vSphere IaaS 制御プレーンのサービスとワークロード

Update 3

VMware vSphere 8.0
VMware vCenter 8.0
VMware ESXi 8.0

vmware by Broadcom

VMware by Broadcom の Web サイトで最新の技術ドキュメントを確認できます

https://docs.vmware.com/jp/

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

目次

vSphere IaaS 制御プレーンのサービスとワークロード について

1

『vSphere IaaS 制御プレーンのサービスとワークロード』（旧称『vSphere with Tanzu のサービスとワークロード』）では、vSphere IaaS control plane（旧称 vSphere with Tanzu）環境内の vSphere 名前空間 でサービスとワークロードを実行する方法について説明しています。名前空間を作成し、スーパーバイザー サービス、仮想マシン、vSphere ポッド などのワークロードをデプロイする方法について学習できます。

対象読者

この情報は主に、vSphere IaaS control plane を使用して vSphere リソースを構成し、スーパーバイザー の vSphere 名前空間 に割り当てる vSphere 管理者を対象としています。これらのリソースは、名前空間内で実行されるさまざまなサービスとワークロードで後で使用することができます。vSphere IaaS control plane を使用する vSphere 管理者は通常、コンテナおよびその他の Kubernetes の概念に関する基本的な知識を持っています。

このガイドは、スーパーバイザー で vSphere ポッド、仮想マシン、および スーパーバイザー サービス などのワークロードをデプロイする DevOps チームも使用することができます。

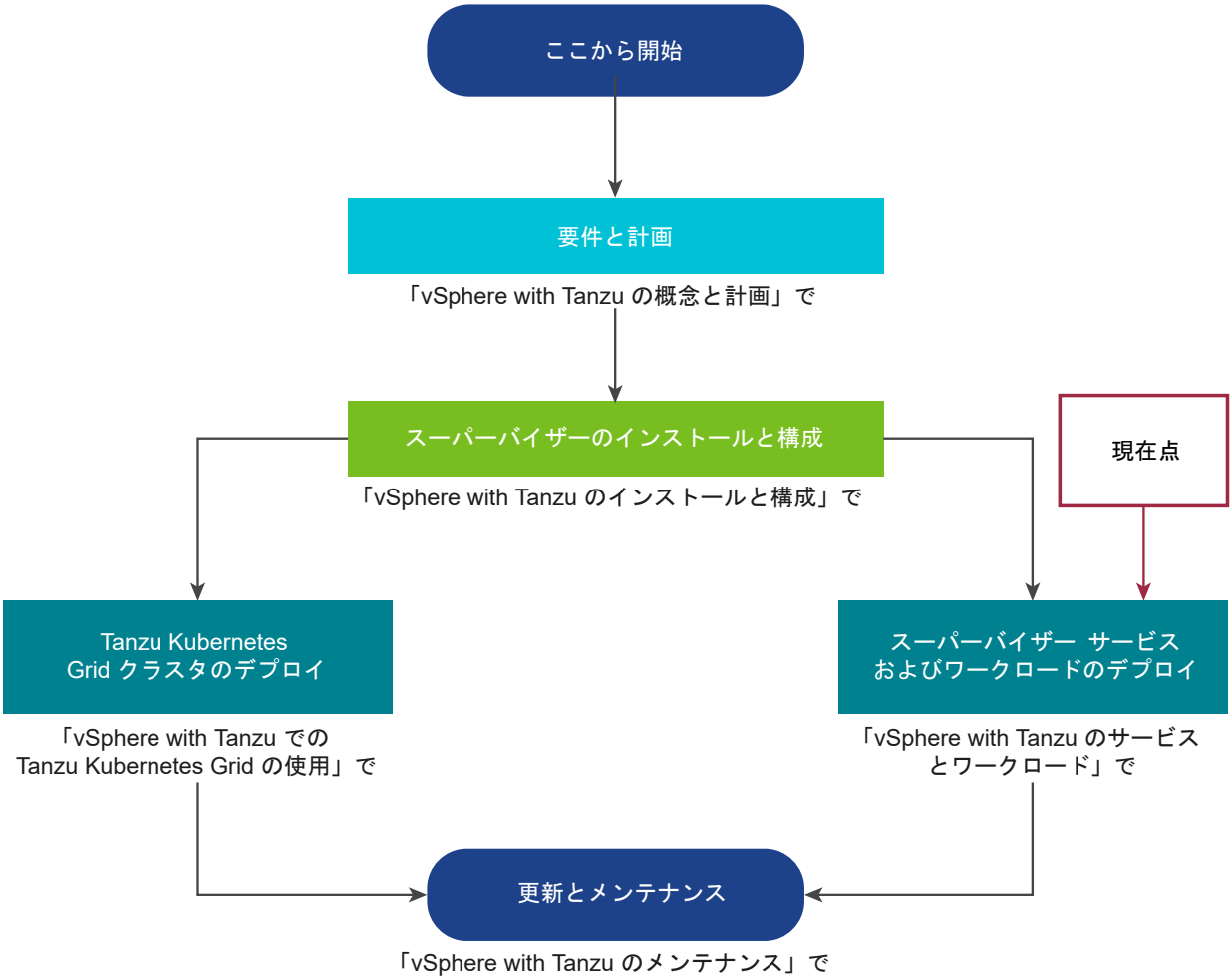
注： vSphere IaaS 制御プレーンのサービスとワークロードには、Tanzu Kubernetes Grid クラスタで実行されているワークロードの情報は含まれていません。Tanzu Kubernetes Grid クラスタを使用する方法については、「Using Tanzu Kubernetes Grid on Supervisor with vSphere IaaS Control Plane」を参照してください。

vSphere IaaS 制御プレーンのサービスとワークロード ドキュメントの使用

vSphere IaaS control plane ドキュメントには、この『vSphere IaaS 制御プレーンのサービスとワークロード』ガイドの他にもいくつかのガイドが含まれています。ガイドによっては、この vSphere IaaS 制御プレーンのサービスとワークロードの前提条件となる場合があるため、vSphere IaaS control plane ドキュメントの階層について理解していることを確認してください。

この図では、vSphere IaaS control plane ドキュメント セットの使用方法と、各ガイドで確認できる情報について説明します。

図 1-1. vSphere IaaS control plane ドキュメント マップ



更新情報

『vSphere IaaS 制御プレーンのサービスとワークロード』は、製品のリリースごとに、または必要に応じて更新されます。

『vSphere IaaS 制御プレーンのサービスとワークロード』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
2024 年 6 月 25 日	次の機能を含む vSphere 8.0 Update 3 リリースの一般的な更新と機能強化。 ■ エアギャップ環境への スーパーバイザー サービス のデプロイ。10 章 プロキシからイメージをプルしてエアギャップ環境に スーパーバイザー サービス をデプロイするを参照してください。 ■ 名前空間に対するネットワークとロード バランサのパラメータの構成。名前空間に対するネットワークとロード バランサのパラメータの構成を参照してください。 ■ 仮想マシン オペレータ v1alpha2 のサポート。6 章 vSphere IaaS control plane での仮想マシンのデプロイと管理を参照してください。 ■ `kubectl get virtualmachineclass` を使用して、特定の スーパーバイザー 名前空間の仮想マシン クラスを一覧表示する機能。以前は、仮想マシン クラスはクラスタ スコープのリソースであり、どの仮想マシン クラスが特定の名前空間に割り当てられていたかを簡単には判断できませんでした。vSphere IaaS control plane 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。
2024 年 4 月 29 日	推奨イメージからダウンロードできる OVA イメージに関するメモを追加しました。vSphere IaaS control plane 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。
2024 年 4 月 1 日	vSphere IaaS control plane での vGPU および他の PCI デバイスを含む仮想マシンのデプロイを更新し、NVIDIA GRID vGPU で使用される詳細パラメータに関する情報を追加しました。
2024 年 2 月 29 日	製品の変更に合わせて、仮想マシン クラスの編集に関するコンテンツを更新しました。vSphere Client を使用した仮想マシン クラスの編集を参照してください。
2023 年 12 月 11 日	vSAN ファイル サービスによってバッキングされる RWX ボリュームの制限に関する記述を追加しました。vSAN ファイル サービスを使用した vSphere IaaS control plane での ReadWriteMany ボリュームの作成を参照してください。
2023 年 11 月 7 日	vSAN ファイル サービスを使用した vSphere IaaS control plane での ReadWriteMany ボリュームの作成を更新して、ファイル ボリュームが Tanzu Kubernetes Grid クラスタ内のワークロードでのみサポートされていることを明確にしました。
2023 年 9 月 29 日	■ スーパーバイザー サービス に関する新しいプラットフォーム サポート情報を追加しました。4 章 vSphere IaaS control plane 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。 ■ マイナー更新。
2023 年 9 月 21 日	■ データセンター CLI (DCLI) コマンドを使用してコンテンツ ライブラリを名前空間または スーパーバイザー に関連付ける方法に関する新しいコンテンツを追加しました。vSphere IaaS control plane での仮想マシン コンテンツ ライブラリの追加と管理を参照してください。 ■ 名前空間に関連付けられた書き込み可能コンテンツ ライブラリに新しい仮想マシン イメージを公開する方法に関するトピックを追加しました。vSphere IaaS control plane でのコンテンツ ライブラリ イメージの管理と公開を参照してください。 ■ DCLI コマンドを使用して仮想マシン クラスを作成および管理する方法に関する新しいコンテンツを追加しました。Data Center CLI を使用した仮想マシン クラスの作成と管理を参照してください。

リビジョン	説明
2023 年 7 月 14 日	■ スーパーバイザー サービス の使用方法の説明を強化しました。4 章 vSphere IaaS control plane 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。 ■ 「スーパーバイザー での vSphere 名前空間 の作成と構成」に Tier-0 ワークロード ネットワーク設定をオーバーライドするための要件を追加しました。
2023 年 6 月 30 日	スーパーバイザー からの vSphere 名前空間 の削除 のトピックを追加しました。
2023 年 6 月 21 日	StatefulSet 定義を使用するときに StatefulSet の一部として作成されたボリュームを拡張できないことを明確にしました。vSphere IaaS control plane でのボリュームの拡張を参照してください。
2023 年 5 月 16 日	vSphere 8 Update 1 リリース以降では、NSX と VDS の両方のタイプのネットワークを使用してデプロイされた スーパーバイザー で スーパーバイザー サービス が使用できることを示す注意事項を追加しました。4 章 vSphere IaaS control plane 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。
2023 年 5 月 15 日	マイナー改訂。
2023 年 5 月 11 日	■ 2 章 vSphere IaaS control plane サービスとワークロードのワークフロー にワークロード サポート テーブルを追加しました。 ■ 未使用の場合、2 分以内に期限切れになる URL を仮想マシンの Web コンソールに追加しました。vSphere 仮想マシン Web コンソールを使用した仮想マシンのトラブルシューティングを参照してください。 ■ vSphere IaaS control plane ドキュメントの階層を示す図を追加しました。vSphere IaaS 制御プレーンのサービスとワークロード ドキュメントの使用を参照してください。
2023 年 5 月 9 日	■ vSAN ファイル サービスによってバッキングされている ReadWriteMany ボリュームがボリューム拡張をサポートしていないという情報を追加しました。vSphere IaaS control plane でのボリュームの拡張を参照してください。 ■ 7 章 vSphere ポッド へのワークロードのデプロイに、vSphere ポッド に関する概念の詳細を追加しました。
2023 年 5 月 5 日	■ 9 章 vSphere IaaS control plane での Harbor と Contour のインストールと構成 の章を更新しました。 ■ vSphere ポッド は NSX ネットワーク スタックでのみサポートされるという情報を追加しました。 ■ vSphere ポッド デプロイの例を追加しました。『vSphere IaaS control plane での vSphere ポッド ワークロードのデプロイ』を参照してください
2023 年 4 月 26 日	3 章 vSphere 名前空間の構成と管理の章を追加しました。
2023 年 4 月 18 日	■ 構成可能な OVF プロパティを使用した仮想マシンのデプロイに関するトピックを追加しました。vSphere IaaS control plane での構成可能な OVF プロパティを使用した仮想マシンのデプロイを参照してください。 ■ DevOps エンジニアがトラブルシューティングを目的として仮想マシンとそのゲスト OS に直接アクセスするために使用できる vSphere の仮想マシン Web コンソールに関する情報を追加しました。vSphere 仮想マシン Web コンソールを使用した仮想マシンのトラブルシューティングを参照してください。

vSphere IaaS control plane サービスとワークロードのワークフロー

2

これらのワークフローは、vSphere IaaS control plane をすでに有効にしていること、スーパーバイザー を設定していること、および vSphere 名前空間 を作成してワークロードに使用する準備ができていることが前提となります。

ユーザー ロール

通常、スーパーバイザー とやり取りしてワークロードを実行するには、vSphere 管理者と DevOps エンジニアの 2 つのロールが必要です。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。

vSphere 管理者

vSphere 管理者は通常、vSphere Client を使用して、DevOps エンジニアが Kubernetes ワークロードをデプロイできる スーパーバイザー と名前空間を構成します。

スーパーバイザー を作成しておらず、作成方法を確認する必要がある場合は、vSphere IaaS 制御プレーンのインストールと構成を参照してください。

DevOps エンジニア

スーパーバイザー では、DevOps エンジニアのロールは、Kubernetes 開発者、アプリケーション所有者、および Kubernetes 管理者によって通常実行されるアクティビティの組み合わせになることがあります。DevOps エンジニアは、kubectl コマンドを使用します。vSphere ポッド、仮想マシン、その他のワークロードは、vSphere 管理者が作成した スーパーバイザー 名前空間にデプロイして実行できます。セルフサービスの名前空間を作成することもできます。

この vSphere IaaS 制御プレーンのサービスとワークロードガイドでは、DevOps エンジニアが Tanzu Kubernetes Grid クラスタで実行するタスクについては説明していません。これらのタスクの詳細については、vSphere IaaS 制御プレーンを使用したスーパーバイザーでの Tanzu Kubernetes Grid の使用を参照してください。

スーパーバイザー でサポートされるワークロードのタイプ

さまざまなタイプのワークロードに対する スーパーバイザー のサポートは、スーパーバイザー で使用される構成とネットワークによって異なります。

ワークロードのタイプ	Distributed Switch ネットワークを使用する 1 ゾーン スーパーバイザー	NSX ネットワークを使用する 1 ゾーン スーパーバイザー	Distributed Switch ネットワークを使用する 3 ゾーン スーパーバイザー	NSX ネットワークを使用する 3 ゾーン スーパーバイザー
7 章 vSphere ポッド へのワークロードのデプロイ	なし	はい	いいえ	なし
6 章 vSphere IaaS control plane での仮想マシンのデプロイと管理	はい	はい	はい	はい
4 章 vSphere IaaS control plane を使用したスーパーバイザー サービスの管理	はい	はい	いいえ	なし
Tanzu Kubernetes Grid クラスタ	はい	はい	はい	はい

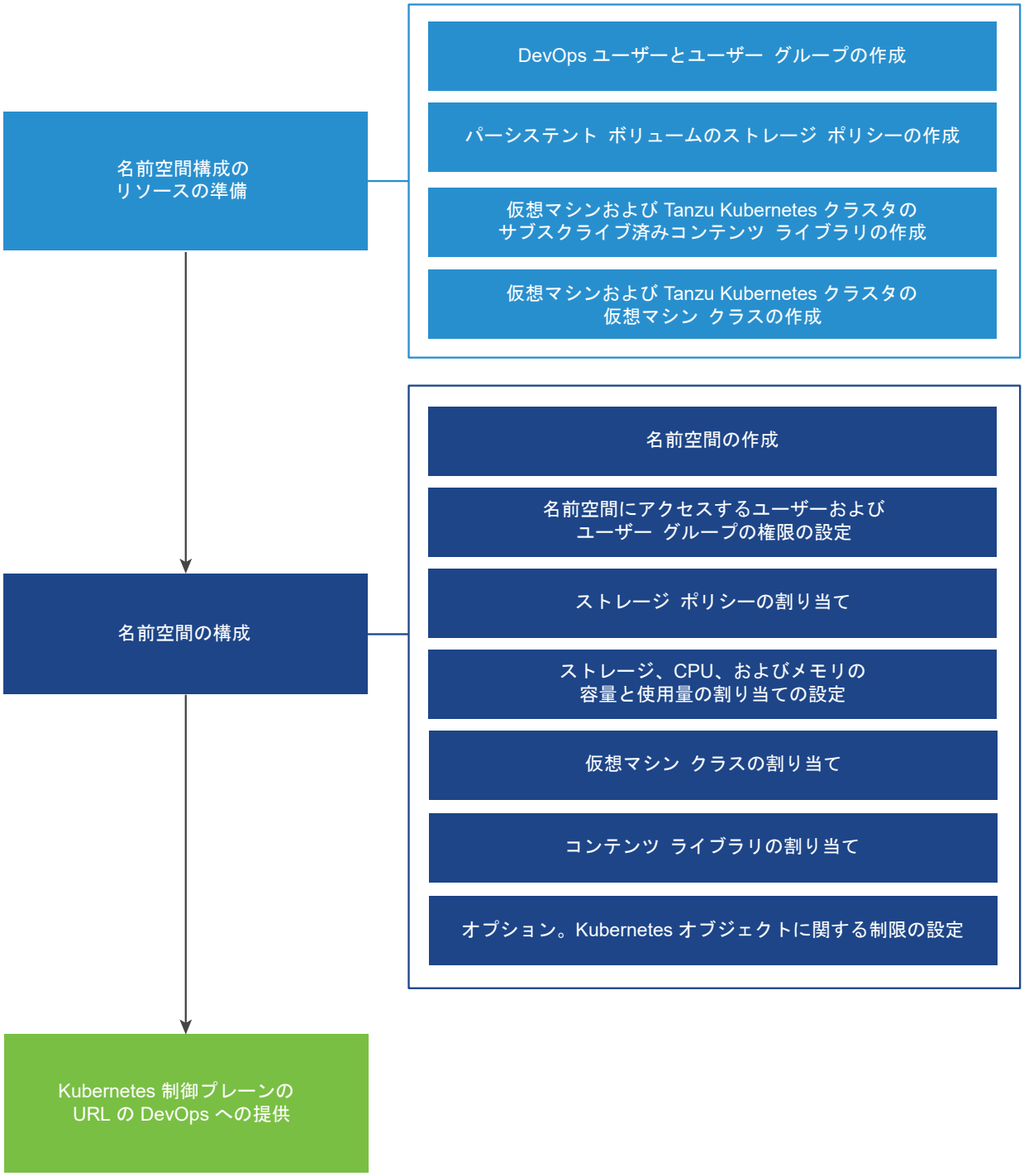
名前空間の構成のワークフロー

vSphere 管理者は、スーパーバイザー で vSphere 名前空間 を作成および管理できます。Tanzu Kubernetes Grid クラスタ

名前空間を作成する前に、DevOps エンジニアから実行するアプリケーションとワークロードに関する特定のリソース要件を収集する必要があります。これらの仕様に基づいて適切なリソースを構成し、名前空間に割り当てることができます。

詳細については、『3 章 vSphere 名前空間の構成と管理』を参照してください。

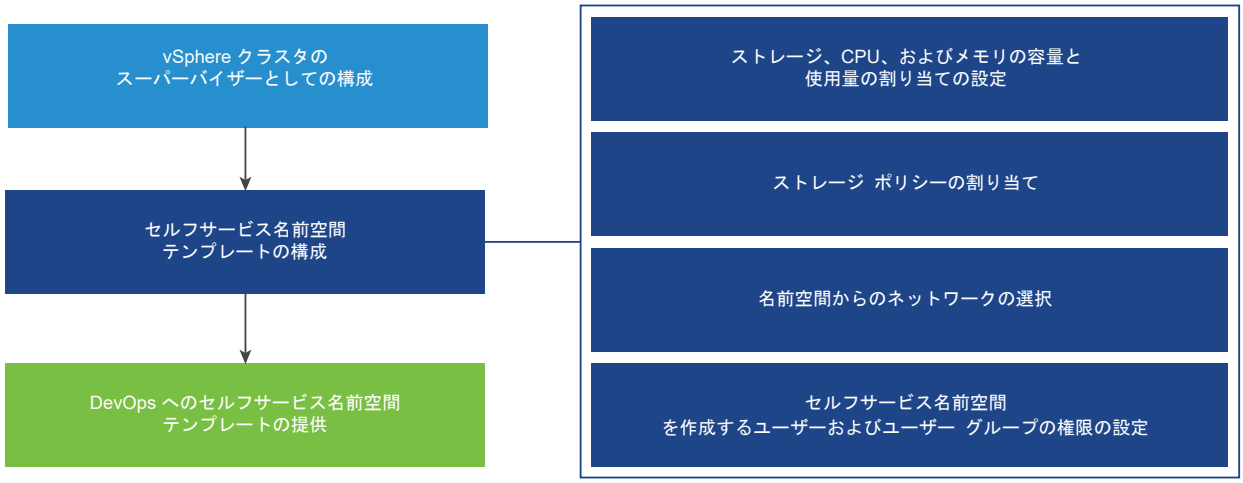
図 2-1. 名前空間の構成のワークフロー



セルフサービス名前空間の構成のワークフロー

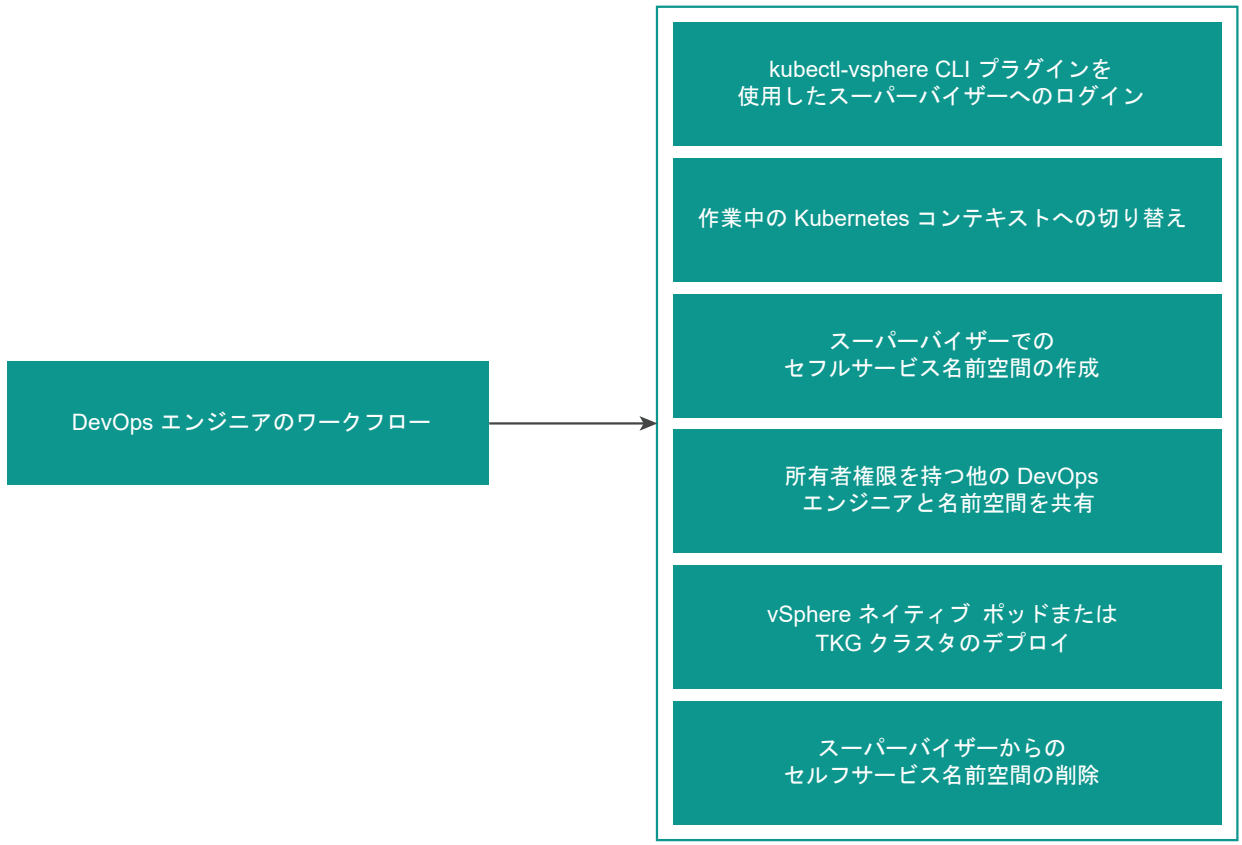
vSphere 管理者は、vSphere 名前空間 の作成、名前空間に対する CPU、メモリ、ストレージの制限の設定、権限の割り当て、およびクラスタの名前空間サービスのテンプレートとしてのプロビジョニングと有効化を行うことができます。詳細については、『vSphere IaaS control plane でのセルフサービス名前空間テンプレートのプロビジョニング』を参照してください。

図 2-2. セルフサービス名前空間の vSphere 管理者ワークフロー



DevOps エンジニアは、セルフ サービス方式で vSphere 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 2-3. セルフサービス名前空間の DevOps ワークフロー

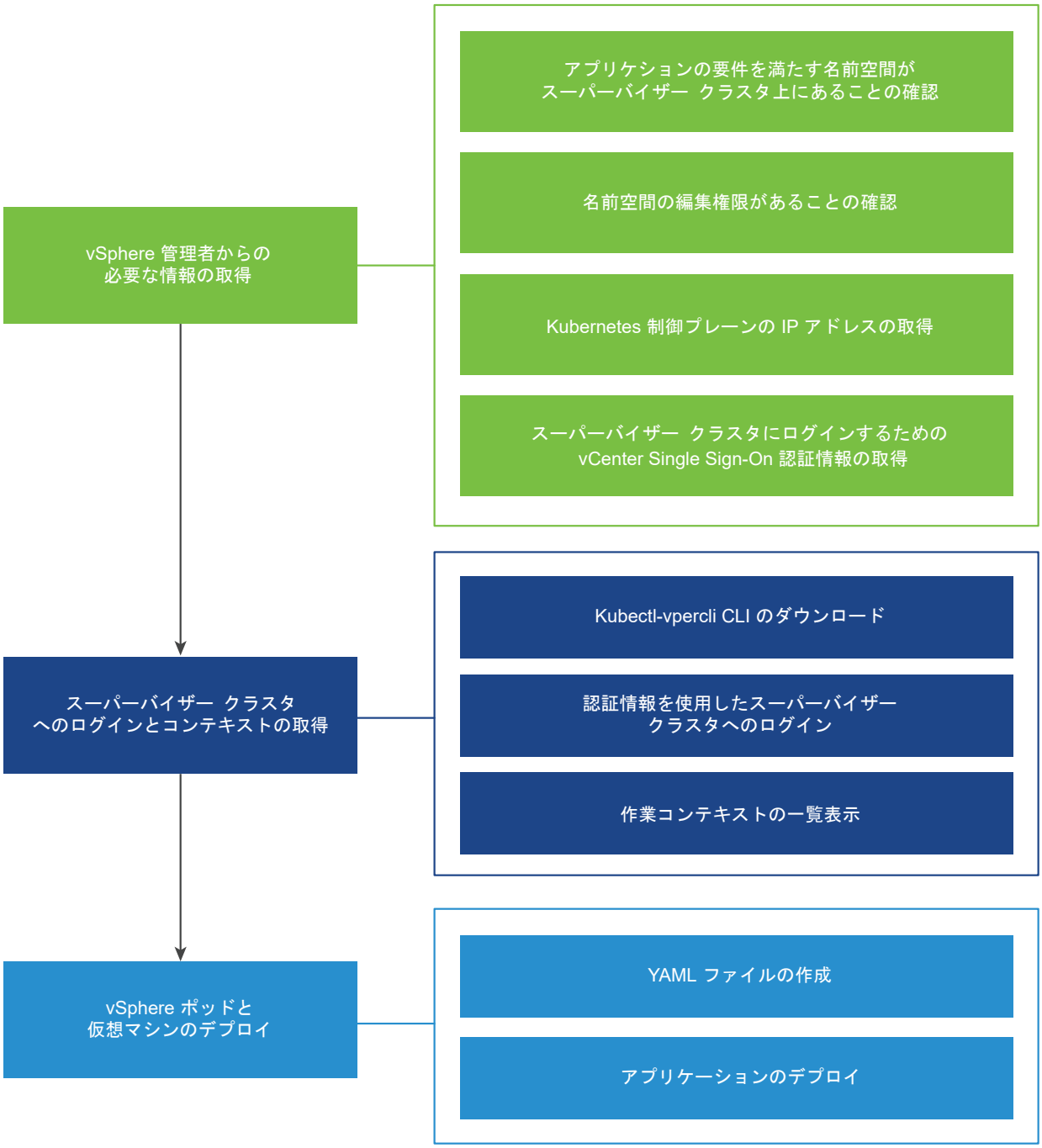


vSphere ポッド および仮想マシンをプロビジョニングするためのワークフロー

DevOps エンジニアは、スーパーバイザー 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、『7 章 vSphere ポッド へのワークロードのデプロイ』および『6 章 vSphere IaaS control plane 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 2-4. 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



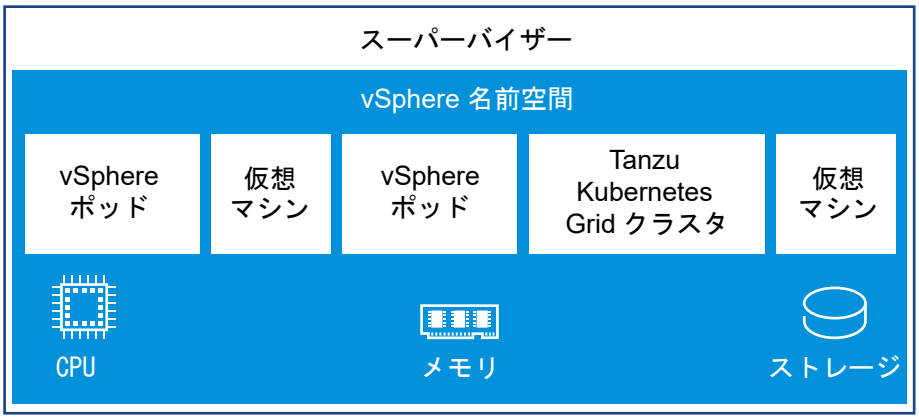
vSphere 名前空間の構成と管理

3

vSphere ポッド、仮想マシン、Tanzu Kubernetes クラスタなどの vSphere IaaS control plane ワークロードは、vSphere 名前空間 にデプロイされます。vSphere 管理者は、スーパーバイザー で名前空間を定義し、リソース割り当てとユーザー権限を使用して名前空間を構成します。DevOps に関するニーズと実行予定のワークロードに応じて、vSphere 管理者はストレージ 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、仮想マシン イメージを取得することもできます。

最初に作成された名前空間には、スーパーバイザー 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。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。リソース プールは、スーパーバイザー の名前空間ごとに vSphere 内に作成されます。

vSphere Zones で有効になっている スーパーバイザー では、ゾーンにマッピングされた各 vSphere クラスタに名前空間リソース プールが作成されます。3 ゾーンの スーパーバイザー の名前空間で使用されるリソースは、基盤となる 3 つのすべての vSphere クラスタから均等に取得されます。たとえば、300 MHz の CPU を使用する場合は、各 vSphere クラスタから 100 MHz が取得されます。



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、または スーパーバイザー に登録された ODIC プロバイダから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。詳細については、「vSphere IaaS 制御プレーンの ID とアクセス管理」を参照してください。

名前空間が作成され、リソース、オブジェクトの制限、権限、およびストレージ ポリシーが構成されたら、DevOps エンジニアは名前空間にアクセスして、次のワークロードを実行することができます。

- スーパーバイザー サービス の実行の詳細については、「4 章 vSphere IaaS control plane 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。
- vSphere ポッド の実行の詳細については、「7 章 vSphere ポッド へのワークロードのデプロイ」を参照してください。
- 仮想マシンのデプロイの詳細については、「6 章 vSphere IaaS control plane での仮想マシンのデプロイと管理」を参照してください。

注： この『vSphere IaaS 制御プレーンのサービスとワークロード』ガイドには、Tanzu Kubernetes Grid クラスタで実行されているワークロードの情報は含まれていません。Tanzu Kubernetes Grid クラスタを使用する方法については、「Using Tanzu Kubernetes Grid on Supervisor with vSphere IaaS Control Plane」を参照してください。

次のトピックを参照してください。

- スーパーバイザー での vSphere 名前空間 の作成と構成
- スーパーバイザー からの vSphere 名前空間 の削除
- vSphere 名前空間 へのリソース制限の設定
- vSphere 名前空間 でのオブジェクト制限の構成
- vSphere 名前空間 でのリソースの監視と管理
- vSphere IaaS control plane での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 名前空間のストレージ設定の変更
- NSX vSphere 名前空間へのセキュリティ ポリシーの追加
- 名前空間に対するネットワークとロード バランサのパラメータの構成

スーパーバイザー での vSphere 名前空間 の作成と構成

スーパーバイザー に vSphere 名前空間 を作成して構成する方法を確認します。vSphere 管理者は vSphere 名前空間 の作成後、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ワークロードを実行できるようにします。

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 の名前空間と、NSX 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。3 つの vSphere Zones にデプロイされた スーパーバイザー 上に構成した名前空間では、1 つのゾーンの スーパーバイザー 上の名前空間とは異なる機能セットがサポートされます。

- NSX を使用して構成された 1 ゾーンの スーパーバイザー。そのような スーパーバイザー 上の vSphere 名前空間 では、vSphere ポッド、仮想マシン、Tanzu Kubernetes Grid クラスタ、スーパーバイザー サービスがサポートされます。これらの vSphere 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 によって提供されます。

- NSX を使用して構成された 3 ゾーンの スーパーバイザー。NSX を使用して構成された 3 ゾーン スーパーバイザー 上の vSphere 名前空間 では、Tanzu Kubernetes Grid クラスタおよび仮想マシンのみサポートされます。vSphere ポッド や スーパーバイザー サービス はサポートされません。
- VDS を使用して構成された 1 ゾーンの スーパーバイザー。VDS を使用した 1 ゾーン スーパーバイザー 上の vSphere 名前空間 では、Tanzu Kubernetes Grid、仮想マシン、スーパーバイザー サービス がサポートされます。スーパーバイザー サービス が独自の用途でデプロイするものを除き、vSphere ポッド はサポートされません。
- VDS を使用して構成された 3 ゾーンの スーパーバイザー。VDS を使用した 3 ゾーン スーパーバイザー を実行する vSphere 名前空間 では、Tanzu Kubernetes Grid クラスタおよび仮想マシンのみサポートされます。vSphere ポッド や スーパーバイザー サービス はサポートされません。

詳細については、『vSphere IaaS 制御プレーンの概念と計画』の「Requirements for Enabling a Three-Zone Supervisor with HA Proxy Load Balancer」および「Requirements for Enabling a Single-Cluster Supervisor with VDS Networking and HAProxy Load Balancer」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、『vSphere IaaS control plane での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。

スーパーバイザー に NSX を使用する場合は、vSphere 名前空間 レベルでネットワーク設定をオーバーライドするオプションがあります。このオプションを選択する場合は、以下の点に注意してください。

表 3-1. vSphere 名前空間 ネットワーク プランニングに関する考慮事項

検討事項	説明
NSX インストール	特定の vSphere 名前空間 の スーパーバイザー ネットワーク設定をオーバーライドするには、NSX に、Tier-0 ゲートウェイ（ルーター）専用の Edge クラスタと、Tier-1 ゲートウェイ専用の別の Edge クラスタを含める必要があります。ガイド「vSphere IaaS 制御プレーンのインストールと構成」に記載されている NSX のインストール手順を参照してください。
必要な IP アドレス管理	特定の vSphere 名前空間 の スーパーバイザー ネットワーク設定をオーバーライドする場合、新しい vSphere 名前空間 ネットワークでは、スーパーバイザー および他の vSphere 名前空間 ネットワークについて一意の Ingress サブネット、Egress サブネット、および名前空間ネットワーク サブネットを指定する必要があります。設定に応じて、IP アドレスの割り当てを管理する必要があります。
スーパーバイザー ルーティング	スーパーバイザー は、TKG クラスタ ノードと Ingress サブネットに直接ルーティングできる必要があります。vSphere 名前空間 に Tier-0 ゲートウェイを選択する場合、必要なルーティングを構成する方法は 2 つあります。 ■ Virtual Routing and Forwarding (VRF) ゲートウェイを使用して、スーパーバイザー Tier-0 ゲートウェイから構成を継承する ■ Border Gateway Protocol (BGP) を使用して、スーパーバイザー Tier-0 ゲートウェイと専用 Tier-0 ゲートウェイの間にルートを構成する これらのオプションの詳細については、NSX Tier-0 ゲートウェイのドキュメントを参照してください。

前提条件

- スーパーバイザー をデプロイします。

- vSphere 名前空間 にアクセスする必要がある DevOps エンジニアおよび開発者のユーザーとグループを作成します。vCenter Single Sign-On に接続されている ID ソース、または スーパーバイザー を使用して構成された OIDC プロバイダに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。名前空間が 3 ゾーン スーパーバイザー にある場合は、トポロジ対応のポリシーを使用します。トポロジに対応していないストレージ ポリシーを 3 ゾーン名前空間に割り当てることはできません。
- 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。
- 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. vSphere 名前空間 を置く スーパーバイザー 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、vSphere 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 スーパーバイザー の NSX 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 スーパーバイザー 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
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。 **注：** スーパーバイザー は、TKG クラスタ ノードと Ingress サブネットに直接ルーティングできる必要があります。 ■ 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、Ingress、および Egress 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 Ingress 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。
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 Grid 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。Egress 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。
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。
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、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	vSphere ポッド または Tanzu Kubernetes Grid 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 Ingress 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	SNAT IP アドレスの Egress 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。

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 に作成されます。

9 名前空間にアクセスできるユーザーの権限を設定します。

vSphere 名前空間 に対する権限は、名前空間にアクセスする必要がある開発者および DevOps エンジニアのために、vSphere 管理者が設定します。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 上のすべての名前空間にアクセスできます。

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。

ロール	説明
表示可能	ユーザーまたはグループの読み取り専用アクセス。ユーザーまたはグループは、スーパーバイザー 制御プレーンにログインして、vSphere 名前空間 で実行されているワークロード（vSphere ポッド、Tanzu Kubernetes Grid クラスタ、仮想マシンなど）を一覧表示できます。
編集可能	ユーザーまたはグループは、vSphere ポッド、Tanzu Kubernetes Grid クラスタ、仮想マシンを作成、読み取り、更新、および削除できます。管理者グループに属するユーザーには、スーパーバイザー 内のすべての名前空間に対する編集権限があります。
所有者	所有者権限を持つユーザー アカウントでは、次のことができます。 ■ 名前空間でワークロードをデプロイおよび管理する。 ■ 名前空間を他のユーザーまたはグループと共有する。 ■ kubectl を使用して追加の vSphere 名前空間 を作成および削除する。所有者権限を持つユーザーは名前空間を共有する場合、表示、編集、または所有者権限を他のユーザーまたはグループに割り当てることができます。 **注：** 所有者ロールは、vCenter Single Sign-On ID ソースで使用可能なユーザーに対してサポートされます。外部 ID プロバイダからのユーザーまたはグループで所有者ロールを使用することはできません。

ユーザーまたはグループを**表示可能**ロールまたは**編集可能**ロールに割り当てると、システムでは RoleBinding を作成して ClusterRole にマッピングします。たとえば、**編集可能**ロールに割り当てられたユーザーまたはグループは、RoleBinding を使用して Kubernetes `edit` ClusterRole にマッピングされます。`edit` ロールにより、ユーザーはクラスタのプロビジョニングと運用を行うことができます。このマッピングを表示するには、ターゲットの vSphere 名前空間 から `kubectl get rolebinding` コマンドを使用します。

```
kubectl get rolebinding -n tkg2-cluster-namespace
NAME
ROLE                          AGE
wcp:tkg-cluster-namespace:group:vsphere.local:administrators   ClusterRole/
edit             33d
wcp:tkg-cluster-namespace:user:vsphere.local:administrator     ClusterRole/
edit             33d
```

ユーザーまたはグループに所有者ロールを割り当てると、システムでは ClusterRoleBinding を作成して ClusterRole にマッピングします。これにより、そのユーザーまたはグループが kubectl を使用して vSphere 名前空間 を作成および削除できます。このマッピングを表示するには、スーパーバイザー 制御プレーン ノードに SSH 接続します。

10 名前空間にストレージを割り当てます。

名前空間にストレージ ポリシーを割り当てると、DevOps チームはパーシステント ストレージを使用できるようになります。

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 IaaS control plane 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。Tanzu Kubernetes Grid を使用する場合、ストレージ クラスは名前空間から Tanzu Kubernetes Grid 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、『6 章 vSphere IaaS control plane での仮想マシンのデプロイと管理』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 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。「vSphere IaaS 制御プレーン クラスタへの接続」を参照してください。

注： この『vSphere IaaS 制御プレーンのサービスとワークロード』ガイドには、Tanzu Kubernetes Grid クラスタで実行されているワークロードの情報は含まれていません。Tanzu Kubernetes Grid クラスタを使用する方法については、「Using Tanzu Kubernetes Grid on Supervisor with vSphere IaaS Control Plane」を参照してください。

スーパーバイザー からの vSphere 名前空間 の削除

スーパーバイザー から vSphere 名前空間 を削除できます。

前提条件

- 仮想マシン、vSphere ポッド、TKG クラスタなど、デプロイされたすべてのワークロードを削除します。TKG クラスタの削除の詳細については、kubectl または Tanzu CLI を使用した TKG 2.0 クラスタの削除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] タブをクリックします。

3 スーパーバイザー で使用可能な名前空間のリストから、削除する vSphere 名前空間 を選択します。

4 [削除] をクリックします。

vSphere 名前空間 が削除されます。プロセスが完了するまでに時間がかかる場合があります。

vSphere 名前空間 へのリソース制限の設定

vSphere 管理者は、vSphere 名前空間 でリソース制限とコンテナのデフォルトを設定できます。DevOps エンジニアは、後でポッド仕様のコンテナのデフォルトをオーバーライドできますが、vSphere 管理者が名前空間に設定したリソース制限の合計を超えることはできません。コンテナ要求は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 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 vSphere 名前空間 を選択し、[構成] を選択して、[リソースの制限] をクリックします。

3 [編集] をクリックします。

Tanzu Kubernetes Grid 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。『vSphere IaaS 制御プレーンでの TKG サービスの使用』の Tanzu Kubernetes クラスタでの仮想マシン クラスを参照してください。

オプション	説明
CPU	vSphere 名前空間 の CPU 使用量に制限を設定します。
メモリ	vSphere 名前空間 のメモリ使用量に制限を設定します。

オプション	説明
ストレージ	使用されるストレージ ポリシーごとに、vSphere 名前空間 のストレージ使用量に制限を設定します。
コンテナのデフォルト	vSphere 名前空間 で CPU 制限、CPU 要求、メモリ要求、コンテナのメモリ制限のデフォルト値を設定します。

vSphere 名前空間 でのオブジェクト制限の構成

デプロイ、ジョブ、デーモン セット、ステートフル セットの数など、vSphere 名前空間 で実行されているオブジェクトに対する制限を構成できます。オブジェクトに対して構成される制限は、アプリケーションの仕様や、オブジェクトが vSphere 名前空間 内のリソースをどのように使用するかの設定方法によって異なります。

前提条件

- スーパーバイザー 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 vSphere 名前空間 を選択します。
3. [構成] を選択し、[オブジェクト制限] を選択します。
4. [編集] をクリックします。

オプション	説明
vSphere ポッド	vSphere 名前空間 内で実行できる vSphere ポッド の数。
展開	vSphere 名前空間 内で実行できるデプロイの数。
ジョブ	vSphere 名前空間 内で実行できるジョブの数。
デーモンのセット	vSphere 名前空間 内で実行できるデーモン セットの数。
レプリカのセット	vSphere 名前空間 内のレプリカセットの数。
レプリケーション コントローラ	vSphere 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。
ステートフル セット	vSphere 名前空間 内で実行できる StatefulSet の数。
構成マップ	vSphere 名前空間 内で実行できる ConfigMap の数。
シークレット	vSphere 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	vSphere 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	vSphere 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 名前空間を作成し、構成していること。

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。

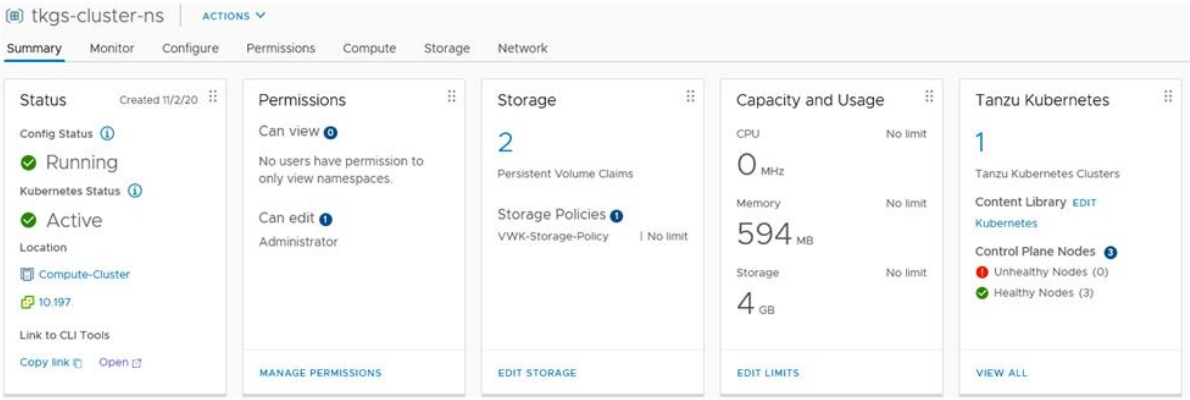
3 [ワークロード管理] が有効な vCenter Server クラスタを選択します。

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 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 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。



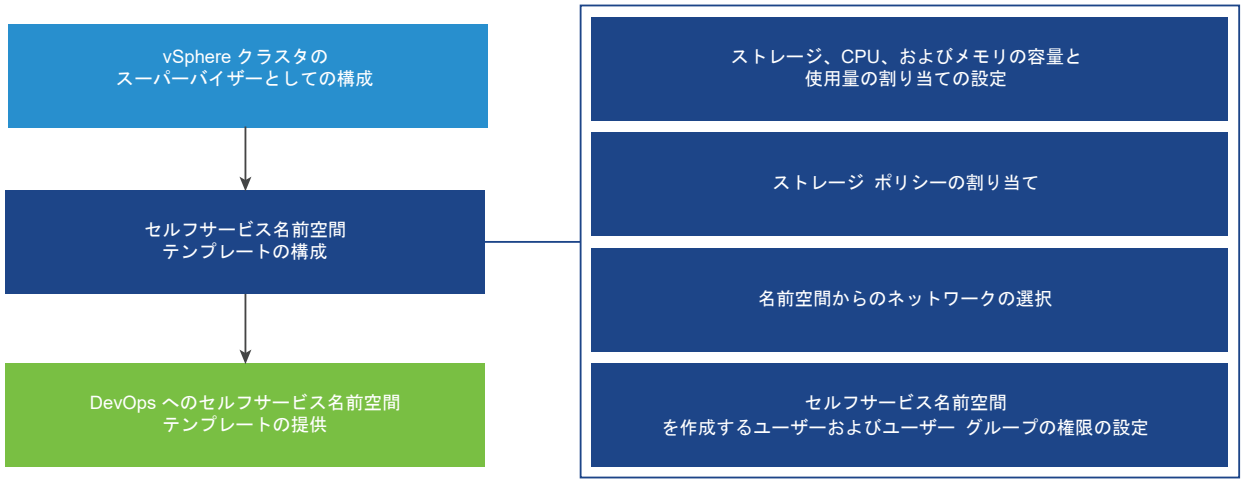
vSphere IaaS control plane でのセルフサービス名前空間テンプレートのプロビジョニング

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

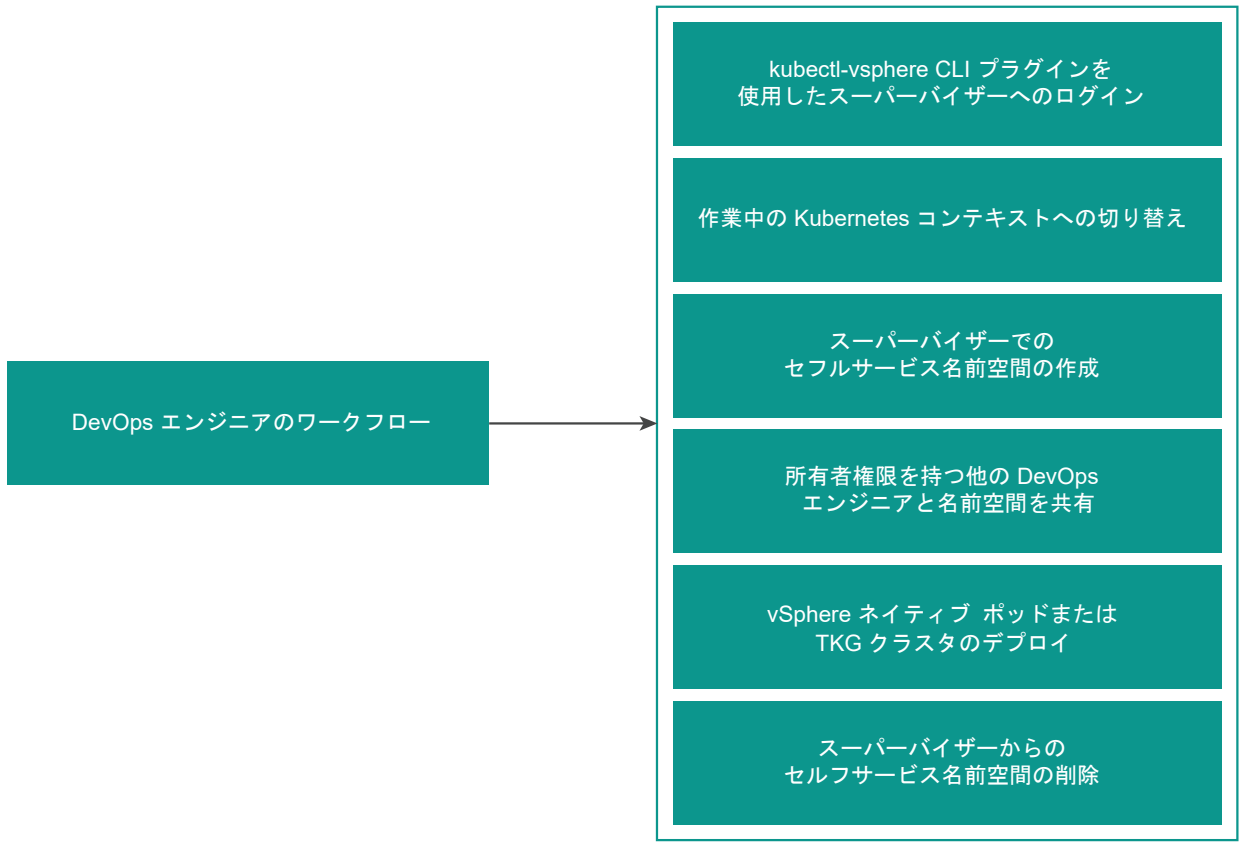
vSphere 管理者は、スーパーバイザー名前空間の作成、名前空間に対する CPU、メモリ、ストレージの制限の設定、権限の割り当て、およびクラスタの名前空間サービスのテンプレートとしてのプロビジョニングと有効化を行うことができます。

図 3-1. 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 3-2. 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 IaaS control plane 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 に移動します。

2 [構成] タブをクリックし、[スーパーバイザー] の下で [全般] を選択します。

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、名前空間のリソースを構成します。

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	パーシステント ストレージを必要とするワークロードで使用するストレージ ポリシー。
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。
仮想マシン クラス	スタンドアローン仮想マシンをデプロイするための仮想マシン クラス。
コンテンツ ライブラリ	仮想マシンのデプロイに使用する仮想マシン イメージを含むコンテンツ ライブラリ。

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 に移動します。

2 [構成] タブをクリックし、[スーパーバイザー] の下で [全般] を選択します。

3 [名前空間サービス] 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 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 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に接続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 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 IaaS control plane 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 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 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 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 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に接続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 名前空間のストレージ設定の変更

スーパーバイザー の名前空間にストレージ ポリシーを割り当てると、DevOps チームはパーシステント ストレージを使用できるようになります。これらのストレージ ポリシーにより、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が制御されます。vSphere 管理者は、通常、名前空間を構成するときにストレージ ポリシーを割り当てます。初期ストレージ ポリシーの割り当てを変更する必要がある場合は、次のタスクを実行します。

前提条件

- VMware vCenter 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。
- 名前空間が 3 ゾーン スーパーバイザー にある場合は、トポロジ対応のポリシーを使用します。トポロジに対応していないストレージ ポリシーを 3 ゾーン名前空間に割り当てることはできません。

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。

NSX vSphere 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 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 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 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 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。

名前空間に対するネットワークとロード バランサのパラメータの構成

vSphere IaaS control plane では、NCP 構成ファイル `ncp.ini` の編集はサポートされていません。NCP で CustomResourceDefinitions (CRD) を作成して、ネットワークとロード バランサのパラメータを構成できます。

[NCPSetting] CRD

[NCPSetting] CRD を作成し、NCP 構成の値を設定します。

次の表に、ネットワークとロード バランサの構成可能なパラメータを示します。

パラメータ	説明
log_dropped_traffic	分散ファイアウォールの拒否ルールがログに記録されるかどうかを示します。 値：True、False デフォルトは false
log_firewall_traffic	DFW ルールがログに記録されるかどうかを示します。 値は次のとおりです。 ■ ALL。すべての DFW ルールのログを有効にします。 ■ DENY。拒否ルールのログのみを有効にします。
pool_algorithm	ロード バランサ プール オブジェクトでロード バランシング アルゴリズムを設定するオプション。 値は次のとおりです。 ■ Round_Robin ■ Weighted_Round_Robin ■ Least_Connection ■ Weighted_Least_Connection ■ IP-Hash デフォルトは Round-Robin です。
service_size	ロード バランサのサイズを設定するオプション。 値は、Small、Medium、Large です。 デフォルトは Small です。
l7_persistence	ロード バランサのパーシステンス オプションを設定するオプション。 値は次のとおりです。 ■ cookie。 ■ source_ip
l7_persistence_timeout	L7 パーシステンス プロファイルのパーシステンス タイムアウト値（秒単位）。
cookie_name	l7_persistence type が cookie に設定されている場合は、Cookie 名を指定します。
x_forward_for	入力方向でヘッダーの X_forward_for を有効にします。 値は次のとおりです。 ■ Insert ■ Replace
snat_rule_logging	SNAT ルールのログ作成を選択するオプション。 値は次のとおりです。 ■ None ■ Basic。すべての名前空間のログ作成。 ■ Extended。すべての名前空間とサービスのログ作成。

パラメータ	説明
vs_access_log	入力方向とルートに関する仮想サーバのログ プロパティ。 値は次のとおりです。 ■ VS_access_log_none ■ access_log_enabled。レイヤー 7 仮想サーバのログを有効にします。 ■ log_significant_event_only。HTTP 応答状態が > = 400 の要求は、重要なイベントとして扱われます。 デフォルトは VS_access_log_none です。
ip_reallocation_time	解放された IP アドレスが再割り当てされるまでの時間（秒単位）。

NCP および NSX オブジェクトの詳細については、NSX のドキュメントを参照してください。

この機能を有効にするには、次の操作を行います。

1 [enable_ncp_setting_crd] を [True] に設定します。

2 次のテンプレートを使用して、YAML ファイルを作成します。

```
apiVersion: apiextensions.k8s.io/v1
kind: CustomResourceDefinition
metadata:
  name: ncpsettings.vmware.com
spec:
  group: vmware.com
  versions:
    - name: v1
      served: true
      storage: true
      schema:
        openAPIV3Schema:
          type: object
            properties:
              spec:
                type: object
                properties:
                  nsx_v3:
                    type: object
                    properties:
                      log_dropped_traffic:
                        description: 'Indicates whether distributed firewall DENY rules
are logged.'
                        type: boolean
                      log_firewall_traffic:
                        description: 'Indicate whether DFW rules are logged.'
........ All configs that are allow to be configured via CRD........

  scope: Cluster
  names:
    plural: ncpsettings
```

```
    singular: ncpsetting
    kind: NCPSetting
    shortNames:
    - ncpstg
```

例：

```
apiVersion: vmware.com/v1alpha2
kind: NCPSetting
metadata:
    name: ncp-setting-crd
spec:
    nsx_v3:
        log_dropped_traffic: True
        log_firewall_traffic: ALL
        pool_algorithm: Round_Robin
        l7_persistence: cookie
        x_forwarded_for: Insert
```

3 次のコマンドを使用して、YAML ファイルを適用します。

```
kubectl apply -f ncp-setting-crd.yaml.j2
```

複数の CRD で同じ構成値がオーバーライドされないように、NCP は名前が [ncp-setting-crd] の CRD オブジェクトのみを処理します。別の名前を持つ他の CRD にはエラーの注釈が付けられ、NCP はそれらの CRD を処理しません。

NCP 構成のオーバーライド

CRD での構成パラメータには、対応する NSX オブジェクトがある場合があります。CRD を作成してパラメータをオーバーライドしても、次の場合を除いて、CRD はオブジェクト内のパラメータを変更しません。

- `l7_persistence`、`l7_persistence_timeout`、`cookie_name`。`l7_persistence` が CRD によって変更された場合、NCP は、`l7_persistence`、`l7_persistence_timeout`、`cookie_name` の値を含む新しいパーシステンス プロファイルを作成します。

  CRD を通じて `l7_persistence_timeout` と `cookie_name` が変更されると、既存のプロファイルは新しい値に基づいて更新されます。

- `x_forwarded_for`。`x_forwarded_for` が CRD によって変更された場合、NCP はその値に基づいて新しいアプリケーション プロファイルを作成します。

- `vs_access_log`。`vs_access_log` が CRD によって変更された場合、NCP はそれに応じて仮想サーバのログ オプションを更新します。

vSphere IaaS control plane を使用した スーパーバイザー サービス の管理

4

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 IaaS control plane 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、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 にインストールされている場合、DevOps エンジニアはそれらをさまざまな方法で使用できます。

- Harbor などの共有 スーパーバイザー サービス は、TKG クラスタ、vSphere ポッド、または仮想マシンで実行されているワークロードに機能を直接提供します。
- MinIO などのオペレータを含む スーパーバイザー サービス には通常、API またはグラフィカル インターフェイスが用意されています。DevOps エンジニアはこれらを使用して、CRD を介して vSphere 名前空間 でサービスのインスタンスを作成および管理することができます。たとえば、MinIO バケットを作成するには、CRD を使用して vSphere 名前空間 でバケットを作成します。

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。

スーパーバイザー サービス でサポートされている スーパーバイザー のデプロイ

スーパーバイザー サービス は vSphere ポッド としてデプロイされます。vSphere 8.0 リリースでは、NSX ネットワーク スタックで構成された スーパーバイザー のみが vSphere ポッド とそれぞれの スーパーバイザー サービス をサポートします。vSphere 8 Update 1 リリース以降では、NSX または VDS の両方のタイプのネットワークを使用してデプロイされた スーパーバイザー で、スーパーバイザー サービス によってデプロイされた vSphere ポッド がサポートされます。

注： スーパーバイザー に VDS ネットワーク スタックが構成されている場合、NSX がバッキングするネットワーク（NSX によって作成された分散ポート グループ）で スーパーバイザー サービス を実行することはできません。

次の表に、vSphere 8 以降の既存の スーパーバイザー のデプロイにおける、スーパーバイザー サービス によってデプロイされた vSphere ポッド のサポートを示します。

vSphere バージョン	NSX ネットワーク	VDS ネットワーク	スーパーバイザー のバージョン	1 ゾーン スーパーバイザー	3 ゾーン スーパーバイザー
vSphere 8	はい	いいえ	1.23 以降	はい	いいえ
vSphere 8.0.1 以降	はい	はい	1.24 以降	はい	いいえ
vSphere 8.0.3 以降	はい	はい	1.28 以降	はい	はい

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 サービス は vSphere Client から管理します。スーパーバイザー への スーパーバイザー サービス のインストール、バージョンのアップグレード、または スーパーバイザー 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行うことができます。スーパーバイザー サービス の複数のバージョンを vCenter Server に登録できますが、スーパーバイザー に一度にインストールできるバージョンは 1 つのみです。

表 4-1. 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 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 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 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 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 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 のライフ サイクルの管理には、次の操作が含まれます。

操作	説明
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 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 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 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。

操作	説明
スーパーバイザー への スーパーバイザー サービス のインストール	スーパーバイザー サービス を スーパーバイザー にインストールすると、サービス YAML ファイルが スーパーバイザー に適用されて、すべての vSphere 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 にインストールする スーパーバイザー サービス ごとに、vSphere 名前空間 が自動的に作成されます。その vSphere 名前空間 からサービス リソースを管理できます。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 のアップグレード	スーパーバイザー 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 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 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 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。
特定のバージョンの スーパーバイザー サービス 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 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、すべてのバージョンを非アクティブにしてから、これらのバージョンを スーパーバイザー からアンインストールし、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。

スーパーバイザー サービス のコア機能

コア スーパーバイザー サービス は、スーパーバイザー の有効化の際にオペレータが vSphere IaaS control plane に事前にインストールされているサービスです。スーパーバイザー にコア スーパーバイザー サービス をインストールして、そのバージョンをアップグレードできます。最初に スーパーバイザー を更新する必要はありません。ただし、コア スーパーバイザー サービス のオペレータを vSphere IaaS control plane から削除することはできません。

コア スーパーバイザー サービス の例として、TKG サービスと Velero vSphere Operator サービスがあります。

次のトピックを参照してください。

- 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 への スーパーバイザー サービス のインストール
- スーパーバイザー 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 への新しいバージョンの追加

- スーパーバイザー サービス の新しいバージョンへのアップグレード
- スーパーバイザー 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 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

vSphere IaaS control plane 環境が稼動している vCenter Server システムに スーパーバイザー サービス を追加する方法を確認してください。vCenter Server にサービスを追加した後、スーパーバイザー 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。

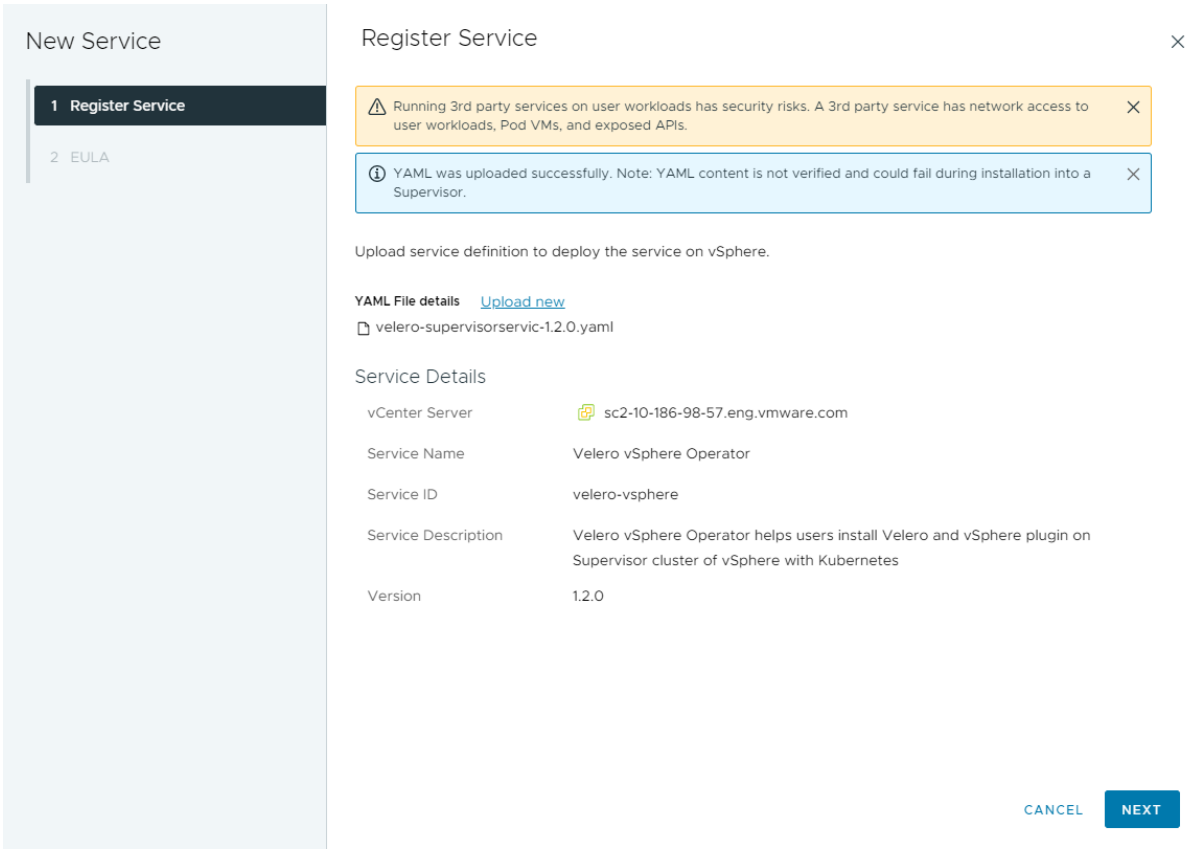
前提条件

- 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。

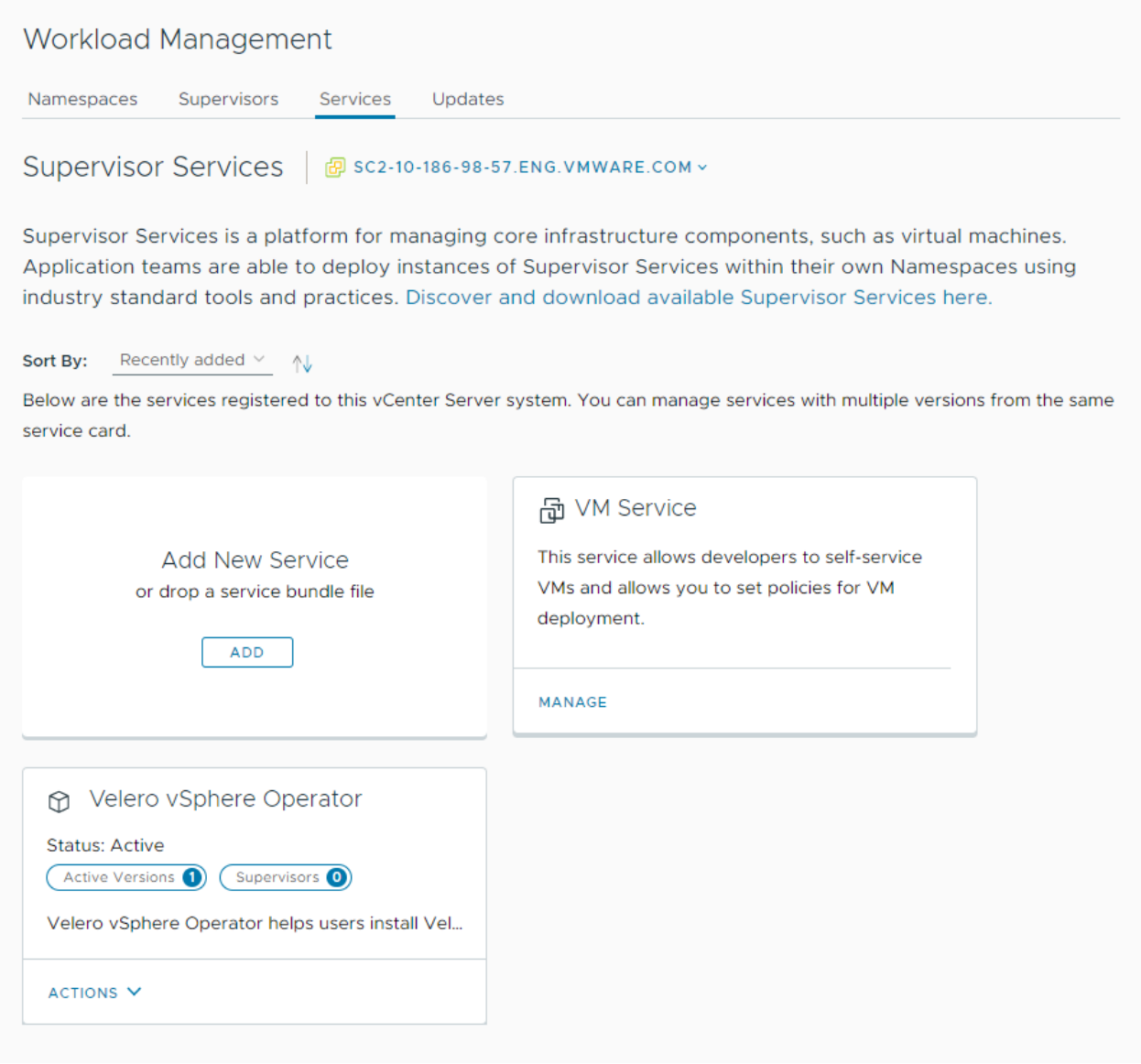


5 [次へ] をクリックします。EULA が表示された場合は、同意します。

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 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 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 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 IaaS control plane 環境内の スーパーバイザー 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 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 では、一度に 1 つのバージョンの スーパーバイザー サービス 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。
- サービスをインストールする スーパーバイザー で、スーパーバイザーでのスーパーバイザー 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。5 章 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 のカードで、[アクション] - [サービスの管理] の順に選択します。

4 [インストール バージョン] ドロップダウン メニューから、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。

5 サービスをインストールする スーパーバイザー を選択します。

6 [次へ] をクリックします。

互換性の事前チェックは、インストールする スーパーバイザー サービス のバージョンが スーパーバイザー と互換性があるかどうかを判断するために実行されます。サービスのバージョンがスーパーバイザーと互換性がある場合は、インストールを続行できます。選択したサービスのバージョンが スーパーバイザー と互換性がない場合は、完全に互換性がないことを示す 2 種類のメッセージが表示されます。

- 警告メッセージ：警告メッセージはスキップできますが、インストールを続行するには確認する必要があります。
- エラー メッセージ：エラー メッセージは、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 と互換性がなく、インストールできないことを示します。エラー メッセージが表示された場合は、特定の スーパーバイザー にサービスをインストールする前に、検出された非互換性を解決する必要があります。

7 サービスに構成プロパティが必要な場合は、[YAML サービス構成] フィールドに入力します。

8 スーパーバイザー でサービスのインストールの進行状況を確認します。

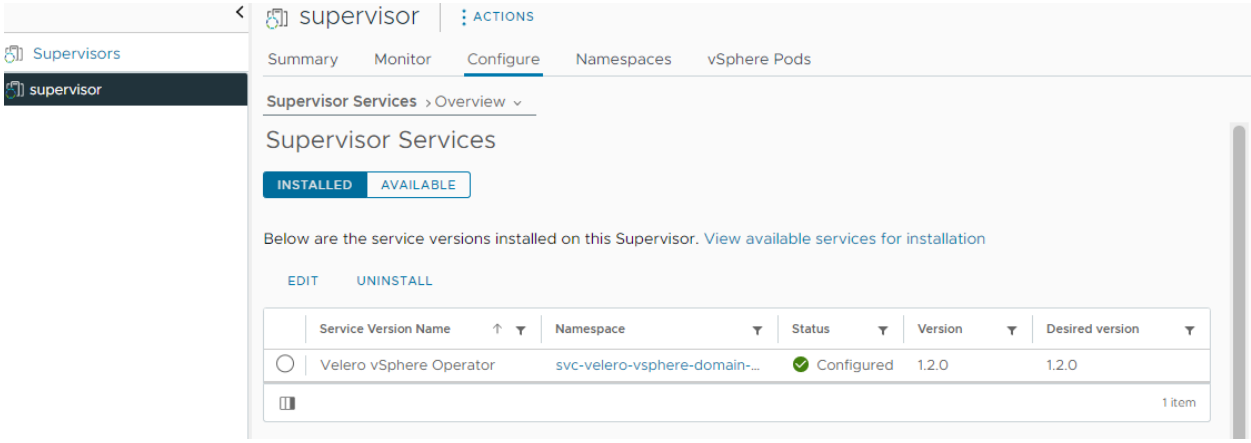
a [スーパーバイザー] タブを選択し、サービスをインストールするスーパーバイザーを選択します。

b [構成] をクリックし、[スーパーバイザー サービス] - [概要] の順に選択します。

c [インストール] タブを選択します。

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 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 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 エンジニアはワークロードで使用できます。



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 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 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 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 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 インベントリで、スーパーバイザー に変換したホスト クラスタに移動します。

2 [構成] タブをクリックして、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 IaaS control plane 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 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 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 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 サービス の新しいバージョンへのアップグレード

新しいバージョンの スーパーバイザー サービス を vCenter Server に追加したら、そのバージョンを スーパーバイザー に対してインストールできます。インストールできるのは、アクティブかつインストール先の スーパーバイザー と互換性のある スーパーバイザー サービス のバージョンのみです。新しいバージョンの スーパーバイザー サービス がターゲット スーパーバイザー と互換性があることを確認するために、互換性の事前チェックが実行されます。

前提条件

- vCenter Server に新しいバージョンの スーパーバイザー サービス を追加します。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。
- サービスをインストールする スーパーバイザー で、スーパーバイザーでのスーパーバイザー サービスの管理権限があることを確認します。

手順

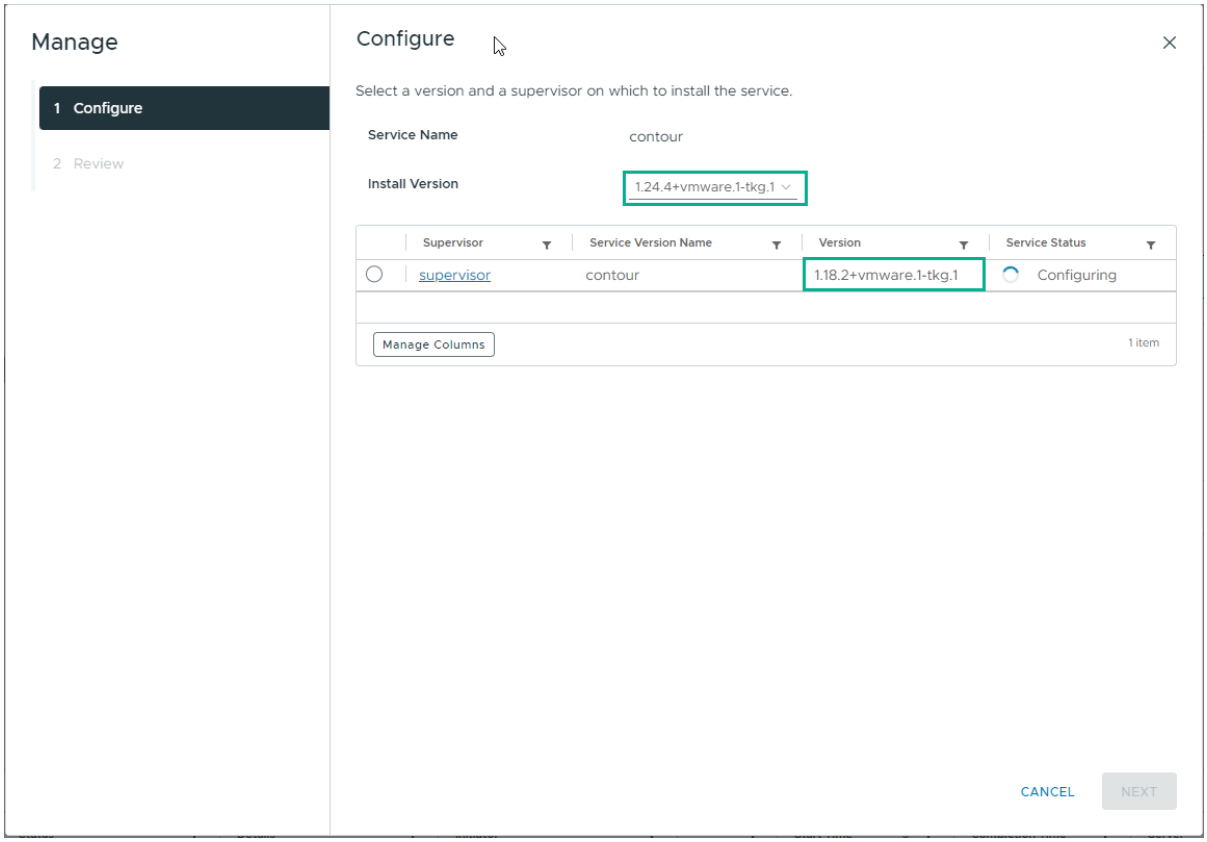
1 vSphere Client ホーム メニューから、[ワークロード管理] を選択します。

2 [[サービス]] を選択します。

3 [サービスの管理] を選択します。

4 インストールする新しいバージョンを選択し、インストール先の スーパーバイザー を選択します。

スーパーバイザー に現在インストールされているサービス バージョンが古いバージョンであることを確認します。



5 [次へ] をクリックします。

互換性の事前チェックは、インストールする スーパーバイザー サービス のバージョンが スーパーバイザー と互換性があるかどうかを判断するために実行されます。サービスのバージョンがスーパーバイザーと互換性がある場合は、インストールを続行できます。選択したサービスのバージョンが スーパーバイザー と互換性がない場合は、完全に互換性がないことを示す 2 種類のメッセージが表示されます。

- 警告メッセージ：警告メッセージはスキップできますが、インストールを続行するには確認する必要があります。
- エラー メッセージ：エラー メッセージは、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 と互換性がなく、インストールできないことを示します。エラー メッセージが表示された場合は、特定の スーパーバイザー にサービスをインストールする前に、検出された非互換性を解決する必要があります。

6 サービスに構成プロパティが必要な場合は、[YAML サービス構成] フィールドに入力します。

7 スーパーバイザー でサービスのインストールの進行状況を確認します。

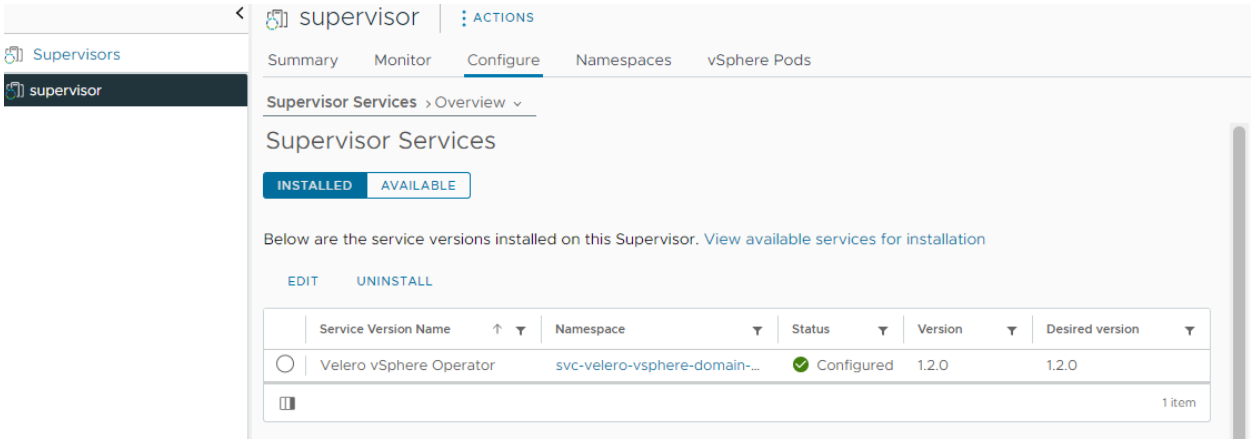
a [スーパーバイザー] タブを選択し、サービスをインストールするスーパーバイザーを選択します。

b [構成] をクリックし、[スーパーバイザー サービス] - [概要] の順に選択します。

c [インストール] タブを選択します。

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 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 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 エンジニアはワークロードで使用できます。



スーパーバイザー 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 IaaS control plane 環境の スーパーバイザー 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 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

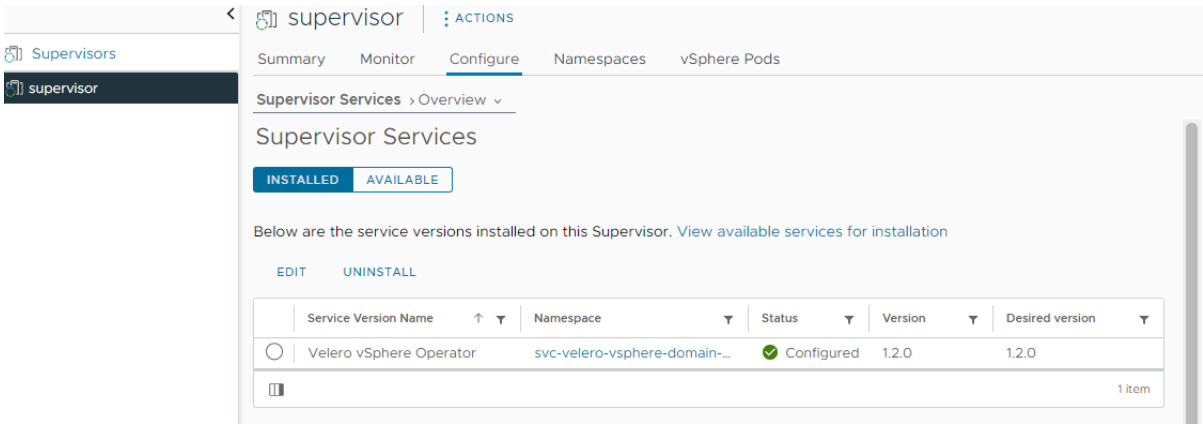
- vCenter Server に スーパーバイザー サービス を追加します。vCenter Server への スーパーバイザー サービス の追加を参照してください。
- スーパーバイザー に スーパーバイザー サービス をインストールします。スーパーバイザー への スーパーバイザー サービス のインストールを参照してください。

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] タブをクリックし、リストから スーパーバイザー を選択します。

3 [構成] タブをクリックし、[スーパーバイザー サービス] の [概要] 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 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 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 IaaS control plane 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 では引き続き実行されますが、他の スーパーバイザー 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 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。

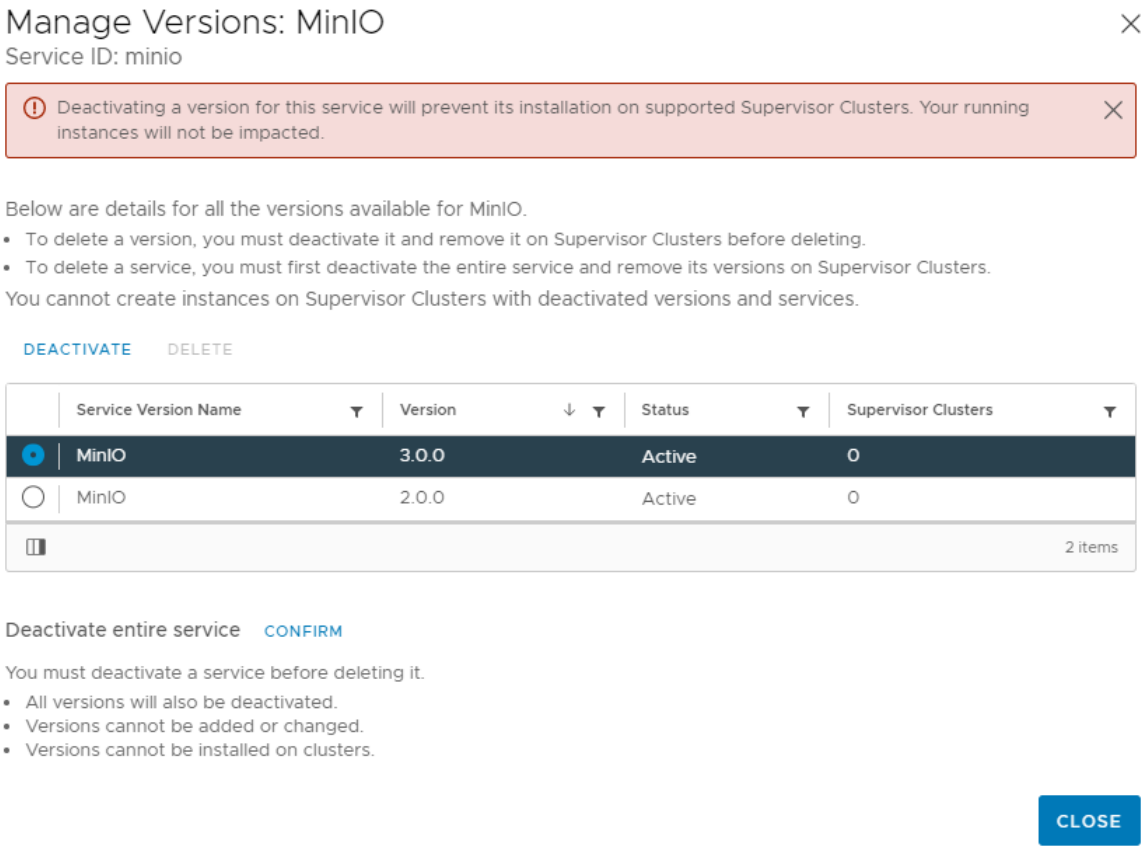
手順

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 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 IaaS control plane 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 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 IaaS control plane 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 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 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] タブをクリックし、リストから スーパーバイザー を選択します。

3 [構成] タブをクリックし、[スーパーバイザー サービス] の [概要] をクリックします。

4 [インストール済み] で、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 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 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 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 IaaS control plane 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 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 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 IaaS control plane 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。

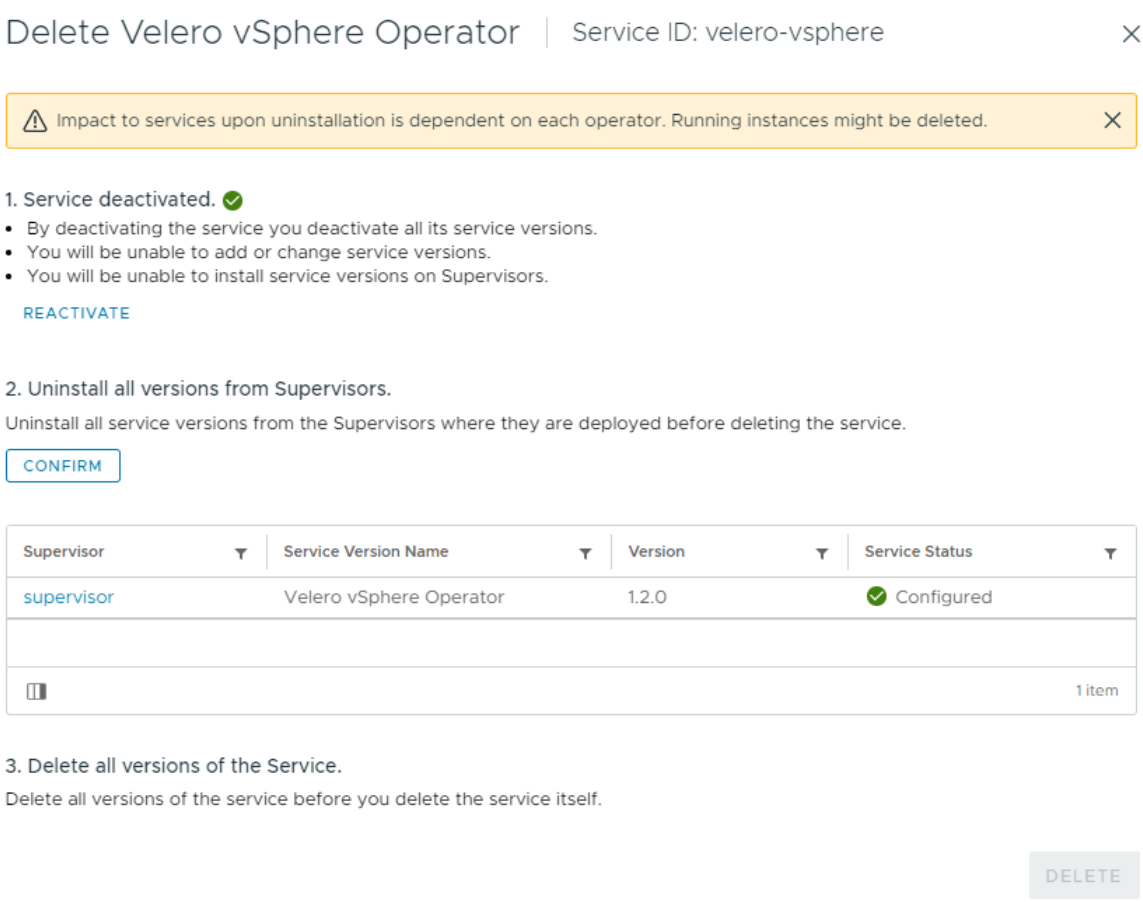
2 [[サービス]] を選択します。

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 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 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。

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

5

vSphere IaaS control plane では、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できます。このプラットフォームが提供するフレームワークを使用すると、サードパーティは、基盤となる vSphere インフラストラクチャとサービス アプリケーションを統合できます。

vSAN データ パーシステンス プラットフォームについて

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 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。

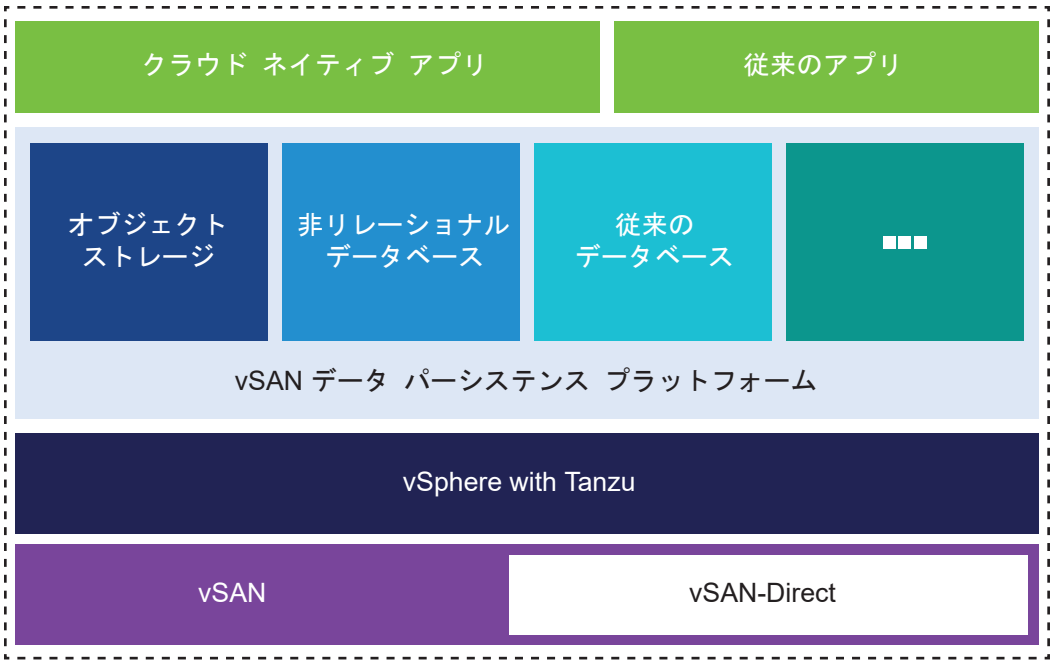
プラットフォームは、次のタイプのサービスをサポートしています。

- 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。

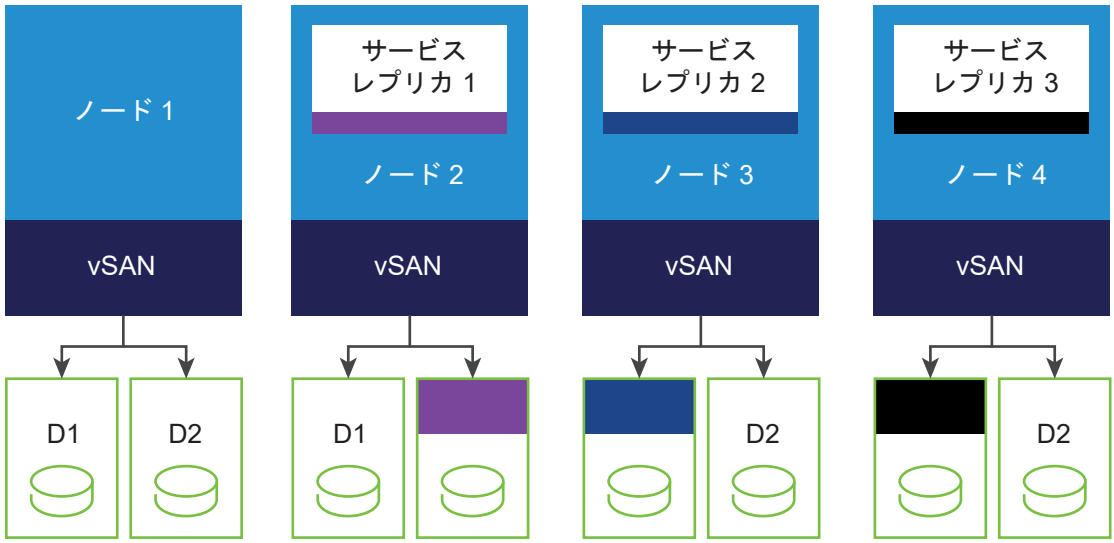
ポッドなどのコンピューティング インスタンスは、最初は、vSAN クラスタ内のいずれかのノードに配置されます。その後、vSAN SNA ポリシーによって作成された vSAN オブジェクトのすべてのデータは、ポッドが実行されているノードに自動的に配置されます。

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

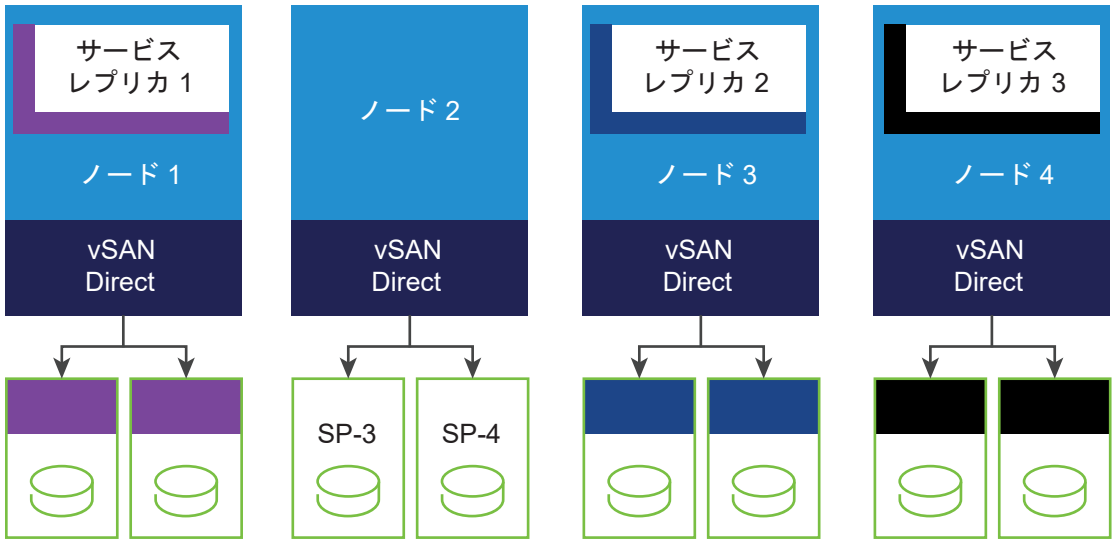


vSAN Direct

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

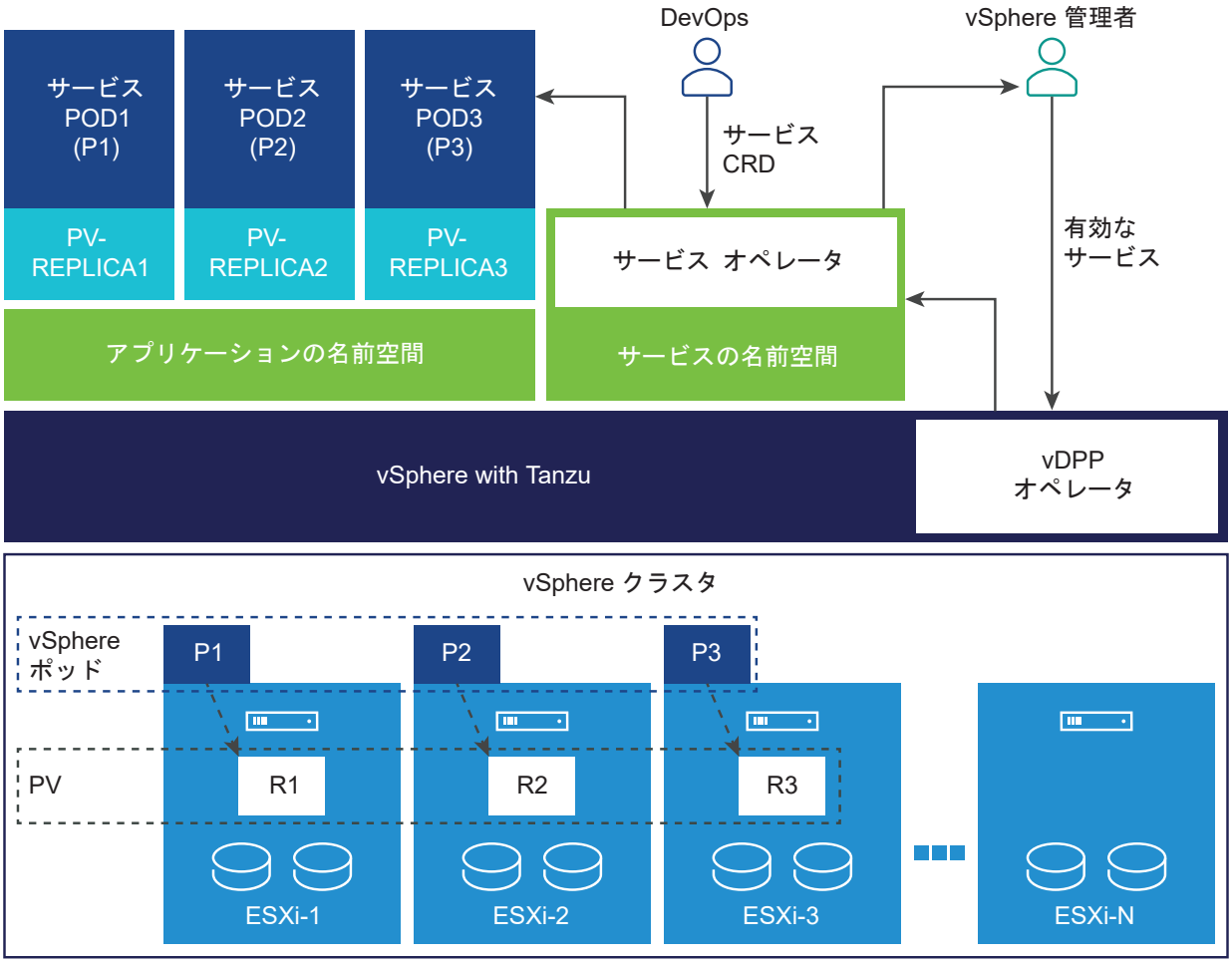
使用する vSAN のタイプを決定する場合は、次の一般的な推奨事項に従ってください。

- 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 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 IaaS control plane 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

次のトピックを参照してください。

- vSphere IaaS control plane でのステートフル サービスの有効化

- ステートフル サービス用の vSAN Direct データストアの設定
- vSphere IaaS control plane でのステートフル サービスの監視
- 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
- vSAN データ パーシステンス プラットフォームのカスタム ストレージ ポリシーの作成

vSphere IaaS control plane でのステートフル サービスの有効化

vSphere IaaS control plane 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。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 にインストールします。

前提条件

必要な権限：Supervisor Services.Manage Supervisor Services

1 パーシステント ストレージの構成

vSAN データ パーシステンス プラットフォームでは、ステートフル サービスは次の 2 つのモードで vSAN ストレージを使用できます。

- vSAN Direct。vSAN Direct を設定するには、vSAN Direct データストアの作成を参照してください。

  注： vSAN Direct データストア内のディスクでは、ボリューム割り当てタイプの変更はサポートされません。vSAN Direct データストア内のディスクのボリューム割り当てタイプを選択後に変更することはできません。ただし、クローン作成や再配置などの操作の実行中は、新しいディスクのボリューム割り当てタイプの変更が許可されます。

- SNA ストレージ ポリシーを使用する通常の vSAN。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。

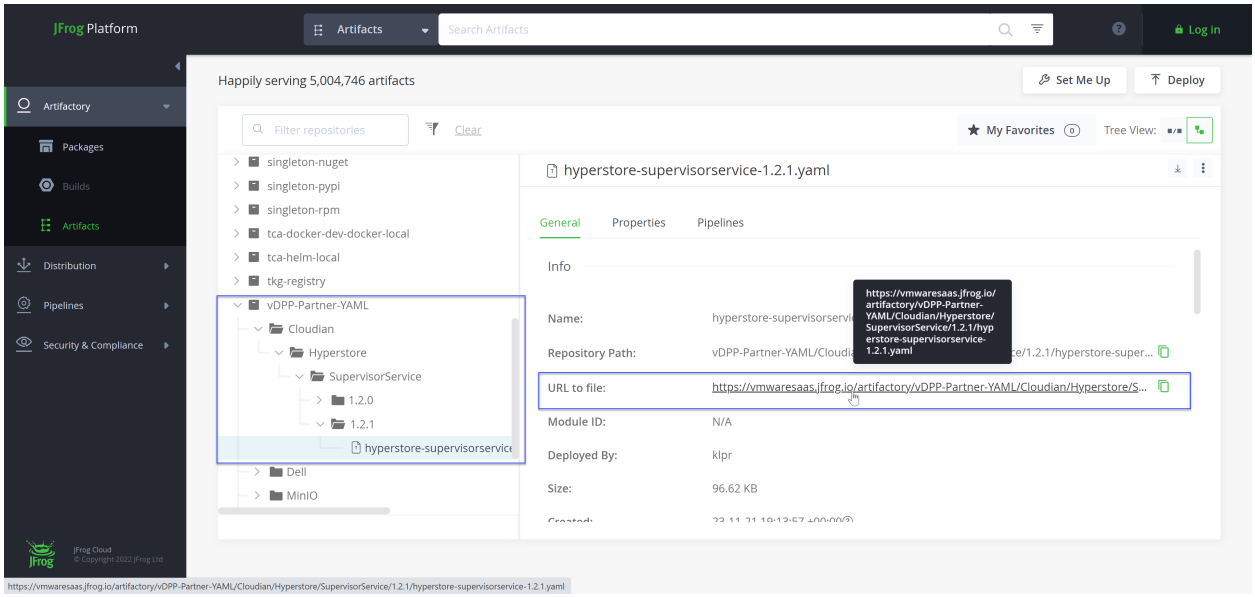
2 サービス YAML ファイルのダウンロード

サービスの YAML ファイルを VMware で保守されているリポジトリから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 環境をアップグレードした後で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。

1 https://vmwaresaas.jfrog.io/ リポジトリで、[Artifacts] - [vDPP-Partner-YAML] の適切なパートナー フォルダに移動します。

2 ファイルの URL をクリックし、YAML ファイルをダウンロードします。



3 vCenter Server へのサービスの追加

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。

4 スーパーバイザー へのサービスのインストール

スーパーバイザー 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 内にこのサービスの名前空間を作成します。
- デフォルトのストレージ ポリシーと対応するストレージ クラスを作成し、名前空間に割り当てます。

  ポリシーは、vSAN Shared-Nothing-Architecture (SNA) および vSAN Direct データストア用です。

  注： vSphere 管理者が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 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 Grid 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、カスタム 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 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。

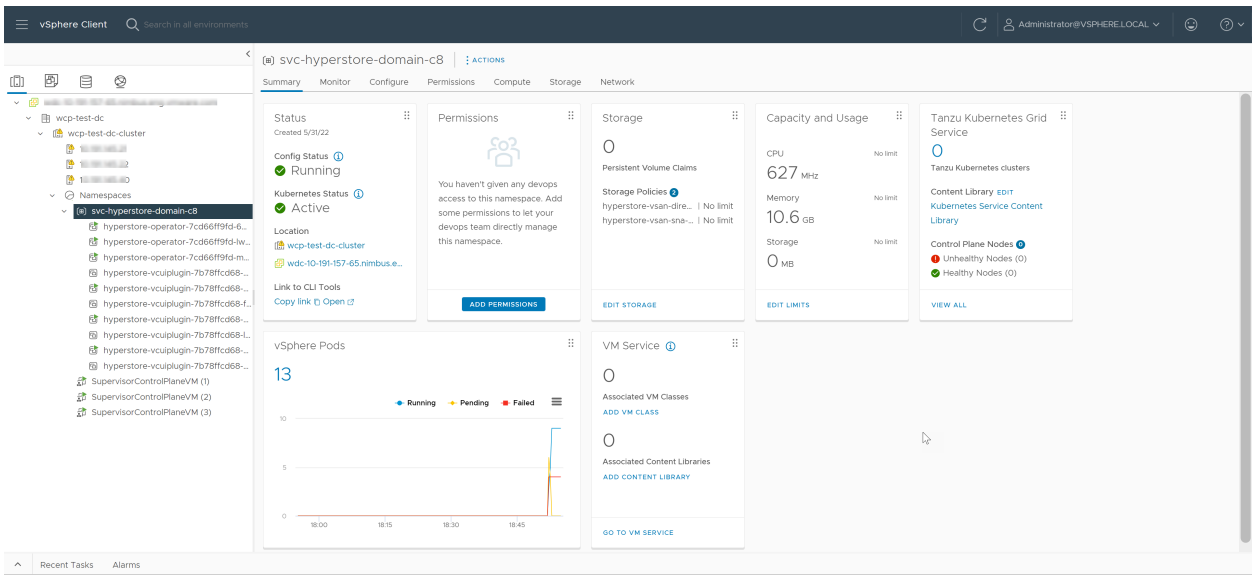
- サードパーティからカスタム ユーザー インターフェイス プラグインが提供されている場合は、そのプラグインが vSphere Client に表示されます。vSphere 管理者は、プラグインを使用してサービスを管理できます。

5 サービス用に作成されたリソースの確認

vSphere 管理者は、サービスに適切なすべてのリソースが作成されていることを確認できます。

サービス用に作成された名前空間に移動し、[サマリ] タブをクリックします。

[サマリ] 画面に、名前空間に割り当てられたストレージ ポリシー、名前空間で実行されている vSphere ポッドなどが表示されます。



6 サービスの管理と監視

- 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 IaaS control plane でのステートフル サービスの監視を参照してください。

- デフォルトではなくカスタム ストレージ ポリシーを作成する場合は、vSAN データ パーシステンス プラットフォームのカスタム ストレージ ポリシーの作成を参照してください。

7 サービス使用の開始

DevOps エンジニアは、`kubectl` コマンドを使用してサービス名前空間にアクセスします。

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できます。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 サービス用の vSAN Direct データストアの設定

ステートフル サービス専用のハードウェア クラスタを vSphere IaaS control plane に作成する場合は、vSAN Direct データストアを使用できます。vSAN Direct は、ほぼ raw のデータストアであり、これは ESXi ホストのローカルにある未要求のストレージ デバイスにデプロイします。

vSAN Direct のストレージ デバイスへのタグ付け

vSAN Direct では、vSAN クラスタ内の各 ESXi ホストに、未要求ディスクが必要です。ただし、特定の環境では、vSAN は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。

`esxcli` コマンドを使用して、デバイスを vSAN Direct としてマークします。

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

次の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もできます。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()
```

vSAN Direct データストアの作成

vSphere 管理者は、vSAN データ パーシステンス プラットフォームや仮想マシン インスタンス ストレージなどの機能で使用する vSAN Direct データストアを設定します。データストアを作成するには、ESXi ホストに対してローカルな未要求のストレージ デバイスを使用します。

スーパーバイザー に対して vSAN を有効にするときに、vSAN Direct データストアを作成できます。次のタスクは、クラスタで vSAN がすでに有効になっている場合に、vSAN Direct としてローカル ストレージ デバイスを要求する方法を示しています。

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。

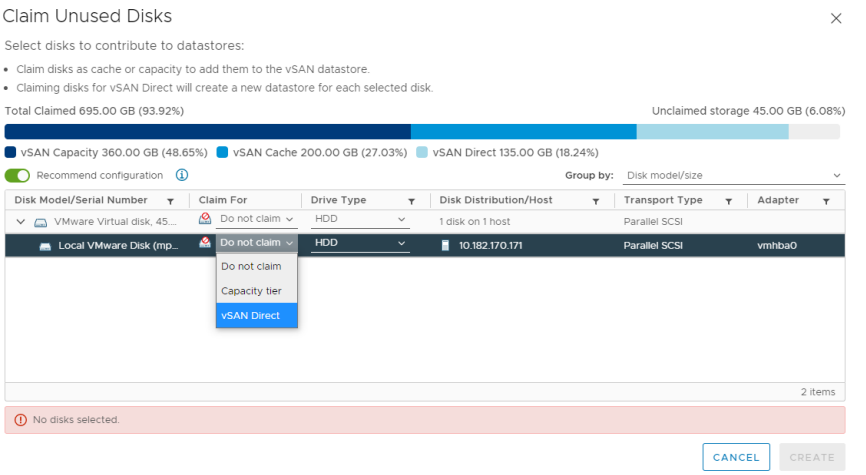
3 [vSAN] の下で、[ディスク管理] をクリックします。

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。

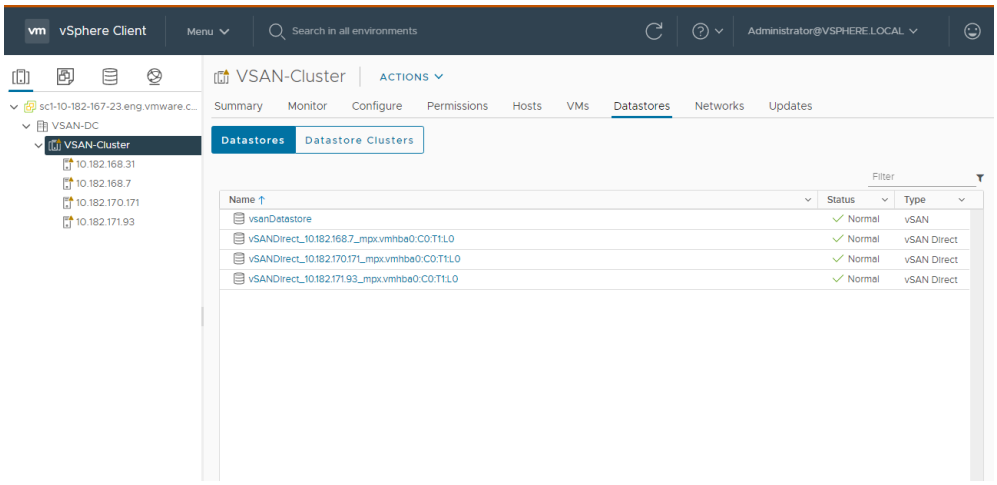
注： 通常の vSAN データストアのデバイスを要求した場合、これらのデバイスは [vSAN Direct] タブには表示されません。



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、『vSphere IaaS 制御プレーンのメンテナンス』ドキュメントの vSAN Direct と外部ストレージの併用を参照してください。

vSphere IaaS control plane 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 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。

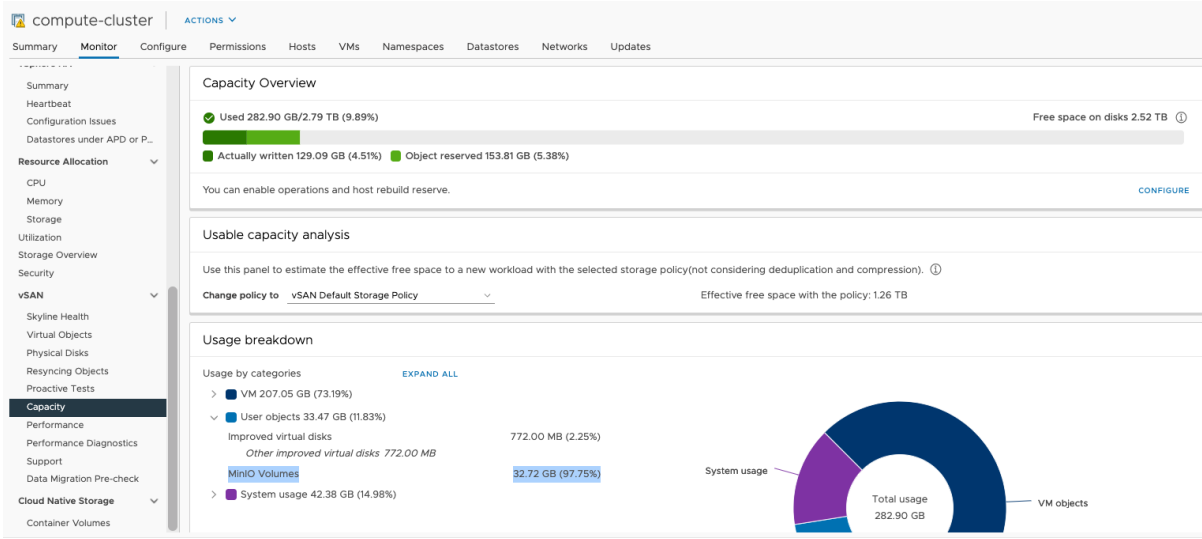
      仮想オブジェクト（MinIO オペレータ オブジェクトなど）を参照して、状態を確認することができます。

   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。

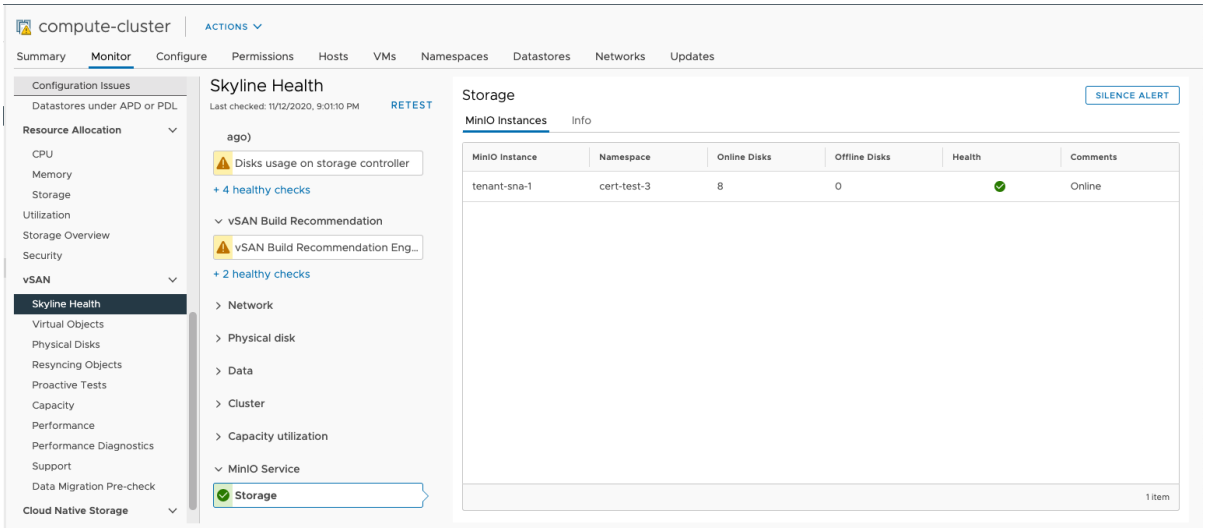
   b [使用量の内訳] ペインの[ユーザー オブジェクト] にサービス オブジェクトを表示します。



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 IaaS control plane 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 IaaS control plane 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 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 Grid 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 データ パーシステンス プラットフォームのカスタム ストレージ ポリシーの作成

vSphere IaaS control plane の スーパーバイザー でステートフル 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デフォルトのストレージ ポリシーと対応するストレージ クラスを作成し、それらをサービス名前空間に割り当てます。ポリシーは、vSAN Shared-Nothing-Architecture (SNA) および vSAN Direct データストア用です。デフォルトの代わりに、カスタム ストレージ ポリシーを作成できます。

使用するデータストアのタイプを判別するには、次の一般的な推奨事項に従います。

- 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。
- 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。

詳細については、『vSphere Shared Nothing Storage』を参照してください。

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。スーパーバイザーでの vSphere 名前空間 の作成と構成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 を使用する場合は、スーパーバイザー 名前空間で使用するストレージ ポリシーを作成します。このストレージ ポリシーに関連付ける名前空間では、ステートフル サービスやインスタンス ストレージ仮想マシンなど、vSAN Direct と互換性のあるワークロードを実行できます。

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] 画面で、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] 画面で、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] 画面で、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。

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。

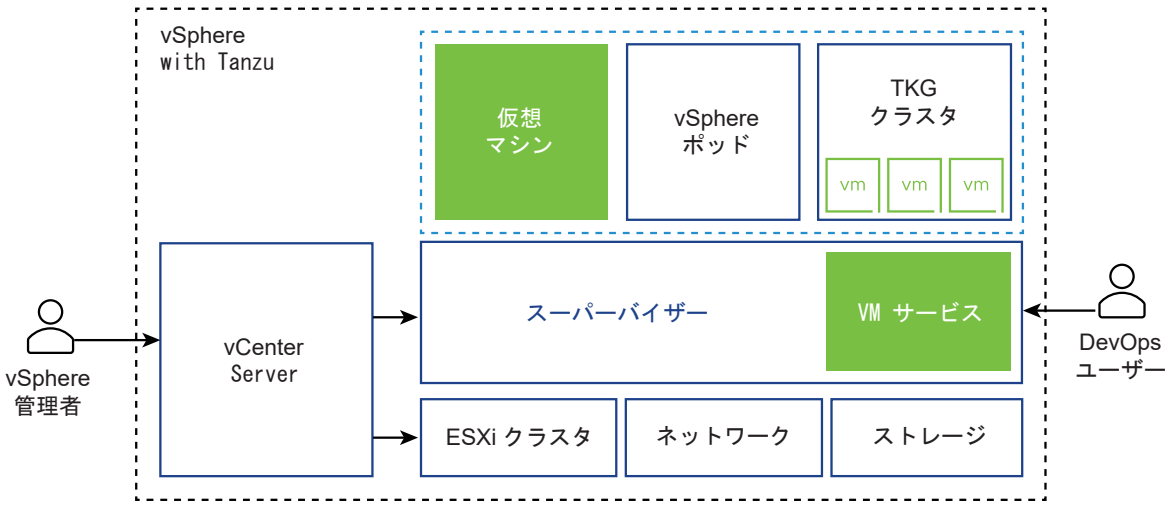
vSphere IaaS control plane での仮想マシンのデプロイと管理

6

vSphere IaaS control plane 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、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 Grid クラスタを構成する仮想マシンを管理します。

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 Grid 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、『vSphere IaaS 制御プレーンでの TKG サービスの使用』ドキュメント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 IaaS control plane 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 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。

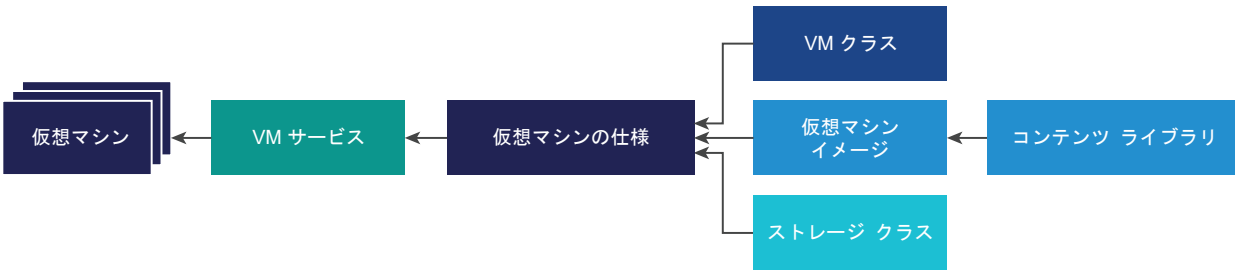
Kubernetes プラットフォームで仮想マシンを使用する場合

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 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 Grid 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 IaaS control plane 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 IaaS control plane 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 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` 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。

vSphere 管理者は、vSphere IaaS control plane と互換性のある仮想マシン イメージを作成し、コンテンツ ライブラリにアップロードできます。

コンテンツ ライブラリ

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、既存のコンテンツ ライブラリを名前空間またはクラスタに割り当てて、DevOps エンジニアがこれらを使用できるようにします。vSphere 管理者は、名前空間のコンテンツ ライブラリを書き込み可能にすることもできます。この追加の権限があることで、DevOps ユーザーはイメージをライブラリに公開することができます。

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、8 章 vSphere IaaS control plane の スーパーバイザー ワークロード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。

Kubernetes の仮想マシン オペレータ

仮想マシン オペレータでは、Kubernetes 形式の宣言型 API を使用して仮想マシンを管理できます。

vSphere 8.0 Update 3 リリース以降、vSphere IaaS control plane では仮想マシン オペレータ v1alpha2 がサポートされます。その他のメリットとして、このバージョンには次の特徴があります。

- インライン Cloud-Init および Windows のサポートを含むブートストラップ プロバイダのサポートを強化。
- ゲスト ネットワーク構成の強化。
- ステータス機能の強化。
- ユーザー定義の準備ゲートのサポート。
- 新しい VirtualMachineWebConsoleRequest API。

v1alpha2 に固有の新しい API の変更を除き、v1alpha2 の他のほとんどの API は v1alpha1 と同等に機能します。仮想マシン仕様のほとんどのフィールドは、v1alpha1 との下位互換性があります。

v1alpha2 のリリース後も、v1alpha1 オブジェクトを引き続き使用できます。すべての v1alpha1 オブジェクトは、仮想マシン オペレータに組み込まれた変換 Webhook を使用して v1alpha2 に自動的に変換されます。

仮想マシン オペレータ v1alpha2 とサポートされるフィールドについては、https://vm-operator.readthedocs.io/en/stable/ref/api/v1alpha2/ を参照してください。

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 IaaS control plane 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 の両方のタイプのネットワークをサポートします。仮想マシンが展開されると、使用可能なネットワーク プロバイダは仮想マシンに固定 IP アドレスを割り当てます。詳細については、『vSphere IaaS 制御プレーンの概念と計画』ドキュメントのスーパーバイザー ネットワークの説明を参照してください。

vSphere ゾーンを使用した仮想マシン サービスと スーパーバイザー

3 つのゾーンの スーパーバイザー に仮想マシンを作成すると、仮想マシン インスタンスがすべての使用可能なゾーンに複製されます。YAML ファイルを使用して仮想マシンの配置を制御するために、DevOps チームは Kubernetes ラベル `topology.kubernetes.io/zone` を使用できます。たとえば、`topology.kubernetes.io/zone: zone-a02` です。

仮想マシンのプロビジョニングと監視のワークフロー

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	実行者	説明
1	vSphere 管理者	vSphere IaaS control plane 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
2	vSphere 管理者	vSphere IaaS control plane での仮想マシン クラスの操作 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 IaaS control plane での vGPU および他の PCI デバイスを含む仮想マシンのデプロイを参照してください。
3	DevOp エンジニア	vSphere IaaS control plane でのスタンドアローン仮想マシンのデプロイ Tanzu Kubernetes Grid クラスタの仮想マシンの場合は、vSphere IaaS 制御プレーンでの TKG サービスの使用を参照してください。
4	vSphere 管理者	vSphere IaaS control plane で利用可能な仮想マシンの監視
5	DevOps エンジニア	vSphere IaaS control plane でのコンテンツ ライブラリ イメージの管理と公開

次のトピックを参照してください。

- vSphere IaaS control plane 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 IaaS control plane での仮想マシン クラスの操作
- Data Center CLI を使用した仮想マシン クラスの作成と管理
- vSphere IaaS control plane でのスタンドアローン仮想マシンのデプロイ
- vSphere IaaS control plane での vGPU および他の PCI デバイスを含む仮想マシンのデプロイ
- vSphere IaaS control plane でのインスタンス ストレージを使用した仮想マシンのデプロイ
- vSphere IaaS control plane での構成可能な OVF プロパティを使用した仮想マシンのデプロイ
- vSphere IaaS control plane で利用可能な仮想マシンの監視
- vSphere 仮想マシン Web コンソールを使用した仮想マシンのトラブルシューティング

vSphere IaaS control plane 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 IaaS control plane 環境内に仮想マシンをデプロイするには、DevOps ユーザーは、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、vSphere 管理者が仮想マシン 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。vSphere 8.0 Update 2 以降では、vSphere 管理者がコンテンツ ライブラリを スーパーバイザー レベルで割り当て、すべての名前空間で使用できるようにすることもできます。

vSphere IaaS control plane 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 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。

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

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。

コンテンツ ライブラリのアイテムを保護するには、OVF セキュリティ ポリシーを適用します。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、『vSphere の仮想マシン管理』のコンテンツ 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、[仮想マシン サービス] カードで [管理] をクリックします。

2 [仮想マシン 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。

コンテンツ ライブラリに対する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

コンテンツ ライブラリを作成したら、OVA または OVF 形式の仮想マシン テンプレート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 IaaS control plane 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。

ライブラリにポピュレートするには、いくつかの方法を使用できます。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、『vSphere の仮想マシン管理』の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。

注： 使用する仮想マシン イメージに制限はありません。すぐに使用可能な OVA イメージをテストする場合は、「Recommended Images」ページからイメージをダウンロードできます。これらのイメージは POC での使用を目的としたものであることに注意してください。本番環境では、最新のパッチと必要なセキュリティ設定を使用し、企業のセキュリティ ポリシーに従ったイメージを作成してください。

前提条件

- vSphere IaaS control plane と互換性のある仮想マシン イメージを作成します。

  イメージ仕様では、すべての仮想マシン イメージに VMware Tools または同等のオープン ソース パッケージが含まれている必要があります。ゲスト OS とそのネットワーク スタックをブートストラップするには、イメージで次のいずれかを使用する必要があります。詳細については、ブートストラップ プロバイダを参照してください。

  - Linux + Cloud-Init バージョン 17.9-21.2 と DataSourceVMwareGuestInfo。
  - Linux + Cloud-Init バージョン 21.3+
  - Windows + Cloudbase-Init バージョン 1.1.0 以降
  - Windows + Sysprep（システムの準備）

  Cloud-Init の詳細については、公式ドキュメントのクラウド インスタンスをカスタマイズするための標準を参照してください。

  Sysprep の詳細については、公式ドキュメントの Sysprep の概要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。

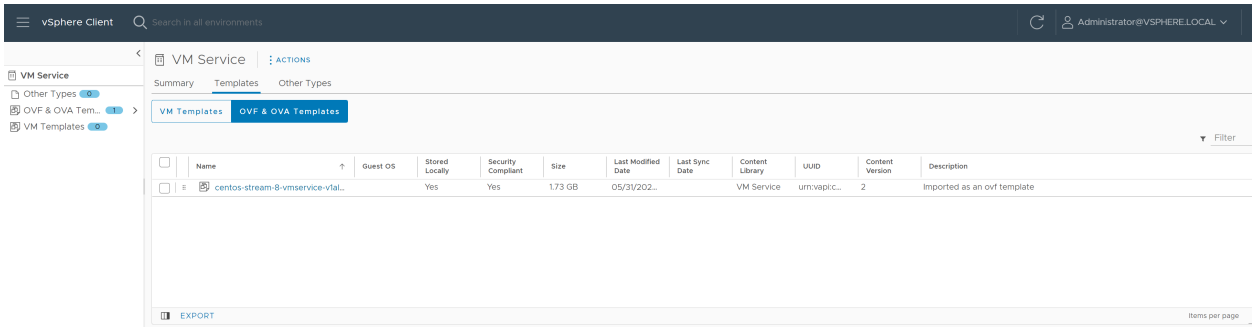
注： OVF パッケージがローカル マシンの `.ovf` ファイルからインポートされた場合、vCenter Server は署名済みコンテンツを読み取りません。

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。



vSphere IaaS control plane での仮想マシン コンテンツ ライブラリの追加と管理

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、vSphere Client を使用してそのライブラリを名前空間に追加します。ライブラリを名前空間に追加することで、DevOps ユーザーにライブラリへのアクセス権が付与されます。また、Data Center CLI (DCLI) コマンドを使用して、書き込み可能または読み取り専用のコンテンツ ライブラリを名前空間に追加したり、クラスタ レベルで読み取り専用ライブラリを割り当てたりできます。

vSphere Client を使用した名前空間への仮想マシン コンテンツ ライブラリの追加

vSphere Client で追加するコンテンツ ライブラリは読み取り専用です。DevOps ユーザーはこのコンテンツ ライブラリのイメージにアクセスできますが、このライブラリに仮想マシン イメージを公開することはできません。

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 のコンテンツ ライブラリは、Tanzu Kubernetes Grid カードから管理する必要があります。

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

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] カードで [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。

vSphere Client を使用した名前空間での仮想マシン コンテンツ ライブラリの管理

ライブラリと名前空間を関連付けた後で、vSphere Client を使用して名前空間からそれを削除できます。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 コンテンツ ライブラリは、Tanzu Kubernetes Grid カードから管理する必要があります。

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

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 [仮想マシン サービス] カードで [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 OVF テンプレートは 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、これを使用できます。vSphere IaaS control plane への仮想マシンのデプロイを参照してください。

注： 名前空間にはライブラリの OVF テンプレートのみが表示されます。他のタイプのコンテンツは名前空間に表示されません。

Data Center CLI を使用した名前空間への仮想マシン コンテンツ ライブラリの追加

vSphere 管理者は、Data Center CLI (DCLI) コマンドを使用して、名前空間にコンテンツ ライブラリを割り当てることができます。名前空間に関連付けるライブラリは、その割り当て時に、書き込み可能にすることができます。ライブラリが書き込み可能な場合、DevOps ユーザーはライブラリとライブラリ内のイメージを表示するだけでなく、そこに新しい仮想マシン イメージを公開できます。

ローカル、公開済み、サブスクライブ済みなど、あらゆるタイプのライブラリを、DCLI コマンドで名前空間に追加できます。ただし、書き込み可能ライブラリとしてリンクできるのは、ローカル ライブラリと公開済みライブラリのみです。コンテンツ ライブラリとライブラリ アイテムは、関連付けられた名前空間でのみ使用できます。

手順

1 root ユーザー アカウントを使用して vCenter Server にログインします。

2 DCLI を対話モードで使用するために、`dcli +i` と入力します。

3 名前空間に関連付けるコンテンツ ライブラリの ID を取得します。

```
dcli > namespacemanagement content library list
```

4 次のコマンドを実行して、コンテンツ ライブラリを名前空間に関連付けます。

更新操作は増分操作ではありません。名前空間に関連付けられるのは、指定したライブラリだけです。以前に追加されたライブラリは、その ID が指定されていない限り削除されます。たとえば、`'[{"content_library": "CLA", "writable": "true"}]'` を更新した後に

`'[{"content_library": "CLB", "writable": "true"}]'` を更新した場合、CLA は削除され、CLB だけが追加されます。CLA と CLB の両方を関連付ける場合は、`'[{"content_library": "CLA", "writable": "true"}, {"content_library": "CLB", "writable": "true"}]'` のように、両方のライブラリを指定する必要があります。

```
dcli > namespaces instances update --namespace namespace_name --content-libraries
'[{"content_library": "content_library_ID", "writable": "true | false"}]'
```

次の引数を使用します。

- `--namespace namespace_name` – 名前空間の名前。
- `--content_libraries content_library_ID writable: true | false` – 名前空間に関連付けるコンテンツ ライブラリの ID と、ライブラリが書き込み可能かどうか。

次に例を示します。

```
dcli > namespaces instances update --namespace lb-edit-ns --content-libraries
'[{"content_library": "cl-b585915ddxxxxxxxx", "writable": "true"}]'
```

5 名前空間からコンテンツ ライブラリを削除するには、`namespaces instances update` コマンドを繰り返して、アレイ リストからコンテンツ ライブラリ エントリを削除します。

次に例を示します。

```
dcli > namespaces instances update --namespace lb-edit-ns --content-libraries '[]'
```

結果

追加されたコンテンツ ライブラリが、DevOps 名前空間ビューで使用できるようになります。

DevOps ユーザーは、次のコマンドを実行して、コンテンツ ライブラリが追加または削除されたことを確認できます。

```
kubectl get cl -n lb-edit-ns
   NAMESPACE    NAME                   VSPHERENAME   TYPE    WRITABLE   STORAGETYPE   AGE
   lb-edit-ns   cl-b585915ddxxxxxxxx   Test-ns-cl    Local   true       Datastore     3m9s
kubectl describe cl cl-b585915ddxxxxxxxx -n lb-edit-ns
kubectl get clitem -n lb-edit-ns
```

Data Center CLI を使用した スーパーバイザー への仮想マシン コンテンツ ライブラリの追加

コンテンツ ライブラリは、名前空間レベルで割り当てるだけでなく、vSphere 管理者が Data Center CLI (DCLI) コマンドを使用して スーパーバイザー クラスタに関連付けることもできます。コンテンツ ライブラリは、スーパーバイザー 内のすべての名前空間で使用できるようになります。

ローカル、公開済み、サブスクライブ済みなど、あらゆるタイプのライブラリを関連付けることができます。

注： スーパーバイザー に関連付けられたコンテンツ ライブラリは読み取り専用です。DevOps ユーザーはこのコンテンツ ライブラリの仮想マシン イメージにアクセスすることはできますが、このライブラリに仮想マシン イメージを公開することはできません。

前提条件

DCLI コマンドの詳細については、「VMware Data Center CLI」を参照してください。

手順

1 root ユーザー アカウントを使用して vCenter Server にログインします。

2 DCLI を対話モードで使用するために、`dcli +i` と入力します。

3 スーパーバイザー の名前と スーパーバイザー に接続するコンテンツ ライブラリの ID を取得します。

a クラスタの一覧から スーパーバイザー の名前を取得します。

このコマンドを実行すると、vCenter Server で使用可能なクラスタがすべて一覧表示されます。

```
dcli > namespacemanagement clusters list
```

b vCenter Server で使用できるあらゆるタイプのすべてのコンテンツ ライブラリの ID を一覧表示します。

```
dcli > library list
```

c 特定のライブラリの詳細を確認します。

```
dcli > library get --library-id content_library_ID
```

4 1 つまたは複数のコンテンツ ライブラリを スーパーバイザー に関連付けます。

更新操作は増分操作ではありません。名前空間に関連付けられるのは、指定したライブラリだけです。以前に追加されたライブラリは、その ID が指定されていない限り削除されます。たとえば、`'[{"content_library": "CLA", "writable": "true"}]'` を更新した後に `'[{"content_library": "CLB", "writable": "true"}]'` を更新した場合、CLA は削除され、CLB だけが追加されます。CLA と CLB の両方を関連付ける場合は、`'[{"content_library": "CLA", "writable": "true"}, {"content_library": "CLB", "writable": "true"}]'` のように、両方のライブラリを指定する必要があります。

```
dcli > namespacemanagement clusters update --cluster cluster_name --content-libraries
'[{"content_library": content_library_ID_1}, {"content-library": content_library_ID_2}]'
```

次の引数を使用します。

- `--cluster cluster_name` – スーパーバイザー クラスタの ID。
- `--content-libraries content_library_ID` – スーパーバイザー に関連付けるコンテンツ ライブラリの ID。複数の ID を一覧表示できます。

  次に例を示します。

  ```
  dcli > namespacemanagement clusters update --cluster cluster_name --content-libraries
  '[{"content_library": 535d4b3d-xxxx-xxxx-xxxx-xxxxxxxxxxxx}, {"content-library":
  b5aa7f68-xxxx-xxxx-xxxx-xxxxxxxxxxxx}]'
  ```

5 コンテンツ ライブラリがクラスタに接続されていることを確認します。

```
dcli > namespacemanagement clusters get --cluster cluster_name
```

出力には、接続されたコンテンツ ライブラリの ID が含まれている必要があります。

6 関連付けられたコンテンツ ライブラリをクラスタから削除するには、`namespacemanagement clusters update` コマンドを繰り返して、コンテンツ ライブラリのアレイ リストからコンテンツ ライブラリ エントリを削除します。

次に例を示します。

```
dcli > namespacemanagement clusters update --cluster cluster_name --content-libraries '[]'
```

結果

新しく追加されたコンテンツ ライブラリが、DevOps クラスタ ビューで使用できるようになります。vSphere 管理者がコンテンツ ライブラリに加えた変更は、DevOps ビューに反映されます。DevOps ユーザーは、次のコマンドを実行してコンテンツ ライブラリを一覧表示し、その内容を記述できます。

- `kubectl get ccl` – クラスタ レベルで使用可能なすべてのコンテンツ ライブラリのリスト。次のような出力が得られます。

```
NAME                    VSPHERENAME                          TYPE          STORAGETYPE   AGE
cl-f28af8153fb849bd7    Kubernetes Service Content Library   Subscribed    Datastore     6d5h
cl-knounwp7xxxxxxxxx    Image Registry Content Library       Local         Datastore     6d4h
```

- `kubectl get cclitem` – クラスタ レベルのコンテンツ ライブラリにあるすべてのアイテムのリスト。
- `kubectl describe ccl NAME` – クラスタ レベルの特定のコンテンツ ライブラリに関する詳細情報。

vSphere IaaS control plane でのコンテンツ ライブラリ イメージの管理と公開

vSphere 管理者がコンテンツ ライブラリを名前空間またはクラスタに割り当てた後に、DevOps ユーザーはライブラリにアクセスし、そのアイテムを使用してライブラリ内の仮想マシン イメージから仮想マシンをデプロイできます。名前空間に割り当てられたライブラリが書き込み可能な場合、編集権限を持つ DevOps ユーザーもライブラリ アイテムを管理し、新しい仮想マシン イメージを公開できます。

注： 使用する仮想マシン イメージに制限はありません。すぐに使用可能な OVA イメージをテストする場合は、「Recommended Images」ページからイメージをダウンロードできます。これらのイメージは POC での使用を目的としたものであることに注意してください。本番環境では、最新のパッチと必要なセキュリティ設定を使用し、企業のセキュリティ ポリシーに従ったイメージを作成してください。

前提条件

DevOps ユーザーとして、次の要件を満たしていることを確認します。

- vSphere 名前空間に対する `Edit` 権限がある。
- vSphere 管理者が、書き込み可能なコンテンツ ライブラリを名前空間に割り当てた。Data Center CLI を使用した スーパーバイザー への仮想マシン コンテンツ ライブラリの追加 を参照してください。
- コンテンツ ライブラリはローカルまたは公開されています。サブスクライブ済みライブラリは編集できません。

手順

1 ライブラリ アイテムを管理します。

a 名前空間でコンテンツ ライブラリが使用可能であることを確認します。

注： ライブラリ内のライブラリ アイテムを管理する場合、またはライブラリに仮想マシン イメージを公開する場合は、その書き込み可能ステータスが true であることを確認します。

```
kubectl get cl -n <namespace-name>
   NAME                    VSPHERENAME    TYPE    WRITABLE   STORAGETYPE   AGE
   cl-b585915ddxxxxxxxxx   Test-ns-cl-1   Local   true       Datastore     3m9s
   cl-535d4b3dnxxxyyyyyy   Test-ns-cl-1   Local   false      Datastore     3m9s
```

b ライブラリのコンテンツを確認します。

```
kubectl get clitem -n <namespace-name>
 NAME                 VSPHERENAME     CONTENTLIBRAYREF          TYPE   READY   AGE
 clitem-d2wnmq.....   item 1          cl-b585915ddxxxxxxxxx     Ovf    True    26c
 clitem-55088d.....   item 2          cl-b585915ddxxxxxxxxx     Ovf    True    26c
 clitem-xyzxyz.....   xyzxyz          cl-535d4b3dnxxxyyyyyy     Ovf    True    26c
```

c コンテンツ ライブラリからイメージを削除します。

注： アイテムを削除できるのは、書き込み可能なライブラリからのみで、`Edit` 権限かそれより上のレベルの権限がある場合です。

アイテムを削除すると、対応する vmi リソースも削除されます。

```
kubectl delete clitem clitem-55088d.....
 NAME                 VSPHERENAME     CONTENTLIBRAYREF          TYPE   READY   AGE
 clitem-d2wnmq.....   item 1          cl-b585915ddxxxxxxxxx     Ovf    True    26c
 clitem-xyzxyz.....   xyzxyz          cl-535d4b3dnxxxyyyyyy     Ovf    True    26c
```

d イメージの詳細を取得します。

```
kubectl get vmi -n <namespace-name>
 NAME              PROVIDER-NAME      CONTENT-LIBRAY-NAME    IMAGE-NAME VERSION   OS-
TYPE        FORMAT   AGE
 vmi-d2wnmq.....   clitem-d2wnmq..... cl-b585915ddxxxxxxxxx   item 1
ubuntu64guest  ovf      26c
 vmi-55088d.....   clitem-55088d..... cl-b585915ddxxxxxxxxx   item 2
otherguest     ovf      26c
```

2 イメージをコンテンツ ライブラリに公開します。

a yaml ファイルを作成して、ソース仮想マシンをデプロイします。

VirtualMachine 仕様の imageName が、コンテンツ ライブラリ内のいずれかの仮想マシン イメージを参照していることを確認します。

たとえば、`source-vm.yaml` です。

```
apiVersion: vmoperator.vmware.com/v1alpha2
kind: VirtualMachine
metadata:
  name: source-vm
  namespace: test-publish-ns
spec:
  className: best-effort-small
  storageClass: wcpglobal-storage-profile
  imageName: vmi-d2wnmq.....
  powerState: poweredOn
  vmMetadata:
    transport: CloudInit
```

b デプロイされた仮想マシンに関する情報を取得し、仮想マシンに接続して、実行されていることを確認します。

次のような出力が表示されます。

```
kubectl get vm -n <namespace-name>
NAME        POWER-STATE   CLASS             IMAGE           PRIMARY-IP        AGE
source-vm   poweredOn     best-effort-small vmi-d2wnmq..... 192.168.000.00   9m32s
```

c　新しいターゲット イメージの公開要求を作成します。

たとえば、vmpub.yaml など。要求で、イメージを公開するソース仮想マシンとターゲット コンテンツ ライブラリの名前を指定します。ライブラリが書き込み可能であることを確認します。

```
apiVersion: vmoperator.vmware.com/v1alpha2
kind: VirtualMachinePublishRequest
metadata:
  name: vmpub-1
  namespace: test-publish-ns
spec:
  source:
    apiVersion: vmoperator.vmware.com/v1alpha2
    kind: VirtualMachine
    name: source-vm # If empty, the name of this VirtualMachinePublishRequest will be
used as the source VM name ("vmpub-1" in this example).
  target:
    item:
      name: publish-image-1 # If empty, the target item name is <source-vm-name>-image
by default
    location:
      apiVersion: imageregistry.vmware.com/v1alpha2
      kind: ContentLibrary
      name: cl-b585915ddxxxxxxxx
```

d　公開要求について説明します。

公開要求が Ready ステータスで、ImageName が設定されていることを確認します。

```
kubectl describe vmpub vmpub-1 -n <namespace-name>
==================
Status:
  imageName: vmi-12980cddd...
  ready: true
==================
```

e　公開要求が完了した後で、新しいイメージがコンテンツ ライブラリに追加されていることを確認します。

```
kubectl get vmi
 NAME               PROVIDER-NAME      CONTENT-LIBRAY-NAME    IMAGE-NAME      VERSION
OS-TYPE        FORMAT   AGE
 vmi-12980cddd..   clitem-12980cddd.. cl-b585915ddxxxxxxxx   publish-image-1
ubuntu64guest  ovf      7m12s
 vmi-d2wnmq.....   clitem-d2wnmq..... cl-b585915ddxxxxxxxx   item 1
ubuntu64guest  ovf      26m
 vmi-55088d.....   clitem-55088d..... cl-b585915ddxxxxxxxx   item 2
otherguest     ovf      26m
```

この新しいイメージを使用して、新しい仮想マシンをデプロイできます。

vSphere IaaS control plane での仮想マシン クラスの操作

vSphere IaaS control plane で仮想マシンのセルフサービスを行えるようにするには、DevOps ユーザーが仮想マシン クラスにアクセスできることが必要です。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 IaaS control plane 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。vSphere 管理者は、これらをそのまま使用することも、カスタム仮想マシン クラスを作成することもできます。DevOps ユーザーがクラスを使用できるようにするには、vSphere 管理者がクラスを名前空間に追加します。名前空間に割り当てられた仮想マシン クラスは、スタンドアローンの仮想マシンと、Tanzu Kubernetes Grid クラスタを構成する仮想マシンで使用できます。

vSphere Client を使用したカスタム仮想マシン クラスの作成

vSphere 管理者は、使用可能なデフォルト クラスを使用できます。また、デフォルト クラスの代わりにカスタム仮想マシン クラスを作成し、名前空間内の仮想マシンのデプロイに使用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。

注： DCLI コマンドを使用して仮想マシン クラスを作成することもできます。Data Center CLI を使用した仮想マシン クラスの作成と管理を参照してください。

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 [名前] 画面で、仮想マシン クラス名を指定し、[次へ] をクリックします。

   仮想マシン クラス名は、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。

4 [互換性] 画面で、仮想マシン クラスのハードウェア互換性を選択し、[次へ] をクリックします。

詳細については、「仮想マシンの互換性」を参照してください。

注： 仮想マシン クラスのハードウェア互換性は、その作成中にのみ設定でき、後で変更することはできません。

5 [構成] 画面で、設定をデフォルトのままにします。

6 [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

仮想マシンのハードウェアや仮想マシンのオプションなど、仮想マシン クラスの構成を編集します。

vSphere Client を使用した仮想マシン クラスの編集

仮想マシン クラスを作成後に編集する方法を確認します。CPU、メモリ、デバイスなどのハードウェア リソースを構成することや、仮想マシンのオプションおよび詳細パラメータを編集することができます。vSphere IaaS control plane 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。

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 Grid 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 Grid 仮想マシンは影響を受けません。新しい Tanzu Kubernetes Grid 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 Grid 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。スケーリングの詳細については、『vSphere IaaS 制御プレーンでの TKG サービスの使用』のワークロード クラスタのスケールの説明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

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 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。

3 [仮想ハードウェア] 画面で、メモリ、CPU、各種デバイスなど、仮想マシン クラスのハードウェア リソースを構成します。

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ると、すべての仮想マシン ハードウェア設定が適用されます。たとえば、CPU 構成値は、仮想マシン クラスを使用して DevOps ユーザーが作成するすべての仮想マシン専用の CPU リソースになります。

注： vSphere 8.0 Update 2b 以降では、仮想マシン クラスの作成および編集用ウィザードで CPU およびメモリ リソースを設定する単位が、パーセンテージから MB、GB、TB、MHz の数値に変更されます。過去に作成した仮想マシン クラスはすべて、CPU とメモリがパーセンテージで表示されますが、現在は新しい数値形式でそれらの値を編集できます。

仮想マシン構成オプション	説明
[CPU]	仮想マシン専用の CPU リソースを定義します。CPU リソースの構成の詳細については、「仮想 CPU の構成と制限」および「仮想マシンの CPU リソースの構成」を参照してください。
[メモリ]	仮想マシン用に構成されるメモリを MB、GB、または TB 単位で定義します。仮想マシンのメモリ リソースの詳細については、「仮想メモリの構成」を参照してください。
[ビデオ カード]	Windows AERO、CAD、Google Earth、およびその他の 3D 設計、モデリング、マルチメディア アプリケーションを利用するように 3D グラフィックスを構成します。ビデオ カード設定の詳細については、「3D グラフィックを構成する方法」を参照してください。
[セキュリティ デバイス]	Software Guard Extensions® (vSGX) を構成して、仮想マシン クラスのセキュリティを強化します。「Intel Software Guard Extensions による仮想マシンのセキュリティ強化」を参照してください。

4 [仮想ハードウェア] オプションで、[新規デバイスを追加] をクリックして仮想マシン クラスにデバイスを追加および構成します。

ストレージ コントローラ、ネットワーク アダプタ、USB、PCI デバイスなど、さまざまなデバイスを仮想マシン クラスに構成します。

仮想マシン構成オプション	説明
[RDM ディスク]	RAW デバイス マッピング (RDM) を追加して、仮想マシン データを仮想ディスク ファイルに格納するのではなく、SAN LUN に直接保存します。「仮想マシンへの RDM ディスクの追加」を参照してください。
[ホストの USB デバイス]	物理デバイスが仮想マシンが実行されているホストに接続されている場合は、ESXi ホストから仮想マシンに 1 つ以上の USB パススルー デバイスを追加します。「ESXi ホストから仮想マシンへの USB デバイスの追加」を参照してください。
[NVDIMM]	仮想マシン クラスに仮想 NVDIMM デバイスを構成すると、不揮発性メモリまたは永続的なコンピュータ メモリを使用できます。「仮想マシンへの NVDIMM デバイスの追加」を参照してください。
[CD/DVD ドライブ]	仮想マシン クラスに CD/DVD デバイスを構成します。「仮想マシンの CD または DVD ドライブを追加または変更する方法」を参照してください。
[NVMe コントローラ]、[SATA コントローラ]、[SCSI コントローラ]	仮想マシン クラスにストレージ コントローラを構成します。「SCSI、SATA、NVMe ストレージ コントローラの条件、制限事項、および互換性」を参照してください。
[USB コントローラ]	ESXi ホストまたはクライアント コンピューティングからの USB パススルーをサポートするために、仮想マシン クラスに USB コントローラを追加します。「USB コントローラの仮想マシンへの追加」を参照してください。
[PCI デバイス]	vSphere IaaS control plane 環境内の ESXi ホストに 1 つ以上の NVIDIA GRID GPU グラフィックス デバイスがある場合、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します。パススルー モードで仮想マシンが使用できるように、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。 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。 詳細と追加の要件については、「vSphere IaaS control plane での PCI デバイスを使用した仮想マシンのデプロイ」を参照してください。
[ウォッチドッグ タイマー]	仮想ウォッチドッグ タイマー (VWDT) デバイスを追加して、仮想マシン内のシステム パフォーマンスに関連する自立性を確保します。「仮想ウォッチドッグ タイマー デバイスの仮想マシンへの追加方法」を参照してください。
[プレシジョン クロック]	仮想マシンにプレシジョン クロック デバイスを追加します。プレシジョン クロックは、仮想マシンにプライマリ ESXi ホストのシステム時刻へのアクセスを提供する仮想クロック デバイスです。「仮想マシンにプレシジョン クロック デバイスを追加する方法」を参照してください。
[シリアル ポート]	物理シリアル ポートまたはホスト コンピュータ上のファイルへの仮想シリアル ポートの接続を構成します。「シリアル ポート構成の変更」を参照してください。

仮想マシン構成オプション	説明
[インスタンス ストレージ]	仮想マシンにインスタンス ストレージを構成します。仮想マシンは、パーシステント ストレージ ボリュームとともにインスタンス ストレージを使用できます。仮想マシンとは別に存在するパーシステント ボリュームとは異なり、インスタンス ストレージ ボリュームは仮想マシン インスタンスのライフサイクルに依存します。 [インスタンス ストレージ] オプションを使用して、適切なストレージ ポリシーを追加し、仮想マシンで使用するボリュームを構成することができます。 その他の要件については、vSphere IaaS control plane でのインスタンス ストレージを使用した仮想マシンのデプロイを参照してください。
[ネットワーク アダプタ]	仮想マシン クラスにネットワーク アダプタを構成します。DevOps ユーザーが仮想マシン クラスを使用して仮想マシンをデプロイするときは、アダプタへのワークロード ネットワークを指定できます。ワークロード ネットワークは、仮想マシンが実行されている vSphere 名前空間 に構成されている必要があります。サポートされているアダプタ タイプの詳細については、「ネットワーク アダプタの基本」を参照してください。

5 [仮想マシン オプション] 画面では、VMware Tools スクリプトを実行する仮想マシン オプションを設定、変更したり、リモート コンソールへのユーザー アクセスを制御したり、起動動作を構成したりできます。

仮想マシン クラスに構成できる仮想マシン オプションの詳細については、「仮想マシン オプションの設定」を参照してください。

6 [詳細パラメータ] 画面では、VMware の技術サポート担当者による指示があった場合、あるいは VMware のドキュメントでシステムの問題を修正するためにパラメータの追加または変更が指示されている場合に、仮想マシン構成パラメータを変更または追加します。

仮想マシンの詳細パラメータの詳細については、「仮想マシンの詳細ファイル パラメータの構成」を参照してください。

7 仮想マシン クラスを編集する準備ができたら、変更内容を確認し、[終了] をクリックします。

vSphere Client を使用した仮想マシン クラスと名前空間の関連付け

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 スーパーバイザー の 1 つ以上の名前空間に追加します。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 Grid 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 Grid 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 Grid 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。

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. 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 Grid 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。

vSphere Client を使用した名前空間での仮想マシン クラスの管理

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

- 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 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 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。

Data Center CLI を使用した仮想マシン クラスの作成と管理

仮想マシン クラスの作成と管理には、vSphere Client に加え、Data Center CLI (DCLI) コマンドを使用することもできます。DCLI コマンドを使用すると、vSphere Client では使用できない仮想マシン構成オプションも柔軟に利用することができます。

前提条件

root ユーザー アカウントを使用して vCenter Server にログインし、`dcli +i` と入力して DCLI を対話モードで使用します。

DCLI コマンドについては、「Overview of Running DCLI Commands」を参照してください。

使用可能な DCLI コマンド

コマンド	説明
`namespacemanagement virtualmachineclasses create`	仮想マシン クラス オブジェクトを作成します。
`namespacemanagement virtualmachineclasses delete`	仮想マシン クラス オブジェクトを削除します。
`namespacemanagement virtualmachineclasses get`	仮想マシン クラスに関する情報を返します。
`namespacemanagement virtualmachineclasses list`	すべての仮想マシン クラスに関する情報を返します。
`namespacemanagement virtualmachineclasses update`	仮想マシン クラス オブジェクトの構成を更新します。

Data Center CLI を使用した仮想マシン クラスの作成

vSphere 管理者として、DCLI `com vmware vcenter namespacemanagement virtualmachineclasses create` コマンドを使用して仮想マシン クラスを作成します。CPU、メモリ、メモリ予約、ネットワーク アダプタなどの仮想マシン プロパティを構成できます。

このコマンドには次の引数があります。

引数	説明
-h、--help	このヘルプ メッセージを表示して終了します。
--config-spec CONFIG_SPEC	仮想マシン クラスに関連付けられた VirtualMachineConfigSpec (json 入力)。

引数	説明
--cpu-count CPU_COUNT	必須。このクラスの仮想マシン用に構成される CPU の数（整数）。
--cpu-reservation CPU_RESERVATION	仮想マシン用に予約されている使用可能な CPU の合計の割合（整数）。
--description DESCRIPTION	仮想マシン クラスの説明（文字列）。
--devices-dynamic-direct-path-io-devices DEVICES_DYNAMIC_DIRECT_PATH_IO_DEVICES	動的 DirectPath I/O デバイスのリスト（json 入力）。
--devices-vgpu-devices DEVICES_VGPU_DEVICES	vGPU デバイスのリスト（json 入力）。
--id ID	必須。仮想マシン クラスの識別子（文字列）。
--instance-storage-policy INSTANCE_STORAGE_POLICY	インスタンス ストレージに対応するストレージ ポリシー（文字列）。
--instance-storage-volumes INSTANCE_STORAGE_VOLUMES	インスタンス ストレージ ボリュームのリスト（json 入力）。
--memory-mb MEMORY_MB	必須。このクラスの仮想マシン用に構成されたメモリ量 (MB)。
--memory-reservation MEMORY_RESERVATION	このクラスの仮想マシン用に予約される使用可能なメモリの割合。

さまざまなプロパティを持つ仮想マシン クラスの作成例を次に示します。

CPU およびメモリ

```
com vmware vcenter namespacemanagement virtualmachineclasses create --id cpu-mem-class --cpu-
count 2 --memory-mb 2048 --config-spec
'{"_typeName":"VirtualMachineConfigSpec","numCPUs":2,"memoryMB":2048}'
```

Config Spec での `numCPUs` と `memoryMB` の設定はオプションです。これらを設定する場合は、必須の vAPI フィールドである `--cpu-count` および `--memory-mb` と同じ値を使用する必要があります。

CPU およびメモリの予約

CPU とメモリの予約が含まれる Config Spec を使用して仮想マシン クラスを作成すると、`memoryAllocation` でのメモリの予約または制限は MB 単位、`cpuAllocation` は MHz 単位になります。

```
com vmware vcenter namespacemanagement virtualmachineclasses create --id cpu-res-class-1 --
config-spec
'{"_typeName":"VirtualMachineConfigSpec","numCPUs":2,"memoryMB":2048,"cpuAllocation":
{"_typeName":"ResourceAllocationInfo","reservation":200,"limit":200},"memoryAllocation":
{"_typeName":"ResourceAllocationInfo","reservation":200,"limit":200}}' --cpu-count 2 --memory-
mb 2048
```

ネットワーク アダプタ

次のコマンドを実行すると、E1000 タイプのネットワーク アダプタが作成されます。

```
com vmware vcenter namespacemanagement virtualmachineclasses create --id class-w-e1000 --cpu-
count 2 --memory-mb 2048 --config-spec
'{"_typeName":"VirtualMachineConfigSpec","deviceChange":
[{"_typeName":"VirtualDeviceConfigSpec","operation":"add","device":
{"_typeName":"VirtualE1000","key":-100}}]}'
```

vGPU

これらの例では、最初のコマンドにより、--devices-vgpu-devices フィールドを使用して vGPU を含む仮想マシン クラスが作成されます。2 番目のコマンドでは、configSpec を使用することにより、vGPU を含む仮想マシン クラスを作成します。

```
com vmware vcenter namespacemanagement virtualmachineclasses create --id vmclass-1 --devices-
vgpu-devices '[{"profile_name": "mockup-vmiop-8c"}]' --memory-reservation 100 --cpu-count 2 --
memory-mb 4096
```

```
com vmware vcenter namespacemanagement virtualmachineclasses create --id vmclass-2 --cpu-
count 2 --memory-mb 4096 --config-spec
'{"_typeName":"VirtualMachineConfigSpec","deviceChange":
[{"_typeName":"VirtualDeviceConfigSpec","operation":"add","device":
{"_typeName":"VirtualPCIPassthrough","key":20,"backing":
{"_typeName":"VirtualPCIPassthroughVmiopBackingInfo","vgpu":"mockup-vmiop-8c"}}}]}' --memory-
reservation 100
```

インスタンス ストレージ

次の例では、--instance-storage-volumes フィールドと --instance-storage-policy フィールドを使用することにより、インスタンス ストレージを含む仮想マシン クラスを作成します。

```
com vmware vcenter namespacemanagement virtualmachineclasses create --id vmclass-ist-1 --
instance-storage-volumes '[{"size":47}]' --instance-storage-policy "e28d4352-1d1e-431b-
b3f7-528bef5838a0" --cpu-count 2 --memory-mb 4096
```

この例の ID フィールドは、仮想マシン サービスの仮想マシンのインスタンス ストレージ デバイスを表す既知の virtualDisk ID です。

```
com vmware vcenter namespacemanagement virtualmachineclasses create --id vmclass-ist-2 --cpu-
count 2 --memory-mb 2048 --config-spec
'{"_typeName":"VirtualMachineConfigSpec","deviceChange":
[{"_typeName":"VirtualDeviceConfigSpec","operation":"add","fileOperation":"create","device":
{"_typeName":"VirtualDisk","key":0,"backing":
{"_typeName":"VirtualDiskFlatVer2BackingInfo","fileName":"","diskMode":"","thinProvisioned":fa
lse},"capacityInKB":0,"capacityInBytes":49283072,"vDiskId":
{"_typeName":"ID","id":"e28d4352-1d1e-431b-b3f7-528bef5838a0"}},"profile":
[{"_typeName":"VirtualMachineDefinedProfileSpec","profileId":"e28d4352-1d1e-431b-
b3f7-528bef5838a0","profileData":
{"_typeName":"VirtualMachineProfileRawData","extensionKey":"com.vmware.vim.sps"}}]}]}'
```

データセンター CLI を使用した仮想マシン クラスの更新

vSphere 管理者は、DCLI com vmware vcenter namespacemanagement virtualmachineclasses update コマンドを使用して仮想マシン クラスを変更します。

次に例を示します。

CPU およびメモリの変更

```
com vmware vcenter namespacemanagement virtualmachineclasses get --vm-class cpu-mem-class
com vmware vcenter namespacemanagement virtualmachineclasses update --vm-class cpu-mem-class
--cpu-count 4 --memory-mb 4096
```

CPU およびメモリ予約の変更

CPU とメモリの予約が含まれる Config Spec を使用して仮想マシン クラスを作成すると、`memoryAllocation` でのメモリの予約または制限は MB 単位、`cpuAllocation` は MHz 単位になります。

```
com vmware vcenter namespacemanagement virtualmachineclasses get --vm-class cpu-res-class-1
com vmware vcenter namespacemanagement virtualmachineclasses update --vm-class cpu-res-
class-1 --cpu-reservation 100 --memory-reservation 100
com vmware vcenter namespacemanagement virtualmachineclasses get --vm-class cpu-res-class-1
```

構成仕様を使用して CPU およびメモリの予約を更新することもできます。既存の CPU またはメモリの予約はすべて上書きされます。次に例を示します。

```
com vmware vcenter namespacemanagement virtualmachineclasses update --vm-class  cpu-res-
class-1 --config-spec '{"_typeName":"VirtualMachineConfigSpec","cpuAllocation":
{"_typeName":"ResourceAllocationInfo","reservation":200,"limit":200},"memoryAllocation":
{"_typeName":"ResourceAllocationInfo","reservation":200,"limit":200}}'
```

vGPU の追加

```
com vmware vcenter namespacemanagement virtualmachineclasses get --vm-class class-w-e1000
com vmware vcenter namespacemanagement virtualmachineclasses update --vm-class class-w-e1000
--devices-vgpu-devices '[{"profile_name": "mockup-vmiop-8c"}]'
com vmware vcenter namespacemanagement virtualmachineclasses get --vm-class class-w-e1000
```

```
com vmware vcenter namespacemanagement virtualmachineclasses get --vm-class vmclass-1
com vmware vcenter namespacemanagement virtualmachineclasses update --vm-class vmclass-1 --
config-spec '{"_typeName":"VirtualMachineConfigSpec","deviceChange":
[{"_typeName":"VirtualDeviceConfigSpec","operation":"add","device":
{"_typeName":"VirtualPCIPassthrough","key":20,"backing":
{"_typeName":"VirtualPCIPassthroughVmiopBackingInfo","vgpu":"mockup-vmiop-8c"}}},
{"_typeName":"VirtualDeviceConfigSpec","operation":"add","device":
{"_typeName":"VirtualPCIPassthrough","key":20,"backing":
{"_typeName":"VirtualPCIPassthroughVmiopBackingInfo","vgpu":"mockup-vmiop"}}}]}'
com vmware vcenter namespacemanagement virtualmachineclasses get --vm-class vmclass-1
```

vGPU の削除

```
com vmware vcenter namespacemanagement virtualmachineclasses update --vm-class vmclass-1 --
devices-vgpu-devices '[]'
 com vmware vcenter namespacemanagement virtualmachineclasses get --vm-class vmclass-1
```

インスタンス ストレージの追加

```
com vmware vcenter namespacemanagement virtualmachineclasses update --vm-class vmclass-1 --
instance-storage-volumes '[{"size":47}]' --instance-storage-policy "e28d4352-1d1e-431b-
b3f7-528bef5838a0"
 com vmware vcenter namespacemanagement virtualmachineclasses get --vm-class vmclass-1
```

```
com vmware vcenter namespacemanagement virtualmachineclasses get --vm-class vmclass-ist-2
com vmware vcenter namespacemanagement virtualmachineclasses update --vm-class vmclass-ist-2
--instance-storage-volumes '[{"size":51},{"size":50}]' --instance-storage-policy
"e28d4352-1d1e-431b-b3f7-528bef5838a0"
com vmware vcenter namespacemanagement virtualmachineclasses get --vm-class vmclass-ist-2
com vmware vcenter namespacemanagement virtualmachineclasses update --vm-class vmclass-ist-2
--config-spec '{"_typeName":"VirtualMachineConfigSpec","deviceChange":
[{"_typeName":"VirtualDeviceConfigSpec","operation":"add","fileOperation":"create","device":
{"_typeName":"VirtualDisk","key":0,"backing":
{"_typeName":"VirtualDiskFlatVer2BackingInfo","fileName":"","diskMode":"","thinProvisioned":fa
lse},"capacityInKB":0,"capacityInBytes":52428800,"vDiskId":
{"_typeName":"ID","id":"cc737f33-2aa3-4594-aa60-df7d6d4cb984"}},"profile":
[{"_typeName":"VirtualMachineDefinedProfileSpec","profileId":"e28d4352-1d1e-431b-
b3f7-528bef5838a0","profileData":
{"_typeName":"VirtualMachineProfileRawData","extensionKey":"com.vmware.vim.sps"}}]}]}'
com vmware vcenter namespacemanagement virtualmachineclasses get --vm-class vmclass-ist-2
```

インスタンス ストレージの削除

```
com vmware vcenter namespacemanagement virtualmachineclasses update --vm-class vmclass-ist-1
--instance-storage-volumes '[]' --instance-storage-policy ""
com vmware vcenter namespacemanagement virtualmachineclasses get --vm-class vmclass-ist-1
```

ネットワーク アダプタの追加

このコマンドを使用すると、インスタンス ストレージと e1000 NIC の両方が仮想マシン クラスに追加されます。

```
com vmware vcenter namespacemanagement virtualmachineclasses update --vm-class vmclass-ist-2
--config-spec '{"_typeName":"VirtualMachineConfigSpec","deviceChange":
[{"_typeName":"VirtualDeviceConfigSpec","operation":"add","fileOperation":"create","device":
{"_typeName":"VirtualDisk","key":0,"backing":
{"_typeName":"VirtualDiskFlatVer2BackingInfo","fileName":"","diskMode":"","thinProvisioned":fa
lse},"capacityInKB":0,"capacityInBytes":52428800,"vDiskId":
{"_typeName":"ID","id":"cc737f33-2aa3-4594-aa60-df7d6d4cb984"}},"profile":
[{"_typeName":"VirtualMachineDefinedProfileSpec","profileId":"e28d4352-1d1e-431b-
b3f7-528bef5838a0","profileData":
{"_typeName":"VirtualMachineProfileRawData","extensionKey":"com.vmware.vim.sps"}}]},
{"_typeName":"VirtualDeviceConfigSpec","operation":"add","device":
{"_typeName":"VirtualE1000","key":-100}}]}'
com vmware vcenter namespacemanagement virtualmachineclasses get --vm-class vmclass-ist-2
```

構成仕様を空にする

```
com vmware vcenter namespacemanagement virtualmachineclasses update --vm-class class-w-e1000
--config-spec ''
```

次の手順

DCLI を使用して作成した仮想マシン クラスは、vCenter Server で使用できるようになります。vSphere Client を使用して、これらの仮想マシン クラスを名前空間に割り当てることができます。vSphere Client を使用した仮想マシン クラスと名前空間の関連付けを参照してください。

vSphere IaaS control plane でのスタンドアローン仮想マシンのデプロイ

DevOps エンジニアは、`kubectl` コマンドを使用して、使用可能な仮想マシン リソースを確認し、スタンドアローンの Linux 仮想マシンまたは Windows 仮想マシンを スーパーバイザー 上の名前空間にプロビジョニングできます。vGPU 用に構成された PCI デバイスが仮想マシンに含まれている場合は、vSphere IaaS control plane 環境で仮想マシンを作成して起動した後、NVIDIA vGPU グラフィック ドライバをインストールして GPU 操作をに有効にできます。

前提条件

スタンドアローン仮想マシンを vSphere IaaS control plane 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。vSphere 管理者が次の手順を実行して仮想マシン リソースを使用可能にしたことを確認します。

- 名前空間を作成し、ストレージ ポリシーを割り当てます。スーパーバイザー での vSphere 名前空間 の作成と構成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す。vSphere IaaS control plane 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。
  - 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。
  - vGPU デバイスを使用して仮想マシンをデプロイする場合は、起動モードを EFI に設定したイメージ（CentOS など）にアクセスできる必要があります。
- 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す。vSphere IaaS control plane での仮想マシン クラスの操作を参照してください。

仮想マシンに NVIDIA vGPU またはその他の PCI デバイスを使用する場合は、追加の要件に従う必要があります。詳細については、『vSphere IaaS control plane での PCI デバイスを使用した仮想マシンのデプロイ』を参照してください。

仮想マシン オペレータとサポートされるフィールドについては、「仮想マシン サービスの概念」および https://vm-operator.readthedocs.io/en/stable/ref/api/v1alpha2/ を参照してください。

vSphere IaaS control plane の名前空間で使用可能な仮想マシン リソースの表示

DevOps エンジニアは、名前空間の仮想マシン 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 Grid クラスタを展開するために必要なリソースと、クラスタを構成する仮想マシンの詳細については、『vSphere IaaS 制御プレーンでの TKG サービスの使用』ドキュメントの TKG クラスタの仮想マシンのクラスを参照してください。

手順

1 Kubernetes 環境の名前空間にアクセスします。

vSphere IaaS control plane での スーパーバイザー 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 virtualmachineclass
```

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。`vmi-xxxxxxxxxxxxx` などのイメージ名は、システムによって自動生成されます。

```
NAME                                                VERSION  OSTYPE                FORMAT
IMAGESUPPORTED  AGE
vmi-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 vmi-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。

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 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
```

vSphere IaaS control plane 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します。

前提条件

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、「vSphere IaaS control plane での PCI デバイスを使用した仮想マシンのデプロイ」を参照してください。

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/v1alpha2**。
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	Kubernetes クラスタ内の仮想マシン イメージ リソース名を指定します。
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 クラスを指定します。
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。
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。必要に応じて、名前を指定するか、デフォルトの名前のままにします。
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。 Microsoft システム準備ツール (Sysprep) を使用して Windows 仮想マシンをブートストラップおよびカスタマイズする方法など、詳細については、「Customizing a Guest」を参照してください。
topology.kubernetes.io/zone	3 ゾーン スーパーバイザー で仮想マシンの配置を制御します。たとえば、topology.kubernetes.io/zone: zone-a02 です。

次の例の仮想マシン YAML ファイル my-vm では、ブートストラップ方法として CloudInit を使用します。この例は、シークレット リソース my-vm-bootstrap-data でユーザー データを指定する VirtualMachine リソースを示しています。シークレットは、ゲスト OS のブートストラップとカスタマイズに使用されます。

シークレットのデータには、CloudInit cloud-config が含まれます。cloud-config 形式の詳細については、公式ドキュメント「Cloud config examples」を参照してください。

ブートストラップ方法として Sysprep を使用する例については、「Sysprep」を参照してください。

```
apiVersion: vmoperator.vmware.com/v1alpha2
kind: VirtualMachine
metadata:
  name:      my-vm
  namespace: my-namespace
spec:
  className:    small
  imageName:    vmi-xxxxxxxxxxxxx
```

```
  storageClass: iscsi
  vmMetadata:
    transport: CloudInit
    secretName: my-vm-bootstrap-data
```

```
apiVersion: v1
kind: Secret
metadata:
  name:      my-vm-bootstrap-data
  namespace: my-namespace
stringData:
  user-data: |
    #cloud-config
    users:
    - default
    - name: xyz..
      primary_group: xyz..
      groups: users
      ssh_authorized_keys:
      - ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDSL7uWGj...
    runcmd:
    - "ls /"
    - [ "ls", "-a", "-l", "/" ]
    write_files:
    - path: /etc/my-plaintext
      permissions: '0644'
      owner: root:root
      content: |
        Hello, world.
```

ゾーンのある環境に仮想マシンをデプロイする場合は、次の例を使用します。

ZONE_NAME の値を取得するには、`kubectl get vspherezones` コマンドを実行します。

```
apiVersion: vmoperator.vmware.com/v1alpha2
kind: VirtualMachine
metadata:
  name: <vm-name>
  namespace: <vm-ns>
  labels:
    topology.kubernetes.io/zone: ZONE_NAME
...
```

2 仮想マシンをデプロイします。

```
kubectl apply -f my-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
NAME              AGE
my-vm             28s
```

4 仮想マシンの詳細とそのステータスを確認します。

```
kubectl describe virtualmachine my-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:         my-vm
Namespace:    my-namespace
API Version:  vmoperator.vmware.com/v1alpha2
Kind:         VirtualMachine
Metadata:
  Creation Timestamp:  2021-03-23T19:07:36Z
  Finalizers:
    virtualmachine.vmoperator.vmware.com
  Generation:  1
  Managed Fields:
 ...
 ...
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 IaaS control plane 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。

vSphere IaaS control plane での vGPU および他の PCI デバイスを含む仮想マシンのデプロイ

vSphere IaaS control plane 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。

vSphere IaaS control plane での vGPU を使用した仮想マシンのデプロイ

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。

考慮事項

NVIDIA vGPU を使用するときは次の考慮事項が適用されます。

- 3 ゾーン スーパーバイザー は、vGPU を使用する仮想マシンをサポートしていません。
- 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。
- DRS は、vGPU 仮想マシンを幅優先方式でクラスタのホスト全体に分散します。詳細については、『vSphere のリソース管理』ガイドの「DRS Placement of vGPU VMs」を参照してください。

要件

NVIDIA vGPU を構成するには、次の要件に従います。

- ESXi 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。
- [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。『vSphere のリソース管理』ドキュメントのホスト グラフィックの構成を参照してください。
- vGPU デバイスを含む仮想マシンに対して使用するコンテンツ ライブラリには、起動モードが EFI に設定されたイメージ（CentOS など）が含まれている必要があります。
- 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 IaaS control plane 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 Client を使用した仮想マシン クラスへの vGPU デバイスの追加

仮想マシン クラスを作成、または既存の仮想マシン クラスを編集して、NVIDIA GRID 仮想 GPU (vGPU) を追加します。

前提条件

必要な権限：

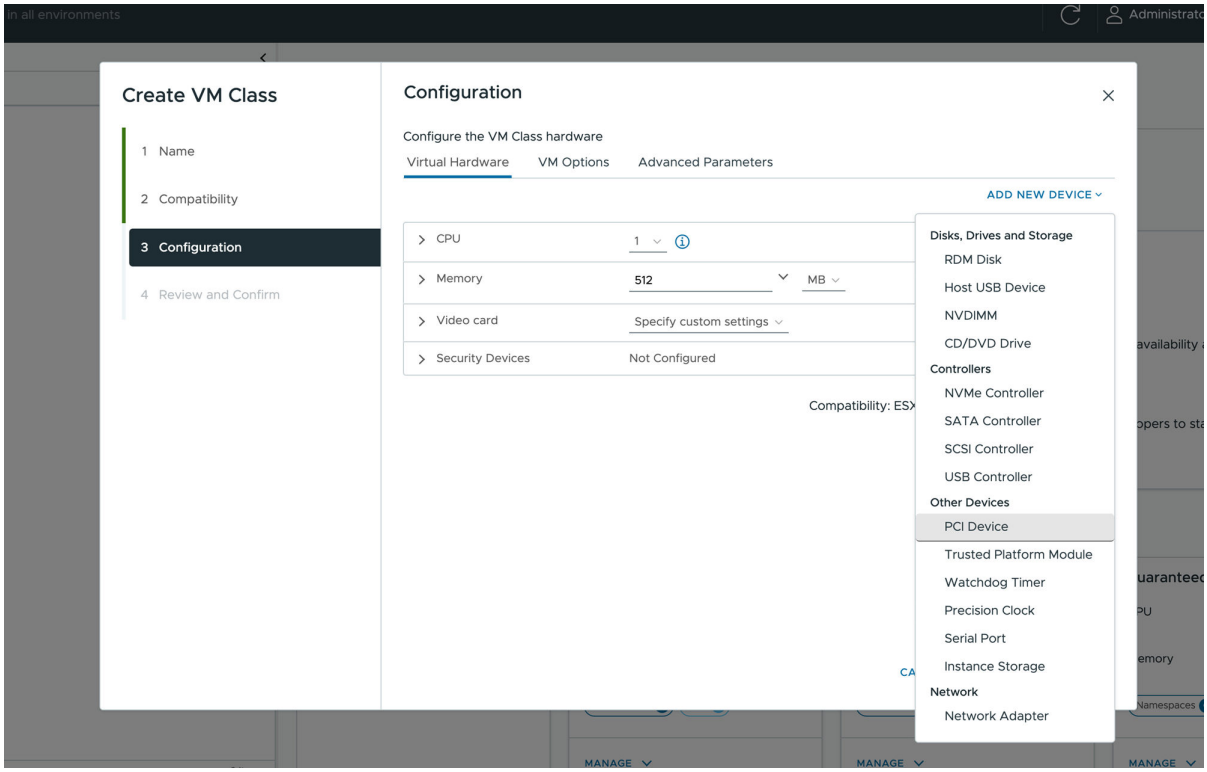
- 名前空間.クラスタ全体の構成の変更
- 名前空間.名前空間構成の変更
- 仮想マシン クラス.仮想マシン クラスの管理

手順

1 仮想マシン クラスを作成、または既存の仮想マシン クラスを編集します。

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 プロンプトに従います。
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。 e プロンプトに従います。

2 [構成] 画面で [仮想ハードウェア] タブをクリックし、[新規デバイスを追加] をクリックして、[PCI デバイス] を選択します。

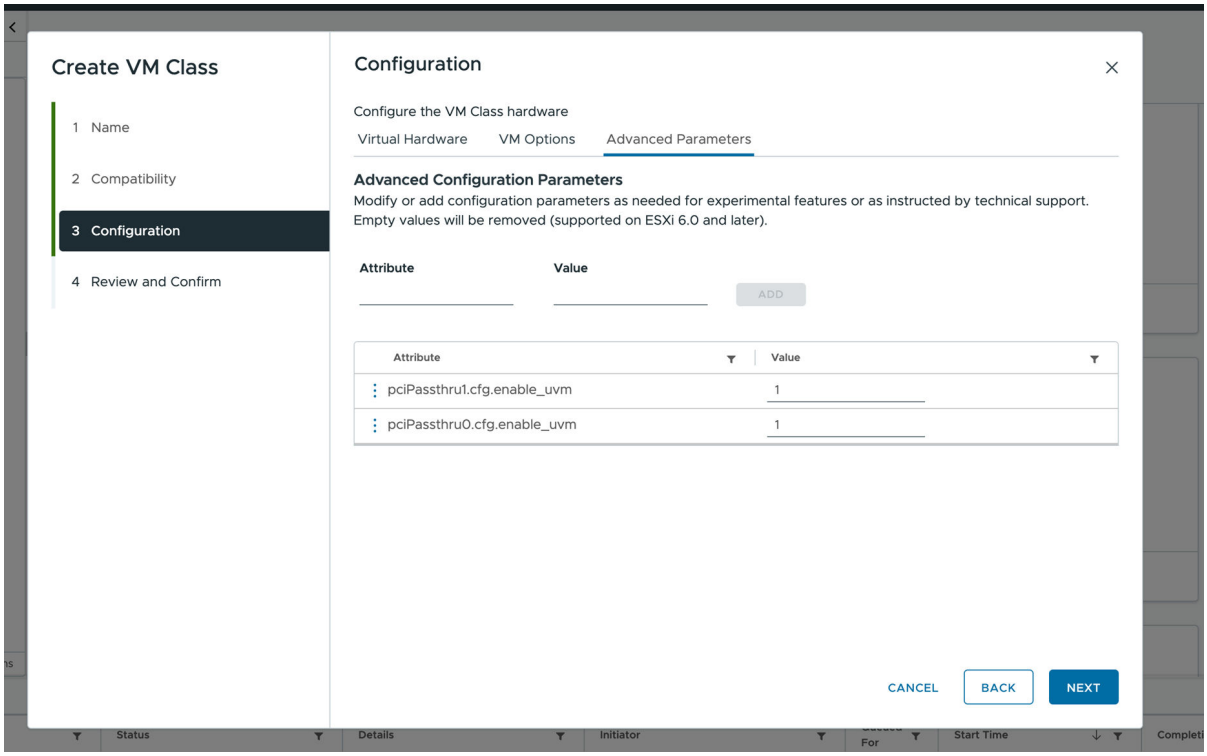


3 [デバイスの選択] 画面で、使用可能なデバイスのリストから NVIDIA GRID vGPU を選択し、[選択] をクリックします。

デバイスが [仮想ハードウェア] 画面に表示されます。

4 [詳細パラメータ] タブをクリックし、以下の属性と値を使用してパラメータを設定します。

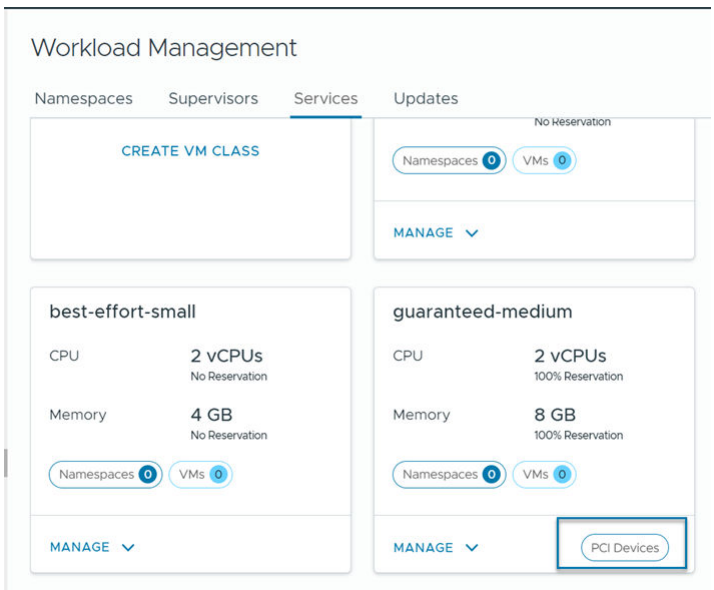
オプション	説明
パラメータ	値
pciPassthru0.cfg.enable_uvm	1
pciPassthru1.cfg.enable_uvm	1



5 構成を確認して [終了] をクリックします。

結果

仮想マシン クラスのペインに [PCI デバイス] タグが表示されている場合は、仮想マシン クラスが vGPU 対応となっています。



データセンター CLI を使用した仮想マシン クラスへの vGPU デバイスの追加

vGPU と詳細構成の追加には、vSphere Client に加え、データセンター CLI (DCLI) コマンドを使用することもできます。

DCLI コマンドの詳細については、「Data Center CLI を使用した仮想マシン クラスの作成と管理」を参照してください。

手順

1 root ユーザー アカウントを使用して vCenter Server にログインし、`dcli +i` と入力して DCLI を対話モードで使用します。

2 次のコマンドを実行して、仮想マシン クラスを作成します。

次の例で示されている my-class 仮想マシン クラスには、2 個の CPU、2,048 MB のメモリ、2 つのサンプル vGPU プロファイル `mockup-vmiop-8c` および `mockup-vmiop` を備えた VirtualMachineConfigSpec が含まれています。extraConfig のフィールド `pciPassthru0.cfg.enable_uvm` および `pciPassthru1.cfg.enable_uvm` は、1 に設定されます。

```
dcli +i +show-unreleased com vmware vcenter namespacemanagement virtualmachineclasses
create --id my-class --cpu-count 2 --memory-mb 2048 --config-spec
'{"_typeName":"VirtualMachineConfigSpec","deviceChange":
[{"_typeName":"VirtualDeviceConfigSpec","operation":"add","device":
{"_typeName":"VirtualPCIPassthrough","key":20,"backing":
{"_typeName":"VirtualPCIPassthroughVmiopBackingInfo","vgpu":"mockup-vmiop-8c"}}},
{"_typeName":"VirtualDeviceConfigSpec","operation":"add","device":
{"_typeName":"VirtualPCIPassthrough","key":20,"backing":
{"_typeName":"VirtualPCIPassthroughVmiopBackingInfo","vgpu":"mockup-
vmiop"}}}],"extraConfig":
[{"_typeName":"OptionValue","key":"pciPassthru0.cfg.enable_uvm","value":
{"_typeName":"string","_value":"1"}},
{"_typeName":"OptionValue","key":"pciPassthru1.cfg.enable_uvm","value":
{"_typeName":"string","_value":"1"}}]}'
```

vSphere IaaS control plane の仮想マシンへの NVIDIA ゲスト ドライバのインストール

仮想マシンに vGPU 用に構成された PCI デバイスが含まれている場合は、vSphere IaaS control plane 環境で仮想マシンを作成して起動した後、NVIDIA vGPU グラフィック 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

- vGPU を備えた仮想マシンをデプロイします。仮想マシンの YAML ファイルが、vGPU 定義を持つ仮想マシン クラスを参照していることを確認します。vSphere IaaS control plane への仮想マシンのデプロイ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。

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 IaaS control plane での PCI デバイスを使用した仮想マシンのデプロイ

パススルー モードで仮想マシンを使用できるように、vGPU に加えて 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。

vSphere IaaS control plane では、動的 DirectPath I/O デバイスがサポートされます。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。

注： PCI パススルー デバイスに動的 DirectPath I/O を構成する場合は、PCI デバイスをホストに接続し、そのデバイスをパススルーで使用可能とマークします。『vSphere のネットワーク』ドキュメントのホストでのネットワーク デバイスのパススルーを有効にするを参照してください。

vSphere IaaS control plane でのインスタンス ストレージを使用した仮想マシンのデプロイ

仮想マシンは、パーシステント ストレージ ボリュームとともにインスタンス ストレージを使用できます。仮想マシンとは別に存在するパーシステント ボリュームとは異なり、インスタンス ストレージ ボリュームは仮想マシン インスタンスのライフサイクルに依存します。通常、このストレージは、ESXi ホストに対してローカルな、NVMe などの高速デバイスに配置されます。

インスタンス ストレージのライフサイクル

仮想マシンの作成時に、システムによってインスタンス ストレージ ボリュームが作成され、仮想マシンに接続されます。インスタンス ストレージ ボリューム内のデータは、関連付けられた仮想マシン インスタンスの有効期間中にのみ保持されます。ボリュームは、仮想マシンが削除されると削除されます。

インスタンス ストレージを備えた仮想マシンは、ESXi ホストのメンテナンス モードをサポートします。仮想マシンは、ESXi ホストがメンテナンス モードになるとオフになり、ホストがメンテナンス モードを終了するとオンになります。

インスタンス ストレージ仮想マシンに関する考慮事項

インスタンス ストレージで仮想マシンを使用する場合は、次の項目を考慮してください。

- VDS ネットワーク スタックを使用する スーパーバイザー は、インスタンス ストレージをサポートしていません。
- 3 ゾーン スーパーバイザー は、インスタンス ストレージをサポートしていません。
- vSphere 管理者が、インスタンス ストレージに必要な適切なストレージ ポリシーのない名前空間にインスタンス ストレージを持つ仮想マシン クラスを適用すると、警告が表示されます。
- インスタンス ボリュームを持つ仮想マシンは、他の ESXi ホストに移行できません。
- ボリュームがすでに使用中の場合、インスタンス ストレージ ボリュームは編集できません。
- vSphere 管理者が仮想マシンの作成後に名前空間からインスタンス ストレージ ポリシーを削除した場合、仮想マシンは引き続き実行されます。
- DevOps エンジニアは、インスタンス ストレージ リソースを削除または更新することはできません。インスタンス ストレージ ボリュームをある仮想マシン インスタンスから接続解除し、別のインスタンスに接続することはできません。

インスタンス ストレージ仮想マシンをプロビジョニングおよび監視するためのワークフロー

手順	実行者	説明
1	vSphere 管理者	vSphere IaaS control plane 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
2	vSphere 管理者	vSAN Direct データストアを作成します。
3	vSphere 管理者	vSAN Direct と互換性のあるストレージ ポリシーを作成し、名前空間に割り当てます。

手順	実行者	説明
4	vSphere 管理者	インスタンス ストレージ仮想マシン クラスを作成し、名前空間に割り当てます。
5	DevOps エンジニア	インスタンス ストレージを持つ仮想マシンを名前空間にプロビジョニングします。
6	vSphere 管理者	vSphere IaaS control plane で利用可能な仮想マシンの監視

vSAN Direct データストアの作成

vSphere 管理者は、vSAN データ パーシステンス プラットフォームや仮想マシン インスタンス ストレージなどの機能で使用する vSAN Direct データストアを設定します。データストアを作成するには、ESXi ホストに対してローカルな未要求のストレージ デバイスを使用します。

スーパーバイザー に対して vSAN を有効にするときに、vSAN Direct データストアを作成できます。次のタスクは、クラスタで vSAN がすでに有効になっている場合に、vSAN Direct としてローカル ストレージ デバイスを要求する方法を示しています。

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。

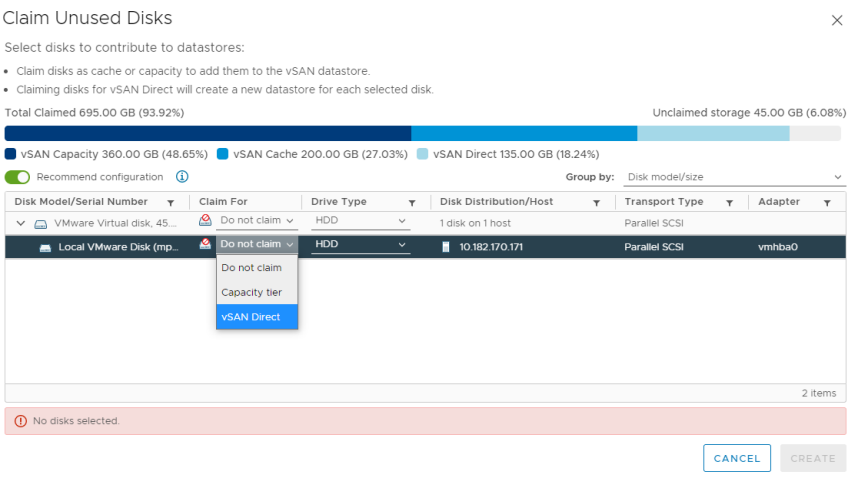
3 [vSAN] の下で、[ディスク管理] をクリックします。

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。

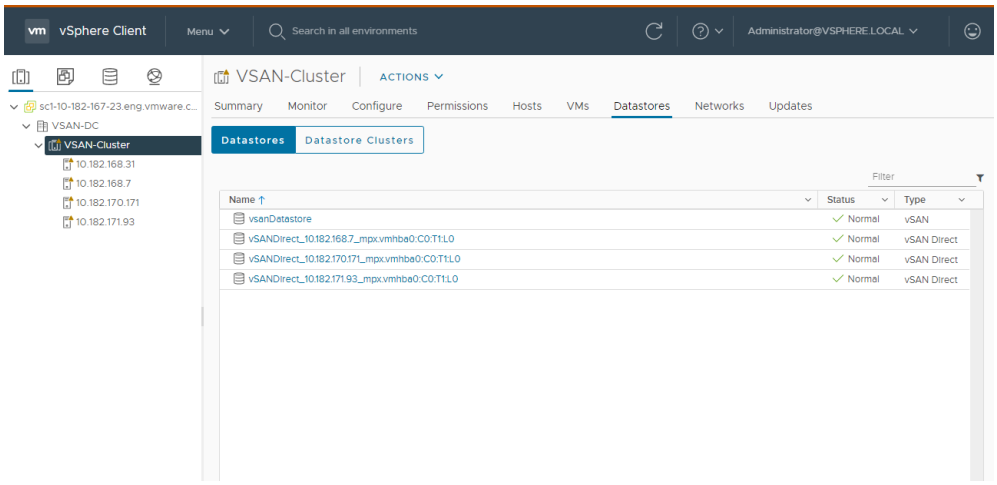
注： 通常の vSAN データストアのデバイスを要求した場合、これらのデバイスは [vSAN Direct] タブには表示されません。



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、『vSphere IaaS 制御プレーンのメンテナンス』ドキュメントの vSAN Direct と外部ストレージの併用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 を使用する場合は、スーパーバイザー 名前空間で使用するストレージ ポリシーを作成します。このストレージ ポリシーに関連付ける名前空間では、ステートフル サービスやインスタンス ストレージ仮想マシンなど、vSAN Direct と互換性のあるワークロードを実行できます。

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] 画面で、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] 画面で、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] 画面で、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。

インスタンス ストレージを使用する仮想マシン クラスの作成

仮想マシン クラスでは、vSAN Direct ストレージ ポリシーを参照し、インスタンス ストレージに使用するボリュームのサイズを設定します。仮想マシン クラスを作成したら、インスタンス ストレージ仮想マシンに使用する名前空間に割り当てます。

前提条件

- vSAN Direct データストアと互換性のあるストレージ ポリシーを作成します。
- インスタンス ストレージ仮想マシンで使用する名前空間に、vSAN Direct ストレージ ポリシーを追加します。スーパーバイザー での vSphere 名前空間 の作成と構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

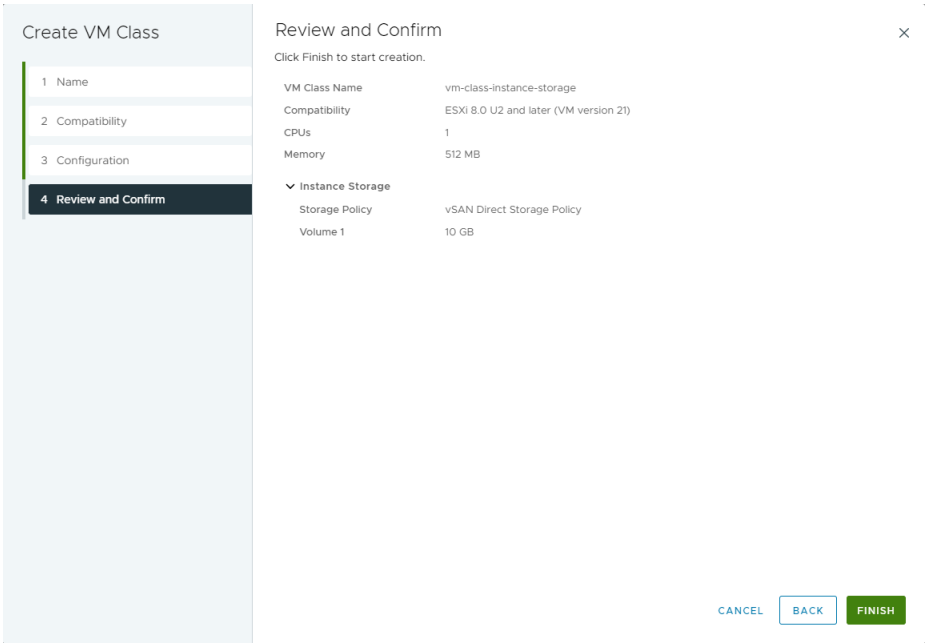
1 仮想マシン クラスを作成または編集するときに、インスタンス ストレージを追加します。

オプション	操作
仮想マシン 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、[仮想マシン サービス] カードで [管理] をクリックします。 c [仮想マシン サービス] 画面で、[仮想マシン クラスの作成] をクリックします。 d 必要に応じて仮想マシン クラスを構成します。使用可能なオプションについては、「vSphere Client を使用した仮想マシン クラスの編集」を参照してください。 e インスタンス ストレージを追加するには、[構成] 画面で [仮想ハードウェア] を選択してから、[新規デバイスを追加] - [インスタンス ストレージ] の順に選択します。 [インスタンス ストレージ] オプションが [仮想ハードウェア] に表示されます。
既存の仮想マシン 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 クラスのカードで、[管理] をクリックして [編集] をクリックします。 e インスタンス ストレージを追加するには、[仮想ハードウェア] を選択してから、[新規デバイスを追加] - [インスタンス ストレージ] の順に選択します。 [インスタンス ストレージ] オプションが [仮想ハードウェア] に表示されます。

2 [インスタンス ストレージ] オプションを展開して、インスタンス ストレージ設定を編集します。

オプション	操作
ストレージ ポリシー	vSAN Direct ストレージ ポリシーを選択します。
ボリューム	ボリュームのサイズを指定します。複数のストレージ ボリュームを追加できます。

3 [確認] 画面で詳細を確認して、[完了] をクリックします。



4 インスタンス ストレージ仮想マシンに使用する名前空間に、作成した仮想マシン クラスを割り当てます。

vSphere Client を使用した仮想マシン クラスと名前空間の関連付けを参照してください。

インスタンス ストレージを使用した仮想マシンのデプロイ

DevOps エンジニアは、インスタンス ストレージ仮想マシンの作成に必要な仮想マシン リソースにアクセスできることを確認します。リソースを使用して仮想マシンをデプロイします。

インスタンス ストレージ仮想マシンをデプロイする場合は、一般的な仮想マシンのデプロイ手順に従います。vSphere IaaS control plane でのスタンドアローン仮想マシンのデプロイを参照してください。この手順では、インスタンス ストレージ仮想マシンに適用されるその他の特定の項目について説明します。

手順

- インスタンス ストレージ仮想マシンに固有の次の項目を確認します。
  - 名前空間に、vSAN Direct データストアと互換性のあるストレージ クラスが含まれます。
  - インスタンス ストレージ仮想マシン クラスは、このストレージ クラスを参照します。

    インスタンス ストレージ仮想マシン クラスの詳細を確認するときは、`instanceStorage` セクションが含まれていることを確認します。

`kubectl describe virtualmachineclasses` *vm-class-instance-storage*

```
apiVersion: vmoperator.vmware.com/v1alpha2
kind: VirtualMachineClass
metadata:
  name: vm-class-instance-storage
spec:
  hardware:
    cpus: 8
    memory: 64Gi
    devices:
...
    instanceStorage:
      storageClass: vsan-direct
      volumes:
      - size: 256Gi
      - size: 512Gi

...
```

- 仮想マシン YAML ファイルは、適切なインスタンス ストレージ仮想マシン クラスを参照します。

vSphere IaaS control plane での構成可能な OVF プロパティを使用した仮想マシンのデプロイ

通常、DevOps エンジニアが vSphere IaaS control plane 環境で仮想マシンをプロビジョニングする場合、OVF テンプレートには基本的なネットワーク構成などのハードコーディングされた詳細が含まれます。ただし、多くの場合、仮想マシンの CR が作成されるまで、IP アドレス管理によって提供されるネットワーク データなどの特定の値を仮想マシンの OVF プロパティに割り当てることはできません。テンプレート文字列のサポートにより、ネットワーク情報を事前に把握しておく必要はありません。Golang ベースのテンプレート化を使用して OVF プロパティ値をポピュレートし、仮想マシンのネットワーク スタックを構成できます。

手順

1 構成するすべてのプロパティについて、OVF ファイルに `ovf:userConfigurable="true"` エントリが含まれていることを確認します。

このエントリにより、ネーム サーバや管理 IP アドレスなどのネットワーク値のプレースホルダは、データ収集後に実際のデータに置き換えられます。

次の例を使用します。

```
<Property ovf:key="hostname" ovf:type="string" ovf:userConfigurable="true"
ovf:value="ubuntuguest">
      <Description>Specifies the hostname for the appliance</Description>
</Property>
<Property ovf:key="nameservers" ovf:type="string" ovf:userConfigurable="true"
ovf:value="1.1.1.1, 1.0.0.1">
      <Label>2.2. DNS</Label>
      <Description>A comma-separated list of IP addresses for up to three DNS servers</
Description>
```

```
</Property>
<Property ovf:key="management_ip" ovf:type="string" ovf:userConfigurable="true">
      <Label>2.3. Management IP</Label>
      <Description>The static IP address for the appliance on the Management Port Group in
CIDR format (Eg. ip/subnet mask bits). This cannot be DHCP.</Description>
</Property>
```

2 テンプレート文字列を使用して仮想マシンの YAML ファイルを作成します。

ブートストラップ リソースのテンプレート文字列によって、OVF プロパティ値のポピュレートに必要なデータが収集されます。

次のいずれかの方法を使用して、テンプレート文字列を設定できます。

- vm-operator-api を使用します。

  詳細については、GitHub のページ (https://github.com/vmware-tanzu/vm-operator/blob/main/api/v1alpha2/virtualmachinetempl_types.go) を参照してください。

  サンプルの YAML ファイルを次に示します。

```
apiVersion: vmoperator.vmware.com/v1alpha2
kind: VirtualMachine
metadata:
  name: template-vm
  namespace: test-ns
  annotations:
    vmoperator.vmware.com/image-supported-check: disable
spec:
  className: best-effort-xsmall
  imageName: vmi-xxxx0000
  powerState: poweredOn
  storageClass: wcpglobal-storage-profile
  vmMetadata:
    configMapName: template-vm-1
    transport: vAppConfig

---
apiVersion: v1
kind: ConfigMap
metadata:
  name: template-vm-1
  namespace: test-ns
data:
  nameservers: "{{ (index .v1alpha2.Net.Nameservers 0) }}"
  hostname: "{{ .v1alpha2.VM.Name }} "
  management_ip: "{{ (index (index .v1alpha2.Net.Devices 0).IPAddresses 0) }}"
  management_gateway: "{{ (index .v1alpha2.Net.Devices 0).Gateway4 }}"
```

- 次の関数を使用します。

関数名	署名	説明
V1alpha2_FirstIP	func () string	最初の NIC から最初の非ループバック IP アドレスを取得します。
V1alpha2_FirstIPFromNIC	func (index int) string	i 番目の NIC から非ループバック IP アドレスを取得します。インデックスが範囲外の場合、テンプレート文字列は解析されません。
V1alpha2_FormatIP	func (IP string, netmask string) string	ネットワークの長さを使用して IP アドレスの書式を設定します。ネットマスクには、長さ（例：/24）または 10 進表記（例：255.255.255.0）のいずれかを使用できます。 入力されたネットマスクが無効な場合や、デフォルトのマスクとは異なる場合、ネットマスクは解析されません。
V1alpha2_FormatNameservers	func (count int, delimiter string) string	特定の区切り文字を使用して、最初に発生したネーム サーバの数の書式を設定します。負の数は、すべてのネーム サーバを意味します。
V1alpha2_IP	func(IP string) string	デフォルトのネットマスク CIDR を使用して固定 IP アドレスの書式を設定します。 IP アドレスが無効な場合、テンプレート文字列は解析されません。
V1alpha2_IPsFromNIC	func (index int) []string	i 番目の NIC からのすべての IP アドレスを一覧表示します。 インデックスが範囲外の場合、テンプレート文字列は解析されません。

関数を使用すると、YAML ファイルは次のようになります。

```
apiVersion: vmoperator.vmware.com/v1alpha2
kind: VirtualMachine
metadata:
  name: template-vm
  namespace: test-ns
spec:
  className: best-effort-xsmall
  imageName: vmi-xxxx0000
  powerState: poweredOn
  storageClass: wcpglobal-storage-profile
  vmMetadata:
    configMapName: template-vm-2
    transport: vAppConfig

---
apiVersion: v1
kind: ConfigMap
metadata:
  name: template-vm-2
  namespace: test-ns
data:
```

```
nameservers: "{{ V1alpha2_FormatNameservers 2 \",\" }}"
hostname: "{{ .v1alpha2.VM.Name }} "
management_ip: "{{ V1alpha2_FormatIP \"192.168.1.10\" \"255.255.255.0\" }}"
management_gateway: "{{ (index .v1alpha2.Net.Devices 0).Gateway4 }}"
```

3 仮想マシンをデプロイします。

```
kubectl apply -f file_name.yaml
```

次のステップ

カスタマイズに失敗して仮想マシンが IP アドレスを取得しない場合は、vSphere 仮想マシン Web コンソールを使用して仮想マシンを検査します。vSphere 仮想マシン Web コンソールを使用した仮想マシンのトラブルシューティングを参照してください。

vSphere IaaS control plane で利用可能な仮想マシンの監視

vSphere 管理者として、vSphere IaaS control plane 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 IaaS control plane でのスタンドアローン仮想マシンのデプロイ」を参照してください。

手順

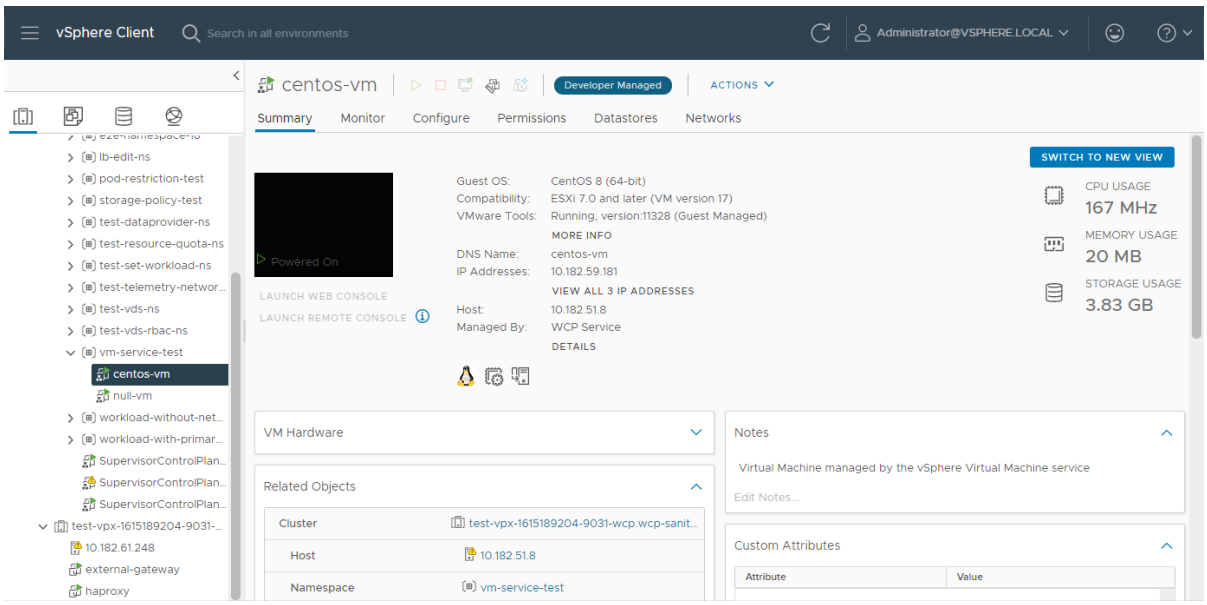
1 vSphere Client で、vSphere IaaS control plane が有効になっているホスト クラスタに移動します。

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。

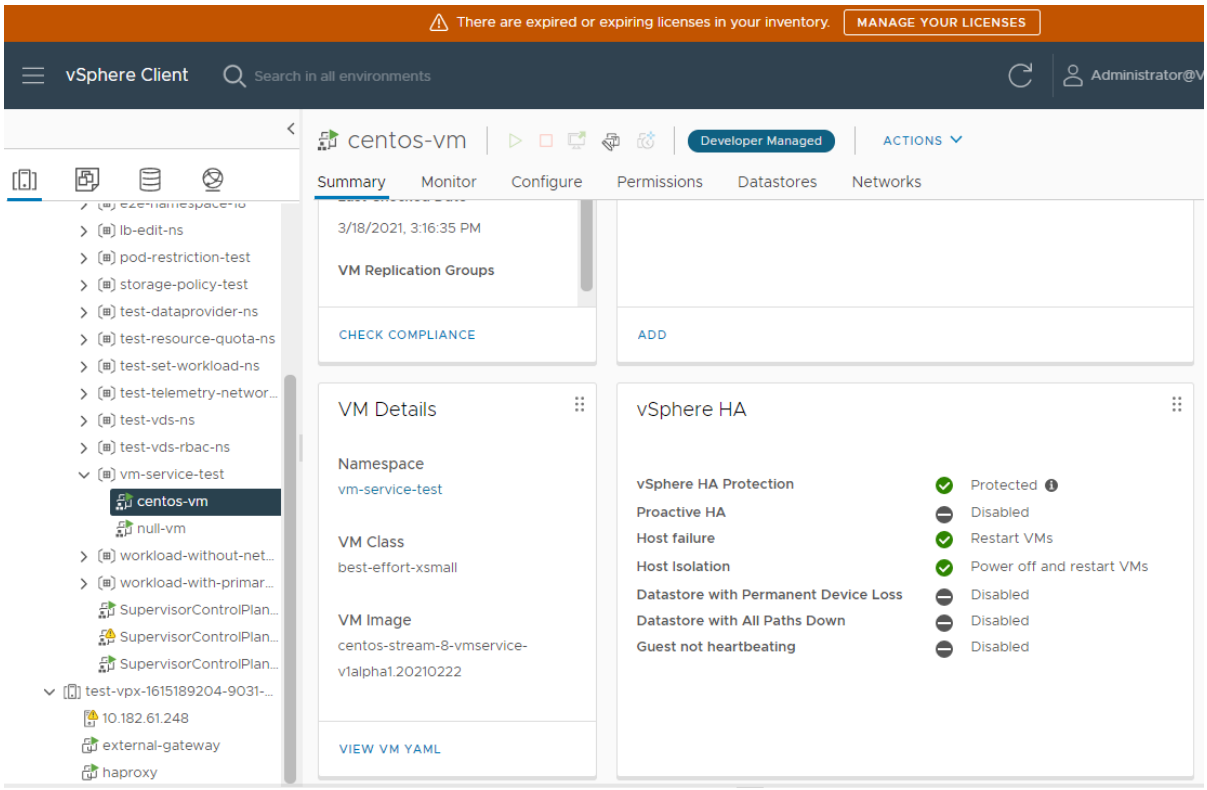
3 表示する仮想マシンを選択し、[サマリ] タブをクリックします。

[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。



vSphere 仮想マシン Web コンソールを使用した仮想マシンのトラブルシューティング

DevOps エンジニアは、vSphere 仮想マシン Web コンソールを使用して、問題のある仮想マシンにアクセスし、トラブルシューティングを行うことができます。仮想マシン Web コンソールを使用すると、通常のネットワークを

介して仮想マシンにアクセスできない場合（たとえば、ゲスト OS が初回起動中に正しいネットワーク設定を構成できなかった場合など）に役立ちます。

仮想マシン Web コンソールは、構成可能な OVF プロパティを持つ仮想マシンをデプロイする場合に特に役立ちます。詳細については、『vSphere IaaS control plane での構成可能な OVF プロパティを使用した仮想マシンのデプロイ』を参照してください。

前提条件

問題のある仮想マシンがデプロイされている名前空間に対する編集権限または所有者権限がある。詳細については、「vSphere IaaS 制御プレーンの ID とアクセス管理」を参照してください。

手順

1 Kubernetes 環境の名前空間にアクセスします。

vSphere IaaS control plane での スーパーバイザー コンテキストの取得と使用を参照してください。

2 仮想マシンが展開されていることを確認します。

```
kubectl get vm -n namspace-name
```

出力は次のようになります。

```
NAME       POWERSTATE         AGE
vm-name    poweredOn          175m
```

3 仮想マシン Web コンソールの URL を取得します。

```
kubectl vsphere vm web-console vm-name -n namspace-name
```

注： コマンドは、認証された URL を仮想マシン Web コンソールに出力として返します。変更不可の期間内に URL を使用しない場合は、2 分に設定します。URL が期限切れになります。URL を開いて Web コンソール ページに接続すると、セッション時間は WebMKS によって制御され、長く存続するようになります。

4 URL をクリックし、仮想マシンに必要なトラブルシューティング アクションを実行します。

vSphere ポッド へのワークロードのデプロイ

7

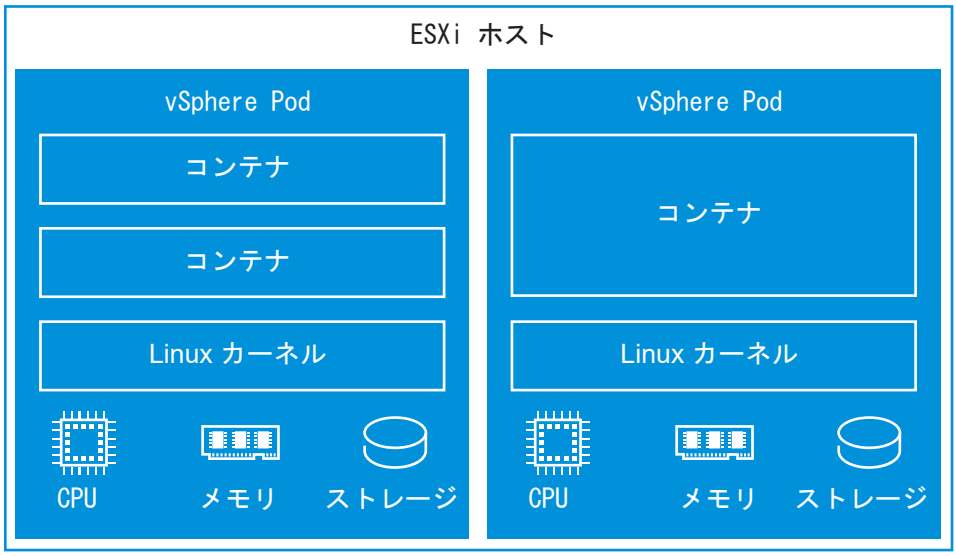
DevOps エンジニアは、スーパーバイザー で実行されている vSphere 名前空間 の境界内で vSphere ポッド をデプロイし、そのライフサイクルを管理できます。

注： vSphere ポッド は、NSX 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 にのみデプロイできます。Distributed Switch スタックが構成された スーパーバイザー に vSphere ポッドをデプロイすることはできません。スーパーバイザー サービス は、NSX または Distributed Switch が構成された両方の スーパーバイザー でサポートされ、自身が使用する vSphere ポッド をデプロイします。ただし、Distributed Switch が構成されている スーパーバイザー での一般的な使用を目的として vSphere ポッド をデプロイすることはできません。

vSphere ポッド について

vSphere IaaS control plane では、Kubernetes ポッドに相当する vSphere ポッド と呼ばれる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 を使用して構成された スーパーバイザー でのみサポートされます。

図 7-1. vSphere ポッド



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

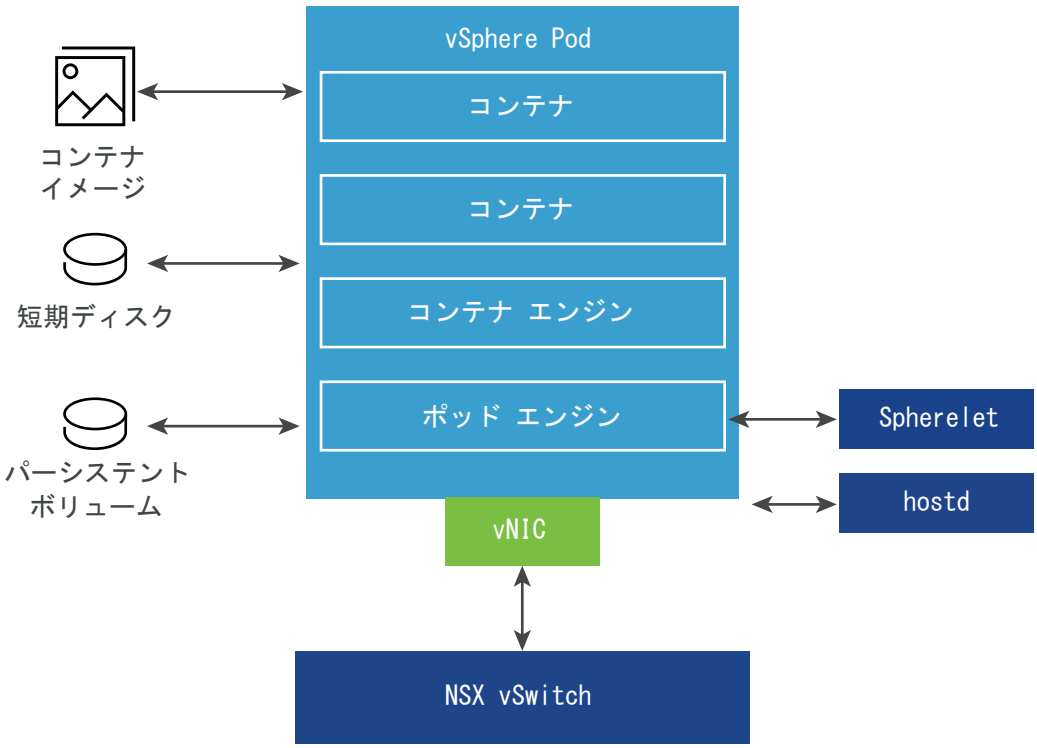
- 高レベルの隔離。vSphere ポッド は、仮想マシンと同じ方法で隔離されます。各 vSphere ポッド には、Photon OS で使用されるカーネルに基づく独自の Linux カーネルがあります。vSphere ポッド では、ベアメタル構成と異なり、多くのコンテナでカーネルを共有することはありません。コンテナごとに一意の Linux カーネルがあります
- リソース管理。vSphere DRS により、スーパーバイザー 上の vSphere ポッド の配置が処理されます。
- 高パフォーマンス。vSphere ポッド では、仮想マシンと同じレベルのリソース隔離が実現するため、高速な起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

vSphere ポッド のデプロイのガイドライン

vSphere ポッド をデプロイする前に、環境が次の要件を満たしていることを確認します。

図 7-2. vSphere ポッド ネットワークおよびストレージ



名前空間

スーパーバイザー には、編集権限または所有者権限を持つ vSphere 名前空間 が構成されている必要があります。vSphere ポッド は、NSX ネットワークを使用する 1 ゾーンの スーパーバイザー のみでサポートされます。

スーパーバイザー を作成するには、「NSX ネットワークを使用する 1 ゾーン スーパーバイザーのデプロイ」を参照してください。

名前空間の作成の詳細については、「スーパーバイザー での vSphere 名前空間 の作成と構成」を参照してください。

権限の割り当ての詳細については、「ID とアクセス管理」を参照してください。

ネットワーク

ネットワークの場合、vSphere ポッド は NSX 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 ネットワーク」を参照してください。

Spherelet は、各ホストで作成される追加のプロセスです。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。

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 を有効にする場合、コンテナ イメージ キャッシュと短期 VMDK を配置するためのストレージ ポリシーを構成します。

vSphere 名前空間 レベルでは、パーシステント ボリュームを配置するためのストレージ ポリシーを構成します。vSphere IaaS control plane におけるストレージの要件と概念の詳細については、8 章 vSphere IaaS control plane の スーパーバイザー ワークロード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。

次のトピックを参照してください。

- vSphere IaaS control plane での スーパーバイザー 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
- 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
- vSphere IaaS control plane での vSphere ポッド ワークロードのデプロイ

vSphere IaaS control plane での スーパーバイザー 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 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 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。vSphere IaaS control plane では、コンテキストは スーパーバイザー での名前空間に対応します。

スーパーバイザー 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 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。
- Kubernetes CLI Tools for vSphere のダウンロードとインストール。『vSphere IaaS 制御プレーンのインストールと構成』ドキュメントを参照してください。
- 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 IaaS 制御プレーンのインストールと構成』ドキュメントの「vSphere IaaS 制御プレーン 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 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 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

Tanzu Kubernetes Grid クラスタに接続するには、『vSphere IaaS 制御プレーンでの TKG サービスの使用』の vCenter Single Sign-On ユーザーとして TKG クラスタへの接続の説明を参照してください。

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

vSphere IaaS control plane では、vSphere 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 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。

前提条件

- vSphere 管理者から スーパーバイザー 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 Kubernetes 環境の名前空間にアクセスします。

vSphere IaaS control plane での スーパーバイザー コンテキストの取得と使用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
```

vSphere ポッド アプリケーションのスケーリング

vSphere IaaS control plane では、スーパーバイザー 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 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 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 IaaS control plane を使用すると、スーパーバイザー 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。

機密性の高い vSphere ポッド を作成するには、追加の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。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、『vSphere のセキュリティ』ドキュメントの 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。
- 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 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。

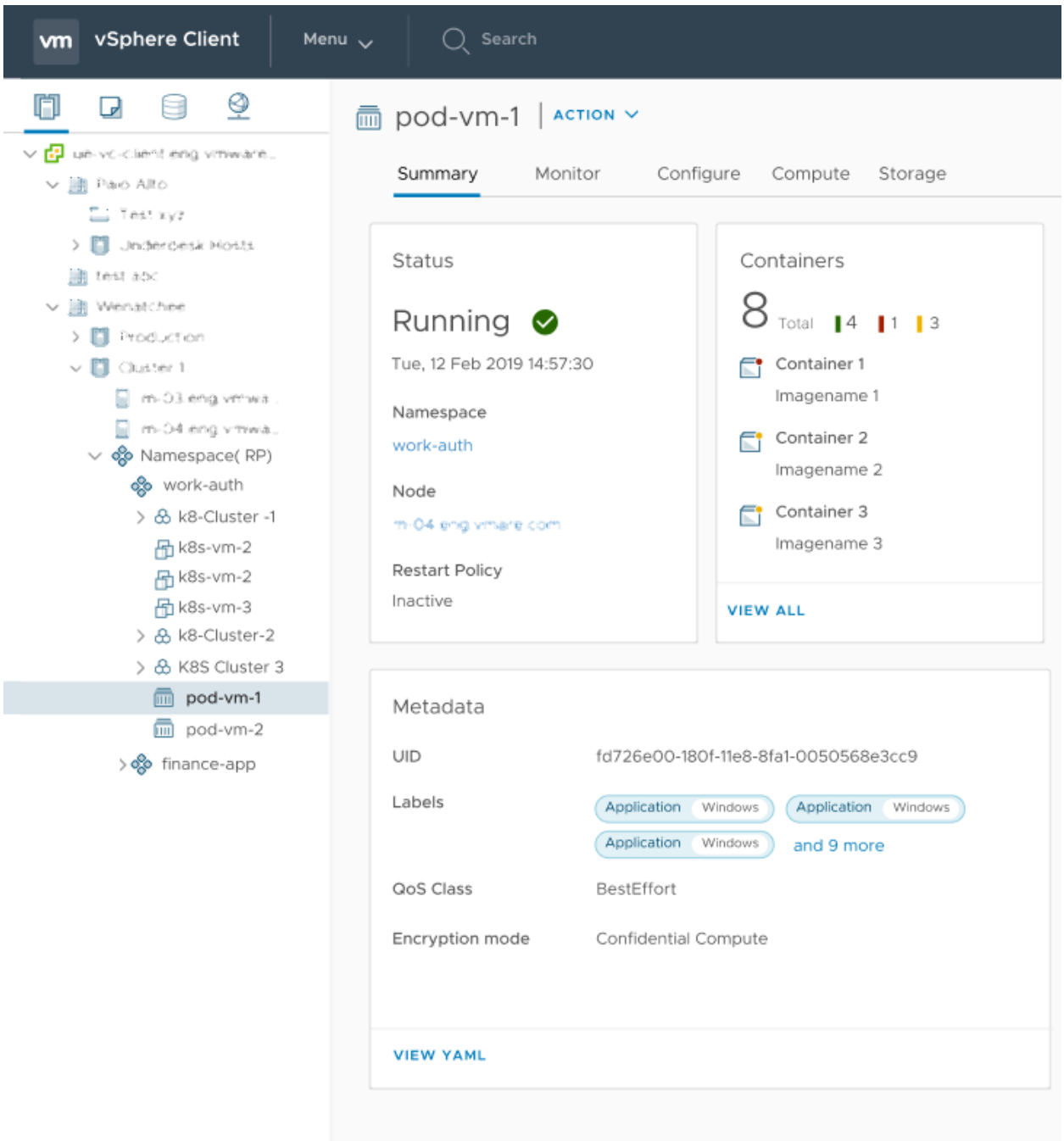
機密性の確保された vSphere ポッド を起動すると、このポッドで実行されているすべてのワークロードに対して、ハードウェア メモリ暗号化がサポートされます。

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 IaaS control plane での vSphere ポッド ワークロードのデプロイ

このチュートリアルの例では、vSphere IaaS control plane 環境の vSphere ポッド を使用して WordPress アプリケーションをデプロイする方法について説明します。

WordPress のデプロイには、WordPress フロントエンドと MySQL バックエンド、そして両サービスをまとめたコンテナが含まれます。シークレット オブジェクトも必要です。

このチュートリアルでは、デプロイ オブジェクトを使用します。本番環境では、WordPress コンテナと MySQL コンテナの両方に StatefulSets を使用するのが一般的です。

前提条件

- NSX ネットワークを使用して、1 ゾーンの スーパーバイザー を作成します。vSphere ポッド は、NSX を使用する 1 ゾーンの スーパーバイザー のみでサポートされます。「NSX ネットワークを使用した 1 ゾーン スーパーバイザーのデプロイ」を参照してください。
- vSphere ポッド をデプロイするための名前空間を作成します。スーパーバイザー での vSphere 名前空間 の作成と構成を参照してください。
- ストレージ ポリシー（例：`vwt-storage-policy`）を作成して名前空間に割り当てます。
- vSphere Kubernetes CLI ツールをダウンロードします。「Kubernetes CLI Tools for vSphere のダウンロードとインストール」を参照してください。
- このチュートリアルに必要な YAML ファイルを作成し、ファイルへのコマンド ライン アクセスを確認します。

カテゴリ	ファイル
ストレージ	■ mysql-pvc.yaml ■ wordpress-pvc.yaml **注：** ファイルが正しいストレージ クラスを参照していることを確認します。
シークレット	■ regcred.yaml ■ mysql-pass.yaml
サービス	■ mysql-service.yaml ■ wordpress-service.yaml
展開	■ mysql-deployment-vsphere-pod.yaml ■ wordpress-deployment.yaml

WordPress のデプロイ

このワークフローを使用して、vSphere ポッド で WordPress アプリケーションをデプロイします。

パート 1。名前空間へのアクセス

名前空間にアクセスするには、次の手順を使用します。

手順

1 Supervisor にログインします。

```
kubectl vsphere login --server=SVC-IP-ADDRESS --vsphere-username wcp-user@vsphere.local
```

2 vSphere 名前空間に切り替えます。

```
kubectl config use-context VSPHERE-PODS-NAMESPACE
```

3 作成したストレージ ポリシー (`vwt-storage-policy`) がストレージ クラスとして名前空間で使用可能であることを確認します。

vSphere 名前空間 でのストレージ クラスの表示を参照してください。

パート 2。WordPress PVC の作成

WordPress PVC を作成するには、次のコマンドを使用します。

手順

1 MySQL PVC を作成します。

```
kubectl apply -f mysql-pvc.yaml
```

2 WordPress PVC を作成します。

```
kubectl apply -f wordpress-pvc.yaml
```

3 PVC を確認します。

```
kubectl get pvc,pv
```

パート 3。シークレットの作成

パブリック Docker Hub は、Kubernetes のデフォルトのコンテナ レジストリです。Docker Hub でイメージ プルが制限されるようになりました。有償アカウントを使用していること、および `data.dockerconfigjson` フィールドのシークレット YAML にアカウント キーを追加することが必要です。

手順

1 Docker Hub レジストリ シークレットを作成します。

```
kubectl apply -f regcred.yaml
```

2 mysql パスワード シークレットを作成します。

MySQL DB のパスワードは必須です。シークレット内では、パスワードは base64 でエンコードする必要があります。

```
kubectl apply -f mysql-pass.yaml
```

3 シークレットを確認します。

```
kubectl get secrets
```

パート 4。サービスの作成

サービスを作成するには、次の手順を実行します。

手順

1 MySQL サービスを作成します。

```
kubectl apply -f mysql-service.yaml
```

2 WordPress サービスを作成します。

```
kubectl apply -f wordpress-service.yaml
```

3 サービスを確認します。

```
kubectl get services
```

パート 5。ポッドのデプロイの作成

このタスクを使用して、ポッドのデプロイを作成します。

このチュートリアルでは、デプロイ オブジェクトを使用します。本番環境では、WordPress コンテナと MySQL コンテナの両方に StatefulSets を使用する必要があります。

手順

1 MySQL デプロイを作成します。

```
kubectl apply -f mysql-deployment-vsphere-pod.yaml
```

注： vSphere ポッド が作成されると、ポッド内にコンテナ用の仮想マシンが作成されます。デフォルトでは、仮想マシンの RAM には 512 MB の制限があります。MySQL コンテナには、より多くのメモリが必要です。ポッドのデプロイ仕様 `mysql-deployment-vsphere-pod.yaml` には、vSphere ポッド 仮想マシンに指定されたメモリを増やすセクションが含まれています。このセクションを含めない場合は、メモリ不足 (OOM) 例外が発生してポッドのデプロイは失敗します。MySQL ポッドを TKG クラスタにデプロイするときに RAM を増やす必要はありません。

2 WordPress デプロイを作成します。

```
kubectl apply -f wordpress-deployment.yaml
```

3 デプロイを確認します。

```
kubectl get deployments
```

パート 6。WordPress のテスト

WordPress のデプロイをテストするには、次の手順を実行します。

手順

1 すべてのオブジェクトが作成され、実行されていることを確認します。

```
kubectl get pv,pvc,secrets,rolebinding,services,deployments,pods
```

2 WordPress サービスから EXTERNAL-IP アドレスを取得します。

```
kubectl get service wordpress

NAME        TYPE           CLUSTER-IP    EXTERNAL-IP     PORT(S)        AGE
wordpress   LoadBalancer   10.96.9.180   10.197.154.73   80:30941/TCP   87s
```

3 EXTERNAL-IP アドレスを参照します。

4 WordPress インスタンスを構成します。

ユーザー名：administrator

パスワード：指定された強力なパスワードを使用

WordPress 展開のサンプル YAML ファイル

vSphere ポッド を使用して WordPress アプリケーションをデプロイする場合は、次のサンプル YAML ファイルを使用します。

mysql-pvc.yaml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mysql-pvc
  labels:
    app: wordpress
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: vwt-storage-policy
  resources:
    requests:
      storage: 20Gi
```

wordpress-pvc.yaml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: wordpress-pvc
  labels:
    app: wordpress
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: vwt-storage-policy
  resources:
    requests:
      storage: 20Gi
```

regcred.yaml

```
apiVersion: v1
kind: Secret
metadata:
  name: regcred
data:
  .dockerconfigjson: ewoJImF1dGhzIjog....zZG1KcE5WUmtXRUozWpc
type: kubernetes.io/dockerconfigjson
```

mysql-pass.yaml

```
apiVersion: v1
data:
  password: YWRtaW4=  #admin base64 encoded
kind: Secret
metadata:
  name: mysql-pass
```

mysql-service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: wordpress-mysql
  labels:
    app: wordpress
spec:
  ports:
    - port: 3306
  selector:
    app: wordpress
    tier: mysql
  clusterIP: None
```

wordpress-service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: wordpress
  labels:
    app: wordpress
spec:
  ports:
    - port: 80
  selector:
    app: wordpress
    tier: frontend
  type: LoadBalancer
```

mysql-deployment-vsphere-pod.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wordpress-mysql
  labels:
    app: wordpress
spec:
  replicas: 1
  strategy:
    type: Recreate
  selector:
    matchLabels:
      app: wordpress
      tier: mysql
  template:
    metadata:
      labels:
        app: wordpress
        tier: mysql
    spec:
      containers:
      - image: mysql:5.6
        name: mysql
        #increased resource limits required for this pod vm
        #default pod VM RAM is 512MB; MySQL container needs more
        #without extra RAM OOM error prevents deployment
        #extra RAM not required for Kuberentes cluster
        resources:
          limits:
            memory: 1024Mi
            cpu: 1
        env:
        - name: MYSQL_ROOT_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mysql-pass
              key: password
        ports:
        - containerPort: 3306
          name: mysql
        volumeMounts:
        - name: mysql-persistent-storage
          mountPath: /var/lib/mysql
      volumes:
      - name: mysql-persistent-storage
        persistentVolumeClaim:
          claimName: mysql-pvc
      imagePullSecrets:
      - name: regcred
```

wordpress-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wordpress
  labels:
    app: wordpress
spec:
  selector:
    matchLabels:
      app: wordpress
      tier: frontend
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: wordpress
        tier: frontend
    spec:
      containers:
      - image: wordpress:4.8-apache
        name: wordpress
        env:
        - name: WORDPRESS_DB_HOST
          value: wordpress-mysql
        - name: WORDPRESS_DB_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mysql-pass
              key: password
        ports:
        - containerPort: 80
          name: wordpress
        volumeMounts:
        - name: wordpress-persistent-storage
          mountPath: /var/www/html
      volumes:
      - name: wordpress-persistent-storage
        persistentVolumeClaim:
          claimName: wordpress-pvc
      imagePullSecrets:
      - name: regcred
```

vSphere IaaS control plane の スーパーバイザー ワークロードでのパーシステント ストレージの使用

8

DevOps が スーパーバイザー の名前空間で実行する Kubernetes ワークロードによっては、データを永続的に保存するためのパーシステント ストレージが必要になることがあります。パーシステント ストレージは、vSphere ポッド、Tanzu Kubernetes Grid クラスタ、仮想マシン、および名前空間で実行するその他のワークロードで使用できます。

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 クラスを記述するストレージ ポリシーを作成します。次に、管理者はストレージ ポリシーを割り当て、名前空間レベルでストレージ制限を構成します。

vSphere IaaS control plane がパーシステント ストレージとどのように連携するかを理解するには、ストレージ クラス、パーシステント ボリューム、パーシステント ボリュームの要求など、Kubernetes の基本的な概念について理解しておく必要があります。詳細については、Kubernetes のドキュメント (https://kubernetes.io/docs/home/) を参照してください。

vSphere IaaS control plane コンポーネントとストレージの統合方法については、vSphere IaaS 制御プレーンの概念と計画のスーパーバイザー ストレージを参照してください。

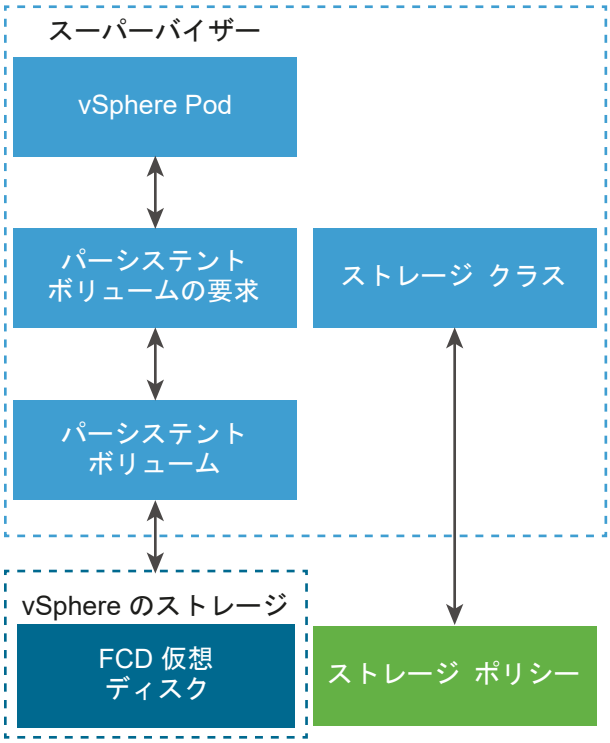
パーシステント ストレージのワークフロー

vSphere IaaS control plane 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。

操作	実行者	説明
DevOps チームにパーシステント ストレージ リソースを提供する	vSphere 管理者	vSphere 管理者は、さまざまなストレージ要件およびサービス クラスを記述するストレージ ポリシーを作成します。 vSphere IaaS 制御プレーンのインストールと構成ドキュメントにある vSphere IaaS 制御プレーンのストレージ ポリシーの作成を参照してください。 次に、管理者はストレージ ポリシーを名前空間に割り当て、名前空間のストレージ制限を設定します。 スーパーバイザー での vSphere 名前空間 の作成と構成を参照してください。
名前空間にストレージ クラスを作成する	vSphere IaaS control plane	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、Kubernetes 環境に自動的に表示されます。vSphere 管理者が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 を使用する場合、各クラスタは、このクラスタがプロビジョニングされている名前空間からストレージ クラスを継承します。 DevOps チームは、パーシステント ストレージのニーズに応じてストレージ クラスを使用できます。 vSphere 名前空間 でのストレージ クラスの表示を参照してください。
ワークロードのパーシステント ストレージ リソースを要求する	DevOps	DevOps チームがストレージ 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。 vSphere IaaS control plane での動的パーシステント ボリュームのプロビジョニングと vSphere IaaS control plane でのスタンドアローン仮想マシンのデプロイを参照してください。

操作	実行者	説明
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を作成する	vSphere IaaS control plane	vSphere IaaS control plane 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。
パーシステント ボリュームを監視する	vSphere 管理者	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 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。

次に、vSphere ポッド 用のパーシステント 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 FCD 仮想ディスクがどのように作成されるかを示します。パーシステント ストレージの要求は、特定のストレージ クラスを参照します。



次のトピックを参照してください。

- vSphere 名前空間 でのストレージ クラスの表示
- vSphere IaaS control plane での動的パーシステント ボリュームのプロビジョニング
- vSphere IaaS control plane 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
- vSAN ファイル サービスを使用した vSphere IaaS control plane での ReadWriteMany ボリュームの作成

- vSphere IaaS control plane 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 Grid クラスタでのボリュームの健全性の監視
- 3 ゾーン スーパーバイザー でのパーシステント ストレージの使用に関するベスト プラクティス

vSphere 名前空間 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 IaaS control plane の vSphere 名前空間 に割り当てると、そのストレージ ポリシーは一致する Kubernetes ストレージ クラスとして vSphere 名前空間 に表示されます。また、使用可能な Tanzu Kubernetes Grid クラスタにもレプリケートされます。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
```

vSphere IaaS control plane で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 ボリュームが必要です。vSphere IaaS control plane を使用すると、アプリケーションのパーシステント ボリュームを動的にプロビジョニングできます。

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 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 Grid 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。

vSphere IaaS control plane での スーパーバイザー コンテキストの取得と使用を参照してください。

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 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。vSAN ファイル サービスを使用した vSphere IaaS control plane での ReadWriteMany 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 Grid 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。

vSphere IaaS control plane 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 からパーシステント ボリューム要求 (PVC) を使用して、Tanzu Kubernetes Grid 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 Grid 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 の vSphere ポッド または Tanzu Kubernetes Grid 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 Grid クラスタで不要になった PVC を新しい Tanzu Kubernetes Grid 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 Grid 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 Grid 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 内の元の PVC の名前を書き留めます。

古い Tanzu Kubernetes Grid クラスタの PVC を再利用する場合は、Tanzu Kubernetes Grid 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 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 Grid クラスタのボリュームを再利用する場合は、新しい Tanzu Kubernetes Grid クラスタで PV を作成する前に、古い Tanzu Kubernetes Grid 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.
```

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
```

vSAN ファイル サービスを使用した vSphere IaaS control plane での ReadWriteMany ボリュームの作成

vSphere IaaS control plane 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 のサポートにより、1 つの TKG クラスタ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で 1 つのボリュームを同時にマウントできます。vSphere IaaS control plane は、ReadWriteMany パーシステント ボリュームに、vSAN ファイル共有によってバッキングされる CNS ファイル ボリュームを使用します。vSAN 共有を使用するには、vSAN 環境で vSAN ファイル サービスを設定し、スーパーバイザー でファイル ボリュームのサポートを有効にする必要があります。

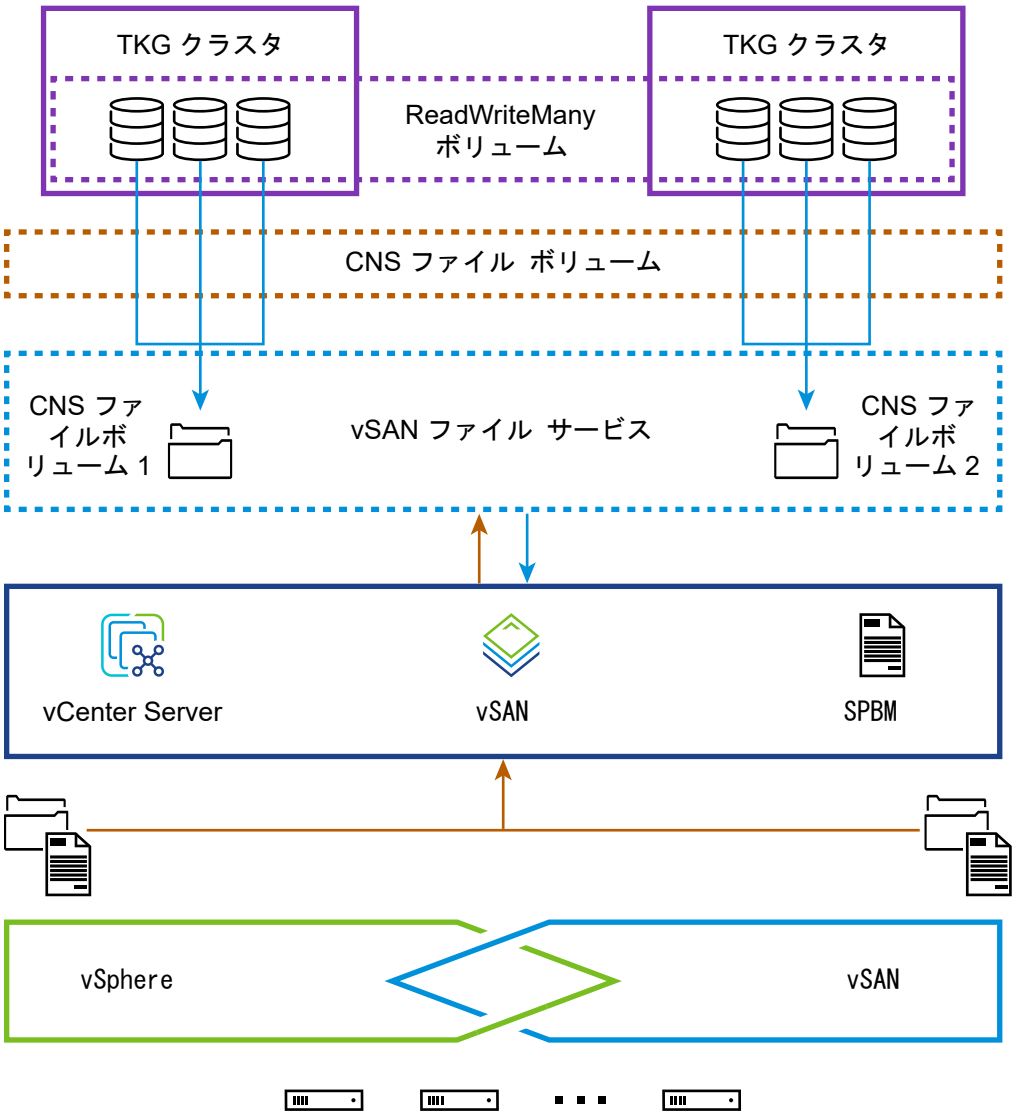
ファイル ボリュームに関する考慮事項

vSphere IaaS control plane でパーシステント ボリュームに対するファイル ボリュームの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。

- ファイル ボリュームがサポートされるのは、Tanzu Kubernetes Grid クラスタ内のワークロードのみです。スーパーバイザー 名前空間内の vSphere ポッド や仮想マシン サービス仮想マシンなどのワークロードではサポートされません。
- Kubernetes で RWX ボリュームを要求すると、vSAN ファイル サービスにより、要求されたサイズおよび適切な SPBM ポリシーを持つ NFS ベースのファイル共有が作成されます。RWX ボリュームごとに vSAN ファイル共有が 1 つ作成されます。VMware は、vSAN ファイル サービス クラスタごとに 100 個の共有をサポートします。つまり、使用できる RWX ボリュームは 100 個までです。
- TKG クラスタでは、TKr バージョン 1.22 以降を使用します。

  詳細については、VMware Tanzu Kubernetes リリースのリリース ノートを参照してください。

- vSphere IaaS control plane 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 IaaS control plane のネットワークに NSX を使用している場合は、スーパーバイザー 名前空間で NAT モードが有効になっていることを確認します。スーパーバイザー での vSphere 名前空間 の作成と構成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 が設定されていないことを確認します。
  - vSAN ファイル サービスと vSphere IaaS control plane 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 ボリュームでファイル ボリュームのサポートを有効にするためのワークフロー

次のプロセスに従って、ファイル ボリュームの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。

- vSAN ファイル サービスの有効化およびファイル サービスの構成を参照してください。
- vSAN ストレッチ クラスタを使用する環境の固有の設定については、ストレッチ クラスタを使用する vSAN ファイル サービスを参照してください。

2 vSphere 管理者は、スーパーバイザー でファイル ボリュームのサポートを有効にします。

『vSphere IaaS 制御プレーンのインストールと構成』ドキュメントのスーパーバイザーでのストレージ設定の変更を参照してください。

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。

例：

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: gold
  resources:
    requests:
        storage: 3Gi
```

vSphere IaaS control plane でのボリュームの拡張

DevOps エンジニアは、パーシステント ブロック ボリュームの作成後に、このボリュームを拡張できます。vSphere IaaS control plane では、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、スーパーバイザーと Tanzu Kubernetes Grid の両方のタイプのクラスタでサポートされます。

注： 拡張できるのは、パーシステント ブロック ボリュームのみです。現在、vSphere IaaS control plane は、ReadWriteMany ボリュームのボリューム拡張をサポートしていません。

デフォルトでは、vSphere IaaS control plane 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。

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 IaaS control plane 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、スーパーバイザーおよび Tanzu Kubernetes Grid クラスタでそのボリュームのサイズを変更できます。

- StatefulSet の定義を使用する場合、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。現在、Kubernetes ではこの機能がサポートされていません。そのため、StatefulSet 定義内でストレージ サイズを増やしてボリュームを拡張しようとすると失敗します。
- 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 IaaS control plane 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 と Tanzu Kubernetes Grid 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。

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 と Tanzu Kubernetes Grid 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。

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 IaaS control plane 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] タブをクリックし、リストから名前空間を選択します。

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。

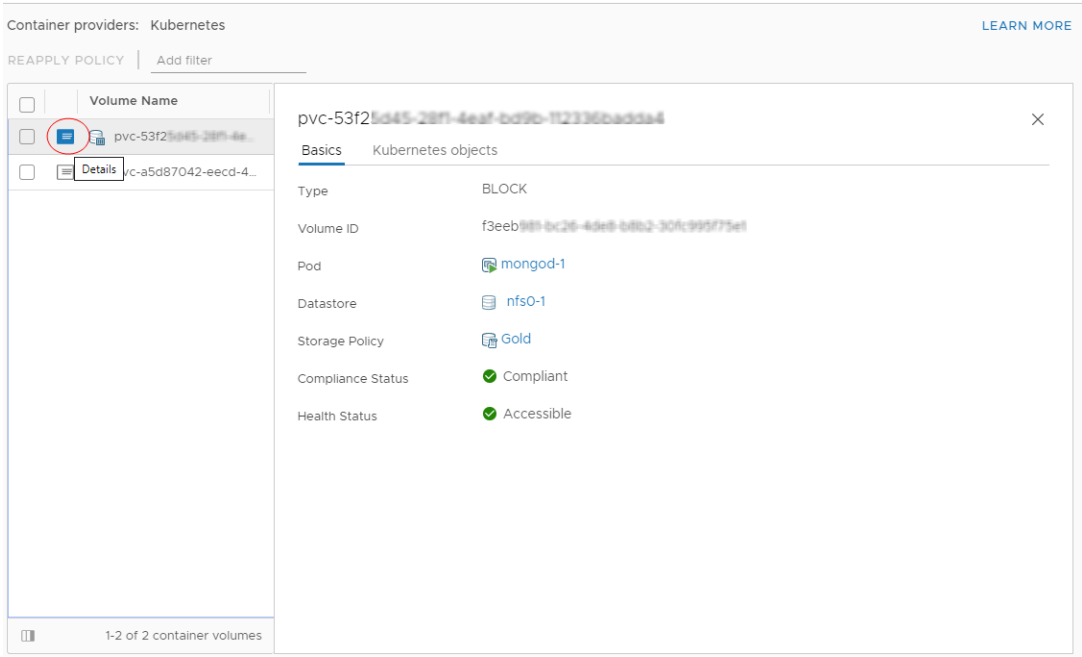
   vSphere Client に、すべてのパーシステント ボリュームの要求オブジェクトと、名前空間内で使用可能な対応するボリュームが一覧表示されます。

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 Grid 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。

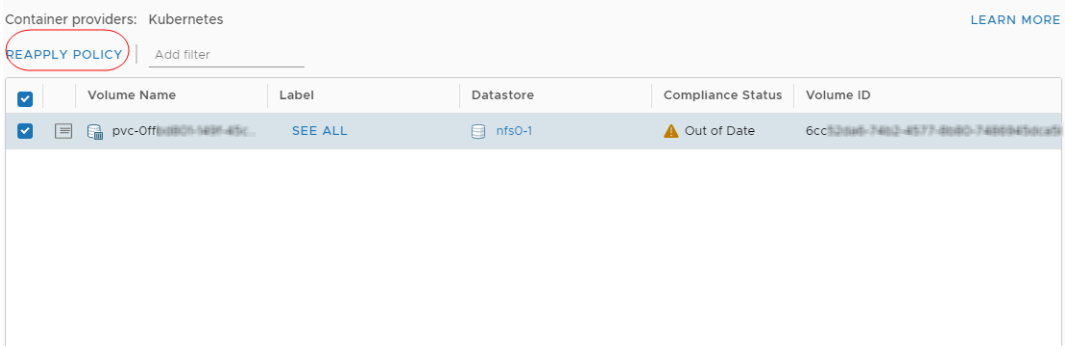
健全性ステータス	説明
アクセスの可能性	パーシステント ボリュームにアクセスして、使用できます。
アクセス不可	パーシステント ボリュームにアクセスしたり、使用したりできません。データストアに接続しているホストからボリュームが格納されているデータストアにアクセスできない場合、パーシステント ボリュームにはアクセスできなくなります。

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 Grid クラスタでのボリュームの健全性の監視

vSphere IaaS control plane を使用している場合、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 IaaS control plane 環境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
```

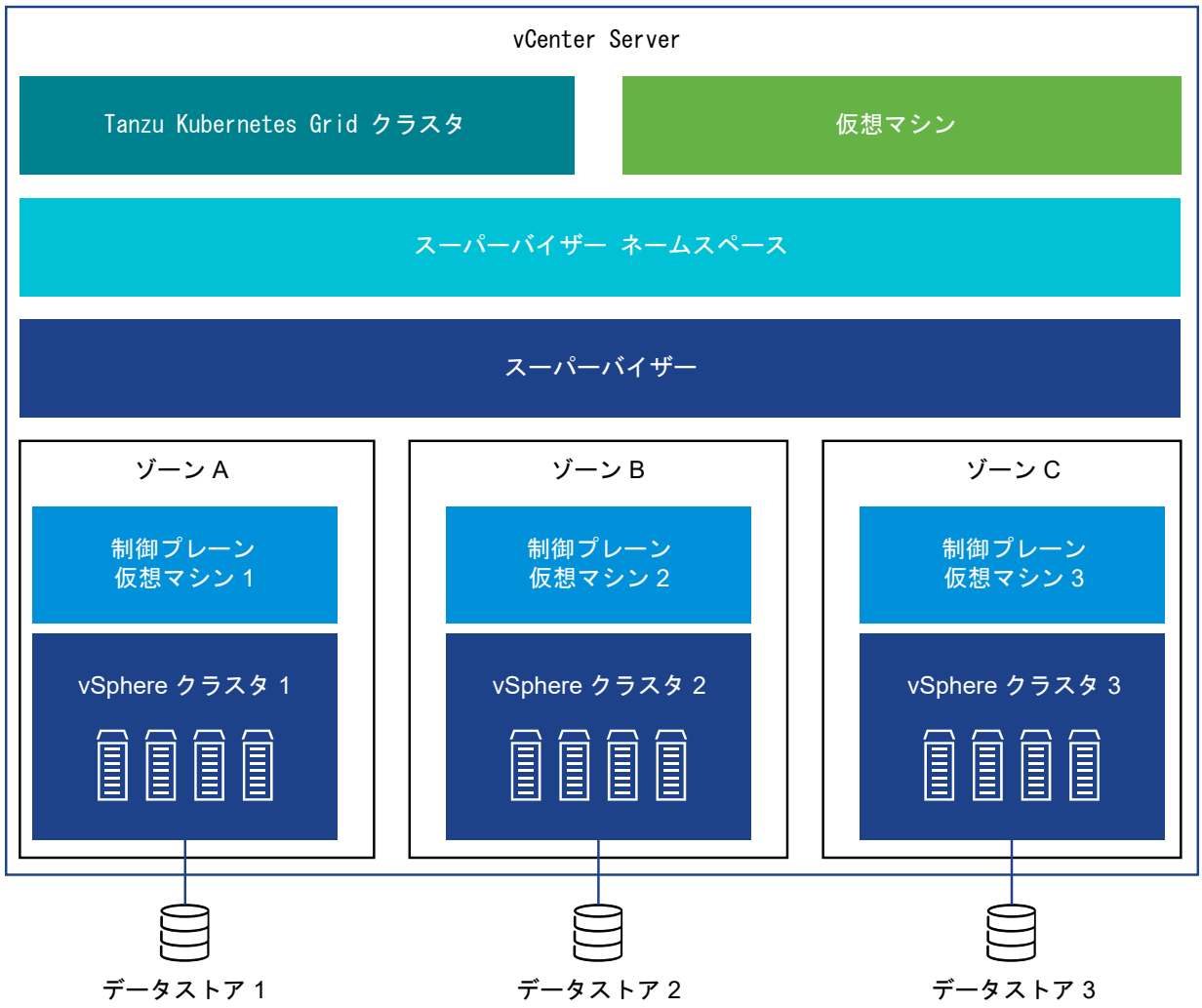
次のサンプル出力では、`volumehealth.storage.kubernetes.io/messages` フィールドに健全性ステータスがアクセス可能と表示されています。

```
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
```

3 ゾーン スーパーバイザー でのパーシステント ストレージの使用に関するベスト プラクティス

vSphere IaaS control plane の 3 ゾーン スーパーバイザー は、1 つのゾーン内のすべてのホスト間でデータストアが共有されるゾーン ストレージをサポートしています。



3 ゾーン スーパーバイザー にストレージ リソースを準備する場合は、次の考慮事項に注意してください。

- 3 つすべてのゾーン内のストレージが同じタイプである必要はありません。ただし、3 つすべてのクラスタで同じタイプのストレージを使用すると、一貫したパフォーマンスが得られます。

- 3 ゾーン スーパーバイザー の名前空間には、各クラスタの共有ストレージに準拠したストレージ ポリシーを使用します。ストレージ ポリシーはトポロジ対応である必要があります。
- 名前空間に割り当てた後は、ストレージ ポリシーからトポロジの制約を削除しないでください。
- ゾーン データストアを他のゾーンにマウントしないでください。
- 3 ゾーン スーパーバイザー では、次のアイテムはサポートされません。
  - ゾーン間ボリューム
  - vSAN ファイル ボリューム (ReadWriteMany ボリューム)
  - ボリューム登録 API を使用した静的ボリュームのプロビジョニング
  - vSAN データ パーシステンス プラットフォームを使用するワークロード
  - vSphere ポッド
  - vSAN ストレッチ クラスタ
  - vGPU とインスタンス ストレージを使用する仮想マシン

3 ゾーン スーパーバイザーのストレージ ポリシーの作成

パーシステント ストレージを使用できるようにするには、3 ゾーン スーパーバイザー で実行されるワークロードから、ゾーン トポロジを持つストレージ クラスにアクセスできる必要があります。これらのストレージ クラスを使用できるようにするために、vSphere 管理者はトポロジ対応のストレージ ポリシーを作成し、名前空間に割り当てます。

3 ゾーン スーパーバイザー の名前空間を使用すると、ユーザーはトポロジに対応しないストレージ ポリシーを割り当てることができなくなります。

3 ゾーン スーパーバイザー を有効にする方法については、3 ゾーン スーパーバイザーの有効化を参照してください。

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。

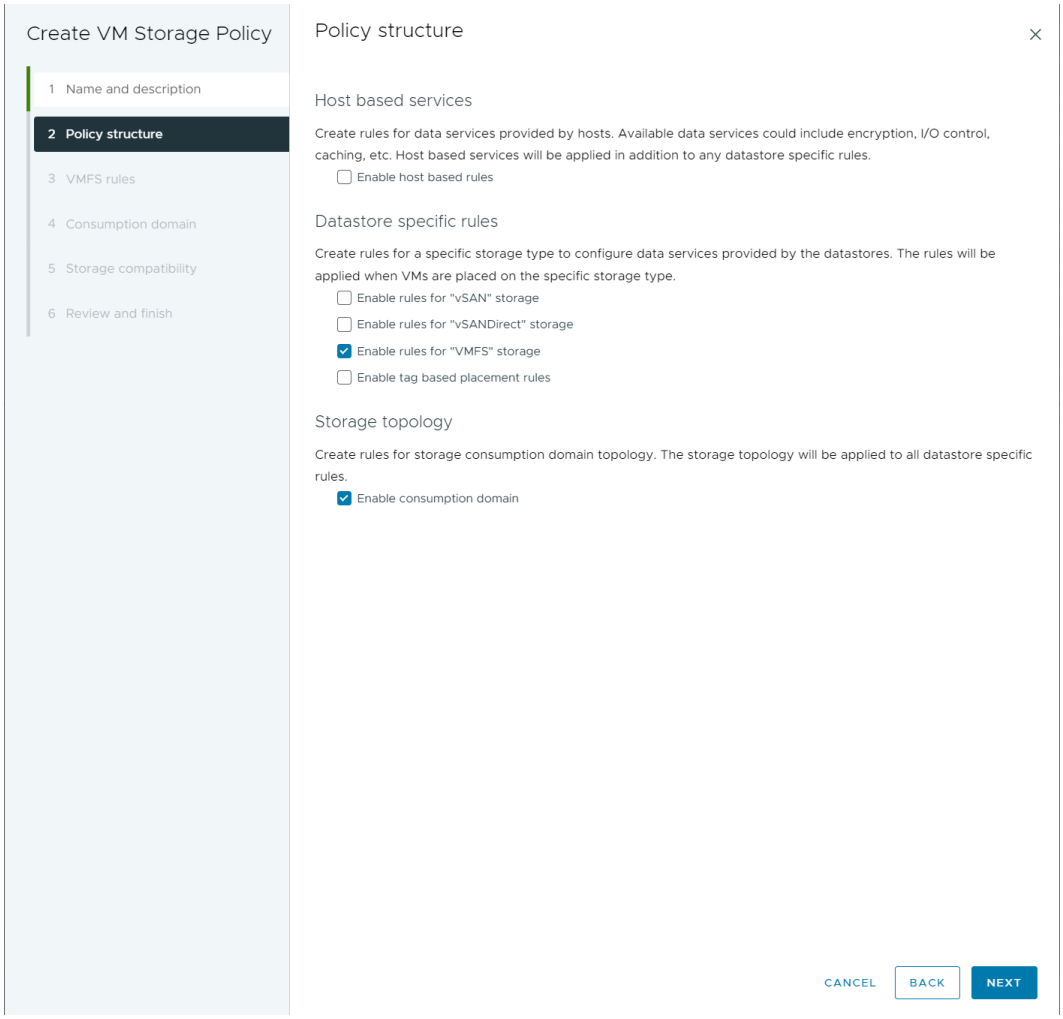
   c [作成] をクリックします。

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 プロンプトの指示に従って、[ポリシー構造] 画面を表示します。

4 [ストレージ トポロジ] で [使用ドメインの有効化] を選択し、プロンプトに従って [使用ドメイン] 画面に移動します。



5 [使用ドメイン] 画面で、ストレージ トポロジのタイプを指定します。

オプション	説明
ゾーン	データストアは、1 つのゾーン内のすべてのホストで共有されます。

3 ゾーン スーパーバイザーでの PVC の作成

3 ゾーン スーパーバイザー で動的 PVC を作成する場合は、ボリュームをプロビジョニングするゾーンを指定します。

手順

- PVC ゾーンの配置を制御するには、PVC YAML ファイルで Kubernetes `csi.vsphere.volume-requested-topology` の注釈を使用します。

  注意： このパラメータは、スーパーバイザー で PVC を直接作成する場合に必要です。ただし、Tanzu Kubernetes Grid クラスタ用に作成する PVC にはゾーンの注釈を含めないでください。この操作を行うと、PVC が機能しません。

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: svcpvc4
  annotations:
    csi.vsphere.volume-requested-topology: '[{"topology.kubernetes.io/zone":"zone-1"},
{"topology.kubernetes.io/zone":"zone-2"},{"topology.kubernetes.io/zone":"zone-3"}]'
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Mi
  storageClassName: zonal2
```

  3 つのゾーンをすべて指定した場合、ボリュームは zone-1、zone-2、または zone-3 のいずれかに作成されます。

次のステップ

Tanzu Kubernetes Grid クラスタにステートフル アプリケーションをデプロイする方法については、遅延バインド ボリューム接続を使用した vSphere Zone 全体への StatefulSet アプリケーションのデプロイを参照してください。

vSphere IaaS control plane での Harbor と Contour のインストールと構成

9

vSphere IaaS control plane 環境で スーパーバイザー サービス として Harbor と Contour をデプロイして構成する方法を確認します。Harbor は、vSphere IaaS control plane で実行されているワークロードで使用可能なオープン ソースのクラウドネイティブ レジストリです。Contour は、Envoy プロキシをリバース プロキシおよびロード バランサとしてデプロイすることで機能する、Kubernetes の Ingress コントローラです。Contour は、軽量なプロファイルを維持しながら、特別な設定が不要な動的構成の更新をサポートします。

Contour をアプリケーションの Ingress コントローラである スーパーバイザー サービス として使用できます。Contour は、Harbor スーパーバイザー サービス を実行するための要件でもあります。

注： スーパーバイザー サービス は、Distributed Switch または NSX ネットワーク スタックのいずれかで実行されている単一クラスタの スーパーバイザー でサポートされます。スーパーバイザー サービス を 3 ゾーンの スーパーバイザー にデプロイすることはできません。

スーパーバイザー サービス として使用される Harbor には、次の機能があります。

- Harbor オープンソース レジストリの最新バージョン。
- admin アカウントと root アカウントを使用した Harbor へのアクセス。
- アップストリーム Harbor レジストリとの完全な機能パリティ。
- DNS を使用する、Ingress (Contour) を介した Harbor へのアクセス。

注： デプロイされると、Harbor と Contour の両方の スーパーバイザー サービス が、これらのサービス用に作成された vSphere 名前空間 内に vSphere ポッド を作成します。これらの vSphere ポッド は、サービスが動作するために必要です。VDS ネットワーク スタックで実行されている スーパーバイザー または 3 ゾーン スーパーバイザー 上の スーパーバイザー サービス の外部に vSphere ポッド をデプロイすることはできません。一般的用途の vSphere ポッド は、NSX を使用してデプロイされた単一クラスタ スーパーバイザー のみにデプロイできます。

次のトピックを参照してください。

- vSphere IaaS control plane での スーパーバイザー サービス としての Contour のインストール
- vSphere IaaS control plane での スーパーバイザー への Harbor のインストールと構成
- vSphere IaaS control plane での組み込みレジストリから Harbor へのイメージの移行

vSphere IaaS control plane での スーパーバイザー サービス としての Contour のインストール

vSphere IaaS control plane 環境の スーパーバイザー に スーパーバイザー サービス として Contour をインストールする方法を確認します。インストール後、Contour をアプリケーションの Ingress コントローラとして使用できます。Contour は、Harbor を スーパーバイザー サービス として実行するための要件でもあります。

前提条件

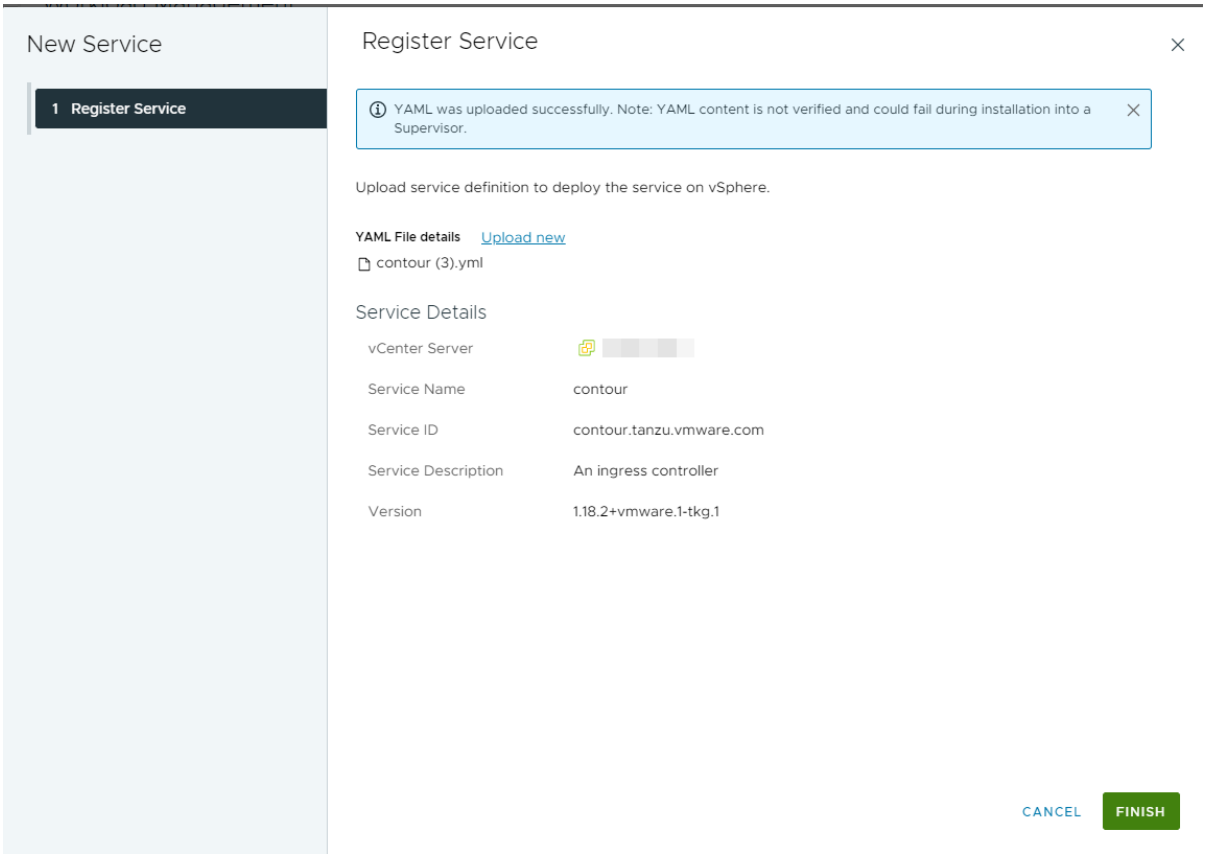
- サービスを追加する vCenter Server システムに関するスーパーバイザー サービスの管理権限があることを確認します。
- vCenter Server 8.0a 以降にアップグレードしたことを確認します。Contour および Harbor の スーパーバイザー サービス は、vCenter Server 8.0a 以降でサポートされます。

手順

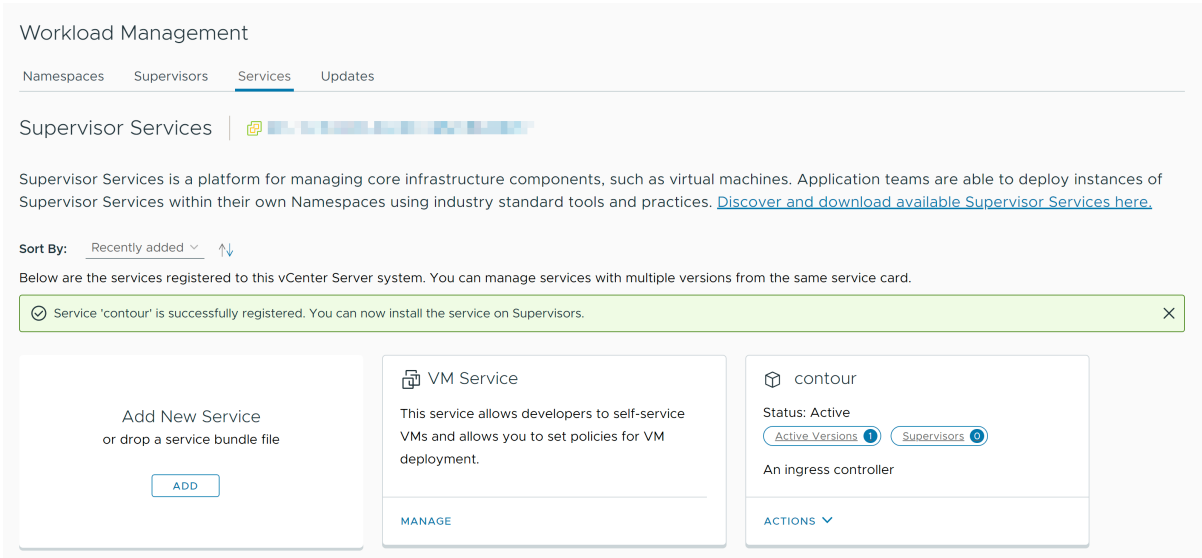
1. Supervisor-Services リポジトリの Contour Versions セクションに移動して、次のファイルをダウンロードします。
   - Contour サービス定義。リンクの名前は `Contour vX.X.X.` です。例：`Contour 1.18.2`
   - Contour 構成ファイル。リンクの名前は `values for vX.X.X` です。例：`values 1.18.2`

   結果ファイルは次のとおりです。
   - `contour.yml`
   - `contour-data-values.yml`
2. vSphere Client で [ワークロード管理] に移動し、[サービス] を選択します。

3 [サービスの新規追加] をクリックし、`contour.yml` サービス定義をアップロードして、Contour のサービス オペレータをデプロイします。



Contour オペレータが正常にデプロイされると、[サービス] タブにサービス カードが表示されます。



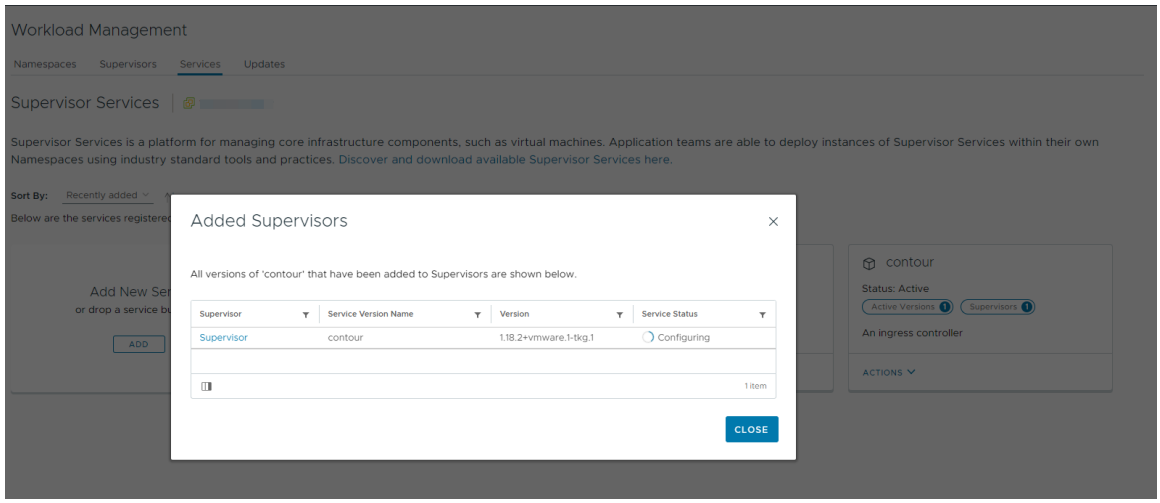
4 これで Contour オペレータがデプロイされたので、スーパーバイザー に スーパーバイザー サービス をインストールできます。

a [Contour] サービス カードで [アクション] - [スーパーバイザーへのインストール] を選択します。

b スーパーバイザー を選択し、[YAML サービス構成] で、`contour-data-values.yml` ファイルの内容を、デフォルト値を変更せずにコピーして貼り付けます。

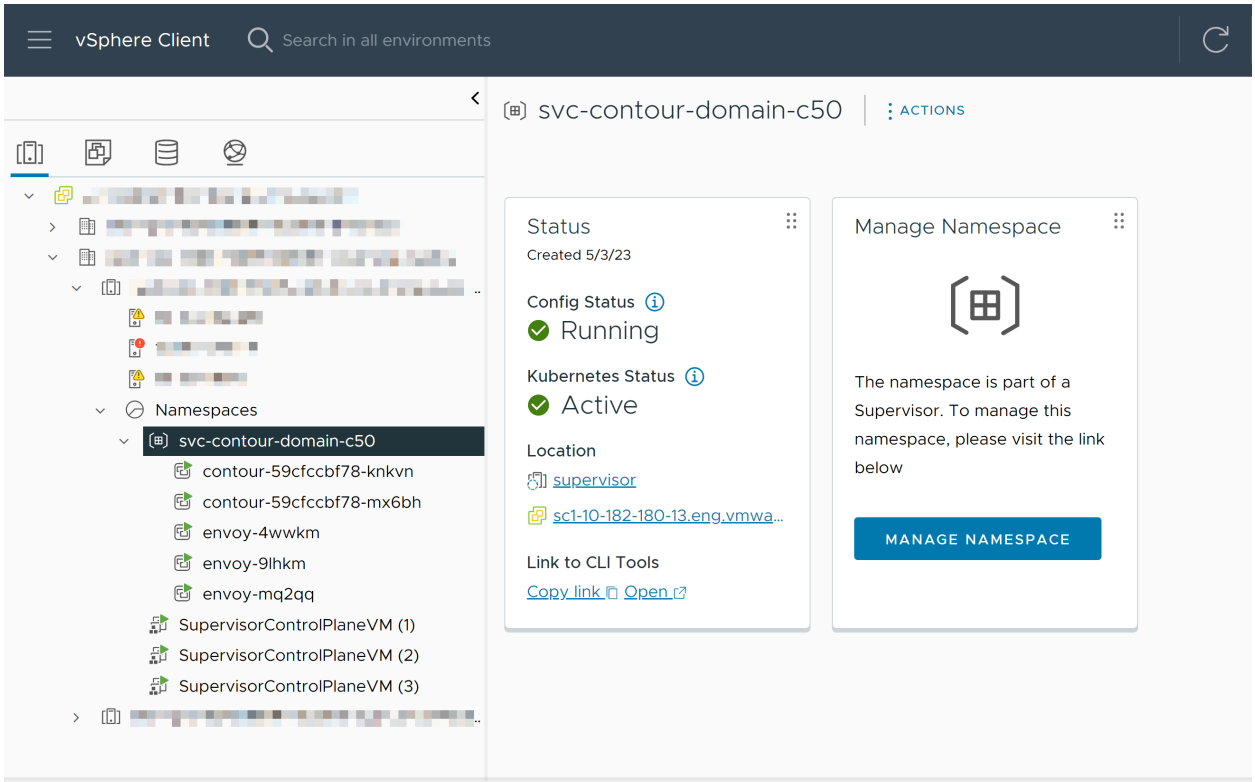
c [OK] をクリックします。

インストールが開始したら、Contour サービス カードの [スーパーバイザー] フィールドをクリックしてインストールを追跡できます。[スーパーバイザー] の横にある数字が増えるまで、数秒かかる場合があります。目的の状態に達するまで、サービスの状態は [設定中] になります。目的の状態に達すると、サービスの状態は [実行中] に変化します。

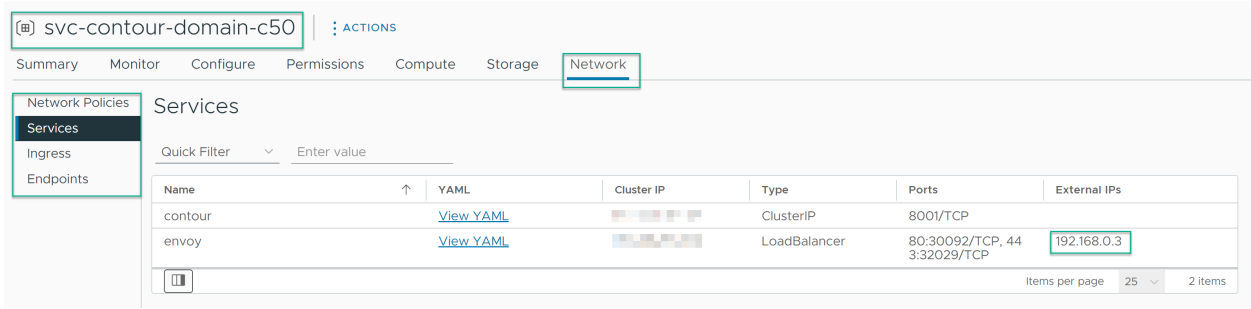


結果

Contour がインストールされると、サービス インスタンス用に作成された vSphere 名前空間 と、対応する vSphere ポッド がデプロイされます。



また、スーパーバイザー を使用して構成される外部 DNS サーバのドメイン名にマッピングできる Envoy サービスの IP アドレスを表示することもできます。マッピングを使用すると、Contour を介してアプリケーションに Ingress を提供できます。Contour vSphere 名前空間 の [ネットワーク] オプションの Envoy IP アドレスを表示できます。



次のステップ

ワークロードのレジストリを提供するために Harbor を スーパーバイザー サービス として使用する場合は、サービスをインストールし、ワークロードで使用するように構成できます。vSphere IaaS control plane での スーパーバイザー への Harbor のインストールと構成を参照してください。

vSphere IaaS control plane での スーパーバイザー への Harbor のインストールと構成

Harbor をインストールし、スーパーバイザー サービス として構成する方法を確認してください。この操作の後、Tanzu Kubernetes Grid クラスタおよび vSphere ポッド で実行されているワークロードでは、Harbor をレジストリとして使用できます。Harbor には Ingress コントローラとして Contour が必要であるため、最初に Contour スーパーバイザー サービス をインストールしてから Harbor をインストールします。

スーパーバイザー サービス としての Harbor のインストール

vSphere Client の [Workload Management] オプションを使用して Harbor を スーパーバイザー サービス としてインストールします。

前提条件

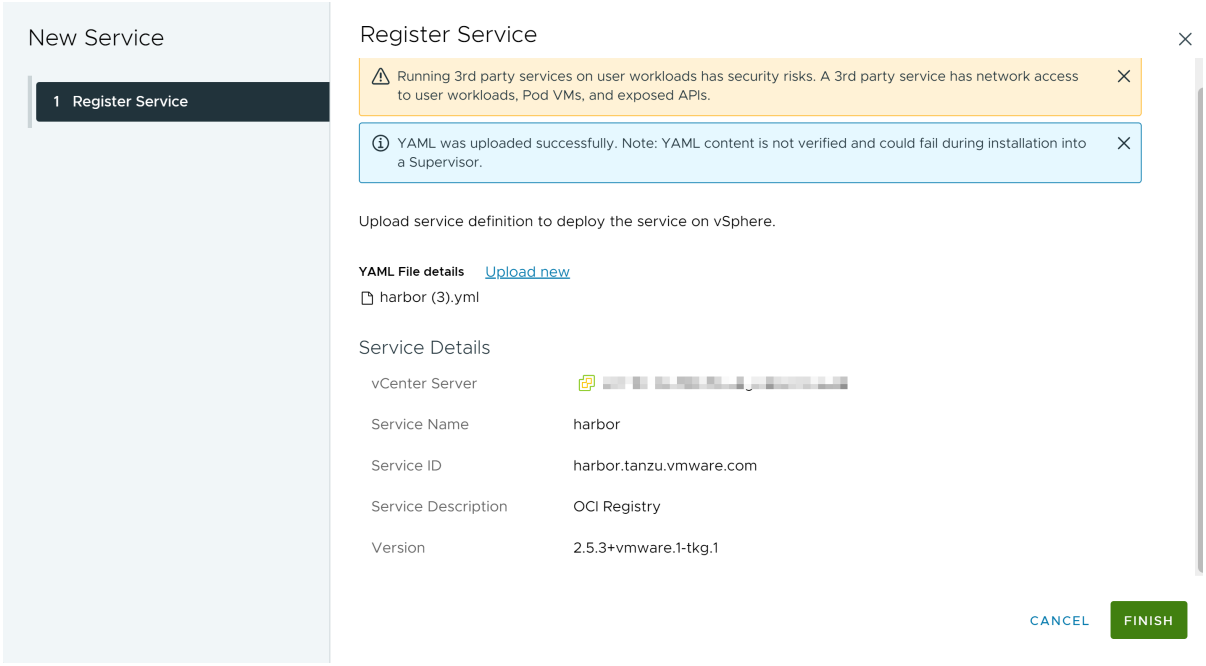
- vCenter Server 8.0a 以降にアップグレードしたことを確認します。Contour および Harbor の スーパーバイザー サービス は、vCenter Server 8.0a 以降でサポートされます。
- サービスを追加する vCenter Server システムに関するスーパーバイザー サービスの管理権限があることを確認します。
- Harbor をインストールする スーパーバイザー に、Contour を スーパーバイザー サービスとしてインストールします。vSphere IaaS control plane での スーパーバイザー サービス としての Contour のインストールを参照してください。
- Harbor 管理ユーザー インターフェイスにアクセスするための FQDN を指定します。

手順

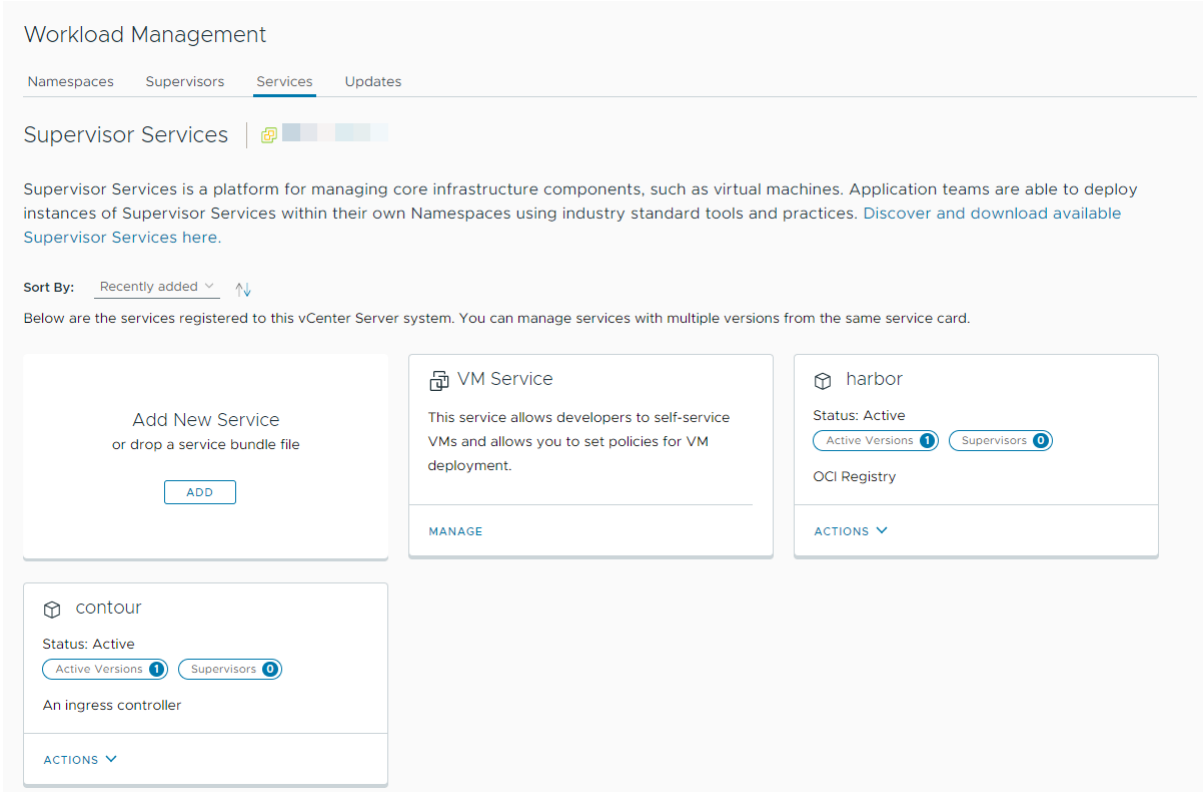
1. Supervisor-Services リポジトリの Harbor Versions セクションに移動して、次のファイルをダウンロードします。
   - Harbor サービス定義。リンクの名前は `Harbor vX.X.X.` です。例：`Harbor 2.5.3`
   - Harbor 構成ファイル。リンクの名前は `values for vX.X.X` です。例：`values 2.5.3`

   生成されたファイルは、次のようになります。
   - `harbor.yml`
   - `harbor-data-values.yml`
2. vSphere Client で [ワークロード管理] に移動し、[サービス] を選択します。

3 [サービスの新規追加] をクリックし、`harbor.yml` サービス定義をアップロードして、Harbor オペレータをデプロイします。



Harbor オペレータがデプロイされると、このオペレータが [サービス] タブに表示されます。



4 Harbor オペレータがデプロイされ、Contour が実行されている スーパーバイザー に スーパーバイザー サービス をインストールできます。

a `harbor-data-values.yml` ファイルを開き、必要に応じてプロパティを編集します。

プロパティ	値	説明

`hostname: myharbor.com`
 	
`https: 443`
	FQDN	Harbor 管理ユーザー インターフェイスにアクセスするために指定した FQDN に変更します。

`tlsCertificate:`
`tlsSecretLabels:`
`{"managed-by": "vmware-vRegistry"}`
	注： 変更しない	TKG 統合が機能するためには、この値が必要です。

`harborAdminPassword:` `Harbor12345`	必要に応じて変更する	インストール中に使用される Harbor のパスワード。サービスがインストールされたら、Harbor 管理ユーザー インターフェイスを使用して変更できます。
`secretKey:` `0123456789ABCDEF`	16 文字からなる文字列	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: change-it`	安全なパスワード	Postgres データベースで使用される初期パスワード。

`core:`
`replicas:`
`secret: change-it`
`xsrfKey:`
`0123456789ABCDEF0123456789ABCDEF`
`jobservice:`
`replicas: 1`
`secret: change-it`
`registry:`
`replicas:`
`secret: change-it`
	シークレットの文字列と 32 文字の XSRF キー文字列	変更して独自のシークレットを設定します。

`persistence:`
`persistentVolumeClaim:`
`registry:`
`storageClass:`
`"insert-storage-class-name-here"`
`subPath: ""`
`accessMode:`
`ReadWriteOnce`
`size: 10Gi`
`jobservice:`
`storageClass:`
`"insert-storage-class-name-here"`
`subPath: ""`
`accessMode:`
`ReadWriteOnce`
	ストレージ クラスの名前	Harbor のレジストリ、ジョブ サービス、データベースなど、PVC をプロビジョニングするためのストレージ クラスとして使用されるストレージ ポリシー。
		各プロパティを、環境内で使用可能な既存のストレージ ポリシーに設定します。すべての大文字を小文字に置き換えて、ストレージ ポリシー名を有効なストレージ クラス名に変更するとともに、すべての「_」記号とスペースをダッシュ「-」に置き換えます。たとえば、**`Harbor Storage Policy`** を **`harbor-storage-policy`** に変更します。

プロパティ	値	説明

<pre>size: 1Gi
database:
storageClass:
"insert-storage-class-
name-here"
subPath: ""
accessMode:
ReadWriteOnce
size: 1Gi
redis:
storageClass:
"insert-storage-class-
name-here"
subPath: ""
accessMode:
ReadWriteOnce
size: 1Gi
trivy:
storageClass:
"insert-storage-class-
name-here"
subPath: ""
accessMode:
ReadWriteOnce
size: 5Gi</pre>		

<pre>network:
ipFamilies: ["IPv4"]</pre>
	注： 変更しない	IPv6 はサポートされていません。

b [ワークロード管理] - [サービス] に戻り、[Harbor] サービス カードで [アクション] - [スーパーバイザーへのインストール] の順に選択します。

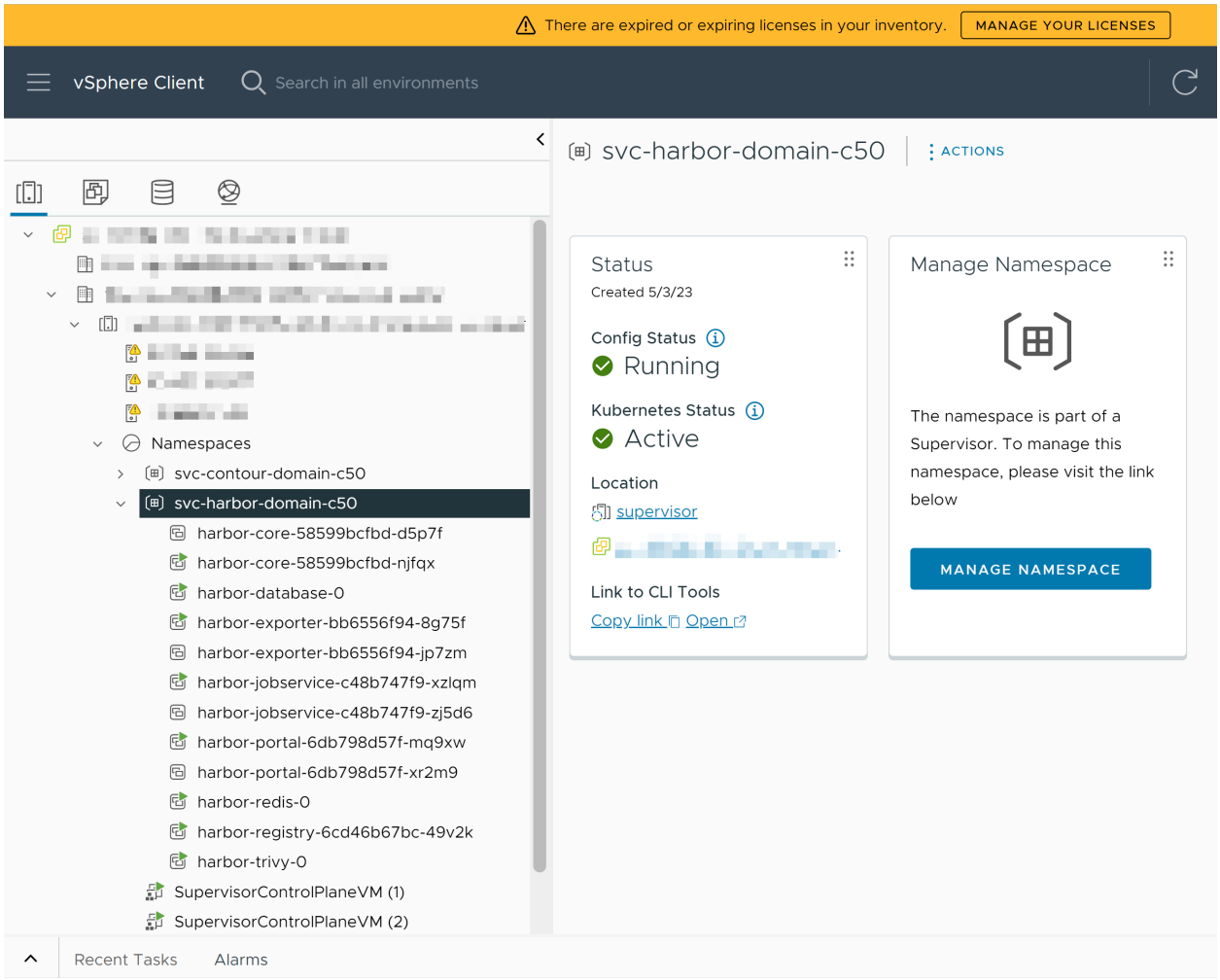
c Contour が実行されている スーパーバイザー を選択し、[YAML サービス構成] で、変更した `harbor-data-values.yml` ファイルの内容をコピーし、貼り付けます。

d [OK] をクリックします。

インストールが開始したら、Harbor サービス カードの [スーパーバイザー] フィールドをクリックしてインストールを追跡できます。[スーパーバイザー] の横にある数字が増えるまで、数秒かかる場合があります。目的の状態に達するまで、サービスの状態は [設定中] になります。目的の状態に達すると、サービスの状態は [実行中] に変化します。

結果

Harbor 用に作成された vSphere 名前空間 と vSphere ポッド は、[ホストおよびクラスタ] ビューで確認できます。

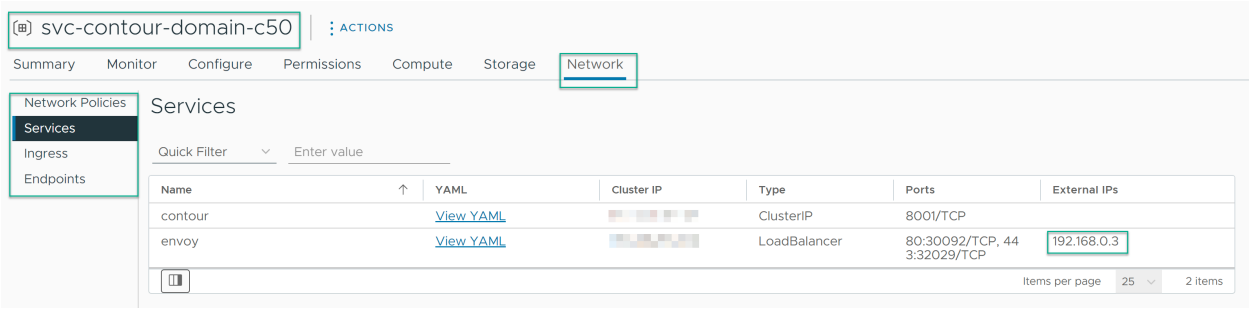


Harbor FQDN の Envoy Ingress IP アドレスへのマッピング

Harbor が正常にインストールされたら、スーパーバイザー が構成された外部 DNS サーバに、Envoy Ingress IP アドレスに対する Harbor FQDN のマッピングのレコードを追加します。

Tanzu Kubernetes Grid クラスタ、vSphere ポッド、および スーパーバイザー のレジストリからイメージをプルするには、Harbor FQDN を解決できる必要があります。

Envoy Ingress IP アドレスを見つけるには、Contour 名前空間に移動し、[ネットワーク] を選択して [サービス] を選択します。



Harbor スーパーバイザー サービス との信頼の確立

Harbor をインストールした後に Harbor を vSphere ポッド のレジストリとして使用するには、スーパーバイザー と Harbor 間に信頼を構成する必要があります。Harbor と同じ スーパーバイザー にある Tanzu Kubernetes Grid クラスタには、Harbor に対する信頼が自動的に確立されます。異なる スーパーバイザー で実行される Tanzu Kubernetes Grid クラスタのレジストリとして Harbor を使用するには、Harbor とこれらの Tanzu Kubernetes Grid クラスタ間に信頼を構成する必要があります。

Harbor と スーパーバイザー 間の信頼の確立

Harbor と スーパーバイザー 間に信頼を確立するには：

1 Harbor ユーザー インターフェイスを使用するか、スーパーバイザー 制御プレーンの TLS シークレットを使用して、Harbor CA を抽出します。Harbor の `ca.cert` は、Harbor 管理ユーザー インターフェイスの [管理] - [構成] - [レジストリ ルート証明書] - [ダウンロード] で取得できます。

2 `kube-system` 名前空間の `image-fetcher-ca-bundle` ConfigMap に Harbor CA を追加します。vCenter Single Sing-On 管理アカウントを使用してログインしているほかに、`image-fetcher-ca-bundle` を編集する権限を保持している必要があります。

   a こちらの説明に従って、KUBE_EDITOR 環境変数を構成します。

   b 次のコマンドを使用して、ConfigMap を編集します。

```
kubectl edit configmap image-fetcher-ca-bundle -n kube-system
```

   c Harbor の `ca.cert` ファイルの内容を、既存の スーパーバイザー 証明書の下にある ConfigMap に追加します。スーパーバイザー 証明書を変更しないでください。

```
apiVersion: v1
data:
  ca-bundle: |-
    -----BEGIN CERTIFICATE-----
    MIIC/jCCAeagAwIBAgIBADANBgkqhkiG9w0BAQsFADAVMRMwEQYDVQQDEwprdWJl
    ...
    qB72tWi8M5++h2RGcVash0P1CUZOHkpHxGdUGYv1Z97Wl89dT2OTn3iXqn8d1JAK
    aF8=
    -----END CERTIFICATE-----
    -----BEGIN CERTIFICATE-----
    MIIDKDCCAhCgAwIBAgIQBbUsj7mqXXC5XRhqqU3GiDANBgkqhkiG9w0BAQsFADAU
    ...
    5q7y87vOLTr7+0MG4O01zK0dJYx2jVhZlsuduMYpfqRLLewVl0eGu/6vr2M=
```

```
    -----END CERTIFICATE-----
kind: ConfigMap
metadata:
  creationTimestamp: "2023-03-15T14:28:34Z"
  name: image-fetcher-ca-bundle
  namespace: kube-system
  resourceVersion: "713"
  uid: 6b7611a0-25fa-40f7-b4f5-e2a13bd0afe3
```

d ファイルに加えた編集内容を保存します。その結果、kubectl から次のように報告されます。

```
configmap/image-fetcher-ca-bundle edited
```

Harbor とは異なる スーパーバイザー で実行されている Tanzu Kubernetes Grid クラスタと Harbor の間の信頼の確立

Harbor のインストール場所とは異なる スーパーバイザー で実行されている Tanzu Kubernetes Grid クラスタは、Harbor にネットワーク接続されている必要があります。これらの Tanzu Kubernetes Grid クラスタで Harbor の FQDN を解決できる必要があります。

Harbor と Tanzu Kubernetes Grid クラスタ間の信頼を確立するには、Harbor ユーザー インターフェイスを使用するか、スーパーバイザー 制御プレーンの TLS シークレットを使用して、Harbor CA を抽出します。その後、TKG 2 クラスタとプライベート コンテナ レジストリの統合に記載されている手順に従います。

vSphere IaaS control plane での組み込みレジストリから Harbor へのイメージの移行

スーパーバイザー で組み込みの Harbor レジストリを使用している場合は、組み込みのレジストリから スーパーバイザー サービス としてインストールした Harbor レジストリにイメージを移行できます。

前提条件

- スーパーバイザー に Contour と Harbor の スーパーバイザー サービス がインストールされていることを確認します。
- スーパーバイザー で使用する DNS に、Envoy サービスの Ingress IP アドレスにマッピングされた Harbor FQDN のエントリが含まれていることを確認します。
- スーパーバイザー と Harbor の間に信頼が確立されていることを確認します。Harbor の実行場所とは異なる スーパーバイザー で実行されている Tanzu Kubernetes Grid クラスタによってイメージが参照されている場合は、これらの Tanzu Kubernetes Grid クラスタと Harbor の間に信頼があることを確認します。

手順

1 vCenter Single Sign-On ユーザーとして スーパーバイザー にログインします。

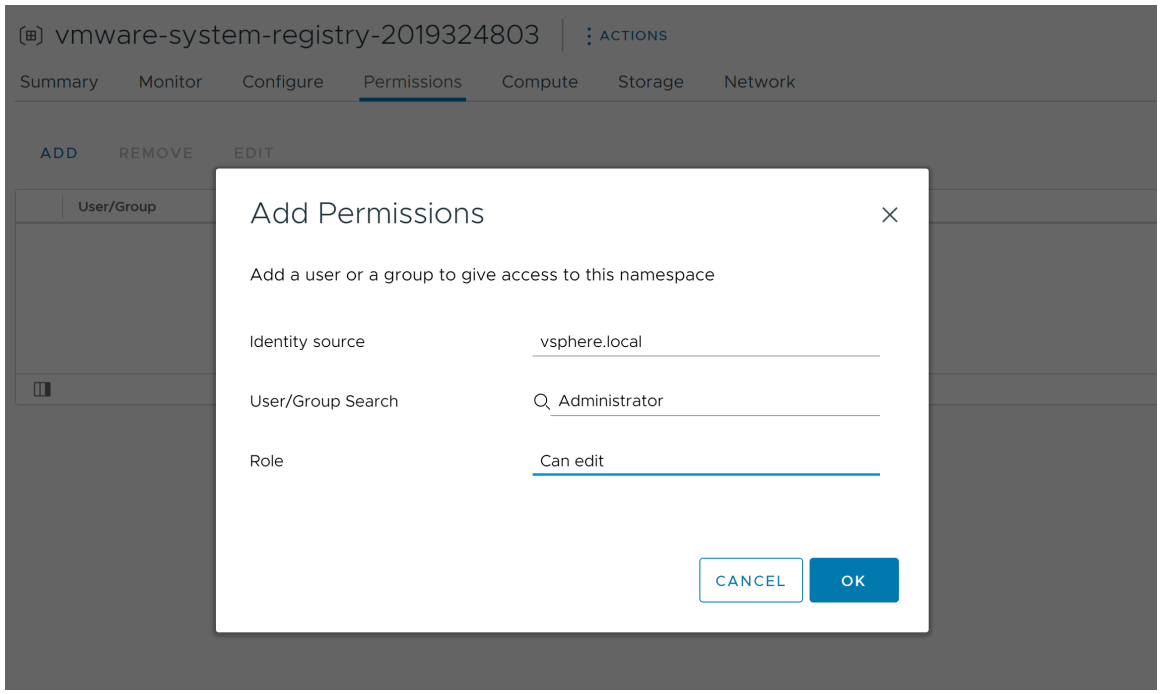
2 Harbor スーパーバイザー サービス へのネットワーク アクセスの Egress を設定します。

a Harbor のサービス名前空間に `allow-all-egress-harbor-supervisor-service` という名前のネットワーク ポリシー CRD を作成します。名前の例は、`svc-harbor-domain-c9` などです。

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-all-egress-harbor-supervisor-service
  namespace: svc-harbor-domain-c9
spec:
  podSelector:
    matchLabels:
      app: harbor
  egress:
  - {}
```

3 後で Harbor にレプリケーション エンドポイントとしてレジストリを追加できるように、組み込みレジストリのシークレットにアクセスします。

a 組み込みレジストリ名前空間の vCenter Single Sign-On 管理ユーザーに編集権限を付与します。名前の例は、`vmware-system-registry-437393318` などです。



b 組み込みのレジストリ名前空間からシークレットにアクセスします。

```
# kubectl get secrets -n vmware-system-registry-437393318 harbor-437393318-controller-registry -o yaml
apiVersion: v1
data:
  harborAdminPassword: UDNSak4wQk5VbFlrY1VZeVprUmpKQT09
  harborAdminUsername: WVdSdGFXND0=
  harborPostgresPassword: TlRoS1ZHeEFLa1lrVkdjaGN6aGtXZz09
kind: Secret
...
```

c ユーザー名とパスワードをデコードします。

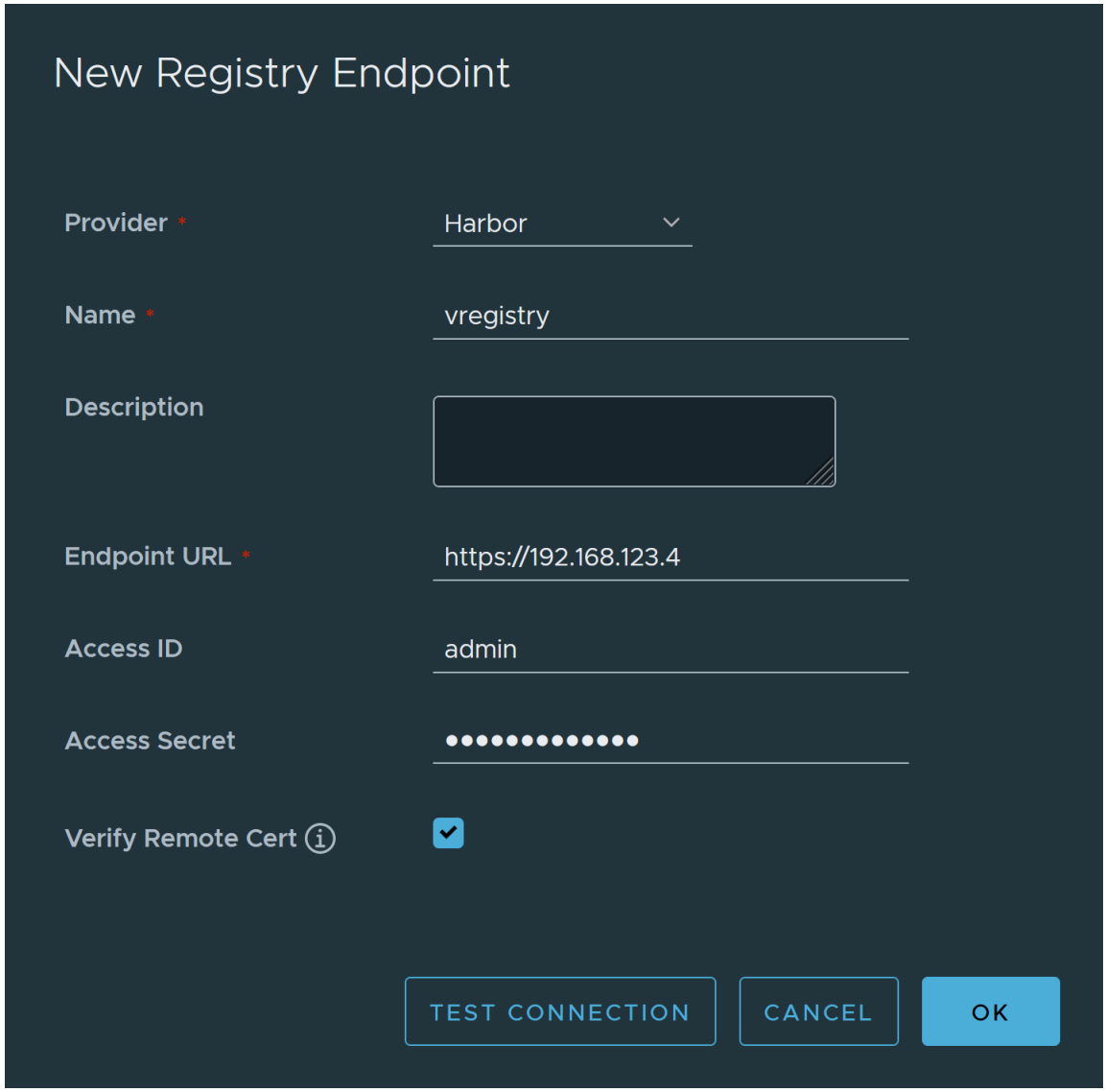
```
# echo 'WVdSdGFXND0=' | base64 -d | base64 -d
admin

# echo 'UDNSak4wQk5VbFlrY1VZeVprUmpKQT09' | base64 -d | base64 -d
?tc7@MRV$qF2fDc$
```

4 組み込みレジストリのレプリケーション エンドポイントおよびレプリケーション ルールを Harbor スーパーバイザー サービス に追加します。

a Harbor スーパーバイザー サービス のユーザー インターフェイスに root としてログインします。

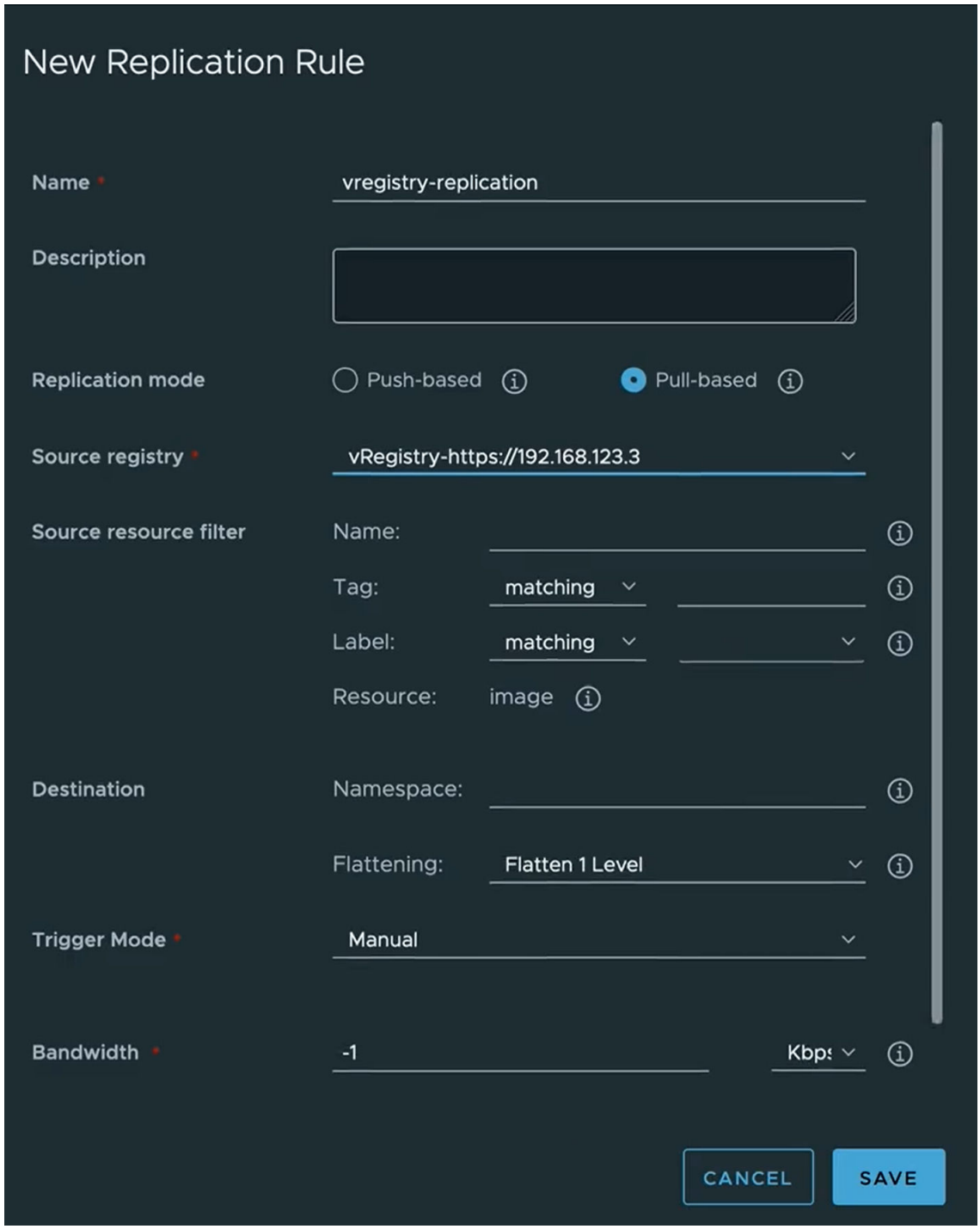
b [レジストリ] をクリックし、[新規エンドポイント] をクリックします。



c [レプリケーション] タブを選択し、[新規レプリケーション ルール] をクリックします。

次の設定を入力し、残りの値はデフォルト値のままにします。

- [名前] - ルールの名前を指定します。
- [レプリケーション モード] - [プルベース] を選択します。
- [ソース レジストリ] - 追加したレジストリ エンドポイントを選択します。



d [保存] をクリックします。

5 新しく作成したレプリケーション ルールを選択し、[レプリケーション] をクリックします。

結果

組み込みレジストリの内容が Harbor レジストリにレプリケートされます。

プロキシからイメージをプルしてエアギャップ環境に スーパーバイザー サービス をデプロイする

10

スーパーバイザー サービス をエアギャップ環境にデプロイして、イントラネット内のプライベート コンテナ レジストリを活用できます。

スーパーバイザー サービス は、Kubernetes YAML マニフェストとコンテナ イメージのコレクションとして保存される Kubernetes Operator であり、複数の名前空間にリソースをデプロイできます。これには、スーパーバイザー に共通する一連のレジストリの詳細が必要です。プライベート イメージから スーパーバイザー サービス コンテナ イメージをデプロイするには、プライベート レジストリが自己署名認証局を使用している場合や、イメージ プルの認証が必要な場合に、スーパーバイザー の信頼と認証を構成する必要があります。プライベート レジストリがパブリック認証局によって署名された TLS 証明書を使用する場合、認証局の構成は必要ありません。

エアギャップ環境でプロキシまたはプライベート レジストリを介してプルされたすべての スーパーバイザー サービス をデプロイできます。スーパーバイザー サービス の完全なセットについては、「https://vsphere-tmm.github.io/Supervisor-Services/」を参照してください。

次の手順を実行してください。

1 スーパーバイザー サービス をプライベート コンテナ レジストリに再配置します。

2 プライベート コンテナ イメージ レジストリでホストされている スーパーバイザー サービス をインストールして使用します。

次のトピックを参照してください。

- プライベート レジストリへのスーパーバイザー サービスの再配置
- スーパーバイザー サービスのインストールと使用

プライベート レジストリへのスーパーバイザー サービスの再配置

スーパーバイザー サービスをプライベート コンテナ レジストリに再配置します。

前提条件

プライベート コンテナ イメージ レジストリがあることを確認します。

手順

1 Carvel `imgpkg` ユーティリティをインストールします。

a `imgpkg` のインストール

```
wget -O- https://carvel.dev/install.sh > install.sh
sudo bash install.sh
```

b インストールを確認します。

```
imgpkg version
```

Carvel `imgpkg` ユーティリティの詳細については、https://carvel.dev/imgpkg/docs/v0.42.x/install/ を参照してください。

2 サービスの YAML マニフェストを取得します。

`imgpkg` バンドルを見つけます。

Contour の例を次に示します。

```
template:
  spec:
    fetch:
    - imgpkgBundle:
        image: projects.registry.vmware.com/tkg/packages/standard/contour:v1.24.4_vmware.1-
tkg.1
```

3 `imgpkg` バンドルの tar をダウンロードします。

```
imgpkg copy -b projects.registry.vmware.com/tkg/packages/standard/contour:v1.24.4_vmware.1-
tkg.1 --to-tar contour-v1.24.4.tar --cosign-signatures
```

重要： `push` コマンドと `pull` コマンドでは参照先のすべてのイメージがプルダウンされないため、イメージを再配置するには、これらのコマンドではなく `copy` コマンドを使用する必要があります。

4 `imgpkg` バンドルをプライベート コンテナ イメージ レジストリにアップロードします。

```
imgpkg copy --tar contour-v1.24.4.tar --to-repo ${registry_url}/contour --cosign-signatures
```

注： `imgpkg` では、認証のためのシステムの信頼設定と Docker の構成が考慮されます。レジストリで認証が必要な場合は、まず Docker CLI コマンド `docker login ${registry_url}` を使用してログインします。

5 `imgpkg` バンドルの新しい URL を使用して、スーパーバイザー サービスの YAML を更新します。

例：

```
template:
  spec:
    fetch:
    - imgpkgBundle:
        image: n.n.n.n/contour:v1.24.4_vmware.1-tkg.1
```

スーパーバイザー サービスのインストールと使用

スーパーバイザー サービスをプライベート コンテナ イメージ レジストリに再配置した後で、それらをインストールして使用できます。

スーパーバイザー サービスを使用するには、まずプライベート レジストリを追加してから、スーパーバイザー サービスを登録してインストールします。

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 プライベート レジストリを追加します。

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 [スーパーバイザー] タブをクリックし、リストから スーパーバイザー を選択します。

c [構成] タブをクリックし、[コンテナ レジストリ]、[追加] の順にクリックします。

d プライベート レジストリの名前を入力し、必要に応じて CA、ユーザー名、パスワードを入力します。

2 vCenter Server にスーパーバイザー サービスを追加します。

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]] を選択します。

c 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。

d [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。

3 スーパーバイザー サービス のインストール

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]] を選択します。

c インストールする スーパーバイザー サービス のカードで、[アクション] - [スーパーバイザーへのインストール] の順に選択します。

d サービスをインストールする スーパーバイザー を選択します。

e サービスに構成プロパティが必要な場合は、[YAML サービス構成] フィールドに入力します。

スーパーバイザー サービスが `SupervisorServiceDefinition` 形式であり、レジストリで認証が必要な場合は、`registryName`、`registryUsername`、および `registryPasswd` プロパティを入力します。