Cisco 8000 Hardware Emulator Installation Guide

Last updated: 03/25/2022

Release Files

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

File	Content
)0-xxxx.tar	main software package. Xxxx represents the release version
)0-images-xxxx.tar	-XR qcow2 files and other images
\256SUM	256 checksum of the respective tar files
)0*_installation_guide.pdf	; file

To validate the tar file, run the Linux command:

```
sha256sum -c SHA256SUM
```

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

System Requirements

The emulators employ hypervisor technology to implement 8000 system models and perform best when run directly on X86 hardware without any other virtualization layer. Currently the emulator binaries are compiled for Ubuntu 18.04. Future editions will include CentOS8 binaries. To support a topology of routers, high scale systems based on newer generation of X86 CPUs is recommended. Servers should be ideally using NVMe/SSD drives for highest IO bandwidth.

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

System	Type	Minimal System	Operating System	Note
licated Server	e :al Server	+ cores 3+ Mem	ıntu18.04	ional: CentOS8(docker(Ubuntu18 .4))
S	e Metal lic Cloud	e metal tance d.metal	ıntu18.04	ional: CentOS8(docker(Ubuntu18 .4))
re	ual lic Cloud	cores 3+ Mem	ıntu18.04	quires nested Virtualization

I	ual ate Cloud	+ cores 3+ Mem	ıntu18.04	ıuires nested Virtualization
dows10	ual ktop	cores 3+ Mem	rosoft erV untu18.04)	ıuires nested Virtualization
dows10	ual ktop	cores 3+ Mem	ware rkstation Pro untu18.04)	ıuires nested Virtualization
cOS	ual ktop	cores 3+ Mem	ion untu18.04)	ıuires 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	n Memory	Disk	Comment
1	-XR7		3	3	3	
2	-XR7		3	3	3	ure release
8	-XR7	+LC)	3*	3*	3	ure Release

* 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.

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 18.04 onto server. Verify /dev/kvm present.

```
ls /dev/kvm
```

3. Download all the 8000*.tar files from the 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
```

Install Emulator on Virtualized Environments

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

1. Install Ubuntu 18.04 onto 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 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 18 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+, Ubuntu18+, 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 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.n.m/8201/8201-7nm.yaml .
   # launch
   vxr.py start 8201-7nm.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:

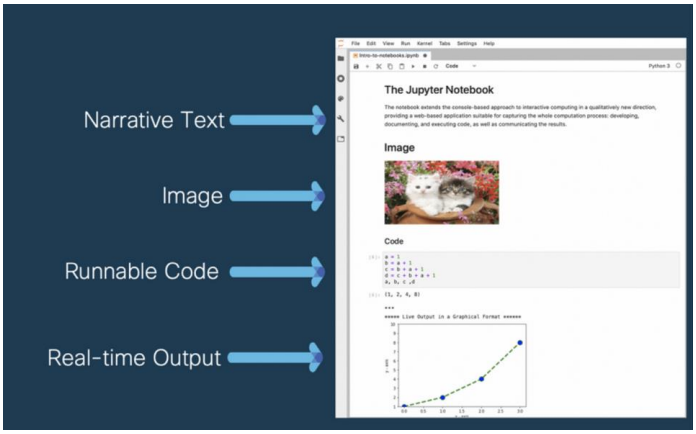
```
./scripts/aws/awsLaunchInstance.sh -r region
```

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 2 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.