Cisco 8000 Hardware Emulator Installation Guide

Last updated: 07/06/2026

© 2026 Cisco and/or its affiliates. All rights reserved. This document is Cisco Confidential Information.

Introduction

The Cisco 8000 Series Hardware Emulator is a groundbreaking virtual router platform that emulates the physical hardware design of Cisco's router series. Thus, a focus on accurate modeling of the router chassis, whether fixed form factor or modules chassis, and accurate ASIC models deployed in dataplane and fabric roles. Designed to be Network Operating System (NOS) agnostic, the emulator can run IOS-XR ISO or SONIC NOS on the same virtual hardware, functioning like a hypervisor.

The 8000-simulation platform comes fully self-contained, consisting of:

- Simulation binaries for each supported platform.
- Reference XR ISO and pre-built virtual disks to speed boot process.
- Python based orchestration library to compose and launch topologies via cli and automation.
- Scripts to generate standalone docker instances for integration into third party environments.
- Scripts to automate installation onto Ubuntu based Linux servers.

The Python library is key as most NOS orchestrating systems assume a one-to-one relationship between the router and a virtual machine/container. Our orchestrator can interconnect single VM/process and multi-VM/process routers in the same topology.

To ensure seamless integration with external orchestration systems and hide the multi-VM architecture, Docker wrapped version of each instance can be built enabling integration into Cisco Modeling Labs (CML), ContainerLab, KNE, GNS3, and EVE-NG.

Release Files

Each Cisco 8000 Hardware Emulator software release consists of the following files.

Notation:

rrrr : The emulator release version. (such eft17.2)

xxxx : The ISO XR release. (Such as 26.1.2 represented as 2612)

pppp : One of the 8000 series platform. (such as 8201)

© 2026 Cisco and/or its affiliates. All rights reserved. This document is Cisco Public Information.

Cisco Confidential

File	Content
)00-emulator-rrrr.tar	se software package.
)00-xxxx-f-pppp-image-rrrr.tar	ebuilt IOS-XR virtual disks for fixed chassis (f)
)00-xxxx-d-pppp-image-rrrr.tar	ebuilt IOS-XR virtual disksfor modular/distributed chassis (d)
)00-xxxx-iso-rrrr.tar	oduction IOS-XR ISO installation file
)00-sonic-rrrr.tar	otional SONIC related images
HA256SUM	checksum of the respective tar files
sco_8000_*_installation_guide.pdf	is fileThis file

To validate the tar file, run the Linux command:

```
sha256sum -c SHA256SUM
```

Minimal Required Tar files

Use Case	Tar Files
nimal single platform simulation (2 tar files)	)00-emulator-rrrr.tar d one or more platform image files as needed 8000-xxxx-f-pppp-image-rrrr.tar
nimal modular/distributed chassis simulation tar files)	)00-emulator-rrrr.tar d one or more platform image files as needed 8000-xxxx-d-pppp-image-rrrr.tar d the corresponding ISO tar file)00-xxxx-iso-rrrr.tar
nimal SONIC chassis Simulation	)00-emulator-rrrr.tar)00-soand 8000-sonic-rrrr.tar)

© 2026 Cisco and/or its affiliates. All rights reserved. This document is Cisco Public Information.

Supported Platforms & Boards

Platform Type	Platform
Fixed Chassis	8201 8201-32FH 8202 8202-32FH-M 8212-48FH-M 8101-32H 8102-64H 8111-32EH
Centralized chassis	86-MPA-14H2FH-M
Modular Chassis	8808 8804
Modular Chassis Boards	8800-RP 88-LC036FH-M 88-LC36FH 88-LC48H 88-LC1-36EH 8808-FC 88-LC0-34H14FH
Optical router	NCS1010 NCS1014
Other routers	Any other Cisco virtual router or third party VM can be integrated into the tolology.

© 2026 Cisco and/or its affiliates. All rights reserved. This document is Cisco Public Information.

Included Images

Platform Type	Platform
IOS-XR	Recent Cisco FCS images such as 26.x
SONIC	SONIC 2305 BUILD 8748 (sdk: 1.71.10.2)

System Requirements

Optimal performance is achieved when the emulator is run directly on x86 hardware without any intermediate virtualization layers. Specific recommendations are provided for different system types, including bare metal servers, cloud instances, and virtual private clouds.

If the emulator is running on any environment other than baremetal, then nested virtualization must be enabled on the underlying hypervisor. Please refer to documentation per platform on how to enable this feature.

Currently the emulator binaries are compiled for Ubuntu 22. To support a wide range of router topologies, use of high core and memory instances based on newer generation of X86 CPUs is recommended. Servers should ideally use NVMe/SSD drives for the highest IO bandwidth.

Below is a list of compute options ranked from best to acceptable.

System	Type	Minimal System	Operating System	Note
Dedicated Server	Bare Metal Server	16+ cores 64G+ Mem	Ubuntu22	Optional: CentOS8(docker (Ubuntu22))
AWS	Bare Metal Instance	Bare metal Instance M5d.metal	Ubuntu22	Optional: CentOS8(docker (Ubuntu22))
Azure	Virtual Machine	16+ cores 64G+ Mem	Ubuntu22	Requires nested Virtualization.
Google Cloud	Virtual Machine	16+ cores 64G+ Mem	Ubuntu22	Requires nested Virtualization.
ESXI	Virtual	16+ cores	Ubuntu22	Requires nested Virtualization.

© 2026 Cisco and/or its affiliates. All rights reserved. This document is Cisco Public Information.

Private Cloud	64G+ Mem

Runtime Requirements

The smallest instantiation of IOS-XR will be on fixed platforms with 4 VCPU's and 20 Gbytes of memory. The minimal recommended setting for running SONIC is 4 VCPU's and 12G of memory. Depending on configuration scale applied to the router, the VCPU/memory settings need to be raised. The default hardware memory footprint of Cisco routers is 32G.

Sample Configurations

Emulator	Operating System	CPU	Memory	Min Memory	Disk	Comment
8201	IOS-XR	4	20-32G	12G	30G/board	
8802	IOS-XR	4	20-32G	12G	30G/board	
8804/8808	IOS-XR	8* (RP+LC)	64G* (2x32)	40G* (2x20)	30G/board	
8201	SONIC	4	12-20G	12	30G+	

* Modular chassis such as the 8808 consist of one or two route processors, and a range of linecards. Each will consume 2-4 cores, and 12-32G.

Minimal Installation Options

For a minimal IOSXR installation, the 8000-emulator and one 8000-*-f-images.*.tar should be downloaded. For modular chassis emulation, the 8804 and or the 8808 packages and the relevant ISO file should be downloaded.

For a minimal SONIC installation, the 8000-emulator and the 8000-sonic tar files should be downloaded.

Multiple IOSXR releases are supported, and users should download additional tar files corresponding to each release.

Install Emulator on Linux Server

This section shows you how to install the Cisco 8000 emulator on a Linux Server:

1. Verify HW assist virtualization is enabled in system BIOS.
2. Install Ubuntu 22 onto server. Verify /dev/kvm present.

© 2026 Cisco and/or its affiliates. All rights reserved. This document is Cisco Public Information.

```
ls /dev/kvm
```

3. Download all the 8000*.tar files from the **General EFTx.y Release** folder.
4. Extract the contents of the tar files using the following command:

```
find . -name '*.tar' -exec tar -xvf {} \;
```

5. Run the set-up scripts as shown below:

```
cd 8000-xxxx
sudo scripts/ubuntuServerManualSetup.sh
sudo reboot
```

Install Emulator on Virtualized Environments

This section shows you how to install the Cisco 8000 emulator on a Virtualized Environment:

1. On cloud platforms, start with their marketplace reference Ubuntu 22 image. On private clouds, install Ubuntu 22 onto the hypervisor.
2. Verify nested virtualization is configured correctly by checking presence of /dev/kvm

```
ls /dev/kvm
```

3. Download all the 8000*.tar files from the **General EFTx.y Release** Emulator software download page.
4. Extract the contents of the tar files using the following command:

```
find . -name '*.tar' -exec tar -xvf {} \;
```

5. Run the set-up scripts as shown below:

```
cd 8000-xxxx
sudo scripts/ubuntuServerManualSetup.sh
sudo reboot
```

All the tools and binaries will be installed to /opt/cisco. Follow the **Cisco 8000 user guide** for running simulations.

Install Emulator on Docker Containers

If you prefer to use docker to run the emulator, this section shows you how to build and run a docker image with Ubuntu 22 OS.

Requirements:

- A bare metal server meeting the requirements specified in the **System Requirements** section.
- Docker 18+ must be installed and /dev/kvm should be available.
- The underlying operating system can be Redhat/CentOS8+, Ubuntu22+, or Fedora.

© 2026 Cisco and/or its affiliates. All rights reserved. This document is Cisco Public Information.

The following steps shows you how to build the docker image and run it:

1. Download all the 8000 tar files from the **General EFTx.y Release** Emulator software download page.
2. Extract the contents of the downloaded tar files using the following command:

   ```
   find . -name '*.tar' -exec tar -xvf {} \;
   ```

3. Change directory to the newly extracted **8000-x.y** directory and run the script to build the image for the docker container. This command takes at least 12 minutes to complete execution. Assuming **x.y** is the current emulator release; you can use the below command:

   cd 8000-x.y; ./scripts/build_docker_image.sh 8000:x.y ./docker/Dockerfile.generic

   Note: For the Cisco 8000 Emulator Notebooks, use the following command for this step:

   cd 8000-x.y; ./scripts/build_linux_docker_images.sh -v x.y -p <proxy>

4. Run the docker image using the command:

   docker run --cap-add=NET_ADMIN -p 8889:8889 --device /dev/kvm:/dev/kvm --rm -it 8000:x.y

5. Run this simple test on the docker container:

   ```
   # copy sample single router yaml file to /nobackup
   cd /nobackup
   cp /opt/cisco/pyvxr/examples/xr/26.2.1/8201/8201-2621.yaml .
   # launch
   vxr.py start 8201-732.yaml
   # Wait for script to return to command line with "INFO Sim up" as last status line.
   # Acquire route console connection information
   vxr.py ports
   # telnet to "HostAgent" ip and "Serial0" port
   telnet <HostAgent-ip> <Serial0-port>
   # to end simulation type, type the below at the docker prompt
   vxr.py clean
   ```

The simulation life cycle is managed by the **pyvxr** python library. Instructions for unpacking the pyvxr html documentation is available at **8000-x.y/docs/README.python_lib**.

Install Emulator on AWS

Requirements:

© 2026 Cisco and/or its affiliates. All rights reserved. This document is Cisco Public Information.

- AWS cli installed and configured.
- AWS user access/secret keys

The following steps will guide you to launch the AWS instance with the emulator:

1. Download and extract the tar files of the software package.
2. Change directory to the newly extracted 8000-x.y. Assuming x.y is the current emulator release; you can use the commands below:

cd 8000-x.y/

3. Create the AWS AMI using the command: This command takes at least 1 hour to complete execution.

./scripts/aws/awsCreateAMI.sh -r region -t tar_files_path -v eft_version

Note:

- Choose an AWS region that is closest to you.
- eft_version: example “17.2”.

4. Launch an AWS instance with the newly created AWS AMI using the “awsLaunchInstance.sh” under scripts folder:

./scripts/aws/awsLaunchInstance.sh -r region

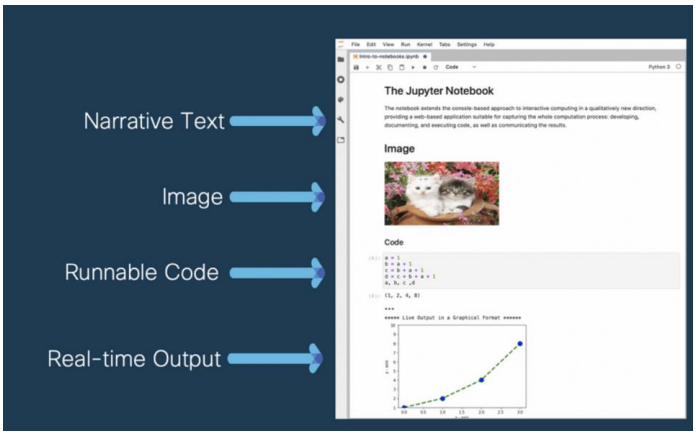
Note: region must be the same as the previous step.

5. Follow the directions displayed to access the launched AWS instance.
6. Follow step 5 on the previous section to run a simple test on the Emulator.

Cisco 8000 Emulator Notebooks

These notebooks combine narrative text, images, videos, interactive visualizations, runnable code, and real-time outputs.

© 2026 Cisco and/or its affiliates. All rights reserved. This document is Cisco Public Information.



The notebook communicates with the Cisco 8000 emulator running in the background and at the click of the **play** button in the notebook, brings up multi-router topologies within minutes. This enables users to execute configurations/commands on the emulated routers directly from the notebook.

The Cisco 8000 Emulator software package includes Cisco 8000 Emulator Notebooks as well.

The instructions in the following sections show how to get the notebook set up ready.

There are two options to install and access notebooks:

- Option 1: Notebooks on Docker Containers

  Or

- Option 2: Notebooks on AWS

Option 1: Notebooks on Docker Containers

The following steps show how to install notebooks that interact with the emulator in a docker container:

1. Execute steps 1-4 in the section "Install Emulator on Docker Containers"
2. Set the appropriate proxy settings if behind a firewall.
3. Run the script installJupyterNotebooks.sh to install and start jupyter notebooks service:

   ```
   . /opt/cisco/notebooks/installJupyterNotebooks.sh
   ```

4. When prompted, set up a password for the notebooks.
5. From your external computer terminal, set up ssh tunnels to port 8889 of the server that hosts the docker container, by using the command:

   ```
   ssh -L 8889:localhost:8889 username@server
   ```

© 2026 Cisco and/or its affiliates. All rights reserved. This document is Cisco Public Information.

Cisco Confidential

6. Open the browser on your computer and access jupyter using the URL **localhost:8889**. Use the notebooks password that you have set up in step 4 of this section.
7. Once Jupyter lab has opened in your browser, double-click the README file README.ipynb from the file explorer on the left pane. This document explains how to use notebooks and provides a list of available notebooks.
8. To exit Jupyter lab on the docker container, use ctrl-c twice. After that, if you want to access the notebooks again, enter the following command on the docker container and then follow the steps 5-6 above:

```
jupyter lab --no-browser --port=8889 --ip=0.0.0.0 --allow-root --notebook-dir=~/notebooks
```

For more information on using notebooks, refer to the following files in the docker container:

~/notebooks/README_notebooks.pdf

~/notebooks/README_notebooks.txt

Option 2: Notebooks on AWS

The following steps show how to install notebooks on your AWS Instance:

1. Follow the steps 1-5 listed in the section "Install Emulator on AWS" to launch and access the AWS instance.
2. Run the Notebook installation script in the SSH terminal as shown below. This script installs Jupyter notebooks and starts the notebooks service. When prompted, set up a password for the notebooks.

```
. /opt/cisco/notebooks/installJupyterNotebooks.sh
```

3. Open browser on your computer and access notebooks by entering `<public-ipv4-address-of-AWS-instance>:8889 in` the address-bar. Use the notebook password that you have set up in step 1 of this section.
4. Once the Jupyter application has opened in your browser, double-click on the file README.ipynb from the file explorer on the left pane. This notebook has links to various other notebooks which can help you get started. It also provides a list of available notebooks.

CML Installation

Prerequisites

- It is recommended to run CML on a bare metal server
- CML 2.9 or later installed and licensed (Docker is not supported in earlier versions)
- SSH access to the CML server
- OVXR .tar file for CML

© 2026 Cisco and/or its affiliates. All rights reserved. This document is Cisco Public Information.

Cisco Confidential

1. Copy the OVXR package to the CML server

```
# Copy the OVXR package into /var/tmp on the CML server using scp:
scp cml_8201-32fh_25.1.1.tar sysadmin@cml-server:/var/tmp/
```

2. Access the CML Server

```
# SSH into the CML server:
ssh sysadmin@cml-server
```

3. Extract the OVXR tar file

```
# Change to /var/tmp and extract the OVXR package:
cd /var/tmp
sudo tar -xvf cml_8201-32fh_25.1.1.tar
```

The extracted package contains the following structure:

```
├── node-definitions
│   └── 8201-32fh.yaml
└── virl-base-images
    └── 8201-32fh-25.1.1
        ├── 8201-32fh-25.1.1.yaml
        └── 8201-32fh_25.1.1.tar
```

4. Copy files to the CML configuration directories

```
sudo cp node-definitions/8201-32fh.yaml /var/lib/libvirt/images/node-definitions/
sudo cp -r virl-base-images/8201-32fh-25.1.1/ /var/lib/libvirt/images/virl-base-images/
# Change files ownership:
sudo chown libvirt-qemu:virl2 /var/lib/libvirt/images/node-definitions/8201-32fh.yaml
sudo chown -R libvirt-qemu:virl2 /var/lib/libvirt/images/virl-base-images/8201-32fh-25.1.1
```

5. Restart CML services

```
sudo systemctl restart virl2.target
```

6. Start the OVXR node

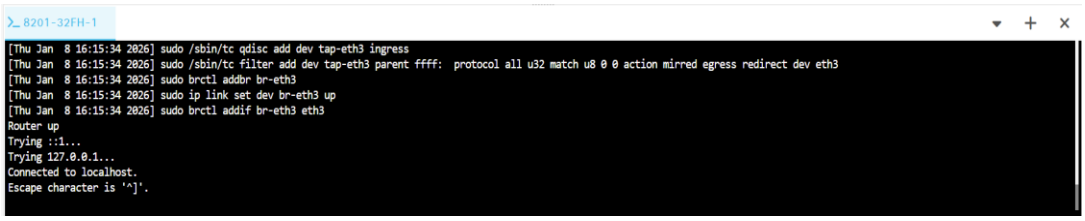
In the CML web UI, add the **OVXR** node to your lab and start it.

7. Configure the node interfaces

Open the OVXR node console in the CML web UI. You will see the node boot process in the console. Once booted, press Enter to connect to the IOS XR console.

© 2026 Cisco and/or its affiliates. All rights reserved. This document is Cisco Public Information.

Cisco Confidential



Install OVXR in EVE-NG

Prerequisites

- EVE-NG Professional Edition installed and licensed
- SSH access to the EVE-NG server
- OVXR Docker image available as a .tar file

1. Access the EVE-NG Server

   ```
   # SSH into the EVE-NG server:
   ssh root@<eve-ng-ip>
   ```

2. Load the OVXR Docker Image

   ⚠ Do NOT extract the OVXR tar file.

   ```
   docker load -i 8202-32fh_25.1.1.tar
   #The OVXR Docker image requires privileged permission. To achieve this, the image needs to be renamed:
   docker tag 8201-32fh:25.1.1 8201-32fh-privileged:25.1.1
   ```
   OR
   ```
   docker tag 8201-32fh:25.1.1 8201-32fh:privileged
   ```

3. Verify the Image

   ```
   # Confirm that the image is available on the EVE-NG host:
   docker images
   ```

4. Create the Docker Node in EVE-NG

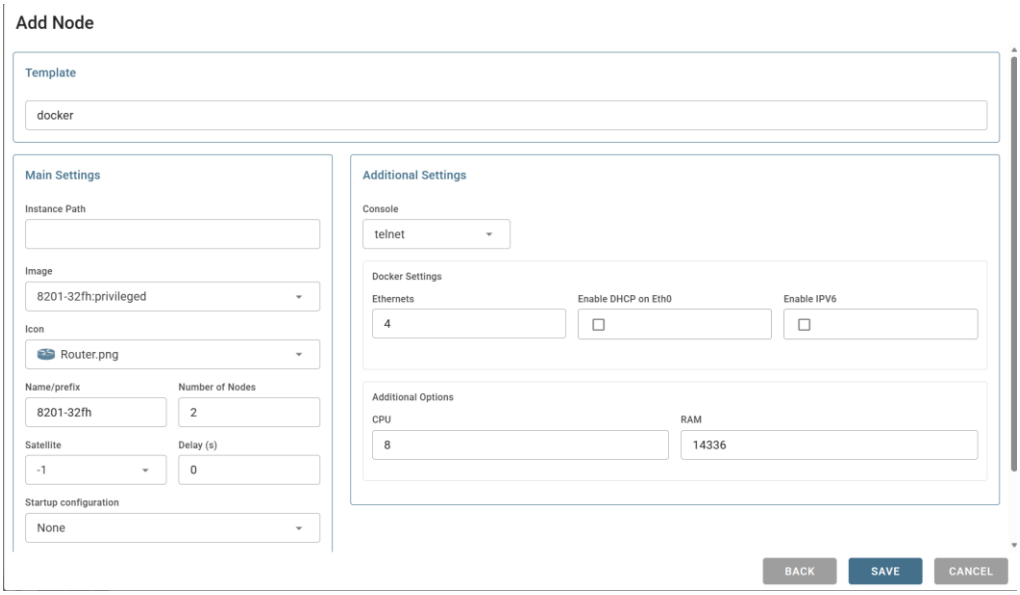
   In the EVE-NG web UI:

   1. Open your lab
   2. Click **Add Node**
   3. Select **Docker** as the node type
   4. Choose the OVXR Docker image from the list
   5. Configure the node resources:
      - **CPU:** 8 CPUs

© 2026 Cisco and/or its affiliates. All rights reserved. This document is Cisco Public Information.

Cisco Confidential

- **RAM:** 14336 MB
- **Ethernet interfaces:** As required by the topology



5. Deploy and Start

   Save the node and start it. Open the OVXR node console directly from the EVE-NG web UI. From within the container, connect to the IOS XR console using the following command to begin configuring the node interfaces:

   ```
   connect
   ```

   Note: No EVE-NG restart is required after loading Docker images

Install OVXR in GNS3

Prerequisites

- GNS3 installed (local server or GNS3 VM)
- Docker Engine installed and running on the GNS3 server
- OVXR Docker image available as a .tar file

1. Load the OVXR Docker Image

   ⚠ Do NOT extract the OVXR tar file.

   ```
   # Load the Docker image on the system running the GNS3 server:
   docker load -i 8202-32fh_25.1.1.tar
   ```

© 2026 Cisco and/or its affiliates. All rights reserved. This document is Cisco Public Information.

Cisco Confidential

2. Verify the Image

```
# Verify that the image is available:
docker images
```

3. Create the OVXR Template

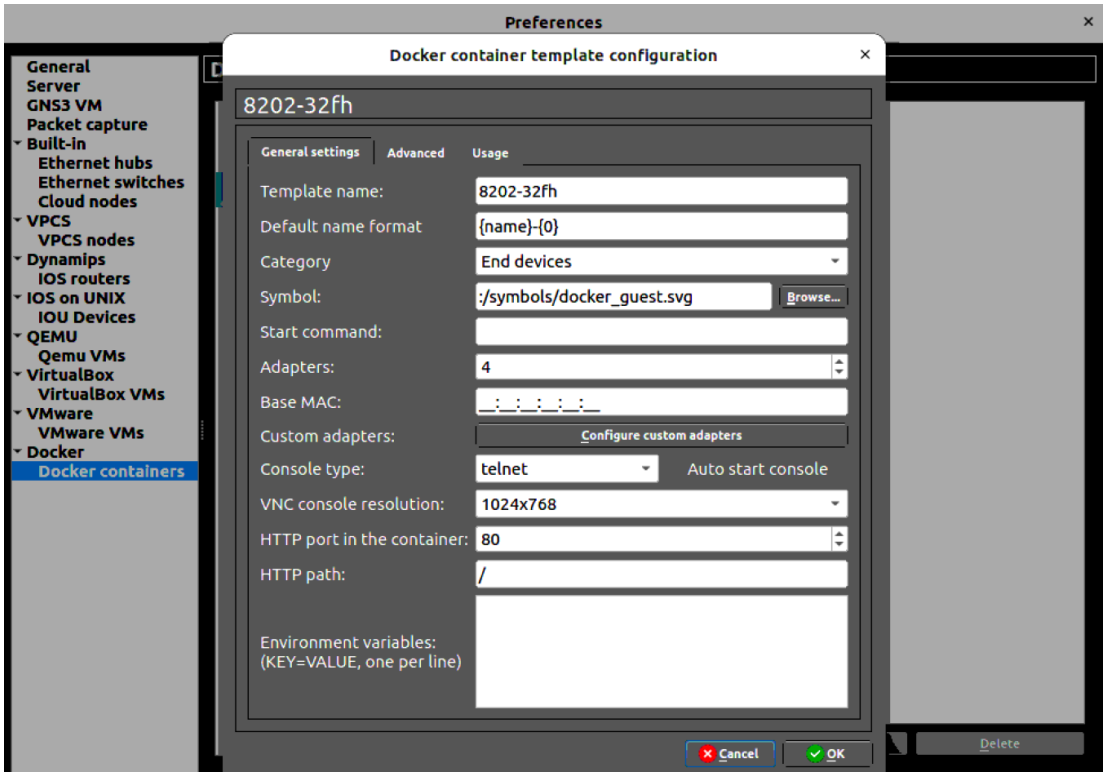
   In the GNS3 desktop application:

   1. Open **Edit** > **Preferences**.
   2. Select **Docker Containers**.
   3. Click **New**.
   4. Select the OVXR Docker image from the list.
   5. Configure the OVXR template:
      - **Template name**
      - **Console type:** Telnet (default)
      - **Adapters:** As required by the topology

4. Click Finish to create the template.

© 2026 Cisco and/or its affiliates. All rights reserved. This document is Cisco Public Information.

Cisco Confidential



5. Add the OVXR Node

   Open your GNS3 project and deploy the OVXR container:

   1. Open your project.
   2. Select Browse End Devices from the left toolbar.
   3. Drag the OVXR Docker template onto the workspace.
   4. Connect the required Ethernet interfaces to your topology.

6. Deploy and Start

   Right-click the OVXR node and select **Start**. After the node is running, right-click it again and select Console. From within the container, connect to the IOS XR **console**.

   Notes

   - The Docker daemon must be running before launching GNS3.
   - If using the GNS3 VM, the OVXR Docker image must be loaded into the GNS3 VM, not the local host.
   - No GNS3 restart is required after loading the Docker image. If the image does not appear, refresh the Docker templates or reopen the Docker Containers preferences.

© 2026 Cisco and/or its affiliates. All rights reserved. This document is Cisco Public Information.

Cisco Confidential

Install OVXR in CLAB

Prerequisites

- Linux host with Docker and Containerlab installed
- SSH access to the server
- OVXR Docker image available as a .tar file

1. Load the Docker Image

   ⚠ Do NOT extract the OVXR tar file.

   ```
   # Load the OVXR Docker image locally using the docker load command:
   docker load -i 8202-32fh-m_25.1.1.tar
   ```

2. Deploy the Lab

   ```
   # Use the following Containerlab topology file to deploy the test lab:
       name: testLab
       topology:
       nodes:
       Cisco8202-1:
       kind: cisco_c8000
       image: 8202-32fh-m:25.1.1
       image-pull-policy: Never
       Cisco8202-2:
       kind: cisco_c8000
       image: 8202-32fh-m:25.1.1
       image-pull-policy: Never
       links:
       - endpoints: ["Cisco8202-1:FH0_0_0_0", "Cisco8202-2:FH0_0_0_0"]
   # Deploy the lab using the following command:
   containerlab deploy -t test_CLAB.yaml
   ```

3. Configure Node Interfaces

   Log in to the running node container using its container ID, then configure the node interface:

   ```
       docker exec -it 42284c2b5121 /bin/bash
       root@Cisco8202-1:/# telnet localhost 60000
       RP/0/RP0/CPU0:Cisco8202-1# config
       RP/0/RP0/CPU0:Cisco8202-1(config)# interface FourHundredGigE0/0/0/0
       RP/0/RP0/CPU0:Cisco8202-1(config-if)# ip address 10.1.1.1 255.255.255.0
       RP/0/RP0/CPU0:Cisco8202-1(config-if)# no shut
       RP/0/RP0/CPU0:Cisco8202-1(config-if)# commit
       RP/0/RP0/CPU0:Cisco8202-1(config-if)# end
   ```

4. Destroy the Lab

   When you are finished, destroy the lab and remove the Containerlab nodes using the following command:

   ```
       containerlab destroy -t test_CLAB.yaml
   ```

© 2026 Cisco and/or its affiliates. All rights reserved. This document is Cisco Public Information.

Cisco Confidential

© 2026 Cisco and/or its affiliates. All rights reserved. This document is Cisco Public Information.

Cisco Confidential