Network Automation and Programmability

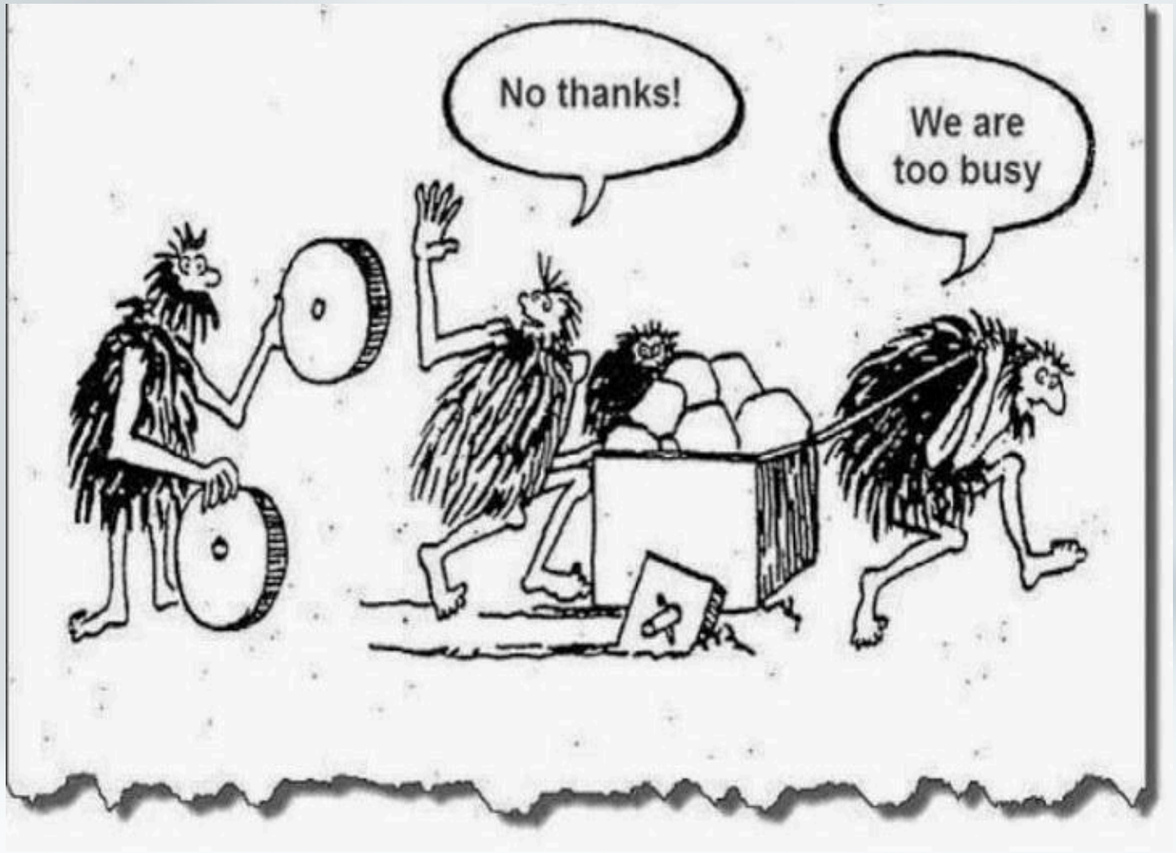
DENOG7

Peter Sievers

AGENDA

- Intro
- Automation
  - Junos Stack
  - Tools
- NETCONF & YANG
- IT Frameworks
- Building Blocks
- Examples
- Key Takeaways

Why should we care?



Networking industry is “stuck in the past”
(performance improvement taken for granted)

Networks seen as unique, complex and slow to evolve

Network engineers still value their “CLI” expertise
(xCIE)
(culture + politics issues)

Little to no automation in place for a vast of customers
(complex infrastructure, multi-vendor, legacy)

Minutes to deliver a VM, weeks to attach it to the infra:
“Networking is in the way”

Automation and Programmability

Automation:

Provide the ability for users to ***optimize workflows***, and bring a **coherent view** of operational resources.

Programmability #1:

Provide a ***core system*** that enables every function to be controlled

Programmability #2:

Provide ***programmatic interfaces*** to access core programmability features.

Programmability #3:

Provide users the ability to ***extend the platform*** to meet their unique needs.

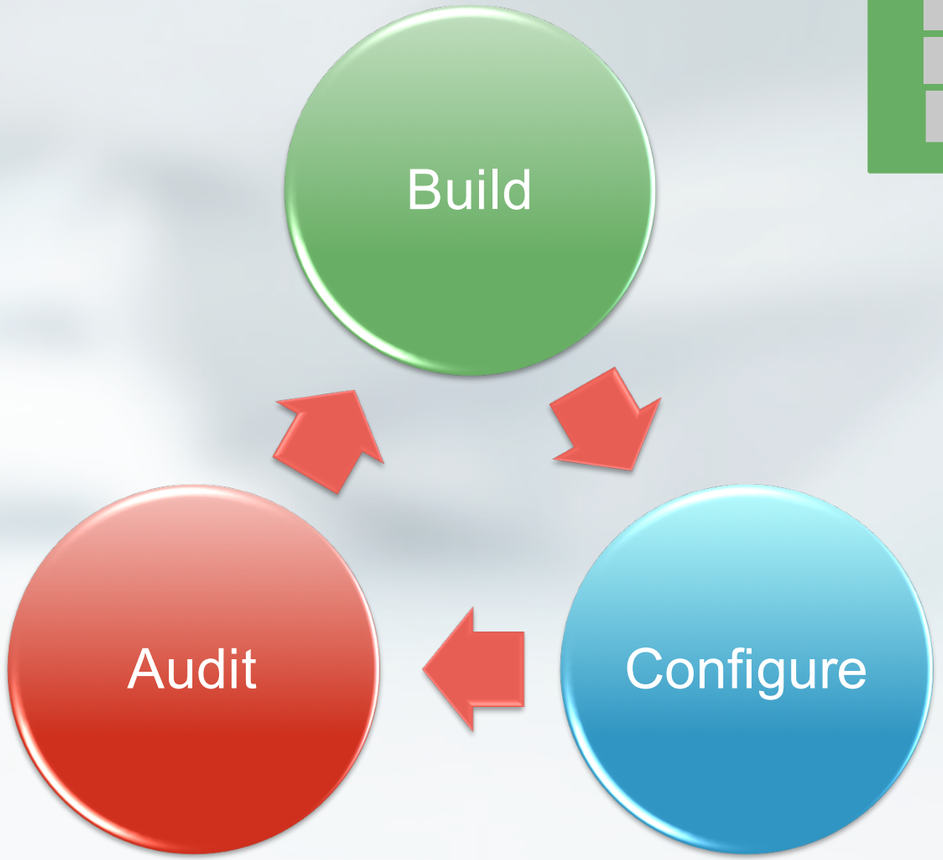
Programmability enables Automation

Classification of (opened) Automation Tools

The ***Build*** phase centers around the initial design and installation of a network component.

Build

- ZTP
- Ansible
- PyEZ
- OpenClos



Audit

- SNMP
- Netconf / DMI
- JunosScript
- PyEZ

Configure

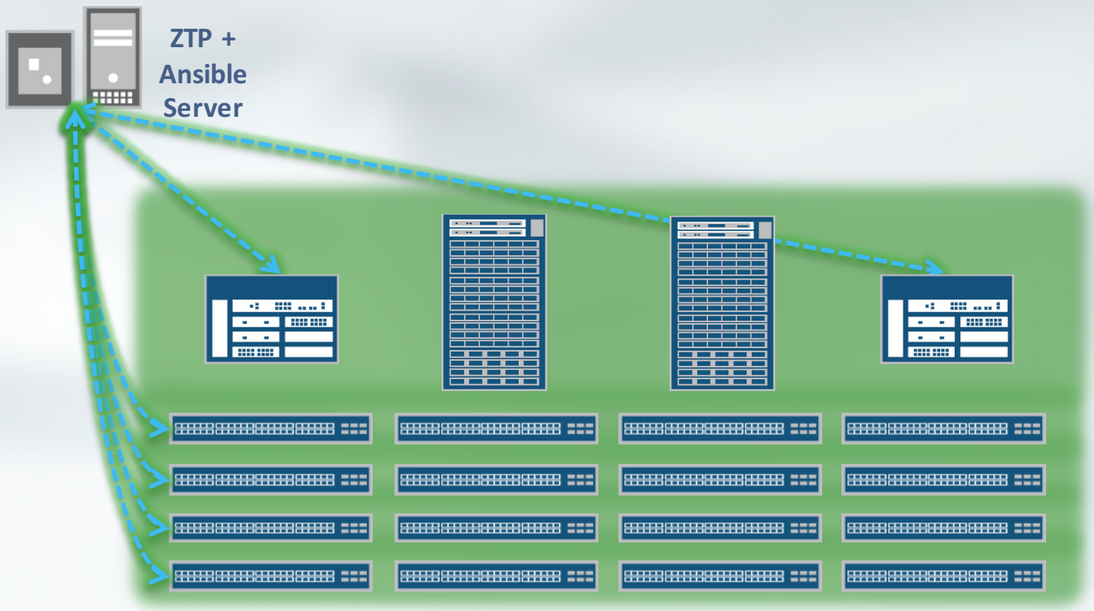
- Chef
- Puppet
- Ansible
- PyEZ

The ***Audit*** phase deals with automating the process of monitoring operational state of the platform and reacting on state conditions.

The ***Configure*** phase covers methods to deploy on demand configuration and software changes to the platform.

Use case – Automate the DC Build

Build	Configure	Operate



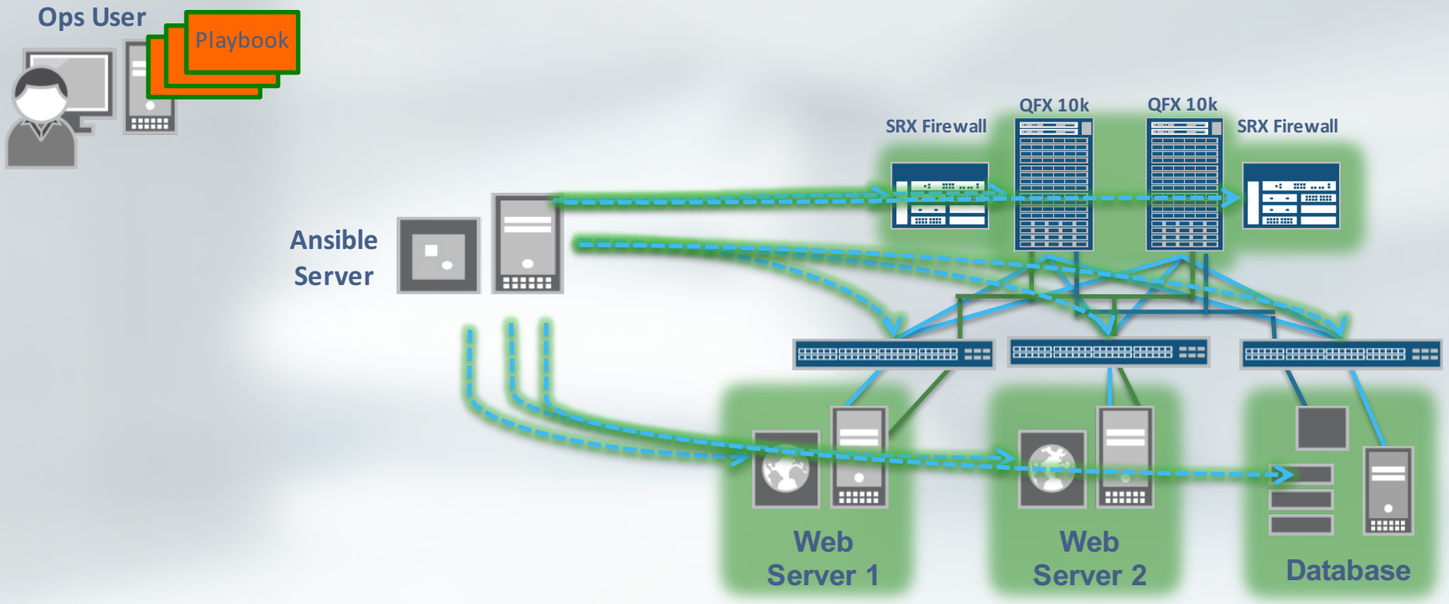
1. New Equipment Gets Connected
2. All configurations downloaded
3. All equipment online and operational

BENEFITS

- Minimal skill required by onsite deployment team
- Ensure Consistent deployment in line with company policies
- Reduces Data Center Build out from days to minutes

Use case – Configuration Management

Build	Configure	Operate



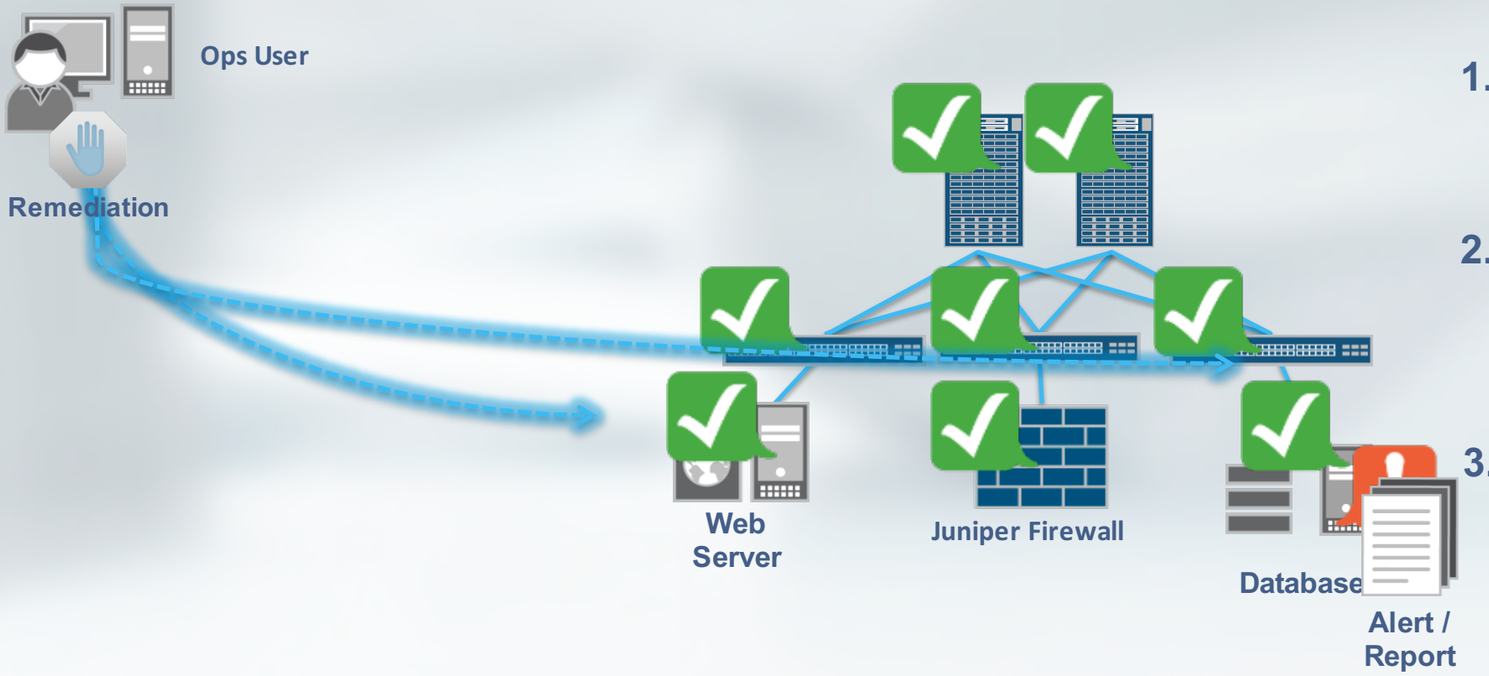
1. User runs a Playbook that describes actions across multiple nodes
2. Ansible Server then uses a *push* method via SSH or NETCONF to make changes across nodes
3. Changes are orchestrated in order across the network and compute infrastructure

BENEFITS

- Agentless Interaction via SSH, Telnet/Console or NETCONF
- Workflow Engine allows ordered and coordinated changes between nodes
- Available across Junos product families – EX and QFX Switches, MX and PTX Routers, and SRX Firewalls

Use case – Network Troubleshooting

Build	Configure	Operate



1. Administrators create scripts for data collection to check network health
2. In the event of error in automated tests, an alert / report is generated for the administrator
3. Administrator creates automated remediation to fix errors on detection

BENEFITS

- Operational Workflow Automation allows operations staff to schedule tasks
- Create reports based on “Out of Profile” events
- Ability to automate “Remediation Actions” based on report data to improve network availability and reduce MTTR

Automation

AUTOMATION

Solving real customer technical and business problems at every level

Domain

Domain Automation takes the concepts of NaaS, and extends it to an entire domain, including network, compute and storage to deliver end services at a rapid pace and high scale.

Network

Network Automation aims to abstract the individual platforms, and operate as a “Network As A Service (NaaS)” model. As such, the network is treated as a single entity.

Platform

Platform Automation focuses on delivering programmatic access to individual components in the network to automate day to day activities and reduce response times..

AUTOMATION

Platform

Platform Automation focuses on delivering programmatic access to individual components in the network to automate day to day activities and reduce response times..

python™

junos JET

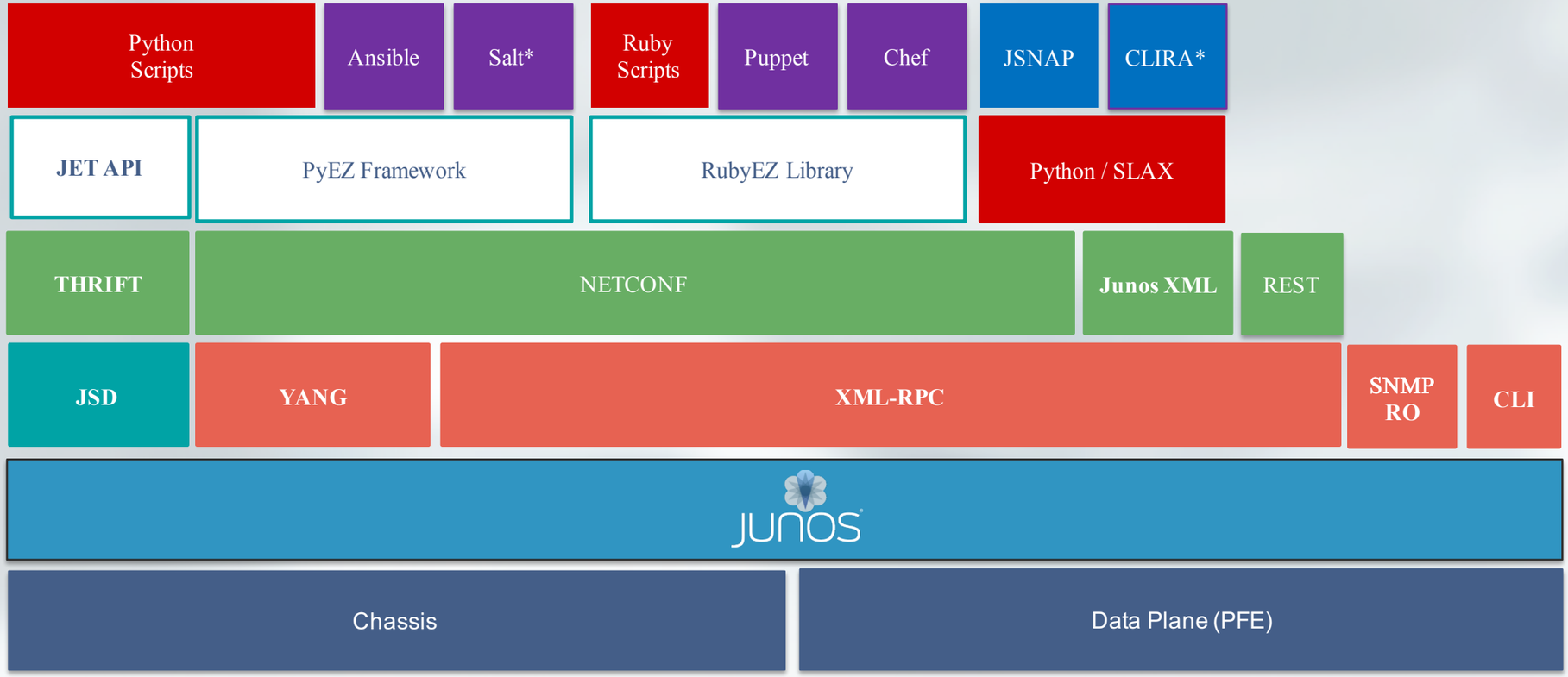
puppet labs

<SLAX>

ANSIBLE

CHEF CODE CAN

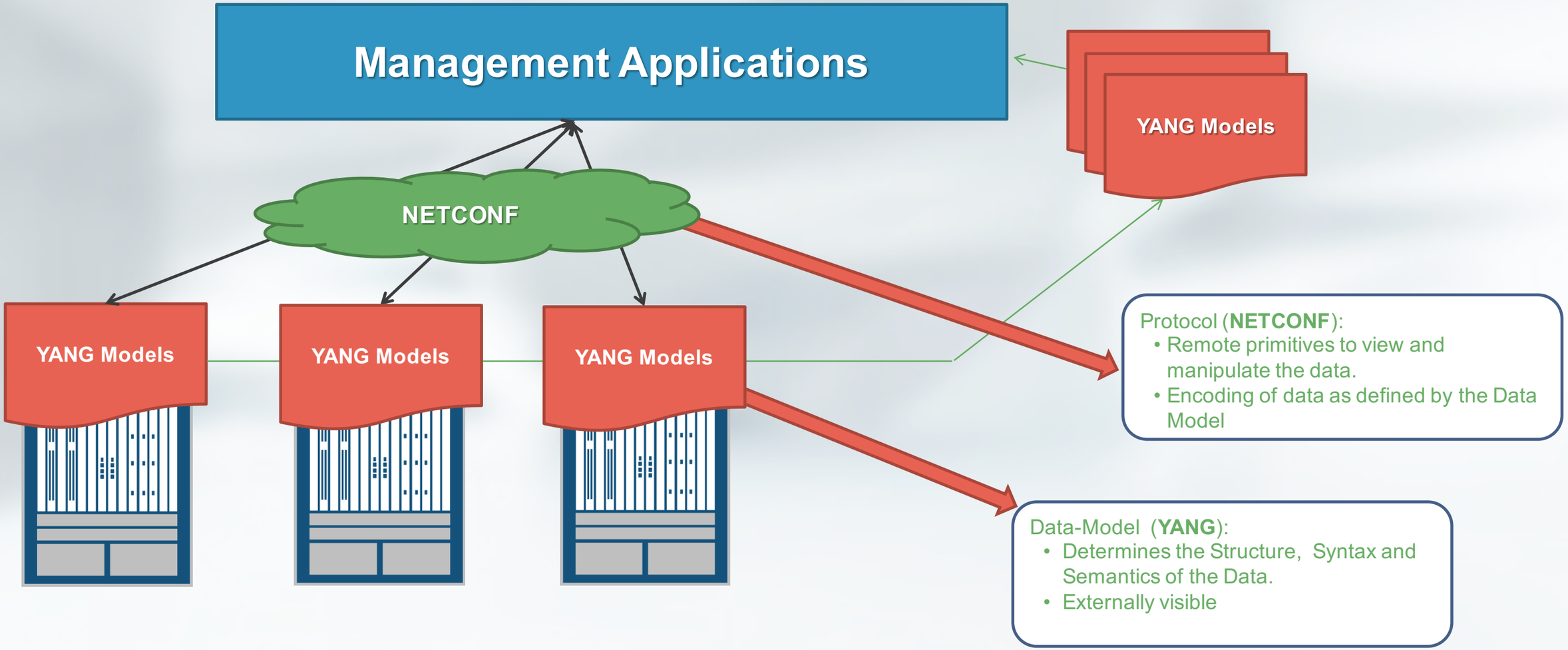
Junos Automation stack



Junos Platform Automation Stack

NETCONF & YANG

NETCONF and YANG



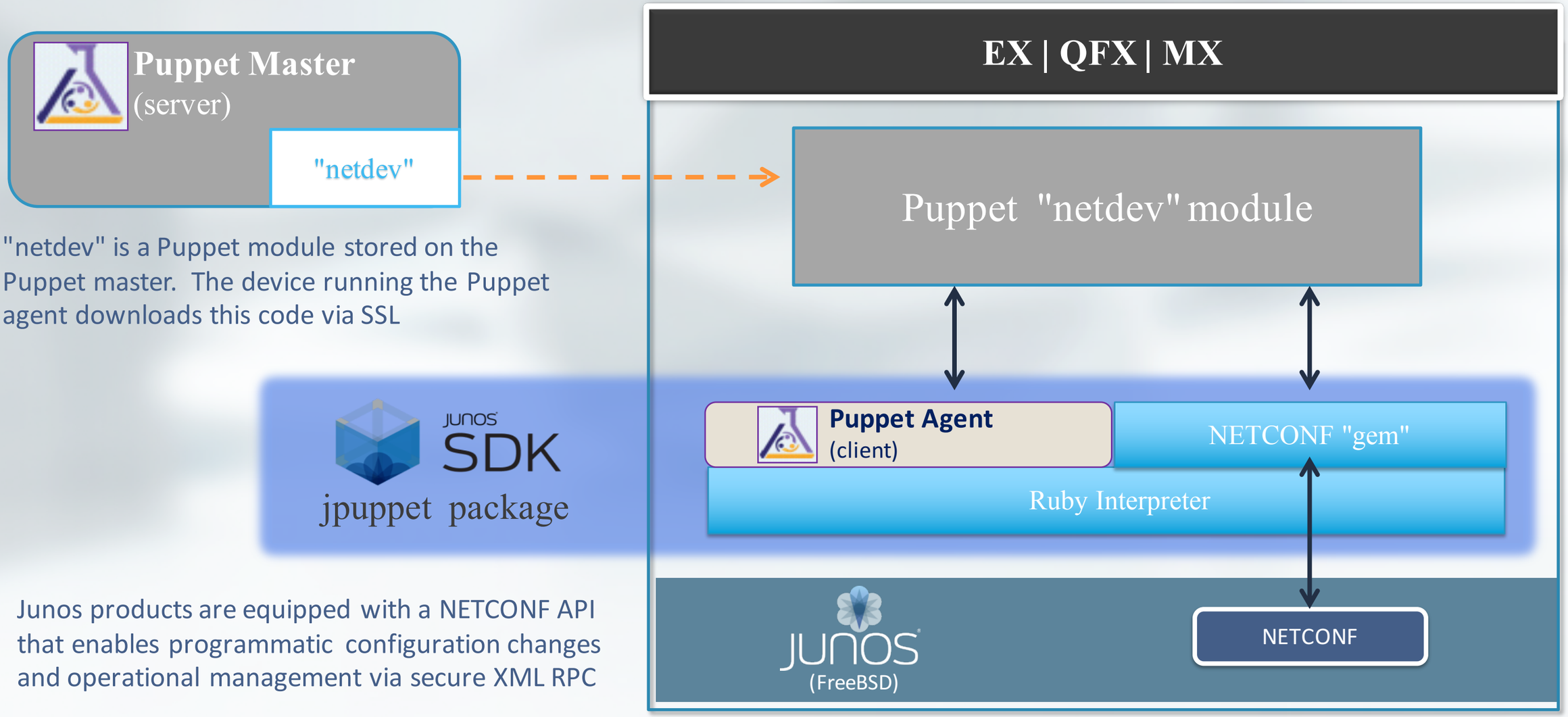
What does junos have

- NETCONF (RFC 6241)
  - Releases - All Junos releases support NETCONF
  - Platforms - All Junos Platforms support NETCONF
  - As the NETCONF RFC is amended, we keep adding the support on Junos

- YANG (RFC 6020)
  - Releases
    - 14.2 (partial support - configuration schema in yang)
    - 15.2+ (full support - configuration and operational schema in yang)
  - Platforms - All

IT Frameworks

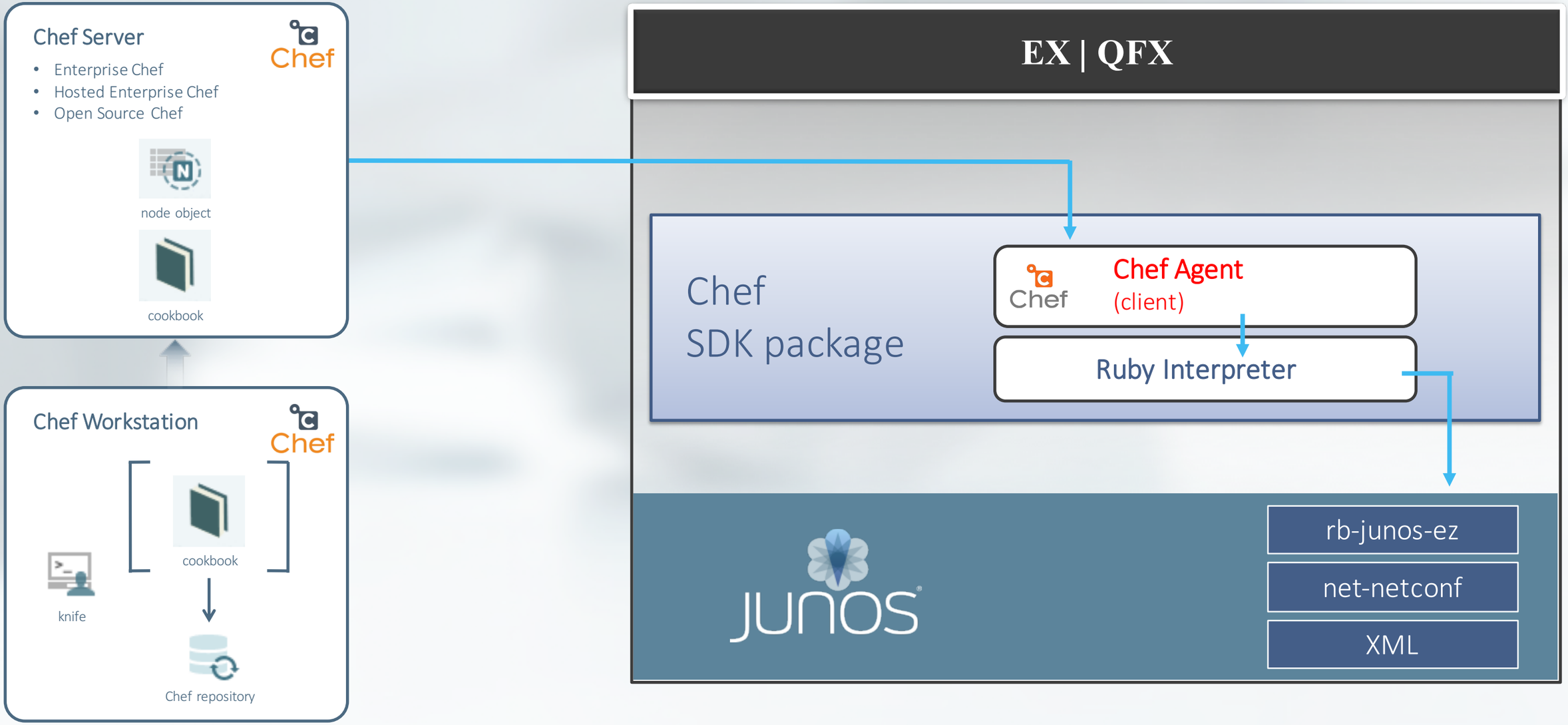
PUPPET



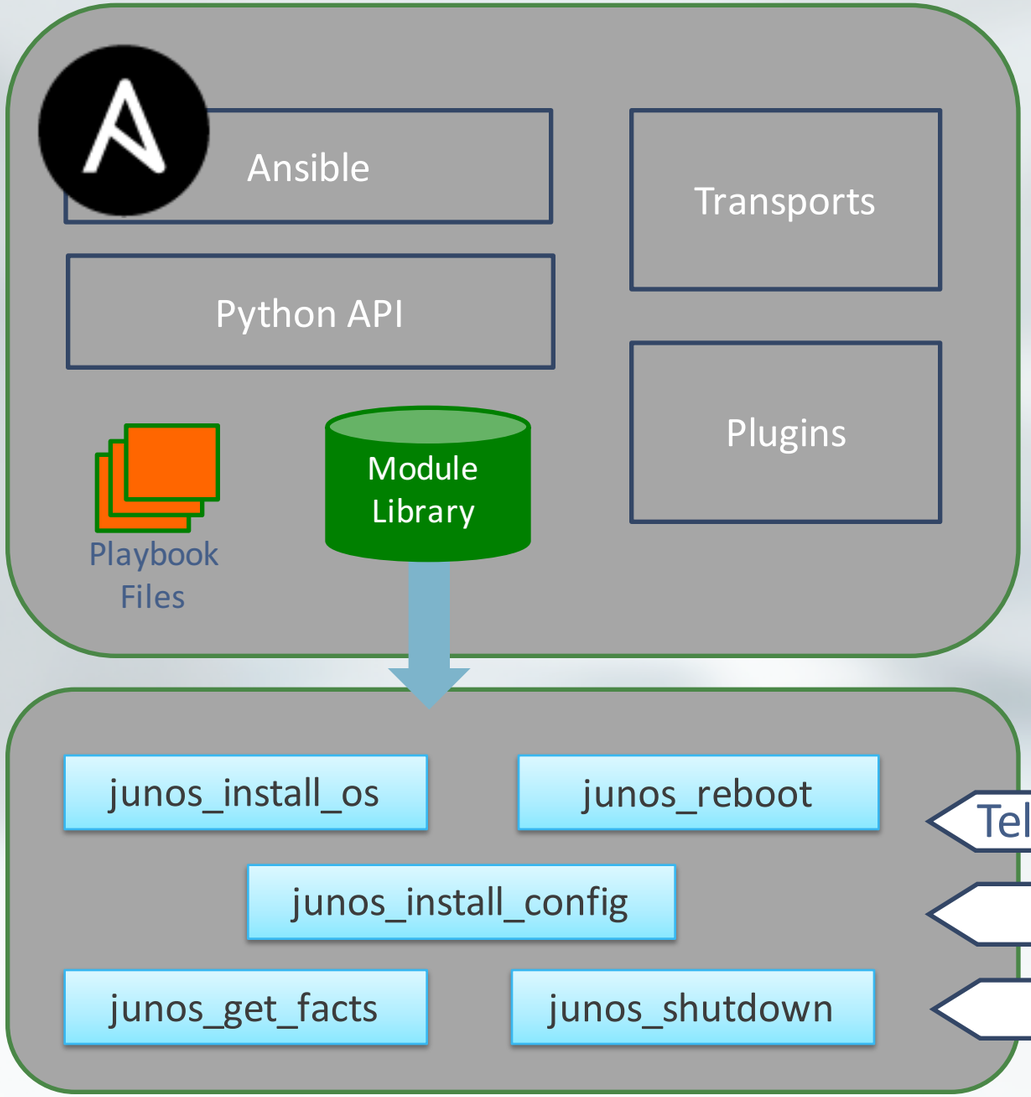
"netdev" is a Puppet module stored on the Puppet master. The device running the Puppet agent downloads this code via SSL

Junos products are equipped with a NETCONF API that enables programmatic configuration changes and operational management via secure XML RPC

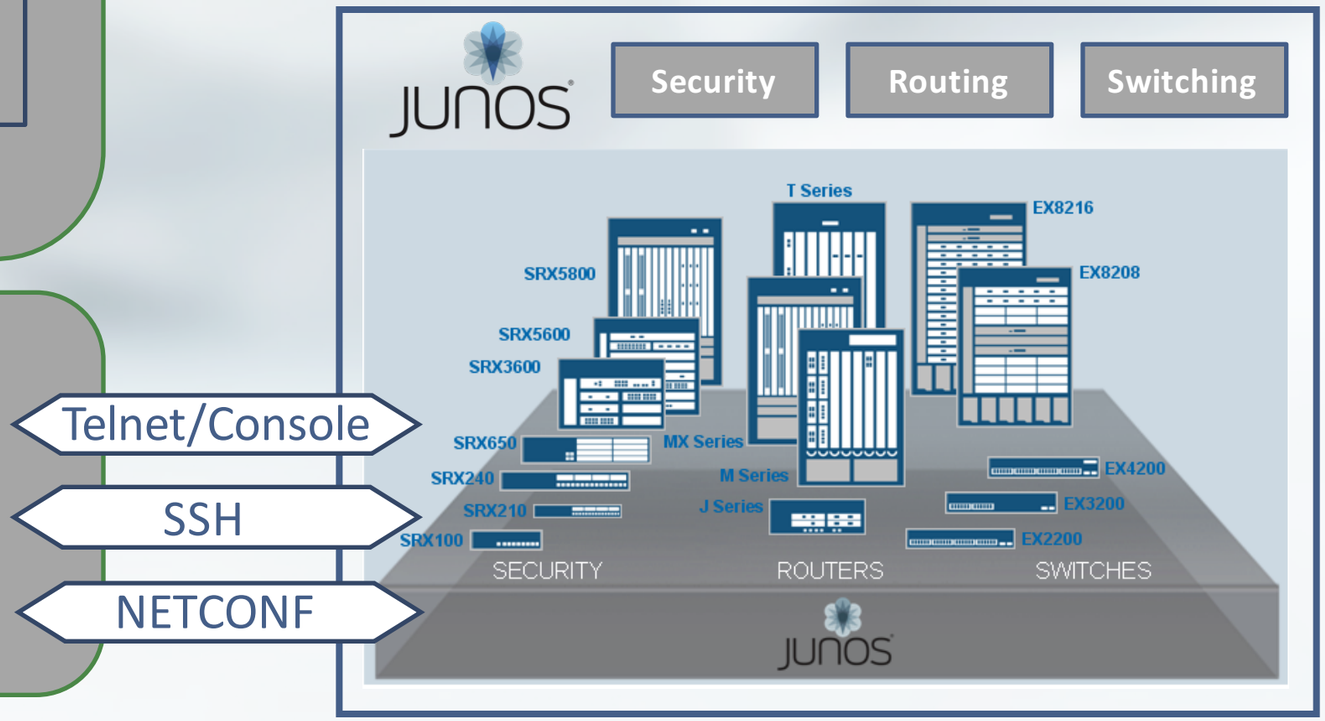
CHEF



Ansible



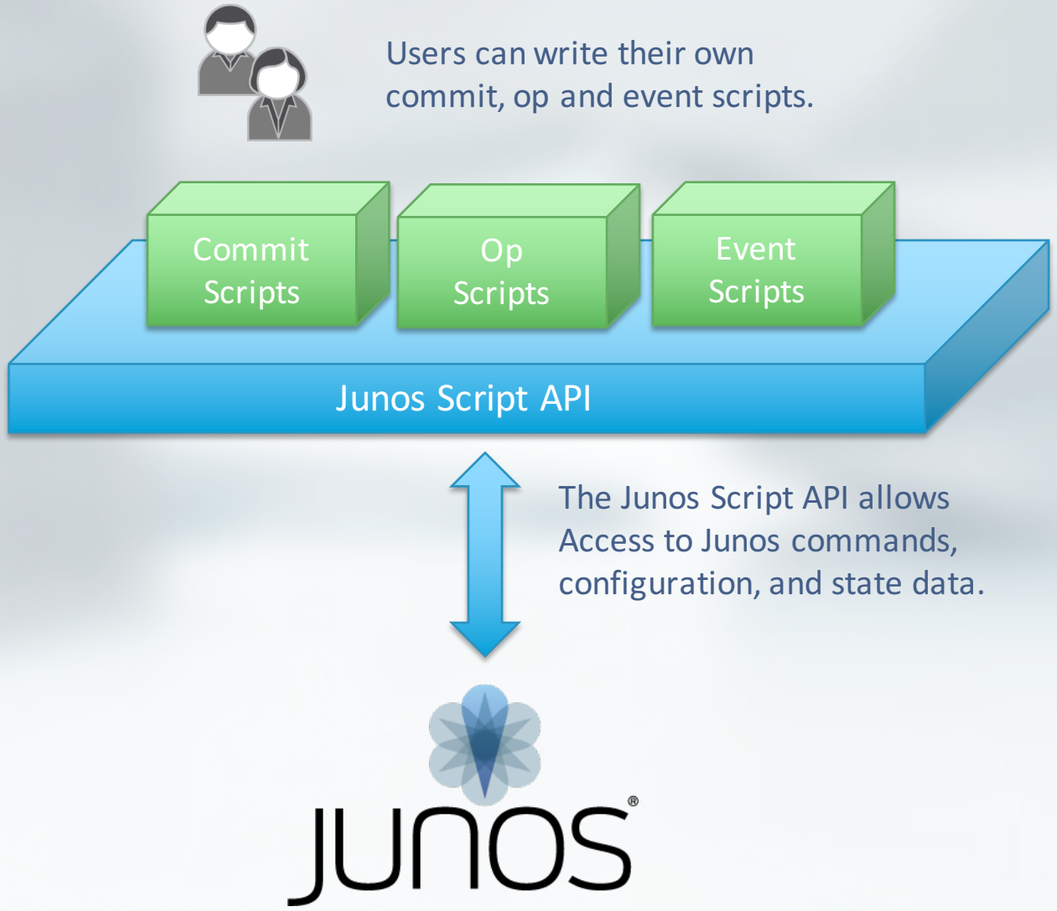
- Agentless and simple approach
- Does not require coding skills (YAML)
- Work flow Engine



Building Blocks

ON BOX AUTOMATION

Think globally, automate locally

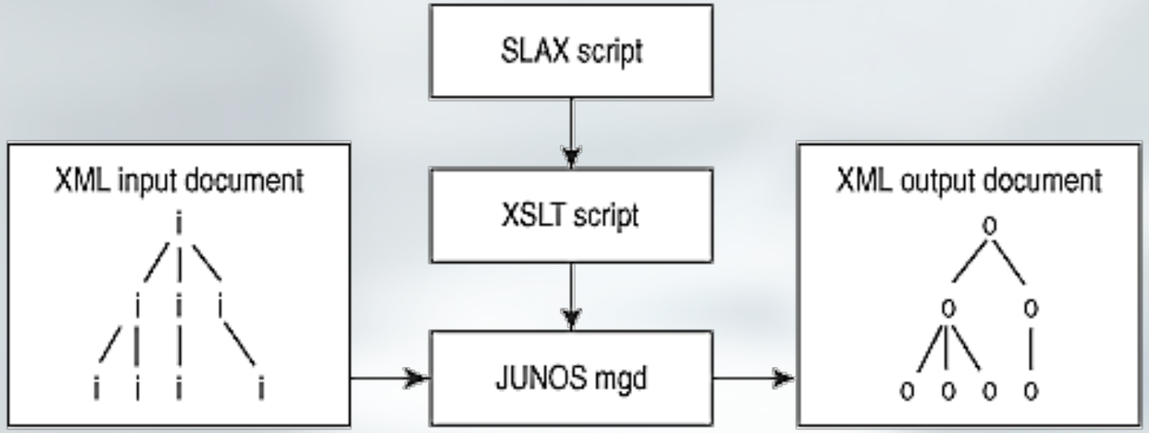


Commit Script
Run every time a user commits the configuration, can help with automation and consistency

Op Script
Initiated by an operator, help in troubleshooting, configuration, monitoring

Event Script
Initiated by an event policy and allow automation and troubleshooting

SLAX



- Supports both On and Off-box automation
- Syntax overlay for XSLT programming language
- On-box scripting for op, event and commit
- Off-box scripting for op and pre-emptive commit

PyEZ

- Python framework with easy learning curve
- Works with any Junos device running 11.4 or later
- Operational and Configuration Data/Management
- Generalized utilities for file-system, software-upgrade, scp
- Community supported

So why is it called “PyEZ” anyway?

netconf_example.py | ncclient_example.py | pyez.py | temp.py

```python
import paramiko
import socket
import time
import sys

ssh = paramiko.SSHClient()
ssh.set_missing_host_key_policy(
    paramiko.AutoAddPolicy())

CLOSE = """
<rpc>
  <close-session/>
</rpc>"""

SOFT_ADD = """
<rpc>
   <get-software-information/>
</rpc>"""

socket = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
socket.connect(("10.10.11.129",830))

trans = paramiko.Transport(socket)
trans.connect(username="apiuser", password="juniper123")

#CREATE CHANNEL FOR DATA COMM
ch = trans.open_session()
name = ch.set_name('netconf')

#Invoke NETCONF
ch.invoke_subsystem('netconf')

#SEND COMMAND
ch.send(SOFT_ADD)

#Recieve data returned
data = ch.recv(2048)
while data:
   data = ch.recv(1024)
   print data,
   if data.find('</rpc-reply>') == 0:
     #We have reached the end of reply
     ch.send(CLOSE)

ch.close()
trans.close()
socket.close()
```

So why is it called “PyEZ” anyway?

Raw Python – 48 lines

netconf_example.py | ncclient_example.py | pyez.py | temp.py

```python
import paramiko
import socket
import time
import sys

ssh = paramiko.SSHClient()
ssh.set_missing_host_key_policy(
    paramiko.AutoAddPolicy())

CLOSE = """
<rpc>
  <close-session/>
</rpc>"""

SOFT_ADD = """
<rpc>
   <get-software-information/>
</rpc>"""

socket = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
socket.connect(("10.10.11.129",830))

trans = paramiko.Transport(socket)
trans.connect(username="apiuser", password="juniper123")

#CREATE CHANNEL FOR DATA COMM
ch = trans.open_session()
name = ch.set_name('netconf')

#Invoke NETCONF
ch.invoke_subsystem('netconf')

#SEND COMMAND
ch.send(SOFT_ADD)

#Recieve data returned
data = ch.recv(2048)
while data:
   data = ch.recv(1024)
   print data,
   if data.find('</rpc-reply>') == 0:
     #We have reached the end of reply
     ch.send(CLOSE)

ch.close()
trans.close()
socket.close()
```

netconf_example.py | ncclient_example.py | pyez.py | temp.py

```python
from ncclient import manager

def connect(host, port, user, password):
    conn = manager.connect(host=host,
            port=port,
            username=user,
            password=password,
            timeout=10,
            device_params = {'name':'junos'},
            hostkey_verify=False)

    print 'show version'
    print '*' * 30
    result = conn.command('show version', format='text')
    print result.xpath('output')[0].text

if __name__ == '__main__':
    connect('10.10.11.129', '22', 'apiuser', 'juniper123')
```

So why is it called “PyEZ” anyway?

Raw Python – 48 lines

netconf_example.py | ncclient_example.py | pyez.py | temp.py

```python
import paramiko
import socket
import time
import sys

ssh = paramiko.SSHClient()
ssh.set_missing_host_key_policy(
    paramiko.AutoAddPolicy())

CLOSE = """
<rpc>
  <close-session/>
</rpc>"""

SOFT_ADD = """
<rpc>
   <get-software-information/>
</rpc>"""

socket = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
socket.connect(("10.10.11.129",830))

trans = paramiko.Transport(socket)
trans.connect(username="apiuser", password="juniper123")

#CREATE CHANNEL FOR DATA COMM
ch = trans.open_session()
name = ch.set_name('netconf')

#Invoke NETCONF
ch.invoke_subsystem('netconf')

#SEND COMMAND
ch.send(SOFT_ADD)

#Recieve data returned
data = ch.recv(2048)
while data:
   data = ch.recv(1024)
   print data,
   if data.find('</rpc-reply>') == 0:
     #We have reached the end of reply
     ch.send(CLOSE)

ch.close()
trans.close()
socket.close()
```

ncclient – 19 lines

netconf_example.py | ncclient_example.py | pyez.py | temp.py

```python
from ncclient import manager

def connect(host, port, user, password):
    conn = manager.connect(host=host,
            port=port,
            username=user,
            password=password,
            timeout=10,
            device_params = {'name':'junos'},
            hostkey_verify=False)

    print 'show version'
    print '*' * 30
    result = conn.command('show version', format='text')
    print result.xpath('output')[0].text

if __name__ == '__main__':
    connect('10.10.11.129', '22', 'apiuser', 'juniper123')
```

So why is it called “PyEZ” anyway?

Raw Python – 48 lines

netconf_example.py | ncclient_example.py | pyez.py | temp.py

```python
import paramiko
import socket
import time
import sys

ssh = paramiko.SSHClient()
ssh.set_missing_host_key_policy(
    paramiko.AutoAddPolicy())

CLOSE = """
<rpc>
  <close-session/>
</rpc>"""

SOFT_ADD = """
<rpc>
   <get-software-information/>
</rpc>"""

socket = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
socket.connect(("10.10.11.129",830))

trans = paramiko.Transport(socket)
trans.connect(username="apiuser", password="juniper123")

#CREATE CHANNEL FOR DATA COMM
ch = trans.open_session()
name = ch.set_name('netconf')

#Invoke NETCONF
ch.invoke_subsystem('netconf')

#SEND COMMAND
ch.send(SOFT_ADD)

#Recieve data returned
data = ch.recv(2048)
while data:
   data = ch.recv(1024)
   print data,
   if data.find('</rpc-reply>') == 0:
     #We have reached the end of reply
     ch.send(CLOSE)

ch.close()
trans.close()
socket.close()
```

PyEZ – 7 lines (and gathers more info)

netconf_example.py | ncclient_example.py | pyez.py | temp.py

```python
from jnpr.junos import Device

dev = Device(host='10.10.11.129', user='apiuser', password='juniper123' )
dev.open()
print(dev.facts)
dev.close()
```

ncclient – 19 lines

netconf_example.py | ncclient_example.py | pyez.py | temp.py

```python
from ncclient import manager

def connect(host, port, user, password):
    conn = manager.connect(host=host,
            port=port,
            username=user,
            password=password,
            timeout=10,
            device_params = {'name':'junos'},
            hostkey_verify=False)

    print 'show version'
    print '*' * 30
    result = conn.command('show version', format='text')
    print result.xpath('output')[0].text

if __name__ == '__main__':
    connect('10.10.11.129', '22', 'apiuser', 'juniper123')
```

All 3 scripts gather ‘show version’ info from the same device

YAML & Jinja2

- Yet Another Markup Language
- simplified language structure
- Ansible Playbooks
  - Name
    - The name of the running playbook
  - Hosts
    - Hosts to apply the tasks to
  - Tasks
    - Tasks to apply to the hosts
  - (Optionally) Variables
    - Variables allow for the customization of a running task
- # ansible-galaxy install Juniper.junos

- Jinja2 is a templating language for Python
- sticking to Python's principles
- Example:
  - Loop in Jinja2 template
    - {% for domain in domains %}"{{ domain }}",{% endfor %}

Examples

SLAX – Check Backbone MTU

sievers@gemini: ~/JAUT

Router — sievers@gemini: ~/JAUT/juise 117x19

```
psievers@R01# show ge-0/0/1
description "CORE to R02 ge-0/0/1";
unit 0 {
    description "core to R02 ge-0/0/1.0";
    family inet {
        address 172.27.0.1/30;
    }
}

[edit interfaces]
psievers@R01# set ge-0/0/1 mtu 1900

[edit interfaces]
psievers@R01# commit
warning: MTU on backbone interfacege-0/0/1.0 is not set to 2000 (1900)
commit complete

[edit interfaces]
psievers@R01#
```

backbone-mtu.slax

```
version 1.0;

ns junos = "http://xml.juniper.net/junos/*/junos";
ns xnm = "http://xml.juniper.net/xnm/1.1/xnm";
ns jcs = "http://xml.juniper.net/junos/commit-scripts/1.0";
ns ext = "http://xmlsoft.org/XSLT/namespace";

import "../import/junos.xsl";

/*
 * This commit script checks that any interfaces with "CORE" or "core"
 * in the description is configured with an MTU of 2000.
 * The search criteria and MTU should be changed to fit the environment
 * and requirements.
 */

match configuration {
        for-each (interfaces/interface) {
                var $int = name;
                var $unit = unit/name;
                var $desc = unit/description;
                var $mtu = mtu;
                if (contains($desc, "CORE") || contains($desc, "core")) {
                        if (not ($mtu == 2000)) {
                                <xnm:warning> {
                                        <message> {
                                                expr "MTU on backbone interface";
                                                expr $int;
                                                expr ".";
                                                expr $unit;
                                                expr " is not set to 2000";
                                                expr " (";
                                                expr $mtu;
                                                expr ")";
                                        }
                                }
                        }
                }
        }
```

Line 20, Column 39 | Spaces: 8 | Plain Text

Ansible – Deploy Playbook

deploy.yml

```yaml
---
### ------------------------------------------------------------------
### deploy.yml
### ------------------------------------------------------------------
###
###   1. check reachability via NETCONF
###   2. upgrade Junos OS if necessary; reboot & wait for restart
###   3. template build device specific Junos configuration
###   4. load/override the Junos configuration file
###
### ------------------------------------------------------------------

### ------------------------------------------------------------------
###    First ensure the hosts are reachable via the NETCONF protocol
### ------------------------------------------------------------------

- include: junos/nc_ready.yml timeout=1

### ------------------------------------------------------------------
### Install Junos OS on devices that need it
### ------------------------------------------------------------------

- include: junos/install_os.yml

### ------------------------------------------------------------------
### Now generate the target specific junos.conf initial config
### and install it on the target
### ------------------------------------------------------------------

- include: config/make.yml
- include: config/install.yml
```

Ansible – Playbook to load Junos Software

install_os.yml

```yaml
### ------------------------------------------------------------------
### Install Junos OS image as specified by the host's
### junos_os_tag variable
### ------------------------------------------------------------------

- hosts: all
  name: Junos OS image installation
  connection: local
  gather_facts: no
  vars_files:
    - /usr/local/junos/packages/catalog.yml

  tasks:
    - name: Junos OS install, please wait, this could take a bit ...
      junos_install_os: >
        host={{ inventory_hostname }}
        version={{ PACKAGE_TAGS[junos_os_tag].version }}
        package={{ PACKAGE_DIR }}/{{ PACKAGE_TAGS[junos_os_tag].package }}
        reboot=True
      notify:
        - Junos reboot
        - Junos wait for restart

  handlers:
    - name: Junos reboot
      pause: seconds={{ reboot_wait_time }}
    - name: Junos wait for restart
      wait_for: host={{ inventory_hostname }} port=830
```

`junos_install_os` performs installation of Junos OS only if the device does not have it already installed

`handlers` only called if OS is changed

PyEZ – BGP Config Example

sievers@gemini: ~/JAUT/juise

Router — sievers@gemini: ~/JAUT/juise 73x25

```
psievers@R01#

[edit]
psievers@R01#

[edit]
psievers@R01#

[edit]
psievers@R01#

[edit]
psievers@R01#

[edit]
psievers@R01#

[edit]
psievers@R01#

[edit]
psievers@R01#

[edit]
psievers@R01#
```

Router — sievers@gemini: ~/JAUT/juise 74x25

```
psievers@R02#

[edit]
psievers@R02#

[edit]
psievers@R02#

[edit]
psievers@R02#

[edit]
psievers@R02#

[edit]
psievers@R02#

[edit]
psievers@R02#

[edit]
psievers@R02#

[edit]
psievers@R02#
```

Router — sievers@gemini: ~/JAUT/juise 68x25

```
psievers@R03#

[edit]
psievers@R03#

[edit]
psievers@R03#

[edit]
psievers@R03#

[edit]
psievers@R03#

[edit]
psievers@R03#

[edit]
psievers@R03#

[edit]
psievers@R03#

[edit]
psievers@R03#
```

```
        }
    }
}
##
# R02 CONFIGURAITON
##
protocols {
    bgp {
        group iBGP {
            type internal;
            neighbor 172.27.255.1;
            neighbor 172.27.255.2;
            neighbor 172.27.255.3;
        }
    }
}
##
# R03 CONFIGURATION
##
protocols {
    bgp {
        group iBGP {
            type internal;
            neighbor 172.27.255.1;
            neighbor 172.27.255.2;
        }
    }
}
sievers@gemini:~/JAUT/community-NCE/bgp-full-mesh$ python2.7 bgp-full-mesh.py
```

File Edit Selection Find View Goto Tools Project Preferences Help

bgp-full-mesh.j2

```
##
# R01 CONFIGURAITON
##
protocols {
    bgp {
        group iBGP {
            type internal;
            neighbor {{ R02_lo0 }};
            neighbor {{ R03_lo0 }};
        }
    }
}
##
# R02 CONFIGURAITON
##
protocols {
    bgp {
        group iBGP {
            type internal;
{%- for neighbor in LoopBack %}
            neighbor {{ neighbor }};
{%- endfor %}
        }
    }
}
##
# R03 CONFIGURATION
##
protocols {
    bgp {
        group iBGP {
```

datavars.yml

```
---
R01_lo0: 172.27.255.1
R02_lo0: 172.27.255.2
R03_lo0: 172.27.255.3

LoopBack:
    - 172.27.255.1
    - 172.27.255.2
    - 172.27.255.3
```

Line 1, Column 1 | Tab Size: 4 | YAML

PyEZ – NETCONF Session

sievers@gemini: ~/JAUT/juise

Router — sievers@gemini: ~/JAUT/juise 73x25

```
ritable-running:1.0 capability capability urn:ietf:params:netconf:capabil
ity:rollback-on-error:1.0 capability capability urn:ietf:params:netconf:c
apability:validate:1.0 capability capability urn:ietf:params:netconf:capa
bility:confirmed-commit:1.0 capability capability "urn:ietf:params:netcon
f:capability:url:1.0?scheme=http,ftp,file,https,sftp" capability capabili
ty urn:ietf:params:netconf:base:1.0 capability capability urn:liberouter:
params:netconf:capability:power-control:1.0 capability capability urn:iet
f:params:netconf:capability:candidate:1.0 capability capability urn:ietf:
params:netconf:capability:xpath:1.0 capability capability urn:ietf:params
:netconf:capability:startup:1.0 capability capability urn:ietf:params:net
conf:capability:interleave:1.0 capability capabilities hello rpc get-chas
sis-inventory rpc rpc get-virtual-chassis-information rpc rpc ge
Oct 27 18:43:37  R01 file[3185]: UI_NETCONF_CMD: User 'ansible' used NETC
ONF client to run command 'all-routing-engines'
Oct 27 18:43:37  R01 file[3185]: UI_NETCONF_CMD: User 'ansible' used NETC
ONF client to run command 'get-software-information'
Oct 27 18:43:37  R01 file[3185]: UI_NETCONF_CMD: User 'ansible' used NETC
ONF client to run command 'get-configuration inherit="inherit"'
Oct 27 18:43:38  R01 file[3185]: UI_CMDLINE_READ_LINE: User 'ansible', co
mmand 'command rpc rpc get-configuration configuration system domain-name
 system configuration get-configuration rpc rpc command show cli director
y '
Oct 27 18:43:38  R01 file[3185]: UI_NETCONF_CMD: User 'ansible' used NETC
ONF client to run command 'show cli directory'
```

Router — sievers@gemini: ~/JAUT/juise 76x25

```
psievers@R02>
psievers@R02>
psievers@R02>
psievers@R02>
psievers@R02>
psievers@R02>
psievers@R02>
psievers@R02>
psievers@R02>
psievers@R02>
psievers@R02>
psievers@R02> monitor start interactive-commands
psievers@R02>
```

Router — sievers@gemini: ~/JAUT/juise 67x25

```
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03> monitor start interactive-commands
psievers@R03>
```

```
sievers@gemini:~/JAUT/community-NCE/bgp-full-mesh$ python2.7
Python 2.7.9 (default, Mar  1 2015, 12:57:24)
[GCC 4.9.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from jnpr.junos import Device
>>> dev = Device(host='r01', user='ansible', password='Ansible01')
>>> dev.open()
Device(192.168.178.231)
>>> dev.facts['version']
'12.1X47-D20.7'
>>>
```

File Edit Selection Find View Goto Tools Project Preferences Help

bgp-full-mesh.j2

```
##
# R01 CONFIGURAITON
##
protocols {
    bgp {
        group iBGP {
            type internal;
            neighbor {{ R02_lo0 }};
            neighbor {{ R03_lo0 }};
        }
    }
}
##
# R02 CONFIGURAITON
##
protocols {
    bgp {
        group iBGP {
            type internal;
{%- for neighbor in LoopBack %}
            neighbor {{ neighbor }};
{%- endfor %}
        }
    }
}
##
# R03 CONFIGURATION
##
protocols {
    bgp {
        group iBGP {
```

datavars.yml

```
---
R01_lo0: 172.27.255.1
R02_lo0: 172.27.255.2
R03_lo0: 172.27.255.3

LoopBack:
    - 172.27.255.1
    - 172.27.255.2
    - 172.27.255.3
```

Line 1, Column 1 | Tab Size: 4 | YAML

PyEZ – Load YAML File

sievers@gemini: ~/JAUT/juise

Router — sievers@gemini: ~/JAUT/juise 73x25

```
ritable-running:1.0 capability capability urn:ietf:params:netconf:capabil
ity:rollback-on-error:1.0 capability capability urn:ietf:params:netconf:c
apability:validate:1.0 capability capability urn:ietf:params:netconf:capa
bility:confirmed-commit:1.0 capability capability "urn:ietf:params:netcon
f:capability:url:1.0?scheme=http,ftp,file,https,sftp" capability capabili
ty urn:ietf:params:netconf:base:1.0 capability capability urn:liberouter:
params:netconf:capability:power-control:1.0 capability capability urn:iet
f:params:netconf:capability:candidate:1.0 capability capability urn:ietf:
params:netconf:capability:xpath:1.0 capability capability urn:ietf:params
:netconf:capability:startup:1.0 capability capability urn:ietf:params:net
conf:capability:interleave:1.0 capability capabilities hello rpc get-chas
sis-inventory rpc rpc get-virtual-chassis-information rpc rpc ge
Oct 27 18:43:37  R01 file[3185]: UI_NETCONF_CMD: User 'ansible' used NETC
ONF client to run command 'all-routing-engines'
Oct 27 18:43:37  R01 file[3185]: UI_NETCONF_CMD: User 'ansible' used NETC
ONF client to run command 'get-software-information'
Oct 27 18:43:37  R01 file[3185]: UI_NETCONF_CMD: User 'ansible' used NETC
ONF client to run command 'get-configuration inherit="inherit"'
Oct 27 18:43:38  R01 file[3185]: UI_CMDLINE_READ_LINE: User 'ansible', co
mmand 'command rpc rpc get-configuration configuration system domain-name
 system configuration get-configuration rpc rpc command show cli director
y '
Oct 27 18:43:38  R01 file[3185]: UI_NETCONF_CMD: User 'ansible' used NETC
ONF client to run command 'show cli directory'
```

Router — sievers@gemini: ~/JAUT/juise 76x25

```
psievers@R02>
psievers@R02>
psievers@R02>
psievers@R02>
psievers@R02>
psievers@R02>
psievers@R02>
psievers@R02>
psievers@R02>
psievers@R02>
psievers@R02>
psievers@R02>
psievers@R02> monitor start interactive-commands
psievers@R02>
```

Router — sievers@gemini: ~/JAUT/juise 67x25

```
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03> monitor start interactive-commands
psievers@R03>
```

```
sievers@gemini:~/JAUT/community-NCE/bgp-full-mesh$ python2.7
Python 2.7.9 (default, Mar  1 2015, 12:57:24)
[GCC 4.9.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from jnpr.junos import Device
>>> dev = Device(host='r01', user='ansible', password='Ansible01')
>>> dev.open()
Device(192.168.178.231)
>>> dev.facts['version']
'12.1X47-D20.7'
>>> import yaml
>>> mydata = yaml.load(open('datavars.yml').read())
>>> from pprint import pprint as pp
>>> pp(mydata)
{'LoopBack': ['172.27.255.1', '172.27.255.2', '172.27.255.3'],
 'R01_lo0': '172.27.255.1',
 'R02_lo0': '172.27.255.2',
 'R03_lo0': '172.27.255.3'}
>>>
```

File Edit Selection Find View Goto Tools Project Preferences Help

bgp-full-mesh.j2

```
1  ##
2  # R01 CONFIGURAITON
3  ##
4  protocols {
5      bgp {
6          group iBGP {
7              type internal;
8              neighbor {{ R02_lo0 }};
9              neighbor {{ R03_lo0 }};
10         }
11     }
12 }
13 ##
14 # R02 CONFIGURAITON
15 ##
16 protocols {
17     bgp {
18         group iBGP {
19             type internal;
20 {%- for neighbor in LoopBack %}
21             neighbor {{ neighbor }};
22 {%- endfor %}
23         }
24     }
25 }
26 ##
27 # R03 CONFIGURATION
28 ##
29 protocols {
30     bgp {
31         group iBGP {
```

datavars.yml

```
1  ---
2  R01_lo0: 172.27.255.1
3  R02_lo0: 172.27.255.2
4  R03_lo0: 172.27.255.3
5
6  LoopBack:
7      - 172.27.255.1
8      - 172.27.255.2
9      - 172.27.255.3
10
```

Line 1, Column 1 | Tab Size: 4 | YAML

PyEZ – Merge Jinja2 with YAML Template

sievers@gemini: ~/JAUT/juise

Router sievers@gemini: ~/JAUT/juise 73x25

```
bility:confirmed-commit:1.0 capability capability "urn:ietf:params:netcon
f:capability:url:1.0?scheme=http,ftp,file,https,sftp" capability capabili
ty urn:ietf:params:netconf:base:1.0 capability capability urn:liberouter:
params:netconf:capability:power-control:1.0 capability capability urn:iet
f:params:netconf:capability:candidate:1.0 capability capability urn:ietf:
params:netconf:capability:xpath:1.0 capability capability urn:ietf:params
:netconf:capability:startup:1.0 capability capability urn:ietf:params:net
conf:capability:interleave:1.0 capability capabilities hello rpc get-chas
sis-inventory rpc rpc get-virtual-chassis-information rpc rpc ge
Oct 27 18:43:37  R01 file[3185]: UI_NETCONF_CMD: User 'ansible' used NETC
ONF client to run command 'all-routing-engines'
Oct 27 18:43:37  R01 file[3185]: UI_NETCONF_CMD: User 'ansible' used NETC
ONF client to run command 'get-software-information'
Oct 27 18:43:37  R01 file[3185]: UI_NETCONF_CMD: User 'ansible' used NETC
ONF client to run command 'get-configuration inherit="inherit"'
Oct 27 18:43:38  R01 file[3185]: UI_CMDLINE_READ_LINE: User 'ansible', co
mmand 'command rpc rpc get-configuration configuration system domain-name
 system configuration get-configuration rpc rpc command show cli director
y '
Oct 27 18:43:38  R01 file[3185]: UI_NETCONF_CMD: User 'ansible' used NETC
ONF client to run command 'show cli directory'
Oct 27 18:53:55  R01 file[3185]: UI_NETCONF_CMD: User 'ansible' used NETC
ONF client to run command 'load-configuration action="replace" format="te
xt"'
```

Router sievers@gemini: ~/JAUT/juise 76x25

```
psievers@R02>
psievers@R02>
psievers@R02>
psievers@R02>
psievers@R02>
psievers@R02>
psievers@R02>
psievers@R02>
psievers@R02>
psievers@R02>
psievers@R02>
psievers@R02> monitor start interactive-commands
psievers@R02>
```

Router sievers@gemini: ~/JAUT/juise 67x25

```
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03> monitor start interactive-commands
psievers@R03>
```

```
sievers@gemini:~/JAUT/community-NCE/bgp-full-mesh$ python2.7
Python 2.7.9 (default, Mar  1 2015, 12:57:24)
[GCC 4.9.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from jnpr.junos import Device
>>> dev = Device(host='r01', user='ansible', password='Ansible01')
>>> dev.open()
Device(192.168.178.231)
>>> dev.facts['version']
'12.1X47-D20.7'
>>> import yaml
>>> mydata = yaml.load(open('datavars.yml').read())
>>> from pprint import pprint as pp
>>> pp(mydata)
{'LoopBack': ['172.27.255.1', '172.27.255.2', '172.27.255.3'],
 'R01_lo0': '172.27.255.1',
 'R02_lo0': '172.27.255.2',
 'R03_lo0': '172.27.255.3'}
>>> from jnpr.junos.utils.config import Config
>>> dev.bind(cfg=Config)
>>> dev.cfg
jnpr.junos.utils.Config(192.168.178.231)
>>> dev.cfg.load(template_path='bgp-full-mesh.j2', template_vars=mydata, format='text')
<Element load-configuration-results at 0x7f97bcb43320>
>>>
```

File Edit Selection Find View Goto Tools Project Preferences Help

bgp-full-mesh.j2

```
1  ##
2  # R01 CONFIGURAITON
3  ##
4  protocols {
5      bgp {
6          group iBGP {
7              type internal;
8              neighbor {{ R02_lo0 }};
9              neighbor {{ R03_lo0 }};
10         }
11     }
12 }
13 ##
14 # R02 CONFIGURAITON
15 ##
16 protocols {
17     bgp {
18         group iBGP {
19             type internal;
20 {%- for neighbor in LoopBack %}
21             neighbor {{ neighbor }};
22 {%- endfor %}
23         }
24     }
25 }
26 ##
27 # R03 CONFIGURATION
28 ##
29 protocols {
30     bgp {
31         group iBGP {
```

datavars.yml

```
1  ---
2  R01_lo0: 172.27.255.1
3  R02_lo0: 172.27.255.2
4  R03_lo0: 172.27.255.3
5
6  LoopBack:
7      - 172.27.255.1
8      - 172.27.255.2
9      - 172.27.255.3
10
```

Line 1, Column 1 | Tab Size: 4 | YAML

PyEZ – Diff

sievers@gemini: ~/JAUT/juise

Router sievers@gemini: ~/JAUT/juise 73x25

```
params:netconf:capability:power-control:1.0 capability capability urn:iet
f:params:netconf:capability:candidate:1.0 capability capability urn:ietf:
params:netconf:capability:xpath:1.0 capability capability urn:ietf:params
:netconf:capability:startup:1.0 capability capability urn:ietf:params:net
conf:capability:interleave:1.0 capability capabilities hello rpc get-chas
sis-inventory rpc rpc get-virtual-chassis-information rpc rpc ge
Oct 27 18:43:37  R01 file[3185]: UI_NETCONF_CMD: User 'ansible' used NETC
ONF client to run command 'all-routing-engines'
Oct 27 18:43:37  R01 file[3185]: UI_NETCONF_CMD: User 'ansible' used NETC
ONF client to run command 'get-software-information'
Oct 27 18:43:37  R01 file[3185]: UI_NETCONF_CMD: User 'ansible' used NETC
ONF client to run command 'get-configuration inherit="inherit"'
Oct 27 18:43:38  R01 file[3185]: UI_CMDLINE_READ_LINE: User 'ansible', co
mmand 'command rpc rpc get-configuration configuration system domain-name
 system configuration get-configuration rpc rpc command show cli director
y '
Oct 27 18:43:38  R01 file[3185]: UI_NETCONF_CMD: User 'ansible' used NETC
ONF client to run command 'show cli directory'
Oct 27 18:53:55  R01 file[3185]: UI_NETCONF_CMD: User 'ansible' used NETC
ONF client to run command 'load-configuration action="replace" format="te
xt"'
Oct 27 18:56:22  R01 file[3185]: UI_NETCONF_CMD: User 'ansible' used NETC
ONF client to run command 'get-configuration compare="rollback" rollback=
"0" format="text"'
```

Router sievers@gemini: ~/JAUT/juise 76x25

```
psievers@R02>
psievers@R02>
psievers@R02>
psievers@R02>
psievers@R02>
psievers@R02>
psievers@R02>
psievers@R02>
psievers@R02>
psievers@R02>
psievers@R02>
psievers@R02> monitor start interactive-commands
psievers@R02>
```

Router sievers@gemini: ~/JAUT/juise 67x25

```
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03> monitor start interactive-commands
psievers@R03>
```

```
>>> mydata = yaml.load(open('datavars.yml').read())
>>> from pprint import pprint as pp
>>> pp(mydata)
{'LoopBack': ['172.27.255.1', '172.27.255.2', '172.27.255.3'],
 'R01_lo0': '172.27.255.1',
 'R02_lo0': '172.27.255.2',
 'R03_lo0': '172.27.255.3'}
>>> from jnpr.junos.utils.config import Config
>>> dev.bind(cfg=Config)
>>> dev.cfg
jnpr.junos.utils.Config(192.168.178.231)
>>> dev.cfg.load(template_path='bgp-full-mesh.j2', template_vars=mydata, format='text')
<Element load-configuration-results at 0x7f97bcb43320>
>>> dev.cfg.pdiff()

[edit]
+  # R03 CONFIGURATION
   protocols { ... }
[edit protocols]
+   bgp {
+       group iBGP {
+           type internal;
+           neighbor 172.27.255.2;
+           neighbor 172.27.255.3;
+           neighbor 172.27.255.1;
+       }
+   }

>>>
```

File Edit Selection Find View Goto Tools Project Preferences Help

bgp-full-mesh.j2

```
1  ##
2  # R01 CONFIGURAITON
3  ##
4  protocols {
5      bgp {
6          group iBGP {
7              type internal;
8              neighbor {{ R02_lo0 }};
9              neighbor {{ R03_lo0 }};
10         }
11     }
12 }
13 ##
14 # R02 CONFIGURAITON
15 ##
16 protocols {
17     bgp {
18         group iBGP {
19             type internal;
20 {%- for neighbor in LoopBack %}
21             neighbor {{ neighbor }};
22 {%- endfor %}
23         }
24     }
25 }
26 ##
27 # R03 CONFIGURATION
28 ##
29 protocols {
30     bgp {
31         group iBGP {
```

datavars.yml

```
1  ---
2  R01_lo0: 172.27.255.1
3  R02_lo0: 172.27.255.2
4  R03_lo0: 172.27.255.3
5
6  LoopBack:
7      - 172.27.255.1
8      - 172.27.255.2
9      - 172.27.255.3
10
```

Line 1, Column 1 | Tab Size: 4 | YAML

PyEZ – Activate Configuration

sievers@gemini: ~/JAUT/juise

Router sievers@gemini: ~/JAUT/juise 73x25

```
progress: Multicast Snooping process checking new configuration
Oct 27 18:59:08  R01 file[3185]: UI_CHILD_START: Starting child '/usr/sbi
n/mcsnoopd'
Oct 27 18:59:08  R01 file[3185]: UI_CHILD_STATUS: Cleanup child '/usr/sbi
n/mcsnoopd', PID 3213, status 0
Oct 27 18:59:08  R01 file[3185]: UI_CHILD_START: Starting child '/usr/sbi
n/ffp'
Oct 27 18:59:08  R01 file[3185]: UI_CHILD_STATUS: Cleanup child '/usr/sbi
n/ffp', PID 3214, status 0
Oct 27 18:59:08  R01 file[3185]: UI_COMMIT_PROGRESS: Commit operation in
progress: signaling 'Alarm control process', pid 1165, signal 30, status
0 with notification errors enabled
Oct 27 18:59:24  R01 file[3185]: UI_NETCONF_CMD: User 'ansible' used NETC
ONF client to run command 'commit-configuration'
Oct 27 18:59:24  R01 file[3185]: UI_COMMIT: User 'ansible' requested 'com
mit' operation (comment: none)
Oct 27 18:59:24  R01 file[3185]: UI_COMMIT_PROGRESS: Commit operation in
progress: start loading commit script changes
Oct 27 18:59:24  R01 file[3185]: UI_COMMIT_PROGRESS: Commit operation in
progress: no commit script changes
Oct 27 18:59:24  R01 file[3185]: UI_COMMIT_PROGRESS: Commit operation in
progress: no transient commit script changes
Oct 27 18:59:24  R01 file[3185]: UI_COMMIT_PROGRESS: Commit operation in
progress: finished loading commit script changes
Oct 27 18:59:24  R01 file[3185]: UI_COMMIT_PROGRESS: Commit operation in
```

Router sievers@gemini: ~/JAUT/juise 76x25

```
psievers@R02>
psievers@R02>
psievers@R02>
psievers@R02>
psievers@R02>
psievers@R02>
psievers@R02>
psievers@R02>
psievers@R02>
psievers@R02>
psievers@R02>
psievers@R02> monitor start interactive-commands
psievers@R02>
```

Router sievers@gemini: ~/JAUT/juise 67x25

```
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03>
psievers@R03> monitor start interactive-commands
psievers@R03>
```

```
 'R01_lo0': '172.27.255.1',
 'R02_lo0': '172.27.255.2',
 'R03_lo0': '172.27.255.3'}
>>> from jnpr.junos.utils.config import Config
>>> dev.bind(cfg=Config)
>>> dev.cfg
jnpr.junos.utils.Config(192.168.178.231)
>>> dev.cfg.load(template_path='bgp-full-mesh.j2', template_vars=mydata, format='text')
<Element load-configuration-results at 0x7f97bcb43320>
>>> dev.cfg.pdiff()

[edit]
+  # R03 CONFIGURATION
   protocols { ... }
[edit protocols]
+   bgp {
+       group iBGP {
+           type internal;
+           neighbor 172.27.255.2;
+           neighbor 172.27.255.3;
+           neighbor 172.27.255.1;
+       }
+   }

>>> dev.cfg.commit_check()
True
>>> dev.cfg.commit()
True
>>>
```

File Edit Selection Find View Goto Tools Project Preferences Help

bgp-full-mesh.j2

```
1  ##
2  # R01 CONFIGURAITON
3  ##
4  protocols {
5      bgp {
6          group iBGP {
7              type internal;
8              neighbor {{ R02_lo0 }};
9              neighbor {{ R03_lo0 }};
10         }
11     }
12 }
13 ##
14 # R02 CONFIGURAITON
15 ##
16 protocols {
17     bgp {
18         group iBGP {
19             type internal;
20 {%- for neighbor in LoopBack %}
21             neighbor {{ neighbor }};
22 {%- endfor %}
23         }
24     }
25 }
26 ##
27 # R03 CONFIGURATION
28 ##
29 protocols {
30     bgp {
31         group iBGP {
```

datavars.yml

```
1  ---
2  R01_lo0: 172.27.255.1
3  R02_lo0: 172.27.255.2
4  R03_lo0: 172.27.255.3
5
6  LoopBack:
7      - 172.27.255.1
8      - 172.27.255.2
9      - 172.27.255.3
10
```

Line 1, Column 1 | Tab Size: 4 | YAML

PyEZ – Result & Help

sievers@gemini: ~/JAUT/juise

Router sievers@gemini: ~/JAUT/juise 73x25

```
progress: commit complete
Oct 27 18:59:27  R01 file[3185]: UI_COMMIT_PROGRESS: Commit operation in
progress: signaling 'Alarm control process', pid 1165, signal 30, status
0 with notification errors enabled


psievers@R01>

psievers@R01>

psievers@R01>

psievers@R01> show configuration protocols bgp
group iBGP {
    type internal;
    neighbor 172.27.255.2;
    neighbor 172.27.255.3;
    neighbor 172.27.255.1;
}

psievers@R01> Oct 27 19:04:30  R01 mgd[2034]: UI_CMDLINE_READ_LINE: User
'psievers', command 'show configuration protocols bgp '


psievers@R01>
```

Router sievers@gemini: ~/JAUT/juise 76x25

```
psievers@R02>

psievers@R02>

psievers@R02>

psievers@R02> monitor start interactive-commands

psievers@R02>

psievers@R02>

psievers@R02>

psievers@R02>

psievers@R02> show configuration protocols bgp

psievers@R02>
*** interactive-commands ***
Oct 27 19:04:30  R02 mgd[1868]: UI_CMDLINE_READ_LINE: User 'psievers', comma
nd 'show configuration protocols bgp '


psievers@R02>
```

Router sievers@gemini: ~/JAUT/juise 67x25

```
psievers@R03>

psievers@R03>

psievers@R03>

psievers@R03> monitor start interactive-commands

psievers@R03>

psievers@R03>

psievers@R03>

psievers@R03>

psievers@R03> show configuration protocols bgp

psievers@R03>
*** interactive-commands ***
Oct 27 19:04:30  R03 mgd[1724]: UI_CMDLINE_READ_LINE: User 'psiever
s', command 'show configuration protocols bgp '


psievers@R03>
```

```
Help on Config in module jnpr.junos.utils.config object:

class Config(jnpr.junos.utils.util.Util)
 |  Overivew of Configuration Utilities:
 |
 |  * :meth:`commit`: commit changes
 |  * :meth:`commit_check`: perform the commit check operation
 |  * :meth:`diff`: return the diff string between running and candidate config
 |  * :meth:`load`: load changes into the candidate config
 |  * :meth:`lock`: take an exclusive lock on the candidate config
 |  * :meth:`pdiff`: prints the diff string (debug/helper)
 |  * :meth:`rescue`: controls "rescue configuration"
 |  * :meth:`rollback`: perform the load rollback command
 |  * :meth:`unlock`: release the exclusive lock
 |
 |  Method resolution order:
 |      Config
 |      jnpr.junos.utils.util.Util
 |      __builtin__.object
 |
 |  Methods defined here:
 |
 |  commit(self, **kvargs)
 |      Commit a configuration.
 |
 |      :param str comment: If provided logs this comment with the commit.
 |      :param int confirm: If provided activates confirm safeguard with
 |                          provided value as timeout (minutes).
:
```

File Edit Selection Find View Goto Tools Project Preferences Help

bgp-full-mesh.j2

```
##
# R01 CONFIGURAITON
##
protocols {
    bgp {
        group iBGP {
            type internal;
            neighbor {{ R02_lo0 }};
            neighbor {{ R03_lo0 }};
        }
    }
}
##
# R02 CONFIGURAITON
##
protocols {
    bgp {
        group iBGP {
            type internal;
{%- for neighbor in LoopBack %}
            neighbor {{ neighbor }};
{%- endfor %}
        }
    }
}
##
# R03 CONFIGURATION
##
protocols {
    bgp {
        group iBGP {
```

datavars.yml

```
---
R01_lo0: 172.27.255.1
R02_lo0: 172.27.255.2
R03_lo0: 172.27.255.3

LoopBack:
    - 172.27.255.1
    - 172.27.255.2
    - 172.27.255.3
```

Line 1, Column 1 | Tab Size: 4 | YAML

Ansible – get facts

sievers@gemini: ~/JAUT/juise

Router — sievers@gemini: ~/JAUT/juise 77x19

```
bility capability urn:ietf:params:netconf:capability:startup:1.0 capability c
apability urn:ietf:params:netconf:capability:interleave:1.0 capability capabi
lities hello rpc get-chassis-inventory rpc rpc get-virtual-chassis-informatio
n rpc rpc get
Oct 28 11:17:38  R01 file[6345]: UI_NETCONF_CMD: User 'ansible' used NETCONF
client to run command 'all-routing-engines'
Oct 28 11:17:38  R01 file[6345]: UI_NETCONF_CMD: User 'ansible' used NETCONF
client to run command 'get-software-information'
Oct 28 11:17:38  R01 file[6345]: UI_NETCONF_CMD: User 'ansible' used NETCONF
client to run command 'get-configuration inherit="inherit"'
Oct 28 11:17:38  R01 file[6345]: UI_CMDLINE_READ_LINE: User 'ansible', comman
d 'command rpc rpc get-configuration configuration system domain-name system
configuration get-configuration rpc rpc command show cli directory '
Oct 28 11:17:38  R01 file[6345]: UI_NETCONF_CMD: User 'ansible' used NETCONF
client to run command 'show cli directory'
Oct 28 11:17:38  R01 file[6345]: UI_NETCONF_CMD: User 'ansible' used NETCONF
client to run command 'close-session'
Oct 28 11:17:38  R01 file[6345]: UI_LOGOUT_EVENT: User 'ansible' logout
```

Router — sievers@gemini: ~/JAUT/juise 75x19

```
ility:xpath:1.0 capability capability urn:ietf:params:netconf:capability:st
artup:1.0 capability capability urn:ietf:params:netconf:capability:interlea
ve:1.0 capability capabilities hello rpc get-chassis-inventory rpc rpc get-
virtual-chassis-information rpc rpc ge
Oct 28 11:17:38  R02 file[4091]: UI_NETCONF_CMD: User 'ansible' used NETCON
F client to run command 'all-routing-engines'
Oct 28 11:17:38  R02 file[4091]: UI_NETCONF_CMD: User 'ansible' used NETCON
F client to run command 'get-software-information'
Oct 28 11:17:38  R02 file[4091]: UI_NETCONF_CMD: User 'ansible' used NETCON
F client to run command 'get-configuration inherit="inherit"'
Oct 28 11:17:38  R02 file[4091]: UI_CMDLINE_READ_LINE: User 'ansible', comm
and 'command rpc rpc get-configuration configuration system domain-name sys
tem configuration get-configuration rpc rpc command show cli directory '
Oct 28 11:17:38  R02 file[4091]: UI_NETCONF_CMD: User 'ansible' used NETCON
F client to run command 'show cli directory'
Oct 28 11:17:38  R02 file[4091]: UI_NETCONF_CMD: User 'ansible' used NETCON
F client to run command 'close-session'
Oct 28 11:17:38  R02 file[4091]: UI_LOGOUT_EVENT: User 'ansible' logout
```

Router — sievers@gemini: ~/JAUT/juise 66x19

```
-information rpc rpc ge
Oct 28 11:17:38  R03 file[3689]: UI_NETCONF_CMD: User 'ansible' us
ed NETCONF client to run command 'all-routing-engines'
Oct 28 11:17:38  R03 file[3689]: UI_NETCONF_CMD: User 'ansible' us
ed NETCONF client to run command 'get-software-information'
Oct 28 11:17:38  R03 file[3689]: UI_NETCONF_CMD: User 'ansible' us
ed NETCONF client to run command 'get-configuration inherit="inher
it"'
Oct 28 11:17:38  R03 file[3689]: UI_CMDLINE_READ_LINE: User 'ansib
le', command 'command rpc rpc get-configuration configuration syst
em domain-name system configuration get-configuration rpc rpc comm
and show cli directory '
Oct 28 11:17:38  R03 file[3689]: UI_NETCONF_CMD: User 'ansible' us
ed NETCONF client to run command 'show cli directory'
Oct 28 11:17:38  R03 file[3689]: UI_NETCONF_CMD: User 'ansible' us
ed NETCONF client to run command 'close-session'
Oct 28 11:17:38  R03 file[3689]: UI_LOGOUT_EVENT: User 'ansible' l
ogout
```

Datei Bearbeiten Ansicht Suchen Terminal Hilfe

```
sievers@gemini:~/JAUT/ansible-bgp-full-mesh$ ansible-playbook get_facts.pb.yml
Login Credentials: ansible
Login Password:

PLAY [Facts:Netconf] ***********************************************************

TASK: [gathering info from device] *********************************************
ok: [r01]
ok: [r02]
ok: [r03]

TASK: [version] ****************************************************************
ok: [r01] => {
    "msg": "12.1X47-D20.7"
}
ok: [r03] => {
    "msg": "12.1X47-D20.7"
}
ok: [r02] => {
    "msg": "12.1X47-D20.7"
}

TASK: [serial-number] **********************************************************
ok: [r01] => {
    "msg": "925eedcb453c"
}
ok: [r03] => {
    "msg": "a3cbfa34ce0b"
}
ok: [r02] => {
    "msg": "e98860d48db4"
}

PLAY RECAP *********************************************************************
r01                        : ok=3    changed=0    unreachable=0    failed=0
r02                        : ok=3    changed=0    unreachable=0    failed=0
r03                        : ok=3    changed=0    unreachable=0    failed=0

sievers@gemini:~/JAUT/ansible-bgp-full-mesh$
```

get_facts.pb.yml

```yaml
---
- name: "Facts:Netconf"
  hosts: router
  roles:
    - Juniper.junos
  connection: local
  gather_facts: no

  vars_prompt:
    - name: USERNAME
      prompt: Login Credentials
      private: no
    - name: PASSWORD
      prompt: Login Password
      private: yes

  tasks:
    - name: gathering info from device
      junos_get_facts:
        host={{ inventory_hostname }}
        user={{ USERNAME }}
        passwd={{ PASSWORD }}
      register: junos
    - name: version
      debug: msg={{ junos.facts.version }}
    - name: serial-number
      debug: msg={{ junos.facts.serialnumber }}
```

Line 28, Column 1 | Spaces: 2 | YAML

Ansible – deploy BGP

sievers@gemini: ~/JAUT/juise

Router — sievers@gemini: ~/JAUT/juise 77x19

```
[edit]
psievers@R01#

[edit]
psievers@R01# show protocols bgp
group iBGP {
    type internal;
    local-address 172.27.255.1;
    log-updown;
    family inet {
        unicast;
    }
    neighbor 172.27.255.3;
    neighbor 172.27.255.2;
}

[edit]
psievers@R01#
```

Router — sievers@gemini: ~/JAUT/juise 75x19

```
psievers@R02#

[edit]
psievers@R02#

[edit]
psievers@R02#

[edit]
psievers@R02#

[edit]
psievers@R02#

[edit]
psievers@R02# show protocols bgp

[edit]
psievers@R02#
```

Router — sievers@gemini: ~/JAUT/juise 66x19

```
psievers@R03#

[edit]
psievers@R03#

[edit]
psievers@R03#

[edit]
psievers@R03#

[edit]
psievers@R03#

[edit]
psievers@R03# show protocols bgp

[edit]
psievers@R03#
```

Datei Bearbeiten Ansicht Suchen Terminal Hilfe

```
sievers@gemini:~/JAUT/ansible-bgp-full-mesh$ ansible-playbook merge_bgp_config.yml
Login Credentials: ansible
Login Password:

PLAY [Config:BGP] ***************************************************************

TASK: [Retrive information from device running junos] ************************
ok: [r01]

TASK: [Create Junos configuration from Jinja2] ******************************
ok: [r01] => (item={'r01': {'neighbors': {'r03': '172.27.255.3', 'r02': '172.27.255.2'}, 'bgp_group_name': 'iBGP', 'lo
cal_address': '172.27.255.1'}})

TASK: [load-merge BGP config] ***********************************************
changed: [r01]

PLAY RECAP ******************************************************************
r01                        : ok=3    changed=1    unreachable=0    failed=0

sievers@gemini:~/JAUT/ansible-bgp-full-mesh$
```

merge_bgp_config.yml

```
10      - name: USERNAME
11        prompt: Login Credentials
12        private: no
13      - name: PASSWORD
14        prompt: Login Password
15        private: yes
16
17    vars:
18      - junos_jinja_template: "bgp_
19      - path: "junos_configs"
20
21    vars_files:
22      - "deploy_bgp.yml"
23
24    tasks:
25      - name: Retrive information f
26        junos_get_facts:
27          host={{ inventory_hostnam
28          user={{ USERNAME }}
29          passwd={{ PASSWORD }}
30        register: junos
31
32      - name: Create Junos configur
33        template: src={{ junos_jinj
34        with_items: bgp_data
35
36      - name: load-merge BGP config
37        junos_install_config:
38          host={{ inventory_hostnam
39          user={{ USERNAME }}
40          passwd={{ PASSWORD }}
41          file="{{ path }}/{{ junos
42          logfile={{ inventory_host
43
```

deploy_bgp.yml

```
1  ---
2  bgp_data:
3  - r01:
4      bgp_group_name: iB
5      local_address: 172.
6      neighbors:
7        r02: 172.27.255.
8        r03: 172.27.255.
```

bgp_jinja_template.j2

```
1  {% for bgp_vars in item.
   itervalues() %}
2  protocols {
3      bgp {
4          group {{ bgp_vars.
           bgp_group_name }} {
5              type internal;
6              local-address
               {{ bgp_vars.
               local_address
               }};
7              log-updown;
8              family inet {
9                  unicast;
10             }
11 {% for host, ip in bgp_vars
   .neighbors.iteritems()
   %}
12             neighbor {{ ip
               }};
13 {% endfor %}
14         }
15     }
16 }
17 {% endfor %}
18
```

Line 18, Column 1 | Tab Size: 4 | Plain Text

JUNOS FOR NETWORK AUTOMATION

Key takeaways

- Automation is a core part of Junos, not a “bolt on” or external API
- Current IETF standards are based on Juniper invented technologies
- Extensible, open-source frameworks
- Multiple access protocols supported for integration into OSS/BSS systems
- Enabling use of modern programming languages
- Integrations with DevOps frameworks for abstracted automation
- YANG models

Thank You!