Ivanti Connect Secure Deployment on OpenShift

Beta

Copyright Notice

This document is provided strictly as a guide. No guarantees can be provided or expected. This document contains the confidential information and/or proprietary property of Ivanti, Inc. and its affiliates (referred to collectively as "Ivanti") and may not be disclosed or copied without prior written consent of Ivanti.

Ivanti retains the right to make changes to this document or related product specifications and descriptions, at any time, without notice. Ivanti makes no warranty for the use of this document and assumes no responsibility for any errors that can appear in the document nor does it make a commitment to update the information contained herein. For the most current product information, please visit www.Ivanti.com.

Protected by patents, see https://www.ivanti.com/patents.

Contents

Contents

Overview

This document describes how to deploy Ivanti Connect Secure as a virtual machine on Red Hat OpenShift by using KubeVirt, Ivanti-provided OVMF files, Secure Boot, vTPM, and configuration initialization through an ISO image.
Use this guide when preparing an OpenShift environment, uploading the ICS image, creating the VM definition, and optionally initializing network and administrator settings during first boot.

Deployment workflow

1. Review the OpenShift, KubeVirt, storage, and networking prerequisites.
2. Prepare the Ivanti OVMF files and enable the KubeVirt sidecar feature.
3. Create the DomainHook that maps the VM firmware paths to the Ivanti OVMF files.
4. Upload the ICS KVM image to the OpenShift cluster.
5. Create the ICS virtual machine and validate compute, storage, networking, Secure Boot, and vTPM settings.
6. Optionally initialize configuration by attaching an ISO image during VM creation.

Pre-Requisites

1. OpenShift server.
2. Kubeconfig file.
3. Kubevirt operator installed.
4. Storage class provisioned.
5. Network provisioned.

Key concepts

Review these terms before starting the deployment steps. They explain the OpenShift virtualization components and security features referenced throughout this guide.

- Kubevirt: KubeVirt is an open-source Kubernetes extension that allows you to run and manage Virtual Machines (VMs) alongside standard containers on a single platform.
- Secure Boot: Secure Boot is a UEFI firmware feature that prevents malicious software, like rootkits or deep-level cheats, from loading when you turn on your PC. It acts like a bouncer by checking the cryptographic signature of every boot file against a trusted database before allowing Linux to start.
- Virtual TPM: A Virtual Trusted Platform Module (vTPM) is a software-based emulation of a physical TPM 2.0 chip. It provides virtual machines (VMs) with hardware-level security,

allowing guest operating systems to securely store cryptographic keys, perform remote system attestation, and meet OS requirements.

- OVMF: OVMF (Open Virtual Machine Firmware) is an open-source project that provides UEFI (Unified Extensible Firmware Interface) support for virtual machines. It serves as the virtual equivalent of a motherboard's BIOS/UEFI, allowing hypervisors like QEMU and KVM to boot guest operating systems using modern firmware standards.

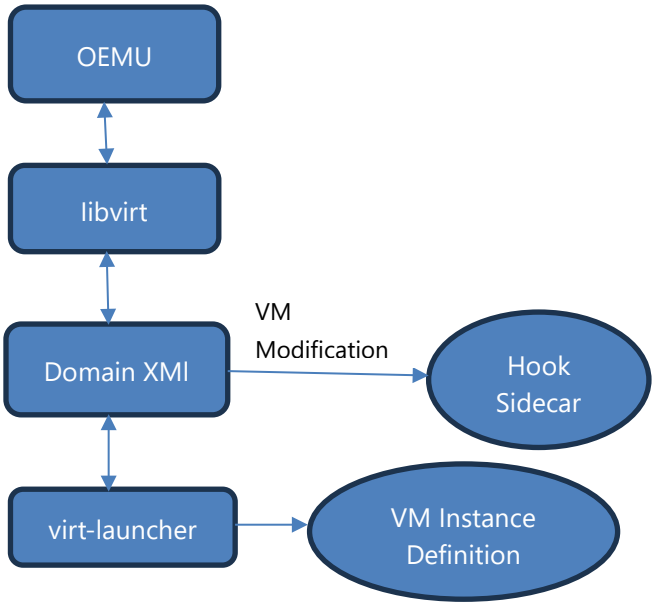
Prepare Ivanti OVMF in the Red Hat OpenShift cluster

Note: Complete this OVMF preparation once per OpenShift cluster. All ICS virtual machines can use the same Ivanti OVMF volumes.

ICS supports secure boot. ICS's bootloaders and Kernel are signed with Ivanti generated private key. Corresponding public key/cert is appended to OVMF files that is released as part of ICS KVM tar artifact.

For deploying ICS virtual machine with released OVMF files, **Kubevirt sidecar feature is used**.

Between generation of libvirt XML and start of virtual machine, OpenShift provides sidecar hook to modify XML file. That approach is used to modify ICS virsh XML to use Ivanti's OVMF.



Reference: https://developers.redhat.com/articles/2025/06/24/enable-confidential-computing-openshift-virtualization#openshift_virtualization_custom_features

Both OVMF_CODE.secboot.ivanti.fd and OVMF_VARS_4M.ivanti.fd are readonly files. The VM doesn't modify them.

Enable Sidecar feature

Enable the KubeVirt sidecar feature before creating the DomainHook. This allows the VM definition to be modified before the virtual machine starts.

```
oc annotate --overwrite -n openshift-cnv hco kubevirt-hyperconverged \
kubevirt.kubevirt.io/jsonpatch='[
  {
    "op": "add",
    "path": "/spec/configuration/developerConfiguration/featureGates/-",
    "value": "Sidecar"
  }
]'
```

Use the following command to enable the KubeVirt sidecar feature.

Upload OVMF files to PVC

Create a PVC for the Ivanti OVMF files, mount it temporarily, copy the released OVMF files into the PVC, and then remove the temporary loader pod.

Following command can be used to create ovmf PVC. PVC will have Ivanti released OVMF files.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: ivanti-ovmf-pvc
  namespace: openshift-cnv
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: hostpath-csi
  resources:
    requests:
      storage: 100Mi
```

Copy ICS KVM OVMF files to ivanti-ovmf-pvc PVC. OVMF files can be found in Ivanti released ICS KVM artifact tar file.

Mount ivanti-ovmf-pvc to busybox container.

```
apiVersion: v1
kind: Pod
metadata:
  name: ovmf-loader
  namespace: openshift-cnv
spec:
  containers:
    - name: ovmf-loader
```

```
      image: busybox
      command: ["/bin/sh", "-c", "sleep 3600"]
      volumeMounts:
        - name: ovmf
          mountPath: /data
  volumes:
    - name: ovmf
      persistentVolumeClaim:
        claimName: ivanti-ovmf-pvc
```

Copy OVMF files to ivanti-ovmf-pvc.

```
oc cp OVMF_CODE.secboot.ivanti.fd -n openshift-cnv ovmf-loader:/data/
oc cp OVMF_VARS_4M.ivanti.fd -n openshift-cnv ovmf-loader:/data/
```

ICS released OVMF files (OVMF_CODE.secboot.ivanti.fd and OVMF_VARS_4M.ivanti.fd) are now in ivanti-ovmf-pvc.

They will be used when creating ICS virtual machine.

Delete busybox POD as it is no longer needed.

Create DomainHook

Create the DomainHook after the Ivanti OVMF files are available in the PVC. The hook updates the VM firmware paths so the VM uses the Ivanti OVMF files during startup.

Domainhook modifies OVMF loader path to use Ivanti OVMF from ivanti-ovmf-pvc PVC.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: ics-ovmf-define-domain-hook
  namespace: openshift-cnv
data:
  onDefineDomain.py: |
    #!/usr/bin/env python3
    import sys
    import xml.etree.ElementTree as ET
    domain_xml = None
    for i, arg in enumerate(sys.argv):
        if arg == "--domain" and i + 1 < len(sys.argv):
            domain_xml = sys.argv[i + 1]
            break
    if not domain_xml:
```

```
        print("missing --domain", file=sys.stderr)
        sys.exit(1)
    root = ET.fromstring(domain_xml)
    os_elem = root.find("os")
    if os_elem is None:
        os_elem = ET.SubElement(root, "os")
    loader = os_elem.find("loader")
    if loader is None:
        loader = ET.SubElement(os_elem, "loader")
    loader.set("readonly", "yes")
    loader.set("secure", "yes")
    loader.set("type", "pflash")
    loader.text = "/custom-ovmf/OVMF_CODE.secboot.ivanti.fd"
    nvram = os_elem.find("nvram")
    if nvram is None:
        nvram = ET.SubElement(os_elem, "nvram")
    nvram.set("template", "/custom-ovmf/OVMF_VARS_4M.ivanti.fd")
    print(ET.tostring(root, encoding="unicode"))
```

Domain hook is run before VM creation. It sets OVMF paths to Ivanti's OVMF files. i.e., /custom-ovmf/OVMF_CODE.secboot.ivanti.fd and /custom-ovmf/OVMF_VARS_4M.ivanti.fd.

When VM is created, domain hook is triggered and it leads to following XML.

```
oc exec virt-launcher-ics-demo-77rrn -n openshift-cnv -- virsh dumpxml
openshift-cnv_ics-demo
  <os>
    <type arch='x86_64' machine='pc-q35-rhel9.6.0'>hvm</type>
    <loader readonly='yes' secure='yes' type='pflash' format='raw'>/custom-
ovmf/OVMF_CODE.secboot.ivanti.fd</loader>
    <nvram template='/custom-ovmf/OVMF_VARS_4M.ivanti.fd'
format='raw'>/var/run/kubevirt-private/libvirt/qemu/nvram/ics-
demo_VARS.fd</nvram>
    <boot dev='hd'/>
    <smbios mode='sysinfo'/>
  </os>
```

OVMF_VARS_4M.ivanti.fd VARs file is used as template to create virtual machine specific VARs file i.e. /var/run/kubevirt-private/libvirt/qemu/nvram/ics-demo_VARS.fd.

ICS VM DomainHook usage

When creating ICS VM, following needs to be added to virtual machine YAML file to use domain hook. Domain hook onDefineDomain.py script is invoked before creating virsh XML defining ICS VM.

ivanti-ovmf-pvc PVC is mounted to /custom-ovmf path so that domain hook python script can change default OVMF path to /custom-ovmf/OVMF_CODE.secboot.ivanti.fd and /custom-ovmf/OVMF_VARS_4M.ivanti.fd.

```
template:
  metadata:
    annotations:
      hooks.kubevirt.io/hookSidecars: >-
        [
          {
            "args": ["--version", "v1alpha2"],
            "configMap": {
              "name": "ics-ovmf-define-domain-hook",
              "key": "onDefineDomain.py",
              "hookPath": "/usr/bin/onDefineDomain"
            },
            "pvc": {
              "name": "ivanti-ovmf-pvc",
              "volumePath": "/ivanti-ovmf-pvc",
              "sharedComputePath": "/custom-ovmf"
            }
          }
        ]
    labels:
      kubevirt.io/domain: ics-demo
```

Upload ICS Image

Upload the ICS KVM image after the OVMF and DomainHook setup is complete. The uploaded image becomes the source PVC used by the virtual machine definition.

Following command can be used to upload ICS image to single node Redhat Openshift cluter.

```
virtctl image-upload pvc ics-25.1.4.0-18366.1 --size=80Gi --image-path=/tmp/ISA-V-KVM-ICS-25.1.4.0-18366.1/ISA-V-KVM-ICS-25.1.4.0-18366.1-VT-kvm.qcow2 --storage-class=hostpath-csi --insecure -n openshift-cnv
```

In above example, hostpatch-csi storage class is being used. Customers need to change it to suit their needs.

Create ICS VM

Create the VM YAML

Start with the following sample YAML file to define the ICS virtual machine.
Update the storage class, network attachments, CPU, memory, and image names to match the target deployment.

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  name: ics-vm3
  namespace: openshift-cnv
spec:
  runStrategy: RerunOnFailure
  dataVolumeTemplates:
    - metadata:
        name: rootdisk3
      spec:
        source:
          pvc:
            name: ics-25.1.4.0-18668.1
            namespace: openshift-cnv
        pvc:
          accessModes:
            - ReadWriteOnce
          storageClassName: hostpath-csi
          resources:
            requests:
              storage: 100Gi
  template:
    metadata:
      annotations:
        hooks.kubevirt.io/hookSidecars: >-
          [
            {
              "args": ["--version", "v1alpha2"],
              "configMap": {
                "name": "ics-ovmf-define-domain-hook",
                "key": "onDefineDomain.py",
                "hookPath": "/usr/bin/onDefineDomain"
              },
              "pvc": {
                "name": "ivanti-ovmf-pvc",
                "volumePath": "/ivanti-ovmf-pvc",
                "sharedComputePath": "/custom-ovmf"
              }
            }
```

```
        ]
    labels:
      kubevirt.io/domain: ics-vm3
  spec:
    domain:
      cpu:
        cores: 2
        model: host-passthrough
      resources:
        requests:
          memory: 4Gi
      features:
        acpi: {}
        smm:
          enabled: true
      firmware:
        bootloader:
          efi:
            secureBoot: true
      devices:
        tpm:
          persistent: true
        disks:
          - name: rootdisk3
            disk:
              bus: virtio
        interfaces:
          - name: extnet
            bridge: {}
        rng: {}
    networks:
      - name: extnet
        multus:
          networkName: openshift-cnv/br-ens8f0-net
    volumes:
      - name: rootdisk3
        dataVolume:
          name: rootdisk3
```

Review VM configuration areas

The following subsections explain the main parts of the VM YAML that typically require review or customization before deployment.

Select the source image

The following section specifies the uploaded ICS image PVC that is used as the VM source image.

```
    source:
        pvc:
          name: ics-25.1.4.0-18668.1
```

```
            namespace: openshift-cnv
```

Mount Ivanti OVMF for the DomainHook

The following annotation mounts the Ivanti OVMF PVC to /custom-ovmf/ so the DomainHook script can point firmware settings to the Ivanti OVMF files.

```
 hooks.kubevirt.io/hookSidecars: >-
          [
            {
              "args": ["--version", "v1alpha2"],
              "configMap": {
                "name": "ics-ovmf-define-domain-hook",
                "key": "onDefineDomain.py",
                "hookPath": "/usr/bin/onDefineDomain"
              },
              "pvc": {
                "name": "ivanti-ovmf-pvc",
                "volumePath": "/ivanti-ovmf-pvc",
                "sharedComputePath": "/custom-ovmf"
              }
            }
          ]
```

Configure compute resources

The following settings define the CPU and memory allocation for the VM.

```
        cpu:
          cores: 2
          model: host-passthrough
        resources:
          requests:
            memory: 4Gi
```

CPUs and memory needs to be based on ICS model like 4500V, 6500V and 8500V.

- ISA4500V: 4 vCPUs and 8 GB RAM.
- ISA6500V: 8 vCPUs and 16 GB RAM.
- ISA8500V: 12 vCPUs and 32 GB RAM.

Enable Secure Boot and vTPM

The following settings enable UEFI Secure Boot and persistent vTPM support for the VM.

```
    firmware:
          bootloader:
            efi:
              secureBoot: true
        devices:
```

```
        tpm:
          persistent: true
```

Configure networking

The following settings create the VM network interface and bind it to the specified Multus network attachment.

```
        interfaces:
          - name: extnet
            bridge: {}
    networks:
      - name: extnet
        multus:
          networkName: openshift-cnv/br-ens8f0-net
```

In above example, only one interface is created. Depending on ICS, three interfaces need to be created with appropriate bridges.

1. Internal.
2. External.
3. Management.

Configuration Initialization

ICS running on OpenShift supports configuration initialization through ISO image.

During bootup if ISO is detected then configuration like IP address, subnet, username and password is initialized from that ISO image.

Example configuration initialization:

```
<?xml version="1.0" encoding="UTF-8"?>
<Environment
     xmlns="http://schemas.dmtf.org/ovf/environment/1"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns:oe="http://schemas.dmtf.org/ovf/environment/1"
     xmlns:ve="http://www.kvmv.com/schema/ovfenv"
     oe:id="">
   <PlatformSection>
      <Kind>ICS KVM Image</Kind>
      <Version>1.0.0</Version>
      <Vendor>Ivanti</Vendor>
      <Locale>en</Locale>
   </PlatformSection>
   <PropertySection>
         <Property oe:key="vaNetworkStack" oe:value=""/>
         <Property oe:key="vaIPAddress" oe:value="10.39.29.11"/>
         <Property oe:key="vaNetmask" oe:value="255.255.254.0"/>
         <Property oe:key="vaGateway" oe:value="10.39.28.1"/>
         <Property oe:key="vaIPv6Address" oe:value=""/>
         <Property oe:key="vaPrefix" oe:value=""/>
         <Property oe:key="vaIPv6Gateway" oe:value=""/>
         <Property oe:key="vaDefaultVlan" oe:value="-1"/>
         <Property oe:key="vaExternalIPAddress" oe:value=""/>
         <Property oe:key="vaExternalNetmask" oe:value=""/>
         <Property oe:key="vaExternalGateway" oe:value=""/>
         <Property oe:key="vaExternalIPv6Address" oe:value=""/>
         <Property oe:key="vaExternalPrefix" oe:value=""/>
         <Property oe:key="vaExternalIPv6Gateway" oe:value=""/>
         <Property oe:key="vaExternalDefaultVlan" oe:value=""/>
         <Property oe:key="vaManagementIPAddress" oe:value=""/>
         <Property oe:key="vaManagementNetmask" oe:value=""/>
         <Property oe:key="vaManagementGateway" oe:value=""/>
         <Property oe:key="vaManagementIPv6Address" oe:value=""/>
         <Property oe:key="vaManagementPrefix" oe:value=""/>
         <Property oe:key="vaManagementIPv6Gateway" oe:value=""/>
         <Property oe:key="vaManagementDefaultVlan" oe:value=""/>
```

```
        <Property oe:key="vaExternalPortReconfigWithValueInVAppProperties"
oe:value="1"/>
        <Property oe:key="vaInternalPortReconfigWithValueInVAppProperties"
oe:value="1"/>
        <Property oe:key="vaManagementPortReconfigWithValueInVAppProperties"
oe:value="1"/>
        <Property oe:key="vaPrimaryDNS" oe:value="10.34.0.220"/>
        <Property oe:key="vaSecondaryDNS" oe:value=""/>
        <Property oe:key="vaWINSServer" oe:value=""/>
        <Property oe:key="vaDNSDomain" oe:value="engroot.com"/>
        <Property oe:key="vaDNSTrafficInterface" oe:value=""/>
        <Property oe:key="vaAdminUsername" oe:value="admin"/>
        <Property oe:key="vaAdminPassword" oe:value="myPassword"/>
        <Property oe:key="vaCommonName" oe:value="ivanti.com"/>
        <Property oe:key="vaOrganization" oe:value="Ivanti Inc."/>
        <Property oe:key="vaRandomText" oe:value="anhbybvevbyhjb"/>
        <Property oe:key="vaAcceptLicenseAgreement" oe:value="y"/>
        <Property oe:key="vaEnableLicenseServer" oe:value="n"/>
        <Property oe:key="vaAdminEnableREST" oe:value="n"/>
        <Property oe:key="vaRegistrationFQDN" oe:value=""/>
        <Property oe:key="vaRegistrationCode" oe:value=""/>
        <Property oe:key="vaRegisterNetworkInterface" oe:value="internal"/>
        <Property oe:key="vaEnableProxy" oe:value=""/>
        <Property oe:key="vaProxyHost" oe:value=""/>
        <Property oe:key="vaProxyPort" oe:value=""/>
        <Property oe:key="vaProxyUsername" oe:value=""/>
        <Property oe:key="vaProxyPassword" oe:value=""/>
        <Property oe:key="vaAuthCodeLicense" oe:value=""/>
        <Property oe:key="vaConfigURL" oe:value=""/>
        <Property oe:key="vaConfigServerCACertPEM" oe:value=""/>
        <Property oe:key="vaConfigData" oe:value=""/>
        <Property oe:key="vaDisableSourceIPRestrictionForDefaultAdminRealm="
oe:value=""/>
   </PropertySection>
</Environment>
Save XML as kvm_template.xml.
```

Note: File name MUST be kvm_template.xml, otherwise configuration initialization fails.

Create ISO image from template file,

```
mkisofs -l -o openshift.iso kvm_template.xml
```

Upload openshift.iso file to OpenShift data volume.

Following yaml file creates datastore volume.

```
apiVersion: cdi.kubevirt.io/v1beta1
kind: DataVolume
metadata:
  name: iso-dv
  namespace: openshift-cnv
spec:
  source:
    upload: {}    # we will upload ISO
  storage:
    accessModes:
      - ReadWriteOnce
    resources:
      requests:
        storage: 100Mi
    storageClassName: hostpath-csi    # or your SC
```

Upload ISO image to iso-dv data volume.

```
virtctl image-upload dv iso-dv -n openshift-cnv  --image-path=openshift.iso   --
uploadproxy-url=https://cdi-uploadproxy-openshift-cnv.apps.ivanti-
cluster.engdevroot.com   --insecure --size=100Mi
```

Verify that upload is successful.

```
$ oc get dv -n openshift-cnv
NAME         PHASE       PROGRESS   RESTARTS   AGE
iso-dv       Succeeded   N/A                   11m
```

Attach iso-dev to VM during creation.

Following is sample yaml file where ISO is attached.

```
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  name: ics-demo-init-cdrom
  namespace: openshift-cnv
spec:
  runStrategy: RerunOnFailure
  dataVolumeTemplates:
    - metadata:
        name: rootdisk5
      spec:
        source:
          pvc:
            name: ics-25.1.4.0-19348.1
            namespace: openshift-cnv
```

```
      pvc:
        accessModes:
          - ReadWriteOnce
        storageClassName: hostpath-csi
        resources:
          requests:
            storage: 100Gi
template:
  metadata:
    annotations:
      hooks.kubevirt.io/hookSidecars: >-
        [
          {
            "args": ["--version", "v1alpha2"],
            "configMap": {
              "name": "ics-ovmf-define-domain-hook",
              "key": "onDefineDomain.py",
              "hookPath": "/usr/bin/onDefineDomain"
            },
            "pvc": {
              "name": "ivanti-ovmf-pvc",
              "volumePath": "/ivanti-ovmf-pvc",
              "sharedComputePath": "/custom-ovmf"
            }
          }
        ]
    labels:
      kubevirt.io/domain: ics-demo-init-cdrom
  spec:
    domain:
      cpu:
        cores: 2
        model: host-passthrough
      resources:
        requests:
          memory: 4Gi
      features:
        acpi: {}
        smm:
          enabled: true
      firmware:
        bootloader:
          efi:
            secureBoot: true
      devices:
        tpm:
          persistent: true
        disks:
```

```
        - name: rootdisk5
          disk:
            bus: virtio
        # Attach ISO here
        - name: cdromiso
          cdrom:
            bus: sata

      interfaces:
        - name: extnet
          bridge: {}
      rng: {}
  networks:
    - name: extnet
      multus:
        networkName: openshift-cnv/br-ens8f0-net
  volumes:
    - name: rootdisk5
      dataVolume:
        name: rootdisk5

    - name: cdromiso
      persistentVolumeClaim:
        claimName: iso-dv
```

If KVM initialization drive is successfully attached then following message will be seen during boot up.

```
This is KVM environment and configuration drive is present
```

Revision History

The following table lists the revision history for this document:

Document Revision	Date	Description
1.0	July 2026	First version for 25.1.3.0 Beta

Documentation

Ivanti documentation is available at https://www.ivanti.com/support/product-documentation.

Technical Support

When you need additional information or assistance, you can contact "Support Center:

- https://forums.ivanti.com/s/contactsupport
- support@ivanti.com

For more technical support resources, browse the support website
https://forums.ivanti.com/s/contactsupport