Interop ITX
May 15-19
MGM Grand
Las Vegas

Hands-on Practical Network Automation

Configuration with Templates

Jere Julian @julianje

Jere Julian

DevOps and Infrastructure-as-code Evangelist
18+ years automating network devices
Extensibility Engineer @ Arista
Raleigh Puppet User Group
Firefighter & EMT-B

@julianje

Structured data

- Text blobs
  - ➢human consumption
- Structured data
  - ➢machine consumption

Flickr: Fabio Omero

Raw text

```
localhost#show version
Arista vEOS
Hardware version:
Serial number:
System MAC address:  0800.2789.fd90
Software image version: 4.16.7M
Architecture:           i386
Uptime:                 12 minutes
Total memory:           1897592 kB
Free memory:            46952 kB
```

Structured data - XML

```
<?xml version="1.0" ?>
<root>
        <memTotal type="int">1897592</memTotal>
        <systemMacAddress type="str">08:00:27:89:fd:90</systemMacAddress>
        <bootupTimestamp type="float">1494892747.32</bootupTimestamp>
        <memFree type="int">46828</memFree>
        <version type="str">4.16.7M</version>
        <modelName type="str">vEOS</modelName>
        <architecture type="str">i386</architecture>
</root>
```

Structured data - JSON

```
localhost#show version | json
{
    "modelName": "vEOS",
    "systemMacAddress": "08:00:27:89:fd:90",
    "memTotal": 1897592,
    "bootupTimestamp": 1493394075.23,
    "memFree": 46828,
    "version": "4.16.7M",
    "architecture": "i386",
}
```

Structured data - YAML

```
---
architecture: 'i386'
bootupTimestamp: 1494892747.32
memFree: 46828
memTotal: 1897592
modelName: 'vEOS'
serialNumber: ''
systemMacAddress: '08:00:27:89:fd:90'
version: '4.16.7M'
```

Transforming structured data

Explore structured_data.py in the repo

Structured data - YAML

```
---
- name: Name of this resource
  hosts: arista
  vars:
    # A comment
    ports:
      - Ethernet1:
          description: ESXi32 eth0
          enable: True
      - Ethernet2
          enable: False
      - Ethernet3
```

Typical configuration

```
queue-monitor length

!

queue-monitor length cpu thresholds
524288000 524288000

no queue-monitor length cpu

!

Hostname dc01-rack20-tor01

ip name-server vrf cinnamonbits
8.8.8.8

ip domain-name example.com

!

ntp server 192.0.2.240 iburst

!

snmp-server host 192.0.2.43 version
2c snmp-user

snmp-server enable traps lldp

!
```

```
spanning-tree mode mstp

!

aaa authorization exec default local

aaa authorization commands all
default local

aaa accounting commands all default
start-stop logging

!

username admin privilege 15 role
network-admin secret 7 *****

!

clock timezone EST5EDT

!

vlan 1920

   name LAB_PROD_19-20

!

vrf definition cinnamonbits
```

```
   rd 100:101

!

interface Port-Channel56

   description bigserver1 bond0

   switchport access vlan 1920

   spanning-tree portfast

   spanning-tree bpduguard enable

!

interface Ethernet1

   description server1 vmnic0 Mgmt

   load-interval 5

   switchport access vlan 1920

   switchport trunk native vlan 1920

   switchport trunk allowed vlan
1920,2526
```

```
   switchport mode trunk

   queue-monitor length thresholds
40962 10241

   spanning-tree portfast

   spanning-tree bpduguard enable

   vmtracer vmware-esx

!

interface Loopback1

   description VxLAN VTEP

   ip address 192.0.2.234/32

!

interface Management1

   vrf forwarding cinnamonbits

   ip address 192.0.2.213/27

!
```

Typical configuration… Cont'd

```
interface Vlan1920
   no autostate
   ip address 192.168.110.67/26
   ip helper-address 192.168.1.24
   ip virtual-router address
192.168.110.65
!
interface Vlan2526
   no autostate
   ip address 192.0.2.3/26
   ip helper-address 192.168.1.24
   ip virtual-router address
192.0.2.1
!
interface Vxlan1
   vxlan source-interface Loopback1
   vxlan udp-port 4789
   vxlan vlan 104 vni 20104
   vxlan flood vtep 192.0.2.235
!
ip virtual-router mac-address
be:ef:ca:fe:19:20
!
ip route vrf cinnamonbits 0.0.0.0/0
192.0.2.193
!
ip routing
no ip routing vrf cinnamonbits
!
ipv6 unicast-routing
!
router ospf 1
   network 192.168.110.64/26 area
192.168.110.64
   network 192.0.2.0/26 area
192.0.2.0
   network 192.0.2.234/32 area
0.0.0.0
   network 192.0.2.236/31 area
0.0.0.0
   network 192.0.2.238/31 area
0.0.0.0
   max-lsa 12000
   maximum-paths 4
!
vmtracer session eosplus-vc
   url https://192.0.2.87/sdk
   username rtp-rack20-
tor01@vsphere.local
   password 7 *****
   autovlan disable
   no allowed-vlan
!
!
management api http-commands
   protocol http
   protocol unix-socket
   no shutdown
   vrf cinnamonbits
      no shutdown
!
!
end
```

Logical pieces

- Management connectivity
- Time Synchronization (NTP)
- AAA
- Logging
- Routing
- Uplink ports
- Device ports

Flickr: Horia Varlan

Splitting configuration

- Reusable chunks
- DRY principal – Don’t Repeat Yourself

Data Driven Networking

Data → Template → Device

Flickr: John Spencer

Template capabilities

- Variable substitution
- Conditionals
- Iteration
- Inheritance

Jinja2 – basic

```
interface {{ port_name }}
    description {{ description }}
    no shutdown
```

Jinja2 – intermediate

```
{% for port in ports %}
interface {{ port.name }}
    description {{ port.descr }}
    {% if port.vlan is defined-%}
    switchport access vlan {{ port.vlan }}
    {% else -%}
    no switchport access vlan
    {% endif -%}
{% endfor %}
```

YAML data for the template

```
---
- ports:
    - name: 'ethernet1/1'
      descr: host32-eth0
      vlan: 100
    - name: 'ethernet1/2'
```

Templates – Jinja2 & Python

```
#!/usr/bin/env python
''' Render a template '''

from jinja2 import Template

template = Template("interface {{ intf }}\n   shutdown")

for x in [1, 2]:
    print template.render(intf="Ethernet{}”.format(x))
```

Render a template with Python

render_template.py

render_template_file.py

creates → render_output.cfg

Ansible

Open-source automation tool

Playbooks and Roles

YAML syntax

Extensible with Python

Ansible Galaxy modules and roles

Ansible Tower - Commercial UI, tooling, support

Gather some data with Ansible

ansible-playbook -i ansible-hosts check_version.yml -v

Templates – Jinja2 & Ansible

```
- vars:
    ports:
      - { name: Ethernet2, descr: Available }
      - { name: Ethernet3, descr: host32-eth0, vlan: 100 }

- name: Render templates to configs
    template:
      src: templates/{{ ansible_host }}-adv.j2
      dest: configs/{{ ansible_host }}.cfg
```

Push with Ansible playbook

```
$ ansible-playbook -i ansible-hosts config_push.yml

PLAY [Configure nodes from templates] ******************************************************************
TASK [Gathering Facts] *********************************************************************************
ok: [arista1]
TASK [Ensure config dir exists] ************************************************************************
ok: [arista1]
TASK [Render configs from templates] *******************************************************************
ok: [arista1]
TASK [Push config commands to nodes with napalm] *******************************************************
ok: [arista1]
PLAY RECAP *********************************************************************************************
arista1                    : ok=4    changed=0    unreachable=0    failed=0
```

Render a template with Ansible

…and push configuration to a device

ansible-playbook -i ansible-hosts push_config.yml -v

Resources

- http://jinja.pocoo.org/docs/latest/
- http://docs.ansible.com/ansible/template_module.html
- https://www.ansible.com/network-automation
- https://github.com/napalm-automation/napalm-ansible
- https://pynet.twb-tech.com/blog/automation/napalm-ios.html

Hands-on Practical Network Automation

Thank You

Jere Julian @julianje