

Build Sonic Test Environment

Michael Hung, Alvin Ko
August 10, 2018





Introduction

Setting up the sonic-mgmt testbed from Github to your own environment can be a tedious process. There are 10+ files that need to be updated before you can run test cases.

However, this process can be automated with:

- testbed-new.yaml - <https://github.com/delta46101/sonic-mgmt/blob/master/ansible/testbed-new.yaml>
- TestbedProcessing.py - <https://github.com/delta46101/sonic-mgmt/blob/master/ansible/TestbedProcessing.py>

The testbed-new.yaml file is a configuration file that compiles all the data needed to run the test cases into one file. TestbedProcess.py works by pulling information from that configuration file and pushing the data into the files where they belong. This guide will outline and facilitate the testbed set up of sonic-mgmt.



Objective

The objective of this guide is to outline and facilitate the process of using the testbed-new.yaml and TestbedProcessing.py files. At the end of this guide, you should be able to setup the sonic-mgmt testbed and run the testcases.

Information for basic set up can be referenced at [Sonic-Mgmt Testbed Setup](#).



sonic-mgmt github

GitHub link: <https://github.com/Azure/sonic-mgmt>

Azure / sonic-mgmt

Unwatch 48

Star 12

Fork 80

<> Code

Issues 61

Pull requests 27

Projects 0

Wiki

Insights

Configuration management examples for SONiC

526 commits

5 branches

0 releases

40 contributors

Branch: master

New pull request

Create new file

Upload files

Find file

Clone or download

sihuihan88 [pfcwd]: reduce the number of tested port each day (#676)

Latest commit 86a47ef 5 days ago

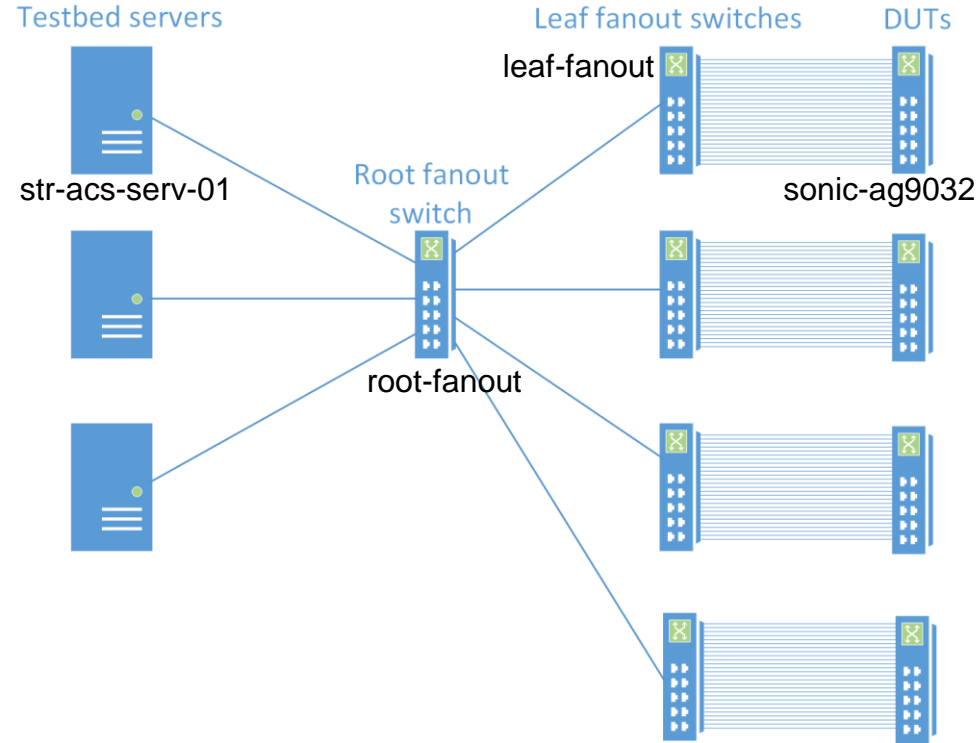
.github	Update PULL_REQUEST_TEMPLATE.md	13 days ago
ansible	[pfcwd]: reduce the number of tested port each day (#676)	5 days ago
.gitignore	[DHCP Relay test]: Update test to verify new Option 82 field contents (...)	a year ago
LICENSE	Adding readme and license	3 years ago
README.md	Update README.md	a month ago

Pre-migration Setup

There are (at minimum) several devices needed to get sonic-mgmt up and running with the test cases:

- Lab Server
- Root Fanout
- Leaf Fanout
- DUT (Device Under Test)

Information for testbed and topology can be referenced at [Sonic-Mgmt Testbed Overview](#)





Set Up Docker with sonic-mgmt



Install Docker on Ubuntu Server 16.04



Procedure

- Add the GPG key for the official Docker repository to the system
`curl -fsSL https://download.docker.com/linux/Ubuntu/gpg | sudo apt-key add -`
- Add the Docker repositories to APT sources:
`sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"`
- Update the package database with the Docker packages from the newly formed repo
`sudo apt-get update`
- Make sure you are about to install from the Docker repo instead of the default Ubuntu 16.04 repo:
`apt-cache policy docker-ce`

Source: <https://medium.com/@Grigorkh/how-to-install-docker-on-ubuntu-16-04-3f509070d29c>



Install Docker on Ubuntu Server 16.04



Procedure

- Finally, install Docker
`sudo apt-get install -y docker-ce`
- Check that it is running.
`Sudo systemctl status docker`

```
sonic@sonic1:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2018-07-26 10:50:50 PDT; 2 weeks 5 days ago
     Docs: https://docs.docker.com
    Main PID: 2253 (dockerd)
      Tasks: 106
     Memory: 1.6G
        CPU: 2h 2min 37.950s
    CGroup: /system.slice/docker.service
            └─ 621 docker-containerd-shim -namespace moby -workdir /var/lib/docker/containerd/daemon/io.containerd.runtime.v1
               914 docker-containerd-shim -namespace moby -workdir /var/lib/docker/containerd/daemon/io.containerd.runtime.v1
               2253 /usr/bin/dockerd -H fd://
               3330 docker-containerd --config /var/run/docker/containerd/containerd.toml
               3997 /usr/bin/docker-proxy -proto tcp -host-ip 0.0.0.0 -host-port 5000 -container-ip 172.17.0.2 -container-port
               4067 docker-containerd-shim -namespace moby -workdir /var/lib/docker/containerd/daemon/io.containerd.runtime.v1
               10440 docker-containerd-shim -namespace moby -workdir /var/lib/docker/containerd/daemon/io.containerd.runtime.v1
               15032 docker-containerd-shim -namespace moby -workdir /var/lib/docker/containerd/daemon/io.containerd.runtime.v1
               17980 docker-containerd-shim -namespace moby -workdir /var/lib/docker/containerd/daemon/io.containerd.runtime.v1

Aug 10 15:25:36 sonic1 dockerd[2253]: time="2018-08-10T15:25:36.699711344-07:00" level=info msg="Error logging in to v2 endpoint
Aug 10 15:25:36 sonic1 dockerd[2253]: time="2018-08-10T15:25:36.723574347-07:00" level=warning msg="Error getting v2 registry:
Aug 10 15:25:36 sonic1 dockerd[2253]: time="2018-08-10T15:25:36.723766568-07:00" level=info msg="Attempting next endpoint for
Aug 10 15:25:37 sonic1 dockerd[2253]: time="2018-08-10T15:25:37-07:00" level=info msg="shim docker-containerd-shim started" ad
Aug 14 10:48:12 sonic1 dockerd[2253]: time="2018-08-14T10:48:12-07:00" level=info msg="shim reaped" id=61b5c8ac60b32b0e3fa14b9
Aug 14 10:48:12 sonic1 dockerd[2253]: time="2018-08-14T10:48:12.359267382-07:00" level=info msg="ignoring event" module=libcon
Aug 14 10:48:33 sonic1 dockerd[2253]: time="2018-08-14T10:48:33.641930545-07:00" level=info msg="Error logging in to v2 endpoint
Aug 14 10:48:33 sonic1 dockerd[2253]: time="2018-08-14T10:48:33.661773409-07:00" level=info msg="ignoring event" module=libcon
Aug 14 10:48:33 sonic1 dockerd[2253]: time="2018-08-14T10:48:33.661840003-07:00" level=info msg="Attempting next endpoint for
Aug 14 10:48:34 sonic1 dockerd[2253]: time="2018-08-14T10:48:34-07:00" level=info msg="shim docker-containerd-shim started" ad
lines 1-29/29 (END)
```

Source: <https://medium.com/@Grigorkh/how-to-install-docker-on-ubuntu-16-04-3f509070d29c>



Create the Docker Registry



Follow these steps to create the Docker registry and load the docker-ptf image onto the local registry if you don't already have it set up:

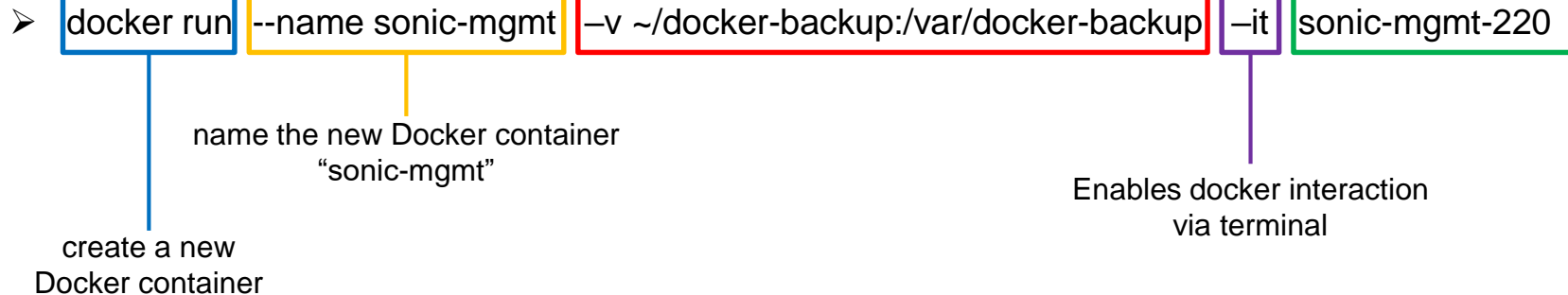
- `docker run -d -p 5000:5000 -v ~/docker-storage:/var/lib/registry --restart=always --name sonictest registry:2.6.2`
- `docker load -i ~/sonic-buildimage/docker-ptf.gz` (docker-ptf.gz built from <https://github.com/Azure/sonic-buildimage>)
- `docker tag docker-ptf localhost:5000/docker-ptf`
- `docker push localhost:5000/docker-ptf`

```
sonic@sonic1:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
sonic-mgmt           Michael             3862a4bf912d       13 days ago        1.46GB
sonic-mgmt-220       latest             8e8219228c99       2 weeks ago        1.45GB
registry             2.6.2              b2b03e9146e1       5 weeks ago        33.3MB
jenkins/sonic        latest             804870bc8b6d       6 weeks ago        1.61GB
localhost:5000/docker-ptf latest            c269dca80dcb       5 months ago        617MB
docker-ptf           latest            c269dca80dcb       5 months ago        617MB
docker-sonic-mgmt    latest            b29924a45f98       5 months ago        1.43GB
sonic@sonic1:~$
```

Start a Docker Container

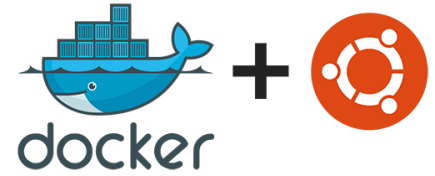


From here on out, you will need to run everything within a Docker container.
Set up a Docker container with the following command:

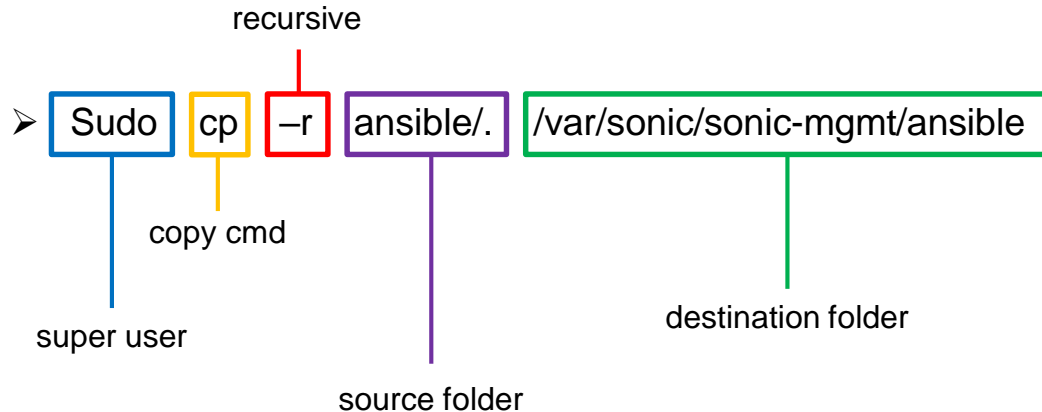




Copy Github sonic-mgmt to Docker sonic-mgmt



- Download the sonic-mgmt Github repo to your host machine (either download and upload or git clone)
- Move the repo on your host machine (~/.docker-backup/latest/) to the shared drive (/var/docker-backup)
- Move the repo from the shared drive to /var/sonic-mgmt/ansible



Source: https://www.docker.com/sites/default/files/Docker_CheatSheet_08.09.2016_0.pdf



Transfer Script and Config file to /var/sonic/sonic-mgmt/ansible

Transfer the script and config file to /var/sonic/sonic-mgmt/ansible using the copy command

- `sudo cp ansible/TestbedProcessing.py /var/sonic/sonic-mgmt/ansible`
- `sudo cp ansible/testbed-new.yaml /var/sonic/sonic-mgmt/ansible`

```
sonic@sonic1: ~/docker-backup/latest/sonic-mgmt/ansible$ ls
ansible.cfg      library
basic_check.yml  linkstate
boot_onie.yml    minigraph
config_sonic_basedon_testbed.yml  ocp
deploy_sonic.yml  plugins
doc              README.deploy.md
eos.yml          README.md
fanout_connect.yml  README.testbed.md
fanout.yml       README.test.md
files            roles
group_vars       shell_plugins
host_vars        swap_syncd.yml
inventory        templates
lab              testbed_add_vm_topology.yml
sonic@sonic1:~/docker-backup/latest/sonic-mgmt/ansible$

testbed-cli.sh
testbed_connect_vms.yml
testbed.csv
testbed_disconnect_vms.yml
testbed_remove_vm_topology.yml
testbed_renumber_vm_topology.yml
testbed_start_VMs.yml
testbed_stop_VMs.yml
test_sonic.yml
upgrade_sonic.yml
vars
veos
veos.yml
```

```
sonic@7cabaf8d4208: ~/sonic-mgmt/ansible$ ls
README.deploy.md  helloWorld.txt  testbed-alvin-v3.yaml
README.md         host_vars       testbed-cli.sh
README.test.md    inventory       testbed-connect_vms.yml
README.testbed.md  lab            testbed-add_vm_topology.yml
TestbedProcessing.py  latestPull.txt  testbed_disconnect_vms.yml
ansible.cfg        library        testbed_remove_vm_topology.yml
backup             linkstate      testbed_renumber_vm_topology.yml
basic_check.yml    minigraph      testbed_start_VMs.yml
boot_onie.yml      ocp            testbed_stop_VMs.yml
config_sonic_basedon_testbed.yml  password.txt    upgrade_sonic.yml
deploy_sonic.yml   plugins        vars
doc               roles          vars\docker_registry.yml
eos.yml           shell_plugins  veos
fanout.yml        sonic-buildimage
fanout_connect.yml  swap_syncd.yml  veos.yml
files             templates
group_vars        test_sonic.yml
sonic@7cabaf8d4208:~/sonic-mgmt/ansible$
```

Host machine
(without TestbedProcessing.py and testbed-new.yaml)

Docker Container
(with TestbedProcessing.py and testbed-new.yaml)



Working Docker Environment



Currently, there is a Docker image that contains a working environment. It can be loaded into an new environment. This image should be able to run test cases after editing the testbed-new.yaml, running TestbedProcessing.py, and checking credentials in labinfo.json.

Location: local Docker registry. Available upon request.

```
sonic@sonic1:~$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
sonic-mgmt	Michael	3862a4bf912d	13 days ago	1.46GB
sonic-mgmt-220	latest	8e8219228c99	2 weeks ago	1.45GB
registry	2.6.2	b2b03e9146e1	5 weeks ago	33.3MB
jenkins/sonic	latest	804870bc8b6d	6 weeks ago	1.61GB
localhost:5000/docker-ptf	latest	c269dca80dcb	5 months ago	617MB
docker-ptf	latest	c269dca80dcb	5 months ago	617MB
docker-sonic-mgmt	latest	b29924a45f98	5 months ago	1.43GB

```
sonic@sonic1:~$
```



Ready to Run Test Cases?

Limitation:

- The default root/leaf fanout switches are Arista switches. The test scripts are running on it.
- We have our own switches loading ICOS as fanout switches. **The scripts can't run on it.**

Solution:

1. Rewrite the scripts to bypass the configuration on fanout switch and configure it manually.
2. Files to update:
 - sonic-mgmt/ansible/group_vars/all/labinfo.json
 - sonic-mgmt/ansible/plugins/connection/switch.py
 - Add a new template at sonic-mgmt/ansible/roles/fanout/templates/icos_connect.j2
 - sonic-mgmt/ansible/roles/fanout/tasks/rootfanout_connect.yml
 - sonic-mgmt/ansible/roles/test/tasks/interface.yml
 - sonic-mgmt/ansible/roles/test/tasks/link_flap/link_flap_helper.yml
 - Add sonic-mgmt/ansible/roles/test/templates/neighbor_interface_no_shut_dni.j2



Quick Start

Step 1:

Option1 : Download the working docker image to create your container.

Option2 : copy working sonic-mgmt directory to your container.

Step 2:

Modify the testbed configuration file.

Run the script to push all configurations.

Step 3:

Configure the port between root fanout switch and test server as trunk mode.

Configure the port between root and leaf fanout switch as trunk mode.

Step 4:

Configure the port between leaf fanout switch and DUT as access mode.

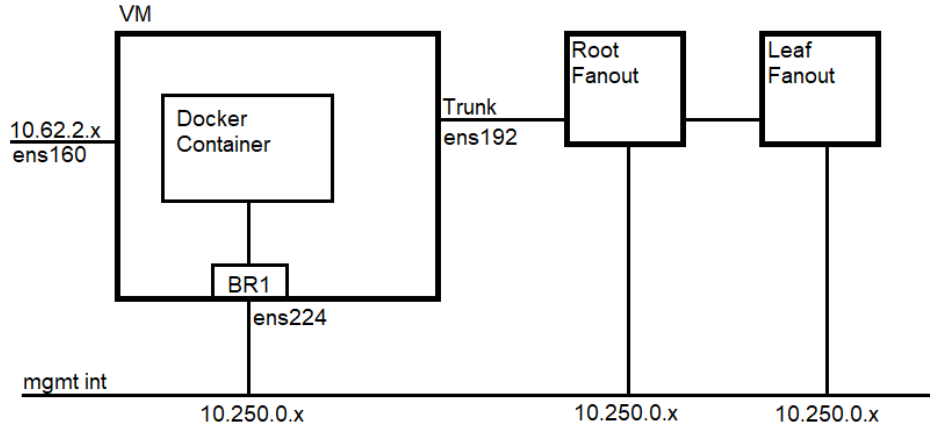
Set vlan id and bandwidth based on sonic-mgmt/ansible/files/sonic_lab_links.csv .



Set Up Management Port Configuration



Testbed Server



- vm mgmt int: ens160
- sonic docker mgmt int: br1/ ens224
- interface to root fanout: ens192

Warning: the lab server (str-ac-serv-01) mgmt_bridge is the management port interface (br1 and ens224). The external_iface is actually the trunk to the switches. This was a source of confusion when modifying the testbed.

```
leaf-fanout:
  interfaces:
    0/25:
      EndDevice: str-ac-serv-01
      EndPort: ens192
      Bandwidth: 40000
      VlanID: 1781-1812
      VlanMode: Trunk
```

```
host_vars:
  str-ac-serv-01:
    mgmt_bridge: br1
    mgmt_prefixlen: 24
    mgmt_gw: 10.250.0.1
    external_iface: ens192
```



Modify /etc/network/interfaces

To edit the /etc/network/interfaces file:

- `sudo nano /etc/network/interfaces`
- Look for the line in the file that says “# The primary network interface” and directly beneath it, you’ll see the default DHCP configuration:
auto ens160
iface ens160 inet dhcp
- Comment out “iface ens160 inet dhcp”
- Replace with the following information below:
iface ens160 inet static
 address <address>
 netmask <netmask>
 gateway <gateway>
 dns-nameservers <nameserver>

```
sonic@sonic1:~$ cat /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

#source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto ens160
#iface ens160 inet dhcp
iface ens160 inet static
    address 10.62.2.220
    netmask 255.255.255.0
    gateway 10.62.2.254
    dns-nameservers 10.62.2.1

# docker mgmt interface
auto ens224
iface ens224 inet manual

#
auto ens192
iface ens192 inet manual

auto br1
iface br1 inet static
    bridge_ports ens224
    bridge_stp off
    bridge_maxwait 0
    bridge_fd 0
    address 10.250.0.201
    netmask 255.255.255.0
    network 10.250.0.0
    #gateway 10.250.0.1
    #dns-nameservers 10.250.0.1 10.250.0.2

sonic@sonic1:~$
```



Modify /etc/network/interfaces

Continue editing the /etc/network/interfaces file:

Specify how to communicate with docker

- Identify the Docker interface to the mgmt interface:
auto ens224
iface ens224 inet manual
- Define br1, its members, and additional relevant detail:
auto br1
iface br1 inet static
 bridge_ports ens224
 bridge_stp off
 bridge_maxwait 0
 address <address>
 netmask <netmask>
 network <network>

```
sonic@sonic1:~$ cat /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

#source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto ens160
#iface ens160 inet dhcp
iface ens160 inet static
    address 10.62.2.220
    netmask 255.255.255.0
    gateway 10.62.2.254
    dns-nameservers 10.62.2.1

# docker mgmt interface
auto ens224
iface ens224 inet manual

#
auto ens192
iface ens192 inet manual

auto br1
iface br1 inet static
    bridge_ports ens224
    bridge_stp off
    bridge_maxwait 0
    bridge_fd 0
    address 10.250.0.201
    netmask 255.255.255.0
    network 10.250.0.0
    #gateway 10.250.0.1
    #dns-nameservers 10.250.0.1 10.250.0.2

sonic@sonic1:~$
```



Modify /etc/network/interfaces

Continue editing the /etc/network/interfaces file:

Finally, declare the trunk connection from the testbed server to the root fanout switch:

- auto ens192
 iface ens192 inet manual

When you are done editing the interfaces file:

- Save the file
- Restart the network or reboot

```
sonic@sonic1:~$ cat /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

#source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto ens160
#iface ens160 inet dhcp
iface ens160 inet static
    address 10.62.2.220
    netmask 255.255.255.0
    gateway 10.62.2.254
    dns-nameservers 10.62.2.1

# docker mgmt interface
auto ens224
iface ens224 inet manual

#
auto ens192
iface ens192 inet manual

auto br1
iface br1 inet static
    bridge_ports ens224
    bridge_stp off
    bridge_maxwait 0
    bridge_fd 0
    address 10.250.0.201
    netmask 255.255.255.0
    network 10.250.0.0
    #gateway 10.250.0.1
    #dns-nameservers 10.250.0.1 10.250.0.2

sonic@sonic1:~$
```



VLAN Configuration



VLAN Configuration (General)

Each switch running ICOS needs to have these components:

- vlan database (with range from 2 through x)
- IP routing
- line console
- line telnet
- line ssh
- router ospf (both ipv4 and ipv6)

```
router ospf
exit
ipv6 router ospf
exit
```

```
vlan database
vlan 2-2000
exit
```

```
configure
ip routing
```

```
line console
exit
```

```
line telnet
exit
```

```
line ssh
exit
```

```
!
```



VLAN Configuration (Root Fanout)

The root fanout primarily serves as a trunk between the testbed server and the leaf fanout. Therefore, it needs the following components:

- “no spanning-tree”
- Interface declaration

The number of interfaces that you want to declare as trunks depends the setup. In this guide, interface 0/1 to 0/5 and interfaces 0/50 to 0/52 are declared as trunks.

```
no spanning-tree
!  
interface 0/1  
no shutdown  
switchport mode trunk  
exit  
  
interface 0/2  
no shutdown  
switchport mode trunk  
exit
```



VLAN Configuration (Leaf Fanout)

The leaf fanout serves as access ports to the DUT. However, it also connects to the root fanout. Therefore, it is important to declare interfaces for both switchport mode access and switchport mode trunk.

The number of interfaces that you bring up as access ports depends on the number of cables that you have. If you have a 100G cable at interface 0/1 on the leaf fanout that connects to the DUT, you must bring interface 0/1 as switchport mode access and declare the vlan you will be using (see image to the right).

```
interface 0/7
no fec
no shutdown
switchport mode access
switchport access vlan 1787
exit
```

```
interface 0/8
no shutdown
switchport mode access
switchport access vlan 1788
exit
```

```
interface 0/25
speed 40G full-duplex
no shutdown
switchport mode trunk
exit
```




Install PyYAML (3.12)

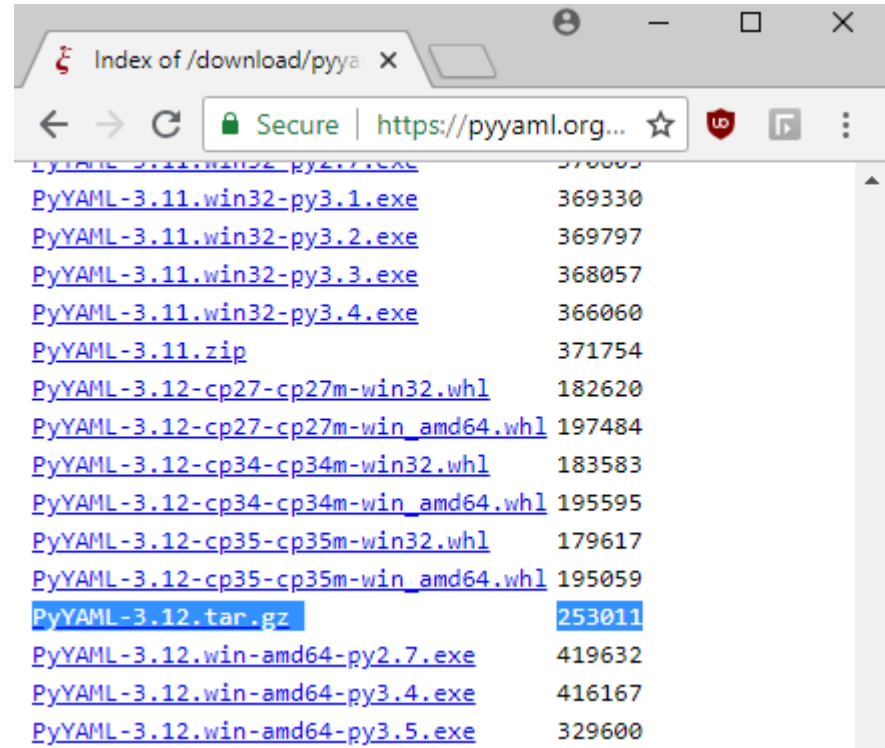
TestbedProcessing.py requires PyYAML.
PyYAML is a YAML parser and emitter for Python

Installation:

- To install from source, download the source package (we used PyYAML 3.12) and unpack it.
- Go to the directory, PyYAML-3.12
- Run “python setup.py install” to install
- Run “python setup.py test” to test for correct installation

References:

- <https://pyyaml.org/download/pyyaml/>
- <https://pyyaml.org/wiki/PyYAMLDocumentation>





Modify the (testbed-new.yaml) Configuration File



testbed-new.yaml sections

There are 8 main sections in testbed-new.yaml that need to be edited:

1. device_groups
2. devices
3. host_vars
4. veos_groups
5. veos
6. testbed
7. topology
8. docker_registry

Each of these sections above contribute to the files that need to be written into in order for the test cases to run. For more information about what each file does, please reference [Sonic-Mgmt Testbed Configuration](#).



Files to Update

Base Directory: sonic/ansible/

```
sonic@7cabaf8d4208:~/sonic-mgmt/ansible$ ls
README.deploy.md      ocp
README.md             password.txt
README.test.md        plugins
README.testbed.md     roles
TestbedProcessing.py  shell_plugins
ansible.cfg           sonic-buildimage
backup               swap_syncd.yml
basic_check.yml       templates
boot_onie.yml         test_sonic.yml
config_sonic_basedon_testbed.yml testbed-alvin-v3.yml
deploy_sonic.yml      testbed-cli.sh
doc                  testbed.csv
eos.yml              testbed_add_vm_topology.yml
fanout.yml           testbed_connect_vms.yml
fanout_connect.yml   testbed_disconnect_vms.yml
files                testbed_remove_vm_topology.yml
group_vars            testbed_renumber_vm_topology.yml
helloWorld.txt        testbed_start_VMs.yml
host_vars             testbed_stop_VMs.yml
inventory             upgrade_sonic.yml
lab                  vars
latestPull.txt        vars\docker_registry.yml
library              veos
linkstate             veos.yml
minigraph
```

Here is a list of files that will get updated:

- group_vars/vm_host/main.yml
- group_vars/vm_host/creds.yml
- files/sonic_lab_devices.csv
- testbed.csv
- files/sonic_lab_links.csv
- group_vars/eos/creds.yml
- group_vars/fanout/secrets.yml
- group_vars/lab/secrets.yml
- lab
- veos
- host variables (0 or more files will be generated)
- vars/docker_registry.yml



(OPTIONAL) testbed_config section

- Name – choose a name for this testbed config file
- Alias – choose an alias for this testbed config file

```
testbed_config:  
  name: sonic-mgmt-220 Testbed  
  alias: topologyDeltaTestbed  
  type: Physical # Physical or Virtual
```



device_groups section

USAGE: lab

The device_groups section generates the lab file which is the inventory file necessary for setting up the testbed. While the format in the configuration file is in yaml format, the script converts it to INI format. The device groups section includes all lab DUTs, fanout switches, and testbed server topologies. Group children are referenced from the devices section below. For the most part this section can be left alone.

```
device_groups:
  lab:
    children: [sonic-ag9032, fanout]
  sonic:
    children: [sonic-ag9032]
  sonic-ag9032:
    host: [sonic-ag9032]
    vars:
      hwsku: "Delta-ag9032v1"
      iface_speeds: "100000"
  fanout:
    host: [leaf-fanout]
    vars:
      hwsku: "ICOS"
      mgmt_subnet_mask_length: "24"
  ptf:
    host: [ptf32-1]
```



devices section

USAGE: `files/sonic_lab_devices`, `group_vars/fanout/secrets`, `group_vars/lab/secrets`, `lab`

The devices section is a dictionary that contains all devices and hosts. This section does not contain information on PTF containers. For more information on PTF containers, see the `testbed.csv` file.

The lab server section requires different fields to be entered:

- `ansible_become_pass`
- `sonicadmin_user`
- `sonicadmin_password`
- `sonicadmin_initial_password`

`Sonicadmin_user` is still just the username. The other fields is the password. These fields were selected because they are variables taken directly `group_var/lab/secrets.yml`. So for convenience, this section of the config file takes a copy of the variable labels.



devices section

For each device that you add, add the following (example given below):

hostname	ansible_ host	ansible_ ssh_user	ansible_ ssh_pass	hwSKU	device_type
str-ac-serv-01	10.250.0.201/24	sonic	[password]	TestServ	Server
leaf-fanout	10.250.0.36/24	admin	broadcom	ICOS	FanoutLeaf
root-fanout	10.250.0.47/24	admin	broadcom	ICOS	FanoutRoot
sonic-ag9032	10.250.0.45/24	admin	YourPaSsWoRd	Delta-ag9032v1	DevSonic



devices section

Header descriptions:

- hostname - names the devices you will use
- ansible_host - this is the management IP where you can connect to the device
- ansible_ssh_user - this is your username to login to the device
- ansible_ssh_pass - this is your password to login to the device
- hwsku - this is the look up value for credentials in /group_vars/all/labinfo.json. Without this section, things will fail. Make sure this field is filled out and verify labinfo.json is accurate.
- device type - the type of device. If you only have 4 devices, you can leave the provided labels alone

```
devices:
  sonic-ag9032:
    device_type: DevSonic
    hwsku: Delta-ag9032v1
    alias:
    credentials:
      username:
      password:
      enable_password:
    ansible:
      ansible_host: 10.250.0.45/24
      ansible_ssh_user: admin
      ansible_ssh_pass: YourPaSsWoRd
```



host_vars section

USAGE: all host_var values

Define the host variables here. In this guide, we define the lab server (str-acis-serv-01) here.

For each host device that you add; define or verify the following:

- mgmt_bridge
- mgmt_prefixlen (this should match with the mgmt_subnet_mask_length)
- mgmt_gw
- external_iface

```
host_vars:
  str-acis-serv-01:
    mgmt_bridge: br1
    mgmt_prefixlen: 24
    mgmt_gw: 10.250.0.1
    external_iface: ens192
```



veos_groups section

USAGE: veos

While the lab file is an ansible inventory file that includes all lab DUTs, fanout switches, and testbed server topologies, the veos file is an ansible inventory file that contains all servers and VMs.

```
veos_groups:
  vm_host:
    children: [vm_host_1] # source: sonic-mgmt/veos
  vm_host_1:
    host: [STR-ACS-SERV-01] # source: sonic-mgmt/veos
  vms_1:
    host: [VM0200, VM0201, VM0202, VM0203] # source: sonic-mgmt/veos
  eos:
    children: [vms_1] # source: sonic-mgmt/veos
  server_1:
    children: [vm_host_1, vms_1] # source: sonic-mgmt/veos
    vars:
      host_var_file: host_vars/STR-ACS-SERV-01.yml # source: sonic-mgmt/veos
  servers:
    children: [server_1]
    vars:
      topologies:
        ['t1', 't1-lag', 't1-64-lag', 't0', 't0-52', 'ptf32', 'ptf64', 't0-64',
```



veos section

USAGE: `group_vars/eos/creds, main.yml,`
`group_vars/vm_host/creds`

Like the `veos_groups` section, this section contains information about the servers and VMs within your testbed. There are two sets of tasks to perform.

```
veos:
  credentials:
    username:
    password:
  root_path: /home/sonic/sonic-mgmt/ansible/veos-vm/
  vm_images_url:
  cd_image_filename: Aboot-veos-serial-8.0.0.iso
  hdd_image_filename: vEOS-lab-4.15.10M.vmdk
  skip_image_downloading: true
  vm_console_base: 7000
  memory: 2097152
  max_fp_num: 4
  proxy_env:
    http_proxy:
    https_proxy:
  vm_host_ansible:
    ansible_user: sonic
    ansible_password:
    ansible_sudo_pass:
  eos_ansible:
    ansible_user: root
    ansible_password: 123456
  vms_1:
    VM0200:
      ansible_host: 10.250.0.50
    VM0201:
      ansible_host: 10.250.0.51
```



veos section

Confirm the following:

- root_path - server's root path to building the VMs
- cd_image_filename - you should be able to locate "Aboot-veos-serial-8.0.0.iso"
- hdd_image_file: you should also be able to locate "vEOS-lab-4.15.10M.vmdk"

Define:

- vm_console_base - if you are running multiple sets of sonic-mgmt VMs, define a conflict-free vm_console_base
- ansible_user - username to access VM
- ansible_password - password to access VM
- ansible_sudo_pass - same as password above
- vms_1
 - define the VMs that you want to bring up (i.e.: VM0200, VM0201, VM0202, etc...)
 - define the IPs of the VMs (i.e." 10.250.1.0, 10.250.1.1, 10.250.1.2, etc...)



testbed section

This is where the topology configuration file for the testbed will collect information from when running TestbedProcessing.py

#conf-name	group-name	topo	ptf_image_name	ptf_ip	server	vm_base	dut	comment
ptf32-2	ptf32-2	ptf32	docker-ptf	10.250.0.191/24	server_1	VM0200	sonic-ag9032	Test ptf
t0-1	t0-1	t0	docker-ptf	10.250.0.192/24	server_1	VM0200	sonic-ag9032	Test ptf



testbed section

For each topology you use in your testbed environment, define the following:

- conf-name - to address row in table
- group-name - used in interface names, up to 8 characters. The variable can be anything but should be identifiable.
- topo - name of topology
- ptf_image_name - defines PTF image. In this guide, the docker-ptf was an image already on the local registry.

If you don't have docker-ptf from the sonic-buildimage github, follow the steps below:

- git clone --recursive
<https://github.com/Azure/sonic-buildimage.git>

- make configure PLATFORM=generic

- make target/docker-ptf.gz

```
testbed:
  ptf32-2:
    group-name: ptf32-2
    topo: ptf32
    ptf_image_name: docker-ptf
    ptf_ip: 10.250.0.191/24
    server: server_1
    vm_base: VM0200
    dut: sonic-ag9032
    comment: Test ptf
    ansible:
      ansible_host: 10.250.0.191/24
      ansible_ssh_user: root
      ansible_ssh_pass: root
```



testbed section

For each topology you use in your testbed environment, define the following (continued):

- ptf_ip - ip address for mgmt interface of PTF container. Choose an IP address that is available
- server - server where the testbed resides. Choose a veos_group to use that contains both the lab server and virtual machines
- vm_base - enter in the lowest ID value for the VMs you will be using to run the test cases. The lowest VM ID value can be found under the veos section of the testbed configuration file. IF empty, no VMs are used
- dut - enter in the target DUT that is used in the testbed environment
- comment - make a little note here
- ansible
 - ansible_host - IP address with port number
 - ansible_ssh_user - username to login to lab server
 - ansible_ssh_pass - password to login to lab server



topology section

USAGE: `files/sonic_lab_links.csv`

This section of the testbed configuration file defines the connection between the DUT to the leaf-fanout and the leaf-fanout to the lab server.

Static values: the bandwidth remains at 100000. VlanMode from the DUT to the leaf-fanout is always "Access." However, VlanMode from the leaf-fanout to the lab server is "Trunk".



topology section

From the DUT to the leaf-fanout, make sure to define:

- end ports – which DUT port is connected to which leaf-fanout (end) port
- vlanIDs - this is very important because the VlanIDs will bind to the ports

From the leaf-fanout to the server, make sure to define:

- lab device - the testbed server you will be using (probably str-ac-serv-01)
- endport - find this in ifconfig on your testbed server
- vlanID (range) - if you have 32 ports, the range is from the lowest VlanID you defined +31, totaling 32 ports



topology section

From leaf-fanout to DUT (sonic-ag9032)

```
sonic-ag9032:
  interfaces:
    Ethernet0:
      EndDevice: leaf-fanout
      EndPort: 0/1
      Bandwidth: 100000
      VlanID: 1781
      VlanMode: Access
    Ethernet4:
      EndDevice: leaf-fanout
      EndPort: 0/2
      Bandwidth: 100000
      VlanID: 1782
      VlanMode: Access
    Ethernet8:
      EndDevice: leaf-fanout
      EndPort: 0/3
      Bandwidth: 100000
      VlanID: 1783
      VlanMode: Access
    Ethernet12:
      EndDevice: leaf-fanout
      EndPort: 0/4
      Bandwidth: 100000
      VlanID: 1784
      VlanMode: Access
```

From leaf-fanout to str-acsv-serv-01 (lab server)

```
leaf-fanout:
  interfaces:
    0/25:
      EndDevice: str-acsv-serv-01
      EndPort: ens192
      Bandwidth: 40000
      VlanID: 1781-1812
      VlanMode: Trunk
```



Update docker_registry.yml

TestbedProcessing.py has a function called updateDockerRegistry() that will perform this task for you.

If the docker_registry.yml hasn't been updated already, the following data will be written into it:

- docker_registry_host: localhost:5000 <= ansible will load docker-ptf image from here
- docker_registry_username: root
- docker_registry_password: root

```
docker_registry:  
  docker_registry_host: localhost:5000  
  docker_registry_username: root  
  docker_registry_password: root
```



docker_registry section

USAGE: vars/docker_registry.yml

The docker registry contains 3 pieces of information:

1. docker_registry_host
2. docker_registry_username
3. docker_registry_password

If you already have this information, you can choose to leave it blank and the script will skip this section.

```
docker_registry_host: localhost:5000  
  
docker_registry_username: root  
docker_registry_password: root
```



Running Test Cases



Run TestbedProcessing.py Script

Run the following command to automate the testbed configuration process:

➤ **python TestbedProcessing.py -i testbed-new.yaml**

Options:

- i = the testbed-new.yaml file to parse
- basedir = the basedir for the project
- backup = the backup directory for the files

TestbedProcessing.py conveniently backs up files that it will write into.

Backup directory location (within container):
/sonic-mgmt/ansible/backups/<timestamp>

```
sonic@df471490d97d:~/sonic-mgmt/ansible$ python TestbedProcessing.py -i testbed-alvin.yaml
PROCESS STARTED
BACKUP PROCESS STARTED
Error: could not backup vars/docker_registry.yml
BACKUP PROCESS COMPLETED
LOADING PROCESS STARTED
LOADING: testbed-alvin.yaml
LOADING PROCESS COMPLETED
GENERATING FILES FROM CONFIG FILE
  CREATING SONIC LAB LINKS: files/sonic_lab_links.csv
  CREATING SONIC LAB DEVICES: files/sonic_lab_devices.csv
  CREATING TEST BED: testbed.csv
  CREATING VM_HOST/CREDS: group_vars/vm_host/creds.yml
  CREATING EOS/CREDS: group_vars/eos/creds.yml
  CREATING FANOUT/SECRETS: group_vars/fanout/secrets.yml
  CREATING LAB SECRETS: group_vars/lab/secrets.yml
  CREATING MAIN.YML: group_vars/vm_host/main.yml
  CREATING LAB FILE: lab
  CREATING VEOS FILE: veos
  CREATING HOST VARS FILE(S): one or more files generated
UPDATING FILES FROM CONFIG FILE
  UPDATING DOCKER REGISTRY
PROCESS COMPLETED
sonic@df471490d97d:~/sonic-mgmt/ansible$
sonic@df471490d97d:~/sonic-mgmt/ansible$ |
```



Create lab_connect_graph.xml

The ansible/files/creategraph.py file is a helper script that helps you generate a lab_connection_graph.xml based on ansible/files/sonic_lab_devices.csv and ansible/files/sonic_lab_links.csv

➤ **python creategraph.py -o lab_connection_graph.xml**

The lab_connection_graph.xml is the lab graph file for library/conn_graph_facts.py to parse and get all the lab fanout switch connections information.

```
sonic@7cabaf8d4208:~/sonic-mgmt/ansible/files$ ls
creategraph.py          lab_connection_graph.xml.bak2  sonic_lab_devices.csv.old
lab_connection_graph.xml sonic_lab_devices.csv          sonic_lab_links.csv
lab_connection_graph.xml.bak sonic_lab_devices.csv.bak      sonic_lab_links.csv.bak
sonic@7cabaf8d4208:~/sonic-mgmt/ansible/files$
```




VMS Commands

Start VMS (using vms_1):

➤ **`./testbed-cli.sh start-vms server_1 password.txt`**

Stop VMS (using vms_1):

➤ **`./testbed-cli.sh stop-vms server_1 password.txt`**

Once the VMs have been successfully created, the VMs should be reachable on both the Docker container and the host machine.

```
sonic@7cabaf8d4208:~/sonic-mgmt/ansible$ ping 10.250.0.50
PING 10.250.0.50 (10.250.0.50) 56(84) bytes of data.
64 bytes from 10.250.0.50: icmp_seq=1 ttl=63 time=0.594 ms
64 bytes from 10.250.0.50: icmp_seq=2 ttl=63 time=0.674 ms
64 bytes from 10.250.0.50: icmp_seq=3 ttl=63 time=0.626 ms
^C
--- 10.250.0.50 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2026ms
rtt min/avg/max/mdev = 0.594/0.631/0.674/0.038 ms
sonic@7cabaf8d4208:~/sonic-mgmt/ansible$ ping 10.250.0.51
PING 10.250.0.51 (10.250.0.51) 56(84) bytes of data.
64 bytes from 10.250.0.51: icmp_seq=1 ttl=63 time=0.748 ms
64 bytes from 10.250.0.51: icmp_seq=2 ttl=63 time=0.439 ms
64 bytes from 10.250.0.51: icmp_seq=3 ttl=63 time=0.566 ms
^C
--- 10.250.0.51 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2046ms
rtt min/avg/max/mdev = 0.439/0.584/0.748/0.128 ms
sonic@7cabaf8d4208:~/sonic-mgmt/ansible$ ping 10.250.0.52
PING 10.250.0.52 (10.250.0.52) 56(84) bytes of data.
64 bytes from 10.250.0.52: icmp_seq=1 ttl=63 time=0.666 ms
64 bytes from 10.250.0.52: icmp_seq=2 ttl=63 time=0.564 ms
64 bytes from 10.250.0.52: icmp_seq=3 ttl=63 time=0.538 ms
^C
--- 10.250.0.52 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2026ms
rtt min/avg/max/mdev = 0.538/0.589/0.666/0.058 ms
sonic@7cabaf8d4208:~/sonic-mgmt/ansible$ ping 10.250.0.53
PING 10.250.0.53 (10.250.0.53) 56(84) bytes of data.
64 bytes from 10.250.0.53: icmp_seq=1 ttl=63 time=0.659 ms
64 bytes from 10.250.0.53: icmp_seq=2 ttl=63 time=0.558 ms
64 bytes from 10.250.0.53: icmp_seq=3 ttl=63 time=0.556 ms
^C
--- 10.250.0.53 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2037ms
rtt min/avg/max/mdev = 0.556/0.591/0.659/0.048 ms
sonic@7cabaf8d4208:~/sonic-mgmt/ansible$
```



Deploy Topology Container (PTF32)

In this guide, ptf32-1 will be added using the testbed-cli.sh script as an example. However, other topologies can be added as well.

Remove topology ptf32-1:

➤ **`./testbed-cli.sh remove-topo ptf32-1 password.txt`**

Add topology ptf32-1:

➤ **`./testbed-cli.sh add-topo ptf32-1 password.txt`**

You can check to see if it was removed or added using the "docker ps" or "docker container ls"

```
sonic@sonic1: ~  
sonic@sonic1:~$ docker ps  
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS        NAMES  
a6cad2e9a033   localhost:5000/docker-ptf         "/usr/local/bin/supe..." 40 minutes ago Up 40 minutes      ptf_ptf32-2  
8e7872a4ce0d   localhost:5000/docker-ptf         "/usr/local/bin/supe..." 4 days ago    Up 4 days      ptf_t0-16-1  
bdbcb3feae70   sonic-mgmt:Michael                "/bin/bash"             11 days ago   Up 11 days      22/tcp       sonic-mgmt-mike  
3a216ad84ead   sonic-mgmt-220                    "/bin/bash"             2 weeks ago   Up 2 weeks      22/tcp       sonic-mgmt  
7cabaf8d4208   docker-sonic-mgmt                 "/bin/bash"             2 weeks ago   Up 11 days      22/tcp       sonic-mgmt-220  
63385c1e6cce   registry:2.6.2                    "/entrypoint.sh /etc..." 4 weeks ago   Up 2 weeks      0.0.0.0:5000->5000/tcp sonictest  
sonic@sonic1:~$
```



Run the following commands:

- [illegible]



Troubleshooting

Issue: Trying to update some libraries and packages but pip (version 10) throw “import error: cannot import name main”

Resolution: The ideal solution is to uninstall pip 10. Use “**python –m pip uninstall pip**” to uninstall the latest pip version but retain the patched pip version 8

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
libmysqlclient-dev is already the newest version (5.7.21-0ubuntu0.16.04.1).
python-dev is already the newest version (2.7.12-1~16.04).
0 upgraded, 0 newly installed, 0 to remove and 14 not upgraded.
ubuntu@ip-172-31-18-143:~$ sudo pip install MySQL-python
Traceback (most recent call last):
  File "/usr/bin/pip", line 9, in <module>
    from pip import main
ImportError: cannot import name main
ubuntu@ip-172-31-18-143:~$ ^C
ubuntu@ip-172-31-18-143:~$
```

Source: <https://stackoverflow.com/questions/49964093/file-usr-bin-pip-line-9-in-module-from-pip>



Troubleshooting

Issue: Trying to run start-vms CLI command but receiving error “No hosts matched”

Resolution: Check to see if the server group you are bringing up actually contains a host device (str-acs-serv-01). You can check this in the sonic-mgmt/ansible/veos file

```
sonic@7cabaf8d4208:~/sonic-mgmt/ansible$ cat veos
[eos:children]
vms_1

[vm_host_1]
STR-ACS-SERV-01 ansible_host=10.250.0.201

[servers:children]
server_1

[servers:vars]
topologies=[t1, t1-lag, t1-64-lag, t0, t0-52, ptf32, ptf64, t0-64, t0-64-32, t0-116]

[vm_host:children]
vm_host_1

[vms_1]
vM0200 ansible_host=10.250.0.50
vM0201 ansible_host=10.250.0.51
vM0202 ansible_host=10.250.0.52
vM0203 ansible_host=10.250.0.53

[server_1:children]
vm_host_1
vms_1

[server_1:vars]
host_var_file=host_vars/STR-ACS-SERV-01.yml

sonic@7cabaf8d4208:~/sonic-mgmt/ansible$
```



Troubleshooting

Issue: Testbed Command Line complains there is no password file available

Resolution: You can bypass this issue by creating an empty password file. CLI should be able to run afterwards

```
sonic@7cabaf8d4208:~/sonic-mgmt/ansible$ ./testbed-cli.sh remove-topo ptf32-2 password.txt
Removing topology 'ptf32-2'
reading
Found topology ptf32-2
ERROR! The vault password file /var/sonic/sonic-mgmt/ansible/password.txt was not found
```



Troubleshooting

Issue: Fails on task [Read dut minigraph]. Error message: “Can not find lab graph file lab_connection_graph.xml”

Resolution: To generate the lab_connection_graph.xml:

- Within your docker container, cd into /sonic-mgmt/ansible/files
- Run the following command: “python creategraph.py -o lab_connection_graph.xml”

```
TASK [Read dut minigraph] *****
Tuesday 14 August 2018 17:46:47 +0000 (0:00:00.057)    0:00:00.400 *****
fatal: [STR-ACS-SERV-01]: FAILED! => {"changed": false, "failed": true, "msg": "Can not find lab graph file lab_connection_graph.xml"}
```


Troubleshooting

Issue: The IPs I want to use are unavailable even after running the stop-vms command.

Resolution: If you ran the stop-vms command and still face this issue, you can run the following (outside your docker container)

- **virsh**
- **list**
- **destroy <VM_NAME> #delete the VM that is occupying the IP**
- **exit #exit out of virsh**

```
sonic@cord01:~$ virsh
Welcome to virsh, the virtualization interactive terminal.

Type:  'help' for help with commands
       'quit' to quit

virsh # list
  Id   Name              State
-----
  45   VM0102             running
  46   VM0103             running
  48   VM0100             running

virsh #
```



Troubleshooting

Issue: Task setup failure. SSH Error: data could not be sent to the remote host.

Resolution: There are a plethora of things that could be wrong here. Here are some potential issues:

1. Make sure this host can be reached over SSH
2. Does your group_vars/all/lab_info.json file contain the correct credentials?
3. Does your device have the correct hwsku in files/sonic_lab_devices.csv?
4. Confirm that your lab file does not have "/"s after the Ips. "/"s are a way to denote the subnet mask which INI files do not recognize.
5. Recheck your testbed-new.yaml configuration file to see if you go the IPs and credentials correct.



Troubleshooting

Issue: After importing the docker image, run returns error message “no command specified.”

Resolution: An image imported via docker import won't know what command to run. Any image will lose all of its associated metadata on export, so the default command won't be available after importing it somewhere else

Definitions:

1. docker import – Creates an empty file system and imports the contents of the tarball (so you only get the file system but none of the image metadata)
2. docker load – loads the tarred repo (since “docker save” can save multiple images in a tarball)

Source: <https://forums.docker.com/t/docker-save-or-import-losing-labels-and-entrypoints/25588>



Troubleshooting

Issue: Error message: “Network auto configuration failed. Your network is probably not using the DHCP protocol. Alternatively, the DHCP server may be slow or some network HW is not working properly”

Resolution:

Configure the network manually.

- IP address: 10.62.2.221
- Netmask: 255.255.255.0
- Gateway: 10.62.2.1
- Name Server Address: 10.62.2.1
- Hostname: sonic1
- Domain name: none

Troubleshooting

VMWARE ESXi ISSUE

Issue: The traffic cannot reach the root fanout switch from the lab server

Resolution: Edit your virtual network's security policy to:

- allow promiscuous mode
- all forged transmits
- allow MAC changes

▼ Security policy	
Allow promiscuous mode	Yes
Allow forged transmits	Yes
Allow MAC changes	Yes
▼ NIC teaming policy	
Notify switches	Yes
Policy	Route based on originating port ID
Reverse policy	Yes
Failback	Yes
▼ Shaping policy	
Enabled	No

Smarter. Greener. Together.

To learn more about Delta, please visit www.deltaww.com
or scan the QR code



English



Traditional
Chinese



Simplified
Chinese

