# Real-Life Automation Wins

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

Present

- Network architect, consultant, blogger, webinar and book author

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

**More @ ipSpace.net/About**

# It All Started with a Skeptic Comment

# This Is What You Get when Writing about Automation

**Anonymous skeptic**

> Would you mind elaborating bit more on vendor mix for these devices? Some of the environments I am looking at have around 2000-3000 devices and 6-7 vendors for various functions and 15-20 different device platform from those vendors. I am trying to understand what all environments can Ansible scale up to and what would be an idea environment enterprises should be looking at more enterprise grade automation platforms while keeping in mind that platform allows extensibility.

**Reply from someone who got it done**

> 2 platforms, compute fabric, cookie-cutter, rinse-and-repeat. You're trying to boil the ocean. Our data centers are large-scale as well, and I'd never get anything done if I worried about automating the whole blasted thing. Using that as an excuse for an enterprise to not adopt automation is like saying you won't adopt cloud because you still need bare-metal workloads.

More @ http://blog.ipspace.net/2016/08/network-automation-in-enterprise.html

# We Need an Automation Path for Networking Engineers

Read-Only Access

Device Provisioning

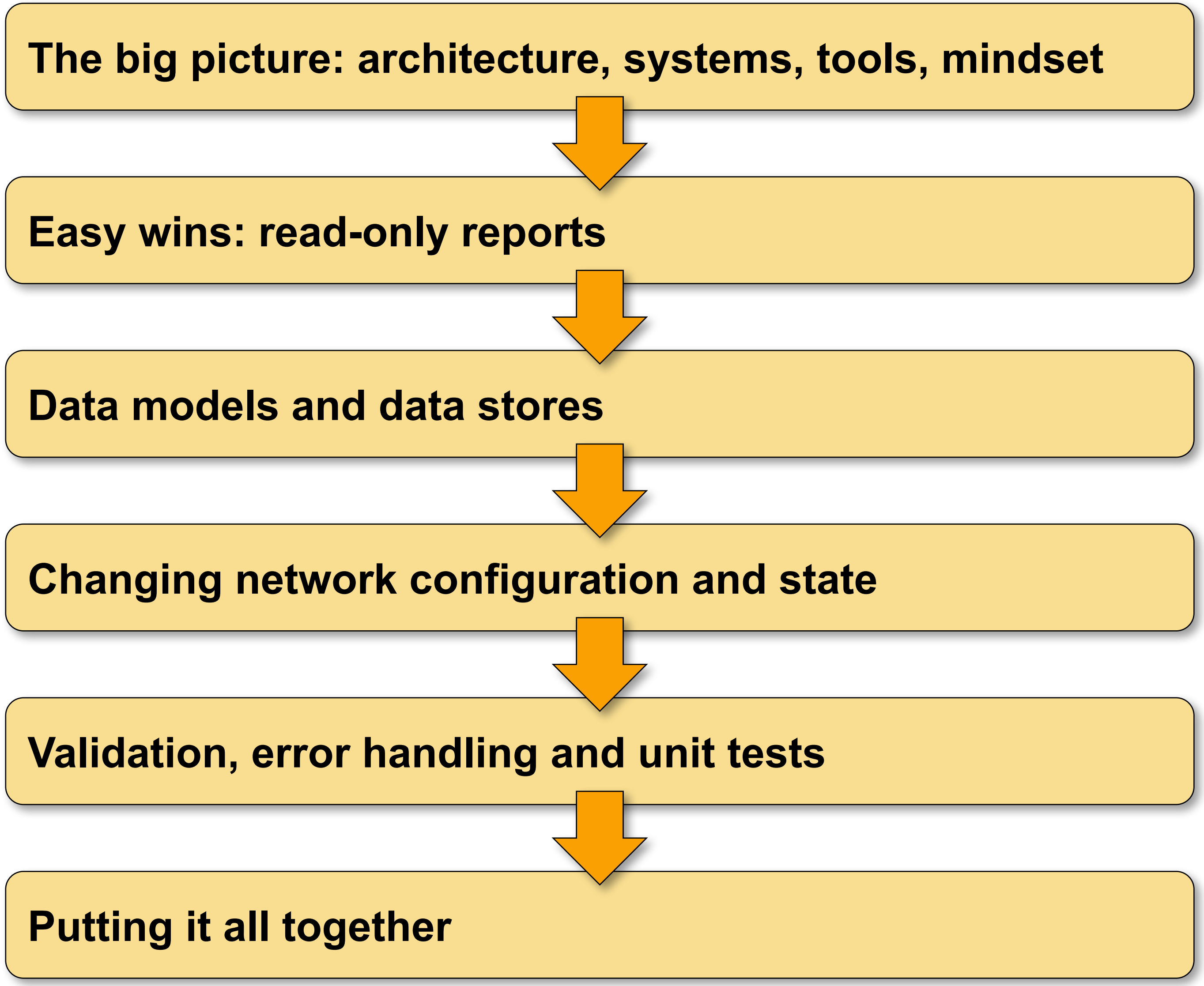
Service Provisioning

Traffic Rerouting

Real-Time and Data Plane

This is how you start

# We Need Structure ➔ Network Automation Online Course



The big picture: architecture, systems, tools, mindset

Easy wins: read-only reports

Data models and data stores

Changing network configuration and state

Validation, error handling and unit tests

Putting it all together

# Why Did Everyone Decide to Build?

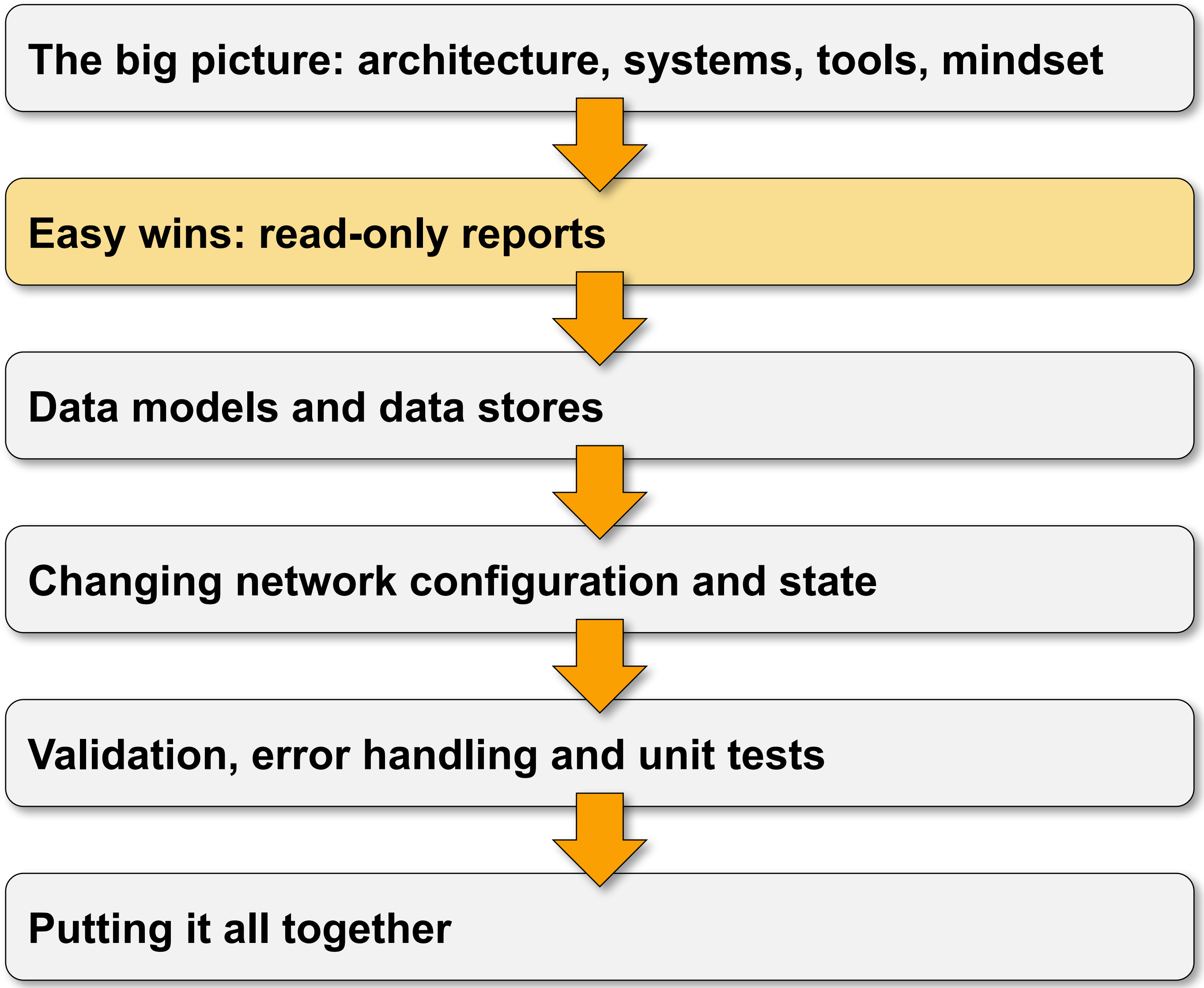
Sample size: ~300 networking engineers attending the automation online course

Selection bias: they wouldn't have been in the course otherwise ;)

- Tools exist, but are not available (or are too expensive)
- Too much work to deploy a high-end tool to get a simple job done
- Tools don't do exactly what you need ➔ you have to build or adapt anyway
- Interacting with black-box tools (even when they have API) is a nightmare
- Building is fun ;)

# Simple Reports

# Progressing on Automation Path



The big picture: architecture, systems, tools, mindset

Easy wins: read-only reports

Data models and data stores

Changing network configuration and state

Validation, error handling and unit tests

Putting it all together

# Uptime Report

| | |
|---|---|
| Author: | Jaakko Rautanen |
| Challenge: | Verify correct operation of UPS systems |
| Industry: | System Integration |

> My customer had major power outage for two huge distribution centers next to each other yesterday. After this the customer asked me to report what devices rebooted during that time to see if the UPS systems were working correctly. There are about 100 switch stacks in the building. Perfect use case for uptime report.

**More @ http://blog.ipspace.net/2017/04/network-automation-is-much-more-than.html**

# Monitoring Progress of a Construction Project

| | |
|---|---|
| Author: | Jaakko Rautanen |
| Challenge: | Monitor Progress of a Construction Project |
| Industry: | System Integration |

> Our subcontractor installs plenty on switches every week to huge site as building construction gets ready. This whole project takes couple of  years. After they're done I have to manually check LLDP neigbors, optical RX powers etc.

> My Ansible playbook gathers LLDP neighbor information, transform it to YAML data model, compare it to predefined reference data and report if all links are connected according to the design. Based solely on Ansible playbook run, I can see which devices are online and if links are connected correctly.

- Another student created a similar solution to verify data center wiring

# Automate End-to-End Latency Measurement

| | |
|---|---|
| Author: | Ruben Tripiana |
| Challenge: | Monitor WAN latency |
| Industry: | Pharmaceutical |

Based on a list of managed devices

- Create a set of unique pairs
- Measure latency using whatever tool is available on the device
- Generate a summary report

More @ http://blog.ipspace.net/2017/11/automate-end-to-end-latency-measurements.html

# Monitor SFP Transceiver Levels

Author: Steve Krause

Challenge: Monitor SFP Transceiver Levels

- Executes **show interface transceiver details** on all managed Nexus switches
- Uses an external Python script to parse the data and extract alarms
- Creates a summary alarm report

```
Ethernet1/47
  type:          10Gbase-SR
  name:          OEM
  part_number:   SFP-10G-SR
  serial_number: ***********
  fault:         Tx Power   N/A     --    1.99 dBm  -10.00 dBm    0.99 dBm     -9.03 dBm
```

**More @ https://github.com/steve-krause/netauto/tree/lab/transciever_report**

# Find Objects in Fortinet Rulesets

|  |  |
|---|---|
| Author: | Simon Thibaudeau |
| Industry: | Financial |
| Challenge: | Replace Fortinet GUI with a useful search tool |

> I am currently writing a tool to search the firewall policy on Fortinet firewalls. Looking for a specific policy on a Fortinet is hard with the GUI and even worse with the CLI. Fortinet's have a JSON API but it is pretty limited in function. However, I can download the whole firewall policy and the whole object database in one blob each and use them.

> Sounds dumb, but a huge time saver so far. "Never underestimate the power of brute force." But in time, I am pretty sure I can use this to start automating the creation of new firewall rules.

> The day I manage to have a web form for IT personnel to make a firewall request to us with the right object names... and all we have to do is make sure their request makes sense before firing it off... will be a great day.

# Create Cisco ACI Reports

| | |
|---|---|
| Author: | Dirk Feldhaus |
| Industry: | Media |
| Challenge: | Hard to Get Quick Overview with Cisco ACI GUI |

- The APIC GUI can difficult to get a quick overview for certain configuration parameters because these values are hidden in different subtrees of the GUI. For example, if you want to check if all Bridge Domains have a certain parameter set this can be really complicated. A list containing only those parameters you're interested in would be a great help.
- The report will be compiled in an excel sheet. Most people are familiar with Excel and sorting and filtering mechanisms.In that way they can easily get the information they're looking for from the report.

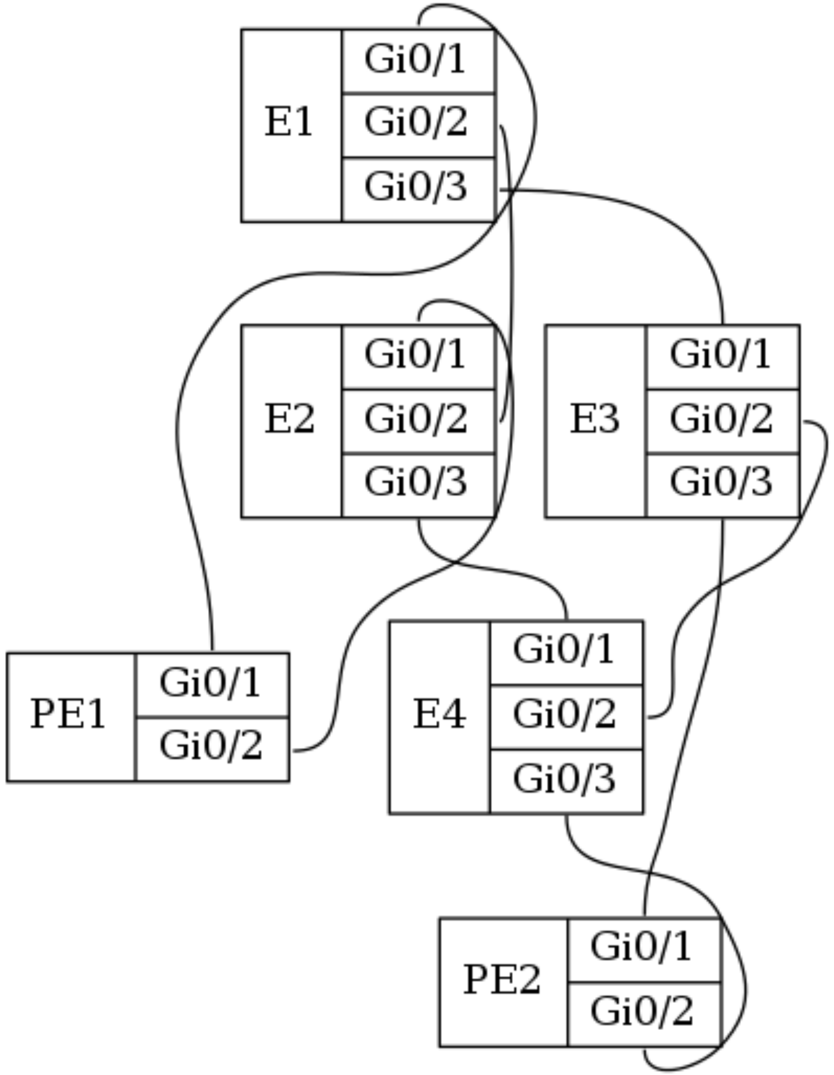
**More @ https://github.com/somebody78/network-automation2/blob/master/summary-report.txt**

# Graphing

# Create Network Diagram from LLDP (or OSPF or IS-IS) Information

Author: Sample Playbook

- Collect network topology data (using LLDP or IGP)
- Convert network topology data into graph description (DOT file format)
- Use open-source tools to draw network diagrams

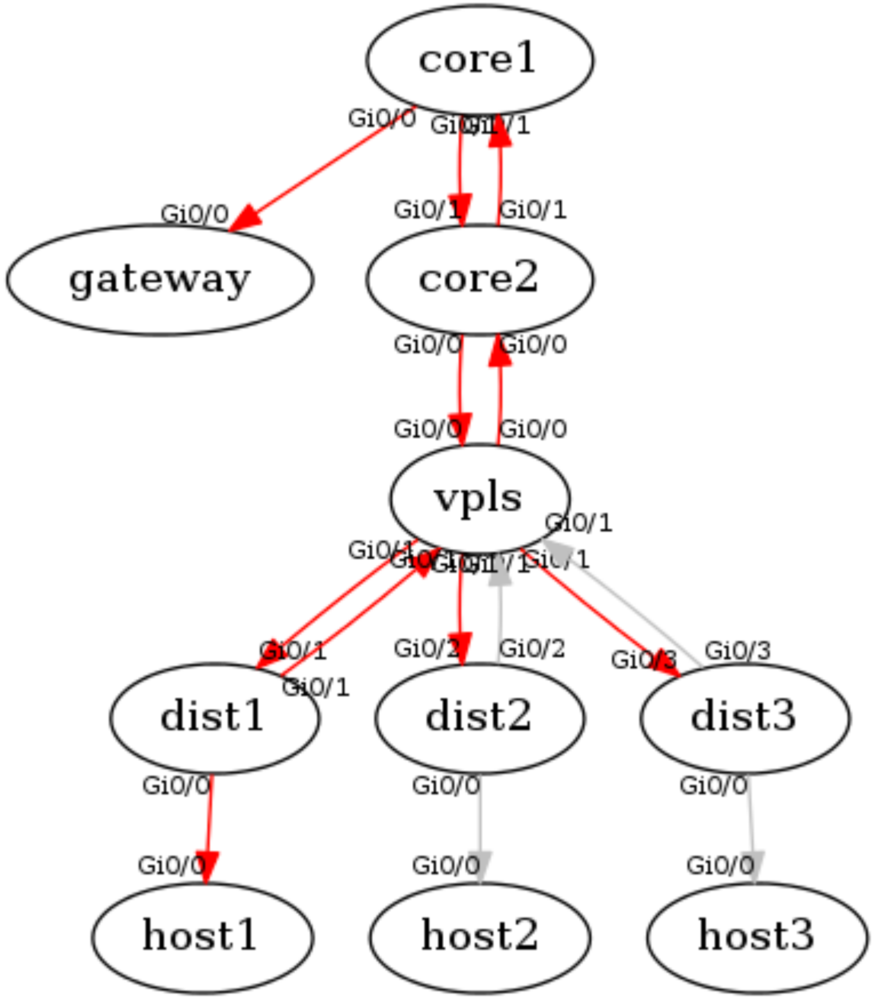


# VLAN Diagrams

| | |
|---|---|
| Author: | James McCutcheon |
| Challenge: | Troubleshoot VLANs |

- Parse **show** command printouts to collect per-VLAN data
- Collect network topology information
- Graph spanning tree for individual VLANs with device- and port names

VLAN 20 Trunking Diagram



More @ http://blog.ipspace.net/2017/11/create-vlan-map-from-network.html

# Creating IP Multicast Tree Graphs

| | |
|---|---|
| Industry: | Financial |
| Challenge: | Troubleshoot IP Multicast |
| Solution: | Draw (*,G) trees based on data collected with **show** commands |

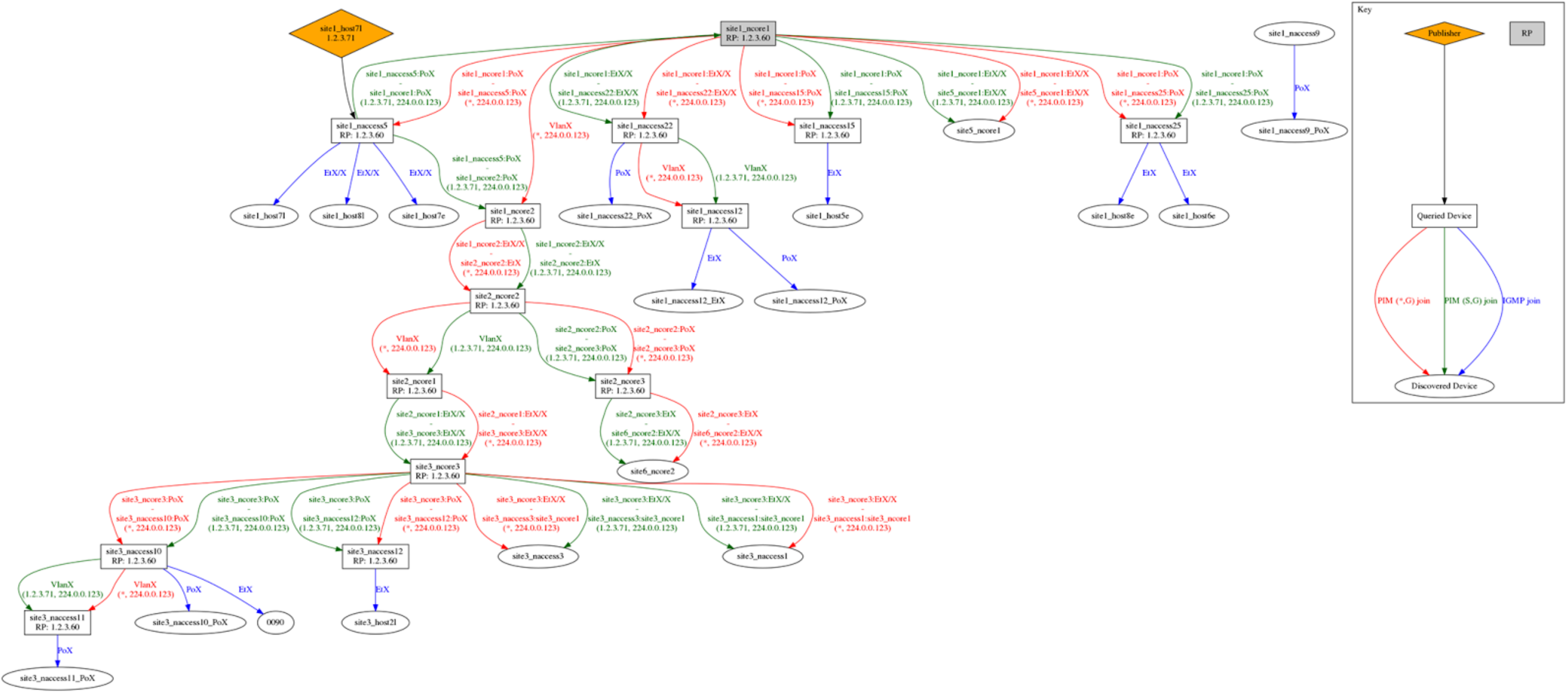
> Data is scraped from switches using ntc_show_command and some custom textfsm scripts. Ansible does the scraping of data and writes out intermediate yaml files per host, then compiles them all into a single document. Then it generates a dot file using a jinja2 template, in which I'm abusing filters to do all the business logic in python.

> Now when an $end_user complains that they're having problems with a particular group, we can kick off this job (which takes a couple mins to fetch all the data, and then a few seconds to graph it) to work out where the flow is coming from and being subscribed to ;)

Ben Roberts

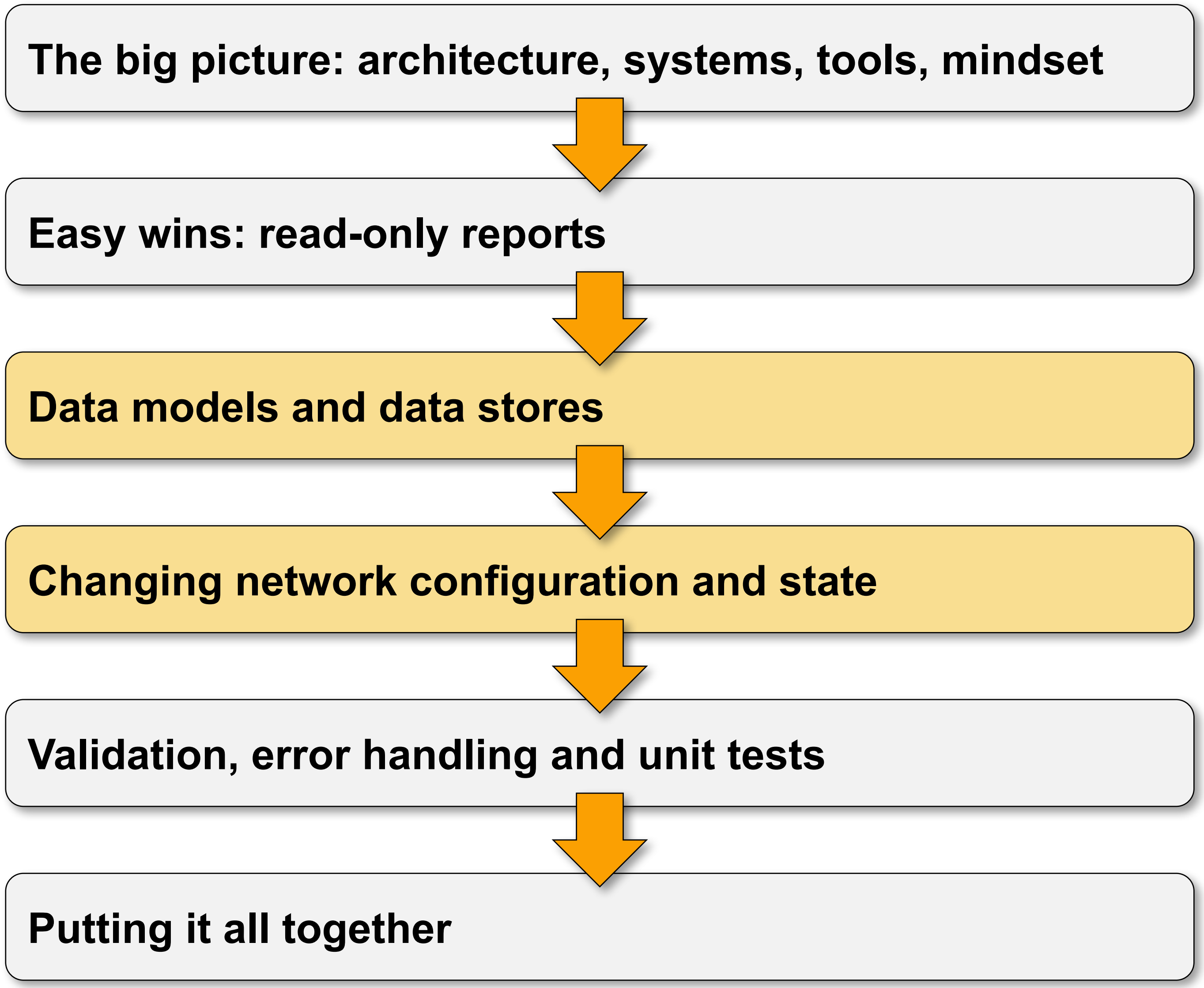
**More @ http://blog.ipspace.net/2017/12/create-ip-multicast-tree-graphs-from.html**

Multicast routing for 224.0.0.123 (compiled )


site1_host71
1.2.3.71
site1_ncore1
RP: 1.2.3.60
site1_naccess5:PoX - site1_ncore1:PoX (1.2.3.71, 224.0.0.123)
site1_ncore1:PoX - site1_naccess5:PoX (*, 224.0.0.123)
site1_ncore1:EtX/X - site1_naccess22:EtX/X (1.2.3.71, 224.0.0.123)
site1_ncore1:EtX/X - site1_naccess22:EtX/X (*, 224.0.0.123)
site1_ncore1:PoX - site1_naccess15:PoX (*, 224.0.0.123)
site1_ncore1:PoX - site1_naccess15:PoX (1.2.3.71, 224.0.0.123)
site1_ncore1:EtX/X - site5_ncore1:EtX/X (1.2.3.71, 224.0.0.123)
site1_ncore1:EtX/X - site5_ncore1:EtX/X (*, 224.0.0.123)
site1_ncore1:PoX - site1_naccess25:PoX (*, 224.0.0.123)
site1_ncore1:PoX - site1_naccess25:PoX (1.2.3.71, 224.0.0.123)
site1_naccess9
PoX
site1_naccess9_PoX
site1_naccess5
RP: 1.2.3.60
VlanX (*, 224.0.0.123)
site1_naccess22
RP: 1.2.3.60
site1_naccess15
RP: 1.2.3.60
site5_ncore1
site1_naccess25
RP: 1.2.3.60
EtX/X
EtX/X
EtX/X
site1_naccess5:PoX - site1_ncore2:PoX (1.2.3.71, 224.0.0.123)
PoX
VlanX (*, 224.0.0.123)
VlanX (1.2.3.71, 224.0.0.123)
EtX
EtX
EtX
site1_host71
site1_host81
site1_host7e
site1_ncore2
RP: 1.2.3.60
site1_naccess22_PoX
site1_naccess12
RP: 1.2.3.60
site1_host5e
site1_host8e
site1_host6e
site1_ncore2:EtX/X - site2_ncore2:EtX (*, 224.0.0.123)
site1_ncore2:EtX/X - site2_ncore2:EtX (1.2.3.71, 224.0.0.123)
EtX
PoX
site2_ncore2
RP: 1.2.3.60
site1_naccess12_EtX
site1_naccess12_PoX
VlanX (*, 224.0.0.123)
VlanX (1.2.3.71, 224.0.0.123)
site2_ncore2:PoX - site2_ncore3:PoX (1.2.3.71, 224.0.0.123)
site2_ncore2:PoX - site2_ncore3:PoX (*, 224.0.0.123)
site2_ncore1
RP: 1.2.3.60
site2_ncore3
RP: 1.2.3.60
site2_ncore1:EtX/X - site3_ncore3:EtX/X (1.2.3.71, 224.0.0.123)
site2_ncore1:EtX/X - site3_ncore3:EtX/X (*, 224.0.0.123)
site2_ncore3:EtX - site6_ncore2:EtX/X (1.2.3.71, 224.0.0.123)
site2_ncore3:EtX - site6_ncore2:EtX/X (*, 224.0.0.123)
site6_ncore2
site3_ncore3
RP: 1.2.3.60
site3_ncore3:PoX - site3_naccess10:PoX (*, 224.0.0.123)
site3_ncore3:PoX - site3_naccess10:PoX (1.2.3.71, 224.0.0.123)
site3_ncore3:PoX - site3_naccess12:PoX (1.2.3.71, 224.0.0.123)
site3_ncore3:PoX - site3_naccess12:PoX (*, 224.0.0.123)
site3_ncore3:EtX/X - site3_naccess3:site3_ncore1 (*, 224.0.0.123)
site3_ncore3:EtX/X - site3_naccess3:site3_ncore1 (1.2.3.71, 224.0.0.123)
site3_ncore3:EtX/X - site3_naccess1:site3_ncore1 (1.2.3.71, 224.0.0.123)
site3_ncore3:EtX/X - site3_naccess1:site3_ncore1 (*, 224.0.0.123)
site3_naccess10
RP: 1.2.3.60
site3_naccess12
RP: 1.2.3.60
site3_naccess3
site3_naccess1
VlanX (1.2.3.71, 224.0.0.123)
VlanX (*, 224.0.0.123)
PoX
EtX
EtX
site3_naccess11
RP: 1.2.3.60
site3_naccess10_PoX
0090
site3_host21
PoX
site3_naccess11_PoX
Key
Publisher
RP
Queried Device
PIM (*,G) join
PIM (S,G) join
IGMP join
Discovered Device

# Configuration Consistency

# Progressing on Automation Path



The big picture: architecture, systems, tools, mindset

Easy wins: read-only reports

Data models and data stores

Changing network configuration and state

Validation, error handling and unit tests

Putting it all together

# Configuration Consistency

Author: Gabriel Sulbaran

Challenge: Common configuration parameters on Checkpoint firewalls

> I had no idea what Ansible was when I started your webinar, and now I already did a really simple but helpful playbook to automate changing the timezone and adding and deleting admin users in a Checkpoint firewall using the command and raw modules. Had to use those modules because there are no official Checkpoint module for the version I'm working on (R77.30).

More @ http://blog.ipspace.net/2018/02/automation-win-cleanup-checkpoint.html

# Configuration Consistency

| | |
|---|---|
| Author: | Pavel Rovnov |
| Challenge: | Common configuration parameters in multi-vendor environment |
| Industry: | System Integrator |

- Consistency of network management parameters: NTP, SNMP, SSH, Syslog
- Extreme switches (used external Perl script to work with EXOS-API)
- Fortinet firewalls
- Cumulus switches and Linux servers

**More @ http://blog.ipspace.net/2018/01/synchronize-network-management.html**

# Simple Device Configurations

# Using YANG Data Models to Configure BGP Neighbors

| | |
|---|---|
| Author: | Ruben Tripiana |
| Industry: | Pharmaceutical |
| Challenge: | Configure Cisco IOS devices using NETCONF and YANG data models |
| Solution: | Ansible playbook that uses NETCONF to configure BGP neighbors using IETF YANG models |

- IETF data model to configure interfaces and BGP neighbors
- Cisco's proprietary data model to enable BFD and configure route maps
- Couldn't get operational data using NETCONF and standard Ansible networking modules

Lessons learned: we're far away from multi-vendor management-plane interoperability

# Zero-Touch Provisioning

Author: Hans Verkerk

Challenge: Implement zero-touch provisioning for Cisco switches

- Store MAC addresses (as printed on the switch) in an Excel spreadsheet
- Use DHCP to assign a random IP address on power-up
- Use TFTP to download minimal configuration to the switch
- Log into the newly-provisioned switch and configure SSH
- Scan for SSH key and store it to **.known_hosts**
- Generate Ansible inventory information from Excel spreadsheet
- Generate final device configuration and push it to the device

**More @ https://github.com/hans-vvv/NetAutLab**

# Automate Equipment Staging

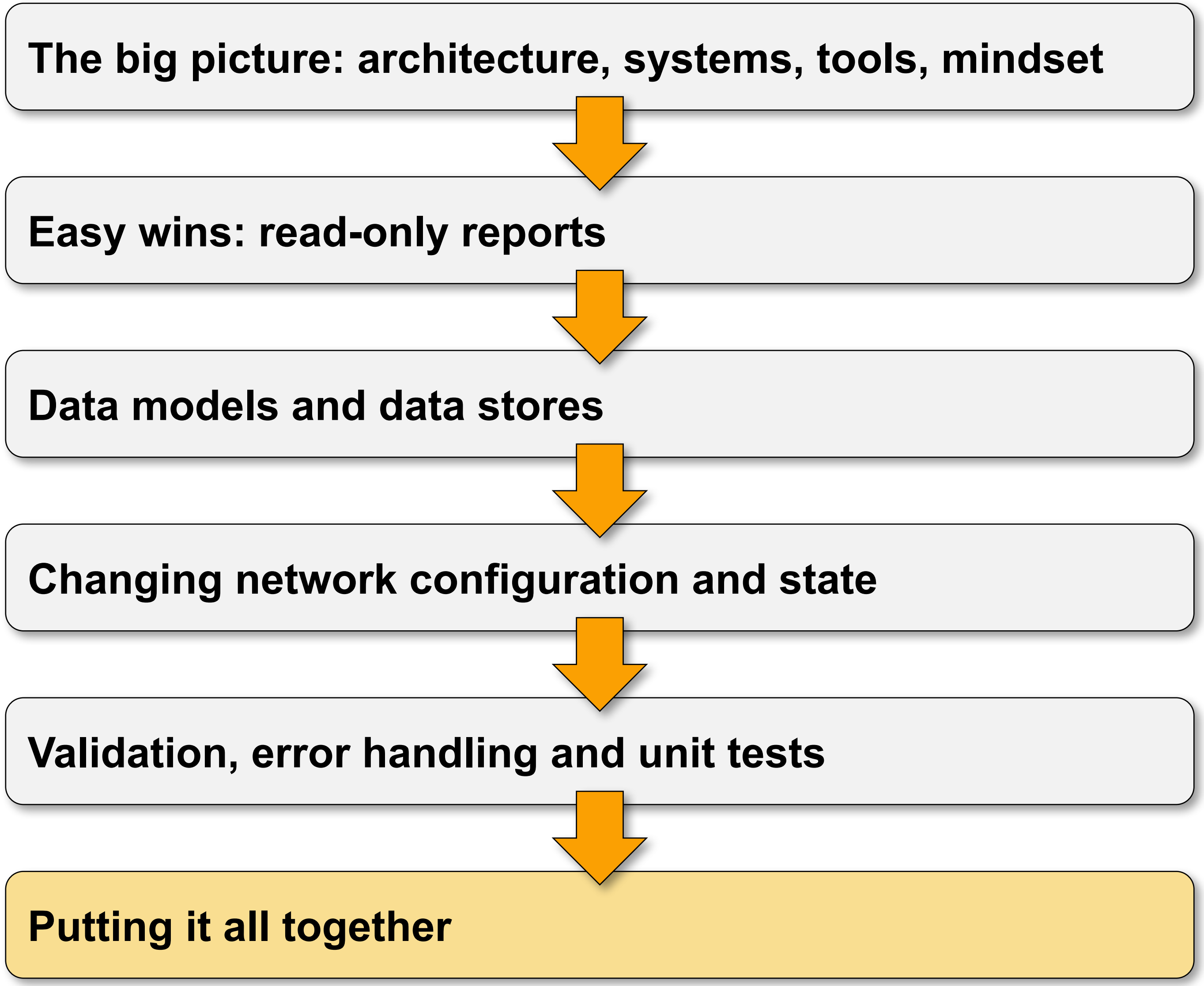
| | |
|---|---|
| Author: | Simon Bitterli |
| Industry: | Service Provider / System Integration |
| Challenge: | Automate Equipment Staging |
| Solution: | Ansible playbooks prepare network devices for customer site deployment |

- Generate device configurations based on a data model describing customer deployment
- Connect to devices via terminal server
- Stop the **setup** process, set up SSH access, read DHCP-assigned OOB IP address
- Cleanup initial configuration
- Deploy final device configuration
- Verify final configurations
- Report results to a Slack channel

**More @ https://github.com/bittsi/bnas/tree/master/Staging**

# Full-Blown Solutions

# Progressing on Automation Path



The big picture: architecture, systems, tools, mindset

Easy wins: read-only reports

Data models and data stores

Changing network configuration and state

Validation, error handling and unit tests

Putting it all together

# MPLS/VPN Service Deployment

| | |
|---|---|
| Industry: | Service Provider |
| Challenge: | Minimize the time to deploy new sites connected to MPLS/VPN network |
| Solution: | Fully-automated configuration generation and deployment based on back-end services database |

> When field techs go on site to complete installation, we can bring site live for our customer 10 minutes after CPE is powered on.

> I’m now almost at full speed: over the last 2 weeks, I have been able to deliver (alone) ~40 dual homed sites (80 CPEs) using Ansible.

> I built a cron job that checks for management loopback interface reachability every 5 minutes and runs all-in-one playbook to check/configure PE/CPE, automate final tests and sends confirmation email to internal teams and customer with tech details so they can start monitoring the devices.

Francois Herbet

**More @ http://blog.ipspace.net/2018/02/automation-win-mplsvpn-service.html**

# Remote Site Hardware Refresh

Industry: Government

Challenge: Replace and reconfigure equipment at ~ 1900 sites

Solution: Fully-automated configuration generation and device provisioning

> We now deploy whole WAN sites within an hour.
>
> Stan Strijakov

**More @ http://blog.ipspace.net/2017/12/automate-remote-site-hardware-refresh.html**

# Remote Site Hardware Refresh (Details)

- Site information is collected in an Excel spreadsheet and converted to YAML data file
- Web-based GUI for the conversion process

Conversion steps

- **Pre-configuration**: generate new device configurations for all devices deployed on site
- **Router configuration**: download new configuration to WAN router, enable temporary DHCP scope
- **Core switch setup**: software upgrade, new configuration, final DHCP scope
- **Building topology discovery**: connect to all access switches to collect CDP information
- **Topology check**: GUI-based validation of site topology and wiring
- **Final software upgrade**: tier-by-tier upgrade of switch software
- **Final checks**: Rediscover the topology, verify it matches the expected topology
- **Security setup**: static or reserved DHCP addresses, DHCP/ARP trust settings, VLANs on edge ports
- **Wireless configuration**: Configuring Aruba Cluster Master WAP
- **Documentation**: update switch port names

**More @ http://blog.ipspace.net/2017/12/automate-remote-site-hardware-refresh.html**

# Large-Scale Firewall Deployment

| | |
|---|---|
| Author: | Benjamin Papillon |
| Challenge: | Deploy firewall clusters at ~100 sites and integrate them with existing network |
| Solution: | Fully-automated configuration generation and device provisioning |

- Create and deploy Cisco configuration snippet
- Configure the firewalls, create local firewall objects
- Derive site rules from template ruleset, change the objects
- Bootstrap the firewalls (manual process)
- Finalize the firewall configuration (templated)
- Move into production, validate
- Insert the new firewalls into monitoring system

# Large-Scale Firewall Deployment: Achievements

Author: Benjamin Papillon

Challenge: Deploy firewall clusters at ~100 sites and integrate them with existing network

Solution: Fully-automated configuration generation and device provisioning

- Centralize data used to generate configurations
- All network parameters (firewall ip/netmasks, routing, cisco ports informations, ...) are inside 1 yaml file
- All network objects used for firewall rules are separated into 3 files: firewall policy, global objects, local objects
- 1 playbook to create all the base configurations, using roles to separate technologies (cisco / fortinet / check point)
- 1 playbook to add a site and checks to monitoring system
- 1 playbook to automatize configuration backups, disregard to the equipment type
- 1 playbook to push arbitrary command to appliances
- Configuration changes are tracked in git

# Large-Scale Firewall Deployment: Lessons Learned

Author: Benjamin Papillon

Challenge: Deploy firewall clusters at ~100 sites and integrate them with existing network

Solution: Fully-automated configuration generation and device provisioning

- The migration time improved a lot from what the client used to do
- We gain the usual automation bonus, like backup configurations or push commands whenever we wish
- We learned to have "faith" in the toolset we built: if we encounter an error during a migration, we know it's not the configuration. It's either a typo in the dataset or onsite problem like cabling error
- Cisco configurations were reviewed only once by client's network experts. They reviewed the templates and not the configs for each and every deployment.
- We had some young engineers very reluctant to push this kind of change, the client was much more open than the techies.

# Network Infrastructure as Code

| | |
|---|---|
| Author: | Mark Prior |
| Customer: | Betfair |
| Environment: | Private cloud |

Phase 1: Automate the build

- Arista ZTP: check topology, upgrade firmware, deploy configurations
- Automated tests with Ansible and Python: check LLDP neighbors, BGP neighbors, MLAG…

Phase 2: Network Infrastructure as code

- Device configurations generated from templates and data models (currently in YAML files)
- Version control with Git
- Automated staging and testing with Jenkins
- Automated configuration deployment (including rollback on errors) with Ansible
- Status reporting via Slack

**More in Building Network Automation Solutions course**

# Build or Buy?

# Can You Get the Tool You Need?

# Classic Stages of Vendor Acceptance

1. I hear you
2. I see your point
3. I understand what you’re saying
4. No one has asked for that before
5. I’ll see what we can do
6. I’ll get back to you
7. It’s on the roadmap

# Software-Defined Stages of Vendor Acceptance

1. I hear you
2. I see your point
3. I understand what you’re saying
4. No one has asked for that before
5. **We have an API**
6. I’ll see what we can do
7. I’ll get back to you
8. It’s on the roadmap

# You’ll Have to Build Anyway

# How Low Do You Want to Start?

**Fixed-functionality systems with API**

- VMware NSX / Cisco ACI / APIC-EM

**Platforms**

- Cisco NSO
- Apstra AOS
- VMware vRealize Orchestrator

**Low-level tools**

- Ansible, SaltStack, Chef, Puppet…
- StackStorm

# You’ll Be Developing Software No Matter What

## Get used to it

- The only way to get agile is to automate deployments
- The only way to automate deployments is to buy or build automation solutions
- You’ll have to adapt (or write add-on modules) most out-of-the-box solutions anyway
- Don’t trust vendors’ PowerPoints (or their solutions)
- You don’t have to become programmer
- You **MUST** think about **SYSTEMS** and **PROCESSES**

> The real tiger is never a match for the paper one, unless actual use is wanted.

Mythical Man-Month (Frederick P. Brooks, 1975)

# Make the Network a Better Place

# Questions?

Send them to ip@ipSpace.net or @ioshints