
University of Colorado Boulder

Network Management and Automation

Network Programming - Automation

Levi Perigo, Ph.D.
University of Colorado Boulder
Department of Computer Science
Network Engineering

Review

- **Discussions**
  - Automation Essential?
  - Challenges with Automation?

- **Upcoming lectures**
  - Faster
    - ***Labs, challenges, guest speakers***

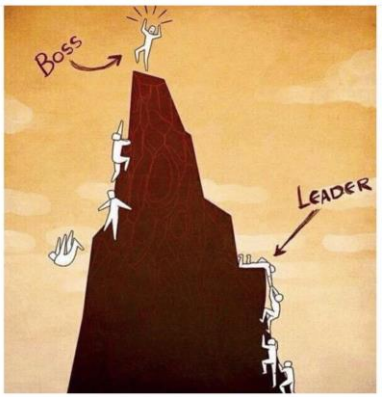
- **Upcoming labs**
  - Interviews
    - ***Understand what you're doing and why!***

- **Virtualization**
- **Cloud**
- **Data Center**
- **SDN/NFV**


KEEP
CALM
AND
AUTOMATE YOURSELF
OUT OF A JOB
KeepCalmAndPosters.com

Invaluable Employees

- **Eliminate waste & automate processes**
- **Coaching successors**
- **Share what they know**
- **Teach themselves out of a job**



- **When you spread your capabilities – you create new opportunities!**
  - Get rid of part of your job, so you can take on new challenges
    - ***Get promoted without disruption (people grow beneath you)***
- **What do these traits sound like?**

Dispensable & Invaluable

- **Document and automate, from the beginning, with your successor in mind**
  - Write instructions
  - Organize resources
  - Make it easy to turn your project over, so you can take a better opportunity
  - Share knowledge
    - ***Team works more efficiently, reduces your workload***
  - Side Benefit: Vacation without being bothered, and don't have to worry about your project falling behind!

Automate, Automate, Automate

- **Goal = Work yourself out of the bottom 10% of your job each year**
  - Opens you up for increased productivity and new opportunities

- **There is always something else to be done, something that can be improved, or new technologies to learn**
  - Time management
    - ***Use your time in grad. school wisely – this is where you can make the largest career jump!***
      - Where to find it!

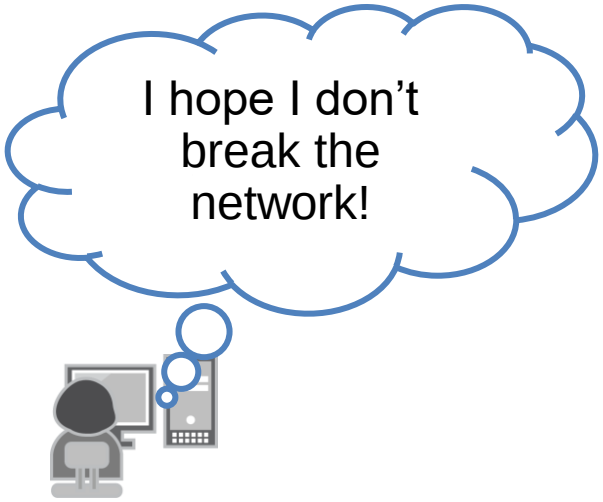
What is network automation?

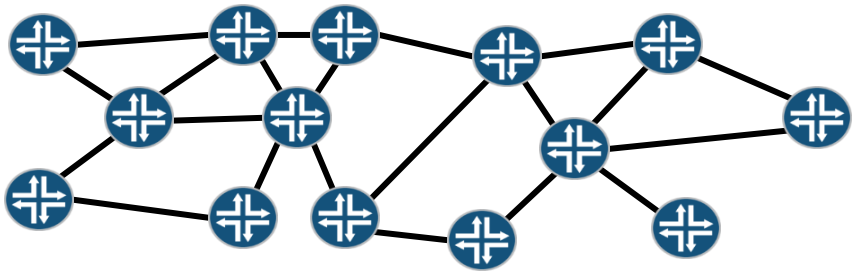
- The process of automating the configuration, management, and operations of a computer network.

- What about SDN?
  - ***The networking world is changing—we don't have to rely on vendors for innovation***

Why automation?

- **Scalability**
  - ISP network
- **Network Operations (OpEx)**
- **Human error factor (*kind of??*)**
  - AWS outage = typo!





Why automation?

- **Network Engineers (“CLI jockeys”) are choke point**
  - VM environment can spin up in seconds
  - Configure VLAN, routed interface, firewall zone, etc.
- **Replace equipment**
  - Zero Touch Provisioning (ZTP)
- **Much faster recovery & discovery**
  - Machine learning (ML) and AI

Benefits of Automation

- **Versionable**
  - Change management
    - ***GitHub***
  - Records/audit
  - CI/CD

- **Repeatable & customizable**
  - Templates, Playbooks, Cookbooks, etc.

- **Testable**
  - Jenkins, Travis, Batfish, Pytest, Custom

- **Rapid deployment**

Benefits of Automation

- **Automated back-up system**
  - "Single" point of control = one point of back-up for configurations, templates, etc.
    - ***Logically centralized, physically distributed***
  - Additional layer of backup
    - ***Hosted in VM***
      - Backup VM = backup ENTIRE network!

- **Forget about manually configuring devices**
  - ("conf t" is dead)

Goal of Automation

- **“Single” point of control (NMAS)**
  - Drive the network from management station
    - ***Configuration change (VLAN, Route)***
- **Automation tools**
- **Run infrastructure as code (IaC)**
  - What does this mean?

Goal of Automation

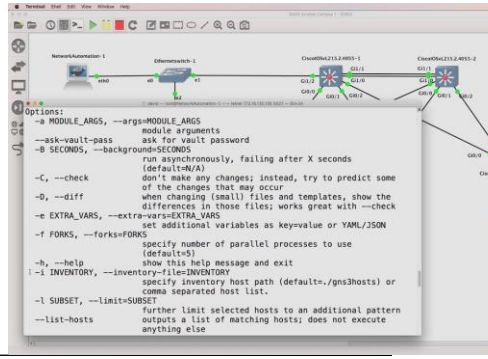
- **Configuration files – create templates for network**
  - ToR switches
    - ***Similar VLANs and type of services***
    - ***Repeated and reused / common code multiple devices***
  - Don't manually configure 'x' # of switches
    - ***Create templates/automation tool to push out config.***
    - ***Have a variables file to configure minor differences (IP addresses)***
      - Scripting (.py) or Tool (Ansible) or template (j2)
    - ***ZTP***
  - It is often better to configure even 1-2 devices via automation instead of CLI. Why?

Automation Framework

- **Scalable**
  - Proof of Concept (PoC)
    - ***Solve for few, can solve for many***
      - *Code must change at large scale
- **Easily configurable and customizable**
- **Config. verification and enforcement**
- **Statistics collection**

Devices, Processes, and Automation

- **Make every device as similar as possible**
  - Standardized configurations and config. process
  - Reduce variation in vendors, platforms, and versions (COTS; NAPALM)
  - Reduce variation in topologies and features
- **Purchase hardware that has API**
  - Get away from screen-scraping
- **Virtual devices and test environment to validate changes (see DevOps)**
- **Organizational commitment to automation**



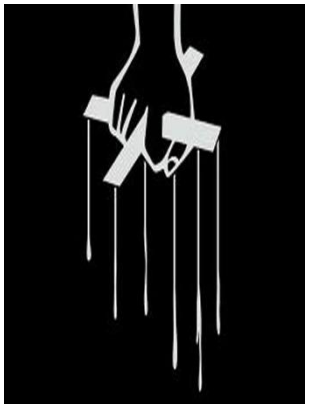
Orchestration & Abstraction

- **Orchestration**
  - System to automatically create, initialize, coordinate & manage physical and virtual resources for cloud service delivery
    - ***Airplane***

- **Abstraction**
  - Hides all but the relevant data in order to reduce complexity and increase efficiency
    - ***Automobile***

- **Orchestration, Abstraction, and Automation all work together!**

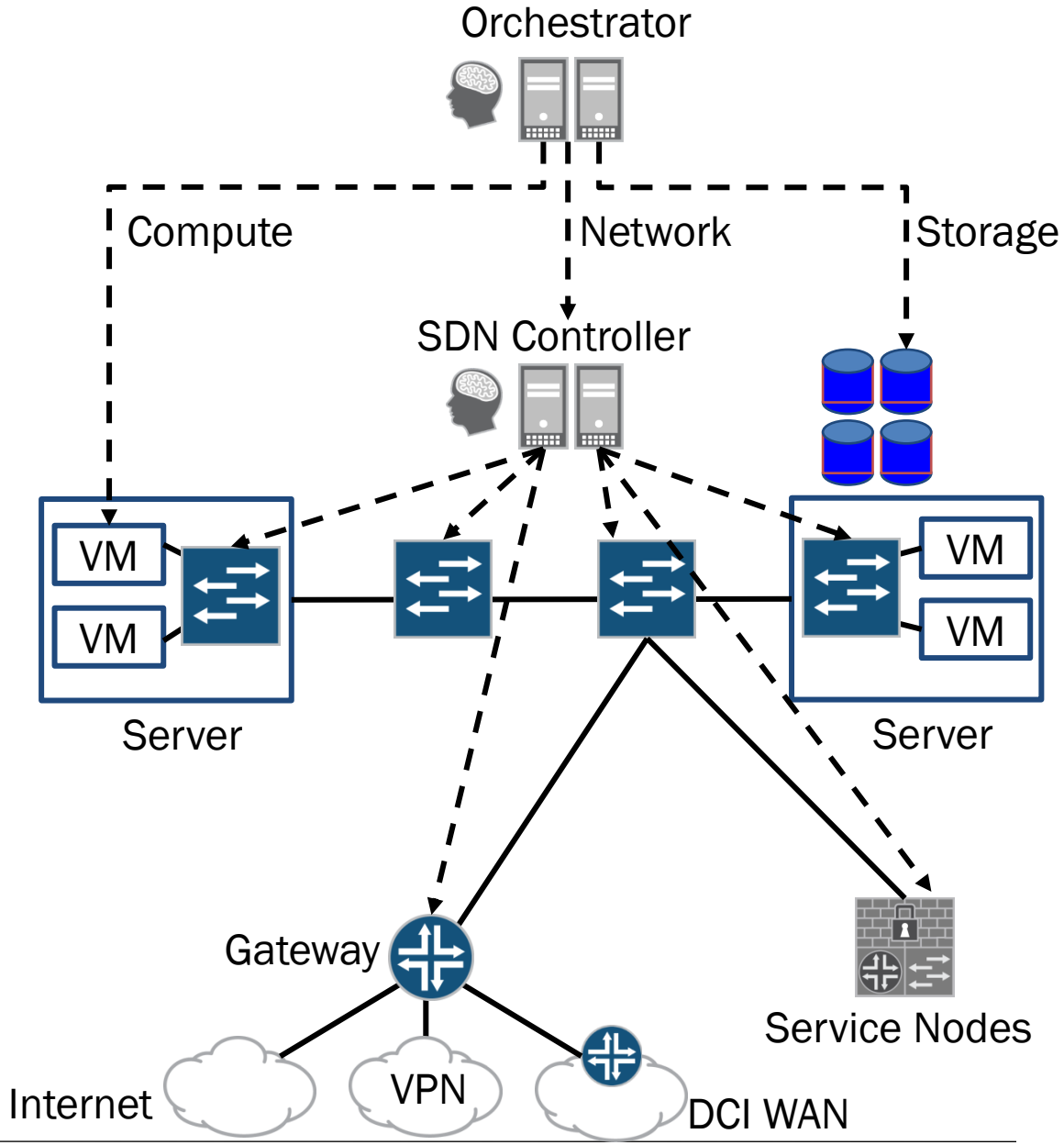
Orchestration



- **Early automation applications**
- **Application Programmer Interfaces (API)**
  - REST, CLI, SNMP, XMPP, NetConf
- **Limited support for legacy equipment**
- **Typically, vendor specific**
  - CLI commands – Cisco vs. Juniper

Data Center Orchestration

- **Orchestration**
  - Compute
    - ***Deliver the VM***
  - Network
    - ***Connect the VM to the network***
  - Storage
    - ***Connect the VM to storage***

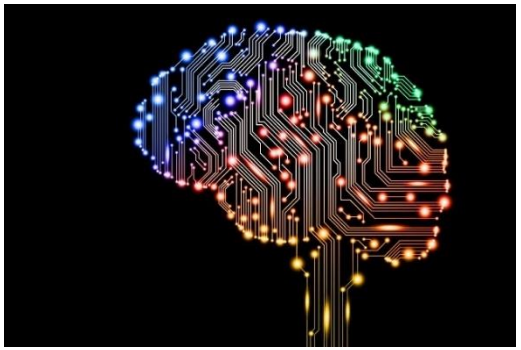


Abstraction

- **Distributed State Abstraction**
  - Provides the network programmer with a global network view; not all the various machines
- **Forwarding Abstraction**
  - Programmer can configure/specify forwarding behaviors without knowing vendor CLI
- **Configuration Abstraction**
  - Just need to know the general goals of the network (intent)

Self-reliant Network

- **Networks are dynamic not static**
- **Probes and scripts to auto-adjust network config./design**
- **Machine learning (ML) / AI**
  - What is ML & AI?
  - What will this be in the future?

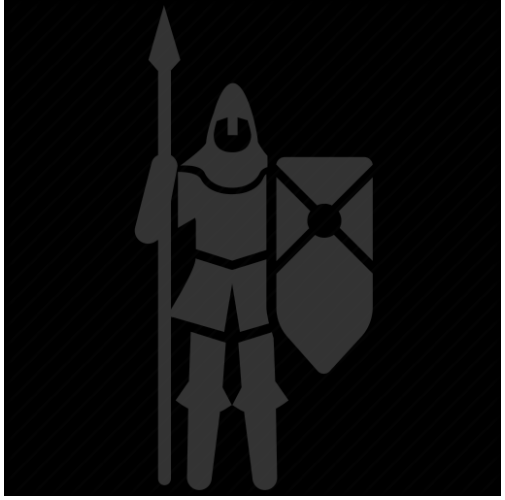


Audit & Documentation

- **DTD Cycle = Document > Test > Deploy**

- **Generate errors and completed tasks**
  - Python has many “error” libraries
  - Send update to NMS for visualization and readability
    - ***Prometheus & Grafana***

Safeguards



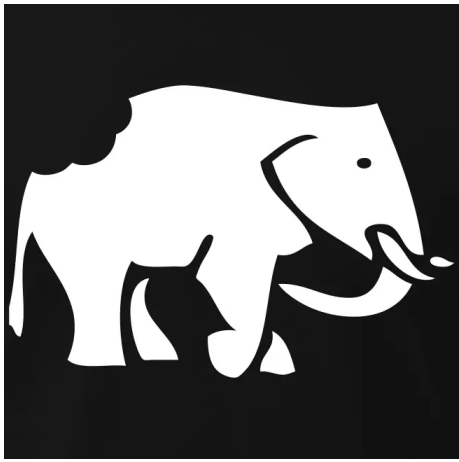
- **Input validation**
  - Test if valid IP address
  - Test if IP address is accessible
  - Test if file is available
  - Test if commands are correct
- **Monitor how changes propagate through the network**
  - More than just “print” output
  - Real-time visualization
- **Peer-review mechanism**
- **Unit Testing**

Safeguards

- **Enable/disable “button”**
  - Pause for devices that don’t accept changes?
  - (big red STOP button)
- **Manual rollback CLI**
  - Generate template for rollback too!

Safeguards

- **Test a small part of the network!**
  - How do you eat an elephant?
- **Test in staging environment (DevOps)**
- **Test/evaluate all steps of the automation:**
  - Start with “show” commands
  - Add description and IP address
    - ***Test if on the unit***
  - Add routing
    - ***Test***
  - Then no shut / enable
  - Have a backup plan
    - ***“Big red button”***



Legacy Devices (Configurations)

- **Very hard to adapt configurations "enhanced" manually over the years**
  - Inconsistencies hard to fix while in production
    - ***(is this config. section still necessary?)***
  - One of the biggest drawbacks of automation
    - ***"band-aid" fixes without documentation***

- **New sites**
  - Fully automate from beginning (ZTP)
  - Fix old sites as time allows (internship)
  - Upgrade to new design
  - Use tools/templates
    - ***Ansible/J2/NAPALM – replace "vendor A" with "vendor B"***
      - Change variable

Network Programming

- **No complex code necessary**
- **Foreign to most network engineers without programming background**
  - This is your competitive advantage!
- **Data structures, loops, regex**
- **Learn the basics:**
  - Scripting, Ansible variables, playbooks, etc.

Getting Started in Automation (Golden Rules)

1. **Start with low-risk, easy problems**
2. **Don’t assume an all-or-nothing mindset**
3. **Don’t reinvent – Reuse!**
4. **Don’t copy code/patterns without comprehension**
5. **Learn good debugging**

Getting Started in Automation (Golden Rules)

6. **Don’t over-engineer the solution**
7. **Apply things you learn at small scale**
8. **You’re never too busy to automate**
9. **Reuse your code long term**
10. **Use available developer tools and libraries**

Come on, MANE!

- **Be a MANE**
  - Machine Augmented Network Engineer

Programmer (CS) vs. Network Engineering

- **Google, Meta, Linux Foundation**

- **Get it done! (doesn't have to be perfect)**
  - Send to developers to make it better/more efficient
  - "Perfect is the enemy of good."

Lab

- **Python IS automation!**

- **Automation**
  - Powerful capability
  - Stakes are significantly increased

Key Automation Libraries

- **Netmiko**
- **NAPALM**
- **Trigger**
- **Thrift**
- **Threading/Parallel Processing/Async IO**

Netmiko

```
#!/usr/bin/env python

from netmiko import ConnectHandler

iosv_l2 = {
    'device_type': 'cisco_ios',
    'ip': '192.168.122.72',
    'username': 'david',
    'password': 'cisco',
}


net_connect = ConnectHandler(**iosv_l2)
#net_connect.find_prompt()
output = net_connect.send_command('show ip int brief')
print output

config_commands = ['int loop 0', 'ip address 1.1.1.1 255.255.255.0']
output = net_connect.send_config_set(config_commands)
print output

for n in range (2,21):
    print "Creating VLAN " + str(n)
    config_commands = ['vlan ' + str(n), 'name Python_VLAN ' + str(n)]
    output = net_connect.send_config_set(config_commands)
    print output
```

```
#!/usr/bin/env python

from netmiko import ConnectHandler

iosv_l2_s1 = {
    'device_type': 'cisco_ios',
    'ip': '192.168.122.71',
    'username': 'david',
    'password': 'cisco',
}

iosv_l2_s2 = {
    'device_type': 'cisco_ios',
    'ip': '192.168.122.72',
    'username': 'david',
    'password': 'cisco',
}

iosv_l2_s3 = {
    'device_type': 'cisco_ios',
    'ip': '192.168.122.73',
    'username': 'david',
    'password': 'cisco',
}


all_devices = [iosv_l2_s1, iosv_l2_s2, iosv_l2_s3]

for devices in all_devices:
    net_connect = ConnectHandler(**devices)
    for n in range (2,21):
        print "Creating VLAN " + str(n)
        config_commands = ['vlan ' + str(n), 'name Python_VLAN ' + str(n)]
        output = net_connect.send_config_set(config_commands)
        print output
```

```
#!/usr/bin/env python

from netmiko import ConnectHandler

iosv_l2_s1 = {
    'device_type': 'cisco_ios',
    'ip': '192.168.122.71',
    'username': 'david',
    'password': 'cisco',
}

iosv_l2_s2 = {
    'device_type': 'cisco_ios',
    'ip': '192.168.122.72',
    'username': 'david',
    'password': 'cisco',
}

iosv_l2_s3 = {
    'device_type': 'cisco_ios',
    'ip': '192.168.122.73',
    'username': 'david',
    'password': 'cisco',
}

with open('iosv_l2_config1') as f:
    lines = f.read().splitlines()
print lines

all_devices = [iosv_l2_s1, iosv_l2_s2, iosv_l2_s3]

for devices in all_devices:
    net_connect = ConnectHandler(**devices)
    output = net_connect.send_config_set(lines)
    print output
```

NAPALM

- **Network Automation and Programmability Abstraction Layer with Multivendor support**
  - Python library that implements a set of functions to interact with different router devices via unified API
  - Supported devices: https://napalm.readthedocs.io/en/latest/support/index.html
    - ***EOS, Junos, IOS-XR, IOS, NX-OS, etc.***

“Getters” Support

	EOS	IOS	IOSXR	JUNOS	NXOS	NXOS_SSH
get_arp_table	✓	✓	✓	✓	✓	✓
get_bgp_config	✓	✗	✓	✓	✗	✗
get_bgp_neighbors	✓	✓	✓	✓	✓	✓
get_bgp_neighbors_detail	✓	✓	✓	✓	✗	✗
get_config	✓	✓	✓	✓	✓	✓
get_environment	✓	✓	✓	✓	✗	✗
get_facts	✓	✓	✓	✓	✓	✓
get_firewall_policies	✗	✗	✗	✗	✗	✗
get_interfaces	✓	✓	✓	✓	✓	✓
get_interfaces_counters	✓	✓	✓	✓	✗	✗
get_interfaces_ip	✓	✓	✓	✓	✓	✓
get_ipv6_neighbors_table	✗	✓	✗	✓	✗	✗
get_lldp_neighbors	✓	✓	✓	✓	✓	✓
get_lldp_neighbors_detail	✓	✓	✓	✓	✓	✓
get_mac_address_table	✓	✓	✓	✓	✓	✓
get_network_instances	✓	✓	✗	✓	✗	✗
get_ntp_peers	✗	✓	✓	✓	✓	✓
get_ntp_servers	✓	✓	✓	✓	✓	✓
get_ntp_stats	✓	✓	✓	✓	✓	✗
get_optics	✓	✓	✗	✓	✗	✗
get_probes_config	✗	✓	✓	✓	✗	✗
get_probes_results	✗	✗	✓	✓	✗	✗
get_route_to	✓	✗	✓	✓	✗	✗

“Getting” Device Info & Print (JSON)

```
1  import json
2  from napalm import get_network_driver
3
4  #get_network_driver "name" is from NAPALM documentation (i.e. 'ios' in this example
5  driver = get_network_driver('ios')
6  iosvl2 = driver('192.168.122.72', 'a', 'a')
7
8  #opens the SSH connection
9  iosvl2.open()
10
11 #uses a standard function (get_facts) from NAPALM documentation to gather facts and prints in JSON
12 ios_output = iosvl2.get_facts()
13 #prints output from "get_facts" function in JSON format
14 print (json.dumps(ios_output, indent=4))
15
16 #uses a standard function (get_interfaces) from NAPALM documentation to get all interfaces and prints in JSON
17 ios_output = iosvl2.get_interfaces()
18 print (json.dumps(ios_output, indent=4))
19
20 #uses a standard function (get_interfaces_counters) from NAPALM documentation to gather counters and prints in JSON
21 ios_output = iosvl2.get_interfaces_counters()
22 print (json.dumps(ios_output, indent=4))
23
24 #uses a standard function (ping) from NAPALM documentation and pings (from device) and prints output in JSON
25 ios_output = iosvl2.ping('google.com')
26 print (json.dumps(ios_output, indent=4))
27
28 #closes the SSH connection
29 iosvl2.close()
```

Device Config. Audit & Changes

```
access-list 100 permit icmp any any
access-list 100 permit tcp any any eq domain
access-list 100 permit tcp any any eq www
access-list 100 permit tcp any any eq 443
```

```python
import json
from napalm import get_network_driver
driver = get_network_driver('ios')
iosvl2 = driver('192.168.122.72', 'a', 'a')
iosvl2.open()

print ('Accessing 192.168.122.72')

#opens the configuration (CLI commands) file ('ACL.txt')
iosvl2.load_merge_candidate(filename='ACL.cfg')

#uses the (compare_config) method to determine if the configuration has the commands from the text file
diffs = iosvl2.compare_config()

#if statement to determine if commands are present
if len(diffs) > 0:
    #prints the lines changed with a '+' in front of the command
    print(diffs)
    #commands (<= 1) from file are present, so commit them to the unit
    iosvl2.commit_config()
else:
    print('No ACL changes required.')
    #all commands are present, so don't implement commands
    iosvl2.discard_config()

iosvl2.close()
```

Device Config. Audit & Changes (multiple)

```
1  import json
2  from napalm import get_network_driver
3
4  devicelist = ['192.168.122.72',
5                '192.168.122.73'
6                ]
7
8  for ip_address in devicelist:
9      print ("Connecting to " + str(ip_address))
10     driver = get_network_driver('ios')
11     iosv = driver(ip_address, 'a', 'a')
12     iosv.open()
13     iosv.load_merge_candidate(filename='ACL1.cfg')
14     diffs = iosv.compare_config()
15     if len(diffs) > 0:
16         print(diffs)
17         iosv.commit_config()
18     else:
19         print('No ACL changes required.')
20         iosv.discard_config()
21
22     iosv.load_merge_candidate(filename='ospf1.cfg')
23
24     diffs = iosv.compare_config()
25     if len(diffs) > 0:
26         print(diffs)
27         iosv.commit_config()
28     else:
29         print('No OSPF changes required.')
30         iosv.discard_config()
31
32     iosv.close()
```

Lab: Pseudo Tips

- **Verify file exists**
- **Open "login" file**
  - Read username, password, IP address into elements
- **Verify the IP address is a valid IP address**
- **Verify the IP address is reachable**
- **SSH to device**
- **Open "commands" file**
  - Start at beginning of file
    - ***Create for loop and "send all commands"***
  - Verify the commands do not produce CLI syntax errors
- **Receive output**
- **Print in user-friendly format**
- **Save running-config**
- **Backup config**
- **Close all files**
- **Close SSH connection**

Network Automation - Tshooting

- **Valid IP address**
- **IP connectivity from device hosting script (i.e. ping)**
- **SSH enabled**
- **Valid UN/PW**
- **Interfaces enabled**
- **Commands = correct syntax**
- **Try manually first!**

Questions?