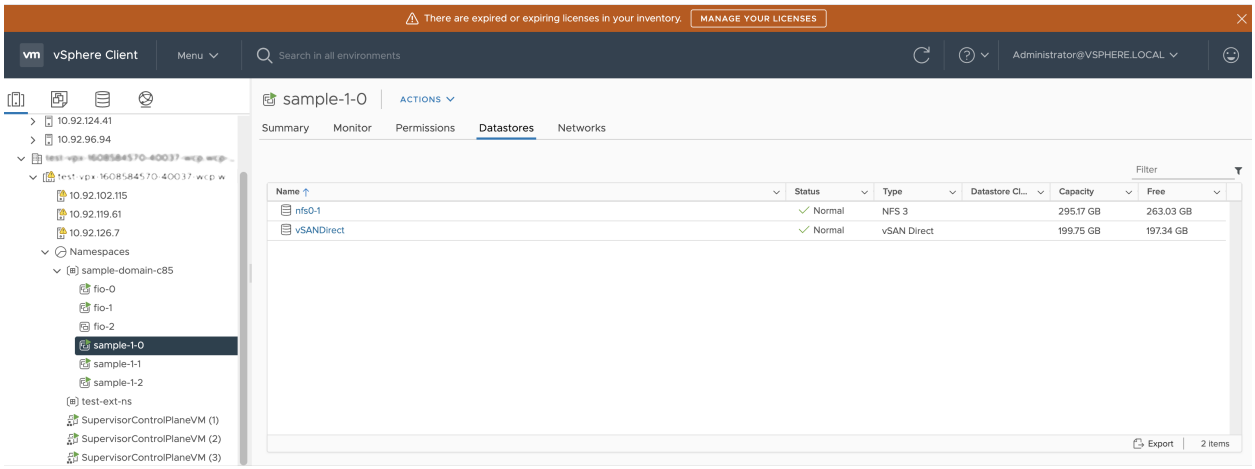
詳細については、『vSAN Direct ストレージ ポリシーの作成』を参照してください。

6 ワークロード管理を構成して有効にします。

詳細については、『5 章 スーパーバイザー クラスタ の構成と管理』を参照してください。

例

この例では、セットアップに外部 NFS ストレージと vSAN Direct データストアが含まれています。制御プレーン仮想マシンと vSphere ポッド は外部 NFS ストレージで実行されています。パーシステント ボリュームの要求は vSAN Direct で実行されます。



ネットワークのトラブルシューティング

NSX で vSphere with Tanzu を設定するときに発生する可能性があるネットワークの問題のトラブルシューティングを行うことができます。

NSX Manager による vCenter Server の登録

状況に応じて、vCenter Server OIDC を NSX Manager に再登録する必要があります。たとえば、vCenter Server の FQDN/PNID が変更された場合などです。

手順

1 SSH [Secure SHell] を介して vCenter Server アプライアンスに接続します。

2 `shell` コマンドを実行します。

3 vCenter Server サムプリントを取得するには、`- openssl s_client -connect <vcenterserver-FQDN:443 </dev/null 2>/dev/null | openssl x509 -fingerprint -sha256 -noout -in /dev/stdin` コマンドを実行します。

サムプリントが表示されます。たとえば、`08:77:43:29:E4:D1:6F:29:96:78:5F:BF:D6:45:21:F4:0E:3B:2A:68:05:99:C3:A4:89:8F:F2:0B:EA:3A:BE:9D`。

4 SHA256 サムプリントをコピーし、コロンを削除します。

`08774329E4D16F2996785FBFD64521F40E3B2A680599C3A4898FF20BEA3ABE9D`

5 vCenter Server の OIDC を更新するには、次のコマンドを実行します。

```
curl --location --request POST 'https://<NSX-T_ADDRESS>/api/v1/trust-management/oidc-uris'
\
    --header 'Content-Type: application/json' \
    --header 'Authorization: Basic <AUTH_CODE>' \
    --data-raw '{
 "oidc_type": "vcenter",
     "oidc_uri": "https://<VC_ADDRESS>/openidconnect/vsphere.local/.well-known/openid-
configuration",
     "thumbprint": "<VC_THUMBPRINT>"
    }'
```

NSX アプライアンスのパスワードを変更できない

`root`、`admin`、または `audit` ユーザーの NSX アプライアンス パスワードを変更できない場合があります。

問題

`root`、`admin`、または `audit` ユーザーの NSX アプライアンス パスワードを、vSphere Client を介して変更すると失敗することがあります。

原因

NSX Manager のインストール時は、3 つのロールすべてに対して 1 つのパスワードのみが受け入れられます。このパスワードを後で変更すると失敗することがあります。

解決方法

- NSX API を使用してパスワードを変更します。

  詳細については、https://kb.vmware.com/s/article/70691 と『NSX-T Data Center 管理ガイド』を参照してください。

障害が発生したワークフローと不安定な NSX Edge のトラブルシューティング

ワークフローで障害が発生している場合や NSX Edge が不安定になっている場合は、トラブルシューティング手順を実行することで対処できます。

問題

vSphere Client で分散ポート グループ設定を変更すると、ワークフローで障害が発生し、NSX Edge が不安定になることがあります。

原因

クラスタ設定の NSX Edge クラスタの設定中に作成されたオーバーレイおよびアップリンクの分散ポート グループを削除または変更することは、設計上許可されていません。

解決方法

NSX Edge の VLAN または IP アドレス プール設定を変更する必要がある場合は、まず NSX-T Data Center の要素と vSphere with Tanzu の設定をクラスタから削除する必要があります。

NSX-T Data Center の要素の削除については、『NSX-T Data Center インストール ガイド』を参照してください。

NSX-T のトラブルシューティングのためのサポート バンドルの収集

トラブルシューティングのために登録済みのクラスタおよびファブリック ノードのサポート バンドルを収集して、そのバンドルをマシンにダウンロードするか、ファイル サーバにアップロードすることができます。

バンドルをマシンにダウンロードする場合は、マニフェスト ファイルと各ノードのサポート バンドルで構成される単一のアーカイブ ファイルが取得されます。バンドルをファイル サーバにアップロードする場合は、マニフェスト ファイルと個々のバンドルがファイル サーバに個別にアップロードされます。

手順

1. ブラウザから、管理者権限で NSX Manager にログインします。
2. [システム] - [サポート バンドル] の順に選択します。
3. ターゲット ノードを選択します。

   選択可能なノードのタイプは、[管理ノード]、[Edge]、[ホスト]、および [Public Cloud Gateway] です。

4 （オプション） ログの有効期間を日数で指定します。指定した日数を経過したログは除外されます。

5 （オプション） コア ファイルおよび監査ログを含めるか除外するかを示すスイッチを切り替えます。

注： コア ファイルと監査ログには、パスワードや暗号化キーなどの機密情報が含まれている場合があります。

6 （オプション） バンドルをファイル サーバにアップロードするには、このチェック ボックスを選択します。

7 [バンドル収集の開始] をクリックして、サポート バンドルの収集を開始します。

各ノードのログ ファイルの数によって、サポート バンドルの収集にかかる時間が決まります。

8 収集プロセスのステータスを監視します。

[ステータス] タブに、サポート バンドルの収集の進行状況が表示されます。

9 バンドルをファイル サーバに送信するオプションを設定しなかった場合は、[ダウンロード] をクリックしてバンドルをダウンロードします。

NSX-T のログ ファイルの収集

エラーの検出やトラブルシューティングのために、vSphere with Tanzu および NSX-T Data Center コンポーネントのログを収集することができます。ログ ファイルは VMware サポートによって要求される場合があります。

手順

1 vSphere Client を使用して、vCenter Server にログインします。

2 以下のログ ファイルを収集します。

ログ ファイル	説明
/var/log/vmware/wcp/wcpsvc.log	vSphere with Tanzu の有効化に関する情報が含まれています。
/var/log/vmware/wcp/nsxd.log	NSX-T Data Center コンポーネントの構成に関する情報が含まれています。

3 NSX Manager にログインします。

4 特定の vSphere with Tanzu の操作が失敗したときに NSX Manager が返すエラーに関する情報については、`/var/log/proton/nsxapi.log` を収集します。

NSX-T の管理証明書、サムプリント、または IP アドレスが変更された場合の WCP サービスの再起動

vSphere with Tanzu のインストール後に NSX-T の管理証明書、サムプリント、または IP アドレスが変更された場合は、`WCP` サービスを再起動する必要があります。

NSX-T 証明書が変更された場合の vSphere with Tanzu サービスの再起動

現在 vSphere with Tanzu では、NSX-T の証明書、サムプリント、または IP アドレスが変更された場合、`WCP` サービスを再起動して変更を有効にする必要があります。サービスを再起動せずに変更が発生した場合、vSphere with Tanzu と NSX-T 間の通信が失敗し、NCP が CrashLoopBackoff ステージに切り替わったり、スーパーバイザー クラスタ リソースがデプロイ不可になったりするなど特定の症状が発生する可能性があります。

WCP サービスを再起動するには、vmon-cli を使用します。

1 vCenter Server に SSH 接続し、root ユーザーとしてログインします。

2 shell コマンドを実行します。

3 vmon-cli -h コマンドを実行して、使用する構文とオプションを表示します。

4 vmon-cli -l コマンドを実行して、wcp プロセスを表示します。

リストの一番下に wcp サービスが表示されます。

5 vmon-cli --restart wcp コマンドを実行し、wcp サービスを再起動します。

Completed Restart service request というメッセージが表示されます。

6 vmon-cli -s wcp コマンドを実行し、wcp サービスが開始されていることを確認します。

例：

```
root@localhost [ ~ ]# vmon-cli -s wcp
Name: wcp
Starttype: AUTOMATIC
RunState: STARTED
RunAsUser: root
CurrentRunStateDuration(ms): 22158
HealthState: HEALTHY
FailStop: N/A
MainProcessId: 34372
```

ホスト トランスポート ノードのトラフィックに必須の Distributed Switch

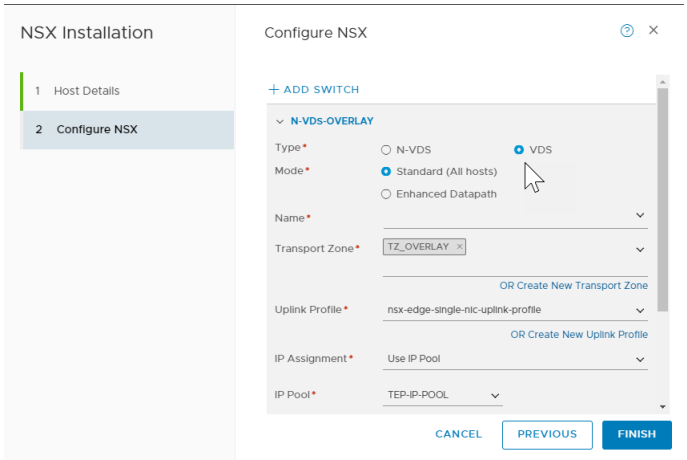
vSphere with Tanzu では、ホスト トランスポート ノードのトラフィックに vSphere 7 Virtual Distributed Switch (VDS) を使用する必要があります。vSphere with Tanzu のホスト トランスポート ノードのトラフィックには、NSX-T Distributed Switch (N-VDS) を使用できません。

Distributed Switch は必須

vSphere with Tanzu では、同一の Distributed Switch 上にある vSphere 7 トラフィックと NSX-T 3 トラフィックの両方をサポートする統合 Distributed Switch が必要です。vSphere および NSX-T の以前のリリースでは、vSphere トラフィック用に 1 台の Distributed Switch (または VSS) と、NSX-T トラフィック用に 1 台の N-VDS があります。この構成は、vSphere with Tanzu ではサポートされていません。N-VDS を使用してワークロード管理を有効にすると、vCenter Server クラスタに互換性がないことが報告されます。詳細については、『ワークロード管理有効化クラスタの互換性エラーのトラブルシューティング』を参照してください。

統合 Distributed Switch を使用するには、vCenter Server を使用して vSphere 7 VDS を作成し、ESXi ホストをトランスポート ノードとして準備する際に NSX-T でこの Distributed Switch を指定します。vCenter Server 側への VDS-DSwitch の配置のみでは不十分です。トランスポート ノード プロファイルの作成のトピックに記載されているとおり、VDS-DSwitch 7.0 は次のように NSX-T トランスポート ノード プロファイルを使用して構成する必要があります。

図 19-1. NSX-T での Distributed Switch の構成



以前のバージョンから vSphere 7 および NSX-T 3 にアップグレードした場合は、各 ESXi トランスポート ノードから N-VDS をアンインストールして、各ホストを Distributed Switch で再構成する必要があります。ガイダンスについては、VMware グローバル サポート サービスにお問い合わせください。

NSX Advanced Load Balancer のトラブルシューティング

発生する可能性のある NSX Advanced Load Balancer の問題のトラブルシューティングを行うには、サポート バンドルを収集します。

トラブルシューティングのためのサポート バンドルの収集

NSX Advanced Load Balancer の問題のトラブルシューティングを行うには、サポート バンドルを収集します。サポート バンドルは VMware のサポートによって要求される場合があります。

サポート バンドルを生成すると、ダウンロードできるデバッグ ログ用の単一のファイルが取得されます。

手順

1 [Avi Controller] ダッシュボードで、左上隅のメニューをクリックし、[管理] を選択します。

2 [管理] セクションで、[システム] を選択します。

3 [システム] 画面で、[Tech Support] を選択します。

4 診断バンドルを生成するには、[Tech Support の生成] をクリックします。

5 [Tech Support の生成] ウィンドウで、[デバッグ ログ] タイプを選択し、[生成] をクリックします。

6 バンドルが生成されたら、[ダウンロード] アイコンをクリックしてマシンにダウンロードします。

ネットワーク トポロジのアップグレードのトラブルシューティング

vSphere with Tanzu バージョン 7.0 Update 1c をインストールするか、スーパーバイザー クラスタ をバージョン 7.0 Update 1 からバージョン 7.0 Update 1c にアップグレードすると、ネットワーク トポロジは単一の

Tier-1 ゲートウェイ トポロジから、スーパーバイザー クラスタ 内の名前空間ごとに 1 つの Tier-1 ゲートウェイがあるトポロジにアップグレードされます。

アップグレード中に発生する可能性のある問題のトラブルシューティングを行うことができます。

Edge ロード バランサのキャパシティ不足によるアップグレード事前チェックの失敗

アップグレードの事前チェックが失敗し、エラー メッセージに、ロード バランサのキャパシティが不足していることが示されます。

問題

アップグレードの事前チェック プロセスが失敗し、ロード バランサのキャパシティが スーパーバイザー クラスタで必要なキャパシティよりも少ないことを示すエラー メッセージが表示されます。

解決方法

この問題を解決するには次のいずれかの手順を実行します。

- エラー メッセージ内の [強制アップグレード] ボタンをクリックしてアップグレードを強制実行するか、vCenter Server コマンド ラインで `--ingnore-precheck-warnings true` フラグを使用します。

  注： このソリューションは、Edge クラスタが既存の名前空間ワークロードをサポートできる場合のみ推奨されます。この条件に当てはまらない場合は、アップグレード中にこれらのワークロードがスキップされることがあります。

- 未使用のワークロードを削除します。
- Edge ノードをクラスタに追加します。

アップグレード中にスキップされる スーパーバイザー クラスタ ワークロードの名前空間

スーパーバイザー クラスタ のアップグレード中に、アップグレードされない名前空間ワークロードがあります。

問題

スーパーバイザー クラスタ のアップグレードは成功しますが、一部の名前空間ワークロードはアップグレード中にスキップされます。Kubernetes リソースはリソースが不足していることを示し、新たに作成された Tier-1 ゲートウェイは `ERROR` 状態になります。

原因

ワークロードをサポートするためのロード バランサのキャパシティが不足しています。

解決方法

この問題を解決するには次のいずれかの手順を実行します。

- 未使用のワークロードを削除し、NCP を再起動して、アップグレードを再度実行します。

- Edge ノードをクラスタに追加して、Tier-1 ゲートウェイへの再割り当てをトリガします。NCP を再起動して、アップグレードを再実行します。

アップグレード中にスキップされるロード バランサ サービス

スーパーバイザー クラスタ のアップグレード中に、アップグレードされないロード バランサ サービスがあります。

問題

スーパーバイザー クラスタ のアップグレードは成功しますが、一部の Kubernetes ロード バランサ サービスはアップグレード中にスキップされます。

原因

スーパーバイザー クラスタ のワークロードおよび関連付けられた Tanzu Kubernetes クラスタ内の Kubernetes ロード バランサ タイプのサービスの数が、NSX Edge 仮想サーバの制限を超えています。

解決方法

未使用のワークロードを削除し、NCP を再起動して、アップグレードを再度実行します。

Tanzu Kubernetes クラスタのトラブルシューティング

Tanzu Kubernetes クラスタのトラブルシューティングを行うには、いずれかのクラスタ ノードに接続し、クラスタのリソース階層を表示して、ログ ファイルを収集します。

Tanzu Kubernetes クラスタのサポート バンドルの収集

Tanzu Kubernetes クラスタ エラーのトラブルシューティングを行うには、ユーティリティを実行して、診断ログ バンドルを収集します。

VMware では、Tanzu Kubernetes クラスタのログ ファイルの収集と問題のトラブルシューティングに使用できる TKC Support Bundler ユーティリティを提供しています。

このユーティリティを入手して使用するには、VMware サポートのナレッジベースの記事 How to collect a diagnostic log bundle from a Tanzu Kubernetes cluster (KB80949)を参照してください。

vCenter Single Sign-On 接続エラーのトラブルシューティング

vSphere 名前空間 に関して必要な権限がない場合は、vCenter Single Sign-On ユーザーとして スーパーバイザー クラスタ または Tanzu Kubernetes クラスタに接続できません。

問題

vCenter Single Sign-On ユーザーとして スーパーバイザー クラスタ または Tanzu Kubernetes クラスタに接続しようとすると、kubectl 向けの vSphere プラグイン から「`Error from server (Forbidden)`」というエラー メッセージが返されます。

原因

vSphere 名前空間 に関して必要な権限がないか、クラスタにアクセスできません。

解決方法

操作しているユーザーが、クラスタを運用している DevOps エンジニアである場合は、vSphere 名前空間 の 編集 権限が自分に付与されているか vSphere 管理者に確認してください。vSphere 名前空間 の作成と設定を参照してください。

操作しているユーザーが、クラスタを使用してワークロードをデプロイしている開発者である場合は、クラスタへのアクセス権が自分に付与されているかクラスタ管理者に確認してください。開発者に対する Tanzu Kubernetes クラスタへのアクセス権の付与を参照してください。

サブスクライブ済みコンテンツ ライブラリのエラーのトラブルシューティング

サブスクライブ済みコンテンツ ライブラリがストレージ容量の制限に達した場合は、Tanzu Kubernetes クラスタをプロビジョニングできません。

問題

Tanzu Kubernetes クラスタをプロビジョニングしようとすると、サブスクライブ済みコンテンツ ライブラリからアイテムを取得することができず、次のエラーが表示されます。

```
Internal error occurred: get library items failed for.
```

原因

サブスクライブ済みコンテンツ ライブラリが容量に達しています。コンテンツ ライブラリは、接続されたストレージによってバッキングされます。時間の経過とともにリリースされる Kubernetes のバージョンが増えて、OVA ファイルがライブラリに取り込まれるため、ストレージ容量がいっぱいになることがあります。

解決方法

新しいコンテンツ ライブラリに移行します。Tanzu Kubernetes クラスタの新しいコンテンツ ライブラリへの移行を参照してください。

ローカル コンテンツ ライブラリのエラーのトラブルシューティング

ローカル コンテンツ ライブラリの一般的なエラーのトラブルシューティングについては、このトピックを参照してください。

スーパーバイザー クラスタ に TKR が見つからない

ローカル コンテンツ ライブラリは、インターネット制限環境で使用できます。ローカル コンテンツ ライブラリを作成するには、Tanzu Kubernetes リリース 用のローカル コンテンツ ライブラリの作成、セキュリティ保護、同期を参照してください。

コンテンツ ライブラリにセキュリティ ポリシーを適用する際、次のいずれかの条件が存在する場合、スーパーバイザー クラスタ に Tanzu Kubernetes リリース (TKR) は表示されません。

- コンテンツ ライブラリ内の OVF パッケージに署名がない。
- OVF パッケージが無効な証明書で署名されている。

- OVF パッケージは、ローカル コンテンツ ライブラリが構成されている vCenter Server によって信頼されていない証明書で署名されている。

クラスタ プロビジョニング エラーのトラブルシューティング

Tanzu Kubernetes クラスタをプロビジョニングした際に制御プレーンの仮想マシンが起動しない場合は、仮想マシンのサイズまたはクラスの変更が必要になることがあります。

問題

Tanzu Kubernetes クラスタをプロビジョニングしました。システムは制御プレーンの仮想マシンのパワーオンを試行しますが、エラーが発生し、次のメッセージが表示されます。

```
The host does not have sufficient CPU resources to satisfy the reservation.
```

原因

仮想マシンのサイズまたはクラスが、クラスタをデプロイするのに十分ではありません。

解決方法

仮想マシンのタイプまたはクラスを変更します。Tanzu Kubernetes クラスタの仮想マシンのクラスを参照してください。

ワークロード デプロイ エラーのトラブルシューティング

認証されたユーザーに PodSecurityPolicy とバインドが設定されていない場合は、ワークロード デプロイ エラーが発生することがあります。

問題

コンテナ ワークロードを Tanzu Kubernetes クラスタにデプロイしても、ワークロードが起動しません。次のようなエラーが表示されます。

```
Error: container has runAsNonRoot and image will run as root.
```

原因

Tanzu Kubernetes クラスタは、PodSecurityPolicy アドミッション コントローラが有効になるようにプロビジョニングされています。認証されたユーザーは、クラスタ管理者が認証されたユーザーに PodSecurityPolicy をバインドするまで、権限のあるポッドまたは権限のないポッドを作成できません。

解決方法

デフォルトの PodSecurityPolicy に適したバインドを作成するか、カスタムの PodSecurityPolicy を定義します。『Tanzu Kubernetes クラスタでのポッド セキュリティ ポリシーの使用』と『Tanzu Kubernetes Guestbook のチュートリアル』を参照してください。

仮想マシン クラスのエラーに関するトラブルシューティング

仮想マシン クラスは、Tanzu Kubernetes クラスタおよび仮想マシン クラスの vSphere 名前空間にバインドする必要があります。

仮想マシン クラス バインド エラー

仮想マシン クラスは vSphere 名前空間にバインドする必要があります。仮想マシン クラスと vSphere with Tanzu の名前空間の関連付けを参照してください。仮想マシン クラスを名前空間に関連付けていない場合は、次のような `VirtualMachineClassBindingNotFound` エラーが表示されます。

```
conditions:
  - lastTransitionTime: "2021-04-25T02:50:58Z"
    message: 1 of 2 completed
    reason: VirtualMachineClassBindingNotFound @ Machine/test-cluster
    severity: Error
    status: "False"
    type: ControlPlaneReady
  - lastTransitionTime: "2021-04-25T02:49:21Z"
    message: 0/1 Control Plane Node(s) healthy. 0/2 Worker Node(s) healthy
    reason: WaitingForNodesHealthy
    severity: Info
    status: "False"
    type: NodesHealthy
```

失敗した Tanzu Kubernetes クラスタ更新ジョブの再開

Tanzu Kubernetes クラスタの更新に失敗した場合は、更新ジョブを再開して更新をもう一度試すことができます。

問題

Tanzu Kubernetes クラスタの更新に失敗し、クラスタの状態が `upgradefailed` になります。

原因

クラスタ更新が失敗した場合、ストレージの不足などいくつかの理由が考えられます。失敗した更新ジョブを再開してクラスタ更新をもう一度試すには、次の手順を実行します。

解決方法

1. 管理者として スーパーバイザー クラスタ にログインします。vCenter Single Sign-On ユーザーとして スーパーバイザー クラスタ に接続するを参照してください。
2. `update_job_name` を検索します。

   ```
   kubectl get jobs -n vmware-system-tkg -l "run.tanzu.vmware.com/cluster-namespace=$
   {cluster_namespace},cluster.x-k8s.io/cluster-name=${cluster_name}"
   ```

3. `curl` を使用して要求を発行できるように、`kubectl proxy` を実行します。

   ```
   kubectl proxy &
   ```

`Starting to serve on 127.0.0.1:8001` が表示されます。

注： `kubectl` を使用して、リソースの `.status` にパッチを適用したり、更新したりすることはできません。

4 `curl` を使用して次のパッチコマンドを発行し `.spec.backoffLimit` を発生させます。

```
curl -H "Accept: application/json" -H "Content-Type: application/json-patch+json"
--request PATCH --data '[{"op": "replace", "path": "/spec/backoffLimit", "value": 8}]'
http://127.0.0.1:8001/apis/batch/v1/namespaces/vmware-system-tkg/jobs/${update_job_name}
```

5 ジョブ コントローラで新しいポッドが作成されるように、`curl` を使用して次のパッチコマンドを発行し `.status.conditions` をクリアします。

```
$ curl -H "Accept: application/json" -H "Content-Type: application/json-patch+json"
--request PATCH --data '[{"op": "remove", "path": "/status/conditions"}]'
http://127.0.0.1:8001/apis/batch/v1/namespaces/vmware-system-tkg/jobs/${update_job_name}/
status
```

ワークロード管理のトラブルシューティング

ワークロード管理のトラブルシューティングを行う場合、またはサポート バンドルを収集する必要がある場合は、このセクションを参照してください。

ワークロード管理のサポート バンドルの収集

この手順に従って、ワークロード管理のサポート バンドルを収集します。

次のタスクを実行して、スーパーバイザー クラスタ のログを取得します。これは、クラスタがエラー状態または構成中の状態で停止している場合でも実行できます。

手順

1 vSphere Client を使用して、vSphere with Tanzu 環境にログインします。

2 [メニュー] - [ワークロー管理] の順に選択します。

3 [クラスタ] タブを選択します。

4 ターゲット スーパーバイザー クラスタ を選択します。

5 [ログのエクスポート] を選択します。

結果

サポート バンドルを収集したら、ナレッジベースの記事「Secure FTP ポータル経由での VMware への診断情報のアップロード」(http://kb.vmware.com/kb/2069559) を参照してください。

ワークロード管理のログ ファイルのテール

ワークロード管理のログ ファイルをテールすると、有効化の問題やスーパーバイザー クラスタのデプロイ エラーのトラブルシューティングに役立ちます。

解決方法

1 vCenter Server Appliance への SSH 接続を確立します。

2 `root` ユーザーとしてログインします。

3 `shell` コマンドを実行します。

次のメッセージが表示されます。

```
Shell access is granted to root
root@localhost [ ~ ]#
```

4 次の コマンドを実行して、ログをテールします。

```
tail -f /var/log/vmware/wcp/wcpsvc.log
```

ワークロード管理有効化クラスタの互換性エラーのトラブルシューティング

vSphere クラスタにワークロード管理を有効にするための互換性がないことをシステムが示す場合は、次のトラブルシューティングのヒントを参照してください。

問題

ワークロード管理を有効にしようとしたときに、vCenter Server クラスタに互換性がないことが [ワークロード管理] 画面に示されます。

原因

これには複数の原因があります。まず、環境がワークロード管理を有効にするための最小要件を満たしていることを確認します。

- 有効なライセンス：Kubernetes のアドオンを含む VMware vSphere 7 Enterprise Plus
- 少なくとも 2 台の ESXi ホスト
- 完全に自動化された DRS
- vSphere HA
- vSphere Distributed Switch 7.0
- 十分なストレージ容量

環境がこれらの前提条件を満たしているが、ターゲットの vCenter Server クラスタに互換性がない場合は、VMware Datacenter CLI (DCLI) を使用して問題を特定します。

解決方法

1 vCenter Server に SSH 接続します。

2 root ユーザーとしてログインします。

3 `dcli` コマンドを実行して、VMware Datacenter CLI のヘルプを一覧表示します。

4 次の DCLI コマンドを実行して、使用可能な vCenter Server クラスタを一覧表示します。

```
dcli com vmware vcenter cluster list
```

例：

```
dcli +username VI-ADMIN-USER-NAME +password VI-ADMIN-PASSWORD com vmware vcenter cluster
list
```

結果の例：

```
|-----------|---------|------------|----------|
|drs_enabled|cluster  |name        |ha_enabled|
|-----------|---------|------------|----------|
|True       |domain-d7|vSAN Cluster|True      |
|-----------|---------|------------|----------|
```

5 次の DCLI コマンドを実行して、vCenter Server クラスタの互換性を確認します。

```
dcli com vmware vcenter namespacemanagement clustercompatibility list
```

例：

```
dcli +username VI-ADMIN-USER-NAME +password VI-ADMIN-PASSWORD com vmware vcenter
namespacemanagement clustercompatibility list
```

次の結果の例は、互換性のある NSX-T Distributed Switch が環境に存在しないことを示しています。

```
|---------|----------|------------------------------------------------------------------------
-------------------|
|cluster  |compatible|
incompatibility_reasons                                                                  |
|---------|----------|------------------------------------------------------------------------
-------------------|
|domain-d7|False     |Failed to list all distributed switches in vCenter 2b1c1fa5-
e9d4-45d7-824c-fa4176da96b8.|
|         |          |Cluster domain-d7 is missing compatible NSX-T
VDS.                                    |
|---------|----------|------------------------------------------------------------------------
-------------------|
```

6 必要に応じて追加の DCLI コマンドを実行して、その他の互換性の問題を特定します。非互換の一般的な理由には、NSX-T エラーのほかに、DNS や NTP 接続の問題があります。

7 さらにトラブルシューティングを行うには、次の手順を実行します。

a `wcpsvc.log` ファイルをテールします。ワークロード管理のログ ファイルのテールを参照してください。

b [ワークロード管理] 画面に移動し [有効化] をクリックします。

vSphere with Tanzu ワークロード ドメインのシャットダウンと起動

データ損失を回避し、vSphere with Tanzu 環境のコンポーネントおよびワークロードを運用可能な状態に維持するには、コンポーネントのシャットダウンまたは起動時に、特定の順序で実行する必要があります。

通常は、vSphere with Tanzu 環境にパッチの適用、アップグレード、またはリストアを実行した後に、シャットダウンおよび起動の操作を実行します。

Tanzu Kubernetes Grid サービス によってプロビジョニングされた Tanzu Kubernetes クラスタを含む vSphere with Tanzu ソリューションは、vSphere Software-Defined Data Center (SDDC) の一部です。したがって、vSphere with Tanzu 環境をシャットダウンして起動する場合は、vSphere インフラストラクチャ スタック全体を考慮する必要があります。vSphere with Tanzu を含む vSphere SDDC のシャットダウンと起動については、次に示す検証済みの一連の手順を参照してください。

- vSphere with Tanzu を含む vSphere SDDC のシャットダウン手順
- vSphere with Tanzu を含む vSphere SDDC の起動手順