PYTHON DOCUMENT

Advanced Networking Solutions

ABSTRACT

How to master implementing Next-Generation programmability in your LAN, WAN & Datacenter to have a fast paced Software-Defined Network up and running perfectly in no time.

Ahmed Ossama

Nexus VPC Automation - Implementation & Programmability Lab

Direct from the Expert

Nexus VPC Automation

First Edition

Introduction

The purpose of this document is to provide a complete description about the proposed networking automation using Python.

The document will provide a step-by-step guide on how to implement Nexus upgrade, initial configuration & VPC domain configuration without touching the switches command line interface using Python application interactive interface.

Please refer to Cisco press books + Python certified books for a solid understanding about the theories behind this solution.

Python is one of the major new programmability coding languages nowadays used in large next generation networking solutions based on implementation automation.

This lab is intended to provide a full enterprise solution consists of the following technologies:

- LAN (***L***ocal ***A***rea ***N***etwork).
- Datacenter.

This lab will help in getting a solid understanding of those technologies, which include specifically the following:

- POAP (***P***ower ***O***n ***A***uto ***P***rovisioning).
- SDN (***S***oftware-***D***efined ***N***etwork)
- VLAN (***V***irtual ***L***ocal ***A***rea ***N***etwork).
- VRF (***V***irtual ***R***outing & ***F***orwarding).
- DHCP (***D***ynamic ***H***ost ***C***onfiguration ***P***rotocol).
- TFTP (***T***rivial ***F***ile ***T***ransfer ***P***rotocol)
- LACP (***L***ink ***A***ggregation ***C***ontrol ***P***rotocol).
- VPC (***V***irtual ***P***ort ***C***hannel).

A complete understanding of the theories will help moving forward with this lab in a perfect way.

There is no much time like earlier days to configure devices manually especially with the standardized configuration required over multiple networking devices.

Automation is required in everyday task to facilitate faster working with day-to-day activities.

The first thing to understand automation is to define the best programming language that will be used for automation.

Python as open language for packages & libraries implement endless type of applications in unlimited no. of field technologies like the following:

- Networking.
- Automation.
- Big data & analytics.

- Data modeling & sampling.

Adding to Python scripts, the use of YANG (***Y***et ***A***nother ***N***ext ***G***eneration) along with NETCONF (***NET***work ***CONF***iguration) & RESTAPI (***REST***ful ***A***pplication ***P***rogramming ***I***nterface) as certified by IETF (***I***nternet ***E***ngineering ***T***ask ***F***orce) provides the required tool for this modeling and device interaction.

A lot of examples regarding YANG & Python which is out of the scope of this lab will help in the implementation of this lab.

Some examples of those automation scripts are as listed below:

- Devices bootstrapping & initialization.
- BGP automation.
- OSPF automation.
- LISP (***L***ocator / ***I***dentity ***S***eparation ***P***rotocol) configuration automation on map resolvers & map servers.
- Configuration export for backups.

NETCONF is using RPC (***R***emote ***P***rocedure ***C***all) which is a great protocol to interact with network elements & creates two datastore for the configuration as below:

- Candidate store: proposed configuration.
- Running store: working configuration.

Anyone dealing with Cisco devices will understand the great benefit for that protocol (do you know why?).

In our lab here, we will be using Python to define all the required programmability & automation tasks as a DMD (***D***ata-***M***odel ***D***riven) as well as a coding software to the appliances.

It also requires some hands-on experience with the mentioned technologies to support the implementation of this datacenter programmability setup.

Understanding datacenter topologies implemented with VPC is something mandatory nowadays.

A network engineer with fair networking knowledge is the minimum required skillset to run / understand this lab.

Ahmed Ossama

Master experience in R&S (***R***outing & ***S***witching), Datacenter, Security and Programmability & Automation.

https://Linkedin.com/in/aossama

Table of Contents

Document Control 8
Datacenter Topology 9
Enterprise Network Schema 12
Solution Overview 13
Datacenter Network Configuration 14
Section 1: Why Python 14
Section 2: Solution Flowchart 17
Section 3: Application Structure 22
Section 4: Phase-1 Meet The Application 25
Section 5: Phase-2 Checking For Upgrades 28
Section 6: Phase-3 POAP 29
Section 7: Phase-4 Device Initialization 34
Section 8: Phase-5 VPC Configuration 38
Section 9: Phase-6 Complementary Configuration 40
Section 10: Phase-7 Finalizations & Confirmations 41
Best Troubleshooting Ways 44
Complete Python Application 46
References 64
Thank You Message 65

Table of Tables

Table 1 Document Version 8
Table 2 GNS3 Simulation Devices 9
Table 3 Connectivity 12
Table 4 IP Addresses 12
Table 5 Master Application Packages 15
Table 6 Master Application Pre-requisites 25
Table 7 Master Application Questionnaire Evaluation 25

Table of Figures

Figure 1 Network Diagram 9
Figure 2 CMD As Administrator 16
Figure 3 Run switchosos.py 16
Figure 4 Mater Application Flowchart – Part 1 17
Figure 5 Mater Application Flowchart – Part 2 18
Figure 6 Mater Application Flowchart – Part 3 19
Figure 7 Mater Application Flowchart – Part 4 20
Figure 8 Mater Application Flowchart – Part 5 21
Figure 9 Dependencies Flowchart 22
Figure 10 NexusSwitchAuto Folder Components 23
Figure 11 server Folder Components 23
Figure 12 Master Application Score Bar 26
Figure 13 Master Application Start Page 27
Figure 14 Checking Laptop IP Address 27
Figure 15 Upgrade File Confirmation 28
Figure 16 Upgrade File Missing 28
Figure 17 Upgrade File MD5 Passed 28
Figure 18 Upgrade File MD5 Failed 28
Figure 19 POAP Script Flowchart 30
Figure 20 POAP Script First Line 31
Figure 21 POAP Script First & Second Lines 31
Figure 22 POAP Script MD5 Check 32
Figure 23 POAP Script MD5 Calculation BY Master Application 32
Figure 24 POAP Script Calculated MD5 Validation 32
Figure 25 Nexus Switch POAP Initialization 33
Figure 26 Nexus Switch POAP Script Execution Status 33
Figure 27 Nexus Switch Starting With New Software Image 33
Figure 28 Nexus Switch Python Preparation – Initial Configuration 34
Figure 29 Nexus Switch – Initial Configration Applied 34
Figure 30 Nexus Switch – Checking Logging 35
Figure 31 Nexus Switch Management Interface 36

Figure 32 POAP Server Logs 36
Figure 33 Switches Reachability Check By Master Application 37
Figure 34 Switches Software Version Check By Master Application 38
Figure 35 Nexus Switch Feature Installation 38
Figure 36 Master Application Questionnaire – 1 Part 1 39
Figure 37 Master Application Questionnaire - 2 39
Figure 38 Master Application Questionnaire – 1 Part 2 40
Figure 39 Master Application Login To Nexus Switch 40
Figure 40 Nexus Switch Configuration Notifications 41
Figure 41 Switch Reachability Test After Applying Full Configuration 41
Figure 42 Nexus Switch VPC Status – P art 1 42
Figure 43 Nexus Switch VPC Status – P art 2 43
Figure 44 Master Application Last Message 43
Figure 45 POAP Logs File 44
Figure 46 POAP Packet Capture 45
Figure 47 Master Application – Part 1 46
Figure 48 Master Application – Part 2 46
Figure 49 Master Application – Part 3 46
Figure 50 Master Application – Part 4 47
Figure 51 Master Application – Part 5 47
Figure 52 Master Application – Part 6 48
Figure 53 Master Application – Part 7 48
Figure 54 Master Application – Part 8 49
Figure 55 Master Application – Part 9 49
Figure 56 Master Application – Part 10 50
Figure 57 Master Application – Part 11 50
Figure 58 Master Application – Part 12 51
Figure 59 Master Application – Part 13 52
Figure 60 Master Application – Part 14 52
Figure 61 Master Application – Part 15 53
Figure 62 Master Application Run 53

Document Control

The following table will maintain the consistency of regular updates to the document.

Each update is added here.

Author: Ahmed Ossama

Table 1 Document Version

Version	Issue Date	Status	Reason for Change
1	16-Feb-20	Active	Initial Version
1.1	19-Feb-20	Update	Updated Information
1.2	26-Feb-20	Update	Updated Information
2.0	15-Mar-20	Update	Major Update
2.1	16-Mar-20	Update	Added Configuration
2.2	20-Mar-20	Update	Updated Information
3.0	30-Mar-20	Closed	First Edition

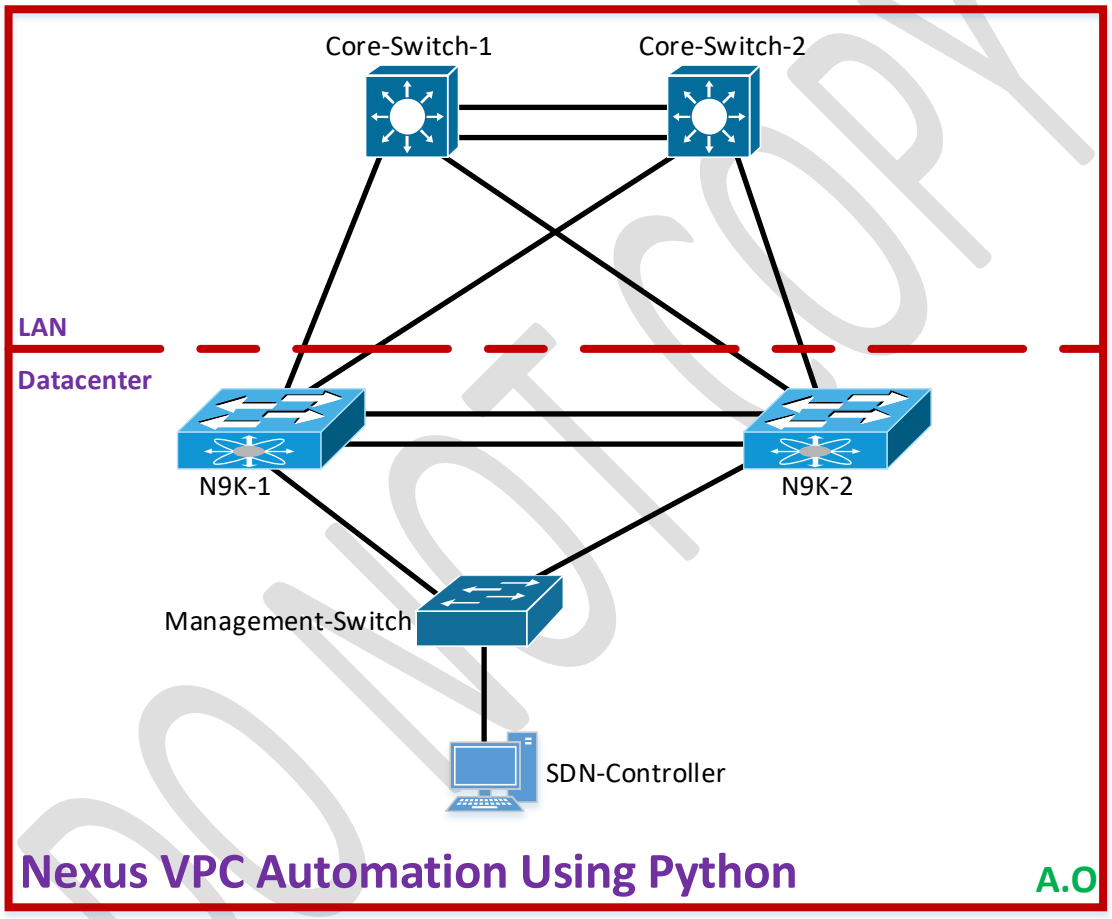
Datacenter Topology

Topology is something mandatory to know the connectivity matrix between all devices in a representable diagram.

We have a simple topology here representing the required setup for our software lab.

The following is the topology used for the designated lab, which is represented by the physical connectivity between all devices:

Figure 1 Network Diagram



This topology had been built using GNS3 (***G***raphical ***N***etwork ***S***imulator) application (headend) running on Microsoft Windows 10 computer connected to "GNS3 VM (***V***irtual ***M***achine)" (backend) running on VMware ESXi 6.7 with the following QEMU / containers:

Table 2 GNS3 Simulation Devices

No.	Device	Container
1	Datacenter Switches	nxosv.9.3.1.qcow2

QEMU (***Q***uick ***EMU***lator) is a fast low computing power piece that implements software using containers running on a base operating system.

Using the same versions of software will guarantee running the lab without any issues & achieving the same results mentioned in this document.

Topology scenario:

At one of the big partners, there is a datacenter team who is implementing everyday many of the Nexus switches at different clients & they decided to automate the implementation of VPC domain on the Nexus switches for the following reasons:

- They are implementing hundreds of Nexus switches / day, so, they need a faster way to initialize the switches.
- They need to introduce a way for non-implementation engineers to prepare the Nexus appliances for a production network without much effort to understand the Nexus switches / datacenter technologies.
- The result of using the application is to have a consistent Nexus configuration across all the partner clients'.
- Very fast way to implement a lot of switches in less time (target of automation).

The topology consists of the following locations:

- Datacenter:
  - Two Nexus VPC datacenter switches.
  - One management switch.
  - One SDN controller.
- LAN:
  - Two Catalyst core switches.

The following is what needs to be achieved as an output from the topology:

- Boot strapping the two Nexus switches.
- Configuring the management interfaces on both Nexus switches.
- Upgrading the Nexus switches to Cisco stable release.
- Configuring VPC domain between the two Nexus switches.
- Configuring uplinks from the Nexus switches to the LAN zone.
- Configure switching the traffic from Nexus to Catalyst switches.
- Configure routing the traffic to the nearest gateway on the Nexus switch despite the FHR configuration.
- Confirming a working VPC domain.

Achieving this great implementation requires a systematic configuration approach to the required technologies.

Quick explanations to some of the Python terminologies that will be used throughout this lab are as below:

- OOP: **O**bject ***O***riented ***P***rogramming.
- PIP: ***P***ython ***I***nstallation ***P***ackage.
- >>>: Python default prompt interactive shell.
- …: Python default prompt indented block interactive shell.

We will be going through this lab according to the following procedure:

- Why Python.
- Solution Flowchart.
- Application Structure.
- Phase-1: Meet The Application.
- Phase-2: Checking For Upgrades.
- Phase-3: POAP.
- Phase-4: Device Initialization.
- Phase-5: VPC Configuration.
- Phase-6: Complementary Configuration.
- Phase-7: Finalizations & Confirmations.

Let us start now & we will follow the below methodology to get a solid understanding of what is implemented:

- Divide the enterprise network to sections.
- Configure each section separately.
- Integrate sections with each other to achieve a complete enterprise setup as required.

This also will help in understanding the solution in a much better & easier way.

Enterprise Network Schema

Network schema for any topology consists of the following:

- Connectivity matrix.
- Addressing information.

The following table highlights the connectivity matrix for the mentioned topology:

Table 3 Connectivity

No.	Device-1	Port	Device-2	Port
1	N9K-1	e1/1	N9K-2	e1/1
2		e1/2		e1/2
3		e1/3	Core-SW-1	TenGig1/0/1
4		e1/4	Core-SW-2	TenGig2/0/1
5	N9K-2	e1/1	N9K-1	e1/1
6		e1/2		e1/2
7		e1/3	Core-SW-1	TenGig1/0/2
8		e1/4	Core-SW-2	TenGig2/0/2
9	Mgmt-Switch	port1	N9K-1	mgmt0
10		port2	N9K-2	
11		port3	SDN-Controller	e0

The following table highlights the IP addresses used for the mentioned topology:

Table 4 IP Addresses

No.	Device	Interface	IP Address
1	N9K-1	mgmt0	10.10.1.1
2	N9K-2		10.10.1.2
3	SDN-Controller	e0	192.168.101.253

This topology is built using IPv4 (***I***nternet ***P***rotocol ***V***ersion ***4***) to support the POAP operation.

Solution Overview

The datacenter technical team at Cisco gold partner decided to implement a programmability coding application to simplify their day-to-day activities of initializing Cisco Nexus switches as VPC domain to connect different servers (physical or virtual), networking appliances as well as FEX's (***F***abric ***EX***tenders) to the network.

The first thing you need to understand here is what is a VPC domain?

In Cisco Nexus switches (an open software switches that can be used with any networking software & not locked to Cisco Nexus software), you can implement one aggregation domain consisting of two different control planes to segregate the failure domain & control packets transfer.

VPC domain is the de-facto standard if you need to implement stacking of two Nexus switches.

The application that is implemented here will help in automating the initial configuration of any Nexus switch according to the flowchart that will be discussed later in the "Application Flowchart" section.

The application will make a use case of ZTP (**Z**ero ***T***ouch ***P***rovisioning) programmability & network automation.

As the application is a pure piece of software, so, the following is mandatory to notice:

- Bugs could arise.
- Open for enhancements.
- Open for use from others.
- Open for integration with third party applications.

This applications is licensed under GPL (***G***eneral ***P***ublic ***L***icense) licensing.

SDN controller is the required tool to do all the required automation & programmability which can be totally built as open-source on Ubuntu or simply using Python on Windows 10 machine as in our lab here.

Actually, using Ubuntu (a Linux flavor), you can build all the following & much more:

- Cloud computing system.
- Virtual programmable SAN's (***S***torage ***A***rea ***N***etwork).
- Programmable network.

Here we will be using Python language as a self-constructing application coding to implement the requirements of this lab.

Let's start and dive into the application pieces.

Datacenter Network Configuration

All datacenter switches will be initially configured through management interfaces according to the following:

- Management 0.

Management interface 0 will be used all the time for switches reachability & configuration.

Section 1: Why Python

The first thing to build our application which will login to the Nexus switches & configure it in a very intuitive & interactive way is to choose the programming & coding language.

So, what is the best coding language to implement the required application?

Many coding languages are exist in the technology field nowadays, all of them are good, you can not specify something is good and another is bad but definitely there is better languages over another.

One of the best languages used for programmability & automation is ***Python*** because it is supporting the object oriented programming + hierarchy of the application structure with almost all kind of libraries for everything required to be configured / handled & automated.

Those packages (libraries) are even got updated using PIP to overcome bugs on older versions & adding more features to the package which lead to a more versatile programming language.

Dealing with different types of devices around the world requires defining a standard data-model driven that could be easily interpreted & understood by human and to deal with any vendor networking element which could be easily implemented also in ***Python***.

Another important reason for using ***Python*** is the wide-spectrum of information availability on the internet or on books to develop different applications.

There is no need to memorize any command-syntax in ***Python*** coding at all, just prepare your thoughts, write them down & start the coding by either searching for the packages format (attributes & objects), finding the code structure using package help or even using tools that help find the functions inside any package.

You can use the following tools to help you build a Python application:

- ***Python*** 2.7.5.
- ***Python*** 3.7.2.
- Pycharm Community Edition.

A good practice in ***Python*** is to divide your coded application into pieces to get the following outcome:

- Simplify the coding.
- Less errors.
- Greater dependencies.
- Small master application code file.

- Reducing required computation cycle (CPU (***C***entral ***P***rocessing ***U***nit) operation).
- Not all the code will be installed in memory, only the piece required to run now → lowering the RAM (***R***andom ***A***ccess ***M***emory) requirements.

A field notice is to start ***Python*** coding using Pycharm tool for Windows to make you very familiar with the language in a very fast way.

You can download the community (free version) from the below link:

- https://www.jetbrains.com/pycharm/download/#section=windows.

With ***Python***, you can define the following & much more:

- Variables from almost any type.
- Functions.
- Classes.
- Boolean decisions.
- Loops.

An example of ***Python*** variables are: Numbers, Strings, Lists & Dictionary variables.

An example of ***Python*** functions are: add_first & add_second functions in our lab.

An example of ***Python*** classes are: add_first & add_second classes in our lab.

An example of ***Python*** decisions are: If statement boolean decision.

An example of ***Python*** loops are: With & For loops.

The following table highlights the libraries (packages) used throughout this application:

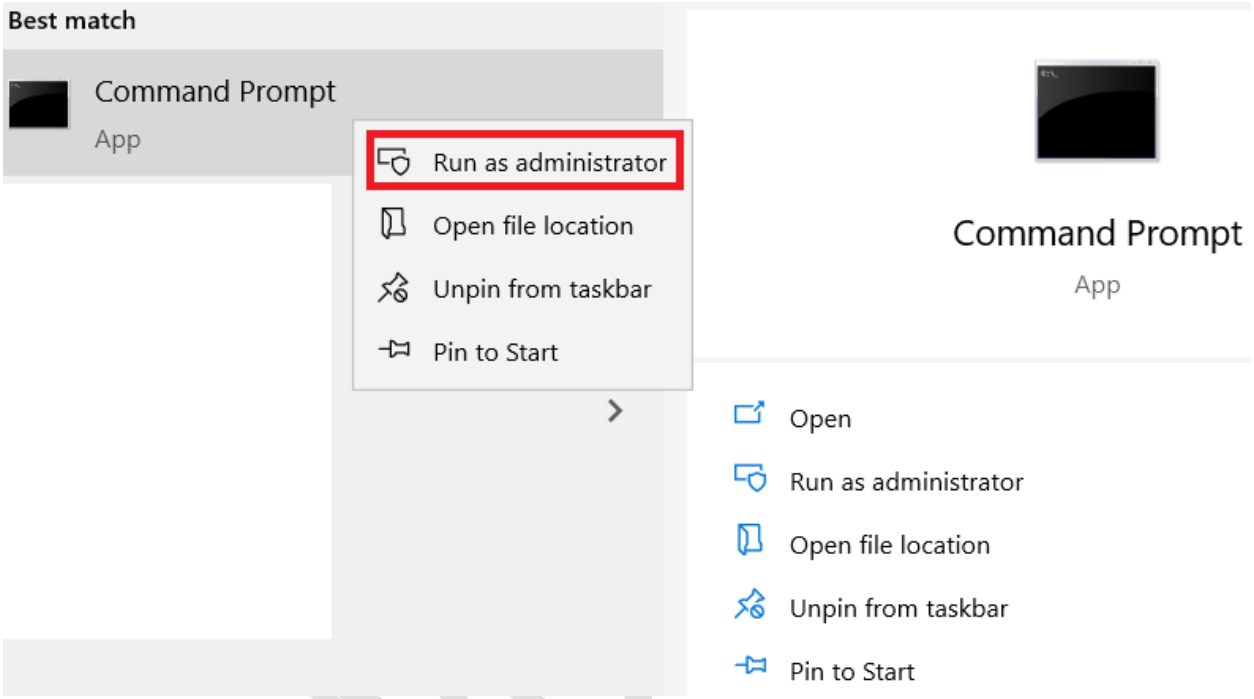
Table 5 Master Application Packages

No.	Package	Used For
1	from pythonping import ping	Ping
2	import paramiko	SSH
3	import time	Timer
4	import os	Operating System Configuration
5	import socket	TCP / IP Stack Interaction
6	import shutil	Manipulate Folders & Files
7	import signal	Operating System Process Interaction
8	import subprocess	Operating System Run Commands
9	import getpass	Password Encryption
10	import hashlib	MD5 Calculation
11	from add_first import add_first	Self-Prepared Add First Line
12	from add_second import add_second	Self-Prepared Add Second Line

I have developed this application to run on Windows 10 machines & it is a must to run it as administrator on your laptop to overcome the UAC (***U***ser ***A***ccess ***C***ontrol) stopping any feature or configuration in the application.

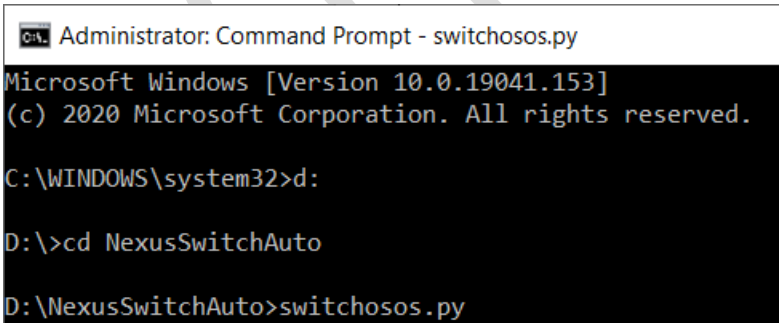
To run the application as administrator, first you need to run CMD as administrator as in the screenshot below:

Figure 2 CMD As Administrator



Then, go to the location of the folder "NexusSwitchAuto" which contains the application & run the Python file "switchosos.py" as in the screenshot below:

Figure 3 Run switchosos.py



Section 2: Solution Flowchart

The flowchart is the first thing to build any software application.

A perfect flowchart will lead to a perfect working application with a minimum no. of bugs / errors.

In a nutshell, the flowchart will be translated to a Python code very easily, so, simply do not panic, just use your mental thinking and prepare a perfect working flowchart & with the help of any internet source, you will be able to develop the Python code easily.

The flowchart is your first step to define a step-by-step "Application Structure" discussed in more details in the following section.

If you read the following flowchart carefully, you will understand the complete application perfectly without any more aggressive studies required.

The below screenshot showing the complete flowchart divided on pages for easy reading:

Figure 4 Mater Application Flowchart – Part 1

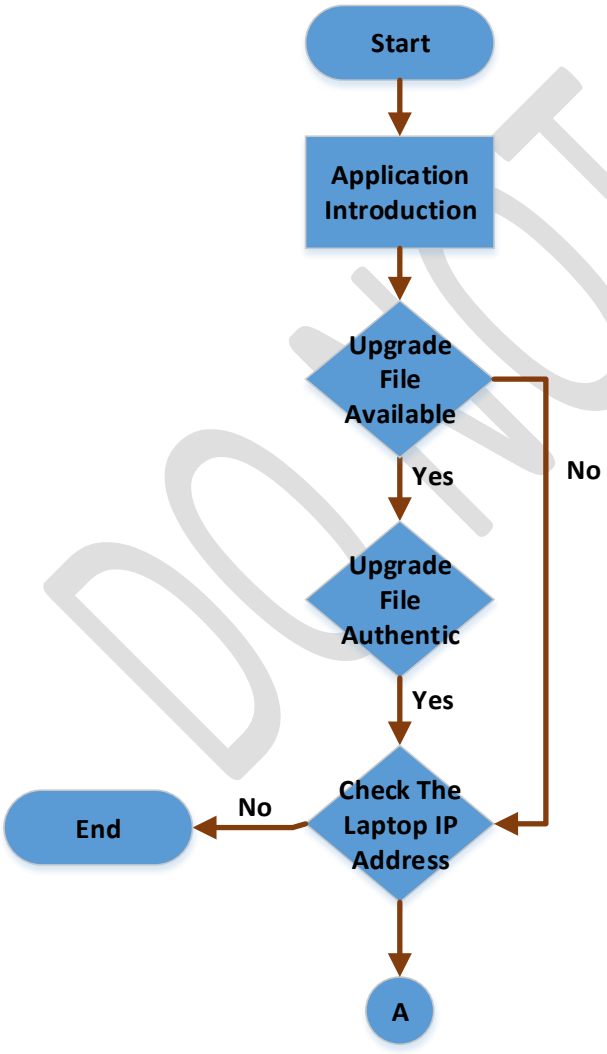


Figure 5 Mater Application Flowchart – Part 2

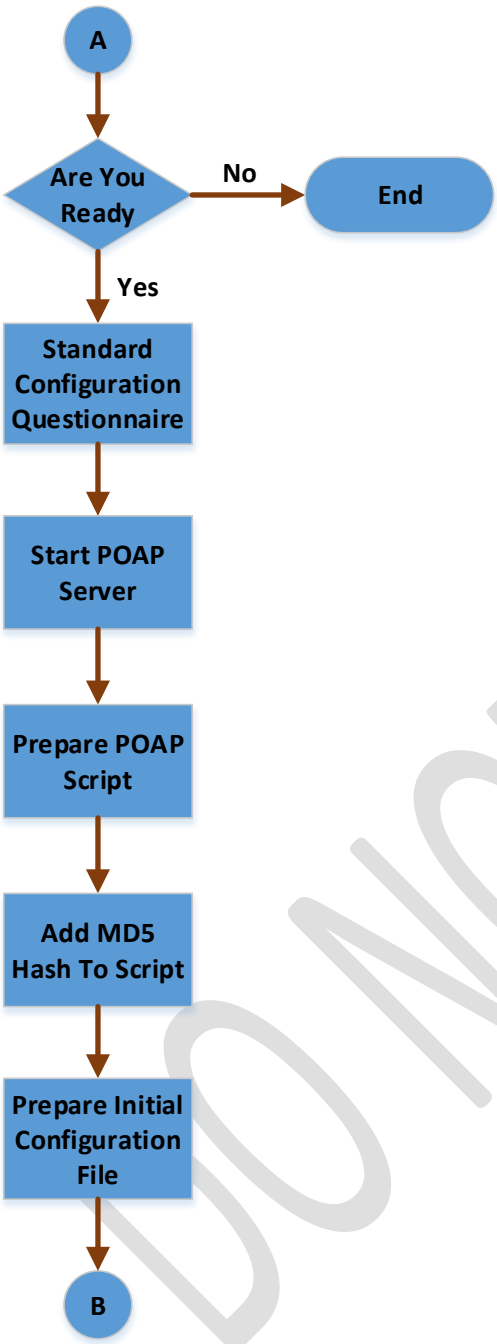


Figure 6 Mater Application Flowchart – Part 3

Figure 7 Mater Application Flowchart – Part 4

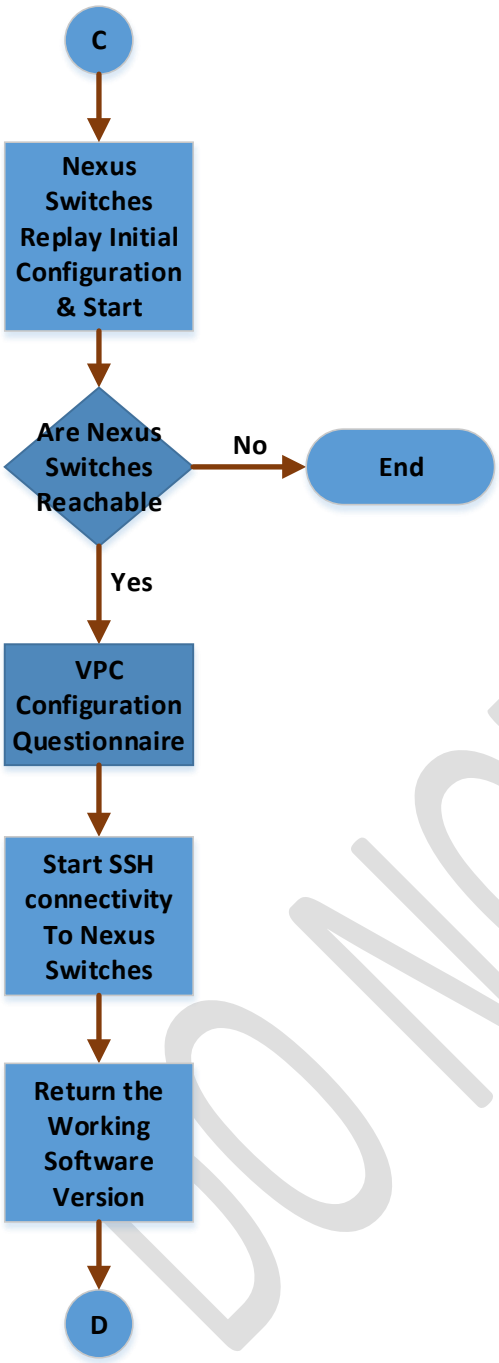
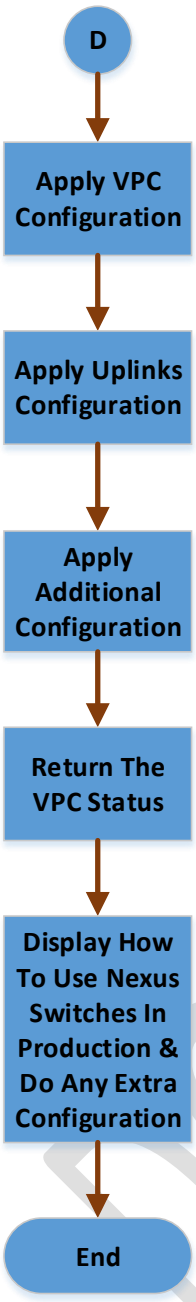


Figure 8 Mater Application Flowchart – Part 5



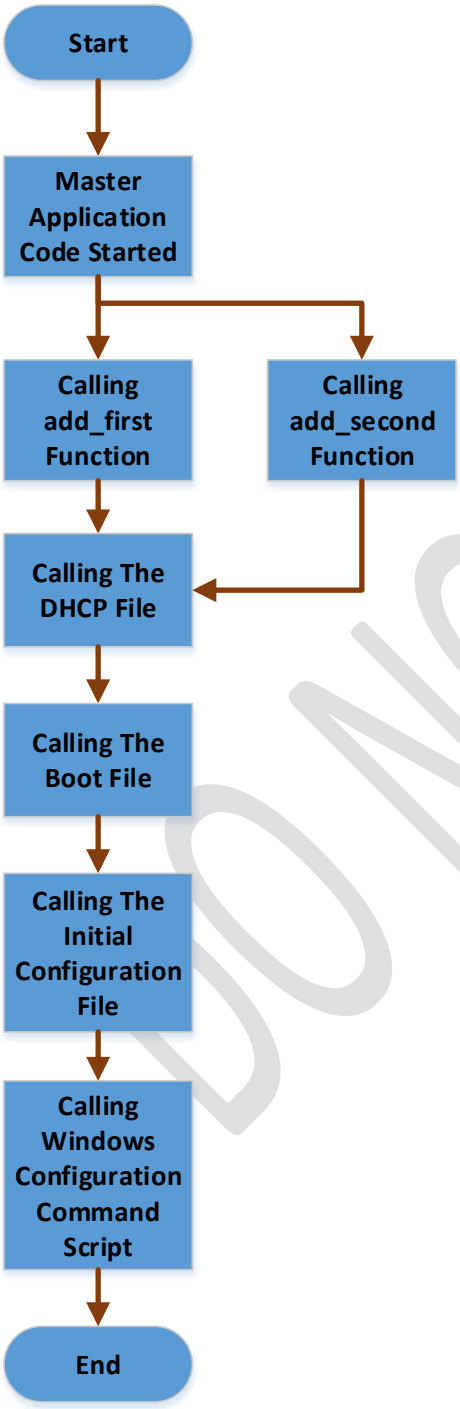
This is the complete flowchart that we will convert to a Python code.

Section 3: Application Structure

In this section we will understand the components of the application which in together will build the complete structure of the program.

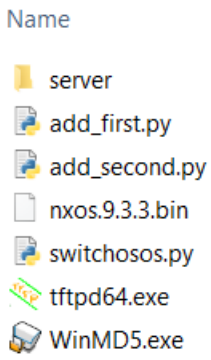
The application here had a lot of dependencies according to the following flowchart:

Figure 9 Dependencies Flowchart



The following screenshot showing the contents of the main folder of the application:

Figure 10 NexusSwitchAuto Folder Components

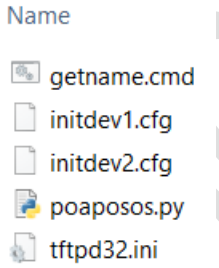


A quick description of the application folder files is as below:

- server: a folder contains the variable working files.
- add_first.py: a function & class to add text to the first line in a file.
- add_second.py: a function & class to add text to the second line in a file.
- nxos.9.3.3.bin: an upgrade software file (you have to put only 1 upgrade file in the application folder).
- switchosos.py: Python master application.
- tftpd64.exe: POAP server providing DHCP + TFTP + boot file.
- WinMD5.exe: application to compute the MD5 of any file contents.

The following screenshot showing the contents of the "server" subfolder of the application:

Figure 11 server Folder Components



A quick description of the server folder files is as below:

- getname.cmd: a command line Windows script to get the interface name for specific IP address configured on the laptop.
- initdev1.cfg: N9K-1 initialization configuration file.
- initdev2.cfg: N9K-2 initialization configuration file.
- poaposos.py: POAP script.
- tftpd32.ini: tftpd64.exe configuration file.

The following list highlights the bugs that will be thrown in case of failed dependencies on the application:

- If add_first function & class cannot be called by the master application (switchosos.py).
- If add_second function & class cannot be called by the master application.
- If tftpd32.ini cannot be called by tftpd64.exe.
- If poaposos.py cannot be called by tftpd64.exe.
- If tftpd64.exe cannot be called by the master application.
- If initdev1.cfg cannot be called by the master application.
- If initdev2.cfg cannot be called by the master application.
- If poaposos.py failed to bootstrap one of the Nexus switches.
- If poaposos.py failed to validate MD5 on one of the Nexus switches.
- If getname.cmd cannot be called by the master application.

The following list highlights the exit codes that will be thrown during the application execution:

- The user is not ready to use the application.
- The upgrade file is wrong.
- The MD5 of the upgrade file is wrong.
- The laptop IP address is wrong.
- The VPC domain question is answered wrong.
- The switches are not reachable after a complete bootstrap.
- The VPC peer links not entered during the questionnaire.
- You need to exit after your complete answer to the whole questionnaire.
- Reaching the end of the application (successfull finish).
- Expired timer after a non-completed task.
- Providing same IP address to be configured on both Nexus switches as a final configuration.

All timers in this application are adjustable & can be configured easily on the master application code file.

Also, all files in the server folder will be extracted in the main application folder during the use of the application.

Section 4: Phase-1 Meet The Application

In this section we will understand the first section of the application & how it will introduce itself to a non-technical datacenter engineer to do the required configuration on the appliances.

The application had some pre-requisites which must be met before starting the application, otherwise, you will lose the power of using this application.

The list of Pre-requisites is in the below table along with a column where you need to put your answer if you meet that requirement or not (your answer should be "Ready" or "Not Ready"):

Table 6 Master Application Pre-requisites

No.	Pre-requisite	Status
1	Nexus switches readiness	
2	Management switch connectivity	
3	Laptop IP Address 192.168.101.253/25	
4	Nexus switches factory default	
5	Aggregation using LACP only	
6	All firewalls are off on your laptop	
7	Software upgrade file in the application folder	

The first three pre-requisites are mandatory, the second two pre-requisites are highly recommended & the last two pre-requisites are highly preferred.

After answering the "Status" column, refer to the table below to know your overall score which will tell you the probability of the application to succeed or fail in the implementation of the VPC configuration:

Table 7 Master Application Questionnaire Evaluation

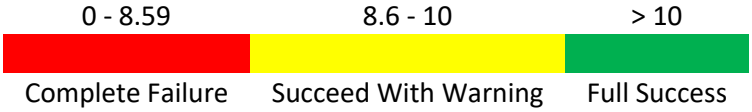
No.	Question	Ready Score	Not Ready Score
1	Question 1	20	0
2	Question 2	20	0
3	Question 3	20	0
4	Question 4	7	0
5	Question 5	7	0
6	Question 6	3	0
7	Question 7	3	0

You need to calculate your overall score with the formula below:

Total Score = (Q1 + Q2 + Q3 + Q4 + Q5 + Q6 + Q7) / 7

And refer to the coloring index below to know how much your application will succeed implementing the requirements:

Figure 12 Master Application Score Bar



For example if you answered the questions according to the following:

- Question 1: Ready.
- Question 2: Ready.
- Question 3: Not Ready.
- Question 4: Ready.
- Question 5: Ready.
- Question 6: Ready.
- Question 7: Ready.

Then your score will be = Total Score = (20 + 20 + 0 + 7 + 7 + 3 + 3) / 7 = 8.5714 which means a "Complete Failure".

Another example if you answered the questions according to the following:

- Question 1: Ready.
- Question 2: Ready.
- Question 3: Ready.
- Question 4: Not Ready.
- Question 5: Not Ready.
- Question 6: Ready.
- Question 7: Not Ready.

Then your score will be = Total Score = (20 + 20 + 20 + 0 + 0 + 3 + 0) / 7 = 9 which means a "Success With Warning".

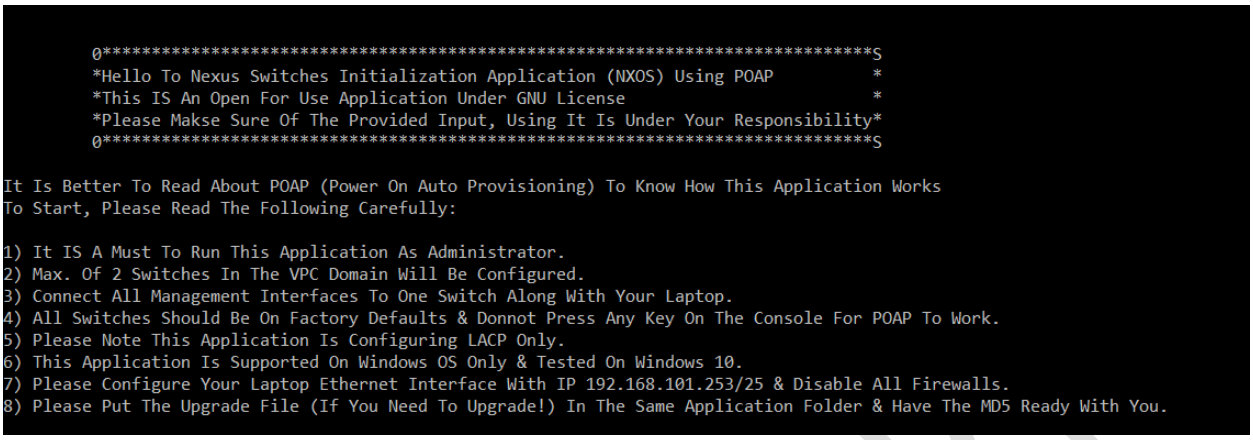
To have a "Full Success", please meet all the pre-requisites.

The introductory part of the application will give you a detailed explanation about the pre-requisites & how you will proceed with the application.

The introduction part is coded using the "print" Python function.

The Below screenshot showing the introduction interface of the application:

Figure 13 Master Application Start Page

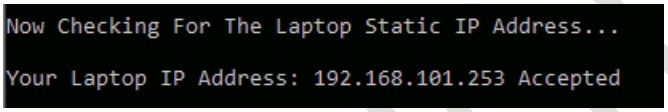


Only eight points that requires the user of the application to be fully aware of the process during the setup of the Nexus switches.

→ 192.168.101.253/25 = 192.168.101.253 255.255.255.128

After the startup page, the first check is to confirm the laptop IP address is configured statically with the IP address mentioned in the pre-requisite according to the screenshot below:

Figure 14 Checking Laptop IP Address



If the laptop static IP address is different from the required one, the application will silently exit with a notification message.

The above can be simply coded using a “get hostname” function in the Python application along with “if” statement to validate that IP address collected from the laptop interface.

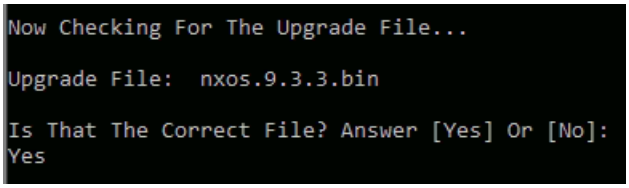
If the laptop static IP address had a different subnet mask, you may lose the connectivity throughout the automation of the Nexus switches.

Section 5: Phase-2 Checking For Upgrades

As part of the application flowchart is to search if there is a valid software upgrade available in the application folder.

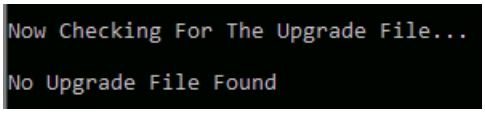
The second phase of the application is to check if there is a ".bin" software upgrade available in the folder according to the screenshot below:

Figure 15 Upgrade File Confirmation



If there is no valid software upgrade file ".bin" available in the application folder, so, the application will report that there is no software upgrade file available & no upgrade will be considered during the programmability of the infrastructure according to the screenshot below:

Figure 16 Upgrade File Missing



That part of the automation is simply checked using a search Python function (a built in library) to look into the application folder for any ".bin" file if it is exist.

Part of the automation is to confirm the checksum of the software upgrade file which you can copy from Cisco website & push it to the application question, so, the application will confirm the authenticy & integrity of the upgrade file according to the screenshots below:

Figure 17 Upgrade File MD5 Passed

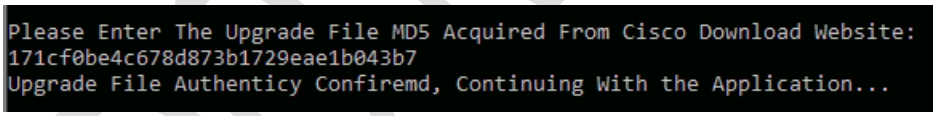
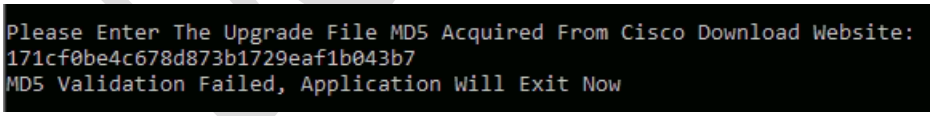


Figure 18 Upgrade File MD5 Failed



The above checks & validations could be done with very simple Python code using "i" statement which will search for the upgrade file as well as confirming the authenticy & integrity of the upgrade file.

Section 6: Phase-3 POAP

POAP agent is installed on all Nexus switches.

POAP application is required to initialize & automate the lunch of any Nexus appliance without touching it by your hand with the concept of zero touch provisioning.

The POAP application requires the following as a pre-requisite for a successfull operation of bootstrapping any appliance (Do you know bootstrapping?):

- DHCP server.
- TFTP server.
- Boot file.

The DHCP server will provide an initial IP address to the Nexus switch, so, it can start communication with the POAP application server.

The TFTP server will be responsible to download the boot file required to bootstrap the Nexus switch.

The boot file contains the piece of code required to bring the switch to a state that can continue working on the network till have the full startup configuration which is the VPC domain in our use case.

All Nexus switches are following the exact operation of the POAP application which is simply working on two things before the execution:

- Determining the software upgrade file & downloads it.
- Determining the configuration file & downloads it.

The boot file here is written in Python language which is a great feature in Nexus switches to accept the bootstrap file written in Python & executed as a shell script.

You can combine the DHCP server + TFTP server + boot file server in one application server (as in our use case) & consider it as the POAP application server.

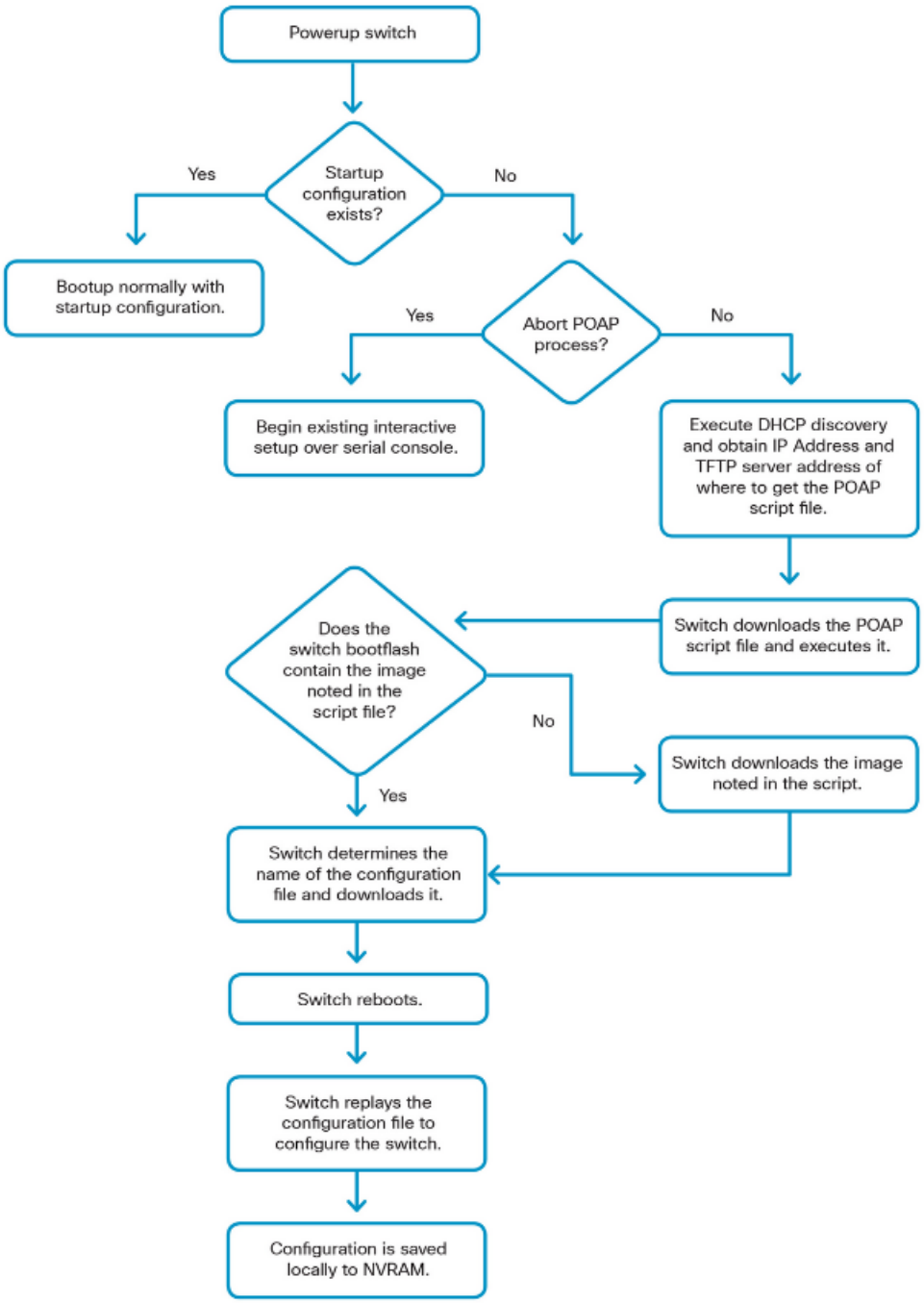
Bootstrap scripting is required to initialize any device with ZTP & without any interaction with the appliance (either physical or virtual) itself.

The script file is simply needs to achieve the following:

- Transfer the software upgrade file to the Nexus switch.
- Determine the switch role in the VPC domain & according to that:
    - Either send the primary role configuration file (N9K-1).
    - Or sending the secondary role configuration file (N9K-2).
- Configure the boot option with the new upgrade file.
- Schedule the initial configuration file to be applied after reboot.

The below screenshot is showing the flowchart of the POAP application operation:

Figure 19 POAP Script Flowchart



The POAP operation is implementing a strict flowchart for bootstrapping the Nexus switches, this operation is fixed & cannot be changed in any case.

The good thing is you can completely automate the generation of the POAP script (poaposos.py) using the master application file (switchosos.py) as in our lab use case.

This is mainly done for the following:

- Defining the management IP address acquired by the switch during the DHCP to decide which initial configuration file will be downloaded to the Nexus switch.
- Defining the variables outside the POAP script to make it as simpler as we can for faster switch bootstrap operation.
- Adding the required bootstrap operation commands to the POAP script file before calculating the MD5 checksum.

The challenging part regarding the POAP agent installed on the Nexus switches is it requires to find the MD5SUM (***M***essage ***D***igest ***5*** ***SUM***mation) hash on the second line in the script file.

The POAP script file must start with the following in the first line to inform the POAP agent that it is a bootstrapping scripting file:

Figure 20 POAP Script First Line

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

To achieve that, two functions had been created outside the application code (a great feature of Python coding) & called by the Python application code to do the mentioned above according to the following:

- add_first function which can add any statement to the first line of any file.
- add_second function which can add any statement to the second line of any file.

The add_first function will add the variable configuration to the POAP script before sending it to the switch & the add_second function will add the MD5SUM to the second line of the POAP script file, so, the POAP script file will contain the following in the first & second lines when it is totally configured & ready for distribution to the Nexus switches:

Figure 21 POAP Script First & Second Lines

```
#!/bin/env python
#md5sum="f3839851e1f1ccce856ae825b266daf9"
```

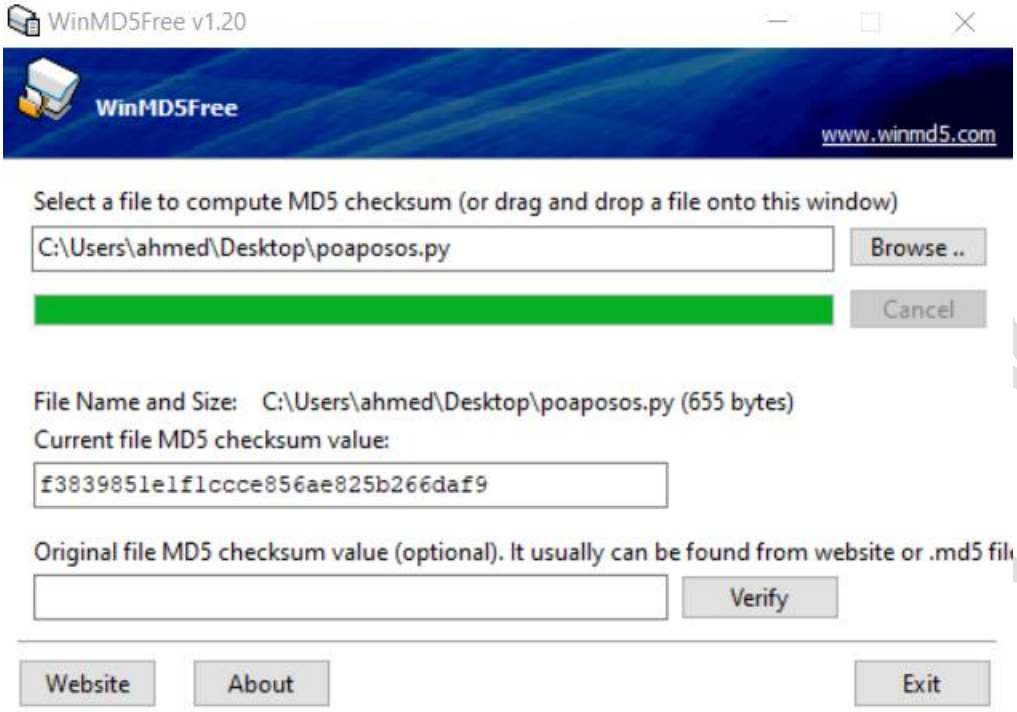
For sure you know that the challenge here is the MD5SUM should be calculated after all the required configuration is added to the POAP script & to make the challenge even more complex, that MD5SUM should be added to the second line not to the first one (from that you can expect the need for the add_second function).

If you didn't change any configuration in the POAP script file (poaposos.py), so, the MD5 checksum will be f3839851e1f1ccce856ae825b266daf9 which will be accepted by the Nexus switches.

To help you calculate the MD5 for any file, the application "WinMD5.exe" will present to you the MD5 in the field "Current file MD5 checksum value:".

This MD5 value is for the file contents & not for the file itself as in the screenshot below:

Figure 22 POAP Script MD5 Check

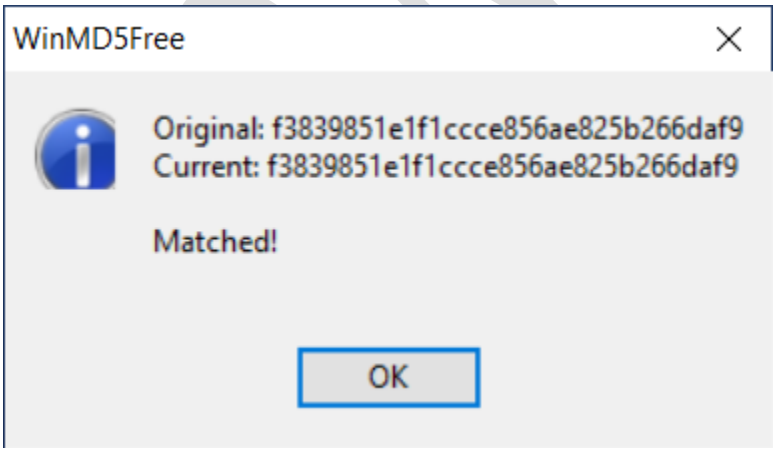


If you take the MD5 received from the Python application & verified it against the calculated by the "WinMD5.exe" application, you will get a matched MD5 which means that it will be accepted by the Nexus switch according to the screenshot below:

Figure 23 POAP Script MD5 Calculation BY Master Application

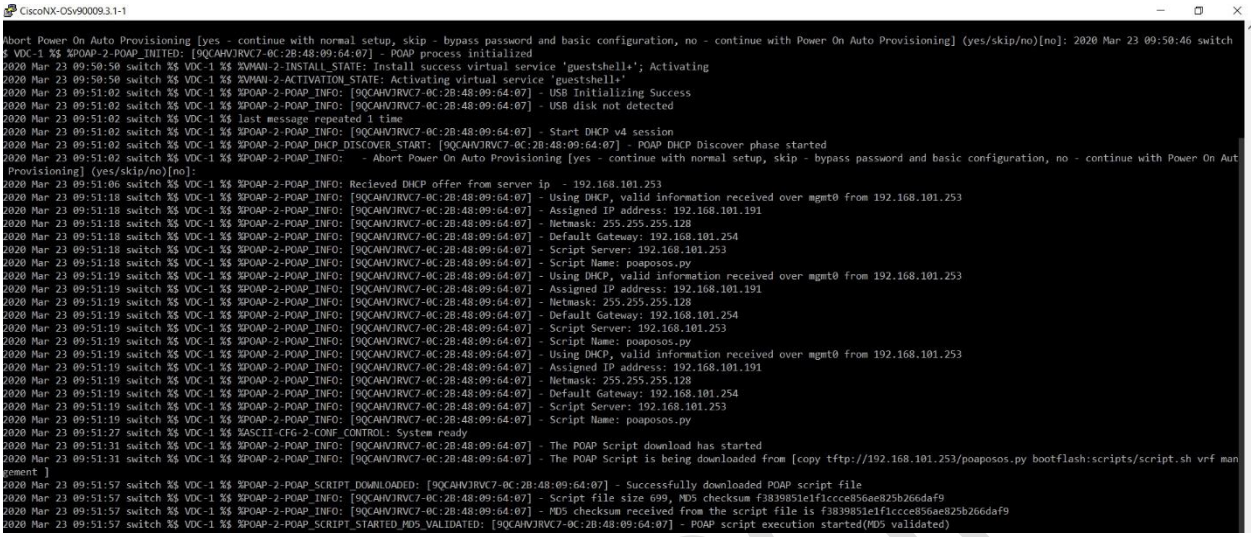
```
Current POAP File MD5SUM Is: f3839851e1f1ccce856ae825b266daf9 Please Keep It For Your Reference.
```

Figure 24 POAP Script Calculated MD5 Validation



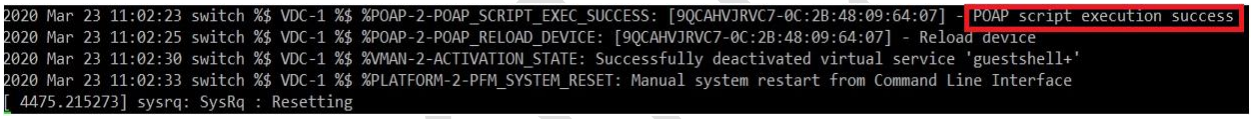
The following screenshot showing a successfull operation of the POAP agent:

Figure 25 Nexus Switch POAP Initialization



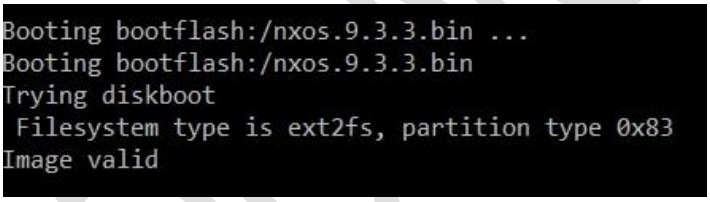
After a successfull download of the upgrade file, the Nexus switch will reboot automatically as in the screenshot below:

Figure 26 Nexus Switch POAP Script Execution Status



After the reboot, the Nexus switch confirmed a valid software upgrade file & did boot it according to the screenshot below:

Figure 27 Nexus Switch Starting With New Software Image



Now the switch had successfully executed the boot file downloaded from the POAP server & it will start the next phase operation.

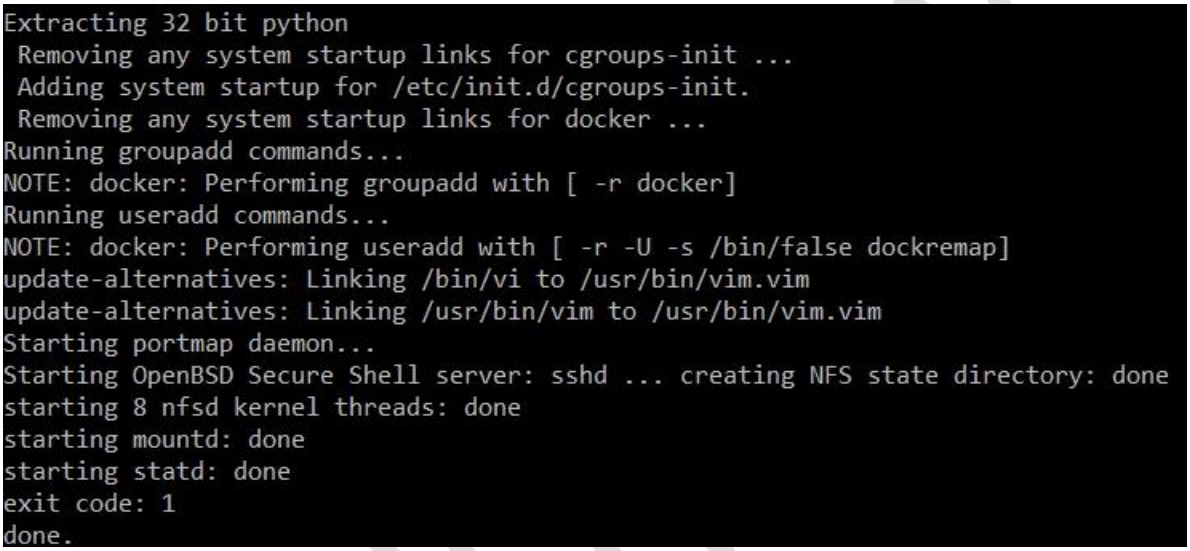
Section 7: Phase-4 Device Initialization

As mentioned in the flowchart in the previous phase, after a successfull execution of the bootstrap script file & after a long wait ☺ to download the software upgrade file (if exist), the switch do the following:

- Reboots to execute any software upgrade added in the script file.
- Apply the configuration file to the switch.

The switch will start the Python daemon again according to the screenshot below:

Figure 28 Nexus Switch Python Preparation – Initial Configuration

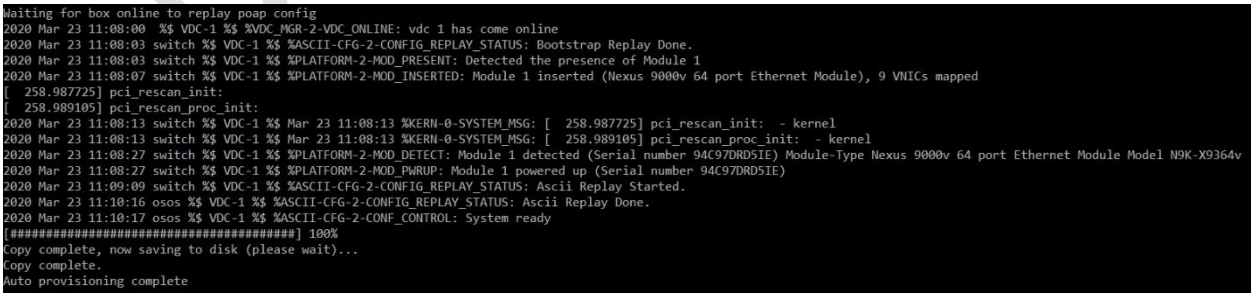


The initial configuration is a very simple configuration file needs to achieve the following:

- Apply a new temporary IP address to be used for the remaining configuration of the Nexus switch.
- Applying a temporary hostname to the Nexus switches.
- Applying a temporary username / password credentials to the Nexus switches.
- Adding the new software upgrade file to the permanent boot option.

The switch will start replaying & applying the configuration file according to the screenshot below:

Figure 29 Nexus Switch – Initial Configration Applied



The configuration file could be one of the following files as mentioned in the previous section “POAP”:

- initdev1.cfg for N9K-1.
- initdev2.cfg for N9K-2.

The screenshot below showing the switch that had successfully finished applying the ZTP configuration, passed the configuration prompt & ready for the VPC configuration:

Figure 30 Nexus Switch – Checking Logging

```
Auto provisioning complete


User Access Verification
osos login: cisco
Password:

Cisco NX-OS Software
Copyright (c) 2002-2019, Cisco Systems, Inc. All rights reserved.
Nexus 9000v software ("Nexus 9000v Software") and related documentation,
files or other reference materials ("Documentation") are
the proprietary property and confidential information of Cisco
Systems, Inc. ("Cisco") and are protected, without limitation,
pursuant to United States and International copyright and trademark
laws in the applicable jurisdiction which provide civil and criminal
penalties for copying or distribution without Cisco's authorization.

Any use or disclosure, in whole or in part, of the Nexus 9000v Software
or Documentation to any third party for any purposes is expressly
prohibited except as otherwise authorized by Cisco in writing.
The copyrights to certain works contained herein are owned by other
third parties and are used and distributed under license. Some parts
of this software may be covered under the GNU Public License or the
GNU Lesser General Public License. A copy of each such license is
available at
http://www.gnu.org/licenses/gpl.html and
http://www.gnu.org/licenses/lgpl.html
*******************************************************************************
*  Nexus 9000v is strictly limited to use for evaluation, demonstration   *
*  and NX-OS education. Any use or disclosure, in whole or in part of     *
*  the Nexus 9000v Software or Documentation to any third party for any   *
*  purposes is expressly prohibited except as otherwise authorized by     *
*  Cisco in writing.                                                      *
*******************************************************************************
osos#
```

At this point, the management interfaces are configured with the IP address in their respective initdev***X***.cfg (where ***X*** is equal to 1 or 2).

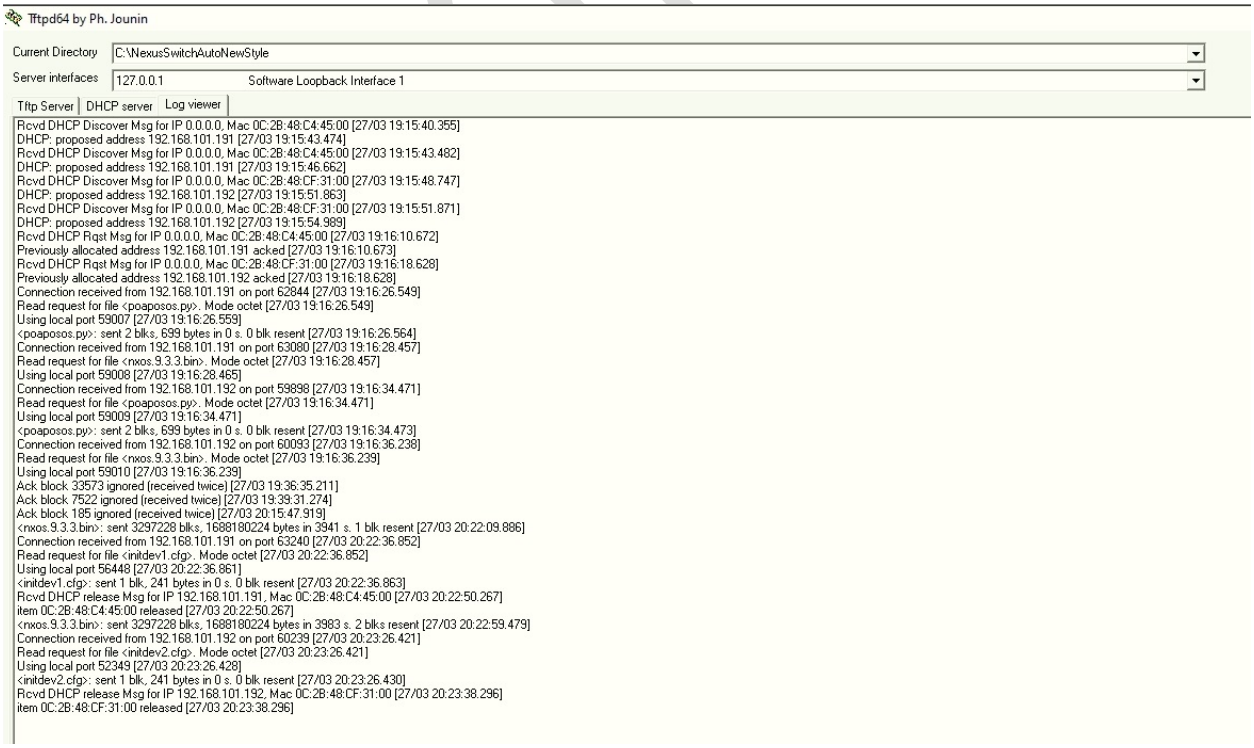
The screenshot below showing the IP address configured on the management interface on one of the Nexus switches:

Figure 31 Nexus Switch Management Interface

```
osos# show interface mgmt 0
mgmt0 is up
admin state is up,
  Hardware: Ethernet, address: 0c2b.48c5.e500 (bia 0c2b.48c5.e500)
  Internet Address is 192.168.101.182/25
  MTU 1500 bytes, BW 1000000 Kbit , DLY 10 usec
  reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation ARPA, medium is broadcast
  full-duplex, 1000 Mb/s
  Auto-Negotiation is turned on
  Auto-mdix is turned off
  EtherType is 0x0000
  1 minute input rate 416 bits/sec, 0 packets/sec
  1 minute output rate 32 bits/sec, 0 packets/sec
  Rx
    50490 input packets 88 unicast packets 48510 multicast packets
    1892 broadcast packets 4665856 bytes
  Tx
    1477 output packets 90 unicast packets 1386 multicast packets
    1 broadcast packets 328939 bytes
```

A complete look on the process execution logs on the POAP server can be seen as in the screenshot below:

Figure 32 POAP Server Logs



The important point here is to adjust the master application timer for the POAP server to accommodate the upload for the upgrade file from the TFTP server to the Nexus switch, usually this could be adjusted to 1800 seconds = 30 minutes.

The last step here for the Python application is to confirm the reachability to the Nexus switches after the complete bootstrap according to the screenshoot below:

Figure 33 Switches Reachability Check By Master Application

```
Now Pinging To Switches To Confirm Reachability, Please Wait...

Switch: 192.168.101.181  UP
Switch: 192.168.101.182  UP
Reachability Test Finished...
```

Pinging the switches for reachability is essential part to continue with the execution of the remaining part of the master application & now the switches are ready for the VPC configuration.

Section 8: Phase-5 VPC Configuration

VPC is a virtual port channel between totally two different control planes implemented on Nexus switches.

The VPC configuration includes the following:

- Defining a keepalive link between the switches which is the management interface according to the best practices.
- Defining the peer links between the Nexus switches which required for the traffic distribution & configuration status synchronization.
- Defining the VPC role for the switch.
- Defining the peer switch configuration if the Nexus switch is connected to normal spanning tree switches.
- Defining the peer gateway if first hop router will be implemented on the Nexus switches.

Please refer to Cisco website & Cisco press books to know more details about the VPC theory, configuration & operation.

The first thing after logging into the Nexus switches for the first time after the successful POAP operation is to show you the current software version of the switch, this such information will confirm if the requested upgrade was successfull or not as in the screenshot below:

Figure 34 Switches Software Version Check By Master Application

```
Getting Software Version From Switches, Please Wait...

Switch 1 Current Software Version:  9.3(3)
Switch 2 Current Software Version:  9.3(3)
```

Am using here “Paramiko” library to SSH (***S***ecure ***SH***ell) to the switch with the credentials just added to the configuration during the initial configuration bootstrapping on the Nexus switch.

“Paramiko” is the most stable library to SSH to IT (***I***nformation ***T***echnology) devices using Python coding format.

It is a normal SSH package that can send shell commands to any device accepting SSH access.

The master application will silently configure the required features to make the Nexus switch ready for VPC configuration according to the screenshot below:

Figure 35 Nexus Switch Feature Installation

```
osos# show feature | include vpc
vpc                   1          enabled
osos#
osos# show feature | include lacp
lacp                  1          enabled
```

There is one challenge here which is to find the software version in the output of “show version” which can be simply searched using the “index & replace” Python functions.

The below screenshot is showing the first part of the questionnaire that will be initiated by the Python piece of code to determine the required configuration to be built for distribution on the Nexus switches:

Figure 36 Master Application Questionnaire – 1 Part 1

```
When You Are Ready, Press Enter. Else, Close The Application - 0S0S

Do you Need To Create A VPC Domain? Answer [Yes] or [No]:
Yes
What Is The VPC Domain ID?
100
Is The VPC Domain Will Be Connected To Normal Spanning-tree Switches? Choose 1)Yes     2)No:
1
Is The VPC Domain Will Have SVI Configured? Choose 1)Yes     2)No:
1
Enter First Switch Management IP Address:
10.10.1.1
Enter Second Switch Management IP Address:
10.10.1.2
Enter Management Subnet Mask:
255.255.255.0
Enter Management Default Gateway:
10.10.1.254
```

The next part of the questionnaire is to ask for the interfaces that will make the VPC domain as well as the uplinks to the outside network connected to the Nexus switches as in the screenshot below:

Figure 37 Master Application Questionnaire - 2

```
Please Enter VPC Link Ports (Port Number Only) One Per Line - Enter X To Finish:
1/1
1/2
X
Please Enter Uplink Ports (Port Number Only) One Per Line- Enter X To Finish:
1/3
1/4
X
VPC Ports Are: ['1/1', '1/2']
Uplink Ports Are: ['1/3', '1/4']
Is The Information Correct? - Answer [Yes] Or [No]:
Yes
```

According to the information provided in the first screen when the application starts, this Python piece of code will be configuring LACP for all bundled uplinks.

LACP is part of the IEEE (***I***nstitute of ***E***lectrical and ***E***lectronics ***E***ngineers) specification 802.3ad that allows you to bundle several physical ports to form a single logical channel.

Section 9: Phase-6 Complementary Configuration

There are some additional configuration that should be applied on each Nexus switch to complete the device configuration according to the client information before it is ready to be installed in production which include the following:

- Configuring the username / password credentials.
- Configuring a hostnames for the Nexus switches.
- Configuring a default route for the management interface, so, the Nexus switch will be reachable through the network.

The below screenshot showing the answers to the questionnaire raised by the Python application regarding this information:

Figure 38 Master Application Questionnaire – 1 Part 2

```
Please Enter Username Required To Be Configured:
adminst
Please Enter Password Required To Be Configured (Complex Password Is A Must):
Password123!
Please Enter The Client Name:
lab
```

All these configuration will be stored in the Python application till it is pushed to the Nexus switches using “Paramiko” Python function after the POAP operation & VPC configuration is completed.

The screenshot below is showing the Python application logged into the Nexus switches to start the additional configuration on the switches:

Figure 39 Master Application Login To Nexus Switch

```
N9K-2# show users
NAME     LINE         TIME         IDLE          PID COMMENT
cisco    pts/0        Mar 26 09:14   .         32362 *
cisco    pts/3        Mar 26 10:47 00:34       16218 (192.168.101.253) session=s
sh
cisco    pts/7        Mar 26 10:48 00:32       16775 (192.168.101.253) session=s
sh
cisco    pts/10       Mar 26 10:49 00:31       17081 (10.10.1.6) session=ssh
```

Section 10: Phase-7 Finalizations & Confirmations

As part of finalizing the configuration of the VPC domain on the Nexus switches is to check & validate the VPC domain status on both Nexus switches.

The VPC domain should be up & working fine after the full configuration of the Nexus switches.

The challenge here is the Nexus switches will have a different management IP address according to the topology required, so, after the complete configuration of the VPC domain, the application will silently change your laptop IP address to any IP address in the same range of the management interfaces to be able to login to the Nexus switches again & get the last thing in this application which is the VPC domain status using the command “show vpc”.

In the screenshot below, the configuration confirmation messages that you will receive from the application:

Figure 40 Nexus Switch Configuration Notifications

```
***Now Configuring Nexus Switches - VPC Part, Please Wait...***

***VPC Configuration Finished...***

***Now Configuring Nexus Switches - Uplinks Part, Please Wait...***

***Uplinks Configuration Finished...***

***Starting Additional Configuration, Please Wait...***

Socket exception: An existing connection was forcibly closed by the remote host (10054)
***Additional Configuration Finished...***
```

After the switch is configured with the new fixed IP address, the application will confirm the reachability to the Nexus switches with the added IP addresses in the beginning of the application according to the screenshot below:

Figure 41 Switch Reachability Test After Applying Full Configuration

```
Now Pinging To Switches With The Required Management IP Address To Confirm Reachability, Please Wait...

Switch: 10.10.1.1 Up
Switch: 10.10.1.2 Up
Reachability Test Finished...
```

Then the application will login to both Nexus switches & display the VPC domain status according to the screenshots below:

Figure 42 Nexus Switch VPC Status – P art 1

```
***Displaying VPC Configuration - show vpc -, Please Wait...***

Show VPC On Switch 1

Legend:

               (*) - local vPC is down, forwarding via vPC peer-link


vPC domain id                     : 100

Peer status                       : peer adjacency formed ok

vPC keep-alive status             : peer is alive

Configuration consistency status  : success

Per-vlan consistency status       : success

Type-2 consistency status         : success

vPC role                          : secondary

Number of vPCs configured         : 1

Peer Gateway                      : Enabled

Dual-active excluded VLANs        : -

Graceful Consistency Check        : Enabled

Auto-recovery status              : Disabled

Delay-restore status              : Timer is off.(timeout = 30s)

Delay-restore SVI status          : Timer is off.(timeout = 10s)

Operational Layer3 Peer-router    : Disabled

Virtual-peerlink mode             : Disabled
```

Figure 43 Nexus Switch VPC Status – P art 2

```
vPC Peer-link status

---------------------------------------------------------------------

id    Port   Status Active vlans

--    ----   ------ --------------------------------------------------

1     Po100  up     1


vPC status

----------------------------------------------------------------------------

Id    Port          Status Consistency Reason                Active vlans

--    ------------  ------ ----------- ------                ---------------

200   Po200         down*  success     success               -



Please check "show vpc consistency-parameters vpc <vpc-num>" for the

consistency reason of down vpc and for type-2 consistency reasons for

any vpc.
```

This is the last part of the application & by this the application is finished its task in the lab use case, now it will close ☺ according to the screenshot below:

Figure 44 Master Application Last Message

```
***Configuration & Verification Are Completed...***



Please Refer To Cisco Guides For How To Confirm VPC Configuration & Troubleshooting
Bye :) Ahmed Ossama

C:\NexusSwitchAutoNewStyle>
```

Best Troubleshooting Ways

Many “show” commands had been used throughout that lab to display a successful implementation of different features regarding the VPC datacenter setup.

As a network engineer, there is two other methods you have to master to be an expert in troubleshooting as mentioned below:

- Debugging on the potential device that have an issue.
- Packet capture throughout the network part related to the raised issue.

Both functions can be implemented easily across all kind of platforms including Cisco devices.

The first part here is listing some of the useful troubleshooting ways for POAP execution, the list is for learning purposes only & not complete, please refer to Cisco documentation for a complete list of applicable POAP troubleshooting ways.

Some of the good POAP troubleshooting steps are as mentioned below:

- Console to the Nexus switch during POAP operation & monitor the POAP execution status.
- Download the latest log file from the Nexus switch & read the issues written in it.
- Run a packet capture during the POAP process.

Part of the troubleshooting is to collect the POAP execution logs from the Nexus switch to read all the logs generated during the POAP operation, this file is very important in troubleshooting of POAP operation failure.

The screenshot below showing part of the POAP logs file collected from one of the Nexus switches after successful bootstrapping:

Figure 45 POAP Logs File

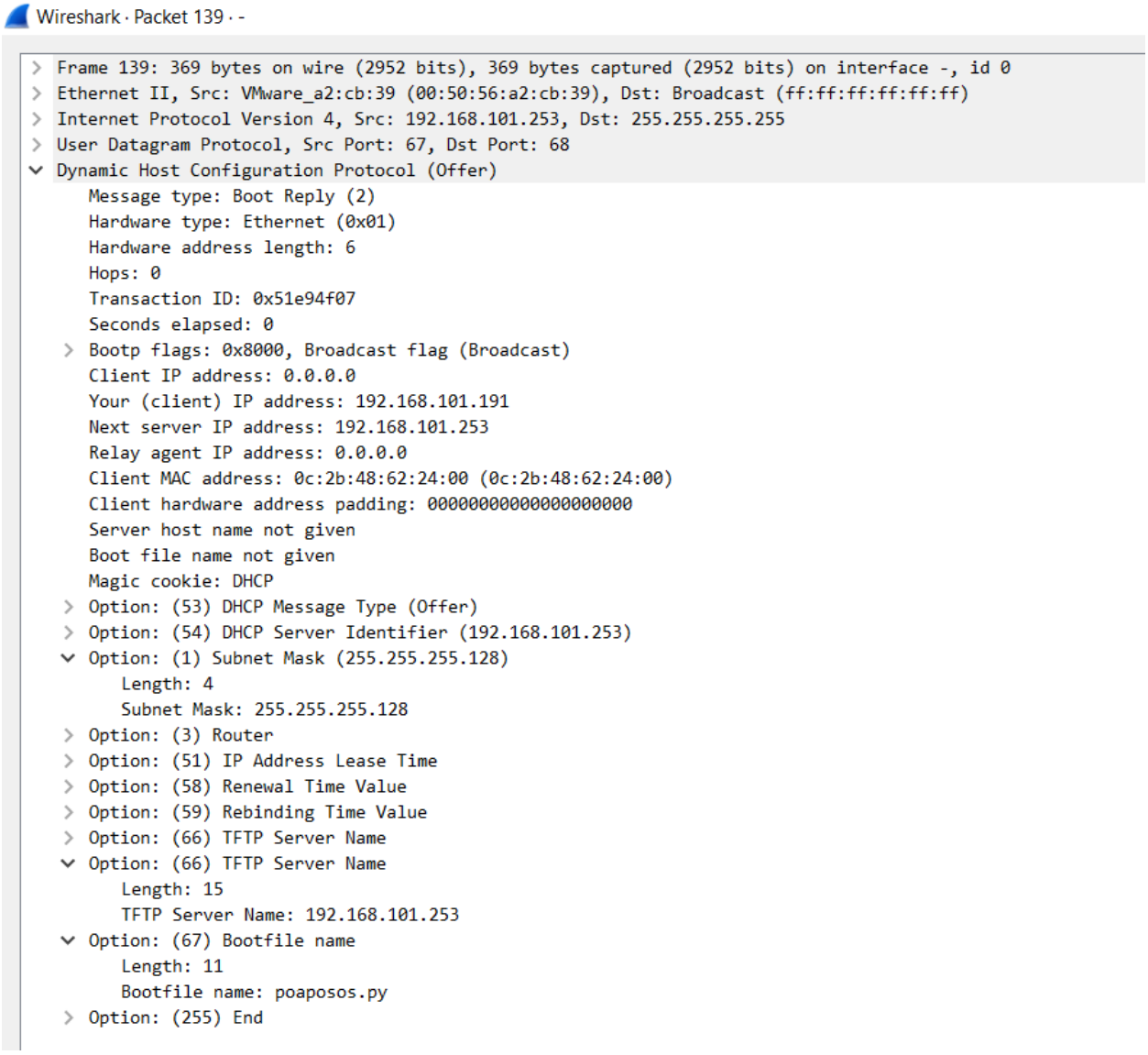
```
N9K-2# tail bootflash:20200323_095049_poap_2713_init.log 33
Mon Mar 23 09:51:18 2020:Received BOOT Reply
Mon Mar 23 09:51:18 2020:poap_handle_bootp_reply Entered
Mon Mar 23 09:51:18 2020:cookie: 2, 1, 6, 0
Mon Mar 23 09:51:18 2020:Options cookie ok
Mon Mar 23 09:51:18 2020:tlv type=53 length=1
Mon Mar 23 09:51:18 2020:poap_handle_bootp_reply: Received DHCPACK
Mon Mar 23 09:51:18 2020:poap_demux: event_id: 22
Mon Mar 23 09:51:18 2020:poap_parse_options_in_ack: TLV type 58 not required
Mon Mar 23 09:51:18 2020:poap_parse_options_in_ack: TLV type 59 not required
Mon Mar 23 09:51:18 2020:poap_handle_dhcp_ack:tg_start_timer failed:-1
Mon Mar 23 09:51:18 2020:poap_dhcp_intf_ac_action_receive_dhcp_ack: Failed to handle dhcp ack
Mon Mar 23 09:51:18 2020:poap_net_rx_pkt: received msg
Mon Mar 23 09:51:18 2020:poap_net_rx_pkt: Received ipv4 pkt
Mon Mar 23 09:51:18 2020:Received BOOT Reply
Mon Mar 23 09:51:18 2020:poap_handle_bootp_reply Entered
Mon Mar 23 09:51:18 2020:cookie: 2, 1, 6, 0
Mon Mar 23 09:51:18 2020:Options cookie ok
Mon Mar 23 09:51:18 2020:tlv type=53 length=1
Mon Mar 23 09:51:18 2020:poap_handle_bootp_reply: Received DHCPACK
Mon Mar 23 09:51:18 2020:poap_process_dhcp_message: xid=851952959 does not match sent xid=1609699785 on mgmt0(5000000) msg_type=5
Mon Mar 23 09:51:28 2020:poap_demux: event_id: 25
Mon Mar 23 09:51:29 2020:poap_demux: event_id: 29
Mon Mar 23 09:51:29 2020:poap_demux: event_id: 23
Mon Mar 23 09:51:29 2020:poap_global_ac_action_start_script_execution_phase: Controller found sent to PNP succesfully
Mon Mar 23 09:51:29 2020:poap_dhcpdiscover_abort: Destroy port fsm start
Mon Mar 23 09:51:29 2020:poap_dhcpdiscover_abort: Destroy port fsm end count=2
Mon Mar 23 09:51:29 2020:poap_phase_script_create_timers: creating POAP_PH_SCRIPT_EXEC_VERFY_TIMER
Mon Mar 23 09:51:29 2020:poap_phase_script_create_timers: creating POAP_PH_SCRIPT_EXEC_REPLY_TIMER
Mon Mar 23 10:58:11 2020:poap_demux: event_id: 20
Mon Mar 23 10:58:11 2020:poap_net_init: Initialize with netstack
Mon Mar 23 10:58:11 2020:poap_prepare_user_class_option_tlv: Added User Class Information TLV type 77, length 11, value is Cisco-POAP val_length: 10
Mon Mar 23 10:58:11 2020:poap_net_uninit: Uninitialize with netstack
Mon Mar 23 10:58:11 2020:poap_demux: event_id: 25
```

Packet captures can be read inline from where it is collected on the appliance or through a decoding application like Wireshark.

Wireshark is one of the best applications used for decoding packet captures as it keeps updating with newly added applications / protocols (highly recommended).

The screenshot below showing the packet capture for the Nexus switch once it is started its POAP operation to show how the workflow is going from the Nexus switch to the POAP application server which is starting by the DHCP request from the Nexus switch followed by a DHCP offer from the DHCP server contained inside the POAP application:

Figure 46 POAP Packet Capture



Complete Python Application

In this section am putting the full master application in your hand, so, you can use it to automate your VPC deployment on Nexus switches without a single touch to the CLI (***C***ommand ***L***ine ***I***nterface).

The script is divided into sections according to the coming screenshots.

This screenshot showing the libraries used throughout the master application:

Figure 47 Master Application – Part 1

```
from pythonping import ping
import paramiko
import time
import os
import socket
import shutil
import signal
import subprocess
import getpass
import hashlib
from add_first import add_first
from add_second import add_second
```

This screenshot showing the start page of the master application:

Figure 48 Master Application – Part 2

```
print('\n\n')
print('         0*******************************************************************************S')
print('         *Hello To Nexus Switches Initialization Application (NXOS) Using POAP          *')
print('         *This IS An Open For Use Application Under GNU License                         *')
print('         *Please Makse Sure Of The Provided Input, Using It Is Under Your Responsibility*')
print('         0*******************************************************************************S\n')
print('It Is Better To Read About POAP (Power On Auto Provisioning) To Know How This Application Works')
print('To Start, Please Read The Following Carefully:\n')
print('1) It IS A Must To Run This Application As Administrator.')
print('2) Max. Of 2 Switches In The VPC Domain Will Be Configured.')
print('3) Connect All Management Interfaces To One Switch Along With Your Laptop.')
print('4) All Switches Should Be On Factory Defaults & Donnot Press Any Key On The Console For POAP To Work.')
print('5) Please Note This Application Is Configuring LACP Only.')
print('6) This Application Is Supported On Windows OS Only & Tested On Windows 10.')
print('7) Please Configure Your Laptop Ethernet Interface With IP 192.168.101.253/25 & Disable All Firewalls.')
print('8) Please Put The Upgrade File (If You Need To Upgrade!) In The Same Application Folder & Have The MD5 Ready With You.\n')
time.sleep(11)
```

This screenshot showing the check for the laptop IP address as a pre-requisite by the master application:

Figure 49 Master Application – Part 3

```
if myip == '192.168.101.253':
    print('Your Laptop IP Address:', myip, 'Accepted\n')
else:
    print('Wrong Laptop IP Address, Please Change To 192.168.101.253\n')
    time.sleep(1)
    exit()
```

This screenshot showing the check for the upgrade file:

Figure 50 Master Application – Part 4

```
print('Now Checking For The Upgrade File...\n')
time.sleep(3)
g= 0
for file in os.listdir(os.getcwd()):
    if file.endswith(".bin"):
        print('Upgrade File: ', file, '\n')
        g= file
        if 'Yes' not in input('Is That The Correct File? Answer [Yes] Or [No]:\n'):
            exit()
if g==0:
    print('No Upgrade File Found\n')
print('\n')
if g != 0:
    k= hashlib.md5(open(g,'rb').read()).hexdigest()
    l= input('Please Enter The Upgrade File MD5 Acquired From Cisco Download Website:\n')
    if l == k:
        print ('Upgrade File Authenticy Confiremd, Continuing With the Application...\n')
        time.sleep(3)
    if l != k:
        print ('MD5 Validation Failed, Application Will Exit Now\n')
        time.sleep(3)
        exit()
```

This screenshot showing the first questionnaire in the master application:

Figure 51 Master Application – Part 5

```
inios1= os.getcwd() + '/server/initdev1.cfg'
inios2= os.getcwd() + '/server/initdev2.cfg'
shutil.copy(inios1, os.getcwd())
shutil.copy(inios2, os.getcwd())
if g !=0:
    add_first ('initdev1.cfg', 'boot nxos bootflash:/' + g)
    add_first ('initdev2.cfg', 'boot nxos bootflash:/' + g)
input('When You Are Ready, Press Enter. Else, Close The Application - 0S0S\n')
s= input('Do you Need To Create A VPC Domain? Answer [Yes] or [No]:\n')
if s != ('Yes' or 'No'):
    print ('Wrong Answer')
    exit()
if s == 'No':
    print('Only Switch Upgrade (If Included) Will Be Done Along With Configuring Management IP Address & Credentials\n')
a=0
b=0
c=0
d1= []
d2= []
e= []
f= []
if s == 'Yes':
    a= int(input('What Is The VPC Domain ID:\n'))
    b= int(input('Is The VPC Domain Will Be Connected To Normal Spanning-tree Switches: Choose 1)Yes , 2)No\n'))
    c= int(input('Is The VPC Domain Will Have SVI Configured: Choose 1)Yes , 2)No\n'))
d1=input('Enter First Switch Management IP Address:\n')
d2=input('Enter Second Switch Management IP Address:\n')
if d1 == d2:
    print ('Same Management IP Address Cannot Be Configured On Both Nexus Switches\n')
    exit()
e= input('Enter Management Subnet Mask:\n')
f= input('Enter Management Default Gateway:\n')
username= input('Please Enter Username Required To Be Configured:\n')
password= input('Please Enter Password Required To Be Configured (Complex Password Is A Must):\n')
client= input('Please Enter The Client Name:\n')
```

This screenshot showing the POAP process startup:

Figure 52 Master Application – Part 6

```
tfos= os.getcwd() + '/server/tftpd32.ini'
shutil.copy(tfos, os.getcwd())
pos= os.getcwd() + '/server/poaposos.py'
shutil.copy(pos, os.getcwd())
time.sleep(1)
if g != 0:
    m = "cli (" + "'copy tftp://cisco:cisco@192.168.101.253/" + g + " bootflash: vrf management" + "')"
    n = "cli (" + "'configure terminal ;  boot nxos bootflash:" + g + "')"
    add_first('poaposos.py', n)
    add_first ('poaposos.py', m)
add_first ('poaposos.py', 'from cli import *')
add_first ('poaposos.py', '#!/bin/env python')
time.sleep(1)
h= hashlib.md5(open('poaposos.py','rb').read()).hexdigest()
print('Current POAP File MD5SUM Is: ', h, 'Please Keep It For Your Reference.')
time.sleep(1)
add_first ('poaposos.py', '#!/bin/env python')
add_second ('poaposos.py', '#md5sum=' + '''"''' + h + '''"''')
print ('Now Bootstrapping Switches, Please Wait...\n')
tfpid = subprocess.Popen("tftpd64.exe")
time.sleep(1911)
os.kill(tfpid.pid, signal.SIGTERM)
os.remove(os.getcwd() + '/initdev1.cfg')
os.remove(os.getcwd() + '/initdev2.cfg')
os.remove(os.getcwd() + '/poaposos.py')
os.remove(os.getcwd() + '/tftpd32.ini')
```

This screenshot showing the check for the switches reachability after successfull operation of the POAP script execution:

Figure 53 Master Application – Part 7

```
allip= ['192.168.101.181','192.168.101.182']
time.sleep(3)
swlst= []
i= 0
print('Now Pinging To Switches To Confirm Reachability, Please Wait...\n')
time.sleep(3)
for i in range(len(allip)):
    response= ping(allip[i], count=7)
    response1= response.success()
    if response1 == True:
        print('Switch:', allip[i], 'Up')
        swlst.append(allip[i])
    elif response1 == False:
        print ('Switches Are Unreachable')
        exit()
print('Reachability Test Finished...')
print('\n')
```

This screenshot showing the check of the switch current software version as part of confirming a successful upgrade to the Nexus switches:

Figure 54 Master Application – Part 8

```
client= paramiko.SSHClient()
client.load_system_host_keys()
client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
i=0
print('Getting Software Version From Switches, Please Wait...\n')
time.sleep(3)
for i in range(len(swlst)):
    client.connect(swlst[i], port= 22, username= 'cisco', password= 'Cisco123')
    shell= client.invoke_shell()
    shell.send('show version\n')
    time.sleep(2)
    output = shell.recv(65535).strip()
    output2 = str(output).split()
    shell.send('exit\n')
    output3 = output2[output2.index('NXOS:') + 2]
    output4 = output3.replace('\\r\\n', '')
    print('Switch', i+1, 'Current Software Version: ', output4, '\n')
```

This screenshot showing the second questionnaire in the master application:

Figure 55 Master Application – Part 9

```
vpclnk= []
uplnk= []
vpclnk1= input('Please Enter VPC Link Ports (Port Number Only) One Per Line - Enter X To Finish:\n')
while vpclnk1 != "X":
    vpclnk.append(vpclnk1)
    vpclnk1= input()
uplnk1= input('Please Enter Uplink Ports (Port Number Only) One Per Line- Enter X To Finish:\n')
while uplnk1 != "X":
    uplnk.append(uplnk1)
    uplnk1= input()
if len(vpclnk) == 0:
    print('No VPC Links Added - Application Will Close')
    time.sleep(1)
    exit()
else:
    print("VPC Ports Are:", vpclnk)
if len(uplnk) == 0:
    print('No Uplinks Added - No Uplinks Will Be Configured')
    time.sleep(1)
else:
    print("Uplink Ports Are:", uplnk)
time.sleep(3)
if 'Yes' not in input("Is The Information Correct? - Answer [Yes] Or [No]:\n"):
    print("Please Restart The Program.\n")
    time.sleep(1)
    exit()
print("Now Configuring Nexus Switches - VPC Part, Please Wait...\n")
time.sleep(1)
```

This screenshot showing the VPC configuration as pushed by the master application:

Figure 56 Master Application – Part 10

```
print('***Now Configuring Nexus Switches - VPC Part, Please Wait...***\n')
i= 0
for i in range(2):
    client.connect(swlst[i], port= 22, username= 'cisco', password= 'Cisco123')
    shell = client.invoke_shell()
    shell.send('configure terminal\n')
    shell.send('feature vpc\n')
    time.sleep(1)
    shell.send('feature lacp\n')
    time.sleep(1)
    shell.send('vpc domain ' + str(a) + '\n')
    time.sleep(1)
    if i == 0:
        shell.send('peer-keepalive destination ' +  d2 + ' source ' + d1 + ' vrf management \n')
        time.sleep(1)
        shell.send('role priority 100\n')
        time.sleep(1)
    if i == 1:
        shell.send('peer-keepalive destination ' + d1 + ' source ' + d2 + ' vrf management \n')
        time.sleep(1)
        shell.send('role priority 90\n')
        time.sleep(1)
    shell.send('\n')
    if b == 1:
        shell.send('peer-switch\n')
        time.sleep(1)
    if c == 1:
        shell.send('peer-gateway\n')
        time.sleep(1)
    shell.send('interface port-channel 100\n')
    time.sleep(1)
    shell.send('vpc peer-link\n')
    time.sleep(1)
    shell.send('switchport mode trunk\n')
    time.sleep(1)
    shell.send('no shutdown\n')
    time.sleep(1)
    shell.send('interface ethernet ' + vpclnk[0] + ', ethernet' + vpclnk[1] + '\n')
    time.sleep(1)
    shell.send('switchport mode trunk\n')
    time.sleep(1)
    shell.send('channel-group 100 mode active\n')
    time.sleep(1)
    shell.send('no shutdown\n')
    time.sleep(1)
print('***VPC Configuration Finished...***\n')
```

This screenshot showing the uplinks configuration as pushed by the master application:

Figure 57 Master Application – Part 11

```
print('***Now Configuring Nexus Switches - Uplinks Part, Please Wait...***\n')
for i in range(len(swlst)):
    for j in range (len(uplnk)):
        client.connect(swlst[i], port= 22, username= 'cisco', password= 'Cisco123')
        shell = client.invoke_shell()
        shell.send('configure terminal\n')
        time.sleep(1)
        shell.send('interface ethernet ' + uplnk[j] + '\n')
        time.sleep(1)
        shell.send('switchport mode trunk\n')
        time.sleep(1)
        shell.send('channel-group 200 mode active\n')
        time.sleep(1)
        shell.send('no shutdown\n')
        time.sleep(1)
        shell.send('interface port-channel 200\n')
        time.sleep(1)
        shell.send('switchport mode trunk\n')
        time.sleep(1)
        shell.send('vpc 200\n')
        time.sleep(3)
        shell.send('no shutdown\n')
        time.sleep(1)
print('***Uplinks Configuration Finished...***\n')
```

This screenshot showing the adaptation of the laptop interface to be able to make the master application to connect to the Nexus switch again to collect show “show” information:

Figure 58 Master Application – Part 12

```
getos= os.getcwd() + '/server/getname.cmd'
shutil.copy(getos, os.getcwd())
time.sleep(3)
q = subprocess.Popen('getname 192.168.101.253', shell=True, stdout=subprocess.PIPE)
r1, error=  q.communicate()
r2= r1.decode('utf-8')
r3= r2.replace('\r\n','')
d3= str(d1).split('.')
d31= d3[0]
d32= d3[1]
d33= d3[2]
d34= int(d3[3])
d4= d34 + 5
d5= str(d31) + '.' + str(d32) + '.' + str(d33) + '.' + str(d4)
t1 = str('netsh interface ipv4 add address ' + '"' + r3 + '" ' + d5 +  ' 255.255.255.0')
t2 = str('netsh interface ipv4 delete address ' + '"' + r3 + '" ' + d5 +  ' 255.255.255.0')
subprocess.Popen( t1, shell=True, stdout=subprocess.PIPE)
time.sleep(3)
```

This screenshot showing the miscellaneous configuration as pushed by the master application:

Figure 59 Master Application – Part 13

```
print('***Starting Additional Configuration, Please Wait...***\n')
i= 0
for i in range(2):
    client.connect(swlst[i], port= 22, username= 'cisco', password= 'Cisco123')
    shell = client.invoke_shell()
    shell.send('configure terminal\n')
    time.sleep(1)
    shell.send('username ' + username + 'password ' + password + ' role network-admin\n')
    time.sleep(1)
    shell.send('ip route 0.0.0.0 0.0.0.0 ' + f + ' vrf management\n')
    time.sleep(1)
    shell.send('show running-config interface mgmt 0\n')
    time.sleep(1)
    output = shell.recv(65535).strip()
    output2 = str(output).split()
    output3 = output2[output2.index('address') + 1]
    output4 = output3.replace("\\r\\n\\r\\n\\rosos(config)#'",'')
    time.sleep(11)
    if output4 == '192.168.101.181/25':
        shell.send('hostname N9K-1\n')
        time.sleep(1)
        shell.send('interface mgmt 0\n')
        time.sleep(1)
        shell.send('ip address ' + d1 + ' 255.255.255.0\n')
        time.sleep(1)
        shell.send('copy running-config startup-config\n')
        time.sleep(1)
    if output4 == '192.168.101.182/25':
         shell.send('hostname N9K-2\n')
         shell.send('interface mgmt 0\n')
         time.sleep(1)
         shell.send('ip address ' + d2 + ' 255.255.255.0\n')
         time.sleep(1)
         shell.send('copy running-config startup-config\n')
         time.sleep(1)
    time.sleep(1)
print('***Additional Configuration Finished...***\n')
```

This screenshot showing the collected “show” information from the Nexus switches:

Figure 60 Master Application – Part 14

```
print('***Displaying VPC Configuration - show vpc -, Please Wait...***\n')
i= 0
for i in range(2):
    print ('Show VPC On Switch', i + 1, '\n')
    client.connect(dd[i], port= 22, username= 'cisco', password= 'Cisco123')
    shell = client.invoke_shell()
    stdin, stdout, stderr = client.exec_command('show vpc\n')
    time.sleep(1)
    u= 0
    for u in stdout:
        print(u)
    print ('\n')
print('***Configuration & Verification Are Completed...***\n')
```

This screenshot showing the finalization of the master application:

Figure 61 Master Application – Part 15

```
print('*****Please Refer To Cisco Guides For How To Confirm VPC Configuration & Troubleshooting*****')
print('Bye :) Ahmed Ossama')
time.sleep(5)
exit()
```

To see the full output of the master application in one page, please refer to the screenshot below:

Figure 62 Master Application Run

```
C:\NexusSwitchAutoNewStyle>switchosos.py



      0**************************************************************************S
      *Hello To Nexus Switches Initialization Application (NXOS) Using POAP        *
      *This IS An Open For Use Application Under GNU License                       *
      *Please Make Sure Of The Provided Input, Using It Is Under Your Responsibility*
      0**************************************************************************S


It Is Better To Read About POAP (Power On Auto Provisioning) To Know How This Application Works
To Start, Please Read The Following Carefully:


1) It IS A Must To Run This Application As Administrator.
2) Max. Of 2 Switches In The VPC Domain Will Be Configured.
3) Connect All Management Interfaces To One Switch Along With Your Laptop.
4) All Switches Should Be On Factory Defaults & Don not Press Any Key On The Console For POAP To
Work.
5) Please Note This Application Is Configuring LACP Only.
6) This Application Is Supported On Windows OS Only & Tested On Windows 10.
7) Please Configure Your Laptop Ethernet Interface With IP 192.168.101.253/25 & Disable All Firewalls.
8) Please Put The Upgrade File (If You Need To Upgrade!) In The Same Application Folder & Have The
MD5 Ready With You.
```

Now Checking For The Laptop Static IP Address...

Your Laptop IP Address: 192.168.101.253 Accepted

Now Checking For The Upgrade File...

Upgrade File: nxos.9.3.3.bin

Is That The Correct File? Answer [Yes] Or [No]:

Yes

Please Enter The Upgrade File MD5 Acquired From Cisco Download Website:

171cf0be4c678d873b1729eae1b043b7

Upgrade File Authenticy Confirmed, Continuing With the Application...

When You Are Ready, Press Enter. Else, Close The Application - 0S0S

Do you Need To Create A VPC Domain? Answer [Yes] or [No]:

Yes

What Is The VPC Domain ID?

100

Is The VPC Domain Will Be Connected To Normal Spanning-tree Switches? Choose 1)Yes 2)No:

1

Is The VPC Domain Will Have SVI Configured? Choose 1)Yes 2)No:

1

Enter First Switch Management IP Address:

10.10.1.1

```
Enter Second Switch Management IP Address:
10.10.1.2
Enter Management Subnet Mask:
255.255.255.255.0
Enter Management Default Gateway:
10.10.1.254
Please Enter Username Required To Be Configured:
administrator
Please Enter Password Required To Be Configured (Complex Password Is A Must):
Password123!
Please Enter The Client Name:
lab
Current POAP File MD5SUM Is: f3839851e1f1ccce856ae825b266daf9 Please Keep It For Your Reference.
Now Bootstrapping Switches, Please Wait...


Now Pinging To Switches To Confirm Reachability, Please Wait...


Switch: 192.168.101.181 Up
Switch: 192.168.101.182 Up
Reachability Test Finished...



Getting Software Version From Switches, Please Wait...


Switch 1 Current Software Version:  9.3(3)
Switch 2 Current Software Version:  9.3(3)
```

```
Please Enter VPC Link Ports (Port Number Only) One Per Line - Enter X To Finish:
1/1
1/2
X
Please Enter Uplink Ports (Port Number Only) One Per Line- Enter X To Finish:
1/3
1/4
X
VPC Ports Are: ['1/1', '1/2']
Uplink Ports Are: ['1/3', '1/4']
Is The Information Correct? - Answer [Yes] Or [No]:
Yes



***Now Configuring Nexus Switches - VPC Part, Please Wait...***


***VPC Configuration Finished...***


***Now Configuring Nexus Switches - Uplinks Part, Please Wait...***


***Uplinks Configuration Finished...***


***Starting Additional Configuration, Please Wait...***


Socket exception: An existing connection was forcibly closed by the remote host (10054)
***Additional Configuration Finished...***
```

```
Now Pinging To Switches With The Required Management IP Address To Confirm Reachability, Please
Wait...

Switch: 10.10.1.1 Up

Switch: 10.10.1.2 Up

Reachability Test Finished...



***Displaying VPC Configuration - show vpc -, Please Wait...***

Show VPC On Switch 1

Legend:

            (*) - local vPC is down, forwarding via vPC peer-link



vPC domain id                  : 100

Peer status                    : peer adjacency formed ok

vPC keep-alive status          : peer is alive

Configuration consistency status  : success

Per-vlan consistency status      : success
```

```
Type-2 consistency status       : success

vPC role                        : secondary

Number of vPCs configured       : 1

Peer Gateway                    : Enabled

Dual-active excluded VLANs      : -

Graceful Consistency Check      : Enabled

Auto-recovery status            : Disabled

Delay-restore status            : Timer is on.(timeout = 30s, 24s left)

Delay-restore SVI status        : Timer is off.(timeout = 10s)

Operational Layer3 Peer-router  : Disabled

Virtual-peerlink mode           : Disabled



vPC Peer-link status

---------------------------------------------------------------------
```

```
id    Port   Status Active vlans

--    ----   ------ --------------------------------------------------

1     Po100  up     1



vPC status

----------------------------------------------------------------------------

Id    Port        Status Consistency Reason              Active vlans

--    ------------  ------ ----------- ------            ---------------

200   Po200       down*  Not       Consistency Check Not -

                    Applicable   Performed



Please check "show vpc consistency-parameters vpc <vpc-num>" for the

consistency reason of down vpc and for type-2 consistency reasons for

any vpc.
```

```
Show VPC On Switch 2

Legend:

        (*) - local vPC is down, forwarding via vPC peer-link



vPC domain id                  : 100

Peer status                    : peer adjacency formed ok

vPC keep-alive status          : peer is alive

Configuration consistency status  : success

Per-vlan consistency status      : success

Type-2 consistency status        : success

vPC role                       : primary

Number of vPCs configured        : 1
```

```
Peer Gateway                  : Enabled

Dual-active excluded VLANs    : -

Graceful Consistency Check    : Enabled

Auto-recovery status          : Disabled

Delay-restore status          : Timer is on.(timeout = 30s, 22s left)

Delay-restore SVI status      : Timer is off.(timeout = 10s)

Operational Layer3 Peer-router  : Disabled

Virtual-peerlink mode         : Disabled



vPC Peer-link status

---------------------------------------------------------------------

id   Port   Status Active vlans

--   ----   ------ --------------------------------------------------

1    Po100  up     1
```

```
vPC status

----------------------------------------------------------------------------

Id    Port          Status Consistency Reason              Active vlans

--    ------------  ------ ----------- ------              ---------------

200   Po200         down*  Not         Consistency Check Not -

                           Applicable  Performed



Please check "show vpc consistency-parameters vpc <vpc-num>" for the

consistency reason of down vpc and for type-2 consistency reasons for

any vpc.



***Configuration & Verification Are Completed...***
```

*****Please Refer To Cisco Guides For How To Confirm VPC Configuration & Troubleshooting*****

Bye :) Ahmed Ossama

C:\NexusSwitchAutoNewStyle>

Feel free to request all configuration files directly from me through e-mail.

This script could be CPU intensive & this is why you need to be fully aware of the capabilities of the machine running this master application.

References

The following list is an advanced reference for all the study material related to this lab:

- Cisco press books:
  - www.ciscopress.com.
- Cisco learning network:
  - learningnetwork.cisco.com.
- Cisco community:
  - community.cisco.com.
- IETF:
  - www.ietf.org.
- Python official page:
  - python.org.
- Cisco NXOS guides:
  - https://www.cisco.com/c/en/us/support/ios-nx-os-software/nx-os-software/products-installation-and-configuration-guides-list.html.

Please consider reading the theory first before dealing with the implementation either here through the lab or in your real life.

Thank You Message

I would like to thank you for taking the time to read & understand this lab.

It took from me days & weeks to completely prepare that Python automation lab and I really appreciate that you clearly understand programmability as a new next generation networking technique for implementation inside the campus LAN, WAN & datacenter networks.

As I have mentioned at the begging of this lab, this document is not meant to provide a complete understanding of the network automation behavior but it does provide a working configuration for a setup that had been created to show the power of Python nowadays.

To be an expert engineer, you need either to attend a training for Python coding or reading a different books, white papers or blogs about the programmability implementation using Python & YANG in different networks.

Augmenting this information with SDN techniques would be great & remember all the time "**Hard Work Pays OFF**".

A special thanks to my family who spared me the time to write this small lab, I love you all.

See you in my next lab.

Ahmed Ossama

CCIE#26611

ahmed-ossama@hotmail.com

Python Programmability