Testbed in a Box

SONiC Case Study
Agenda

• Testbed in a box
• SONiC testing, current state
• SONiC Community Test bed
• SONiC Testbed in a box
  • A peek inside
  • Configuration simplified
  • Benefits
• Demo
Testbed in a box

- **Switch Fabric** (IP, BGP, ECMP)
- **DC Overlay** (VxLAN, Geneve, EVPN, OVSDB)
- **Virtual Infrastructure Validation**
- **Ethernet Storage Fabric** (RoCE, NVMe)
- **Containerized Traffic Generators** (UDP, Apps, Threats)
- **Elastic Multi-Cloud**
- **System under test**

- **Orchestration layer to abstract the apps**

**Disaggregated, Multiple personalities, HW/FW/SW**

**UHD100T32**

**OCP 1RU formfactor**

**Connect. Collaborate. Accelerate.**
SONiC testing - Current state

- SONiC successfully led the disaggregation through open NW Operating System (NOS)
- Disaggregation requires additional testing & validation
  - Testing functionality of each layer
  - Testing for interoperability between multiple vendors
  - Testing end to end performance
  - Verifying vendor’s compliance
- Test Working Group developed test bed topology
- Testbed in a box is our approach to simplification
SONiC Community Test bed

What’s required to build one

- **Hardware/Software**
  - Multiple servers
  - Root/Fan-out switches
  - Multiple VMs for vEOS
  - Multiple PTF containers

- **Configuration**
  - Multiple configuration files, formats, locations and dependencies
  - Feature rich CLI commands prone to human errors

- Building & stabilizing is time consuming
SONiC Testbed in a box

**Orchestration** built in to abstract complex steps from users
- Build virtual root, fan-out switches and connectivity using SDN
- Create cEOS and PTF instances
- Use lightweight cEOS instead of vEOS
- End to end workflow using ansible scripts

**Testing**
- Verified capacity & capabilities of the virtual instances and connectivity
- Ran the community test scripts from Github and similar results observed
What does Orchestration Look Like?

- **Controller**
  - Central entity that manages complexity of testbed orchestration, test logs, test reports, SDN etc.

- **Client**
  - Simple yet powerful user-facing CLI that can talk to the Controller
  - Idiomatic commands to manage testbed, run tests, get test logs, get test reports, manage licenses etc.
Configuration Simplified

topo start [flags]
Flags:
--dut-host string    IP of DUT
--dut-hwsku string   DUT type.
--dut-pass string    Password for DUT
--dut-user string    User for
--topology string    topology type(t1, t0...)

```
./sonic-mgmt/
  ansible
    files
      sonic_labDevices.csv
      sonic_lab_links.csv
    group_vars
      eos
        creds.yml
        eos.yml
      fanout
        secrets.yml
      lab
        lab.yml
        secrets.yml
    vm_host
      creds.yml
      main.yml
  host_vars
    STR-ACS-SERV-01.yml
  inventory
  lab
  minigraph
    sonic Dut.t1.xml
  password.txt
  testbed.csv
  vars
    docker_registry.yml
  veos
  veos-vm
  images
    Aboot-veos-serial-8.0.0.iso
    vEOS-lab-4.20.15M.vmdk
```
## Benefits

<table>
<thead>
<tr>
<th>Simplification</th>
<th>Increase Efficiency &amp; Higher productivity</th>
<th>Space &amp; Power</th>
<th>Ease of Operation</th>
</tr>
</thead>
</table>
| o No need of multiple configuration files  
o Single CLI command to build the test bed connection and execute the suite | o Avoid serial allocation of the setup  
o Each dev engineer could have access to their own setup to develop  
o test & fix faster | o OCP form factor 1U box  
o ~3X power & space reduction | o Connects to Github and downloads relevant files  
o Supports CI/CD  
o Seamless integration into automation |
