vSphere with Tanzu 구성 및 관리

업데이트 3
VMware vSphere 7.0
vCenter Server 7.0
VMware ESXi 7.0

vmware®

다음 VMware 웹 사이트에서 최신 기술 문서를 확인할 수 있습니다.

https://docs.vmware.com/kr/

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

VMware 코리아
서울시 강남구
영동대로 517
아셈타워 13층
(우) 06164
전화:+82 2 3016 6500
팩스: +82 2 3016 6501
www.vmware.com/kr

목차

vSphere with Tanzu 구성 및 관리

"vSphere with Tanzu 구성 및 관리" 는 vSphere Client를 사용하여 vSphere with Tanzu을 구성하고 관리하는 방법에 대한 정보를 제공합니다. 또한 kubectl을 사용하여 vSphere with Tanzu에서 실행되는 네임스페이스에 연결하고 지정된 네임스페이스에서 Kubernetes 워크로드를 실행하는 방법에 대해서도 설명합니다.

"vSphere with Tanzu 구성 및 관리" 는 플랫폼 아키텍처에 대한 개요는 물론 vSphere with Tanzu의 특정 요구 사항을 충족하는 스토리지, 계산 및 네트워킹 설정에 대한 고려 사항 및 모범 사례를 제공합니다. 기존 vSphere 클러스터에서 vSphere with Tanzu을 사용하도록 설정하고, 네임스페이스를 생성 및 관리하고, VMware Tanzu™ Kubernetes Grid™ 서비스를 사용하여 생성된 Tanzu Kubernetes 클러스터를 모니터링하기 위한 지침을 제공합니다.

이 정보는 kubectl을 통해 vSphere with Tanzu Kubernetes 제어부와 세션을 설정하고, 샘플 애플리케이션을 실행하고 VMware Tanzu™ Kubernetes Grid™ 서비스를 사용하여 Tanzu Kubernetes 클러스터를 생성하는 방법에 대한 지침도 제공합니다.

VMware는 포용성을 중요하게 생각합니다. 고객, 파트너 및 내부 커뮤니티 안에서 이러한 원칙을 강화하기 위해 포용성 있는 언어를 사용하여 컨텐츠를 만듭니다.

대상 사용자

"vSphere with Tanzu 구성 및 관리" 는 vSphere에서 vSphere with Tanzu을 사용하도록 설정하고, 네임스페이스를 구성하여 DevOps 팀에 제공하는 것은 물론 vSphere에서 Kubernetes 워크로드를 관리 및 모니터링하려는 vSphere 관리자를 대상으로 합니다. vSphere with Tanzu을 사용하려는 vSphere 관리자는 컨테이너 및 Kubernetes에 대한 기본적인 지식이 있어야 합니다.

또한, 이 정보는 vSphere with Tanzu 제어부와 세션을 설정하고, Kubernetes 워크로드를 실행하고, VMware Tanzu™ Kubernetes Grid™ 서비스를 사용하여 Kubernetes 클러스터를 배포하려는 DevOps 엔지니어를 대상으로 합니다. 또한 플랫폼에 애플리케이션을 배포하는 개발자는 지침을 위해 예제를 참조할 수 있습니다.

업데이트된 정보

1

"vSphere with Tanzu 구성 및 관리" 는 필요에 따라 새로운 정보와 수정 사항으로 정기적으로 업데이트됩니다.

이 표는 "vSphere with Tanzu 구성 및 관리" 의 업데이트 기록을 제공합니다.

개정	설명
2022년 12월 8일	네트워크 보안 정책에 대한 정보가 추가되었습니다. NSX 감독자 클러스터 네임스페이스에 보안 정책 추가의 내용을 참조하십시오.
2022년 10월 17일	맞춤법 오류가 수정되었습니다.
2022년 10월 12일	■ TKG 1.6 패키지 설치를 위한 링크가 추가되었습니다. Tanzu Kubernetes 클러스터에 TKG 패키지 배포의 내용을 참조하십시오. ■ TKGS v1alpha2 API 규격이 nodeDrainTimeout에 대한 올바른 데이터 유형(문자열)으로 업데이트되었습니다. Tanzu Kubernetes 클러스터 프로비저닝을 위한 TKGS v1alpha2 API의 내용을 참조하십시오. ■ 방명록 애플리케이션 소스 YAML 파일이 업데이트되었습니다. 방명록 예제 YAML 파일의 내용을 참조하십시오.
2022년 9월 15일	사소한 버그가 수정되었습니다.
2022년 7월 29일	vSphere with Tanzu 보안의 비밀 암호화에 대한 설명을 명확히 했습니다.
2022년 7월 07일	vCenter Server와 NSX-T 간의 호환성을 확인하기 위한 VMware 상호 운용성 매트릭스에 대한 링크가 추가되었습니다. vSphere with Tanzu 업데이트 정보 항목을 참조하십시오.
2022년 6월 28일	최신 절차에 대한 링크와 함께 vSphere with Tanzu 종료 및 시작 항목이 업데이트되었습니다. vSphere with Tanzu 워크로드 도메인 종료 및 시작의 내용을 참조하십시오.
2022년 6월 24일	vSphere 네임스페이스 생성 절차가 업데이트되었습니다. vSphere 네임스페이스 생성 및 구성의 내용을 참조하십시오.
2022년 6월 03일	■ 외부 DNS 확장 설명서가 업데이트되었습니다. 외부 DNS 서비스 검색을 위한 TKG 확장 배포 및 관리의 내용을 참조하십시오. ■ 오타가 수정되었습니다.
2022년 5월 24일	vSphere with Tanzu에서 ReadWriteMany 영구 볼륨 생성 항목이 Tanzu Kubernetes 클러스터가 있는 ReadWriteMany에 대한 지원 설명으로 업데이트되었습니다.
2022년 5월 20일	redis-leader 배포를 위해 redis:v6.0.5를 지정하여 방명록 예제 YAML이 업데이트되었습니다. 방명록 예제 YAML 파일의 내용을 참조하십시오.

개정	설명
2022년 5월 13일	■ vSphere용 Velero 플러그인 설치에 대한 설명서가 업데이트되었습니다. 감독자 클러스터에 vSphere용 Velero 플러그인 설치 및 구성의 내용을 참조하십시오. ■ 사소한 오타가 수정되었습니다.
2022년 5월 06일	■ TKGS 클러스터 작업자 노드 볼륨을 확장/축소하면 기존 볼륨 데이터가 삭제된다는 참고 사항이 추가되었습니다. 노드 볼륨 확장/축소의 내용을 참조하십시오. ■ SVC-TMC 통합 항목이 업데이트되었습니다. 감독자 클러스터의 Tanzu Kubernetes Grid 서비스를 Tanzu Mission Control과 통합의 내용을 참조하십시오. ■ TKGS 클러스터 프로비저닝 워크플로가 업데이트되었습니다. TKGS v1alpha2 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하는 워크플로의 내용을 참조하십시오. ■ TKGS 네트워킹 개념이 업데이트되었습니다. Tanzu Kubernetes Grid 서비스 클러스터 네트워킹 의 내용을 참조하십시오. ■ 오타가 수정되고 부분적으로 편집되었습니다.
2022년 4월 21일	부분적 개정.
2022년 4월 18일	■ 리소스 경합 및 잠재적 클러스터 다운타임을 방지하기 위해 운영 클러스터에 대해 보장됨 VM 클래스를 사용하도록 권장하는 문구가 강화되었습니다. Tanzu Kubernetes 클러스터에 대한 가상 시스템 클래스 및 vSphere with Tanzu의 네임스페이스에서 사용 가능한 VM 리소스 보기의 내용을 참조하십시오. ■ Tanzu Kubernetes 클러스터가 사용 시도 VM 클래스로 프로비저닝되는 vSphere 네임스페이스에 대한 제한 설정에 대한 참고 사항이 추가되었습니다. vSphere 네임스페이스의 Kubernetes 개체에 대한 제한 사항 구성의 내용을 참조하십시오.
2022년 4월 15일	■ TKGS 프로비저닝 설명서에 작업자 노드/노드 풀이 0개로 프로비저닝된 클러스터에는 로드 밸런서 서비스가 할당되지 않는다는 참고 사항이 추가되었습니다. ■ 로컬 컨텐츠 라이브러리 설명서가 업데이트되었습니다. Tanzu Kubernetes 릴리스용 로컬 컨텐츠 라이브러리 생성, 보안 및 동기화의 내용을 참조하십시오. ■ Sysadmin 로그인 인증 키에 대한 자세한 내용이 추가되었습니다. 컨트롤러 배포의 내용을 참조하십시오. ■ SAN(주체 대체 이름)에 대한 자세한 내용이 추가되었습니다. 컨트롤러에 인증서 할당의 내용을 참조하십시오. ■ vsan-direct 및 vsan-sna 스토리지 클래스는 감독자 클러스터의 애플리케이션에서만 사용할 수 있고 Tanzu Kubernetes 클러스터 내부에서는 사용할 수 없다는 점을 명확히 설명했습니다. vSphere with Tanzu에서 상태 저장 서비스 사용. ■ 감독자 클러스터 제어부 로그의 원격 스트리밍을 구성하는 방법에 대한 정보가 추가되었습니다. 감독자 클러스터 제어부의 로그를 원격 rsyslog로 스트리밍의 내용을 참조하십시오.
2022년 3월 28일	vSphere with Tanzu 환경의 HTTP 프록시 구성에 대한 정보가 추가되었습니다. vSphere with Tanzu에서 HTTP 프록시 설정 구성의 내용을 참조하십시오.
2022년 3월 18일	사소한 오타가 수정되었습니다.
2022년 3월 04일	로컬 컨텐츠 라이브러리에 대한 문제 해결 항목이 추가되었습니다. 로컬 컨텐츠 라이브러리 오류 문제 해결의 내용을 참조하십시오.
2022년 2월 28일	■ TKG 1.5 패키지를 설치하기 위한 링크가 추가되었습니다. Tanzu Kubernetes 클러스터에 TKG 패키지 배포의 내용을 참조하십시오. ■ 다이어그램에서 제어부 VM 이름이 수정되었습니다. NSX-T Data Center를 사용하는 감독자 클러스터용 토폴로지의 내용을 참조하십시오. ■ 사소한 오타가 수정되었습니다.

개정	설명
2022년 2월 18일	■ DHCP가 사용되도록 설정되지 않은 경우 구성 옵션에 대한 단계로 컨트롤러 구성 항목이 업데이트되었습니다. ■ 미리 생성된 유효한 인증서를 업로드하는 단계로 컨트롤러에 인증서 할당 항목이 업데이트되었습니다. ■ IPv6이 지원되지 않음을 나타내도록 시스템 요구 사항이 업데이트되었습니다. 장 4 vSphere with Tanzu에 대한 네트워킹의 내용을 참조하십시오. ■ 클러스터 규격에서 `pods.cidrBlocks` 설정을 사용자 지정하는 경우 TKGS 클러스터에 대한 네트워킹 요구 사항이 업데이트되었습니다. TKGS v1alpha2 API 사용에 대한 요구 사항 및 Tanzu Kubernetes 클러스터 프로비저닝을 위한 TKGS v1alpha2 API의 내용을 참조하십시오. ■ TKGS 클러스터에 대한 예제 네트워킹 명령 목록이 업데이트되었습니다. Tanzu Kubernetes 클러스터 네트워킹 명령 사용의 내용을 참조하십시오.
2022년 2월 11일	■ 사용 설정 후 다음 단계로 vSphere 네임스페이스를 생성하고 구성하는 절차에 대한 링크를 포함하도록 vSphere 네트워킹으로 워크로드 관리 사용 항목이 업데이트되었습니다. ■ 다음 단계로 감독자 클러스터에 로그인하기 위한 링크를 포함하도록 vSphere 네임스페이스 생성 및 구성 항목이 업데이트되었습니다.
2022년 2월 8일	■ HA Proxy 로드 밸런서를 설치하기 위한 시스템 요구 사항이 워크로드 네트워크가 관리 네트워크와 다른 서브넷에 있어야 한다는 참고 사항으로 업데이트되었습니다. vSphere 네트워킹 및 HAProxy 로드 밸런서를 사용하여 vSphere with Tanzu를 설정하기 위한 시스템 요구 사항의 내용을 참조하십시오.
2022년 2월 4일	■ 네트워킹에 대한 오래된 지침을 제거하기 위해 vSphere with Tanzu에서 ReadWriteMany 영구 볼륨 생성 항목이 업데이트되었습니다. ■ 추가 컨텍스트 및 예제로 TKGS v1alpha2 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하는 워크플로 항목이 업데이트되었습니다. ■ 오타를 수정하기 위해 감독자 클러스터 네트워킹 항목이 업데이트되었습니다. ■ v1alpah2 TKGS API를 준수하도록 클러스터 규격을 업데이트하는 데 `kubectl patch` 메서드를 사용하지 말라는 주의로 패치 방법을 사용하여 Tanzu Kubernetes 클러스터 업데이트 항목이 업데이트되었습니다. ■ 이 작업을 수행하려면 `kubectl edit` 메서드를 사용해야 한다는 명시적 언급으로 클러스터 규격이 TKGS v1alpha2 API로 변환된 후 Tanzu Kubernetes 릴리스 업데이트 항목이 업데이트되었습니다.
2022년 1월 28일	■ 사용자 탐색을 더 잘 돕기 위한 스크린샷으로 Kubernetes CLI 다운로드 항목이 업데이트되었습니다. vSphere에 대한 Kubernetes CLI 도구 다운로드 및 설치의 내용을 참조하십시오. ■ 클러스터 노드 볼륨 추가 또는 변경에 대한 정보로 Tanzu Kubernetes 클러스터 확장/축소 항목이 업데이트되었습니다. TKGS v1alpha2 API를 사용하여 Tanzu Kubernetes 클러스터 확장/축소의 내용을 참조하십시오. ■ 다운로드할 확장 매니페스트를 보려면 1.3.1 버전을 선택해야 한다고 알리기 위해 TKG 확장 1.3.1 다운로드 항목이 업데이트되었습니다. TKG 확장 v1.3.1 번들 다운로드의 내용을 참조하십시오. ■ vSphere with Tanzu용 NSX Advanced Load Balancer 및 vSphere 네트워킹을 구성하는 경우 vSphere 포드 CIDR 범위에 대한 요구 사항이 업데이트되었습니다. vSphere 네트워킹 및 NSX Advanced Load Balancer를 사용하여 vSphere with Tanzu를 설정하기 위한 시스템 요구 사항의 내용을 참조하십시오.
2021년 12월 17일	Kapp 컨트롤러 구성을 편집하여 프록시 서버를 추가하는 방법을 설명하기 위해 TKG 1.3.1 확장 사전 요구 사항 항목이 업데이트되었습니다. TKG 확장 사전 요구 사항 설치의 내용을 참조하십시오.

개정	설명
2021년 12월 10일	■ Tanzu Kubernetes Grid 서비스 TKGS v1alpha2 API를 사용하여 Tanzu Kubernetes 클러스터 프로비저닝를 지원하도록 Tanzu Kubernetes 클러스터 업데이트 항목을 새로 고쳤습니다. 클러스터 규격이 TKGS v1alpha2 API로 변환된 후 Tanzu Kubernetes 릴리스 업데이트 항목도 참조하십시오. ■ 고객 피드백을 해결하기 위해 TKG 확장에 대한 설명서가 업데이트되었습니다. Tanzu Kubernetes 클러스터에 TKG 패키지 배포의 내용을 참조하십시오. ■ TKGS 프록시 구성 스크린샷 및 요구 사항이 업데이트되었습니다. TKGS v1alpha2 API에 대한 구성 매개 변수의 내용을 참조하십시오.
2021년 11월 24일	vSphere with Tanzu에 대한 vSphere 네트워킹 및 NSX Advanced Load Balancer 구성에 대한 설명서가 업데이트되었습니다. vSphere with Tanzu에 대한 vSphere 네트워킹 및 NSX Advanced Load Balancer 구성 항목을 참조하십시오.
2021년 11월 5일	Tanzu Kubernetes Grid 서비스에서 프로비저닝된 Tanzu Kubernetes 클러스터에 TKG 1.4 패키지를 설치하기 위한 링크가 추가되었습니다. Tanzu Kubernetes 클러스터에 TKG 패키지 배포의 내용을 참조하십시오.
2021년 10월 29일	■ TKGS 클러스터에 vGPU 워크로드를 배포하기 위한 설명서가 업데이트되었습니다. Tanzu Kubernetes 클러스터에 AI/ML 워크로드 배포의 내용을 참조하십시오. ■ vSphere용 Velero 플러그인 설치 설명서가 업데이트되었습니다. 감독자 클러스터에 vSphere용 Velero 플러그인 설치 및 구성의 내용을 참조하십시오. ■ RBAC 예제가 업데이트되었습니다. 포드 보안 정책에 대한 역할 바인딩 예의 내용을 참조하십시오. ■ 감독자 클러스터 네트워킹이 업데이트되었습니다. 감독자 클러스터 네트워킹의 내용을 참조하십시오. ■ 네트워킹 및 워크로드 관리 사용 사전 요구 사항 항목에 감독자 클러스터에서 DRS를 사용하지 않도록 설정해서는 안 되며 DRS를 사용하지 않도록 설정하면 클러스터 중단이 발생한다는 주의가 추가되었습니다.
2021년 10월 21일	■ vGPU 지원 TKGS 클러스터에 AI/ML 워크로드를 배포하기 위한 설명서가 추가되었습니다. Tanzu Kubernetes 클러스터에 AI/ML 워크로드 배포의 내용을 참조하십시오. ■ 패스스루 모드의 PCI 디바이스 지원에 대한 정보로 vSphere with Tanzu의 VM 클래스에 PCI 디바이스 추가 항목이 업데이트되었습니다. ■ Tanzu Kubernetes 릴리스 목록을 전용 릴리스 정보로 이동했습니다. Tanzu Kubernetes 릴리스와 관련된 모든 정보는 이 릴리스 정보를 참조하십시오. ■ 오타 및 부분적인 설명서 버그가 수정되었습니다.
2021년 10월 8일	■ Tanzu Kubernetes 릴리스 버전이 업데이트되었습니다. 업데이트를 위한 Tanzu Kubernetes 클러스터 호환성 확인 및 TKGS v1alpha2 API를 사용하여 Tanzu Kubernetes 클러스터 프로비저닝의 내용을 참조하십시오. ■ 에어갭 환경에 vSphere용 Velero 플러그인를 설치하는 절차가 추가되었습니다. 자세한 내용은 감독자 클러스터에 vSphere용 Velero 플러그인 설치 및 구성의 내용을 참조하십시오. ■ 감독자 클러스터 백업 및 복원에 대한 정보가 업데이트되었습니다. 자세한 내용은 vSphere with Tanzu 백업 및 복원에 대한 고려 사항의 내용을 참조하십시오. ■ 오타가 수정되었습니다.
2021년 10월 05일	최초 릴리스

vSphere with Tanzu 개념

2

vSphere with Tanzu를 사용하면 vSphere 클러스터를 플랫폼으로 전환하여 전용 리소스 풀에서 Kubernetes 워크로드를 실행할 수 있습니다. vSphere 클러스터에서 vSphere with Tanzu가 사용되도록 설정되면 하이퍼바이저 계층에 Kubernetes 제어부가 생성됩니다. 그러면 vSphere 포드를 배포하여 Kubernetes 컨테이너를 실행하거나 VMware Tanzu™ Kubernetes Grid™ 서비스를 통해 업스트림 Kubernetes 클러스터를 생성하고 이러한 클러스터 내에서 애플리케이션을 실행할 수 있습니다.

본 장은 다음 항목을 포함합니다.

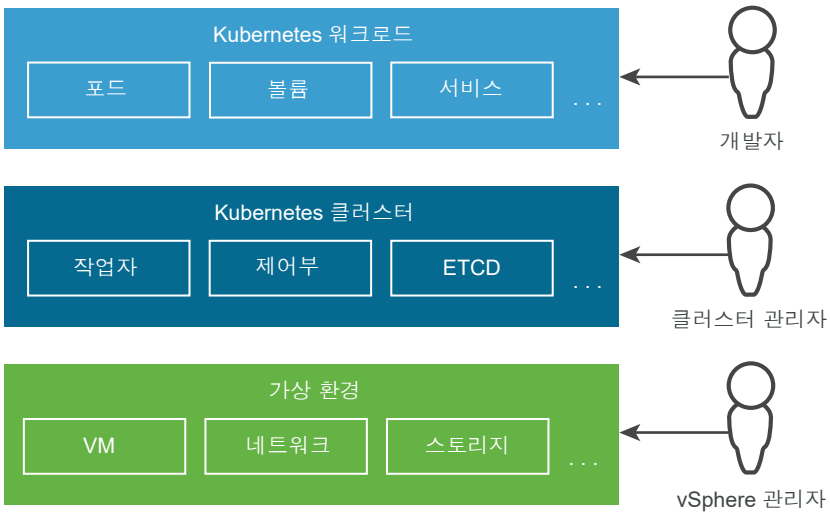
vSphere with Tanzu이란?

vSphere with Tanzu을 사용하여 vSphere를 하이퍼바이저 계층에서 Kubernetes 워크로드를 기본적으로 실행하는 플랫폼으로 변환할 수 있습니다. vSphere 클러스터에서 사용하도록 설정되면 vSphere with Tanzu은 ESXi 호스트에서 직접 Kubernetes 워크로드를 실행하고 전용 리소스 풀 내에 업스트림 Kubernetes 클러스터를 생성하는 기능을 제공합니다.

현재 애플리케이션 스택의 과제

오늘날의 분산 시스템은 일반적으로 다수의 Kubernetes 포드 및 VM을 실행하는 여러 마이크로서비스로 구성됩니다. vSphere with Tanzu 기반이 아닌 일반적인 스택은 VM 내부에 배포되는 Kubernetes 인프라와 이러한 VM에서 각각 실행되는 Kubernetes 포드가 있는 기본 가상 환경으로 구성됩니다. 애플리케이션 개발자, Kubernetes 클러스터 관리자 및 vSphere 관리자라는 3가지 개별 역할이 스택의 각 부분을 운영합니다.

그림 2-1. 현재 애플리케이션 스택



서로 다른 역할은 서로의 환경을 보거나 제어할 수 없습니다.

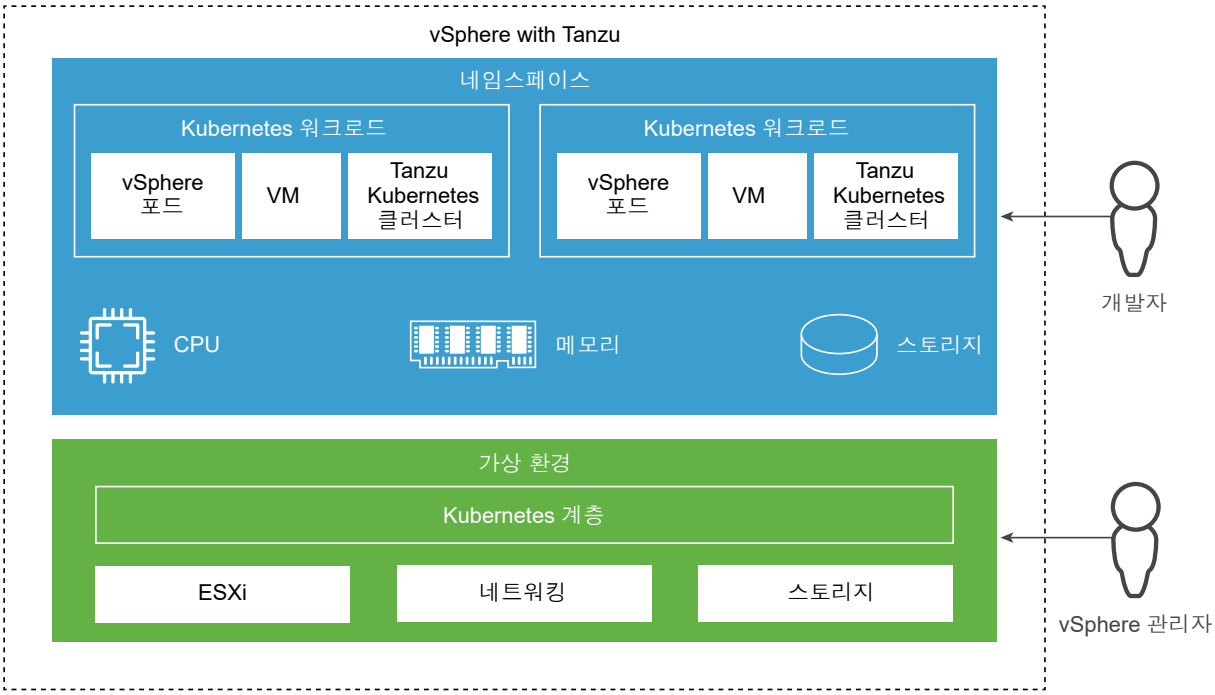
- 애플리케이션 개발자는 Kubernetes 포드를 실행하고 Kubernetes 기반 애플리케이션을 배포하고 관리할 수 있습니다. 수백 개의 애플리케이션을 실행하는 전체 스택에 대한 가시성은 없습니다.
- DevOps 엔지니어는 Kubernetes 인프라만 제어할 수 있으며, 가상 환경을 관리 또는 모니터링하고 리소스 관련 및 기타 문제를 해결하는 도구가 없습니다.
- vSphere 관리자는 기본 가상 환경을 완전히 제어할 수 있지만 Kubernetes 인프라, 가상 환경에 있는 다른 Kubernetes 개체의 배치 및 이러한 개체가 리소스를 소비하는 방식에 대한 가시성은 없습니다.

전체 스택에 대한 작업은 세 가지 역할 간에 통신이 필요하기 때문에 어려울 수 있습니다. 스택의 서로 다른 계층 간에 통합이 부족한 경우에도 문제가 발생할 수 있습니다. 예를 들어 Kubernetes 스케줄러는 vCenter Server 인벤토리에 대한 가시성을 가지고 있지 않으며 포드를 지능적으로 배치할 수 없습니다.

vSphere with Tanzu 사용의 이점

vSphere with Tanzu은 하이퍼바이저 계층에서 직접 Kubernetes 제어부를 생성합니다. vSphere 관리자가 **워크로드 관리**를 위해 기존 vSphere 클러스터를 사용하도록 설정하기 때문에 클러스터의 일부인 ESXi 호스트 내에 Kubernetes 계층이 생성됩니다. **워크로드 관리**와 함께 사용되도록 설정된 클러스터를 감독자 클러스터라고 합니다.

그림 2-2. vSphere with Tanzu



하이퍼바이저 계층에 Kubernetes 제어부가 있으면 vSphere에서 다음과 같은 기능을 사용할 수 있습니다.

- vSphere 관리자는 감독자 클러스터에서 vSphere 네임스페이스라는 네임스페이스를 생성하고 이를 지정된 양의 메모리, CPU 및 스토리지를 사용하여 구성할 수 있습니다. vSphere 네임스페이스를 DevOps 엔지니어에게 제공합니다.
- DevOps 엔지니어는 vSphere 네임스페이스 내에 공유 리소스 풀이 있는 동일한 플랫폼에서 Kubernetes 컨테이너로 구성된 워크로드를 실행할 수 있습니다. vSphere with Tanzu에서 컨테이너는 vSphere 포드이라는 특수한 유형의 VM 내에서 실행됩니다. 일반 VM을 배포할 수도 있습니다.
- DevOps 엔지니어는 네임스페이스 내에서 여러 Kubernetes 클러스터를 생성 및 관리하고 Tanzu Kubernetes Grid 서비스를 사용하여 해당 수명 주기를 관리할 수 있습니다. Tanzu Kubernetes Grid 서비스를 사용하여 생성된 Kubernetes 클러스터를 Tanzu Kubernetes 클러스터라고 합니다.
- vSphere 관리자는 vSphere Client를 사용하여 vSphere 포드, VM 및 Tanzu Kubernetes 클러스터를 관리하고 모니터링할 수 있습니다.
- vSphere 관리자는 서로 다른 네임스페이스 내에서 실행되는 vSphere 포드, VM 및 Tanzu Kubernetes 클러스터, 이러한 항목의 환경 내 배치 및 리소스 소비 방식에 대한 완전한 가시성을 갖습니다.

하이퍼바이저 계층에서 Kubernetes를 실행하면 vSphere 관리자와 DevOps 팀 간의 협업이 쉬워집니다. 두 역할이 모두 동일한 개체로 작동하기 때문입니다.

워크로드란?

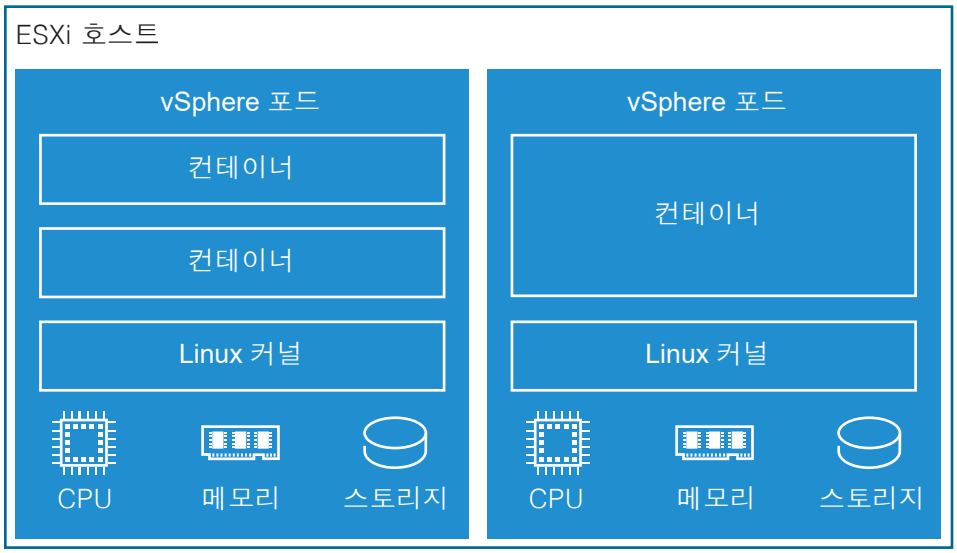
vSphere with Tanzu에서 워크로드는 다음 중 한 가지 방법으로 배포되는 애플리케이션입니다.

- vSphere 포드, 일반 VM 또는 둘 모두의 내부에서 실행되는 컨테이너로 구성된 애플리케이션.
- VMware TanzuTM Kubernetes GridTM 서비스를 사용하여 배포된 Tanzu Kubernetes 클러스터.
- VMware TanzuTM Kubernetes GridTM 서비스를 사용하여 배포된 Tanzu Kubernetes 클러스터 내에서 실행되는 애플리케이션.

vSphere 포드란?

vSphere with Tanzu에는 Kubernetes 포드와 동일한 vSphere 포드라는 새로운 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 NSX-T Data Center를 사용하여 네트워킹 스택으로 구성된 감독자 클러스터에서만 지원됩니다.

그림 2-3. vSphere 포드



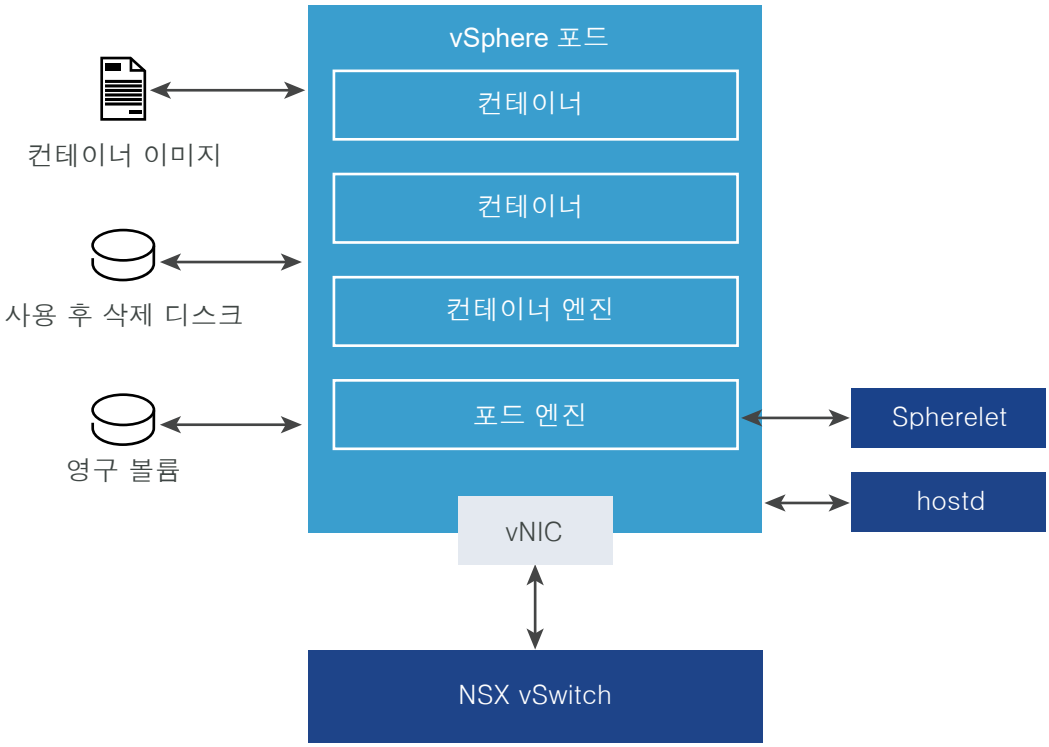
vSphere 포드는 vCenter Server의 개체이므로 워크로드에 대해 다음 기능을 사용할 수 있습니다.

- 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 감독자 클러스터에서 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.

그림 2-4. vSphere 포드 네트워킹 및 스토리지



vSphere 포드는 저장된 개체에 따라, 사용 후 삭제되는 VMDK, 영구 볼륨 VMDK 및 컨테이너 이미지 VMDK라는 세 가지 유형의 스토리지를 사용합니다. vSphere 관리자는 감독자 클러스터 수준에서 컨테이너 이미지 캐시, 사용 후 삭제되는 VMDK 및 제어부 VM 배치에 대한 스토리지 정책을 구성합니다. vSphere 네임스페이스 수준에서 영구 볼륨 배치 및 Tanzu Kubernetes 클러스터의 VM 배치를 위한 스토리지 정책을 구성합니다. 장 10 vSphere with Tanzu에서 영구 스토리지 사용에서 vSphere with Tanzu의 스토리지 요구 사항 및 개념에 대한 자세한 내용을 참조하십시오.

네트워킹을 위해 vSphere 포드 및 Tanzu Kubernetes Grid 서비스를 통해 생성된 Tanzu Kubernetes 클러스터의 VM은 NSX-T Data Center에서 제공하는 토폴로지를 사용합니다. 자세한 내용은 감독자 클러스터 네트워킹 항목을 참조하십시오.

vSphere 포드는 NSX-T Data Center를 네트워킹 스택으로 사용하는 감독자 클러스터에서만 지원됩니다. vSphere 네트워킹 스택으로 구성된 클러스터에서는 지원되지 않습니다.

Tanzu Kubernetes 클러스터란?

Tanzu Kubernetes 클러스터는 VMware에서 구축, 서명 및 지원하는 오픈 소스 Kubernetes 컨테이너 오케스트레이션 플랫폼의 전체 배포입니다. Tanzu Kubernetes Grid 서비스를 사용하여 감독자 클러스터에

서 Tanzu Kubernetes 클러스터를 프로비저닝하고 운영할 수 있습니다. 감독자 클러스터는 vSphere with Tanzu와 함께 사용되도록 설정된 vSphere 클러스터입니다.

Tanzu Kubernetes Grid 서비스에서 생성된 Tanzu Kubernetes 클러스터의 주요 특성

Tanzu Kubernetes Grid 서비스에서 프로비저닝된 Tanzu Kubernetes 클러스터에는 다음과 같은 특성이 있습니다.

- Kubernetes의 개인 맞춤화된 설치
- vSphere 인프라와 통합
- 즉시 사용 가능
- VMware에서 완전히 지원됨
- Kubernetes에서 관리됨

참고 VMware에서는 Kubernetes에 중점을 둔 제품군을 Tanzu 브랜드로 판매합니다. Tanzu Kubernetes Grid 서비스를 사용하여 생성한 Tanzu Kubernetes 클러스터는 Tanzu Edition 라이센스의 구성 요소입니다. VMware에서 Tanzu 브랜드로 판매하는 기타 Kubernetes 기반 제품에 대한 자세한 내용은 VMware Tanzu 설명서를 참조하십시오. vSphere with Tanzu의 라이센싱에 대한 자세한 내용은 vSphere with Tanzu에 대한 라이센싱 항목을 참조하십시오.

Kubernetes의 개인 맞춤화된 설치

Tanzu Kubernetes 클러스터는 Kubernetes의 개인 맞춤화된 설치입니다.

Tanzu Kubernetes Grid 서비스는 Tanzu Kubernetes 클러스터 프로비저닝을 위해 vSphere에 최적화된 기본값을 제공합니다. Tanzu Kubernetes Grid 서비스를 사용하면 일반적으로 엔터프라이즈급 Kubernetes 클러스터를 배포하고 실행하는 데 필요한 시간과 노력을 줄일 수 있습니다.

자세한 내용은 Tanzu Kubernetes Grid 서비스 아키텍처의 내용을 참조하십시오.

vSphere 인프라와 통합

Tanzu Kubernetes 클러스터는 Kubernetes 실행에 최적화된 기본 vSphere 인프라와 통합됩니다.

Tanzu Kubernetes 클러스터는 스토리지, 네트워킹 및 인증을 포함하여 vSphere SDDC 스택과 통합됩니다. 또한 Tanzu Kubernetes 클러스터는 vCenter Server 클러스터에 매핑되는 감독자 클러스터에 구축됩니다. 긴밀하게 통합되어 있기 때문에 Tanzu Kubernetes 클러스터 실행에는 일관적인 제품 경험이 가능합니다.

자세한 내용은 vSphere with Tanzu 아키텍처를 참조하십시오.

즉시 사용 가능

운영 워크로드를 실행하기 위해 Tanzu Kubernetes 클러스터가 조정되었습니다.

Tanzu Kubernetes Grid 서비스는 운영 준비가 된 Tanzu Kubernetes 클러스터를 프로비저닝합니다. 추가 구성을 수행하지 않아도 운영 워크로드를 실행할 수 있습니다. 또한 가용성을 보장하고 Kubernetes 소프트웨어 업그레이드 롤링을 허용하고 서로 다른 버전의 Kubernetes를 별도의 클러스터에 실행할 수 있습니다.

자세한 내용은 장 13 TKGS 클러스터 프로비저닝 및 운영의 내용을 참조하십시오.

VMware에서 완전히 지원됨

Tanzu Kubernetes 클러스터는 VMware에서 지원됩니다.

Tanzu Kubernetes 클러스터는 VMware의 오픈 소스 Linux 기반 Photon OS를 사용하며 vSphere 인프라에 배포되고 ESXi 호스트에서 실행됩니다. 하이퍼바이저에서 Kubernetes 클러스터에 이르기까지 스택의 계층에 문제가 발생하는 경우 VMware에 문의하면 그 해답을 찾을 수 있습니다.

자세한 내용은 VMware 지원에 문의하십시오.

Kubernetes에서 관리됨

Tanzu Kubernetes 클러스터는 Kubernetes에서 관리됩니다.

Tanzu Kubernetes 클러스터는 자체적으로 Kubernetes 클러스터인 감독자 클러스터 위에 구축됩니다. Tanzu Kubernetes 클러스터는 사용자 지정 리소스를 사용하여 vSphere 네임스페이스에 정의됩니다. 익숙한 kubectl 명령을 사용하여 셀프 서비스 방식으로 Tanzu Kubernetes 클러스터를 프로비저닝합니다. 도구체인 간에는 일관성이 있습니다. 클러스터를 프로비저닝하든 워크로드를 배포하든 동일한 명령, 익숙한 YAML 및 공통 워크플로를 사용합니다.

자세한 내용은 Tanzu Kubernetes 클러스터 테넌시 모델의 내용을 참조하십시오.

vSphere 포드 및 Tanzu Kubernetes 클러스터를 사용하는 경우

Tanzu Kubernetes Grid 서비스에서 프로비저닝된 vSphere 포드 또는 Tanzu Kubernetes 클러스터 사용은 감독자 클러스터에서 Kubernetes 워크로드 배포 및 관리와 관련된 목표에 따라 다릅니다.

다음을 수행하려는 vSphere 관리자 또는 DevOps 엔지니어의 경우 vSphere 포드를 사용합니다.

- Kubernetes 클러스터를 사용자 지정할 필요 없이 컨테이너를 실행합니다.
- 강력한 리소스 및 보안 격리를 사용하여 컨테이너화된 애플리케이션을 생성합니다.

- vSphere 포드를 ESXi 호스트에 직접 배포합니다.

다음을 수행하려는 DevOps 엔지니어 또는 개발자인 경우 Tanzu Kubernetes Grid 서비스에서 프로비저닝된 Tanzu Kubernetes 클러스터를 사용합니다.

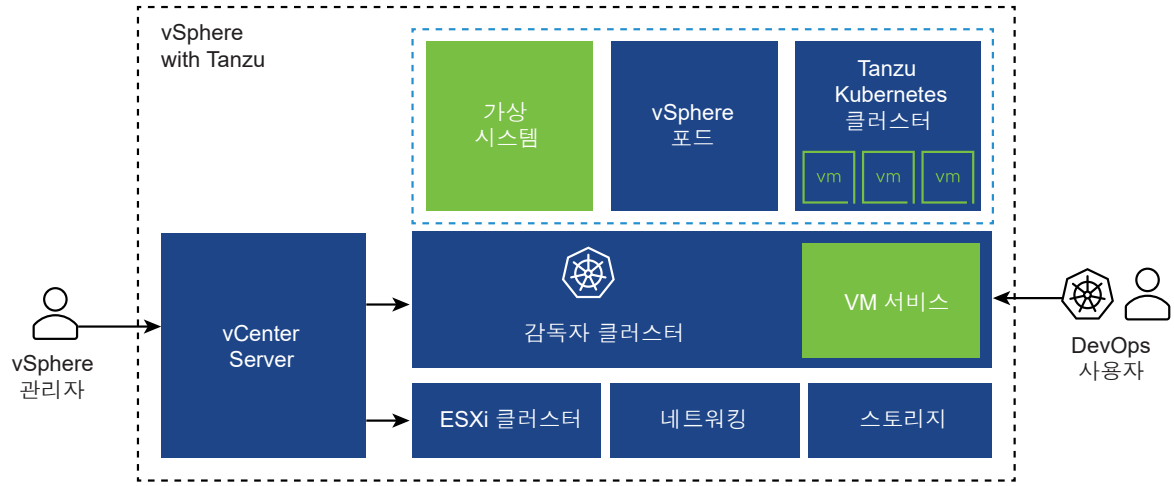
- 오픈 소스, 커뮤니티에 맞춘 Kubernetes 소프트웨어에서 컨테이너화된 애플리케이션을 실행합니다.
- 제어부 및 작업자 노드에 대한 루트 수준 액세스를 포함하여 Kubernetes 클러스터를 제어합니다.
- 인프라 업그레이드 없이도 Kubernetes 버전을 최신 상태로 유지합니다.
- CI/CD 파이프라인을 사용하여 단기간의 Kubernetes 클러스터를 프로비저닝합니다.
- Kubernetes 클러스터를 사용자 지정합니다(예: 사용자 지정 리소스 정의, 운영자 및 Helm 차트 설치).
- kubectl CLI를 사용하여 Kubernetes 네임스페이스를 생성합니다.
- 클러스터 수준 액세스 제어 관리 및 `PodSecurityPolicies`. 구성
- `NodePort`. 유형의 서비스를 생성합니다.
- `HostPath` 볼륨을 사용합니다.
- 권한 있는 포드를 실행합니다.

vSphere with Tanzu에서 가상 시스템 사용

vSphere with Tanzu는 DevOps 엔지니어가 공통의 공유 Kubernetes 환경에서 컨테이너뿐 아니라 VM을 배포하고 실행할 수 있는 VM 서비스 기능을 제공합니다. 컨테이너와 VM 둘 다 동일한 vSphere 네임스페이스 리소스를 공유하며 단일 vSphere with Tanzu 인터페이스를 통해 관리할 수 있습니다.

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 클러스터를 구성하는 VM을 관리합니다. 클러스터에 대한 자세한 내용은 Tanzu Kubernetes Grid 서비스 아키텍처 및 장 13 TKGS 클러스터 프로비저닝 및 운영 항목을 참조하십시오.



VM 서비스를 통해 배포된 각 VM은 vSphere with Tanzu 인프라를 기반으로 자체 운영 체제를 포함한 모든 구성 요소를 실행하는 완전한 시스템으로 작동합니다. VM은 감독자 클러스터가 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.

가상 시스템 배포 및 관리에 대한 자세한 내용은 장 12 vSphere with Tanzu에서 가상 시스템 배포 및 관리 항목을 참조하십시오.

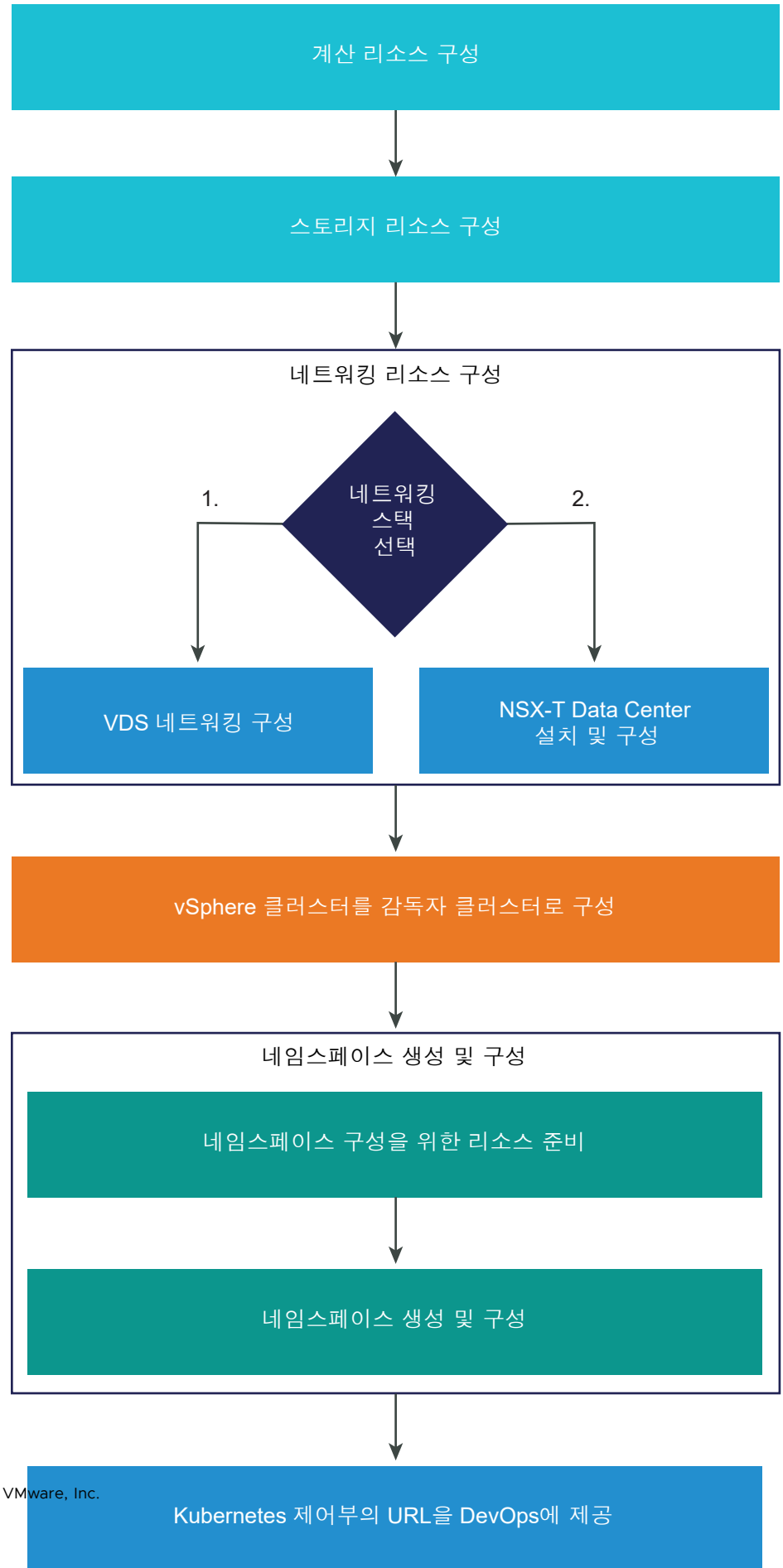
vSphere with Tanzu 사용자 역할 및 워크플로

vSphere with Tanzu 플랫폼에는 vSphere 관리자와 DevOps 엔지니어라는 두 가지 역할이 포함됩니다. 두 역할은 서로 다른 인터페이스를 통해 플랫폼과 상호 작용하며 vCenter Single Sign-On에서 각자에 대해 정의된 사용자 또는 사용자 그룹 및 연결된 사용 권한이 있을 수 있습니다. vSphere 관리자 및 DevOps 엔지니어 역할에 대한 워크플로는 고유하며 이러한 역할에 필요한 특정 전문 기술 영역에 따라 결정됩니다.

사용자 역할 및 워크플로

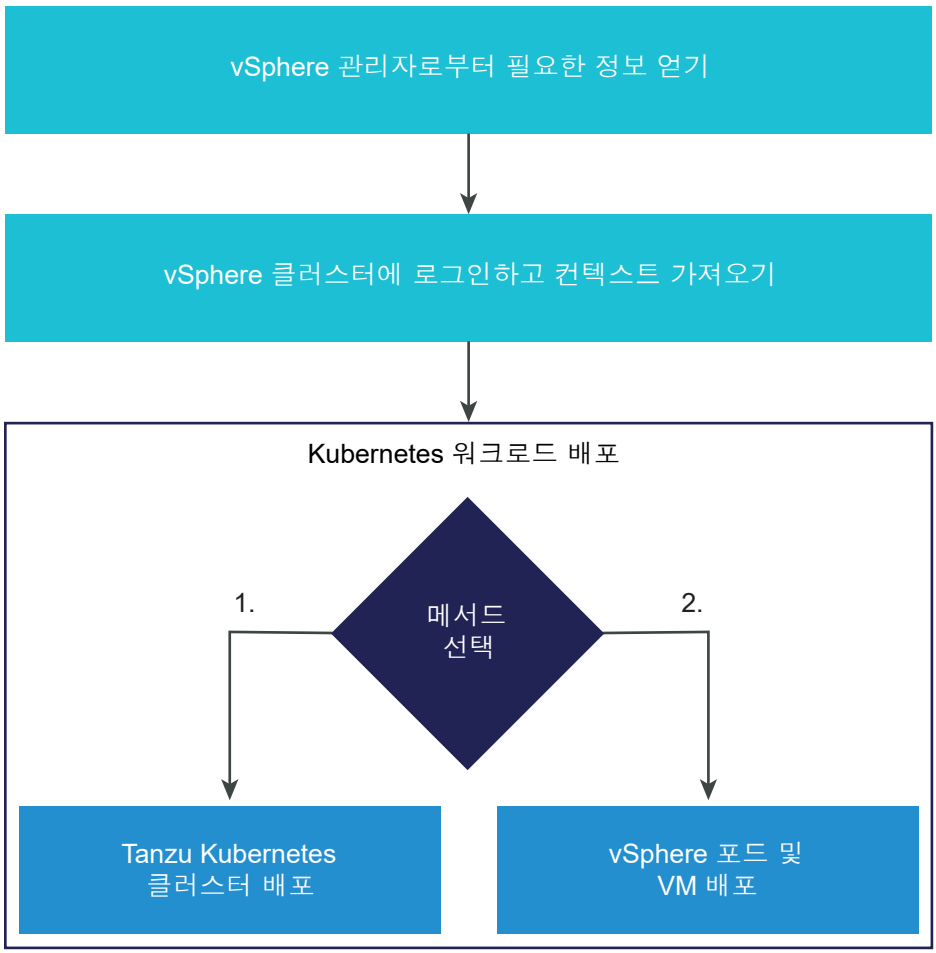
vSphere 관리자는 vSphere with Tanzu 플랫폼과 상호 작용하는 기본 인터페이스가 vSphere Client입니다. 개략적으로는 DevOps 엔지니어가 Kubernetes 워크로드를 배포할 수 있는 감독자 클러스터 및 네임스페이스를 구성해야 합니다. vSphere 및 NSX-T 기술에 대한 지식이 풍부하고 Kubernetes에 대한 기본적인 이해가 있어야 합니다.

그림 2-5. vSphere 관리자 개략적인 워크플로



DevOps 엔지니어는 Kubernetes 개발자 및 애플리케이션 소유자, Kubernetes 관리자이거나 두 기능을 모두 결합한 것일 수 있습니다. DevOps 엔지니어는 kubectl 명령을 사용하여 vSphere 포드, VM 및 Tanzu Kubernetes 클러스터를 감독자 클러스터의 기존 네임스페이스에 배포합니다. 일반적으로 DevOps 엔지니어는 vSphere 및 NSX-T의 전문가가 아니어도 되지만 vSphere 관리자와 보다 효율적으로 상호 작용할 수 있도록 이러한 기술과 vSphere with Tanzu 플랫폼에 대한 기본적인 이해가 있어야 합니다.

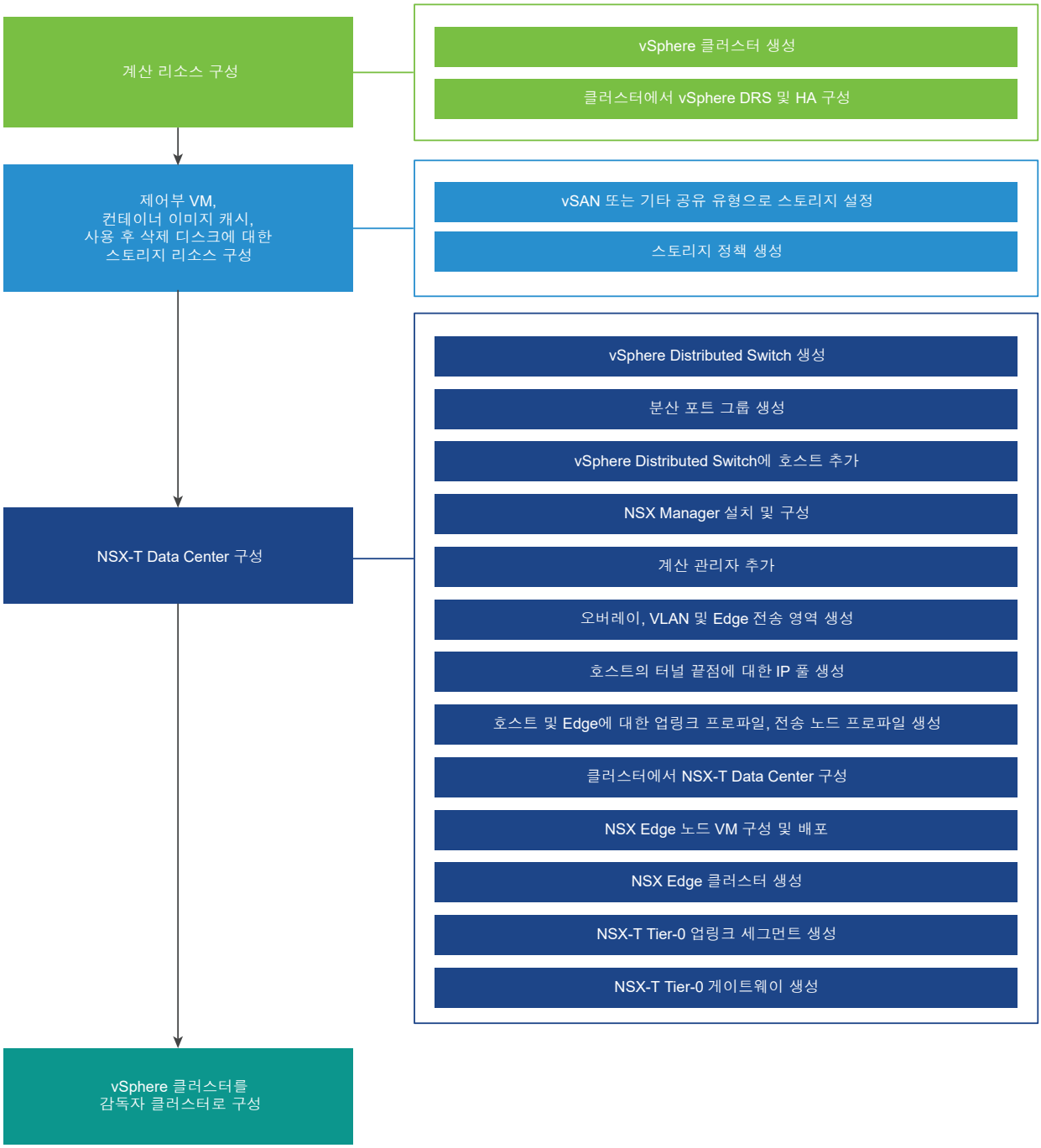
그림 2-6. DevOps 엔지니어 개략적인 워크플로



NSX-T Data Center가 포함된 감독자 클러스터 워크플로

vSphere 관리자는 필요한 계산, 스토리지 및 네트워킹 구성 요소가 포함된 vSphere with Tanzu 플랫폼을 구성합니다. NSX-T Data Center를 감독자 클러스터에 대한 네트워킹 스택으로 사용할 수 있습니다. 시스템 요구 사항에 대한 자세한 내용은 NSX-T Data Center를 사용하여 vSphere with Tanzu를 설정하기 위한 시스템 요구 사항의 내용을 참조하십시오.

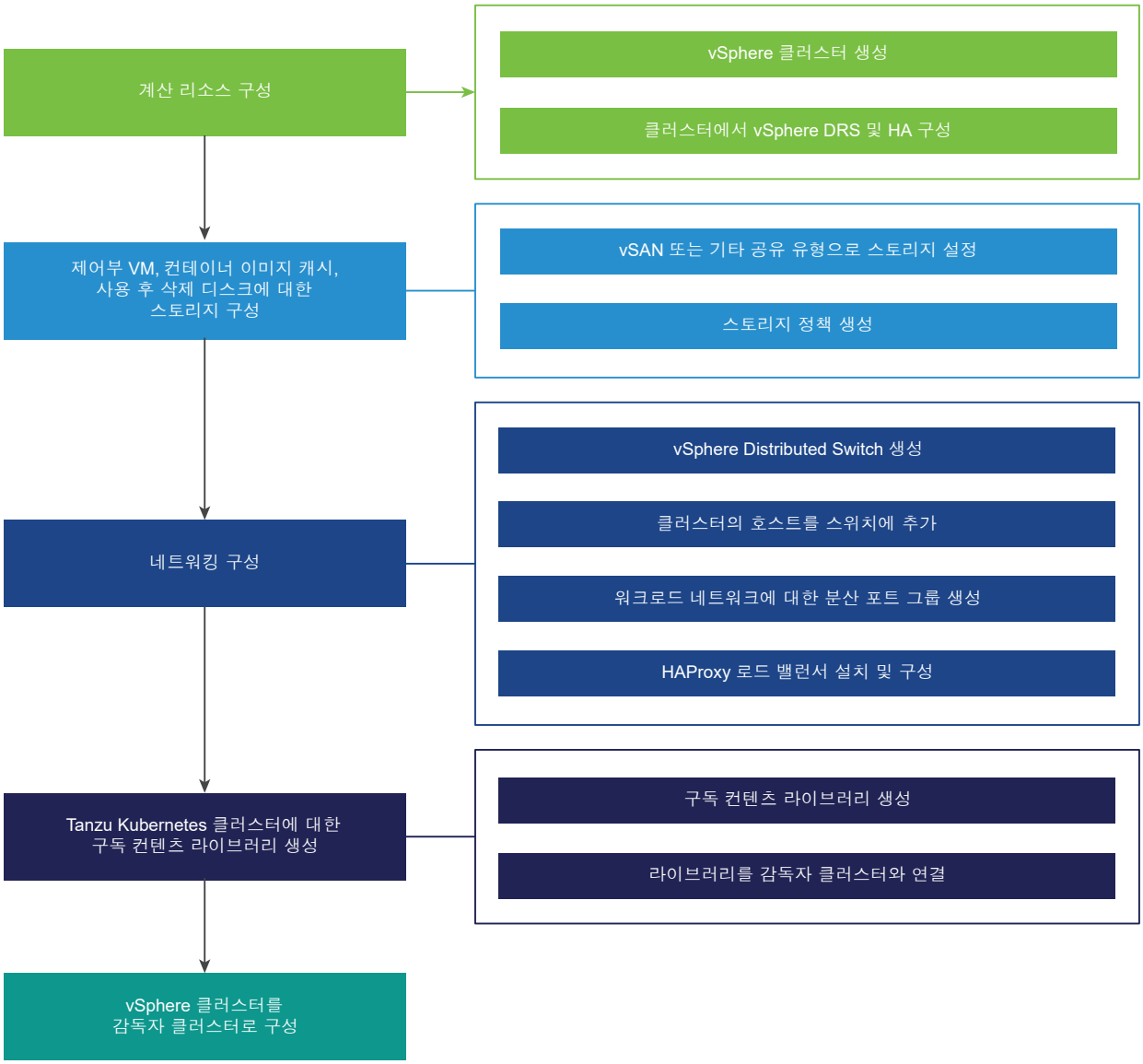
그림 2-7. NSX-T Data Center 네트워킹이 포함된 감독자 클러스터 워크플로



vSphere 네트워킹 스택이 포함된 감독자 클러스터 워크플로

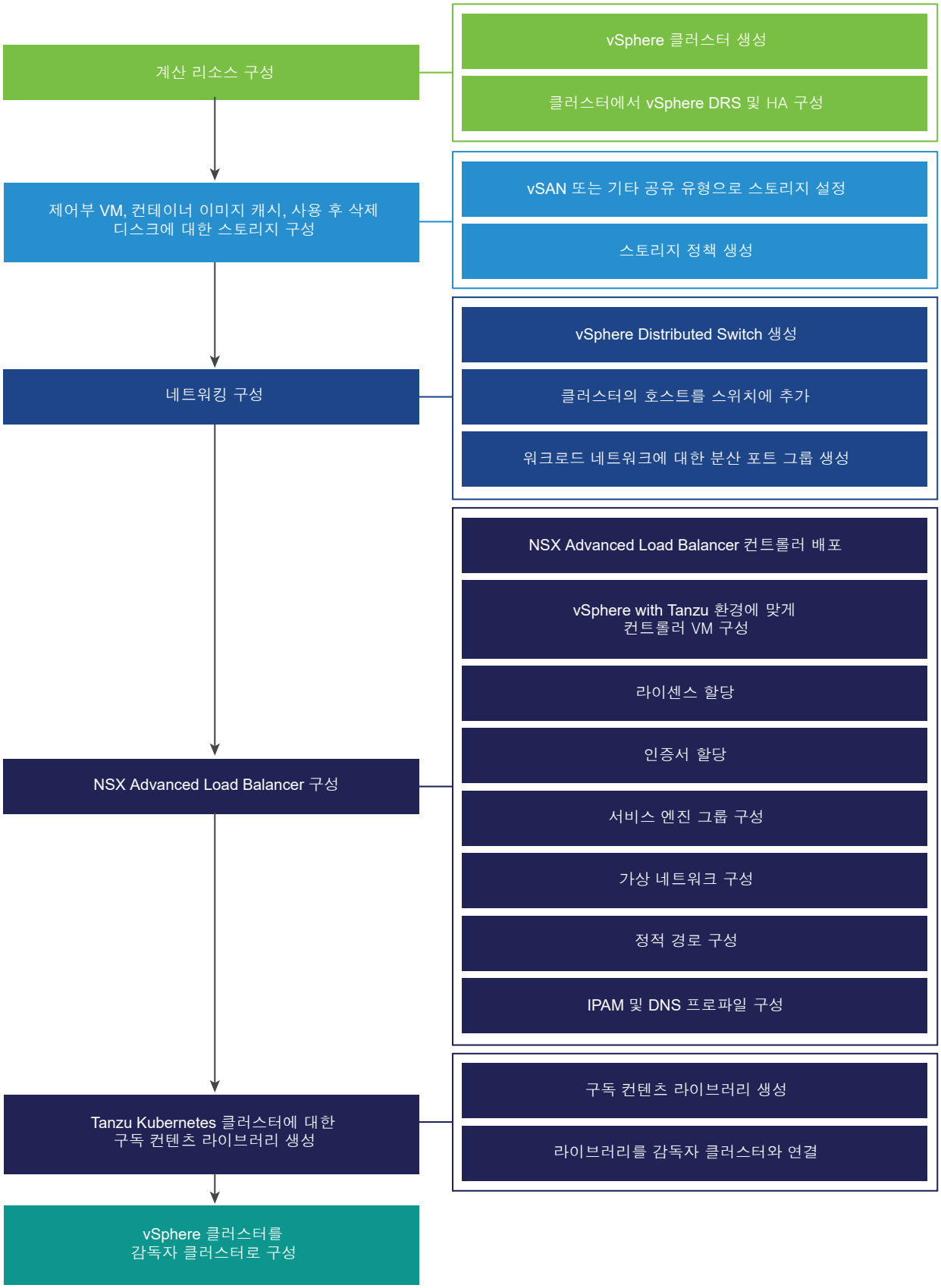
vSphere 관리자는 vSphere 네트워킹 스택이 포함된 감독자 클러스터로 vSphere 클러스터를 구성할 수 있습니다. 시스템 요구 사항에 대한 자세한 내용은 vSphere 네트워킹 및 HAProxy 로드 밸런서를 사용하여 vSphere with Tanzu를 설정하기 위한 시스템 요구 사항의 내용을 참조하십시오.

그림 2-8. vSphere 네트워킹 스택이 포함된 감독자 클러스터 구성 워크플로



vSphere 네트워킹 및 NSX Advanced Load Balancer 워크플로가 포함된 감독자 클러스터

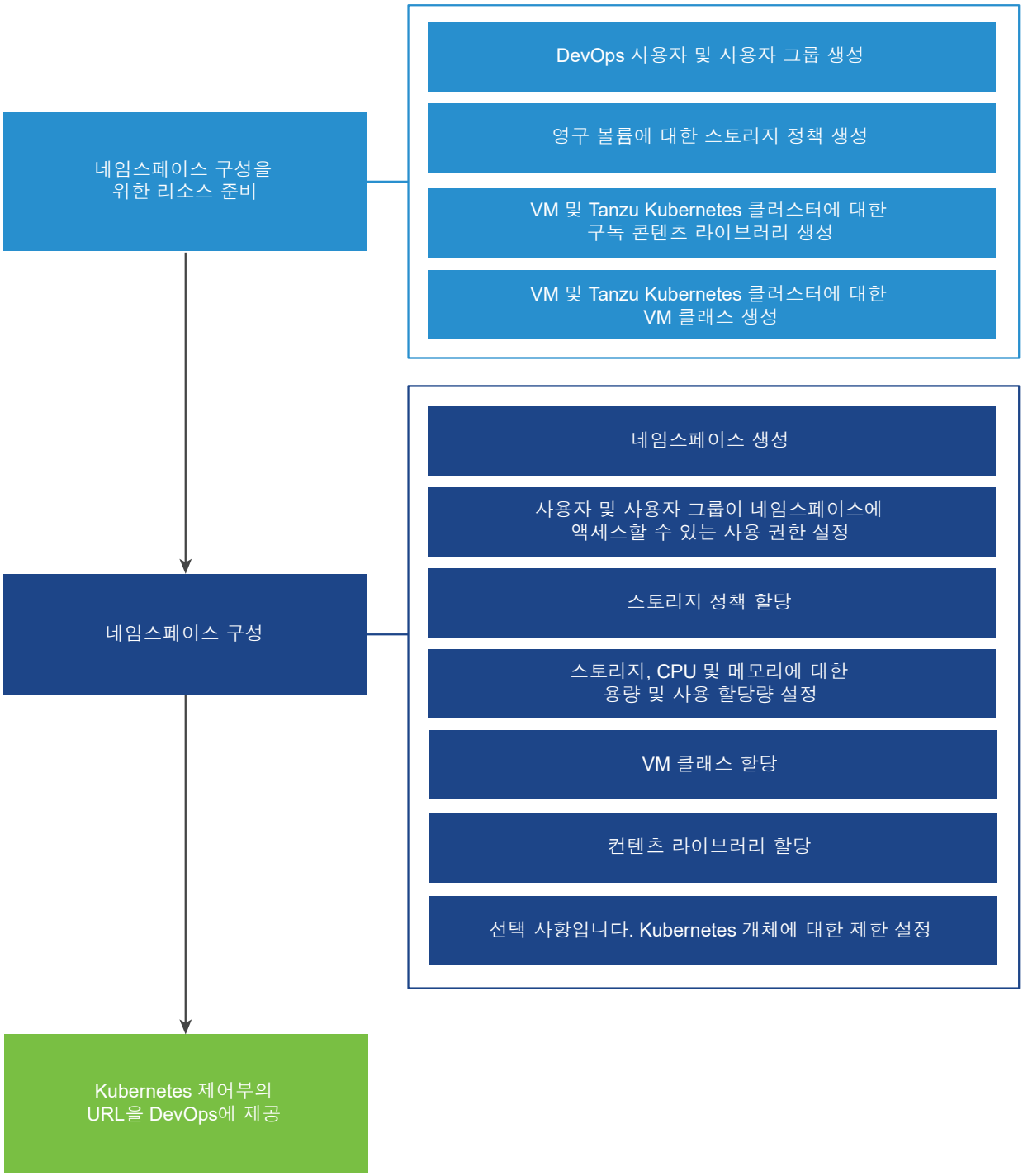
이 다이어그램은 vSphere with Tanzu에 대한 vSphere 네트워킹 및 NSX Advanced Load Balancer를 구성하는 워크플로를 보여줍니다. 자세한 내용은 NSX Advanced Load Balancer 설치 및 구성의 내용을 참조하십시오.


계산 리소스 구성
vSphere 클러스터 생성
클러스터에서 vSphere DRS 및 HA 구성
제어부 VM, 컨테이너 이미지 캐시, 사용 후 삭제 디스크에 대한 스토리지 구성
vSAN 또는 기타 공유 유형으로 스토리지 설정
스토리지 정책 생성
네트워킹 구성
vSphere Distributed Switch 생성
클러스터의 호스트를 스위치에 추가
워크로드 네트워크에 대한 분산 포트 그룹 생성
NSX Advanced Load Balancer 구성
NSX Advanced Load Balancer 컨트롤러 배포
vSphere with Tanzu 환경에 맞게 컨트롤러 VM 구성
라이센스 할당
인증서 할당
서비스 엔진 그룹 구성
가상 네트워크 구성
정적 경로 구성
IPAM 및 DNS 프로파일 구성
Tanzu Kubernetes 클러스터에 대한 구독 컨텐츠 라이브러리 생성
구독 컨텐츠 라이브러리 생성
라이브러리를 감독자 클러스터와 연결
vSphere 클러스터를 감독자 클러스터로 구성

네임스페이스 생성 및 구성 워크플로

vSphere 관리자는 감독자 클러스터에서 네임스페이스를 생성 및 구성합니다. DevOps 엔지니어로부터 실행할 애플리케이션 및 워크로드에 대한 특정 리소스 요구 사항을 수집하고 그에 따라 네임스페이스를 구성해야 합니다. 자세한 내용은 장 7 vSphere 네임스페이스 구성 및 관리을(를) 참조하십시오.

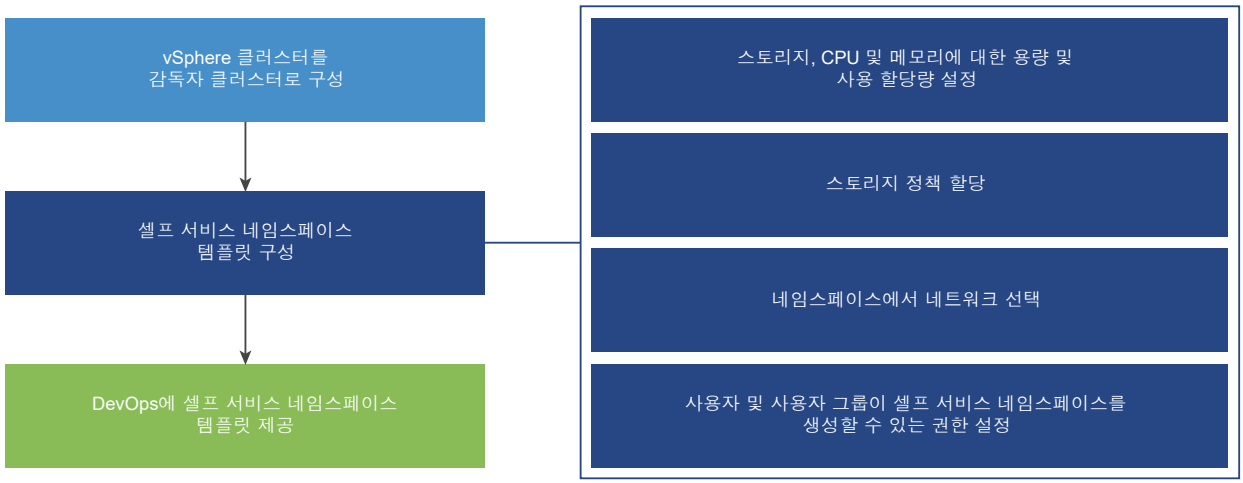
그림 2-9. 네임스페이스 구성 워크플로



셀프 서비스 네임스페이스 생성 및 구성 워크플로

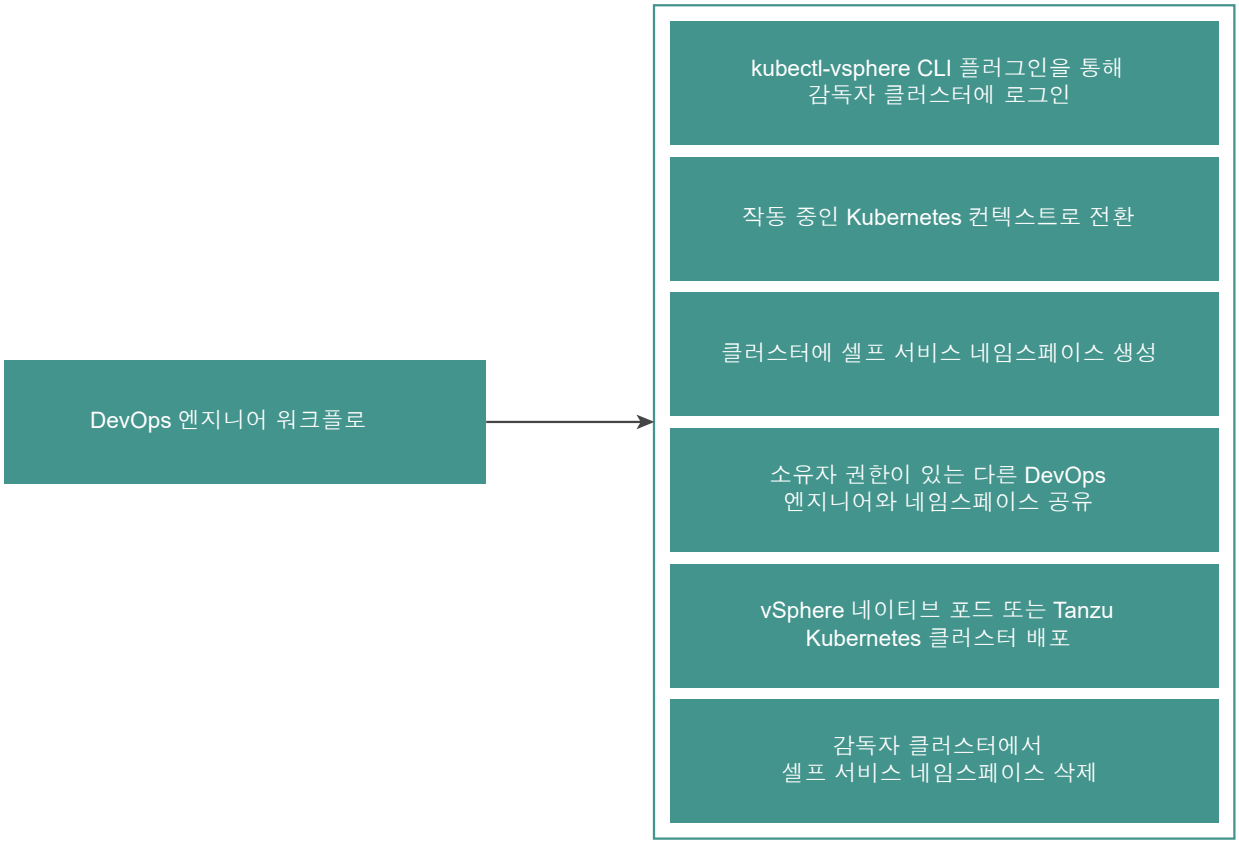
vSphere 관리자는 감독자 네임스페이스를 생성하고, CPU, 메모리 및 스토리지 제한을 네임스페이스에 설정하고, 사용 권한을 할당하고, 클러스터에서 네임스페이스 서비스를 템플릿으로 프로비저닝 또는 활성화할 수 있습니다.

그림 2-10. 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.

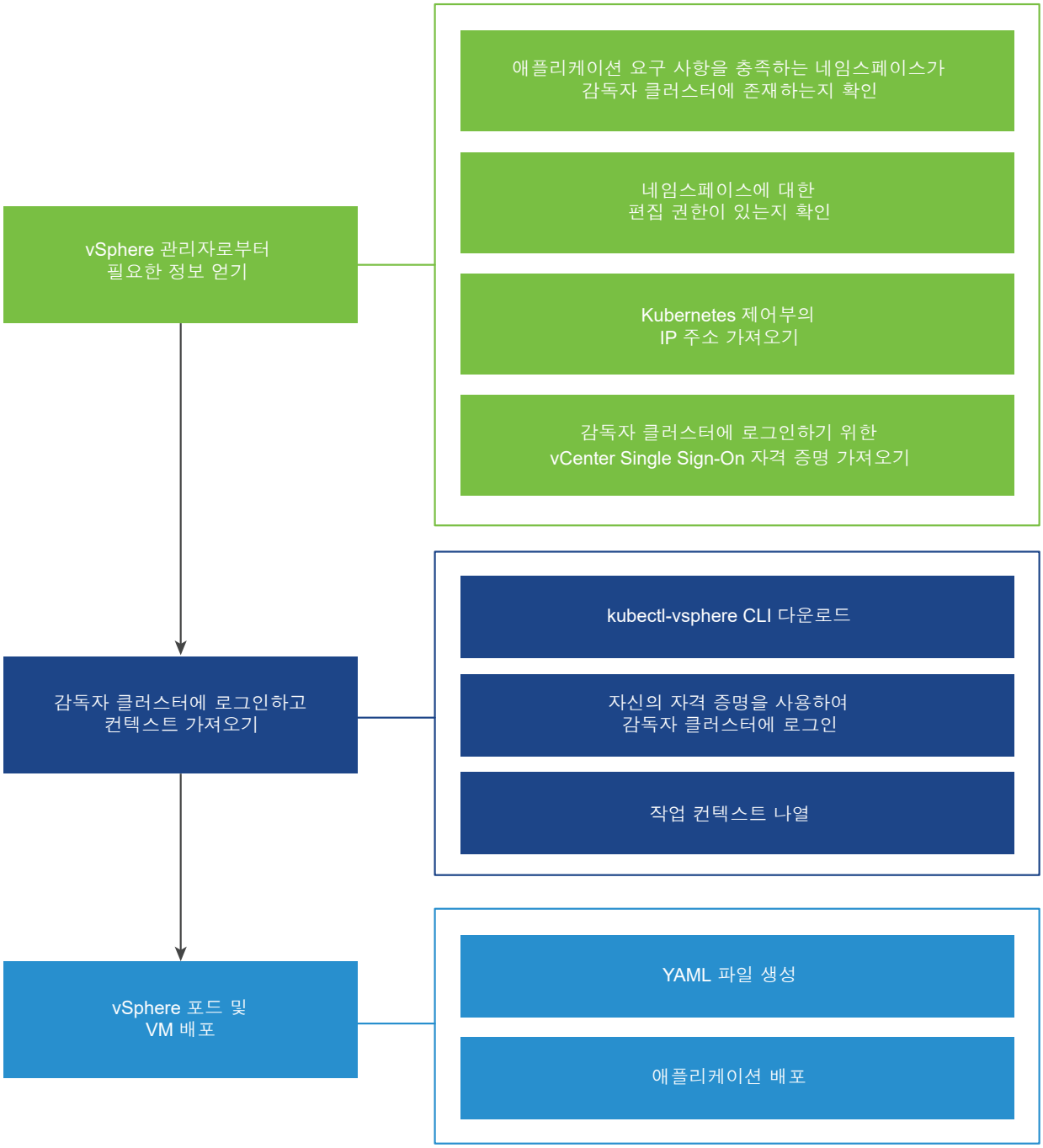
그림 2-11. 셀프 서비스 네임스페이스 생성 워크플로



vSphere 포드 및 VM 프로비저닝 워크플로

DevOps 엔지니어는 감독자 클러스터에서 실행 중인 네임스페이스의 리소스 경계 내에서 vSphere 포드 및 VM을 배포할 수 있습니다. 자세한 내용은 장 11 vSphere 포드에 워크로드 배포 및 장 12 vSphere with Tanzu에서 가상 시스템 배포 및 관리 항목을 참조하십시오.

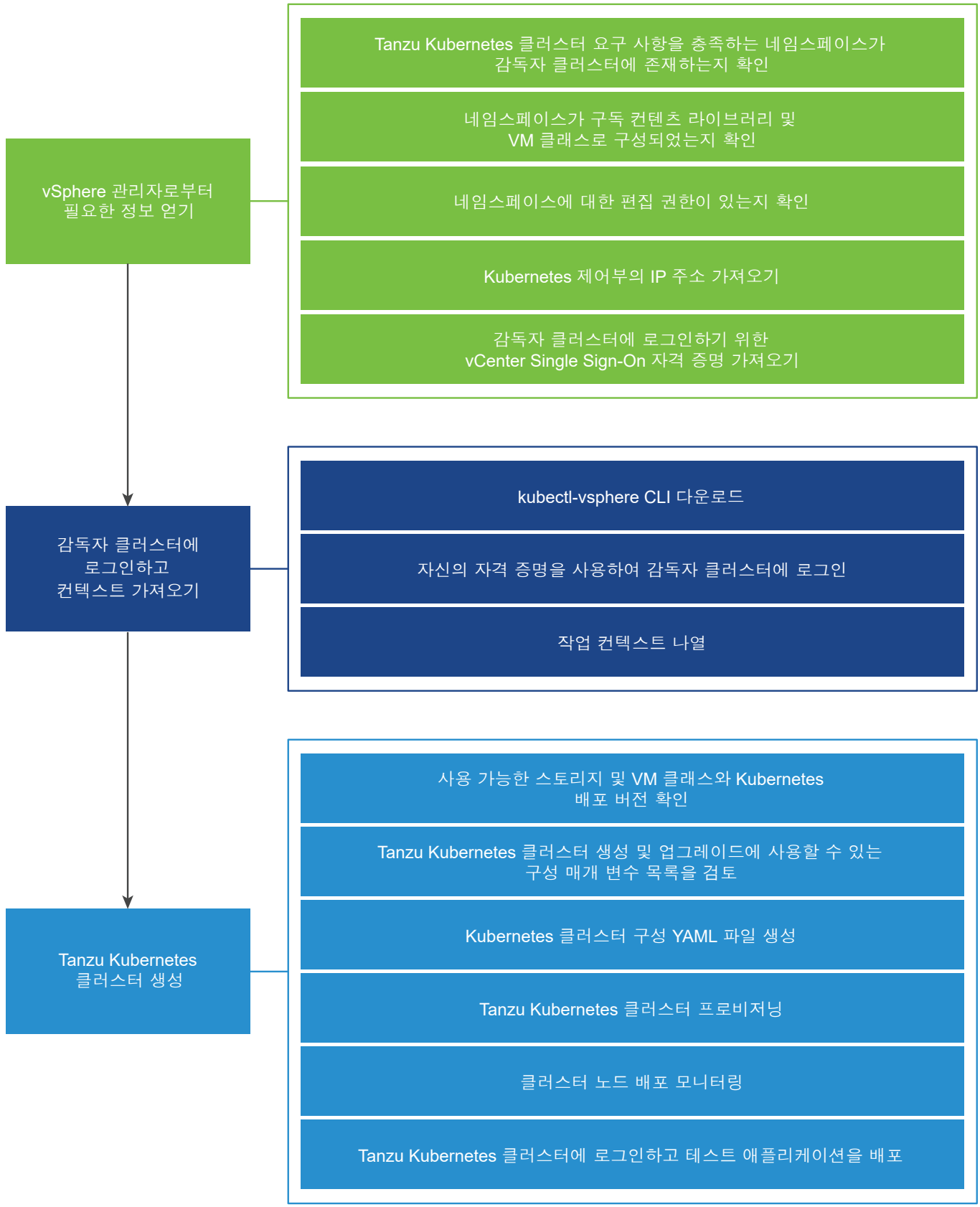
그림 2-12. vSphere 포드 및 VM 프로비저닝 워크플로



Tanzu Kubernetes 클러스터 프로비저닝 워크플로

DevOps 엔지니어는 vSphere 관리자가 생성하고 구성한 네임스페이스에 Tanzu Kubernetes 클러스터를 생성하고 구성합니다. 자세한 내용은 TKGS v1alpha2 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하는 워크플로의 내용을 참조하십시오.

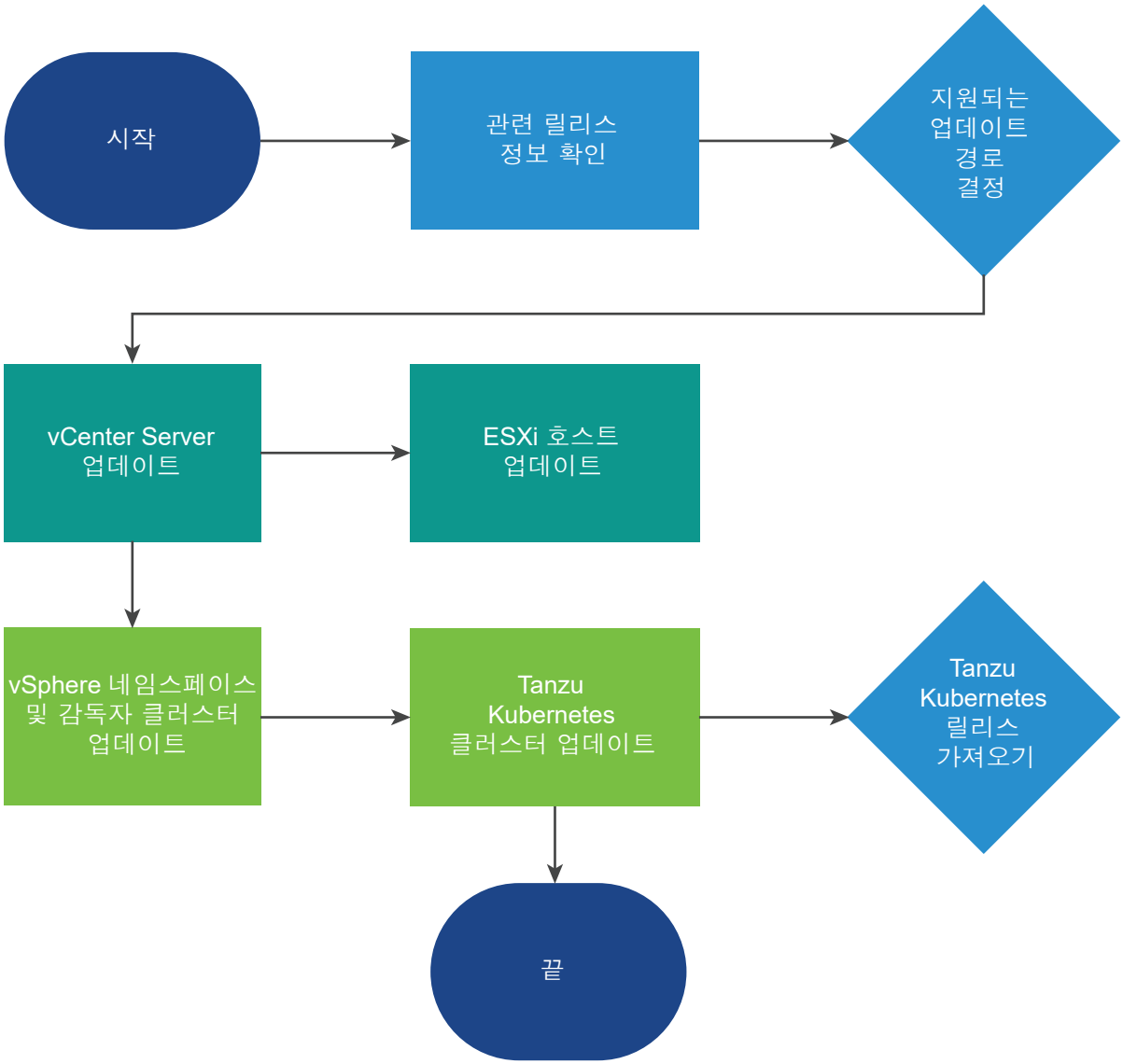
그림 2-13. Tanzu Kubernetes 클러스터 프로비저닝 워크플로



vSphere with Tanzu 업데이트 워크플로

이 다이어그램은 감독자 클러스터 및 Tanzu Kubernetes 클러스터를 포함한 vSphere with Tanzu 환경을 업데이트하는 워크플로를 보여줍니다. 자세한 내용은 장 17 vSphere with Tanzu 환경 업데이트의 내용을 참조하십시오.

그림 2-14. vSphere with Tanzu 업데이트 워크플로



vSphere with Tanzu이 vSphere 환경을 변경하는 방식

vSphere 클러스터가 Kubernetes 워크로드에 대해 구성되어 감독자 클러스터가 되면 Tanzu Kubernetes Grid 서비스를 사용하여 프로비저닝된 네임스페이스, vSphere 포드 및 Tanzu Kubernetes 클러스터와 같은 개체를 vCenter Server 인벤토리에 추가합니다.

각 감독자 클러스터에서 다음을 볼 수 있습니다.

- 클러스터에서 실행 중인 논리적 애플리케이션을 나타내는 네임스페이스.
- 감독자 클러스터의 각 네임스페이스에 대한 리소스 풀.

모든 네임스페이스 내에서 다음을 볼 수 있습니다.

- vSphere 포드.
- Tanzu Kubernetes Grid 서비스를 통해 생성된 Kubernetes 클러스터
- Kubernetes 제어부 VM.
- 네트워킹 및 스토리지 리소스.
- 해당 네임스페이스에 대한 사용자 권한입니다.

vSphere with Tanzu에 대한 라이센싱

vSphere with Tanzu에 대한 vSphere 클러스터를 구성하고 감독자 클러스터가 되면 60일 평가 기간이 만료되기 전에 클러스터에 Tanzu Edition 라이센스를 할당해야 합니다.

Tanzu 라이센스 정보

Tanzu 라이센스를 통해 vSphere 7.0 업데이트 1 이상에서 워크로드 관리 기능을 사용할 수 있습니다. vSphere 네트워킹 스택 또는 NSX-T Data Center로 구성된 감독자 클러스터에 적용할 수 있습니다. vSphere 7.0에서 실행되는 감독자 클러스터의 경우 감독자 클러스터의 각 호스트에 할당된 VMware vSphere 7 Enterprise Plus with Add-on for Kubernetes 라이센스가 필요합니다.

vSphere 관리자는 Tanzu 라이센스를 감독자 클러스터에 할당할 때 네임스페이스를 생성 및 구성하고 해당 네임스페이스에 대한 액세스를 DevOps 엔지니어에게 제공할 수 있습니다. DevOps 엔지니어는 액세스 권한이 있는 네임스페이스 내에 Tanzu Kubernetes 클러스터 및 vSphere 포드를 배포할 수 있습니다. 감독자 클러스터가 vSphere 네트워킹 스택으로 구성된 경우 여기에는 Tanzu Kubernetes 클러스터만 배포할 수 있습니다.

감독자 클러스터 라이센싱

감독자 클러스터를 배포하는 vSphere 클러스터에서 **워크로드 관리**를 사용하도록 설정하면 60일 평가 기간 내에 클러스터의 전체 기능 집합을 사용할 수 있습니다. 60일 평가 기간이 만료되기 전에 감독자 클러스터에 Tanzu 라이센스를 할당해야 합니다.

NSX-T Data Center를 감독자 클러스터에 대한 네트워킹 스택으로 구성하는 경우 NSX-T Data Center Advanced 이상 라이센스를 NSX Manager에 할당해야 합니다. NSX Advanced Load Balancer를 사용하여 vSphere 네트워킹 스택으로 감독자 클러스터를 구성하는 경우 Tanzu 라이센스 버전에 따라 로드 밸런서에 적합한 라이센스가 필요합니다.

환경이 vSphere 7.0 위에서 실행되고 감독자 클러스터를 vSphere 7.0 업데이트 1 이상으로 업그레이드하면 업그레이드가 완료된 후 클러스터가 평가 모드로 전환됩니다. 호스트에 할당된 VMware vSphere 7 Enterprise Plus with Add-on for Kubernetes 라이센스는 일반 vSphere Enterprise 7 Plus 라이센스로 작동하며 vSphere with Tanzu 기능을 사용하도록 설정하지 않습니다. 이 경우 60일 평가 기간이 만료되기 전에 감독자 클러스터에 Tanzu Edition 라이센스를 할당해야 합니다.

Tanzu 라이센스 만료

- vSphere 7.0 업데이트 3. vSphere 7.0 업데이트 3부터 Tanzu Edition 라이센스가 만료되면 유효한 라이센스를 획득할 때까지 vSphere with Tanzu의 전체 기능을 계속 사용할 수 있습니다. 하지만 새 감독자 클러스터에 만료된 라이센스를 할당할 수 없습니다. 60일 평가 기간이 만료되기 전에 새로 생성된 감독자 클러스터에 유효한 Tanzu Edition 라이센스를 할당해야 합니다.
- vSphere 7.0 업데이트 2 및 업데이트 1. vSphere 업데이트 2 또는 업데이트 1에서 실행되는 환경에서 Tanzu Edition 라이센스가 만료되면 vSphere 관리자는 새 네임스페이스를 생성하거나 감독자 클러스터의 Kubernetes 버전을 업데이트할 수 없습니다. DevOps 엔지니어는 새 워크로드를 배포할 수 없습니다. 기존 Tanzu Kubernetes 클러스터의 구성을 변경(예: 새 노드 추가)할 수 없습니다.

  Tanzu Kubernetes 클러스터에 워크로드를 계속 배포할 수 있으며 기존의 모든 워크로드는 예상대로 실행됩니다. 이미 배포된 모든 Kubernetes 워크로드는 정상적인 작업을 계속합니다.

Tanzu 라이센스 규정 준수

ESXi 호스트 라이센스와 유사하게, Tanzu 라이센스 키에는 CPU당 최대 32개의 코어가 포함된 CPU당 용량이 있습니다. Tanzu 라이센스를 감독자 클러스터에 할당하는 경우 사용되는 용량은 클러스터의 호스트에 있는 CPU 수와 각 CPU의 코어 수에 따라 결정됩니다. Tanzu Edition 라이센스 키를 한 번에 여러 감독자 클러스터에 할당할 수 있지만 여러 라이센스 키를 하나의 클러스터에 할당할 수는 없습니다.

- vSphere 7.0 업데이트 3. vSphere 7.0 업데이트 3부터 예를 들어 새 호스트를 추가하여 감독자 클러스터를 확장했을 때 클러스터에 할당한 라이센스 키의 용량이 부족해지면 동일한 라이센스 키를 계속 사용할 수 있습니다. 하지만 EULA 준수 상태를 유지하려면 감독자 클러스터의 모든 CPU 및 코어를 지원할 수 있는 충분한 용량의 새 라이센스 키를 확보해야 합니다.
- vSphere 7.0 업데이트 2 및 업데이트 1. vSphere with Tanzu 환경이 vSphere 7.0 업데이트 2 및 업데이트 1에서 실행되는 경우 감독자 클러스터의 총 CPU 수는 클러스터에 할당된 Tanzu Edition 라이센스의 CPU 용량을 초과하면 안 됩니다.

평가 기간 만료

감독자 클러스터의 평가 기간이 만료되면 vSphere 관리자는 새 네임스페이스를 생성하거나 감독자 클러스터의 Kubernetes 버전을 업데이트할 수 없습니다. DevOps 엔지니어는 새 워크로드를 배포할 수 없으며 기존 Tanzu Kubernetes 클러스터의 구성을 변경(예: 새 노드 추가)할 수 없습니다.

Tanzu Kubernetes 클러스터에 워크로드를 계속 배포할 수 있으며 기존의 모든 워크로드는 예상대로 실행됩니다. 이미 배포된 모든 Kubernetes 워크로드는 정상적인 작업을 계속합니다.

평가 기간 만료 동작은 vSphere 7.0 업데이트 2 및 업데이트 3 둘 다에 유효합니다.

vSphere with Tanzu 아키텍처 및 구성 요소

3

vSphere with Tanzu와 함께 사용되도록 설정된 클러스터를 감독자 클러스터라고 합니다. 이 클러스터는 vSphere 포드, VM 및 Tanzu Kubernetes 클러스터를 포함하는 워크로드를 실행하는 데 필요한 구성 요소와 리소스를 제공하는 vSphere with Tanzu의 기초에 있습니다.

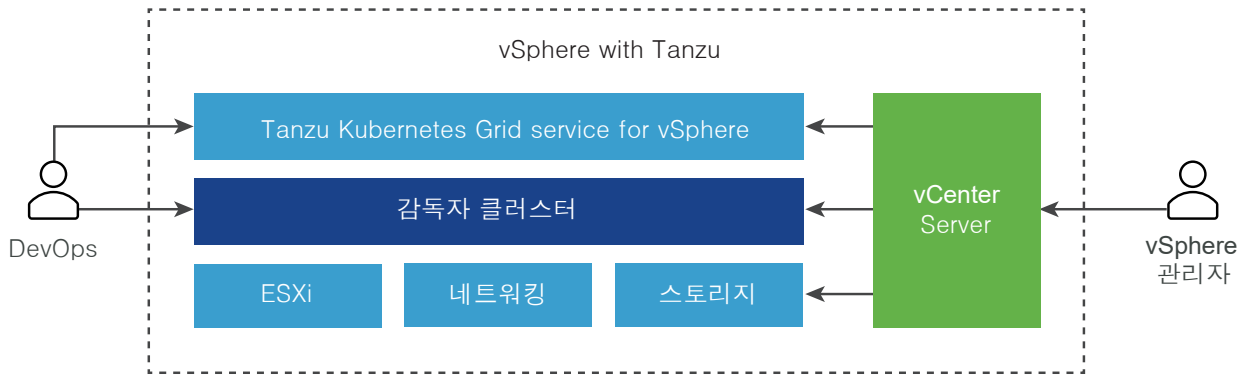
본 장은 다음 항목을 포함합니다.

- vSphere with Tanzu 아키텍처
- Tanzu Kubernetes Grid 서비스 아키텍처
- Tanzu Kubernetes 클러스터 테넌시 모델
- vSphere with Tanzu 인증
- vSphere with Tanzu 네트워킹
- vSphere with Tanzu 보안
- vSphere with Tanzu 스토리지

vSphere with Tanzu 아키텍처

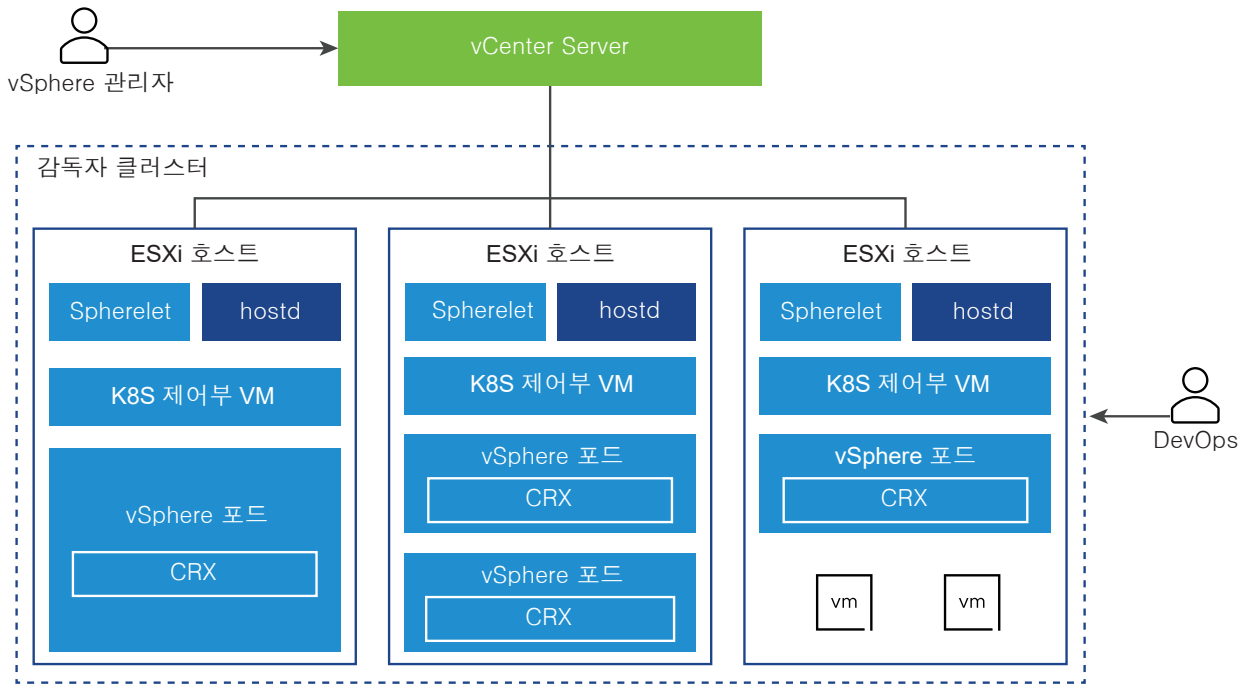
vSphere 클러스터에서 vSphere with Tanzu을 사용하도록 설정하면 하이퍼바이저 계층 내에 Kubernetes 제어부가 생성됩니다. 이 계층에는 ESXi 내에서 Kubernetes 워크로드를 실행하는 기능이 있는 특정 개체가 포함되어 있습니다.

그림 3-1. 감독자 클러스터 일반 아키텍처



vSphere with Tanzu에 사용하도록 설정된 클러스터를 감독자 클러스터라고 합니다. 이는 계산용 ESXi, NSX-T Data Center 또는 vSphere 네트워킹, vSAN 또는 다른 공유 스토리지 솔루션으로 구성된 SDDC 계층의 위에서 실행됩니다. 공유 스토리지는 vSphere 포드의 영구 볼륨, 감독자 클러스터 내에서 실행되는 VM, Tanzu Kubernetes 클러스터의 포드에 사용됩니다. 감독자 클러스터가 생성되면 vSphere 관리자는 vSphere 네임스페이스라는 감독자 클러스터 내에 네임스페이스를 생성할 수 있습니다. DevOps 엔지니어는 vSphere 포드 내에서 실행되는 컨테이너로 구성된 워크로드를 실행하고 Tanzu Kubernetes 클러스터를 생성할 수 있습니다.

그림 3-2. 감독자 클러스터 아키텍처



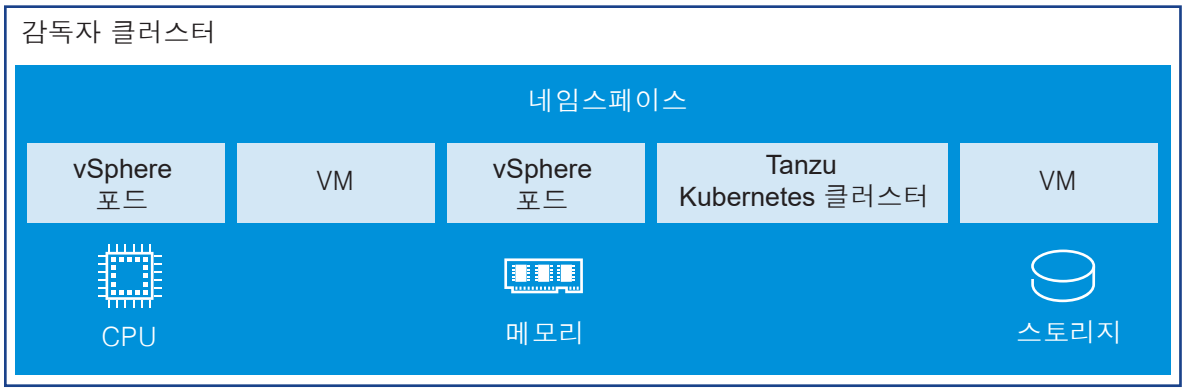
- Kubernetes 제어부 VM. 총 3개의 Kubernetes 제어부 VM이 감독자 클러스터의 일부인 호스트에 생성됩니다. 3개의 제어부 VM 각각에는 고유한 IP 주소가 있으므로 로드 균형이 조정됩니다. 또한 VM 중 하나에는 부동 IP 주소가 할당됩니다. vSphere DRS는 ESXi 호스트에서 제어부 VM의 정확한 배치를 결정하고 필요할 때 마이그레이션합니다. vSphere DRS는 제어부 VM의 Kubernetes 스케줄러와도 통합되므로 DRS가 vSphere 포드의 배치를 결정합니다. DevOps 엔지니어가 vSphere 포드을 스케줄링하는 경우, 요청은 일반 Kubernetes 워크플로를 거쳐서 DRS로 이동하여 최종 배치 결정을 내립니니다.
- Spherelet. Spherelet이라는 추가 프로세스는 각 호스트에 생성됩니다. kubelet은 기본적으로 ESXi로 이식되고 ESXi 호스트가 Kubernetes 클러스터의 일부가 되도록 허용합니다.
- CRX(Container Runtime Executive). CRX는 Hostd와 vCenter Server의 관점에서 VM과 유사합니다. CRX에는 하이퍼바이저와 함께 작동하는 반가상화 Linux 커널이 포함되어 있습니다. CRX는 VM과 동일한 하드웨어 가상화 기술을 사용하며 주변에 VM 경계가 있습니다. 직접 부팅 기술이 사용되기 때문에 CRX의 Linux 게스트가 커널 초기화를 통해 전달하지 않고도 기본 초기화 프로세스를 시작할 수 있습니다. 따라서 vSphere 포드가 거의 컨테이너만큼 빠르게 부팅할 수 있습니다.

- 클러스터 API 및 VMware Tanzu™ Kubernetes Grid™ 서비스는 감독자 클러스터에서 실행되고 Tanzu Kubernetes 클러스터의 프로비저닝 및 관리를 사용하도록 설정하는 모듈입니다. 가상 시스템 서비스 모듈은 독립형 VM 및 Tanzu Kubernetes 클러스터를 구성하는 VM의 배포 및 실행을 담당합니다.

vSphere 네임스페이스

vSphere 네임스페이스는 Tanzu Kubernetes Grid 서비스를 사용하여 생성된 vSphere 포드 및 Tanzu Kubernetes 클러스터가 실행될 수 있는 리소스 경계를 설정합니다. 처음 생성되면 네임스페이스는 감독자 클러스터 내에 제한 없는 리소스가 포함됩니다. vSphere 관리자는 CPU, 메모리, 스토리지는 물론 네임스페이스 내에서 실행할 수 있는 Kubernetes 개체의 수에 대한 제한을 설정할 수 있습니다. vSphere의 각 네임스페이스별로 리소스 풀이 생성됩니다. 스토리지 제한은 Kubernetes에서 스토리지 할당량으로 표시됩니다.

그림 3-3. vSphere 네임스페이스



DevOps 엔지니어에게 네임스페이스에 대한 액세스를 제공하기 위해, vSphere 관리자는 vCenter Single Sign-On에 연결된 ID 소스 내에서 사용 가능한 사용자나 사용자 그룹에 사용 권한을 할당합니다.

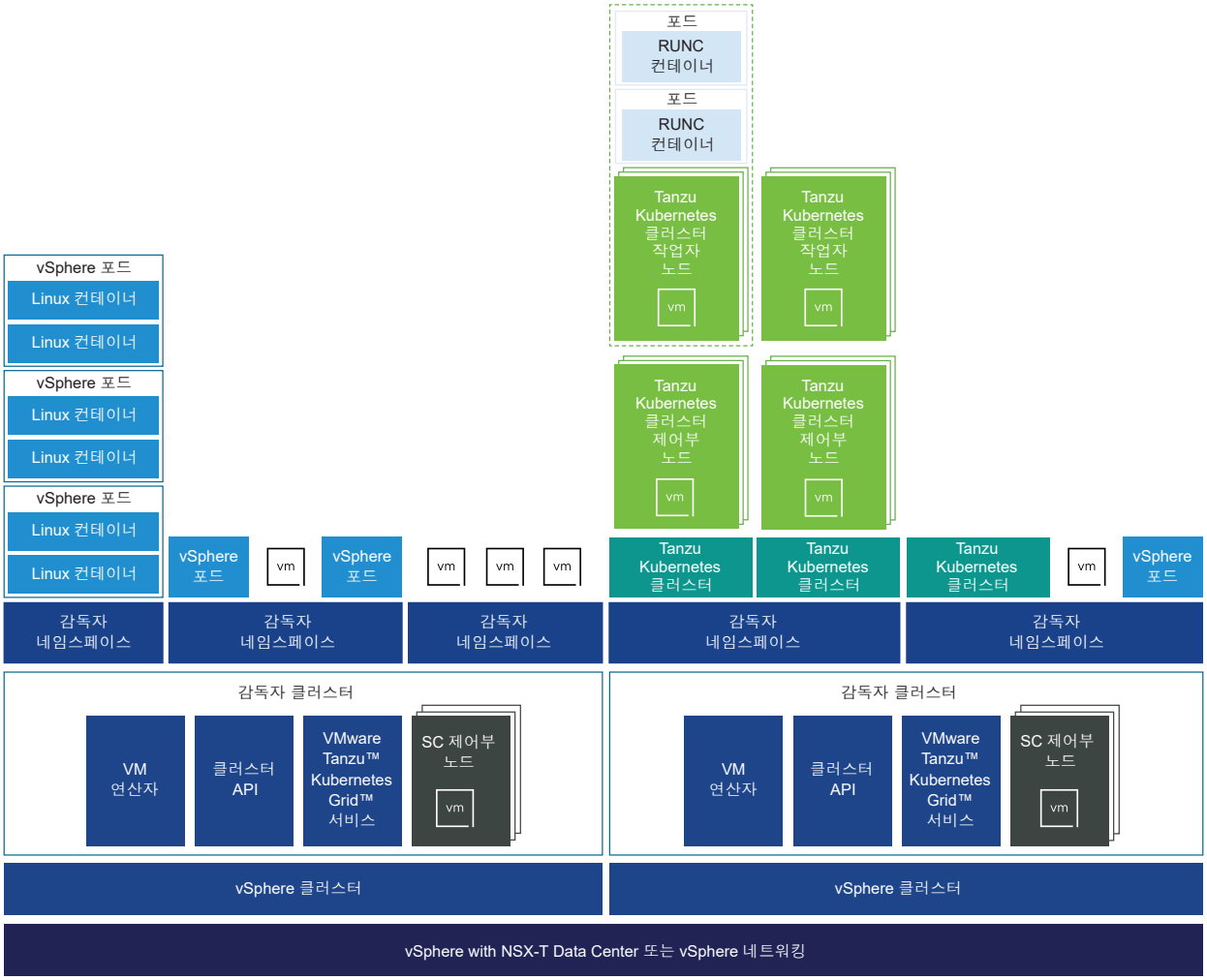
리소스 및 개체 제한은 물론 사용 권한 및 스토리지 정책으로 네임스페이스를 생성하고 구성한 후, DevOps 엔지니어가 네임스페이스에 액세스하여 Kubernetes 워크로드를 실행하고 Tanzu Kubernetes Grid 서비스를 사용하여 Tanzu Kubernetes 클러스터를 생성할 수 있습니다.

Tanzu Kubernetes 클러스터

Tanzu Kubernetes 클러스터는 VMware에서 패키지, 서명 및 지원하는 오픈 소스 Kubernetes 소프트웨어의 전체 배포입니다. vSphere with Tanzu의 컨텍스트에서, Tanzu Kubernetes Grid 서비스를 사용하여 감독자 클러스터에 Tanzu Kubernetes 클러스터를 프로비저닝할 수 있습니다. kubectl 및 YAML 정의를 사용하여 Tanzu Kubernetes Grid 서비스 API를 선언적으로 호출할 수 있습니다.

Tanzu Kubernetes 클러스터는 vSphere 네임스페이스에 상주합니다. 표준 Kubernetes 클러스터와 동일한 도구를 사용하여 워크로드 및 서비스를 동일한 방식으로 Tanzu Kubernetes에 배포할 수 있습니다.

그림 3-4. Tanzu Kubernetes 클러스터의 vSphere with Tanzu 아키텍처



vSphere 네트워킹 스택으로 구성된 감독자 클러스터

vSphere 네트워킹 스택으로 구성된 감독자 클러스터는 Tanzu Kubernetes Grid 서비스를 사용하여 생성된 Tanzu Kubernetes 클러스터 실행만 지원합니다. 클러스터는 vSphere 네트워크 서비스 및 스토리지 서비스도 지원합니다.

vSphere 네트워킹 스택으로 구성된 감독자 클러스터는 vSphere 포드를 지원하지 않습니다. 따라서 Spherelet 구성 요소는 이러한 감독자 클러스터에서 사용할 수 없으며 Kubernetes 포드는 Tanzu Kubernetes 클러스터 내에서만 실행됩니다. vSphere 네트워킹 스택으로 구성된 감독자 클러스터는 Harbor 레지스트리도 지원하지 않습니다. 이 서비스는 vSphere 포드에서만 사용되기 때문입니다.

vSphere 네트워킹 스택으로 구성된 클러스터에서 생성된 vSphere 네임스페이스도 vSphere 포드 실행을 지원하지 않으며 Tanzu Kubernetes 클러스터만 지원합니다.

Tanzu Kubernetes Grid 서비스 아키텍처

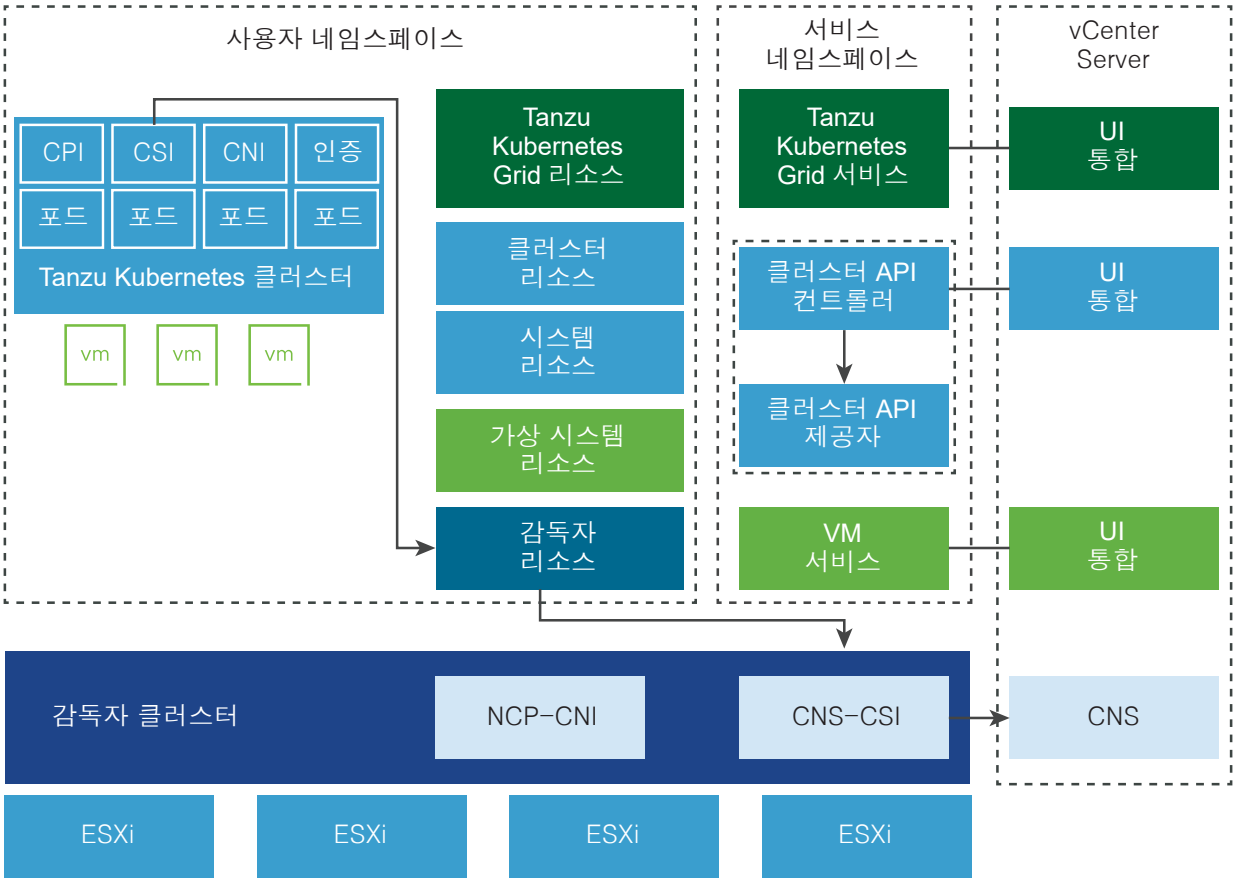
Tanzu Kubernetes Grid 서비스는 Tanzu Kubernetes 클러스터의 셀프 서비스 수명 주기 관리를 제공합니다. Tanzu Kubernetes Grid 서비스를 사용하여 Kubernetes 운영자와 개발자에게 친숙한 선언적 방식으로 Tanzu Kubernetes 클러스터를 만들고 관리할 수 있습니다.

Tanzu Kubernetes Grid 서비스 구성 요소

Tanzu Kubernetes Grid 서비스는 Tanzu Kubernetes 클러스터의 수명 주기를 관리하기 위한 세 계층의 컨트롤러를 노출합니다.

- Tanzu Kubernetes Grid 서비스는 기본 vSphere 네임스페이스 리소스와 통합하는 데 필요한 구성 요소를 포함하는 클러스터를 프로비저닝합니다. 이러한 구성 요소에는 감독자 클러스터와 통합되는 클라우드 제공자 플러그인이 포함됩니다. 또한 Tanzu Kubernetes 클러스터는 VMware CNS(클라우드 네이티브 스토리지)와 통합된 감독자 클러스터에 영구 볼륨에 대한 요청을 전달합니다. 장 10 vSphere with Tanzu에서 영구 스토리지 사용의 내용을 참조하십시오.
- 클러스터 API는 클러스터 생성, 구성 및 관리를 위한 선언적 Kubernetes 스타일의 API를 제공합니다. 클러스터 API에 대한 입력에는 클러스터를 설명하는 리소스, 클러스터를 구성하는 가상 시스템을 설명하는 리소스 집합 및 클러스터 추가 기능을 설명하는 리소스 집합이 포함됩니다.
- 가상 시스템 서비스는 VM 및 연결된 vSphere 리소스 관리를 위한 선언적 Kubernetes 스타일의 API를 제공합니다. 가상 시스템 서비스는 재사용 가능한 추상적인 하드웨어 구성을 나타내는 가상 시스템 클래스의 개념을 소개합니다. 가상 시스템 서비스에서 제공하는 기능은 Tanzu Kubernetes 클러스터를 호스팅하는 제어부 및 작업자 노드 VM의 수명 주기를 관리하는 데 사용됩니다.

그림 3-5. Tanzu Kubernetes Grid 서비스 아키텍처 및 구성 요소



Tanzu Kubernetes 클러스터 구성 요소

Tanzu Kubernetes 클러스터에서 실행되는 구성 요소는 인증 및 권한 부여, 스토리지 통합, 포드 네트워킹 및 로드 밸런싱의 네 가지 영역에 걸쳐 있습니다.

- 인증 Webhook: 사용자 인증 토큰을 검증하기 위해 클러스터 내부의 포드로 실행되는 Webhook입니다.
- 컨테이너 스토리지 인터페이스 플러그인: 감독자 클러스터를 통해 CNS와 통합되는 반가상화 CSI 플러그인입니다.
- 컨테이너 네트워크 인터페이스 플러그인: 포드 네트워킹을 제공하는 CNI 플러그인입니다.
- 클라우드 제공자 구현: Kubernetes 로드 밸런서 서비스 생성을 지원합니다.

Tanzu Kubernetes Grid 서비스 API

Tanzu Kubernetes Grid 서비스 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하고 관리할 수 있습니다. kubectl 및 YAML을 사용하여 호출하는 선언적 API입니다.

선언적 API를 사용하여 시스템에 대한 명령형 명령을 수행하는 대신 Tanzu Kubernetes 클러스터의 원하는 상태(노드 수, 사용 가능한 스토리지, VM 크기, Kubernetes 소프트웨어 버전)를 지정합니다. Tanzu Kubernetes Grid 서비스는 원하는 상태와 일치하는 클러스터를 프로비저닝하는 작업을 수행합니다.

Tanzu Kubernetes Grid 서비스 API를 호출하려면 YAML 파일을 사용하여 kubectl을 호출하고 이를 통해 API를 호출합니다. 클러스터가 생성되면 YAML을 업데이트하여 클러스터를 업데이트합니다.

Tanzu Kubernetes Grid 서비스 인터페이스

vSphere 관리자는 vSphere Client를 사용하여 vSphere 네임스페이스를 구성하고 사용 권한을 부여합니다. 또한 클러스터 구성 요소에서 사용하는 리소스를 모니터링하고 vSphere 인벤토리에서 해당 리소스의 관련 정보를 봅니다.

DevOps 엔지니어는 kubectl용 vSphere 플러그인을 사용하여 vCenter Single Sign-On 자격 증명으로 vSphere 네임스페이스에 연결합니다. 연결 후 DevOps 엔지니어는 kubectl을 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝합니다.

개발자는 kubectl용 vSphere 플러그인 및 해당 vCenter Single Sign-On 자격 증명을 사용하여 프로비저닝된 클러스터에 연결할 수 있습니다. 또는 클러스터 관리자가 지원되는 Kubernetes 인증 제공자를 구성한 경우 개발자는 kubectl을 사용하여 연결할 수 있습니다. Kubernetes에 워크로드를 배포하고 클러스터 환경과 상호 작용하기 위해 개발자는 kubectl을 사용합니다.

Tanzu Kubernetes Grid 서비스 데모

다음 비디오를 시청하면 Tanzu Kubernetes Grid 서비스를 사용하여 Tanzu Kubernetes 클러스터를 생성하고 작동하는 방법을 알아볼 수 있습니다. vSphere 7 with Kubernetes - Tanzu Kubernetes 클러스터 - 기술 개요.

Tanzu Kubernetes 클러스터 테넌시 모델

감독자 클러스터는 Tanzu Kubernetes Grid 서비스에서 프로비저닝된 Tanzu Kubernetes 클러스터의 관리부입니다. 테넌시 모델은 Tanzu Kubernetes 클러스터가 상주하는 vSphere 네임스페이스를 사용하여 적용됩니다.

감독자 클러스터

감독자 클러스터는 Tanzu Kubernetes 클러스터가 구축되는 관리 계층을 제공합니다. Tanzu Kubernetes Grid 서비스는 감독자 클러스터에 속하는 컨트롤러 집합이 있는 사용자 지정 컨트롤러 관리자입니다. Tanzu Kubernetes Grid 서비스의 목적은 Tanzu Kubernetes 클러스터를 프로비저닝하는 것입니다.

감독자 클러스터와 vSphere 클러스터 사이에는 일대일 관계가 있지만 감독자 클러스터와 Tanzu Kubernetes 클러스터 사이에는 일대다 관계가 있습니다. 단일 감독자 클러스터 내에 여러 Tanzu Kubernetes 클러스터를 프로비저닝할 수 있습니다. 감독자 클러스터에서 제공하는 워크로드 관리 기능을 사용하면 클러스터 구성 및 수명 주기를 제어하고 업스트림 Kubernetes와의 동시성을 유지할 수 있습니다.

자세한 내용은 장 5 감독자 클러스터 구성 및 관리의 내용을 참조하십시오.

vSphere 네임스페이스

vSphere 네임스페이스에 하나 이상의 Tanzu Kubernetes 클러스터를 배포합니다. 리소스 할당량과 스토리지 정책은 vSphere 네임스페이스에 적용되고 여기에 배포된 Tanzu Kubernetes 클러스터에 상속됩니다.

Tanzu Kubernetes 클러스터를 프로비저닝할 때 리소스 풀과 VM 폴더가 vSphere 네임스페이스에 생성됩니다. Tanzu Kubernetes 클러스터 제어부와 작업자 노드 VM은 이 리소스 풀과 VM 폴더 내에 배치됩니다. vSphere Client를 사용하면 **호스트 및 클러스터** 관점을 선택하여 이 계층을 볼 수 있고 **VM 및 템플릿** 보기를 선택하여 볼 수도 있습니다.

자세한 내용은 장 7 vSphere 네임스페이스 구성 및 관리의 내용을 참조하십시오.

컨텐츠 라이브러리

vSphere 컨텐츠 라이브러리는 Tanzu Kubernetes 클러스터 노드를 생성하는 데 사용되는 가상 시스템 템플릿을 제공합니다. Tanzu Kubernetes 클러스터를 배포하려는 각 감독자 클러스터에 대해, Tanzu Kubernetes Grid 서비스가 클러스터 노드를 구축하는 데 사용하는 OVA를 소스로 하는 구독 컨텐츠 라이브러리 개체를 정의해야 합니다. 여러 감독자 클러스터에 대해 동일한 구독 컨텐츠 라이브러리를 구성할 수 있습니다. 구독 컨텐츠 라이브러리와 vSphere 네임스페이스 간에는 관계가 없습니다. 구독 컨텐츠 라이브러리는 VMware에서 직접 최신 템플릿을 다운로드합니다. 사용할 OVA 템플릿을 로컬 컨텐츠 라이브러리에 직접 업로드합니다.

자세한 내용은 Tanzu Kubernetes 릴리스용 컨텐츠 라이브러리 생성 및 관리의 내용을 참조하십시오.

vSphere with Tanzu 인증

vSphere 관리자는 감독자 클러스터를 구성하고 네임스페이스를 관리하기 위한 권한이 필요합니다. 네임스페이스에 대한 사용 권한을 정의하여 네임스페이스에 액세스할 수 있는 DevOps 엔지니어를 결정합니다. DevOps 엔지니어는 vCenter Single Sign-On 자격 증명을 사용하여 감독자 클러스터를 인증하며, 사용 권한이 있는 네임스페이스에만 액세스할 수 있습니다.

vSphere 관리자에 대한 사용 권한

vSphere 관리자가 vSphere 클러스터를 감독자 클러스터로 구성하고 네임스페이스를 생성 및 관리하려면 vSphere 클러스터에 대한 사용 권한이 필요합니다. vSphere 관리자는 vSphere 클러스터의 사용자 계정과 연결된 다음 권한 중 하나 이상이 있어야 합니다.

- **네임스페이스 구성 수정**. 감독자 클러스터에서 네임스페이스를 생성하고 구성할 수 있습니다.
- **클러스터 전체 구성 수정**. vSphere 클러스터를 감독자 클러스터로 구성할 수 있습니다.

DevOps 엔지니어에 대한 사용 권한 설정

vSphere 관리자는 네임스페이스 수준에서 사용자 계정에 보기, 편집 또는 소유자 권한을 부여합니다. 사용자 계정은 vCenter Single Sign-On에 연결된 ID 소스에서 사용할 수 있어야 합니다. 하나의 사용자 계정이 여러 네임스페이스에 액세스할 수 있습니다. 관리자 그룹의 멤버인 사용자는 감독자 클러스터의 모든 네임스페이스에 액세스할 수 있습니다.

사용 권한, 리소스 할당량 및 스토리지가 포함된 네임스페이스를 구성한 후에는 Kubernetes 제어부의 URL을 DevOps 엔지니어에게 제공합니다. DevOps 엔지니어는 이 URL을 사용하여 제어부에 로그인할 수 있습니다. 로그인하면 DevOps 엔지니어가 vCenter Server 시스템에 속한 모든 감독자 클러스터에서 사용 권한이 있는 모든 네임스페이스에 액세스할 수 있습니다. vCenter Server 시스템이 고급 연결 모드인 경우 DevOps 엔지니어는 링크드 모드 그룹에서 사용 가능한 모든 감독자 클러스터에서 사용 권한이 있는 모든 네임스페이스에 액세스할 수 있습니다. Kubernetes 제어부의 IP 주소는 VDS 네트워킹 스택에서 사용 중인 로드 밸런서 또는 NSX-T에서 생성되는 가상 IP로, Kubernetes 제어부에 대한 액세스 지점으로 사용됩니다.

소유자 권한이 있는 DevOps 엔지니어는 워크로드를 배포할 수 있습니다. 다른 DevOps 엔지니어 또는 그룹과 네임스페이스를 공유하고 더 이상 필요하지 않을 때 삭제할 수 있습니다. DevOps 엔지니어가 네임스페이스를 공유할 때 다른 DevOps 엔지니어 및 그룹에 보기, 편집 또는 소유자 권한을 할당할 수 있습니다.

감독자 클러스터로 인증

DevOps 엔지니어는 vSphere에 대한 Kubernetes CLI 도구를 사용하여 vCenter Single Sign-On 자격 증명과 Kubernetes 제어부 IP 주소로 감독자 클러스터에 인증합니다. 자세한 내용은 vCenter Single Sign-On 사용자로 감독자 클러스터에 연결의 내용을 참조하십시오.

DevOps 엔지니어가 감독자 클러스터에 로그인하면 인증 프록시가 요청을 vCenter Single Sign-On으로 리디렉션합니다. vSphere kubectl 플러그인은 vCenter Server와 세션을 설정하고 vCenter Single Sign-On에서 인증 토큰을 가져옵니다. 또한 DevOps 엔지니어가 액세스할 수 있는 네임스페이스 목록을 가져오고 이러한 네임스페이스로 구성을 채웁니다. 사용자 계정의 사용 권한이 변경되면, 다음 로그인 시 네임스페이스 목록이 업데이트됩니다.

감독자 클러스터에 로그인하는 데 사용하는 계정은 자신에게 할당된 네임스페이스에 대한 액세스만 제공합니다. 이 계정으로는 vCenter Server에 로그인할 수 없습니다. vCenter Server에 로그인하려면 명시적 권한이 필요합니다.

참고 kubectl에 대한 세션은 10시간 동안 지속됩니다. 세션이 만료된 후에는 감독자 클러스터를 다시 인증해야 합니다. 토큰은 로그아웃 시 사용자 계정의 구성 파일에서 삭제되지만 세션이 종료될 때까지 유효합니다.

Tanzu Kubernetes 클러스터를 사용하여 인증

DevOps 엔지니어, 개발자 및 관리자를 포함한 Tanzu Kubernetes 클러스터 사용자는 다양한 방식으로 클러스터를 인증할 수 있습니다. 자세한 내용은 Tanzu Kubernetes 클러스터에서 인증 수행의 내용을 참조하십시오.

참고 Tanzu Kubernetes 클러스터에는 사용자 및 시스템 계정에 포드 및 리소스를 클러스터에 배포하기 위한 포드 보안 정책이 있어야 합니다. 자세한 내용은 Tanzu Kubernetes 클러스터에서 포드 보안 정책 사용의 내용을 참조하십시오.

vSphere with Tanzu 네트워킹

감독자 클러스터는 vSphere 네트워킹 스택 또는 VMware NSX-T™ Data Center를 사용하여 Kubernetes 제어부 VM, 서비스 및 워크로드에 대한 연결을 제공할 수 있습니다. Tanzu Kubernetes Grid 서비스에서 프로비저닝된 Tanzu Kubernetes 클러스터에 사용되는 네트워킹은 클러스터 포드, 서비스 및 수신을 위한 네트워킹을 제공하는 오픈 소스 소프트웨어 및 vSphere with Tanzu 인프라의 기반이 되는 패브릭의 조합입니다.

자세한 내용은 장 4 vSphere with Tanzu에 대한 네트워킹 항목을 참조하십시오.

vSphere with Tanzu 보안

vSphere with Tanzu는 vSphere 보안 기능을 활용하며 기본적으로 안전한 클러스터 Tanzu Kubernetes 클러스터를 프로비저닝합니다.

vSphere with Tanzu는 vCenter Server 및 ESXi에 내장된 보안 기능을 활용할 수 있는 vSphere에 대한 추가 기능 모듈입니다. 자세한 내용은 vSphere 보안 설명서를 참조하십시오.

감독자 클러스터는 데이터베이스(etcd)에 저장된 모든 암호를 암호화합니다. 암호는 부팅 시 vCenter Server에 의해 제공되는 로컬 암호 해독 키 파일을 통해 암호화됩니다. 암호 해독 키는 감독자 클러스터 노드의 메모리(tempfs)와 vCenter Server 데이터베이스 내의 암호화된 형식으로 디스크에 저장됩니다. 키는 각 시스템의 루트 사용자에게 일반 텍스트로 제공됩니다. 각 워크로드 클러스터의 데이터베이스 내에 보관된 암호는 일반 텍스트로 저장됩니다. 모든 etcd 연결은 설치 시 생성되고 업그레이드 중에 순환되는 인증서를 사용하여 인증됩니다. 현재는 인증서를 수동으로 순환하거나 업데이트할 수 없습니다.

vSphere 7.0 업데이트 2부터 AMD 시스템의 감독자 클러스터에서 기밀 vSphere 포드를 실행할 수 있습니다. SEV-ES(Secure Encrypted Virtualization-Encrypted State)를 보안 강화 항목으로 추가하여 기밀 vSphere 포드를 생성할 수 있습니다. 자세한 내용은 기밀 vSphere 포드 배포의 내용을 참조하십시오.

Tanzu Kubernetes 클러스터는 기본적으로 안전합니다. Tanzu Kubernetes Grid 서비스에서 프로비저닝된 모든 Tanzu Kubernetes 클러스터에 대해 제한적인 PSP(PodSecurityPolicy)를 사용할 수 있습니다. 개발자가 권한 있는 포드 또는 루트 컨테이너를 실행해야 하는 경우, 최소한 클러스터 관리자는 권한이 있는 기본 PSP에 대한 액세스 권한을 사용자에게 부여하는 RoleBinding을 생성해야 합니다. 자세한 내용은 Tanzu Kubernetes 클러스터에서 포드 보안 정책 사용의 내용을 참조하십시오.

Tanzu Kubernetes 클러스터에는 인프라 자격 증명이 없습니다. Tanzu Kubernetes 클러스터 내에 저장된 자격 증명은 Tanzu Kubernetes 클러스터에 테넌시가 있는 vSphere 네임스페이스에만 액세스하기에 충분합니다. 따라서 클러스터 운영자 또는 사용자에 대해 권한을 에스컬레이션할 수 있는 방법은 없습니다.

Tanzu Kubernetes 클러스터에 액세스하는 데 사용되는 인증 토큰은 감독자 클러스터에 액세스하는 데 토큰을 사용할 수 없도록 범위가 지정됩니다. 이렇게 하면 클러스터 운영자 또는 클러스터를 손상시키려는 개인이 Tanzu Kubernetes 클러스터에 로그인할 때 vSphere 관리자의 토큰을 캡처하기 위해 루트 수준 액세스를 사용하는 것을 방지할 수 있습니다.

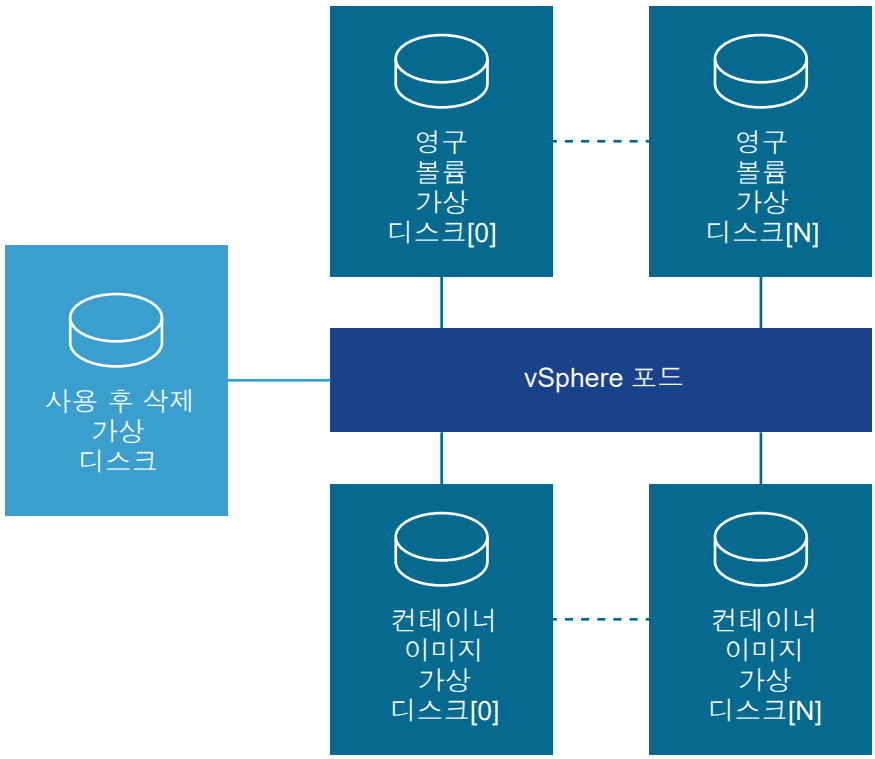
vSphere with Tanzu 스토리지

vSphere with Tanzu는 스토리지 정책을 사용하여 환경에서 사용 가능한 공유 데이터스토어(VMFS, NFS, vSAN 또는 vVols 데이터스토어 등)와 통합합니다. 정책은 데이터스토어를 나타내며 제어부 VM, 포드 사용 후 삭제 디스크, 컨테이너 이미지, 영구 스토리지 볼륨과 같은 개체의 스토리지 배치를 관리합니다. Tanzu Kubernetes 클러스터를 사용하는 경우 스토리지 정책은 Tanzu Kubernetes 클러스터 노드가 배포되는 방식도 지정합니다.

vSphere with Tanzu을 사용하도록 설정하기 전에 감독자 클러스터 및 네임스페이스에서 사용할 스토리지 정책을 생성합니다.

vSphere 스토리지 환경 및 DevOps의 요구 사항에 따라 서로 다른 스토리지 클래스를 나타내는 여러 스토리지 정책을 생성할 수 있습니다.

예를 들어 vSphere 포드가 세 가지 유형의 가상 디스크를 모두 마운트하고 vSphere 스토리지 환경에 Bronze, Silver 및 Gold의 3가지 데이터스토어 클래스가 있는 경우 모든 데이터스토어에 대한 스토리지 정책을 생성할 수 있습니다. 그런 다음 사용 후 삭제 및 컨테이너 이미지 가상 디스크에 대해 Bronze 데이터스토어를 사용하고 영구 볼륨 가상 디스크에 대해 Silver 및 Gold 데이터스토어를 사용할 수 있습니다.



스토리지 정책에 대한 일반적인 정보는 "vSphere 스토리지" 설명서의 스토리지 정책 기반 관리 장을 참조하십시오. 스토리지 정책 생성에 대한 자세한 내용은 vSphere with Tanzu에 대한 스토리지 정책 생성 항목을 참조하십시오.

사용 후 삭제 가상 디스크

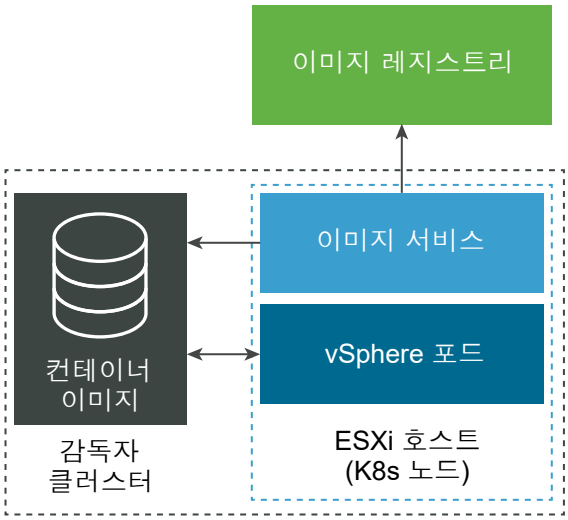
vSphere 포드 및 Tanzu Kubernetes 클러스터에서 실행되는 포드는 작업 중에, `emptyDir` 볼륨 및 `ConfigMaps`와 같은 Kubernetes 개체를 로그로 저장하기 위해 사용 후 삭제되는 스토리지가 필요합니다. 이 사용 후 삭제 또는 임시 스토리지는 포드가 계속 존재하는 한 지속됩니다. 사용 후 삭제 데이터는 컨테이너를 다시 시작해도 유지되지만 사용 후 삭제 가상 디스크는 포드의 수명이 다하면 사라집니다.

각 포드에는 사용 후 삭제 가상 디스크가 하나씩 있습니다. vSphere 관리자는 감독자 클러스터에 대한 스토리지를 구성할 때 스토리지 정책을 사용하여 모든 사용 후 삭제 가상 디스크에 대한 데이터스토어 위치를 정의합니다.

컨테이너 이미지 가상 디스크

포드 내의 컨테이너는 실행할 소프트웨어가 포함된 이미지를 사용합니다. 포드는 컨테이너에서 사용하는 이미지를 이미지 가상 디스크로 마운트합니다. 포드의 수명 주기가 완료되면 이미지 가상 디스크가 포드에서 분리됩니다.

ESXi 구성 요소인 이미지 서비스는 이미지 레지스트리에서 컨테이너 이미지를 끌어와서 가상 디스크로 변환하여 포드 내에서 실행하는 작업을 담당합니다.



ESXi는 포드에서 실행 중인 컨테이너에 대해 다운로드된 이미지를 캐시할 수 있습니다. 동일한 이미지를 사용하는 후속 포드는 외부 컨테이너 레지스트리가 아닌 로컬 캐시에서 이미지를 끌어옵니다.

사용 후 삭제 디스크와 마찬가지로 vSphere 관리자는 감독자 클러스터 수준에서 이미지 캐시의 데이터스토어 위치를 지정합니다. 장 5 감독자 클러스터 구성 및 관리 및 감독자 클러스터의 스토리지 설정 변경의 내용을 참조하십시오.

컨테이너 이미지 작업에 대한 자세한 내용은 장 15 vSphere with Tanzu 워크로드에 컨테이너 레지스트리 사용 항목을 참조하십시오.

영구 스토리지 가상 디스크

특정 Kubernetes 워크로드에는 데이터를 영구적으로 저장하기 위한 영구 스토리지가 필요합니다. Kubernetes 워크로드에 대한 영구 스토리지를 프로비저닝하기 위해 vSphere with Tanzu은 영구 볼륨을 관리하는 vCenter Server 구성 요소인 CNS(클라우드 네이티브 스토리지)와 통합됩니다.

영구 스토리지는 vSphere 포드, Tanzu Kubernetes 클러스터 및 VM에서 사용할 수 있습니다. DevOps 팀이 영구 스토리지를 사용할 수 있도록 vSphere 관리자는 다양한 스토리지 요구 사항 및 서비스 클래스를 설명하는 VM 스토리지 정책을 생성합니다. 그런 다음 스토리지 정책을 vSphere 네임스페이스에 할당할 수 있습니다. vSphere 네임스페이스 생성 및 구성 및 네임스페이스의 스토리지 설정 변경의 내용을 참조하십시오.

감독자 클러스터 및 Tanzu Kubernetes 클러스터에서 영구 스토리지를 사용하는 방법에 대한 자세한 내용 및 세부 사항은 장 10 vSphere with Tanzu에서 영구 스토리지 사용 및 장 13 TKGS 클러스터 프로비저닝 및 운영 항목을 참조하십시오.

vSphere with Tanzu에 대한 네트워킹

4

감독자 클러스터는 vSphere 네트워킹 스택 또는 VMware NSX-T™ Data Center를 사용하여 Kubernetes 제어부 VM, 서비스 및 워크로드에 대한 연결을 제공할 수 있습니다. Tanzu Kubernetes Grid 서비스에서 프로비저닝된 Tanzu Kubernetes 클러스터에 사용되는 네트워킹은 클러스터 포드, 서비스 및 수신을 위한 네트워킹을 제공하는 오픈 소스 소프트웨어 및 vSphere with Tanzu 인프라의 기반이 되는 패브릭의 조합입니다.

본 장은 다음 항목을 포함합니다.

감독자 클러스터 네트워킹

vSphere with Tanzu 환경에서 감독자 클러스터는 vSphere 네트워킹 스택 또는 VMware NSX-T Data Center™를 사용하여 Kubernetes 제어부 VM, 서비스 및 워크로드에 대한 연결을 제공할 수 있습니다. 감독자 클러스터가 vSphere 네트워킹 스택으로 구성되면 클러스터의 모든 호스트가 Kubernetes 워크로드 및 제어부 VM에 대한 연결을 제공하는 vSphere Distributed Switch에 연결됩니다. vSphere 네트워킹 스택을 사용하는 감독자 클러스터에는 DevOps 사용자 및 외부 서비스에 대한 연결을 제공하기 위해 vCenter Server 관리 네트워크에 로드 밸런서가 필요합니다. VMware NSX-T Data Center™로 구성된 감독자 클러스터는 솔루션의 소프트웨어 기반 네트워크 및 NSX Edge 로드 밸런서를 사용하여 외부 서비스 및 DevOps 사용자에 대한 연결을 제공합니다.

NSX-T Data Center를 사용한 감독자 클러스터 네트워킹

VMware NSX-T Data Center™는 외부 네트워크 및 감독자 클러스터 내부의 개체에 대한 네트워크 연결을 제공합니다. 클러스터를 구성하는 ESXi 호스트에 대한 연결은 표준 vSphere 네트워크를 통해 처리됩니다.

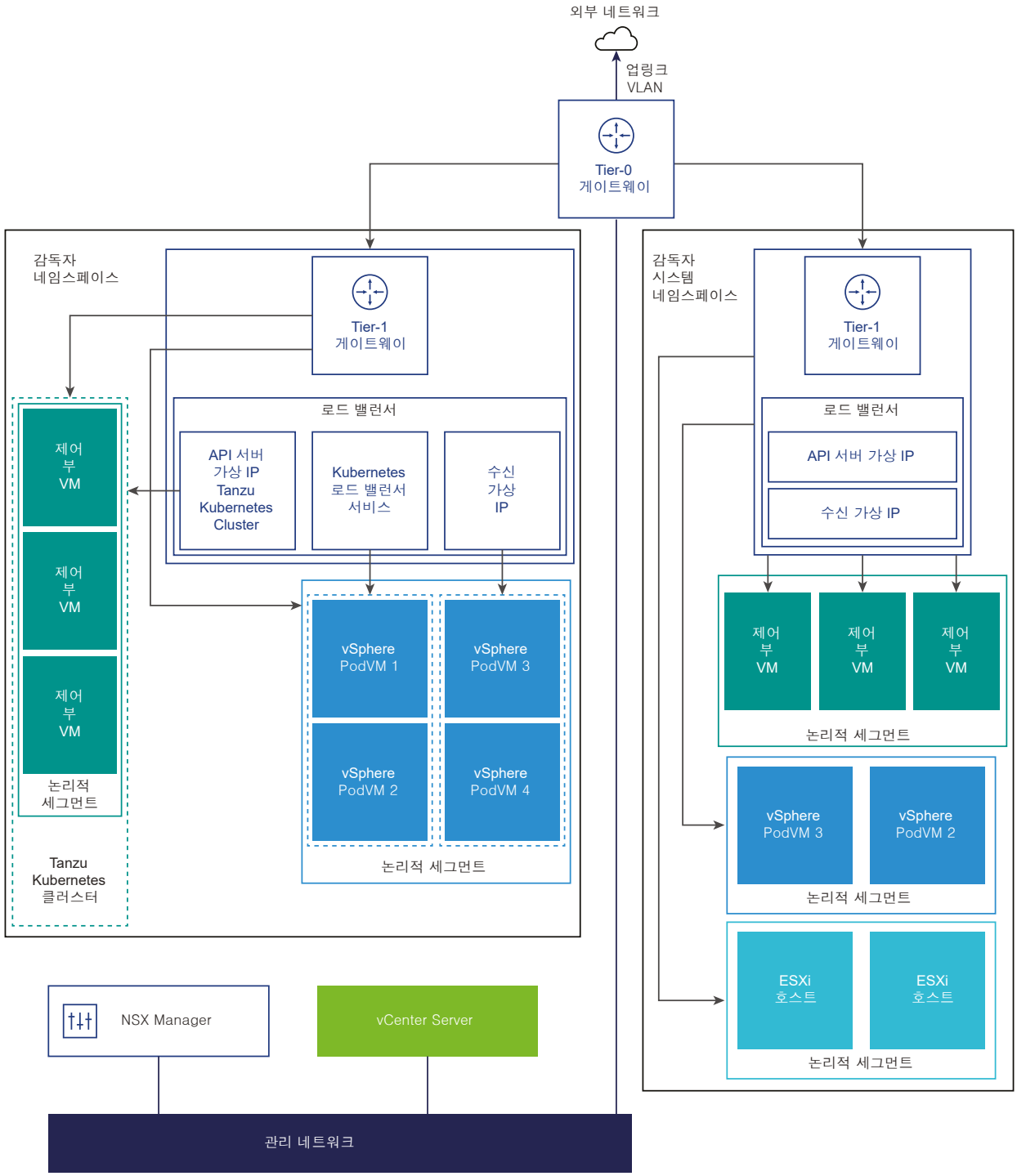
기존 NSX-T Data Center 배포를 사용하거나 NSX-T Data Center의 새 인스턴스를 배포하여 감독자 클러스터 네트워킹을 수동으로 구성할 수도 있습니다.

다음 표에는 지원되는 NSX-T Data Center 버전이 나열되어 있습니다.

vSphere with Tanzu	NSX-T Data Center
버전 7.0 업데이트 3	버전 3.0, 3.0.x, 3.1, 3.1.1, 3.1.2 및 3.1.3.
버전 7.0 업데이트 2	버전 3.0, 3.0.x, 3.1, 3.1.1, 3.1.2.
버전 7.0 업데이트 1c	버전 3.0, 3.0.x, 3.1, 3.1.1.
버전 7.0 업데이트 1	버전 3.0, 3.0.1, 3.0.1.1, 3.0.2.
Version 7.0	Version 3.0.

이 섹션에서는 vSphere with Tanzu 버전 7.0 업데이트 2를 설치하고 구성할 때의 네트워킹 토폴로지를 설명합니다. vSphere with Tanzu 버전 7.0 업데이트 1에서 버전 7.0 업데이트 2로 업그레이드하는 경우 업그레이드에 대한 자세한 내용은 네트워크 토폴로지 업그레이드 항목을 참조하십시오.

그림 4-1. 감독자 클러스터 네트워킹



- NCP(NSX Container Plug-in)는 NSX-T Data Center와 Kubernetes 간의 통합을 제공합니다. NCP의 주요 구성 요소는 컨테이너에서 실행되며 NSX Manager 및 Kubernetes 제어부와 통신합니다. NCP는 컨테이너 및 기타 리소스에 대한 변경 사항을 모니터링하고 NSX API를 호출하여 컨테이너에 대한 논리적 포트, 세그먼트, 라우터 및 보안 그룹과 같은 네트워킹 리소스를 관리합니다.

NCP는 기본적으로 시스템 네임스페이스에 대해 하나의 공유 Tier-1 게이트웨이를 생성하고 각 네임스페이스에 대해 Tier-1 게이트웨이와 로드 밸런서를 생성합니다. Tier-1 게이트웨이는 Tier-0 게이트웨이 및 기본 세그먼트에 연결됩니다.

시스템 네임스페이스는 감독자 클러스터 및 Tanzu Kubernetes가 작동하는 데 필수적인 핵심 구성 요소에서 사용되는 네임스페이스입니다. Tier-1 게이트웨이, 로드 밸런서 및 SNAT IP를 포함하는 공유 네트워크 리소스는 시스템 네임스페이스에 그룹화됩니다.

- NSX Edge는 외부 네트워크에서 감독자 클러스터 개체로 연결을 제공합니다. NSX Edge 클러스터에는 제어부 VM에 상주하는 Kubernetes API 서버와 감독자 클러스터 외부에서 게시하고 액세스할 수 있는 모든 애플리케이션에 이중화를 제공하는 로드 밸런서가 있습니다.
- Tier-0 게이트웨이가 NSX Edge 클러스터와 연결되어 외부 네트워크에 대한 라우팅을 제공합니다. 업링크 인터페이스는 동적 라우팅 프로토콜 , BGP 또는 정적 라우팅 중 하나를 사용합니다.
- 각 vSphere 네임스페이스에는 별도의 네트워크 및 네임스페이스 내의 애플리케이션이 공유하는 네트워킹 리소스 집합(예: Tier-1 게이트웨이, 로드 밸런서 서비스, SNAT IP 주소)이 있습니다.
- 동일한 네임스페이스에 있는 vSphere 포드, 일반 VM 또는 Tanzu Kubernetes 클러스터에서 실행되는 워크로드는 North-South 연결에 대해 동일한 SNAT IP를 공유합니다.
- vSphere 포드 또는 Tanzu Kubernetes 클러스터에서 실행되는 워크로드에는 기본 방화벽에 의해 구현되는 것과 동일한 격리 규칙이 있습니다.
- 각 Kubernetes 네임스페이스에 대해 별도의 SNAT IP가 필요하지 않습니다. 네임스페이스 간의 East-West 연결은 SNAT가 아닙니다.
- 각 네임스페이스의 세그먼트는 NSX Edge 클러스터에 연결된, 표준 모드에서 작동하는 VDS(vSphere Distributed Switch)에 상주합니다. 세그먼트는 감독자 클러스터에 오버레이 네트워크를 제공합니다.
- 감독자 클러스터는 공유 Tier-1 게이트웨이 내에 별도의 세그먼트가 있습니다. 각 Tanzu Kubernetes 클러스터에 대해 세그먼트는 네임스페이스의 Tier-1 게이트웨이 내에 정의됩니다.
- 각 ESXi 호스트의 Spherelet 프로세스는 관리 네트워크의 인터페이스를 통해 vCenter Server와 통신합니다.

감독자 클러스터 네트워킹에 대해 자세히 알아보려면 vSphere 7 with Kubernetes 네트워크 서비스 - 1부 - 감독자 클러스터 비디오를 시청하십시오.

NSX-T Data Center를 사용한 네트워킹 구성 방법

감독자 클러스터는 고유한 네트워킹 구성을 사용합니다. 동일한 네트워킹 모델을 배포하는 NSX-T Data Center를 사용한 감독자 클러스터 네트워킹을 구성하기 위한 두 가지 방법이 있습니다.

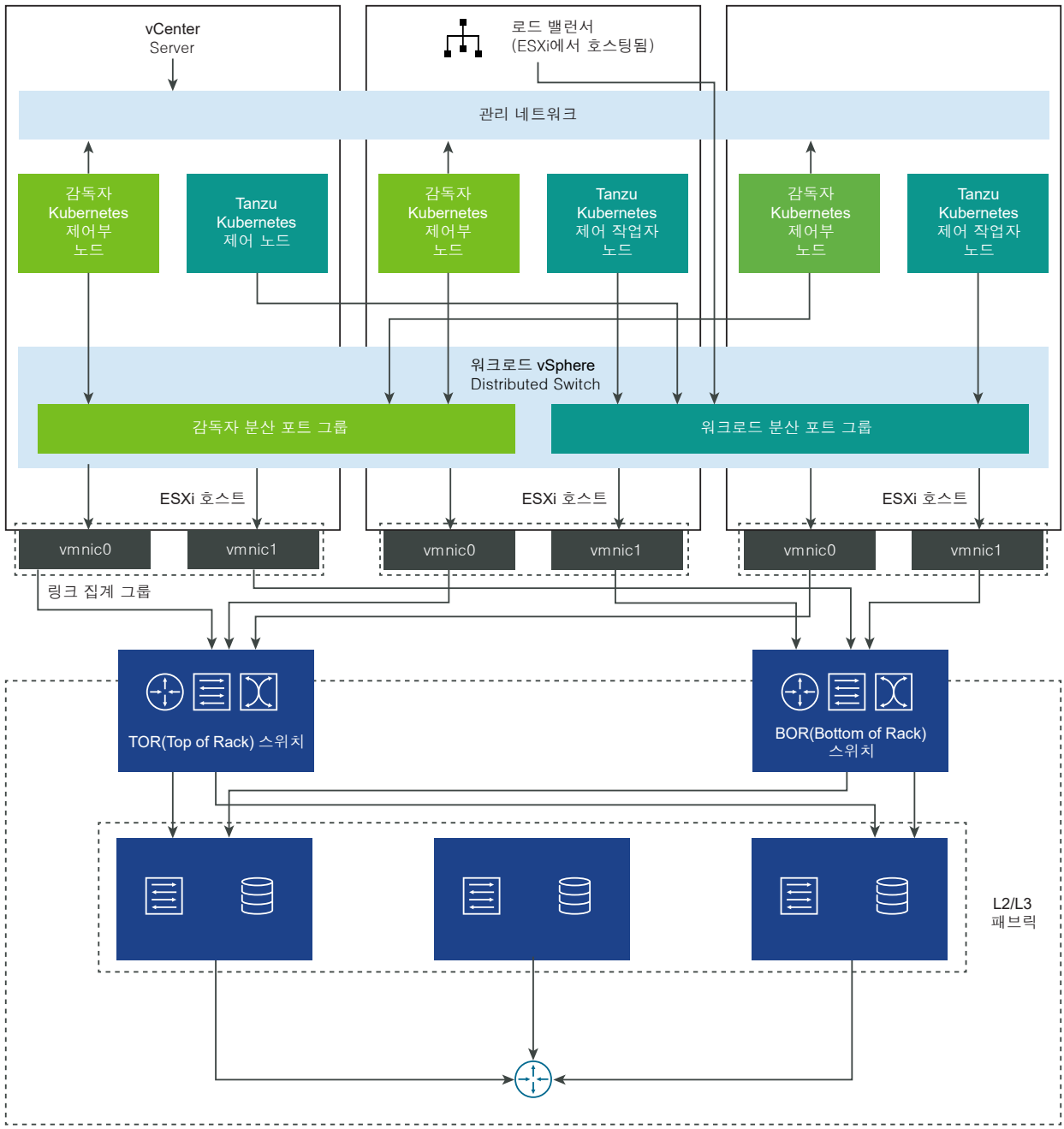
- 감독자 클러스터 네트워킹을 구성하는 가장 간단한 방법은 VMware Cloud Foundation SDDC Manager를 사용하는 것입니다. 자세한 내용은 VMware Cloud Foundation SDDC Manager 설명서를 참조하십시오. 자세한 내용은 워크로드 관리 작업을 참조하십시오.

- 기존 NSX-T Data Center 배포를 사용하거나 NSX-T Data Center의 새 인스턴스를 배포하여 감독자 클러스터 네트워킹을 수동으로 구성할 수도 있습니다. 자세한 내용은 vSphere with Tanzu에 대한 NSX-T Data Center 설치 및 구성의 내용을 참조하십시오.

vSphere Distributed Switch를 사용한 감독자 클러스터 네트워킹

vSphere Distributed Switch에서 지원되는 감독자 클러스터는 분산 포트 그룹을 네임스페이스에 대한 워크로드 네트워크로 사용합니다.

그림 4-2. vSphere Distributed Switch를 사용한 네임스페이스 네트워킹



감독자 클러스터에 대해 구현하는 토폴로지에 따라 하나 이상의 분산 포트 그룹을 워크로드 네트워크로 사용할 수 있습니다. Kubernetes 제어부 VM에 대한 연결을 제공하는 네트워크를 기본 워크로드 네트워크라고 합니다. 이 네트워크를 감독자 클러스터의 모든 네임스페이스에 할당하거나 각 네임스페이스에 대해 서로 다른 네트워크를 사용할 수 있습니다. Tanzu Kubernetes 클러스터는 클러스터가 상주하는 네임스페이스에 할당된 워크로드 네트워크에 연결됩니다.

vSphere Distributed Switch에서 지원되는 감독자 클러스터는 DevOps 사용자 및 외부 서비스에 대한 연결을 제공하기 위해 로드 밸런서를 사용합니다. NSX Advanced Load Balancer 또는 HAProxy 로드 밸런서를 사용할 수 있습니다.

자세한 내용은 vSphere with Tanzu에 대한 vSphere 네트워킹 및 NSX Advanced Load Balancer 구성 및 HAProxy 로드 밸런서 설치 및 구성 항목을 참조하십시오.

Tanzu Kubernetes Grid 서비스 클러스터 네트워킹

Tanzu Kubernetes Grid 서비스에서 프로비저닝된 Tanzu Kubernetes Grid 서비스 클러스터는 Antrea(기본값) 및 Calico라는 두 가지 CNI 옵션을 지원합니다. 둘 다 클러스터 포드, 서비스 및 수신을 위한 네트워킹을 제공하는 오픈 소스 소프트웨어입니다.

Tanzu Kubernetes Grid 서비스에서 프로비저닝된 Tanzu Kubernetes Grid 서비스 클러스터는 다음 CNI(Container Network Interface) 옵션을 지원합니다.

- Antrea
- Calico

Antrea는 새 Tanzu Kubernetes Grid 서비스 클러스터에 대한 기본 CNI입니다. Antrea를 사용하는 경우 클러스터 프로비저닝 중에 CNI로 지정할 필요가 없습니다. Calico를 CNI로 사용하기 위해 다음 두 가지 옵션을 사용할 수 있습니다.

- 클러스터 YAML에서 직접 CNI를 지정합니다. Tanzu Kubernetes Grid 서비스 v1alpha1 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하는 예의 내용을 참조하십시오.
- 기본 CNI를 변경합니다. Tanzu Kubernetes Grid 서비스 v1alpha1 API 구성 예시의 내용을 참조하십시오.

참고 기본 CNI로 Antrea를 사용하려면 Tanzu Kubernetes Grid 서비스 클러스터에 대한 최소 버전의 OVA 파일이 필요합니다. 업데이트를 위한 Tanzu Kubernetes 클러스터 호환성 확인의 내용을 참조하십시오.

이 표에는 Tanzu Kubernetes Grid 서비스 클러스터 네트워킹 기능과 구현이 요약되어 있습니다.

표 4-1. Tanzu Kubernetes Grid 서비스 클러스터 네트워킹

끝점	제공자	설명
포드 연결	Antrea 또는 Calico	포드용 컨테이너 네트워크 인터페이스입니다. Antrea는 Open vSwitch를 사용합니다. Calico는 BGP와 함께 Linux 브리지를 사용합니다.
서비스 유형: ClusterIP	Antrea 또는 Calico	클러스터 내에서만 액세스할 수 있는 기본 Kubernetes 서비스 유형입니다.
서비스 유형: NodePort	Antrea 또는 Calico	Kubernetes 네트워크 프록시에 의해 각 작업자 노드에 열린 포트를 통한 외부 액세스를 허용합니다.
서비스 유형: LoadBalancer	NSX-T 로드 밸런서, NSX Advanced Load Balancer, HAProxy	NSX-T의 경우, 서비스 유형 정의당 하나의 가상 서버입니다. NSX Advanced Load Balancer는 이 설명서에서 해당 섹션을 참조하십시오. **참고** 일부 로드 밸런싱 기능은 HAProxy에서 사용할 수 없습니다(예: 정적 IP 지원).
클러스터 수신	타사 수신 컨트롤러	인바운드 포드 트래픽을 위한 라우팅, Contour와 같은 타사 수신 컨트롤러를 사용할 수 있습니다.
네트워크 정책	Antrea 또는 Calico	선택한 포드 및 네트워크 끝점 사이에서 허용되는 트래픽을 제어합니다. Antrea는 Open vSwitch를 사용합니다. Calico는 Linux IP 테이블을 사용합니다.

vSphere with Tanzu에 대한 NSX-T Data Center 구성

vSphere with Tanzu를 사용하려면 감독자 클러스터, vSphere 네임스페이스 및 네임스페이스 내에서 실행되는 모든 개체(예: vSphere 포드, VM 및 Tanzu Kubernetes 클러스터)에 대한 연결을 사용하도록 설정하기 위한 특정 네트워킹 구성이 필요합니다. vSphere 관리자는 vSphere with Tanzu에 대한 NSX-T Data Center를 설치하고 구성합니다.

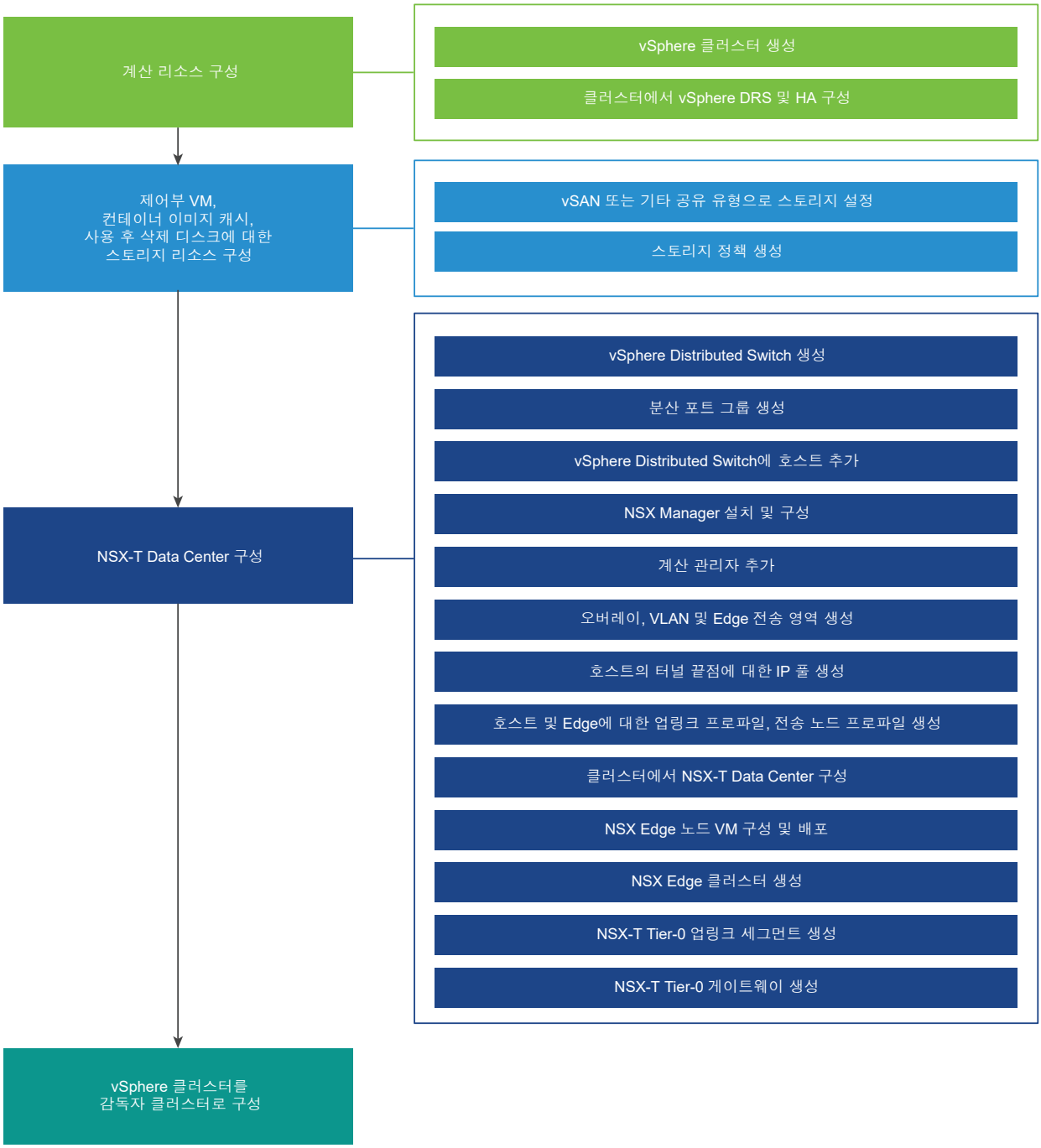
감독자 클러스터는 고유한 네트워킹 구성을 사용합니다. 동일한 네트워킹 모델을 배포하는 감독자 클러스터 네트워킹을 구성하기 위한 두 가지 방법이 있습니다.

- 감독자 클러스터 네트워킹을 구성하는 가장 간단한 방법은 VMware Cloud Foundation SDDC Manager를 사용하는 것입니다. 자세한 내용은 VMware Cloud Foundation SDDC Manager 설명서를 참조하십시오. 자세한 내용은 워크로드 관리 작업을 참조하십시오.
- 기존 NSX-T Data Center 배포를 사용하거나 NSX-T Data Center의 새 인스턴스를 배포하여 감독자 클러스터 네트워킹을 수동으로 구성할 수도 있습니다.

NSX-T Data Center가 포함된 감독자 클러스터 워크플로

vSphere 관리자는 vSphere 네트워킹 스택이 포함된 감독자 클러스터로 vSphere 클러스터를 구성할 수 있습니다.

그림 4-3. NSX-T Data Center 네트워킹이 포함된 감독자 클러스터 워크플로



절차

1 NSX-T Data Center를 사용하여 vSphere with Tanzu를 설정하기 위한 시스템 요구 사항

NSX-T Data Center 네트워킹 스택을 사용하여 vSphere 클러스터에서 vSphere with Tanzu를 구성하기 위한 시스템 요구 사항을 검토합니다.

2 NSX-T Data Center를 사용하는 감독자 클러스터용 토폴로지

Kubernetes 워크로드의 요구 사항 및 기본 네트워킹 인프라에 따라 클러스터에 다른 토폴로지를 적용할 수 있습니다.

3 NSX-T Data Center를 사용하여 감독자 클러스터를 구성하기 위한 모범 사례 고려 사항

NSX-T Data Center를 사용하여 vSphere 클러스터를 감독자 클러스터로 구성하는 경우 다음과 같은 모범 사례를 고려하십시오.

4 vSphere with Tanzu에 대한 NSX-T Data Center 설치 및 구성

vSphere with Tanzu를 사용하려면 감독자 클러스터, vSphere 네임스페이스 및 네임스페이스 내에서 실행되는 모든 개체(예: vSphere 포드, VM 및 Tanzu Kubernetes 클러스터)에 대한 연결을 사용하도록 설정하기 위한 특정 네트워킹 구성이 필요합니다. vSphere 관리자는 vSphere with Tanzu에 대한 NSX-T Data Center를 설치하고 구성합니다.

NSX-T Data Center를 사용하여 vSphere with Tanzu를 설정하기 위한 시스템 요구 사항

NSX-T Data Center 네트워킹 스택을 사용하여 vSphere 클러스터에서 vSphere with Tanzu를 구성하기 위한 시스템 요구 사항을 검토합니다.

vSphere with Tanzu 클러스터에 대한 구성 제한

VMware는 VMware 구성 최대값 도구에 구성 제한을 제공합니다.

감독자 클러스터 및 Tanzu Kubernetes 클러스터를 포함하여 vSphere with Tanzu와 관련된 구성 제한을 보려면 **vSphere > vSphere 7.0 > vSphere with Kubernetes > VMware Tanzu Kubernetes Grid Service for vSphere**를 선택하고 **제한 보기**를 클릭하거나 이 링크를 따라가십시오.

관리, Edge 및 워크로드 도메인 클러스터에 대한 요구 사항

단일 vSphere 클러스터에서 관리, Edge 및 워크로드 관리 기능이 결합된 vSphere with Tanzu을 배포할 수 있습니다.

표 4-2. 관리, Edge 및 워크로드 관리 클러스터에 대한 최소 계산 요구 사항

시스템	최소 배포 크기	CPU	메모리	스토리지
vCenter Server 7.0	작음	2	16GB	290GB
ESXi 호스트 7.0	호스트당 고정 IP가 1개인 ESXi 호스트 3개. vSAN을 사용하는 경우: 호스트당 물리적 NIC가 2개 이상인 ESXi 호스트 3개가 최소입니다. 하지만 패치 적용 및 업그레이드 중 복원력을 위해 ESXi 호스트 4개가 권장됩니다. 호스트는 vSphere DRS 및 HA를 사용하도록 설정된 클러스터에 가입되어 있어야 합니다. vSphere DRS는 완전히 자동화되거나 부분적으로 자동화된 모드여야 합니다. **경고** 감독자 클러스터를 구성한 후에는 vSphere DRS를 사용하지 않도록 설정하지 마십시오. 감독자 클러스터에서 워크로드를 실행하려면 항상 DRS를 사용하도록 설정해야 합니다. DRS를 사용하지 않도록 설정하면 Tanzu Kubernetes 클러스터가 중단됩니다.	8	호스트당 64GB	해당 없음
NSX Manager	중간	6	24GB	300GB
NSX Edge 1	큼	8	32GB	200GB
NSX Edge 2	큼	8	32GB	200GB
Kubernetes 제어부 VM	3	4	16GB	16GB

별도의 관리 및 Edge 클러스터 및 워크로드 관리 클러스터가 포함된 토폴로지

두 개의 클러스터, 즉 관리 및 Edge 기능을 위한 클러스터와 워크로드 관리 전용 클러스터에 vSphere with Tanzu을 배포할 수 있습니다.

표 4-3. 관리 및 Edge 클러스터에 대한 최소 계산 요구 사항

시스템	최소 배포 크기	CPU	메모리	스토리지
vCenter Server 7.0	작음	2	16GB	290GB
ESXi 호스트 7.0	ESXi 호스트 2개	8	호스트당 64GB	해당 없음
NSX Manager	중간	6	24GB	300GB
NSX Edge 1	큼	8	32GB	200GB
NSX Edge 2	큼	8	32GB	200GB

표 4-4. 워크로드 관리 클러스터에 대한 최소 계산 요구 사항

시스템	최소 배포 크기	CPU	메모리	스토리지
ESXi 호스트 7.0	호스트당 고정 IP가 1개인 ESXi 호스트 3개. vSAN을 사용하는 경우: 호스트당 물리적 NIC가 2개 이상인 ESXi 호스트 3개가 최소입니다. 단, 패치 적용 및 업그레이드 중에는 복원력을 위해 ESXi 호스트 4개가 권장됩니다. 호스트는 vSphere DRS 및 HA를 사용하도록 설정된 클러스터에 가입되어 있어야 합니다. vSphere DRS는 완전히 자동화된 모드여야 합니다. **경고** 감독자 클러스터를 구성한 후에는 vSphere DRS를 사용하지 않도록 설정하지 마십시오. 감독자 클러스터에서 워크로드를 실행하려면 항상 DRS를 사용하도록 설정해야 합니다. DRS를 사용하지 않도록 설정하면 Tanzu Kubernetes 클러스터가 중단됩니다.	8	호스트당 64GB	해당 없음
Kubernetes 제어부 VM	3	4	16GB	16GB

네트워킹 요구 사항

vSphere에서 Kubernetes 워크로드 관리를 위해 구현한 토폴로지에 관계없이, 배포는 아래 표에 나열된 네트워킹 요구 사항을 충족해야 합니다.

참고 vSphere 7 감독자 클러스터를 사용하여 IPv6 클러스터를 생성하거나 IPv6 클러스터를 Tanzu Mission Control에 등록할 수 없습니다.

구성 요소	최소 수량	필수 구성
Kubernetes 제어부 VM에 대한 고정 IP	5개의 블록	감독자 클러스터의 Kubernetes 제어부 VM에 할당할 5개의 연속적 고정 IP 주소 블록.
관리 트래픽 네트워크	1	ESXi 호스트, vCenter Server 및 DHCP 서버로 라우팅할 수 있는 관리 네트워크. 컨테이너 레지스트리가 외부 네트워크에 있는 경우 네트워크는 컨테이너 레지스트리에 액세스할 수 있어야 하고 인터넷에 연결되어 있어야 합니다. 컨테이너 레지스트리는 DNS를 통해 확인할 수 있어야 하며 아래에 설명된 송신 설정이 도달할 수 있어야 합니다.
NTP 및 DNS 서버	1	vCenter Server에서 사용할 수 있는 DNS 서버 및 NTP 서버. **참고** 모든 ESXi 호스트, vCenter Server 시스템 및 NSX Manager 인스턴스에 NTP를 구성합니다.
DHCP 서버	1	선택 사항입니다. 관리를 위한 IP 주소를 자동으로 획득하도록 DHCP 서버를 구성합니다. DHCP 서버는 클라이언트 식별자를 지원하고 호환되는 DNS 서버, DNS 검색 도메인 및 NTP 서버를 제공해야 합니다.
이미지 레지스트리	1	서비스를 위해 레지스트리에 액세스.
관리 네트워크 서브넷	1	ESXi 호스트와 vCenter Server, NSX 장치 및 Kubernetes 제어부 사이의 관리 트래픽에 사용되는 서브넷. 서브넷의 크기는 다음과 같아야 합니다. ■ 호스트 VMkernel 어댑터당 IP 주소 1개. ■ vCenter Server Appliance에 대해 IP 주소 1개. ■ NSX Manager에 대해 IP 주소 1개 또는 4개. 3개 노드 및 1개 VIP(가상 IP)의 NSX Manager 클러스터링 수행 시 4개. ■ Kubernetes 제어부에 대해 IP 주소 5개. 3개 노드 각각에 대해 1개, 가상 IP에 1개, 롤링 클러스터 업그레이드에 1개. **참고** 관리 네트워크와 워크로드 네트워크는 서로 다른 서브넷에 있어야 합니다. 관리 및 워크로드 네트워크에 동일한 서브넷을 할당하는 것은 지원되지 않으며 시스템 오류 및 문제가 발생할 수 있습니다.
관리 네트워크 VLAN	1	관리 네트워크 서브넷의 VLAN ID.

구성 요소	최소 수량	필수 구성
VLAN	3	이러한 VLAN IP는 TEP(터널 끝점)의 IP 주소입니다. ESXi 호스트 TEP와 Edge TEP는 라우팅이 가능해야 합니다. 다음에 VLAN IP 주소가 필요합니다. ■ ESXi 호스트 VTEP ■ 고정 IP를 사용하는 Edge VTEP ■ 전송 노드에 대한 업링크 및 Tier 0 게이트웨이. **참고** ESXi 호스트 VTEP 및 Edge VTEP의 MTU 크기가 1600보다 커야 합니다. ESXi 호스트와 NSX-T Edge 노드는 터널 끝점으로 작동하며 각 호스트와 Edge 노드에 TEP(터널 끝점) IP가 할당됩니다. ESXi 호스트의 TEP IP가 Edge 노드에 TEP IP로 오버레이 터널을 생성하기 때문에 VLAN IP를 라우팅할 수 있어야 합니다. Tier-0 게이트웨이에 대한 North-South 연결을 제공하려면 추가 VLAN이 필요합니다. IP 풀은 여러 클러스터 간에 공유될 수 있습니다. 하지만 호스트 오버레이 IP 풀/VLAN은 Edge 오버레이 IP 풀/VLAN과 공유해서는 안 됩니다. **참고** 호스트 TEP와 Edge TEP가 서로 다른 물리적 NIC를 사용하는 경우 동일한 VLAN을 사용할 수 있습니다.
Tier-0 업링크 IP	/개인 IP 주소 24개	Tie-0 업링크에 사용되는 IP 서브넷. Tie-0 업링크의 IP 주소에 대한 요구 사항은 다음과 같습니다. ■ IP 1개, Edge 이중화를 사용하지 않는 경우. ■ IP 4개, BGP와 Edge 이중화를 사용하는 경우. Edge당 IP 주소 2개. ■ IP 3개, 정적 경로와 Edge 이중화를 사용하는 경우. Edge 관리 IP, 서브넷, 게이트웨이, 업링크 IP, 서브넷, 게이트웨이는 고유해야 합니다.
물리적 네트워크 MTU	1600	MTU 크기는 오버레이 트래픽을 전달하는 모든 네트워크에서 1600 이상이어야 합니다.
vSphere 포드 CIDR 범위	/개인 IP 주소 23개	vSphere 포드의 IP 주소를 제공하는 개인 CIDR 범위입니다. 이러한 주소는 Tanzu Kubernetes 클러스터 노드에도 사용됩니다. 각 클러스터에 대해 고유한 vSphere 포드 CIDR 범위를 지정해야 합니다. **참고** vSphere 포드 CIDR 범위와 Kubernetes 서비스 주소의 CIDR 범위는 겹치지 않아야 합니다.
Kubernetes 서비스 CIDR 범위	/개인 IP 주소 16개	Kubernetes 서비스에 IP 주소를 할당할 개인 CIDR 범위. 각 감독자 클러스터에 대해 고유한 Kubernetes 서비스 CIDR 범위를 지정해야 합니다.

구성 요소	최소 수량	필수 구성
송신 CIDR 범위	/고정 IP 주소 27개	Kubernetes 서비스의 송신 IP를 결정하는 개인 CIDR 주석. 감독자 클러스터의 각 네임스페이스에는 송신 IP 주소가 하나만 할당됩니다. 송신 IP는 외부 엔티티가 네임스페이스의 서비스와 통신하는 데 사용하는 주소입니다. 송신 IP 주소 수는 감독자 클러스터가 포함할 수 있는 송신 정책의 수를 제한합니다. 최소값은 /27 이상의 CIDR입니다. 예를 들어 `10.174.4.96/27`입니다. **참고** 송신 IP 주소 및 수신 IP 주소는 겹치지 않아야 합니다.
수신 CIDR	/고정 IP 주소 27개	수신 IP 주소에 사용되는 개인 CIDR 범위. 수신을 사용하면 외부 네트워크에서 감독자 클러스터로 들어오는 요청에 트래픽 정책을 적용할 수 있습니다. 수신 IP 주소 수는 클러스터가 포함할 수 있는 수신의 수를 제한합니다. 최소값은 /27 이상의 CIDR입니다. **참고** 송신 IP 주소 및 수신 IP 주소는 겹치지 않아야 합니다.

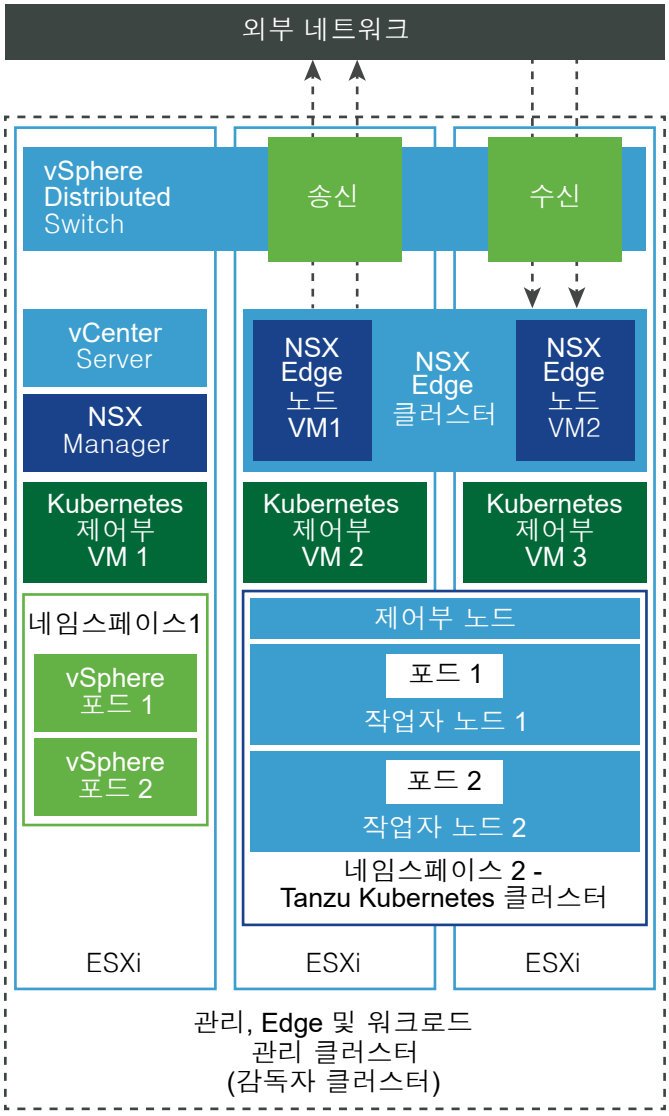
NSX-T Data Center를 사용하는 감독자 클러스터용 토폴로지

Kubernetes 워크로드의 요구 사항 및 기본 네트워킹 인프라에 따라 클러스터에 다른 토폴로지를 적용할 수 있습니다.

관리, Edge 및 워크로드 도메인 클러스터에 대한 토폴로지

단일 vSphere 클러스터에서 관리, Edge 및 워크로드 관리 기능이 결합된 vSphere with Tanzu을 배포할 수 있습니다.

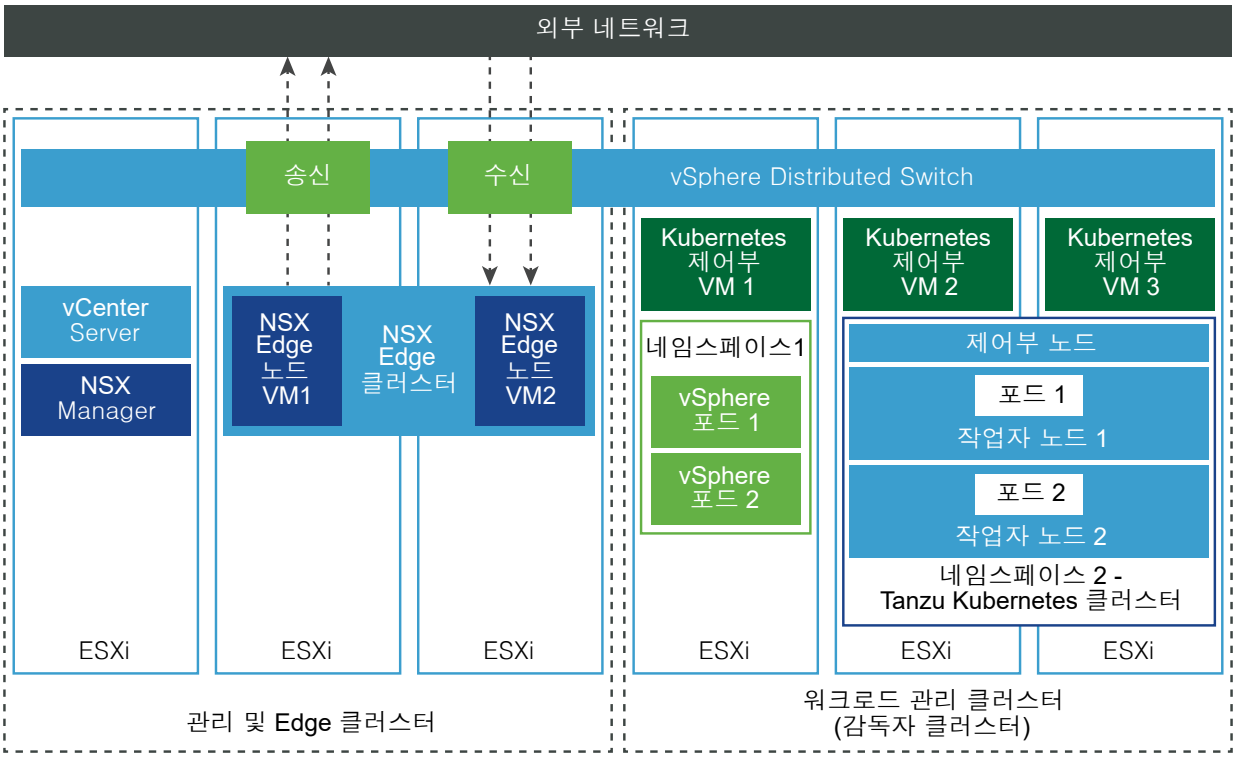
그림 4-4. 관리, Edge 및 워크로드 관리 클러스터



별도의 관리 및 Edge 클러스터 및 워크로드 관리 클러스터가 포함된 토폴로지

두 개의 클러스터, 즉 관리 및 Edge 기능을 위한 클러스터와 워크로드 관리 전용 클러스터에 vSphere with Tanzu을 배포할 수 있습니다.

그림 4-5. 관리 및 Edge 및 워크로드 관리 클러스터



NSX-T Data Center를 사용하여 감독자 클러스터를 구성하기 위한 모범 사례 고려 사항

NSX-T Data Center를 사용하여 vSphere 클러스터를 감독자 클러스터로 구성하는 경우 다음과 같은 모범 사례를 고려하십시오.

- NSX Edge에 vSAN 데이터스토어를 사용합니다.
- vSAN 데이터스토어를 사용하는 경우, 해당 워크로드에 대해 vSAN 환경의 크기를 적절히 조정해야 합니다. vSAN에 커널 메모리가 사용되기 때문에 vSAN 설정에 더 많은 메모리가 필요합니다. 그러면 NSX Edge VM에 사용할 수 있는 메모리가 줄어듭니다. 적절한 크기 조정을 위해 vSAN 계산기를 사용하십시오. 자세한 내용은 vSAN ReadyNode 크기 조정기를 참조하십시오.
- NFS 데이터스토어를 사용하는 경우 클러스터의 모든 호스트에서 공유되는지 확인합니다. NSX Edge 노드마다 고유한 NFS 데이터스토어를 생성합니다.
- NSX Edge 클러스터마다 전용 리소스 풀을 구성합니다. 리소스 풀을 다른 VM과 공유하지 마십시오.
- ESXi 호스트 오버레이를 구성할 때는 1-4094 범위의 VLAN을 사용합니다.
- Edge 오버레이를 구성할 때는 1-4094 범위의 VLAN을 사용합니다.

vSphere with Tanzu에 대한 NSX-T Data Center 설치 및 구성

vSphere with Tanzu를 사용하려면 감독자 클러스터, vSphere 네임스페이스 및 네임스페이스 내에서 실행되는 모든 개체(예: vSphere 포드, VM 및 Tanzu Kubernetes 클러스터)에 대한 연결을 사용하도록 설정

하기 위한 특정 네트워킹 구성이 필요합니다. vSphere 관리자는 vSphere with Tanzu에 대한 NSX-T Data Center를 설치하고 구성합니다.

이 섹션에서는 새 NSX-T Data Center 인스턴스를 배포하여 감독자 클러스터 네트워킹을 구성하는 방법에 대해 설명하지만 절차는 기존 NSX-T Data Center 배포에 대해서도 적용됩니다. 또한 이 섹션에서는 감독자 클러스터 워크로드 도메인을 설정할 때 VMware Cloud Foundation SDDC Manager가 수행하는 작업을 이해하기 위한 배경 지식을 제공합니다.

사전 요구 사항

- 사용 중인 환경이 vSphere 클러스터를 감독자 클러스터로 구성하기 위한 시스템 요구 사항을 충족하는지 확인합니다. 요구 사항에 대한 자세한 내용은 NSX-T Data Center를 사용하여 vSphere with Tanzu를 설정하기 위한 시스템 요구 사항 항목을 참조하십시오.
- 감독자 클러스터의 일부가 될 모든 ESXi 호스트에 VMware vSphere 7 Enterprise Plus with Add-on for Kubernetes 라이센스를 할당합니다.
- 제어부 VM, 포드 사용 후 삭제 디스크 및 컨테이너 이미지를 배치하기 위한 스토리지 정책을 생성합니다.
- 클러스터의 공유 스토리지를 구성합니다. 공유 스토리지는 vSphere DRS, HA 및 컨테이너의 영구 볼륨 저장에 필요합니다.
- vSphere 클러스터에서 DRS 및 HA가 사용되도록 설정되어 있고 DRS가 완전히 자동화된 모드에 있는지 확인합니다.
- 클러스터에서 **클러스터 전체 구성 수정** 권한이 있는지 확인합니다.

vSphere Distributed Switch 생성

감독자 클러스터의 모든 호스트에 대한 네트워킹 구성을 처리하려면 vSphere Distributed Switch를 생성합니다.

절차

1 vSphere Client에서 데이터 센터로 이동합니다.

2 탐색기에서 데이터 센터를 마우스 오른쪽 버튼으로 클릭하고 **Distributed Switch > 새 Distributed Switch**를 선택합니다.

3 새 분산 스위치의 이름을 입력합니다.

예: `DSwitch`

4 **버전 선택**에서 Distributed Switch의 버전을 입력합니다.

7.0.0 - ESXi 7.0 이상을 선택합니다.

5 **설정 구성**에서 업링크 포트 수를 입력합니다.

값 2를 입력합니다.

6 설정을 검토하고 **완료**를 클릭합니다.

7 생성된 분산 스위치를 마우스 오른쪽 버튼으로 클릭하고 **설정 > 설정 편집**을 선택합니다.

8 **고급** 탭에서 MTU(바이트) 값으로 1600을 초과하는 값을 입력하고 **확인**을 클릭합니다.

MTU 크기는 오버레이 트래픽을 전달하는 모든 네트워크에서 1600 이상이어야 합니다.

예: `9000`

다음에 수행할 작업

분산 포트 그룹을 추가합니다. 분산 포트 그룹 생성의 내용을 참조하십시오.

분산 포트 그룹 생성

각 NSX Edge 노드 업링크, Edge 노드 TEP, 관리 네트워크 및 공유 스토리지에 대한 분산 포트 그룹을 생성합니다.

사전 요구 사항

vSphere Distributed Switch를 생성했는지 확인합니다.

절차

1 vSphere Client에서 데이터 센터로 이동합니다.

2 탐색기에서 Distributed Switch를 마우스 오른쪽 버튼으로 클릭하고 **분산 포트 그룹 > 새 분산 포트 그룹**을 선택합니다.

3 NSX Edge 업링크에 대한 포트 그룹을 생성합니다.

예: `DPortGroup-EDGE-UPLINK`

4 **VLAN 유형**을 [VLAN 트렁킹]으로 구성합니다.

5 분산 스위치를 마우스 오른쪽 버튼으로 클릭하고 **작업** 메뉴에서 **분산 포트 그룹 > 분산 포트 그룹 관리**를 선택합니다.

6 **팀 구성 및 페일오버**를 선택하고 **다음**을 클릭합니다.

7 활성 및 대기 업링크를 구성합니다.

예를 들어 활성 업링크는 `Uplink1`이고 대기 업링크는 `Uplink2`입니다.

8 Edge 노드 TEP, 관리 네트워크 및 공유 스토리지에 대해 4-7단계를 반복합니다.

예를 들어 다음과 같은 포트 그룹을 생성합니다.

포트 그룹	이름	VLAN 유형
Edge 노드 TEP	DPortGroup-EDGE-TEP	**VLAN 유형**을 [VLAN 트렁킹]으로 구성합니다. 활성 업링크를 Uplink2로 구성하고 대기 업링크를 Uplink1로 구성합니다. **참고** Edge 노드 TEP에 사용되는 VLAN은 ESXi TEP에 사용되는 VLAN과 달라야 합니다.
관리	DPortGroup-MGMT	**VLAN 유형**을 VLAN으로 구성하고 관리 네트워크의 VLAN ID를 입력합니다. 예: 1060
공유 스토리지 또는 vSAN	DPortGroup-VSAN	**VLAN 유형**을 VLAN으로 구성하고 VLAN ID를 입력합니다. 예: 3082

9 (선택 사항) 다음 구성 요소에 대한 포트 그룹을 생성합니다.

- vSphere vMotion
- VM 트래픽

다음에 수행할 작업

vSphere Distributed Switch에 호스트를 추가합니다. vSphere Distributed Switch에 호스트 추가의 내용을 참조하십시오.

vSphere Distributed Switch에 호스트 추가

vSphere Distributed Switch를 사용하여 사용자 환경의 네트워킹을 관리하려면 호스트를 스위치와 연결해야 합니다. 호스트의 물리적 NIC, VMkernel 어댑터 및 가상 시스템 네트워크 어댑터를 Distributed Switch에 연결합니다.

사전 요구 사항

- 스위치에 연결하려는 물리적 NIC에 할당할 수 있는 업링크가 Distributed Switch에 충분히 있는지 확인합니다.
- Distributed Switch에서 하나 이상의 분산 포트 그룹을 사용할 수 있는지 확인합니다.
- 분산 포트 그룹의 팀 구성 및 페일오버 정책에 활성 업링크가 구성되어 있는지 확인합니다.

절차

1 vSphere Client에서 **네트워킹**을 선택하고 Distributed Switch로 이동합니다.

2 **작업** 메뉴에서 **호스트 추가 및 관리**를 선택합니다.

3 **작업 선택** 페이지에서 **호스트 추가**를 선택하고 **다음**을 클릭합니다.

4 **호스트 선택** 페이지에서 **새 호스트**를 클릭하고 데이터 센터에서 호스트를 선택한 후 **확인**을 클릭하고 **다음**을 클릭합니다.

5 **물리적 어댑터 관리** 페이지에서 Distributed Switch에 물리적 NIC를 구성합니다.

a **다른 스위치/할당되지 않음** 목록에서 물리적 NIC를 선택합니다.

다른 스위치에 이미 연결된 물리적 NIC를 선택하면 현재 Distributed Switch로 마이그레이션됩니다.

b **업링크 할당**을 클릭합니다.

c 업링크를 선택합니다.

d 클러스터의 모든 호스트에 업링크를 할당하려면 **이 업링크 할당을 나머지 호스트에 적용**을 선택합니다.

e **확인**을 클릭합니다.

예를 들어 `Uplink 1`을 `vmnic0`에 할당하고 `Uplink 2`를 `vmnic1`에 할당합니다.

6 **다음**을 클릭합니다.

7 **VMkernel 어댑터 관리** 페이지에서 VMkernel 어댑터를 구성합니다.

a VMkernel 어댑터를 선택하고 **포트 그룹 할당**을 클릭합니다.

b 분산 포트 그룹을 선택합니다.

예를 들어 **DPortGroup**을 선택합니다.

c 클러스터의 모든 호스트에 포트 그룹을 적용하려면 **이 포트 그룹 할당을 나머지 호스트에 적용**을 선택합니다.

d **확인**을 클릭합니다.

8 **다음**을 클릭합니다.

9 (선택 사항) **VM 네트워킹 마이그레이션** 페이지에서 **가상 시스템 네트워킹 마이그레이션** 확인란을 선택하여 가상 시스템 네트워킹을 구성합니다.

a 가상 시스템의 모든 네트워크 어댑터를 분산 포트 그룹에 연결하려면 가상 시스템을 선택하고, 개별 네트워크 어댑터를 연결하려면 해당 네트워크 어댑터만 선택합니다.

b **포트 그룹 할당**을 클릭합니다.

c 목록에서 분산 포트 그룹을 선택하고 **확인**을 클릭합니다.

d **다음**을 클릭합니다.

다음에 수행할 작업

NSX Manager를 배포하고 구성합니다. NSX Manager 배포 및 구성을 참조하십시오.

NSX Manager 배포 및 구성

vSphere Client를 사용하여 NSX Manager를 vSphere 클러스터에 배포하고 vSphere with Tanzu에서 사용할 수 있습니다.

OVA 파일을 사용하여 NSX Manager를 배포하려면 이 절차의 단계를 수행합니다.

사용자 인터페이스 또는 CLI를 통해 NSX Manager를 배포하는 방법에 대한 자세한 내용은 "NSX-T Data Center 설치 가이드" 를 참조하십시오.

사전 요구 사항

- 환경이 네트워킹 요구 사항을 충족하는지 확인합니다. 자세한 내용은 NSX-T Data Center를 사용하여 vSphere with Tanzu를 설정하기 위한 시스템 요구 사항의 내용을 참조하십시오.
- 필수 포트가 열려 있는지 확인합니다. 포트 및 프로토콜에 대한 자세한 내용은 " NSX-T Data Center 설치 가이드" 의 내용을 확인하십시오.

절차

1 VMware 다운로드 포털에서 NSX-T Data Center OVA 파일을 찾습니다.

다운로드 URL을 복사하거나 OVA 파일을 다운로드합니다.

2 마우스 오른쪽 버튼을 클릭하고 **OVF 템플릿 배포**를 선택하여 설치 마법사를 시작합니다.

3 **OVF 템플릿 선택** 탭에서 OVA 다운로드 URL을 입력하거나 OVA 파일로 이동합니다.

4 **이름 및 폴더 선택** 탭에서 NSX Manager VM(가상 시스템)의 이름을 입력합니다.

5 **계산 리소스 선택** 탭에서 NSX Manager를 배포할 vSphere 클러스터를 선택합니다.

6 **다음**을 클릭하고 세부 정보를 검토합니다.

7 **구성** 탭에서 NSX-T 배포 크기를 선택합니다.

권장되는 최소 배포 크기는 중간입니다.

8 **스토리지 선택** 탭에서 배포할 공유 스토리지를 선택합니다.

9 **가상 디스크 형식 선택**에서 **씬 프로비저닝**을 선택하여 씬 프로비저닝을 사용하도록 설정합니다.

가상 디스크는 기본적으로 씩 프로비저닝됩니다.

10 **네트워크 선택** 탭의 **대상 네트워크**에서 NSX Manager에 대한 대상 네트워크 또는 관리 포트 그룹을 선택합니다.

예: `DPortGroup-MGMT`

11 **템플릿 사용자 지정** 탭에서 시스템 루트, CLI 관리자 및 NSX Manager에 대한 감사 암호를 입력합니다. 암호는 암호 강도 제한을 준수 해야 합니다.

- 12자 이상.
- 소문자 하나 이상.
- 대문자 하나 이상.
- 숫자 하나 이상.
- 특수 문자 하나 이상.
- 5개 이상의 다른 문자.

- 기본 암호 복잡성 규칙은 Linux PAM 모듈에 의해 적용됩니다.

12 기본 IPv4 게이트웨이, 관리 네트워크 IPv4, 관리 네트워크 넷마스크, DNS 서버, 도메인 검색 목록 및 NTP IP 주소를 입력합니다.

13 SSH를 사용하도록 설정하고 NSX Manager 명령줄에 루트 SSH 로그인을 허용합니다.

기본적으로 SSH 옵션은 보안상의 이유로 사용되지 않도록 설정됩니다.

14 사용자 지정 OVF 템플릿 규격이 정확한지 확인하고 **마침**을 클릭하여 설치를 시작합니다.

15 NSX Manager가 부팅되면 CLI에 관리자로 로그인하고 `get interface eth0` 명령을 실행하여 IP 주소가 예상대로 적용되었는지 확인합니다.

16 `get services` 명령을 입력하여 모든 서비스가 실행되고 있는지 확인합니다.

NSX Manager 노드를 배포하여 클러스터 구성

NSX Manager 클러스터는 고가용성을 제공합니다. vCenter Server에서 관리하는 ESXi호스트에서만 사용자 인터페이스를 사용하여 NSX Manager 노드를 배포할 수 있습니다. NSX Manager 클러스터를 생성하려면 두 개의 추가 노드를 배포하여 총 3개 노드로 이루어진 클러스터를 구성합니다. UI에서 새 노드를 배포하면 노드가 처음 배포된 노드에 연결되어 클러스터를 구성합니다. 처음 배포된 노드의 모든 저장소 세부 정보 및 암호가 새로 배포된 노드와 동기화됩니다.

사전 요구 사항

- NSX Manager 노드가 설치되었는지 확인합니다.
- 계산 관리자가 구성되어 있는지 확인합니다.
- 필수 포트가 열려 있는지 확인합니다.
- ESXi 호스트에서 데이터스토어가 구성되었는지 확인합니다.
- 사용할 NSX Manager에 대한 IP 주소 및 게이트웨이, DNS 서버 IP 주소, 도메인 검색 목록 및 NTP 서버 IP 주소가 있는지 확인합니다.
- 대상 VM 포트 그룹 네트워크가 있는지 확인합니다. NSX-T Data Center 장치를 관리 VM 네트워크에 배치합니다.

절차

1 브라우저에서 https://<manager-ip-address>로 이동한 후 관리자 권한으로 NSX Manager에 로그인합니다.

2 장치를 배포하려면 **시스템 > 장치 > NSX 장치 추가**를 선택합니다.

3 장치 세부 정보를 입력합니다.

옵션	설명
호스트 이름	노드에 사용할 호스트 이름 또는 FQDN을 입력합니다.
관리 IP/넷마스크	노드에 할당할 IP 주소를 입력합니다.

옵션	설명
관리 게이트웨이	노드에서 사용할 게이트웨이 IP 주소를 입력합니다.
DNS 서버	노드에서 사용할 DNS 서버 IP 주소 목록을 입력합니다.
NTP 서버	NTP 서버 IP 주소 목록을 입력합니다.
노드 크기	옵션에서 **중형(6 vCPU, 24GB RAM, 300GB 스토리지)** 폼 팩터를 선택합니다.

4 장치 구성 세부 정보를 입력합니다.

옵션	설명
계산 관리자	계산 관리자로 구성한 vCenter Server를 선택합니다.
계산 클러스터	노드가 가입해야 하는 클러스터를 선택합니다.
데이터스토어	노드 파일에 대한 데이터스토어를 선택합니다.
가상 디스크 형식	**씬 프로비저닝** 형식을 선택합니다.
네트워크	**네트워크 선택**을 클릭하여 노드에 대한 관리 네트워크를 선택합니다.

5 액세스 및 자격 증명 세부 정보를 입력합니다.

옵션	설명
SSH 사용	버튼을 전환하여 새 노드에 대한 SSH 로그인을 허용합니다.
루트 액세스 사용	버튼을 전환하여 새 노드에 대한 루트 액세스를 허용합니다.
시스템 루트 자격 증명	새 노드에 대한 루트 암호를 설정하고 확인합니다. 암호는 암호 강도 제한을 준수해야 합니다. ■ 12자 이상. ■ 소문자 하나 이상. ■ 대문자 하나 이상. ■ 숫자 하나 이상. ■ 특수 문자 하나 이상. ■ 5개 이상의 다른 문자. ■ 기본 암호 복잡성 규칙은 Linux PAM 모듈에 의해 적용됩니다.
관리 CLI 자격 증명 및 감사 CLI 자격 증명	**루트 암호와 동일** 확인란을 선택하여 루트에 대해 구성한 것과 동일한 암호를 사용하거나 확인란을 선택 취소하고 다른 암호를 설정합니다.

6 **장치 설치**를 클릭합니다.

새 노드가 배포됩니다. **시스템 > 장치** 페이지에서 배포 프로세스를 추적할 수 있습니다. 설치가 완료되고 클러스터가 안정화될 때까지 다른 노드를 추가하지 마십시오.

7 배포, 클러스터 구성 및 저장소 동기화가 완료될 때까지 기다립니다.

가입 및 클러스터 안정화 프로세스는 10~15분 정도 걸릴 수 있습니다. 다른 클러스터 변경 작업을 수행하기 전에 모든 클러스터 서비스 그룹의 상태가 UP인지 확인합니다.

8 노드가 부팅되면 CLI에 관리자로 로그인하고 `get interface eth0` 명령을 실행하여 IP 주소가 예상대로 적용되었는지 확인합니다.

9 클러스터에 노드가 두 개만 있는 경우 다른 장치를 추가합니다. **시스템 > 장치 > NSX 장치 추가**를 선택하고 구성 단계를 반복합니다.

라이센스 추가

NSX Manager를 사용하여 라이센스를 추가합니다.

사전 요구 사항

NSX-T Data Center 고급 라이센스 또는 더 상위의 라이센스를 얻습니다.

절차

1 NSX Manager에 로그인합니다.

2 **시스템 > 라이센스 > 추가**를 선택합니다.

3 라이센스 키를 입력합니다.

4 **추가**를 클릭합니다.

계산 관리자 추가

계산 관리자는 호스트 및 가상 시스템과 같은 리소스를 관리하는 애플리케이션입니다. NSX-T Data Center와 연결된 vCenter Server를 NSX Manager에서 계산 관리자로 구성합니다.

절차

1 NSX Manager에 로그인합니다.

2 **시스템 > 패브릭 > 계산 관리자 > 추가**를 선택합니다.

3 계산 관리자 세부 정보를 입력합니다.

옵션	설명
이름 및 설명	vCenter Server의 이름 및 설명을 입력합니다.
FQDN 또는 IP 주소	vCenter Server의 FQDN 또는 IP 주소를 입력합니다.
사용자 이름 및 암호	vCenter Server 로그인 자격 증명을 입력합니다.

4 **신뢰 사용**을 선택하여 vCenter Server가 NSX-T Data Center와 통신하도록 허용합니다.

5 NSX Manager에 대한 지문 값을 제공하지 않으면 시스템에서 지문을 식별하여 표시합니다.

6 **추가**를 클릭하여 지문을 수락합니다.

결과

잠시 후에 계산 관리자가 vCenter Server에 등록되고 연결 상태가 접속 중으로 변경됩니다. vCenter Server의 FQDN/PNID가 변경되면 NSX Manager에 다시 등록해야 합니다. 자세한 내용은 NSX Manager에 vCenter Server 등록의 내용을 참조하십시오.

참고 vCenter Server 등록이 완료된 후 먼저 계산 관리자를 삭제하지 않은 상태에서 NSX Manager VM의 전원을 끄고 삭제하면 안 됩니다. 이 지침을 따르지 않으면 새 NSX Manager를 배포할 때 동일한 vCenter Server를 다시 등록할 수 없게 됩니다. vCenter Server가 이미 다른 NSX Manager에 등록되어 있다는 오류 메시지가 표시됩니다.

계산 관리자 이름을 클릭하여 세부 정보를 보거나, 계산 관리자를 편집하거나, 계산 관리자에 적용되는 태그를 관리할 수 있습니다.

전송 영역 생성

전송 영역은 특정 네트워크를 사용할 수 있는 호스트 및 VM을 나타냅니다. 전송 영역은 하나 이상의 호스트 클러스터에 걸쳐 있을 수 있습니다.

vSphere 관리자는 기본 전송 영역을 사용하거나 다음을 생성합니다.

- 감독자 클러스터 제어부 VM에서 사용하는 오버레이 전송 영역.
- 물리적 네트워크에 대한 업링크에 사용할 NSX Edge 노드의 VLAN 전송 영역.

절차

1 NSX Manager에 로그인합니다.

2 **시스템 > 패브릭 > 전송 영역 > 추가**를 선택합니다.

3 전송 영역의 이름과 필요한 경우 설명을 입력합니다.

4 트래픽 유형을 선택합니다.

오버레이 또는 **VLAN**을 선택할 수 있습니다.
기본적으로 다음 전송 영역이 존재합니다.

- 이름이 `nsx-vlan-transportzone`인 VLAN 전송 영역
- 이름이 `nsx-overlay-transportzone`인 오버레이 전송 영역

5 (선택 사항) 하나 이상의 업링크 팀 구성 정책 이름을 입력합니다.

전송 영역에 연결된 세그먼트는 이러한 명명된 팀 구성 정책을 사용합니다. 세그먼트에서 일치하는 명명된 팀 구성 정책을 찾지 못하면 기본 업링크 팀 구성 정책이 사용됩니다.

결과

전송 영역 페이지에 새 전송 영역이 나타납니다.

호스트 터널 끝점 IP 주소에 대한 IP 풀 생성

ESXi 호스트 TEP(터널 끝점) 및 Edge 노드에 대한 IP 풀을 생성합니다. TEP는 외부 IP 헤더에 사용되는 소스 및 대상 IP 주소로, 오버레이 프레임의 NSX-T 캡슐화를 시작하고 종료하는 ESXi 호스트를 식별합니다. TEP IP 주소에 DHCP 또는 수동으로 구성한 IP 풀을 사용할 수 있습니다.

절차

1 NSX Manager에 로그인합니다.

2 **네트워킹 > IP 주소 풀 > IP 주소 풀 추가**를 선택합니다.

3 다음 IP 풀 세부 정보를 입력합니다.

옵션	설명
이름 및 설명	IP 풀 이름 및 설명(선택 사항)을 입력합니다. 예: ESXI-TEP-IP-POOL
IP 범위	IP 할당 범위를 입력합니다. 예를 들어 10.197.79.158 - 10.197.79.160입니다.
게이트웨이	게이트웨이 IP 주소를 입력합니다. 예: 10.197.79.253
CIDR	CIDR 표기법으로 네트워크 주소를 입력합니다. 예: 10.197.79.0/24

4 **추가** 및 **적용**을 클릭합니다.

5 2-4단계를 반복하여 Edge 노드에 대한 IP풀을 생성합니다.

예: EDGE-TEP-IP-POOL

6 생성한 TEP IP 풀이 **IP 풀** 페이지에 나열되어 있는지 확인합니다.

호스트 업링크 프로파일 생성

호스트 업링크 프로파일은 ESXi 호스트에서 NSX-T Data Center 세그먼트까지의 업링크에 대한 정책을 정의합니다.

절차

1 NSX Manager에 로그인합니다.

2 **시스템 > 패브릭 > 프로파일 > 업링크 프로파일 > 추가**를 선택합니다.

3 업링크 프로파일 이름을 입력하고 필요한 경우 업링크 프로파일 설명을 입력합니다.

예: ESXI-UPLINK-PROFILE

4 **팀 구성** 섹션에서 **추가**를 클릭하여 명명 팀 구성 정책을 추가하고 **페일오버 명령** 정책을 구성합니다.

활성 업링크 목록이 지정되고 전송 노드의 각 인터페이스가 하나의 활성 업링크에 고정됩니다. 이 구성을 사용하면 여러 활성 업링크를 동시에 사용할 수 있습니다.

5 활성 및 대기 업링크를 구성합니다.

예를 들어 활성 업링크로 uplink-1을 구성하고 대기 업링크로 uplink-2를 구성합니다.

6 전송 VLAN 값을 입력합니다.

업링크 프로파일에 설정된 전송 VLAN은 오버레이 트래픽에 태그를 지정하고 VLAN ID는 터널 끝점(TEP)에서 사용됩니다.

예: 1060

7 MTU 값을 입력합니다.

업링크 프로파일 MTU의 기본값은 1600입니다.

참고 값은 1600 이상이어야 하지만 물리적 스위치 및 vSphere Distributed Switch의 MTU 값보다 높아서는 안 됩니다.

Edge 업링크 프로파일 생성

Edge 가상 시스템 오버레이 트래픽에 대해 하나의 활성 업링크가 있는 페일오버 순서 팀 구성 정책으로 업링크 프로파일을 생성합니다.

절차

1 NSX Manager에 로그인합니다.

2 **시스템 > 패브릭 > 프로파일 > 업링크 프로파일 > 추가**를 선택합니다.

3 업링크 프로파일 이름을 입력하고 필요한 경우 업링크 프로파일 설명을 추가합니다.

예: EDGE-UPLINK-PROFILE

4 **팀 구성** 섹션에서 **추가**를 클릭하여 명명 팀 구성 정책을 추가하고 **페일오버** 정책을 구성합니다.

활성 업링크 목록이 나열되고 전송 노드의 각 인터페이스가 하나의 활성 업링크에 고정됩니다. 이 구성을 사용하면 여러 활성 업링크를 동시에 사용할 수 있습니다.

5 활성 업링크를 구성합니다.

예를 들어 uplink-1을 활성 업링크로 구성합니다.

6 **업링크 프로파일** 페이지에서 업링크를 확인합니다.

전송 노드 프로파일 생성

전송 노드 프로파일은 프로파일이 연결된 특정 클러스터의 호스트에서 NSX-T Data Center가 설치되고 구성되는 방법을 정의합니다.

사전 요구 사항

오버레이 전송 영역을 생성했는지 확인합니다.

절차

1 NSX Manager에 로그인합니다.

2 **시스템 > 패브릭 > 프로파일 > 전송 노드 프로파일 > 추가**를 선택합니다.

3 전송 노드 프로파일의 이름과 필요한 경우 설명을 입력합니다.

예: `HOST-TRANSPORT-NODE-PROFILE`

4 **새 노드 스위치** 섹션에서 **유형**을 `VDS`로 선택합니다.

5 **모드**를 `Standard`으로 선택합니다.

6 목록에서 vCenter Server 및 Distributed Switch 이름을 선택합니다.

예를 들어 `DSwitch`입니다.

7 이전에 생성된 오버레이 전송 영역을 선택합니다.

예: `NSX-OVERLAY-TRANSPORTZONE`

8 이전에 생성된 호스트 업링크 프로파일을 선택합니다.

예: `ESXI-UPLINK-PROFILE`

9 **IP 할당** 목록에서 **IP 풀 사용**을 선택합니다.

10 이전에 생성된 호스트 TEP 풀을 선택합니다.

예: `ESXI-TEP-IP-POOL`

11 **팀 구성 정책 스위치 매핑**에서 편집 아이콘을 클릭하고 NSX-T 업링크 프로파일에 정의된 업링크를 vSphere Distributed Switch 업링크에 매핑합니다.

예를 들어 `uplink-1 (active)`을 `Uplink 1`에 매핑하고 `uplink-2 (standby)`를 `Uplink 2`에 매핑합니다.

12 **추가**를 클릭합니다.

13 생성한 프로파일이 **전송 노드 프로파일** 페이지에 나열되어 있는지 확인합니다.

클러스터에서 NSX-T Data Center 구성

NSX-T Data Center를 설치하고 오버레이 TEP를 준비하려면 vSphere 클러스터에 전송 노드 프로파일을 적용합니다.

사전 요구 사항

전송 노드 프로파일을 생성했는지 확인합니다.

절차

1 NSX Manager에 로그인합니다.

2 **시스템 > 패브릭 > 노드 > 호스트 전송 노드**를 선택합니다.

3 **관리자** 드롭다운 메뉴에서 기존 vCenter Server를 선택합니다.

페이지에 사용 가능한 vSphere 클러스터가 나열됩니다.

4 NSX-T Data Center를 구성하려는 계산 클러스터를 선택합니다.

5 **NSX 구성**을 클릭합니다.

6 이전에 생성한 전송 노드 프로파일을 선택하고 **적용**을 클릭합니다.

예: `HOST-TRANSPORT-NODE-PROFILE`

7 **호스트 전송 노드** 페이지에서 NSX-T Data Center 구성 상태가 `Success`이고 클러스터에 있는 호스트의 NSX Manager 연결 상태가 `Up`인지 확인합니다.

결과

이전에 생성된 전송 노드 프로파일이 vSphere 클러스터에 적용되어 NSX-T Data Center를 설치하고 오버레이 TEP를 준비합니다.

NSX Edge 전송 노드 구성 및 배포

NSX Edge VM(가상 시스템)을 NSX-T Data Center 패브릭에 추가하고 이를 NSX Edge 전송 노드 VM으로 구성할 수 있습니다.

사전 요구 사항

전송 영역, Edge 업링크 프로파일 및 Edge TEP IP 풀을 생성했는지 확인합니다.

절차

1 NSX Manager에 로그인합니다.

2 **시스템 > 패브릭 > 노드 > Edge 전송 노드 > Edge VM 추가**를 선택합니다.

3 **이름 및 설명**에서 NSX Edge의 이름을 입력합니다.

예를 들어 `nsx-edge-1`입니다.

4 vCenter Server의 호스트 이름이나 FQDN을 입력합니다.

예: `nsx-edge-1.lab.com`

5 `Large` 폼 팩터를 선택합니다.

6 **자격 증명**에서 NSX Edge에 대한 루트 암호 및 CLI를 입력합니다. 암호는 암호 강도 제한을 준수 해야 합니다.

- 12자 이상.
- 소문자 하나 이상.
- 대문자 하나 이상.
- 숫자 하나 이상.
- 특수 문자 하나 이상.

- 5개 이상의 다른 문자.
- 기본 암호 복잡성 규칙은 Linux PAM 모듈에 의해 적용됩니다.

7 CLI 및 루트 자격 증명에 대해 **SSH 로그인 허용**을 사용하도록 설정합니다.

8 **배포 구성**에서 다음 속성을 구성합니다.

옵션	설명
계산 관리자	드롭다운 메뉴에서 계산 관리자를 선택합니다. 예를 들어 vCenter를 선택합니다.
클러스터	드롭다운 메뉴에서 클러스터를 선택합니다. 예를 들어 Compute-Cluster를 선택합니다.
데이터스토어	목록에서 공유 데이터스토어를 선택합니다. 예: vsanDatastore

9 노드 설정을 구성합니다.

옵션	설명
IP 할당	[정적]을 선택합니다. 다음에 대한 값을 입력합니다. ■ **관리 IP**: vCenter Server 관리 네트워크와 동일한 VLAN의 IP 주소를 입력합니다. 예: 10.197.79.146/24 ■ **기본 게이트웨이**: 관리 네트워크의 기본 게이트웨이입니다. 예: 10.197.79.253
관리 인터페이스	**인터페이스 선택**을 클릭하고 드롭다운 메뉴에서 이전에 생성한 관리 네트워크와 동일한 VLAN에 있는 vSphere Distributed Switch 포트 그룹을 선택합니다. 예: DPortGroup-MGMT

10 **NSX 구성**에서 **스위치 추가**를 클릭하여 스위치 속성을 구성합니다.

11 **Edge 스위치 이름**의 기본 이름을 사용합니다.

예: nvds1

12 전송 노드가 속하는 전송 영역을 선택합니다.

이전에 생성된 오버레이 전송 영역을 선택합니다.

예: nsx-overlay-transportzone

13 이전에 생성 된 Edge 업링크 프로파일을 선택합니다.

예: EDGE-UPLINK-PROFILE

14 **IP 할당**에서 **IP 풀 사용**을 선택합니다.

15 이전에 생성된 Edge TEP IP 풀을 선택합니다.

예: EDGE-TEP-IP-POOL

16 **팀 구성 정책 스위치 매핑** 섹션에서 업링크를 이전에 생성된 Edge 업링크 프로파일에 매핑합니다.

예를 들어 Uplink1의 경우 DPortGroup-EDGE-TEP를 선택합니다.

17 10-16단계를 반복하여 새 스위치를 추가합니다.

예를 들어 다음 값을 구성합니다.

속성	값
Edge 스위치 이름	nvds2
전송 영역	nsx-vlan-transportzone
Edge 업링크 프로파일	EDGE-UPLINK-PROFILE
팀 구성 정책 스위치 매핑	DPortGroup-EDGE-UPLINK

18 **마침**을 클릭합니다.

19 두 번째 NSX Edge VM에 대해 2-18단계를 반복합니다.

20 **Edge 전송 노드** 페이지에서 연결 상태를 확인합니다.

NSX Edge 클러스터 생성

하나 이상의 NSX Edge를 항상 사용할 수 있도록 하려면 NSX Edge 클러스터를 생성합니다.

절차

1 NSX Manager에 로그인합니다.

2 **시스템 > 패브릭 > 노드 > Edge 클러스터 > 추가**를 선택합니다.

3 NSX Edge 클러스터 이름을 입력합니다.

예: EDGE-CLUSTER

4 드롭다운 메뉴에서 기본 NSX Edge 클러스터 프로파일을 선택합니다.

nsx-default-edge-high-availability-profile을 선택합니다.

5 **멤버 유형** 드롭다운 메뉴에서 **Edge 노드**를 선택합니다.

6 **사용 가능** 열에서 이전에 생성된 NSX Edge VM을 선택하고 오른쪽 화살표를 클릭하여 **선택됨** 열로 VM을 이동합니다.

7 예를 들어 nsx-edge-1 및 nsx-edge-2를 선택합니다.

8 **저장**을 클릭합니다.

Tier-0 업링크 세그먼트 생성

Tier-0 업링크 세그먼트는 NSX-T Data Center의 North-South 연결을 물리적 인프라에 제공합니다.

사전 요구 사항

Tier-0 게이트웨이를 생성했는지 확인합니다.

절차

1 NSX Manager에 로그인합니다.

2 **네트워킹 > 세그먼트 > 세그먼트 추가**를 선택합니다.

3 세그먼트의 이름을 입력합니다.

예: `TIER-0-LS-UPLINK`

4 이전에 생성한 전송 영역을 선택합니다.

예를 들어 `nsx-vlan-transportzone`를 선택합니다.

5 **관리 상태**를 전환하여 사용하도록 설정합니다.

6 Tier-0 게이트웨이의 VLAN ID를 입력합니다.

예: `1089`

7 **저장**을 클릭합니다.

Tier-0 게이트웨이 생성

Tier-0 게이트웨이는 NSX-T Data Center 논리적 네트워킹에 대한 North-South 연결을 물리적 인프라에 제공하는 NSX-T Data Center 논리적 라우터입니다. vSphere with Tanzu는 동일한 전송 영역에 있는 여러 NSX Edge 클러스터에서 여러 개의 Tier-0 게이트웨이를 지원합니다.

사전 요구 사항

NSX Edge 클러스터를 생성했는지 확인합니다.

절차

1 NSX Manager에 로그인합니다.

2 **네트워킹 > Tier-0 게이트웨이**를 선택합니다.

3 **Tier-0 게이트웨이 추가**를 클릭합니다.

4 Tier-0 게이트웨이의 이름을 입력합니다.

예: `Tier-0_VWT`

5 활성-대기 HA 모드를 선택합니다.

활성-대기 모드에서는 선택된 활성 멤버가 모든 트래픽을 처리합니다. 활성 멤버가 실패하면 새 멤버가 활성 상태로 선택됩니다.

6 이전에 생성한 NSX Edge 클러스터를 선택합니다.

예를 들어 `EDGE-CLUSTER`를 선택합니다.

7 **저장**을 클릭합니다.

Tier-0 게이트웨이가 생성됩니다.

8 **예**를 선택하여 구성을 계속합니다.

9 인터페이스를 구성합니다.

a **인터페이스**를 확장하고 **설정**을 클릭합니다.

b **인터페이스 추가**를 클릭합니다.

c 이름을 입력하십시오.

예를 들어 이름 `TIER-0_VWT-UPLINK1`을 입력합니다.

d **유형**을 **외부**로 선택합니다.

e Edge 논리적 라우터 - 업링크 VLAN에서 IP 주소를 입력합니다. IP 주소는 이전에 생성된 NSX Edge VM에 대해 구성된 관리 IP 주소와 달라야 합니다.

예: `10.197.154.1/24`

f **연결 대상**에서 이전에 생성된 Tier-0 업링크 세그먼트를 선택합니다.

예를 들어 `TIER-0-LS-UPLINK`입니다.

g 목록에서 NSX Edge 노드를 선택합니다.

예: `nsx-edge-1`

h **저장**을 클릭합니다.

i 두 번째 인터페이스에 대해 a - h 단계를 반복합니다.

예를 들어 `nsx-edge-2` Edge 노드에 연결된 IP 주소가 `10.197.154.2/24`인 두 번째 업링크 `TIER-0_VWT-UPLINK2`를 생성합니다.

j **닫기**를 클릭합니다.

10 고가용성을 구성하려면 **HA VIP 구성**에서 **설정**을 클릭합니다.

a **HA VIP 구성 추가**를 클릭합니다.

b IP 주소를 입력합니다.

예를 들어 `10.197.154.3/24`입니다.

c 인터페이스를 선택합니다.

예를 들어 `TIER-0_WVT-UPLINK1` 및 `TIER-0_WVT-UPLINK2`를 선택합니다.

d **추가** 및 **적용**을 클릭합니다.

11 라우팅을 구성하려면 **라우팅**을 클릭합니다.

a 정적 경로에서 **설정**을 클릭합니다.

b **정적 경로 추가**를 클릭합니다.

c 이름을 입력하십시오.

예: `DEFAULT-STATIC-ROUTE`

d 네트워크 IP 주소로 `0.0.0.0/0`을 입력합니다.

e 다음 홉을 구성하려면 **다음 홉 설정**을 클릭한 후 **다음 홉 추가**를 클릭합니다.

f 다음 홉 라우터의 IP 주소를 입력합니다. 일반적으로 이것은 NSX Edge 논리적 라우터 업링크 VLAN에서 관리 네트워크 VLAN의 기본 게이트웨이입니다.

예: `10.197.154.253`

g **추가** 및 **적용**을 클릭하고 **저장**을 클릭합니다.

h **닫기**를 클릭합니다.

12 연결을 확인하려면 물리적 아키텍처의 외부 디바이스에서 사용자가 구성한 업링크를 ping할 수 있는지 확인합니다.

다음에 수행할 작업

감독자 클러스터를 구성합니다. NSX-T Data Center 네트워킹으로 워크로드 관리 사용 항목을 참조하십시오.

vSphere with Tanzu에 대한 vSphere 네트워킹 및 NSX Advanced Load Balancer 구성

vSphere with Tanzu는 NSX Advanced Load Balancer(Avi Load Balancer라고도 함) Essentials 및 Enterprise 버전을 지원합니다. NSX Advanced Load Balancer에 대해 VDS(vSphere Distributed Switch) 네트워킹을 사용하는 경우에만 vSphere with Tanzu 환경에 감독자 클러스터 20.1.7을 설치하고 구성할 수 있습니다.

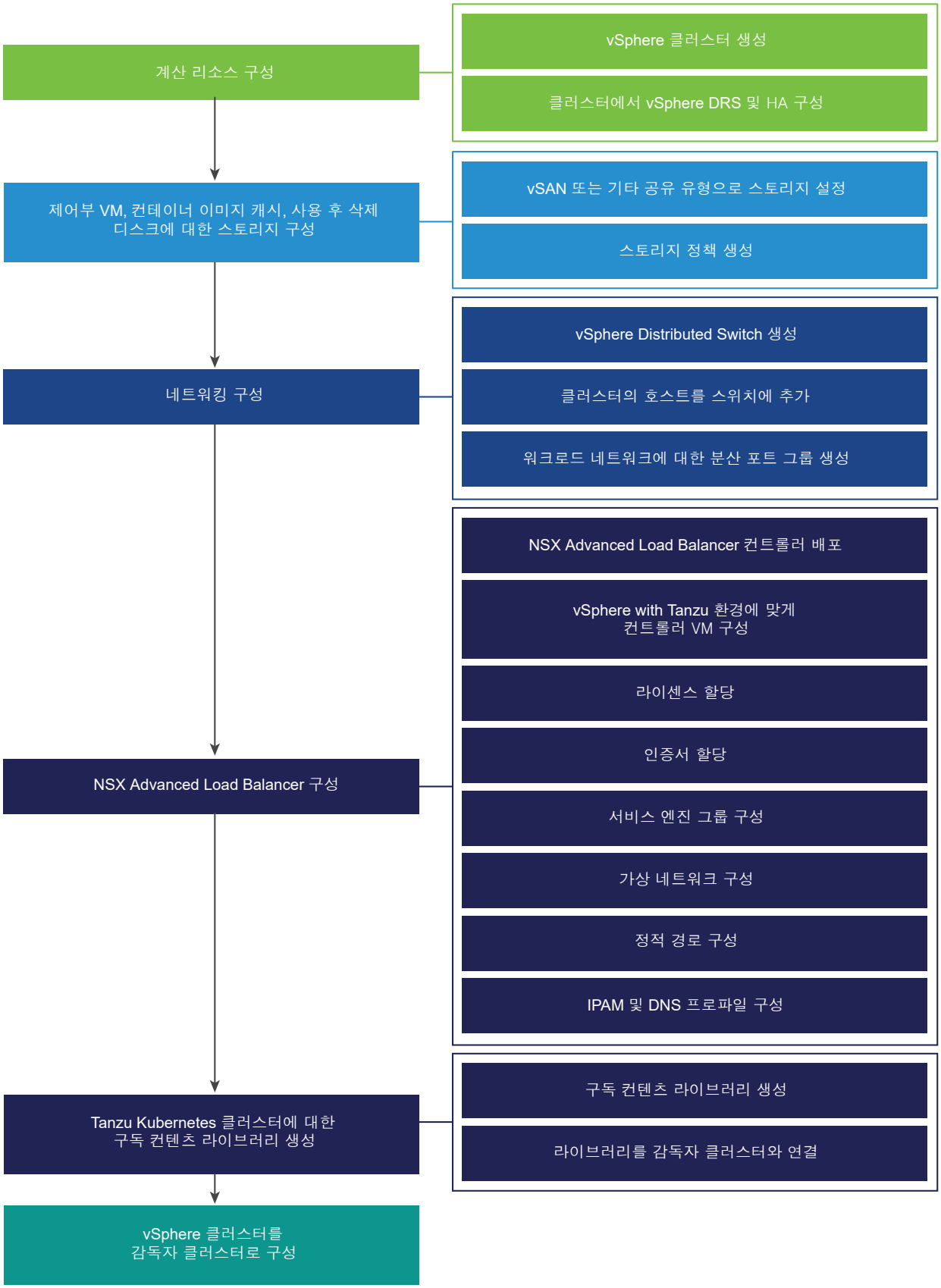
로드 밸런서가 Tanzu Kubernetes 클러스터에서 작동하는 방식

NSX Advanced Load Balancer는 Tanzu Kubernetes Grid 서비스에서 프로비저닝되는 Tanzu Kubernetes 클러스터에 대해 동적으로 확장/축소하는 로드 밸런싱 끝점을 제공합니다. Avi 컨트롤러 VM을 설치하고 구성합니다. 컨트롤러를 구성하면 로드 밸런싱 끝점이 자동으로 프로비저닝됩니다. 예를 들어 Tanzu Kubernetes 클러스터를 프로비저닝하면 컨트롤러는 가상 서비스를 생성하고 이 서비스를 호스팅할 서비스 엔진 VM을 배포합니다. 이 가상 서비스는 Kubernetes 제어부에 대한 로드 밸런싱을 제공합니다.

해당 클러스터에 대해 로드 밸런서 유형의 Kubernetes 서비스를 생성하면 컨트롤러는 가상 서비스를 자동으로 생성하여 서비스 엔진에 배포합니다. 첫 번째 서비스 엔진은 첫 번째 가상 서비스가 구성된 후에만 생성됩니다. 후속으로 구성된 가상 서비스는 기존 서비스 엔진을 사용합니다. 하나의 서비스 엔진 VM에 여러 가상 서비스를 배포할 수 있습니다.

vSphere 네트워킹 및 NSX Advanced Load Balancer 워크플로가 포함된 감독자 클러스터

이 다이어그램은 vSphere with Tanzu에 대한 vSphere 네트워킹 및 NSX Advanced Load Balancer를 구성하는 워크플로를 보여줍니다.


계산 리소스 구성
vSphere 클러스터 생성
클러스터에서 vSphere DRS 및 HA 구성
제어부 VM, 컨테이너 이미지 캐시, 사용 후 삭제 디스크에 대한 스토리지 구성
vSAN 또는 기타 공유 유형으로 스토리지 설정
스토리지 정책 생성
네트워킹 구성
vSphere Distributed Switch 생성
클러스터의 호스트를 스위치에 추가
워크로드 네트워크에 대한 분산 포트 그룹 생성
NSX Advanced Load Balancer 구성
NSX Advanced Load Balancer 컨트롤러 배포
vSphere with Tanzu 환경에 맞게 컨트롤러 VM 구성
라이센스 할당
인증서 할당
서비스 엔진 그룹 구성
가상 네트워크 구성
정적 경로 구성
IPAM 및 DNS 프로파일 구성
Tanzu Kubernetes 클러스터에 대한 구독 컨텐츠 라이브러리 생성
구독 컨텐츠 라이브러리 생성
라이브러리를 감독자 클러스터와 연결
vSphere 클러스터를 감독자 클러스터로 구성

절차

1 NSX Advanced Load Balancer 구성 요소

Avi Load Balancer라고도 하는 NSX Advanced Load Balancer의 구성 요소에는 Avi 컨트롤러 클러스터, 서비스 엔진(데이터부) VM 및 AKO(Avi Kubernetes Operator)가 포함되어 있습니다.

2 vSphere 네트워킹 및 NSX Advanced Load Balancer를 사용하여 vSphere with Tanzu를 설정하기 위한 시스템 요구 사항

Avi Load Balancer라고도 하는 NSX Advanced Load Balancer를 사용하여 vSphere with Tanzu를 구성하려면 환경이 특정 요구 사항을 충족해야 합니다. vSphere with Tanzu는 Avi 네트워킹을 위한 여러 토폴로지(Avi 서비스 엔진 및 로드 밸런서 서비스용 단일 VDS 네트워크, Avi 관리부용 VDS, NSX Advanced Load Balancer용 다른 VDS)를 지원합니다.

3 vSphere 네트워킹 및 NSX Advanced Load Balancer를 사용하는 감독자 클러스터용 토폴로지

Avi 컨트롤러는 항상 vCenter Server, ESXi 호스트 및 감독자 클러스터 제어부 노드와 상호 작용할 수 있는 관리 네트워크에 배포됩니다. 서비스 엔진은 관리 네트워크 및 데이터 네트워크에 대한 인터페이스와 함께 배포됩니다.

4 NSX Advanced Load Balancer 설치 및 구성

VDS(vSphere Distributed Switch) 네트워킹을 사용하는 경우 vSphere with Tanzu 환경에 NSX Advanced Load Balancer 20.1.7을 설치하고 구성할 수 있습니다.

NSX Advanced Load Balancer 구성 요소

Avi Load Balancer라고도 하는 NSX Advanced Load Balancer의 구성 요소에는 Avi 컨트롤러 클러스터, 서비스 엔진(데이터부) VM 및 AKO(Avi Kubernetes Operator)가 포함되어 있습니다.

컨트롤러

컨트롤러라고도 하는 Avi 컨트롤러는 vCenter Server과 상호 작용하여 Tanzu Kubernetes 클러스터에 대한 로드 밸런싱을 자동화합니다. 컨트롤러는 서비스 엔진 프로비저닝, 서비스 엔진 전반의 리소스 조정, 서비스 엔진 메트릭 및 로깅 집계를 담당합니다. 컨트롤러는 사용자 작업 및 프로그래밍 방식 통합을 위한 API, 웹 인터페이스, 명령줄 인터페이스를 제공합니다.

vSphere에서 컨트롤러 VM을 배포하고 구성한 후 HA에 대한 제어부 클러스터를 설정하는 방법에 대한 자세한 내용은 컨트롤러 클러스터 배포 항목을 참조하십시오.

서비스 엔진

서비스 엔진이라고도 하는 Avi 서비스 엔진은 데이터부 가상 시스템입니다. 서비스 엔진은 하나 이상의 가상 서비스를 실행합니다. 서비스 엔진은 컨트롤러에 의해 관리됩니다. 컨트롤러는 가상 서비스를 호스팅하는 서비스 엔진을 프로비저닝합니다.

서비스 엔진에는 두 가지 유형의 네트워크 인터페이스가 있습니다.

- 첫 번째 네트워크 인터페이스인 VM의 `vnic0`은 Avi 컨트롤러에 연결할 수 있는 관리 네트워크에 연결됩니다.

- 나머지 인터페이스인 `vnic1 - 8`은 가상 서비스가 실행되는 데이터 네트워크에 연결됩니다.

서비스 엔진 인터페이스는 올바른 VDS 포트 그룹에 자동으로 연결됩니다. 사용되지 않은 인터페이스는 자동으로 생성되어 나중에 사용할 수 있도록 예약된 포트 그룹 `Avi Internal`에 연결됩니다. 각 서비스 엔진은 가상 서비스를 1000개까지 지원할 수 있습니다.

가상 서비스는 Tanzu Kubernetes 클러스터 워크로드에 대한 계층 4 및 계층 7 로드 밸런싱 서비스를 제공합니다. 가상 서비스는 하나의 가상 IP와 여러 포트로 구성됩니다. 가상 서비스가 배포되면 컨트롤러는 ESX 서버를 자동으로 선택하고 서비스 엔진을 가동하여 올바른 네트워크(포트 그룹)에 연결합니다.

첫 번째 서비스 엔진은 첫 번째 가상 서비스가 구성된 후에만 생성됩니다. 후속으로 구성된 가상 서비스는 기존 서비스 엔진을 사용합니다.

각 가상 서버는 Tanzu Kubernetes 클러스터에 대한 로드 밸런서 유형의 고유 IP 주소를 사용하여 계층 4 로드 밸런서를 노출합니다. 각 가상 서버에 할당된 IP 주소는 서버를 구성할 때 컨트롤러에 제공된 IP 주소 블록에서 선택됩니다.

AVI에는 네이티브 IPAM 및 외부 IPAM 제공자 지원이 포함됩니다. vSphere에서는 AVI 네이티브 IPAM이 활용됩니다.

Avi Kubernetes Operator

AKO(Avi Kubernetes Operator)는 Kubernetes 리소스를 감시하고 컨트롤러와 통신하여 해당 로드 밸런싱 리소스를 요청합니다.

Avi Kubernetes Operator는 사용 설정 프로세스의 일부로 감독자 클러스터에 설치됩니다.

vSphere 네트워킹 및 NSX Advanced Load Balancer를 사용하여 vSphere with Tanzu를 설정하기 위한 시스템 요구 사항

Avi Load Balancer라고도 하는 NSX Advanced Load Balancer를 사용하여 vSphere with Tanzu를 구성하려면 환경이 특정 요구 사항을 충족해야 합니다. vSphere with Tanzu는 Avi 네트워킹을 위한 여러 토폴로지(Avi 서비스 엔진 및 로드 밸런서 서비스용 단일 VDS 네트워크, Avi 관리부용 VDS, NSX Advanced Load Balancer용 다른 VDS)를 지원합니다.

워크로드 네트워크

vSphere 네트워킹 스택을 사용하여 감독자 클러스터를 구성하려면 클러스터의 모든 호스트를 vSphere Distributed Switch에 연결해야 합니다. 감독자 클러스터에 대해 구현하는 토폴로지에 따라 하나 이상의 분산 포트 그룹을 생성합니다. 포트 그룹을 vSphere 네임스페이스에 대한 워크로드 네트워크로 지정합니다.

감독자 클러스터에 호스트를 추가하기 전에 클러스터에 포함된 모든 vSphere Distributed Switch에 호스트를 추가해야 합니다.

워크로드 네트워크는 Tanzu Kubernetes 클러스터 노드 및 감독자 클러스터 제어부 VM에 대한 연결을 제공합니다. Kubernetes 제어부 VM에 대한 연결을 제공하는 워크로드 네트워크를 기본 워크로드 네트워크라고 합니다. 각 감독자 클러스터에는 기본 워크로드 네트워크가 하나씩 있어야 합니다. 분산 포트 그룹 중 하나를 감독자 클러스터에 대한 기본 워크로드 네트워크로 지정해야 합니다.

감독자 클러스터의 Kubernetes 제어부 VM은 기본 워크로드 네트워크에 할당된 IP 주소 범위에서 3개의 IP 주소를 사용합니다. Tanzu Kubernetes 클러스터의 각 노드에는 Tanzu Kubernetes 클러스터가 실행되는 네임스페이스로 구성된 워크로드 네트워크의 주소 범위에서 할당된 별도의 IP 주소가 있습니다.

네트워킹 요구 사항

NSX Advanced Load Balancer에는 라우팅 가능한 서브넷이 2개 필요합니다.

- 관리 네트워크. 관리 네트워크는 컨트롤러라고도 하는 Avi 컨트롤러가 상주하는 곳입니다. 관리 네트워크는 컨트롤러에 vCenter Server, ESXi 호스트, 감독자 클러스터 제어부 노드에 대한 연결을 제공합니다. 이 네트워크에는 Avi 서비스 엔진의 관리 인터페이스가 배치됩니다. 이 네트워크에는 VDS(vSphere Distributed Switch) 및 분산 포트 그룹이 필요합니다.
- 데이터 네트워크. 서비스 엔진이라고도 하는 Avi 서비스 엔진의 데이터 인터페이스가 이 네트워크에 연결됩니다. 로드 밸런서 VIP(가상 IP)는 이 네트워크에서 할당됩니다. 이 네트워크에는 VDS(vSphere Distributed Switch) 및 분산 포트 그룹이 필요합니다. 로드 밸런서를 설치하기 전에 VDS 및 포트 그룹을 반드시 구성해야 합니다.

IP 주소 할당

컨트롤러와 서비스 엔진은 관리 네트워크에 연결됩니다. NSX Advanced Load Balancer를 설치하고 구성할 때 각 컨트롤러 VM에 정적, 라우팅 가능 IP 주소를 제공합니다.

서비스 엔진은 DHCP를 사용할 수 있습니다. DHCP를 사용할 수 없는 경우 서비스 엔진에 대한 IP 주소 풀을 구성할 수 있습니다.

자세한 내용은 기본 게이트웨이 구성의 내용을 참조하십시오.

최소 계산 요구 사항

이 표에는 NSX Advanced Load Balancer를 사용하는 vSphere 네트워킹에 대한 최소 계산 요구 사항이 나열되어 있습니다.

경고 감독자 클러스터를 구성한 후에는 vSphere DRS를 사용하지 않도록 설정하지 마십시오. 감독자 클러스터에서 워크로드를 실행하려면 항상 DRS를 사용하도록 설정해야 합니다. DRS를 사용하지 않도록 설정하면 Tanzu Kubernetes 클러스터가 중단됩니다.

표 4-5. 최소 계산 요구 사항

시스템	최소 배포 크기	CPU	메모리	스토리지
vCenter Server 7.0, 7.0.2, 7.0.3	작음	2	16GB	290GB
ESXi 호스트 7.0	호스트당 고정 IP가 1개인 ESXi 호스트 3개. vSAN을 사용하는 경우: 물리적 NIC가 2개 이상인 ESXi 호스트 3개가 최소입니다. 단, 패치 적용 및 업그레이드 중에는 복원력을 위해 ESXi 호스트 4개가 권장됩니다. 호스트는 vSphere DRS 및 HA를 사용하도록 설정된 클러스터에 가입되어 있어야 합니다. vSphere DRS는 완전히 자동화되거나 부분적으로 자동화된 모드여야 합니다. **참고** 클러스터에 참여하는 호스트의 이름에 소문자를 사용하는지 확인합니다. 그렇지 않으면 워크로드 관리를 위한 클러스터 사용 설정이 실패할 수 있습니다.	8	호스트당 64GB	해당 없음
Kubernetes 제어부 VM	3	4	16GB	16GB
Avi 컨트롤러	Essentials Enterprise **참고** 소규모 배포의 경우 Essentials 크기 컨트롤러를 단일 컨트롤러 노드로 배포할 수 있습니다. Avi 컨트롤러 클러스터를 생성할 수 있지만 성능상의 이점이 없으며 낮은 리소스 활용률의 목적을 달성하지 못합니다. 이 경우 재해 복구에 원격 백업을 사용할 수 있습니다. 이 크기는 Avi Essentials 라이센싱 모드에서만 사용해야 하며 50개의 가상 서비스와 10개의 서비스 엔진으로 제한됩니다. 운영 환경의 경우 3개의 Avi 컨트롤러 VM으로 구성된 클러스터를 설치하는 것이 좋습니다. HA에는 최소 2개의 서비스 엔진 VM이 필요합니다.	4 8	12 GB 24GB	128 GB 128 GB
서비스 엔진	HA에는 최소 2개의 서비스 엔진 VM이 필요합니다.	1	2GB	15 GB

최소 네트워크 요구 사항

이 표에는 NSX Advanced Load Balancer를 사용하는 vSphere 네트워킹에 대한 최소 네트워크 요구 사항이 나열되어 있습니다.

참고 vSphere 7 감독자 클러스터를 사용하여 IPv6 클러스터를 생성하거나 IPv6 클러스터를 Tanzu Mission Control에 등록할 수 없습니다. NSX Advanced Load Balancer 서비스는 현재 IPv6을 지원하지 않습니다.

표 4-6. 최소 네트워킹 요구 사항

구성 요소	최소 수량	필수 구성
Kubernetes 제어부 VM에 대한 고정 IP	5개의 블록	관리 네트워크에서 감독자 클러스터의 Kubernetes 제어부 VM으로 할당할 5개의 연속적 고정 IP 주소 블록.
관리 트래픽 네트워크	1	ESXi 호스트, vCenter Server, 감독자 클러스터 및 로드 밸런서로 라우팅할 수 있는 관리 네트워크. 이미지 레지스트리가 외부 네트워크에 있는 경우 네트워크는 이미지 레지스트리에 액세스할 수 있어야 하고 인터넷에 연결되어 있어야 합니다. DNS를 통해 이미지 레지스트리를 확인할 수 있어야 합니다.
vSphere Distributed Switch 7.0 이상	1	클러스터의 모든 호스트가 vSphere Distributed Switch에 연결되어 있어야 합니다.
워크로드 네트워크	1	기본 워크로드 네트워크로 구성한 vSphere Distributed Switch에 분산 포트 그룹을 최소 하나 생성해야 합니다. 선택한 토폴로지에 따라 네임스페이스의 워크로드 네트워크와 동일한 분산 포트 그룹을 사용하거나 더 많은 포트 그룹을 생성하고 이를 워크로드 네트워크로 구성할 수 있습니다. 워크로드 네트워크는 다음 요구 사항을 충족해야 합니다. ■ Tanzu Kubernetes 클러스터 트래픽에 사용되는 워크로드 네트워크는 서로 간에 그리고 감독자 클러스터 기본 워크로드 네트워크 간에 라우팅이 가능해야 합니다. ■ NSX Advanced Load Balancer가 가상 IP 할당에 사용하는 네트워크가 있는 모든 워크로드 네트워크 간에 라우팅이 가능해야 합니다. ■ 감독자 클러스터 내의 모든 워크로드 네트워크에서 IP 주소 범위가 겹치지 않아야 합니다.
NTP 및 DNS 서버	1	vCenter Server에서 사용할 수 있는 DNS 서버 및 NTP 서버. **참고** 모든 ESXi 호스트 및 vCenter Server에서 NTP를 구성합니다.

표 4-6. 최소 네트워킹 요구 사항 (계속)

구성 요소	최소 수량	필수 구성
DHCP 서버	1	선택 사항입니다. 관리 및 워크로드 네트워크와 부동 IP에 대한 IP 주소를 자동으로 획득하도록 DHCP 서버를 구성합니다. DHCP 서버는 클라이언트 식별자를 지원하고 호환되는 DNS 서버, DNS 검색 도메인 및 NTP 서버를 제공해야 합니다. 관리 네트워크의 경우 제어부 VM IP, 부동 IP, DNS 서버, DNS, 검색 도메인 및 NTP 서버와 같은 모든 IP 주소가 DHCP 서버에서 자동으로 획득됩니다. DHCP 구성은 감독자 클러스터에서 사용됩니다. 로드 밸런서에는 관리를 위한 고정 IP 주소가 필요할 수 있습니다. DHCP 범위는 이러한 고정 IP와 겹치지 않아야 합니다. DHCP는 가상 IP에 사용되지 않습니다. (VIP)
관리 네트워크 서브넷	1	관리 네트워크는 컨트롤러라고도 하는 Avi 컨트롤러가 상주하는 곳입니다. 또한 서비스 엔진 관리 인터페이스가 연결되는 위치이기도 합니다. Avi 컨트롤러는 이 네트워크의 ESXi 관리 IP 및 vCenter Server에 연결되어야 합니다. **참고** 관리 네트워크와 워크로드 네트워크는 서로 다른 서브넷에 있어야 합니다. 관리 및 워크로드 네트워크에 동일한 서브넷을 할당하는 것은 지원되지 않으며 시스템 오류 및 문제가 발생할 수 있습니다.
데이터 네트워크 서브넷	1	서비스 엔진이라고도 하는 Avi 서비스 엔진의 데이터 인터페이스가 이 네트워크에 연결됩니다. 서비스 엔진에 대한 IP 주소 풀을 구성합니다. 로드 밸런서 VIP(가상 IP)는 이 네트워크에서 할당됩니다.
물리적 네트워크 MTU	1500	MTU 크기는 모든 vSphere Distributed Switch 포트 그룹에서 1500 이상이어야 합니다.
vSphere 포드 CIDR 범위	/개인 IP 주소 24개	vSphere 포드의 IP 주소를 제공하는 개인 CIDR 범위입니다.
Avi 컨트롤러 IP	1 또는 4	Avi 컨트롤러를 단일 노드로 배포하는 경우 해당 관리 인터페이스에 하나의 고정 IP가 필요합니다. 3노드 클러스터의 경우 4개의 IP 주소가 필요합니다. 각 Avi 컨트롤러 VM에 대해 1개씩 그리고 클러스터 VIP에 대해 1개입니다. 이러한 IP는 관리 네트워크 서브넷의 IP여야 합니다.
VIP IPAM 범위	-	Kubernetes 서비스에 IP 주소를 할당할 개인 CIDR 범위. IP는 데이터 네트워크 서브넷의 IP여야 합니다. 각 감독자 클러스터에 대해 고유한 Kubernetes 서비스 CIDR 범위를 지정해야 합니다.
NTP 및 DNS 서버	1	Avi 컨트롤러가 vCenter Server 및 ESXi 호스트 이름을 올바르게 확인하려면 DNS 서버 IP가 필요합니다. 공용 NTP 서버가 기본적으로 사용되므로 NTP는 선택 사항입니다.

포트 및 프로토콜

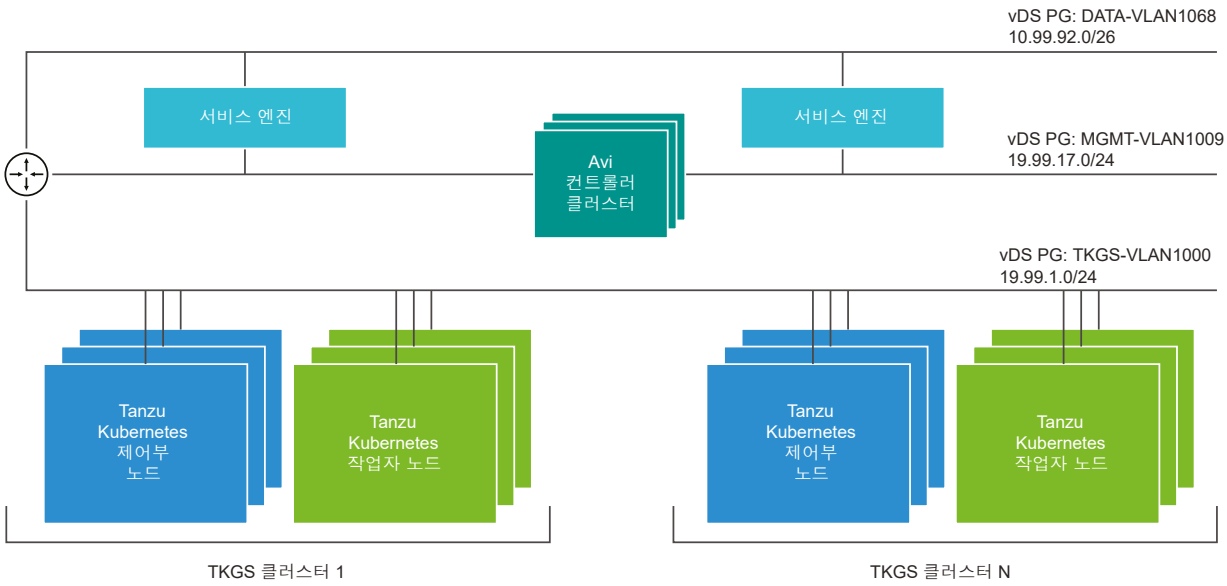
이 표에는 NSX Advanced Load Balancer, vCenter 및 기타 vSphere with Tanzu 구성 요소 간의 IP 연결을 관리하는 데 필요한 프로토콜 및 포트가 나열되어 있습니다.

소스	대상	프로토콜 및 포트
Avi 컨트롤러	Avi 컨트롤러(클러스터 내)	TCP 22(SSH) TCP 443(HTTPS) TCP 8443(HTTPS)
서비스 엔진	HA의 서비스 엔진	VMware, LSC 및 NSX-T 클라우드에 대해 TCP 9001
서비스 엔진	Avi 컨트롤러	TCP 22(SSH) TCP 8443(HTTPS) UDP 123(NTP)
Avi 컨트롤러	vCenter Server, ESXi, NSX-T Manager	TCP 443(HTTPS)
감독자 제어부 노드(AKO)	Avi 컨트롤러	TCP 443(HTTPS)

NSX Advanced Load Balancer의 포트 및 프로토콜에 대한 자세한 내용은 https://ports.esp.vmware.com/home/NSX-Advanced-Load-Balancer 항목을 참조하십시오.

vSphere 네트워킹 및 NSX Advanced Load Balancer를 사용하는 감독자 클러스터용 토폴로지

Avi 컨트롤러는 항상 vCenter Server, ESXi 호스트 및 감독자 클러스터 제어부 노드와 상호 작용할 수 있는 관리 네트워크에 배포됩니다. 서비스 엔진은 관리 네트워크 및 데이터 네트워크에 대한 인터페이스와 함께 배포됩니다.



관리 네트워크(예: MGMT-VLAN1009)는 컨트롤러가 상주하고 서비스 엔진의 관리 인터페이스가 연결되는 곳입니다.

데이터 네트워크(예: DATA-VLAN1068)는 VIP 배치를 위해 서비스 엔진 인터페이스가 연결되는 곳입니다. 클라이언트 트래픽은 VIP에 도달하고 서비스 엔진은 이 네트워크를 통해 트래픽을 워크로드 네트워크 IP로 로드 밸런싱합니다.

워크로드 네트워크(예: TKGS-VLAN1000)는 Tanzu Kubernetes 클러스터가 실행되는 곳입니다. 서비스 엔진에는 워크로드 네트워크에 대한 인터페이스가 필요하지 않습니다.

서비스 엔진은 단일 암 모드로 실행됩니다. 서비스 엔진은 로드 밸런싱된 트래픽을 라우터를 통해 워크로드 네트워크로 라우팅합니다. 서비스 엔진은 데이터 네트워크의 DHCP에서 기본 게이트웨이 IP를 가져오지 않습니다. 서비스 엔진이 트래픽을 워크로드 네트워크 및 클라이언트 IP로 올바르게 라우팅할 수 있도록 정적 경로를 구성해야 합니다. 정적 경로 구성에 대한 자세한 내용은 기본 게이트웨이 구성 항목을 참조하십시오.

이 토폴로지를 사용하면 서비스 엔진을 단일 네트워크에 배치하고 여러 워크로드 네트워크(있는 경우)에 로드 밸런싱 서비스를 제공할 수 있습니다. 서비스 엔진 생성 및 네트워크 연결은 Avi 컨트롤러에 의해 자동화됩니다.

NSX Advanced Load Balancer 설치 및 구성

VDS(vSphere Distributed Switch) 네트워킹을 사용하는 경우 vSphere with Tanzu 환경에 NSX Advanced Load Balancer 20.1.7을 설치하고 구성할 수 있습니다.

- 환경이 NSX Advanced Load Balancer로 vSphere with Tanzu를 구성하기 위한 요구 사항을 충족하는지 확인합니다. vSphere 네트워킹 및 NSX Advanced Load Balancer를 사용하여 vSphere with Tanzu를 설정하기 위한 시스템 요구 사항의 내용을 참조하십시오.
- NSX Advanced Load Balancer OVA를 다운로드합니다. VMware는 워크로드 관리를 사용하도록 설정할 vSphere 환경에 배포하는 NSX Advanced Load Balancer OVA 파일을 제공합니다. 제품 다운로드 페이지에서 vSphere with Tanzu에서 지원되는 최신 버전의 OVA 파일을 다운로드합니다.

NSX Advanced Load Balancer와 함께 사용할 감독자 클러스터용 vSphere Distributed Switch 생성

vSphere 클러스터를 vSphere 네트워킹 스택 및 NSX Advanced Load Balancer를 사용하는 감독자 클러스터로 구성하려면 vSphere Distributed Switch를 생성해야 합니다. 감독자 클러스터에 대한 워크로드 네트워크로 구성할 수 있는 Distributed Switch에서 포트 그룹을 생성합니다. Avi 서비스 엔진 데이터 인터페이스를 연결하려면 NSX Advanced Load Balancer에 분산 포트 그룹이 필요합니다. 포트 그룹은 서비스 엔진에 애플리케이션 VIP(가상 IP)를 배치하는 데 사용됩니다.

사전 요구 사항

NSX Advanced Load Balancer와 함께 감독자 클러스터에 vSphere 네트워킹을 사용하기 위한 시스템 요구 사항 및 네트워크 토폴로지를 검토합니다. vSphere 네트워킹 및 NSX Advanced Load Balancer를 사용하여 vSphere with Tanzu를 설정하기 위한 시스템 요구 사항의 내용을 참조하십시오.

절차

1 vSphere Client에서 데이터 센터로 이동합니다.

2 데이터 센터를 마우스 오른쪽 버튼으로 클릭하고 **Distributed Switch > 새 Distributed Switch**를 선택합니다.

3 스위치의 이름(예: **Workload Distributed Switch**)을 입력하고 **다음**을 클릭합니다.

4 스위치에 대해 버전 7.0을 선택하고 **다음**을 클릭합니다.

5 **포트 그룹 이름**에서 **Primary Workload Network**를 입력하고 **다음**을 클릭한 후 **마침**을 클릭합니다.

6 워크로드 네트워크에 대한 분산 포트 그룹을 생성합니다.

   a 새로 생성된 Distributed Switch로 이동합니다.

   b 스위치를 마우스 오른쪽 버튼으로 클릭하고 **분산 포트 그룹 > 새 분산 포트 그룹**을 선택합니다.

   c 포트 그룹의 이름(예: **Workload Network**)을 입력하고 **다음**을 클릭합니다.

   d 기본값을 그대로 두고 **다음**을 클릭한 다음 **마침**을 클릭합니다.

7 데이터 네트워크에 대한 포트 그룹을 생성합니다.

a 분산 스위치를 마우스 오른쪽 버튼으로 클릭하고 **분산 포트 그룹 > 새 분산 포트 그룹**을 선택합니다.

b 포트 그룹의 이름(예: **Data Network**)을 입력하고 **다음**을 클릭합니다.

c **설정 구성** 페이지에서 새 분산 포트 그룹에 대한 일반 속성을 입력하고 **다음**을 클릭합니다.

속성	설명
포트 바인딩	이 분산 포트 그룹에 연결된 가상 시스템에 포트가 할당되는 시점을 선택합니다. 가상 시스템이 분산 포트 그룹에 연결할 때 가상 시스템에 포트를 할당하려면 **정적 바인딩**을 선택합니다.
포트 할당	**탄력적** 포트 할당을 선택합니다. 기본 포트 수는 8개입니다. 포트가 모두 할당되면 새로 여덟 개의 포트가 생성됩니다.
포트 수	기본값을 유지합니다.
네트워크 리소스 풀	드롭다운 메뉴에서 사용자 정의 네트워크 리소스 풀에 새 분산 포트 그룹을 할당합니다. 네트워크 리소스 풀을 생성하지 않은 경우 이 메뉴는 비어 있습니다.
VLAN	드롭다운 메뉴에서 VLAN 트래픽 필터링 및 표시 유형을 선택합니다. ■ **없음**: VLAN을 사용하지 않습니다. External Switch Tagging을 사용하는 경우 이 옵션을 선택합니다. ■ **VLAN**: [VLAN ID] 텍스트 상자에 Virtual Switch Tagging에 대해 1~4094 사이의 값을 입력합니다. ■ **VLAN 트렁킹**: Virtual Guest Tagging에 대해 그리고 ID가 있는 VLAN 트래픽을 게스트 OS에 전달하려는 경우 이 옵션을 사용합니다. VLAN 트렁크 범위를 입력합니다. 쉼표로 구분된 목록을 사용하여 여러 개의 범위 또는 개별 VLAN을 설정할 수 있습니다. 예: 1702-1705, 1848-1849 ■ **전용 VLAN**: 트래픽을 Distributed Switch에서 생성된 전용 VLAN과 연결합니다. 전용 VLAN을 생성하지 않은 경우에 이 메뉴는 비어 있습니다.
고급	이 옵션은 선택되지 않은 상태로 둡니다.

8 **완료 준비** 페이지에서 구성을 검토하고 **마침**을 클릭합니다.

결과

Distributed Switch가 생성되고 Distributed Switch 아래에 분산 포트 그룹이 표시됩니다. 이제 생성한 포트 그룹을 NSX Advanced Load Balancer에 대한 데이터 네트워크로 사용할 수 있습니다.

컨트롤러 배포

vSphere with Tanzu 환경의 관리 네트워크에 컨트롤러 VM을 배포합니다.

사전 요구 사항

- NSX Advanced Load Balancer를 배포할 관리 네트워크가 있는지 확인합니다. vDS(vSphere Distributed Switch) 또는 vSS(vSphere Standard Switch)일 수 있습니다.

- 데이터 네트워크를 위한 vDS 스위치 및 포트 그룹을 생성했는지 확인합니다. NSX Advanced Load Balancer와 함께 사용할 감독자 클러스터용 vSphere Distributed Switch 생성의 내용을 참조하십시오.
- 사전 요구 사항을 완료했는지 확인합니다. vSphere 네트워킹 및 NSX Advanced Load Balancer를 사용하여 vSphere with Tanzu를 설정하기 위한 시스템 요구 사항의 내용을 참조하십시오.

절차

1 vSphere Client를 사용하여 vCenter Server에 로그인합니다.

2 관리 구성 요소용으로 지정된 vSphere 클러스터를 선택합니다.

3 **AVI-LB**라는 리소스 풀을 생성합니다.

4 리소스 풀을 마우스 오른쪽 버튼으로 클릭하고 **OVF 템플릿 배포**를 선택합니다.

5 **로컬 파일**을 선택하고 **파일 업로드**를 클릭합니다.

6 사전 요구 사항으로 다운로드한 `controller-VERSION.ova` 파일을 찾아서 선택합니다.

7 이름을 입력하고 컨트롤러의 폴더를 선택합니다.

옵션	설명
가상 시스템 이름	**avi-controller-1**
가상 시스템 위치	**데이터 센터**

8 **AVI-LB** 리소스 풀을 계산 리소스로 선택합니다.

9 구성 세부 정보를 검토하고 **다음**을 클릭합니다.

10 **VM 스토리지 정책**(예: **vsanDatastore**)을 선택합니다.

11 관리 네트워크(예: **MGMT-VLAN1009**)를 선택합니다.

12 다음과 같이 구성을 사용자 지정하고 완료되면 **다음**을 클릭합니다.

옵션	설명
관리 인터페이스 IP 주소	컨트롤러 VM의 IP 주소(예: **10.999.17.51**)를 입력합니다.
관리 인터페이스 서브넷 마스크	서브넷 마스크(예: **255.255.255.0**)를 입력합니다.
기본 게이트웨이	관리 네트워크의 기본 게이트웨이(예: **10.199.17.235**)를 입력합니다.
Sysadmin 로그인 인증 키	개인 키의 컨텐츠를 붙여넣습니다(선택 사항). SSH를 사용하여 VM에 연결하는 데 필요한 개인 SSH 키입니다. OpenSSH 또는 PuTTY를 사용하여 생성할 수 있습니다.

13 배포 설정을 검토합니다.

14 **마침**을 클릭하여 구성을 완료합니다.

15 vSphere Client를 사용하여 **작업** 패널에서 컨트롤러 VM의 프로비저닝을 모니터링합니다.

16 vSphere Client를 사용하여 컨트롤러 VM을 배포한 후 전원을 켭니다.

컨트롤러 전원 켜기

컨트롤러 VM을 배포한 후 전원을 켤 수 있습니다. 부팅 프로세스 동안 배포 중에 지정된 IP 주소가 VM에 할당됩니다.

전원을 켠 후 컨트롤러 VM의 첫 번째 부팅 프로세스에 최대 10분이 소요될 수 있습니다.

사전 요구 사항

컨트롤러를 배포합니다.

절차

1 vCenter Server에서, 배포한 `avi-controller-1` VM을 마우스 오른쪽 버튼으로 클릭합니다.

2 **전원 > 전원 켜기**를 선택합니다.

배포 중에 지정한 IP 주소가 VM에 할당됩니다.

3 VM의 전원이 켜져 있는지 확인하기 위해 브라우저에서 IP 주소에 액세스합니다.

VM이 온라인 상태가 되면 TLS 인증서 및 연결에 대한 주의가 표시됩니다.

4 **연결이 비공개가 아닙니다.** 주의에서 **세부 정보 표시**를 클릭합니다.

5 창이 나타나면 **이 웹 사이트 방문**을 클릭합니다.

사용자 자격 증명을 입력하라는 메시지가 표시됩니다.

컨트롤러 구성

사용 중인 vSphere with Tanzu 환경에 맞게 컨트롤러 VM을 구성합니다.

로드 밸런서 제어부를 vCenter Server 환경에 연결하려면 컨트롤러에 몇 가지 배포 후 구성 매개 변수가 필요합니다.

사전 요구 사항

- 사용 중인 환경이 NSX Advanced Load Balancer 구성을 위한 시스템 요구 사항을 충족하는지 확인합니다. vSphere 네트워킹 및 NSX Advanced Load Balancer를 사용하여 vSphere with Tanzu를 설정하기 위한 시스템 요구 사항의 내용을 참조하십시오.
- 컨트롤러를 배포합니다. 컨트롤러 배포 항목을 참조하십시오.

절차

1 브라우저를 사용하여 컨트롤러를 배포할 때 지정한 IP 주소로 이동합니다.

2 **관리자 계정**을 생성합니다.

옵션	설명
사용자 이름	초기 구성을 위한 관리자 사용자 이름입니다. 이 필드는 편집할 수 없습니다.
암호	컨트롤러 VM의 관리자 암호를 입력합니다. 암호는 8자 이상이어야 하며 숫자, 특수 문자, 대문자 및 소문자의 조합을 포함해야 합니다.
암호 확인	관리자 암호를 다시 입력합니다.
이메일 주소(선택 사항)	관리자 이메일 주소를 입력합니다. 운영 환경에서 암호 복구를 위한 이메일 주소를 제공하는 것이 좋습니다.

3 **시스템 설정**을 구성합니다.

옵션	설명
암호	컨트롤러 백업을 위한 암호를 입력합니다. 컨트롤러 구성은 정기적으로 로컬 디스크에 자동으로 백업됩니다. 자세한 내용은 백업 및 복원을 참조하십시오. 암호는 8자 이상이어야 하며 숫자, 특수 문자, 대문자 및 소문자의 조합을 포함해야 합니다.
암호 확인	백업 암호를 다시 입력합니다.
DNS 확인자	vSphere with Tanzu 환경에서 사용하는 DNS 서버의 IP를 입력합니다. 예를 들면 `10.14.7.12`입니다.
DNS 검색 도메인	도메인 문자열을 입력합니다.

4 (선택 사항) **이메일/SMTP**를 구성합니다.

옵션	설명
SMTP 소스	없음, 로컬 호스트, SMTP 서버 또는 익명 서버
보낸 사람 주소	이메일 주소

5 다중 테넌트 설정을 구성합니다.

a 기본 테넌트 액세스를 유지합니다.

b **다음 이후에 클라우드 설정**을 선택하고 **저장**을 클릭합니다.

> **참고** 저장하기 전에 **다음 이후에 클라우드 설정** 옵션을 선택하지 않은 경우 초기 구성 마법사가 종료됩니다. 클라우드 구성 창이 자동으로 실행되지 않고 컨트롤러의 대시보드 보기로 연결됩니다. 이 경우 **인프라 > 클라우드**로 이동하여 **기본 클라우드**를 편집하고 아래 단계를 계속 진행합니다.

6 **기본 클라우드**를 구성합니다.

a **클라우드**를 선택합니다.

b 인프라 유형으로 **VMware vCenter/vSphere ESX**를 선택합니다.

7 **인프라** 설정을 구성합니다.

vCenter/vSphere 로그인 정보를 제공합니다.

옵션	설명
사용자 이름	vCenter 관리자의 사용자 이름(예: administrator@vsphere.local)을 입력합니다. 더 적은 사용 권한을 사용하려면 전용 역할을 생성합니다. 자세한 내용은 VMware 사용자 역할을 참조하십시오.
암호	사용자 암호를 입력합니다.
vCenter 주소	vCenter Server 환경에 대한 vSphere with Tanzu 호스트 이름 또는 IP 주소를 입력합니다.
액세스 권한	**읽기**: 서비스 엔진 VM을 직접 생성하고 관리합니다. **쓰기**: 컨트롤러가 서비스 엔진 VM을 생성하고 관리합니다. [쓰기]를 선택해야 합니다.

IPAM 및 DNS 프로파일은 비워 둘 수 있습니다.

8 **데이터 센터** 설정을 구성합니다.

a **워크로드 관리**를 사용하도록 설정할 vSphere **데이터 센터**를 선택합니다.

b **기본 네트워크 IP 주소 관리** 모드를 선택합니다.

- vSphere 포트 그룹에서 DHCP를 사용할 수 있는 경우 **DHCP 사용**을 선택합니다.
- 서비스 엔진 인터페이스에서 고정 IP 주소만 사용하도록 하려면 옵션을 선택하지 않은 상태로 둡니다. 각 네트워크에 대해 개별적으로 구성할 수 있습니다.

  자세한 내용은 가상 IP 네트워크 구성의 내용을 참조하십시오.

c **가상 서비스 배치 설정**을 구성합니다.

옵션	설명
가상 서비스 배치를 위해 직접 연결된 네트워크보다 정적 경로 선호	서비스 엔진 VM이 기본 게이트웨이를 통해 라우팅하여 서버 네트워크에 강제로 액세스하도록 하려면 이 옵션을 선택합니다. 기본적으로 컨트롤러는 NIC를 서버 네트워크에 직접 연결하며, 서비스 엔진이 데이터 네트워크에만 연결하고 워크로드 네트워크에 라우팅되도록 강제로 적용해야 합니다.
가상 서비스 배치를 위해 VIP의 네트워크 확인에 정적 경로 사용	이 옵션은 선택되지 않은 상태로 둡니다.

9 **네트워크** 설정을 구성하고 **저장**을 클릭합니다.

옵션	설명
관리 네트워크	관리 네트워크를 선택합니다. 이 네트워크 인터페이스는 서비스 엔진이 컨트롤러와 연결하는 데 사용됩니다. 예를 들면 `Primary Workload Network`입니다.
서비스 엔진	**템플릿 서비스 엔진 그룹**을 비워 둡니다.
관리 네트워크 IP 주소 관리	**DHCP 사용**을 선택합니다.

10 (선택 사항) **DHCP 사용**을 선택하지 않은 경우에만 다음 네트워크 설정을 구성합니다.

옵션	설명
IP 서브넷	관리 네트워크의 IP 서브넷을 입력합니다. 예: `192.168.110.0/24` **참고** DHCP를 사용할 수 없는 경우에만 IP 서브넷을 입력합니다.
고정 IP 주소 풀 추가	하나 이상의 IP 주소 또는 IP 주소 범위를 입력합니다. 예: `192.168.110.66-192.168.110.90` **참고** DHCP를 사용할 수 없는 경우에만 IP 서브넷을 입력합니다.
기본 게이트웨이	관리 네트워크의 기본 게이트웨이(예: `192.168.110.1`)를 입력합니다. **참고** DHCP를 사용할 수 없는 경우에만 IP 서브넷을 입력합니다.

11 (선택 사항) 내부 NTP 서버를 사용하려면 NTP 설정을 구성합니다.

a **관리 > 설정 > DNS/NTP**를 선택합니다.

b 기존 NTP 서버가 있는 경우 삭제하고 사용 중인 DNS 서버의 IP 주소를 입력합니다. 예: `192.168.100.1`.

결과

구성을 완료하면 컨트롤러 **대시보드**가 보입니다. **인프라 > 클라우드**를 선택하고 **기본 클라우드**에 대한 컨트롤러의 상태가 녹색인지 확인합니다. 종종 Avi 컨트롤러가 vCenter 환경의 모든 포트 그룹을 검색할 때까지 상태가 잠시 노란색으로 표시되었다가 녹색으로 바뀔 수 있습니다.

라이센스 추가

NSX Advanced Load Balancer를 구성하고 나면 여기에 라이센스를 추가해야 합니다. 컨트롤러는 Enterprise 버전 라이센스에 해당하는 모든 기능을 사용할 수 있는 평가 모드로 부팅됩니다. 평가 기간이 만료되기 전에 유효한 Enterprise 라이센스를 컨트롤러에 할당해야 합니다.

사전 요구 사항

Enterprise 라이센스가 있는지 확인합니다.

절차

1 [Avi 컨트롤러] 대시보드에서 왼쪽 상단 모서리에 있는 메뉴를 클릭하고 **관리**를 선택합니다.

2 **설정 > 라이센싱**을 선택합니다.

3 라이센스를 추가하려면 **컴퓨터에서 업로드**를 선택합니다.

라이센스 파일이 업로드되면 컨트롤러 라이센스 목록에 파일이 나타납니다. 시작 날짜 및 만료 날짜를 포함하여 라이센스에 대한 정보가 표시됩니다.

4 (선택 사항) Enterprise 라이센스가 없는 경우 Essentials 버전을 사용할 수 있습니다.

참고 Enterprise 또는 평가 모드에서 Essentials 버전으로 전환하는 경우 Avi 컨트롤러를 구성하기 전에 전환해야 합니다. Enterprise 버전 기능을 이미 구성했다면 Essentials 버전으로 전환하기 전에 구성된 설정을 삭제해야 합니다.

a **설정 > 라이센싱**을 선택합니다.

b **라이센싱** 옆에 있는 기어 아이콘을 클릭합니다.

c **Essentials 라이센스**를 선택하고 **저장**을 클릭합니다.

d 팝업 창에서 **예**를 선택하여 버전을 확인합니다.

구성이 저장되고 Avi 컨트롤러가 Enterprise 기능을 비활성화하는 데 다소 시간이 걸릴 수 있습니다.

컨트롤러 클러스터 배포

필요한 경우 3개의 컨트롤러 노드로 구성된 클러스터를 배포할 수 있습니다. 운영 환경에서는 HA 및 재해 복구를 위해 클러스터를 구성하는 것이 좋습니다. 단일 노드 Avi 컨트롤러를 실행하는 경우 백업 및 복원 기능을 사용해야 합니다.

3노드 클러스터를 실행하려면 첫 번째 컨트롤러 VM을 배포한 후 두 개의 추가 컨트롤러 VM을 배포하고 전원을 켭니다. 초기 구성 마법사를 실행하거나 이러한 컨트롤러에 대한 관리자 암호를 변경하면 안 됩니다. 첫 번째 컨트롤러 VM의 구성이 두 개의 새 컨트롤러 VM에 할당됩니다.

절차

1 **관리 > 컨트롤러**로 이동합니다.

2 **노드**를 선택합니다.

3 편집 아이콘을 클릭합니다.

4 **컨트롤러 클러스터 IP**에 대한 고정 IP를 추가합니다.

이 IP 주소는 관리 네트워크의 IP 주소여야 합니다.

5 **클러스터 노드**에서 두 개의 새 클러스터 노드를 구성합니다.

옵션	설명
IP	컨트롤러 노드의 IP 주소입니다.
이름	노드의 이름입니다. 이 이름은 IP 주소일 수 있습니다.

옵션	설명
암호	컨트롤러 노드의 암호입니다. 암호는 비워 둡니다.
공용 IP	컨트롤러 노드의 공용 IP 주소입니다. 비워 둡니다.

6 **저장**을 클릭합니다.

참고 클러스터를 배포한 후에는 추가 구성에 컨트롤러 노드 IP가 아닌 컨트롤러 클러스터 IP를 사용해야 합니다.

컨트롤러에 인증서 할당

보안 통신을 설정하려면 컨트롤러가 클라이언트에 인증서를 보내야 합니다. 이 인증서에는 Avi 컨트롤러 클러스터 호스트 이름 또는 IP 주소와 일치하는 SAN(주체 대체 이름)이 있어야 합니다.

컨트롤러에는 기본 자체 서명된 인증서가 있습니다. 하지만 이 인증서에는 올바른 SAN이 없습니다. 이 인증서를 올바른 SAN이 있는 유효한 인증서 또는 자체 서명된 인증서로 교체해야 합니다. 자체 서명된 인증서를 생성하거나 외부 인증서를 업로드합니다.

인증서에 대한 자세한 내용은 Avi 설명서를 참조하십시오.

절차

1 Avi 컨트롤러 대시보드에서 왼쪽 상단 모서리에 있는 메뉴를 클릭하고 **템플릿 > 보안**을 선택합니다.

2 **SSL/TLS인증서**를 선택합니다.

3 인증서를 생성하려면 **생성**을 클릭하고 **컨트롤러 인증서**를 선택합니다.

새 인증서(SSL/TLS) 창이 나타납니다.

4 인증서의 이름을 입력합니다.

5 미리 생성된 유효한 인증서가 없는 경우 **유형**을 `Self Signed`로 선택하여 자체 서명된 인증서를 추가합니다.

a 다음과 같은 세부 정보를 입력합니다.

옵션	설명
일반 이름	사이트의 정규화된 이름을 지정합니다. 사이트가 신뢰할 수 있는 사이트로 간주되려면 이 항목이 클라이언트가 브라우저에 입력한 호스트 이름과 일치해야 합니다.
SAN(주체 대체 이름)	Avi 컨트롤러가 단일 노드로 배포된 경우 클러스터 IP 주소나 FQDN 또는 둘 다를 입력합니다. IP 주소 또는 FQDN만 사용되는 경우 배포 중에 지정한 컨트롤러 VM의 IP 주소와 일치해야 합니다. 컨트롤러 배포의 내용을 참조하십시오. Avi 컨트롤러 클러스터가 3개의 노드로 구성된 클러스터로 배포된 경우 클러스터 IP 또는 FQDN을 입력합니다. 3개의 컨트롤러 노드로 구성된 클러스터를 배포하는 방법에 대한 자세한 내용은 컨트롤러 클러스터 배포 항목을 참조하십시오.
알고리즘	EC(타원 곡선) 암호화 또는 RSA를 선택합니다. EC가 권장됩니다.
키 크기	핸드셰이크에 사용할 암호화 수준을 선택합니다. ■ `SECP256R1`은 EC 인증서에 사용됩니다. ■ 2048비트는 RSA 인증서에 권장됩니다.

b **저장**을 클릭합니다.

워크로드 관리 기능을 사용하도록 감독자 클러스터를 구성할 때 이 인증서가 필요합니다.

6 생성한 자체 서명된 인증서를 다운로드합니다.

a **보안 > SSL/TLS 인증서**를 선택합니다.

인증서가 표시되지 않으면 페이지를 새로 고칩니다.

b 생성한 인증서를 선택하고 다운로드 아이콘을 클릭합니다.

c **인증서 내보내기** 페이지가 나타나면 인증서에 대해 **클립보드에 복사**를 클릭합니다. 키를 복사하지 마십시오.

d 나중에 워크로드 관리를 사용하도록 설정할 때 사용할 수 있도록 복사한 인증서를 저장합니다.

7 미리 생성된 유효한 인증서가 있는 경우 **유형**을 `Import`로 선택하여 업로드합니다.

a **인증서**에서 **파일 업로드**를 클릭하고 인증서를 가져옵니다.

업로드한 인증서의 SAN 필드에는 컨트롤러의 클러스터 IP 주소 또는 FQDN이 있어야 합니다.

참고 인증서의 컨텐츠를 한 번만 업로드하거나 붙여넣어야 합니다.

b **키(PEM) 또는 PKCS12**에서 **파일 업로드**를 클릭하고 키를 가져옵니다.

c **유효성 검사**를 클릭하여 인증서와 키의 유효성을 검사합니다.

d **저장**을 클릭합니다.

8 포털 인증서를 변경하려면 다음 단계를 수행합니다.

a Avi 컨트롤러 대시보드에서 왼쪽 상단 모서리에 있는 메뉴를 클릭하고 **관리 > 설정**을 선택합니다.

b **액세스 설정**을 선택합니다.

c 편집 아이콘을 클릭합니다.

d **SSL/TLS 인증서**에서 기존 기본 포털 인증서를 제거합니다.

e 드롭다운에서 새로 생성 또는 업로드된 인증서를 선택합니다.

f **저장**을 클릭합니다.

서비스 엔진 그룹 구성

vSphere with Tanzu는 **기본 그룹** 서비스 엔진 그룹을 사용합니다. 필요한 경우 vCenter 내 서비스 엔진 VM의 수와 배치를 정의하는 그룹 내에서 **기본 그룹** 서비스 엔진을 구성할 수 있습니다. Avi 컨트롤러가 Enterprise 모드에 있는 경우에도 고가용성을 구성할 수 있습니다.

페일오버가 발생할 경우 초과 용량을 프로비저닝하는 방법에 대한 자세한 내용은 Avi 설명서를 참조하십시오.

절차

1 Avi 컨트롤러 대시보드에서 왼쪽 상단 모서리에 있는 메뉴를 클릭하고 먼저 **인프라**를 선택한 다음 **클라우드 리소스**를 선택합니다.

2 설정 페이지에서 **서비스 엔진 그룹**을 클릭합니다.

3 **서비스 엔진 그룹** 페이지에서 **기본 그룹**의 편집 아이콘을 클릭합니다.

기본 설정 페이지가 나타납니다.

4 **고가용성 및 배치 설정** 섹션에서 **고가용성 모드**를 선택합니다.

Essentials 라이센스의 기본 옵션은 `Active/Standby`입니다. Enterprise 라이센스를 사용하는 경우 `Elastic HA N + M Mode` 또는 `Elastic HA Active/Active Mode` 모드를 구성할 수도 있습니다.

5 **고급** 탭을 클릭합니다.

6 (선택 사항) **호스트 및 데이터스토어 범위** 섹션에서 다음 설정을 구성합니다.

a **포함**을 클릭하고 **클러스터**의 목록에서 vSphere 클러스터를 선택합니다.

b **포함**을 클릭하고 **호스트**의 목록에서 vSphere 클러스터를 선택합니다.

7 (선택 사항) **고급 HA 및 배치** 섹션에서는 Enterprise 라이센스를 사용하는 경우에만 서비스 엔진 그룹에 대해 초과 용량을 구성할 수 있습니다.

초과 용량을 구성하려면 **버퍼 서비스 엔진**에 값을 지정합니다. 지정하는 값은 페일오버가 발생할 경우 초과 용량을 보장하기 위해 배포되는 VM의 수입니다.

예를 들어 값을 0으로 설정합니다.

8 **저장**을 클릭합니다.

가상 IP 네트워크 구성

데이터 네트워크에 대한 VIP(가상 IP) 서브넷을 구성합니다. 가상 서비스가 특정 VIP 네트워크에 배치될 때 사용할 VIP 범위를 구성할 수 있습니다. 서비스 엔진에 대해 DHCP를 구성할 수 있습니다. 필요한 경우, DHCP를 사용할 수 없으면 해당 네트워크의 서비스 엔진 인터페이스에 할당될 IP 주소 풀을 구성할 수 있습니다.

절차

1 Avi 컨트롤러 대시보드에서 왼쪽 상단 모서리에 있는 메뉴를 클릭하고 **인프라**를 선택합니다.

2 **네트워크**를 클릭하여 vCenter Server의 네트워크 목록을 표시합니다.

3 가상 IP 주소를 제공하는 데이터 네트워크를 찾고 편집 아이콘을 클릭하여 네트워크 설정을 편집합니다.

예를 들면 `Data Nework`입니다.

4 데이터 네트워크에서 DHCP를 사용할 수 있는 경우 **DHCP 사용**을 선택한 상태로 유지합니다.

DHCP를 사용할 수 없는 경우 이 옵션을 선택 취소합니다.

5 **가상 서비스 배치를 위해 검색된 서브넷 제외**를 선택 취소합니다. 이 옵션을 선택 취소하면 가상 IP 주소 배치를 위해 구성된 서브넷을 사용할 수 있습니다.

Avi 컨트롤러는 VM이 네트워크에서 실행 중인 경우 네트워크 CIDR을 자동으로 검색하고 **검색됨** 유형으로 표시합니다.

6 Avi 컨트롤러가 IP 서브넷을 자동으로 검색하는 경우 서브넷의 IP 범위를 구성합니다.

a 검색된 네트워크의 편집 아이콘을 클릭합니다.

b **고정 IP 주소 풀 추가**를 선택합니다.

c 하나 이상의 IP 주소 또는 IP 주소 범위를 입력합니다.

예를 들면 `10.202.35.1-10.202.35.254`입니다.

> **참고** `0`으로 끝나는 IP 주소를 입력할 수 있습니다. 예를 들어 `192.168.0.0`를 입력하고 경고가 표시되면 무시합니다.

d 서비스 엔진 IP 주소에 대해 DHCP를 사용할 수 있는 경우 **VIP 및 SE에 고정 IP 주소 사용**을 선택 취소하고 **VIP에 사용**을 선택합니다.

e **저장**을 클릭합니다.

7 컨트롤러에서 IP 서브넷과 해당 유형이 검색되지 않으면 다음 단계를 수행합니다.

a **서브넷 추가**를 클릭합니다.

b **IP 서브넷**에서 가상 IP 주소를 제공하는 네트워크의 CIDR을 입력합니다.

예를 들어 `10.202.35.0/22`입니다.

c **고정 IP 주소 풀 추가**를 선택합니다.

d 하나 이상의 IP 주소 또는 IP 주소 범위를 입력합니다.

범위는 **IP 서브넷**에 있는 네트워크 CIDR의 하위 집합이어야 합니다. 예를 들면 `10.202.35.1-10.202.35.254`입니다.

참고 0으로 끝나는 IP 주소를 입력할 수 있습니다. 예를 들어 `192.168.0.0`를 입력하고 경고가 표시되면 무시합니다.

e 서비스 엔진 IP 주소에 대해 DHCP를 사용할 수 있는 경우 **VIP 및 SE에 고정 IP 주소 사용**을 선택 취소하고 **VIP에 사용**을 선택합니다.

f **저장**을 클릭하여 서브넷 구성을 저장합니다.

네트워크 설정 편집 페이지에 **구성됨** 유형의 IP 서브넷과 IP 주소 풀이 나열됩니다.

8 **저장**을 클릭하여 네트워크 설정을 저장합니다.

결과

네트워크 페이지에 구성된 네트워크가 나열됩니다.

예

`Primary Workload Network` 네트워크는 검색된 네트워크를 `10.202.32.0/22`로 표시하고 구성된 서브넷을 `10.202.32.0/22 [254/254]`로 표시합니다. 이것은 가상 IP 주소 254개가 `10.202.32.0/22`에서 할당된다는 것을 나타냅니다. 요약 보기에는 IP 범위 `10.202.35.1-10.202.35.254`가 나열되지 않습니다.

기본 게이트웨이 구성

기본 게이트웨이를 사용하면 서비스 엔진이 워크로드 네트워크의 풀 서버로 트래픽을 라우팅할 수 있습니다. 데이터 네트워크 게이트웨이 IP를 기본 게이트웨이로 구성해야 합니다.

절차

1 Avi 컨트롤러 대시보드에서 왼쪽 상단 모서리에 있는 메뉴를 클릭하고 **인프라**를 선택합니다.

2 **라우팅**을 클릭합니다.

3 **정적 경로** 섹션에서 **생성**을 클릭합니다.

4 **게이트웨이 서브넷**에 `0.0.0.0/0`을 입력합니다.

5 **다음 홉**에 데이터 네트워크의 게이트웨이 IP 주소를 입력합니다.

예: `192.168.0.1`

6 **저장**을 클릭합니다.

IPAM 구성

컨트롤러에 대한 IPAM을 구성하여 기본 클라우드 구성에 할당합니다. 현재는 기본 클라우드 구성만 지원됩니다.

가상 서비스가 생성될 때 가상 IP 주소를 할당하려면 IPAM이 필요합니다.

절차

1 Avi 컨트롤러 대시보드에서 **템플릿 > 프로파일 > IPAM/DNS 프로파일**로 이동합니다.

2 **생성**을 클릭하고 드롭다운 메뉴에서 **IPAM 프로파일**을 선택합니다.

3 **IPAM 프로파일**을 구성합니다.

옵션	설명
이름	사용자 정의 문자열(예: **ipam-profile**)
유형	**AVI Vantage IPAM** 선택
VRF에서 IP 할당	이 옵션을 선택 취소합니다.

4 **사용 가능한 네트워크 추가**를 클릭하고 구성합니다.

옵션	설명
사용 가능한 네트워크를 위한 클라우드	**기본 클라우드**
사용 가능한 네트워크	구성한 가상 IP 네트워크를 선택합니다.

5 **저장**을 클릭합니다.

IPAM/DNS 프로파일 페이지에 **ipam-profile**이 나열됩니다.

6 **기본 클라우드** 구성에 IPAM을 할당합니다.

a **인프라 > 클라우드**로 이동합니다.

b **기본 클라우드** 구성을 편집합니다.

IPAM 프로파일: **ipam-profile**

c 다른 모든 값은 기본값으로 둡니다.

d **저장**을 클릭합니다.

NSX Advanced Load Balancer 테스트

NSX Advanced Load Balancer 제어부를 배포하고 구성한 후 해당 기능을 검증합니다.

절차

1 Avi 컨트롤러 대시보드에서 **인프라 > 클라우드**로 이동합니다.

2 **기본 클라우드**에 대한 컨트롤러의 상태가 녹색인지 확인합니다.

문제가 발생할 경우 해결하려면 NSX Advanced Load Balancer 문제 해결 항목을 참조하십시오.

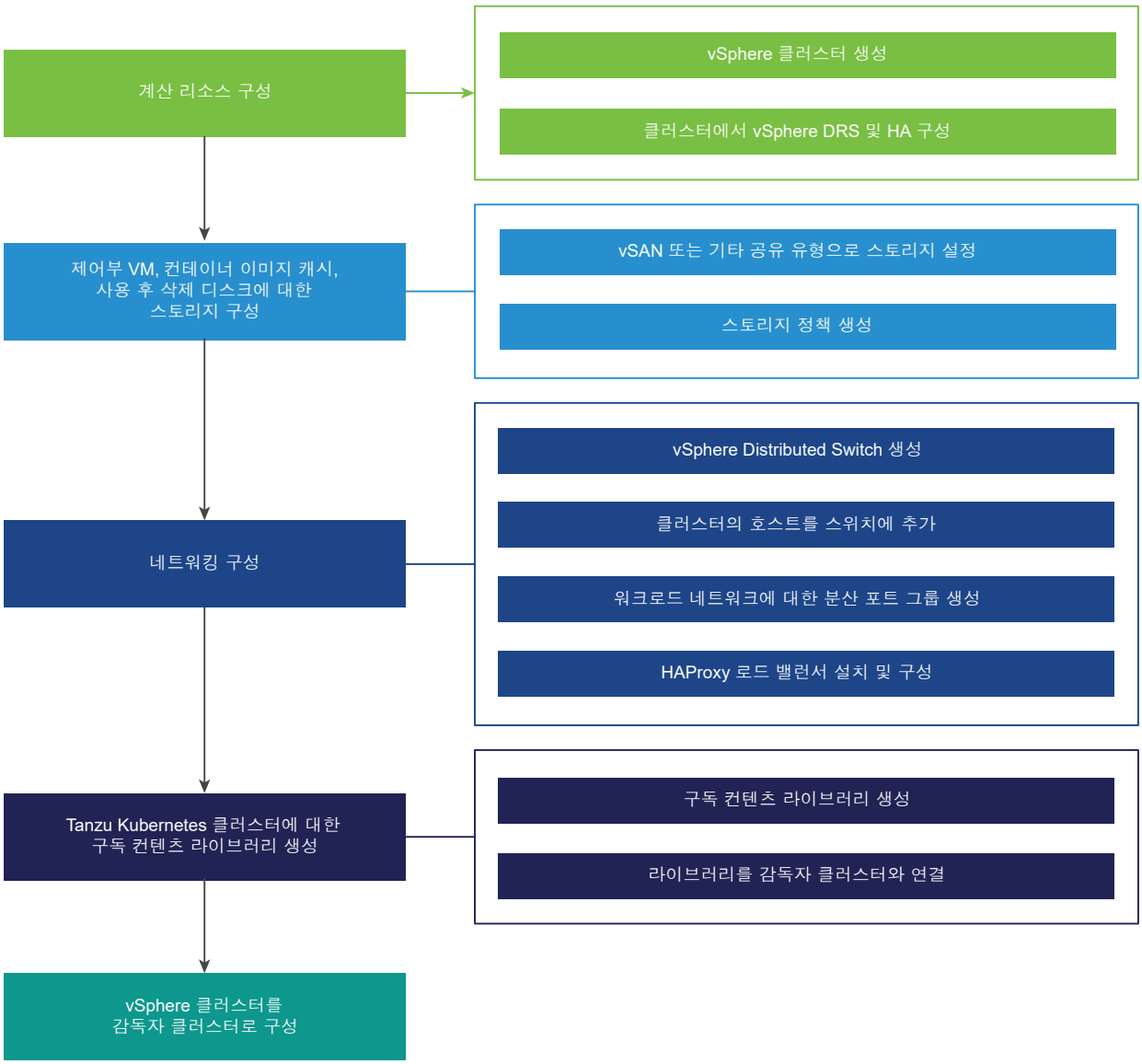
vSphere with Tanzu에 대한 vSphere 네트워킹 및 HAProxy 로드 밸런서 구성

vSphere with Tanzu 환경에서 vSphere Distributed Switch 네트워킹을 사용하는 경우 오픈 소스 HAProxy 로드 밸런서를 설치하고 구성할 수 있습니다. VMware는 OVA 파일을 통해 배포할 수 있는 HAProxy 구현을 제공합니다.

vSphere 네트워킹 및 HAProxy 로드 밸런서가 있는 감독자 클러스터 워크플로

이 다이어그램은 vSphere with Tanzu에 대한 vSphere 네트워킹 및 HAProxy 로드 밸런서를 구성하는 워크플로를 보여줍니다.

그림 4-6. HAProxy로 vDS 네트워킹을 구성하는 워크플로



vSphere 네트워킹 및 HAProxy 로드 밸런서를 사용하여 vSphere with Tanzu를 설정하기 위한 시스템 요구 사항

vSphere 네트워킹 스택 및 HAProxy 로드 밸런서를 사용하여 vSphere 클러스터를 감독자 클러스터로 설정하기 위한 시스템 요구 사항을 확인하십시오.

최소 계산 요구 사항

시스템	최소 배포 크기	CPU	메모리	스토리지
vCenter Server 7.0	작음	2	16GB	290GB
ESXi 호스트 7.0	vSAN 미포함: 호스트당 고정 IP가 1개인 ESXi 호스트 3개. vSAN 포함: 물리적 NIC가 2개 이상인 ESXi 호스트 4개. 호스트는 vSphere DRS 및 HA를 사용하도록 설정된 클러스터에 가입되어 있어야 합니다. vSphere DRS는 완전히 자동화되거나 부분적으로 자동화된 모드여야 합니다. **참고** 클러스터에 참여하는 호스트의 이름에 소문자를 사용하는지 확인합니다. 그렇지 않으면 워크로드 관리를 위한 클러스터 사용 설정이 실패할 수 있습니다.	8	호스트당 64GB	해당 없음
Kubernetes 제어부 VM	3	4	16GB	16GB

최소 네트워크 요구 사항

참고 vSphere 7 감독자 클러스터를 사용하여 IPv6 클러스터를 생성하거나 IPv6 클러스터를 Tanzu Mission Control에 등록할 수 없습니다.

구성 요소	최소 수량	필수 구성
Kubernetes 제어부 VM에 대한 고정 IP	5개의 블록	감독자 클러스터의 Kubernetes 제어부 VM에 할당할 5개의 연속적 고정 IP 주소 블록.
관리 트래픽 네트워크	1	ESXi 호스트, vCenter Server, 감독자 클러스터 및 로드 밸런서로 라우팅할 수 있는 관리 네트워크. 이미지 레지스트리가 외부 네트워크에 있는 경우 네트워크는 이미지 레지스트리에 액세스할 수 있어야 하고 인터넷에 연결되어 있어야 합니다. DNS를 통해 이미지 레지스트리를 확인할 수 있어야 합니다.
vSphere Distributed Switch	1	클러스터의 모든 호스트가 vSphere Distributed Switch에 연결되어 있어야 합니다.

구성 요소	최소 수량	필수 구성
HAProxy 로드 밸런서	1	vCenter Server 인스턴스로 구성된 HAProxy 로드 밸런서의 인스턴스입니다. ■ 동일한 HAProxy 인스턴스가 여러 감독자 클러스터를 제공하는 경우 모든 감독자 클러스터의 모든 워크로드 네트워크에서 들어오고 나가는 트래픽을 라우팅할 수 있어야 합니다. HAProxy가 제공하는 모든 감독자 클러스터의 워크로드 네트워크에서 IP 범위가 겹치지 않아야 합니다. ■ 가상 IP에 대한 전용 IP 범위입니다. HAProxy VM은 이 가상 IP 범위의 유일한 소유자여야 합니다. 범위는 임의의 감독자 클러스터가 소유한 워크로드 네트워크에 할당된 IP 범위와 겹치지 않아야 합니다. ■ HAProxy가 가상 IP를 할당하는 데 사용하는 네트워크는 HAProxy가 연결된 모든 감독자 클러스터에서 사용되는 워크로드 네트워크로 라우팅할 수 있어야 합니다.
워크로드 네트워크	1	기본 워크로드 네트워크로 구성한 vSphere Distributed Switch에 분산 포트 그룹을 최소 하나 생성해야 합니다. 선택한 토폴로지에 따라 네임스페이스의 워크로드 네트워크와 동일한 분산 포트 그룹을 사용하거나 더 많은 포트 그룹을 생성하고 이를 워크로드 네트워크로 구성할 수 있습니다. 워크로드 네트워크는 다음 요구 사항을 충족해야 합니다. ■ Tanzu Kubernetes 클러스터 트래픽에 사용되는 워크로드 네트워크는 서로 간에 그리고 감독자 클러스터 기본 워크로드 네트워크 간에 라우팅이 가능해야 합니다. ■ HAProxy가 가상 IP 할당에 사용하는 네트워크가 있는 모든 워크로드 네트워크 간에 라우팅이 가능해야 합니다. ■ 감독자 클러스터 내의 모든 워크로드 네트워크에서 IP 주소 범위가 겹치지 않아야 합니다. **중요** 워크로드 네트워크는 관리 네트워크와 다른 서브넷에 있어야 합니다.
NTP 및 DNS 서버	1	vCenter Server에서 사용할 수 있는 DNS 서버 및 NTP 서버. **참고** 모든 ESXi 호스트 및 vCenter Server에서 NTP를 구성합니다.

구성 요소	최소 수량	필수 구성
DHCP 서버	1	선택 사항입니다. 관리 및 워크로드 네트워크와 부동 IP에 대한 IP 주소를 자동으로 획득하도록 DHCP 서버를 구성합니다. DHCP 서버는 클라이언트 식별자를 지원하고 호환되는 DNS 서버, DNS 검색 도메인 및 NTP 서버를 제공해야 합니다. DHCP 구성은 감독자 클러스터에서 사용됩니다. 로드 밸런서에는 관리를 위한 고정 IP 주소가 필요할 수 있습니다. DHCP 범위는 이러한 고정 IP와 겹치지 않아야 합니다. DHCP는 가상 IP에 사용되지 않습니다. (VIP)
관리 네트워크 서브넷	1	ESXi 호스트, vCenter Server 및 Kubernetes 제어부 사이의 관리 트래픽에 사용되는 서브넷. 서브넷의 크기는 다음과 같아야 합니다. ■ 호스트 VMkernel 어댑터당 IP 주소 1개. ■ vCenter Server Appliance에 대해 IP 주소 1개. ■ Kubernetes 제어부에 대해 IP 주소 5개. 3개 노드 각각에 대해 1개, 가상 IP에 1개, 롤링 클러스터 업그레이드에 1개. **참고** 관리 네트워크와 워크로드 네트워크는 서로 다른 서브넷에 있어야 합니다. 관리 및 워크로드 네트워크에 동일한 서브넷을 할당하는 것은 지원되지 않으며 시스템 오류 및 문제가 발생할 수 있습니다.
관리 네트워크 VLAN	1	관리 네트워크 서브넷의 VLAN ID.
물리적 네트워크 MTU	1600	MTU 크기는 오버레이 트래픽을 전달하는 모든 네트워크에서 1600 이상이어야 합니다.
Kubernetes 서비스 CIDR 범위	/개인 IP 주소 16개	Kubernetes 서비스에 IP 주소를 할당할 개인 CIDR 범위. 각 감독자 클러스터에 대해 고유한 Kubernetes 서비스 CIDR 범위를 지정해야 합니다.

HAProxy 로드 밸런서 배포를 위한 토폴로지

vDS 네트워킹에 vSphere with Tanzu를 사용하는 경우 HAProxy는 Tanzu Kubernetes 제어부에 액세스하는 개발자를 위해 그리고 로드 밸런서 유형의 Kubernetes 서비스에 대해 로드 밸런싱을 제공합니다. HAProxy 로드 밸런서에 대해 구현할 수 있는 가능한 토폴로지를 검토합니다.

감독자 클러스터의 워크로드 네트워크

vSphere 네트워킹 스택을 사용하여 감독자 클러스터를 구성하려면 클러스터의 모든 호스트를 vSphere Distributed Switch에 연결해야 합니다. 감독자 클러스터 워크로드 네트워크에 대해 구현하는 토폴로지에 따라 하나 이상의 분산 포트 그룹을 생성합니다. 포트 그룹을 vSphere 네임스페이스에 대한 워크로드 네트워크로 지정합니다.

감독자 클러스터에 호스트를 추가하기 전에 클러스터에 포함된 모든 vSphere Distributed Switch에 호스트를 추가해야 합니다.

워크로드 네트워크는 Tanzu Kubernetes 클러스터 노드 및 감독자 클러스터 제어부 VM에 대한 연결을 제공합니다. Kubernetes 제어부 VM에 대한 연결을 제공하는 워크로드 네트워크를 기본 워크로드 네트워크라고 합니다. 각 감독자 클러스터에는 기본 워크로드 네트워크가 하나씩 있어야 합니다. 분산 포트 그룹 중 하나를 감독자 클러스터에 대한 기본 워크로드 네트워크로 지정해야 합니다.

참고 워크로드 네트워크는 감독자 클러스터를 사용하도록 설정할 때만 추가되며 나중에 추가할 수 없습니다.

감독자 클러스터의 Kubernetes 제어부 VM은 기본 워크로드 네트워크에 할당된 IP 주소 범위에서 3개의 IP 주소를 사용합니다. Tanzu Kubernetes 클러스터의 각 노드에는 Tanzu Kubernetes 클러스터가 실행되는 네임스페이스로 구성된 워크로드 네트워크의 주소 범위에서 할당된 별도의 IP 주소가 있습니다.

IP 범위 할당

HA Proxy 로드 밸런서가 있는 감독자 클러스터의 네트워킹 토폴로지를 계획하는 경우 두 가지 유형의 IP 범위를 갖도록 계획합니다.

- HAProxy에 대한 가상 IP 할당 범위입니다. HAProxy의 가상 서버에 대해 구성하는 IP 범위는 로드 밸런서 장치에 의해 예약됩니다. 예를 들어 가상 IP 범위가 `192.168.1.0/24`인 경우 가상 IP 트래픽 이외의 트래픽은 해당 범위의 모든 호스트에 액세스할 수 없습니다.

  참고 해당 게이트웨이에 대한 모든 경로가 실패하기 때문에 HAProxy 가상 IP 범위 내에 게이트웨이를 구성하지 않아야 합니다.

- 감독자 클러스터 및 Tanzu Kubernetes 클러스터의 노드에 대한 IP 범위입니다. 감독자 클러스터의 Kubernetes 제어부 VM에는 각각 하나의 IP 주소가 할당되어 총 3개의 IP 주소가 할당됩니다. 또한 Tanzu Kubernetes 클러스터의 각 노드에는 별도의 IP가 할당됩니다. 네임스페이스에 구성하는 감독자 클러스터의 각 워크로드 네트워크에 고유한 IP 범위를 할당해야 합니다.

하나의 /24 네트워크를 포함하는 구성 예:

- 네트워크: 192.168.120.0/24
- HAProxy VIP: 192.168.120.128/25
- HAProxy 워크로드 인터페이스에 대한 IP 주소 1개: 192.168.120.5

처음 128개 주소 내에서 사용 가능한 IP에 따라 감독자 클러스터의 워크로드 네트워크에 대한 IP 범위를 정의할 수 있습니다. 예:

- 기본 워크로드 네트워크용 192.168.120.31-192.168.120.40
- 다른 워크로드 네트워크용 192.168.120.51-192.168.120.60

참고 워크로드 네트워크에 대해 정의하는 범위는 HAProxy VIP 범위와 겹치지 않아야 합니다.

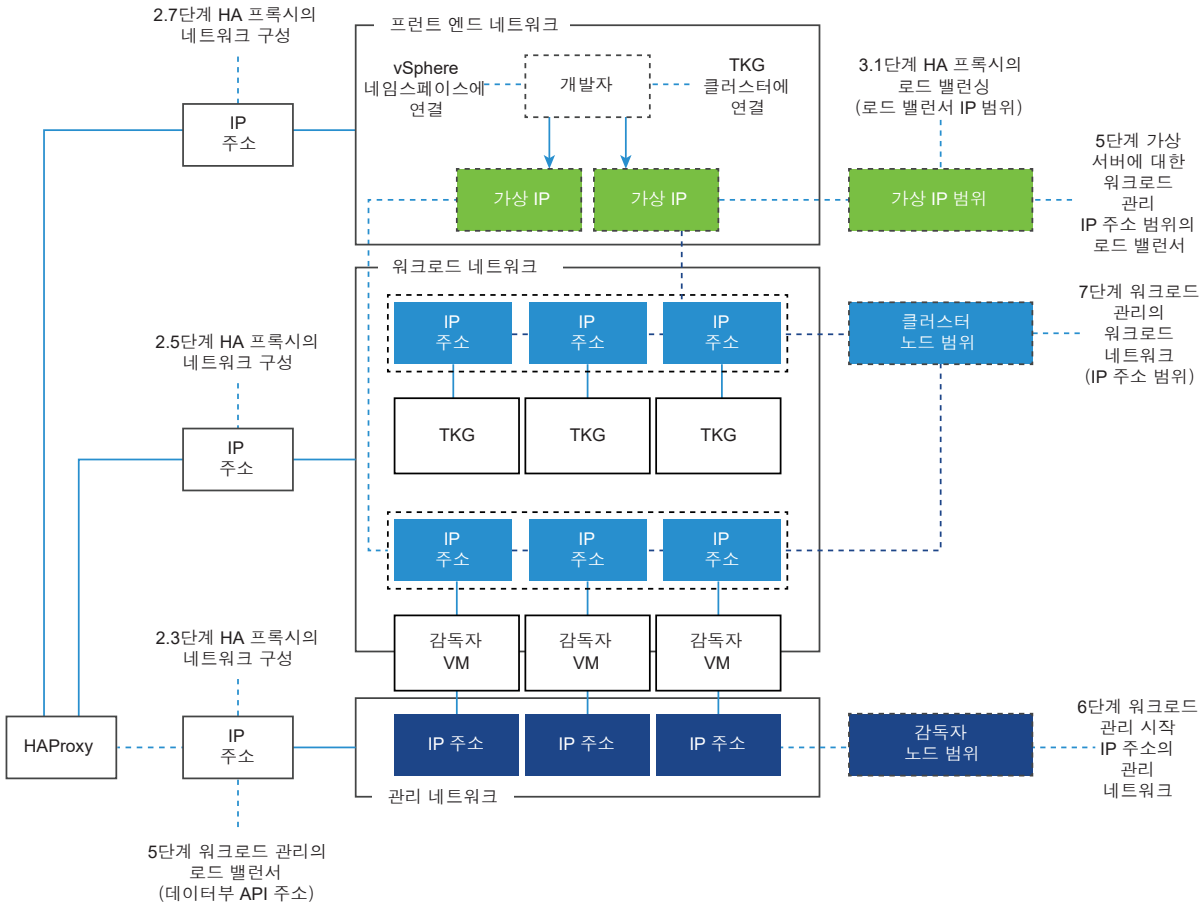
HAProxy 네트워크 토폴로지

HAProxy 배포를 위한 네트워크 구성 옵션에는 **기본** 및 **프런트 엔드**라는 두 가지가 있습니다. 기본 네트워크에는 2가지 NIC가 있으며, 하나는 관리 네트워크용이고, 다른 하나는 워크로드 네트워크용입니다. 프런트 엔드 네트워크에는 관리 네트워크, 워크로드 네트워크, 클라이언트용 프런트 엔드 네트워크라는 3가지 NIC가 있습니다. 각 네트워크의 특성은 표에 나열 및 설명되어 있습니다.

운영 설치의 경우 **프런트 엔드 네트워크** 구성을 사용하여 HAProxy 로드 밸런서를 배포하는 것이 좋습니다. **기본** 구성을 사용하여 HAProxy 로드 밸런서를 배포하는 경우에는 워크로드 네트워크에 /24 IP 주소 블록 크기를 할당하는 것이 좋습니다. 두 가지 구성 옵션 모두에서 DHCP는 권장되지 않습니다.

네트워크	특성
관리	감독자 클러스터는 관리 네트워크를 사용하여 HAProxy 로드 밸런서를 연결하고 프로그래밍합니다. ■ HAProxy 데이터부 API 끝점은 관리 네트워크에 연결된 네트워크 인터페이스에 바인딩됩니다. ■ HAProxy 제어부 VM에 할당된 관리 IP 주소는 관리 네트워크의 고정 IP여야 합니다. 그래야 감독자 클러스터가 로드 밸런서 API에 안정적으로 연결할 수 있습니다. ■ HAProxy VM의 기본 게이트웨이는 이 네트워크에 있어야 합니다. ■ 이 네트워크에서 DNS 쿼리가 발생해야 합니다.
워크로드	HAProxy 제어부 VM은 워크로드 네트워크를 사용하여 감독자 클러스터 및 Tanzu Kubernetes 클러스터 노드의 서비스에 액세스합니다. ■ HAProxy 제어부 VM은 이 네트워크에 있는 감독자 및 Tanzu Kubernetes 클러스터 노드로 트래픽을 전달합니다. ■ HAProxy 제어부 VM이 기본 모드(NIC 2개)로 배포된 경우 워크로드 네트워크는 로드 밸런서 서비스에 액세스하는 데 사용되는 논리적 네트워크를 제공해야 합니다. ■ **기본** 구성에서는 로드 밸런서 가상 IP와 Kubernetes 클러스터 노드 IP를 이 네트워크에서 가져옵니다. 네트워크 내에서 서로 겹치지 않는 별도의 범위로 정의됩니다. **참고** 워크로드 네트워크는 관리 네트워크와 다른 서브넷에 있어야 합니다. vSphere 네트워킹 및 HAProxy 로드 밸런서를 사용하여 vSphere with Tanzu를 설정하기 위한 시스템 요구 사항을 참조하십시오.
프런트 엔드(선택 사항)	클러스터 워크로드에 액세스하는 외부 클라이언트(예: 사용자 또는 애플리케이션)는 프런트 엔드 네트워크를 사용하여 가상 IP 주소를 사용하는 백엔드 로드 밸런싱된 서비스에 액세스합니다. ■ 프런트 엔드 네트워크는 HAProxy 제어부 VM이 3개의 NIC를 사용하여 배포된 경우에만 사용됩니다. ■ 운영 설치에 권장됩니다. ■ 프런트 엔드 네트워크는 VIP(가상 IP 주소)를 노출하는 위치입니다. HAProxy는 트래픽을 밸런싱하고 적절한 백엔드로 전달합니다.

아래 다이어그램은 **프런트 엔드 네트워크** 토폴로지를 사용하는 HAProxy 배포를 보여줍니다. 이 다이어그램은 설치 및 구성 프로세스 중에 구성 필드가 필요한 위치를 나타냅니다.



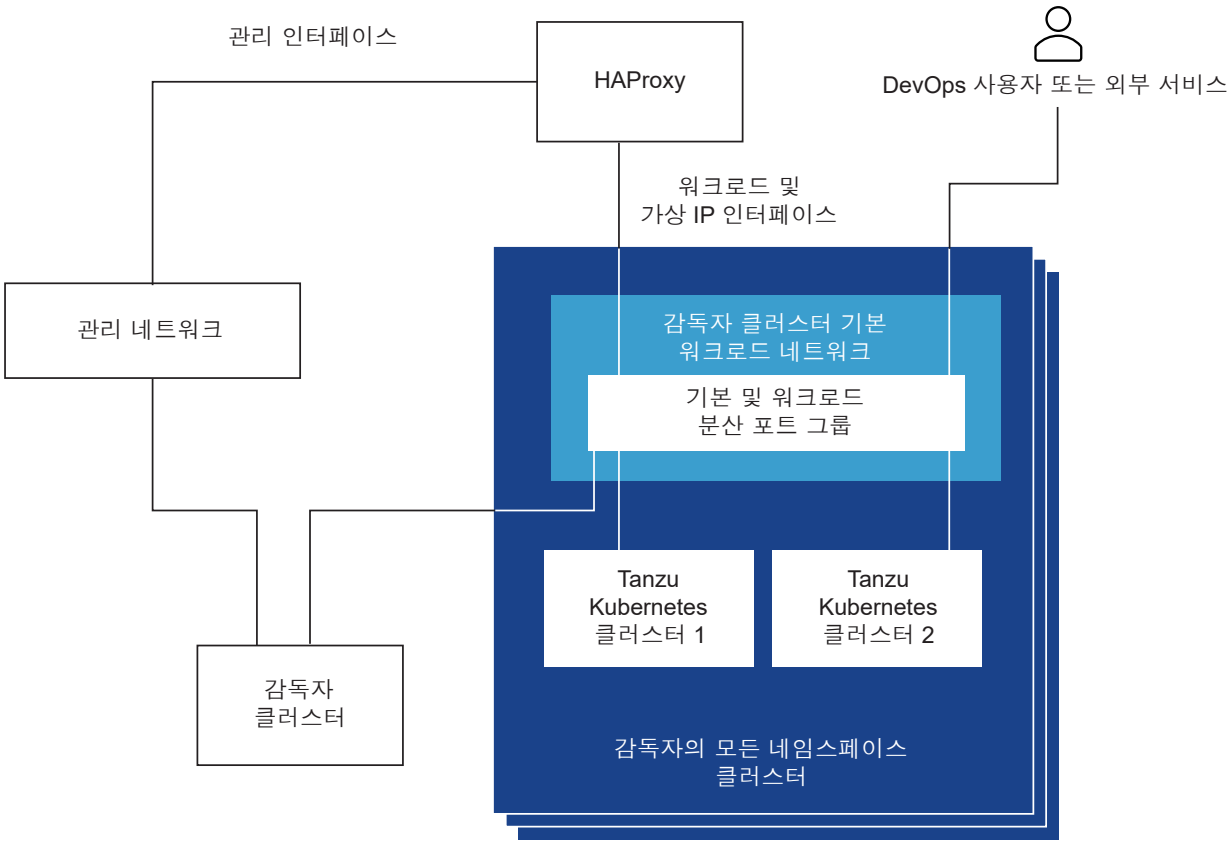
가상 NIC가 두 개인 HA 프록시 및 워크로드 네트워크가 한 개인 감독자 클러스터 토폴로지

이 토폴로지에서는 다음 구성 요소에 대해 워크로드 네트워크가 한 개인 감독자 클러스터를 구성합니다.

- Kubernetes 제어부 VM
- Tanzu Kubernetes 클러스터의 노드.
- 외부 서비스 및 DevOps 사용자가 연결되는 HAProxy 가상 IP 범위. 이 구성에서는 HAProxy가 가상 NIC 두 개(**기본** 구성)로 배포되며, 하나는 관리 네트워크에 연결되고 다른 하나는 기본 워크로드 네트워크에 연결됩니다. 기본 워크로드 네트워크와는 별도의 서브넷에 가상 IP를 할당하도록 계획해야 합니다.

하나의 포트 그룹을 감독자 클러스터에 대한 기본 워크로드 네트워크로 지정한 다음, 동일한 포트 그룹을 vSphere 네임스페이스용 워크로드 네트워크로 사용합니다. 감독자 클러스터, Tanzu Kubernetes 클러스터, HAProxy, DevOps 사용자, 외부 서비스는 모두 기본 워크로드 네트워크로 설정된 동일한 분산 포트 그룹에 연결됩니다.

그림 4-7. 하나의 네트워크에서 지원하는 감독자 클러스터



DevOps 사용자 또는 외부 애플리케이션에 대한 트래픽 경로는 다음과 같습니다.

1 DevOps 사용자 또는 외부 서비스는 분산 포트 그룹의 워크로드 네트워크 서브넷에 있는 가상 IP로 트래픽을 전송합니다.

2 HAProxy는 가상 IP 트래픽을 Tanzu Kubernetes 노드 IP 또는 제어부 VM IP로 로드 밸런싱합니다. HAProxy는 해당 IP에서 들어오는 트래픽을 로드 밸런싱할 수 있도록 가상 IP 주소를 할당합니다.

3 제어부 VM 또는 Tanzu Kubernetes 클러스터 노드는 각각 감독자 클러스터 또는 Tanzu Kubernetes 클러스터 내에서 실행되는 대상 포드로 트래픽을 전달합니다.

가상 NIC가 두 개인 HA 프록시 및 격리된 워크로드 네트워크가 있는 감독자 클러스터 토폴로지

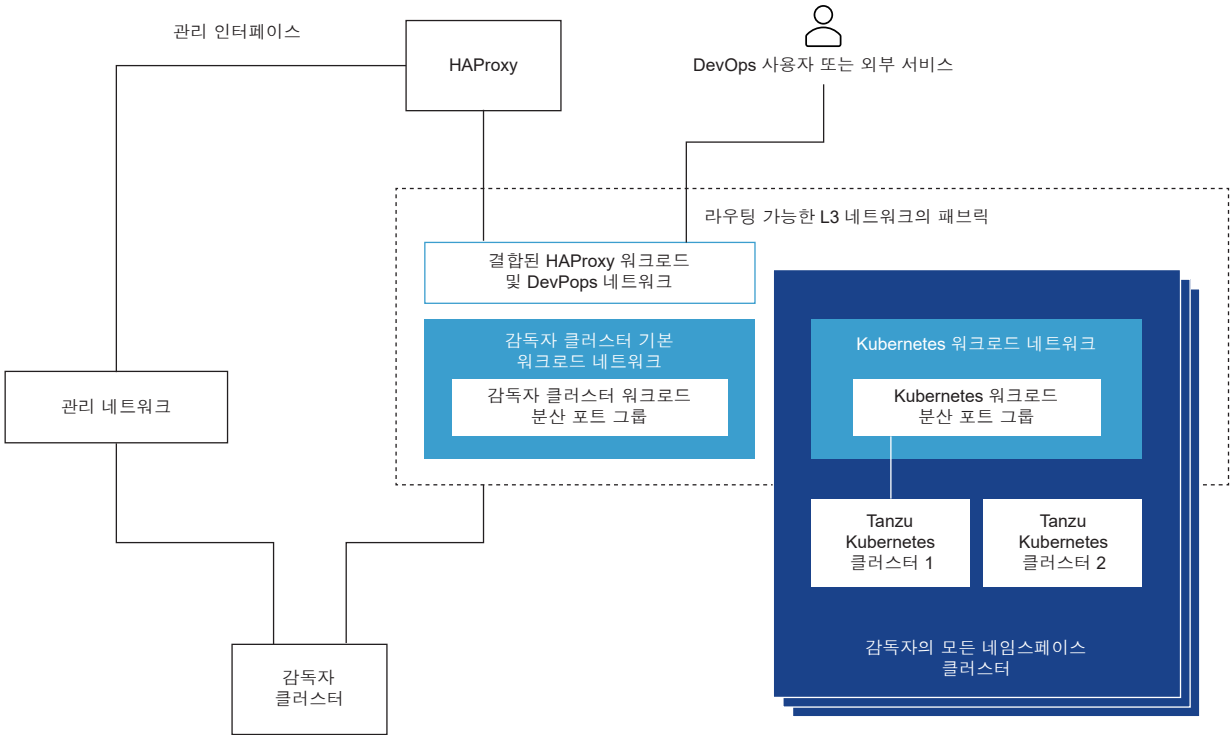
이 토폴로지에서는 다음 구성 요소에 대한 네트워크를 구성합니다.

- Kubernetes 제어부 VM. Kubernetes 제어부 VM에 대한 트래픽을 처리하기 위한 기본 워크로드 네트워크입니다.
- Tanzu Kubernetes 클러스터 노드. 사용자가 감독자 클러스터의 모든 네임스페이스에 할당하는 워크로드 네트워크. 이 네트워크는 Tanzu Kubernetes 클러스터 노드를 연결합니다.

- HAProxy 가상 IP. 이 구성에서는 HAProxy VM이 가상 NIC 두 개(**기본** 구성)로 배포됩니다. HAProxy VM을 기본 워크로드 네트워크 또는 네임스페이스에 사용하는 워크로드 네트워크에 연결할 수 있습니다. vSphere에 이미 있으며 기본 및 워크로드 네트워크에 라우팅할 수 있는 VM 네트워크에 HAProxy를 연결할 수도 있습니다.

감독자 클러스터는 기본 워크로드 네트워크를 지원하는 분산 포트 그룹에 연결되고, Tanzu Kubernetes 클러스터는 워크로드 네트워크를 지원하는 분산 포트 그룹에 연결됩니다. 두 포트 그룹은 계층 3 라우팅이 가능해야 합니다. VLAN을 통해 계층 2 분리를 구현할 수 있습니다. 계층 3 트래픽 필터링은 IP 방화벽과 게이트웨이를 통해 가능합니다.

그림 4-8. 하나의 분리된 워크로드 네트워크가 있는 감독자 클러스터



DevOps 사용자 또는 외부 서비스에 대한 트래픽 경로는 다음과 같습니다.

1 DevOps 사용자 또는 외부 서비스는 트래픽을 가상 IP로 전송합니다. 트래픽은 HAProxy가 연결된 네트워크로 라우팅됩니다.

2 HAProxy는 가상 IP 트래픽을 Tanzu Kubernetes 노드 IP 또는 제어부 VM으로 로드 밸런싱합니다. HAProxy는 해당 IP에서 들어오는 트래픽을 로드 밸런싱할 수 있도록 가상 IP 주소를 할당합니다.

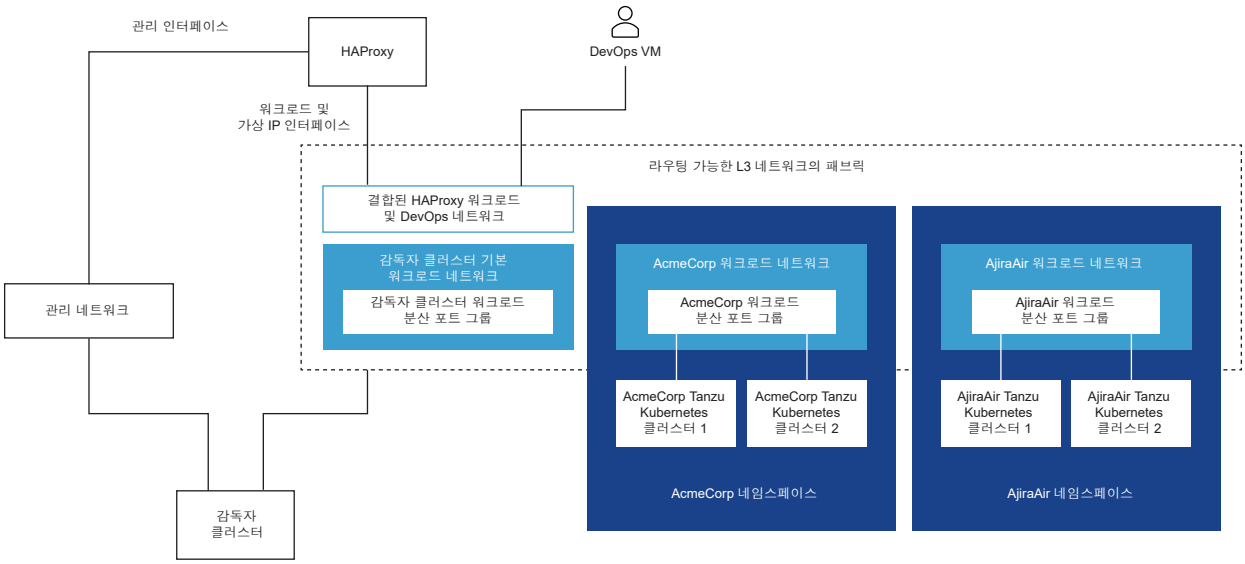
3 제어부 VM 또는 Tanzu Kubernetes 클러스터 노드는 Tanzu Kubernetes 클러스터 내에서 실행되는 대상 포드로 트래픽을 전달합니다.

가상 NIC가 두 개인 HA 프록시 및 다중 워크로드 네트워크가 있는 감독자 클러스터 토폴로지

이 토폴로지에서는 기본 워크로드 네트워크 역할을 하는 하나의 포트 그룹과 각 네임스페이스에 대한 워크로드 네트워크 역할을 하는 전용 포트 그룹을 구성할 수 있습니다. HAProxy는 가상 NIC 두 개를 사용하여 배포되며(**기본** 구성), 기본 워크로드 네트워크 또는 임의의 워크로드 네트워크에 연결할 수 있습니다. 기본 및 워크로드 네트워크로 라우팅할 수 있는 기존 VM 네트워크를 사용할 수도 있습니다.

이 토폴로지의 DevOps 사용자 및 외부 서비스에 대한 트래픽 경로는 분리된 워크로드 네트워크 토폴로지와 동일합니다.

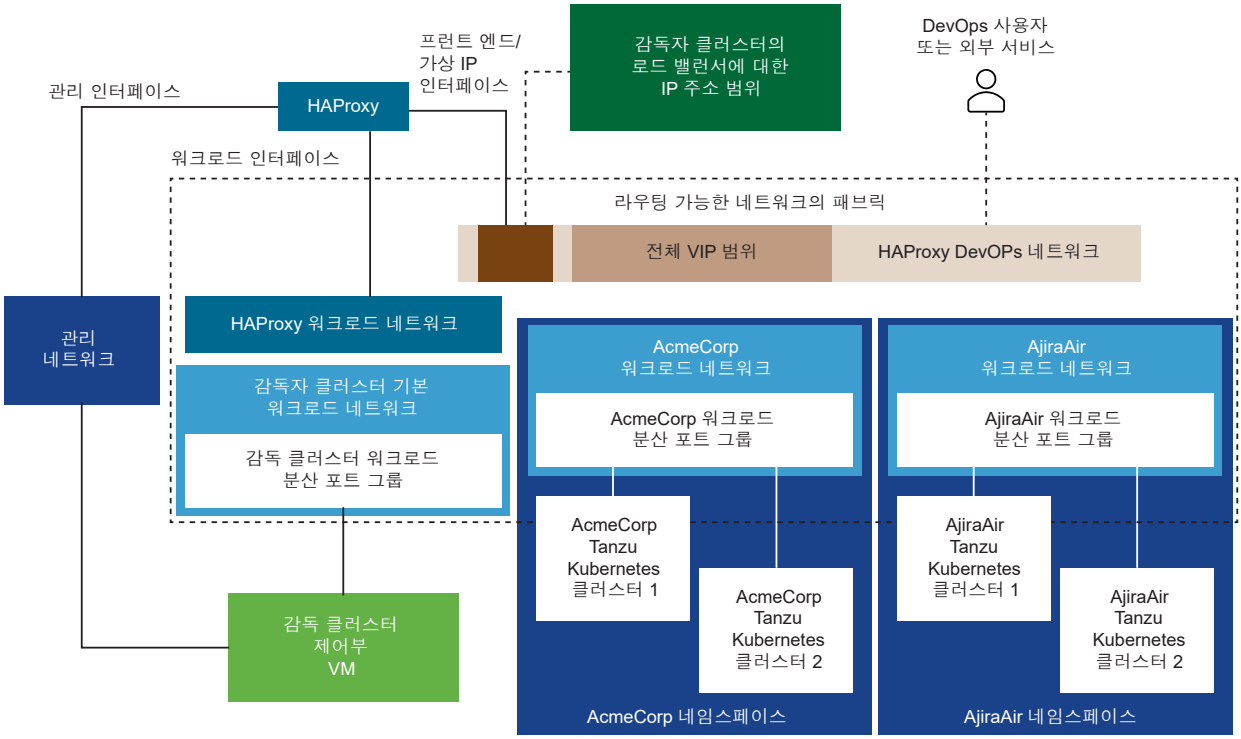
그림 4-9. 여러 개의 분리된 워크로드 네트워크에서 지원하는 감독자 클러스터



가상 NIC가 세 개인 HA 프록시 및 다중 워크로드 네트워크가 있는 감독자 클러스터 토폴로지

이 구성에서는 3개의 가상 NIC를 사용하여 HAProxy VM을 배포합니다. 따라서 HAProxy를 프런트 엔드 네트워크에 연결합니다. DevOps 사용자 및 외부 서비스는 프런트 엔드 네트워크의 가상 IP를 통해 HAProxy에 액세스할 수 있습니다. 운영 환경에는 가상 NIC가 세 개인 HA 프록시를 배포하는 것이 좋습니다.

그림 4-10. 3개의 가상 NIC를 사용하여 배포된 HAProxy



가능한 토폴로지 중에서 선택

가능한 각 토폴로지 중에서 선택하기 전에 해당 환경의 요구 사항을 평가합니다.

1 감독자 클러스터 및 Tanzu Kubernetes 클러스터 간에 계층 2 분리가 필요합니까?

   a 아니요: 모든 구성 요소를 처리하는 하나의 워크로드 네트워크를 사용하는 가장 간단한 토폴로지입니다.

   b 예: 별도의 기본 및 워크로드 네트워크를 사용하는 분리된 워크로드 네트워크 토폴로지입니다.

2 Tanzu Kubernetes 클러스터 간에 추가적인 계층 2 분리가 필요합니까?

   a 아니요: 별도의 기본 및 워크로드 네트워크를 사용하는 분리된 워크로드 네트워크 토폴로지입니다.

   b 예: 각 네임스페이스 및 전용 기본 워크로드 네트워크에 별도의 워크로드 네트워크를 사용하는 다중 워크로드 네트워크 토폴로지입니다.

3 DevOps 사용자 및 외부 서비스가 Kubernetes 제어부 VM 및 Tanzu Kubernetes 클러스터 노드로 직접 라우팅하지 못하도록 하시겠습니까?

   a 아니요: 2개 NIC의 HAProxy 구성입니다.

   b 예: 3개 NIC HAProxy 구성입니다. 이 구성은 운영 환경에 권장됩니다.

HAProxy 로드 밸런서와 함께 사용할 감독자 클러스터용 vSphere Distributed Switch 생성

vSphere 클러스터를 vSphere 네트워킹 스택 및 HA Proxy 로드 밸런서를 사용하는 감독자 클러스터로 구성하려면 vSphere Distributed Switch에 호스트를 추가해야 합니다. 감독자 클러스터에 대한 워크로드 네트워크로 구성할 Distributed Switch에서 포트 그룹을 생성해야 합니다.

클러스터에서 실행될 Kubernetes 워크로드에 제공하려는 분리 수준에 따라 감독자 클러스터에 대해 서로 다른 토폴로지를 선택할 수 있습니다.

사전 요구 사항

- HAProxy 로드 밸런서와 함께 감독자 클러스터에 vSphere 네트워킹을 사용하기 위한 시스템 요구 사항을 검토합니다. vSphere 네트워킹 및 HAProxy 로드 밸런서를 사용하여 vSphere with Tanzu를 설정하기 위한 시스템 요구 사항의 내용을 참조하십시오.
- 감독자 클러스터에서 HA Proxy를 사용하여 워크로드 네트워크를 설정하기 위한 토폴로지를 결정합니다. HAProxy 로드 밸런서 배포를 위한 토폴로지의 내용을 참조하십시오.

절차

1 vSphere Client에서 데이터 센터로 이동합니다.

2 데이터 센터를 마우스 오른쪽 버튼으로 클릭하고 **Distributed Switch > 새 Distributed Switch**를 선택합니다.

3 스위치의 이름(예: **`Workload Distributed Switch`**)을 입력하고 **다음**을 클릭합니다.

4 스위치에 대해 버전 7.0을 선택하고 **다음**을 클릭합니다.

5 **포트 그룹 이름**에서 **`Primary Workload Network`**를 입력하고 **다음**을 클릭한 후 **마침**을 클릭합니다.

   하나의 포트 그룹이 포함된 새 Distributed Switch가 데이터 센터에 생성됩니다. 이 포트 그룹을 생성할 감독자 클러스터에 대한 기본 워크로드 네트워크로 사용할 수 있습니다. 기본 워크로드 네트워크는 Kubernetes 제어부 VM에 대한 트래픽을 처리합니다.

6 워크로드 네트워크에 대한 분산 포트 그룹을 생성합니다.

   생성하는 포트 그룹의 수는 감독자 클러스터에 대해 구현하려는 토폴로지에 따라 다릅니다. 하나의 분리된 워크로드 네트워크가 있는 토폴로지의 경우 감독자 클러스터의 모든 네임스페이스에 대한 네트워크로 사용할 하나의 분산 포트 그룹을 생성합니다. 네임스페이스별로 분리된 네트워크가 있는 토폴로지의 경우 생성할 네임스페이스 수와 동일한 수의 포트 그룹을 생성합니다.

   a 새로 생성된 Distributed Switch로 이동합니다.

   b 스위치를 마우스 오른쪽 버튼으로 클릭하고 **분산 포트 그룹 > 새 분산 포트 그룹**을 선택합니다.

   c 포트 그룹의 이름(예: **`Workload Network`**)을 입력하고 **다음**을 클릭합니다.

   d 기본값을 그대로 두고 **다음**을 클릭한 다음 **마침**을 클릭합니다.

7 Distributed Switch에 감독자 클러스터로 구성할 vSphere 클러스터의 호스트를 추가합니다.

a Distributed Switch를 마우스 오른쪽 버튼으로 클릭하고 **호스트 추가 및 관리**를 선택합니다.

b **호스트 추가**를 선택합니다.

c **새 호스트**를 클릭하고 감독자 클러스터로 구성할 vSphere 클러스터에서 호스트를 선택한 후 **다음**을 클릭합니다.

d 각 호스트에서 물리적 NIC를 선택하고 Distributed Switch에서 여기에 업링크를 할당합니다.

e 마법사의 나머지 화면에서 **다음**을 클릭하고 **마침**을 클릭합니다.

결과

호스트가 Distributed Switch에 추가됩니다. 이제 스위치에서 생성한 포트 그룹을 감독자 클러스터의 워크로드 네트워크로 사용할 수 있습니다.

HAProxy 로드 밸런서 설치 및 구성

VMware는 vSphere with Tanzu 환경에서 사용할 수 있는 오픈 소스 HAProxy 로드 밸런서의 구현을 제공합니다. **워크로드 관리**에 vDS(vSphere Distributed Switch) 네트워킹을 사용하는 경우에는 HAProxy 로드 밸런서를 설치하고 구성할 있습니다.

HAProxy 로드 밸런서 제어부 VM 배포

Kubernetes 워크로드에 vSphere 네트워킹 스택을 사용하려면 HAProxy 제어부 VM을 설치하여 Tanzu Kubernetes 클러스터에 로드 밸런싱 서비스를 제공합니다.

사전 요구 사항

- 환경이 HA Proxy 배포를 위한 계산 및 네트워킹 요구 사항을 충족하는지 확인합니다. vSphere 네트워킹 및 HAProxy 로드 밸런서를 사용하여 vSphere with Tanzu를 설정하기 위한 시스템 요구 사항의 내용을 참조하십시오.
- HAProxy 로드 밸런서를 배포할 vSphere Standard 또는 Distributed Switch에 관리 네트워크가 있는지 확인합니다. 감독자 클러스터는 이 관리 네트워크의 HAProxy 로드 밸런서와 통신합니다.
- 워크로드 네트워크에 대한 vSphere Distributed Switch 및 포트 그룹을 생성합니다. HAProxy 로드 밸런서는 워크로드 네트워크를 통해 감독자 클러스터 및 Tanzu Kubernetes 클러스터 노드와 통신합니다. HAProxy 로드 밸런서와 함께 사용할 감독자 클러스터용 vSphere Distributed Switch 생성의 내용을 참조하십시오. 워크로드 네트워크에 대한 자세한 내용은 감독자 클러스터의 워크로드 네트워크 항목을 참조하십시오.
- VMware-HAProxy 사이트에서 최신 버전의 VMware HAProxy OVA 파일을 다운로드합니다.
- 감독자 클러스터에 HAProxy 로드 밸런서 및 워크로드 네트워크를 배포하기 위한 토폴로지를 선택합니다. HAProxy 로드 밸런서 배포를 위한 토폴로지 항목을 참조하십시오.

vDS 네트워킹 및 HAProxy를 통해 vSphere with Tanzu를 사용하는 방법에 대한 데모를 보는 것이 유용할 수 있습니다. vSphere with Tanzu 사용 시작 비디오를 확인하십시오.

절차

1 vSphere Client를 사용하여 vCenter Server에 로그인합니다.

2 HAProxy OVA 파일에서 새 VM을 생성합니다.

옵션	설명
컨텐츠 라이브러리	OVA를 로컬 컨텐츠 라이브러리에 가져온 경우: ■ **메뉴 > 컨텐츠 라이브러리**로 이동합니다. ■ OVA를 가져온 라이브러리를 선택합니다. ■ `vmware-haproxy-vX.X.X` 템플릿을 선택합니다. ■ 마우스 오른쪽 버튼을 클릭하고 **이 템플릿에서 새 VM 생성**을 선택합니다.
로컬 파일	OVA 파일을 로컬 호스트에 다운로드한 경우: ■ **워크로드 관리**를 사용하도록 설정할 vCenter 클러스터를 선택합니다. ■ 마우스 오른쪽 버튼을 클릭하고 **OVF 템플릿 배포**를 선택합니다. ■ **로컬 파일**을 선택하고 **파일 업로드**를 클릭합니다. ■ `vmware-haproxy-vX.X.X.ova` 파일을 찾아서 선택합니다.

3 **가상 시스템 이름**(예: **`haproxy`**)을 입력합니다.

4 HAProxy를 배포할 **데이터 센터**를 선택하고 **다음**을 클릭합니다.

5 **워크로드 관리**를 사용하도록 설정할 vCenter 클러스터를 선택하고 **다음**을 클릭합니다.

6 배포 세부 정보를 검토 및 확인하고 **다음**을 클릭합니다.

7 라이센스 계약에 동의하고 **다음**을 클릭합니다.

8 배포 구성을 선택합니다. 자세한 내용은 HAProxy 네트워크 토폴로지 항목을 참조하십시오.

구성	설명
기본값	NIC가 2개(관리 네트워크 및 단일 워크로드 네트워크)인 장치를 배포하려면 이 옵션을 선택합니다.
프런트 엔드 네트워크	NIC가 3개인 장치를 배포하려면 이 옵션을 선택합니다. 프런트 엔드 서브넷은 클러스터 노드를 개발자가 클러스터 제어부에 액세스하는 데 사용하는 네트워크에서 분리하는 데 사용됩니다.

9 VM에 사용할 스토리지 정책을 선택하고 **다음**을 클릭합니다.

10 로드 밸런서에 사용할 네트워크 인터페이스를 선택하고 **다음**을 클릭합니다.

소스 네트워크	대상 네트워크
관리	관리 네트워크(예: **VM 네트워크**)를 선택합니다.
워크로드	**워크로드 관리**를 위해 구성된 vDS 포트 그룹을 선택합니다.
프런트 엔드	프런트 엔드 서브넷에 대해 구성된 vDS 포트 그룹을 선택합니다. 프런트 엔드 구성을 선택하지 않으면 설치 중에 이 설정이 무시되므로 기본값을 그대로 둘 수 있습니다.

참고 워크로드 네트워크는 관리 네트워크와 다른 서브넷에 있어야 합니다. vSphere 네트워킹 및 HAProxy 로드 밸런서를 사용하여 vSphere with Tanzu를 설정하기 위한 시스템 요구 사항을 참조하십시오.

11 애플리케이션 구성 설정을 사용자 지정합니다. 장치 구성 설정의 내용을 참조하십시오.

12 네트워크 구성 세부 정보를 제공합니다. 네트워크 구성의 내용을 참조하십시오.

13 로드 밸런싱을 구성합니다. 로드 밸런싱 설정의 내용을 참조하십시오.

14 **다음**을 클릭하여 OVA 구성을 완료합니다.

15 배포 구성 세부 정보를 검토하고 **마침**을 클릭하여 OVA를 배포합니다.

16 **작업** 패널을 사용하여 VM 배포를 모니터링합니다.

17 VM 배포가 완료되면 전원을 켭니다.

다음에 수행할 작업

HAProxy 로드 밸런서가 성공적으로 배포되고 전원이 켜지면 **워크로드 관리**를 사용하도록 설정합니다. 장 5 감독자 클러스터 구성 및 관리의 내용을 참조하십시오.

HAProxy 로드 밸런서 사용자 지정

구성 설정, 네트워크 설정 및 로드 밸런싱 설정을 포함하여 HAProxy 제어부 VM을 사용자 지정합니다.

장치 구성 설정

이 표에는 HAProxy 장치 구성에 대한 매개 변수가 나열 및 설명되어 있습니다.

매개 변수	설명	주석 또는 예제
루트 암호	루트 사용자의 초기 암호입니다(6~128자).	이후 암호 변경은 운영 체제에서 수행해야 합니다.
루트 로그인 허용	루트 사용자가 SSH를 통해 원격으로 VM에 로그인할 수 있는 옵션입니다.	문제 해결을 위해 루트 로그인이 필요할 수도 있지만, 루트 로그인 허용 시 보안에 미치는 영향을 염두에 두어야 합니다.

매개 변수	설명	주석 또는 예제
TLS CA(인증 기관)(ca.crt)	자체 서명된 CA 인증서를 사용하려면 이 필드를 비워둡니다. 자체 CA 인증서(ca.crt)를 사용하려면 해당 컨텐츠를 이 필드에 붙여넣습니다. 컨텐츠를 Base64로 인코딩해야 할 수도 있습니다. https://www.base64encode.org/	자체 서명된 CA 인증서를 사용하는 경우 인증서에서 공용 및 개인 키가 생성됩니다.
키(ca.key)	자체 서명된 인증서를 사용하는 경우에는 이 필드를 비워둡니다. CA 인증서를 제공한 경우, 인증서 개인 키의 컨텐츠를 이 필드에 붙여넣습니다.	

네트워크 구성

이 표에는 HAProxy 네트워크 구성에 대한 매개 변수가 나열 및 설명되어 있습니다.

매개 변수	설명	주석 또는 예제
호스트 이름	HAProxy 제어부 VM에 할당할 호스트 이름(또는 FQDN)	기본값: `haproxy.local`
DNS	쉼표로 구분된 DNS 서버 IP 주소 목록입니다.	기본값: `1.1.1.1, 1.0.0.1` 예제 값: `10.8.8.8`
관리 IP	관리 네트워크에 있는 HAProxy 제어부 VM의 고정 IP 주소입니다.	네트워크의 접두사 길이가 있는 유효한 IPv4 주소(예: `192.168.0.2/24`)입니다.
관리 게이트웨이	관리 네트워크에 대한 게이트웨이의 IP 주소입니다.	예:`192.168.0.1`
워크로드 IP	워크로드 네트워크에 있는 HAProxy 제어부 VM의 고정 IP 주소입니다. 이 IP 주소는 로드 밸런서 IP 주소 범위 밖에 있어야 합니다.	네트워크의 접두사 길이가 있는 유효한 IPv4 주소(예: `192.168.10.2/24`)입니다.
워크로드 게이트웨이	워크로드 네트워크에 대한 게이트웨이의 IP 주소입니다.	예:`192.168.10.1` 프런트 엔드 구성을 선택하는 경우 게이트웨이를 입력해야 합니다. 프런트 엔드를 선택하고 게이트웨이를 지정하지 않으면 배포가 성공하지 못합니다.
프런트 엔드 IP	프런트 엔드 네트워크에 있는 HAProxy 장치의 고정 IP 주소입니다. 이 값은 프런트 엔드 배포 모델을 선택한 경우에만 사용됩니다.	네트워크의 접두사 길이가 있는 유효한 IPv4 주소(예: `192.168.100.2/24`)입니다.
프런트 엔드 게이트웨이	프런트 엔드 네트워크에 대한 게이트웨이의 IP 주소입니다. 이 값은 프런트 엔드 배포 모델을 선택한 경우에만 사용됩니다.	예:`192.168.100.1`

로드 밸런싱 설정

이 표에는 HAProxy 로드 밸런서 구성에 대한 매개 변수가 나열 및 설명되어 있습니다.

매개 변수	설명	예제 또는 주석
로드 밸런서 IP 범위	이 필드에는 CIDR 형식을 사용하여 IPv4 주소의 범위를 지정합니다. 값은 유효한 CIDR 범위여야 합니다. 그렇지 않으면 설치에 실패합니다. HAProxy는 VIP(가상 IP)에 대한 IP 주소를 예약합니다. 할당되면 각 VIP 주소가 할당되고 HAProxy는 해당 주소에 대한 요청에 응답합니다. 여기서 지정하는 CIDR 범위는 vSphere Client를 사용하여 vCenter Server에서 **워크로드 관리**를 사용하도록 설정할 때 가상 서버에 할당하는 IP와 겹치지 않아야 합니다.	예를 들어 네트워크 CIDR `192.168.100.0/24`는 로드 밸런서에 범위가 `192.168.100.0 - 192.168.100.255`인 256개의 가상 IP 주소를 제공합니다. 예를 들어 네트워크 CIDR `192.168.100.0/25`는 로드 밸런서에 범위가 `192.168.100.0 - 192.168.100.127`인 128개의 가상 IP 주소를 제공합니다.
Dataplane API 관리 포트	로드 밸런서의 API 서비스가 수신 대기하는 HAProxy VM의 포트입니다.	유효한 포트. 포트 22는 SSH용으로 예약되어 있습니다. 기본값은 `5556`입니다.
HAProxy 사용자 ID	로드 밸런서 API 사용자 이름	클라이언트가 로드 밸런서의 API 서비스에 인증하는 데 사용하는 사용자 이름입니다. **참고** 이 사용자 이름은 감독자 클러스터를 사용하도록 설정할 때 필요합니다.
HAProxy 암호	로드 밸런서 API 암호	클라이언트가 로드 밸런서의 API 서비스에 인증하는 데 사용하는 암호입니다. **참고** 이 암호는 감독자 클러스터를 사용하도록 설정할 때 필요합니다.

감독자 클러스터 구성 및 관리

5

vSphere 관리자는 감독자 클러스터를 생성하여 vSphere 클러스터를 워크로드 관리에 사용하도록 설정합니다. vSphere 네트워킹 스택 또는 NSX-T Data Center를 네트워킹 솔루션으로 사용하여 감독자 클러스터를 생성할 수 있습니다. NSX-T Data center로 구성된 클러스터는 VMware Tanzu™ Kubernetes Grid™ 서비스를 통해 생성된 vSphere 포드 및 Tanzu Kubernetes 클러스터 실행을 지원합니다. vSphere 네트워킹 스택으로 구성된 감독자 클러스터는 Tanzu Kubernetes 클러스터만 지원합니다.

감독자 클러스터를 사용하도록 설정한 후에는 vSphere Client를 사용하여 클러스터를 관리하고 모니터링할 수 있습니다.

본 장은 다음 항목을 포함합니다.

- vSphere 클러스터에서 vSphere with Tanzu를 구성하기 위한 사전 요구 사항
- vSphere 네트워킹으로 워크로드 관리 사용
- NSX-T Data Center 네트워킹으로 워크로드 관리 사용
- 감독자 클러스터에 Tanzu Edition 라이센스 할당
- VIP 인증서를 교체하여 감독자 클러스터 API 끝점에 안전하게 연결
- 감독자 클러스터의 Tanzu Kubernetes Grid 서비스를 Tanzu Mission Control과 통합
- Tanzu Kubernetes 클러스터에 대한 기본 CNI 설정
- VDS 네트워킹으로 구성된 감독자 클러스터에 워크로드 네트워크 추가
- 감독자 클러스터의 제어부 크기 변경
- 감독자 클러스터에서 관리 네트워크 설정 변경
- VDS 네트워킹으로 구성된 감독자 클러스터에서 워크로드 네트워크 설정 변경
- NSX-T Data Center로 구성된 감독자 클러스터에서 워크로드 네트워크 설정 변경
- 초기 구성 또는 업그레이드 중 감독자 클러스터의 오류 상태 해결
- vSphere with Tanzu에서 HTTP 프록시 설정 구성
- 감독자 클러스터 제어부의 로그를 원격 rsyslog로 스트리밍

vSphere 클러스터에서 vSphere with Tanzu를 구성하기 위한 사전 요구 사항

vSphere 환경에서 vSphere with Tanzu를 사용하도록 설정하기 위한 사전 요구 사항을 확인합니다. vSphere에서 기본적으로 컨테이너 기반 워크로드를 실행하려면 vSphere 관리자가 vSphere 클러스터에서 **워크로드 관리**를 사용하도록 설정합니다. 그 결과 vSphere 포드를 실행하고 Tanzu Kubernetes 클러스터 및 VM을 프로비저닝하는 감독자 클러스터라는 Kubernetes 관리 클러스터가 만들어집니다.

vSphere 클러스터 생성 및 구성

vSphere 클러스터는 vCenter Server 시스템에서 관리되는 ESXi 호스트의 모음입니다. 감독자 클러스터는 vSphere 클러스터에서 실행됩니다. 다음 요구 사항을 충족하는 vSphere 클러스터를 생성하여 **워크로드 관리**를 사용하도록 설정할 수 있게 합니다.

- ESXi 호스트가 3개 이상인 vSphere 클러스터를 생성하고 구성합니다. vSAN을 사용하는 경우 4개의 ESXi 호스트를 사용하는 것이 좋지만 필수는 아닙니다. 클러스터 생성 및 구성을 참조하십시오.
- vSAN과 같은 공유 스토리지를 사용하여 클러스터를 구성합니다. 공유 스토리지는 vSphere HA, DRS에 필요하며 영구 컨테이너 볼륨을 저장하는 데 필요합니다. vSAN 클러스터 생성을 참조하십시오.
- ReadWriteMany 모드에서 영구 볼륨을 사용하려는 경우 vSAN 클러스터에서 파일 서비스를 사용하도록 설정합니다. vSphere with Tanzu에서 ReadWriteMany 영구 볼륨 생성의 내용을 참조하십시오.
- vSphere HA를 사용하여 클러스터를 사용하도록 설정합니다. vSphere HA 클러스터 생성 및 사용을 참조하십시오.
- 완전 자동화 모드에서 vSphere DRS를 사용하여 클러스터를 사용하도록 설정합니다. DRS 클러스터 생성을 참조하십시오.
- **워크로드 관리** 기능을 사용하도록 설정할 수 있도록 사용자 계정에 vSphere 클러스터의 **클러스터 전체 구성 수정**이 있는지 확인합니다.

경고 감독자 클러스터를 구성한 후에는 vSphere DRS를 사용하지 않도록 설정하지 마십시오. 감독자 클러스터에서 워크로드를 실행하려면 항상 DRS를 사용하도록 설정해야 합니다. DRS를 사용하지 않도록 설정하면 Tanzu Kubernetes 클러스터가 중단됩니다.

네트워킹 스택 선택 및 구성

vSphere 클러스터에서 **워크로드 관리**를 사용하도록 설정하려면 감독자 클러스터에 사용할 네트워킹 스택을 구성해야 합니다. 사용 가능한 두 가지 옵션은 NSX-T Data Center 또는 로드 밸런서를 사용하는 vDS(vSphere Distributed Switch) 네트워킹입니다. NSX Advanced Load Balancer 또는 HAProxy 로드 밸런서를 구성할 수 있습니다.

이 표에는 지원되는 두 네트워킹 스택 간의 개략적인 차이점이 나열되어 있습니다. 아키텍처 차이에 대한 자세한 내용은 vSphere 네트워킹 스택으로 구성된 감독자 클러스터의 내용을 참조하십시오.

기능	NSX-T 네트워킹	vDS 네트워킹
vSphere 포드	예	아니요
Tanzu Kubernetes개 클러스터	예	예
내장된 Harbor 레지스트리	예	아니요
로드 밸런싱	예	예, NSX Advanced Load Balancer 또는 HAProxy 로드 밸런서를 설치하고 구성하여 수행합니다.

감독자 클러스터에 NSX-T Data Center 네트워킹을 사용하려면 다음을 수행합니다.

- NSX-T 네트워킹에 대한 시스템 요구 사항 및 토폴로지를 검토합니다. NSX-T Data Center를 사용하여 vSphere with Tanzu를 설정하기 위한 시스템 요구 사항의 내용을 참조하십시오.
- vSphere with Tanzu에 대한 NSX-T Data Center를 설치 및 구성합니다. vSphere with Tanzu에 대한 NSX-T Data Center 설치 및 구성의 내용을 참조하십시오.

감독자 클러스터에 대해 NSX Advanced Load Balancer와 vSphere vDS 네트워킹을 사용하려면 다음을 수행합니다.

- NSX Advanced Load Balancer 요구 사항을 검토합니다. vSphere 네트워킹 및 NSX Advanced Load Balancer를 사용하여 vSphere with Tanzu를 설정하기 위한 시스템 요구 사항의 내용을 참조하십시오.
- vDS(vSphere Distributed Switch)를 생성하고 클러스터의 모든 ESXi 호스트를 vDS에 추가하고 워크로드 네트워크용 포트 그룹을 생성합니다. NSX Advanced Load Balancer와 함께 사용할 감독자 클러스터용 vSphere Distributed Switch 생성의 내용을 참조하십시오.
- NSX Advanced Load Balancer를 배포하고 구성합니다. 컨트롤러 배포의 내용을 참조하십시오.

참고 vSphere with Tanzu는 vSphere 7 U2 이상에서 NSX Advanced Load Balancer를 지원합니다.

감독자 클러스터에 대해 HAProxy 로드 밸런싱과 vSphere vDS 네트워킹을 사용하려면 다음을 수행합니다.

- 외부 로드 밸런서를 사용하는 vSphere 네트워킹에 대한 시스템 요구 사항 및 네트워크 토폴로지를 검토합니다. vSphere 네트워킹 및 HAProxy 로드 밸런서를 사용하여 vSphere with Tanzu를 설정하기 위한 시스템 요구 사항 및 HAProxy 로드 밸런서 배포를 위한 토폴로지의 내용을 참조하십시오.
- vDS(vSphere Distributed Switch)를 생성하고 클러스터의 모든 ESXi 호스트를 vDS에 추가하고 워크로드 네트워크용 포트 그룹을 생성합니다. HAProxy 로드 밸런서와 함께 사용할 감독자 클러스터용 vSphere Distributed Switch 생성의 내용을 참조하십시오.
- vSphere 클러스터의 호스트에 연결된 vSphere Distributed Switch로 라우팅할 수 있는 HAProxy 로드 밸런서 인스턴스를 설치하고 구성합니다. HAProxy 로드 밸런서는 클라이언트 네트워크의 워크로드에 대한 네트워크 연결 및 Tanzu Kubernetes 클러스터 간에 트래픽을 로드 밸런싱을 위한 네트워크 연결을 지원합니다. HAProxy 로드 밸런서 설치 및 구성의 내용을 참조하십시오.

참고 vSphere with Tanzu는 vSphere 7 U1 이상에서 HAProxy 로드 밸런서를 지원합니다.

스토리지 정책 생성

Kubernetes 제어부 VM, 컨테이너 및 이미지의 데이터스토어 배치를 결정하는 스토리지 정책을 생성해야 합니다. 다른 스토리지 클래스와 연결된 스토리지 정책을 생성할 수 있습니다.

vSphere 클러스터에서 **워크로드 관리**를 사용하도록 설정하기 전에 Kubernetes 제어부 VM 배치에 대한 스토리지 정책을 생성합니다. vSphere with Tanzu에 대한 스토리지 정책 생성의 내용을 참조하십시오.

컨텐츠 라이브러리 생성

Tanzu Kubernetes 클러스터 및 VM을 프로비저닝하려면 감독자 클러스터가 실행되는 vSphere 클러스터를 관리하는 vCenter Server에 **컨텐츠 라이브러리**를 생성해야 합니다.

컨텐츠 라이브러리는 Tanzu Kubernetes 릴리스의 배포판을 OVA 템플릿 형태로 시스템에 제공합니다. Tanzu Kubernetes 클러스터를 프로비저닝하는 경우 선택한 버전에 대한 OVA 템플릿이 Kubernetes 클러스터 노드를 생성하는 데 사용됩니다.

구독 컨텐츠 라이브러리 생성하여 최신 릴리스 이미지를 자동으로 끌어오거나, **로컬 컨텐츠 라이브러리**를 생성하고 이미지를 수동으로 업로드할 수 있으며, Tanzu Kubernetes 클러스터의 에어갭 프로비저닝에 필요할 수 있습니다.

Tanzu Kubernetes 릴리스용 컨텐츠 라이브러리 생성 및 관리의 내용을 참조하십시오.

vSphere with Tanzu 데모 보기

어려운 요구 사항은 아니지만 시작하기 전에 감독자 클러스터 배포에 대비한 vSphere 환경 설정, **워크로드 관리**를 사용하도록 설정, Tanzu Kubernetes 클러스터 프로비저닝을 비롯한 몇 가지 vSphere with Tanzu 데모를 보는 것이 유용할 수 있습니다. 유용하다고 생각되면 VMware vSphere 채널에서 vSphere with Tanzu 심화 연구 비디오 시리즈를 확인하십시오. vDS 네트워킹 및 HAProxy 로드 밸런서를 사용하여 **워크로드 관리**를 구성하는 방법에 대한 vSphere Tanzu Quick Bytes 시리즈의 짧은 비디오를 참조할 수도 있습니다.

vSphere 네트워킹으로 워크로드 관리 사용

vSphere 관리자는 워크로드에 대한 연결을 제공하도록 vSphere 네트워킹 스택을 구성하여 vSphere 클러스터에서 **워크로드 관리** 플랫폼을 사용하도록 설정할 수 있습니다. vSphere 네트워킹으로 구성된 감독자 클러스터는 Tanzu Kubernetes Grid 서비스를 사용하여 생성된 Tanzu Kubernetes 클러스터의 배포를 지원합니다. vSphere 포드를 실행하거나 내장된 Harbor 레지스트리를 사용하는 것은 지원하지 않습니다.

경고 감독자 클러스터를 구성한 후에는 vSphere DRS를 사용하지 않도록 설정하지 마십시오. 감독자 클러스터에서 워크로드를 실행하려면 항상 DRS를 사용하도록 설정해야 합니다. DRS를 사용하지 않도록 설정하면 Tanzu Kubernetes 클러스터가 중단됩니다.

사전 요구 사항

- vSphere 클러스터를 감독자 클러스터로 구성하기 위한 사전 요구 사항을 완료합니다. vSphere 클러스터에서 vSphere with Tanzu를 구성하기 위한 사전 요구 사항의 내용을 참조하십시오.

절차

1 홈 메뉴에서 **워크로드 관리**를 선택합니다.

2 감독자 클러스터에 대한 라이센싱 옵션을 선택합니다.

- 유효한 Tanzu Edition 라이센스가 있는 경우 **라이센스 추가**를 클릭하여 vSphere 라이센스 인벤토리에 라이센스 키를 추가합니다.
- Tanzu Edition 라이센스가 아직 없는 경우에는 VMware에서 통신을 받을 수 있도록 연락처 세부 정보를 입력하고 **시작**을 클릭합니다.

감독자 클러스터의 평가 기간은 60일 동안 지속됩니다. 이 기간 내에는 클러스터에 유효한 Tanzu Edition라이센스를 할당해야 합니다. Tanzu Edition 라이센스 키를 추가한 경우 감독자 클러스터 설정을 완료하면 60일 평가 기간 내에 해당 키를 할당할 수 있습니다.

3 **워크로드 관리** 화면에서 **시작**을 다시 클릭합니다.

4 vCenter Server 시스템을 선택하고 **vCenter Server 네트워크**를 선택한 후 **다음**을 클릭합니다.

5 호환되는 클러스터 목록에서 클러스터를 선택합니다.

6 **제어부 크기** 페이지에서 클러스터의 각 호스트에 생성될 Kubernetes 제어부 VM에 대한 크기를 선택합니다.

제어부 VM에 할당하는 리소스의 양에 따라 감독자 클러스터가 관리할 수 있는 Kubernetes 워크로드 수가 결정됩니다.

7 **로드 밸런서** 화면에서 사용할 로드 밸런서를 선택합니다. NSX Advanced Load Balancer 또는 HAProxy를 선택할 수 있습니다.

- NSX Advanced Load Balancer에 대해 다음 설정을 입력합니다.

옵션	설명
이름	NSX Advanced Load Balancer의 이름을 입력합니다.
Avi 컨트롤러 IP	NSX Advanced Load Balancer 컨트롤러의 IP 주소입니다. 기본 포트는 443입니다.
사용자 이름	NSX Advanced Load Balancer을 사용하여 구성된 사용자 이름입니다. 이 사용자 이름을 사용하여 컨트롤러에 액세스합니다.
암호	사용자 이름에 대한 암호입니다.
서버 CA(인증 기관)	컨트롤러에 사용되는 인증서입니다. 구성 중에 할당한 인증서를 제공할 수 있습니다. 자세한 내용은 컨트롤러에 인증서 할당의 내용을 참조하십시오.

- HAProxy에 대한 다음 설정을 입력합니다.

옵션	설명
이름	로드 밸런서에 대해 사용자에게 친숙한 이름입니다.
데이터부 API 주소	HAProxy 데이터부 API의 IP 주소 및 포트입니다. 이 구성 요소는 HAProxy 서버를 제어하고 HAProxy VM 내에서 실행됩니다. HAProxy 장치의 관리 네트워크 IP 주소입니다.
사용자 이름	HAProxy OVA 파일을 사용하여 구성된 사용자 이름입니다. 이 이름을 사용하여 HAProxy 데이터부 API를 인증합니다.
암호	사용자 이름에 대한 암호입니다.

옵션	설명
가상 서버에 대한 IP 주소 범위	Tanzu Kubernetes 클러스터가 워크로드 네트워크에서 사용하는 IP 주소 범위입니다. 이 IP 범위는 HAProxy 장치 배포 중에 구성한 CIDR에 정의된 IP 목록에서 가져옵니다. 일반적으로 이 범위는 HAProxy 배포에 지정된 전체 범위이지만 해당 CIDR의 하위 집합일 수도 있습니다. 여러 감독자 클러스터를 생성하고 해당 CIDR 범위의 IP를 사용할 수 있기 때문입니다. 이 범위는 마법사에서 워크로드 네트워크에 대해 정의된 IP 범위와 겹치지 않아야 합니다. 또한 범위는 이 워크로드 네트워크의 DHCP 범위와 겹치지 않아야 합니다.

서버 CA(인증 기관)	데이터부 API가 제공하는 서버 인증서의 신뢰할 수 있는 루트이거나 서명된 PEM 형식의 인증서입니다.
	■ 옵션 1: 루트 액세스를 사용하도록 설정된 경우 SSH를 사용하여 HAProxy VM에 루트로 연결하고 `/etc/haproxy/ca.crt`를 **서버 CA(인증 기관)**에 복사합니다. `\n` 형식으로 이스케이프 줄을 사용하지 마십시오.
	■ 옵션 2: HAProxy VM을 마우스 오른쪽 버튼으로 클릭하고 **설정 편집**을 선택합니다. 해당 필드에서 CA 인증서를 복사한 후 https://www.base64decode.org/와 같은 변환 도구를 사용하여 Base64에서 변환합니다.
	■ 옵션 3: 다음 PowerCLI 스크립트를 실행합니다. `$vc`, `$vc_user`, `$vc_password` 변수를 적절한 값으로 바꿉니다.
	`$vc = "10.21.32.43"`
	`$vc_user = "administrator@vsphere.local"`
	`$vc_password = "PASSWORD"`
	`Connect-VIServer -User $vc_user -Password $vc_password -Server $vc`
	`$VMname = "haproxy-demo"`
	`$AdvancedSettingName = "guestinfo.dataplaneapi.cacert"`
	`$Base64cert = get-vm $VMname |Get-AdvancedSetting -Name $AdvancedSettingName`
	`while ([string]::IsNullOrEmpty($Base64cert.Value)) {`
	`Write-Host "Waiting for CA Cert Generation... This may take a under 5-10 minutes as the VM needs to boot and generate the CA Cert (if you haven't provided one already)."`
	`$Base64cert = get-vm $VMname | Get-AdvancedSetting -Name $AdvancedSettingName`
	`Start-sleep -seconds 2`
	`}`
	`Write-Host "CA Cert Found... Converting from BASE64"`
	`$cert =`

옵션	설명
	`[Text.Encoding]::Utf8.GetString([Convert]::FromBase64String($Base64cert.Value))` `Write-Host $cert`

8 **관리 네트워크** 화면에서 Kubernetes 제어부 VM에 사용될 네트워크에 대한 매개 변수를 구성합니다.

a **네트워크 모드**를 선택합니다.

- **DHCP 네트워크**. 이 모드에서는 관리 네트워크의 모든 IP 주소(예: 제어부 VM IP, 부동 IP, DNS 서버, DNS, 검색 도메인 및 NTP 서버)가 DHCP 서버에서 자동으로 획득됩니다. 부동 IP를 얻으려면 DHCP 서버는 클라이언트 식별자를 지원하도록 구성되어야 합니다. DHCP 모드에서 모든 제어부 VM은 안정적인 DHCP 클라이언트 식별자를 사용하여 IP 주소를 획득합니다. 이러한 클라이언트 식별자는 DHCP 서버에서 제어부 VM의 IP에 대한 고정 IP 할당을 설정하여 변경되지 않도록 하는 데 사용할 수 있습니다. 제어부 VM의 IP 및 부동 IP를 변경하는 것은 지원되지 않습니다.
- **정적**. 관리 네트워크에 대한 모든 네트워킹 설정을 수동으로 입력합니다.

b 관리 네트워크에 대한 설정을 구성합니다.

DHCP 네트워크 모드를 선택했지만 DHCP에서 획득한 설정을 재정의하려면 **추가 설정**을 클릭하고 새 값을 입력합니다. 정적 네트워크 모드를 선택한 경우에는 관리 네트워크 설정에 대한 값을 수동으로 입력합니다.

옵션	설명
네트워크	관리 트래픽에 대해 구성된 VMkernel 어댑터가 있는 네트워크를 선택합니다.
제어 IP 주소 시작	다음과 같이 Kubernetes 제어부 VM에 대해 5개의 연속 IP 주소를 예약하기 위한 시작 지점을 결정하는 IP 주소를 입력합니다. ■ Kubernetes 제어부 VM 각각에 대한 IP 주소입니다. ■ 관리 네트워크에 대한 인터페이스로 제공할 Kubernetes 제어부 VM 중 하나에 대한 부동 IP 주소입니다. 부동 IP 주소가 할당된 제어부 VM은 세 개의 Kubernetes 제어부 VM 모두에 대해 선행 VM으로 작동합니다. 부동 IP는 Kubernetes 클러스터의 etcd 리더인 제어부 노드로 이동합니다. 그러면 네트워크 파티션 이벤트의 경우 가용성이 향상됩니다. ■ Kubernetes 제어부 VM이 실패하여 새로운 제어부 VM으로 교체하는 경우 버퍼 역할을 하는 IP 주소입니다.
서브넷 마스크	고정 IP 구성에만 적용됩니다. 관리 네트워크에 대한 서브넷 마스크를 입력합니다. 예를 들어 `255.255.255.0`입니다.
DNS 서버	환경에서 사용하는 DNS 서버의 주소를 입력합니다. vCenter Server 시스템이 FQDN으로 등록되어 있으면 vSphere 환경에 사용하는 DNS 서버의 IP 주소를 입력해야 합니다. 그래야 감독자 클러스터에서 FQDN을 확인할 수 있습니다.
DNS 검색 도메인	Kubernetes 제어부 노드 내에서 DNS가 검색하는 도메인 이름(예: `corp.local`)을 입력합니다. 그래야 DNS 서버가 확인할 수 있습니다.
NTP	환경에서 사용하는 NTP 서버의 주소를 입력합니다(있는 경우).

9 **워크로드 네트워크** 페이지에서 감독자 클러스터에서 실행되는 Kubernetes 워크로드에 대한 네트워킹 트래픽을 처리하는 네트워크에 대한 설정을 입력합니다.

참고 워크로드 네트워크에 대한 네트워킹 설정을 제공하기 위해 DHCP 서버를 사용하도록 선택하는 경우 감독자 클러스터 구성을 완료하면 새 워크로드 네트워크를 생성할 수 없습니다.

a 네트워크 모드를 선택합니다.

- **DHCP 네트워크**. 이 네트워크 모드에서는 워크로드 네트워크에 대한 모든 네트워킹 설정이 DHCP를 통해 획득됩니다.
- **정적**. 워크로드 네트워크 설정을 수동으로 구성합니다.

b 감독자 클러스터에 대한 기본 워크로드 네트워크로 사용할 포트 그룹을 선택합니다.

기본 네트워크는 Kubernetes 제어부 VM 및 Kubernetes 워크로드 트래픽에 대한 트래픽을 처리합니다.

네트워킹 토폴로지에 따라, 나중에 각 네임스페이스에 네트워크로 사용할 다른 포트 그룹을 할당할 수 있습니다. 이 방법을 통해 감독자 클러스터의 네임스페이스 간에 계층 2 분리를 제공할 수 있습니다. 네트워크로 할당된 다른 포트 그룹이 없는 네임스페이스는 기본 네트워크를 사용합니다. Tanzu Kubernetes 클러스터는 배포된 네임스페이스에 할당된 네트워크만 사용하거나 해당 네임스페이스에 명시적으로 할당된 네트워크가 없는 경우 기본 네트워크를 사용합니다.

c 워크로드 네트워크에 대한 설정을 구성합니다.

DHCP 네트워크 모드를 선택한 경우 **추가 설정** 섹션 아래 모든 값이 DHCP 서버에서 자동으로 채워집니다. 이러한 값을 재정의하려면 **추가 설정**을 클릭하고 새 값을 입력합니다. **정적** 네트워크 모드를 선택한 경우에는 모든 설정을 수동으로 입력합니다.

옵션	설명
Kubernetes 서비스를 위한 내부 네트워크	클러스터 내에서 실행되는 Tanzu Kubernetes 클러스터 및 서비스에 대한 IP 주소 범위를 결정하는 CIDR 표기법을 입력합니다.
네트워크 이름	네트워크 이름을 입력합니다.
DNS 서버	환경에서 사용하는 DNS 서버의 IP 주소를 입력합니다(있는 경우). 예: `10.142.7.1`. DNS 서버의 IP 주소를 입력하면 각 제어부 VM에 정적 경로가 추가됩니다. 이것은 DNS 서버에 대한 트래픽이 워크로드 네트워크를 통과한다는 것을 나타냅니다. 지정하는 DNS 서버가 관리 네트워크와 워크로드 네트워크 간에 공유되는 경우에는 제어부 VM의 DNS 조회가 초기 설정 후 워크로드 네트워크를 통해 라우팅됩니다.
게이트웨이	기본 네트워크의 게이트웨이를 입력합니다.
서브넷 마스크 IP	서브넷 마스크 IP 주소를 입력합니다.
IP 주소 범위	Kubernetes 제어부 VM 및 워크로드의 IP 주소를 할당하기 위한 IP 범위를 입력합니다. 이 주소 범위는 감독자 클러스터 노드를 연결하며, 단일 워크로드 네트워크의 경우 Tanzu Kubernetes 클러스터 노드도 연결합니다. HAProxy에 대한 **기본** 구성을 사용하는 경우 이 IP 범위는 로드 밸런서 VIP 범위와 겹치지 않아야 합니다.

10 **스토리지** 페이지에서 스토리지 및 파일 볼륨 지원을 구성합니다.

a 감독자 클러스터에 대한 스토리지 정책을 선택합니다.

다음 각 개체에 대해 선택하는 스토리지 정책은 개체가 스토리지 정책에서 참조되는 데이터스토어에 배치되도록 합니다. 개체에 대해 동일한 또는 서로 다른 스토리지 정책을 사용할 수 있습니다.

옵션	설명
제어부 노드	제어부 VM 배치에 대한 스토리지 정책을 선택합니다.
포드 사용 후 삭제 디스크	vSphere 포드 배치에 대한 스토리지 정책을 선택합니다.
컨테이너 이미지 캐시	컨테이너 이미지의 캐시 배치에 대한 스토리지 정책을 선택합니다.

b (선택 사항) 파일 볼륨 지원을 활성화합니다.

이 옵션은 클러스터에 ReadWriteMany 영구 볼륨을 배포하려는 경우에 필요합니다. vSphere with Tanzu에서 ReadWriteMany 영구 볼륨 생성의 내용을 참조하십시오.

11 **Tanzu Kubernetes Grid** 페이지에서 **추가**를 클릭하고 Tanzu Kubernetes 클러스터의 노드를 배포하기 위한 VM 이미지가 포함된 구독 컨텐츠 라이브러리를 선택합니다.

12 설정을 검토하고 **마침**을 클릭합니다.

결과

감독자 클러스터를 생성하는 vCenter Server에서 작업이 실행됩니다. 작업이 완료되면 vSphere 클러스터에 속한 호스트에 3개의 Kubernetes 제어부 VM이 생성됩니다.

다음에 수행할 작업

감독자 클러스터에서 vSphere 네임스페이스를 생성하고 구성합니다. vSphere 네임스페이스 생성 및 구성의 내용을 참조하십시오.

NSX-T Data Center 네트워킹으로 워크로드 관리 사용

vSphere 관리자는 vSphere 클러스터를 NSX-T Data Center 네트워킹 스택을 사용하여 Kubernetes 워크로드에 대한 연결을 제공하는 감독자 클러스터로 구성할 수 있습니다.

사전 요구 사항

- 사용 중인 환경이 vSphere 클러스터를 감독자 클러스터로 구성하기 위한 사전 요구 사항을 충족하는지 확인합니다. 요구 사항에 대한 자세한 내용은 vSphere 클러스터에서 vSphere with Tanzu를 구성하기 위한 사전 요구 사항 항목을 참조하십시오.

경고 감독자 클러스터를 구성한 후에는 vSphere DRS를 사용하지 않도록 설정하지 마십시오. 감독자 클러스터에서 워크로드를 실행하려면 항상 DRS를 사용하도록 설정해야 합니다. DRS를 사용하지 않도록 설정하면 Tanzu Kubernetes 클러스터가 중단됩니다.

절차

1. vSphere Client 홈 메뉴에서 **워크로드 관리**를 선택합니다.
2. **시작**을 클릭합니다.
3. 구성하려는 vCenter Server 시스템을 선택합니다.
4. **NSX** 네트워킹 스택을 선택합니다.
5. **다음**을 클릭합니다.
6. **클러스터 선택 > 데이터 센터**를 선택합니다.
7. 호환되는 클러스터 목록에서 클러스터를 선택하고 **다음**을 클릭합니다.
8. **제어부 크기** 페이지에서 제어부 VM에 대한 크기 조정을 선택합니다.

   제어부 VM의 크기에 따라 감독자 클러스터에서 실행할 수 있는 워크로드의 양이 결정됩니다.

   지침을 보려면 VMware 구성 최대값 사이트를 참조하십시오.
9. **다음**을 클릭합니다.

10 **관리 네트워크** 화면에서 Kubernetes 제어부 VM에 사용될 네트워크에 대한 매개 변수를 구성합니다.

a **네트워크 모드**를 선택합니다.

- **DHCP 네트워크**. 이 모드에서는 관리 네트워크의 모든 IP 주소(예: 제어부 VM IP, DNS 서버, DNS, 검색 도메인 및 NTP 서버)가 DHCP에서 자동으로 획득됩니다.
- **정적**. 관리 네트워크에 대한 모든 네트워킹 설정을 수동으로 입력합니다.

b 관리 네트워크에 대한 설정을 구성합니다.

DHCP 네트워크 모드를 선택했지만 DHCP에서 획득한 설정을 재정의하려면 **추가 설정**을 클릭하고 새 값을 입력합니다. 정적 네트워크 모드를 선택한 경우에는 관리 네트워크 설정에 대한 값을 수동으로 입력합니다.

옵션	설명
네트워크	관리 트래픽에 대해 구성된 VMkernel 어댑터가 있는 네트워크를 선택합니다.
제어 IP 주소 시작	다음과 같이 Kubernetes 제어부 VM에 대해 5개의 연속 IP 주소를 예약하기 위한 시작 지점을 결정하는 IP 주소를 입력합니다. ■ Kubernetes 제어부 VM 각각에 대한 IP 주소입니다. ■ 관리 네트워크에 대한 인터페이스로 제공할 Kubernetes 제어부 VM 중 하나에 대한 부동 IP 주소입니다. 부동 IP 주소가 할당된 제어부 VM은 세 개의 Kubernetes 제어부 VM 모두에 대해 선행 VM으로 작동합니다. 부동 IP는 이 Kubernetes 클러스터(감독자 클러스터)의 etcd 리더인 제어부 노드로 이동합니다. 그러면 네트워크 파티션 이벤트의 경우 가용성이 향상됩니다. ■ Kubernetes 제어부 VM이 실패하여 새로운 제어부 VM으로 교체하는 경우 버퍼 역할을 하는 IP 주소입니다.
서브넷 마스크	고정 IP 구성에만 적용됩니다. 관리 네트워크에 대한 서브넷 마스크를 입력합니다. 예를 들어 `255.255.255.0`입니다.
DNS 서버	환경에서 사용하는 DNS 서버의 주소를 입력합니다. vCenter Server 시스템이 FQDN으로 등록되어 있으면 vSphere 환경에 사용하는 DNS 서버의 IP 주소를 입력해야 합니다. 그래야 감독자 클러스터에서 FQDN을 확인할 수 있습니다.
DNS 검색 도메인	Kubernetes 제어부 노드 내에서 DNS가 검색하는 도메인 이름(예: `corp.local`)을 입력합니다. 그래야 DNS 서버가 확인할 수 있습니다.
NTP	환경에서 사용하는 NTP 서버의 주소를 입력합니다(있는 경우).

11 **워크로드 네트워크** 창에서 네임스페이스의 네트워크에 대한 설정을 구성합니다.

네임스페이스 네트워크 설정은 감독자 클러스터에서 실행되는 네임스페이스 및 vSphere 포드에 대한 연결을 제공합니다. 기본적으로 네임스페이스는 클러스터 수준 네트워크 구성을 사용합니다.

옵션	설명
vSphere Distributed Switch	감독자 클러스터에 대한 오버레이 네트워킹을 처리하는 vSphere Distributed Switch를 선택합니다. 예를 들어 `DSwitch`를 선택합니다.
DNS 서버	환경에서 사용하는 DNS 서버의 IP 주소를 입력합니다(있는 경우). 예: `10.142.7.1`
API 서버 끝점 FQDN	필요한 경우 API 서버 끝점의 FQDN을 입력합니다.
Edge 클러스터	네임스페이스 네트워킹에 사용할 Tier-0 게이트웨이가 있는 NSX Edge 클러스터를 선택합니다. 예를 들어 `EDGE-CLUSTER`를 선택합니다.
Tier-0 게이트웨이	클러스터 Tier-1 게이트웨이와 연결할 Tier-0 게이트웨이를 선택합니다.
NAT 모드	NAT 모드는 기본적으로 선택되어 있습니다. 이 옵션을 선택 취소하면 vSphere 포드, VM 및 Tanzu Kubernetes 클러스터 노드 IP 주소와 같은 모든 워크로드를 Tier-0 게이트웨이 외부에서 직접 액세스할 수 있으며 송신 CIDR을 구성할 필요가 없습니다. **참고** NAT 모드를 선택 취소하면 파일 볼륨 스토리지가 지원되지 않습니다.
네임스페이스 네트워크	하나 이상의 IP CIDR을 입력하여 서브넷/세그먼트를 생성하고 워크로드에 IP 주소를 할당합니다.
네임스페이스 서브넷 접두사	네임스페이스 세그먼트용으로 예약된 서브넷의 크기를 지정하는 서브넷 접두사를 입력합니다. Default is 28.
포드 CIDR	vSphere 네이티브 포드의 IP 범위를 결정하는 CIDR 주석을 입력합니다. 기본값을 사용할 수 있습니다.
서비스 CIDR	Kubernetes 서비스의 IP 범위를 결정하는 CIDR 주석을 입력합니다. 기본값을 사용할 수 있습니다.
수신 CIDR	Kubernetes 서비스의 수신 IP 범위를 결정하는 CIDR 주석을 입력합니다. 이 범위는 로드 밸런서 및 수신 유형의 서비스에 사용됩니다.
송신 CIDR	Kubernetes 서비스의 송신 IP를 결정하는 CIDR 주석을 입력합니다. 감독자 클러스터의 각 네임스페이스에는 송신 IP 주소가 하나만 할당됩니다. 송신 IP는 특정 네임스페이스의 vSphere 포드가 NSX-T Data Center 외부에서 통신하는 데 사용하는 IP 주소입니다.

12 **다음**을 클릭합니다.

13 **스토리지** 페이지에서 스토리지 및 파일 볼륨 지원을 구성합니다.

a 감독자 클러스터에 대한 스토리지 정책을 선택합니다.

다음 각 개체에 대해 선택하는 스토리지 정책은 개체가 스토리지 정책에서 참조되는 데이터스토어에 배치되도록 합니다. 개체에 대해 동일한 또는 서로 다른 스토리지 정책을 사용할 수 있습니다.

옵션	설명
제어부 노드	제어부 VM 배치에 대한 스토리지 정책을 선택합니다.
포드 사용 후 삭제 디스크	vSphere 포드 배치에 대한 스토리지 정책을 선택합니다.
컨테이너 이미지 캐시	컨테이너 이미지의 캐시 배치에 대한 스토리지 정책을 선택합니다.

b (선택 사항) 파일 볼륨 지원을 활성화합니다.

이 옵션은 클러스터에 ReadWriteMany 영구 볼륨을 배포하려는 경우에 필요합니다. vSphere with Tanzu에서 ReadWriteMany 영구 볼륨 생성의 내용을 참조하십시오.

14 **완료 준비** 섹션에서 설정을 검토하고 **마침**을 클릭합니다.

클러스터에서 vSphere with Tanzu을 사용하도록 설정되며, vSphere 네임스페이스를 생성하여 DevOps 엔지니어에게 제공할 수 있습니다. 클러스터 및 Spherelet 프로세스의 일부인 호스트에 Kubernetes 제어부 노드가 생성됩니다.

다음에 수행할 작업

감독자 클러스터에서 vSphere 네임스페이스를 생성하고 구성합니다. vSphere 네임스페이스 생성 및 구성 항목을 참조하십시오.

감독자 클러스터에 Tanzu Edition 라이센스 할당

평가 모드에서 감독자 클러스터를 사용하는 경우에는 60일 평가 기간이 만료되기 전에 클러스터에 Tanzu Edition 라이센스를 할당해야 합니다.

Tanzu 라이센스의 작동 방식에 대한 자세한 내용은 vSphere with Tanzu에 대한 라이센싱 항목을 참조하십시오.

절차

1 vSphere Client에서 감독자 클러스터로 이동합니다.

2 **구성**을 선택하고 **라이센싱**에서 **감독자 클러스터**를 선택합니다.

3 **라이센스 할당**을 선택합니다.

4 **라이센스 할당** 대화 상자에서 **새 라이센스**를 클릭합니다.

5 유효한 라이센스 키를 입력하고 **확인**을 클릭합니다.

VIP 인증서를 교체하여 감독자 클러스터 API 끝점에 안전하게 연결

vSphere 관리자는 VIP(가상 IP 주소)의 인증서를 교체하여 호스트가 이미 신뢰하는 CA에서 서명한 인증서로 감독자 클러스터 API 끝점에 안전하게 연결할 수 있습니다. 이 인증서는 로그인하는 동안과 감독자 클러스터와의 후속 상호 작용 동안 DevOps 엔지니어에 대한 Kubernetes 제어부를 인증합니다.

사전 요구 사항

CSR에 서명할 수 있는 CA에 대한 액세스 권한이 있는지 확인합니다. DevOps 엔지니어의 경우 시스템에 CA를 신뢰할 수 있는 루트로 설치해야 합니다.

절차

1 vSphere Client에서 감독자 클러스터로 이동합니다.

2 **구성**을 클릭한 다음 **네임스페이스**에서 **인증서**를 선택합니다.

3 **워크로드 플랫폼 MTG** 창에서 **작업 > CSR 생성**을 선택합니다.

4 인증서에 대한 세부 정보를 제공합니다.

5 CSR이 생성되면 **복사**를 클릭합니다.

6 CA를 사용하여 인증서에 서명합니다.

7 **워크로드 플랫폼 MTG** 창에서 **작업 > 인증서 바꾸기**를 선택합니다.

8 서명된 인증서 파일을 업로드하고 **인증서 바꾸기**를 클릭합니다.

9 Kubernetes 제어부의 IP 주소의 인증서를 확인합니다.

   예를 들어 vSphere에 대한 Kubernetes CLI 도구 다운로드 페이지를 열고 브라우저를 사용하여 인증서가 성공적으로 바뀌었는지 확인할 수 있습니다. Linux 또는 Unix 시스템에서는 `echo | openssl s_client -connect https://ip:6443`을 사용할 수도 있습니다.

감독자 클러스터의 Tanzu Kubernetes Grid 서비스를 Tanzu Mission Control과 통합

감독자 클러스터에서 실행되는 Tanzu Kubernetes Grid 서비스를 Tanzu Mission Control과 통합할 수 있습니다. 이렇게 하면 Tanzu Mission Control을 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하고 관리할 수 있습니다.

Tanzu Mission Control에 대한 자세한 내용은 Tanzu Kubernetes 클러스터 수명 주기 관리를 참조하십시오. 데모를 보려면 Tanzu Kubernetes Grid 서비스와 통합된 Tanzu Mission Control 비디오를 참조하십시오.

감독자 클러스터에서 Tanzu Mission Control 네임스페이스 보기

vSphere with Tanzu v7.0.1 U1 이상에는 Tanzu Mission Control용 vSphere 네임스페이스가 제공됩니다. 이 네임스페이스는 Tanzu Mission Control 에이전트를 설치한 감독자 클러스터에 있습니다. 에이전트가 설치되면 Tanzu Mission Control 웹 인터페이스를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하고 관리할 수 있습니다.

1 kubectl용 vSphere 플러그인을 사용하여 감독자 클러스터로 인증합니다. vCenter Single Sign-On 사용자로 감독자 클러스터에 연결의 내용을 참조하십시오.

2 컨텍스트를 감독자 클러스터로 전환합니다. 예:

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

3 다음 명령을 실행하여 네임스페이스를 나열합니다.

```
kubectl get ns
```

4 Tanzu Mission Control에 제공되는 vSphere 네임스페이스는 `svc-tmc-cXX`(여기서 XX는 숫자)로 식별됩니다.

5 이 네임스페이스에 Tanzu Mission Control 에이전트를 설치합니다. 감독자 클러스터에 Tanzu Mission Control 에이전트 설치의 내용을 참조하십시오.

감독자 클러스터에 Tanzu Mission Control 에이전트 설치

Tanzu Kubernetes Grid 서비스를 Tanzu Mission Control과 통합하려면 감독자 클러스터에 에이전트를 설치합니다.

참고 다음 절차를 수행하려면 감독자 클러스터 버전 1.21.0 이상이 있는 vSphere 7.0 U3이 필요합니다.

1 Tanzu Mission Control 웹 인터페이스를 사용하여 Tanzu Mission Control에 감독자 클러스터를 등록합니다. Tanzu Mission Control에 관리 클러스터 등록을 참조하십시오.

2 Tanzu Mission Control 웹 인터페이스를 사용하여 **관리 > 관리 클러스터**로 이동하여 등록 URL을 가져옵니다.

3 Tanzu Mission Control에 필요한 포트(일반적으로 443)에 대해 vSphere with Tanzu 환경에서 방화벽 포트를 엽니다. 클러스터 에이전트 확장에 의한 아웃바운드 연결을 참조하십시오.

4 vSphere Client를 사용하여 vSphere with Tanzu 환경에 로그인합니다.

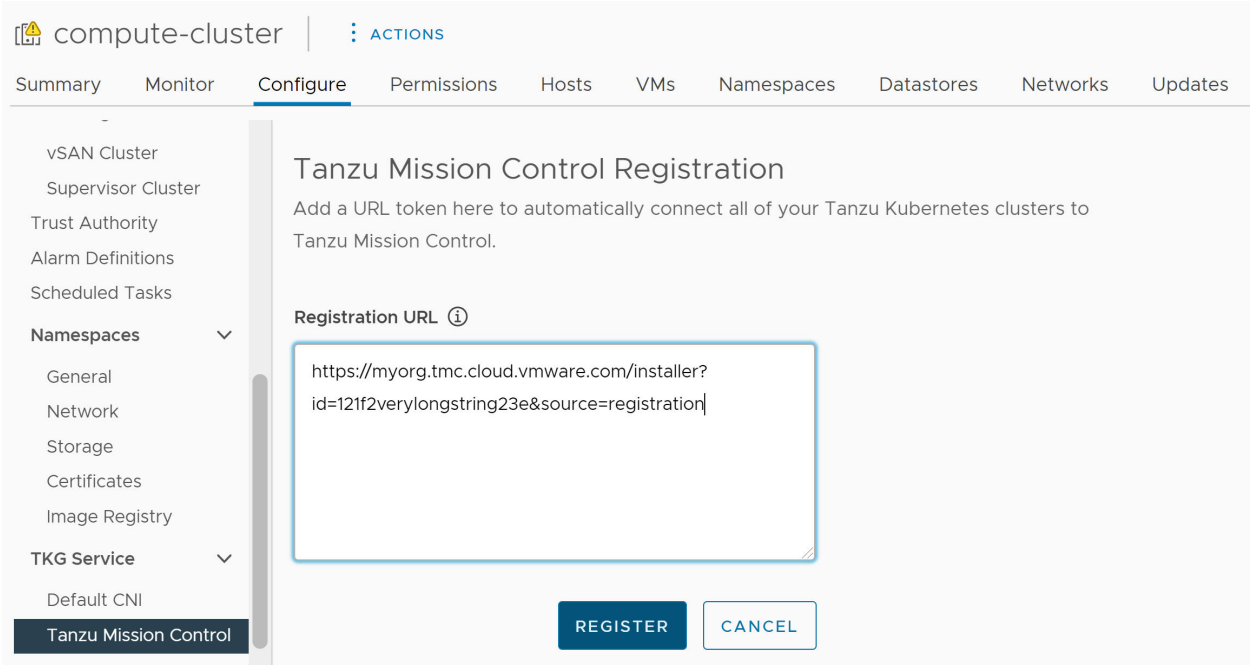
5 **워크로드 관리**를 사용하도록 설정된 vCenter 클러스터를 선택합니다.

6 **구성** 탭을 선택합니다.

7 **TKG 서비스 > Tanzu Mission Control**을 선택합니다.

8 **등록 URL** 필드에 등록 URL을 제공합니다.

9 **등록**을 클릭합니다.



Tanzu Mission Control 에이전트 제거

감독자 클러스터에서 Tanzu Mission Control 에이전트를 제거하려면 vSphere with Tanzu의 감독자 클러스터에서 클러스터 에이전트를 수동으로 제거를 참조하십시오.

Tanzu Kubernetes 클러스터에 대한 기본 CNI 설정

vSphere 관리자는 Tanzu Kubernetes 클러스터에 대한 기본 CNI(Container Network Interface)를 설정할 수 있습니다.

기본 CNI

Tanzu Kubernetes Grid 서비스는 Tanzu Kubernetes 클러스터에 대해 Antrea및 Calico라는 두 가지 CNI 옵션을 지원합니다.

시스템 정의 기본 CNI는 Antrea입니다. 기본 CNI 설정에 대한 자세한 내용은 Tanzu Kubernetes Grid 서비스 v1alpha1 API에 대한 구성 매개 변수 항목을 참조하십시오.

vSphere Client를 사용하여 기본 CNI를 변경할 수 있습니다. 기본 CNI를 설정하려면 다음 절차를 완료합니다.

경고 기본 CNI를 변경하는 것은 글로벌 작업입니다. 새로 설정된 기본값은 서비스에서 생성된 모든 새 클러스터에 적용됩니다. 기존 클러스터는 변경되지 않습니다.

1 vSphere Client를 사용하여 vSphere with Tanzu 환경에 로그인합니다.

2 워크로드 관리를 사용하도록 설정된 vCenter 클러스터를 선택합니다.

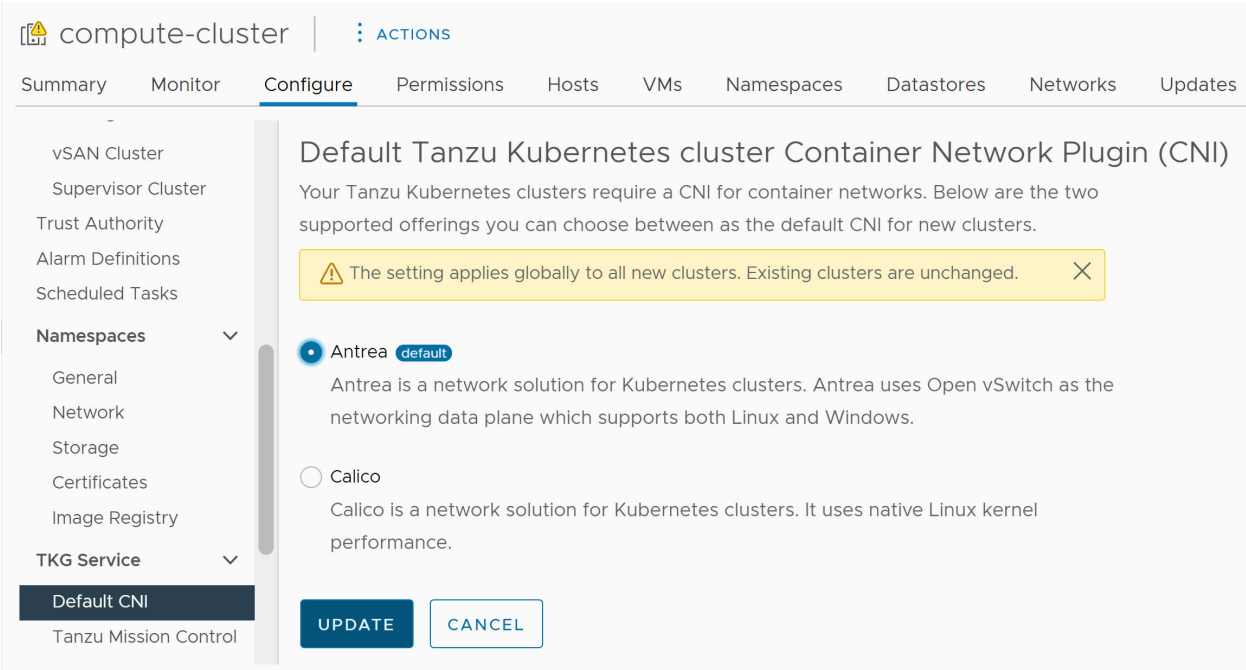
3 **구성** 탭을 선택합니다.

4 **TKG 서비스 > 기본 CNI**를 선택합니다.

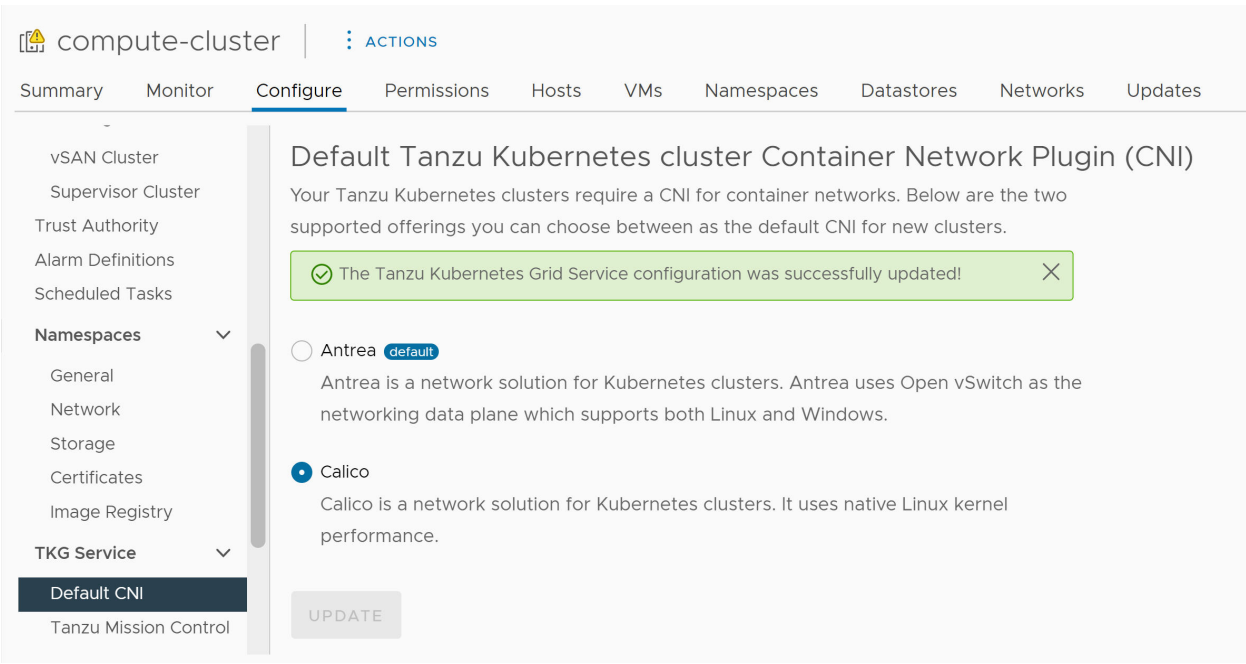
5 새 클러스터에 대한 기본 CNI를 선택합니다.

6 **업데이트**를 클릭합니다.

다음 이미지는 기본 CNI 선택 항목을 보여줍니다.



다음 이미지는 CNI 선택을 Antrea에서 Calico로 변경하는 것을 보여줍니다.



VDS 네트워킹으로 구성된 감독자 클러스터에 워크로드 네트워크 추가

vSphere 네트워킹 스택으로 구성된 감독자 클러스터의 경우 워크로드 네트워크를 생성하고 이것을 네임스페이스에 할당하여 Kubernetes 워크로드에 계층 2 분리를 제공할 수 있습니다. 워크로드 네트워크는 네임스페이스의 Tanzu Kubernetes 클러스터에 대한 연결을 제공하며 감독자 클러스터의 호스트에 연결된 스위치의 분산 포트 그룹에서 지원합니다.

감독자 클러스터에 대해 구현할 수 있는 토폴로지에 대한 자세한 내용은 vSphere 네트워킹 및 NSX Advanced Load Balancer를 사용하는 감독자 클러스터용 토폴로지 또는 HAProxy 로드 밸런서 배포를 위한 토폴로지.

참고 워크로드 네트워크에 대한 네트워킹 설정을 제공하는 DHCP 서버로 감독자 클러스터를 구성한 경우에는 감독자 클러스터 구성 후에 새 워크로드 네트워크를 생성할 수 없습니다.

사전 요구 사항

- 워크로드 네트워크를 지원할 분산 포트 그룹을 생성합니다.
- 워크로드 네트워크에 할당할 IP 범위가 환경에서 사용 가능한 모든 감독자 클러스터 내에서 고유한지 확인합니다.

절차

1. vSphere Client에서 감독자 클러스터로 이동합니다.
2. **구성**을 선택합니다.
3. **감독자 클러스터**에서 **네트워크**를 선택합니다.
4. **워크로드 네트워크**를 선택하고 **추가**를 클릭합니다.

옵션	설명
포트 그룹	이 워크로드 네트워크와 연결할 분산 포트 그룹을 선택합니다. **감독자 클러스터** 네트워킹에 대해 구성된 VDS(vSphere Distributed Switch)에는 포트 그룹이 포함되어 있으며 그 중에 선택할 수 있습니다.
네트워크 이름	네임스페이스에 할당된 경우 워크로드 네트워크를 식별하는 네트워크 이름입니다. 이 값은 선택한 포트 그룹의 이름에서 자동으로 채워지지만 적절히 변경할 수 있습니다.
IP 주소 범위	Tanzu Kubernetes 클러스터 노드의 IP 주소 할당을 위한 IP 범위를 입력합니다. IP 범위는 서브넷 마스크로 표시된 서브넷에 있어야 합니다. **참고** 각 워크로드 네트워크에 대해 고유한 IP 주소 범위를 사용해야 합니다. 여러 네트워크에 대해 동일한 IP 주소 범위를 구성하지 마십시오.

옵션	설명
서브넷 마스크	포트 그룹의 네트워크에 대한 서브넷 마스크의 IP 주소를 입력합니다.
게이트웨이	포트 그룹의 네트워크에 대한 기본 게이트웨이를 입력합니다. 게이트웨이는 서브넷 마스크로 표시된 서브넷에 있어야 합니다. **참고** HAProxy 로드 밸런서에 할당된 게이트웨이는 사용하지 마십시오.

5 **추가**를 클릭합니다.

다음에 수행할 작업

새로 생성된 워크로드 네트워크를 vSphere 네임스페이스에 할당합니다.

감독자 클러스터의 제어부 크기 변경

vSphere with Tanzu 환경에서 감독자 클러스터의 Kubernetes 제어부 VM의 크기를 변경하는 방법을 알아봅니다.

사전 요구 사항

- 클러스터에서 **클러스터 전체 구성 수정** 권한이 있는지 확인합니다.

절차

1 vSphere Client에서 감독자 클러스터로 이동합니다.

2 **구성**을 선택하고 **일반**을 클릭합니다.

3 **제어부 크기**를 확장하고 **편집**을 클릭한 후 드롭다운 메뉴에서 새 제어부 크기를 선택합니다.

4 **저장**을 클릭합니다.

제어부 크기는 스케일 업만 가능합니다.

감독자 클러스터에서 관리 네트워크 설정 변경

vSphere with Tanzu 환경의 감독자 클러스터 관리 네트워크에서 DNS 및 NTP 설정을 업데이트하는 방법을 알아봅니다.

사전 요구 사항

- 클러스터에서 **클러스터 전체 구성 수정** 권한이 있는지 확인합니다.

절차

1 vSphere Client에서 감독자 클러스터로 이동합니다.

2 **구성**을 선택합니다.

3 **감독자 클러스터**에서 **네트워크**를 선택합니다.

4 **관리 네트워크**를 클릭합니다.

5 DNS 및 NTP 설정을 편집합니다.

옵션	설명
DNS 서버	환경에서 사용하는 DNS 서버의 주소를 입력합니다. vCenter Server 시스템이 FQDN으로 등록되어 있으면 vSphere 환경에 사용하는 DNS 서버의 IP 주소를 입력해야 합니다. 그래야 감독자 클러스터에서 FQDN을 확인할 수 있습니다.
DNS 검색 도메인	Kubernetes 제어부 노드 내에서 DNS가 검색하는 도메인 이름(예: `corp.local`)을 입력합니다. 그래야 DNS 서버가 확인할 수 있습니다.
NTP 서버	환경에서 사용하는 NTP 서버의 주소를 입력합니다(있는 경우).

VDS 네트워킹으로 구성된 감독자 클러스터에서 워크로드 네트워크 설정 변경

VDS 네트워킹 스택으로 구성된 감독자 클러스터의 워크로드 네트워크에 대한 NTP 및 DNS 서버 설정을 변경하는 방법을 알아봅니다. 워크로드 네트워크에 대해 구성하는 DNS 서버는 Kubernetes 워크로드에 노출되는 외부 DNS 서버이며 감독자 클러스터 외부에서 호스팅되는 기본 도메인 이름을 확인합니다.

사전 요구 사항

- 클러스터에서 **클러스터 전체 구성 수정** 권한이 있는지 확인합니다.

절차

1 vSphere Client에서 감독자 클러스터로 이동합니다.

2 **구성**을 선택합니다.

3 **감독자 클러스터**에서 **네트워크**를 선택합니다.

4 **워크로드 네트워크**를 선택합니다.

5 DNS 서버 설정을 편집합니다.

vSphere 관리 구성 요소의 도메인 이름을 확인할 수 있는 DNS 서버(예: vCenter Server)의 주소를 입력합니다.

예: **`10.142.7.1`**.

DNS 서버의 IP 주소를 입력하면 각 제어부 VM에 정적 경로가 추가됩니다. 이것은 DNS 서버에 대한 트래픽이 워크로드 네트워크를 통과한다는 것을 나타냅니다.

지정하는 DNS 서버가 관리 네트워크와 워크로드 네트워크 간에 공유되는 경우에는 제어부 VM의 DNS 조회가 초기 설정 후 워크로드 네트워크를 통해 라우팅됩니다.

6 필요에 따라 NTP 설정을 편집합니다.

NSX-T Data Center로 구성된 감독자 클러스터에서 워크로드 네트워크 설정 변경

NSX-T Data Center에 대해 네트워킹 스택으로 구성된 감독자 클러스터의 DNS 서버, 네임스페이스 네트워크, 수신 및 송신에 대한 네트워킹 설정을 변경하는 방법을 알아봅니다.

사전 요구 사항

- 클러스터에서 **클러스터 전체 구성 수정** 권한이 있는지 확인합니다.

절차

1. vSphere Client에서 감독자 클러스터로 이동합니다.
2. **구성**을 선택합니다.
3. **감독자 클러스터**에서 **네트워크**를 선택합니다.
4. **워크로드 네트워크**를 선택합니다.
5. 필요에 따라 네트워킹 설정을 변경합니다.

옵션	설명
DNS 서버	vSphere 관리 구성 요소의 도메인 이름을 확인할 수 있는 DNS 서버(예: vCenter Server)의 주소를 입력합니다. 예: `10.142.7.1` DNS 서버의 IP 주소를 입력하면 각 제어부 VM에 정적 경로가 추가됩니다. 이것은 DNS 서버에 대한 트래픽이 워크로드 네트워크를 통과한다는 것을 나타냅니다. 지정하는 DNS 서버가 관리 네트워크와 워크로드 네트워크 간에 공유되는 경우에는 제어부 VM의 DNS 조회가 초기 설정 후 워크로드 네트워크를 통해 라우팅됩니다.
네임스페이스 네트워크	감독자 클러스터의 네임스페이스 세그먼트에 연결된 Kubernetes 워크로드의 IP 범위를 변경하려면 CIDR 주석을 입력합니다. NAT 모드가 구성되지 않은 경우 이 IP CIDR 범위는 라우팅 가능한 IP 주소여야 합니다.
수신	Kubernetes 서비스의 수신 IP 범위를 변경하려면 CIDR 주석을 입력합니다. 이 범위는 로드 밸런서 및 수신 유형의 서비스에 사용됩니다. Tanzu Kubernetes 클러스터의 경우 ServiceType 로드 밸런서를 통해 서비스를 게시하면 이 IP CIDR 블록의 IP 주소도 가져옵니다. **참고** 수신 및 워크로드 네트워크 필드에 CIDR을 추가할 수만 있고 기존 CIDR을 편집하거나 제거할 수는 없습니다.
송신	외부 서비스에 액세스하기 위해 감독자 클러스터를 나가는 트래픽에 대해 SNAT(소스 네트워크 주소 변환)의 IP 주소를 할당하기 위한 CIDR 주석을 입력합니다. 감독자 클러스터의 각 네임스페이스에는 송신 IP 주소가 하나만 할당됩니다. 송신 IP는 특정 네임스페이스의 vSphere 포드가 NSX-T Data Center 외부에서 통신하는 데 사용하는 IP 주소입니다.

초기 구성 또는 업그레이드 중 감독자 클러스터의 오류 상태 해결

처음에 vSphere 클러스터를 감독자 클러스터로 구성하거나 기존 감독자 클러스터의 설정을 업그레이드하거나 편집한 후 지정한 모든 설정은 구성이 완료될 때까지 검증되고 클러스터에 적용됩니다. 입력한 매개변수에 대해 상태 점검이 수행되어 구성에서 오류가 감지되면 감독자 클러스터가 오류 상태가 될 수 있습니다. 이러한 오류 상태를 해결해야 감독자 클러스터의 구성 또는 업그레이드를 재개할 수 있습니다.

감독자 클러스터의 상태는 vSphere Client의 **워크로드 관리 > 감독자 클러스터**에서 볼 수 있습니다. 클러스터 구성의 상태는 **구성 상태** 열에 표시됩니다.

표 5-1. vCenter Server 연결 오류

오류 메시지	원인	솔루션
제어부 VM <*VM name*>에 구성된 관리 DNS 서버로 vCenter 기본 네트워크 식별자 <*FQDN*>을(를) 확인할 수 없습니다. 관리 DNS 서버 <server name>에서 <*network name*>을(를) 확인할 수 있는지 확인합니다.	■ 하나 이상의 관리 DNS 서버에 연결할 수 있습니다. ■ 하나 이상의 관리 DNS가 정적으로 제공됩니다. ■ 관리 DNS 서버에 vCenter Server PNID에 대한 호스트 이름 조회가 없습니다. ■ vCenter Server PNID는 정적 IP 주소가 아닌 도메인 이름입니다.	■ 관리 DNS 서버에 vCenter Server PNID에 대한 호스트 항목을 추가합니다. ■ 구성된 DNS 서버가 올바른지 확인합니다.
제어부 VM <*VM name*>의 관리 네트워크에서 DHCP를 통해 획득한 DNS 서버로 vCenter 기본 네트워크 식별자 <*network name*>을(를) 확인할 수 없습니다. 관리 DNS 서버에서 <*network name*>을(를) 확인할 수 있는지 확인합니다.	■ DHCP 서버(하나 이상)에서 제공하는 관리 DNS 서버에 연결할 수 있습니다. ■ 관리 DNS 서버는 정적으로 제공됩니다. ■ 관리 DNS 서버에 vCenter Server PNID에 대한 호스트 이름 조회가 없습니다. ■ 관리 DNS 서버에 vCenter Server PNID에 대한 호스트 이름 조회가 없습니다. ■ vCenter Server PNID는 정적 IP 주소가 아닌 도메인 이름입니다.	■ 구성된 DHCP 서버에서 제공하는 관리 DNS 서버에 vCenter Server PNID에 대한 호스트 항목을 추가합니다. ■ DHCP 서버에서 제공한 DNS 서버가 올바른지 확인합니다.
구성된 관리 DNS 서버가 없기 때문에 제어부 VM <*VM name*>에서 호스트 <*host name*>을(를) 확인할 수 없습니다.	■ vCenter Server PNID는 정적 IP 주소가 아닌 도메인 이름입니다. ■ 구성된 DNS 서버가 없습니다.	관리 DNS 서버를 구성합니다.
제어부 VM <*VM name*>에서 호스트 <*host name*>을(를) 확인할 수 없습니다. 호스트 이름은 '.local' 최상위 도메인으로 끝나며, 따라서 관리 DNS 검색 도메인에 'local'이 포함되어야 합니다.	vCenter Server PNID에 **`.local`**이 TLD(최상위 도메인)로 포함되어 있지만 구성된 검색 도메인에는 **`local.`**이 포함되지 않습니다.	관리 DNS 검색 도메인에 **`local`**을 추가합니다.

표 5-1. vCenter Server 연결 오류 (계속)

오류 메시지	원인	솔루션
제어부 VM <*VM name*>에서 관리 DNS 서버 <*server name*>에 연결할 수 없습니다. 워크로드 네트워크를 통해 연결을 시도했습니다.	■ 관리 DNS 서버를 vCenter Server에 연결할 수 없습니다. ■ 제공된 `worker_dns` 값은 제공된 관리 DNS 값을 완전히 포함합니다. 즉, 트래픽은 워크로드 네트워크를 통해 라우팅됩니다. 감독자 클러스터는 정적 트래픽을 이러한 IP로 전송하기 위해 하나의 네트워크 인터페이스를 선택해야 하기 때문입니다.	■ 워크로드 네트워크를 확인하여 구성된 관리 DNS 서버로 라우팅할 수 있는지 확인합니다. ■ 워크로드 네트워크의 일부 다른 서버와 DNS 서버 간에 대체 라우팅을 트리거할 수 있는 충돌하는 IP 주소가 없는지 확인합니다. ■ 구성된 DNS 서버가 실제로 DNS 서버이고 포트 53에서 해당 DNS 포트를 호스팅하고 있는지 확인합니다. ■ 워크로드 DNS 서버가 제어부 VM의 IP(워크로드 네트워크 제공 IP)로부터의 연결을 허용하도록 구성되어 있는지 확인합니다. ■ 관리 DNS 서버의 주소에 오타가 없는지 확인합니다. ■ 검색 도메인에 불필요한 '~'가 포함되어 있지 않은지 확인합니다. 그러면 호스트 이름이 잘못 확인될 수 있습니다.
제어부 VM <*VM name*>에서 관리 DNS 서버 <*server name*>에 연결할 수 없습니다.	DNS 서버에 연결할 수 없습니다.	■ 관리 네트워크를 확인하여 관리 DNS 서버에 대한 경로가 존재하는지 확인합니다. ■ DNS 서버와 다른 서버 간에 대체 라우팅을 트리거할 수 있는 충돌하는 IP 주소가 없는지 확인합니다. ■ 구성된 DNS 서버가 실제로 DNS 서버이고 포트 53에서 해당 DNS 포트를 호스팅하고 있는지 확인합니다. ■ 관리 DNS 서버가 제어부 VM의 IP로부터의 연결을 허용하도록 구성되어 있는지 확인합니다. ■ 관리 DNS 서버의 주소에 오타가 없는지 확인합니다. ■ 검색 도메인에 불필요한 '~'가 포함되어 있지 않은지 확인합니다. 그러면 호스트 이름이 잘못 확인될 수 있습니다.
제어부 VM <vm name>에서 <*component name*> <*component address*>에 연결할 수 없습니다. 오류: *error message text*	■ 일반 네트워크 오류가 발생했습니다. ■ vCenter Server에 실제 연결하는 동안 오류가 발생했습니다.	■ vCenter Server, HAProxy, NSX Manager 또는 NSX Advanced Load Balancer와 같은 구성된 구성 요소의 호스트 이름 또는 IP 주소가 올바른지 확인합니다. ■ 관리 네트워크에서 충돌하는 IP, 방화벽 규칙 등의 외부 네트워크 설정이 있는지 확인합니다.

표 5-1. vCenter Server 연결 오류 (계속)

오류 메시지	원인	솔루션
제어부 VM <*VM name*>에서 vCenter <*vCenter Server name*> 인증서의 유효성을 검사할 수 없습니다. vCenter Server 인증서가 잘못되었습니다.	vCenter Server에서 제공한 인증서가 잘못된 형식이므로 신뢰할 수 없습니다.	■ `wcpsvc`를 다시 시작하여 제어부 VM의 신뢰할 수 있는 루트 번들이 최신 vCenter Server 루트 인증서로 최신 상태인지 확인합니다. ■ vCenter Server 인증서가 실제로 유효한 인증서인지 확인합니다.
제어부 VM <*VM name*>이(가) vCenter <*vCenter Server name*> 인증서를 신뢰하지 않습니다.	■ vCenter Server에서 제공한 `vmca.pem` 인증서가 제어부 VM에 구성된 것과 다릅니다. ■ 신뢰할 수 있는 루트 인증서가 vCenter Server Appliance에서 교체되었지만 `wcpsvc`가 다시 시작되지 않았습니다.	■ `wcpsvc`를 다시 시작하여 제어부 VM의 신뢰할 수 있는 루트 번들이 최신 vCenter Server 인증서 루트를 사용하는 최신 상태인지 확인합니다.

표 5-2. NSX Manager 연결 오류

제어부 VM <*VM name*>에서 NSX Server(<*NSX server name*>) 인증서의 유효성을 검사할 수 없습니다. 서버에서 반환된 지문(<*NSX-T address*>)이 vCenter에 등록된 예상 클라이언트 인증서 지문(<*vCenter Server name*>)과 일치하지 않습니다.	감독자 클러스터에 등록된 SSL 지문이 NSX Manager에서 제공한 인증서의 SHA-1 해시와 일치하지 않습니다.	■ NSX와 vCenter Server 인스턴스 간에 NSX Manager에서 신뢰를 다시 사용하도록 설정합니다. ■ vCenter Server에서 `wcpsvc`를 다시 시작합니다.
제어부 VM <*vm name*>에서 <*component name*> <*component address*>에 연결할 수 없습니다. 오류: *error message text*	일반 네트워크 오류가 발생했습니다.	■ NSX Manager의 관리 네트워크에서 외부 네트워크 설정, 충돌하는 IP, 방화벽 규칙 등을 확인합니다. ■ NSX 확장의 NSX Manager IP가 올바른지 확인합니다. ■ NSX Manager가 실행 중인지 확인합니다.

표 5-3. 로드 밸런서 오류

제어부 VM <*vm name*>이(가) 로드 밸런서(<*load balancer*>~<*load balancer endpoint*>)의 인증서를 신뢰하지 않습니다.	로드 밸런서가 제공하는 인증서가 제어부 VM에 구성된 인증서와 다릅니다.	로드 밸런서에 올바른 관리 TLS 인증서를 구성했는지 확인합니다.
제어부 VM <*vm name*>에서 로드 밸런서(<*load balancer*>~<*load balancer endpoint*>)의 인증서를 확인할 수 없습니다. 인증서가 잘못되었습니다.	로드 밸런서가 제공하는 인증서의 형식이 잘못되었거나 만료되었습니다.	구성된 로드 밸런서의 서버 인증서를 수정하십시오.

표 5-3. 로드 밸런서 오류 (계속)

제어부 VM <*vm name*>에서 사용자 이름 <*user name*> 및 제공된 암호를 사용하여 로드 밸런서(<*load balancer*>~<*load balancer endpoint*>)에 인증할 수 없습니다.	로드 밸런서의 사용자 이름 또는 암호가 잘못되었습니다.	로드 밸런서에 구성된 사용자 이름 및 암호가 올바른지 확인합니다.
제어부 VM <*vm name*>에서 로드 밸런서(<*load balancer*>~<*load balancer endpoint*>)에 연결하려고 시도하는 중 HTTP 오류가 발생했습니다.	제어부 VM이 로드 밸런서 끝점에 연결할 수 있지만 끝점이 성공적인(200) HTTP 응답을 반환하지 않습니다.	로드 밸런서가 정상이고 요청을 수락하는지 확인합니다.
제어부 VM <*vm name*>에서 <*load balancer*>(<*load balancer endpoint*>)에 연결할 수 없습니다. 오류: <*error text*>	■ 일반 네트워크 오류가 발생했습니다. ■ 일반적으로 로드 밸런서가 작동하지 않거나 일부 방화벽이 연결을 차단한다는 의미입니다.	■ 로드 밸런서 끝점에 액세스할 수 있는지 확인합니다. ■ 로드 밸런서에 대한 연결을 차단하는 방화벽이 없는지 확인합니다.

vSphere with Tanzu에서 HTTP 프록시 설정 구성

감독자 클러스터 및 Tanzu Kubernetes 클러스터에 대한 HTTP 프록시 설정을 구성하는 방법을 알아봅니다. 감독자 클러스터 및 Tanzu Kubernetes 클러스터를 Tanzu Mission Control에 등록할 때 이러한 클러스터에 대한 HTTP 프록시를 구성하는 워크플로가 무엇인지 확인합니다. Tanzu Mission Control에서 관리 클러스터로 등록하는 온-프레미스 감독자 클러스터에 대한 컨테이너 트래픽 및 이미지 가져오기에 HTTP 프록시를 사용합니다.

Tanzu Mission Control에서 사용할 감독자 클러스터 및 Tanzu Kubernetes 클러스터에서 HTTP 프록시 설정을 구성하는 워크플로

Tanzu Mission Control에서 관리 클러스터로 등록하려는 감독자 클러스터에서 HTTP 프록시를 구성하려면 다음 단계를 수행합니다.

1 vSphere에서 vCenter Server의 HTTP 프록시 설정을 상속하거나 네임스페이스 관리 클러스터 API 또는 DCLI 명령줄을 통해 개별 감독자 클러스터에서 프록시 설정을 구성하여 감독자 클러스터에서 HTTP 프록시를 구성합니다.

2 Tanzu Mission Control에서 vSphere with Tanzu의 감독자 클러스터에 구성한 프록시 설정을 사용하여 프록시 구성 개체를 생성합니다. vSphere with Tanzu에서 실행되는 Tanzu Kubernetes Grid 서비스 클러스터에 대한 프록시 구성 개체 생성을 참조하십시오.

3 Tanzu Mission Control에서 감독자 클러스터를 관리 클러스터로 등록할 때 이 프록시 구성 개체를 사용합니다. Tanzu Mission Control에 관리 클러스터 등록 및 vSphere with Tanzu에서 감독자 클러스터 등록 완료를 참조하십시오.

Tanzu Mission Control에서 워크로드 클러스터로 추가하거나 프로비저닝하는 Tanzu Kubernetes 클러스터에 HTTP 프록시를 구성하려면 다음을 수행합니다.

1 Tanzu Kubernetes 클러스터에 사용할 프록시 설정으로 프록시 구성 개체를 생성합니다. vSphere with Tanzu에서 실행되는 Tanzu Kubernetes Grid 서비스 클러스터에 대한 프록시 구성 개체 생성을 참조하십시오.

2 Tanzu Kubernetes 클러스터를 워크로드 클러스터로 추가하거나 프로비저닝할 때 해당 프록시 구성 개체를 사용합니다. vSphere with Tanzu에서 클러스터 프로비저닝 및 Tanzu Mission Control 관리에 워크로드 클러스터 추가를 참조하십시오.

Tanzu Kubernetes에서 vSphere with Tanzu 클러스터에 대한 HTTP 프록시 구성

다음 방법 중 하나를 사용하여 vSphere with Tanzu에서 Tanzu Kubernetes 클러스터에 대한 프록시를 구성합니다.

- 개별 Tanzu Kubernetes 클러스터에 대한 프록시 설정을 구성합니다. Tanzu Kubernetes Grid 서비스 v1alpha2 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하기 위한 구성 매개 변수를 참조하십시오. 구성 YAML의 예는 Tanzu Kubernetes Grid 서비스 v1alpha2 API를 사용하여 사용자 지정 Tanzu Kubernetes 클러스터를 프로비저닝하기 위한 예제 YAML을 참조하십시오.
- 모든 Tanzu Kubernetes 클러스터에 적용될 글로벌 프록시 구성을 생성합니다. Tanzu Kubernetes Grid 서비스 v1alpha2 API에 대한 구성 매개 변수를 참조하십시오.

참고 Tanzu Mission Control을 사용하여 Tanzu Kubernetes 클러스터를 관리하는 경우 vSphere with Tanzu에서 클러스터 YAML 파일을 통해 프록시 설정을 구성할 필요가 없습니다. Tanzu Kubernetes 클러스터를 Tanzu Mission Control에 워크로드 클러스터로 추가할 때 프록시 설정을 구성할 수 있습니다.

새로 생성된 vSphere 7.0 업데이트 3 감독자 클러스터에서 프록시 설정 구성

vSphere 7.0 업데이트 3 환경에서 새로 생성된 감독자 클러스터의 경우 HTTP 프록시 설정이 vCenter Server에서 상속됩니다. vCenter Server에서 HTTP 프록시 설정을 구성하기 전이나 구성한 후에 감독자 클러스터를 생성하든 상관없이 설정은 클러스터에 상속됩니다.

vCenter Server에서 HTTP 프록시 설정을 구성하는 방법에 대한 자세한 내용은 DNS, IP 주소 및 프록시 설정 구성을 참조하십시오.

클러스터 관리 API 또는 DCLI를 통해 개별 감독자 클러스터에서 상속된 HTTP 프록시 구성을 재정의할 수도 있습니다.

vCenter Server 프록시 설정을 상속하는 것이 새로 생성된 vSphere 7.0.3 감독자 클러스터의 기본 구성이므로 감독자 클러스터에 프록시가 필요하지 않지만 vCenter Server에는 계속 필요한 경우 클러스터 관리 API 또는 DCLI를 사용하여 HTTP 프록시 설정을 상속하지 않을 수도 있습니다.

vSphere 7.0 업데이트 3으로 업그레이드된 감독자 클러스터에서 프록시 설정 구성

감독자 클러스터를 vSphere 7.0 업데이트 3으로 업그레이드한 경우 vCenter Server의 HTTP 프록시 설정이 자동으로 상속되지 않습니다. 이 경우 `vcenter/namespace-management/clusters` API 또는 DCLI 명령줄을 사용하여 감독자 클러스터의 프록시 설정을 구성합니다.

클러스터 관리 API를 사용하여 감독자 클러스터에서 HTTP 프록시 구성

`vcenter/namespace-management/clusters` API를 통해 감독자 클러스터 프록시 설정을 구성합니다. API는 감독자 클러스터의 프록시 구성을 위한 세 가지 옵션을 제공합니다.

API 설정	새로 생성된 vSphere 7.0.3 감독자 클러스터	vSphere 7.0.3으로 업그레이드된 감독자 클러스터
`VC_INHERITED`	이것은 새 감독자 클러스터에 대한 기본 설정이며 API를 사용하여 감독자 클러스터 프록시 설정을 구성할 필요가 없습니다. vCenter Server에서 관리 인터페이스를 통해 프록시 설정을 구성할 수 있습니다.	이 설정을 사용하여 HTTP 프록시 구성을 vSphere 7.0.3으로 업그레이드된 감독자 클러스터에 푸시합니다.
`CLUSTER_CONFIGURED`	다음 중 하나의 경우 이 설정을 사용하여 vCenter Server에서 상속된 HTTP 프록시 구성을 재정의합니다. ■ 감독자 클러스터가 vCenter Server와 다른 서브넷에 있으며 다른 프록시 서버가 필요합니다. ■ 프록시 서버가 사용자 지정 CA 번들을 사용합니다.	다음 중 하나의 경우 이 설정을 사용하여 vSphere 7.0.3으로 업그레이드된 개별 감독자 클러스터에 대한 HTTP 프록시를 구성합니다. ■ 감독자 클러스터가 vCenter Server와 다른 서브넷에 있고 다른 프록시 서버가 필요하므로 vCenter Server 프록시를 사용할 수 없습니다. ■ 프록시 서버가 사용자 지정 CA 번들을 사용합니다.
`NONE`	감독자 클러스터가 인터넷에 직접 연결되어 있고 vCenter Server에 프록시가 필요한 경우 이 설정을 사용합니다. NONE 설정은 vCenter Server의 프록시 설정이 감독자 클러스터에서 상속되는 것을 방지합니다.	

HTTP 프록시를 감독자 클러스터로 설정하거나 기존 설정을 수정하려면 vCenter Server와의 SSH 세션에서 다음 명령을 사용합니다.

```
vc_address=<IP address>
cluster_id=domain-c<number>
session_id=$(curl -ksX POST --user '<SSO user name>:<password>' https://$vc_address/api/
session | xargs -t)
curl -k -X PATCH -H "vmware-api-session-id: $session_id" -H "Content-Type: application/json"
-d '{ "cluster_proxy_config": { "proxy_settings_source": "CLUSTER_CONFIGURED",
"http_proxy_config":"<proxy_url>" } }' https://$vc_address/api/vcenter/namespace-management/
clusters/$cluster_id
```

전체 클러스터 ID에서 domain_c<number>만 전달하면 됩니다. 예를 들어 클러스터 ID ClusterComputeResource:domain-c50:5bbb510f-759f-4e43-96bd-97fd703b4edb에서 domain-c50을 가져옵니다.

VC_INHERITED 또는 NONE 설정을 사용하는 경우 명령에서 "http_proxy_config:<proxy_url>"을 생략합니다.

사용자 지정 CA 번들을 사용하려면 TSL CA 인증서를 일반 텍스트로 제공하여 명령에 "tlsRootCaBundle": "<TLS_certificate>"를 추가합니다.

DCLI를 사용하여 감독자 클러스터에서 HTTP 프록시 설정 구성

다음 DCLI 명령을 사용하면 CLUSTER_CONFIGURED 설정을 사용하여 감독자 클러스터에 대한 HTTP 프록시 설정을 구성할 수 있습니다.

```
<dcli> namespacemanagement clusters update --cluster domain-c57 --cluster-proxy-config-http-proxy-config <proxy URL> --cluster-proxy-config-https-proxy-config <proxy URL> --cluster-proxy-config-proxy-settings-source CLUSTER_CONFIGURED
```

감독자 클러스터 제어부의 로그를 원격 rsyslog로 스트리밍

귀중한 로깅 데이터의 손실을 방지하기 위해 감독자 클러스터 제어부 VM에서 원격 rsyslog 수신기로의 로그 스트리밍을 구성하는 방법을 알아봅니다.

감독자 클러스터 제어부 VM의 구성 요소에서 생성된 로그는 VM의 파일 시스템에 로컬로 저장됩니다. 많은 양의 로그가 누적되면 로그가 높은 속도로 순환되어 다양한 문제의 근본 원인을 식별하는 데 도움이 될 수 있는 귀중한 메시지가 손실됩니다. vCenter Server 및 감독자 클러스터 제어부 VM은 로컬 로그를 원격 rsyslog 수신기로 스트리밍하는 기능을 지원합니다. 이 기능은 다음 서비스 및 구성 요소에 대한 로그를 캡처하는 데 도움이 됩니다.

- vCenter Server: 워크로드 제어부 서비스, ESX Agent Manager 서비스, CA(인증 기관) 서비스 및 vCenter Server에서 실행 중인 기타 모든 서비스.
- 감독자 클러스터 제어부 구성 요소 및 감독자 클러스터 내장형 서비스(예: VM 서비스 및 Tanzu Kubernetes Grid 서비스).

로컬 로그 데이터를 수집하고 원격 rsyslog 수신기로 스트리밍하도록 vCenter Server Appliance를 구성할 수 있습니다. 이 구성이 vCenter Server에 적용되면 vCenter Server 내부에서 실행되는 rsyslog 발신자가 해당 vCenter Server 시스템 내부의 서비스에서 생성된 로그를 보내기 시작합니다.

감독자 클러스터는 vCenter Server와 동일한 메커니즘을 사용하여 로컬 로그를 오프로드하여 구성 관리 부담을 줄여줍니다. 워크로드 제어부 서비스는 로그를 주기적으로 폴링하여 vCenter Server rsyslog 구성을 모니터링합니다. 워크로드 제어부 서비스가 원격 vCenter Server rsyslog 구성이 비어 있지 않은 것을 감지하면 서비스는 이 구성을 모든 감독자 클러스터의 각 제어부 VM에 전파합니다. 이로 인해 원격 rsyslog 수신기에 부담을 줄 수 있는 대량의 rsyslog 메시지 트래픽이 생성될 수 있습니다. 따라서 수신기 시스템에는 대량의 rsyslog 메시지를 유지하기에 충분한 스토리지 용량이 있어야 합니다.

vCenter Server에서 rsyslog 구성을 제거하면 vCenter Server에서 rsyslog 메시지가 중지됩니다. 워크로드 제어부 서비스는 변경 내용을 감지하고 모든 감독자 클러스터의 각 제어부 VM에 전파하여 결국 제어부 VM 스트림도 중지됩니다.

구성 단계

다음 단계를 수행하여 감독자 클러스터 제어부 VM에 대한 rsyslog 스트리밍을 구성합니다.

1 다음에 해당하는 시스템을 프로비저닝하여 rsyslog 수신기를 구성합니다.

- 수신기 모드에서 rsyslog 서비스를 실행합니다. rsyslog 설명서에서 고성능으로 대량의 메시지 수신 예를 참조하십시오.
- 대량의 로그 데이터를 수용하기에 충분한 스토리지 공간이 있습니다.
- vCenter Server 및 감독자 클러스터 제어부 VM에서 데이터를 수신하기 위한 네트워크 연결이 있습니다.

2 "https://<vcenter server address>:5480" 에서 vCenter Server Appliance 관리 인터페이스에 루트로 로그인합니다.

3 vCenter Server Appliance 관리 인터페이스를 통해 rsyslog 수신기로 스트리밍하도록 vCenter Server를 구성합니다. vCenter Server 로그 파일을 원격 Syslog 서버에 전달을 참조하십시오.

vCenter Server의 rsyslog 구성이 감독자 클러스터 제어부 VM에 적용되는 데 몇 분 정도 걸릴 수 있습니다. vCenter Server Appliance의 워크로드 제어부 서비스는 5분마다 장치 구성을 폴링하여 사용 가능한 모든 감독자 클러스터에 전파합니다. 전파를 완료하는 데 필요한 시간은 환경의 감독자 클러스터 수에 따라 다릅니다. 감독자 클러스터의 제어부 VM 중 일부가 비정상 상태이거나 다른 작업을 수행하는 경우 워크로드 제어부 서비스는 성공할 때까지 rsyslog 구성 적용을 재시도합니다.

제어부 VM 구성 요소의 로그 검사

감독자 클러스터 제어부 VM의 rsyslog는 이러한 로그 메시지의 소스 구성 요소를 나타내는 태그를 로그 메시지에 포함합니다.

로그 태그	설명

`vns-control-plane-pods <pod_name>/<instance_number>.log`	제어부 VM의 Kubernetes 포드에서 발생한 로그. 예:
	`vns-control-plane-pods etcd/0.log`
	또는
	`vns-control-plane-pods nsx-ncp/573.log`

`vns-control-plane-imc`	제어부 VM의 초기 구성 로그.
`vns-control-plane-boostrap`	Kubernetes 노드 제어부 배포의 부트스트랩 로그.
`vns-control-plane-upgrade-logs`	제어부 노드 패치 및 부 버전 업그레이드의 로그.
`vns-control-plane-svchost-logs`	제어부 VM 시스템 수준 서비스 호스트 또는 에이전트 로그.

로그 태그	설명
vns-control-plane-update-controller	제어부의 원하는 상태 동기화 프로그램 및 실현 프로그램 (realizer) 로그.
vns-control-plane-compact-etcd-logs	제어부 etcd 서비스 스토리지 압축을 유지하기 위한 로그.

vSphere with Tanzu에서 컨텐츠 라이브러리 생성 및 관리

6

독립형 VM 및 Tanzu Kubernetes 클러스터와 같은 vSphere with Tanzu 개체는 컨텐츠 라이브러리를 템플릿, 이미지, 배포 및 배포와 관련된 기타 파일의 중앙 집중식 저장소로 사용합니다.

본 장은 다음 항목을 포함합니다.

- Tanzu Kubernetes 릴리스용 컨텐츠 라이브러리 생성 및 관리
- vSphere with Tanzu에서 독립형 VM에 대한 컨텐츠 라이브러리 생성 및 관리

Tanzu Kubernetes 릴리스용 컨텐츠 라이브러리 생성 및 관리

VMware Tanzu는 Kubernetes 소프트웨어 버전을 Tanzu Kubernetes 릴리스로 배포합니다. 이러한 릴리스를 사용하려면 vSphere 컨텐츠 라이브러리를 구성하고 사용 가능한 릴리스를 동기화합니다. 이 작업은 구독 기반 모델을 사용하거나 요청 시 수행할 수 있습니다. 인터넷이 제한된 환경에서 Tanzu Kubernetes를 프로비저닝하려는 경우에는 로컬 라이브러리를 생성하고 수동으로 릴리스를 가져올 수 있습니다.

Tanzu Kubernetes 릴리스 배포 정보

Tanzu Kubernetes 릴리스는 Tanzu Kubernetes 클러스터와 함께 사용하기 위해 VMware에서 서명 및 지원하는 Kubernetes 소프트웨어 배포를 제공합니다. Tanzu Kubernetes 릴리스는 vSphere 컨텐츠 라이브러리를 사용하여 확보하고 관리합니다.

각 Tanzu Kubernetes 릴리스는 OVA 템플릿으로 배포됩니다. Tanzu Kubernetes Grid 서비스는 OVA 템플릿을 사용하여 Tanzu Kubernetes 클러스터의 가상 시스템 노드를 구성합니다.

Tanzu Kubernetes 릴리스 목록 및 감독자 클러스터와의 호환성에 대해서는 Tanzu Kubernetes 릴리스의 릴리스 정보를 참조하십시오.

Tanzu Kubernetes 릴리스는 Photon OS 및 Ubuntu에서 지원됩니다. OVA 템플릿에서 구축된 가상 시스템의 디스크 크기는 고정되어 있습니다. CPU 및 RAM 리소스는 Tanzu Kubernetes 클러스터를 프로비저닝할 때 지정합니다. Tanzu Kubernetes 클러스터에 대한 가상 시스템 클래스의 내용을 참조하십시오.

Tanzu Kubernetes Grid 서비스는 vSphere 컨텐츠 라이브러리에서 Tanzu Kubernetes 릴리스 OVA 템플릿을 끌어옵니다. 구독 컨텐츠 라이브러리를 사용하여 프로세스를 자동화하거나 인터넷이 제한된 환경에 로컬 컨텐츠 라이브러리를 사용할 수 있습니다. Tanzu Kubernetes 릴리스용 구독 컨텐츠 라이브러리 생성, 보안 및 동기화 및 Tanzu Kubernetes 릴리스용 로컬 컨텐츠 라이브러리 생성, 보안 및 동기화의 내용을 참조하십시오.

컨텐츠 라이브러리의 크기는 새로운 Tanzu Kubernetes 릴리스가 배포되면서 시간에 따라 증가할 수 있습니다. 기본 스토리지의 공간이 부족해지면 새 컨텐츠 라이브러리로 마이그레이션할 수 있습니다. Tanzu Kubernetes 클러스터를 새 컨텐츠 라이브러리로 마이그레이션의 내용을 참조하십시오.

컨텐츠 라이브러리를 생성하고 동기화한 후에는 Tanzu Kubernetes 클러스터를 프로비저닝하려는 각 vSphere 네임스페이스를 구성합니다. 여기에는 컨텐츠 라이브러리 및 가상 시스템 클래스를 vSphere 네임스페이스와 연결하는 것이 포함됩니다. 이 시점에 vSphere 네임스페이스에 로그인하여 Tanzu Kubernetes 릴리스를 사용할 수 있는지 확인할 수 있습니다. Tanzu Kubernetes 릴리스에 대한 vSphere 네임스페이스 구성의 내용을 참조하십시오.

Tanzu Kubernetes 릴리스용 구독 컨텐츠 라이브러리 생성, 보안 및 동기화

Tanzu Kubernetes 클러스터에서 사용할 Tanzu Kubernetes 릴리스를 저장하려면 vSphere with Tanzu를 사용하도록 설정된 vCenter Server에 구독 컨텐츠 라이브러리를 생성합니다.

구독 컨텐츠 라이브러리는 게시된 컨텐츠 라이브러리에서 만들어집니다. 구독이 생성되면 시스템에서 이를 게시된 라이브러리와 동기화합니다. 동기화 모드를 즉시 또는 요청 시 중에 선택할 수 있습니다. 자세한 내용은 구독 라이브러리 관리를 참조하십시오.

사전 요구 사항

Tanzu Kubernetes 릴리스 배포 정보 항목을 검토하십시오.

컨텐츠 라이브러리를 생성하려면 다음 vSphere 권한이 필요합니다.

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

절차

1 vSphere Client를 사용하여 vCenter Server에 로그인합니다.

2 **메뉴 > 컨텐츠 라이브러리**를 선택합니다.

3 **생성**을 클릭합니다.

새 컨텐츠 라이브러리 마법사가 열립니다.

4 컨텐츠 라이브러리의 **이름 및 위치**를 지정하고 완료되면 **다음**을 클릭합니다.

필드	설명
이름	알아보기 쉬운 이름을 입력합니다(예: **`TanzuKubernetesRelease-subscriber`**).
참고	설명을 포함합니다(예: **`On-demand subscription library for Tanzu Kubernetes releases`**).
vCenter Server	vSphere with Tanzu를 사용하도록 설정된 vCenter Server 인스턴스를 선택합니다.

5 **컨텐츠 라이브러리 구성** 페이지에서 컨텐츠 라이브러리 구독을 구성하고 완료되면 **다음**을 클릭합니다.

a **구독 컨텐츠 라이브러리** 옵션을 선택합니다.

참고 로컬 컨텐츠 라이브러리를 사용하려면 Tanzu Kubernetes 릴리스용 로컬 컨텐츠 라이브러리 생성, 보안 및 동기화 항목을 참조하십시오.

b 게시자의 **구독 URL** 주소를 입력합니다.

https://wp-content.vmware.com/v2/latest/lib.json

c **컨텐츠 다운로드** 옵션의 경우 다음 중 하나를 선택합니다.

옵션	설명
즉시	구독 프로세스는 라이브러리 메타데이터와 이미지를 모두 동기화합니다. 게시된 라이브러리에서 항목이 삭제된 경우, 해당 컨텐츠는 구독 라이브러리 스토리지에 남아 있으므로 수동으로 삭제해야 합니다.
필요한 경우	구독 프로세스는 라이브러리 메타데이터만 동기화합니다. Tanzu Kubernetes Grid 서비스는 게시될 때 이미지를 다운로드합니다. 항목이 더 이상 필요하지 않은 경우에는 항목 컨텐츠를 삭제하여 스토리지 공간을 확보할 수 있습니다. 스토리지를 절약하려면 이 옵션을 사용하는 것이 좋습니다.

6 메시지가 표시되면 SSL 인증서 지문을 수락합니다.

SSL 인증서 지문은 인벤토리에서 구독 컨텐츠 라이브러리를 삭제할 때까지 시스템에 저장됩니다.

7 **보안 정책 적용** 페이지에서 OVF 보안 정책을 구성하고 완료되면 **다음**을 클릭합니다.

a **보안 정책 적용**을 선택합니다.

b **OVF 기본 정책**을 선택합니다.

이 옵션을 선택하면 동기화 프로세스 중에 OVF 서명 인증서가 확인됩니다. 인증서 검증을 통과하지 못한 OVF 템플릿은 **확인 실패** 태그로 표시됩니다. 템플릿 메타데이터는 유지되지만 OVF 파일은 동기화할 수 없습니다.

참고 현재 **OVF 기본 정책**은 지원되는 유일한 보안 정책입니다.

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

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

10 **컨텐츠 라이브러리** 페이지에서 생성한 새 컨텐츠 라이브러리를 선택합니다.

11 라이브러리 컨텐츠의 동기화를 확인하거나 완료합니다.

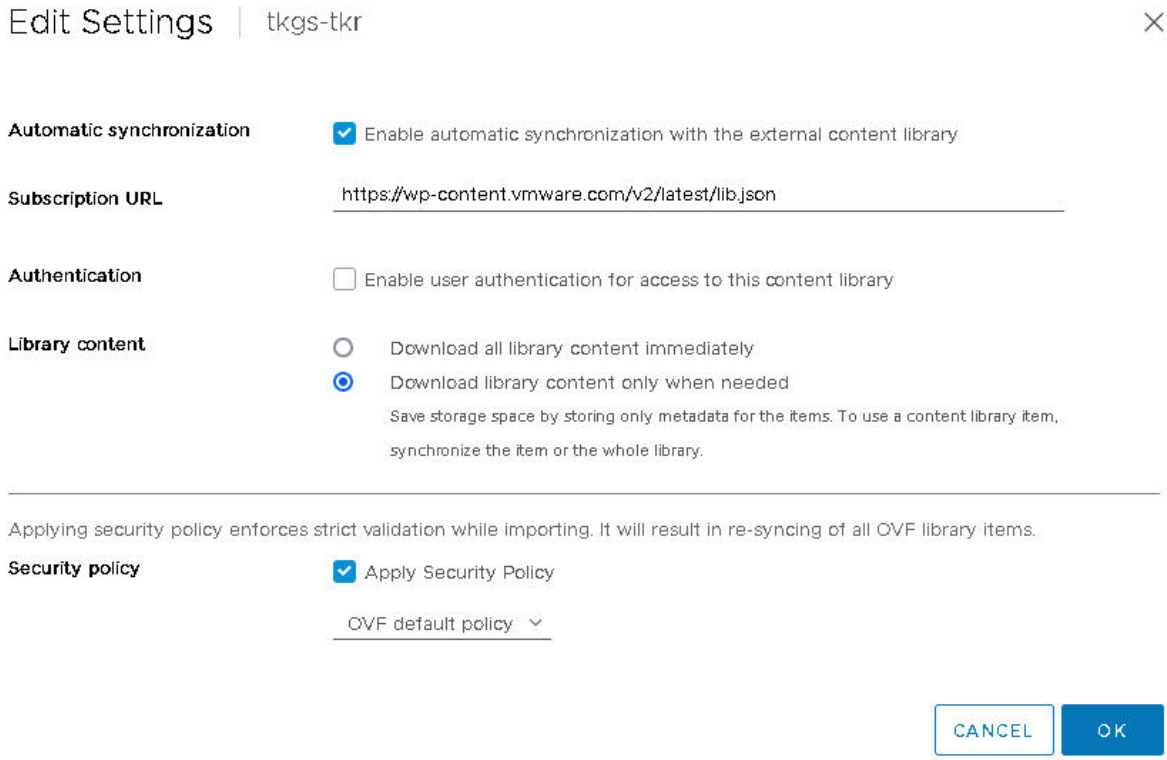
동기화 옵션	설명
즉시	모든 컨텐츠를 즉시 다운로드하도록 선택한 경우 라이브러리가 동기화되었는지 확인합니다. 동기화된 라이브러리 컨텐츠를 보려면 **템플릿 > OVF 및 OVA 템플릿**을 선택합니다.
필요한 경우	요청 시 라이브러리를 동기화하도록 선택한 경우 다음 두 가지 옵션을 사용할 수 있습니다. ■ **작업 > 동기화**를 사용하여 전체 라이브러리를 동기화합니다. ■ 항목을 마우스 오른쪽 버튼으로 클릭하고 **동기화**를 선택하여 항목만 동기화합니다. 동기화된 라이브러리 컨텐츠를 보려면 **템플릿 > OVF 및 OVA 템플릿**을 선택합니다.

12 **필요한 경우** 옵션을 선택한 경우 사용하려는 OVF 템플릿을 다운로드합니다.

필요한 경우 옵션을 선택하면 이미지 파일은 로컬로 저장되지 않고 메타데이터만 저장됩니다. 템플릿 파일을 다운로드하려면 항목을 선택하고 마우스 오른쪽 버튼을 클릭한 다음 **항목 동기화**를 선택합니다.

13 구독 컨텐츠 라이브러리 설정을 업데이트하려면 **작업 > 설정 편집**을 선택합니다.

설정	값
구독 URL	https://wp-content.vmware.com/v2/latest/lib.json
인증	사용 안 함
라이브러리 컨텐츠	필요한 경우 다운로드
보안 정책	OVF 기본 정책



다음에 수행할 작업

컨텐츠 라이브러리 및 가상 시스템 클래스를 네임스페이스와 연결하여 Tanzu Kubernetes 클러스터를 프로비저닝할 각 vSphere 네임스페이스를 구성합니다. Tanzu Kubernetes 릴리스에 대한 vSphere 네임스페이스 구성의 내용을 참조하십시오.

Tanzu Kubernetes 릴리스용 로컬 컨텐츠 라이브러리 생성, 보안 및 동기화

인터넷 제한("에어갭") 환경에서 Tanzu Kubernetes 클러스터를 프로비저닝하려면 로컬 컨텐츠 라이브러리를 생성하고 각 Tanzu Kubernetes 릴리스를 수동으로 가져옵니다.

로컬 컨텐츠 라이브러리를 생성하려면 라이브러리를 구성하고 OVA 파일을 다운로드하여 로컬 컨텐츠 라이브러리로 가져와야 합니다.

사전 요구 사항

Tanzu Kubernetes 릴리스 배포 정보 항목을 검토하십시오.

구독 컨텐츠 라이브러리를 생성하려면 다음 권한이 필요합니다.

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

절차

1 vSphere Client를 사용하여 vCenter Server에 로그인합니다.

2 **메뉴**를 클릭합니다.

3 **컨텐츠 라이브러리**를 클릭합니다.

4 **생성**을 클릭합니다.

시스템에 **새 컨텐츠 라이브러리** 마법사가 표시됩니다.

5 컨텐츠 라이브러리의 **이름 및 위치**를 지정하고 완료되면 **다음**을 클릭합니다.

필드	설명
이름	알아보기 쉬운 이름을 입력합니다(예: **`TanzuKubernetesRelease-local`**).
참고	설명을 포함합니다(예: **`Local library for Tanzu Kubernetes releases`**).
vCenter Server	vSphere with Tanzu를 사용하도록 설정된 vCenter Server 인스턴스를 선택합니다.

6 **컨텐츠 라이브러리 구성** 페이지에서 **로컬 컨텐츠 라이브러리**를 선택하고 **다음**을 클릭합니다.

아래 설명된 대로 로컬 컨텐츠 라이브러리의 경우 사용하려는 OVF 템플릿을 수동으로 가져옵니다.

참고 구독 컨텐츠 라이브러리를 사용하려면 Tanzu Kubernetes 릴리스용 구독 컨텐츠 라이브러리 생성, 보안 및 동기화 항목을 참조하십시오.

7 **보안 정책 적용** 페이지에서 OVF 보안 정책을 구성하고 완료되면 **다음**을 클릭합니다.

a **보안 정책 적용**을 선택합니다.

b **OVF 기본 정책**을 선택합니다.

이 옵션을 선택하면 동기화 프로세스 중에 OVF 서명 인증서가 확인됩니다. 인증서 검증을 통과하지 못한 OVF 템플릿은 **확인 실패** 태그로 표시됩니다. 템플릿 메타데이터는 유지되지만 OVF 파일은 동기화할 수 없습니다.

참고 현재 **OVF 기본 정책**은 지원되는 유일한 보안 정책입니다.

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

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

10 **컨텐츠 라이브러리** 페이지에서 생성한 새 컨텐츠 라이브러리를 선택합니다.

11 로컬 컨텐츠 라이브러리로 가져오려는 각 Tanzu Kubernetes 릴리스에 대한 OVA 파일을 다운로드합니다.

a 브라우저를 사용하여 다음 URL로 이동합니다.

https://wp-content.vmware.com/v2/latest/

b 원하는 이미지의 디렉토리를 클릭합니다. 일반적으로 이 디렉토리는 Kubernetes 배포의 최신 또는 최신 버전입니다.

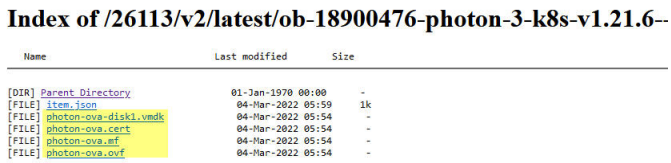
예:

```
ob-18186591-photon-3-k8s-v1.20.7---vmware.1-tkg.1.7fb9067
```

참고 배포 이름은 파일을 로컬 컨텐츠 라이브러리로 가져오는 데 필요하므로 파일에 복사해두거나 절차를 완료할 때까지 브라우저를 열어 두는 것이 좋습니다.

c 다음 각 파일에 대해 마우스 오른쪽 버튼을 클릭하고 **다른 이름으로 링크 저장**을 선택합니다.

- photon-ova-disk1.vmdk
- photon-ova.cert
- photon-ova.mf
- photon-ova.ovf



d 각 파일이 로컬 파일 시스템으로 다운로드되었는지 확인합니다.

참고 가져오기 프로세스 중 소스에서 인증서 및 매니페스트 파일을 사용할 수 없는 경우 가져온 라이브러리 항목을 사용할 수 없습니다. 즉, 보안 정책으로 구성된 로컬 컨텐츠 라이브러리의 경우 4개의 필수 파일이 모두 ovf 및 vmdk를 가져온 로컬 디렉토리에 있어야 합니다. ovf 및 vmdk 파일 외에 인증서 및 매니페스트 파일도 다운로드하고 4개의 파일을 모두 동일한 소스 디렉토리에 배치해야 합니다.

12 OVA 파일을 로컬 컨텐츠 라이브러리로 가져옵니다.

a **메뉴 > 컨텐츠 라이브러리 >** 를 선택합니다.

b **컨텐츠 라이브러리** 목록에서 직접 생성한 로컬 컨텐츠 라이브러리의 이름 링크를 클릭합니다.

c **작업**을 클릭합니다.

d **항목 가져오기**를 선택합니다.

e **라이브러리 항목 가져오기** 창에서 **로컬 파일**를 선택합니다.

f **파일 업로드**를 클릭합니다.

g `photon-ova.ovf` 및 `photon-ova-disk1.vmdk` 파일을 둘 다 선택합니다.

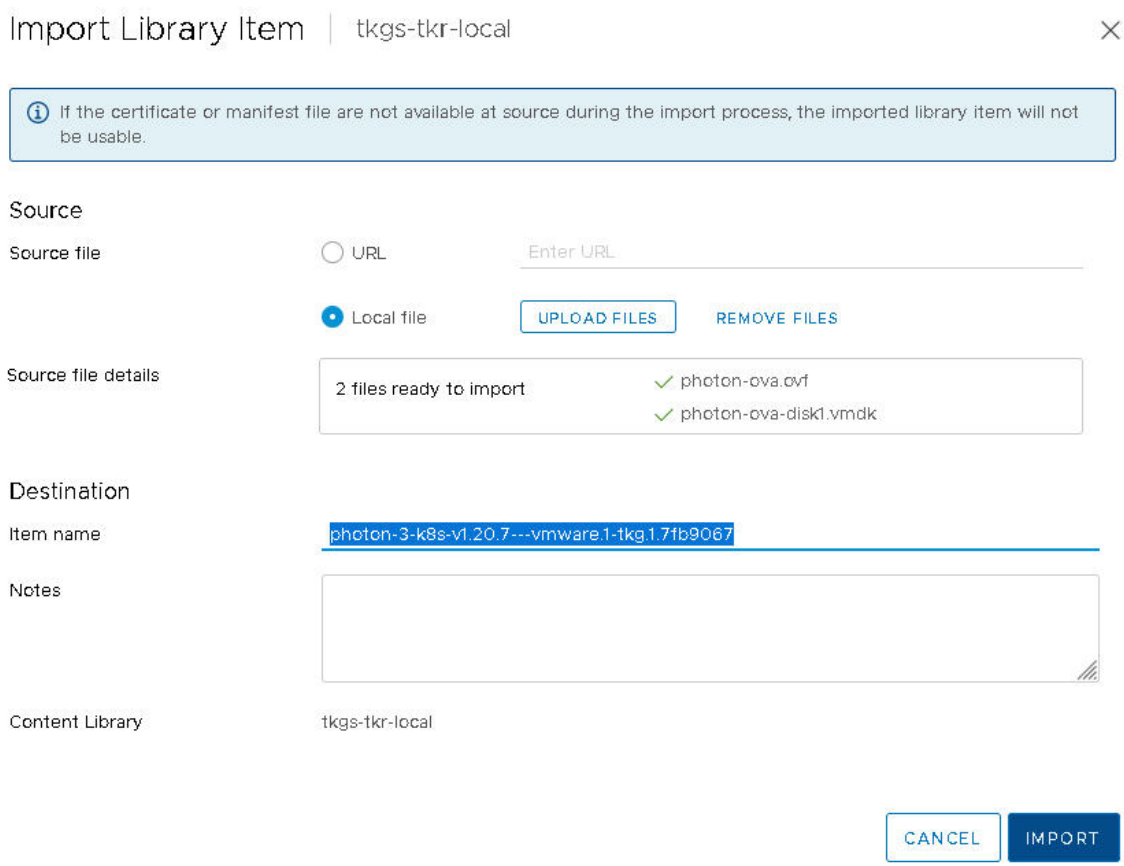
`2 files ready to import` 메시지가 표시됩니다. 각 파일의 이름 옆에 녹색 확인 표시가 나타납니다.

h **대상 항목** 이름을 Photon 이미지 버전과 파일을 다운로드한 디렉토리의 Kubernetes 버전으로 변경합니다.

예:

```
photon-3-k8s-v1.20.7---vmware.1-tkg.1.7fb9067
```

i **가져오기**를 클릭합니다.



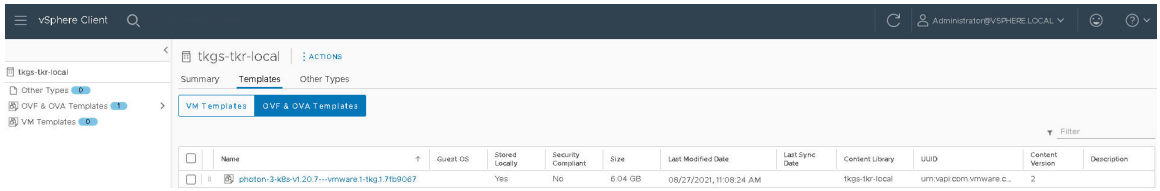
13 로컬 컨텐츠 라이브러리가 Tanzu Kubernetes 릴리스로 채워졌는지 확인합니다.

a 페이지 하단의 **최근 작업** 창을 나타냅니다.

b **라이브러리 항목의 컨텐츠 가져오기** 작업을 모니터링하고 성공적으로 **완료**되었는지 확인합니다.

c 로컬 컨텐츠 라이브러리에서 **템플릿 > OVF 및 OVA 템플릿**을 선택합니다.

d Tanzu Kubernetes 릴리스 메타데이터가 나열되고 해당 컨텐츠가 로컬에 저장되어 있는지 확인합니다.



다음에 수행할 작업

컨텐츠 라이브러리 및 가상 시스템 클래스를 네임스페이스와 연결하여 Tanzu Kubernetes 클러스터를 프로비저닝할 각 vSphere 네임스페이스를 구성합니다. Tanzu Kubernetes 릴리스에 대한 vSphere 네임스페이스 구성의 내용을 참조하십시오.

Tanzu Kubernetes 클러스터를 새 컨텐츠 라이브러리로 마이그레이션

구독 컨텐츠 라이브러리가 용량에 도달하면 Tanzu Kubernetes 클러스터를 마이그레이션하여 추가 스토리지 용량이 있는 새 라이브러리를 사용할 수 있습니다.

vSphere 관리자가 구독 컨텐츠 라이브러리를 생성할 때 관리자는 라이브러리 컨텐츠를 저장할 데이터스토어를 지정합니다(이 경우 OVA 파일). 더 많은 Kubernetes 버전이 배포되면 각 업데이트에 대해 OVA 파일이 추가되면서 구독 컨텐츠 라이브러리의 크기가 확장됩니다. 구독 컨텐츠 라이브러리에 명시적 용량은 없지만 해당 데이터스토어 용량에 의해 제한됩니다.

구독 컨텐츠 라이브러리가 용량에 도달하면 `Internal error occurred: get library items failed for.` 메시지가 표시될 수 있습니다. 이런 경우 Tanzu Kubernetes 클러스터를 새 구독 컨텐츠 라이브러리로 마이그레이션하여 스토리지 용량을 늘릴 수 있습니다. 마이그레이션은 vSphere 관리자가 vSphere Client를 사용하여 수행합니다.

절차

1 대상 클러스터에 충분한 용량을 가진 새 구독 컨텐츠 라이브러리를 생성합니다. Tanzu Kubernetes 릴리스용 구독 컨텐츠 라이브러리 생성, 보안 및 동기화의 내용을 참조하십시오.

2 vSphere Client를 사용하여 vCenter Server에 로그인합니다.

3 **메뉴 > 호스트 및 클러스터**를 선택합니다.

4 Tanzu Kubernetes 클러스터가 포함된 감독자 클러스터가 프로비저닝된 vSphere 클러스터 개체를 선택합니다.

5 **구성** 탭을 선택합니다.

6 탐색 패널에서 **네임스페이스 > 일반 >** 옵션을 선택합니다.

7 기본 패널의 **컨텐츠 라이브러리** 섹션 옆에 있는 **편집**을 클릭합니다.

8 생성한 새 컨텐츠 라이브러리를 선택하고 **확인**을 클릭합니다.

이 작업은 클러스터 구성에 대한 업데이트를 트리거합니다.

참고 컨텐츠 라이브러리를 수정한 후 Tanzu Kubernetes 클러스터가 컨텐츠 소스의 변경 내용을 적용하는 데 최대 10분이 소요될 수 있습니다.

HAProxy OVA를 로컬 컨텐츠 라이브러리로 가져오기

vDS 네트워킹을 사용하는 경우에는 편의상 HAProxy OVA 파일을 컨텐츠 라이브러리로 가져올 수 있습니다. HAProxy OVA를 로컬 컨텐츠 라이브러리로 가져오기는 에어갭 배포에 사용할 수 있습니다.

사전 요구 사항

로컬 컨텐츠 라이브러리를 생성합니다. Tanzu Kubernetes 릴리스용 로컬 컨텐츠 라이브러리 생성, 보안 및 동기화의 내용을 참조하십시오.

VMware-HAProxy 사이트에서 최신 버전의 VMware HAProxy OVA 파일을 다운로드합니다.

절차

1 vSphere Client를 사용하여 vCenter Server에 로그인합니다.

2 **메뉴 > 컨텐츠 라이브러리 >** 를 선택합니다.

3 **컨텐츠 라이브러리** 목록에서 직접 생성한 로컬 컨텐츠 라이브러리의 이름 링크를 클릭합니다(예: **HAProxy**).

4 **작업**을 클릭합니다.

5 **항목 가져오기**를 선택합니다.

6 **라이브러리 항목 가져오기** 창에서 **로컬 파일**를 선택합니다.

7 **파일 업로드**를 클릭합니다.

8 `vmware-haproxy-vX.X.X.ova` 파일을 선택합니다.

`1 file ready to import` 메시지가 표시됩니다. 파일이 나열되고 파일 이름 옆에 녹색 확인 표시가 있습니다.

9 **가져오기**를 클릭합니다.

10 페이지 하단의 **최근 작업** 창을 나타냅니다.

11 **라이브러리 항목의 컨텐츠 가져오기** 작업을 모니터링하고 성공적으로 **완료**되었는지 확인합니다.

다음에 수행할 작업

HAProxy 제어부 VM을 배포합니다. HAProxy 로드 밸런서 제어부 VM 배포의 내용을 참조하십시오.

vSphere with Tanzu에서 독립형 VM에 대한 컨텐츠 라이브러리 생성 및 관리

vSphere with Tanzu 환경의 독립형 VM에는 운영 체제, 애플리케이션 및 데이터를 포함한 소프트웨어 구성이 포함된 VM 이미지 또는 템플릿에 대한 액세스 권한이 필요합니다. 이미지에 대한 액세스를 제공하려면 VM 컨텐츠 라이브러리를 구성하고 VM이 배포된 네임스페이스와 연결합니다.

절차

1 vSphere with Tanzu에서 독립형 VM에 대한 컨텐츠 라이브러리 생성

vSphere with Tanzu 환경에서 가상 시스템을 배포하려면 DevOps 사용자가 VM 템플릿 및 이미지에 액세스할 수 있어야 합니다. vSphere 관리자는 컨텐츠 라이브러리를 생성하여 VM 템플릿을 저장하고 관리합니다.

2 vSphere with Tanzu에서 독립형 VM에 대한 VM 이미지로 컨텐츠 라이브러리 채우기

vSphere 관리자는 컨텐츠 라이브러리를 OVA 또는 OVF 형식의 VM 템플릿으로 채웁니다. DevOps 엔지니어는 이러한 템플릿을 사용하여 vSphere with Tanzu 환경에서 새 독립형 가상 시스템을 프로비저닝할 수 있습니다.

3 vSphere with Tanzu에서 VM 컨텐츠 라이브러리를 네임스페이스와 연결

vSphere 관리자는 DevOps 엔지니어가 템플릿을 사용하여 vSphere with Tanzu 환경에서 새로운 독립형 VM을 프로비저닝할 수 있도록 DevOps 사용자에게 VM 템플릿 소스에 대한 액세스 권한을 부여해야 합니다. 액세스 권한을 부여하려면 VM 템플릿이 포함된 컨텐츠 라이브러리를 네임스페이스에 추가합니다.

4 vSphere with Tanzu의 네임스페이스에서 VM 컨텐츠 라이브러리 관리

vSphere 관리자는 VM 템플릿이 포함된 컨텐츠 라이브러리를 네임스페이스와 연결합니다. 그러면 DevOps 엔지니어가 해당 템플릿을 사용하여 vSphere with Tanzu 환경에서 독립형 VM을 프로비저닝할 수 있습니다. 라이브러리를 네임스페이스와 연결한 후 라이브러리를 제거하여 Kubernetes 네임스페이스에서 게시를 취소할 수 있습니다. 라이브러리를 더 추가할 수도 있습니다.

vSphere with Tanzu에서 독립형 VM에 대한 컨텐츠 라이브러리 생성

vSphere with Tanzu 환경에서 가상 시스템을 배포하려면 DevOps 사용자가 VM 템플릿 및 이미지에 액세스할 수 있어야 합니다. vSphere 관리자는 컨텐츠 라이브러리를 생성하여 VM 템플릿을 저장하고 관리합니다.

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.

vSphere 7.0 업데이트 3부터는 OVF 보안 정책을 적용하여 컨텐츠 라이브러리의 항목을 보호할 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 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** 창에서 **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.

다음에 수행할 작업

컨텐츠 라이브러리를 생성한 후 라이브러리에 VM 템플릿을 채우면 DevOps 엔지니어가 템플릿을 사용하여 새 가상 시스템을 프로비저닝할 수 있습니다. vSphere with Tanzu에서 독립형 VM에 대한 VM 이미지로 컨텐츠 라이브러리 채우기의 내용을 참조하십시오.

vSphere with Tanzu에서 독립형 VM에 대한 VM 이미지로 컨텐츠 라이브러리 채우기

vSphere 관리자는 컨텐츠 라이브러리를 OVA 또는 OVF 형식의 VM 템플릿으로 채웁니다. DevOps 엔지니어는 이러한 템플릿을 사용하여 vSphere with Tanzu 환경에서 새 독립형 가상 시스템을 프로비저닝할 수 있습니다.

컨텐츠 라이브러리를 생성한 후에는 여러 가지 방법으로 항목을 채울 수 있습니다. 이 항목에서는 로컬 시스템이나 웹 서버에서 파일을 가져와서 로컬 컨텐츠 라이브러리에 항목을 추가하는 방법을 설명합니다. 컨텐츠 라이브러리를 채우는 다른 방법은 컨텐츠로 라이브러리 채우기를 참조하십시오.

사전 요구 사항

- VM 프로비저닝을 위한 컨텐츠 라이브러리를 생성합니다. vSphere with Tanzu에서 독립형 VM에 대한 컨텐츠 라이브러리 생성.
- VMware Cloud Marketplace에 OVF로 표시되는 호환되는 VM 이미지만 사용합니다. 호환되는 이미지를 찾으려면 VMware Cloud Marketplace 웹 사이트에서 **VM 서비스 이미지**를 검색합니다. CentOS용 VM 서비스 이미지의 예는 CentOS용 VM 서비스 이미지를 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.

다음에 수행할 작업

컨텐츠 라이브러리를 생성하여 VM 템플릿으로 채운 후에는, 라이브러리를 네임스페이스에 추가하고 DevOps 사용자에게 컨텐츠 라이브러리에 대한 액세스 권한을 부여합니다. vSphere with Tanzu에서 VM 컨텐츠 라이브러리를 네임스페이스와 연결의 내용을 참조하십시오.

vSphere with Tanzu에서 VM 컨텐츠 라이브러리를 네임스페이스와 연결

vSphere 관리자는 DevOps 엔지니어가 템플릿을 사용하여 vSphere with Tanzu 환경에서 새로운 독립형 VM을 프로비저닝할 수 있도록 DevOps 사용자에게 VM 템플릿 소스에 대한 액세스 권한을 부여해야 합니다. 액세스 권한을 부여하려면 VM 템플릿이 포함된 컨텐츠 라이브러리를 네임스페이스에 추가합니다.

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 Grid 서비스 컨텐츠 라이브러리는 Tanzu Kubernetes Grid 서비스 창에서 관리해야 합니다.

사전 요구 사항

- 컨텐츠 라이브러리를 생성합니다. vSphere with Tanzu에서 독립형 VM에 대한 컨텐츠 라이브러리 생성의 내용을 참조하십시오.
- 라이브러리를 VM 템플릿으로 채웁니다. vSphere with Tanzu에서 독립형 VM에 대한 VM 이미지로 컨텐츠 라이브러리 채우기의 내용을 참조하십시오.
- 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** 창에서 **컨텐츠 라이브러리 추가**를 클릭합니다.

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

결과

추가한 라이브러리의 컨텐츠는 Kubernetes 네임스페이스에서 VM 이미지로 사용할 수 있으며, DevOps에서 셀프 서비스 VM에 사용할 수 있습니다. vSphere with Tanzu에서 가상 시스템 배포의 내용을 참조하십시오.

다음에 수행할 작업

컨텐츠 라이브러리를 네임스페이스와 연결한 후, 컨텐츠 라이브러리를 더 추가하거나 라이브러리를 제거하여 Kubernetes 네임스페이스에서 게시를 취소할 수 있습니다. vSphere with Tanzu의 네임스페이스에서 VM 컨텐츠 라이브러리 관리의 내용을 참조하십시오.

vSphere with Tanzu의 네임스페이스에서 VM 컨텐츠 라이브러리 관리

vSphere 관리자는 VM 템플릿이 포함된 컨텐츠 라이브러리를 네임스페이스와 연결합니다. 그러면 DevOps 엔지니어가 해당 템플릿을 사용하여 vSphere with Tanzu 환경에서 독립형 VM을 프로비저닝할 수 있습니다. 라이브러리를 네임스페이스와 연결한 후 라이브러리를 제거하여 Kubernetes 네임스페이스에서 게시를 취소할 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 서비스 컨텐츠 라이브러리는 Tanzu Kubernetes Grid 서비스 창에서 관리해야 합니다.

사전 요구 사항

- 네임스페이스에 컨텐츠 라이브러리를 추가합니다. vSphere with Tanzu에서 VM 컨텐츠 라이브러리를 네임스페이스와 연결.
- 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** 창에서 **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 컨텐츠를 Kubernetes 네임스페이스에서 VM 이미지로 사용할 수 있으며, DevOps에서 셀프 서비스 VM에 사용할 수 있습니다. vSphere with Tanzu에서 가상 시스템 배포의 내용을 참조하십시오.

vSphere 네임스페이스 구성 및 관리 7

vSphere 포드, VM 및 Tanzu Kubernetes 클러스터를 포함하는 vSphere with Tanzu 워크로드는 vSphere 네임스페이스에 배포됩니다. 감독자 클러스터에서 vSphere 네임스페이스를 정의하고 리소스 할당량 및 사용자 사용 권한으로 구성합니다. DevOps의 요구 사항 및 실행할 계획인 워크로드에 따라 최신 Tanzu Kubernetes 릴리스 및 VM 이미지를 가져오기 위해 스토리지 정책, VM 클래스 및 컨텐츠 라이브러리를 할당할 수도 있습니다.

본 장은 다음 항목을 포함합니다.

- vSphere 네임스페이스 생성 및 구성
- vSphere 포드 컨테이너에 대한 기본 메모리 및 CPU 예약 및 제한 설정
- vSphere 네임스페이스의 Kubernetes 개체에 대한 제한 사항 구성
- vSphere 네임스페이스에서 리소스 모니터링 및 관리
- Tanzu Kubernetes 릴리스에 대한 vSphere 네임스페이스 구성
- NSX 감독자 클러스터 네임스페이스에 보안 정책 추가
- 셀프 서비스 네임스페이스 템플릿 프로비저닝

vSphere 네임스페이스 생성 및 구성

vSphere 관리자는 감독자 클러스터에서 vSphere 네임스페이스를 생성합니다. DevOps 엔지니어가 액세스할 수 있도록 사용 권한 및 네임스페이스에 대한 리소스 제한을 설정합니다. 사용 권한이 있는 네임스페이스에서 Kubernetes 워크로드를 실행할 수 있는 Kubernetes 제어부의 URL을 DevOps 엔지니어에게 제공합니다.

vSphere 네트워킹 스택으로 구성된 감독자 클러스터의 네임스페이스와 NSX-T Data Center로 구성된 클러스터의 네임스페이스는 네트워킹 구성 및 기능이 서로 다릅니다.

NSX-T Data Center로 구성된 감독자 클러스터에서 생성하는 네임스페이스는 워크로드 관리 플랫폼의 전체 기능 집합을 지원합니다. vSphere 포드, VM 및 Tanzu Kubernetes 클러스터를 지원합니다. 이러한 네임스페이스에 대한 워크로드 네트워킹 지원은 NSX-T Data Center에서 제공합니다. 자세한 내용은 NSX-T Data Center를 사용하여 vSphere with Tanzu를 설정하기 위한 시스템 요구 사항의 내용을 참조하십시오.

vSphere 네트워킹 스택으로 구성된 감독자 클러스터에서 생성하는 네임스페이스는 Tanzu Kubernetes 클러스터와 VM만 지원하고 vSphere 포드는 지원하지 않으며 Harbor 레지스트리를 함께 사용할 수 없습니다. 이러한 네임스페이스에 대한 워크로드 네트워킹 지원은 감독자 클러스터의 호스트에 연결된 vSphere Distributed Switch에서 제공됩니다. 자세한 내용은 vSphere 네트워킹 및 HAProxy 로드 밸런서를 사용하여 vSphere with Tanzu를 설정하기 위한 시스템 요구 사항의 내용을 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 셀프 서비스 네임스페이스 템플릿 프로비저닝의 내용을 참조하십시오.

사전 요구 사항

- vSphere with Tanzu으로 클러스터를 구성합니다.
- 네임스페이스에 액세스하는 모든 DevOps 엔지니어를 위한 사용자 또는 그룹을 생성합니다.
- 영구 스토리지에 대한 스토리지 정책을 생성합니다. 스토리지 정책은 다양한 스토리지 유형과 클래스를 정의할 수 있습니다(예: Gold, Silver 및 Bronze).
- 독립형 VM에 대한 VM 클래스 및 컨텐츠 라이브러리를 생성합니다.
- Tanzu Kubernetes 클러스터에서 사용할 Tanzu Kubernetes 릴리스용 컨텐츠 라이브러리를 생성합니다. Tanzu Kubernetes 릴리스용 컨텐츠 라이브러리 생성 및 관리의 내용을 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 선택합니다.

3 **네임스페이스 생성**을 클릭합니다.

4 네임스페이스를 배치할 감독자 클러스터를 선택합니다.

5 네임스페이스의 이름을 입력합니다.

이름은 DNS 호환 형식이어야 합니다.

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

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

7 클러스터에 대한 NSX-T Data Center 네트워킹 스택을 구성한 경우 **클러스터 네트워크 설정 재정의**를 선택하여 클러스터 네트워크 설정을 재정의하고 네임스페이스에 대한 네트워크 설정을 구성할 수 있습니다.

네임스페이스에 대해 다음 네트워크 설정을 구성합니다.

옵션	설명
NAT 모드	NAT 모드는 기본적으로 선택되어 있습니다. 이 옵션을 선택 취소하면 vSphere 포드, VM, Tanzu Kubernetes 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.
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. Tier-0 게이트웨이에 연결된 VRF 게이트웨이를 선택하면 네트워크와 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 선택한 후에는 변경할 수 없습니다.
네임스페이스 네트워크 CIDR	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.
네임스페이스 서브넷 접두사	네임스페이스 세그먼트용으로 예약된 서브넷의 크기를 지정하는 서브넷 접두사를 입력합니다. Default is 28. **참고** 서브넷 접두사를 지정한 후에는 변경할 수 없습니다.
수신 CIDR	vSphere 포드 또는 Tanzu Kubernetes 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 CIDR	SNAT IP 주소의 송신 IP 범위를 결정하는 CIDR 주석을 입력합니다. 네임스페이스를 생성한 후 네임스페이스 네트워크 설정을 편집하여 추가 CIDR을 구성할 수 있습니다.
로드 밸런서 크기	네임스페이스의 Tier-1 게이트웨이에서 로드 밸런서 인스턴스의 크기를 선택합니다.

8 설명을 입력하고 **생성**을 클릭합니다.

감독자 클러스터에 네임스페이스가 생성됩니다.

9 DevOps 엔지니어가 네임스페이스에 액세스할 수 있도록 사용 권한을 설정합니다.

a **사용 권한** 창에서 **사용 권한 추가**를 선택합니다.

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

10 네임스페이스에 대한 영구 스토리지를 설정합니다.

네임스페이스에 할당하는 스토리지 정책은 영구 볼륨과 Tanzu Kubernetes 클러스터 노드가 vSphere 스토리지 환경의 데이터스토어 내에 배치되는 방식을 제어합니다. 영구 볼륨에 해당하는 영구 볼륨 할당은 vSphere 포드 및 VM 또는 Tanzu Kubernetes 클러스터에서 발생할 수 있습니다. 자세한 내용은 장 10 vSphere with Tanzu에서 영구 스토리지 사용의 내용을 참조하십시오.

a **스토리지** 창에서 **스토리지 추가**를 선택합니다.

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

스토리지 정책을 할당하면 vSphere with Tanzu은 vSphere 네임스페이스에 일치하는 Kubernetes 스토리지 클래스를 생성합니다. VMware Tanzu™ Kubernetes Grid™ 서비스를 사용하는 경우 스토리지 클래스는 네임스페이스에서 Kubernetes 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 12 vSphere with Tanzu에서 가상 시스템 배포 및 관리 항목을 참조하십시오.

13 다음을 포함하여 Tanzu Kubernetes 클러스터에 대한 네임스페이스를 구성합니다.

- Tanzu Kubernetes 릴리스 컨텐츠 라이브러리를 네임스페이스와 연결합니다.
- 네임스페이스에 기본 VM 클래스를 추가합니다.

자세한 내용은 Tanzu Kubernetes 릴리스에 대한 vSphere 네임스페이스 구성의 내용을 참조하십시오.

다음에 수행할 작업

Kubernetes 제어부 URL을 DevOps 엔지니어와 공유하고 vSphere에 대한 Kubernetes CLI 도구을 통해 감독자 클러스터에 로그인하는 데 사용할 수 있는 사용자 이름도 공유합니다. DevOps 엔지니어에게 둘 이상의 네임스페이스에 대한 액세스 권한을 부여할 수 있습니다. 장 9 vSphere with Tanzu 클러스터에 연결의 내용을 참조하십시오.

vSphere 포드 컨테이너에 대한 기본 메모리 및 CPU 예약 및 제한 설정

vSphere Client를 통해 네임스페이스의 컨테이너에 대한 기본 메모리 및 CPU 예약 및 제한을 설정할 수 있습니다. DevOps 엔지니어가 정의한 포드 규격에서 이러한 값을 나중에 재정의할 수 있습니다. 컨테이너 요청은 vSphere 포드의 리소스 예약으로 변환됩니다.

사전 요구 사항

- 감독자 클러스터에서 **네임스페이스 구성 수정** 권한이 있는지 확인합니다.

절차

1. vSphere Client 홈 메뉴에서 **워크로드 관리**를 선택합니다.
2. 네임스페이스를 선택하고 **구성**을 선택하고 **리소스 제한**을 클릭합니다.
3. 네임스페이스에서 컨테이너에 대한기본 메모리와 CPU 예약 및 제한을 구성합니다.

옵션	설명
CPU 요청	네임스페이스의 컨테이너에 대한 기본 CPU 예약을 설정합니다.
CPU 제한	네임스페이스에서 컨테이너의 CPU 사용량에 대한기본 제한을 설정합니다.
메모리 요청	네임스페이스에서 컨테이너의 기본 메모리 예약을 설정합니다.
메모리 제한	네임스페이스에서 컨테이너의 메모리 사용량에 대한 기본 제한을 설정합니다.

vSphere 네임스페이스의 Kubernetes 개체에 대한 제한 사항 구성

vSphere 네임스페이스에서 실행되는 포드에 대한 제한 사항은 물론 다양한 Kubernetes 개체에 대한 제한 사항을 구성할 수 있습니다. 개체에 대해 구성하는 제한 사항은 애플리케이션의 세부 사항 및 애플리케이션이 vSphere 네임스페이스 내에서 리소스를 사용하는 방식에 따라 다릅니다.

사전 요구 사항

- 감독자 클러스터에서 **네임스페이스 구성 수정** 권한이 있는지 확인합니다.

절차

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

2 개체 또는 컨테이너 제한 사항을 적용할 네임스페이스를 선택합니다.

3 컨테이너 제한을 설정하려면 **리소스 제한**을 선택하고 **편집**을 클릭합니다.

옵션	설명
CPU 요청	컨테이너에 대한 CPU 요청량을 설정합니다.
CPU 제한	컨테이너에서 사용할 수 있는 CPU 양을 설정합니다.
메모리 요청	컨테이너에 대한 메모리 요청량을 설정합니다.
메모리 제한	컨테이너가 사용할 수 있는 메모리의 양을 설정합니다.

참고 Tanzu Kubernetes 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. Tanzu Kubernetes 클러스터에 대한 가상 시스템 클래스의 내용을 참조하십시오.

4 네임스페이스에 있을 수 있는 Kubernetes 개체에 대한 제한을 설정하려면 **개체 제한**을 선택하고 **편집**을 클릭합니다.

옵션	설명
포드	네임스페이스에서 실행할 수 있는 vSphere 포드의 수입니다.
배포	네임스페이스에서 실행할 수 있는 배포의 수입니다.
작업	네임스페이스에서 실행할 수 있는 작업 수입니다.
DaemonSets	네임스페이스에서 실행할 수 있는 데몬 집합의 수입니다.
ReplicaSets	네임스페이스의 복제 집합 수입니다.
ReplicationControllers	네임스페이스에서 실행할 수 있는 복제 컨트롤러의 수입니다.
StatefulSets	네임스페이스에서 실행할 수 있는 StatefulSet의 수입니다.
ConfigMaps	네임스페이스에서 실행할 수 있는 ConfigMap의 수입니다.
암호	네임스페이스에서 실행할 수 있는 암호의 수입니다.
영구 볼륨 할당	네임스페이스에 존재할 수 있는 영구 볼륨 할당입니다.
서비스	네임스페이스에 있을 수 있는 서비스입니다.

vSphere 네임스페이스에서 리소스 모니터링 및 관리

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

사전 요구 사항

vSphere 네임스페이스 생성 및 구성.

절차

1 vSphere Client를 사용하여 vCenter Server에 로그인합니다.

2 **메뉴 > 호스트 및 클러스터** 보기로 이동합니다.

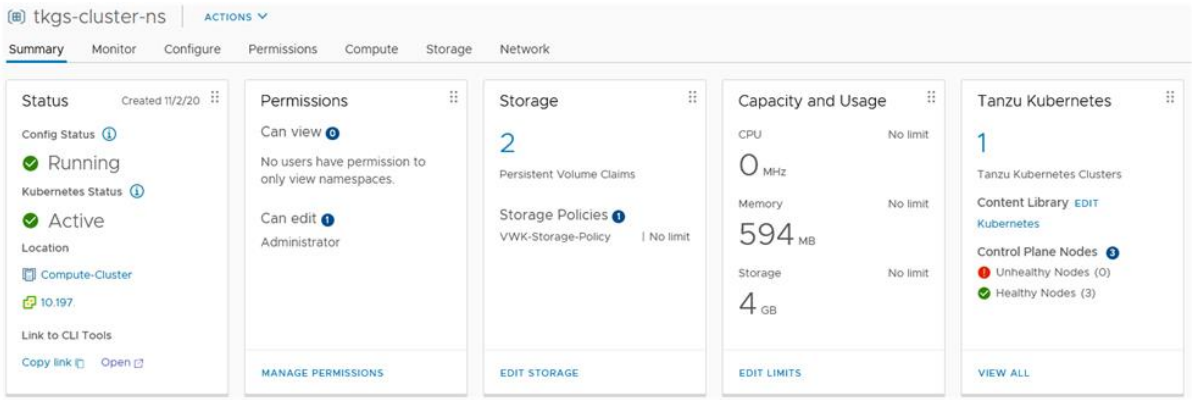
3 **워크로드 관리**를 사용하도록 설정한 vCenter 클러스터를 선택합니다.

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

감독자 클러스터 제어부 노드는 네임스페이스 리소스 풀에 있습니다. 또한 이 감독자 클러스터에 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.



Tanzu Kubernetes 릴리스에 대한 vSphere 네임스페이스 구성

네임스페이스를 Tanzu Kubernetes 릴리스용 컨텐츠 라이브러리 및 사용하려는 VM 클래스와 연결하여 Tanzu Kubernetes 클러스터를 프로비저닝할 vSphere 네임스페이스를 구성합니다.

사전 요구 사항

vSphere 네임스페이스를 생성합니다. vSphere 네임스페이스 생성 및 구성의 내용을 참조하십시오.

Tanzu Kubernetes 릴리스를 호스팅할 컨텐츠 라이브러리를 생성합니다. Tanzu Kubernetes 릴리스용 구독 컨텐츠 라이브러리 생성, 보안 및 동기화 또는 Tanzu Kubernetes 릴리스용 로컬 컨텐츠 라이브러리 생성, 보안 및 동기화의 내용을 참조하십시오.

vSphere 네임스페이스와 컨텐츠 라이브러리 연결

Tanzu Kubernetes 릴리스에 대해 생성된 컨텐츠 라이브러리를 vSphere 네임스페이스와 연결하려면 vSphere Client를 사용하여 vCenter Server에 로그인하고 다음 절차 중 하나를 완료합니다.

vSphere 인벤토리 경로를 사용하여 연결	워크로드 관리 경로를 사용하여 연결
1 **메뉴 > 호스트 및 클러스터**를 선택합니다. 2 **워크로드 관리**를 사용하도록 설정된 vSphere 클러스터를 선택합니다. 3 **구성** 탭을 선택합니다. 4 **네임스페이스 > 일반**을 선택합니다. 5 **Tanzu Kubernetes Grid 서비스 구성**을 선택합니다. 6 **컨텐츠 라이브러리** 레이블 옆의 **편집**을 클릭합니다. 7 Tanzu Kubernetes 릴리스용 컨텐츠 라이브러리를 선택합니다. 8 **확인**을 클릭합니다.	1 **메뉴 > 워크로드 관리**를 선택합니다. 2 **네임스페이스** 탭을 선택합니다. 3 대상 vSphere 네임스페이스를 선택합니다. 4 **Tanzu Kubernetes Grid 서비스** 타일을 찾습니다. 5 **컨텐츠 라이브러리** 레이블 옆의 **편집**을 클릭합니다. 6 Tanzu Kubernetes 릴리스용 컨텐츠 라이브러리를 선택합니다. 7 **확인**을 클릭합니다.

참고 컨텐츠 라이브러리를 vSphere 네임스페이스와 연결한 후 몇 분 정도 지나야 가상 시스템 템플릿을 Tanzu Kubernetes 클러스터 프로비저닝에 사용할 수 있습니다. vSphere 네임스페이스 구성 확인의 내용을 참조하십시오.

VM 클래스를 vSphere 네임스페이스와 연결

vSphere with Tanzu는 몇 가지 기본 가상 시스템 클래스를 제공하며, 직접 생성할 수 있습니다. Tanzu Kubernetes 클러스터에 대한 가상 시스템 클래스의 내용을 참조하십시오.

Tanzu Kubernetes 클러스터를 프로비저닝하려면 사용하려는 가상 시스템 클래스를 Tanzu Kubernetes 클러스터를 프로비저닝하려는 각 vSphere 네임스페이스와 연결해야 합니다.

기본 VM 클래스를 vSphere 네임스페이스와 연결하려면 vSphere Client를 사용하여 vCenter Server에 로그인하고 다음 절차를 완료합니다.

1 **메뉴 > 워크로드 관리**를 선택합니다.

2 **네임스페이스** 탭을 선택합니다.

3 Tanzu Kubernetes 클러스터를 프로비저닝할 대상 vSphere 네임스페이스를 선택합니다.

4 **VM 서비스** 타일을 찾습니다.

5 **VM 클래스 추가** 링크를 클릭합니다.

6 추가할 VM 클래스를 선택합니다.

   a 기본 VM 클래스를 추가하려면 목록의 1페이지에서 테이블 헤더의 확인란을 선택하고 2페이지로 이동한 후 해당 페이지의 테이블 헤더에 있는 확인란을 선택합니다. 모든 클래스가 선택되었는지 확인합니다.

   b 사용자 지정 클래스를 생성하려면 **새 VM 클래스 생성**을 클릭합니다. vSphere with Tanzu에서 VM 클래스 생성의 내용을 참조하십시오.

7 **확인**을 클릭하여 작업을 완료합니다.

8 클래스가 추가되었는지 확인합니다. **VM 서비스** 타일에는 **VM 클래스 관리**가 표시됩니다.

참고 **VM 서비스** 타일에 참조된 컨텐츠 라이브러리는 Tanzu Kubernetes 릴리스가 아닌 독립형 VM에 사용하기 위한 것입니다. vSphere with Tanzu에서 독립형 VM에 대한 컨텐츠 라이브러리 생성 및 관리의 내용을 참조하십시오.

vSphere 네임스페이스 구성 확인

컨텐츠 라이브러리 및 가상 시스템 클래스를 vSphere 네임스페이스와 연결한 후에는 감독자 클러스터에 로그인하여 동기화된 각 Tanzu Kubernetes 릴리스를 사용할 수 있는지 그리고 선택한 각 VM 클래스를 사용할 수 있는지 확인합니다.

1 vSphere에 대한 Kubernetes CLI 도구를 설치합니다. vSphere에 대한 Kubernetes CLI 도구 다운로드 및 설치의 내용을 참조하십시오.

2 감독자 클러스터에 로그인합니다.

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

3 컨텍스트를 대상 vSphere 네임스페이스로 전환합니다.

```
kubectl config use-context SUPERVISOR-NAMESPACE
```

4 사용 가능한 Tanzu Kubernetes 릴리스를 나열하고 설명합니다.

```
kubectl get tanzukubernetesreleases
```

```
kubectl describe tanzukubernetesreleases
```

5 사용 가능한 가상 시스템 클래스를 나열합니다.

```
kubectl get virtualmachineclassbindings
```

네임스페이스가 구성되면 이제 Tanzu Kubernetes 클러스터를 프로비저닝할 수 있습니다. TKGS v1alpha2 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하는 워크플로의 내용을 참조하십시오. 로컬 컨텐츠 라이브러리를 사용하는 경우에는 라이브러리에 업로드한 OVA를 지정해야 합니다. Tanzu Kubernetes Grid 서비스 v1alpha1 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하는 예의 내용을 참조하십시오.

NSX 감독자 클러스터 네임스페이스에 보안 정책 추가

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

보안 정책 생성

DevOps는 NSX 기반 보안 정책을 감독자 클러스터 네임스페이스에 적용하도록 보안 정책 CRD를 구성할 수 있습니다. 보안 정책은 vSphere 포드 및 VM에 대한 트래픽을 보호합니다. VM에는 TKG 클러스터 노드 및 감독자 클러스터에 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 클러스터 컨텍스트 가져오기 및 사용의 내용을 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.

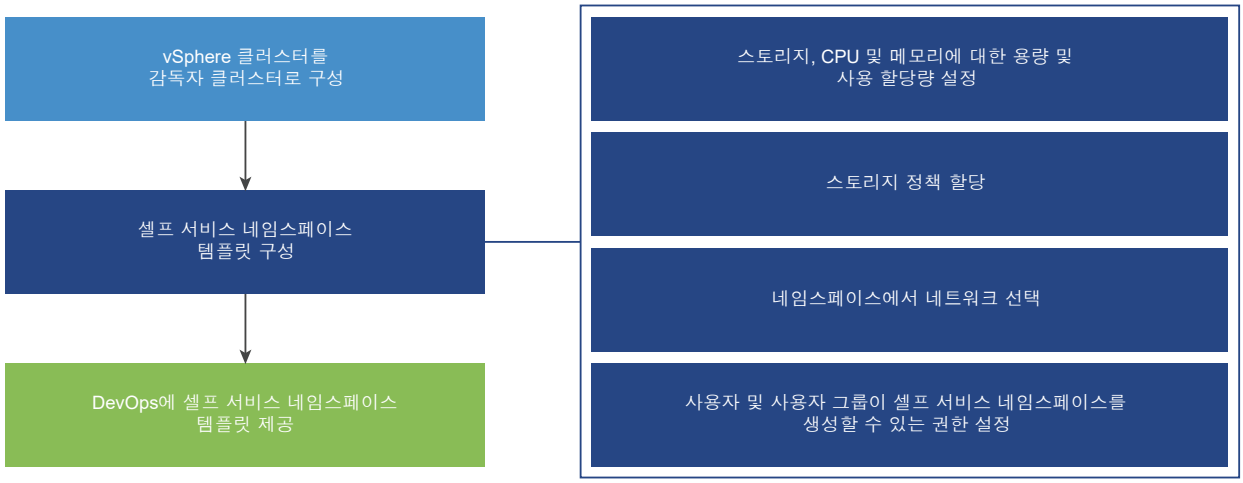
셀프 서비스 네임스페이스 템플릿 프로비저닝

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

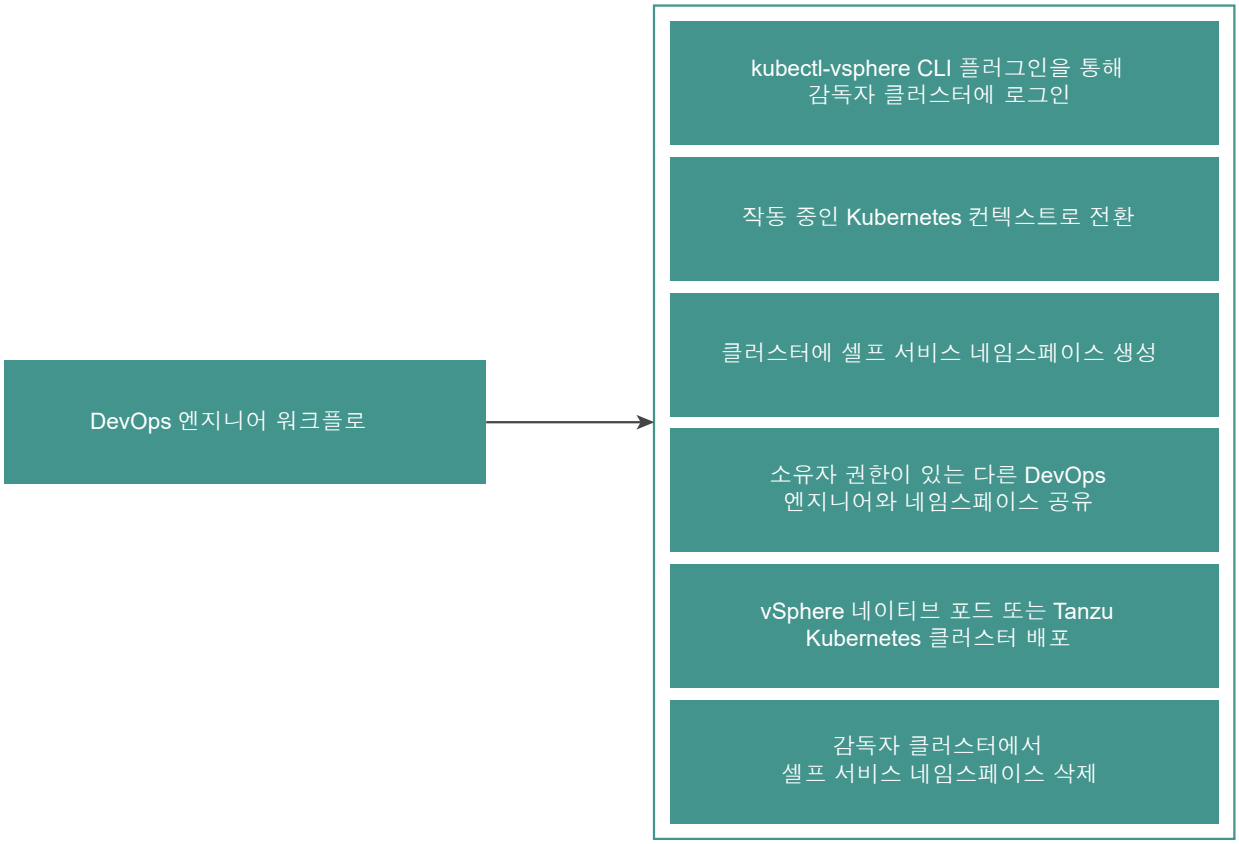
vSphere 관리자는 감독자 네임스페이스를 생성하고, CPU, 메모리 및 스토리지 제한을 네임스페이스에 설정하고, 사용 권한을 할당하고, 클러스터에서 네임스페이스 서비스를 템플릿으로 프로비저닝 또는 활성화할 수 있습니다.

그림 7-1. 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 7-2. 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 with Tanzu으로 클러스터를 구성합니다.

절차

1 vSphere Client에서 감독자 클러스터를 사용하도록 설정된 vSphere 클러스터를 선택합니다.

2 **구성** 탭에서 **네임스페이스 > 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 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	네임스페이스와 연결한 각 스토리지 정책에 개별적으로 전용 스토리지 양을 설정합니다.
네트워크	**네트워크** 드롭다운 메뉴에서 네임스페이스에 대한 네트워크를 선택합니다.

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 감독자 클러스터로 이동합니다.

2 **구성** 탭의 **네임스페이스**에서 **일반**을 선택합니다.

3 **네임스페이스 셀프 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 클러스터 제어부의 IP 주소를 가져옵니다.

절차

1 kubectl용 vSphere 플러그인을 사용하여 감독자 클러스터로 인증합니다. vCenter Single Sign-On 사용자로 감독자 클러스터에 연결의 내용을 참조하십시오.

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

2 컨텍스트를 감독자 클러스터로 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 with Tanzu를 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 감독자 클러스터에 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 감독자 클러스터에 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 감독자 클러스터에 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 감독자 클러스터로 인증합니다. vCenter Single Sign-On 사용자로 감독자 클러스터에 연결의 내용을 참조하십시오.

2 클러스터에서 셀프 서비스 네임스페이스를 삭제합니다.

```
kubectl delete namespace NAMESPACE NAME
```

예:

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

vSphere with Tanzu를 사용하여 감독자 서비스 관리

8

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 Kubernetes 워크로드에서 사용할 수 있도록 vSphere with Tanzu 환경에서 설치하고 관리할 수 있습니다. 감독자 서비스가 감독자 클러스터에 설치되면 DevOps 엔지니어는 서비스 API 를 사용하여 사용자 네임스페이스의 감독자 클러스터에 인스턴스를 생성할 수 있습니다. 이러한 인스턴스를 vSphere 포드 및 Tanzu Kubernetes 클러스터에서 사용할 수 있습니다.

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

vSphere Client에서 vSphere Services 플랫폼의 감독자 서비스를 관리합니다. 플랫폼을 사용하여 감독자 서비스의 수명 주기를 관리하고, 감독자 클러스터에 설치하고, 버전 제어를 수행할 수 있습니다. 감독자 서비스에는 감독자 클러스터에 설치할 수 있는 여러 버전이 있을 수 있으며, 감독자 클러스터에서는 한 번에 하나의 버전만 실행할 수 있습니다.

표 8-1. 감독자 서비스 상태

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

감독자 서비스 수명 주기 관리 작업

감독자 서비스의 수명 주기 관리에는 다음 작업이 포함됩니다.

- vCenter Server에 새 감독자 서비스 추가. 새 서비스를 vCenter Server에 추가하면 서비스 및 서비스에 대한 모든 정보가 vCenter Server에 등록됩니다. 서비스가 아직 감독자 클러스터에 설치되지 않았습니다. 서비스가 vCenter Server에 등록되면 활성 상태가 됩니다. 그러면 해당 서비스를 감독자 클러스터에 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 감독자 클러스터에 설치할 수 있습니다.
- 감독자 클러스터에 감독자 서비스 설치. 감독자 클러스터에 서비스를 설치하면 서비스 YAML 파일이 클러스터에 적용되고 서비스가 작동하기 위해 필요한 리소스와 모든 포드가 생성됩니다. 감독자 클러스터에 설치하는 각 서비스에는 서비스 리소스를 관리할 수 있는 전용 네임스페이스가 있습니다. 감독자 서비스에는 서비스 구성을 관리할 수 있는 vCenter Server용 UI 플러그인이 있을 수도 있습니다.
- 감독자 서비스 업그레이드. 먼저 vCenter Server에 새 서비스 버전을 추가한 다음 감독자 클러스터에 새 버전을 설치하여 감독자 클러스터에 설치된 서비스를 업그레이드할 수 있습니다. 서비스 업그레이드 중에 새 버전의 YAML 파일이 감독자 클러스터에 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. 감독자 클러스터에서 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 감독자 클러스터에서 제거해야 합니다. 그런 다음 vCenter Server에서 서비스 버전을 삭제할 수 있습니다.
- 전체 감독자 서비스 삭제. 전체 서비스를 삭제하려면 해당 버전을 모두 비활성화한 다음 감독자 클러스터에서 이러한 버전을 제거하고 최종적으로 모든 서비스 버전을 삭제해야 합니다.

본 장은 다음 항목을 포함합니다.

vCenter Server에 감독자 서비스 추가

vSphere with Tanzu 환경이 실행되는 vCenter Server 시스템에 감독자 서비스를 추가할 수 있습니다. vCenter Server에 서비스를 추가한 후 DevOps 엔지니어가 Kubernetes 워크로드에서 서비스를 사용할 수 있도록 감독자 클러스터에 감독자 서비스를 설치합니다.

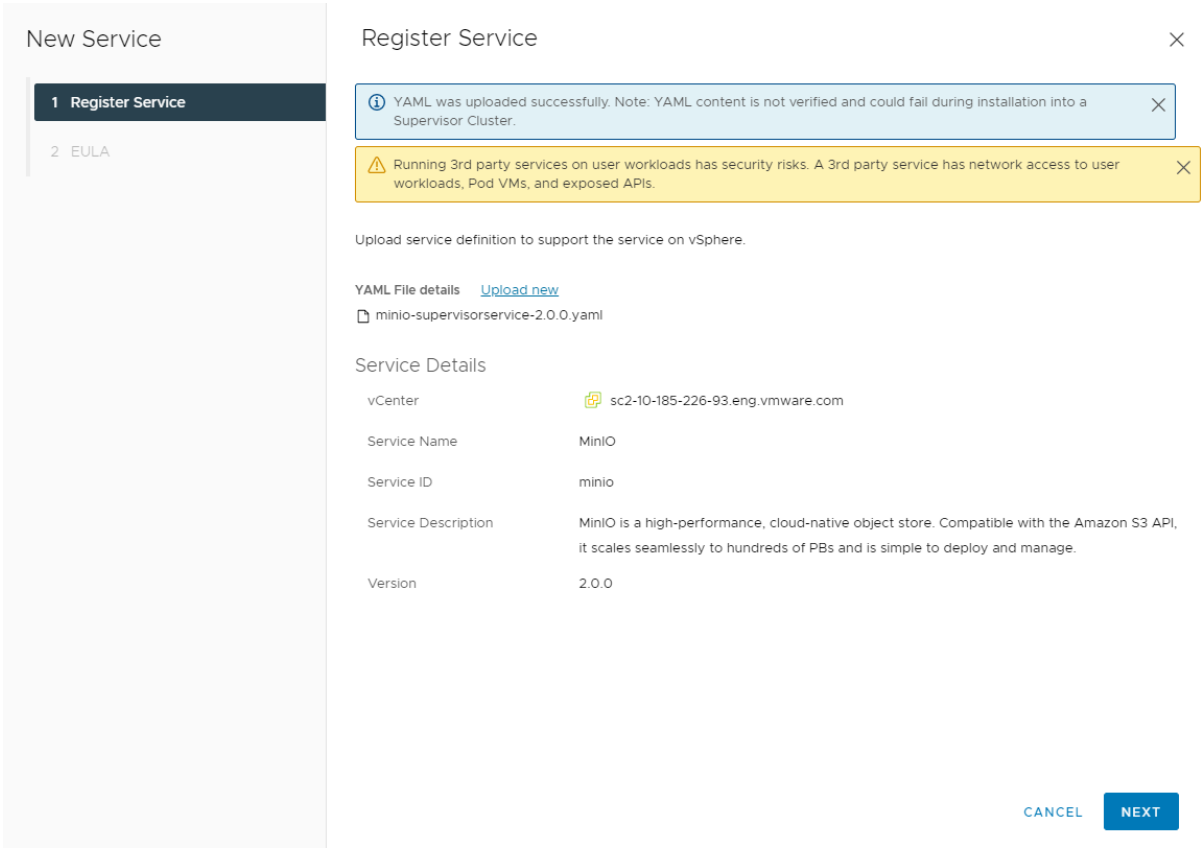
- 지원되는 감독자 서비스에 대해 자세한 내용 및 해당 서비스 YAML 파일을 다운로드하는 방법은 http://vmware.com/go/supervisor-service에서 참조하십시오.

사전 요구 사항

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

절차

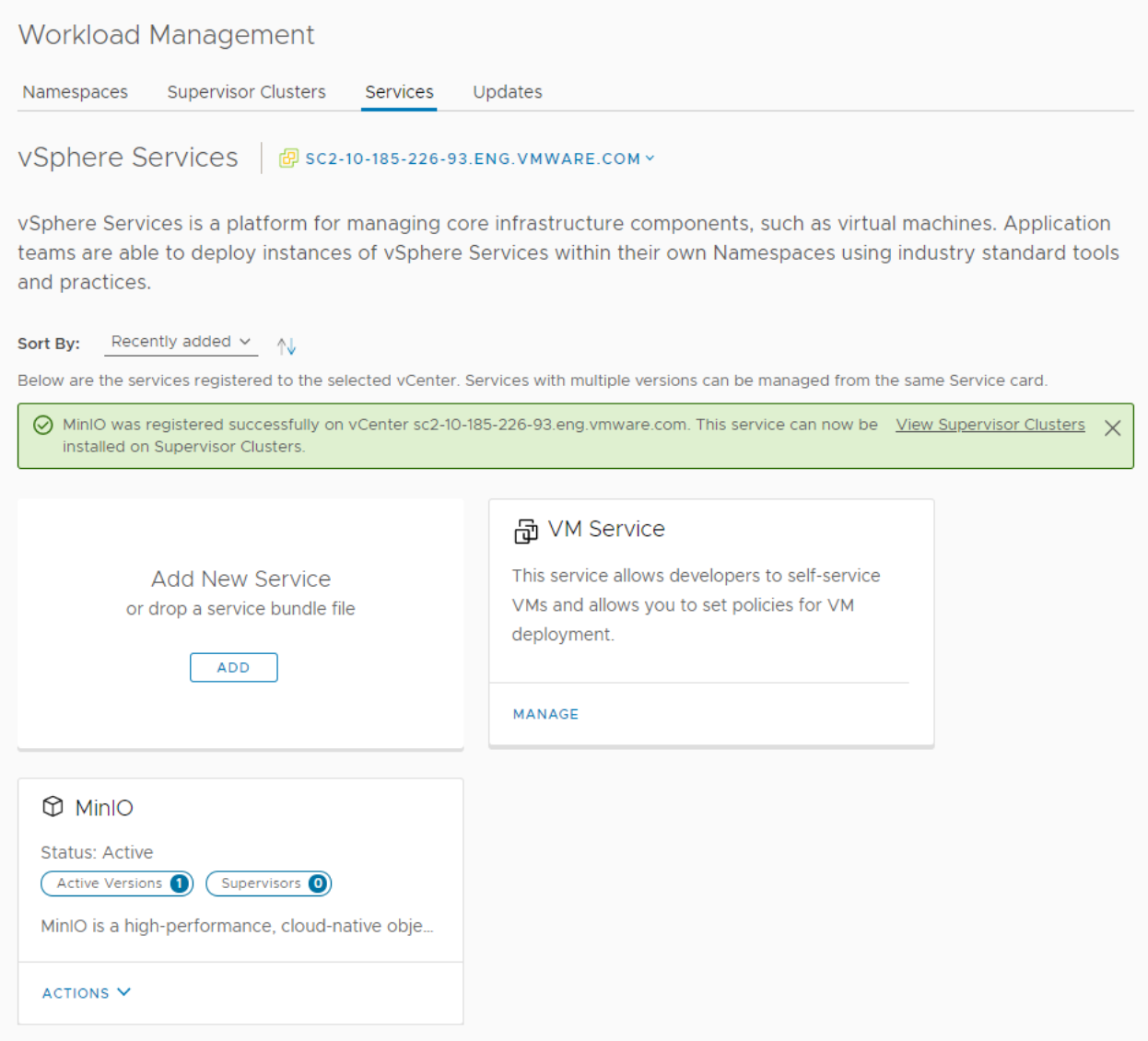
1. vSphere Client 홈 메뉴에서 **워크로드 관리**를 선택합니다.
2. **서비스**를 선택합니다.
3. 맨 위에 있는 드롭다운 메뉴에서 vCenter Server 시스템을 선택합니다.
4. **새 서비스 추가** 카드에서 서비스 YAML 파일을 끌어서 놓습니다.



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 감독자 클러스터에 감독자 서비스를 설치합니다. 감독자 클러스터에 감독자 서비스 설치의 내용을 참조하십시오.

감독자 클러스터에 감독자 서비스 설치

감독자 서비스를 vCenter Server에 추가한 후 vSphere with Tanzu 환경의 감독자 클러스터에 설치할 수 있습니다. 최신 버전의 감독자 서비스를 설치하면 해당 감독자 클러스터의 이전 서비스 버전이 재정의됩니다. 감독자 클러스터에서 한 번에 하나의 감독자 클러스터 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 추가 항목을 참조하십시오.
- 서비스를 설치하는 감독자 클러스터를 호스팅하는 vCenter Server 시스템에 대한 **감독자 서비스 관리** 권한이 있는지 확인합니다.
- 감독자 서비스에 영구 스토리지가 필요한 경우 vSAN 데이터 지속성 플랫폼을 구성합니다. 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 서비스 버전을 선택합니다.

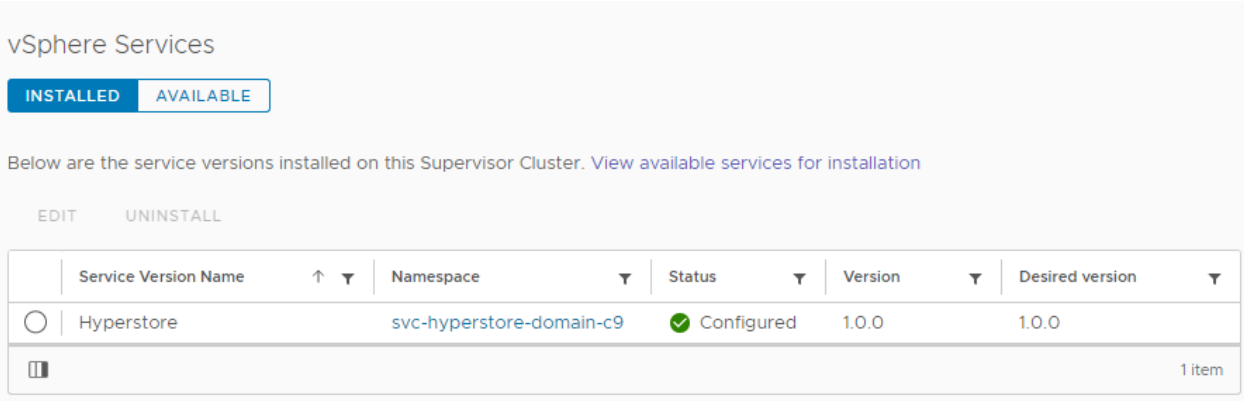
   참고 비활성화된 감독자 서비스 버전은 설치할 수 없습니다.

4. 서비스를 설치할 감독자 클러스터를 선택합니다.
5. 개인 레지스트리에서 이미지를 호스팅한 경우 레지스트리 끝점, 사용자 이름 및 암호를 입력합니다.

   서비스에 필요한 경우 다른 키-값 쌍이 존재할 수 있습니다. 자세한 내용은 서비스 설명서를 참조하십시오.

결과

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



참고 감독자 클러스터는 감독자 서비스가 실제로 작동하는지 모니터링하지 않습니다. 예를 들어, 레지스트리 사용자 이름 또는 암호를 잘못 지정하면 서비스가 실행해야 하는 이미지를 가져오지 못할 수 있습니다. 이런 경우, 서비스가 [구성됨]으로 표시될 수 있지만 실제로는 작동하지 않습니다. 감독자 서비스가 작동하는지 여부를 확인하려면 서비스 네임스페이스 및 실행 중인 포드를 확인합니다.

다음에 수행할 작업

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

감독자 클러스터에서 감독자 서비스의 관리 인터페이스에 액세스

감독자 클러스터에 감독자 서비스를 설치한 후 관리 UI를 찾을 수 있는 위치를 알아봅니다. 감독자 서비스는 vSphere Client의 감독자 클러스터 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 Client에서 감독자 클러스터로 이동합니다.

2 **구성**을 선택하고 서비스 인터페이스까지 아래로 스크롤합니다. 서비스 인터페이스는 일반적으로 서비스 이름을 따서 명명됩니다(예: **MinIO**).

감독자 서비스에 새 버전 추가

vSphere with Tanzu 환경이 있는 vCenter Server에 감독자 서비스를 추가한 후 해당 서비스에 새 버전을 추가할 수 있습니다. 감독자 클러스터에 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

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

감독자 클러스터에 설치된 감독자 서비스 보기

vSphere with Tanzu 환경의 감독자 클러스터에 설치된 vSphere 서비스를 봅니다. 감독자 클러스터에 설치된 감독자 서비스는 클러스터의 각 네임스페이스에서 사용할 수 있습니다.

사전 요구 사항

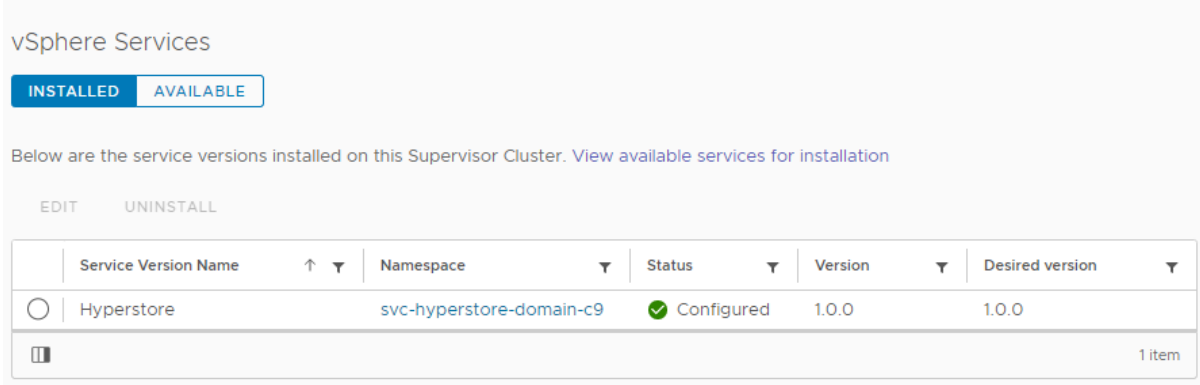
- vCenter Server에 감독자 서비스를 추가합니다. vCenter Server에 감독자 서비스 추가의 내용을 참조하십시오.
- 감독자 클러스터에 감독자 서비스를 설치합니다. 감독자 클러스터에 감독자 서비스 설치의 내용을 참조하십시오.

절차

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

2 **감독자 클러스터** 탭을 선택합니다.

3 감독자 클러스터의 **서비스** 열에서 **보기**를 클릭합니다.



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

다음에 수행할 작업

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

감독자 서비스 또는 버전 비활성화

vSphere with Tanzu 환경의 Kubernetes 워크로드에서 감독자 서비스 버전을 더 이상 사용하지 않으려면 해당 버전을 비활성화합니다. 비활성화된 서비스 버전은 설치된 감독자 클러스터에서 계속 실행되지만 다른 감독자 클러스터에 비활성화된 서비스 버전을 설치할 수는 없습니다. 전체 서비스를 비활성화하면 모든 서비스 버전이 비활성화되고 서비스를 다시 활성화할 때까지는 감독자 클러스터에 새 서비스 버전을 추가하거나 설치할 수 없습니다.

사전 요구 사항

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

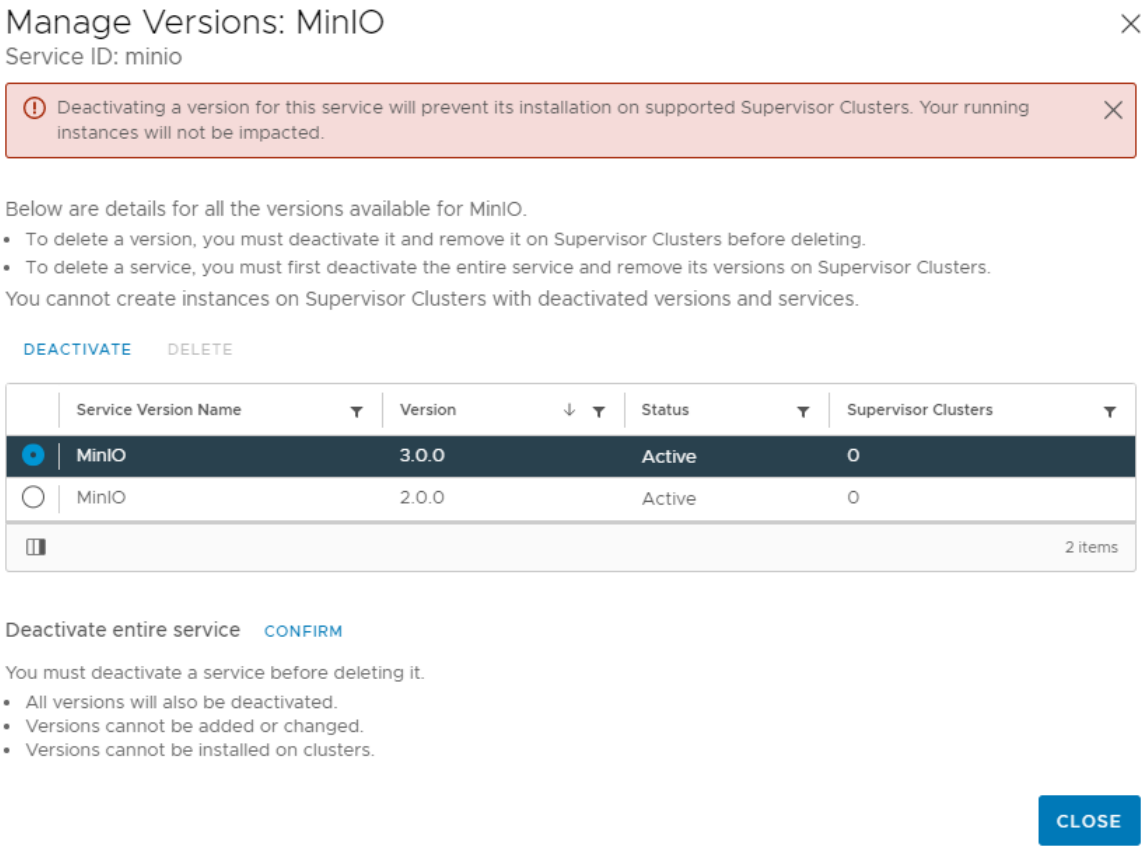
절차

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 감독자 클러스터에 설치할 수 없습니다.

vCenter Server에서 감독자 서비스 버전 활성화

감독자 서비스 버전이 비활성화되면 DevOps팀이 vSphere with Tanzu에서 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.

감독자 클러스터에서 감독자 서비스 제거

DevOps 팀이 vSphere with Tanzu 환경에서 실행되는 Kubernetes 워크로드에 대해 감독자 서비스가 더 이상 필요하지 않으면 이 서비스를 감독자 클러스터에서 제거합니다.

사전 요구 사항

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

절차

1. vSphere Client 홈 메뉴에서 **워크로드 관리**를 선택합니다.
2. **감독자 클러스터**를 선택합니다.
3. 감독자 클러스터의 **서비스** 열에서 **보기**를 클릭합니다.
4. 제거할 감독자 서비스를 선택하고 **제거**를 클릭합니다.

결과

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

감독자 서비스 버전 삭제

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

사전 요구 사항

- 삭제하려는 감독자 서비스 버전이 감독자 클러스터에 설치되어 있지 않은지 확인합니다. 감독자 클러스터에서 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 with Tanzu 환경에서 삭제합니다.

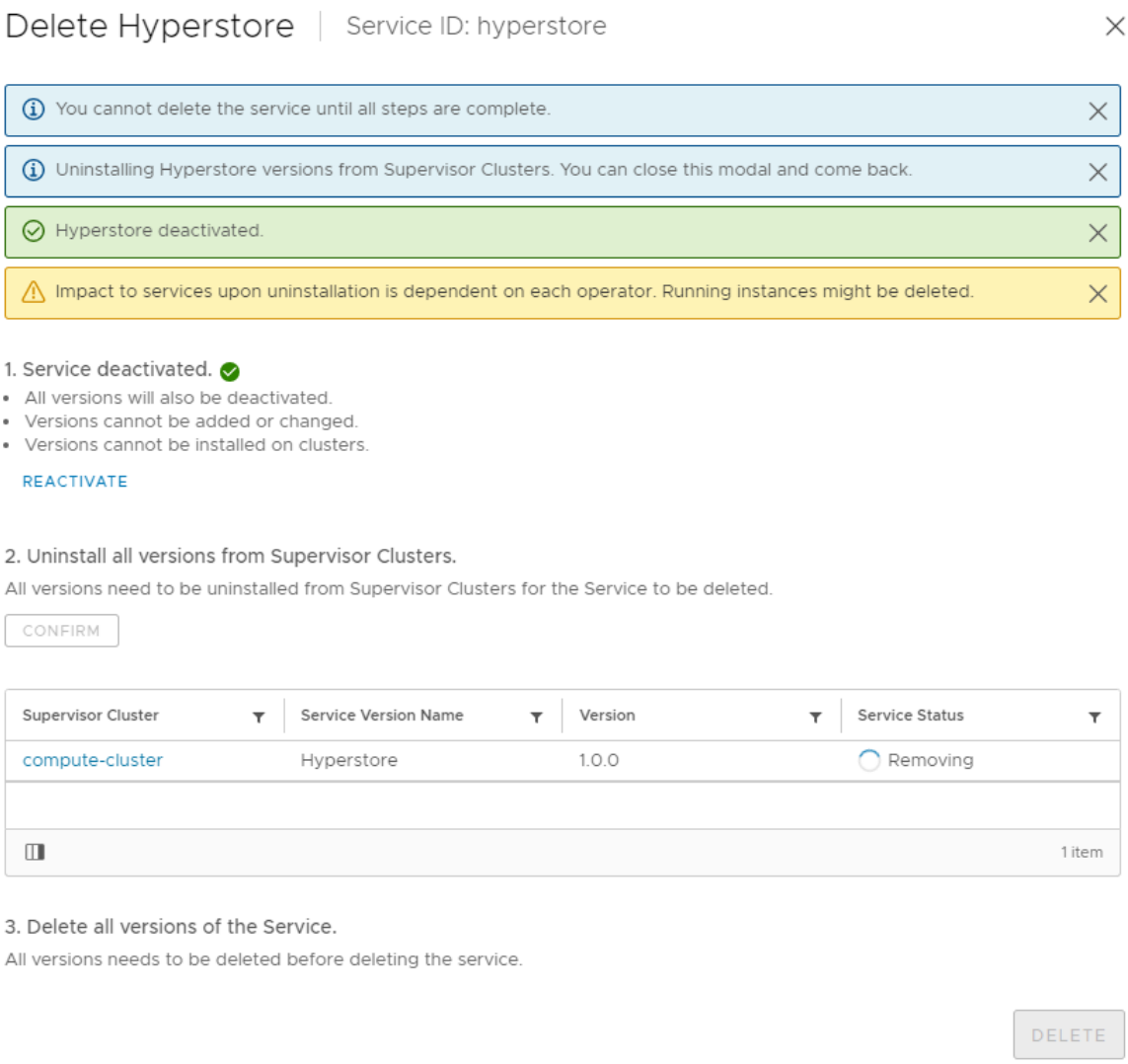
사전 요구 사항

- 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. 감독자 클러스터에서 서비스를 제거한다고 확인합니다.

   실행되는 감독자 클러스터에서 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.

vSphere with Tanzu 클러스터에 연결

9

감독자 클러스터에 연결하여 Tanzu Kubernetes 클러스터를 프로비저닝합니다. 프로비저닝한 후에는 다양한 방법을 사용하여 Tanzu Kubernetes 클러스터에 연결하고 사용자의 역할과 목표를 기반으로 인증할 수 있습니다.

본 장은 다음 항목을 포함합니다.

- vSphere에 대한 Kubernetes CLI 도구 다운로드 및 설치
- vSphere with Tanzu 클러스터에 대한 보안 로그인 구성
- vCenter Single Sign-On 사용자로 감독자 클러스터에 연결
- Tanzu Kubernetes 클러스터에서 인증 수행
- vCenter Single Sign-On 사용자로 Tanzu Kubernetes 클러스터에 연결
- 관리자로 Tanzu Kubernetes 클러스터 제어부에 연결
- 개인 키를 사용하여 시스템 사용자로 Tanzu Kubernetes 클러스터 노드에 SSH를 통해 연결
- 암호를 사용하여 시스템 사용자로 Tanzu Kubernetes 클러스터 노드에 SSH를 통해 연결
- 개발자에게 Tanzu Kubernetes 클러스터에 대한 액세스 권한 부여

vSphere에 대한 Kubernetes CLI 도구 다운로드 및 설치

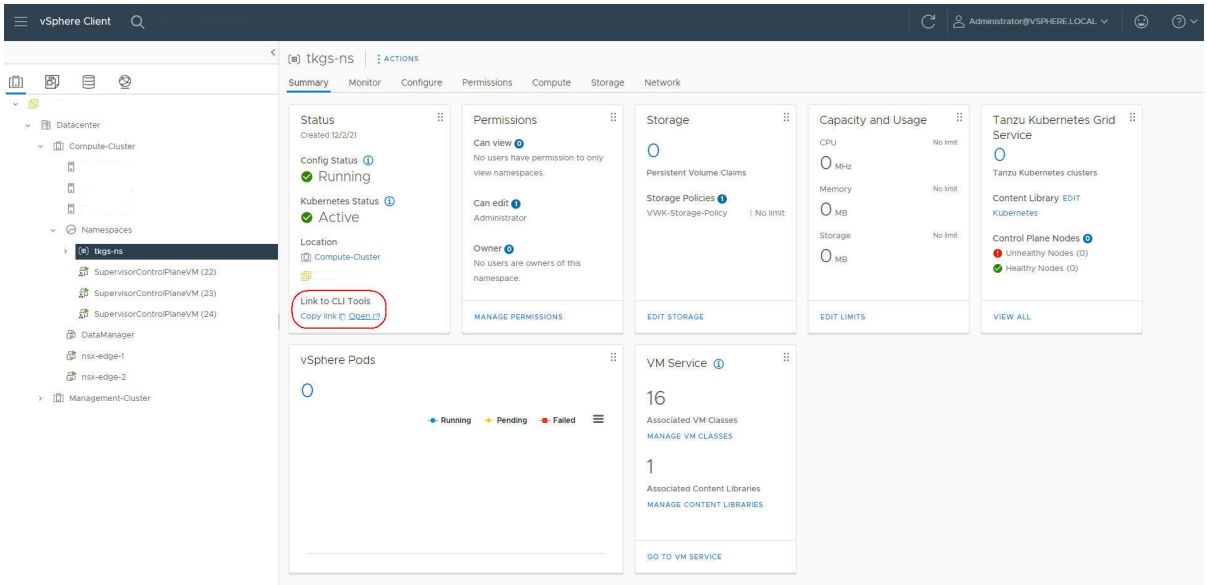
vSphere에 대한 Kubernetes CLI 도구를 사용하여 vSphere with Tanzu 네임스페이스 및 클러스터를 보고 제어할 수 있습니다.

Kubernetes CLI 도구 다운로드 패키지에는 표준 오픈 소스 kubectl 및 kubectl용 vSphere 플러그인의 두 실행 파일이 포함됩니다. kubectl CLI에는 플러그형 아키텍처가 있습니다. kubectl용 vSphere 플러그인은 vCenter Single Sign-On 자격 증명을 사용하여 감독자 클러스터 및 Tanzu Kubernetes 클러스터에 연결할 수 있도록 kubectl에 사용 가능한 명령을 확장합니다.

참고 vSphere 네임스페이스 업데이트를 수행하고 감독자 클러스터를 업그레이드한 후 kubectl용 vSphere 플러그인을 업데이트하는 것이 가장 좋습니다. kubectl용 vSphere 플러그인 업데이트의 내용을 참조하십시오.

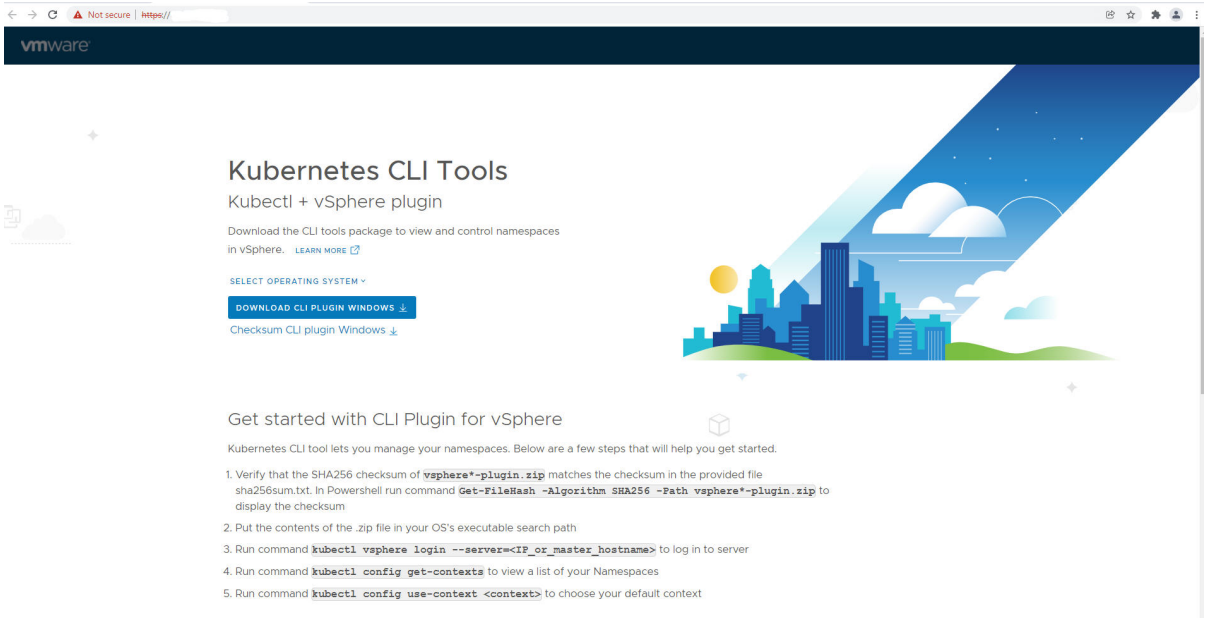
사전 요구 사항

- vSphere 관리자로부터 Kubernetes CLI 도구 다운로드 페이지 링크를 가져옵니다.
- 또는 vCenter Server에 액세스할 수 있으면 다음과 같이 링크를 가져옵니다.
  - vSphere Client를 사용하여 vCenter Server에 로그인합니다.
  - **vSphere 클러스터 > 네임스페이스**로 이동하여 작업 중인 vSphere 네임스페이스를 선택합니다. 아래에 표시된 예에서 이것은 Tanzu Kubernetes 클러스터에 대해 생성한 "tkgs-ns" 네임스페이스입니다.
  - **요약** 탭을 선택하고 이 페이지의 **상태** 영역을 찾습니다.
  - **CLI 도구에 연결** 머리글 아래의 **열기**를 선택하여 다운로드 페이지를 엽니다. 또는 링크를 **복사**할 수 있습니다.



절차

1 브라우저를 사용하여 사용 중인 환경에 해당하는 **Kubernetes CLI 도구** 다운로드 URL로 이동합니다. 다운로드 URL을 찾는 방법에 대한 지침은 위의 필수 구성 요소 섹션을 참조하십시오.



2 운영 체제를 선택합니다.

3 `vsphere-plugin.zip` 파일을 다운로드합니다.

4 작업 디렉토리에 ZIP 파일 컨텐츠의 압축을 풉니다.

`vsphere-plugin.zip` 패키지에는 kubectl 및 kubectl용 vSphere 플러그인의 두 실행 파일이 포함되어 있습니다. `kubectl`은 표준 Kubernetes CLI입니다. `kubectl-vsphere`는 vCenter Single Sign-On 자격 증명을 사용하여 감독자 클러스터 및 Tanzu Kubernetes 클러스터로 인증하는 데 도움이 되는 kubectl용 vSphere 플러그인입니다.

5 두 실행 파일의 위치를 시스템의 PATH 변수에 추가합니다.

6 kubectl CLI의 설치를 확인하려면 셸, 터미널 또는 명령 프롬프트 세션을 시작하고 `kubectl` 명령을 실행합니다.

kubectl 배너 메시지 및 CLI에 대한 명령줄 옵션 목록이 표시됩니다.

7 kubectl용 vSphere 플러그인의 설치를 확인하려면 `kubectl vsphere` 명령을 실행합니다.

kubectl용 vSphere 플러그인 배너 메시지 및 플러그인에 대한 명령줄 옵션 목록이 표시됩니다.

다음에 수행할 작업

vSphere with Tanzu 클러스터에 대한 보안 로그인 구성.

vSphere with Tanzu 클러스터에 대한 보안 로그인 구성

감독자 클러스터 및 Tanzu Kubernetes 클러스터를 포함한 vSphere with Tanzu 클러스터에 안전하게 로그인하려면 적절한 TLS 인증서로 kubectl용 vSphere 플러그인을 구성하고 최신 버전의 플러그인을 실행하고 있는지 확인합니다.

감독자 클러스터 CA 인증서

vSphere with Tanzu는 kubectl용 vSphere 플러그인 명령인 `kubectl vsphere login …`을 사용하여 클러스터에 액세스할 수 있도록 vCenter Single Sign-On을 지원합니다. 이 유틸리티를 설치하고 사용하려면 vSphere에 대한 Kubernetes CLI 도구 다운로드 및 설치 항목을 참조하십시오.

kubectl용 vSphere 플러그인은 기본적으로 보안 로그인을 사용하고 신뢰할 수 있는 인증서가 필요하며, 기본값은 vCenter Server 루트 CA에서 서명한 인증서입니다. 플러그인은 `--insecure-skip-tls-verify` 플래그를 지원하지만 보안상의 이유로 이 플래그는 권장되지 않습니다.

kubectl용 vSphere 플러그인을 사용하여 감독자 클러스터 및 Tanzu Kubernetes 클러스터에 안전하게 로그인하기 위한 두 가지 옵션이 있습니다.

옵션	지침
각 클라이언트 시스템에 vCenter Server 루트 CA 인증서를 다운로드하고 설치합니다.	VMware 기술 자료 문서 vCenter Server 루트 인증서를 다운로드하고 설치하는 방법을 참조하십시오.
감독자 클러스터에 사용되는 VIP 인증서를 각 클라이언트 시스템이 신뢰하는 CA에서 서명한 인증서로 바꿉니다.	VIP 인증서를 교체하여 감독자 클러스터 API 끝점에 안전하게 연결 항목을 참조하십시오.

참고 vCenter Single Sign-On, vCenter Server 인증서 관리 및 순환, 인증 문제 해결을 비롯한 vSphere 인증에 대한 추가 정보는 vSphere 인증 설명서를 참조하십시오.

Tanzu Kubernetes 클러스터 CA 인증서

`kubectl` CLI를 사용하여 Tanzu Kubernetes 클러스터 API 서버와 안전하게 연결하려면 Tanzu Kubernetes 클러스터 CA 인증서를 다운로드해야 합니다.

최신 버전의 kubectl용 vSphere 플러그인을 사용하는 경우에는 Tanzu Kubernetes 클러스터에 처음 로그인할 때 플러그인이 Tanzu Kubernetes 클러스터 CA 인증서를 kubeconfig 파일에 등록합니다. 이 인증서는 *`TANZU-KUBERNETES-CLUSTER-NAME`*`-ca`이라는 이름의 Kubernetes 암호에 저장됩니다. 플러그인은 이 인증서를 사용하여 해당 클러스터의 CA(인증 기관) 데이터스토어에 CA 정보를 채웁니다.

vSphere with Tanzu를 업데이트하는 경우 최신 버전의 플러그인으로 업데이트해야 합니다. kubectl용 vSphere 플러그인 업데이트의 내용을 참조하십시오.

vCenter Single Sign-On 사용자로 감독자 클러스터에 연결

Tanzu Kubernetes Grid 서비스를 사용하여 vSphere 포드 또는 Tanzu Kubernetes 클러스터를 프로비저닝하려면 kubectl용 vSphere 플러그인을 사용하여 감독자 클러스터에 연결하고 vCenter Single Sign-On 자격 증명을 사용하여 인증합니다.

감독자 클러스터에 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 클러스터 제어부의 IP 주소를 가져옵니다.
- vSphere 네임스페이스의 이름을 가져옵니다.
- vSphere 네임스페이스에 대한 **편집** 권한이 있는지 확인합니다.
- vSphere에 대한 Kubernetes CLI 도구 다운로드 및 설치.
- 서명 CA를 신뢰 루트로 설치하거나 인증서를 신뢰 루트로 직접 추가하여 Kubernetes 제어부가 제공하는 인증서를 시스템에서 신뢰할 수 있는지 확인합니다. vSphere with Tanzu 클러스터에 대한 보안 로그인 구성의 내용을 참조하십시오.

절차

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

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

2 감독자 클러스터에 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.

감독자 클러스터에 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

vCenter Single Sign-On 사용자로 Tanzu Kubernetes 클러스터에 연결.

Tanzu Kubernetes 클러스터에서 인증 수행

사용자의 역할 및 목적에 따라 다양한 방법으로 Tanzu Kubernetes 클러스터 환경에서 인증을 수행할 수 있습니다.

DevOps 엔지니어는 Tanzu Kubernetes 클러스터를 프로비저닝하고 작동합니다. 개발자는 워크로드를 Tanzu Kubernetes 클러스터에 배포합니다. 관리자는 Tanzu Kubernetes 클러스터의 문제를 해결해야 할 수 있습니다. vSphere with Tanzu는 각 역할 또는 목표를 지원하는 인증 방법을 제공합니다.

- DevOps 엔지니어는 감독자 클러스터에 연결하여 Tanzu Kubernetes 클러스터를 프로비저닝 및 업데이트합니다. 인증은 kubectl용 vSphere 플러그인 및 vCenter Single Sign-On 자격 증명을 사용하여 수행됩니다. vCenter Single Sign-On 사용자로 감독자 클러스터에 연결의 내용을 참조하십시오.
- 클러스터 관리자는 프로비저닝된 Tanzu Kubernetes 클러스터에 연결하여 작동 및 관리합니다.
  - 클러스터가 배포된 vSphere 네임스페이스에 대한 **편집** 권한이 부여된 사용자가 `cluster-admin` 역할에 할당됩니다. 클러스터 관리자는 kubectl용 vSphere 플러그인 및 해당 vCenter Single Sign-On 자격 증명을 사용하여 인증합니다. vCenter Single Sign-On 사용자로 Tanzu Kubernetes 클러스터에 연결의 내용을 참조하십시오.
  - 또는 클러스터 관리자가 Tanzu Kubernetes 클러스터에 `kubernetes-admin` 사용자로 연결할 수 있습니다. vCenter Single Sign-On 인증을 사용할 수 없는 경우 이 방법이 적합할 수 있습니다. 관리자로 Tanzu Kubernetes 클러스터 제어부에 연결의 내용을 참조하십시오.

- 클러스터 사용자 또는 개발자는 포드, 서비스, 로드 밸런서 및 기타 리소스를 포함한 워크로드를 배포하기 위해 Tanzu Kubernetes 클러스터에 연결합니다.
  - 클러스터 관리자는 사용자 또는 그룹을 기본 또는 사용자 지정 포드 보안 정책에 바인딩하여 개발자에게 클러스터 액세스 권한을 부여합니다. 자세한 내용은 개발자에게 Tanzu Kubernetes 클러스터에 대한 액세스 권한 부여의 내용을 참조하십시오.
  - 바인딩된 개발자는 kubectl용 vSphere 플러그인 및 해당 vCenter Single Sign-On 자격 증명을 사용하여 Tanzu Kubernetes 클러스터에서 인증합니다. vCenter Single Sign-On 사용자로 Tanzu Kubernetes 클러스터에 연결의 내용을 참조하십시오.
- 문제 해결을 위해 시스템 관리자는 SSH 및 개인 키를 사용하여 Tanzu Kubernetes 클러스터에 `vmware-system-user`로 연결할 수 있습니다. 개인 키를 사용하여 시스템 사용자로 Tanzu Kubernetes 클러스터 노드에 SSH를 통해 연결의 내용을 참조하십시오.

vCenter Single Sign-On 사용자로 Tanzu Kubernetes 클러스터에 연결

kubectl용 vSphere 플러그인을 사용하여 Tanzu Kubernetes 클러스터에 연결하고 vCenter Single Sign-On 자격 증명을 사용하여 인증할 수 있습니다.

Tanzu Kubernetes 클러스터에 로그인한 후 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 컨텍스트와 함께 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

vSphere 관리자로부터 다음 정보를 얻습니다.

- vCenter Single Sign-On 자격 증명을 가져옵니다.
- 감독자 클러스터 제어부의 IP 주소를 가져옵니다.
- vSphere 네임스페이스의 이름을 가져옵니다.
- vSphere에 대한 Kubernetes CLI 도구 다운로드 및 설치.

절차

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

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

2 Tanzu Kubernetes 클러스터에 연결하려면 다음 명령을 실행합니다.

```
kubectl vsphere login --server=SUPERVISOR-CLUSTER-CONTROL-PLANE-IP
--tanzu-kubernetes-cluster-name TANZU-KUBERNETES-CLUSTER-NAME
--tanzu-kubernetes-cluster-namespace SUPERVISOR-NAMESPACE-WHERE-THE-CLUSTER-IS-DEPLOYED
--vsphere-username VCENTER-SSO-USER-NAME
```

예:

```
kubectl vsphere login --server=10.92.42.137
--tanzu-kubernetes-cluster-name tanzu-kubernetes-cluster-01
--tanzu-kubernetes-cluster-namespace tanzu-ns-1
--vsphere-username administrator@example.com
```

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

3 인증하려면 vCenter Single Sign-On 암호를 입력합니다.

작업이 성공하면 `Logged in successfully` 메시지가 표시되고 클러스터에 기반하여 kubectl 명령을 실행할 수 있습니다. 명령이 `Error from server (Forbidden)`를 반환하는 경우 일반적으로 이 오류는 사용자에게 필요한 사용 권한이 없음을 의미합니다. 자세한 내용은 vCenter Single Sign-On 연결 오류 문제 해결의 내용을 참조하십시오.

4 사용 가능한 컨텍스트의 목록을 가져오려면 다음 명령을 실행합니다.

```
kubectl config get-contexts
```

이 명령은 액세스 권한이 있는 구성 컨텍스트를 나열합니다. `tkg-cluster-01`와 같은 대상 클러스터의 구성 컨텍스트를 볼 수 있습니다.

5 대상 클러스터의 컨텍스트를 사용하려면 다음 명령을 실행합니다.

```
kubectl config use-context CLUSTER-NAME
```

6 클러스터 노드를 나열하려면 다음 명령을 실행합니다.

```
kubectl get nodes
```

이 클러스터의 제어부 및 작업자 노드가 표시됩니다.

7 클러스터 포드를 모두 나열하려면 다음 명령을 실행합니다.

```
kubectl get pods -A
```

사용자에게 액세스 권한이 있는 모든 Kubernetes 네임스페이스에 걸쳐 이 클러스터의 모든 포드가 표시됩니다. 워크로드를 배포하지 않은 경우 기본 네임스페이스에 포드가 표시되지 않습니다.

관리자로 Tanzu Kubernetes 클러스터 제어부에 연결

kubernetes-admin 사용자로 Tanzu Kubernetes 클러스터 제어부에 연결하여 관리 작업을 수행하고 클러스터 문제를 해결할 수 있습니다.

프로비저닝된 Tanzu Kubernetes 클러스터에 대한 유효한 kubeconfig 파일은 감독자 클러스터에서 *TKGS-CLUSTER-NAME*-kubeconfig이라는 이름의 암호 개체로 제공됩니다. 이 암호를 사용하여 kubernetes-admin 사용자로 클러스터 제어부에 연결할 수 있습니다. 자세한 내용은 Tanzu Kubernetes 클러스터 암호 얻기의 내용을 참조하십시오.

절차

1 감독자 클러스터에 연결합니다. vCenter Single Sign-On 사용자로 감독자 클러스터에 연결의 내용을 참조하십시오.

2 대상 Tanzu Kubernetes 클러스터가 프로비저닝된 vSphere 네임스페이스로 컨텍스트를 전환합니다.

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

3 네임스페이스에서 암호 개체를 봅니다.

```
kubectl get secrets
```

암호 이름은 *TKGS-CLUSTER-NAME*-kubeconfig입니다.

```
kubectl config use-context tkgs-cluster-ns
Switched to context "tkgs-cluster-ns".
ubuntu@ubuntu:~$ kubectl get secrets
NAME                                  TYPE                                  DATA   AGE
...
tkgs-cluster-1-kubeconfig             Opaque                                1      23h
...
```

4 다음 명령을 실행하여 암호를 디코딩합니다.

암호는 Base64로 인코딩됩니다. 디코딩하려면 Linux에서는 base64 --decode(또는 base64 -d)를 사용하고, MacOS에서는 base64 --Decode(또는 base64 -D)를 사용하고, Windows에서는 온라인 도구를 사용합니다.

```
kubectl get secret TKGS-CLUSTER-NAME-kubeconfig -o jsonpath='{.data.value}' | base64 -d >
tkgs-cluster-kubeconfig-admin
```

이 명령은 암호를 디코딩하고 tkgs-cluster-kubeconfig-admin라는 이름의 로컬 파일에 씁니다. cat 명령을 사용하여 파일 컨텐츠를 확인할 수 있습니다.

5 디코딩된 `tkgs-cluster-kubeconfig-admin` 파일을 사용하여 Tanzu Kubernetes 클러스터에 Kubernetes 관리자로 연결합니다.

이 작업을 수행하는 데에는 두 가지 옵션이 있습니다.

옵션	설명
--kubeconfig <path\to\kubeconfig>	`--kubeconfig` 플래그와 로컬 kubeconfig 파일에 대한 경로를 사용합니다. 예를 들어 kubeconfig 파일이 위치한 디렉토리가 명령을 실행하는 디렉토리와 동일하다고 가정하면 다음과 같습니다. `kubectl --kubeconfig tkgs-cluster-kubeconfig-admin get nodes`
KUBECONFIG	디코딩된 kubeconfig 파일을 가리키도록 KUBECONFIG 환경 변수를 설정하고 kubectl을 실행합니다(예: `kubectl get nodes`).

클러스터에 노드가 표시됩니다. 예:

```
kubectl --kubeconfig tkgs-cluster-kubeconfig-admin get nodes
NAME                                          STATUS   ROLES    AGE   VERSION
tkgs-cluster-1-control-plane-4ncm4            Ready    master   23h   v1.18.5+vmware.1
tkgs-cluster-1-control-plane-jj9gq            Ready    master   23h   v1.18.5+vmware.1
tkgs-cluster-1-control-plane-r4hm6            Ready    master   23h   v1.18.5+vmware.1
tkgs-cluster-1-workers-6njk7-84dd7f48c6-nz2n8 Ready    <none>   23h   v1.18.5+vmware.1
tkgs-cluster-1-workers-6njk7-84dd7f48c6-rk9pk Ready    <none>   23h   v1.18.5+vmware.1
tkgs-cluster-1-workers-6njk7-84dd7f48c6-zzngh Ready    <none>   23h   v1.18.5+vmware.1
```

개인 키를 사용하여 시스템 사용자로 Tanzu Kubernetes 클러스터 노드에 SSH를 통해 연결

개인 키를 사용하여 `vmware-system-user`로 Tanzu Kubernetes 클러스터 노드에 SSH를 통해 연결할 수 있습니다.

SSH를 통해 `vmware-system-user` 사용자로 Tanzu Kubernetes 클러스터 노드에 연결할 수 있습니다. SSH 개인 키가 포함된 암호는 `CLUSTER-NAME-ssh`라고 이름이 지정됩니다. 자세한 내용은 Tanzu Kubernetes 클러스터 암호 얻기의 내용을 참조하십시오.

개인 키를 사용하여 SSH를 통해 Tanzu Kubernetes 클러스터 노드에 연결하려면 감독자 클러스터에서 점프 박스(jump box) vSphere 포드를 생성합니다.

사전 요구 사항

이 작업에서는 vSphere 포드를 SSH 연결을 위한 점프 호스트로 프로비저닝합니다. 감독자 클러스터에는 vSphere 포드용 NSX-T 네트워킹이 필요합니다. 감독자 클러스터에 vDS 네트워킹을 사용하는 경우 암호를 사용하여 시스템 사용자로 Tanzu Kubernetes 클러스터 노드에 SSH를 통해 연결 대신 다음 방식을 사용합니다.

절차

1 감독자 클러스터에 연결합니다.

vCenter Single Sign-On 사용자로 감독자 클러스터에 연결의 내용을 참조하십시오.

2 **NAMESPACE**라는 환경 변수를 생성합니다. 이 값은 대상 Tanzu Kubernetes 클러스터가 프로비저닝된 vSphere 네임스페이스의 이름입니다.

```
export NAMESPACE=VSPHERE-NAMESPACE
```

3 Tanzu Kubernetes 클러스터가 프로비저닝된 vSphere 네임스페이스로 컨텍스트를 전환합니다.

```
kubectl config use-context $NAMESPACE
```

4 *TKGS-CLUSTER-NAME*-ssh 암호 개체를 확인합니다.

```
kubectl get secrets
```

5 다음 jumpbox.yaml을 사용하여 vSphere 포드를 생성합니다.

namespace 값 YOUR-NAMESPACE를 대상 클러스터가 프로비저닝된 vSphere 네임스페이스로 바꿉니다. secretName 값 YOUR-CLUSTER-NAME-ssh를 대상 클러스터의 이름으로 바꿉니다.

```
apiVersion: v1
kind: Pod
metadata:
  name: jumpbox
  namespace: YOUR-NAMESPACE     #REPLACE
spec:
  containers:
  - image: "photon:3.0"
    name: jumpbox
    command: [ "/bin/bash", "-c", "--" ]
    args: [ "yum install -y openssh-server; mkdir /root/.ssh; cp /root/ssh/ssh-privatekey /
root/.ssh/id_rsa; chmod 600 /root/.ssh/id_rsa; while true; do sleep 30; done;" ]
    volumeMounts:
      - mountPath: "/root/ssh"
        name: ssh-key
        readOnly: true
    resources:
      requests:
        memory: 2Gi
  volumes:
    - name: ssh-key
      secret:
        secretName: YOUR-CLUSTER-NAME-ssh     #REPLACE
```

6 jumpbox.yaml 규격을 적용하여 포드를 배포합니다.

```
kubectl apply -f jumpbox.yaml
```

```
pod/jumpbox created
```

7 포드가 실행 중인지 확인합니다.

```
kubectl get pods
```

```
NAME      READY   STATUS    RESTARTS   AGE
jumpbox   1/1     Running   0          3h9m
```

참고 vSphere 네임스페이스의 vCenter에도 jumpbox 포드가 보여야 합니다.

8 다음 명령 집합을 실행하여 대상 클러스터 노드의 IP 주소를 사용하여 환경 변수를 생성합니다.

a 대상 가상 시스템의 이름을 가져옵니다.

```
kubectl get virtualmachines
```

b 값이 대상 노드의 이름인 환경 변수 VMNAME을 생성합니다.

```
export VMNAME=NAME-OF-THE-VIRTUAL-MACHINE
```

c 값이 대상 노드 VM의 IP 주소인 환경 변수 VMIP를 생성합니다.

```
export VMIP=$(kubectl -n $NAMESPACE get virtualmachine/$VMNAME -o
jsonpath='{.status.vmIp}')
```

9 다음 명령을 실행하여 점프 박스(jump box) 포드를 사용하여 SSH를 통해 클러스터 노드에 연결합니다.

```
kubectl exec -it jumpbox  /usr/bin/ssh vmware-system-user@$VMIP
```

중요 컨테이너를 생성하고 소프트웨어를 설치하는 데 약 60초가 걸립니다. "error executing command in container: container_linux.go:370: starting container process caused: exec: "/usr/bin/ssh": stat /usr/bin/ssh: no such file or directory"가 표시되면 몇 초 후에 명령을 다시 시도합니다.

10 **yes**를 입력하여 호스트의 신뢰성을 확인합니다.

```
The authenticity of host '10.249.0.999 (10.249.0.999)' can't be established.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.249.0.999' (ECDSA) to the list of known hosts.
Welcome to Photon 3.0
```

11 대상 노드에 vmware-system-user로 로그인되어 있는지 확인합니다.

예를 들어, 다음 출력은 사용자가 제어부 노드에 시스템 사용자로 로그인했음을 나타냅니다.

```
vmware-system-user@tkgs-cluster-1-control-plane-66tbr [ ~ ]$
```

12 노드에서 원하는 작업을 수행합니다.

주의 노드에서 kubelet 다시 시작과 같은 특정 작업을 수행하려면 `sudo` 또는 `sudo su`를 사용해야 할 수 있습니다.

13 완료되면 **`exit`**를 입력하여 vSphere 포드의 SSH 세션에서 로그아웃합니다.

14 포드를 삭제하려면 `kubectl delete pod jumpbox` 명령을 실행합니다.

경고 보안을 위해 작업을 완료한 후에는 jumpbox 포드를 삭제하는 것이 좋습니다. 필요하면 나중에 다시 생성할 수 있습니다.

암호를 사용하여 시스템 사용자로 Tanzu Kubernetes 클러스터 노드에 SSH를 통해 연결

암호를 사용하여 `vmware-system-user`로 Tanzu Kubernetes 클러스터 노드에 SSH를 통해 연결할 수 있습니다.

암호를 사용하여 클러스터 노드에 `vmware-system-user` 사용자로 연결할 수 있습니다. 암호는 `CLUSTER-NAME-ssh-password`라는 이름의 암호로 저장됩니다. 암호는 `.data.ssh-passwordkey`에 Base64로 인코딩됩니다. SSH 세션을 통해 암호를 제공할 수 있습니다. 이 암호에 대한 자세한 내용은 Tanzu Kubernetes 클러스터 암호 얻기 항목을 참조하십시오.

사전 요구 사항

SSH 연결을 적절한 워크로드 네트워크로 라우팅하려면 **워크로드 관리**를 사용하도록 설정된 vSphere 환경에서 Linux 점프 호스트 VM을 배포합니다. Linux 점프 호스트 VM 생성의 내용을 참조하십시오.

참고 SSH를 사용하여 클러스터 노드에 연결하고 vSphere 포드를 지원하지 않는 vDS 네트워킹을 사용하는 경우에는 어려운 요구 사항입니다. 개인 키 대신 암호를 사용하여 SSH를 통해 연결하려는 경우에는 NSX-T 네트워킹에서도 이 방식을 사용할 수 있습니다.

절차

1 점프 호스트 VM의 IP 주소, 사용자 이름 및 암호를 가져옵니다. Linux 점프 호스트 VM 생성의 내용을 참조하십시오.

2 감독자 클러스터에 연결합니다.

vCenter Single Sign-On 사용자로 감독자 클러스터에 연결.

3 대상 Tanzu Kubernetes 클러스터가 프로비저닝된 vSphere 네임스페이스로 컨텍스트를 전환합니다.

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

4 대상 클러스터 노드의 IP 주소를 가져옵니다.

노드를 나열합니다.

```
kubectl get virtualmachines
```

노드를 설명하여 대상 노드의 IP 주소를 가져옵니다.

```
kubectl describe virtualmachines
```

5 `TKGS-CLUSTER-NAME-ssh-password` 암호를 봅니다.

```
kubectl get secrets
```

6 대상 클러스터에 대한 ssh-passwordkey를 가져옵니다.

```
kubectl get secrets TKGS-CLUSTER-NAME-ssh-password -o yaml
```

ssh-passwordkey가 반환됩니다. 예:

```
apiVersion: v1
data:
  ssh-passwordkey: RU1pQllLTC9TRjVFV0RBcCtmd1zwOTROeURYSWNGeXNReXJhaXRBUllYaz0=
```

7 ssh-passwordkey를 디코딩합니다.

암호는 Base64로 인코딩됩니다. 디코딩하려면 Linux에서는 `base64 --decode`(또는 `base64 -d`)를 사용하고, MacOS에서는 `base64 --Decode`(또는 `base64 -D`)를 사용하고, Windows에서는 온라인 도구를 사용합니다.

```
echo <ssh-passwordkey> | base64 --decode
```

8 `vmware-system-user`로 대상 클러스터 노드에 SSH를 통해 연결합니다.

```
ssh vmware-system-user@TKGS-CLUSTER-NODE-IP-ADDRESS
```

9 디코딩한 암호를 사용하여 로그인합니다.

Linux 점프 호스트 VM 생성

암호를 사용하여 Tanzu Kubernetes 클러스터 노드에 SSH로 연결하려면 먼저 워크로드 네트워크에 연결하는 점프 박스(jump box) VM과 SSH 터널링을 위한 관리 또는 프런트 엔드 네트워크를 생성합니다.

Linux 점프 호스트 VM 생성

다음 단계에 따라 Linux 점프 박스(jump-box) VM을 생성합니다. 이것을 수행하는 방법은 여러 가지이며, 다음 방법은 그 중 한 가지입니다. 이 지침에서는 PhotonOS를 사용하며, https://github.com/vmware/photon/wiki/Downloading-Photon-OS에서 다운로드할 수 있습니다.

1 vSphere Client를 사용하여 vCenter Server에 로그인합니다.

2 새 가상 시스템을 생성합니다.

3 Linux 게스트 운영 체제(이 예에서는 VMware Photon OS(64비트))를 선택합니다.

4 운영 체제를 설치합니다. 이렇게 하려면 ISO를 다운로드하여 VM에 연결하고 부팅합니다.

5 워크로드 네트워크에서 IP 주소를 사용하여 VM을 구성합니다.

6 두 번째 가상 NIC를 VM에 추가하고 프런트 엔드 네트워크에 할당합니다.

7 운영 체제 구성을 완료하고 재부팅 후 VM의 전원을 켭니다.

8 VM의 vSphere 콘솔에 루트 사용자로 로그인합니다.

9 새 NIC에 대한 네트워크 인터페이스를 생성하고 프런트 엔드 네트워크에 IP를 제공합니다.

```
ifconfig eth1 IP-ADDRESS netmask NETMASK up
```

참고 이 방법은 재부팅 시 지속되지 않습니다.

10 이 인터페이스를 통해 게이트웨이 및 DNS 서버를 ping할 수 있는지 확인합니다.

11 VM의 vSphere 콘솔에서 인증서를 사용하여 SSH 사용자를 설정합니다. 중첩된 셸을 생성하여 작동하는지 확인합니다.

12 SSH를 사용하여 프런트 엔드 네트워크에서 Jumpbox에 SSH 사용자로 로그인하여 작동하는지 확인합니다.

13 sshpass를 VM에 설치합니다. (그러면 SSH를 통해 암호를 사용하여 로그인할 수 있습니다.) PhotonOS의 경우 명령은 다음과 같습니다.

```
tdnf install -y sshpass
```

14 클라이언트의 공용 키를 `~/.ssh/authorized_keys` 파일에 추가하고 `sshd` 프로세스를 다시 시작하면 ssh가 암호 없이 작동할 수 있습니다.

- 공용 키를 가져옵니다(예: `cat ~/.ssh/id_rsa.pub`).
- 점프 호스트 VM에 액세스합니다.
- SSH 디렉토리(없는 경우)를 생성합니다. `mkdir -p ~/.ssh`.
- 공용 키를 `authorized_keys` 파일에 추가합니다. `echo ssh-rsa AAAA.... >> ~/.ssh/authorized_keys`. `cat ~/.ssh/id_rsa.pub` 명령에서 출력한 전체 공용 키 문자열로 `ssh-rsa AAAA....`를 대체합니다.
- `~/.ssh` 디렉토리 및 `authorized_keys` 파일에 적절한 사용 권한이 설정되어 있는지 확인합니다(예: `chmod -R go= ~/.ssh`).

개발자에게 Tanzu Kubernetes 클러스터에 대한 액세스 권한 부여

개발자는 Kubernetes의 대상 사용자입니다. Tanzu Kubernetes 클러스터가 프로비저닝되면 vCenter Single Sign-On 인증을 사용하여 개발자에게 액세스 권한을 부여할 수 있습니다.

개발자에 대한 인증

클러스터 관리자는 개발자와 같은 다른 사용자에게 클러스터 액세스 권한을 부여할 수 있습니다. 개발자는 사용자 계정을 사용하여 직접 클러스터에 포드를 배포하거나 간접적으로 서비스 계정을 사용할 수 있습니다. 자세한 내용은 Tanzu Kubernetes 클러스터에서 포드 보안 정책 사용의 내용을 참조하십시오.

- 사용자 계정 인증의 경우 Tanzu Kubernetes 클러스터가 vCenter Single Sign-On 사용자 및 그룹을 지원합니다. 사용자 또는 그룹은 vCenter Server에 대해 로컬이거나 지원되는 디렉토리 서버에서 동기화될 수 있습니다.
- 서비스 계정 인증의 경우 서비스 토큰을 사용할 수 있습니다. 자세한 내용은 Kubernetes 설명서를 참조하십시오.

클러스터에 개발자 사용자 추가

개발자에게 클러스터 액세스 권한을 부여하려면 다음을 수행합니다.

1. 사용자 또는 그룹에 대한 역할 또는 ClusterRole을 정의하고 클러스터에 적용합니다. 자세한 내용은 Kubernetes 설명서를 참조하십시오.
2. 사용자 또는 그룹에 대한 RoleBinding 또는 ClusterRoleBinding을 생성하고 클러스터에 적용합니다. 다음 예를 참조하십시오.

RoleBinding 예

vCenter Single Sign-On 사용자 또는 그룹에 액세스 권한을 부여하려면 RoleBinding의 주체에 `name` 매개 변수에 대한 다음 값 중 하나를 포함해야 합니다.

표 9-1. 지원되는 사용자 및 그룹 필드

필드	설명
`sso:`*`USER-NAME@DOMAIN`*	예를 들어 로컬 사용자 이름(예: `sso:joe@vsphere.local`)을 입력합니다.
`sso:`*`GROUP-NAME@DOMAIN`*	예를 들어 디렉토리 서버의 그룹 이름이 vCenter Server(예: `sso:devs@ldap.example.com`)와 통합되었습니다.

다음 RoleBinding 예는 이름이 Joe인 vCenter Single Sign-On 로컬 사용자를 이름이 edit인 기본 ClusterRole에 바인딩합니다. 이 역할은 네임스페이스에 있는 대부분의 개체(이 경우 default 네임스페이스)에 대한 읽기/쓰기 액세스를 허용합니다.

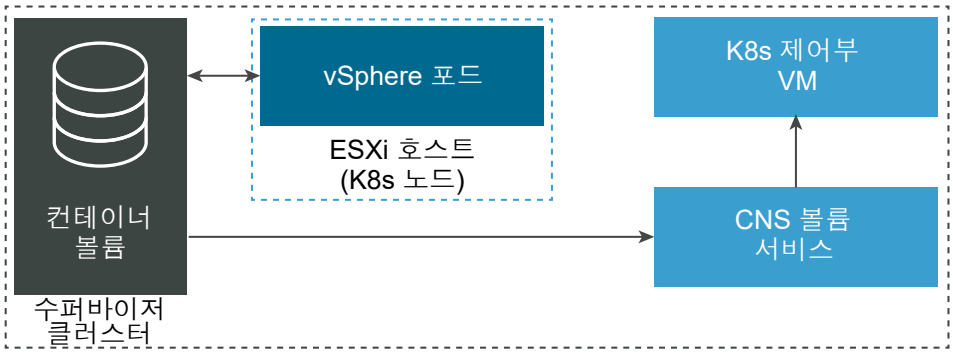
```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: rolebinding-cluster-user-joe
  namespace: default
roleRef:
  kind: ClusterRole
  name: edit                              #Default ClusterRole
  apiGroup: rbac.authorization.k8s.io
subjects:
- kind: User
  name: sso:joe@vsphere.local             #sso:<username>@<domain>
  apiGroup: rbac.authorization.k8s.io
```

vSphere with Tanzu에서 영구 스토리지 사용

10

특정 Kubernetes 워크로드에는 데이터를 영구적으로 저장하기 위한 영구 스토리지가 필요합니다. Kubernetes 워크로드에 대한 영구 스토리지를 프로비저닝하기 위해 vSphere with Tanzu은 영구 볼륨을 관리하는 vCenter Server 구성 요소인 CNS(클라우드 네이티브 스토리지)와 통합됩니다.

영구 스토리지는 vSphere 포드 Tanzu Kubernetes 클러스터 및 VM에서 사용됩니다. 다음 예는 vSphere 포드에서 영구 스토리지를 사용하는 방법을 보여줍니다.



vSphere with Tanzu이 영구 스토리지와 작동하는 방식을 이해하려면 다음과 같은 필수 개념을 숙지해야 합니다.

영구 볼륨

영구 스토리지를 제공하기 위해 Kubernetes는 상태 및 데이터를 유지할 수 있는 영구 볼륨을 사용합니다. 포드에 의해 마운트된 영구 볼륨은 포드가 삭제되거나 재구성되어도 계속 존재합니다. vSphere with Tanzu 환경에서 영구 볼륨 개체는 데이터스토어의 First Class Disk에 의해 지원됩니다.

vSphere with Tanzu은 단일 포드에서 볼륨을 마운트할 수 있는 ReadWriteOnce 모드에서 볼륨의 동적 및 정적 프로비저닝을 지원합니다.

vSphere 7.0 업데이트 3 릴리스부터 vSphere with Tanzu은 Tanzu Kubernetes 클러스터의 영구 볼륨에 대해 ReadWriteMany 모드도 지원합니다. ReadWriteMany 지원을 사용하면 클러스터에서 실행되는 여러 포드 또는 애플리케이션에서 단일 볼륨을 동시에 마운트할 수 있습니다. vSphere with Tanzu는 ReadWriteMany 유형의 영구 볼륨에 vSAN 파일 공유를 사용합니다. 자세한 내용은 vSphere with Tanzu에서 ReadWriteMany 영구 볼륨 생성 항목을 참조하십시오.

동적 및 정적 프로비저닝

동적 볼륨 프로비저닝을 사용하면 스토리지를 미리 프로비저닝할 필요가 없으며 요청 시 영구 볼륨을 생성할 수 있습니다. DevOps 엔지니어는 네임스페이스에서 사용할 수 있는 스토리지 클래스를 참조하는 영구 볼륨 할당을 발급합니다. vSphere with Tanzu는 해당 영구 볼륨과 백업 가상 디스크를 자동으로 프로비저닝합니다.

감독자 클러스터와 Tanzu Kubernetes 클러스터 모두 동적 프로비저닝을 지원합니다.

영구 볼륨을 동적으로 생성하는 방법의 예는 상태 저장 애플리케이션에 대한 동적 영구 볼륨 프로비저닝 항목을 참조하십시오.

정적 프로비저닝을 사용하면 기존 스토리지 개체를 사용하여 클러스터에서 사용할 수 있습니다.

일반적으로 DevOps 엔지니어는 기존 스토리지 개체의 세부 정보, 지원되는 구성 및 마운트 옵션을 알고 있어야 재사용할 수 있습니다.

정적 영구 볼륨을 프로비저닝하는 방법의 예는 Tanzu Kubernetes 클러스터에서 정적 영구 볼륨 프로비저닝 항목을 참조하십시오.

First Class Disk

vSphere with Tanzu은 FCD(First Class Disk) 유형의 가상 디스크를 사용하여 영구 볼륨을 백업합니다. First Class Disk(향상된 가상 디스크라고도 함)는 VM과 연결되지 않은 명명된 가상 디스크입니다.

First Class Disk는 UUID로 식별됩니다. 이 UUID는 전역적으로 고유하며 FCD의 기본 식별자입니다. FCD가 재배치되거나 스냅샷을 만들어도 UUID는 유효하게 유지됩니다.

영구 볼륨 할당

DevOps 엔지니어는 영구 스토리지 리소스를 요청하기 위한 영구 할당을 생성합니다. 이 요청은 영구 볼륨 개체와 일치하는 가상 디스크를 프로비저닝합니다. vSphere Client에서 영구 볼륨 할당은 vSphere 관리자가 모니터링할 수 있는 FCD 가상 디스크로 매니페스트됩니다.

할당은 영구 볼륨에 바인딩됩니다. 워크로드는 할당을 사용하여 영구 볼륨을 마운트하고 스토리지에 액세스할 수 있습니다.

DevOps 엔지니어가 할당을 삭제하면 해당 영구 볼륨 개체와 프로비저닝된 가상 디스크도 삭제됩니다.

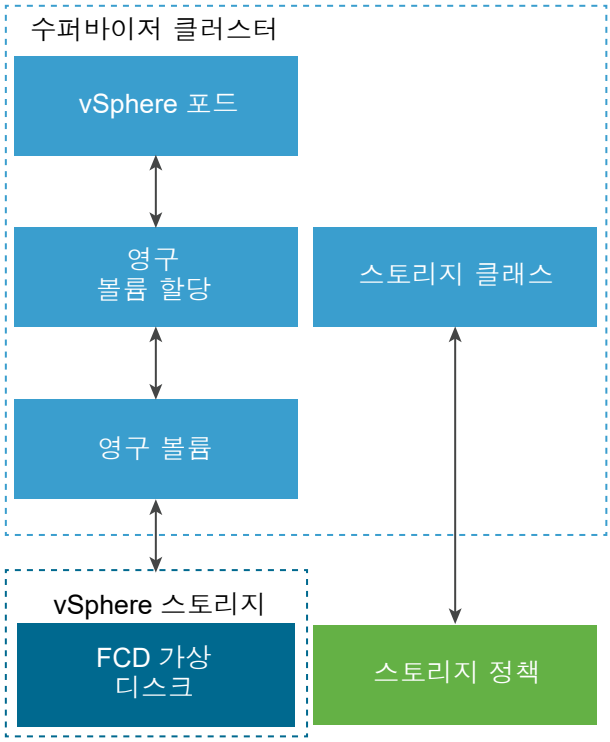
스토리지 클래스

Kubernetes는 스토리지 클래스를 사용하여 영구 볼륨을 백업하는 스토리지에 대한 요구 사항을 설명합니다. DevOps 엔지니어는 영구 볼륨 할당 규격에 특정 스토리지 클래스를 포함하여 클래스가 설명하는 스토리지 유형을 요청할 수 있습니다.

영구 스토리지 워크플로

vSphere with Tanzu에서 영구 스토리지를 프로비저닝하는 워크플로에는 일반적으로 다음과 같은 순차적 작업이 포함됩니다.

단계	작업	설명
1	vSphere 관리자가 영구 스토리지 리소스를 DevOps 팀에 제공합니다.	vSphere 관리자가 다양한 스토리지 요구 사항 및 서비스 클래스를 설명하는 VM 스토리지 정책을 생성합니다. 그런 다음 스토리지 정책을 vSphere 네임스페이스에 할당할 수 있습니다.
2	vSphere with Tanzu은 vSphere 네임스페이스에 할당된 스토리지 정책과 일치하는 스토리지 클래스를 생성합니다.	스토리지 클래스는 Kubernetes 환경에 자동으로 나타나며 DevOps 팀에서 사용할 수 있습니다. vSphere 관리자가 vSphere 네임스페이스에 여러 스토리지 정책을 할당하면 각 스토리지 정책에 대해 별도의 스토리지 클래스가 생성됩니다. Tanzu Kubernetes Grid 서비스를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하는 경우 각 Tanzu Kubernetes 클러스터는 클러스터가 프로비저닝된 vSphere 네임스페이스에서 스토리지 클래스를 상속합니다.
3	DevOps 엔지니어는 스토리지 클래스를 사용하여 워크로드에 대한 영구 스토리지 리소스를 요청합니다.	요청은 특정 스토리지 클래스를 참조하는 영구 볼륨 할당의 형태로 제공됩니다.
4	vSphere with Tanzu은 워크로드에 대해 일치하는 영구 가상 디스크와 영구 볼륨 개체를 생성합니다.	vSphere with Tanzu은 원래 스토리지 정책 및 일치하는 스토리지 클래스에 지정된 요구 사항을 충족하는 데이터스토어에 가상 디스크를 배치합니다. 가상 디스크는 워크로드에 의해 마운트될 수 있습니다.
5	vSphere 관리자는 vSphere with Tanzu 환경의 영구 볼륨을 모니터링합니다.	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 with Tanzu의 영구 스토리지에 대해 알아보려면 이 비디오를 시청하십시오.

(vSphere with Kubernetes의 영구 스토리지)

vSphere 관리자의 스토리지 관리 작업

일반적으로 vSphere with Tanzu의 영구 스토리지 관리 작업에는 다음이 포함됩니다. vSphere 관리자는 vSphere Client를 사용하여 이러한 작업을 수행합니다.

- VM 스토리지 정책에 대한 수명 주기 작업을 수행합니다.

  감독자 클러스터를 사용하도록 설정하고 네임스페이스를 구성하기 전에 영구 스토리지에 대한 스토리지 정책을 생성합니다. 스토리지 정책은 DevOps 엔지니어가 사용자에게 전달하는 스토리지 요구 사항에 기반합니다. vSphere with Tanzu에 대한 스토리지 정책 생성의 내용을 참조하십시오.

  참고 해당 스토리지 클래스를 사용하는 영구 볼륨 할당이 네임스페이스에서 실행되고 있는 경우 vCenter Server 또는 vSphere 네임스페이스에서 스토리지 정책을 삭제하지 마십시오. 이 권장 사항은 Tanzu Kubernetes 클러스터에도 적용됩니다.

- 스토리지 정책을 네임스페이스에 할당하고 스토리지 제한을 설정하여 DevOps 엔지니어에게 스토리지 리소스를 제공합니다. 스토리지 정책 할당 변경에 대한 자세한 내용은 네임스페이스의 스토리지 설정 변경 항목을 참조하십시오. 제한 설정에 대한 자세한 내용은 vSphere 네임스페이스의 Kubernetes 개체에 대한 제한 사항 구성 항목을 참조하십시오.

- vSphere Client에서 Kubernetes 개체 및 해당 스토리지 정책 규정 준수를 모니터링합니다. vSphere Client에서 영구 볼륨 모니터링의 내용을 참조하십시오.

DevOps 엔지니어의 스토리지 관리 작업

일반적으로 DevOps 엔지니어는 `kubectl`을 사용하여 다음 스토리지 작업을 수행합니다.

- 스토리지 클래스를 관리합니다. vSphere 네임스페이스 또는 Tanzu Kubernetes 클러스터에서 스토리지 클래스 표시의 내용을 참조하십시오.
- 상태 저장 애플리케이션을 배포하고 관리합니다. 상태 저장 애플리케이션에 대한 동적 영구 볼륨 프로비저닝의 내용을 참조하십시오.
- 영구 볼륨에 대한 수명 주기 작업을 수행합니다. Tanzu Kubernetes 영구 볼륨 할당 예.

본 장은 다음 항목을 포함합니다.

- vSphere with Tanzu가 vSphere 스토리지와 통합되는 방식
- vSphere with Tanzu에서 vSphere CNS-CSI 및 반가상화 CSI가 지원하는 기능
- vSphere with Tanzu의 스토리지 사용 권한
- vSphere with Tanzu에 대한 스토리지 정책 생성
- 감독자 클러스터의 스토리지 설정 변경
- 네임스페이스의 스토리지 설정 변경
- vSphere 네임스페이스 또는 Tanzu Kubernetes 클러스터에서 스토리지 클래스 표시
- 상태 저장 애플리케이션에 대한 동적 영구 볼륨 프로비저닝
- Tanzu Kubernetes 클러스터에서 정적 영구 볼륨 프로비저닝
- vSphere with Tanzu에서 ReadWriteMany 영구 볼륨 생성
- vSphere with Tanzu의 볼륨 확장
- vSphere Client에서 영구 볼륨 모니터링
- vSphere 네임스페이스 또는 Tanzu Kubernetes 클러스터의 볼륨 상태 모니터링
- 최신 상태 저장 서비스에 vSAN 데이터 지속성 플랫폼 사용

vSphere with Tanzu가 vSphere 스토리지와 통합되는 방식

vSphere with Tanzu는 몇 가지 구성 요소를 사용하여 vSphere 스토리지와 통합합니다.

vCenter Server의 CNS(클라우드 네이티브 스토리지)

CNS 구성 요소는 vCenter Server에 상주합니다. 이것은 영구 볼륨에 대한 수명 주기 작업과 프로비저닝을 구현하는 vCenter Server 관리의 확장입니다.

컨테이너 볼륨을 프로비저닝할 때 이 구성 요소는 vSphere First Class Disk 기능과 상호 작용하여 볼륨을 지원하는 가상 디스크를 생성합니다. 또한 CNS 서버 구성 요소는 스토리지 정책 기반 관리와 통신하여 디스크에 필요한 서비스 수준을 보장합니다.

CNS는 vSphere 관리자가 vCenter Server를 통해 영구 볼륨 및 이를 지원하는 스토리지 개체를 관리하고 모니터링할 수 있도록 하는 쿼리 작업도 수행합니다.

First Class Disk(FCD)

향상된 가상 디스크라고도 합니다. VM과 연결되지 않은 명명된 가상 디스크입니다. 이러한 디스크는 VMFS, NFS 또는 vSAN 데이터스토어 및 백 ReadWriteOnce 영구 볼륨에 상주합니다.

FCD 기술은 VM 또는 포드 수명 주기 범위 밖의 영구 볼륨과 관련된 수명 주기 작업을 수행합니다.

FCD를 사용하는 경우 다음 사항에 유의하십시오.

- FCD는 NFS 4.x 프로토콜을 지원하지 않습니다. 대신 NFS 3을 사용하십시오.
- vCenter Server는 동일한 FCD에서 작업을 직렬화하지 않습니다. 그 결과, 애플리케이션이 동일한 FCD에서 동시에 작업을 수행할 수 없습니다. 복제, 재배치, 삭제, 검색 등의 작업을 서로 다른 스레드에서 동시에 수행하면 예측할 수 없는 결과가 발생하게 됩니다. 문제를 방지하려면 애플리케이션이 동일한 FCD에서 순차적으로 작업을 수행해야 합니다.
- FCD는 관리되는 개체가 아니며 단일 FCD에 대한 다중 쓰기를 보호하는 글로벌 잠금을 지원하지 않습니다. 그 결과 FCD는 동일한 FCD를 관리하는 여러 vCenter Server 인스턴스를 지원하지 않습니다. FCD에서 여러 vCenter Server 인스턴스를 사용해야 하는 경우 다음 옵션이 있습니다.
  - 여러 vCenter Server 인스턴스가 서로 다른 데이터스토어를 관리할 수 있습니다.
  - 여러 vCenter Server 인스턴스가 동일한 FCD에서 작동하지 않습니다.

스토리지 정책 기반 관리

스토리지 정책 기반 관리는 스토리지 정책에 설명된 스토리지 요구 사항에 따라 영구 볼륨 및 해당 백업 가상 디스크의 프로비저닝을 지원하는 vCenter Server 서비스입니다. 프로비저닝 후에 서비스는 스토리지 정책 특성으로 볼륨의 규정 준수를 모니터링합니다. 스토리지 정책 기반 관리에 대한 자세한 내용은 스토리지 정책 기반 관리를 참조하십시오.

vSphere CNS-CSI

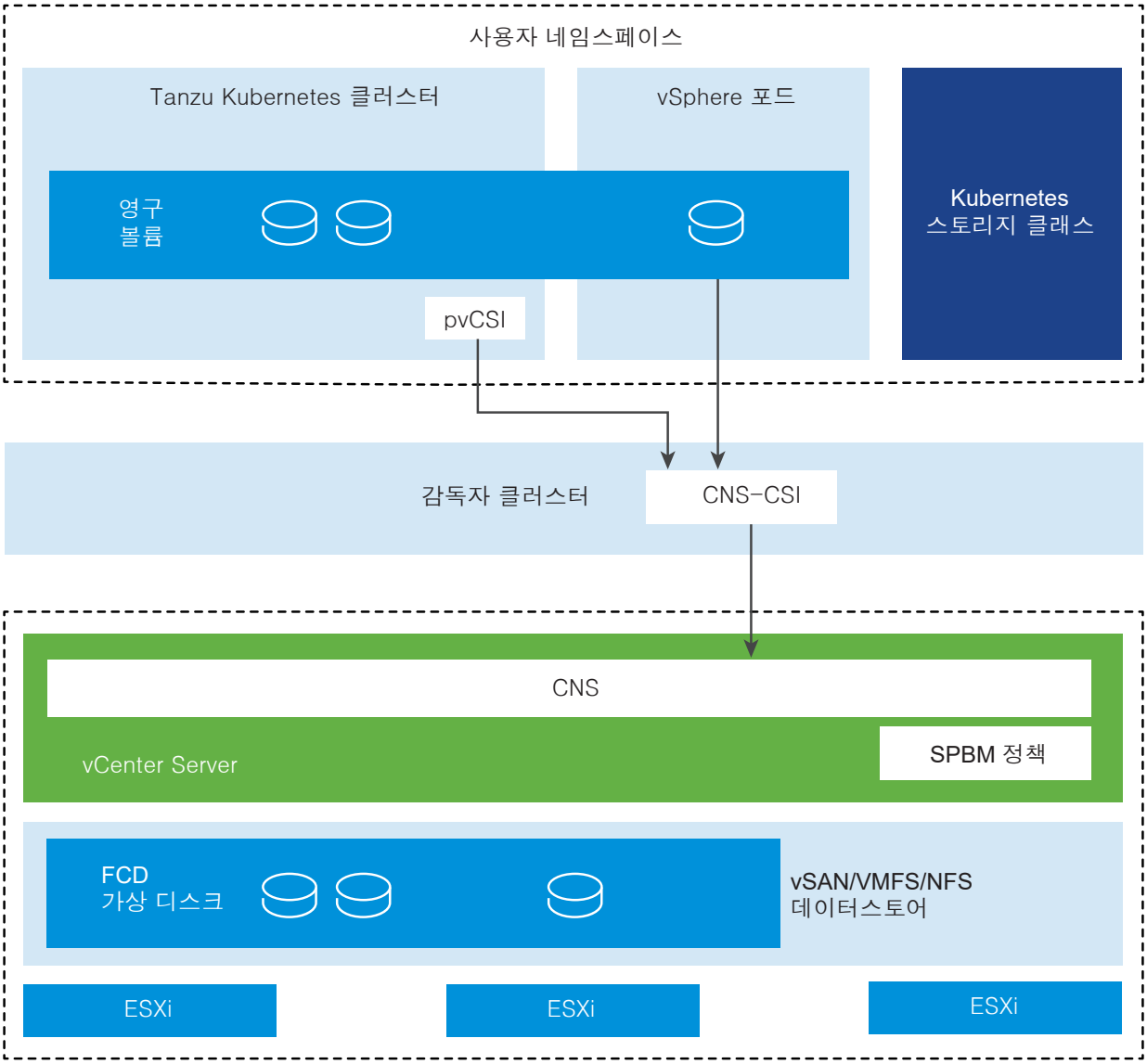
vSphere CNS-CSI 구성 요소는 CSI(Container Storage Interface) 규격을 준수합니다. 이것은 Kubernetes와 같은 컨테이너 Orchestrator가 영구 스토리지를 프로비저닝하는 데 사용하는 인터페이스를 제공하도록 설계된 업계 표준입니다. CNS-CSI 드라이버는 감독자 클러스터에서 실행되고 vSphere 네임스페이스의 Kubernetes 환경에 vSphere 스토리지를 연결합니다. vSphere CNS-CSI는 vSphere 포드 및 네임스페이스의 Tanzu Kubernetes 클러스터에서 실행되는 포드로부터 전송되는 모든 스토리지 프로비저닝 요청에 대해 CNS 제어부와 직접 통신합니다.

pvCSI(반가상화 CSI)

pvCSI는 Tanzu Kubernetes 클러스터에 대해 수정된 vSphere CNS-CSI 드라이버의 버전입니다. pvCSI는 Tanzu Kubernetes 클러스터에 상주하며 Tanzu Kubernetes 클러스터에서 시작되는 모든

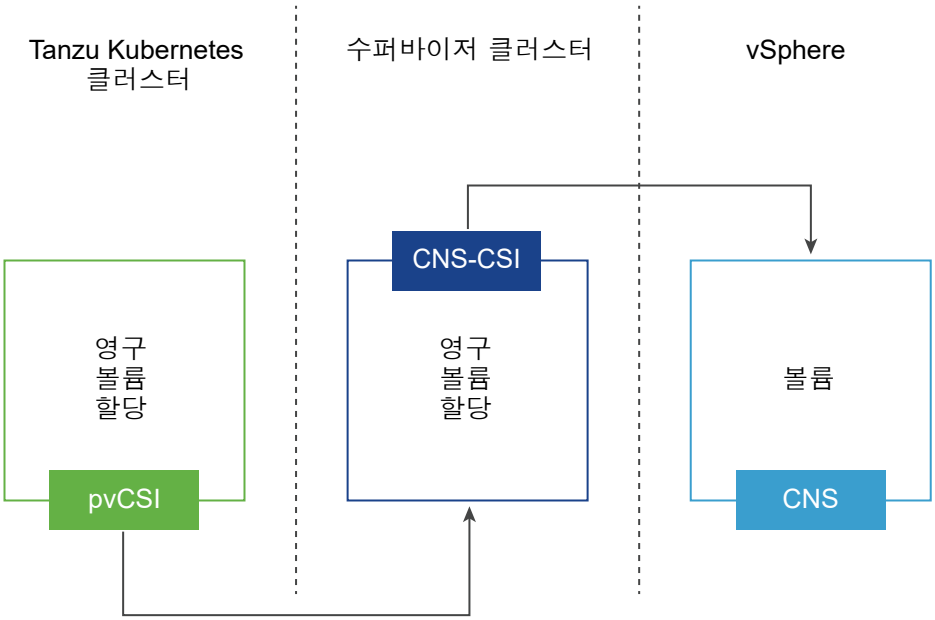
스토리지 관련 요청을 담당합니다. 요청은 CNS-CSI로 전달된 다음 vCenter Server의 CNS로 전파됩니다. 결과적으로 pvCSI는 CNS 구성 요소와 직접적으로 통신하지 않지만 대신 CNS-CSI를 통해 모든 스토리지 프로비저닝 작업을 수행합니다.

CNS-CSI와 달리 pvCSI에는 인프라 자격 증명이 필요하지 않습니다. pvCSI는 vSphere 네임스페이스에서 서비스 계정으로 구성됩니다.



다음은 DevOps 엔지니어가 Tanzu Kubernetes 클러스터 내에서 스토리지 관련 작업을 수행할 때 서로 다른 구성 요소가 어떤 방식으로 상호 작용하는지 보여 줍니다. 예를 들어 PVC(영구 볼륨 할당)가 생성됩니다.

DevOps 엔지니어가 Tanzu Kubernetes 클러스터에서 명령줄을 사용하여 PVC를 생성합니다. 이 작업을 수행하면 감독자 클러스터에서 일치하는 PVC가 생성되고 CNS-CSI가 트리거됩니다. CNS-CSI는 CNS 볼륨 생성 API를 호출합니다.



볼륨 생성이 완료되면 작업이 감독자 클러스터를 통해 Tanzu Kubernetes 클러스터로 다시 전파됩니다. 이 전파의 결과로 사용자는 감독자 클러스터에서 바인딩된 상태의 영구 볼륨과 영구 볼륨 할당을 볼 수 있습니다. 또한 Tanzu Kubernetes 클러스터에서 바인딩된 상태의 영구 볼륨과 영구 볼륨 할당을 볼 수 있습니다.

vSphere with Tanzu에서 vSphere CNS-CSI 및 반가상화 CSI가 지원하는 기능

감독자 클러스터 및 pvCSI 드라이버에서 실행되는 vSphere CNS-CSI 드라이버, Tanzu Kubernetes 클러스터에 대해 수정된 CSI 버전은 여러 vSphere 및 Kubernetes 스토리지 기능을 지원합니다. 하지만 특정 제한 사항이 적용됩니다.

지원되는 기능	감독자 클러스터가 있는 vSphere CNS-CSI	Tanzu Kubernetes 클러스터가 있는 pvCSI
vSphere Client의 CNS 지원	예	예
vSphere Client의 향상된 개체 상태	예(vSAN만)	예(vSAN만)
동적 블록 영구 볼륨(ReadWriteOnce 액세스 모드)	예	예
동적 파일 영구 볼륨(ReadWriteMany 액세스 모드)	아니요	예(vSAN 파일 서비스 사용)
vSphere 데이터스토어	VMFS/NFS/vSAN/vVols	VMFS/NFS/vSAN/vVols
정적 영구 볼륨	예	예
암호화	아니요	아니요
오프라인 볼륨 확장	예	예
온라인 볼륨 확장	예	예

지원되는 기능	감독자 클러스터가 있는 vSphere CNS-CSI	Tanzu Kubernetes 클러스터가 있는 pvCSI
볼륨 토폴로지 및 영역	아니요	아니요
Kubernetes 다중 제어부 인스턴스	예	예
WaitForFirstConsumer	아니요	아니요
VolumeHealth	예	예

vSphere with Tanzu의 스토리지 사용 권한

vSphere with Tanzu는 스토리지 작업에 대한 권한 집합이 포함된 샘플 역할인 워크로드 스토리지 관리자를 제공합니다. 이 역할을 복제하여 유사한 역할을 생성할 수 있습니다.

권한 이름	설명	필수
CNS.검색 가능	스토리지 관리자가 클라우드 네이티브 스토리지 UI를 볼 수 있습니다.	루트 vCenter Server
데이터스토어.공간 할당 **데이터스토어.하위 수준 파일 작업**	데이터스토어에서 가상 시스템, 스냅샷, 복제본 또는 가상 디스크를 위한 공간을 할당할 수 있습니다. 데이터스토어 브라우저에서 읽기, 쓰기, 삭제 및 이름 변경 작업을 수행할 수 있습니다.	영구 볼륨이 상주하는 공유 데이터스토어
ESX Agent Manager.수정	에이전트 가상 시스템에 대해 가상 시스템 전원 끄기 또는 삭제와 같은 수정 작업을 수행할 수 있습니다.	vSphere 포드
리소스.리소스 풀에 가상 시스템 할당	리소스 풀에 가상 시스템을 할당할 수 있습니다.	리소스 풀
프로파일 구동 스토리지.프로파일 구동 스토리지 보기	정의된 스토리지 정책을 볼 수 있습니다.	루트 vCenter Server
가상 시스템.구성 변경.기존 디스크 추가 **가상 시스템.구성 변경.새 디스크 추가** **가상 시스템.구성 변경.디바이스 추가 또는 제거** **가상 시스템.구성 변경.설정 변경** **가상 시스템.구성 변경.디스크 제거** **가상 시스템.인벤토리 편집.새로 생성** **가상 시스템.인벤토리 편집.제거**	가상 시스템을 생성 및 삭제할 수 있습니다. 가상 시스템 옵션 및 디바이스를 구성할 수 있습니다.	vSphere 포드

vSphere with Tanzu에 대한 스토리지 정책 생성

vSphere with Tanzu을 사용하도록 설정하기 전에 감독자 클러스터 및 네임스페이스에서 사용될 스토리지 정책을 생성합니다. 정책은 vSphere 환경에서 사용할 수 있는 데이터스토어를 나타냅니다. 정책은 제어부 VM, 포드 사용 후 삭제 디스크, 컨테이너 이미지 및 영구 스토리지 볼륨과 같은 개체의 스토리지 배치를 제

어합니다. Tanzu Kubernetes 클러스터를 사용하는 경우 스토리지 정책은 Tanzu Kubernetes 클러스터 노드가 배포되는 방식도 지정합니다.

vSphere 스토리지 환경 및 DevOps의 요구 사항에 따라 서로 다른 스토리지 클래스를 나타내는 여러 스토리지 정책을 생성할 수 있습니다. 예를 들어 vSphere 스토리지 환경에 Bronze, Silver 및 Gold의 3가지 데이터스토어 클래스가 있는 경우 모든 데이터스토어에 대한 스토리지 정책을 생성할 수 있습니다. 그런 다음 사용 후 삭제 및 컨테이너 이미지 가상 디스크에 대해 Bronze 데이터스토어를 사용하고 영구 볼륨 가상 디스크에 대해 Silver 및 Gold 데이터스토어를 사용할 수 있습니다. 스토리지 정책에 대한 자세한 내용은 "vSphere 스토리지" 설명서에서 스토리지 정책 기반 관리 장을 참조하십시오.

다음 예에서는 Gold로 태그 지정된 데이터스토어에 대한 스토리지 정책을 생성합니다.

vSAN 데이터 지속성 플랫폼을 사용하는 경우에는 vSAN Direct 또는 vSAN SNA 데이터스토어에 대한 스토리지 정책을 생성할 수 있습니다. 자세한 내용은 vSAN Direct 스토리지 정책 생성 및 vSAN SNA 스토리지 정책 생성 항목을 참조하십시오.

사전 요구 사항

- 스토리지 정책에서 참조하는 데이터스토어가 클러스터의 모든 ESXi 호스트 간에 공유되는지 확인합니다.
- 필요한 권한: **VM 스토리지 정책. 업데이트** 및 **VM 스토리지 정책. 보기**.

절차

1 데이터스토어에 태그를 추가합니다.

   a 태그를 지정하려는 데이터스토어를 마우스 오른쪽 버튼으로 클릭하고 **태그 및 사용자 지정 특성 > 태그 할당**을 선택합니다.

   b **태그 추가**를 클릭하고 태그의 속성을 지정합니다.

속성	설명
이름	데이터스토어 태그의 이름을 지정합니다(예: **Gold**).
설명	태그에 대한 설명을 추가합니다. 예를 들어 **Datastore for Kubernetes objects**라고 입력합니다.
범주	기존 범주를 선택하거나 새 범주를 생성합니다. 예를 들어 **Storage for Kubernetes**라는 범주를 선택하거나 생성합니다.

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

   a **메뉴 > 정책 및 프로파일**을 클릭합니다.

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

   c **VM 스토리지 정책 생성**을 클릭합니다.

3 정책 이름 및 설명을 입력합니다.

옵션	작업
vCenter Server	vCenter Server 인스턴스를 선택합니다.
이름	스토리지 정책의 이름(예: **goldsp**)을 입력합니다. **참고** vSphere with Tanzu는 네임스페이스에 할당하는 스토리지 정책을 Kubernetes 스토리지 클래스로 변환할 때 모든 대문자를 소문자로 변경하고 공백을 대시(-)로 바꿉니다. 혼동을 방지하려면 VM 스토리지 정책 이름에 소문자를 사용하고 공백을 사용하지 마십시오.
설명	스토리지 정책의 설명을 입력합니다.

4 **정책 구조** 페이지의 **데이터스토어별 규칙**에서 태그 기반 배치 규칙을 사용하도록 설정합니다.

5 **태그 기반 배치** 페이지에서 태그 규칙을 생성합니다.

다음 예를 사용하여 옵션을 선택합니다.

옵션	설명
태그 범주	드롭다운 메뉴에서 태그의 범주(예: **Storage for Kubernetes**)를 선택합니다.
사용 옵션	**다음으로 태그 지정된 스토리지 사용**을 선택합니다.
태그	**태그 찾아보기**를 클릭하고 데이터스토어 태그(예: **Gold**)를 선택합니다.

6 **스토리지 호환성** 페이지에서 이 정책과 일치하는 데이터스토어 목록을 검토합니다.

이 예에서는 Gold로 태그 지정된 데이터스토어만 표시됩니다.

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

결과

Gold로 태그 지정된 데이터스토어에 대한 새로운 스토리지 정책이 기존 스토리지 정책의 목록에 나타납니다.

다음에 수행할 작업

스토리지 정책을 생성한 후 vSphere 관리자는 다음 작업을 수행할 수 있습니다.

- 감독자 클러스터에 스토리지 정책을 할당합니다. 감독자 클러스터에 구성된 스토리지 정책은 제어부 VM, 포드 사용 후 삭제 디스크 및 컨테이너 이미지가 정책이 나타내는 데이터스토어에 배치되도록 합니다. NSX-T Data Center 네트워킹으로 워크로드 관리 사용의 내용을 참조하십시오.
- vSphere 네임스페이스에 스토리지 정책을 할당합니다. 네임스페이스에 표시되는 스토리지 정책은 네임스페이스가 영구 볼륨에 대해 액세스하고 사용할 수 있는 데이터스토어를 결정합니다. 스토리지 정책은 일치하는 Kubernetes 스토리지 클래스로 네임스페이스에 나타납니다. 또한 이 네임스페이스의 Tanzu Kubernetes 클러스터에도 전파됩니다. DevOps 엔지니어는 스토리지 클래스를 영구 볼륨 할당 규격에 사용할 수 있습니다. vSphere 네임스페이스 생성 및 구성의 내용을 참조하십시오.

감독자 클러스터의 스토리지 설정 변경

감독자 클러스터에 할당된 스토리지 정책은 제어부 VM, vSphere 포드, 컨테이너 이미지 캐시와 같은 개체가 vSphere 스토리지 환경의 데이터스토어 내에 배치되는 방식을 관리합니다. vSphere 관리자는 일반적으로 감독자 클러스터를 사용하도록 설정할 때 스토리지 정책을 구성합니다. 초기 감독자 클러스터 구성 후 스토리지 정책 할당을 변경해야 하는 경우 이 작업을 수행합니다. 이 작업을 사용하여 ReadWriteMany 영구 볼륨에 대한 파일 볼륨 지원을 활성화하거나 비활성화할 수도 있습니다.

스토리지 설정에 대한 변경 내용은 새 개체에만 적용됩니다.

사전 요구 사항

ReadWriteMany 모드에서 영구 볼륨에 대한 파일 볼륨 지원을 활성화하려면 vSphere with Tanzu에서 ReadWriteMany 영구 볼륨 생성의 사전 요구 사항을 따르십시오.

절차

1 vSphere Client에서 vSphere with Tanzu를 사용하도록 설정한 호스트 클러스터로 이동합니다.

2 **구성** 탭을 클릭하고 **네임스페이스**에서 **스토리지**를 클릭합니다.

3 다음 항목에 대한 스토리지 정책 할당을 변경합니다.

옵션	설명
제어부 노드	제어부 VM 배치에 대한 스토리지 정책을 선택합니다.
포드 사용 후 삭제 디스크	vSphere 포드 배치에 대한 스토리지 정책을 선택합니다.
컨테이너 이미지 캐시	컨테이너 이미지의 캐시 배치에 대한 스토리지 정책을 선택합니다.

4 파일 볼륨 지원을 사용하도록 설정하여 ReadWriteMany 영구 볼륨을 배포합니다.

네임스페이스의 스토리지 설정 변경

vSphere 관리자가 감독자 클러스터의 네임스페이스에 할당하는 스토리지 정책은 영구 볼륨과 Tanzu Kubernetes 클러스터 노드가 vSphere 데이터스토어 내에 배치되는 방식을 제어합니다. 영구 볼륨에 해당하는 영구 볼륨 할당은 vSphere 포드 또는 Tanzu Kubernetes 클러스터에서 발생할 수 있습니다. 원래 스토리지 정책 할당을 변경할 수 있습니다.

사전 요구 사항

vCenter Server 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.

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.

vSphere 네임스페이스 또는 Tanzu Kubernetes 클러스터에서 스토리지 클래스 표시

vSphere 관리자가 스토리지 정책을 생성하고 이를 vSphere 네임스페이스에 할당하면 이 스토리지 정책이 네임스페이스 및 사용 가능한 모든 Tanzu Kubernetes 클러스터에 일치하는 Kubernetes 스토리지 클래스로 나타납니다. 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
```

상태 저장 애플리케이션에 대한 동적 영구 볼륨 프로비저닝

상태 저장 애플리케이션(예: 데이터베이스)은 세션 간 데이터를 저장하며, 데이터 저장을 위해 영구 스토리지가 필요합니다. 보존된 데이터를 애플리케이션의 상태라고 합니다. 나중에 데이터를 검색하여 다음 세션에서 사용할 수 있습니다. Kubernetes는 해당 상태와 데이터를 보존할 수 있는 개체로 영구 볼륨을 제공합니다.

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

2 스토리지 클래스를 사용할 수 있는지 확인합니다.

vSphere 네임스페이스 또는 Tanzu Kubernetes 클러스터에서 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. vSphere with Tanzu에서 ReadWriteMany 영구 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 클러스터의 볼륨 상태 모니터링 항목을 참조하십시오. vSphere Client의 영구 볼륨을 검토하고 모니터링하려면 vSphere Client에서 영구 볼륨 모니터링 항목을 참조하십시오.

Tanzu Kubernetes 클러스터에서 정적 영구 볼륨 프로비저닝

감독자 클러스터 에서 사용되지 않는 PVC(영구 볼륨 할당)를 사용하여 Tanzu Kubernetes 클러스터에서 블록 볼륨을 정적으로 생성할 수 있습니다.

PVC는 다음 조건을 충족해야 합니다.

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

정적 프로비저닝을 사용하면 다른 Tanzu Kubernetes 호스트에 더 이상 필요하지 않은 PVC를 새 Tanzu Kubernetes 클러스터에서 재사용할 수 있습니다. 이렇게 하려면 원래 Tanzu Kubernetes 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 클러스터에 PVC를 정적으로 생성합니다.

절차

1 감독자 클러스터의 원래 PVC 이름을 기록해둡니다.

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

2 PV를 생성합니다.

YAML 파일에서 다음 항목의 값을 지정합니다.

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

다른 Tanzu Kubernetes 클러스터의 볼륨을 재사용하는 경우에는 새 Tanzu Kubernetes 클러스터에 PV를 생성하기 전에 이전 Tanzu Kubernetes 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 cluster."
```

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
```

vSphere with Tanzu에서 ReadWriteMany 영구 볼륨 생성

vSphere 7.0 업데이트 3 릴리스부터 vSphere with Tanzu는 ReadWriteMany 모드에서 영구 볼륨을 지원합니다. ReadWriteMany 지원을 사용하면 클러스터에서 실행되는 여러 포드 또는 애플리케이션에서 단일 볼륨을 동시에 마운트할 수 있습니다. vSphere with Tanzu는 vSAN 파일 서비스를 사용하여 ReadWriteMany 영구 볼륨에 대한 파일 공유를 제공합니다.

ReadWriteMany 영구 볼륨에 대한 고려 사항

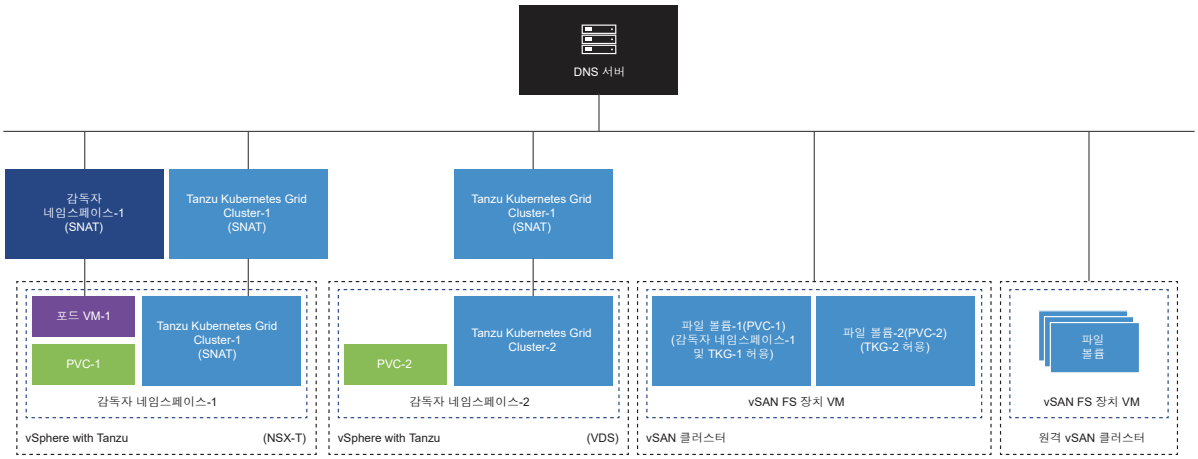
vSphere with Tanzu에서 영구 볼륨에 대해 ReadWriteMany 지원을 사용하도록 설정하는 경우 다음 고려 사항에 유의하십시오.

- Tanzu Kubernetes 클러스터에서는 TKr 버전 1.22를 사용합니다.

  참고 예정된 TKr 버전 1.22가 릴리스된 경우에만 ReadWriteMany 기능을 사용할 수 있습니다. TKr 버전에 대한 자세한 내용은 VMware Tanzu Kubernetes 릴리스 정보를 참조하십시오.

- vSphere with Tanzu에 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 네임스페이스가 NAT 모드에 있는지 확인합니다. 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 클러스터에 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 ReadWriteMany 지원을 사용하도록 설정하는 워크플로

영구 볼륨에 대한 ReadWriteMany 지원을 사용하도록 설정하려면 이 프로세스를 따르십시오.

1 vSphere 관리자는 구성된 vSAN 파일 서비스로 vSAN 클러스터를 설정합니다. 파일 서비스 구성을 참조하십시오.

2 vSphere 관리자가 감독자 클러스터에서 파일 볼륨 지원을 활성화합니다.

작업	설명
워크로드 관리 플랫폼을 사용하도록 설정할 때 파일 볼륨 지원을 활성화합니다.	■ vSphere 네트워킹으로 워크로드 관리 사용 ■ NSX-T Data Center 네트워킹으로 워크로드 관리 사용
기존 클러스터에서 파일 볼륨 지원을 활성화합니다(예: vSphere with Tanzu 업그레이드 후).	감독자 클러스터의 스토리지 설정 변경

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

동일한 PVC로 여러 포드를 프로비저닝할 수 있습니다.

상태 저장 애플리케이션에 대한 동적 영구 볼륨 프로비저닝의 내용을 참조하십시오.

vSphere with Tanzu의 볼륨 확장

DevOps 엔지니어는 Kubernetes 볼륨 확장 기능을 사용하여 영구 블록 볼륨을 생성한 후 확장할 수 있습니다. 두 가지 유형의 클러스터 즉, 감독자 클러스터 및 Tanzu Kubernetes 클러스터는 모두 오프라인 및 온라인 볼륨 확장을 지원합니다.

vSphere with Tanzu 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.

Tanzu Kubernetes를 사용하는 경우에는 Tanzu Kubernetes 클러스터와 감독자 클러스터 모두를 지원할 기능에 적합한 버전으로 업그레이드해야 합니다. Tanzu Kubernetes 클러스터의 기능은 감독자 클러스터에서 해당 기능을 사용하도록 설정하는지 여부에 달려 있습니다.

예를 들어 Tanzu Kubernetes 클러스터를 vSphere 7.0 업데이트 2로 업그레이드하고 감독자 클러스터는 7.0 업데이트 1로 남겨두면 Tanzu Kubernetes 클러스터에서 온라인 볼륨 확장이 작동하지 않습니다.

	감독자 클러스터 7.0	감독자 클러스터 7.0 업데이트 1	감독자 클러스터 7.0 업데이트 2
Tanzu Kubernetes 클러스터 7.0	Tanzu Kubernetes 클러스터 또는 감독자 클러스터의 오프라인 및 온라인 확장: `not supported`	Tanzu Kubernetes 클러스터 또는 감독자 클러스터의 오프라인 및 온라인 확장: `not supported`	■ Tanzu Kubernetes 클러스터의 오프라인 및 온라인 확장: `not supported` ■ 감독자 클러스터의 오프라인 및 온라인 확장: `supported`
Tanzu Kubernetes 클러스터 7.0 업데이트 1	Tanzu Kubernetes 클러스터 또는 감독자 클러스터의 오프라인 및 온라인 확장: `not supported`	■ Tanzu Kubernetes 클러스터의 오프라인 확장: `supported` ■ 감독자 클러스터의 오프라인 확장: `not supported` ■ Tanzu Kubernetes 클러스터 또는 감독자 클러스터의 온라인 확장: `not supported`	■ Tanzu Kubernetes 클러스터의 오프라인 확장: `supported` ■ Tanzu Kubernetes 클러스터의 온라인 확장: `not supported` ■ 감독자 클러스터의 오프라인 및 온라인 확장: `supported`
Tanzu Kubernetes 클러스터 7.0 업데이트 2	Tanzu Kubernetes 클러스터 또는 감독자 클러스터의 오프라인 및 온라인 확장: `not supported`	■ Tanzu Kubernetes 클러스터의 오프라인 확장: `supported` ■ 감독자 클러스터의 오프라인 확장: `not supported` ■ Tanzu Kubernetes 클러스터 또는 감독자 클러스터의 온라인 확장: `not supported`	Tanzu Kubernetes 클러스터 또는 감독자 클러스터의 오프라인 및 온라인 확장: `supported`

볼륨을 확장할 때는 다음 사항에 유의하십시오.

- 볼륨은 스토리지 할당량에 지정된 제한까지 확장할 수 있습니다. vSphere with Tanzu는 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 클러스터 및 Tanzu Kubernetes 클러스터에서 정적으로 프로비저닝된 볼륨의 크기를 조절할 수 있습니다(볼륨에 연결된 스토리지 클래스가 있는 경우).
- StatefulSet의 일부로 생성된 볼륨은 확장할 수 없습니다.
- 볼륨을 지원하는 가상 디스크에 스냅샷이 있으면 크기를 조정할 수 없습니다.
- vSphere with Tanzu는 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 클러스터 및 Tanzu Kubernetes 클러스터는 모두 오프라인 볼륨 확장을 지원합니다.

사전 요구 사항

vSphere 환경을 오프라인 볼륨 확장을 지원하는 적절한 버전으로 업그레이드해야 합니다. vSphere with Tanzu의 볼륨 확장의 내용을 참조하십시오.

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 클러스터 및 Tanzu Kubernetes 클러스터는 모두 온라인 볼륨 확장을 지원합니다.

사전 요구 사항

vSphere 환경을 온라인 볼륨 확장을 지원하는 적절한 버전으로 업그레이드해야 합니다. vSphere with Tanzu의 볼륨 확장의 내용을 참조하십시오.

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 with Tanzu은 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 네임스페이스를 클릭합니다.

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

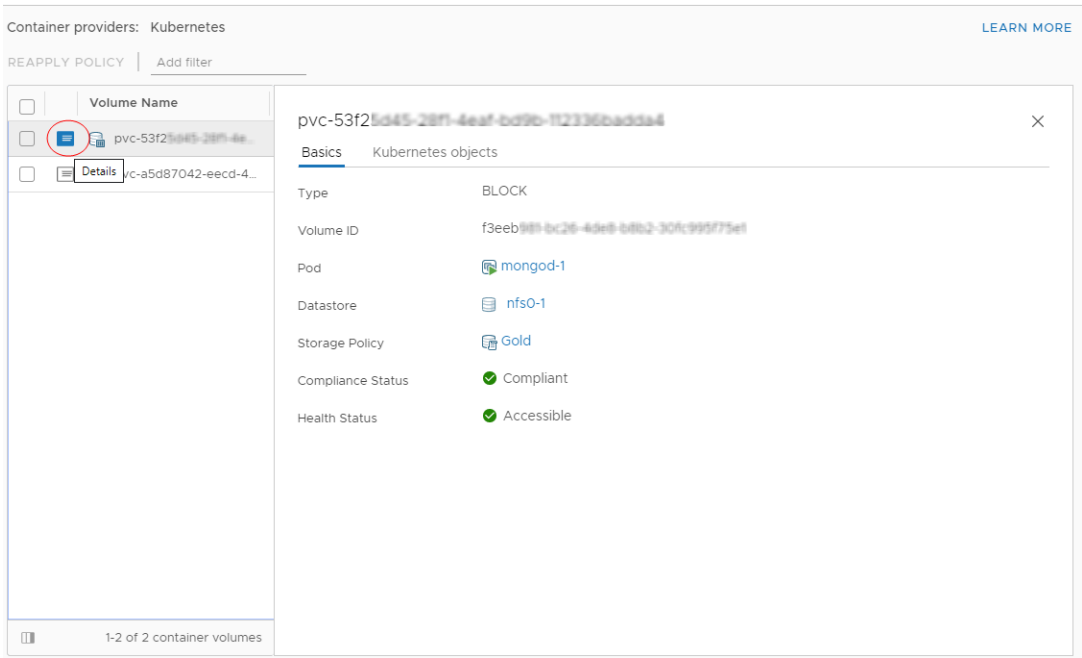
   vSphere Client가 모든 영구 볼륨 할당 개체와 네임스페이스에서 사용할 수 있는 해당 볼륨을 나열합니다.

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 클러스터의 볼륨 상태 모니터링 항목을 참조하십시오.



b 볼륨의 상태를 확인합니다.

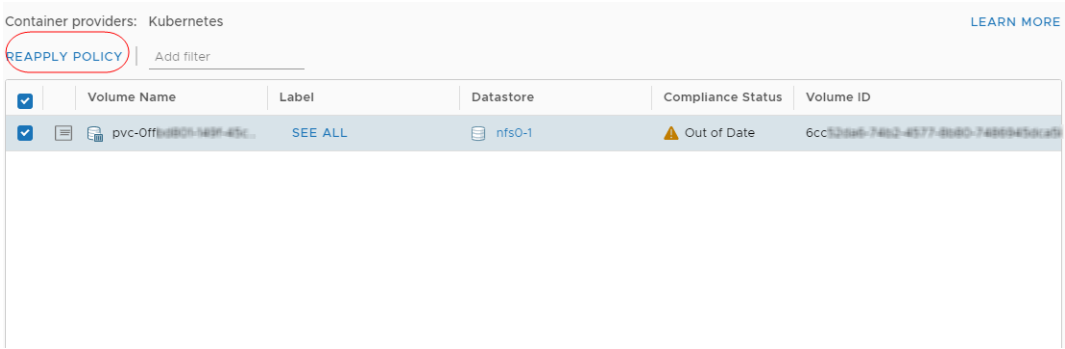
상태	설명
액세스 가능	영구 볼륨에 액세스할 수 있고 사용할 수 있습니다.
액세스할 수 없음	영구 볼륨에 액세스할 수 없고 사용할 수 없습니다. 데이터스토어에 연결하는 호스트가 볼륨을 저장하는 데이터스토어에 연결할 수 없으면 영구 볼륨에 액세스할 수 없게 됩니다.

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 클러스터의 볼륨 상태 모니터링

DevOps 엔지니어는 바인딩된 상태의 영구 볼륨 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 Kubernetes 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
```

최신 상태 저장 서비스에 vSAN 데이터 지속성 플랫폼 사용

영구 스토리지가 필요한 최신 상태 저장 서비스에 vSAN 데이터 지속성 플랫폼을 사용할 수 있습니다. 이 플랫폼은 타사가 서비스 애플리케이션을 기본 vSphere 인프라와 통합하여 타사 소프트웨어를 vSphere with Tanzu에서 최적으로 실행할 수 있도록 하는 프레임워크를 제공합니다.

vSAN 데이터 지속성 사용의 이점은 다음과 같습니다.

자동 서비스 배포 및 확장/축소

vSphere Client를 사용하면 관리자가 감독자 클러스터에 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.

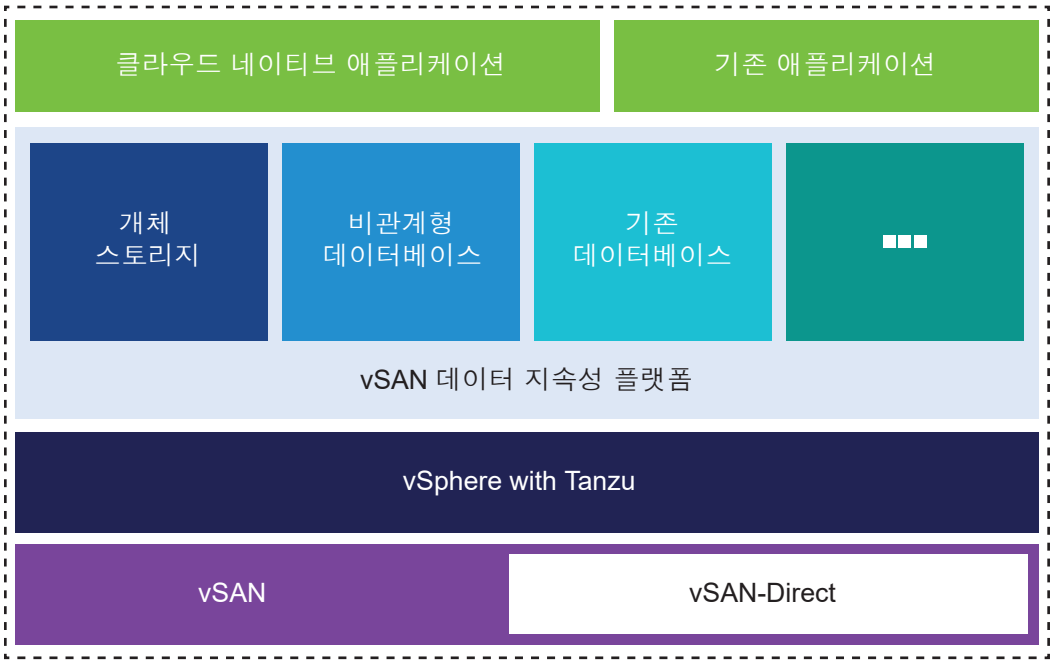
플랫폼은 다음과 같은 유형의 서비스를 지원합니다.

- 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.

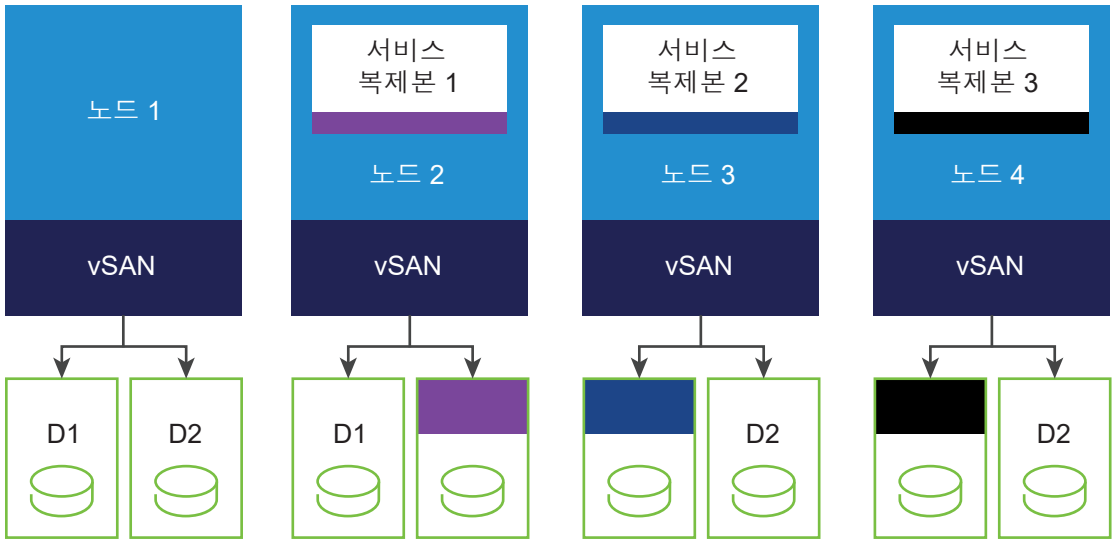
포드 같은 계산 인스턴스는 vSAN 클러스터의 노드 중 하나에 먼저 나타납니다. 그런 다음 vSAN SNA 정책으로 생성된 vSAN 개체는 포드를 실행하는 동일한 노드에 모든 데이터가 자동으로 배치됩니다.

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

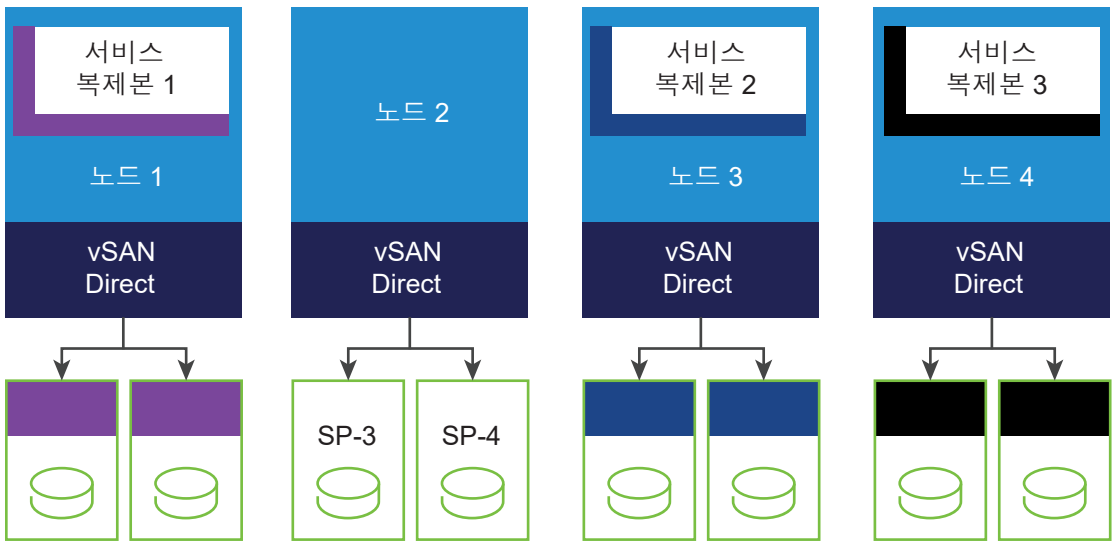


vSAN Direct

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

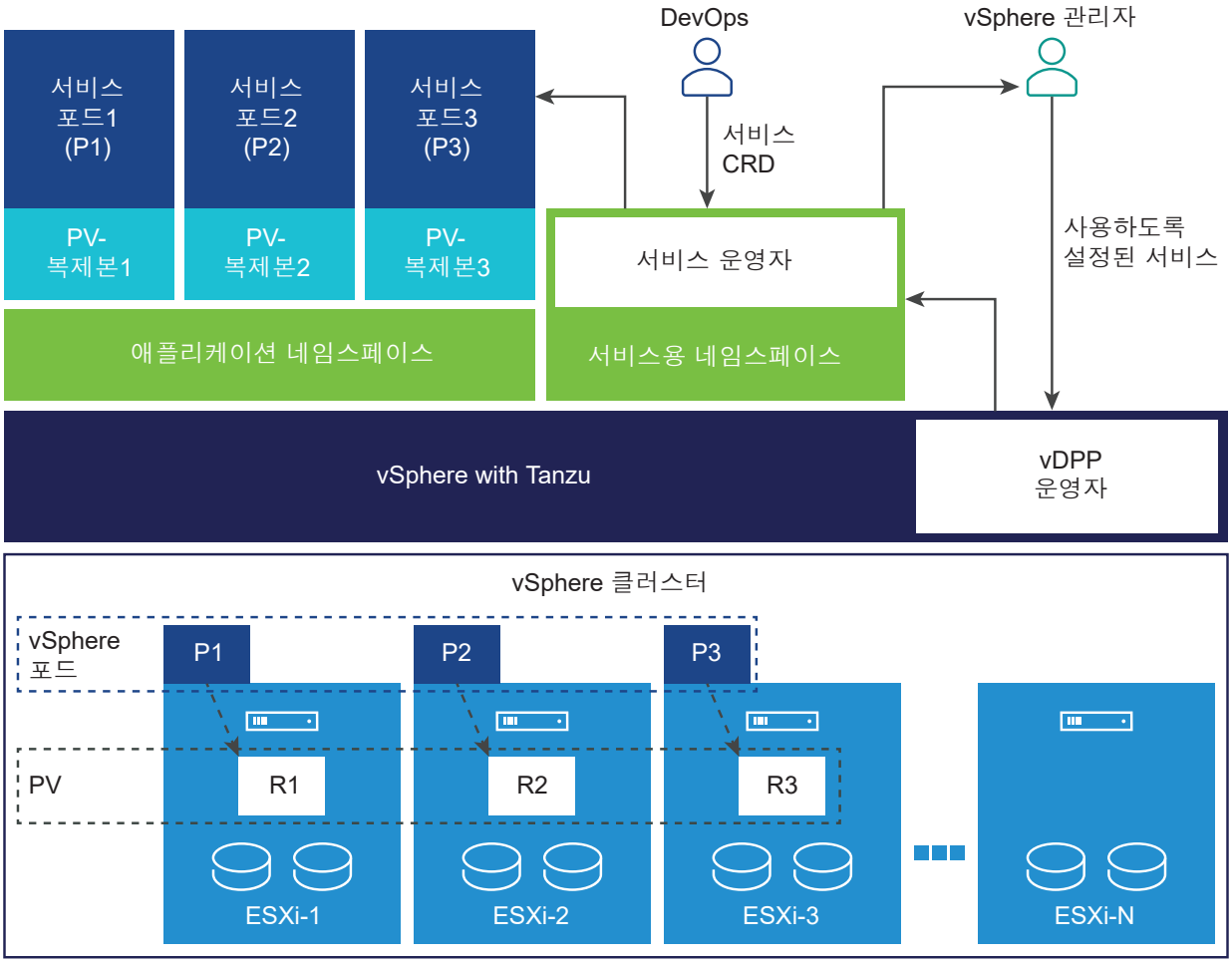
사용할 vSAN 유형을 결정할 때는 다음과 같은 일반적인 권장 사항을 따르십시오.

- 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 감독자 클러스터의 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 with Tanzu는 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

vSAN Direct에 대한 스토리지 디바이스 태그 지정

VMware Cloud Foundation 배포에서 vSAN은 ESXi 호스트의 모든 로컬 스토리지 디바이스를 자동으로 할당합니다. 디바이스를 일반 vSAN에는 부적합하고 vSAN Direct에는 사용할 수 있도록 설정할 수 있습니다.

이 항목에서는 `esxcli` 명령을 사용하여 디바이스를 vSAN Direct으로 표시하는 방법에 대해 설명합니다. 또는 스크립트를 사용할 수 있습니다. 스크립트를 사용하여 vSAN Direct용 스토리지 디바이스에 태그 지정의 내용을 참조하십시오.

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

VMware Cloud Foundation 배포에서는 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()
```

vSphere with Tanzu에 대한 vSAN Direct 설정

vSphere 관리자는 vSAN 데이터 지속성 플랫폼에 사용할 vSAN Direct를 설정합니다. ESXi 호스트에 로컬인 할당되지 않은 스토리지 디바이스를 사용합니다.

사전 요구 사항

vSAN이 배포의 모든 로컬 스토리지 디바이스를 자동으로 할당하는 경우 `esxcli` 명령을 사용하여 vSAN Direct에 사용할 수 있는 디바이스에 태그를 지정합니다. vSAN Direct에 대한 스토리지 디바이스 태그 지정의 내용을 참조하십시오. 또는 스크립트를 사용할 수 있습니다. 자세한 내용은 스크립트를 사용하여 vSAN Direct용 스토리지 디바이스에 태그 지정 항목을 참조하십시오.

절차

1 vSphere Client에서 vSAN 클러스터로 이동합니다.

2 **구성** 탭을 클릭합니다.

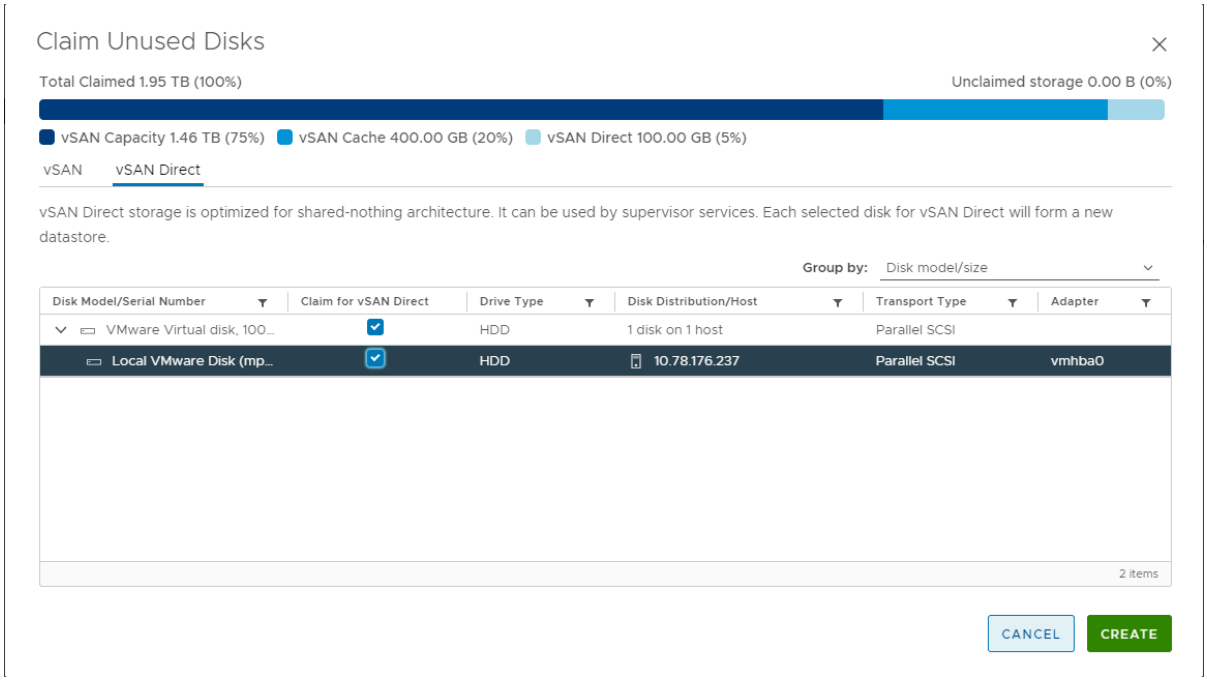
3 vSAN에서 **디스크 관리**를 클릭합니다.

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.

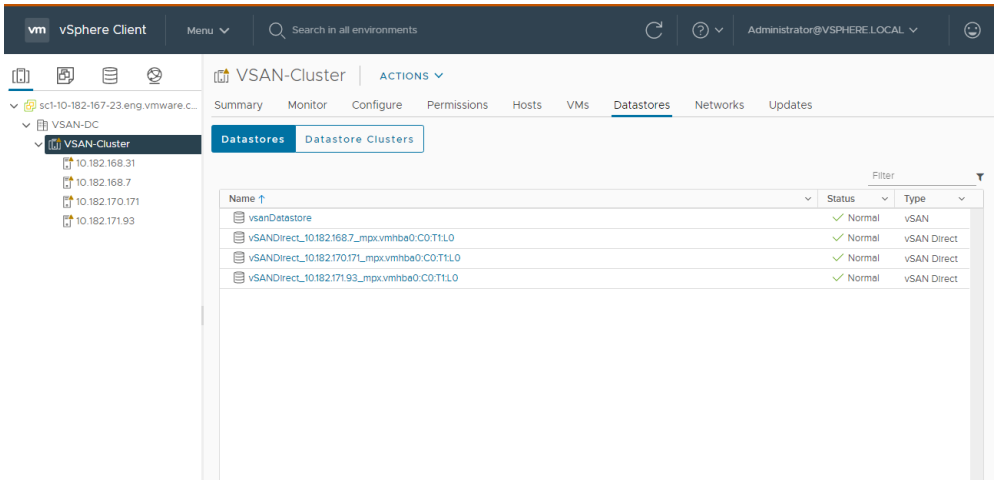
참고 일반 vSAN 데이터스토어에 대한 디바이스를 할당한 경우 해당 디바이스는 **vSAN Direct** 탭에 나타나지 않습니다.



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 vSAN Direct에서 외부 스토리지 사용의 내용을 참조하십시오.

vSphere with Tanzu에서 상태 저장 서비스 사용

vSphere with Tanzu는 해당 영구 스토리지 요구 사항에 맞게 vSAN 데이터 지속성 플랫폼을 사용하는 여러 타사 서비스와 통합됩니다. vSphere 관리자가 vCenter Server에서 서비스를 사용하도록 설정합니다.

vSphere with Tanzu 7.0 업데이트 3 릴리스부터 VMware에서 지원되는 저장소에서 사용 가능한 타사 서비스를 다운로드할 수 있습니다.

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 감독자 클러스터에 서비스를 설치합니다.

사전 요구 사항

- 필요한 권한: **감독자 서비스.감독자 서비스 관리**
- 감독자 클러스터에서 NSX-T Data Center 네트워킹 스택을 사용하는지 확인합니다. vSAN 데이터 지속성 플랫폼은 vDS(vSphere Distributed Switch) 네트워킹을 지원하지 않습니다.

  NSX-T 설정에 대한 자세한 내용은 vSphere with Tanzu에 대한 NSX-T Data Center 구성 항목을 참조하십시오.
- VMware에서 유지 보수하는 저장소에서 파트너 서비스 YAML 파일을 다운로드합니다.

  서비스 YAML 파일을 다운로드할 때는 사용하려는 vSphere 버전과 호환되는 올바른 서비스 버전을 사용해야 합니다.

  이전 버전의 파트너 서비스인 MinIO 및 Cloudian Hyperstore를 설치한 경우 vSphere를 버전 7.0 업데이트 3으로 업그레이드한 후 호환 가능한 버전으로 업그레이드합니다. 최신 버전의 파트너 운영자는 특정 문제를 해결하고 새 플랫폼 기능을 사용합니다. 자세한 내용은 파트너 설명서를 참조하십시오.

표 10-1. vSphere 및 파트너 서비스의 호환성 매트릭스

vSphere 버전	파트너 서비스	서비스 버전	Kubernetes 버전
vSphere 7.0 업데이트 3	MinIO	2.0.0	1.19, 1.20, 1.21
	Cloudian	1.2.0	1.19, 1.20, 1.21

다음 방법 중 하나를 사용하여 YAML 파일을 다운로드합니다.

- https://vmwaresaas.jfrog.io/ 저장소에서 **Artifacts > vDPP-Partner-YAML**의 적절한 파트너 폴더로 이동하고 다운로드할 YAML 파일을 선택합니다.

  최신 버전의 파트너 YAML 은 최상위 파트너 디렉토리에 있습니다.

- `wget` 또는 `curl` 명령을 사용하여 YAML 파일을 다운로드합니다.

  예:

```
wget https://vmwaresaas.jfrog.io/artifactory/vDPP-Partner-YAML/Cloudian/Hyperstore/
SupervisorService/hyperstore-supervisorservice.yaml
```

절차

1 vSAN 또는 vSAN Direct 스토리지를 구성합니다.

vSAN 스토리지 설정에 대한 자세한 내용은 "VMware vSAN 관리" 항목을 참조하십시오. vSAN Direct를 설정하려면 vSphere with Tanzu에 대한 vSAN Direct 설정의 내용을 참조하십시오.

vSAN Direct 데이터스토어는 Kubernetes에 스토리지 풀로 나타납니다.

2 vCenter Server 시스템에 상태 저장 서비스를 추가합니다.

VMware에서 유지 관리하는 저장소에서 다운로드한 파트너 서비스 YAML 파일을 사용합니다.

vCenter Server에 감독자 서비스 추가의 내용을 참조하십시오.

3 감독자 클러스터에 서비스를 설치합니다.

감독자 클러스터에 감독자 서비스 설치의 내용을 참조하십시오.

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

- 감독자 클러스터에서 이 서비스에 대한 네임스페이스가 생성됩니다.
- vSAN SNA(비공유 아키텍처) 및 vSAN Direct 데이터스토어에 사용할 기본 스토리지 정책 및 해당 스토리지 클래스를 생성합니다.

  참고 vSAN 데이터 지속성 플랫폼은 vSphere 관리자가 상태 저장 서비스를 사용하도록 설정한 후 네임스페이스에 vsan-direct 및 vsan-sna 스토리지 클래스를 자동으로 생성합니다. 감독자 클러스터에서 실행되는 애플리케이션만 vsan-direct 및 vsan-sna 스토리지 클래스를 사용할 수 있습니다. 이러한 스토리지 클래스는 Tanzu Kubernetes 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.

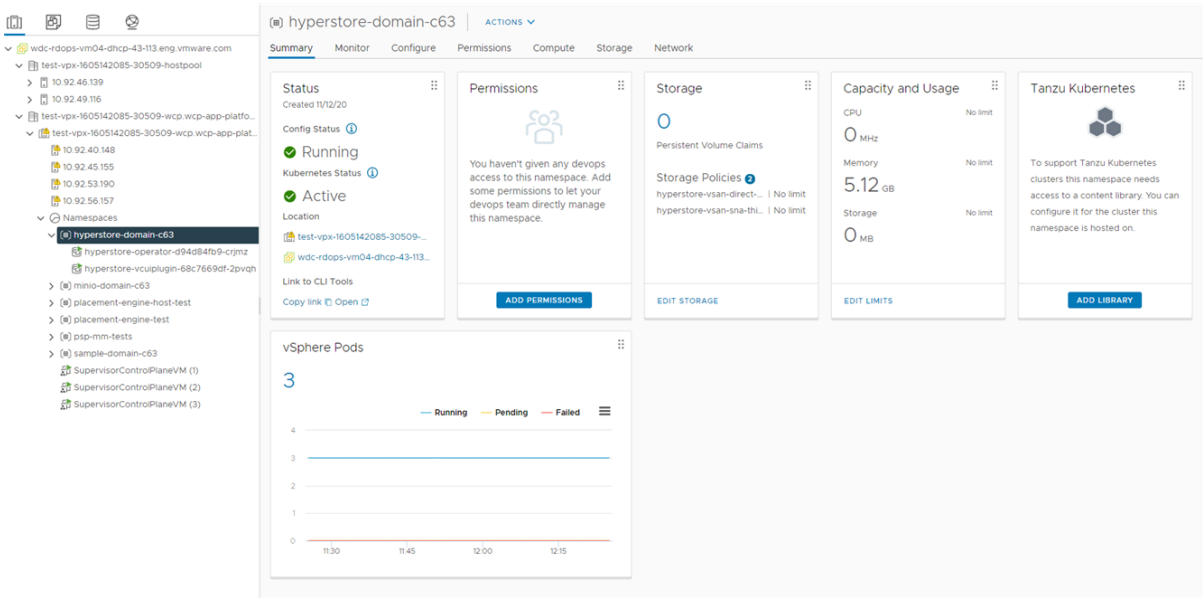
새 스토리지 정책을 생성하여 기본값 대신 서비스 네임스페이스에 할당하려면 vSAN Direct 스토리지 정책 생성 및 vSAN SNA 스토리지 정책 생성 항목을 참조하십시오.

- 편집 및 보기 권한이 있는 역할을 포함하여 DevOps 역할을 생성합니다.

  서비스 연산자가 배포되면 해당 사용자 지정 CRD가 감독자 클러스터에 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.

4 서비스에 대해 생성된 네임스페이스를 선택하고 **요약** 탭을 클릭하여 서비스에 적합한 모든 리소스가 생성되었는지 확인합니다.



다음에 수행할 작업

- DevOps 엔지니어는 `kubectl` 명령을 사용하여 서비스 네임스페이스에 액세스하고 타사 CRD를 사용하여 타사 애플리케이션 서비스의 인스턴스를 배포합니다. 자세한 내용은 타사 설명서를 참조하십시오.

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

- 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 with Tanzu에서 상태 저장 서비스 모니터링 항목을 참조하십시오.

vSphere with Tanzu에서 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 감독자 클러스터로 이동합니다.

2 **모니터** 탭을 클릭합니다.

3 사용하도록 설정된 서비스에 해당하는 네임스페이스에서 실행되는 가상 개체를 모니터링합니다.

   a **vSAN**에서 **가상 개체**를 클릭합니다.

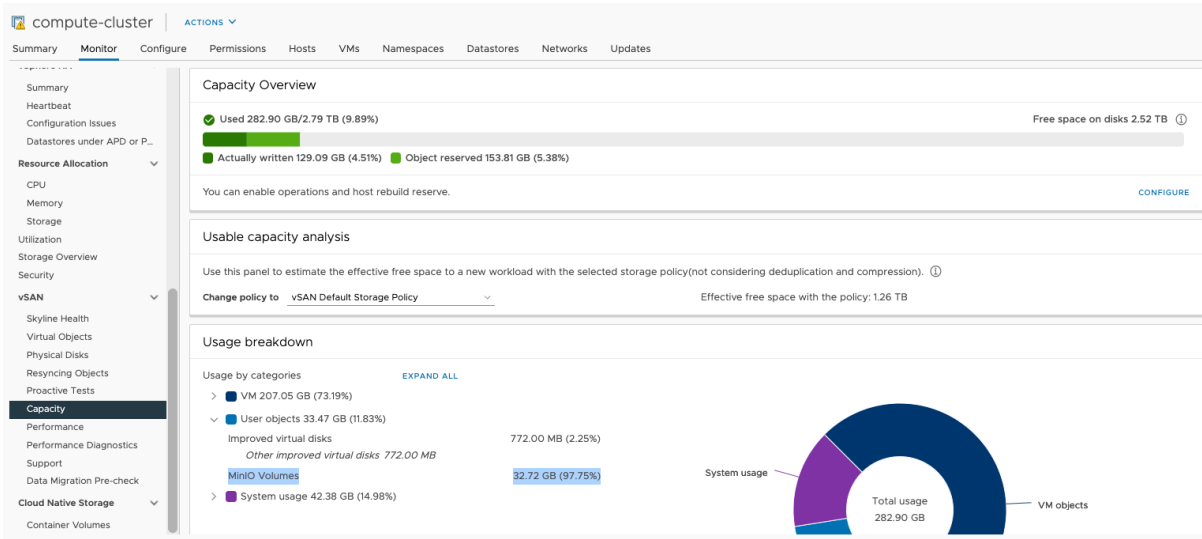
   가상 개체(예: MinIO 연산자 개체)를 찾아보고 해당 상태를 확인할 수 있습니다.

   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.

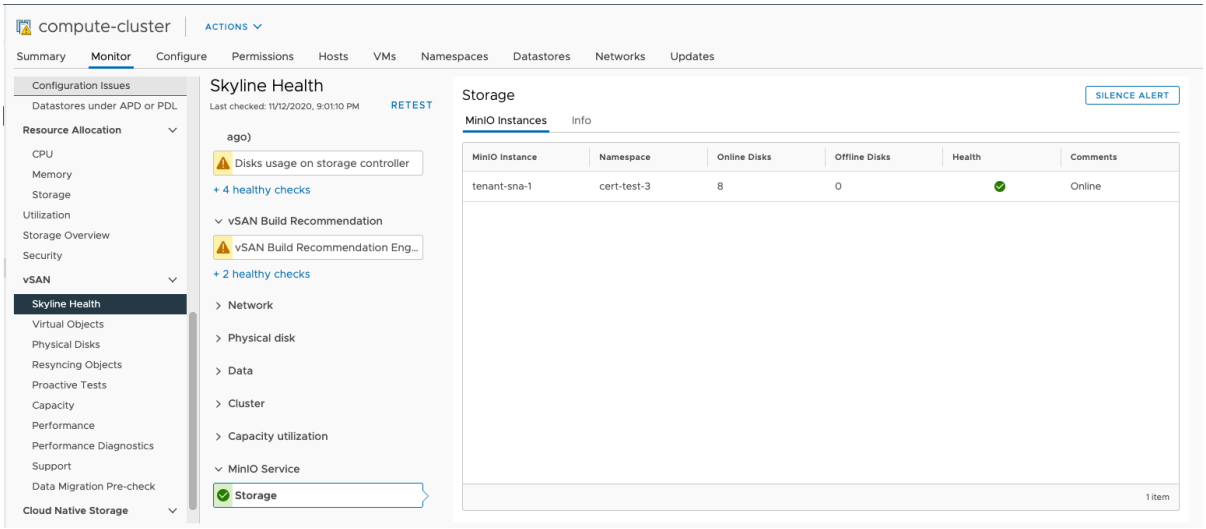
   b **사용량 분석** 창에서 **사용자 개체** 아래에 서비스 개체를 표시합니다.



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 with Tanzu 환경의 상태 저장 서비스에 사용하는 네임스페이스에 적절한 스토리지 클래스가 있는지 확인합니다. 스토리지 클래스는 vSAN SNA(비공유 아키텍처) 및 vSAN Direct 일 수 있습니다.

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

참고 감독자 클러스터에서 실행되는 애플리케이션만 vsan-direct 및 vsan-sna 스토리지 클래스를 사용할 수 있습니다. 이러한 스토리지 클래스는 Tanzu Kubernetes 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 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.

다음에 수행할 작업

정책을 생성한 후에는 상태 저장 서비스가 실행되는 네임스페이스에 정책을 할당할 수 있습니다. 네임스페이스의 스토리지 설정 변경의 내용을 참조하십시오.

vSAN Direct 스토리지 정책 생성

vSAN Direct 플랫폼과 함께 vSAN 데이터 지속성를 사용하는 경우 상태 저장 서비스가 실행되는 네임스페이스에 사용할 기능 기반 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 vSAN Direct Thick`**)을 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 vSAN Direct를 스토리지 배치 유형으로 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.

다음에 수행할 작업

정책을 생성한 후에는 상태 저장 서비스가 실행되는 네임스페이스에 정책을 할당할 수 있습니다. 네임스페이스의 스토리지 설정 변경의 내용을 참조하십시오.

vSphere 포드에 워크로드 배포

11

DevOps 엔지니어는 감독자 클러스터에서 실행 중인 네임스페이스의 리소스 경계 내에서 vSphere 포드의 수명주기를 배포하고 관리할 수 있습니다. 여기에 vSphere 포드를 배포하려면 네임스페이스에 대한 쓰기 권한이 있어야 합니다.

본 장은 다음 항목을 포함합니다.

- 감독자 클러스터 컨텍스트 가져오기 및 사용
- vSphere 네임스페이스의 vSphere 포드에 애플리케이션 배포
- 내장된 Harbor 레지스트리를 사용하여 vSphere 포드에 애플리케이션 배포
- vSphere 포드 애플리케이션 스케일 업/다운
- 기밀 vSphere 포드 배포

감독자 클러스터 컨텍스트 가져오기 및 사용

vSphere 관리자가 감독자 클러스터에서 Kubernetes 제어부의 IP 주소를 제공하면 감독자 클러스터에 로그인하여 액세스 권한이 있는 컨텍스트를 가져올 수 있습니다. 컨텍스트는 감독자 클러스터의 네임스페이스에 해당합니다.

사전 요구 사항

- vSphere 관리자로부터 감독자 클러스터에 있는 Kubernetes 제어부의 IP 주소를 가져옵니다.
- vCenter Single Sign-On에서 사용자 계정을 가져옵니다.
- vSphere 관리자에게 필요한 컨텍스트에 액세스할 수 있는 사용 권한이 있는지 확인합니다.
- 서명 CA를 신뢰 루트로 설치하거나 인증서를 신뢰 루트로 직접 추가하여 Kubernetes 제어부가 제공하는 인증서를 시스템에서 신뢰할 수 있는지 확인합니다.

절차

1 브라우저 창에서 Kubernetes 제어부의 URL을 엽니다.

2 `vsphere-plugin.zip`의 SHA256 체크섬이 `sha256sum.txt` 파일의 체크섬과 일치하는지 확인합니다.

3 vsphere-plugin.zip 파일을 시스템에 다운로드하고 운영 체제의 실행 파일 검색 경로에 설정합니다.

4 명령 프롬프트 창에서 다음 명령을 실행하여 vCenter Server에 로그인합니다.

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

5 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.

6 컨텍스트 간에 전환하려면 다음 명령을 사용합니다.

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

결과

Kubernetes 제어부의 로그인 API가 호출됩니다. 인증 프록시는 인증 요청을 vCenter Single Sign-On으로 리디렉션합니다. vCenter Server는 JSON 웹 토큰을 반환하여 kubeconfig 파일에 추가합니다. 이 토큰은 모든 새로운 kubectl 명령으로 Kubernetes 제어부에 전송되어 사용자를 인증합니다.

vSphere 네임스페이스의 vSphere 포드에 애플리케이션 배포

감독자 클러스터의 네임스페이스에 애플리케이션을 배포할 수 있습니다. 애플리케이션을 배포한 후에는 해당하는 수의 vSphere 포드가 네임스페이스 내의 감독자 클러스터에 생성됩니다.

Harbor 이미지 레지스트리에 저장된 이미지에서 애플리케이션을 배포할 수도 있습니다. 내장된 Harbor 레지스트리를 사용하여 vSphere 포드에 애플리케이션 배포의 내용을 참조하십시오.

사전 요구 사항

- vSphere 관리자로부터 감독자 클러스터에 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 감독자 클러스터로 인증합니다.

vCenter Single Sign-On 사용자로 감독자 클러스터에 연결의 내용을 참조하십시오.

2 애플리케이션을 배포하려는 컨텍스트로 전환합니다.

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

3 애플리케이션을 배포합니다.

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

내장된 Harbor 레지스트리를 사용하여 vSphere 포드에 애플리케이션 배포

Harbor 레지스트리에 저장된 이미지를 사용하여 감독자 클러스터의 네임스페이스에 vSphere 포드를 배포할 수 있습니다.

사전 요구 사항

- 애플리케이션을 배포하려는 네임스페이스와 이름이 동일한 Harbor 레지스트리의 프로젝트에 이미지를 푸시합니다. 내장된 Harbor 레지스트리로 이미지 푸시 의 내용을 참조하십시오.
- `vspere-plugin.zip`의 컨텐츠를 사용하는 환경의 실행 파일 경로에 추가합니다.

절차

1 다음 매개 변수를 포함하는 YAML 파일을 생성합니다.

```
...
namespace: <namespace-name>
...
spec:
...
image: <image registry URL>/<namespace name>/<image name>
```

2 감독자 클러스터에 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 해당 YAML 파일에서 vSphere 포드을 배포합니다.

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

5 다음 명령을 실행하여 이미지를 Harbor 레지스트리에서 가져왔고 vSphere 포드의 상태가 실행 중인지 확인합니다.

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

결과

생성한 YAML 파일은 네임스페이스에 따라 이름이 지정된 Harbor 레지스트리에서 프로젝트의 이미지를 사용하여 지정된 네임스페이스에 배포됩니다.

예제:

Harbor 레지스트리에서 demoapp1 프로젝트의 busybox 이미지를 사용하여 demoapp1 네임스페이스에 다음 YAML 파일을 생성하고 배포합니다.

```
apiVersion: v1
kind: Pod
metadata:
  name: busybox
  namespace: demoapp1
spec:
  containers:
  - name: busybox
    image: <harbor_IP>/demoapp1/busybox:latest
    command:
      - sleep
      - "3600"
    imagePullPolicy: IfNotPresent
  restartPolicy: Always
```

vSphere 포드 애플리케이션 스케일 업/다운

감독자 클러스터에서 실행 중인 각 애플리케이션의 복제본 수를 늘리거나 줄일 수 있습니다.

사전 요구 사항

- vSphere 관리자로부터 감독자 클러스터에 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 감독자 클러스터로 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 with Tanzu를 사용하면 감독자 클러스터에서 기밀 vSphere 포드를 실행할 수 있습니다. 기밀 vSphere 포드는 게스트 운영 체제 메모리를 암호화된 상태로 유지하여 하이퍼바이저의 액세스로부터 보호하는 하드웨어 기술을 사용합니다.

vSphere 7.0 업데이트 2부터는 추가적인 보안 향상으로 SEV-ES(Secure Encrypted Virtualization-Encrypted State)를 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 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. 현재 SEV-ES는 AMD EPYC 7xx2 CPU(코드명 "Rome") 이상 CPU만 지원합니다.
- 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 감독자 클러스터에 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.

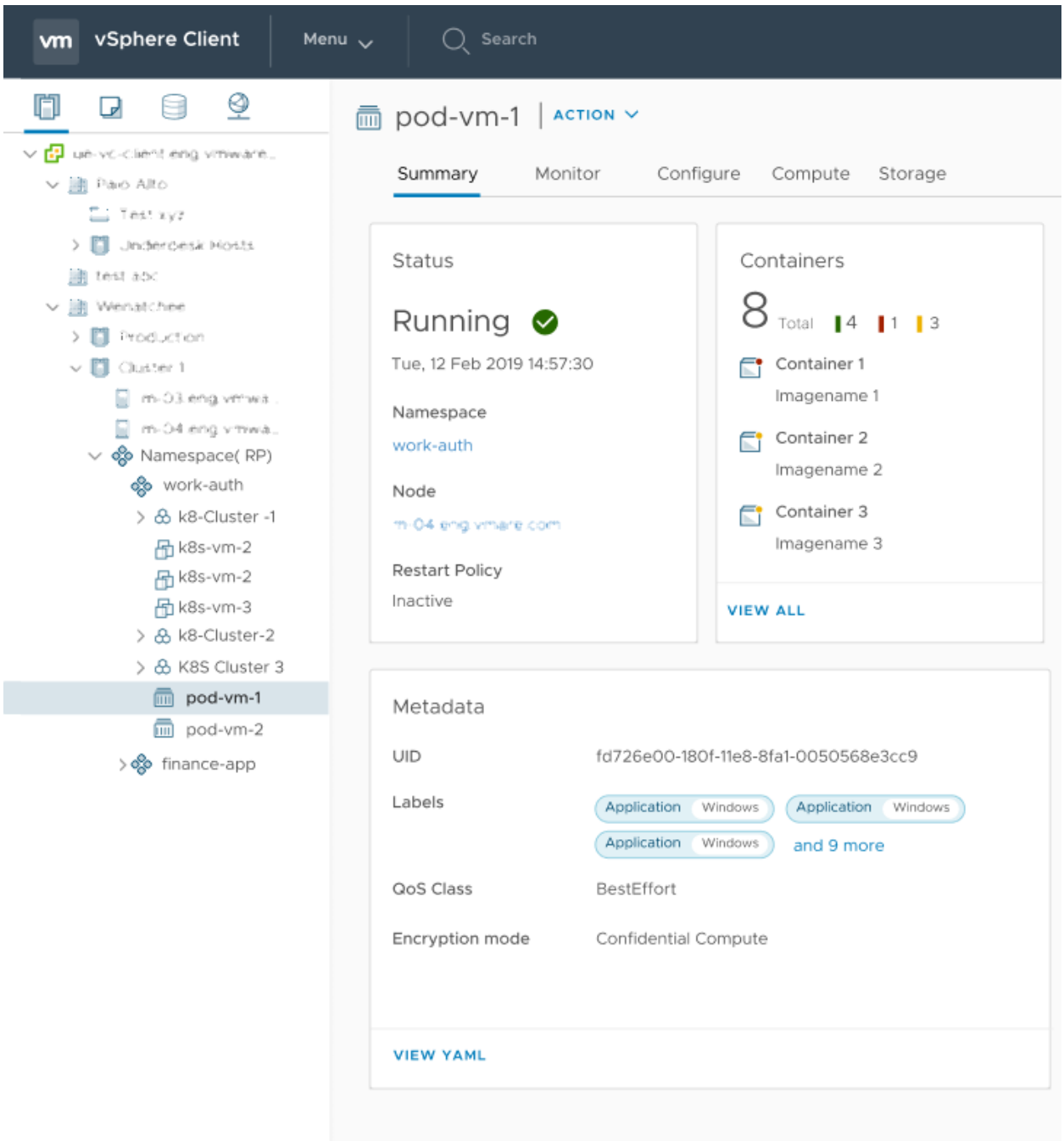
시작된 기밀 vSphere 포드은 이 포드에서 실행 중인 모든 워크로드에 대한 하드웨어 메모리 암호화 지원을 제공합니다.

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 Client
Menu
Search
Namespace(RP)
work-auth
k8-Cluster -1
k8s-vm-2
k8s-vm-2
k8s-vm-3
k8-Cluster-2
K8S Cluster 3
pod-vm-1
pod-vm-2
finance-app
pod-vm-1
ACTION
Summary
Monitor
Configure
Compute
Storage
Status
Running
Tue, 12 Feb 2019 14:57:30
Namespace
work-auth
Node
Restart Policy
Inactive
Containers
8 Total 4 1 3
Container 1
Imagename 1
Container 2
Imagename 2
Container 3
Imagename 3
VIEW ALL
Metadata
UID
fd726e00-180f-11e8-8fa1-0050568e3cc9
Labels
Application Windows
Application Windows
Application Windows
and 9 more
QoS Class
BestEffort
Encryption mode
Confidential Compute
VIEW YAML

vSphere with Tanzu에서 가상 시스템 배포 및 관리

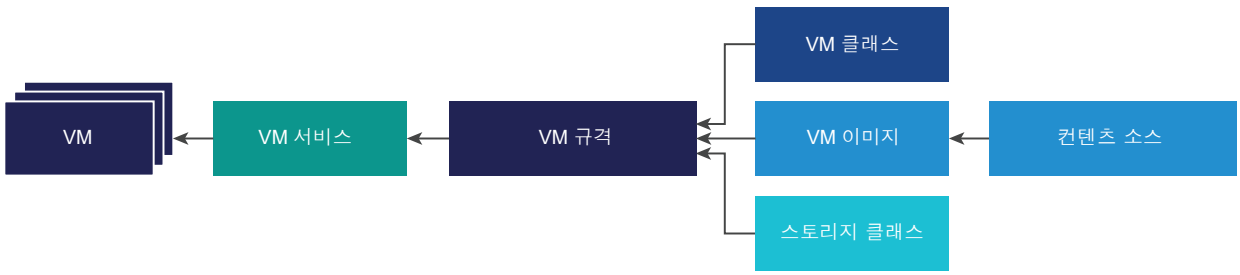
12

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

일반적으로 비즈니스 요구와 목표에 따라 컨테이너가 아닌 VM에서 워크로드를 실행하기로 결정합니다. VM 실행 시기에 대한 자세한 내용은 vSphere with Tanzu에서 가상 시스템 사용 항목을 참조하십시오.

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 클러스터를 지원하는 VM을 생성합니다.



VM 서비스

VM 서비스는 VM 및 연결된 vSphere 리소스를 관리하기 위한 선언적 Kubernetes 스타일 API를 제공하는 vSphere with Tanzu의 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 with Tanzu에서 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 감독자 클러스터의 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` 및 `VirtualMachineClassBinding` 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.

VM 서비스는 제한된 수의 VM 이미지와 게스트 운영 체제를 지원합니다. 호환되는 VM 이미지는 VMware Marketplace에 OVF로 표시됩니다. VM 서비스에서 지원되는 VM 이미지만 사용해야 합니다. 호환되는 이미지를 찾으려면 VMware Cloud Marketplace 웹 사이트에서 **VM 서비스 이미지**를 검색합니다. CentOS용 VM 서비스 이미지의 예는 CentOS용 VM 서비스 이미지를 참조하십시오.

컨텐츠 소스

DevOps 엔지니어는 컨텐츠 라이브러리를 이미지 소스로 사용하여 VM을 생성합니다. VM 클래스와 마찬가지로 vSphere 관리자는 기존 컨텐츠 라이브러리를 네임스페이스에 할당하여 DevOps 엔지니어가 사용하도록 할 수 있습니다.

스토리지 클래스

VM 서비스는 스토리지 클래스를 사용하여 가상 디스크를 배치하고 영구 볼륨을 동적으로 연결합니다. 스토리지 클래스에 대한 자세한 내용은 장 10 vSphere with Tanzu에서 영구 스토리지 사용 항목을 참조하십시오.

VM 규격

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

네트워킹

VM 서비스에는 특정 요구 사항이 없으며 vSphere with Tanzu에서 사용할 수 있는 네트워크 구성에 의존합니다. VM 서비스는 두 가지 유형의 네트워킹 즉, vSphere 네트워킹 또는 NSX-T를 지원합니다. VM이 배포되면 사용 가능한 네트워크 제공자가 정적 IP 주소를 VM에 할당합니다. 자세한 내용은 장 4 vSphere with Tanzu에 대한 네트워킹 항목을 참조하십시오.

VM 프로비저닝을 위한 vSphere 관리자 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	설명	지침
1	VM 클래스를 생성하고 관리합니다.	■ vSphere with Tanzu에서 VM 클래스 생성 ■ NVIDIA vGPU를 사용하려면 VM 클래스에서 PCI 디바이스를 구성합니다. vSphere with Tanzu의 VM 클래스에 PCI 디바이스 추가의 내용을 참조하십시오. ■ vSphere with Tanzu에서 VM 클래스 편집 또는 삭제
2	VM 클래스 집합을 네임스페이스와 연결합니다.	vSphere with Tanzu에서 VM 클래스를 네임스페이스와 연결
3	컨텐츠 라이브러리를 생성하고 관리합니다.	■ 독립형 VM에 대한 자세한 내용은 vSphere with Tanzu에서 독립형 VM에 대한 컨텐츠 라이브러리 생성 및 관리 항목을 참조하십시오. ■ Tanzu Kubernetes 클러스터에 대한 자세한 내용은 Tanzu Kubernetes 릴리스용 컨텐츠 라이브러리 생성 및 관리 항목을 참조하십시오.

단계	설명	지침
4	컨텐츠 라이브러리를 네임스페이스와 연결합니다.	■ 독립형 VM에 대한 자세한 내용은 vSphere with Tanzu에서 VM 컨텐츠 라이브러리를 네임스페이스와 연결 항목을 참조하십시오. ■ Tanzu Kubernetes 클러스터에 대한 자세한 내용은 Tanzu Kubernetes 릴리스에 대한 vSphere 네임스페이스 구성 항목을 참조하십시오.
5	스토리지 클래스를 네임스페이스와 연결합니다.	vSphere 네임스페이스 생성 및 구성
6	배포된 VM을 모니터링합니다.	vSphere with Tanzu에서 사용 가능한 가상 시스템 모니터링

VM 프로비저닝을 위한 DevOps 엔지니어 워크플로

권한이 있는 DevOps 엔지니어는 사용 가능한 VM 리소스를 검토하고 네임스페이스에 VM을 배포할 수 있습니다. `kubectl` 명령을 사용하여 다음 작업을 수행합니다.

단계	설명	지침
1	VM 클래스,이미지 및 네임스페이스와 연결된 기타 리소스를 나열합니다.	vSphere with Tanzu의 네임스페이스에서 사용 가능한 VM 리소스 보기
2	VM을 생성합니다.	■ 독립형 VM에 대한 자세한 내용은 vSphere with Tanzu에서 가상 시스템 배포 항목을 참조하십시오. ■ Tanzu Kubernetes 클러스터 VM에 대한 자세한 내용은 TKGS v1alpha2 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하는 워크플로 항목을 참조하십시오.

본 장은 다음 항목을 포함합니다.

- vSphere with Tanzu에서 VM 클래스 생성
- vSphere with Tanzu의 VM 클래스에 PCI 디바이스 추가
- vSphere with Tanzu에서 VM 클래스 편집 또는 삭제
- vSphere with Tanzu에서 VM 클래스를 네임스페이스와 연결
- vSphere with Tanzu의 네임스페이스에서 VM 클래스 관리
- vSphere with Tanzu의 네임스페이스에서 사용 가능한 VM 리소스 보기
- vSphere with Tanzu에서 가상 시스템 배포
- vSphere with Tanzu에서 사용 가능한 가상 시스템 모니터링

vSphere with Tanzu에서 VM 클래스 생성

vSphere 관리자는 vSphere with Tanzu의 네임스페이스에서 VM 배포에 사용할 사용자 지정 VM 클래스를 생성합니다. 사용자 지정 VM 클래스는 네임스페이스에서 실행되는 독립형 VM 및 Tanzu Kubernetes 클러스터를 호스팅하는 VM에서 사용할 수 있습니다.

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 with Tanzu는 몇 가지 기본 VM 클래스를 제공합니다. 이것을 그대로 사용하거나 편집 또는 삭제할 수 있습니다.

사용자 지정 VM 클래스를 생성할 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.

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 **구성** 페이지에서 일반 VM 클래스 특성을 지정합니다.

VM 클래스 특성	설명
이름	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.
vCPU 수	VM의 vCPU(가상 CPU) 수를 정의합니다. 이것은 VM 하드웨어 구성입니다. DevOps 사용자가 VM 클래스를 VM에 할당하면 이 개수는 VM에 대해 구성된 vCPU 수가 됩니다.
CPU 리소스 예약	선택적 매개 변수입니다. 가상 시스템에 보장된 최소 CPU 리소스 할당량을 지정합니다. 이 값은 백분율(%)로 표시됩니다. 값이 0%이면 CPU 예약이 없음을 정의합니다. 입력한 백분율에 모든 클러스터 노드에서 사용 가능한 최소 CPU를 곱합니다. 결과 값(MHz)은 vSphere가 VM에 대해 보장하는 CPU 리소스의 양을 지정합니다.
메모리	VM에 대해 구성된 메모리를 MB, GB 또는 TB 단위로 정의합니다. 이것은 VM 하드웨어 구성입니다. DevOps 사용자가 VM 클래스 정책을 VM에 할당하면 VM은 이 특성에 정의된 메모리 양을 받습니다. 값은 4MB에서 24TB 사이이며 4MB의 배수여야 합니다.
메모리 리소스 예약	선택적 매개 변수입니다. VM에 대해 구성된 예약된 메모리의 양을 정의합니다. 특성 값의 범위는 0~100%입니다. VM 클래스 구성에 PCI 디바이스를 추가하는 경우 매개 변수를 100%로 설정합니다.

4 (선택 사항) PCI 디바이스를 추가하려면 **구성** 페이지의 **PCI 디바이스** 드롭다운 메뉴에서 **예**를 선택하고 **다음**을 클릭합니다.

이 옵션을 선택하면 메모리 리소스 예약 값이 자동으로 100%로 변경됩니다.

요구 사항 및 추가 세부 정보는 vSphere with Tanzu의 VM 클래스에 PCI 디바이스 추가 항목을 참조하십시오.

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

다음에 수행할 작업

VM 클래스를 생성한 후 해당 매개 변수를 편집하거나 환경에서 삭제할 수 있습니다. vSphere with Tanzu에서 VM 클래스 편집 또는 삭제의 내용을 참조하십시오.

VM 클래스를 DevOps 엔지니어가 사용할 수 있도록 하려면 네임스페이스와 연결합니다. VM 클래스의 연결은 네임스페이스 수준에서 발생합니다. vSphere with Tanzu에서 VM 클래스를 네임스페이스와 연결의 내용을 참조하십시오.

vSphere with Tanzu의 VM 클래스 특성

vSphere 관리자는 vSphere 네임스페이스의 VM이 사용하는 VM(가상 시스템) 클래스를 생성하거나 편집할 수 있습니다. 각 VM 클래스에 대해 사용 가능한 특성의 하위 집합을 지정합니다.

다음 표에는 VM 클래스 내에서 정의할 수 있는 모든 특성이 나열되어 있습니다.

VM 클래스 특성	설명
이름	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.
vCPU 수	VM의 vCPU(가상 CPU) 수를 정의합니다. 이것은 VM 하드웨어 구성입니다. DevOps 사용자가 VM 클래스를 VM에 할당하면 이 개수는 VM에 대해 구성된 vCPU 수가 됩니다.
CPU 리소스 예약	선택적 매개 변수입니다. 가상 시스템에 보장된 최소 CPU 리소스 할당량을 지정합니다. 이 값은 백분율(%)로 표시됩니다. 값이 0%이면 CPU 예약이 없음을 정의합니다. 입력한 백분율에 모든 클러스터 노드에서 사용 가능한 최소 CPU를 곱합니다. 결과 값(MHz)은 vSphere가 VM에 대해 보장하는 CPU 리소스의 양을 지정합니다.
메모리	VM에 대해 구성된 메모리를 MB, GB 또는 TB 단위로 정의합니다. 이것은 VM 하드웨어 구성입니다. DevOps 사용자가 VM 클래스 정책을 VM에 할당하면 VM은 이 특성에 정의된 메모리 양을 받습니다. 값은 4MB에서 24TB 사이이며 4MB의 배수여야 합니다.
메모리 리소스 예약	선택적 매개 변수입니다. VM에 대해 구성된 예약된 메모리의 양을 정의합니다. 특성 값의 범위는 0~100%입니다. VM 클래스 구성에 PCI 디바이스를 추가하는 경우 매개 변수를 100%로 설정합니다.

vSphere with Tanzu의 VM 클래스에 PCI 디바이스 추가

vSphere with Tanzu 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.

NVIDIA GRID GPU

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.

VM용 NVIDIA vGPU를 구성할 때는 vGPU용 PCI 디바이스를 VM 클래스에 추가합니다.

NVIDIA vGPU를 사용하는 경우 다음 고려 사항이 적용됩니다.

- 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.

동적 DirectPath I/O

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.

사전 요구 사항

- VMware 호환성 가이드에서 호스트 시스템이 지원되는지 확인하고 벤더에 문의하여 호스트가 전원 및 구성 요구 사항을 충족하는지 확인합니다. ESXi 호스트에 PCI 디바이스를 설치합니다.
- NVIDIA vGPU를 구성하려면 다음 사전 요구 사항을 따릅니다.
  - vSphere 버전 7.0 업데이트 3 이상을 사용합니다.
  - **Shared Direct** 모드에서 하나 이상의 디바이스를 사용하여 ESXi 호스트 그래픽 설정을 구성합니다. 호스트 그래픽 구성을 참조하십시오.
  - 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 with Tanzu의 VM에 NVIDIA 게스트 드라이버 설치의 내용을 참조하십시오.
- PCI 패스스루 디바이스에 대한 동적 DirectPath I/O를 구성하려면 다음 사전 요구 사항을 따릅니다.
  - vSphere 버전 7.0 업데이트 3 MP01을 사용합니다.
  - PCI 디바이스를 호스트에 연결하고 이를 패스스루에 사용할 수 있는 것으로 표시합니다. PCI 디바이스를 패스스루로 표시를 참조하십시오.
- 필요한 권한:
  - **네임스페이스.클러스터 전체 구성 수정**

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

절차

1 기존 VM 클래스를 생성하거나 편집할 때 VM 클래스에 PCI 디바이스를 추가합니다.

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 **구성** 페이지에서 일반 VM 클래스 특성을 지정합니다. vSphere with Tanzu의 VM 클래스 특성의 내용을 참조하십시오. 메모리 리소스 예약 값이 100%로 설정되어 있는지 확인합니다. e PCI 디바이스를 추가하려면 **구성** 페이지의 **PCI 디바이스** 드롭다운 메뉴에서 **예**를 선택하고 **다음**을 클릭합니다.
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. 메모리 리소스 예약 값이 100%로 설정되어 있는지 확인합니다. e PCI 디바이스를 추가하려면 **구성** 페이지의 **PCI 디바이스** 드롭다운 메뉴에서 **예**를 선택하고 **다음**을 클릭합니다.

2 **PCI 디바이스** 페이지에서 **PCI 디바이스 추가** 메뉴를 확장하고 액세스 유형 및 기타 적절한 옵션을 선택한 후 **다음**을 클릭합니다.

옵션	작업
NVIDIA GRID vGPU	다음 옵션을 지정합니다. ■ **모델**. 물리적 디바이스의 이름입니다. 호스트에서 사용할 수 있는 디바이스 목록에서 디바이스를 선택합니다. ■ **GPU 공유**. 물리적 GPU가 VM 간에 공유되는 방식을 나타냅니다. 예: **시간 공유**. ■ **GPU 모드**. VM 내의 GPU 모드입니다. 예를 들어 **계산**은 고성능 컴퓨팅 애플리케이션에 최적화된 구성입니다. 반면 **Workstation**은 그래픽 사용량이 많은 워크로드에 사용됩니다. ■ **GPU 메모리**. VM당 최소 GPU 메모리(GB)입니다. ■ **vGPU 수**. VM당 vGPU 디바이스 수입니다.
동적 DirectPath IO	**PCI 디바이스** 목록에서 벤더, 모델 이름 또는 하드웨어 레이블별로 PCI 패스스루 디바이스를 선택합니다.

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

결과

VM 클래스 창의 **GPU** 태그는 VM 클래스가 GPU 지원임을 나타냅니다.

vSphere with Tanzu에서 VM 클래스 편집 또는 삭제

vSphere 관리자는 vSphere 네임스페이스에서 VM에 대한 사용자 지정 VM 클래스를 생성할 수 있습니다. VM 클래스를 생성한 후 해당 매개 변수를 편집할 수 있습니다. 또한 vSphere with Tanzu에 제공된 기본 VM 클래스를 편집할 수 있습니다. 기존 VM 클래스가 더 이상 필요하지 않으면 환경에서 삭제할 수 있습니다.

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

경고 클러스터에서 사용되는 VM 클래스를 편집한 후 Tanzu Kubernetes 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. 확장/축소에 대한 자세한 내용은 Tanzu Kubernetes Grid 서비스 v1alpha1 API를 사용하여 Tanzu Kubernetes 클러스터 확장/축소 항목을 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

- 하나 이상의 VM 클래스가 있는지 확인합니다. vSphere with Tanzu에서 VM 클래스 생성의 내용을 참조하십시오.
- 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 기존 VM 클래스를 편집하거나 삭제합니다.

옵션	설명
VM 클래스 편집	a 선택한 VM 클래스 창에서 **관리**를 클릭하고 **편집**을 클릭합니다. b VM 클래스 매개 변수를 수정합니다. vSphere with Tanzu의 VM 클래스 특성의 내용을 참조하십시오. **참고** VM 클래스의 이름은 변경할 수 없습니다.
VM 클래스 삭제	a 선택한 VM 클래스 창에서 **관리**를 클릭하고 **삭제**를 클릭합니다. b VM 클래스를 삭제할 것인지 확인합니다.

다음에 수행할 작업

DevOps 엔지니어가 VM 클래스를 사용할 수 있도록 하려면 VM 클래스를 네임스페이스와 연결합니다. VM 클래스의 연결은 네임스페이스 수준에서 발생합니다. vSphere with Tanzu에서 VM 클래스를 네임스페이스와 연결의 내용을 참조하십시오.

vSphere with Tanzu에서 VM 클래스를 네임스페이스와 연결

vSphere 관리자는 VM 클래스를 감독자 클러스터에 있는 하나 이상의 네임스페이스에 추가할 수 있습니다. 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 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 클러스터를 배포할 수 있으려면 DevOps 엔지니어에게 VM 클래스에 액세스할 수 있는 권한이 있어야 합니다. vSphere 관리자는 기본 또는 사용자 지정 VM 클래스를 Tanzu Kubernetes 클러스터가 배포된 새 네임스페이스에 명시적으로 연결해야 합니다. Tanzu Kubernetes 클러스터가 이미 프로비저닝되어 있는 기존 네임스페이스는 기본 VM 클래스에 계속해서 자동으로 액세스합니다. 하지만 사용자 지정 VM 클래스를 기존 네임스페이스와 연결할 수도 있습니다.

사전 요구 사항

VMware에서 제공하는 기본 VM 클래스를 사용하거나 새 클래스를 생성합니다. vSphere with Tanzu에서 VM 클래스 생성의 내용을 참조하십시오.

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 클러스터를 구성하는 VM에서도 사용할 수 있습니다.

다음에 수행할 작업

VM 클래스를 네임스페이스와 연결한 후, VM 클래스를 더 추가하거나 클래스를 제거하여 네임스페이스에서 게시를 취소할 수 있습니다. vSphere with Tanzu의 네임스페이스에서 VM 클래스 관리의 내용을 참조하십시오.

vSphere with Tanzu의 네임스페이스에서 VM 클래스 관리

vSphere 관리자는 VM 클래스를 감독자 클러스터에 있는 하나 이상의 네임스페이스와 연결할 수 있습니다. VM 클래스를 네임스페이스와 연결한 후, VM 클래스를 더 추가하거나 클래스를 제거하여 Kubernetes 네임스페이스에서 게시를 취소할 수 있습니다.

사전 요구 사항

- 하나 이상의 VM 클래스가 네임스페이스와 연결되어 있는지 확인합니다. vSphere with Tanzu에서 VM 클래스를 네임스페이스와 연결의 내용을 참조하십시오.
- 네임스페이스에서 VM 클래스를 제거하려면 Tanzu Kubernetes Grid 서비스에 사용되지 않는지 확인합니다. 제거하면 Tanzu Kubernetes Grid 서비스 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.

vSphere with Tanzu의 네임스페이스에서 사용 가능한 VM 리소스 보기

vSphere with Tanzu에서 독립형 VM을 배포하려면 DevOps 엔지니어에게 특정 VM 리소스에 대한 액세스 권한이 있어야 합니다. DevOps 엔지니어는 사용자가 이러한 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 클러스터 및 클러스터를 구성하는 VM을 배포하는 데 필요한 리소스에 대한 자세한 내용은 Tanzu Kubernetes 클러스터에 대한 가상 시스템 클래스 항목을 참조하십시오.

사전 요구 사항

vSphere 관리자가 다음 단계를 수행했습니다.

- 네임스페이스를 생성하여 스토리지 정책과 연결했습니다. vSphere 네임스페이스 생성 및 구성의 내용을 참조하십시오.
- 기본 또는 사용자 지정 클래스를 네임스페이스와 연결했습니다. vSphere with Tanzu에서 VM 클래스를 네임스페이스와 연결의 내용을 참조하십시오.
- 컨텐츠 라이브러리를 생성하여 네임스페이스와 연결했습니다. vSphere with Tanzu에서 VM 컨텐츠 라이브러리를 네임스페이스와 연결의 내용을 참조하십시오.

참고 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.

절차

1 Kubernetes 환경의 네임스페이스에 액세스합니다.

감독자 클러스터 컨텍스트 가져오기 및 사용의 내용을 참조하십시오.

2 네임스페이스에서 사용 가능한 VM 클래스를 보려면 다음 명령을 실행합니다.

```
kubectl get virtualmachineclassbindings
```

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 보장된 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`

보이는 출력은 다음과 유사합니다.

```
NAME                                              VERSION  OSTYPE                FORMAT
IMAGESUPPORTED  AGE
centos-stream-8-vmservice-v1alpha1-xxxxxxxxxxxxx           centos8_64Guest       ovf
true            4d3h
```

5 특정 이미지를 설명하려면 다음 명령을 사용합니다.

`kubectl describe virtualmachineimage/centos-stream-8-vmservice-v1alpha1-xxxxxxxxxxxxx`

vGPU 디바이스가 있는 VM에는 부팅 모드가 EFI로 설정된 이미지(예: CentOS)가 필요합니다. 이러한 이미지에 액세스할 수 있는지 확인합니다. 지원되는 이미지에 대한 자세한 내용은 VMware Cloud Marketplace 웹 사이트에서 **VM 서비스 이미지**를 검색하십시오.

6 스토리지 클래스에 액세스할 수 있는지 확인합니다.

```
kubectl get resourcequotas
```

자세한 내용은 vSphere 네임스페이스 또는 Tanzu Kubernetes 클러스터에서 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
```

7 워크로드 네트워킹에 vSphere Distributed Switch를 사용하는 경우 네트워크의 이름을 확보합니다.

참고 `networkType`이 **`vsphere-distributed`**인 경우 이 정보를 사용하여 VM YAML 파일에서 `networkName` 매개 변수를 지정합니다. VMware NSX-T를 사용하는 경우에는 네트워크 이름을 확보하여 지정할 필요가 없습니다.

```
kubectl get network
```

```
NAME      AGE
primary   7d2h
```

다음에 수행할 작업

이제 VM을 배포할 수 있습니다. vSphere with Tanzu에서 가상 시스템 배포의 내용을 참조하십시오.

vSphere with Tanzu에서 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

- 네임스페이스에 VM을 배포하는 데 사용할 수 있는 리소스가 있는지 확인합니다. vSphere with Tanzu의 네임스페이스에서 사용 가능한 VM 리소스 보기의 내용을 참조하십시오.
- VM용 NVIDIA vGPU 또는 기타 PCI 디바이스를 사용하는 경우 다음 고려 사항이 적용됩니다.
  - PCI 구성에서 적절한 VM 클래스를 사용해야 합니다. vSphere with Tanzu의 VM 클래스에 PCI 디바이스 추가의 내용을 참조하십시오.

- vGPU 디바이스가 있는 VM에는 부팅 모드가 EFI로 설정된 이미지(예: CentOS)가 필요합니다. 이러한 이미지에 액세스할 수 있는지 확인합니다. 지원되는 이미지에 대한 자세한 내용은 VMware Cloud Marketplace 웹 사이트에서 **VM 서비스 이미지**를 검색하십시오.
- 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.

절차

1 VM YAML 파일을 준비합니다.

파일에서 다음 매개 변수를 지정합니다.

옵션	설명
apiVersion	VM 서비스 API의 버전을 지정합니다. 예: **vmoperator.vmware.com/v1alpha1**.
kind	생성할 Kubernetes 리소스의 유형을 지정합니다. 사용 가능한 유일한 값은 **VirtualMachine**입니다.
spec.imageName	VM이 사용해야 하는 컨텐츠 라이브러리 이미지를 지정합니다. 예: **centos-stream-8-vmservice-v1alpha1-xxxxxxxxxxxxx**.
spec.storageClass	영구 볼륨의 스토리지에 사용할 스토리지 클래스를 식별합니다. 예: **wcpglobal-storage-profile**.
spec.className	사용할 가상 하드웨어 설정을 설명하는 VM 클래스의 이름을 지정합니다. 예: **custom**.
spec.networkInterfaces	VM에 대한 네트워크 관련 설정을 지정합니다. ■ networkType. 이 키의 값은 **nsx-t** 또는 **vsphere-distributed**일 수 있습니다. ■ networkName. networkType이 **vsphere-distributed**인 경우에만 이름을 지정합니다. 이 정보는 kubectl get network 명령을 사용하여 얻을 수 있습니다. networkType이 **nsx-t**이면 networkName을 지정할 필요가 없습니다.
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. 아래의 예제 YAML은 ConfigMap을 사용하여 메타데이터를 저장합니다.

YAML 파일 vmsvc-centos-vm.yaml의 예로 다음을 사용합니다.

```
apiVersion: vmoperator.vmware.com/v1alpha1
kind: VirtualMachine
metadata:
  name: vmsvc-centos-vm
  namespace: my-ns-centos
spec:
  imageName: centos-stream-8-vmservice-v1alpha1-xxxxxxxxxxxxx
  className: custom
```

```
  powerState: poweredOn
  storageClass: wcpglobal-storage-profile
  networkInterfaces:
  - networkName: primary
    networkType: vsphere-distributed
  vmMetadata:
      configMapName: vmsvc-centos-nginx-cm
      transport: OvfEnv
---
apiVersion: v1
kind: ConfigMap
metadata:
    name: vmsvc-centos-nginx-cm
    namespace: my-ns-centos
data:
  user-data: >-

I2Nsb3VkLWNvbmZpZwoKcGFzc3dvcmQ6IFZNV0FSRQpzc2hfcHdhdXRoOiB0cnVlCgp1c2VyczoKICAtIG5hbWU6IHZ
td2FyZQogICAgc3VkbzogQUxMPShBTEwpIE5PUEFTU1dEOkFMTAogICAgbG9ja19wYXNzd2Q6IGZhbHNlCiAgICAjIF
Bhc3N3b3JkIHNldCB0byBBZG1pbiEyMwogICAgcGFzc3dkOiAnJDEkc2FsdCRTT0MzM2ZWYkEvWnhlSXdENXl3MXUxJ
wogICAgc2hlbGw6IC9iaW4vYmFzaAoKd3JpdGVfZmlsZXM6CiAgLSBjb250ZW50OiB8CiAgICAgVk1TVkMgU2F5cyBIZWxsbyBXb3JsZAogICAgcGF0aDogL2hlbGxvd29ybGQKICAgIHBhdGg6IC9oZWxsb3dvcmxkCg==
```

ConfigMap에는 게스트 운영 체제의 사용자 이름 및 암호를 지정하는 cloud-config blob이 포함되어 있습니다. 이 예에서 vmsvc-centos-nginx-cm ConfigMap의 user-data는 base64 형식의 다음 코드 조각을 나타냅니다.

```
#cloud-config
password: VMWARE
ssh_pwauth: true
users:
  - name: vmware
    sudo: ALL=(ALL) NOPASSWD:ALL
    lock_passwd: false
    passwd: '$1$salt$SOC33fVbA/ZxeIwD5yw1u1'
    shell: /bin/bash
write_files:
  - content: |
      VMSVC Says Hello World
    path: /helloworld
```

cloud-config 규격에 대한 자세한 내용은 https://cloudinit.readthedocs.io/en/latest/topics/examples.html 항목을 참조하십시오.

2 VM을 배포합니다.

```
kubectl apply -f vmsvc-centos-vm.yaml
```

3 VM이 생성되었는지 확인합니다.

```
kubectl get vm -n my-ns-centos
NAME              AGE
vmsvc-centos-vm   28s
```

4 VM 및 관련 이벤트의 상태를 확인합니다.

```
kubectl describe virtualmachine vmsvc-centos-vm
```

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

```
Name:         vmsvc-centos-vm
Namespace:    my-ns-centos
Annotations:  vmoperator.vmware.com/image-supported-check: disabled
API Version:  vmoperator.vmware.com/v1alpha1
Kind:         VirtualMachine
Metadata:
  Creation Timestamp:  2021-03-23T19:07:36Z
  Finalizers:
    virtualmachine.vmoperator.vmware.com
  Generation:  1
  Managed Fields:
 ...
 ...
Spec:
  Class Name:  custom
  Image Name:  vmservice-centos-20-10-server-cloudimg-amd64
  Network Interfaces:
    Network Name:  primary
    Network Type:  vsphere-distributed
  Power State:     poweredOn
  Storage Class:   wcpglobal-storage-profile
  Vm Metadata:
    Config Map Name:  vmsvc-centos-nginx-cm
    Transport:        OvfEnv
Status:
  Bios UUID:              4218ec42-aeb3-9491-fe22-19b6f954ce38
  Change Block Tracking:  false
  Conditions:
    Last Transition Time:  2021-03-23T19:08:59Z
    Status:                True
    Type:                  VirtualMachinePrereqReady
  Host:                    10.185.240.10
  Instance UUID:           50180b3a-86ee-870a-c3da-90ddbaffc950
  Phase:                   Created
  Power State:             poweredOn
  Unique ID:               vm-73
  Vm Ip:                   10.161.75.162
Events:                    <none>
...
```

5 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 with Tanzu에서 사용 가능한 가상 시스템 모니터링의 내용을 참조하십시오.

다음에 수행할 작업

자세한 내용은 가상 시스템 프로비저닝 소개 블로그를 참조하십시오.

VM에 vGPU용으로 구성된 PCI 디바이스가 포함된 경우 NVIDIA 디스플레이 드라이버를 설치합니다. vSphere with Tanzu의 VM에 NVIDIA 게스트 드라이버 설치의 내용을 참조하십시오.

vSphere with Tanzu의 VM에 NVIDIA 게스트 드라이버 설치

vSphere with Tanzu 환경에서 VM을 생성하고 부팅한 후 VM에 NVIDIA vGPU 그래픽 드라이버를 설치하여 GPU 작업을 완전히 사용하도록 설정합니다.

사전 요구 사항

- NVIDIA vGPU 디바이스를 사용하여 VM을 생성합니다. VM은 vGPU 정의가 포함된 VM 클래스를 참조해야 합니다. vSphere with Tanzu의 VM 클래스에 PCI 디바이스 추가의 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. vSphere with Tanzu의 VM 클래스에 PCI 디바이스 추가의 내용을 참조하십시오.

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 with Tanzu에서 사용 가능한 가상 시스템 모니터링

vSphere 관리자는 vSphere Client를 사용하여 Kubernetes 환경에서 DevOps가 배포한 VM을 모니터링합니다.

VM 수명 주기는 vSphere Client에서 관리할 수 없습니다.

사전 요구 사항

DevOps 엔지니어가 VM을 배포했습니다. vSphere with Tanzu에서 가상 시스템 배포의 내용을 참조하십시오.

절차

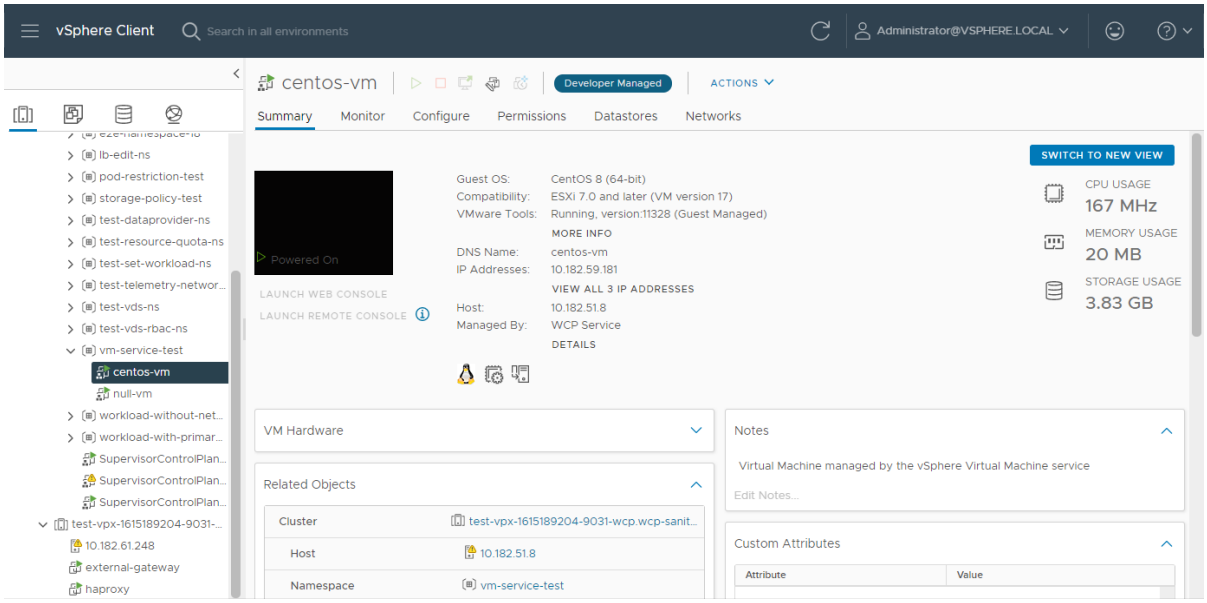
1 vSphere Client에서 vSphere with Tanzu를 사용하도록 설정한 호스트 클러스터로 이동합니다.

2 **네임스페이스**에서 VM이 배포된 네임스페이스를 확장합니다.

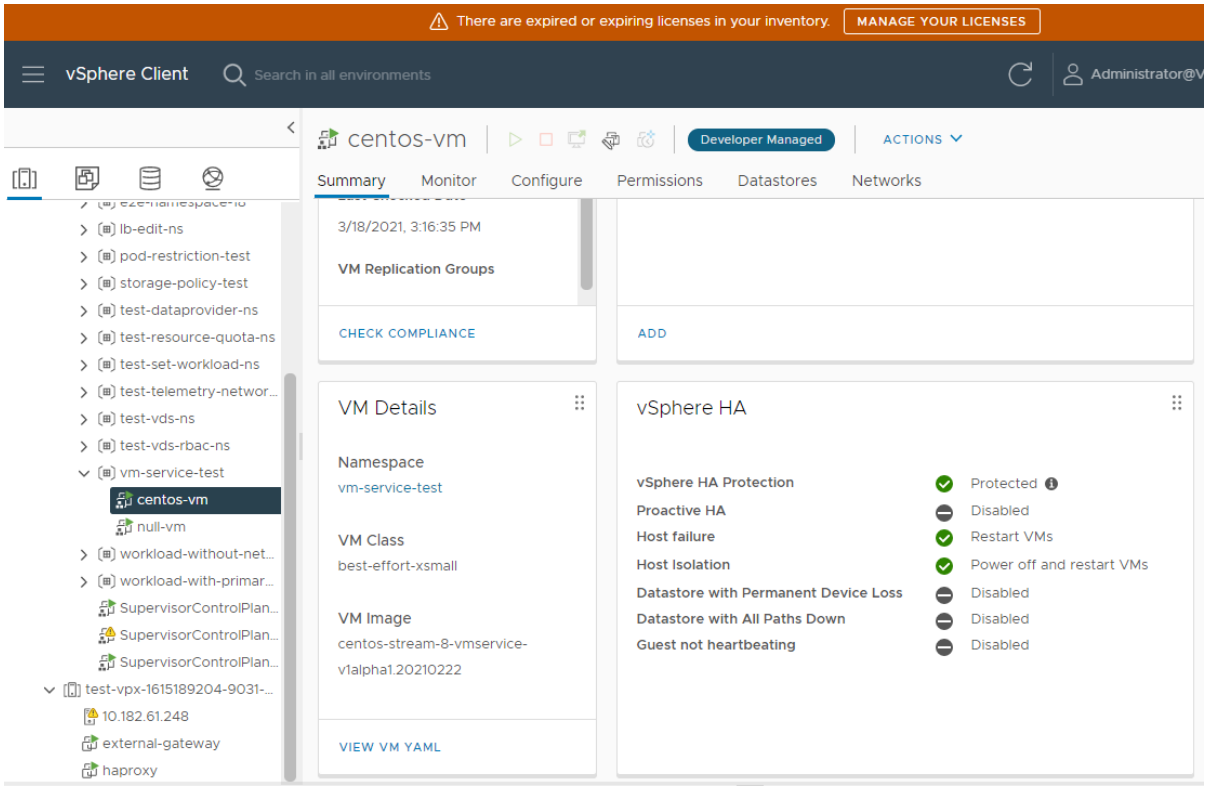
3 보려는 VM을 선택하고 **요약** 탭을 클릭합니다.

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.



TKGS 클러스터 프로비저닝 및 운영

13

vSphere with Tanzu는 Tanzu Kubernetes 클러스터를 프로비저닝하고 운영할 수 있는 편리한 도구와 간단한 워크플로를 제공합니다. 절차 및 예를 참조하여 요구 사항을 충족하는 다양한 구성으로 클러스터를 생성하고 사용자 지정할 수 있습니다.

본 장은 다음 항목을 포함합니다.

- TKGS v1alpha2 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하는 워크플로
- Tanzu Kubernetes 클러스터에 대한 가상 시스템 클래스
- TKGS v1alpha2 API를 사용하여 Tanzu Kubernetes 클러스터 프로비저닝
- Tanzu Kubernetes Grid 서비스 v1alpha1 API를 사용하여 Tanzu Kubernetes 클러스터 프로비저닝
- Tanzu Kubernetes 클러스터 삭제
- kubectl용 기본 텍스트 편집기 지정
- Tanzu Kubernetes 클러스터 운영

TKGS v1alpha2 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하는 워크플로

Tanzu Kubernetes 클러스터는 YAML에 정의된 클러스터 규격 및 kubectl을 사용하여 Tanzu Kubernetes Grid 서비스 선언적 API를 호출하여 프로비저닝합니다. 클러스터를 프로비저닝한 후 클러스터를 운영하고 kubectl을 사용하여 워크로드를 배포합니다.

이 워크플로는 Tanzu Kubernetes Grid 서비스 Tanzu Kubernetes 클러스터 프로비저닝을 위한 TKGS v1alpha2 API를 지원합니다. Tanzu Kubernetes Grid 서비스 v1alpha1 API를 사용하는 Tanzu Kubernetes 클러스터에 대한 구성 매개 변수를 사용하는 경우 해당 Tanzu Kubernetes Grid 서비스 v1alpha1 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하는 워크플로를 참조하십시오.

사전 요구 사항

워크플로 절차를 시작하기 전에 다음 사전 요구 사항이 완료되었는지 확인합니다.

- Tanzu Kubernetes Grid 서비스 Tanzu Kubernetes 클러스터 프로비저닝을 위한 TKGS v1alpha2 API를 지원하도록 환경을 설치하거나 업데이트합니다. 자세한 내용은 TKGS v1alpha2 API 사용에 대한 요구 사항을 참조하십시오. Tanzu Kubernetes 클러스터 프로비저닝을 위한 TKGS v1alpha2 API를 지원하는 최소 Tanzu Kubernetes 릴리스는 `v1.21.2`입니다. 자세한 내용은 VMware Tanzu Kubernetes 릴리스 정보를 참조하십시오.
- Tanzu Kubernetes 클러스터를 호스팅하기 위해 vSphere 네임스페이스를 구성합니다. 네임스페이스에는 DevOps 엔지니어 및 공유 스토리지에 대한 편집 권한이 필요합니다. vSphere 네임스페이스 생성 및 구성의 내용을 참조하십시오.
- Tanzu Kubernetes 릴리스용 컨텐츠 라이브러리를 생성하고 사용하려는 릴리스를 동기화합니다. Tanzu Kubernetes 릴리스용 컨텐츠 라이브러리 생성 및 관리의 내용을 참조하십시오.
- 사용할 기본 VM 클래스와 사용자 지정 VM 클래스가 필요한지 여부를 결정합니다. Tanzu Kubernetes 클러스터에 대한 가상 시스템 클래스의 내용을 참조하십시오.
- 컨텐츠 라이브러리 및 가상 시스템 클래스를 vSphere 네임스페이스와 연결합니다. Tanzu Kubernetes 릴리스에 대한 vSphere 네임스페이스 구성의 내용을 참조하십시오.

절차

1 vSphere에 대한 Kubernetes CLI 도구를 다운로드하고 설치합니다.

지침은 vSphere에 대한 Kubernetes CLI 도구 다운로드 및 설치 항목을 참조하십시오.

2 kubectl용 vSphere 플러그인을 사용하여 감독자 클러스터로 인증합니다.

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

지침은 vCenter Single Sign-On 사용자로 감독자 클러스터에 연결 항목을 참조하십시오.

3 감독자 클러스터에 성공적으로 로그인되었는지 확인합니다.

다음과 유사한 메시지가 표시됩니다.

```
Logged in successfully.

You have access to the following contexts:
   192.197.2.65
   tkgs-ns
```

여기서 `192.197.2.65`는 감독자 클러스터 컨텍스트이고 `tkgs-ns`는 Tanzu Kubernetes 클러스터를 프로비저닝할 vSphere 네임스페이스의 컨텍스트입니다.

4 대상 vSphere 네임스페이스가 현재 컨텍스트인지 확인합니다.

```
kubectl config get-contexts
```

```
CURRENT   NAME             CLUSTER         AUTHINFO                                  NAMESPACE
          192.197.2.65     192.197.2.65    wcp:192.197.2.65:user@vsphere.local
*         tkgs-ns          192.197.2.65    wcp:192.197.2.65:user@vsphere.local       tkgs-ns
```

대상 vSphere 네임스페이스가 현재 컨텍스트가 아니면 해당 컨텍스트로 전환합니다.

```
kubectl config use-context tkgs-ns
```

5 대상 vSphere 네임스페이스에서 사용할 수 있는 가상 시스템 클래스 바인딩을 나열합니다.

```
kubectl get virtualmachineclassbindings
```

대상 네임스페이스에 바인딩된 VM 클래스만 사용할 수 있습니다. VM 클래스가 표시되지 않으면 vSphere 네임스페이스에 기본 VM 클래스가 추가되어 있는지 확인합니다.

6 사용 가능한 영구 볼륨 스토리지 클래스를 가져옵니다.

```
kubectl describe storageclasses
```

7 사용 가능한 Tanzu Kubernetes 릴리스를 나열합니다.

다음 명령 중 하나를 사용하여 이 작업을 수행할 수 있습니다.

```
kubectl get tkr
```

```
kubectl get tanzukubernetesreleases
```

이 명령을 통해 반환된 릴리스만 사용할 수 있습니다. 릴리스 또는 원하는 릴리스가 표시되지 않으면 원하는 OVA 파일을 컨텐츠 라이브러리와 동기화했는지 확인합니다.

8 Tanzu Kubernetes 클러스터 프로비저닝을 위한 YAML 파일을 만듭니다.

a Tanzu Kubernetes 클러스터 프로비저닝을 위한 TKGS v1alpha2 API 규격을 검토합니다.

b 클러스터 프로비저닝을 위한 TKGS v1alpha2 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하기 위한 예제 YAML(요구 사항에 따라 기본 Tanzu Kubernetes 클러스터 프로비저닝을 위한 예제 YAML 또는 사용자 지정 Tanzu Kubernetes 클러스터 프로비저닝을 위한 예제 YAML) 중 하나로 시작합니다.

c YAML 파일을 `tkgs-cluster-1.yaml` 또는 이와 유사하게 저장합니다.

d 요구 사항을 기반으로 다음을 포함하여 이전 명령의 출력에서 수집한 정보를 사용하여 YAML 파일을 채웁니다.

- 클러스터의 이름(예: `tkgs-cluster-1`)
- 대상 vSphere 네임스페이스(예: `tkgs-ns`)
- 바인딩된 VM 클래스(예: `guaranteed-medium` 및 `guaranteed-small`)
- 클러스터 노드 및 워크로드에 대한 스토리지 클래스(예: `vwt-storage-policy`)
- 제어부 및 작업자 노드(복제본) 수
- TKR NAME 문자열로 지정된 Tanzu Kubernetes 릴리스(예: `v1.21.6---vmware.1-tkg.1.b3d708a`)

e 필요에 따라 YAML 파일을 사용자 지정합니다. 예:

- `etcd` 및 `containerd`와 같이 변동률이 높은 구성 요소에 대해 별도의 볼륨 추가
- 클러스터 노드에 대한 기본 영구 스토리지 클래스 지정
- CNI, 포드 및 서비스 CIDR을 포함한 클러스터 네트워킹 사용자 지정

이 단계의 결과는 TKGS 클러스터를 프로비저닝하기에 유효한 YAML입니다. 예:

```
apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TanzuKubernetesCluster
metadata:
  name: tkgs-cluster-1
  namespace: tkgs-ns
spec:
  topology:
    controlPlane:
      replicas: 3
      vmClass: guaranteed-medium
      storageClass: vwt-storage-policy
      volumes:
        - name: etcd
          mountPath: /var/lib/etcd
          capacity:
            storage: 4Gi
      tkr:
        reference:
          name: v1.21.6---vmware.1-tkg.1.b3d708a
    nodePools:
    - name: worker-nodepool-a1
      replicas: 3
      vmClass: guaranteed-medium
      storageClass: vwt-storage-policy
      volumes:
        - name: containerd
          mountPath: /var/lib/containerd
          capacity:
            storage: 16Gi
      tkr:
```

```
        reference:
          name: v1.21.6---vmware.1-tkg.1.b3d708a
    - name: worker-nodepool-a2
      replicas: 2
      vmClass: guaranteed-small
      storageClass: vwt-storage-policy
      tkr:
        reference:
          name: v1.21.6---vmware.1-tkg.1.b3d708a
  settings:
    storage:
      defaultClass: vwt-storage-policy
```

참고 위의 예에서는 기본 클러스터 네트워킹, 즉 클러스터 포드 및 서비스에 대한 Antrea CNI 및 기본 CIDR 범위가 사용됩니다.

9 다음 kubectl 명령을 실행하여 클러스터를 프로비저닝합니다.

```
kubectl apply -f tkgs-cluster-1.yaml
```

예상 결과:

```
tanzukubernetescluster.run.tanzu.vmware.com/tkgs-cluster-1 created
```

10 kubectl을 사용하여 클러스터 노드의 배포를 모니터링합니다.

```
kubectl get tanzukubernetesclusters
```

처음에는 클러스터가 프로비저닝 중이기 때문에 준비되지 않았습니다.

```
NAME             CONTROL PLANE   WORKER   TKR NAME                           AGE
READY   TKR COMPATIBLE   UPDATES AVAILABLE
tkgs-cluster-1   3               5        v1.21.6---vmware.1-tkg.1.b3d708a   2m4s
False   True
```

몇 분 후에 READY 상태가 True여야 합니다.

```
NAME             CONTROL PLANE   WORKER   TKR NAME                           AGE   READY
TKR COMPATIBLE   UPDATES AVAILABLE
tkgs-cluster-1   3               5        v1.21.6---vmware.1-tkg.1.b3d708a   13m   True
True
```

추가 지침은 kubectl을 사용하여 Tanzu Kubernetes 클러스터 상태 모니터링 항목을 참조하십시오.

11 vSphere Client를 사용하여 클러스터 노드의 배포를 모니터링합니다.

vSphere **호스트 및 클러스터** 인벤토리에서 가상 시스템 노드가 대상 vSphere 네임스페이스에 배포되는 것을 볼 수 있습니다.

추가 지침은 vSphere Client를 사용하여 Tanzu Kubernetes 클러스터 상태 모니터링 항목을 참조하십시오.

12 kubectl 명령을 추가로 실행하여 클러스터 프로비저닝을 확인합니다.

```
kubectl get tanzukubernetescluster,cluster-
api,virtualmachinesetresourcepolicy,virtualmachineservice,virtualmachine
```

추가 지침은 Tanzu Kubernetes 클러스터 작업 명령 사용을 참조하십시오.

문제 해결은 Tanzu Kubernetes 클러스터 문제 해결을 참조하십시오.

13 kubectl용 vSphere 플러그인을 사용하여 클러스터에 로그인합니다.

```
kubectl vsphere login --server=IP-ADDRESS --vsphere-username USERNAME \
--tanzu-kubernetes-cluster-name CLUSTER-NAME \
--tanzu-kubernetes-cluster-namespace NAMESPACE-NAME
```

예:

```
kubectl vsphere login --server=192.197.2.65 --vsphere-username user@vsphere.local \
--tanzu-kubernetes-cluster-name tkgs-cluster-1 --tanzu-kubernetes-cluster-namespace tkgs-ns
```

추가 지침은 vCenter Single Sign-On 사용자로 Tanzu Kubernetes 클러스터에 연결 항목을 참조하십시오.

14 Tanzu Kubernetes 클러스터에 성공적으로 로그인되었는지 확인합니다.

다음과 유사한 메시지가 표시됩니다.

```
Logged in successfully.

You have access to the following contexts:
   192.197.2.65
   tkgs-cluster-1
   tkgs-ns
```

여기서 192.197.2.65는 감독자 클러스터 컨텍스트이고, tkgs-ns는 vSphere 네임스페이스 컨텍스트이고, tkgs-cluster-1는 Tanzu Kubernetes 클러스터 컨텍스트입니다.

15 kubectl을 사용하여 사용 가능한 클러스터 컨텍스트를 나열합니다.

```
kubectl config get-contexts
```

예:

```
CURRENT   NAME              CLUSTER        AUTHINFO
NAMESPACE
          192.197.2.65      192.197.2.65   wcp:192.197.2.65:administrator@vsphere.local
*         tkgs-cluster-1    192.197.2.67   wcp:192.197.2.67:administrator@vsphere.local
          tkgs-ns           192.197.2.65   wcp:192.197.2.65:administrator@vsphere.local
tkgs-ns
```

필요한 경우 `kubect config use-context tkgs-cluster-1`을 사용하여 현재 컨텍스트가 되도록 Tanzu Kubernetes 클러스터로 전환합니다.

16 다음 kubectl 명령을 사용하여 클러스터 프로비저닝을 확인합니다.

```
kubectl get nodes
```

예:

```
NAME                                                STATUS   ROLES
AGE   VERSION
tkgs-cluster-1-control-plane-6ln2h                  Ready    control-plane,master
30m   v1.21.6+vmware.1
tkgs-cluster-1-control-plane-6q67n                  Ready    control-plane,master
33m   v1.21.6+vmware.1
tkgs-cluster-1-control-plane-jw964                  Ready    control-plane,master
37m   v1.21.6+vmware.1
tkgs-cluster-1-worker-nodepool-a1-4vvkb-65494d66d8-h5fp8   Ready    <none>
32m   v1.21.6+vmware.1
tkgs-cluster-1-worker-nodepool-a1-4vvkb-65494d66d8-q4g24   Ready    <none>
33m   v1.21.6+vmware.1
tkgs-cluster-1-worker-nodepool-a1-4vvkb-65494d66d8-vdcn4   Ready    <none>
33m   v1.21.6+vmware.1
tkgs-cluster-1-worker-nodepool-a2-2n22f-bd59d7b96-nh4dg    Ready    <none>
34m   v1.21.6+vmware.1
tkgs-cluster-1-worker-nodepool-a2-2n22f-bd59d7b96-vvfmf    Ready    <none>
33m   v1.21.6+vmware.1
```

17 kubectl 명령을 추가로 사용하여 클러스터 프로비저닝을 확인합니다.

```
kubectl get namespaces
```

```
kubectl get pods -A
```

```
kubectl cluster-info
```

```
kubectl api-resources
```

18 적절한 포드 보안 정책을 정의합니다.

Tanzu Kubernetes 클러스터에는 기본적으로 PodSecurityPolicy 승인 컨트롤러가 사용하도록 설정되어 있습니다. 지침은 Tanzu Kubernetes 클러스터에서 포드 보안 정책 사용 항목을 참조하십시오.

워크로드 및 사용자에 따라 시스템에서 제공된 PodSecurityPolicy에 대한 바인딩을 생성하거나 사용자 지정 PodSecurityPolicy를 생성해야 합니다. 포드 보안 정책에 대한 역할 바인딩 예의 내용을 참조하십시오.

19 예제 워크로드를 배포하고 클러스터 생성을 확인합니다.

지침은 Tanzu Kubernetes 클러스터에 워크로드 배포 항목을 참조하십시오.

20 TKG 확장을 배포하여 클러스터를 운영합니다.

지침은 Tanzu Kubernetes 클러스터에 TKG 패키지 배포 항목을 참조하십시오.

Tanzu Kubernetes 클러스터에 대한 가상 시스템 클래스

Tanzu Kubernetes 클러스터 노드의 크기를 조정하려면 가상 시스템 클래스를 지정합니다. vSphere with Tanzu는 기본 클래스를 제공하며 직접 생성할 수도 있습니다. 클래스를 사용하려면 대상 vSphere 네임스페이스와 연결하고 매니페스트에서 클래스를 참조합니다.

가상 시스템 클래스 정보

가상 시스템 클래스는 CPU 및 메모리(RAM)를 포함한 VM(가상 시스템)의 처리 능력을 위한 리소스 예약 요청입니다. 예를 들어 "guranteed-large"라는 VM 클래스 유형은 4개의 CPU와 16GB의 RAM을 예약합니다. 기본 VM 클래스 목록과 해당 CPU 및 RAM 예약은 기본 가상 시스템 클래스에서 참조하십시오.

참고 VM 디스크 크기는 VM 클래스 정의가 아닌 OVA 템플릿에 의해 설정됩니다. Tanzu Kubernetes 릴리스의 경우 디스크 크기는16GB입니다. Tanzu Kubernetes 릴리스 배포 정보의 내용을 참조하십시오.

VM 클래스에는 두 가지 예약 유형(보장됨 및 사용 시도)이 있습니다. 보장됨 클래스는 구성된 리소스를 완전히 예약합니다. 즉, 주어진 클러스터에 대해 `spec.policies.resources.requests`가 `spec.hardware` 설정과 일치합니다. 사용 시도 클래스를 사용하면 리소스가 오버 커밋될 수 있습니다. 운영 워크로드의 경우 보장됨 VM 클래스 유형을 사용하는 것이 좋습니다.

경고 사용 시도 VM 클래스 유형을 사용하면 리소스가 오버 커밋될 수 있으므로 Tanzu Kubernetes 클러스터를 프로비저닝하는 vSphere 네임스페이스에 대해 제한을 설정한 경우 리소스가 부족해질 수 있습니다. 경합이 발생하고 제어부가 영향을 받는 경우 클러스터 실행이 중지될 수 있습니다. 이러한 이유로 운영 클러스터에는 항상 보장됨 VM 클래스 유형을 사용해야 합니다. 모든 운영 노드에 대해 보장됨 VM 클래스 유형을 사용할 수 없다면 적어도 제어부 노드에 대해서는 보장됨 VM 클래스 유형을 사용해야 합니다.

가상 시스템 클래스 사용

Tanzu Kubernetes 클러스터에서 가상 시스템 클래스를 사용하려면 VM 클래스는 클러스터가 프로비저닝된 vSphere 네임스페이스 클래스에 바인딩되어야 합니다. 이렇게 하려면 클래스를 대상 네임스페이스와 연결합니다. Tanzu Kubernetes 릴리스에 대한 vSphere 네임스페이스 구성의 내용을 참조하십시오.

대상 vSphere 네임스페이스에서 사용할 수 있는 VM 클래스를 나열하려면 `kubectl get virtualmachineclassbinding` 명령을 사용합니다. 감독자 클러스터에 있는 모든 가상 시스템 클래스를 보려면 `kubectl describe virtualmachineclasses` 명령을 실행합니다. 단, 바인딩된 클래스만 클러스터를 프로비저닝하는 데 사용할 수 있기 때문에 후자의 명령은 정보 제공용입니다. TKGS v1alpha2 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하는 워크플로의 내용을 참조하십시오.

참고 VM 클래스를 vSphere 네임스페이스와 연결하기 위한 요구 사항은 새 클러스터에만 적용됩니다. 기본 VM 클래스를 사용하는 기존 Tanzu Kubernetes 클러스터는 네임스페이스 연결없이 계속 작동됩니다.

기본 가상 시스템 클래스

표 13-1. 기본 가상 시스템 클래스 표에는 Tanzu Kubernetes 클러스터 노드의 VM 배포 크기로 사용되는 가상 시스템 클래스 유형이 나열되어 있습니다.

리소스 오버 커밋을 방지하려면 운영 워크로드에서 보장된 클래스 유형을 사용해야 합니다. 메모리 부족 문제를 방지하려면 모든 환경(개발, 테스트 또는 운영)에서 워크로드를 배포하는 작업자 노드에 대해 소형 또는 초소형 클래스 크기를 사용하지 마십시오.

표 13-1. 기본 가상 시스템 클래스

클래스	CPU	메모리(GB)	예약된 CPU 및 메모리
guaranteed-8xlarge	32	128	예
best-effort-8xlarge	32	128	아니요
guaranteed-4xlarge	16	128	예
best-effort-4xlarge	16	128	아니요
guaranteed-2xlarge	8	64	예
best-effort-2xlarge	8	64	아니요
guaranteed-xlarge	4	32	예
best-effort-xlarge	4	32	아니요
guaranteed-large	4	16	예
best-effort-large	4	16	아니요
guaranteed-medium	2	8	예
best-effort-medium	2	8	아니요
guaranteed-small	2	4	예
best-effort-small	2	4	아니요
guaranteed-xsmall	2	2	예
best-effort-xsmall	2	2	아니요

사용자 지정 가상 시스템 클래스

vSphere with Tanzu는 Tanzu Kubernetes 클러스터에서 사용할 사용자 지정 가상 시스템 클래스를 지원합니다. 사용자 지정 VM 클래스를 정의한 후에 대상 vSphere 네임스페이스와 연결해야 클러스터에서 사용할 수 있습니다. vSphere with Tanzu에서 VM 클래스 생성의 내용을 참조하십시오.

가상 시스템 클래스 편집

VM 클래스 정의는 변경할 수 없습니다. Tanzu Kubernetes 클러스터에 대한 가상 시스템 클래스를 포함하여 모든 VM 클래스는 vSphere with Tanzu에서 VM 클래스 편집 또는 삭제할 수 있습니다. VM 클래스를 편집해도 기존 Tanzu Kubernetes 클러스터 노드는 영향을 받지 않습니다. 새 Tanzu Kubernetes 클러스터는 수정된 클래스 정의를 사용합니다.

경고 Tanzu Kubernetes 클러스터에서 사용 중인 VM 클래스를 편집하고 해당 클러스터를 확장하는 경우 새 노드는 편집된 클래스 정의를 사용하지만 기존 노드는 초기 클래스 정의를 사용하기 때문에 클래스 불일치가 발생합니다.

TKGS v1alpha2 API를 사용하여 Tanzu Kubernetes 클러스터 프로비저닝

이 섹션에서는 TKGS v1alpha2 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하는 방법을 설명합니다.

TKGS v1alpha2 API 사용에 대한 요구 사항

Tanzu Kubernetes 클러스터 프로비저닝에 Tanzu Kubernetes Grid 서비스 v1alpha2 API를 사용하려면 전체 요구 사항 목록을 준수합니다.

TKGS v1alpha2 API 사용에 대한 요구 사항

Tanzu Kubernetes Grid 서비스 v1alpha2 API는 Tanzu Kubernetes 클러스터 프로비저닝을 위해 향상된 기능을 제공합니다. 자세한 내용은 Tanzu Kubernetes 클러스터 프로비저닝을 위한 TKGS v1alpha2 API의 내용을 참조하십시오.

Tanzu Kubernetes Grid 서비스 v1alpha2 API에서 제공하는 새로운 기능을 활용하려면 환경이 다음 요구 사항을 충족해야 합니다.

요구 사항	참조
워크로드 관리가 지원되는 네트워킹(NSX-T Data Center 또는 네이티브 vSphere vDS)을 통해 사용되도록 설정됩니다. **참고** 특정 기능에는 특정 유형의 네트워킹이 필요할 수 있습니다. 그러한 경우 해당 기능에 대한 주제에서 언급됩니다.	vSphere 클러스터에서 vSphere with Tanzu를 구성하기 위한 사전 요구 사항의 내용을 참조하십시오. NSX-T Data Center 네트워킹으로 워크로드 관리 사용의 내용을 참조하십시오. vSphere 네트워킹으로 워크로드 관리 사용의 내용을 참조하십시오.
워크로드 관리를 호스팅하는 vCenter Server가 버전 7 업데이트 3 이상으로 업데이트되었습니다.	**vCenter Server 업데이트 및 패치 릴리스** 릴리스 정보를 참조하십시오. 업데이트 지침은 vCenter Server Appliance 업그레이드를 참조하십시오.
워크로드 관리를 사용하도록 설정된 vCenter Server 클러스터를 지원하는 모든 ESXi 호스트가 버전 7 업데이트 3 이상으로 업데이트되었습니다.	**ESXi 업데이트 및 패치 릴리스 정보**를 참조하십시오. 업데이트 지침은 ESXi 호스트 업그레이드를 참조하십시오.

요구 사항	참조
vSphere 네임스페이스가 `v0.0.11` 이상으로 업데이트되었습니다.	릴리스 세부 정보는 vSphere with Tanzu 릴리스 정보를 참조하십시오. 업데이트 지침은 장 17 vSphere with Tanzu 환경 업데이트 항목을 참조하십시오.
감독자 클러스터가 `v1.21.0+vmware.wcp.2` 이상으로 업데이트되었습니다.	릴리스 세부 정보는 vSphere with Tanzu 릴리스 정보를 참조하십시오. 업데이트 지침은 vSphere 네임스페이스 업데이트를 수행하여 감독자 클러스터 업데이트 항목을 참조하십시오.
Tanzu Kubernetes 릴리스 `v1.21.2---vmware.1-tkg.1.ee25d55` 이상을 사용해야 합니다.	릴리스 세부 정보는 업데이트를 위한 Tanzu Kubernetes 클러스터 호환성 확인 항목을 참조하십시오. 새 클러스터 프로비저닝에 대한 지침은 TKGS v1alpha2 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하기 위한 예제 YAML 항목을 참조하십시오. 기존 클러스터 업데이트에 대한 지침은 클러스터 규격이 TKGS v1alpha2 API로 변환된 후 Tanzu Kubernetes 릴리스 업데이트 항목을 참조하십시오.
노드 제한에 대한 CNI 고려 사항	클러스터 규격 설정 `spec.settings.network.pods.cidrBlocks`의 기본값은 192.168.0.0/16입니다. Tanzu Kubernetes 클러스터 프로비저닝을 위한 TKGS v1alpha2 API의 내용을 참조하십시오. 사용자 지정하는 경우 최소 포드 CIDR 블록 크기는 /24입니다. 하지만 `pods.cidrBlocks` 서브넷 마스크를 /16 이상으로 제한하는 경우에는 주의해야 합니다. TKGS는 각 클러스터 노드에 `pods.cidrBlocks`에서 분할된 /24 서브넷을 할당합니다. 이러한 할당은 클러스터의 노드 CIDR에 대한 서브넷 마스크 크기를 설정하는 NodeCIDRMaskSize라는 Kubernetes 컨트롤러 관리자 > NodeIPAMController 매개 변수에 의해 정의합니다. IPv4의 경우 기본 노드 서브넷 마스크는 /24입니다. 클러스터의 각 노드는 `pods.cidrBlocks`에서 /24 서브넷을 가져오기 때문에 프로비저닝 중인 클러스터에 너무 제한적인 서브넷 마스크 크기를 사용하면 노드 IP 주소가 부족해질 수 있습니다. 다음 노드 제한은 Antrea 또는 Calico CNI로 프로비저닝된 Tanzu Kubernetes 클러스터에 적합합니다. /16 == 최대 150개 노드(ConfigMax당) /17 == 최대 128개 노드 /18 == 최대 64개 노드 /19 == 최대 32개 노드 /20 == 최대 16개 노드 /21 == 최대 8개 노드 /22 == 최대 4개 노드 /23 == 최대 2개 노드 /24 == 최대 1개 노드

Tanzu Kubernetes 클러스터 프로비저닝을 위한 TKGS v1alpha2 API

Tanzu Kubernetes Grid 서비스 v1alpha2 API를 사용하면 Tanzu Kubernetes 클러스터를 선언적으로 프로비저닝할 수 있습니다. 클러스터를 생성하고 사용자 지정하려면 모든 매개 변수 및 사용 지침의 목록과 설명을 참조하십시오.

Tanzu Kubernetes 클러스터 프로비저닝을 위한 Tanzu Kubernetes Grid 서비스 v1alpha2 API 규격

YAML 규격에는 Tanzu Kubernetes Grid 서비스 v1alpha2 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하는 데 사용할 수 있는 모든 매개 변수가 나열됩니다.

```
apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TanzuKubernetesCluster
metadata:
  name: string
  namespace: string
spec:
  topology:
    controlPlane:
      replicas: int32
      vmClass: string
      storageClass: string
      volumes:
        - name: string
          mountPath: string
          capacity:
            storage: size in GiB
      tkr:
        reference:
          name: string
      nodeDrainTimeout: string
    nodePools:
    - name: string
      labels: map[string]string
      taints:
        -  key: string
           value: string
           effect: string
           timeAdded: time
      replicas: int32
      vmClass: string
      storageClass: string
      volumes:
        - name: string
          mountPath: string
          capacity:
            storage: size in GiB
      tkr:
        reference:
          name: string
      nodeDrainTimeout: string
  settings:
    storage:
```

```
      classes: [string]
      defaultClass: string
    network:
      cni:
        name: string
      pods:
        cidrBlocks: [string]
      services:
        cidrBlocks: [string]
      serviceDomain: string
      proxy:
        httpProxy: string
        httpsProxy: string
        noProxy: [string]
      trust:
        additionalTrustedCAs:
          - name: string
            data: string
```

Tanzu Kubernetes 클러스터 프로비저닝을 위한 주석이 달린 Tanzu Kubernetes Grid 서비스 v1alpha2 API 규격

주석이 달린 YAML규격에는 각 필드에 대한 설명서와 함께 Tanzu Kubernetes Grid 서비스 v1alpha2 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하는 데 사용할 수 있는 모든 매개 변수가 나열됩니다.

참고 현재는 모든 `tkr.reference.name` 필드가 일치해야 합니다. 향후에는 노드 풀에 대한 다양한 Tanzu Kubernetes 릴리스가 지원될 수 있습니다.

```
apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TanzuKubernetesCluster
#metadata defines cluster information
metadata:
  #name for this Tanzu Kubernetes cluster
  name: string
  #namespace vSphere Namespace where to provision this cluster
  namespace: string
#spec defines cluster configuration
spec:
  #topology describes the number, purpose, organization
  #of nodes and the resources allocated for each
  #nodes are grouped into pools based on their purpose
  #`controlPlane` is special kind of a node pool
  #`nodePools` is for groups of worker nodes
  #each node pool is homogeneous: its nodes have the same
  #resource allocation and use the same storage
  topology:
    #controlPlane defines the topology of the cluster
    #controller, including the number of nodes and
    #the resources allocated for each
    #control plane must have an odd number of nodes
    controlPlane:
      #replicas is the number of nodes in the pool
```

```
      #the control plane can have 1 or 3 nodes
      #defaults to 1 if `nil`
      replicas: int32
      #vmClass is the name of the VirtualMachineClass
      #which describes the virtual hardware settings
      #to be used for each node in the node pool
      #vmClass controls the CPU and memory available
      #to the node and the requests and limits on
      #those resources; to list available vm classes run
      #`kubectl describe virtualmachineclasses`
      vmClass: string
      #storageClass to be used for storage of the disks
      #which store the root filesystems of the nodes
      #to list available storage classes run
      #`kubectl describe storageclasses`
      storageClass: string
      #volumes is the optional set of PVCs to create
      #and attach to each node; use for high-churn
      #control plane components such as etcd
      volumes:
        #name of the PVC to be used as the suffix (node.name)
        - name: string
          #mountPath is the directory where the volume
          #device is mounted; takes the form /dir/path
          mountPath: string
          #capacity is the PVC capacity
          capacity:
            #storage to be used for the disk
            #volume; if not specified defaults to
            #`spec.controlPlane.storageClass`
            storage: size in GiB
      #tkr.reference.name is the TKR NAME
      #to be used by control plane nodes; supported
      #format is `v1.21.2---vmware.1-tkg.1.ee25d55`
      #currently all `tkr.reference.name` fields must match
      tkr:
        reference:
          name: string
      #nodeDrainTimeout is the total amount of time
      #the controller will spend draining a node
      #the default value is 0 which means the node is
      #drained without any time limit
      nodeDrainTimeout: string
    #nodePools is an array that describes a group of
    #worker nodes in the cluster with the same configuration
    nodePools:
    #name of the worker node pool
    #must be unique in the cluster
    - name: string
      #labels are an optional map of string keys and values
      #to organize and categorize objects
      #propagated to the created nodes
      labels: map[string]string
      #taints specifies optional taints to register the
      #Node API object with; user-defined taints are
```

```
      #propagated to the created nodes
      taints:
        #key is the taint key to be applied to a node
        -  key: string
        #value is the taint value corresponding to the key
           value: string
        #effect is the effect of the taint on pods
        #that do not tolerate the taint; valid effects are
        #`NoSchedule`, `PreferNoSchedule`, `NoExecute`
           effect: string
        #timeAdded is the time when the taint was added
        #only written by the system for `NoExecute` taints
           timeAdded: time
      #replicas is the number of nodes in the pool
      #worker nodePool can have from 0 to 150 nodes
      #value of `nil` means the field is not reconciled,
      #allowing external services like autoscalers
      #to choose the number of nodes for the nodePool
      #by default CAPI's `MachineDeployment` will pick 1
      #NOTE: a cluster provisioned with 0 worker nodes/nodepools
      #is not assigned any load balancer services
      replicas: int32
      #vmClass is the name of the VirtualMachineClass
      #which describes the virtual hardware settings
      #to be used for each node in the pool
      #vmClass controls the CPU and memory available
      #to the node and the requests and limits on
      #those resources; to list available vm classes run
      #`kubectl describe virtualmachineclasses`
      vmClass: string
      #storageClass to be used for storage of the disks
      #which store the root filesystems of the nodes
      #to list available storage classes run
      #`kubectl describe ns`
      storageClass: string
      #volumes is the optional set of PVCs to create
      #and attach to each node for high-churn worker node
      #components such as the container runtime
      volumes:
        #name of this PVC to be used as the suffix (node.name)
        - name: string
          #mountPath is the directory where the volume
          #device is mounted; takes the form /dir/path
          mountPath: string
          #capacity is the PVC capacity
          capacity:
            #storage to be used for the disk
            #volume; if not specified defaults to
            #`topology.nodePools[*].storageClass`
            storage: size in GiB
      #tkr.reference.name points to the TKR NAME
      #to be used by `spec.topology.nodePools[*]` nodes; supported
      #format is `v1.21.2---vmware.1-tkg.1.ee25d55`
      #currently all `tkr.reference.name` fields must match
      tkr:
```

```
        reference:
          name: string
      #nodeDrainTimeout is the total amount of time
      #the controller will spend draining a node
      #the default value is 0 which means the node is
      #drained without any time limit
      nodeDrainTimeout: string
  #settings are optional runtime configurations
  #for the cluster, including persistent storage
  #for pods and node network customizations
  settings:
    #storage defines persistent volume (PV) storage entries
    #for container workloads; note that the storage used for
    #node disks is defined by `topology.controlPlane.storageClass`
    #and by `spec.topology.nodePools[*].storageClass`
    storage:
      #classes is a list of persistent volume (PV) storage
      #classes to expose for container workloads on the cluster
      #any class specified must be associated with the
      #vSphere Namespace where the cluster is provisioned
      #if omitted, all storage classes associated with the
      #namespace will be exposed in the cluster
      classes: [string]
      #defaultClass treats the named storage class as the default
      #for the cluster; because all namespaced storage classes
      #are exposed if specific `classes` are not named,
      #classes is not required to specify a defaultClass
      #many workloads, including TKG Extensions and Helm,
      #require a default storage class
      #if omitted, no default storage class is set
      defaultClass: string
    #netowrk defines custom networking for cluster workloads
    network:
      #cni identifies the CNI plugin for the cluster
      #use to override the default CNI set in the
      #tkgservicesonfiguration spec, or when customizing
      #network settings for the default CNI
      cni:
        #name is the name of the CNI plugin to use; supported
        #values are `antrea`, `calico`, `antrea-nsx-routed`
        name: string
      #pods configures custom networks for pods
      #defaults to 192.168.0.0/16 if CNI is `antrea` or `calico`
      #defaults to empty if CNI is `antrea-nsx-routed`
      #custom subnet size must equal or exceed /24
      #use caution before seeting CIDR range other than /16
      #cannot overlap with Supervisor Cluster workload network
      pods:
        #cidrBlocks is an array of network ranges; supplying
        #multiple ranges may not be supported by all CNI plugins
        cidrBlocks: [string]
      #services configures custom network for services
      #defaults to 10.96.0.0/12
      #cannot overlap with Supervisor Cluster workload network
      services:
```

```
    #cidrBlocks is an array of network ranges; supplying
    #multiple ranges many not be supported by all CNI plugins
    cidrBlocks: [string]
  #serviceDomain specifies the service domain for the cluster
  #defaults to `cluster.local`
  serviceDomain: string
  #proxy configures proxy server to be used inside the cluster
  #if omitted no proxy is configured
  proxy:
    #httpProxy is the proxy URI for HTTP connections
    #to endpoints outside the cluster
    #takes form `http://<user>:<pwd>@<ip>:<port>`
    httpProxy: string
    #httpsProxy is the proxy URL for HTTPS connections
    #to endpoints outside the cluster
    #takes the frorm `http://<user>:<pwd>@<ip>:<port>`
    httpsProxy: string
    #noProxy is the list of destination domain names, domains,
    #IP addresses, and other network CIDRs to exclude from proxying
    #must include Supervisor Cluster Pod, Egress, Ingress CIDRs
    noProxy: [string]
  #trust configures additional certificates for the cluster
  #if omitted no additional certificate is configured
  trust:
    #additionalTrustedCAs are additional trusted certificates
    #can be additional CAs or end certificates
    additionalTrustedCAs:
      #name is the name of the additional trusted certificate
      #must match the name used in the filename
      - name: string
        #data holds the contents of the additional trusted cert
        #PEM Public Certificate data encoded as base64 string
        #such as `LS0tLS1C...LS0tCg==` where "..." is the
        #middle section of the long base64 string
        data: string
```

TKGS v1alpha2 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하기 위한 예제 YAML

TKGS API는 Tanzu Kubernetes 클러스터를 사용자 지정하기 지능형 기본값과 다양한 옵션을 제공합니다. 요구 사항을 충족하는 다양한 구성 및 사용자 지정으로 다양한 유형의 클러스터를 프로비저닝하려면 다음 예를 참조하십시오.

기본 Tanzu Kubernetes 클러스터 프로비저닝을 위한 예제 YAML

다음 예제 YAML은 TKGS Tanzu Kubernetes 클러스터 프로비저닝을 위한 TKGS v1alpha2 API를 사용하여 기본 Tanzu Kubernetes 클러스터를 프로비저닝합니다. 이 예제 YAML은 사용 가능한 모든 기본값을 사용하므로 클러스터를 프로비저닝하는 데 필요한 최소 구성을 나타냅니다.

```
apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TanzuKubernetesCluster
metadata:
  name: tkgs-v2-cluster-default
```

```
  namespace: tkgs-cluster-ns
spec:
  topology:
    controlPlane:
      replicas: 3
      vmClass: guaranteed-medium
      storageClass: vwt-storage-policy
      tkr:
        reference:
          name: v1.21.2---vmware.1-tkg.1.ee25d55
    nodePools:
    - name: worker-nodepool-a1
      replicas: 3
      vmClass: guaranteed-large
      storageClass: vwt-storage-policy
      tkr:
        reference:
          name: v1.21.2---vmware.1-tkg.1.ee25d55
```

사용자 지정 Tanzu Kubernetes 클러스터 프로비저닝을 위한 예제 YAML

다음 예제 YAML은 Tanzu Kubernetes Grid 서비스 Tanzu Kubernetes 클러스터 프로비저닝을 위한 TKGS v1alpha2 API를 사용하여 사용자 지정 Tanzu Kubernetes 클러스터를 프로비저닝합니다. 이 예제 YAML은 변동률이 높은 노드 구성 요소에 대해 별도의 볼륨을 지정하고 특정 스토리지 및 네트워크 설정을 사용자 지정합니다.

```
apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TanzuKubernetesCluster
metadata:
  name: tkgs-v2-cluster-custom
  namespace: tkgs-cluster-ns
spec:
  topology:
    controlPlane:
      replicas: 3
      vmClass: guaranteed-medium
      storageClass: vwt-storage-policy
      volumes:
        - name: etcd
          mountPath: /var/lib/etcd
          capacity:
            storage: 4Gi
      tkr:
        reference:
          name: v1.21.2---vmware.1-tkg.1.ee25d55
    nodePools:
    - name: worker-nodepool-a1
      replicas: 3
      vmClass: guaranteed-large
      storageClass: vwt-storage-policy
      volumes:
        - name: containerd
          mountPath: /var/lib/containerd
          capacity:
```

```
            storage: 16Gi
      tkr:
        reference:
          name: v1.21.2---vmware.1-tkg.1.ee25d55
    - name: worker-nodepool-a2
      replicas: 2
      vmClass: guaranteed-medium
      storageClass: vwt-storage-policy
      tkr:
        reference:
          name: v1.21.2---vmware.1-tkg.1.ee25d55
    - name: worker-nodepool-a3
      replicas: 1
      vmClass: guaranteed-small
      storageClass: vwt-storage-policy
      tkr:
        reference:
          name: v1.21.2---vmware.1-tkg.1.ee25d55
  settings:
    storage:
      defaultClass: vwt-storage-policy
    network:
      cni:
        name: antrea
      services:
        cidrBlocks: ["198.53.100.0/16"]
      pods:
        cidrBlocks: ["192.0.5.0/16"]
      serviceDomain: cluster.local
      proxy:
        httpProxy: http://<user>:<pwd>@<ip>:<port>
        httpsProxy: http://<user>:<pwd>@<ip>:<port>
        noProxy: [10.246.0.0/16,192.168.144.0/20,192.168.128.0/20]
      trust:
        additionalTrustedCAs:
          - name: CompanyInternalCA-1
            data: LS0tLS1C...LS0tCg==
          - name: CompanyInternalCA-2
            data: MTLtMT1C...MT0tPg==
```

클러스터 규격이 TKGS v1alpha2 API로 변환된 후 Tanzu Kubernetes 릴리스 업데이트

Tanzu Kubernetes 클러스터 규격이 v1alpha2 API 형식으로 자동 변환된 후 일반적으로 Tanzu Kubernetes 릴리스 버전을 변경하여 수행되는 Tanzu Kubernetes 클러스터의 롤링 업데이트를 수행하려면 오류를 방지하기 위해 클러스터 규격의 일부 사전 처리를 수행해야 할 수 있습니다.

클러스터 규격 자동 변환

vSphere with Tanzu 환경을 Tanzu Kubernetes Grid 서비스 v1alpha2 API로 업데이트하려면 서비스가 실행되는 감독자 클러스터를 업데이트합니다.

Tanzu Kubernetes Grid 서비스가 v1alpha2 API를 실행하면 시스템은 기존의 모든 Tanzu Kubernetes 클러스터 규격을 v1alpha1 형식에서 v1alpha2 형식으로 자동 변환합니다. 자동 변환 프로세스 중에 시스템은 각 클러스터 매니페스트에 대한 예상 필드를 생성하고 채웁니다. API 사용 중단 및 추가에는 v1alpha2 API에서 더 이상 사용되지 않는 클러스터 규격 필드와 신규 필드가 나열되어 있습니다.

매니페스트가 v1alpha2 형식으로 자동 변환된 클러스터의 Tanzu Kubernetes 릴리스 버전을 업데이트하려면 오류를 방지하기 위해 몇 가지 사전 처리를 수동으로 수행해야 합니다. 클러스터 업데이트 예에 다양한 옵션이 나열되어 있습니다.

API 사용 중단 및 추가

이 표에는 v1alpha2 API에서 더 이상 사용되지 않고 새 설정으로 대체된 클러스터 규격 설정이 나열되어 있습니다.

더 이상 사용되지 않는 설정	새 설정	주석
`spec.distribution.version` `spec.distribution.fullVersion`	`spec.topology.controlPlane.tkr.refernece.name` `spec.topology.nodePools[*].tkr.reference.name`	TKR NAME 형식을 사용해야 합니다. 예를 참조하십시오.
`spec.topology.workers`	`spec.topology.nodePools[*]`	변환된 클러스터에서 `spec.topology.workers` 블록은 `spec.topology.nodePools[0]`이 됩니다. `nodePools` 목록의 첫 번째 항목은 `name: workers`입니다.
`spec.topology.controlPlane.count` `spec.topology.workers.count`	`spec.topology.controlPlane.replicas` `spec.topology.nodePools[*].replicas`	`count`가 `replicas`로 대체됩니다.
`spec.topology.controlPlane.class` `spec.topology.workers.class`	`spec.topology.controlPlane.vmClass` `spec.topology.nodePools[*].vmClass`	`class`가 `vmClass`로 대체됩니다.
해당 없음	`spec.topology.nodePools[*].labels`	개체를 구성하고 분류하기 위한 선택적 키-쌍 값. 생성된 노드로 레이블이 전파됨
해당 없음	`spec.topology.nodePools[*].taints`	노드를 등록할 선택적 taint. 생성된 노드로 사용자 정의 taint가 전파됨

TKR NAME 형식이 필수임

더 이상 사용되지 않는 `spec.distribution.version` 필드 외에도 Tanzu Kubernetes 릴리스 버전을 지정하기 위한 DISTRIBUTION 형식은 지원되지 않습니다. 즉, `1.21.2+vmware.1-tkg.1.ee25d55`, `1.21.2`, `1.21` 문자열 형식을 사용하여 대상 릴리스를 참조할 수 없습니다.

v1alpha2 API 클러스터 규격에서 Tanzu Kubernetes 릴리스 버전을 참조하는 경우 더 이상 사용되지 않는 DISTRIBUTION 형식이 아닌 TKR NAME 형식을 사용해야 합니다. 더 이상 사용되지 않는 형식이 UPDATES AVAILABLE 열에 표시되지만 TKR NAME 열에 나열된 형식만 지원됩니다.

```
kubectl get tanzukubernetescluster
NAMESPACE        NAME            CONTROL PLANE   WORKER   TKR
NAME                             AGE    READY    TKR COMPATIBLE   UPDATES AVAILABLE
tkgs-cluster-1   test-cluster    3               3        v1.21.2---vmware.1-
tkg.1.ee25d55     38h    True    True             [1.21.2+vmware.1-tkg.1.ee25d55]
```

kubectl edit을 사용하여 클러스터 규격 업데이트

TKGS v1alpha2 API를 준수하도록 클러스터 규격을 편집해야 하는 경우 `kubectl edit` 메서드를 사용합니다. 이러한 유형의 업데이트에는 `kubectl patch` 메서드를 사용하지 마십시오. 클러스터 매니페스트 편집 방법의 내용을 참조하십시오. 편집기를 사용하여 `kubectl`을 구성하려면 kubectl용 기본 텍스트 편집기 지정 항목을 참조하십시오.

클러스터 업데이트 예

`spec.distribution.version`을 변경하는 것이 클러스터의 롤링 업데이트를 트리거하는 가장 일반적인 방법이며(Tanzu Kubernetes 클러스터 업데이트 참조) 이 필드는 v1alpha2 API에서 더 이상 사용되지 않기 때문에 잠재적인 클러스터 업데이트 문제를 방지하기 위해 알아야 할 몇 가지 고려 사항과 따라야 할 몇 가지 사전 처리 권장 사항이 있습니다.

다음 예는 v1alpha1 API를 사용하여 프로비저닝된 Tanzu Kubernetes 클러스터의 버전을 v1alpha2 API를 실행하는 시스템으로 업데이트하는 방법을 보여줍니다.

클러스터 업그레이드 예 1: 제어부에서 단일 TKR NAME 참조 사용

권장되는 방식은 변환된 규격에서 모든 `nodePools[*].tkr.reference.name` 블록을 제거하고 `controlPlane.tkr.reference.name`을 대상 릴리스의 TKR NAME으로 업데이트하는 것입니다. 이 경우 동일한 Tanzu Kubernetes 릴리스가 모든 `nodePools[*]` 노드에 전파됩니다.

앞으로 Tanzu Kubernetes 릴리스 버전은 `controlPlane` 및 `nodePools[*]`에서 다를 수 있습니다. 하지만 현재는 클러스터의 모든 릴리스가 일치해야 하므로 `controlPlane`에 단일 TKR NAME 참조를 배치하는 것으로 충분합니다.

예:

```
apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TanzuKubernetesCluster
metadata:
  name: tkgs-cluster-update-example1
  namespace: tkgs-cluster-ns
spec:
  settings:
    network:
      cni:
        name: antrea
      pods:
```

```
        cidrBlocks:
        - 192.0.2.0/16
      serviceDomain: cluster.local
      services:
        cidrBlocks:
        - 198.51.100.0/12
  topology:
    controlPlane:
      replicas: 3
      storageClass: vwt-storage-policy
      tkr:
        reference:
          name: v1.21.2---vmware.1-tkg.1.ee25d55
      vmClass: best-effort-medium
    nodePools:
    - name: workers
      replicas: 3
      storageClass: vwt-storage-policy
      vmClass: best-effort-medium
```

클러스터 업그레이드 예 2: 각 노드 풀에 대해 TKR NAME 참조 사용

두 번째 예는 TKR NAME을 `controlPlane` 및 `nodePools[*]` 토폴로지 모두에 대한 `tkr.reference.name` 블록에 배치하는 것입니다.

이 방식은 Tanzu Kubernetes 릴리스가 노드 풀 간에 다를 수 있는 향후 릴리스에 대비할 수 있다는 이점이 있습니다. 현재는 일치해야 합니다.

예:

```
apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TanzuKubernetesCluster
metadata:
  name: tkgs-cluster-update-example2
  namespace: tkgs-cluster-ns
spec:
  settings:
    network:
      cni:
        name: antrea
      pods:
        cidrBlocks:
        - 192.0.2.0/16
      serviceDomain: cluster.local
      services:
        cidrBlocks:
        - 198.51.100.0/12
  topology:
    controlPlane:
      replicas: 3
      storageClass: vwt-storage-policy
      vmClass: best-effort-medium
      tkr:
        reference:
```

```
          name: v1.21.2---vmware.1-tkg.1.ee25d55
    nodePools:
    - name: workers
      replicas: 3
      storageClass: vwt-storage-policy
      vmClass: best-effort-medium
      tkr:
        reference:
          name: v1.21.2---vmware.1-tkg.1.ee25d55
```

클러스터 업그레이드 예 3: 더 이상 사용되지 않는 Distribution 필드 사용

마지막 옵션은 더 이상 사용되지 않는 필드 `spec.distribution.fullVersion` 및 `spec.distribution.version`을 사용하고 모든 `tkr.reference.name` 블록을 수동으로 제거하는 것입니다. 하나는 TKR NAME 형식을 사용하고 다른 하나는 null로 표시된 두 필드를 모두 포함해야 합니다. `v1.21.2` 및 `v1.21`과 같은 버전 바로 가기는 지원되지 않습니다.

참고 Ubuntu의 Tanzu Kubernetes 릴리스에서는 `spec.distribution.version` 사용이 지원되지 않습니다.

다음 예에서는 `fullVersion`에 TKR NAME이 있고 `version` 필드에 null(빈) 값이 있습니다. 모든 `tkr.reference.name` 항목은 제거됩니다.

```
apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TanzuKubernetesCluster
metadata:
  name: tkgs-cluster-update-example3a
  namespace: tkgs-cluster-ns
spec:
  distribution:
    fullVersion: v1.21.2---vmware.1-tkg.1.ee25d55
    version: ""
  settings:
    network:
      cni:
        name: antrea
      pods:
        cidrBlocks:
        - 192.0.2.0/16
      serviceDomain: cluster.local
      services:
        cidrBlocks:
        - 198.51.100.0/12
  topology:
    controlPlane:
      replicas: 3
      storageClass: vwt-storage-policy
      vmClass: best-effort-medium
    nodePools:
```

```
    - name: workers
      replicas: 3
      storageClass: vwt-storage-policy
      vmClass: best-effort-medium
```

또는 `version` 필드에 TKR NAME을 포함하고 `fullVersion` 필드에 null(빈) 값을 사용할 수 있습니다. `version` 필드를 사용하더라도 버전 바로 가기는 지원되지 않습니다. 모든 `tkr.reference.name` 항목은 제거됩니다.

```
apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TanzuKubernetesCluster
metadata:
  name: tkgs-cluster-update-example3b
  namespace: tkgs-cluster-ns
spec:
  distribution:
    fullVersion: ""
    version: v1.21.2---vmware.1-tkg.1.ee25d55
  settings:
    network:
      cni:
        name: antrea
      pods:
        cidrBlocks:
        - 192.0.2.0/16
      serviceDomain: cluster.local
      services:
        cidrBlocks:
        - 198.51.100.0/12
  topology:
    controlPlane:
      replicas: 3
      storageClass: vwt-storage-policy
      vmClass: best-effort-medium
    nodePools:
    - name: workers
      replicas: 3
      storageClass: vwt-storage-policy
      vmClass: best-effort-medium
```

v1alpha2 API를 사용하여 라우팅 가능한 포드 네트워크로 Tanzu Kubernetes 클러스터 구성

`antrea-nsx-routed`를 클러스터의 CNI로 지정하여 라우팅 가능한 포드 네트워킹을 사용하도록 Tanzu Kubernetes 클러스터를 구성할 수 있습니다.

라우팅 가능한 포드 네트워킹 소개

Kubernetes 네트워크 모델을 사용하려면 클러스터의 노드 네트워크에 있는 포드가 NAT(네트워크 주소 변환) 없이 동일한 클러스터의 모든 노드에 있는 모든 포드와 통신할 수 있어야 합니다. 이 요구 사항을 충족하기 위해 각 Kubernetes 포드에는 전용 포드 네트워크에서 할당된 IP 주소가 지정됩니다.

antrea 또는 calico CNI 플러그인을 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하면 기본 포드 네트워크 192.168.0.0/16이 생성됩니다. 이 서브넷은 클러스터 내에서만 고유하며 인터넷에서 라우팅할 수 없는 개인 주소 공간입니다. network.pods.cidrBlocks를 사용자 지정할 수 있지만 이러한 CNI 플러그인을 사용하여 포드 네트워크를 라우팅할 수 없습니다. 자세한 내용은 Tanzu Kubernetes 클러스터 프로비저닝을 위한 TKGS v1alpha2 API의 내용을 참조하십시오.

Tanzu Kubernetes Grid 서비스 v1alpha2 API는 antrea-nsx-routed CNI 플러그인을 사용하여 라우팅 가능한 포드 네트워킹을 지원합니다. 이 네트워크 인터페이스는 Tanzu Kubernetes 클러스터에 대해 라우팅 가능한 포드 네트워크를 지원하도록 구성된 사용자 지정된 Antrea 플러그인입니다. 클러스터 규격에서 포드 CIDR 블록 필드는 명시적으로 null이어야 합니다. 그래야 IPAM(IP 주소 관리)이 감독자 클러스터에서 처리됩니다.

라우팅 가능한 포드 네트워킹을 사용하도록 설정하면 클러스터 외부의 클라이언트에서 직접 포드의 주소를 지정할 수 있습니다. 또한 외부 네트워크 서비스 및 서버가 소스 포드를 식별하고 IP 주소를 기반으로 정책을 적용할 수 있도록 포드 IP 주소가 보존됩니다. 지원되는 트래픽 패턴에는 다음이 포함됩니다.

- Tanzu Kubernetes 클러스터 포드와 동일한 vSphere 네임스페이스에 있는 vSphere 포드 간에 트래픽이 허용됩니다.
- Tanzu Kubernetes 클러스터 포드와 다른 vSphere 네임스페이스에 있는 vSphere 포드 간에 트래픽이 삭제됩니다.
- 감독자 클러스터 제어부 노드는 Tanzu Kubernetes 클러스터 포드에 연결할 수 있습니다.
- Tanzu Kubernetes 클러스터 포드는 외부 네트워크에 연결할 수 있습니다.
- 외부 네트워크는 Tanzu Kubernetes 클러스터 포드에 연결할 수 없습니다. Tanzu Kubernetes 클러스터 노드의 DFW(분산 방화벽) 격리 규칙에 따라 트래픽이 삭제됩니다.

라우팅 가능한 포드에 대한 시스템 요구 사항

라우팅 가능 포드 네트워킹을 사용하려면 NSX-T Data Center로 감독자 클러스터를 구성해야 합니다. 네이티브 vSphere vDS 네트워킹에는 라우팅 가능한 포드를 사용할 수 없습니다.

라우팅 가능한 포드에는 Tanzu Kubernetes Grid 서비스 v1alpha2 API가 필요합니다. TKGS v1alpha2 API 사용에 대한 요구 사항의 내용을 참조하십시오.

라우팅 가능한 포드에 대한 NSX-T 구성 요구 사항

기본 요구 사항 외에, Tanzu Kubernetes 클러스터에서 라우팅 가능한 포드 네트워크를 사용하는 데 필요한 특별한 NSX-T 구성은 없습니다. NSX-T가 포함된 vSphere U3을 실행하는 vSphere with Tanzu 환경에는 라우팅 가능한 포드 네트워크를 지원하는 NCP 버전이 포함됩니다. 추가적인 NSX-T 구성은 필요하지 않습니다.

NCP는 다음 두 가지 소스 중 하나에서 라우팅 가능한 포드 네트워크에 대한 IP 풀을 생성합니다.

- 워크로드 네트워크가 네임스페이스 네트워크로 구성된 경우 NCP는 이 네임스페이스 네트워크에 대해 지정된 IP 블록에서 하나 이상의 IP 풀을 생성합니다.

- 워크로드 네트워크에 대해 지정된 네임스페이스 네트워크가 없는 경우 NCP는 감독자 클러스터 포드 CIDR에서 하나 이상의 IP 풀을 생성합니다.

자세한 내용은 VDS 네트워킹으로 구성된 감독자 클러스터에 워크로드 네트워크 추가 및 NSX-T Data Center로 구성된 감독자 클러스터에서 워크로드 네트워크 설정 변경 항목을 참조하십시오.

라우팅 가능한 포드에 대한 감독자 클러스터 구성 요구 사항

기본 요구 사항 외에 Tanzu Kubernetes 클러스터에서 라우팅 가능한 포드 네트워크를 사용하는 데 필요한 특별한 감독자 클러스터 구성은 없습니다.

라우팅 가능한 포드 네트워킹을 아래 설명된 대로 사용하도록 설정한 경우 Tanzu Kubernetes 클러스터 포드 CIDR은 네임스페이스 네트워크에서 생성된 IP 풀에서 할당되거나, 없는 경우 감독자 클러스터 포드 CIDR에서 할당됩니다.

클러스터 노드에 대한 IP 주소를 할당하는 감독자 클러스터 서비스 CIDR이 네임스페이스 네트워크 CIDR 또는 감독자 클러스터 포드 CIDR과 겹치지 않는지 확인해야 합니다.

라우팅 가능한 포드에 대한 클러스터 구성 예

다음 예제 YAML은 라우팅 가능한 포드 네트워크로 클러스터를 구성하는 방법을 보여줍니다. v1alpha2 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하고 Tanzu Kubernetes Grid 서비스를 호출하기 위한 사용자 지정 구성입니다.

클러스터 규격은 `antrea-nsx-routed`를 라우팅 가능한 포드 네트워킹을 사용하도록 설정하기 위한 CNI로 선언합니다. CNI로 `antrea-nsx-routed`가 지정되면 `pods.cidrBlock` 필드는 비어 있어야 합니다. `antrea-nsx-routed`가 지정된 경우 NSX-T 네트워킹이 사용되지 않으면 클러스터 프로비저닝이 실패합니다.

```
apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TanzuKubernetesCluster
metadata:
  name: tkgs-v2-cluster-routable-pods
  namespace: tkgs-cluster-ns
spec:
  topology:
    controlPlane:
      replicas: 3
      vmClass: guaranteed-medium
      storageClass: vwt-storage-policy
      tkr:
        reference:
          name: v1.21.2---vmware.1-tkg.1.ee25d55
    nodePools:
    - name: worker-nodepool-a1
      replicas: 3
      vmClass: guaranteed-large
      storageClass: vwt-storage-policy
      tkr:
        reference:
          name: v1.21.2---vmware.1-tkg.1.ee25d55
  settings:
```

```
    storage:
      defaultClass: vwt-storage-policy
    network:
      #`antrea-nsx-routed` is the required CNI
      #for routable pods
      cni:
        name: antrea-nsx-routed
      services:
        cidrBlocks: ["10.97.0.0/24"]
      serviceDomain: tanzukubernetescluster.local
      #`pods.cidrBlocks` value must be empty
      #when `antrea-nsx-routed` is the CNI
      pods:
        cidrBlocks:
```

TKGS v1alpha2 API에 대한 구성 매개 변수

CNI(Container Network Interface), 프록시 서버 및 TLS 인증서를 비롯한 주요 기능에 대한 글로벌 설정을 사용하여 Tanzu Kubernetes Grid 서비스를 사용자 지정할 수 있습니다. 글로벌 기능을 구현하는 것과 클러스터당 기능을 구현하는 것의 장단점 및 고려 사항을 알고 있어야 합니다.

TkgServiceConfiguration v1alpha2 규격

`TkgServiceConfiguration` 규격은 Tanzu Kubernetes Grid 서비스 인스턴스 구성을 위한 필드를 제공합니다.

```
apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TkgServiceConfiguration
metadata:
  name: tkg-service-configuration-spec
spec:
  defaultCNI: string
  proxy:
    httpProxy: string
    httpsProxy: string
    noProxy: [string]
  trust:
    additionalTrustedCAs:
      - name: string
        data: string
  defaultNodeDrainTimeout: time
```

경고 Tanzu Kubernetes Grid 서비스 구성은 글로벌 작업입니다. `TkgServiceConfiguration` 규격을 변경하면 해당 서비스에서 프로비저닝된 모든 Tanzu Kubernetes 클러스터에 적용됩니다. 롤링 업데이트가 시작되면(수동으로 또는 업그레이드를 통해) 변경된 서비스 규격에 따라 클러스터가 업데이트됩니다.

주석이 추가된 TkgServiceConfiguration v1alpha2 규격

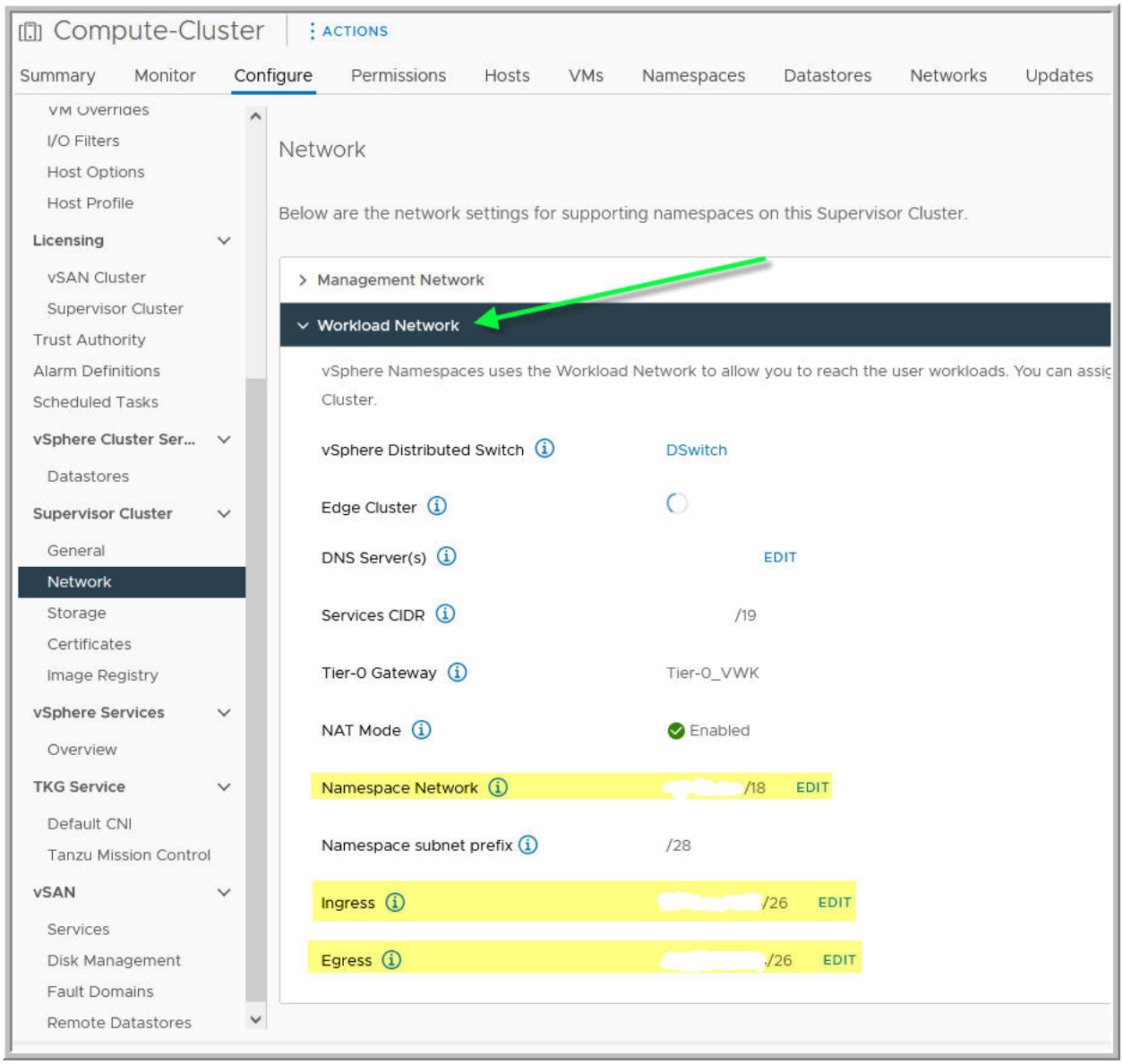
다음 YAML은 각 `TkgServiceConfiguration` 규격 매개 변수에 대해 구성 가능한 필드를 나열하고 설명합니다. 예제에 대해서는 Tanzu Kubernetes Grid 서비스 v1alpha1 API 구성 예시 항목을 참조하십시오.

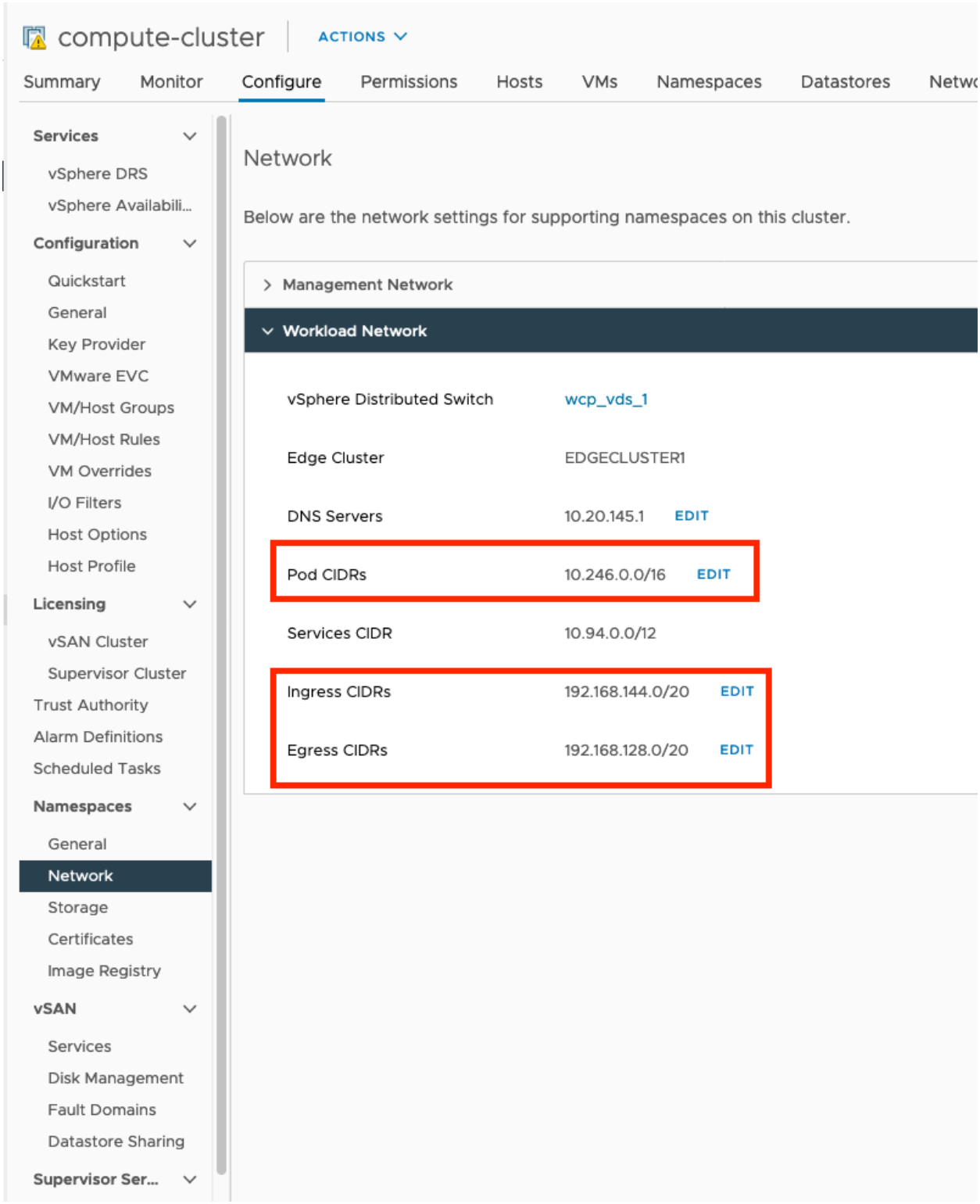
```
apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TkgServiceConfiguration
metadata:
  name: tkg-service-configuration-spec
spec:
  #defaultCNI is the default CNI for all Tanzu Kubernetes
  #clusters to use unless overridden on a per-cluster basis
  #supported values are antrea, calico, antrea-nsx-routed
  #defaults to antrea
  defaultCNI: string
  #proxy configures a proxy server to be used inside all
  #clusters provisioned by this TKGS instance
  #if implemented all fields are required
  #if omitted no proxy is configured
  proxy:
    #httpProxy is the proxy URI for HTTP connections
    #to endpionts outside the clusters
    #takes the form http://<user>:<pwd>@<ip>:<port>
    httpProxy: string
    #httpsProxy is the proxy URI for HTTPS connections
    #to endpoints outside the clusters
    #takes the frorm http://<user>:<pwd>@<ip>:<port>
    httpsProxy: string
    #noProxy is the list of destination domain names, domains,
    #IP addresses, and other network CIDRs to exclude from proxying
    #must include Supervisor Cluster Pod, Egress, Ingress CIDRs
    noProxy: [string]
  #trust configures additional trusted certificates
  #for the clusters provisioned by this TKGS instance
  #if omitted no additional certificate is configured
  trust:
    #additionalTrustedCAs are additional trusted certificates
    #can be additional CAs or end certificates
    additionalTrustedCAs:
      #name is the name of the additional trusted certificate
      #must match the name used in the filename
      - name: string
        #data holds the contents of the additional trusted cert
        #PEM Public Certificate data encoded as a base64 string
        data: string
  #defaultNodeDrainTimeout is the total amount of time the
  #controller spends draining a node; default is undefined
  #which is the value of 0, meaning the node is drained
  #without any time limitations; note that `nodeDrainTimeout`
  #is different from `kubectl drain --timeout`
  defaultNodeDrainTimeout: time
```

프록시 서버 구성 요구 사항

구성된 경우 프록시 서버는 클러스터에서 아웃바운드 HTTP 및 HTTPS 트래픽에 사용됩니다. Tanzu Kubernetes 클러스터에 대한 프록시 서버를 구성하려면 다음 요구 사항에 유의하십시오.

- 필수 `proxy` 매개 변수는 `httpProxy`, `httpsProxy`, `noProxy`입니다. `proxy` stanza를 추가하는 경우 세 개의 필드가 모두 필수입니다.
- HTTP를 사용하여 프록시 서버에 연결할 수 있습니다. HTTPS 연결은 지원되지 않습니다.
- 필요한 `spec.proxy.noProxy` 값은 **워크로드 네트워크**에서 가져옵니다. **네임스페이스 네트워크**(이전의 포드 CIDR), **수신**(이전의 수신 CIDR) 및 **송신**(이전의 송신 CIDR)를 `noProxy` 필드에 포함하여 프록시로 지정하면 안 됩니다. 아래의 예제 이미지를 참조하십시오.
- 서비스 CIDR을 `noProxy` 필드에 포함할 필요가 없습니다. Tanzu Kubernetes 클러스터는 이 서브넷과 상호 작용하지 않습니다.
- Tanzu Kubernetes 클러스터 규격의 `network.services.cidrBlocks` 및 `network.pods.cidrBlocks` 값을 `noProxy` 필드에 포함할 필요가 없습니다. 이러한 서브넷은 자동으로 프록시로 지정되지 않습니다.
- `localhost` 및 `127.0.0.1`은 `noProxy` 필드에 포함할 필요가 없습니다. 끝점은 자동으로 프록시로 지정되지 않습니다.


Compute-Cluster
ACTIONS
Summary
Monitor
Configure
Permissions
Hosts
VMs
Namespaces
Datastores
Networks
Updates
VM Overrides
I/O Filters
Host Options
Host Profile
Licensing
vSAN Cluster
Supervisor Cluster
Trust Authority
Alarm Definitions
Scheduled Tasks
vSphere Cluster Ser...
Datastores
Supervisor Cluster
General
Network
Storage
Certificates
Image Registry
vSphere Services
Overview
TKG Service
Default CNI
Tanzu Mission Control
vSAN
Services
Disk Management
Fault Domains
Remote Datastores
Network
Below are the network settings for supporting namespaces on this Supervisor Cluster.
Management Network
Workload Network
vSphere Namespaces uses the Workload Network to allow you to reach the user workloads. You can assig
Cluster.
vSphere Distributed Switch
DSwitch
Edge Cluster
DNS Server(s)
EDIT
Services CIDR
/19
Tier-0 Gateway
Tier-0_VWK
NAT Mode
Enabled
Namespace Network
/18
EDIT
Namespace subnet prefix
/28
Ingress
/26
EDIT
Egress
/26
EDIT



글로벌 또는 클러스터별 구성 옵션을 사용하는 경우

`TkgServiceConfiguration`은 Tanzu Kubernetes Grid 서비스 인스턴스로 프로비저닝된 모든 Tanzu Kubernetes 클러스터에 영향을 미치는 글로벌 규격입니다.

TkgServiceConfiguration 규격을 편집하기 전에, 글로벌 구성 대신 사용 사례를 충족할 수 있는 클러스터별 대안을 알고 있어야 합니다.

표 13-2. 글로벌 및 클러스터별 구성 옵션

설정	글로벌 옵션	클러스터별 옵션
기본 CNI	TkgServiceConfiguration 규격을 편집합니다. Tanzu Kubernetes Grid 서비스 v1alpha1 API 구성 예시의 내용을 참조하십시오.	클러스터 규격에서 CNI를 지정합니다. 예를 들어 기본 CNI는 Antrea입니다. Calico를 사용하려면 클러스터 YAML에서 지정합니다. Tanzu Kubernetes Grid 서비스 v1alpha1 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하는 예 항목을 참조하십시오.
프록시 서버	TkgServiceConfiguration 규격을 편집합니다. Tanzu Kubernetes Grid 서비스 v1alpha1 API 구성 예시의 내용을 참조하십시오.	클러스터 규격에 프록시 서버 구성 매개변수를 포함합니다. Tanzu Kubernetes Grid 서비스 v1alpha1 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하는 예의 내용을 참조하십시오.
트러스트 인증서	TkgServiceConfiguration 규격을 편집합니다. 외부 컨테이너 레지스트리 구성과 인증서 기반 프록시 구성이라는 두 가지 사용 사례가 있습니다. Tanzu Kubernetes Grid 서비스 v1alpha1 API 구성 예시 항목을 참조하십시오.	예, 클러스터별로 사용자 지정 인증서를 포함하거나 클러스터 규격에서 전체적으로 설정된 trust 설정을 재정의할 수 있습니다. Tanzu Kubernetes Grid 서비스 v1alpha1 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하는 예의 내용을 참조하십시오.

참고 TkgServiceConfiguration에 글로벌 프록시가 구성된 경우 이 프록시 정보는 클러스터의 초기 배포 후 클러스터 매니페스트로 전파됩니다. 글로벌 프록시 구성은 클러스터를 생성할 때 프록시 구성 필드가 없는 경우에만 클러스터 매니페스트에 추가됩니다. 다시 말해, 클러스터별 구성이 우선하며 글로벌 프록시 구성을 덮어씁니다. 자세한 내용은 Tanzu Kubernetes Grid 서비스 v1alpha1 API에 대한 구성 매개변수의 내용을 참조하십시오.

TkgServiceConfiguration 규격을 편집하기 전에 글로벌 수준에서 설정을 적용하는 결과에 대해 알고 있어야 합니다.

필드	적용	추가/변경된 경우 기존 클러스터에 미치는 영향	클러스터 생성 시 클러스터별 재정의	클러스터 업데이트 시 클러스터별 재정의
defaultCNI	전체적으로	없음	예, 클러스터 생성 시 글로벌 설정을 재정의할 수 있습니다.	아니요, 기존 클러스터의 CNI를 변경할 수 없습니다. 클러스터 생성 시 전체적으로 설정된 기본 CNI를 사용한 경우에는 변경할 수 없습니다.
proxy	전체적으로	없음	예, 클러스터 생성 시 글로벌 설정을 재정의할 수 있습니다.	예, U2+를 사용하면 클러스터 업데이트 시 글로벌 설정을 재정의할 수 있습니다.
trust	전체적으로	없음	예, 클러스터 생성 시 글로벌 설정을 재정의할 수 있습니다.	예, U2+를 사용하면 클러스터 업데이트 시 글로벌 설정을 재정의할 수 있습니다.

글로벌 구성 변경을 기존 클러스터에 전파

TkgServiceConfiguration의 글로벌 수준에서 지정한 설정은 기존 클러스터에 자동으로 전파되지 않습니다. 예를 들어 TkgServiceConfiguration에서 proxy 또는 trust 설정을 변경하는 경우 이미 프로비저닝된 클러스터에는 이러한 변경 사항이 영향을 주지 않습니다.

기존 클러스터에 글로벌 변경 내용을 전파하려면 Tanzu Kubernetes 클러스터에 패치를 적용하여 TkgServiceConfiguration에 대한 변경 내용을 클러스터가 상속하도록 해야 합니다.

예:

```
kubectl patch tkc <CLUSTER_NAME> -n <NAMESPACE> --type merge -p "{\"spec\":{\"settings\":
{\"network\":{\"proxy\": null}}}}"
```

```
kubectl patch tkc <CLUSTER_NAME> -n <NAMESPACE> --type merge -p "{\"spec\":{\"settings\":
{\"network\":{\"trust\": null}}}}"
```

v1alpha2 API를 사용하여 TKGS 인스턴스를 구성하는 예시

컨테이너 네트워크 인터페이스, 프록시 서버 및 TLS 인증서에 대한 글로벌 구성 설정을 사용하여 Tanzu Kubernetes Grid 서비스 v1alpha2 API를 사용자 지정하려면 예시를 참조하십시오.

v1alpha2 API를 사용하여 Tanzu Kubernetes Grid 서비스 구성

기본 CNI를 변경하고 글로벌 프록시 서버를 추가하고 신뢰할 수 있는 인증서를 추가하여 Tanzu Kubernetes Grid 서비스를 사용자 지정할 수 있습니다. Tanzu Kubernetes Grid 서비스 v1alpha1 API에 대한 구성 매개 변수의 내용을 참조하십시오.

```
apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TkgServiceConfiguration
metadata:
  name: tkg-service-v2-configuration-example
spec:
  defaultCNI: antrea
  proxy:
    #supported format is `http://<user>:<pwd>@<ip>:<port>`
    httpProxy: http://admin:PaSsWoRd@10.66.100.22:80
    httpsProxy: http://admin:PaSsWoRd@10.66.100.22:80
    noProxy: [10.246.0.0/16,192.168.144.0/20,192.168.128.0/20]
  trust:
    additionalTrustedCAs:
      - name: CompanyInternalCA-1
        data: LS0tLS1C...LS0tCg==
        #where "..." is the middle section of the long base64 string
      - name: CompanyInternalCA-2
        data: MTLtMT1C...MT0tPg==
  defaultNodeDrainTimeout: 0
```

경고 Tanzu Kubernetes Grid 서비스 규격을 편집하면 해당 서비스에서 프로비저닝된 모든 클러스터가 전체적으로 변경되며, 여기에는 수동으로 또는 자동으로 업그레이드되는 기존 클러스터 및 새 클러스터가 포함됩니다.

사전 요구 사항: Kubectl 편집 구성

Tanzu Kubernetes 클러스터 크기를 조정하려면 `kubectl edit tanzukubernetescluster/CLUSTER-NAME` 명령을 사용하여 클러스터 매니페스트를 업데이트합니다. kubectl edit 명령은 KUBE_EDITOR 또는 EDITOR 환경 변수로 정의된 텍스트 편집기에서 클러스터 매니페스트를 엽니다. 환경 변수 설정에 대한 지침은 kubectl용 기본 텍스트 편집기 지정 항목을 참조하십시오.

규격 변경 내용을 저장하면 `kubectl`은 편집 내용이 기록되었다고 보고합니다. 취소하려면 저장하지 않고 편집기를 닫기만 하면 됩니다.

기본 CNI 구성

Tanzu Kubernetes Grid 서비스는 Tanzu Kubernetes 클러스터에 대한 기본 CNI(Container Network Interface)를 제공합니다. 기본 구성을 사용하면 CNI를 지정하지 않고도 클러스터를 생성할 수 있습니다. 서비스 규격을 편집하여 기본 CNI를 변경할 수 있습니다.

Tanzu Kubernetes Grid 서비스는 두 가지(Antrea 및 Calico) CNI를 지원하며 Antrea가 기본값입니다. 자세한 내용은 Tanzu Kubernetes Grid 서비스 클러스터 네트워킹 의 내용을 참조하십시오.

사용할 CNI를 명시적으로 지정하여 기본 CNI를 재정의할 수 있습니다. 또는 CNI용 TKG 서비스 컨트롤러를 편집하여 기본 CNI를 변경할 수 있습니다.

1 감독자 클러스터로 인증합니다.

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

2 컨텍스트를 대상 vSphere 네임스페이스로 전환합니다.

```
kubectl config use-context tkgs-cluster-ns
```

3 기본 CNI를 나열합니다.

```
kubectl get tkgserviceconfigurations
```

예제 결과:

```
NAME                                DEFAULT CNI
tkg-service-configuration           antrea
```

4 Tanzu Kubernetes Grid 서비스 규격을 편집하기 위해 로드합니다.

```
kubectl edit tkgserviceconfigurations tkg-service-configuration
```

KUBE_EDITOR 또는 EDITOR 환경 변수로 정의된 기본 텍스트 편집기에서 `tkg-service-configuration` 규격이 열립니다.

5 `spec.defaultCNI` 값을 편집합니다.

예를 들어, 변경 전:

```
spec:
  defaultCNI: antrea
```

변경 후:

```
spec:
  defaultCNI: calico
```

6 변경 내용을 적용하려면 텍스트 편집기에서 파일을 저장합니다. 취소하려면 저장하지 않고 편집기를 닫습니다.

텍스트 편집기에서 변경 내용을 저장하면 kubectl은 `tkg-service-configuration` 서비스 규격을 업데이트합니다.

7 기본 CNI가 업데이트되었는지 확인합니다.

```
kubectl get tkgserviceconfigurations
```

기본 CNI가 업데이트됩니다. 기본 네트워크 설정으로 프로비저닝된 모든 클러스터는 기본 CNI를 사용합니다.

```
NAME                                DEFAULT CNI
tkg-service-configuration           calico
```

글로벌 프록시 서버 구성

글로벌 프록시 서버를 사용하도록 설정하려면 프록시 서버 매개 변수를 TkgServiceConfiguration에 추가합니다. 필수 필드에 대한 설명은 Tanzu Kubernetes Grid 서비스 v1alpha1 API에 대한 구성 매개 변수 항목을 참조하십시오.

1 감독자 클러스터로 인증합니다.

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

2 컨텍스트를 대상 vSphere 네임스페이스로 전환합니다.

```
kubectl config use-context tkgs-cluster-ns
```

3 현재 구성을 가져옵니다.

```
kubectl get tkgserviceconfigurations
```

예제 결과:

```
NAME                         DEFAULT CNI
tkg-service-configuration    antrea
```

4 Tanzu Kubernetes Grid 서비스 규격을 편집하기 위해 로드합니다.

```
kubectl edit tkgserviceconfigurations tkg-service-configuration
```

KUBE_EDITOR 또는 EDITOR 환경 변수로 정의된 기본 텍스트 편집기에서 tkg-service-configuration 규격이 열립니다.

5 httpProxy, httpsProxy, noProxy를 포함하여 각 필수 필드에 spec.proxy 하위 섹션을 추가합니다.

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TkgServiceConfiguration
metadata:
  ...
  name: tkg-service-configuration-example
  resourceVersion: "44170525"
  selfLink: /apis/run.tanzu.vmware.com/v1alpha1/tkgserviceconfigurations/tkg-service-
configuration
  uid: 10347195-5f0f-490e-8ae1-a758a724c0bc
spec:
  defaultCNI: antrea
```

```
  proxy:
    httpProxy: http://<user>:<pwd>@<ip>:<port>
    httpsProxy: http://<user>:<pwd>@<ip>:<port>
    noProxy: [SVC-POD-CIDRs, SVC-EGRESS-CIDRs, SVC-INGRESS-CIDRs]
```

6 각 프록시 필드를 적절한 값으로 채웁니다. 각 필드에 대한 설명은 Tanzu Kubernetes Grid 서비스 v1alpha1 API에 대한 구성 매개 변수 항목을 참조하십시오.

noProxy 필드에 필요한 값은 감독자 클러스터의 **워크로드 네트워크**에서 가져옵니다. 이러한 값을 얻을 수 있는 위치는 위 항목의 그림을 참조하십시오.

예:

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TkgServiceConfiguration
metadata:
  ...
  name: tkg-service-configuration-example
  resourceVersion: "44170525"
  selfLink: /apis/run.tanzu.vmware.com/v1alpha1/tkgserviceconfigurations/tkg-service-
configuration
  uid: 10347195-5f0f-490e-8ae1-a758a724c0bc
spec:
  defaultCNI: antrea
  proxy:
    httpProxy: http://user:password@10.186.102.224:3128
    httpsProxy: http://user:password@10.186.102.224:3128
    noProxy: [10.246.0.0/16,192.168.144.0/20,192.168.128.0/20]
```

7 변경 내용을 적용하려면 텍스트 편집기에서 파일을 저장합니다. 취소하려면 저장하지 않고 편집기를 닫습니다.

텍스트 편집기에서 변경 내용을 저장하면 kubectl은 tkg-service-configuration 서비스 규격에 정의된 구성으로 Tanzu Kubernetes Grid 서비스를 업데이트합니다.

8 Tanzu Kubernetes Grid 서비스가 프록시 설정으로 업데이트되었는지 확인합니다.

```
kubectl get tkgserviceconfigurations -o yaml
```

9 확인하려면 Tanzu Kubernetes 클러스터를 프로비저닝합니다. TKGS v1alpha2 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하는 워크플로의 내용을 참조하십시오.

다음 명령을 사용하여 클러스터가 프록시를 사용하고 있는지 확인합니다.

```
kubectl get tkc CLUSTER-NAME -n NAMESPACE -o yaml
```

인증서 기반 프록시 구성

프록시 서버를 사용하여 인터넷 트래픽을 라우팅하는 것이 일부 환경에서는 엄격한 요구 사항입니다. 예를 들어, 금융 기관과 같이 규제가 엄격한 업계의 회사에서는 모든 인터넷 트래픽이 회사 프록시를 통과해야 합니다.

아웃바운드 HTTP/S 트래픽에 프록시 서버를 사용하는 Tanzu Kubernetes 클러스터를 프로비저닝하도록 Tanzu Kubernetes Grid 서비스를 구성할 수 있습니다. 자세한 내용은 Tanzu Kubernetes Grid 서비스 v1alpha1 API에 대한 구성 매개 변수의 내용을 참조하십시오.

예에서 볼 수 있듯이, 프록시 서버에 대한 신뢰할 수 있는 인증서를 `TkgServiceConfiguration` 규격에 추가할 수 있습니다.

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TkgServiceConfiguration
metadata:
  name: tkg-service-configuration-example
spec:
  defaultCNI: antrea
  proxy:
    httpProxy: http://user:password@10.186.102.224:3128
    httpsProxy: http://user:password@10.186.102.224:3128
    noProxy: [10.246.0.0/16,192.168.144.0/20,192.168.128.0/20]
  trust:
    additionalTrustedCAs:
      - name: first-cert-name
        data: base64-encoded string of a PEM encoded public cert 1
      - name: second-cert-name
        data: base64-encoded string of a PEM encoded public cert 2
  defaultNodeDrainTimeout: 0
```

외부 개인 레지스트리 구성

Tanzu Kubernetes 클러스터를 외부 개인 레지스트리와 연결하기 위한 사용자 지정 인증서를 사용하여 Tanzu Kubernetes Grid 서비스를 구성할 수 있습니다. 자세한 내용은 Tanzu Kubernetes 클러스터에서 외부 컨테이너 레지스트리 사용의 내용을 참조하십시오.

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TkgServiceConfiguration
metadata:
  name: tkg-service-configuration-example
spec:
  defaultCNI: antrea
  trust:
    additionalTrustedCAs:
      - name: harbor-vm-cert
        data: <<<base64-encoded string of a PEM encoded public cert>>>>
```

TKGS v1alpha2 API를 사용하여 Tanzu Kubernetes 클러스터 확장/축소

노드 수를 변경하여 수평으로 또는 노드를 호스팅하는 가상 시스템 클래스를 변경하여 수직으로 Tanzu Kubernetes 클러스터를 확장/축소할 수 있습니다.

지원되는 확장/축소 작업

다음 표에는 Tanzu Kubernetes 클러스터에 대해 지원되는 확장/축소 작업이 나열되어 있습니다.

표 13-3. Tanzu Kubernetes 클러스터에 대해 지원되는 확장/축소 작업

노드	수평 확장	수평 축소	수직 확장/축소	볼륨 확장/축소
제어부	예	아니요	예	아니요
작업자	예	예	예	예

다음 고려 사항에 유의하십시오.

- 클러스터 노드를 수직으로 확장/축소하는 동안은 사용 가능한 리소스가 부족하여 노드에서 워크로드가 더 이상 실행되지 못할 수 있습니다. 이러한 이유로 인해 수평으로 확장/축소하는 방식을 선호하기도 합니다.
- VM 클래스는 변경할 수 없습니다. 클러스터에서 사용되는 VM 클래스를 편집한 후 Tanzu Kubernetes 클러스터를 확장하면, 새 클러스터 노드는 업데이트된 클래스 정의를 사용하지만 기존 클러스터 노드는 초기 클래스 정의를 계속 사용하여 불일치가 발생합니다. Tanzu Kubernetes 클러스터에 대한 가상 시스템 클래스의 내용을 참조하십시오.
- 작업자 노드 볼륨은 프로비저닝 후에 변경할 수 있습니다. 제어부 노드 볼륨은 할 수 없습니다.

확장/축소 사전 요구 사항: Kubectl 편집 구성

Tanzu Kubernetes 클러스터 크기를 조정하려면 `kubectl edit tanzukubernetescluster/CLUSTER-NAME` 명령을 사용하여 클러스터 매니페스트를 업데이트합니다. kubectl edit 명령은 KUBE_EDITOR 또는 EDITOR 환경 변수로 정의된 텍스트 편집기에서 클러스터 매니페스트를 엽니다. 환경 변수 설정에 대한 지침은 kubectl용 기본 텍스트 편집기 지정 항목을 참조하십시오.

매니페스트 변경 내용을 저장하면 `kubectl`은 편집 내용이 기록되었다고 보고하고 이 변경 내용으로 클러스터가 업데이트됩니다.

```
kubectl edit tanzukubernetescluster/tkgs-cluster-1
tanzukubernetescluster.run.tanzu.vmware.com/tkgs-cluster-1 edited
```

취소하려면 저장하지 않고 편집기를 닫기만 하면 됩니다.

```
kubectl edit tanzukubernetescluster/tkgs-cluster-1
Edit cancelled, no changes made.
```

제어부 확장

제어부 노드 수를 1에서 3으로 늘려서 Tanzu Kubernetes 클러스터를 확장할 수 있습니다. 제어부 노드의 수는 홀수여야 합니다. 제어부는 축소할 수 없습니다.

1 감독자 클러스터로 인증합니다.

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

2 Tanzu Kubernetes 클러스터를 실행 중인 vSphere 네임스페이스로 컨텍스트를 전환합니다.

```
kubectl config use-context tkgs-cluster-ns
```

3 네임스페이스에서 실행 중인 Kubernetes 클러스터를 나열합니다.

```
kubectl get tanzukubernetescluster -n tkgs-cluster-ns
```

4 대상 클러스터에서 실행 중인 노드 수를 구합니다.

```
kubectl get tanzukubernetescluster tkgs-cluster-1
```

예를 들어 다음 클러스터에는 제어부 노드 1개와 작업자 노드 3개가 있습니다.

```
NAMESPACE             NAME             CONTROL PLANE   WORKER   TKR
NAME                           AGE     READY
tkgs-cluster-ns       test-cluster     1               3        v1.21.2---vmware.1-
tkg.1.13da849   5d12h   True
```

5 `kubectl edit` 명령을 사용하여 편집할 클러스터 매니페스트를 로드합니다.

```
kubectl edit tanzukubernetescluster/tkgs-cluster-1
```

KUBE_EDITOR 또는 EDITOR 환경 변수로 정의된 텍스트 편집기에서 클러스터 매니페스트가 열립니다.

6 `spec.topology.controlPlane.count` 매개 변수를 찾아서 노드 수를 1에서 3으로 늘립니다.

```
...
controlPlane:
    replicas: 1
...
```

```
...
ControlPlane:
    replicas: 3
...
```

7 변경 내용을 적용하려면 텍스트 편집기에서 파일을 저장합니다. 취소하려면 저장하지 않고 편집기를 닫습니다.

파일을 저장하면 kubectl이 변경 내용을 클러스터에 적용합니다. 백그라운드에서 감독자 클러스터의 가상 시스템 서비스는 새 작업자 노드를 프로비저닝합니다.

8 새 노드가 추가되었는지 확인합니다.

```
kubectl get tanzukubernetescluster tkgs-cluster-1
```

확장된 제어부에 이제 3개의 노드가 있습니다.

```
NAMESPACE             NAME             CONTROL PLANE   WORKER   TKR
NAME                            AGE     READY
tkgs-cluster-ns       test-cluster     3               3        v1.21.2---vmware.1-
tkg.1.13da849   5d12h   True
```

작업자 노드 확장

kubectl을 사용하여 작업자 노드 수를 늘려서 Tanzu Kubernetes 클러스터를 확장할 수 있습니다.

1 감독자 클러스터로 인증합니다.

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

2 Tanzu Kubernetes 클러스터를 실행 중인 vSphere 네임스페이스로 컨텍스트를 전환합니다.

```
kubectl config use-context tkgs-cluster-ns
```

3 네임스페이스에서 실행 중인 Kubernetes 클러스터를 나열합니다.

```
kubectl get tanzukubernetescluster -n tkgs-cluster-ns
```

4 대상 클러스터에서 실행 중인 노드 수를 구합니다.

```
kubectl get tanzukubernetescluster tkgs-cluster-1
```

예를 들어 다음 클러스터에는 제어부 노드 3개와 작업자 노드 3개가 있습니다.

```
NAMESPACE             NAME             CONTROL PLANE   WORKER   TKR
NAME                            AGE     READY
tkgs-cluster-ns       test-cluster     3               3        v1.21.2---vmware.1-
tkg.1.13da849   5d12h   True
```

5 `kubectl edit` 명령을 사용하여 편집할 클러스터 매니페스트를 로드합니다.

```
kubectl edit tanzukubernetescluster/tkgs-cluster-1
```

KUBE_EDITOR 또는 EDITOR 환경 변수로 정의된 텍스트 편집기에서 클러스터 매니페스트가 열립니다.

6 spec.topology.workers.count 매개 변수를 찾아서 노드 수를 늘립니다.

```
...
workers:
    replicas: 3
...
```

```
...
workers:
    replicas: 4
...
```

7 변경 내용을 적용하려면 텍스트 편집기에서 파일을 저장합니다. 취소하려면 저장하지 않고 편집기를 닫습니다.

파일을 저장하면 kubectl이 변경 내용을 클러스터에 적용합니다. 백그라운드에서 감독자 클러스터의 가상 시스템 서비스는 새 작업자 노드를 프로비저닝합니다.

8 새 작업자 노드가 추가되었는지 확인합니다.

```
kubectl get tanzukubernetescluster tkgs-cluster-1
```

확장한 후에는 클러스터에 작업자 노드가 4개 있습니다.

```
NAMESPACE              NAME              CONTROL PLANE   WORKER   TKR
NAME                              AGE     READY
tkgs-cluster-ns        test-cluster      3               4        v1.21.2---vmware.1-
tkg.1.13da849   5d12h   True
```

작업자 노드 축소

작업자 노드 수를 줄여서 Tanzu Kubernetes 클러스터를 축소할 수 있습니다. 제어부에서는 확장/축소가 지원되지 않습니다.

1 감독자 클러스터로 인증합니다.

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

2 Tanzu Kubernetes 클러스터를 실행 중인 vSphere 네임스페이스로 컨텍스트를 전환합니다.

```
kubectl config use-context tkgs-cluster-ns
```

3 네임스페이스에서 실행 중인 Kubernetes 클러스터를 나열합니다.

```
kubectl get tanzukubernetescluster -n tkgs-cluster-ns
```

4 대상 클러스터에서 실행 중인 노드 수를 구합니다.

```
kubectl get tanzukubernetescluster tkgs-cluster-1
```

예를 들어 다음 클러스터에는 제어부 노드 3개와 작업자 노드 4개가 있습니다.

```
NAMESPACE             NAME             CONTROL PLANE   WORKER   TKR
NAME                            AGE     READY
tkgs-cluster-ns       test-cluster     3               4        v1.21.2---vmware.1-
tkg.1.13da849   5d12h   True
```

5 `kubectl edit` 명령을 사용하여 편집할 클러스터 매니페스트를 로드합니다.

```
kubectl edit tanzukubernetescluster/tkgs-cluster-1
```

KUBE_EDITOR 또는 EDITOR 환경 변수로 정의된 텍스트 편집기에서 클러스터 매니페스트가 열립니다.

6 `spec.topology.workers.count` 매개 변수를 찾아서 노드 수를 줄입니다.

```
...
workers:
    replicas: 4
...
```

```
...
workers:
    replicas: 2
...
```

7 변경 내용을 적용하려면 텍스트 편집기에서 파일을 저장합니다. 취소하려면 저장하지 않고 편집기를 닫습니다.

파일을 저장하면 kubectl이 변경 내용을 클러스터에 적용합니다. 백그라운드에서 감독자 클러스터의 가상 시스템 서비스는 새 작업자 노드를 프로비저닝합니다.

8 작업자 노드가 제거되었는지 확인합니다.

```
kubectl get tanzukubernetescluster tkgs-cluster-1
```

축소한 후에는 클러스터에 작업자 노드가 2개 있습니다.

```
NAMESPACE             NAME             CONTROL PLANE   WORKER   TKR
NAME                            AGE     READY
tkgs-cluster-ns       test-cluster     3               2        v1.21.2---vmware.1-
tkg.1.13da849   5d12h   True
```

클러스터를 수직으로 확장/축소

클러스터 노드를 호스팅하는 데 사용되는 가상 시스템 클래스를 변경하여 Tanzu Kubernetes 클러스터를 수직으로 확장/축소할 수 있습니다. 수직 확장/축소는 제어부 및 작업자 노드 모두에서 지원됩니다.

Tanzu Kubernetes Grid 서비스는 서비스에 내장된 롤링 업데이트 메커니즘을 통해 클러스터 노드를 수직으로 확장/축소하도록 지원합니다. `VirtualMachineClass` 정의를 변경하는 경우 이 서비스는 새 클래스를 사용하여 새 노드를 롤아웃하고 이전 노드를 스핀 다운합니다. Tanzu Kubernetes 클러스터 업데이트의 내용을 참조하십시오.

1 감독자 클러스터로 인증합니다.

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

2 Tanzu Kubernetes 클러스터를 실행 중인 vSphere 네임스페이스로 컨텍스트를 전환합니다.

```
kubectl config use-context tkgs-cluster-ns
```

3 네임스페이스에서 실행 중인 Kubernetes 클러스터를 나열합니다.

```
kubectl get tanzukubernetescluster -n tkgs-cluster-ns
```

4 대상 Tanzu Kubernetes 클러스터를 설명하고 VM 클래스를 확인합니다.

```
kubectl describe tanzukubernetescluster tkgs-cluster-2
```

예를 들어 다음 클러스터는 best-effort-medium VM 클래스를 사용합니다.

```
Spec:
  ...
  Topology:
    Control Plane:
      Class:          best-effort-medium
      ...
    nodePool-a1:
      Class:          best-effort-medium
      ...
```

5 사용 가능한 VM 클래스를 나열하고 설명합니다.

```
kubectl get virtualmachineclassbinding
```

```
kubectl describe virtualmachineclassbinding
```

참고 사용하려는 VM 클래스는 vSphere 네임스페이스에 바인딩되어야 합니다. Tanzu Kubernetes 클러스터에 대한 가상 시스템 클래스의 내용을 참조하십시오.

6 대상 클러스터 매니페스트를 편집하기 위해 엽니다.

```
kubectl edit tanzukubernetescluster/tkgs-cluster-2
```

KUBE_EDITOR 또는 EDITOR 환경 변수로 정의된 텍스트 편집기에서 클러스터 매니페스트가 열립니다.

7 VM 클래스를 변경하여 매니페스트를 편집합니다.

예를 들어 제어부 및 작업자 노드에 대해 `guaranteed-large` VM 클래스를 사용하도록 클러스터 매니페스트를 편집합니다.

```
spec:
  topology:
    controlPlane:
      class: guaranteed-large
      ...
    nodePool-a1:
      class: guaranteed-large
      ...
```

8 변경 내용을 적용하려면 텍스트 편집기에서 파일을 저장합니다. 취소하려면 저장하지 않고 편집기를 닫습니다.

파일을 저장하면 kubectl이 변경 내용을 클러스터에 적용합니다. 백그라운드에서 Tanzu Kubernetes Grid 서비스는 새 노드를 프로비저닝하고 이전 노드를 삭제합니다. 롤링 업데이트 프로세스에 대한 설명은 Tanzu Kubernetes Grid 서비스 클러스터 업데이트 정보 항목을 참조하십시오.

9 클러스터가 업데이트되었는지 확인합니다.

```
kubectl get tanzukubernetescluster
NAMESPACE             NAME             CONTROL PLANE   WORKER   TKR
NAME                              AGE     READY
tkgs-cluster-ns       test-cluster     3               3        v1.21.2---vmware.1-
tkg.1.13da849   5d12h   True
```

노드 볼륨 확장/축소

노드에 대한 Tanzu Kubernetes 클러스터 규격에는 하나 이상의 영구 볼륨을 선언하는 옵션이 있습니다. 노드 볼륨 선언은 제어부의 ectd 데이터베이스 및 작업자 노드의 컨테이너 런타임과 같이 변동률이 높은 구성 요소에 유용합니다. 참조를 위해 이러한 노드 볼륨이 모두 선언된 클러스터 규격 발췌가 아래에 제공됩니다. (전체 예제 클러스터 규격은 TKGS v1alpha2 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하기 위한 예제 YAML에서 확인할 수 있습니다.)

클러스터 생성 후 하나 이상의 노드 볼륨을 추가하거나 변경하려면 다음 고려 사항에 유의하십시오.

볼륨 노드	설명
작업자 노드 볼륨 변경이 허용됩니다.	Tanzu Kubernetes 클러스터가 프로비저닝된 후 작업자 노드 볼륨을 추가하거나 업데이트할 수 있습니다. 롤링 업데이트를 시작하면 클러스터가 새 볼륨 또는 변경된 볼륨으로 업데이트됩니다. **경고** 새 볼륨 또는 변경된 볼륨으로 작업자 노드를 확장하는 경우 롤링 업데이트 중에 현재 볼륨의 데이터가 삭제됩니다.
제어부 노드 볼륨 변경은 허용되지 않습니다.	Tanzu Kubernetes 클러스터가 프로비저닝된 후에는 제어부 노드 볼륨을 추가하거나 업데이트할 수 없습니다. Kubernetes CAPI(Cluster API)는 `spec.toplogy.controlPlane.volumes`에 대한 생성 후 변경을 금지합니다. 클러스터 생성 후 제어부 볼륨을 추가하거나 변경하려고 하면 요청이 거부되고 "불변 필드에 대한 업데이트는 허용되지 않습니다."라는 오류 메시지가 표시됩니다.

```
spec:
  topology:
    controlPlane:
      replicas: 3
      vmClass: guaranteed-medium
      storageClass: vwt-storage-policy
      volumes:
        - name: etcd
          mountPath: /var/lib/etcd
          capacity:
            storage: 4Gi
      tkr:
        reference:
          name: v1.21.2---vmware.1-tkg.1.ee25d55
    nodePools:
    - name: worker-nodepool-a1
      replicas: 3
      vmClass: guaranteed-large
      storageClass: vwt-storage-policy
      volumes:
        - name: containerd
          mountPath: /var/lib/containerd
          capacity:
            storage: 16Gi
```

Tanzu Kubernetes Grid 서비스 v1alpha1 API를 사용하여 Tanzu Kubernetes 클러스터 프로비저닝

이 섹션에서는 Tanzu Kubernetes Grid 서비스 v1alpha1 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하는 방법을 설명합니다.

Tanzu Kubernetes Grid 서비스 v1alpha1 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하는 워크플로

Tanzu Kubernetes 클러스터는 YAML을 사용하여 정의한 클러스터 규격 및 kubectl을 사용하여 Tanzu Kubernetes Grid 서비스 선언적 API를 호출하여 프로비저닝합니다. 클러스터를 프로비저닝한 후 클러스터를 운영하고 kubectl을 사용하여 워크로드를 배포합니다.

이 워크플로는 클러스터 프로비저닝 프로세스에 대한 전체 절차를 제공합니다. 각 단계에는 특정 작업에 대한 자세한 정보를 제공하는 링크가 있습니다.

사전 요구 사항

다음 사전 요구 사항을 완료하십시오.

- 감독자 클러스터 인스턴스를 구성합니다. 장 5 감독자 클러스터 구성 및 관리의 내용을 참조하십시오.
- Tanzu Kubernetes 클러스터를 프로비저닝할 vSphere 네임스페이스를 생성합니다. vSphere 네임스페이스 생성 및 구성의 내용을 참조하십시오.

  초기 네임스페이스 구성에는 다음이 필요합니다.

  - 한 명 이상의 DevOps 엔지니어가 vCenter Single Sign-On 자격 증명을 사용하여 네임스페이스에 액세스할 수 있는 권한을 편집합니다.
  - 네임스페이스에 대한 태그 기반 공유 스토리지 정책.
  - 필요에 따라 확인 및 조정된 네임스페이스에 대한 용량 및 사용 할당량.
- 공유 데이터스토어에 Tanzu Kubernetes 릴리스용 컨텐츠 라이브러리를 생성하고 사용하려는 릴리스를 동기화합니다. Tanzu Kubernetes 릴리스용 컨텐츠 라이브러리 생성 및 관리의 내용을 참조하십시오.
- 컨텐츠 라이브러리 및 가상 시스템 클래스를 vSphere 네임스페이스와 연결합니다. Tanzu Kubernetes 릴리스에 대한 vSphere 네임스페이스 구성의 내용을 참조하십시오.

절차

1 vSphere에 대한 Kubernetes CLI 도구를 다운로드하고 설치합니다. vSphere에 대한 Kubernetes CLI 도구 다운로드 및 설치의 내용을 참조하십시오.

2 kubectl용 vSphere 플러그인을 사용하여 감독자 클러스터로 인증합니다. vCenter Single Sign-On 사용자로 감독자 클러스터에 연결의 내용을 참조하십시오.

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

3 kubectl을 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하려는 vSphere 네임스페이스로 컨텍스트를 전환합니다.

```
kubectl config get-contexts
```

```
kubectl config use-context SUPERVISOR-NAMESPACE
```

예:

```
kubectl config use-context tkgs-cluster-ns
```

4 사용 가능한 가상 시스템 클래스 바인딩을 나열합니다. Tanzu Kubernetes 클러스터에 대한 가상 시스템 클래스의 내용을 참조하십시오.

다음 명령을 사용하여 클러스터를 배포하는 vSphere 네임스페이스에서 사용할 수 있는 모든 VM 클래스 바인딩을 나열합니다.

```
kubectl get virtualmachineclassbindings
```

참고 `kubectl get virtualmachineclasses` 명령은 감독자 클러스터에 있는 모든 VM 클래스를 나열합니다. VM 클래스를 vSphere 네임스페이스와 연결해야 하기 때문에 대상 네임스페이스에 바인딩된 VM 클래스만 사용할 수 있습니다.

5 네임스페이스를 설명하여 사용 가능한 기본 스토리지 클래스를 가져옵니다.

```
kubectl describe namespace SUPERVISOR-NAMESPACE
```

6 사용 가능한 Tanzu Kubernetes 릴리스를 나열합니다.

참고 호환성은 Tanzu Kubernetes 릴리스 목록을 참조하십시오. 업데이트를 위한 Tanzu Kubernetes 클러스터 호환성 확인의 내용을 참조하십시오.

```
kubectl get tanzukubernetesreleases
```

참고 `kubectl get virtualmachineimages` 명령은 가상 시스템에 대한 일반 정보를 반환합니다.

7 Tanzu Kubernetes 클러스터 프로비저닝을 위한 YAML 파일을 생성합니다.

a 예제 YAML 파일 중 하나로 시작합니다. Tanzu Kubernetes Grid 서비스 v1alpha1 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하는 예의 내용을 참조하십시오.

예를 들어 다음 YAML 파일은 사용 가능한 모든 기본값을 사용하여 최소 클러스터를 프로비저닝합니다.

```
apiVersion: run.tanzu.vmware.com/v1alpha1  #TKGS API endpoint
kind: TanzuKubernetesCluster               #required parameter
metadata:
  name: tkgs-cluster-1                     #cluster name, user defined
  namespace: tgks-cluster-ns               #vsphere namespace
spec:
  distribution:
    version: v1.19                         #Resolves to latest TKR 1.19 version
  topology:
    controlPlane:
      count: 1                             #number of control plane nodes
      class: best-effort-medium            #vmclass for control plane nodes
      storageClass: vwt-storage-policy     #storageclass for control plane
    workers:
      count: 3                             #number of worker nodes
      class: best-effort-medium            #vmclass for worker nodes
      storageClass: vwt-storage-policy     #storageclass for worker nodes
```

b 앞의 명령 출력에서 모은 정보를 사용하여 네임스페이스, 스토리지 클래스 및 가상 시스템 클래스를 포함하여 클러스터 YAML을 채웁니다.

c 클러스터 구성 매개 변수의 전체 목록을 참조하여 필요에 따라 클러스터를 사용자 지정합니다. Tanzu Kubernetes Grid 서비스 v1alpha1 API를 사용하는 Tanzu Kubernetes 클러스터에 대한 구성 매개 변수의 내용을 참조하십시오.

d 파일을 `tkgs-cluster-1.yaml` 또는 이와 유사하게 저장합니다.

8 다음 kubectl 명령을 실행하여 클러스터를 프로비저닝합니다.

```
kubectl apply -f CLUSTER-NAME.yaml
```

예:

```
kubectl apply -f tkgs-cluster-1.yaml
```

예상 결과:

```
tanzukubernetescluster.run.tanzu.vmware.com/tkgs-cluster-1 created
```

9 kubectl을 사용하여 클러스터 노드의 배포를 모니터링합니다. kubectl을 사용하여 Tanzu Kubernetes 클러스터 상태 모니터링의 내용을 참조하십시오.

```
kubectl get tanzukubernetesclusters
```

샘플 결과:

```
NAME             CONTROL PLANE   WORKER   DISTRIBUTION                      AGE     PHASE
tkgs-cluster-2   1               3        v1.19.7+vmware.1-tkg.1.c40d30d    7m59s   running
```

10 vSphere Client를 사용하여 클러스터 노드의 배포를 모니터링합니다. vSphere Client를 사용하여 Tanzu Kubernetes 클러스터 상태 모니터링의 내용을 참조하십시오.

예를 들어 vSphere 인벤토리에서 네임스페이스에 배포되는 가상 시스템 노드를 볼 수 있습니다.

11 추가 명령을 실행하여 클러스터 프로비저닝을 확인합니다. 자세한 내용은 Tanzu Kubernetes 클러스터 작동 명령 사용을 참조하십시오.

예:

```
kubectl get tanzukubernetescluster,cluster-
api,virtualmachinesetresourcepolicy,virtualmachineservice,virtualmachine
```

참고 추가 문제 해결은 Tanzu Kubernetes 클러스터 문제 해결을 참조하십시오.

12 kubectl용 vSphere 플러그인을 사용하여 클러스터에 로그인합니다. vCenter Single Sign-On 사용자로 Tanzu Kubernetes 클러스터에 연결의 내용을 참조하십시오.

```
kubectl vsphere login --server=IP-ADDRESS --vsphere-username USERNAME \
--tanzu-kubernetes-cluster-name CLUSTER-NAME --tanzu-kubernetes-cluster-namespace
NAMESPACE-NAME
```

13 다음 kubectl 명령을 사용하여 클러스터 프로비저닝을 확인합니다.

```
kubectl cluster-info
```

```
kubectl get nodes
```

```
kubectl get namespaces
```

```
kubectl api-resources
```

14 예제 워크로드를 배포하고 클러스터 생성을 확인합니다. Tanzu Kubernetes 클러스터에 워크로드 배포의 내용을 참조하십시오.

참고 Tanzu Kubernetes 클러스터에는 포드 보안 정책이 사용되도록 설정되어 있습니다. 워크로드 및 사용자에 따라 적절한 RoleBinding 또는 사용자 지정 PodSecurityPolicy를 생성해야 할 수도 있습니다. Tanzu Kubernetes 클러스터에서 포드 보안 정책 사용의 내용을 참조하십시오.

15 TKG 확장을 배포하여 클러스터를 운영합니다. Tanzu Kubernetes 클러스터에 TKG 패키지 배포의 내용을 참조하십시오.

Tanzu Kubernetes Grid 서비스 v1alpha1 API를 사용하는 Tanzu Kubernetes 클러스터에 대한 구성 매개 변수

Tanzu Kubernetes Grid 서비스 선언적 API는 Tanzu Kubernetes 클러스터 구성을 위한 여러 매개 변수를 노출합니다. 클러스터를 프로비저닝하고 사용자 지정하려면 모든 매개 변수 및 사용 지침에 대한 목록과 설명을 참조하십시오.

Tanzu Kubernetes 클러스터 프로비저닝을 위한 주석이 달린 YAML

주석이 달린 YAML에는 Tanzu Kubernetes 클러스터를 프로비저닝하는 데 사용할 수 있는 모든 매개 변수가 각 필드에 대한 요약 설명과 함께 나열되어 있습니다.

참고 주석이 달린 YAML은 클러스터 프로비저닝을 위한 유효성이 검사되지 않습니다. 이러한 지침에 대한 예는 Tanzu Kubernetes Grid 서비스 v1alpha1 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하는 예 항목을 참조하십시오.

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TanzuKubernetesCluster
metadata:
  name: <tanzu kubernetes cluster name>
  namespace: <vsphere namespace where the cluster will be provisioned>
spec:
  distribution:
    version: <tanzu kubernetes release version string: full, point, short>
  topology:
    controlPlane:
      count: <integer either 1 or 3>
      class: <vm class bound to the target vsphere namespace>
      storageClass: <vsphere storage policy bound to the target vsphere namespace>
      volumes: #optional setting for high-churn control plane component (such as etcd)
        - name: <user-defined string>
          mountPath: </dir/path>
          capacity:
            storage: <size in GiB>
    workers:
      count: <integer from 0 to 150>
      class: <vm class bound to the target vsphere namespace>
      storageClass: <vsphere storage policy bound to the target vsphere namespace>
      volumes: #optional setting for high-churn worker node component (such as containerd)
        - name: <user-defined string>
          mountPath: </dir/path>
          capacity:
            storage: <size in GiB>
  settings: #all spec.settings are optional
    storage: #optional storage settings
      classes: [<array of kubernetes storage classes for dynamic pvc provisioning>]
      defaultClass: <default kubernetes storage class>
    network: #optional network settings
      cni: #override default cni set in the tkgservicesonfiguration spec
        name: <antrea or calico>
      pods: #custom pod network
```

```
    cidrBlocks: [<array of pod cidr blocks>]
  services: #custom service network
    cidrBlocks: [<array of service cidr blocks>]
  serviceDomain: <custom service domain>
  proxy: #proxy server for outbound connections
    httpProxy: http://<IP:PORT>
    httpsProxy: http://<IP:PORT>
    noProxy: [<array of CIDRs to not proxy>]
  trust: #trust fields for custom public certs for tls
    additionalTrustedCAs:
      - name: <first-cert-name>
        data: <base64-encoded string of PEM encoded public cert 1>
      - name: <second-cert-name>
        data: <base64-encoded string of PEM encoded public cert 2>
```

Tanzu Kubernetes 클러스터 프로비저닝을 위한 매개 변수

다음 표에는 Tanzu Kubernetes 클러스터를 프로비저닝하기 위한 모든 매개 변수와 허용되는 값이 나열 및 설명되어 있습니다. 예제에 대해서는 Tanzu Kubernetes Grid 서비스 v1alpha1 API 구성 예시 항목을 참조하십시오.

표 13-4. Tanzu Kubernetes 클러스터 프로비저닝을 위한 매개 변수

이름	값	설명
apiVersion	run.tanzu.vmware.com/v1alpha1	Tanzu Kubernetes Grid 서비스 API의 버전을 지정합니다.
kind	TanzuKubernetesCluster	생성할 Kubernetes 리소스의 유형을 지정합니다. 허용되는 유일한 값은 TanzuKubernetesCluster(대/소문자 구분)입니다.
metadata	클러스터 메타데이터에 대한 섹션	클러스터 메타데이터(예: name 및 namespace)를 포함합니다. 이는 표준 Kubernetes 메타데이터이므로 name 대신 generateName을 사용하고 레이블과 주석 등을 추가할 수 있습니다.
name	영숫자 문자와 대시를 허용하는 사용자 정의 문자열(예: my-tkg-cluster-1)	생성할 클러스터의 이름을 지정합니다. 현재 클러스터 이름 지정 제약 조건: ■ 이름 길이는 41자 이하여야 합니다. ■ 이름은 문자로 시작해야 합니다. ■ 이름에는 문자, 숫자 및 하이픈이 포함될 수 있습니다. ■ 이름은 문자 또는 숫자로 끝나야 합니다.
namespace	영숫자 문자와 대시를 허용하는 사용자 정의 문자열(예: my-sns-1)	클러스터가 배포될 감독자 네임스페이스의 이름을 식별합니다. 감독자 클러스터에 있는 감독자 네임스페이스에 대한 참조입니다.

표 13-4. Tanzu Kubernetes 클러스터 프로비저닝을 위한 매개 변수 (계속)

이름	값	설명
spec	클러스터의 기술 규격에 대한 섹션	노드 topology 및 Kubernetes 소프트웨어 distribution을 포함하여 클러스터의 최종 상태에 대해 선언적 방식으로 표시된 규격을 포함합니다.
distribution	Tanzu Kubernetes 릴리스 버전 지정에 대한 섹션	클러스터(Kubernetes 자체를 포함하여 제어부 및 작업자 노드에 설치된 Tanzu Kubernetes 클러스터 소프트웨어)에 대한 배포를 나타냅니다.
version	Kubernetes 버전을 나타내는 대시가 포함된 영숫자 문자열(예: v1.20.2+vmware.1-tkg.1 또는 v1.20.2 또는 v1.20)	의미 버전 표기법을 사용하여 클러스터 노드에 설치할 Kubernetes 배포의 소프트웨어 버전을 지정합니다. 정규화된 버전을 지정하거나 버전 바로 가기를 사용할 수 있습니다. 해당 패치 버전과 일치하는 최신 이미지로 확인된 "버전: v1.20.2" 또는 일치하는 최신 패치 버전으로 확인된 "버전: v1.20"을 예로 들 수 있습니다. 확인된 버전은 생성된 후 클러스터 설명에 "fullVersion"으로 표시됩니다.
topology	클러스터 노드 토폴로지에 대한 섹션	클러스터 노드의 수, 용도 및 구성과 각 클러스터 노드에 할당된 리소스를 설명하는 필드를 포함합니다. 클러스터 노드는 해당 용도에 따라 control-plane 또는 worker 풀로 그룹화됩니다. 각 풀은 동일한 리소스 할당을 가지며 동일한 스토리지를 사용하는 동종으로 구성됩니다.
controlPlane	제어부 설정에 대한 섹션	노드 수(count), VM 유형(class), 각 노드에 할당된 스토리지 리소스(storageClass)를 포함하여 클러스터 제어부의 토폴로지를 지정합니다.
count	1 또는 3인 정수	제어부 노드의 수를 지정합니다. 제어부에는 홀수의 노드가 있어야 합니다.
class	열거된 집합(예: guaranteed-small 또는 best-effort-large)의 문자열 형식으로 된 시스템 정의 요소	풀의 각 노드에 사용될 가상 하드웨어 설정을 설명하는 VirtualMachineClass의 이름을 지정합니다. 이것은 노드(CPU 및 메모리)에 사용 가능한 하드웨어와 그러한 리소스에 대한 제한 및 요청을 제어합니다. Tanzu Kubernetes 클러스터에 대한 가상 시스템 클래스의 내용을 참조하십시오.

표 13-4. Tanzu Kubernetes 클러스터 프로비저닝을 위한 매개 변수 (계속)

이름	값	설명
storageClass	node-storage(예)	제어부 노드의 루트 파일 시스템을 저장하는 디스크의 스토리지에 사용될 스토리지 클래스를 식별합니다. 네임스페이스에서 kubectl describe ns를 실행하여 사용 가능한 스토리지 클래스를 봅니다. 네임스페이스에 사용 가능한 스토리지 클래스는 vSphere 관리자가 설정한 스토리지에 따라 다릅니다. 감독자 네임스페이스와 연결된 스토리지 클래스는 클러스터에 복제됩니다. 즉, 스토리지 클래스가 이 필드에 대해 유효한 값이 되려면 감독자 네임스페이스에서 사용 가능해야 합니다. 장 7 vSphere 네임스페이스 구성 및 관리의 내용을 참조하십시오.
volumes	선택적 스토리지 설정 ■ 볼륨: ■ 이름: *string* ■ 마운트 경로: */dir/path* ■ 용량 ■ 스토리지: GiB 크기	제어부 노드의 etcd에 대해 별도의 디스크 및 스토리지 매개 변수를 지정할 수 있습니다. 별도의 디스크 및 스토리지 매개 변수가 있는 클러스터의 예를 참조하십시오.
workers	작업자 노드 설정에 대한 섹션	노드 수(count), VM 유형(class), 각 노드에 할당된 스토리지 리소스(storageClass)를 포함하여 클러스터 작업자 노드의 토폴로지를 지정합니다.
count	0에서 150 사이의 정수(예: 1 또는 2 또는 7)	클러스터의 작업자 노드 수를 지정합니다. 작업자 노드가 없는 클러스터를 생성하여 제어부 노드만 있는 클러스터를 허용할 수 있습니다. 작업자 노드 수에 대해 규정된 최대값은 없지만 적절한 제한은 150입니다. **참고** 0개의 작업자 노드로 프로비저닝된 클러스터에는 로드 밸런서 서비스가 할당되지 않습니다.
class	열거된 집합(예: guaranteed-small 또는 best-effort-large)의 문자열 형식으로 된 시스템 정의 요소	풀의 각 노드에 사용될 가상 하드웨어 설정을 설명하는 VirtualMachineClass의 이름을 지정합니다. 이것은 노드(CPU 및 메모리)에 사용 가능한 하드웨어와 그러한 리소스에 대한 제한 및 요청을 제어합니다. Tanzu Kubernetes 클러스터에 대한 가상 시스템 클래스의 내용을 참조하십시오.

표 13-4. Tanzu Kubernetes 클러스터 프로비저닝을 위한 매개 변수 (계속)

이름	값	설명
storageClass	node-storage(예)	작업자 노드의 루트 파일 시스템을 저장하는 디스크의 스토리지에 사용될 스토리지 클래스를 식별합니다. 네임스페이스에서 kubectl describe ns를 실행하여 사용 가능한 스토리지 클래스를 나열합니다. 네임스페이스에 사용 가능한 스토리지 클래스는 vSphere 관리자가 설정한 스토리지에 따라 다릅니다. 감독자 네임스페이스와 연결된 스토리지 클래스는 클러스터에 복제됩니다. 즉, 스토리지 클래스가 유효하려면 감독자 네임스페이스에서 사용 가능해야 합니다. 장 7 vSphere 네임스페이스 구성 및 관리의 내용을 참조하십시오.
volumes	선택적 스토리지 설정 ■ 볼륨: ■ 이름: *string* ■ 마운트 경로: */dir/path* ■ 용량 ■ 스토리지: GiB 크기	작업자 노드의 컨테이너 이미지에 대해 별도의 디스크 및 스토리지 매개 변수를 지정할 수 있습니다. 별도의 디스크 및 스토리지 매개 변수가 있는 클러스터의 예를 참조하십시오.
settings	클러스터별 설정에 대한 섹션; 모든 spec.settings는 선택 사항입니다.	포드에 대한 영구 storage 및 노드 network 세부 정보를 포함하여 클러스터에 대한 선택적 런타임 구성 정보를 식별합니다.
storage	스토리지 지정에 대한 섹션	컨테이너 워크로드에 대한 PV(영구 볼륨) 스토리지 항목을 식별합니다.
classes	하나 이상의 사용자 정의 문자열의 어레이(예: ["gold", "silver"])	컨테이너 워크로드에 대한 명명된 PV(영구 볼륨) 스토리지 클래스를 지정합니다. 감독자 네임스페이스와 연결된 스토리지 클래스는 클러스터에 복제됩니다. 즉, 스토리지 클래스는 유효한 값이 되려면 감독자 네임스페이스에서 사용 가능해야 합니다. 스토리지 클래스및 영구 볼륨에 대한 기본 클래스가 있는 클러스터의 예를 참조하십시오.
defaultClass	silver(예)	클러스터에서 기본값으로 주석 처리될 명명된 스토리지 클래스를 지정합니다. 이를 지정하지 않으면 기본값이 없습니다. defaultClass를 지정하기 위해 하나 이상의 classes를 지정하지 않아도 됩니다. 일부 워크로드에는 Helm과 같은 기본 클래스가 필요할 수 있습니다. 스토리지 클래스및 영구 볼륨에 대한 기본 클래스가 있는 클러스터의 예를 참조하십시오.

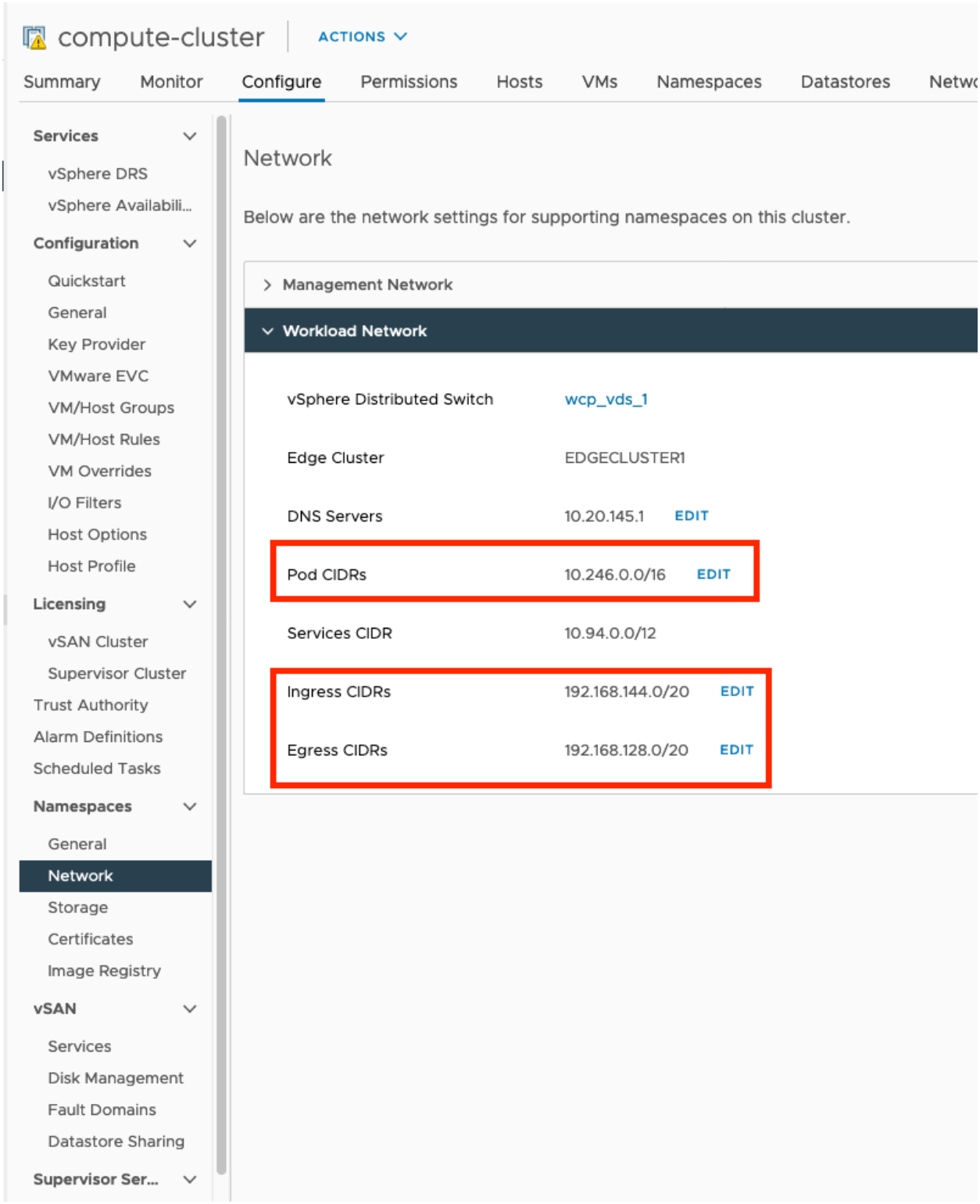
표 13-4. Tanzu Kubernetes 클러스터 프로비저닝을 위한 매개 변수 (계속)

이름	값	설명
network	네트워킹 설정에 대한 섹션 마커	클러스터에 대한 네트워크 관련 설정을 지정합니다.
cni	CNI 지정에 대한 섹션 마커	클러스터에 대한 CNI(컨테이너 네트워킹 인터페이스) 플러그인을 식별합니다. 기본값은 Antrea이며 새 클러스터에 대해 지정할 필요가 없습니다.
name	문자열 antrea 또는 calico	사용할 CNI를 지정합니다. Antrea 및 Calico가 지원됩니다. 시스템 구성은 Antrea를 기본 CNI로 설정합니다. 기본 CNI를 변경할 수 있습니다. 기본값을 사용하는 경우에는 이 필드를 지정할 필요가 없습니다.
services	Kubernetes 서비스 서브넷을 지정하기 위한 섹션 마커	Kubernetes 서비스에 대한 네트워크 설정을 식별합니다. 기본값은 10.96.0.0/12입니다.
cidrBlocks	어레이 ["198.51.100.0/12"](예)	Kubernetes 서비스에 사용할 IP 주소 범위를 지정합니다. 기본값은 10.96.0.0/12입니다. 감독자 클러스터에 대해 선택한 설정과 겹치지 않아야 합니다. 이 필드는 어레이이므로 여러 범위를 허용하기는 하지만 현재는 단일 IP 범위만 허용됩니다. Tanzu Kubernetes Grid 서비스 v1alpha1 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하는 예에서 네트워킹 예제를 참조하십시오.
pods	Kubernetes 포드 서브넷을 지정하기 위한 섹션 마커	포드에 대한 네트워크 설정을 지정합니다. 기본값은 192.168.0.0/16입니다. 최소 블록 크기는 /24입니다.
cidrBlocks	어레이 ["192.0.2.0/16"](예)	Kubernetes 포드에 사용할 IP 주소 범위를 지정합니다. 기본값은 192.168.0.0/16입니다. 감독자 클러스터에 대해 선택한 설정과 겹치지 않아야 합니다. 포드 서브넷 크기는 /24보다 크거나 같아야 합니다. 이 필드는 어레이이므로 여러 범위를 허용하기는 하지만 현재는 단일 IP 범위만 허용됩니다. Tanzu Kubernetes Grid 서비스 v1alpha1 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하는 예에서 네트워킹 예제를 참조하십시오.
serviceDomain	"cluster.local"	클러스터에 대한 서비스 도메인을 지정합니다. 기본값은 cluster.local입니다.

표 13-4. Tanzu Kubernetes 클러스터 프로비저닝을 위한 매개 변수 (계속)

이름	값	설명
proxy	클러스터에 대한 HTTP(s) 프록시 구성을 지정하는 섹션. 구현된 경우 모든 필드가 필요합니다.	지정된 프록시 설정에 대한 필드를 제공합니다. 글로벌 프록시가 구성되고 개별 클러스터 프록시가 구성되지 않은 경우 자동으로 채워집니다. 프록시 서버가 있는 클러스터의 예를 참조하십시오.
httpProxy	http://<user>:<pwd>@<ip>:<port>	클러스터 외부에서 HTTP 연결을 생성하는 데 사용할 프록시 URL을 지정합니다.
httpsProxy	http://<user>:<pwd>@<ip>:<port>	클러스터 외부에서 HTTPS 연결을 생성하는 데 사용할 프록시 URL을 지정합니다.
noProxy	프록시를 사용하지 않을 CIDR 블록의 어레이입니다(예: [10.246.0.0/16,192.168.144.0/20,192.168.128.0/20]). 감독자 클러스터의 워크로드 네트워크에서 필요한 값을 가져옵니다(예: Pod CIDRs, Ingress CIDRs 및 Egress CIDRs). noProxy 어레이 필드에 포함할 값은 아래 이미지를 참조하십시오.	포드, 수신 및 송신을 위해 감독자 클러스터의 워크로드 네트워크에서 사용하는 서브넷에는 프록시를 사용하지 말아야 합니다. 감독자 클러스터의 서비스 CIDR을 noProxy 필드에 포함할 필요가 없습니다. Tanzu Kubernetes 클러스터는 그러한 서비스와 상호 작용하지 않습니다. 끝점 localhost 및 127.0.0.1은 자동으로 프록시로 사용되지 않습니다. noProxy 필드에 추가할 필요가 없습니다. Tanzu Kubernetes 클러스터의 포드 및 서비스 CIDR은 자동으로 프록시로 사용되지 않습니다. noProxy 필드에 추가할 필요가 없습니다. 프록시 서버가 있는 클러스터의 예를 참조하십시오.
trust	trust 매개 변수의 섹션 마커입니다.	데이터를 허용하지 않습니다.
additionalTrustedCAs	각각 name 및 data가 있는 인증서 어레이를 허용합니다.	데이터를 허용하지 않습니다.
name	문자열	TLS 인증서의 이름입니다.
data	문자열	PEM으로 인코딩된 공용 인증서의 base64로 인코딩된 문자열입니다.

이미지에 표시된 대로 감독자 클러스터의 **워크로드 네트워크**에서 필요한 noProxy 값을 가져옵니다.


compute-cluster
ACTIONS
Summary
Monitor
Configure
Permissions
Hosts
VMs
Namespaces
Datastores
Netwo
Services
vSphere DRS
vSphere Availabili...
Configuration
Quickstart
General
Key Provider
VMware EVC
VM/Host Groups
VM/Host Rules
VM Overrides
I/O Filters
Host Options
Host Profile
Licensing
vSAN Cluster
Supervisor Cluster
Trust Authority
Alarm Definitions
Scheduled Tasks
Namespaces
General
Network
Storage
Certificates
Image Registry
vSAN
Services
Disk Management
Fault Domains
Datastore Sharing
Supervisor Ser...
Network
Below are the network settings for supporting namespaces on this cluster.
Management Network
Workload Network
vSphere Distributed Switch
wcp_vds_1
Edge Cluster
EDGECLUSTER1
DNS Servers
10.20.145.1
EDIT
Pod CIDRs
10.246.0.0/16
EDIT
Services CIDR
10.94.0.0/12
Ingress CIDRs
192.168.144.0/20
EDIT
Egress CIDRs
192.168.128.0/20
EDIT

Tanzu Kubernetes Grid 서비스 v1alpha1 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하는 예

Tanzu Kubernetes Grid 서비스 API는 Tanzu Kubernetes 클러스터를 사용자 지정하기 지능형 기본값과 다양한 옵션을 제공합니다. 요구 사항을 충족하는 다양한 구성 및 사용자 지정으로 다양한 유형의 클러스터를 프로비저닝하려면 다음 예를 참조하십시오.

Tanzu Kubernetes 클러스터 프로비저닝을 위한 최소 YAML

다음 예제 YAML은 Tanzu Kubernetes Grid 서비스를 호출하고 모든 기본 설정을 사용하는 Tanzu Kubernetes 클러스터를 프로비저닝하는 데 필요한 최소 구성입니다.

최소 예제 YAML의 특징은 다음과 같습니다.

- v1.19로 나열된 Tanzu Kubernetes 릴리스 버전은 해당 부 버전(예: `v1.19.7+vmware.1-tkg.1.xxxxxx`)과 일치하는 최신 배포판으로 확인됩니다. 업데이트를 위한 Tanzu Kubernetes 클러스터 호환성 확인의 내용을 참조하십시오.
- VM 클래스 `best-effort-<size>`에 예약이 없습니다. 자세한 내용은 Tanzu Kubernetes 클러스터에 대한 가상 시스템 클래스의 내용을 참조하십시오.
- 클러스터에는 컨테이너용 영구 스토리지가 포함되어 있지 않습니다. 필요한 경우 `spec.settings.storage`에서 설정합니다. 아래의 스토리지 예제를 참조하십시오.
- Helm과 같은 일부 워크로드에는 `spec.settings.storage.defaultClass`가 필요할 수 있습니다. 아래의 스토리지 예제를 참조하십시오.
- `spec.settings.network` 섹션이 지정되지 않았습니다. 즉, 클러스터에 다음과 같은 기본 네트워크 설정이 사용됩니다.
  - 기본 CNI: `antrea`
  - 기본 포드 CIDR: `192.168.0.0/16`
  - 기본 서비스 CIDR: `10.96.0.0/12`
  - 기본 서비스 도메인: `cluster.local`

  참고 포드의 기본 IP 범위는 `192.168.0.0/16`입니다. 이 서브넷을 이미 사용 중이면 다른 CIDR 범위를 지정해야 합니다. 아래의 사용자 지정 네트워크 예를 참조하십시오.

```
apiVersion: run.tanzu.vmware.com/v1alpha1      #TKGS API endpoint
kind: TanzuKubernetesCluster                   #required parameter
metadata:
  name: tkgs-cluster-1                         #cluster name, user defined
  namespace: tgks-cluster-ns                   #vsphere namespace
spec:
  distribution:
    version: v1.20                             #Resolves to latest TKR 1.20
  topology:
    controlPlane:
```

```
      count: 1                                  #number of control plane nodes
      class: best-effort-medium                 #vmclass for control plane nodes
      storageClass: vwt-storage-policy          #storageclass for control plane
    workers:
      count: 3                                  #number of worker nodes
      class: best-effort-medium                 #vmclass for worker nodes
      storageClass: vwt-storage-policy          #storageclass for worker nodes
```

별도의 디스크 및 스토리지 매개 변수가 있는 클러스터

다음 예제 YAML은 클러스터 제어부 및 작업자 노드에 대해 별도의 디스크 및 스토리지 매개 변수를 사용하여 클러스터를 프로비저닝하는 방법을 보여줍니다.

변동률이 높은 데이터에 대해 분리 디스크 및 스토리지 매개 변수를 사용하면 무엇보다 연결된 클론의 사용과 관련된 읽기-쓰기 오버헤드를 최소화하는 데 도움이 됩니다. 두 가지 주요 사용 사례는 다음과 같습니다.

- etcd 데이터베이스에 대한 제어부 노드의 스토리지 성능을 사용자 지정
- 작업자 노드의 컨테이너 이미지에 대한 디스크 크기를 사용자 지정

이 예에는 다음과 같은 특징이 있습니다.

- `spec.topology.controlPlane.volumes` 설정은 etcd 데이터베이스에 대해 별도의 볼륨을 지정합니다.
- `spec.topology.workers.volumes` 설정은 컨테이너 이미지용에 대해 별도의 볼륨을 지정합니다.
- 컨테이너 이미지에 대한 `mountPath: /var/lib/containerd`는 Tanzu Kubernetes 릴리스 1.17 이상에서 지원됩니다.

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TanzuKubernetesCluster
metadata:
  name: tkgs-cluster-5
  namespace: tgks-cluster-ns
spec:
  distribution:
    version: v1.20
  topology:
    controlPlane:
      count: 3
      class: best-effort-medium
      storageClass: vwt-storage-policy
      volumes:
        - name: etcd
          mountPath: /var/lib/etcd
          capacity:
            storage: 4Gi
    workers:
      count: 3
      class: best-effort-medium
      storageClass: vwt-storage-policy
```

```
      volumes:
        - name: containerd
          mountPath: /var/lib/containerd
          capacity:
            storage: 16Gi
```

사용자 지정 Antrea 네트워크가 있는 클러스터

다음 YAML은 Antrea CNI에 대한 사용자 지정 네트워크 범위를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하는 방법을 보여줍니다.

- 사용자 지정 네트워크 설정이 적용되기 때문에 기본 Antrea CNI를 사용하는 경우에도 `cni.name` 매개 변수가 필요합니다.
  - CNI 이름: `antrea`
  - 사용자 지정 포드 CIDR: `193.0.2.0/16`
  - 사용자 지정 서비스 CIDR: `195.51.100.0/12`
  - 사용자 지정 서비스 도메인: `managedcluster.local`
- 사용자 지정 CIDR 블록은 감독자 클러스터와 겹칠 수 없습니다. 자세한 내용은 Tanzu Kubernetes Grid 서비스 v1alpha1 API를 사용하는 Tanzu Kubernetes 클러스터에 대한 구성 매개 변수의 내용을 참조하십시오.

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TanzuKubernetesCluster
metadata:
  name: tkg-cluster-3-antrea
  namespace: tkgs-cluster-ns
spec:
  distribution:
    version: v1.20
  topology:
    controlPlane:
      class: guaranteed-medium
      count: 3
      storageClass: vwt-storage-policy
    workers:
      class: guaranteed-medium
      count: 5
      storageClass: vwt-storage-policy
  settings:
    network:
      cni:
        name: antrea                        #Use Antrea CNI
      pods:
        cidrBlocks:
        - 193.0.2.0/16                      #Must not overlap with SVC
```

```
    services:
      cidrBlocks:
      - 195.51.100.0/12                  #Must not overlap with SVC
    serviceDomain: managedcluster.local
```

사용자 지정 Calico 네트워크가 있는 클러스터

다음 YAML은 사용자 지정 Calico 네트워크를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하는 방법을 보여줍니다.

- Calico는 기본 CNI가 아니므로 매니페스트에서 명시적으로 이름이 지정됩니다. 서비스 수준에서 기본 CNI를 변경하려면 Tanzu Kubernetes Grid 서비스 v1alpha1 API 구성 예시 항목을 참조하십시오.
  - CNI 이름: `calico`
  - 사용자 지정 포드 CIDR: `198.51.100.0/12`
  - 사용자 지정 서비스 CIDR: `192.0.2.0/16`
  - 사용자 지정 서비스 도메인: `managedcluster.local`
- 네트워크 설정은 기본값이 아닌 사용자 지정 CIDR 범위를 사용합니다. 이러한 범위는 감독자 클러스터와 겹치지 않아야 합니다. Tanzu Kubernetes Grid 서비스 v1alpha1 API를 사용하는 Tanzu Kubernetes 클러스터에 대한 구성 매개 변수의 내용을 참조하십시오.

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TanzuKubernetesCluster
metadata:
  name: tkgs-cluster-2
  namespace: tkgs-cluster-ns
spec:
  distribution:
    version: v1.20
  topology:
    controlPlane:
      count: 3
      class: guaranteed-large
      storageClass: vwt-storage-policy
    workers:
      count: 5
      class: guaranteed-xlarge
      storageClass: vwt-storage-policy
  settings:
    network:
      cni:
        name: calico                          #Use Calico CNI for this cluster
      services:
        cidrBlocks: ["198.51.100.0/12"]       #Must not overlap with SVC
      pods:
        cidrBlocks: ["192.0.2.0/16"]          #Must not overlap with SVC
      serviceDomain: managedcluster.local
```

스토리지 클래스및 영구 볼륨에 대한 기본 클래스가 있는 클러스터

다음 YAML 예에서는 동적 PVC 프로비저닝을 위한 스토리지 클래스 및 기본 스토리지 클래스로 클러스터를 프로비저닝하는 방법을 보여줍니다.

- `spec.settings.storage.classes` 설정은 클러스터의 컨테이너용 영구 스토리지에 대해 두 개의 스토리지 클래스를 지정합니다.
- `spec.settings.storage.defaultClass`가 지정됩니다. 일부 애플리케이션에는 기본 클래스가 필요합니다. 예를 들어 Helm 또는 Kubeapps를 여러 차트에서 참조되는 `defaultClass`로 사용하려는 경우입니다.

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TanzuKubernetesCluster
metadata:
  name: default-storage-spec
  namespace: tkgs-cluster-ns
spec:
  topology:
    controlPlane:
      count: 3
      class: best-effort-medium
      storageClass: vwt-storage-policy
    workers:
      count: 3
      class: best-effort-medium
      storageClass: vwt-storage-policy
  distribution:
    version: v1.20
  settings:
    network:
      cni:
        name: antrea
      services:
        cidrBlocks: ["198.51.100.0/12"]
      pods:
        cidrBlocks: ["192.0.2.0/16"]
      serviceDomain: "tanzukubernetescluster.local"
    storage:
      classes: ["gold", "silver"]              #Array of named PVC storage classes
      defaultClass: silver                     #Default PVC storage class
```

프록시 서버가 있는 클러스터

프록시 서버 구성을 클러스터 매니페스트에 적용하여 개별 Tanzu Kubernetes 클러스터에서 프록시 서버를 사용할 수 있습니다.

다음과 같은 특징을 참고하십시오.

- `spec.settings.network.proxy` 섹션은 Tanzu Kubernetes 클러스터에 대한 HTTP(s) 프록시 구성을 지정합니다.

- 두 proxy 서버 값의 구문은 http://<user>:<pwd>@<ip>:<port>입니다.
- localhost 및 127.0.0.1과 Tanzu Kubernetes 클러스터용 포드 및 서비스CIDR을 비롯한 특정 끝점은 자동으로 프록시되지 않습니다. 이러한 항목은 noProxy 필드에 포함할 필요가 없습니다.
- noProxy 프록시를 사용하지 않을 CIDR 어레이를 허용합니다. 감독자 클러스터의 워크로드 네트워크에서 필요한 값을 가져옵니다. Tanzu Kubernetes Grid 서비스 v1alpha1 API를 사용하는 Tanzu Kubernetes 클러스터에 대한 구성 매개 변수에서 이미지를 참조하십시오.
- TkgServiceConfiguration에 글로벌 프록시가 구성된 경우 이 프록시 정보는 클러스터의 초기 배포 후 클러스터 매니페스트로 전파됩니다. 글로벌 프록시 구성은 클러스터를 생성할 때 프록시 구성 필드가 없는 경우에만 클러스터 매니페스트에 추가됩니다. 다시 말해, 클러스터별 구성이 우선하며 글로벌 프록시 구성을 덮어씁니다.

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TanzuKubernetesCluster
metadata:
  name: tkgs-cluster-with-proxy
  namespace: tkgs-cluster-ns
spec:
  distribution:
    version: v1.20
  topology:
    controlPlane:
      count: 3
      class: guaranteed-medium
      storageClass: vwt-storage-policy
    workers:
      count: 5
      class: guaranteed-xlarge
      storageClass: vwt-storage-policy
  settings:
    storage:
      classes: ["gold", "silver"]
      defaultClass: silver
    network:
      cni:
        name: antrea
      pods:
        cidrBlocks:
        - 193.0.2.0/16
      services:
        cidrBlocks:
        - 195.51.100.0/12
      serviceDomain: managedcluster.local
      proxy:
        httpProxy: http://10.186.102.224:3128  #Proxy URL for HTTP connections
        httpsProxy: http://10.186.102.224:3128 #Proxy URL for HTTPS connections
        noProxy: [10.246.0.0/16,192.168.144.0/20,192.168.128.0/20] #SVC Pod, Egress, Ingress
CIDRs
```

TLS용 사용자 지정 인증서가 있는 클러스터

TkgServiceConfiguration에서 trust.additionalTrustedCAs를 지정하는 방식과 유사하게(Tanzu Kubernetes Grid 서비스 v1alpha1 API에 대한 구성 매개 변수 참조) TanzuKubernetesCluster 규격의 spec.settings.network 아래 trust.additionalTrustedCAs를 포함할 수 있습니다. 예:

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TanzuKubernetesCluster
metadata:
  name: tkgs-cluster-with-custom-certs-tls
  namespace: tkgs-cluster-ns
spec:
  topology:
    controlPlane:
      count: 3
      class: guaranteed-medium
      storageClass: vwt-storage-profile
    workers:
      count: 3
      class: guaranteed-large
      storageClass: vwt-storage-profile
  distribution:
    version: 1.20.2
  settings:
    network:
      cni:
        name: antrea
      services:
        cidrBlocks: ["198.51.100.0/12"]
      pods:
        cidrBlocks: ["192.0.2.0/16"]
      serviceDomain: "managedcluster.local"
      trust:
        additionalTrustedCAs:
          - name: custom-selfsigned-cert-for-tkc
            data: |
              LS0aaaaaaaaaaaaaaabase64...
```

TkgServiceConfiguration 규격에서 글로벌 설정을 상속하거나 상속하지 않는 클러스터

참고 다음 예제 클러스터 구성에는 vCenter Server 버전 7.0U2a 이상 및 감독자 클러스터 버전1.18.10 이상이 필요합니다.

TkgServiceConfiguration에서 전역 설정을 상속하는 Tanzu Kubernetes 클러스터를 프로비저닝하려면 전역 설정이 지정되지 않았거나 null인 클러스터를 구성합니다.

예를 들어 proxy 설정을 상속하는 클러스터를 구성하려면 다음 방식 중 하나를 사용할 수 있습니다.

옵션1: 클러스터 규격에 proxy 설정을 포함하지 마십시오.

```
...
settings:
  network:
    cni:
      name: antrea
    services:
      cidrBlocks: ["198.51.100.0/12"]
    pods:
      cidrBlocks: ["192.0.2.0/16"]
    serviceDomain: "tanzukubernetescluster.local"
```

옵션2: 규격에 proxy 설정을 포함하되 해당 값을 명시적으로 null로 설정합니다.

```
settings:
  network:
    proxy: null
```

TkgServiceConfiguration에서 기본값을 상속하지 않는 Tanzu Kubernetes 클러스터를 프로비저닝하려면 모든 요소가 포함되지만 값이 비어 있는 클러스터 규격을 구성합니다.

예를 들어 TkgServiceConfiguration에 글로벌 proxy가 구성되어 있고 글로벌 proxy 설정을 상속하지 않는 클러스터를 프로비저닝하려면 클러스터 규격에 다음 구문을 포함합니다.

```
...
settings:
  network:
    proxy:
      httpProxy: ""
      httpsProxy: ""
      noProxy: null
```

로컬 컨텐츠 라이브러리를 사용하는 클러스터

에어갭 환경에서 Tanzu Kubernetes 클러스터를 프로비저닝하려면 로컬 컨텐츠 라이브러리에서 동기화된 가상 시스템 이미지를 사용하여 클러스터를 생성합니다.

로컬 컨텐츠 라이브러리 이미지를 사용하여 클러스터를 프로비저닝 하려면 클러스터 사양에 해당 이미지를 지정해야 합니다. distribution.version 값의 경우, 전체 이미지 이름을 입력하거나 이미지 디렉토리의 이름 형식을 유지한 경우 Kubernetes 버전으로 단축할 수 있습니다. 정규화된 버전 번호를 사용하려면 -----를 +로 바꿉니다. 예를 들어 이름이 photon-3-k8s-v1.20.2---vmware.1-tkg.1.1d4f79a인 OVA 이미지가 있으면 다음 형식을 사용할 수 있습니다.

```
spec:
  distribution:
    version: v1.20
```

```
spec:
  distribution:
    version: v1.20.2
```

```
spec:
  distribution:
    version: v1.20.2+vmware.1-tkg.1
```

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TanzuKubernetesCluster
metadata:
    name: tgks-cluster-9
    namespace: tkgs-cluster-ns
spec:
   topology:
       controlPlane:
           count: 3
           class: best-effort-medium
           storageClass: vwt-storage-policy
       workers:
           count: 3
           class: best-effort-medium
           storageClass: vwt-storage-policy
   distribution:
        version: v1.20.2
   settings:
         network:
           cni:
               name: antrea
           services:
              cidrBlocks: ["198.51.100.0/12"]
           pods:
              cidrBlocks: ["192.0.2.0/16"]
```

Tanzu Kubernetes Grid 서비스 v1alpha1 API에 대한 구성 매개 변수

CNI(Container Network Interface), 프록시 서버 및 TLS 인증서를 비롯한 주요 기능에 대한 글로벌 설정을 사용하여 Tanzu Kubernetes Grid 서비스를 사용자 지정할 수 있습니다. 글로벌 기능을 구현하는 것과 클러스터당 기능을 구현하는 것의 장단점 및 고려 사항을 알고 있어야 합니다.

필요한 경우 글로벌 매개 변수를 사용하여 Tanzu Kubernetes Grid 서비스를 구성할 수 있습니다.

경고 Tanzu Kubernetes Grid 서비스 구성은 글로벌 작업입니다. 즉, `TkgServiceConfiguration` 규격을 변경하면 해당 서비스에서 프로비저닝된 모든 Tanzu Kubernetes 클러스터에 관련 변경 내용이 적용됩니다. 롤링 업데이트가 시작되면(수동으로 또는 업그레이드를 통해) 변경된 서비스 규격에 따라 클러스터가 업데이트됩니다.

TkgServiceConfiguration 규격

`TkgServiceConfiguration` 규격은 Tanzu Kubernetes Grid 서비스 인스턴스 구성을 위한 필드를 제공합니다.

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TkgServiceConfiguration
metadata:
  name: tkg-service-configuration-example
spec:
  defaultCNI: <antrea or calico>
  proxy:
    httpProxy: http://<user>:<pwd>@<ip>:<port>
    httpsProxy: http://<user>:<pwd>@<ip>:<port>
    noProxy: [<array of CIDRs to not proxy>]
  trust:
    additionalTrustedCAs:
      - name: <first-cert-name>
        data: <base64-encoded string of a PEM encoded public cert 1>
      - name: <second-cert-name>
        data: <base64-encoded string of a PEM encoded public cert 2>
```

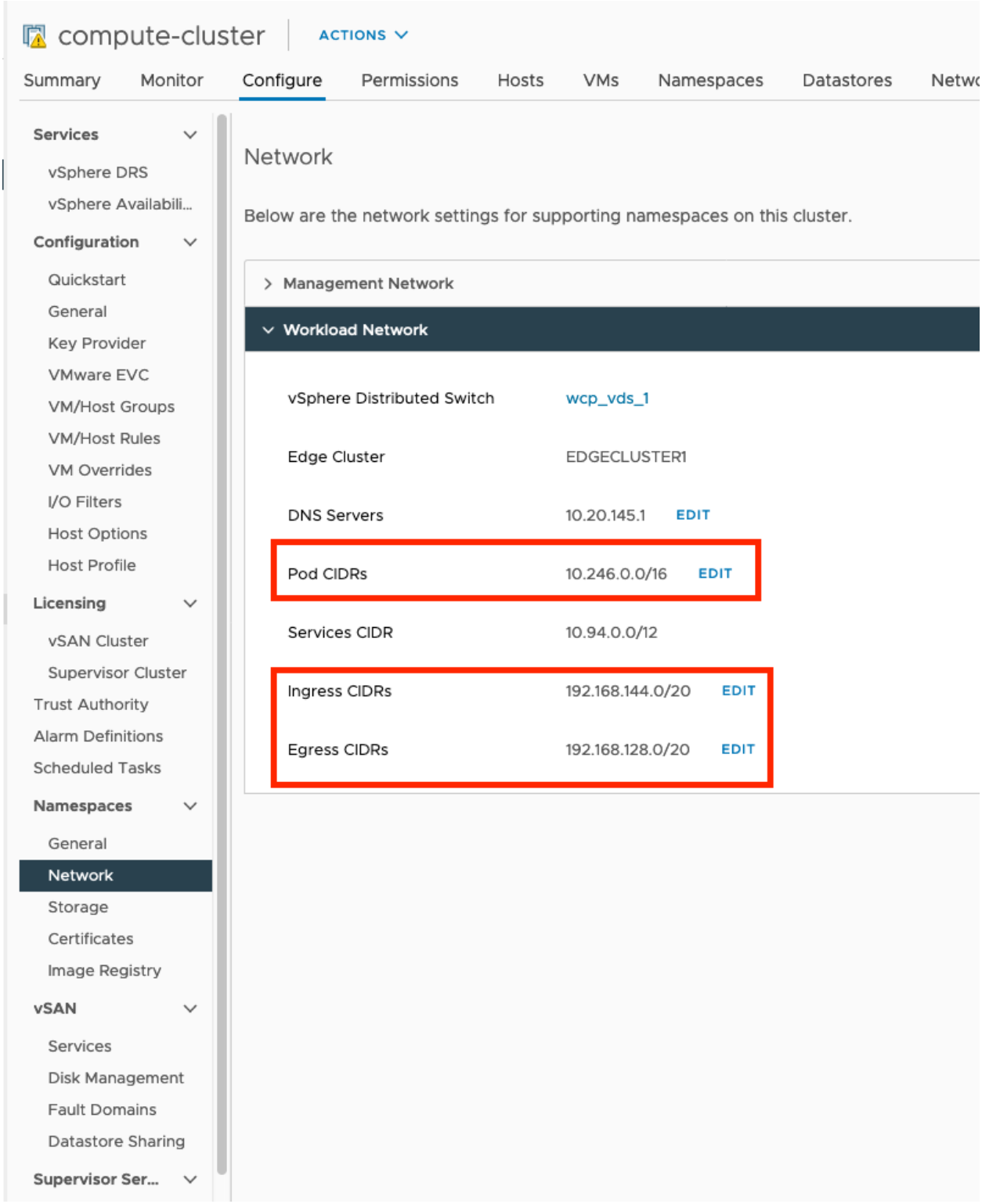
TkgServiceConfiguration 규격 매개 변수

이 표에는 `TkgServiceConfiguration` 규격 매개 변수와 그에 대한 설명이 있습니다. 예제에 대해서는 Tanzu Kubernetes Grid 서비스 v1alpha1 API 구성 예시 항목을 참조하십시오.

필드	값	설명
`defaultCNI`	`antrea or calico`	클러스터에서 사용할 기본 CNI입니다. 기본값은 `antrea`입니다. 지원되는 다른 CNI는 `calico`입니다.
`proxy`	`proxy` 매개 변수의 섹션 마커입니다.	`proxy` 매개 변수는 `httpProxy`, `httpsProxy`, `noProxy`입니다. 모든 매개 변수는 필수입니다. `proxy` 매개 변수가 누락된 경우에는 Tanzu Kubernetes 클러스터를 생성할 수 없습니다.

필드	값	설명
httpProxy	http://<user>:<pwd>@<ip>:<port> 형식의 URI	https 프로토콜을 허용하지 않습니다. https를 사용하는 경우 Tanzu Kubernetes 클러스터를 생성할 수 없습니다.
httpsProxy	http://<user>:<pwd>@<ip>:<port> 형식의 URI	https 프로토콜을 허용하지 않습니다. https를 사용하는 경우 Tanzu Kubernetes 클러스터를 생성할 수 없습니다.
noProxy	프록시를 사용하지 않을 CIDR 블록의 어레이입니다(예: [10.246.0.0/16,192.168.144.0/20,192.168.128.0/20]). 감독자 클러스터의 워크로드 네트워크에서 필요한 값을 가져옵니다(예: Pod CIDRs, Ingress CIDRs 및 Egress CIDRs). noProxy 어레이 필드에 포함할 값은 아래 이미지를 참조하십시오.	포드, 수신 및 송신을 위해 감독자 클러스터의 워크로드 네트워크에서 사용하는 서브넷에는 프록시를 사용하지 말아야 합니다. 감독자 클러스터의 서비스 CIDR을 noProxy 필드에 포함할 필요가 없습니다. Tanzu Kubernetes 클러스터는 그러한 서비스와 상호 작용하지 않습니다. 끝점 localhost 및 127.0.0.1은 자동으로 프록시로 사용되지 않습니다. noProxy 필드에 추가할 필요가 없습니다. Tanzu Kubernetes 클러스터의 포드 및 서비스 CIDR은 자동으로 프록시로 사용되지 않습니다. noProxy 필드에 추가할 필요가 없습니다.
trust	trust 매개 변수의 섹션 마커입니다.	데이터를 허용하지 않습니다.
additionalTrustedCAs	각각 name 및 data가 있는 인증서 어레이를 허용합니다.	데이터를 허용하지 않습니다.
name	문자열	TLS 인증서의 이름입니다.
data	문자열	PEM으로 인코딩된 공용 인증서의 base64로 인코딩된 문자열입니다.

이미지에 표시된 대로 감독자 클러스터의 **워크로드 네트워크**에서 필요한 noProxy 값을 가져옵니다.



글로벌 또는 클러스터별 구성 옵션을 사용하는 경우

`TkgServiceConfiguration`은 Tanzu Kubernetes Grid 서비스 인스턴스로 프로비저닝된 모든 Tanzu Kubernetes 클러스터에 영향을 미치는 글로벌 규격입니다.

TkgServiceConfiguration 규격을 편집하기 전에, 글로벌 구성 대신 사용 사례를 충족할 수 있는 클러스터별 대안을 알고 있어야 합니다.

표 13-5. 글로벌 및 클러스터별 구성 옵션

설정	글로벌 옵션	클러스터별 옵션
기본 CNI	TkgServiceConfiguration 규격을 편집합니다. Tanzu Kubernetes Grid 서비스 v1alpha1 API 구성 예시의 내용을 참조하십시오.	클러스터 규격에서 CNI를 지정합니다. 예를 들어 기본 CNI는 Antrea입니다. Calico를 사용하려면 클러스터 YAML에서 지정합니다. Tanzu Kubernetes Grid 서비스 v1alpha1 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하는 예 항목을 참조하십시오.
프록시 서버	TkgServiceConfiguration 규격을 편집합니다. Tanzu Kubernetes Grid 서비스 v1alpha1 API 구성 예시의 내용을 참조하십시오.	클러스터 규격에 프록시 서버 구성 매개변수를 포함합니다. Tanzu Kubernetes Grid 서비스 v1alpha1 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하는 예의 내용을 참조하십시오.
트러스트 인증서	TkgServiceConfiguration 규격을 편집합니다. 외부 컨테이너 레지스트리 구성과 인증서 기반 프록시 구성이라는 두 가지 사용 사례가 있습니다. Tanzu Kubernetes Grid 서비스 v1alpha1 API 구성 예시 항목을 참조하십시오.	예, 클러스터별로 사용자 지정 인증서를 포함하거나 클러스터 규격에서 전체적으로 설정된 trust 설정을 재정의할 수 있습니다. Tanzu Kubernetes Grid 서비스 v1alpha1 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하는 예의 내용을 참조하십시오.

참고 TkgServiceConfiguration에 글로벌 프록시가 구성된 경우 이 프록시 정보는 클러스터의 초기 배포 후 클러스터 매니페스트로 전파됩니다. 글로벌 프록시 구성은 클러스터를 생성할 때 프록시 구성 필드가 없는 경우에만 클러스터 매니페스트에 추가됩니다. 다시 말해, 클러스터별 구성이 우선하며 글로벌 프록시 구성을 덮어씁니다. 자세한 내용은 Tanzu Kubernetes Grid 서비스 v1alpha1 API에 대한 구성 매개변수의 내용을 참조하십시오.

TkgServiceConfiguration 규격을 편집하기 전에 글로벌 수준에서 설정을 적용하는 결과에 대해 알고 있어야 합니다.

필드	적용	추가/변경된 경우 기존 클러스터에 미치는 영향	클러스터 생성 시 클러스터별 재정의	클러스터 업데이트 시 클러스터별 재정의
defaultCNI	전체적으로	없음	예, 클러스터 생성 시 글로벌 설정을 재정의할 수 있습니다.	아니요, 기존 클러스터의 CNI를 변경할 수 없습니다. 클러스터 생성 시 전체적으로 설정된 기본 CNI를 사용한 경우에는 변경할 수 없습니다.
proxy	전체적으로	없음	예, 클러스터 생성 시 글로벌 설정을 재정의할 수 있습니다.	예, U2+를 사용하면 클러스터 업데이트 시 글로벌 설정을 재정의할 수 있습니다.
trust	전체적으로	없음	예, 클러스터 생성 시 글로벌 설정을 재정의할 수 있습니다.	예, U2+를 사용하면 클러스터 업데이트 시 글로벌 설정을 재정의할 수 있습니다.

글로벌 구성 변경을 기존 클러스터에 전파

TkgServiceConfiguration의 글로벌 수준에서 지정한 설정은 기존 클러스터에 자동으로 전파되지 않습니다. 예를 들어 TkgServiceConfiguration에서 proxy 또는 trust 설정을 변경하는 경우 이미 프로비저닝된 클러스터에는 이러한 변경 사항이 영향을 주지 않습니다.

기존 클러스터에 글로벌 변경 내용을 전파하려면 Tanzu Kubernetes 클러스터에 패치를 적용하여 TkgServiceConfiguration에 대한 변경 내용을 클러스터가 상속하도록 해야 합니다.

예:

```
kubectl patch tkc <CLUSTER_NAME> -n <NAMESPACE> --type merge -p "{\"spec\":{\"settings\":
{\"network\":{\"proxy\": null}}}}"
```

```
kubectl patch tkc <CLUSTER_NAME> -n <NAMESPACE> --type merge -p "{\"spec\":{\"settings\":
{\"network\":{\"trust\": null}}}}"
```

Tanzu Kubernetes Grid 서비스 v1alpha1 API 구성 예시

컨테이너 네트워크 인터페이스, 프록시 서버 및 TLS 인증서에 대한 글로벌 구성 설정을 사용하여 Tanzu Kubernetes Grid 서비스를 사용자 지정하려면 예제를 참조하십시오.

Tanzu Kubernetes Grid 서비스 구성 정보

기본 CNI를 변경하고 글로벌 프록시 서버를 추가하고 신뢰할 수 있는 인증서를 추가하여 Tanzu Kubernetes Grid 서비스를 사용자 지정할 수 있습니다. Tanzu Kubernetes Grid 서비스 v1alpha1 API에 대한 구성 매개 변수의 내용을 참조하십시오.

경고 Tanzu Kubernetes Grid 서비스 규격을 편집하면 해당 서비스에서 프로비저닝된 모든 클러스터가 전체적으로 변경되며, 여기에는 수동으로 또는 자동으로 업그레이드되는 기존 클러스터 및 새 클러스터가 포함됩니다.

사전 요구 사항: Kubectl 편집 구성

Tanzu Kubernetes 클러스터 크기를 조정하려면 `kubectl edit tanzukubernetescluster/CLUSTER-NAME` 명령을 사용하여 클러스터 매니페스트를 업데이트합니다. kubectl edit 명령은 KUBE_EDITOR 또는 EDITOR 환경 변수로 정의된 텍스트 편집기에서 클러스터 매니페스트를 엽니다. 환경 변수 설정에 대한 지침은 kubectl용 기본 텍스트 편집기 지정 항목을 참조하십시오.

규격 변경 내용을 저장하면 `kubectl`은 편집 내용이 기록되었다고 보고합니다. 취소하려면 저장하지 않고 편집기를 닫기만 하면 됩니다.

기본 CNI 구성

Tanzu Kubernetes Grid 서비스는 Tanzu Kubernetes 클러스터에 대한 기본 CNI(Container Network Interface)를 제공합니다. 기본 구성을 사용하면 CNI를 지정하지 않고도 클러스터를 생성할 수 있습니다. 서비스 규격을 편집하여 기본 CNI를 변경할 수 있습니다.

Tanzu Kubernetes Grid 서비스는 두 가지(Antrea 및 Calico) CNI를 지원하며 Antrea가 기본값입니다. 자세한 내용은 Tanzu Kubernetes Grid 서비스 클러스터 네트워킹 의 내용을 참조하십시오.

사용할 CNI를 명시적으로 지정하여 기본 CNI를 재정의할 수 있습니다. 또는 CNI용 TKG 서비스 컨트롤러를 편집하여 기본 CNI를 변경할 수 있습니다.

1 감독자 클러스터로 인증합니다.

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

2 컨텍스트를 대상 vSphere 네임스페이스로 전환합니다.

```
kubectl config use-context tkgs-cluster-ns
```

3 기본 CNI를 나열합니다.

```
kubectl get tkgserviceconfigurations
```

예제 결과:

```
NAME                                DEFAULT CNI
tkg-service-configuration           antrea
```

4 Tanzu Kubernetes Grid 서비스 규격을 편집하기 위해 로드합니다.

```
kubectl edit tkgserviceconfigurations tkg-service-configuration
```

KUBE_EDITOR 또는 EDITOR 환경 변수로 정의된 기본 텍스트 편집기에서 `tkg-service-configuration` 규격이 열립니다.

5 `spec.defaultCNI` 값을 편집합니다.

예를 들어, 변경 전:

```
spec:
  defaultCNI: antrea
```

변경 후:

```
spec:
  defaultCNI: calico
```

6 변경 내용을 적용하려면 텍스트 편집기에서 파일을 저장합니다. 취소하려면 저장하지 않고 편집기를 닫습니다.

텍스트 편집기에서 변경 내용을 저장하면 kubectl은 `tkg-service-configuration` 서비스 규격을 업데이트합니다.

7 기본 CNI가 업데이트되었는지 확인합니다.

```
kubectl get tkgserviceconfigurations
```

기본 CNI가 업데이트됩니다. 기본 네트워크 설정으로 프로비저닝된 모든 클러스터는 기본 CNI를 사용합니다.

```
NAME                                DEFAULT CNI
tkg-service-configuration           calico
```

글로벌 프록시 서버 구성

글로벌 프록시 서버를 사용하도록 설정하려면 프록시 서버 매개 변수를 `TkgServiceConfiguration`에 추가합니다. 필수 필드에 대한 설명은 Tanzu Kubernetes Grid 서비스 v1alpha1 API에 대한 구성 매개 변수 항목을 참조하십시오.

1 감독자 클러스터로 인증합니다.

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

2 컨텍스트를 대상 vSphere 네임스페이스로 전환합니다.

```
kubectl config use-context tkgs-cluster-ns
```

3 현재 구성을 가져옵니다.

```
kubectl get tkgserviceconfigurations
```

예제 결과:

```
NAME                          DEFAULT CNI
tkg-service-configuration     antrea
```

4 Tanzu Kubernetes Grid 서비스 규격을 편집하기 위해 로드합니다.

```
kubectl edit tkgserviceconfigurations tkg-service-configuration
```

KUBE_EDITOR 또는 EDITOR 환경 변수로 정의된 기본 텍스트 편집기에서 `tkg-service-configuration` 규격이 열립니다.

5 `httpProxy`, `httpsProxy`, `noProxy`를 포함하여 각 필수 필드에 `spec.proxy` 하위 섹션을 추가합니다.

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TkgServiceConfiguration
metadata:
  ...
  name: tkg-service-configuration-example
  resourceVersion: "44170525"
  selfLink: /apis/run.tanzu.vmware.com/v1alpha1/tkgserviceconfigurations/tkg-service-
configuration
  uid: 10347195-5f0f-490e-8ae1-a758a724c0bc
spec:
  defaultCNI: antrea
  proxy:
    httpProxy: http://<user>:<pwd>@<ip>:<port>
    httpsProxy: http://<user>:<pwd>@<ip>:<port>
    noProxy: [SVC-POD-CIDRs, SVC-EGRESS-CIDRs, SVC-INGRESS-CIDRs]
```

6 각 프록시 필드를 적절한 값으로 채웁니다. 각 필드에 대한 설명은 Tanzu Kubernetes Grid 서비스 v1alpha1 API에 대한 구성 매개 변수 항목을 참조하십시오.

`noProxy` 필드에 필요한 값은 감독자 클러스터의 **워크로드 네트워크**에서 가져옵니다. 이러한 값을 얻을 수 있는 위치는 위 항목의 그림을 참조하십시오.

예:

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TkgServiceConfiguration
metadata:
  ...
  name: tkg-service-configuration-example
  resourceVersion: "44170525"
  selfLink: /apis/run.tanzu.vmware.com/v1alpha1/tkgserviceconfigurations/tkg-service-
configuration
```

```
  uid: 10347195-5f0f-490e-8ae1-a758a724c0bc
spec:
  defaultCNI: antrea
  proxy:
    httpProxy: http://user:password@10.186.102.224:3128
    httpsProxy: http://user:password@10.186.102.224:3128
    noProxy: [10.246.0.0/16,192.168.144.0/20,192.168.128.0/20]
```

7 변경 내용을 적용하려면 텍스트 편집기에서 파일을 저장합니다. 취소하려면 저장하지 않고 편집기를 닫습니다.

텍스트 편집기에서 변경 내용을 저장하면 kubectl은 `tkg-service-configuration` 서비스 규격에 정의된 구성으로 Tanzu Kubernetes Grid 서비스를 업데이트합니다.

8 Tanzu Kubernetes Grid 서비스가 프록시 설정으로 업데이트되었는지 확인합니다.

```
kubectl get tkgserviceconfigurations -o yaml
```

9 확인하려면 Tanzu Kubernetes 클러스터를 프로비저닝합니다. TKGS v1alpha2 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하는 워크플로의 내용을 참조하십시오.

다음 명령을 사용하여 클러스터가 프록시를 사용하고 있는지 확인합니다.

```
kubectl get tkc CLUSTER-NAME -n NAMESPACE -o yaml
```

인증서 기반 프록시 구성

프록시 서버를 사용하여 인터넷 트래픽을 라우팅하는 것이 일부 환경에서는 엄격한 요구 사항입니다. 예를 들어, 금융 기관과 같이 규제가 엄격한 업계의 회사에서는 모든 인터넷 트래픽이 회사 프록시를 통과해야 합니다.

아웃바운드 HTTP/S 트래픽에 프록시 서버를 사용하는 Tanzu Kubernetes 클러스터를 프로비저닝하도록 Tanzu Kubernetes Grid 서비스를 구성할 수 있습니다. 자세한 내용은 Tanzu Kubernetes Grid 서비스 v1alpha1 API에 대한 구성 매개 변수의 내용을 참조하십시오.

예에서 볼 수 있듯이, 프록시 서버에 대한 신뢰할 수 있는 인증서를 `TkgServiceConfiguration` 규격에 추가할 수 있습니다.

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TkgServiceConfiguration
metadata:
  name: tkg-service-configuration-example
spec:
  defaultCNI: antrea
  proxy:
    httpProxy: http://user:password@10.186.102.224:3128
    httpsProxy: http://user:password@10.186.102.224:3128
    noProxy: [10.246.0.0/16,192.168.144.0/20,192.168.128.0/20]
  trust:
    additionalTrustedCAs:
```

```
      - name: first-cert-name
        data: base64-encoded string of a PEM encoded public cert 1
      - name: second-cert-name
        data: base64-encoded string of a PEM encoded public cert 2
```

외부 개인 레지스트리 구성

Tanzu Kubernetes 클러스터를 외부 개인 레지스트리와 연결하기 위한 사용자 지정 인증서를 사용하여 Tanzu Kubernetes Grid 서비스를 구성할 수 있습니다. 자세한 내용은 Tanzu Kubernetes 클러스터에서 외부 컨테이너 레지스트리 사용의 내용을 참조하십시오.

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TkgServiceConfiguration
metadata:
  name: tkg-service-configuration-example
spec:
  defaultCNI: antrea
  trust:
    additionalTrustedCAs:
      - name: harbor-vm-cert
        data: <<<base64-encoded string of a PEM encoded public cert>>>>
```

Tanzu Kubernetes Grid 서비스 v1alpha1 API를 사용하여 Tanzu Kubernetes 클러스터 확장/축소

노드 수를 변경하여 수평으로 또는 노드를 호스팅하는 가상 시스템 클래스를 변경하여 수직으로 Tanzu Kubernetes 클러스터를 확장/축소할 수 있습니다.

지원되는 확장/축소 작업

다음 표에는 Tanzu Kubernetes 클러스터에 대해 지원되는 확장/축소 작업이 나열되어 있습니다.

표 13-6. Tanzu Kubernetes 클러스터에 대해 지원되는 확장/축소 작업

노드	수평 확장	수평 축소	수직 확장/축소
제어부	예	아니요	예
작업자	예	예	예

다음 고려 사항에 유의하십시오.

- 클러스터 노드를 수직으로 확장/축소하는 동안은 사용 가능한 리소스가 부족하여 노드에서 워크로드가 더 이상 실행되지 못할 수 있습니다. 이러한 이유로 인해 수평으로 확장/축소하는 방식을 선호하기도 합니다.
- VM 클래스는 변경할 수 없습니다. 클러스터에서 사용되는 VM 클래스를 편집한 후 Tanzu Kubernetes 클러스터를 확장하면, 새 클러스터 노드는 업데이트된 클래스 정의를 사용하지만 기존 클러스터 노드는 초기 클래스 정의를 계속 사용하여 불일치가 발생합니다. Tanzu Kubernetes 클러스터에 대한 가상 시스템 클래스의 내용을 참조하십시오.

확장/축소 사전 요구 사항: Kubectl 편집 구성

Tanzu Kubernetes 클러스터 크기를 조정하려면 `kubectl edit tanzukubernetescluster/CLUSTER-NAME` 명령을 사용하여 클러스터 매니페스트를 업데이트합니다. kubectl edit 명령은 KUBE_EDITOR 또는 EDITOR 환경 변수로 정의된 텍스트 편집기에서 클러스터 매니페스트를 엽니다. 환경 변수 설정에 대한 지침은 kubectl용 기본 텍스트 편집기 지정 항목을 참조하십시오.

매니페스트 변경 내용을 저장하면 `kubectl`은 편집 내용이 기록되었다고 보고하고 이 변경 내용으로 클러스터가 업데이트됩니다.

```
kubectl edit tanzukubernetescluster/tkgs-cluster-1
tanzukubernetescluster.run.tanzu.vmware.com/tkgs-cluster-1 edited
```

취소하려면 저장하지 않고 편집기를 닫기만 하면 됩니다.

```
kubectl edit tanzukubernetescluster/tkgs-cluster-1
Edit cancelled, no changes made.
```

제어부 확장

제어부 노드 수를 1에서 3으로 늘려서 Tanzu Kubernetes 클러스터를 확장할 수 있습니다. 제어부 노드의 수는 홀수여야 합니다. 제어부는 축소할 수 없습니다.

1 감독자 클러스터로 인증합니다.

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

2 Tanzu Kubernetes 클러스터를 실행 중인 vSphere 네임스페이스로 컨텍스트를 전환합니다.

```
kubectl config use-context tkgs-cluster-ns
```

3 네임스페이스에서 실행 중인 Kubernetes 클러스터를 나열합니다.

```
kubectl get tanzukubernetescluster -n tkgs-cluster-ns
```

4 대상 클러스터에서 실행 중인 노드 수를 구합니다.

```
kubectl get tanzukubernetescluster tkgs-cluster-1
```

예를 들어 다음 클러스터에는 제어부 노드 1개와 작업자 노드 3개가 있습니다.

```
NAME                CONTROL PLANE   WORKER   DISTRIBUTION                      AGE   PHASE
tkgs-cluster-1      1               3        v1.18.5+vmware.1-tkg.1.886c781    1d    running
```

5 `kubectl edit` 명령을 사용하여 편집할 클러스터 매니페스트를 로드합니다.

```
kubectl edit tanzukubernetescluster/tkgs-cluster-1
```

KUBE_EDITOR 또는 EDITOR 환경 변수로 정의된 텍스트 편집기에서 클러스터 매니페스트가 열립니다.

6 spec.topology.controlPlane.count 매개 변수를 찾아서 노드 수를 1에서 3으로 늘립니다.

```
...
controlPlane:
    count: 1
...
```

```
...
ControlPlane:
    count: 3
...
```

7 변경 내용을 적용하려면 텍스트 편집기에서 파일을 저장합니다. 취소하려면 저장하지 않고 편집기를 닫습니다.

파일을 저장하면 kubectl이 변경 내용을 클러스터에 적용합니다. 백그라운드에서 감독자 클러스터의 가상 시스템 서비스는 새 작업자 노드를 프로비저닝합니다.

8 새 노드가 추가되었는지 확인합니다.

```
kubectl get tanzukubernetescluster tkgs-cluster-1
```

확장된 제어부에 이제 3개의 노드가 있습니다.

```
NAME                CONTROL PLANE   WORKER   DISTRIBUTION                      AGE   PHASE
tkgs-cluster-1      3               3        v1.18.5+vmware.1-tkg.1.886c781   1d    running
```

작업자 노드 확장

kubectl을 사용하여 작업자 노드 수를 늘려서 Tanzu Kubernetes 클러스터를 확장할 수 있습니다.

1 감독자 클러스터로 인증합니다.

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

2 Tanzu Kubernetes 클러스터를 실행 중인 vSphere 네임스페이스로 컨텍스트를 전환합니다.

```
kubectl config use-context tkgs-cluster-ns
```

3 네임스페이스에서 실행 중인 Kubernetes 클러스터를 나열합니다.

```
kubectl get tanzukubernetescluster -n tkgs-cluster-ns
```

4 대상 클러스터에서 실행 중인 노드 수를 구합니다.

```
kubectl get tanzukubernetescluster tkgs-cluster-1
```

예를 들어 다음 클러스터에는 제어부 노드 3개와 작업자 노드 3개가 있습니다.

```
NAME                CONTROL PLANE   WORKER   DISTRIBUTION                      AGE   PHASE
tkgs-cluster-1      3               3        v1.18.5+vmware.1-tkg.1.886c781   1d    running
```

5 `kubectl edit` 명령을 사용하여 편집할 클러스터 매니페스트를 로드합니다.

```
kubectl edit tanzukubernetescluster/tkgs-cluster-1
```

KUBE_EDITOR 또는 EDITOR 환경 변수로 정의된 텍스트 편집기에서 클러스터 매니페스트가 열립니다.

6 `spec.topology.workers.count` 매개 변수를 찾아서 노드 수를 늘립니다.

```
...
workers:
    count: 3
...
```

```
...
workers:
    count: 4
...
```

7 변경 내용을 적용하려면 텍스트 편집기에서 파일을 저장합니다. 취소하려면 저장하지 않고 편집기를 닫습니다.

파일을 저장하면 kubectl이 변경 내용을 클러스터에 적용합니다. 백그라운드에서 감독자 클러스터의 가상 시스템 서비스는 새 작업자 노드를 프로비저닝합니다.

8 새 작업자 노드가 추가되었는지 확인합니다.

```
kubectl get tanzukubernetescluster tkgs-cluster-1
```

확장한 후에는 클러스터에 작업자 노드가 4개 있습니다.

```
NAME                CONTROL PLANE   WORKER   DISTRIBUTION                      AGE   PHASE
tkgs-cluster-1      3               4        v1.18.5+vmware.1-tkg.1.886c781   1d    running
```

작업자 노드 축소

작업자 노드 수를 줄여서 Tanzu Kubernetes 클러스터를 축소할 수 있습니다. 제어부에서는 확장/축소가 지원되지 않습니다.

1 감독자 클러스터로 인증합니다.

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

2 Tanzu Kubernetes 클러스터를 실행 중인 vSphere 네임스페이스로 컨텍스트를 전환합니다.

```
kubectl config use-context tkgs-cluster-ns
```

3 네임스페이스에서 실행 중인 Kubernetes 클러스터를 나열합니다.

```
kubectl get tanzukubernetescluster -n tkgs-cluster-ns
```

4 대상 클러스터에서 실행 중인 노드 수를 구합니다.

```
kubectl get tanzukubernetescluster tkgs-cluster-1
```

예를 들어 다음 클러스터에는 제어부 노드 3개와 작업자 노드 3개가 있습니다.

```
NAME                CONTROL PLANE   WORKER   DISTRIBUTION                      AGE   PHASE
tkgs-cluster-1      3               4        v1.18.5+vmware.1-tkg.1.886c781   1d    running
```

5 `kubectl edit` 명령을 사용하여 편집할 클러스터 매니페스트를 로드합니다.

```
kubectl edit tanzukubernetescluster/tkgs-cluster-1
```

KUBE_EDITOR 또는 EDITOR 환경 변수로 정의된 텍스트 편집기에서 클러스터 매니페스트가 열립니다.

6 `spec.topology.workers.count` 매개 변수를 찾아서 노드 수를 늘립니다.

```
...
workers:
    count: 4
...
```

```
...
workers:
    count: 2
...
```

7 변경 내용을 적용하려면 텍스트 편집기에서 파일을 저장합니다. 취소하려면 저장하지 않고 편집기를 닫습니다.

파일을 저장하면 kubectl이 변경 내용을 클러스터에 적용합니다. 백그라운드에서 감독자 클러스터의 가상 시스템 서비스는 새 작업자 노드를 프로비저닝합니다.

8 작업자 노드가 제거되었는지 확인합니다.

```
kubectl get tanzukubernetescluster tkgs-cluster-1
```

축소한 후에는 클러스터에 작업자 노드가 2개 있습니다.

```
NAME                CONTROL PLANE   WORKER   DISTRIBUTION                      AGE   PHASE
tkgs-cluster-1      3               2        v1.18.5+vmware.1-tkg.1.886c781    1d    running
```

클러스터를 수직으로 확장/축소

클러스터 노드를 호스팅하는 데 사용되는 가상 시스템 클래스를 변경하여 Tanzu Kubernetes 클러스터를 수직으로 확장/축소할 수 있습니다. 수직 확장/축소는 제어부 및 작업자 노드 모두에서 지원됩니다.

Tanzu Kubernetes Grid 서비스는 서비스에 내장된 롤링 업데이트 메커니즘을 통해 클러스터 노드를 수직으로 확장/축소하도록 지원합니다. `VirtualMachineClass` 정의를 변경하는 경우 이 서비스는 새 클래스를 사용하여 새 노드를 롤아웃하고 이전 노드를 스핀 다운합니다. Tanzu Kubernetes 클러스터 업데이트의 내용을 참조하십시오.

1 감독자 클러스터로 인증합니다.

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

2 Tanzu Kubernetes 클러스터를 실행 중인 vSphere 네임스페이스로 컨텍스트를 전환합니다.

```
kubectl config use-context tkgs-cluster-ns
```

3 네임스페이스에서 실행 중인 Kubernetes 클러스터를 나열합니다.

```
kubectl get tanzukubernetescluster -n tkgs-cluster-ns
```

4 대상 Tanzu Kubernetes 클러스터를 설명하고 VM 클래스를 확인합니다.

```
kubectl describe tanzukubernetescluster tkgs-cluster-2
```

예를 들어 다음 클러스터는 best-effort-small VM 클래스를 사용합니다.

```
Spec:
  ...
  Topology:
    Control Plane:
      Class:          best-effort-small
      ...
    Workers:
      Class:          best-effort-small
      ...
```

5 사용 가능한 VM 클래스를 나열하고 설명합니다.

```
kubectl get virtualmachineclassbinding
```

```
kubectl describe virtualmachineclassbinding
```

참고 사용하려는 VM 클래스는 vSphere 네임스페이스에 바인딩되어야 합니다. Tanzu Kubernetes 클러스터에 대한 가상 시스템 클래스의 내용을 참조하십시오.

6 대상 클러스터 매니페스트를 편집하기 위해 엽니다.

```
kubectl edit tanzukubernetescluster/tkgs-cluster-2
```

KUBE_EDITOR 또는 EDITOR 환경 변수로 정의된 텍스트 편집기에서 클러스터 매니페스트가 열립니다.

7 VM 클래스를 변경하여 매니페스트를 편집합니다.

예를 들어 제어부 및 작업자 노드에 대해 `guaranteed-xlarge` VM 클래스를 사용하도록 클러스터 매니페스트를 편집합니다.

```
spec:
  topology:
    controlPlane:
      class: guaranteed-xlarge
      ...
    workers:
      class: guaranteed-xlarge
      ...
```

8 변경 내용을 적용하려면 텍스트 편집기에서 파일을 저장합니다. 취소하려면 저장하지 않고 편집기를 닫습니다.

파일을 저장하면 kubectl이 변경 내용을 클러스터에 적용합니다. 백그라운드에서 Tanzu Kubernetes Grid 서비스는 새 노드를 프로비저닝하고 이전 노드를 삭제합니다. 롤링 업데이트 프로세스에 대한 설명은 Tanzu Kubernetes Grid 서비스 클러스터 업데이트 정보 항목을 참조하십시오.

9 클러스터가 업데이트되고 있는지 확인합니다.

```
kubectl get tanzukubernetescluster
NAME             CONTROL PLANE   WORKER   DISTRIBUTION                      AGE   PHASE
tkgs-cluster-1   3               3        v1.18.5+vmware.1-tkg.1.c40d30d   21h   updating
```

Tanzu Kubernetes 클러스터 삭제

kubectl을 사용하여 Tanzu Kubernetes Grid 서비스에서 프로비저닝된 Tanzu Kubernetes 클러스터를 삭제합니다.

kubectl을 사용하여 Tanzu Kubernetes 클러스터를 삭제하면 Kubernetes 가비지 수집을 통해 모든 종속 리소스가 삭제됩니다.

참고 vSphere Client 또는 vCenter Server CLI를 사용하여 Tanzu Kubernetes 클러스터를 삭제하려고 시도하지 마십시오.

절차

1 감독자 클러스터로 인증합니다.

vCenter Single Sign-On 사용자로 감독자 클러스터에 연결의 내용을 참조하십시오.

2 삭제하려는 Tanzu Kubernetes가 프로비저닝된 vSphere 네임스페이스로 컨텍스트를 변경합니다.

```
kubectl config use-context CLUSTER-NAMESPACE
```

예:

```
kubectl config use-context tkgs-ns-1
```

3 네임스페이스의 Tanzu Kubernetes 클러스터를 나열합니다.

```
kubectl get clusters
```

예:

```
kubectl get clusters
NAME             PHASE
tkgs-cluster-1   provisioned
```

4 다음 구문을 사용하여 Tanzu Kubernetes 클러스터를 삭제합니다.

```
kubectl delete tanzukubernetescluster --namespace CLUSTER-NAMESPACE CLUSTER-NAME
```

예:

```
kubectl delete tanzukubernetescluster --namespace tkgs-ns-1 tkgs-cluster-1
```

예상 결과:

```
tanzukubernetescluster.run.tanzu.vmware.com "tkgs-cluster-1" deleted
```

5 클러스터가 삭제되었는지 확인합니다.

```
kubectl get clusters
```

예:

```
kubectl get clusters
No resources found in tkgs-ns-1 namespace.
```

6 kubeconfig 파일에서 클러스터 컨텍스트를 삭제합니다.

```
kubectl config delete-context CONTEXT
```

예:

```
kubectl config get-contexts
CURRENT   NAME             CLUSTER         AUTHINFO
NAMESPACE
          192.0.2.1        192.0.2.1       wcp:192.0.2.1:administrator@vsphere.local
          tkgs-cluster-1   192.0.2.6       wcp:192.0.2.6:administrator@vsphere.local
*         tkgs-ns-1        192.0.2.7       wcp:192.0.2.7:administrator@vsphere.local
tkgs-ns-1
```

```
kubectl config delete-context tkgs-cluster-1
deleted context tkgs-cluster-1 from $HOME/.kube/config
```

```
kubectl config get-contexts
CURRENT   NAME             CLUSTER         AUTHINFO
NAMESPACE
          192.0.2.1        192.0.2.1       wcp:192.0.2.1:administrator@vsphere.local
*         tkgs-ns-1        192.0.2.7       wcp:192.0.2.7:administrator@vsphere.local
tkgs-ns-1
```

kubectl용 기본 텍스트 편집기 지정

Tanzu Kubernetes 클러스터를 프로비저닝, 운영, 유지 보수하는 데 도움이 되도록 kubectl용 기본 텍스트 편집기를 지정합니다.

용도

Tanzu Kubernetes 클러스터를 프로비저닝한 후에는 유지 보수가 필요합니다. 일반적인 유지 보수 작업에는 Kubernetes 버전 업그레이드 및 클러스터 노드 확장/축소 등이 포함됩니다. 이러한 작업을 수행하려면 클러스터 매니페스트를 업데이트합니다.

프로비저닝된 클러스터의 매니페스트를 업데이트하는 가장 편리한 방법은 kubectl edit 명령을 사용하는 것입니다. 이 명령은 사용자가 선택한 텍스트 편집기에서 Kubernetes 매니페스트를 엽니다. 변경 내용을 저장하면 Kubernetes가 변경 사항을 자동으로 적용하고 클러스터를 업데이트합니다.

`kubectl edit` 명령을 사용하려면 KUBE_EDITOR 환경 변수를 생성하고 선호하는 텍스트 편집기를 변수 값으로 지정합니다. 또한 이 값에 watch 플래그(`-w`)를 추가합니다. 이렇게 하면, 변경 내용을 커밋(저장)할 때 kubectl이 알 수 있습니다.

Windows

Windows에서는 선호하는 텍스트 편집기의 경로로 값이 설정된 `KUBE_EDITOR`라는 시스템 환경 변수를 생성합니다. 이 값에 watch 플래그(`-w`)를 추가합니다.

예를 들어, 다음 환경 변수는 Visual Studio Code를 kubectl용 기본 텍스트 편집기로 설정하고, 사용자가 변경 사항을 저장할 때 Kubernetes가 알 수 있도록 watch 플래그를 포함합니다.

```
KUBE_EDITOR=code -w
```

Mac OS

Mac OS에서는 선호하는 텍스트 편집기의 경로로 값이 설정된 KUBE_EDITOR라는 환경 변수를 생성합니다. 이 값에 watch 플래그(-w)를 추가합니다.그러면 변경 내용을 커밋(저장)할 때 kubectl이 알 수 있습니다.

예를 들어, .bash_profile에 다음을 추가하면 Sublime이 kubectl용 기본 텍스트 편집기로 설정되고, 사용자가 변경 사항을 저장하면 kubectl이 알 수 있도록 watch 플래그가 포함됩니다.

```
export KUBE_EDITOR="/Applications/Sublime.app/Contents/SharedSupport/bin/subl -w"
```

Linux

Linux(예: Ubuntu)에서 일반적인 기본 명령줄 EDITOR는 Vim입니다. 그렇다면 kubectl edit 명령을 사용하는 데 추가 작업이 필요하지 않습니다. 다른 편집기를 사용하려면, 선호하는 텍스트 편집기의 경로로 값이 설정된 KUBE_EDITOR라는 환경 변수를 생성합니다.

Tanzu Kubernetes 클러스터 운영

Tanzu Kubernetes Grid 서비스에는 Tanzu Kubernetes 클러스터를 운영하는 데 사용하는 사용자 지정 리소스가 포함되어 있습니다. 또한 vSphere 인프라와 긴밀히 통합되어 있기 때문에 익숙한 vSphere 도구를 사용하여 Tanzu Kubernetes 클러스터를 관리하고 유지할 수 있습니다.

kubectl을 사용하여 Tanzu Kubernetes 클러스터 상태 모니터링

kubectl을 사용하여 프로비저닝된 Tanzu Kubernetes 클러스터의 상태를 모니터링할 수 있습니다.

절차

1. 감독자 클러스터로 인증합니다. vCenter Single Sign-On 사용자로 감독자 클러스터에 연결의 내용을 참조하십시오.
2. 클러스터를 실행 중인 vSphere 네임스페이스로 전환합니다.

   ```
   kubectl config use-context SUPERVISOR-NAMESPACE
   ```

3. 네임스페이스에서 실행 중인 Tanzu Kubernetes 클러스터 목록을 봅니다.

   ```
   kubectl get tanzukubernetesclusters
   ```

   이 명령은 클러스터의 상태를 반환합니다. 상태 필드에 대한 설명은 kubectl의 Tanzu Kubernetes 클러스터 수명 주기 상태에서 참조하십시오.

4 클러스터의 세부 정보를 봅니다.

```
kubectl describe tanzukubernetescluster <cluster-name>
```

이 명령은 클러스터의 세부 정보를 반환합니다. 명령 출력의 [상태] 섹션에 클러스터에 대한 세부 정보가 표시됩니다.

```
...
Status:
  Addons:
    Cni:
      Name:    calico
      Status:  applied
    Csi:
      Name:    pvcsi
      Status:  applied
    Psp:
      Name:    defaultpsp
      Status:  applied
  Cloudprovider:
    Name:  vmware-guest-cluster
  Cluster API Status:
    API Endpoints:
      Host:  10.161.90.22
      Port:  6443
    Phase:   provisioned
  Node Status:
    test-tanzu-cluster-control-plane-0:                  ready
    test-tanzu-cluster-workers-0-749458f97c-97ljv:       ready
  Phase:                                                 running
  Vm Status:
    test-tanzu-cluster-control-plane-0:                  ready
    test-tanzu-cluster-workers-0-749458f97c-97ljv:       ready
Events:                                                  <none>
```

5 추가 `kubectl` 명령을 실행하여 클러스터에 대한 자세한 정보를 봅니다. Tanzu Kubernetes 클러스터 작동 명령 사용의 내용을 참조하십시오.

Tanzu Kubernetes 클러스터 준비 상태 확인

Tanzu Kubernetes Grid 서비스가 Tanzu Kubernetes 클러스터를 프로비저닝할 때 몇 가지 상태 조건이 보고되며, 이것을 사용하여 시스템 상태의 주요 측면에 대한 직접적인 인사이트를 얻을 수 있습니다.

TanzuKubernetesCluster 준비 상태 확인

TanzuKubernetesCluster 준비 상태를 사용하면 어떤(있는 경우) 단계 또는 구성 요소가 준비되지 않았는지 판단할 수 있습니다. ControlPlaneReady 조건 및 이유의 내용을 참조하십시오.

클러스터 준비 상태 조건을 확인한 후 자세한 진단을 위해 capwcluster 및 시스템 조건을 사용하여 장애를 더 자세히 살펴볼 수 있습니다. Tanzu Kubernetes 시스템 상태 확인 및 Tanzu Kubernetes 클러스터 상태 확인의 내용을 참조하십시오.

Tanzu Kubernetes 클러스터의 준비 상태를 확인하려면 다음을 수행합니다.

1 감독자 클러스터에 로그인합니다.

2 대상 클러스터가 프로비저닝된 네임스페이스로 컨텍스트를 전환합니다. 예:

```
kubectl config use-context tkgs-cluster-ns
```

3 `kubectl get tkc -o yaml` 명령을 실행합니다. 시스템에 클러스터 준비 상태 조건이 표시됩니다. 예:

```
status:
  addons:
    authsvc:
      conditions:
      - lastTransitionTime: "2021-01-30T19:53:54Z"
        status: "True"
        type: AuthServiceProvisioned
      name: authsvc
      status: applied
      version: 0.1-66-g8b8f07f
    cloudprovider:
      conditions:
      - lastTransitionTime: "2021-01-30T19:53:53Z"
        status: "True"
        type: CPIProvisioned
      name: vmware-guest-cluster
      status: applied
      version: 0.1-77-g5875817
    cni:
      conditions:
      - lastTransitionTime: "2021-01-30T19:53:53Z"
        status: "True"
        type: CNIProvisioned
      name: calico
      status: applied
      version: 1.16.14+vmware.1-tkg.1.ada4837
    csi:
      conditions:
      - lastTransitionTime: "2021-01-30T19:53:54Z"
        status: "True"
        type: CSIProvisioned
      name: pvcsi
      status: applied
      version: v0.0.1.alpha+vmware.79-7ecdcb1
    dns:
      conditions:
      - lastTransitionTime: "2021-01-30T19:53:48Z"
        status: "True"
        type: CoreDNSProvisioned
      name: CoreDNS
      status: applied
      version: v1.6.2_vmware.10
    proxy:
      conditions:
```

```
      - lastTransitionTime: "2021-01-30T19:53:48Z"
        status: "True"
        type: KubeProxyProvisioned
      name: kube-proxy
      status: applied
      version: 1.16.14+vmware.1
    psp:
      conditions:
      - lastTransitionTime: "2021-01-30T19:53:47Z"
        status: "True"
        type: PSPProvisioned
      name: defaultpsp
      status: applied
      version: v1.16.14+vmware.1-tkg.1.ada4837
  clusterApiStatus:
    apiEndpoints:
    - host: 192.168.1.2
      port: 6443
    phase: Provisioned
  conditions:
  - lastTransitionTime: "2021-01-30T19:53:54Z"
    status: "True"
    type: AddonsReady
  - lastTransitionTime: "2021-01-30T19:51:11Z"
    status: "True"
    type: ControlPlaneReady
  - lastTransitionTime: "2021-01-30T19:51:04Z"
    message: 3/3 Control Plane Node(s) healthy. 1/1 Worker Node(s) healthy
    status: "True"
    type: NodesHealthy
  - lastTransitionTime: "2021-01-31T21:22:45Z"
    status: "True"
    type: ProviderServiceAccountsReady
  - lastTransitionTime: "2021-01-30T19:53:50Z"
    status: "True"
    type: RoleBindingSynced
  - lastTransitionTime: "2021-01-30T19:53:58Z"
    status: "True"
    type: ServiceDiscoveryReady
  - lastTransitionTime: "2021-01-30T19:53:59Z"
    status: "True"
    type: StorageClassSynced
  - lastTransitionTime: "2021-01-27T11:34:53Z"
    status: "True"
    type: TanzuKubernetesReleaseCompatible
  - lastTransitionTime: "2021-01-27T11:34:54Z"
    message: '[1.17.13+vmware.1-tkg.2.2c133ed]'
    severity: Info
    status: "True"
    type: UpdatesAvailable
```

ControlPlaneReady 조건 및 이유

다음 표에는 `ControlPlaneReady` 조건과 그에 대한 설명이 나와 있습니다.

표 13-7. ControlPlaneReady 조건

조건 유형	설명
ControlPlaneReady	제어부 노드가 준비되어 있고 클러스터에 대해 작동하는지 보고합니다.

다음 표에는 ControlPlaneReady 조건이 false일 수 있는 이유와 그에 대한 설명이 나와 있습니다.

표 13-8. ControlPlaneReady False 이유

이유	심각도	설명
WaitingForClusterInfrastructure		클러스터가 시스템 실행에 필요한 사전 요구 사항(예: 로드 밸런서)을 기다리고 있음을 나타냅니다. 이 이유는 InfrastructureCluster가 자체 준비 조건을 보고하지 않는 경우에만 사용됩니다.
WaitingForControlPlaneInitialized		첫 번째 제어부 노드가 초기화 중임을 나타냅니다.
WaitingForControlPlane		KubeadmControlPlane의 조건을 반영합니다. 이 이유는 KubeadmControlPlane이 자체 준비 조건을 보고하지 않는 경우 사용됩니다.
클러스터 인프라가 준비되기를 기다리는 중	메시지	클러스터가 시스템 실행에 필요한 사전 요구 사항(예: 네트워킹 및 로드 밸런서)을 기다리고 있음을 나타냅니다.

NodesHealthy 조건 및 이유

다음 표에는 NodesHealthy 조건과 그에 대한 설명이 나와 있습니다.

표 13-9. NodesHealthy 조건

조건 유형	설명
NodesHealthy	TanzuKubernetesCluster 노드의 상태를 보고합니다.

다음 표에는 NodesHealthy 조건이 true가 아닌 이유와 그에 대한 설명이 나와 있습니다.

표 13-10. NodesHealthy False 이유

이유	심각도	설명
WaitingForNodesHealthy		일부 노드는 정상이 아니라는 설명입니다.

추가 기능 조건 및 이유

다음 표에는 클러스터 추가 기능 구성 요소와 관련된 조건 및 그에 대한 설명이 나와 있습니다.

표 13-11. 추가 기능 조건

조건 유형	설명
AddonsReady	TanzuKubernetesCluster 추가 기능(CoreDNS, KubeProxy, CSP, CPI, CNI, AuthSvc)에 대한 조건 요약.
CNIProvisioned	TanzuKubernetesCluster CNI(Container Network Interface) 추가 기능의 상태를 설명합니다.
CSIProvisioned	TanzuKubernetesCluster CSI(Container Storage Interface) 추가 기능의 상태를 설명합니다.
CPIProvisioned	TanzuKubernetesCluster CPI(Cloud Provider Interface) 추가 기능의 상태를 설명합니다.
KubeProxyProvisioned	TanzuKubernetesCluster KubeProxy 추가 기능의 상태를 설명합니다.
CoreDNSProvisioned	TanzuKubernetesCluster CoreDNS 추가 기능의 상태를 설명합니다.
AuthServiceProvisioned	TanzuKubernetesCluster AuthService 추가 기능의 상태를 설명합니다.
PSPProvisioned	PodSecurityPolicy의 상태를 설명합니다.

다음 표에는 추가 기능 조건이 true가 아닌 이유와 그에 대한 설명이 나와 있습니다.

표 13-12. 추가 기능 False 이유

이유	심각도	설명
AddonsReconciliationFailed		모든 추가 기능 조정 실패에 대해 요약된 이유입니다.
CNIProvisioningFailed	주의	CNI 추가 기능이 생성 또는 업데이트에 실패했음을 설명합니다.
CSIProvisioningFailed	주의	CSI 추가 기능이 생성 또는 업데이트에 실패했음을 설명합니다.
CPIProvisioningFailed	주의	CPI 추가 기능이 생성 또는 업데이트에 실패했음을 설명합니다.
KubeProxyProvisioningFailed	주의	KubeProxy 추가 기능이 생성 또는 업데이트에 실패했음을 설명합니다.
CoreDNSProvisioningFailed	주의	CoreDNS 추가 기능이 생성 또는 업데이트에 실패했음을 설명합니다.
AuthServiceProvisioningFailed	주의	AuthService 추가 기능이 생성 또는 업데이트에 실패했음을 설명합니다.
AuthServiceUnManaged		AuthService가 컨트롤러에서 관리되지 않음을 설명합니다.
PSPProvisioningFailed	주의	PodSecurityPolicy 추가 기능이 생성 또는 업데이트에 실패했음을 설명합니다.

기타 조건 및 이유

다음 표에는 StorageClass 및 RoleBinding 동기화, ProviderServiceAccount 리소스 조정, ServiceDiscovery, TanzuKubernetesCluster 호환성에 대한 조건과 그에 대한 설명이 나와 있습니다.

표 13-13. 기타 조건

조건	설명
StorageClassSynced	감독자 클러스터에서 워크로드 클러스터로의 StorageClass 동기화 상태를 설명합니다.
RoleBindingSynced	감독자 클러스터에서 워크로드 클러스터로의 RoleBinding 동기화 상태를 설명합니다.
ProviderServiceAccountsReady	제공자 서비스 계정의 상태를 설명하고 관련 역할, RoleBinding 및 암호가 생성됩니다.
ServiceDiscoveryReady	서비스 검색 상태를 설명합니다.
TanzuKubernetesReleaseCompatible	TanzuKubernetesCluster가 TanzuKubernetesRelease와 호환되는지 여부를 나타냅니다.

다음 표에는 다른 조건이 true가 아닌 이유와 그에 대한 설명이 나와 있습니다.

표 13-14. 기타 이유

이유	심각도	설명
StorageClassSyncFailed		StorageClass 동기화 실패를 보고합니다.
RoleBindingSyncFailed		RoleBinding 동기화 실패를 보고합니다.
ProviderServiceAccountsReconciliationFailed		제공자 서비스 계정 관련 리소스 조정에 실패했음을 보고합니다.
SupervisorHeadlessServiceSetupFailed		감독자 클러스터 API 서버에 대한 헤드리스 서비스 설정에 실패했음을 설명합니다.

Tanzu Kubernetes 클러스터의 전체 리소스 계층 보기

Tanzu Kubernetes 클러스터의 전체 리소스 계층은 kubectl을 사용하여 볼 수 있습니다. 클러스터 리소스의 전체 목록을 보면 문제를 유발할 수 있는 리소스를 정확하게 찾아낼 수 있습니다.

사전 요구 사항

감독자 클러스터에 연결합니다. vCenter Single Sign-On 사용자로 감독자 클러스터에 연결의 내용을 참조하십시오.

절차

1 대상 클러스터 컨텍스트를 사용하도록 컨텍스트를 전환합니다.

```
kubectl config use-context CLUSTER-NAME
```

2 다음 명령을 실행하여 클러스터 API 클러스터 리소스를 확인합니다.

```
kubectl describe clusters.cluster.x-k8s.io CLUSTER-NAME
```

이 명령은 네임스페이스, API 버전, 리소스 버전과 같은 명명된 클러스터의 리소스 계층을 반환합니다.

Tanzu Kubernetes 클러스터 수명 주기 상태 보기

Tanzu Kubernetes 클러스터의 수명 주기 상태는 vSphere 인벤토리에서 kubectl을 사용하여 볼 수 있습니다.

vSphere의 Tanzu Kubernetes 클러스터 수명 주기 상태

아래 표에는 vSphere 인벤토리에 표시되는 Tanzu Kubernetes 클러스터 상태 정보가 나열되고 설명되어 있습니다. 이 정보를 보려면 vSphere Client를 사용하여 Tanzu Kubernetes 클러스터 상태 모니터링에서 참조하십시오.

표 13-15. vSphere 인벤토리의 Tanzu Kubernetes 클러스터 상태

필드	설명	예
이름	클러스터의 사용자 정의 이름입니다.	tkg-cluster-01
생성 시간	클러스터 생성 날짜 및 시간입니다.	Mar 17, 2020, 11:42:46 PM
단계	클러스터의 수명 주기 상태입니다. 표 13-17. 클러스터 수명 주기 단계 상태의 내용을 참조하십시오.	creating
작업자 수	클러스터의 작업자 노드 수입니다.	1 또는 2 또는 5
배포 버전	클러스터에서 실행 중인 Kubernetes 소프트웨어의 버전입니다.	v1.16.6+vmware.1-tkg.1.7144628
제어부 주소	클러스터 제어부 로드 밸런서의 IP 주소입니다.	192.168.123.2

kubectl의 Tanzu Kubernetes 클러스터 수명 주기 상태

아래 표에는 kubectl에 나타나는 Tanzu Kubernetes 클러스터 상태 정보가 나열되고 설명되어 있습니다. 이 정보를 보려면 kubectl을 사용하여 Tanzu Kubernetes 클러스터 상태 모니터링에서 참조하십시오.

표 13-16. kubectl의 Tanzu Kubernetes 클러스터 상태

필드	설명	예
NAME	클러스터의 이름입니다.	tkg-cluster-01
CONTROL PLANE	클러스터의 제어부 노드 수입니다.	3
WORKER	클러스터의 작업자 노드 수입니다.	5
DISTRIBUTION	클러스터가 실행되고 있는 Kubernetes 버전입니다.	v1.16.6+vmware.1-tkg.1.5b5608b

표 13-16. kubectl의 Tanzu Kubernetes 클러스터 상태 (계속)

필드	설명	예
AGE	클러스터가 실행된 기간(일)입니다.	13d
PHASE	클러스터의 수명 주기 상태입니다. 표 13-17. 클러스터 수명 주기 단계 상태의 내용을 참조하십시오.	running

클러스터 수명 주기 단계 상태

아래 표에는 클러스터 수명 주기의 각 단계에 대한 상태가 나열되고 설명되어 있습니다. 표 13-17. 클러스터 수명 주기 단계 상태 항목을 참조하십시오.

표 13-17. 클러스터 수명 주기 단계 상태

단계	설명
creating	클러스터 프로비저닝을 시작할 수 있거나, 제어부가 생성되고 있거나, 제어부가 생성되었지만 초기화되지 않았습니다.
deleting	클러스터가 삭제되고 있습니다.
failed	클러스터 제어부 생성에 실패했으며 사용자 개입이 필요할 수 있습니다.
running	인프라가 생성 및 구성되고 제어부가 완전히 초기화되었습니다.
updating	클러스터가 업데이트되고 있습니다.

Tanzu Kubernetes 클러스터 작동 명령 사용

사용자 지정 kubectl 명령을 사용하여 Tanzu Kubernetes 클러스터를 관리할 수 있습니다. 이러한 명령은 Tanzu Kubernetes Grid 서비스에서 생성한 사용자 지정 리소스에서 사용할 수 있습니다.

Tanzu Kubernetes 클러스터를 관리하는 사용자 지정 명령

이 표에는 Tanzu Kubernetes 클러스터 관리를 위한 kubectl 명령이 나열 및 설명되어 있습니다.

표 13-18. Tanzu Kubernetes 클러스터를 관리하는 사용자 지정 명령

명령	설명
`kubectl get tanzukubernetescluster`	현재 네임스페이스의 클러스터를 나열합니다.
`kubectl get tkc`	앞선 명령의 짧은 버전입니다.
`kubectl describe tanzukubernetescluster CLUSTER-NAME`	표시된 상태, 상태 및 이벤트를 표시하는 방식으로 지정된 클러스터를 설명합니다. 프로비저닝이 완료되면 이 명령은 Kubernetes API 끝점을 향하고 있는 로드 밸런서에 대해 생성된 가상 IP를 표시합니다.

표 13-18. Tanzu Kubernetes 클러스터를 관리하는 사용자 지정 명령 (계속)

명령	설명
`kubectl get cluster-api`	클러스터 API 프로젝트의 리소스와 Tanzu Kubernetes Grid 서비스에 사용되는 클러스터 API 제공자의 리소스를 포함하여 현재 네임스페이스에서 클러스터를 지원하는 클러스터 API 리소스를 나열합니다.
`kubectl get tanzukubernetesreleases`	사용 가능한 Tanzu Kubernetes 릴리스를 나열합니다.
`kubectl get tkr`	앞선 명령의 짧은 버전입니다.
`kubectl get tkr v1.17.8---vmware.1-tkg.1.5417466 -o yaml`	명명된 Tanzu Kubernetes 릴리스에 대한 세부 정보를 제공합니다.
`kubectl get virtualmachine`	현재 네임스페이스에서 클러스터 노드를 지원하는 가상 시스템 리소스를 나열합니다.
`kubectl get vm`	앞선 명령의 짧은 버전입니다.
`kubectl describe virtualmachine VIRTUAL-MACHINE-NAME`	상태, 현재 상태 및 이벤트를 표시하는 방식으로 지정된 가상 시스템을 설명합니다.
`kubectl describe virtualmachinesetresourcepolicy`	현재 네임스페이스에서 클러스터를 지원하는 가상 시스템 설정 리소스 정책 리소스를 나열합니다. 이 리소스는 클러스터에 사용되는 vSphere 개체 리소스 풀 및 폴더를 나타냅니다.
`kubectl get virtualmachineservice`	현재 네임스페이스에서 클러스터 노드를 지원하는 가상 시스템 서비스 리소스를 나열합니다. 이러한 리소스는 서비스와 유사하지만 포드가 아닌 가상 시스템에 사용됩니다. 가상 시스템 서비스는 클러스터의 제어부 노드에 로드 밸런서를 제공하는 데 사용되며 반가상화 클라우드 제공자가 클러스터 내에서 LoadBalancer 유형의 Kubernetes 서비스를 지원하는 데 사용됩니다. `kubectl loadbalancer` 명령도 참조하십시오.
`kubectl get vmservice`	앞선 명령의 짧은 버전입니다.
`kubectl describe virtualmachineservice VIRTUAL-MACHINE-SERVICE-NAME`	표시된 클러스터 상태, 현재 상태 및 이벤트를 표시하는 방식으로 지정된 가상 시스템 서비스를 설명합니다.
`kubectl get virtualmachineimage`	사용 가능한 Tanzu Kubernetes 릴리스를 나열합니다.
`kubectl get vmimage`	앞선 명령의 바로 가기 버전입니다.
`kubectl describe vmimage VM_IMAGE_NAME`	명명된 VM 이미지에 대한 세부 정보를 봅니다.
`kubectl get loadbalancer`	클러스터에 사용되는 리소스를 포함하여 현재 네임스페이스의 로드 밸런서 리소스를 나열합니다. 가상 시스템 서비스를 위한 로드 밸런서가 생성됩니다.

표 13-18. Tanzu Kubernetes 클러스터를 관리하는 사용자 지정 명령 (계속)

명령	설명
`kubectl get virtualnetwork`	클러스터에 사용되는 리소스를 포함하여 현재 네임스페이스의 가상 네트워크 리소스를 나열합니다. 가상 네트워크는 클러스터가 프로비저닝된 각 네임스페이스와 각 클러스터 자체에 대해 생성됩니다.
`kubectl get persistentvolumeclaim`	클러스터에 사용되는 리소스를 포함하여 현재 네임스페이스의 영구 볼륨 할당 리소스를 나열합니다. 장 10 vSphere with Tanzu에서 영구 스토리지 사용의 내용을 참조하십시오.
`kubectl get cnsnodevmattachment`	현재 네임스페이스의 CNS 노드 가상 시스템 첨부 리소스를 나열합니다. 이러한 리소스는 CNS에서 관리되는 영구 볼륨이 클러스터의 노드로 작동하는 가상 시스템에 연결되었음을 나타냅니다. 장 10 vSphere with Tanzu에서 영구 스토리지 사용의 내용을 참조하십시오.
`kubectl get configmap`	클러스터 노드 생성에 사용되는 구성 맵을 포함하여 현재 네임스페이스의 구성 맵을 나열합니다. 구성 맵은 사용자가 수정할 수 없으며 모든 변경 내용을 덮어씁니다.
`kubectl get secret`	클러스터 노드의 생성 및 관리에 사용되는 암호를 포함하여 현재 네임스페이스의 암호를 나열합니다. Tanzu Kubernetes 클러스터 암호 얻기의 내용을 참조하십시오.

Tanzu Kubernetes 클러스터 네트워킹 명령 사용

Tanzu Kubernetes Grid 서비스는 노드, 포드 및 서비스에 대한 기본 네트워킹을 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝합니다. 사용자 지정 kubectl 명령을 사용하여 클러스터 네트워킹을 확인할 수 있습니다.

Tanzu Kubernetes 클러스터 네트워킹을 확인하는 사용자 지정 명령

다음 명령을 참조하여 클러스터 네트워킹을 확인합니다.

표 13-19. 클러스터 네트워킹을 확인하는 사용자 지정 kubectl 명령

명령	설명

컨텍스트를 vSphere 네임스페이스로 전환합니다. 예:
`kubectl config use-context tkgs-ns`
명령을 실행합니다.
`kubectl get tkgserviceconfigurations`
샘플 결과입니다.
`NAME DEFAULT CNI`
`tkg-service-configuration antrea`
	기본 CNI를 반환하며, 이것은 변경되지 않는 한 `antrea`입니다.
	클러스터 YAML에 명시적으로 재정의된 경우를 제외하고 기본 CNI는 클러스터 생성에 사용됩니다.
	기본 CNI를 변경하려면 Tanzu Kubernetes Grid 서비스 v1alpha1 API 구성 예시 항목을 참조하십시오.

컨텍스트를 vSphere 네임스페이스로 전환합니다. 예:
`kubectl config use-context tkgs-ns`
명령을 실행합니다.
`kubectl get virtualnetwork`
샘플 결과입니다.
`NAME SNAT`
`READY AGE`
`tkgs-cluster-12-vnet 10.191.152.133`
`True 4h3m`
	클러스터 노드의 가상 네트워크를 반환합니다.
	SNAT(소스 네트워크 주소 변환) IP 주소가 할당되었는지 확인하는 데 사용합니다.

표 13-19. 클러스터 네트워킹을 확인하는 사용자 지정 kubectl 명령 (계속)

명령	설명

컨텍스트를 vSphere 네임스페이스로 전환합니다. 예:
`kubectl config use-context tkgs-ns`
명령을 실행합니다.
`kubectl get virtualmachines -o wide`
샘플 결과입니다.
`NAME POWERSTATE CLASS IMAGE PRIMARY-IP AGE tkgs-cluster-12-control-plane-... poweredOn guaranteed-medium ob-...-v1.21.6---vmware.1-tkg.1.b3d708a 10.244.0.66 4h6m tkgs-cluster-12-worker-... poweredOn guaranteed-medium ob-...-v1.21.6---vmware.1-tkg.1.b3d708a 10.244.0.68 4h3m tkgs-cluster-12-worker-... poweredOn guaranteed-medium ob-...-v1.21.6---vmware.1-tkg.1.b3d708a 10.244.0.67 4h3m`
	클러스터 노드에 대한 가상 네트워크 인터페이스를 반환합니다.
	각 클러스터 노드에 대한 가상 시스템에 IP 주소가 할당되었는지 확인하는 데 사용합니다.

컨텍스트를 vSphere 네임스페이스로 전환합니다. 예:
`kubectl config use-context tkgs-ns`
명령을 실행합니다.
`kubectl get virtualmachineservices`
샘플 결과입니다.
`NAME TYPE AGE tkgs-cluster-12-control-plane-service LoadBalancer 3h53m`
	각 클러스터 노드에 대한 가상 시스템 서비스를 반환합니다.
	상태가 업데이트 되었는지 그리고 로드 밸런서 VIP(가상 IP) 주소를 포함하는지 확인하려는 경우 사용합니다.

표 13-19. 클러스터 네트워킹을 확인하는 사용자 지정 kubectl 명령 (계속)

명령	설명

컨텍스트를 TKGS 클러스터 네임스페이스로 전환합니다. 예:
`kubectl config use-context tkgs-cluster-10`
명령을 실행합니다.
`kubectl get services -n NAMESPACE`
확인합니다.
`curl -k https://EXTERNAL-IP:PORT/healthz`
	클러스터 API 액세스를 위해 생성된 Kubernetes 서비스 로드 밸런서를 반환합니다. 외부 IP가 할당되었는지 확인하려는 경우 사용합니다.
	로드 밸런서 서비스의 외부 IP 주소 및 포트를 사용하여 API에 액세스할 수 있는지 확인하려면 `curl`을 사용합니다.

컨텍스트를 vSphere 네임스페이스로 전환합니다. 예:
`kubectl config use-context tkgs-ns`
명령을 실행합니다.
`kubectl get endpoints`
샘플 결과입니다.
`NAME ENDPOINTS AGE tkgs-cluster-12-control-plane-service 10.244.0.66:6443 3h44m`
	클러스터에 대한 끝점(제어부 노드)을 반환합니다. 각 끝점이 생성되었고 끝점 풀에 포함되었는지 확인하는 데 사용합니다.

Tanzu Kubernetes 클러스터 암호 얻기

Tanzu Kubernetes 클러스터는 암호를 사용하여 Tanzu Kubernetes 클러스터 운영을 위한 토큰, 키 및 암호를 저장합니다.

Tanzu Kubernetes 클러스터 암호 목록

Kubernetes 암호는 암호, 토큰 또는 SSH 키와 같은 소량의 중요한 데이터를 저장하는 개체입니다. Tanzu Kubernetes 클러스터 관리자는 클러스터를 운영하는 동안 몇 가지 암호를 사용할 수 있습니다. 아래 표에는 관리자가 사용할 수 있는 주요 암호 클러스터가 나열되고 설명되어 있습니다.

참고 이 목록은 포괄적이지 않습니다. 문제 해결을 위해 클러스터 노드에 액세스하는 데 사용하거나 수동으로 순환해야 할 수도 있는 암호만 포함됩니다.

암호	설명
TANZU-KUBERNETES-CLUSTER-NAME-ccm-token-RANDOM	반가상화 클라우드 제공자의 클라우드 컨트롤러 관리자가 vSphere 네임스페이스에 연결하는 데 사용하는 서비스 계정 토큰입니다. 이 자격 증명의 순환을 트리거하려면 암호를 삭제합니다.
TANZU-KUBERNETES-CLUSTER-NAME-pvcsi-token-RANDOM	반가상화 CSI 플러그인이 vSphere 네임스페이스에 연결하는 데 사용하는 서비스 계정 토큰입니다. 이 자격 증명의 순환을 트리거하려면 암호를 삭제합니다. vSphere with Tanzu가 vSphere 스토리지와 통합되는 방식의 내용을 참조하십시오.
TANZU-KUBERNETES-CLUSTER-NAME-kubeconfig	kubernetes-admin 사용자로 클러스터의 제어부에 연결하는 데 사용할 수 있는 kubeconfig 파일입니다. 이 암호는 vCenter Single Sign-On 인증을 사용할 수 없는 경우 클러스터에 액세스하여 문제를 해결하는 데 사용할 수 있습니다. 관리자로 Tanzu Kubernetes 클러스터 제어부에 연결의 내용을 참조하십시오.
TANZU-KUBERNETES-CLUSTER-NAME-ssh	vmware-system-user로 클러스터 노드에 연결하는 데 사용할 수 있는 SSH 개인 키입니다. 이 암호는 클러스터 노드에 SSH로 사용하여 문제를 해결하는 데 사용할 수 있습니다. 개인 키를 사용하여 시스템 사용자로 Tanzu Kubernetes 클러스터 노드에 SSH를 통해 연결의 내용을 참조하십시오.
TANZU-KUBERNETES-CLUSTER-NAME-ssh-password	vmware-system-user로 클러스터 노드에 연결하는 데 사용할 수 있는 암호입니다. 개인 키를 사용하여 시스템 사용자로 Tanzu Kubernetes 클러스터 노드에 SSH를 통해 연결의 내용을 참조하십시오.
TANZU-KUBERNETES-CLUSTER-NAME-ca	kubectl에서 Kubernetes API 서버에 안전하게 연결하는 데 사용되는 Tanzu Kubernetes 클러스터 제어부의 루트 CA 인증서입니다.

Tanzu Kubernetes 시스템 상태 확인

Tanzu Kubernetes Grid 서비스가 Tanzu Kubernetes 클러스터를 프로비저닝할 때 몇 가지 상태 조건이 보고되며, 이것을 사용하여 시스템 상태의 주요 측면에 대한 직접적인 인사이트를 얻을 수 있습니다.

시스템 상태 조건 정보

Tanzu Kubernetes Grid 서비스에 의해 프로비저닝된 Tanzu Kubernetes 클러스터는 여러 움직이는 부분으로 구성되며, 모두 독립적이지만 관련된 컨트롤러에 의해 운영되고 함께 작동하여 Kubernetes 노드 집합을 구축하고 유지합니다. `TanzuKubernetesCluster` 개체는 시스템 상태에 대해 세분화된 정보를 제공하는 상태 조건을 제공합니다.

시스템 상태 확인

Tanzu Kubernetes 시스템의 상태를 확인하려면 다음을 수행합니다.

1 `kubectl describe machine` 명령을 실행합니다.

상태가 Ready이면 시스템은 정상입니다. 하지만 시스템 조건이 false(예: `InfrastructureReady`)이면 시스템이 준비되지 않은 것입니다.

2 시스템이 준비되지 않은 경우 다음 명령을 실행하고 인프라에 어떤 문제가 있는지 확인합니다.

```
kubectl describe wcpmachine
```

시스템 상태 조건 목록

이 표에는 Tanzu Kubernetes 클러스터에 사용할 수 있는 시스템 상태 조건이 나열 및 정의되어 있습니다.

조건	설명
`ResourcePolicyReady`	리소스 정책이 성공적으로 생성되었다고 보고합니다.
`ResourcePolicyCreationFailed`	ResourcePolicy를 생성하는 동안 오류가 발생하면 보고됩니다.
`ClusterNetworkReady`	클러스터 네트워크가 성공적으로 프로비저닝되었다고 보고합니다.
`ClusterNetworkProvisionStarted`	클러스터 네트워크가 준비되기를 기다릴 때 보고됩니다.
`ClusterNetworkProvisionFailed`	네트워크 프로비저닝 중에 오류가 발생하면 보고됩니다.
`LoadBalancerReady`	정적 제어부 끝점이 성공적으로 조정되었다고 보고합니다.
`LoadBalancerCreationFailed`	로드 밸런서 관련 리소스 생성이 실패한 경우 보고됩니다.
`WaitingForLoadBalancerIP`	로드 밸런서 IP가 존재하기를 기다릴 때 보고됩니다.
`VMProvisioned`	가상 시스템이 생성되고 전원이 켜지고 IP가 할당된 것을 보고합니다.
`WaitingForBootstrapData`	프로비저닝 프로세스를 시작하기 전에 vSphereMachine이 부트스트랩 스크립트가 준비될 때까지 기다릴 때 보고됩니다.
`VMCreationFailed`	VM CRD 또는 해당 부트스트랩 ConfigMap 생성에 실패했다고 보고합니다.
`VMProvisionStarted`	가상 시스템이 현재 생성 프로세스에 있을 때 보고됩니다.
`PoweringOn`	가상 시스템에서 현재 전원 켜기 순서를 실행하고 있을 때 보고됩니다.
`WaitingForNetworkAddress`	시스템 네트워크 설정이 활성 상태가 되기를 기다릴 때 보고됩니다.
`WaitingForBIOSUUID`	시스템에 BIOS UUID가 지정되기를 기다릴 때 보고됩니다.

조건 필드

각 조건에는 여러 필드가 포함될 수 있습니다.

`Type`	조건 유형을 설명합니다. 예를 들면 `ResourcePolicyReady`입니다. `Ready` 조건의 경우 다른 모든 조건에 대한 요약입니다.
`Status`	유형의 상태를 설명합니다. 상태는 `True`, `False` 또는 `Unknown`일 수 있습니다.

Severity	Reason의 분류입니다. Info는 조정이 진행 중임을 의미합니다. Warning은 오류가 발생했을 수 있으며 재시도할 수 있음을 의미합니다. Error는 오류가 발생했고 해결하려면 수동 작업이 필요함을 의미합니다.
Reason	상태가 False인 이유를 제공합니다. 준비 대기 또는 실패 이유일 수 있습니다. 일반적으로 상태가 False인 경우에 throw됩니다.
Message	사람이 읽을 수 있는 Reason을 설명하는 정보입니다.

Tanzu Kubernetes 클러스터 상태 확인

Tanzu Kubernetes Grid 서비스가 Tanzu Kubernetes 클러스터를 프로비저닝할 때 몇 가지 상태 조건이 보고되며, 이것을 사용하여 클러스터 상태의 주요 측면에 대한 직접적인 인사이트를 얻을 수 있습니다.

클러스터 상태 조건 정보

Tanzu Kubernetes Grid 서비스에 의해 프로비저닝된 Tanzu Kubernetes 클러스터는 여러 움직이는 부분으로 구성되며, 모두 독립적이지만 관련된 컨트롤러에 의해 운영되고 함께 작동하여 Kubernetes 노드 집합을 구축하고 유지합니다. TanzuKubernetesCluster 개체는 클러스터 및 시스템 상태에 대해 세분화된 정보를 제공하는 상태 조건을 제공합니다.

클러스터 상태 확인

Tanzu Kubernetes 클러스터의 상태를 확인하려면 다음을 수행합니다.

1 kubectl describe cluster 명령을 실행합니다.

상태가 Ready이면 클러스터 인프라와 클러스터 제어부 모두가 준비되었음을 의미합니다. 예:

```
Status:
  Conditions:
    Last Transition Time:     2020-11-24T21:37:32Z
    Status:                   True
    Type:                     Ready
    Last Transition Time:     2020-11-24T21:37:32Z
    Status:                   True
    Type:                     ControlPlaneReady
    Last Transition Time:     2020-11-24T21:31:34Z
    Status:                   True
    Type:                     InfrastructureReady
```

하지만 클러스터 조건이 false이면 클러스터가 준비되지 않은 것이며 메시지 필드에 무엇이 잘못되었는지 설명됩니다. 예를 들어 다음은 상태가 False이고 인프라가 준비되지 않았기 때문입니다.

```
Status:
  Conditions:
    Last Transition Time:     2020-11-24T21:37:32Z
    Status:                   False
    Type:                     Ready
```

```
    Last Transition Time:     2020-11-24T21:37:32Z
    Status:                   True
    Type:                     ControlPlaneReady
    Last Transition Time:     2020-11-24T21:31:34Z
    Status:                   False
    Type:                     InfrastructureReady
```

2 클러스터가 준비되지 않은 경우 다음 명령을 실행하여 클러스터 인프라에 어떤 문제가 있는지 확인합니다.

```
kubectl describe wcpcluster
```

클러스터 상태 조건 목록

이 표에는 Tanzu Kubernetes 클러스터에 사용할 수 있는 상태 조건이 나열 및 정의되어 있습니다.

조건	설명
Ready	클러스터 API 개체의 작동 상태를 요약합니다.
Deleting	상태가 True가 아니며 기본 개체가 현재 삭제되고 있기 때문입니다.
DeletionFailed	상태가 True가 아니며 삭제하는 동안 기본 개체에 문제가 발생했기 때문입니다. 조정자가 삭제를 재시도하기 때문에 주의입니다.
Deleted	상태가 True가 아니며 기본 개체가 삭제되었기 때문입니다.
InfrastructureReady	이 클러스터에 대해 정의된 인프라 개체의 현재 상태에 대한 요약을 보고합니다.
WaitingForInfrastructure	클러스터가 기본 인프라를 사용할 수 있기를 기다릴 때 보고됩니다. 참고: 이 조건은 인프라가 준비 상태를 보고하지 않을 때 폴백으로 사용됩니다.
ControlPlaneReady	클러스터 제어부가 준비되었을 때 보고됩니다.
WaitingForControlPlane	클러스터가 제어부를 사용할 수 있기를 기다릴 때 보고됩니다. 참고: 이 조건은 제어부가 준비 상태를 보고하지 않을 때 폴백으로 사용됩니다.

조건 필드

각 조건에는 여러 필드가 포함될 수 있습니다.

Type	조건 유형을 설명합니다. 예를 들면 ControlPlaneReady입니다. Ready 조건의 경우 다른 모든 조건에 대한 요약입니다.
Status	유형의 상태를 설명합니다. 상태는 True, False 또는 Unknown일 수 있습니다.
Severity	Reason의 분류입니다. Info는 조정이 진행 중임을 의미합니다. Warning은 오류가 발생했을 수 있으며 재시도할 수 있음을 의미합니다. Error는 오류가 발생했고 해결하려면 수동 작업이 필요함을 의미합니다.

Reason	상태가 False인 이유를 제공합니다. 준비 대기 또는 실패 이유일 수 있습니다. 일반적으로 상태가 False인 경우에 throw됩니다.
Message	사람이 읽을 수 있는 Reason을 설명하는 정보입니다.

vSphere Client를 사용하여 Tanzu Kubernetes 클러스터 상태 모니터링

vSphere Client를 사용하여 Tanzu Kubernetes 클러스터의 상태를 모니터링할 수 있습니다.

절차

1. vSphere Client를 사용하여 vCenter Server에 로그인합니다.
2. **메뉴**에서 **호스트 및 클러스터** 보기를 선택합니다.
3. 감독자 클러스터가 생성된 **데이터 센터 > 클러스터** 개체를 확장합니다.
4. **네임스페이스** 리소스 풀을 확장합니다.
5. Tanzu Kubernetes 클러스터를 배포한 vSphere 네임스페이스를 선택합니다.

   각 Tanzu Kubernetes 클러스터는 해당 네임스페이스 리소스 풀 내에 폴더로 나열됩니다. 각 Tanzu Kubernetes는 이름 옆 3개의 육각형 아이콘을 통해 그래픽으로 표시됩니다.
6. **메뉴 > VM 및 템플릿** 보기로 전환합니다.

   클러스터 폴더 내에 클러스터 노드를 구성하는 VM이 표시됩니다.
7. vSphere 네임스페이스를 선택한 다음, **계산** 탭을 선택합니다.
8. **VMware 리소스**에서 **Tanzu Kubernetes**를 선택합니다.

   이 vSphere 네임스페이스에 배포된 각 Tanzu Kubernetes 클러스터가 나열됩니다. 각 상태 필드에 대한 설명은 vSphere의 Tanzu Kubernetes 클러스터 수명 주기 상태에서 참조하십시오.

TKGS 클러스터에 워크로드 및 패키지 배포

14

Tanzu Kubernetes 클러스터에 워크로드 및 패키지를 설치하려면 이 섹션의 컨텐츠를 참조하십시오.

본 장은 다음 항목을 포함합니다.

- Tanzu Kubernetes 클러스터에 워크로드 배포
- Tanzu Kubernetes 클러스터에 TKG 패키지 배포
- Tanzu Kubernetes 클러스터에 AI/ML 워크로드 배포

Tanzu Kubernetes 클러스터에 워크로드 배포

포드, 서비스, 영구 볼륨 및 더 높은 수준의 리소스(예: 배포 및 복제 집합)를 사용하여 애플리케이션 워크로드를 Tanzu Kubernetes 클러스터에 배포할 수 있습니다.

Tanzu Kubernetes 클러스터에 테스트 워크로드 배포

Tanzu Kubernetes 클러스터를 프로비저닝했으면 테스트 워크로드를 배포하고 클러스터 기능을 검증하는 것이 좋습니다.

kuard 데모 앱을 사용하여 Tanzu Kubernetes 클러스터가 가동되어 실행 중인지 확인합니다.

사전 요구 사항

- Tanzu Kubernetes 클러스터를 프로비저닝합니다. TKGS v1alpha2 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하는 워크플로의 내용을 참조하십시오.
- vCenter Single Sign-On 사용자로 Tanzu Kubernetes 클러스터에 연결

절차

1 구성 컨텍스트를 대상 Tanzu Kubernetes 클러스터로 전환합니다.

```
kubectl config use-context TANZU-KUBERNETES-CLUSTER-NAME
```

예:

```
kubectl config use-context tkgs-cluster-1
Switched to context "tkgs-cluster-1".
```

2 kuard 데모 앱을 배포합니다.

```
kubectl run --restart=Never --image=gcr.io/kuar-demo/kuard-amd64:blue kuard
```

예상 결과:

```
pod/kuard created
```

3 포드가 실행 중인지 확인합니다.

```
kubectl get pods
```

예상 결과:

```
NAME                    READY   STATUS    RESTARTS   AGE
kuard                   1/1     Running   0          10d
```

4 포트 컨테이너 포트 8080을 로컬 호스트 포트 8080으로 전달합니다.

```
kubectl port-forward kuard 8080:8080
```

예상 결과:

```
Forwarding from 127.0.0.1:8080 -> 8080
Forwarding from [::1]:8080 -> 8080
Handling connection for 8080
```

5 브라우저를 사용하여 http://localhost:8080으로 이동합니다.

kuard 데모 애플리케이션 웹 페이지가 나타나면 클러스터의 여러 측면과 상호 작용하고 확인할 수 있습니다. 예를 들어 작동 여부 및 준비 여부 프로브를 수행합니다.

6 kubectl 세션에서 Ctrl+C 키를 눌러 포트 전달을 중지합니다.

7 kuard 포드를 삭제합니다.

```
kubectl delete pod kuard
```

예상 결과:

```
pod "kuard" deleted
```

8 포드가 삭제되었는지 확인합니다.

```
kubectl get pods
```

Octant 설치 및 실행

Octant 웹 인터페이스를 설치하여 Tanzu Kubernetes 클러스터 워크로드, 네임스페이스, 메타데이터 등을 시각화할 수 있습니다.

Octant 정보

Octant는 Kubernetes 클러스터 및 애플리케이션을 볼 수 있는 오픈 소스 웹 인터페이스입니다.

Octant는 `kubectl`을 실행하는 동일한 클라이언트에서 설치하고 실행합니다. 일반 플랫폼에 대한 설치 지침은 아래에 제공됩니다. 자세한 내용은 Octant 사이트를 참조하십시오.

Octant가 설치되면 이를 통해 `kubectl`을 사용하는 Tanzu Kubernetes 클러스터에 로그인하고 `octant` 명령을 실행합니다.

Windows에 Octant 설치

Windows PowerShell용 Chocolatey 패키지 관리자를 설치합니다.

PowerShell 세션을 관리자로 실행합니다.

다음 명령을 사용하여 Octant를 설치합니다.

```
choco install octant --confirm
```

Mac에 Octant 설치

Homebrew 패키지 관리자를 설치합니다.

다음 명령을 사용하여 Octant를 설치합니다.

```
brew install octant
```

Ubuntu에 Octant 설치

릴리스 페이지에서 `.deb`를 다운로드합니다.

`dpkg -i` 명령을 사용하여 설치합니다.

Tanzu Kubernetes 서비스 로드 밸런서 예

Tanzu Kubernetes 클러스터에서 외부 로드 밸런서를 프로비저닝하기 위해 LoadBalancer 유형의 서비스를 생성할 수 있습니다. 로드 밸런서 서비스는 공용 IP 주소를 노출합니다. 외부 로드 밸런서의 트래픽은 클러스터 포드에서 리디렉션할 수 있습니다.

서비스로 노출된 Kubernetes 포드에 대한 외부 로드 밸런서를 프로비저닝할 수 있습니다. 예를 들어 Nginx 컨테이너를 배포하고 LoadBalancer 유형의 Kubernetes 서비스로 노출할 수 있습니다.

사전 요구 사항

- Kubernetes 설명서에서 LoadBalancer 서비스 유형을 검토합니다.
- Tanzu Kubernetes 클러스터를 프로비저닝합니다. TKGS v1alpha2 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하는 워크플로의 내용을 참조하십시오.
- 대상 Tanzu Kubernetes 클러스터에 연결합니다. vCenter Single Sign-On 사용자로 Tanzu Kubernetes 클러스터에 연결의 내용을 참조하십시오.

절차

1 기본 권한이 있는 PSP에 적절한 역할 바인딩을 생성합니다. 포드 보안 정책에 대한 역할 바인딩 예의 내용을 참조하십시오.

2 다음 `nginx-lbsvc.yaml` YAML 파일을 생성합니다.

이 YAML 파일은 LoadBalancer 유형의 Kubernetes 서비스를 정의하고 서비스에 대한 외부 로드 밸런서로 Nginx 컨테이너를 배포합니다.

```
kind: Service
apiVersion: v1
metadata:
  name: srvclb-ngnx
spec:
  selector:
    app: hello
    tier: frontend
  ports:
  - protocol: "TCP"
    port: 80
    targetPort: 80
  type: LoadBalancer
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: loadbalancer
spec:
  replicas: 2
  selector:
    matchLabels:
      app: hello
  template:
    metadata:
      labels:
        app: hello
        tier: frontend
    spec:
      containers:
      - name: nginx
        image: "nginxdemos/hello"
```

3 YAML을 적용합니다.

```
kubectl apply -f nginx-lbsvc.yaml
```

4 Nginx 서비스의 배포를 확인합니다.

```
kubectl get services
```

`srvclb-ngnx`는 외부 및 내부 IP 주소를 사용하여 작동합니다.

```
NAME          TYPE           CLUSTER-IP       EXTERNAL-IP     PORT(S)        AGE
srvclb-ngnx   LoadBalancer   10.11.12.19      10.19.15.89     80:30818/TCP   18m
```

5 브라우저를 사용하여 Nginx LoadBalancer 서비스의 외부 IP 주소를 입력합니다.

로드 밸런서의 세부 정보 및 `NGINX` 배너 메시지가 보입니다.

고정 IP 주소가 있는 Tanzu Kubernetes 서비스 로드 밸런서 예

고정 IP 주소를 사용하도록 LoadBalancer 유형의 Kubernetes 서비스를 구성할 수 있습니다. 이 기능을 구현하기 전에 최소 구성 요소 요구 사항, 중요한 보안 고려 사항 및 클러스터 강화 지침을 알고 있어야 합니다.

최소 요구 사항

LoadBalancer 유형의 Kubernetes 서비스에 대한 정적 IP 주소는 다음 요구 사항을 충족하는 Tanzu Kubernetes 클러스터에서 지원됩니다.

구성 요소	최소 요구 사항	추가 정보
vCenter Server 및 ESXi	vSphere 7.0 업데이트 2	릴리스 정보를 참조하십시오.
감독자 클러스터	`v1.19.1+vmware.2-vsc0.0.8-17610687`	vSphere 네임스페이스 업데이트를 수행하여 감독자 클러스터 업데이트의 내용을 참조하십시오.
로드 밸런서	NSX-T Data Center v3.1 또는 NSX Advanced 20.1.x	릴리스 정보를 참조하십시오.
Tanzu Kubernetes 릴리스	최신 Tanzu Kubernetes 릴리스 중 하나입니다.	업데이트를 위한 Tanzu Kubernetes 클러스터 호환성 확인의 내용을 참조하십시오.

LoadBalancer 유형의 서비스에 고정 IP 사용

일반적으로 LoadBalancer 유형의 Kubernetes 서비스를 정의할 때는 로드 밸런서에서 할당된 사용 후 삭제 IP 주소를 얻습니다. Tanzu Kubernetes 서비스 로드 밸런서 예의 내용을 참조하십시오.

또는 로드 밸런서에 대한 고정 IP 주소를 지정할 수 있습니다. 서비스를 생성할 때 로드 밸런서 인스턴스는 사용자가 할당한 고정 IP 주소로 프로비저닝됩니다.

다음 예제 서비스는 고정 IP 주소로 지원되는 로드 밸런서를 구성하는 방법을 보여줍니다. 서비스 규격에는 `loadBalancerIP` 매개 변수와 IP 주소 값(이 예제의 경우 `10.11.12.49`)이 포함됩니다.

```
kind: Service
apiVersion: v1
metadata:
  name: load-balancer-service-with-static-ip
spec:
```

```
  selector:
    app: hello-world
    tier: frontend
  ports:
  - protocol: "TCP"
    port: 80
    targetPort: 80
  type: LoadBalancer
  loadBalancerIP: 10.11.12.49
```

NSX Advanced Load Balancer의 경우 로드 밸런서가 설치될 때 구성된 IPAM 풀의 IP 주소를 사용합니다. 서비스가 생성되고 고정 IP 주소가 할당되면 로드 밸런서는 이 서비스를 할당된 것으로 표시하고 사용 후 삭제 IP 주소와 동일한 방식으로 IP 주소의 수명 주기를 관리합니다. 즉, 서비스가 제거되면 IP 주소 할당이 취소되고 재할당이 가능해집니다.

NSX-T 로드 밸런서의 경우 두 가지 옵션이 있습니다. 기본 메커니즘은 NSX Advanced Load Balancer와 동일합니다. 로드 밸런서가 설치될 때 구성된 IP 풀에서 가져온 IP 주소를 사용하는 것입니다. 고정 IP 주소가 할당되면 로드 밸런서는 자동으로 할당된 것으로 표시하고 해당 수명 주기를 관리합니다.

두 번째 NSX-T 옵션은 고정 IP 주소를 수동으로 미리 할당하는 것입니다. 이 경우 로드 밸런서에 할당된 외부 로드 밸런서 IP 풀의 일부가 아닌 부동 IP 풀에서 가져온 IP 주소를 대신 사용합니다. 이 경우 NSX Manager를 사용하여 IP 주소의 할당 및 수명 주기를 수동으로 관리합니다.

중요한 보안 고려 사항 및 강화 요구 사항

이 기능을 사용할 때 주의해야 할 잠재적인 보안 문제가 있습니다. 개발자가 `Service.status.loadBalancerIP` 값에 패치를 적용할 수 있으면 패치가 적용되어 있는 IP 주소로 향하는 클러스터의 트래픽을 개발자가 강탈할 수 있습니다. 특히, `patch` 권한이 있는 Role 또는 ClusterRole이 이 기능이 구현된 클러스터의 서비스 또는 사용자 계정에 바인딩된 경우 해당 계정 소유자는 자체 자격 증명을 사용하여 `kubectl` 명령을 실행하고 로드 밸런서에 할당된 고정 IP 주소를 변경할 수 있습니다.

로드 밸런서 서비스에 고정 IP 할당을 사용할 경우 발생할 수 있는 보안 위험을 피하려면 이 기능을 구현할 각 클러스터를 강화해야 합니다. 이렇게 하려면 개발자에 대해 정의한 Role 또는 ClusterRole이 `apiGroups:""` 및 `resources: services/status`에 대한 `patch` 동사를 허용하지 않아야 합니다. 예제 역할 코드 조각은 이 기능을 구현할 때 하지 말아야 할 내용을 보여줍니다.

패치를 허용하지 안 됩니다.

```
- apiGroups:
  - ""
  resources:
  - services/status
  verbs:
  - patch
```

개발자에게 패치 권한이 있는지 확인하려면 해당 사용자로 다음 명령을 실행합니다.

```
kubectl --kubeconfig <KUBECONFIG> auth can-i patch service/status
```

명령이 yes를 반환하면 사용자에게 패치 권한이 있습니다. 자세한 내용은 Kubernetes 설명서에서 API 액세스 확인을 참조하십시오.

개발자에게 클러스터에 대한 액세스 권한을 부여하려면 개발자에게 Tanzu Kubernetes 클러스터에 대한 액세스 권한 부여 항목을 참조하십시오. 사용자 지정할 수 있는 샘플 역할 템플릿에 대한 자세한 내용은 포드 보안 정책에 대한 역할 예 항목을 참조하십시오. 클러스터 액세스를 제한하는 방법에 대한 예는 https://kubernetes.io/docs/reference/access-authn-authz/rbac/#role-example에서 참조하십시오.

로컬 트래픽 정책 및 소스 IP 범위에 대한 Tanzu Kubernetes 서비스 로드 밸런서 예제

수신 요청의 소스 IP 주소를 기반으로 로드 밸런서 트래픽을 허용하고 로컬 포드 트래픽만 허용하도록 LoadBalancer 유형의 Kubernetes 서비스를 구성할 수 있습니다.

최소 요구 사항

다음과 같은 최소 요구 사항을 충족하는 Tanzu Kubernetes 클러스터에서 LoadBalancer 유형의 Kubernetes 서비스와 함께 `externalTrafficPolicy` 및 `LoadBalancerSourceRanges` 기능을 사용할 수 있습니다.

구성 요소	최소 요구 사항	추가 정보
vCenter Server 및 ESXi	vSphere 7.0 업데이트 2	릴리스 정보를 참조하십시오.
감독자 클러스터	`v1.19.1+vmware.2-vsc0.0.8-17610687`	vSphere 네임스페이스 업데이트를 수행하여 감독자 클러스터 업데이트의 내용을 참조하십시오.
로드 밸런서	NSX-T Data Center v3.1	장 4 vSphere with Tanzu에 대한 네트워킹의 내용을 참조하십시오.
Tanzu Kubernetes 릴리스	최신 Tanzu Kubernetes 릴리스 중 하나입니다.	업데이트를 위한 Tanzu Kubernetes 클러스터 호환성 확인의 내용을 참조하십시오.

로컬 트래픽 정책 및 소스 IP 범위에 대한 지원 정보

NSX-T Data Center 네트워킹을 사용하는 경우에는 외부 트래픽 정책 및 로드 밸런서 소스 IP 범위를 허용하도록 LoadBalancer 유형의 Kubernetes 서비스를 구성할 수 있습니다. `externalTrafficPolicy` 기능을 사용하면 포드 트래픽을 로컬 노드로 제한할 수 있습니다. `LoadBalancerSourceRange` 기능을 사용하면 허용하거나 차단할 소스 IP 주소를 지정할 수 있습니다.

로컬 트래픽 전용 서비스 예

다음 로드 밸런서 서비스 규격은 `externalTrafficPolicy` 매개 변수가 `Local`로 설정된 로드 밸런서 인스턴스를 구성합니다. 그 결과 포드 트래픽은 로컬 포드가 실행 중인 노드로만 라우팅됩니다.

```
apiVersion: v1
kind: Service
metadata:
```

```
  name: local-only
spec:
  selector:
    app: testApp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
  externalTrafficPolicy: Local
  type: LoadBalancer
```

이 기능은 NSX-T 상태 점검 모니터를 사용하여 작동합니다. NSX-T 관리 측면에서 이 기능의 내부 작업을 알고 있어야 합니다.

NSX-T 상태 점검 모니터는 LoadBalancer 유형의 서비스에 해당하는 서버 풀에 대해 kube-proxy가 할당한 Kubernetes 상태 점검 NodePort를 감시합니다. NSX-T 상태 점검 모니터는 HTTP GET 요청을 대상 상태 점검 NodePort로 전송합니다. 노드의 kube-proxy는 실행 중인 로컬 포드가 없는 경우 HTTP 상태 코드 500을 반환합니다. 로컬 포드가 없는 노드는 NSX-T에 의해 DOWN으로 표시되고 NSX Manager에 표시됩니다. 트래픽은 로컬 포드가 실행 중인 노드로만 라우팅됩니다.

소스 IP 범위를 기반으로 트래픽을 허용하는 서비스 예

다음 로드 밸런서 서비스 규격은 허용되는 소스 IP CIDR 어레이로 `loadBalancerSourceRanges` 매개 변수를 구성합니다. 이러한 소스 IP 범위에서 발생하는 인바운드 요청만 허용되며, 다른 모든 인바운드 트래픽은 삭제됩니다.

```
apiVersion: v1
kind: Service
metadata:
  name: allow-based-on-source-IPs
spec:
  selector:
    app: testApp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
  loadBalancerSourceRanges:
  - 10.0.0.0/24
  - 10.1.0.0/24
  type: LoadBalancer
```

Nginx를 사용하는 Tanzu Kubernetes 수신 예

Kubernetes 수신 리소스는 클러스터 외부에서 클러스터 내부의 하나 이상의 서비스로 HTTP 또는 HTTPS 라우팅을 제공합니다. Tanzu Kubernetes 클러스터는 Nginx와 같은 타사 컨트롤러를 통한 수신을 지원합니다.

이 자습서에서는 외부 트래픽을 Tanzu Kubernetes 클러스터의 서비스로 라우팅하기 위해 NGINX를 기반으로 Kubernetes 수신 서비스를 배포하는 방법을 보여줍니다. 수신 서비스에는 수신 컨트롤러가 필요합니다. NGINX 수신 컨트롤러는 Helm을 사용하여 설치합니다. Helm은 Kubernetes용 패키지 관리자입니다.

참고 이 작업을 수행하는 방법은 여러 가지입니다. 여기에 나와있는 단계는 한 가지 접근 방법을 제공합니다. 사용 환경에 따라 다른 방법이 더 적합할 수 있습니다.

사전 요구 사항

- Kubernetes 설명서에서 수신 리소스를 검토합니다.
- Nginx 수신 컨트롤러 설명서를 검토합니다.
- Tanzu Kubernetes 클러스터를 프로비저닝합니다. TKGS v1alpha2 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하는 워크플로의 내용을 참조하십시오.
- 포드 보안 정책을 사용하도록 설정합니다. 포드 보안 정책에 대한 역할 예의 내용을 참조하십시오.
- Tanzu Kubernetes 클러스터에 연결합니다. vCenter Single Sign-On 사용자로 Tanzu Kubernetes 클러스터에 연결의 내용을 참조하십시오.

절차

1 설명서를 참조하여 Helm을 설치합니다.

2 Helm을 사용하여 NGINX 수신 컨트롤러를 설치합니다.

```
helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
helm install ingress-nginx ingress-nginx/ingress-nginx
```

3 Nginx 수신 컨트롤러가 LoadBalancer 유형의 서비스로 배포되었는지 확인합니다.

```
kubectl get services
```

```
NAME                                 TYPE           CLUSTER-IP     EXTERNAL-IP
PORT(S)                      AGE
ingress-nginx-controller             LoadBalancer   10.16.18.20    10.19.14.76   80:30635/
TCP,443:30873/TCP   59m
ingress-nginx-controller-admission   ClusterIP      10.87.41.25    <none>
443/TCP                      59m
```

4 외부 IP 주소를 사용하여 로드 밸런서를 Ping합니다.

```
ping 10.19.14.76
```

```
Pinging 10.19.14.76 with 32 bytes of data:
Reply from 10.19.14.76: bytes=32 time<1ms TTL=62
Reply from 10.19.14.76: bytes=32 time=1ms TTL=62
```

5 Nginx 수신 컨트롤러가 실행 중인지 확인합니다.

```
kubectl get pods
```

```
NAME                                        READY   STATUS    RESTARTS   AGE
ingress-nginx-controller-7c6c46898c-v6blt   1/1     Running   0          76m
```

6 수신 규칙 및 경로를 사용하여 ingress-hello.yaml이라는 수신 리소스를 생성합니다.

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-hello
spec:
  rules:
  - http:
      paths:
      - path: /hello
        backend:
          serviceName: hello
          servicePort: 80
```

7 ingress-hello 리소스를 배포합니다.

```
kubectl apply -f ingress-hello.yaml
```

```
ingress.networking.k8s.io/ingress-hello created
```

8 수신 리소스가 배포되었는지 확인합니다.

IP 주소는 수신 컨트롤러의 외부 IP에 매핑됩니다.

```
kubectl get ingress
```

```
NAME            CLASS    HOSTS   ADDRESS         PORTS   AGE
ingress-hello   <none>   *       10.19.14.76     80      51m
```

9 ingress-hello-test.yaml이라는 hello 테스트 애플리케이션 및 서비스를 생성합니다.

```
kind: Service
apiVersion: v1
metadata:
  name: hello
spec:
  selector:
    app: hello
    tier: backend
  ports:
  - protocol: TCP
```

```
    port: 80
    targetPort: http
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello
spec:
  replicas: 3
  selector:
    matchLabels:
      app: hello
      tier: backend
      track: stable
  template:
    metadata:
      labels:
        app: hello
        tier: backend
        track: stable
    spec:
      containers:
        - name: hello
          image: "gcr.io/google-samples/hello-go-gke:1.0"
          ports:
            - name: http
              containerPort: 80
```

10 ingress-hello-test 리소스를 배포합니다.

```
kubectl apply -f ingress-hello-test.yaml
```

```
service/hello created
deployment.apps/hello created
```

11 hello 배포를 사용할 수 있는지 확인합니다.

```
kubectl get deployments
```

```
NAME                       READY   UP-TO-DATE   AVAILABLE   AGE
hello                      3/3     3            3           4m59s
ingress-nginx-controller   1/1     1            1           3h39m
```

12 Nginx 수신 컨트롤러에서 사용하는 로드 밸런서의 공용 IP 주소를 가져옵니다.

```
kubectl get ingress
```

```
NAME            CLASS    HOSTS   ADDRESS         PORTS   AGE
ingress-hello   <none>   *       10.19.14.76     80      13m
```

13 브라우저를 사용하여 공용 IP로 이동하고 수신 경로를 포함합니다.

```
http://10.19.14.76/hello
```

"hello" 메시지가 반환됩니다.

```
{"message":"Hello"}
```

결과

클러스터 내에서 실행되는 서비스가 앞에 있는 백엔드 앱은 로드 밸런서의 외부 IP 주소를 사용하여 수신 컨트롤러를 통해 외부에서 브라우저로 액세스합니다.

Tanzu Kubernetes 스토리지 클래스 예

지속성을 요구하는 워크로드의 경우 기본 스토리지 클래스를 사용하거나 영구 볼륨에 사용할 고유한 스토리지 클래스를 정의할 수 있습니다. Tanzu Kubernetes 클러스터는 CSI(컨테이너 스토리지 인터페이스) 프로비저너를 지원합니다.

CSI(컨테이너 스토리지 인터페이스)가 지원됨

Tanzu Kubernetes 클러스터는 CSI(컨테이너 스토리지 인터페이스)를 지원합니다. `StorageClass` 정의에서는 이 유형의 프로비저너가 `csi.vsphere.vware.com`으로 식별됩니다.

다음 YAML 정의를 템플릿으로 사용하여 Tanzu Kubernetes 클러스터에 대한 스토리지 클래스를 정의할 수 있습니다. 스토리지 클래스를 기본값("true")으로 할지 지정하고 스토리지 환경에 대한 데이터스토어 URL을 제공합니다.

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: tkgs-storage-class
  annotations:
    storageclass.kubernetes.io/is-default-class: "true" or "false"
provisioner: csi.vsphere.vmware.com
parameters:
  datastoreurl: "ds:///vmfs/volumes/vsan:52d8eb4842dbf493-41523be9cd4ff7b7/"
```

스토리지 클래스를 생성합니다.

```
kubectl apply -f tkgs-storage-class.yaml

storageclass.storage.k8s.io/tkgs-storage-class created
```

스토리지 클래스가 생성되었는지 확인합니다.

```
kubectl get storageclass
```

또는 바로 가기를 사용합니다.

```
kubectl get sc
```

vCP(VMware Cloud Provider)가 지원되지 않음

Tanzu Kubernetes 클러스터는 아래와 같이 레거시 vCP(VMware Cloud Provider) `StorageClass`를 지원하지 않습니다. vCP 프로비저너를 사용하여 `StorageClass`를 생성하려고 하면 `StorageClass`가 생성되지 않습니다.

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: demo-sts-sc
provisioner: kubernetes.io/vsphere-volume
parameters:
  diskformat: thin
```

Tanzu Kubernetes 영구 볼륨 할당 예

Tanzu Kubernetes 클러스터에서 상태 저장 워크로드를 실행하려면 기본 스토리지 인프라의 세부 정보를 모르는 상태에서 영구 스토리지 리소스를 요청하는 PVC(영구 볼륨 할당)를 생성할 수 있습니다. PVC에 사용되는 스토리지는 vSphere 네임스페이스에 대한 스토리지 할당량을 넘어 할당됩니다.

컨테이너는 기본적으로 사용 후 삭제 및 상태 비저장입니다. 상태 저장 워크로드의 경우 일반적인 방식은 PVC(영구 볼륨 할당)를 생성하는 것입니다. PVC를 사용하여 영구 볼륨을 마운트하고 스토리지에 액세스할 수 있습니다. 이 요청은 영구 볼륨 개체와 일치하는 가상 디스크를 동적으로 프로비저닝합니다. 할당은 영구 볼륨에 바인딩됩니다. 할당을 삭제하면 해당하는 영구 볼륨 개체와 프로비저닝된 가상 디스크도 삭제됩니다.

절차

1 대상 Tanzu Kubernetes 클러스터에 로그인합니다. vCenter Single Sign-On 사용자로 Tanzu Kubernetes 클러스터에 연결의 내용을 참조하십시오.

2 클러스터가 실행되고 있는 네임스페이스로 전환합니다.

```
kubectl config use-context NAMESPACE
```

3 스토리지 클래스를 확인하거나 새로 생성합니다.

기존 스토리지 클래스를 확인하려면:

```
kubectl get storageclass
```

스토리지 클래스를 생성하려면 Tanzu Kubernetes 스토리지 클래스 예 항목을 참조하십시오.

4 네임스페이스를 생성합니다.

```
kubectl create namespace guestbook
```

5 방명록 PVC YAML 파일을 생성합니다.

- Redis 리더 PVC
- Redis 팔로워 PVC

6 방명록 PVC를 클러스터에 적용합니다.

```
kubectl apply -f redis-leader-pvc.yaml -n guestbook
```

```
kubectl apply -f redis-follower-pvc.yaml -n guestbook
```

7 PVC의 상태를 확인합니다.

```
kubectl get pvc,pv -n guestbook
```

PVC 및 PV(영구 볼륨)가 나열되고 사용할 수 있습니다.

```
NAME                                        STATUS
VOLUME                                      CAPACITY   ACCESS MODES   STORAGECLASS
AGE
persistentvolumeclaim/redis-follower-pvc    Bound    pvc-37b72f35-3de2-4f84-
be7d-50d5dd968f62   2Gi        RWO            tkgs-storage-class   66s
persistentvolumeclaim/redis-leader-pvc      Bound    pvc-2ef51f31-dd4b-4fe2-bf4c-
f0149cb4f3da   2Gi        RWO            tkgs-storage-class   66s

NAME                                                CAPACITY   ACCESS MODES   RECLAIM
POLICY STATUS   CLAIM                 STORAGECLASS
persistentvolume/pvc-2ef51f31-dd4b-4fe2-bf4c   2Gi  RWO            Delete
Bound    guestbook/redis-leader-pvc     tkgs-storage-class
persistentvolume/pvc-37b72f35-3de2-4f84-be7d   2Gi  RWO            Delete
Bound    guestbook/redis-follower-pvc   tkgs-storage-class
```

Tanzu Kubernetes 방명록 자습서

Tanzu Kubernetes 클러스터에 방명록 애플리케이션을 배포하여 서비스 계정, 배포 및 서비스 생성에 대한 포드 보안 정책을 탐색합니다.

방명록 애플리케이션을 배포하는 것은 Kubernetes를 살펴볼 수 있는 일반적인 방법입니다. Tanzu Kubernetes Grid 서비스에서 프로비저닝된 Tanzu Kubernetes 클러스터에 모든 방명록 YAML 파일을 배포하면 애플리케이션 포드가 생성되지 않습니다. `kubectl describe pod` 명령을 실행하면 다음 오류 메시지가 나타납니다.

```
"Error: container has runAsNonRoot and image will run as root"
```

방명록 애플리케이션에서는 `deployment` 및 `replicaset` 리소스를 모두 사용하여 기본 네임스페이스에 권한 있는 컨테이너를 배포합니다. Tanzu Kubernetes 클러스터에 대해 PodSecurityPolicy 컨트롤러가 사용되도록 설정되어 있으므로 클러스터 사용자가 방명록 애플리케이션 포드를 생성하려고 하면 이러한 컨트롤러의 서비스 계정이 PodSecurityPolicy에 기반하여 확인됩니다. 적절한 PSP가 이러한 서비스 계정에 바인딩되지 않으면 해당 애플리케이션이 배포되지 않습니다.

기본적으로 Tanzu Kubernetes 관리자는 사용자 계정을 사용하여 네임스페이스에 권한 있는 포드를 직접 생성할 수 있습니다. 그러나 방명록 애플리케이션에서는 서비스 계정을 사용하여 권한 있는 컨테이너를 배포합니다. 클러스터 관리자는 `kube-system` 네임스페이스에 Deployment, StatefulSet 및 DaemonSet를 생성할 수 있습니다. 그러나 방명록 애플리케이션에서는 이러한 리소스를 기본 네임스페이스에 배포합니다. 또한 관리자가 아닌 사용자는 적절한 PSP 및 바인딩 없이는 권한 있는 또는 권한 없는 포드를 전혀 생성할 수 없습니다.

한 가지 해결 방법은 기본 권한 있는 PSP에 대한 바인딩을 생성하여 방명록 애플리케이션 배포를 허용하는 것입니다. 권한 있는 PodSecurityPolicy는 바인딩된 계정에 대해 루트로 실행 포드 및 권한 있는 컨테이너를 허용합니다. 클러스터 전체에 `vmware-system-privileged`를 적용하는 ClusterRoleBinding을 생성할 수 있지만 이렇게 하면 필요한 것보다 많은 권한을 부여하여 최소 권한의 원칙을 위반할 수 있습니다. 더 나은 접근 방법은 시스템 서비스 계정이 기본 네임스페이스에서 권한 있는 PodSecurityPolicy를 사용할 수 있도록 하는 RoleBinding을 생성하는 것입니다. 자세한 내용은 포드 보안 정책에 대한 역할 바인딩 예의 내용을 참조하십시오.

사전 요구 사항

다음 항목을 검토하십시오.

- Kubernetes 설명서의 방명록 애플리케이션 자습서
- Tanzu Kubernetes 클러스터에서 포드 보안 정책 사용
- 포드 보안 정책에 대한 역할 바인딩 예

절차

1 Tanzu Kubernetes 클러스터에 로그인합니다. vCenter Single Sign-On 사용자로 Tanzu Kubernetes 클러스터에 연결의 내용을 참조하십시오.

2 방명록 네임스페이스를 생성합니다.

```
kubectl create namespace guestbook
```

확인:

```
kubectl get ns
```

3 기본 권한이 있는 PSP를 사용하여 역할 기반 액세스 제어를 생성합니다.

```
kubectl create clusterrolebinding default-tkg-admin-privileged-binding --
clusterrole=psp:vmware-system-privileged --group=system:authenticated
```

참고 보안을 강화해야 하는 경우에는 방명록 네임스페이스에 RoleBinding을 적용합니다. 포드 보안 정책에 대한 역할 바인딩 예 항목을 참조하십시오.

4 스토리지 클래스를 확인하거나 새로 생성합니다.

기존 스토리지 클래스를 확인하려면:

```
kubectl get storageclass
```

스토리지 클래스를 생성하려면 Tanzu Kubernetes 스토리지 클래스 예 항목을 참조하십시오.

5 스토리지 클래스를 사용하는 PVC(영구 볼륨 할당)을 생성합니다.

다음 YAML 파일을 사용합니다.

- Redis 리더 PVC
- Redis 팔로워 PVC

PVC를 생성하려면 Tanzu Kubernetes 영구 볼륨 할당 예 항목을 참조하십시오.

6 방명록 YAML 파일을 생성합니다.

다음 YAML 파일을 사용합니다.

- Redis 리더 배포
- Redis 리더 서비스
- Redis 팔로워 배포
- Redis 팔로워 서비스
- 방명록 프런트 엔드 배포
- 방명록 프런트 엔드 서비스

7 네임스페이스에 방명록 애플리케이션을 배포합니다.

```
kubectl apply -f . --namespace guestbook
```

8 방명록 리소스의 생성을 확인합니다.

```
kubectl get all -n guestbook
```

```
NAME                                                READY   STATUS
RESTARTS   AGE
pod/guestbook-frontend-deployment-56fc5b6b47-cd58r   1/1     Running
0          65s
```

```
pod/guestbook-frontend-deployment-56fc5b6b47-fh6dp   1/1     Running
0          65s
pod/guestbook-frontend-deployment-56fc5b6b47-hgd2b   1/1     Running
0          65s
pod/redis-follower-deployment-6fc9cf5759-99fgw       1/1     Running
0          65s
pod/redis-follower-deployment-6fc9cf5759-rhxf7       1/1     Running
0          65s
pod/redis-leader-deployment-7d89bbdbcf-flt4q         1/1     Running
0          65s

NAME                                 TYPE           CLUSTER-IP       EXTERNAL-IP
PORT(S)        AGE
service/guestbook-frontend           LoadBalancer   10.10.89.59      10.19.15.99
80:31513/TCP   65s
service/redis-follower               ClusterIP      10.111.163.189   <none>
6379/TCP       65s
service/redis-leader                 ClusterIP      10.111.70.189    <none>
6379/TCP       65s

NAME                                          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/guestbook-frontend-deployment   3/3     3            3           65s
deployment.apps/redis-follower-deployment       1/2     2            1           65s
deployment.apps/redis-leader-deployment         1/1     1            1           65s

NAME                                                    DESIRED   CURRENT   READY   AGE
replicaset.apps/guestbook-frontend-deployment-56fc5b6b47   3         3         3       65s
replicaset.apps/redis-follower-deployment-6fc9cf5759       2         2         1       65s
replicaset.apps/redis-leader-deployment-7d89bbdbcf         1         1         1       65s
```

9 `service/guestbook-frontend` 로드 밸런서의 `External-IP` 주소(이 예에서는 `10.19.15.99`)를 사용하여 방명록 웹 페이지에 액세스합니다.

방명록 웹 인터페이스가 표시되고 방명록 데이터베이스에 값을 입력할 수 있습니다. 애플리케이션을 다시 시작하면 데이터가 유지됩니다.

방명록 예제 YAML 파일

예제 YAML 파일을 사용하여 영구 데이터로 방명록 애플리케이션을 배포합니다.

Redis 리더 PVC

`redis-leader-pvc.yaml` 파일은 명명된 스토리지 클래스를 참조하는 영구 볼륨 클레임의 예입니다. 이 예제를 사용하려면 스토리지 클래스의 이름을 입력합니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: redis-leader-pvc
spec:
  accessModes:
    - ReadWriteOnce
```

```
  storageClassName: tkgs-storage-class-name
  resources:
    requests:
      storage: 2Gi
```

Redis 팔로워 PVC

redis-follower-pvc.yaml 파일은 명명된 스토리지 클래스를 참조하는 영구 볼륨 클레임의 예입니다. 이 예제를 사용하려면 스토리지 클래스의 이름을 입력합니다.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: redis-follower-pvc
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: tkgs-storage-class-name
  resources:
    requests:
      storage: 2Gi
```

Redis 리더 배포

redis-leader-deployment.yaml 파일은 영구 볼륨이 있는 Redis 리더 배포의 예입니다.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: redis-leader-deployment
spec:
  selector:
    matchLabels:
      app: redis
      role: leader
      tier: backend
  replicas: 1
  template:
    metadata:
      labels:
        app: redis
        role: leader
        tier: backend
    spec:
      containers:
      - name: leader
        image: redis:6.0.5
        resources:
          requests:
            cpu: 100m
            memory: 100Mi
        ports:
        - containerPort: 6379
        volumeMounts:
```

```
        - name: redis-leader-data
          mountPath: /data
      volumes:
      - name: redis-leader-data
        persistentVolumeClaim:
          claimName: redis-leader-pvc
```

Redis 팔로워 배포

redis-follower-deployment.yaml 파일은 영구 볼륨이 있는 Redis 팔로워 배포의 예입니다.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: redis-follower-deployment
  labels:
    app: redis
spec:
  selector:
    matchLabels:
      app: redis
      role: follower
      tier: backend
  replicas: 1
  template:
    metadata:
      labels:
        app: redis
        role: follower
        tier: backend
    spec:
      containers:
      - name: follower
        image: gcr.io/google_samples/gb-redis-follower:v2
        resources:
          requests:
            cpu: 100m
            memory: 100Mi
        env:
        - name: GET_HOSTS_FROM
          value: dns
        ports:
        - containerPort: 6379
        volumeMounts:
        - name: redis-follower-data
          mountPath: /data
      volumes:
      - name: redis-follower-data
        persistentVolumeClaim:
          claimName: redis-follower-pvc
```

Redis 리더 서비스

redis-leader-service.yaml 파일은 Redis 리더 서비스의 예입니다.

```
apiVersion: v1
kind: Service
metadata:
  name: redis-leader
  labels:
    app: redis
    role: leader
    tier: backend
spec:
  ports:
  - port: 6379
    targetPort: 6379
  selector:
    app: redis
    role: leader
    tier: backend
```

Redis 팔로워 서비스

redis-follower-service.yaml 파일은 Redis 팔로워 서비스의 예입니다.

```
apiVersion: v1
kind: Service
metadata:
  name: redis-follower
  labels:
    app: redis
    role: follower
    tier: backend
spec:
  ports:
  - port: 6379
  selector:
    app: redis
    role: follower
    tier: backend
```

방명록 프런트 엔드 배포

guestbook-frontend-deployment.yaml 파일은 방명록 프런트 엔드 배포의 예입니다.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: guestbook-frontend-deployment
spec:
  selector:
    matchLabels:
      app: guestbook
      tier: frontend
```

```
  replicas: 3
  template:
    metadata:
      labels:
        app: guestbook
        tier: frontend
    spec:
      containers:
      - name: php-redis
        image: gcr.io/google_samples/gb-frontend:v5
        resources:
          requests:
            cpu: 100m
            memory: 100Mi
        env:
        - name: GET_HOSTS_FROM
          value: dns
        ports:
        - containerPort: 80
```

방명록 프런트 엔드 서비스

`guestbook-frontend-service.yaml` 파일은 방명록 프런트 엔드 로드 밸런서 서비스의 예입니다.

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

Tanzu Kubernetes 클러스터에서 포드 보안 정책 사용

Tanzu Kubernetes Grid 서비스는 PodSecurityPolicy 승인 컨트롤러가 사용되도록 설정된 Tanzu Kubernetes 클러스터를 프로비저닝합니다. 즉, 워크로드를 배포하려면 포드 보안 정책이 필요합니다. 클러스터 관리자는 사용자 계정에서 모든 네임스페이스로 그리고 서비스 계정에서 kube-system 네임스페이스로 포드를 배포할 수 있습니다. 다른 모든 사용 사례의 경우 PodSecurityPolicy 개체에 명시적으로 바인딩해야 합니다. 클러스터에 바인딩할 수 있는 기본 포드 보안 정책이 포함되거나 고유한 정책을 생성합니다.

Kubernetes 포드 보안 정책 정보

Kubernetes PSP(포드 보안 정책)는 포드의 보안을 제어하는 클러스터 수준 리소스입니다. PSP를 사용하면 배포할 수 있는 포드 유형 및 배포 가능한 계정 유형을 제어할 수 있습니다.

PodSecurityPolicy 리소스는 포드를 배포하기 위해 포드가 충족해야 하는 조건 집합을 정의합니다. 조건이 충족되지 않으면 포드를 배포할 수 없습니다. 단일 PodSecurityPolicy가 포드를 전체적으로 검증해야 합니다. 두 개의 서로 다른 정책에서 포드의 규칙을 나누어 가질 수 없습니다.

Kubernetes에서 포드 보안 정책의 사용을 구현하는 방법에는 여러 가지가 있습니다. 일반적인 방식은 RBAC(역할 기반 액세스 제어) 개체를 사용하는 것입니다. ClusterRole 및 ClusterRoleBinding은 클러스터 전체에 적용됩니다. 역할 및 RoleBinding은 특정 네임스페이스에 적용됩니다. RoleBinding이 사용되는 경우에는 포드를 바인딩과 동일한 네임스페이스에만 실행할 수 있습니다.

Kubernetes 포드를 생성하는 방법에는 직접과 간접의 두 가지가 있습니다. 사용자 계정을 사용하여 포드 규격을 배포하여 포드를 직접 생성합니다. 배포 또는 DaemonSet 같은 더 높은 수준의 리소스를 정의하여 포드를 간접적으로 생성합니다. 이 경우 서비스 계정은 기본 포드를 생성합니다.

PSP를 효과적으로 사용하려면 포드 생성 워크플로를 둘 다 고려해야 합니다. 사용자가 포드를 직접 생성하면 사용자 계정에 바인딩된 PSP가 작업을 제어합니다. 사용자가 서비스 계정을 통해 포드를 생성하는 경우 포드를 생성하는 데 사용되는 서비스 계정에 PSP를 바인딩해야 합니다. 포드 규격에 서비스 계정이 지정되지 않은 경우 네임스페이스의 기본 서비스 계정이 사용됩니다.

자세한 내용은 Kubernetes 설명서의 포드 보안 정책, RBAC 및 서비스 계정을 참조하십시오.

Tanzu Kubernetes 클러스터에 대한 기본 PodSecurityPolicy

이 표에는 Tanzu Kubernetes 클러스터에 대한 권한 있는 및 제한된 기본 포드 보안 정책과 각 정책과 연결된 기본 ClusterRole이 나열 및 설명되어 있습니다.

표 14-1. 연결된 ClusterRole이 있는 기본 PodSecurityPolicy

기본 PSP	사용 권한	설명	연결된 기본 ClusterRole
`vmware-system-privileged`	**임의로 실행**	허용 PSP. PSP 승인 컨트롤러를 사용하도록 설정하지 않고 클러스터를 실행하는 것과 같습니다.	`psp:vmware-system-privileged`가 이 PSP를 사용할 수 있음
`vmware-system-restricted`	**비루트로 실행해야 함**	제한 PSP. 포드 컨테이너에 대한 권한 있는 액세스를 허용하지 않으며, 루트에 대한 가능한 에스컬레이션을 차단하며, 여러 보안 메커니즘을 사용해야 합니다.	`psp:vmware-system-restricted`가 이 PSP를 사용할 수 있음

Tanzu Kubernetes 클러스터에 대한 기본 바인딩 없음

Tanzu Kubernetes Grid 서비스는 Tanzu Kubernetes 클러스터에 대한 기본 RoleBinding 및 ClusterRoleBinding을 제공하지 않습니다.

vSphere 네임스페이스에 대한 **편집** 권한을 부여받은 vCenter Single Sign-On 사용자는 해당 네임스페이스에 배포된 모든 Tanzu Kubernetes 클러스터에 대한 **cluster-admin** 역할에 할당됩니다. 인증된 클러스터 관리자는 `vmware-system-privileged` PSP를 암시적으로 사용할 수 있습니다. 기술적으로 ClusterRoleBinding은 아니지만 동일한 효과를 갖습니다.

클러스터 관리자는 사용자가 클러스터에 배포할 수 있는 포드 유형을 허용하거나 제한하기 위한 바인딩을 정의해야 합니다. RoleBinding이 사용되는 경우 바인딩은 포드가 바인딩과 동일한 네임스페이스에서만 실행되도록 허용합니다. 이는 네임스페이스에서 실행되는 모든 포드에 대한 액세스 권한을 부여할 시스템 그룹과 연결될 수 있습니다. 클러스터에 인증하는 관리자가 아닌 사용자는 `authenticated` 역할에 할당되며, 이처럼 기본 PSP에 바인딩될 수 있습니다. 개발자에게 Tanzu Kubernetes 클러스터에 대한 액세스 권한 부여의 내용을 참조하십시오.

Tanzu Kubernetes 클러스터에 대한 기본 PodSecurityPolicy 효과

Tanzu Kubernetes 클러스터에는 다음과 같은 동작이 적용됩니다.

- 클러스터 관리자는 자신의 사용자 계정을 사용하여 모든 네임스페이스에 권한 있는 포드를 직접 생성할 수 있습니다.
- 클러스터 관리자는 kube-system 네임스페이스에서 배포, StatefulSets 및 DaemonSet(각각 권한 있는 포드 생성)를 생성할 수 있습니다. 다른 네임스페이스를 사용하려면 Tanzu Kubernetes 방명록 자습서의 내용을 참조하십시오.
- 클러스터 관리자는 자체 PSP(두 개의 기본 PSP 외)를 생성하고 이러한 PSP를 모든 사용자에게 바인딩할 수 있습니다. 자체 PSP를 정의하는 경우 Kubernetes 설명서에서 정책 순서를 참조하십시오.
- 클러스터 관리자가 PSP를 인증된 사용자에게 바인딩할 때까지 인증된 사용자가 권한 있거나 권한 없는 포드를 생성할 수 없습니다. Tanzu Kubernetes 방명록 자습서의 내용을 참조하십시오.

포드 보안 정책에 대한 역할 바인딩 예

Tanzu Kubernetes 클러스터에는 권한 있는 워크로드 및 제한된 워크로드 배포를 위해 바인딩할 수 있는 기본 PodSecurityPolicy가 포함됩니다.

기본 포드 보안 정책 정보

이 섹션에서는 ClusterRoleBinding 및 RoleBinding을 포함하여 기본 포드 보안 정책에 대한 역할 바인딩 개체를 생성하기 위한 YAML 및 CLI 명령을 제공합니다. 자세한 내용은 Tanzu Kubernetes 클러스터에서 포드 보안 정책 사용의 내용을 참조하십시오.

RoleBinding은 특정 네임스페이스 내에서 사용 권한을 부여하지만 ClusterRoleBinding은 클러스터 전체에서 사용 권한을 부여합니다. RoleBinding을 사용할지 아니면 ClusterRoleBinding을 사용할지는 사용 사례에 따라 다릅니다. 예를 들어 ClusterRoleBinding을 사용하고 `system:serviceaccounts:<namespace>`를 사용하도록 주체를 구성하는 경우 네임스페이스가 생성되기 전에 PSP에 바인딩할 수 있습니다. 자세한 내용은 Kubernetes 설명서에서 RoleBinding 및 ClusterRoleBinding을 참조하십시오.

예 1: 권한 있는 워크로드 집합을 실행하는 ClusterRoleBinding

다음 kubectl 명령은 기본 PSP `vmware-system-privileged`를 사용하여 권한 있는 워크로드 집합을 실행하도록 인증된 사용자에게 액세스 권한을 부여하는 ClusterRoleBinding을 생성합니다.

경고 예 1을 선언적으로 또는 명령적으로 적용하면 권한 있는 워크로드를 클러스터 전체에 배포할 수 있습니다. 실제로 예 1은 네이티브 보안 제어를 사용하지 않도록 설정하므로 영향을 충분히 인식하고 주의를 기울여서 사용해야 합니다. 보안을 강화하려면 예 2, 3, 4를 고려하십시오.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: psp:privileged
rules:
- apiGroups: ['policy']
  resources: ['podsecuritypolicies']
  verbs:     ['use']
  resourceNames:
  - vmware-system-privileged
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: all:psp:privileged
roleRef:
  kind: ClusterRole
  name: psp:privileged
  apiGroup: rbac.authorization.k8s.io
subjects:
- kind: Group
  name: system:serviceaccounts
  apiGroup: rbac.authorization.k8s.io
```

YAML을 적용하는 대신 다음 `kubectl` 명령을 실행할 수 있습니다.

```
kubectl create clusterrolebinding default-tkg-admin-privileged-binding --
clusterrole=psp:vmware-system-privileged --group=system:authenticated
```

예 2: 권한 있는 워크로드 집합을 실행하는 RoleBinding

다음 kubectl 명령은 기본 PSP `vmware-system-privileged`를 사용하여 권한 있는 워크로드 집합을 실행하도록 기본 네임스페이스 내의 모든 서비스 계정에 액세스 권한을 부여하는 RoleBinding을 생성합니다.

```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: rolebinding-default-privileged-sa-ns_default
  namespace: default
roleRef:
  kind: ClusterRole
```

```
  name: psp:vmware-system-privileged
  apiGroup: rbac.authorization.k8s.io
subjects:
- kind: Group
  apiGroup: rbac.authorization.k8s.io
  name: system:serviceaccounts
```

YAML을 적용하는 대신 다음 kubectl 명령을 실행할 수 있습니다.

```
kubectl create rolebinding rolebinding-default-privileged-sa-ns_default --namespace=default --
clusterrole=psp:vmware-system-privileged --group=system:serviceaccounts
```

예 3: 제한된 워크로드 집합을 실행하는 ClusterRoleBinding

다음 YAML은 기본 PSP `vmware-system-restricted`를 사용하여 제한된 워크로드 집합을 실행하도록 인증된 사용자에게 클러스터 전체 액세스 권한을 부여하는 ClusterRoleBinding을 생성합니다.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: psp:authenticated
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: system:authenticated
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: psp:vmware-system-restricted
```

YAML을 적용하는 대신 다음 kubectl 명령을 실행할 수 있습니다.

```
kubectl create clusterrolebinding psp:authenticated --clusterrole=psp:vmware-system-
restricted --group=system:authenticated
```

예 4: 제한된 워크로드 집합을 실행하는 RoleBinding

다음 YAML은 기본 PSP `vmware-system-restricted`를 사용하여 제한된 워크로드 집합을 실행하도록 특정 네임스페이스 내의 모든 서비스 계정에 액세스 권한을 부여하는 RoleBinding을 생성합니다.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: psp:serviceaccounts
  namespace: some-namespace
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: system:serviceaccounts
```

```
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: psp:vmware-system-restricted
```

YAML을 적용하는 대신 다음 kubectl 명령을 실행할 수 있습니다.

```
kubectl create rolebinding psp:serviceaccounts --clusterrole=psp:vmware-system-restricted --
group=system:serviceaccounts
```

포드 보안 정책에 대한 역할 예

Tanzu Kubernetes 클러스터에는 워크로드를 배포하기 위해 PSP(포드 보안 정책)가 필요합니다. 자체 PSP를 정의하는 경우에는 PSP를 참조하는 역할 또는 ClusterRole을 생성해야 합니다.

PodSecurityPolicy에 대한 역할 예

다음 예는 PodSecurityPolicy에 바인딩된 역할을 보여줍니다. 역할 정의에서 `example-role`에는 직접 정의한 사용자 지정 PSP 리소스에 대한 `use` 동사가 부여됩니다. 또는 기본 PSP 중 하나를 사용합니다. 그런 다음 Tanzu Kubernetes 방명록 자습서을 생성합니다.

```
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: Role
metadata:
  name: example-role
  namespace: tkgs-cluster-ns
rules:
- apiGroups:
  - ""
  resources:
  - configmaps
  verbs:
  - create
  - get
  - list
  - watch
  - update
- apiGroups:
  - ""
  resources:
  - events
  verbs:
  - create
  - update
  - patch
- apiGroups:
  - extensions
  resourceNames:
  - CUSTOM-OR-DEFAULT-PSP
  resources:
  - podsecuritypolicies
```

```
verbs:
- use
```

Tanzu Kubernetes 클러스터에 TKG 패키지 배포

TKGS에서 프로비저닝된 Tanzu Kubernetes 클러스터에 TKG 패키지를 배포하려면 여기에서 일련의 지침을 참조하십시오.

TKG 1.6 패키지

TKGS에서 프로비저닝된 Tanzu Kubernetes 클러스터에 TKG 1.6 패키지를 설치하려면 TKG 1.6 패키지 설치 및 구성을 참조하십시오.

TKG 1.4 패키지

TKGS에서 프로비저닝된 Tanzu Kubernetes 클러스터에 TKG 1.4 패키지를 설치하려면 TKG 1.4 패키지 설치 및 구성을 참조하십시오.

TKG 1.3.1 확장

TKGS에서 프로비저닝된 Tanzu Kubernetes 클러스터에 TKG 1.3.1 확장을 설치하려면 이 섹션의 설명서를 참조하십시오.

참고 TKG 패키지는 TKG 확장을 대체합니다. 확장 대신 패키지를 사용하는 것이 좋습니다.

TKG 확장 v1.3.1 번들 다운로드

하나 이상의 TKG 확장 배포를 준비하려면 VMware에서 TKG 확장 v1.3.1 번들을 다운로드합니다.

TKG 확장은 VMware에서 다운로드하는 별도의 번들로 패키징됩니다.

참고 vSphere with Tanzu는 vSphere 7 U2 이상에서 TKG 확장 v1.3.1을 지원합니다.

절차

1 https://www.vmware.com/go/get-tkg로 이동합니다.

2 My VMware 자격 증명을 사용하여 로그인합니다.

3 **제품 다운로드 > 다운로드로 이동**을 선택합니다.

4 드롭다운 메뉴에서 버전 **1.3.1**을 선택합니다.

5 스크롤하여 **VMware Tanzu Kubernetes Grid Extensions Manifest 1.3.1**을 찾습니다.

6 **지금 다운로드**를 클릭하여 로컬 시스템에 `tkg-extensions-manifests-v1.3.1-vmware.1.tar.gz` 파일을 다운로드합니다.

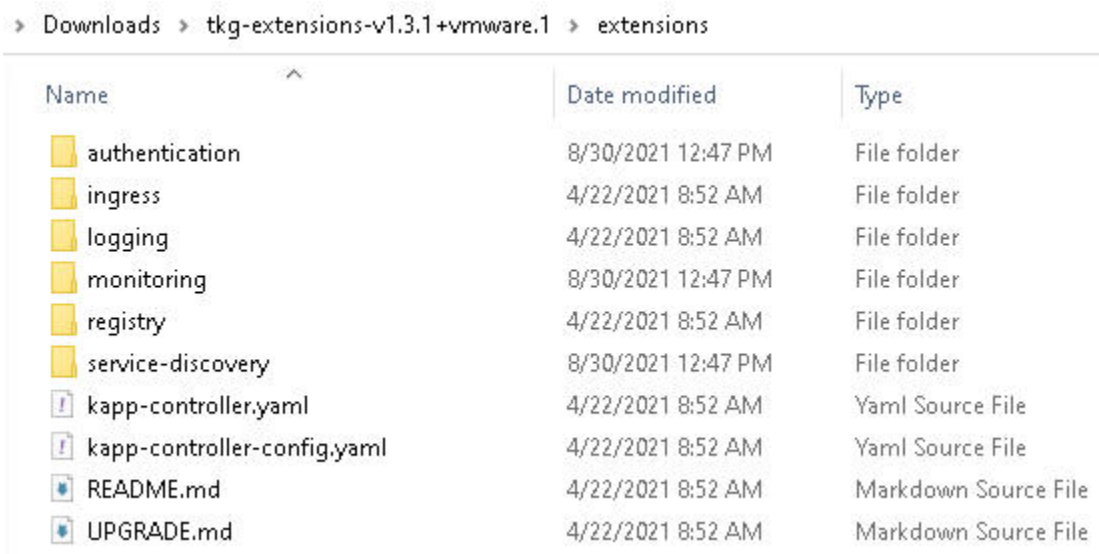
7 `kubectl` 명령을 실행하는 컴퓨터에 `tkg-extensions-manifests-v1.3.1-vmware.1.tar.gz` 파일을 복사합니다.

8 tar 명령 또는 선택한 추출 도구를 사용하여 TKG 확장 파일을 추출합니다.

```
tar -xzf tkg-extensions-manifests-v1.3.1-vmware.1.tar.gz
```

9 다음 파일 경로로 이동하여 TKG 확장 파일을 확인합니다.

```
cd /tkg-extensions-v1.3.1+vmware.1/extensions
```



여기는 TKG 확장의 홈 디렉토리입니다.

TKG 확장 사전 요구 사항 설치

하나 이상의 TKG 확장 v1.3.1을 설치할 각 Tanzu Kubernetes 클러스터에 사전 요구 사항 애플리케이션을 설치합니다.

TKG 확장 v1.3.1에는 Kapp Controller와 Cert Manager라는 두 가지 필수 구성 요소가 필요합니다.

참고 Cert Manager의 대안으로 자체 TLS 인증서를 사용할 수 있습니다. TKG 확장에 자체 TLS 인증서 사용의 내용을 참조하십시오.

사전 요구 사항

- Tanzu Kubernetes 클러스터를 프로비저닝합니다. TKGS v1alpha2 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하는 워크플로의 내용을 참조하십시오.
- Tanzu Kubernetes 클러스터에 연결합니다. vCenter Single Sign-On 사용자로 Tanzu Kubernetes 클러스터에 연결의 내용을 참조하십시오.

절차

1 TKG 확장 v1.3.1 번들 다운로드.

2 클러스터에 Cert Manager를 설치합니다.

다운로드하고 추출한 TKG 확장 번들의 루트 디렉토리로 이동합니다.

```
cd /tkg-extensions-v1.3.1+vmware.1
```

Cert Manager에는 몇 가지 구성 요소가 포함되어 있습니다. `/cert-manager`라는 디렉토리에 세 개의 YAML 파일이 있습니다. `ls`를 사용하여 이 디렉토리가 있는지 확인합니다.

다음 단일 명령을 실행하여 모든 Cert Manager 구성 요소를 설치합니다.

```
kubectl apply -f cert-manager/
```

이 작업을 수행하면 `cert-manager` 네임스페이스, 구성 요소, 인증서 및 연결된 개체가 생성됩니다.

3 클러스터에 Kapp Controller를 설치합니다.

`kapp-controller.yaml`을 사용하여 Kapp 컨트롤러를 설치합니다. 필요한 경우 `kapp-controller-config.yaml`을 사용하여 Kapp 컨트롤러 구성을 사용자 지정할 수 있습니다.

TKG 확장의 홈 디렉토리로 이동합니다.

```
cd /tkg-extensions-v1.3.1+vmware.1/extensions
```

`ls`를 사용하여 Kapp 컨트롤러 파일 `kapp-controller.yaml` 및 `kapp-controller-config.yaml`이 있는지 확인합니다.

Kapp 컨트롤러 컨테이너는 기본 구성 매개 변수를 사용하여 설치됩니다. 일반적으로 Kapp 컨트롤러는 구성을 사용자 지정하지 않고 설치할 수 있습니다. Kapp 컨트롤러를 사용자 지정해야 하는 경우 `kapp-controller-config.yaml`을 편집합니다. 예를 들어 프록시 뒤에 Kapp 컨트롤러를 배포하는 경우 이 파일을 편집해야 합니다.

필요한 경우 `kapp-controller-config.yaml` 파일을 편집합니다. 구성 파일을 편집하는 경우 파일을 저장하고 다음 명령을 사용하여 변경 내용을 적용합니다.

```
kubectl apply -f kapp-controller-config.yaml
```

Kapp 컨트롤러 컨테이너는 `kapp-controller.yaml`을 사용하여 설치됩니다. 이 YAML 파일의 `spec.containers.image` 경로는 공용 VMware 레지스트리를 가리킵니다. 에어갭 설치의 경우 개인 레지스트리를 가리키도록 이 경로를 업데이트하십시오.

다음 명령을 실행하여 Kapp Controller를 설치합니다.

```
kubectl apply -f kapp-controller.yaml
```

이 작업을 수행하면 `tkg-system` 네임스페이스, kapp-controller 애플리케이션 및 역할 개체가 생성됩니다.

4 Cert Manager 및 Kapp Controller의 설치를 확인합니다.

`kubectl get pods -A` 명령을 실행합니다. 각각이 실행 중인지 볼 수 있습니다.

```
cert-manager       cert-manager-cainjector-...    1/1     Running    0      7h54m
cert-manager       cert-manager-...               1/1     Running    0      7h54m
cert-manager       cert-manager-webhook-...       1/1     Running    0      7h54m
tkg-system         kapp-controller-...            1/1     Running    0      16m
```

TKG 확장에 대한 영구 스토리지 요구 사항 검토

모니터링 확장인 Prometheus 및 Grafana 모두 영구 스토리지가 필요합니다. 이러한 확장 배포를 준비하려면 대상 Tanzu Kubernetes 클러스터가 기본 스토리지 클래스로 구성되어 있는지 그리고 vSphere 네임스페이스에 충분한 스토리지가 있는지 확인해야 합니다.

TKG 확장에 대한 영구 스토리지 요구 사항

Prometheus 또는 Grafana 확장을 배포하는 Tanzu Kubernetes 클러스터는 기본 스토리지 클래스로 프로비저닝해야 합니다. Tanzu Kubernetes Grid 서비스 v1alpha1 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하는 예의 내용을 참조하십시오.

또는 확장을 배포할 때 영구 볼륨 할당을 사용하도록 확장을 구성할 수 있습니다. 이러한 방식에 대한 구성은 각 확장에 대한 배포 지침의 일부로 제공됩니다.

확장을 설치하는 클러스터가 프로비저닝된 vSphere 네임스페이스에 대한 스토리지 제한은 총 영구 볼륨 할당 크기보다 커야 합니다.

표 14-2. TKG 확장에 대한 기본 스토리지 요구 사항

구성 요소	TKG 확장	기본 스토리지 크기
Grafana	Grafana	8Gi
Prometheus 서버	Prometheus	8Gi
경고 관리자	Prometheus	8Gi
Harbor	Harbor 레지스트리	PVC에 따라 다름

vSphere 네임스페이스 스토리지 제한 조정

Tanzu Kubernetes 클러스터가 프로비저닝된 vSphere 네임스페이스에 대한 스토리지 제한을 조정하려면 다음을 수행합니다.

1 vSphere Client를 사용하여 vSphere with Tanzu를 사용하도록 설정된 vCenter Server에 로그인합니다.

2 대상 Tanzu Kubernetes 클러스터가 프로비저닝된 vSphere 네임스페이스를 선택합니다.

3 **구성 > 리소스 제한**을 선택합니다.

4 **편집**을 클릭합니다.

5 스토리지 제한이 Prometheus 및 Grafana 확장에 필요한 영구 볼륨 할당의 총 크기보다 크도록 **스토리지** 제한을 조정합니다.

TKG 확장에 자체 TLS 인증서 사용

TKG 확장에는 TLS 인증서가 필요합니다. 사전 요구 사항을 충족하도록 cert-manager를 설치하거나 자체 서명된 인증서를 사용할 수 있습니다. CA의 인증서도 지원됩니다.

TKG 확장에 자체 TLS 인증서 사용 정보

cert-manager 설치는 각 TKG 확장에 대한 배포 프로세스의 일부로 문서화되어 있습니다. 또는 자체 TLS 인증서를 사용할 수 있습니다.

참고 이 작업은 OpenSSL이 설치된 Linux 호스트를 사용한다고 가정합니다.

인증 기관 인증서 생성

운영 환경에서는 CA로부터 인증서를 확보해야 합니다. 개발 또는 테스트 환경에서는 자체 서명된 인증서를 생성할 수 있습니다. CA 인증서를 생성하려면 다음 지침을 완료하십시오.

1 CA 인증서 개인 키를 생성합니다.

```
openssl genrsa -out ca.key 4096
```

2 CA 인증서를 생성합니다.

다음 명령을 템플릿으로 사용합니다. 환경에 따라 `-subj` 옵션의 값을 업데이트합니다. FQDN을 사용하여 TKG 확장 호스트를 연결하는 경우 이 FQDN을 CN(일반 이름) 특성으로 지정해야 합니다.

```
openssl req -x509 -new -nodes -sha512 -days 3650 \
 -subj "/C=US/ST=PA/L=PA/O=example/OU=Personal/CN=tkg-extensions.system.tanzu" \
 -key ca.key \
 -out ca.crt
```

서버 인증서 생성

인증서에는 일반적으로 .crt 파일과 .key 파일이 포함되어 있습니다(예: `tls.crt` 및 `tls.key`).

1 개인 키를 생성합니다.

```
openssl genrsa -out tls.key 4096
```

2 CSR(인증서 서명 요청)을 생성합니다.

다음 명령을 템플릿으로 사용합니다. 환경에 따라 -subj 옵션의 값을 업데이트합니다. FQDN을 사용하여 TKG 확장 호스트를 연결하는 경우에는 이 FQDN을 CN(일반 이름) 특성으로 지정하고 키 및 CSR 파일 이름에 FQDN을 사용해야 합니다.

```
openssl req -sha512 -new \
    -subj "/C=US/ST=PA/L=PA/O=example/OU=Personal/CN=tkg-extensions.system.tanzu" \
    -key tls.key \
    -out tls.csr
```

3 x509 v3 확장 파일을 생성합니다.

다음 명령을 템플릿으로 사용합니다. SAN(주체 대체 이름) 및 x509 v3 확장 요구 사항을 준수하는 TKG 확장 호스트에 대한 인증서를 생성할 수 있도록 이 파일을 생성합니다.

```
cat > v3.ext <<-EOF
authorityKeyIdentifier=keyid,issuer
basicConstraints=CA:FALSE
keyUsage = digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment
extendedKeyUsage = serverAuth
subjectAltName = @alt_names

[alt_names]
DNS.1=tkg-extensions.system.tanzu
EOF
```

4 x509 v3 확장 파일을 사용하여 TKG 확장 호스트에 대한 인증서를 생성합니다.

```
openssl x509 -req -sha512 -days 3650 \
-extfile v3.ext \
-CA ca.crt -CAkey ca.key -CAcreateserial \
-in tls.csr \
-out tls.crt
```

5 다음 형식을 사용하여 ca.crt, tls.crt, tls.key 파일의 내용을 TKG-EXTENSION-data-values.yaml 파일에 복사합니다.

```
ingress:
  tlsCertificate:
    tls.crt: |
        -----BEGIN ...
```

6 설명에 따라 지원되는 TKG 확장 배포를 진행합니다.

Fluent Bit 로깅을 위한 TKG 확장 배포 및 관리

이 항목에서는 Fluent Bit용 TKG 확장 v1.3.1을 배포하는 방법을 설명합니다. Fluent Bit는 빠르고 가벼운 로그 프로세서 및 전달자이며, 다양한 소스에서 애플리케이션 데이터와 로그를 수집하고 통합하여 여러 대

상으로 보내는 데 사용할 수 있습니다. Fluent Bit용 TKG 확장을 배포하여 Tanzu Kubernetes 클러스터 로그를 수집하고 선택한 대상에 전달합니다.

확장 사전 요구 사항

Fluent Bit용 TKG 확장 v1.3.1을 배포하기 전에 다음 요구 사항을 준수합니다.

- 클러스터를 프로비저닝합니다. TKGS v1alpha2 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하는 워크플로의 내용을 참조하십시오.
- 클러스터에 연결합니다. vCenter Single Sign-On 사용자로 Tanzu Kubernetes 클러스터에 연결의 내용을 참조하십시오.
- kubectl을 실행하는 클라이언트 호스트에 TKG 확장 v1.3.1 번들 다운로드합니다.
- 대상 클러스터에 TKG 확장 사전 요구 사항 설치합니다.

Fluent Bit 확장 배포

Fluent Bit용 TKG 확장은 클러스터에 Fluent Bit 컨테이너를 설치합니다. 이 컨테이너에 대한 자세한 내용은 https://fluentbit.io/에서 참조하십시오.

컨테이너	리소스 유형	복제	설명
Fluent Bit	DaemonSet	6	로그 수집기, 집계기, 전달자

확장은 VMware 공용 레지스트리(https://projects.registry.vmware.com/)에서 컨테이너를 가져오도록 구성됩니다. 개인 레지스트리를 사용하는 경우 데이터 값 및 확장 구성 파일에서 끝점 URL이 일치하도록 변경합니다. Fluent Bit 확장 구성에서 필드 및 옵션에 대한 설명을 참조하십시오.

1 각 확장 사전 요구 사항을 완료했는지 확인합니다. 확장 사전 요구 사항의 내용을 참조하십시오.

2 디렉토리를 Fluent Bit 확장으로 변경합니다.

```
cd /tkg-extensions-v1.3.1+vmware.1/extensions/logging/fluent-bit
```

3 `tanzu-system-logging` 네임스페이스와 Fluent Bit 서비스 계정 및 역할 개체를 생성합니다.

```
kubectl apply -f namespace-role.yaml
```

4 Fluent Bit에 사용할 로그 대상을 결정합니다. 지원되는 출력에는 Elasticsearch, HTTP, Kafka, Splunk 및 Syslog가 포함됩니다. 자세한 내용은 https://docs.fluentbit.io/manual/pipeline/outputs의 내용을 참조하십시오.

5 `<LOG_BACKEND>/fluent-bit-data-values.example.yaml` 파일 중 하나를 복사하여 선택한 로그 대상에 대한 Fluent Bit 데이터 값 파일을 생성합니다.

지원되는 각 로그 대상에 대한 예제 데이터 값 파일이 있습니다. 이 예에는 로그 대상에 대한 최소 구성이 제공됩니다.

```
cp elasticsearch/fluent-bit-data-values.yaml.example elasticsearch/fluent-bit-data-values.yaml
```

```
cp http/fluent-bit-data-values.yaml.example http/fluent-bit-data-values.yaml
```

```
cp kafka/fluent-bit-data-values.yaml.example kafka/fluent-bit-data-values.yaml
```

```
cp splunk/fluent-bit-data-values.yaml.example splunk/fluent-bit-data-values.yaml
```

```
cp syslog/fluent-bit-data-values.yaml.example syslog/fluent-bit-data-values.yaml
```

6 <LOG_BACKEND>/fluent-bit-data-values.yaml을 채워서 Fluent Bit 확장을 구성합니다. Fluent Bit 확장 구성에서 필드 및 옵션에 대한 설명을 참조하십시오.

예를 들어 Fluent Bit syslog 구성에는 다음 값이 필요합니다.

```
logging:
  image:
    repository: projects.registry.vmware.com/tkg # Public registry
tkg:
  instance_name: "<TKG_INSTANCE_NAME>" #mandatory but arbitrary; appears in logs
  cluster_name: "<CLUSTER_NAME>" #name of the target tkgs cluster
fluent_bit:
  output_plugin: "syslog"
  syslog:
    host: "<SYSLOG_HOST>"
    port: "<SYSLOG_PORT>"
    mode: "<SYSLOG_MODE>"
    format: "<SYSLOG_FORMAT>"
```

Fluent Bit syslog에 대해 채워진 데이터 값 파일에는 다음과 같은 구성이 있을 수 있습니다.

```
logging:
  image:
    repository: projects.registry.vmware.com/tkg
tkg:
  instance_name: "tkgs-cluster-1"
  cluster_name: "tkgs-cluster-1"
fluent_bit:
  output_plugin: "syslog"
  syslog:
    host: "10.192.175.59"
    port: "514"
    mode: "tcp"
    format: "rfc5424"
```

7 로그 대상에 대한 데이터 값을 사용하여 Fluent Bit 암호를 생성합니다.

```
kubectl create secret generic fluent-bit-data-values --from-file=values.yaml=elasticsearch/
fluent-bit-data-values.yaml -n tanzu-system-logging
```

```
kubectl create secret generic fluent-bit-data-values --from-file=values.yaml=kafka/fluent-
bit-data-values.yaml -n tanzu-system-logging
```

```
kubectl create secret generic fluent-bit-data-values --from-file=values.yaml=splunk/fluent-
bit-data-values.yaml -n tanzu-system-logging
```

```
kubectl create secret generic fluent-bit-data-values --from-file=values.yaml=http/fluent-
bit-data-values.yaml -n tanzu-system-logging
```

```
kubectl create secret generic fluent-bit-data-values --from-file=values.yaml=syslog/fluent-
bit-data-values.yaml -n tanzu-system-logging
```

`tanzu-system-logging` 네임스페이스에 `secret/fluent-bit-data-values`가 생성됩니다. 다음 명령을 사용하여 확인합니다.

```
kubectl get secrets -n tanzu-system-logging
```

8 Fluent Bit 애플리케이션을 배포합니다.

```
kubectl apply -f fluent-bit-extension.yaml
```

성공하면 `app.kappctrl.k14s.io/fluent-bit created`가 표시됩니다.

9 Fluent Bit 애플리케이션의 상태를 확인합니다.

```
kubectl get app fluent-bit -n tanzu-system-logging
```

성공하면 상태가 `Reconciling`에서 `Reconcile succeeded`로 변경됩니다. 상태가 `Reconcile failed`인 경우 Fluent Bit 배포 문제 해결 항목을 참조하십시오.

10 애플리케이션에 대한 세부 상태를 봅니다.

```
kubectl get app fluent-bit -n tanzu-system-logging -o yaml
```

11 Fluent Bit DeamonSet을 확인합니다.

```
kubectl get daemonsets -n tanzu-system-logging
```

성공하면 다음이 표시됩니다.

```
NAME         DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
fluent-bit   6         6         6       6            6           <none>          105s
```

Fluent Bit 배포 문제 해결

배포 또는 조정이 실패하면 `kubectl get pods -A`를 실행하여 포드 상태를 확인합니다. `fluent-bit` 포드는 `Running`이어야 합니다. 포드 상태가 `ImagePullBackOff` 또는 `ImageCrashLoopBackOff`이면 컨테이너 이미지를 끌어올 수 없습니다. 데이터 값 및 확장 YAML 파일에서 레지스트리 URL을 확인하고 정확한지 확인합니다.

컨테이너 로그를 확인합니다.여기서 `name-XXXX`는 `kubectl get pods -A`를 실행할 때 볼 수 있는 고유한 포드 이름입니다.

```
kubectl logs pod/fluent-bit-XXXXX -c fluent-bit -n tanzu-system-logging
```

Fluent Bit 확장 업데이트

Tanzu Kubernetes 클러스터에 배포된 Fluent Bit 확장을 업데이트합니다.

1 암호에서 Fluent Bit 데이터 값을 얻습니다.

```
kubectl get secret fluent-bit-data-values -n tanzu-system-logging -o 'go-
template={{ index .data "values.yaml" }}' | base64 -d > fluent-bit-data-values.yaml
```

2 `fluent-bit-data-values.yaml`에서 Fluent Bit 데이터 값을 업데이트합니다. Fluent Bit 확장 구성의 내용을 참조하십시오.

3 Fluent Bit 데이터 값 암호를 업데이트합니다.

```
kubectl create secret generic fluent-bit-data-values --from-file=values.yaml=fluent-bit-
data-values.yaml -n tanzu-system-logging -o yaml --dry-run | kubectl replace -f-
```

Fluent Bit 확장이 위의 데이터 값으로 조정됩니다.

참고 기본적으로 kapp-controller는 5분마다 애플리케이션을 동기화합니다. 업데이트는 5분 이내에 적용됩니다. 업데이트를 즉시 적용하려면 `fluent-bit-extension.yaml`의 `syncPeriod`를 더 작은 값으로 변경하고 `kubectl apply -f fluent-bit-extension.yaml`을 사용하여 Fluent Bit 확장을 적용합니다.

4 확장의 상태를 확인합니다.

```
kubectl get app fluent-bit -n tanzu-system-logging
```

5 자세한 상태를 살펴보고 문제를 해결합니다.

```
kubectl get app fluent-bit -n tanzu-system-logging -o yaml
```

6 필요한 경우 문제를 해결합니다. Fluent Bit 배포 문제 해결의 내용을 참조하십시오.

Fluent Bit 확장 삭제

Tanzu Kubernetes 클러스터에서 Fluent Bit 확장을 삭제합니다.

참고 순서대로 단계를 완료하십시오. Fluent Bit 애플리케이션이 완전히 삭제되기 전에 네임스페이스, 서비스 계정 및 역할 개체를 삭제하지 마십시오. 그러면 시스템 오류가 발생할 수 있습니다.

1 디렉토리를 Fluent Bit 확장으로 변경합니다.

```
cd extensions/logging/fluent-bit/
```

2 Fluent Bit 애플리케이션을 삭제합니다.

```
kubectl delete app fluent-bit -n tanzu-system-logging
```

예상 결과: `app.kappctrl.k14s.io "fluent-bit" deleted`.

3 Fluent Bit 애플리케이션이 삭제되었는지 확인합니다.

```
kubectl get app fluent-bit -n tanzu-system-logging
```

예상 결과: `apps.kappctrl.k14s.io "fluent-bit" not found`.

4 `tanzu-system-logging` 네임스페이스와 Fluent Bit 확장 서비스 계정 및 역할 개체를 삭제합니다.

```
kubectl delete -f namespace-role.yaml
```

Fluent Bit 확장 업그레이드

기존 Fluent Bit 확장이 배포된 경우 최신 버전으로 업그레이드할 수 있습니다.

1 Fluent Bit configmap을 내보냅니다.

```
kubectl get configmap fluent-bit -n tanzu-system-logging -o 'go-template={{ index .data
"fluent-bit.yaml" }}' > fluent-bit-configmap.yaml
```

2 기존 Fluent Bit 배포를 삭제합니다. Fluent Bit 확장 삭제의 내용을 참조하십시오.

3 최신 Fluent Bit 확장을 배포합니다. Fluent Bit 확장 배포의 내용을 참조하십시오.

Fluent Bit 확장 구성

구성 값은 `extensions/logging/fluent-bit/<LOG_BACKEND>/fluent-bit-data-values.yaml`에 설정되어 있습니다.

표 14-3. Fluent Bit 확장 구성

매개 변수	설명	유형	기본값
logging.namespace	Fluent Bit가 배포될 네임스페이스	문자열	tanzu-system-logging
logging.service_account_name	Fluent Bit 서비스 계정의 이름	문자열	fluent-bit

표 14-3. Fluent Bit 확장 구성 (계속)

매개 변수	설명	유형	기본값
logging.cluster_role_name	Fluent Bit에 대한 get, watch 및 list 권한을 부여하는 클러스터 역할의 이름	문자열	fluent-bit-read
logging.image.name	Fluent Bit 이미지의 이름	문자열	fluent-bit
logging.image.tag	Fluent Bit 이미지 태그. 버전을 업그레이드하는 경우 이 값을 업데이트해야 할 수 있습니다.	문자열	v1.6.9_vmware.1
logging.image.repository	Fluent Bit 이미지가 있는 저장소의 위치입니다. 기본값은 공용 VMware 레지스트리입니다. 비공개 저장소(예: 에어 갭 환경)를 사용하는 경우 이 값을 변경합니다.	문자열	projects.registry.vmware.com/tkg
logging.image.pullPolicy	Fluent Bit 이미지 Pull 정책	문자열	IfNotPresent
logging.update_strategy	DaemonSet 업데이트 시 사용할 업데이트 전략	문자열	RollingUpdate
tkg.cluster_name	Tanzu Kubernetes 클러스터의 이름	문자열	Null(필수 매개 변수)
tkg.instance_name	하나의 배포에서 모든 Tanzu Kubernetes 클러스터 및 감독자 클러스터가 공유하는 TKG 인스턴스의 사용자 정의 이름. 설치와 관련된 모든 이름을 사용할 수 있습니다.	문자열	Null(필수 매개 변수) **참고** 이 필드는 필수이지만 임의의 필드입니다. 로그에 나타나는 이름입니다.
fluent_bit.log_level	Fluent Bit에 사용할 로그 수준	문자열	정보
fluent_bit.output_plugin	Fluent-Bit가 수집한 정보를 플러시해야 하는 백엔드 설정	문자열	Null(필수 매개 변수)
fluent_bit.elasticsearch.host	대상 Elasticsearch 인스턴스의 IP 주소 또는 호스트 이름	문자열	Null(output_plugin이 elastic search인 경우 매개 변수)
fluent_bit.elasticsearch.port	대상 Elasticsearch 인스턴스의 TCP 포트	정수	Null(output_plugin이 elastic search인 경우 매개 변수)
fluent_bit.elasticsearch.buffer_size	Elasticsearch 서비스의 응답을 읽는 데 사용되는 버퍼 크기를 지정. False인 경우 제한 없음으로 설정	문자열	False
fluent_bit.elasticsearch.tls	Elasticsearch에 대한 TLS의 기본 설정 지정	문자열	꺼짐

표 14-3. Fluent Bit 확장 구성 (계속)

매개 변수	설명	유형	기본값
fluent_bit.kafka.broker_service_name	Kafka 브로커의 여러 목록 중 하나(예: 192.168.1.3:9092)	문자열	Null(output_plugin이 kafka인 경우 필수 매개 변수)
fluent_bit.kafka.topic_name	Fluent Bit가 Kafka에 메시지를 보내는 데 사용할 단일 항목 또는 (,)로 구분된 항목의 목록	문자열	Null(output_plugin이 kafka인 경우 필수 매개 변수)
fluent_bit.splunk.host	대상 Splunk 서버의 IP 주소 또는 호스트 이름	문자열	Null(output_plugin이 splunk인 경우 필수 매개 변수)
fluent_bit.splunk.port	대상 Splunk 서버의 TCP 포트	정수	Null(output_plugin이 splunk인 경우 필수 매개 변수)
fluent_bit.splunk.token	HTTP 이벤트 수집기 인터페이스에 대한 인증 토큰 지정	문자열	Null(output_plugin이 splunk인 경우 필수 매개 변수)
fluent_bit.http.host	대상 HTTP 서버의 IP 주소 또는 호스트 이름	문자열	Null(output_plugin이 http인 경우 필수 매개 변수)
fluent_bit.http.port	대상 HTTP 서버의 TCP 포트	정수	Null(output_plugin이 http인 경우 필수 매개 변수)
fluent_bit.http.mode	대상 웹 서버에 대한 HTTP URI 지정	문자열	Null(output_plugin이 http인 경우 필수 매개 변수)
fluent_bit.http.header_key_value	HTTP 헤더 키/값 쌍. 여러 헤더를 설정할 수 있음	문자열	Null(output_plugin이 http인 경우 필수 매개 변수)
fluent_bit.http.format	HTTP 요청 본문에 사용할 데이터 형식 지정	문자열	Null(output_plugin이 http인 경우 필수 매개 변수)
fluent_bit.syslog.host	원격 Syslog 서버의 도메인 또는 IP 주소	문자열	Null(output_plugin이 syslog인 경우 필수 매개 변수)
fluent_bit.syslog.port	원격 Syslog 서버의 TCP 또는 UDP 포트	정수	Null(output_plugin이 syslog인 경우 필수 매개 변수)
fluent_bit.syslog.mode	TCP, UDP 및 TLS 중에 전송 유형 지정	문자열	Null(output_plugin이 syslog인 경우 필수 매개 변수)
fluent_bit.syslog.format	HTTP 요청 본문에 사용할 데이터 형식 지정	문자열	Null(output_plugin이 syslog인 경우 필수 매개 변수)
host_path.volume_1	볼륨 1에 대한, 호스트 노드의 파일 시스템에서 포드로의 디렉토리 경로	문자열	/var/log

표 14-3. Fluent Bit 확장 구성 (계속)

매개 변수	설명	유형	기본값
host_path.volume_2	볼륨 2에 대한, 호스트 노드의 파일 시스템에서 포드로의 디렉토리 경로	문자열	/var/lib/docker/containers
host_path.volume_3	볼륨 3에 대한, 호스트 노드의 파일 시스템에서 포드로의 디렉토리 경로	문자열	/run/log
systemd.path	Systemd 저널 디렉토리의 경로	문자열	/var/log/journal

Contour 수신용 TKG 확장 배포 및 관리

이 항목에서는 Contour 수신용 TKG 확장 v1.3.1을 배포하는 방법을 설명합니다. Contour는 엔보이 역방향 프록시를 사용하는 Kubernetes 수신 컨트롤러입니다. Contour 수신용 TKG 확장을 배포하여 Tanzu Kubernetes 클러스터에서 실행되는 서비스에 대한 수신 경로를 노출합니다.

확장 사전 요구 사항

Contour 수신용 TKG 확장 v1.3.1을 배포하기 전에 다음 요구 사항을 준수합니다.

- 클러스터를 프로비저닝합니다. TKGS v1alpha2 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하는 워크플로의 내용을 참조하십시오.
- 클러스터에 연결합니다. vCenter Single Sign-On 사용자로 Tanzu Kubernetes 클러스터에 연결의 내용을 참조하십시오.
- kubectl을 실행하는 클라이언트 호스트에 TKG 확장 v1.3.1 번들 다운로드합니다.
- 대상 클러스터에 TKG 확장 사전 요구 사항 설치합니다.

Contour 확장 배포

Contour 수신용 TKG 확장은 클러스터에 엔보이 및 Contour라는 두 가지 컨테이너를 설치합니다. 자세한 내용은 https://projectcontour.io/의 내용을 참조하십시오.

컨테이너	리소스 유형	복제	설명
엔보이	DaemonSet	3	고성능 역방향 프록시
Contour	배포	2	엔보이용 관리 및 구성 서버

확장은 VMware 공용 레지스트리(https://projects.registry.vmware.com/)에서 컨테이너를 가져오도록 구성됩니다. 개인 레지스트리를 사용하는 경우 데이터 값 및 확장 구성에서 끝점 URL이 일치하도록 변경합니다. Contour 확장 구성의 내용을 참조하십시오.

1 각각의 확장 사전 요구 사항을 완료했는지 확인합니다. 확장 사전 요구 사항의 내용을 참조하십시오.

2 디렉토리를 Contour 확장 파일을 다운로드한 위치로 변경합니다.

```
cd /tkg-extensions-v1.3.1+vmware.1/extensions/ingress/contour
```

3 다음 명령을 실행하여 tanzu-system-ingress 네임스페이스와 Contour 서비스 계정 및 역할 개체를 생성합니다.

```
kubectl apply -f namespace-role.yaml
```

4 vSphere에 대한 Contour 데이터 값 파일을 생성합니다.

```
cp vsphere/contour-data-values-lb.yaml.example vsphere/contour-data-values.yaml
```

5 vsphere/contour-data-values.yaml 파일을 업데이트하여 Contour를 구성합니다.

예제 데이터 값 파일은 필요한 최소 구성을 제공합니다. Contour 확장 구성에서 모든 구성 필드 및 옵션에 대한 설명을 참조하십시오.

예를 들어 다음 vSphere용 Contour 구성은 LoadBalancer 유형의 서비스를 사용합니다.

```
infrastructure_provider: "vsphere"
contour:
  image:
    repository: projects.registry.vmware.com/tkg
envoy:
  image:
    repository: projects.registry.vmware.com/tkg
    tag: v1.17.3_vmware.1
  service:
    type: "LoadBalancer"
```

참고 CVE가 있는 엔보이 이미지 버전 v1.16.2_vmware.1을 사용하지 않도록 엔보이 이미지 버전 v1.17.3_vmware.1을 지정하는 것이 좋습니다. 자세한 내용은 릴리스 정보를 참조하십시오.

6 데이터 값으로 암호를 생성합니다.

```
kubectl create secret generic contour-data-values --from-file=values.yaml=vsphere/contour-data-values.yaml -n tanzu-system-ingress
```

tanzu-system-ingress 네임스페이스에 secret/contour-data-values가 생성됩니다. 다음 명령을 사용하여 확인합니다.

```
kubectl get secrets -n tanzu-system-ingress
```

7 Contour 수신 컨트롤러 애플리케이션을 배포합니다.

```
kubectl apply -f contour-extension.yaml
```

성공하면 app.kappctrl.k14s.io/contour created가 표시됩니다.

8 Contour 수신 컨트롤러 애플리케이션의 상태를 확인합니다.

```
kubectl get app contour -n tanzu-system-ingress
```

성공하면 상태가 `Reconciling`에서 `Reconcile succeeded`로 변경됩니다. 상태가 `Reconcile failed`인 경우 Contour 수신 배포 문제 해결 항목을 참조하십시오.

9 Contour 수신 컨트롤러 애플리케이션에 대한 자세한 정보를 확인합니다.

```
kubectl get app contour -n tanzu-system-ingress -o yaml
```

10 LoadBalancer 유형의 엔보이 서비스를 봅니다.

```
kubectl get service envoy -n tanzu-system-ingress -o wide
```

성공하면 엔보이 LoadBalancer 세부 정보가 표시됩니다.

```
NAME    TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)                      AGE
SELECTOR
envoy   LoadBalancer   10.79.65.110   10.178.147.73   80:30437/TCP,443:30589/TCP   2m42s
app=envoy,kapp.k14s.io/app=1629916985840017976
```

11 엔보이 DaemonSet을 확인합니다.

```
kubectl get daemonsets -n tanzu-system-ingress
```

성공하면 3-포드 엔보이 DaemonSet이 표시됩니다.

```
NAME    DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
envoy   3         3         3       3            3           <none>          6m10s
```

12 Contour 배포를 확인합니다.

```
kubectl get deployments -n tanzu-system-ingress
```

성공하면 2-포드 Contour 배포가 표시됩니다.

```
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
contour   2/2     2            2           8m7s
```

13 Contour 수신 컨트롤러가 올바르게 설치되어 있고 사용할 준비가 되었는지 확인합니다.

```
kubectl get pod,svc -n tanzu-system-ingress
```

Contour 및 엔보이 포드의 상태는 `Running`이어야 하며 엔보이 서비스의 LoadBalancer에는 `EXTERNAL-IP`가 할당됩니다.

```
NAME                           READY   STATUS    RESTARTS   AGE
pod/contour-84bb5475cf-7h4cx   1/1     Running   0          9m52s
pod/contour-84bb5475cf-v8k9r   1/1     Running   0          9m52s
pod/envoy-4828j                2/2     Running   0          9m52s
pod/envoy-c54dw                2/2     Running   0          9m52s
```

```
pod/envoy-qpjqp                 2/2     Running   0          9m52s

NAME              TYPE           CLUSTER-IP     EXTERNAL-IP
PORT(S)                      AGE
service/contour   ClusterIP      10.105.6.207   <none>
8001/TCP                     9m52s
service/envoy     LoadBalancer   10.79.65.110   10.178.147.73   80:30437/
TCP,443:30589/TCP   9m52s
```

Contour 수신 배포 문제 해결

배포 또는 조정이 실패하면 `kubectl get pods -n tanzu-system-ingress`를 실행하여 포드 상태를 확인합니다. `contour` 및 `envoy` 포드는 `Running`이어야 합니다. 포드 상태가 `ImagePullBackOff` 또는 `ImageCrashLoopBackOff`이면 컨테이너 이미지를 끌어올 수 없습니다. 데이터 값 및 확장 YAML 파일에서 레지스트리 URL을 확인하고 정확한지 확인합니다.

컨테이너 로그를 확인합니다. 여기서 `name-XXXX`는 `kubectl get pods -A`를 실행할 때 고유한 포드 이름입니다.

```
kubectl logs pod/envoy-XXXXX -c envoy -n tanzu-system-ingress
```

```
kubectl logs pod/contour-XXXXX -c contour -n tanzu-system-ingress
```

Contour 포드가 위의 이미지 오류 중 하나가 표시되면서 실패하지 않고 진행되지 않으면서("contour-xxxxx 진행 중 시간이 초과됨") `ContainerCreating` 상태에서 멈춰 있으면 IP 주소 충돌이 있을 수 있습니다. 워크로드 네트워크를 구성할 때 지정한 노드 CIDR 범위가 클러스터 규격의 포드 CIDR 범위(기본적으로 `192.168.0.0/16`)와 충돌하지 않는지 확인합니다. 충돌이 있는 경우 클러스터를 다른 포드 서브넷으로 업데이트하거나 노드 네트워크를 변경합니다.

Contour 확장 업데이트

Tanzu Kubernetes 클러스터에 배포된 Contour 확장을 업데이트합니다.

1 암호에서 Contour 데이터 값을 얻습니다.

```
kubectl get secret contour-data-values -n tanzu-system-ingress -o 'go-
template={{ index .data "values.yaml" }}' | base64 -d > contour-data-values.yaml
```

2 `ingress/contour/values.yaml`에서 Contour 수신 데이터 값을 업데이트합니다. Contour 확장 구성의 내용을 참조하십시오.

예를 들어 다음 vSphere용 Contour 구성은 LoadBalancer 유형의 서비스를 사용합니다.

```
infrastructure_provider: "vsphere"
contour:
  image:
    repository: projects.registry.vmware.com/tkg
envoy:
  image:
```

```
    repository: projects.registry.vmware.com/tkg
    tag: v1.17.3_vmware.1
  service:
    type: "LoadBalancer"
```

참고 CVE가 있는 엔보이 이미지 버전 `v1.16.2_vmware.1`을 사용하지 않도록 엔보이 이미지 버전 `v1.17.3_vmware.1`을 지정하는 것이 좋습니다. 자세한 내용은 릴리스 정보를 참조하십시오.

3 Contour 데이터 값 암호를 업데이트합니다.

```
kubectl create secret generic contour-data-values --from-file=values.yaml=contour-data-values.yaml -n tanzu-system-ingress -o yaml --dry-run | kubectl replace -f-
```

Contour 확장이 새 데이터 값으로 조정됩니다.

참고 기본적으로 kapp-controller는 5분마다 애플리케이션을 동기화합니다. 업데이트는 5분 이내에 적용됩니다. 즉시 적용하려면 `contour-extension.yaml`의 `syncPeriod`를 더 작은 값으로 변경하고 `kubectl apply -f contour-extension.yaml`을 사용하여 확장을 다시 배포합니다.

4 애플리케이션의 상태를 확인합니다.

```
kubectl get app contour -n tanzu-system-ingress
```

Contour가 업데이트되면 상태가 `Reconcile Succeeded`로 변경됩니다.

5 자세한 상태를 봅니다.

```
kubectl get app contour -n tanzu-system-ingress -o yaml
```

6 필요한 경우 문제를 해결합니다. Contour 수신 배포 문제 해결의 내용을 참조하십시오.

Contour 확장 삭제

Tanzu Kubernetes 클러스터에서 Contour 확장을 삭제합니다.

참고 순서대로 단계를 완료하십시오. Contour 수신 컨트롤러 애플리케이션이 완전히 삭제되기 전에 네임스페이스, 서비스 계정 및 역할 개체를 삭제하지 마십시오. 그러면 시스템 오류가 발생할 수 있습니다.

1 디렉토리를 Contour 확장으로 변경합니다.

```
cd extensions/ingress/contour/
```

2 Contour 수신 컨트롤러 애플리케이션을 삭제합니다.

```
kubectl delete app contour -n tanzu-system-ingress
```

예상 결과: `app.kappctrl.k14s.io "contour" deleted`.

3 Contour 수신 컨트롤러 애플리케이션이 삭제되었는지 확인합니다.

```
kubectl get app contour -n tanzu-system-ingress
```

예상 결과: `apps.kappctrl.k14s.io "contour" not found.`

4 `tanzu-system-ingress` 네임스페이스와 Contour 확장 서비스 계정 및 역할 개체를 삭제합니다.

```
kubectl delete -f namespace-role.yaml
```

Contour 확장 업그레이드

기존 Contour 확장이 배포된 경우 최신 버전으로 업그레이드할 수 있습니다.

1 Contour configmap을 내보내서 백업으로 저장합니다.

```
kubectl get configmap contour -n tanzu-system-ingress -o 'go-template={{ index .data
"contour.yaml" }}' > contour-configmap.yaml
```

2 기존 Contour 배포를 삭제합니다. Contour 확장 삭제의 내용을 참조하십시오.

3 최신 Contour 확장을 배포합니다. Contour 확장 배포의 내용을 참조하십시오.

Contour 확장 구성

Contour 수신 컨트롤러 구성 값은 `/extensions/ingress/contour/vsphere/contour-data-values.yaml`에 설정됩니다.

표 14-4. Contour 수신 구성 매개 변수

매개 변수	설명	유형	기본값
infrastructure_provider	인프라 제공자. 지원되는 값: vsphere, aws, azure	문자열	필수 매개 변수
contour.namespace	Contour가 배포될 네임스페이스	문자열	tanzu-system-ingress
contour.config.requestTimeout	엔보이에 전달할 클라이언트 요청 시간 초과	time.Duration	0s 파일 다운로드에 대한 경로 시간 초과의 내용을 참조하십시오.
contour.config.server.xdsServerType	사용할 XDS 서버 유형: 지원되는 값: contour 또는 envoy	문자열	Null
contour.config.tls.minimumProtocolVersion	Contour가 협상할 최소 TLS 버전	문자열	1.1
contour.config.tls.fallbackCertificate.name	vhost에 대해 정의된 SNI와 일치하지 않는 요청에 대한 폴백 인증서가 포함된 암호의 이름	문자열	Null
contour.config.tls.fallbackCertificate.namespace	폴백 인증서가 포함된 암호의 네임스페이스	문자열	Null

표 14-4. Contour 수신 구성 매개 변수 (계속)

매개 변수	설명	유형	기본값
contour.config.tls.envoyClientCertificate.name	클라이언트 인증서로 사용할 암호의 이름, 백엔드 서비스에 대한 TLS 연결용 개인 키	문자열	Null
contour.config.tls.envoyClientCertificate.namespace	클라이언트 인증서로 사용할 암호의 네임스페이스, 백엔드 서비스에 대한 TLS 연결용 개인 키	문자열	Null
contour.config.leaderelection.configmapName	contour leaderelection에 사용할 configmap의 이름	문자열	leader-elect
contour.config.leaderelection.configmapNamespace	contour leaderelection configmap의 네임스페이스	문자열	tanzu-system-ingress
contour.config.disablePermitInsecure	ingressroute permitInsecure 필드를 사용하지 않도록 설정	부울	false
contour.config.accesslogFormat	액세스 로그 형식	문자열	envoy
contour.config.jsonFields	로깅될 필드	문자열 어레이	https://godoc.org/github.com/projectcontour/contour/internal/envoy#JSONFields
contour.config.useProxyProtocol	https://projectcontour.io/guides/proxy-proto/	부울	false
contour.config.defaultHTTPVersions	Contour가 제공할 엔보이를 프로그래밍해야 하는 HTTP 버전	문자열 어레이	"HTTP/1.1 HTTP2"
contour.config.timeouts.requestTimeout	전체 요청에 대한 시간 제한	time.Duration	Null(시간 제한을 사용하지 않도록 설정함)
contour.config.timeouts.connectionIdleTimeout	유휴 연결을 종료하기 전에 대기할 시간	time.Duration	60s
contour.config.timeouts.streamIdleTimeout	활동이 없는 요청 또는 스트림을 종료하기 전에 대기할 시간	time.Duration	5m
contour.config.timeouts.maxConnectionDuration	활동 여부에 관계없이 연결을 종료하기 전에 대기할 시간	time.Duration	Null(시간 제한을 사용하지 않도록 설정함)
contour.config.timeouts.ConnectionShutdownGracePeriod	초기 및 최종 GOAWAY 전송 사이의 대기 시간	time.Duration	5s
contour.config.cluster.dnsLookupFamily	HTTPProxy 경로에서 externalName 유형 서비스에 대한 업스트림 요청에 사용할 dns-lookup-family	문자열	Null(지원되는 값: auto, v4, v6)
contour.config.debug	Contour 디버깅 켜기	부울	false

표 14-4. Contour 수신 구성 매개 변수 (계속)

매개 변수	설명	유형	기본값
contour.config.ingressStatusAddress	모든 수신 리소스의 상태에 설정할 주소	문자열	Null
contour.certificate.duration	Contour 인증서 기간	time.Duration	8760h
contour.certificate.renewBefore	Contour 인증서를 갱신하기 전까지의 기간	time.Duration	360h
contour.deployment.replicas	Contour 복제본 수	정수	2
contour.image.repository	Contour 이미지가 있는 저장소의 위치입니다. 기본값은 공용 VMware 레지스트리입니다. 비공개 저장소(예: 에어갭 환경)를 사용하는 경우 이 값을 변경합니다.	문자열	projects.registry.vmware.com/tkg
contour.image.name	Contour 이미지의 이름	문자열	contour
contour.image.tag	Contour 이미지 태그. Contour 버전을 업그레이드하는 경우 이 값을 업데이트해야 할 수 있습니다.	문자열	v1.11.0_vmware.1
contour.image.pullPolicy	Contour 이미지 Pull 정책	문자열	IfNotPresent
envoy.image.repository	엔보이 이미지가 있는 저장소의 위치입니다. 기본값은 공용 VMware 레지스트리입니다. 비공개 저장소(예: 에어갭 환경)를 사용하는 경우 이 값을 변경합니다.	문자열	projects.registry.vmware.com/tkg
envoy.image.name	엔보이 이미지의 이름	문자열	envoy
envoy.image.tag	엔보이 이미지 태그. 엔보이 버전을 업그레이드하는 경우 이 값을 업데이트해야 할 수 있습니다.	문자열	`v1.17.3_vmware.1` **참고** CVE로 인해 엔보이 이미지 `v1.16.2_vmware.1`을 사용하지 마십시오. 자세한 내용은 릴리스 정보를 참조하십시오.
envoy.image.pullPolicy	엔보이 이미지 Pull 정책	문자열	IfNotPresent
envoy.hostPort.enable	호스트에서 엔보이 포트를 노출하는 플래그	부울	true
envoy.hostPort.http	엔보이 HTTP 호스트 포트	정수	80
envoy.hostPort.https	엔보이 HTTPS 호스트 포트	정수	443

표 14-4. Contour 수신 구성 매개 변수 (계속)

매개 변수	설명	유형	기본값
envoy.service.type	엔보이를 노출할 서비스 유형. 지원되는 값: ClusterIP, NodePort, LoadBalancer	문자열	vSphere에 대한 필수 매개 변수: NodePort 또는 LoadBalancer, AWS: LoadBalancer, Azure: LoadBalancer
envoy.service.annotations	엔보이 서비스 주석	맵(키-값)	빈 맵
envoy.service.externalTrafficPolicy	엔보이 서비스의 외부 트래픽 정책. 지원되는 값: Local, Cluster	문자열	클러스터
envoy.service.nodePort.http	http 요청에 사용되는 NodePort 유형의 서비스에 필요한 nodePort	정수	Null - Kubernetes가 동적 노드 포트를 할당함
envoy.service.nodePort.https	HTTPS 요청에 사용되는 NodePort 유형의 서비스에 필요한 nodePort	정수	Null - Kubernetes가 동적 노드 포트를 할당함
envoy.deployment.hostNetwork	hostNetwork에서 엔보이 실행	부울	false
envoy.service.aws.LBType	엔보이 서비스를 노출하는 데 사용할 AWS LB 유형. 지원되는 값: classic, nlb	문자열	classic
envoy.loglevel	엔보이에 사용할 로그 수준	문자열	정보

파일 다운로드에 대한 경로 시간 초과

`contour.config.requestTimeout` 매개 변수는Contour 경로 시간 초과 기간을 정의합니다. 기본값은 `0s`입니다. 파일 전송에 Contour를 사용하는 경우 이 값을 조정해야 할 수 있습니다.

Contour 설명서에 따르면 시간 초과 값이 `0s`이면 Contour가 엔보이 시간 초과를 사용하도록 지시합니다. 엔보이 설명서에 따르면 엔보이의 기본 시간 초과 값은 15초입니다. 또한 엔보이는 전체 요청-응답 작업이 시간 초과 간격 내에 완료될 것으로 예상합니다.

따라서 기본 Contour 시간 초과 설정이 `0s`인 경우 파일 전송이 15초 내에 완료되어야 합니다. 대용량 파일 전송에는 이 시간이 충분하지 않을 수 있습니다. 기본 엔보이 시간 초과를 사용하지 않도록 설정하려면 `contour.config.requestTimeout` 값을 `0`으로 설정합니다.

Prometheus 모니터링을 위한 TKG 확장 배포 및 관리

이 항목에서는 Prometheus용 TKG 확장 v1.3.1을 배포하는 방법을 설명합니다. Prometheus는 시스템 및 서비스 모니터링 시스템입니다. 구성된 대상에서 일정 간격으로 메트릭을 수집하고, 규칙 표현식을 평가하고, 결과를 표시하며, 일부 조건이 true로 확인되면 경고를 트리거할 수 있습니다. Alertmanager는 Prometheus에서 생성된 경고를 처리하고 수신 끝점으로 라우팅합니다. Prometheus용 TKG 확장을 배포하여 Tanzu Kubernetes 클러스터에 대한 메트릭을 수집하고 볼 수 있습니다.

확장 사전 요구 사항

클러스터 모니터링을 위해 Alertmanager와 함께 Prometheus용 TKG 확장 v1.3.1을 배포하기 전에 다음 요구 사항을 준수합니다.

- 클러스터를 프로비저닝합니다. TKGS v1alpha2 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하는 워크플로의 내용을 참조하십시오.

  참고 Prometheus 확장을 설치하려면 기본 serviceDomain(`cluster.local`)을 사용하는 클러스터를 배포해야 합니다.

- 클러스터에 연결합니다. vCenter Single Sign-On 사용자로 Tanzu Kubernetes 클러스터에 연결의 내용을 참조하십시오.
- kubectl을 실행하는 클라이언트 호스트에 TKG 확장 v1.3.1 번들 다운로드합니다.
- 대상 Tanzu Kubernetes 클러스터에 TKG 확장 사전 요구 사항 설치합니다.

일반 요구 사항 외에도 Prometheus 모니터링에는 기본 영구 스토리지 클래스가 필요합니다. 기본 영구 스토리지 클래스를 사용하여 클러스터를 생성하거나 확장을 배포할 때 Prometheus 구성 파일에서 클러스터를 지정할 수 있습니다. TKG 확장에 대한 영구 스토리지 요구 사항 검토의 내용을 참조하십시오.

Prometheus 확장 배포

Prometheus용 TKG 확장은 여러 컨테이너를 설치합니다. 자세한 내용은 https://prometheus.io/의 내용을 참조하십시오.

컨테이너	리소스 유형	복제	설명
`prometheus-alertmanager`	배포	1	Prometheus 서버와 같은 클라이언트 애플리케이션에서 보낸 경고를 처리합니다.
`prometheus-cadvisor`	DaemonSet	5	실행 중인 컨테이너의 리소스 사용량 및 성능 데이터를 분석하고 노출합니다.
`prometheus-kube-state-metrics`	배포	1	노드 상태 및 용량, 복제 세트 규정 준수, 포드, 작업 및 cronjob 상태, 리소스 요청 및 제한을 모니터링합니다.
`prometheus-node-exporter`	DaemonSet	5	커널에 의해 노출되는 하드웨어 및 OS 메트릭용 내보내기입니다.
`prometheus-pushgateway`	배포	1	스크레이핑할 수 없는 작업에서 메트릭을 푸시할 수 있는 서비스입니다.
`prometheus-server`	배포	1	스크레이핑, 규칙 처리 및 경고를 비롯한 핵심 기능을 제공합니다.

확장은 VMware 공용 레지스트리(https://projects.registry.vmware.com/)에서 컨테이너를 가져오도록 구성됩니다. 개인 레지스트리를 사용하는 경우 데이터 값 및 확장 구성에서 끝점 URL이 일치하도록 변경합니다. Prometheus 확장 구성의 내용을 참조하십시오.

1 각각의 확장 사전 요구 사항을 완료했는지 확인합니다. 확장 사전 요구 사항의 내용을 참조하십시오.

2 디렉토리를 Prometheus 확장으로 변경합니다.

```
cd /tkg-extensions-v1.3.1+vmware.1/extensions/monitoring/prometheus
```

3 tanzu-system-monitoring 네임스페이스와 Prometheus 서비스 계정 및 역할 개체를 생성합니다.

```
kubectl apply -f namespace-role.yaml
```

4 Prometheus 데이터 값 파일을 생성합니다.

예제 데이터 값 파일은 최소 구성을 제공합니다.

```
cp prometheus-data-values.yaml.example prometheus-data-values.yaml
```

5 prometheus-data-values.yaml을 업데이트하여 Prometheus 확장을 구성합니다. Prometheus 확장 구성에서 필드 및 옵션에 대한 설명을 참조하십시오.

클러스터가 기본 영구 스토리지 클래스로 프로비저닝되지 않은 경우 데이터 값 파일에서 지정할 수 있습니다. 네임스페이스에 영구 볼륨 할당을 위한 충분한 스토리지가 있는지도 확인합니다.

```
monitoring:
  prometheus_server:
    image:
      repository: projects.registry.vmware.com/tkg/prometheus
    pvc:
      storage_class: vwt-storage-policy
      storage: "8Gi"
  alertmanager:
    image:
      repository: projects.registry.vmware.com/tkg/prometheus
    pvc:
      storage_class: vwt-storage-policy
      storage: "8Gi"
...
```

6 prometheus-data-values 파일을 사용하여 Prometheus 암호를 생성합니다.

```
kubectl create secret generic prometheus-data-values --from-file=values.yaml=prometheus-
data-values.yaml -n tanzu-system-monitoring
```

tanzu-system-monitoring 네임스페이스에 prometheus-data-values 암호가 생성됩니다. kubectl get secrets -n tanzu-system-monitoring을 사용하여 확인합니다.

7 Prometheus 확장을 배포합니다.

```
kubectl apply -f prometheus-extension.yaml
```

성공하면 Prometheus 애플리케이션이 생성됩니다(app.kappctrl.k14s.io/prometheus created).

8 Prometheus 애플리케이션의 상태를 확인합니다.

```
kubectl get app prometheus -n tanzu-system-monitoring
```

상태가 Reconciling에서 Reconcile succeeded로 변경되어야 합니다. 상태가 Reconcile failed인 경우 문제 해결을 참조하십시오.

9 Prometheus 애플리케이션에 대한 자세한 정보를 봅니다.

```
kubectl get app prometheus -n tanzu-system-monitoring -o yaml
```

10 Prometheus DaemonSets를 확인합니다.

```
kubectl get daemonsets -n tanzu-system-monitoring
```

11 Prometheus 배포를 확인합니다.

```
kubectl get deployments -n tanzu-system-monitoring
```

Prometheus 배포 문제 해결

배포 또는 조정이 실패하면 kubectl get pods -A를 실행하여 포드의 상태를 봅니다. 정상적인 조건에서는 포드가 Running으로 표시되어야 합니다. 상태가 ImagePullBackOff 또는 ImageCrashLoopBackOff이면 저장소에서 컨테이너 이미지를 끌어올 수 없습니다. 데이터 값 및 확장 YAML 파일에서 URL을 확인하고 정확한지 확인합니다.

컨테이너 로그를 확인합니다. 여기서 name-XXXX는 kubectl get pods -A를 실행할 때 고유한 포드 이름입니다.

```
kubectl logs pod/prometheus-alertmanager-XXXXX -c prometheus-alertmanager -n tanzu-system-monitoring
```

```
kubectl logs pod/prometheus-server-XXXXX -c prometheus-server -n tanzu-system-monitoring
```

Prometheus 확장 업데이트

Tanzu Kubernetes 클러스터에 배포된 Prometheus 확장에 대한 구성을 업데이트합니다.

1 암호에서 Prometheus 데이터 값을 얻습니다.

```
kubectl get secret prometheus-data-values -n tanzu-system-monitoring -o 'go-template={{ index .data "values.yaml" }}' | base64 -d > prometheus-data-values.yaml
```

2 Prometheus 데이터 값 암호를 업데이트합니다.

```
kubectl create secret generic prometheus-data-values --from-file=values.yaml=prometheus-data-values.yaml -n tanzu-system-monitoring -o yaml --dry-run | kubectl replace -f-
```

Prometheus 확장이 업데이트된 데이터 값으로 조정됩니다.

참고 기본적으로 kapp-controller는 5분마다 애플리케이션을 동기화합니다. 업데이트는 5분 이내에 적용됩니다. 업데이트를 즉시 적용하려면 `prometheus-extension.yaml`의 `syncPeriod`를 더 작은 값으로 변경하고 `kubectl apply -f prometheus-extension.yaml`을 사용하여 Fluent Bit 확장을 적용합니다.

3 확장의 상태를 확인합니다.

```
kubectl get app prometheus -n tanzu-system-monitoring
```

Prometheus가 업데이트되면 상태가 `Reconcile Succeeded`로 변경됩니다.

4 자세한 상태를 살펴보고 문제를 해결합니다.

```
kubectl get app prometheus -n tanzu-system-monitoring -o yaml
```

Prometheus 확장 삭제

Tanzu Kubernetes 클러스터에서 Prometheus 확장을 삭제합니다.

참고 순서대로 단계를 완료하십시오. Prometheus 애플리케이션이 완전히 삭제되기 전에 네임스페이스, 서비스 계정 및 역할 개체를 삭제하지 마십시오. 그러면 시스템 오류가 발생할 수 있습니다.

경고 Prometheus와 Grafana는 모두 동일한 네임스페이스를 사용합니다. 네임스페이스를 삭제하면 거기에 배포된 확장에 방해가 됩니다. Grafana가 배포된 경우 Grafana를 삭제하기 전에 네임스페이스를 삭제하지 마십시오.

1 디렉토리를 Prometheus 확장으로 변경합니다.

```
cd /extensions/monitoring/prometheus/
```

2 Prometheus 애플리케이션을 삭제합니다.

```
kubectl delete app prometheus -n tanzu-system-monitoring
```

예상 결과: `app.kappctrl.k14s.io "prometheus" deleted`.

3 Prometheus 애플리케이션이 삭제되었는지 확인합니다.

```
kubectl get app prometheus -n tanzu-system-monitoring
```

예상 결과: `apps.kappctrl.k14s.io "prometheus" not found`.

4 `tanzu-system-monitoring` 네임스페이스와 Prometheus 서비스 계정 및 역할 개체를 삭제합니다.

경고 Grafana가 배포된 경우에는 이 단계를 수행하지 마십시오.

```
kubectl delete -f namespace-role.yaml
```

5 Prometheus를 다시 배포하려면 암호 `prometheus-data-values`를 제거합니다.

```
kubectl delete secret prometheus-data-values -n tanzu-system-monitoring
```

예상 결과: `secret "prometheus-data-values" deleted`.

Prometheus 확장 업그레이드

기존 Prometheus 확장이 배포된 경우 최신 버전으로 업그레이드할 수 있습니다.

1 Prometheus configmap을 내보내서 백업으로 저장합니다.

```
kubectl get configmap prometheus -n tanzu-system-monitoring -o 'go-template={{ index .data
"prometheus.yaml" }}' > prometheus-configmap.yaml
```

2 기존 Prometheus 배포를 삭제합니다. Prometheus 확장 삭제의 내용을 참조하십시오.

3 Prometheus 확장을 배포합니다. Prometheus 확장 배포의 내용을 참조하십시오.

Prometheus 확장 구성

Prometheus 구성은 `/extensions/monitoring/prometheus/prometheus-data-values.yaml`에서 설정됩니다.

표 14-5. Prometheus 구성 매개 변수

매개 변수	설명	유형	기본값
monitoring.namespace	Prometheus가 배포될 네임스페이스	문자열	tanzu-system-monitoring
monitoring.create_namespace	이 플래그는 monitoring.namespace를 통해 지정된 네임스페이스를 생성할지 여부를 나타냅니다.	부울	false
monitoring.prometheus_server.config.prometheus_yaml	Prometheus에 전달할 Kubernetes 클러스터 모니터링 구성 세부 정보	yaml 파일	prometheus.yaml
monitoring.prometheus_server.config.alerting_rules_yaml	Prometheus에 정의된 자세한 경고 규칙	yaml 파일	alerting_rules.yaml
monitoring.prometheus_server.config.recording_rules_yaml	Prometheus에 정의된 자세한 기록 규칙	yaml 파일	recording_rules.yaml
monitoring.prometheus_server.service.type	Prometheus를 노출할 서비스 유형. 지원되는 값: ClusterIP	문자열	ClusterIP
monitoring.prometheus_server.enable_alerts.kubernetes_api	Prometheus의 Kubernetes API에 대해 SLO 경고 사용	부울	true
monitoring.prometheus_server.sc.aws_type	AWS의 storageclass에 대해 정의된 AWS 유형	문자열	gp2

표 14-5. Prometheus 구성 매개 변수 (계속)

매개 변수	설명	유형	기본값
monitoring.prometheus_server.sc.aws_fsType	AWS의 storageclass에 대해 정의된 AWS 파일 시스템 유형	문자열	ext4
monitoring.prometheus_server.sc.allowVolumeExpansion	AWS의 storageclass에 대해 볼륨 확장이 허용되는지 정의	부울	true
monitoring.prometheus_server.pvc.annotations	스토리지 클래스 주석	맵	{}
monitoring.prometheus_server.pvc.storage_class	영구 볼륨 할당에 사용할 스토리지 클래스. 기본값은 null이고 기본 프로비저너가 사용됩니다.	문자열	null
monitoring.prometheus_server.pvc.accessMode	영구 볼륨 할당에 대한 액세스 모드 정의. 지원되는 값: ReadWriteOnce, ReadOnlyMany, ReadWriteMany	문자열	ReadWriteOnce
monitoring.prometheus_server.pvc.storage	영구 볼륨 할당에 대한 스토리지 크기 정의	문자열	8Gi
monitoring.prometheus_server.deployment.replicas	Prometheus 복제본 수	정수	1
monitoring.prometheus_server.image.repository	Prometheus 이미지가 있는 저장소의 위치입니다. 기본값은 공용 VMware 레지스트리입니다. 비공개 저장소(예: 에어갭 환경)를 사용하는 경우 이 값을 변경합니다.	문자열	projects.registry.vmware.com/tkg/prometheus
monitoring.prometheus_server.image.name	Prometheus 이미지의 이름	문자열	prometheus
monitoring.prometheus_server.image.tag	Prometheus 이미지 태그. 버전을 업그레이드하는 경우 이 값을 업데이트해야 할 수 있습니다.	문자열	v2.17.1_vmware.1
monitoring.prometheus_server.image.pullPolicy	Prometheus 이미지 Pull 정책	문자열	IfNotPresent

monitoring.alertmanager.config.slack_demo	Alertmanager에 대한 Slack 알림 구성	문자열	`slack_demo:`
			`  name: slack_demo`
			`  slack_configs:`
			`  - api_url:`
			`https://`
			`hooks.slack.com`
			`  channel:`
			`'#alertmanager-`
			`test'`

표 14-5. Prometheus 구성 매개 변수 (계속)

매개 변수	설명	유형	기본값

monitoring.alertmanager.config.email_receiver	Alertmanager에 대한 이메일 알림 구성	문자열	`email_receiver:`
			`  name: email-receiver`
			`  email_configs:`
			`  - to: demo@tanzu.com`
			`  send_resolved: false`
			`  from: from-email@tanzu.com`
			`  smarthost: smtp.eample.com:25`
			`  require_tls: false`

monitoring.alertmanager.service.type	Alertmanager를 노출할 서비스 유형. 지원되는 값: ClusterIP	문자열	ClusterIP
monitoring.alertmanager.image.repository	Alertmanager 이미지가 있는 저장소의 위치입니다. 기본값은 공용 VMware 레지스트리입니다. 비공개 저장소(예: 에어갭 환경)를 사용하는 경우 이 값을 변경합니다.	문자열	projects.registry.vmware.com/tkg/prometheus
monitoring.alertmanager.image.name	Alertmanager 이미지의 이름	문자열	alertmanager
monitoring.alertmanager.image.tag	Alertmanager 이미지 태그. 버전을 업그레이드하는 경우 이 값을 업데이트해야 할 수 있습니다.	문자열	v0.20.0_vmware.1
monitoring.alertmanager.image.pullPolicy	Alertmanager 이미지 Pull 정책	문자열	IfNotPresent
monitoring.alertmanager.pvc.annotations	스토리지 클래스 주석	맵	{}
monitoring.alertmanager.pvc.storage_class	영구 볼륨 할당에 사용할 스토리지 클래스. 기본값은 null이고 기본 프로비저너가 사용됩니다.	문자열	null
monitoring.alertmanager.pvc.accessMode	영구 볼륨 할당에 대한 액세스 모드 정의. 지원되는 값: ReadWriteOnce, ReadOnlyMany, ReadWriteMany	문자열	ReadWriteOnce
monitoring.alertmanager.pvc.storage	영구 볼륨 할당에 대한 스토리지 크기 정의	문자열	2Gi

표 14-5. Prometheus 구성 매개 변수 (계속)

매개 변수	설명	유형	기본값
monitoring.alertmanager.deployment.replicas	Alertmanager 복제본 수	정수	1
monitoring.kube_state_metrics.image.repository	kube-state-metircs 이미지가 포함된 저장소. 기본값은 공용 VMware 레지스트리입니다. 비공개 저장소(예: 에어갭 환경)를 사용하는 경우 이 값을 변경합니다.	문자열	projects.registry.vmware.com/tkg/prometheus
monitoring.kube_state_metrics.image.name	kube-state-metircs 이미지의 이름	문자열	kube-state-metrics
monitoring.kube_state_metrics.image.tag	kube-state-metircs 이미지 태그. 버전을 업그레이드하는 경우 이 값을 업데이트해야 할 수 있습니다.	문자열	v1.9.5_vmware.1
monitoring.kube_state_metrics.image.pullPolicy	kube-state-metircs 이미지 Pull 정책	문자열	IfNotPresent
monitoring.kube_state_metrics.deployment.replicas	kube-state-metrics 복제본 수	정수	1
monitoring.node_exporter.image.repository	node-exporter 이미지가 포함된 저장소. 기본값은 공용 VMware 레지스트리입니다. 비공개 저장소(예: 에어갭 환경)를 사용하는 경우 이 값을 변경합니다.	문자열	projects.registry.vmware.com/tkg/prometheus
monitoring.node_exporter.image.name	node-exporter 이미지의 이름	문자열	node-exporter
monitoring.node_exporter.image.tag	node-exporter 이미지 태그. 버전을 업그레이드하는 경우 이 값을 업데이트해야 할 수 있습니다.	문자열	v0.18.1_vmware.1
monitoring.node_exporter.image.pullPolicy	node-exporter 이미지 Pull 정책	문자열	IfNotPresent
monitoring.node_exporter.hostNetwork	`hostNetwork: true`로 설정된 경우 포드는 노드의 네트워크 네임스페이스 및 네트워크 리소스를 사용할 수 있습니다.	부울	false
monitoring.node_exporter.deployment.replicas	node-exporter 복제본 수	정수	1

표 14-5. Prometheus 구성 매개 변수 (계속)

매개 변수	설명	유형	기본값
monitoring.pushgateway.image.repository	pushgateway 이미지가 포함된 저장소. 기본값은 공용 VMware 레지스트리입니다. 비공개 저장소(예: 에어갭 환경)를 사용하는 경우 이 값을 변경합니다.	문자열	projects.registry.vmware.com/tkg/prometheus
monitoring.pushgateway.image.name	pushgateway 이미지의 이름	문자열	pushgateway
monitoring.pushgateway.image.tag	pushgateway 이미지 태그. 버전을 업그레이드하는 경우 이 값을 업데이트해야 할 수 있습니다.	문자열	v1.2.0_vmware.1
monitoring.pushgateway.image.pullPolicy	pushgateway 이미지 Pull 정책	문자열	IfNotPresent
monitoring.pushgateway.deployment.replicas	pushgateway 복제본 수	정수	1
monitoring.cadvisor.image.repository	cadvisor 이미지가 포함된 저장소. 기본값은 공용 VMware 레지스트리입니다. 비공개 저장소(예: 에어갭 환경)를 사용하는 경우 이 값을 변경합니다.	문자열	projects.registry.vmware.com/tkg/prometheus
monitoring.cadvisor.image.name	cadvisor 이미지의 이름	문자열	cadvisor
monitoring.cadvisor.image.tag	cadvisor 이미지 태그. 버전을 업그레이드하는 경우 이 값을 업데이트해야 할 수 있습니다.	문자열	v0.36.0_vmware.1
monitoring.cadvisor.image.pullPolicy	cadvisor 이미지 풀 정책	문자열	IfNotPresent
monitoring.cadvisor.deployment.replicas	cadvisor 복제본 수	정수	1
monitoring.ingress.enabled	Prometheus 및 Alertmanager에 수신 사용/사용 안 함	부울	false 수신을 사용하려면 이 필드를 `true`로 설정하고 Contour 수신용 TKG 확장 배포 및 관리를 배포합니다. Prometheus에 액세스하려면 `prometheus.system.tanzu`를 작업자 노드 IP 주소에 매핑하는 항목으로 로컬 `/etc/hosts`를 업데이트합니다.
monitoring.ingress.virtual_host_fqdn	Prometheus 및 Alertmanager에 액세스하기 위한 호스트 이름	문자열	prometheus.system.tanzu

표 14-5. Prometheus 구성 매개 변수 (계속)

매개 변수	설명	유형	기본값
monitoring.ingress.prometheus_prefix	Prometheus의 경로 접두사	문자열	/
monitoring.ingress.alertmanager_prefix	Alertmanager의 경로 접두사	문자열	/alertmanager/
monitoring.ingress.tlsCertificate.tls.crt	자체 TLS 인증서를 사용하려는 경우 수신을 위한 선택적 인증서. 자체 서명된 인증서가 기본적으로 생성됨	문자열	생성된 인증서
monitoring.ingress.tlsCertificate.tls.key	자체 TLS 인증서를 사용하려는 경우, 수신을 위한 선택적 인증서 개인 키.	문자열	생성된 인증서 키

표 14-6. Prometheus_Server Configmap에 대한 구성 가능 필드

매개 변수	설명	유형	기본값
evaluation_interval	규칙을 평가하는 빈도	지속 시간	1m
scrape_interval	대상을 스크레이핑하는 빈도	지속 시간	1m
scrape_timeout	스크레이핑 요청 시간이 초과될 때까지 걸리는 시간	지속 시간	10s
rule_files	규칙 파일은 glob 목록을 지정함. 일치하는 모든 파일에서 규칙 및 경고를 읽음	yaml 파일	
scrape_configs	스크레이핑 구성 목록.	목록	
job_name	스크레이핑된 메트릭에 기본적으로 할당되는 작업 이름	문자열	
kubernetes_sd_configs	Kubernetes 서비스 검색 구성 목록.	목록	
relabel_configs	대상 레이블 재지정 구성 목록.	목록	
작업	regex 매칭을 기반으로 수행할 작업.	문자열	
regex	추출된 값을 매칭할 정규식.	문자열	
source_labels	소스 레이블은 기존 레이블에서 값을 선택합니다.	문자열	
scheme	요청에 사용되는 프로토콜 체계를 구성합니다.	문자열	
tls_config	스크레이핑 요청의 TLS 설정을 구성합니다.	문자열	

표 14-6. Prometheus_Server Configmap에 대한 구성 가능 필드 (계속)

매개 변수	설명	유형	기본값
ca_file	API 서버 인증서의 유효성을 검사하는 데 사용할 CA 인증서.	filename	
insecure_skip_verify	서버 인증서의 유효성 검사를 사용하지 않도록 설정.	부울	
bearer_token_file	보유자 토큰 파일 인증 정보 (선택 사항).	filename	
replacement	정규식이 일치하는 경우 정규식 대체를 수행할 대체 값.	문자열	
target_label	대체 작업에서 결과 값이 써지는 레이블.	문자열	

표 14-7. Alertmanager Configmap에 대한 구성 가능 필드

매개 변수	설명	유형	기본값
resolve_timeout	ResolveTimeout은 경고에 EndsAt가 포함되지 않은 경우 Alertmanager에서 사용하는 기본값임	지속 시간	5m
smtp_smarthost	이메일 전송에 사용되는 SMTP 호스트.	지속 시간	1m
slack_api_url	Slack webhook URL.	문자열	global.slack_api_url
pagerduty_url	API 요청을 보낼 pagerduty URL.	문자열	global.pagerduty_url
템플릿	사용자 지정 알림 템플릿 정의를 읽을 파일	파일 경로	
group_by	경고를 레이블로 그룹화	문자열	
group_interval	그룹에 추가된 새 경고에 대한 알림을 보내기 전에 대기할 시간 설정	지속 시간	5m
group_wait	경고 그룹에 대한 알림을 보내기 위해 초기에 대기하는 시간	지속 시간	30s
repeat_interval	경고에 대한 알림이 이미 전송된 경우 알림을 다시 보내기 전에 대기할 시간	지속 시간	4h
receivers	알림 수신기 목록.	목록	
severity	인시던트의 심각도.	문자열	
channel	알림을 보낼 대상 채널 또는 사용자.	문자열	

표 14-7. Alertmanager Configmap에 대한 구성 가능 필드 (계속)

매개 변수	설명	유형	기본값
html	이메일 알림의 HTML 본문.	문자열	
text	이메일 알림의 텍스트 본문.	문자열	
send_resolved	해결된 경고에 대해 알릴지 여부.	filename	
email_configs	이메일 통합 구성	부울	

포드에 대한 주석을 사용하면 스크레이핑 프로세스를 세밀하게 제어할 수 있습니다. 이러한 주석은 포드 메타데이터의 일부여야 합니다. Services 또는 DaemonSets와 같은 다른 개체에 설정하면 효과가 없습니다.

표 14-8. Prometheus 포드 주석

포드 주석	설명
`prometheus.io/scrape`	기본 구성은 모든 포드를 스크레이핑하며 false로 설정하면 이 주석이 스크레이핑 프로세스에서 포드를 제외합니다.
`prometheus.io/path`	메트릭 경로가 /metrics가 아니면 이 주석을 사용하여 정의합니다.
`prometheus.io/port`	포드의 선언된 포트 대신 지정한 포트에서 포드를 스크레이핑합니다(아무것도 선언되지 않은 경우 기본값은 포트 없는 대상임).

아래의 DaemonSet 매니페스트는 Prometheus가 포트 9102에서 모든 포드를 스크레이핑하도록 지시합니다.

```
apiVersion: apps/v1beta2 # for versions before 1.8.0 use extensions/v1beta1
kind: DaemonSet
metadata:
  name: fluentd-elasticsearch
  namespace: weave
  labels:
    app: fluentd-logging
spec:
  selector:
    matchLabels:
      name: fluentd-elasticsearch
  template:
    metadata:
      labels:
        name: fluentd-elasticsearch
      annotations:
        prometheus.io/scrape: 'true'
        prometheus.io/port: '9102'
```

```
    spec:
      containers:
      - name: fluentd-elasticsearch
        image: gcr.io/google-containers/fluentd-elasticsearch:1.20
```

Grafana 모니터링을 위한 TKG 확장 배포 및 관리

이 항목에서는 Grafana용 TKG 확장 v1.3.1을 배포하고 관리하는 방법을 설명합니다. Grafana를 사용하면 저장된 위치에 상관없이 메트릭을 쿼리, 시각화, 경고 및 탐색할 수 있습니다. Grafana는 애플리케이션 데이터에서 그래프 및 시각화를 형성하는 도구를 제공합니다. Grafana용 TKG 확장을 배포하여 Tanzu Kubernetes 클러스터에 대한 메트릭을 생성하고 확인합니다.

Grafana 확장 사전 요구 사항

확장을 배포하려면 다음 사전 요구 사항을 준수합니다.

- 클러스터를 프로비저닝합니다. TKGS v1alpha2 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하는 워크플로의 내용을 참조하십시오.

  참고 기본 serviceDomain(`cluster.local`)을 사용하는 클러스터를 배포해야 합니다.

- 클러스터에 연결합니다. vCenter Single Sign-On 사용자로 Tanzu Kubernetes 클러스터에 연결의 내용을 참조하십시오.
- kubectl 명령을 실행하는 클라이언트 호스트에 TKG 확장 v1.3.1 번들 다운로드합니다.
- 대상 클러스터에 TKG 확장 사전 요구 사항 설치합니다.

Grafana 확장 추가 요구 사항

Grafana 모니터링용 TKG 확장 v1.3.1에는 설치 전후에 추가적인 요구 사항이 있습니다.

- Grafana 모니터링 확장에는 기본 영구 스토리지 클래스가 필요합니다. 기본 영구 스토리지 클래스를 사용하여 클러스터를 생성하거나 확장을 배포할 때 Grafana 구성 파일에서 클러스터를 지정할 수 있습니다. TKG 확장에 대한 영구 스토리지 요구 사항 검토의 내용을 참조하십시오.
- Grafana 확장이 배포되면 Kuberetes 서비스 유형인 ClusterIP(기본값), NodePort 또는 LoadBalancer 중 하나에서 노출되는 IP 주소를 사용하여 HTTP/S를 통해 Grafana 대시보드에 액세스합니다. 클러스터 외부에서 Grafana 대시보드에 액세스하려면 Grafana를 배포하기 전에 Contour 확장을 배포합니다. Contour 확장을 배포하려면 Grafana 모니터링을 위한 TKG 확장 배포 및 관리 항목을 참조하십시오.

Grafana는 다음과 같은 Kuberentes 서비스 유형을 지원합니다.

서비스 유형	설명	액세스 지원
ClusterIP	클러스터 내부 IP에서 서비스를 노출합니다.	서비스는 클러스터 내에서만 액세스할 수 있습니다.
NodePort	정적 포트에서 각 노드의 IP에 서비스를 노출합니다.	클러스터 외부에서 서비스에 액세스할 수 있습니다.
LoadBalancer	로드 밸런서를 사용하여 서비스를 외부에 노출합니다.	클러스터 외부에서 서비스에 액세스할 수 있습니다.

ClusterIP가 기본값이지만 클러스터 내에서만 액세스할 수 있습니다. 감독자 클러스터에 NSX-T 네트워킹을 사용하는 경우 LoadBalancer 유형의 Contour 엔보이 서비스를 생성합니다. 감독자 클러스터에 vSphere vDS 네트워킹을 사용하는 경우에는 요구 사항에 따라 LoadBalancer 또는 NodePort 유형의 Contour 엔보이 서비스를 생성합니다.

시각화 및 분석을 위해 Grafana 확장 배포

Grafana용 TKG 확장은 단일 컨테이너를 배포합니다. 자세한 내용은 https://grafana.com/의 내용을 참조하십시오.

컨테이너	리소스 유형	복제	설명
Grafana	배포	2	데이터 시각화

확장은 VMware 공용 레지스트리(https://projects.registry.vmware.com/)에서 컨테이너를 가져오도록 구성됩니다. 개인 레지스트리를 사용하는 경우 데이터 값 및 확장 구성에서 끝점 URL이 일치하도록 변경합니다. Grafana 확장 구성의 내용을 참조하십시오.

1 각각의 Grafana 확장 사전 요구 사항을 완료했는지 확인합니다. Grafana 확장 사전 요구 사항의 내용을 참조하십시오.

2 디렉토리를 Grafana 확장으로 변경합니다.

```
cd /tkg-extensions-v1.3.1+vmware.1/extensions/monitoring/grafana
```

3 tanzu-system-monitoring 네임스페이스와 Grafana 서비스 계정 및 역할 개체를 생성합니다.

```
kubectl apply -f namespace-role.yaml
```

4 Grafana 데이터 값 파일을 생성합니다.

예제 데이터 값 파일은 필요한 최소 구성을 제공합니다.

```
cp grafana-data-values.yaml.example grafana-data-values.yaml
```

5 grafana-data-values.yaml을 업데이트하여 Grafana 확장을 구성합니다.

필요에 따라 구성을 사용자 지정합니다. Grafana 확장 구성의 내용을 참조하십시오.

admin_password를 base64로 인코딩해야 하지만 그렇지 않더라도 확장 배포가 차단되지 않습니다. 아래 예에서 암호 "admin"은 base64로 인코딩됩니다. https://www.base64encode.org/에서 자신의 Grafana 암호를 인코딩하십시오.

클러스터가 기본 스토리지 클래스로 프로비저닝되지 않은 경우 데이터 값 파일에서 지정할 수 있습니다. 네임스페이스에 영구 볼륨 할당을 위한 충분한 스토리지가 있는지도 확인합니다.

```
monitoring:
  grafana:
    image:
      repository: "projects.registry.vmware.com/tkg/grafana"
    pvc:
      storage_class: vwt-storage-policy
      storage: "8Gi"
    secret:
      admin_password: "YWRtaW4="
  grafana_init_container:
    image:
      repository: "projects.registry.vmware.com/tkg/grafana"
  grafana_sc_dashboard:
    image:
      repository: "projects.registry.vmware.com/tkg/grafana"
```

LoadBalancer 또는 NodePort 유형의 엔보이 서비스를 사용하여 Contour를 배포한 경우 다음과 같이 구성 파일에 지정합니다. 자세한 내용은 Grafana 확장 구성의 내용을 참조하십시오.

```
monitoring:
  grafana:
    service:
      type: LoadBalancer OR NodePort
```

기본적으로 Grafana 확장은 Grafana 대시보드에 액세스하기 위한 FQDN(정규화된 도메인 이름) grafana.system.tanzu를 생성합니다. 구성 파일 (monitoring.grafana.ingress.virtual_host_fqdn)에 원하는 호스트 이름을 지정하여 이 FQDN을 사용자 지정할 수 있습니다. 자세한 내용은 Grafana 확장 구성의 내용을 참조하십시오.

6 grafana-data-values 파일을 사용하여 Grafana 암호를 생성합니다.

```
kubectl create secret generic grafana-data-values --from-file=values.yaml=grafana-data-values.yaml -n tanzu-system-monitoring
```

tanzu-system-monitoring 네임스페이스에 grafana-data-values 암호가 생성됩니다. kubectl get secrets -n tanzu-system-monitoring을 사용하여 확인합니다.

7 Grafana 확장을 배포합니다.

```
kubectl apply -f grafana-extension.yaml
```

성공하면 Grafana 애플리케이션이 생성됩니다(app.kappctrl.k14s.io/grafana created).

8 Grafana 애플리케이션의 상태를 확인합니다.

```
kubectl get app grafana -n tanzu-system-monitoring
```

상태가 `Reconciling`에서 `Reconcile succeeded`로 변경되어야 합니다. 상태가 `Reconcile failed`인 경우 문제 해결을 참조하십시오.

9 Grafana 애플리케이션에 대한 세부 상태를 표시합니다.

```
kubectl get app grafana -n tanzu-system-monitoring -o yaml
```

10 Grafana 배포를 확인합니다.

```
kubectl get deployments -n tanzu-system-monitoring
```

LoadBalancer 유형의 Contour 엔보이 서비스를 사용하여 Grafana 대시보드에 액세스

LoadBalancer 유형의 필수 Contour 엔보이 서비스가 배포되고 Grafana 구성 파일에서 이를 지정한 경우 로드 밸런서의 외부 IP 주소를 가져오고 Grafana FQDN에 대한 DNS 레코드를 생성합니다.

1 LoadBalancer 유형의 엔보이 서비스에 대한 `External-IP` 주소를 가져옵니다.

```
kubectl get service envoy -n tanzu-system-ingress
```

반환된 `External-IP` 주소가 다음과 같이 표시됩니다.

```
NAME    TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)                      AGE
envoy   LoadBalancer   10.99.25.220   10.195.141.17   80:30437/TCP,443:30589/TCP   3h27m
```

또는 다음 명령을 사용하여 `External-IP` 주소를 가져올 수 있습니다.

```
kubectl get svc envoy -n tanzu-system-ingress -o
jsonpath='{.status.loadBalancer.ingress[0]}'
```

2 Grafana 확장 설치를 확인하려면 로드 밸런서의 `External-IP` 주소에 매핑된 Grafana FQDN으로 로컬 `/etc/hosts` 파일을 업데이트합니다. 예를 들면 다음과 같습니다.

```
127.0.0.1 localhost
127.0.1.1 ubuntu
# TKG Grafana Extension with Envoy Load Balancer
10.195.141.17 grafana.system.tanzu
```

3 https://grafana.system.tanzu로 이동하여 Grafana 대시보드에 액세스합니다.

사이트는 자체 서명된 인증서를 사용하기 때문에 대시보드에 액세스하려면 먼저 브라우저별 보안 주의를 거쳐 이동해야 할 수 있습니다.

4 운영 액세스를 위해 엔보이 서비스 로드 밸런서 `External-IP` 주소를 Grafana 대시보드에 매핑하는 DNS 서버에 두 개의 CNAME 레코드를 생성합니다.

NodePort 유형의 Contour 엔보이 서비스를 사용하여 Grafana 대시보드에 액세스

NodePort 유형의 필수 Contour 엔보이 서비스가 배포되고 Grafana 구성 파일에서 이를 지정한 경우 작업자 노드의 가상 시스템 IP 주소를 가져오고 Grafana FQDN에 대한 DNS 레코드를 생성합니다.

1 클러스터가 프로비저닝된 vSphere 네임스페이스로 컨텍스트를 전환합니다.

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

2 클러스터의 노드를 나열합니다.

```
kubectl get virtualmachines
```

클러스터 노드가 표시됩니다. 예를 들면 다음과 같습니다.

```
NAME                                                POWERSTATE   AGE
tkgs-cluster-X-control-plane-6dgln                  poweredOn    6h7m
tkgs-cluster-X-control-plane-j6hq6                  poweredOn    6h10m
tkgs-cluster-X-control-plane-xc25f                  poweredOn    6h14m
tkgs-cluster-X-workers-9twdr-59bc54dc97-kt4cm       poweredOn    6h12m
tkgs-cluster-X-workers-9twdr-59bc54dc97-pjptr       poweredOn    6h12m
tkgs-cluster-X-workers-9twdr-59bc54dc97-t45mn       poweredOn    6h12m
```

3 작업자 노드 중 하나를 선택하고 다음 명령을 사용하여 설명합니다.

```
kubectl describe virtualmachines tkgs-cluster-X-workers-9twdr-59bc54dc97-kt4cm
```

4 가상 시스템의 IP 주소(예: `Vm Ip: 10.115.22.43`)를 찾습니다.

5 Grafana 확장 설치를 확인하려면 작업자 노드 IP 주소에 매핑된 Grafana FQDN으로 로컬 `/etc/hosts` 파일을 업데이트합니다. 예를 들면 다음과 같습니다.

```
127.0.0.1 localhost
127.0.1.1 ubuntu
# TKGS Grafana with Envoy NodePort
10.115.22.43 grafana.system.tanzu
```

6 https://grafana.system.tanzu로 이동하여 Grafana 대시보드에 액세스합니다.

사이트는 자체 서명된 인증서를 사용하기 때문에 대시보드에 액세스하려면 먼저 브라우저별 보안 주의를 거쳐 이동해야 할 수 있습니다.

Grafana 배포 문제 해결

배포 또는 조정이 실패하면 `kubectl get pods -A`를 실행하여 포드 상태를 확인합니다. `contour` 및 `envoy` 포드는 `Running`이어야 합니다. 포드 상태가 `ImagePullBackOff` 또는 `ImageCrashLoopBackOff`이면 컨테이너 이미지를 끌어올 수 없습니다. 데이터 값 및 확장 YAML 파일에서 레지스트리 URL을 확인하고 정확한지 확인합니다.

컨테이너 로그를 확인합니다. 여기서 name-XXXX는 kubectl get pods -A를 실행할 때 고유한 포드 이름입니다.

```
kubectl logs pod/grafana-XXXX -c grafana -n tanzu-system-monitoring
```

Grafana 확장 업데이트

Tanzu Kubernetes 클러스터에 배포된 Grafana 확장을 업데이트합니다.

1 grafana-data-values 암호에서 현재 Grafana 데이터 값을 얻습니다.

```
kubectl get secret grafana-data-values -n tanzu-system-monitoring -o 'go-
template={{ index .data "values.yaml" }}' | base64 -d > grafana-data-values.yaml
```

2 grafana-data-values.yaml에서 Grafana 데이터 값을 업데이트합니다. Grafana 확장 구성의 내용을 참조하십시오.

3 Grafana 데이터 값 암호를 업데이트합니다.

```
kubectl create secret generic grafana-data-values --from-file=values.yaml=grafana-data-
values.yaml -n tanzu-system-monitoring -o yaml --dry-run | kubectl replace -f-
```

Grafana 확장이 업데이트된 데이터 값으로 조정됩니다.

참고 기본적으로 kapp-controller는 5분마다 애플리케이션을 동기화합니다. 업데이트는 5분 이내에 적용됩니다. 업데이트를 즉시 적용하려면 grafana-extension.yaml의 syncPeriod를 더 작은 값으로 변경하고 kubectl apply -f grafana-extension.yaml을 사용하여 Grafana 확장을 적용합니다.

4 확장의 상태를 확인합니다.

```
kubectl get app grafana -n tanzu-system-monitoring
```

Grafana가 업데이트되면 상태가 Reconcile Succeeded로 변경됩니다.

5 자세한 상태를 살펴보고 필요한 경우 문제를 해결합니다.

```
kubectl get app grafana -n tanzu-system-monitoring -o yaml
```

Grafana 확장 삭제

Tanzu Kubernetes 클러스터에서 Grafana 확장을 삭제합니다.

참고 순서대로 단계를 완료하십시오. Grafana 애플리케이션이 완전히 삭제되기 전에 네임스페이스, 서비스 계정 및 역할 개체를 삭제하지 마십시오. 그러면 시스템 오류가 발생할 수 있습니다.

참고 Prometheus 및 Grafana 확장은 동일한 네임스페이스인 `tanzu-system-monitoring`에 배포됩니다. 두 확장을 동일한 클러스터에 배포한 경우 네임스페이스를 삭제하기 전에 각 확장을 삭제합니다.

1 디렉토리를 Grafana 확장으로 변경합니다.

```
cd /tkg-extensions-v1.3.1+vmware.1/extensions/monitoring/grafana
```

2 Grafana 애플리케이션을 삭제합니다.

```
kubectl delete app grafana -n tanzu-system-monitoring
```

예상 결과: `app.kappctrl.k14s.io "grafana" deleted`.

3 Grafana 애플리케이션이 삭제되었는지 확인합니다.

```
kubectl get app grafana -n tanzu-system-monintoring
```

예상 결과: `apps.kappctrl.k14s.io "grafana" not found`.

4 `tanzu-system-monitoring` 네임스페이스와 Grafana 서비스 계정 및 역할 개체를 삭제합니다.

경고 Prometheus가 배포된 경우 이 단계를 수행하지 마십시오.

```
kubectl delete -f namespace-role.yaml
```

5 Grafana를 다시 배포하려면 암호 `grafana-data-values`를 제거합니다.

```
kubectl delete secret grafana-data-values -n tanzu-system-monitoring
```

예상 결과: `secret "grafana-data-values" deleted`.

Grafana 확장 업그레이드

기존 Grafana 확장이 배포된 경우 최신 버전을 사용하도록 업그레이드할 수 있습니다.

1 Grafana configmap을 내보내서 백업으로 저장합니다.

```
kubectl get configmap grafana -n tanzu-system-monitoring -o 'go-template={{ index .data
"grafana.yaml" }}' > grafana-configmap.yaml
```

2 기존 Grafana 확장을 삭제합니다. Grafana 확장 삭제의 내용을 참조하십시오.

3 Grafana 확장을 배포합니다. 시각화 및 분석을 위해 Grafana 확장 배포의 내용을 참조하십시오.

Grafana 확장 구성

Grafana 구성은 `/tkg-extensions-v1.3.1+vmware.1/extensions/monitoring/grafana/grafana-data-values.yaml`에서 설정됩니다.

표 14-9. Grafana 구성 매개 변수

매개 변수	설명	유형	기본값
monitoring.namespace	Prometheus가 배포될 네임스페이스	문자열	tanzu-system-monitoring
monitoring.create_namespace	이 플래그는 monitoring.namespace를 통해 지정된 네임스페이스를 생성할지 여부를 나타냅니다.	부울	false
monitoring.grafana.cluster_role.apiGroups	grafana clusterrole에 대해 정의된 API 그룹	목록	[""]
monitoring.grafana.cluster_role.resources	grafana clusterrole에 대해 정의된 리소스	목록	["configmaps", "secrets"]
monitoring.grafana.cluster_role.verbs	clusterrole에 대해 정의된 액세스 권한	목록	["get", "watch", "list"]
monitoring.grafana.config.grafana_ini	Grafana 구성 파일 세부 정보	구성 파일	grafana.ini 이 파일에서 `grafana_net` URL은 Grafana와 통합하는 데 사용됨(예: Grafana.com에서 직접 대시보드 가져오기)
monitoring.grafana.config.datasource.type	Grafana 데이터소스 유형	문자열	prometheus
monitoring.grafana.config.datasource.access	액세스 모드. proxy 또는 direct(UI의 서버 또는 브라우저)	문자열	proxy
monitoring.grafana.config.datasource.isDefault	기본 Grafana 데이터소스로 표시	부울	true
monitoring.grafana.config.provider_yaml	Grafana 대시보드 제공자를 정의하는 구성 파일	yaml 파일	provider.yaml
monitoring.grafana.service.type	Grafana를 노출할 서비스 유형. 지원되는 값: ClusterIP, NodePort, LoadBalancer	문자열	vSphere: NodePort, aws/azure: LoadBalancer
monitoring.grafana.pvc.storage_class	영구 볼륨 할당에 대한 액세스 모드 정의. 지원되는 값: ReadWriteOnce, ReadOnlyMany, ReadWriteMany	문자열	ReadWriteOnce
monitoring.grafana.pvc.storage	영구 볼륨 할당에 대한 스토리지 크기 정의	문자열	2Gi
monitoring.grafana.deployment.replicas	Grafana 복제본 수	정수	1

표 14-9. Grafana 구성 매개 변수 (계속)

매개 변수	설명	유형	기본값
monitoring.grafana.image.repository	Grafana 이미지가 있는 저장소의 위치입니다. 기본값은 공용 VMware 레지스트리입니다. 비공개 저장소(예: 에어갭 환경)를 사용하는 경우 이 값을 변경합니다.	문자열	projects.registry.vmware.com/tkg/grafana
monitoring.grafana.image.name	Grafana 이미지의 이름	문자열	grafana
monitoring.grafana.image.tag	Grafana 이미지 태그. 버전을 업그레이드하는 경우 이 값을 업데이트해야 할 수 있습니다.	문자열	v7.3.5_vmware.1
monitoring.grafana.image.pullPolicy	Grafana 이미지 Pull 정책	문자열	IfNotPresent
monitoring.grafana.secret.type	Grafana 대시보드에 대해 정의된 암호 유형	문자열	불투명
monitoring.grafana.secret.admin_user	Grafana 대시보드에 액세스하기 위한 사용자 이름	문자열	YWRtaW4= 값이 base64로 인코딩됨. 디코딩하려면: `echo "xxxxxx" | base64 --decode`
monitoring.grafana.secret.admin_password	Grafana 대시보드에 액세스하기 위한 암호	문자열	null
monitoring.grafana.secret.ldap_toml	ldap auth를 사용하는 경우 ldap 구성 파일 경로	문자열	""
monitoring.grafana_init_container.image.repository	grafana init 컨테이너 이미지가 포함된 저장소. 기본값은 공용 VMware 레지스트리입니다. 비공개 저장소(예: 에어갭 환경)를 사용하는 경우 이 값을 변경합니다.	문자열	projects.registry.vmware.com/tkg/grafana
monitoring.grafana_init_container.image.name	grafana init 컨테이너 이미지의 이름	문자열	k8s-sidecar
monitoring.grafana_init_container.image.tag	grafana init 컨테이너 이미지 태그. 버전을 업그레이드하는 경우 이 값을 업데이트해야 할 수 있습니다.	문자열	0.1.99
monitoring.grafana_init_container.image.pullPolicy	grafana init 컨테이너 이미지 풀 정책	문자열	IfNotPresent
monitoring.grafana_sc_dashboard.image.repository	Grafana 대시보드 이미지가 포함된 저장소. 기본값은 공용 VMware 레지스트리입니다. 비공개 저장소(예: 에어갭 환경)를 사용하는 경우 이 값을 변경합니다.	문자열	projects.registry.vmware.com/tkg/grafana

표 14-9. Grafana 구성 매개 변수 (계속)

매개 변수	설명	유형	기본값
monitoring.grafana_sc_dashboard.image.name	Grafana 대시보드 이미지의 이름	문자열	k8s-sidecar
monitoring.grafana_sc_dashboard.image.tag	Grafana 대시보드 이미지 태그. 버전을 업그레이드하는 경우 이 값을 업데이트해야 할 수 있습니다.	문자열	0.1.99
monitoring.grafana_sc_dashboard.image.pullPolicy	Grafana 대시보드 이미지 Pull 정책	문자열	IfNotPresent
monitoring.grafana.ingress.enabled	grafana에 수신 사용/사용 안 함	부울	true
monitoring.grafana.ingress.virtual_host_fqdn	Grafana에 액세스하기 위한 호스트 이름	문자열	grafana.system.tanzu
monitoring.grafana.ingress.prefix	Grafana의 경로 접두사	문자열	/
monitoring.grafana.ingress.tlsCertificate.tls.crt	자체 TLS 인증서를 사용하려는 경우 수신을 위한 선택적 인증서. 자체 서명된 인증서가 기본적으로 생성됨	문자열	생성된 인증서
monitoring.grafana.ingress.tlsCertificate.tls.key	자체 TLS 인증서를 사용하려는 경우, 수신을 위한 선택적 인증서 개인 키.	문자열	생성된 인증서 키

Harbor 레지스트리용 TKG 확장 배포 및 관리

Harbor 는 오픈 소스 컨테이너 레지스트리입니다. Tanzu Kubernetes 클러스터에 배포하려는 컨테이너 이미지의 개인 레지스트리 저장소로 Harbor 레지스트리용 TKG 확장을 배포할 수 있습니다.

Harbor 확장 버전 종속성

Tanzu Kubernetes Grid 서비스에서 프로비저닝된 Tanzu Kubernetes 클러스터에 Harbor 레지스트리용 TKG 확장을 설치하려면 다음 최소 버전 요구 사항을 준수합니다.

구성 요소	최소 버전
vCenter Server	7.0.2.00400
vSphere 네임스페이스	0.0.10-18245956
감독자 클러스터	v1.20.2+vmware.1-vsc0.0.10-18245956
Tanzu Kubernetes 릴리스	v1.20.7+vmware.1-tkg.1.7fb9067

Harbor 확장 사전 요구 사항

Harbor 레지스트리용 TKG 확장 v1.3.1을 배포하기 전에 다음 사전 요구 사항을 준수합니다.

- 클러스터를 프로비저닝합니다. TKGS v1alpha2 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하는 워크플로의 내용을 참조하십시오.
- 클러스터에 연결합니다. vCenter Single Sign-On 사용자로 Tanzu Kubernetes 클러스터에 연결의 내용을 참조하십시오.
- kubectl 명령을 실행하는 클라이언트 호스트에 TKG 확장 v1.3.1 번들 다운로드합니다.
- 대상 클러스터에 TKG 확장 사전 요구 사항 설치합니다.

Harbor 확장 추가 요구 사항

Harbor 레지스트리용 TKG 확장 v1.3.1에는 설치 전후에 추가적인 요구 사항이 있습니다.

- Harbor 확장에는 기본 PVC 스토리지 클래스가 필요합니다. TKG 확장에 대한 영구 스토리지 요구 사항 검토의 내용을 참조하십시오.
- Harbor 확장에는 HTTP/S 수신이 필요합니다. 특히 Harbor 서비스는 Contour 확장의 엔보이 서비스를 통해 노출됩니다. 사전 요구 사항으로 Contour 확장을 배포합니다. Contour 수신용 TKG 확장 배포 및 관리의 내용을 참조하십시오.
  - 감독자 클러스터에 NSX-T 네트워킹을 사용하는 경우 LoadBalancer 유형의 엔보이 서비스를 생성합니다.
  - 감독자 클러스터에 vSphere vDS 네트워킹을 사용하는 경우에는 환경에 따라 LoadBalancer 유형 또는 NodePort 유형의 엔보이 서비스를 생성합니다.
- Harbor 확장에는 DNS가 필요합니다. Harbor 확장을 설치한 후에는 DNS를 구성해야 합니다.
  - 테스트 및 확인을 위해 Harbor 및 Notary FQDN을 로컬 `/etc/hosts` 파일에 추가합니다. 이 작업을 수행하는 방법은 아래 지침에 설명되어 있습니다.
  - 운영 환경에서 Harbor는 로컬 DNS 서버(예: BIND) 또는 공용 클라우드(예: AWS Route53, Azure DNS 또는 Google CloudDNS)에 DNS 영역이 필요합니다. DNS를 설정한 후 Harbor FQDN을 DNS 서버에 자동으로 등록하려면 외부 DNS 확장을 설치합니다. 외부 DNS 서비스 검색을 위한 TKG 확장 배포 및 관리의 내용을 참조하십시오.

Harbor 확장 배포

Harbor 레지스트리용 TKG 확장은 클러스터에 여러 컨테이너를 설치합니다. 자세한 내용은 https://goharbor.io/의 내용을 참조하십시오.

컨테이너	리소스 유형	복제	설명
harbor-core	배포	1	엔보이용 관리 및 구성 서버
harbor-database	포드	1	Postgres 데이터베이스
harbor-jobservice	배포	1	Harbor 작업 서비스

컨테이너	리소스 유형	복제	설명
harbor-notary-server	배포	1	Harbor Notary 서비스
harbor-notary-signer	배포	1	Harbor Notary
harbor-portal	배포	1	Harbor 웹 인터페이스
harbor-redis	포드	1	Harbor Redis 인스턴스
harbor-registry	배포	2	Harbor 컨테이너 레지스트리 인스턴스
harbor-trivy	포드	1	Harbor 이미지 취약성 스캐너

TKG 확장을 사용하여 Harbor 레지스트리를 설치하려면 다음 단계를 완료합니다.

1 각각의 확장 사전 요구 사항을 완료했는지 확인합니다. Harbor 확장 사전 요구 사항 및 Harbor 확장 추가 요구 사항의 내용을 참조하십시오.

2 디렉토리를 Harbor 확장으로 변경합니다.

```
cd /tkg-extensions-v1.3.1+vmware.1/extensions/registry/harbor
```

3 tanzu-system-registry 네임스페이스와 Harbor 서비스 계정 및 역할을 생성합니다.

```
kubectl apply -f namespace-role.yaml
```

4 Harbor 데이터 값 파일을 생성합니다.

```
cp harbor-data-values.yaml.example harbor-data-values.yaml
```

5 harbor-data-values.yaml에 필수 암호를 지정합니다.

Harbor 레지스트리에 필요한 몇 가지 암호가 아래 표에 나열되고 설명되어 있습니다.

암호	설명
harborAdminPassword	Harbor 관리자의 초기 암호입니다.
secretKey	암호화에 사용되는 비밀 키입니다. 16자의 문자열이어야 합니다.
database.password	Postgres 데이터베이스의 초기 암호입니다.
core.secret	암호는 코어 서버가 다른 구성 요소와 통신할 때 사용됩니다.
core.xsrfKey	XSRF 키입니다. 32자의 문자열이어야 합니다.
jobservice.secret	암호는 작업 서비스가 다른 구성 요소와 통신할 때 사용됩니다.
registry.secret	암호는 클라이언트 및 레지스트리 스토리지 백엔드에서 업로드 상태를 보호하는 데 사용됩니다.

임의의 암호를 자동으로 생성하고 harbor-data-values.yaml 파일을 채우려면 다음 명령을 실행합니다.

```
bash generate-passwords.sh harbor-data-values.yaml
```

성공하면 다음 메시지가 표시됩니다.

```
Successfully generated random passwords and secrets in harbor-data-values.yaml
```

harbor-data-values.yaml 파일을 열고 필수 암호를 확인합니다.

6 필요한 경우 harbor-data-values.yaml에 다른 Harbor 구성 값을 지정합니다. 일반적으로 업데이트되는 값에는 다음이 포함될 수 있습니다.

구성 필드	설명
hostname	기본 Harbor 호스트 이름은 core.harbor.domain입니다. 필요한 경우 요구 사항에 맞게 이 값을 변경합니다.
port.https	기본값은 443입니다. 감독자 클러스터에 NSX-T 네트워킹을 사용하며 그 결과 LoadBalancer 유형의 엔보이 수신 서비스를 사용하는 경우 이 설정을 기본값 443으로 둡니다. 감독자 클러스터에 vDS 네트워킹을 사용하며 그 결과 NodePort 유형의 엔보이 수신 서비스를 사용하는 경우에는 이 값을 엔보이 노드 포트와 일치하도록 설정합니다.
clair.enabled	Clair 이미지 스캐너는 Trivy를 위해 더 이상 사용되지 않습니다. 둘 다 구성 파일에서 사용하도록 설정됩니다. 값을 false로 설정하여 Clair를 사용하지 않도록 설정합니다.
persistence.persistentVolumeClaim.<component>.accessMode	이 설정에는 몇 가지 인스턴스가 있습니다. 기본값은 ReadWriteOnce입니다. ReadWriteMany는 향후 릴리스에서 지원될 예정입니다.
imageChartStorage.type	기본값은 filesystem입니다. 필요한 경우 변경하고 사용 중인 스토리지를 구성합니다.
proxy	원하는 경우 Harbor에 대한 프록시를 구성합니다. 프록시가 구성된 경우 기본 noProxy 값이 필요합니다.

7 데이터 값으로 암호를 생성합니다.

```
kubectl create secret generic harbor-data-values --from-file=values.yaml=harbor-data-values.yaml -n tanzu-system-registry
```

tanzu-system-registry 네임스페이스에 secret/harbor-data-values가 생성됩니다. 다음 명령을 실행하여 이를 확인합니다.

```
kubectl get secrets -n tanzu-system-registry
```

8 Harbor 확장을 배포합니다.

```
kubectl apply -f harbor-extension.yaml
```

성공하면 app.kappctrl.k14s.io/harbor created가 표시됩니다.

9 Harbor 애플리케이션의 상태를 확인합니다.

```
kubectl get app harbor -n tanzu-system-registry
```

성공하면 상태가 `Reconciling`에서 `Reconcile succeeded`로 변경됩니다.

```
NAME     DESCRIPTION   SINCE-DEPLOY   AGE
harbor   Reconciling   96s            98s
```

```
NAME     DESCRIPTION           SINCE-DEPLOY   AGE
harbor   Reconcile succeeded   39s            2m29s
```

상태가 `Reconcile failed`인 경우 Harbor 레지스트리 배포 문제 해결 항목을 참조하십시오.

10 Harbor 확장에 대한 자세한 정보를 봅니다.

```
kubectl get app harbor -n tanzu-system-registry -o yaml
```

11 Harbor 배포 개체의 상태를 봅니다.

```
kubectl get deployments -n tanzu-system-registry
```

성공하면 다음 배포가 표시됩니다.

```
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
harbor-core           1/1     1            1           5m16s
harbor-jobservice     1/1     1            1           5m16s
harbor-notary-server  1/1     1            1           5m16s
harbor-notary-signer  1/1     1            1           5m16s
harbor-portal         1/1     1            1           5m16s
harbor-registry       1/1     1            1           5m16s
```

12 Harbor 포드의 상태를 봅니다.

```
kubectl get pods -n tanzu-system-registry
```

```
NAME                                 READY   STATUS    RESTARTS   AGE
harbor-core-9cbf4b79d-gxvgx          1/1     Running   0          7m11s
harbor-database-0                    1/1     Running   0          7m11s
harbor-jobservice-6b656ccb95-lm47d   1/1     Running   0          7m11s
harbor-notary-server-8494c684db-gm7jf  1/1   Running   0          7m11s
harbor-notary-signer-6f96b549d4-dzcnm  1/1   Running   0          7m11s
harbor-portal-5b8f4ddbd-qdnp2        1/1     Running   0          7m11s
harbor-redis-0                       1/1     Running   0          7m11s
harbor-registry-688894c58d-72txm     2/2     Running   0          7m11s
harbor-trivy-0                       1/1     Running   0          7m11s
```

13 필요한 경우 Harbor 설치 문제를 해결합니다. Harbor 레지스트리 배포 문제 해결의 내용을 참조하십시오.

LoadBalancer 유형의 엔보이 서비스를 사용하여 Harbor용 DNS 구성(NSX-T 네트워킹)

사전 요구 사항 엔보이 서비스가 LoadBalancer를 통해 노출되는 경우 로드 밸런서의 외부 IP 주소를 가져오고 Harbor FQDN에 대한 DNS 레코드를 생성합니다.

1 LoadBalancer 유형의 엔보이 서비스에 대한 `External-IP` 주소를 가져옵니다.

```
kubectl get service envoy -n tanzu-system-ingress
```

반환된 `External-IP` 주소가 다음과 같이 표시됩니다.

```
NAME    TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)                      AGE
envoy   LoadBalancer   10.99.25.220   10.195.141.17   80:30437/TCP,443:30589/TCP   3h27m
```

또는 다음 명령을 사용하여 `External-IP` 주소를 가져올 수 있습니다.

```
kubectl get svc envoy -n tanzu-system-ingress -o
jsonpath='{.status.loadBalancer.ingress[0]}'
```

2 Harbor 확장 설치를 확인하려면 로드 밸런서의 `External-IP` 주소에 매핑된 Harbor 및 Notary FQDN으로 로컬 `/etc/hosts` 파일을 업데이트합니다. 예를 들면 다음과 같습니다.

```
127.0.0.1 localhost
127.0.1.1 ubuntu
# TKGS Harbor with Envoy Load Balancer IP
10.195.141.17 core.harbor.domain
10.195.141.17 core.notary.harbor.domain
```

3 Harbor 확장 설치를 확인하려면 Harbor에 로그인합니다. Harbor 웹 인터페이스에 로그인의 내용을 참조하십시오.

4 엔보이 서비스 로드 밸런서 `External-IP` 주소를 Harbor FQDN 및 Notary FQDN에 매핑하는 2개의 CNAME 레코드를 DNS 서버에 생성합니다.

5 외부 DNS 확장을 설치합니다. 외부 DNS 서비스 검색을 위한 TKG 확장 배포 및 관리의 내용을 참조하십시오.

NodePort 유형의 엔보이 서비스를 사용하여 Harbor용 DNS 구성(vDS 네트워킹)

사전 요구 사항 엔보이 서비스가 NodePort를 통해 노출되는 경우 작업자 노드의 가상 시스템 IP 주소를 가져오고 Harbor FQDN에 대한 DNS 레코드를 생성합니다.

참고 NodePort를 사용하려면 `harbor-data-values.yaml` 파일에 올바른 `port.https` 값을 지정해야 합니다.

1 클러스터가 프로비저닝된 vSphere 네임스페이스로 컨텍스트를 전환합니다.

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

2 클러스터의 노드를 나열합니다.

```
kubectl get virtualmachines
```

클러스터 노드가 표시됩니다. 예를 들면 다음과 같습니다.

```
NAME                                          POWERSTATE   AGE
tkgs-cluster-X-control-plane-6dgln            poweredOn    6h7m
tkgs-cluster-X-control-plane-j6hq6            poweredOn    6h10m
tkgs-cluster-X-control-plane-xc25f            poweredOn    6h14m
tkgs-cluster-X-workers-9twdr-59bc54dc97-kt4cm poweredOn    6h12m
tkgs-cluster-X-workers-9twdr-59bc54dc97-pjptr poweredOn    6h12m
tkgs-cluster-X-workers-9twdr-59bc54dc97-t45mn poweredOn    6h12m
```

3 작업자 노드 중 하나를 선택하고 다음 명령을 사용하여 설명합니다.

```
kubectl describe virtualmachines tkgs-cluster-X-workers-9twdr-59bc54dc97-kt4cm
```

4 가상 시스템의 IP 주소(예: `Vm Ip: 10.115.22.43`)를 찾습니다.

5 Harbor 확장 설치를 확인하려면 작업자 노드 IP 주소에 매핑된 Harbor 및 Notary FQDN으로 로컬 `/etc/hosts` 파일을 업데이트합니다. 예를 들면 다음과 같습니다.

```
127.0.0.1 localhost
127.0.1.1 ubuntu
# TKGS Harbor with Envoy NodePort
10.115.22.43 core.harbor.domain
10.115.22.43 core.notary.harbor.domain
```

6 Harbor 확장 설치를 확인하려면 Harbor에 로그인합니다. Harbor 웹 인터페이스에 로그인의 내용을 참조하십시오.

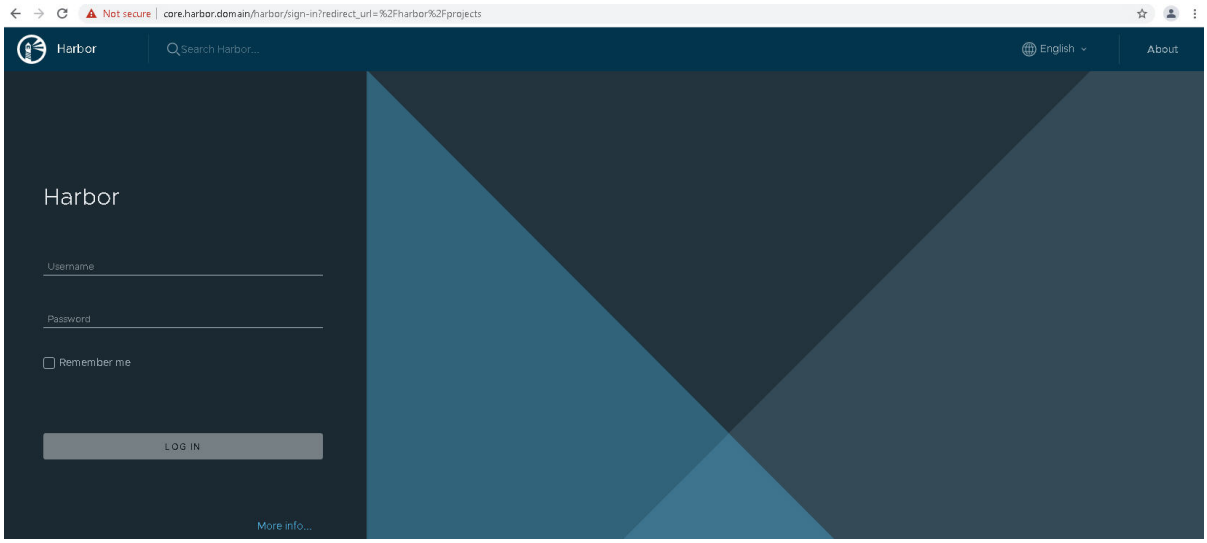
7 작업자 노드 IP 주소를 Harbor FQDN 및 Notary FQDN에 매핑하는 2개의 CNAME 레코드를 DNS 서버에 생성합니다.

8 외부 DNS 확장을 설치합니다. 외부 DNS 서비스 검색을 위한 TKG 확장 배포 및 관리의 내용을 참조하십시오.

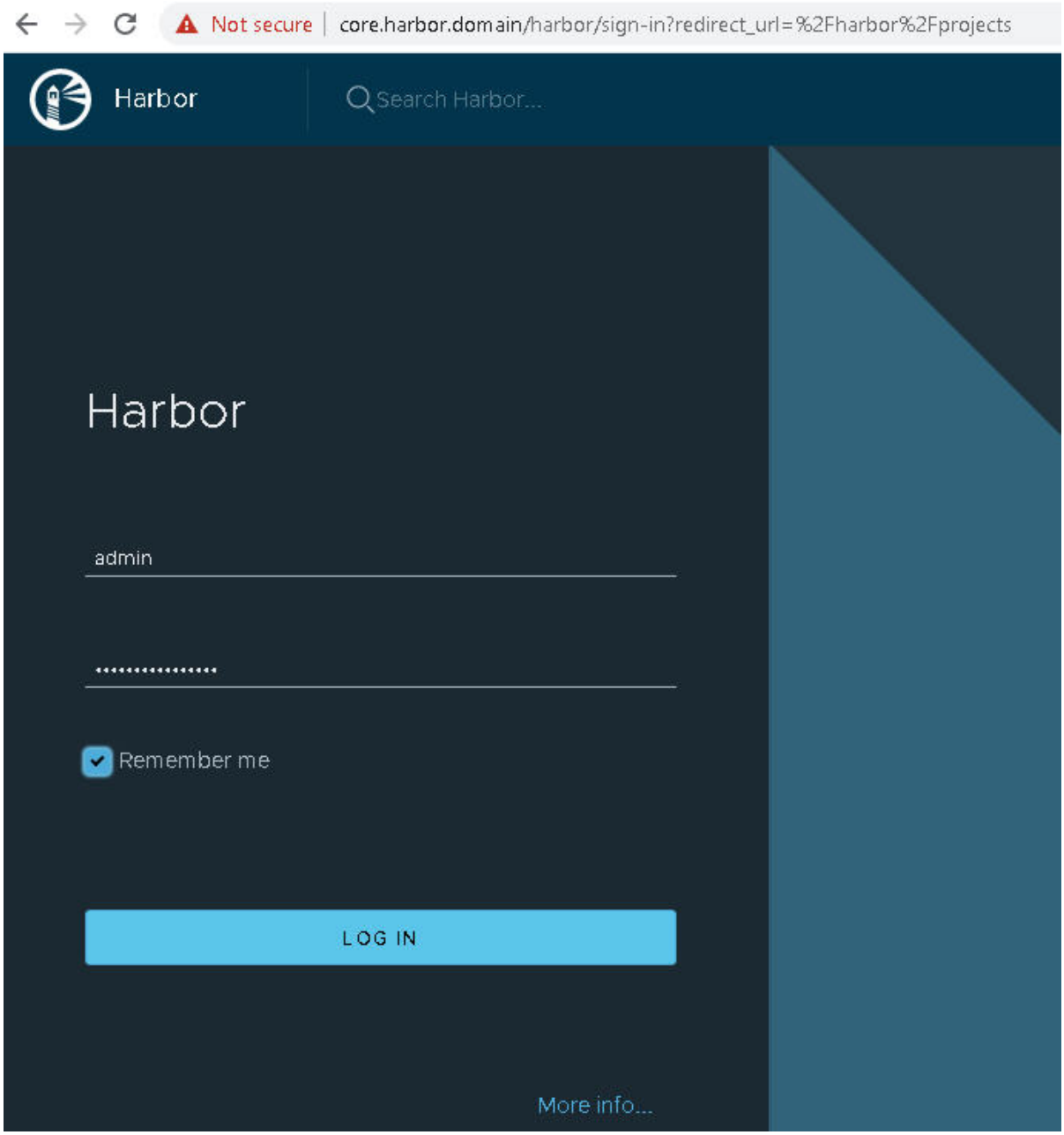
Harbor 웹 인터페이스에 로그인

Harbor가 설치 및 구성되면 로그인하여 사용을 시작합니다.

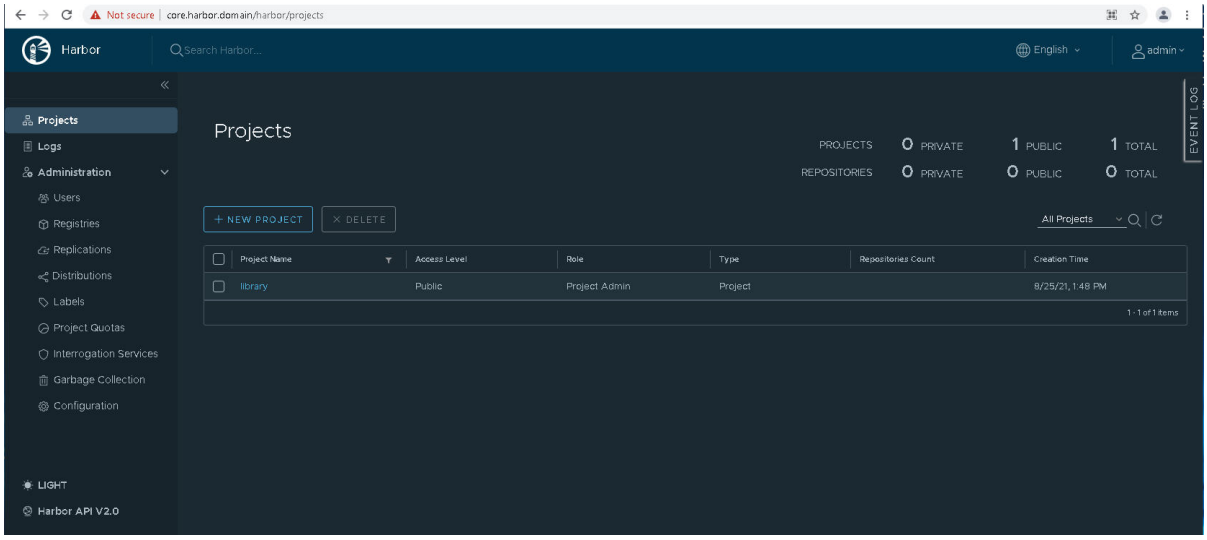
1 Harbor 레지스트리 웹 인터페이스(https://core.harbor.domain) 또는 사용한 호스트 이름에 액세스합니다.



2 `harbor-data-values.yaml` 파일에 입력한 사용자 이름 **`admin`**과 생성된 암호를 사용하여 Harbor에 로그인합니다.



3 Harbor 사용자 인터페이스에 액세스할 수 있는지 확인합니다.



4 Harbor CA 인증서를 가져옵니다.

Harbor 인터페이스에서 **프로젝트 > 라이브러리**를 선택하거나 **새 프로젝트**를 생성합니다.

레지스트리 인증서를 클릭하고 Harbor CA 인증서(`ca.crt`)를 다운로드합니다.

5 Harbor 레지스트리에서 컨테이너 이미지를 푸시하고 끌어올 수 있도록 Harbor CA 인증서를 Docker 클라이언트의 신뢰 저장소에 추가합니다. 내장된 Harbor 레지스트리 인증서를 사용하여 Docker 클라이언트 구성의 내용을 참조하십시오.

6 Harbor 사용에 대한 자세한 내용은 Harbor 설명서를 참조하십시오.

Harbor 레지스트리 배포 문제 해결

배포 또는 조정이 실패하면 `kubectl get pods -n tanzu-system-registry`를 실행하여 포드 상태를 확인합니다. `harbor` 포드는 `Running`이어야 합니다. 포드 상태가 `ImagePullBackOff` 또는 `ImageCrashLoopBackOff`이면 컨테이너 이미지를 끌어올 수 없습니다. 데이터 값 및 확장 YAML 파일에서 레지스트리 URL을 확인하고 정확한지 확인합니다.

컨테이너 로그를 확인합니다. 여기서 `name-XXXX`는 `kubectl get pods -A`를 실행할 때 고유한 포드 이름입니다.

```
kubectl logs pod/harbor-XXXXX -c harbor -n tanzu-system-registry
```

Harbor 확장 업데이트

Tanzu Kubernetes 클러스터에 배포된 Contour 확장을 업데이트합니다.

1 암호에서 Harbor 데이터 값을 얻습니다.

```
kubectl get secret harbor-data-values -n tanzu-system-registry -o 'go-
template={{ index .data "values.yaml" }}' | base64 -d > harbor-data-values.yaml
```

2 `harbor-data-values.yaml`에서 Harbor 데이터 값을 업데이트합니다.

3 Harbor 데이터 값 암호를 업데이트합니다.

```
kubectl create secret generic harbor-data-values --from-file=values.yaml=harbor-data-values.yaml -n tanzu-system-registry -o yaml --dry-run | kubectl replace -f-
```

Harbor 확장이 새 데이터 값으로 조정됩니다.

참고 기본적으로 kapp-controller는 5분마다 애플리케이션을 동기화합니다. 업데이트는 5분 이내에 적용됩니다. 업데이트를 즉시 적용하려면 `harbor-extension.yaml`의 `syncPeriod`를 더 작은 값으로 변경하고 `kubectl apply -f harbor-extension.yaml`을 사용하여 Contour 확장을 적용합니다.

4 확장의 상태를 확인합니다.

```
kubectl get app harbor -n tanzu-system-registry
```

Contour가 업데이트되면 Contour 애플리케이션 상태가 `Reconcile Succeeded`로 변경됩니다.

5 자세한 상태를 살펴보고 문제를 해결합니다.

```
kubectl get app harbor -n tanzu-system-registry -o yaml
```

Harbor 확장 삭제

Tanzu Kubernetes 클러스터에서 Harbor 확장을 삭제합니다.

참고 순서대로 단계를 완료하십시오. Contour 확장 및 애플리케이션이 삭제되기 전에 Contour 네임스페이스 및 역할 개체를 삭제하지 마십시오. Contour 네임스페이스 및 역할 개체를 삭제하면 kapp-controller에서 사용하는 서비스 계정이 삭제됩니다. 애플리케이션 및 확장을 삭제하기 전에 이 서비스 계정을 삭제하면 시스템 오류가 발생할 수 있습니다.

1 디렉토리를 Harbor 확장 파일을 다운로드한 위치로 변경합니다.

```
cd /extensions/registry/harbor/
```

2 Harbor 애플리케이션을 삭제합니다.

```
kubectl delete app harbor -n tanzu-system-registry
```

예상 결과:

```
app.kappctrl.k14s.io "harbor" deleted
```

3 Harbor 애플리케이션이 삭제되었는지 확인합니다.

```
kubectl get app Harbor -n tanzu-system-registry
```

예상 결과: 애플리케이션이 `Not Found` 상태입니다.

```
apps.kappctrl.k14s.io "harbor" not found
```

4 레지스트리 네임스페이스를 삭제합니다.

Harbor 확장 및 애플리케이션이 완전히 삭제된 것을 확인한 후에만 네임스페이스 및 역할 개체를 삭제해도 안전합니다.

```
kubectl delete -f namespace-role.yaml
```

예상 결과: Harbor가 배포된 네임스페이스 및 연결된 역할 기반 액세스 제어 개체가 삭제됩니다.

```
namespace "tanzu-system-registry" deleted
serviceaccount "harbor-extension-sa" deleted
role.rbac.authorization.k8s.io "harbor-extension-role" deleted
rolebinding.rbac.authorization.k8s.io "harbor-extension-rolebinding" deleted
clusterrole.rbac.authorization.k8s.io "harbor-extension-cluster-role" deleted
clusterrolebinding.rbac.authorization.k8s.io "harbor-extension-cluster-rolebinding" deleted
```

Harbor 확장 업그레이드

기존 Harbor 확장이 배포된 경우 최신 버전으로 업그레이드할 수 있습니다.

1 Harbor configmap을 얻습니다.

```
kubectl get configmap harbor -n tanzu-system-harbor -o 'go-template={{ index .data
"harbor.yaml" }}' > harbor-configmap.yaml
```

2 기존 Harbor 배포를 삭제합니다. Harbor 확장 삭제의 내용을 참조하십시오.

3 Harbor 확장을 배포합니다. Harbor 확장 배포의 내용을 참조하십시오.

외부 DNS 서비스 검색을 위한 TKG 확장 배포 및 관리

외부 DNS를 사용하면 Kubernetes 로드 밸런싱된 서비스를 기반으로 DNS 레코드를 동적으로 구성할 수 있습니다. 외부 DNS용 TKG 확장을 배포하여 클러스터에 대한 동적 서비스 검색을 제공할 수 있습니다.

확장 사전 요구 사항

외부 DNS용 TKG 확장 v1.3.1을 배포하기 전에 다음 요구 사항을 준수합니다.

- Tanzu Kubernetes 클러스터를 프로비저닝합니다. TKGS v1alpha2 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하는 워크플로의 내용을 참조하십시오.
- Tanzu Kubernetes 클러스터에 연결합니다. vCenter Single Sign-On 사용자로 Tanzu Kubernetes 클러스터에 연결의 내용을 참조하십시오.
- kubectl을 실행하는 클라이언트 호스트에 TKG 확장 v1.3.1 번들 다운로드합니다.
- 대상 Tanzu Kubernetes 클러스터에 TKG 확장 사전 요구 사항 설치합니다.

기타 요구 사항

외부 DNS 확장과 함께 제공되는 샘플 구성에는 Contour 수신 컨트롤러가 있거나 없는 예제가 포함되어 있습니다. Contour를 사용하는 경우에는 외부 DNS 확장을 설치하기 전에 Contour를 설치합니다. Contour 수신용 TKG 확장 배포 및 관리를 참조하십시오.

외부 DNS 확장을 사용하면 동적 서비스 검색이 가능합니다. 일반적인 사용 사례는 Harbor 레지스트리입니다. Harbor를 사용하려면 RFC 2136 규격 동적 DNS 제공자(예: AWS Route53, Azure DNS, Google Cloud DNS) 또는 로컬 DNS 서버(예: BIND)에 DNS 영역이 설정되어 있어야 합니다. Harbor 레지스트리용 TKG 확장 배포 및 관리를 참조하십시오.

외부 DNS 확장 배포

다음 단계를 완료하여 외부 DNS용 TKG 확장 v1.3.1을 설치합니다.

1 디렉토리를 외부 DNS 확장 파일을 다운로드한 위치로 변경합니다.

```
cd /tkg-extensions-v1.3.1+vmware.1/extensions/service-discovery/external-dns
```

2 외부 DNS 확장에서 사용할 네임스페이스 및 여러 역할 기반 액세스 제어 개체를 생성합니다.

```
kubectl apply -f namespace-role.yaml
```

이 명령은 네임스페이스 `tanzu-system-service-discovery` 및 연결된 RBAC 개체를 생성합니다. `kubect get ns`를 실행하여 확인합니다.

3 데이터 값 파일을 생성합니다. 예제 데이터 값 파일은 필요한 최소 구성을 제공합니다.

AWS, Azure 및 RFC 2136 규격 동적 DNS 제공자에 대한 예제 데이터 값 파일(각각 Contour 수신이 있거나 없음)이 있습니다. 적절한 예제 파일을 선택하고 복사합니다.

예를 들어 AWS Route 53을 Contour와 함께 사용하는 경우 다음 명령을 실행합니다.

```
cp external-dns-data-values-aws-with-contour.yaml.example external-dns-data-values-aws-
with-contour.yaml
```

또는 Azure를 Contour와 함께 사용하는 경우 다음 명령을 실행합니다.

```
cp external-dns-data-values-azure-with-contour.yaml.example external-dns-data-values-azure-
with-contour.yaml
```

4 외부 DNS 데이터 값을 구성합니다.

예를 들어 다음은 Azure DNS에 대한 구성입니다. `domain-filter` 및 `azure-resource-group` 값을 제공합니다.

```
#@data/values
#@overlay/match-child-defaults missing_ok=True
---
externalDns:
  image:
    repository: projects.registry.vmware.com/tkg
  deployment:
    #@overlay/replace
    args:
    - --provider=azure
    - --source=service
```

```
    - --source=ingress
    - --domain-filter=my-zone.example.org #! zone where services are deployed
    - --azure-resource-group=my-resource-group #! Azure resource group
    #@overlay/replace
    volumeMounts:
    - name: azure-config-file
      mountPath: /etc/kubernetes
      readOnly: true
    #@overlay/replace
    volumes:
    - name: azure-config-file
      secret:
        secretName: azure-config-file
```

5 직접 채운 데이터 값 파일을 사용하여 일반 암호를 생성합니다.

예를 들어 다음 명령은 Azure DNS 데이터 값 파일을 사용하여 암호를 생성합니다.

```
kubectl create secret generic external-dns-data-values --from-file=values.yaml=external-dns-data-values-azure-with-contour.yaml -n tanzu-system-service-discovery
```

`tanzu-system-service-discovery` 네임스페이스에 `secret/external-dns-data-values created`가 생성된 것을 볼 수 있습니다. `kubectl get secrets -n tanzu-system-service-discovery` 명령을 사용하여 확인할 수 있습니다.

6 외부 DNS 확장을 배포합니다.

```
kubectl apply -f external-dns-extension.yaml
```

성공하면 `app.kappctrl.k14s.io/external-dns created`가 표시됩니다.

7 확장 배포의 상태를 확인합니다.

```
kubectl get app external-dns -n tanzu-system-service-discovery
```

외부 DNS가 성공적으로 배포되면 애플리케이션 상태가 `Reconciling`에서 `Reconcile succeeded`로 변경됩니다. 상태가 `Reconcile failed`인 경우 배포 문제 해결 항목을 참조하십시오.

8 자세한 상태를 봅니다.

```
kubectl get app external-dns -n tanzu-system-service-discovery -o yaml
```

배포 문제 해결

조정이 실패하면 `kubectl get pods -A` 명령을 실행하여 포드의 상태를 확인합니다. 정상적인 조건에서는 `external-dns-XXXXX` 포드가 `Running`인 것을 볼 수 있습니다. 조정이 실패하거나 포드 상태가 `ImagePullBackOff` 또는 `ImagecRashLoopBackOff`이면 저장소에서 컨테이너 이미지를 끌어올 수 없음을 의미합니다. 데이터 값 및 확장 YAML 파일에서 저장소 URL을 확인하고 정확한지 확인합니다.

컨테이너 로그를 확인하려면 다음 명령을 실행합니다. 여기서 name-XXXX는 kubectl get pods -A를 실행할 때 볼 수 있는 고유한 포드 이름입니다.

```
kubectl logs pod/external-dns-XXXXX -c external-dns -n tanzu-system-service-discovery
```

외부 DNS 확장 업데이트

Tanzu Kubernetes 클러스터에 배포된 외부 DNS 확장을 업데이트합니다.

1 암호에서 Contour 데이터 값을 얻습니다.

```
kubectl get secret external-dns-data-values -n tanzu-system-service-discovery -o 'go-template={{ index .data "values.yaml" }}' | base64 -d > external-dns-data-values.yaml
```

2 external-dns-data-values.yaml에서 외부 DNS 데이터 값을 업데이트합니다. 외부 DNS 확장 구성의 내용을 참조하십시오.

3 Contour 데이터 값 암호를 업데이트합니다.

```
kubectl create secret generic external-dns-data-values --from-file=values.yaml=external-dns-data-values.yaml -n tanzu-system-service-discovery -o yaml --dry-run | kubectl replace -f-
```

외부 DNS 확장이 새 데이터 값으로 조정됩니다.

참고 기본적으로 kapp-controller는 5분마다 애플리케이션을 동기화합니다. 업데이트는 5분 이내에 적용됩니다. 업데이트를 즉시 적용하려면 external-dns-extension의 syncPeriod를 더 작은 값으로 변경하고 kubectl apply -f external-dns-extension을 사용하여 Contour 확장을 적용합니다.

4 확장의 상태를 확인합니다.

```
kubectl get app external-dns -n tanzu-system-service-disovery
```

업데이트되면 애플리케이션 상태가 Reconcile Succeeded로 변경됩니다.

5 자세한 상태를 살펴보고 문제를 해결합니다.

```
kubectl get app external-dns -n tanzu-system-service-disovery -o yaml
```

외부 DNS 확장 삭제

Tanzu Kubernetes 클러스터에서 외부 DNS 확장을 삭제합니다.

참고 순서대로 단계를 완료하십시오. 확장 및 애플리케이션이 삭제되기 전에 대상 네임스페이스 및 역할 개체를 삭제하지 마십시오. 네임스페이스 및 역할 개체를 삭제하면 kapp-controller에서 사용하는 서비스 계정이 삭제됩니다. 애플리케이션 및 확장을 삭제하기 전에 이 서비스 계정을 삭제하면 시스템 오류가 발생할 수 있습니다.

1 디렉토리를 확장 파일을 다운로드한 위치로 변경합니다.

```
cd /tkg-extensions-v1.3.1+vmware.1/extensions/service-discovery/external-dns
```

2 외부 DNS 확장을 삭제합니다.

```
kubectl delete -f external-dns-extension.yaml
```

3 확장이 삭제되었는지 확인합니다.

```
kubectl get app contour -n tanzu-system-ingress
```

예상 결과: 애플리케이션이 `Not Found` 상태입니다.

4 네임스페이스를 삭제합니다.

Contour 확장 및 애플리케이션이 완전히 삭제된 것을 확인한 후에만 네임스페이스 및 역할 개체를 삭제해도 안전합니다.

```
kubectl delete -f namespace-role.yaml
```

예상 결과: 확장이 배포된 네임스페이스 및 연결된 역할 기반 액세스 제어 개체가 삭제됩니다.

외부 DNS 확장 구성

사용자 지정 설정을 사용하여 외부 DNS 확장을 구성할 수 있습니다.

외부 DNS 제공자에 대한 배포 매개 변수를 구성합니다. 추가 지침은 Kubernetes 사이트 https://github.com/kubernetes-sigs/external-dns#running-externaldns에서 참조하십시오.

표 14-10. Harbor 확장 구성 매개 변수

매개 변수	설명	유형	기본값
externalDns.namespace	external-dns가 배포될 네임스페이스	문자열	tanzu-system-service-discovery
externalDns.image.repository	external-dns 이미지가 포함된 저장소	문자열	projects.registry.vmware.com/tkg
externalDns.image.name	external-dns의 이름	문자열	external-dns
externalDns.image.tag	ExternalDNS 이미지 태그	문자열	v0.7.4_vmware.1

표 14-10. Harbor 확장 구성 매개 변수 (계속)

매개 변수	설명	유형	기본값
externalDns.image.pullPolicy	ExternalDNS 이미지 Pull 정책	문자열	IfNotPresent
externalDns.deployment.annotations	external-dns 배포에 대한 주석	map<string,string>	{}
externalDns.deployment.args	명령줄을 통해 external-dns로 전달된 인수	list<string>	[](필수 매개 변수)
externalDns.deployment.env	external-dns에 전달할 환경 변수	list<string>	[]
externalDns.deployment.securityContext	external-dns 컨테이너의 보안 컨텍스트	보안 컨텍스트	{}
externalDns.deployment.volumeMounts	external-dns 컨테이너의 볼륨 마운트	list<VolumeMount>	[]
externalDns.deployment.volumes	external-dns 포드의 볼륨	list<Volume>	[]

Tanzu Kubernetes 클러스터에 AI/ML 워크로드 배포

Tanzu Kubernetes Grid 서비스에서 프로비저닝된 Tanzu Kubernetes 클러스터에 AI/ML 워크로드를 배포할 수 있습니다. AI/ML 워크로드를 배포하려면 vSphere 관리자의 초기 설정과 클러스터 운영자의 일부 구성이 필요합니다. vSphere with Tanzu 환경에서 vGPU를 지원하는 경우 개발자는 AI/ML 워크로드를 Kubernetes 워크로드와 마찬가지로 해당 TKGS 클러스터에 배포할 수 있습니다.

TKGS 클러스터에 AI/ML 워크로드 배포 정보

vSphere with Tanzu 및 NVIDIA vGPU 기술을 사용하여 TKGS 클러스터에 AI/ML 워크로드를 배포할 수 있습니다.

AI/ML 워크로드에 대한 TGKS 지원 발표

vSphere with Tanzu 버전 7 업데이트 3 월간 패치 1 릴리스부터 Tanzu Kubernetes Grid 서비스에서 프로비저닝된 Tanzu Kubernetes 클러스터에 계산 집약적인 워크로드를 배포할 수 있습니다. 이 컨텍스트에서 계산 집약적인 워크로드는 GPU 가속기 디바이스를 사용해야 하는 AI(인공 지능) 또는 ML(기계 학습) 애플리케이션입니다.

Kubernetes 환경에서 AI/ML 워크로드의 실행을 용이하게 하기 위해 VMware는 NVIDIA와 제휴하여 vSphere with Tanzu에서 NVIDIA GPU Cloud 플랫폼을 지원합니다. 즉, Tanzu Kubernetes Grid 서비스에서 프로비저닝된 Tanzu Kubernetes 클러스터에 NGC 카탈로그의 컨테이너 이미지를 배포할 수 있습니다.

AI-Ready Enterprise를 위한 NVIDIA 및 VMware의 공동 아키텍처에 대해 자세히 알아보려면 vSphere 7 with Tanzu에서 워크로드 가속화 - GPU가 있는 Kubernetes 클러스터의 기술 미리 보기를 참조하십시오.

지원되는 vGPU 모드

TKGS에 AI/ML 워크로드를 배포하려면 vSphere with Tanzu 컨텐츠 전송 네트워크를 통해 사용할 수 있는 Ubuntu OVA를 사용해야 합니다. TKGS는 두 가지 GPU 작업 모드인 vGPU 및 NIC 패스스루가 포함된 vGPU를 지원합니다. 다음 표에는 이 두 가지 모드에 대한 자세한 설명이 나와 있습니다.

모드	구성	설명
NVIDIA + TKGS + Ubuntu + vGPU	NVIDIA vGPU	GPU 디바이스는 각 ESXi 호스트에 설치된 NVIDIA 호스트 관리자 드라이버에 의해 가상화됩니다. 그런 다음 GPU 디바이스는 여러 NVIDIA vGPU(가상 GPU)에서 공유됩니다. 각 NVIDIA vGPU는 GPU 디바이스의 메모리 양으로 정의됩니다. 예를 들어 GPU 디바이스의 총 RAM 용량이 32GB인 경우 각각 약 4GB의 메모리가 포함된 8개의 vGPU를 생성할 수 있습니다.
NVIDIA + TKGS + Ubuntu + vGPU + NIC 패스스루	NVIDIA vGPU **및** 동적 DirectPath IO	NVIDIA vGPU 프로파일을 구성하는 동일한 VM 클래스에서, 동적 DirectPath IO를 사용하는 패스스루 네트워킹 디바이스에 대한 지원을 포함합니다. 이 경우 vSphere DRS에 따라 VM 배치가 결정됩니다.

시작

TKGS에 대해 NVIDIA vGPU를 구성하려면 다음 항목을 참조하십시오.

- TKGS 클러스터(vGPU)에 AI/ML 워크로드를 배포하기 위한 vSphere 관리자 워크플로
- TKGS 클러스터에 AI/ML 워크로드를 배포하기 위한 클러스터 운영자 워크플로

NIC 패스스루가 포함된 vGPU를 사용하는 경우에는 TKGS 클러스터에 AI/ML 워크로드를 배포하기 위한 vSphere 관리자 부록(vGPU 및 동적 DirectPath IO) 항목도 참조하십시오.

NVAIE 계정에 대해 NVIDIA DLS(Delegated Licensing Server)를 사용하는 경우에는 TKGS 클러스터(DLS)에 AI/ML 워크로드를 배포하기 위한 클러스터 운영자 부록 항목도 참조하십시오.

TKGS 클러스터(vGPU)에 AI/ML 워크로드를 배포하기 위한 vSphere 관리자 워크플로

개발자가 TKGS 클러스터에 AI/ML 워크로드를 배포할 수 있도록 하려면 vSphere 관리자가 NVIDIA GPU 하드웨어를 지원하도록 vSphere with Tanzu 환경을 설정해야 합니다.

TKGS 클러스터에 AI/ML 워크로드를 배포하기 위한 vSphere 관리자 워크플로

vSphere 관리자가 TKGS 클러스터에 AI/ML 워크로드를 배포할 수 있도록 하는 개략적인 워크플로가 표에 나열되어 있습니다. 각 단계에 대한 세부 지침은 다음과 같습니다.

단계	작업	연결
0	시스템 요구 사항을 검토합니다.	관리자 0단계: 시스템 요구 사항 검토의 내용을 참조하십시오.
1	ESXi 호스트에 지원되는 NVIDIA GPU 디바이스를 설치합니다.	관리자 1단계: ESXi 호스트에 지원되는 NVIDIA GPU 디바이스 설치의 내용을 참조하십시오.

단계	작업	연결
2	vGPU 작업에 대한 ESXi 디바이스 그래픽 설정을 구성합니다.	관리자 2단계: vGPU 작업을 위한 각 ESXi 호스트 구성의 내용을 참조하십시오.
3	각 ESXi 호스트에 NVIDIA vGPU Manager(VIB)를 설치합니다.	관리자 3단계: 각 ESXi 호스트에 NVIDIA 호스트 관리자 드라이버 설치의 내용을 참조하십시오.
4	NVIDIA 드라이버 작업 및 GPU 가상화 모드를 확인합니다.	관리자 4단계: ESXi 호스트가 NVIDIA vGPU 작업을 수행할 준비가 되었는지 확인의 내용을 참조하십시오.
5	GPU 구성 클러스터에서 워크로드 관리를 사용하도록 설정합니다. 그러면 감독자 클러스터가 vGPU 지원 ESXi 호스트에서 실행됩니다.	관리자 5단계: vGPU 구성 vCenter 클러스터에서 워크로드 관리 사용의 내용을 참조하십시오.
6	Tanzu Kubernetes 릴리스에 대한 컨텐츠 라이브러리를 생성* 또는 업데이트하고 라이브러리를 vGPU 워크로드에 필요한 지원되는 Ubuntu OVA로 채웁니다.	관리자 6단계: Tanzu Kubernetes Ubuntu 릴리스로 컨텐츠 라이브러리 생성 또는 업데이트의 내용을 참조하십시오. **참고** *필요한 경우. TKGS 클러스터 Photon 이미지에 대한 컨텐츠 라이브러리가 이미 있는 경우 Ubuntu 이미지에 대한 새 컨텐츠 라이브러리를 생성하지 마십시오.
7	특정 vGPU 프로파일이 선택된 사용자 지정 VM 클래스를 생성합니다.	관리자 7단계: vGPU 프로파일을 사용하여 사용자 지정 VM 클래스 생성 항목을 참조하십시오.
8	TKGS GPU 클러스터에 대한 vSphere 네임스페이스를 생성하고 구성합니다. 편집 권한이 있는 사용자와 영구 볼륨에 대한 스토리지를 추가합니다.	관리자 8단계: TKGS GPU 클러스터에 대한 vSphere 네임스페이스 생성 및 구성 항목을 참조하십시오.
9	컨텐츠 라이브러리를 Ubuntu OVA와 연결하고 vGPU에 대한 사용자 지정 VM 클래스를 TGKS용으로 생성한 vSphere 네임스페이스와 연결합니다.	관리자 9단계: 컨텐츠 라이브러리 및 VM 클래스를 vSphere 네임스페이스와 연결 항목을 참조하십시오.
10	감독자 클러스터가 프로비저닝되었고 클러스터 운영자가 액세스할 수 있는지 확인합니다.	관리자 10단계: 감독자 클러스터에 액세스할 수 있는지 확인 항목을 참조하십시오.

관리자 0단계: 시스템 요구 사항 검토

TKGS 클러스터에 AI/ML 워크로드를 배포하기 위한 환경을 설정하려면 다음 시스템 요구 사항을 참조하십시오.

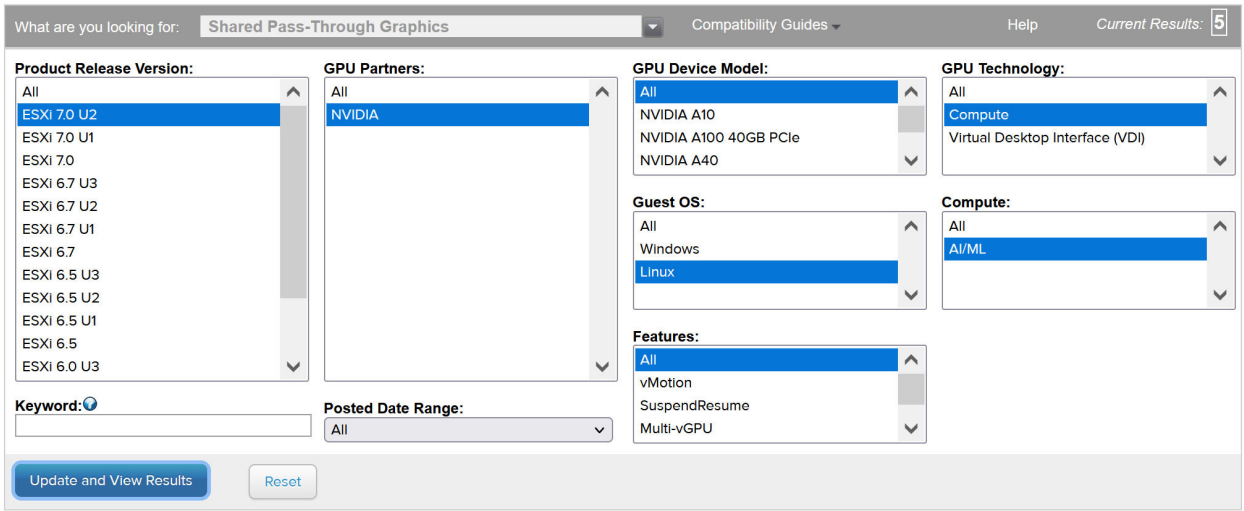
요구 사항	설명
vSphere 인프라	vSphere 7 업데이트 3 월간 패치 1 ESXi 빌드 `18778458` 이상 vCenter Server 빌드 `18644231` 이상
워크로드 관리	vSphere 네임스페이스 버전 `0.0.11-18610518` 이상
감독자 클러스터	감독자 클러스터 버전 `v1.21.0+vmware.1-vsc0.0.11-18610518` 이상

요구 사항	설명
TKR Ubuntu OVA	Tanzu Kubernetes 릴리스 Ubuntu `ob-18691651-tkgs-ova-ubuntu-2004-v1.20.8---vmware.1-tkg.2`
NVIDIA vGPU 호스트 드라이버	NGC 웹 사이트에서 VIB를 다운로드합니다. 자세한 내용은 vGPU 소프트웨어 드라이버 설명서를 참조하십시오. 예: `NVIDIA-AIE_ESXi_7.0.2_Driver_470.51-1OEM.702.0.0.17630552.vib`
vGPU용 NVIDIA 라이센스 서버	조직에서 제공한 FQDN

관리자 1단계: ESXi 호스트에 지원되는 NVIDIA GPU 디바이스 설치

TKGS에 AI/ML 워크로드를 배포하려면 **워크로드 관리**가 사용되도록 설정될 vCenter 클러스터를 구성하는 각 ESXi 호스트에 하나 이상의 지원되는 NVIDIA GPU 디바이스를 설치합니다.

호환되는 NVIDIA GPU 디바이스를 보려면 VMware 호환성 가이드를 참조하십시오.



Click here to **Read** Important Support Information

Click on the 'GPU Device Model' to view more details and to subscribe to RSS feeds.

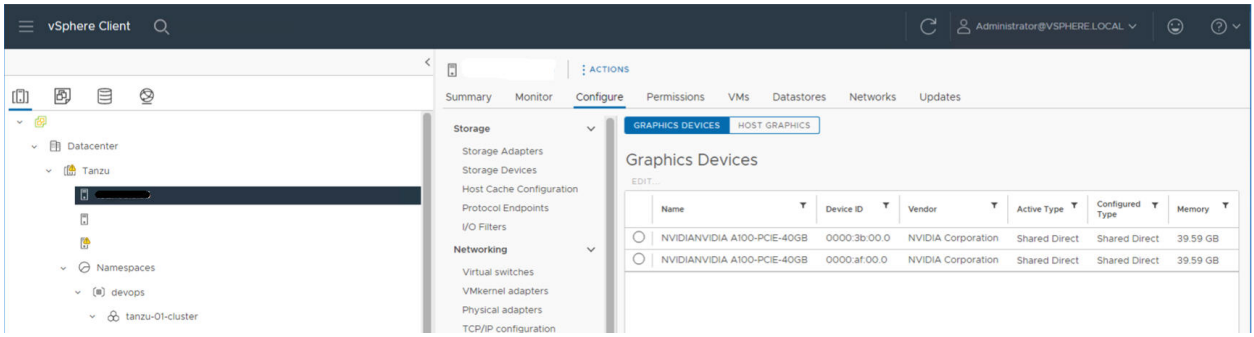
Bookmark | Print | Export to CSV

Search Results: Your search for" Shared Pass-Through Graphics " returned **5 results.** Back to Top Turn Off Auto Scroll Display: 10

GPU Partner	GPU Device Model	ESX Version	Compute
NVIDIA	NVIDIA A10	ESXi 7.0 U2	AI/ML
NVIDIA	NVIDIA A100 40GB PCIe	ESXi 7.0 U2	AI/ML
NVIDIA	NVIDIA A100 80GB PCIe	ESXi 7.0 U2	AI/ML
NVIDIA	NVIDIA A30	ESXi 7.0 U2	AI/ML
NVIDIA	NVIDIA A40	ESXi 7.0 U2	AI/ML

NVIDA GPU 디바이스는 최신 NVAIE(NVIDIA AI Enterprise) vGPU 프로파일을 지원해야 합니다. 지침을 보려면 NVIDIA 가상 GPU 소프트웨어 지원 GPU 설명서를 참조하십시오.

예를 들어 다음 ESXi 호스트에는 두 개의 NVIDIA GPU A100 디바이스가 설치되어 있습니다.



관리자 2단계: vGPU 작업을 위한 각 ESXi 호스트 구성

Shared Direct 및 SR-IOV를 사용하도록 설정하여 vGPU에 대해 각 ESXi 호스트를 구성합니다.

각 ESXi 호스트에서 Shared Direct 사용

NVIDIA vGPU 기능의 잠금을 해제하려면 **워크로드 관리**가 사용되도록 설정될 vCenter 클러스터를 구성하는 각 ESXi 호스트에서 **Shared Direct** 모드를 사용하도록 설정합니다.

Shared Direct를 사용하도록 설정하려면 다음 단계를 완료하십시오. 추가 지침은 vSphere 설명서에서 그래픽 디바이스 구성을 참조하십시오.

1 vSphere Client를 사용하여 vCenter Server에 로그인합니다.

2 vCenter 클러스터에서 ESXi 호스트를 선택합니다.

3 **구성 > 하드웨어 > 그래픽**을 선택합니다.

4 NVIDIA GPU 가속기 디바이스를 선택합니다.

5 그래픽 디바이스 설정을 **편집**합니다.

6 **Shared Direct**를 선택합니다.

7 **X.Org 서버 다시 시작**을 선택합니다.

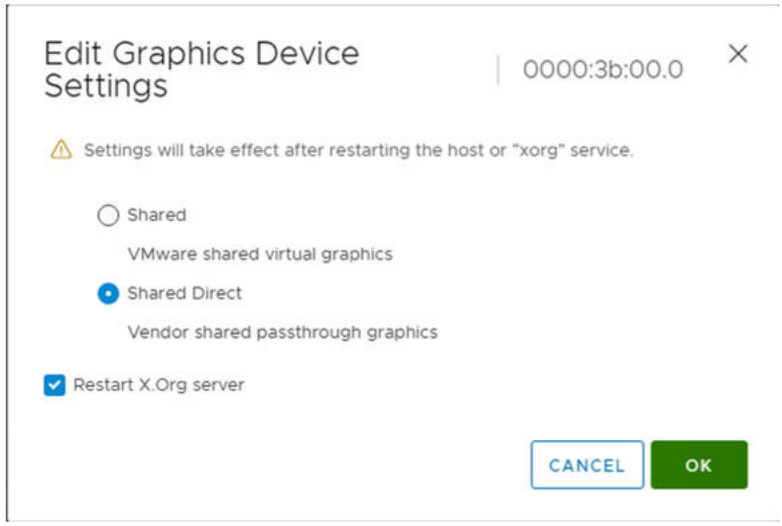
8 **확인**을 클릭하여 구성을 저장합니다.

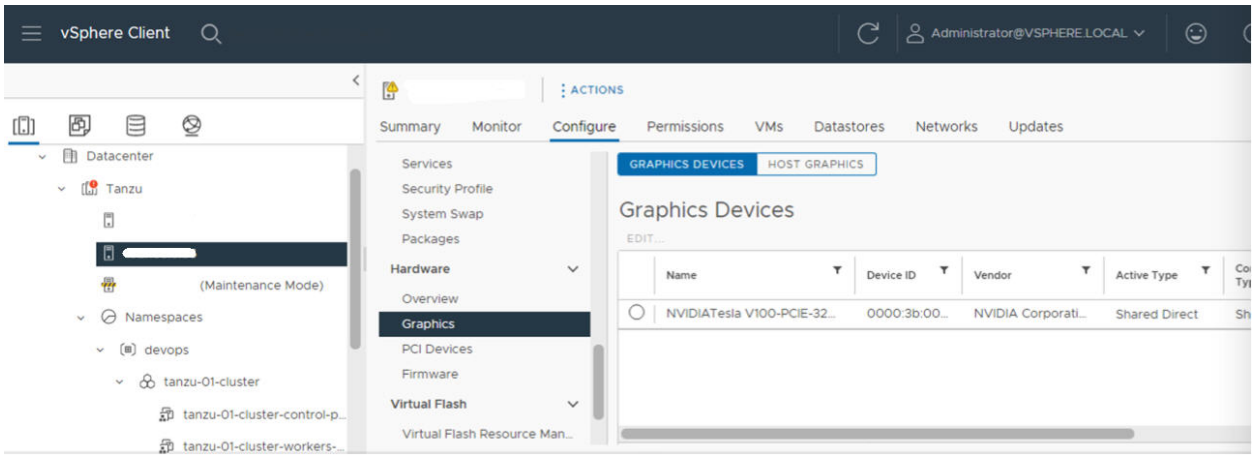
9 ESXi 호스트를 마우스 오른쪽 버튼으로 클릭하고 유지 보수 모드로 전환합니다.

10 호스트를 재부팅합니다.

11 호스트가 다시 실행되면 호스트를 유지 보수 모드에서 해제합니다.

12 **워크로드 관리**가 사용되도록 설정될 vCenter 클러스터의 각 ESXi 호스트에 대해 이 프로세스를 반복합니다.





NVIDIA GPU A30 및 A100 디바이스용 SR-IOV BIOS 켜기

다중 인스턴스 GPU(MIG 모드)에 필요한 NVIDIA A30 또는 A100 GPU 디바이스를 사용 중인 경우 ESXi 호스트에서 SR-IOV를 사용하도록 설정해야 합니다. SR-IOV를 사용하도록 설정하지 않으면 Tanzu Kubernetes 클러스터 노드 VM을 시작할 수 없습니다. 이 문제가 발생하면 **워크로드 관리**가 사용되도록 설정된 vCenter Server의 **최근 작업** 창에 다음과 같은 오류 메시지가 표시됩니다.

```
Could not initialize plugin libnvidia-vgx.so for vGPU nvidia_aXXX-xx. Failed to start the virtual machine. Module DevicePowerOn power on failed.
```

SR-IOV를 사용하도록 설정하려면 웹 콘솔을 사용하여 ESXi 호스트에 로그인합니다. **관리 > 하드웨어**를 선택합니다. NVIDIA GPU 디바이스를 선택하고 **SR-IOV 구성**을 클릭합니다. 여기에서 SR-IOV를 켤 수 있습니다. 추가 지침은 vSphere 설명서에서 SR-IOV(Single Root I/O Virtualization)를 참조하십시오.

참고 NIC 패스스루가 포함된 vGPU를 사용 중인 경우 추가 ESXi 구성 단계에 대해 다음 항목을 참조하십시오. TKGS 클러스터에 AI/ML 워크로드를 배포하기 위한 vSphere 관리자 부록(vGPU 및 동적 DirectPath IO).

관리자 3단계: 각 ESXi 호스트에 NVIDIA 호스트 관리자 드라이버 설치

NVIDIA vGPU 그래픽 가속을 사용하여 Tanzu Kubernetes 클러스터 노드 VM을 실행하려면 **워크로드 관리**가 사용되도록 설정될 vCenter 클러스터를 구성하는 각 ESXi 호스트에 NVIDIA 호스트 관리자 드라이버를 설치합니다.

NVIDIA vGPU 호스트 관리자 드라이버 구성 요소는 VIB(vSphere 설치 번들)에 패키지됩니다. NVAIE VIB는 NVIDIA GRID 라이센싱 프로그램을 통해 조직에서 제공합니다. VMware는 NVAIE VIB를 제공하지 않으며 이에 대한 다운로드 서비스도 제공하지 않습니다. 조직에서는 NVIDIA 라이센싱 프로그램의 일부로 라이센싱 서버를 설정합니다. 자세한 내용은 NVIDIA 가상 GPU 소프트웨어 빠른 시작 가이드를 참조하십시오.

NVIDIA 환경이 설정되면 각 ESXi 호스트에서 다음 명령을 실행하고 NVIDIA 라이센스 서버 주소와 NVAIE VIB 버전을 환경에 적합한 값으로 바꿉니다. 추가 지침은 VMware 지원 기술 자료에서 ESXi에서 NVIDIA VIB 설치 및 구성을 참조하십시오.

참고 ESXi 호스트에 설치된 NVAIE VIB 버전은 노드 VM에 설치된 vGPU 소프트웨어 버전과 일치해야 합니다. 아래 버전은 예시일 뿐입니다.

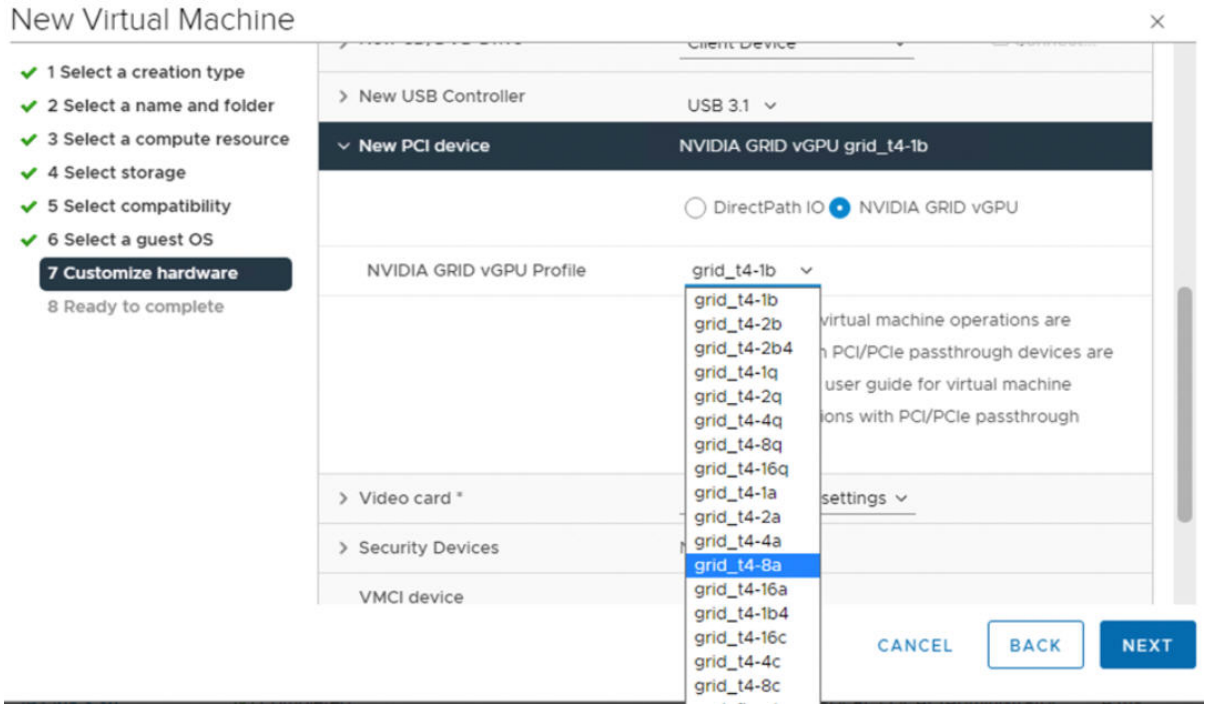
```
esxcli system maintenanceMode set --enable true
esxcli software vib install -v ftp://server.domain.example.com/nvidia/signed/
NVIDIA_bootbank_NVIDIA-VMware_ESXi_7.0_Host_Driver_460.73.02-1OEM.700.0.0.15525992.vib
esxcli system maintenanceMode set --enable false
/etc/init.d/xorg restart
```

관리자 4단계: ESXi 호스트가 NVIDIA vGPU 작업을 수행할 준비가 되었는지 확인

각 ESXi 호스트가 NVIDIA vGPU 작업을 수행할 준비가 되었는지 확인하려면 **워크로드 관리**가 사용되도록 설정될 vCenter 클러스터의 각 ESXi 호스트에 대해 다음 검사를 수행합니다.

- SSH를 사용하여 ESXi 호스트에 연결하고 셸 모드로 전환한 후 명령 `nvidia-smi`를 실행합니다. NVIDIA 시스템 관리 인터페이스는 NVIDA vGPU 호스트 관리자가 제공하는 명령줄 유틸리티입니다. 이 명령을 실행하면 호스트에서 GPU 및 드라이버가 반환됩니다.
- 다음 명령을 실행하여 NVIDIA 드라이버가 제대로 설치되었는지 확인합니다. `esxcli software vib list | grep NVIDA`.
- 호스트가 GPU Shared Direct로 구성되어 있고 SR-IOV가 켜져 있는지 확인합니다(NVIDIA A30 또는 A100 디바이스를 사용 중인 경우).

- GPU용으로 구성된 ESXi 호스트에서 vSphere Client를 사용하여 PCI 디바이스가 포함된 새 가상 시스템을 생성합니다. NVIDIA vGPU 프로파일이 표시되면서 선택 가능한 상태가 됩니다.



관리자 5단계: vGPU 구성 vCenter 클러스터에서 워크로드 관리 사용

이제 ESXi 호스트가 NVIDIA vGPU를 지원하도록 구성되었으므로 이러한 호스트로 구성된 vCenter 클러스터를 생성합니다. **워크로드 관리**를 지원하려면 vCenter 클러스터가 공유 스토리지, 고가용성, 완전히 자동화된 DRS를 비롯한 특정 요구 사항을 충족해야 합니다.

또한 **워크로드 관리**를 사용하도록 설정하려면 네이티브 vSphere vDS 네트워킹 또는 NSX-T Data Center 네트워킹 중에서 하나의 네트워킹 스택을 선택해야 합니다. vDS 네트워킹을 사용하는 경우 NSX Advanced 또는 HAProxy 로드 밸런서를 설치해야 합니다.

워크로드 관리를 사용하도록 설정하면 감독자 클러스터가 vGPU 지원 ESXi 호스트에서 실행됩니다. **워크로드 관리**를 사용하도록 설정하려면 다음 작업 및 설명서를 참조하십시오.

참고 **워크로드 관리**가 사용되도록 설정된 vCenter 클러스터가 이미 있는 경우 클러스터가 vGPU용으로 구성한 ESXi 호스트를 사용하고 있다고 가정하고 이 단계를 건너뜁니다.

작업	지침
워크로드 관리를 사용하도록 설정하기 위한 요구 사항을 충족하는 vCenter 클러스터 생성	vSphere 클러스터에서 vSphere with Tanzu를 구성하기 위한 사전 요구 사항
감독자 클러스터, NSX-T 또는 로드 밸런서가 포함된 vDS에 대한 네트워킹 구성	vSphere with Tanzu에 대한 NSX-T Data Center 구성. vSphere with Tanzu에 대한 vSphere 네트워킹 및 NSX Advanced Load Balancer 구성. vSphere with Tanzu에 대한 vSphere 네트워킹 및 HAProxy 로드 밸런서 구성.
워크로드 관리 사용	NSX-T Data Center 네트워킹으로 워크로드 관리 사용. vSphere 네트워킹으로 워크로드 관리 사용.

관리자 6단계: Tanzu Kubernetes Ubuntu 릴리스로 컨텐츠 라이브러리 생성 또는 업데이트

GPU 구성 vCenter 클러스터에서 **워크로드 관리**가 사용되도록 설정되었다면 다음 단계는 Tanzu Kubernetes 릴리스 OVA 이미지에 대한 컨텐츠 라이브러리를 생성하는 것입니다.

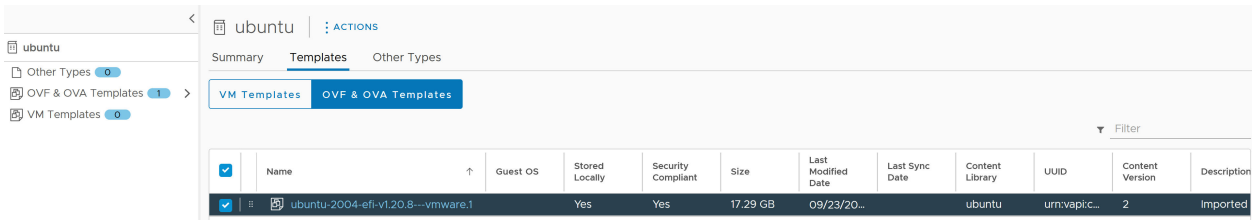
경고 Photon 이미지로 구성된 Tanzu Kubernetes 릴리스에 컨텐츠 라이브러리가 이미 있는 경우 기존 컨텐츠 라이브러리를 필요한 Ubuntu 이미지와 동기화하기만 하면 됩니다. TKGS 클러스터에 대한 두 번째 컨텐츠 라이브러리를 생성하지 마십시오. 그러면 시스템이 불안정해질 수 있습니다.

NVIDIA vGPU에는 Ubuntu 운영 체제가 필요합니다. VMware 이러한 목적으로 Ubuntu OVA를 제공합니다. vGPU 클러스터에는 PhotonOS Tanzu Kubernetes 릴리스를 사용할 수 없습니다.

이 이미지를 vSphere with Tanzu 환경으로 가져오려면 표에 나열된 방법 중 하나를 선택하고 해당 지침을 따릅니다.

컨텐츠 라이브러리 유형	설명
구독 컨텐츠 라이브러리를 생성하고 Ubuntu OVA를 환경과 자동으로 동기화합니다.	Tanzu Kubernetes 릴리스용 구독 컨텐츠 라이브러리 생성, 보안 및 동기화
로컬 컨텐츠 라이브러리를 생성하고 Ubuntu OVA를 환경에 수동으로 업로드합니다.	Tanzu Kubernetes 릴리스용 로컬 컨텐츠 라이브러리 생성, 보안 및 동기화

이 작업을 완료하면 Ubuntu OVA가 컨텐츠 라이브러리에 표시됩니다.



관리자 7단계: vGPU 프로파일을 사용하여 사용자 지정 VM 클래스 생성

다음 단계에서는 vGPU 프로파일을 사용하여 사용자 지정 VM 클래스를 생성합니다. 시스템에서는 Tanzu Kubernetes 클러스터 노드를 생성할 때 이 클래스 정의를 사용합니다.

아래 지침에 따라 vGPU 프로파일을 사용하여 사용자 지정 VM 클래스를 생성합니다. 추가 지침은 vSphere with Tanzu의 VM 클래스에 PCI 디바이스 추가 항목을 참조하십시오.

참고 NIC 패스스루가 포함된 vGPU를 사용 중인 경우 추가 단계에 대해 다음 항목을 참조하십시오. TKGS 클러스터에 AI/ML 워크로드를 배포하기 위한 vSphere 관리자 부록(vGPU 및 동적 DirectPath IO).

1 vSphere Client를 사용하여 vCenter Server에 로그온합니다.

2 **워크로드 관리**를 선택합니다.

3 **서비스**를 선택합니다.

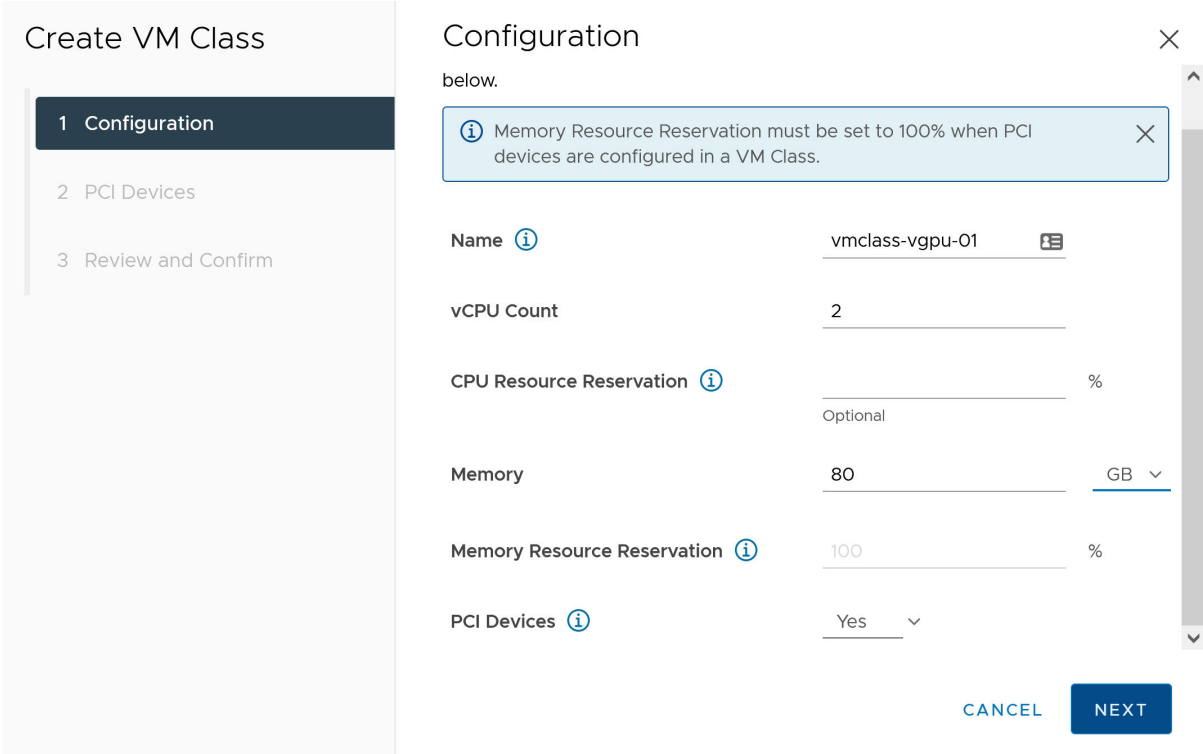
4 **VM 클래스**를 선택합니다.

5 **VM 클래스 생성**을 클릭합니다.

6 **구성** 탭에서 사용자 지정 VM 클래스를 구성합니다.

구성 필드	설명
이름	사용자 지정 VM 클래스에 대한 자체 설명 이름을 입력합니다(예: **vmclass-vgpu-1**).
vCPU 수	**2**
CPU 리소스 예약	선택 사항, 비워 둘 수 있음
메모리	예: **80GB**
메모리 리소스 예약	**100%**(PCI 디바이스가 VM 클래스에 구성된 경우 필수)
PCI 디바이스	**예** **참고** PCI 디바이스에 대해 [예]를 선택하면 GPU 디바이스를 사용 중임을 시스템에 알리고 vGPU 구성을 지원하도록 VM 클래스 구성이 변경됩니다.

예:



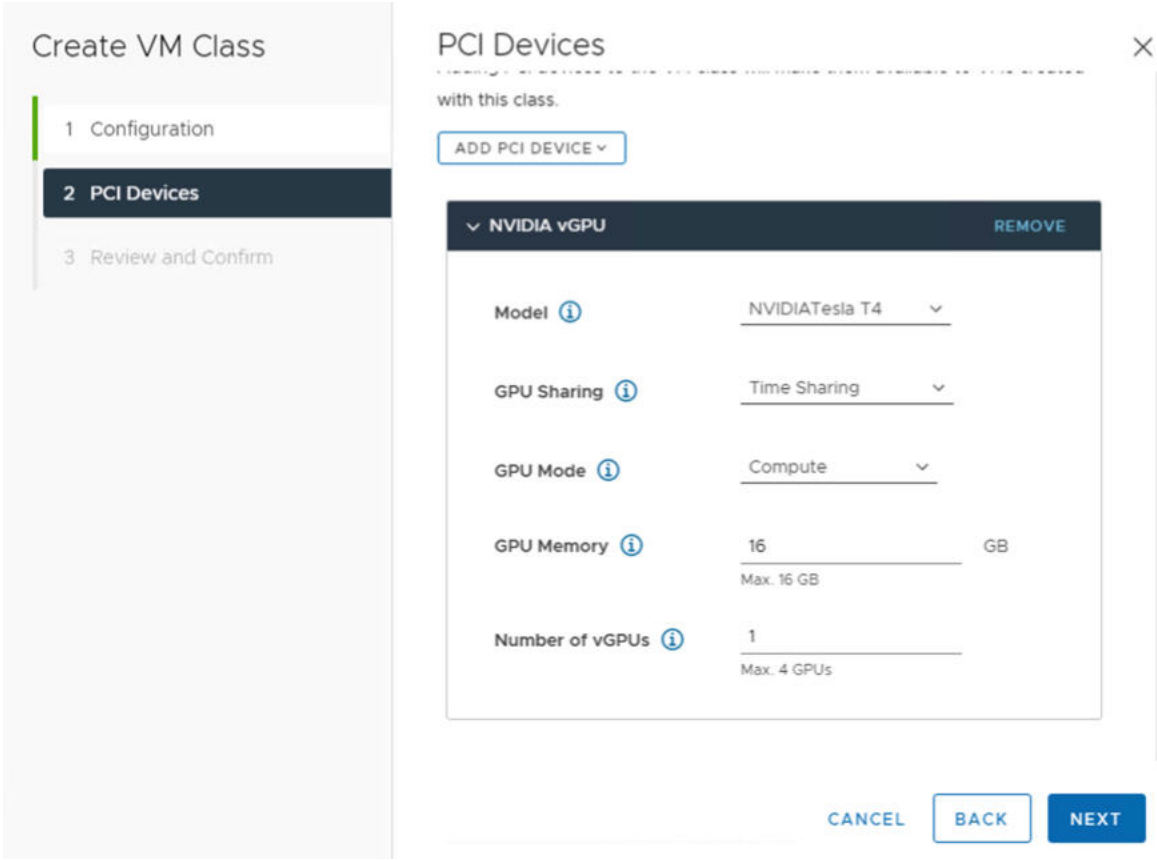
7 **다음**을 클릭합니다.

8 **PCI 디바이스** 탭에서 **PCI 디바이스 추가 > NVIDIA vGPU** 옵션을 선택합니다.

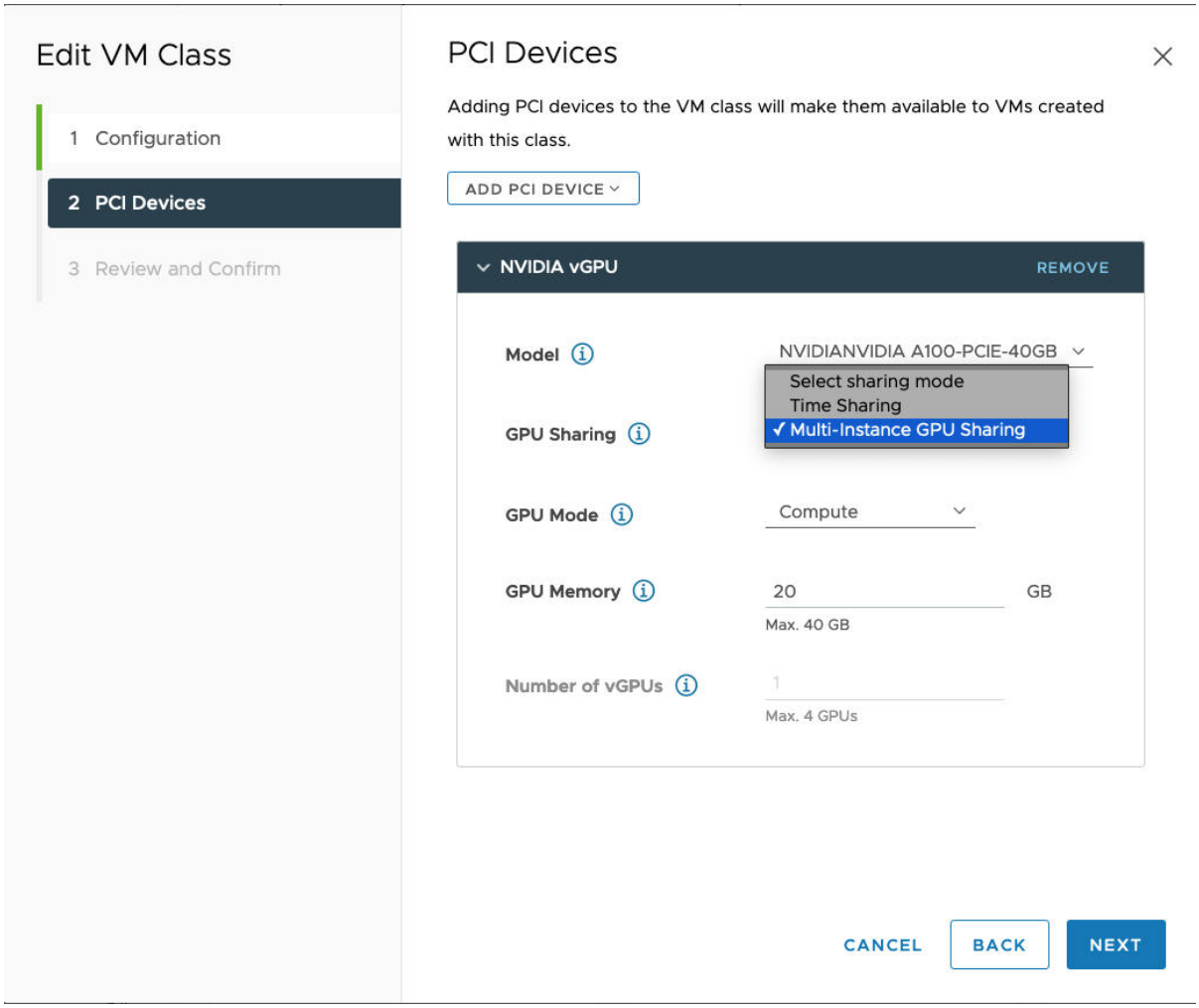
9 NVIDIA vGPU 모델을 구성합니다.

NVIDIA vGPU 필드	설명
모델	**NVIDIA vGPU > 모델** 메뉴에서 사용 가능한 NVIDIA GPU 하드웨어 디바이스 모델을 선택합니다. 시스템에 프로파일이 표시되지 않으면 클러스터의 어떤 호스트에도 지원되는 PCI 디바이스가 없는 것입니다.
GPU 공유	이 설정은 GPU 디바이스가 GPU 지원 VM 간에 공유되는 방식을 정의합니다. vGPU 구현에는 **시간 공유**와 **다중 인스턴스 GPU 공유**의 두 가지 유형이 있습니다. 시간 공유 모드에서 vGPU 스케줄러는 vGPU 간에 성능을 밸런싱하는 최선의 목표로 일정 기간 동안 각 vGPU 지원 VM에 대해 직렬로 작업을 수행하도록 GPU에 지시합니다. MIG 모드에서는 여러 vGPU 지원 VM을 단일 GPU 디바이스에서 병렬로 실행할 수 있습니다. MIG 모드는 최신 GPU 아키텍처를 기반으로 하며 NVIDIA A100 및 A30 디바이스에서만 지원됩니다. MIG 옵션이 표시되지 않으면 선택한 PCI 디바이스가 MIG 옵션을 지원하지 않는 것입니다.
GPU 모드	**계산**
GPU 메모리	예: **8GB**
vGPU 수	예: **1**

예를 들어 시간 공유 모드로 구성된 NVIDIA vGPU 프로파일은 다음과 같습니다.



예를 들어 지원되는 GPU 디바이스에서 MIG 모드로 구성된 NVIDIA vGPU 프로파일은 다음과 같습니다.



10 **다음**을 클릭합니다.

11 선택 항목을 검토하고 확인합니다.

12 **마침**을 클릭합니다.

13 새 사용자 지정 VM 클래스를 VM 클래스 목록에서 사용할 수 있는지 확인합니다.

관리자 8단계: TKGS GPU 클러스터에 대한 vSphere 네임스페이스 생성 및 구성

프로비저닝할 각 TKGS GPU 클러스터에 대해 vSphere 네임스페이스를 생성합니다. 편집 권한이 있는 vSphere SSO 사용자를 추가하여 네임스페이스를 구성하고 영구 볼륨에 대한 스토리지 정책을 연결합니다.

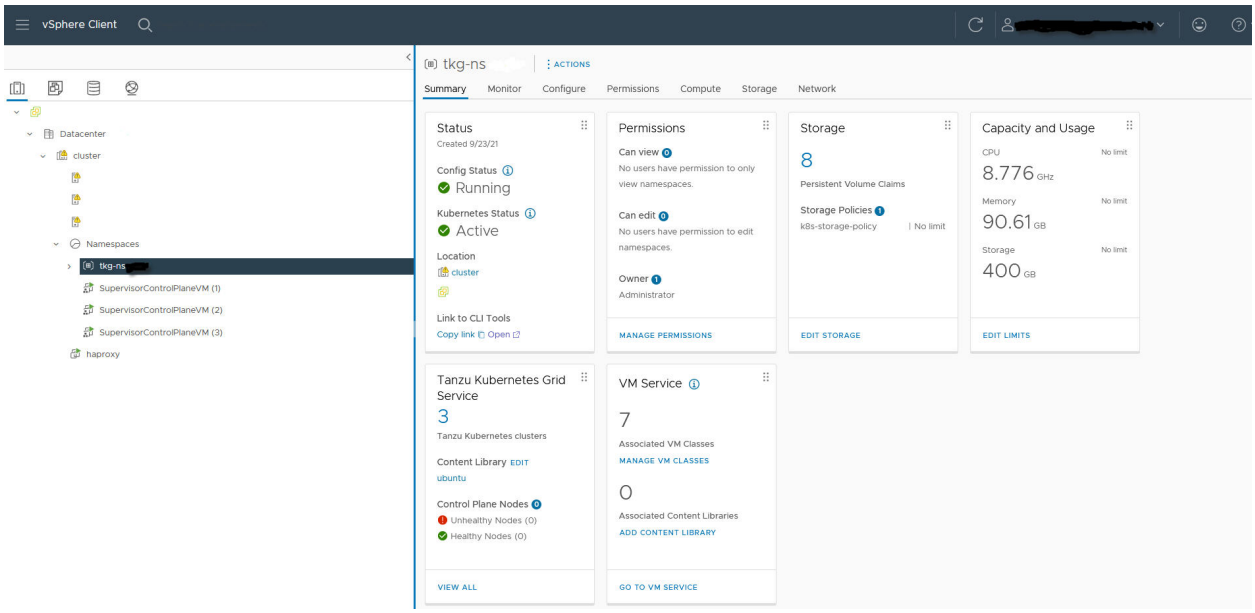
이 작업을 수행하려면 vSphere 네임스페이스 생성 및 구성 항목을 참조하십시오.

관리자 9단계: 컨텐츠 라이브러리 및 VM 클래스를 vSphere 네임스페이스와 연결

vSphere 네임스페이스를 생성하고 구성한 후 Ubuntu OVA를 포함하는 컨텐츠 라이브러리를 vSphere 네임스페이스와 연결하고 vGPU 프로파일을 포함하는 사용자 지정 VM 클래스를 동일한 vSphere 네임스페이스와 연결합니다.

작업	설명
vGPU용 Ubuntu OVA를 포함하는 컨텐츠 라이브러리를 TKGS 클러스터를 프로비저닝할 vSphere 네임스페이스와 연결합니다.	Tanzu Kubernetes 릴리스에 대한 vSphere 네임스페이스 구성의 내용을 참조하십시오.
vGPU 프로파일을 포함하는 사용자 지정 VM 클래스를 TKGS 클러스터를 프로비저닝할 vSphere 네임스페이스와 연결합니다.	vSphere with Tanzu에서 VM 클래스를 네임스페이스와 연결의 내용을 참조하십시오.

다음 예에서는 vGPU 클러스터에 사용할 연결된 컨텐츠 라이브러리 및 사용자 지정 VM 클래스가 있는 구성된 vSphere 네임스페이스를 보여 줍니다.



관리자 10단계: 감독자 클러스터에 액세스할 수 있는지 확인

마지막 관리 작업은 감독자 클러스터가 프로비저닝되었는지 그리고 클러스터 운영자가 AI/ML 워크로드에 대한 TKGS 클러스터를 프로비저닝하는 데 이를 사용할 수 있는지 확인하는 것입니다.

1 vSphere에 대한 Kubernetes CLI 도구를 다운로드하고 설치합니다.

vSphere에 대한 Kubernetes CLI 도구 다운로드 및 설치의 내용을 참조하십시오.

2 감독자 클러스터에 연결합니다.

vCenter Single Sign-On 사용자로 감독자 클러스터에 연결의 내용을 참조하십시오.

3 클러스터 운영자에게 vSphere에 대한 Kubernetes CLI 도구를 다운로드할 수 있는 링크와 vSphere 네임스페이스의 이름을 제공합니다.

TKGS 클러스터에 AI/ML 워크로드를 배포하기 위한 클러스터 운영자 워크플로의 내용을 참조하십시오.

TKGS 클러스터에 AI/ML 워크로드를 배포하기 위한 클러스터 운영자 워크플로

개발자가 TKGS 클러스터에 AI/ML 워크로드를 배포할 수 있도록 하려면 클러스터 운영자가 NVIDIA vGPU 작업을 지원하도록 Kubernetes 환경을 구성해야 합니다.

TKGS 클러스터에 AI/ML 워크로드를 배포하기 위한 클러스터 운영자 워크플로

TKGS 클러스터에 AI/ML 워크로드를 배포하는 개략적인 단계는 다음과 같습니다.

단계	작업	연결
0	시스템 요구 사항을 검토합니다.	운영자 0단계: 시스템 요구 사항 검토의 내용을 참조하십시오.
1	kubectl 및 kubectl용 vSphere 플러그인을 로컬 워크스테이션에 다운로드합니다.	운영자 1단계: 워크스테이션에 vSphere에 대한 Kubernetes CLI 도구 설치의 내용을 참조하십시오.
2	kubectl을 사용하여 감독자 클러스터에 로그인합니다. 그러면 .kube/config가 새 감독자 클러스터의 컨텍스트로 채워집니다.	운영자 2단계: 감독자 클러스터에 로그인의 내용을 참조하십시오.
3	kubectl을 사용하여 컨텍스트를 vSphere 네임스페이스로 전환합니다.	운영자 3단계: vSphere 네임스페이스로 컨텍스트 전환의 내용을 참조하십시오.
4	kubectl을 사용하여 VM 클래스를 나열하고 NVIDIA vGPU 지원 클래스가 있는지 확인합니다.	운영자 4단계: vGPU 워크로드에 대한 사용자 지정 VM 클래스 가져오기의 내용을 참조하십시오.
5	kubectl을 사용하여 사용 가능한 Tanzu Kubernetes 릴리스를 나열하고 Ubuntu 이미지가 있는지 확인합니다.	운영자 5단계: GPU 노드용 Ubuntu Tanzu Kubernetes 릴리스 가져오기의 내용을 참조하십시오.
6	GPU 지원 TKGS 클러스터를 프로비저닝하기 위한 YAML 규격을 만듭니다. TKR 버전 및 VM 클래스를 지정합니다.	운영자 6단계: vGPU 지원 TKGS 클러스터 프로비저닝을 위한 YAML 만들기의 내용을 참조하십시오.
7	TKGS 클러스터를 프로비저닝합니다.	운영자 7단계: TKGS 클러스터 프로비저닝의 내용을 참조하십시오.
8	클러스터에 로그인하고 프로비저닝을 확인합니다.	운영자 8단계: TKGS 클러스터에 로그인하고 프로비저닝 확인의 내용을 참조하십시오.
9	네임스페이스, 역할 바인딩, 이미지 암호, 라이센스 configmap을 비롯한 몇 가지 사전 요구 사항 개체를 TKGS 클러스터에 생성하여 NVAIE GPU Operator 설치를 준비합니다.	운영자 9단계: NVAIE GPU Operator 설치 준비의 내용을 참조하십시오.
10	클러스터에 NVAIE GPU Operator를 설치합니다.	운영자 10단계: 클러스터에 NVIDIA GPU Operator 설치의 내용을 참조하십시오.
11	AI/ML 워크로드를 vGPU 지원 TKGS 클러스터에 배포합니다.	운영자 11단계: AI/ML 워크로드 배포의 내용을 참조하십시오.

운영자 0단계: 시스템 요구 사항 검토

TKGS 클러스터에 AI/ML 워크로드를 배포하기 위한 환경을 설정하려면 다음 시스템 요구 사항을 참조하십시오.

요구 사항	설명
vSphere 관리자가 NVIDIA vGPU에 대한 환경을 설정함	TKGS 클러스터(vGPU)에 AI/ML 워크로드를 배포하기 위한 vSphere 관리자 워크플로 항목을 참조하십시오.
TKR Ubuntu OVA	Tanzu Kubernetes 릴리스 Ubuntu `ob-18691651-tkgs-ova-ubuntu-2004-v1.20.8---vmware.1-tkg.2`
TKG 클러스터 프로비저너	Tanzu Kubernetes Grid 서비스 API 버전: `run.tanzu.vmware.com/v1alpha2`
NVIDIA GPU Operator	GPU Operator v1.8.0
NVIDIA GPU 드라이버 컨테이너	`nvcr.io/nvstating/cnt-ea/driver:470.51-ubuntu20.04`

운영자 1단계: 워크스테이션에 vSphere에 대한 Kubernetes CLI 도구 설치

vSphere에 대한 Kubernetes CLI 도구를 다운로드하고 설치합니다.

Linux를 사용하는 경우 다음 명령을 실행하여 도구를 다운로드할 수 있습니다.

```
curl -LOk https://${SC_IP}/wcp/plugin/linux-amd64/vsphere-plugin.zip
unzip vsphere-plugin.zip
mv -v bin/* /usr/local/bin/
```

추가 지침은 vSphere에 대한 Kubernetes CLI 도구 다운로드 및 설치 항목을 참조하십시오.

운영자 2단계: 감독자 클러스터에 로그인

kubectl용 vSphere 플러그인을 사용하여 감독자 클러스터를 인증합니다.

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

운영자 3단계: vSphere 네임스페이스로 컨텍스트 전환

kubectl을 사용하여 vSphere 관리자가 TKGS GPU 클러스터에 대해 생성한 vSphere 네임스페이스로 컨텍스트를 전환합니다.

```
kubectl config get-contexts
```

```
kubectl config use-context TKGS-GPU-CLUSTER-NAMESPACE
```

운영자 4단계: vGPU 워크로드에 대한 사용자 지정 VM 클래스 가져오기

vSphere 관리자가 생성한 vGPU 프로파일과 함께 사용자 지정 VM 클래스를 대상 vSphere 네임스페이스에서 사용할 수 있는지 확인합니다.

```
kubectl get virtualmachineclassbindings
```

참고 VM 클래스는 대상 vSphere 네임스페이스에 바인딩되어야 합니다. vGPU 워크로드에 대한 사용자 지정 VM 클래스가 표시되지 않으면 vSphere 관리자에게 확인하십시오.

운영자 5단계: GPU 노드용 Ubuntu Tanzu Kubernetes 릴리스 가져오기

vSphere 관리자가 컨텐츠 라이브러리에서 동기화한 필수 Ubuntu Tanzu Kubernetes 릴리스를 vSphere 네임스페이스에서 사용할 수 있는지 확인합니다.

```
kubectl get tanzukubernetesreleases
```

또는 바로 가기를 사용합니다.

```
kubectl get tkr
```

운영자 6단계: vGPU 지원 TKGS 클러스터 프로비저닝을 위한 YAML 만들기

Tanzu Kubernetes 클러스터 프로비저닝을 위한 YAML 파일을 생성합니다.

아래 예제 중 하나로 시작합니다. 이전 명령의 출력에서 수집한 정보를 사용하여 클러스터 규격을 사용자 지정합니다. 구성 매개 변수의 전체 목록을 참조하십시오. Tanzu Kubernetes 클러스터 프로비저닝을 위한 TKGS v1alpha2 API.

예제 1은 두 개의 작업자 노드 풀을 지정합니다.

```
apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TanzuKubernetesCluster
metadata:
   #cluster name
   name: tkgs-cluster-gpu-a100
   #target vsphere namespace
   namespace: tkgs-gpu-operator
spec:
   topology:
     controlPlane:
       replicas: 3
       #storage class for control plane nodes
       #use `kubectl describe storageclasses`
       #to get available pvcs
       storageClass: vwt-storage-policy
       vmClass: guaranteed-medium
       #TKR NAME for Ubuntu ova supporting GPU
       tkr:
         reference:
           name: 1.20.8---vmware.1-tkg.1
     nodePools:
     - name: nodepool-a100-primary
       replicas: 3
       storageClass: vwt-storage-policy
       #custom VM class for vGPU
       vmClass: class-vgpu-a100
       #TKR NAME for Ubuntu ova supporting GPU
       tkr:
         reference:
           name: 1.20.8---vmware.1-tkg.1
     - name: nodepool-a100-secondary
       replicas: 3
       vmClass: class-vgpu-a100
```

```
        storageClass: vwt-storage-policy
        #TKR NAME for Ubuntu ova supporting GPU
        tkr:
          reference:
            name: 1.20.8---vmware.1-tkg.1
    settings:
      storage:
        defaultClass: vwt-storage-policy
      network:
        cni:
         name: antrea
        services:
         cidrBlocks: ["198.51.100.0/12"]
        pods:
         cidrBlocks: ["192.0.2.0/16"]
        serviceDomain: managedcluster.local
```

예제 2는 용량이 50GiB인 컨테이너화된 런타임에 대해 작업자 노드에 별도 볼륨을 지정합니다. 이 설정은 구성이 가능합니다. 컨테이너 기반 AI/ML 워크로드에 대해 양호한 크기의 별도 볼륨을 제공하는 것이 좋습니다.

```
apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TanzuKubernetesCluster
metadata:
  name: tkc
  namespace: tkg-ns-auto
spec:
  distribution:
    fullVersion: v1.20.8+vmware.1-tkg.1
  topology:
    controlPlane:
      replicas: 3
      storageClass: vwt-storage-policy
      tkr:
        reference:
          name: v1.20.8---vmware.1-tkg.1
      vmClass: best-effort-medium
    nodePools:
    - name: workers
      replicas: 3
      storageClass: k8s-storage-policy
      tkr:
        reference:
          name: v1.20.8---vmware.1-tkg.1
      vmClass: vmclass-vgpu
      volumes:
      - capacity:
          storage: 50Gi
        mountPath: /var/lib/containerd
        name: containerd
      - capacity:
          storage: 50Gi
        mountPath: /var/lib/kubelet
        name: kubelet
```

```
    - name: nodepool-1
      replicas: 1
      storageClass: vwt-storage-policy
      vmClass: best-effort-medium
```

예제 3에는 레이블과 같은 클러스터 추가 메타데이터가 포함됩니다.

```
apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TanzuKubernetesCluster
metadata:
  annotations:
  labels:
    run.tanzu.vmware.com/tkr: v1.20.8---vmware.1-tkg.1
  name: tkgs-gpu-direct-rdma
  namespace: tkgs-ns
spec:
  settings:
    network:
      cni:
        name: antrea
      pods:
        cidrBlocks:
        - 192.168.0.0/16
      serviceDomain: cluster.local
      services:
        cidrBlocks:
        - 10.96.0.0/12
  topology:
    controlPlane:
      replicas: 3
      storageClass: tkgs-storage-policy
      vmClass: guaranteed-medium
      tkr:
        reference:
          name: v1.20.8---vmware.1-tkg.1
    nodePools:
    - name: workers
      replicas: 5
      storageClass: tkgs-storage-policy
      vmClass: claire-gpu-direct-rdma
      volumes:
      - capacity:
          storage: 50Gi
        mountPath: /var/lib/containerd
        name: containerd
      - capacity:
          storage: 50Gi
        mountPath: /var/lib/kubelet
        name: kubelet
      tkr:
        reference:
          name: v1.20.8---vmware.1-tkg.1
```

운영자 7단계: TKGS 클러스터 프로비저닝

다음 kubectl 명령을 실행하여 클러스터를 프로비저닝합니다.

```
kubectl apply -f CLUSTER-NAME.yaml
```

예:

```
kubectl apply -f tkgs-gpu-cluster-1.yaml
```

kubectl을 사용하여 클러스터 노드의 배포를 모니터링합니다.

```
kubectl get tanzukubernetesclusters -n NAMESPACE
```

운영자 8단계: TKGS 클러스터에 로그인하고 프로비저닝 확인

kubectl용 vSphere 플러그인을 사용하여 TKGS 클러스터에 로그인합니다.

```
kubectl vsphere login --server=IP-ADDRESS --vsphere-username USERNAME \
--tanzu-kubernetes-cluster-name CLUSTER-NAME --tanzu-kubernetes-cluster-namespace NAMESPACE-
NAME
```

다음 명령을 사용하여 클러스터를 확인합니다.

```
kubectl cluster-info
```

```
kubectl get nodes
```

```
kubectl get namespaces
```

```
kubectl api-resources
```

운영자 9단계: NVAIE GPU Operator 설치 준비

NVIDIA AI Enterprise를 사용하여 GPU Operator를 설치하기 전에 프로비저닝한 TKGS 클러스터에 대해 다음 작업을 완료합니다. 추가 지침은 NVAIE 설명서에서 사전 구성 작업을 참조하십시오.

참고 NVIDIA DLS(Delegated Licensing Server)를 사용하는 경우에는 TKGS 클러스터(DLS)에 AI/ML 워크로드를 배포하기 위한 클러스터 운영자 부록 항목에서 지침을 참조하십시오.

1 Kubernetes 네임스페이스 gpu-operator-resources를 생성합니다. 가장 좋은 방법은 항상 모든 항목을 이 네임스페이스에 배포하는 것입니다.

```
kubectl create ns gpu-operator-resources
```

2 역할 바인딩을 생성합니다.

Tanzu Kubernetes 클러스터에는 포드 보안 정책이 사용되도록 설정되어 있습니다.

rolebidings.yaml을 생성합니다.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: psp:vmware-system-privileged:default
  namespace: default
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: psp:vmware-system-privileged
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: system:nodes
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: system:serviceaccounts
```

역할 바인딩을 적용합니다.

```
kubectl apply -f rolebindings.yaml
```

post-rolebindings.yaml을 생성합니다.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: psp:vmware-system-privileged:gpu-operator-resources
  namespace: gpu-operator-resources
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: psp:vmware-system-privileged
subjects:
- kind: Group
  apiGroup: rbac.authorization.k8s.io
  name: system:serviceaccounts
```

역할 바인딩을 적용합니다.

```
kubectl apply -f post-rolebindings.yaml
```

3 Docker가 NVIDIA GPU Cloud 카탈로그에서 컨테이너 이미지를 끌어오는 데 사용할 수 있는 NGC 자격 증명을 사용하여 이미지 암호를 생성합니다.

```
kubectl create secret docker-registry registry-secret \
     --docker-server=server-name --docker-username='$oauthtoken' \
     --docker-password=<place_holder> \
     --docker-email=email-name -n gpu-operator-resources
```

4 NVIDIA 라이센스 서버에 대한 configmap을 생성합니다.

```
kubectl create configmap licensing-config -n gpu-operator-resources --from-file=gridd.conf
```

`gridd.conf`는 NVIDIA 라이센스 서버 주소를 참조합니다. 예:

```
# Description: Set License Server Address
# Data type: string
# Format:  "<address>"
ServerAddress=<place_holder>
```

운영자 10단계: 클러스터에 NVIDIA GPU Operator 설치

TKGS 클러스터에 NVAIE GPU Operator 버전 1.8.0을 설치합니다. 추가 지침은 GPU Operator 설명서를 참조하십시오.

참고 NVIDIA DLS(Delegated Licensing Server)를 사용하는 경우에는 TKGS 클러스터(DLS)에 AI/ML 워크로드를 배포하기 위한 클러스터 운영자 부록 항목에서 지침을 참조하십시오.

1 Helm 설명서를 참조하여 Helm을 설치합니다.

2 `gpu-operator` Helm 저장소를 추가합니다.

```
helm repo add nvidia https://nvidia.github.io/gpu-operator
```

3 다음 명령을 실행하여 NVAIE GPU Operator를 설치합니다.

필요한 경우 환경 변수 값을 환경과 일치하는 값으로 대체합니다.

```
export PRIVATE_REGISTRY="private/registry/path"
export OS_TAG=ubuntu20.04
export VERSION=460.73.01
export VGPU_DRIVER_VERSION=460.73.01-grid
export NGC_API_KEY=ZmJjMHZya...LWExNTRi
export REGISTRY_SECRET_NAME=registry-secret

helm install nvidia/gpu-operator \
   --set driver.repository=$PRIVATE_REGISTRY \
   --set driver.version=$VERSION \
   --set driver.imagePullSecrets={$REGISTRY_SECRET_NAME} \
   --set operator.defaultRuntime=containerd \
   --set driver.licensingConfig.configMapName=licensing-config
```

운영자 11단계: AI/ML 워크로드 배포

NVIDIA GPU Cloud 카탈로그는 vGPU 지원 Tanzu Kubernetes 클러스터에서 AI/ML 워크로드를 실행하는 데 사용할 수 있는 몇 가지 기본 제공 컨테이너 이미지를 제공합니다. 사용 가능한 이미지에 대한 자세한 내용은 NGC 설명서를 참조하십시오.

TKGS 클러스터에 AI/ML 워크로드를 배포하기 위한 vSphere 관리자 부록 (vGPU 및 동적 DirectPath IO)

vGPU 및 동적 DirectPath IO를 사용하여 AI/ML 워크로드를 지원하도록 TKGS를 구성하는 경우 이 델타 항목을 참조하십시오.

동적 DirectPath IO가 포함된 vGPU에 대한 vSphere 관리자 워크플로 조정

vGPU 및 동적 DirectPath IO를 사용하려면 동일한 TKGS 클러스터(vGPU)에 AI/ML 워크로드를 배포하기 위한 vSphere 관리자 워크플로를 수행하되 다음 변경 사항에 유의하십시오.

관리자 2단계: PCI 디바이스에 대한 패스스루 사용

vGPU 및 동적 DirectPath IO를 사용하려면 관리자 2단계: vGPU 작업을 위한 각 ESXi 호스트 구성에 설명된 대로 vGPU에 대해 각 ESXi 호스트를 구성합니다.

또한 GPU 디바이스를 다음과 같이 구성합니다.

1 vSphere Client를 사용하여 vCenter Server에 로그인합니다.

2 vCenter 클러스터에서 대상 ESXi 호스트를 선택합니다.

3 **구성 > 하드웨어 > PCI 디바이스**를 선택합니다.

4 **모든 PCI 디바이스** 탭을 선택합니다.

5 대상 NVIDIA GPU 가속기 디바이스를 선택합니다.

6 **패스스루 전환**을 클릭합니다.

7 ESXi 호스트를 마우스 오른쪽 버튼으로 클릭하고 유지 보수 모드로 전환합니다.

8 호스트를 재부팅합니다.

9 호스트가 다시 실행되면 호스트를 유지 보수 모드에서 해제합니다.

관리자 7단계: vGPU 및 동적 DirectPath IO를 사용하여 사용자 지정 VM 클래스 생성

vGPU 및 동적 DirectPath IO를 사용하려면 먼저 관리자 7단계: vGPU 프로파일을 사용하여 사용자 지정 VM 클래스 생성에 설명된 대로 **NVIDIA vGPU** 프로파일을 사용하여 사용자 지정 VM 클래스를 구성합니다.

그런 다음 이 VM 클래스에 **동적 DirectPath IO**가 지정되고 지원되는 PCI 디바이스가 선택된 두 번째 PCI 디바이스 구성을 추가합니다. 이 유형의 VM 클래스가 인스턴스화되면 vSphere DRS(Distributed Resource Scheduler)에 따라 VM 배치가 결정됩니다.

vGPU 및 동적 DirectPath IO를를 지원하는 사용자 지정 VM 클래스를 생성하려면 다음 지침을 참조하십시오. 추가 지침은 vSphere with Tanzu의 VM 클래스에 PCI 디바이스 추가 항목을 참조하십시오.

1 vSphere Client를 사용하여 vCenter Server에 로그온합니다.

2 **워크로드 관리**를 선택합니다.

3 **서비스**를 선택합니다.

4 **VM 클래스**를 선택합니다.

5 **NVIDIA vGPU** 프로파일로 이미 구성된 사용자 지정 VM 클래스를 편집합니다.

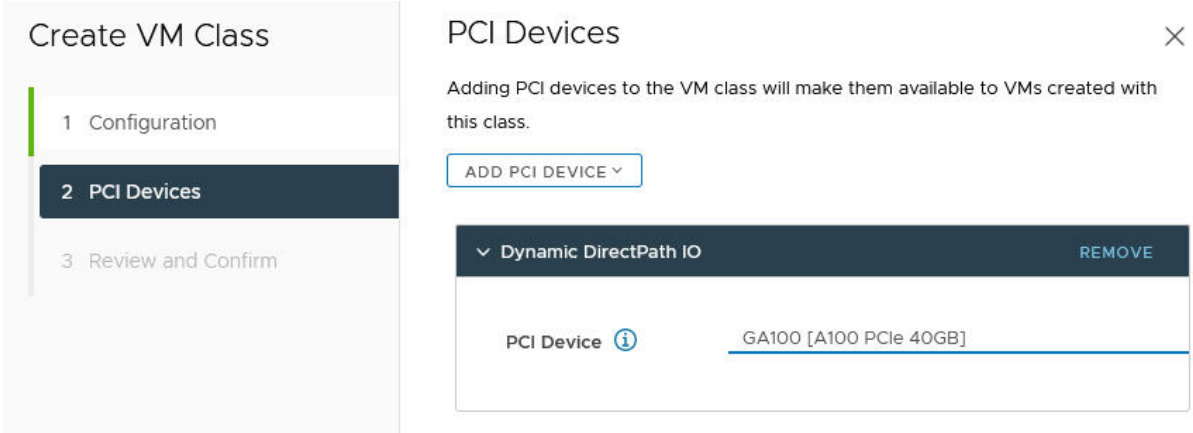
6 **PCI 디바이스** 탭을 선택합니다.

7 **PCI 디바이스 추가**를 클릭합니다.

8 **동적 DirectPath IO** 옵션을 선택합니다.

9 **PCI 디바이스**를 선택합니다.

예:



10 **다음**을 클릭합니다.

11 선택 항목을 검토하고 확인합니다.

12 **마침**을 클릭합니다.

13 새 사용자 지정 VM 클래스를 VM 클래스 목록에서 사용할 수 있는지 확인합니다.

TKGS 클러스터(DLS)에 AI/ML 워크로드를 배포하기 위한 클러스터 운영자 부록

NVIDIA AI Enterprise 계정에 대해 NVIDIA DLS(Delegated Licensing Server)를 사용하는 경우 이 델타 항목을 참조하십시오.

TKGS 클러스터에 AI/ML 워크로드를 배포하기 위한 클러스터 운영자 부록

NVIDIA는 Delegated Licensing Server를 의미하는 DLS라는 새로운 NLS(NVIDIA Licensing Server) 시스템을 제공합니다. 자세한 내용은 NVIDIA 설명서를 참조하십시오.

NVAIE 계정에 대해 DLS를 사용하는 경우 NVAIE GPU Operator 준비 및 배포 단계는 TKGS 클러스터에 AI/ML 워크로드를 배포하기 위한 클러스터 운영자 워크플로에 설명된 것과는 다릅니다. 특히 9단계와 10단계는 다음과 같이 수정됩니다.

운영자 9단계: NVAIE GPU Operator 설치 준비

DLS를 사용하여 GPU Operator 설치를 준비하려면 다음 단계를 완료하십시오.

1 암호를 생성합니다.

```
kubectl create secret docker-registry registry-secret \
  --docker-server=<users private NGC registry name>
  --docker-username='$oauthtoken' \
  --docker-password=ZmJj·····Ri \
  --docker-email=<user-email-address> -n gpu-operator-resources
```

참고 암호는 이전에 NGC(NVIDIA GPU Cloud) 포털에서 생성된 사용자 API 키입니다.

2 DLS 서버에서 클라이언트 토큰을 가져옵니다.

vGPU 라이센스를 사용하려는 사용자는 해당 DLS 라이센스 서버에서 '클라이언트 토큰'이라고 하는 토큰을 가져와야 합니다. 이 작업을 수행하기 위한 메커니즘은 NVIDIA 설명서에 나와 있습니다.

3 클라이언트 토큰을 사용하여 TKGS 클러스터에서 ConfigMap 개체를 생성합니다.

클라이언트 토큰 파일을 <path>/client_configuration_token.tok에 배치합니다.

그런 후 다음 명령을 실행합니다.

```
kubectl delete configmap licensing-config -n gpu-operator-resources; > gridd.conf
kubectl create configmap licensing-config \
  -n gpu-operator-resources --from-file=./gridd.conf --from-file=./
client_configuration_token.tok
```

참고 DLS에서 사용하는 grid.conf 파일은 비어 있습니다. 그러나 두 "--from-file" 매개 변수는 모두 필요합니다.

운영자 10단계: NVAIE GPU Operator 설치

DLS를 사용하여 NVAIE GPU Operator를 설치하려면 다음 단계를 완료하십시오. 추가 지침은 GPU Operator 설명서를 참조하십시오.

1 TKGS 클러스터에 NVAIE GPU Operator를 설치합니다.

- Helm 설명서를 참조하여 Helm을 설치합니다.

- gpu-operator Helm 저장소를 추가합니다.

```
helm repo add nvidia https://nvidia.github.io/gpu-operator
```

- Helm을 사용하여 GPU Operator를 설치합니다.

```
export PRIVATE_REGISTRY="<user' s private registry name>"
export OS_TAG=ubuntu20.04
export VERSION=470.63.01
export VGPU_DRIVER_VERSION=470.63.01-grid
export NGC_API_KEY=Zm……Ri  <- The user' s NGC AP Key
export REGISTRY_SECRET_NAME=registry-secret

helm show chart .
kubectl delete crd clusterpolicies.nvidia.com
helm install gpu-operator . -n gpu-operator-resources \
  --set psp.enabled=true \
  --set driver.licensingConfig.configMapName=licensing-config \
  --set operator.defaultRuntime=containerd \
  --set driver.imagePullSecrets={$REGISTRY_SECRET_NAME} \
  --set driver.version=$VERSION \
  --set driver.repository=$PRIVATE_REGISTRY \
  --set driver.licensingConfig.nlsEnabled=true
```

2 DLS가 작동했는지 확인합니다.

GPU Operator가 배포한 NVIDIA 드라이버 DaemonSet 포드 내에서 nvidia-smi 명령을 실행하여 DLS가 작동 중인지 확인합니다.

먼저 다음 명령을 실행하여 포드로 이동하고 셸 세션을 불러옵니다.

```
kubectl exec -it nvidia-driver-daemonset-cvxx6 nvidia-driver-ctr -n gpu-operator-resources
– bash
```

이제 명령을 실행하여 DLS 설정을 확인할 수 있습니다.

```
nvidia-smi
```

DLS가 올바르게 설정되었다면 이 명령을 실행했을 때 출력에 "라이센스가 부여됨"이 반환됩니다.

vSphere with Tanzu 워크로드에 컨테이너 레지스트리 사용

15

컨테이너 레지스트리는 Kubernetes 운영자에게 컨테이너 이미지를 저장하고 공유하기 편리한 저장소를 제공합니다. vSphere with Tanzu에는 감독자 클러스터에서 사용하도록 설정할 수 있는 내장된 Harbor 레지스트리가 포함되어 있습니다. 또한 Tanzu Kubernetes 클러스터에서 외부 개인 컨테이너 레지스트리를 사용할 수도 있습니다.

vSphere 포드 및 Tanzu Kubernetes 클러스터 워크로드 배포를 위한 개인 컨테이너 레지스트리로 작동하도록 감독자 클러스터에 내장된 Harbor 레지스트리를 사용하도록 설정할 수 있습니다. vSphere Docker 자격 증명 도우미를 사용하여 레지스트리에 안전하게 액세스하고 컨테이너 이미지를 끌어오거나 푸시할 수 있는 개발자에게 레지스트리 URL을 제공합니다.

참고 내장된 Harbor 레지스트리를 사용하려면 감독자 클러스터가 NSX-T 네트워킹을 사용하도록 구성되어 있어야 합니다.

내장된 Harbor 레지스트리 외에 또는 이것의 대안으로 외부 개인 컨테이너 레지스트리를 사용하도록 Tanzu Kubernetes 클러스터를 구성할 수 있습니다.

본 장은 다음 항목을 포함합니다.

- 감독자 클러스터에서 내장된 Harbor 레지스트리를 사용하도록 설정
- 내장된 Harbor 레지스트리 콘솔에 로그인
- 내장된 Harbor 레지스트리 인증서 다운로드 및 설치
- 내장된 Harbor 레지스트리 인증서를 사용하여 Docker 클라이언트 구성
- vSphere Docker 자격 증명 도우미를 설치하고 레지스트리에 연결
- 내장된 Harbor 레지스트리로 이미지 푸시
- 내장된 Harbor 레지스트리에서 이미지 지우기
- Tanzu Kubernetes 클러스터에서 내장된 Harbor 레지스트리 사용
- Tanzu Kubernetes 클러스터에서 외부 컨테이너 레지스트리 사용

감독자 클러스터에서 내장된 Harbor 레지스트리를 사용하도록 설정

vSphere 관리자는 vSphere with Tanzu에 내장된 Harbor 레지스트리를 사용하도록 설정할 수 있습니다. 레지스트리에서 컨테이너 이미지를 끌어오거나 푸시하고 이러한 이미지를 사용하여 컨테이너를 배포할 수도 있습니다.

Harbor 레지스트리를 사용하도록 설정되면 감독자 클러스터의 모든 네임스페이스에는 전용 이미지 레지스트리와 동일한 이름을 가진 일치하는 프로젝트가 있습니다. 네임스페이스에 대한 편집 또는 보기 사용 권한이 있는 모든 사용자 또는 그룹은 전용 이미지 레지스트리의 동일한 이름을 가진 일치하는 프로젝트에서 해당 역할을 가진 멤버가 됩니다. 전용 이미지 레지스트리의 프로젝트 및 멤버의 수명 주기는 자동으로 관리되며 네임스페이스의 수명 주기와 네임스페이스의 사용자 또는 그룹 사용 권한에 연결됩니다.

사전 요구 사항

내장된 Harbor 레지스트리를 사용하도록 설정하려면 **워크로드 관리**를 사용하도록 설정하고 감독자 클러스터를 배포해야 합니다. 또한, 컨테이너 이미지 배치를 위한 스토리지 정책을 생성합니다. 이 스토리지 정책은 레지스트리에서 컨테이너 이미지에 대한 백업 저장소로 사용할 영구 볼륨을 프로비저닝하는 데 사용됩니다.

참고 내장된 Harbor 레지스트리를 사용하려면 NSX-T Data Center가 있는 감독자 클러스터를 네트워킹 솔루션으로 배포해야 합니다. vSphere with Tanzu에 대한 NSX-T Data Center 구성의 내용을 참조하십시오.

절차

1 vSphere Client에서 **워크로드 관리**를 사용하도록 설정된 vCenter 클러스터로 이동합니다.

2 **구성**을 선택합니다.

3 **감독자 클러스터**를 선택합니다.

4 **이미지 레지스트리**를 선택합니다.

5 **Harbor 사용**을 클릭합니다.

6 컨테이너 이미지 배치를 위한 **스토리지 정책**을 선택합니다.

7 **확인**을 클릭하여 프로세스를 완료합니다.

결과

몇 분 후에 전용 이미지 레지스트리를 사용할 수 있게 됩니다. 전용 이미지 레지스트리의 해당 인스턴스에 대해 특수 네임스페이스가 생성됩니다. 해당 네임스페이스에 대한 작업을 수행할 수 없습니다. vSphere 사용자에 대해서는 읽기 전용입니다.

다음에 수행할 작업

내장된 Harbor 레지스트리 콘솔에 로그인.

내장된 Harbor 레지스트리 콘솔에 로그인

내장된 Harbor 레지스트리 관리자 콘솔을 사용하여 개인 레지스트리를 관리하고 운영합니다.

vSphere 관리자는 내장된 Harbor 레지스트리 관리자 콘솔을 사용하여 프로젝트를 생성 및 관리하고, 로그를 보고, Harbor API를 살펴볼 수 있습니다.

사전 요구 사항

감독자 클러스터에서 내장된 Harbor 레지스트리를 사용하도록 설정

절차

1 vSphere Client에서 **워크로드 관리**를 사용하도록 설정된 vCenter 클러스터로 이동합니다.

2 **구성**을 선택합니다.

3 **네임스페이스**에서 **이미지 레지스트리**를 선택합니다.

4 **Harbor UI에 연결**을 클릭합니다.

내장된 Harbor 레지스트리 콘솔 로그인 페이지가 나타납니다.

5 vSphere 관리자 자격 증명을 사용하여 로그인합니다.

내장된 Harbor 레지스트리 인증서 다운로드 및 설치

내장된 Harbor 레지스트리 루트 CA 인증서를 다운로드하여 클라이언트를 레지스트리에 연결하는 데 사용할 수 있습니다.

Docker 클라이언트를 사용하여 내장된 Harbor 레지스트리에 로그인하려면 해당 클라이언트에 루트 CA 인증서를 설치해야 합니다.

사전 요구 사항

이 작업은 클라이언트 호스트 시스템에 Docker를 설치하고 구성했다고 가정합니다. 또한 내장된 Harbor 레지스트리를 사용하도록 설정되어 있어야 합니다. 감독자 클러스터에서 내장된 Harbor 레지스트리를 사용하도록 설정의 내용을 참조하십시오.

절차

1 내장된 Harbor 레지스트리 인증서를 다운로드합니다. 이 작업을 수행하는 방법은 두 가지입니다.

- 내장된 Harbor 레지스트리 콘솔 인터페이스를 사용합니다.
  - URL을 사용하여 내장된 Harbor 레지스트리 콘솔에 로그인합니다. 내장된 Harbor 레지스트리 콘솔에 로그인의 내용을 참조하십시오.
  - **프로젝트 > 프로젝트 이름** 페이지에서 프로젝트 링크를 클릭합니다.
  - **저장소** 탭을 선택합니다.
  - **레지스트리 인증서**를 클릭합니다.

- ca.crt 인증서 파일을 로컬 시스템에 저장합니다.

- vSphere Client를 사용합니다.
  - **워크로드 관리** 및 내장된 Harbor 레지스트리가 사용되도록 설정된 vCenter 클러스터를 선택합니다.
  - **구성 > 네임스페이스 > 이미지 레지스트리**를 선택합니다.
  - **루트 인증서** 필드에서 **SSL 루트 인증서 다운로드** 링크를 클릭합니다.
  - root-certificate.txt 파일을 로컬 시스템에 저장합니다.
  - 파일의 이름을 ca.crt로 변경합니다.

2 다운로드한 내장된 Harbor 레지스트리 ca.crt 파일을 Docker가 설치된 호스트의 적절한 디렉토리에 복사합니다. 기본 인증서 위치는 Docker 클라이언트가 실행 중인 OS 유형에 따라 다릅니다.

- Linux

```
/etc/docker/certs.d/ca.crt
```

- Mac OS

```
security add-trusted-cert -d -r trustRoot -k ~/Library/Keychains/login.keychain ca.crt
```

참고 기본 위치에 ca.crt를 설치하지 않을 경우 vSphere Docker 자격 증명 도우미를 사용하여 로그인할 때 --tlscacert /path/to/ca.crt 플래그를 전달할 수 있습니다.

3 가져오기가 완료되면 Docker 데몬을 다시 시작합니다.

- Linux

```
sudo systemctl restart docker.service
```

- Mac

  Docker 데스크톱 메뉴 옵션 **Docker 다시 시작** 또는 command R 키보드 바로가기를 사용합니다.

다음에 수행할 작업

vSphere Docker 자격 증명 도우미를 설치하고 레지스트리에 연결

내장된 Harbor 레지스트리 인증서를 사용하여 Docker 클라이언트 구성

Docker를 사용하여 내장형 Harbor 레지스트리에서 컨테이너 이미지로 작업하려면 레지스트리 인증서를 Docker 클라이언트에 추가해야 합니다. 인증서는 로그인 중에 Docker에 인증하는 데 사용됩니다.

내장된 Harbor 레지스트리와 상호 작용하도록 Docker 클라이언트를 구성합니다. 이 작업은 vSphere가 내장된 Harbor 레지스트리와 연결하고 상호 작용하기 위해 제공하는 Docker 자격 증명 도우미를 사용하기 위한 준비에 필요합니다.

사전 요구 사항

이 작업은 내장된 Harbor 레지스트리가 사용하도록 설정되어 있고 사용자가 로그인할 수 있다고 가정합니다.

- 감독자 클러스터에서 내장된 Harbor 레지스트리를 사용하도록 설정
- 내장된 Harbor 레지스트리 콘솔에 로그인

또한 이 지침에서는 Docker 데몬이 설치되어 있는 Linux 호스트(Ubuntu)를 사용하고 있다고 가정합니다. Docker가 설치되어 있고 Docker 허브에서 이미지를 끌어올 수 있는지 확인하려면 다음 명령을 실행합니다.

```
docker run hello-world
```

예상 결과:

```
Hello from Docker!
This message shows that your installation appears to be working correctly.
```

참고 이 지침은 Ubuntu 20.04 및 Docker 19.03을 사용하여 확인되었습니다.

절차

1 내장된 Harbor 레지스트리 인증서 `root-certificate.txt`를 다운로드합니다. 내장된 Harbor 레지스트리 인증서 다운로드 및 설치의 내용을 참조하십시오.

2 인증서의 이름을 `ca.crt`로 변경합니다.

3 `ca.crt` 파일을 Docker 호스트에 안전하게 복사합니다.

4 Docker 호스트에서 Harbor IP 주소를 사용하여 개인 레지스트리에 대한 디렉토리 경로를 생성합니다.

```
/etc/docker/certs.d/IP-address-of-harbor/
```

예:

```
mkdir /etc/docker/certs.d/10.179.145.77
```

5 `ca.crt`를 이 디렉토리로 이동합니다.

예:

```
 mv ca.crt /etc/docker/certs.d/10.179.145.77/ca.crt
```

6 Docker 데몬을 다시 시작합니다.

```
sudo systemctl restart docker.service
```

7 Docker 클라이언트를 사용하여 내장형 Harbor 레지스트리에 로그인합니다.

```
docker login https://10.179.145.77
```

다음 메시지가 표시됩니다.

```
WARNING! Your password will be stored unencrypted in /home/ubuntu/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

다음에 수행할 작업

메시지에 표시된 대로 보안을 위해 vSphere Docker 자격 증명 도우미를 다운로드하여 설치합니다. vSphere Docker 자격 증명 도우미를 설치하고 레지스트리에 연결의 내용을 참조하십시오.

vSphere Docker 자격 증명 도우미를 설치하고 레지스트리에 연결

vSphere Docker 자격 증명 도우미 CLI를 사용하여 내장된 Harbor 레지스트리에 컨테이너 이미지를 안전하게 푸시하고 내장된 Harbor 레지스트리에서 컨테이너 이미지를 끌어옵니다.

Kubernetes CLI 도구 다운로드 페이지에는 vSphere Docker 자격 증명 도우미를 다운로드하는 링크가 포함되어 있습니다. vSphere Docker 자격 증명 도우미를 사용하여 Docker 클라이언트를 내장된 Harbor 레지스트리에 안전하게 연결합니다.

사전 요구 사항

- 감독자 클러스터에서 내장된 Harbor 레지스트리를 사용하도록 설정
-
- vSphere 관리자로부터 vSphere에 대한 Kubernetes CLI 도구 다운로드 페이지 링크를 받습니다.
- 또는 vCenter Server에 액세스할 수 있으면 다음과 같이 링크를 가져옵니다.
  - vSphere Client를 사용하여 vCenter Server에 로그인합니다.
  - **vSphere 클러스터 > 네임스페이스**로 이동하여 작업 중인 vSphere 네임스페이스를 선택합니다.
  - **요약** 탭을 선택하고 **상태** 타일을 찾습니다.
  - **CLI 도구에 연결** 머리글 아래의 **열기**를 선택하여 다운로드 페이지를 엽니다. 또는 링크를 **복사**할 수 있습니다.
- Docker 클라이언트를 구성합니다. 내장된 Harbor 레지스트리 인증서를 사용하여 Docker 클라이언트 구성의 내용을 참조하십시오.

절차

1 브라우저를 사용하여 사용 중인 환경에 해당하는 **Kubernetes CLI 도구** 다운로드 URL로 이동합니다.

2 vSphere Docker 자격 증명 도우미 섹션까지 아래로 스크롤합니다.

3 운영 체제를 선택합니다.

4 vsphere-docker-credential-helper.zip 파일을 다운로드합니다.

5 작업 디렉토리에 ZIP 파일 컨텐츠의 압축을 풉니다.

docker-credential-vsphere 바이너리 실행 파일을 사용할 수 있습니다.

6 docker-credential-vsphere 바이너리를 Docker 클라이언트 호스트에 복사합니다.

7 바이너리의 위치를 시스템 경로에 추가합니다.

예를 들어 Linux:

```
mv docker-credential-vsphere /usr/local/bin/docker-credential-vsphere
```

8 셸 또는 터미널 세션에서 docker-credential-vsphere 명령을 실행하여 vSphere Docker 자격 증명 도우미 설치를 확인합니다.

배너 메시지 및 CLI에 대한 명령줄 옵션 목록이 표시됩니다.

```
vSphere login manager is responsible for vSphere authentication.
It allows vSphere users to securely login and logout to access Harbor images.

Usage:
  docker-credential-vsphere [command]

Available Commands:
  help        Help about any command
  login       Login into specific harbor server and get authentication
  logout      Logout from Harbor server and erase user token

Flags:
  -h, --help   help for docker-credential-vsphere

Use "docker-credential-vsphere [command] --help" for more information about a command.
```

9 내장된 Harbor 레지스트리에 로그인합니다.

우선 사용량을 확인합니다.

```
docker-credential-vsphere login -help
Usage:
  docker-credential-vsphere login [harbor-registry] [flags]

Flags:
  -h, --help               help for login
  -s, --service string     credential store service
      --tlscacert string   location to CA certificate (default "/etc/docker/certs.d/*.crt")
  -u, --user string        vSphere username and password
```

그런 후 다음 명령을 사용하여 로그인합니다.

```
docker-credential-vsphere login <container-registry-IP>
```

인증 토큰을 가져와서 저장되고 로그인됩니다.

```
docker-credential-vsphere login 10.179.145.77
Username: administrator@vsphere.local
Password: INFO[0017] Fetched username and password
INFO[0017] Fetched auth token
INFO[0017] Saved auth token
```

10 내장된 Harbor 레지스트리에서 로그아웃합니다.

```
docker-credential-vsphere logout 10.179.145.77
```

다음에 수행할 작업

내장된 Harbor 레지스트리로 이미지 푸시 .

내장된 Harbor 레지스트리로 이미지 푸시

Docker에서 내장된 Harbor 레지스트리의 프로젝트로 이미지를 푸시할 수 있습니다. 내장된 Harbor 레지스트리의 프로젝트는 감독자 클러스터의 vSphere 네임스페이스에 해당합니다.

사전 요구 사항

다음과 같은 작업이 완료된 것으로 가정합니다.

- 감독자 클러스터에서 내장된 Harbor 레지스트리를 사용하도록 설정
- vSphere Docker 자격 증명 도우미를 설치하고 레지스트리에 연결

또한, 이미지를 푸시하려는 Harbor 레지스트리의 프로젝트에 해당하는 네임스페이스에 대한 쓰기 권한이 있는 사용자 계정을 확보합니다.

마지막으로, 레지스트리로 푸시할 수 있는 로컬 이미지가 필요합니다. 다음 명령은 Docker Hub에서 hello-world 이미지를 끌어옵니다. 이미지를 끌어오려면 이 곳의 계정이 필요합니다.

```
docker run hello-world
```

예상 결과:

```
Hello from Docker!
This message shows that your installation appears to be working correctly.
```

docker images 명령을 사용하여 이미지를 확인합니다.

```
docker images
REPOSITORY     TAG                 IMAGE ID            CREATED             SIZE
hello-world    latest              bf756fb1ae65        10 months ago       13.3kB
```

절차

1 vSphere Docker Credential Helper를 통해 Harbor 레지스트리에 로그인합니다.

```
docker-credential-vsphere login <container-registry-IP> --user username@domain.com
```

참고 로그인에 --user *username* 제공이 허용되지만 UPN(사용자 계정 이름) 구문(--user *username@domain.com*)을 사용하여 로그인하고 docker push 명령을 사용해야 합니다.

2 Harbor 레지스트리의 프로젝트로 푸시할 이미지에 사용하려는 네임스페이스와 동일한 이름으로 태그를 지정합니다.

```
docker tag <image-name>[:TAG] <container-registry-IP>/<project-name>/<image-name>[:TAG]
```

예:

```
docker tag hello-world:latest 10.179.145.77/tkgs-cluster-ns/hello-world:latest
```

```
docker images
REPOSITORY                                      TAG                 IMAGE ID
CREATED             SIZE
10.179.145.77/tkgs-cluster-ns/hello-world     latest              bf756fb1ae65        10
months ago       13.3kB
hello-world                                     latest              bf756fb1ae65        10
months ago       13.3kB
```

3 Harbor의 프로젝트로 이미지를 푸시하려면 다음 명령을 실행합니다.

구문:

```
docker push <container-registry-IP>/<namespace-name>/<image_name>
```

예:

```
docker push 10.179.145.77/tkgs-cluster-ns/hello-world:latest
```

예상 결과:

```
The push refers to repository [10.179.145.77/tkgs-cluster-ns/hello-world]
9c27e219663c: Pushed
latest: digest: sha256:90659bf80b44ce6be8234e6ff90a1ac34acbeb826903b02cfa0da11c82cbc042
size: 525
```

4 내장된 Harbor 레지스트리에서 이제 이미지를 사용할 수 있는지 확인합니다.

- 내장된 Harbor 레지스트리 콘솔에 로그인
- **프로젝트 > 프로젝트 이름**에서 프로젝트 링크를 클릭합니다.
- **저장소** 탭을 선택합니다.
- 레지스트리에 푸시한 이미지가 `namespace/image-name` 형식(예: `tkgs-cluster-ns/hello-world`)으로 표시되어야 합니다.
- 이 이미지를 선택하면 `latest` 태그와 기타 메타데이터가 표시됩니다.

5 **저장소** 탭으로 다시 이동합니다.

6 **이미지 푸시 Docker 명령** 드롭다운 메뉴를 선택합니다. 이미지에 태그를 지정하고 이 저장소에 푸시하는 명령이 제공됩니다.

예

다음은 내장된 Harbor 레지스트리로 이미지를 푸시하는 또 다른 예입니다.

```
docker tag busybox:latest <container-registry-IP>/<namespace-name>/busybox:latest
docker push <container-registry-IP>/busybox/busybox:latest
```

다음에 수행할 작업

Harbor 레지스트리의 이미지를 사용하여 vSphere 포드를 배포합니다. 내장된 Harbor 레지스트리를 사용하여 vSphere 포드에 애플리케이션 배포의 내용을 참조하십시오.

내장된 Harbor 레지스트리에서 이미지 지우기

vSphere 관리자는 DevOps 엔지니어의 요청에 따라 전용 이미지 레지스트리에서 프로젝트의 이미지를 지울 수 있습니다. 전용 이미지 레지스트리에서 이미지를 지우면 포드에서 생성된 이미지에 대한 모든 참조가 삭제되지만 이미지 레지스트리에서 이미지가 제거되지는 않습니다.

DevOps 엔지니어가 프로젝트에 이미지가 너무 많이 저장되었다고 보고하면, vSphere 관리자는 전용 이미지 레지스트리에서 해당 프로젝트에 대한 이미지를 제거할 수 있습니다. 프로젝트에서 이미지를 제거하면 해당 이미지에 대한 모든 참조가 삭제되지만 vSphere 데이터스토어에서는 삭제되지 않습니다. 매주 토요일 오전 2시에 가비지 수집기 서비스는 애플리케이션에서 참조하지 않는 전체 전용 이미지 레지스트리에서 모든 이미지를 삭제합니다.

절차

1 vSphere Client에서 네임스페이스로 이동합니다.

2 **구성**을 선택하고 **일반**을 선택합니다.

3 **내장된 레지스트리** 옆에 있는 **제거**를 클릭합니다.

Tanzu Kubernetes 클러스터에서 내장된 Harbor 레지스트리 사용

내장된 Harbor 레지스트리를 Tanzu Kubernetes Grid 서비스에서 프로비저닝된 Tanzu Kubernetes 클러스터에 배포하는 이미지에 대한 개인 컨테이너 레지스트리로 사용할 수 있습니다.

vSphere with Tanzu에는 감독자 클러스터에서 사용하도록 설정하고 Tanzu Kubernetes 클러스터에 컨테이너 기반 워크로드를 배포하는 데 사용할 수 있는 Harbor 레지스트리 인스턴스가 포함됩니다.

내장된 Harbor 레지스트리가 감독자 클러스터에서 사용되도록 설정되면 Tanzu Kubernetes Grid 서비스는 레지스트리 인스턴스에 대한 루트 CA 인증서를 Tanzu Kubernetes 클러스터 노드에 설치합니다. 이 인증서는 새 클러스터와 기존 클러스터 모두에 설치됩니다(조정 루프를 통해). 여기에서 워크로드 YAML에 개인 레지스트리를 지정하여 클러스터에서 이미지를 실행할 수 있습니다.

워크플로

다음 워크플로를 사용하여 Tanzu Kubernetes 클러스터 노드에서 개인 레지스트리에 안전하게 액세스하고 컨테이너 이미지를 끌어옵니다.

단계	작업	지침
0	Tanzu Kubernetes 클러스터에서 내장된 Harbor 레지스트리를 사용하기 위한 워크플로를 검토합니다.	장 15 vSphere with Tanzu 워크로드에 컨테이너 레지스트리 사용의 내용을 참조하십시오.
1	내장된 Harbor 레지스트리를 감독자 클러스터에서 사용하도록 설정합니다.	감독자 클러스터에서 내장된 Harbor 레지스트리를 사용하도록 설정의 내용을 참조하십시오.
2	레지스트리 서비스 암호를 사용하여 각 클러스터에 대한 kubeconfig를 구성합니다.	아래 지침 참조: 내장된 Harbor 레지스트리에 대한 이미지 끌어오기 암호를 사용하여 Tanzu Kubernetes 클러스터를 구성합니다.
3	개인 컨테이너 레지스트리를 지정하도록 워크로드 YAML을 구성합니다.	아래 지침 참조: 내장된 Harbor 레지스트리에 대한 이미지 끌어오기 암호를 사용하여 Tanzu Kubernetes 클러스터를 구성합니다.
4	이미지를 내장된 Harbor 레지스트리로 푸시하려면 Docker 클라이언트를 구성하고 vSphere Docker 자격 증명 도우미를 설치합니다.	내장된 Harbor 레지스트리 인증서를 사용하여 Docker 클라이언트 구성 및 내장된 Harbor 레지스트리로 이미지 푸시 의 내용을 참조하십시오.

내장된 Harbor 레지스트리에 대한 이미지 Pull 암호를 사용하여 Tanzu Kubernetes 클러스터 구성

이미지 Pull 암호를 사용하여 kubeconfig를 구성하여 Tanzu Kubernetes 클러스터를 내장된 Harbor 레지스트리 또는 외부 개인 레지스트리 중 하나인 개인 컨테이너 레지스트리에 연결합니다.

1 감독자 클러스터에 연결합니다. vCenter Single Sign-On 사용자로 감독자 클러스터에 연결의 내용을 참조하십시오.

2 대상 Tanzu Kubernetes 클러스터가 프로비저닝된 vSphere 네임스페이스로 컨텍스트를 전환합니다.

```
kubectl config use-context tkgs-cluster-ns
```

3 vSphere 네임스페이스의 이미지 Pull 암호를 가져와서 파일에 저장합니다.

```
kubectl get secret -n <vsphere-namespace> <vsphere-namespace>-default-image-pull-secret -o
yaml > <path>/image-pull-secret.yaml
```

예:

```
kubectl get secret -n tkgs-cluster-ns tkgs-cluster-ns-default-image-pull-secret -o yaml >
tanzu/image-pull-secret.yaml
```

4 텍스트 편집기로 image-pull-secret.yaml을 엽니다. 최소한 필요한 변경을 수행하고 완료되면 파일을 저장합니다.

```
apiVersion: v1
data:
  .dockerconfigjson: ewoJCQkJImF1dGhzJUV2s1ZVZwWVFuWmp...
kind: Secret
metadata:
  creationTimestamp: "2020-11-12T02:41:08Z"
  managedFields:
  - apiVersion: v1
    ...
  name: harbor-registry-secret   #OPTIONAL: Change if desired
  namespace: default           #REQUIRED: Enter the Kuberentes namespace
  ownerReferences:
  - apiVersion: registryagent.vmware.com/v1alpha1
    ...
  resourceVersion: "675868"
```

```
  selfLink: /api/v1/namespaces/tkgs-cluster-ns/secrets/tkgs-cluster-ns-default-image-pull-
secret
  uid: 66606b41-7363-4b74-a3f2-4436f83f
type: kubernetes.io/dockerconfigjson
```

- 필수: namespace 값을 클러스터의 적절한 Kubernetes 네임스페이스(예: **default**)와 일치하도록 변경합니다.

  참고 이미지 Pull 암호를 구성하려면 Kubernetes 네임스페이스를 지정합니다. Tanzu Kubernetes 클러스터가 이미 있는 경우, 해당 클러스터로 컨텍스트를 전환하고 `kubectl get namespaces`를 실행하여 사용 가능한 Kubernetes 네임스페이스를 나열합니다. 필요한 경우 계속하기 전에 대상 네임스페이스를 생성합니다. Tanzu Kubernetes 클러스터가 없는 경우 `default` 네임스페이스를 사용할 수 있습니다.

- 선택 사항: name 값을 의미 있는 텍스트(예: **harbor-registry-secret** 또는 **private-registry-secret**)로 변경합니다.

5 Tanzu Kubernetes 클러스터에 액세스하는 데 사용할 수 있는 kubeconfig 파일을 생성합니다.

```
kubectl get secret -n <vsphere-namespace> <cluster-name>-kubeconfig -o
jsonpath='{.data.value}' | base64 -d > <path>/cluster-kubeconfig
```

*<vsphere-namespace>*를 대상 Tanzu Kubernetes 클러스터가 프로비저닝된 vSphere 네임스페이스의 이름으로 바꿉니다. *<cluster-name>*은 Tanzu Kubernetes 클러스터의 이름으로 바꿉니다.

```
kubectl get secret -n tkgs-cluster-ns tkgs-cluster-5-kubeconfig -o
jsonpath='{.data.value}' | base64 -d > tanzu/cluster-kubeconfig
```

6 Tanzu Kubernetes 클러스터에서 레지스트리 서비스 암호를 생성합니다. 로컬에 저장하고 업데이트한 이미지 Pull 암호 파일을 참조합니다.

```
kubectl --kubeconfig=<path>/cluster-kubeconfig apply -f <path>/image-pull-secret.yaml
```

예:

```
kubectl --kubeconfig=tanzu/cluster-kubeconfig apply -f tanzu/image-pull-secret.yaml
```

레지스트리 서비스 암호가 생성된 것을 볼 수 있습니다.

```
secret/harbor-registry-secret created
```

개인 컨테이너 레지스트리에서 이미지를 끌어오도록 Tanzu Kubernetes 워크로드 구성

Tanzu Kubernetes 클러스터 워크로드를 위해 개인 컨테이너 레지스트리에서 이미지를 끌어오려면 개인 레지스트리 세부 정보를 사용하여 워크로드 YAML을 구성합니다.

이 절차는 전용 컨테이너 레지스트리 또는 내장된 Harbor 레지스트리에서 이미지를 끌어오는 데 사용할 수 있습니다. 이 예에서는 내장된 Harbor 레지스트리에 저장된 이미지를 사용하고 이전에 구성된 이미지 끌어오기 암호를 활용하는 포드 규격을 생성합니다.

1 개인 레지스트리에 대한 세부 정보를 사용하여 예제 포드 규격을 생성합니다.

```
apiVersion: v1
kind: Pod
metadata:
  name: <workload-name>
  namespace: <kubernetes-namespace>
spec:
  containers:
  - name: private-reg-container
    image: <Registry-IP-Address>/<vsphere-namespace>/<image-name>:<version>
  imagePullSecrets:
  - name: <registry-secret-name>
```

- <workload-name>을 포드 워크로드의 이름으로 바꿉니다.
- <kubernetes-namespace>는 포드를 생성할 클러스터의 Kubernetes 네임스페이스로 바꿉니다. 이것은 레지스트리 서비스 이미지 끌어오기 암호가 클러스터 Tanzu Kubernetes에 저장되는 동일한 Kubernetes 네임스페이스(예: 기본 네임스페이스)여야 합니다.
- <Registry-IP-Address>를 감독자 클러스터에서 실행되는 내장된 Harbor 레지스트리 인스턴스의 IP 주소로 바꿉니다.
- *<vsphere-namespace>*는 대상 Tanzu Kubernetes가 프로비저닝된 vSphere 네임스페이스로 바꿉니다.
- <image-name>은 원하는 이미지 이름으로 바꿉니다.
- <version>은 적절한 이미지 버전(예: "latest")으로 바꿉니다.
- <registry-secret-name>은 이전에 생성한 레지스트리 서비스 이미지 끌어오기 암호의 이름으로 바꿉니다.

2 정의한 포드 규격을 기반으로 Tanzu Kubernetes 클러스터에 워크로드를 생성합니다.

```
kubectl --kubeconfig=<path>/cluster-kubeconfig apply -f <pod.yaml>
```

포드는 레지스트리에서 끌어온 이미지에서 생성해야 합니다.

Tanzu Kubernetes 클러스터에서 외부 컨테이너 레지스트리 사용

Tanzu Kubernetes 클러스터 포드에서 외부 컨테이너 레지스트리를 사용할 수 있습니다. 이것은 내장된 Harbor 레지스트리 사용에 대한 대안입니다.

외부 개인 레지스트리 사용 사례

컨테이너 레지스트리는 컨테이너 이미지를 저장 및 공유하기 위한 중앙 집중식 저장소 역할을 하는 Kubernetes 배포에 중요한 기능을 제공합니다. 가장 일반적으로 사용되는 공용 컨테이너 레지스트리는 DockerHub입니다. 개인 컨테이너 레지스트리 오퍼링은 많이 있습니다. VMware Harbor는 오픈 소스, 클라우드 네이티브, 개인 컨테이너 레지스트리입니다. vSphere with Tanzu에는 vSphere 포드 및 Tanzu Kubernetes 클러스터에서 실행되는 포드에 대한 개인 컨테이너 레지스트리로 사용할 수 있는 Harbor 인스턴스가 내장되어 있습니다. 자세한 내용은 감독자 클러스터에서 내장된 Harbor 레지스트리를 사용하도록 설정의 내용을 참조하십시오.

vSphere with Tanzu와 함께 제공되는 내장된 Harbor 레지스트리에는 NSX-T 네트워킹이 필요합니다. vSphere 네트워킹을 사용하는 경우에는 사용할 수 없습니다. 또한 Tanzu Kubernetes 클러스터와 통합하려는 자체적인 개인 컨테이너 레지스트리를 이미 실행 중일 수 있습니다. 이 경우 자체 서명된 인증서가 있는 개인 레지스트리를 신뢰하도록 Tanzu Kubernetes Grid 서비스를 구성하여 Tanzu Kubernetes 클러스터에서 실행되는 Kubernetes 포드가 외부 레지스트리를 사용하도록 할 수 있습니다.

외부 개인 레지스트리 요구 사항

Tanzu Kubernetes 클러스터에서 외부 개인 레지스트리를 사용하려면 vSphere with Tanzu 버전 7 U2 이상을 사용해야 합니다.

Tanzu Kubernetes 클러스터 및 Tanzu Kubernetes 릴리스 노드 VM에서 실행되는 Kubernetes 포드에서만 자체적인 개인 레지스트리를 사용할 수 있습니다. ESXi 호스트에서 기본적으로 실행되는 vSphere 포드에서는 자체적인 개인 레지스트리를 사용할 수 없습니다. vSphere 포드에 지원되는 레지스트리는 vSphere with Tanzu 플랫폼에 내장된 Harbor 레지스트리입니다.

개인 레지스트리에 대해 Tanzu Kubernetes Grid 서비스를 구성하면 프로비저닝된 새 클러스터가 개인 레지스트리를 지원하게 됩니다. 기존 클러스터가 개인 레지스트리를 지원하려면 `TkgServiceConfiguration`을 적용하기 위해 롤링 업데이트가 필요합니다. Tanzu Kubernetes 클러스터 업데이트의 내용을 참조하십시오. 또한 사용자 지정 `TkgServiceConfiguration`을 처음 생성하면 시스템에서 롤링 업데이트가 시작됩니다.

외부 개인 레지스트리 구성

Tanzu Kubernetes 클러스터에서 자체적인 개인 레지스트리를 사용하려면 HTTPS를 통해 개인 레지스트리 컨텐츠를 제공하도록 하나 이상의 자체 서명된 인증서로 Tanzu Kubernetes Grid 서비스를 구성합니다.

`TkgServiceConfiguration`은 개인 레지스트리에 대해 자체 서명된 인증서를 지원하도록 업데이트됩니다. 특히 `trust` 필드가 있는 새 `additionalTrustedCAs` 섹션이 추가되어 Tanzu Kubernetes 클러스터가 신뢰해야 하는 자체 서명된 인증서를 얼마든지 정의할 수 있습니다. 이 기능을 사용하면 인증서 목록을 쉽게 정의하고 순환이 필요할 경우 인증서를 업데이트할 수 있습니다.

TkgServiceConfiguration이 업데이트되고 적용되고 나면 다음에 클러스터가 생성될 때 새 클러스터에 TLS 인증서가 적용됩니다. 다시 말해, TkgServiceConfiguration.trust.additionalTrustedCAs에 업데이트를 적용해도 Tanzu Kubernetes 클러스터에 대한 자동 롤링 업데이트가 트리거되지 않습니다.

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TkgServiceConfiguration
metadata:
  name: tkg-service-configuration
spec:
  defaultCNI: antrea
  trust:
    additionalTrustedCAs:
      - name: first-cert-name
        data: base64-encoded string of a PEM encoded public cert 1
      - name: second-cert-name
        data: base64-encoded string of a PEM encoded public cert 2
```

업데이트를 적용하려면 다음 명령을 실행합니다.

```
kubectl apply -f tkgserviceconfiguration.yaml
```

Tanzu Kubernetes Grid 서비스 규격이 개인 레지스트리 인증서로 업데이트되기 때문에 Tanzu Kubernetes 클러스터에서 내장된 Harbor 레지스트리를 사용할 때처럼 Tanzu Kubernetes 클러스터 kubeconfig에 공용 키를 추가할 필요가 없습니다.

개인 컨테이너 레지스트리에서 이미지를 끌어오도록 Tanzu Kubernetes 워크로드 구성

Tanzu Kubernetes 클러스터 워크로드를 위해 개인 컨테이너 레지스트리에서 이미지를 끌어오려면 개인 레지스트리 세부 정보를 사용하여 워크로드 YAML을 구성합니다.

이 절차는 전용 컨테이너 레지스트리 또는 내장된 Harbor 레지스트리에서 이미지를 끌어오는 데 사용할 수 있습니다. 이 예에서는 내장된 Harbor 레지스트리에 저장된 이미지를 사용하고 이전에 구성된 이미지 끌어오기 암호를 활용하는 포드 규격을 생성합니다.

1 개인 레지스트리에 대한 세부 정보를 사용하여 예제 포드 규격을 생성합니다.

```
apiVersion: v1
kind: Pod
metadata:
  name: <workload-name>
  namespace: <kubernetes-namespace>
spec:
  containers:
```

```
  - name: private-reg-container
    image: <Registry-IP-Address>/<vsphere-namespace>/<image-name>:<version>
  imagePullSecrets:
  - name: <registry-secret-name>
```

- <workload-name>을 포드 워크로드의 이름으로 바꿉니다.
- <kubernetes-namespace>는 포드를 생성할 클러스터의 Kubernetes 네임스페이스로 바꿉니다. 이것은 레지스트리 서비스 이미지 끌어오기 암호가 클러스터 Tanzu Kubernetes에 저장되는 동일한 Kubernetes 네임스페이스(예: 기본 네임스페이스)여야 합니다.
- <Registry-IP-Address>를 감독자 클러스터에서 실행되는 내장된 Harbor 레지스트리 인스턴스의 IP 주소로 바꿉니다.
- *<vsphere-namespace>*는 대상 Tanzu Kubernetes가 프로비저닝된 vSphere 네임스페이스로 바꿉니다.
- <image-name>은 원하는 이미지 이름으로 바꿉니다.
- <version>은 적절한 이미지 버전(예: "latest")으로 바꿉니다.
- <registry-secret-name>은 이전에 생성한 레지스트리 서비스 이미지 끌어오기 암호의 이름으로 바꿉니다.

2 정의한 포드 규격을 기반으로 Tanzu Kubernetes 클러스터에 워크로드를 생성합니다.

```
kubectl --kubeconfig=<path>/cluster-kubeconfig apply -f <pod.yaml>
```

포드는 레지스트리에서 끌어온 이미지에서 생성해야 합니다.

외부 개인 레지스트리의 신뢰 필드

TkgServiceConfiguration의 additionalTrustedCAs 섹션에 인증서 항목(PEM으로 인코딩된 공용 인증서의 base64로 인코딩된 문자열)을 추가합니다. 데이터는 TkgServiceConfiguration에 일반 텍스트로 저장된 공용 인증서입니다.

표 15-1. 개인 레지스트리의 신뢰 필드

필드	설명
trust	섹션 마커. 데이터를 허용하지 않습니다.
additionalTrustedCAs	섹션 마커. 각각에 대한 이름과 데이터가 는 인증서 어레이를 포함합니다.
name	TLS 인증서의 이름입니다.
data	PEM으로 인코딩된 공용 인증서의 base64로 인코딩된 문자열입니다.

외부 개인 레지스트리 인증서 제거

`TkgServiceConfiguration`의 `additionalTrustedCAs` 섹션에 있는 인증서 목록에서 인증서를 제거하고 `TkgServiceConfiguration`을 Tanzu Kubernetes Grid 서비스에 적용합니다.

외부 개인 레지스트리 인증서 순환

인증서를 순환하기 위해 VI 관리자 또는 DevOps 엔지니어가 `TkgServiceConfiguration` 또는 Tanzu Kubernetes 클러스터 규격의 인증서 내용을 변경하고 이 구성을 적용하여 TKC에 대한 롤링 업데이트를 트리거합니다.

외부 개인 레지스트리 인증서 문제 해결

신뢰할 수 있는 인증서로 Tanzu Kubernetes Grid 서비스를 구성하고 자체 서명된 인증서를 kubeconfig 클러스터에 추가하면 자체 서명된 인증서를 사용하는 개인 레지스트리에서 컨테이너 이미지를 성공적으로 끌어올 수 있습니다.

다음 명령은 포드 워크로드에 대해 컨테이너 이미지를 성공적으로 끌어왔는지 확인하는 데 유용할 수 있습니다.

```
kubectl describe pod PODNAME
```

이 명령은 주어진 포드에 대한 자세한 상태 및 오류 메시지를 보여줍니다. 클러스터에 사용자 지정 인증서를 추가하기 전에 이미지 끌어오기를 시도하는 예:

```
Events:
  Type     Reason                          Age                From               Message
  ----     ------                          ----               ----               -------
  Normal   Scheduled                       33s                default-scheduler  ...
  Normal   Image                           32s                image-controller   ...
  Normal   Image                           15s                image-controller   ...
  Normal   SuccessfulRealizeNSXResource  7s (x4 over 31s)  nsx-container-ncp  ...
  Normal   Pulling                         7s                 kubelet            Waiting test-gc-
e2e-demo-ns/testimage-8862e32f68d66f727d1baf13f7eddef5a5e64bbd-v10612
  Warning  Failed                          4s                 kubelet            failed to get
images: ... Error: ... x509: certificate signed by unknown authority
```

다음 명령을 실행하는 경우:

```
kubectl get pods
```

전체 포드 상태 보기에서도 `ErrImagePull` 오류를 볼 수 있습니다.

```
NAME                                      READY   STATUS         RESTARTS   AGE
testimage-nginx-deployment-89d4fcff8-2d9pz   0/1     Pending        0          17s
testimage-nginx-deployment-89d4fcff8-7kp9d   0/1     ErrImagePull   0          79s
```

```
testimage-nginx-deployment-89d4fcff8-7mpkj   0/1     Pending         0          21s
testimage-nginx-deployment-89d4fcff8-fszth   0/1     ErrImagePull    0          50s
testimage-nginx-deployment-89d4fcff8-sjnjw   0/1     ErrImagePull    0          48s
testimage-nginx-deployment-89d4fcff8-xr5kg   0/1     ErrImagePull    0          79s
```

"x509: certificate signed by unknown authority" 및 "ErrImagePull" 오류는 클러스터가 개인 컨테이너 레지스트리에 연결할 수 있는 올바른 인증서로 구성되지 않았음을 나타냅니다. 인증서가 누락되었거나 잘못 구성되었습니다.

인증서를 구성한 후 개인 레지스트리에 연결하는 데 오류가 발생하는 경우 구성에 적용된 인증서가 클러스터에 적용되었는지 확인할 수 있습니다. SSH를 사용하여 구성에 적용했던 인증서가 제대로 적용되었는지 여부를 확인할 수 있습니다.

SSH를 통해 작업자 노드에 연결하여 두 가지 조사 단계를 수행할 수 있습니다.

1 `/etc/ssl/certs/` 폴더에서 `tkg-<cert_name>.pem`이라는 파일을 확인합니다. 여기서 `<cert_name>`은 `TkgServiceConfiguration`에 추가된 인증서의 "name" 속성입니다. 인증서가 `TkgServiceConfiguration`에 있는 인증서와 일치하고 개인 레지스트리 사용이 여전히 작동하지 않으면 다음 단계를 수행하여 추가 진단을 진행합니다.

2 `openssl s_client -connect hostname:port_num` 명령을 실행하여 자체 서명된 인증서를 사용하는 대상 서버에 다음 openssl 연결 테스트를 실행합니다. 여기서 hostname은 자체 서명된 인증서를 사용하는 개인 레지스트리의 호스트 이름/DNS 이름이고 port_num은 서비스가 실행되고 있는 포트 번호(일반적으로 HTTPS의 경우 443임)입니다. 자체 서명된 인증서를 사용하는 끝점에 연결하려고 할 때 openssl이 정확히 어떤 오류를 반환하는지 확인하고, 거기서부터 상황을 해결할 수 있습니다(예를 들어 `TkgServiceConfiguration`에 올바른 인증서를 추가하여). Tanzu Kubernetes 클러스터에 잘못된 인증서가 내장되어 있는 경우 올바른 인증서로 Tanzu Kubernetes Grid 서비스 구성을 업데이트하고, Tanzu Kubernetes 클러스터를 삭제한 다음 올바른 인증서가 포함된 구성을 사용하여 다시 생성해야 합니다.

vSphere Lifecycle Manager 작업 16

vSphere 관리자는 단일 VMware vSphere Lifecycle Manager 이미지로 관리하는 vSphere 클러스터에서 vSphere with Tanzu를 사용하도록 설정할 수 있습니다. 그런 다음 vSphere Lifecycle Manager에서 관리되는 동안 감독자 클러스터를 사용할 수 있습니다.

vSphere Lifecycle Manager를 사용하면 환경에서 ESXi 호스트 및 클러스터를 관리할 수 있습니다. 감독자 클러스터를 최신 버전의 vSphere with Tanzu로 업그레이드할 수 있습니다. 감독자 클러스터에서 호스트의 ESXi 버전을 업그레이드할 수도 있습니다.

vSphere Lifecycle Manager는 vCenter Server에서 실행되는 서비스입니다. vCenter Server을 배포하면 vSphere Lifecycle Manager 사용자 인터페이스가 HML5 기반 vSphere Client에서 사용되도록 설정됩니다.

vSphere Lifecycle Manager에 대한 자세한 내용은 "호스트 및 클러스터 수명 주기 관리" 설명서를 참조하십시오.

본 장은 다음 항목을 포함합니다.

요구 사항

vSphere Lifecycle Manager에서 관리되는 vSphere 클러스터에 vSphere with Tanzu을 구성하려면 환경이 특정 요구 사항을 충족해야 합니다.

시스템 요구 사항

vSphere with Tanzu을 사용하도록 설정하려면 vSphere 클러스터의 구성 요소가 다음 요구 사항을 충족하는지 확인합니다.

- NSX-T Data Center를 사용하는 경우 vCenter Server와 ESXi 버전이 7.0 업데이트 2인지 확인합니다.
- vSphere 네트워킹을 사용하는 경우 vCenter Server와 ESXi 버전이 7.0 업데이트 1 이상인지 확인합니다.
- 감독자 클러스터의 일부로 사용하려는 모든 ESXi 호스트에 VMware vSphere 7 Enterprise Plus with Add-on for Kubernetes 라이센스가 할당되어 있는지 확인합니다.
- vSphere 클러스터에 HA 및 DRS를 사용하도록 설정되어 있는지 확인합니다.
- vSphere Distributed Switch 버전 7.0 업데이트 2가 구성되어 있는지 확인합니다.
- 클러스터에 vSphere 네트워킹 또는 NSX-T Data Center 3.1 이상 버전이 구성되어 있는지 확인합니다. vSphere Lifecycle Manager 이미지를 사용하여 이전 NSX-T Data Center 버전으로 구성된 클러스터를 관리할 수 없습니다.

vSphere Lifecycle Manager로 관리되는 클러스터에서 vSphere with Tanzu사용

Kubernetes 워크로드를 실행하기 위해 단일 vSphere Lifecycle Manager 이미지로 관리하는 클러스터에서 vSphere with Tanzu를 사용하도록 설정할 수 있습니다. 사용하도록 설정되면 vSphere Lifecycle Manager를 사용하여 감독자 클러스터를 관리할 수 있습니다.

NSX-T Data Center를 사용하는 vSphere with Tanzu에서 클러스터를 사용하도록 설정하면 vSphere Lifecycle Manager는 클러스터의 모든 ESXi 호스트에 Spherelet VIB(vSphere 설치 번들)를 설치합니다. 클러스터를 사용하도록 설정하면 vCenter와 함께 제공되는 Kubernetes 버전이 할당됩니다. 설치가 완료되면 WCP 서비스는 Spherelet 시작 및 구성과 같은 설치 후 작업을 수행합니다.

클러스터를 사용하도록 설정하는 단계는 vSphere 네트워킹으로 워크로드 관리 사용에서 참조하십시오.

감독자 클러스터 업그레이드

단일 vSphere Lifecycle Manager 이미지를 사용하는 vSphere에 대한 vSphere with Tanzu 클러스터, Kubernetes 버전 및 Kubernetes CLI 도구를 지원하는 vSphere 인프라를 포함한 최신 버전의 vSphere with Tanzu로 업데이트할 수 있습니다.

감독자 클러스터에서 호스트의 ESXi 버전을 업그레이드합니다. 업그레이드하는 동안 모든 ESXi 호스트의 Spherelet VIB가 업그레이드됩니다.

vSphere Lifecycle Manager는 DRS를 사용하고 업데이트 적용 전에 호스트를 유지 보수 모드로 전환합니다. DRS는 업데이트 적용이 성공할 수 있도록 우선 vCenter Server를 실행하는 가상 시스템을 다른 호스트(예: 호스트에 선호되거나 호스트의 로컬 스토리지에서 실행되는 VM)로 마이그레이션하고 vSphere 포드를 포함한 워크로드를 다른 호스트로 마이그레이션하려고 합니다.

참고 단일 vSphere Lifecycle Manager 이미지를 사용하는 클러스터에서만 vSphere Lifecycle Manager를 사용하여 감독자 클러스터를 업그레이드할 수 있습니다.

절차

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

2 **업데이트** 탭을 선택합니다.

3 업데이트하려는 **사용 가능한 버전**을 선택합니다.

예를 들어, `v1.17.4-vsc0.0.2-16293900` 버전을 선택합니다.

4 업데이트를 적용할 감독자 클러스터를 선택합니다.

5 업데이트를 시작하려면 **업데이트 적용**을 클릭합니다.

6 **최근 작업** 창을 사용하여 업데이트 상태를 모니터링합니다.

감독자 클러스터에 호스트 추가

vSphere 관리자는 더 많은 워크로드를 실행하기 위해 감독자 클러스터를 확장해야 할 수 있습니다. 클러스터에 용량을 추가하기 위해 단일 vSphere Lifecycle Manager 이미지를 사용하는 클러스터에 ESXi 호스트를 추가할 수 있습니다.

NSX-T Data Center로 구성된 감독자 클러스터에 호스트를 추가하면 vSphere Lifecycle Manager는 호스트에 Spherelet VIB와 이미지를 설치합니다. 설치 후 vSphere with Tanzu는 새로 추가된 호스트에서 Spherelet 프로세스를 구성하여 ESXi에서 기본적으로 컨테이너를 실행할 수 있습니다.

사전 요구 사항

- 호스트에 대한 루트 사용자 계정의 사용자 이름 및 암호를 가져옵니다.
- 방화벽 뒤의 호스트가 vCenter Server와 통신할 수 있는지 확인합니다.

절차

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

2 데이터 센터, 클러스터 또는 폴더를 오른쪽 버튼으로 클릭하고 **호스트 추가**를 선택합니다.

3 호스트의 IP 주소나 이름을 입력하고 **다음**을 클릭합니다.

4 관리자 자격 증명을 입력한 후 **다음**을 클릭합니다.

5 호스트 요약을 확인한 후 **다음**을 클릭합니다.

6 호스트에 라이센스 키를 할당하고 **마침**을 클릭합니다.

7 **호스트 추가** 마법사에서 **다음**을 클릭합니다.

8 요약을 검토하고 **마침**을 클릭합니다.

참고 호스트가 동일한 데이터 센터에 속하는 경우 호스트를 감독자 클러스터로 이동할 수 있습니다. 호스트를 이동하려면 호스트를 유지 보수 모드로 전환하고 클러스터로 끌어옵니다.

감독자 클러스터에서 호스트 제거

vSphere 관리자는 비용 절감을 위해 감독자 클러스터를 축소해야 할 수 있습니다. 감독자 클러스터의 용량을 줄이려면 단일 vSphere Lifecycle Manager 이미지를 사용하는 클러스터에서 ESXi 호스트를 제거하면 됩니다.

NSX-T Data Center로 구성된 감독자 클러스터에서 호스트를 제거하면 vSphere with Tanzu는 Spherelet의 구성을 지우고 ESXi 호스트에서 Spherelet 프로세스를 중지합니다. 그런 다음 vSphere with Tanzu가 호스트에서 Spherelet VIB와 이미지를 제거하고 vSphere Lifecycle Manager가 클러스터 제어부에서 호스트 메타데이터를 제거합니다.

사전 요구 사항

클러스터에서 호스트를 제거하려면 먼저 호스트에서 실행 중인 모든 가상 시스템의 전원을 끄거나 가상 시스템을 새 호스트로 마이그레이션해야 합니다.

절차

1 vSphere Client에서 호스트를 제거할 클러스터로 이동합니다.

2 호스트를 마우스 오른쪽 버튼으로 클릭하고 팝업 메뉴에서 **유지 보수 모드 시작**을 선택합니다.

3 표시되는 확인 대화상자에서 **예**를 클릭합니다.

확인 대화상자는 전원이 꺼져 있는 가상 시스템을 다른 호스트로 이동할 것인지 여부를 묻습니다. 해당 가상 시스템을 클러스터 내의 호스트에 등록된 상태로 유지하려는 경우 이 옵션을 선택합니다.

호스트 아이콘이 변경되고 “유지 보수 모드”라는 용어가 괄호 안에 표시된 이름에 추가됩니다.

4 인벤토리에서 호스트 아이콘을 선택한 후 새 위치로 끌어갑니다.

호스트를 다른 클러스터나 다른 데이터 센터로 이동할 수 있습니다.

vCenter Server가 호스트를 새 위치로 이동합니다.

5 호스트를 마우스 오른쪽 버튼으로 클릭하고 팝업 메뉴에서 **유지 보수 모드 종료**를 선택합니다.

6 (선택 사항) 필요에 따라 가상 시스템을 다시 시작합니다.

감독자 클러스터 사용 안 함

단일 vSphere Lifecycle Manager 이미지를 사용하는 vSphere 클러스터에서 vSphere with Tanzu를 사용하지 않도록 설정하여 기존 워크로드에 사용할 수 있도록 할 수 있습니다.

클러스터에서 vSphere with Tanzu를 사용하지 않도록 설정하면 vSphere Lifecycle Manager는 각 ESXi 호스트에서 Spherelet VIB 및 이미지를 제거하고 WCP 서비스는 클러스터에서 모든 워크로드를 중지하고 삭제합니다.

절차

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

2 **클러스터** 탭을 선택합니다.

3 vSphere with Tanzu를 사용하지 않도록 설정하려는 클러스터를 선택합니다.

4 **사용 안 함**을 클릭합니다.

클러스터 사용 안 함 대화 상자가 나타나고 모든 Kubernetes 워크로드 및 NSX-T Data Center 구성이 클러스터에서 사용되지 않도록 설정된다는 메시지가 표시됩니다.

5 **사용 안 함**을 클릭합니다.

vSphere with Tanzu 환경 업데이트 17

vSphere with Tanzu 클러스터를 지원하는 vSphere 인프라, Kubernetes 버전 및 vSphere에 대한 Kubernetes CLI 도구를 포함하여 최신 버전의 vSphere with Tanzu로 업데이트할 수 있습니다.

본 장은 다음 항목을 포함합니다.

vSphere with Tanzu 업데이트 정보

vSphere with Tanzu는 감독자 클러스터와 Tanzu Kubernetes Grid 서비스 클러스터 및 이러한 클러스터를 지원하는 인프라에 대한 롤링 업데이트를 지원합니다.

감독자 클러스터 및 Tanzu Kubernetes Grid 서비스 클러스터 업데이트 방법

vSphere with Tanzu는 감독자 클러스터 및 Tanzu Kubernetes Grid 서비스 클러스터에 대해 롤링 업데이트 모델을 사용합니다. 롤링 업데이트 모델은 업데이트 프로세스 중에 클러스터 워크로드에 대한 다운타임을 최소화합니다. 롤링 업데이트에는 Kubernetes 소프트웨어 버전 및 Kubernetes 클러스터를 지원하는 인프라 및 서비스(예: 가상 시스템 구성 및 리소스, vSphere 서비스 및 네임스페이스, 사용자 지정 리소스) 업그레이드가 포함됩니다.

업데이트가 성공하려면 구성이 몇 가지 호환성 요구 사항을 충족해야 합니다. 그래야 시스템이 재확인 조건을 적용하여 클러스터가 업데이트될 준비가 되었는지 확인하고 클러스터 업그레이드가 실패할 경우 롤백을 지원할 수 있습니다.

참고 vSphere with Tanzu 업데이트에는 Kubernetes 소프트웨어 버전을 업그레이드하는 것 이상이 포함됩니다. 이러한 프로세스는 "업데이트"라는 용어를 사용하여 설명합니다. "업그레이드"라는 용어는 소프트웨어 버전을 증가시키는 제한된 형태의 업데이트이기 때문입니다.

감독자 클러스터 업데이트와 Tanzu Kubernetes Grid 서비스 클러스터 업데이트 간의 종속성

감독자 클러스터 및 Tanzu Kubernetes Grid 서비스 클러스터를 개별적으로 업데이트합니다. 하지만 이 둘 사이에는 종속성이 있습니다.

감독자 클러스터를 업데이트하면 여기에 배포된 Tanzu Kubernetes Grid 서비스 클러스터의 롤링 업데이트가 트리거될 수 있습니다. vSphere 네임스페이스 업데이트를 수행하여 감독자 클러스터 업데이트의 내용을 참조하십시오.

Tanzu Kubernetes Grid 서비스 클러스터가 대상 감독자 클러스터 버전을 준수하지 않는 경우 감독자 클러스터를 업데이트하기 전에 하나 이상의 Tanzu Kubernetes Grid 서비스 클러스터를 업데이트해야 할 수 있습니다. 업데이트를 위한 Tanzu Kubernetes 클러스터 호환성 확인의 내용을 참조하십시오.

감독자 클러스터 업데이트 정보

감독자 클러스터에 대한 업데이트를 시작하면 시스템은 새 제어부 노드를 생성하고 이를 기존 제어부에 연결합니다. 이 업데이트 단계에서 시스템은 업데이트된 새 노드를 추가한 후 오래된 이전 노드를 제거하기 때문에 vSphere 인벤토리에 4개의 제어부 노드가 표시됩니다. 개체가 이전 제어부 노드 중 하나에서 새 노드로 마이그레이션되고 이전 제어부 노드는 제거됩니다. 이 프로세스는 모든 제어부 노드가 업데이트될 때까지 하나씩 반복됩니다. 제어부가 업데이트되면 유사한 롤링 업데이트 방식으로 작업자 노드가 업데이트됩니다. 작업자 노드는 ESXi 호스트이며 각 ESXi 호스트의 각 Spherelet 프로세스는 하나씩 업데이트됩니다.

다음 업데이트 중에서 선택할 수 있습니다.

- vSphere 네임스페이스를 업데이트합니다.
- VMware 버전 및 Kubernetes 버전을 포함한 모든 항목을 업데이트합니다.

vSphere 네임스페이스 업데이트 워크플로를 사용하여 감독자 클러스터가 실행 중인 Kubernetes 버전을 업데이트(예: Kubernetes 1.16.7에서 Kubernetes 1.17.4로)하고 감독자 클러스터 및 Tanzu Kubernetes Grid 서비스 클러스터를 지원하는 인프라를 업데이트할 수 있습니다. 이러한 유형의 업데이트는 더 빈번하며 Kubernetes 릴리스 속도와 보조를 맞추는 데 사용됩니다. vSphere 네임스페이스 업데이트 순서는 다음과 같습니다.

1 vCenter Server를 업그레이드합니다.

2 vSphere 네임스페이스 업데이트를 수행합니다(Kubernetes 업그레이드 포함).

vSphere 네임스페이스 업데이트를 수행하려면 vSphere 네임스페이스 업데이트를 수행하여 감독자 클러스터 업데이트 항목을 참조하십시오.

모든 vSphere with Tanzu 구성 요소를 업데이트하려면 모두 업데이트 워크플로를 사용합니다. 이런 유형의 업데이트는 주요 릴리스를 업데이트하는 경우에 필요합니다(예: NSX-T 3.X에서 4로, vSphere 7.X에서 8로). 이 업데이트 워크플로는 새로운 VMware 제품 릴리스가 있는 시기에 따라 드물게 발생합니다. 모두 업데이트 순서는 다음과 같습니다.

1 VMware 상호 운용성 매트릭스 https://interopmatrix.vmware.com/Interoperability에서 vCenter Server 및 NSX-T Data Center에 대한 호환성을 확인합니다. vSphere with Tanzu 기능은 vCenter Server와 함께 제공되는 WCP(워크로드 제어부) 소프트웨어를 통해 제공됩니다.

2 호환되는 경우 NSX-T Data Center를 업그레이드합니다.

3 vCenter Server를 업그레이드합니다.

4 vSphere Distributed Switch를 업그레이드합니다.

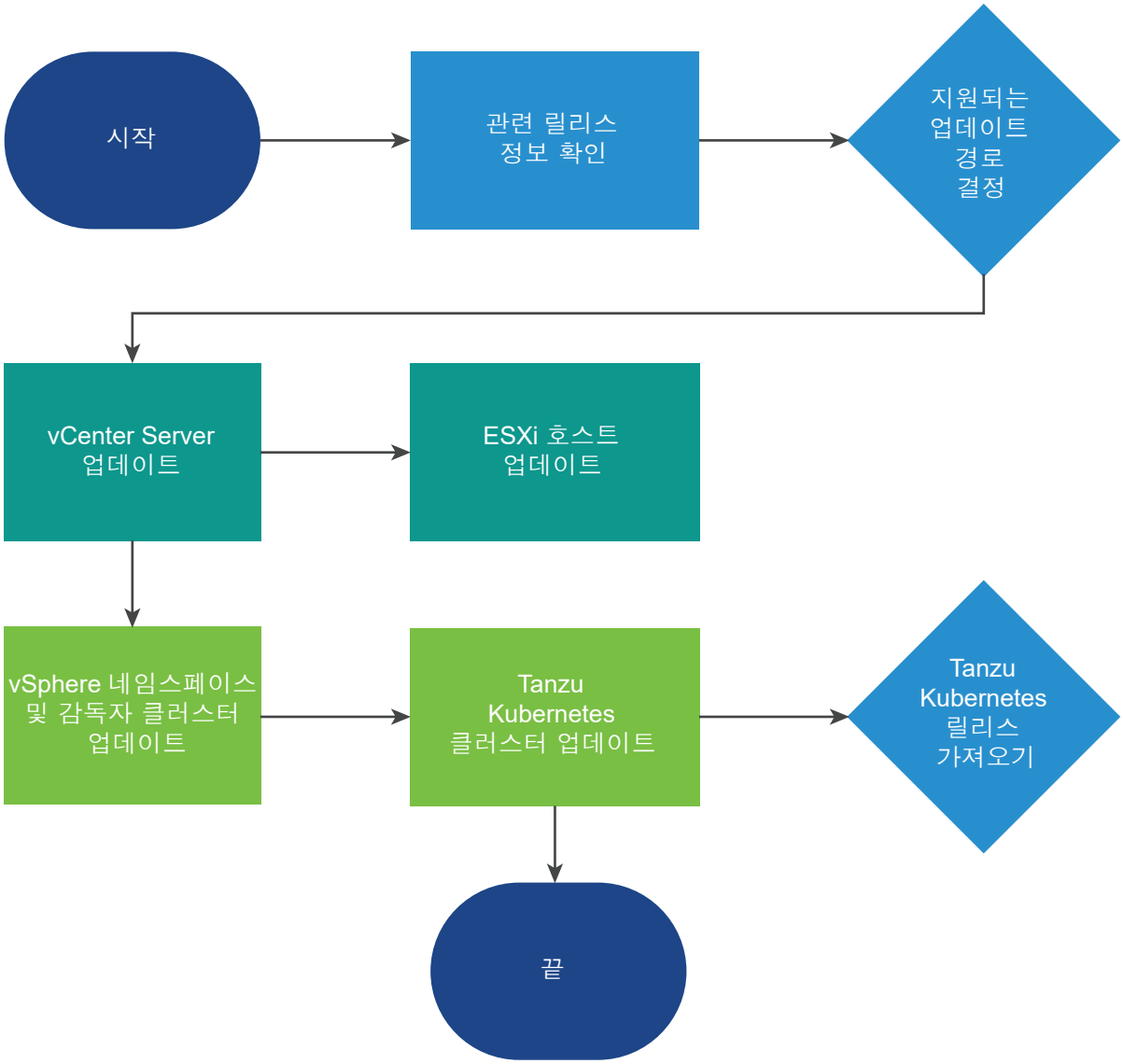
5 ESXi 호스트를 업그레이드합니다.

6 프로비저닝된 Tanzu Kubernetes Grid 서비스 클러스터가 대상 감독자 클러스터 버전과 호환되는지 확인합니다.

7 vSphere 네임스페이스를 업데이트합니다(감독자 클러스터 Kubernetes 버전 포함).

8 Tanzu Kubernetes Grid 서비스 클러스터를 업데이트합니다.

이 다이어그램은 vSphere with Tanzu 업데이트의 일반적인 워크플로를 보여줍니다.



Tanzu Kubernetes Grid 서비스 클러스터 업데이트 정보

감독자 클러스터를 업데이트하면 해당 감독자 클러스터에 배포된 Tanzu Kubernetes Grid 서비스 클러스터를 지원하는 인프라 구성 요소(예: Tanzu Kubernetes Grid 서비스)도 마찬가지로 업데이트됩니다. 각 인프라 업데이트에는 Tanzu Kubernetes Grid 서비스(CNI, CSI, CPI)를 지원하는 서비스에 대한 업데이트와 기존 Tanzu Kubernetes Grid 서비스 클러스터에 적용할 수 있는 제어부 및 작업자 노드에 대한 업데이트된 구성 설정이 포함될 수 있습니다. 구성이 호환성 요구 사항을 충족하는지 확인하기 위해 vSphere with Tanzu는 롤링 업데이트 중에 사전 확인을 수행하고 규정 준수를 적용합니다.

Tanzu Kubernetes Grid 서비스 클러스터의 롤링 업데이트를 수행하려면 클러스터 매니페스트를 업데이트합니다. Tanzu Kubernetes 클러스터 업데이트의 내용을 참조하십시오. 하지만 vSphere 네임스페이스 업데이트가 수행되면 시스템은 업데이트된 구성을 모든 Tanzu Kubernetes Grid 서비스 클러스터에 즉시 전파합니다. 이러한 업데이트는 Tanzu Kubernetes Grid 서비스 제어부 및 작업자 노드의 롤링 업데이트를 자동으로 트리거할 수 있습니다.

클러스터 노드를 교체하는 롤링 업데이트 프로세스는 Kubernetes 배포의 포드에 대한 롤링 업데이트와 비슷합니다. Tanzu Kubernetes Grid 서비스 클러스터의 롤링 업데이트를 수행하는 두 가지 개별 컨트롤러가 있습니다. 추가 기능 컨트롤러와 TanzuKubernetesCluster 컨트롤러입니다. 두 가지 컨트롤러 내에는 롤링 업데이트에 대한 세 가지 주요 단계인 추가 기능 업데이트, 제어부 업데이트 및 작업자 노드 업데이트 단계가 있습니다. 이러한 단계는 순서 대로 진행되며, 이전 단계가 충분히 진행될 때까지 단계를 시작하지 못하게 하는 사전 확인 기능이 있습니다. 이러한 단계가 불필요한 것으로 확인되면 건너뛸 수 있습니다. 예를 들어 업데이트가 작업자 노드에만 영향을 주기 때문에 추가 기능 또는 제어부 업데이트가 필요하지 않을 수 있습니다.

업데이트 프로세스 중에 시스템은 새 클러스터 노드를 추가하고 노드가 대상 Kubernetes 버전으로 온라인 상태가 될 때까지 기다립니다. 그런 다음, 시스템은 이전 노드를 삭제하도록 표시하고 다음 노드로 이동하여 프로세스를 반복합니다. 이전 노드는 모든 포드가 제거될 때까지 삭제되지 않습니다. 예를 들어, 노드가 완전히 드레이닝되는 것을 방지하는 PodDisruptionBudgets를 사용하여 포드를 정의하면 노드가 차단되지만 해당 포드를 제거할 때까지 제거되지 않습니다. 시스템은 모든 제어부 노드를 먼저 업그레이드한 후 작업자 노드를 업그레이드합니다. 업데이트하는 동안 Tanzu Kubernetes Grid 서비스 클러스터 상태가 "업데이트 중"으로 변경됩니다. 롤링 업데이트 프로세스가 완료되면 Tanzu Kubernetes Grid 서비스 클러스터 상태가 "실행 중"으로 변경됩니다.

복제 컨트롤러에서 관리되지 않는 Tanzu Kubernetes Grid 서비스 클러스터에서 실행되는 포드는 Tanzu Kubernetes Grid 서비스 클러스터 업데이트 동안 작업자 노드 드레이닝의 일부로 Kubernetes 버전 업그레이드 중에 삭제됩니다. 이는 클러스터 업데이트가 vSphere 네임스페이스 업데이트를 통해 수동으로 또는 자동으로 트리거되는 경우에 적용됩니다. 복제 컨트롤러에서 관리되지 않는 포드에는 배포 또는 ReplicaSet 규격의 일부로 생성되지 않는 포드가 포함됩니다. 자세한 내용은 Kubernetes 설명서에서 포드 수명 주기: 포드 수명 항목을 참조하십시오.

네트워크 토폴로지 업그레이드

vSphere with Tanzu 버전 7.0 업데이트 1c를 설치하거나 감독자 클러스터를 버전 7.0 업데이트 1에서 버전 7.0 업데이트 1c로 업그레이드하는 경우 NCP(NSX Container Plug-in)를 업그레이드합니다. 그러면 감독자 클러스터, 네임스페이스 및 Tanzu Kubernetes 클러스터의 네트워킹 토폴로지가 마이그레이션됩니다. 업그레이드 후에 네트워킹 토폴로지는 단일 Tier-1 게이트웨이 토폴로지에서 감독자 클러스터 내의 각 네임스페이스에 대해 Tier-1 게이트웨이가 있는 토폴로지로 업그레이드됩니다.

업그레이드하는 동안 NCP는 새 토폴로지를 지원하도록 NSX-T Data Center 리소스를 구성합니다. NCP는 계층 4 및 계층 7 로드 밸런싱 서비스가 적은 네임스페이스에 대해 공유 네트워크 인프라를 제공합니다. 이렇게 하면 NSX의 리소스가 줄어들고 Tanzu Kubernetes 클러스터에서 더 많이 사용할 수 있습니다.

시스템 네임스페이스는 감독자 클러스터 및 Tanzu Kubernetes 클러스터가 작동하는 데 필수적인 핵심 구성 요소에서 사용되는 네임스페이스입니다. Tier-1 게이트웨이, 로드 밸런서 및 SNAT IP를 포함하는 공유 네트워크 리소스는 시스템 네임스페이스에 그룹화됩니다.

NCP는 기본적으로 시스템 네임스페이스에 대해 하나의 공유 Tier-1 게이트웨이를 생성하고 각 네임스페이스에 대해 Tier-1 게이트웨이와 로드 밸런서를 생성합니다. Tier-1 게이트웨이는 Tier-0 게이트웨이 및 기본 세그먼트에 연결됩니다.

NSX-T 로드 밸런서는 가상 서버의 형태로 로드 밸런싱 서비스를 제공합니다.

마이그레이션 후 네트워킹 토폴로지에는 다음과 같은 특성이 있습니다.

- 각 vSphere 네임스페이스에는 별도의 네트워크 및 네임스페이스 내의 애플리케이션이 공유하는 네트워킹 리소스 집합(예: Tier-1 게이트웨이, 로드 밸런서 서비스, SNAT IP 주소)이 있습니다.
- 동일한 네임스페이스에 있는 vSphere 포드, 일반 VM 또는 Tanzu Kubernetes 클러스터에서 실행되는 워크로드는 North-South 연결에 대해 동일한 SNAT IP를 공유합니다.
- vSphere 포드 또는 Tanzu Kubernetes 클러스터에서 실행되는 워크로드에는 기본 방화벽에 의해 구현되는 것과 동일한 격리 규칙이 있습니다.
- 각 Kubernetes 네임스페이스에 대해 별도의 SNAT IP가 필요하지 않습니다. 네임스페이스 간의 East-West 연결은 SNAT가 아닙니다.

실행할 수 있는 최대 네임스페이스 수는 Edge 노드 크기(중간, 대형 또는 초대형) 및 NSX Edge 클러스터에 있는 Edge 노드 수에 따라 달라 집니다. 실행할 수 있는 네임스페이스의 수는 Edge 노드 수의 20배보다 작습니다. 예를 들어 NSX Edge 클러스터에 크기가 대형인 Edge 노드 10개가 있는 경우 생성할 수 있는 감독자 네임스페이스의 최대 수는 199입니다.

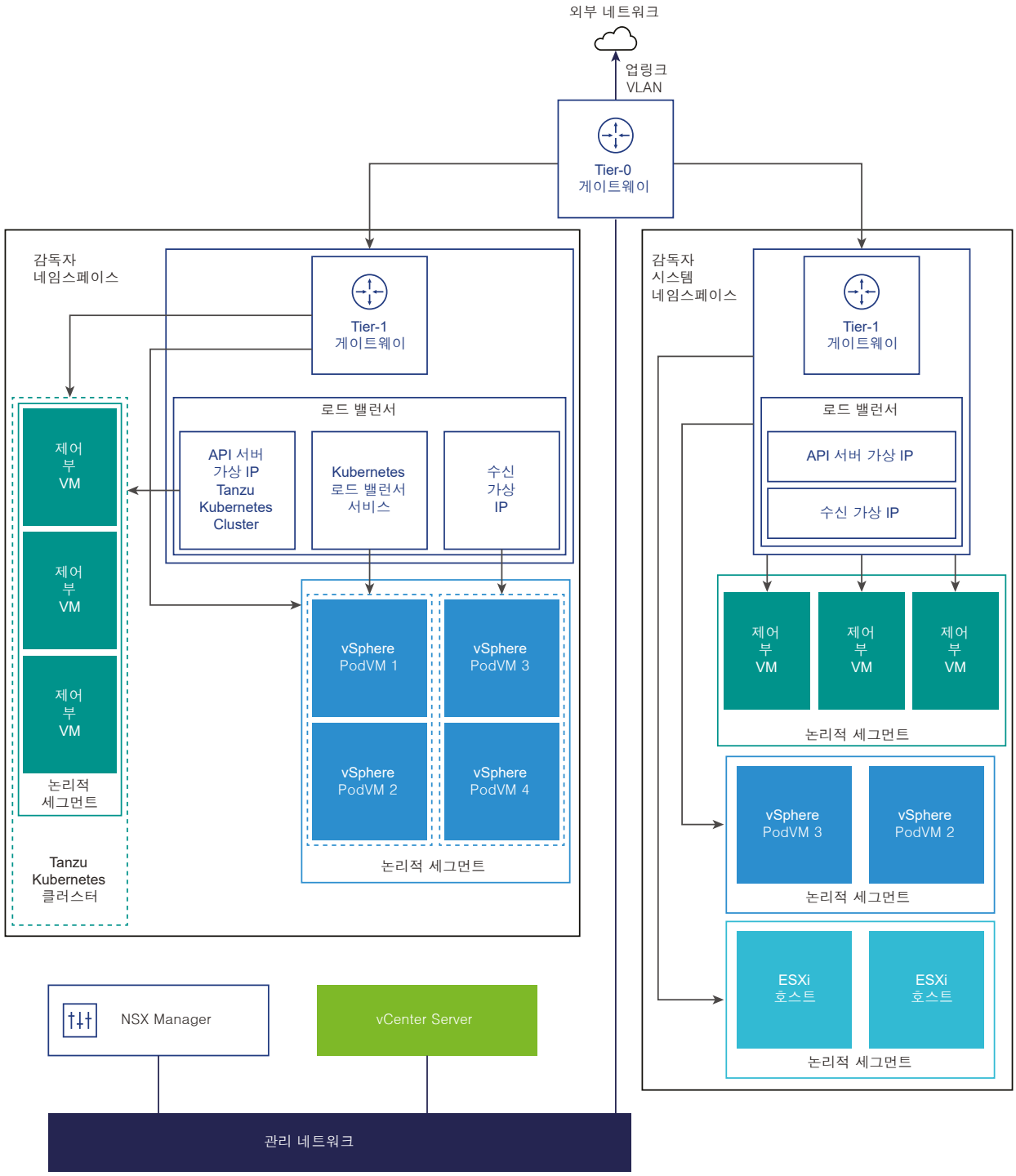
Edge 노드 크기에 대한 자세한 내용은 "NSX-T Data Center 설치 가이드" 를 참조하십시오.

감독자 클러스터 네트워킹

감독자 클러스터는 공유 Tier-1 게이트웨이 내에 별도의 세그먼트가 있습니다. 각 Tanzu Kubernetes 클러스터에 대해 세그먼트는 네임스페이스의 Tier-1 게이트웨이 내에 정의됩니다.

동일한 네임스페이스 내에 있는 vSphere 포드 및 Tanzu Kubernetes 클러스터를 포함하는 워크로드는 North-South 연결을 위해 SNAT IP를 공유합니다. 네임스페이스 간의 East-West 연결은 SNAT가 아닙니다.

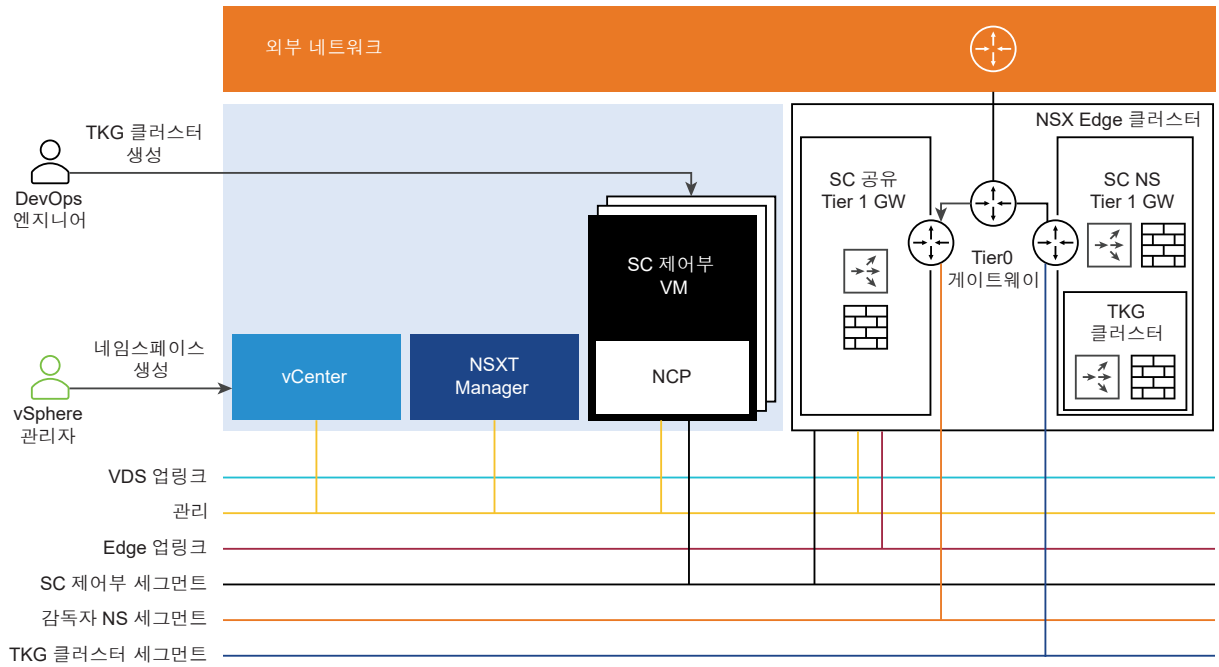
그림 17-1. 감독자 클러스터 네트워킹



Tanzu Kubernetes 클러스터 네트워킹

감독자 클러스터 업그레이드 후 DevOps 엔지니어가 감독자 네임스페이스에서 첫 번째 Tanzu Kubernetes 클러스터를 프로비저닝하면 클러스터는 네임스페이스와 동일한 Tier-1 게이트웨이와 로드 밸런서를 공유합니다. 이 네임스페이스에 프로비저닝된 각 Tanzu Kubernetes 클러스터에 대해 해당 클러스터에 대한 세그먼트가 생성되고 해당 감독자 네임스페이스의 공유 Tier-1 게이트웨이에 연결됩니다.

Tanzu Kubernetes Grid 서비스에서 Tanzu Kubernetes 클러스터가 프로비저닝되면 Kubernetes API에 대해 계층 4 로드 밸런싱을 제공하는 단일 가상 서버가 생성됩니다. 이 가상 서버는 네임스페이스를 사용하여 공유 로드 밸런서에 호스팅되며 kubectl 트래픽을 제어부로 라우팅하는 작업을 담당합니다. 또한 클러스터에서 리소스가 제공되는 각 Kubernetes 서비스 로드 밸런서에 대해 해당 서비스에 계층 4 로드 밸런싱을 제공하는 가상 서버가 생성됩니다.



NSX-T 네트워크 토폴로지 업그레이드

네트워크 토폴로지를 업그레이드하려면 NSX-T Data Center, vCenter Server 및 모든 vSphere with Tanzu 구성 요소를 업그레이드합니다.

NSX-T Data Center 버전을 업그레이드할 수 있습니다. 업그레이드를 수행하면 NSX-T Data Center 구성 요소(데이터부, 제어부 및 관리부 포함)가 최소한의 시스템 다운타임으로 업그레이드됩니다.

NSX-T Data Center 구성 요소 업그레이드에 대한 단계별 정보는 NSX-T Data Center 업그레이드 가이드의 내용을 참조하십시오.

참고 지원되는 버전은 vSphere with Tanzu 릴리스 정보를 참조하십시오.

사전 요구 사항

- 환경이 시스템 요구 사항을 충족하는지 확인합니다. 요구 사항에 대한 자세한 내용은 NSX-T Data Center를 사용하여 vSphere with Tanzu를 설정하기 위한 시스템 요구 사항 항목을 참조하십시오.
- vSphere with Tanzu와 관련된 구성 제한은 VMware 구성 최대값 도구에서 vSphere 구성 제한을 참조하십시오.

절차

1 NSX-T Data Center를 NSX-T Data Center 3.0.x에서 NSX-T Data Center 3.1로 업그레이드합니다.

2 vCenter Server를 업그레이드합니다.

3 ESXi 호스트를 업그레이드합니다.

4 감독자 네임스페이스를 업데이트합니다.

업데이트를 수행하려면 vSphere 네임스페이스 업데이트를 수행하여 감독자 클러스터 업데이트의 내용을 참조하십시오.

vSphere Distributed Switch 업그레이드

vSphere Distributed Switch 버전 7.0을 이후 버전으로 업그레이드할 수 있습니다. Distributed Switch를 업그레이드하면 이 스위치에 연결된 호스트와 가상 시스템에서 잠시 동안 다운타임이 발생합니다.

사전 요구 사항

- vCenter Server를 버전 7.0으로 업그레이드합니다.
- Distributed Switch에 연결된 모든 호스트를 ESXi 7.0으로 업그레이드합니다.

절차

1 vSphere Client 홈 메뉴에서 **호스트 및 클러스터**를 선택합니다.

2 **네트워킹**을 선택하고 분산 스위치로 이동합니다.

예, **DSwitch**.

vSphere Client UI에 업그레이드할 수 있는 버전이 나열됩니다.

3 **작업** 메뉴에서 **업그레이드 > Distributed Switch 업그레이드**를 선택합니다.

4 스위치를 업그레이드할 vSphere Distributed Switch 버전을 선택하고 다음을 클릭합니다.

5 호스트 호환성을 검토하고 **다음**을 클릭합니다.

6 업그레이드 구성을 완료하고 **마침**을 클릭합니다.

vSphere 네임스페이스 업데이트를 수행하여 감독자 클러스터 업데이트

감독자 클러스터가 실행되는 Kubernetes 버전 및 Tanzu Kubernetes 클러스터를 지원하는 인프라를 포함하고 Tanzu Kubernetes Grid 서비스를 포함하여 감독자 클러스터를 하나 이상 업데이트하려면 vSphere 네임스페이스 업데이트를 수행합니다.

vSphere with Tanzu에 대한 버전 엔티티가 있습니다. 버전 엔티티는 `v1.19.1+vmware.2-vsc0.0.8-17610687` 형태의 의미 체계 버전 문자열이며, 접두사는 Kubernetes 버전(`v1.19.1`)이고 접미사는 vSphere 네임스페이스 버전(`vsc0.0.8-17610687`)입니다.

표에는 사용 가능한 감독자 클러스터 릴리스가 나와 있습니다.

버전	설명
`v1.19.1+vmware.2-vsc0.0.8-17610687`	최신 감독자 클러스터 릴리스, vSphere 7.0 업데이트 2 지원.
`v1.18.2-vsc0.0.5-16762486`	vSphere 7.0 업데이트 1 지원.
`v1.17.4-vsc0.0.5-16762486`	Antrea CNI를 지원하는 Tanzu Kubernetes Grid 서비스가 포함된 최소 감독자 클러스터 릴리스입니다.
`v1.16.7-vsc0.0.5-16762486`	이 릴리스를 실행하는 경우에는 최소한 1.17로 업데이트해야 합니다.

사전 요구 사항

vSphere 네임스페이스 업데이트를 수행하기 전에 릴리스 정보를 읽어 보십시오.

필요한 바이너리를 설치하려면 vCenter Server Appliance 및 ESXi 호스트를 지원되는 버전으로 업그레이드합니다. vCenter Server 설명서에서 vCenter Server Appliance 업그레이드를 참조하십시오.

참고 vSphere 네임스페이스 업데이트를 수행할 때는 프로비저닝된 모든 Tanzu Kubernetes 클러스터의 상태가 실행 중이어야 합니다. Tanzu Kubernetes의 상태가 생성 중 또는 삭제 중이면 감독자 클러스터를 업데이트하기 전에 프로세스가 완료될 때까지 기다리십시오. 그러지 않으면 성공하지 못할 수 있습니다.

참고 감독자 클러스터를 업데이트하면 여기에 배포된 Tanzu Kubernetes 클러스터의 롤링 업데이트가 트리거될 수 있습니다. Tanzu Kubernetes 클러스터 업데이트의 내용을 참조하십시오.

절차

1 vCenter Server에 vSphere 관리자로 로그인합니다.

2 **메뉴 > 워크로드 관리**를 선택합니다.

3 **네임스페이스 > 업데이트** 탭을 선택합니다.

4 업데이트하려는 **사용 가능한 버전**을 선택합니다.

예를 들어, `v1.18.2-vsc0.0.5-16762486` 버전을 선택합니다.

참고 증분 업데이트를 해야 합니다. 업데이트를 건너뛰지 마십시오. 예를 들어, 1.16에서1.18로 업데이트하지 마십시오. 1.16, 1.17, 1.18 순으로 업데이트해야 합니다.

5 업데이트를 적용할 감독자 클러스터를 하나 이상 선택합니다.

6 업데이트를 시작하려면 **업데이트 적용**을 클릭합니다.

7 **최근 작업** 창을 사용하여 업데이트 상태를 모니터링합니다.

감독자 클러스터 자동 업그레이드

vCenter Server Appliance를 업그레이드할 때 감독자 클러스터를 자동으로 업그레이드할 수 있습니다.

vSphere with Tanzu 구성 요소에는 vCenter Server의 구성 요소, Kubernetes 구성 요소, ESXi 구성 요소가 포함됩니다. vCenter Server를 업그레이드하는 경우 vCenter Server의 vSphere with Tanzu 구성 요소만 업그레이드됩니다. Kubernetes 구성 요소 및 ESXi 구성 요소는 수동으로 업그레이드해야 합니다.

자동 업그레이드 기능을 사용하면 vCenter Server를 업그레이드할 때 감독자 클러스터를 자동으로 업그레이드할 수 있습니다. vCenter Server 릴리스 목록은 다음 KB 문서를 참조하십시오. https://kb.vmware.com/s/article/2143838.

vCenter Server를 업그레이드할 때 감독자 클러스터 버전과 vCenter Server 버전 간의 호환성을 확인하기 위해 사전 검사가 실행됩니다. 다음과 같은 시나리오에서는 주의가 표시됩니다.

- 대상 vCenter Server에 감독자 클러스터 버전을 하나 더 낮게 만들 수 있는 Kubernetes 버전이 있습니다. 이 시나리오에서 vCenter Server 업그레이드를 진행하면 클러스터가 자동으로 업그레이드됩니다.

  예를 들어 감독자 클러스터 버전은 1.16이고, 대상 vCenter Server 버전은 1.17, 1.18 또는 1.19입니다.

- 대상 vCenter Server에 감독자 클러스터 버전을 둘 이상 낮게 만드는 Kubernetes 버전이 있습니다. 이 시나리오에서는 vCenter Server가 업그레이드되지 않습니다.

  예를 들어 감독자 클러스터 버전은 1.15이고, 대상 vCenter Server 버전은 1.17, 1.18 또는 1.19입니다.

- 감독자 클러스터 라이센스가 만료되었습니다. 이 시나리오에서 vCenter Server 업그레이드를 진행하면 만료된 라이센스가 있는 감독자 클러스터를 사용할 수 없고 복구할 수 없게 됩니다.

kubectl용 vSphere 플러그인 업데이트

vSphere 네임스페이스 업데이트를 수행하고 감독자 클러스터를 업그레이드했으면 kubectl용 vSphere 플러그인을 업데이트합니다.

가장 최신 버전의 kubectl용 vSphere 플러그인이 *TANZU-KUBERNETES-CLUSTER-NAME*-ca이라는 Kubernetes 암호에 Tanzu Kubernetes 클러스터 루트 CA 인증서를 다운로드하여 설치합니다. 플러그인은 이 인증서를 사용하여 해당 클러스터의 CA(인증 기관) 데이터스토어에 CA 정보를 채웁니다.

kubectl용 vSphere 플러그인을 다운로드하여 설치하려면 vSphere에 대한 Kubernetes CLI 도구 다운로드 및 설치 항목을 참조하십시오. *TANZU-KUBERNETES-CLUSTER-NAME*-ca 암호에 대한 자세한 내용은 Tanzu Kubernetes 클러스터 암호 얻기 항목을 참조하십시오.

업데이트를 위한 Tanzu Kubernetes 클러스터 호환성 확인

Tanzu Kubernetes 릴리스는 Tanzu Kubernetes Grid 서비스에 의해 프로비저닝된 Tanzu Kubernetes 클러스터에 대해 VMware에서 서명 및 지원하는 Kubernetes 소프트웨어 배포를 제공합니다.

Tanzu Kubernetes 릴리스의 릴리스 정보

모든 Tanzu Kubernetes 릴리스 목록은 릴리스 정보를 참조하십시오.

업데이트를 위한 Tanzu Kubernetes 클러스터 호환성 확인

vCenter Server 및 감독자 클러스터를 포함한 vSphere with Tanzu 인프라를 업그레이드하기 전에 기술 자료 문서 https://kb.vmware.com/s/article/82592에서 Tanzu Kubernetes 클러스터가 업그레이드와 호환되지 않는지 검사하는 방법에 대한 지침을 참조하십시오. Tanzu Kubernetes 클러스터가 대상 인프라와 호환되지 않으면 시스템 업그레이드를 진행하기 전에 클러스터를 업그레이드합니다.

KubectlI을 사용하여 사용 가능한 Tanzu Kubernetes 릴리스 나열

다음 명령을 사용하여 Tanzu Kubernetes 릴리스를 나열하고 각 릴리스의 호환성 및 업데이트 가능성을 볼 수 있습니다.

```
kubectl get tanzukubernetesreleases
```

COMPATIBLE 열은 Tanzu Kubernetes 릴리스가 현재 감독자 클러스터와 호환되는지 여부를 나타냅니다. UPDATES AVAILABLE 열은 사용 가능한 Kubernetes 업그레이드가 있는지 그리고 사용이 권장되는 다음 Tanzu Kubernetes 릴리스가 있는지 여부를 나타냅니다.

```
NAME                                VERSION                          READY   COMPATIBLE
CREATED   UPDATES AVAILABLE
v1.18.15---vmware.1-tkg.1.600e412   1.18.15+vmware.1-tkg.1.600e412   True    True
21h       [1.19.7+vmware.1-tkg.1.fc82c41]
v1.19.7---vmware.1-tkg.1.fc82c41    1.19.7+vmware.1-tkg.1.fc82c41    True    True
21h       [1.20.2+vmware.1-tkg.1.1d4f79a]
v1.20.2---vmware.1-tkg.1.1d4f79a    1.20.2+vmware.1-tkg.1.1d4f79a    True    True         21h
```

kubectl get tkc <tkgs-cluster-name>을 사용하여 동일한 정보를 사용할 수도 있습니다.

Tanzu Kubernetes 클러스터 업데이트

Tanzu Kubernetes 릴리스, 가상 시스템 클래스 또는 스토리지 클래스를 업데이트하여 Kubernetes 버전을 포함한 Tanzu Kubernetes 클러스터의 롤링 업데이트를 시작할 수 있습니다.

Tanzu Kubernetes 클러스터 업데이트 준비 검사 목록

Tanzu Kubernetes 클러스터 업데이트를 수행하기 전에 다음 사전 요구 사항 작업 목록을 완료합니다.

단계	작업
1	vSphere with Tanzu 릴리스 정보를 읽습니다.
2	Tanzu Kubernetes 릴리스 릴리스 정보를 읽습니다.
3	vSphere with Tanzu vSphere with Tanzu 업데이트 정보를 검토합니다.
4	대상 업그레이드 버전과의 Tanzu Kubernetes 업데이트를 위한 Tanzu Kubernetes 클러스터 호환성 확인을 확인합니다.
5	Tanzu Kubernetes Grid 서비스 API 대상 버전(예: TKGS v1alpha2 API를 사용하여 Tanzu Kubernetes 클러스터 프로비저닝) 및 현재 버전(예: Tanzu Kubernetes Grid 서비스 v1alpha1 API를 사용하여 Tanzu Kubernetes 클러스터 프로비저닝)의 기능을 검토합니다. (아래 중요 참고 사항을 참조하십시오.)
6	프로비저닝된 모든 Tanzu Kubernetes 클러스터가 Tanzu Kubernetes 클러스터 수명 주기 상태 보기 상태인지 확인합니다.
7	감독자 클러스터 및 Tanzu Kubernetes Grid 서비스를 업그레이드하는 vSphere 네임스페이스 vSphere 네임스페이스 업데이트를 수행하여 감독자 클러스터 업데이트를 수행합니다.
8	Tanzu Kubernetes 클러스터의 Tanzu Kubernetes 클러스터의 롤링 업데이트 시작 옵션을 검토합니다.
9	클러스터 매니페스트 업데이트를 위해 클러스터 매니페스트 편집 방법을 검토합니다.

중요 vSphere with Tanzu 버전 7 업데이트 3, 특히 감독자 클러스터 버전 `v1.21.0+vmware.wcp.2`에는 Tanzu Kubernetes Grid 서비스 v1alpha2 API에 대한 자동 업그레이드가 포함되어 있습니다. Tanzu Kubernetes 클러스터 규격의 일부 필드는 더 이상 사용되지 않으며 Kubernetes 버전을 업그레이드하기 전에 클러스터 매니페스트를 수동으로 편집해야 할 수 있습니다. 클러스터 규격이 TKGS v1alpha2 API로 변환된 후 Tanzu Kubernetes 릴리스 업데이트의 내용을 참조하십시오.

Tanzu Kubernetes 클러스터의 롤링 업데이트 시작

롤링 업데이트는 `TanzuKubernetesCluster` 규격에 대해 다음 중 하나 이상을 수정하여 시작합니다.

- Tanzu Kubernetes 릴리스 버전을 업그레이드하여 Tanzu Kubernetes 클러스터 업데이트
- VirtualMachineClass를 변경하여 Tanzu Kubernetes 클러스터 업데이트
- 스토리지 클래스를 변경하여 Tanzu Kubernetes 클러스터 업데이트

참고 이러한 방식은 롤링 업데이트를 시작하는 가장 일반적인 방법이지만 유일한 방법은 아닙니다. 구성 요소를 변경하여 롤링 업데이트를 시작할 수도 있습니다. 예를 들어, 배포 버전에 해당하는 `VirtualMachineImage`을 교체하거나 이름을 변경하면, 시스템이 새 이미지에서 실행되는 모든 노드를 가져오려고 하면서 롤링 업데이트가 시작됩니다. 또한, 감독자 클러스터를 업데이트하면 여기에 배포된 Tanzu Kubernetes 클러스터의 롤링 업데이트가 트리거될 수 있습니다. 예를 들어, `vmware-system-tkg-controller-manager`가 업데이트되면 새 값을 시스템이 매니페스트 생성기에 도입하고 이 값을 배포하기 위해 컨트롤러가 롤링 업데이트를 시작합니다.

클러스터 매니페스트 편집 방법

클러스터를 업데이트하려면 클러스터 매니페스트를 업데이트해야 합니다. 이 작업은 다음과 같이 다양한 방법으로 수행할 수 있습니다.

- `kubectl edit tanzukubernetescluster/CLUSTER-NAME` 명령을 사용합니다. 이 명령은 KUBE_EDITOR 또는 EDITOR 환경 변수로 정의된 전체 클러스터 매니페스트를 텍스트 편집기에서 엽니다. 파일을 저장하면 클러스터가 변경 사항으로 업데이트됩니다. `kubectl edit` 명령에 대한 자세한 내용은 Kubernetes 설명서에서edit 명령을 참조하십시오. `kubectl edit` 방식을 사용하려면 다음 항목을 참조하십시오.
  - Tanzu Kubernetes 릴리스 버전을 업그레이드하여 Tanzu Kubernetes 클러스터 업데이트
  - VirtualMachineClass를 변경하여 Tanzu Kubernetes 클러스터 업데이트
  - 스토리지 클래스를 변경하여 Tanzu Kubernetes 클러스터 업데이트
- `kubectl patch` 명령을 사용합니다. 이 명령은 클러스터에 대한 "인플레이스" 업데이트를 수행합니다. 이 명령의 목적은 Kubernetes 버전을 업그레이드하는 방법을 제공하는 것이며 그 접근 방법이 여기에 설명되어 있습니다. `kubectl patch` 명령에 대한 자세한 내용은 Kubernetes 설명서에서 kubectl patch를 사용하여 API 개체 인플레이스 업데이트를 참조하십시오. `kubectl patch` 방식을 사용하려면 다음 항목을 참조하십시오.
  - 패치 방법을 사용하여 Tanzu Kubernetes 클러스터 업데이트
- 수동으로 업데이트하는 로컬 YAML 파일과 함께 `kubectl apply` 명령을 사용합니다. 이 접근 방식은 TKGS v1alpha2 API를 사용하여 Tanzu Kubernetes 클러스터를 프로비저닝하는 워크플로 방식과 유사하다는 장점이 있지만 현재 클러스터 YAML에 대한 액세스 권한이 없는 경우 파괴적일 수 있기 때문에 권장되지 않습니다.

Tanzu Kubernetes 릴리스 버전을 업그레이드하여 Tanzu Kubernetes 클러스터 업데이트

Tanzu Kubernetes 릴리스 버전을 업그레이드하여 Tanzu Kubernetes 클러스터를 업데이트합니다.

Tanzu Kubernetes 릴리스 버전을 업그레이드하여 Tanzu Kubernetes 클러스터의 롤링 업데이트를 시작할 수 있습니다. 이 작업을 수행하는 방법은 사용 중인 Tanzu Kubernetes Grid 서비스 API의 버전에 따라 달라집니다.

TKGS API 버전	버전 업데이트 방법
v1alpha2 API	클러스터 매니페스트의 `spec.topology.controlPlane.tkr.refernece.name` 및 `spec.topology.nodePools[*].tkr.reference.name` 속성에서 TKR NAME 문자열을 업데이트합니다. 클러스터 규격이 TKGS v1alpha2 API로 변환된 후 Tanzu Kubernetes 릴리스 업데이트의 내용을 참조하십시오.
v1alpha1 API	클러스터 매니페스트의 `spec.distribution.version` 및 `spec.distribution.fullVersion` 속성에서 DISTRIBUTION 버전을 업데이트합니다. 아래를 참조하십시오.

사전 요구 사항

Tanzu Kubernetes 클러스터 업데이트를 위한 사전 요구 사항이 완료되었는지 확인합니다. Tanzu Kubernetes 클러스터 업데이트의 내용을 참조하십시오.

이 작업은 `kubectl edit tanzukubernetescluster/CLUSTER-NAME` 명령을 사용하여 클러스터 매니페스트를 업데이트합니다. kubectl edit 명령은 KUBE_EDITOR 또는 EDITOR 환경 변수로 정의된 텍스트 편집기에서 클러스터 매니페스트를 엽니다. 파일을 저장하면 클러스터가 변경 사항으로 업데이트됩니다. kubectl용 기본 텍스트 편집기 지정의 내용을 참조하십시오.

절차

1 감독자 클러스터로 인증합니다. vCenter Single Sign-On 사용자로 감독자 클러스터에 연결의 내용을 참조하십시오.

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

2 대상 Tanzu Kubernetes 클러스터가 프로비저닝된 vSphere 네임스페이스로 컨텍스트를 전환합니다.

```
kubectl config use-context SUPERVISOR-NAMESPACE
```

3 대상 Tanzu Kubernetes 클러스터 및 버전을 가져옵니다.

```
kubectl get tanzukubernetescluster
```

예를 들어 TKGS v1alpha2 API를 사용한 결과는 다음과 같습니다.

```
kubectl get tanzukubernetescluster
NAMESPACE        NAME            CONTROL PLANE   WORKER   TKR
NAME                             AGE    READY   TKR COMPATIBLE   UPDATES AVAILABLE
tkgs-cluster-1   test-cluster    3               3        v1.21.2---vmware.1-
tkg.1.ee25d55     38h    True    True             [1.21.2+vmware.1-tkg.1.ee25d55]
```

예를 들어 TKGS v1alpha1 API를 사용한 결과는 다음과 같습니다.

```
kubectl get tanzukubernetescluster
NAME             CONTROL PLANE   WORKER   DISTRIBUTION                      AGE   PHASE
tkgs-cluster-1   3               3        v1.19.16+vmware.1-tkg.1.df910e2   19h   running
```

4 사용 가능한 Tanzu Kubernetes 릴리스를 나열합니다.

```
kubectl get tanzukubernetesreleases
```

5 다음 명령을 실행하여 클러스터 매니페스트를 편집합니다.

```
kubectl edit tanzukubernetescluster/CLUSTER-NAME
```

6 Tanzu Kubernetes 릴리스를 업데이트하여 매니페스트를 편집합니다. 이 작업을 수행하는 방법은 사용 중인 TKGS API의 버전에 따라 달라집니다.

TKGS v1alpha2 API를 사용하는 경우 TKR NAME 문자열을 업데이트합니다. 클러스터 규격이 TKGS v1alpha2 API로 변환된 후 Tanzu Kubernetes 릴리스 업데이트의 내용을 참조하십시오.

TKGS v1alpha1 API를 사용하는 경우 예를 들면 다음에서:

```
spec:
  distribution:
    fullVersion: v1.19.16+vmware.1-tkg.1.df910e2
    version: v1.19.16
```

다음으로 업데이트합니다.

```
spec:
  distribution:
    fullVersion: null
    version: v1.20.12
```

참고 TKGS v1alpha1 API를 사용하는 경우, 정규화된 버전을 지정하거나 버전 바로 가기를 사용할 수 있습니다. 예를 들어, `version: v1.20.12`는 해당 패치 버전과 일치하는 최신 이미지로 확인되며 `version: v1.20`은 일치하는 최신 패치 버전으로 확인됩니다. 확인된 버전은 클러스터 매니페스트에서 `fullVersion`으로 표시됩니다. 바로 가기를 사용하여 버전 업그레이드를 수행하려면 검색 중에 잠재적인 버전 불일치를 방지하기 위해 `fullVersion`을 설정 해제하거나 null로 설정해야 합니다.

7 변경 내용을 매니페스트 파일에 적용합니다.

파일을 저장하면 kubectl이 변경 내용을 클러스터에 적용합니다. 백그라운드에서 감독자 클러스터의 가상 시스템 서비스는 새 작업자 노드를 프로비저닝합니다.

8 kubectl이 매니페스트 편집이 기록되었다고 보고하는지 확인합니다.

```
kubectl edit tanzukubernetescluster/tkgs-cluster-1
tanzukubernetescluster.run.tanzu.vmware.com/tkgs-cluster-1 edited
```

참고 오류가 발생하거나 kubectl이 클러스터 매니페스트가 편집되었다고 보고하지 않는 경우에는 KUBE_EDITOR 환경 변수를 사용하여 기본 텍스트 편집기를 올바르게 구성했는지 확인합니다. kubectl용 기본 텍스트 편집기 지정의 내용을 참조하십시오.

9 클러스터가 업데이트되고 있는지 확인합니다.

```
kubectl get tanzukubernetescluster
NAME             CONTROL PLANE   WORKER   DISTRIBUTION                      AGE   PHASE
tkgs-cluster-1   3               3        v1.20.12+vmware.1-tkg.1.b9a42f3   21h   updating
```

10 클러스터가 업데이트되었는지 확인합니다.

```
kubectl get tanzukubernetescluster
NAME             CONTROL PLANE   WORKER   DISTRIBUTION                      AGE   PHASE
tkgs-cluster-1   3               3        v1.20.12+vmware.1-tkg.1.b9a42f3   22h   running
```

VirtualMachineClass를 변경하여 Tanzu Kubernetes 클러스터 업데이트

클러스터 노드를 호스팅하는 데 사용되는 가상 시스템 클래스를 변경하여 Tanzu Kubernetes 클러스터를 업데이트할 수 있습니다.

Tanzu Kubernetes Grid 서비스는 `VirtualMachineClass` 정의를 변경하여 클러스터 업데이트를 지원합니다. 이렇게 하면 서비스는 새 클래스를 사용하여 새 노드를 롤아웃하고 이전 노드를 스핀 다운합니다. Tanzu Kubernetes Grid 서비스 클러스터 업데이트 정보의 내용을 참조하십시오.

참고 `VirtualMachineClass`는 Tanzu Kubernetes 클러스터가 프로비저닝된 vSphere 네임스페이스에 바인딩되어야 합니다. Tanzu Kubernetes 클러스터에 대한 가상 시스템 클래스의 내용을 참조하십시오.

사전 요구 사항

이 작업은 `kubectl edit tanzukubernetescluster/CLUSTER-NAME` 명령을 사용하여 클러스터 매니페스트를 업데이트합니다. kubectl edit 명령은 KUBE_EDITOR 또는 EDITOR 환경 변수로 정의된 텍스트 편집기에서 클러스터 매니페스트를 엽니다. 파일을 저장하면 클러스터가 변경 사항으로 업데이트됩니다. kubectl용 기본 텍스트 편집기 지정의 내용을 참조하십시오.

절차

1 감독자 클러스터로 인증합니다. vCenter Single Sign-On 사용자로 감독자 클러스터에 연결의 내용을 참조하십시오.

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

2 대상 Tanzu Kubernetes 클러스터가 프로비저닝된 vSphere 네임스페이스로 컨텍스트를 전환합니다.

```
kubectl config use-context SUPERVISOR-NAMESPACE
```

3 대상 Tanzu Kubernetes 클러스터를 설명하고 VM 클래스를 확인합니다.

```
kubectl describe tanzukubernetescluster CLUSTER-NAME
```

예를 들어 이 클러스터는 best-effort-medium VM 클래스를 사용합니다.

```
Spec:
  ...
  Topology:
    Control Plane:
      Class:          best-effort-medium
```

```
      ...
    Workers:
      Class:          best-effort-medium
      ...
```

4 네임스페이스에서 사용 가능한 VM 클래스를 나열하고 설명합니다.

```
kubectl get virtualmachineclassbindings
```

참고 `kubectl get virtualmachineclasses` 명령은 감독자 클러스터에 있는 모든 VM 클래스를 나열합니다. VM 클래스를 vSphere 네임스페이스와 연결해야 하기 때문에 대상 네임스페이스에 바인딩된 VM 클래스만 사용할 수 있습니다.

5 다음 명령을 실행하여 클러스터 매니페스트를 편집합니다.

```
kubectl edit tanzukubernetescluster/CLUSTER-NAME
```

6 검색 중에 잠재적인 버전 불일치를 방지하기 위해 `version` 문자열을 변경하고 `fullVersion`을 설정 해제하거나 null로 표시하여 매니페스트를 편집합니다.

예를 들어 제어부 및 작업자 노드에 대해 `best-effort-medium` VM 클래스를 사용하여 클러스터 매니페스트를 변경합니다.

```
spec:
  topology:
    controlPlane:
      class: best-effort-medium
      ...
    workers:
      class: best-effort-medium
      ...
```

제어부 및 작업자 노드에 대해 `guaranteed-large` VM 클래스를 사용하려면 다음을 수행합니다.

```
spec:
  topology:
    controlPlane:
      class: guaranteed-large
      ...
    workers:
      class: guaranteed-large
      ...
```

7 변경 내용을 매니페스트 파일에 적용합니다.

파일을 저장하면 kubectl이 변경 내용을 클러스터에 적용합니다. 백그라운드에서 Tanzu Kubernetes Grid 서비스는 새 노드 VM을 프로비저닝하고 이전 노드를 스핀 다운합니다.

8 kubectl이 매니페스트 편집이 기록되었다고 보고하는지 확인합니다.

```
kubectl edit tanzukubernetescluster/tkgs-cluster-1
tanzukubernetescluster.run.tanzu.vmware.com/tkgs-cluster-1 edited
```

참고 오류가 발생하거나 kubectl이 클러스터 매니페스트가 편집되었다고 보고하지 않는 경우에는 KUBE_EDITOR 환경 변수를 사용하여 기본 텍스트 편집기를 올바르게 구성했는지 확인합니다. kubectl용 기본 텍스트 편집기 지정의 내용을 참조하십시오.

9 클러스터가 업데이트되고 있는지 확인합니다.

```
kubectl get tanzukubernetescluster
NAME             CONTROL PLANE   WORKER   DISTRIBUTION                      AGE   PHASE
tkgs-cluster-1   3               3        v1.18.5+vmware.1-tkg.1.c40d30d   21h   updating
```

10 클러스터가 업데이트되었는지 확인합니다.

```
kubectl get tanzukubernetescluster
NAME             CONTROL PLANE   WORKER   DISTRIBUTION                      AGE   PHASE
tkgs-cluster-1   3               3        v1.18.5+vmware.1-tkg.1.c40d30d   22h   running
```

스토리지 클래스를 변경하여 Tanzu Kubernetes 클러스터 업데이트

클러스터 노드에서 사용하는 스토리지 클래스를 변경하여 Tanzu Kubernetes 클러스터를 업데이트할 수 있습니다.

Tanzu Kubernetes Grid 서비스는 노드 풀에 대한 `StorageClass`를 변경하여(예를 들어,속성 `.spec.topology.controlPlane.storageClass` 또는 속성 `.spec.topology.workers.storageClass`를 변경하여) 클러스터 업데이트를 지원합니다. Tanzu Kubernetes Grid 서비스 클러스터 업데이트 정보의 내용을 참조하십시오.

사전 요구 사항

이 작업은 `kubectl edit tanzukubernetescluster/CLUSTER-NAME` 명령을 사용하여 클러스터 매니페스트를 업데이트합니다. kubectl edit 명령은 KUBE_EDITOR 또는 EDITOR 환경 변수로 정의된 텍스트 편집기에서 클러스터 매니페스트를 엽니다. 파일을 저장하면 클러스터가 변경 사항으로 업데이트됩니다. kubectl용 기본 텍스트 편집기 지정의 내용을 참조하십시오.

절차

1 감독자 클러스터로 인증합니다. vCenter Single Sign-On 사용자로 감독자 클러스터에 연결의 내용을 참조하십시오.

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

2 대상 Tanzu Kubernetes 클러스터가 프로비저닝된 vSphere 네임스페이스로 컨텍스트를 전환합니다.

```
kubectl config use-context SUPERVISOR-NAMESPACE
```

3 사용 가능한 스토리지 클래스를 확인하고 무엇을 사용할지 결정하려면 다음 명령을 실행합니다.

```
kubectl describe tanzukubernetescluster CLUSTER-NAME
```

4 다음 명령을 실행하여 클러스터 매니페스트를 편집합니다.

```
kubectl edit tanzukubernetescluster/CLUSTER-NAME
```

5 `storageClass` 값을 변경하여 매니페스트를 편집합니다.

예를 들어 클러스터 매니페스트를 제어부 및 작업자 노드에 대한 `silver-storage-class` 클래스에서

```
spec:
  topology:
    controlPlane:
      ...
      storageClass: silver-storage-class
    workers:
      ...
      storageClass: silver-storage-class
```

제어부 및 작업자 노드에 대한 `gold-storage-class` 클래스로 변경합니다.

```
spec:
  topology:
    controlPlane:
      ...
      storageClass: gold-storage-class
    workers:
      ...
      storageClass: gold-storage-class
```

6 변경 내용을 매니페스트 파일에 적용합니다.

파일을 저장하면 kubectl이 변경 내용을 클러스터에 적용합니다. 백그라운드에서 Tanzu Kubernetes Grid 서비스는 새 노드 VM을 프로비저닝하고 이전 노드를 스핀 다운합니다.

7 kubectl이 매니페스트 편집이 기록되었다고 보고하는지 확인합니다.

```
kubectl edit tanzukubernetescluster/tkgs-cluster-1
tanzukubernetescluster.run.tanzu.vmware.com/tkgs-cluster-1 edited
```

참고 오류가 발생하거나 kubectl이 클러스터 매니페스트가 편집되었다고 보고하지 않는 경우에는 KUBE_EDITOR 환경 변수를 사용하여 기본 텍스트 편집기를 올바르게 구성했는지 확인합니다. kubectl용 기본 텍스트 편집기 지정의 내용을 참조하십시오.

8 클러스터가 업데이트되고 있는지 확인합니다.

```
kubectl get tanzukubernetescluster
NAME             CONTROL PLANE   WORKER   DISTRIBUTION                      AGE   PHASE
tkgs-cluster-1   3               3        v1.18.5+vmware.1-tkg.1.c40d30d   21h   updating
```

9 클러스터가 업데이트되었는지 확인합니다.

```
kubectl get tanzukubernetescluster
NAME             CONTROL PLANE   WORKER   DISTRIBUTION                      AGE   PHASE
tkgs-cluster-1   3               3        v1.18.5+vmware.1-tkg.1.c40d30d   22h   running
```

패치 방법을 사용하여 Tanzu Kubernetes 클러스터 업데이트

`kubectl patch` 방법을 사용하여 Tanzu Kubernetes 클러스터에 대한 "인플레이스" 업데이트를 수행할 수 있습니다. `kubectl patch` 방법은 `kubectl edit` 명령을 사용하여 지원되는 클러스터 업데이트 작업 중 하나를 수행하는 방법의 대안입니다.

Kubectl Patch 명령 정보

제한 사항 이 항목에 설명된 대로 `kubectl patch`를 사용하여 TKGS v1alpha2 API를 준수하도록 클러스터 규격을 업데이트하지 마십시오. 이러한 유형의 업데이트에는 `kubectl edit`를 사용해야 합니다. 클러스터 규격이 TKGS v1alpha2 API로 변환된 후 Tanzu Kubernetes 릴리스 업데이트의 내용을 참조하십시오.

`kubectl patch` 명령은 클러스터에 대한 "인플레이스" 업데이트를 수행합니다. 이 명령의 목적은 Kubernetes 버전을 업그레이드하는 방법을 제공하는 것이며 그 접근 방법이 여기에 설명되어 있습니다. `kubectl patch` 명령에 대한 자세한 내용은 Kubernetes 설명서에서 kubectl patch를 사용하여 API 개체 인플레이스 업데이트를 참조하십시오.

여기에 설명된 방식은 UNIX 셸 명령 `read`를 사용하여 키보드 입력을 가져와서 `$PATCH`라는 변수에 할당합니다. `kubectl patch` 명령은 Kubernetes API를 호출하여 클러스터 매니페스트를 업데이트합니다. `--type=merge` 플래그는 데이터에 기존 매니페스트와 다른 속성만 포함되어 있다는 것을 나타냅니다.

패치 방법을 사용하여 Kubernetes 버전 업그레이드

롤링 업데이트를 트리거하는 가장 일반적인 방법은 .spec.distribution.version 및 .spec.distribution.fullVersion 속성을 사용하여 클러스터에 대한 Kubernetes 배포 버전을 변경하는 것입니다. 검색하는 동안 잠재적인 버전 불일치를 방지하기 위해 version 힌트를 업데이트하고 fullVersion을 설정 해제하거나 null로 표시합니다.

```
$ read -r -d '' PATCH <<'EOF'
spec:
  distribution:
    fullVersion: null    # NOTE: Must set to null when updating just the version field
    version: v1.18.5
EOF
```

kubectl patch 명령을 사용하여 업데이트를 적용합니다. 클러스터 매니페스트에서 줄 바꿈 문자를 유지하려면 "$PATCH" 변수 앞뒤에 따옴표를 포함해야 합니다. *TKG-CLUSTER-NAME* 값을 클러스터의 실제 이름으로 바꿉니다.

```
kubectl patch --type=merge tanzukubernetescluster TKG-CLUSTER-NAME --patch "$PATCH"
```

예상 결과:

```
tanzukubernetescluster.run.tanzu.vmware.com/TKG-CLUSTER-NAME patched
```

패치 방법을 사용하여 노드의 VirtualMachineClass를 변경하여 클러스터 업데이트

Tanzu Kubernetes 클러스터의 롤링 업데이트를 트리거하는 또 다른 방법은 노드 풀의 VirtualMachineClass를 변경하는 것입니다. 즉, .spec.topology.controlPlane.class 속성 또는 .spec.topology.workers.class 속성을 변경하는 것입니다.

```
read -r -d '' PATCH <<'EOF'
spec:
  topology:
    controlPlane:
      class: best-effort-xlarge
    workers:
      class: best-effort-xlarge
EOF
```

kubectl patch 명령을 사용하여 변수를 클러스터 이름으로 바꾸고 업데이트를 적용합니다.

```
kubectl patch --type=merge tanzukubernetescluster TKG-CLUSTER-NAME --patch "$PATCH"
```

예상 결과:

```
tanzukubernetescluster.run.tanzu.vmware.com/TKG-CLUSTER-NAME patched
```

패치 방법을 사용하여 노드의 StorageClass를 변경하여 클러스터 업데이트

Tanzu Kubernetes 클러스터의 롤링 업데이트를 트리거하는 또 다른 방법은 노드 풀의 StorageClass를 변경하는 것입니다. 즉, .spec.topology.controlPlane.storageClass 속성 또는 .spec.topology.workers.storageClass 속성을 변경하는 것입니다.

```
$ read -r -d '' PATCH <<'EOF'
spec:
  topology:
    controlPlane:
      storageClass: gc-storage-profile
    workers:
      storageClass: gc-storage-profile
EOF
```

kubectl patch 명령을 사용하여 변수를 클러스터 이름으로 바꾸고 업데이트를 적용합니다.

```
kubectl patch --type=merge tanzukubernetescluster TKG-CLUSTER-NAME --patch "$PATCH"
```

예상 결과:

```
tanzukubernetescluster.run.tanzu.vmware.com/TKG-CLUSTER-NAME patched
```

vSphere with Tanzu 백업 및 복원

18

vSphere 포드 및 Tanzu Kubernetes 클러스터는 물론 vSphere with Tanzu 설치를 지원하는 vCenter Server 및 NSX-T 인프라에서 실행되는 워크로드를 백업하고 복원할 수 있습니다.

본 장은 다음 항목을 포함합니다.

- vSphere with Tanzu 백업 및 복원에 대한 고려 사항
- 감독자 클러스터에 vSphere용 Velero 플러그인 설치 및 구성
- vSphere용 Velero 플러그인를 사용하여 vSphere 포드 백업 및 복원
- Tanzu Kubernetes 클러스터에 vSphere용 Velero 플러그인 설치 및 구성
- vSphere용 Velero 플러그인를 사용하여 Tanzu Kubernetes 클러스터 워크로드 백업 및 복원
- Tanzu Kubernetes 클러스터에 독립형 Velero 및 Restic 설치 및 구성
- 독립형 Velero 및 Restic을 사용하여 Tanzu Kubernetes 클러스터 워크로드 백업 및 복원
- vCenter Server 백업 및 복원
- NSX-T Data Center 백업 및 복원

vSphere with Tanzu 백업 및 복원에 대한 고려 사항

이 항목에서는 vSphere with Tanzu의 백업 및 복원 프로세스에 대한 개요를 제공하고 vSphere with Tanzu의 백업 및 복원 전략을 구현하기 위한 개괄적인 고려 사항을 제공합니다.

vSphere with Tanzu 백업 및 복원은 다양한 계층과 도구로 구성됩니다.

이 표에는 하향식 워크로드에서 인프라에 이르는 해당 계층 및 도구가 요약되어 있습니다. 해당 계층에 대한 백업 및 복원 수행에 대한 자세한 내용은 개별 섹션을 참조하십시오.

시나리오	Tools	주석
vSphere 포드 백업 및 복원	Velero Plugin for vSphere	감독자 클러스터에 플러그인을 설치하고 구성합니다. **참고** 플러그인은 감독자 클러스터 상태를 백업하지 않습니다. 감독자 클러스터에 vSphere용 Velero 플러그인 설치 및 구성의 내용을 참조하십시오. vSphere용 Velero 플러그인를 사용하여 vSphere 포드 백업 및 복원의 내용을 참조하십시오.
Tanzu Kubernetes 클러스터에서 상태 비저장 및 상태 저장 워크로드를 백업하고 Tanzu Kubernetes Grid 서비스에서 프로비저닝된 클러스터로 복원	Velero Plugin for vSphere	Kubernetes 메타데이터와 영구 볼륨을 모두 백업하고 복원할 수 있습니다. Velero 스냅샷 생성(Restic 아님)은 영구 볼륨에 사용됩니다. Tanzu Kubernetes 클러스터에 vSphere용 Velero 플러그인 설치 및 구성의 내용을 참조하십시오. vSphere용 Velero 플러그인를 사용하여 Tanzu Kubernetes 클러스터 워크로드 백업 및 복원의 내용을 참조하십시오.
Tanzu Kubernetes 클러스터에서 상태 비저장 및 상태 저장 워크로드를 백업하고 Tanzu Kubernetes Grid 서비스에서 프로비저닝되지 않은 적합한 Kubernetes 클러스터로 복원	독립형 Velero 및 Restic	이식성이 필요한 경우 독립형 Velero를 사용합니다. 상태 저장 애플리케이션의 경우 Restic을 포함해야 합니다. Tanzu Kubernetes 클러스터에 독립형 Velero 및 Restic 설치 및 구성의 내용을 참조하십시오. 독립형 Velero 및 Restic을 사용하여 Tanzu Kubernetes 클러스터 워크로드 백업 및 복원의 내용을 참조하십시오.
감독자 클러스터 감독자 클러스터 업그레이드 후에는 새 백업을 수행해야 합니다. 이전 버전의 감독자 클러스터가 필요한 백업으로 vCenter Server를 복원하는 것은 지원되지 않습니다.	vCenter Server Velero Plugin for vSphere 독립형 Velero 및 Restic	백업에서 vCenter Server를 복원합니다. vCenter에서 세 개의 감독자 클러스터 제어부 VM을 모두 다시 생성합니다. 플러그인 또는 독립형 Velero 및 Restic을 사용하여 백업에서 클러스터 워크로드를 복원합니다.

시나리오	Tools	주석
vCenter 구성	vCenter Server	vCenter가 손실된 경우 vCenter Server를 사용하여 vCenter 개체를 백업하고 복원합니다. vCenter Server 백업 및 복원의 내용을 참조하십시오.
NSX-T Data Center	NSX-T Manager	로드 밸런서 및 수신 서비스는 NSX-T 백업에 따라 다릅니다. NSX-T Manager를 사용하여 NSX-T 데이터베이스를 백업 및 복원합니다. NSX-T Data Center 백업 및 복원의 내용을 참조하십시오.

감독자 클러스터에 vSphere용 Velero 플러그인 설치 및 구성

감독자 클러스터에 vSphere용 Velero 플러그인을 설치하면 vSphere용 Velero 플러그인을 사용하여 vSphere 포드에서 실행되는 워크로드를 백업하고 복원할 수 있습니다.

개요

vSphere용 Velero 플러그인는 vSphere with Tanzu 워크로드를 백업하고 복원하기 위한 솔루션을 제공합니다. 이 솔루션을 사용하려면 여러 구성 요소를 설치하고 구성해야 합니다. vSphere용 Velero 플러그인가 감독자 클러스터에 설치 및 구성되면 vSphere 포드를 백업 및 복원할 수 있습니다. 영구 워크로드의 경우 vSphere용 Velero 플러그인를 사용하여 영구 볼륨의 스냅샷을 생성할 수 있습니다.

감독자 클러스터에 vSphere용 Velero 플러그인를 설치하는 것은 vSphere용 Velero 플러그인를 사용하여 Tanzu Kubernetes 클러스터 워크로드를 백업하고 복원하기 위한 사전 요구 사항이기도 합니다.

참고 vSphere용 Velero 플러그인는 감독자 클러스터 상태를 백업하고 복원하는 데 사용할 수 없습니다. vSphere with Tanzu 백업 및 복원에 대한 고려 사항의 내용을 참조하십시오.

참고 vSphere용 Velero 플러그인 자체로는 증분 백업을 수행하지 않습니다. Dell EMC PowerProtect는 증분 백업을 지원하고 Velero 및 vSphere용 Velero 플러그인를 활용합니다.

사전 요구 사항

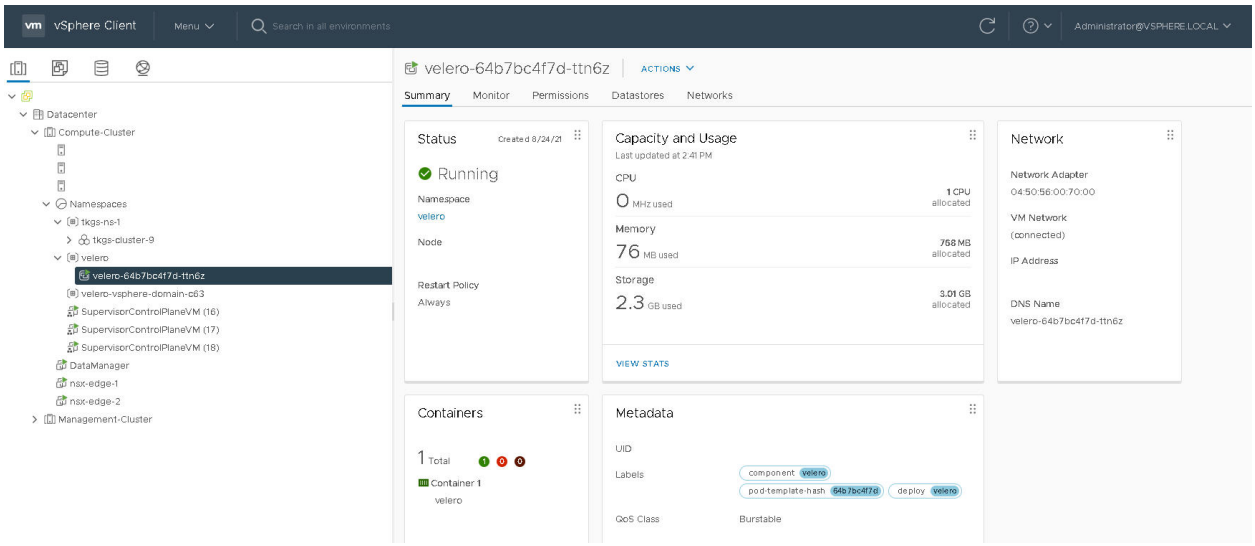
vSphere용 Velero 플러그인를 설치하기 전에 다음 사전 요구 사항을 준수합니다.

- 워크로드 관리는 NSX-T Data Center 네트워킹을 통해 사용하도록 설정됩니다. NSX-T Data Center 네트워킹으로 워크로드 관리 사용 항목을 참조하십시오.
- 감독자 클러스터는 버전 1.21.1 이상입니다.
- vSphere 네임스페이스가 생성 및 구성됩니다.
- vSphere 관리자 역할의 멤버이거나 다음과 같은 vSphere 권한이 있어야 합니다.
  - **SupervisorServices.Manage**

- **Namespaces.Manage**
- **Namespaces.Configure**

이 스크린샷은 vSphere용 Velero 플러그인 설치의 종료 상태를 보여줍니다.

- NSX-T 네트워킹은 vSphere 포드 배포를 지원하는 데 사용됩니다.
- Data Manager VM이 배포됩니다.
- Velero Operator가 사용되도록 설정되고 `velero-vsphere-domain-cXX` 네임스페이스에서 실행되고 있습니다.
- `velero`라는 네임스페이스가 구성되었습니다.
- vSphere용 Velero 플러그인가 `velero` 네임스페이스에서 vSphere 포드로 실행되고 있습니다.



업그레이드

이 지침에서는 vSphere 7 U3을 실행하고 있다고 가정합니다. 이전에 vSphere 7 U2 P3 환경에 vSphere용 Velero 플러그인를 설치한 경우에는 업그레이드 시 Data Manager VM 및 **Velero vSphere Operator**가 새 프레임워크로 마이그레이션됩니다. **Velero vSphere Operator**가 새 vSphere 서비스 형식으로 변환됩니다. 어떠한 작업도 필요하지 않습니다.

백업 및 복원 트래픽을 위한 전용 네트워크 생성(선택 사항)

필수는 아니지만 운영 환경에서는 백업 및 복원 트래픽을 vSphere with Tanzu 관리 네트워크 트래픽과 분리하는 것이 좋습니다. 이 작업에는 두 가지 측면이 있습니다.

- NFC(Network File Copy)를 지원하도록 ESXi 호스트에 태그 지정
- NSX-T Data Center를 사용하여 백업 및 복원 네트워크 구성

전용 NBD(Network Block Device) 전송을 지원하도록 vSphere 7.x ESXi 호스트를 구성하려면, 워크로드 관리를 사용하도록 설정한 vCenter Server 클러스터의 각 ESXi 호스트에 VMkernel NIC를 추가하고 이 NIC에 `vSphereBackupNFC`를 설정합니다. `vSphereBackupNFC` 태그가 VMkernel 어댑터의 NIC 유형에 적용되면 백업 및 복원 트래픽은 선택한 가상 NIC를 통과합니다.

이 구성을 수행하려면 Virtual Disk Development Kit를 사용합니다. NBD 설명서를 참조하십시오.

참고 VMkernel NIC에서 `vSphereBackupNFC`를 사용하도록 설정하지 않으면 백업 및 복원 트래픽은 백업 및 복원 네트워크를 구성하더라도 이 네트워크를 통해 전송되지 않습니다. `vSphereBackupNFC`를 사용하도록 설정하지 않으면 트래픽은 vSphere 관리 네트워크를 통해 이동합니다.

`vSphereBackupNFC` 태그를 사용하도록 설정되면 클러스터에 대한 기존 vDS(vSphere Distributed Switch)를 다음과 같이 업데이트하여 NSX-T를 사용하는 백업 및 복원 네트워크를 구성합니다.

- vSphere Client에서 **메뉴 > 네트워킹**을 선택합니다.
- 클러스터에 대한 기존 vDS를 선택합니다.
- vDS를 마우스 오른쪽 버튼으로 클릭하고 **분산 포트 그룹 > 새 분산 포트 그룹**을 선택합니다.
- **BackupRestoreNetwork**라는 새 분산 포트 그룹을 생성합니다.
- VMkernel 어댑터를 **BackupRestoreNetwork** 분산 포트 그룹에 추가합니다.
- 워크로드 관리를 사용하도록 설정된 vCenter 클러스터의 모든 ESXi 호스트를 **BackupRestoreNetwork** 분산 포트 그룹에 연결합니다.
- `vSphereBackupNFC` 태그를 사용하도록 설정합니다.

기존 vDS에서 NSX-T 네트워크를 생성하는 방법에 대한 지침은 vSphere with Tanzu에 대한 NSX-T Data Center 설치 및 구성 항목을 참조하십시오.

S3 호환 개체 저장소 생성

영구 볼륨의 백업 및 복원을 위해서는 S3 호환 개체 저장소를 제공해야 합니다. Velero는 여러 개체 저장소 제공자를 지원합니다.

vSphere용 Velero 플러그인을 설치하려면 S3 호환 개체 저장소에 대한 다음 정보를 제공해야 합니다.

데이터 항목	예제 값
s3Url	http://my-s3-store.example.com
aws_access_key_id	ACCESS-KEY-ID-STRING
aws_secret_access_key	SECRET-ACCESS-KEY-STRING

다음 정보를 사용하여 암호 파일 이름 `s3-credentials`를 생성합니다. 이 파일은 vSphere용 Velero 플러그인를 설치할 때 참조합니다.

```
[default]
aws_access_key_id = ACCESS-KEY-ID-STRING
aws_secret_access_key = SECRET-ACCESS-KEY-STRING
```

MinIO는 설치하고 사용하기 쉬운 S3 호환 개체 저장소입니다. vSphere with Tanzu에는 사용하도록 설정할 수 있는 MinIO 감독자 서비스가 함께 제공됩니다. 자세한 내용은 vSphere with Tanzu에서 상태 저장 서비스 사용의 내용을 참조하십시오.

또는 Linux VM에 MinIO 서버를 수동으로 설치할 수 있습니다. 자세한 내용은 Tanzu Kubernetes 클러스터에 독립형 Velero 및 Restic 설치 및 구성에서 참조하십시오.

Data Manager 설치 및 구성

vSphere용 Velero 플러그인를 사용하여 백업 및 복원을 용이하게 하려면 하나 이상의 Data Manager VM을 배포하여 영구 볼륨 백업 데이터를 S3 호환 객체 스토리지 내부 및 외부로 이동합니다. Data Manager는 백업 시에는 vSphere 볼륨에서 원격 지속형 S3 호환 스토리지로, 복원 중에는 원격 S3 호환 스토리지에서 vSphere 볼륨으로 볼륨 스냅샷 데이터를 이동합니다.

vSphere with Tanzu 환경에서 Data Manager를 가상 시스템으로 설치합니다.

참고 Velero vSphere Operator를 사용하도록 설정할 때까지 Data Manager VM의 전원을 켜지 마십시오.

1 Data Manager OVA를 다운로드합니다.

   https://vsphere-velero-datamgr.s3-us-west-1.amazonaws.com/datamgr-ob-17253392-photon-3-release-1.1.ova

2 vSphere Client를 사용하여 워크로드 관리를 사용하도록 설정된 **데이터 센터**를 마우스 오른쪽 버튼으로 클릭하고 **OVF 템플릿 배포**를 선택합니다.

3 다운로드한 Data Manager OVA 파일을 선택하여 vCenter Server에 업로드합니다.

4 가상 시스템의 이름을 지정합니다(예: **`DataManager`**).

5 계산 리소스를 선택합니다. 이것은 감독자 클러스터가 구성된 vCenter 클러스터입니다.

6 VM 배포 세부 정보를 검토하고 **다음**을 클릭합니다.

7 라이센스 계약에 동의하고 **다음**을 클릭합니다.

8 스토리지를 선택하고 **다음**을 클릭합니다.

9 Data Manager VM에 대한 대상 네트워크를 선택합니다.

   - 전용 백업 및 복원 네트워크를 구성한 경우 **BackupRestoreNetwork**를 선택합니다. 백업 및 복원 트래픽을 위한 전용 네트워크 생성(선택 사항)의 내용을 참조하십시오.
   - 전용 백업 및 복원 네트워크를 구성하지 않은 경우 **관리 네트워크**를 선택합니다.

10 선택 항목을 검토하고 **마침**을 클릭하여 프로세스를 완료합니다.

11 최근 작업 패널을 사용하여 배포 진행률을 모니터링합니다.

   참고 "OVF 설명자를 사용할 수 없습니다."라는 오류가 표시되면 Chrome 브라우저를 사용하십시오.

12 Data Manager VM이 배포되면 VM에 대한 입력 매개 변수를 구성합니다.

13 VM을 마우스 오른쪽 버튼으로 클릭하고 **설정 편집**을 선택합니다.

14 [가상 하드웨어] 탭에서 CD/DVD 드라이브의 경우 **호스트 디바이스**를 **클라이언트 디바이스**로 변경합니다.

참고 이렇게 하지 않으면 필요한 고급 구성 설정을 저장할 수 없습니다.

15 **설정 편집 > VM 옵션** 탭에서**고급 > 구성 매개 변수 편집**을 선택합니다.

16 다음 설정 각각에 대한 입력 매개 변수를 구성합니다.

매개 변수	값
`guestinfo.cnsdp.vcUser`	VM을 배포할 수 있는 충분한 권한이 있는 vCenter Server 사용자 이름을 입력합니다.
`guestinfo.cnsdp.vcAddress`	vCenter Server IP 주소 또는 FQDN을 입력합니다.
`guestinfo.cnsdp.vcPasswd`	vCenter Server 사용자 암호를 입력합니다.
`guestinfo.cnsdp.vcPort`	기본값은 **`443`**입니다. 이 값을 변경하지 마십시오.
`guestinfo.cnsdp.wcpControlPlaneIP`	감독자 클러스터 IP 주소를 입력합니다. 이 값은 워크로드 관리를 사용하도록 설정된 vCenter 클러스터로 이동하고 **구성 > 네임스페이스 > 네트워크 > 관리 네트워크 > 시작 IP 주소**를 선택하여 가져옵니다.
`guestinfo.cnsdp.updateKubectl`	기본값은 **`false`**입니다. 이 값을 변경하지 마십시오.
`guestinfo.cnsdp.veleroNamespace`	기본값은 **`velero`**이며 변경해야 할 이유가 없는 한 그대로 두어야 합니다. 나중에 이 프로세스에서 이름이 `velero`인 감독자 클러스터에 vSphere 네임스페이스를 생성합니다. 이 이름은 일치해야 합니다.
`guestinfo.cnsdp.datamgrImage`	구성되지 않은 경우(설정되지 않은 경우) 시스템은 기본적으로 Docker Hub(`vsphereveleroplugin/data-manager-for-plugin:1.1.0`)에서 컨테이너 이미지를 가져옵니다.

17 확인을 클릭하여 구성을 저장하고 확인을 다시 클릭하여 VM 설정을 저장합니다.

참고 CD/DVD 드라이브를 **호스트 디바이스**에서 **클라이언트 디바이스**로 변경하지 않았으면 설정을 저장할 수 없습니다. 이 경우 작업을 취소하고 드라이브를 변경한 후 고급 구성 설정을 반복합니다.

18 Velero vSphere Operator(다음 섹션)를 사용하도록 설정할 때까지 Data Manager VM의 전원을 켜지 마십시오.

감독자 클러스터에 Velero vSphere Operator 서비스 설치

vSphere with Tanzu는 Velero vSphere Operator를 vSphere 서비스로 제공합니다. Velero vSphere Operator 서비스는 vSphere용 Velero 플러그인와 함께 작동하여 영구 볼륨 스냅샷 생성을 포함한 Kubernetes 워크로드의 백업 및 복원을 지원합니다. vSphere 서비스에 대한 자세한 내용은 장 8 vSphere with Tanzu를 사용하여 감독자 서비스 관리 항목을 참조하십시오.

다음 작업을 완료하여 **워크로드 관리**를 사용하도록 설정된 vCenter Server에 **Velero vSphere Operator** 규격을 등록하고 **Velero vSphere Operator**를 감독자 클러스터에 서비스로 설치합니다.

참고 **Velero vSphere Operator**는 vSphere 포드로 실행되므로 NSX-T 네트워킹이 필요합니다.

1 다음 위치에서 **Velero vSphere Operator**용 YAML을 다운로드합니다.

http://vmware.com/go/supervisor-service

서비스 규격 파일의 이름은 `velero-supervisorservice-1.0.0.yaml`입니다.

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

3 **서비스** 탭을 선택합니다.

4 맨 위에 있는 드롭다운 메뉴에서 대상 vCenter Server를 선택합니다.

5 다운로드한 서비스 규격 파일 `velero-supervisorservice-1.0.0.yaml`을 **새 서비스 추가** 카드에 끌어서 놓습니다.

또는 **추가**를 클릭하고 `velero-supervisorservice-1.0.0.yaml` 파일을 찾아 선택할 수 있습니다.

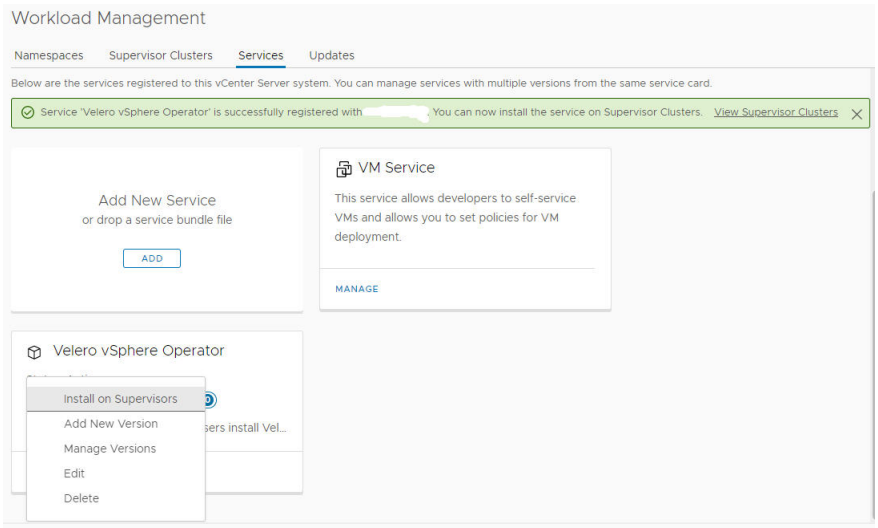
6 **다음**을 클릭하고 라이센스 계약에 동의합니다.

7 **마침**을 클릭합니다.

Velero vSphere Operator가 vCenter Server에 등록되었습니다. 서비스가 **활성** 상태인지 확인합니다. 비활성화된 상태이면 서비스를 설치할 수 없습니다.

8 **서비스** 탭에서 **Velero vSphere Operator** 규격을 찾습니다.

9 **작업 > 감독자에 설치**를 클릭합니다.



10 서비스를 설치할 대상 감독자 클러스터를 선택합니다.

참고 감독자 클러스터가 표시되지 않으면 NSX-T 네트워킹을 사용하고 있는지 확인합니다.

11 **Velero vSphere Operator** 서비스 설치를 다음과 같이 구성합니다.

   a 드롭다운에서 **1.1.0** 버전을 선택합니다.

   b **저장소 끝점**을 지정하지 마십시오.

   c 사용자 이름이나 암호를 입력하지 마십시오.

   d **다음**을 클릭합니다.

12 **마침**을 클릭하여 서비스 설치를 완료합니다.

감독자 클러스터에서 **Velero vSphere Operator** 서비스를 확인하고 Data Manager VM을 시작합니다.

1 vSphere Client 홈 메뉴에서 **인벤토리**를 선택합니다.

2 **워크로드 관리**를 사용하도록 설정된 vCenter 클러스터를 선택합니다.

3 **구성 > vSphere 서비스 > 개요**를 선택합니다.

4 **Velero vSphere Operator**가 설치되어 있고 해당 상태가 **구성됨**인지 확인합니다.

5 **네임스페이스** 리소스 풀에 `svc-velero-vsphere-domain-xxx`(xxx는 고유한 영숫자 토큰임)라는 새 네임스페이스가 보이는지 확인합니다. 이것은 시스템에서 **Velero vSphere Operator**용으로 생성된 네임스페이스입니다.

   참고 이 네임스페이스는 구성할 필요가 없으며 편집해서는 안 됩니다.

6 **호스트 및 클러스터** 보기에서 **Data Manager** VM을 선택합니다.

7 **Data Manager** VM을 마우스 오른쪽 버튼으로 클릭하고 전원을 켭니다.

vSphere용 Velero 플러그인용 vSphere 네임스페이스 생성

vSphere Client를 사용하여 감독자 클러스터에 vSphere 네임스페이스를 수동으로 생성합니다. 이 네임스페이스는 vSphere용 Velero 플러그인에 필요합니다. 지침은 vSphere 네임스페이스 생성 및 구성 항목을 참조하십시오.

- 네임스페이스의 이름을 **`velero`**라고 지정합니다.
- **velero** 네임스페이스를 선택하여 구성합니다.
- **velero** 네임스페이스에 대한 스토리지를 지정합니다.
- 적절한 권한이 있는 사용자에게 **velero** 네임스페이스에 대한 편집 권한을 부여합니다.

vSphere용 Velero 플러그인 설치

이제 vSphere용 Velero 플러그인를 설치할 준비가 되었습니다. 이를 위해 **velero-vsphere** CLI를 다운로드하여 실행합니다.

참고 이 절차를 수행하려면 Linux VM이 필요합니다. `kubectl-vsphere` 및 `kubectl` CLI를 실행하는 Linux 점프 호스트에 **velero-vsphere**를 다운로드하고 실행해야 합니다.

1 CLI를 실행할 수 있는 Linux VM을 생성합니다. 또는 감독자 클러스터에 액세스하는 기존 Linux 점프 호스트를 사용합니다.

2 다음 위치에서 vSphere용 Velero 플러그인 CLI를 다운로드합니다.

https://github.com/vmware-tanzu/velero-plugin-for-vsphere/releases/download/v1.1.0/velero-vsphere-1.1.0-linux-amd64.tar.gz

3 CLI를 Linux 점프 호스트에 안전하게 복사합니다. 예:

```
pscp -P 22 C:\temp\velero-vsphere-1.1.0-linux-amd64.tar.gz ubuntu@10.117.29.131:/home/ubuntu/tanzu
```

4 `velero-vsphere` CLI를 추출하고 쓰기 가능으로 만듭니다.

```
tar -xf velero-vsphere-1.1.0-linux-amd64.tar.gz
chmod +x velero-vsphere
```

5 다음 내용으로 `s3-credentials` 파일을 생성합니다.

```
aws_access_key_id = ACCESS-KEY-ID-STRING
aws_secret_access_key = SECRET-ACCESS-KEY-STRING
```

6 S3 호환 개체 저장소의 영역, URL 및 버킷 이름을 가져옵니다.

7 kubectl용 vSphere 플러그인를 사용하여 감독자 클러스터에 로그인합니다.

8 컨텍스트를 감독자 클러스터로 전환합니다.

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

9 다음 `velero-vsphere` CLI 명령을 실행하여 직접 생성한 **velero** 네임스페이스에 vSphere용 Velero 플러그인를 설치합니다.

BUCKET-NAME, **REGION**(두 개의 인스턴스) 및 **s3Url** 필드에 대한 자리 표시자 값을 적절한 값으로 바꿉니다. 앞의 지침에서 벗어난 값(암호 파일의 이름 또는 위치, 수동으로 생성된 `velero` 네임스페이스의 이름 등)이 있으면 해당 값도 조정합니다.

```
./velero-vsphere install \
    --namespace velero \
    --image velero/velero:v1.5.1 \
    --provider aws \
    --plugins velero/velero-plugin-for-aws:v1.1.0,vsphereveleroplugin/velero-plugin-for-
```

```
vsphere:1.1.0 \
    --bucket BUCKET-NAME \
    --secret-file s3-credentials \
    --snapshot-location-config region=REGION \
    --backup-location-config region=REGION,s3ForcePathStyle="true",s3Url=http://my-s3-
store.example.com
```

참고 감독자 클러스터에서 vSphere용 Velero 플러그인 v1.1.0 이상(예: `vspherevelero plugin/velero-plugin-for-vsphere:v1.1.1` 또는 `vspherevelero plugin/velero-plugin-for-vsphere:v1.2.0`)을 사용할 수 있습니다. Velero 버전은 v1.5.1(`velero/velero:v1.5.1`)이어야 합니다.

10 vSphere용 Velero 플러그인가 성공적으로 설치되었는지 확인합니다.

설치에 성공하면 다음 메시지가 표시됩니다.

```
Send the request to the operator about installing Velero in namespace velero
```

다음 명령을 실행하여 추가로 확인합니다. "Completed" 및 버전이 표시되어야 합니다.

```
kubectl -n velero get veleroservice default -o json | jq '.status'
```

예상 결과:

```
{
  "enabled": true,
  "installphase": "Completed",
  "version": "v1.5.1"
}
```

참고 위의 명령은 터미널로 전송되는 JSON 출력의 형식을 지정하는 `jq` 유틸리티가 설치되어 있다고 가정합니다. `jq`가 설치되어 있지 않으면 설치하거나 명령에서 해당 부분(`json` 이후의 모든 항목)을 제거합니다.

11 필요에 따라 문제를 해결합니다.

설치가 실패하면 설치를 제거하고 다시 시도합니다. 설치를 제거하려면 다음 섹션의 단계를 나열된 순서대로 완료합니다.

에어갭 환경에 Velero 플러그인 설치

에어갭 환경에 vSphere용 Velero 플러그인를 설치하려는 경우에는 사용자 지정된 이미지로 설치해야 합니다. 사용자 지정된 이미지의 일치하는 `backup-driver` 및 `data-manager-for-plugin` 이미지를 필요한 레지스트리에서 사용할 수 있고 Kubernetes 클러스터에서 액세스할 수 있는지 확인해야 합니다. 에어갭 환경에서는 Docker Hub의 릴리스된 이미지에 액세스할 수 없으므로 개인 레지스트리의 사용자 지정된 이미지가 필요합니다.

이 플러그인을 설치하려면 다음 단계를 수행하십시오.

1 velero-plugin-for-vsphere, backup-driver 및 data-manager-for-plugin의 릴리스된 이미지를 다운로드합니다.

2 이미지의 이름을 변경합니다. 즉, 일치하는 <Registry endpoint and path> 및 <Version tag>로 이미지에 태그를 지정하고 사용자 지정된 저장소에 업로드합니다.

3 사용자 지정한 velero-plugin-for-vsphere 이미지를 사용하여 플러그인을 설치합니다.

vanilla 클러스터에 vSphere용 Velero 플러그인을 설치하면 두 개의 추가 구성 요소인 backup-driver 배포와 data-manager-for-plugin DaemonSet가 백그라운드에 배포됩니다. 감독자 클러스터 및 Tanzu Kubernetes 클러스터에서는 backup-driver 배포만 배포됩니다.

velero-plugin-for-vsphere의 컨테이너 이미지를 제공하면 일치하는 backup-driver 및 data-manager-for-plugin 이미지는 이미지 구문 분석 메커니즘을 사용하여 구문 분석됩니다.

컨테이너 이미지는 다음 패턴으로 공식화됩니다.

```
<Registry endpoint and path>/<Container name>:<Version tag>
```

velero-plugin-for-vsphere 컨테이너 이미지를 제공하면 일치하는 <Registry endpoint and path> 및 <Version tag>와 함께 backup-driver 및 data-manager-for-plugin의 해당 이미지가 구문 분석됩니다.

예를 들어 다음 velero-plugin-for-vsphere 컨테이너 이미지를 고려합니다.

```
abc.io:8989/x/y/.../z/velero-plugin-for-vsphere:vX.Y.Z
```

backup-driver 및 data-manager-for-plugin의 일치하는 다음 이미지를 끌어와야 합니다.

```
abc.io:8989/x/y/.../z/backup-driver:vX.Y.Z
abc.io:8989/x/y/.../z/data-manager-for-plugin:vX.Y.Z
```

4 설치 문제를 해결합니다.

backup-driver 및 data-manager-for-plugin의 일치하는 이미지를 구문 분석하는 데 문제가 있거나 오류가 있는 경우 설치는 Docker Hub에 있는 공식 velerovspherepluginplugin 저장소의 해당 이미지로 폴백됩니다. 다음 문제는 폴백 메커니즘을 트리거합니다.

a 사용자 입력의 사용자 지정된 velero-plugin-for-vsphere 이미지에 예기치 않은 컨테이너 이름이 사용됩니다.

예를 들어 x/y/velero-velero-plugin-for-vsphere:v1.1.1이 사용됩니다.

b Velero 배포 이름이 velero 이외의 이름으로 사용자 지정됩니다. 예를 들어 Velero를 배포하기 전에 Velero 배포 이름이 Velero manifests 파일에서 velero-server로 업데이트되면 문제가 트리거됩니다.

velero-plugin-for-vsphere의 기존 이미지 구문 분석 메커니즘은 고정된 이름인 velero를 사용하는 Velero 배포만 인식할 수 있습니다.

vSphere용 Velero 플러그인 제거

vSphere용 Velero 플러그인를 제거하려면 다음 단계를 완료합니다. 나열된 순서대로 단계를 완료하십시오.

1 `velero-vsphere` CLI를 실행하여 vSphere용 Velero 플러그인를 제거합니다.

```
./velero-vsphere uninstall -n velero
```

2 `velero`라는 vSphere 포드가 제거되었는지 확인합니다.

```
kubectl get pods -n velero
```

포드가 "종료 중"이라고 표시되면 제거될 때까지 기다렸다가 계속합니다.

3 vSphere Client를 사용하여 수동으로 생성한 `velero`라는 vSphere 네임스페이스를 삭제합니다.

참고 네임스페이스 삭제가 완료될 때까지 다음 단계를 진행하지 마십시오. kubectl을 사용하여 `velero` 네임스페이스가 제거되었는지 확인할 수 있습니다. (하지만 kubectl을 사용하여 `velero` 네임스페이스를 제거하지는 마십시오.)

4 vSphere Client를 사용하여 감독자 클러스터에서 **Velero vSphere Operator**를 제거합니다.

a **워크로드 관리**를 사용하도록 설정된 vCenter 클러스터를 선택합니다.

b **구성 > vSphere 서비스 > 개요**를 선택합니다.

c **Velero vSphere Operator**를 선택합니다.

d **제거**를 클릭합니다.

이 작업을 수행하면 감독자 클러스터에서 **Velero vSphere Operator**가 제거됩니다. Operator는 **워크로드 관리 > 서비스** 페이지에서 다시 설치할 수 있습니다. 서비스를 완전히 제거하려면 **작업 > 삭제**를 선택하십시오.

vSphere용 Velero 플러그인를 사용하여 vSphere 포드 백업 및 복원

vSphere용 Velero 플러그인를 사용하여 vSphere 포드에서 실행되는 워크로드를 백업하고 복원할 수 있습니다.

개요

vSphere용 Velero 플러그인을 사용하여 감독자 클러스터의 vSphere 포드에서 실행되는 워크로드를 백업하고 복원할 수 있습니다. vSphere 포드에서 실행되는 상태 비저장 애플리케이션과 상태 저장 애플리케이션을 모두 백업하고 복원할 수 있습니다. 상태 저장 애플리케이션의 경우 vSphere용 Velero 플러그인을 사용하여 PV(영구 볼륨)의 스냅샷을 생성합니다.

참고 독립형 Velero를 Restic과 함께 사용하여 vSphere 포드를 백업 및 복원할 수 없습니다. 감독자 클러스터에 설치된 vSphere용 Velero 플러그인을 사용해야 합니다.

사전 요구 사항

vSphere 포드를 백업 및 복원하려면 vSphere용 Velero 플러그인을 설치하고 구성해야 합니다. 감독자 클러스터에 vSphere용 Velero 플러그인 설치 및 구성의 내용을 참조하십시오.

참고 vSphere용 Velero 플러그인는 감독자 클러스터의 상태를 백업 및 복원하지 않습니다.

vSphere 포드 백업

상태 비저장 vSphere 포드를 백업하려면 다음 명령을 실행합니다.

```
velero backup create <backup name> --include-namespaces=my-namespace
```

모든 로컬 스냅샷이 생성되고 Kubernetes 메타데이터가 개체 저장소에 업로드되면 백업이 `Completed`로 표시됩니다. 하지만 볼륨 스냅샷의 백업은 비동기식으로 발생하며 백그라운드에서 계속 수행 중일 수 있고 완료하는 데 다소 시간이 걸릴 수 있습니다.

볼륨 스냅샷의 상태는 스냅샷 및 업로드 사용자 지정 리소스를 모니터링하여 확인할 수 있습니다.

스냅샷 CRD

각 볼륨 스냅샷에 대해 스냅샷이 생성된 PVC와 동일한 네임스페이스에 스냅샷 사용자 지정 리소스가 생성됩니다. 다음 명령을 실행하여 PVC 네임스페이스의 모든 스냅샷을 가져올 수 있습니다.

```
kubectl get -n <pvc namespace> snapshot
```

스냅샷 CRD에는 `status.phase` 필드에 대해 다음과 같은 몇 가지 단계가 있습니다.

상태	설명
New	아직 처리되지 않았습니다.
Snapshotted	로컬 스냅샷이 생성되었습니다.
SnapshotFailed	로컬 스냅샷을 생성하지 못했습니다.
Uploading	스냅샷이 업로드되고 있습니다.
Uploaded	스냅샷이 업로드되었습니다.
UploadFailed	스냅샷을 업로드하지 못했습니다.
Canceling	스냅샷 업로드가 취소되고 있습니다.

상태	설명
Canceled	스냅샷 업로드가 취소되었습니다.
CleanupAfterUploadFailed	스냅샷 업로드 후 로컬 스냅샷을 정리하지 못했습니다.

업로드 CRD

개체 저장소에 업로드할 각 볼륨 스냅샷에 대해 Velero와 동일한 네임스페이스에 업로드 CR이 생성됩니다. 다음 명령을 실행하여 Velero 네임스페이스의 모든 업로드를 가져올 수 있습니다.

```
kubectl get -n <velero namespace> upload
```

업로드 CRD에는 `status.phase` 필드에 대해 다음과 같은 몇 가지 단계가 있습니다.

상태	설명
New	아직 처리되지 않았습니다.
InProgress	업로드 진행 중
UploadError	업로드하지 못했습니다.
CleanupFailed	업로드 후 로컬 스냅샷을 삭제하지 못했습니다. 재시도됩니다.
Canceling	업로드가 취소되고 있습니다. 스냅샷 업로드가 진행되는 동안 `velero backup delete`가 호출되는 경우 발생할 수 있습니다.
Canceled	업로드가 취소되었습니다.

업로드 오류 업로드는 주기적으로 재시도됩니다. 이때는 단계가 진행 중 단계로 돌아갑니다. 업로드가 성공적으로 완료된 후 해당 기록은 일정 기간 동안 남아 있다가 결국 제거됩니다.

vSphere 포드 복원

vSphere용 Velero 플러그인를 사용하여 백업된 vSphere 포드 워크로드를 복원하려면 다음 단계를 완료합니다.

1 복원할 워크로드에 대한 vSphere 네임스페이스를 생성합니다.

2 네임스페이스에 대한 스토리지 정책을 구성합니다.

3 다음 Velero 명령을 실행하여 워크로드를 복원합니다.

```
velero restore create --from-backup backup-name
```

볼륨 스냅샷 및 기타 Kubernetes 메타데이터가 현재 클러스터에 성공적으로 복원되면 Velero 복원이 `Completed`로 표시됩니다. 이때 이 복원과 관련된 vSphere 플러그인의 모든 작업도 완료됩니다. Velero 백업의 경우처럼 백그라운드에서 비동기식 데이터 이동 작업이 없습니다.

Velero 복원이 `Completed` 상태가 되기 전에 CloneFromSnapshot/Download CR을 아래와 같이 모니터링하여 볼륨 복원의 상태를 확인할 수 있습니다.

CloneFromSnapshot CRD

각 볼륨 스냅샷에서 복원하기 위해 원래 스냅샷이 생성된 PVC와 동일한 네임스페이스에 CloneFromSnapshot CR이 생성됩니다. 다음 명령을 실행하여 PVC 네임스페이스의 모든 CloneFromSnapshot을 가져올 수 있습니다.

```
kubectl -n <pvc namespace> get clonefromsnapshot
```

CloneFromSnapshot CRD에는 `status.phase` 필드에 대해 다음과 같은 몇 가지 단계가 있습니다.

상태	설명
New	스냅샷에서 복제가 완료되지 않았습니다.
Completed	스냅샷에서 복제가 완료되었습니다.
Failed	스냅샷에서 복제에 실패했습니다.

다운로드 CRD

개체 저장소에서 다운로드할 볼륨 스냅샷을 복원할 때마다 Velero와 동일한 네임스페이스에 다운로드 CR이 생성됩니다. 다음 명령을 실행하여 Velero 네임스페이스에서 모든 다운로드를 가져올 수 있습니다.

```
kubectl -n <velero namespace> get download
```

다운로드 CRD에는 `status.phase` 필드에 대해 다음과 같은 몇 가지 단계가 있습니다.

상태	설명
New	아직 처리되지 않았습니다.
InProgress	다운로드를 진행 중입니다.
Completed	다운로드가 완료되었습니다.
Retry	다운로드가 재시도됩니다. 백업 데이터를 다운로드하는 동안 오류가 발생하면 다운로드가 재시도됩니다.
Failed	다운로드에 실패했습니다.

Tanzu Kubernetes 클러스터에 vSphere용 Velero 플러그인 설치 및 구성

Tanzu Kubernetes 클러스터에 vSphere용 Velero 플러그인을 설치하면 vSphere용 Velero 플러그인을 사용하여 해당 클러스터에서 실행되는 워크로드를 백업하고 복원할 수 있습니다.

개요

vSphere용 Velero 플러그인는 Tanzu Kubernetes Grid 서비스에서 프로비저닝된 클러스터에 대한 Tanzu Kubernetes 클러스터 워크로드를 백업 및 복원하기 위한 솔루션을 제공합니다. 영구 워크로드의 경우 vSphere용 Velero 플러그인를 사용하여 영구 볼륨의 스냅샷을 생성할 수 있습니다.

참고 백업 및 복원하려는 Tanzu Kubernetes 클러스터 워크로드에 대해 이식성이 필요한 경우에는 vSphere용 Velero 플러그인를 사용하지 마십시오. Kubernetes 클러스터 간 이식성을 위해서는 독립형 Velero를 Restic과 함께 사용하십시오. Tanzu Kubernetes 클러스터에 독립형 Velero 및 Restic 설치 및 구성의 내용을 참조하십시오.

사전 요구 사항: 감독자 클러스터에 vSphere용 Velero 플러그인 설치

Tanzu Kubernetes 클러스터에 vSphere용 Velero 플러그인를 설치하려면 감독자 클러스터에 vSphere용 Velero 플러그인가 설치되어 있어야 합니다. 또한 감독자 클러스터가 NSX-T 네트워킹으로 구성되어야 합니다.

Tanzu Kubernetes 클러스터에 vSphere용 Velero 플러그인를 설치하기 전에 먼저 감독자 클러스터에 vSphere용 Velero 플러그인를 설치해야 합니다. 감독자 클러스터에 vSphere용 Velero 플러그인 설치 및 구성의 내용을 참조하십시오.

Linux Workstation에 Velero CLI 설치

Velero CLI는 Velero와 상호 작용하기 위한 표준 도구입니다. Velero CLI는 vSphere용 Velero 플러그인 CLI(`velero-vsphere`)보다 더 많은 기능을 제공하며 Tanzu Kubernetes 클러스터 워크로드를 백업하고 복원하는 데 필요합니다.

Linux 워크스테이션에 Velero CLI를 설치합니다. 이상적으로는 `kubectl`, `kubectl-vsphere`, `velero-vsphere`를 포함한 vSphere with Tanzu 환경에 대해 연결된 CLI를 실행하는 Linux와 동일한 점프 호스트입니다.

Velero CLI를 설치하려면 다음 단계를 완료합니다.

1 VMware 제품 다운로드 페이지에서 지원되는 버전의 Velero CLI를 다운로드합니다. 지원되는 Velero 버전에 대한 자세한 내용은 릴리스 정보를 참조하십시오.

2 명령줄을 열고 디렉토리를 Velero CLI 다운로드로 변경합니다.

```
gunzip velero-linux-v1.x.x_vmware.1.gz
```

3 Velero 바이너리를 확인합니다.

```
ls -l

-rw-r--r-- 1 root root 7142128 Aug 14 14:14 velero-linux-v1.x.x_vmware.1
```

4 Velero CLI에 실행 권한을 부여합니다.

```
chmod +x velero-linux-v1.x.x_vmware.1
```

5 Velero CLI를 시스템 경로로 이동하여 전체적으로 사용할 수 있도록 합니다.

```
cp velero-linux-v1.x.x_vmware.1 /usr/local/bin/velero
```

6 Velero CLI 설치를 확인합니다.

```
velero version

Client:
    Version: v1.x.x
```

S3 호환 버킷 세부 정보 가져오기

편의를 위해 이 단계에서는 감독자 클러스터에 vSphere용 Velero 플러그인을 설치할 때 구성한 것과 동일한 S3 호환 개체 저장소를 사용 중이라고 가정합니다. 운영 환경에서는 별도의 개체 저장소를 생성하는 것이 좋습니다.

vSphere용 Velero 플러그인을 설치하려면 S3 호환 개체 저장소에 대한 다음 정보를 제공해야 합니다.

데이터 항목	예제 값
s3Url	http://my-s3-store.example.com
aws_access_key_id	ACCESS-KEY-ID-STRING
aws_secret_access_key	SECRET-ACCESS-KEY-STRING

다음 정보를 사용하여 암호 파일 이름 `s3-credentials`를 생성합니다. 이 파일은 vSphere용 Velero 플러그인을 설치할 때 참조합니다.

```
aws_access_key_id = ACCESS-KEY-ID-STRING
aws_secret_access_key = SECRET-ACCESS-KEY-STRING
```

Tanzu Kubernetes 클러스터에 vSphere용 Velero 플러그인 설치

Velero CLI를 사용하여 백업 및 복원하려는 대상 Tanzu Kubernetes 클러스터에 vSphere용 Velero 플러그인을 설치하려고 합니다.

Velero CLI 컨텍스트는 `kubectl` 컨텍스트를 자동으로 따릅니다. Velero CLI 명령을 실행하여 대상 클러스터에 Velero 및 vSphere용 Velero 플러그인을 설치하기 전에 `kubectl` 컨텍스트를 대상 클러스터로 설정해야 합니다.

1 kubectl용 vSphere 플러그인을 사용하여 감독자 클러스터로 인증합니다. vCenter Single Sign-On 사용자로 감독자 클러스터에 연결의 내용을 참조하십시오.

2 `kubectl` 컨텍스트를 대상 Tanzu Kubernetes 클러스터로 전환합니다.

```
kubectl config use-context TARGET-TANZU-KUBERNETES-CLUSTER
```

3 다음 Velero CLI 명령을 실행하여 대상 클러스터에 Velero를 설치합니다.

BUCKET-NAME, **REGION**(두 개의 인스턴스) 및 **s3Url** 필드에 대한 자리 표시자 값을 적절한 값으로 바꿉니다. 앞의 지침에서 벗어난 값(암호 파일의 이름 또는 위치, 수동으로 생성된 `velero` 네임스페이스의 이름 등)이 있으면 해당 값도 조정합니다.

```
./velero install --provider aws \
--bucket BUCKET-NAME \
--secret-file ./s3-credentials \
--features=EnableVSphereItemActionPlugin \
--plugins velero/velero-plugin-for-aws:v1.1.0 \
--snapshot-location-config region=REGION \
--backup-location-config region=REGION,s3ForcePathStyle="true",s3Url=http://my-s3-
store.example.com
```

4 대상 클러스터에 vSphere용 Velero 플러그인를 설치합니다. 설치된 Velero는 Kubernetes API 서버와 통신하여 플러그인을 설치합니다.

```
velero plugin add vsphereveleroplugin/velero-plugin-for-vsphere:1.1.0
```

클러스터에서 vSphere용 Velero 플러그인 제거

vSphere용 Velero 플러그인를 제거하려면 다음 단계를 완료합니다.

1 `kubectl` 컨텍스트를 대상 Tanzu Kubernetes 클러스터로 전환합니다.

```
kubectl config use-context TARGET-TANZU-KUBERNETES-CLUSTER
```

2 플러그인을 제거하려면 다음 명령을 실행하여 Velero 배포에서 velero-plugin-for-vsphere의 InitContainer를 제거합니다.

```
velero plugin remove vsphereveleroplugin/velero-plugin-for-vsphere:1.1.0
```

3 제거를 완료하려면 백업 드라이버 배포 및 관련 CRD를 삭제합니다.

```
kubectl -n velero delete deployment.apps/backup-driver
```

```
kubectl delete crds \
backuprepositories.backupdriver.cnsdp.vmware.com \
backuprepositoryclaims.backupdriver.cnsdp.vmware.com \
clonefromsnapshots.backupdriver.cnsdp.vmware.com \
deletesnapshots.backupdriver.cnsdp.vmware.com \
snapshots.backupdriver.cnsdp.vmware.com
```

```
kubectl delete crds uploads.datamover.cnsdp.vmware.com downloads.datamover.cnsdp.vmware.com
```

vSphere용 Velero 플러그인를 사용하여 Tanzu Kubernetes 클러스터 워크로드 백업 및 복원

vSphere용 Velero 플러그인를 사용하여 Tanzu Kubernetes 클러스터 워크로드를 백업하고 복원할 수 있습니다. 단, 이식성이 필요한 경우에는 독립형 Velero를 사용합니다.

사전 요구 사항

vSphere용 Velero 플러그인를 사용하여 Tanzu Kubernetes 클러스터 워크로드를 백업 및 복원하려면 먼저 대상 클러스터에 Velero 및 vSphere용 Velero 플러그인를 설치해야 합니다. Tanzu Kubernetes 클러스터에 vSphere용 Velero 플러그인 설치 및 구성의 내용을 참조하십시오.

워크로드 백업

다음은 Velero 백업을 생성하는 명령의 예입니다.

```
velero backup create <backup name> --include-namespaces=my-namespace
```

모든 로컬 스냅샷이 생성되고 Kubernetes 메타데이터(볼륨 스냅샷 제외)가 개체 저장소에 업로드되면 Velero 백업이 `Completed`로 표시됩니다. 이 시점에 비동기 데이터 이동 작업(즉 볼륨 스냅샷 업로드)이 백그라운드에서 계속 진행되며 완료하는 데 다소 시간이 걸릴 수 있습니다. 볼륨 스냅샷의 상태는 스냅샷 CR(사용자 지정 리소스)을 모니터링하여 확인할 수 있습니다.

스냅샷

스냅샷은 영구 볼륨을 백업하는 데 사용됩니다. 각 볼륨 스냅샷에 대해 스냅샷이 생성된 PVC(영구 볼륨 할당)와 동일한 네임스페이스에 스냅샷 CR이 생성됩니다.

다음 명령을 실행하여 PVC 네임스페이스의 모든 스냅샷을 가져올 수 있습니다.

```
kubectl get -n <pvc namespace> snapshot
```

스냅샷 CRD(사용자 지정 리소스 정의)에는 `.status.phase` 필드에 대해 다음을 포함한 여러 단계가 있습니다.

스냅샷 단계	설명
New	아직 처리되지 않았습니다.
Snapshotted	로컬 스냅샷이 생성되었습니다.
SnapshotFailed	로컬 스냅샷을 생성하지 못했습니다.
Uploading	스냅샷이 업로드되고 있습니다.
Uploaded	스냅샷이 업로드되었습니다.
UploadFailed	스냅샷을 업로드하지 못했습니다.
Canceling	스냅샷 업로드가 취소되고 있습니다.

스냅샷 단계	설명
Canceled	스냅샷 업로드가 취소되었습니다.
CleanupAfterUploadFailed	스냅샷 업로드 후 로컬 스냅샷을 정리하지 못했습니다.

워크로드 복원

다음은 Velero 복원 명령의 예입니다.

```
velero restore create --from-backup <velero-backup-name>
```

볼륨 스냅샷 및 기타 Kubernetes 메타데이터가 현재 클러스터에 성공적으로 복원되면 Velero 복원이 `Completed`로 표시됩니다. 이때 이 복원과 관련된 vSphere 플러그인의 모든 작업도 완료됩니다. Velero 백업의 경우처럼 백그라운드에서 비동기식 데이터 이동 작업이 없습니다.

CloneFromSnapshot

각 볼륨 스냅샷에서 복원하기 위해 원래 스냅샷이 생성된 PVC와 동일한 네임스페이스에 `CloneFromSnapshot` CR(사용자 지정 리소스)이 생성됩니다. 다음 명령을 실행하여 PVC 네임스페이스의 모든 CloneFromSnapshot을 가져올 수 있습니다.

```
kubectl -n <pvc namespace> get clonefromsnapshot
```

CloneFromSnapshot CRD에는 `.status.phase` 필드에 대한 몇 가지 주요 단계가 있습니다.

스냅샷 단계	설명
New	스냅샷에서 복제가 완료되지 않았습니다.
InProgress	원격 저장소에서 vSphere 볼륨 스냅샷을 다운로드하고 있습니다.
Completed	스냅샷에서 복제가 완료되었습니다.
Failed	스냅샷에서 복제에 실패했습니다.

Tanzu Kubernetes 클러스터에 독립형 Velero 및 Restic 설치 및 구성

Tanzu Kubernetes에서 워크로드를 백업 및 복원하려면 데이터스토어를 생성하고 Kubernetes 클러스터에 Velero를 Restic과 함께 설치합니다.

개요

Tanzu Kubernetes 클러스터는 가상 시스템 노드에서 실행됩니다. Tanzu Kubernetes 클러스터를 백업하고 복원하려면 클러스터에 Velero 및 Restic을 설치합니다.

사전 요구 사항

Tanzu Kubernetes 클러스터에서 실행되는 워크로드를 백업하고 복원하기 위해 Velero 및 Restic을 설치하려면 환경이 다음 사전 요구 사항을 충족하는지 확인합니다.

- 여러 워크로드 백업을 저장하기에 충분한 스토리지가 있는 Linux VM. 이 VM에 MinIO를 설치합니다.
- kubectl용 vSphere 플러그인 및 kubectl을 포함하여 vSphere에 대한 Kubernetes CLI 도구가 설치되어 있는 Linux VM. 이 클라이언트 VM에 Velero CLI를 설치합니다. 이러한 VM이 없는 경우 Velero CLI를 로컬로 설치할 수 있지만 그에 따라 설치 단계를 조정해야 합니다.
- Kubernetes 환경을 인터넷에 액세스할 수 있으며 클라이언트 VM에서 연결할 수 있습니다.

MinIO 개체 저장소 설치 및 구성

Velero에는 Kubernetes 워크로드 백업의 대상으로 S3 호환 개체 저장소가 필요합니다. Velero는 이러한 개체 저장소 제공자를 여러 개 지원합니다. 간단히 하기 위해 이 지침에서는 개체 저장소 VM에서 로컬로 실행되는 S3 호환 스토리지 서비스인 MinIO를 사용합니다.

1 MinIO를 설치합니다.

```
wget https://dl.min.io/server/minio/release/linux-amd64/minio
```

2 MinIO에 실행 권한을 부여합니다.

```
chmod +x minio
```

3 MinIO용 파일 시스템에 디렉토리를 생성합니다.

```
mkdir /DATA-MINIO
```

4 MinIO 서버를 시작합니다.

```
./minio server /DATA-MINIO
```

5 MinIO 서버가 시작되면 끝점 URL, AccessKey 및 SecretKey를 비롯한 중요 데이터스토어 인스턴스 세부 정보가 제공됩니다. 테이블에 끝점 URL, AccessKey 및 SecretKey를 기록하십시오.

데이터스토어 메타데이터	값
끝점 URL	
AccessKey	
SecretKey	

6 브라우저에서 MinIO 서버 끝점 URL을 열어 MinIO 데이터스토어로 이동합니다.

7 MinIO 서버에 로그인하고 AccessKey 및 SecretKey를 제공합니다.

8 MinIO as a Service를 사용하도록 설정하려면 `minio.service` 스크립트를 다운로드하여 자동 시작을 위해 MinIO를 구성합니다.

```
curl -O https://raw.githubusercontent.com/minio/minio-service/master/linux-systemd/
minio.service
```

9 `minio.service` 스크립트를 편집하고 `ExecStart`에 대해 다음 값을 추가합니다.

```
ExecStart=/usr/local/bin/minio server /DATA-MINIO path
```

10 수정된 스크립트를 저장합니다.

11 다음 명령을 실행하여 MinIO 서비스를 구성합니다.

```
cp minio.service /etc/systemd/system
cp minio /usr/local/bin/
systemctl daemon-reload
systemctl start minio
systemctl status minio
systemctl enable minio
```

12 MinIO 브라우저를 시작하고 개체 저장소에 로그인하여 백업 및 복원을 위한 MinIO 버킷을 생성합니다.

13 버킷 생성 아이콘을 클릭합니다.

14 버킷 이름(예: `my-cluster-backups`)을 입력합니다.

15 버킷이 생성되었는지 확인합니다.

16 기본적으로 새 MinIO 버킷은 읽기 전용입니다. Velero 독립형 백업 및 복원을 위해서는 MinIO 버킷이 읽기-쓰기여야 합니다. 버킷을 읽기-쓰기로 설정하려면 버킷을 선택하고 말줄임표(점) 링크를 클릭합니다.

17 **정책 편집**을 선택합니다.

18 정책을 **읽기 및 쓰기**로 변경합니다.

19 **추가**를 클릭합니다.

20 대화상자를 닫으려면 X를 클릭합니다.

Velero CLI 설치

VM 클라이언트 또는 로컬 시스템에 Velero CLI를 설치합니다.

1 VMware 제품 다운로드 페이지에서 vSphere with Tanzu용으로 서명된 Velero 바이너리 버전을 다운로드합니다.

참고 VMware의 지원을 받으려면 VMware에서 서명한 Velero 바이너리를 사용해야 합니다.

2 명령줄을 열고 디렉토리를 Velero CLI 다운로드로 변경합니다.

3 다운로드 파일의 압축을 풉니다. 예:

```
gunzip velero-linux-vX.X.X_vmware.1.gz
```

4 Velero 바이너리를 확인합니다.

```
ls -l
```

5 Velero CLI에 실행 권한을 부여합니다.

```
chmod +x velero-linux-vX.X.X_vmware.1
```

6 Velero CLI를 시스템 경로로 이동하여 전체적으로 사용할 수 있도록 합니다.

```
cp velero-linux-vX.X.X_vmware.1 /usr/local/bin/velero
```

7 설치를 확인합니다.

```
velero version
```

Tanzu Kubernetes 클러스터에 Velero 및 Restic 설치

Velero CLI 컨텍스트는 kubectl 컨텍스트를 자동으로 따릅니다. Velero CLI 명령을 실행하여 대상 클러스터에 Velero 및 Restic을 설치하기 전에 kubectl 컨텍스트를 설정합니다.

1 MinIO 버킷의 이름을 검색합니다. 예: `my-cluster-backups`

2 MinIO 버킷에 대한 AccessKey 및 SecretKey를 가져옵니다.

3 Velero CLI가 작동할 클러스터를 알 수 있도록 대상 Kubernetes 클러스터에 대한 컨텍스트를 설정합니다.

```
kubectl config use-context tkgs-cluster-name
```

4 `credentials-minio`라는 암호 파일을 생성합니다. 이 파일을 수집한 MinIO 서버 액세스 자격 증명으로 업데이트합니다. 예:

```
aws_access_key_id = 0XXNO8JCCGV41QZBV0RQ
aws_secret_access_key = clZ1bf8Ljkvkmq7fHucrKCkxV39BRbcycGeXQDfx
```

참고 "NoCredentialProviders: 체인에 올바른 제공자가 없습니다."라는 설명과 함께 "백업 저장소를 가져오는 동안 오류가 발생했습니다." 오류 메시지가 표시되면 자격 증명 파일의 시작 부분에 `[default]` 줄을 추가합니다. 예:

```
[default]
aws_access_key_id = 0XXNO8JCCGV41QZBV0RQ
aws_secret_access_key = clZ1bf8Ljkvkmq7fHucrKCkxV39BRbcycGeXQDfx
```

5 파일을 저장하고 파일이 제자리에 있는지 확인합니다.

```
ls
```

6 다음 명령을 실행하여 대상 Kubernetes 클러스터에 Velero 및 Restic을 설치합니다. 두 URL을 모두 MinIO 인스턴스의 URL로 바꿉니다.

```
velero install \
--provider aws \
--plugins velero/velero-plugin-for-aws:v1.0.0 \
--bucket tkgs-velero \
--secret-file ./credentials-minio \
--use-volume-snapshots=false \
--use-restic \
--backup-location-config \
region=minio,s3ForcePathStyle="true",s3Url=http://10.199.17.63:9000,publicUrl=http://
10.199.17.63:9000
```

7 Velero 및 Restic의 설치를 확인합니다.

```
kubectl logs deployment/velero -n velero
```

8 `velero` 네임스페이스를 확인합니다.

```
kubectl get ns
```

9 `velero` 및 `restic` 포드를 확인합니다.

```
kubectl get all -n velero
```

Restic DaemonSet 문제 해결(필요한 경우)

Kubernetes 클러스터에서 3-포드 Restic DaemonSet을 실행하려면 Restic DaemonSet 규격을 업데이트하고 hostPath를 수정해야 할 수 있습니다. 이 문제에 대한 자세한 내용은 Velero 설명서에서 Restic 통합을 참조하십시오.

1 3-포드 Restic DaemonSet을 확인합니다.

```
kubectl get pod -n velero
```

포드가 CrashLoopBackOff 상태인 경우 다음과 같이 편집합니다.

2 `edit` 명령을 실행합니다.

```
kubectl edit daemonset restic -n velero
```

3 hostPath를 `/var/lib/kubelet/pods`에서 `/var/vcap/data/kubelet/pods`로 변경합니다.

```
- hostPath:
    path: /var/vcap/data/kubelet/pods
```

4 파일을 저장합니다.

5 3-포드 Restic DaemonSet을 확인합니다.

```
kubectl get pod -n velero
```

```
NAME                      READY   STATUS    RESTARTS   AGE
restic-5jln8              1/1     Running   0          73s
restic-bpvtq              1/1     Running   0          73s
restic-vg8j7              1/1     Running   0          73s
velero-72c84322d9-le7bd   1/1     Running   0          10m
```

Velero 메모리 제한 조정(필요한 경우)

Velero 백업이 여러 시간 동안 `status=InProgress`를 반환하는 경우 제한 및 요청 메모리 설정을 늘립니다.

1 다음 명령을 실행합니다.

```
kubectl edit deployment/velero -n velero
```

2 제한 및 요청 메모리 설정을 기본값인 `256Mi` 및 `128Mi`에서 `512Mi` 및 `256Mi`로 변경합니다.

```
ports:
- containerPort: 8085
  name: metrics
  protocol: TCP
resources:
  limits:
    cpu: "1"
    memory: 512Mi
  requests:
    cpu: 500m
    memory: 256Mi
terminationMessagePath: /dev/termination-log
terminationMessagePolicy: File
```

독립형 Velero 및 Restic을 사용하여 Tanzu Kubernetes 클러스터 워크로드 백업 및 복원

독립형 Velero 및 Restic을 사용하여 Tanzu Kubernetes 클러스터 워크로드를 백업하고 복원할 수 있습니다. 이 방법은 vSphere용 Velero 플러그인 사용에 대한 대안입니다. vSphere용 Velero 플러그인 대신 독립형 Velero를 사용하는 주된 이유는 이식성이 필요한 경우입니다. 상태 저장 워크로드에는 Restic이 필요합니다.

사전 요구 사항

독립형 Velero 및 Restic을 사용하여 Tanzu Kubernetes 클러스터 워크로드를 백업하고 복원하려면 대상 클러스터에 독립형 버전의 Velero 및 Restic을 설치해야 합니다. Tanzu Kubernetes 클러스터에 독립형 Velero 및 Restic 설치 및 구성의 내용을 참조하십시오.

참고 Restic과 함께 독립형 Velero를 사용하여 Kubernetes 클러스터를 백업하고 복원하면 이식성이 지원됩니다. 즉, Tanzu Kubernetes Grid 서비스에 의해 프로비저닝되지 않은 Kubernetes 클러스터로 클러스터 워크로드를 복원하려면 독립형 Velero를 사용해야 합니다.

Tanzu Kubernetes 클러스터에서 실행되는 상태 비저장 애플리케이션 백업

Tanzu Kubernetes 클러스터에서 실행되는 상태 비저장 애플리케이션을 백업하려면 Velero를 사용해야 합니다.

이 예에서는 `--include namespaces` 태그(모든 애플리케이션 구성 요소가 해당 네임스페이스에 있는 경우)를 사용하여 예제 상태 비저장 애플리케이션을 백업하고 복원하는 방법을 보여줍니다.

```
velero backup create example-backup --include-namespaces example-backup
```

다음이 표시됩니다.

```
Backup request "example-backup" submitted successfully.
Run `velero backup describe example-backup` or `velero backup logs example-backup` for more
details.
```

생성된 백업을 확인합니다.

```
velero backup get
```

```
velero backup describe example-backup
```

S3 호환 개체 저장소(예: MinIO 서버)에서 Velero 버킷을 확인합니다.

Velero는 Kubernetes CRD(사용자 지정 리소스 정의)에 일부 메타데이터를 씁니다.

```
kubectl get crd
```

Velero CRD를 사용하면 다음과 같은 특정 명령을 실행할 수 있습니다.

```
kubectl get backups.velero.io -n velero
```

```
kubectl describe backups.velero.io guestbook-backup -n velero
```

Tanzu Kubernetes 클러스터에서 실행되는 상태 비저장 애플리케이션 복원

Tanzu Kubernetes 클러스터에서 실행되는 상태 비저장 애플리케이션을 복원하려면 Velero를 사용해야 합니다.

예제 애플리케이션의 복원을 테스트하려면 예제 애플리케이션을 삭제합니다.

네임스페이스를 삭제합니다.

```
kubectl delete ns guestbook
namespace "guestbook" deleted
```

애플리케이션을 복원합니다.

```
velero restore create --from-backup example-backup
```

다음이 표시됩니다.

```
Restore request "example-backup-20200721145620" submitted successfully.
Run `velero restore describe example-backup-20200721145620` or `velero restore logs example-
backup-20200721145620` for more details.
```

애플리케이션이 복원되었는지 확인합니다.

```
velero restore describe example-backup-20200721145620
```

다음 명령을 실행하여 확인합니다.

```
velero restore get
```

```
kubectl get ns
```

```
kubectl get pod -n example
```

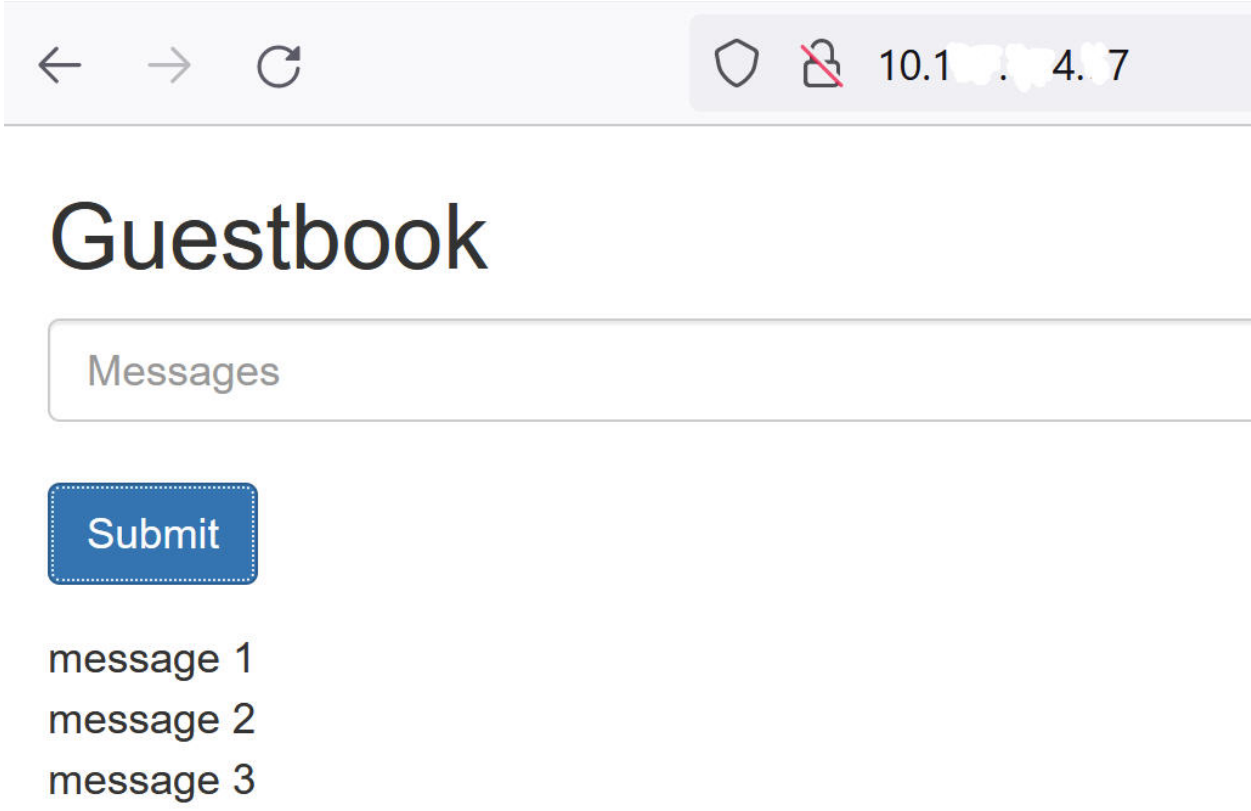
```
kubectl get svc -n example
```

Tanzu Kubernetes 클러스터에서 실행되는 상태 저장 애플리케이션 백업

Tanzu Kubernetes 클러스터에서 실행되는 상태 저장 애플리케이션을 백업하려면 애플리케이션 메타데이터 및 영구 볼륨에 저장된 애플리케이션 데이터를 모두 백업해야 합니다. 이렇게 하려면 Velero와 Restic이 모두 필요합니다.

이 예에서는 방명록 애플리케이션을 사용합니다. Tanzu Kubernetes 클러스터에 방명록 애플리케이션을 배포했다고 가정합니다. 지침은 Tanzu Kubernetes 방명록 자습서 항목을 참조하십시오.

상태 저장 백업 및 복원을 시연할 수 있도록 프런트 엔드 웹 페이지를 사용하여 방명록 애플리케이션에 메시지를 제출하여 메시지가 지속되도록 합니다. 예:



이 예에서는 --include namespace 태그와 포드 주석을 사용하여 방명록 애플리케이션을 백업하고 복원하는 방법을 보여줍니다.

참고 이 예에서는 주석을 사용합니다. 하지만 Velero 버전 1.5 이상에서는 더 이상 주석이 필요하지 않습니다. 주석을 사용하지 않으려면 백업을 생성할 때 --default-volumes-to-restic 옵션을 사용하면 됩니다. 그러면 Restic을 사용하여 모든 PV가 자동으로 백업됩니다. 자세한 내용은 https://velero.io/docs/v1.5/restic/의 내용을 참조하십시오.

백업 절차를 시작하려면 포드의 이름을 가져옵니다.

```
kubectl get pod -n guestbook
```

예:

```
kubectl get pod -n guestbook

NAME                                          READY   STATUS    RESTARTS   AGE
guestbook-frontend-deployment-85595f5bf9-h8cff  1/1     Running             0          55m
guestbook-frontend-deployment-85595f5bf9-lw6tg  1/1     Running             0          55m
guestbook-frontend-deployment-85595f5bf9-wpqc8  1/1     Running             0          55m
redis-leader-deployment-64fb8775bf-kbs6s        1/1     Running             0          55m
redis-follower-deployment-84cd76b975-jrn8v      1/1     Running             0          55m
redis-follower-deployment-69df9b5688-zml4f      1/1     Running             0          55m
```

영구 볼륨은 Redis 포드에 연결됩니다. Restic을 사용하여 이러한 상태 저장 포드를 백업하기 때문에 volumeMount라는 이름으로 상태 저장 포드에 주석을 추가해야 합니다.

상태 저장 포드에 주석을 추가하려면 volumeMount를 알고 있어야 합니다. mountName을 가져오려면 다음 명령을 실행합니다.

```
kubectl describe pod redis-leader-deployment-64fb8775bf-kbs6s -n guestbook
```

결과에 redis-leader-data의 Containers.leader.Mounts: /data가 표시됩니다. 이 마지막 토큰이 리더 포드 주석에 사용할 volumeMount 이름입니다. 팔로워의 경우 redis-follower-data입니다. 소스 YAML에서 volumeMount 이름을 가져올 수도 있습니다.

각 Redis 포드에 주석을 추가합니다. 예를 들면 다음과 같습니다.

```
kubectl -n guestbook annotate pod redis-leader-64fb8775bf-kbs6s backup.velero.io/backup-
volumes=redis-leader-data
```

다음이 표시됩니다.

```
pod/redis-leader-64fb8775bf-kbs6s annotated
```

주석을 확인합니다.

```
kubectl -n guestbook describe pod redis-leader-64fb8775bf-kbs6s | grep Annotations
Annotations:  backup.velero.io/backup-volumes: redis-leader-data
```

```
kubectl -n guestbook describe pod redis-follower-779b6d8f79-5dphr | grep Annotations
Annotations:  backup.velero.io/backup-volumes: redis-follower-data
```

Velero 백업을 수행합니다.

```
velero backup create guestbook-backup --include-namespaces guestbook
```

다음이 표시됩니다.

```
Backup request "guestbook-backup" submitted successfully.
Run `velero backup describe guestbook-pv-backup` or `velero backup logs guestbook-pv-backup`
for more details.
```

생성된 백업을 확인합니다.

```
velero backup get

NAME                  STATUS      ERRORS   WARNINGS   CREATED
EXPIRES   STORAGE LOCATION   SELECTOR
guestbook-backup      Completed   0        0          2020-07-23 16:13:46 -0700 PDT
29d       default            <none>
```

백업 세부 정보를 확인합니다.

```
velero backup describe guestbook-backup --details
```

Velero를 사용하면 다음과 같은 다른 명령을 실행할 수 있습니다.

```
kubectl get backups.velero.io -n velero

NAME               AGE
guestbook-backup   4m58s
```

그리고:

```
kubectl describe backups.velero.io guestbook-backup -n velero
```

Tanzu Kubernetes 클러스터에서 실행되는 상태 저장 애플리케이션 복원

Tanzu Kubernetes 클러스터에서 실행되는 상태 저장 애플리케이션을 복원하려면 애플리케이션 메타데이터 및 영구 볼륨에 저장된 애플리케이션 데이터를 모두 복원해야 합니다. 이렇게 하려면 Velero와 Restic이 모두 필요합니다.

이 예에서는 이전 섹션에서 설명한 대로 상태 저장 방명록 애플리케이션을 백업했다고 가정합니다.

상태 저장 애플리케이션의 복원을 테스트하려면 해당 네임스페이스를 삭제합니다.

```
kubectl delete ns guestbook
namespace "guestbook" deleted
```

애플리케이션 삭제를 확인합니다.

```
kubectl get ns
kubectl get pvc,pv --all-namespaces
```

애플리케이션을 복원합니다.

```
Restore request "guestbook-backup-20200723161841" submitted successfully.
Run `velero restore describe guestbook-backup-20200723161841` or `velero restore logs
guestbook-backup-20200723161841` for more details.
```

상태 저장 방명록 애플리케이션이 복원되었는지 확인합니다.

```
velero restore describe guestbook-backup-20200723161841

Name:         guestbook-backup-20200723161841
Namespace:    velero
Labels:       <none>
Annotations:  <none>
```

```
Phase:  Completed

Backup:  guestbook-backup

Namespaces:
  Included:  all namespaces found in the backup
  Excluded:  <none>

Resources:
  Included:        *
  Excluded:        nodes, events, events.events.k8s.io, backups.velero.io,
restores.velero.io, resticrepositories.velero.io
  Cluster-scoped:  auto

Namespace mappings:  <none>

Label selector:  <none>

Restore PVs:  auto

Restic Restores (specify --details for more information):
  Completed:  3
```

다음 추가 명령을 실행하여 복원을 확인합니다.

```
velero restore get

NAME                                BACKUP                STATUS      ERRORS   WARNINGS
CREATED                         SELECTOR
guestbook-backup-20200723161841      guestbook-backup      Completed   0        0
2021-08-11 16:18:41 -0700 PDT   <none>
```

네임스페이스가 복원되었는지 확인합니다.

```
kubectl get ns

NAME              STATUS   AGE
default           Active   16d
guestbook         Active   76s
...
velero            Active   2d2h
```

애플리케이션이 복원되었는지 확인합니다.

```
vkubectl get all -n guestbook

NAME                                READY   STATUS    RESTARTS   AGE
pod/frontend-6cb7f8bd65-h2pnb       1/1     Running   0          6m27s
pod/frontend-6cb7f8bd65-kwlpr       1/1     Running   0          6m27s
pod/frontend-6cb7f8bd65-snwl4       1/1     Running   0          6m27s
pod/redis-leader-64fb8775bf-kbs6s   1/1     Running   0          6m28s
pod/redis-follower-779b6d8f79-5dphr 1/1     Running   0          6m28s
pod/redis-follower-899c7e2z65-8apnk 1/1     Running   0          6m28s
```

```
NAME                                    TYPE           CLUSTER-IP       EXTERNAL-IP
PORT(S)        AGE
service/guestbook-frontend              LoadBalancer   10.10.89.59      10.19.15.99
80:31513/TCP   65s
service/redis-follower                  ClusterIP      10.111.163.189   <none>
6379/TCP       65s
service/redis-leader                    ClusterIP      10.111.70.189    <none>
6379/TCP       65s

NAME                                            READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/guestbook-frontend-deployment   3/3     3            3           65s
deployment.apps/redis-follower-deployment       1/2     2            1           65s
deployment.apps/redis-leader-deployment         1/1     1            1           65s

NAME                                                      DESIRED   CURRENT   READY   AGE
replicaset.apps/guestbook-frontend-deployment-56fc5b6b47   3         3         3       65s
replicaset.apps/redis-follower-deployment-6fc9cf5759       2         2         1       65s
replicaset.apps/redis-leader-deployment-7d89bbdbcf         1         1         1       65s
```

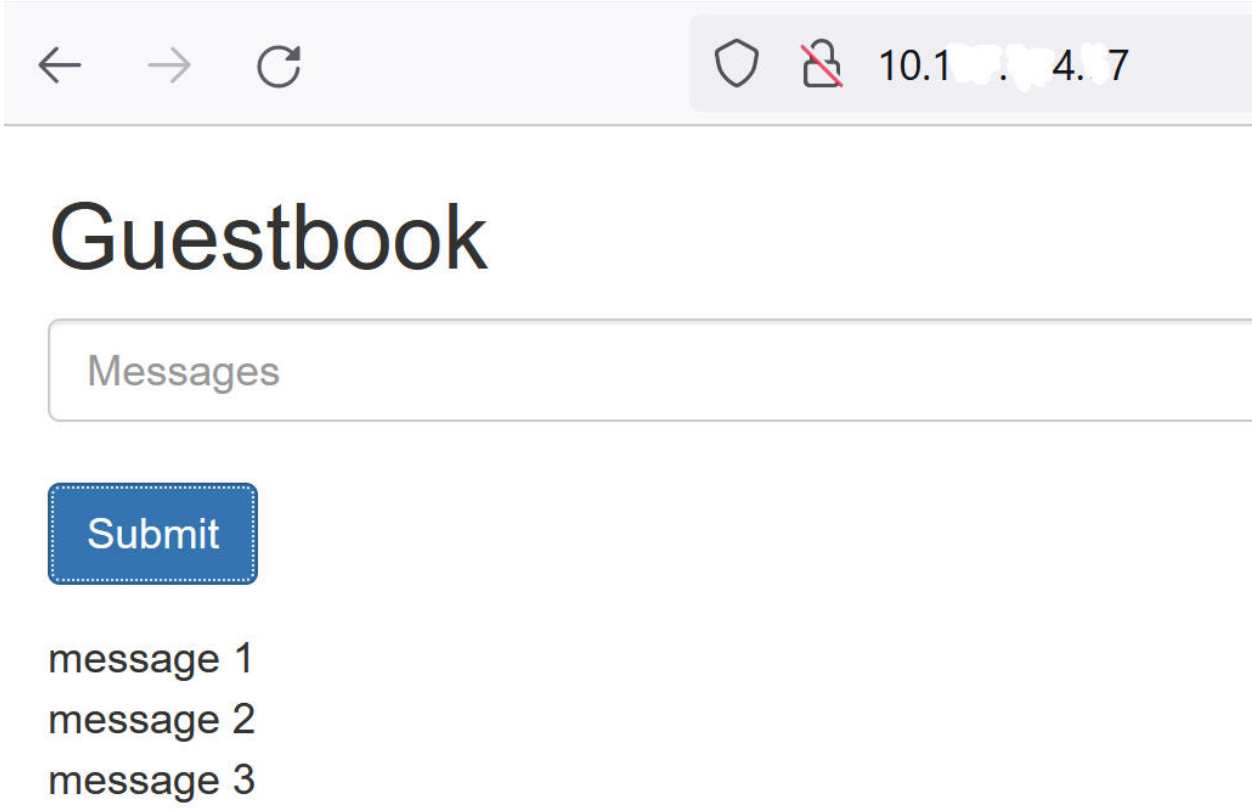
영구 볼륨이 복원되었는지 확인합니다.

```
kubectl get pvc,pv -n guestbook

NAME                                       STATUS
VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
persistentvolumeclaim/redis-leader-claim   Bound    pvc-a2f6e6d4-42db-4fb8-
a198-5379a2552509   2Gi        RWO            thin-disk      2m40s
persistentvolumeclaim/redis-follower-claim Bound    pvc-55591938-921f-452a-
b418-2cc680c0560b   2Gi        RWO            thin-disk      2m40s

NAME                                                       CAPACITY   ACCESS MODES   RECLAIM
POLICY   STATUS   CLAIM                                STORAGECLASS   REASON   AGE
persistentvolume/pvc-55591938-921f-452a-b418-2cc680c0560b   2Gi        RWO
Delete           Bound    guestbook/redis-follower-claim    thin-disk               2m40s
persistentvolume/pvc-a2f6e6d4-42db-4fb8-a198-5379a2552509   2Gi        RWO
Delete           Bound    guestbook/redis-leader-claim      thin-disk               2m40s
```

마지막으로 방명록 프런트 엔드 서비스의 외부 IP를 사용하여 방명록 프런트 엔드에 액세스하고 자습서 시작 부분에서 제출한 메시지가 복원되었는지 확인합니다. 예:



vCenter Server 백업 및 복원

이 항목에서는 vSphere with Tanzu 배포의 컨텍스트에서 vCenter Server를 백업하고 복원하는 방법을 설명합니다.

vCenter 클러스터 HA

고가용성 감독자 클러스터를 지원하려면 vSphere with Tanzu를 사용하기 위해 vSphere with Tanzu가 사용되도록 설정된 vCenter 클러스터가 고가용성으로 구성되어 있어야 합니다. 자세한 내용은 VMware vSphere 설명서에서 vSphere 가용성을 참조하십시오.

vCenter Server 백업 및 복구

vCenter Server는 네트워크 연결 스토리지에 대한 파일 백업을 지원합니다. vCenter Server를 백업하고 복원하려면 기본 vCenter Server의 백업을 생성합니다. 자세한 내용은 vCenter 설명서에서 vCenter Server의 파일 기반 백업 및 복원을 참조하십시오.

NSX-T Data Center 백업 및 복원

vSphere with Tanzu 워크로드 중단 시 네트워크 기능을 지원하려면 NSX-T Data Center를 백업 및 복원합니다.

요구 사항

NSX-T Data Center를 백업 및 복원하기 위해 3개의 NSX Manager 노드가 배포되고 NSX 관리부에 액세스하도록 구성된 HA VIP가 있다고 가정합니다. 또한 Edge 노드에 대해 HA VIP를 사용하여 배포된 Edge 노드가 2개 이상 있습니다. 자세한 내용은 vSphere with Tanzu에 대한 NSX-T Data Center 구성의 내용을 참조하십시오.

NSX-T 백업 및 복원

NSX-T Data Center는 NSX Manager 노드 및 개체의 백업 및 복원을 지원하는 제품 내 백업 및 복구를 제공합니다. 자세한 내용은 NSX-T 설명서에서 NSX Manager 백업 및 복원을 참조하십시오.

vSphere with Tanzu 문제 해결

19

다음과 같은 문제 해결 기법 및 모범 사례를 vSphere with Tanzu의 인프라에 대해 사용할 수 있습니다.

본 장은 다음 항목을 포함합니다.

스토리지 모범 사례 및 문제 해결

vSphere with Tanzu의 스토리지 환경에 대해 다음과 같은 모범 사례 및 문제 해결 방법을 사용할 수 있습니다.

vSAN이 아닌 데이터스토어에서 제어부 VM에 대한 반선호도 규칙 사용

클러스터에서 vSAN 이외의 데이터스토어를 vSphere with Tanzu와 함께 사용하는 경우 가용성을 위해 서로 다른 데이터스토어에 3개의 제어부 VM을 배치합니다.

제어부 VM은 시스템에서 관리되므로 수동으로 마이그레이션할 수 없습니다. 데이터스토어 클러스터와 Storage DRS의 조합을 사용하여 제어부 VM을 재조정하고 별도의 데이터스토어에 배치합니다.

절차

1 vSphere Client에서 데이터스토어 클러스터를 생성합니다.

   a 데이터 센터로 이동합니다.

   b 데이터 센터 개체를 마우스 오른쪽 버튼으로 클릭하고 **새 데이터스토어 클러스터**를 선택합니다.

   c 데이터스토어 클러스터의 이름을 지정하고 **Storage DRS 설정**을 사용하도록 설정했는지 확인합니다.

   d 클러스터 자동화 수준을 **자동화 안 함(수동 모드)**으로 설정합니다.

e Storage DRS 런타임 설정을 기본값으로 둡니다.

f vSphere with Tanzu와 함께 사용하도록 설정된 ESXi 클러스터를 선택합니다.

g 데이터스토어 클러스터에 추가할 모든 공유 데이터스토어를 선택합니다.

h **마침**을 클릭합니다.

2 제어부 VM에 대한 Storage DRS 규칙을 정의합니다.

a 데이터스토어 클러스터로 이동합니다.

b **구성** 탭을 클릭하고 **구성**에서 **규칙**을 클릭합니다.

c **추가** 아이콘을 클릭하고 규칙의 이름을 입력합니다.

d **규칙 사용**을 사용하도록 설정했는지 확인합니다.

e **규칙 유형**을 **VM 반선호도**로 설정합니다.

f **추가** 아이콘을 클릭하고 감독자 제어부 VM 3개를 선택합니다.

g **확인**을 클릭하여 구성을 마칩니다.

3 VM 재정의를 생성합니다.

a 데이터스토어 클러스터로 이동합니다.

b **구성** 탭을 클릭하고 **구성**에서 **VM 재정의**를 클릭합니다.

c **추가** 아이콘을 클릭하고 제어부 VM 3개를 선택합니다.

d Storage DRS 자동화 수준을 사용하도록 설정하려면 **재정의** 확인란을 선택하고 값을 **완전히 자동화됨**으로 설정합니다.

e **마침**을 클릭합니다.

결과

이 작업은 제어부 VM에 대해서만 Storage DRS를 사용하도록 설정하고 VM이 서로 다른 데이터스토어에 있도록 재조정합니다.

Storage vMotion이 사용되면 SDRS 규칙 및 재정의를 제거하고, Storage DRS를 사용하지 않도록 설정하고, 데이터스토어 클러스터를 제거할 수 있습니다.

vSphere에서 제거된 스토리지 정책이 계속 Kubernetes 스토리지 클래스로 표시됨

vSphere Client를 사용하여 vCenter Server 또는 감독자 클러스터의 네임스페이스에서 스토리지 정책을 제거할 경우 일치하는 스토리지 클래스는 Kubernetes 환경에 남아 있지만 사용할 수는 없습니다.

문제

kubectl get sc 명령을 실행하면 출력이 네임스페이스에서 사용할 수 있는 스토리지 클래스를 계속 나열합니다. 그러나 스토리지 클래스는 사용할 수 없습니다. 예를 들어 새 영구 볼륨 할당에 대해 스토리지 클래스를 사용하려는 시도가 실패합니다.

스토리지 클래스가 Kubernetes 배포에서 이미 사용되고 있는 경우 배포가 예기치 않게 동작할 수 있습니다.

해결책

1 네임스페이스에 있는 스토리지 클래스를 확인하려면 **kubectl describe namespace *namespace_name*** 명령을 실행합니다.

일치하는 스토리지 정책이 제거된 경우 이 명령의 출력은 스토리지 클래스를 나열하지 않습니다.

2 스토리지 클래스가 배포에서 이미 사용되고 있는 경우 스토리지 클래스를 복원합니다.

a vSphere Client를 사용하여 제거한 정책과 동일한 이름의 새 스토리지 정책을 생성합니다.

예를 들어 *Gold* 정책을 삭제한 경우 새 정책의 이름을 *Gold*로 지정합니다. vSphere with Tanzu에 대한 스토리지 정책 생성의 내용을 참조하십시오.

b 네임스페이스에 정책을 할당합니다.

네임스페이스의 스토리지 설정 변경의 내용을 참조하십시오.

네임스페이스에 정책을 할당한 후 vSphere with Tanzu는 이전 스토리지 클래스를 삭제하고 동일한 이름의 일치하는 스토리지 클래스를 생성합니다.

vSAN Direct에서 외부 스토리지 사용

vSphere with Tanzu 환경에서 vSAN Direct를 사용하는 경우 외부 공유 스토리지를 사용하여 관리 내부 VM 및 기타 메타데이터를 저장할 수 있습니다.

문제

동종 vSAN Direct 클러스터를 배포할 때는 클러스터의 각 ESXi 호스트에 복제된 vSAN 데이터스토어를 생성하여 vSphere with Tanzu 관리 VM 및 기타 메타데이터를 저장해야 합니다. vSAN 데이터스토어는 공간을 사용하고 각 호스트에 추가 I/O 컨트롤러가 필요하며 vSAN Direct를 지원할 수 있는 하드웨어 구성을 제한합니다.

vSAN 데이터스토어를 구성하는 대신, 외부 공유 스토리지를 사용하여 관리 내부 VM 및 기타 메타데이터를 저장할 수 있습니다.

해결책

1 vSAN 또는 vSAN Direct가 클러스터의 ESXi 호스트에 배포된 경우 구성에서 호스트를 지웁니다.

   a vSAN 또는 vSAN Direct에 할당된 디스크를 제거합니다. vSAN에서 디스크 그룹 또는 디바이스 제거를 참조하십시오.

   b (선택 사항) 스크립트를 사용하여 vSAN Direct용 호스트의 디스크에 태그를 지정합니다. 스크립트를 사용하여 vSAN Direct용 스토리지 디바이스에 태그 지정의 내용을 참조하십시오.

2 VMware Cloud Foundation을 사용하여 외부 스토리지가 있는 워크로드 도메인을 생성합니다.

   NFS, vVols 또는 FC와 같은 스토리지 옵션 중 하나를 선택해야 합니다. 이러한 옵션 중 하나만 선택할 수 있습니다.

   자세한 내용은 VMware Cloud Foundation 운영 및 관리 가이드에서 "워크로드 도메인 작업" 을 참조하십시오.

   이 단계에서는 vCenter Server 및 지정된 ESXi 호스트가 있는 워크로드 도메인을 배포합니다. 외부 스토리지는 모든 호스트에 마운트되고 기본 클러스터에 추가됩니다.

3 vSAN을 사용하도록 설정합니다.

   vSAN에 대해 할당된 디스크가 없는지 확인합니다.

   자세한 내용은 기존 클러스터에서 vSAN 사용을 참조하십시오.

   이 단계에서는 vSAN 네트워크와 함께 0바이트 vSAN 데이터스토어를 생성합니다. vSAN에 로컬 디스크가 사용되지 않습니다.

4 vSAN Direct용 호스트에 로컬 디스크를 할당합니다.

   자세한 내용은 vSphere with Tanzu에 대한 vSAN Direct 설정 항목을 참조하십시오.

   할당하는 각 디바이스에 대해 vSAN Direct는 별도의 데이터스토어를 생성합니다.

5 vSAN Direct에 대한 스토리지 정책을 생성합니다.

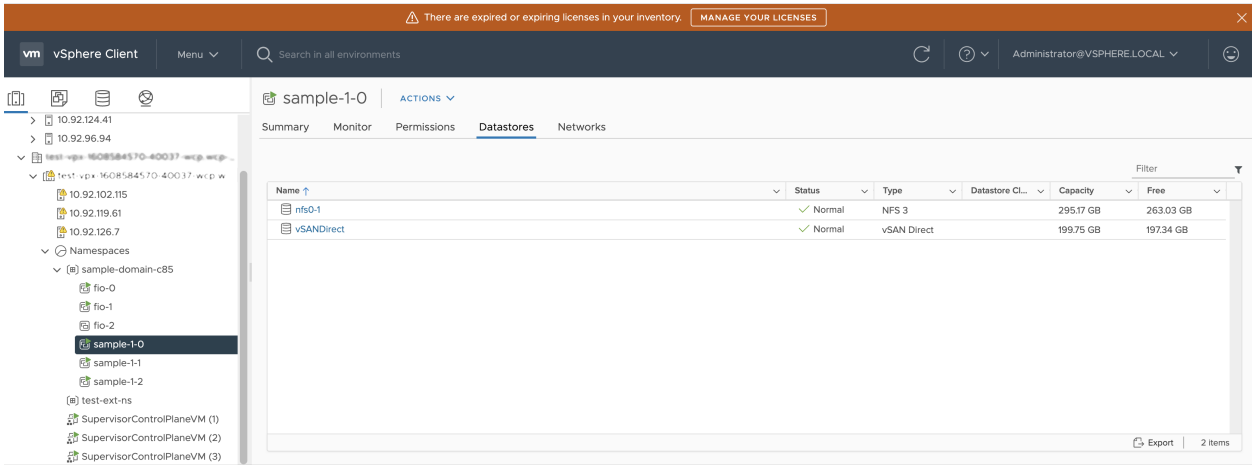
   자세한 내용은 vSAN Direct 스토리지 정책 생성 항목을 참조하십시오.

6 워크로드 관리를 구성하고 사용하도록 설정합니다.

   자세한 내용은 장 5 감독자 클러스터 구성 및 관리 항목을 참조하십시오.

예

이 예에서는 설정에 외부 NFS 스토리지와 vSAN Direct 데이터스토어가 포함됩니다. 제어부 VM 및 vSphere 포드는 외부 NFS 스토리지에서 실행되고 있습니다. 영구 볼륨 클레임은 vSAN Direct에서 실행됩니다.



네트워킹 문제 해결

NSX로 vSphere with Tanzu을 구성할 때 발생할 수 있는 네트워킹 문제를 해결할 수 있습니다.

NSX Manager에 vCenter Server 등록

vCenter Server의 FQDN/PNID가 변경될 때와 같은 특정 상황에서 NSX Manager에 vCenter Server OIDC를 다시 등록해야 할 수 있습니다.

절차

1 SSH를 통해 vCenter Server Appliance에 연결합니다.

2 `shell` 명령을 실행합니다.

3 vCenter Server 지문을 가져오려면 - `openssl s_client -connect <vcenterserver-FQDN:443 </dev/null 2>/dev/null | openssl x509 -fingerprint -sha256 -noout -in /dev/stdin` 명령을 실행합니다.

   지문이 표시됩니다. 예를 들어 `08:77:43:29:E4:D1:6F:29:96:78:5F:BF:D6:45:21:F4:0E:3B:2A:68:05:99:C3:A4:89:8F:F2:0B:EA:3A:BE:9D`입니다.

4 SHA256 지문을 복사하고 콜론을 제거합니다.

   `08774329E4D16F2996785FBFD64521F40E3B2A680599C3A4898FF20BEA3ABE9D`

5 vCenter Server의 OIDC를 업데이트하려면 다음 명령을 실행합니다.

```
curl --location --request POST 'https://<NSX-T_ADDRESS>/api/v1/trust-management/oidc-uris'
\
    --header 'Content-Type: application/json' \
    --header 'Authorization: Basic <AUTH_CODE>' \
    --data-raw '{
 "oidc_type": "vcenter",
```

```
     "oidc_uri": "https://<VC_ADDRESS>/openidconnect/vsphere.local/.well-known/openid-
configuration",
     "thumbprint": "<VC_THUMBPRINT>"
    }'
```

NSX 장치 암호를 변경할 수 없음

root, admin 또는 audit 사용자의 NSX 장치 암호를 변경하지 못할 수 있습니다.

문제

vSphere Client를 통해 root, admin 또는 audit 사용자의 NSX 장치 암호를 변경하려는 시도가 실패할 수 있습니다.

원인

NSX Manager를 설치하는 동안 절차는 세 가지 역할 모두에 대해 하나의 암호만 허용합니다. 나중에 이 암호를 변경하려고 하면 실패할 수 있습니다.

해결책

- 암호를 변경하려면 NSX API를 사용합니다.

  자세한 내용은 https://kb.vmware.com/s/article/70691 및 " NSX-T Data Center 관리 가이드" 의 내용을 참조하십시오.

실패한 워크플로 및 불안정한 NSX Edge 문제 해결

워크플로가 실패하거나 NSX Edge가 불안정한 경우 문제 해결 단계를 수행할 수 있습니다.

문제

vSphere Client에서 분산 포트 그룹 구성을 변경하면 워크플로가 실패하고 NSX Edge가 불안정해질 수 있습니다.

원인

클러스터 구성의 NSX Edge 클러스터 설정 동안 생성된 오버레이 및 업링크에 대한 분산 포트 그룹의 제거 또는 수정은 설계상 허용되지 않습니다.

해결책

NSX Edge의 VLAN 또는 IP 풀 구성을 변경해야 하는 경우 먼저 NSX-T Data Center의 요소 및 vSphere with Tanzu 구성 요소를 클러스터에서 제거해야 합니다.

NSX-T Data Center의 요소 제거에 대한 자세한 내용은 " NSX-T Data Center 설치 가이드" 의 내용을 참조하십시오.

NSX-T 문제 해결을 위한 지원 번들 수집

문제 해결을 위해 등록된 클러스터 및 패브릭 노드에 대한 지원 번들을 수집하고 번들을 시스템에 다운로드하거나 파일 서버에 업로드할 수 있습니다.

번들을 시스템에 다운로드하도록 선택하면, 각 노드에 대한 지원 번들 및 매니페스트 파일로 구성된 단일 아카이브 파일이 제공됩니다. 번들을 파일 서버에 업로드하도록 선택하면 매니페스트 파일과 개별 번들이 파일 서버에 별도로 업로드됩니다.

절차

1 브라우저에서 관리자 권한으로 NSX Manager에 로그인합니다.

2 **시스템 > 지원 번들**을 선택합니다.

3 대상 노드를 선택합니다.

사용 가능한 노드 유형은 **관리 노드**, **Edge**, **호스트** 및 **공용 클라우드 게이트웨이**입니다.

4 (선택 사항) 로그 보존 기간을 일 단위로 지정하면 지정된 일 수보다 오래된 로그를 제외할 수 있습니다.

5 (선택 사항) 코어 파일 및 감사 로그를 포함할지 아니면 제외할지를 나타내는 스위치를 전환합니다.

참고 코어 파일 및 감사 로그에는 암호나 암호화 키와 같은 중요한 정보가 포함될 수 있습니다.

6 (선택 사항) 파일 서버에 번들을 업로드하려면 이 확인란을 선택합니다.

7 지원 번들 수집을 시작하려면 **번들 수집 시작**을 클릭합니다.

각 노드의 로그 파일 수에 따라 지원 번들을 수집하는 데 걸리는 시간이 결정됩니다.

8 수집 프로세스 상태를 모니터링합니다.

상태 탭에 지원 번들 수집에 대한 진행률이 표시됩니다.

9 번들을 파일 서버로 전송하는 옵션이 설정되지 않은 경우 **다운로드**를 클릭하여 번들을 다운로드합니다.

NSX-T에 대한 로그 파일 수집

vSphere with Tanzu 및 NSX-T Data Center 구성 요소에 있는 로그를 수집하여 오류를 감지하고 문제를 해결할 수 있습니다. 로그 파일은 VMware 지원에서 요청할 수 있습니다.

절차

1 vSphere Client를 사용하여 vCenter Server에 로그인합니다.

2 다음 로그 파일을 수집합니다.

로그 파일	설명
/var/log/vmware/wcp/wcpsvc.log	vSphere with Tanzu 사용 설정과 관련된 정보를 포함합니다.
/var/log/vmware/wcp/nsxd.log	NSX-T Data Center 구성 요소 구성과 관련된 정보를 포함합니다.

3 NSX Manager에 로그인합니다.

4 특정 vSphere with Tanzu 작업이 실패하는 경우 NSX Manager가 반환하는 오류에 대한 정보는 `/var/log/proton/nsxapi.log`를 수집합니다.

NSX-T 관리 인증서, 지문 또는 IP 주소가 변경되면 WCP 서비스 다시 시작

vSphere with Tanzu를 설치한 후 NSX-T 관리 인증서, 지문 또는 IP 주소가 변경되면 `WCP` 서비스를 다시 시작해야 합니다.

NSX-T 인증서가 변경되면 vSphere with Tanzu 서비스 다시 시작

현재 vSphere with Tanzu에서는 NSX-T 인증서, 지문 또는 NSX-T IP 주소가 변경되면 `WCP` 서비스를 다시 시작해야 변경 사항이 적용됩니다. 변경되었는데도 서비스를 다시 시작하지 않으면 vSphere with Tanzu와 NSX-T 간의 통신이 실패하고 NCP가 CrashLoopBackoff 단계로 진입하거나 감독자 클러스터 리소스가 배포 불가 상태가 되는 등의 특정 증상이 발생할 수 있습니다.

`WCP` 서비스를 다시 시작하려면 `vmon-cli`를 사용합니다.

1 SSH를 사용하여 vCenter Server에 연결하고 루트 사용자로 로그인합니다.

2 `shell` 명령을 실행합니다.

3 `vmon-cli -h` 명령을 실행하여 사용법 구문 및 옵션을 확인합니다.

4 `vmon-cli -l` 명령을 실행하여 `wcp` 프로세스를 봅니다.

목록 맨 아래에 `wcp`서비스가 표시됩니다.

5 `vmon-cli --restart wcp` 명령을 실행하여 `wcp` 서비스를 다시 시작합니다.

`Completed Restart service request` 메시지가 표시됩니다.

6 `vmon-cli -s wcp` 명령을 실행하고 `wcp` 서비스가 시작되었는지 확인합니다.

예:

```
root@localhost [ ~ ]# vmon-cli -s wcp
Name: wcp
Starttype: AUTOMATIC
RunState: STARTED
RunAsUser: root
```

```
CurrentRunStateDuration(ms): 22158
HealthState: HEALTHY
FailStop: N/A
MainProcessId: 34372
```

호스트 전송 노드 트래픽에 필요한 VDS

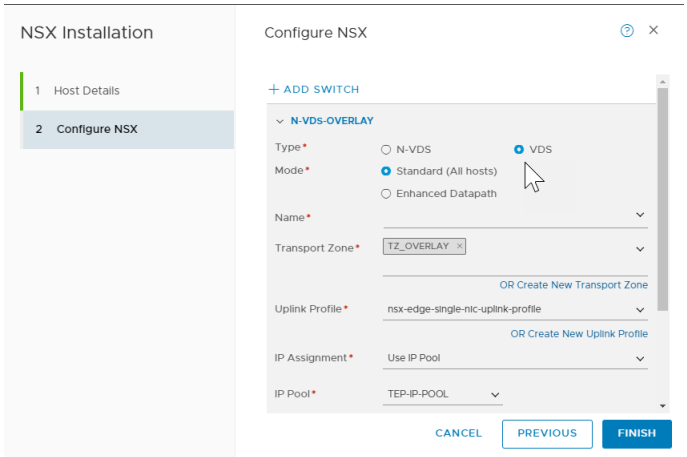
vSphere with Tanzu에는 호스트 전송 노드 트래픽에 대해 vSphere 7 VDS(가상 Distributed Switch)를 사용해야 합니다. vSphere with Tanzu에서는 호스트 전송 노드 트래픽에 대해 N-VDS(NSX-T VDS)를 사용할 수 없습니다.

VDS가 필요함

vSphere with Tanzu에는 동일한 VDS에서 vSphere 7 트래픽과 NSX-T 3 트래픽을 모두 지원하는 Converged VDS가 필요합니다. vSphere 및 NSX-T의 이전 릴리스에는 vSphere 트래픽을 위한 VDS(또는 VSS) 하나와 NSX-T 트래픽을 위한 N-VDS 하나가 있습니다. 이 구성은 vSphere with Tanzu에서 지원되지 않습니다. N-VDS를 사용하여 워크로드 관리를 사용하도록 설정하려고 하면 시스템에서 vCenter 클러스터가 호환되지 않는 것으로 보고합니다. 자세한 내용은 워크로드 관리 사용 설정 클러스터 호환성 오류 문제 해결의 내용을 참조하십시오.

Converged VDS를 사용하려면 vCenter를 사용하여 vSphere 7 VDS를 생성하고 NSX-T에서 ESXi 호스트를 전송 노드로 준비할 때 이 VDS를 지정합니다. vCenter 측에 VDS-DSwitch를 포함하는 것만으로는 충분하지 않습니다. VDS-DSwitch 7.0은 전송 노드 프로파일 생성 항목에 설명되고 아래에 표시된 대로 NSX-T 전송 노드 프로파일로 구성해야 합니다.

그림 19-1. NSX-T의 VDS 구성



이전 버전에서 vSphere 7 및 NSX-T 3으로 업그레이드한 경우에는 각 ESXi 전송 노드에서 N-VDS를 제거하고 각 호스트를 VDS로 재구성해야 합니다. 지침을 보려면 VMware 글로벌 지원 서비스에 문의하십시오.

NSX Advanced Load Balancer 문제 해결

지원 번들을 수집하여 NSX Advanced Load Balancer 문제가 발생할 경우 해결할 수 있습니다.

문제 해결을 위한 지원 번들 수집

NSX Advanced Load Balancer 문제를 해결하기 위해 지원 번들을 수집할 수 있습니다. VMware 지원에서 지원 번들을 요청할 수도 있습니다.

지원 번들을 생성하면 디버그 로그에 대한 단일 파일이 생성되어 다운로드할 수 있습니다.

절차

1 [Avi 컨트롤러] 대시보드에서 왼쪽 상단 모서리에 있는 메뉴를 클릭하고 **관리**를 선택합니다.

2 **관리** 섹션에서 **시스템**을 선택합니다.

3 **시스템** 화면에서 **기술 지원**을 선택합니다.

4 진단 번들을 생성하려면 **기술 지원 생성**을 클릭합니다.

5 **기술 지원 생성** 창에서 **디버그 로그** 유형을 선택하고 **생성**을 클릭합니다.

6 번들이 생성되면 다운로드 아이콘을 클릭하여 시스템에 다운로드합니다.

네트워크 토폴로지 업그레이드 문제 해결

vSphere with Tanzu 버전 7.0 업데이트 1c를 설치하거나 감독자 클러스터를 버전 7.0 업데이트 1에서 버전 7.0 업데이트 1c로 업그레이드하는 경우 네트워킹 토폴로지는 단일 Tier-1 게이트웨이 토폴로지에서 감독자 클러스터 내의 각 네임스페이스에 대한 Tier-1 게이트웨이가 있는 토폴로지로 업그레이드됩니다.

업그레이드 중에 발생할 수 있는 문제를 해결할 수 있습니다.

Edge 로드 밸런서 용량이 부족하여 업그레이드 사전 검사가 실패함

업그레이드 사전 검사가 실패하고 로드 밸런서 용량이 부족함을 나타내는 오류 메시지가 표시됩니다.

문제

업그레이드 사전 검사 프로세스가 실패하고 로드 밸런서 용량이 감독자 클러스터에 필요한 용량 보다 적다는 오류 메시지가 표시됩니다.

해결책

이 문제를 해결하려면 다음 단계 중 하나를 수행하십시오.

- 오류 메시지에서 **강제 업그레이드** 버튼을 클릭하거나 vCenter Server 명령줄을 `--ingnore-precheck-warnings true` 플래그와 함께 사용하여 강제로 업그레이드합니다.

  참고 이 솔루션은 Edge 클러스터가 기존 네임스페이스 워크로드를 지원할 수 있는 경우에만 권장됩니다. 그렇지 않으면 업그레이드 중에 해당 워크로드를 건너뛸 수 있습니다.

- 사용되지 않는 워크로드를 삭제합니다.
- 클러스터에 Edge 노드를 더 추가합니다.

업그레이드 중에 감독자 클러스터 워크로드 네임스페이스를 건너뜀

감독자 클러스터 업그레이드 중에 일부 네임스페이스 워크로드가 업그레이드되지 않습니다.

문제

감독자 클러스터 업그레이드는 성공하지만 업그레이드 중에 일부 네임스페이스 워크로드를 건너뜁니다. Kubernetes 리소스는 리소스가 부족하고 새로 생성된 Tier-1 게이트웨이가 ERROR 상태임을 나타냅니다.

원인

로드 밸런서 용량이 부족하여 워크로드를 지원할 수 없습니다.

해결책

이 문제를 해결하려면 다음 단계 중 하나를 수행하십시오.

- 사용되지 않는 워크로드를 삭제하고 NCP를 다시 시작한 후 업그레이드를 다시 실행합니다.
- 클러스터에 Edge 노드를 더 추가하고 Tier-1 게이트웨이에 대한 재할당을 트리거합니다. NCP를 다시 시작하고 업그레이드를 다시 실행합니다.

업그레이드하는 동안 로드 밸런서 서비스를 건너뜀

감독자 클러스터를 업그레이드하는 동안 일부 로드 밸런서 서비스가 업그레이드되지 않습니다.

문제

감독자 클러스터 업그레이드는 성공하지만 업그레이드 중에 일부 Kubernetes 로드 밸런서 서비스를 건너뜁니다.

원인

감독자 클러스터 워크로드 및 연결된 Tanzu Kubernetes 클러스터의 Kubernetes 로드 밸런서 유형 서비스 수가 NSX Edge 가상 서버 제한을 초과합니다.

해결책

사용되지 않는 워크로드를 삭제하고 NCP를 다시 시작한 후 업그레이드를 다시 실행합니다.

Tanzu Kubernetes 클러스터 문제 해결

Tanzu Kubernetes 클러스터 문제는 클러스터 노드에 연결하여 클러스터 리소스 계층을 살펴보고 로그 파일을 수집하여 해결할 수 있습니다.

Tanzu Kubernetes 클러스터에 대한 지원 번들 수집

Tanzu Kubernetes 클러스터 오류 문제를 해결하기 위해 유틸리티를 실행하여 진단 로그 번들을 수집할 수 있습니다.

VMware는 TKC 지원 번들러 유틸리티를 제공합니다. 이 유틸리티는 Tanzu Kubernetes 클러스터 로그 파일을 수집하고 문제를 해결하는 데 사용할 수 있습니다.

유틸리티를 가져와서 사용하려면 VMware 지원 기술 자료에서 Tanzu Kubernetes클러스터에서 진단 로그 번들을 수집하는 방법(80949) 문서를 참조하십시오.

vCenter Single Sign-On 연결 오류 문제 해결

vSphere 네임스페이스에 대한 충분한 사용 권한이 없으면 감독자 클러스터 또는 Tanzu Kubernetes 클러스터에 vCenter Single Sign-On 사용자로 연결할 수 없습니다.

문제

vCenter Single Sign-On 사용자로 감독자 클러스터 또는 Tanzu Kubernetes 클러스터에 연결하려고 하면 kubectl용 vSphere 플러그인에서 `Error from server (Forbidden)` 오류 메시지를 반환합니다.

원인

vSphere 네임스페이스에 대한 충분한 사용 권한이 없거나 클러스터 액세스 권한이 없습니다.

해결책

클러스터를 운영하는 DevOps 엔지니어인 경우 vSphere 네임스페이스에 대한 **편집** 권한이 부여되었는지 vSphere 관리자에게 확인하십시오. vSphere 네임스페이스 생성 및 구성의 내용을 참조하십시오.

클러스터를 사용하여 워크로드를 배포하는 개발자인 경우 클러스터 액세스 권한이 부여되었는지 관리자에게 확인하십시오. 개발자에게 Tanzu Kubernetes 클러스터에 대한 액세스 권한 부여의 내용을 참조하십시오.

구독 컨텐츠 라이브러리 오류 문제 해결

구독 컨텐츠 라이브러리가 스토리지 용량 제한에 도달하면 Tanzu Kubernetes 클러스터를 프로비저닝할 수 없습니다.

문제

Tanzu Kubernetes 클러스터를 프로비저닝하려고 하면 구독 컨텐츠 라이브러리에서 항목을 끌어올 수 없고 다음 오류가 표시됩니다.

```
Internal error occurred: get library items failed for.
```

원인

구독 컨텐츠 라이브러리가 용량에 도달했습니다. 컨텐츠 라이브러리는 연결된 스토리지를 통해 지원됩니다. 시간이 경과하면 Kubernetes 버전이 더 많이 릴리스되고 OVA 파일을 라이브러리로 가져오기 때문에 스토리지 용량이 가득 찰 수 있습니다.

해결책

새 컨텐츠 라이브러리로 마이그레이션합니다. Tanzu Kubernetes 클러스터를 새 컨텐츠 라이브러리로 마이그레이션의 내용을 참조하십시오.

로컬 컨텐츠 라이브러리 오류 문제 해결

일반적인 로컬 컨텐츠 라이브러리 오류를 해결하려면 이 항목을 참조하십시오.

감독자 클러스터에서 TKR을 찾을 수 없음

로컬 컨텐츠 라이브러리는 인터넷이 제한된 환경에서 사용할 수 있습니다. 로컬 컨텐츠 라이브러리를 생성하려면 Tanzu Kubernetes 릴리스용 로컬 컨텐츠 라이브러리 생성, 보안 및 동기화 항목을 참조하십시오.

컨텐츠 라이브러리에 보안 정책을 적용할 때 다음 조건 중 하나에 해당하는 경우 감독자 클러스터의 TKR(Tanzu Kubernetes 릴리스)이 표시되지 않습니다.

- 컨텐츠 라이브러리의 OVF 패키지가 서명되지 않았습니다.
- OVF 패키지가 잘못된 인증서를 사용하여 서명되었습니다.
- OVF 패키지가 로컬 컨텐츠 라이브러리가 구성된 vCenter Server에서 신뢰하지 않는 인증서를 사용하여 서명되었습니다.

클러스터 프로비저닝 오류 문제 해결

Tanzu Kubernetes 클러스터를 프로비저닝했는데 제어부 가상 시스템이 시작되지 않으면 가상 시스템 크기 또는 클래스를 변경해야 할 수도 있습니다.

문제

Tanzu Kubernetes 클러스터를 프로비저닝했습니다. 시스템에서 제어부 가상 시스템의 전원을 켜려고 시도하는데, 오류가 발생하고 다음 메시지가 표시됩니다.

```
The host does not have sufficient CPU resources to satisfy the reservation.
```

원인

가상 시스템 크기 또는 클래스가 클러스터 배포에 충분하지 않습니다.

해결책

가상 시스템 유형 또는 클래스를 변경합니다. Tanzu Kubernetes 클러스터에 대한 가상 시스템 클래스의 내용을 참조하십시오.

워크로드 배포 오류 문제 해결

PodSecurityPolicy 및 바인딩이 인증된 사용자에 대해 구성되지 않은 경우 워크로드 배포 오류가 발생할 수 있습니다.

문제

Tanzu Kubernetes 클러스터에 컨테이너 워크로드를 배포했지만 워크로드가 시작되지 않습니다. 다음과 유사한 오류가 표시됩니다.

```
Error: container has runAsNonRoot and image will run as root.
```

원인

Tanzu Kubernetes 클러스터는 PodSecurityPolicy 승인 컨트롤러를 사용하도록 설정된 상태로 프로비저닝됩니다. 클러스터 관리자가 PodSecurityPolicy를 인증된 사용자에게 바인딩할 때까지 인증된 사용자는 권한이 있거나 없는 포드를 생성할 수 없습니다.

해결책

기본 PodSecurityPolicy에 대한 적절한 바인딩을 생성하거나 사용자 지정 PodSecurityPolicy를 정의합니다. Tanzu Kubernetes 클러스터에서 포드 보안 정책 사용 및 Tanzu Kubernetes 방명록 자습서의 내용을 참조하십시오.

가상 시스템 클래스 오류 문제 해결

가상 시스템 클래스는 Tanzu Kubernetes 클러스터 및 VM 클래스의 vSphere 네임스페이스에 바인딩되어야 합니다.

가상 시스템 클래스 바인딩 오류

가상 시스템 클래스는 vSphere 네임스페이스에 바인딩되어야 합니다. vSphere with Tanzu에서 VM 클래스를 네임스페이스와 연결의 내용을 참조하십시오. VM 클래스를 네임스페이스와 연결하지 않으면 `VirtualMachineClassBindingNotFound` 오류가 표시되며, 다음과 유사합니다.

```
conditions:
  - lastTransitionTime: "2021-04-25T02:50:58Z"
    message: 1 of 2 completed
    reason: VirtualMachineClassBindingNotFound @ Machine/test-cluster
    severity: Error
    status: "False"
    type: ControlPlaneReady
  - lastTransitionTime: "2021-04-25T02:49:21Z"
    message: 0/1 Control Plane Node(s) healthy. 0/2 Worker Node(s) healthy
    reason: WaitingForNodesHealthy
    severity: Info
    status: "False"
    type: NodesHealthy
```

실패한 Tanzu Kubernetes 클러스터 업데이트 작업 다시 시작

Tanzu Kubernetes 클러스터의 업데이트가 실패한 경우 업데이트 작업을 다시 시작하고 업데이트를 다시 시도할 수 있습니다.

문제

Tanzu Kubernetes 클러스터의 업데이트가 실패하고 클러스터 상태가 `upgradefailed`가 됩니다.

원인

클러스터 업데이트 실패에는 스토리지 부족과 같은 여러 가지 이유가 있을 수 있습니다. 실패한 업데이트 작업을 다시 시작하고 클러스터 업데이트를 다시 시도하려면 다음 절차를 완료하십시오.

해결책

1 감독자 클러스터에 관리자로 로그인합니다. vCenter Single Sign-On 사용자로 감독자 클러스터에 연결의 내용을 참조하십시오.

2 `update_job_name`을 조회합니다.

```
kubectl get jobs -n vmware-system-tkg -l "run.tanzu.vmware.com/cluster-namespace=$
{cluster_namespace},cluster.x-k8s.io/cluster-name=${cluster_name}"
```

3 `curl`을 사용하여 요청을 실행할 수 있도록 `kubectl proxy`를 실행합니다.

```
kubectl proxy &
```

`Starting to serve on 127.0.0.1:8001`이 표시되어야 합니다.

참고 `kubectl`을 사용하여 리소스의 `.status`를 패치하거나 업데이트할 수 없습니다.

4 `curl`을 사용하면 다음 패치 명령을 실행하여 `.spec.backoffLimit`을 높입니다.

```
curl -H "Accept: application/json" -H "Content-Type: application/json-patch+json"
--request PATCH --data '[{"op": "replace", "path": "/spec/backoffLimit", "value": 8}]'
http://127.0.0.1:8001/apis/batch/v1/namespaces/vmware-system-tkg/jobs/${update_job_name}
```

5 `curl`을 사용하면 다음 패치 명령을 실행하여 작업 컨트롤러가 새 포드를 생성하도록 `.status.conditions`를 지웁니다.

```
$ curl -H "Accept: application/json" -H "Content-Type: application/json-patch+json"
--request PATCH --data '[{"op": "remove", "path": "/status/conditions"}]'
http://127.0.0.1:8001/apis/batch/v1/namespaces/vmware-system-tkg/jobs/${update_job_name}/
status
```

워크로드 관리 문제 해결

워크로드 관리 문제 해결을 위해 또는 지원 번들을 수집해야 하는 경우 이 섹션을 참조하십시오.

워크로드 관리를 위한 지원 번들 수집

다음 절차에 따라 워크로드 관리를 위한 지원 번들을 수집합니다.

다음 작업을 완료하여 감독자 클러스터에 대한 로그를 가져옵니다. 이 작업은 클러스터가 오류 또는 구성 중 상태에 있는 경우에도 수행할 수 있습니다.

절차

1 vSphere Client를 사용하여 vSphere with Tanzu 환경에 로그인합니다.

2 **메뉴 > 워크로드 관리**를 선택합니다.

3 **클러스터** 탭을 선택합니다.

4 대상 감독자 클러스터를 선택합니다.

5 **로그 내보내기**를 선택합니다.

결과

지원 번들을 수집한 후에는 KB 문서: 보안 FTP 포털을 통해 VMware에 대한 진단 정보 업로드(http://kb.vmware.com/kb/2069559)를 참조하십시오.

워크로드 관리 로그 파일에 tail 명령 사용

워크로드 관리 로그 파일에 tail 명령을 사용하면 사용 설정 문제 및 감독자 클러스터 배포 오류를 해결하는 데 유용할 수 있습니다.

해결책

1 vCenter Server Appliance에 대한 SSH 연결을 설정합니다.

2 `root` 사용자로 로그인합니다.

3 `shell` 명령을 실행합니다.

다음 내용이 보입니다

```
Shell access is granted to root
root@localhost [ ~ ]#
```

4 다음 명령을 실행하여 로그를 살펴봅니다.

```
tail -f /var/log/vmware/wcp/wcpsvc.log
```

워크로드 관리 사용 설정 클러스터 호환성 오류 문제 해결

vSphere 클러스터가 호환되지 않아서 워크로드 관리를 사용하도록 설정할 수 없다고 시스템에 표시되면, 다음 문제 해결 팁을 따르십시오.

문제

워크로드 관리를 사용하도록 설정하려고 하면 **워크로드 관리** 페이지에 vCenter 클러스터가 호환되지 않는다고 표시됩니다.

원인

여기에는 여러 가지 원인이 있을 수 있습니다. 먼저 환경이 워크로드 관리를 사용하도록 설정하기 위한 최소 요구 사항을 충족하는지 확인합니다.

- 유효한 라이센스: VMware vSphere 7 Enterprise Plus with Add-on for Kubernetes
- 둘 이상의 ESXi 호스트
- 완전히 자동화된 DRS
- vSphere HA
- vSphere Distributed Switch 7.0
- 충분한 스토리지 용량

환경이 이러한 사전 요구 사항을 충족해도 대상 vCenter 클러스터가 호환되지 않는 경우에는 VMware DCLI(Datacenter CLI)를 사용하여 문제를 식별합니다.

해결책

1. vCenter Server에 SSH를 실행합니다.
2. 루트 사용자로 로그인합니다.
3. `dcli` 명령을 실행하여 VMware Datacenter CLI 도움말을 나열합니다.
4. 다음 DCLI 명령을 실행하여 사용 가능한 vCenter 클러스터를 나열합니다.

   ```
   dcli com vmware vcenter cluster list
   ```

   예:

   ```
   dcli +username VI-관리자-사용자-이름 +password VI-관리자-암호 com vmware vcenter cluster list
   ```

   예제 결과:

   ```
   |-----------|---------|------------|----------|
   |drs_enabled|cluster  |name        |ha_enabled|
   |-----------|---------|------------|----------|
   |True       |domain-d7|vSAN Cluster|True      |
   |-----------|---------|------------|----------|
   ```

5. 다음 DCLI 명령을 실행하여 vCenter 클러스터 호환성을 확인합니다.

   ```
   dcli com vmware vcenter namespacemanagement clustercompatibility list
   ```

   예:

   ```
   dcli +username VI-관리자-사용자-이름 +password VI-관리자-암호 com vmware vcenter
   namespacemanagement clustercompatibility list
   ```

다음 예제 결과는 호환되는 NSX-T VDS 스위치가 환경에 없다는 것을 나타냅니다.

```
|---------|----------|-----------------------------------------------------------------------
------------------|
|cluster  |compatible|
incompatibility_reasons                                                                 |
|---------|----------|-----------------------------------------------------------------------
------------------|
|domain-d7|False     |Failed to list all distributed switches in vCenter 2b1c1fa5-
e9d4-45d7-824c-fa4176da96b8.|
|         |          |Cluster domain-d7 is missing compatible NSX-T
VDS.                                    |
|---------|----------|-----------------------------------------------------------------------
------------------|
```

6 추가 호환성 문제를 확인하려면 필요에 따라 추가 DCLI 명령을 실행합니다. NSX-T 오류 외에 호환되지 않는 다른 일반적인 이유는 DNS와 NTP 연결 문제입니다.

7 추가로 문제를 해결하려면 다음 단계를 완료하십시오.

a `wcpsvc.log` 파일에 tail 명령을 사용합니다. 워크로드 관리 로그 파일에 tail 명령 사용의 내용을 참조하십시오.

b **워크로드 관리** 페이지로 이동하여 **사용**을 클릭합니다.

vSphere with Tanzu 워크로드 도메인 종료 및 시작

데이터 손실을 방지하고 vSphere with Tanzu 환경의 구성 요소 및 워크로드를 작동 상태로 유지하려면 구성 요소를 종료하거나 시작할 때 지정된 순서를 따라야 합니다.

일반적으로 종료 및 시작 작업은 패치를 적용하거나, 업그레이드하거나, vSphere with Tanzu 환경을 복원한 후에 수행합니다.

Tanzu Kubernetes Grid 서비스에서 프로비저닝된 Tanzu Kubernetes 클러스터를 포함한 vSphere with Tanzu 솔루션은 vSphere SDDC(소프트웨어 정의 데이터 센터)의 일부입니다. 따라서 vSphere with Tanzu 환경의 종료 및 시작을 수행할 때는 전체 vSphere 인프라 스택을 고려해야 합니다. vSphere with Tanzu를 포함한 vSphere SDDC의 종료 및 시작에 대해 검증된 다음과 같은 일련의 절차를 참조하십시오.

- vSphere with Tanzu를 포함한 vSphere SDDC 종료 절차
- vSphere with Tanzu를 포함한 vSphere SDDC 시작 절차