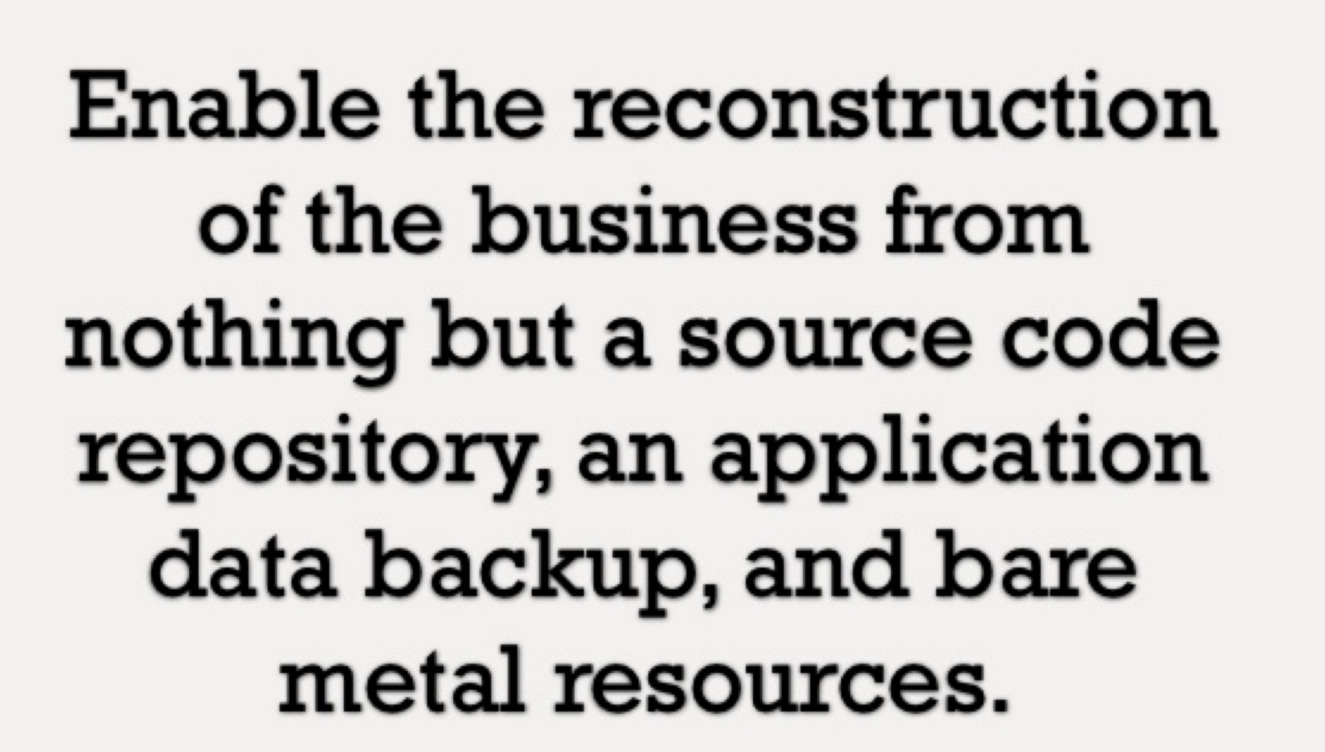
Network as Code

Amanda Galligan – Infrastructure Services

Agenda

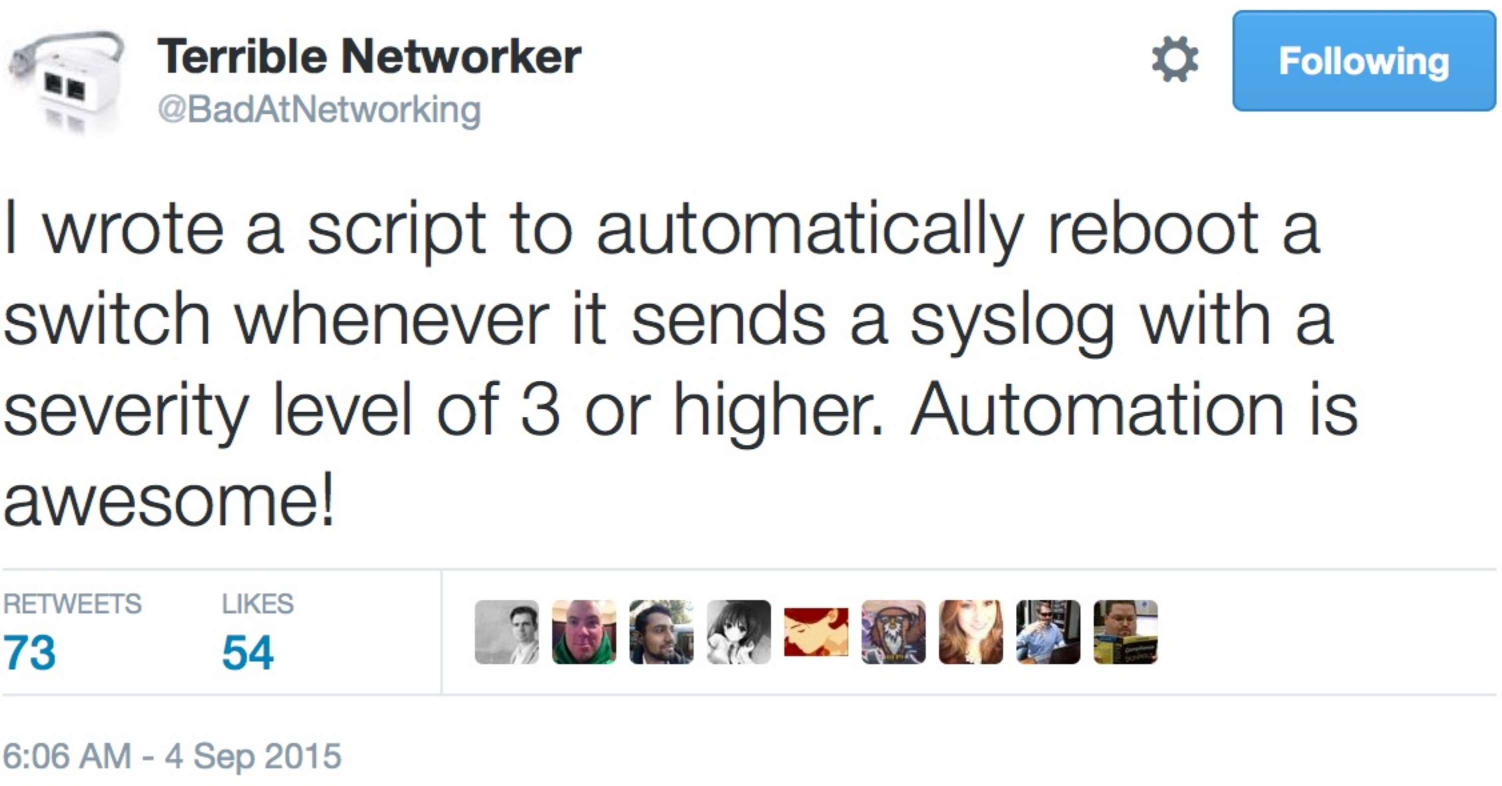
- A lofty goal..
- Network Automation.. Could we do better?
- Workday automation path to enlightenment ..
- Why NETCONF..?
- Why YANG..?
- Service model – Rack provisioning – Walk through
- Adopting development best practices
- Source Control example for Network Validation tests
- Continuous Integration workflow
- A quick demo

A lofty goal.. Infrastructure as code?



Enable the reconstruction of the business from nothing but a source code repository, an application data backup, and bare metal resources.

Network Automation.. Could we do better?

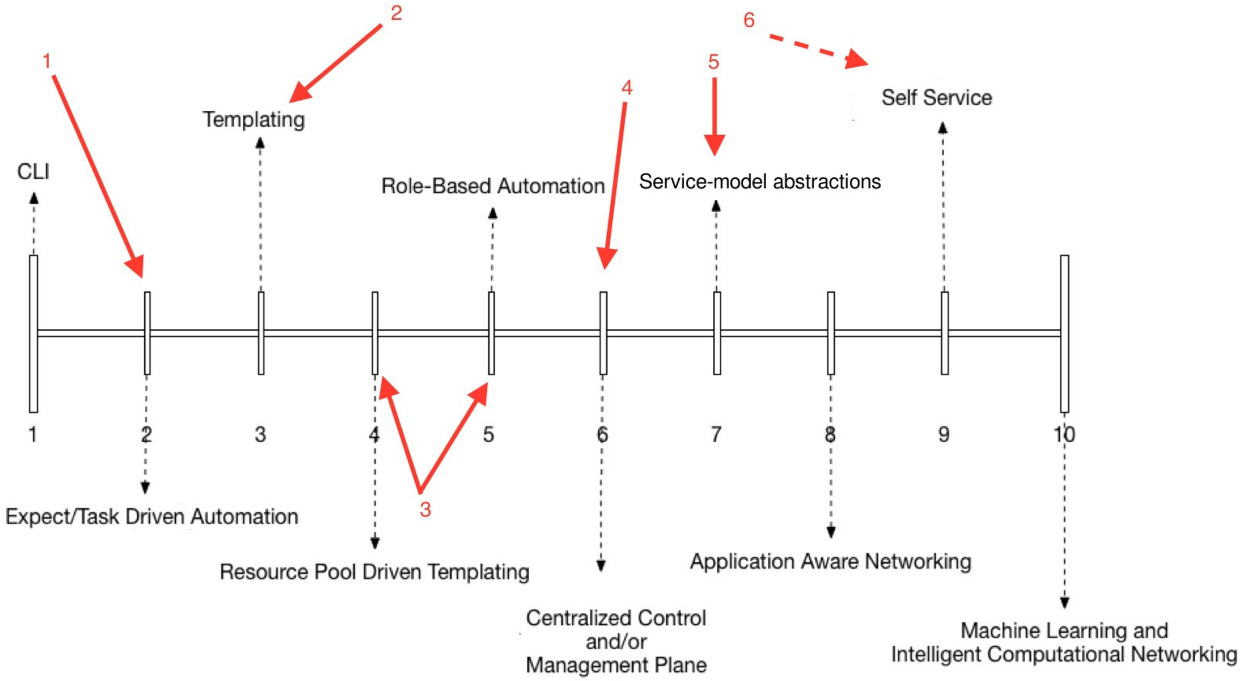


I wrote a script to automatically reboot a switch whenever it sends a syslog with a severity level of 3 or higher. Automation is awesome!

6:06 AM - 4 Sep 2015

Workday automation path to enlightenment ..

1. pExpect and paramiko scripts to perform large scale simple changes
2. Ansible templates mass device configuration consistency
3. Automation of datacenter expansion levering vendor zero touch provisioning tools and centralized inventory source
4. Single network wide interface to all network devices
5. NETCONF/YANG based abstractions coupled with CI pipeline delivery.
6. Self service API leveraging fully tested abstraction layer

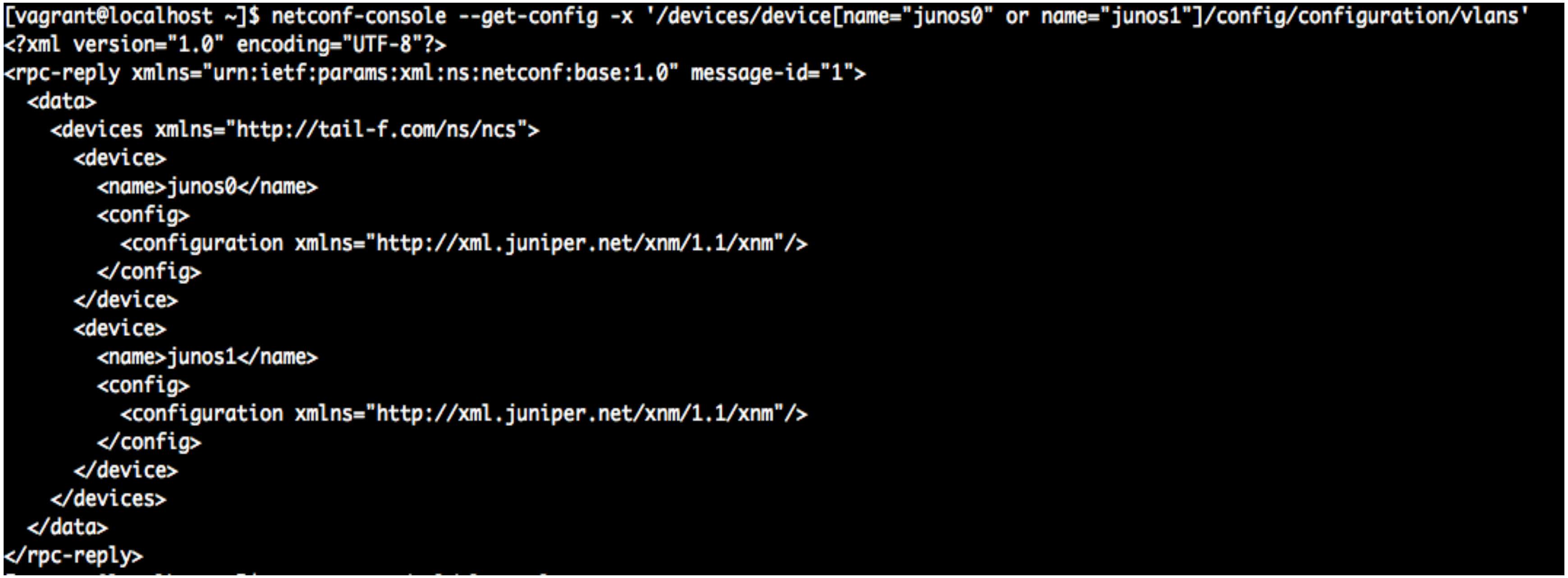

1
2
6
5
4
CLI
Templating
Role-Based Automation
Service-model abstractions
Self Service
1
2
3
4
5
6
7
8
9
10
3
Expect/Task Driven Automation
Resource Pool Driven Templating
Centralized Control and/or Management Plane
Application Aware Networking
Machine Learning and Intelligent Computational Networking

Why NETCONF..?

RFC6241 Network Configuration Protocol

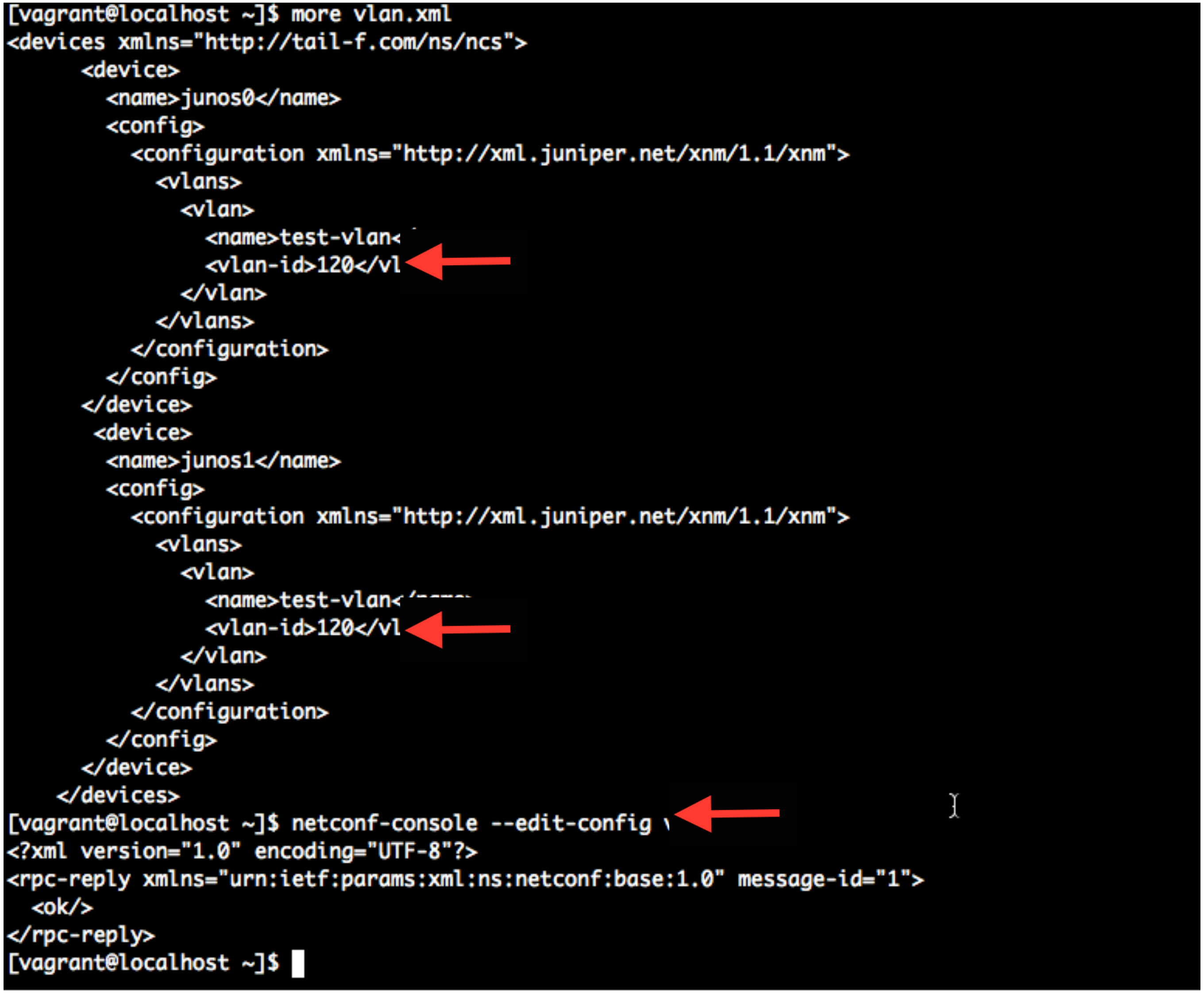
Ability to make configuration changes across multiple devices simultaneously based on abstracted requirements

ACID principal - *Atomicity, Consistency, Isolation, Durability*



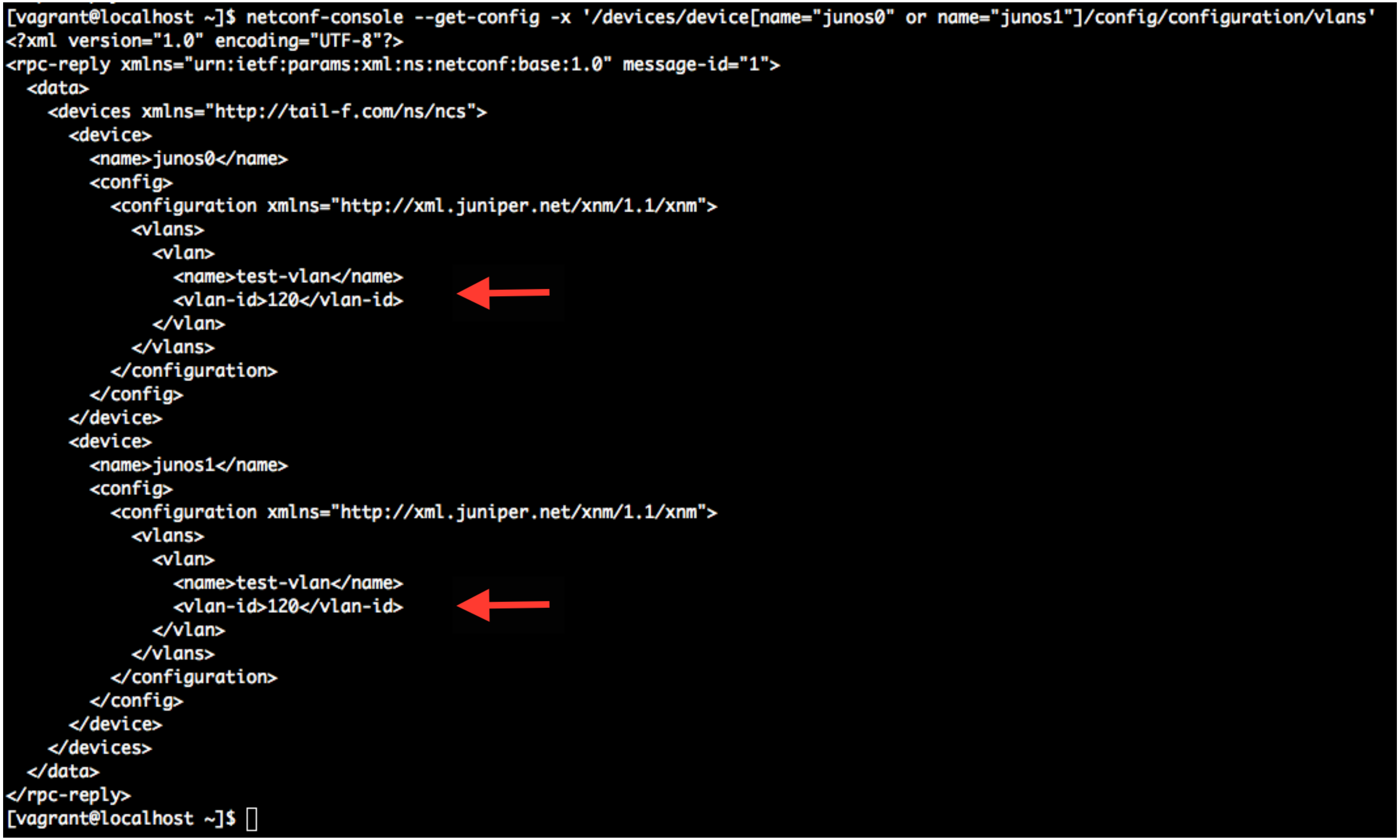
```
[vagrant@localhost ~]$ netconf-console --get-config -x '/devices/device[name="junos0" or name="junos1"]/config/configuration/vlans'
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <data>
    <devices xmlns="http://tail-f.com/ns/ncs">
      <device>
        <name>junos0</name>
        <config>
          <configuration xmlns="http://xml.juniper.net/xnm/1.1/xnm"/>
        </config>
      </device>
      <device>
        <name>junos1</name>
        <config>
          <configuration xmlns="http://xml.juniper.net/xnm/1.1/xnm"/>
        </config>
      </device>
    </devices>
  </data>
</rpc-reply>
```

Why NETCONF..?



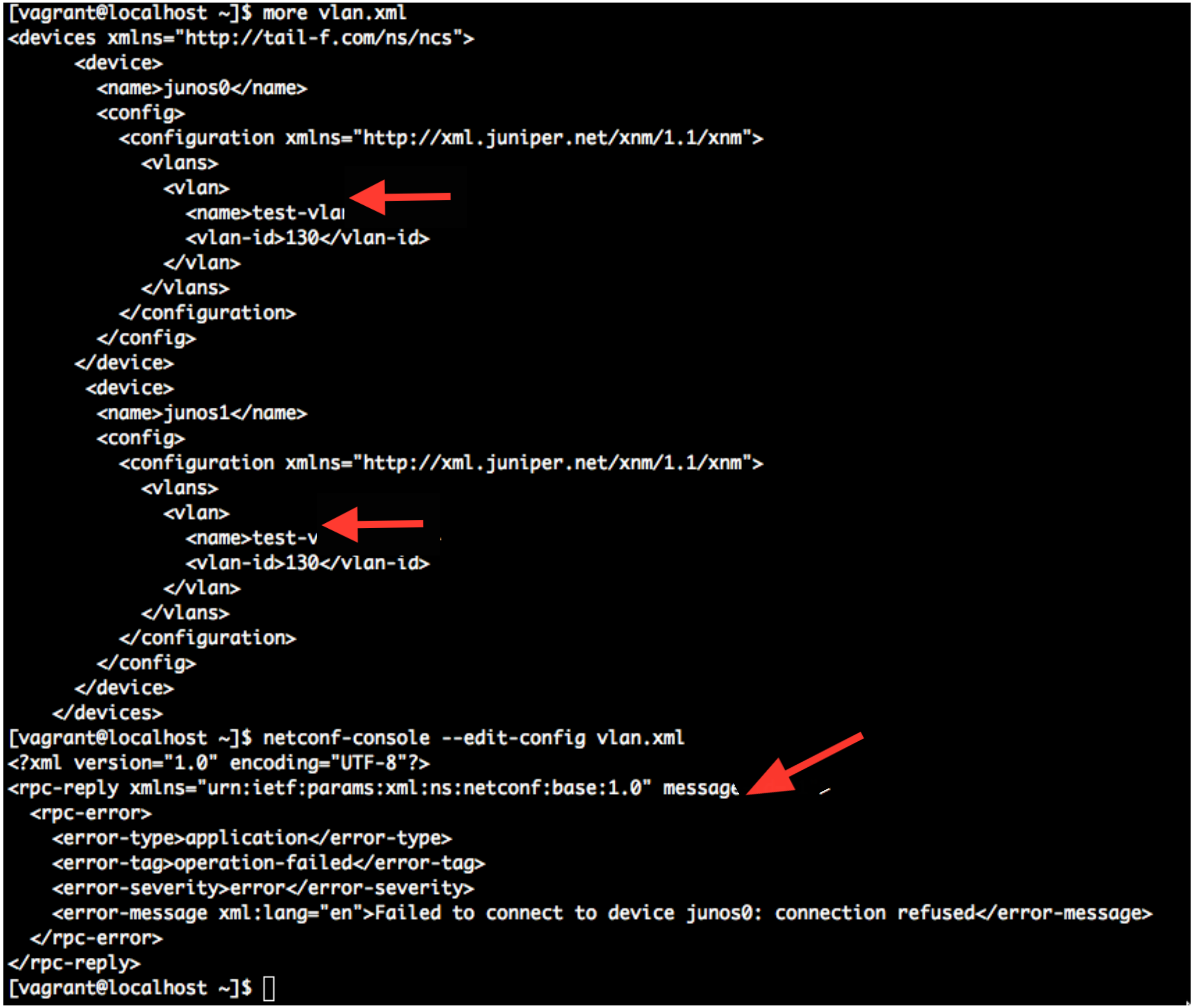
```
[vagrant@localhost ~]$ more vlan.xml
<devices xmlns="http://tail-f.com/ns/ncs">
      <device>
        <name>junos0</name>
        <config>
          <configuration xmlns="http://xml.juniper.net/xnm/1.1/xnm">
            <vlans>
              <vlan>
                <name>test-vlan<
                <vlan-id>120</vl
              </vlan>
            </vlans>
          </configuration>
        </config>
      </device>
       <device>
        <name>junos1</name>
        <config>
          <configuration xmlns="http://xml.juniper.net/xnm/1.1/xnm">
            <vlans>
              <vlan>
                <name>test-vlan<
                <vlan-id>120</vl
              </vlan>
            </vlans>
          </configuration>
        </config>
      </device>
    </devices>
[vagrant@localhost ~]$ netconf-console --edit-config
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <ok/>
</rpc-reply>
[vagrant@localhost ~]$
```

Why NETCONF..?



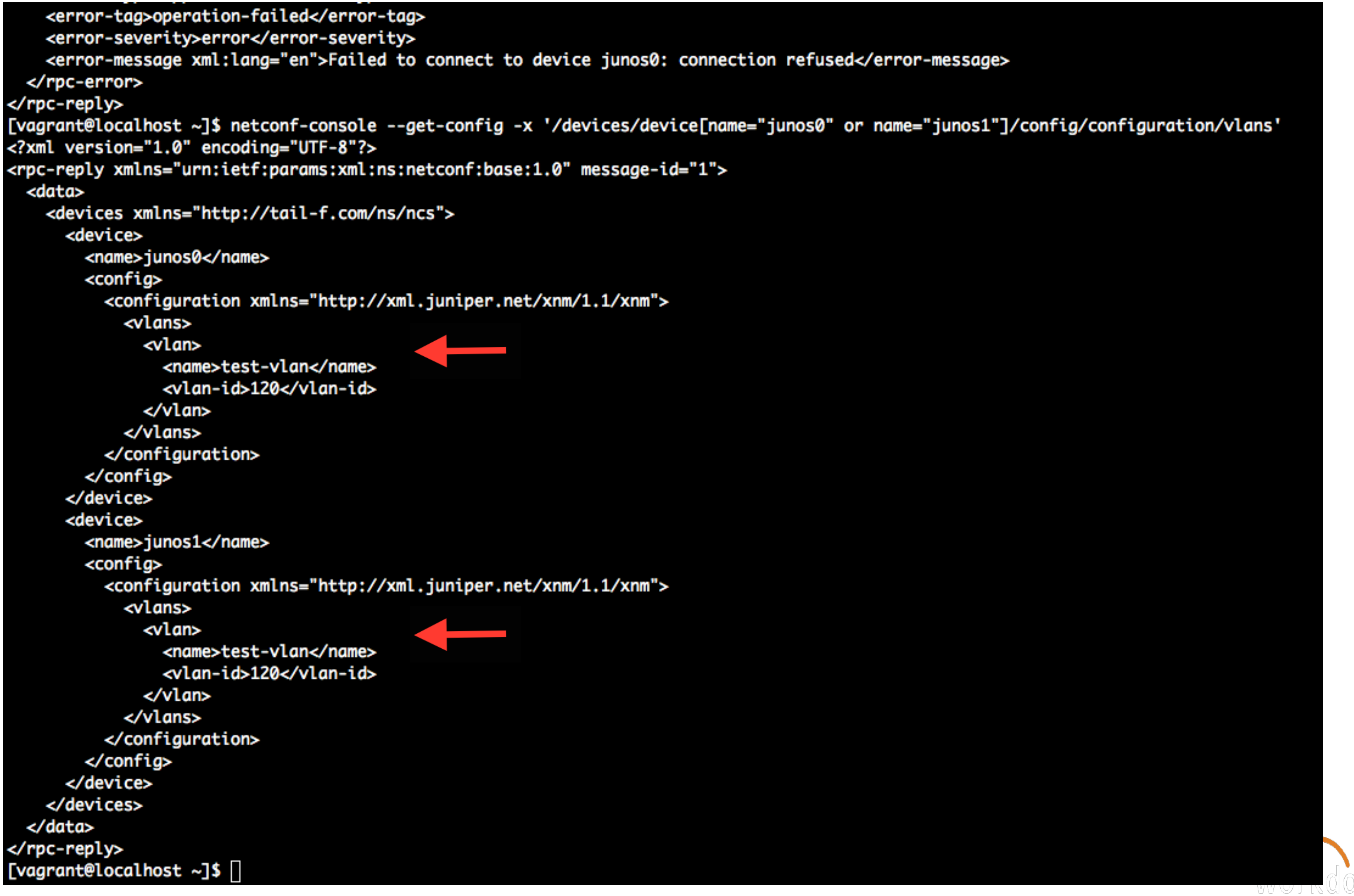
```
[vagrant@localhost ~]$ netconf-console --get-config -x '/devices/device[name="junos0" or name="junos1"]/config/configuration/vlans'
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <data>
    <devices xmlns="http://tail-f.com/ns/ncs">
      <device>
        <name>junos0</name>
        <config>
          <configuration xmlns="http://xml.juniper.net/xnm/1.1/xnm">
            <vlans>
              <vlan>
                <name>test-vlan</name>
                <vlan-id>120</vlan-id>
              </vlan>
            </vlans>
          </configuration>
        </config>
      </device>
      <device>
        <name>junos1</name>
        <config>
          <configuration xmlns="http://xml.juniper.net/xnm/1.1/xnm">
            <vlans>
              <vlan>
                <name>test-vlan</name>
                <vlan-id>120</vlan-id>
              </vlan>
            </vlans>
          </configuration>
        </config>
      </device>
    </devices>
  </data>
</rpc-reply>
[vagrant@localhost ~]$ 
```

Why NETCONF..?



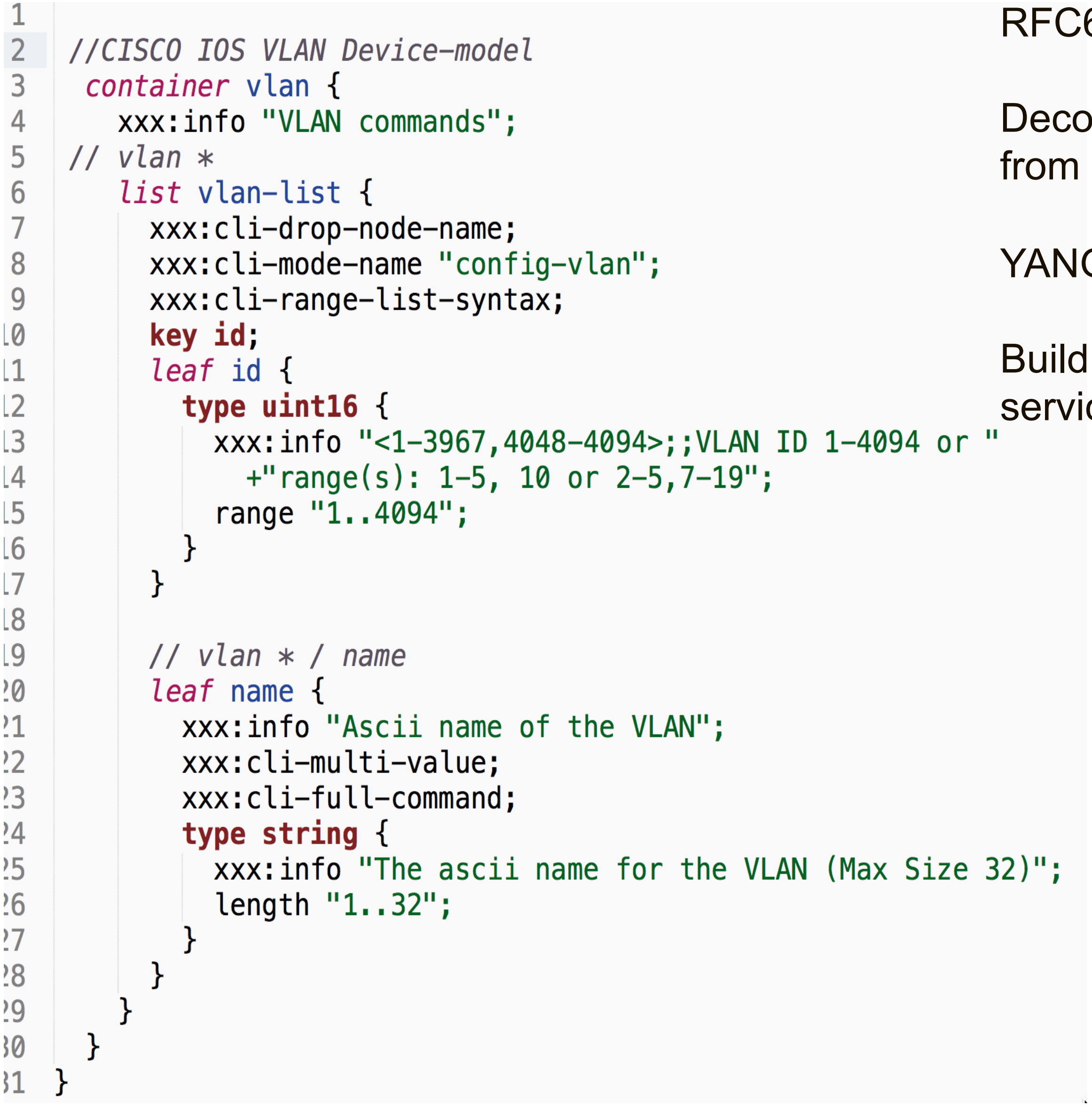
```
[vagrant@localhost ~]$ more vlan.xml
<devices xmlns="http://tail-f.com/ns/ncs">
      <device>
        <name>junos0</name>
        <config>
          <configuration xmlns="http://xml.juniper.net/xnm/1.1/xnm">
            <vlans>
              <vlan>
                <name>test-vla
                <vlan-id>130</vlan-id>
              </vlan>
            </vlans>
          </configuration>
        </config>
      </device>
       <device>
        <name>junos1</name>
        <config>
          <configuration xmlns="http://xml.juniper.net/xnm/1.1/xnm">
            <vlans>
              <vlan>
                <name>test-v
                <vlan-id>130</vlan-id>
              </vlan>
            </vlans>
          </configuration>
        </config>
      </device>
    </devices>
[vagrant@localhost ~]$ netconf-console --edit-config vlan.xml
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message
  <rpc-error>
    <error-type>application</error-type>
    <error-tag>operation-failed</error-tag>
    <error-severity>error</error-severity>
    <error-message xml:lang="en">Failed to connect to device junos0: connection refused</error-message>
  </rpc-error>
</rpc-reply>
[vagrant@localhost ~]$ 
```

Why NETCONF..?



```
    <error-tag>operation-failed</error-tag>
    <error-severity>error</error-severity>
    <error-message xml:lang="en">Failed to connect to device junos0: connection refused</error-message>
  </rpc-error>
</rpc-reply>
[vagrant@localhost ~]$ netconf-console --get-config -x '/devices/device[name="junos0" or name="junos1"]/config/configuration/vlans'
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <data>
    <devices xmlns="http://tail-f.com/ns/ncs">
      <device>
        <name>junos0</name>
        <config>
          <configuration xmlns="http://xml.juniper.net/xnm/1.1/xnm">
            <vlans>
              <vlan>
                <name>test-vlan</name>
                <vlan-id>120</vlan-id>
              </vlan>
            </vlans>
          </configuration>
        </config>
      </device>
      <device>
        <name>junos1</name>
        <config>
          <configuration xmlns="http://xml.juniper.net/xnm/1.1/xnm">
            <vlans>
              <vlan>
                <name>test-vlan</name>
                <vlan-id>120</vlan-id>
              </vlan>
            </vlans>
          </configuration>
        </config>
      </device>
    </devices>
  </data>
</rpc-reply>
[vagrant@localhost ~]$ 
```

Why YANG..?



```

//CISCO IOS VLAN Device-model
 container vlan {
  xxx:info "VLAN commands";
// vlan *
  list vlan-list {
   xxx:cli-drop-node-name;
   xxx:cli-mode-name "config-vlan";
   xxx:cli-range-list-syntax;
   key id;
   leaf id {
    type uint16 {
     xxx:info "<1-3967,4048-4094>;;VLAN ID 1-4094 or "
      +"range(s): 1-5, 10 or 2-5,7-19";
     range "1..4094";
    }
   }

   // vlan * / name
   leaf name {
    xxx:info "Ascii name of the VLAN";
    xxx:cli-multi-value;
    xxx:cli-full-command;
    type string {
     xxx:info "The ascii name for the VLAN (Max Size 32)";
     length "1..32";
    }
   }
  }
 }
}
```

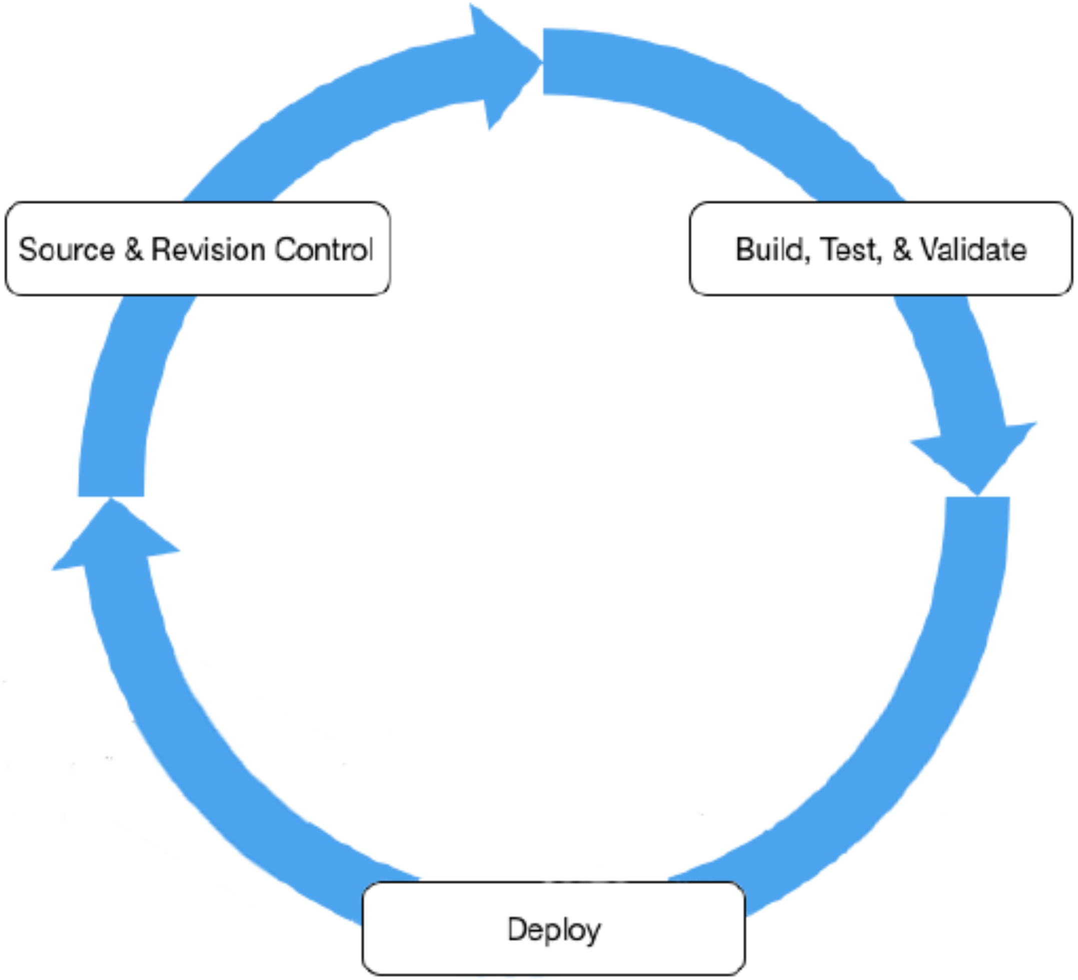
RFC6020 – Data modeling language

Decouple the device specific configuration from desired configuration state

YANG enforces conventions and structure

Build on device-models by creating service-models

Adopting development best practices

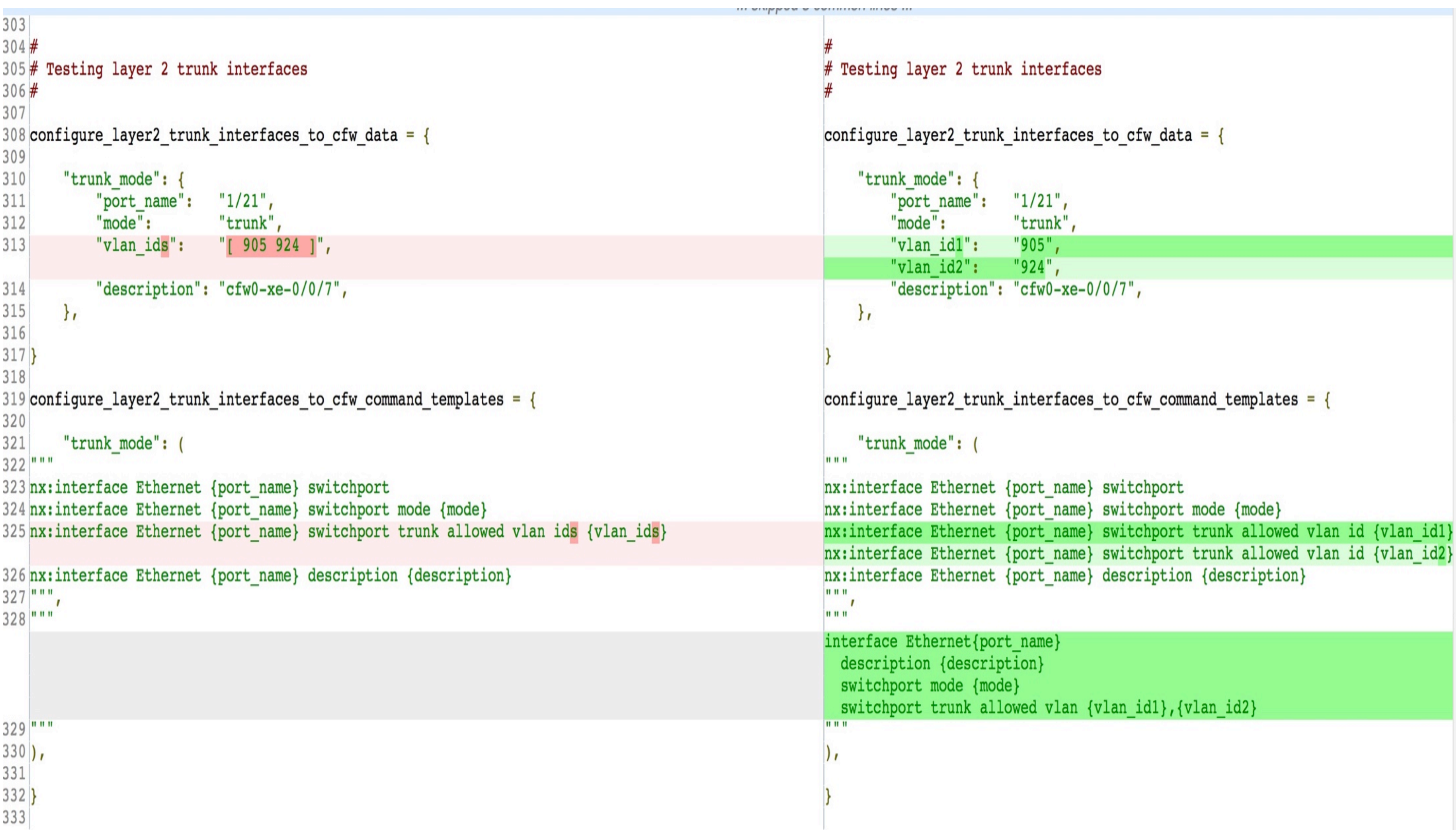


Source & Revision Control

Build, Test, & Validate

Deploy

Source Control example for Network Validation tests

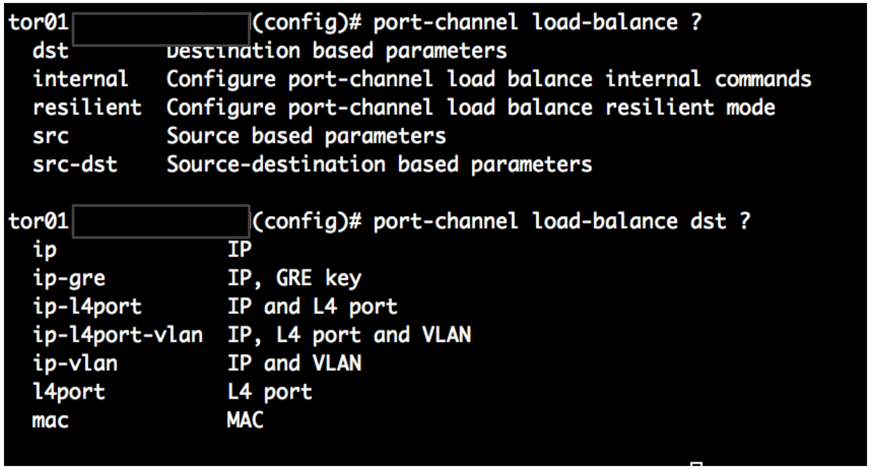


... skipped 3 common lines ...

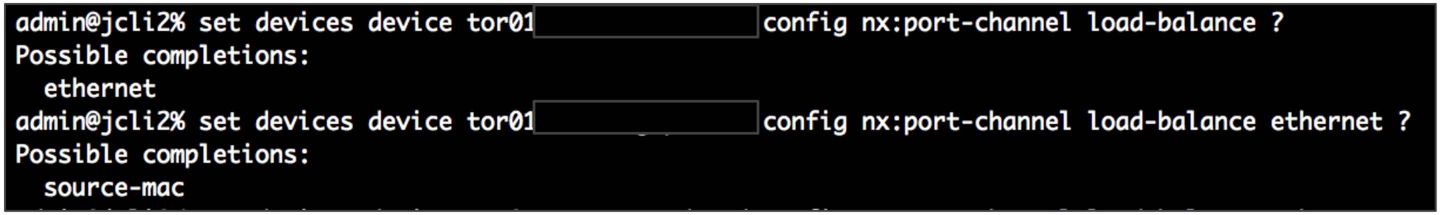
```
#
# Testing layer 2 trunk interfaces
#

configure_layer2_trunk_interfaces_to_cfw_data = {

    "trunk_mode": {
        "port_name":   "1/21",
        "mode":        "trunk",
        "vlan_ids":    "[ 905 924 ]",

        "description": "cfw0-xe-0/0/7",
    },

}

configure_layer2_trunk_interfaces_to_cfw_command_templates = {

    "trunk_mode": (
"""
nx:interface Ethernet {port_name} switchport
nx:interface Ethernet {port_name} switchport mode {mode}
nx:interface Ethernet {port_name} switchport trunk allowed vlan ids {vlan_ids}

nx:interface Ethernet {port_name} description {description}
""",
"""
"""
),

}
```

```
#
# Testing layer 2 trunk interfaces
#

configure_layer2_trunk_interfaces_to_cfw_data = {

    "trunk_mode": {
        "port_name":   "1/21",
        "mode":        "trunk",
        "vlan_id1":    "905",
        "vlan_id2":    "924",
        "description": "cfw0-xe-0/0/7",
    },

}

configure_layer2_trunk_interfaces_to_cfw_command_templates = {

    "trunk_mode": (
"""
nx:interface Ethernet {port_name} switchport
nx:interface Ethernet {port_name} switchport mode {mode}
nx:interface Ethernet {port_name} switchport trunk allowed vlan id {vlan_id1}
nx:interface Ethernet {port_name} switchport trunk allowed vlan id {vlan_id2}
nx:interface Ethernet {port_name} description {description}
""",
"""
interface Ethernet{port_name}
  description {description}
  switchport mode {mode}
  switchport trunk allowed vlan {vlan_id1},{vlan_id2}
"""
),

}
```

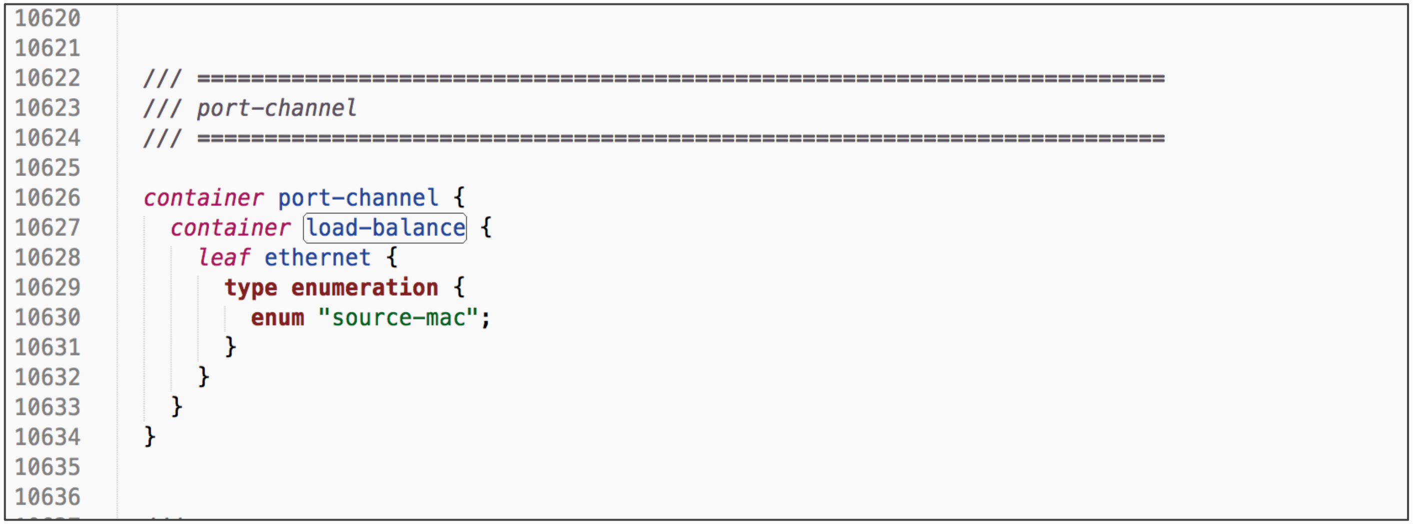
Network Element Driver – Unsupported items?



```
tor01[redacted](config)# port-channel load-balance ?
  dst         Destination based parameters
  internal    Configure port-channel load balance internal commands
  resilient   Configure port-channel load balance resilient mode
  src         Source based parameters
  src-dst     Source-destination based parameters

tor01[redacted](config)# port-channel load-balance dst ?
  ip               IP
  ip-gre           IP, GRE key
  ip-l4port        IP and L4 port
  ip-l4port-vlan   IP, L4 port and VLAN
  ip-vlan          IP and VLAN
  l4port           L4 port
  mac              MAC
```

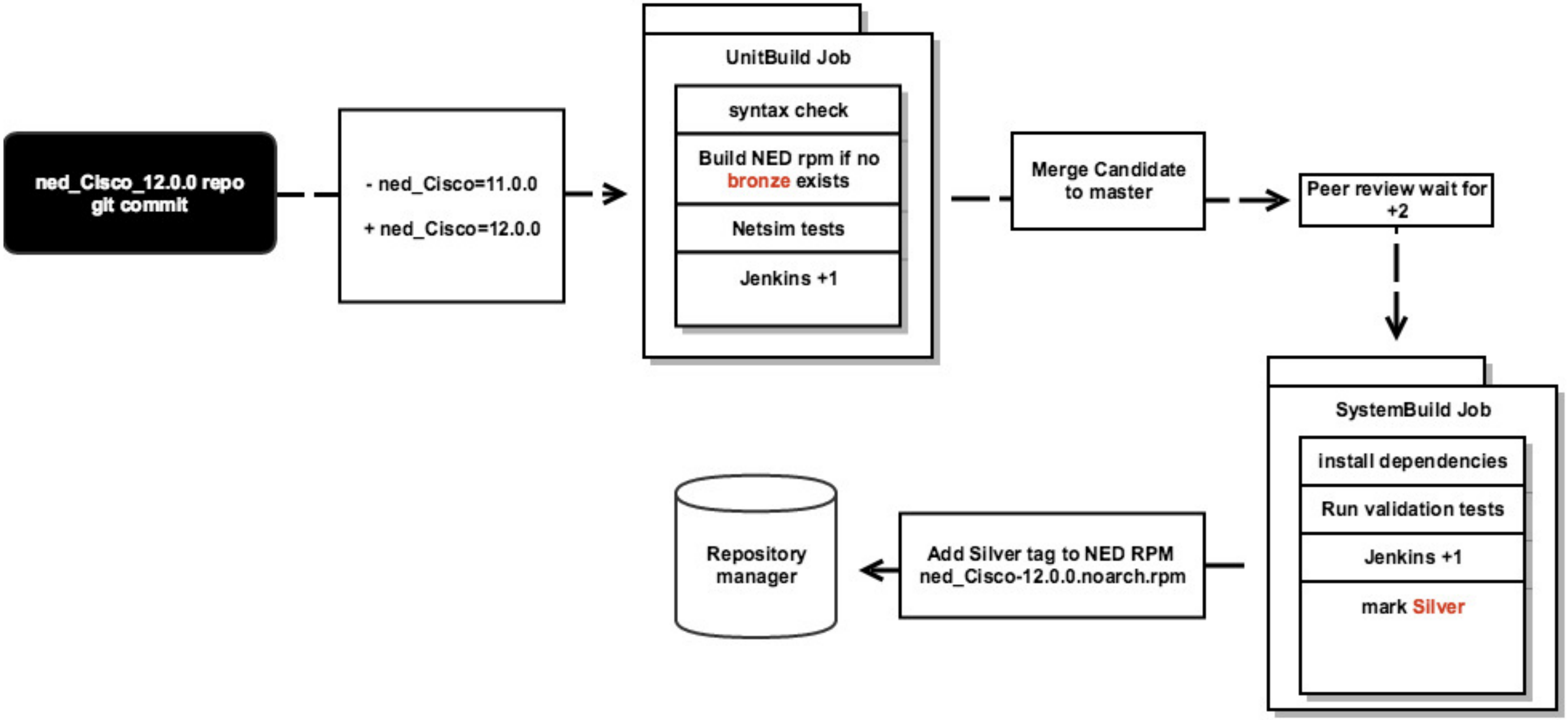


```
admin@jcli2% set devices device tor01[redacted]config nx:port-channel load-balance ?
Possible completions:
  ethernet
admin@jcli2% set devices device tor01[redacted]config nx:port-channel load-balance ethernet ?
Possible completions:
  source-mac
```



```


  /// ======================================================================
  /// port-channel
  /// ======================================================================

  container port-channel {
    container load-balance {
      leaf ethernet {
        type enumeration {
          enum "source-mac";
        }
      }
    }
  }


```

Continuous Integration workflow



Network automation is not about boiling the ocean