Up and Running: Windows Containers with OpenShift

18 March 2021

Angela Vuong
Red Hat Solution Architect

Hanvitha Gavini
Red Hat Solution Architect

Faizal Khader
Red Hat Specialist Solution Architect

Agenda -

Theory

01

- Windows Containers
- Why OpenShift Windows Containers
- OpenShift stack - with Windows
- Use cases

Technique

02

- Windows Machine config Operator
- WMCO Platform Support
- WMCO Workflow
- Targeting .Net workload

Demo

03 Demo-1: OpenShift IPI
Installation PreRequisites

04 Demo-2: WMCO Installation

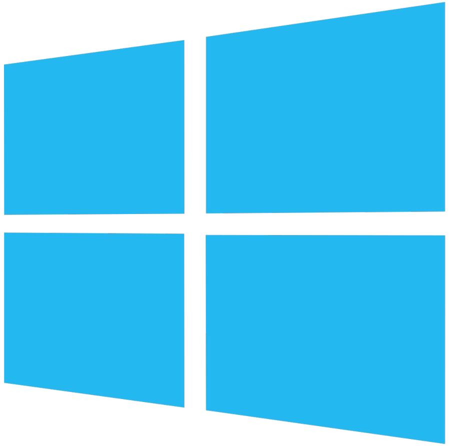
05 Demo-3: First Windows Containers

06 Demo-4: Mixing Windows & Linux Workloads

Messaging and the need for Windows Containers

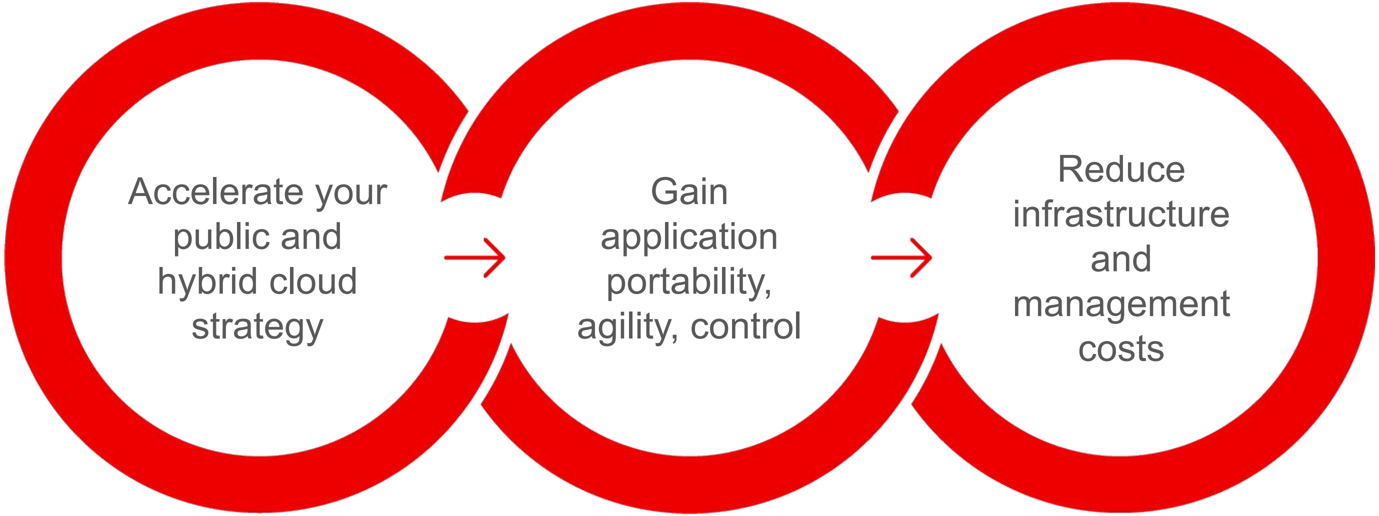
Red Hat OpenShift for Windows Containers

Background



- Windows Server still enjoys significant presence amongst server operating systems in the data center
- .NET has been and continues to be used widely for application development
- Traditionally Windows ran largely independent of Linux
- Adoption of microservices and containers requires Windows to embrace open source and Linux-based technologies
- To fully embrace containers and microservices Windows-based machines must now:
  - Lift legacy workloads
  - Containerize legacy Windows workloads
  - Strangle the monolith and support hybrid deployments

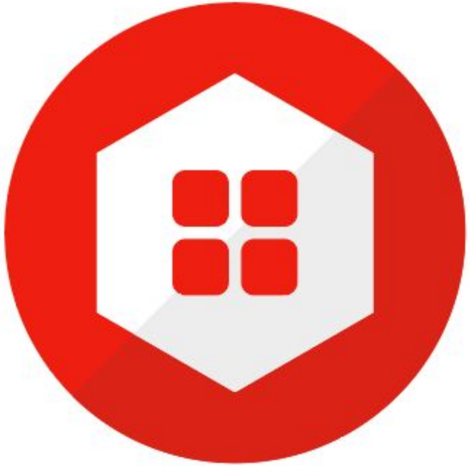
Why run Windows in containers?



Accelerate your public and hybrid cloud strategy → Gain application portability, agility, control → Reduce infrastructure and management costs

Red Hat

Why Red Hat OpenShift for Windows Containers



Realize the benefits of containers

Application portability, speed, flexibility

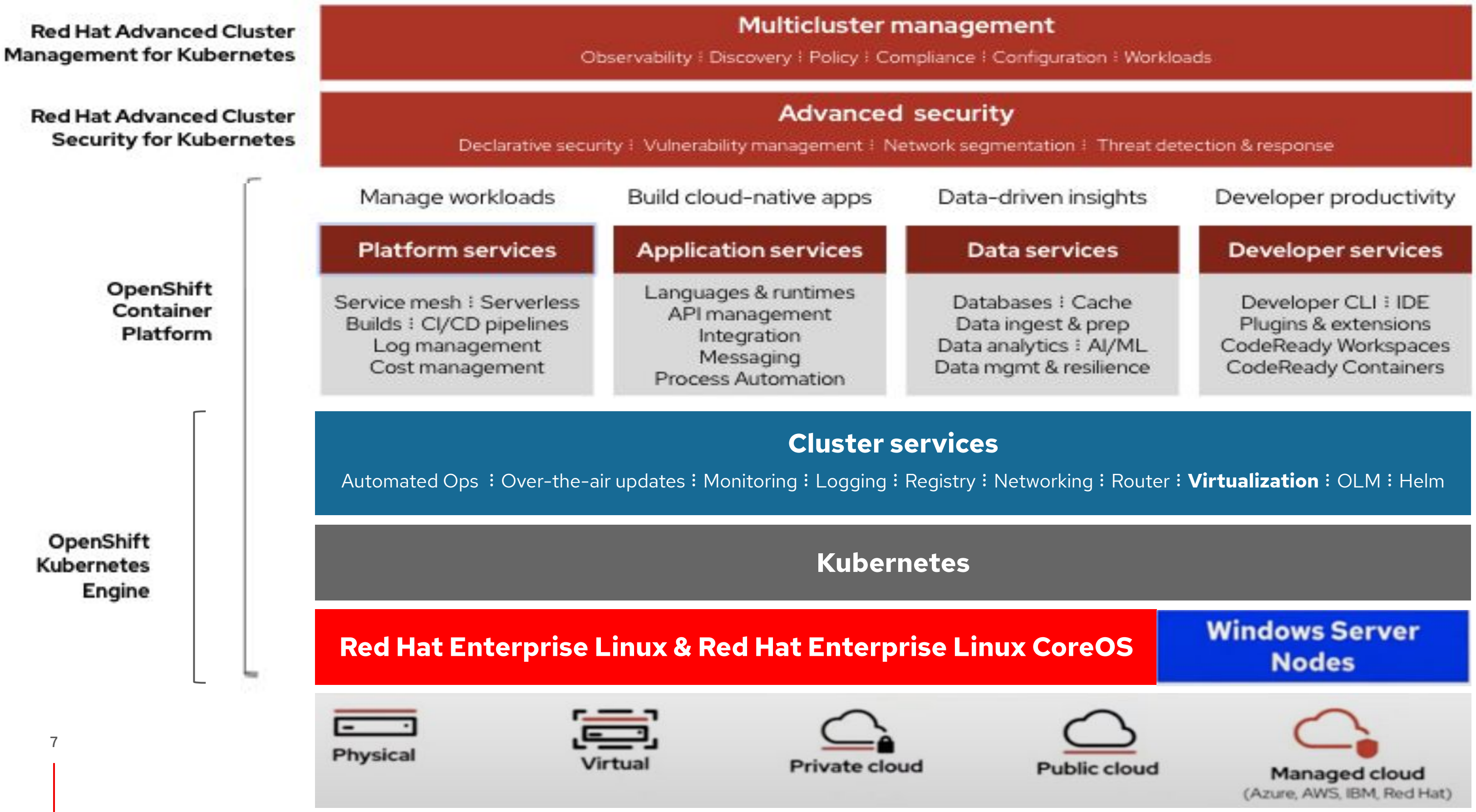
Modernize and gain efficiencies

Support legacy workloads efficiently

Developer Productivity

Get exposure to Kubernetes without having to rebuild applications

OpenShift Container Platform complete Stack



Red Hat Advanced Cluster Management for Kubernetes

Multicluster management

Observability ⁝ Discovery ⁝ Policy ⁝ Compliance ⁝ Configuration ⁝ Workloads

Red Hat Advanced Cluster Security for Kubernetes

Advanced security

Declarative security ⁝ Vulnerability management ⁝ Network segmentation ⁝ Threat detection & response

OpenShift Container Platform

Manage workloads	Build cloud-native apps	Data-driven insights	Developer productivity
Platform services	**Application services**	**Data services**	**Developer services**
Service mesh ⁝ Serverless Builds ⁝ CI/CD pipelines Log management Cost management	Languages & runtimes API management Integration Messaging Process Automation	Databases ⁝ Cache Data ingest & prep Data analytics ⁝ AI/ML Data mgmt & resilience	Developer CLI ⁝ IDE Plugins & extensions CodeReady Workspaces CodeReady Containers

OpenShift Kubernetes Engine

Cluster services

Automated Ops ⁝ Over-the-air updates ⁝ Monitoring ⁝ Logging ⁝ Registry ⁝ Networking ⁝ Router ⁝ **Virtualization** ⁝ OLM ⁝ Helm

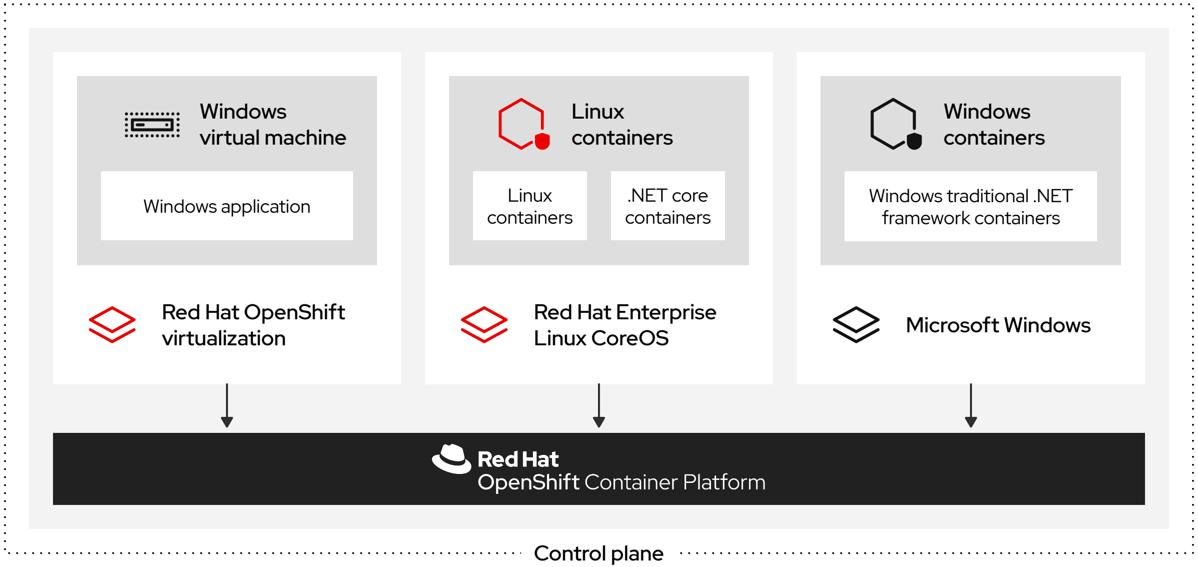
Kubernetes

Red Hat Enterprise Linux & Red Hat Enterprise Linux CoreOS

Windows Server Nodes

Mixed Windows and Linux workloads

Mixed Windows and Linux workloads

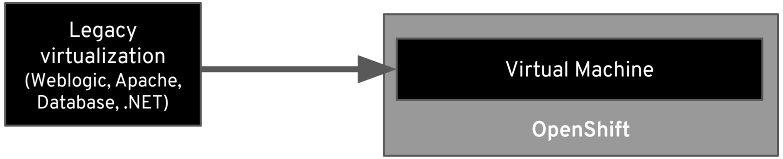

Windows virtual machine
Windows application
Red Hat OpenShift virtualization
Linux containers
Linux containers
.NET core containers
Red Hat Enterprise Linux CoreOS
Windows containers
Windows traditional .NET framework containers
Microsoft Windows
Red Hat OpenShift Container Platform
Control plane

- Run Linux containers on RHEL
- Run .NET core containers on RHEL
- Run traditional **.NET framework containers on Windows**
- Run **Windows VMs with CNV** (Container Native Virtualization)
- All scheduled and managed by Red Hat OpenShift

Windows for Containers or Virtualization?

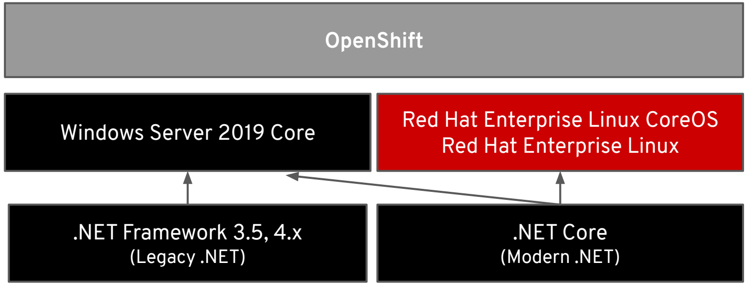
OpenShift Virtualization

- **Rehost** existing virtual machines within OpenShift with the goal of modernizing applications over time without having to rebuild



Windows for Containers

- **Refactor** traditional .NET applications on Windows Server Containers and deploy to Windows nodes on OpenShift

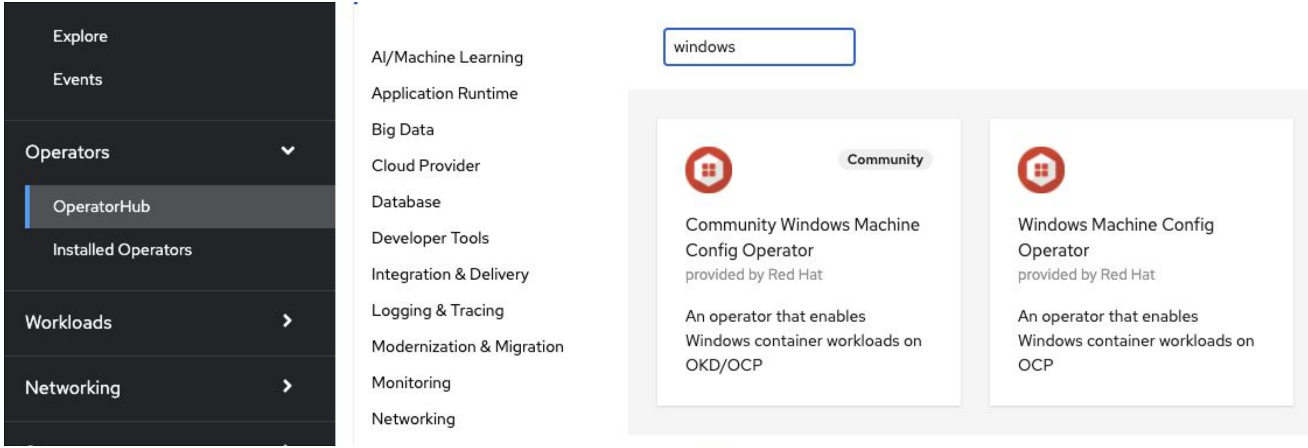


Use cases for Windows container workloads on OpenShift

Step	OpenShift Feature	Use case	Advantages	Trade Offs
Rehost	OpenShift Virtualization	Lift & Shift Windows VMs to OpenShift	Easy and low friction	No benefits of containerization
Refactor	Windows Machine Config Operator	Containerize and run traditional .Net framework apps on Windows Server Containers and deploy to Windows worker nodes on OCP	Benefits of containerization & OpenShift	Evolving Windows container ecosystem, supported only for newer version of Windows including Windows Server 2019
Rearchitect	RHEL/RHCOS containers	Migrate traditional .Net frameworks apps to .Net Core and deploy to RHEL containers in OpenShift.	Full benefit of containerization and OpenShift, highly evolved community	Migration effort involved, time consuming
Rebuild	RHEL/RHCOS containers	Build Cloud Native apps using Linux containers and deploy to RHEL/RHELCoreOS on OpenShift.	Full benefit of containerization and OpenShift highly evolved community	Net new development may not be an option for customers running in maintenance mode

Technique

Windows Machine Config Operator: Available in Cluster Operator Hub

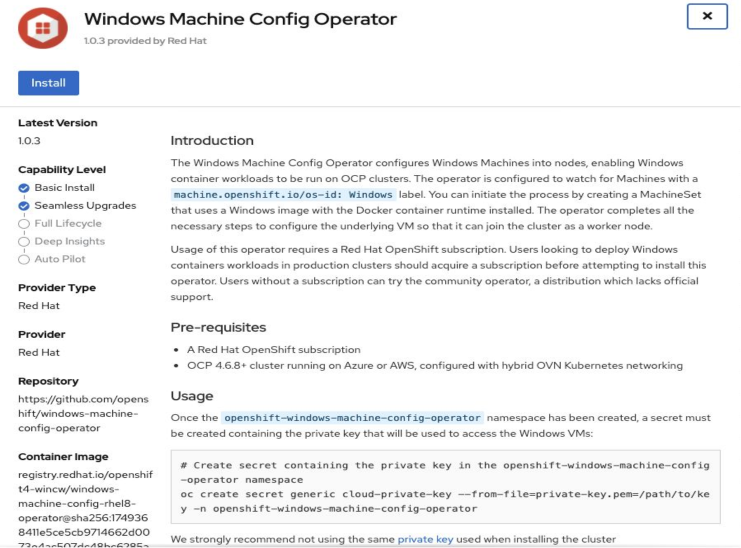


Community Windows Machine Config Operator
provided by Red Hat

An operator that enables Windows container workloads on OKD/OCP

Windows Machine Config Operator
provided by Red Hat

An operator that enables Windows container workloads on OCP



Windows Machine Config Operator
1.0.3 provided by Red Hat

Latest Version
1.0.3

Capability Level
- ● Basic Install
- ● Seamless Upgrades
- ○ Full Lifecycle
- ○ Deep Insights
- ○ Auto Pilot

Provider Type
Red Hat

Provider
Red Hat

Repository
https://github.com/openshift/windows-machine-config-operator

Container Image
registry.redhat.io/openshift4-wincw/windows-machine-config-rhel8-operator@sha256:1749368411e5ce5cb9714662d0073e4ac507dc48bc6285a

Introduction

The Windows Machine Config Operator configures Windows Machines into nodes, enabling Windows container workloads to be run on OCP clusters. The operator is configured to watch for Machines with a `machine.openshift.io/os-id: Windows` label. You can initiate the process by creating a MachineSet that uses a Windows image with the Docker container runtime installed. The operator completes all the necessary steps to configure the underlying VM so that it can join the cluster as a worker node.

Usage of this operator requires a Red Hat OpenShift subscription. Users looking to deploy Windows containers workloads in production clusters should acquire a subscription before attempting to install this operator. Users without a subscription can try the community operator, a distribution which lacks official support.

Pre-requisites

- A Red Hat OpenShift subscription
- OCP 4.6.8+ cluster running on Azure or AWS, configured with hybrid OVN Kubernetes networking

Usage

Once the `openshift-windows-machine-config-operator` namespace has been created, a secret must be created containing the private key that will be used to access the Windows VMs:

```
# Create secret containing the private key in the openshift-windows-machine-config
-operator namespace
oc create secret generic cloud-private-key --from-file=private-key.pem=/path/to/ke
y -n openshift-windows-machine-config-operator
```

We strongly recommend not using the same private key used when installing the cluster

Entry Point

The Windows Machine Config Operator (WMCO) is the entry point for OpenShift customers who want to run Windows workloads on their clusters.

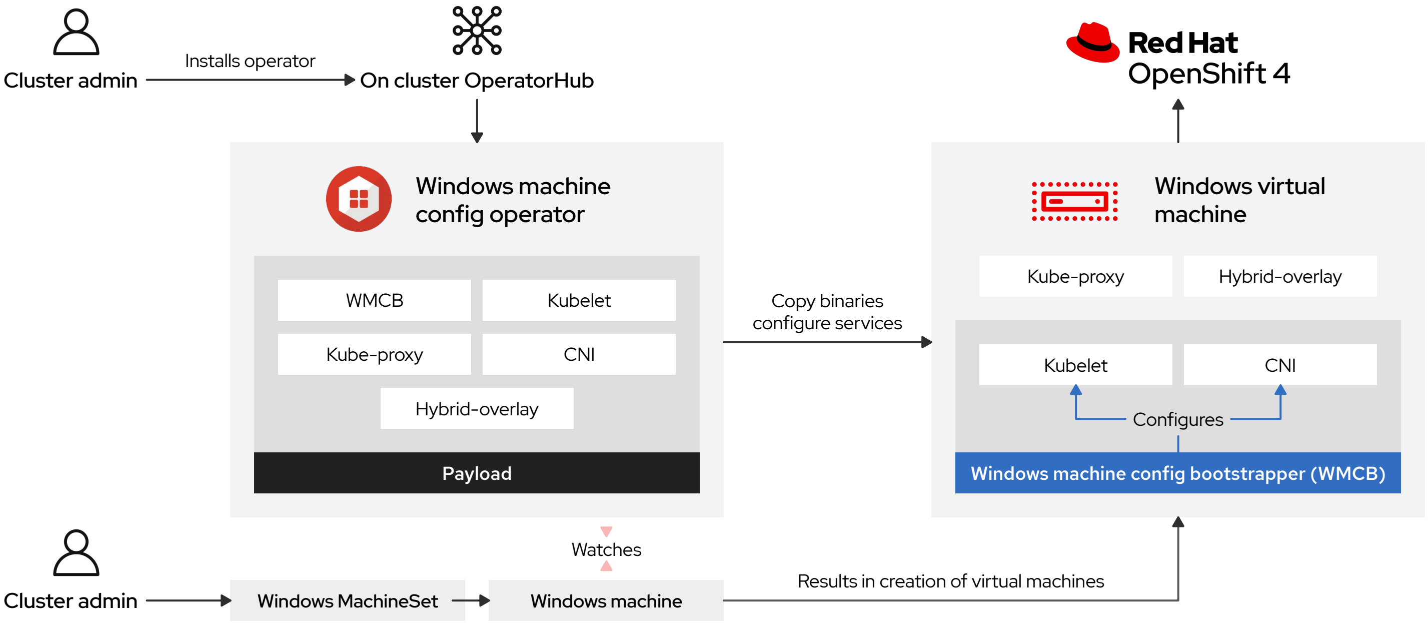
Day 2 Operations

The intent of this feature is to allow a cluster administrator to add a Windows worker node as a day 2 operation with a prescribed configuration to an installer provisioned OpenShift 4.6+ cluster and enable scheduling of Windows workloads.

OVN-Hybrid

The Prerequisite is an OpenShift 4.6+ cluster configured with hybrid OVN Kubernetes networking.

Windows Machine Config Operator Architecture



Red Hat

Windows Machine Config Operator Workflow

Install Operator

Navigate to the in-cluster Operator Hub and search for the Windows Operator and Click Install

Create a MachineSet

- The way a user will initiate the process is by creating a MachineSet which uses a Windows image with the Docker container runtime installed.
- It usually takes about 15 minutes for the Windows Machine to be configured as a worker node. Ensure the Windows Node is in a Ready state before deploying a workload

Operator configures machines to worker nodes

- The operator is configured to watch for Machines with a **machine.openshift.io/os-id**: Windows label.
- The operator will do all the necessary steps to configure the underlying VM so that it can join the cluster as a worker node.

Windows Machine Config Operator (WMCO) Workflow

Transfer binaries

Includes Windows machine config bootstrapper

Configure kubelet

Remotely execute WMCB to configure kubelet

Run hybrid-overlay

Create Red Hat Openshift HNS network

Configure CNI

Configure kubelet for CNI plugin

Set up kube-proxy

Maintains network rules on nodes allowing outside communication

Red Hat OpenShift for Windows Containers Supported Platforms

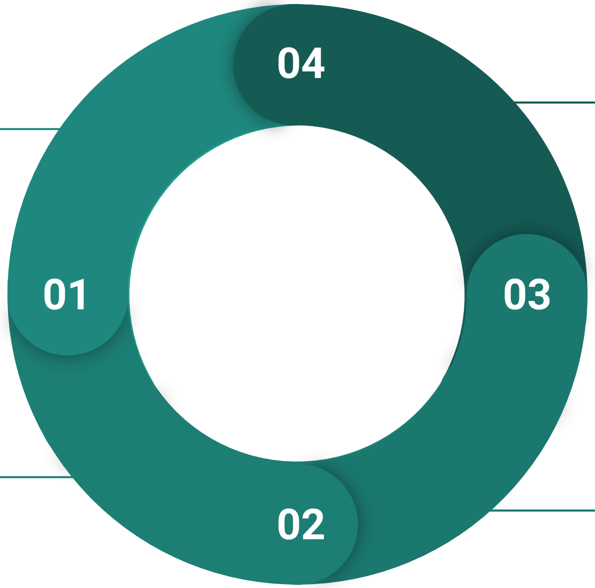
Platform	Supported	Coming Soon
Azure	Yes	
AWS	Yes	
vSphere	Yes (OCP 4.7)	
Bare metal	No	Yes
Red Hat Virtualization	No	Yes
OpenStack	No	Yes
Host Offerings (Azure Red Hat OpenShift etc)	No	Yes

Supported Operating Systems for Windows Worker Nodes

The following Windows Server operating systems are supported in the initial release of the WMCO: Windows Server Long-Term Servicing Channel (LTSC): **Windows Server 2019***

** Has to be on version 10.0.17763.1457 or older*

Windows Machine Config Operator Upgrade Process



New Version Released

If the current cluster version fulfills the minimum Kubernetes version requirement, OLM upgrades WMCO. If the cluster version is not high enough, the WMCO upgrade will occur once it is.

Annotate

The new WMCO reconciles as usual, ensuring that all unconfigured Windows Machines are configured and join the cluster as a node. Each of them are given an annotation indicating the WMCO version that configured them.

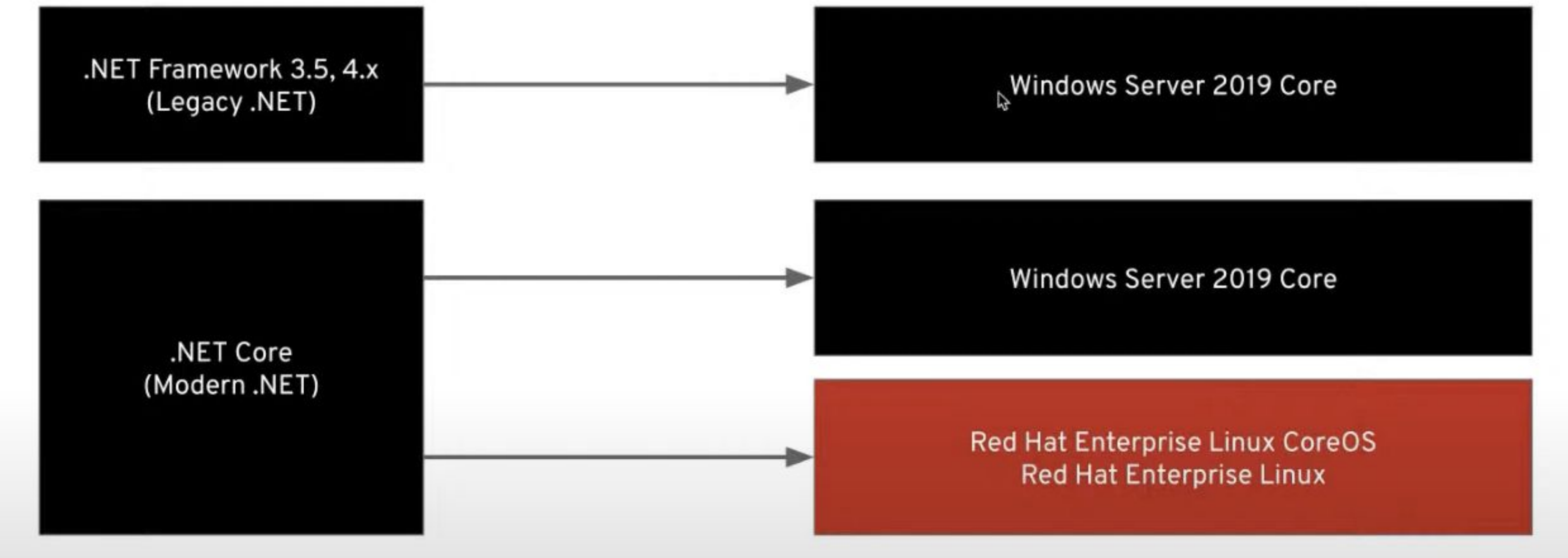
Verify Annotation

Each Windows node is checked for the WMCO version annotation, if the annotated version of a Windows node does not match the WMCO version, and the number of unavailable Windows nodes is less than maxUnhealthy value, the associated Machine is deleted.

Detect Replacement

When a replacement Machine is created by the Machine API Operator, WMCO will reconcile again and configure the VM. This will repeat until all Windows nodes have been configured by the upgraded WMCO.

What OS to target with .NET OpenShift Containers?



.NET Framework 3.5, 4.x
(Legacy .NET)

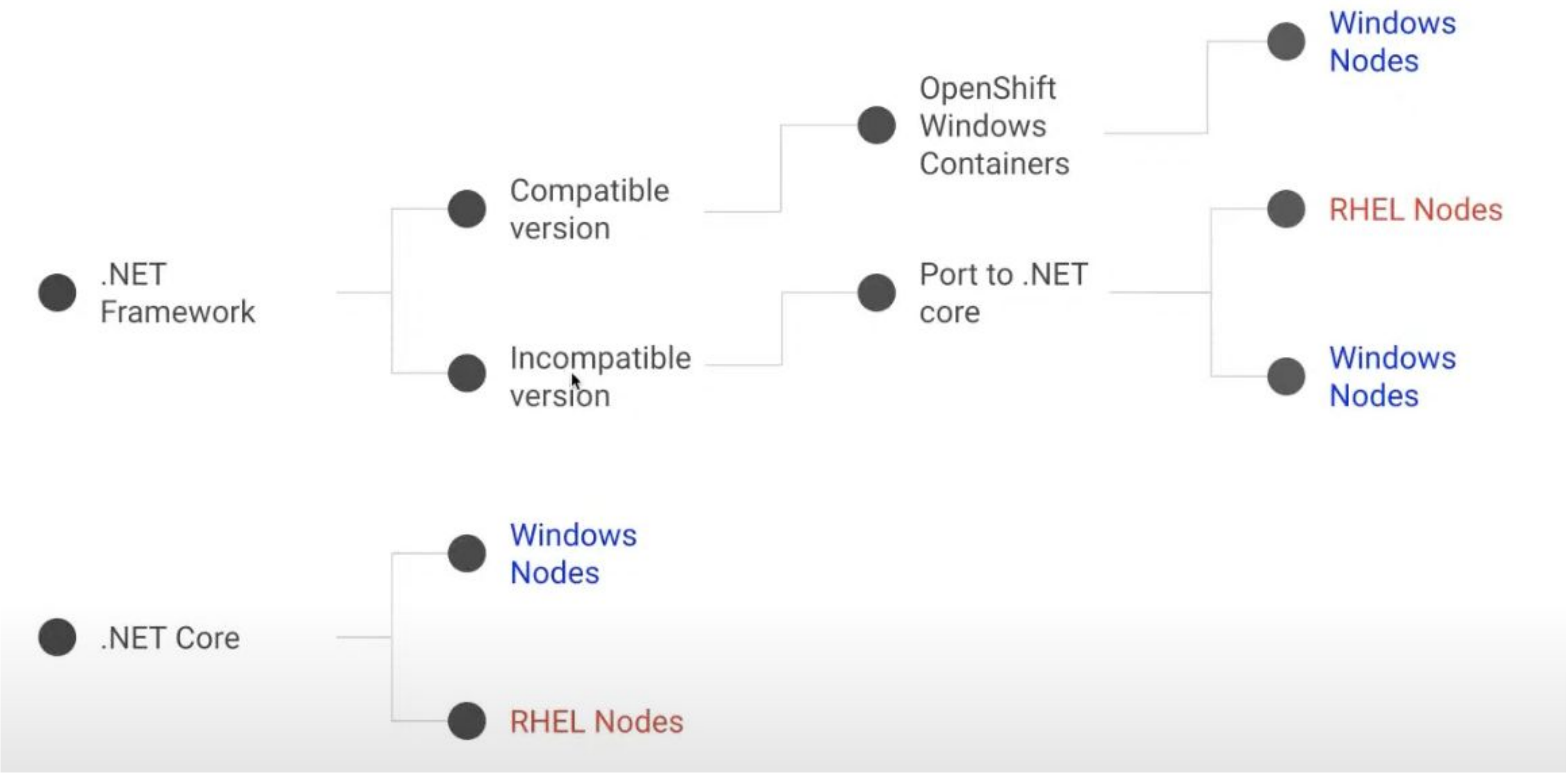
Windows Server 2019 Core

.NET Core
(Modern .NET)

Windows Server 2019 Core

Red Hat Enterprise Linux CoreOS
Red Hat Enterprise Linux

Decision Tree for .NET workloads in OpenShift



.NET Framework

Compatible version

OpenShift Windows Containers

Windows Nodes

Incompatible version

Port to .NET core

RHEL Nodes

Windows Nodes

.NET Core

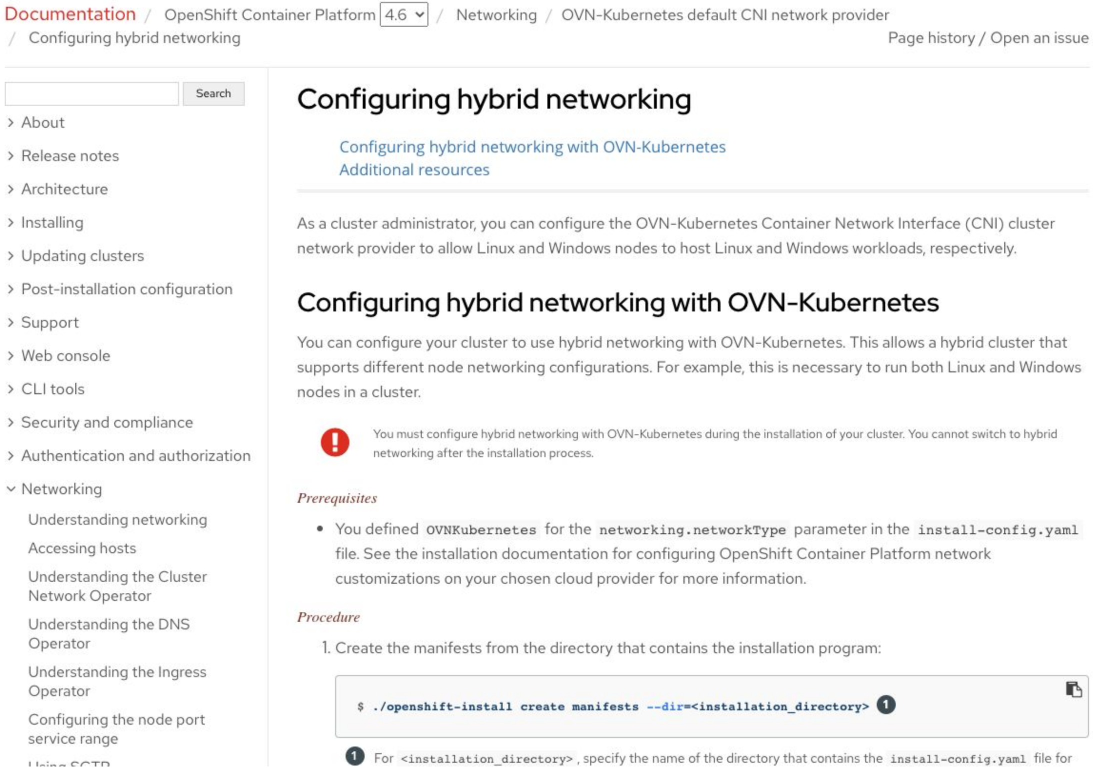
Windows Nodes

RHEL Nodes

Demo #1 OpenShift IPI installation PreRequisites

PreRequisite Setup Customization

OpenShift IPI Installation - Change OpenShift SDN to OVN Hybrid Networking


Documentation / OpenShift Container Platform 4.6 / Networking / OVN-Kubernetes default CNI network provider / Configuring hybrid networking
Page history / Open an issue
Search
› About
› Release notes
› Architecture
› Installing
› Updating clusters
› Post-installation configuration
› Support
› Web console
› CLI tools
› Security and compliance
› Authentication and authorization
⌄ Networking
Understanding networking
Accessing hosts
Understanding the Cluster Network Operator
Understanding the DNS Operator
Understanding the Ingress Operator
Configuring the node port service range
Using SCTP

Configuring hybrid networking

Configuring hybrid networking with OVN-Kubernetes
Additional resources

As a cluster administrator, you can configure the OVN-Kubernetes Container Network Interface (CNI) cluster network provider to allow Linux and Windows nodes to host Linux and Windows workloads, respectively.

Configuring hybrid networking with OVN-Kubernetes

You can configure your cluster to use hybrid networking with OVN-Kubernetes. This allows a hybrid cluster that supports different node networking configurations. For example, this is necessary to run both Linux and Windows nodes in a cluster.

❗ You must configure hybrid networking with OVN-Kubernetes during the installation of your cluster. You cannot switch to hybrid networking after the installation process.

Prerequisites

- You defined `OVNKubernetes` for the `networking.networkType` parameter in the `install-config.yaml` file. See the installation documentation for configuring OpenShift Container Platform network customizations on your chosen cloud provider for more information.

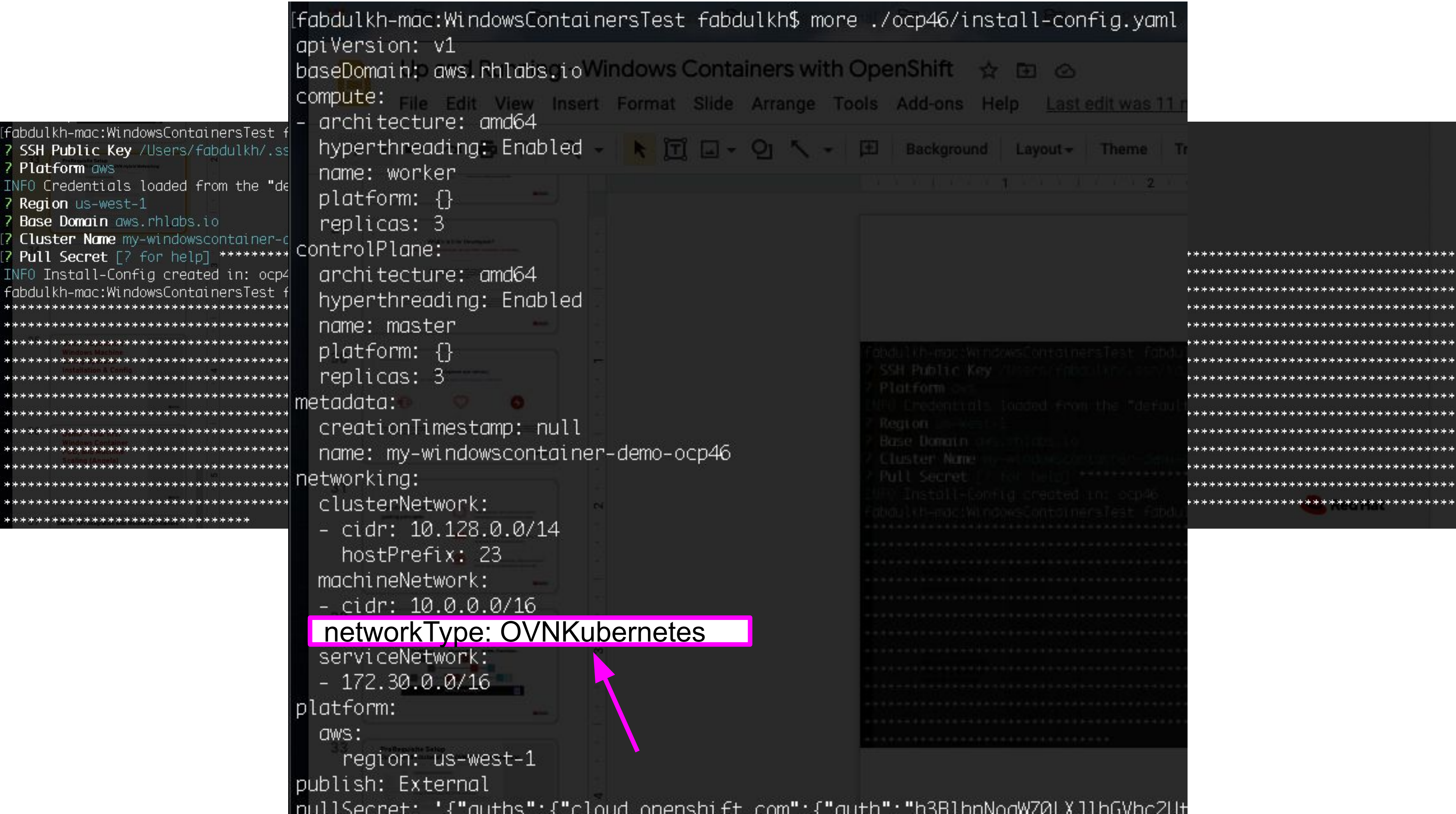
Procedure

1. Create the manifests from the directory that contains the installation program:

```
$ ./openshift-install create manifests --dir=<installation_directory> (1)
```

(1) For `<installation_directory>`, specify the name of the directory that contains the `install-config.yaml` file for

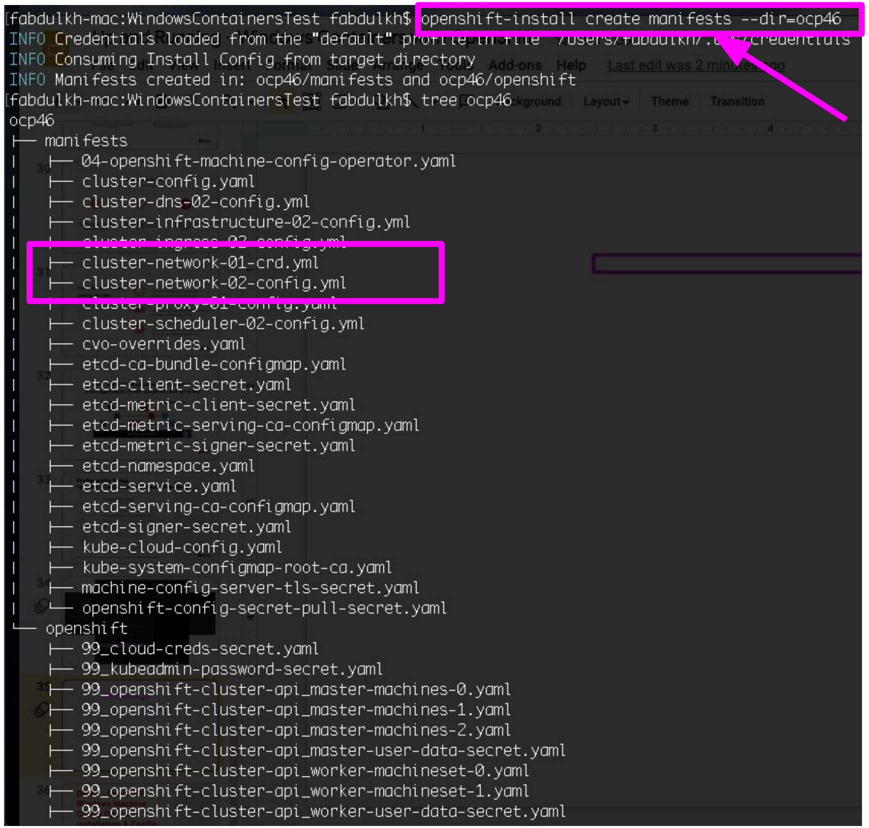
- https://docs.openshift.com/container-platform/4.6/networking/ovn_kubernetes_network_provider/configuring-hybrid-networking.html



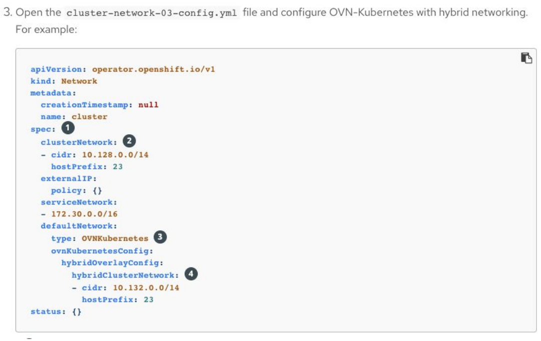
```
[fabdulkh-mac:WindowsContainersTest f
? SSH Public Key /Users/fabdulkh/.ss
? Platform aws
INFO Credentials loaded from the "de
? Region us-west-1
? Base Domain aws.rhlabs.io
[? Cluster Name my-windowscontainer-c
[? Pull Secret [? for help] *********
INFO Install-Config created in: ocp4
fabdulkh-mac:WindowsContainersTest f
```

```
[fabdulkh-mac:WindowsContainersTest fabdulkh$ more ./ocp46/install-config.yaml
apiVersion: v1
baseDomain: aws.rhlabs.io
compute:
- architecture: amd64
  hyperthreading: Enabled
  name: worker
  platform: {}
  replicas: 3
controlPlane:
  architecture: amd64
  hyperthreading: Enabled
  name: master
  platform: {}
  replicas: 3
metadata:
  creationTimestamp: null
  name: my-windowscontainer-demo-ocp46
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OVNKubernetes
  serviceNetwork:
  - 172.30.0.0/16
platform:
  aws:
    region: us-west-1
publish: External
pullSecret: '{"auths":{"cloud.openshift.com":{"auth":"b3BlbnNoaWZ0LXJlbGVhc2Ut
```

Up and Running Windows Containers with OpenShift
File Edit View Insert Format Slide Arrange Tools Add-ons Help Last edit was 11 m
Background
Layout
Theme

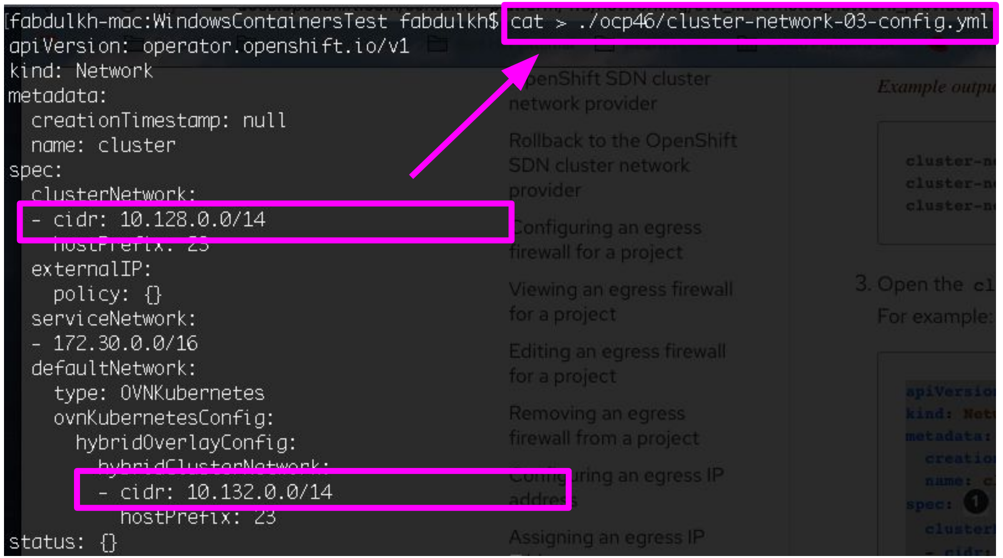


```
fabdulkh-mac:WindowsContainersTest fabdulkh$ openshift-install create manifests --dir=ocp46
INFO Credentials loaded from the "default" profile in file "/Users/fabdulkh/.aws/credentials"
INFO Consuming Install Config from target directory
INFO Manifests created in: ocp46/manifests and ocp46/openshift
fabdulkh-mac:WindowsContainersTest fabdulkh$ tree ocp46
ocp46
├── manifests
│   ├── 04-openshift-machine-config-operator.yaml
│   ├── cluster-config.yaml
│   ├── cluster-dns-02-config.yml
│   ├── cluster-infrastructure-02-config.yml
│   ├── cluster-ingress-02-config.yml
│   ├── cluster-network-01-crd.yml
│   ├── cluster-network-02-config.yml
│   ├── cluster-proxy-01-config.yaml
│   ├── cluster-scheduler-02-config.yml
│   ├── cvo-overrides.yaml
│   ├── etcd-ca-bundle-configmap.yaml
│   ├── etcd-client-secret.yaml
│   ├── etcd-metric-client-secret.yaml
│   ├── etcd-metric-serving-ca-configmap.yaml
│   ├── etcd-metric-signer-secret.yaml
│   ├── etcd-namespace.yaml
│   ├── etcd-service.yaml
│   ├── etcd-serving-ca-configmap.yaml
│   ├── etcd-signer-secret.yaml
│   ├── kube-cloud-config.yaml
│   ├── kube-system-configmap-root-ca.yaml
│   ├── machine-config-server-tls-secret.yaml
│   └── openshift-config-secret-pull-secret.yaml
└── openshift
    ├── 99_cloud-creds-secret.yaml
    ├── 99_kubeadmin-password-secret.yaml
    ├── 99_openshift-cluster-api_master-machines-0.yaml
    ├── 99_openshift-cluster-api_master-machines-1.yaml
    ├── 99_openshift-cluster-api_master-machines-2.yaml
    ├── 99_openshift-cluster-api_master-user-data-secret.yaml
    ├── 99_openshift-cluster-api_worker-machineset-0.yaml
    ├── 99_openshift-cluster-api_worker-machineset-1.yaml
    ├── 99_openshift-cluster-api_worker-user-data-secret.yaml
```



3. Open the `cluster-network-03-config.yml` file and configure OVN-Kubernetes with hybrid networking. For example:

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  creationTimestamp: null
  name: cluster
spec: 1
  clusterNetwork: 2
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  externalIP:
    policy: {}
  serviceNetwork:
  - 172.30.0.0/16
  defaultNetwork:
    type: OVNKubernetes 3
    ovnKubernetesConfig:
      hybridOverlayConfig:
        hybridClusterNetwork: 4
        - cidr: 10.132.0.0/14
          hostPrefix: 23
status: {}
```



```
fabdulkh-mac:WindowsContainersTest fabdulkh$ cat > ./ocp46/cluster-network-03-config.yml
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  creationTimestamp: null
  name: cluster
spec:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  externalIP:
    policy: {}
  serviceNetwork:
  - 172.30.0.0/16
  defaultNetwork:
    type: OVNKubernetes
    ovnKubernetesConfig:
      hybridOverlayConfig:
        hybridClusterNetwork:
        - cidr: 10.132.0.0/14
          hostPrefix: 23
status: {}
```

Demo #2 OpenShift Windows Machine Config Operator Installation & Config (Hanvitha)

Windows Machine Config Operator Workflow

Create a MachineSet

- The way a user will initiate the process is by creating a MachineSet which uses a Windows image with the Docker container runtime installed.
- It usually takes about 15 minutes for the Windows Machine to be configured as a worker node. Ensure the Windows Node is in a Ready state before deploying a workload

Operator configures machines to worker nodes

- The operator is configured to watch for Machines with a **machine.openshift.io/os-id**: Windows label.
- The operator will do all the necessary steps to configure the underlying VM so that it can join the cluster as a worker node.

Install Operator

Navigate to the in-cluster Operator Hub and search for the Windows Operator and Click Install

Demo #3:
Build a Windows Container/Application and Machine Scaling

Openshift MachineSets

MachineSets help define configurations for node deployments (e.g. availability zones, node name, labels, etc.)

To view current machines:

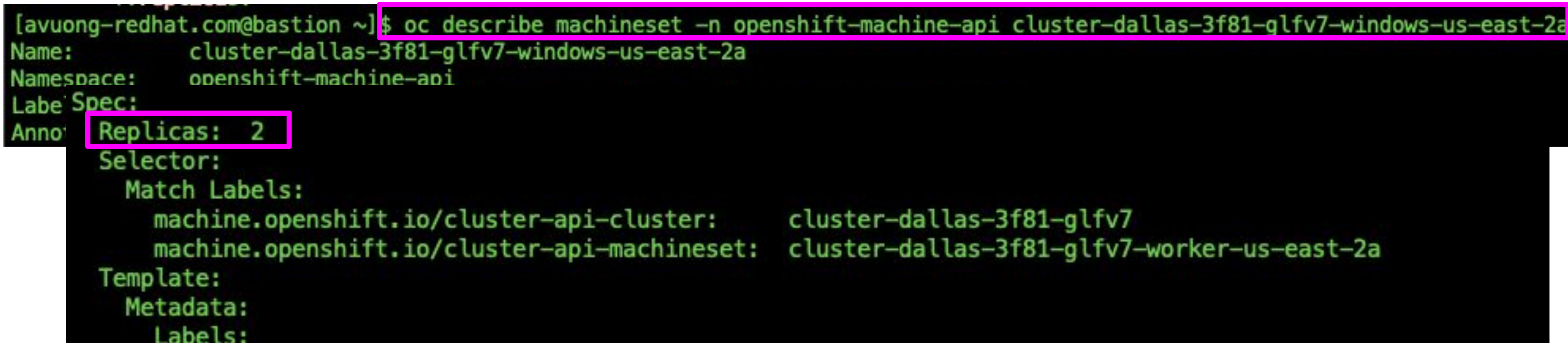
```
$ oc get machines -n openshift-machine-api
```

To view MachineSet configuration of machine:

```
$ oc describe machineset -n openshift-machine-api <machine-name>
```

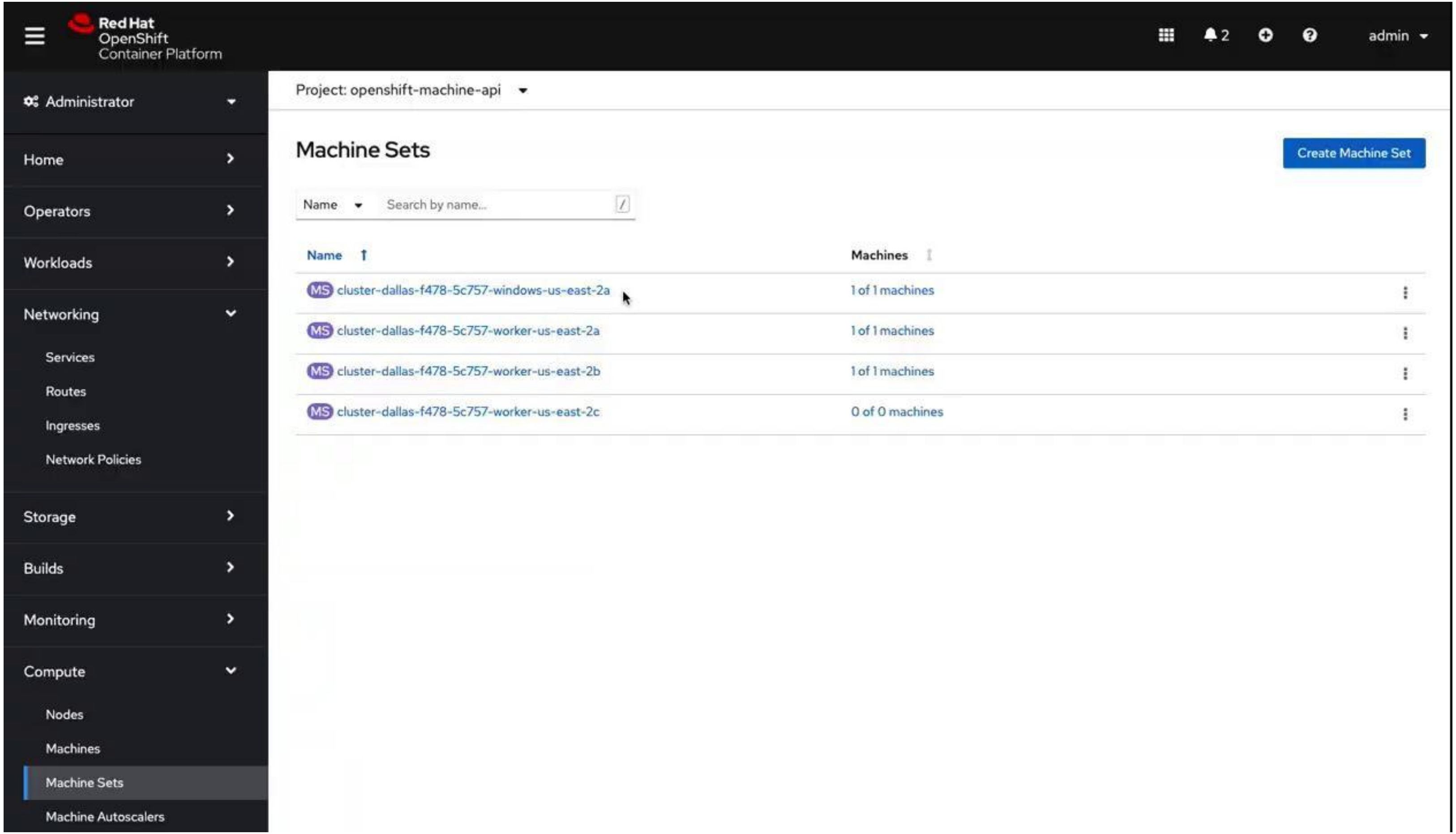
To scale MachineSet:

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



```
[avuong-redhat.com@bastion ~]$ oc describe machineset -n openshift-machine-api cluster-dallas-3f81-glfv7-windows-us-east-2a
Name:          cluster-dallas-3f81-glfv7-windows-us-east-2a
Namespace:     openshift-machine-api
Labe Spec:
Anno   Replicas:  2
       Selector:
         Match Labels:
           machine.openshift.io/cluster-api-cluster:     cluster-dallas-3f81-glfv7
           machine.openshift.io/cluster-api-machineset:  cluster-dallas-3f81-glfv7-worker-us-east-2a
       Template:
         Metadata:
           Labels:
```

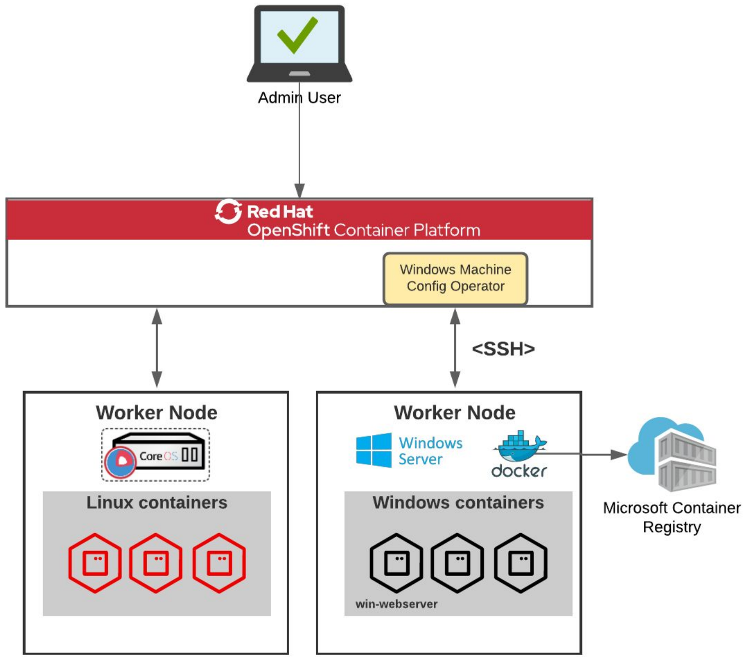
Scaling up Machines in OCP



Project: openshift-machine-api

Machine Sets

Name	Machines
MS cluster-dallas-f478-5c757-windows-us-east-2a	1 of 1 machines
MS cluster-dallas-f478-5c757-worker-us-east-2a	1 of 1 machines
MS cluster-dallas-f478-5c757-worker-us-east-2b	1 of 1 machines
MS cluster-dallas-f478-5c757-worker-us-east-2c	0 of 0 machines



In this demo:

We will deploy a **Windows Web Server application** on the Windows node in Openshift.

Procedure:

1. Pull a Windows Server 2019 base image from mcr.microsoft.com repository using docker
2. Deploy the Windows Web Server application container on Openshift (using ‘oc’ command)
3. Verify a Windows Web Server pod was created for the application
4. Create a route for the Windows Web Server service
5. Verify we can access the Windows Web Server application externally

Win-WebServer configuration files:

https://github.com/angelavuong/ocp4-windows-containers

Building Windows Server app on OCP

Important ‘oc’ Commands

To view OCP clusters:

```
$ oc get nodes -o wide
```

To remote shell into WMCO pod:

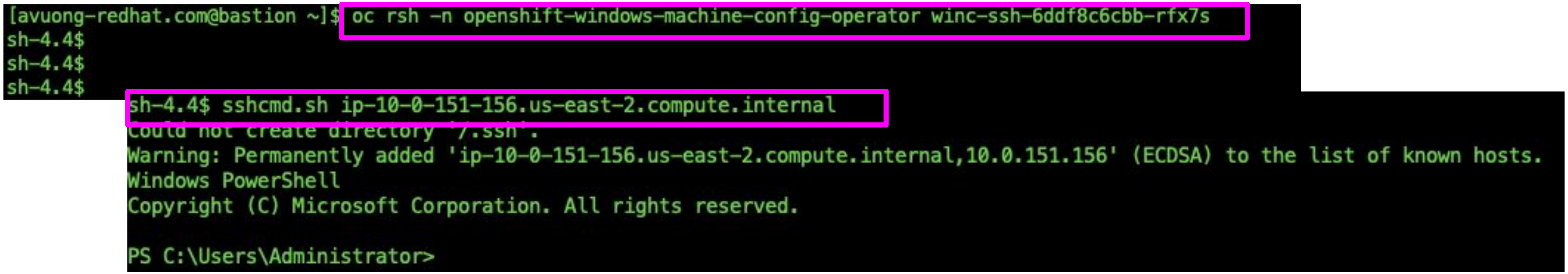
```
$ oc -n openshift-windows-machine-config-operator rsh $(oc get pods -n
openshift-windows-machine-config-operator -l app=winc-ssh -o name)
```

To remote shell into Windows node (from WMCO pod):

```
$ sshcmd.sh <windows-wmco-pod>
```

To access Windows application pod:

```
$ oc exec -it $(oc get pods -l app=win-webserver -o name) powershell
```



```
[avuong-redhat.com@bastion ~]$ oc rsh -n openshift-windows-machine-config-operator winc-ssh-6ddf8c6cbb-rfx7s
sh-4.4$
sh-4.4$
sh-4.4$
```

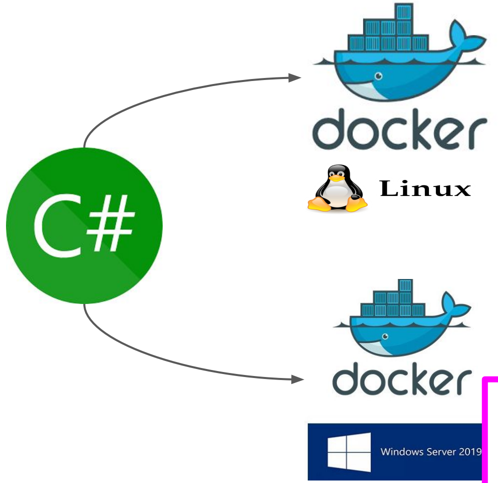
```
sh-4.4$ sshcmd.sh ip-10-0-151-156.us-east-2.compute.internal
Could not create directory '/.ssh'.
Warning: Permanently added 'ip-10-0-151-156.us-east-2.compute.internal,10.0.151.156' (ECDSA) to the list of known hosts.
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\Administrator>
```

Demo #4:

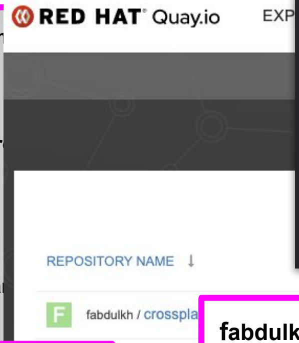
API Integration between Linux and Windows Containers

Demo #4

API Integration between Linux and Windows



```
FROM mcr.microsoft.cor
WORKDIR /app

# copy csproj and restor
COPY *.csproj .
RUN dotnet restore

# copy everything else a
COPY . .
```



fabdulkh / crosspla

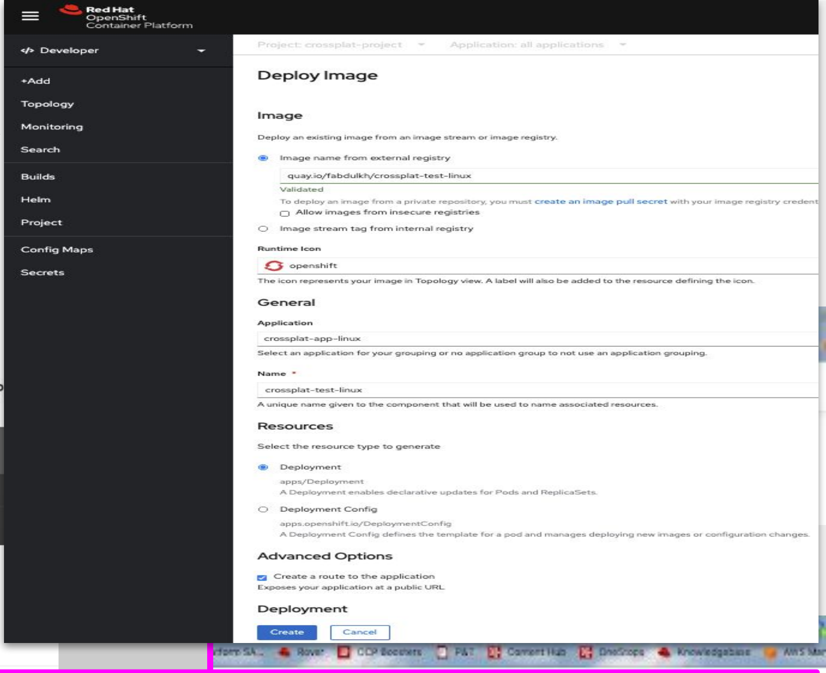
lkh / crosspla

e/aspnet:

docker.dll

```
........
   spec:
     Containers:
........
     nodeSelector:
       kubernetes.io/os: windows
........
     tolerations:
       - key: os
         value: Windows
........
```

```
fabdulkh-mac:cross-plat-docker-test fabdulkh$ oc create -f
crossplat-app-linux.yaml
deployment.apps/crossplat-app-linux created
service/crossplat-app-linux created
route.route.openshift.io/crossplat-app-linux created

fabdulkh-mac:cross-plat-docker-test fabdulkh$ oc create -f
crossplat-app-windows.yaml
deployment.apps/crossplat-app-windows created
service/crossplat-app-windows created
route.route.openshift.io/crossplat-app-windows created
```



Deploy Image

Image

Deploy an existing image from an image stream or image registry.

● Image name from external registry

quay.io/fabdulkh/crossplat-test-linux

Validated

To deploy an image from a private repository, you must create an image pull secret with your image registry credent

□ Allow images from insecure registries

○ Image stream tag from internal registry

Runtime Icon

openshift

The icon represents your image in Topology view. A label will also be added to the resource defining the icon.

General

Application

crossplat-app-linux

Select an application for your grouping or no application group to not use an application grouping.

Name *

crossplat-test-linux

A unique name given to the component that will be used to name associated resources.

Resources

Select the resource type to generate

● Deployment

apps/Deployment

A Deployment enables declarative updates for Pods and ReplicaSets.

○ Deployment Config

apps.openshift.io/DeploymentConfig

A Deployment Config defines the template for a pod and manages deploying new images or configuration changes.

Advanced Options

☑ Create a route to the application

Exposes your application at a public URL

Deployment

https://github.com/faizalak s-den

Thank you!

in linkedin.com/company/red-hat

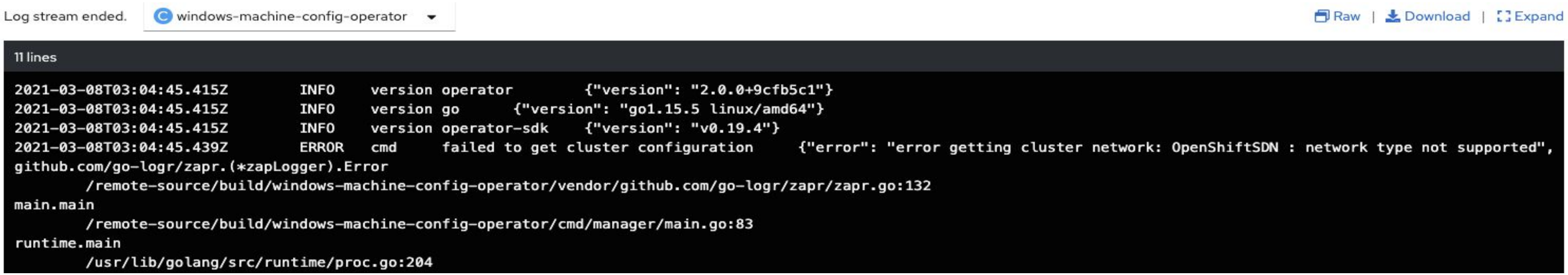
▶ youtube.com/user/RedHatVideos

f facebook.com/redhatinc

twitter.com/RedHat

FAQ

- What happens without a OVN Hybrid set at the OpenShift Cluster?



```
2021-03-08T03:04:45.415Z        INFO    version operator        {"version": "2.0.0+9cfb5c1"}
2021-03-08T03:04:45.415Z        INFO    version go      {"version": "go1.15.5 linux/amd64"}
2021-03-08T03:04:45.415Z        INFO    version operator-sdk    {"version": "v0.19.4"}
2021-03-08T03:04:45.439Z        ERROR   cmd     failed to get cluster configuration     {"error": "error getting cluster network: OpenShiftSDN : network type not supported",
github.com/go-logr/zapr.(*zapLogger).Error
        /remote-source/build/windows-machine-config-operator/vendor/github.com/go-logr/zapr/zapr.go:132
main.main
        /remote-source/build/windows-machine-config-operator/cmd/manager/main.go:83
runtime.main
        /usr/lib/golang/src/runtime/proc.go:204
```

- Where to locate machine set examples
  https://github.com/openshift/windows-machine-config-operator

- **What's the difference between Linux and Windows Server containers?**

  Linux and Windows Server both implement similar technologies within their kernel and core operating systems. The difference comes from the platform and workloads that run within the containers.

  When a customer uses Windows Server containers, they can integrate with existing Windows technologies, such as .NET, ASP.NET, and PowerShell.