Got it — here’s a clean, repeatable way to automate a **Windows VM build on RHEL OpenShift (OpenShift Virtualization/KubeVirt)** from a Linux admin box using YAML + cloudbase-init. This gives you a GitOps-friendly flow you can oc apply -f (or sync via Argo CD).

I’ll show two common approaches for first-boot automation:

* **A. cloudInitNoCloud** (works with Cloudbase-Init; easiest to start)
* **B. Sysprep (Unattend.xml)** (when you already use Windows sysprep)

**Prereqs (one-time)**

* OpenShift Virtualization is installed and the **CDI** operator is running.
* You have a **generalized Windows image** (QCOW2) with VirtIO drivers and Cloudbase-Init installed (or use an image provided by your org).
* Your Linux admin box has oc logged in to the target cluster/project.

Optional but recommended:

* A **golden DV/PVC** for Windows that you clone per-tenant.

**0) Namespace (project)**

apiVersion: v1

kind: Namespace

metadata:

name: win-tenant-a

**1) Import or clone the Windows base disk (DataVolume)**

**Option 1: Import from an HTTP(S) URL**

apiVersion: cdi.kubevirt.io/v1beta1

kind: DataVolume

metadata:

name: win10-golden-dv

namespace: win-tenant-a

spec:

source:

http:

url: https://example.local/images/windows/win10-cloudbase.qcow2

pvc:

accessModes: [ "ReadWriteOnce" ]

resources:

requests:

storage: 60Gi

volumeMode: Filesystem

**Option 2: Clone an existing golden PVC (faster, common in prod)**

apiVersion: cdi.kubevirt.io/v1beta1

kind: DataVolume

metadata:

name: win10-workload-dv

namespace: win-tenant-a

spec:

source:

pvc:

name: win10-golden-pvc

namespace: virt-golden-images

pvc:

accessModes: [ "ReadWriteOnce" ]

resources:

requests:

storage: 60Gi

volumeMode: Filesystem

Apply:

oc apply -f 00-namespace.yaml

oc apply -f 10-dv-import-or-clone.yaml

# Wait until DV is 'Succeeded'

oc -n win-tenant-a get dv -w

**2A) Cloudbase-Init via cloudInitNoCloud (recommended to start)**

Cloudbase-Init understands standard #cloud-config. We’ll set hostname, admin password, enable RDP/WinRM, and install the Windows guest agent if needed.

**2A.1 Secret for admin password (avoid plain text in VM spec)**

apiVersion: v1

kind: Secret

metadata:

name: win-admin-secret

namespace: win-tenant-a

type: Opaque

stringData:

adminPassword: "Str0ng!Passw0rd"

**2A.2 ConfigMap with userData (cloud-config for Windows)**

apiVersion: v1

kind: ConfigMap

metadata:

name: win-cloudinit

namespace: win-tenant-a

data:

userData: |

#cloud-config

hostname: winvm-auto

# Create/ensure local Administrator password (Cloudbase-Init plugin)

chpasswd:

list: |

Administrator:REDACTED\_ADMIN\_PASS

expire: false

# Run first-boot PowerShell (Cloudbase-Init runcmd runs in order)

runcmd:

- powershell: |

# Set password from metadata secret injected below

$pwd = Get-Content -Path "C:\cloudbase-init\adminpass.txt" -Raw

net user Administrator $pwd

# Enable RDP

Set-ItemProperty -Path "HKLM:\System\CurrentControlSet\Control\Terminal Server" -Name "fDenyTSConnections" -Value 0

Enable-NetFirewallRule -DisplayGroup "Remote Desktop"

# Enable WinRM

winrm quickconfig -q

Enable-PSRemoting -Force

# (Optional) Install QEMU guest agent if not in image

# choco install qemu-guest-agent -y

# Reboot to finalize

shutdown /r /t 5

Note: We’ll map the secret value into the VM as a small file so the script can read it, avoiding cleartext in the ConfigMap.

**2A.3 The VM (attaches DV + cloudInit + secret as a disk/volume)**

apiVersion: kubevirt.io/v1

kind: VirtualMachine

metadata:

name: winvm-auto

namespace: win-tenant-a

labels:

app: winvm-auto

spec:

runStrategy: RerunOnFailure

template:

metadata:

labels:

kubevirt.io/domain: winvm-auto

spec:

terminationGracePeriodSeconds: 180

domain:

cpu:

cores: 4

resources:

requests:

memory: 8Gi

devices:

disks:

- name: rootdisk

disk:

bus: virtio

- name: cloudinitdisk

disk:

bus: virtio

- name: adminpass

disk:

bus: virtio

rng: {} # entropy source

features:

acpi: {}

hyperv:

synic: {}

spinlocks:

spinlocks: 8191

vapic: {}

firmware:

bootloader:

efi:

secureBoot: false

networks:

- name: default

pod: {}

volumes:

- name: rootdisk

dataVolume:

name: win10-workload-dv # or win10-golden-dv if you imported

- name: cloudinitdisk

cloudInitNoCloud:

userDataFrom:

configMapRef:

name: win-cloudinit

- name: adminpass

secret:

secretName: win-admin-secret

items:

- key: adminPassword

path: adminpass.txt

Apply:

oc apply -f 20-secret.yaml

oc apply -f 21-configmap-cloudinit.yaml

oc apply -f 30-vm-cloudinit.yaml

**2B) Windows Sysprep (Unattend.xml) route**

If you maintain Windows images with Sysprep, you can supply **Unattend.xml** via a ConfigMap and mount as a **sysprep volume**.

**2B.1 ConfigMap with Unattend.xml**

apiVersion: v1

kind: ConfigMap

metadata:

name: win-sysprep

namespace: win-tenant-a

data:

Unattend.xml: |

<?xml version="1.0" encoding="utf-8"?>

<unattend xmlns="urn:schemas-microsoft-com:unattend">

<settings pass="oobeSystem">

<component name="Microsoft-Windows-Shell-Setup" processorArchitecture="amd64" publicKeyToken="31bf3856ad364e35" language="neutral" versionScope="nonSxS" xmlns:wcm="http://schemas.microsoft.com/WMIConfig/2002/State" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

<AutoLogon>

<Password>

<Value>Str0ng!Passw0rd</Value>

<PlainText>true</PlainText>

</Password>

<Enabled>true</Enabled>

<Username>Administrator</Username>

</AutoLogon>

<RegisteredOwner>OpenShift</RegisteredOwner>

<TimeZone>Eastern Standard Time</TimeZone>

<OOBE>

<HideEULAPage>true</HideEULAPage>

<NetworkLocation>Work</NetworkLocation>

</OOBE>

<FirstLogonCommands>

<SynchronousCommand wcm:action="add">

<Order>1</Order>

<CommandLine>cmd /c reg add "HKLM\System\CurrentControlSet\Control\Terminal Server" /v fDenyTSConnections /t REG\_DWORD /d 0 /f</CommandLine>

<Description>Enable RDP</Description>

</SynchronousCommand>

<SynchronousCommand wcm:action="add">

<Order>2</Order>

<CommandLine>powershell -ExecutionPolicy Bypass -Command "Enable-NetFirewallRule -DisplayGroup 'Remote Desktop' ; winrm quickconfig -q ; Enable-PSRemoting -Force"</CommandLine>

<Description>Enable Firewall & WinRM</Description>

</SynchronousCommand>

</FirstLogonCommands>

</component>

</settings>

<cpi:offlineImage cpi:source="wim://wim\_file/install.wim#Windows 10 Enterprise" xmlns:cpi="urn:schemas-microsoft-com:cpi" />

</unattend>

**2B.2 VM referencing the sysprep volume**

apiVersion: kubevirt.io/v1

kind: VirtualMachine

metadata:

name: winvm-sysprep

namespace: win-tenant-a

spec:

runStrategy: RerunOnFailure

template:

metadata:

labels:

kubevirt.io/domain: winvm-sysprep

spec:

domain:

cpu:

cores: 4

resources:

requests:

memory: 8Gi

devices:

disks:

- name: rootdisk

disk:

bus: virtio

- name: sysprep

cdrom:

bus: sata

networks:

- name: default

pod: {}

volumes:

- name: rootdisk

dataVolume:

name: win10-workload-dv

- name: sysprep

sysprep:

configMap:

name: win-sysprep

Apply:

oc apply -f 22-configmap-sysprep.yaml

oc apply -f 31-vm-sysprep.yaml

**3) (Optional) Expose RDP**

If you want external RDP access quickly (test/lab), expose port 3389 via a NodePort or LoadBalancer Service. (For production, prefer an Ingress/LB with TCP passthrough or a bastion.)

apiVersion: v1

kind: Service

metadata:

name: winvm-rdp

namespace: win-tenant-a

spec:

type: NodePort

selector:

kubevirt.io/domain: winvm-auto # or winvm-sysprep

ports:

- name: rdp

port: 3389

targetPort: 3389

Apply and fetch nodePort:

oc apply -f 40-service-rdp.yaml

oc -n win-tenant-a get svc winvm-rdp -o wide

**4) Automate from Linux (end-to-end)**

Put the YAMLs under Git (e.g., win-tenant-a/) and run:

oc apply -f 00-namespace.yaml

oc apply -f 10-dv-import-or-clone.yaml

oc wait -n win-tenant-a --for=condition=Ready dv/win10-workload-dv --timeout=30m

# Choose A (cloudinit) or B (sysprep)

oc apply -f 20-secret.yaml

oc apply -f 21-configmap-cloudinit.yaml

oc apply -f 30-vm-cloudinit.yaml

# OR

# oc apply -f 22-configmap-sysprep.yaml

# oc apply -f 31-vm-sysprep.yaml

# (Optional) RDP

oc apply -f 40-service-rdp.yaml

For GitOps: commit these files and let **Argo CD** sync them to your cluster(s).

**5) Verify the VM is running (CLI)**

# VM object created?

oc -n win-tenant-a get vm

# VMI (live instance) status + IP (once guest agent reports)

oc -n win-tenant-a get vmi

oc -n win-tenant-a get vmi winvm-auto -o jsonpath='{.status.interfaces[0].ipAddress}{"\n"}'

# Watch phase

oc -n win-tenant-a get vmi -w

# Console (serial)

virtctl -n win-tenant-a console winvm-auto

If the IP doesn’t appear: ensure the **Windows guest agent** is installed, and DHCP is available; otherwise configure a static IP (see tips below).

**Tips & gotchas (Windows on KubeVirt)**

* **Image prep**: Best results with a **generalized/sysprepped** Windows image that already has **VirtIO drivers**, **Cloudbase-Init**, and **qemu-guest-agent** installed.
* **Secure Boot**: If you use unsigned drivers, disable SecureBoot in the VM firmware (as shown). If your org signs drivers, enable it.
* **cloudbase-init vs static IP**: DHCP “just works”. For static IPs, you can:
  + pass a small PowerShell in runcmd to set IP (Cloudbase-Init runs it on first boot), or
  + encode NIC settings in **Unattend.xml** (Sysprep route).
* **Disks**: Use **virtio** bus for performance.
* **RunStrategy**: For one-shot provisioning, RerunOnFailure is fine; for purely manual lifecycle, use Manual.
* **Security**: Keep passwords in **Secrets**. In the example, we avoided embedding it directly in the VM spec.
* **Networking**: For production RDP, prefer exposing via a Layer-4 LoadBalancer or jump-host; lock down with **NetworkPolicy**.

If you want, tell me:

* your Windows version (2016/2019/2022/10/11),
* whether you need **DHCP or static IP**,
* and if you’re using **golden images** already,

…I’ll tailor the cloud-config / PowerShell snippet to your exact network and harden the VM spec (CPU model, features, anti-affinity, labels/annotations) for your cluster.