Deploying and using vJunos in a bare metal EVE-NG server

Authors: Aninda Chatterjee & Shalini Mukherjee, TME, Juniper Networks Cloud-Ready Data Center

In this article, we will look at how vJunos-switch is deployed on a bare metal install of EVE-NG. This deployment will then be used to integrate with Juniper Apstra and build a complete Data Center fabric.

We'll start with instructions on how to deploy EVE-NG as a bare metal install on an Ubuntu server and run through the EVE-NG installation itself as an example of how this can be done. Once EVE-NG is setup, we'll show how vJunos-switch can be added to EVE-NG as a deployable node and the EVE-NG template that is needed for this to work.

Finally, we'll wrap this up by onboarding vJunos-switch devices in Juniper Apstra and building a simple Data Center fabric to show that these devices work seamlessly with Apstra as well.

What is EVE-NG and vJunos-switch?

EVE-NG is a popular network emulation software that offers multivendor support. It allows a user to build network topologies on a blank canvas, providing means to test and simulate various network technologies and features across different network operating systems and products.

EVE-NG has various deployment options – installing EVE-NG as a VM on top of a hypervisor like ESXi, a bare metal install of EVE-NG which implies installing it over a bare metal server and some cloud deployment options as well like GCP.

vJunos-switch is a new virtual software offering from Juniper Networks. vJunos-switch emulates the software functionality of Junos OS. Since vJunos-switch is only supported on a bare metal install of EVE-NG, that is the deployment option that we'll be going through in this blog post.

Installing EVE-NG on a bare metal Ubuntu server

EVE-NG comes in two flavors – a community (free) edition, and a professional (paid) edition. We'll take a look at how to do a bare metal EVE-NG community edition install in this post, since that is more complicated. The .iso file for the community edition can be downloaded from the EVE-NG website - https://www.eve-ng.net/index.php/download/

Once this file is downloaded, mount the .iso file on your local machine. For example, on a MAC, you can simply double click the file to mount it. This should create a new drive, whose contents can be accessed now:

```
anindac@ubuntu EVE-NG Community % ls -l
total 216
dr-xr-xr-x  3 aninchat  staff   2048 Feb 23  2022 EFI
dr-xr-xr-x  3 aninchat  staff   2048 Feb 23  2022 boot
dr-xr-xr-x  3 aninchat  staff   2048 Feb 23  2022 casper
dr-xr-xr-x  3 aninchat  staff   2048 Feb 23  2022 dists
dr-xr-xr-x  2 aninchat  staff   2048 Feb 23  2022 install
dr-xr-xr-x  2 aninchat  staff  36864 Feb 23  2022 isolinux
-r--r--r--  1 aninchat  staff  27257 May 21  2022 md5sum.txt
-r--r--r--  1 aninchat  staff  27389 Feb 23  2022 md5sum.txt.orig
dr-xr-xr-x  3 aninchat  staff   2048 Feb 23  2022 pool
```

```
dr-xr-xr-x  2 aninchat  staff   2048 Feb 23  2022 preseed
dr-xr-xr-x  2 aninchat  staff   2048 May 20  2022 server
```

Here, should see a folder titled ‘server’. This has a shell script that installs EVE-NG as a bare metal install. The script is as follows:

```
anindac@ubuntu EVE-NG Community % cd server
anindac@ubuntu server % ls -l
total 8
-r-xr-xr-x  1 aninchat  staff  802 May 20  2022 eve-setup.sh
-r--r--r--  1 aninchat  staff    0 May 14  2022 meta-data
-r--r--r--  1 aninchat  staff  956 May 20  2022 user-data

anindac@ubuntu server % cat eve-setup.sh
#!/bin/sh
#Modify /etc/ssh/sshd_config with: PermitRootLogin yes
sed -i -e "s/.*PermitRootLogin .*/PermitRootLogin yes/" /etc/ssh/sshd_config
sed -i -e 's/.*DefaultTimeoutStopSec=.*/DefaultTimeoutStopSec=5s/'
/etc/systemd/system.conf
systemctl restart ssh
apt-get update
apt-get -y install software-properties-common
wget -O - http://www.eve-ng.net/focal/eczema@ecze.com.gpg.key | sudo apt-key add -
sudo add-apt-repository "deb [arch=amd64]  http://www.eve-ng.net/focal focal main"
apt-get update
DEBIAN_FRONTEND=noninteractive apt-get -y install eve-ng
/etc/init.d/mysql restart
DEBIAN_FRONTEND=noninteractive apt-get -y install eve-ng
echo root:eve | chpasswd
ROOTLV=$(mount | grep ' / ' | awk '{print $1}')
echo $ROOTLV
lvextend -l +100%FREE $ROOTLV
echo Resizing ROOT FS
resize2fs $ROOTLV
reboot
```

On your Ubuntu 20.04 server, simply run through these steps one by one. Once you reboot (the final step of the script), and the server comes back online, you should be able to SSH into the device with the default credentials of root/eve.

Once you’ve logged in, the actual EVE-NG installation is kicked off – this includes providing a hostname, IP address and gateway, primary and secondary DNS servers, NTP servers and so on. When this finishes, EVE-NG should be up and running, and accessible via both the CLI and UI (default credentials for UI login is admin/eve).

To deploy the professional version of EVE-NG, similar steps can be followed. Alternatively, the .iso file can be mounted directly as a virtual drive and you can boot from it to start both the Ubuntu and the EVE-NG installation together.

Deploying vJunos-switch in EVE-NG

EVE-NG uses templates to boot various operating systems it supports – these templates are pre-built and come packaged within the installation of EVE-NG. They contain instructions on how to boot the devices, including how many interfaces to assign to the device, the kind of CPU/memory the device needs and so on.

The templates are stored in the path '*/opt/unetlab/html/templates/intel/*' and are written in YAML format, as seen below:

```
root@eve-ng:/opt/unetlab/html/templates/intel# ls -l
total 588
-rw-r--r-- 1 root root   10 Jan 31 07:15 '*.yml'
-rw-r--r-- 1 root root 1944 May 14  2022  128T.yml
-rw-r--r-- 1 root root 1853 May 14  2022  a10.yml
-rw-r--r-- 1 root root 1897 May 14  2022  acs.yml
-rw-r--r-- 1 root root 1913 May 14  2022  alienvault.yml
-rw-r--r-- 1 root root 1911 May 14  2022  alteon.yml
-rw-r--r-- 1 root root 1848 May 14  2022  ampcloud.yml
-rw-r--r-- 1 root root 1885 May 14  2022  android.yml
-rw-r--r-- 1 root root 1921 Jan 31 07:15  aos.yml
-rw-r--r-- 1 root root 1890 May 14  2022  apicem.yml
-rw-r--r-- 1 root root 1952 May 14  2022  apstra-aos.yml
-rw-r--r-- 1 root root 1951 May 14  2022  apstra-ztp.yml
-rw-r--r-- 1 root root 1950 May 14  2022  aruba.yml
-rw-r--r-- 1 root root 1919 May 14  2022  arubaamp.yml
-rw-r--r-- 1 root root 2002 May 14  2022  arubacx.yml
-rw-r--r-- 1 root root 1930 May 14  2022  arubanet.yml
-rw-r--r-- 1 root root 1971 May 14  2022  asa.yml
-rw-r--r-- 1 root root 1950 May 14  2022  asav.yml
-rw-r--r-- 1 root root 1899 May 14  2022  barracuda.yml
-rw-r--r-- 1 root root 1930 May 14  2022  bigip.yml
-rw-r--r-- 1 root root 1915 May 14  2022  brocadevadx.yml
-rw-r--r-- 1 root root 1829 May 14  2022  c1710.yml
-rw-r--r-- 1 root root 1931 May 14  2022  c3725.yml
-rw-r--r-- 1 root root 1993 May 14  2022  c7200.yml
-rw-r--r-- 1 root root 2017 May 14  2022  c8000v.yml
-rw-r--r-- 1 root root 1944 May 14  2022  c9800cl.yml
-rw-r--r-- 1 root root 1839 May 14  2022  cda.yml
-rw-r--r-- 1 root root 1981 May 14  2022  cexpresw.yml
-rw-r--r-- 1 root root 1973 May 14  2022  cips.yml
-rw-r--r-- 1 root root 1974 Jul  8  2022  clavisterc.yml
-rw-r--r-- 1 root root 1925 May 14  2022  clearpass.yml
-rw-r--r-- 1 root root 1917 May 14  2022  cms.yml
-rw-r--r-- 1 root root 1862 May 14  2022  coeus.yml
-rw-r--r-- 1 root root 1935 May 14  2022  cpsg.yml
-rw-r--r-- 1 root root 1938 May 14  2022  csr1000v.yml
-rw-r--r-- 1 root root 1967 May 14  2022  csr1000vng.yml
-rw-r--r-- 1 root root 1997 May 14  2022  ctxsdw.yml
-rw-r--r-- 1 root root 1966 May 14  2022  cuc.yml
-rw-r--r-- 1 root root 1931 May 14  2022  cucm.yml
-rw-r--r-- 1 root root 1909 May 14  2022  cue.yml
-rw-r--r-- 1 root root 1870 May 14  2022  cumulus.yml
-rw-r--r-- 1 root root 1998 May 14  2022  cup.yml
-rw-r--r-- 1 root root 1955 May 14  2022  cvp.yml
-rw-r--r-- 1 root root 1878 May 14  2022  cyberoam.yml
-rw-r--r-- 1 root root 1957 May 14  2022  dcnm.yml
-rw-r--r-- 1 root root 1969 May 14  2022  dellos10.yml
-rw-r--r-- 1 root root 1769 May 14  2022  docker.yml
-rw-r--r-- 1 root root 1920 May 14  2022  esxi.yml
-rw-r--r-- 1 root root 1862 May 14  2022  extremexos.yml
-rw-r--r-- 1 root root 1898 May 14  2022  firepower.yml

*snip*
```

Since vJunos-switch is an entirely new offering, there is no pre-built template for this. The template must be created manually. The following templates can be used for vJunos-switch:

vJunos-switch:

```
root@eve-ng:/opt/unetlab/html/templates/intel# cat vJunos-switch.yml
---
type: qemu
description: vJunos-switch
name: vJunos-switch
cpulimit: 4
icon: JunipervQFXre.png
cpu: 4
ram: 5120
eth_name:
- fxp0
- ge-0/0/0
- ge-0/0/1
- ge-0/0/2
- ge-0/0/3
- ge-0/0/4
- ge-0/0/5
- ge-0/0/6
- ge-0/0/7
- ge-0/0/8
- ge-0/0/9
ethernet: 11
console: telnet
qemu_arch: x86_64
qemu_version: 4.1.0
qemu_nic: virtio-net-pci
qemu_options: -machine type=pc,accel=kvm -serial mon:stdio -nographic -smbios
type=1,product=VM-VEX -cpu IvyBridge,ibpb=on,md-clear=on,spec-
ctrl=on,ssbd=on,vmx=on
```

The next step is to create a new directory for this device and copy the image to this directory. We'll create a directory with the 'vJunos-switch' followed by a suffix which specifies the version of the software image. The prefix naming convention is important – it must match the name of the template that exists for the device. EVE-NG uses this name to determine which template must be used to boot the device (hence, the need for them to match).

The images (and their directories) are stored under '*/opt/unetlab/addons/qemu/*'.

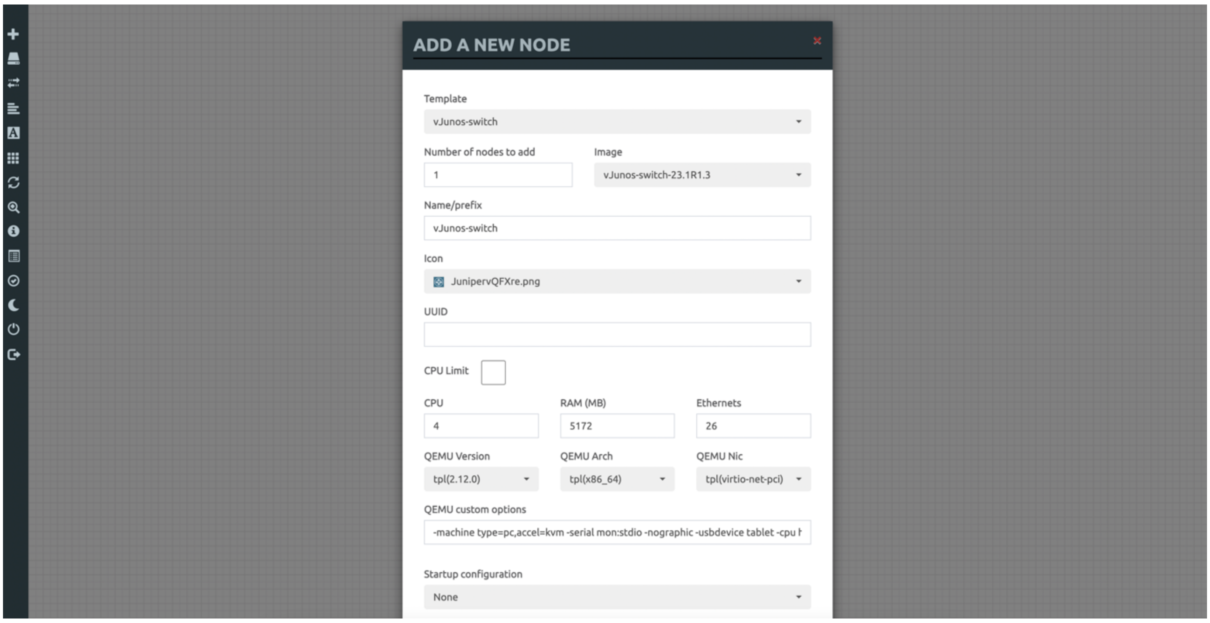
```
root@eve-ng:/opt/unetlab/addons/qemu# pwd
/opt/unetlab/addons/qemu
root@eve-ng:/opt/unetlab/addons/qemu# mkdir vJunos-switch-23.1R1.3

root@eve-ng:/opt/unetlab/addons/qemu# cd vJunos-switch-23.1R1.3/
root@eve-ng:/opt/unetlab/addons/qemu/vJunos-switch-23.1R1.3# ls -l
total 3850436
-rw-r--r-- 1 root root 3942842368 Feb 25 11:15 vjunos-switch-23.1R1.3.qcow2
```

Once the image is copied into the folder, it must be renamed to 'virtioa.qcow2' as per EVE-NGs naming convention. Finally, EVE-NG requires you to fix some permissions to use the image - they have a pre-built script for this which can be invoked using '*/opt/unetlab/wrappers/unl_wrapper -a fixpermissions*' as seen below:

```
root@eve-ng:/opt/unetlab/addons/qemu/vJunos-switch-23.1R1.3# mv vjunos-switch-
23.1R1.3.qcow2 virtioa.qcow2
root@eve-ng:/opt/unetlab/addons/qemu/vJunos-switch-23.1R1.3#
/opt/unetlab/wrappers/unl_wrapper -a fixpermissions
```

On the EVE-NG UI, you should see ‘vJunos-switch’ as a device available to use and deploy:



ADD A NEW NODE

Template: vJunos-switch

Number of nodes to add: 1

Image: vJunos-switch-23.1R1.3

Name/prefix: vJunos-switch

Icon: JunipervQFXre.png

UUID:

CPU Limit □

CPU: 4

RAM (MB): 5172

Ethernets: 26

QEMU Version: tpl(2.12.0)

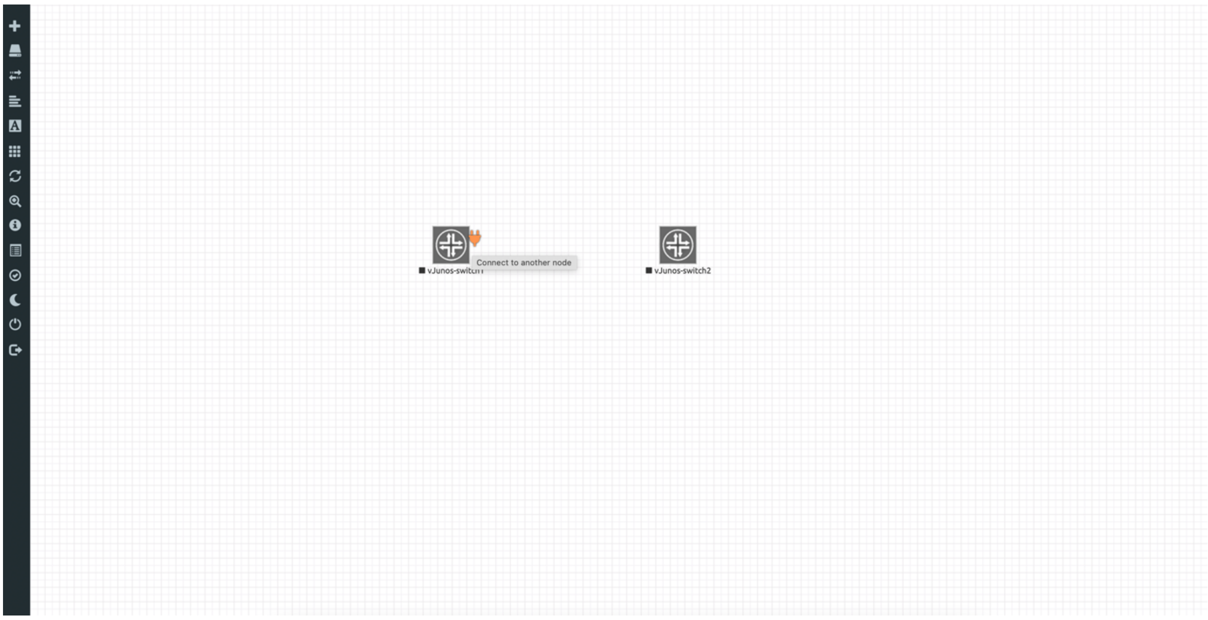
QEMU Arch: tpl(x86_64)

QEMU Nic: tpl(virtio-net-pci)

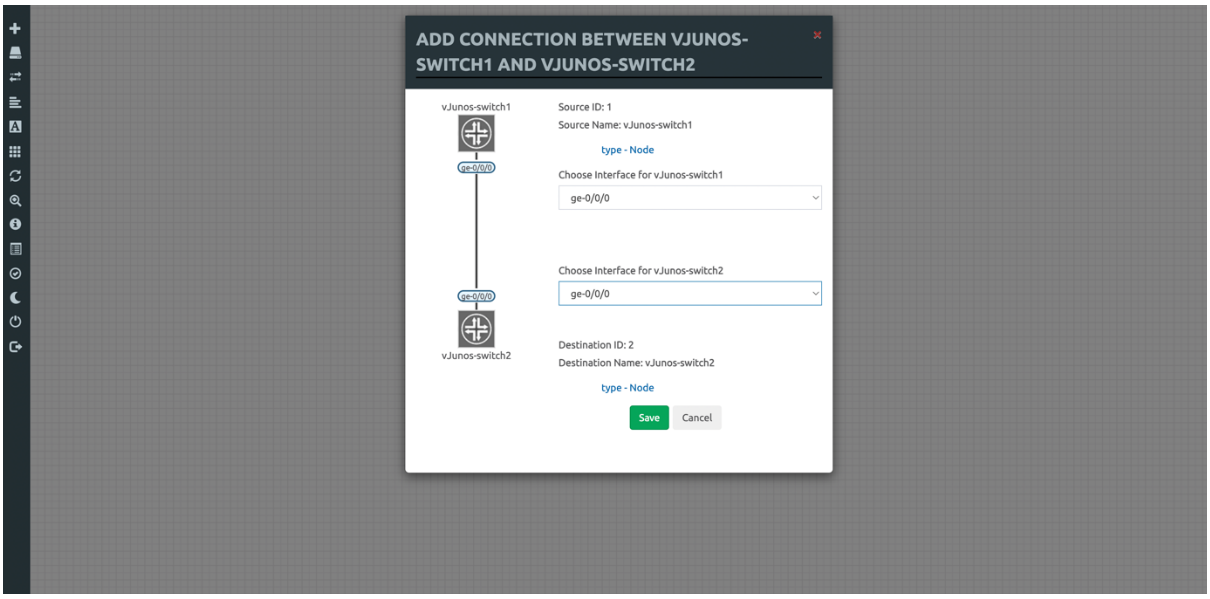
QEMU custom options: -machine type=pc,accel=kvm -serial mon:stdio -nographic -usbdevice tablet -cpu h

Startup configuration: None

These devices can now be deployed and interconnected. To interconnect two devices, simply click on the orange icon seen when you hover over a node, as seen below:



Once clicked, you can drag it to the destination node and let the connector go, which then gives you a pop-up to choose which interface is being connected on each side, like below:


ADD CONNECTION BETWEEN VJUNOS-SWITCH1 AND VJUNOS-SWITCH2

Source ID: 1

Source Name: vJunos-switch1

type - Node

Choose Interface for vJunos-switch1

ge-0/0/0

Choose Interface for vJunos-switch2

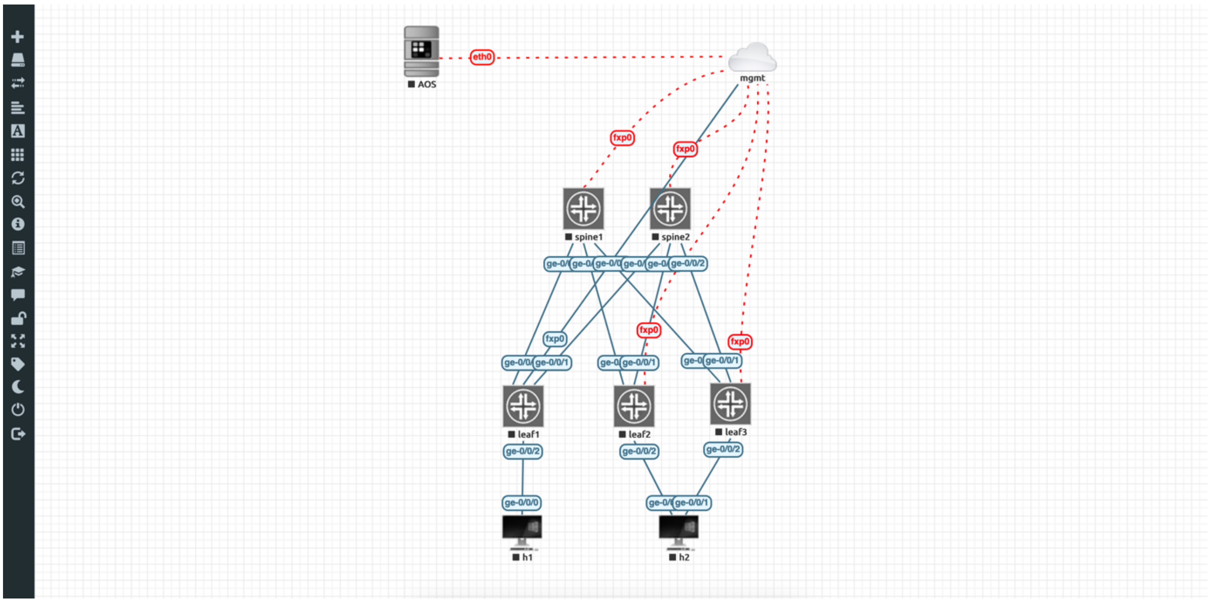
ge-0/0/0

Destination ID: 2

Destination Name: vJunos-switch2

type - Node

To demonstrate the use of vJunos-switch, we've built the following topology on EVE-NG:



This topology a typical 3-stage Clos fabric with two spines and three leafs. Two leafs (leaf2 and leaf3) are ESI peers and connect down to the same host, h2.

Deploying Juniper Apstra in EVE-NG

As seen in the topology above, we also have Juniper Apstra deployed within EVE-NG itself. For the sake of completeness, we'll also document how this is done.

The Juniper Apstra KVM image can be downloaded from the Juniper software downloads page - https://support.juniper.net/support/downloads/

The image is zipped with an extension of gz. Once unzipped, move the file to a folder created with the EVE-NG naming convention for Apstra – the directory must start with the prefix 'aos' and you can include the Apstra version as a suffix for easy identification of the image version. For example, we have created a folder named ‘aos-4.1.2-269’ and the image is under that. The image is finally renamed to ‘hda.qcow2’:

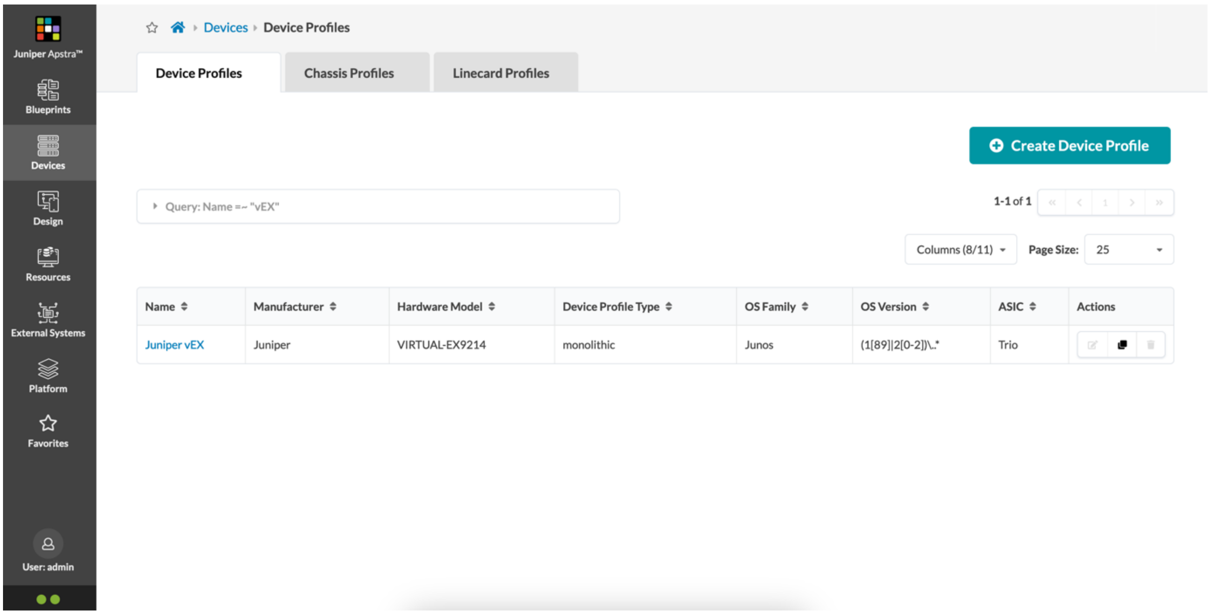
```
root@eve-ng:/opt/unetlab/addons/qemu# cd aos-4.1.2-269/
root@eve-ng:/opt/unetlab/addons/qemu/aos-4.1.2-269# ls -l
total 2762612
-rw-r--r-- 1 root root 2828908544 Feb 25 12:14 hda.qcow2
```

The Apstra server can now be started, and some basic bootstrapping is needed – the Apstra service needs to be started, along with setting a password for the CLI and UI. Apstra allows you to pull an IP address via DHCP or set one up manually as well.

Deploying a vJunos-switch based fabric in Juniper Apstra

Now that we have all our pieces in EVE-NG, we can start to build the fabric. Nodes can be started from the EVE-NG UI by right-clicking on them and simply choosing ‘Start’. Once the spines and the leafs are assigned IP addresses via DHCP, we can onboard them into Apstra.

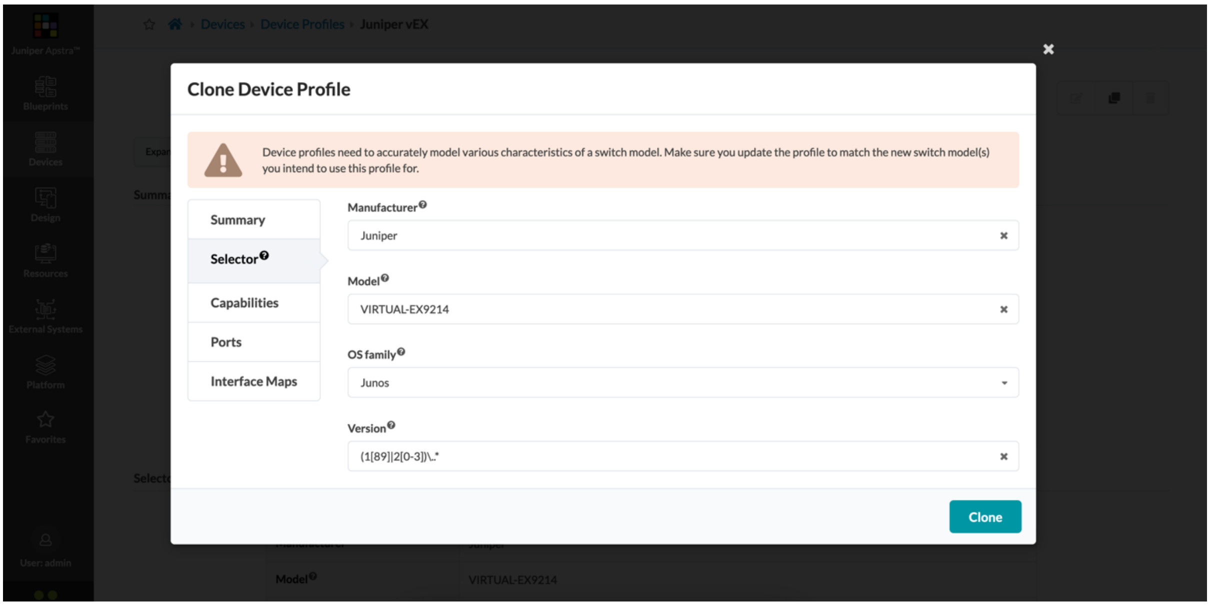
Since the current version of Apstra has not been updated with the vJunos-switch version, we need to make some adjustments. We will create a new ‘Device Profile’ for vJunos-switch by cloning the existing Juniper vEX device profile.



Query: Name =~ "vEX"

Name	Manufacturer	Hardware Model	Device Profile Type	OS Family	OS Version	ASIC	Actions
Juniper vEX	Juniper	VIRTUAL-EX9214	monolithic	Junos	(1[89]|2[0-2])\..*	Trio	

The important change in this device profile is to extend selector to include version ‘23’ like below:



Clone Device Profile

Device profiles need to accurately model various characteristics of a switch model. Make sure you update the profile to match the new switch model(s) you intend to use this profile for.

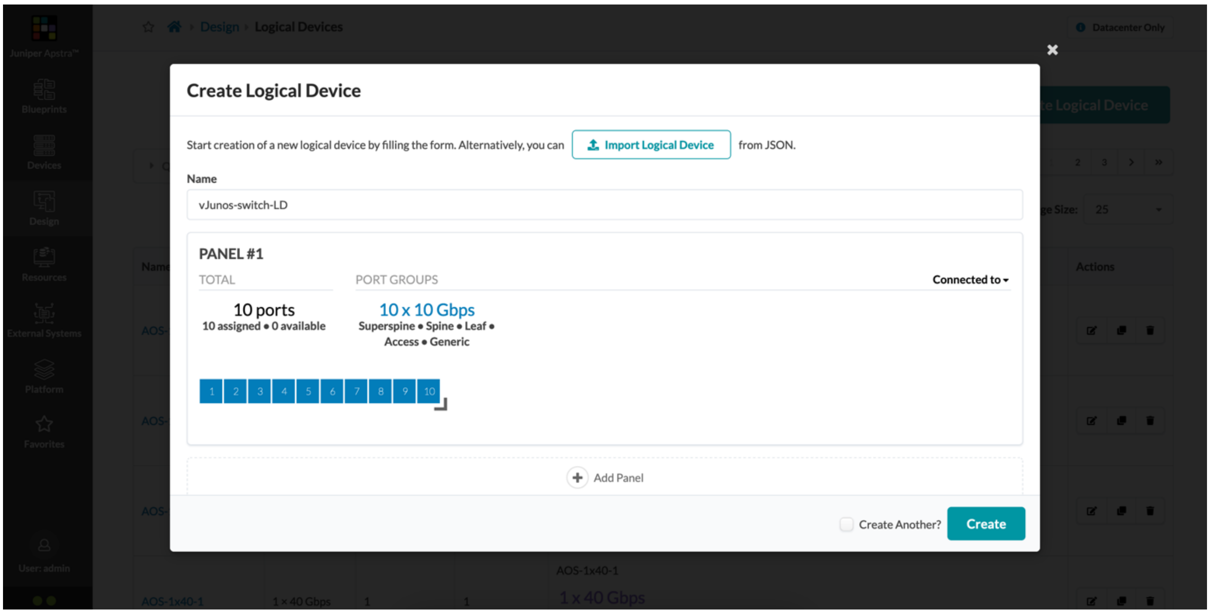
Manufacturer: Juniper

Model: VIRTUAL-EX9214

OS family: Junos

Version: (1[89]|2[0-3])\..*

The next step is to create a ‘Logical Device’ and ‘Interface Map’ in Apstra for these devices. The Logical Device simply creates the port grouping as below:



Create Logical Device

Start creation of a new logical device by filling the form. Alternatively, you can Import Logical Device from JSON.

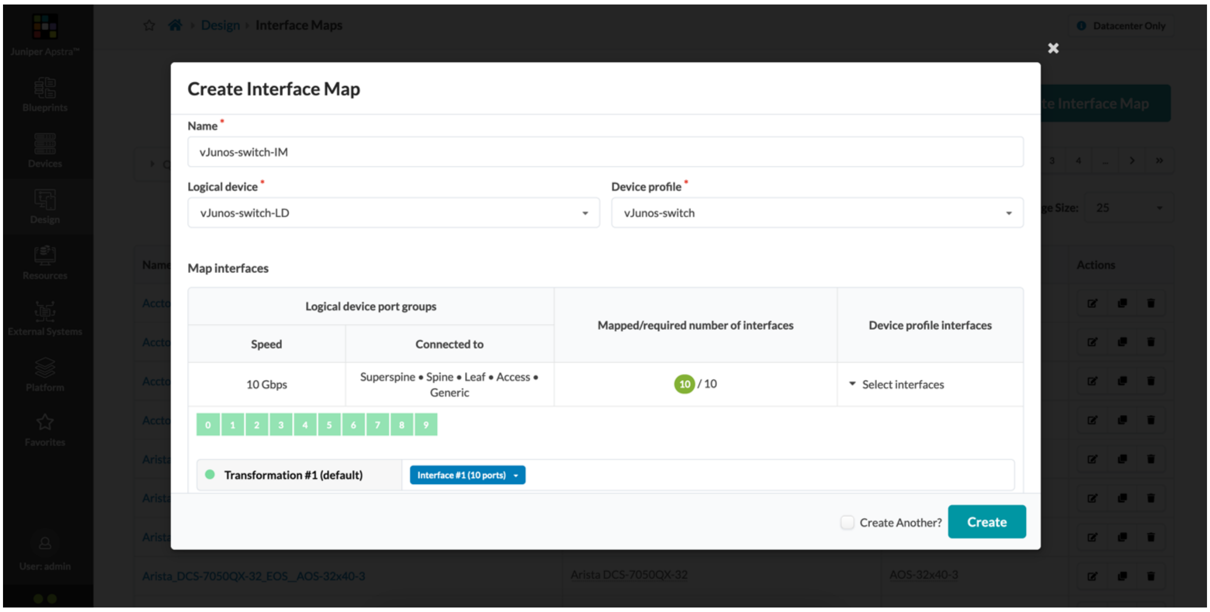
Name: vJunos-switch-LD

PANEL #1

TOTAL: 10 ports, 10 assigned • 0 available

PORT GROUPS: 10 x 10 Gbps, Superspine • Spine • Leaf • Access • Generic

The Interface Map creates the appropriate transformation (naming convention for interfaces, speed, port breakouts and so on) and glues together a logical device and a device profile:



Create Interface Map

Name *

vJunos-switch-IM

Logical device * vJunos-switch-LD

Device profile * vJunos-switch

Map interfaces

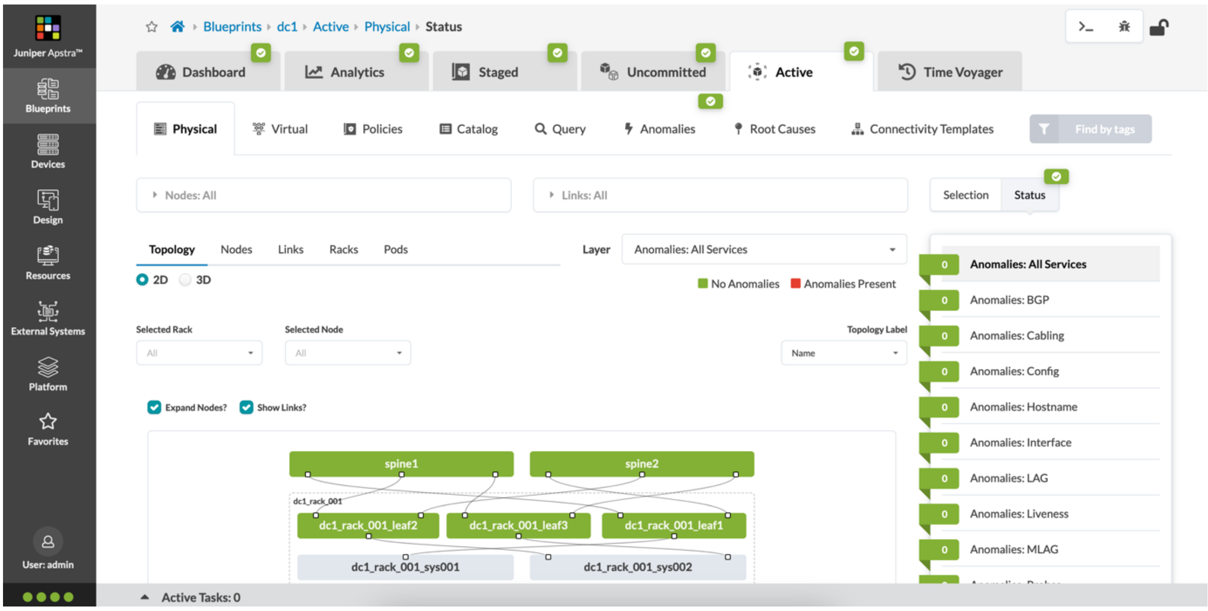
Logical device port groups		Mapped/required number of interfaces	Device profile interfaces
Speed	Connected to		
10 Gbps	Superspine • Spine • Leaf • Access • Generic	10 / 10	Select interfaces

0 1 2 3 4 5 6 7 8 9

Transformation #1 (default) — Interface #1 (10 ports)

□ Create Another? Create

Once all of this is configured, we can start building our racks and templates. The template is finally fed into a blueprint, an example of which we've deployed below:



Blueprints › dc1 › Active › Physical › Status

Dashboard | Analytics | Staged | Uncommitted | Active | Time Voyager

Physical | Virtual | Policies | Catalog | Query | Anomalies | Root Causes | Connectivity Templates | Find by tags

Nodes: All | Links: All | Selection | Status

Topology | Nodes | Links | Racks | Pods

Layer: Anomalies: All Services

● 2D ○ 3D

🟩 No Anomalies 🟥 Anomalies Present

Selected Rack: All | Selected Node: All | Topology Label: Name

☑ Expand Nodes? ☑ Show Links?

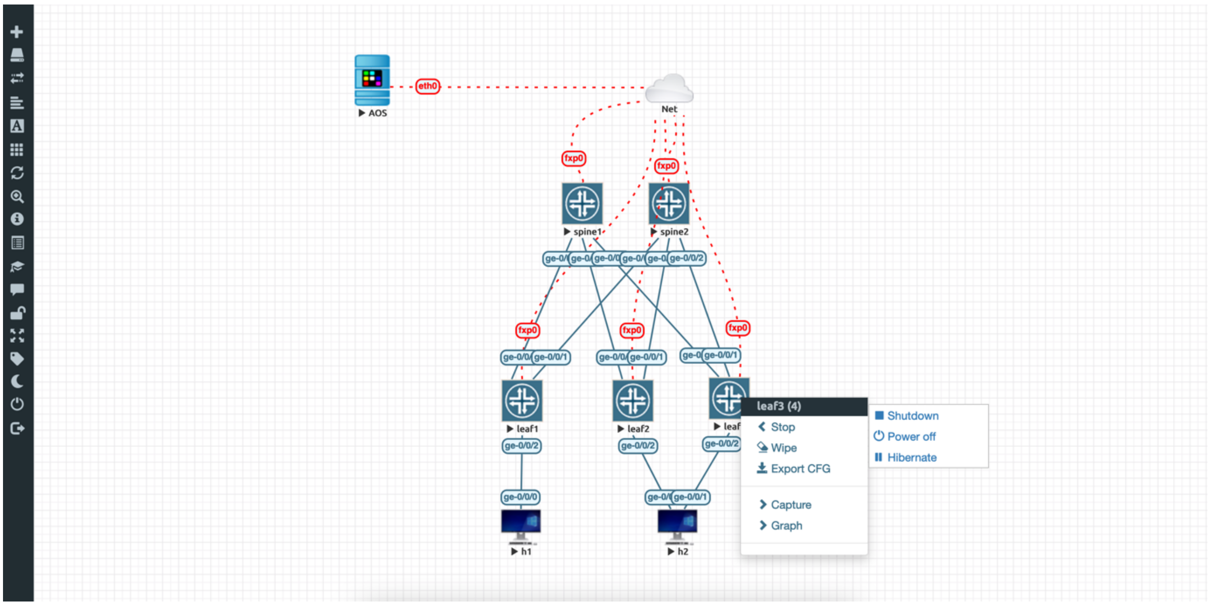
0	Anomalies: All Services
0	Anomalies: BGP
0	Anomalies: Cabling
0	Anomalies: Config
0	Anomalies: Hostname
0	Anomalies: Interface
0	Anomalies: LAG
0	Anomalies: Liveness
0	Anomalies: MLAG

Active Tasks: 0

This blueprint includes host h1 which is single attached to leaf1 and host h2 which is dual homed to leaf2 and leaf3.

A note on shutting down vJunos-switch in EVE-NG

EVE-NG offers multiple ways to shut down a node – this includes a graceful shutdown, a poweroff (not graceful) and a hibernate option. These options can be seen below when right-clicking on a node and clicking on 'Stop' (screenshot from EVE-NG Professional Edition shown below):



It is important to shut down the device gracefully (using the 'Shutdown' option). A power off (which is not graceful) has the potential of corruption the disk, rendering the device unusable. In EVE-NG community edition, issue a '*request system poweroff*' prior to shutting down the node via the UI, since these options are not available in the UI itself in this edition.

Summary

Through this post, we were able to demonstrate a working deployment of Juniper's new virtual offering, vJunos-switch, in EVE-NG and integrate it with Juniper Apstra to deploy a Data Center fabric.

Useful links

vJunos software download - https://support.juniper.net/support/downloads/?p=vjunos
vJunos documentation - https://www.juniper.net/documentation/product/us/en/vjunos-switch/
EVE-NG home page - https://www.eve-ng.net/
Juniper Apstra documentation - https://www.juniper.net/documentation/product/us/en/apstra/

Acknowledgments

Co-authors Shalini Mukherjee and Aninda Chatterjee would like to thank TME Manager Ridha Hamidi, Product Manager Yogesh Kumar, and the entire Juniper engineering staff behind this product, led by Art Stine and Kaveh Moezzi, for their guidance and help.