vSphere IaaS 제어부 서비스 및 워크로드

업데이트 3

VMware vSphere 8.0
VMware vCenter 8.0
VMware ESXi 8.0

vmware by Broadcom

VMware by Broadcom 웹 사이트

https://docs.vmware.com/kr에서 최신 기술 문서를 찾을 수 있습니다.

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

목차

"vSphere IaaS 제어부 서비스 및 워크로드" 정보

1

"vSphere IaaS 제어부 서비스 및 워크로드" (공식적 이름: "vSphere with Tanzu 서비스 및 워크로드")는 vSphere IaaS control plane(공식적 이름: vSphere with Tanzu) 환경의 vSphere 네임스페이스에서 서비스 및 워크로드를 실행하는 방법을 설명합니다. 네임스페이스를 생성하고 감독자 서비스, 가상 시스템 및 vSphere 포드와 같은 워크로드를 배포하는 방법을 알아볼 수 있습니다.

대상 사용자

이 정보는 주로 vSphere IaaS control plane를 사용하여 vSphere 리소스를 구성하고 감독자의 vSphere 네임스페이스에 할당하는 vSphere 관리자를 대상으로 합니다. 이러한 리소스는 나중에 네임스페이스 내에서 실행되는 다양한 서비스 및 워크로드에서 사용할 수 있습니다. 일반적으로 vSphere IaaS control plane로 작업하는 vSphere 관리자는 컨테이너 및 기타 Kubernetes 개념에 대한 기본 지식을 가지고 있습니다.

이 가이드는 감독자에서 vSphere 포드, VM 및 감독자 서비스와 같은 워크로드를 배포하려는 DevOps 팀에서도 사용할 수 있습니다.

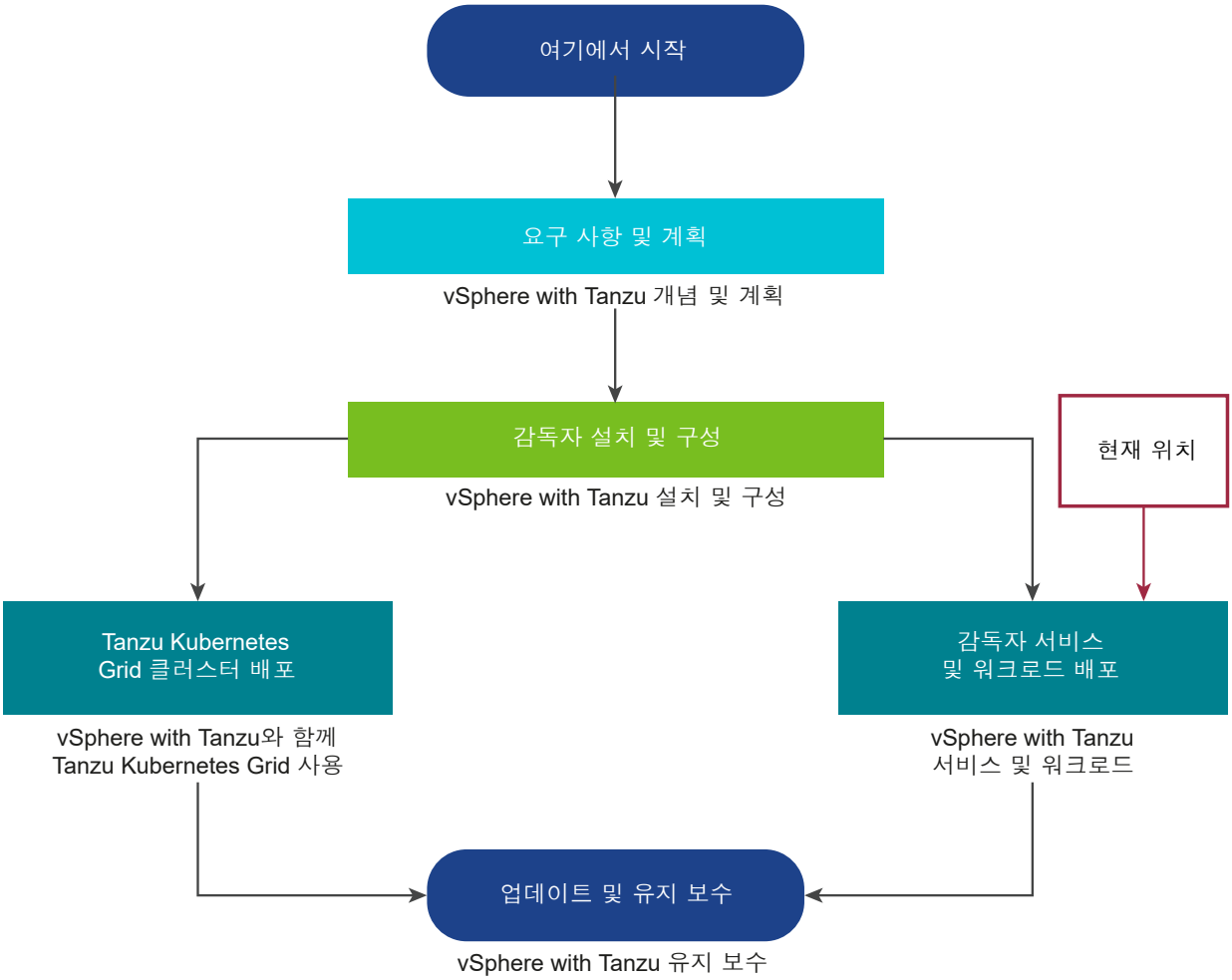
참고 "vSphere IaaS 제어부 서비스 및 워크로드" 에는 Tanzu Kubernetes Grid 클러스터에서 워크로드를 실행하는 데 대한 정보는 포함되어 있지 않습니다. Tanzu Kubernetes Grid 클러스터에서 작업하는 방법을 알아보려면 vSphere IaaS Control Plane과 함께 감독자에서 Tanzu Kubernetes Grid 사용을 참조하십시오.

"vSphere IaaS 제어부 서비스 및 워크로드" 설명서 사용

이 "vSphere IaaS 제어부 서비스 및 워크로드" 가이드 외에도 vSphere IaaS control plane 설명서에는 몇 가지 다른 가이드가 포함되어 있습니다. vSphere IaaS control plane 설명서의 계층 구조를 숙지해야 합니다. 이러한 가이드 중 일부는 "vSphere IaaS 제어부 서비스 및 워크로드" 의 사전 요구 사항이기 때문입니다.

이 그림에는 vSphere IaaS control plane 설명서 세트를 사용하는 방법과 각 가이드에서 찾을 수 있는 정보가 설명되어 있습니다.

그림 1-1. vSphere IaaS control plane 설명서 맵



업데이트된 정보

이 "vSphere IaaS 제어부 서비스 및 워크로드" 게시물은 제품의 각 릴리스에 따라 또는 필요할 때 업데이트됩니다.

이 표에는 "vSphere IaaS 제어부 서비스 및 워크로드" 의 업데이트 기록이 나와 있습니다.

개정	설명
2024년 6월 25일	vSphere 8.0 업데이트 3 릴리스에 대한 일반 업데이트 및 개선 사항(다음 기능 포함): ■ 에어갭 환경에서 감독자 서비스 배포. 장 10 프록시에서 이미지를 끌어와 에어갭 환경에 감독자 서비스 배포의 내용을 참조하십시오. ■ 네임스페이스에 대한 네트워크 및 로드 밸런서 매개 변수 구성. 네임스페이스에 대한 네트워크 및 로드 밸런서 매개 변수 구성의 내용을 참조하십시오. ■ VM Operator v1alpha2 지원. 장 6 vSphere IaaS control plane에서 가상 시스템 배포 및 관리의 내용을 참조하십시오. ■ `kubectl get virtualmachineclass`를 사용하여 특정 감독자 네임스페이스에 대한 VM 클래스를 나열할 수 있습니다. 이전에는 VM 클래스가 클러스터 범위의 리소스였기 때문에 특정 네임스페이스에 어떤 VM 클래스가 할당되었는지 확인하기 어려웠습니다. vSphere IaaS control plane의 네임스페이스에서 사용 가능한 VM 리소스 보기의 내용을 참조하십시오.
2024년 4월 29일	권장 이미지에서 다운로드할 수 있는 사용 가능한 OVA 이미지에 대한 참고 사항이 추가되었습니다. vSphere IaaS control plane에서 독립형 VM에 대한 컨텐츠 라이브러리 생성의 내용을 참조하십시오.
2024년 4월 1일	NVIDIA GRID vGPU에 사용되는 고급 매개 변수에 대한 정보를 포함하도록 vSphere IaaS control plane에서 vGPU 및 기타 PCI 디바이스를 사용하여 VM 배포업데이트되었습니다.
2024년 2월 29일	제품의 변경 내용을 반영하도록 VM 클래스 편집에 관한 컨텐츠가 업데이트되었습니다. vSphere Client를 사용하여 VM 클래스 편집의 내용을 참조하십시오.
2023년 12월 11일	vSAN 파일 서비스에서 지원하는 RWX 볼륨의 제한 사항에 대한 설명이 추가되었습니다. vSAN 파일 서비스를 사용하여 vSphere IaaS control plane에서 ReadWriteMany 볼륨 생성의 내용을 참조하십시오.
2023년 11월 7일	파일 볼륨이 Tanzu Kubernetes Grid 클러스터의 워크로드에 대해서만 지원된다는 점을 명시하기 위해 vSAN 파일 서비스를 사용하여 vSphere IaaS control plane에서 ReadWriteMany 볼륨 생성업데이트되었습니다.
2023년 9월 29일	■ 감독자 서비스에 대한 새로운 플랫폼 지원 정보가 추가되었습니다. 장 4 vSphere IaaS control plane를 사용하여 감독자 서비스 관리의 내용을 참조하십시오. ■ 부분적 업데이트.
2023년 9월 21일	■ 컨텐츠 라이브러리를 네임스페이스 또는 감독자에 연결하기 위해 DCLI(Data Center CLI) 명령을 사용하는 방법에 대한 새 컨텐츠가 추가되었습니다. vSphere IaaS control plane에서 VM 컨텐츠 라이브러리 추가 및 관리의 내용을 참조하십시오. ■ 새 VM 이미지를 네임스페이스와 연결된 쓰기 가능 컨텐츠 라이브러리에 게시하는 방법에 대한 항목이 추가되었습니다. vSphere IaaS control plane에서 컨텐츠 라이브러리 이미지 관리 및 게시의 내용을 참조하십시오. ■ DCLI 명령을 사용하여 VM 클래스를 생성하고 관리하는 방법에 대한 새 컨텐츠가 추가되었습니다. 데이터 센터 CLI를 사용하여 VM 클래스 생성 및 관리의 내용을 참조하십시오.

개정	설명
2023년 7월 14일	■ 감독자 서비스가 사용되는 방식에 대한 설명이 향상되었습니다. 장 4 vSphere IaaS control plane를 사용하여 감독자 서비스 관리의 내용을 참조하십시오. ■ Tier-0 워크로드 네트워크 설정을 재정의하기 위한 요구 사항이 감독자에서 vSphere 네임스페이스 생성 및 구성에 추가되었습니다.
2023년 6월 30일	감독자에서 vSphere 네임스페이스 제거 항목이 추가되었습니다.
2023년 6월 21일	StatefulSet 정의를 사용할 때 StatefulSet의 일부로 생성된 볼륨을 확장할 수 없음을 명확히 설명했습니다. vSphere IaaS control plane의 볼륨 확장의 내용을 참조하십시오.
2023년 5월 16일	vSphere 8 업데이트 1 릴리스부터 두 가지 네트워킹 유형(NSX 또는 VDS)으로 배포된 감독자에서 감독자 서비스를 사용할 수 있다는 참고 사항이 추가되었습니다. 장 4 vSphere IaaS control plane를 사용하여 감독자 서비스 관리의 내용을 참조하십시오.
2023년 5월 15일	부분적 개정.
2023년 5월 11일	■ 워크로드 지원 가능성 테이블이 장 2 vSphere IaaS control plane 서비스 및 워크로드에 대한 워크플로에 추가되었습니다. ■ 사용하지 않으면 2분 이내에 만료되는 VM의 웹 콘솔 URL에 대한 참고 사항이 추가되었습니다. vSphere VM 웹 콘솔을 사용하여 VM 문제 해결의 내용을 참조하십시오. ■ vSphere IaaS control plane 설명서의 계층 구조를 보여주는 그림이 추가되었습니다. vSphere IaaS 제어부 서비스 및 워크로드 설명서 사용의 내용을 참조하십시오.
2023년 5월 9일	■ vSAN 파일 서비스가 지원하는 ReadWriteMany 볼륨이 볼륨 확장을 지원하지 않는다는 정보가 추가되었습니다. vSphere IaaS control plane의 볼륨 확장의 내용을 참조하십시오. ■ vSphere 포드에 대한 개념 정보가 장 7 vSphere 포드에 워크로드 배포에 추가되었습니다.
2023년 5월 5일	■ 장 9 vSphere IaaS control plane에서 Harbor 및 Contour 설치 및 구성 장이 업데이트되었습니다. ■ vSphere 포드는 NSX 네트워킹 스택에서만 지원된다는 정보가 추가되었습니다. ■ vSphere 포드 배포 예시가 추가되었습니다. vSphere IaaS control plane에서 vSphere 포드 워크로드 배포 항목을 참조하십시오.
2023년 4월 26일	장 3 vSphere 네임스페이스 구성 및 관리 장이 추가되었습니다.
2023년 4월 18일	■ 구성 가능한 OVF 속성을 사용하여 VM을 배포하는 방법에 대한 항목이 추가되었습니다. vSphere IaaS control plane에서 구성 가능한 OVF 속성을 사용하여 VM 배포의 내용을 참조하십시오. ■ DevOps 엔지니어가 문제 해결을 위해 VM 및 게스트 운영 체제에 직접 액세스하는 데 사용할 수 있는 vSphere VM 웹 콘솔에 대한 정보가 추가되었습니다. vSphere VM 웹 콘솔을 사용하여 VM 문제 해결의 내용을 참조하십시오.

vSphere IaaS control plane 서비스 및 워크로드에 대한 워크플로

2

이 워크플로에서는 vSphere IaaS control plane를 이미 사용하도록 설정하고, 감독자를 설정했으며, 이제 vSphere 네임스페이스를 생성하여 워크로드에 사용할 준비가 되었다고 가정합니다.

사용자 역할

일반적으로 감독자와 상호 작용하고 워크로드를 실행하려면 두 가지 역할(vSphere 관리자 및 DevOps 엔지니어)이 필요합니다. vSphere 관리자 및 DevOps 엔지니어 역할에 대한 워크플로는 고유하며 이러한 역할에 필요한 특정 전문 기술 영역에 따라 결정됩니다.

vSphere 관리자

vSphere 관리자는 일반적으로 vSphere Client를 사용하여 감독자 및 네임스페이스를 구성하며 여기에서 DevOps 엔지니어가 Kubernetes 워크로드를 배포할 수 있습니다.

감독자를 생성하지 않았고 생성 방법에 대해 알아보려면 vSphere IaaS 제어부 설치 및 구성을 참조하십시오.

DevOps 엔지니어

감독자에서 DevOps 엔지니어의 역할은 일반적으로 Kubernetes 개발자, 애플리케이션 소유자 및 Kubernetes 관리자가 수행하는 작업을 결합할 수 있습니다. DevOps 엔지니어는 kubectl 명령을 사용합니다. vSphere 관리자가 생성해준 감독자 네임스페이스에서 vSphere 포드, VM 및 기타 워크로드를 배포하고 실행할 수 있습니다. 셀프 서비스 네임스페이스를 생성할 수도 있습니다.

이 "vSphere IaaS 제어부 서비스 및 워크로드" 가이드는 DevOps 엔지니어가 Tanzu Kubernetes Grid 클러스터에서 수행하는 작업을 다루지 않기 때문에 이러한 작업에 대해 알아보려면 vSphere IaaS 제어부를 사용하여 감독자에서 Tanzu Kubernetes Grid 사용을 참조하십시오.

감독자가 지원하는 워크로드 유형

다양한 유형의 워크로드에 대한 감독자 지원은 감독자가 사용하는 구성 및 네트워킹에 따라 다릅니다.

워크로드 유형	VDS 네트워킹을 사용하는 1개 영역 감독자	NSX 네트워킹을 사용하는 1개 영역 감독자	VDS 네트워킹을 사용하는 3개 영역 감독자	NSX 네트워킹을 사용하는 3개 영역 감독자
장 7 vSphere 포드에 워크로드 배포	아니요	예	아니요	아니요
장 6 vSphere IaaS control plane에서 가상 시스템 배포 및 관리	예	예	예	예
장 4 vSphere IaaS control plane를 사용하여 감독자 서비스 관리	예	예	아니요	아니요
Tanzu Kubernetes Grid 클러스터	예	예	예	예

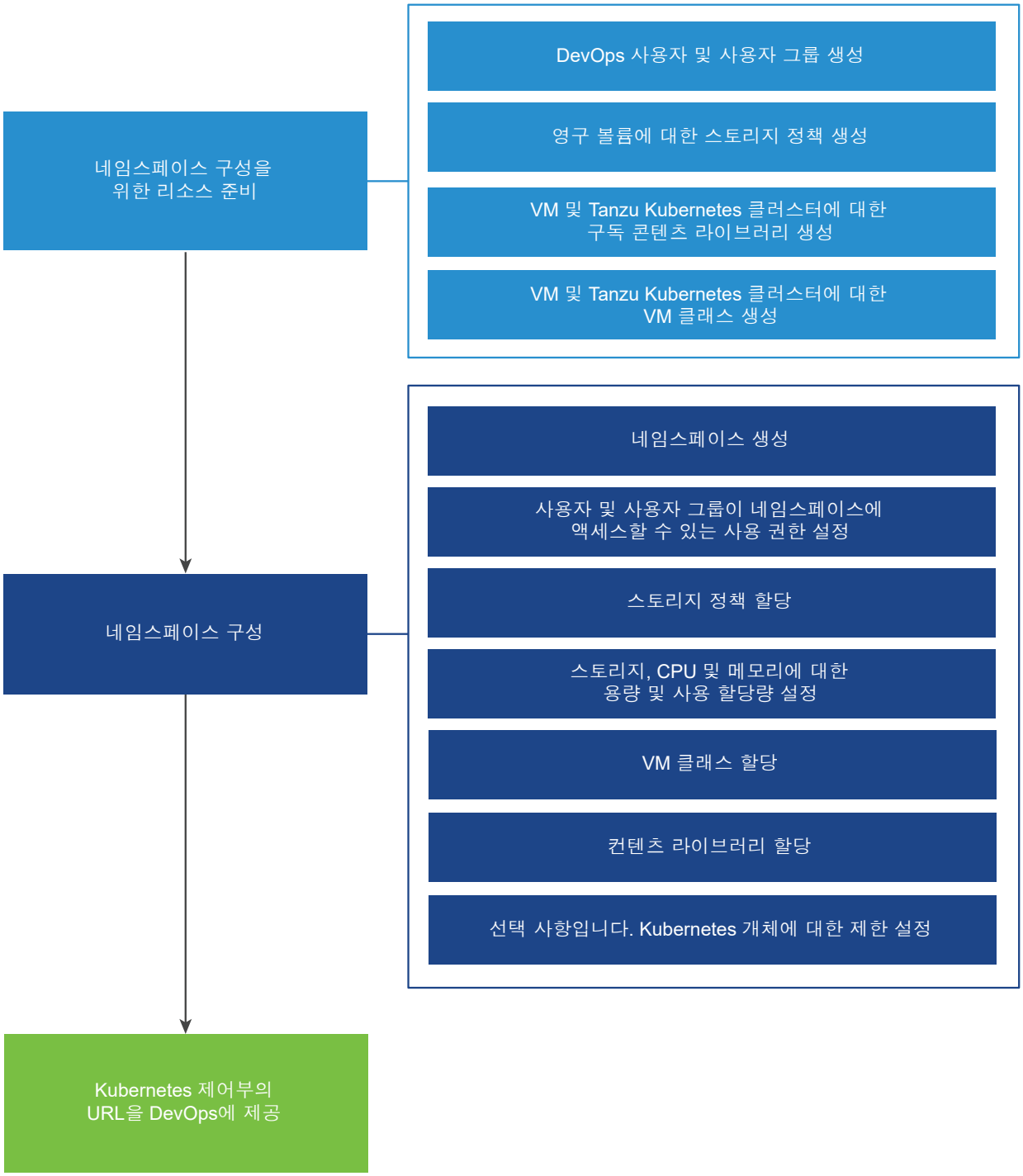
네임스페이스를 구성하는 워크플로

vSphere 관리자는 감독자에서 vSphere 네임스페이스를 생성하고 관리할 수 있습니다. Tanzu Kubernetes Grid개 클러스터

네임스페이스를 생성하기 전에 DevOps 엔지니어가 실행하려는 애플리케이션 및 워크로드에 대한 특정 리소스 요구 사항을 수집해야 합니다. 이러한 규격을 기반으로 적절한 리소스를 구성하여 네임스페이스에 할당할 수 있습니다.

자세한 내용은 장 3 vSphere 네임스페이스 구성 및 관리의 내용을 참조하십시오.

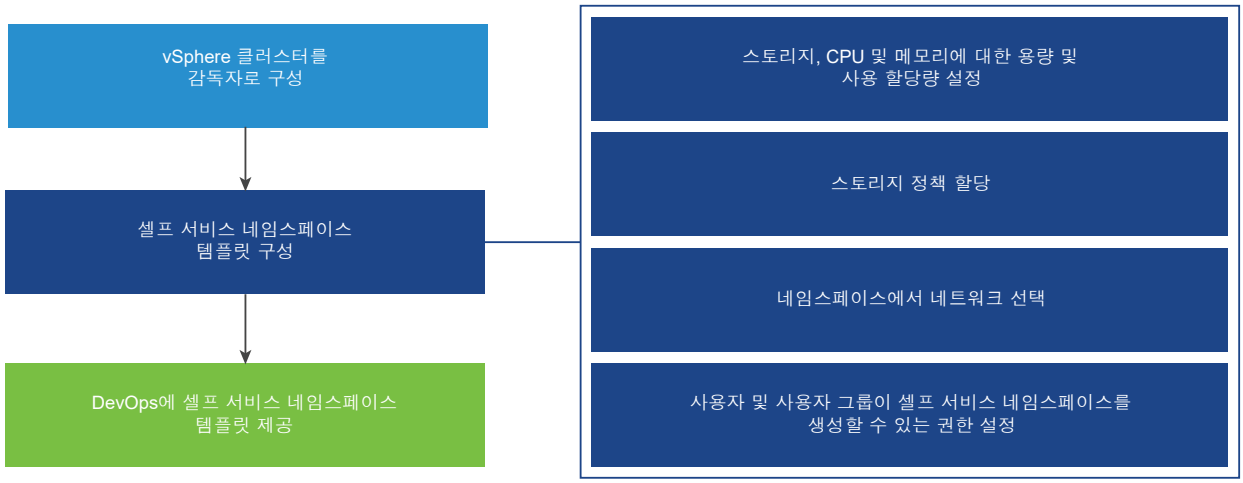
그림 2-1. 네임스페이스를 구성하는 워크플로



셀프 서비스 네임스페이스를 구성하는 워크플로

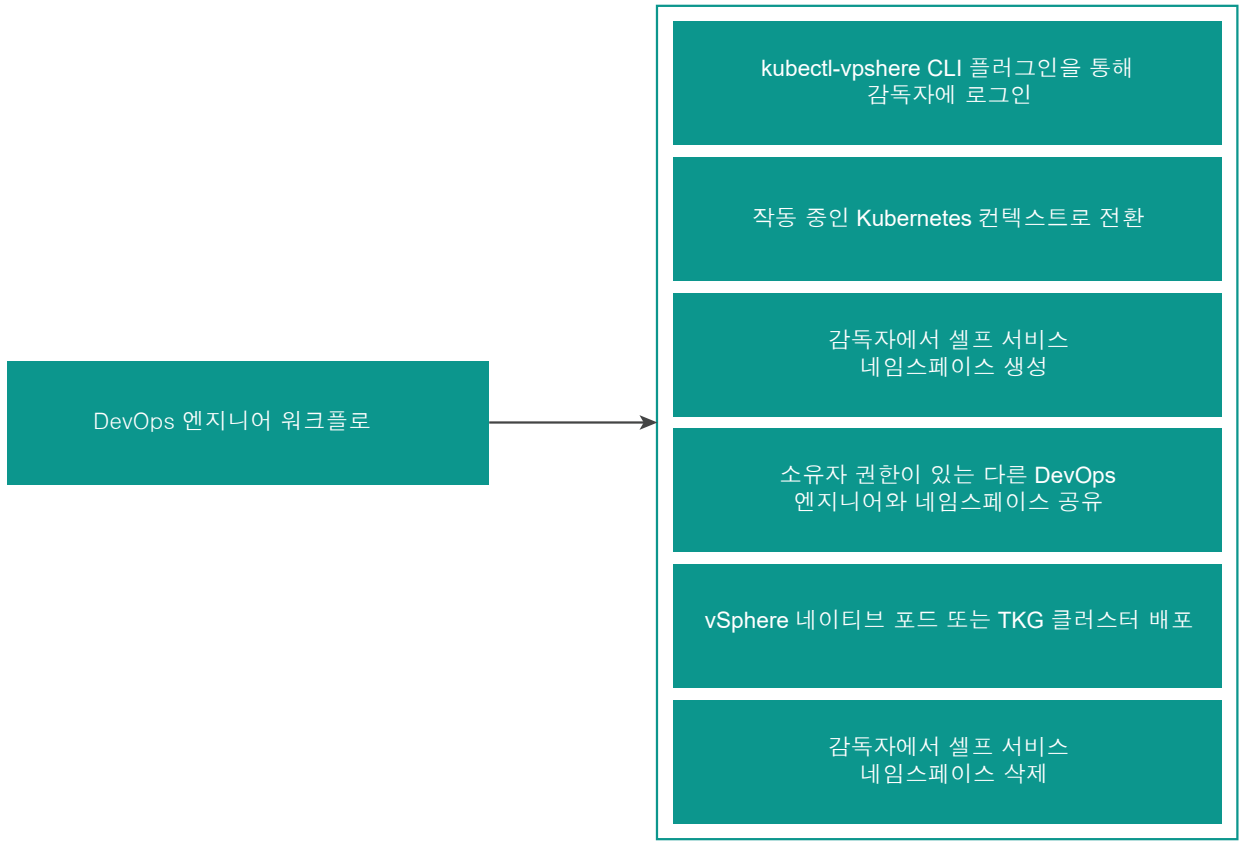
vSphere 관리자는 vSphere 네임스페이스를 생성하고, CPU, 메모리 및 스토리지 제한을 네임스페이스에 설정하고, 사용 권한을 할당하고, 클러스터에서 네임스페이스 서비스를 템플릿으로 프로비저닝 또는 활성화할 수 있습니다. 자세한 내용은 vSphere IaaS control plane에서 셀프 서비스 네임스페이스 템플릿 프로비저닝의 내용을 참조하십시오.

그림 2-2. 셀프 서비스 네임스페이스에 대한 vSphere 관리자 워크플로



DevOps 엔지니어는 셀프 서비스 방식으로 vSphere 네임스페이스를 생성하고 그 안에 워크로드를 배포할 수 있습니다. 다른 DevOps 엔지니어와 공유하거나 더 이상 필요하지 않으면 삭제할 수 있습니다.

그림 2-3. 셀프 서비스 네임스페이스에 대한 DevOps 워크플로

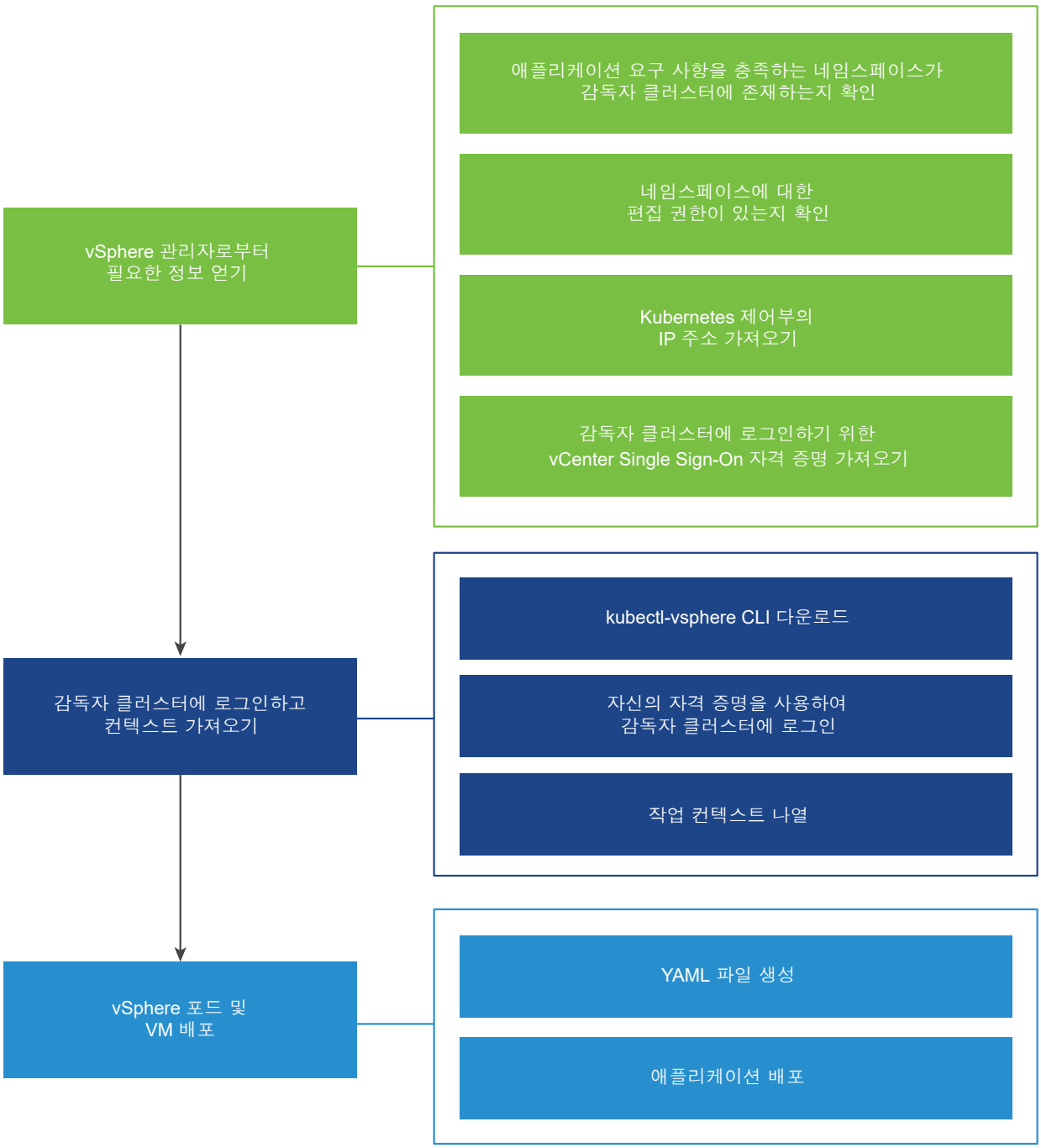


vSphere 포드 및 VM을 프로비저닝하는 워크플로

DevOps 엔지니어는 감독자에서 실행 중인 네임스페이스의 리소스 경계 내에서 vSphere 포드 및 VM을 배포할 수 있습니다.

자세한 내용은 장 7 vSphere 포드에 워크로드 배포 및 장 6 vSphere IaaS control plane에서 가상 시스템 배포 및 관리 항목을 참조하십시오.

그림 2-4. vSphere 포드 및 VM 프로비저닝 워크플로



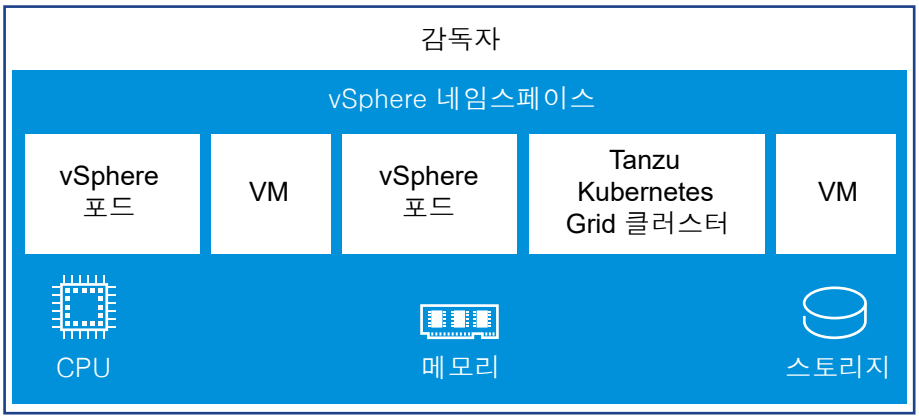
vSphere 네임스페이스 구성 및 관리

3

vSphere 포드, VM 및 Tanzu Kubernetes 클러스터를 포함하는 vSphere IaaS control plane 워크로드는 vSphere 네임스페이스에 배포됩니다. vSphere 관리자는 감독자에서 네임스페이스를 정의하고 리소스 할당량 및 사용자 권한을 사용하여 구성합니다. DevOps의 요구 사항 및 실행할 계획인 워크로드에 따라 vSphere 관리자는 VM 이미지를 가져오기 위해 스토리지 정책, VM 클래스 및 컨텐츠 라이브러리를 할당할 수도 있습니다.

처음 생성되면 네임스페이스는 감독자 내에 제한 없는 리소스가 포함됩니다. vSphere 관리자는 CPU, 메모리, 스토리지 및 네임스페이스 내에서 실행할 수 있는 Kubernetes 개체의 수에 대한 제한을 설정할 수 있습니다. 스토리지 제한은 Kubernetes에서 스토리지 할당량으로 표시됩니다. 리소스 풀은 감독자의 각 네임스페이스별로 vSphere에 생성됩니다.

vSphere 영역에서 활성화된 감독자에서 영역에 매핑된 각 vSphere 클러스터에 네임스페이스 리소스 풀이 생성됩니다. 3개 영역 감독자의 네임스페이스에 사용되는 리소스는 기본 vSphere 클러스터 3개 모두에서 동일한 양이 사용됩니다. 예를 들어 300MHz의 CPU를 할당하면 각 vSphere 클러스터에서 100MHz가 사용됩니다.



DevOps 엔지니어에게 네임스페이스에 대한 액세스를 제공하기 위해, vSphere 관리자는 vCenter Single Sign-On에 연결된 ID 소스 내에서 또는 감독자에 등록된 ODIC 제공자에서 사용 가능한 사용자나 사용자 그룹에 사용 권한을 할당합니다. 자세한 내용은 vSphere IaaS Control Plane ID 및 액세스 관리를 참조하십시오.

리소스, 개체 제한, 권한 및 스토리지 정책으로 네임스페이스를 생성하고 구성한 후 DevOps 엔지니어는 네임스페이스에 액세스하여 다음 워크로드를 실행할 수 있습니다.

- 감독자 서비스 실행에 대한 자세한 내용은 장 4 vSphere IaaS control plane를 사용하여 감독자 서비스 관리 항목을 참조하십시오.
- vSphere 포드 실행에 대한 자세한 내용은 장 7 vSphere 포드에 워크로드 배포 항목을 참조하십시오.

- VM 배포에 대한 자세한 내용은 장 6 vSphere IaaS control plane에서 가상 시스템 배포 및 관리 항목을 참조하십시오.

참고 이 "vSphere IaaS 제어부 서비스 및 워크로드" 가이드에 Tanzu Kubernetes Grid 클러스터에서 워크로드를 실행하는 데 대한 정보는 포함되어 있지 않습니다. Tanzu Kubernetes Grid 클러스터에서 작업하는 방법을 알아보려면 vSphere IaaS Control Plane과 함께 감독자에서 Tanzu Kubernetes Grid 사용을 참조하십시오.

다음으로 아래 항목을 읽으십시오.

- 감독자에서 vSphere 네임스페이스 생성 및 구성
- 감독자에서 vSphere 네임스페이스 제거
- vSphere 네임스페이스 에 대한 리소스 제한 설정
- vSphere 네임스페이스에 대한 개체 제한 사항 구성
- vSphere 네임스페이스에서 리소스 모니터링 및 관리
- vSphere IaaS control plane에서 셀프 서비스 네임스페이스 템플릿 프로비저닝
- vSphere 네임스페이스의 스토리지 설정 변경
- NSX vSphere 네임스페이스에 보안 정책 추가
- 네임스페이스에 대한 네트워크 및 로드 밸런서 매개 변수 구성

감독자에서 vSphere 네임스페이스 생성 및 구성

감독자에서 vSphere 네임스페이스를 생성하고 구성하는 방법을 알아봅니다. vSphere 관리자는 vSphere 네임스페이스를 생성한 후 DevOps 엔지니어가 액세스할 수 있도록 네임스페이스 및 사용 권한에 대한 리소스 제한을 설정합니다. 사용 권한이 있는 네임스페이스에서 워크로드를 실행할 수 있는 Kubernetes 제어부의 URL을 DevOps 엔지니어에게 제공합니다.

VDS 네트워킹 스택으로 구성된 감독자의 네임스페이스와 NSX로 구성된 클러스터의 네임스페이스는 네트워킹 구성 및 기능이 서로 다릅니다. 세 개의 vSphere 영역에 배포된 감독자에 구성된 네임스페이스는 단일 영역 감독자의 네임스페이스와 다른 기능 집합도 지원합니다.

- NSX로 구성된 1개 영역 감독자. 이러한 감독자의 vSphere 네임스페이스는 vSphere 포드, VM, Tanzu Kubernetes Grid 클러스터 및 감독자 서비스를 지원합니다. 이러한 vSphere 네임스페이스에 대한 워크로드 네트워킹 지원은 NSX에서 제공합니다.
- NSX로 구성된 3개 영역 감독자. NSX로 구성된 3개 영역 감독자의 vSphere 네임스페이스는 Tanzu Kubernetes Grid 클러스터 및 VM만 지원합니다. vSphere 포드 및 감독자 서비스는 지원하지 않습니다.
- VDS로 구성된 1개 영역 감독자. VDS가 있는 1개 영역 감독자의 vSphere 네임스페이스은 Tanzu Kubernetes Grid, VM 및 감독자 서비스를 지원합니다. 감독자 서비스가 자체 사용을 위해 배포하는 것 외에는 vSphere 포드를 지원하지 않습니다.
- VDS로 구성된 3개 영역 감독자. VDS가 있는 3개 영역 감독자를 실행하는 vSphere 네임스페이스는 Tanzu Kubernetes Grid 클러스터 및 VM만 지원합니다. vSphere 포드 및 감독자 서비스는 지원하지 않습니다.

자세한 내용은 "vSphere IaaS 제어부 개념 및 계획" 에서 HA 프록시 로드 밸런서를 사용하여 3개 영역 감독자를 사용하도록 설정하기 위한 요구 사항 및 VDS 네트워킹 및 HAProxy 로드 밸런서에서 단일 클러스터 감독자를 사용하도록 설정하기 위한 요구 사항을 참조하십시오.

또한 네임스페이스에 대한 리소스 제한을 설정하고, 사용 권한을 할당하고, 클러스터에서 네임스페이스 서비스를 템플릿으로 프로비저닝하거나 활성화할 수 있습니다. 그러면 DevOps 엔지니어는 셀프 서비스 방식으로 감독자 네임스페이스를 생성하고 그 안에 워크로드를 배포할 수 있습니다. 자세한 내용은 vSphere IaaS control plane에서 셀프 서비스 네임스페이스 템플릿 프로비저닝의 내용을 참조하십시오.

감독자에 NSX를 사용하는 경우 vSphere 네임스페이스 수준에서 네트워킹 설정을 재정의할 수 있는 옵션이 있습니다. 해당 옵션을 선택하는 경우 다음 고려 사항에 유의하십시오.

표 3-1. vSphere 네임스페이스 네트워크 계획 시 고려 사항

고려 사항	설명
NSX설치	특정 vSphere 네임스페이스에 대한 감독자 네트워크 설정을 재정의하려면 NSX에는 Tier-0 게이트웨이(라우터) 전용 Edge 클러스터와 Tier-1 게이트웨이 전용 다른 Edge 클러스터가 포함되어야 합니다. 가이드 "vSphere IaaS 제어부 설치 및 구성" 에 제공된 NSX 설치 지침을 참조하십시오.
IPAM 필요	특정 vSphere 네임스페이스에 대한 감독자 네트워크 설정을 재정의하는 경우 새 vSphere 네임스페이스 네트워크는 감독자 및 기타 vSphere 네임스페이스 네트워크에서 고유한 수신, 송신 및 네임스페이스 네트워크 서브넷을 지정해야 합니다. 이에 따라 IP 주소 할당을 관리해야 합니다.
감독자 라우팅	감독자는 TKG 클러스터 노드 및 수신 서브넷으로 직접 라우팅할 수 있어야 합니다. vSphere 네임스페이스에 대해 Tier-0 게이트웨이를 선택할 때 필요한 라우팅을 구성하는 두 가지 옵션이 있습니다. ■ VRF(가상 라우팅 및 전달) 게이트웨이를 사용하여 감독자 Tier-0 게이트웨이의 구성을 상속 ■ BGP(Border Gateway Protocol)를 사용하여 감독자 Tier-0 게이트웨이와 전용 Tier-0 게이트웨이 간의 경로를 구성 이러한 옵션에 대한 자세한 내용은 NSX Tier-0 게이트웨이 설명서를 참조하십시오.

사전 요구 사항

- 감독자를 배포합니다.
- vSphere 네임스페이스에 액세스해야 하는 DevOps 엔지니어 및 개발자를 위한 사용자 및 그룹을 생성합니다. vCenter Single Sign-On에 연결된 ID 소스에 또는 감독자로 구성된 OIDC 제공자에 사용자 또는 그룹을 생성합니다.
- 영구 스토리지에 대한 스토리지 정책을 생성합니다. 네임스페이스가 3개 영역 감독자에 있는 경우 토폴로지 인식 정책을 사용합니다. 토폴로지를 인식하지 못하는 스토리지 정책은 3개 영역 네임스페이스에 할당할 수 없습니다.
- 독립형 VM에 대한 VM 클래스 및 컨텐츠 라이브러리를 생성합니다.
- 필요한 권한:
  - **네임스페이스.클러스터 전체 구성 수정**
  - **네임스페이스.네임스페이스 구성 수정**

절차

1 vSphere Client 홈 메뉴에서 **워크로드 관리**를 선택합니다.

2 **네임스페이스** 탭을 선택합니다.

3 **네임스페이스 생성**을 클릭합니다.

4 vSphere 네임스페이스를 배치할 감독자를 선택합니다.

5 네임스페이스의 이름을 입력합니다.

이름은 DNS 호환 형식이어야 합니다.

6 **네트워크** 드롭다운 메뉴에서 vSphere 네임스페이스에 대한 워크로드 네트워크를 선택합니다.

참고 이 단계는 vSphere 네트워킹 스택으로 구성된 클러스터에서 네임스페이스를 생성하는 경우에만 사용할 수 있습니다.

7 감독자에 대한 NSX 네트워킹 스택을 구성한 경우 **클러스터 네트워크 설정 재정의**를 선택하여 감독자 네트워크 설정을 재정의하고 네임스페이스에 대한 네트워크 설정을 구성할 수 있습니다.

네임스페이스에 대해 다음 네트워크 설정을 구성합니다.

옵션	설명
Tier-0 게이트웨이	네임스페이스 Tier-1 게이트웨이와 연결할 Tier-0 게이트웨이를 선택합니다. Tier-0 게이트웨이를 선택하면 클러스터를 사용하도록 설정하는 동안 구성한 Tier-0 게이트웨이가 재정의되므로 CIDR 범위를 다시 구성해야 합니다. **참고** 감독자는 TKG 클러스터 노드 및 수신 서브넷으로 직접 라우팅할 수 있어야 합니다. ■ Tier-0 게이트웨이에 연결된 VRF 게이트웨이를 선택하면 네트워크 및 서브넷이 자동으로 구성됩니다. ■ NAT 모드를 선택한 경우에는 서브넷, 수신 및 송신 CIDR을 구성해야 합니다. ■ NAT 모드를 선택 취소하는 경우 서브넷 및 수신 CIDR만 구성해야 합니다. **참고** Tier-0 게이트웨이를 선택하면 변경할 수 없습니다.
NAT 모드	NAT 모드는 기본적으로 선택되어 있습니다. 이 옵션을 선택 취소하면 vSphere 포드, VM, Tanzu Kubernetes Grid 클러스터 노드 IP 주소와 같은 모든 워크로드를 Tier-0 게이트웨이 외부에서 직접 액세스할 수 있으며 송신 CIDR을 구성할 필요가 없습니다. **참고** 네임스페이스 모드를 사용하도록 설정한 후에는 변경할 수 없습니다.
로드 밸런서 크기	네임스페이스의 Tier-1 게이트웨이에서 로드 밸런서 인스턴스의 크기를 선택합니다.
네임스페이스 네트워크	서브넷/세그먼트를 생성하고 네임스페이스에 연결된 워크로드에 대한 IP 주소를 할당하도록 IP CIDR을 하나 이상 입력합니다. **참고** 클러스터에 대해 CIDR 범위를 구성하지 않은 경우 해당 범위를 입력합니다. 네임스페이스를 생성한 후 네임스페이스 네트워크 설정을 편집하여 추가 CIDR을 구성할 수 있습니다.

옵션	설명
네임스페이스 서브넷 접두사	네임스페이스 세그먼트용으로 예약된 서브넷의 크기를 지정하는 서브넷 접두사를 입력합니다. Default is 28. **참고** 서브넷 접두사를 지정한 후에는 변경할 수 없습니다.
수신	vSphere 포드 또는 Tanzu Kubernetes Grid 클러스터용 로드 밸런서 서비스에서 게시한 가상 IP 주소의 수신 IP 범위를 결정하는 CIDR 주석을 입력합니다. 네임스페이스를 생성한 후 네임스페이스 네트워크 설정을 편집하여 추가 CIDR을 구성할 수 있습니다.
송신	SNAT IP 주소의 송신 IP 범위를 결정하는 CIDR 주석을 입력합니다. 네임스페이스를 생성한 후 네임스페이스 네트워크 설정을 편집하여 추가 CIDR을 구성할 수 있습니다.

8 설명을 입력하고 **생성**을 클릭합니다.

감독자에 네임스페이스가 생성됩니다.

9 네임스페이스에 액세스할 수 있는 사용자의 사용 권한을 설정합니다.

vSphere 관리자는 네임스페이스에 액세스해야 하는 개발자 및 DevOps 엔지니어를 위해 vSphere 네임스페이스에 대한 사용 권한을 설정합니다. 하나의 사용자 계정이 여러 네임스페이스에 액세스할 수 있습니다. 관리자 그룹의 멤버인 사용자는 감독자의 모든 네임스페이스에 액세스할 수 있습니다.

a **사용 권한** 창에서 **사용 권한 추가**를 선택합니다.

b ID 소스, 사용자 또는 그룹 및 역할을 선택하고 **확인**을 클릭합니다.

역할	설명
볼 수 있음	사용자 또는 그룹에 대한 읽기 전용 액세스. 사용자 또는 그룹은 감독자 제어부에 로그인하고 vSphere 네임스페이스(예: vSphere 포드 및 Tanzu Kubernetes Grid 클러스터 및 VM)에서 실행 중인 워크로드를 나열할 수 있습니다.
편집할 수 있음	사용자 또는 그룹은 vSphere 포드, Tanzu Kubernetes Grid 클러스터 및 VM을 생성, 읽기, 업데이트 및 삭제할 수 있습니다. 관리자 그룹에 속한 사용자는 감독자의 모든 네임스페이스에 대한 편집 권한이 있습니다.
소유자	소유자 권한이 있는 사용자 계정은 다음을 수행할 수 있습니다. ■ 네임스페이스에서 워크로드를 배포하고 관리합니다. ■ 네임스페이스를 다른 사용자 또는 그룹과 공유합니다. ■ kubectl을 사용하여 추가 vSphere 네임스페이스를 생성하고 삭제합니다. 소유자 권한이 있는 사용자가 네임스페이스를 공유하면 다른 사용자 또는 그룹에 보기, 편집 또는 소유자 권한을 할당할 수 있습니다. **참고** 소유자 역할은 vCenter Single Sign-On ID 소스에서 사용할 수 있는 사용자에 대해 지원됩니다. 외부 ID 제공자의 사용자 또는 그룹에는 소유자 역할을 사용할 수 없습니다.

볼 수 있음 또는 **편집할 수 있음** 역할에 사용자 또는 그룹을 할당하면 시스템에서 RoleBinding이 생성되어 ClusterRole에 매핑됩니다. 예를 들어 **편집할 수 있음** 역할에 할당된 사용자 또는 그룹은 RoleBinding을 사용하여 Kubernetes `edit` ClusterRole에 매핑됩니다. `edit` 역할을 통해 사용자는 클러스터를 프로비저닝하고 운영할 수 있습니다. 이 매핑은 대상 vSphere 네임스페이스에서 `kubectl get rolebinding` 명령을 사용하여 볼 수 있습니다.

```
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
```

사용자 또는 그룹에 소유자 역할을 할당하면 시스템에서 ClusterRoleBinding이 생성되어 ClusterRole에 매핑됩니다. 그러면 사용자 또는 그룹이 kubectl을 사용하여 vSphere 네임스페이스를 생성하고 삭제할 수 있습니다. 이 매핑을 보려면 SSH를 통해 감독자 제어부 노드에 연결하면 됩니다.

10 네임스페이스에 스토리지를 할당합니다.

네임스페이스에 할당하는 스토리지 정책은 DevOps 팀에서 영구 스토리지를 사용할 수 있도록 합니다.

a **스토리지** 창에서 **스토리지 추가**를 선택합니다.

b 영구 볼륨의 데이터스토어 배치를 제어할 스토리지 정책을 선택하고 **확인**을 클릭합니다.

스토리지 정책을 할당하면 vSphere IaaS control plane은 vSphere 네임스페이스에 일치하는 Kubernetes 스토리지 클래스를 생성합니다. Tanzu Kubernetes Grid를 사용하는 경우 스토리지 클래스는 네임스페이스에서 Tanzu Kubernetes Grid 클러스터로 자동 복제됩니다. 네임스페이스에 여러 스토리지 정책을 할당할 때 각 스토리지 정책에 대해 별도의 스토리지 클래스가 생성됩니다.

11 [용량 및 사용량] 창에서 **제한 편집**을 선택하고 네임스페이스에 대한 리소스 제한을 구성합니다.

옵션	설명
CPU	네임스페이스에 대해 예약할 CPU 리소스 양입니다.
메모리	네임스페이스에 대해 예약할 메모리 양입니다.
스토리지	네임스페이스에 대해 예약할 총 스토리지 공간의 양입니다.
스토리지 정책 제한	네임스페이스와 연결한 각 스토리지 정책에 개별적으로 전용 스토리지 양을 설정합니다.

네임스페이스에 대한 리소스 풀이 vCenter Server에 생성됩니다. 스토리지 제한에 따라 네임스페이스에 사용할 수 있는 전체 스토리지 양이 결정되지만 스토리지 정책은 연결된 스토리지 클래스에서 vSphere 포드에 대한 영구 볼륨의 배치를 결정합니다.

12 독립형 VM에 대한 VM 서비스를 설정합니다.

자세한 내용은 장 6 vSphere IaaS control plane에서 가상 시스템 배포 및 관리 항목을 참조하십시오.

다음에 수행할 작업

Kubernetes 제어부 URL을 DevOps 엔지니어와 공유하고 vSphere에 대한 Kubernetes CLI 도구를 통해 감독자에 로그인하는 데 사용할 수 있는 사용자 이름도 공유합니다. DevOps 엔지니어에게 둘 이상의 네임스페이스에 대한 액세스 권한을 부여할 수 있습니다. vSphere IaaS Control Plane 클러스터에 연결을 참조하십시오.

참고 이 "vSphere IaaS 제어부 서비스 및 워크로드" 가이드에 Tanzu Kubernetes Grid 클러스터에서 워크로드를 실행하는 데 대한 정보는 포함되어 있지 않습니다. Tanzu Kubernetes Grid 클러스터에서 작업하는 방법을 알아보려면 vSphere IaaS Control Plane과 함께 감독자에서 Tanzu Kubernetes Grid 사용을 참조하십시오.

감독자에서 vSphere 네임스페이스 제거

감독자에서 vSphere 네임스페이스를 제거할 수 있습니다.

사전 요구 사항

- VM, vSphere 포드 및 TKG 클러스터를 포함하여 배포된 워크로드를 제거합니다. TKG 클러스터 제거에 대한 자세한 내용은 Kubectl 또는 Tanzu CLI를 사용하여 TKG 2.0 클러스터 삭제를 참조하십시오.

- 필요한 권한:
  - **네임스페이스.클러스터 전체 구성 수정**
  - **네임스페이스.네임스페이스 구성 수정**

절차

1 vSphere Client 홈 메뉴에서 **워크로드 관리**를 선택합니다.

2 **네임스페이스** 탭을 클릭합니다.

3 감독자에서 사용 가능한 네임스페이스 목록에서 제거할 vSphere 네임스페이스를 선택합니다.

4 **제거**를 클릭합니다.

시스템에서 vSphere 네임스페이스가 제거됩니다. 프로세스가 완료되는 데 시간이 걸릴 수 있습니다.

vSphere 네임스페이스 에 대한 리소스 제한 설정

vSphere 관리자는 vSphere 네임스페이스에서 리소스 제한 및 컨테이너 기본값을 설정할 수 있습니다. DevOps 엔지니어는 나중에 vSphere 관리자가 네임스페이스에 설정한 총 리소스 제한을 초과하지 않고도 포드 규격에서 컨테이너 기본값을 재정의할 수 있습니다. 컨테이너 요청은 포드의 리소스 예약으로 변환됩니다.

사전 요구 사항

- 감독자에서 **네임스페이스 구성 수정** 권한이 있는지 확인합니다.

절차

1 vSphere Client 홈 메뉴에서 **워크로드 관리**를 선택합니다.

2 vSphere 네임스페이스를 선택하고 **구성**을 선택하고 **리소스 제한**을 클릭합니다.

3 **편집**을 클릭합니다.

Tanzu Kubernetes Grid 클러스터가 프로비저닝된 vSphere 네임스페이스에 대한 리소스 제한 설정의 영향은 클러스터 노드에 사용되는 VM 클래스 유형에 따라 다릅니다. 리소스 제한을 설정하기 전에 사용 시도와 보장됨의 차이점을 알고 있어야 합니다. "vSphere IaaS 제어부에서 TKG 서비스 사용" 에서 Tanzu Kubernetes 클러스터의 가상 시스템 클래스를 참조하십시오.

옵션	설명
CPU	vSphere 네임스페이스의 CPU 사용량에 대한 제한을 설정합니다.
메모리	vSphere 네임스페이스의 메모리 사용량에 대한 제한을 설정합니다.
스토리지	사용되는 스토리지 정책당 vSphere 네임스페이스의 스토리지 사용량에 대한 제한을 설정합니다.
컨테이너 기본값	vSphere 네임스페이스에서 CPU 제한, CPU 요청, 메모리 요청 및 컨테이너의 메모리 제한에 대한 기본값을 설정합니다.

vSphere 네임스페이스에 대한 개체 제한 사항 구성

vSphere 네임스페이스에서 실행 중인 개체에 대한 제한 사항(배포, 작업, 데몬 집합, 상태 저장 집합 등의 수)을 구성할 수 있습니다. 개체에 대해 구성하는 제한 사항은 애플리케이션의 세부 사항과 vSphere 네임스페이스 내에서 리소스를 사용하려는 방식에 따라 달라집니다.

사전 요구 사항

- 감독자에서 **네임스페이스 구성 수정** 권한이 있는지 확인합니다.

절차

1. vSphere Client 홈 메뉴에서 **워크로드 관리**를 선택합니다.
2. 개체 또는 컨테이너 제한 사항을 적용할 vSphere 네임스페이스를 선택합니다.
3. **구성**을 선택하고 **개체 제한**을 선택합니다.
4. **편집**을 클릭합니다.

옵션	설명
vSphere 포드	vSphere 네임스페이스에서 실행할 수 있는 vSphere 포드의 수입니다.
배포	vSphere 네임스페이스에서 실행할 수 있는 배포의 수입니다.
작업	vSphere 네임스페이스에서 실행할 수 있는 작업 수입니다.
데몬 집합	vSphere 네임스페이스에서 실행할 수 있는 데몬 집합의 수입니다.
복제 집합	vSphere 네임스페이스의 복제 집합 수입니다.
복제 컨트롤러	vSphere 네임스페이스에서 실행할 수 있는 복제 컨트롤러의 수입니다.
상태 저장 집합	vSphere 네임스페이스에서 실행할 수 있는 StatefulSet의 수입니다.
구성 맵	vSphere 네임스페이스에서 실행할 수 있는 ConfigMap의 수입니다.
암호	vSphere 네임스페이스에서 실행할 수 있는 암호의 수입니다.
영구 볼륨 할당	vSphere 네임스페이스에 존재할 수 있는 영구 볼륨 할당입니다.
서비스	vSphere 네임스페이스에 존재할 수 있는 서비스입니다.

vSphere 네임스페이스에서 리소스 모니터링 및 관리

네임스페이스에 대한 리소스 사용량과 네임스페이스에 존재하는 다양한 Kubernetes 개체의 수 및 상태와 같은 vSphere 네임스페이스의 다양한 측면을 모니터링하고 관리할 수 있습니다.

사전 요구 사항

vSphere 네임스페이스를 생성하고 구성합니다.

절차

1. vSphere Client를 사용하여 vCenter Server에 로그인합니다.

2 **메뉴 > 호스트 및 클러스터** 보기로 이동합니다.

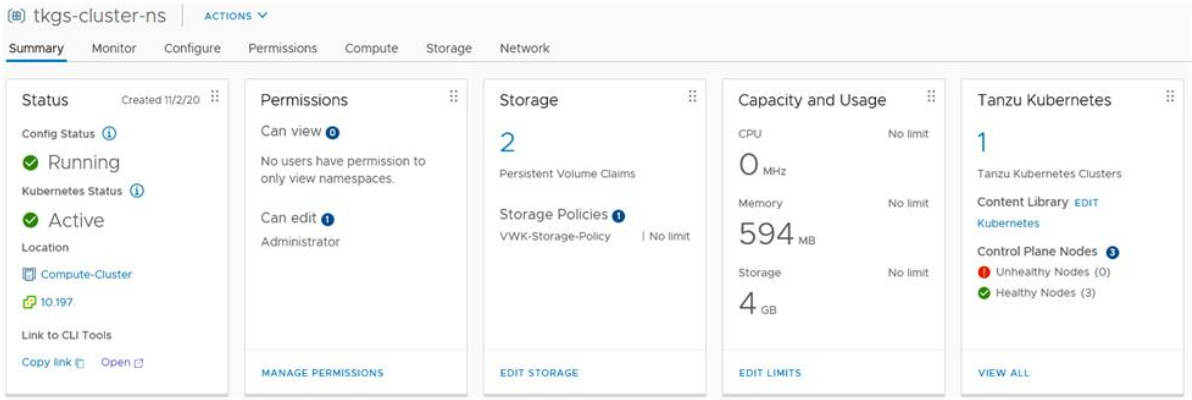
3 **워크로드 관리**를 사용하도록 설정한 vCenter 클러스터를 선택합니다.

4 **네임스페이스** 리소스 풀을 선택하고 해당 컨텐츠를 확장합니다.

감독자 제어부 노드는 네임스페이스 리소스 풀에 있습니다. 또한 이 감독자에 대해 생성된 각 vSphere 네임스페이스는 **네임스페이스** 리소스 풀에 있습니다.

5 창 아이콘으로 표시되는 vSphere 네임스페이스 개체를 선택합니다.

요약 탭에 **상태**, **사용 권한**, **스토리지**, **용량 및 사용량**, **Tanzu Kubernetes** 등 vSphere 네임스페이스에 대한 다양한 구성 섹션이 표시됩니다. 이 화면에서 이러한 설정을 관리할 수 있습니다.



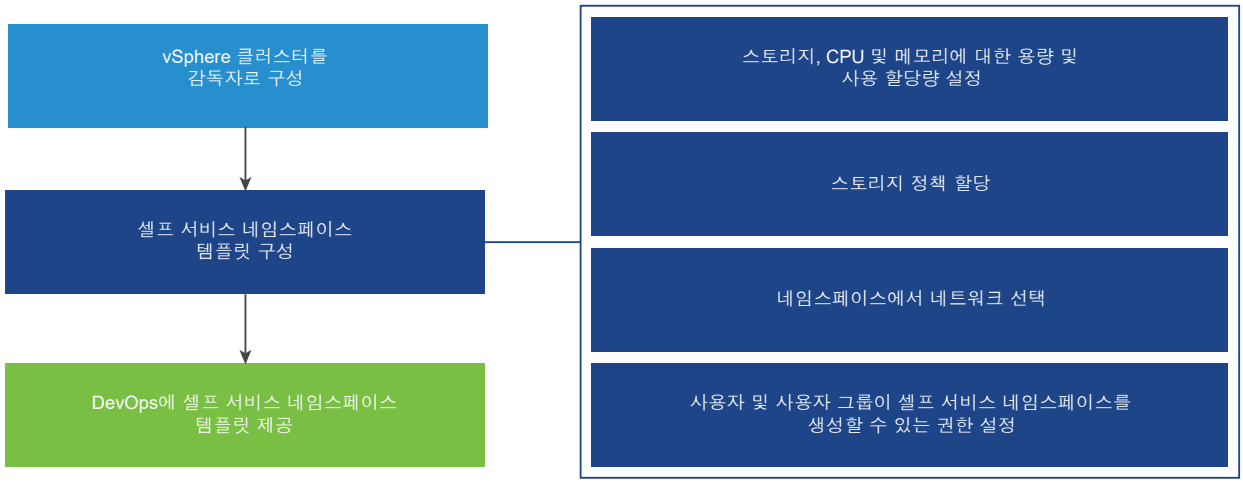
vSphere IaaS control plane에서 셀프 서비스 네임스페이스 템플릿 프로비저닝

vSphere 관리자는 감독자 네임스페이스를 생성하고, CPU, 메모리 및 스토리지 제한을 네임스페이스에 설정하고, 사용 권한을 할당하고, 클러스터에서 네임스페이스 서비스를 템플릿으로 활성화할 수 있습니다. 그러면 DevOps 엔지니어는 셀프 서비스 방식으로 감독자 네임스페이스를 생성하고 그 안에 워크로드를 배포할 수 있습니다.

셀프 서비스 네임스페이스 생성 및 구성 워크플로

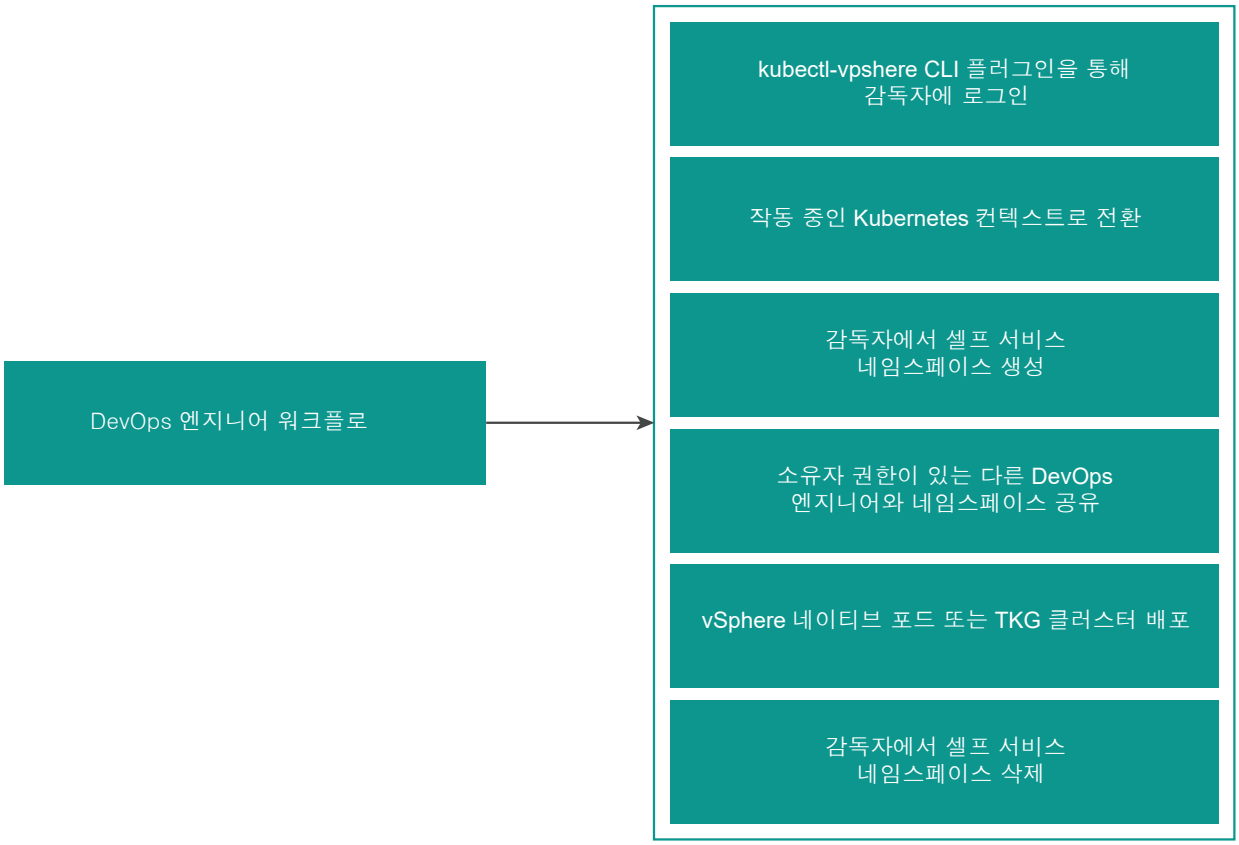
vSphere 관리자는 감독자 네임스페이스를 생성하고, CPU, 메모리 및 스토리지 제한을 네임스페이스에 설정하고, 사용 권한을 할당하고, 클러스터에서 네임스페이스 서비스를 템플릿으로 프로비저닝 또는 활성화할 수 있습니다.

그림 3-1. 셀프 서비스 네임스페이스 템플릿 프로비저닝 워크플로



DevOps 엔지니어는 셀프 서비스 방식으로 감독자 네임스페이스를 생성하고 그 안에 워크로드를 배포할 수 있습니다. 다른 DevOps 엔지니어와 공유하거나 더 이상 필요하지 않으면 삭제할 수 있습니다. 네임스페이스를 다른 DevOps 엔지니어와 공유하려면 vSphere 관리자에게 문의하십시오.

그림 3-2. 셀프 서비스 네임스페이스 생성 워크플로



셀프 서비스 네임스페이스 템플릿 생성 및 구성

vSphere 관리자는 감독자 네임스페이스를 셀프 서비스 네임스페이스 템플릿으로 생성하고 구성할 수 있습니다. 그런 다음 DevOps 엔지니어는 `kubectl` 명령줄을 사용하여 감독자 네임스페이스를 생성하고 삭제할 수 있습니다.

사전 요구 사항

vSphere IaaS control plane으로 클러스터를 구성합니다.

절차

1. vSphere Client에서 감독자로 이동합니다.
2. **구성** 탭을 클릭하고 **감독자**에서 **일반**을 선택합니다.
3. **네임스페이스 서비스**를 선택합니다.
4. **상태** 스위치를 전환하여 기능을 사용하도록 설정합니다.

   네임스페이스 템플릿 생성 페이지가 나타납니다.
5. **구성** 창에서 네임스페이스에 대한 리소스를 구성합니다.

옵션	설명
CPU	네임스페이스에 대해 예약할 CPU 리소스 양입니다.
메모리	네임스페이스에 대해 예약할 메모리 양입니다.
스토리지	네임스페이스에 대해 예약할 총 스토리지 공간의 양입니다.
스토리지 정책	영구 스토리지가 필요한 워크로드에 사용할 스토리지 정책입니다.
네트워크	**네트워크** 드롭다운 메뉴에서 네임스페이스에 대한 네트워크를 선택합니다.
VM 클래스	독립형 VM을 배포하기 위한 VM 클래스입니다.
컨텐츠 라이브러리	VM 배포에 사용할 VM 이미지가 있는 컨텐츠 라이브러리입니다.

6. **다음**을 클릭합니다.
7. **사용 권한** 창에서 DevOps 엔지니어 및 그룹을 추가하여 이들이 템플릿을 사용하여 네임스페이스를 생성할 수 있도록 합니다.

   ID 소스와 사용자 또는 그룹을 선택하고 **다음**을 클릭합니다.
8. **검토 및 확인** 창에 구성한 속성이 표시됩니다.

   속성을 검토하고 **완료**를 클릭합니다.

결과

네임스페이스 템플릿이 구성되었고 활성 상태입니다. 관리자 vSphere 템플릿을 편집할 수 있습니다. DevOps 엔지니어는 템플릿을 사용하여 네임스페이스를 생성할 수 있습니다.

셀프 서비스 네임스페이스 비활성화

vSphere 관리자는 클러스터에서 셀프 서비스 네임스페이스를 비활성화할 수 있습니다.

셀프 서비스 네임스페이스 템플릿을 비활성화하면 DevOps 엔지니어가 템플릿을 사용하여 클러스터에 새 네임스페이스를 생성할 수 없습니다. 이미 생성한 네임스페이스를 삭제할 수 있습니다.

절차

1 vSphere Client에서 감독자로 이동합니다.

2 **구성** 탭을 클릭하고 **감독자**에서 **일반**을 선택합니다.

3 **네임스페이스 서비스** 창에서 **상태** 스위치를 전환하여 템플릿을 비활성화합니다.

4 템플릿을 다시 활성화하려면 **상태** 스위치를 전환합니다.

다른 셀프 서비스 네임스페이스를 생성하거나 기존 네임스페이스를 사용할 수 있습니다.

셀프 서비스 네임스페이스 생성

DevOps 엔지니어는 셀프 서비스 네임스페이스를 생성하고 그 안의 워크로드를 실행할 수 있습니다. 네임스페이스를 생성한 후에는 다른 DevOps 엔지니어와 공유하거나 더 이상 필요하지 않으면 삭제할 수 있습니다.

사전 요구 사항

- vSphere 관리자가 클러스터에서 셀프 서비스 네임스페이스 템플릿을 생성하고 활성화했는지 확인합니다. 셀프 서비스 네임스페이스 템플릿 생성 및 구성 의 내용을 참조하십시오.
- 셀프 서비스 네임스페이스 템플릿의 사용 권한 목록에 개별적으로 또는 그룹의 멤버로 추가되었는지 확인합니다.
- 감독자 제어부의 IP 주소를 가져옵니다.

절차

1 kubectl용 vSphere 플러그인을 사용하여 감독자로 인증합니다. vCenter Single Sign-On 사용자로 감독자에 연결을 참조하십시오.

```
kubectl vsphere login --server=IP-ADDRESS --vsphere-username USERNAME
```

2 컨텍스트를 감독자로 전환합니다.

```
kubectl config use-context SUPERVISOR-CLUSTER-IP
```

3 클러스터에 셀프 서비스 네임스페이스를 생성합니다.

```
kubectl create namespace NAMESPACE NAME
```

예

```
kubectl create namespace test-ns
```

참고 소유자 권한은 vSphere IaaS control plane를 사용하도록 설정하고 클러스터를 업그레이드한 후에 DevOps 엔지니어가 사용할 수 있습니다. 클러스터가 아닌 vCenter Server만 업그레이드하는 경우 Devops 엔지니어는 네임스페이스에 대한 편집 권한만 갖게 됩니다.

생성한 네임스페이스가 클러스터에 표시됩니다. 네임스페이스를 다른 DevOps 엔지니어와 공유하려면 vSphere 관리자에게 문의하십시오.

주석 및 레이블이 있는 셀프 서비스 네임스페이스 생성

DevOps 엔지니어는 kubectl 명령줄을 사용하여 주석 및 레이블이 있는 셀프 서비스 네임스페이스를 생성할 수 있습니다.

DevOps 엔지니어는 사용자 정의 주석 및 레이블이 있는 YAML 매니페스트를 사용할 수 있습니다.

절차

1 감독자에 로그인합니다.

```
kubectl vsphere login --server IP-ADDRESS-SUPERVISOR-CLUSTER --vsphere-username VCENTER-SSO-USERNAME
```

2 주석 및 레이블이 있는 네임스페이스 YAML 매니페스트 파일을 생성합니다.

```
kubectl create -f ns-create.yaml
```

예를 들어 다음과 같은 `ns-create.yaml` 파일을 생성합니다.

```
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 매니페스트를 적용합니다.

```
kubectl create -f ns-create.yaml
```

또는

```
kubectl apply -f ns-create.yaml
```

4 변경 내용을 확인하기 위해 생성한 네임스페이스에 대해 설명합니다.

```
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 및 kubectl label을 사용하여 셀프 서비스 네임스페이스 업데이트

DevOps 엔지니어는 `kubectl annotate` 및 `kubectl label` 명령을 사용하여 셀프 서비스 네임스페이스 주석 및 레이블을 업데이트하거나 삭제할 수 있습니다.

사전 요구 사항

업데이트하려는 네임스페이스에 대한 소유자 권한이 있는지 확인합니다.

절차

1 감독자에 로그인합니다.

```
kubectl vsphere login --server IP-ADDRESS-SUPERVISOR-CLUSTER --vsphere-
username VCENTER-SSO-USERNAME
```

2 업데이트할 네임스페이스에 대해 설명합니다.

```
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` 명령을 사용하여 주석을 업데이트합니다.

예를 들어 `kubectl annotate --overwrite ns testns my-ann="test-ann-3"`입니다.

주석을 삭제하려면 `kubectl annotate --overwrite ns testns my-ann-` 명령을 실행합니다.

4 `kubectl label` 명령을 사용하여 레이블을 업데이트합니다.

예를 들어 `kubectl label --overwrite ns testns my-label="test-label-3"`입니다.

레이블을 삭제하려면 `kubectl label --overwrite ns testns my-label-` 명령을 실행합니다.

5 업데이트를 확인할 네임스페이스에 대해 설명합니다.

```
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 edit을 사용하여 셀프 서비스 네임스페이스 업데이트

DevOps 엔지니어는 `kubectl edit` 명령을 사용하여 셀프 서비스 네임스페이스를 업데이트할 수 있습니다.

사전 요구 사항

업데이트하려는 네임스페이스에 대한 소유자 권한이 있는지 확인합니다.

절차

1 감독자에 로그인합니다.

```
kubectl vsphere login --server IP-ADDRESS-SUPERVISOR-CLUSTER --vsphere-
username VCENTER-SSO-USERNAME
```

2 업데이트할 네임스페이스에 대해 설명합니다.

```
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` 명령을 사용하여 네임스페이스를 편집합니다.

예를 들어 `kubectl edit ns testns-1`입니다.

`kubectl edit` 명령은 KUBE_EDITOR 또는 EDITOR 환경 변수로 정의된 네임스페이스 매니페스트를 텍스트 편집기에서 엽니다.

4 레이블을 업데이트합니다.

예를 들어 `my-label=test-label`입니다.

5 주석을 업데이트합니다.

예를 들어 `my-ann: test-ann`입니다.

6 업데이트를 확인할 네임스페이스에 대해 설명합니다.

```
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.
```

셀프 서비스 네임스페이스 삭제

DevOps 엔지니어는 사용자가 생성한 셀프 서비스 네임스페이스를 삭제할 수 있습니다.

사전 요구 사항

kubectl용 vSphere 플러그인을 사용하여 셀프 서비스 네임스페이스를 생성했는지 확인합니다.

절차

1 kubectl용 vSphere 플러그인을 사용하여 감독자로 인증합니다. vCenter Single Sign-On 사용자로 감독자에 연결을 참조하십시오.

2 클러스터에서 셀프 서비스 네임스페이스를 삭제합니다.

```
kubectl delete namespace NAMESPACE NAME
```

예:

```
kubectl delete namespace test-ns
```

vSphere 네임스페이스의 스토리지 설정 변경

감독자의 네임스페이스에 할당된 스토리지 정책을 통해 DevOps 팀이 영구 스토리지를 사용할 수 있습니다. 이러한 스토리지 정책은 영구 볼륨 및 Tanzu Kubernetes 클러스터 노드가 vSphere 데이터스토어 내에 배치되는 방식을 제어합니다. vSphere 관리자는 일반적으로 네임스페이스를 구성할 때 스토리지 정책을 할당합니다. 초기 스토리지 정책 할당을 변경해야 하는 경우 다음 작업을 수행합니다.

사전 요구 사항

- VMware vCenter 또는 vSphere 네임스페이스에서 스토리지 정책을 삭제하거나 스토리지 정책 할당을 변경하기 전에 해당 스토리지 클래스가 포함된 영구 볼륨 할당이 네임스페이스에서 실행되지 않도록 해야 합니다. 또한 스토리지 클래스를 사용하는 Tanzu Kubernetes 클러스터가 없는지 확인합니다.
- 네임스페이스가 3개 영역 감독자에 있는 경우 토폴로지 인식 정책을 사용합니다. 토폴로지를 인식하지 못하는 스토리지 정책은 3개 영역 네임스페이스에 할당할 수 없습니다.

절차

1 vSphere Client에서 네임스페이스로 이동합니다.

   a vSphere Client 홈 메뉴에서 **워크로드 관리**를 선택합니다.

   b **네임스페이스** 탭을 클릭하고 네임스페이스를 클릭합니다.

2 **스토리지** 탭을 클릭하고 **스토리지 정책**을 클릭합니다.

3 **편집** 아이콘을 클릭하여 스토리지 정책 할당을 변경합니다.

NSX vSphere 네임스페이스에 보안 정책 추가

NSX 네트워킹을 사용하는 감독자는 보안 정책 CRD를 통해 구성된 네트워크 보안 정책을 지원합니다.

보안 정책 생성

DevOps는 NSX 기반 보안 정책을 감독자 네임스페이스에 적용하도록 보안 정책 CRD를 구성할 수 있습니다. 보안 정책은 vSphere 포드 및 VM에 대한 트래픽을 보호합니다. VM에는 TKG 클러스터 노드 및 감독자에 배포된 기타 VM이 포함됩니다.

사전 요구 사항

NSX 버전 3.2 이상을 사용합니다.

절차

1 보안 정책 CRD를 생성합니다.

사용할 필드 및 CRD 예는 GitHub에서 NSX 운영자 보안 정책 CRD 설명서를 참조하십시오.

2 Kubernetes 환경의 네임스페이스에 액세스합니다.

감독자 컨텍스트 가져오기 및 사용을 참조하십시오.

3 보안 정책을 네임스페이스에 적용합니다.

```
kubectl apply -f policy-name.yaml
```

4 보안 정책을 살펴봅니다.

a 보안 정책에 대한 세부 정보를 살펴봅니다.

```
kubectl get securitypolicy policy-name
```

b 보안 정책에 대한 설명을 살펴봅니다.

```
kubectl describe securitypolicy policy-name
```

결과

NSX UI를 사용하여 정책의 세부 정보를 볼 수도 있습니다. 자세한 내용은 "VMware NSX 설명서" 페이지를 참조하십시오.

네임스페이스에 대한 네트워크 및 로드 밸런서 매개 변수 구성

vSphere IaaS control plane는 NCP구성 파일 `ncp.ini` 편집을 지원하지 않습니다. NCP에서 CRD(CustomResourceDefinition)를 생성하여 네트워크 및 로드 밸런서 매개 변수를 구성할 수 있습니다.

NCPSetting CRD

NCPSetting CRD를 생성하고 NCP 구성에 대한 값을 설정합니다.

다음 표에서는 구성할 수 있는 네트워크 및 로드 밸런서 매개 변수에 대해 설명합니다.

매개 변수	설명
log_dropped_traffic	분산 방화벽 DENY 규칙이 로깅되었는지 여부를 나타냅니다. 값: True, False 기본값은 false입니다.
log_firewall_traffic	DFW 규칙이 로깅되었는지 여부를 나타냅니다. 값은 다음과 같습니다. ■ ALL. 모든 DFW 규칙에 대한 로깅을 사용하도록 설정합니다. ■ DENY. 거부 규칙에 대해서만 로깅을 사용하도록 설정합니다.
pool_algorithm	로드 밸런서 풀 개체에서 로드 밸런싱 알고리즘을 설정하는 옵션입니다. 값은 다음과 같습니다. ■ Round_Robin ■ Weighted_Round_Robin ■ Least_Connection ■ Weighted_Least_Connection ■ IP-Hash 기본값은 Round-Robin입니다.
service_size	로드 밸런서 크기를 설정하는 옵션입니다. 값은 Small, Medium, Large입니다. 기본값은 Small입니다.
l7_persistence	로드 밸런서 지속성 옵션을 설정하는 옵션입니다. 값은 다음과 같습니다. ■ cookie. ■ source_ip
l7_persistence_timeout	L7 지속성 프로파일의 지속성 시간 초과 값(초)입니다.
cookie_name	l7_persistence type이 cookie로 설정되면 쿠키 이름을 지정합니다.
x_forward_for	수신에서 X_forward_for 헤더를 사용하도록 설정합니다. 값은 다음과 같습니다. ■ Insert ■ Replace
snat_rule_logging	SNAT 규칙에 대한 로깅을 선택하는 옵션입니다. 값은 다음과 같습니다. ■ None ■ Basic. 모든 네임스페이스에 대한 로깅. ■ Extended. 모든 네임스페이스 및 서비스에 대한 로깅.

매개 변수	설명
vs_access_log	수신 및 경로에 대한 가상 서버의 로그 속성입니다. 값은 다음과 같습니다. ■ VS_access_log_none ■ access_log_enabled. 계층 7 가상 서버에 대한 로깅을 사용하도록 설정합니다. ■ log_significant_event_only. HTTP 응답 상태가 400 이상인 요청은 중요한 이벤트로 처리됩니다. 기본값은 VS_access_log_none입니다.
ip_reallocation_time	해제된 IP를 재할당할 수 있을 때까지의 시간(초)입니다.

NCP 및 NSX 개체에 대한 자세한 내용은 "NSX" 설명서를 참조하십시오.

다음 단계를 수행하여 이 기능을 사용하도록 설정합니다.

1 **enable_ncp_setting_crd**를 **True**로 설정합니다.

2 다음 템플릿을 사용하여 YAML 파일을 생성합니다.

```
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
```

예:

```
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 다음 명령을 사용하여 YAML 파일을 적용합니다.

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

여러 CRD가 동일한 구성 값을 재정의하는 것을 방지하기 위해 NCP는 이름이 **ncp-setting-crd**인 CRD 개체만 처리합니다. 다른 이름을 가진 다른 CRD에는 오류 주석이 추가되고 그러한 CRD는 NCP가 처리하지 않습니다.

NCP 구성 재정의

CRD의 구성 매개 변수에는 해당하는 NSX 개체가 있을 수 있습니다. 매개 변수를 재정의하기 위해 CRD를 생성하는 경우 CRD는 다음 경우를 제외하고 개체의 매개 변수를 변경하지 않습니다.

- `l7_persistence`, `l7_persistence_timeout` 및 `cookie_name`. CRD에 따라 `l7_persistence`가 변경되면 NCP는 `l7_persistence`, `l7_persistence_timeout` 및 `cookie_name` 값으로 새 지속성 프로파일을 생성합니다.

  CRD를 통해 `l7_persistence_timeout` 및 `cookie_name`이 변경되면 새 값을 기반으로 기존 프로파일이 업데이트됩니다.

- `x_forwarded_for`. CRD에 따라 `x_forwarded_for`가 변경되면 NCP는 해당 값을 기반으로 새 애플리케이션 프로파일을 생성합니다.

- `vs_access_log`. CRD에 따라 `vs_access_log`가 변경되면 NCP는 그에 따라 가상 서버의 로깅 옵션을 업데이트합니다.

vSphere IaaS control plane를 사용하여 감독자 서비스 관리

4

감독자 서비스는 IaaS(Infrastructure-as-a-Service) 구성 요소 및 긴밀하게 통합된 ISV(독립 소프트웨어 벤더) 서비스를 개발자에게 제공하는 vSphere 인증 Kubernetes 운영자입니다. 감독자 서비스를 워크로드에서 사용할 수 있도록 vSphere IaaS control plane 환경에서 설치하고 관리할 수 있습니다.

감독자 서비스가 감독자에 설치되어 있으면 DevOps 엔지니어가 다음과 같은 다양한 방식으로 사용할 수 있습니다.

- Harbor와 같은 공유 감독자 서비스는 TKG 클러스터, vSphere 포드 또는 VM에서 실행되는 워크로드에 기능을 직접 제공합니다.
- MinIO와 같은 Operator를 포함하는 감독자 서비스는 일반적으로 DevOps 엔지니어가 CRD를 통해 vSphere 네임스페이스에서 서비스의 인스턴스를 생성하고 관리하는 데 사용할 수 있는 API 또는 그래픽 인터페이스를 제공합니다. 예를 들어 MinIO 버킷을 생성하려면 CRD를 사용하여 vSphere 네임스페이스에 버킷을 생성합니다.

지원되는 감독자 서비스에 대한 자세한 내용 및 해당 서비스 YAML 파일을 다운로드하는 방법은 http://vmware.com/go/supervisor-service 항목을 참조하십시오.

감독자 서비스를 사용한 지원되는 감독자 배포

감독자 서비스 서비스는 vSphere 포드로 배포됩니다. vSphere 8.0 릴리스에서는 NSX 네트워킹 스택으로 구성된 감독자만 vSphere 포드 및 감독자 서비스를 각각 지원합니다. vSphere 8 업데이트 1 릴리스부터 감독자 서비스를 통해 배포된 vSphere 포드는 두 가지 유형(NSX 또는 VDS)의 네트워킹으로 배포된 감독자에서 지원됩니다.

참고 감독자가 VDS 네트워킹 스택으로 구성된 경우 NSX 지원 네트워크(NSX에 의해 생성된 분산 포트 그룹)에서 감독자 서비스를 실행할 수 없습니다.

다음 표에는 vSphere 8 이상에 대한 기존 감독자 배포에서 감독자 서비스를 통해 배포된 vSphere 포드에 대한 지원이 나열되어 있습니다.

vSphere 버전	NSX 네트워킹	VDS 네트워킹	감독자 버전	1개 영역 감독자	3개 영역 감독자
vSphere 8	예	아니요	1.23 이상	예	아니요
vSphere 8.0.1 이상	예	예	1.24 이상	예	아니요
vSphere 8.0.3 이상	예	예	1.28 이상	예	예

감독자 서비스 수명 주기 관리

감독자 서비스는 vSphere Client에서 관리합니다. 감독자에 감독자 서비스를 설치하거나, 해당 버전을 업그레이드하거나, 감독자에서 감독자 서비스를 제거할 수 있습니다. 감독자 서비스는 vCenter Server에 여러 버전을 등록할 수 있지만 감독자에는 한 번에 하나의 버전만 설치할 수 있습니다.

표 4-1. 감독자 서비스 상태

상태	서비스 버전	전체 서비스
활성	서비스 버전을 감독자에 설치할 준비가 되었습니다.	하나 이상의 서비스 버전이 활성 상태입니다.
비활성화됨	서비스 버전을 감독자에 설치할 수 없습니다. 설치되어 있는 감독자에서 계속 실행할 수 있지만 비활성화된 버전을 새 감독자에 설치할 수는 없습니다.	전체 감독자 서비스가 비활성화되면 모든 해당 버전도 비활성화되고 서비스를 다시 활성화할 때까지는 감독자에 설치하거나 새 서비스 버전을 추가할 수 없습니다.

감독자 서비스의 수명 주기 관리에는 다음 작업이 포함됩니다.

작업	설명
vCenter Server에 새 감독자 서비스 추가	새 서비스를 vCenter Server에 추가하면 서비스 및 서비스에 대한 모든 정보가 vCenter Server에 등록됩니다. 서비스가 아직 감독자에 설치되지 않았습니다. 서비스가 vCenter Server에 등록되면 활성 상태가 됩니다. 그러면 해당 서비스를 감독자에 설치할 수 있습니다.
vCenter Server에 새 감독자 서비스 버전 추가	감독자 서비스를 vCenter Server에 추가한 후에는 해당 서비스의 새 버전을 추가할 수 있습니다. 새 서비스 버전이 vCenter Server에 등록되면 활성 상태가 되고 해당 버전을 감독자에 설치할 수 있습니다.
감독자에 감독자 서비스 설치	감독자에 감독자 서비스를 설치하면 서비스 YAML 파일이 감독자에 적용되고 서비스가 작동하기 위해 필요한 리소스와 모든 vSphere 포드가 생성됩니다. 감독자에 설치하는 각 감독자 서비스에 대해 vSphere 네임스페이스가 자동으로 생성됩니다. 해당 vSphere 네임스페이스에서 서비스 리소스를 관리할 수 있습니다. 감독자 서비스에는 서비스 구성을 관리할 수 있는 vCenter Server용 UI 플러그인이 있을 수도 있습니다.

작업	설명
감독자 서비스 업그레이드	먼저 vCenter Server에 새 서비스 버전을 추가한 다음 감독자에 새 버전을 설치하여 감독자에 설치된 서비스를 업그레이드할 수 있습니다. 서비스 업그레이드 중에 새 버전의 YAML 파일이 감독자에 적용됩니다. 새 버전에 필요하지 않은 이전 서비스 버전에 지정된 모든 리소스는 삭제됩니다. 예를 들어 버전 1이 포드 A를 지정하고 버전 2가 포드 B를 지정하는 경우 버전 2로 업그레이드한 후에는 새 포드 B가 생성되고 포드 A는 삭제됩니다. 현재 실행 중인 워크로드는 프로세스 중에 영향을 받지 않습니다.
감독자 서비스 버전 제거	감독자에서 서비스 버전을 제거하면 서비스 네임스페이스를 포함한 모든 서비스 리소스가 클러스터에서 제거됩니다. Kubernetes 워크로드에 있는 서비스의 애플리케이션 인스턴스는 계속 실행됩니다.
감독자 서비스 버전 삭제	서비스 버전을 삭제하려면 먼저 해당 버전을 비활성화하고 해당 버전이 실행되는 감독자에서 제거해야 합니다. 그런 다음 vCenter Server에서 서비스 버전을 삭제할 수 있습니다.
전체 감독자 서비스 삭제	전체 서비스를 삭제하려면 해당 버전을 모두 비활성화한 다음 감독자에서 이러한 버전을 제거하고 최종적으로 모든 서비스 버전을 삭제해야 합니다.

코어 감독자 서비스

코어 감독자 서비스는 감독자 활성화 중에 Operator가 vSphere IaaS control plane에 미리 설치된 서비스입니다. 먼저 감독자를 업데이트하지 않고도 감독자에 코어 감독자 서비스를 설치하고 해당 버전을 업그레이드할 수 있습니다. 그러나 코어 감독자 서비스의 Operator는 vSphere IaaS control plane에서 제거할 수 없습니다.

핵심 감독자 서비스의 예는 TKG 서비스 및 Velero vSphere Operator 서비스입니다.

다음으로 아래 항목을 읽으십시오.

- vCenter Server에 감독자 서비스 추가
- 감독자에 감독자 서비스 설치
- 감독자에서 감독자 서비스의 관리 인터페이스에 액세스
- 감독자 서비스에 새 버전 추가
- 감독자 서비스를 최신 버전으로 업그레이드
- 감독자에 설치된 감독자 서비스 보기
- 감독자 서비스 또는 버전 비활성화
- vCenter Server에서 감독자 서비스 버전 활성화
- 감독자에서 감독자 서비스 제거
- 감독자 서비스 버전 삭제
- 감독자 서비스 삭제

vCenter Server에 감독자 서비스 추가

vSphere IaaS control plane 환경이 실행되는 vCenter Server 시스템에 감독자 서비스를 추가하는 방법을 알아봅니다. vCenter Server에 서비스를 추가한 후 DevOps 엔지니어가 Kubernetes 워크로드에서 서비스를 사용할 수 있도록 감독자에 감독자 서비스를 설치합니다.

- 지원되는 감독자 서비스에 대해 자세한 내용 및 해당 서비스 YAML 파일을 다운로드하는 방법은 http://vmware.com/go/supervisor-service에서 참조하십시오.

사전 요구 사항

- 서비스를 추가하는 vCenter Server 시스템에 대한 **감독자 서비스 관리** 권한이 있는지 확인합니다.

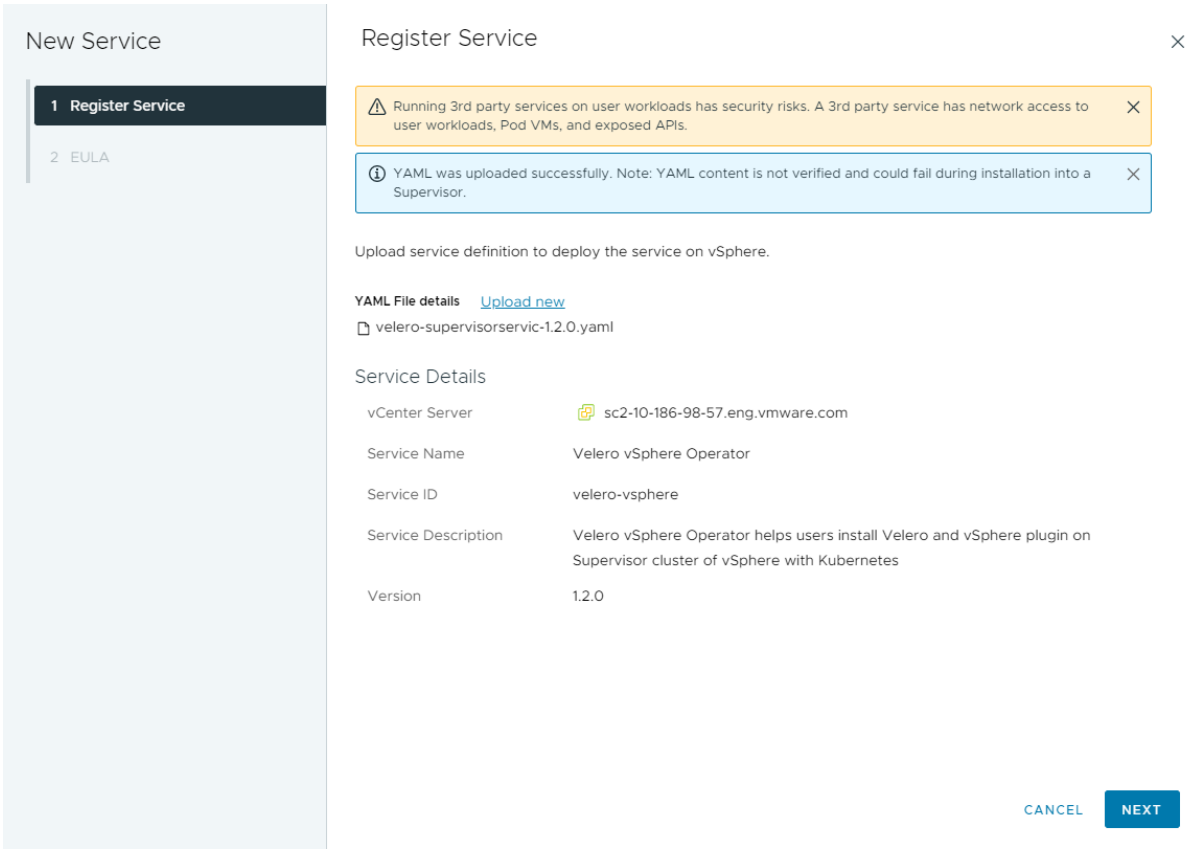
절차

1 vSphere Client 홈 메뉴에서 **워크로드 관리**를 선택합니다.

2 **서비스**를 선택합니다.

3 맨 위에 있는 드롭다운 메뉴에서 vCenter Server 시스템을 선택합니다.

4 **새 서비스 추가** 카드에서 서비스 YAML 파일을 끌어서 놓습니다.

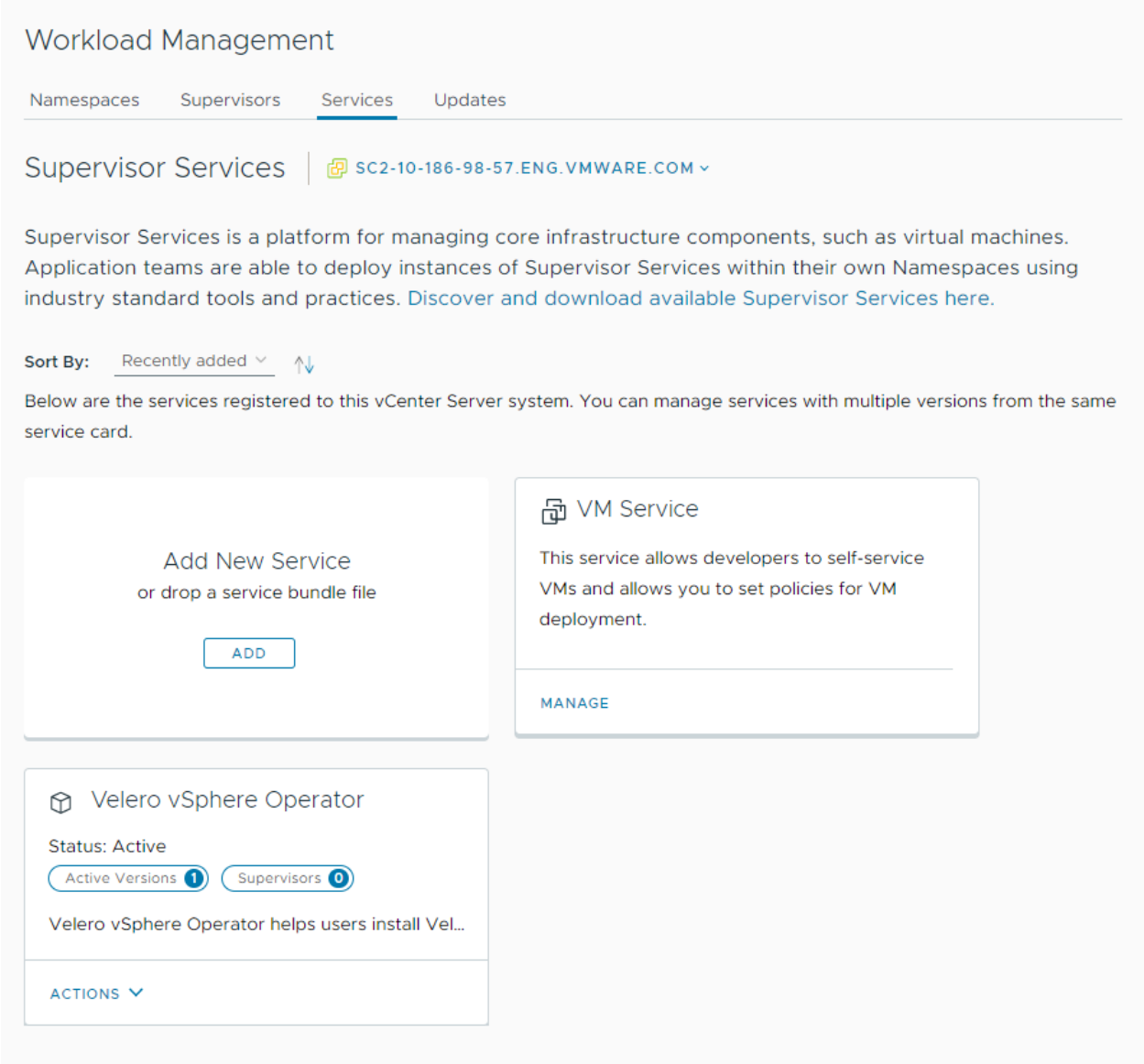


5 **다음**을 클릭하고 EULA가 있는 경우 동의합니다.

6 **마침**을 클릭합니다.

결과

감독자 서비스 및 모든 해당 정보가 vCenter Server 시스템에 등록됩니다. 서비스가 활성 상태입니다.



다음에 수행할 작업

DevOps 엔지니어가 Kubernetes 워크로드에서 사용할 수 있도록 감독자에 감독자 서비스를 설치합니다. 감독자에 감독자 서비스 설치의 내용을 참조하십시오.

감독자에 감독자 서비스 설치

감독자 서비스를 vCenter Server에 추가한 후 vSphere IaaS control plane 환경의 감독자에 설치할 수 있습니다. 최신 버전의 감독자 서비스를 설치하면 해당 감독자의 이전 서비스 버전이 재정의됩니다. 감독자에서 한 번에 하나의 감독자 서비스 버전만 실행할 수 있습니다.

- 지원되는 감독자 서비스에 대해 자세한 내용 및 해당 서비스 YAML 파일을 다운로드하는 방법은 http://vmware.com/go/supervisor-service에서 참조하십시오.

사전 요구 사항

- vCenter Server에 새 감독자 서비스 또는 최신 버전 및 기존 서비스를 추가합니다. vCenter Server에 감독자 서비스 추가 또는 감독자 서비스에 새 버전 추가의 내용을 참조하십시오.
- 서비스를 설치하려는 감독자에 대한 **감독자에 대한 감독자 서비스 관리** 권한이 있는지 확인합니다.
- 감독자 서비스에 영구 스토리지가 필요한 경우 vSAN 데이터 지속성 플랫폼을 구성합니다. 장 5 최신 상태 저장 서비스에 vSAN 데이터 지속성 플랫폼 사용의 내용을 참조하십시오.

절차

1. vSphere Client 홈 메뉴에서 **워크로드 관리**를 선택합니다.
2. **서비스**를 선택합니다.
3. 설치하려는 감독자 서비스의 카드에서 **작업 > 서비스 관리**를 선택합니다.
4. **버전 설치** 드롭다운 메뉴에서 원하는 감독자 서비스 버전을 선택합니다.

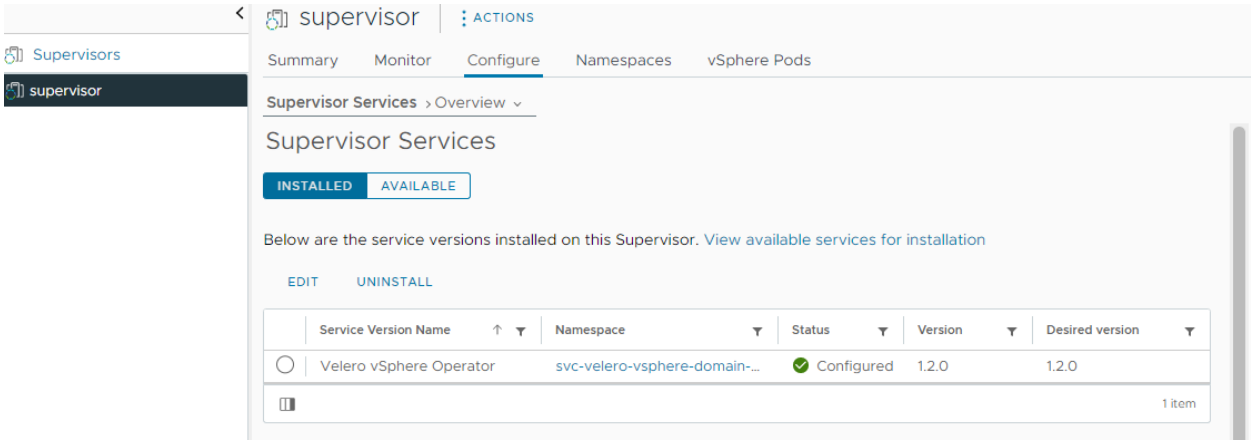
   참고 비활성화된 감독자 서비스 버전은 설치할 수 없습니다.
5. 서비스를 설치할 감독자를 선택합니다.
6. **다음**을 클릭합니다.

   호환성 사전 검사는 설치하려는 감독자 서비스 버전이 감독자와 호환되는지 확인하기 위해 수행됩니다. 서비스 버전이 감독자와 호환되는 경우 설치를 계속할 수 있습니다. 선택한 서비스 버전이 감독자와 호환되지 않는 경우 정확한 비호환성을 설명하는 두 가지 유형의 메시지가 표시됩니다.

   - 주의 메시지. 주의 메시지를 건너뛸 수 있지만 설치를 계속하려면 주의 메시지를 확인해야 합니다.
   - 오류 메시지. 오류 메시지는 감독자 서비스 버전이 감독자와 호환되지 않으며 설치할 수 없음을 나타냅니다. 오류 메시지가 표시되면 먼저 감지된 비호환성을 해결해야 특정 감독자에 서비스를 설치할 수 있습니다.
7. 서비스에 필요한 경우 **YAML 서비스 구성** 필드에 구성 속성을 입력합니다.
8. 감독자에서 서비스의 설치 진행률을 봅니다.
   a. **감독자** 탭을 선택하고 서비스를 설치할 감독자를 선택합니다.
   b. **구성**을 클릭하고 **감독자 서비스 > 개요**를 클릭합니다.
   c. **설치됨** 탭을 선택합니다.

결과

감독자 서비스는 [구성 중] 상태입니다. 즉, 필요한 모든 리소스가 감독자에서 생성되고 있으며 서비스 YAML이 클러스터에 적용되고 있음을 의미합니다. 모든 리소스와 네임스페이스가 생성되거나 업데이트된 상태에서 YAML이 감독자에 성공적으로 적용되면 서비스 상태가 [구성됨]으로 변경됩니다. 이 서비스는 해당 클러스터의 모든 네임스페이스에서 사용할 수 있으며 DevOps 엔지니어는 워크로드와 함께 사용할 수 있습니다.



다음에 수행할 작업

인터페이스를 사용하여 감독자 서비스를 구성합니다. 이것을 찾을 수 있는 위치는 감독자에서 감독자 서비스의 관리 인터페이스에 액세스를 참조하십시오.

감독자에서 감독자 서비스의 관리 인터페이스에 액세스

감독자에 감독자 서비스를 설치한 후 관리 UI를 찾을 수 있는 위치를 알아봅니다. 감독자 서비스는 vSphere Client의 감독자 보기에 서비스 인터페이스를 추가하는 vCenter Server용 자체 UI 플러그인을 제공할 수 있습니다. 감독자 서비스 특성에 따라 해당 인터페이스를 사용하여 서비스를 구성 및 관리하여 해당 서비스의 서비스 인스턴스를 배포할 수도 있습니다.

절차

1. vSphere Client 인벤토리에서 감독자로 변환한 호스트 클러스터로 이동합니다.
2. **구성** 탭을 클릭하고 서비스 인터페이스까지 아래로 스크롤합니다. 서비스 인터페이스는 일반적으로 서비스 이름을 따서 명명됩니다(예: **MinIO**).

감독자 서비스에 새 버전 추가

vSphere IaaS control plane 환경이 있는 vCenter Server에 감독자 서비스를 추가한 후 해당 서비스에 새 버전을 추가할 수 있습니다. 감독자에 다른 서비스 버전을 설치할 수 있습니다.

- 지원되는 감독자 서비스에 대해 자세한 내용 및 해당 서비스 YAML 파일을 다운로드하는 방법은 http://vmware.com/go/supervisor-service에서 참조하십시오.

사전 요구 사항

- vCenter Server에 서비스를 추가합니다. vCenter Server에 감독자 서비스 추가의 내용을 참조하십시오.
- 새 서비스 버전을 추가하는 vCenter Server 시스템에 대한 **감독자 서비스 관리** 권한이 있는지 확인합니다.

절차

1. vSphere Client 홈 메뉴에서 **워크로드 관리**를 선택합니다.
2. **서비스**를 선택합니다.
3. 새 버전을 추가하려는 서비스의 카드에서 **작업 > 새 버전 추가**를 선택합니다.
4. 새 서비스 버전의 YAML 파일을 업로드하고 **다음**을 클릭합니다.
5. EULA가 있는 경우 동의하고 **마침**을 클릭합니다.

결과

새 서비스 버전이 추가되고 활성 상태입니다.

다음에 수행할 작업

감독자에 새 서비스 버전을 설치합니다. 감독자에 감독자 서비스 설치의 내용을 참조하십시오.

감독자 서비스를 최신 버전으로 업그레이드

vCenter Server에 새 감독자 서비스 버전을 추가한 후에는 해당 버전을 감독자에 설치할 수 있습니다. 활성 상태이며 설치하려는 감독자와 호환되는 감독자 서비스 버전만 설치할 수 있습니다. 새 감독자 서비스 버전이 대상 감독자와 호환되는지 확인하기 위해 호환성 사전 검사가 수행됩니다.

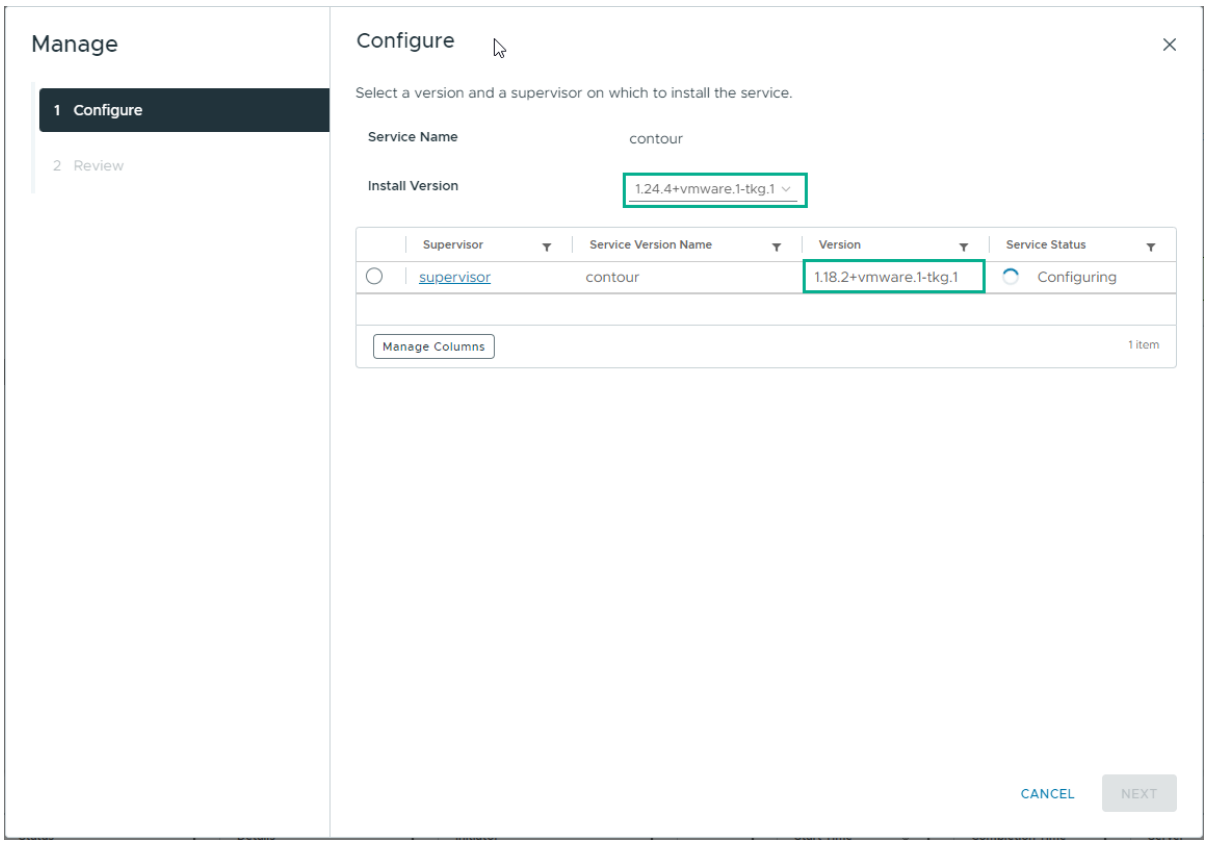
사전 요구 사항

- vCenter Server에 새 감독자 서비스 버전을 추가합니다 감독자 서비스에 새 버전 추가의 내용을 참조하십시오.
- 서비스를 설치하려는 감독자에 대한 **감독자에 대한 감독자 서비스 관리** 권한이 있는지 확인합니다.

절차

1. vSphere Client 홈 메뉴에서 **워크로드 관리**를 선택합니다.
2. **서비스**를 선택합니다.
3. **서비스 관리**를 선택합니다.
4. 설치할 새 버전을 선택하고 이것을 설치할 감독자를 선택합니다.

   감독자에 현재 설치된 서비스 버전이 이전 버전인지 확인합니다.



5 **다음**을 클릭합니다.

호환성 사전 검사는 설치하려는 감독자 서비스 버전이 감독자와 호환되는지 확인하기 위해 수행됩니다. 서비스 버전이 감독자와 호환되는 경우 설치를 계속할 수 있습니다. 선택한 서비스 버전이 감독자와 호환되지 않는 경우 정확한 비호환성을 설명하는 두 가지 유형의 메시지가 표시됩니다.

- 주의 메시지. 주의 메시지를 건너뛸 수 있지만 설치를 계속하려면 주의 메시지를 확인해야 합니다.
- 오류 메시지. 오류 메시지는 감독자 서비스 버전이 감독자와 호환되지 않으며 설치할 수 없음을 나타냅니다. 오류 메시지가 표시되면 먼저 감지된 비호환성을 해결해야 특정 감독자에 서비스를 설치할 수 있습니다.

6 서비스에 필요한 경우 **YAML 서비스 구성** 필드에 구성 속성을 입력합니다.

7 감독자에서 서비스의 설치 진행률을 봅니다.

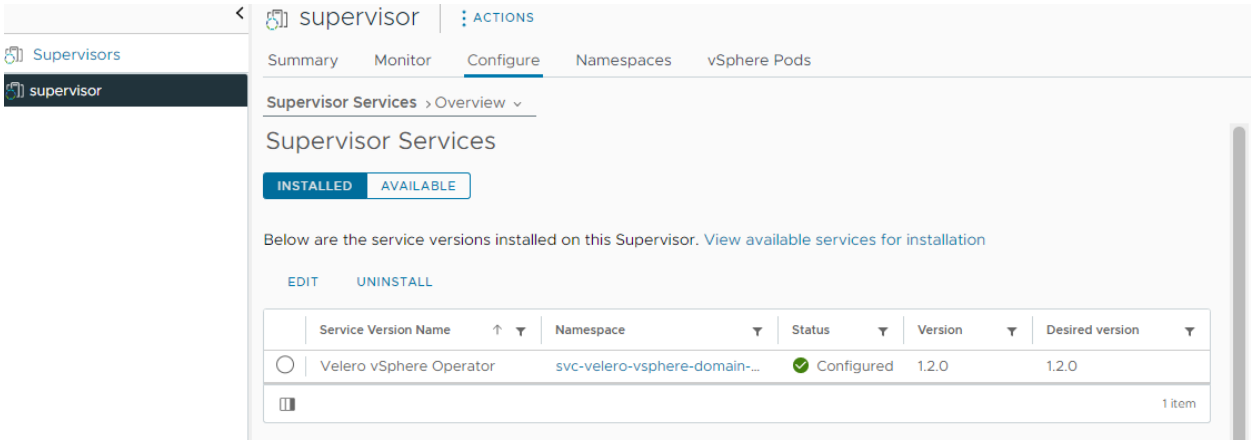
a **감독자** 탭을 선택하고 서비스를 설치할 감독자를 선택합니다.

b **구성**을 클릭하고 **감독자 서비스 > 개요**를 클릭합니다.

c **설치됨** 탭을 선택합니다.

결과

감독자 서비스는 [구성 중] 상태입니다. 즉, 필요한 모든 리소스가 감독자에서 생성되고 있으며 서비스 YAML이 클러스터에 적용되고 있음을 의미합니다. 모든 리소스와 네임스페이스가 생성되거나 업데이트된 상태에서 YAML이 감독자에 성공적으로 적용되면 서비스 상태가 [구성됨]으로 변경됩니다. 이 서비스는 해당 클러스터의 모든 네임스페이스에서 사용할 수 있으며 DevOps 엔지니어는 워크로드와 함께 사용할 수 있습니다.



감독자에 설치된 감독자 서비스 보기

vSphere IaaS control plane 환경의 감독자에 설치된 vSphere 서비스를 봅니다. 감독자에 설치된 감독자 서비스는 클러스터의 각 네임스페이스에서 사용할 수 있습니다.

사전 요구 사항

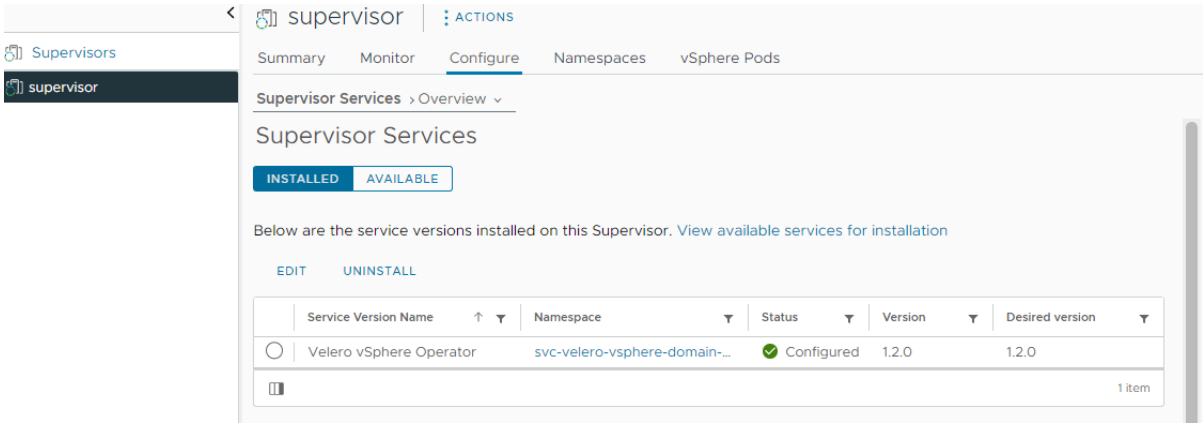
- vCenter Server에 감독자 서비스를 추가합니다. vCenter Server에 감독자 서비스 추가의 내용을 참조하십시오.
- 감독자에 감독자 서비스를 설치합니다. 감독자에 감독자 서비스 설치의 내용을 참조하십시오.

절차

1 vSphere Client 홈 메뉴에서 **워크로드 관리**를 선택합니다.

2 **감독자** 탭을 클릭하고 목록에서 감독자를 선택합니다.

3 **구성** 탭을 클릭하고 **감독자 서비스**에서 **개요**를 클릭합니다.



- **설치됨** 탭에서 현재 감독자에 설치된 감독자 서비스를 봅니다.
- **사용 가능** 탭에서 설치할 수 있는 감독자 서비스를 봅니다.

다음에 수행할 작업

해당 감독자에서 감독자 서비스를 관리하거나, 서비스를 제거하거나, **사용 가능** 탭의 서비스에서 새 서비스를 설치할 수 있습니다.

감독자 서비스 또는 버전 비활성화

vSphere IaaS control plane 환경의 Kubernetes 워크로드에서 감독자 서비스 버전을 더 이상 사용하지 않으려면 해당 버전을 비활성화합니다. 비활성화된 서비스 버전은 설치된 감독자에서 계속 실행되지만 다른 감독자에 비활성화된 서비스 버전을 설치할 수는 없습니다. 전체 서비스를 비활성화하면 모든 서비스 버전이 비활성화되고 서비스를 다시 활성화할 때까지는 감독자에 새 서비스 버전을 추가하거나 설치할 수 없습니다.

사전 요구 사항

- vCenter Server 수준에서 **감독자 서비스 관리** 권한이 있는지 확인합니다.

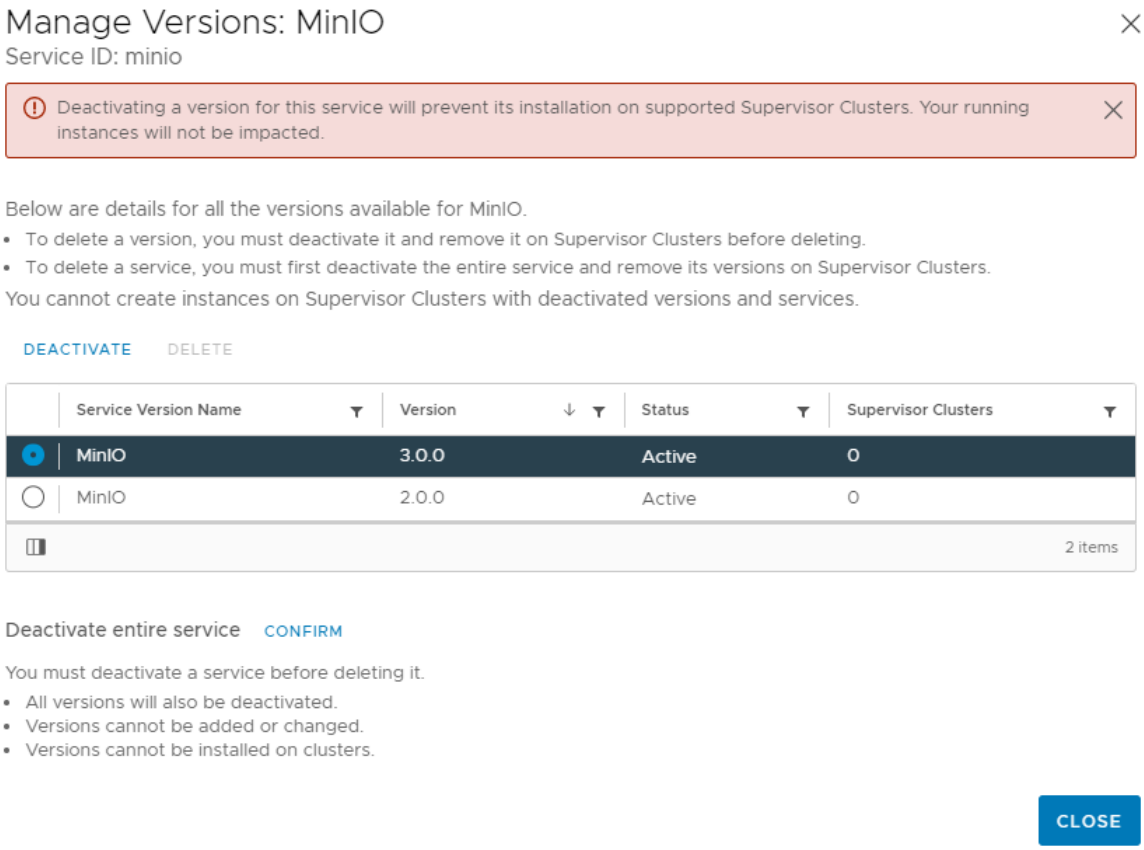
절차

1 vSphere Client 홈 메뉴에서 **워크로드 관리**를 선택합니다.

2 **서비스**를 선택합니다.

3 서비스 카드에서 **작업 > 버전 관리**를 선택합니다.

- 감독자 서비스 버전을 비활성화하려면 버전을 선택하고 **비활성화**를 클릭합니다.
- 전체 서비스를 비활성화하려면 **전체 서비스 비활성화** 옆에 있는 **확인**을 클릭합니다.



결과

서비스 버전이 비활성화되어 감독자에 설치할 수 없습니다.

vCenter Server에서 감독자 서비스 버전 활성화

감독자 서비스 버전이 비활성화되면 DevOps팀이 vSphere IaaS control plane에서 실행되는 Kubernetes 워크로드에서 해당 서비스 버전을 사용하려는 경우 다시 활성화할 수 있습니다.

- 서비스가 등록된 vCenter Server 시스템에 대한 **감독자 서비스 관리** 권한이 있는지 확인합니다.

절차

1. vSphere Client 홈 메뉴에서 **워크로드 관리**를 선택합니다.
2. **서비스**를 선택합니다.
3. 감독자 서비스 카드에서 **활성 버전**을 클릭합니다.
4. **버전 관리**를 선택합니다.
5. 비활성화됨 상태인 감독자 서비스 버전을 선택하고 **재활성화**를 클릭합니다.

감독자에서 감독자 서비스 제거

DevOps 팀이 vSphere IaaS control plane 환경에서 실행되는 Kubernetes 워크로드에 대해 감독자 서비스가 더 이상 필요하지 않으면 이 서비스를 감독자에서 제거합니다.

사전 요구 사항

- 서비스가 설치된 감독자를 호스팅하는 vCenter Server 시스템에 대한 **감독자 서비스 관리** 권한이 있는지 확인합니다.

절차

1. vSphere Client 홈 메뉴에서 **워크로드 관리**를 선택합니다.
2. **감독자** 탭을 클릭하고 목록에서 감독자를 선택합니다.
3. **구성** 탭을 클릭하고 **감독자 서비스**에서 **개요**를 클릭합니다.
4. **설치됨**에서 제거할 감독자 서비스를 선택하고 **제거**를 클릭합니다.

결과

감독자 서비스가 감독자에서 제거됩니다. 모든 서비스 리소스 및 서비스 네임스페이스가 감독자에서 제거됩니다. vSAN 데이터 지속성 플랫폼을 사용하는 서비스의 관리되는 모든 인스턴스가 감독자에서 제거됩니다.

감독자 서비스 버전 삭제

감독자 서비스 버전이 더 이상 사용되지 않고 DevOps 팀의 vSphere IaaS control plane 환경에서 실행되는 Kubernetes 워크로드에 더 이상 필요하지 않은 경우 vCenter Server에서 해당 버전을 삭제합니다.

사전 요구 사항

- 삭제하려는 감독자 서비스 버전이 감독자에 설치되어 있지 않은지 확인합니다. 감독자에서 감독자 서비스 제거의 내용을 참조하십시오.
- vCenter Server 수준에서 **감독자 서비스 관리** 권한이 있는지 확인합니다.

절차

1. vSphere Client 홈 메뉴에서 **워크로드 관리**를 선택합니다.
2. **서비스**를 선택합니다.
3. 감독자 서비스 카드에서 **작업 > 버전 관리**를 선택합니다.
4. 삭제할 버전을 선택하고 **비활성화**를 클릭합니다.
5. 비활성화된 버전을 선택하고 **삭제**를 클릭합니다.

감독자 서비스 삭제

감독자 서비스가 DevOps 엔지니어의 Kubernetes 워크로드에 더 이상 필요하지 않은 경우 vSphere IaaS control plane 환경에서 삭제합니다.

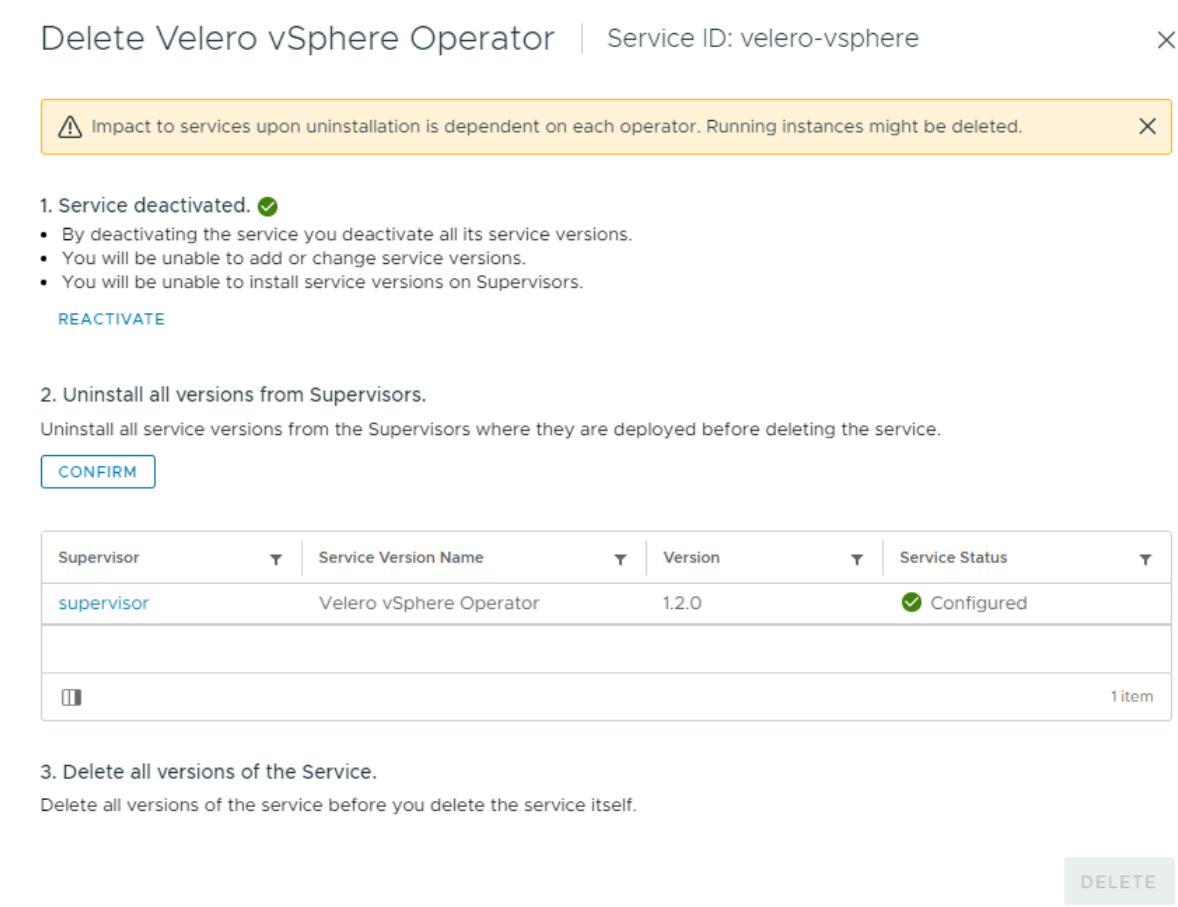
사전 요구 사항

- 서비스가 등록된 vCenter Server 시스템에 대한 **감독자 서비스 관리** 권한이 있는지 확인합니다.

절차

1. vSphere Client 홈 메뉴에서 **워크로드 관리**를 선택합니다.
2. **서비스**를 선택합니다.
3. 제거하려는 감독자 서비스 카드에서 **작업 > 삭제**를 선택합니다.
4. 현재 사용 가능한 모든 서비스 버전 비활성화를 확인합니다.
5. 감독자에서 서비스를 제거한다고 확인합니다.

   실행되는 감독자에서 감독자 서비스를 제거하는 데 다소 시간이 걸릴 수 있습니다. 프로세스가 완료되는 동안 대화상자를 닫았다가 다시 열어서 다음 단계를 계속 진행할 수 있습니다.



6. 사용 가능한 모든 서비스 버전 삭제를 확인합니다.

7 **삭제**를 클릭합니다.

최신 상태 저장 서비스에 vSAN 데이터 지속성 플랫폼 사용

5

vSphere IaaS control plane에서는 영구 스토리지가 필요한 최신 상태 저장 서비스에 vSAN 데이터 지속성 플랫폼을 사용할 수 있습니다. 이 플랫폼은 타사가 서비스 애플리케이션을 기본 vSphere 인프라와 통합할 수 있는 프레임워크를 제공합니다.

vSAN 데이터 지속성 플랫폼 정보

vSAN 데이터 지속성 사용의 이점은 다음과 같습니다.

자동 서비스 배포 및 확장/축소

vSphere Client를 사용하면 관리자가 감독자에 최신 상태 저장 서비스를 설치 및 배포하고 DevOps 엔지니어에게 서비스 네임스페이스에 대한 액세스 권한을 부여할 수 있습니다. DevOps 엔지니어는 Kubernetes API를 통해 셀프 서비스 방식으로 상태 저장 서비스의 인스턴스를 동적으로 프로비저닝하고 확장/축소할 수 있습니다.

vCenter Server와 통합된 서비스 모니터링

파트너는 vCenter Server와 통합되는 대시보드 플러그인을 구축할 수 있습니다. vSphere 관리자는 UI 플러그인을 사용하여 상태 저장 서비스를 관리하고 모니터링할 수 있습니다. 또한 vSAN은 이러한 통합 타사 서비스에 대한 상태 및 용량 모니터링 기능을 제공합니다.

vSAN Direct로 최적화된 스토리지 구성

vSAN Direct를 통해 최신 상태 저장 서비스는 최적화된 I/O 및 스토리지 효율성을 위해 직접 연결된 기본 스토리지와 직접 상호 작용할 수 있습니다.

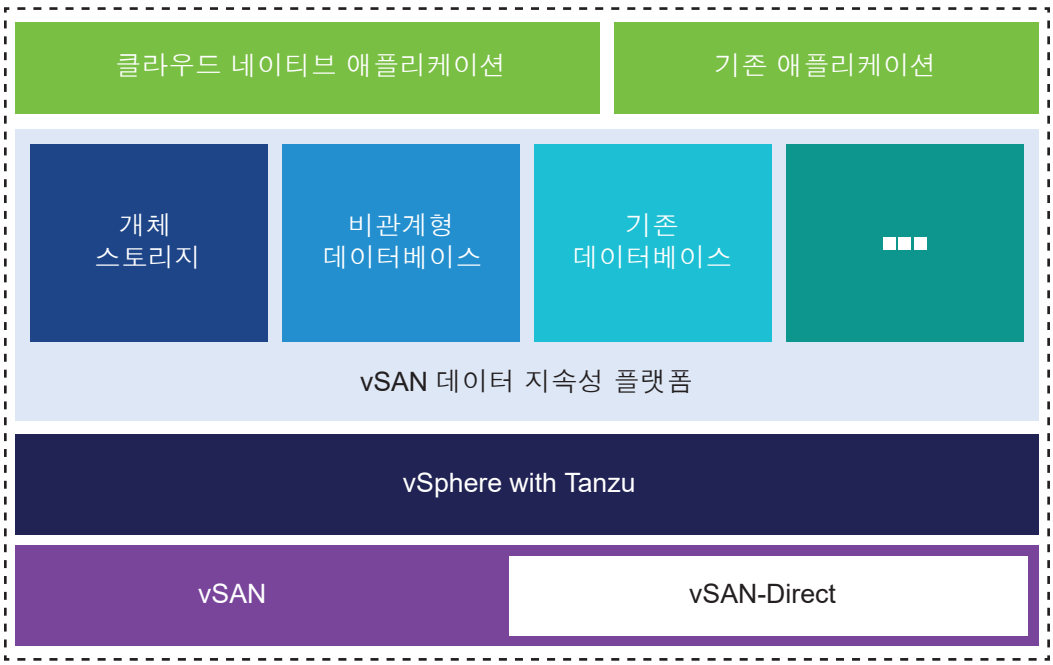
플랫폼은 다음과 같은 유형의 서비스를 지원합니다.

- 개체 스토리지(예: MinIO).
- 비관계형 데이터베이스라고도 하는 NoSQL 데이터베이스.
- 기존 데이터베이스.

vSphere 비공유 스토리지

대부분의 최신 상태 저장 서비스에는 SNA(비공유 아키텍처)가 있습니다. 이러한 서비스는 복제되지 않은 로컬 스토리지를 사용하고 자체 스토리지 복제, 압축 및 기타 데이터 작업을 제공합니다. 따라서 기본 스토리지에서 동일한 작업을 수행하는 경우 서비스 이점이 없습니다.

작업 중복을 방지하기 위해 vSAN 데이터 지속성 플랫폼은 최적화된 데이터 경로가 있는 두 개의 vSAN 솔루션을 제공합니다. 영구 서비스는 SNA 스토리지 정책을 사용하는 vSAN 또는 vSAN Direct라는 원시 로컬 스토리지에서 실행할 수 있습니다.



SNA 스토리지 정책이 있는 vSAN

이 기술을 사용하면 vSAN 호스트-로컬 SNA 정책과 함께 분산 복제된 vSAN 데이터스토어를 사용할 수 있습니다. 그 결과, SNA 서비스 애플리케이션에서 배치를 제어하고 데이터 가용성을 유지 보수하는 작업을 담당할 수 있습니다. 이 기술을 통해 영구 서비스는 계산 인스턴스와 스토리지 개체를 동일한 물리적 ESXi 호스트에서 쉽게 배치할 수 있습니다. 호스트-로컬 배치를 사용하면 스토리지 계층이 아닌 서비스 계층에서 복제와 같은 작업을 수행할 수 있습니다.

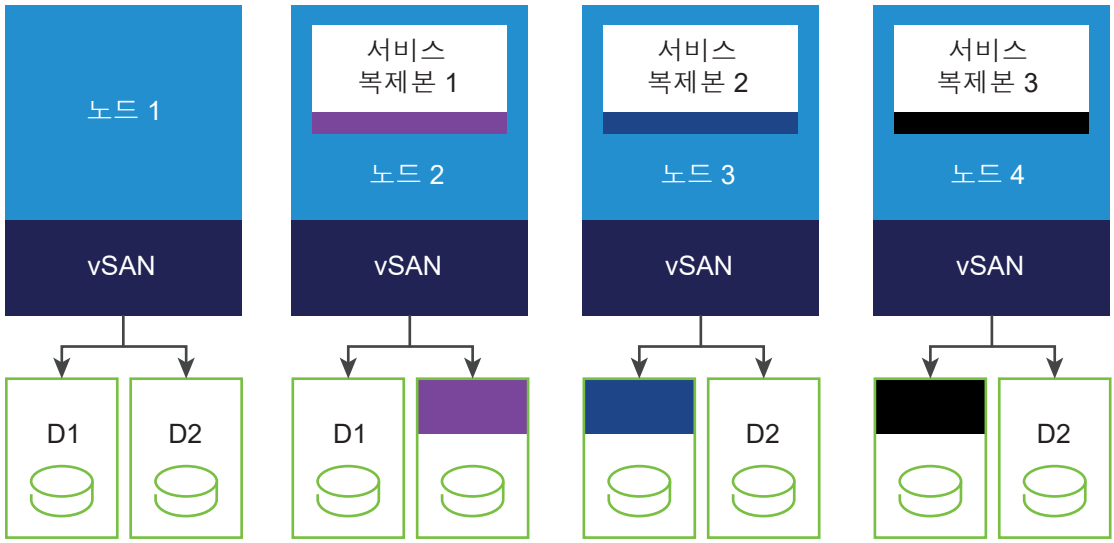
포드 같은 계산 인스턴스는 vSAN 클러스터의 노드 중 하나에 먼저 나타납니다. 그런 다음 vSAN SNA 정책으로 생성된 vSAN 개체는 포드를 실행하는 동일한 노드에 모든 데이터가 자동으로 배치됩니다.

다음 예에서는 영구 볼륨에 대해 SNA 스토리지 클래스를 사용하는 애플리케이션의 스토리지 배포를 보여줍니다. vSAN은 영구 볼륨 배치를 위해 노드에서 원하는 디스크 그룹을 선택할 수 있습니다.

총 데이터 복사본 수 = 3

예상 Fault Tolerance = 2

허용이 보장되는 실제 장애 수 = 2

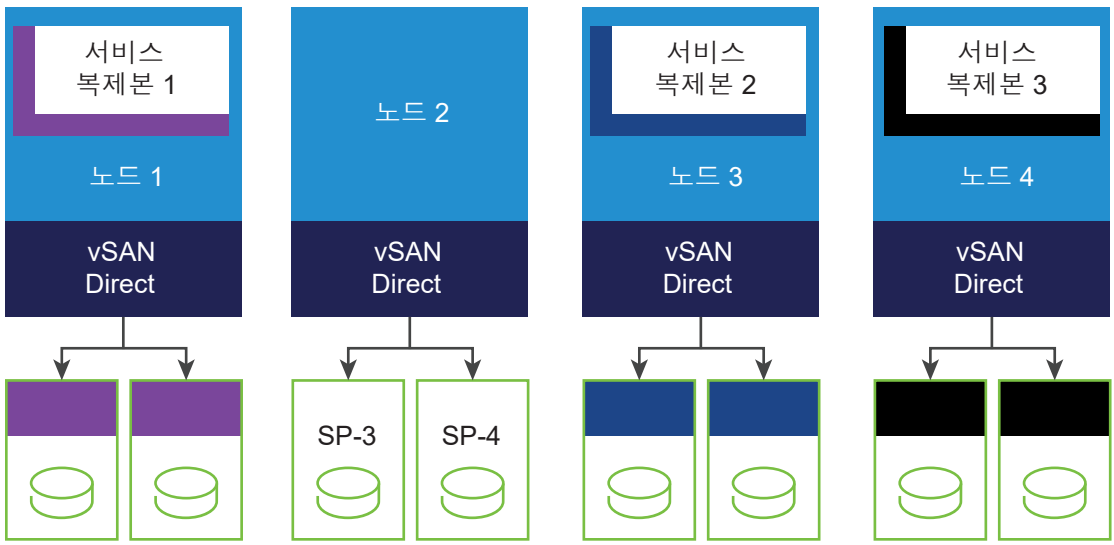


vSAN Direct

SNA 스토리지 정책이 있는 vSAN은 계산 인스턴스에 로컬로 데이터를 배치할 수 있지만 애플리케이션과 물리적 스토리지 디바이스 간의 분산된 vSAN 데이터 경로에 오버헤드가 존재합니다. vSAN Direct를 사용하면 상태 저장 서비스 애플리케이션이 보다 직접적인 데이터 경로를 통해 대부분의 원시 비 vSAN 로컬 스토리지에 액세스할 수 있어서 성능이 가장 최적화된 솔루션을 제공합니다.

vSAN Direct를 사용하여 vSphere 관리자는 호스트-로컬 디바이스를 할당한 다음, 디바이스를 관리하고 모니터링할 수 있습니다. vSAN Direct는 디바이스 상태, 성능 및 용량에 대한 인사이트를 제공합니다. 할당하는 로컬 디바이스마다 vSAN Direct는 독립 VMFS 데이터스토어를 생성하여 애플리케이션에 대한 배치 선택 항목으로 사용할 수 있도록 합니다. vSAN Direct가 관리하는 VMFS 데이터스토어는 Kubernetes에서 스토리지 풀로 노출됩니다. vSphere Client에서는 vSAN Direct 데이터스토어로 나타납니다.

다음은 vSAN Direct 디스크에 로컬로 배치된 영구 볼륨을 보여줍니다.



SNA 또는 vSAN Direct에서 vSAN을 사용하는 경우

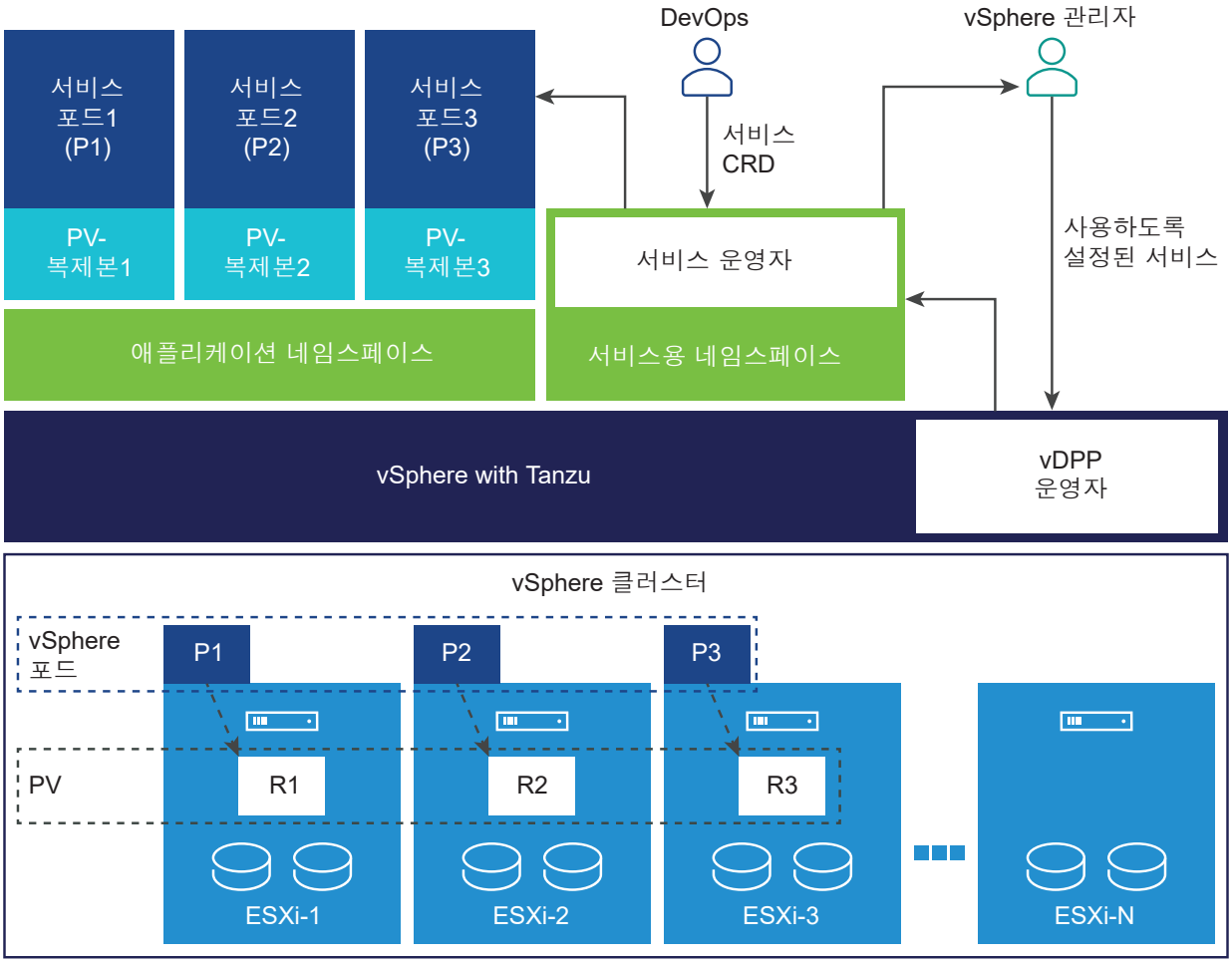
사용할 vSAN 유형을 결정할 때는 다음과 같은 일반적인 권장 사항을 따르십시오.

- 클라우드 네이티브 상태 저장 애플리케이션이 다른 일반 VM 또는 Kubernetes 워크로드와 물리적 인프라를 공유하도록 하려면 SNA가 포함된 vSAN을 사용합니다. 각 워크로드는 자체 스토리지 정책을 정의할 수 있으며, 단일 클러스터에서 두 환경 모두의 장점을 활용할 수 있습니다.
- 비공유 클라우드 네이티브 서비스를 위한 전용 하드웨어 클러스터를 생성하는 경우에는 vSAN Direct를 사용합니다.

vSAN 데이터 지속성 플랫폼 운영자

vDPP(vSAN 데이터 지속성 플랫폼) 운영자는 vSphere와 통합된 파트너 상태 저장 서비스를 실행하고 관리하는 작업을 담당하는 구성 요소입니다. vDPP 운영자는 사용 가능한 상태 저장 서비스를 vSphere 관리자에게 노출합니다. vSphere 관리자가 영구 서비스(예: MinIO)를 사용하도록 설정하면 vDPP 운영자는 감독자의 서비스에 대한 애플리케이션별 운영자를 배포합니다.

애플리케이션별 운영자는 타사에서 제공하며 vDPP 규격이어야 합니다. 이 운영자는 일반적으로 Kubernetes 사용자가 인스턴스를 인스턴스화하기 위한 셀프 서비스 인터페이스를 제공하는 CRD를 제공합니다. vSphere IaaS control plane는 이 운영자와 CRD를 사용하여 새 서비스 인스턴스를 프로비저닝하고 상태 저장 서비스 계층을 통해 관리하고 모니터링합니다. 이러한 운영자의 대부분은 인스턴스 배포에 상태 저장 집합을 사용합니다.



vSphere 관리자가 서비스를 사용하도록 설정하면 다음 작업이 수행됩니다.

- vDPP 운영자는 서비스별 운영자를 활성화합니다.
- 서비스별 운영자는 UI 플러그인을 등록합니다.
- 스토리지 최적화 스토리지 정책이 생성됩니다.

vSAN 데이터 지속성 플랫폼에 대한 구성 제한

VMware는 VMware 구성 최대값 도구에 구성 제한을 제공합니다.

vSAN 데이터 지속성 최대값	제한
vSAN 데이터 지속성 플랫폼당 최대 영구 볼륨 수	1000
vSAN 데이터 지속성 플랫폼의 서비스 인스턴스당 최대 영구 볼륨 수	60 ~ 80

다음으로 아래 항목을 읽으십시오.

- vSphere IaaS control plane에서 상태 저장 서비스 사용
- 상태 저장 서비스에 대한 vSAN Direct 데이터스토어 설정

- vSphere IaaS control plane에서 상태 저장 서비스 모니터링
- 상태 저장 서비스에 사용할 수 있는 스토리지 정책 확인
- vSAN 데이터 지속성 플랫폼에 대한 사용자 지정 스토리지 정책 생성

vSphere IaaS control plane에서 상태 저장 서비스 사용

vSphere IaaS control plane는 해당 영구 스토리지 요구 사항에 맞게 vSAN 데이터 지속성 플랫폼을 사용하는 여러 타사 서비스와 통합됩니다. vSphere 관리자가 vCenter Server에서 서비스를 사용하도록 설정합니다. 상태 저장 서비스를 사용하도록 설정하는 경우 먼저 서비스를 설명하는 다운로드한 YAML 파일을 사용하여 vCenter Server에 서비스를 등록합니다. 그런 다음 DevOps 엔지니어가 Kubernetes 워크로드에서 서비스를 사용할 수 있도록 감독자에 서비스를 설치합니다.

사전 요구 사항

필요한 권한: **감독자 서비스.감독자 서비스 관리**

1 영구 스토리지 구성

vSAN 데이터 지속성 플랫폼을 통해 상태 저장 서비스는 다음 두 가지 모드에서 vSAN 스토리지를 사용할 수 있습니다.

- vSAN Direct와 같은 경보가 표시됩니다. vSAN Direct를 설정하려면 vSAN Direct 데이터스토어 생성의 내용을 참조하십시오.

  참고 vSAN Direct 데이터스토어에 있는 디스크의 경우 볼륨 할당 유형 변경이 지원되지 않습니다. vSAN Direct 데이터스토어의 디스크에 대한 볼륨 할당 유형을 선택한 후에는 변경할 수 없습니다. 단, 복제 및 재배치와 같은 작업 중에는 새 디스크에 대한 볼륨 할당 유형 변경이 허용됩니다.

- SNA 스토리지 정책을 사용하는 일반 vSAN. vSAN 스토리지 설정에 대한 자세한 내용은 "VMware vSAN 관리" 항목을 참조하십시오.

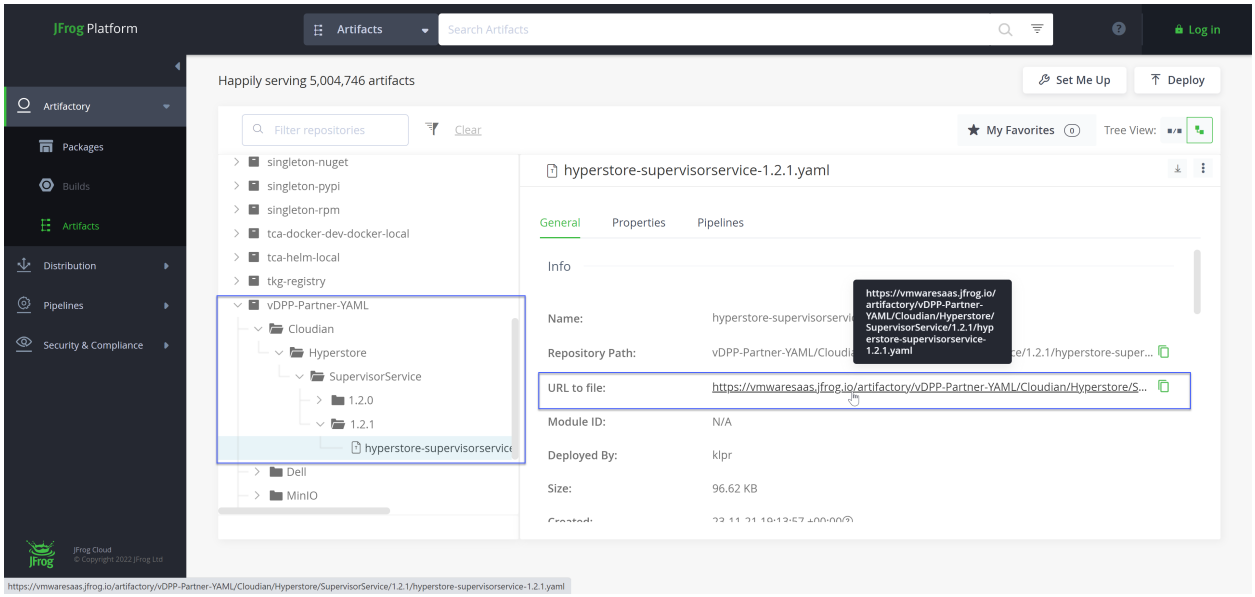
2 서비스 YAML 파일 다운로드

VMware에서 유지 보수하는 저장소에서 서비스 YAML 파일을 다운로드할 때는 사용 중인 vSphere 버전과 호환되는 올바른 서비스 버전을 사용해야 합니다.

이전 버전의 파트너 서비스인 MinIO 및 Cloudian Hyperstore를 설치한 경우 vSphere 환경을 업그레이드한 후 파트너 서비스를 호환 가능한 버전으로 업그레이드합니다. 최신 버전의 파트너 운영자는 특정 문제를 해결하고 새 플랫폼 기능을 사용합니다. 자세한 내용은 파트너 설명서를 참조하십시오.

1 https://vmwaresaas.jfrog.io/ 저장소에서 **아티팩트 > vDPP-파트너-YAML**의 적절한 파트너 폴더로 이동합니다.

2 파일에 대한 URL을 클릭하고 YAML 파일을 다운로드합니다.



3 vCenter Server에 서비스 추가

다운로드한 파트너 서비스 YAML 파일을 사용합니다.

vCenter Server에 감독자 서비스 추가의 내용을 참조하십시오.

4 감독자에 서비스 설치

감독자에 감독자 서비스 설치의 내용을 참조하십시오.

서비스를 사용하도록 설정하면 vSAN 데이터 지속성 플랫폼에서 다음 작업이 수행되어 서비스에 필요한 리소스가 생성됩니다.

- 감독자에서 이 서비스에 대한 네임스페이스를 생성합니다.
- 기본 스토리지 정책 및 해당 스토리지 클래스를 생성하고 이를 네임스페이스에 할당합니다.

  이 정책은 vSAN SNA(비공유 아키텍처) 및 vSAN Direct 데이터스토어를 위한 것입니다.

  참고 vSAN 데이터 지속성 플랫폼은 vSphere 관리자가 서비스를 사용하도록 설정한 후 네임스페이스에 vsan-direct 및 vsan-sna 스토리지 클래스를 자동으로 생성합니다. 감독자에서 실행되는 애플리케이션만 vsan-direct 및 vsan-sna 스토리지 클래스를 사용할 수 있습니다. 이러한 스토리지 클래스는 Tanzu Kubernetes Grid 클러스터 내에서 사용할 수 없습니다.

  vSphere 7.0 업데이트 2 이상에서는 vSAN Direct 스토리지 정책이 기능 기반입니다. vSphere 7.0 업데이트 1에서 태그 기반 정책을 생성했으면 vSphere 7.0 업데이트 2 이상으로 업그레이드한 후에 기능 기반으로 자동 변환됩니다.

  사용자 지정 스토리지 정책을 생성하여 기본값 대신 서비스 네임스페이스에 할당하려면 vSAN Direct 스토리지 정책 생성 및 vSAN SNA 스토리지 정책 생성 항목을 참조하십시오.
- 편집 및 보기 권한이 있는 역할을 포함하여 DevOps 역할을 생성합니다.

서비스 연산자가 배포되면 해당 사용자 지정 CRD가 감독자에 설치됩니다. 편집 권한이 있는 사용자는 네임스페이스에서 이러한 CRD의 리소스에 CRUD를 수행할 수 있습니다. 보기 권한이 있는 사용자는 이 CRD의 리소스를 볼 수만 있습니다.

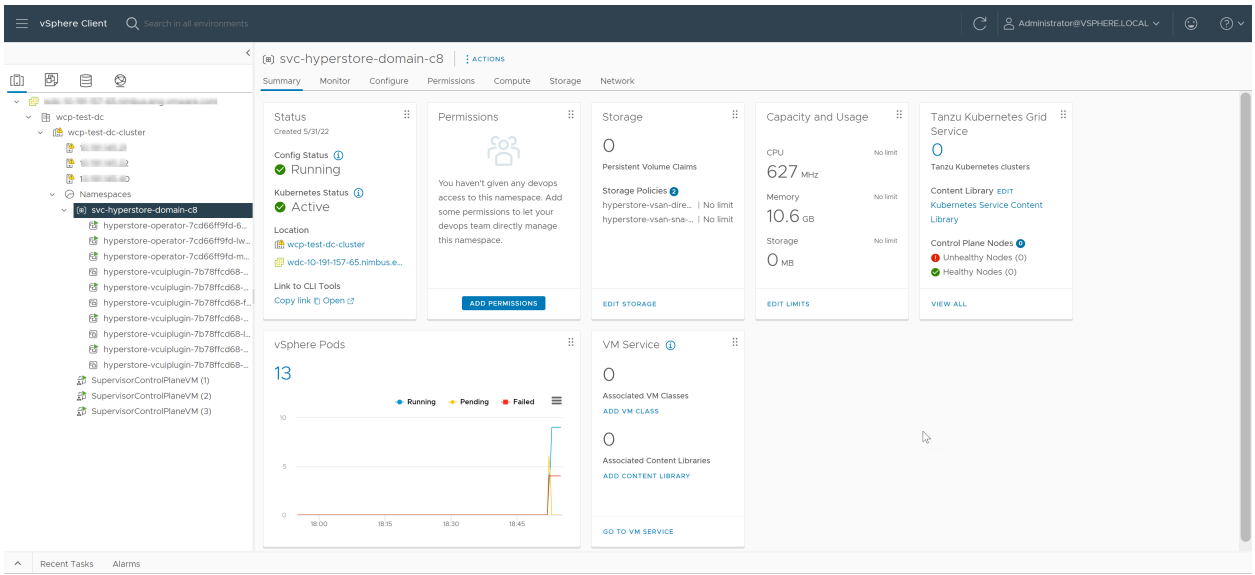
- 타사에서 사용자 지정 UI 플러그인을 제공한 경우에는 vSphere Client에 나타납니다. vSphere 관리자는 플러그인을 사용하여 서비스를 관리할 수 있습니다.

5 서비스에 대해 생성된 리소스 확인

vSphere 관리자는 서비스에 적합한 리소스가 모두 생성되었는지 확인할 수 있습니다.

서비스에 대해 생성된 네임스페이스로 이동하고 **요약** 탭을 클릭합니다.

[요약] 페이지에는 네임스페이스에 할당된 스토리지 정책, 네임스페이스에서 실행되는 vSphere 포드 등이 표시됩니다.



6 서비스 관리 및 모니터링

- 타사에서 사용자 지정 UI 플러그인을 제공한 경우 vSphere 관리자는 플러그인을 사용하여 서비스를 관리하고 모니터링할 수 있습니다.

  자세한 내용은 타사 UI 플러그인 설명서를 참조하십시오.

- 또한 vSphere 관리자는 Skyline Health 점검을 사용하여 서비스를 모니터링할 수 있습니다. vSphere IaaS control plane에서 상태 저장 서비스 모니터링의 내용을 참조하십시오.
- 기본 정책 대신 사용자 지정 스토리지 정책을 생성하려면 vSAN 데이터 지속성 플랫폼에 대한 사용자 지정 스토리지 정책 생성의 내용을 참조하십시오.

7 서비스 사용 시작

DevOps 엔지니어는 `kubectl` 명령을 사용하여 서비스 네임스페이스에 액세스합니다.

상태 저장 서비스에 사용하는 네임스페이스에 적절한 스토리지 클래스가 있는지 확인하려면 상태 저장 서비스에 사용할 수 있는 스토리지 정책 확인 항목을 참조하십시오.

타사 CRD를 사용하여 타사 애플리케이션 서비스의 인스턴스를 배포할 수 있습니다. 자세한 내용은 타사 설명서를 참조하십시오.

상태 저장 서비스에 대한 vSAN Direct 데이터스토어 설정

vSphere IaaS control plane에서 상태 저장 서비스에 대한 전용 하드웨어 클러스터를 생성하려는 경우 vSAN Direct 데이터스토어를 사용할 수 있습니다. vSAN Direct는 ESXi 호스트에 로컬인 할당되지 않은 스토리지 디바이스에 배포하는 원시 데이터스토어입니다.

vSAN Direct에 대한 스토리지 디바이스 태그 지정

vSAN Direct는 vSAN 클러스터 내의 각 ESXi 호스트에 할당되지 않은 디스크가 필요합니다. 하지만 특정 환경에서 vSAN은 호스트의 모든 로컬 스토리지 디바이스를 자동으로 할당합니다. 디바이스를 일반 vSAN에는 부적합하고 vSAN Direct에는 사용할 수 있도록 설정할 수 있습니다.

`esxcli` 명령을 사용하여 디바이스를 vSAN Direct로 표시합니다.

절차

1 vSAN Direct에 대한 로컬 스토리지 디바이스에 태그를 지정합니다.

```
esxcli vsan storage tag add -d diskName -t vsanDirect
```

예를 들면 다음과 같습니다.

```
esxcli vsan storage tag add -d mpx.vmhba0:C0:T1:L0 -t vsanDirect
```

디바이스가 일반 vSAN에 부적합하게 됩니다.

2 디바이스에서 vSAN Direct 태그를 제거합니다.

```
esxcli vsan storage tag remove -d diskName -t vsanDirect
```

예를 들면 다음과 같습니다.

```
esxcli vsan storage tag remove -d mpx.vmhba0:C0:T1:L0 -t vsanDirect
```

스크립트를 사용하여 vSAN Direct용 스토리지 디바이스에 태그 지정

또는 다음 스크립트를 사용하여 ESXi 호스트에 연결된 HDD 디바이스에 태그를 지정할 수 있습니다. 스크립트를 실행한 후 디바이스는 일반 vSAN에는 부적합하게 되고 vSAN Direct에 사용할 수 있게 됩니다.

```
#!/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 데이터스토어 생성

vSphere 관리자는 vSAN 데이터 지속성 플랫폼 또는 VM 인스턴스 스토리지와 같은 기능과 함께 사용할 vSAN Direct 데이터스토어를 설정합니다. 데이터스토어를 생성하려면 ESXi 호스트에 로컬인 할당되지 않은 스토리지 디바이스를 사용합니다.

감독자에 대해 vSAN을 사용하도록 설정할 때 vSAN Direct 데이터스토어를 생성할 수 있습니다. 다음 작업은 클러스터에서 vSAN이 이미 사용되도록 설정된 경우 로컬 스토리지 디바이스를 vSAN Direct로 할당하는 방법을 보여 줍니다.

절차

1 vSphere Client에서 vSAN 클러스터로 이동합니다.

2 **구성** 탭을 클릭합니다.

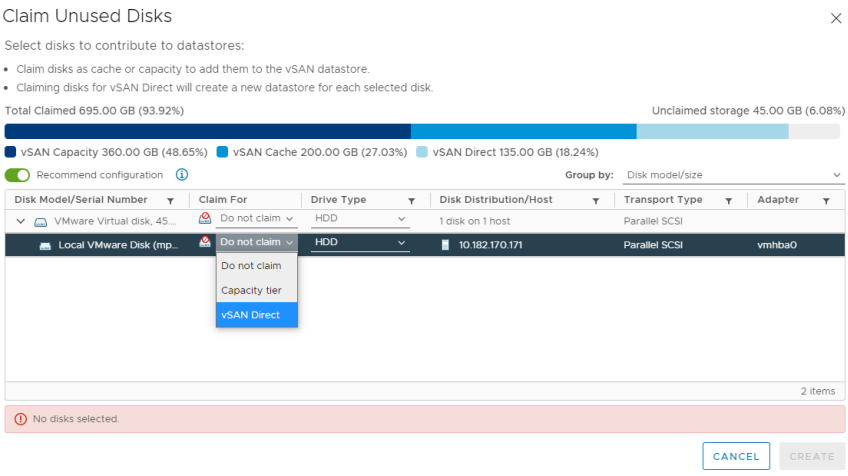
3 vSAN에서 **디스크 관리**를 클릭합니다.

4 **사용되지 않는 디스크 할당**을 클릭합니다.

5 **사용되지 않는 디스크 할당** 대화 상자에서 **vSAN Direct**를 클릭합니다.

6 할당할 디바이스를 선택하고 **vSAN Direct에 대해 할당** 열의 확인란을 선택합니다.

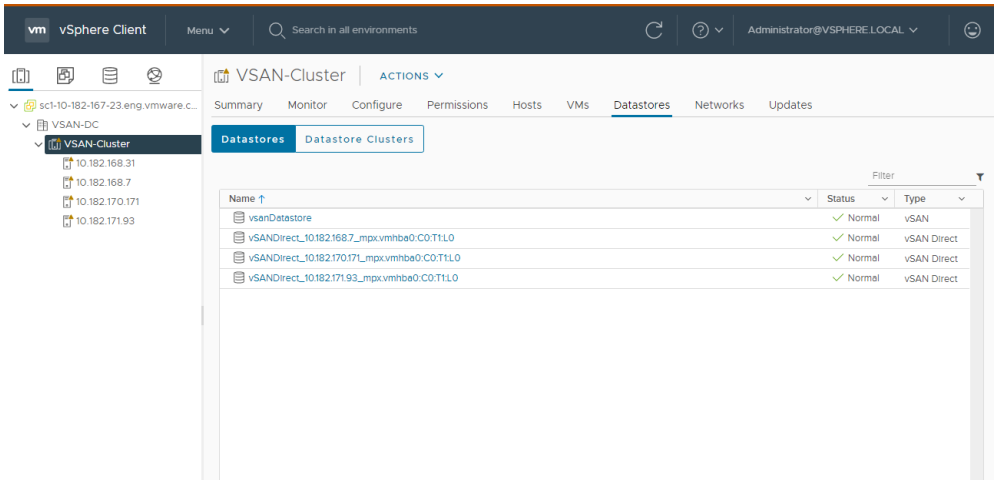
참고 일반 vSAN 데이터스토어에 대한 디바이스를 할당한 경우 해당 디바이스는 **vSAN Direct** 탭에 나타나지 않습니다.



7 **생성**을 클릭합니다.

할당하는 각 디바이스에 대해 vSAN Direct는 새 데이터스토어를 생성합니다.

8 **데이터스토어** 탭을 클릭하여 클러스터의 모든 vSAN Direct 데이터스토어를 표시합니다.



다음에 수행할 작업

vSAN Direct를 외부 스토리지와 함께 사용할 수 있습니다. 자세한 내용은 "vSphere IaaS 제어부 유지 보수" 설명서에서 vSAN Direct에서 외부 스토리지 사용을 참조하십시오.

vSphere IaaS control plane에서 상태 저장 서비스 모니터링

통합된 타사 상태 저장 서비스를 사용하도록 설정한 후에 vSAN 상태 및 용량 모니터링 기능을 사용하여 상태를 살펴보고 서비스 개체의 공간 사용량을 분석합니다.

절차

1 vSphere Client에서 감독자로 이동합니다.

2 **모니터** 탭을 클릭합니다.

3 사용하도록 설정된 서비스에 해당하는 네임스페이스에서 실행되는 가상 개체를 모니터링합니다.

   a **vSAN**에서 **가상 개체**를 클릭합니다.

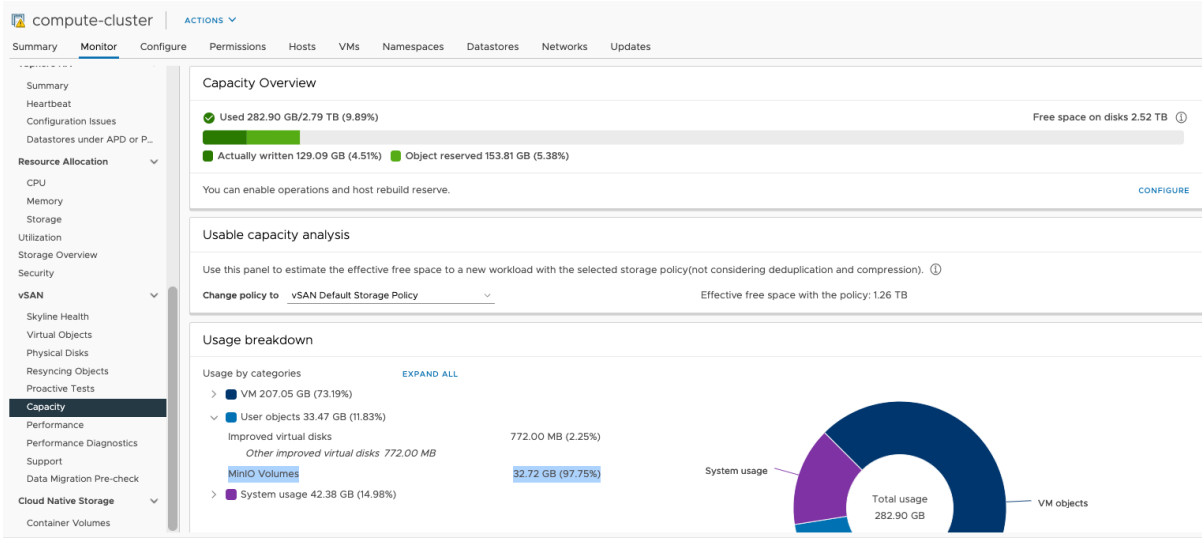
      가상 개체(예: MinIO 연산자 개체)를 찾아보고 해당 상태를 확인할 수 있습니다.

   b 물리적 인프라 전체에서 개체의 배치를 보려면 특정 개체를 선택하고 **배치 세부 정보 보기**를 클릭합니다.

4 서비스 개체가 사용하는 용량을 모니터링합니다.

   a **vSAN**에서 **용량**을 클릭합니다.

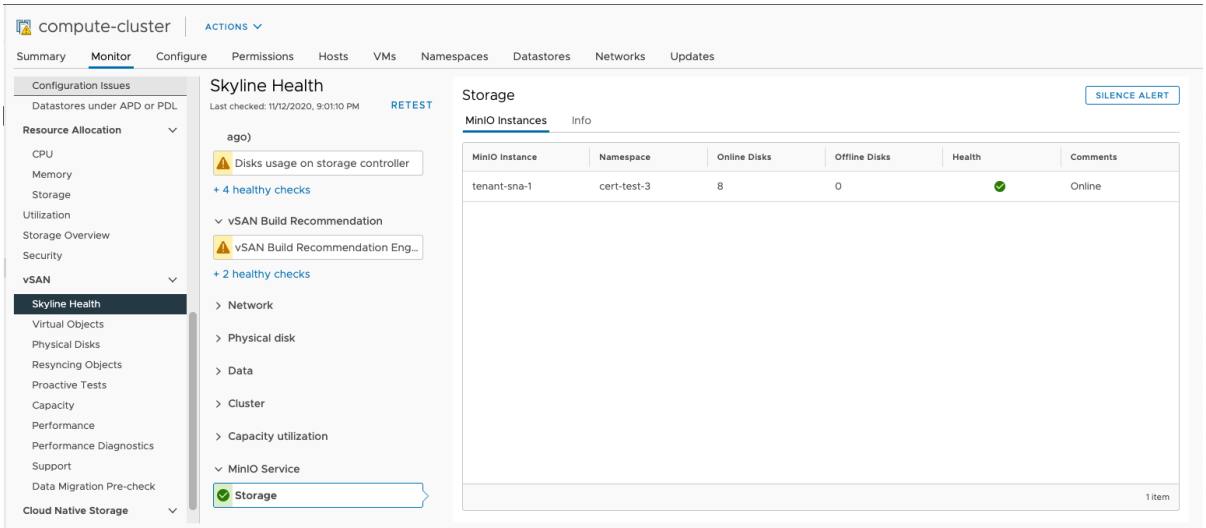
   b **사용량 분석** 창에서 **사용자 개체** 아래에 서비스 개체를 표시합니다.



5 서비스 인스턴스의 상태를 모니터링합니다.

a **vSAN**에서 **Skyline Health**를 선택합니다.

b 세부 정보를 보려면 개별 서비스 상태 점검을 선택합니다.



상태 저장 서비스에 사용할 수 있는 스토리지 정책 확인

DevOps 엔지니어는 vSphere IaaS control plane 환경의 상태 저장 서비스에 사용하는 네임스페이스에 적절한 스토리지 클래스가 있는지 확인합니다. 스토리지 클래스는 vSAN SNA(비공유 아키텍처) 및 vSAN Direct일 수 있습니다.

vSAN 데이터 지속성 플랫폼은 vSphere 관리자가 상태 저장 서비스를 사용하도록 설정한 후 네임스페이스에 이러한 스토리지 클래스를 자동으로 생성합니다. vSphere IaaS control plane에서 상태 저장 서비스 사용의 내용을 참조하십시오.

참고 감독자에서 실행되는 애플리케이션만 vsan-direct 및 vsan-sna 스토리지 클래스를 사용할 수 있습니다. 이러한 스토리지 클래스는 Tanzu Kubernetes Grid 클러스터 내에서 사용할 수 없습니다.

vSphere 관리자는 기본 스토리지 클래스 외에 사용자 지정 스토리지 정책도 생성하여 네임스페이스에 할당할 수 있습니다. vSAN Direct 스토리지 정책 생성 및 vSAN SNA 스토리지 정책 생성의 내용을 참조하십시오.

절차

- vSAN SNA 및 vSAN Direct와 함께 사용할 스토리지 정책을 네임스페이스에서 사용할 수 있는지 확인합니다.

```
# 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 데이터 지속성 플랫폼에 대한 사용자 지정 스토리지 정책 생성

vSphere IaaS control plane의 감독자에서 상태 저장 서비스를 사용하도록 설정하는 경우 vSAN 데이터 지속성 플랫폼은 기본 스토리지 정책 및 해당 스토리지 클래스를 생성하고 이를 서비스 네임스페이스에 할당합니다. 이 정책은 vSAN SNA(비공유 아키텍처) 및 vSAN Direct 데이터스토어를 위한 것입니다. 기본값 대신 사용자 지정 스토리지 정책을 생성할 수 있습니다.

사용할 데이터스토어 유형을 결정하려면 다음 일반 권장 사항을 따르십시오.

- 비공유 클라우드 네이티브 서비스를 위한 전용 하드웨어 클러스터를 생성하는 경우에는 vSAN Direct를 사용합니다.
- 클라우드 네이티브 상태 저장 애플리케이션이 다른 일반 VM 또는 Kubernetes 워크로드와 물리적 인프라를 공유하도록 하려면 SNA가 포함된 vSAN을 사용합니다. 각 워크로드는 자체 스토리지 정책을 정의할 수 있으며, 단일 클러스터에서 두 환경 모두의 장점을 활용할 수 있습니다.

자세한 내용은 vSphere 비공유 스토리지의 내용을 참조하십시오.

정책을 생성한 후에는 상태 저장 서비스가 실행되는 네임스페이스에 정책을 할당할 수 있습니다. 감독자에서 vSphere 네임스페이스 생성 및 구성의 내용을 참조하십시오.

vSAN Direct 스토리지 정책 생성

vSAN Direct를 사용하는 경우 감독자 네임스페이스에서 사용할 스토리지 정책을 생성합니다. 이 스토리지 정책과 연결하는 네임스페이스에서 vSAN Direct와 호환되는 워크로드(예: 상태 저장 서비스 또는 인스턴스 스토리지 VM)를 실행할 수 있습니다.

절차

1 vSphere Client에서 **VM 스토리지 정책 생성** 마법사를 엽니다.

   a **홈** 메뉴에서 **정책 및 프로파일**을 클릭합니다.

   b **정책 및 프로파일**에서 **VM 스토리지 정책**을 클릭합니다.

   c **생성**을 클릭합니다.

2 정책 이름 및 설명을 입력합니다.

옵션	작업
vCenter Server	vCenter Server 인스턴스를 선택합니다.
이름	스토리지 정책의 이름을 입력합니다.
설명	스토리지 정책의 설명을 입력합니다.

3 **정책 구조** 페이지의 **데이터스토어별 규칙**에서 vSAN Direct 스토리지 배치에 대한 규칙을 사용하도록 설정합니다.

4 **vSAN Direct 규칙** 페이지에서 스토리지 배치 유형으로 vSAN Direct를 지정합니다.

5 **스토리지 호환성** 페이지에서 이 정책과 일치하는 vSAN Direct 데이터스토어 목록을 검토합니다.

6 **검토 및 완료** 페이지에서 스토리지 정책 설정을 검토하고 **마침**을 클릭합니다.

   설정을 변경하려면 **뒤로**를 클릭하여 해당 페이지로 이동합니다.

vSAN SNA 스토리지 정책 생성

vSAN 플랫폼에서 vSAN 데이터 지속성을 사용하는 경우 상태 저장 서비스가 실행되는 네임스페이스에 사용할 vSAN SNA(비공유 아키텍처) 스토리지 정책을 생성할 수 있습니다.

절차

1 vSphere Client에서 **VM 스토리지 정책 생성** 마법사를 엽니다.

   a **홈** 메뉴에서 **정책 및 프로파일**을 클릭합니다.

   b **정책 및 프로파일**에서 **VM 스토리지 정책**을 클릭합니다.

   c **생성**을 클릭합니다.

2 정책 이름 및 설명을 입력합니다.

옵션	작업
vCenter Server	vCenter Server 인스턴스를 선택합니다.
이름	스토리지 정책의 이름(예: **`Sample SNA Thick`**)을 입력합니다.
설명	스토리지 정책의 설명을 입력합니다.

3 **정책 구조** 페이지의 **데이터스토어별 규칙**에서 vSAN 스토리지 배치에 대한 규칙을 사용하도록 설정합니다.

4 **vSAN** 페이지에서 **가용성** 탭을 클릭하고 다음 값을 선택합니다. 이 값은 vSAN 데이터 지속성 플랫폼의 SNA 워크로드에만 적용됩니다. VM 워크로드 프로비저닝에는 사용할 수 없습니다.

옵션	설명
옵션	값
사이트 재해 허용	**없음 - 표준 클러스터** **참고** vSAN 데이터 지속성 플랫폼은 표준 클러스터만 지원합니다.
허용되는 실패 수	**호스트 선호도가 있는 데이터 중복성 없음**

SNA 워크로드에 대해 씩 프로비저닝이 적용되어 **고급 정책 규칙** 탭에서 개체 공간 예약 값으로 선택됩니다. 이 값은 변경할 수 없습니다.

5 **스토리지 호환성** 페이지에서 이 정책과 일치하는 vSAN 데이터스토어 목록을 검토합니다.

6 **검토 및 완료** 페이지에서 스토리지 정책 설정을 검토하고 **마침**을 클릭합니다.

설정을 변경하려면 **뒤로**를 클릭하여 해당 페이지로 이동합니다.

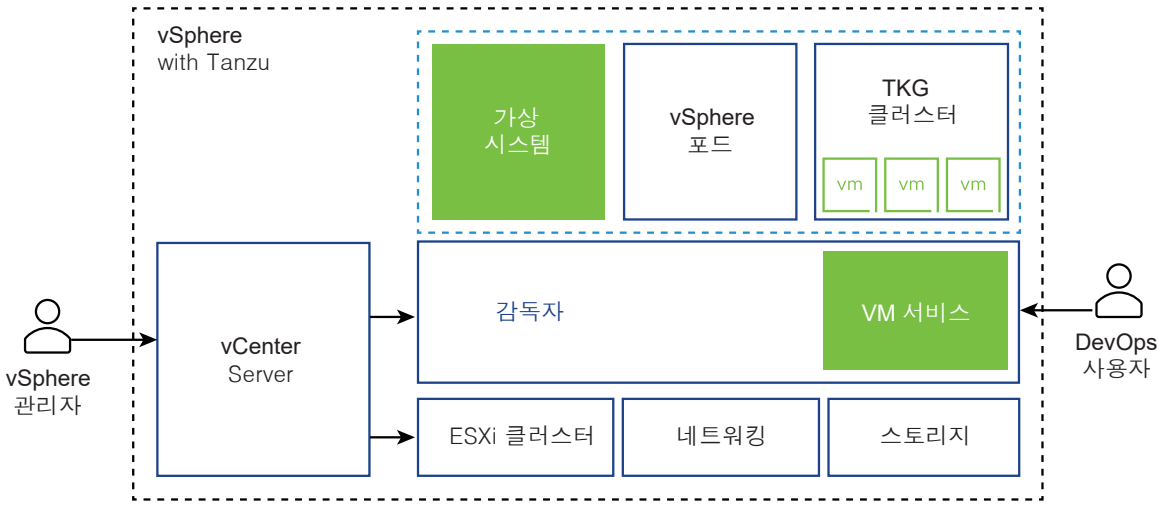
vSphere IaaS control plane에서 가상 시스템 배포 및 관리

6

vSphere IaaS control plane는 DevOps 엔지니어가 공통의 공유 Kubernetes 환경에서 컨테이너뿐 아니라 VM을 배포하고 실행할 수 있는 VM 서비스 기능을 제공합니다. VM 서비스를 사용하여 네임스페이스에서 가상 시스템의 수명 주기를 관리할 수 있습니다. VM 서비스는 Tanzu Kubernetes Grid 클러스터를 구성하는 VM 및 독립형 VM을 관리합니다.

VM 서비스는 Kubernetes를 사용하지만 쉽게 컨테이너화할 수 없는 기존 VM 기반 워크로드가 있는 DevOps 팀의 요구 사항을 해결합니다. 또한 컨테이너 플랫폼과 함께 Kubernetes가 아닌 플랫폼 관리의 오버헤드를 줄이는 데에도 도움이 됩니다. Kubernetes 플랫폼에서 컨테이너와 VM을 실행하는 경우 DevOps팀은 워크로드 공간을 하나의 플랫폼으로 통합할 수 있습니다.

참고 VM 서비스는 독립형 VM 외에도 Tanzu Kubernetes Grid 클러스터를 구성하는 VM을 관리합니다. 클러스터에 대한 자세한 내용은 "vSphere IaaS 제어부에서 TKG 서비스 사용" 설명서를 참조하십시오.



VM 서비스를 통해 배포된 각 VM은 vSphere IaaS control plane 인프라를 기반으로 자체 운영 체제를 포함한 모든 구성 요소를 실행하는 완전한 시스템으로 작동합니다. VM은 감독자가 제공하는 네트워킹 및 스토리지에 액세스할 수 있으며 표준 Kubernetes `kubectl` 명령을 사용하여 관리됩니다. VM은 Kubernetes 환경에서 다른 VM 또는 워크로드의 영향을 받지 않는 완전히 분리된 시스템으로 실행됩니다.

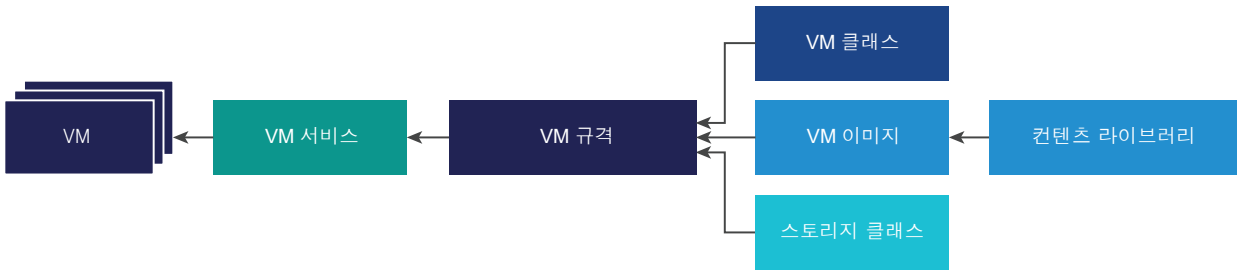
Kubernetes 플랫폼에서 가상 시스템을 사용하는 경우

일반적으로 컨테이너 또는 VM에서 워크로드를 실행하기로 결정하는 경우는 비즈니스 요구와 목표에 따라 다릅니다. VM을 사용하는 이유 중에는 다음과 같은 경우가 있습니다.

- 애플리케이션을 컨테이너화할 수 없습니다.
- 프로젝트에 대한 특정 하드웨어 요구 사항이 있습니다.
- 애플리케이션이 사용자 지정 커널 또는 사용자 지정 운영 체제용으로 설계되었습니다.
- 애플리케이션이 VM에서 실행하는 데 더 적합합니다.
- 일관된 Kubernetes 환경을 유지하고 오버헤드를 방지하려고 합니다. Kubernetes가 아닌 플랫폼 및 컨테이너 플랫폼에 대해 별도의 인프라를 실행하는 대신 해당 스택을 통합하고 익숙한 `kubectl` 명령으로 관리할 있습니다.

VM 서비스의 개념

vSphere 네임스페이스에 배포할 VM의 상태를 설명하려면 VM 클래스, VM 이미지 및 스토리지 클래스와 같은 매개 변수를 사용합니다. 그런 다음 VM 서비스는 이러한 규격을 함께 사용하여 독립형 VM 또는 Tanzu Kubernetes Grid 클러스터를 지원하는 VM을 생성합니다.



VM 서비스

VM 서비스는 VM 및 연결된 vSphere 리소스를 관리하기 위한 선언적 Kubernetes 스타일 API를 제공하는 vSphere IaaS control plane의 구성 요소입니다. VM 서비스를 통해 vSphere 관리자는 리소스를 제공하고 VM 클래스 및 VM 이미지와 같은 템플릿을 Kubernetes에 제공할 수 있습니다. DevOps 엔지니어는 이러한 리소스를 사용하여 원하는 VM 상태를 설명할 수 있습니다. DevOps 엔지니어가 VM 상태를 지정하면 VM 서비스는 지원 인프라 리소스에 대해 원하는 상태를 실현된 상태로 변환합니다.

VM 서비스를 통해 생성된 VM은 `kubectl` 명령을 사용하여 Kubernetes 네임스페이스에서만 관리할 수 있습니다. vSphere 관리자는 vSphere Client에서 VM을 관리할 수 없지만 세부 정보를 표시하고 사용하는 리소스를 모니터링할 수 있습니다. 자세한 내용은 vSphere IaaS control plane에서 사용 가능한 가상 시스템 모니터링 항목을 참조하십시오.

VM 클래스

VM 클래스는 VM에 대한 리소스 집합을 요청하는 데 사용할 수 있는 VM 규격입니다. VM 클래스는 vSphere 관리자가 제어 및 관리하며 가상 CPU 수, 메모리 용량 및 예약 설정과 같은 매개 변수를 정의합니다. 정의된 매개 변수는 감독자의 기본 인프라 리소스에 의해 지원되고 보장됩니다.

vSphere 관리자는 사용자 지정 VM 클래스를 생성할 수 있습니다.

또한 워크로드 관리는 몇 가지 기본 VM 클래스를 제공합니다. 일반적으로 각 기본 클래스 유형에는 보장됨 및 사용 시도라는 두 가지 버전이 있습니다. 보장됨 버전은 VM 규격이 요청하는 리소스를 완전히 예약합니다. 사용 시도 클래스 버전은 그렇지 않으며, 리소스가 오버 커밋되도록 허용합니다. 일반적으로 보장됨 유형은 운영 환경에서 사용됩니다.

기본 VM 클래스의 예에는 다음이 포함됩니다.

클래스	CPU	메모리(GB)	예약된 CPU 및 메모리
`guaranteed-large`	4	16	예
`best-effort-large`	4	16	아니요
`guaranteed-small`	2	4	예
`best-effort-small`	2	4	아니요

vSphere 관리자는 특정 네임스페이스 내에서 DevOps 엔지니어가 사용할 수 있도록 기존 VM 클래스를 원하는 만큼 할당할 수 있습니다.

VM 클래스는 DevOps 엔지니어에게 간소화된 환경을 제공합니다. DevOps는 생성하려는 각 VM의 전체 구성을 이해할 필요가 없습니다. 대신 사용 가능한 옵션에서 VM 클래스를 선택할 수 있으며 VM 서비스를 통해 VM 구성이 관리됩니다.

Kubernetes 측에서 VM 클래스는 `VirtualMachineClass` 리소스로 나타납니다.

VM 이미지

VM 이미지는 운영 체제, 애플리케이션 및 데이터를 비롯한 소프트웨어 구성이 포함된 템플릿입니다.

DevOps 엔지니어는 VM을 생성할 때 네임스페이스와 연결된 컨텐츠 라이브러리에서 이미지를 선택할 수 있습니다. DevOps에 이미지는 `VirtualMachineImage` 개체로 노출됩니다.

vSphere 관리자는 vSphere IaaS control plane와 호환되는 VM 이미지를 생성하고 컨텐츠 라이브러리에 이를 업로드할 수 있습니다.

컨텐츠 라이브러리

DevOps 엔지니어는 컨텐츠 라이브러리를 이미지 소스로 사용하여 VM을 생성합니다. VM 클래스와 마찬가지로 vSphere 관리자는 기존 컨텐츠 라이브러리를 네임스페이스 또는 클러스터에 할당하여 DevOps 엔지니어가 사용할 수 있도록 합니다. vSphere 관리자는 네임스페이스 컨텐츠 라이브러리를 쓰기 가능으로 만들 수도 있습니다. 이 추가 사용 권한을 통해 DevOps 사용자는 해당 이미지를 라이브러리에 게시할 수 있습니다.

스토리지 클래스

VM 서비스는 스토리지 클래스를 사용하여 가상 디스크를 배치하고 영구 볼륨을 동적으로 연결합니다. 스토리지 클래스에 대한 자세한 내용은 장 8 vSphere IaaS control plane에서 감독자 워크로드와 함께 영구 스토리지 사용 항목을 참조하십시오.

VM 규격

DevOps 엔지니어는 VM 이미지, VM 클래스 및 스토리지 클래스를 함께 제공하는 YAML 파일에서 원하는 VM 상태를 설명합니다.

Kubernetes용 VM Operator

VM Operator를 사용하면 Kubernetes 스타일의 선언적 API를 사용하여 가상 시스템을 관리할 수 있습니다.

vSphere 8.0 업데이트 3 릴리스부터 vSphere IaaS control plane은 VM Operator v1alpha2를 지원합니다. 다른 이점 중에서도 이 버전은 다음과 같은 기능을 제공합니다.

- 인라인 Cloud-Init 및 Windows 지원을 비롯한 부트스트랩 제공자 지원 향상.
- 게스트 네트워킹 구성이 향상.
- 확대된 상태 기능.
- 사용자 정의 준비 상태 게이트 지원.
- 새로운 VirtualMachineWebConsoleRequest API.

v1alpha2와 관련된 새로운 API 변경 사항 외에도 v1alpha2의 다른 API 대부분은 v1alpha1과 패리티로 작동할 수 있습니다. VM 규격의 대부분의 필드는 v1alpha1 이전 버전과 호환됩니다.

v1alpha2 릴리스 후에도 v1alpha1 개체를 계속 사용할 수 있습니다. 모든 v1alpha1 개체는 VM Operator에 내장된 변환 Webhook을 사용하여 v1alpha2로 자동 변환됩니다.

VM Operator v1alpha2 및 지원되는 필드에 대한 자세한 내용은 https://vm-operator.readthedocs.io/en/stable/ref/api/v1alpha2/ 항목을 참조하십시오.

네트워킹

VM 서비스에는 특정 요구 사항이 없으며 vSphere IaaS control plane에서 사용할 수 있는 네트워크 구성에 의존합니다. VM 서비스는 두 가지 유형의 네트워킹, 즉 vSphere네트워킹 또는 NSX를 지원합니다. VM이 배포되면 사용 가능한 네트워크 제공자가 정적 IP 주소를 VM에 할당합니다. 자세한 내용은 "vSphere IaaS 제어부 개념 및 계획" 설명서에서 감독자 네트워킹을 참조하십시오.

VM 서비스 및 vSphere 영역이 있는 감독자

3개 영역 감독자에서 VM을 생성하면 VM 인스턴스가 사용 가능한 모든 영역에 복제됩니다. YAML 파일을 통해 VM 배치를 제어하기 위해 DevOps 팀에서 Kubernetes 레이블 `topology.kubernetes.io/zone`을 사용할 수 있습니다. 예: `topology.kubernetes.io/zone: zone-a02`

VM 프로비저닝 및 모니터링 워크플로

vSphere 관리자는 VM의 정책 및 거버넌스에 대한 가드레일을 설정하고 VM 클래스 및 VM 템플릿과 같은 VM 리소스를 DevOps 엔지니어에게 제공합니다. VM이 배포된 후 vSphere Client를 사용하여 모니터링할 수 있습니다.

단계	수행자	설명
1	vSphere 관리자	vSphere IaaS control plane에서 독립형 VM에 대한 컨텐츠 라이브러리 생성 및 관리
2	vSphere 관리자	vSphere IaaS control plane의 VM 클래스 작업 NVIDIA vGPU를 사용하려면 VM 클래스에서 PCI 디바이스를 구성합니다. vSphere IaaS control plane에서 vGPU 및 기타 PCI 디바이스를 사용하여 VM 배포의 내용을 참조하십시오.
3	DevOp 엔지니어	vSphere IaaS control plane에서 독립형 VM 배포 Tanzu Kubernetes Grid 클러스터 VM에 대한 자세한 내용은 "vSphere IaaS 제어부에서 TKG 서비스 사용" 항목을 참조하십시오.
4	vSphere 관리자	vSphere IaaS control plane에서 사용 가능한 가상 시스템 모니터링
5	DevOps 엔지니어	vSphere IaaS control plane에서 컨텐츠 라이브러리 이미지 관리 및 게시

다음으로 아래 항목을 읽으십시오.

- vSphere IaaS control plane에서 독립형 VM에 대한 컨텐츠 라이브러리 생성 및 관리
- vSphere IaaS control plane의 VM 클래스 작업
- 데이터 센터 CLI를 사용하여 VM 클래스 생성 및 관리
- vSphere IaaS control plane에서 독립형 VM 배포
- vSphere IaaS control plane에서 vGPU 및 기타 PCI 디바이스를 사용하여 VM 배포
- vSphere IaaS control plane에 인스턴스 스토리지가 있는 VM 배포
- vSphere IaaS control plane에서 구성 가능한 OVF 속성을 사용하여 VM 배포
- vSphere IaaS control plane에서 사용 가능한 가상 시스템 모니터링
- vSphere VM 웹 콘솔을 사용하여 VM 문제 해결

vSphere IaaS control plane에서 독립형 VM에 대한 컨텐츠 라이브러리 생성 및 관리

vSphere IaaS control plane 환경에 가상 시스템을 배포하려면 DevOps 사용자가 운영 체제, 애플리케이션 및 데이터를 포함한 소프트웨어 구성이 포함된 VM 이미지 또는 템플릿에 액세스할 수 있어야 합니다. 이미지에 대한 액세스를 제공하기 위해 vSphere 관리자는 VM 컨텐츠 라이브러리를 구성한 다음 VM이 배포된 네임스페이스와 연결합니다. vSphere 8.0 업데이트 2부터 vSphere 관리자는 컨텐츠 라이브러리를 모든 네임스페이스에서 사용할 수 있도록 감독자 수준에서 할당할 수 있습니다.

vSphere IaaS control plane에서 독립형 VM에 대한 컨텐츠 라이브러리 생성

vSphere 관리자는 컨텐츠 라이브러리를 생성하여 VM 템플릿을 저장하고 관리합니다.

독립형 VM용 컨텐츠 라이브러리 생성

로컬 컨텐츠 라이브러리를 생성하여 템플릿 및 기타 파일 유형으로 채울 수 있습니다. 기존의 게시된 로컬 라이브러리에 있는 컨텐츠를 사용할 수 있도록 구독 라이브러리를 생성할 수도 있습니다.

컨텐츠 라이브러리의 항목을 보호하기 위해 OVF 보안 정책을 적용할 수 있습니다. OVF 보안 정책은 컨텐츠 라이브러리를 배포 또는 업데이트하거나, 컨텐츠 라이브러리로 항목을 가져오거나, 템플릿을 동기화할 때 엄격한 유효성 검사를 적용합니다. 템플릿이 신뢰할 수 있는 인증서로 서명되었는지 확인하기 위해, 신뢰할 수 있는 CA의 OVF 서명 인증서를 컨텐츠 라이브러리에 추가할 수 있습니다.

vSphere의 컨텐츠 라이브러리 및 VM 템플릿에 대한 자세한 내용은 "vSphere 가상 시스템 관리" 에서 컨텐츠 라이브러리 사용을 참조하십시오.

사전 요구 사항

필요한 권한:

- 라이브러리를 생성할 vCenter Server 인스턴스에 대한 **컨텐츠 라이브러리.로컬 라이브러리 생성** 또는 **컨텐츠 라이브러리.구독 라이브러리 생성**.
- 대상 데이터스토어에서 **데이터스토어.공간 할당**.

절차

1 **VM 서비스** 페이지로 이동합니다.

   a vSphere Client 홈 메뉴에서 **워크로드 관리**를 선택합니다.

   b **서비스** 탭을 클릭하고 **VM 서비스** 카드에서 **관리**를 클릭합니다.

2 **VM 서비스** 페이지에서 **컨텐츠 라이브러리 > 컨텐츠 라이브러리 생성**을 클릭합니다.

   이 작업을 수행하면 vSphere Client의 컨텐츠 라이브러리 섹션으로 이동됩니다.

3 **생성**을 클릭합니다.

   새 컨텐츠 라이브러리 마법사가 열립니다.

4 **이름 및 위치** 페이지에서 이름을 입력하고, 컨텐츠 라이브러리를 위한 vCenter Server 인스턴스를 선택한 후 **다음**을 클릭합니다.

   DevOps 팀이 라이브러리 항목을 쉽게 찾고 액세스할 수 있도록 컨텐츠 라이브러리에 대해 정보를 알려주는 이름을 사용해야 합니다.

5 **컨텐츠 라이브러리 구성** 페이지에서 생성하려는 컨텐츠 라이브러리 유형을 선택하고 **다음**을 클릭합니다.

옵션	설명
로컬 컨텐츠 라이브러리	기본적으로 로컬 컨텐츠 라이브러리는 사용자가 이를 생성한 vCenter Server 인스턴스에서만 액세스할 수 있습니다. a (선택 사항) 라이브러리의 컨텐츠를 다른 vCenter Server 인스턴스에서 사용할 수 있게 하려면 **게시 사용**을 선택합니다. b (선택 사항) 컨텐츠 라이브러리에 액세스할 때 암호를 사용하게 하려면 **인증 사용**을 선택하고 암호를 설정합니다.
구독 컨텐츠 라이브러리	구독 컨텐츠 라이브러리는 게시된 컨텐츠 라이브러리에서 만들어집니다. 기존 컨텐츠 라이브러리를 활용하려면 이 옵션을 사용합니다. 구독 라이브러리를 게시된 라이브러리와 동기화하여 최신 컨텐츠를 확인할 수 있지만, 구독 라이브러리에서 컨텐츠를 추가하거나 제거할 수는 없습니다. 게시된 라이브러리의 관리자만 게시된 라이브러리에서 컨텐츠를 추가하고 수정하고 제거할 수 있습니다. 라이브러리를 구독하려면 다음 정보를 제공합니다. a **구독 URL** 텍스트 상자에 게시된 라이브러리의 URL 주소를 입력합니다. b 게시된 라이브러리에서 인증을 사용할 수 있으면 **인증 사용**을 선택하고 게시자 암호를 입력합니다. c 구독 라이브러리의 컨텐츠에 대한 다운로드 방법을 선택합니다. ■ 구독 직후 게시된 라이브러리에 있는 모든 항목의 로컬 사본을 다운로드하려면 **즉시**를 선택합니다. ■ 스토리지 공간을 절약하려면 **필요한 경우**를 선택합니다. 게시된 라이브러리의 항목에 대한 메타데이터만 다운로드합니다. 항목을 사용해야 할 경우 항목 또는 전체 라이브러리를 동기화하여 해당 컨텐츠를 다운로드합니다. d 메시지가 표시되면 SSL 인증서 지문을 수락합니다. SSL 인증서 지문은 인벤토리에서 구독 컨텐츠 라이브러리를 삭제할 때까지 시스템에 저장됩니다.

6 (선택 사항) **보안 정책 적용** 페이지에서 **보안 정책 적용**을 선택하고 **OVF 기본 정책**을 선택합니다.

구독 라이브러리의 경우 이 옵션은 라이브러리가 보안 정책을 지원하는 경우에만 나타납니다.

이 옵션을 선택하면 로컬 호스트에서 라이브러리로 OVF 항목을 가져오거나 항목을 동기화할 때 엄격한 OVF 인증서 확인이 수행됩니다. 인증서 검증을 통과하지 못한 OVF 항목은 가져올 수 없습니다.

동기화 중에 항목이 검증을 통과하지 못하면 **확인 실패** 태그로 표시됩니다. 항목 및 메타데이터만 유지되고 항목의 파일은 유지되지 않습니다.

7 **스토리지 추가** 페이지에서 컨텐츠 라이브러리 컨텐츠의 스토리지 위치로 사용할 데이터스토어를 선택하고 **다음**을 클릭합니다.

8 **완료 준비** 페이지에서 세부 정보를 검토하고 **마침**을 클릭합니다.

독립형 VM에 대한 VM 이미지로 컨텐츠 라이브러리 채우기

컨텐츠 라이브러리를 생성한 후 OVA 또는 OVF 형식의 VM 템플릿으로 라이브러리를 채웁니다. DevOps 엔지니어는 이러한 템플릿을 사용하여 vSphere IaaS control plane 환경에서 새 독립형 가상 시스템을 프로비저닝할 수 있습니다.

몇 가지 방법을 사용하여 라이브러리를 채울 수 있습니다. 이 항목에서는 로컬 시스템이나 웹 서버에서 파일을 가져와서 로컬 컨텐츠 라이브러리에 항목을 추가하는 방법을 설명합니다. 컨텐츠 라이브러리를 채우는 다른 방법은 "vSphere 가상 시스템 관리" 에서 컨텐츠로 라이브러리 채우기를 참조하십시오.

참고 사용하는 VM 이미지에는 제한이 없습니다. 바로 사용할 수 있는 OVA 이미지를 테스트하려면 권장 이미지 페이지에서 다운로드하면 됩니다. 이러한 이미지는 POC용으로만 사용됩니다. 운영 환경에서는 회사 보안 정책을 따르는 최신 패치 및 필수 보안 설정으로 이미지를 생성합니다.

사전 요구 사항

- vSphere IaaS control plane와 호환되는 VM 이미지를 생성합니다.

  이미지 규격을 사용하려면 모든 VM 이미지에 VMware Tools 또는 이에 상응하는 오픈 소스 패키지가 포함되어야 합니다. 이미지는 다음 중 하나를 사용하여 게스트 운영 체제 및 해당 네트워킹 스택을 부트스트랩해야 합니다. 자세한 내용은 부트스트랩 제공자를 참조하십시오.

  - Linux + Cloud-Init 버전 17.9-21.2(DataSourceVMwareGuestInfo 포함).
  - Linux + Cloud-Init 버전 21.3+
  - Windows + Cloudbase-Init 버전 1.1.0+
  - Windows + Sysprep(시스템 준비)

  Cloud-Init에 대한 자세한 내용은 클라우드 인스턴스를 사용자 지정하기 위한 표준의 공식 설명서를 참조하십시오.

  Sysprep에 대한 자세한 내용은 Sysprep 개요의 공식 설명서를 참조하십시오.
- 라이브러리가 보안 정책으로 보호되는 경우 모든 라이브러리 항목이 준수 상태인지 확인합니다. 보호된 라이브러리에 준수 및 비준수 항목이 혼합되어 있는 경우 `kubectl get virtualmachineimages`가 DevOps 엔지니어에게 VM 이미지를 표시하지 못합니다.
- 필요한 권한: 라이브러리에 대한 **컨텐츠 라이브러리.라이브러리 항목 추가** 및 **컨텐츠 라이브러리.파일 업데이트**.

절차

1 vSphere Client 홈 메뉴에서 **컨텐츠 라이브러리**를 선택합니다.

2 로컬 컨텐츠 라이브러리를 마우스 오른쪽 버튼으로 클릭하고 **항목 가져오기**를 선택합니다.

  라이브러리 항목 가져오기 대화 상자가 열립니다.

3 **소스** 섹션에서 항목의 소스를 선택합니다.

옵션	설명
URL	항목이 있는 웹 서버의 경로를 입력합니다. **참고** `.ovf` 또는 `.ova` 파일을 가져올 수 있습니다. 이 경우 추가되는 컨텐츠 라이브러리 항목은 OVF 템플릿 유형입니다.
로컬 파일	**파일 업로드**를 클릭하고 로컬 시스템에서 가져올 파일로 이동합니다. 드롭다운 메뉴를 사용하여 로컬 시스템에서 파일을 필터링할 수 있습니다. **참고** `.ovf` 또는 `.ova` 파일을 가져올 수 있습니다. OVF 템플릿을 가져올 때는 먼저 OVF 설명자 파일(`.ovf`)을 선택합니다. 그러면 OVF 템플릿의 다른 파일(예: `.vmdk` 파일)을 선택하라는 메시지가 표시됩니다. 이 경우 추가되는 컨텐츠 라이브러리 항목은 OVF 템플릿 유형입니다.

vCenter Server는 가져오기 중에 OVF 패키지에 있는 매니페스트 및 인증서 파일을 읽고 검증합니다. vCenter Server가 만료된 인증서를 감지한 경우와 같이 인증서 문제가 있으면 **라이브러리 항목 가져오기** 마법사에 주의가 표시됩니다.

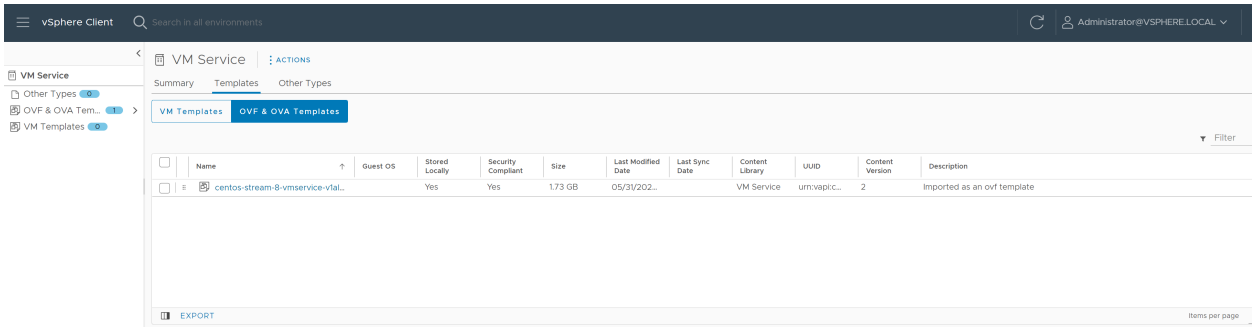
참고 로컬 시스템에 있는 `.ovf` 파일에서 OVF 패키지를 가져오는 경우 vCenter Server는 서명된 컨텐츠를 읽지 않습니다.

4 **대상** 섹션에 항목의 이름 및 설명을 입력합니다.

5 **가져오기**를 클릭합니다.

결과

템플릿 탭 또는 **기타 유형** 탭에 항목이 표시됩니다.



vSphere IaaS control plane에서 VM 컨텐츠 라이브러리 추가 및 관리

컨텐츠 라이브러리를 생성하고 VM 템플릿으로 채운 후 vSphere Client를 사용하여 라이브러리를 네임스페이스에 추가합니다. 라이브러리를 네임스페이스에 추가하여 DevOps 사용자에게 라이브러리에 대한 액세스 권한을 부여합니다. 또한 DCLI(Data Center CLI) 명령을 사용하여 쓰기 가능 또는 읽기 전용 컨텐츠 라이브러리를 네임스페이스에 추가하거나 클러스터 수준에서 읽기 전용 라이브러리를 할당할 수 있습니다.

vSphere Client를 사용하여 네임스페이스에 VM 컨텐츠 라이브러리 추가

vSphere Client를 사용하여 추가하는 컨텐츠 라이브러리는 읽기 전용입니다. DevOps 사용자는 이 컨텐츠 라이브러리의 이미지에 액세스할 수 있지만 이 라이브러리에 VM 이미지를 게시할 수는 없습니다.

여러 컨텐츠 라이브러리를 단일 네임스페이스에 추가할 수 있습니다. 동일한 컨텐츠 라이브러리를 서로 다른 네임스페이스에 추가할 수 있습니다.

참고 이 절차는 VM 서비스의 컨텐츠 라이브러리에만 적용됩니다. Tanzu Kubernetes Grid용 컨텐츠 라이브러리는 Tanzu Kubernetes Grid 카드에서 관리해야 합니다.

사전 요구 사항

필요한 권한:

- **네임스페이스.클러스터 전체 구성 수정**
- **네임스페이스.네임스페이스 구성 수정**

절차

1 vSphere Client에서 네임스페이스로 이동합니다.

   a vSphere Client 홈 메뉴에서 **워크로드 관리**를 선택합니다.

   b **네임스페이스** 탭을 클릭하고 네임스페이스를 클릭합니다.

2 컨텐츠 라이브러리를 추가합니다.

   a **VM 서비스** 카드에서 **컨텐츠 라이브러리 추가**를 클릭합니다.

   b 컨텐츠 라이브러리를 하나 또는 여러 개 선택하고 **확인**을 클릭합니다.

vSphere Client를 사용하여 네임스페이스에서 VM 컨텐츠 라이브러리 관리

라이브러리를 네임스페이스와 연결한 후 vSphere Client를 사용하여 네임스페이스에서 라이브러리를 제거할 수 있습니다. 라이브러리를 더 추가할 수도 있습니다.

네임스페이스에서 컨텐츠 라이브러리를 제거해도 이전에 라이브러리 이미지를 통해 배포된 VM에는 영향을 주지 않습니다.

참고 이 절차는 VM 서비스의 컨텐츠 라이브러리에만 적용됩니다. Tanzu Kubernetes Grid 컨텐츠 라이브러리는 Tanzu Kubernetes Grid 카드에서 관리해야 합니다.

사전 요구 사항

필요한 권한:

- **네임스페이스.클러스터 전체 구성 수정**
- **네임스페이스.네임스페이스 구성 수정**

절차

1 vSphere Client에서 네임스페이스로 이동합니다.

   a vSphere Client 홈 메뉴에서 **워크로드 관리**를 선택합니다.

   b **네임스페이스** 탭을 클릭하고 네임스페이스를 클릭합니다.

2 컨텐츠 라이브러리를 추가하거나 제거합니다.

   a **VM 서비스** 카드에서 **컨텐츠 라이브러리 관리**를 클릭합니다.

   b 다음 작업 중 하나를 수행합니다.

옵션	설명
컨텐츠 라이브러리 제거	컨텐츠 라이브러리를 선택 취소하고 **확인**을 클릭합니다.
컨텐츠 라이브러리 추가	컨텐츠 라이브러리를 하나 또는 여러 개 선택하고 **확인**을 클릭합니다.

다음에 수행할 작업

라이브러리의 OVF 템플릿은 Kubernetes 네임스페이스에서 VM 이미지로 사용할 수 있게 되며 DevOps에서 셀프 서비스 VM에 사용할 수 있습니다. vSphere IaaS control plane에서 가상 시스템 배포의 내용을 참조하십시오.

참고 라이브러리의 OVF 템플릿만 네임스페이스에 표시됩니다. 다른 유형의 컨텐츠는 네임스페이스에 표시되지 않습니다.

데이터 센터 CLI를 사용하여 네임스페이스에 VM 컨텐츠 라이브러리 추가

vSphere 관리자는 DCLI(Data Center CLI) 명령을 사용하여 컨텐츠 라이브러리를 네임스페이스에 할당할 수 있습니다. 라이브러리를 할당할 때 네임스페이스와 연결된 라이브러리를 쓰기 가능으로 만들 수 있습니다. 라이브러리에 쓰기가 가능하면, 라이브러리 및 라이브러리의 이미지를 보는 것 외에도 DevOps 사용자가 새 VM 이미지를 게시할 수 있습니다.

DCLI 명령을 사용하여 로컬, 게시됨 및 구독을 포함한 모든 유형의 라이브러리를 네임스페이스에 추가할 수 있습니다. 단, 로컬 및 게시된 라이브러리만 쓰기 가능한 라이브러리로 연결할 수 있습니다. 컨텐츠 라이브러리 및 라이브러리 항목은 연결된 네임스페이스에서만 사용할 수 있습니다.

절차

1 루트 사용자 계정을 사용하여 vCenter Server에 로그인합니다.

2 대화형 모드에서 DCLI를 사용하기 위해 `dcli +i`를 입력합니다.

3 네임스페이스와 연결할 컨텐츠 라이브러리의 ID를 가져옵니다.

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

4 다음 명령을 실행하여 컨텐츠 라이브러리를 네임스페이스와 연결합니다.

   업데이트 작업이 증분되지 않습니다. 목록에 지정된 라이브러리만 네임스페이스와 연결되며 이전에 추가된 라이브러리는 해당 ID가 지정되지 않는 한 제거됩니다. 예를 들어 `'[{"content_library": "CLA",`

`"writable": "true"}]'`를 업데이트한 후 나중에 `'[{"content_library": "CLB", "writable": "true"}]'`를 업데이트하면 CLA는 제거되고 CLB만 추가됩니다. CLA와 CLB를 모두 연결하려면 두 라이브러리를 모두 지정해야 합니다(`'[{"content_library": "CLA", "writable": "true"}, {"content_library": "CLB", "writable": "true"}]'`).

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

다음 인수를 사용합니다.

- `--namespace` *`namespace_name`* – 네임스페이스의 이름입니다.
- `--content_libraries` *`content_library_ID`* `writable: true | false` – 네임스페이스와 연결할 컨텐츠 라이브러리의 ID 및 라이브러리에 쓰기가 가능한지 여부입니다.

예를 들면 다음과 같습니다.

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

5 네임스페이스에서 컨텐츠 라이브러리를 삭제하려면 어레이 목록에서 컨텐츠 라이브러리 항목을 제거하는 `namespaces instances update` 명령을 반복합니다.

예를 들면 다음과 같습니다.

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

결과

추가된 컨텐츠 라이브러리는 DevOps 네임스페이스 보기에서 사용할 수 있게 됩니다.

DevOps 사용자는 다음 명령을 실행하여 컨텐츠 라이브러리가 추가 또는 삭제되었는지 확인할 수 있습니다.

```
kubectl get cl -n lb-edit-ns
   NAMESPACE    NAME                   VSPHERENAME   TYPE    WRITABLE   STORAGETYPE   AGE
   lb-edit-ns   cl-b585915ddxxxxxxxxx   Test-ns-cl    Local   true       Datastore     3m9s
kubectl describe cl cl-b585915ddxxxxxxxxx -n lb-edit-ns
kubectl get clitem -n lb-edit-ns
```

Data Center CLI를 사용하여 감독자에 VM 컨텐츠 라이브러리 추가

네임스페이스 수준에서 컨텐츠 라이브러리를 할당하는 것 외에도 vSphere 관리자는 DCLI(Data Center CLI) 명령을 사용하여 라이브러리를 감독자 클러스터와 연결할 수 있습니다. 컨텐츠 라이브러리는 감독자의 모든 네임스페이스에서 사용할 수 있게 됩니다.

로컬, 게시됨 및 구독을 포함한 모든 유형의 라이브러리를 연결할 수 있습니다.

참고 감독자와 연결된 컨텐츠 라이브러리는 읽기 전용입니다. DevOps 사용자는 이 컨텐츠 라이브러리의 VM 이미지에만 액세스할 수 있지만 이 라이브러리에 VM 이미지를 게시할 수는 없습니다.

사전 요구 사항

DCLI 명령에 대한 자세한 내용은 VMware Data Center CLI를 참조하십시오.

절차

1 루트 사용자 계정을 사용하여 vCenter Server에 로그인합니다.

2 대화형 모드에서 DCLI를 사용하기 위해 `dcli +i`를 입력합니다.

3 감독자에 연결할 컨텐츠 라이브러리의 ID 및 감독자의 이름을 가져옵니다.

a 클러스터 목록에서 감독자의 이름을 가져옵니다.

명령은 vCenter Server에서 사용 가능한 모든 클러스터를 나열합니다.

```
dcli > namespacemanagement clusters list
```

b vCenter Server에서 사용 가능한 모든 유형의 모든 컨텐츠 라이브러리의 ID를 나열합니다.

```
dcli > library list
```

c 특정 라이브러리에 대한 세부 정보를 확인합니다.

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

4 하나 이상의 컨텐츠 라이브러리를 감독자와 연결합니다.

업데이트 작업이 증분되지 않습니다. 목록에 지정된 라이브러리만 네임스페이스와 연결되며 이전에 추가된 라이브러리는 해당 ID가 지정되지 않는 한 제거됩니다. 예를 들어 `'[{"content_library": "CLA", "writable": "true"}]'`를 업데이트한 후 나중에 `'[{"content_library": "CLB", "writable": "true"}]'`를 업데이트하면 CLA는 제거되고 CLB만 추가됩니다. CLA와 CLB를 모두 연결하려면 두 라이브러리를 모두 지정해야 합니다(`'[{"content_library": "CLA", "writable": "true"}, {"content_library": "CLB", "writable": "true"}]'`).

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

다음 인수를 사용합니다.

- `--cluster cluster_name` – 감독자 클러스터의 식별자입니다.
- `--content-libraries content_library_ID` – 감독자와 연결할 컨텐츠 라이브러리의 ID입니다. 여러 ID를 나열할 수 있습니다.

  예를 들면 다음과 같습니다.

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

5 컨텐츠 라이브러리가 클러스터에 연결되어 있는지 확인하십시오.

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

출력에는 연결된 컨텐츠 라이브러리의 ID가 포함되어야 합니다.

6 클러스터에서 연결된 컨텐츠 라이브러리를 삭제하려면 컨텐츠 라이브러리 어레이 목록에서 컨텐츠 라이브러리 항목을 제거하는 `namespacemanagement clusters update` 명령을 반복합니다.

예를 들면 다음과 같습니다.

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

결과

새로 추가된 컨텐츠 라이브러리는 DevOps 클러스터 보기에서 사용할 수 있게 됩니다. vSphere 관리자가 컨텐츠 라이브러리에 대해 변경한 내용은 DevOps 보기에 반영됩니다. DevOps 사용자는 다음 명령을 실행하여 컨텐츠 라이브러리를 나열하고 컨텐츠를 설명할 수 있습니다.

- `kubectl get ccl` – 클러스터 수준에서 사용할 수 있는 모든 컨텐츠 라이브러리의 목록입니다. 출력은 다음과 유사할 수 있습니다.

```
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` – 클러스터 수준의 컨텐츠 라이브러리에 있는 모든 항목의 목록입니다.
- `kubectl describe ccl NAME` – 클러스터 수준의 특정 컨텐츠 라이브러리에 대한 자세한 정보입니다.

vSphere IaaS control plane에서 컨텐츠 라이브러리 이미지 관리 및 게시

vSphere 관리자가 컨텐츠 라이브러리를 네임스페이스 또는 클러스터에 할당하면 DevOps 사용자는 라이브러리에 액세스하고 해당 항목을 사용하여 라이브러리의 VM 이미지에서 VM을 배포할 수 있습니다. 네임스페이스에 할당된 라이브러리에 쓰기 가능한 경우 편집 권한이 있는 DevOps 사용자는 라이브러리 항목을 관리하고 새 VM 이미지를 게시할 수도 있습니다.

참고 사용하는 VM 이미지에는 제한이 없습니다. 바로 사용할 수 있는 OVA 이미지를 테스트하려면 권장 이미지 페이지에서 다운로드하면 됩니다. 이러한 이미지는 POC용으로만 사용됩니다. 운영 환경에서는 회사 보안 정책을 따르는 최신 패치 및 필수 보안 설정으로 이미지를 생성합니다.

사전 요구 사항

DevOps 사용자는 다음 요구 사항을 따라야 합니다.

- vSphere 네임스페이스에 대한 `Edit` 권한이 있습니다.
- vSphere 관리자가 쓰기 가능한 컨텐츠 라이브러리를 네임스페이스에 할당했습니다. Data Center CLI를 사용하여 감독자에 VM 컨텐츠 라이브러리 추가 의 내용을 참조하십시오.
- 컨텐츠 라이브러리가 로컬이거나 게시되어 있습니다. 구독한 라이브러리는 편집할 수 없습니다.

절차

1 라이브러리 항목을 관리합니다.

a 네임스페이스에서 컨텐츠 라이브러리를 사용할 수 있는지 확인합니다.

참고 라이브러리에서 라이브러리 항목을 관리하거나 라이브러리에 VM 이미지를 게시하려면 writable 상태가 true인지 확인합니다.

```
kubectl get cl -n <namespace-name>
   NAME                    VSPHERENAME   TYPE    WRITABLE   STORAGETYPE   AGE
   cl-b585915ddxxxxxxxx    Test-ns-cl-1  Local   true       Datastore     3m9s
   cl-535d4b3dnxxxyyyyy    Test-ns-cl-1  Local   false      Datastore     3m9s
```

b 라이브러리의 컨텐츠를 확인합니다.

```
kubectl get clitem -n <namespace-name>
 NAME                 VSPHERENAME     CONTENTLIBRAYREF          TYPE   READY   AGE
 clitem-d2wnmq.....   item 1          cl-b585915ddxxxxxxxx      Ovf    True    26c
 clitem-55088d.....   item 2          cl-b585915ddxxxxxxxx      Ovf    True    26c
 clitem-xyzxyz.....   xyzxyz          cl-535d4b3dnxxxyyyyy      Ovf    True    26c
```

c 컨텐츠 라이브러리에서 이미지를 삭제합니다.

참고 쓰기 가능한 라이브러리에서만 항목을 삭제할 수 있으며, `Edit` 권한이나 더 높은 수준의 권한이 있는 경우에만 가능합니다.

clitem을 삭제하면 해당 vmi 리소스도 삭제됩니다.

```
kubectl delete clitem clitem-55088d.....
 NAME                 VSPHERENAME     CONTENTLIBRAYREF          TYPE   READY   AGE
 clitem-d2wnmq.....   item 1          cl-b585915ddxxxxxxxx      Ovf    True    26c
 clitem-xyzxyz.....   xyzxyz          cl-535d4b3dnxxxyyyyy      Ovf    True    26c
```

d 이미지 세부정보를 가져옵니다.

```
kubectl get vmi -n <namespace-name>
 NAME              PROVIDER-NAME      CONTENT-LIBRAY-NAME    IMAGE-NAME VERSION   OS-
TYPE        FORMAT   AGE
 vmi-d2wnmq.....   clitem-d2wnmq..... cl-b585915ddxxxxxxxx   item 1
ubuntu64guest  ovf      26c
 vmi-55088d.....   clitem-55088d..... cl-b585915ddxxxxxxxx   item 2
otherguest     ovf      26c
```

2 컨텐츠 라이브러리에 이미지를 게시합니다.

a 소스 VM을 배포하기 위한 yaml 파일을 생성합니다.

VirtualMachine 규격의 imageName이 컨텐츠 라이브러리의 VM 이미지 중 하나를 참조하는지 확인합니다.

예: `source-vm.yaml`

```
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 배포된 VM에 대한 정보를 얻고 VM에 연결하여 실행 중인지 확인합니다.

다음과 유사한 출력이 표시됩니다.

```
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 새 대상 이미지에 대한 게시 요청을 생성합니다.

예를 들면 `vmpub.yaml`입니다. 요청에서 소스 VM 및 이미지를 게시할 대상 컨텐츠 라이브러리의 이름을 나타냅니다. 라이브러리에 쓰기 가능한지 확인합니다.

```
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 게시 요청을 설명합니다.

게시 요청이 `ImageName`이 설정된 `Ready` 상태인지 확인합니다.

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

e 게시 요청이 완료된 후 새 이미지가 컨텐츠 라이브러리에 추가되었는지 확인합니다.

```
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
```

이 새 이미지를 사용하여 새 VM을 배포할 수 있습니다.

vSphere IaaS control plane의 VM 클래스 작업

vSphere IaaS control plane의 VM을 셀프 서비스하려면 DevOps 사용자가 VM 클래스에 액세스할 수 있어야 합니다. VM 클래스는 VM에 대한 CPU, 메모리 및 예약을 정의하는 템플릿입니다. VM 클래스는 개발 요구 사항을 예측하고 리소스 가용성 및 제약 조건을 고려하여 VM의 정책 및 거버넌스에 대한 가드레일을 설정하는 데 유용합니다.

vSphere IaaS control plane는 몇 가지 기본 VM 클래스를 제공합니다. vSphere 관리자는 그대로 사용하거나 사용자 지정 VM 클래스를 생성할 수 있습니다. DevOps 사용자가 클래스를 사용할 수 있도록 하려면 vSphere 관리자가 해당 클래스를 네임스페이스에 추가합니다. 네임스페이스에 할당된 VM 클래스는 독립형 VM 및 Tanzu Kubernetes Grid 클러스터를 구성하는 VM에서 사용할 수 있습니다.

vSphere Client를 사용하여 사용자 지정 VM 클래스 생성

vSphere 관리자는 사용 가능한 기본 클래스를 사용할 수 있습니다. 기본 클래스 대신 사용자 지정 VM 클래스를 생성하고 이를 네임스페이스에서 VM 배포에 사용할 수도 있습니다.

새 클래스를 생성하는 경우 다음과 같은 사항을 고려해야 합니다.

- vCenter Server 인스턴스에서 생성하는 VM 클래스는 모든 vCenter Server 클러스터 및 이러한 클러스터의 모든 네임스페이스에서 사용할 수 있습니다.

- VM 클래스는 vCenter Server의 모든 네임스페이스에서 사용할 수 있습니다. 단, DevOps 엔지니어는 사용자가 특정 네임스페이스와 연결한 VM 클래스만 사용할 수 있습니다.

참고 DCLI 명령을 사용하여 VM 클래스를 생성할 수도 있습니다. 데이터 센터 CLI를 사용하여 VM 클래스 생성 및 관리의 내용을 참조하십시오.

사전 요구 사항

필요한 권한:

- **네임스페이스.클러스터 전체 구성 수정**
- **네임스페이스.네임스페이스 구성 수정**
- **가상 시스템 클래스.가상 시스템 클래스 관리**

절차

1 **VM 서비스** 페이지로 이동합니다.

   a vSphere Client 홈 메뉴에서 **워크로드 관리**를 선택합니다.

   b **서비스** 탭을 클릭하고 **VM 서비스** 창에서 **관리**를 클릭합니다.

2 **VM 서비스** 페이지에서 **VM 클래스**를 클릭하고 **VM 클래스 생성**을 클릭합니다.

3 **이름** 페이지에서 VM 클래스 이름을 지정하고 **다음**을 클릭합니다.

   VM 클래스 이름은 VM 클래스를 식별합니다. 다음 요구 사항을 따르는 고유한 DNS 규정 준수 이름을 입력합니다.

   - 사용자 환경에서 기본 또는 사용자 지정 VM 클래스의 이름과 중복되지 않는 고유한 이름을 사용합니다.
   - 영숫자 문자열(최대 길이 63자)을 사용합니다.
   - 대문자나 공백은 사용하지 마십시오.
   - 대시는 첫 번째 또는 마지막 문자를 제외한 아무 곳에나 사용합니다. 예: `vm-class1`.

   VM 클래스를 생성한 후에는 이름을 변경할 수 없습니다.

4 **호환성** 페이지에서 VM 클래스 하드웨어 호환성을 선택하고 **다음**을 클릭합니다.

   자세한 내용은 가상 시스템 호환성을 참조하십시오.

   참고 VM 클래스의 하드웨어 호환성은 VM 클래스를 생성하는 동안에만 설정할 수 있으며 나중에 변경할 수 없습니다.

5 **구성** 페이지에서 기본값을 그대로 둡니다.

6 **검토 및 확인** 페이지에서 세부 정보를 검토하고 **마침**을 클릭합니다.

다음에 수행할 작업

VM 하드웨어 및 VM 옵션과 같은 VM 클래스 구성을 편집합니다.

vSphere Client를 사용하여 VM 클래스 편집

생성 후 VM 클래스를 편집하는 방법을 확인합니다. CPU, 메모리 및 디바이스와 같은 하드웨어 리소스를 구성하고 VM 옵션 및 고급 매개 변수를 편집할 수 있습니다. 또한 vSphere IaaS control plane에 제공된 기본 VM 클래스를 편집할 수 있습니다.

VM 클래스를 편집해도 이전에 이 클래스에서 배포된 VM이 자동으로 재구성되지는 않습니다. 예를 들어 DevOps 사용자가 VM 클래스를 사용하여 Tanzu Kubernetes Grid 클러스터를 생성한 후 나중에 사용자가 VM 클래스 정의를 변경해도 기존 Tanzu Kubernetes Grid VM은 영향을 받지 않습니다. 새 Tanzu Kubernetes Grid VM은 수정된 클래스 정의를 사용합니다.

경고 클러스터에서 사용되는 VM 클래스를 편집한 후 Tanzu Kubernetes Grid 클러스터를 확장하면, 새 클러스터 노드는 업데이트된 클래스 정의를 사용하지만 기존 클러스터 노드는 초기 클래스 정의를 계속 사용하여 불일치가 발생합니다. 제어부 및 작업자 노드를 모두 확장/축소할 수 있습니다. 크기 조정에 대한 자세한 내용은 "vSphere IaaS 제어부에서 TKG 서비스 사용" 에서 워크로드 클러스터 크기 조정을 참조하십시오.

VM 클래스를 삭제하면 연결된 모든 네임스페이스에서 제거됩니다. DevOps 사용자는 더 이상 해당 VM 클래스를 사용하여 VM을 셀프 서비스할 수 없습니다. 이 VM 클래스로 이미 생성된 VM은 영향을 받지 않습니다.

사전 요구 사항

필요한 권한:

- **네임스페이스.클러스터 전체 구성 수정**
- **네임스페이스.네임스페이스 구성 수정**
- **가상 시스템 클래스.가상 시스템 클래스 관리**

절차

1 vSphere Client에서 사용 가능한 VM 클래스를 표시합니다.

   a vSphere Client 홈 메뉴에서 **워크로드 관리**를 선택합니다.

   b **서비스** 탭을 클릭하고 **VM 서비스** 창을 클릭합니다.

   c **VM 서비스** 페이지에서 **VM 클래스**를 클릭합니다.

   모든 기본 또는 사용자 생성 VM 클래스는 **사용 가능한 VM 클래스** 아래에 표시됩니다.

2 선택한 VM 클래스 창에서 **관리**를 클릭하고 **편집**을 클릭합니다.

3 **가상 하드웨어** 페이지에서 메모리, CPU, 기타 디바이스 등 VM 클래스의 하드웨어 리소스를 구성합니다.

   모든 VM 하드웨어 설정은 DevOps 사용자가 VM에 VM 클래스를 할당할 때 적용됩니다. 예를 들어 CPU 구성 값은 DevOps 사용자가 VM 클래스를 사용하여 생성하는 모든 VM 전용 CPU 리소스가 됩니다.

   참고 vSphere 8.0 업데이트 2b부터 VM 클래스 생성 및 편집 마법사가 CPU 및 메모리 리소스를 백분율로 설정하는 것에서 숫자 값(MB, GB, TB 및 HMz)으로 변경됩니다. 이전에 생성된 모든 VM 클래스의 경우 CPU 및 메모리가 백분율로 표시되지만 이제 새 숫자 형식으로 이러한 값을 편집할 수 있습니다.

VM 구성 옵션	설명
CPU	VM 전용 CPU 리소스를 정의합니다. CPU 리소스 구성에 대한 자세한 내용은 가상 CPU 구성 및 제한 사항 및 가상 시스템의 CPU 리소스 구성을 참조하십시오.
메모리	VM에 대해 구성된 메모리를 MB, GB 또는 TB로 정의합니다. VM 메모리 리소스에 대한 자세한 내용은 가상 메모리 구성을 참조하십시오.
비디오 카드	Windows AERO, CAD, Google Earth 및 기타 3D 설계, 모델링 및 멀티미디어 애플리케이션을 활용하도록 3D 그래픽을 구성합니다. 비디오 카드 설정에 대한 자세한 내용은 3D 그래픽을 구성하는 방법을 참조하십시오.
보안 디바이스	vSGX(Software Guard Extensions)를 구성®하여 VM 클래스에 추가적인 보안을 제공합니다. Intel Software Guard Extensions를 사용하여 가상 시스템 보호를 참조하십시오.

4 **가상 하드웨어** 옵션에서 **새 디바이스 추가**를 클릭하여 VM 클래스에 디바이스를 추가하고 구성합니다.

스토리지 컨트롤러, 네트워크 어댑터, USB 및 PCI 디바이스와 같은 서로 다른 디바이스를 VM 클래스에 구성합니다.

VM 구성 옵션	설명
RDM 디스크	RDM(원시 디바이스 매핑)을 추가하여 가상 시스템 데이터를 가상 디스크 파일에 저장하는 대신 SAN LUN에 직접 저장합니다. RDM 디스크를 가상 시스템에 추가를 참조하십시오.
호스트 USB 디바이스	물리적 디바이스가 가상 시스템이 실행되는 호스트에 연결된 경우 ESXi 호스트에서 가상 시스템에 USB 패스스루 디바이스를 하나 이상 추가합니다. ESXi 호스트의 USB 디바이스를 가상 시스템에 추가를 참조하십시오.
NVDIMM	가상 NVDIMM 디바이스를 VM 클래스에 구성하여 비휘발성 또는 영구 컴퓨터 메모리를 사용할 수 있도록 합니다. 가상 시스템에 NVDIMM 디바이스 추가를 참조하십시오.
CD/DVD 드라이브	CD/DVD 디바이스를 VM 클래스로 구성합니다. 가상 시스템 CD 또는 DVD 드라이브를 추가하거나 수정하는 방법을 참조하십시오.
NVMe 컨트롤러, SATA 컨트롤러, SCSI 컨트롤러	스토리지 컨트롤러를 VM 클래스로 구성합니다. SCSI, SATA 및 NVMe 스토리지 컨트롤러 조건, 제한 및 호환성을 참조하십시오.
USB 컨트롤러	VM 클래스에 USB 컨트롤러를 추가하여 ESXi 호스트 또는 클라이언트 계산에서 USB 패스스루를 지원합니다. USB 컨트롤러를 가상 시스템에 추가를 참조하십시오.

VM 구성 옵션	설명
PCI 디바이스	vSphere IaaS control plane 환경의 ESXi 호스트에 NVIDIA GRID GPU 그래픽 디바이스가 하나 이상 있는 경우 NVIDIA GRID vGPU(가상 GPU) 기술을 사용하도록 VM을 구성합니다. ESXi 호스트의 다른 PCI 디바이스를 패스스루 모드의 VM에서 사용할 수 있도록 구성할 수도 있습니다. 이 옵션을 선택하면 메모리 리소스 예약 값이 자동으로 100%로 변경됩니다. 자세한 내용 및 추가 요구 사항은 vSphere IaaS control plane에 PCI 디바이스가 있는 VM 배포 항목을 참조하십시오.
감시 타이머	VWDT(가상 감시 타이머) 디바이스를 추가하여 가상 시스템 내의 시스템 성능과 관련된 자립을 보장합니다. 가상 시스템에 가상 감시 타이머 디바이스를 추가하는 방법을 참조하십시오.
정밀 클럭	정밀 클럭 디바이스를 VM에 추가합니다. 정밀 클럭은 가상 시스템이 기본 ESXi 호스트의 시스템 시간에 액세스할 수 있도록 하는 가상 클럭 디바이스입니다. 가상 시스템에 정밀 클럭 디바이스 추가하는 방법을 참조하십시오.
직렬 포트	가상 직렬 포트를 물리적 직렬 포트 또는 호스트 컴퓨터의 파일에 연결하도록 구성합니다. 직렬 포트 구성 변경을 참조하십시오.
인스턴스 스토리지	VM에 인스턴스 스토리지를 구성합니다. VM은 영구 스토리지 볼륨과 함께 인스턴스 스토리지를 사용할 수 있습니다. VM과 별도로 존재하는 영구 볼륨과 달리 인스턴스 스토리지 볼륨은 VM 인스턴스의 수명 주기에 따라 달라집니다. **인스턴스 스토리지** 옵션을 사용하여 적절한 스토리지 정책을 추가하고 VM에서 사용할 볼륨을 구성할 수 있습니다. 추가 요구 사항은 vSphere IaaS control plane에 인스턴스 스토리지가 있는 VM 배포의 내용을 참조하십시오.
네트워크 어댑터	VM 클래스에 대한 네트워크 어댑터를 구성합니다. DevOps 사용자가 VM 클래스를 사용하여 VM을 배포할 때 어댑터에 대한 워크로드 네트워크를 지정할 수 있습니다. 워크로드 네트워크는 VM이 실행되는 vSphere 네임스페이스로 구성해야 합니다. 지원되는 어댑터 유형에 대한 자세한 내용은 네트워크 어댑터 기본 사항을 참조하십시오.

5 **VM 옵션** 페이지에서 VM 옵션을 설정하거나 변경하여 VMware Tools 스크립트를 실행하고, 원격 콘솔에 대한 사용자 액세스를 제어하고, 시작 동작을 구성하는 등의 작업을 수행합니다.

VM 클래스에 구성할 수 있는 VM 옵션에 대한 자세한 내용은 가상 시스템 옵션 구성을 참조하십시오.

6 **고급 매개 변수** 페이지에서 VMware 기술 지원 담당자의 안내가 있을 때 또는 VMware 설명서에 시스템 문제의 해결을 위해 매개 변수를 추가하거나 변경하라는 지침이 있을 때 VM 구성 매개 변수를 변경하거나 추가합니다.

VM 고급 매개 변수에 대한 자세한 내용은 가상 시스템 고급 파일 매개 변수 구성을 참조하십시오.

7 VM 클래스를 편집할 준비가 되면 변경 내용을 검토하고 확인한 후 **마침**을 클릭합니다.

vSphere Client를 사용하여 VM 클래스를 네임스페이스와 연결

vSphere 관리자는 기본 또는 사용자 지정 VM 클래스를 감독자에 있는 하나 이상의 네임스페이스에 추가합니다. VM 클래스를 네임스페이스에 추가할 때 DevOps 사용자가 클래스를 사용할 수 있도록 합니다. 그래야 Kubernetes 네임스페이스 환경에서 셀프 서비스 VM을 시작할 수 있습니다. 네임스페이스에 할당하는 VM 클래스는 Tanzu Kubernetes Grid 클러스터를 구성하는 VM에서도 사용됩니다.

여러 VM 클래스를 단일 네임스페이스에 추가할 수 있습니다. 다양한 VM 클래스는 다양한 서비스 수준의 지표로 사용됩니다. 여러 VM 클래스를 게시하는 경우 DevOps 사용자는 네임스페이스에서 가상 시스템을 생성하고 관리할 때 모든 사용자 지정 클래스와 기본 클래스 중에 선택할 수 있습니다.

참고 새로 생성된 네임스페이스에 Tanzu Kubernetes Grid 클러스터를 배포할 수 있으려면 DevOps 엔지니어에게 VM 클래스에 액세스할 수 있는 권한이 있어야 합니다. vSphere 관리자는 기본 또는 사용자 지정 VM 클래스를 Tanzu Kubernetes Grid 클러스터가 배포된 새 네임스페이스에 명시적으로 연결해야 합니다.

사전 요구 사항

필요한 권한:

- **네임스페이스.클러스터 전체 구성 수정**
- **네임스페이스.네임스페이스 구성 수정**
- **가상 시스템 클래스.가상 시스템 클래스 관리**

절차

1 vSphere Client에서 네임스페이스로 이동합니다.

   a vSphere Client 홈 메뉴에서 **워크로드 관리**를 선택합니다.

   b **네임스페이스** 탭을 클릭하고 네임스페이스를 클릭합니다.

2 VM 클래스를 추가합니다.

   a **VM 서비스** 창에서 **VM 클래스 추가**를 클릭합니다.

   b VM 클래스를 하나 또는 여러 개 선택하고 **확인**을 클릭합니다.

결과

추가한 VM 클래스는 셀프 서비스 VM에 대한 DevOps의 네임스페이스에서 사용할 수 있습니다. 이러한 클래스는 Tanzu Kubernetes Grid 클러스터를 구성하는 VM에서도 사용할 수 있습니다.

vSphere Client를 사용하여 네임스페이스에서 VM 클래스 관리

VM 클래스를 네임스페이스와 연결한 후, VM 클래스를 더 추가하거나 클래스를 제거하여 Kubernetes 네임스페이스에서 게시를 취소할 수 있습니다.

사전 요구 사항

- 네임스페이스에서 VM 클래스를 제거하려면 Tanzu Kubernetes Grid에 사용되지 않는지 확인합니다. 제거하면 Tanzu Kubernetes Grid 작업에 영향을 미칠 수 있습니다.
- 필요한 권한:
  - **네임스페이스.클러스터 전체 구성 수정**
  - **네임스페이스.네임스페이스 구성 수정**
  - **가상 시스템 클래스.가상 시스템 클래스 관리**

절차

1 vSphere Client에서 네임스페이스로 이동합니다.

a vSphere Client 홈 메뉴에서 **워크로드 관리**를 선택합니다.

b **네임스페이스** 탭을 클릭하고 네임스페이스를 클릭합니다.

2 VM 클래스를 추가하거나 제거합니다.

a **VM 서비스** 창에서 **VM 클래스 관리**를 클릭합니다.

b 다음 작업 중 하나를 수행합니다.

옵션	설명
VM 클래스 제거	VM 클래스를 선택 취소하고 **확인**을 클릭합니다.
VM 클래스 추가	VM 클래스를 하나 또는 여러 개 선택하고 **확인**을 클릭합니다.

데이터 센터 CLI를 사용하여 VM 클래스 생성 및 관리

vSphere Client 외에도 DCLI(Data Center CLI) 명령을 사용하여 VM 클래스를 생성하고 관리할 수 있습니다. DCLI 명령을 사용하면 vSphere Client에서 사용할 수 없는 VM 구성 옵션에 더 유연하게 액세스할 수 있습니다.

사전 요구 사항

루트 사용자 계정을 사용하여 vCenter Server에 로그인하고 `dcli +i`를 입력하여 대화형 모드에서 DCLI를 사용합니다.

DCLI 명령에 대한 자세한 내용은 DCLI 명령 실행 개요를 참조하십시오.

사용 가능한 DCLI 명령

명령	설명
`namespacemanagement virtualmachineclasses create`	VM 클래스 개체를 생성합니다.
`namespacemanagement virtualmachineclasses delete`	VM 클래스 개체를 삭제합니다.

명령	설명
`namespacemanagement virtualmachineclasses get`	VM 클래스에 대한 정보를 반환합니다.
`namespacemanagement virtualmachineclasses list`	모든 VM 클래스에 대한 정보를 반환합니다.
`namespacemanagement virtualmachineclasses update`	VM 클래스 개체의 구성을 업데이트합니다.

데이터 센터 CLI를 사용하여 VM 클래스 생성

vSphere 관리자는 DCLI `com vmware vcenter namespacemanagement virtualmachineclasses create` 명령을 사용하여 VM 클래스를 생성합니다. CPU, 메모리, 메모리 예약, 네트워크 어댑터 등과 같은 VM 속성을 구성할 수 있습니다.

이 명령은 다음과 같은 인수를 수행합니다.

인수	설명
-h, --help	이 도움말 메시지를 표시하고 종료합니다.
--config-spec CONFIG_SPEC	VM 클래스(json 입력)와 연결된 VirtualMachineConfigSpec입니다.
--cpu-count CPU_COUNT	필수. 이 클래스의 가상 시스템에 대해 구성된 CPU 수(정수)입니다.
--cpu-reservation CPU_RESERVATION	가상 시스템에 예약된 사용 가능한 총 CPU의 백분율(정수)입니다.
--description DESCRIPTION	VM 클래스에 대한 설명(문자열)입니다.
--devices-dynamic-direct-path-io-devices DEVICES_DYNAMIC_DIRECT_PATH_IO_DEVICES	동적 DirectPath I/O 디바이스 목록(json 입력)입니다.
--devices-vgpu-devices DEVICES_VGPU_DEVICES	vGPU 디바이스 목록(json 입력)입니다.
--id ID	필수. 가상 시스템 클래스의 식별자(문자열)입니다.
--instance-storage-policy INSTANCE_STORAGE_POLICY	인스턴스 스토리지에 해당하는 스토리지 정책입니다(문자열).
--instance-storage-volumes INSTANCE_STORAGE_VOLUMES	인스턴스 스토리지 볼륨 목록(json 입력)입니다.
--memory-mb MEMORY_MB	필수. 이 클래스의 가상 시스템에 대해 구성된 메모리 양(MB)입니다(정수).
--memory-reservation MEMORY_RESERVATION	이 클래스의 가상 시스템에 예약된 사용 가능한 메모리의 백분율입니다.

다음 예를 사용하여 다양한 속성으로 VM 클래스를 생성합니다.

CPU 및 메모리

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

구성 규격에서 `numCPUs` 및 `memoryMB`를 설정하는 것은 선택 사항입니다. 설정하도록 선택한 경우 필수 `--cpu-count` 및 `--memory-mb` vAPI 필드와 동일한 값을 가져야 합니다.

CPU 및 메모리 예약

CPU 및 메모리 예약이 있는 구성 규격을 사용하여 VM 클래스를 생성하는 경우 메모리 예약 또는 제한은 `memoryAllocation`의 경우 MB 단위, `cpuAllocation`의 경우 MHz 단위입니다.

```
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
```

네트워크 어댑터

다음 명령은 E1000 유형의 네트워크 어댑터를 생성합니다.

```
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

이러한 예에서 첫 번째 명령은 필드 `--devices-vgpu-devices`를 사용하여 vGPU가 포함된 VM 클래스를 생성합니다. 두 번째 명령은 configSpec을 사용하여 vGPU가 포함된 VM 클래스를 생성합니다.

```
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
```

인스턴스 스토리지

다음 예에서는 `--instance-storage-volumes` 및 `--instance-storage-policy` 필드를 사용하여 인스턴스 스토리지를 사용하는 VM 클래스를 생성합니다.

```
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
```

이 예의 ID 필드는 VM 서비스 VM의 인스턴스 스토리지 디바이스를 나타내는 잘 알려진 virtualDisk ID입니다.

```
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"}}]}]}'
```

데이터 센터 CLI를 사용하여 VM 클래스 업데이트

vSphere관리자는 DCLI `com vmware vcenter namespacemanagement virtualmachineclasses update` 명령을 사용하여 VM 클래스를 수정합니다.

다음을 예로 사용할 수 있습니다.

CPU 및 메모리 수정

```
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 및 메모리 예약 수정

CPU 및 메모리 예약이 있는 구성 규격을 사용하여 VM 클래스를 생성하는 경우 메모리 예약 또는 제한은 `memoryAllocation`의 경우 MB 단위, `cpuAllocation`의 경우 MHz 단위입니다.

```
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
```

구성 규격을 사용하여 CPU 및 메모리 예약을 업데이트할 수도 있습니다. 기존 CPU 또는 메모리 예약을 덮어씁니다. 다음의 예와 같이 사용할 수 있습니다.

```
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 추가

```
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 제거

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

인스턴스 스토리지 추가

```
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
```

인스턴스 스토리지 제거

```
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
```

네트워크 어댑터 추가

이 명령은 인스턴스 스토리지와 e1000 NIC를 모두 VM 클래스에 추가합니다.

```
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
```

구성 규격 비우기

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

후속 작업

DCLI를 사용하여 생성한 VM 클래스는 vCenter Server에서 사용할 수 있게 됩니다. vSphere Client를 사용하여 이러한 VM 클래스를 네임스페이스에 할당할 수 있습니다. vSphere Client를 사용하여 VM 클래스를 네임스페이스와 연결의 내용을 참조하십시오.

vSphere IaaS control plane에서 독립형 VM 배포

DevOps 엔지니어는 `kubectl` 명령을 사용하여 사용 가능한 VM 리소스를 검토하고 감독자의 네임스페이스에 독립형 Linux 또는 Windows VM을 프로비저닝합니다. VM에 vGPU용으로 구성된 PCI 디바이스가 포함된 경우 vSphere IaaS control plane 환경에서 VM을 생성하고 부팅한 후에 NVIDIA vGPU 그래픽 드라이버를 설치하여 GPU 작업이 사용되도록 설정할 수 있습니다.

사전 요구 사항

vSphere IaaS control plane에서 독립형 VM을 배포하려면 DevOps 엔지니어에게 특정 VM 리소스에 대한 액세스 권한이 있어야 합니다. vSphere 관리자가 VM 리소스를 사용할 수 있도록 다음 단계를 수행했는지 확인합니다.

- 네임스페이스를 생성하고 여기에 스토리지 정책을 할당합니다. 감독자에서 vSphere 네임스페이스 생성 및 구성의 내용을 참조하십시오.

- 컨텐츠 라이브러리를 생성하고 이를 네임스페이스와 연결합니다. vSphere IaaS control plane에서 독립형 VM에 대한 컨텐츠 라이브러리 생성 및 관리의 내용을 참조하십시오.
  - 컨텐츠 라이브러리가 보안 정책에 의해 보호되는 경우 모든 라이브러리 항목은 해당 정책을 준수해야 합니다. 보호된 라이브러리에 준수 및 비준수 항목이 혼합되어 있는 경우 `kubectl get virtualmachineimages` 명령이 DevOps 엔지니어에게 VM 이미지를 표시하지 못합니다.
  - vGPU 디바이스를 사용하여 VM을 배포하려는 경우 CentOS와 같이 부팅 모드가 EFI로 설정된 이미지에 액세스할 수 있어야 합니다.
- 기본 또는 사용자 지정 클래스를 네임스페이스와 연결했습니다. vSphere IaaS control plane의 VM 클래스 작업의 내용을 참조하십시오.

VM에 NVIDIA vGPU 또는 기타 PCI 디바이스를 사용하려는 경우 추가 요구 사항을 따라야 합니다. 자세한 내용은 vSphere IaaS control plane에 PCI 디바이스가 있는 VM 배포 항목을 참조하십시오.

VM Operator 및 지원되는 필드에 대한 자세한 내용은 VM 서비스의 개념 및 https://vm-operator.readthedocs.io/en/stable/ref/api/v1alpha2/ 항목을 참조하십시오.

vSphere IaaS control plane의 네임스페이스에서 사용 가능한 VM 리소스 보기

DevOps 엔지니어는 사용자가 네임스페이스의 VM 리소스에 액세스하고 환경에서 사용할 수 있는 VM 클래스 및 VM 템플릿을 볼 수 있는지 확인합니다. VM을 셀프 서비스하는 데 필요할 수 있는 스토리지 클래스 및 기타 항목을 나열할 수도 있습니다.

이 작업에서는 독립형 VM 배포에 사용 가능한 리소스에 액세스하는 데 사용하는 명령을 설명합니다. Tanzu Kubernetes Grid 클러스터 및 클러스터를 구성하는 VM을 배포하는 데 필요한 리소스에 대한 자세한 내용은 "vSphere IaaS 제어부에서 TKG 서비스 사용" 설명서의 TKG 클러스터에 대한 가상 시스템 클래스를 참조하십시오.

절차

1 Kubernetes 환경의 네임스페이스에 액세스합니다.

vSphere IaaS control plane에서 감독자 컨텍스트 가져오기 및 사용의 내용을 참조하십시오.

2 네임스페이스에서 사용 가능한 VM 클래스를 보려면 다음 명령을 실행합니다.

```
kubectl get virtualmachineclass
```

다음 출력을 볼 수 있습니다.

참고 사용 시도 VM 클래스 유형을 사용하면 리소스가 오버 커밋될 수 있으므로 VM을 프로비저닝하는 네임스페이스에 대한 제한을 설정한 경우 리소스가 부족해질 수 있습니다. 따라서 운영 환경에서는 보장됨 VM 클래스 유형을 사용하십시오.

```
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 특정 VM 클래스에 대한 세부 정보를 보려면 다음 명령을 실행합니다.

- `kubectl describe virtualmachineclasses` *name_vm_class*

  VM 클래스에 vGPU 디바이스가 포함된 경우 `spec: hardware: devices: vgpuDevices`에서 해당 프로파일을 볼 수 있습니다.

```
......
spec:
  hardware:
    cpus: 4
    devices:
      vgpuDevices:
      - profileName: grid_v100-q4
......
```

- `kubectl get virtualmachineclasses -o wide`

  VM 클래스에 vGPU 또는 패스스루 디바이스가 포함된 경우 이러한 디바이스는 출력의 `VGPUDevicesProfileNames` 또는 `PassthroughDeviceIDs` 열에 표시됩니다.

4 VM 이미지를 봅니다.

`kubectl get virtualmachineimages`

표시되는 출력은 다음과 유사합니다. `vmi-xxxxxxxxxxxxx` 같은 이미지 이름은 시스템에서 자동으로 생성됩니다.

```
NAME                                              VERSION  OSTYPE                FORMAT
IMAGESUPPORTED  AGE
vmi-xxxxxxxxxxxxx                                          centos8_64Guest       ovf
true            4d3h
```

5 특정 이미지를 설명하려면 다음 명령을 사용합니다.

`kubectl describe virtualmachineimage vmi-xxxxxxxxxxxxx`

vGPU 디바이스가 있는 VM에는 부팅 모드가 EFI로 설정된 이미지(예: CentOS)가 필요합니다. 이러한 이미지에 액세스할 수 있는지 확인합니다.

6 스토리지 클래스에 액세스할 수 있는지 확인합니다.

```
kubectl get resourcequotas
```

자세한 내용은 vSphere 네임스페이스에서 스토리지 클래스 표시의 내용을 참조하십시오.

```
NAME                          AGE
REQUEST
     LIMIT
my-ns-ubuntu-storagequota    24h   wcpglobal-storage-profile.storageclass.storage.k8s.io/
requests.storage: 0/9223372036854775807
```

vSphere IaaS control plane에서 가상 시스템 배포

DevOps 엔지니어는 Kubernetes YAML 파일에 VM 배포 규격을 작성하여 선언적 방식으로 VM 및 해당 게스트 운영 체제를 프로비저닝할 수 있습니다.

사전 요구 사항

VM용 NVIDIA vGPU 또는 기타 PCI 디바이스를 사용하는 경우 vSphere IaaS control plane에 PCI 디바이스가 있는 VM 배포 항목을 참조하십시오.

절차

1 VM YAML 파일을 준비합니다.

파일에서 다음 매개 변수를 지정합니다.

옵션	설명
apiVersion	VM 서비스 API의 버전을 지정합니다. 예: **vmoperator.vmware.com/v1alpha2**.
kind	생성할 Kubernetes 리소스의 유형을 지정합니다. 사용 가능한 유일한 값은 **VirtualMachine**입니다.
spec.imageName	Kubernetes 클러스터에서 가상 시스템 이미지 리소스 이름을 지정합니다.
spec.storageClass	영구 볼륨의 스토리지에 사용할 스토리지 클래스를 지정합니다.
spec.className	사용할 가상 하드웨어 설정을 설명하는 VM 클래스의 이름을 지정합니다.
spec.networkInterfaces	VM에 대한 네트워크 관련 설정을 지정합니다. ■ networkType. 이 키의 값은 **nsx-t** 또는 **vsphere-distributed**일 수 있습니다. ■ networkName. 필요한 경우 이름을 지정하거나 기본 이름을 그대로 둡니다.

옵션	설명
spec.vmMetadata	VM에 전달할 추가 메타데이터를 포함합니다. 이 키를 사용하여 게스트 운영 체제 이미지를 사용자 지정하고 VM의 `hostname` 및 `user-data`(암호, ssh 키 포함) 등의 항목을 설정할 수 있습니다. Microsoft 시스템 준비 도구(Sysprep)를 사용하여 Windows VM을 부트스트랩하고 사용자 지정하는 방법에 대한 세부 정보를 포함한 자세한 내용은 게스트 사용자 지정을 참조하십시오.
topology.kubernetes.io/zone	3개 영역 감독자에서 VM 배치를 제어합니다. 예: `topology.kubernetes.io/zone: zone-a02`

다음 예제 VM YAML 파일 `my-vm`은 CloudInit를 부트스트랩 방법으로 사용합니다. 이 예에서는 비밀 리소스 `my-vm-bootstrap-data`에서 사용자 데이터를 지정하는 VirtualMachine 리소스를 보여줍니다. 비밀은 게스트 운영 체제를 부트스트랩하고 사용자 지정하는 데 사용됩니다.

암호의 데이터에는 CloudInit cloud-config가 포함됩니다. cloud-config 형식에 대한 자세한 내용은 Cloud config 예시 공식 설명서를 참조하십시오.

Sysprep을 부트스트랩 방법으로 사용하는 예시는 Sysprep을 참조하십시오.

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

영역이 있는 환경에 VM을 배포하는 경우 다음 예를 사용합니다.

*ZONE_NAME*의 값을 구하려면 `kubectl get vspherezones` 명령을 실행합니다.

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

2 VM을 배포합니다.

```
kubectl apply -f my-vm.yaml
```

3 VM이 생성되었는지 확인합니다.

```
kubectl get vm
NAME              AGE
my-vm             28s
```

4 VM 세부 정보 및 해당 상태를 확인합니다.

```
kubectl describe virtualmachine my-vm
```

출력은 다음과 유사합니다. 출력에서 VM의 IP 주소를 얻을 수도 있으며, 이 주소는 `Vm Ip` 필드에 표시됩니다.

```
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 VM IP에 연결할 수 있는지 확인합니다.

```
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
```

결과

VM 서비스를 통해 생성된 VM은 Kubernetes 네임스페이스의 DevOps로만 관리할 수 있습니다. 수명 주기는 vSphere Client에서 관리할 수 없지만 vSphere 관리자는 VM 및 해당 리소스를 모니터링할 수 있습니다. 자세한 내용은 vSphere IaaS control plane에서 사용 가능한 가상 시스템 모니터링의 내용을 참조하십시오.

다음에 수행할 작업

자세한 내용은 가상 시스템 프로비저닝 소개 블로그를 참조하십시오.

vSphere IaaS control plane에서 vGPU 및 기타 PCI 디바이스를 사용하여 VM 배포

vSphere IaaS control plane 환경의 ESXi 호스트에 하나 이상의 NVIDIA GRID GPU 그래픽 디바이스가 있는 경우 NVIDIA GRID vGPU(가상 GPU) 기술을 사용하도록 VM을 구성할 수 있습니다. ESXi 호스트의 다른 PCI 디바이스를 패스스루 모드의 VM에서 사용할 수 있도록 구성할 수도 있습니다.

vSphere IaaS control plane에 vGPU가 있는 VM 배포

NVIDIA GRID GPU 그래픽 디바이스는 CPU에 과부하를 주지 않은 상태로 복잡한 그래픽 작업을 고성능으로 실행 및 최적화하기 위해 설계되었습니다. NVIDIA GRID vGPU는 여러 VM 간에 물리적 단일 GPU를 별도의 vGPU 지원 패스스루 디바이스로 공유하여 뛰어난 그래픽 성능, 비용 효율성 및 확장성을 제공합니다.

고려 사항

NVIDIA vGPU를 사용하는 경우 다음 고려 사항이 적용됩니다.

- 3개 영역 감독자는 vGPU가 있는VM을 지원하지 않습니다.

- VM 서비스에서 관리되는 vGPU 디바이스가 있는 VM은 ESXi 호스트가 유지 보수 모드로 전환되면 자동으로 전원이 꺼집니다. 그러면 VM에서 실행되는 워크로드에 일시적으로 영향을 줄 수 있습니다. 호스트가 유지 보수 모드를 종료하면 VM의 전원이 자동으로 켜집니다.
- DRS는 클러스터의 호스트 전반에 너비 우선 방식으로 vGPU VM을 분산합니다. 자세한 내용은 "vSphere 리소스 관리" 가이드에서 vGPU VM의 DRS 배치를 참조하십시오.

요구 사항

NVIDIA vGPU를 구성하려면 다음 요구 사항을 따릅니다.

- VMware 호환성 가이드에서 ESXi가 지원되는지 확인하고 벤더에 문의하여 호스트가 전원 및 구성 요구 사항을 충족하는지 확인합니다.
- **Shared Direct** 모드에서 하나 이상의 디바이스를 사용하여 ESXi 호스트 그래픽 설정을 구성합니다. "vSphere 리소스 관리" 설명서에서 호스트 그래픽 구성을 참조하십시오.
- vGPU 디바이스가 있는 VM에 사용하는 컨텐츠 라이브러리에는 부팅 모드가 EFI로 설정된 이미지(예: CentOS)가 포함되어야 합니다.
- NVIDIA vGPU 소프트웨어를 설치합니다. NVIDIA는 다음 구성 요소를 포함하는 vGPU 소프트웨어 패키지를 제공합니다.

  자세한 내용은 해당 NVIDIA 가상 GPU 소프트웨어 설명서를 참조하십시오.

  - vGPU Manager - vSphere 관리자가 ESXi 호스트에 설치합니다. VMware 기술 자료 문서 2033434를 참조하십시오.
  - 게스트 VM 드라이버 - VM을 배포하고 부팅한 후 DevOps 엔지니어가 VM에 설치합니다. vSphere IaaS control plane의 VM에 NVIDIA 게스트 드라이버 설치의 내용을 참조하십시오.

vSphere Client를 사용하여 VM 클래스에 vGPU 디바이스 추가

기존 VM 클래스를 생성하거나 편집하여 NVIDIA GRID vGPU(가상 GPU)를 추가합니다.

사전 요구 사항

필요한 권한:

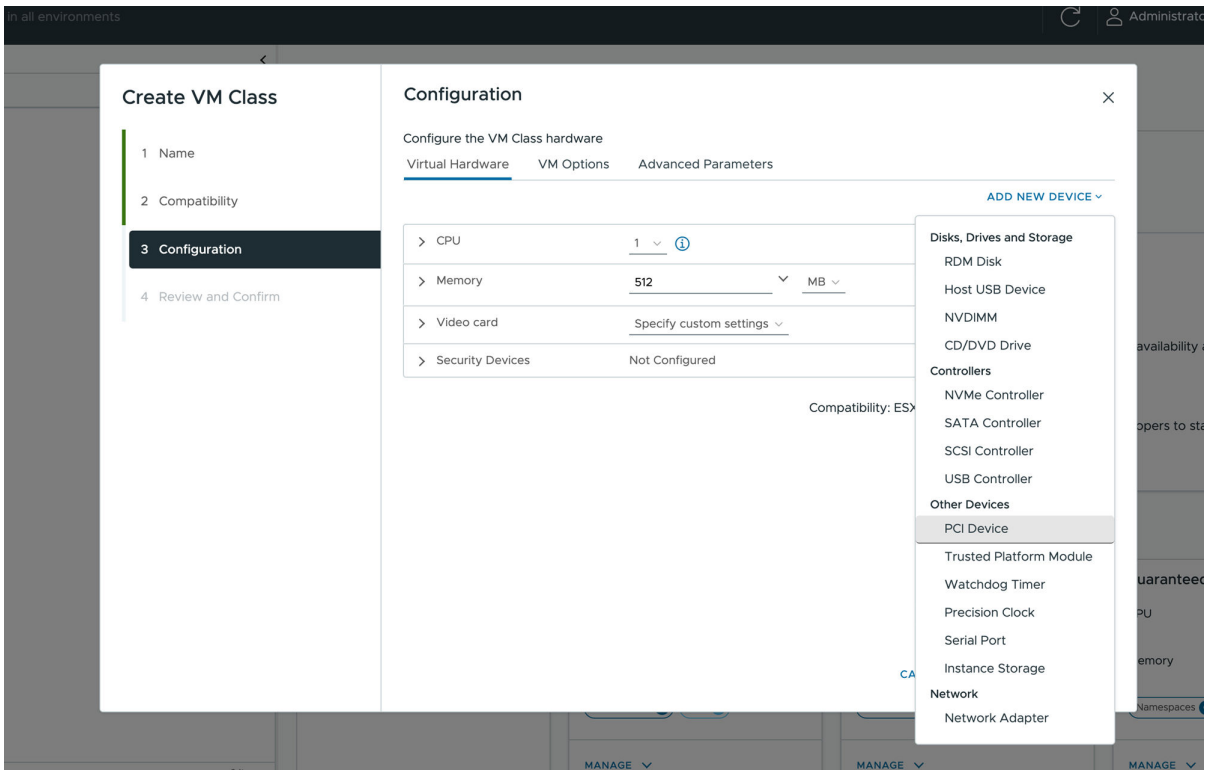
- **네임스페이스.클러스터 전체 구성 수정**
- **네임스페이스.네임스페이스 구성 수정**
- **가상 시스템 클래스.가상 시스템 클래스 관리**

절차

1 기존 VM 클래스를 생성하거나 편집합니다.

옵션	작업
새 VM 클래스 생성	a vSphere Client 홈 메뉴에서 **워크로드 관리**를 선택합니다. b **서비스** 탭을 클릭하고 **VM 서비스** 창에서 **관리**를 클릭합니다. c **VM 서비스** 페이지에서 **VM 클래스**를 클릭하고 **VM 클래스 생성**을 클릭합니다. d 표시되는 메시지를 따릅니다.
VM 클래스 편집	a vSphere Client 홈 메뉴에서 **워크로드 관리**를 선택합니다. b **서비스** 탭을 클릭하고 **VM 서비스** 창에서 **관리**를 클릭합니다. c **VM 서비스** 페이지에서 **VM 클래스**를 클릭합니다. d 기존 VM 클래스 창에서 **관리**를 클릭하고 **편집**을 클릭합니다. e 표시되는 메시지를 따릅니다.

2 **구성** 페이지에서 **가상 하드웨어** 탭을 클릭하고 **새 디바이스 추가**를 클릭한 후 **PCI 디바이스**를 선택합니다.

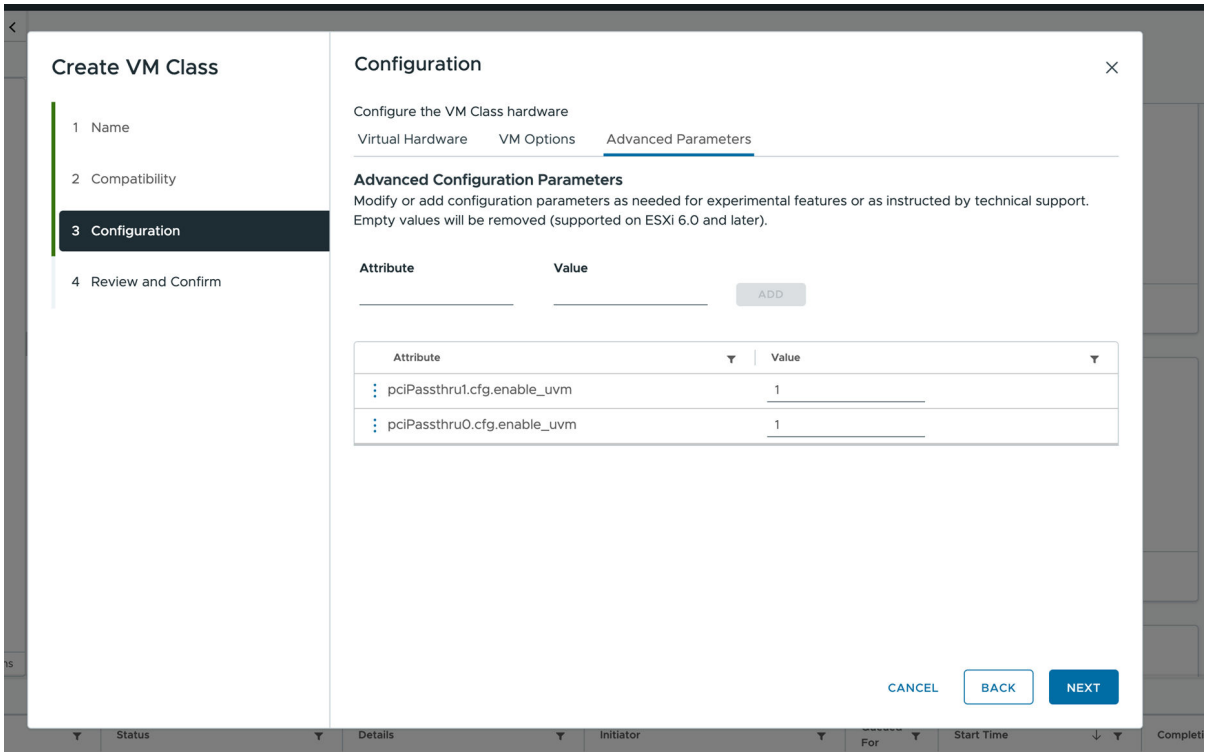


3 **디바이스 선택** 페이지의 사용 가능한 디바이스 목록에서 NVIDIA GRID vGPU를 선택하고 **선택**을 클릭합니다.

디바이스가 가상 하드웨어 페이지에 나타납니다.

4 **고급 매개 변수** 탭을 클릭하고 다음 특성 및 값으로 매개 변수를 설정합니다.

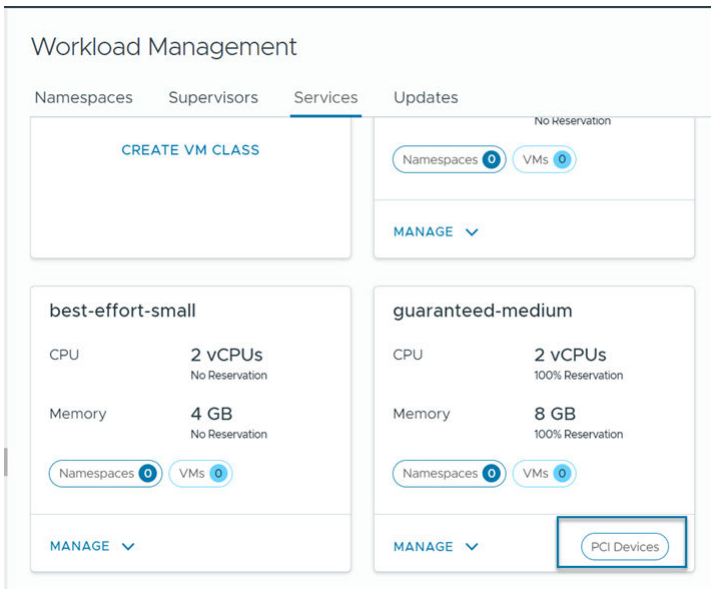
옵션	설명
매개 변수	값
pciPassthru0.cfg.enable_uvm	1
pciPassthru1.cfg.enable_uvm	1



5 구성을 검토하고 **마침**을 클릭합니다.

결과

VM 클래스 창의 **PCI 디바이스** 태그는 VM 클래스가 vGPU 지원임을 나타냅니다.



데이터 센터 CLI를 사용하여 VM 클래스에 vGPU 디바이스 추가

vSphere Client 외에도 DCLI(데이터 센터 CLI) 명령을 사용하여 vGPU 및 고급 구성을 추가할 수 있습니다.

DCLI 명령에 대한 자세한 내용은 데이터 센터 CLI를 사용하여 VM 클래스 생성 및 관리 항목을 참조하십시오.

절차

1 루트 사용자 계정을 사용하여 vCenter Server에 로그인하고 `dcli +i`를 입력하여 대화형 모드에서 DCLI를 사용합니다.

2 다음 명령을 실행하여 VM 클래스를 생성합니다.

다음 예에서 "my-class" VM 클래스에는 CPU 2개, 메모리 2048MB, 샘플 vGPU 프로파일 2개(`mockup-vmiop-8c` 및 `mockup-vmiop`)가 포함된 VirtualMachineConfigSpec이 포함됩니다. extraConfig 필드 `pciPassthru0.cfg.enable_uvm` 및 `pciPassthru1.cfg.enable_uvm`은 1로 설정됩니다.

```
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 plane의 VM에 NVIDIA 게스트 드라이버 설치

VM에 vGPU용으로 구성된 PCI 디바이스가 포함된 경우 vSphere IaaS control plane 환경에서 VM을 생성하고 부팅한 후에 NVIDIA vGPU 그래픽 드라이버를 설치하여 GPU 작업이 완전하게 사용되도록 설정합니다.

사전 요구 사항

- vGPU를 사용하여 VM을 배포합니다. VM YAML 파일이 vGPU 정의가 있는 VM 클래스를 참조하는지 확인합니다. vSphere IaaS control plane에서 가상 시스템 배포의 내용을 참조하십시오.
- NVIDIA 다운로드 사이트에서 vGPU 소프트웨어 패키지를 다운로드하고 패키지 압축을 풀고 게스트 드라이브 구성 요소를 준비했는지 확인합니다. 자세한 내용은 해당 NVIDIA 가상 GPU 소프트웨어 설명서를 참조하십시오.

  참고 드라이버 구성 요소의 버전은 vSphere 관리자가 ESXi 호스트에 설치한 vGPU Manager의 버전과 일치해야 합니다.

절차

1. NVIDIA vGPU 소프트웨어 Linux 드라이버 패키지(예: `NVIDIA-Linux-x86_64-version-grid.run`)를 게스트 VM에 복사합니다.
2. 드라이버 설치 관리자를 실행하기 전에 모든 애플리케이션을 종료합니다.
3. NVIDIA vGPU 드라이버 설치 관리자를 시작합니다.

   ```
   sudo ./NVIDIA-Linux-x86_64-version-grid.run
   ```

4. NVIDIA 소프트웨어 라이센스 계약에 동의하고 **예**를 선택하여 X 구성 설정을 자동으로 업데이트합니다.
5. 드라이버가 설치되었는지 확인합니다.

   예를 들면 다음과 같습니다.

   ```
   ~$ 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 plane에 PCI 디바이스가 있는 VM 배포

vGPU 외에도 패스스루 모드의 VM에서 사용할 수 있도록 ESXi 호스트에서 다른 PCI 디바이스를 구성할 수 있습니다.

vSphere IaaS control plane는 동적 DirectPath I/O 디바이스를 지원합니다. 동적 DirectPath I/O를 사용하면 VM이 호스트에 연결된 물리적 PCI 및 PCIe 디바이스에 직접 액세스할 수 있습니다. 동적 DirectPath I/O를 사용하여 VM에 여러 개의 PCI 패스스루 디바이스를 할당할 수 있습니다. 각 패스스루 디바이스는 해당 PCI 벤더 및 디바이스 식별자로 지정할 수 있습니다.

참고 PCI 패스스루 디바이스에 대한 동적 DirectPath I/O를 구성할 때 PCI 디바이스를 호스트에 연결하고 패스스루에 사용 가능한 것으로 표시합니다. "vSphere 네트워킹" 설명서에서 호스트의 네트워크 디바이스에 대한 패스스루 사용을 참조하십시오.

vSphere IaaS control plane에 인스턴스 스토리지가 있는 VM 배포

VM은 영구 스토리지 볼륨과 함께 인스턴스 스토리지를 사용할 수 있습니다. VM과 별도로 존재하는 영구 볼륨과 달리 인스턴스 스토리지 볼륨은 VM 인스턴스의 수명 주기에 따라 달라집니다. 이 스토리지는 일반적으로 ESXi 호스트에 로컬인 NVMe와 같은 고속 디바이스에 있습니다.

인스턴스 스토리지 수명 주기

VM 생성 시 시스템은 인스턴스 스토리지 볼륨을 생성하고 VM에 연결합니다. 인스턴스 스토리지 볼륨의 데이터는 연결된 해당 VM 인스턴스의 수명 동안에만 유지됩니다. 볼륨은 VM이 삭제될 때 삭제됩니다.

인스턴스 스토리지가 있는 VM은 ESXi.호스트 유지 보수 모드를 지원합니다. VM은 ESXi 호스트가 유지 보수 모드로 전환되면 꺼지고 호스트가 유지 보수 모드를 종료하면 켜집니다.

인스턴스 스토리지 VM 고려 사항

인스턴스 스토리지가 있는 VM을 사용하는 경우 다음 사항을 고려하십시오.

- VDS 네트워킹 스택이 있는 감독자는 인스턴스 스토리지를 지원하지 않습니다.
- 3개 영역 감독자는 인스턴스 스토리지를 지원하지 않습니다.
- vSphere 관리자가 인스턴스 스토리지에 필요한 적절한 스토리지 정책이 누락된 네임스페이스에 인스턴스 스토리지가 있는 VM 클래스를 적용하면 주의가 나타납니다.
- 인스턴스 볼륨이 있는 VM은 다른 ESXi 호스트로 마이그레이션할 수 없습니다.
- 볼륨이 이미 사용 중인 경우에는 인스턴스 스토리지 볼륨을 편집할 수 없습니다.

- vSphere 관리자가 VM 생성 후 네임스페이스에서 인스턴스 스토리지 정책을 제거하면 VM이 계속 실행됩니다.
- DevOps 엔지니어는 인스턴스 스토리지 리소스를 삭제하거나 업데이트할 수 없습니다. 인스턴스 스토리지 볼륨을 한 VM 인스턴스에서 분리하여 다른 인스턴스에 연결할 수 없습니다.

인스턴스 스토리지 VM 프로비저닝 및 모니터링 워크플로

단계	수행자	설명
1	vSphere 관리자	vSphere IaaS control plane에서 독립형 VM에 대한 컨텐츠 라이브러리 생성 및 관리
2	vSphere 관리자	vSAN Direct 데이터스토어를 생성합니다.
3	vSphere 관리자	vSAN Direct와 호환되는 스토리지 정책을 생성하고 네임스페이스에 할당합니다.
4	vSphere 관리자	인스턴스 스토리지 VM 클래스를 생성하고 네임스페이스에 할당합니다.
5	DevOps 엔지니어	네임스페이스에 인스턴스 스토리지가 있는 VM을 프로비저닝합니다.
6	vSphere 관리자	vSphere IaaS control plane에서 사용 가능한 가상 시스템 모니터링

vSAN Direct 데이터스토어 생성

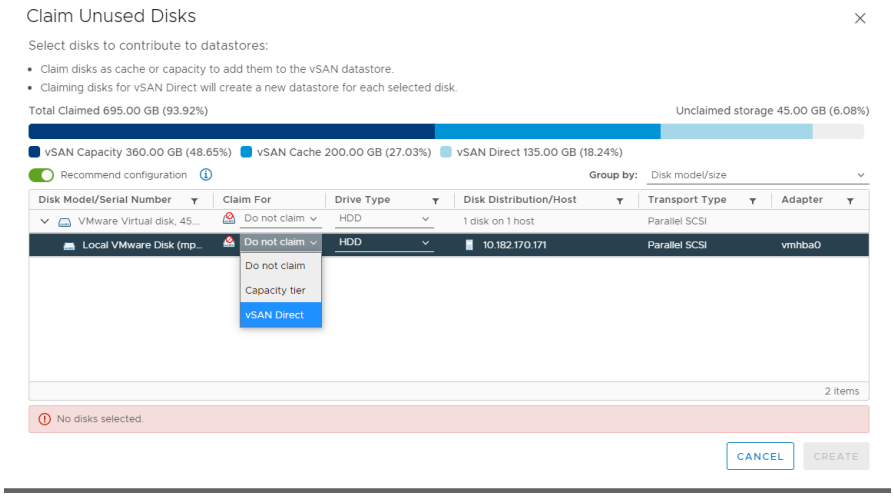
vSphere 관리자는 vSAN 데이터 지속성 플랫폼 또는 VM 인스턴스 스토리지와 같은 기능과 함께 사용할 vSAN Direct 데이터스토어를 설정합니다. 데이터스토어를 생성하려면 ESXi 호스트에 로컬인 할당되지 않은 스토리지 디바이스를 사용합니다.

감독자에 대해 vSAN을 사용하도록 설정할 때 vSAN Direct 데이터스토어를 생성할 수 있습니다. 다음 작업은 클러스터에서 vSAN이 이미 사용되도록 설정된 경우 로컬 스토리지 디바이스를 vSAN Direct로 할당하는 방법을 보여 줍니다.

절차

1. vSphere Client에서 vSAN 클러스터로 이동합니다.
2. **구성** 탭을 클릭합니다.
3. vSAN에서 **디스크 관리**를 클릭합니다.
4. **사용되지 않는 디스크 할당**을 클릭합니다.
5. **사용되지 않는 디스크 할당** 대화 상자에서 **vSAN Direct**를 클릭합니다.
6. 할당할 디바이스를 선택하고 **vSAN Direct에 대해 할당** 열의 확인란을 선택합니다.

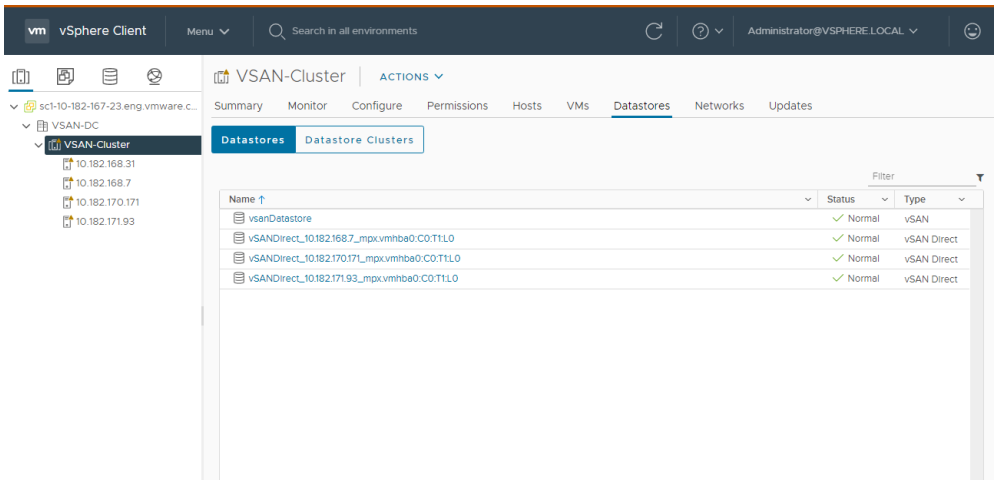
참고 일반 vSAN 데이터스토어에 대한 디바이스를 할당한 경우 해당 디바이스는 **vSAN Direct** 탭에 나타나지 않습니다.



7 **생성**을 클릭합니다.

할당하는 각 디바이스에 대해 vSAN Direct는 새 데이터스토어를 생성합니다.

8 **데이터스토어** 탭을 클릭하여 클러스터의 모든 vSAN Direct 데이터스토어를 표시합니다.



다음에 수행할 작업

vSAN Direct를 외부 스토리지와 함께 사용할 수 있습니다. 자세한 내용은 "vSphere IaaS 제어부 유지 보수" 설명서에서 vSAN Direct에서 외부 스토리지 사용을 참조하십시오.

vSAN Direct 스토리지 정책 생성

vSAN Direct를 사용하는 경우 감독자 네임스페이스에서 사용할 스토리지 정책을 생성합니다. 이 스토리지 정책과 연결하는 네임스페이스에서 vSAN Direct와 호환되는 워크로드(예: 상태 저장 서비스 또는 인스턴스 스토리지 VM)를 실행할 수 있습니다.

절차

1 vSphere Client에서 **VM 스토리지 정책 생성** 마법사를 엽니다.

   a **홈** 메뉴에서 **정책 및 프로파일**을 클릭합니다.

   b **정책 및 프로파일**에서 **VM 스토리지 정책**을 클릭합니다.

   c **생성**을 클릭합니다.

2 정책 이름 및 설명을 입력합니다.

옵션	작업
vCenter Server	vCenter Server 인스턴스를 선택합니다.
이름	스토리지 정책의 이름을 입력합니다.
설명	스토리지 정책의 설명을 입력합니다.

3 **정책 구조** 페이지의 **데이터스토어별 규칙**에서 vSAN Direct 스토리지 배치에 대한 규칙을 사용하도록 설정합니다.

4 **vSAN Direct 규칙** 페이지에서 스토리지 배치 유형으로 vSAN Direct를 지정합니다.

5 **스토리지 호환성** 페이지에서 이 정책과 일치하는 vSAN Direct 데이터스토어 목록을 검토합니다.

6 **검토 및 완료** 페이지에서 스토리지 정책 설정을 검토하고 **마침**을 클릭합니다.

   설정을 변경하려면 **뒤로**를 클릭하여 해당 페이지로 이동합니다.

인스턴스 스토리지가 포함된 VM 클래스 생성

VM 클래스에서 vSAN Direct 스토리지 정책을 참조하고 인스턴스 스토리지에 사용할 볼륨의 크기를 설정합니다. VM 클래스를 생성한 후 이를 인스턴스 스토리지 VM에 사용할 네임스페이스에 할당합니다.

사전 요구 사항

- vSAN Direct 데이터스토어와 호환되는 스토리지 정책을 생성합니다.
- 인스턴스 스토리지 VM에 사용하는 네임스페이스에 vSAN Direct 스토리지 정책을 추가합니다. 감독자에서 vSphere 네임스페이스 생성 및 구성의 내용을 참조하십시오.
- 필요한 권한:
  - **네임스페이스.클러스터 전체 구성 수정**
  - **네임스페이스.네임스페이스 구성 수정**
  - **가상 시스템 클래스.가상 시스템 클래스 관리**

절차

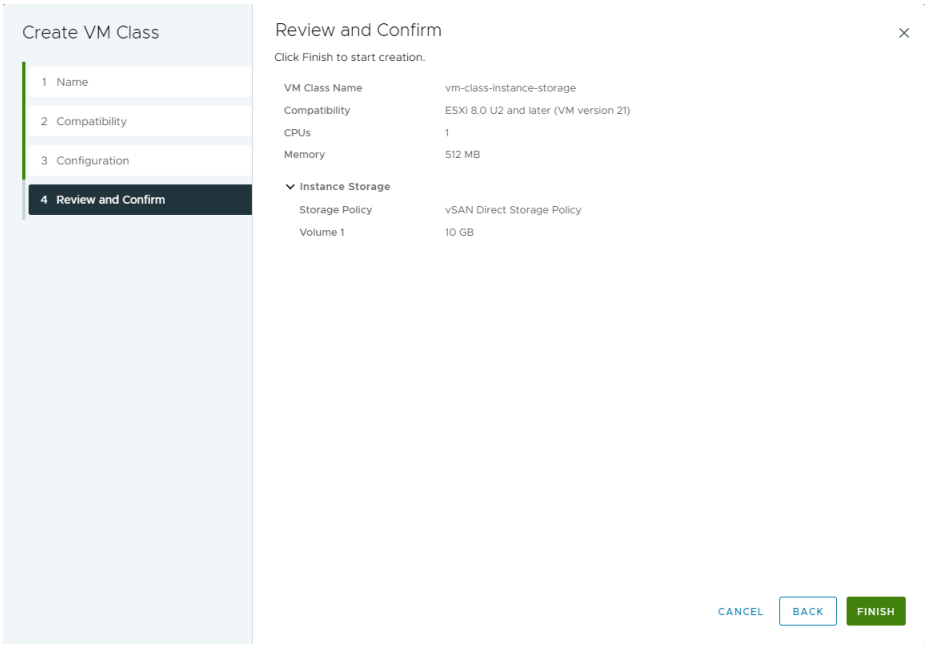
1 VM 클래스를 생성하거나 편집할 때 인스턴스 스토리지를 추가합니다.

옵션	작업
VM 클래스 생성	a vSphere Client 홈 메뉴에서 **워크로드 관리**를 선택합니다. b **서비스** 탭을 클릭하고 **VM 서비스** 카드에서 **관리**를 클릭합니다. c **VM 서비스** 페이지에서 **VM 클래스 생성**을 클릭합니다. d 필요에 따라 VM 클래스를 구성합니다. 사용 가능한 옵션은 vSphere Client를 사용하여 VM 클래스 편집 항목을 참조하십시오. e 인스턴스 스토리지를 추가하려면 **구성** 페이지에서 **가상 하드웨어**를 선택한 다음 **새 디바이스 추가 > 인스턴스 스토리지**를 선택합니다. **인스턴스 스토리지** 옵션이 가상 하드웨어에 나타납니다.
기존 VM 클래스 편집	a vSphere Client 홈 메뉴에서 **워크로드 관리**를 선택합니다. b **서비스** 탭을 클릭하고 **VM 서비스** 창에서 **관리**를 클릭합니다. c **VM 서비스** 페이지에서 **VM 클래스**를 클릭합니다. d 기존 VM 클래스 카드에서 **관리**를 클릭하고 **편집**을 클릭합니다. e 인스턴스 스토리지를 추가하려면 **가상 하드웨어**를 선택한 다음 **새 디바이스 추가 > 인스턴스 스토리지**를 선택합니다. **인스턴스 스토리지** 옵션이 가상 하드웨어에 나타납니다.

2 **인스턴스 스토리지** 옵션을 확장하여 인스턴스 스토리지 설정을 편집합니다.

옵션	작업
스토리지 정책	vSAN Direct 스토리지 정책을 선택합니다.
볼륨	볼륨의 크기를 지정합니다. 여러 스토리지 볼륨을 추가할 수 있습니다.

3 **검토 및 확인** 페이지에서 세부 정보를 검토하고 **마침**을 클릭합니다.



4 생성한 VM 클래스를 인스턴스 스토리지 VM에 사용하는 네임스페이스에 할당합니다.

vSphere Client를 사용하여 VM 클래스를 네임스페이스와 연결의 내용을 참조하십시오.

인스턴스 스토리지가 있는 VM 배포

DevOps 엔지니어는 인스턴스 스토리지 VM을 생성하는 데 필요한 VM 리소스에 액세스할 수 있는지 확인합니다. 리소스를 사용하여 VM을 배포합니다.

인스턴스 스토리지 VM을 배포할 때 일반 VM 배포 단계를 따릅니다. vSphere IaaS control plane에서 독립형 VM 배포의 내용을 참조하십시오. 이 절차에서는 인스턴스 스토리지 VM에 적용되는 추가 특정 항목을 설명합니다.

절차

- 인스턴스 스토리지 VM과 관련된 다음 항목을 확인합니다.
  - 네임스페이스에는 vSAN Direct 데이터스토어와 호환되는 스토리지 클래스가 포함됩니다.
  - 인스턴스 스토리지 VM 클래스는 이 스토리지 클래스를 참조합니다.

    인스턴스 스토리지 VM 클래스의 세부 정보를 검토할 때 `instanceStorage` 섹션이 포함되어 있는지 확인합니다.

    `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

...
```

- VM YAML 파일은 적절한 인스턴스 스토리지 VM 클래스를 가리킵니다.

vSphere IaaS control plane에서 구성 가능한 OVF 속성을 사용하여 VM 배포

일반적으로 DevOps 엔지니어가 vSphere IaaS control plane 환경에서 VM을 프로비저닝하는 경우 OVF 템플릿에는 기본 네트워크 구성과 같은 하드 코딩된 세부 정보가 포함됩니다. 하지만 VM CR이 생성될 때까지 IPAM 제공 네트워크 데이터와 같은 특정 값을 알지 못하고 VM의 OVF 속성에 할당할 수 없는 경우가 많습니다. 템플릿 문자열이 지원되면 네트워크 정보를 미리 알 필요가 없습니다. Golang 기반 템플릿을 사용하여 OVF 속성 값을 채우고 VM의 네트워크 스택을 구성할 수 있습니다.

절차

1 구성할 모든 속성에 대해 OVF 파일에 `ovf:userConfigurable="true"` 항목이 포함되어 있는지 확인합니다.

이 항목을 통해 시스템은 데이터가 수집된 후 네트워킹 값 자리 표시자(예: nameservers 및 management IP)를 실제 데이터로 대체할 수 있습니다.

다음 예를 사용합니다.

```
<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 템플릿 문자열을 사용하여 VM YAML 파일을 생성합니다.

부트스트랩 리소스에 대한 템플릿 문자열은 OVF 속성 값을 채우는 데 필요한 데이터를 수집합니다.

다음 방법 중 하나를 사용하여 템플릿 문자열을 구성할 수 있습니다.

- vm-operator-api를 사용합니다.

  자세한 내용은 GitHub의 https://github.com/vmware-tanzu/vm-operator/blob/main/api/v1alpha2/virtualmachinetempl_types.go 페이지를 참조하십시오.

  다음은 샘플 YAML 파일입니다.

```
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 }}"
```

- 다음 함수를 사용합니다.

함수 이름	서명	설명
V1alpha2_FirstIP	func () string	첫 번째 NIC에서 첫 번째 비루프백 IP를 가져옵니다.
V1alpha2_FirstIPFromNIC	func (index int) string	i번째 NIC에서 비루프백 IP 주소를 가져옵니다. 인덱스가 범위를 벗어나면 템플릿 문자열이 구문 분석되지 않습니다.

함수 이름	서명	설명
V1alpha2_FormatIP	func (IP string, netmask string) string	네트워크 길이로 IP 주소 형식을 지정합니다. 넷마스크는 길이(예: /24)이거나 십진수 표기법(예: 255.255.255.0)일 수 있습니다. 입력된 넷마스크가 유효하지 않거나 기본 마스크와 다르면 구문 분석되지 않습니다.
V1alpha2_FormatNameservers	func (count int, delimiter string) string	특정 구분 기호를 사용하여 처음 발생한 네임서버 수의 형식을 지정합니다. count가 음수이면 모든 네임서버를 의미합니다.
V1alpha2_IP	func(IP string) string	기본 넷마스크 CIDR을 사용하여 정적 IP 주소 형식을 지정합니다. IP가 유효하지 않으면 템플릿 문자열이 구문 분석되지 않습니다.
V1alpha2_IPsFromNIC	func (index int) []string	i번째 NIC의 모든 IP를 나열합니다. 인덱스가 범위를 벗어나면 템플릿 문자열이 구문 분석되지 않습니다.

함수를 사용하는 경우 YAML 파일은 다음과 같습니다.

```
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 VM을 배포합니다.

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

다음에 수행할 작업

사용자 지정이 실패하고 VM이 IP 주소를 가져오지 하면 vSphere VM 웹 콘솔을 사용하여 VM을 검사합니다. vSphere VM 웹 콘솔을 사용하여 VM 문제 해결의 내용을 참조하십시오.

vSphere IaaS control plane에서 사용 가능한 가상 시스템 모니터링

vSphere 관리자는 vSphere Client를 사용하여 vSphere IaaS control plane Kubernetes 환경에서 DevOps가 배포한 VM을 모니터링합니다.

VM 수명 주기는 vSphere Client에서 관리할 수 없습니다.

사전 요구 사항

DevOps 엔지니어가 VM을 배포했습니다. "vSphere IaaS control plane에서 독립형 VM 배포" 의 내용을 참조하십시오.

절차

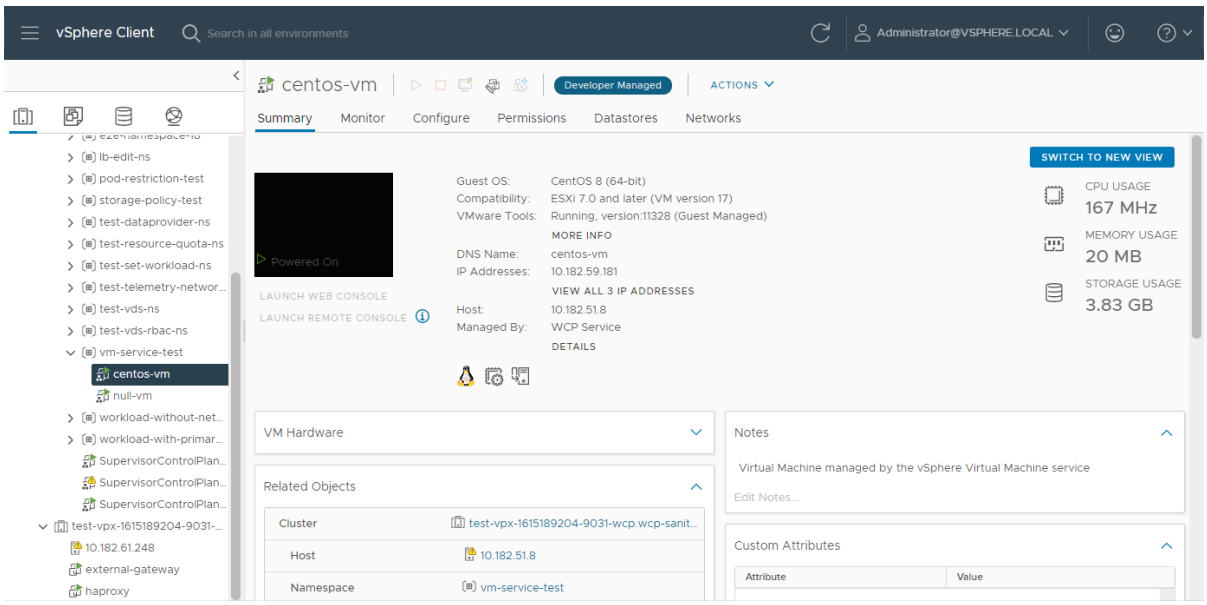
1 vSphere Client에서 vSphere IaaS control plane를 사용하도록 설정한 호스트 클러스터로 이동합니다.

2 **네임스페이스**에서 VM이 배포된 네임스페이스를 확장합니다.

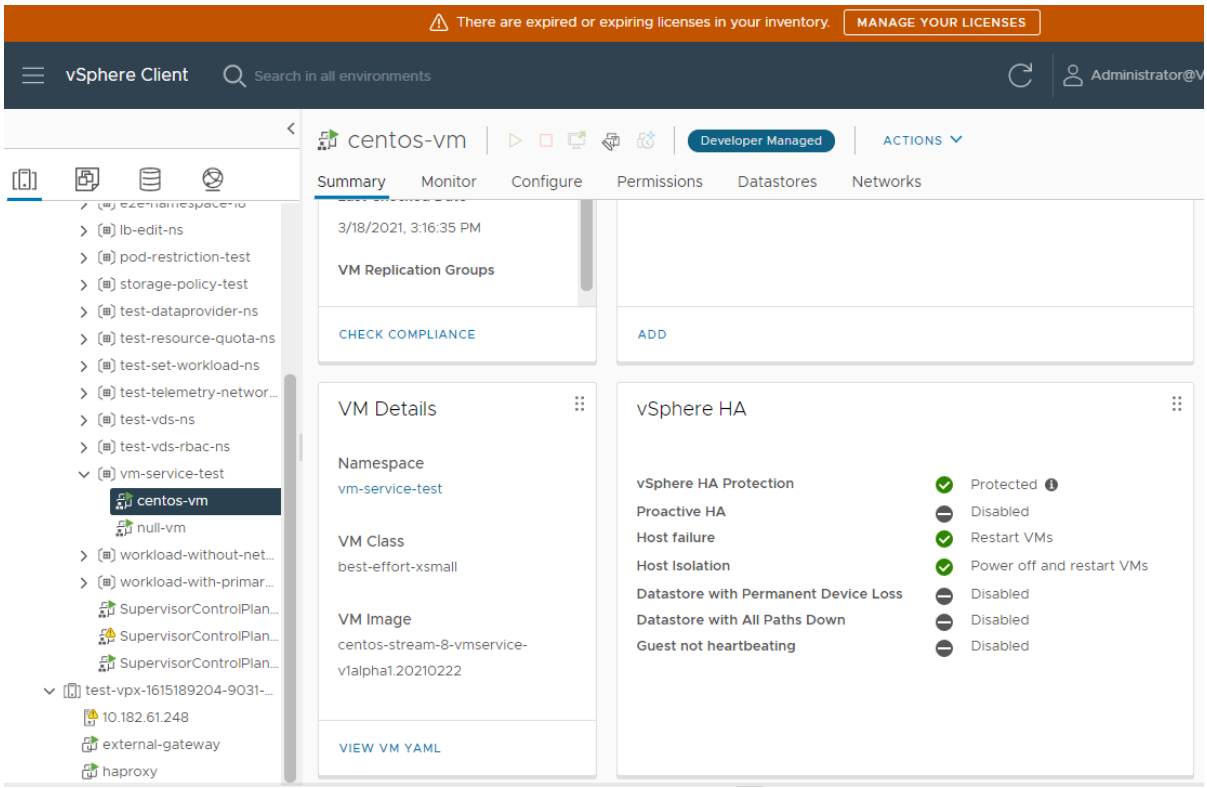
3 보려는 VM을 선택하고 **요약** 탭을 클릭합니다.

요약 페이지의 맨 위에 **개발자 관리** 태그가 보이는지 확인합니다.

이 페이지에는 게스트 운영 체제 및 IP 주소를 비롯한 VM에 대한 정보가 표시됩니다.



4 페이지 오른쪽 상단 모서리에 있는 **새 보기로 전환**을 클릭하면 VM 클래스 및 VM 이미지, VM이 실행되는 네임스페이스와 같은 추가 세부 정보가 표시됩니다.



vSphere VM 웹 콘솔을 사용하여 VM 문제 해결

DevOps 엔지니어는 vSphere VM 웹 콘솔을 사용하여 문제가 있는 VM에 액세스하고 문제를 해결할 수 있습니다. VM 웹 콘솔을 사용하면 일반 네트워크를 통해 VM에 액세스할 수 없는 경우(예: 처음 부팅하는 동안 게스트 운영 체제가 올바른 네트워크 설정을 구성하지 못한 경우) 도움이 될 수 있습니다.

VM 웹 콘솔은 구성 가능한 OVF 속성을 사용하여 VM을 배포할 때 특히 유용합니다. 자세한 내용은 vSphere IaaS control plane에서 구성 가능한 OVF 속성을 사용하여 VM 배포의 내용을 참조하십시오.

사전 요구 사항

문제가 있는 VM이 배포된 네임스페이스에 대한 편집 또는 소유자 권한이 있어야 합니다. 자세한 내용은 vSphere IaaS Control Plane ID 및 액세스 관리를 참조하십시오.

절차

1 Kubernetes 환경의 네임스페이스에 액세스합니다.

vSphere IaaS control plane에서 감독자 컨텍스트 가져오기 및 사용의 내용을 참조하십시오.

2 VM이 배포되었는지 확인합니다.

```
kubectl get vm -n namespace-name
```

출력은 다음과 비슷합니다.

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

3 VM 웹 콘솔에 대한 URL을 가져옵니다.

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

참고 명령은 인증된 URL을 VM의 웹 콘솔에 출력으로 반환합니다. 변경할 수 없는 시간(2분으로 설정) 내에 URL을 사용하지 않으면 URL이 만료됩니다. URL을 열어 웹 콘솔 페이지에 연결하면 세션 시간이 WebMKS에 의해 제어되고 더 오래 지속됩니다.

4 URL을 클릭하고 VM에 필요한 문제 해결 작업을 수행합니다.

vSphere 포드에 워크로드 배포

7

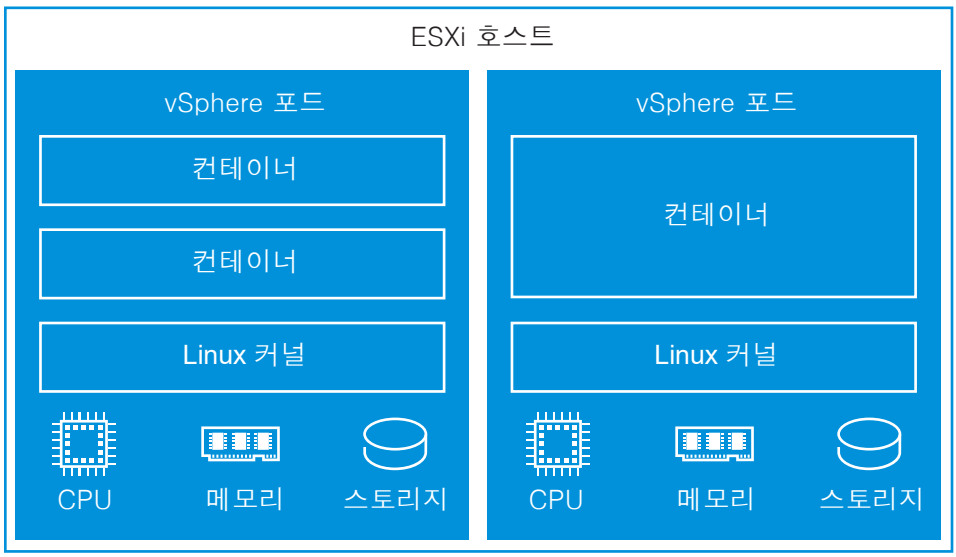
DevOps 엔지니어는 감독자에서 실행 중인 vSphere 네임스페이스의 리소스 경계 내에서 vSphere 포드의 수명 주기를 배포하고 관리할 수 있습니다.

참고 NSX 네트워킹 스택으로 구성된 감독자에만 vSphere 포드를 배포할 수 있습니다. VDS 스택으로 구성된 감독자에는 vSphere 포드를 배포할 수 없습니다. 감독자 서비스는 NSX 또는 VDS로 구성된 감독자 둘 다에서 지원되며 자체 사용을 위해 vSphere 포드를 배포합니다. 하지만 VDS로 구성된 감독자에서 일반 사용을 위해 vSphere 포드를 배포할 수는 없습니다.

vSphere 포드란?

vSphere IaaS control plane에는 Kubernetes 포드와 동일한 vSphere 포드라는 구조가 도입되었습니다. vSphere 포드는 하나 이상의 Linux 컨테이너를 실행하는 설치 공간이 작은 VM입니다. 각 vSphere 포드는 수용하는 워크로드 맞게 크기가 정확하게 조정되며 해당 워크로드에 대한 명시적 리소스 예약이 있습니다. 워크로드를 실행하는 데 필요한 정확한 양의 스토리지, 메모리 및 CPU 리소스를 할당합니다. vSphere 포드는 NSX를 사용하여 네트워킹 스택으로 구성된 감독자에서만 지원됩니다.

그림 7-1. vSphere 포드



vSphere 포드는 vCenter Server의 개체이며 워크로드에 대해 다음 기능을 사용할 수 있습니다.

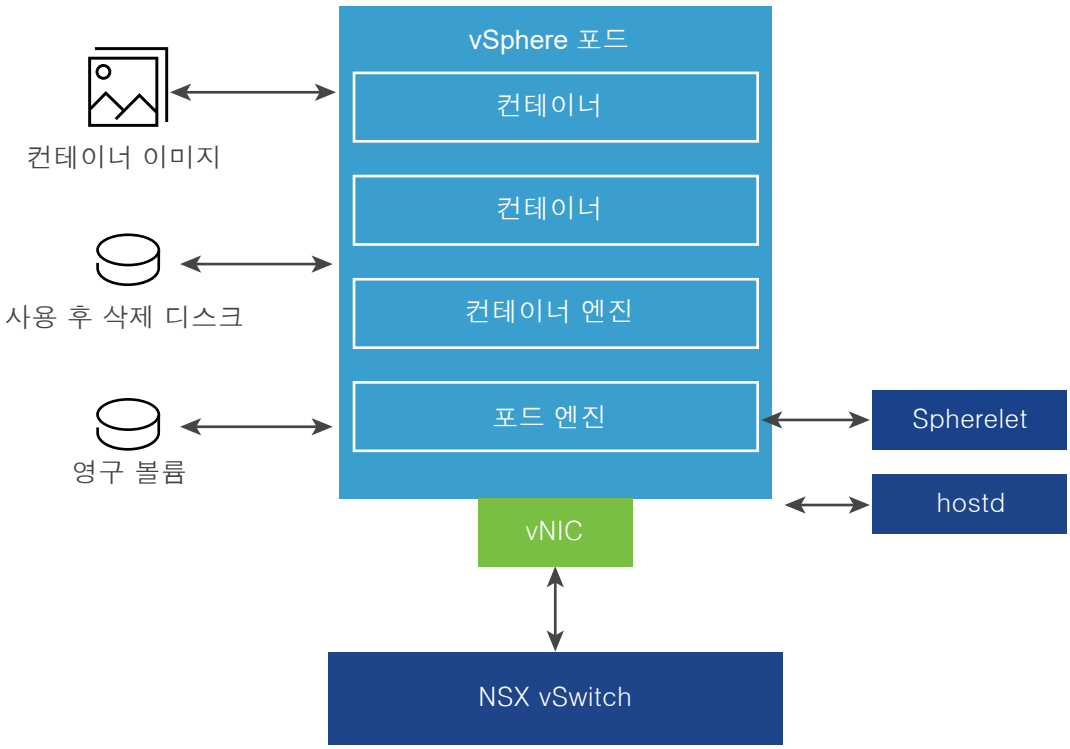
- 강력한 격리. vSphere 포드는 가상 시스템과 동일한 방식으로 격리됩니다. 각 vSphere 포드에는 Photon OS에 사용되는 커널을 기반으로 하는 고유한 Linux 커널이 있습니다. vSphere 포드에서는 베어메탈 구성처럼 많은 컨테이너가 커널을 공유하는 것이 아니라, 각 컨테이너에 고유한 Linux 커널이 있습니다.
- 리소스 관리. vSphere DRS는 감독자에서 vSphere 포드의 배치를 처리합니다.
- 고성능. vSphere 포드는 VM과 동일한 수준의 리소스 격리를 제공하며, 빠른 시작 시간과 낮은 컨테이너 오버헤드를 유지하면서 방해가 되는 인접 네트워크 문제를 제거합니다.
- 진단. vSphere 관리자는 vSphere에서 워크로드에 제공되는 모든 모니터링 및 검사 도구를 사용할 수 있습니다.

vSphere 포드는 OCI(Open Container Initiative)와 호환되며, 컨테이너가 OCI와 호환되는 한 모든 운영 체제에서 컨테이너를 실행할 수 있습니다.

vSphere 포드 배포 지침

vSphere 포드를 배포하기 전에 환경이 다음 요구 사항을 충족하는지 확인해야 합니다.

그림 7-2. vSphere 포드 네트워킹 및 스토리지



네임스페이스

감독자에 편집 또는 소유자 권한으로 구성된 vSphere 네임스페이스가 있어야 합니다. NSX 네트워킹을 사용하는 1개 영역 감독자만 vSphere 포드를 지원합니다.

감독자를 생성하려면 NSX 네트워킹을 사용하여 1개 영역 감독자 배포를 참조하십시오.

네임스페이스 생성에 대한 자세한 내용은 감독자에서 vSphere 네임스페이스 생성 및 구성 항목을 참조하십시오.

사용 권한 할당에 대한 자세한 내용은 ID 및 액세스 관리를 참조하십시오.

네트워킹

네트워킹의 경우 vSphere 포드는 NSX에서 제공된 토폴로지를 사용합니다. 자세한 내용은 감독자 네트워킹을 참조하십시오.

Spherelet은 각 호스트에서 생성되는 추가 프로세스입니다. kubelet은 기본적으로 ESXi로 이식되고 ESXi 호스트가 Kubernetes 클러스터의 일부가 되도록 허용합니다.

스토리지

vSphere 포드는 저장된 개체에 따라, 사용 후 삭제되는 VMDK, 영구 볼륨 VMDK 및 컨테이너 이미지 VMDK라는 세 가지 유형의 스토리지를 사용합니다.

vSphere 관리자는 감독자를 사용하도록 설정할 때 컨테이너 이미지 캐시 및 사용 후 삭제되는 VMDK 배치에 대한 스토리지 정책을 구성합니다.

또한 vSphere 네임스페이스 수준에서 영구 볼륨 배치에 대한 스토리지 정책을 구성합니다. 장 8 vSphere IaaS control plane에서 감독자 워크로드와 함께 영구 스토리지 사용에서 vSphere IaaS control plane의 스토리지 요구 사항 및 개념에 대한 자세한 내용을 참조하십시오.

다음으로 아래 항목을 읽으십시오.

- vSphere IaaS control plane에서 감독자 컨텍스트 가져오기 및 사용
- vSphere 네임스페이스의 vSphere 포드에 애플리케이션 배포
- vSphere 포드 애플리케이션 스케일 업/다운
- 기밀 vSphere 포드 배포
- vSphere IaaS control plane에서 vSphere 포드 워크로드 배포

vSphere IaaS control plane에서 감독자 컨텍스트 가져오기 및 사용

vSphere 관리자가 감독자에서 Kubernetes 제어부의 IP 주소를 제공하면 감독자에 로그인하여 액세스 권한이 있는 컨텍스트를 가져올 수 있습니다. vSphere IaaS control plane에서 컨텍스트는 감독자의 네임스페이스에 해당합니다.

감독자에 로그인하면 kubectl용 vSphere 플러그인은 클러스터에 대한 컨텍스트를 생성합니다. Kubernetes에서 구성 컨텍스트에는 클러스터, 네임스페이스 및 사용자가 포함됩니다. `.kube/config` 파일에서 클러스터 컨텍스트를 볼 수 있습니다. 이 파일은 일반적으로 `kubeconfig` 파일이라고 합니다.

참고 기존 `kubeconfig` 파일이 있는 경우 각 클러스터 컨텍스트에 추가됩니다. kubectl용 vSphere 플러그인은 kubectl 자체에서 사용하는 KUBECONFIG 환경 변수를 고려합니다. 필요하지 않더라도 `kubectl vsphere login ...`을 실행하기 전에 이 변수를 설정하면 정보가 현재의 `kubeconfig` 파일에 추가되지 않고 새 파일에 기록되므로 유용할 수 있습니다.

사전 요구 사항

- vCenter Single Sign-On 자격 증명을 가져옵니다.
- 감독자 제어부의 IP 주소를 가져옵니다.
- vSphere 네임스페이스의 이름을 가져옵니다.
- vSphere 네임스페이스에 대한 **편집** 권한이 있는지 확인합니다.
- vSphere용 Kubernetes CLI 도구 다운로드 및 설치를 수행합니다. "vSphere IaaS 제어부 설치 및 구성" 설명서를 참조하십시오.
- 서명 CA를 신뢰 루트로 설치하거나 인증서를 신뢰 루트로 직접 추가하여 Kubernetes 제어부가 제공하는 인증서를 시스템에서 신뢰할 수 있는지 확인합니다. "vSphere IaaS 제어부 설치 및 구성" 설명서에서 vSphere IaaS 제어부 클러스터에 대한 보안 로그인 구성을 참조하십시오.

절차

1 로그인을 위한 명령 구문 및 옵션을 보려면 다음 명령을 실행합니다.

```
kubectl vsphere login --help
```

2 감독자에 연결하려면 다음 명령을 실행합니다.

```
kubectl vsphere login --server=<KUBERNETES-CONTROL-PLANE-IP-ADDRESS> --vsphere-username
<VCENTER-SSO-USER>
```

예:

```
kubectl vsphere login --server=10.92.42.13 --vsphere-username administrator@example.com
```

이 작업은 Kubernetes API에 인증하기 위한 JWT(JSON Web Token)가 들어 있는 구성 파일을 생성합니다.

3 인증하려면 사용자 암호를 입력합니다.

감독자에 연결한 후에는 구성 컨텍스트가 액세스할 수 있는 것으로 표시됩니다. 예:

```
You have access to the following contexts:
tanzu-ns-1
tkg-cluster-1
tkg-cluster-2
```

4 액세스할 수 있는 구성 컨텍스트의 세부 정보를 보려면 다음 `kubectl` 명령을 실행합니다.

```
kubectl config get-contexts
```

CLI에는 사용 가능한 각 컨텍스트에 대한 세부 정보가 표시됩니다.

5 컨텍스트 간에 전환하려면 다음 명령을 사용합니다.

```
kubectl config use-context <example-context-name>
```

다음에 수행할 작업

Tanzu Kubernetes Grid 클러스터에 연결하려면 "vSphere IaaS 제어부에서 TKG 서비스 사용" 에서 vCenter Single Sign-On 사용자로 TKG 클러스터에 연결을 참조하십시오.

vSphere 네임스페이스의 vSphere 포드에 애플리케이션 배포

vSphere IaaS control plane의 vSphere 네임스페이스에 애플리케이션을 배포할 수 있습니다. 애플리케이션을 배포한 후에는 해당하는 수의 vSphere 포드가 네임스페이스 내의 감독자에 생성됩니다.

사전 요구 사항

- vSphere 관리자로부터 감독자에 있는 Kubernetes 제어부의 IP 주소를 가져옵니다.
- vCenter Single Sign-On에서 사용자 계정을 가져옵니다.
- vSphere 관리자에게 필요한 컨텍스트에 액세스할 수 있는 사용 권한이 있는지 확인합니다.

절차

1 Kubernetes 환경의 네임스페이스에 액세스합니다.

vSphere IaaS control plane에서 감독자 컨텍스트 가져오기 및 사용의 내용을 참조하십시오.

2 애플리케이션을 배포하려는 컨텍스트로 전환합니다.

```
kubectl config use-context <namespace>
```

3 애플리케이션을 배포합니다.

```
kubectl apply -f <application name>.yaml
```

vSphere 포드 애플리케이션 스케일 업/다운

vSphere IaaS control plane의 감독자에서 실행 중인 각 애플리케이션의 복제본 수를 늘리거나 줄일 수 있습니다.

사전 요구 사항

- vSphere 관리자로부터 감독자에 있는 Kubernetes 제어부의 IP 주소를 가져옵니다.
- vCenter Single Sign-On에서 사용자 계정을 가져옵니다.
- vSphere 관리자에게 필요한 컨텍스트에 액세스할 수 있는 사용 권한이 있는지 확인합니다.

절차

1 감독자로 인증합니다.

```
kubectl vsphere login --server <control plane load balancer IP address> --vsphere-username
<vSphere user account name>
```

2 애플리케이션을 스케일 업 또는 스케일 다운합니다.

```
kubectl get deployments
kubectl scale deployment <deployment-name> --replicas=<number-of-replicas>
```

기밀 vSphere 포드 배포

vSphere IaaS control plane를 사용하면 감독자에서 기밀 vSphere 포드를 실행할 수 있습니다. 기밀 vSphere 포드는 게스트 운영 체제 메모리를 암호화된 상태로 유지하여 하이퍼바이저의 액세스로부터 보호하는 하드웨어 기술을 사용합니다.

SEV-ES(Secure Encrypted Virtualization-Encrypted State)를 추가적인 보안 강화 항목으로 추가하여 기밀 vSphere 포드를 생성할 수 있습니다. SEV-ES는 CPU 레지스터가 하이퍼바이저와 같은 구성 요소로 레지스터의 정보를 누출하지 못하도록 합니다. SEV-ES는 또한 CPU 레지스터 상태에 대한 악의적인 수정 사항을 감지할 수 있습니다. vSphere 환경에서 SEV-ES 기술을 사용하는 방법에 대한 자세한 내용은 "vSphere 보안" 설명서에서 AMD Secure Encrypted Virtualization-Encrypted State를 사용하여 가상 시스템 보호를 참조하십시오.

사전 요구 사항

ESXi 호스트에서 SEV-ES를 사용하도록 설정하려면 vSphere 관리자가 다음 지침을 따라야 합니다.

- SEV-ES 기능을 지원하는 호스트를 사용합니다.
- ESXi 버전 7.0 업데이트 2 이상을 사용합니다.
- ESXi 시스템의 BIOS 구성에서 SEV-ES를 사용하도록 설정합니다. BIOS 구성 액세스에 대 한 자세한 내용은 시스템 설명서를 참조하십시오.

- BIOS에서 SEV-ES를 사용하도록 설정할 때 **Minimum SEV non-ES ASID** 설정 값을 호스트의 SEV-ES VM 및 기밀 vSphere 포드 수에 1을 더한 값과 동일하게 입력합니다. 예를 들어 SEV-ES VM 100개와 vSphere 포드 128개를 실행하려면 229 이상을 입력합니다. 이 설정은 최대 500까지 입력할 수 있습니다.

절차

1 다음 매개 변수를 포함하는 YAML 파일을 생성합니다.

a 주석에서 기밀 vSphere 포드 기능을 사용하도록 설정합니다.

```
...
annotations:
        vmware/confidential-pod: enabled
...
```

b 컨테이너에 대한 메모리 리소스를 지정합니다.

이 예제와 같이 메모리 요청과 메모리 제한을 동일한 값으로 설정해야 합니다.

```
resources:
      requests:
        memory: "512Mi"
      limits:
        memory: "512Mi"
```

다음 YAML 파일을 예로 사용할 수 있습니다.

```
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 감독자에 로그인합니다.

```
kubectl vsphere login --server=https://<server_adress> --vsphere-username <your user
account name>
```

3 애플리케이션을 배포하려는 네임스페이스로 전환합니다.

```
kubectl config use-context <namespace>
```

4 YAML 파일에서 기밀 vSphere 포드을 배포합니다.

```
kubectl apply -f <yaml file name>.yaml
```

참고 vSphere 포드이 배포되면 DRS는 SEV-ES를 지원하는 ESXi 노드에 배치합니다. 해당 노드를 사용할 수 없으면 vSphere 포드 노드가 실패로 표시됩니다.

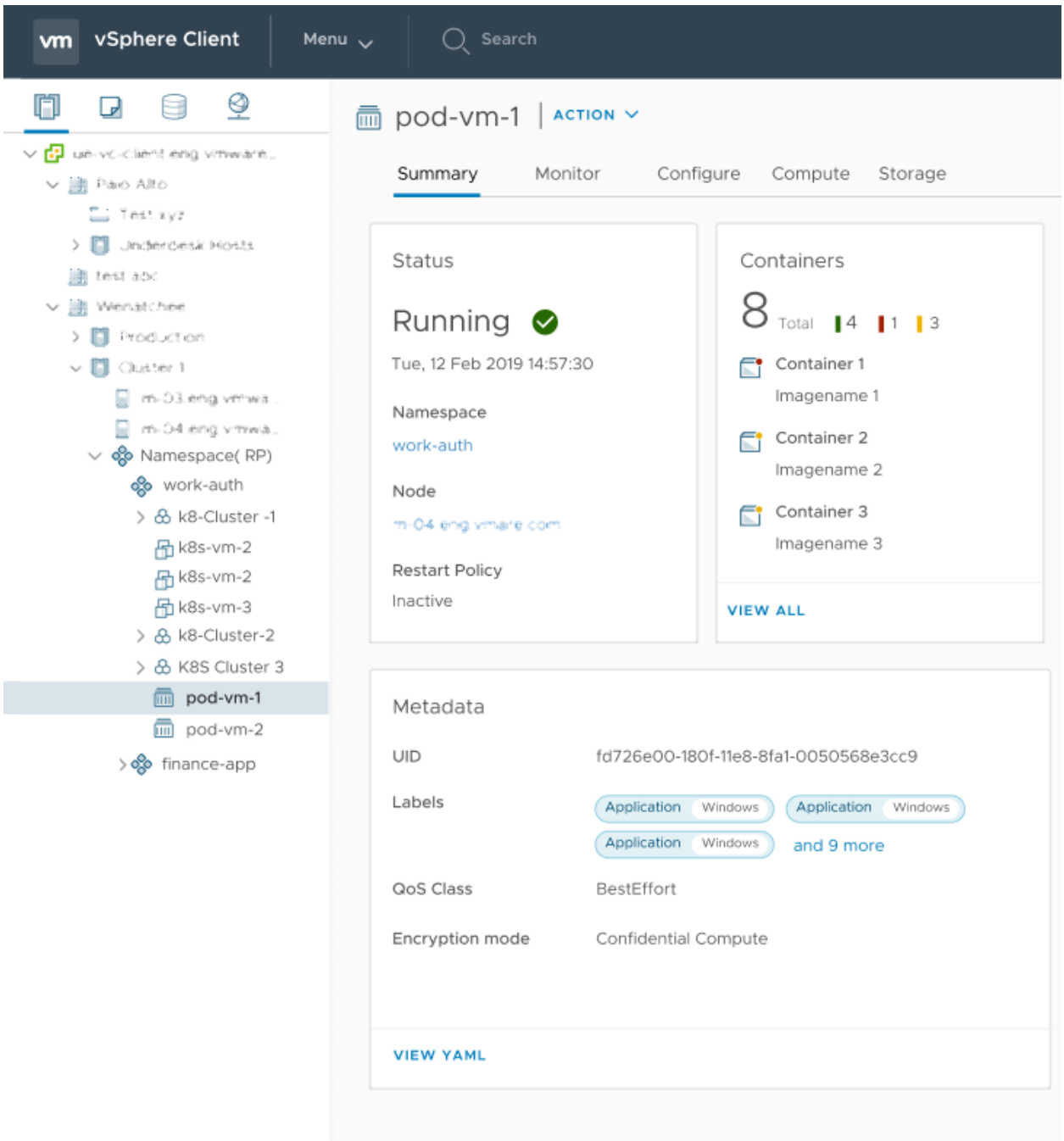
시작된 기밀 vSphere 포드은 이 포드에서 실행 중인 모든 워크로드에 대한 하드웨어 메모리 암호화 지원을 제공합니다.

5 다음 명령을 실행하여 기밀 vSphere 포드이 생성되었는지 확인합니다.

```
kubectl describe pod/<yaml name>
```

다음에 수행할 작업

vSphere 관리자는 기밀 vSphere 포드을 볼 수 있습니다. vSphere Client에는 **암호화 모드: 기밀 계산** 태그와 함께 표시됩니다.



vSphere IaaS control plane에서 vSphere 포드 워크로드 배포

이 예제 자습서에서는 vSphere IaaS control plane 환경에서 vSphere 포드를 사용하여 WordPress 애플리케이션을 배포하는 방법을 설명합니다.

WordPress 배포에는 WordPress 프런트 엔드 및 MySQL 백엔드용 컨테이너와 둘 다에 대한 서비스가 포함됩니다. 비밀 개체도 필요합니다.

이 자습서에서는 배포 개체를 사용합니다. 운영 환경에서는 일반적으로 WordPress 및 MySQL 컨테이너 모두에 대해 StatefulSets를 사용합니다.

사전 요구 사항

- NSX 네트워킹을 사용하여 1개 영역 감독자를 생성합니다. NSX를 사용하는 1개 영역 감독자만 vSphere 포드를 지원합니다. NSX 네트워킹을 사용하여 1개 영역 감독자 배포를 참조하십시오.
- vSphere 포드를 배포하기 위한 네임스페이스를 생성합니다. 감독자에서 vSphere 네임스페이스 생성 및 구성의 내용을 참조하십시오.
- 스토리지 정책(예: `vwt-storage-policy`)을 생성하여 네임스페이스에 할당합니다.
- vSphere Kubernetes CLI 도구를 다운로드합니다. vSphere용 Kubernetes CLI 도구 다운로드 및 설치를 참조하십시오.
- 이 자습서에 필요한 YAML 파일을 생성하고 파일에 대한 명령줄 액세스를 확인합니다.

범주	파일
스토리지	■ mysql-pvc.yaml ■ wordpress-pvc.yaml **참고** 파일이 올바른 스토리지 클래스를 참조하는지 확인합니다.
암호	■ regcred.yaml ■ mysql-pass.yaml
서비스	■ mysql-service.yaml ■ wordpress-service.yaml
배포	■ mysql-deployment-vsphere-pod.yaml ■ wordpress-deployment.yaml

WordPress 배포

이 워크플로를 사용하면 vSphere 포드를 사용하여 WordPress 애플리케이션을 배포합니다.

1부. 네임스페이스 액세스

다음 단계를 사용하여 네임스페이스에 액세스합니다.

절차

1 감독자에 로그인합니다.

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

2 vSphere 네임스페이스로 전환합니다.

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

3 생성한 스토리지 정책(`vwt-storage-policy`)을 네임스페이스에서 스토리지 클래스로 사용할 수 있는지 확인합니다.

vSphere 네임스페이스에서 스토리지 클래스 표시의 내용을 참조하십시오.

2부. WordPress PVC 생성

다음 명령을 사용하여 WordPress PVC를 생성합니다.

절차

1 MySQL PVC를 생성합니다.

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

2 WordPress PVC를 생성합니다.

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

3 PVC를 확인합니다.

```
kubectl get pvc,pv
```

3부. 비밀 생성

공용 Docker Hub는 Kubernetes의 기본 컨테이너 레지스트리입니다. 이제 Docker Hub에서 이미지 끌어오기가 제한됩니다. 유료 계정이 있어야 하고 계정 키를 `data.dockerconfigjson` 필드의 비밀 YAML에 추가해야 합니다.

절차

1 Docker Hub 레지스트리 비밀을 생성합니다.

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

2 mysql 암호 비밀을 생성합니다.

MySQL DB 암호는 필수입니다. 비밀 내에서 암호는 base64로 인코딩되어야 합니다.

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

3 비밀을 확인합니다.

```
kubectl get secrets
```

4부. 서비스 생성

다음 단계에 따라 서비스를 생성합니다.

절차

1 MySQL 서비스를 생성합니다.

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

2 WordPress 서비스를 생성합니다.

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

3 서비스를 확인합니다.

```
kubectl get services
```

5부. 포드 배포 생성

이 작업을 사용하여 포드 배포를 생성합니다.

이 자습서에서는 배포 개체를 사용합니다. 운영 환경에서는 WordPress 및 MySQL 컨테이너 모두에 대해 StatefulSets를 사용해야 합니다.

절차

1 MySQL 배포를 생성합니다.

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

참고 vSphere 포드가 생성되면 시스템은 포드에 컨테이너에 대한 VM을 생성합니다. 기본적으로 VM의 RAM 제한은 512MB입니다. MySQL 컨테이너에는 더 많은 메모리가 필요합니다. 포드 배포 규격 `mysql-deployment-vsphere-pod.yaml`에는 vSphere 포드 VM에 제공된 메모리를 늘리는 섹션이 포함되어 있습니다. 이 섹션을 포함하지 않으면 포드 배포가 실패하고 OOM(메모리 부족) 예외가 발생합니다. MySQL 포드를 TKG 클러스터에 배포할 때는 RAM을 늘릴 필요가 없습니다.

2 WordPress 배포를 생성합니다.

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

3 배포를 확인합니다.

```
kubectl get deployments
```

6부. WordPress 테스트

다음 단계에 따라 WordPress 배포를 테스트합니다.

절차

1 모든 개체가 생성되어 실행 중인지 확인합니다.

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

2 WordPress 서비스에서 EXTERNAL-IP 주소를 가져옵니다.

```
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 주소를 찾습니다.

4 WordPress 인스턴스를 구성합니다.

사용자 이름: administrator

암호: 제공된 강력한 암호 사용

WordPress 배포를 위한 예제 YAML 파일

다음 예제 YAML 파일은 vSphere 포드와 함께 WordPress 애플리케이션을 배포할 때 사용합니다.

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 plane에서 감독자 워크로드와 함께 영구 스토리지 사용

8

DevOps가 감독자의 네임스페이스에서 실행하는 특정 Kubernetes 워크로드에는 데이터를 영구적으로 저장하기 위한 영구 스토리지가 필요합니다. 영구 스토리지는 네임스페이스에서 실행하는 vSphere 포드, Tanzu Kubernetes Grid 클러스터, VM 및 기타 워크로드에서 사용할 수 있습니다.

DevOps 팀이 영구 스토리지를 사용할 수 있도록 vSphere 관리자는 다양한 스토리지 요구 사항 및 서비스 클래스를 설명하는 스토리지 정책을 생성합니다. 그런 다음 관리자는 스토리지 정책을 할당하고 네임스페이스 수준에서 스토리지 제한을 구성합니다.

vSphere IaaS control plane가 영구 스토리지에서 작동하는 방식을 이해하려면 스토리지 클래스, 영구 볼륨 및 영구 볼륨 할당과 같은 필수 Kubernetes 개념을 숙지해야 합니다. 자세한 내용은 https://kubernetes.io/docs/home/에서 Kubernetes 설명서를 참조하십시오.

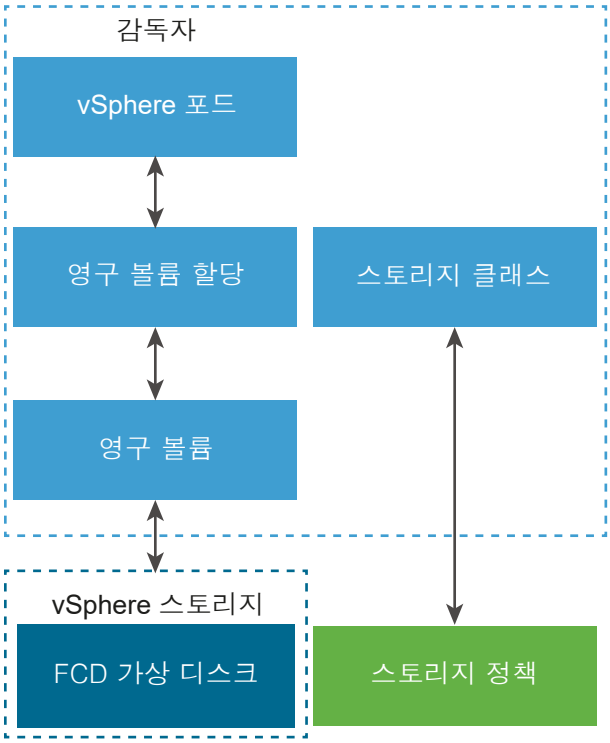
vSphere IaaS control plane 구성 요소가 스토리지와 통합되는 방식에 대한 자세한 내용은 "vSphere IaaS 제어부 개념 및 계획" 의 감독자 스토리지를 참조하십시오.

영구 스토리지 워크플로

vSphere IaaS control plane에서 영구 스토리지를 프로비저닝하는 워크플로에는 일반적으로 다음과 같은 순차적 작업이 포함됩니다.

작업	수행자	설명
DevOps 팀에 영구 스토리지 리소스 제공	vSphere 관리자	vSphere 관리자가 다양한 스토리지 요구 사항 및 서비스 클래스를 설명하는 스토리지 정책을 생성합니다. "vSphere IaaS 제어부 설치 및 구성" 설명서의 vSphere IaaS 제어부에 대한 스토리지 정책 생성을 참조하십시오. 그런 다음 관리자는 네임스페이스에 스토리지 정책을 할당하고 네임스페이스에 대한 스토리지 제한을 설정합니다. 감독자에서 vSphere 네임스페이스 생성 및 구성의 내용을 참조하십시오.
네임스페이스에 스토리지 클래스 생성	vSphere IaaS control plane	네임스페이스에 할당된 스토리지 정책과 일치하는 스토리지 클래스가 Kubernetes 환경에 자동으로 나타납니다. vSphere 관리자가 네임스페이스에 여러 스토리지 정책을 할당하면 각 스토리지 정책에 대해 별도의 스토리지 클래스가 생성됩니다. Tanzu Kubernetes Grid 클러스터를 사용하는 경우 각 클러스터는 클러스터가 프로비저닝된 네임스페이스에서 스토리지 클래스를 상속합니다. DevOps 팀은 영구 스토리지 요구 사항에 맞게 스토리지 클래스를 사용할 수 있습니다. vSphere 네임스페이스에서 스토리지 클래스 표시를 참조하십시오.
워크로드에 대한 영구 스토리지 리소스 요청	DevOps	DevOps 팀은 스토리지 클래스를 사용하여 워크로드에 대한 영구 스토리지 리소스를 요청합니다. 요청은 특정 스토리지 클래스를 참조하는 영구 볼륨 할당의 형태로 제공됩니다. vSphere IaaS control plane에서 동적 영구 볼륨 프로비저닝 및 vSphere IaaS control plane에서 독립형 VM 배포를 참조하십시오.
워크로드에 대해 일치하는 영구 가상 디스크와 영구 볼륨 개체 생성	vSphere IaaS control plane	vSphere IaaS control plane은 원래 스토리지 정책 및 일치하는 스토리지 클래스에 지정된 요구 사항을 충족하는 데이터스토어에 가상 디스크를 배치합니다. 가상 디스크는 워크로드에 의해 마운트될 수 있습니다.
영구 볼륨 모니터링	vSphere 관리자	vSphere 관리자는 vSphere Client를 사용하여 영구 볼륨 및 해당 백업 가상 디스크를 모니터링합니다. 또한 영구 볼륨의 스토리지 규정 준수 상태와 영구 볼륨 상태를 모니터링할 수도 있습니다. vSphere Client에서 영구 볼륨 모니터링을 참조하십시오.

다음은 영구 볼륨 개체 및 일치하는 영구 FCD 가상 디스크가 vSphere 포드에 대해 생성되는 방식을 보여 줍니다. 영구 스토리지 할당은 특정 스토리지 클래스를 참조합니다.



다음으로 아래 항목을 읽으십시오.

- vSphere 네임스페이스에서 스토리지 클래스 표시
- vSphere IaaS control plane에서 동적 영구 볼륨 프로비저닝
- vSphere IaaS control plane에서 정적 영구 볼륨 프로비저닝
- vSAN 파일 서비스를 사용하여 vSphere IaaS control plane에서 ReadWriteMany 볼륨 생성
- vSphere IaaS control plane의 볼륨 확장
- vSphere Client에서 영구 볼륨 모니터링
- vSphere 네임스페이스 또는 Tanzu Kubernetes Grid 클러스터의 볼륨 상태 모니터링
- 3개 영역 감독자에서 영구 스토리지를 사용하는 모범 사례

vSphere 네임스페이스에서 스토리지 클래스 표시

vSphere 관리자가 스토리지 정책을 생성하여 vSphere IaaS control plane의 vSphere 네임스페이스에 할당하면 이 스토리지 정책이 vSphere 네임스페이스에 일치하는 Kubernetes 스토리지 클래스로 나타납니다. 또한 사용 가능한 Tanzu Kubernetes Grid 클러스터에도 복제됩니다. DevOps 엔지니어는 스토리지 클래스를 사용할 수 있는지 확인할 수 있습니다.

명령 실행 기능은 사용 권한에 따라 다릅니다.

사전 요구 사항

vSphere 관리자가 적절한 스토리지 정책을 생성했고 해당 정책을 vSphere 네임스페이스에 할당했는지 확인합니다.

절차

1 다음 명령 중 하나를 사용하여 스토리지 클래스를 사용할 수 있는지 확인합니다.

- **kubectl get storageclass**

  참고 이 명령은 관리자 권한이 있는 사용자만 사용할 수 있습니다.

  다음과 유사한 출력이 표시됩니다. 스토리지 클래스의 이름은 vSphere 측의 스토리지 정책 이름과 일치합니다.

```
NAME     PROVISIONER              AGE
silver   csi.vsphere.vmware.com   2d
gold     csi.vsphere.vmware.com   1d
```

- **kubectl describe namespace *namespace_name***

  출력에서 스토리지 클래스의 이름이 ***storageclass_name*.storageclass.storage.k8s.io/requests.storage** 매개 변수의 일부로 나타납니다. 예:

```
-----------------
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 네임스페이스에서 사용 가능한 스토리지 공간의 양을 확인하려면 다음 명령을 실행합니다.

kubectl describe resourcequotas -namespace *namespace*

다음과 유사한 출력이 표시됩니다.

```
Name:             ns-my-namespace
Namespace:        ns-my-namespace
Resource          Used  Hard
--------          ----  ----
requests.storage  0     200Gi
```

vSphere IaaS control plane에서 동적 영구 볼륨 프로비저닝

상태 저장 애플리케이션(예: 데이터베이스)은 세션 간 데이터를 저장하며, 데이터 저장을 위해 영구 볼륨이 필요합니다. vSphere IaaS control plane를 사용하면 애플리케이션에 대한 영구 볼륨을 동적으로 프로비저닝할 수 있습니다.

vSphere 환경에서 영구 볼륨 개체는 데이터스토어에 상주하는 가상 디스크로 백업됩니다. 데이터스토어는 스토리지 정책으로 표시됩니다. vSphere 관리자가 스토리지 정책(예: **`gold`**)을 생성하고 이를 감독자의 네임스페이스에 할당하면 이 스토리지 정책이 vSphere 네임스페이스 및 사용 가능한 모든 Tanzu Kubernetes Grid 클러스터에 일치하는 Kubernetes 스토리지 클래스로 나타납니다.

DevOps 엔지니어는 스토리지 클래스를 영구 볼륨 할당 규격에 사용할 수 있습니다. 그런 다음 영구 볼륨 할당에서 스토리지를 사용하는 애플리케이션을 배포할 수 있습니다. 이 예제에서는 애플리케이션의 영구 볼륨이 동적으로 생성됩니다.

사전 요구 사항

vSphere 관리자가 적절한 스토리지 정책을 생성했고 해당 정책을 네임스페이스에 할당했는지 확인합니다.

절차

1 vSphere Kubernetes 환경의 네임스페이스에 액세스합니다.

vSphere IaaS control plane에서 감독자 컨텍스트 가져오기 및 사용의 내용을 참조하십시오.

2 스토리지 클래스를 사용할 수 있는지 확인합니다.

vSphere 네임스페이스에서 스토리지 클래스 표시의 내용을 참조하십시오.

3 영구 볼륨 할당을 생성합니다.

a 영구 볼륨 할당 구성을 포함하는 YAML 파일을 생성합니다.

이 예에서 파일은 **gold** 스토리지 클래스를 참조합니다.

ReadWriteMany 영구 볼륨을 프로비저닝하려면 `accessModes`를 `ReadWriteMany`로 설정합니다. vSAN 파일 서비스를 사용하여 vSphere IaaS control plane에서 ReadWriteMany 볼륨 생성의 내용을 참조하십시오.

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

b Kubernetes 클러스터에 영구 볼륨 할당을 적용합니다.

```
kubectl apply -f pvc_name.yaml
```

이 명령은 할당의 스토리지 요구 사항을 충족하는 백업 가상 디스크가 있는 Kubernetes 영구 볼륨과 vSphere 볼륨을 동적으로 생성합니다.

c 영구 볼륨 할당의 상태를 확인합니다.

```
kubectl get pvc my-pvc
```

출력은 볼륨이 영구 볼륨 할당에 바인딩되어 있음을 보여 줍니다.

```
NAME     STATUS    VOLUME   CAPACITY   ACCESSMODES   STORAGECLASS   AGE
my-pvc   Bound     my-pvc   2Gi        RWO           gold           30s
```

4 영구 볼륨을 마운트하는 포드를 생성합니다.

a 영구 볼륨을 포함하는 YAML 파일을 생성합니다.

파일에는 다음과 같은 매개 변수가 포함되어 있습니다.

```
...
volumes:
    - name: my-pvc
      persistentVolumeClaim:
        claimName: my-pvc
```

b YAML 파일에서 포드를 배포합니다.

```
kubectl create -f pv_pod_name.yaml
```

c 포드가 생성되었는지 확인합니다.

```
kubectl get pod
```

```
NAME        READY   STATUS    RESTARTS   AGE
pod_name    1/1     Ready     0          40s
```

결과

구성한 포드는 영구 볼륨 할당에 설명된 영구 스토리지를 사용합니다.

다음에 수행할 작업

영구 볼륨의 상태를 모니터링하려면 vSphere 네임스페이스 또는 Tanzu Kubernetes Grid 클러스터의 볼륨 상태 모니터링 항목을 참조하십시오. vSphere Client의 영구 볼륨을 검토하고 모니터링하려면 vSphere Client에서 영구 볼륨 모니터링 항목을 참조하십시오.

vSphere IaaS control plane에서 정적 영구 볼륨 프로비저닝

감독자의 PVC(영구 볼륨 할당)를 사용하여 Tanzu Kubernetes Grid 클러스터에서 블록 볼륨을 정적으로 생성할 수 있습니다.

PVC는 다음 조건을 충족해야 합니다.

- PVC는 Tanzu Kubernetes Grid 클러스터가 상주하는 동일한 네임스페이스에 있습니다.
- PVC는 감독자의 vSphere 포드 또는 다른 Tanzu Kubernetes Grid 클러스터의 포드에 연결되어 있지 않습니다.

정적 프로비저닝을 사용하면 다른 Tanzu Kubernetes Grid 호스트에 더 이상 필요하지 않은 PVC를 새 Tanzu Kubernetes Grid 클러스터에서 재사용할 수 있습니다. 이렇게 하려면 원래 Tanzu Kubernetes Grid 클러스터에서 PV(영구 볼륨)의 `Reclaim policy`를 `Retain`으로 변경한 후 해당 PVC를 삭제합니다.

다음 단계에 따라 남은 기본 볼륨의 정보를 사용하여 새 Tanzu Kubernetes Grid 클러스터에 PVC를 정적으로 생성합니다.

절차

1 감독자의 원래 PVC 이름을 기록해둡니다.

이전 Tanzu Kubernetes Grid 클러스터의 PVC를 재사용하는 경우 Tanzu Kubernetes Grid 클러스터에 있는 이전 PV 개체의 `volumeHandle`에서 PVC 이름을 검색할 수 있습니다.

2 PV를 생성합니다.

YAML 파일에서 다음 항목의 값을 지정합니다.

- `storageClassName`의 경우 감독자에서 PVC에 사용되는 스토리지 클래스 이름을 입력합니다.
- `volumeHandle`의 경우 1단계에서 얻은 PVC 이름을 입력합니다.

다른 Tanzu Kubernetes Grid 클러스터의 볼륨을 재사용하는 경우에는 새 Tanzu Kubernetes Grid 클러스터에 PV를 생성하기 전에 이전 Tanzu Kubernetes Grid 클러스터에서 PVC 및 PV 개체를 삭제합니다.

다음 YAML 매니페스트를 예제로 사용합니다.

```
   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 단계 2단계에서 생성한 PV 개체와 매칭할 PVC를 생성합니다.

`storageClassName`을 PV에서와 동일한 값으로 설정합니다.

```
   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 생성한 PV에 PVC가 바인딩되었는지 확인합니다.

```
$ 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 파일 서비스를 사용하여 vSphere IaaS control plane에서 ReadWriteMany 볼륨 생성

vSphere IaaS control plane는 ReadWriteMany 모드에서 영구 볼륨을 지원합니다. ReadWriteMany 지원을 사용하면 TKG 클러스터에서 실행되는 여러 포드 또는 애플리케이션에서 단일 볼륨을 동시에 마운트할 수 있습니다. vSphere IaaS control plane는 ReadWriteMany 영구 볼륨에 대해 vSAN 파일 공유가 지원하는 CNS 파일 볼륨을 사용합니다. vSAN 공유를 사용하려면 vSAN 환경에서 vSAN 파일 서비스를 설정하고 감독자에서 파일 볼륨 지원을 활성화해야 합니다.

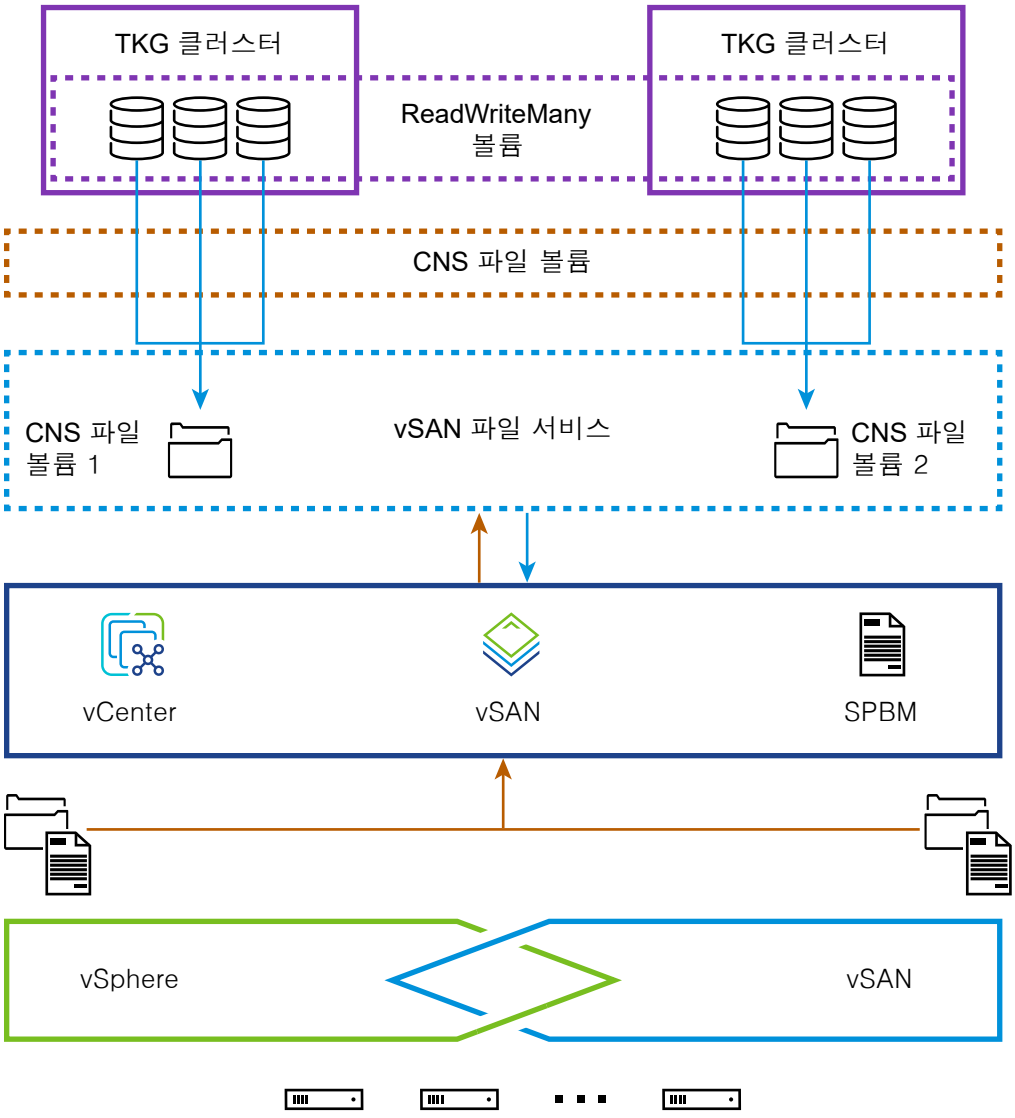
파일 볼륨에 대한 고려 사항

vSphere IaaS control plane에서 영구 볼륨에 대해 파일 지원을 사용하도록 설정하는 경우 다음 고려 사항에 유의하십시오.

- 파일 볼륨은 Tanzu Kubernetes Grid 클러스터의 워크로드에 대해서만 지원됩니다. 감독자 네임스페이스의 vSphere 포드 및 VM 서비스 VM과 같은 워크로드에는 지원되지 않습니다.
- Kubernetes에서 RWX 볼륨을 요청하면 vSAN 파일 서비스는 요청된 크기와 적절한 SPBM 정책의 NFS 기반 파일 공유를 생성합니다. RWX 볼륨당 하나의 vSAN 파일 공유가 생성됩니다. VMware는 vSAN 파일 서비스 클러스터당 100개의 공유를 지원합니다. 즉, RWX 볼륨은 100개 이하가 될 수 있습니다.
- TKG 클러스터에서는 TKr 버전 1.22 이상을 사용합니다.

  자세한 내용은 VMware Tanzu Kubernetes 릴리스 정보를 참조하십시오.
- vSphere IaaS control plane에 대한 파일 볼륨 지원을 사용하도록 설정하는 경우 잠재적인 보안 취약점에 유의하십시오.
  - 볼륨은 암호화 없이 마운트됩니다. 데이터가 네트워크를 통과하는 동안 암호화되지 않은 데이터에 액세스할 수 있습니다.

  - 감독자 네임스페이스 내에서 파일 공유 액세스를 분리하기 위해 ACL(Access Control List)이 파일 공유에 사용됩니다. IP 스푸핑 위험이 있을 수 있습니다.
- 네트워킹에 대한 다음 지침을 따르십시오.
  - vSphere IaaS control plane에서 네트워킹에 NSX를 사용하는 경우 감독자 네임스페이스에 NAT 모드가 사용되도록 설정되어 있는지 확인합니다. 감독자에서 vSphere 네임스페이스 생성 및 구성의 내용을 참조하십시오.
  - 워크로드 네트워크에서 vSAN 파일 서비스를 라우팅할 수 있는지와 워크로드 네트워크와 vSAN 파일 서비스 IP 주소 간에 NAT가 없는지 확인합니다.
  - vSAN 파일 서비스 및 vSphere IaaS control plane에 공통 DNS 서버를 사용합니다.
- 파일 볼륨 지원을 사용하도록 설정한 후 나중에 비활성화하면 클러스터에서 프로비저닝한 기존 ReadWriteMany 영구 볼륨은 영향을 받지 않고 사용 가능한 상태로 유지됩니다. 새 ReadWriteMany 영구 볼륨을 생성할 수 없게 됩니다.



영구 볼륨에 대한 파일 볼륨 지원을 사용하도록 설정하는 워크플로

이 프로세스에 따라 파일 볼륨 지원을 사용하도록 설정합니다.

1 vSphere 관리자는 구성된 vSAN 파일 서비스로 vSAN 클러스터를 설정합니다.

- vSAN 파일 서비스 사용 및 파일 서비스 구성을 참조하십시오.
- vSAN 확장 클러스터가 있는 환경의 특정 설정에 대해서는 확장 클러스터가 있는 vSAN 파일 서비스를 참조하십시오.

2 vSphere 관리자가 감독자에서 파일 볼륨 지원을 활성화합니다.

"vSphere IaaS 제어부 설치 및 구성" 설명서의 감독자에서 스토리지 설정 변경을 참조하십시오.

3 DevOps 엔지니어는 PVC `accessMode`를 `ReadWriteMany`로 설정하여 영구 볼륨을 프로비저닝합니다.

동일한 PVC로 여러 포드를 프로비저닝할 수 있습니다.

예:

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

vSphere IaaS control plane의 볼륨 확장

DevOps 엔지니어는 생성 후 영구 블록 볼륨을 확장할 수 있습니다. vSphere IaaS control plane에서 두 가지 유형의 클러스터 즉, 감독자 및 Tanzu Kubernetes Grid가 모두 오프라인 및 온라인 볼륨 확장을 지원합니다.

참고 영구 블록 볼륨만 확장할 수 있습니다. 현재 vSphere IaaS control plane는 ReadWriteMany 볼륨에 대한 볼륨 확장을 지원하지 않습니다.

vSphere IaaS control plane 환경에 나타나는 모든 스토리지 클래스는 기본적으로 `allowVolumeExpansion`이 `true`로 설정되어 있습니다. 이 매개 변수를 사용하면 오프라인 또는 온라인 볼륨의 크기를 수정할 수 있습니다.

볼륨이 노드 또는 포드에 연결되어 있지 않으면 오프라인으로 간주됩니다. 온라인 볼륨은 노드 또는 포드에서 사용할 수 있는 볼륨입니다.

볼륨 확장 기능에 대한 지원 수준은 vSphere 버전에 따라 다릅니다. 확장을 지원하는 적절한 버전으로 vSphere 환경을 업그레이드하는 경우 이전 버전의 vSphere에서 생성된 볼륨을 확장할 수 있습니다.

볼륨을 확장할 때는 다음 사항에 유의하십시오.

- 볼륨은 스토리지 할당량에 지정된 제한까지 확장할 수 있습니다. vSphere IaaS control plane는 영구 볼륨 할당 개체에 대한 연속적인 크기 조정 요청을 지원합니다.
- VMFS, vSAN, vSAN Direct, vVols 및 NFS를 포함한 모든 유형의 데이터스토어는 볼륨 확장을 지원합니다.
- 배포 또는 독립형 포드에 대한 볼륨 확장을 수행할 수 있습니다.
- 감독자 및 Tanzu Kubernetes Grid 클러스터에서 정적으로 프로비저닝된 볼륨의 크기를 조절할 수 있습니다(볼륨에 연결된 스토리지 클래스가 있는 경우).
- StatefulSet 정의를 사용할 때는 StatefulSet의 일부로 생성된 볼륨을 확장할 수 없습니다. 현재 Kubernetes는 이 기능을 지원하지 않습니다. 그 결과 StatefulSet 정의에서 스토리지 크기를 늘려 볼륨을 확장하려는 시도가 실패합니다.
- 볼륨을 지원하는 가상 디스크에 스냅샷이 있으면 크기를 조정할 수 없습니다.
- vSphere IaaS control plane는 인-트리(in-tree) 또는 마이그레이션된 볼륨에 대한 볼륨 확장을 지원하지 않습니다.

오프라인 모드에서 영구 볼륨 확장

볼륨이 노드 또는 포드에 연결되어 있지 않으면 오프라인으로 간주됩니다. 두 가지 유형의 클러스터 즉, 감독자 및 Tanzu Kubernetes Grid 클러스터는 모두 오프라인 볼륨 확장을 지원합니다.

사전 요구 사항

vSphere 환경을 오프라인 볼륨 확장을 지원하는 적절한 버전으로 업그레이드해야 합니다.

절차

1 스토리지 클래스를 사용하여 PVC(영구 볼륨 할당)를 생성합니다.

a 예를 들어 다음 YAML 매니페스트를 사용하여 PVC를 정의합니다.

이 예에서 요청된 스토리지의 크기는 1Gi입니다.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: example-block-pvc
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: example-block-sc
```

b Kubernetes 클러스터에 PVC를 적용합니다.

```
kubectl apply -f example-block-pvc.yaml
```

2 PVC에 패치를 적용하여 크기를 늘립니다.

PVC가 노드에 연결되어 있지 않거나 포드에서 사용 중이면 다음 명령을 사용하여 PVC에 패치를 적용합니다. 이 예에서 요청된 스토리지 증가량은 2Gi입니다.

```
kubectl patch pvc example-block-pvc -p '{"spec": {"resources": {"requests": {"storage":
"2Gi"}}}}'
```

이 작업은 PVC와 연결된 볼륨에서 확장을 트리거합니다.

3 볼륨의 크기가 증가했는지 확인합니다.

```
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
```

참고 PVC의 크기는 포드에서 PVC가 사용될 때까지 변경되지 않은 상태로 유지됩니다.

다음 예는 PVC 크기가 변경되지 않았음을 보여 줍니다. PVC를 설명하는 경우 PVC에 적용된 `FilesystemResizePending` 조건을 확인할 수 있습니다.

```
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 PVC를 사용하려면 포드를 생성합니다.

포드에서 PVC를 사용하는 경우 파일 시스템이 확장됩니다.

5 PVC의 크기가 수정되었는지 확인합니다.

```
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` 조건이 PVC에서 제거되었습니다. 볼륨 확장이 완료되었습니다.

다음에 수행할 작업

vSphere 관리자는 vSphere Client에서 새 볼륨 크기를 볼 수 있습니다. vSphere Client에서 영구 볼륨 모니터링의 내용을 참조하십시오.

온라인 모드에서 영구 볼륨 확장

온라인 볼륨은 노드 또는 포드에서 사용할 수 있는 볼륨입니다. DevOps 엔지니어는 온라인 영구 블록 볼륨을 확장할 수 있습니다. 두 가지 유형의 클러스터 즉, 감독자 및 Tanzu Kubernetes Grid 클러스터는 모두 온라인 볼륨 확장을 지원합니다.

사전 요구 사항

vSphere 환경을 온라인 볼륨 확장을 지원하는 적절한 버전으로 업그레이드해야 합니다.

절차

1 크기를 조정할 영구 볼륨 할당을 찾습니다.

```
$ 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
```

볼륨이 사용하는 스토리지 크기는 1Gi입니다.

2 PVC에 패치를 적용하여 크기를 늘립니다.

예를 들어 크기를 2Gi로 늘립니다.

```
$ kubectl patch pvc block-pvc -p '{"spec": {"resources": {"requests": {"storage":
"2Gi"}}}}'
persistentvolumeclaim/block-pvc edited
```

이 작업은 PVC와 연결된 볼륨에서 확장을 트리거합니다.

3 PVC와 PV의 크기가 모두 증가했는지 확인합니다.

```
$ 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
```

다음에 수행할 작업

vSphere 관리자는 vSphere Client에서 새 볼륨 크기를 볼 수 있습니다. vSphere Client에서 영구 볼륨 모니터링의 내용을 참조하십시오.

vSphere Client에서 영구 볼륨 모니터링

DevOps 엔지니어가 영구 볼륨 할당과 함께 상태 저장 애플리케이션을 배포할 때 vSphere IaaS control plane은 영구 볼륨 개체 및 일치하는 영구 가상 디스크를 생성합니다. vSphere 관리자는 vSphere Client에서 영구 볼륨의 세부 정보를 검토할 수 있습니다. 또한 해당 스토리지 규정 준수 상태와 영구 볼륨 상태를 모니터링할 수 있습니다.

절차

1 vSphere Client에서 영구 볼륨이 있는 네임스페이스로 이동합니다.

a vSphere Client 홈 메뉴에서 **워크로드 관리**를 선택합니다.

b **네임스페이스** 탭을 클릭하고 목록에서 네임스페이스를 선택합니다.

2 **스토리지** 탭을 클릭하고 **영구 볼륨 할당**을 클릭합니다.

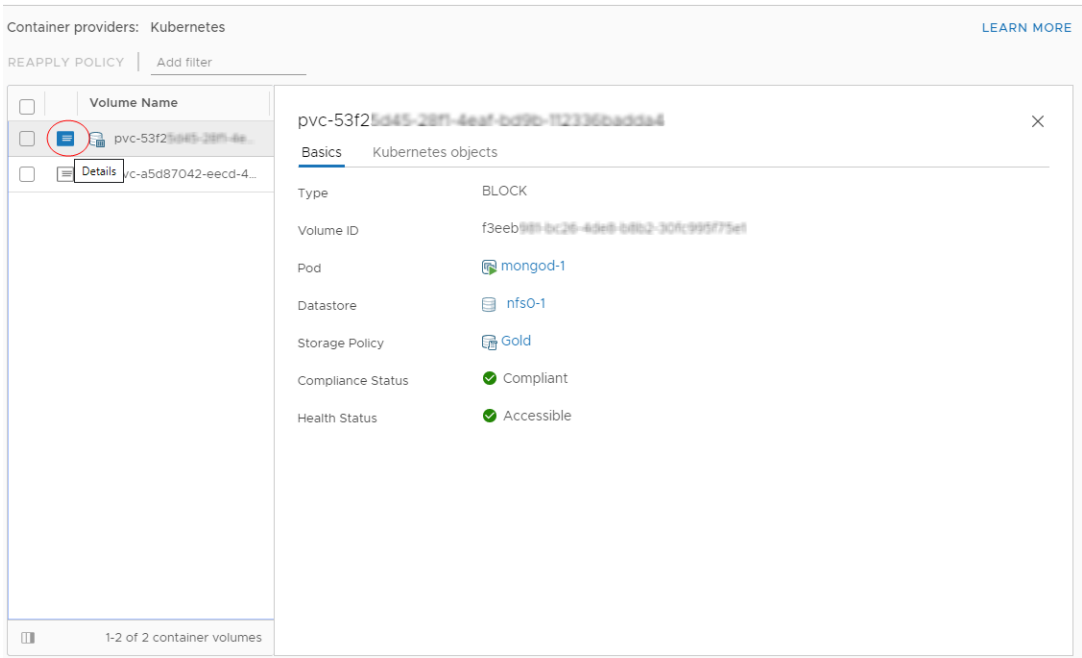
vSphere Client가 모든 영구 볼륨 할당 개체와 네임스페이스에서 사용할 수 있는 해당 볼륨을 나열합니다.

3 선택된 영구 볼륨 할당의 세부 정보를 보려면 **영구 볼륨 이름** 열에서 볼륨의 이름을 클릭합니다.

4 **컨테이너 볼륨** 페이지에서 볼륨의 상태와 스토리지 정책 규정 준수를 확인합니다.

a **세부 정보** 아이콘을 클릭하고 **기본** 및 **Kubernetes 개체** 탭 사이를 전환하여 Kubernetes 영구 볼륨에 대한 추가 정보를 봅니다.

`kubectl` 명령을 사용하여 볼륨 상태를 모니터링하려면 vSphere 네임스페이스 또는 Tanzu Kubernetes Grid 클러스터의 볼륨 상태 모니터링 항목을 참조하십시오.



b 볼륨의 상태를 확인합니다.

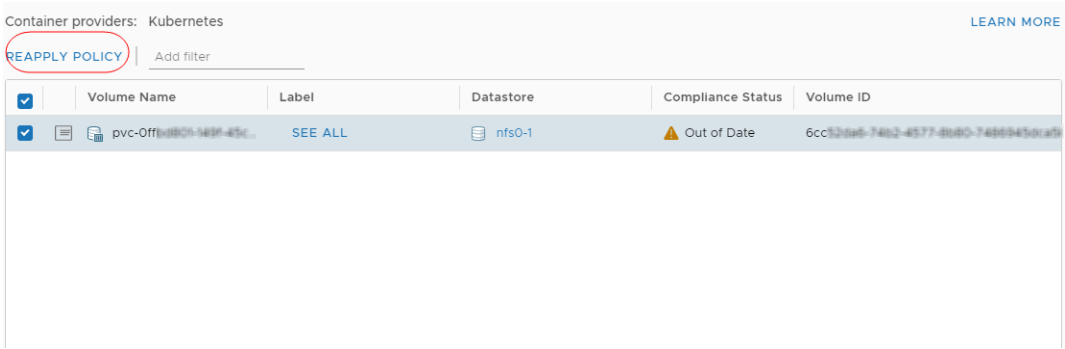
상태	설명
액세스 가능	영구 볼륨에 액세스할 수 있고 사용할 수 있습니다.
액세스할 수 없음	영구 볼륨에 액세스할 수 없고 사용할 수 없습니다. 데이터스토어에 연결하는 호스트가 볼륨을 저장하는 데이터스토어에 연결할 수 없으면 영구 볼륨에 액세스할 수 없게 됩니다.

c 스토리지 규정 준수 상태를 확인합니다.

규정 준수 상태 열에서 다음 중 하나를 볼 수 있습니다.

규정 준수 상태	설명
준수	볼륨 백업 가상 디스크가 상주하는 데이터스토어가 정책에 필요한 스토리지 기능을 갖추고 있습니다.
최신 버전이 아님	이 상태는 정책이 편집되었지만 데이터스토어에 새 스토리지 요구 사항이 전달되지 않았음을 나타냅니다. 변경 사항을 전달하려면 최신 상태가 아닌 볼륨에 정책을 다시 적용합니다.
비준수	데이터스토어가 지정된 스토리지 요구 사항을 지원하지만 현재는 스토리지 정책을 충족할 수 없습니다. 예를 들어 데이터스토어의 물리적 리소스를 사용할 수 없는 경우 상태가 비준수가 될 수도 있습니다. 호스트나 디스크를 클러스터에 추가하는 방법 등으로 호스트 클러스터의 물리적 구성을 변경하면 데이터스토어를 준수 상태로 만들 수 있습니다. 추가적인 리소스가 스토리지 정책을 충족하면 상태가 [준수]로 변경됩니다.
해당 없음	스토리지 정책이 데이터스토어에서 지원되지 않는 데이터스토어 기능을 참조합니다.

d 규정 준수 상태가 [최신 버전이 아님]인 경우 볼륨을 선택하고 **정책 다시 적용**을 클릭합니다.



상태가 [준수]로 변경됩니다.

vSphere 네임스페이스 또는 Tanzu Kubernetes Grid 클러스터의 볼륨 상태 모니터링

vSphere IaaS control plane를 사용하면 바인딩된 상태에서 영구 볼륨의 상태를 확인할 수 있습니다.

바인딩된 상태의 각 영구 볼륨에 대한 상태는 영구 볼륨에 바인딩된 영구 볼륨 클레임의 `Annotations: volumehealth.storage.kubernetes.io/messages:` 필드에 표시됩니다. 가능한 상태 값에는 두 가지가 있습니다.

상태	설명
액세스 가능	영구 볼륨에 액세스할 수 있고 사용할 수 있습니다.
액세스할 수 없음	영구 볼륨에 액세스할 수 없고 사용할 수 없습니다. 데이터스토어에 연결하는 호스트가 볼륨을 저장하는 데이터스토어에 연결할 수 없으면 영구 볼륨에 액세스할 수 없게 됩니다.

vSphere Client에서 볼륨 상태를 모니터링하려면 vSphere Client에서 영구 볼륨 모니터링 항목을 참조하십시오.

절차

1 vSphere IaaS control plane 환경의 네임스페이스에 액세스합니다.

2 영구 볼륨 할당을 생성합니다.

a 영구 볼륨 할당 구성을 포함하는 YAML 파일을 생성합니다.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: gold
  resources:
    requests:
        storage: 2Gi
```

b Kubernetes 클러스터에 영구 볼륨 할당을 적용합니다.

```
kubectl apply -f pvc_name.yaml
```

이 명령은 할당의 스토리지 요구 사항을 충족하는 백업 가상 디스크가 있는 Kubernetes 영구 볼륨과 vSphere 볼륨을 생성합니다.

c 영구 볼륨 할당이 볼륨에 바인딩되어 있는지 확인합니다.

```
kubectl get pvc my-pvc
```

출력은 영구 볼륨 할당 및 볼륨이 바인딩된 상태임을 표시합니다.

```
NAME     STATUS    VOLUME   CAPACITY   ACCESSMODES   STORAGECLASS   AGE
my-pvc   Bound     my-pvc   2Gi        RWO           gold           30s
```

3 볼륨의 상태를 확인합니다.

다음 명령을 실행하여 영구 볼륨에 바인딩된 영구 볼륨 클레임의 볼륨 상태 주석을 확인합니다.

```
kubectl describe pvc my-pvc
```

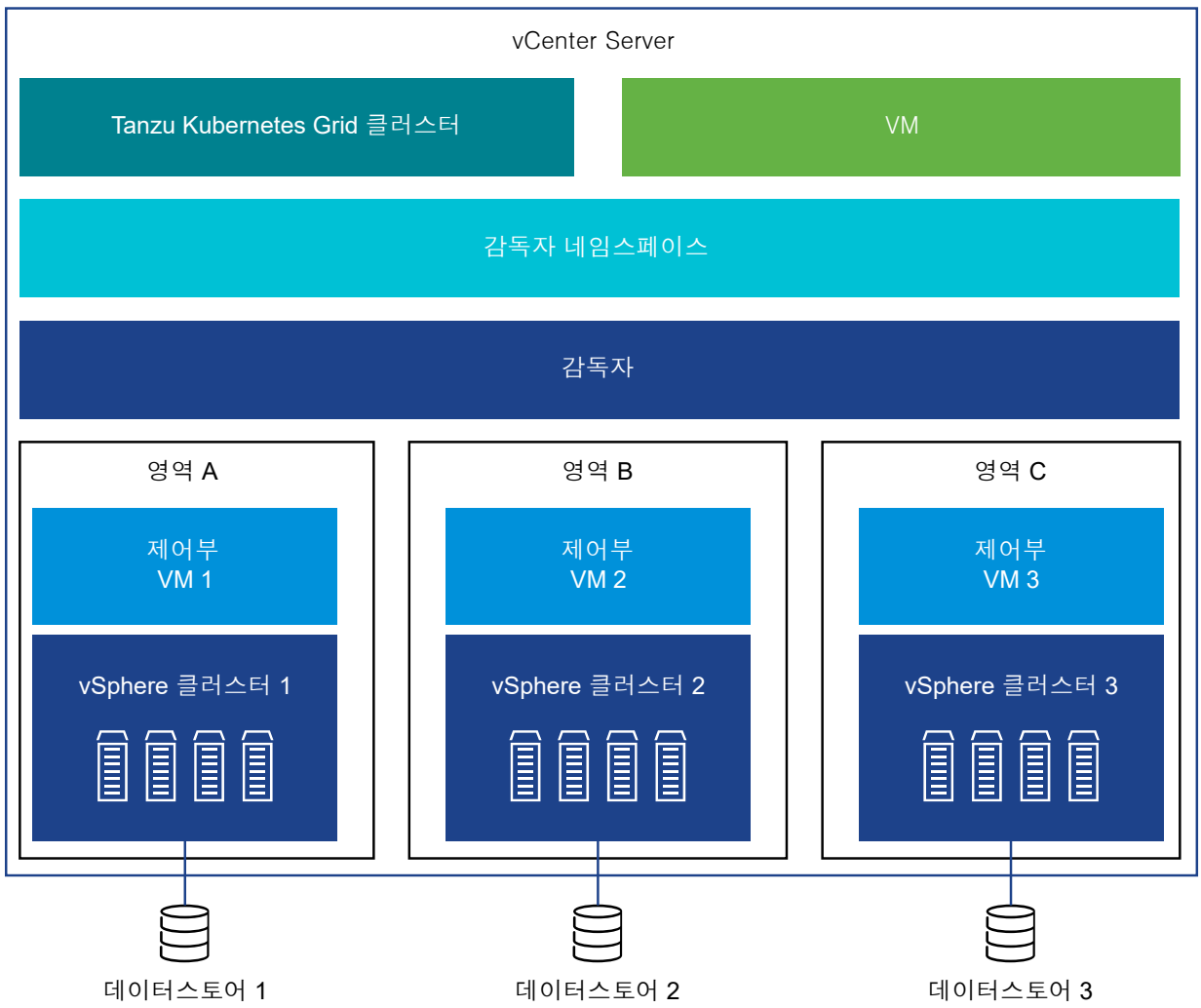
다음 샘플 출력에서 `volumehealth.storage.kubernetes.io/messages` 필드에 상태가 액세스 가능한 것으로 표시됩니다.

```
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
```

3개 영역 감독자에서 영구 스토리지를 사용하는 모범 사례

vSphere IaaS control plane의 3개 영역 감독자는 데이터스토어가 단일 영역의 모든 호스트에서 공유되는 영역 스토리지를 지원합니다.



3개 영역 감독자에 대한 스토리지 리소스를 준비할 때는 다음 사항을 고려하십시오.

- 세 영역 모두의 스토리지 유형이 같을 필요는 없습니다. 단, 세 클러스터 모두에 통일된 스토리지를 사용하면 일관된 성능을 얻을 수 있습니다.

- 3개 영역 감독자의 네임스페이스의 경우 각 클러스터의 공유 스토리지 규정을 준수하는 스토리지 정책을 사용합니다. 스토리지 정책은 토폴로지를 인식해야 합니다.
- 네임스페이스에 할당한 후 스토리지 정책에서 토폴로지 제약 조건을 제거하지 마십시오.
- 영역 데이터스토어를 다른 영역에 마운트하지 마십시오.
- 3개 영역 감독자는 다음 항목을 지원하지 않습니다.
  - 교차 영역 볼륨
  - vSAN 파일 볼륨(ReadWriteMany 볼륨)
  - Register Volume API를 사용한 정적 볼륨 프로비저닝
  - vSAN 데이터 지속성 플랫폼을 사용하는 워크로드
  - vSphere 포드
  - vSAN 확대 클러스터
  - vGPU 및 인스턴스 스토리지가 있는 VM

3개 영역 감독자에 대한 스토리지 정책 생성

영구 스토리지를 사용할 수 있으려면 3개 영역 감독자에서 실행되는 워크로드가 영역 토폴로지를 사용하는 스토리지 클래스에 액세스할 수 있어야 합니다. 이러한 스토리지 클래스를 사용할 수 있도록 하기 위해 vSphere 관리자는 토폴로지 인식 스토리지 정책을 생성하여 네임스페이스에 할당합니다.

3개 영역 감독자의 네임스페이스는 토폴로지를 인식하지 않는 스토리지 정책을 할당하지 못하도록 합니다.

3개 영역 감독자를 사용하도록 설정하는 방법에 대한 자세한 내용은 3개 영역 감독자 사용을 참조하십시오.

절차

1 vSphere Client에서 **VM 스토리지 정책 생성** 마법사를 엽니다.

   a **홈** 메뉴에서 **정책 및 프로파일**을 클릭합니다.

   b **정책 및 프로파일**에서 **VM 스토리지 정책**을 클릭합니다.

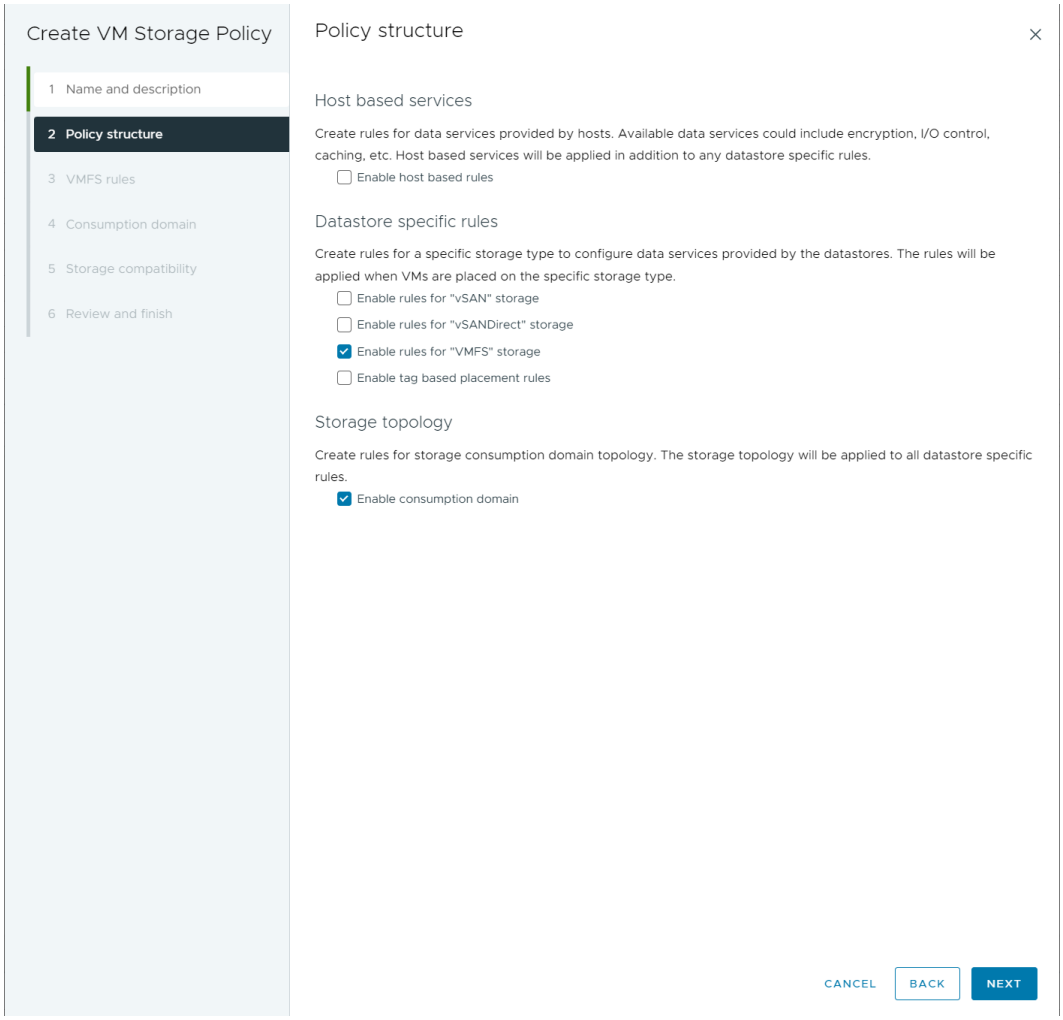
   c **생성**을 클릭합니다.

2 정책 이름 및 설명을 입력합니다.

옵션	작업
vCenter Server	vCenter Server 인스턴스를 선택합니다.
이름	스토리지 정책의 이름을 입력합니다.
설명	스토리지 정책의 설명을 입력합니다.

3 **정책 구조** 페이지의 메시지를 따릅니다.

4 **스토리지 토폴로지**에서 **사용 도메인 사용**을 선택하고 **사용 도메인** 페이지의 메시지를 따릅니다.



5 **사용 도메인** 페이지에서 스토리지 토폴로지 유형을 지정합니다.

옵션	설명
영역	단일 영역의 모든 호스트에서 데이터스토어가 공유됩니다.

3개 영역 감독자에서 PVC 생성

3개 영역 감독자에서 동적 PVC를 생성할 때 볼륨을 프로비저닝할 영역을 지정할 수 있습니다.

절차

- PVC 영역 배치를 제어하려면 PVC YAML 파일에서 Kubernetes `csi.vsphere.volume-requested-topology` 주석을 사용합니다.

  경고 이 매개 변수는 감독자에서 직접 PVC를 생성할 때 필요합니다. 그러나 Tanzu Kubernetes Grid 클러스터에 대해 생성하는 PVC에는 영역 주석을 포함하지 마십시오. 그렇게 하면 PVC가 작동하지 않습니다.

```
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
```

  3개 영역을 모두 지정하면 볼륨이 영역 1, 영역 2 또는 영역 3중 하나에 생성됩니다.

다음에 수행할 작업

Tanzu Kubernetes Grid 클러스터에 상태 저장 애플리케이션을 배포하는 방법에 대한 자세한 내용은 지연 바인딩 볼륨 연결로 여러 vSphere 영역에 StatefulSet 애플리케이션 배포를 참조하십시오.

vSphere IaaS control plane에서 Harbor 및 Contour 설치 및 구성

9

vSphere IaaS control plane 환경에서 Harbor 및 Contour를 감독자 서비스로 배포하고 구성하는 방법을 알아봅니다. Harbor는 vSphere IaaS control plane에서 실행되는 워크로드에 사용할 수 있는 오픈 소스 클라우드 네이티브 레지스트리입니다. Contour는 엔보이 프록시를 역방향 프록시 및 로드 밸런서로 배포하여 작동하는 Kubernetes용 수신 컨트롤러입니다. Contour는 경량 프로파일을 유지하면서 즉시 사용 가능한 동적 구성 업데이트를 지원합니다.

Contour를 애플리케이션의 수신 컨트롤러로 감독자 서비스로 사용할 수 있습니다. Contour는 Harbor 감독자 서비스를 실행하기 위한 요구 사항이기도 합니다.

참고 감독자 서비스는 VDS 또는 NSX 네트워킹 스택에서 실행되는 단일 클러스터 감독자에서 지원됩니다. 3개 영역 감독자에는 감독자 서비스를 배포할 수 없습니다.

감독자 서비스인 Harbor는 다음과 같은 기능 및 특징을 제공합니다.

- Harbor 오픈 소스 레지스트리의 최신 버전.
- 관리자 및 루트 계정으로 Harbor에 액세스합니다.
- 업스트림 Harbor 레지스트리를 사용한 전체 기능 패리티.
- DNS를 사용하여 수신(Contour)을 통해 Harbor에 액세스합니다.

참고 배포되면 Harbor 및 Contour 감독자 서비스는 둘 다 이러한 서비스용으로 생성된 vSphere 네임스페이스에 vSphere 포드를 생성합니다. 이러한 vSphere 포드는 서비스를 운영하려면 필요합니다. VDS 네트워킹 스택에서 실행되는 감독자 또는 3개 영역 감독자의 감독자 서비스 외부에서 vSphere 포드를 배포할 수 없습니다. NSX와 함께 배포된 단일 클러스터 감독자에서는 일반 용도로만 vSphere 포드를 배포할 수 있습니다.

다음으로 아래 항목을 읽으십시오.

- vSphere IaaS control plane에서 Contour를 감독자 서비스로 설치
- vSphere IaaS control plane의 감독자에 Harbor 설치 및 구성
- vSphere IaaS control plane에서 내장된 레지스트리의 이미지를 Harbor로 마이그레이션

vSphere IaaS control plane에서 Contour를 감독자 서비스로 설치

vSphere IaaS control plane 환경의 감독자에 Contour를 감독자 서비스로 설치하는 방법을 알아봅니다. 설치한 후에는 Contour를 애플리케이션의 수신 컨트롤러로 사용할 수 있습니다. Contour는 Harbor를 감독자 서비스로 실행하기 위한 요구 사항이기도 합니다.

사전 요구 사항

- 서비스를 추가하는 vCenter Server 시스템에 대한 **감독자 서비스 관리** 권한이 있는지 확인합니다.
- vCenter Server 8.0a 이상으로 업그레이드했는지 확인합니다. Contour 및 Harbor 감독자 서비스는 vCenter Server 8.0a 이상에서 지원됩니다.

절차

1 Supervisor-Services 저장소의 Contour Versions 섹션으로 이동하여 다음 파일을 다운로드합니다.

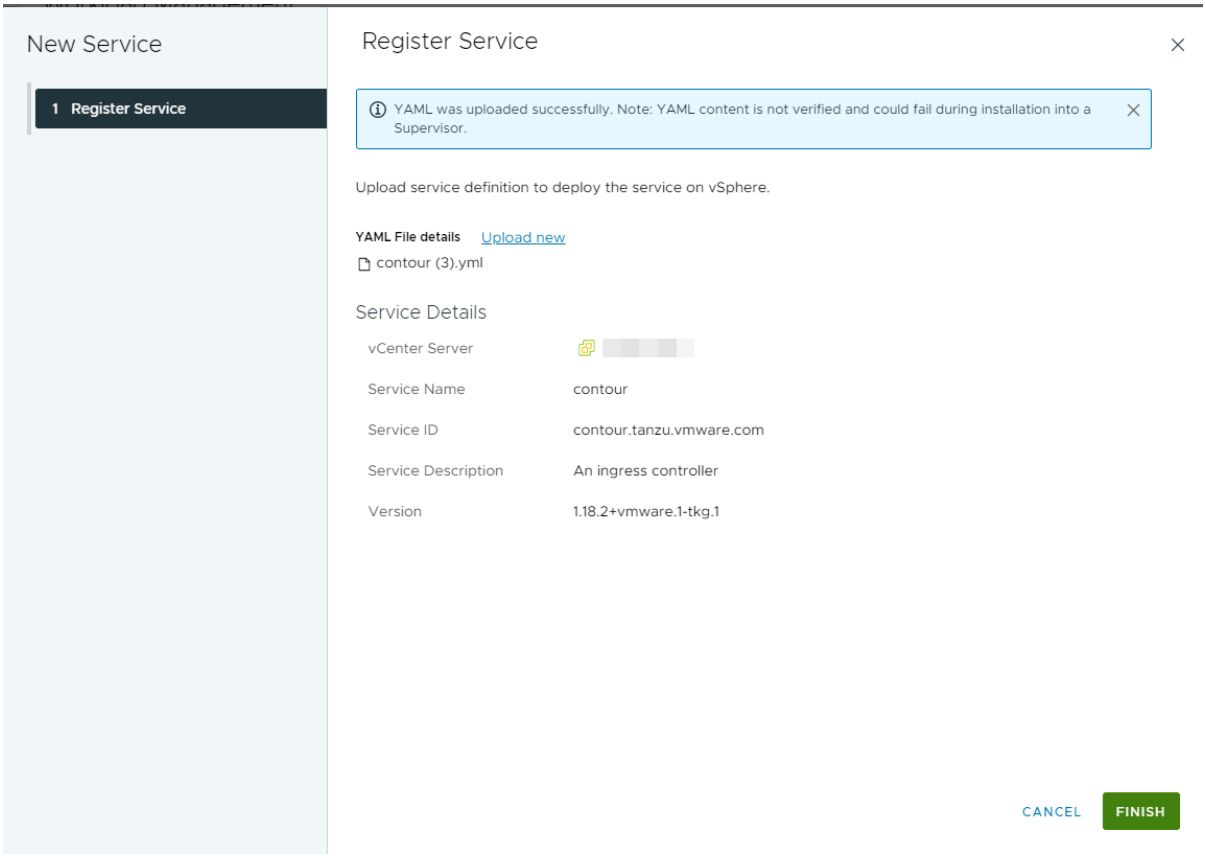
   - Contour 서비스 정의, 링크 이름은 `Contour vX.X.X.`입니다. 예: `Contour 1.18.2`
   - Contour 구성 파일, 링크 이름은 `values for vX.X.X`입니다. 예: `values 1.18.2`

   결과 파일은 다음과 같습니다.

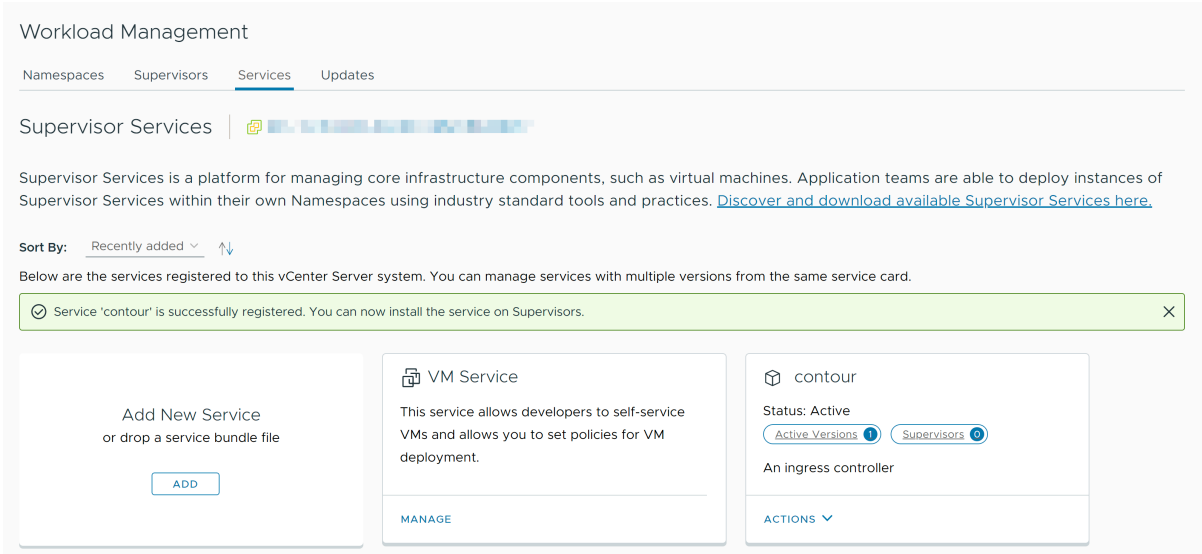
   - `contour.yml`
   - `contour-data-values.yml`

2 vSphere Client에서 **워크로드 관리**로 이동하고 **서비스**를 선택합니다.

3 **새 서비스 추가**를 클릭하고 `harbor.yml` 서비스 정의를 업로드하여 Contour의 서비스 운영자를 배포합니다.



Contour Operator가 성공적으로 배포되면 서비스 카드가 **서비스** 탭에 나타납니다.



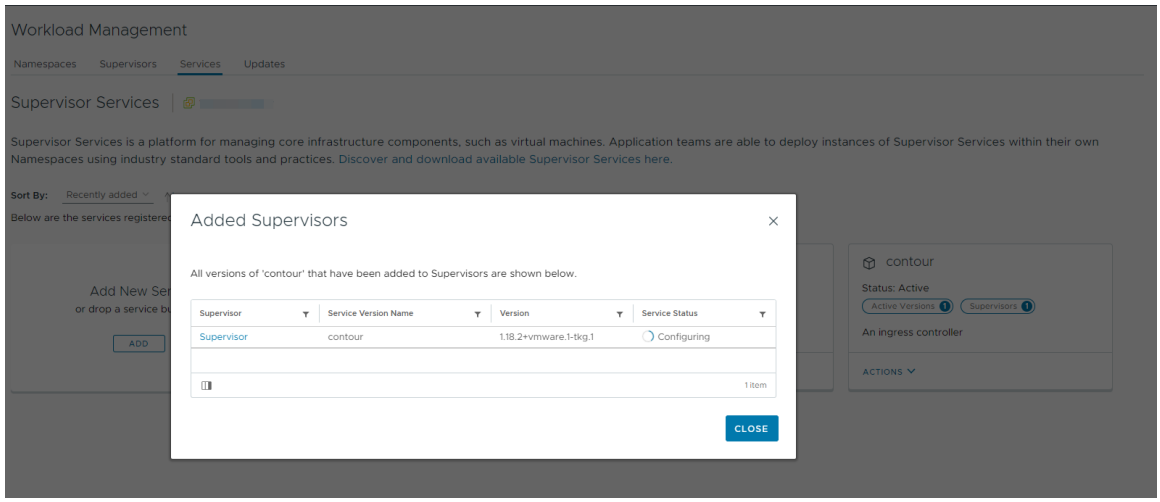
4 Contour Operator가 배포되었으니 감독자에 감독자 서비스를 설치할 수 있습니다

a **Contour** 서비스 카드에서 **작업 > 감독자에 설치**를 선택합니다.

b 감독자를 선택하고 **YAML 서비스 구성**에서 기본값을 변경하지 않고 `contour-data-values.yml` 파일의 컨텐츠를 복사하여 붙여넣습니다.

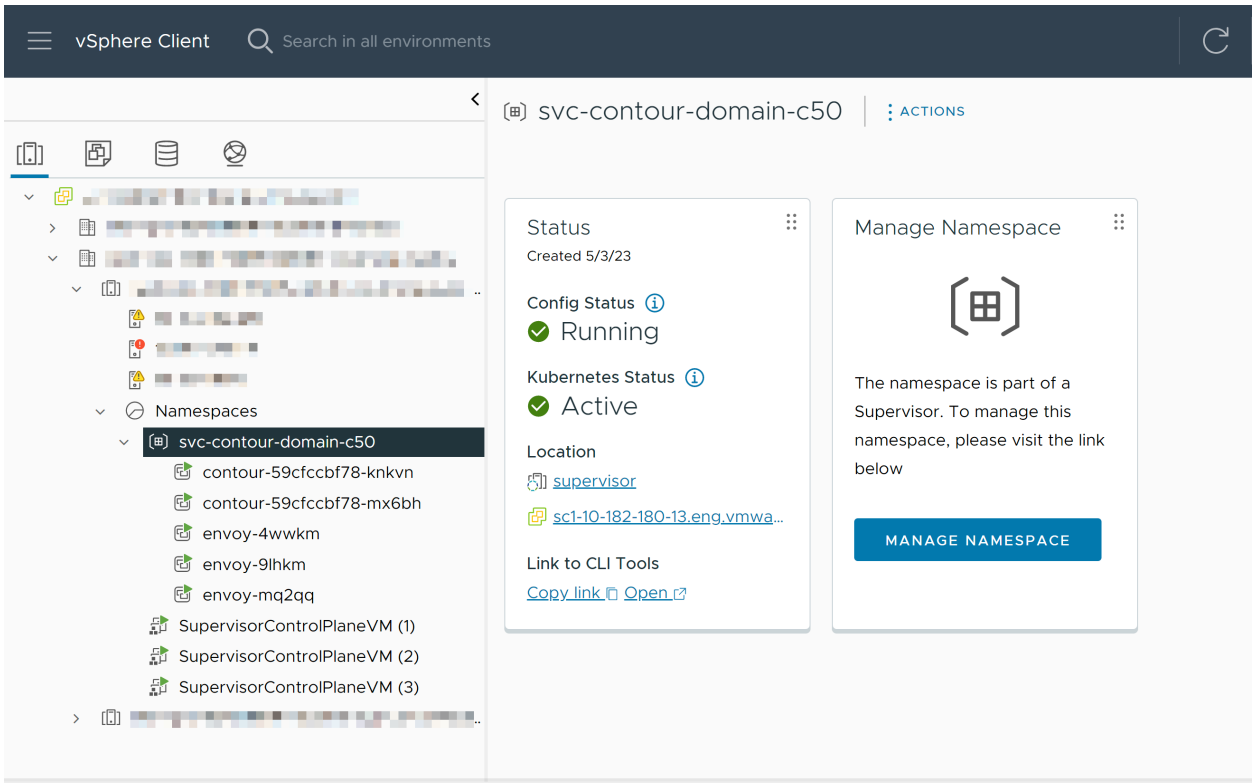
c **확인**을 클릭합니다.

설치가 시작되면 Contour 서비스 카드에서 **감독자** 필드를 클릭하여 설치를 추적할 수 있습니다. **감독자** 옆에 있는 숫자가 증가할 때까지 몇 초 정도 걸릴 수 있습니다. 서비스는 원하는 상태에 도달할 때까지 [구성 중] 상태입니다. 원하는 상태에 도달하면 서비스의 상태가 [실행 중]으로 변경됩니다.

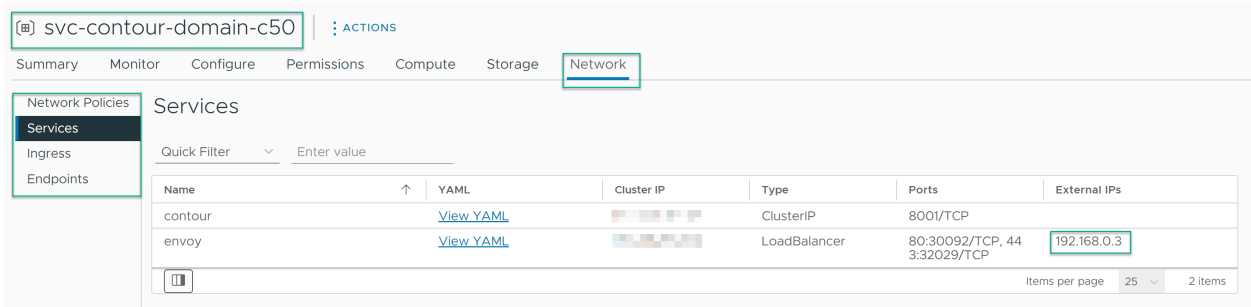


결과

Contour가 설치되면 서비스 인스턴스에 대해 생성된 vSphere 네임스페이스는 물론 해당 vSphere 포드가 배포됩니다.



감독자로 구성된 외부 DNS 서버의 도메인 이름에 매핑할 수 있는 엔보이 서비스의 IP 주소를 볼 수도 있습니다. 매핑을 사용하여 Contour를 통해 애플리케이션에 수신을 제공할 수 있습니다. Contour vSphere 네임스페이스의 **네트워크** 옵션에서 엔보이 IP 주소를 볼 수 있습니다:



다음에 수행할 작업

Harbor를 감독자 서비스로 사용하여 워크로드에 대한 레지스트리를 제공하려는 경우 서비스를 설치하고 워크로드에 사용하도록 구성할 수 있습니다. vSphere IaaS control plane의 감독자에 Harbor 설치 및 구성의 내용을 참조하십시오.

vSphere IaaS control plane의 감독자에 Harbor 설치 및 구성

Harbor를 감독자 서비스로 설치하고 구성하는 방법을 알아봅니다. 그런 다음 Harbor를 Tanzu Kubernetes Grid 클러스터 및 vSphere 포드에서 실행되는 워크로드에 대한 레지스트리로 사용할 수 있습니다. Harbor에는 Contour가 수신 컨트롤러로 필요하므로 먼저 Contour 감독자 서비스를 설치한 다음 Harbor를 설치합니다.

Harbor를 감독자 서비스로 설치

vSphere Client의 **워크로드 관리** 옵션을 통해 Harbor를 감독자 서비스로 설치합니다.

사전 요구 사항

- vCenter Server 8.0a 이상으로 업그레이드했는지 확인합니다. Contour 및 Harbor 감독자 서비스는 vCenter Server 8.0a 이상에서 지원됩니다.
- 서비스를 추가하는 vCenter Server 시스템에 대한 **감독자 서비스 관리** 권한이 있는지 확인합니다.
- Harbor를 설치하려는 동일한 감독자에 Contour를 감독자 서비스로 설치합니다. vSphere IaaS control plane에서 Contour를 감독자 서비스로 설치의 내용을 참조하십시오.
- Harbor 관리 UI에 액세스하기 위한 FQDN을 지정합니다.

절차

1 Supervisor-Services 저장소의 Harbor Versions 섹션으로 이동하여 다음 파일을 다운로드합니다.

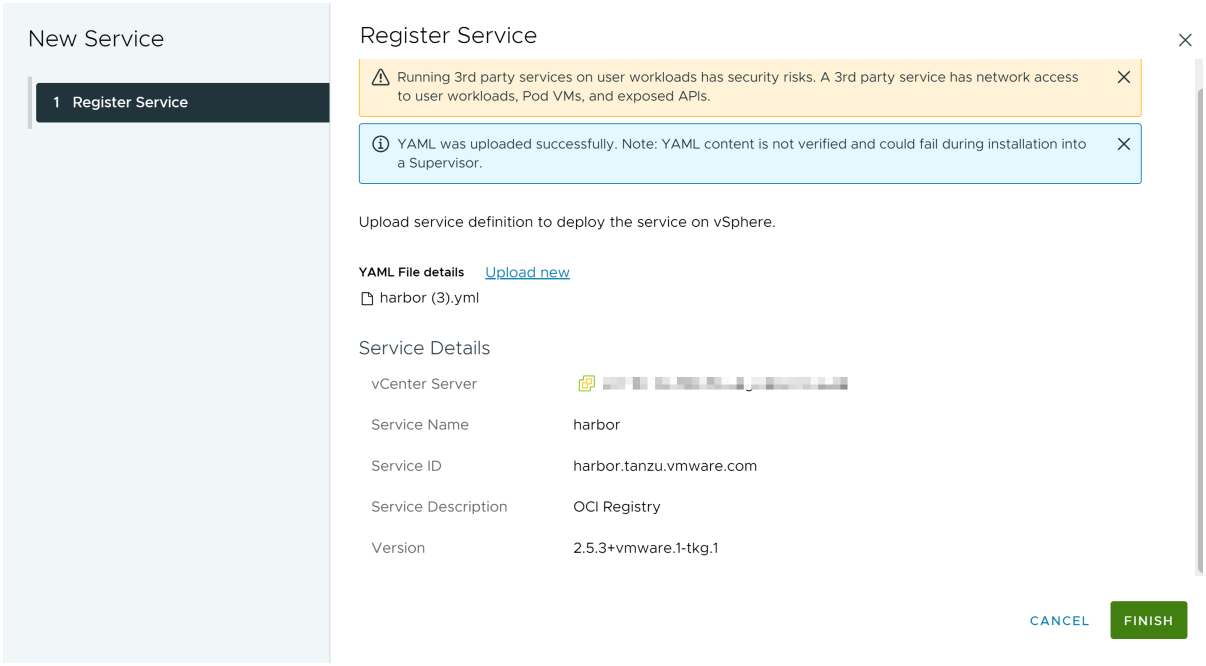
   - Harbor 서비스 정의, 링크 이름은 `Harbor vX.X.X.`입니다. 예: `Harbor 2.5.3`
   - Harbor 구성 파일, 링크 이름은 `values for vX.X.X`입니다. 예: `values 2.5.3`

   결과 파일은 다음과 같습니다.

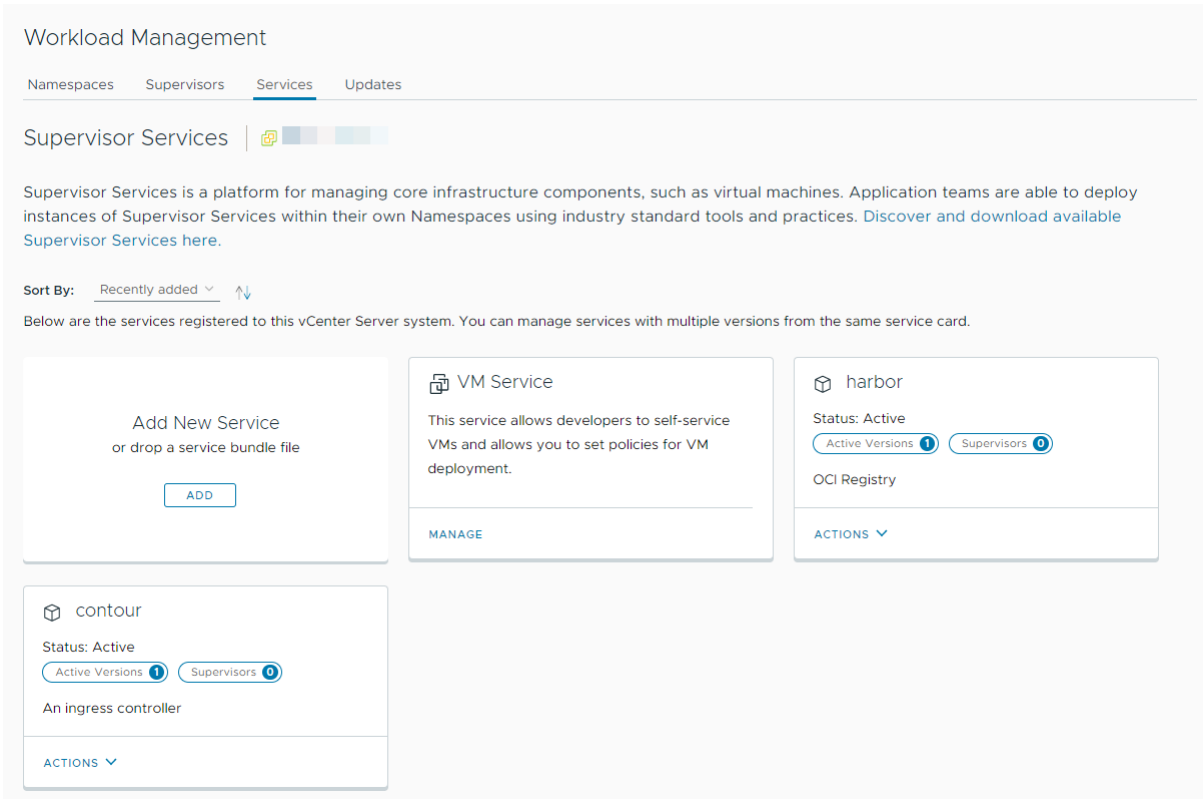
   - `harbor.yml`
   - `harbor-data-values.yml`

2 vSphere Client에서 **워크로드 관리**로 이동하고 **서비스**를 선택합니다.

3 **새 서비스 추가**를 클릭하고 `harbor.yml` 서비스 정의를 업로드하여 Harbor operator를 배포합니다.



Harbor operator가 배포되면 **서비스** 탭에 나타납니다.



4 Harbor operator가 배포되었으므로 Contour가 실행되는 동일한 감독자에 감독자 서비스를 설치할 수 있습니다.

a `harbor-data-values.yml` 파일을 열고 필요에 따라 속성을 편집합니다.

속성	값	설명

`hostname: myharbor.com`
 	
`https: 443`
	FQDN	Harbor 관리 UI에 액세스하도록 지정한 FQDN으로 변경합니다.

`tlsCertificate:`
`tlsSecretLabels:`
`{"managed-by": "vmware-vRegistry"}`
	참고 변경 없음	이 값은 TKG 통합이 작동하는 데 필요합니다.

`harborAdminPassword:` `Harbor12345`	필요에 따라 변경	설치 중에 사용되는 Harbor 암호입니다. 서비스가 설치되면 Harbor 관리 UI를 통해 변경할 수 있습니다.
`secretKey:` `0123456789ABCDEF`	16자 문자열	암호화에 사용되는 비밀 키입니다. 16자의 문자열이어야 합니다.
`database:` `password: change-it`	보안 암호	Postgres 데이터베이스에 사용되는 초기 암호입니다.

`core:`
`replicas:`
`secret: change-it`
`xsrfKey:`
`0123456789ABCDEF0123456789ABCDEF`
`jobservice:`
`replicas: 1`
`secret: change-it`
`registry:`
`replicas:`
`secret: change-it`
	암호 문자열 및 32자 XSRF 키 문자열	자체 암호를 설정하도록 변경합니다.

`persistence:`
`persistentVolumeClaim:`
`registry:`
`storageClass:`
`"insert-storage-class-name-here"`
`subPath: ""`
`accessMode:`
`ReadWriteOnce`
`size: 10Gi`
`jobservice:`
`storageClass:`
`"insert-storage-class-name-here"`
`subPath: ""`
`accessMode:`
`ReadWriteOnce`
`size: 1Gi`
	스토리지 클래스 이름	Harbor 레지스트리, 작업 서비스, 데이터베이스 등에서 PVC 프로비저닝을 위한 스토리지 클래스로 사용될 스토리지 정책입니다.
		각 속성을 환경에서 사용 가능한 기존 스토리지 정책으로 설정합니다. 모든 대문자를 소문자로 바꾸고 "_" 기호와 공백을 모두 대시 "-"로 바꾸어 스토리지 정책 이름을 유효한 스토리지 클래스 이름으로 변경합니다. 예를 들어 **`Harbor Storage Policy`**를 **`harbor-storage-policy`**로 수정합니다.

속성	값	설명
`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`		
`network: ipFamilies: ["IPv4"]`	**참고** 변경 없음	IPv6은 지원되지 않습니다.

b **워크로드 관리 > 서비스**로 돌아가고 **Harbor** 서비스 카드에서 **작업 > 감독자에 설치**를 선택합니다.

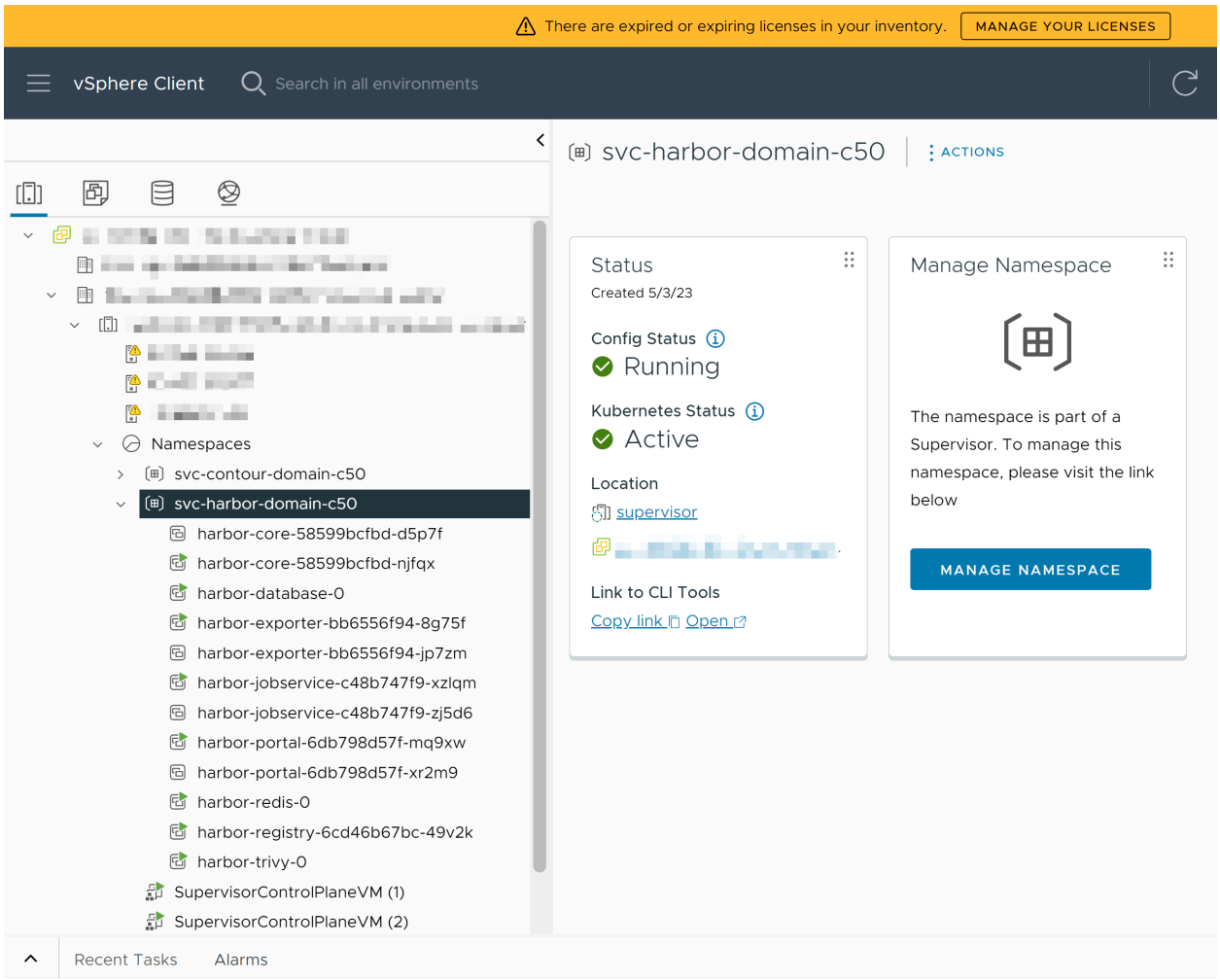
c Contour가 실행되는 감독자를 선택하고 **YAML 서비스 구성**에서 수정된 `harbor-data-values.yml` 파일의 컨텐츠를 복사하여 붙여넣습니다.

d **확인**을 클릭합니다.

설치가 시작되면 Harbor 서비스 카드에서 **감독자** 필드를 클릭하여 설치를 추적할 수 있습니다. **감독자** 옆에 있는 숫자가 증가할 때까지 몇 초 정도 걸릴 수 있습니다. 서비스는 원하는 상태에 도달할 때까지 [구성 중] 상태입니다. 원하는 상태에 도달하면 서비스의 상태가 [실행 중]으로 변경됩니다.

결과

호스트 및 클러스터 보기에서 Harbor에 대해 생성된 vSphere 네임스페이스 및 vSphere 포드를 볼 수 있습니다.

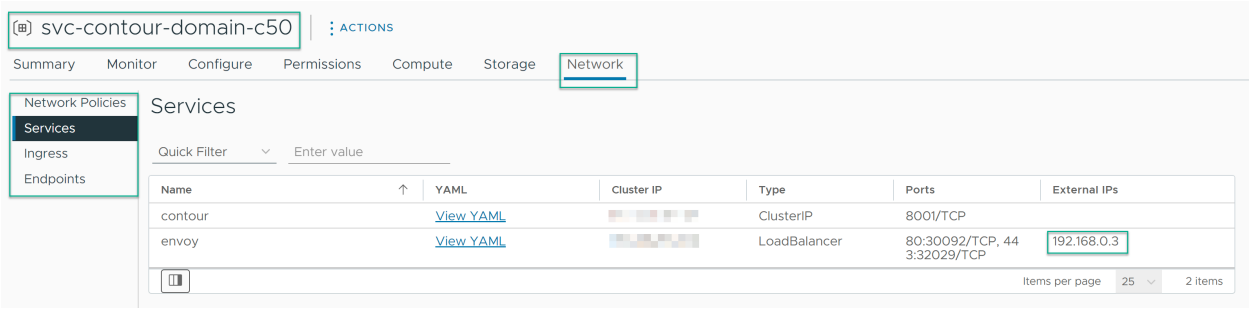


Harbor FQDN을 엔보이 수신 IP 주소에 매핑

Harbor가 설치된 후 엔보이 수신 IP 주소에 대한 Harbor FQDN 매핑 레코드를 감독자로 구성된 외부 DNS 서버에 포함합니다.

Tanzu Kubernetes Grid 클러스터, vSphere 포드 및 감독자는 Harbor FQDN을 확인할 수 있어야 레지스트리에서 이미지를 끌어올 수 있습니다.

엔보이 수신 IP 주소를 찾으려면 Contour 네임스페이스로 이동하여 **네트워크**를 선택하고 **서비스**를 선택합니다.



Harbor 감독자 서비스와의 신뢰 설정

Harbor가 설치되면 Harbor를 vSphere 포드의 레지스트리로 사용할 수 있도록 감독자와 Harbor 간에 신뢰를 구성해야 합니다. Harbor와 동일한 감독자에 있는 Tanzu Kubernetes Grid 클러스터는 Harbor와의 신뢰가 자동으로 설정되어 있습니다. Harbor를 다른 감독자에서 실행되는 Tanzu Kubernetes Grid 클러스터의 레지스트리로 사용하려면 Harbor와 이러한 Tanzu Kubernetes Grid 클러스터 간에 신뢰를 구성해야 합니다.

Harbor와 감독자 간에 신뢰 설정

Harbor와 감독자 간에 신뢰를 설정하려면 다음을 수행합니다.

1 Harbor UI에서 또는 감독자 제어부에서 TLS 암호를 사용하여 Harbor CA를 추출합니다. Harbor 관리 UI(**관리 > 구성 > 레지스트리 루트 인증서 > 다운로드**)에서 Harbor `ca.cert`를 가져올 수 있습니다.

2 `kube-system` 네임스페이스의 `image-fetcher-ca-bundle` ConfigMap에 Harbor CA를 추가합니다. vCenter Single Sign-on 관리 계정으로 로그인해야 하며 `image-fetcher-ca-bundle`을 편집할 수 있는 권한이 있어야 합니다.

   a KUBE_EDITOR 환경 변수를 여기에 설명된 대로 구성합니다.

   b 다음 명령을 사용하여 ConfigMap을 편집합니다.

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

   c Harbor `ca.cert` 파일의 컨텐츠를 기존 감독자 인증서 아래의 ConfigMap에 추가합니다. 감독자 인증서를 변경하지 않아야 합니다.

   ```
   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 파일에 대한 편집 내용을 저장합니다. 그 결과 kubectl 보고서:

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

Harbor 및 Harbor와 다른 감독자에서 실행되는 Tanzu Kubernetes Grid 클러스터 간에 신뢰 설정

Harbor가 설치된 것과 다른 감독자에서 실행되는 Tanzu Kubernetes Grid 클러스터는 Harbor와의 네트워크 연결이 있어야 합니다. 이러한 Tanzu Kubernetes Grid 클러스터는 Harbor FQDN을 확인할 수 있어야 합니다.

Harbor와 Tanzu Kubernetes Grid 클러스터 간에 신뢰를 설정하려면 Harbor UI에서 또는 감독자 제어부에서 TLS 암호를 사용하여 Harbor CA를 추출한 다음 TKG 2 클러스터를 개인 컨테이너 레지스트리와 통합에 나열된 단계를 따르십시오.

vSphere IaaS control plane에서 내장된 레지스트리의 이미지를 Harbor로 마이그레이션

감독자에서 내장된 Harbor 레지스트리를 사용하는 경우 내장된 레지스트리의 이미지를 감독자 서비스로 설치한 Harbor 레지스트리로 마이그레이션할 수 있습니다.

사전 요구 사항

- Contour 및 Harbor 감독자 서비스가 감독자에 설치되어 있는지 확인합니다.
- 감독자에서 사용하는 DNS에 엔보이 서비스 수신 IP에 매핑된 Harbor FQDN의 항목이 포함되어 있는지 확인합니다.
- 감독자와 Harbor 간에 신뢰가 설정되어 있는지 확인합니다. Harbor가 실행되는 것과 다른 감독자에서 실행되는 Tanzu Kubernetes Grid 클러스터에서 이미지를 참조하는 경우 이러한 Tanzu Kubernetes Grid 클러스터와 Harbor 간에 신뢰가 있는지 확인합니다.

절차

1 감독자에 vCenter Single-Sign-On 사용자로 로그인합니다.

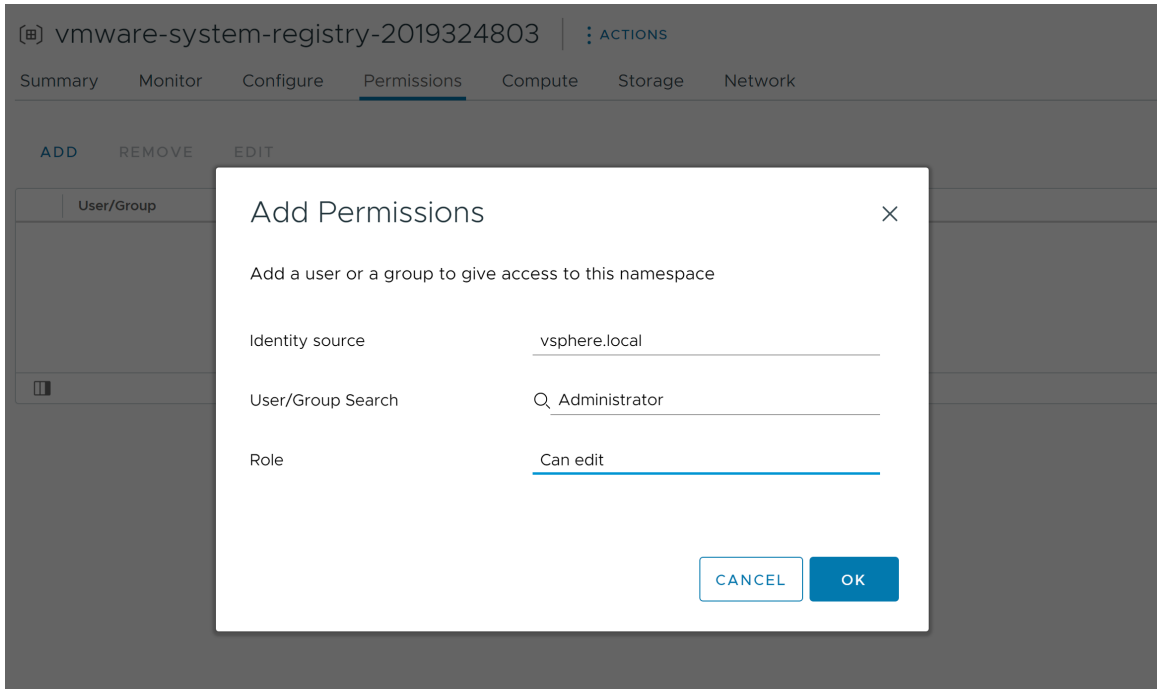
2 Harbor 감독자 서비스에 대한 네트워크 액세스 송신을 설정합니다.

a Harbor의 서비스 네임스페이스(예: `svc-harbor-domain-c9`)에 `allow-all-egress-harbor-supervisor-service`라는 네트워크 정책 CRD를 생성합니다.

```
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 내장된 레지스트리의 암호에 액세스합니다. 그래야 나중에 레지스트리를 Harbor에 대한복제 끝점으로 추가할 수 있습니다.

a 내장된 레지스트리 네임스페이스(예: `vmware-system-registry-437393318`)의 관리 vCenter Single Sign-On 사용자에게 편집 권한을 부여합니다.



b 내장된 레지스트리 네임스페이스에서 암호에 액세스합니다.

```
# 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 사용자 이름과 암호를 디코딩합니다.

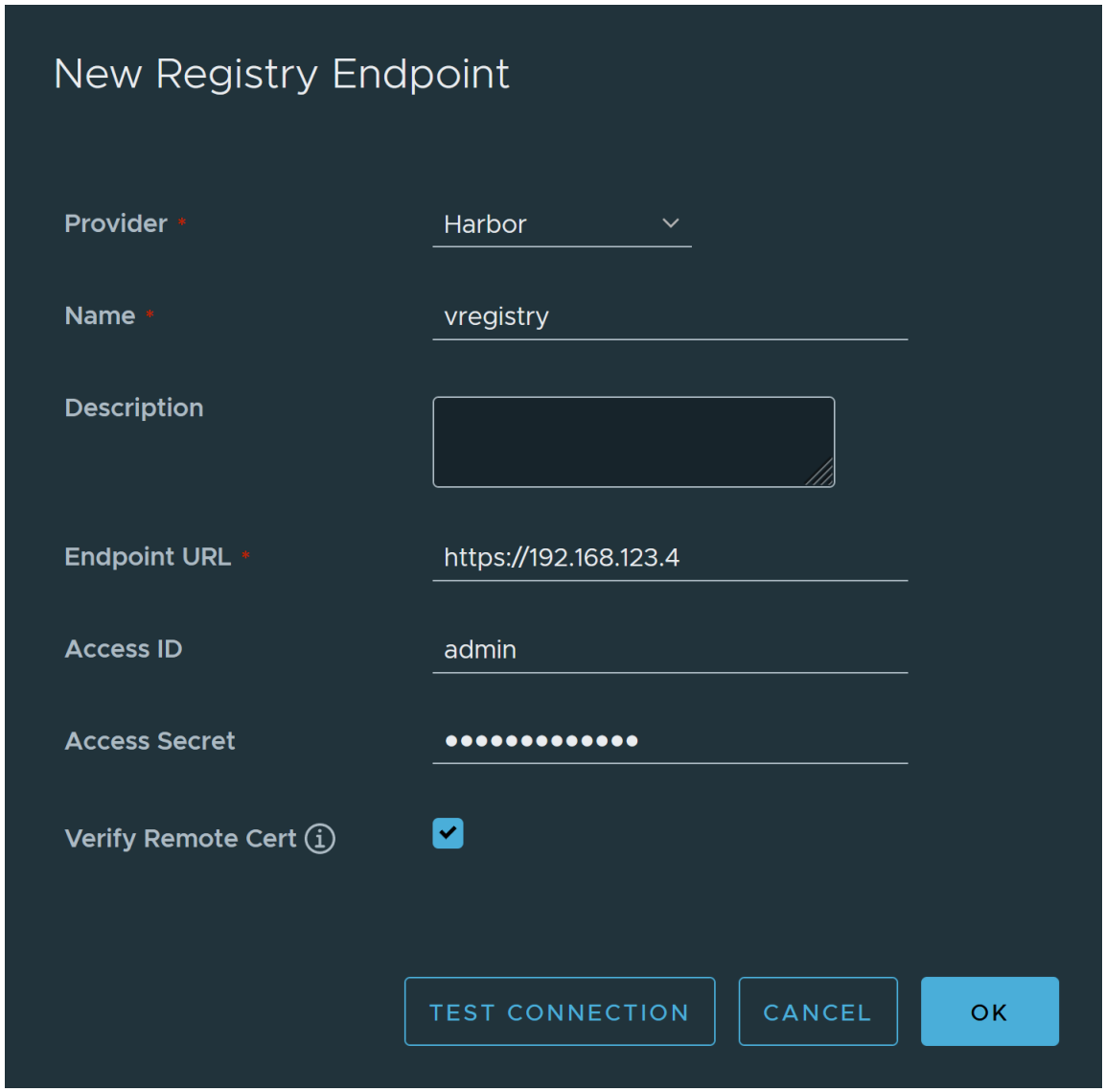
```
# echo 'WVdSdGFXND0=' | base64 -d | base64 -d
admin

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

4 내장된 레지스트리에 대한 복제 끝점 및 복제 규칙을 Harbor 감독자 서비스에 추가합니다.

a Harbor 감독자 서비스의 UI에 루트로 로그인합니다.

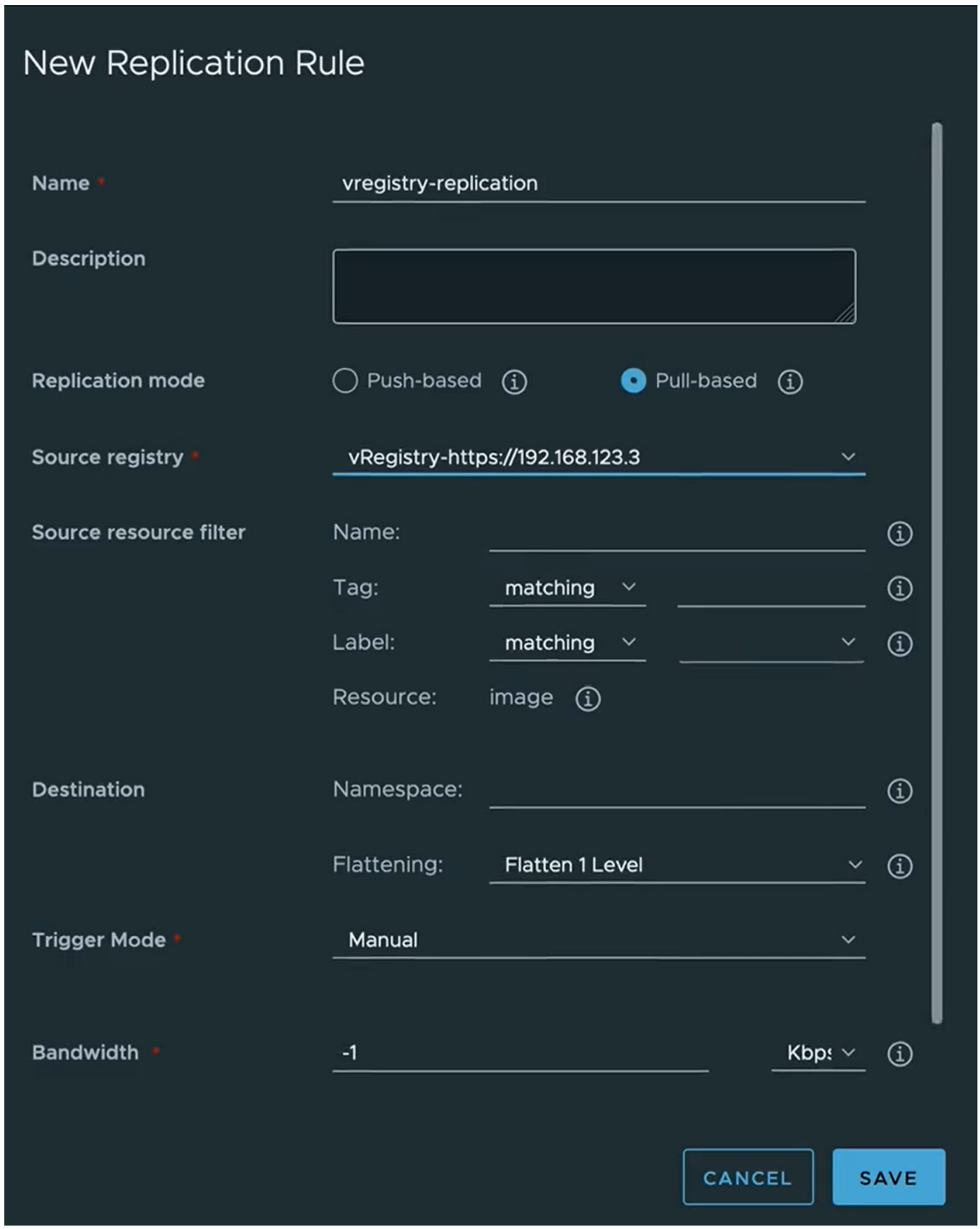
b **레지스트리**를 클릭하고 **새 끝점**을 클릭합니다.



c **복제** 탭을 선택하고 **새 복제 규칙**을 클릭합니다.

나머지 설정은 기본값으로 두고 다음 설정을 채웁니다.

- **이름** - 규칙의 이름을 제공합니다.
- **복제 모드** - **풀 기반**을 선택합니다.
- **소스 레지스트리** - 추가한 레지스트리 끝점을 선택합니다.



d **저장**을 클릭합니다.

5 새로 생성된 복제 규칙을 선택하고 **복제**를 클릭합니다.

결과

내장된 레지스트리의 컨텐츠가 Harbor 레지스트리에 복제됩니다.

프록시에서 이미지를 끌어와 에어갭 환경에 감독자 서비스 배포

10

인트라넷 내에서 개인 컨테이너 레지스트리를 활용하기 위해 에어갭 환경에 감독자 서비스를 배포할 수 있습니다.

감독자 서비스는 Kubernetes YAML 매니페스트 및 컨테이너 이미지의 컬렉션으로 저장되며 여러 네임스페이스에 리소스를 배포할 수 있는 Kubernetes Operator입니다. 이를 위해서는 감독자에 공통적인 레지스트리 세부 정보 집합이 필요합니다. 개인 이미지에서 감독자 서비스 컨테이너 이미지를 배포하려면 개인 레지스트리가 자체 서명된 CA(인증 기관)를 사용하거나 이미지 끌어오기에 대한 인증이 필요한 경우 감독자에 대한 신뢰 및 인증을 구성해야 합니다. 개인 레지스트리가 공용 CA(인증 기관)에서 서명한 TLS 인증서를 사용하는 경우 인증 기관 구성이 필요하지 않습니다.

에어갭 환경에서 프록시 또는 개인 레지스트리를 통해 끌어온 모든 감독자 서비스를 배포할 수 있습니다. 전체 감독자 서비스 집합은 https://vsphere-tmm.github.io/Supervisor-Services/ 항목을 참조하십시오.

다음 단계를 수행합니다.

1 감독자 서비스를 개인 컨테이너 레지스트리로 재배치합니다.

2 개인 컨테이너 이미지 레지스트리에 호스팅되는 감독자 서비스를 설치하고 사용합니다.

다음으로 아래 항목을 읽으십시오.

- 개인 레지스트리로 감독자 서비스 재배치
- 감독자 서비스 설치 및 사용

개인 레지스트리로 감독자 서비스 재배치

개인 컨테이너 레지스트리로 감독자 서비스를 재배치합니다.

사전 요구 사항

개인 컨테이너 이미지 레지스트리가 있는지 확인합니다.

절차

1 Carvel `imgpkg` 유틸리티를 설치합니다.

a `imgpkg` 설치

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

b 설치를 확인합니다.

```
imgpkg version
```

Carvel `imgpkg` 유틸리티에 대한 자세한 내용은 https://carvel.dev/imgpkg/docs/v0.42.x/install/에서 참조하십시오.

2 서비스에 대한 YAML 매니페스트를 가져옵니다.

`imgpkg` 번들을 찾습니다.

Contour 예제는 다음과 같습니다.

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

3 해당 `imgpkg` 번들의 tar를 다운로드합니다.

```
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
```

중요 `push` 및 `pull` 명령은 참조되는 모든 이미지를 끌어오지 않으므로 `copy` 명령을 사용하여 이미지를 재배치해야 합니다.

4 `imgpkg` 번들을 개인 컨테이너 이미지 레지스트리에 업로드합니다.

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

참고 `imgpkg`는 시스템의 신뢰 설정 및 Docker의 인증 구성을 준수합니다. 레지스트리에 인증이 필요한 경우 먼저 Docker CLI 명령 `docker login ${registry_url}`로 로그인합니다.

5 `imgpkg` 번들에 대한 새 URL로 감독자 서비스 YAML을 업데이트합니다.

예:

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

감독자 서비스 설치 및 사용

감독자 서비스는 개인 컨테이너 이미지 레지스트리로 재배치한 후 설치하고 사용할 수 있습니다.

감독자 서비스를 사용하려면 먼저 개인 레지스트리를 추가한 다음 감독자 서비스를 등록하고 설치합니다.

사전 요구 사항

서비스를 추가하는 vCenter Server 시스템에 대한 **감독자 서비스 관리** 권한이 있는지 확인합니다.

절차

1 개인 레지스트리를 추가합니다.

a vSphere Client 홈 메뉴에서 **워크로드 관리**를 선택합니다.

b **감독자** 탭을 클릭하고 목록에서 감독자를 선택합니다.

c **구성** 탭을 클릭하고 **컨테이너 레지스트리**를 클릭한 다음 **추가**를 클릭합니다.

d 개인 레지스트리의 이름을 입력하고 필요한 경우 CA, 사용자 이름 및 암호를 입력합니다.

2 vCenter Server를 감독자 서비스에 추가합니다.

a vSphere Client 홈 메뉴에서 **워크로드 관리**를 선택합니다.

b **서비스**를 선택합니다.

c 맨 위에 있는 드롭다운 메뉴에서 vCenter Server 시스템을 선택합니다.

d **새 서비스 추가** 카드에서 서비스 YAML 파일을 끌어서 놓습니다.

3 감독자 서비스 설치

a vSphere Client 홈 메뉴에서 **워크로드 관리**를 선택합니다.

b **서비스**를 선택합니다.

c 설치하려는 감독자 서비스의 카드에서 **작업 > 감독자에 설치**를 선택합니다.

d 서비스를 설치할 감독자를 선택합니다.

e 서비스에 필요한 경우 **YAML 서비스 구성** 필드에 구성 속성을 입력합니다.

감독자 서비스가 `SupervisorServiceDefinition` 형식이고 레지스트리에 인증이 필요한 경우 `registryName`, `registryUsername` 및 `registryPasswd` 속성을 입력합니다.