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

More @ ipSpace.net/About
Every Well-Defined Repeatable Task Can Be Automated
What Would You Automate?

Common answers:
• Device provisioning
• Service provisioning (= device configurations)
• VLANs
• ACLs
• Firewall rules

How about…
• Troubleshooting
• Consistency checks
• Routing adjustments
• Failure remediation
Automation
↓
Repeatability
↓
Consistency
↓
Validation
Automation = Eliminate Repeatable Manual Tasks

Orchestration = Group Automated Tasks in Coordinated Workflows
A Few Reasons for Lack of Network Automation

**Major ones**
- Mission-critical nature of the networks
- Unique snowflakes that are impossible to automate
- Ad-hoc solutions and non-standard kludges
- Blast radius
- Lack of trust

**There’s also**
- Lack of programming skills
- Lack of reliable automation tools and programmatic interfaces
- Lack of (semi)standardized multi-vendor configuration schema
- Lack of affordable test environment
Hierarchy of Network Needs

Automated Remediation

Automated Provisioning

Abstraction of network state

Operated network

Functioning Network

Source: Jeremy Stretch, packetlife.net
Operated Network
Operated Network

• Box-by-box mentality
• Manual configuration through CLI
• Relationships between boxes are managed in brain-space
• Tight control of changes and maintenance windows due to inherently unreliable configuration processes

Immediate improvement opportunities

• Configuration repository = single source of truth
• Change tracking (version control)
• Configuration changes tied to user requirements or business needs

Tools to use

• RANCID – collect network configurations
• Subversion or Git – version control
Typical Workflow

1. Propose device configuration changes
2. Reviews and approvals
3. Schedule maintenance window
4. Change device configuration
Store Device Configurations in a Repository

1. Propose device configuration changes
2. Reviews and approvals
3. Schedule maintenance window
4. Change device configuration
5. Collect device configurations
6. Store new configurations into repository
Start with Configuration Repository

1. Fork codebase, make proposed changes
2. Submit changes to the repository
3. Review and approve change
4. Make change
5. Collect device configurations
6. Store new configurations into repository

- Start with a single source of truth
- Easy to identify original and changed versions
- Using standard tools for reviews and approvals
- Rollbacks are easier
- Proposed versus implemented change
- Repository again contains single source of truth
The Final Twists

Fork codebase, make proposed changes

Submit changes to the repository

Review and approve change

Deploy changes automatically

Allow your customers to propose changes

More @
- *NAPALM* – Elisa Jasinska & David Barroso, NANOG64
Abstraction of Network State
Hierarchy of Network Needs

- Functioning Network
- Operated network
- Abstraction of network state
- Automated Provisioning
- Automated Remediation

Source: Jeremy Stretch, packetlife.net
Simplify
→ Standardize
→ Abstract
→ Automate
Network State Abstraction: Before and After

upgrade fpd auto
version 15.0
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname R2
!
boot-start-marker
boot-end-marker
!
logging buffered 4096
!
no aaa new-model
!
interface Loopback0
  ip address 10.0.1.5/32
  
  
!
interface Fa0/0
  ip address 172.16.11.1/24
...

hostname: 'R2'
loopback: { ip: 10.0.1.5 }
LAN:
  interface: 'Fa0/0'
  ip: 172.16.11.1
Network Deployment: Before and After

- Business needs
- Network design
- Desired network state
- Configuration templates
- Device configurations
Benefits of Abstracted Network State

- Explicit mapping from network design to desired state and device configurations
- Separation of infrastructure state and service state
- Simplified multi-vendor deployments

Easier to:
- Validate configuration compliance
- Compare current state with desired state
- Identify mismatches or manual changes
- Change device configurations
Automatic Provisioning
Automated Network and Service Provisioning

Automation required by
- Large scale deployment
- Self-service requirements
- Faster service deployment
- Need to improve reliability

Prerequisites
- Standardized services, configurations and deployment processes
- Reliable method of configuring and monitoring network devices (API)

Tools to use
- Configuration state management tools: Chef, Puppet
- Automation frameworks: Ansible
- Workflow and continuous integration tools: Gerrit, Jenkins
Go for Low-Hanging Fruits
Read-Only Access

Device Provisioning

Service Provisioning

Traffic Rerouting

Real-Time and Data Plane
Automated Remediation
Automated Network Remediation

Holy Grail: Networks that fix themselves or adapt to changes

A few examples:

• Identify links with degraded performance ➔ reroute traffic
• Identify router problems (memory leaks) ➔ drain the traffic, reload the device
• ToR switch failure ➔ migrate the virtual machines

Getting there:

• Don’t expect a vendor-supplied miracle
• Someone will have to do extensive customization
• Try to use small, reusable components
Example: Facebook-Defined Networking

Facebook Defined Networking

Source: How Facebook Learned to Stop Worrying and Love the Network (Jose Leitao, David Rothera, RIPE 71)
Network Automation Caveats
This is what makes networking so complex
To make mistake is human. To automatically deploy mistake to all of servers is DevOps.
"I SPEND A LOT OF TIME ON THIS TASK. I SHOULD WRITE A PROGRAM AUTOMATING IT!"

**THEORY:**
- Writing Code
- Work on Original Task
- Automation Takes Over

**FREE TIME**

**REALITY:**
- Writing Code
- Debugging
- Rethinking
- Ongoing Development
- No Time for Original Task Anymore

Source: http://xkcd.com/1319
More Information
Network Automation Track

- Network Automation Use Cases
  - Jinja2, YAML and Ansible
  - NETCONF & YANG Deep Dive
  - Network Automation 101
  - Network Automation Tools
  - Network Programmability 101
  - REST API

More information @ http://www.ipSpace.net/NetOps
SDN, OPENFLOW AND NFV RESOURCES ON IPSPACE.NET

Software-defined networking (SDN) can mean anything, from programmable network elements to architectures in which control and forwarding planes reside on different devices.

The resources listed on this page will help you understand SDN, its implications and its applicability in your environment.

SDN TRAINING AND CONSULTING

- On site and online consulting
- SDN, OpenFlow and NFV Workshop
- Software Defined Data Centers (SDDC) Workshop
- Advanced SDN Training
- Introduction to SDN
- Customized webinars and workshops

INDIVIDUAL SDN WEBINARS

- NETCONF and YANG
- Network Programmability 101
- SDN Architectures and Deployment Considerations
- VMware NSX Architecture

MORE SDN WEBINARS

SDN-RELATED BOOKS

- Overlay Virtual Networks in Software-Defined Data Centers

BUY NOW

- SDN and OpenFlow

BUY NOW

PRESENTATIONS

- SDN - 4 Years Later (video)
- What is SDN?
- Should I program my network? (video)
- Virtual Routers
- From Traditional Silos to SDDC (video)
- What Matters is Your Business (video)
- Automating Network Security, Troopers 15

MORE SDN PRESENTATIONS

MORE SDDC PRESENTATIONS
Stay in Touch

Web: ipSpace.net
Blog: blog.ipSpace.net
Email: ip@ipSpace.net
Twitter: @ioshints

SDN: ipSpace.net/SDN
Webinars: ipSpace.net/Webinars
Consulting: ipSpace.net/Consulting
Even More to Explore

Blogs and web sites:
• Matt Oswalt (keepingitclassless.net)
• Scott Lowe (blog.scottlowe.org)
• Michael Kashin (networkop.github.io)
• Jason Edelman (jedelman.com)
• Chris Young (kontrolissues.net)
• Patrick Ogenstad (networklore.com)
• Josh O’Brien (staticnat.com)

Github repositories:
• NAPALM (https://github.com/napalm-automation)
• David Barroso (https://github.com/dbarrosop) – SIR, NAPALM demos
• Jason Edelman (https://github.com/jedelman8)
• Patrick Ogenstad (https://github.com/networklore)
Questions?

Send them to ip@ipSpace.net or @ioshints