
University of Colorado Boulder

Network Management and Automation

DevOps & ZTP

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

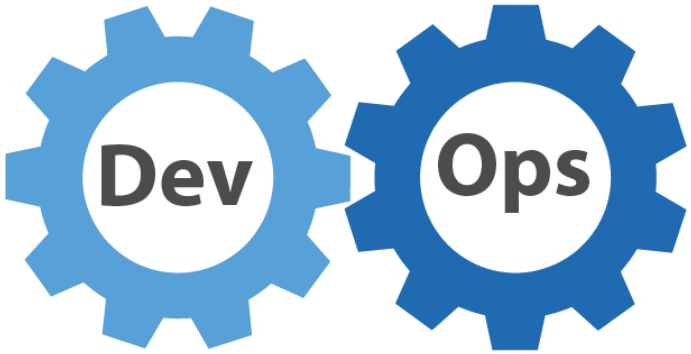
Review

- **Workload & Internships**
  - Cheating
- **In-class Challenge (two weeks)**
- **Orchestration & Abstraction**
- **Network Automation**


DevOps

What is Development and Operations (DevOps)?

– “It is difficult to nail down because it is a movement. That means everyone can say what it means to them.” – Puppet Labs CEO



What is DevOps?

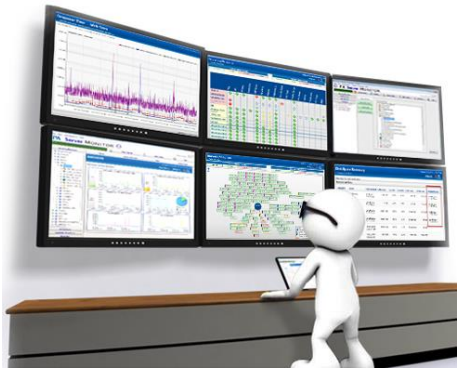
- **First, we need to understand Time to Market (TTM)**
  - The length of time it takes from idea to product
    - ***Technology/Online/Realtime services need to be deployed rapidly***
    - ***Faster TTM = Competitive Advantage***

Developers

- New Products
- New Features
- Security Updates
- Bug Fixes
- Have to wait “weeks” for work to be put into production
  - ***Have to manage: old code, pending code, new products, new features, etc.***
  - ***When finally deployed – runtime errors***
    - Production network and development environment !=
    - “It works on my machine”

System Admin & Network Operations

- 5 9's uptime
  - ***99.999%***
- Manages constantly growing number of servers (company is growing)
- Usually new code/feature/implementation on production network has errors, so they need to schedule "monthly" maintenance window
  - ***Have to determine "what went wrong" with the new code on the network***



Add Feature(FW)/Configuration(VNP) – X# sites

Again - What is DevOps?

- **Development and Operations working together!**
  - Change mindset (movement)
    - ***Work better together - on the same team***
    - ***Think more alike about goals***
    - ***Share responsibilities***

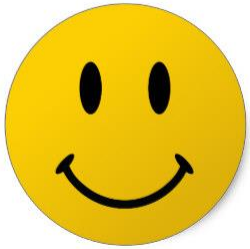


Diagram DevOps Model

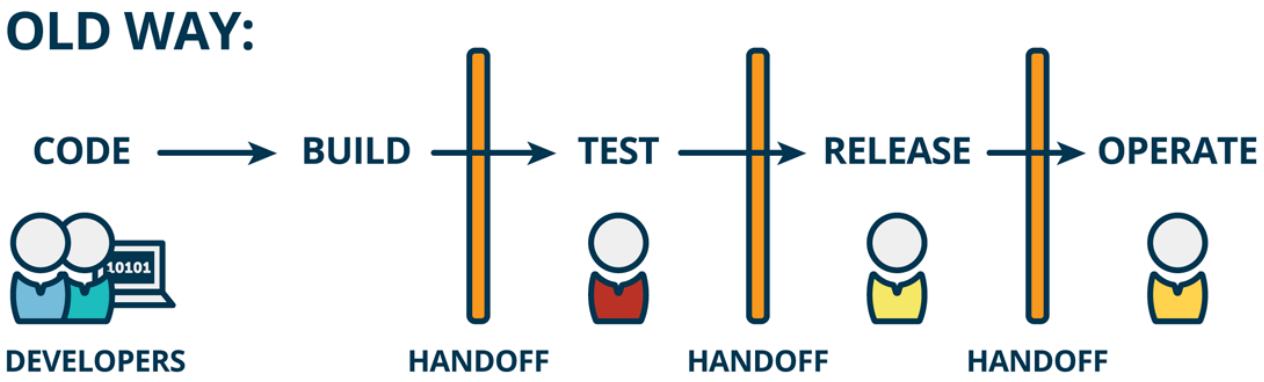
- **Developers**
  - Create code
  - Create servers
    - ***Deploy apps***
  - Update smoothly and rapidly
  - Share responsibilities!
- **SysAdmin/Ops**
  - Busy - Don't have time to setup environment for developers
  - Manage servers
  - Manage network
  - Share responsibilities!

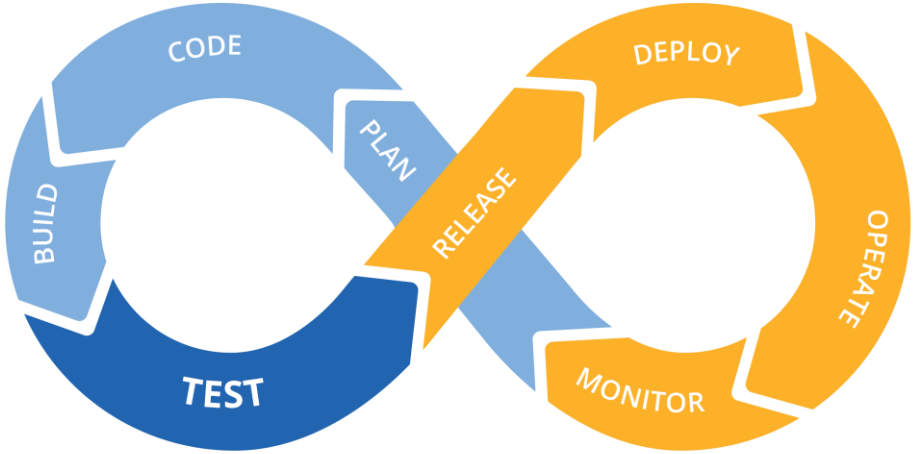
DevOps Solution

Why DevOps?

- **Improve collaboration/productivity by automating infrastructure and workflows**
  - ***Measuring performance and reevaluating***

- **Automation**
  - Code testing
  - Workflows
  - Infrastructure
  - EVERYTHING


OLD WAY:
CODE
BUILD
TEST
RELEASE
OPERATE
10101
DEVELOPERS
HANDOFF
HANDOFF
HANDOFF


CODE
PLAN
BUILD
TEST
RELEASE
DEPLOY
OPERATE
MONITOR

Benefits of DevOps

- **Faster TTM!**

- **Code in small pieces (feature, bug fix, etc.) and integrate & test**
  - Instead of large software "package" and testing everything
    - ***Now the lab modules make sense!***
      - Build and manage small tools

Benefits of DevOps

- **Versionable**
  - Change management
    - ***GitHub***
  - Records/audit
    - ***Iterative process***
      - Monitor, measure, improve code & operations

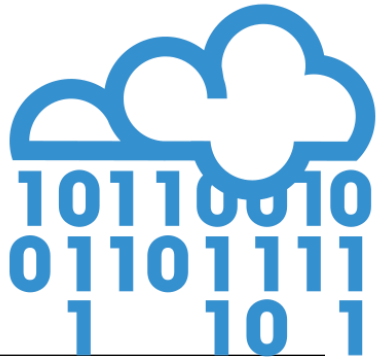
- **Repeatable**
  - Templates

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

- **Rapid deployment**
  - Production and development environment are identical!!
  - GNS3; Mininet; Bench test; lab

Goal of DevOps

- **“Single” point of control**
  - VM connected to OOB network (why?)
    - ***Remember redundancy (vMotion, backup, etc.)***
  - Install automation tool to manage configs
    - ***Could be as simple as .py programs***
    - ***Puppet, Chef, Ansible, Saltstack, NAPALM, etc.***
      - Configuration files are created as templates
  - Network Automation Management Station (NAMS)
    - ***All configuration changes***
      - VLAN, view route table, etc.



- **Run infrastructure as code (IaC)**

How?

- **Change in mindset + add new automation tools**

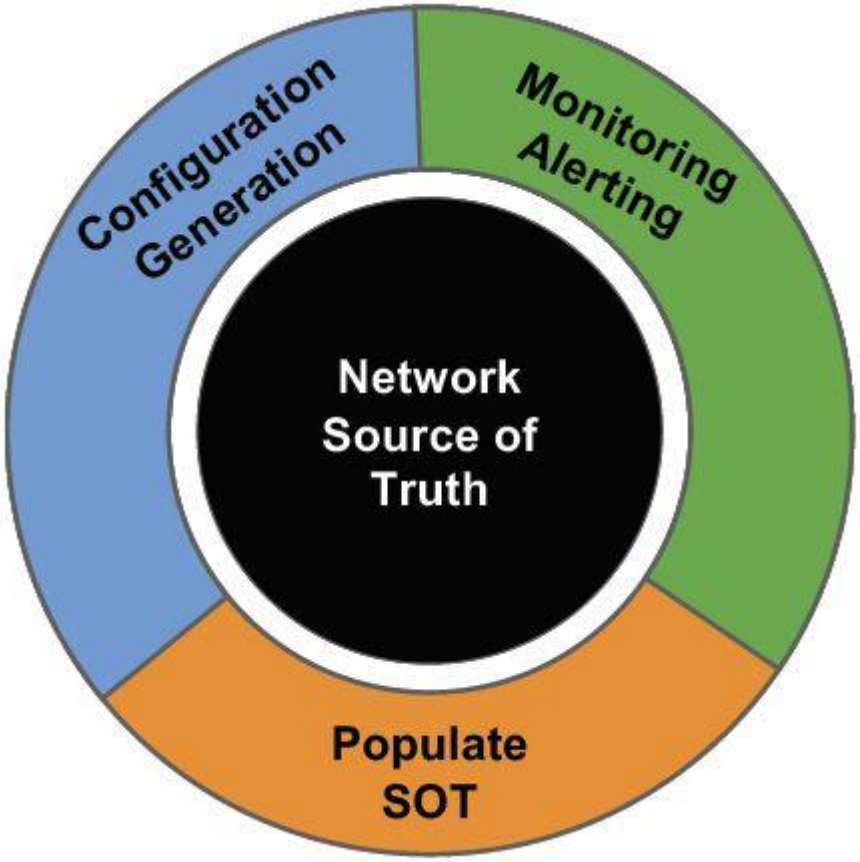
- **Source Control – GitHub**
  - What is SC?

- **Tools to build and test code**
  - Jenkins
- **Configuration Management**
  - How is this achieved?

- **Continuously manage/monitor network**

Network Source of Truth (NSOT)

- **Integrate <u>everything</u> with the NSOT**
- **The Source of Truth is only as good as the quality of the data it contains**
  - Introduces dependencies and requirements, but most important part!



NSOT Device Properties

- **Each device has unique set of properties**
  - Properties reflect the network design
- **Device Properties (template) = Variables**
  - Name / location
  - IP address
  - Cabling / Peer properties
  - Data Center layout
  - VLANs
  - BGP Peering
  - Device/vendor specific info
- **A whole new paradigm of network design!**

```yaml
name: "rsw1-1-sfo"
elevation: 30
type: qfx5100
role: rack-switch
ASN: "<ASN::sfo/private>"
network:
  lo0.0:
    ips:
      - addr: "<LO4::sfo/internal-loopbacks>"
      - addr: "<LO6::external-loopbacks>"
p2p:
  et-0/0/48:
    peer: "<DEV_INT::psw1-sfo/rack-switch>"
    ips:
      - addr: "<NET_IP4::sfo/point-to-point/31>"
  et-0/0/49:
    peer: "<DEV_INT::psw1-sfo/rack-switch>"
    ips:
      - addr: "<NET_IP4::sfo/point-to-point/31>"
```

```yaml
name: "rsw1-1-sfo"
elevation: 30
type: qfx5100
role: rack-switch
ASN: 65100
network:
  lo0.0:
    ips:
      - addr: 10.10.10.1/32
      - addr: 2020:1234:beef::756/128
p2p:
  et-0/0/48:
    peer: psw1-sfo::et-0/0/1
    ips:
      - addr: 10.128.195.124/31
  et-0/0/49:
    peer: psw2-sfo::et-0/0/1
    ips:
      - addr: 10.128.195.126/31
```

Device Versions

- **Be prepared to manage many versions of your device properties**

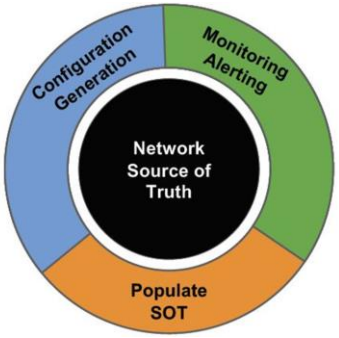
- **Follows coding logic: "if, then, else"**
  - For every rule, there is an exception.
  - So you always follow the rule, except when there is an exception.
  - In which case you follow a new rule based on that exception.

Resource Manager

- **Design and manage all <u>resources</u> like managing IPs with DHCP**
  - Possible to reserve resource in advance
  - Each resource allocated is associated with an ID
  - Same ID always get the same response
  - Example:
    - ***What = loopback***
    - ***Which = "location" Denver***
    - ***Who = device 1***
    - ***Response = 10.10.10.1/32***

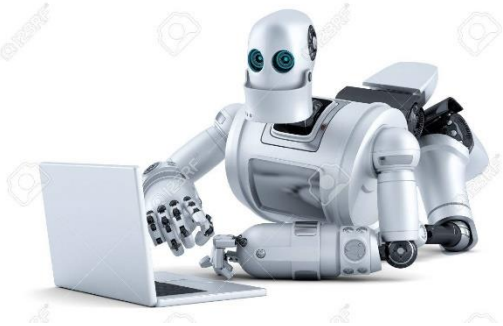
```
name: "rsw1-1-sfo"
elevation: 30
type: qfx5100
role: rack-switch
ASN: "<ASN::sfo/private>"
network:
  lo0.0:
    ips:
      - addr: "<LO4::sfo/internal-loopbacks>"
      - addr: "<LO6::external-loopbacks>"
p2p:
  et-0/0/48:
    peer: "<DEV_INT::psw1-sfo/rack-switch>"
    ips:
      - addr: "<NET_IP4::sfo/point-to-point/31>"
  et-0/0/49:
    peer: "<DEV_INT::psw1-sfo/rack-switch>"
    ips:
      - addr: "<NET_IP4::sfo/point-to-point/31>"
```

NSOT Process



- Front End
- IPAM (CSV/Database/Vars/Resources)
- Template
- Configs
- Automation/Orchestration

DevOps - Operations First Mindset



- **Engineers build robots . . .**
  - Robots manage the network!
- **Think operations first – features second**
- **Failproof vs Failfast**
  - Hardware is going to fail
  - Bugs in software
  - People make mistakes
  - Focus on detection failure and mitigation

Operations First Mindset - Testing

- **Do you have enough network capacity to handle a complete failure (data center)?**
  - Migration!
- **Continuously take down network and test**
  - Faster more dependable recovery
- **Cycle = Automation – Dependence - Validation**

- **Umm, yeah, automate everything, but don't we have to manually configure network hardware?**

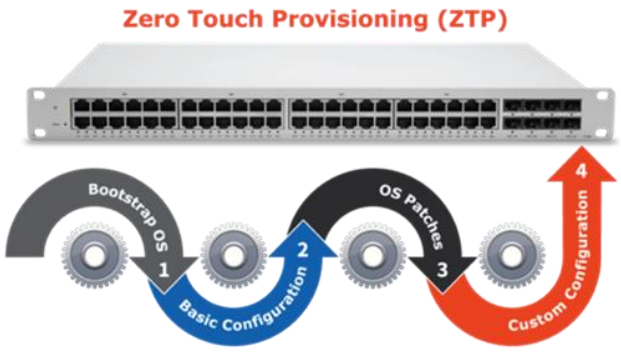
Zero Touch Provisioning (ZTP)

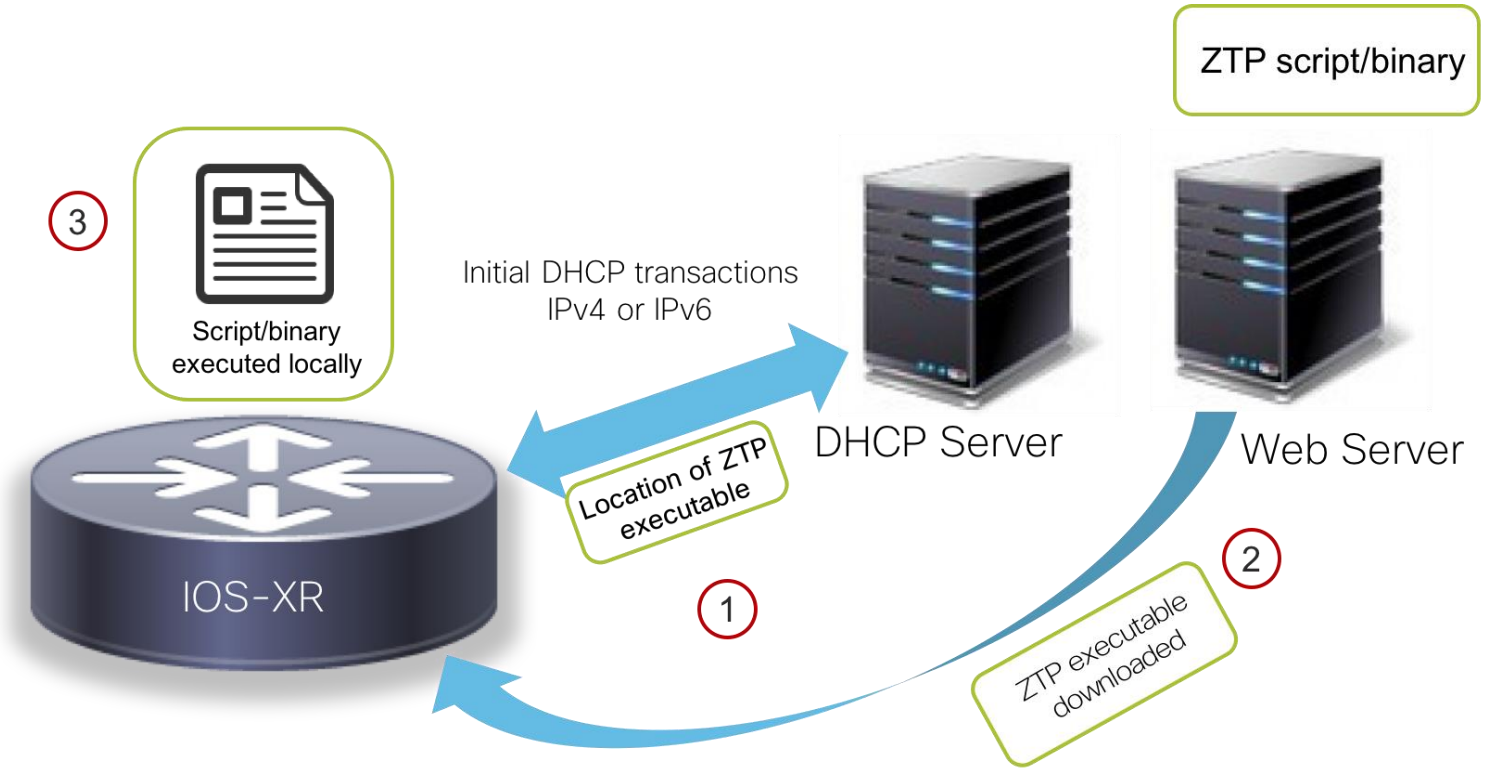
- **Automation of initial device configuration by providing ZTP script or CLI configuration file via DHCP (Offer)**
  - New installation
  - Rip and replace switches
    - ***Take care of themselves***
  - Avoid manual CLI configuration (as much as possible)

- **What is one drawback of automation in network engineering hardware environment?**

How does ZTP work?

- **Boot Up**
- **Obtain IP address (DHCP)**
  - Options
- **Execute script on local device (or download config)**
  - Note: after it runs the script sets a flag so it won't run again by default





- When the router receives a CLI configuration file, it will replace its existing configuration with the configuration it downloads as part of the ZTP process.

- When the router receives a script (bash or python), it will execute the downloaded script which is responsible for applying a config to the router or perform any Linux operation on the system.

ZTP - DHCP

- **DHCP Options**
  - Option number varies between vendors
    - ***i.e. Option 239 = Cumulus; Option 67 = Arista; option 61 "client identifier" = serial number***
  - Script file can be in various locations
    - ***FTP***
    - ***URL***
    - ***TFTP***
  - IMPORTANT: Host scope
    - ***Specific MAC address/serial number obtains /32 or /128 IP address***
      - MAC/ S/N printed on box & on machine
- ***Almost identical process to VoIP phone***

Benefits of ZTP

- **Efficiency**
  - Dramatically speeds up deployment/replacement
- **Consistency**
  - Alleviates manual/human errors
- **Scalability**
- **Visualization**

What to ZTP?

- **Configure SSH**
  - Network automation!
    - ***Ansible or .py***
- **DNS**
- **Hostname**
- **Time zone (NTP)**
- **Licenses**
  - Cumulus; ASA; etc.
- **Install other utilities**
  - Servers
    - ***Apt-get updates***
    - ***Security patches***
  - DevOps agent: Puppet, Chef, etc.
    - ***Look for Puppet master > Grab config. > configure switch***
- **NSOT and Device Parameters**

ZTP Switch Replacement Example

- **Switch 1 Fails**
  - Remove from network
- **Switch 2 (replacement)**
  - Plug in to out of band management network
  - Power up
  - DHCP > IP > Options
    - ***Asks for ZTP script***

ZTP – Hands Off

- **ZTP not only does initial configuration, but much more!**
  - Upgrade firmware version on device
  - Load configuration file

- **Run any additional scripts!!!**

Day 0, Day 1, Day 2

- **Day 0**
  - Day 0 apply base operation configurations

- **Day 1**
  - Day1 operations would comprise any service or feature activation or application deployment that is orchestrated on the devices
    - ***after the Day0 operations have applied a base configuration on the routers***

Day 2

- **Day 2**
  - Day2 operations tend to vary across deployments since they are tied to the nature of the business model and the operator's core focus.
    - ***infrastructure provider***
      - the network itself is the most important entity that needs to be protected from failures and is also the source of telemetry/monitoring information for alarms and/or remediation decisions
    - ***application provider***
      - the network is primarily a plumbing mechanism and the health of the applications running in the datacenters is the core focus. typically involve real-time traffic engineering or route manipulation.

Questions?

Lab – DevOps

- **Front End**
- **Database**
- **Build Config**
- **Diff Config**
- **Migration**

Lab

```
# import the module that SSH to device and either enters commands file and/or backs up configs
from lp_open_ssh_conn import *
# import the module that checks if file paths are correct and file exists
from lp_verify_file import *
# import the module that checks IP connectivity
from lp_ip_connectivity import *
# import the module that validates an IPv4 address
from lp_ip_is_valid import *
# Display help menu, and pass arguments to the csv to know what rows/cells to retrieve
import argparse
# parse the csv file
import csv
# Allows excecuting at the same time
import threading

##########################################ARGPARSE MENU
parser = argparse.ArgumentParser(description='SSH to Device & Save Configurations')
parser.add_argument('details_file', help="Enter csv file path for (IP address, UN, PW)")
parser.add_argument('--commands_file', help="Read a txt file with commands", action="store_true")
parser.add_argument('--save_config', help="Save the running config", action="store_true")
parser.add_argument('--valid_ip', help="Validate proper IPv4 address", action="store_true")
parser.add_argument('--ip_connectivity', help="Test IP connectivty (ping)", action="store_true")
args=parser.parse_args()
##########################################ARGPARSE MENU

# create global variable
global cmd_file
if args.details_file and args.commands_file:
    # Call this function in module (lp_verify_file) to verify the file/s exist
    cmd_file = verify_file(args.details_file, args.commands_file)
elif args.details_file:
    # Call this function in module (lp_verify_file) to verify the file/s exist
    verify_file(args.details_file)

if args.valid_ip:
    # call the function in module (lp_ip_is_valid) to check if the IP addresses in the CSV are valid
    for each_value in ip_table:
        ip_is_valid(each_value)
if args.ip_connectivity:
    # call the function in module (lp_ip_connectivity) to check if the IP addresses are reachable (ping)
    for each_value in ip_table:
        ip_connectivity(each_value)
if args.commands_file:
    # call the function in module (lp_open_ssh_con) to SSH to the device and pass the variable commands
    for a,b,c in zip(ip_table, username_table, password_table):
        open_ssh_conn(a,b,c, cmd_file)
if args.save_config:
    # call the function in module (lp_open_ssh_con) to SSH to the device and pass the variable save
    temp_save = 2
    for a,b,c in zip(ip_table, username_table, password_table):
        open_ssh_conn(a,b,c, temp_save)
```

```
################################FILE VERIFICATION
# Pass this function the argparse arguments (csv with IP,UN,pw & optional commands txt file)
# Using the "isfile" function (passed file path) from the "os.path" module to verify the file exists
# This doesn't check the contents of the file
import os.path
import sys

def verify_file(details_file, *cmd_file):
    #print(details_file, cmd_file)
    if os.path.isfile(details_file) == True:
        print("\n** CSV file %s location has been validated....\n" % details_file)
        return True
    else:
        print("\n\n\n\n***ERROR!!!: File %s does not exist! Please check and try again!\n\n\n\n" % details_file)
        return False
        #sys.exit()

    for a in cmd_file:
        if a == True:
            cmd_file = input("Enter command file: ")
            if os.path.isfile(cmd_file) == True:
                print("\n** Command file %s location has been validated....\n" % cmd_file)
            else:
                print("\n\n***ERROR!!:  File %s does not exist! Please check and try again!\n\n" % cmd_file)
                sys.exit()
        return cmd_file
            #commands_txt_file = cmd_file
################################FILE VERIFICATION
```

```
# Checking IP address file and IP address validity
def ip_is_valid(ip):
    a = ip.split('.')
    print("\n** Checking if IP address %s is valid \n" % ip)
    if (len(a) == 4) and (1 <= int(a[0]) <= 223) and (int(a[0]) != 127) and (int(a[0]) != 169 or int(a[1]) != 254) and (0 <= int(a[1]) <= 255 and 0 <= int(a[2]) <= 255 and 0 <= int(a[3]) <= 255):
        print("\n** IP %s is validated\n" % ip)
        return True
    else:
        print('\n\n*** ERROR:  There was an INVALID IP address (%s)! Please check and try again!\n\n' % ip)
        print("\n###Quiting Program###\n")
        return False
        #sys.exit()
```