Cisco 8000 Hardware Emulator Installation Guide

Last updated: 11/22/2021

Release Files

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

File	Content
8000-xxxx.tar	The main software package. Xxxx represents the release version
8000-images-xxxx.tar	IOS-XR qcow2 files and other images
SHA256SUM	Sha256 checksum of the respective tar files
8000*_installation_guide.pdf	This 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
Dedicated Server	Bare Metal Server	16+ cores 64G+ Mem	Ubuntu18.04	Optional: CentOS8(docker(Ubuntu18.4))
AWS	Bare Metal Public Cloud	Bare metal Instance M5d.metal	Ubuntu18.04	Optional: CentOS8(docker(Ubuntu18.4))
Azure	Virtual Public Cloud	16+ cores 64G+ Mem	Ubuntu18.04	Requires nested Virtualization

ESXI	Virtual Private Cloud	16+ cores 64G+ Mem	Ubuntu18.04	Requires nested Virtualization
Windows10	Virtual Desktop	8+ cores 16G+ Mem	Microsoft HyperV (Ubuntu18.04)	Requires nested Virtualization
Windows10	Virtual Desktop	8+ cores 16G+ Mem	VMware Workstation Pro (Ubuntu18.04)	Requires nested Virtualization
macOS	Virtual Desktop	8+ cores 16G+ Mem	Fusion (Ubuntu18.04)	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	32G	12G	30G	
8802	IOS-XR7	4	32G	12G	30G	Future release
8808	IOS-XR7	8* (RP+LC)	64G*	24G*	30G	Future 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 the 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 the 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 and Run the Emulator in a Docker Container

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.

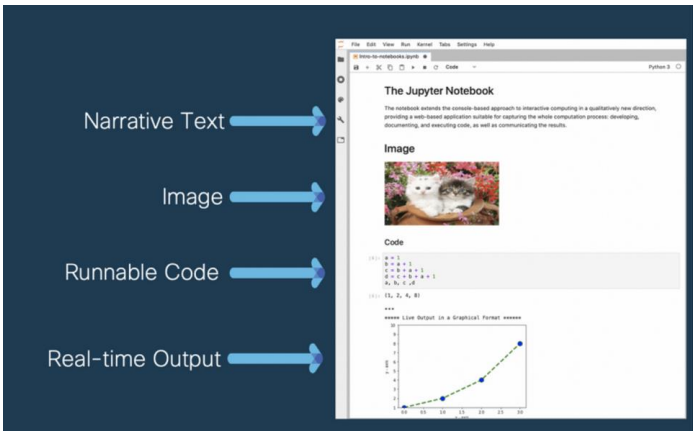
The following steps shows you how to build the docker image and run it:

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 linux image that is to be used in 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 commands:
   ```
   cd 8000-x.y/
   ./packages/packer/centos7Serial/buildCentOS7Serial.sh -d
   ```
4. Build the docker image. Assuming **x.y** is the current emulator release, you can use the commands:
   ```
   scripts/build_docker_image.sh 8000:x.y ./docker/Dockerfile.generic
   ```
5. 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
   ```
6. 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**.

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: Install Notebooks on Docker containers on your Linux server

  Or

- Option 2: Install Notebooks on your AWS instance

Install Notebooks on Docker containers on your Linux server

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

1. Execute steps 1-5 in the previous section “Install and Run the Emulator in a Docker Container”
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 on any of the README files, such as README.ipynb or README_notebooks.pdf from the file explorer on the left pane. This document explains how to use notebooks and provides a list of available notebooks.

There are many more notebooks in the folders Getting-Started and Put-Technology-to-Work. Double-click on any file with the extension .ipynb to open the notebook and try it out.

8. To exit Jupyter lab on the docker container, use ctrl-c twice. After that, if you want to access the notebooks again, just 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

Install Notebooks on your AWS instance

There are 3 main tasks to get started with notebooks on your AWS Instance:

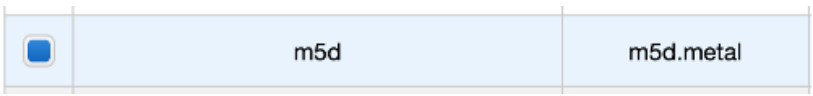
1. Launch an AWS instance with Ubuntu version 18.04 and create the AWS AMI
2. Launch an AWS instance with the newly created AWS AMI
3. Access Cisco Network Notebooks on the AWS instance

Launch an AWS instance with Ubuntu version 18.04 and create the AWS AMI

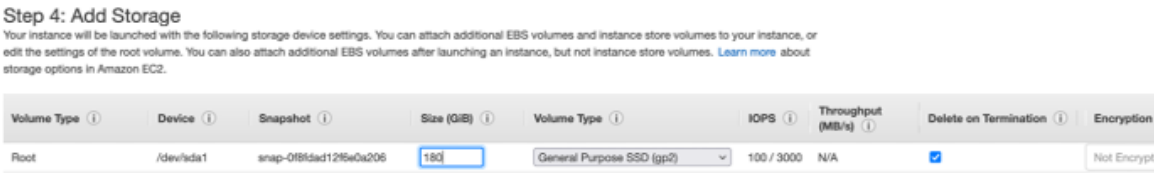
The following steps will guide you to create an AWS AMI (image) from the downloaded software package:

1. Download the tar files of the software package.
2. On the AWS Management Console, select a region that is closest to you.
3. Go to Services menu on top left corner and select EC2 under Compute section
4. Select Instances from the Instances section in the left pane.
5. Click on the orange button Launch Instance on the top right corner.
6. Select Ubuntu version 18.04 LTS.

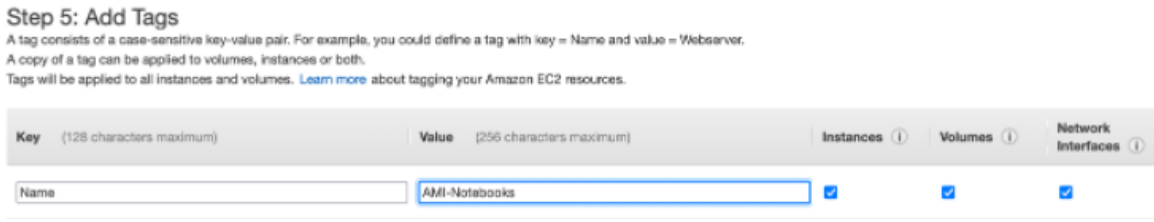
7. Use these specifications in the subsequent steps:
   a. **Instance type**: Select instance type as m5d.metal and then click the button Next: Configure Instance Details in the bottom right corner.



   b. **Instance details:** Do not change from default values. Click the button Next: Add Storage in the bottom right corner.
   c. **Storage:** Update the Size (GiB) for Root to 180 as shown in the picture and then click the button Next: Add Tags in the bottom right corner.



d. **Tags:** Add a tag with the key as Name and Value as per your preference for easy identification. Then click the button Next: Configure Security Group in the bottom right corner.



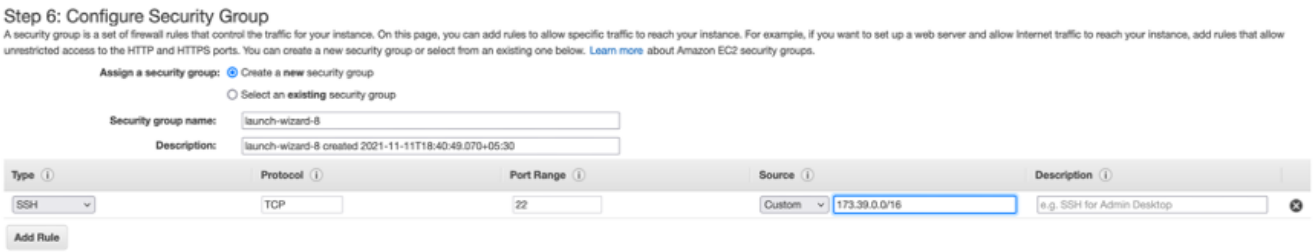
e. **Security Group**: Select "**Create a new security group**" option. Find the public IPv4 address of your computer. You can do this by accessing the following URL - https://checkip.amazonaws.com/.

If you use a server instead of a workstation, use the following command in the terminal to obtain the IPv4 address:

curl https://checkip.amazonaws.com

Choose **Type** as SSH, **Protocol** as TCP, **Port Range** as 22. Set the **Source** as the first 2 octets of the Public IPv4 address that you obtained above, followed by ".0.0/16". as shown in the blue box in the below image. For example, if your public IPv4 address is 173.36.20.5, then set the Source as "173.36.0.0/16".

Set **Description** with any string of your preference.



f. Then click the blue button **Review and Launch** in the bottom right corner.
g. **Review Launch Instance:** Click the blue button Launch in the bottom right corner.
h. Select an existing key pair or create a new key pair as directed in the pop-up window. Tick the acknowledgement and then click the blue button **Launch Instances** in the bottom right corner.
i. Wait until the instance is launched, and the **status** is **2/2 checks** passed in the **Instances summary.**

8. Once the instance is ready, copy the public IPv4 address of the instance.
9. Use the **scp** command to copy the tar files from your computer to the AWS instance:

```
scp -i <path-to-key-pair>/key-pair.pem *.tar ubuntu@<public-ipv4-address-of-AWS-instance>:/tmp/.
```

10. After the copy is complete, access the AWS instance using **ssh**:

```
ssh -i <path-to-key-pair>/key-pair.pem ubuntu@<public-ipv4-address-of-AWS-instance>
```

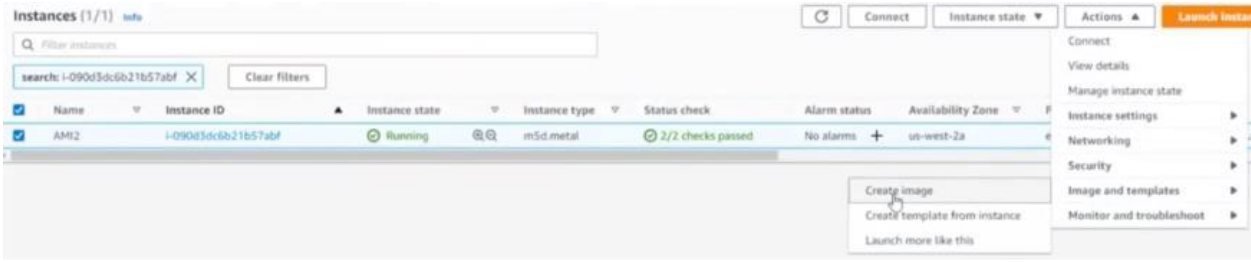
11. Extract the tar files copied in step 9 and run the script **buildAWSAMIPrep.sh**

```
cd /tmp

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

cd 8000-eftx.y

sudo ./scripts/aws/buildAWSAMIPrep.sh
```

12. It takes about 30 minutes to an hour for the script to complete. After the script execution is complete, delete the tar files and the extracted folder.

```
rm -rf /tmp/*.tar

rm -rf /tmp/8000-eftx.y
```

13. Then from the AWS Management Console, choose **Create Image** from the **Actions->Image and templates** menu



14. Add the **Image name** and **Description** as per your preference. Retain the **Volume size** as **180**. Then click on the orange button **Create Image** in the bottom right corner.
15. You can view the status of the Image Creation by clicking the link to the new AMI id.

16. Once the AMI **Status** is available, terminate the current AWS instance and create an instance with the newly created AMI by clicking the blue button **Launch** in the top right corner.

Launch an AWS instance with the newly created AWS AMI

1. Use the following specifications while launching the instance:

a. **Instance type:** Select instance type as **m5d.metal** and then click the button **Next: Configure Instance Details** in the bottom right corner.
b. **Instance details**: Don't change from default values. Click the button **Next: Add Storage** in the bottom right corner.
c. **Storage:** Don't change from default values. Click the button **Next: Add Tags** in the bottom right corner.
d. **Tags:** Add a tag with the **key** as **Name** and **Value** as per your preference, for easy identification. Then click the button **Next: Configure Security Group** in the bottom right corner.
e. **Security Group:** Select "**Create a new security group**" option. Configure a security group with an inbound rule of **type SSH**, **source** as "**MY IP**" and another inbound rule of **type custom TCP**, **port 8889** and **source** as "**MY IP**"
f. Then click the blue **b**utton **Review and Launch** in the bottom right corner.
g. **Review Launch Instance:** Click the blue button **Launch** in the bottom right corner.
h. Select an existing key pair or create a new key pair as directed in the pop-up window. Tick the acknowledgement and then click the blue button **Launch Instances** in the bottom right corner.
i. Wait until the instance is launched, and the **status** is **2/2 checks passed** in the Instances summary.

2. Once the instance is ready, copy the public IPv4 address of the instance.
3. Access the AWS instance using **ssh** from the terminal:

```
ssh -i <path-to-key-pair>/key-pair.pem ubuntu@<public-ipv4-address-of-AWS-instance>
```

Access Cisco Network Notebooks on the AWS Instance:

1. 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
```

2. Open browser on your computer and access notebooks by entering ***<public-ipv4-address-of-AWS-instance>:8889*** in the address-bar. Use the notebooks password that you have set up in step 1 of this section.

3. 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.