**JLoad Development Process**

This document assumes that the reader has a familiarity with Python 2.7 and PyEZ Library. Don’t be too concerned if you aren’t familiar with all the nuances of PyEZ library. However, we will be using a number of PyEZ functions so you must have the library loaded for this script to work. There are a number of different ways to accomplish the goal in this document, so keep that in mind while going over this example. You may see something that can be more efficient.

What was the need or problem you were attempting to solve

JLoad is a python script used for creating, loading, and applying configuration changes to multiple Juniper devices. An example would be adding a new firewall term to a firewall filter on all Juniper devices. Very useful if something needs to be deployed quickly to react to a time sensitive threat or if you have a huge deployment of devices and no management system. This program is based largely on a template script created for the same purpose. That being said, I have modified and enhanced it to be more functional in a production environment.

Before digging into this specific problem, there are some basic rules that may be helpful if you’re new to coding.

* Create functions out of sequences of code that are done over and over again. Function reuse can save you many lines of code.
* Create names for variables that describe the function of the variable or what it contains
* There’s usually more than one way to do everything, use the most simple solution that solves the problem
* It’s better to leave too many comments as opposed to too few
* If at all possible, get input by providing choices in the form of a menu
* Make the program as easy to use as possible

How you approached the problem with PyEZ (idea -> plain English process of the script -> pseudo-code -> final code)

The first thing that should be considered is the various tasks that must be done and how to do them with Python and the PyEZ Library. I usually start by considering the different things this script will have to do and how they can be solved…

* Task: Create a unique configuration for each device (need some standardized way to create these configurations)
  + There is a function in the PyEZ library for loading configurations. You provide a path to the template file and a dictionary of variables to be used in the template.
    - Syntax: cu.load(template\_path=”config-template.conf”, template\_vars=tvars)
  + For the template file we will be using a text file with .CONF extension. Since this function can accept “hierarchical” or “set” format, we will automatically detect the format and use the correct parameters. The parameter to accept “set” format is:
    - format=”set”
  + For the template variables, we will be using a .CSV file. This format is human readable and modifiable and can easily be loaded/converted into the dictionary format required by the load function.
    - We will make the headings of the .CSV file the same as the variable names in the template file, so we can map the values correctly
  + To access the correct template and variable files, we will need to ask the user which files we should use. The following two lines accomplish this for both files:
    - Syntax: fileList = getFileList(csv\_path)
      * This command uses a function to retrieve all the file names from the defined path.
    - Syntax: csv\_file = getOptionAnswer(“Choose a CSV file”, fileList)
      * This command captures the user selected file name using a function that asks the user to “choose a file” from the previously collected fileList variable that is displayed.
* Task: Access a Juniper device (do authentication and need a login/password)
  + To log into the Juniper router we have another function from PyEZ. The following command creates a “Device” object. We can then do actions such as “dev.open()” to open a connection with the system and run commands, such as “load”.
    - Syntax: Device(my\_hostname, user=my\_user, password=my\_pass)
  + The command requires a system hostname/ip, a username, and a password. We will be getting the system hostname/ip from the .CSV file, but the username and password will need to be retrieved from the user.
  + I’ve created a function to simplify getting user input. The function takes an argument which is the question and returns the string response:
    - Syntax: username = getInputAnswer(“\nEnter your device username”)
  + The following function retrieves a password from the user, with the exception that it does not display what was typed. This function is a Python supplied function.
    - Syntax: password = getpass(prompt=”\nEnter your device password: ”)
* Task: Load a configuration file into Juniper (push the configuration and control the router remotely)
  + The following code is used to load the configuration onto a router. This is within the “deployConfig” function:
    - Syntax: cu.load(template\_path=”config-template.conf”, template\_vars=tvars)
* Task: Verify and commit configuration (make sure its syntactically correct and commits successfully)
  + The following code is used to check the configuration and commit
    - Syntax: cu.pdiff()
    - Syntax: cu.commit\_check()
    - Syntax: commit\_status = cu.commit()
  + The printProgress function is used to display informational and error messages.
    - Syntax: printProgress(“INFO”,my\_host,”Performing commit check”)
    - Syntax: printProgress(“ERROR”,my\_host,”str(err))

BASIC PROGRAM FLOW LIST or DIAGRAM

1. Print the Title Page

print("\nWelcome to Junos Configuration Deployment Tool \n")

1. Get the csv and template file name by asking the User

#Get CSV File Name

fileList = getFileList(csv\_path)

csv\_file = getOptionAnswer("Choose a csv file", fileList)

#Get Config Template

fileList = getFileList(template\_path)

template\_file = getOptionAnswer("Choose a template file", fileList)

template\_file = template\_path + template\_file

1. Ask user for username and password

# Get username and password parameters

username = getInputAnswer("\nEnter your device username")

password = getpass(prompt="\nEnter your device password: ")

1. Start looping through the systems in CSV

# Loop through each record

for i in row\_csv\_list:

rownum = 0;

1. In Loop: Log into system

if deployConfig(device\_list\_dict,username,password,template\_file):

The deployConfig function:

def deployConfig(my\_device\_list\_dict, my\_username, my\_password, my\_config\_template\_file):

my\_hostname=""

try:

my\_hostname=my\_device\_list\_dict["mgmt\_ip"]

printProgress("INFO",my\_hostname,"Connecting to device through netconf.")

dev=Device(my\_hostname,user=my\_username,password=my\_password)

dev.open()

dev.timeout=3\*60

1. In Loop: Check if template file is "set" or "hierarchy" and load configuration

# Determine if template file is in "set" or "bracketed" format

if(isSet(my\_config\_template\_file)): rsp=cu.load(template\_path=my\_config\_template\_file,format='set',template\_vars=my\_device\_list\_dict)

else:

rsp=cu.load(template\_path=my\_config\_template\_file,template\_vars=my\_device\_list\_dict)

1. In Loop: Check the loaded configuration

printProgress("INFO",my\_hostname,"Performing diff between active and candidate config.")

cu.pdiff()

printProgress("INFO",my\_hostname,"Performing commit check")

if cu.commit\_check():

1. In Loop: Commit the configuration

printProgress("INFO",my\_hostname,"performing commit now.")

commit\_status=cu.commit()

printProgress("INFO",my\_hostname,"disconnecting from device.")

dev.close()

return commit\_status

1. End Loop

Final script and how to use it

**USING THE SCRIPT**

The operation of this script is very simple. It requires two files which you will be able asked for during execution. The two files will be discussed more below. After selecting the files, you will be asked for a user name and password. This user name and password MUST have super-user or administrator privileges. The devices you are accessing must also have “netconf” configured. The following configuration will enable netconf access:

set system services netconf ssh port 830

Once the user name and password are supplied, the script will start looping over the devices defined in the .CSV file. During the execution, the script will print the current status to the screen. This script is designed to be used for creating and pushing configuration changes to multiple devices. After all devices have been looped through, the script will end.

The script requires two files as input to create the configurations. One file is the template which has the configuration with variables that will change for each configuration. The other file is a .csv, which has the variables and parameters. The parameters will replace the variables in the template to create the system specific configuration file. There is one variable that is required called “mgmt\_ip”, this variable is what the script uses to connect to the device. All other variables used in the template and as headings in the .CSV are user defined. There are some guidelines below you should follow when creating variable names. The format is fairly easy; in the sample code below the variable is “hostname”. The template file uses double curly brackets “{{“ and “}}” to identify the variables in a template.

set system host-name {{ hostname }}

Use the following guidelines when creating variables…

* No spaces or tabs
* Can use alphanumeric characters
* Can use dashes “-“ and underscores “\_”
* Avoid special characters, they may cause problems

Below are some simple examples of template files and the variable .CSV file.

**Template Files**

The template can be in one of two formats, hierarchy or set. The two template formats below will create the same configuration files. You can use either format, although I would recommend using the “set” format. It is typically easier to read and also allows the use of “delete” statements for removing configuration.

**Set Format**

set system host-name {{ hostname }}

set system services netconf ssh port 830

set system login user {{ admin\_user }}

set system login user {{ admin\_user }} uid 2010

set system login user {{ admin\_user }} class super-user

set system login user {{ admin\_user }} authentication encrypted-password "$1$/uUuS3CK$/VC0.qWgKiLWXZaSgiHEw1"

set interfaces {{ access\_intf }} unit 0 family inet address {{ access\_ip }}

set interfaces {{ core\_intf }} unit 0 family inet address {{ core\_ip }}

set interfaces lo0 unit 0 family inet address {{ loopback\_ip }}

**Hierarchy Format**

system {

host-name {{ hostname }};

services {

netconf {

ssh {

port 830;

}

}

}

login {

user {{ admin\_user }} {

uid 2010;

class super-user;

authentication {

encrypted-password "$1$/uUuS3CK$/VC0.qWgKiLWXZaSgiHEw1"; ## SECRET-DATA

}

}

}

}

interfaces {

{{ access\_intf }} {

unit 0 {

family inet {

address {{ access\_ip }};

}

}

}

{{ core\_intf }} {

unit 0 {

family inet {

address {{ core\_ip }};

}

}

}

lo0 {

unit 0 {

family inet {

address {{ loopback\_ip }};

}

}

}

}

**Variable File (.CSV)**

The variable file is a .CSV file. As was mentioned above, the “mgmt\_ip” field is required.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| hostname | loopback\_ip | access\_intf | access\_ip | core\_intf | core\_ip | admin\_user | mgmt\_ip |
| vmx1 | 10.0.0.1/32 | ge-0/0/10 | 10.10.10.1/32 | ge-0/0/11 | 10.10.10.2/32 | nocadmin | 10.92.18.61 |
| vmx2 | 10.0.0.2/32 | ge-0/0/10 | 20.20.20.1/32 | ge-0/0/11 | 20.20.20.2/32 | nocadmin | 10.92.18.119 |
| vmx3 | 10.0.0.3/32 | ge-0/0/10 | 30.30.30.1/32 | ge-0/0/11 | 30.30.30.2/32 | cbpadmin | 10.92.18.55 |

**Python Code**

# File: jload.py

import sys,fileinput,code,re,csv

import utility

from utility import \*

from jnpr.junos import Device

from jnpr.junos.utils.config import Config

from lxml import etree

from getpass import getpass

csv\_path = '.\\csv\\'

template\_path = '.\\template\\'

def importCsv(csvfile):

try:

my\_file = open(csvfile)

csv\_data = csv.reader(my\_file)

my\_list = [row for row in csv\_data]

my\_list.pop(0)

my\_file.close()

return my\_list

except Exception,err:

print("ERROR: Encountered exception while iporting csv file :%s. Exception is : %s . \n Exiting the program.")%(csvfile,str(err))

sys.exit(1)

def isSet(templatefile):

isitSet = False

try:

my\_file = open(templatefile, 'r')

first\_line = my\_file.readline()

if (re.match( r'^set\s.+', first\_line)):

isitSet = True

return isitSet

except Exception,err:

print("ERROR: Encountered exception while checking template file format :%s. Exception is : %s . \n Exiting the program.")%(templatefile,str(err))

sys.exit(1)

def getCSVHeaders(csvfile):

try:

my\_file = open(csvfile)

csv\_data = csv.reader(my\_file)

header\_list = next(csv\_data)

my\_file.close()

return header\_list

except Exception,err:

print("ERROR: Encountered exception while retrieving csv headers : %s . Exception is : %s . \n Exiting the program.")%(csvfile,str(err))

sys.exit(1)

def printProgress(logtype,hostname,message):

print("%s:%s:%s")%(logtype,hostname,message)

def deployConfig(my\_device\_list\_dict, my\_username, my\_password, my\_config\_template\_file):

my\_hostname=""

try:

my\_hostname=my\_device\_list\_dict["mgmt\_ip"]

printProgress("INFO",my\_hostname,"Connecting to device through netconf.")

dev=Device(my\_hostname,user=my\_username,password=my\_password)

dev.open()

dev.timeout=3\*60

cu = Config(dev)

printProgress("INFO",my\_hostname,"Going to load template the config now.")

# Determine if template file is in "set" or "bracketed" format

if(isSet(my\_config\_template\_file)):

rsp=cu.load(template\_path=my\_config\_template\_file,format='set',template\_vars=my\_device\_list\_dict)

else:

rsp=cu.load(template\_path=my\_config\_template\_file,template\_vars=my\_device\_list\_dict)

printProgress("INFO",my\_hostname,"Performing diff between active and candidate config.")

cu.pdiff()

printProgress("INFO",my\_hostname,"Performing commit check")

if cu.commit\_check():

printProgress("INFO",my\_hostname,"commit check was successfull.")

printProgress("INFO",my\_hostname,"performing commit now.")

commit\_status=cu.commit()

printProgress("INFO",my\_hostname,"disconnecting from device.")

dev.close()

return commit\_status

else:

return False

except Exception,err:

printProgress("ERROR",my\_hostname,"Encountered exception while deploying config")

printProgress("ERROR",my\_hostname,str(err))

return False

def main():

print("\nWelcome to Junos Configuration Deployment Tool \n")

#Get CSV File Name

fileList = getFileList(csv\_path)

csv\_file = getOptionAnswer("Choose a csv file", fileList)

#Get Config Template

fileList = getFileList(template\_path)

template\_file = getOptionAnswer("Choose a template file", fileList)

template\_file = template\_path + template\_file

# Get username and password parameters

username = getInputAnswer("\nEnter your device username")

password = getpass(prompt="\nEnter your device password: ")

# Import CSV into array

row\_csv\_list = importCsv(csv\_path + csv\_file)

header\_csv\_list = getCSVHeaders(csv\_path + csv\_file)

device\_list\_dict = {}

# Loop through each record

for i in row\_csv\_list:

rownum = 0;

for h in header\_csv\_list:

device\_list\_dict[h]=i[rownum]

rownum += 1

if deployConfig(device\_list\_dict,username,password,template\_file):

printProgress("INFO",device\_list\_dict["hostname"],"Successfully deployed config on device.")

else:

printProgress("ERROR",device\_list\_dict["hostname"],"Config deployment failed!")

print("")

if \_\_name\_\_ == '\_\_main\_\_':

main()