Overview

Shell Script

Python Script

CSV File

FPing Text File

Output Image

Acknowledgement **Overview**

Written by: Uyen Nguyen

Date: 17th August 2018

Language(s): Shell (Bash), Python 2.7

Files Used:​[S](https://drive.google.com/a/arista.com/open?id=1gCVOelzKPvZUt-4SPIgl3waWMbpD-42vYv8dQqHFlyw) [E Lab Vantage](https://drive.google.com/a/arista.com/open?id=1gCVOelzKPvZUt-4SPIgl3waWMbpD-42vYv8dQqHFlyw)

Operating System: Ubuntu

Mission: Access and update the hundred plus switches in the lab.

Result:

I came up with a script that lets you input the range of the IP addresses and return a list of switches that contain the active IP address that is in the range. Initially I wanted to just use fping but I decided to try out connecting the IP address with its own switch so that it would be easier to locate which switch is "active". This is done by using an Excel sheet containing information about the IP addresses and Hostnames and converting it into CSV (comma delimiter) so that it can be read.

The script for now only deals with general cases and if certain criteria are met but it can possibly be altered in the later future. I couldn't do the configuration part of the project, and Sam says it wouldn't be possible since the switches may be in use and cannot be configured. For now, this is what I have so far.

Most of this project was completed with assistance from Sam, Robert, and Pratamesh, as well as the power of Google/StackOverflow. The script is Bash but has a Python script running inside. I chose to do both languages because I wanted to use Bash for Fping, and I was more confident in using Python to sort and read the CSV file than have it all done in Bash. The Bash script is in charge of fping the range of IP addresses given by user input. The addresses are then written into a text file that can be accessed/altered if needed and used in the Python script. The text file that contains the IP addresses is overridden every time the script runs, so any IP address/switch range can be read.

The Python script is in charge of organizing the csv file and connecting the active IP addresses with their switches. It does this by asking the user for the name of the csv file, the name of the columns that store the IP address and Hostname respectively, and then creating a dictionary out of the file where the keys are the column names and their values are the items in the columns put into a list. With the assumption that the list containing the csv's IP addresses is the same size as the list containing the csv's Hostname (each IP address is associated with a switch, whether named or not), the program then compares the IP addresses received from fping with the IP addresses in the csv file and would take the index of the list containing the csv's IP and use to append the associated switch into another list. This other list will contain all the active switches found. PrettyTable is used to make it more readable.

I chose CSV mainly because I tried to do other libraries like pandas and I tried Google API but they didn't seem to work for their own reasons. I stuck with CSV though, since I like the dictionary format. Because the original excel file has several empty cells, CSV reads them as ‘’, and the part of the code where the information is put into a dictionary omits those single quotes from the values. However, the script ends up printing a blank line after the last switch; it’s not much of a problem but I still don’t know how it happened. CSV doesn’t work if the cells are empty or have merged, so I had to alter the Excel sheet slightly by filling empty cells with other strings and turning off the merge.

PrettyTable was done because PrettyPrint for some reason was not cooperating; I honestly think PrettyTable looks nicer than PrettyPrint for this project.

To test the script, I used the SE Lab Vantage Google Sheet. I didn’t change anything- I downloaded a copy and altered that one to see if the script work.

There are some things that will need to be improved on if possible. The script requires accurate user input and an Excel file properly converted into a csv file, and it has only worked with general cases. Sam had mentioned that the script requires the user to know the IP addresses and it would have been easier if I wrote the code so that it asks for the switch number instead.Overall, there is a lot of preparing and memorizing to do before running the script; hopefully in the later future anyone or I can implement improvements and changes that are better suited to make the script more efficient.

Below are the scripts and their codes as well as other files that are used. More info can be found on my​[G](https://github.com/uyen-carolyn/Arista-Intern-Project) [itHub](https://github.com/uyen-carolyn/Arista-Intern-Project)​account (this will be updated in a later time)

# Shell Script

#!/bin/sh

echo "Hello! Let's get started!"

# To make sure fping works, the user must know the range of the IP addresses they want to ping read -p "Enter the starting IP you want to ping: " start read -p "Enter the ending IP you want to ping: " end

fping -a -r 0 -g $start $end > test\_fping.txt

# the code for switch.py is available in the repository python switch.py

# Python Script

#!/usr/bin/python import csv from prettytable import PrettyTable

from collections import defaultdict

columns = defaultdict(list) # value in each column is appended to a list

# test\_fping.txt contains a list of IP addresses that are considered active from the fping command

# done before switch.py runs read\_this = open("test\_fping.txt", "r") file\_ip = read\_this.read().split('\n')

read\_this.close()

# inputs are used so that any csv file containing the needed information can be used so long

# as the name of the columns are known by the user file\_csv = raw\_input

("Enter the file name (must be a spreadsheet converted into a csv file): ") first\_c = raw\_input

("Enter the name of the column that the IP Addresses are stored in the sheet: ") second\_c = raw\_input

("Enter the name of the column that the switches are stored in the sheet: ")

# returns csv file as a dictionary where column names are keys and items in the column are values put into a list with open(file\_csv + '.csv') as f: reader = csv.DictReader(f) # read rows in dictionary format for row in reader:

for (k,v) in row.items(): # goes over each column name and value if not '' in columns[k]:

columns[k].append(v)

sheet\_ip = columns[first\_c]

sheet\_switch = columns[second\_c]

active\_switch = []

# assuming that all IPs in the file have one switch, whether it is publicly assigned, 'unused', or reserved

for s in sheet\_ip: if s in file\_ip:

active\_switch.append(sheet\_switch[sheet\_ip.index(s)])

# used to print results in a more readable format x = PrettyTable(["Active Switches"]) for switch in active\_switch: x.add\_row([switch]) print(x)

# CSV File

Rack 264 DUT IP,Rack 264 DUT Hostname,Model,Customer,User,,New Device

Name,,,,,,,,,,,,,,,,,,,,

10.92.62.65 = GW,unused,empty,,,,,,,,,,,,,,,,,,,,,,,,

10.92.62.66 - 97,unused,empty,,,,,,,,,,,,,,,,,,,,,,,,

10.92.62.97,dm1-264sw46,7050CX3-32S,,,,,,,A0,,,,,,,,,,,,,,,,,

10.92.62.98,dm1-264sw45,7160-32CQ,Act,Harsh Mehta,,fu623,,3,,,,,,,,,,,,,,,,,,

10.92.62.99,dm1-264sw44,7160-32CQ,Act,Harsh Mehta,,fu624,,4,,,,,,,,,,,,,,,,,,

10.92.62.100,dm1-264sw43,7010T,Edgar,Ascenty,,mn617,,5,,,,,,,,,,,,,,,,,,

10.92.62.101,dm1-264sw42,7280SE-68,Edgar,Ascenty,,gv400,,6,,,,,,,,,,,,,,,,,,

10.92.62.102,dm1-264sw41,7150S-24,Edgar,Ascenty,,ro618,,9,,,,,,,,,,,,,,,,,,

10.92.62.103,dm1-264sw40,7050QX-32S,Edgar,Ascenty,,ck490,,10,,,,,,,,,,,,,,,,,,

10.92.62.104,dm1-264sw39,7050QX-32,Edgar,Ascenty,,cd753,,11,,,,,,,,,,,,,,,,,,

10.92.62.105,dm1-264sw38,7050SX-64,Edgar,Ascenty,,do616,,12,,,,,,,,,,,,,,,,,,

10.92.62.106,dm1-264sw37,7050QX-32S,Edgar,Ascenty,,ck491,,13,,,,,,,,,,,,,,,,,,

10.92.62.107,dm1-264sw36,7050QX-32S,Edgar,Ascenty,,ck492,,14,,,,,,,,,,,,,,,,,,

10.92.62.108,dm1-264sw35,7280SR,Pranav,MSS,,hs492,,17,,,,,,,,,,,,,,,,,,

10.92.62.109,dm1-264sw34,7280SR,Pranav,MSS,,hs493,,18,,,,,,,,,,,,,,,,,, none,dm1-264sw33,7280SE-72,,,,fm427,,19,,,,,,,,,,,,,,,,,, 10.92.62.111,dm1-264sw32,7050SX-64,,,,do617,,20,,,,,,,,,,,,,,,,,,

10.92.62.112,dm1-264sw31,7150S-64,,,,co651,,21,,,,,,,,,,,,,,,,,,

10.92.62.113,dm1-264sw30,7280SR2A-48YC6,Act,Harsh Mehta,,gts463,console on 22,22,A1 chip,,,,,,,,,,,,,,,,,

10.92.62.114,dm1-264sw29,7020TRA-48,,,,wl605,,25,,,,,,,,,,,,,,,,,,

10.92.62.115,dm1-264sw28,7050SX2-72Q,Qualcomm ,Steve K,,psp303,,26,,,,,,,,,,,,,,,,,,

10.92.62.116,dm1-264sw27,7050SX2-72Q,Qualcomm ,Steve K,,psp304,,27,,,,,,,,,,,,,,,,,,

10.92.62.117,dm1-264sw26,7280TR,Edgar,Ascenty,,,,33,,,,,,,,,,,,,,,,,,

10.92.62.118,dm1-264sw25,7020TRA-48,,,,,,34,,,,,,,,,,,,,,,,,,

10.92.62.119,dm1-264sw24,7050SX2-128,,,,snp402,,28,,,,,,,,,,,,,,,,,,

10.92.62.120,unused,"Ixia

Chassis",,,,,,,,,,,,,,,,,,,,,,,,

10.92.62.121,unused,Spirent,,telnet admin/spt\_admin,,,,"39,40",,,,,,,,,,,,,,,,,,

10.92.62.123,unused,,,,,,,,,,,,,,,,,,,,,,,,,

10.92.62.124,Reserved,,,,,,,,,,,,,,,,,,,,,,,,, 10.92.62.125,Reserved,,,,,,,,,,,,,,,,,,,,,,,,,

10.92.62.126,Reserved,,,,,,,,,,,,,,,,,,,,,,,,,

,,,,,,,,,,,,,,,,,,,,,,,,,,

,,,,,,,,,,,,,,,,,,,,,,,,,,

,,,,,,,,,,,,,,,,,,,,,,,,,,

(The actual CSV account itself has way more commas after this, so I just cut it for this document)

FPing Text File

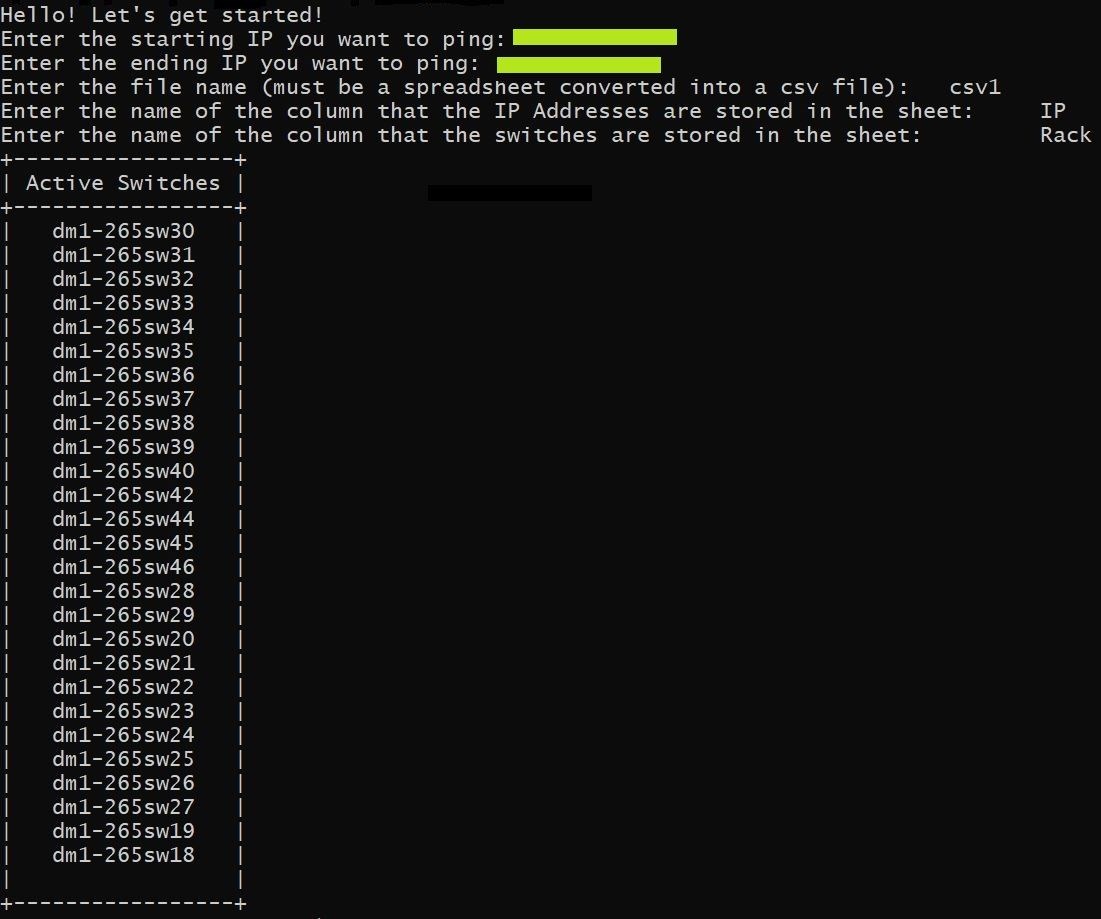
10.92.62.97 10.92.62.98

10.92.62.99

10.92.62.100 10.92.62.101 10.92.62.102 10.92.62.103 10.92.62.104 10.92.62.105 10.92.62.106 10.92.62.107 10.92.62.108 10.92.62.109 10.92.62.110 10.92.62.111 10.92.62.112 10.92.62.113 10.92.62.114 10.92.62.115 10.92.62.116 10.92.62.117 10.92.62.118 10.92.62.119 10.92.62.120 10.92.62.125

10.92.62.126

# Output Image



# Acknowledgement

Thank you Mike Francini for creating this mission and thank you Sam Muthirakalayil, Robert Cartell, and Prathamesh Timse for taking time out of your day to help me. And definitely thank you internet for helping me figure out the gist of the mission.