Test your way to a better Deployment!

Akshat Sharma, TME, Web Solutions, Cisco.

October, 2016
Network Deployment is pretty Straightforward ....

- Purchase
- Installation (Truck Roll)
- Automated NetOps: ZTP, Config Mgmt....
- Service Activation
...maybe a bit of Pre-staging...
... A bit of monitoring, some hands-on deck, more truck-rolls when things go south....

Purchase

Pre-Staging

Installation (Truck Roll)

NetOps: ZTP, Config Mgmt....

Service Activation
Ok, Network Deployment is complex!

• Hence the need for Validation and Test Cycles.

• These cycles must model real-life deployments and variables.

• Takes time – weeks… mostly months …

• Considerable investment on testing hardware and on good Testers/Developers.
Deconstructing Network Test and Validation
Validation Cycle Requirements

Representative Test Environments

Exhaustive Automated Tests

- Cover Test Topographies / Scenarios
- Actions:
  - Functional
  - Negative
- Validations

Time and Cost Constraints

- Time: Reduce months to weeks
- Cost:
  - Reduce man-hour investment
  - Reduce CAP-EX on Test hardware
The problems at hand...

• Vendor network protocol implementations are **notoriously difficult to test**

• **Custom Vendor specific APIs**

• **Lack of Models:** No consensus on outputs/responses and capabilities

• **Non-overlapping tool coverage:** Ansible coming close to multi-vendor support but no other alternative.

• Cannot commit to one tool over the other. **Test Frameworks need to be modular.**
Creating an open-source Test Framework:

Our Journey
The 5 Commandments:

• Workflow and Tool selection should never be bound by architectures. **Be Flexible.**

• Keep the stack modular and composable.

• **Re-use existing industry tools** – Do NOT start from scratch unless there is a gap.

• **Create a community** to share test cases and extend libraries.

• **Stand on the shoulders of giants:** Leverage work already done by communities like opendaylight, fd.io etc.
Creating one piece at a time…
Topologies

A network is just an undirected graph. Nodes and Edges with certain properties that form the connections.
Topologies

Define a schema, put it in YAML or JSON and run kwalify tests to verify the input is valid.

**Schema**

```json
schema;topology_metadata_map:
  type: map
  mapping:
    version:
      type: any
    schema:
      ...

schema;type_interfaces:
  type: map
  mapping: &type_interface_mapping
    regex;(port\d+):
      type: map
      mapping:
        &type_interface_mapping_port_mapping
          name:
            type: str
          ...
```

---

```yaml
metadata:
  version: 0.1
  schema:
    - <path-to-schema-file>
  tags: [vagrant]

nodes:
  DUT1:
    type: DUT
    host: "192.168.255.101"
    port: 22
    username: password: i

interfaces:
  port1:
    mac_address: ""
    driver: e1000
```

**YAML**

---

```yaml
metadata:
  version: 0.1
  schema:
    - <path-to-schema-file>
  tags: [vagrant]

nodes:
  DUT1:
    type: DUT
    host: "192.168.255.101"
    port: 22
    username: password: i

interfaces:
  port1:
    mac_address: ""
    driver: e1000
```
The orchestrator

- Parse the topology, launch (if needed), verify and return connection objects.
- The orchestrator could be Jenkins, Ansible, test-kitchen or something similar.
The Test framework

• Dozens of tools available – BDD, Data Driven Test, Keyword Driven Test, etc.

• We wanted our test cases to be inherently shareable.

• So we made a bold assertion: **Test cases should not be written in code.**

• Keyword driven Tests won and we chose [http://robotframework.org/](http://robotframework.org/)
The Test Suite Structure

- **Topography/Scenario** is topology dependent.

- The entire test suite is written only using keywords.

- Keywords are exposed by the Robot-framework Libraries.

- These test cases can be shared with the community

**Topography/Scenario**
- Topology-independent Actions:
  - Action 1, validation 1
  - Action 2, validation 2
  - ....

**Validation**
- eg. Define Device roles and set up an L3VPN topology
- eg. Clear BGP neighbors
- Flap interface etc.
- Validates that the scenario works post all the actions.

- These test cases can be shared with the community
The Test libraries – Model Driven

Topography/Scenario

Topology-independent Actions:
- Action 1, validation 1
- Action 2, validation 2
- ....

Validation

Keywords

Model-Driven Actions and Validations

robotframework

ydk-py

Napalm

Ansible
Bringing it all together

Testers

- Topography/Scenario
  - Topology-independent
    - Actions:
      - Action 1, validation 1
      - Action 2, validation 2
      - ....

- Validation

Framework Developers

- Orchestrator
  - Launch
  - Verify Topology

DUT/SUT

- MD-test library
- robotframework
Demo!
Shareable BGP Tests written using robotframework.
Check us out on Github!

- Robo-YDK organization: https://github.com/roboydk
- Robotframework YDK library: https://github.com/roboydk/roboydk
- Ansible-Topology Orchestration: https://github.com/roboydk/orchestrator
- Packet Injection based topology verification: https://github.com/roboydk/topo-verify
Thank you!