**Sonic-mgmt setup details for Systest [WIP]**

What is Sonic?

The Software for Open Networking in the Cloud [SONiC] , is a free and open source network operating system based on Linux, the idea was to have a vendor independent OS which could run on multiple vendor devices

This document shall detail the steps needed to bring up a sonic testbed with a specific topology [M0/T0/T1 etc ]

As an example, we are going to illustrate the steps needed to bring up a M0 topology with PikeZ device

Before proceeding, please go through the below link to get an idea about the terminologies/ hardware requirements and some of the setup details

sonic-mgmt/docs.testbed/README.testbed.Overview.md :

<https://github.com/sonic-net/sonic-mgmt/blob/master/docs/testbed/README.testbed.Overview.md>

**Hardware Requirements**

First off,we need to understand the fundamental hardware requirements

For any Topology, we need to the below set of physical devices

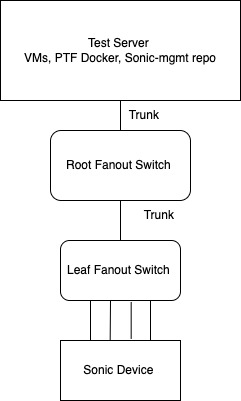
1. **Test Server** - Physical Server which will be used to run the sonic mgmt framework.
2. **Leaf Fanout switch** - Each Leaf fanout port connecting to each device port. Also one trunk interface connecting to root fanout port
3. **Root Fanout switch** - 1 trunk port connecting to the Leaf fanout and 1 trunk port connecting to the test server
4. **Sonic DUT** - Device under test to run Sonic

**Key Physical Topology aspects**

1. Every Sonic DUT port is connected to one of the leaf fanout switch ports.
2. Every leaf fanout switch has a unique VLAN tag for every DUT port. This is how your DUT ports are exposed to your test environment.
3. Root fanout switch connects leaf fanout switches and test servers using 802.1Q trunks. The root fanout switch is not mandatory if there is only one testbed or a test server is only used by one testbed. In this case, the leaf fanout switch can be directly connected with the NIC of the test server by 802.1Q trunk.
4. Any test server can access any DUT port by sending a packet with the port VLAN tag (The root fanout switch should have this VLAN number enabled on the server trunk)

In test servers, a set of VMs can be created (based on the topology that we want to run ) for running network operating system like CEos, Veos to simulate the neighbors of the sonic Dut. Also a PTF container (based on the [PTF test framework](https://github.com/p4lang/ptf)) for injecting/sniffing packets will be created. The PTF docker, the VMs (Virtual Machine) and VLAN interfaces in test server can be interconnected by [open vSwitch](https://www.openvswitch.org/) bridges. Through the VLAN tunnels of the fanout switches, the PTF docker and VMs can be connected to the ports of SONiC DUTs in flexible ways. Then different testbed topologies can be simulated by creating different number of VMs and establish different connections between the VMs and SONiC DUTs.

The high level topology looks something like below.



**Overall, from a very high level, the steps that are needed to be executed are as follows.**

1. Setup the Physical Topology - Setup the physical hardware infrastructure.
2. Setup the Test Server , Install Docker, Copy the sonic-mgmt repo etc
3. Define testbed-new.yaml file - This should contain all the topology related information
4. Run TestbedProcessing.py and make sure that the relevant files are generated. --> This step generated ~ 8 configuration files that are necessary for the add topo step
5. Run creategraph.py and make sure that your lab links file is generated : The script takes the input from the files that were generated in the previous step and creates an xml file with all the lab connections.
6. Run the add topo step: Deploys the necessary VMs on the test server and configures them with all the relevant details (eg IP addresses, desired topology configurations
7. Deploy Minigraph : Configures the sonic dut with the specified configurations for that testbed.

**Bringing up the Setup**

1. **Setting up the physical topology**

Make the physical connections as described in: <https://github.com/sonic-net/sonic-mgmt/blob/master/docs/testbed/README.testbed.Overview.md>

For reference, you can follow the below diagram. This is specific to Pike Z with no Root Fanout switch

[Topology Example](https://app.diagrams.net/#G1QGVVka2mXGSv-njYdg7Dj5g6HM74fjoR)

1. **Setting up the test server / Fanout / Switch**
2. **TEST SERVER**

Follow the steps as mentioned in the following doc: <https://github.com/sonic-net/sonic-mgmt/blob/master/docs/testbed/README.testbed.Setup.md>

NOTES

* The configuration guide explains the steps to clone the “master repo”. Once you have cloned the repo locally, change the branch to 202205 by going to the local created repo. .
* Make sure to configure the server port connected to the fanout switch with MTU 9216.
* Make sure the server NIC does not consume lldp packets. This is specific to the case if the NIC is intel based. Follow the steps as mentioned in : "<https://www.thomas-krenn.com/en/wiki/Intel_Ethernet_700_Series_LACP_Configuration>

1. **FANOUT SWITCH**

A sample fanout configs have been put in [sonic-mgmt](https://github.com/sonic-net/sonic-mgmt/tree/8929a363538b57a5c2da07f70431040930882a31)/[ansible](https://github.com/sonic-net/sonic-mgmt/tree/8929a363538b57a5c2da07f70431040930882a31/ansible)/[roles](https://github.com/sonic-net/sonic-mgmt/tree/8929a363538b57a5c2da07f70431040930882a31/ansible/roles)/[fanout](https://github.com/sonic-net/sonic-mgmt/tree/8929a363538b57a5c2da07f70431040930882a31/ansible/roles/fanout)//templates/

We can reference any fanout configs build our own fanout configs, but primarily, we need to make sure the below pointers are addressed

* Server port should be configured as trunk with all the DUT vlans being allowed
* Bypass LACP/LLDP and EAPOL. Disable LLDP as well. Your fanout switch completely bypass these packets and does not consume any of the control plan packets.
* On interfaces connecting to the dut, spread the vlans as described in the above documents and configure dot1q-tunnel mode on each access interface.

1. **SONIC DUT - SONIC installation on a DUT**

Reference: <https://docs.google.com/presentation/d/1R_J6Q7rc8GYrMaoxmYyzTSukWmxR4h6QQ6eCgNmhJyc/edit#slide=id.ge5f885ff0f_0_31>

These steps are applicable when you try to install SONiC on an Arista DUT with an eos already installed.

STEPS

eos#copy <http://dist.aristanetworks.com/Sonic/202205/sonic-aboot-broadcom/latest/sonic-aboot-broadcom.swi> flash:

eos(config)#boot system flash:sonic-aboot-broadcom.swi

eos(config)#reload

IMAGE SWI LINK for 202205 <http://dist.aristanetworks.com/Sonic/202205/sonic-aboot-broadcom/latest/sonic-aboot-broadcom.swi>

==================

You could also obtain the image from microsoft

<https://sonic-build.azurewebsites.net/ui/sonic/pipelines/1/builds/341668/artifacts/511293?branchName=202205&artifactName=sonic-buildimage.broadcom>

Login credentials for Sonic

Credentials: admin/password

//Credentials: admin/YourPaSsWoRd

Eth0 / Management Interface setup : pkz417 : 172.24.65.107

1. **Define testbed-new.yaml file**

Testbed.yaml file contains all the details about the Physical DUT, VM / PTF IPs, Physical connection details and their respective credentials.

For reference - Follow the below sample testbed-new.yaml file : <https://pb.infra.corp.arista.io/Y9aw>

1. **Run TestbedProcessing.py**

Once your testbed-new.yaml file is completed , you need to generate all the dependent files that need to be created and this can be done using the testbed processing script.

**CLI** : sudo python TestbedProcessing.py -i testbed-new.yaml

This is run in /data/sonic-mgmt/ansible/

The input to this file is your testbed-yaml file

More details can be found here : <https://github.com/sonic-net/sonic-mgmt/blob/master/docs/testbed/README.new.testbed.Configuration.md>

A running output of the script should looks something like below.

<https://pb.infra.corp.arista.io/ZWFk>

1. **Run creategraph.py and make sure that your lab links file is generated**

This step creates the lab\_connection graph xml file which contains all the physical connections in xml format.

Got to /data/sonic-mgmt/ansible/files and run the below CLI

**CLI** : sudo python creategraph.py -d sonic\_lab\_devices.csv -l sonic\_lab\_links.csv -o lab\_connection\_graph.xml

The script should run without any errors

1. **Run the add topo step:**

This step deploys your VMs and pushes all the necessary configs onto the VMs.

CLI : /testbed-cli.sh -k ceos add-topo Ash-m0 password.txt -vvv

1. **Deploy Minigraph**

This step pushes the configs on your sonic dut

CLI : ./testbed-cli.sh deploy-mg Ash-m0 lab [password.txt](https://opengrok.infra.corp.arista.io/source/s?path=password.txt&project=eos-trunk) -vvv

Run under : /data/sonic-mgmt/ansible/

1. **Run tests .**

Directory: data/sonic-mgmt/tests/

Initially, try to run these basic tests and see if there are any known issues , run the below tests and note down their observations

**CLI**

1. sudo ./run\_tests.sh -n Ash-m0 -c test\_interfaces.py -t m0,util,any
2. sudo ./run\_tests.sh -n Ash-m0 -c test\_nbr\_health.py -t m0,util,any
3. sudo ./run\_tests.sh -n Ash-m0 -c tacacs/test\_ro\_user.py -t m0,util,any
4. sudo ./run\_tests.sh -n Ash-m0 -c acl/test\_acl.py::TestBasicAcl::test\_source\_ip\_match\_forwarded -t m0,util,any
5. sudo ./run\_tests.sh -n Ash-m0 -c snmp/test\_snmp\_interfaces.py -t m0,any
6. sudo ./run\_tests.sh -n Ash-m0 -c bgp/test\_bgp\_fact.py -t m0,any
7. sudo ./run\_tests.sh -n Ash-m0 -c bgp/test\_bgpmon.py -t m0,util,any
8. sudo ./run\_tests.sh -n Ash-m0 -c fib/test\_fib.py -t m0,util,any
9. sudo ./run\_tests.sh -n Ash-m0 -c lldp/test\_lldp.py -t m0,util,any

After which, you can run the full suite of tests with the below CLI command

**CLI** : sudo ./run\_tests.sh -n Ash-m0 -s "vrf/test\_vrf.py vrf/test\_vrf\_attr.py mvrf/test\_mgmtvrf.py platform\_tests/broadcom/test\_ser.py" -m individual -r -t any,util,m0,m0

**References:**

1. [sonic-mgmt-README](https://github.com/sonic-net/sonic-mgmt/blob/master/docs/testbed/README.testbed.Overview.md)
2. [TestBed Setup Details](https://github.com/sonic-net/sonic-mgmt/blob/master/docs/testbed/README.testbed.Setup.md)
3. [Deploying MiniGraph](https://github.com/sonic-net/sonic-mgmt/blob/master/docs/testbed/README.testbed.Minigraph.md)
4. [Developement Team Sonic Workflow](https://docs.google.com/document/d/1kbMcv73KXemQIIEVrcmSsHLgEB_kVqbi86i_o0iy6So/edit#heading=h.j4umob1gitc7)
5. [Runing Tests](https://github.com/sonic-net/sonic-mgmt/blob/master/docs/tests/pytest.run.md)
6. [Systest Issues While Deploying Sonic Mgmt setup](https://docs.google.com/spreadsheets/d/15RIyayujUXuP2mwzQujvkAwu7iiXVC8luSw0cUcwGy4/edit#gid=0)