Lessons Learned Testing Network Automation Solutions

Observing Sturgeon's Law in Virtual Labs

Ivan Pepelnjak (ip@ipSpace.net)
Network Architect

ipSpace.net AG

Who is Ivan Pepelnjak (@ioshints)



Past

- Kernel programmer, network OS, and web developer
- Sysadmin, database admin, network engineer, CCIE
- Trainer, course developer, curriculum architect
- Team lead, CTO, business owner
- SDN skeptic and network automation evangelist

Present

- Network architect, consultant, blogger, open-source developer

Focus

- SDN and network automation
- Large-scale data centers, clouds, and network virtualization
- Scalable application design
- Core IP routing/MPLS, IPv6, VPN

CISCO CCIE EMERITUS

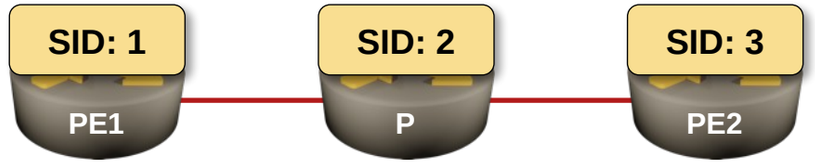
vmware vEXPERT

Also: I built too many labs for one lifetime, and hated that with passion

The Challenge

Imagine You Would Have a Tool That Would…

- Take your high-level intent in infrastructure-as-code format
- Do IPAM and network design for you
- Generate device configurations
- Deploy a running virtual lab



Trivial example:

- Three-router SR-MPLS network running Arista EOS
- IS-IS as the IGP

topology.yml

```
provider: clab
defaults.device: eos
module: [ isis, sr ]

nodes: [ pe1, p, pe2 ]
links: [ pe1-p, p-pe2 ]
```

[brick] $

A Working Lab in 30 Seconds

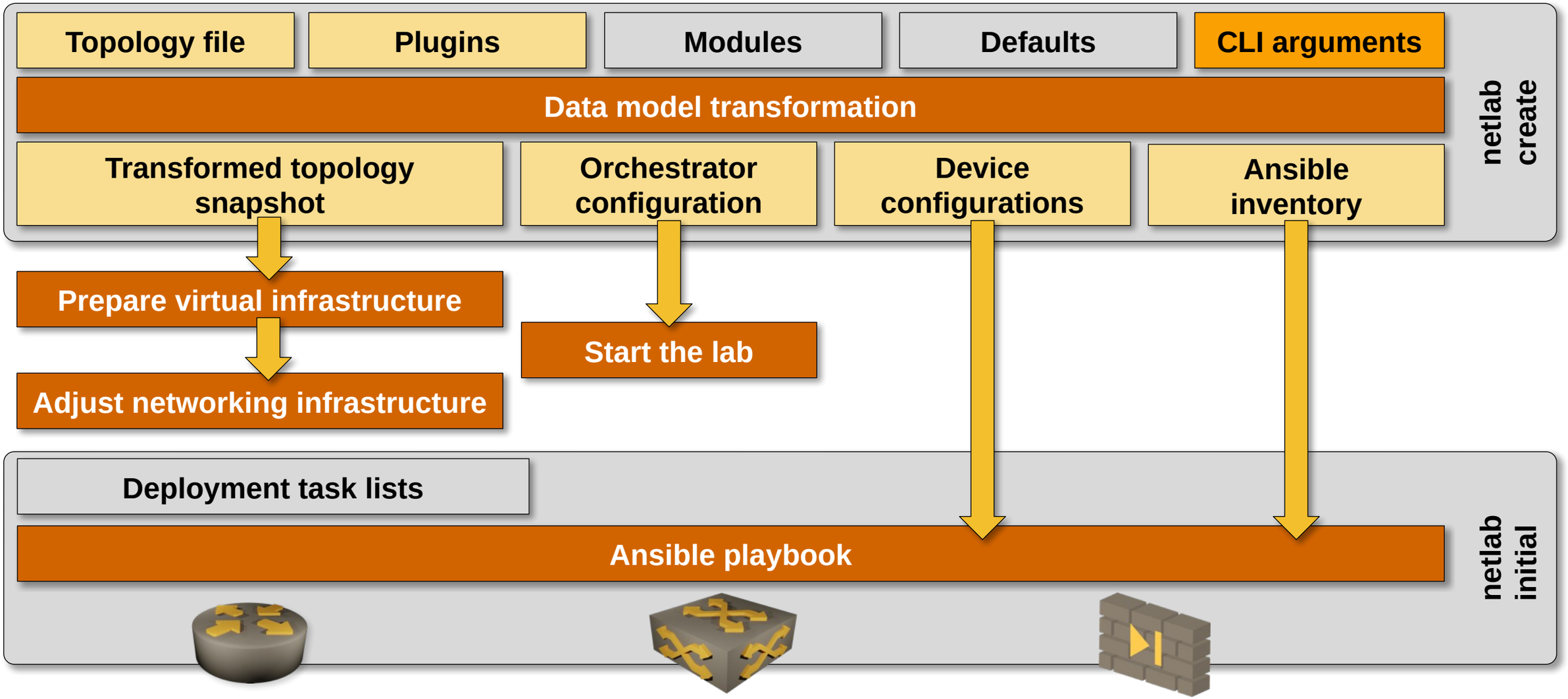
Start the lab ->

SR workshop 0:Python* "sivcek" 18:42 20-May-26

How Do We Know It Works Correctly?

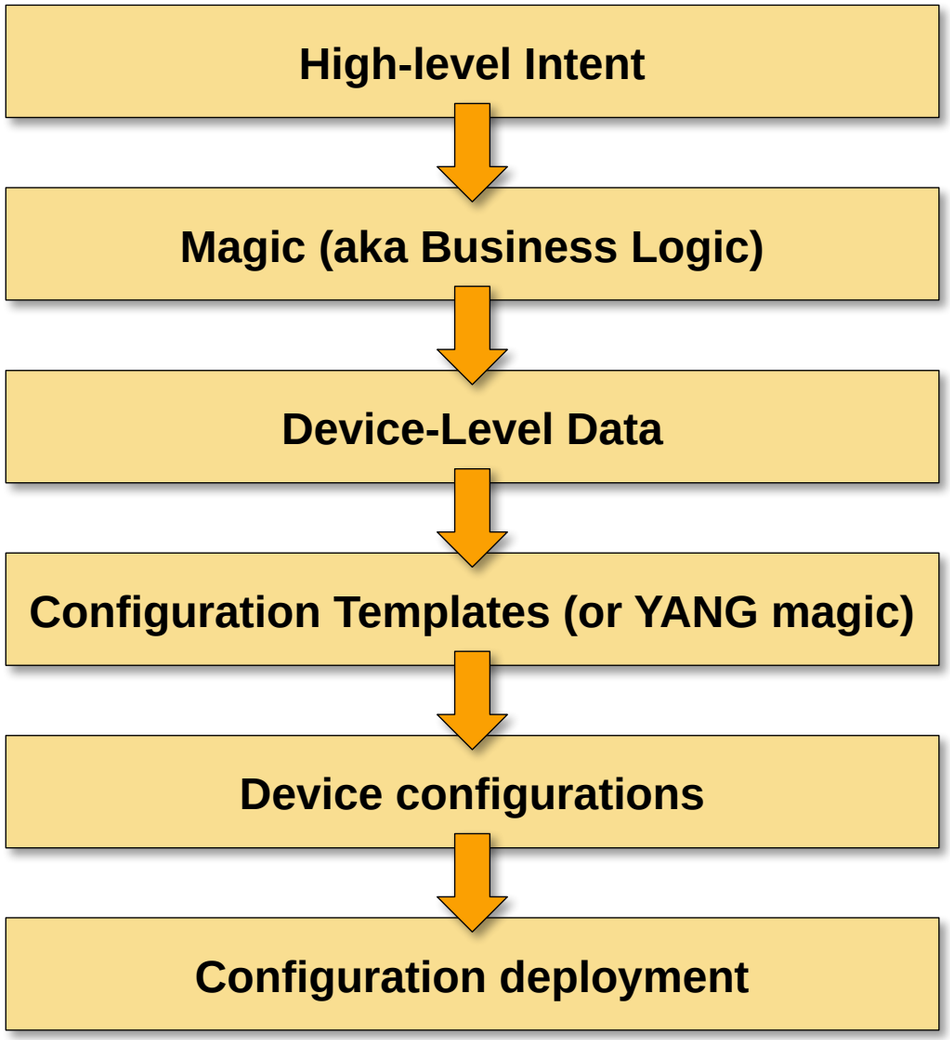
Behind the Scenes

High-Level Architecture Seems Complex…



© ipSpace.net 2022 – 2025

But It Matches Every Decent Service Deployment Automation Project







- Lab topology specifies high-level intent
- **netlab** transforms the lab topology into device-level data structures
- Jinja2 templates are used to transform device-level data structures into device configurations
- Configuration snippets are deployed to network devices (yay, Ansible)

We need more than that to have a lab solution

- The lab has to be started and stopped
- **netlab** has to create some glue around the lab
- A few lab management/reporting functions

© ipSpace.net 2022 – 2025

Testing Methodology

Transformation Tests

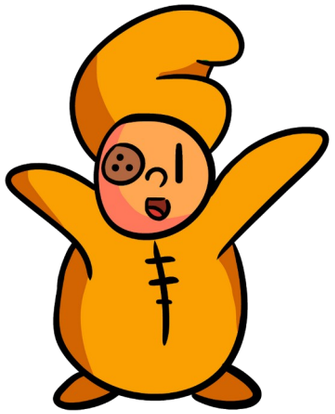
- No traditional unit tests
- Transform predefined lab topologies into device-level data or error messages
- Compare results with expected results

How Does It Work?

- Tests the whole transformation logic
- Input lab topologies drive testing of individual features
- Expected results are stored in YAML format for trivially easy diffs
- Whenever code changes the expected results, the new results have to be part of a PR
- Changes in expected results are reviewed manually

Scope of Transformation Testing

- ~100 transformation tests
- ~100 error tests
- Additional test to improve code coverage



© ipSpace.net 2022 – 2025

Integration Tests

How Does It Work?

- Start a lab topology with one or two devices under test (plus attached hosts and probes)
- Wait for devices to boot
- Use probes (usually FRRouting containers) to check the expected device behavior
- Most checks are done with **show something json** commands

Validates

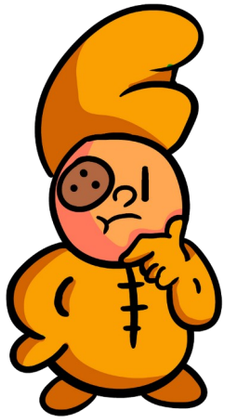
- Jinja2 templates
- Device configuration syntax correctness
- Expected behavior of configured devices

Scope of Integration Testing

- Over 25 test suites
- Over 250 individual tests
- Executed on ~30 different devices ➔ requires over 3700 labs instances

Sad Reality

- It takes days to go through all the tests



Integration test results @ https://tests.netlab.tools/ and https://release.netlab.tools/

Sample Integration Test Results

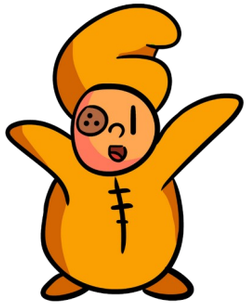
Feature Coverage (Release 26.05)

FEATURE	PASSED	CAVEAT	VALIDATION FAILED	CONFIGURATION FAILED	CRASHED
Initial configuration	217	4			
OSPFv2	253	1			
OSPFv3	283	11			
BGP	391	5			
BGP sessions	212	3			
BGP policies	63				
EBGP multihop	38				
IS-IS	145				
EIGRP	14				
RIPv2/RIPng	69	6			
Routing Policies	147				
Routing Filters	62				
Static Routes	116	2			
VLANs	348	1	4	1	
VRFs	368	1			
VXLAN	77	1			
EVPN/VXLAN	113	6			
EVPN/MPLS	14	1			
EVPN Designs	82	1			

Feature coverage: BGP (Release 26.05)

Test results » Coverage » BGP

DEVICE	EBGP SESSIONS	IBGP SESSIONS	BGP RR	NETWORK ORIGINATION	COMMUNITY PROPAGATION
arubacx/libvirt	✅	✅	✅	✅	✅
bird/clab	✅	✅	✔	✅	✅
cat8000v/libvirt	✅	✅	✅	✅	✅
cisco8000v/clab	✅	✅	✅	✅	✅
crpd/clab	✅	✅	✅	✅	caveat
csr/libvirt	✅	✅	✅	✅	✅
cumulus/libvirt	✅	✅	✅	✅	✅
cumulus_nvue/libvirt	✅	✅	✅	✅	✅
dellos10/libvirt	✅	✅	✅	✅	✅
eos/clab	✅	✅	✅	✅	✅
frr/clab	✅	✅	✅	✅	✅
frr/libvirt		✅			
iol/clab	✅	✅	✅	✅	✅
iol2/clab	✅	✅	✅	✅	✅
iosv/libvirt	✅	✅	✅	✅	✅
iosvl2/libvirt	✅	✅	✅	✅	✅
iosxr/clab	✅	✅	✅	✅	✅
nxos/clab	✅	✅	✅	✅	✅



© ipSpace.net 2022 – 2025

Using pre-commit and GitHub Actions for Fun (and No Profit)

Pre-commit hook (on the user workstation)

- Cannot be enforced
- **yamllint, ruff, mypy** checks
- Some transformation/error tests

On Git Push (GH Action)

- Same as pre-commit hook
- Additional tests for code coverage
- Executed for all supported Python versions

On Pull Request (GH Action)

- Even more testing than on Git Push
- Can take longer (changes will have to be reviewed anyway)
- Includes transformations of all integration tests

Integration testing (on a dedicated server)

- Start manually (takes hours)
- Executed for specific devices/features before merging a PR
- Usually run for all fully-supported devices before a new release goes out

© ipSpace.net 2022 – 2025

Integration Tests: Lessons Learned

It Takes Forever to Start a Lab (without Native Containers)

DEVICE	SW RELEASE	BOOT TIME	INDEX
Cisco IOSv	15.9(3)	00:54.5	100
Arista vEOS	4.34.2F	01:17.3	142
vJunos-switch (QFX)	23.4R2-S2.1	01:39.3	182
vJunos-router (MX)	23.4R2-S2.1	01:39.7	183
Cisco Catalyst 8000v	17.16.01a	01:49.6	201
Cisco 8000v (IOS XR)	24.4.1		
Cisco Nexus OS	9.3.10		



DEVICE	SW RELEASE	BOOT TIME	CONFIG TIME
FRRouting	10.6.1	00:02.0	00:00.3
Juniper cRPD	24.4R1.9	00:07.4	00:01.7
Nokia SR Linux	26.3.2	00:11.6	00:01.5
Arista cEOS	4.35.2F	00:20.9	00:02.2
Nokia SR-SIM (SR-OS)	25.7.R1	00:23.1	00:03.9
Cisco IOL	17.16.01a	00:29.3	00:07.1
Cisco IOS XRd	25.2.1	00:34.6	00:06.0

© ipSpace.net 2022 – 2025

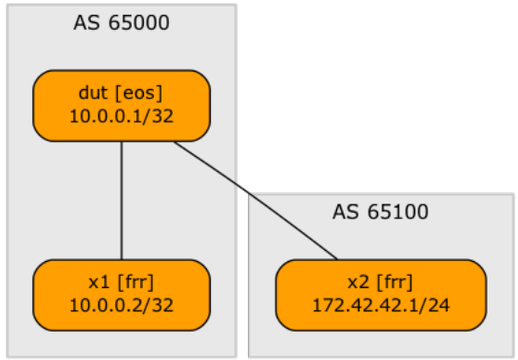
Multi-Platform Testing Is Really Hard

Challenges

- Cannot write tests that would parse **show** printouts for ~20 platforms
- IETF YANG models are still a pipe dream
- SNMP sucks

Solution

- Execute **show** commands (with JSON outputs) on devices **attached to the tested device**
- Indirectly validate configurations through observed protocol behavior



Lessons Learned

- Make sure you're checking the relevant behavior (example: OSPF network type)
- Test small chunks of functionality (ideally a single feature)
- Build a sequence of tests, each one relying on previous tests checking prerequisite features

The Never-Ending Pain

- We have to reboot devices (start new labs) across tests and that takes forever
- Some devices take their sweet time to wake up
- Routing protocol adjacencies sometimes take longer than expected
- Beware of devices having fixed post-boot timers (example: Cisco IOS originating IPv6 BGP prefixes)

© ipSpace.net 2022 – 2025

You Wouldn’t Believe The Network OS Bugs We Found

The Interfaces Aren’t Ready Yet

- Junos VMs start, but the interfaces are not there yet (but can be configured)
- Nexus OS VM starts and rejects interface configuration (for up to a minute)

We Love Stupid Bridges

- Cisco IOS L2 and Extreme EXOS boot as stupid bridges (all ports in VLAN 1)
- Arista cEOS is configured that way by containerlab

Has Anyone Ever Tested This?

- DHCP client does not work on Junos mgmt interface
- Cannot reach SSH server on Dell OS10 mgmt interface
- Linux bridge ports can get stuck in Learning state

Can We Count?

- VXLAN VNI is 16 bit long on Aruba CX VM
- Invalid IP checksum in VXLAN packets on Aruba CX VM
- Junos accepts wide IS-IS metrics and advertises large values as 63

This Can’t Possibly Be True

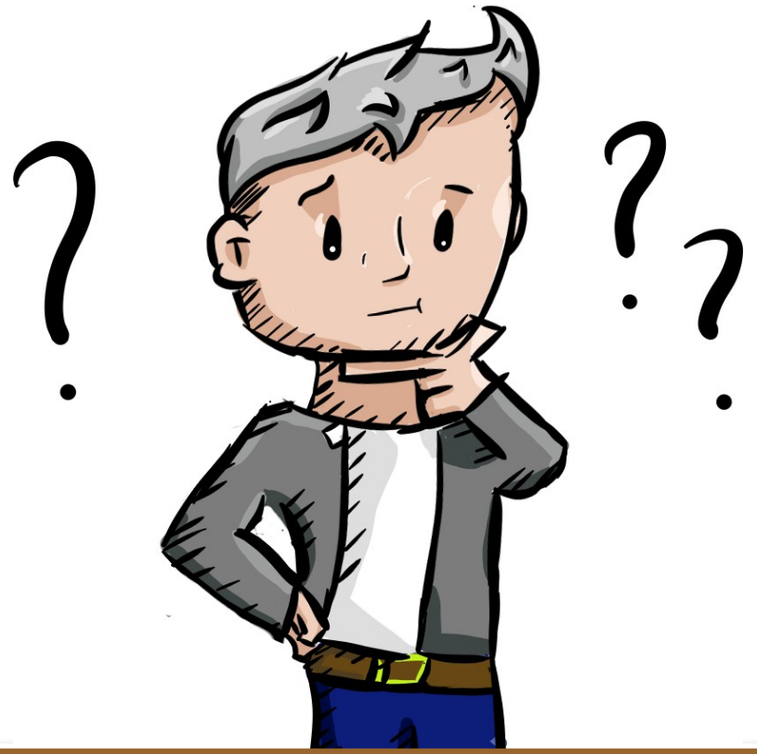
- Most OSPF NSSA implementations are somewhat broken
- We can’t decide how to calculate VRRPv3 checksum
- Every vendor uses slightly different EVPN/ARP defaults
- We also can’t agree on MPLS entropy label with EVPN
- Cisco IOS has two MAC addresses per port (one for L3 interface, one for **switchport**)
- Cisco IOS XR sends a single OSPF hello on passive interface

Questions?

Documentation:	**netlab.tools**
Blog posts:	blog.ipspace.net/tag/netlab.html
Source code:	github.com/ipspace/netlab
Examples:	github.com/ipspace/netlab-examples
Sample project:	bgplabs.net (BGP labs) isis.bgplabs.net (IS-IS labs) evpn.bgplabs.net (EVPN/VXLAN labs)

To reach me

Web:	ipSpace.net
Email:	ip@ipSpace.net



Use GitHub issues / discussions to ask netlab questions and report challenges