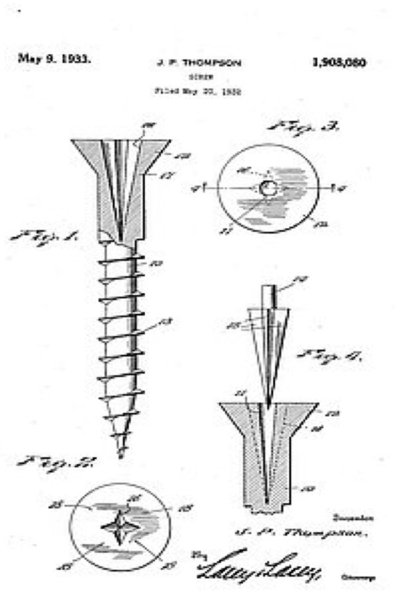
Automating Multi-Vendor Networks

Teren Sapp

Arista Networks

The “Crosshead” screw

- Credited to John P. Thompson in 1932
- Sold to Henry F. Phillips in 1935
- Thompson struggled to get into manufacturing and gain industry support
- Granted the first patent in 1932 for the screw and in 1933 for the screwdriver



Why Did We Need Another?

- The slot varied by screw size and required a closely matched bit on the driver
- Alignment with the bit to the screw aperture is difficult
- The bit often slips out since both ends are open

The Right Tool For the Job

- The Phillips screw overcomes the problems of the slotted screw

  However….

- Single slot screws (aka flat head) are still quite prevalent
- In certain applications, one may be better than the other
- Hardware stores sell both along with many other types

How’s this tie into Multi-Vendor Network Automation?

- Lots of choices when it comes to automation
  - Choose the one that’s right for ***you***

- Ansible, Puppet, Chef, Salt etc are all great and very powerful

So what/where to start?

“ *The first rule of any technology used in a business is that automation applied to an efficient operation will magnify the efficiency. The second is that automation applied to an inefficient operation will magnify the inefficiency - Bill Gates*

Napalm

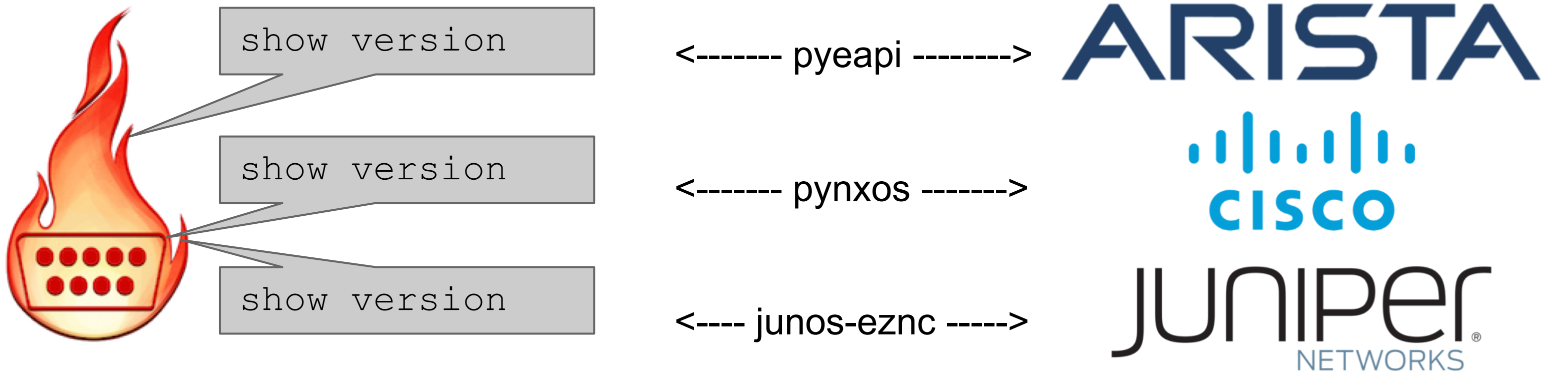
Network Automation and Programmability Abstraction Layer with Multivendor support

- Napalm simplifies the communication to network devices
- Uses modern API’s for interaction
- Simply provide the credentials and transport and you’re off!

Napalm

Network Automation and Programmability Abstraction Layer with Multivendor support

Network Automation and Programmability Abstraction Layer with Multivendor support

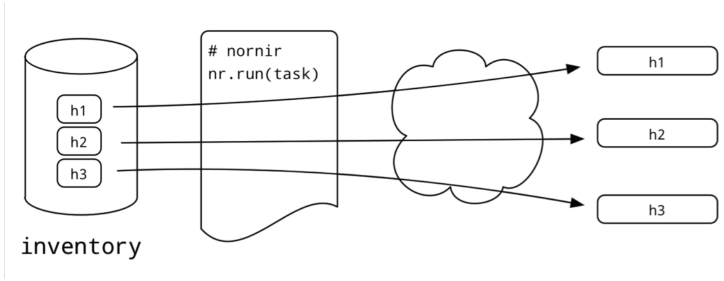


Why not netmiko or paramiko?

Goes back to the

Nornir - Formally Brigade

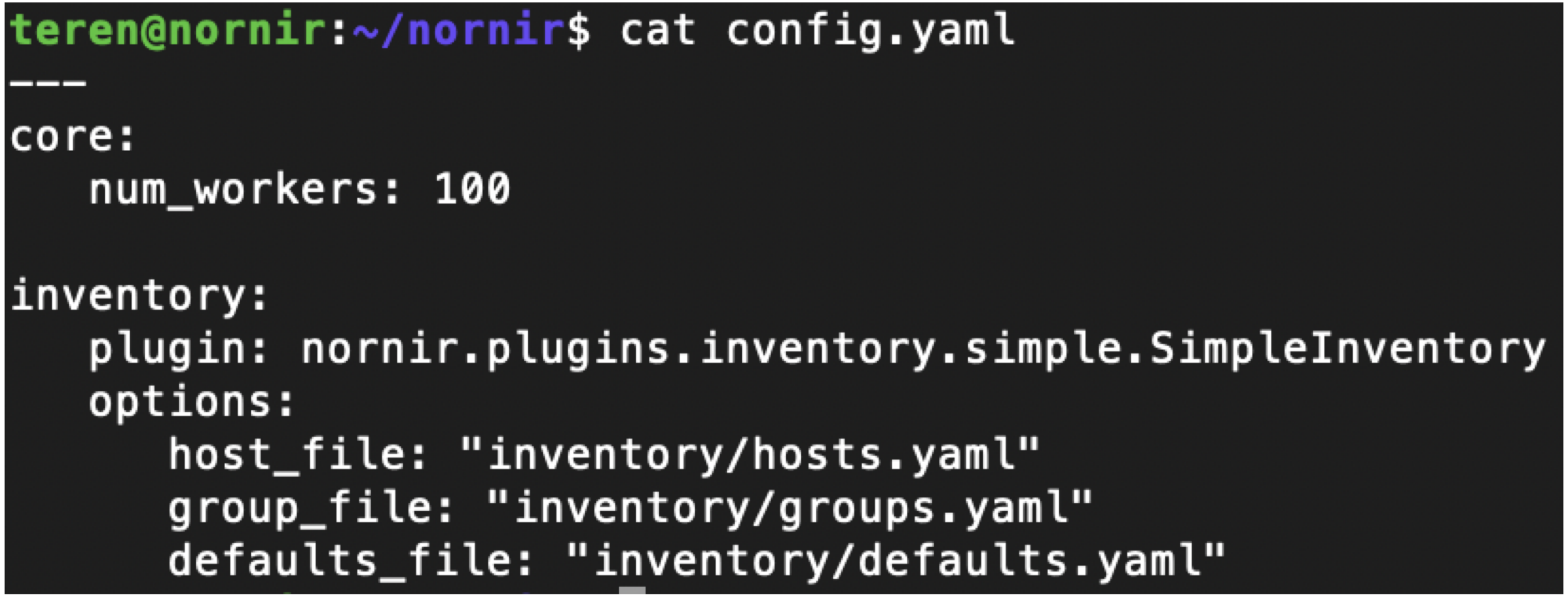
- Written completely in Python
- Purpose built for network automation
- Easy to debug - use Python tools!
- Manages inventory of devices



Parallel Task Execution

- Allows you to define how many “workers” you’d like (default is 20)
- 1 worker means everything is processed serially
- Tasks create more tasks which then run in serial

Nornir Initialization



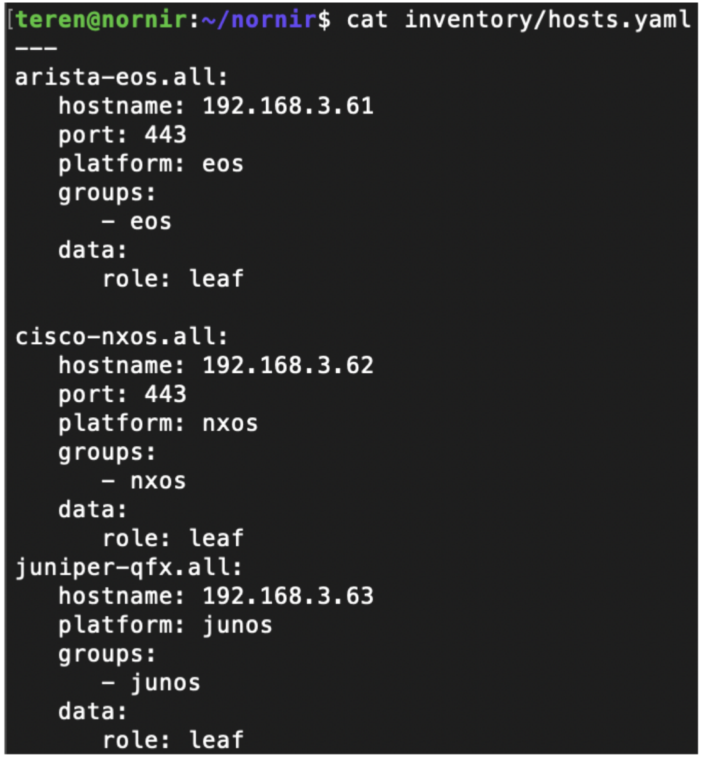
```
teren@nornir:~/nornir$ cat config.yaml
---
core:
   num_workers: 100

inventory:
   plugin: nornir.plugins.inventory.simple.SimpleInventory
   options:
      host_file: "inventory/hosts.yaml"
      group_file: "inventory/groups.yaml"
      defaults_file: "inventory/defaults.yaml"
```

Nornir - Inventory

- Inventory consists of three items:
  - Hosts - Individual devices
  - Groups - Groups of devices
  - Defaults - Default values for all devices

- SimpleInventory (default) uses YAML
  - Network Source of Truth (NSOT)
  - Netbox
  - Ansible

Sample inventory/hosts.yaml

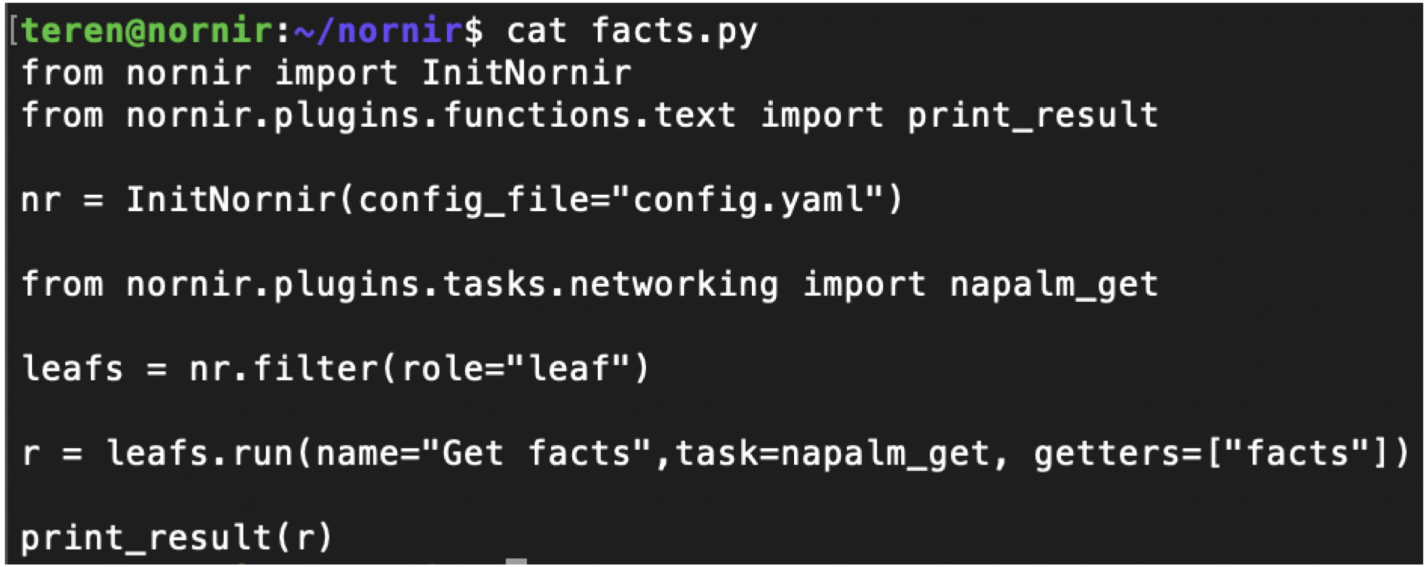


```
[teren@nornir:~/nornir$ cat inventory/hosts.yaml
---
arista-eos.all:
   hostname: 192.168.3.61
   port: 443
   platform: eos
   groups:
      - eos
   data:
      role: leaf

cisco-nxos.all:
   hostname: 192.168.3.62
   port: 443
   platform: nxos
   groups:
      - nxos
   data:
      role: leaf
juniper-qfx.all:
   hostname: 192.168.3.63
   platform: junos
   groups:
      - junos
   data:
      role: leaf
```

Putting Nornir and Napalm together

- Napalm is a plugin for Nornir
  - Could use netmiko or paramiko instead
- Nornir uses Napalm to handle all device communication

Putting Nornir and Napalm together

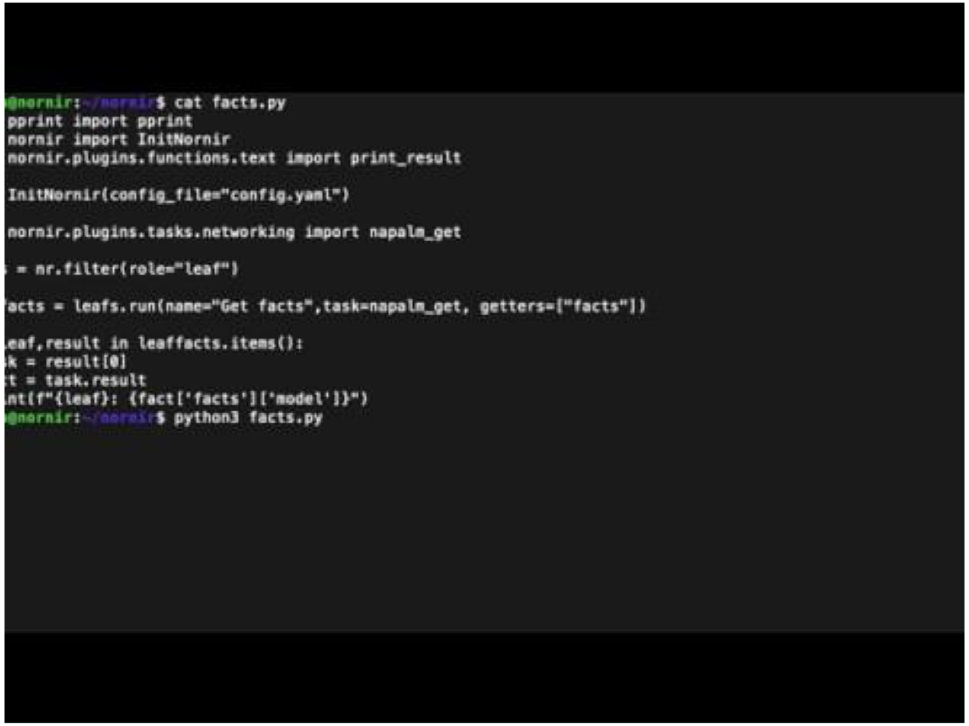


```
[teren@nornir:~/nornir$ cat facts.py
from nornir import InitNornir
from nornir.plugins.functions.text import print_result

nr = InitNornir(config_file="config.yaml")

from nornir.plugins.tasks.networking import napalm_get

leafs = nr.filter(role="leaf")

r = leafs.run(name="Get facts",task=napalm_get, getters=["facts"])

print_result(r)
```

Demo Time

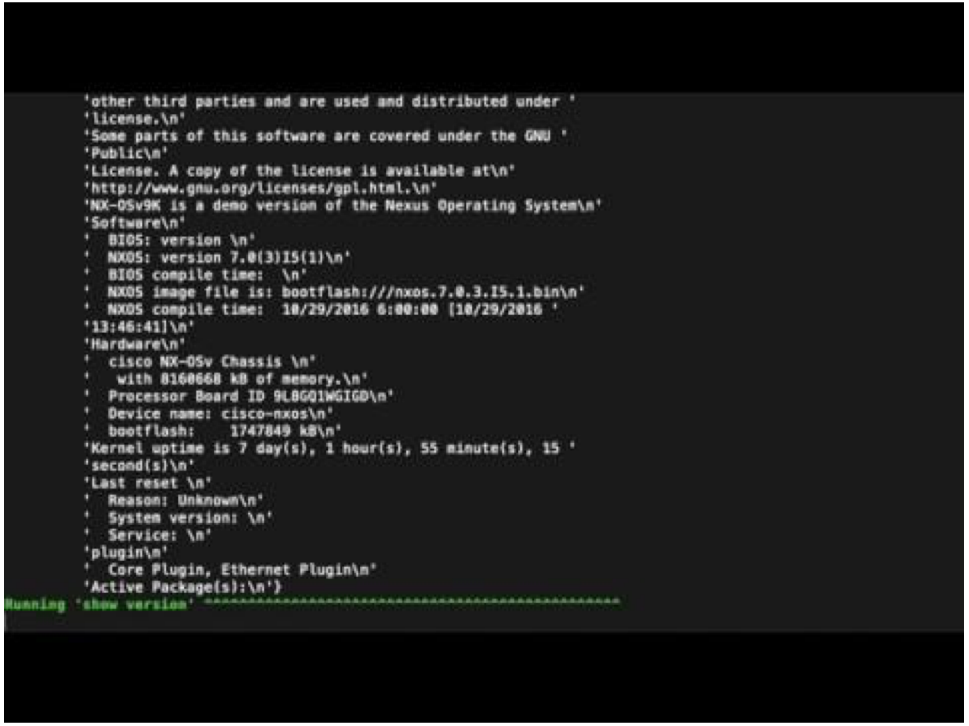
Configuration/state auditing and a few changes

Arista (eos), Cisco (nxos) and Juniper (junos)



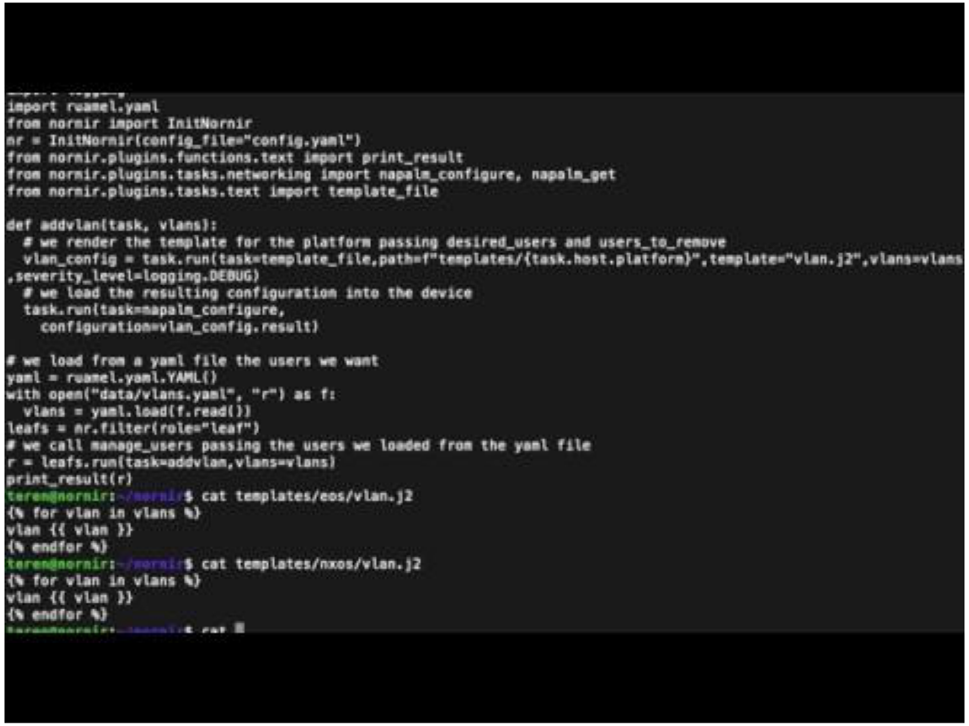
```
@nornir:~/nornir$ cat facts.py
 pprint import pprint
 nornir import InitNornir
 nornir.plugins.functions.text import print_result

 InitNornir(config_file="config.yaml")

 nornir.plugins.tasks.networking import napalm_get

 = nr.filter(role="leaf")

acts = leafs.run(name="Get facts",task=napalm_get, getters=["facts"])

eaf,result in leaffacts.items():
k = result[0]
t = task.result
nt(f"{leaf}: {fact['facts']['model']}")
@nornir:~/nornir$ python3 facts.py
```

Getting basic info like model from the devices



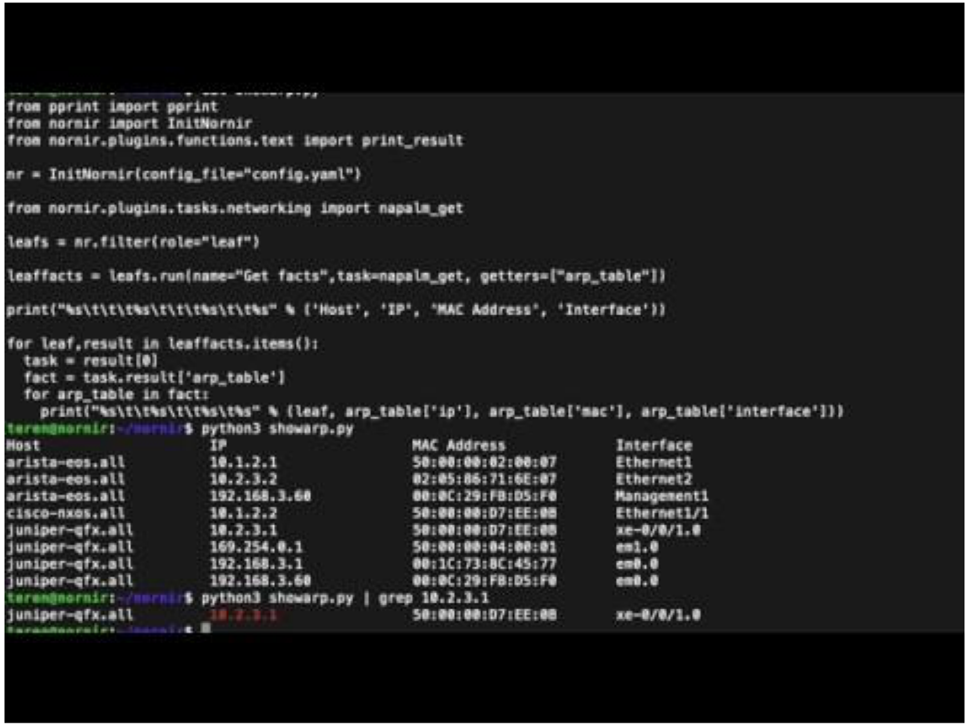
```
         'other third parties and are used and distributed under '
         'license.\n'
         'Some parts of this software are covered under the GNU '
         'Public\n'
         'License. A copy of the license is available at\n'
         'http://www.gnu.org/licenses/gpl.html.\n'
         'NX-OSv9K is a demo version of the Nexus Operating System\n'
         'Software\n'
         '  BIOS: version \n'
         '  NXOS: version 7.0(3)I5(1)\n'
         '  BIOS compile time:  \n'
         '  NXOS image file is: bootflash:///nxos.7.0.3.I5.1.bin\n'
         '  NXOS compile time:  10/29/2016 6:00:00 [10/29/2016 '
         '13:46:41]\n'
         'Hardware\n'
         '  cisco NX-OSv Chassis \n'
         '   with 8160668 kB of memory.\n'
         '  Processor Board ID 9L8GQ1WGIGD\n'
         '  Device name: cisco-nxos\n'
         '  bootflash:    1747849 kB\n'
         'Kernel uptime is 7 day(s), 1 hour(s), 55 minute(s), 15 '
         'second(s)\n'
         'Last reset \n'
         '  Reason: Unknown\n'
         '  System version: \n'
         '  Service: \n'
         'plugin\n'
         '  Core Plugin, Ethernet Plugin\n'
         'Active Package(s):\n'}
Running 'show version' ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

Running the ‘show version’ command on all devices



```
import ruamel.yaml
from nornir import InitNornir
nr = InitNornir(config_file="config.yaml")
from nornir.plugins.functions.text import print_result
from nornir.plugins.tasks.networking import napalm_configure, napalm_get
from nornir.plugins.tasks.text import template_file

def addvlan(task, vlans):
  # we render the template for the platform passing desired_users and users_to_remove
  vlan_config = task.run(task=template_file,path=f"templates/{task.host.platform}",template="vlan.j2",vlans=vlans
,severity_level=logging.DEBUG)
  # we load the resulting configuration into the device
  task.run(task=napalm_configure,
    configuration=vlan_config.result)

# we load from a yaml file the users we want
yaml = ruamel.yaml.YAML()
with open("data/vlans.yaml", "r") as f:
  vlans = yaml.load(f.read())
leafs = nr.filter(role="leaf")
# we call manage_users passing the users we loaded from the yaml file
r = leafs.run(task=addvlan,vlans=vlans)
print_result(r)
teren@nornir:~/nornir$ cat templates/eos/vlan.j2
{% for vlan in vlans %}
vlan {{ vlan }}
{% endfor %}
teren@nornir:~/nornir$ cat templates/nxos/vlan.j2
{% for vlan in vlans %}
vlan {{ vlan }}
{% endfor %}
teren@nornir:~/nornir$ cat
```

Adding a vlan to all devices



```
from pprint import pprint
from nornir import InitNornir
from nornir.plugins.functions.text import print_result

nr = InitNornir(config_file="config.yaml")

from nornir.plugins.tasks.networking import napalm_get

leafs = nr.filter(role="leaf")

leaffacts = leafs.run(name="Get facts",task=napalm_get, getters=["arp_table"])

print("%s\t\t\t%s\t\t\t%s\t\t%s" % ('Host', 'IP', 'MAC Address', 'Interface'))

for leaf,result in leaffacts.items():
  task = result[0]
  fact = task.result['arp_table']
  for arp_table in fact:
    print("%s\t\t%s\t\t%s\t%s" % (leaf, arp_table['ip'], arp_table['mac'], arp_table['interface']))
teren@nornir:~/nornir$ python3 showarp.py
Host                    IP                      MAC Address             Interface
arista-eos.all          10.1.2.1                50:00:00:02:00:07       Ethernet1
arista-eos.all          10.2.3.2                02:05:86:71:6E:07       Ethernet2
arista-eos.all          192.168.3.60            00:0C:29:FB:D5:F0       Management1
cisco-nxos.all          10.1.2.2                50:00:00:D7:EE:08       Ethernet1/1
juniper-qfx.all         10.2.3.1                50:00:00:D7:EE:08       xe-0/0/1.0
juniper-qfx.all         169.254.0.1             50:00:00:04:00:01       em1.0
juniper-qfx.all         192.168.3.1             00:1C:73:8C:45:77       em0.0
juniper-qfx.all         192.168.3.60            00:0C:29:FB:D5:F0       em0.0
teren@nornir:~/nornir$ python3 showarp.py | grep 10.2.3.1
juniper-qfx.all         10.2.3.1                50:00:00:D7:EE:08       xe-0/0/1.0
teren@nornir:~/nornir$
```

Show arp across all devices

Additional Resources

- https://github.com/dravetech/nornir-workshop - Start here!
- https://github.com/nornir-automation/nornir
- https://github.com/napalm-automation/napalm
- https://packetpushers.net/podcast/heavy-networking-445-an-introduction-to-the-nornir-automation-framework/

- Special thanks to David Barroso and the rest of the Nornir team

teren@arista.com
@terensapp

ARISTA