# Cisco 8000 Hardware Emulator Installation Guide

Last updated: 05/30/2024

## 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 consists of two key components: One for one emulator for each 8000 router in the portfolio, and a Python cli/library for orchestrating topologies.

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, two distinct options are offered:

- Docker wrapped instances for deployment in docker centric environments such as containerLab and KNE.
- A specialized 8xxx instance which integrates seamlessly with graphical user environments such as Cisco Modeling Labs, GNS3, and EVG-NG. There is alpha release of this software from EFT7.7.

## Release Files

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

| File | Content |
|---|---|
| )00-emulator-rrrr.tar | in software package. “rrrr” represents the release version |
| )00-xxxx-f-images-rrrr.tar | IOS-XR virtual disk files for fixed chassis |
| )00-xxxx-d-images-rrrr.tar | IOS-XR virtual disk files for modular chassis |
| )00-xxxx-iso-rrrr.tar | ion IOS-XR ISO installation file |
| )00-sonic-rrrr.tar | l SONIC related images |
| )00-yyyy-rrrr.tar | nal optional packages |
| HA256SUM | checksum of the respective tar files |
| )00*_installation_guide.pdf | |

To validate the tar file, run the Linux command:

```
sha256sum -c SHA256SUM
```

## Minimal Required Tar files

| Use Case | Tar Files |
|---|---|
| nimal Fix chassis Simulation | nulator-rrr.tar<br>xx-f-images-rrrr.tar |
| nimal modular chassis Simulation | nulator-rrr.tar<br>xx-d-images-rrrr.tar<br>xx-iso-rrrr.tar |
| nimal SONIC chassis Simulation | nulator-rrr.tar<br>nic-rrrr-tar |

Contact your Cisco Account Manager to download the Cisco 8000 Emulator software package. Then, follow the steps below to install and access the Emulator.

## Supported Platforms & Boards

| Platform Type | Platform |
|---|---|
| Fixed Chassis | 8201<br>8201-32FH<br>8202<br>8202-32FH-M<br>8212-48FH-M<br>8101-32H<br>8102-64H<br>8111-32EH |
| Centralized chassis | 86-MPA-14H2FH-M |
| Modular Chassis | 8808<br>8804 |
| Modular Chassis Boards | 8800-RP<br>88-LC036FH-M<br>88-LC36FH<br>88-LC48H<br>88-LC1-36EH<br>8808-FC<br>88-LC0-34H14FH |

| | |
|---|---|
| Optical router | NCS1010<br>NCS1014 |
| Other routers | Xrv9k |

## Included Images

| Platform Type | Platform |
|---|---|
| IOS-XR | 7.3.6, 7.9.2, 7.11.1 |
| 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<br>64G+ Mem | Ubuntu22 | Optional: CentOS8(docker (Ubuntu22)) |
| AWS | Bare Metal Instance | Bare metal Instance<br>M5d.metal | Ubuntu22 | Optional: CentOS8(docker (Ubuntu22)) |
| Azure | Virtual Machine | 16+ cores<br>64G+ Mem | Ubuntu22 | Requires nested Virtualization. |
| Google | Virtual | 16+ cores | Ubuntu22 | Requires nested Virtualization. |

| | | | | |
|---|---|---|---|---|
| Cloud | Machine | 64G+ Mem | | |
| ESXI | Virtual<br>Private Cloud | 16+ cores<br>64G+ Mem | Ubuntu22 | Requires nested Virtualization. |
| Windows10 | Virtual<br>Desktop | 8+ cores<br>16G+ Mem | Microsoft HyperV<br>untu22) | Requires nested Virtualization. |
| Windows10 | Virtual<br>Desktop | 8+ cores<br>16G+ Mem | VMware<br>Workstation Pro<br>(Ubuntu22) | Requires nested Virtualization |
| MacOS | Virtual<br>Desktop | 8+ cores<br>16G+ Mem | Fusion<br>(Ubuntu22) | Requires Nested Virtualization |

## Runtime Requirements

There are two parts to the runtime CPU and memory requirements: the emulator, and the guest network operating system. The emulator normally requires 2 cores and 2Gbytes of memory to run a virtual board.  The supported “guest” network operating systems are IOSXR7 and SONIC. The smallest instantiation of IOS-XR7 will be on the 8201 with default setting of 4 virtual cores and 32 Gbytes of memory. In resource constraint settings, IOS-XR7 will run with as little as 2 cores and 12 Gbytes of memory.

The memory and CPU requirement per emulated router is dependent on the chassis size and choice of guest network operating system.

| Emulator | Operating System | CPU | Memory | Min Memory | Disk | Comment |
|---|---|---|---|---|---|---|
| 8201 | IOS-XR7 | 4 | 20-32G | 12G | 30G+ | |
| 8802 | IOS-XR7 | 4 | 20-32G | 12G | 30G/board | |
| 8804/8808 | IOS-XR7 | 8*<br>(RP+LC) | 64G*<br>(2x32) | 40G*<br>(2x20) | 30G+ | |
| 8201 | SONIC | 2 | 8-12 | 8 | 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.

```
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/CentOS7+, Ubuntu22+, or Fedora.

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/xr7/7.3.2/8201/8201-732.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:

- 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 below commands:

```
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 “6.6”.

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

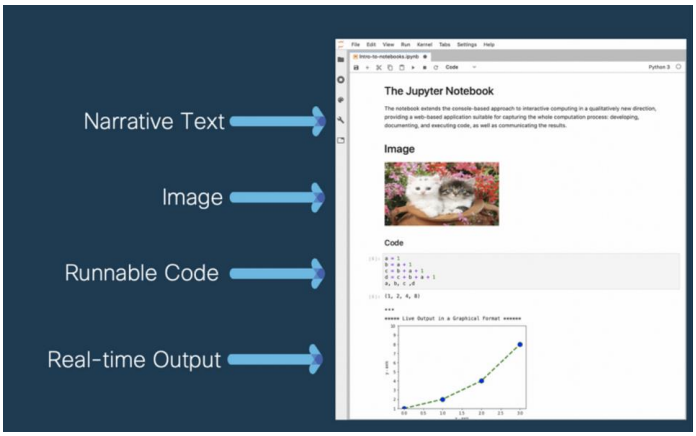
```
./scripts/aws/awsLaunchInstance.sh -r region
```

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

5. Follow the directions displayed to access to 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.




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
```

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.

## 8201e-VM

The 8201e-vm is an experimental virtual machine which includes all required simulation components of 8201 board along with a corresponding IOS-XR image. This form factor enables easier integration with graphical network simulation environments as they expect nodes to consist of a single VM. Validated environments include Cisco Modeling Lab (CML), GNS3, and EVE-NG.

## CML Installation

1. CML 241 or higher is required.

2. A bare metal CML installation is highly recommended.
3. As admin (Linux shell) `scp/wget cml241-8201e-x.y.z.tar` into CML's `/var/tmp` directory
4. Unpack: `tar -xvf cml241-8201e-x.y.z.tar`
5. `sudo bash`
6. `./install-8201e.x.y.z.sh`
7. restart CML services.
8. Verify 8201e x.y.z node type available.

## GNS3 Installation

1. Install latest GNS3 onto a bare metal server.
2. scp gns3-8201e-x.y.z.tar into the server/VM and unpack content.
3. Start GNS3 GUI, import cisco-8201e-x.y.z.gns3a from the unpacked area as a new appliance