



# OpenShift Container Platform 4.9

## OpenShift용 Windows 컨테이너 지원

Windows 컨테이너용 Red Hat OpenShift 가이드



# OpenShift Container Platform 4.9 OpenShift용 Windows 컨테이너 지원

---

Windows 컨테이너용 Red Hat OpenShift 가이드

## 법적 공지

Copyright © 2023 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux<sup>®</sup> is the registered trademark of Linus Torvalds in the United States and other countries.

Java<sup>®</sup> is a registered trademark of Oracle and/or its affiliates.

XFS<sup>®</sup> is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL<sup>®</sup> is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js<sup>®</sup> is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack<sup>®</sup> Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

## 초록

Windows 컨테이너용 Red Hat OpenShift는 OpenShift Container Platform에서 Microsoft Windows Server 컨테이너를 실행하기 위한 기본 지원을 제공합니다. 이 가이드에서는 모든 세부 정보를 제공합니다.

<b>차례</b>	
<b>1장. WINDOWS 컨테이너용 RED HAT OPENSIFT 지원 개요</b> .....	<b>3</b>
<b>2장. RED HAT OPENSIFT 릴리스 노트를 위한 WINDOWS 컨테이너 지원</b> .....	<b>4</b>
2.1. RED HAT OPENSIFT를 위한 WINDOWS 컨테이너 지원 정보	4
2.2. 지원 요청	4
2.3. RED HAT WINDOWS MACHINE CONFIG OPERATOR 4.0.2 릴리스 노트	4
2.4. RED HAT WINDOWS MACHINE CONFIG OPERATOR 4.0.1 릴리스 노트	4
2.5. RED HAT WINDOWS MACHINE CONFIG OPERATOR 4.0.0 릴리스 노트	5
2.6. WINDOWS MACHINE CONFIG OPERATOR 전제 조건	6
2.7. 알려진 제한 사항	8
<b>3장. WINDOWS 컨테이너 워크로드 이해</b> .....	<b>10</b>
3.1. WINDOWS MACHINE CONFIG OPERATOR 전제 조건	10
3.2. WINDOWS 워크로드 관리	12
3.3. WINDOWS 노드 서비스	14
3.4. 알려진 제한 사항	14
<b>4장. WINDOWS 컨테이너 워크로드 활성화</b> .....	<b>16</b>
사전 요구 사항	16
4.1. WINDOWS MACHINE CONFIG OPERATOR 설치	16
4.2. WINDOWS MACHINE CONFIG OPERATOR에 대한 시크릿 구성	19
4.3. 추가 리소스	20
<b>5장. WINDOWS MACHINESSET 오브젝트 생성</b> .....	<b>21</b>
5.1. AWS에서 WINDOWS MACHINESSET 오브젝트 생성	21
5.2. AZURE에서 WINDOWS MACHINESSET 오브젝트 만들기	26
5.3. VSPHERE에서 WINDOWS MACHINESSET 오브젝트 생성	33
<b>6장. WINDOWS 컨테이너 워크로드 예약</b> .....	<b>47</b>
사전 요구 사항	47
6.1. WINDOWS POD 배치	47
6.2. 스케줄링 메커니즘을 캡슐화하기 위해 RUNTIMECLASS 오브젝트 생성	48
6.3. 샘플 WINDOWS 컨테이너 워크로드 배포	50
6.4. 머신 세트 수동 스케일링	51
<b>7장. WINDOWS 노드 업그레이드</b> .....	<b>54</b>
7.1. WINDOWS MACHINE CONFIG OPERATOR 업그레이드	54
<b>8장. BYOH (BRING-YOUR-OWN-HOST) WINDOWS 인스턴스를 노드로 사용</b> .....	<b>55</b>
8.1. BYOH WINDOWS 인스턴스 구성	55
8.2. BYOH WINDOWS 인스턴스 제거	57
<b>9장. WINDOWS 노드 제거</b> .....	<b>58</b>
9.1. 특정 머신 삭제	58
<b>10장. WINDOWS 컨테이너 워크로드 비활성화</b> .....	<b>60</b>
10.1. WINDOWS MACHINE CONFIG OPERATOR 제거	60
10.2. WINDOWS MACHINE CONFIG OPERATOR 네임스페이스 삭제	60



## 1장. WINDOWS 컨테이너용 RED HAT OPENSIFT 지원 개요

Windows 컨테이너에 대한 Red Hat OpenShift 지원은 OpenShift Container Platform 클러스터에서 Windows 컴퓨팅 노드를 실행하는 기능을 제공하는 기능입니다. 이는 Red Hat WMCO(Windows Machine Config Operator)를 사용하여 Windows 노드를 설치 및 관리할 수 있습니다. Red Hat 서브스크립션을 통해 OpenShift Container Platform에서 Windows 워크로드를 실행할 수 있습니다. 자세한 내용은 [릴리스 노트](#)를 참조하십시오.

Linux 및 Windows를 비롯한 워크로드의 경우 OpenShift Container Platform을 사용하면 Windows Server 컨테이너에서 실행되는 Windows 워크로드를 배포할 수 있으며 RHCOS(Red Hat Enterprise Linux CoreOS) 또는 RHEL(Red Hat Enterprise Linux)에서 호스팅되는 기존 Linux 워크로드도 제공할 수 있습니다. 자세한 내용은 [Windows 컨테이너 워크로드 시작하기](#)를 참조하십시오.

클러스터에서 Windows 워크로드를 실행하려면 WMCO가 필요합니다. WMCO는 클러스터에서의 Windows 워크로드 배포 및 관리 프로세스를 오케스트레이션합니다. 자세한 내용은 [Windows 컨테이너 워크로드를 활성화하는 방법](#)을 참조하십시오.

지원되는 Windows 워크로드를 새 Windows 머신으로 이동할 수 있도록 Windows **MachineSet** 오브젝트를 생성하여 인프라 Windows 머신 세트 및 관련 머신을 생성할 수 있습니다. 여러 플랫폼에서 Windows **MachineSet** 오브젝트를 생성할 수 있습니다.

[Windows 워크로드를 Windows 컴퓨팅 노드에 예약](#)할 수 있습니다.

[Windows Machine Config Operator 업그레이드를 수행](#)하여 Windows 노드에 최신 업데이트가 있는지 확인할 수 있습니다.

특정 머신을 삭제하여 [Windows 노드를 제거](#)할 수 있습니다.

[Bring-Your-Own-Host\(BYOH\) Windows 인스턴스를 사용하여 Windows Server VM](#)을 용도 변경 및 OpenShift Container Platform으로 가져올 수 있습니다. BYOH Windows 인스턴스는 Windows 서버가 오프라인 상태가 되는 경우 주요 중단을 완화하려는 사용자에게 도움이 됩니다. BYOH Windows 인스턴스를 OpenShift Container Platform 4.8 이상 버전에서 노드로 사용할 수 있습니다.

다음을 수행하여 [Windows 컨테이너 워크로드를 비활성화](#)할 수 있습니다.

- Windows Machine Config Operator 제거
- Windows Machine Config Operator 네임스페이스 삭제

## 2장. RED HAT OPENSIFT 릴리스 노트를 위한 WINDOWS 컨테이너 지원

### 2.1. RED HAT OPENSIFT를 위한 WINDOWS 컨테이너 지원 정보

Windows 컨테이너에 대한 Red Hat OpenShift 지원을 사용하면 OpenShift Container Platform 클러스터에서 Windows 컴퓨팅 노드를 실행할 수 있습니다. Red Hat WMCO(Windows Machine Config Operator)를 사용하여 Windows 노드를 설치 및 관리하면 Windows 워크로드 실행이 가능합니다. 사용할 수 있는 Windows 노드를 통해 OpenShift Container Platform에서 Windows 컨테이너 워크로드를 실행할 수 있습니다.

Windows Containers에 대한 Red Hat OpenShift 지원의 릴리스 노트는 WMCO의 개발을 추적하며, 이는 OpenShift Container Platform의 모든 Windows 컨테이너 워크로드 기능을 제공합니다.

### 2.2. 지원 요청

Windows 컨테이너에 대한 Red Hat OpenShift 지원은 설치 가능한 선택적 구성 요소로 제공됩니다. Windows 컨테이너에 대한 Red Hat OpenShift 지원은 OpenShift Container Platform 서브스크립션의 일부가 아닙니다. 추가 Red Hat 서브스크립션이 필요하며 [적용 범위](#) 및 [서비스 수준 계약](#)에 따라 지원됩니다.

Windows Containers에 대한 Red Hat OpenShift 지원에 대한 지원을 받으려면 이 별도의 서브스크립션이 있어야 합니다. 이러한 추가 Red Hat 서브스크립션이 없으면 프로덕션 클러스터에 Windows 컨테이너 워크로드를 배포할 수 없습니다. [Red Hat 고객 포털](#)을 통해 지원을 요청할 수 있습니다.

자세한 내용은 [Windows 컨테이너용 Red Hat OpenShift 지원에 대한 Red Hat OpenShift Container Platform 라이프 사이클 정책 문서](#)를 참조하십시오.

이러한 추가 Red Hat 서브스크립션이 없는 경우 공식 지원이 없는 배포판인 Community Windows Machine Config Operator를 사용할 수 있습니다.

### 2.3. RED HAT WINDOWS MACHINE CONFIG OPERATOR 4.0.2 릴리스 노트

출시 날짜: 2023-3-28

이제 WMCO 4.0.2를 버그 수정과 함께 사용할 수 있습니다. WMCO의 구성 요소는 [RHBA-2023:1488](#)에서 릴리스되었습니다.

#### 2.3.1. 버그 수정

- 이전에는 Windows 노드의 컨테이너에 기본 서브넷을 기반으로 하드 코딩된 DNS 서버 IP 주소 (예: 172.30.0.10)가 할당되었습니다. 이로 인해 특정 상황에서 DNS 확인이 실패했습니다. 이번 수정을 통해 하드 코딩된 DNS 주소가 제거되고 Windows 컨테이너에 이제 클러스터 DNS 구성을 기반으로 하는 유효한 DNS 서버 IP 주소가 할당됩니다. 결과적으로 Windows 컨테이너에 대해 DNS 확인이 예상대로 작동합니다. ([BZ#2020350](#))

### 2.4. RED HAT WINDOWS MACHINE CONFIG OPERATOR 4.0.1 릴리스 노트

출시 날짜: 2021-12-13



이제 버그 수정으로 WMCO 4.0.1을 사용할 수 있습니다. WMCO의 구성 요소는 [RHBA-2021:4757](#) 로 출시되었습니다.

### 2.4.1. 버그 수정

- 이전에는 **windows-exporter** 지표 끝점 오브젝트에 삭제된 머신에 대한 참조가 포함되어 있었습니다. 이러한 잘못된 참조로 인해 WMCO에서 잘못된 IP 주소가 있는 머신의 삭제된 이벤트를 무시했습니다. 이 버그 수정을 통해 이벤트 필터링에서 머신 오브젝트의 유효성 검사가 제거되어 머신이 여전히 삭제 단계에 있을 때 **windows-exporter** 지표 끝점 오브젝트를 올바르게 업데이트할 수 있습니다. ([BZ#2008992](#))
- 이전 버전에서는 Windows **Machine** 오브젝트와 연결된 노드를 삭제하면 Operator를 다시 시작할 때 조정 오류가 발생했습니다. 이번 버그 수정을 통해 **실행 중** 상태의 Windows 머신에서 참조하는 노드를 클러스터 내에서 찾을 수 없는 경우 반응하거나 조정하지 않으므로 오류 루프를 방지하고 Linux 머신 개체로 기능을 표준화할 수 있습니다. ([BZ#2009474](#))
- 이전에는 Windows VM의 WMCO에서 특정 명령을 PowerShell에서 올바르게 구문 분석하지 않았습니다. 이로 인해 PowerShell을 기본 SSH 셸로 사용하는 Windows VM이 노드로 클러스터에 연결할 수 없었습니다. 이제 WMCO가 Windows VM의 기본 SSH 셸을 식별하고 그에 따라 관련 명령을 실행합니다. 이 새 기능을 사용하면 PowerShell을 사용하는 Windows VM을 클러스터의 노드로 기본 SSH 셸로 구성할 수 있습니다. ([BZ#2014707](#))
- 이전에는 암호화된 사용자 이름이 추가 태그를 사용하여 생성되어 이로 인해 올바르게 표시되지 않았습니다. 이번 버그 수정을 통해 추가 태그가 제거되어 암호화된 사용자 이름이 올바르게 표시됩니다. ([BZ#2016712](#))
- 이전에는 VM이 DNS 오브젝트로 지정될 때 WMCO가 **Node** 오브젝트와 B4H Windows VM을 올바르게 연결하지 않았습니다. 이로 인해 WMCO가 이미 완전히 구성된 VM 구성을 시도했습니다. 이제 연결된 노드를 찾을 때 DNS 주소에서 지정한 VM을 WMCO에서 확인합니다. ([BZ#2020648](#))

## 2.5. RED HAT WINDOWS MACHINE CONFIG OPERATOR 4.0.0 릴리스 노트

이번 WMCO 릴리스에서는 OpenShift Container Platform 클러스터에서 Windows 컴퓨팅 노드를 실행하기 위한 버그 수정을 제공합니다. WMCO 4.0.0의 구성 요소는 [RHBA-2021:3702](#) 로 릴리스되었습니다.

### 2.5.1. 버그 수정

- 이전에는 BMCO(Bull-your-Own-Host) Windows 노드를 생성할 때 WMCO에서 원시 사용자 제공 인스턴스 주소를 사용했습니다. 이로 인해 B4H Windows 인스턴스가 OpenShift Container Platform 클러스터에 참여하지 않았습니다. 이 버그 수정을 통해 사용자가 제공한 DNS 이름이 유효한 IPv4 주소로 확인되고 B4H Windows 인스턴스를 만들 때 확인된 값이 사용됩니다. 이제 호스트 이름이 다르고 DNS 주소를 Windows 노드로 구성할 수 있습니다. ([BZ#1995684](#))
- 이전에는 WMCO가 instance-to-node 연결을 식별할 때 확인되지 않은 인스턴스 주소를 사용하여 직접 비교했습니다. 이로 인해 OpenShift Container Platform 클러스터에 참여하도록 구성된 B4H Windows 인스턴스가 제거되었습니다. 이 버그 수정을 통해 **windows-instances** 구성 맵에 추가된 항목의 DNS 조회를 수행하여 DNS 주소의 유효성을 검사합니다. 이제 WMCO가 구성된 인스턴스 간 관계를 올바르게 식별하여 BMH 노드의 조기 제거를 방지합니다. ([BZ#2005126](#))

### 2.5.2. 확인된 문제

- 웹 콘솔에서 사용 가능한 파일 시스템 그래프는 Windows 노드에 표시되지 않습니다. 이 문제는 파일 시스템 쿼리의 변경으로 인해 발생하며, 이는 향후 WMCO 릴리스에서 해결될 예정입니다. ([BZ#1930347](#))
- VMware vSphere에 설치된 클러스터의 경우 WMCO는 **삭제** 단계 알림 이벤트를 무시하고 **windows-exporter** 지표 끝점에 잘못된 노드 정보를 남겨 둡니다. 이로 인해 Prometheus 지표 끝점에 대한 잘못된 매핑이 발생했습니다. 이 버그가 수정되었습니다. 이제 WMCO가 **삭제** 단계 알림 이벤트를 인식하고 Prometheus 지표 끝점을 적절하게 매핑합니다. ([BZ#1995341](#))
- Linux 기반 포드의 보안 컨텍스트에 **RunAsUser** 권한이 설정된 경우 예상 파일에는 컨테이너 사용자 소유권을 포함하여 올바른 권한이 설정되어 있습니다. 그러나 Windows와 동등한 **RunAsUsername** 권한이 Windows Pod에 설정된 경우 kubelet은 예상 볼륨의 파일에 올바른 소유권을 설정하지 못합니다. 모범 사례가 따르지 않는 **hostPath 볼륨** 과 함께 사용하면 이 문제가 발생할 수 있습니다. 예를 들어 Pod에서 **C:\var\lib\kubelet\pods\pods\** 폴더에 대한 액세스 권한을 부여하면 Pod가 다른 Pod의 서비스 계정 토큰에 액세스할 수 있습니다. 기본적으로 이 예제와 같이 예상된 파일은 다음과 같은 소유권을 갖게 됩니다. Windows 예상 볼륨 파일은 다음과 같습니다.

```

Path :
Microsoft.PowerShell.Core\FileSystem::C:\var\run\secrets\kubernetes.io\serviceaccount\..2021_08_31_22_22_18.318230061\ca.crt
Owner : BUILTIN\Administrators
Group : NT AUTHORITY\SYSTEM
Access : NT AUTHORITY\SYSTEM Allow FullControl
        BUILTIN\Administrators Allow FullControl
        BUILTIN\Users Allow ReadAndExecute, Synchronize
Audit :
Sddl : O:BAG:SYD:AI(A;ID;FA;;;SY)(A;ID;FA;;;BA)(A;ID;0x1200a9;;;BU)
    
```

이는 **ContainerAdministrator** 역할과 마찬가지로 모든 관리자 사용자에게 읽기, 쓰기, 실행 액세스 권한이 있음을 나타내지만 관리자 이외의 사용자는 읽기 및 실행 액세스 권한이 있습니다.



**중요**

OpenShift Container Platform은 운영 체제와 관계없이 **RunAsUser** 보안 컨텍스트를 모든 Pod에 적용합니다. 즉, Windows Pod에는 보안 컨텍스트에 자동으로 **RunAsUser** 권한이 적용됩니다.

또한 기본 **RunAsUser** 권한이 설정된 예상 볼륨으로 Windows Pod가 생성되면 Pod가 **ContainerCreating** 단계에서 중단됩니다.

이러한 문제를 처리하기 위해 OpenShift Container Platform은 Pod의 보안 컨텍스트에 설정된 예상 서비스 계정 볼륨에서 파일 권한을 처리하도록 강제 적용합니다. Windows에서 예상 볼륨에는 적용되지 않습니다([BZ#1971745](#)). Windows Pod에 대한 이 동작은 OpenShift Container Platform 4.7 이전의 모든 Pod 유형에서 작동하는 데 사용되는 파일 권한 처리 방법입니다.

## 2.6. WINDOWS MACHINE CONFIG OPERATOR 전제 조건

다음 정보는 Windows Machine Config Operator에 지원되는 플랫폼 버전, Windows Server 버전 및 네트워킹 구성에 대해 자세히 설명합니다. 해당 플랫폼과 관련된 모든 정보는 vSphere 설명서를 참조하십시오.

### 2.6.1. OpenShift Container Platform 및 WMCO 버전을 기반으로 지원되는 플랫폼

플랫폼	지원되는 OpenShift Container Platform 버전	지원되는 WMCO 버전	설치 프로그램에서 프로비저닝한 인프라 설치 지원	사용자 프로비저닝 인프라 설치 지원
AWS(Amazon Web Services)	4.6+	WMCO 1.0+	GA	
Microsoft Azure	4.6+	WMCO 1.0+	GA	
VMware vSphere	4.7+	WMCO 2.0+	GA	

### 2.6.2. OpenShift Container Platform 및 WMCO 버전을 기반으로 하는 BYOH(Bring-Your-Own-Host) 인스턴스에 지원되는 플랫폼

플랫폼	지원되는 OpenShift Container Platform 버전	지원되는 WMCO 버전	설치 프로그램에서 프로비저닝한 인프라 설치 지원 용 BYOH	사용자 프로비저닝 인프라 설치 지원 용 BYOH
AWS(Amazon Web Services)	4.8+	WMCO 3.1+	GA	
Microsoft Azure	4.8+	WMCO 3.1+	GA	
VMware vSphere	4.8+	WMCO 3.1+	GA	
공급자와 무관함 (Platform: none)	4.8+	WMCO 3.1+		GA

### 2.6.3. 지원되는 Windows Server 버전

다음 표에는 해당 플랫폼에 따라 지원되는 [Windows Server 버전](#)이 나열되어 있습니다. 목록에 없는 Windows Server 버전은 지원되지 않으며 오류가 발생합니다. 이러한 오류를 방지하려면 사용 중인 플랫폼에 따라 적절한 버전만 사용하십시오.

플랫폼	지원되는 Windows Server 버전
AWS(Amazon Web Services)	Windows Server LTSC(Long-Term Servicing Channel): Windows Server 2019
Microsoft Azure	Windows Server LTSC(Long-Term Servicing Channel): Windows Server 2019
VMware vSphere	Windows Server LTSC(Long-Term Servicing Channel): Windows Server 2022(OS 빌드 <a href="#">20348.681</a> 이상)

플랫폼	지원되는 Windows Server 버전
베어 메탈	Windows Server LTSC(Long-Term Servicing Channel): Windows Server 2019

### 2.6.4. 지원되는 네트워킹

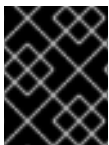
OVN-Kubernetes가 있는 하이브리드 네트워킹은 지원되는 유일한 네트워킹 구성입니다. 이 기능에 대한 자세한 내용은 아래 추가 리소스를 참조하십시오. 다음 표에서는 플랫폼에 따라 사용할 네트워킹 구성 유형과 Windows Server 버전을 간략하게 설명합니다. 클러스터를 설치할 때 네트워크 구성을 지정해야 합니다. OpenShift SDN 네트워킹은 OpenShift Container Platform 클러스터의 기본 네트워킹입니다. 하지만 OpenShift SDN은 WMCO에서 지원되지 않습니다.

표 2.1. 플랫폼 네트워킹 지원

플랫폼	지원되는 네트워킹
AWS(Amazon Web Services)	OVN-Kubernetes로 하이브리드 네트워킹
Microsoft Azure	OVN-Kubernetes로 하이브리드 네트워킹
VMware vSphere	사용자 지정 VXLAN 포트가 있는 OVN-Kubernetes를 사용한 하이브리드 네트워킹
베어 메탈	OVN-Kubernetes로 하이브리드 네트워킹

표 2.2. 하이브리드 OVN-Kubernetes Windows Server 지원

OVN-Kubernetes로 하이브리드 네트워킹	지원되는 Windows Server 버전
기본 VXLAN 포트	Windows Server LTSC(Long-Term Servicing Channel): Windows Server 2019
사용자 지정 VXLAN 포트	Windows Server LTSC(Long-Term Servicing Channel): Windows Server 2022(OS 빌드 20348.681 이상)



**중요**

제한된 네트워크 또는 연결이 끊긴 환경의 클러스터에서는 Windows 컨테이너 워크로드 실행이 지원되지 않습니다.

WMCO 버전 4.x는 OpenShift Container Platform 4.9와만 호환됩니다.

### 2.7. 알려진 제한 사항

WMCO(Windows 노드)에서 관리하는 Windows 노드로 작업할 때는 다음과 같은 제한 사항이 있습니다.

- 다음 OpenShift Container Platform 기능은 Windows 노드에서 지원되지 않습니다.
  - Red Hat OpenShift Developer CLI (odo)
  - 이미지 빌드
  - OpenShift Pipelines
  - OpenShift Service Mesh
  - 사용자 정의 프로젝트의 OpenShift 모니터링
  - OpenShift Serverless
  - 수평 Pod 자동 스케일링
  - 수직 Pod 자동 확장
- 다음 Red Hat 기능은 Windows 노드에서 지원되지 않습니다.
  - [Red Hat 비용 관리](#)
  - [Red Hat OpenShift Local](#)
- Windows 노드는 개인 레지스트리에서 컨테이너 이미지 가져오기를 지원하지 않습니다. 공용 레지스트리에서 이미지를 사용하거나 이미지를 사전 가져올 수 있습니다.
- Windows 노드는 배포 구성을 사용하여 생성된 워크로드를 지원하지 않습니다. 배포 또는 기타 방법을 사용하여 워크로드를 배포할 수 있습니다.
- 클러스터 전체 프록시를 사용하는 클러스터에서는 Windows 노드가 지원되지 않습니다. 이는 WMCO가 워크로드에 대한 프록시 연결을 통해 트래픽을 라우팅할 수 없기 때문입니다.
- 연결이 끊긴 환경에 있는 클러스터에서는 Windows 노드가 지원되지 않습니다.
- Windows Containers 용 Red Hat OpenShift 지원은 트렁크 포트를 통해 클러스터에 Windows 노드를 추가하는 것을 지원하지 않습니다. Windows 노드 추가에 지원되는 유일한 네트워킹 구성은 VLAN에 대한 트래픽을 전송하는 액세스 포트를 사용하는 것입니다.
- Windows Containers 용 Red Hat OpenShift 지원은 모든 클라우드 공급자에 대해 In-tree 스토리지 드라이버만 지원합니다.
- Kubernetes에서 다음 [노드 기능 제한 사항](#)을 확인했습니다.
  - Windows 컨테이너에서는 대규모 페이지가 지원되지 않습니다.
  - 권한이 있는 컨테이너는 Windows 컨테이너에서 지원되지 않습니다.
  - Pod 종료 유예 기간에는 containerd 컨테이너 런타임을 Windows 노드에 설치해야 합니다.
- Kubernetes는 [여러 API 호환성 문제](#)를 확인했습니다.

### 3장. WINDOWS 컨테이너 워크로드 이해

Windows 컨테이너에 대한 Red Hat OpenShift 지원은 OpenShift Container Platform에서 Microsoft Windows Server 컨테이너를 실행하기 위한 기본 지원을 제공합니다. Linux 및 Windows 워크로드가 혼합된 동시 환경을 관리하는 경우 OpenShift Container Platform을 사용하면 RHCOS(Red Hat Enterprise Linux CoreOS) 또는 RHEL(Red Hat Enterprise Linux)에서 호스팅되는 기존 Linux 워크로드를 제공하면서 Windows Server 컨테이너에서 실행 중인 Windows 워크로드를 배포할 수 있습니다.



#### 참고

Windows 노드가 배치된 클러스터에 대한 멀티 테넌트는 지원되지 않습니다. 유해한 멀티 테넌트 사용으로 모든 Kubernetes 환경에서 보안 문제가 발생할 수 있습니다. [Pod 보안 정책](#) 또는 노드에서 보다 세밀하게 조정된 역할 기반 액세스 제어(RBAC)와 같은 추가 보안 기능으로 인해 노드를 악용하기가 더 어렵습니다. 그러나 유해한 멀티 테넌트 워크로드를 실행하도록 선택하는 경우 하이퍼바이저가 사용할 수 있는 유일한 보안 옵션입니다. Kubernetes용 보안 도메인에는 개별 노드가 아닌 전체 클러스터가 포함됩니다. 이러한 유형의 유해한 멀티 테넌트 워크로드의 경우 물리적으로 분리된 클러스터를 사용해야 합니다.

Windows Server 컨테이너는 공유 커널을 사용하여 리소스 격리를 제공하지만 다중 테넌트 시나리오에서는 사용되지 않습니다. 멀티 테넌트와 관련된 시나리오에서는 Hyper-V 격리 컨테이너를 사용하여 테넌트를 강력하게 격리해야 합니다.

### 3.1. WINDOWS MACHINE CONFIG OPERATOR 전제 조건

다음 정보는 Windows Machine Config Operator에 지원되는 플랫폼 버전, Windows Server 버전 및 네트워크 구성에 대해 자세히 설명합니다. 해당 플랫폼과 관련된 모든 정보는 vSphere 설명서를 참조하십시오.

#### 3.1.1. OpenShift Container Platform 및 WMCO 버전을 기반으로 지원되는 플랫폼

플랫폼	지원되는 OpenShift Container Platform 버전	지원되는 WMCO 버전	설치 프로그램에서 프로비저닝한 인프라 설치 지원	사용자 프로비저닝 인프라 설치 지원
AWS(Amazon Web Services)	4.6+	WMCO 1.0+	GA	
Microsoft Azure	4.6+	WMCO 1.0+	GA	
VMware vSphere	4.7+	WMCO 2.0+	GA	

#### 3.1.2. OpenShift Container Platform 및 WMCO 버전을 기반으로 하는 BYOH(Bring-Your-Own-Host) 인스턴스에 지원되는 플랫폼

플랫폼	지원되는 OpenShift Container Platform 버전	지원되는 WMCO 버전	설치 프로그램에서 프로비저닝한 인프라 설치 지원 용 BYOH	사용자 프로비저닝 인프라 설치 지원 용 BYOH
AWS(Amazon Web Services)	4.8+	WMCO 3.1+	GA	
Microsoft Azure	4.8+	WMCO 3.1+	GA	
VMware vSphere	4.8+	WMCO 3.1+	GA	
공급자와 무관함 (Platform: none)	4.8+	WMCO 3.1+		GA

### 3.1.3. 지원되는 Windows Server 버전

다음 표에는 해당 플랫폼에 따라 지원되는 **Windows Server 버전**이 나열되어 있습니다. 목록에 없는 Windows Server 버전은 지원되지 않으며 오류가 발생합니다. 이러한 오류를 방지하려면 사용 중인 플랫폼에 따라 적절한 버전만 사용하십시오.

플랫폼	지원되는 Windows Server 버전
AWS(Amazon Web Services)	Windows Server LTSC(Long-Term Servicing Channel): Windows Server 2019
Microsoft Azure	Windows Server LTSC(Long-Term Servicing Channel): Windows Server 2019
VMware vSphere	Windows Server LTSC(Long-Term Servicing Channel): Windows Server 2022(OS 빌드 <a href="#">20348.681</a> 이상)
베어 메탈	Windows Server LTSC(Long-Term Servicing Channel): Windows Server 2019

### 3.1.4. 지원되는 네트워킹

OVN-Kubernetes가 있는 하이브리드 네트워킹은 지원되는 유일한 네트워킹 구성입니다. 이 기능에 대한 자세한 내용은 아래 추가 리소스를 참조하십시오. 다음 표에서는 플랫폼에 따라 사용할 네트워킹 구성 유형과 Windows Server 버전을 간략하게 설명합니다. 클러스터를 설치할 때 네트워크 구성을 지정해야 합니다. OpenShift SDN 네트워킹은 OpenShift Container Platform 클러스터의 기본 네트워킹입니다. 하지만 OpenShift SDN은 WMCO에서 지원되지 않습니다.

#### 표 3.1. 플랫폼 네트워킹 지원

플랫폼	지원되는 네트워킹
AWS(Amazon Web Services)	OVN-Kubernetes로 하이브리드 네트워킹
Microsoft Azure	OVN-Kubernetes로 하이브리드 네트워킹
VMware vSphere	사용자 지정 VXLAN 포트가 있는 OVN-Kubernetes를 사용한 하이브리드 네트워킹
베어 메탈	OVN-Kubernetes로 하이브리드 네트워킹

표 3.2. 하이브리드 OVN-Kubernetes Windows Server 지원

OVN-Kubernetes로 하이브리드 네트워킹	지원되는 Windows Server 버전
기본 VXLAN 포트	Windows Server LTSC(Long-Term Servicing Channel): Windows Server 2019
사용자 지정 VXLAN 포트	Windows Server LTSC(Long-Term Servicing Channel): Windows Server 2022(OS 빌드 <a href="#">20348.681</a> 이상)

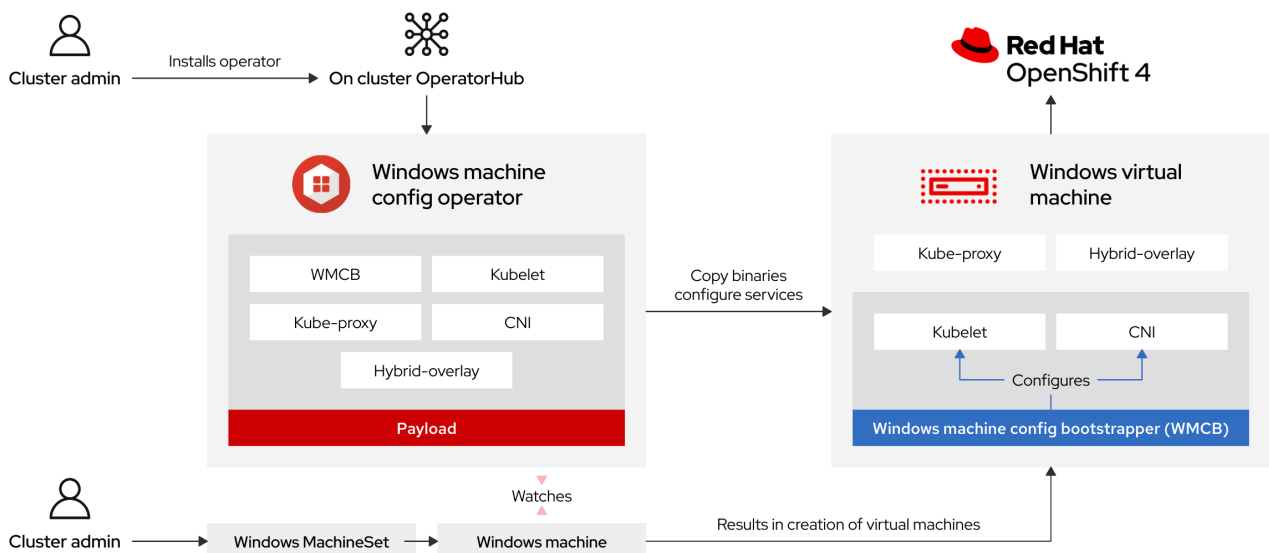
추가 리소스

- [OVN-Kubernetes로 하이브리드 네트워킹 구성 참조](#)

### 3.2. WINDOWS 워크로드 관리

클러스터에서 Windows 워크로드를 실행하려면 먼저 WMCO(Windows Machine Config Operator)를 설치해야 합니다. WMCO는 Linux 기반 Operator로, Linux 기반 컨트롤 플레인 및 컴퓨팅 노드에서 실행됩니다. WMCO는 클러스터에서의 Windows 워크로드 배포 및 관리 프로세스를 오케스트레이션합니다.

그림 3.1. WMCO 설계





Windows 워크로드를 배포하기 전, Windows 컴퓨팅 노드를 생성한 후 클러스터에 참여하도록 해야 합니다. Windows 노드는 클러스터에서 Windows 워크로드를 호스팅하고 다른 Linux 기반 컴퓨팅 노드와 함께 실행할 수 있습니다. 호스트 Windows Server 컴퓨팅 머신으로 설정된 Windows 머신을 생성하여 Windows 컴퓨팅 노드를 생성할 수 있습니다. Docker 형식의 컨테이너 런타임 애드온이 활성화된 Windows OS 이미지를 지정하는 머신 세트에 Windows별 레이블을 적용해야 합니다.

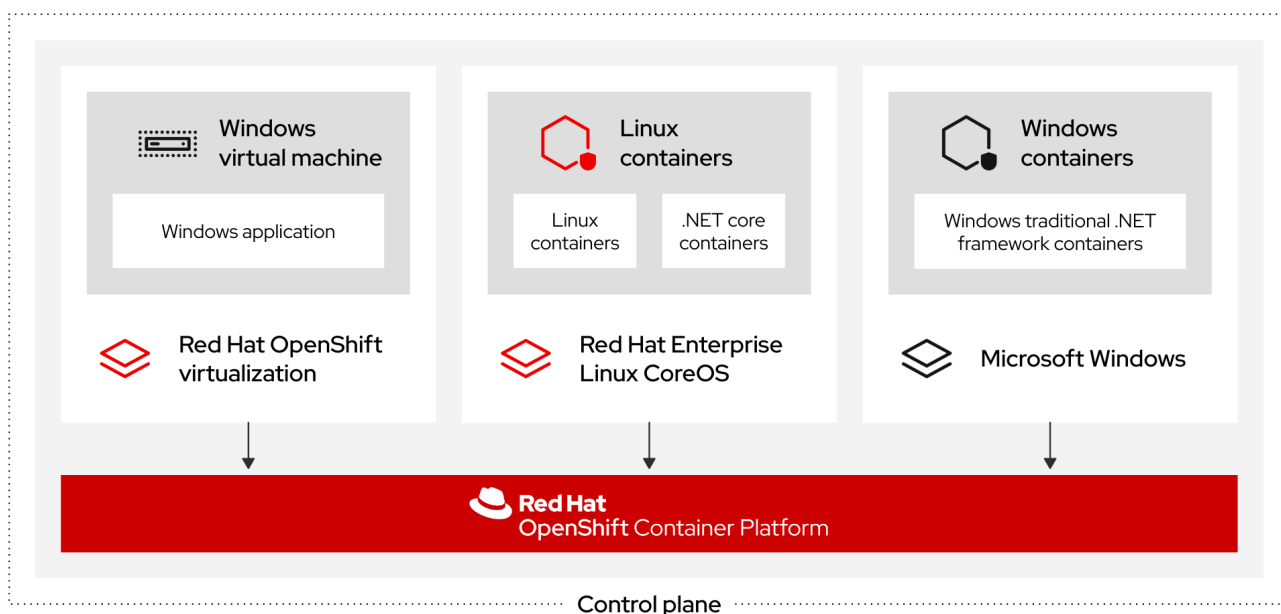


### 중요

현재는 Docker 형식의 컨테이너 런타임이 Windows 노드에서 사용됩니다. Kubernetes는 더 이상 Docker를 컨테이너 런타임으로 사용하지 않습니다. [Docker 사용 중지](#)에 대한 자세한 내용은 Kubernetes 문서를 참조하십시오. 향후 Kubernetes 릴리스에서는 Containerd가 Windows 노드에서 지원되는 새로운 컨테이너 런타임이 됩니다.

WMCO는 Windows 레이블이 있는 머신을 감시합니다. Windows 머신 세트가 감지되고 각 머신이 프로비저닝되면 WMCO는 기본 Windows 가상 머신(VM)을 구성하여 클러스터에 컴퓨팅 노드로 참여할 수 있습니다.

그림 3.2. Windows 및 Linux 워크로드



WMCO는 Windows 인스턴스와 상호 작용하는 데 사용되는 개인 키가 포함된 네임스페이스에 사전 결정된 시크릿이 있을 것으로 예상합니다. WMCO는 부팅 시 이 시크릿을 확인하고 사용자가 생성한 Windows **MachineSet** 오브젝트에서 참조해야 하는 사용자 데이터 시크릿을 생성합니다. 그런 다음 WMCO는 사용자 데이터 시크릿을 개인 키에 해당하는 공개 키로 채웁니다. 이 데이터를 사용하여 SSH 연결을 통해 클러스터가 Windows VM에 연결할 수 있습니다.

클러스터가 Windows VM과 관련된 연결을 설정한 후 Linux 기반 노드와 비슷한 방법을 사용하여 Windows 노드를 관리할 수 있습니다.



### 참고

OpenShift Container Platform 웹 콘솔은 Linux 노드에서 사용 가능할 수 있는 Windows 노드용과 거의 동일한 모니터링 기능을 제공합니다. 그러나 현재 Windows 노드에서 실행 중인 Pod에 대한 워크로드 그래프 모니터링 기능은 사용할 수 없습니다.

Windows 노드에 Windows 워크로드 예약은 테인트, 허용 오차 및 노트 선택기와 같은 일반적인 Pod 예약 방법으로 수행할 수 있습니다. 아니면, **RuntimeClass** 오브젝트를 사용하여 Windows 워크로드를 Linux 워크로드 및 기타 Windows 버전 워크로드와 차별화할 수 있습니다.

### 3.3. WINDOWS 노드 서비스

다음 Windows별 서비스가 각 Windows 노드에 설치됩니다.

Service	설명
kubelet	Windows 노드를 등록하고 상태를 관리합니다.
CNI(컨테이너 네트워크 인터페이스) 플러그인	Windows 노드에 대한 <a href="#">네트워킹</a> 을 노출합니다.
WMCB(Windows Machine Config Bootstrapper)	kubelet 및 CNI 플러그인을 구성합니다.
<a href="#">Windows Exporter</a>	Windows 노드에서 Prometheus 지표 내보내기
hybrid-overlay	OpenShift Container Platform <a href="#">HNS(Host Network Service)</a> 를 생성합니다.
kube-proxy	외부 통신을 허용하는 노드에서 네트워크 규칙을 유지합니다.

### 3.4. 알려진 제한 사항

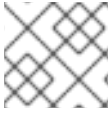
WMCO(Windows 노드)에서 관리하는 Windows 노드로 작업할 때는 다음과 같은 제한 사항이 있습니다.

- 다음 OpenShift Container Platform 기능은 Windows 노드에서 지원되지 않습니다.
  - Red Hat OpenShift Developer CLI (odo)
  - 이미지 빌드
  - OpenShift Pipelines
  - OpenShift Service Mesh
  - 사용자 정의 프로젝트의 OpenShift 모니터링
  - OpenShift Serverless
  - 수평 Pod 자동 스케일링
  - 수직 Pod 자동 확장
- 다음 Red Hat 기능은 Windows 노드에서 지원되지 않습니다.
  - [Red Hat 비용 관리](#)
  - [Red Hat OpenShift Local](#)
- Windows 노드는 개인 레지스트리에서 컨테이너 이미지 가져오기를 지원하지 않습니다. 공용 레지스트리에서 이미지를 사용하거나 이미지를 사전 가져올 수 있습니다.

- Windows 노드는 배포 구성을 사용하여 생성된 워크로드를 지원하지 않습니다. 배포 또는 기타 방법을 사용하여 워크로드를 배포할 수 있습니다.
- 클러스터 전체 프록시를 사용하는 클러스터에서는 Windows 노드가 지원되지 않습니다. 이는 WMCO가 워크로드에 대한 프록시 연결을 통해 트래픽을 라우팅할 수 없기 때문입니다.
- 연결이 끊긴 환경에 있는 클러스터에서는 Windows 노드가 지원되지 않습니다.
- Windows Containers 용 Red Hat OpenShift 지원은 트렁크 포트를 통해 클러스터에 Windows 노드를 추가하는 것을 지원하지 않습니다. Windows 노드 추가에 지원되는 유일한 네트워킹 구성은 VLAN에 대한 트래픽을 전송하는 액세스 포트를 사용하는 것입니다.
- Windows Containers 용 Red Hat OpenShift 지원은 모든 클라우드 공급자에 대해 In-tree 스토리지 드라이버만 지원합니다.
- Kubernetes에서 다음 [노드 기능 제한 사항](#)을 확인했습니다.
  - Windows 컨테이너에서는 대규모 페이지가 지원되지 않습니다.
  - 권한이 있는 컨테이너는 Windows 컨테이너에서 지원되지 않습니다.
  - Pod 종료 유예 기간에는 containerd 컨테이너 런타임을 Windows 노드에 설치해야 합니다.
- Kubernetes는 [여러 API 호환성 문제](#)를 확인했습니다.

## 4장. WINDOWS 컨테이너 워크로드 활성화

Windows 워크로드를 클러스터에 추가하기 전에 OpenShift Container Platform OperatorHub에서 사용할 수 있는 WMCO(Windows Machine Config Operator)를 설치해야 합니다. WMCO는 클러스터에서의 Windows 워크로드 배포 및 관리 프로세스를 오케스트레이션합니다.



### 참고

WMCO에서 관리하는 Windows 인스턴스에서 듀얼 NIC가 지원되지 않습니다.

### 사전 요구 사항

- **cluster-admin** 권한이 있는 계정을 사용하여 OpenShift Container Platform 클러스터에 액세스할 수 있습니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.
- 설치 관리자 프로비저닝 인프라를 사용하거나 **install-config.yaml** 파일에 설정된 **platform: none** 필드가 있는 사용자 프로비저닝 인프라를 사용하여 클러스터를 설치했습니다.
- 클러스터를 위한 OVN-Kubernetes를 사용하여 하이브리드 네트워킹이 구성되었습니다. 클러스터를 설치하는 동안 완료해야 합니다. 자세한 내용은 [하이브리드 네트워킹 구성](#)을 참조하십시오.
- OpenShift Container Platform 클러스터 버전 4.6.8 이상이 실행 중입니다.



### 참고

WMCO는 워크로드에 대한 프록시 연결을 통해 트래픽을 라우팅할 수 없기 때문에 [클러스터 전체 프록시](#)를 사용하는 클러스터에서는 WMCO가 지원되지 않습니다.

### 추가 리소스

- Windows Machine Config Operator에 대한 포괄적인 사전 요구 사항은 [Windows 컨테이너 워크로드 이해](#)를 참조하십시오.

## 4.1. WINDOWS MACHINE CONFIG OPERATOR 설치

웹 콘솔 또는 OpenShift CLI(**oc**)를 사용하여 Windows Machine Config Operator를 설치할 수 있습니다.

### 4.1.1. 웹 콘솔을 사용하여 Windows Machine Config Operator 설치

OpenShift Container Platform 웹 콘솔을 사용하여 WMCO(Windows Machine Config Operator)를 설치할 수 있습니다.



### 참고

WMCO에서 관리하는 Windows 인스턴스에서 듀얼 NIC가 지원되지 않습니다.

### 절차

1. OpenShift Container Platform 웹 콘솔에서 관리자로 **Operator → OperatorHub** 페이지로 이동합니다.

2. 키워드로 필터링 상자를 사용하여 카탈로그에서 **Windows Machine Config Operator**를 검색합니다. **Windows Machine Config Operator**타일을 클릭합니다.
3. Operator에 대한 정보를 확인하고 **설치**를 클릭합니다.
4. **Operator 설치** 페이지에서 다음을 수행합니다.
  - a. **stable** 채널을 **업데이트 채널**로 선택합니다. **stable** 채널을 사용하면 WMCO의 안정적인 최신 릴리스를 설치할 수 있습니다.
  - b. WMCO는 단일 네임스페이스에서만 사용 가능해야 하므로 **설치 모드**가 사전 구성됩니다.
  - c. WMCO용으로 설치된 네임스페이스를 선택합니다. 기본 Operator 권장 네임스페이스는 **openshift-windows-machine-config-operator**입니다.
  - d. 네임스페이스에서 **Operator 권장 클러스터 모니터링 활성화** 확인란을 클릭하여 WMCO에 대한 클러스터 모니터링을 활성화합니다.
  - e. **승인 전략**을 선택합니다.
    - 자동 전략을 사용하면 Operator 새 버전이 준비될 때 OLM(Operator Lifecycle Manager)이 자동으로 Operator를 업데이트할 수 있습니다.
    - 수동 전략을 사용하려면 적절한 자격 증명을 가진 사용자가 Operator 업데이트를 승인해야 합니다.
1. **설치**를 클릭합니다. 설치된 **Operator** 페이지에 이제 WMCO가 나열됩니다.



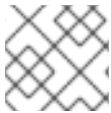
#### 참고

WMCO는 사용자가 정의한 네임스페이스에 **openshift-windows-machine-config-operator**와 같이 자동으로 설치됩니다.

2. WMCO가 성공적으로 설치되었는지 확인하려면 **상태**가 **성공**으로 표시되는지 확인합니다.

### 4.1.2. CLI를 사용하여 Windows Machine Config Operator 설치

OpenShift CLI(**oc**)를 사용하여 WMCO(Windows Machine Config Operator)를 설치할 수 있습니다.



#### 참고

WMCO에서 관리하는 Windows 인스턴스에서 듀얼 NIC가 지원되지 않습니다.

#### 절차

1. WMCO를 위한 네임스페이스를 생성합니다.
  - a. WMCO를 위한 **네임스페이스 오브젝트** YAML 파일을 생성합니다. 예를 들어, **wmco-namespace.yaml**은 다음과 같습니다.

```
apiVersion: v1
kind: Namespace
metadata:
```

```
name: openshift-windows-machine-config-operator 1
labels:
  openshift.io/cluster-monitoring: "true" 2
```

- 1** **openshift-windows-machine-config-operator** 네임스페이스에 WMCO를 배포하는 것이 좋습니다.
- 2** WMCO에 대한 클러스터 모니터링을 활성화하려면 이 레이블이 필요합니다.

b. 네임스페이스를 생성합니다.

```
$ oc create -f <file-name>.yaml
```

예를 들면 다음과 같습니다.

```
$ oc create -f wmco-namespace.yaml
```

2. WMCO를 위한 Operator 그룹을 생성합니다.

- a. **OperatorGroup** 오브젝트 YAML 파일을 생성합니다. 예를 들어, **wmco-og.yaml**은 다음과 같습니다.

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: windows-machine-config-operator
  namespace: openshift-windows-machine-config-operator
spec:
  targetNamespaces:
    - openshift-windows-machine-config-operator
```

b. Operator 그룹을 생성합니다.

```
$ oc create -f <file-name>.yaml
```

예를 들면 다음과 같습니다.

```
$ oc create -f wmco-og.yaml
```

3. 네임스페이스가 WMCO를 구독하도록 합니다.

- a. **서브스크립션** 오브젝트 YAML 파일을 생성합니다. 예를 들어, **wmco-sub.yaml**은 다음과 같습니다.

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: windows-machine-config-operator
  namespace: openshift-windows-machine-config-operator
spec:
  channel: "stable" 1
  installPlanApproval: "Automatic" 2
```

```
name: "windows-machine-config-operator"
source: "redhat-operators" 3
sourceNamespace: "openshift-marketplace" 4
```

- 1 **stable** 을 채널로 지정합니다.
- 2 승인 전략을 설정합니다. 자동 또는 수동을 설정할 수 있습니다.
- 3 **windows-machine-config-operator** 패키지 매니페스트가 포함된 **redhat-operators** 카탈로그 소스를 지정합니다. OpenShift Container Platform이 제한된 네트워크(연결이 끊긴 클러스터)에 설치된 경우 OLM(Operator Lifecycle Manager)을 구성할 때 생성된 **CatalogSource** 오브젝트의 이름을 지정합니다.
- 4 카탈로그 소스의 네임스페이스입니다. 기본 OperatorHub 카탈로그 소스에는 **openshift-marketplace**를 사용합니다.

b. 서브스크립션을 생성합니다.

```
$ oc create -f <file-name>.yaml
```

예를 들면 다음과 같습니다.

```
$ oc create -f wmco-sub.yaml
```

이제 WMCO가 **openshift-windows-machine-config-operator**에 설치됩니다.

4. WMCO 설치를 확인합니다.

```
$ oc get csv -n openshift-windows-machine-config-operator
```

출력 예

```
NAME                                DISPLAY                                VERSION REPLACES PHASE
windows-machine-config-operator.2.0.0  Windows Machine Config Operator  2.0.0
Succeeded
```

## 4.2. WINDOWS MACHINE CONFIG OPERATOR에 대한 시크릿 구성

WMCO(Windows Machine Config Operator)를 실행하려면 개인 키가 포함된 WMCO 네임스페이스에 시크릿을 생성해야 합니다. 이를 위해서는 WMCO가 Windows VM(가상 머신)과 통신할 수 있도록 허용되어야 합니다.

사전 요구 사항

- OLM(Operator Lifecycle Manager)을 사용하여 WMCO(Windows Machine Config Operator)를 설치했습니다.
- RSA 키가 포함된 PEM 인코딩 파일이 생성되었습니다.

절차

- Windows VM에 액세스하기 위해 필요한 시크릿을 정의합니다.

-

```
$ oc create secret generic cloud-private-key --from-file=private-  
key.pem=${HOME}/.ssh/<key> \  
-n openshift-windows-machine-config-operator 1
```

- 1 **openshift-windows-machine-config-operator**와 같이 WMCO 네임스페이스에 개인 키를 생성해야 합니다.

클러스터를 설치할 때 사용한 것과 다른 개인 키를 사용하는 것이 좋습니다.

### 4.3. 추가 리소스

- [클러스터 노드 SSH 액세스를 위한 키 쌍 생성](#)
- [클러스터에 Operator 추가](#).



## 5장. WINDOWS MACHINESET 오브젝트 생성

### 5.1. AWS에서 WINDOWS MACHINESET 오브젝트 생성

AWS(Amazon Web Services)의 OpenShift Container Platform 클러스터에서 특정 목적을 수행하기 위해 Windows **MachineSet** 오브젝트를 생성할 수 있습니다. 예를 들어, 지원되는 워크로드를 새 Windows 머신으로 이동할 수 있도록 인프라 Windows MachineSet 및 관련 머신을 생성할 수 있습니다.

#### 사전 요구 사항

- OLM(Operator Lifecycle Manager)을 사용하여 WMCO(Windows Machine Config Operator)를 설치했습니다.
- 지원되는 Windows Server를 Docker 형식 컨테이너 런타임 애드온이 활성화된 운영 체제 이미지로 사용하고 있습니다.  
다음 **aws** 명령을 사용하여 유효한 AMI 이미지를 쿼리합니다.

```
$ aws ec2 describe-images --region <aws region name> --filters
"Name=name,Values=Windows_Server-2019*English*Full*Containers*" "Name=is-
public,Values=true" --query "reverse(sort_by(Images, &CreationDate))[*].{name: Name, id:
ImageId}" --output table
```

#### 중요

현재는 Docker 형식의 컨테이너 런타임이 Windows 노드에서 사용됩니다. Kubernetes는 더 이상 Docker를 컨테이너 런타임으로 사용하지 않습니다. [Docker 사용 중지](#)에 대한 자세한 내용은 Kubernetes 문서를 참조하십시오. 향후 Kubernetes 릴리스에서는 Containerd가 Windows 노드에서 지원되는 새로운 컨테이너 런타임이 됩니다.

#### 5.1.1. Machine API 개요

Machine API는 업스트림 Cluster API 프로젝트 및 사용자 정의 OpenShift Container Platform 리소스를 기반으로 하는 주요 리소스의 조합입니다.

OpenShift Container Platform 4.9 클러스터의 경우 Machine API는 클러스터 설치가 완료된 후 모든 노드 호스트 프로비저닝 관리 작업을 수행합니다. 이 시스템으로 인해 OpenShift Container Platform 4.9는 퍼블릭 또는 프라이빗 클라우드 인프라에 더하여 탄력적이고 동적인 프로비저닝 방법을 제공합니다.

두 가지 주요 리소스는 다음과 같습니다.

#### Machine

노드의 호스트를 설명하는 기본 단위입니다. 머신에는 **providerSpec** 사양이 있으며 이는 다른 클라우드 플랫폼에 제공되는 컴퓨팅 노드 유형을 설명합니다. 예를 들어 AWS(Amazon Web Services)의 작업자 노드에 대한 머신 유형은 특정 머신 유형과 필요한 메타 데이터를 정의할 수 있습니다.

#### 머신 세트

**MachineSet** 리소스는 머신 그룹입니다. 머신 세트는 머신에 연관되어 있고 복제본 세트는 pod에 연관되어 있습니다. 더 많은 머신이 필요하거나 규모를 줄여야 하는 경우 컴퓨터 요구 사항에 맞게 머신 세트의 **replicas** 필드를 변경합니다.



### 주의

컨트롤 플레인 시스템은 머신 세트에서 관리할 수 없습니다.

다음 사용자 지정 리소스는 클러스터에 더 많은 기능을 추가할 수 있습니다.

#### 머신 자동 스케일러

**MachineAutoscaler** 리소스는 클라우드에서 컴퓨팅 머신을 자동으로 확장합니다. 지정된 컴퓨팅 머신 세트에서 노드의 최소 및 최대 스케일링 경계를 설정할 수 있으며 머신 자동 스케일러는 해당 노드 범위를 유지합니다.

**MachineAutoscaler** 객체는 **ClusterAutoscaler** 객체를 설정한 후에 사용할 수 있습니다.

**ClusterAutoscaler** 및 **MachineAutoscaler** 리소스는 모두 **ClusterAutoscalerOperator** 오브젝트에서 사용 가능합니다.

#### Cluster autoscaler

이 리소스는 업스트림 클러스터 자동 스케일러 프로젝트를 기반으로 합니다. OpenShift Container Platform 구현에서는 머신 세트 API를 확장하여 Machine API와 통합됩니다. 코어, 노드, 메모리, GPU 등과 같은 리소스의 클러스터 전체에서 확장 제한을 설정할 수 있습니다. 중요도가 낮은 Pod에 대해 새 노드가 온라인 상태가 되지 않도록 클러스터가 Pod에 우선 순위를 설정할 수 있습니다. 노드를 확장할 수는 있지만 축소할 수 없도록 확장 정책을 설정할 수도 있습니다.

#### 머신 상태 점검

**MachineHealthCheck** 리소스는 머신의 비정상적인 상태를 감지하여 삭제한 후 지원되는 플랫폼에서 새 머신을 생성합니다.

OpenShift Container Platform 버전 3.11에서는 클러스터가 머신 프로비저닝을 관리하지 않았기 때문에 다중 영역 아키텍처를 쉽게 롤아웃할 수 없었습니다. OpenShift Container Platform 버전 4.1부터 이러한 프로세스가 더 쉬워졌습니다. 각 머신 세트의 범위는 단일 영역에서 지정되므로 설치 프로그램은 사용자를 대신하여 가용성 영역 전체에 머신 세트를 보냅니다. 또한 계산이 동적이고 영역 장애가 발생하여 머신을 재조정해야 하는 경우 처리할 수 있는 영역을 확보할 수 있습니다. Autoscaler는 클러스터의 수명 기간 동안 최적의 균형을 유지합니다.

### 5.1.2. AWS에서 Windows MachineSet 오브젝트를 위한 샘플 YAML

이 샘플 YAML은 WMCO(Windows Machine Config Operator)에서 응답할 수 있는 AWS(Amazon Web Services)에서 실행 중인 Windows **MachineSet** 오브젝트를 정의합니다.

```

apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
  name: <infrastructure_id>-windows-worker-<zone> 2
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 3

```

```

machine.openshift.io/cluster-api-machineset: <infrastructure_id>-windows-worker-<zone> 4
template:
  metadata:
    labels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
      machine.openshift.io/cluster-api-machine-role: worker
      machine.openshift.io/cluster-api-machine-type: worker
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-windows-worker-<zone> 6
      machine.openshift.io/os-id: Windows 7
  spec:
    metadata:
      labels:
        node-role.kubernetes.io/worker: "" 8
    providerSpec:
      value:
        ami:
          id: <windows_container_ami> 9
        apiVersion: awsproviderconfig.openshift.io/v1beta1
        blockDevices:
          - ebs:
              iops: 0
              volumeSize: 120
              volumeType: gp2
        credentialsSecret:
          name: aws-cloud-credentials
        deviceIndex: 0
        iamInstanceProfile:
          id: <infrastructure_id>-worker-profile 10
        instanceType: m5a.large
        kind: AWSMachineProviderConfig
        placement:
          availabilityZone: <zone> 11
          region: <region> 12
        securityGroups:
          - filters:
              - name: tag:Name
                values:
                  - <infrastructure_id>-worker-sg 13
        subnet:
          filters:
            - name: tag:Name
              values:
                - <infrastructure_id>-private-<zone> 14
        tags:
          - name: kubernetes.io/cluster/<infrastructure_id> 15
            value: owned
        userDataSecret:
          name: windows-user-data 16
          namespace: openshift-machine-api

```

1 3 5 10 13 14 15 클러스터를 프로비저닝할 때 설정한 클러스터 ID를 기반으로 하는 인프라 ID를 지정합니다. 다음 명령을 실행하여 인프라 ID를 가져올 수 있습니다.

```
$ oc get -o jsonpath='{.status.infrastructureName}' infrastructure cluster
```

- 2 4 6 인프라 ID, 작업자 레이블 및 영역을 지정합니다.
- 7 머신 세트를 Windows 머신으로 구성합니다.
- 8 Windows 노드를 컴퓨팅 머신으로 구성합니다.
- 9 컨테이너 런타임이 설치된 Windows 이미지의 AMI ID를 지정합니다. Windows Server 2019를 사용해야 합니다.
- 11 **us-east-1a**와 같이 AWS 영역을 지정합니다.
- 12 **us-gov-east-1**과 같이 AWS 리전을 지정합니다.
- 16 첫 번째 Windows 머신을 구성할 때 WMCO에 의해 생성되었습니다. 이후에는, 모든 후속 머신 세트에서 **Windows-user-data**를 사용할 수 있습니다.

### 5.1.3. 머신 세트 만들기

설치 프로그램에서 생성한 컴퓨팅 머신 세트 외에도 고유한 머신 세트를 생성하여 선택한 특정 워크로드의 머신 컴퓨팅 리소스를 동적으로 관리할 수 있습니다.

#### 사전 요구 사항

- OpenShift Container Platform 클러스터를 배포합니다.
- OpenShift CLI(**oc**)를 설치합니다.
- **cluster-admin** 권한이 있는 사용자로 **oc**에 로그인합니다.

#### 절차

1. 머신 세트 CR(사용자 지정 리소스) 샘플이 포함된 이름이 **<file\_name>.yaml**인 새 YAML 파일을 만듭니다.  
**<clusterID>** 및 **<role>** 매개 변수 값을 설정해야 합니다.
2. 선택 사항: 특정 필드에 설정할 값이 확실하지 않은 경우 클러스터에서 기존 컴퓨팅 머신 세트를 확인할 수 있습니다.
  - a. 클러스터의 컴퓨팅 머신 세트를 나열하려면 다음 명령을 실행합니다.

```
$ oc get machinesets -n openshift-machine-api
```

#### 출력 예

NAME	DESIRED	CURRENT	READY	AVAILABLE	AGE
agl030519-vplxk-worker-us-east-1a	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1b	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1c	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1d	0	0			55m
agl030519-vplxk-worker-us-east-1e	0	0			55m
agl030519-vplxk-worker-us-east-1f	0	0			55m

- b. 특정 컴퓨팅 머신 세트 CR(사용자 정의 리소스)의 값을 보려면 다음 명령을 실행합니다.

```
$ oc get machineset <machineset_name> \
-n openshift-machine-api -o yaml
```

### 출력 예

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> ❶
  name: <infrastructure_id>-<role> ❷
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id>
        machine.openshift.io/cluster-api-machine-role: <role>
        machine.openshift.io/cluster-api-machine-type: <role>
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>
    spec:
      providerSpec: ❸
      ...
```

❶ 클러스터 인프라 ID입니다.

❷ 기본 노드 레이블입니다.



### 참고

사용자 프로비저닝 인프라가 있는 클러스터의 경우 컴퓨팅 머신 세트는 작업자 및 인프라 유형 머신만 생성할 수 있습니다.

❸

컴퓨팅 머신 세트 CR의 `<providerSpec>` 섹션에 있는 값은 플랫폼에 따라 다릅니다. CR의 `<providerSpec>` 매개변수에 대한 자세한 내용은 공급자의 샘플 컴퓨팅 머신 세트 CR 구성을 참조하십시오.

3.

다음 명령을 실행하여 **MachineSet CR**을 생성합니다.

```
$ oc create -f <file_name>.yaml
```

검증

- 다음 명령을 실행하여 컴퓨팅 머신 세트 목록을 확인합니다.

```
$ oc get machineset -n openshift-machine-api
```

출력 예

NAME	DESIRED	CURRENT	READY	AVAILABLE	AGE
agl030519-vplxk-windows-worker-us-east-1a	1	1	1	1	11m
agl030519-vplxk-worker-us-east-1a	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1b	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1c	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1d	0	0			55m
agl030519-vplxk-worker-us-east-1e	0	0			55m
agl030519-vplxk-worker-us-east-1f	0	0			55m

새 머신 세트가 사용 가능한 경우 **DESIRED** 및 **CURRENT** 값이 일치합니다. 머신 세트를 사용할 수 없는 경우 몇 분 후에 명령을 다시 실행합니다.

5.1.4. 추가 리소스

- 머신 세트 관리에 대한 자세한 내용은 *머신 관리* 섹션을 참조하십시오.

5.2. AZURE에서 WINDOWS MACHINESET 오브젝트 만들기

Microsoft Azure의 OpenShift Container Platform 클러스터에서 특정 목적을 수행하기 위해 **Windows MachineSet** 오브젝트를 생성할 수 있습니다. 예를 들어, 지원되는 워크로드를 새 **Windows** 머신으로 이동할 수 있도록 인프라 **Windows MachineSet** 및 관련 머신을 생성할 수 있습니다.

사전 요구 사항

- **OLM(Operator Lifecycle Manager)**을 사용하여 **WMCO(Windows Machine Config Operator)**를 설치했습니다.
- 지원되는 **Windows Server**를 **Docker** 형식 컨테이너 런타임 애드온이 활성화된 운영 체제 이미지로 사용하고 있습니다.



### 중요

현재는 **Docker** 형식의 컨테이너 런타임이 **Windows** 노드에서 사용됩니다. **Kubernetes**는 더 이상 **Docker**를 컨테이너 런타임으로 사용하지 않습니다. **Docker 사용 중지**에 대한 자세한 내용은 **Kubernetes** 문서를 참조하십시오. 향후 **Kubernetes** 릴리스에서는 **Containerd**가 **Windows** 노드에서 지원되는 새로운 컨테이너 런타임이 됩니다.

#### 5.2.1. Machine API 개요

**Machine API**는 업스트림 **Cluster API** 프로젝트 및 사용자 정의 **OpenShift Container Platform** 리소스를 기반으로 하는 주요 리소스의 조합입니다.

**OpenShift Container Platform 4.9** 클러스터의 경우 **Machine API**는 클러스터 설치가 완료된 후 모든 노드 호스트 프로비저닝 관리 작업을 수행합니다. 이 시스템으로 인해 **OpenShift Container Platform 4.9**는 퍼블릭 또는 프라이빗 클라우드 인프라에 더하여 탄력적이고 동적인 프로비저닝 방법을 제공합니다.

두 가지 주요 리소스는 다음과 같습니다.

#### Machine

노드의 호스트를 설명하는 기본 단위입니다. 머신에는 **providerSpec** 사양이 있으며 이는 다른 클라우드 플랫폼에 제공되는 컴퓨팅 노드 유형을 설명합니다. 예를 들어 **AWS(Amazon Web Services)**의 작업자 노드에 대한 머신 유형은 특정 머신 유형과 필요한 메타 데이터를 정의할 수 있습니다.

#### 머신 세트

**MachineSet** 리소스는 머신 그룹입니다. 머신 세트는 머신에 연관되어 있고 복제본 세트는 **pod**에 연관되어 있습니다. 더 많은 머신이 필요하거나 규모를 줄여야 하는 경우 컴퓨터 요구 사항에 맞게 머신 세트의 **replicas** 필드를 변경합니다.



### 주의

컨트롤 플레인 시스템은 머신 세트에서 관리할 수 없습니다.

다음 사용자 지정 리소스는 클러스터에 더 많은 기능을 추가할 수 있습니다.

## 머신 자동 스케일러

**MachineAutoscaler** 리소스는 클라우드에서 컴퓨팅 머신을 자동으로 확장합니다. 지정된 컴퓨팅 머신 세트에서 노드의 최소 및 최대 스케일링 경계를 설정할 수 있으며 머신 자동 스케일러는 해당 노드 범위를 유지합니다.

**MachineAutoscaler** 객체는 **ClusterAutoscaler** 객체를 설정한 후에 사용할 수 있습니다. **ClusterAutoscaler** 및 **MachineAutoscaler** 리소스는 모두 **ClusterAutoscalerOperator** 오브젝트에서 사용 가능합니다.

## Cluster autoscaler

이 리소스는 업스트림 클러스터 자동 스케일러 프로젝트를 기반으로 합니다. **OpenShift Container Platform** 구현에서는 머신 세트 API를 확장하여 **Machine API**와 통합됩니다. 코어, 노드, 메모리, **GPU** 등과 같은 리소스의 클러스터 전체에서 확장 제한을 설정할 수 있습니다. 중요도가 낮은 **Pod**에 대해 새 노드가 온라인 상태가 되지 않도록 클러스터가 **Pod**에 우선 순위를 설정할 수 있습니다. 노드를 확장할 수는 있지만 축소할 수 없도록 확장 정책을 설정할 수도 있습니다.

## 머신 상태 점검

**MachineHealthCheck** 리소스는 머신의 비정상적인 상태를 감지하여 삭제한 후 지원되는 플랫폼에서 새 머신을 생성합니다.

**OpenShift Container Platform** 버전 3.11에서는 클러스터가 머신 프로비저닝을 관리하지 않았기 때문에 다중 영역 아키텍처를 쉽게 돌아올 수 없었습니다. **OpenShift Container Platform** 버전 4.1부터 이러한 프로세스가 더 쉬워졌습니다. 각 머신 세트의 범위는 단일 영역에서 지정되므로 설치 프로그램은 사용자를 대신하여 가용성 영역 전체에 머신 세트를 보냅니다. 또한 계산이 동적이고 영역 장애가 발생하여 머신을 재조정해야 하는 경우 처리할 수 있는 영역을 확보할 수 있습니다. **Autoscaler**는 클러스터의 수명 기간 동안 최적의 균형을 유지합니다.

### 5.2.2. Azure에서 Windows MachineSet 오브젝트를 위한 샘플 YAML

이 샘플 YAML은 Microsoft Azure에서 WMCO(Windows Machine Config Operator)가 응답할 수 있는 **Windows MachineSet** 오브젝트를 정의합니다.

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
  name: <windows_machine_set_name> 2
  namespace: openshift-machine-api
```



```

spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 3
      machine.openshift.io/cluster-api-machineset: <windows_machine_set_name> 4
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
        machine.openshift.io/cluster-api-machine-role: worker
        machine.openshift.io/cluster-api-machine-type: worker
        machine.openshift.io/cluster-api-machineset: <windows_machine_set_name> 6
        machine.openshift.io/os-id: Windows 7
spec:
  metadata:
    labels:
      node-role.kubernetes.io/worker: "" 8
  providerSpec:
    value:
      apiVersion: azureproviderconfig.openshift.io/v1beta1
      credentialsSecret:
        name: azure-cloud-credentials
        namespace: openshift-machine-api
      image: 9
        offer: WindowsServer
        publisher: MicrosoftWindowsServer
        resourceID: ""
        sku: 2019-Datacenter-with-Containers
        version: latest
      kind: AzureMachineProviderSpec
      location: <location> 10
      managedIdentity: <infrastructure_id>-identity 11
      networkResourceGroup: <infrastructure_id>-rg 12
      osDisk:
        diskSizeGB: 128
        managedDisk:
          storageAccountType: Premium_LRS
          osType: Windows
      publicIP: false
      resourceGroup: <infrastructure_id>-rg 13
      subnet: <infrastructure_id>-worker-subnet
      userDataSecret:
        name: windows-user-data 14
        namespace: openshift-machine-api
      vmSize: Standard_D2s_v3
      vnet: <infrastructure_id>-vnet 15
      zone: "<zone>" 16

```

1 3 5 11 12 13 15

클러스터를 프로비저닝할 때 설정한 클러스터 ID를 기반으로 하는 인프라 ID를 지정합니다. 다음 명령을 실행하여 인프라 ID를 가져올 수 있습니다.

```
$ oc get -o jsonpath='{.status.infrastructureName}' infrastructure cluster
```

2 4 6

**Windows** 머신 세트의 이름을 지정합니다. **Azure**의 **Windows** 머신 이름은 15자 이하여야 합니다. 따라서 머신 세트 이름은 머신 이름이 생성되는 방식으로 인해 9자를 초과할 수 없습니다.

7

머신 세트를 **Windows** 머신으로 구성합니다.

8

**Windows** 노드를 컴퓨팅 머신으로 구성합니다.

9

**2019-Datacenter-with-Containers SKU**를 정의하는 **WindowsServer** 이미지 제공을 지정합니다.

10

**centralus**와 같이 **Azure** 리전을 지정합니다.

14

첫 번째 **Windows** 머신을 구성할 때 **WMCO**에 의해 생성되었습니다. 이후에는, 모든 후속 머신 세트에서 **Windows-user-data**를 사용할 수 있습니다.

16

머신을 배치할 리전 내 영역을 지정합니다. 해당 리전이 지정한 영역을 지원하는지 확인합니다.

### 5.2.3. 머신 세트 만들기

설치 프로그램에서 생성한 컴퓨팅 머신 세트 외에도 고유한 머신 세트를 생성하여 선택한 특정 워크로드의 머신 컴퓨팅 리소스를 동적으로 관리할 수 있습니다.

#### 사전 요구 사항

- **OpenShift Container Platform** 클러스터를 배포합니다.

- OpenShift CLI(oc)를 설치합니다.
- cluster-admin 권한이 있는 사용자로 oc에 로그인합니다.

## 절차

1. 머신 세트 CR(사용자 지정 리소스) 샘플이 포함된 이름이 <file\_name>.yaml인 새 YAML 파일을 만듭니다.

<clusterID> 및 <role> 매개 변수 값을 설정해야 합니다.

2. 선택 사항: 특정 필드에 설정할 값이 확실하지 않은 경우 클러스터에서 기존 컴퓨팅 머신 세트를 확인할 수 있습니다.

- a. 클러스터의 컴퓨팅 머신 세트를 나열하려면 다음 명령을 실행합니다.

```
$ oc get machinesets -n openshift-machine-api
```

출력 예

NAME	DESIRED	CURRENT	READY	AVAILABLE	AGE
agl030519-vplxk-worker-us-east-1a	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1b	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1c	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1d	0	0			55m
agl030519-vplxk-worker-us-east-1e	0	0			55m
agl030519-vplxk-worker-us-east-1f	0	0			55m

- b. 특정 컴퓨팅 머신 세트 CR(사용자 정의 리소스)의 값을 보려면 다음 명령을 실행합니다.

```
$ oc get machineset <machineset_name> \
-n openshift-machine-api -o yaml
```

출력 예

```

apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
    name: <infrastructure_id>-<role> 2
    namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id>
        machine.openshift.io/cluster-api-machine-role: <role>
        machine.openshift.io/cluster-api-machine-type: <role>
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>
    spec:
      providerSpec: 3
      ...

```

1

클러스터 인프라 ID입니다.

2

기본 노드 레이블입니다.



참고

사용자 프로비저닝 인프라가 있는 클러스터의 경우 컴퓨팅 머신 세트는 작업자 및 인프라 유형 머신만 생성할 수 있습니다.

3

컴퓨팅 머신 세트 CR의 `<providerSpec>` 섹션에 있는 값은 플랫폼에 따라 다릅니다. CR의 `<providerSpec>` 매개변수에 대한 자세한 내용은 공급자의 샘플 컴퓨팅 머신 세트 CR 구성을 참조하십시오.

3.

다음 명령을 실행하여 **MachineSet CR**을 생성합니다.

```
$ oc create -f <file_name>.yaml
```

검증

•

다음 명령을 실행하여 컴퓨팅 머신 세트 목록을 확인합니다.

```
$ oc get machineset -n openshift-machine-api
```

출력 예

NAME	DESIRED	CURRENT	READY	AVAILABLE	AGE
agl030519-vplxk-windows-worker-us-east-1a	1	1	1	1	11m
agl030519-vplxk-worker-us-east-1a	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1b	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1c	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1d	0	0			55m
agl030519-vplxk-worker-us-east-1e	0	0			55m
agl030519-vplxk-worker-us-east-1f	0	0			55m

새 머신 세트가 사용 가능한 경우 **DESIRED** 및 **CURRENT** 값이 일치합니다. 머신 세트를 사용할 수 없는 경우 몇 분 후에 명령을 다시 실행합니다.

#### 5.2.4. 추가 리소스

•

머신 세트 관리에 대한 자세한 내용은 *머신 관리* 섹션을 참조하십시오.

### 5.3. VSPHERE에서 WINDOWS MACHINESSET 오브젝트 생성

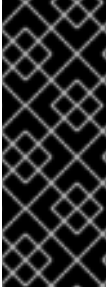
VMware vSphere의 OpenShift Container Platform 클러스터에서 특정 목적을 수행하기 위해 Windows MachineSet 오브젝트를 생성할 수 있습니다. 예를 들어, 지원되는 워크로드를 새 Windows 머신으로 이동할 수 있도록 인프라 Windows MachineSet 및 관련 머신을 생성할 수 있습니다.

사전 요구 사항

•

OLM(Operator Lifecycle Manager)을 사용하여 WMCO(Windows Machine Config Operator)를 설치했습니다.

- 지원되는 Windows Server를 Docker 형식 컨테이너 런타임 애드온이 활성화된 운영 체제 이미지로 사용하고 있습니다.



중요

현재는 Docker 형식의 컨테이너 런타임이 Windows 노드에서 사용됩니다. Kubernetes는 더 이상 Docker를 컨테이너 런타임으로 사용하지 않습니다. [Docker 사용 중지](#)에 대한 자세한 내용은 Kubernetes 문서를 참조하십시오. 향후 Kubernetes 릴리스에서는 Containerd가 Windows 노드에서 지원되는 새로운 컨테이너 런타임이 됩니다.

### 5.3.1. Machine API 개요

Machine API는 업스트림 Cluster API 프로젝트 및 사용자 정의 OpenShift Container Platform 리소스를 기반으로 하는 주요 리소스의 조합입니다.

OpenShift Container Platform 4.9 클러스터의 경우 Machine API는 클러스터 설치가 완료된 후 모든 노드 호스트 프로비저닝 관리 작업을 수행합니다. 이 시스템으로 인해 OpenShift Container Platform 4.9는 퍼블릭 또는 프라이빗 클라우드 인프라에 더하여 탄력적이고 동적인 프로비저닝 방법을 제공합니다.

두 가지 주요 리소스는 다음과 같습니다.

#### Machine

노드의 호스트를 설명하는 기본 단위입니다. 머신에는 providerSpec 사양이 있으며 이는 다른 클라우드 플랫폼에 제공되는 컴퓨팅 노드 유형을 설명합니다. 예를 들어 AWS(Amazon Web Services)의 작업자 노드에 대한 머신 유형은 특정 머신 유형과 필요한 메타 데이터를 정의할 수 있습니다.

#### 머신 세트

MachineSet 리소스는 머신 그룹입니다. 머신 세트는 머신에 연관되어 있고 복제본 세트는 pod에 연관되어 있습니다. 더 많은 머신이 필요하거나 규모를 줄여야 하는 경우 컴퓨터 요구 사항에 맞게 머신 세트의 replicas 필드를 변경합니다.



주의

컨트롤 플레인 시스템은 머신 세트에서 관리할 수 없습니다.

다음 사용자 지정 리소스는 클러스터에 더 많은 기능을 추가할 수 있습니다.

### 머신 자동 스케일러

**MachineAutoscaler** 리소스는 클라우드에서 컴퓨팅 머신을 자동으로 확장합니다. 지정된 컴퓨팅 머신 세트에서 노드의 최소 및 최대 스케일링 경계를 설정할 수 있으며 머신 자동 스케일러는 해당 노드 범위를 유지합니다.

**MachineAutoscaler** 객체는 **ClusterAutoscaler** 객체를 설정한 후에 사용할 수 있습니다. **ClusterAutoscaler** 및 **MachineAutoscaler** 리소스는 모두 **ClusterAutoscalerOperator** 오브젝트에서 사용 가능합니다.

### Cluster autoscaler

이 리소스는 업스트림 클러스터 자동 스케일러 프로젝트를 기반으로 합니다. **OpenShift Container Platform** 구현에서는 머신 세트 API를 확장하여 **Machine API**와 통합됩니다. 코어, 노드, 메모리, GPU 등과 같은 리소스의 클러스터 전체에서 확장 제한을 설정할 수 있습니다. 중요도가 낮은 **Pod**에 대해 새 노드가 온라인 상태가 되지 않도록 클러스터가 **Pod**에 우선 순위를 설정할 수 있습니다. 노드를 확장할 수는 있지만 축소할 수 없도록 확장 정책을 설정할 수도 있습니다.

### 머신 상태 점검

**MachineHealthCheck** 리소스는 머신의 비정상적인 상태를 감지하여 삭제한 후 지원되는 플랫폼에서 새 머신을 생성합니다.

**OpenShift Container Platform** 버전 3.11에서는 클러스터가 머신 프로비저닝을 관리하지 않았기 때문에 다중 영역 아키텍처를 쉽게 돌아올 수 없었습니다. **OpenShift Container Platform** 버전 4.1부터 이러한 프로세스가 더 쉬워졌습니다. 각 머신 세트의 범위는 단일 영역에서 지정되므로 설치 프로그램은 사용자를 대신하여 가용성 영역 전체에 머신 세트를 보냅니다. 또한 계산이 동적이고 영역 장애가 발생하여 머신을 재조정해야 하는 경우 처리할 수 있는 영역을 확보할 수 있습니다. **Autoscaler**는 클러스터의 수명 기간 동안 최적의 균형을 유지합니다.

### 5.3.2. Windows 컨테이너 워크로드를 위한 vSphere 환경 준비

vSphere Windows VM 이미지를 생성하고 WMCO의 내부 API 서버와의 통신을 활성화하여 Windows 컨테이너 워크로드를 위한 vSphere 환경을 준비해야 합니다.

### 5.3.2.1. vSphere Windows VM 골든 이미지 생성

vSphere Windows VM(가상 머신) 골든 이미지를 생성합니다.

#### 사전 요구 사항

- OpenSSH 서버에서 키 기반 인증을 구성하는 데 사용되는 개인/공개 키 쌍을 생성했습니다. 개인 키도 WMCO(Windows Machine Config Operator) 네임스페이스에서 구성해야 합니다. 이는 WMCO가 Windows VM과 통신할 수 있도록 하는 데 필요합니다. 자세한 내용은 "Windows Machine Config Operator의 시크릿 구성" 섹션을 참조하십시오.



#### 참고

Windows VM을 생성할 때 여러 사례에서 Microsoft PowerShell 명령을 사용해야 합니다. 이 가이드의 PowerShell 명령은 PS C:\> 접두사로 구분됩니다.

#### 절차

1. Windows Server Semi-Annual Channel (SAC): Windows Server 20H2 ISO 이미지를 사용하여 vSphere 클라이언트에 새 VM을 생성합니다. KB4565351. 이 패치는 VXLAN UDP 포트를 설정하는데 필요하며, 이는 vSphere에 설치된 클러스터에 필요합니다. 자세한 내용은 VMware 설명서 를 참조하십시오.



#### 중요

VM의 가상 하드웨어 버전은 OpenShift Container Platform의 인프라 요구 사항을 충족해야 합니다. 자세한 내용은 OpenShift Container Platform 설명서의 "VMware vSphere 인프라 요구 사항" 섹션을 참조하십시오. 또한 가상 머신 하드웨어 버전에 대한 VMware 설명서를 참조하십시오.

2. Windows VM에서 VMware Tools 버전 11.0.6 이상을 설치 및 구성합니다. 자세한 내용은 VMware Tools 설명서를 참조하십시오.
3. Windows VM에 VMware Tools를 설치한 후 다음을 확인합니다.
  - a.



다음 항목과 함께 **C:\ProgramData\VMware\VMware Tools\tools.conf** 파일이 있습니다.

**exclude-nics=**

**tools.conf** 파일이 없는 경우 **exclude-nics** 옵션을 사용하여 주석 처리되지 않은 상태로 만들고 빈 값으로 설정합니다.

이 항목은 **hybrid-overlay**를 통해 **Windows VM**에서 생성된 복제 **vNIC**가 무시되지 않도록 합니다.

- b. **Windows VM**은 **vCenter**에 유효한 **IP** 주소가 있습니다.

**C:\> ipconfig**

- c. **VMTools Windows** 서비스가 실행 중입니다.

**PS C:\> Get-Service -Name VMTools | Select Status, StartType**

4. **Windows VM**에 **OpenSSH** 서버를 설치하고 구성합니다. 자세한 내용은 **Microsoft의 OpenSSH 설치** 설명서를 참조하십시오.
5. 관리자에 대한 **SSH** 액세스를 설정합니다. 이를 수행하려면 관리 **사용자에 대한 Microsoft** 문서를 참조하십시오.



#### 중요

지침에 사용된 공개 키는 보안을 보유하는 **WMCO** 네임스페이스의 뒷부분에 생성한 개인 키에 해당해야 합니다. 자세한 내용은 "**Windows Machine Config Operator**의 시크릿 구성" 섹션을 참조하십시오.

6. **Microsoft 설명서**에 따라 **Windows VM**에 **Docker** 컨테이너 런타임을 설치합니다.
7. 컨테이너 로그에 대해 들어오는 연결을 허용하는 **Windows VM**에서 새 방화벽 규칙을 생성해야 합니다. 다음 **PowerShell** 명령을 실행하여 **TCP** 포트 **10250**에서 방화벽 규칙을 생성합니다.

다.

```
PS C:\> New-NetFirewallRule -DisplayName "ContainerLogsPort" -LocalPort 10250 -
Enabled True -Direction Inbound -Protocol TCP -Action Allow -EdgeTraversalPolicy
Allow
```

8.

Windows VM을 복제하여 재사용 가능한 이미지입니다. 자세한 내용은 [기존 가상 시스템을 복제하는 방법](#)에 대한 VMware 설명서를 따르십시오.

9.

복제된 Windows VM에서 [Windows Sysprep](#) 도구를 실행하십시오 :

```
C:\> C:\Windows\System32\Sysprep\sysprep.exe /generalize /oobe /shutdown
/unattend:<path_to_unattend.xml> 1
```

1

unattend.xml 파일의 경로를 지정합니다.



참고

Windows 이미지에서 sysprep 명령을 실행할 수 있는 횟수에 제한이 있습니다. 자세한 내용은 [Microsoft 설명서](#)를 참조하십시오.

WMCO에 필요한 모든 변경 사항을 유지관리하는 unattend.xml 예시가 제공됩니다. 이 예제를 수정해야 하며 직접 사용할 수 없습니다.

예 5.1. 예시 unattend.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<unattend xmlns="urn:schemas-microsoft-com:unattend">
  <settings pass="specialize">
    <component xmlns:wcm="http://schemas.microsoft.com/WMIConfig/2002/State"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="Microsoft-Windows-
      International-Core" processorArchitecture="amd64"
      publicKeyToken="31bf3856ad364e35" language="neutral" versionScope="nonSxS">
      <InputLocale>0409:00000409</InputLocale>
      <SystemLocale>en-US</SystemLocale>
      <UILanguage>en-US</UILanguage>
      <UILanguageFallback>en-US</UILanguageFallback>
      <UserLocale>en-US</UserLocale>
    </component>
    <component xmlns:wcm="http://schemas.microsoft.com/WMIConfig/2002/State"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="Microsoft-Windows-
```

```

Security-SPP-UX" processorArchitecture="amd64" publicKeyToken="31bf3856ad364e35"
language="neutral" versionScope="nonSxS">
  <SkipAutoActivation>true</SkipAutoActivation>
</component>
  <component xmlns:wcm="http://schemas.microsoft.com/WMIConfig/2002/State"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="Microsoft-Windows-
SQMApi" processorArchitecture="amd64" publicKeyToken="31bf3856ad364e35"
language="neutral" versionScope="nonSxS">
  <CEIPEEnabled>0</CEIPEEnabled>
</component>
  <component xmlns:wcm="http://schemas.microsoft.com/WMIConfig/2002/State"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="Microsoft-Windows-
Shell-Setup" processorArchitecture="amd64" publicKeyToken="31bf3856ad364e35"
language="neutral" versionScope="nonSxS">
  <ComputerName>winhost</ComputerName> 1
</component>
</settings>
<settings pass="oobeSystem">
  <component xmlns:wcm="http://schemas.microsoft.com/WMIConfig/2002/State"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="Microsoft-Windows-
Shell-Setup" processorArchitecture="amd64" publicKeyToken="31bf3856ad364e35"
language="neutral" versionScope="nonSxS">
  <AutoLogon>
    <Enabled>>false</Enabled> 2
  </AutoLogon>
  <OOBE>
    <HideEULAPage>true</HideEULAPage>
    <HideLocalAccountScreen>true</HideLocalAccountScreen>
    <HideOEMRegistrationScreen>true</HideOEMRegistrationScreen>
    <HideOnlineAccountScreens>true</HideOnlineAccountScreens>
    <HideWirelessSetupInOOBE>true</HideWirelessSetupInOOBE>
    <NetworkLocation>Work</NetworkLocation>
    <ProtectYourPC>1</ProtectYourPC>
    <SkipMachineOOBE>true</SkipMachineOOBE>
    <SkipUserOOBE>true</SkipUserOOBE>
  </OOBE>
  <RegisteredOrganization>Organization</RegisteredOrganization>
  <RegisteredOwner>Owner</RegisteredOwner>
  <DisableAutoDaylightTimeSet>>false</DisableAutoDaylightTimeSet>
  <TimeZone>Eastern Standard Time</TimeZone>
  <UserAccounts>
    <AdministratorPassword>
      <Value>MyPassword</Value> 3
      <PlainText>true</PlainText>
    </AdministratorPassword>
  </UserAccounts>
</component>
</settings>
</unattend>

```

1

**Kubernetes** 이름 사양을 따라야 하는 **ComputerName** 을 지정합니다. 이러한 사양은 새 **VM**을 생성하는 동안 결과로 생성된 템플릿에서 수행되는 게스트 **OS** 사용자 지정에도 적용됩니다.

2

부팅 시 관리자 권한이 있는 열린 터미널을 종료하는 보안 문제를 방지하려면 자동 로그인을 비활성화합니다. 이 값은 기본값이며 변경해서는 안 됩니다.

3

**MyPassword** 자리 표시자를 **Administrator** 계정의 암호로 바꿉니다. 이렇게 하면 기본 제공 **Administrator** 계정에 기본적으로 빈 암호가 없습니다. [암호 선택에 대한 Microsoft의 모범 사례를](#) 따르십시오.

**Sysprep** 도구가 완료되면 **Windows VM**의 전원이 꺼집니다. 이 **VM**에서 더 이상 또는 전원을 사용하면 안 됩니다.

10.

**Windows VM**을 **vCenter**의 **템플릿**으로 변환합니다.

#### 5.3.2.1.1. 추가 리소스

- [Windows Machine Config Operator에 대한 시크릿 구성](#)
- [VMware vSphere 인프라 요구사항](#)

#### 5.3.2.2. vSphere에서 WMCO를 위한 내부 API 서버와의 통신 활성화

**WMCO**(**Windows Machine Config Operator**)는 내부 **API** 서버 끝점에서 **Ignition** 구성 파일을 다운로드합니다. **Windows** 가상 머신(**VM**)이 **Ignition** 구성 파일을 다운로드할 수 있고 구성된 **VM**의 **kubelet**이 내부 **API** 서버와 통신할 수 있도록 내부 **API** 서버와의 통신을 활성화해야 합니다.

#### 사전 요구 사항

- **vSphere**에 클러스터가 설치되어 있습니다.

#### 절차

- 외부 **API** 서버 URL **api.<cluster\_name>.<base\_domain>**을 가리키는 **api-int.<cluster\_name>.<base\_domain>**에 새로운 **DNS** 항목을 추가합니다. 이 항목은 **CNAME** 또는 추가 **A** 레코드일 수 있습니다.



## 참고

외부 API 끝점이 이미 vSphere의 초기 클러스터 설치의 일부로 생성되었습니다.

### 5.3.3. vSphere에서 Windows MachineSet 오브젝트를 위한 샘플 YAML

이 샘플 YAML은 VMware vSphere에서 WMCO(Windows Machine Config Operator)가 응답할 수 있는 Windows MachineSet 오브젝트를 정의합니다.

```

apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> 1
  name: <windows_machine_set_name> 2
  namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id> 3
      machine.openshift.io/cluster-api-machineset: <windows_machine_set_name> 4
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id> 5
        machine.openshift.io/cluster-api-machine-role: worker
        machine.openshift.io/cluster-api-machine-type: worker
        machine.openshift.io/cluster-api-machineset: <windows_machine_set_name> 6
        machine.openshift.io/os-id: Windows 7
    spec:
      metadata:
        labels:
          node-role.kubernetes.io/worker: "" 8
      providerSpec:
        value:
          apiVersion: vsphereprovider.openshift.io/v1beta1
          credentialsSecret:
            name: vsphere-cloud-credentials
          diskGiB: 128 9
          kind: VSphereMachineProviderSpec
          memoryMiB: 16384
          network:
            devices:
              - networkName: "<vm_network_name>" 10
          numCPUs: 4
          numCoresPerSocket: 1
          snapshot: ""
          template: <windows_vm_template_name> 11
          userDataSecret:

```

```

name: windows-user-data 12
workspace:
  datacenter: <vcenter_datacenter_name> 13
  datastore: <vcenter_datastore_name> 14
  folder: <vcenter_vm_folder_path> 15
  resourcePool: <vsphere_resource_pool> 16
  server: <vcenter_server_ip> 17
    
```

**1 3 5**

클러스터를 프로비저닝할 때 설정한 클러스터 ID를 기반으로 하는 인프라 ID를 지정합니다. 다음 명령을 실행하여 인프라 ID를 가져올 수 있습니다.

```
$ oc get -o jsonpath='{.status.infrastructureName}' infrastructure cluster
```

**2 4 6**

Windows 머신 세트의 이름을 지정합니다. 머신 세트 이름은 vSphere에서 머신 이름이 생성되는 방식으로 인해 9자를 초과할 수 없습니다.

**7**

머신 세트를 Windows 머신으로 구성합니다.

**8**

Windows 노드를 컴퓨팅 머신으로 구성합니다.

**9**

vSphere VMDK(가상 머신 디스크)의 크기를 지정합니다.



참고

이 매개변수는 Windows 파티션의 크기를 설정하지 않습니다. unattend.xml 파일을 사용하거나 필요한 디스크 크기로 vSphere Windows VM(가상 머신) 골든 이미지를 생성하여 Windows 파티션의 크기를 조정할 수 있습니다.

**10**

머신 세트를 배포할 vSphere VM 네트워크를 지정합니다. 이 VM 네트워크는 다른 Linux 컴퓨팅 시스템이 클러스터에 있는 위치여야 합니다.

**11**

사용할 Windows vSphere VM 템플릿의 전체 경로를 지정합니다(예: `green - images/windows-server-template`). 이름은 고유해야 합니다.



#### 중요

원래 VM 템플릿을 지정하지 마십시오. VM 템플릿이 꺼져 있어야 하며 새 Windows 머신에 대해 복제해야 합니다. VM 템플릿을 시작하면 VM 템플릿이 플랫폼의 VM으로 구성되므로 시스템 세트가 구성을 적용할 수 있는 템플릿으로 사용되지 않습니다.

12

`Windows-user-data`는 첫 번째 Windows 머신이 구성될 때 WMCO에 의해 생성됩니다. 이후에는, 모든 후속 머신 세트에서 `Windows-user-data`를 사용할 수 있습니다.

13

머신 세트를 배포할 vCenter Datacenter를 지정합니다.

14

머신 세트를 배포할 vCenter Datastore를 지정합니다.

15

vCenter의 vSphere VM 폴더에 경로(예: `/dc1/vm/user-inst-5ddjd`)를 지정합니다.

16

선택 사항: Windows VM을 위한 vSphere 리소스 풀을 지정합니다.

17

vCenter 서버 IP 또는 정규화된 도메인 이름을 지정합니다.

### 5.3.4. 머신 세트 만들기

설치 프로그램에서 생성한 컴퓨팅 머신 세트 외에도 고유한 머신 세트를 생성하여 선택한 특정 워크로드의 머신 컴퓨팅 리소스를 동적으로 관리할 수 있습니다.

사전 요구 사항

- **OpenShift Container Platform** 클러스터를 배포합니다.
- **OpenShift CLI(oc)**를 설치합니다.
- **cluster-admin** 권한이 있는 사용자로 **oc**에 로그인합니다.

절차

1. 머신 세트 **CR**(사용자 지정 리소스) 샘플이 포함된 이름이 **<file\_name>.yaml**인 새 **YAML** 파일을 만듭니다.

**<clusterID>** 및 **<role>** 매개 변수 값을 설정해야 합니다.

2. 선택 사항: 특정 필드에 설정할 값이 확실하지 않은 경우 클러스터에서 기존 컴퓨팅 머신 세트를 확인할 수 있습니다.

- a. 클러스터의 컴퓨팅 머신 세트를 나열하려면 다음 명령을 실행합니다.

```
$ oc get machinesets -n openshift-machine-api
```

출력 예

NAME	DESIRED	CURRENT	READY	AVAILABLE	AGE
agl030519-vplxk-worker-us-east-1a	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1b	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1c	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1d	0	0		55m	
agl030519-vplxk-worker-us-east-1e	0	0		55m	
agl030519-vplxk-worker-us-east-1f	0	0		55m	

- b. 특정 컴퓨팅 머신 세트 **CR**(사용자 정의 리소스)의 값을 보려면 다음 명령을 실행합니다.



```
$ oc get machineset <machineset_name> \
-n openshift-machine-api -o yaml
```

출력 예

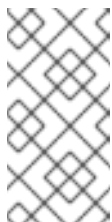
```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: <infrastructure_id> ❶
    name: <infrastructure_id>-<role> ❷
    namespace: openshift-machine-api
spec:
  replicas: 1
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: <infrastructure_id>
      machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: <infrastructure_id>
        machine.openshift.io/cluster-api-machine-role: <role>
        machine.openshift.io/cluster-api-machine-type: <role>
        machine.openshift.io/cluster-api-machineset: <infrastructure_id>-<role>
    spec:
      providerSpec: ❸
      ...
```

❶

클러스터 인프라 ID입니다.

❷

기본 노드 레이블입니다.



참고

사용자 프로비저닝 인프라가 있는 클러스터의 경우 컴퓨팅 머신 세트는 작업자 및 인프라 유형 머신만 생성할 수 있습니다.

❸

3. 다음 명령을 실행하여 **MachineSet CR**을 생성합니다.

```
$ oc create -f <file_name>.yaml
```

검증

- 다음 명령을 실행하여 컴퓨팅 머신 세트 목록을 확인합니다.

```
$ oc get machineset -n openshift-machine-api
```

출력 예

NAME	DESIRED	CURRENT	READY	AVAILABLE	AGE
agl030519-vplxk-windows-worker-us-east-1a	1	1	1	1	11m
agl030519-vplxk-worker-us-east-1a	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1b	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1c	1	1	1	1	55m
agl030519-vplxk-worker-us-east-1d	0	0			55m
agl030519-vplxk-worker-us-east-1e	0	0			55m
agl030519-vplxk-worker-us-east-1f	0	0			55m

새 머신 세트가 사용 가능한 경우 **DESIRED** 및 **CURRENT** 값이 일치합니다. 머신 세트를 사용할 수 없는 경우 몇 분 후에 명령을 다시 실행합니다.

5.3.5. 추가 리소스

- 머신 세트 관리에 대한 자세한 내용은 *머신 관리* 섹션을 참조하십시오.

## 6장. WINDOWS 컨테이너 워크로드 예약

Windows 워크로드를 Windows 컴퓨팅 노드에 예약할 수 있습니다.



### 참고

WMCO는 워크로드에 대한 프록시 연결을 통해 트래픽을 라우팅할 수 없기 때문에 **클러스터 전체 프록시**를 사용하는 클러스터에서는 WMCO가 지원되지 않습니다.

### 사전 요구 사항

- OLM(Operator Lifecycle Manager)을 사용하여 WMCO(Windows Machine Config Operator)를 설치했습니다.
- Docker 형식의 컨테이너 런타임 애드온이 활성화된 OS 이미지로 Windows 컨테이너를 사용하고 있습니다.
- Windows 머신 세트를 생성했습니다.



### 중요

현재는 Docker 형식의 컨테이너 런타임이 Windows 노드에서 사용됩니다. Kubernetes는 더 이상 Docker를 컨테이너 런타임으로 사용하지 않습니다. **Docker 사용 중지**에 대한 자세한 내용은 Kubernetes 문서를 참조하십시오. 향후 Kubernetes 릴리스에서는 Containerd가 Windows 노드에서 지원되는 새로운 컨테이너 런타임이 됩니다.

### 6.1. WINDOWS POD 배치

Windows 워크로드를 클러스터에 배포하기 전에 Pod가 올바르게 할당되도록 Windows 노드 스케줄링을 구성해야 합니다. Windows 노드를 호스팅하는 머신이 있으므로 Linux 기반 노드와 동일하게 관리됩니다. 유사하게, 테인트, 허용 오차, 노드 선택기와 같은 방법을 사용하여 적절한 Windows 노드에 Windows Pod도 예약되어야 합니다.

동일한 클러스터에서 여러 Windows OS 변형을 실행하는 기능과 동일한 클러스터에서 여러 Windows OS 변형을 실행할 수 있는 경우 RuntimeClass 오브젝트를 사용하여 Windows Pod를 기본 Windows OS 변형에 매핑해야 합니다. 예를 들어, 다른 Windows Server 컨테이너 버전에서 여러 Windows 노드가 있는 경우 클러스터는 호환되지 않는 Windows OS 변형에 Windows Pod를 예약할 수 있습니다. 클러

스터의 각 **Windows OS** 변형에 대해 **RuntimeClass** 오브젝트가 구성되어 있어야 합니다. 클러스터에서 사용 가능한 **Windows OS** 변형만 있는 경우에는 **RuntimeClass** 오브젝트를 사용하는 것이 좋습니다.

자세한 내용은 [호스트 및 컨테이너 버전 호환성](#)에 대한 **Microsoft** 문서를 참조하십시오.



**중요**

컨테이너 기본 이미지는 사용자가 예약해야 하는 노드에서 실행 중인 것과 동일한 **Windows OS** 버전이어야 합니다.

또한 **Windows** 노드를 한 버전에서 다른 버전으로 업그레이드하는 경우(예: **20H2**에서 **2022**로 이동) 컨테이너 기본 이미지를 새 버전과 일치시켜야 합니다. 자세한 내용은 [Windows 컨테이너 버전 호환성](#) 을 참조하십시오.

추가 리소스

- [스케줄러를 사용하여 Pod 배치 제어](#)
- [노드 테인트를 사용하여 Pod 배치 제어](#)
- [노드 선택기를 사용하여 특정 노드에 Pod 배치](#)

**6.2. 스케줄링 메커니즘을 캡슐화하기 위해 RUNTIMECLASS 오브젝트 생성**

**RuntimeClass** 오브젝트를 사용하면 테인트 및 허용 오차와 같은 스케줄링 방식을 편리하게 사용할 수 있습니다. 사용자는 테인트 및 허용 오차를 캡슐화하는 런타임 클래스를 배포한 후 이를 **Pod**에 적용하여 적절한 노드에 예약할 수 있습니다. 여러 운영 체제 변형을 지원하는 클러스터에도 런타임 클래스를 생성해야 합니다.

절차

1. **RuntimeClass** 오브젝트 **YAML** 파일을 생성합니다. 예를 들어, `runtime-class.yaml`은 다음과 같습니다.

```

apiVersion: node.k8s.io/v1beta1
kind: RuntimeClass
metadata:

```

```

name: <runtime_class_name> ❶
handler: 'docker'
scheduling:
  nodeSelector: ❷
    kubernetes.io/os: 'windows'
    kubernetes.io/arch: 'amd64'
    node.kubernetes.io/windows-build: '10.0.17763'
  tolerations: ❸
    - effect: NoSchedule
      key: os
      operator: Equal
      value: "Windows"

```

❶

**RuntimeClass** 오브젝트 이름을 지정하며, 이는 이 런타임 클래스로 관리할 **Pod**에 정의됩니다.

❷

이 런타임 클래스를 지원하는 노드에 존재해야 하는 레이블을 지정합니다. 이 런타임 클래스를 사용하는 **Pod**는 이 선택기와 일치하는 노드에만 예약할 수 있습니다. 런타임 클래스의 노드 선택기는 **Pod**의 기존 노드 선택기와 병합됩니다. 충돌이 발생하면 **Pod**를 노드에 예약할 수 없습니다.

❸

허용 중에 이 런타임 클래스와 함께 실행 중인 **pod**(중복 제외)에 추가하려면 허용 오차를 지정합니다. 이 작업을 수행하면 **Pod** 및 런타임 클래스에서 허용되는 노드 집합이 결합됩니다.

2.

**RuntimeClass** 오브젝트를 생성합니다.

```
$ oc create -f <file-name>.yaml
```

예를 들면 다음과 같습니다.

```
$ oc create -f runtime-class.yaml
```

3.

**Pod**에 **RuntimeClass** 오브젝트를 적용하여 적절한 운영 체제 변형에 예약되어 있는지 확인합니다.

```

apiVersion: v1
kind: Pod
metadata:

```

```

name: my-windows-pod
spec:
  runtimeClassName: <runtime_class_name> 1
  ...

```

1

Pod 예약을 관리할 런타임 클래스를 지정합니다.

### 6.3. 샘플 WINDOWS 컨테이너 워크로드 배포

Windows 컴퓨팅 노드를 사용 가능한 경우 Windows 컨테이너 워크로드를 클러스터에 배포할 수 있습니다.



참고

이 샘플 배포는 참조용으로만 제공됩니다.

예시 Service 오브젝트

```

apiVersion: v1
kind: Service
metadata:
  name: win-webserver
  labels:
    app: win-webserver
spec:
  ports:
    # the port that this service should serve on
  - port: 80
    targetPort: 80
  selector:
    app: win-webserver
  type: LoadBalancer

```

예시 Deployment 오브젝트

```

apiVersion: apps/v1

```

```

kind: Deployment
metadata:
  labels:
    app: win-webserver
    name: win-webserver
spec:
  selector:
    matchLabels:
      app: win-webserver
  replicas: 1
  template:
    metadata:
      labels:
        app: win-webserver
        name: win-webserver
    spec:
      tolerations:
        - key: "os"
          value: "Windows"
          Effect: "NoSchedule"
      containers:
        - name: windowswebserver
          image: mcr.microsoft.com/windows/servercore:ltsc2019
          imagePullPolicy: IfNotPresent
          command:
            - powershell.exe
            - -command
              - $listener = New-Object System.Net.HttpListener; $listener.Prefixes.Add('http://*:80/');
                $listener.Start(); Write-Host("Listening at http://*:80/"); while ($listener.IsListening) { $context =
                $listener.GetContext(); $response = $context.Response; $content='<html><body><H1>Red
                Hat OpenShift + Windows Container Workloads</H1></body></html>'; $buffer =
                [System.Text.Encoding]::UTF8.GetBytes($content); $response.ContentLength64 =
                $buffer.Length; $response.OutputStream.Write($buffer, 0, $buffer.Length); $response.Close();
                };
          securityContext:
            runAsNonRoot: false
            windowsOptions:
              runAsUserName: "ContainerAdministrator"
          nodeSelector:
            kubernetes.io/os: windows

```

#### 참고

`mcr.microsoft.com/powershell:<tag>` 컨테이너 이미지를 사용하는 경우 이 명령을 `pwsh.exe`로 정의해야 합니다. `mcr.microsoft.com/windows/servercore:<tag>` 컨테이너 이미지를 사용하는 경우 해당 명령을 `powershell.exe`로 정의해야 합니다. 자세한 내용은 Microsoft 문서를 참조하십시오.

머신 세트에서 머신 인스턴스를 추가하거나 제거하려면 머신 세트를 수동으로 스케일링할 수 있습니다.

이는 완전히 자동화된 설치 프로그램에 의해 프로비저닝된 인프라 설치와 관련이 있습니다. 사용자 지정된 사용자 프로비저닝 인프라 설치에는 머신 세트가 없습니다.

#### 사전 요구 사항

- **OpenShift Container Platform** 클러스터 및 **oc** 명령행을 설치합니다.
- **cluster-admin** 권한이 있는 사용자로 **oc**에 로그인합니다.

#### 절차

1. 클러스터에 있는 머신 세트를 확인합니다.

```
$ oc get machinesets -n openshift-machine-api
```

머신 세트는 <clusterid>-worker-<aws-region-az> 형식으로 나열됩니다.

2. 클러스터에 있는 머신을 확인합니다.

```
$ oc get machine -n openshift-machine-api
```

3. 삭제하려는 머신에 주석을 설정합니다.

```
$ oc annotate machine/<machine_name> -n openshift-machine-api  
machine.openshift.io/cluster-api-delete-machine="true"
```

4. 삭제하려는 노드를 비우고 제외합니다.

```
$ oc adm cordon <node_name>  
$ oc adm drain <node_name>
```



5.

머신 세트를 스케일링합니다.

```
$ oc scale --replicas=2 machineset <machineset> -n openshift-machine-api
```

또는 다음을 수행합니다.

```
$ oc edit machineset <machineset> -n openshift-machine-api
```

작은 정보

다음 YAML을 적용하여 머신 세트를 확장할 수도 있습니다.

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  name: <machineset>
  namespace: openshift-machine-api
spec:
  replicas: 2
```

머신 세트를 확장 또는 축소할 수 있습니다. 새 머신을 사용할 수 있을 때 까지 몇 분 정도 소요 됩니다.

검증

- 원하는 머신 삭제를 확인합니다.

```
$ oc get machines
```

## 7장. WINDOWS 노드 업그레이드

**WMCO(Windows Machine Config Operator)**를 업그레이드하여 **Windows** 노드에 최신 업데이트가 있는지 확인할 수 있습니다.

### 7.1. WINDOWS MACHINE CONFIG OPERATOR 업그레이드

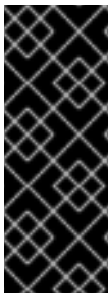
현재 클러스터 버전과 호환되는 새로운 버전의 **WMCO(Windows Machine Config Operator)**가 릴리스되면 **Operator**는 **OLM(Operator Lifecycle Manager)**을 사용할 때 함께 설치된 업그레이드 채널과 서브스크립션 승인 전략을 기반으로 업그레이드됩니다. **WMCO** 업그레이드로 인해 **Windows** 머신의 **Kubernetes** 구성 요소가 업그레이드됩니다.



#### 참고

새로운 버전의 **WMCO**로 업그레이드하고 클러스터 모니터링을 사용하려면 **WMCO** 네임스페이스에 `openshift.io/cluster-monitoring=true` 레이블이 지정되어야 합니다. 기존 **WMCO** 네임스페이스에 레이블을 추가하고 이미 **Windows** 노드가 구성된 경우 **WMCO Pod**를 다시 시작하여 그래프 모니터링을 허용합니다.

장치를 중단할 필요가 없는 업그레이드의 경우 **WMCO**는 이전 버전의 **WMCO**에 의해 구성된 **Windows** 머신을 종료하고 현재 버전을 사용하여 이를 다시 생성합니다. 이는 머신 오브젝트를 삭제하여 수행되며, 이로 인해 **Windows** 노드가 드레이닝 및 삭제됩니다. 편리한 업그레이드를 위해, **WMCO**는 모든 구성된 노드에 버전 주석을 추가합니다. 업그레이드 중에 버전 주석이 일치하지 않으면 **Windows** 머신이 삭제되고 다시 생성됩니다. 업그레이드하는 동안 서비스 중단을 최소화하기 위해 **WMCO**는 한 번에 하나의 **Windows** 머신만 업데이트합니다.



#### 중요

**WMCO**는 **Windows** 운영 체제 업데이트가 아닌 **Kubernetes** 구성 요소를 업데이트해야 합니다. **VM**을 생성할 때 **Windows** 이미지를 제공하므로 업데이트된 이미지를 제공해야 합니다. **MachineSet** 사양에서 이미지 구성을 변경하여 업데이트된 **Windows** 이미지를 제공할 수 있습니다.

**OLM(Operator Lifecycle Manager)**을 사용한 **Operator** 업그레이드에 대한 자세한 내용은 [설치된 Operator 업데이트](#)를 참조하십시오.

## 8장. BYOH (BRING-YOUR-OWN-HOST) WINDOWS 인스턴스를 노드로 사용

BYOH (Bring-Your-Own-Host) 를 사용하면 Windows Server VM의 용도를 변경하여 OpenShift Container Platform에 가져올 수 있습니다. BYOH Windows 인스턴스는 Windows 서버가 오프라인 상태가 되는 경우 주요 중단을 완화하려는 사용자에게 유용합니다.

### 8.1. BYOH WINDOWS 인스턴스 구성

BYOH Windows 인스턴스를 생성하려면 WMCO(Windows Machine Config Operator) 네임스페이스에 구성 맵을 생성해야 합니다.

#### 사전 요구 사항

노드에 따라 클러스터에 연결할 Windows 인스턴스는 다음 요구사항을 충족해야 합니다.

- Docker 컨테이너 런타임은 인스턴스에 설치해야 합니다.
- 인스턴스는 클러스터의 Linux 작업자 노드와 동일한 네트워크에 있어야 합니다.
- 포트 22가 열려 있어야 하며 SSH 서버를 실행 중이어야 합니다.
- SSH 서버의 기본 셸은 Windows 명령 셸 또는 cmd.exe여야 합니다.
- 로그 수집을 위해 포트 10250이 열려 있어야 합니다.
- 관리자는 인증된 SSH 키로 설정된 시크릿에 사용되는 개인 키가 있습니다.
- 설치 관리자 프로비저닝 인프라(IPI) AWS 클러스터에 대해 BYOH Windows 인스턴스를 생성하는 경우 작업자 노드의 머신 세트의 spec.template.spec.value.tag 값과 일치하는 AWS 인스턴스에 태그를 추가해야 합니다. 예를 들어 kubernetes.io/cluster/<cluster\_id>: owned 또는 kubernetes.io/cluster/<cluster\_id>: shared.
- vSphere에서 BH Windows 인스턴스를 생성하는 경우 내부 API 서버와의 통신을 활성화해야 합니다.

- 인스턴스의 호스트 이름은 다음 표준을 포함하는 **RFC 1123 DNS 레이블** 요구 사항을 따라야 합니다.
  - 소문자 영숫자 또는 '-'만 포함합니다.
  - 영숫자 문자로 시작합니다.
  - 영숫자 문자로 끝납니다.

절차

1. 추가할 **Windows** 인스턴스를 설명하는 **WMCO** 네임스페이스에 **windows-instances**라는 **ConfigMap**을 생성합니다.



참고

**username=<username>** 으로 포맷하는 동안 주소를 키로 사용하여 구성 맵의 데이터 섹션에서 각 항목을 포맷합니다.

구성 맵 예

```

kind: ConfigMap
apiVersion: v1
metadata:
  name: windows-instances
  namespace: openshift-windows-machine-config-operator
data:
  10.1.42.1: |- 1
    username=Administrator 2
  instance.example.com: |-
    username=core

```

1

**WMCO**가 **DNS** 이름 또는 **IPv4** 주소 중 **SSH**를 통해 인스턴스에 연결하는 데 사용하는 주소입니다. 이 주소의 **DNS PTR** 레코드가 있어야 합니다. 조직에서 **DHCP**를 사용하여 **IP**

주소를 할당하는 경우 **BYOH** 인스턴스와 함께 **DNS** 이름을 사용하는 것이 좋습니다. 그렇지 않은 경우 인스턴스에 새 **IP** 주소가 할당될 때마다 **windows-instances ConfigMap**을 업데이트해야 합니다.

**2**

사전 요구 사항에서 생성된 관리자의 이름입니다.

## 8.2. BYOH WINDOWS 인스턴스 제거

구성 맵에서 인스턴스의 항목을 삭제하여 클러스터에 연결된 **BYOH** 인스턴스를 제거할 수 있습니다. 인스턴스를 삭제하면 해당 인스턴스가 클러스터에 추가되기 전의 상태로 되돌아갑니다. 이러한 인스턴스에 로그 및 컨테이너 런타임 아티팩트가 추가되지 않습니다.

인스턴스를 완전히 제거하려면 **WMCO**에 제공된 현재 개인 키로 액세스할 수 있어야 합니다. 예를 들어 이전 예제에서 **10.1.42.1** 인스턴스를 제거하려면 구성 맵이 다음으로 변경됩니다.

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: windows-instances
  namespace: openshift-windows-machine-config-operator
data:
  instance.example.com: |-
    username=core
```

**windows-instances** 삭제는 노드에 추가된 모든 **Windows** 인스턴스를 해체하라는 요청으로 간주됩니다.

## 9장. WINDOWS 노드 제거

호스트 **Windows** 머신을 삭제하여 **Windows** 노드를 제거할 수 있습니다.

### 9.1. 특정 머신 삭제

특정 머신을 삭제할 수 있습니다.



참고

컨트롤 플레인 머신을 삭제할 수 없습니다.

#### 사전 요구 사항

- **OpenShift Container Platform** 클러스터를 설치합니다.
- **OpenShift CLI(oc)**를 설치합니다.
- **cluster-admin** 권한이 있는 사용자로 **oc**에 로그인합니다.

#### 절차

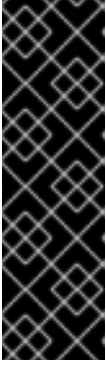
1. 클러스터에있는 머신을 확인하고 삭제할 머신을 식별합니다.

```
$ oc get machine -n openshift-machine-api
```

명령 출력에는 <clusterid>-worker-<cloud\_region> 형식의 머신 목록이 포함되어 있습니다.

2. 머신을 삭제합니다.

```
$ oc delete machine <machine> -n openshift-machine-api
```



## 중요

기본적으로 머신 컨트롤러는 성공할 때까지 머신이 지원하는 노드를 드레인하려고 합니다. 일부 경우, Pod의 중단 예산을 잘못 구성하는 등의 경우와 같이 노드 드레인 작업으로 인해 머신이 삭제되지 않을 수 있습니다. 특정 머신에서 **"machine.openshift.io/exclude-node-draining"**에 주석을 사용하여 노드 드레인 프로세스를 건너 뛸 수 있습니다. 삭제 중인 머신이 머신 세트에 속하는 경우 지정된 복제본 수를 충족하기 위해 새 머신이 즉시 생성됩니다.

## 10장. WINDOWS 컨테이너 워크로드 비활성화

**WMCO(Windows Machine Config Operator)**의 설치를 제거하고 **WMCO**를 설치할 때 기본적으로 추가된 네임스페이스를 삭제하여 **Windows** 컨테이너 워크로드를 실행하는 기능을 비활성화할 수 있습니다.

### 10.1. WINDOWS MACHINE CONFIG OPERATOR 제거

클러스터에서 **WMCO(Windows Machine Config Operator)**의 설치를 제거할 수 있습니다.

#### 사전 요구 사항

- **Windows** 워크로드를 호스팅하는 **Windows** 머신 오브젝트를 삭제합니다.

#### 절차

1. **Operators** → **OperatorHub** 페이지에서 키워드로 필터링 상자를 사용하여 **Red Hat Windows Machine Config Operator**를 검색합니다.
2. **Red Hat Windows Machine Config Operator** 타일을 클릭합니다. **Operator** 타일은 **Operator**가 설치되었음을 나타냅니다.
3. **Windows Machine Config Operator** 설명자 페이지에서 제거를 클릭합니다.

### 10.2. WINDOWS MACHINE CONFIG OPERATOR 네임스페이스 삭제

기본적으로 **WMCO(Windows Machine Config Operator)**에 대해 생성된 네임스페이스를 삭제할 수 있습니다.

#### 사전 요구 사항

- **WMCO**가 클러스터에서 제거됩니다.

#### 절차

1. **openshift-windows-machine-config-operator** 네임스페이스에서 생성된 모든 **Windows** 위



크로드를 제거합니다.

```
$ oc delete --all pods --namespace=openshift-windows-machine-config-operator
```

2.

**openshift-windows-machine-config-operator** 네임스페이스의 모든 **Pod**가 삭제되거나 종료 상태를 보고하는지 확인합니다.

```
$ oc get pods --namespace openshift-windows-machine-config-operator
```

3.

**openshift-windows-machine-config-operator** 네임스페이스를 삭제합니다.

```
$ oc delete namespace openshift-windows-machine-config-operator
```

추가 리소스

- [클러스터에서 Operator 삭제](#)
- [Windows 노드 제거](#)