Network Operations as Code

September 2018
**DevOps**

Aim? To reduce the time, cost and errors of software deployment, while maintaining compliance.

Goal? Of being more agile and responsive to business needs.

How? Defining state in code with configuration management and use automation to maintain state.
NetOps

- NetOps has a similar aim to DevOps
  - i.e. being more responsive to business needs
- However, in networking, stability is critical
  - contradiction with the desire for agility
- For NetOps to be successful it must enable network management personnel to increase agility while ensuring compliance and reducing risk
Puppet Overview

Who we are and what we do
Using a common language

Get a standard way to deliver & operate all of your software

- Define once with an easy-to-understand language
- Improve collaboration by unifying processes and tooling
- Get started quickly by choosing from existing modules, or create your own
- Open-source provides scale for building out content
Infrastructure as Code

Describe the ideal environment with a simple, commonly understood language

```plaintext
building {  'home':
    ensure => 'clean',
    front_door => 'closed',
    keys => 'key_hook',
    jacket => 'closet',
    floor => 'vacuumed',
    litter_box => 'empty',
    remote => 'coffee_table',
}
```
Control & enforce consistency across your devices

Make changes with confidence & deliver faster

- Orchestrate changes to infrastructure on-demand or on-schedule
- Simulate changes using no-op
- Continually enforce desired configurations
- Automatically remediate misconfigurations & unexpected changes
- Run ordered deployments based on dependencies you define
Simulation and no-op

Only change what you need to when you need to

• Puppet is idempotent
  ○ Config is only updated when it doesn’t match the catalogue

• Simulation is possible and strongly advised
  ○ no-op: this is what will change if you run this command for real
package { 'openssh-server':
    ensure => installed,
}

file { '/etc/ssh/sshd_config':
    source => 'puppet:///modules/sshd/sshd_config',
    owner  => 'root',
    group  => 'root',
    mode   => '0640',
    notify => Service['sshd'], # sshd restarts whenever this file is changed.
    require => Package['openssh-server'],
}

service { 'sshd':
    ensure => running,
    enable => true,
}
banner { 'default':
    motd => 'Hello, world!',
}

cisco_interface { 'ethernet1/1':
    ensure => 'present',
    ipv4_address => '192.168.1.1',
    ipv4_netmask_length => '24',
    mtu => '1600',
    shutdown => false,
    access_vlan => 1,
    switchport_mode => disabled,
}

ios_config { $name:
    command => $command,
    idempotent_regex => $regex,
}
How Puppet Works

Continuous Enforcement - Agent

1. Facts
2. Catalog
3. Report

Node (agent)
Puppet Master
How Puppet Works

Continuous Enforcement - Agentless

1. Facts

2. Catalog

3. Report

Puppet device

Node (no agent)

Puppet Master

Puppet device

Puppet device

Puppet device

Puppet device
Know the types of changes

Status values indicating what happened during a Puppet run

• Failure
• Corrective change
• Intentional change
• Corrective no-op
• Intentional no-op
• Skip
# Puppet Enterprise Reports

Know when changes occur and why

## nxos-local-1

<table>
<thead>
<tr>
<th>Reported at</th>
<th>No-op mode</th>
<th>Total resources</th>
<th>Correction applied</th>
<th>Failed</th>
<th>Changed</th>
<th>Unchanged</th>
<th>No-op</th>
<th>Skipped</th>
<th>Failed restarts</th>
<th>Config retrieval (sec)</th>
<th>Run time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018-01-05 00:00:00Z</td>
<td>-</td>
<td>178</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>178</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.9</td>
<td>0.9</td>
</tr>
<tr>
<td>2018-01-05 01:00:00Z</td>
<td>-</td>
<td>178</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>177</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.7</td>
<td>1.4</td>
</tr>
<tr>
<td>2018-01-05 03:00:00Z</td>
<td>-</td>
<td>178</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>83</td>
<td>95</td>
<td>-</td>
<td>-</td>
<td>0.8</td>
<td>4.6</td>
</tr>
<tr>
<td>2018-01-03 23:30:00Z</td>
<td>-</td>
<td>178</td>
<td>2</td>
<td>-</td>
<td>2</td>
<td>176</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.9</td>
<td>1.3</td>
</tr>
<tr>
<td>2018-01-03 23:34:00Z</td>
<td>-</td>
<td>178</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>89</td>
<td>89</td>
<td>-</td>
<td>-</td>
<td>1.4</td>
<td>8.8</td>
</tr>
</tbody>
</table>
Know what you have

Gain situational awareness & understand exactly what’s happening across your software

- Monitor exactly what you have running across your data center & cloud
- View changes taking place in real-time and report on the cause of those changes
- Visualize dependencies across your infrastructure & apps to improve change success rate
Deployment Model

Perform multi-vendor device management at scale with a single language

Puppet Master with catalogues

Puppet Agent, running Puppet Device and module

Cisco Nexus

Palo Alto

Cisco IOS

F5

Modules from: Cisco Nexus, Palo Alto, Cisco IOS, F5 Big-IP, Cisco ACI, Cisco Meraki, Netscaler, NetApp, Huawei, Arista, Cumulus, Lenovo CNOS
Pipeline concept: the future for network automation?

Run changes through pipelines

- Create policies
- Create manifest
- Validate manifest against policies
- Pre-deployment check of change against network
- Apply code in production
- Post-deployment health check
Time for a demo
NetOps Principles

- Automate - move away from the command line as much as possible
- Define state in code
- Manage compliance in code
- Use pipelines to run pre- and post-deployment checks
- Trust the tools
- Be open to change
Adopting NetOps

- Walk before running - take a single device type and try to automate common tasks
- Define policies and desired state in code
- Take a pipeline approach to test before deployment
- Use no-op to simulate before making the change
- Take an open-source approach
- Check out what other people are doing, like Netflix’s Winston: https://bit.ly/2phEqTe
Thanks!

davin.hanlon@puppet.com