CNIT-381

FALL 2020

PARAMIKO

Low Level Interactions

Introduction

Paramiko itself is a pure Python interface around SSH and networking concepts.

It uses the C programming language to obtain the highest performance for low level cryptographic concepts.

This section is especially important because SSH is probably the most to use the network protocol.

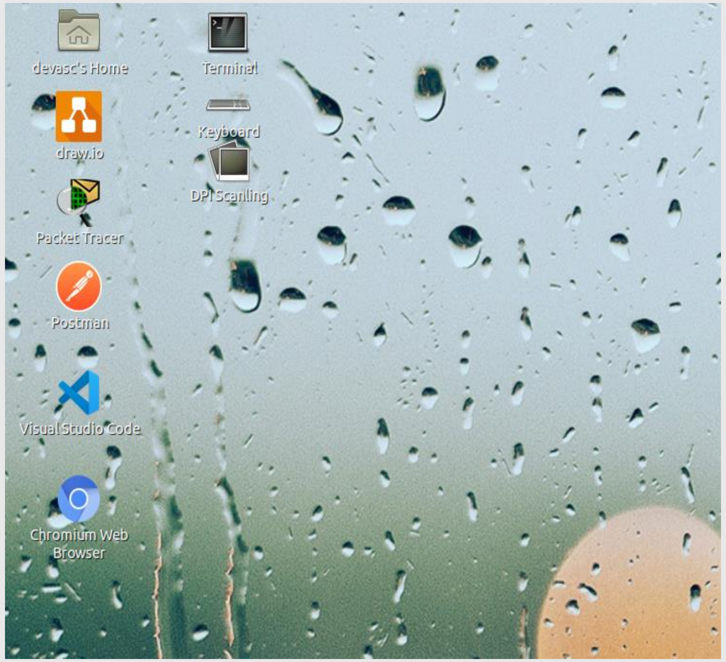
When a network engineer wants to configure or troubleshoot a networking device like a Cisco Router, security appliance or a Linux Enterprise server he will use in most cases SSH.

Introduction (cont)

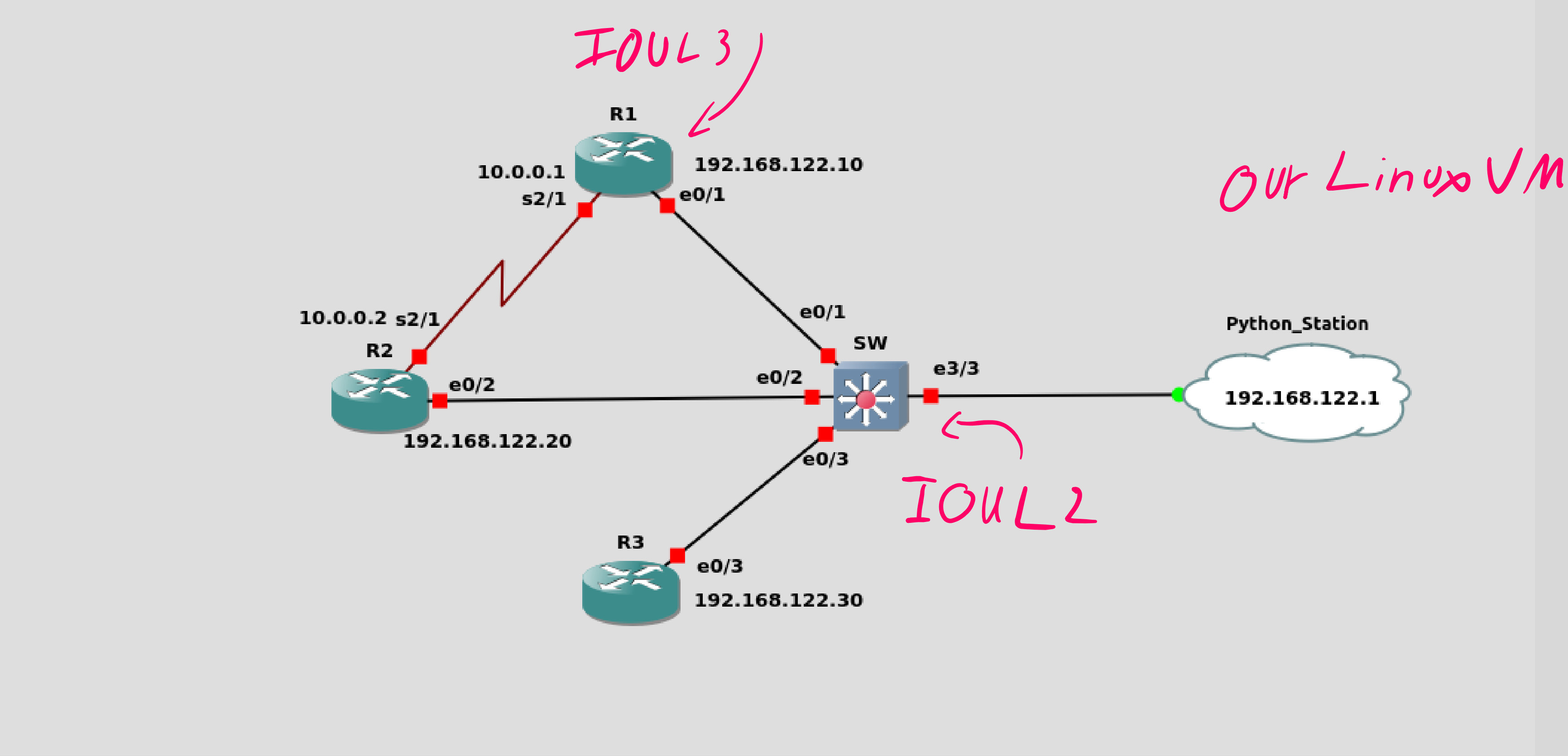
- Paramiko gives us the opportunity to automate the configuration of networking devices using Python scripts.
- Repetitive tasks which are bored but also prone to errors, can be easily automated to save
- Any device that can be configured using SSH can be also configured from Python using Paramiko.
- When a network engineer wants to configure or troubleshoot a networking device like a Cisco Router, security appliance or a Linux Enterprise server he will use in most cases SSH.

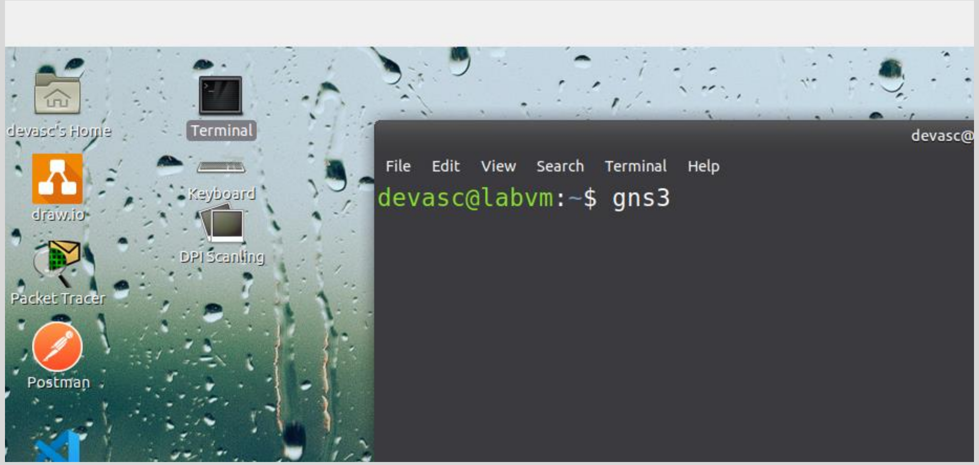
Your Lab Environment

- It is a 7.5GB linux Virtual Machine.
- With Python IDE is Visual Studio Code.
- GNS3 Simulation for Routers and Switches.
- Installed Library are: Paramiko, Netmiko, Ansible, NetConfig, Yang and more.

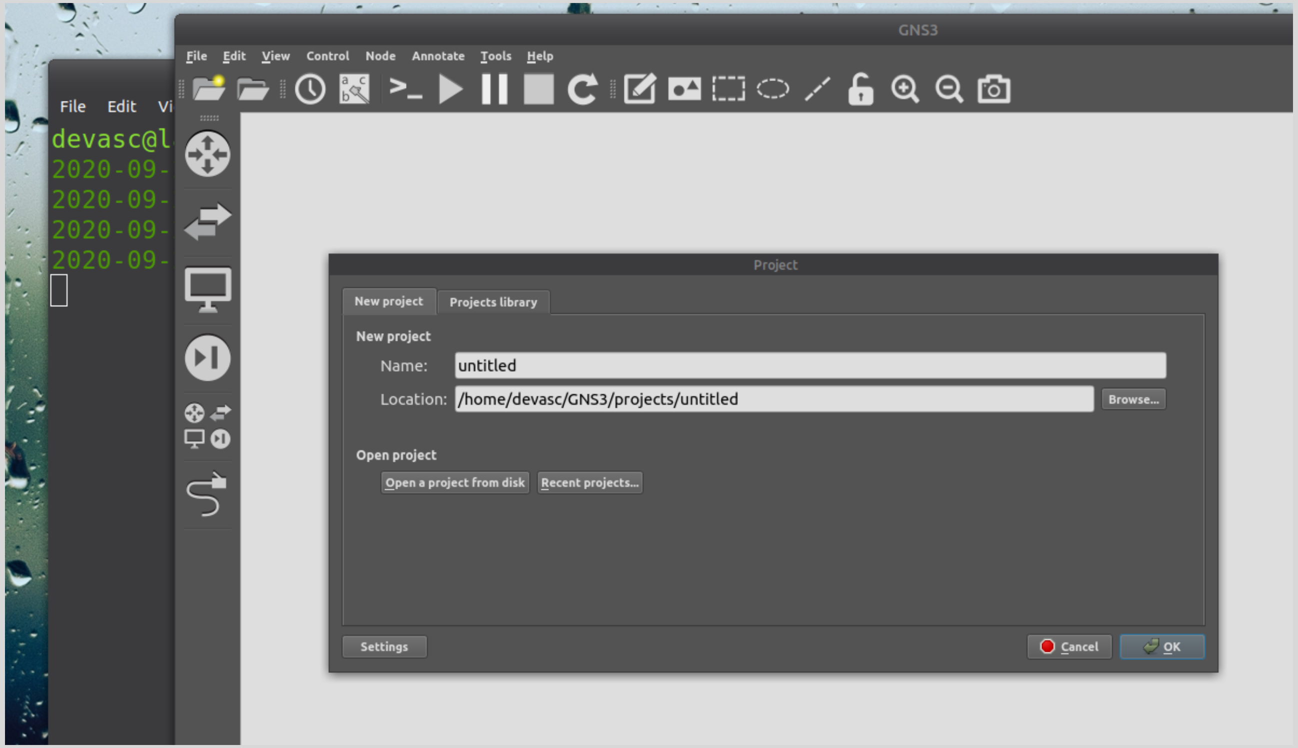


GNS3 Project





```
devasc@labvm:~$ gns3
```



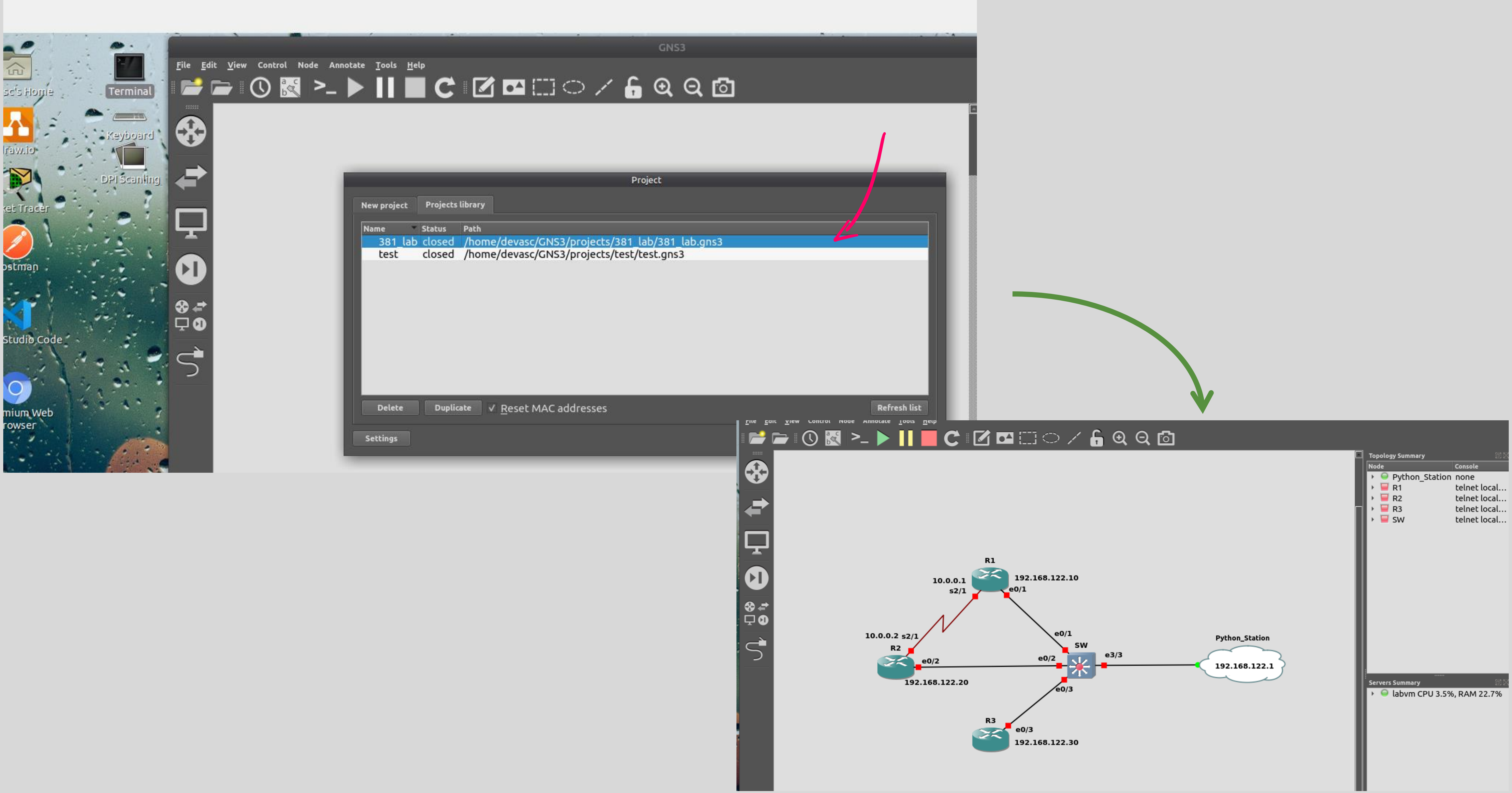
```
devasc@l
2020-09-
2020-09-
2020-09-
2020-09-
```

New project

Name: untitled

Location: /home/devasc/GNS3/projects/untitled

Open project



Project

New project | Projects library

Name	Status	Path
381_lab	closed	/home/devasc/GNS3/projects/381_lab/381_lab.gns3
test	closed	/home/devasc/GNS3/projects/test/test.gns3

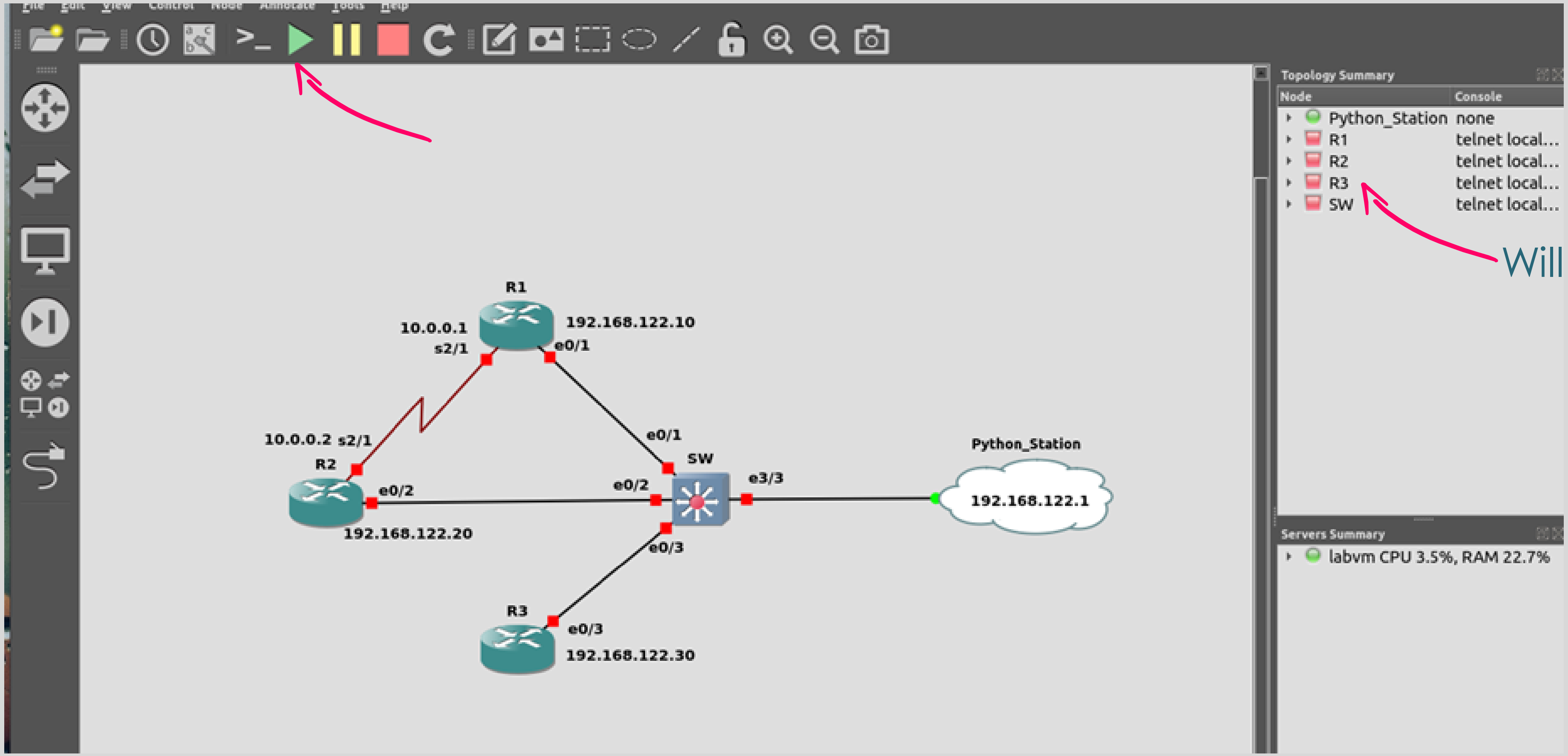
☑ Reset MAC addresses

Topology Summary

Node	Console
Python_Station	none
R1	telnet local...
R2	telnet local...
R3	telnet local...
SW	telnet local...

Servers Summary

labvm CPU 3.5%, RAM 22.7%



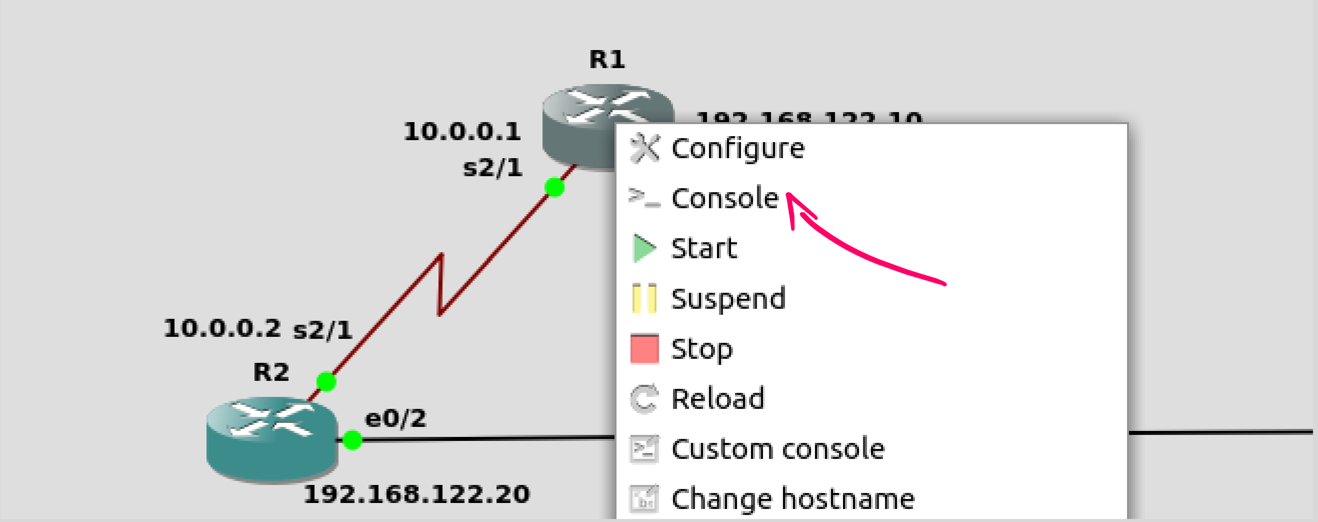
Topology Summary

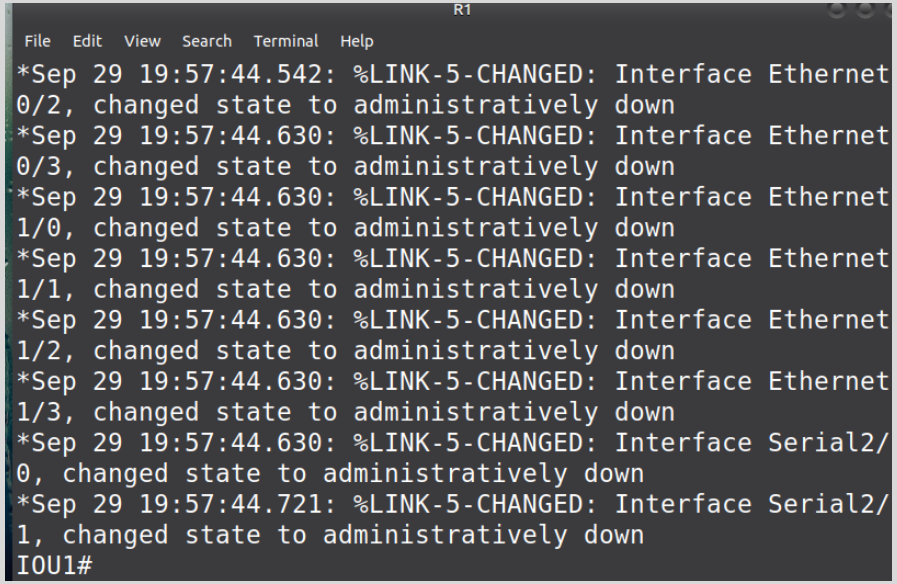
Node	Console
🟢 Python_Station	none
🟥 R1	telnet local...
🟥 R2	telnet local...
🟥 R3	telnet local...
🟥 SW	telnet local...

Servers Summary

🟢 labvm CPU 3.5%, RAM 22.7%

Will go Green

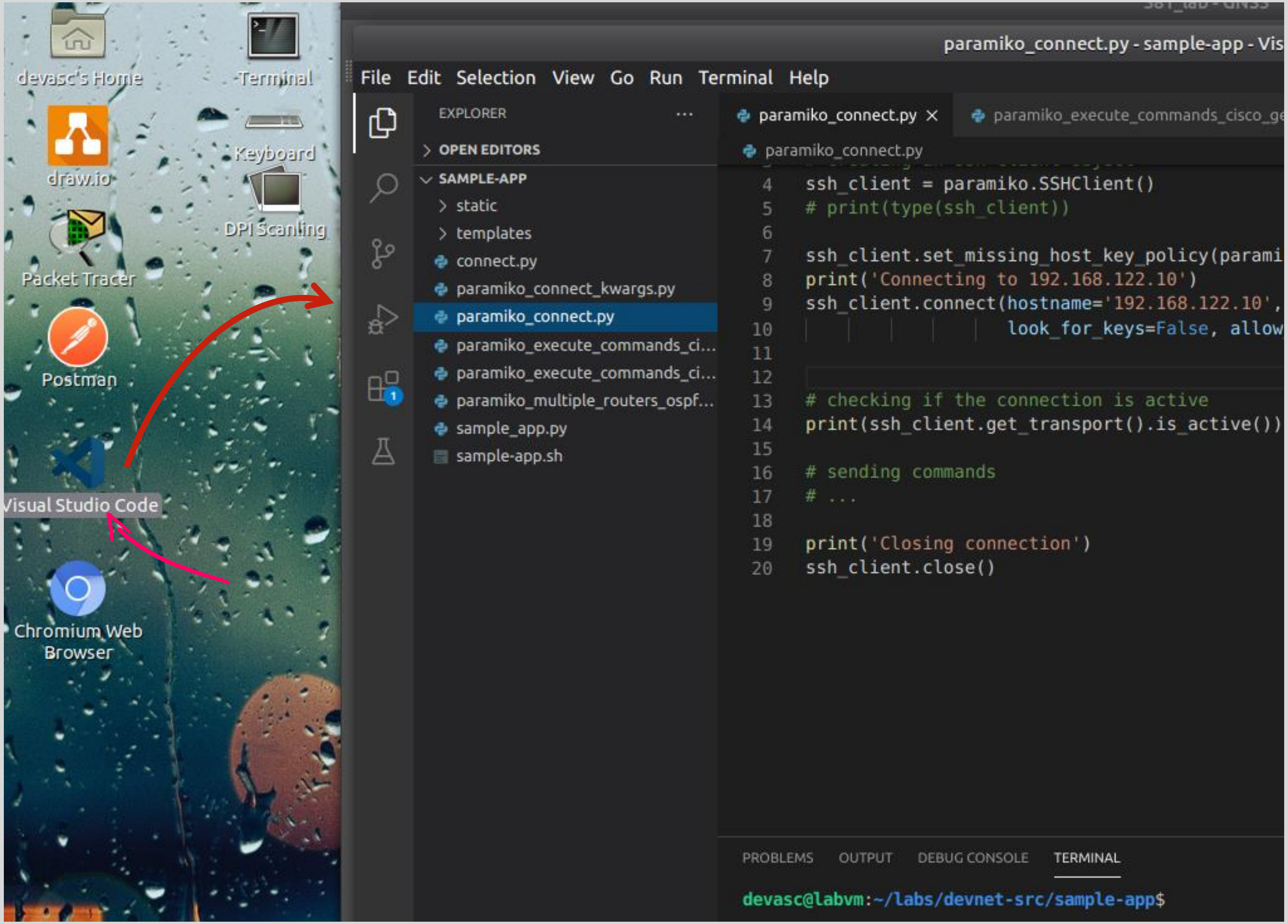




```
*Sep 29 19:57:44.542: %LINK-5-CHANGED: Interface Ethernet
0/2, changed state to administratively down
*Sep 29 19:57:44.630: %LINK-5-CHANGED: Interface Ethernet
0/3, changed state to administratively down
*Sep 29 19:57:44.630: %LINK-5-CHANGED: Interface Ethernet
1/0, changed state to administratively down
*Sep 29 19:57:44.630: %LINK-5-CHANGED: Interface Ethernet
1/1, changed state to administratively down
*Sep 29 19:57:44.630: %LINK-5-CHANGED: Interface Ethernet
1/2, changed state to administratively down
*Sep 29 19:57:44.630: %LINK-5-CHANGED: Interface Ethernet
1/3, changed state to administratively down
*Sep 29 19:57:44.630: %LINK-5-CHANGED: Interface Serial2/
0, changed state to administratively down
*Sep 29 19:57:44.721: %LINK-5-CHANGED: Interface Serial2/
1, changed state to administratively down
IOU1#
```

Setup SSH Server

- NEED A HOSTNAME.
- IP DOMAIN-NAME
- CRYPTO KEY RSA 2048 BITS
- IP SSH VERSION 2
- ENABLE IN VTY 0 4
- LOGIN LOCAL



```
ssh_client = paramiko.SSHClient()
# print(type(ssh_client))

ssh_client.set_missing_host_key_policy(parami
print('Connecting to 192.168.122.10')
ssh_client.connect(hostname='192.168.122.10',
                   look_for_keys=False, allow


# checking if the connection is active
print(ssh_client.get_transport().is_active())

# sending commands
# ...

print('Closing connection')
ssh_client.close()
```

```
devasc@labvm:~/labs/devnet-src/sample-app$
```

module

object

Att

Methods

Variable

```
import paramiko

ssh_client = paramiko.SSHClient() # creating an ssh client object
print(type(ssh_client))
```

```
<class 'paramiko.client.SSHClient'>
```

Method

```
import paramiko

ssh_client = paramiko.SSHClient() # creating an ssh client object

ssh_client.connect(hostname='192.168.122.10', port='22', username='cisco', password='cisco',
             look_for_keys=False, allow_agent=False)
```

Parameters of method Connect

Looks for SSH keys

Optional SSH Agent

```
Raise SSHException(
paramiko.ssh_exception.SSHException: Server '[192.168.122.10]:22' not found in known_hosts
```

```
import paramiko

ssh_client = paramiko.SSHClient() # creating an ssh client object

ssh_client.set_missing_host_key_policy(paramiko.AutoAddPolicy())

ssh_client.connect(hostname='192.168.122.10', port='22', username='cisco', password='cisco',
                look_for_keys=False, allow_agent=False)

print(ssh_client.get_transport().is_active()) # checking if  the connection is active
```

Save key for 1st time

```
True
```

```
import paramiko

ssh_client = paramiko.SSHClient() # creating an ssh client object

ssh_client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
print(“connecting to 192.168.122.10”)
ssh_client.connect(hostname='192.168.122.10', port='22', username='cisco', password='cisco',
                look_for_keys=False, allow_agent=False)

print(ssh_client.get_transport().is_active()) # checking if  the connection is active


print('Closing connection')
ssh_client.close()
```

terminate conn

```
True
Closing connection
```

Dictionary

```
import paramiko

ssh_client = paramiko.SSHClient() # creating an ssh client object

ssh_client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
#ssh_client.connect(hostname='192.168.122.10', port='22', username='cisco', password='cisco’,
#                   look_for_keys=False, allow_agent=False)
router = {'hostname': ‘192.168.122.10', 'port': '22', 'username’:’cicso', 'password':'cisco’}
```

key value

```
ssh_client.connect(**router, look_for_keys=False, allow_agent=False)

print(f’Connecting to {router[“hostname”]}’)
```

f-format Key

```
print(ssh_client.get_transport().is_active()) # checking if  the connection is active
print('Closing connection')
ssh_client.close()
```

```
Connecting to 192.168.122.10
True
Closing connection
```

```
import paramiko
import time  -> for sleep
ssh_client = paramiko.SSHClient() # creating an ssh client object
ssh_client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
router = {'hostname': ‘192.168.122.10', 'port': '22', 'username’:’cicso', 'password':'cisco’}
ssh_client.connect(**router, look_for_keys=False, allow_agent=False)
print(f’Connecting to {router[“hostname”]}’)

shell = ssh_client.invoke_shell()  Shell object
shell.send('show version\n’)       Send commands

time.sleep(1)  ——> wait abit
output = shell.recv(10000)  —> got output in bytes
print(output)

if  print(ssh_client.get_transport().is_active()) == True:
   print('Closing connection’)                                 } Close if active
   ssh_client.close()
```

```
Connecting to 192.168.122.10
b'\r\nR1>show version\r\nCisco IOS Software, Linux Software (I86BI_LINUX-
ADVENTERPRISEK9-M), Version 15.2(2.15)T, ENGINEERING WEEKLY BUILD, synced to
V151_4_M3_5\r\nCopyright (c) 1986-2012 by Cisco Systems, Inc.\r\nCompiled Sun
29-Jan-12 02:33 by \r\n\r\nROM: .............
```

```
import paramiko
import time
ssh_client = paramiko.SSHClient() # creating an ssh client object
ssh_client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
router = {'hostname': ‘192.168.122.10', 'port': '22', 'username’:’cicso', 'password':'cisco’}
ssh_client.connect(**router, look_for_keys=False, allow_agent=False)
print(f’Connecting to {router[“hostname”]}’)
shell = ssh_client.invoke_shell()
shell.send('show version\n’)

time.sleep(1)
output = shell.recv(10000)
output = output.decode('utf-8’)
print(output)

if  print(ssh_client.get_transport().is_active()) == True:
    print('Closing connection’)
    ssh_client.close()
```

decode bytes to string

```
.......compliance with U.S. and local country laws. By using this product you
agree to comply with applicable laws and regulations. If you are unable
to comply with U.S. and local laws, return this product immediately.
 --More--
Closing connection
```

```
import paramiko
import time
ssh_client = paramiko.SSHClient() # creating an ssh client object
ssh_client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
router = {'hostname': ‘192.168.122.10', 'port': '22', 'username’:’cicso', 'password':'cisco’}
ssh_client.connect(**router, look_for_keys=False, allow_agent=False)
print(f’Connecting to {router[“hostname”]}’)
shell = ssh_client.invoke_shell()
shell.send('show version\n’)
shell.send('terminal length 0\n’)
time.sleep(1)
output = shell.recv(10000)
output = output.decode('utf-8’)
print(output)

if  print(ssh_client.get_transport().is_active()) == True:
    print('Closing connection’)
    ssh_client.close()
```

output all without hitting enter

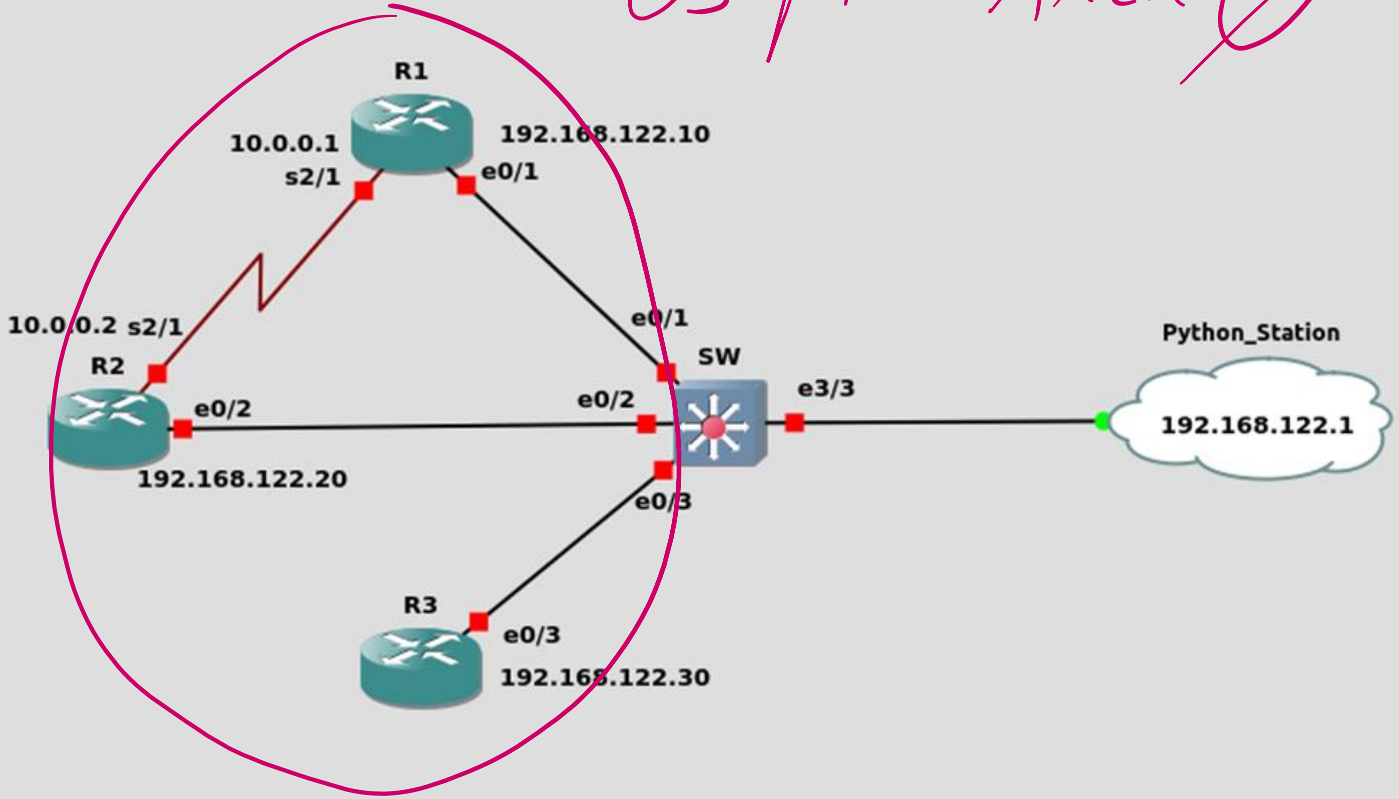
```
………..
Configuration register is 0x0

R1>
Closing connection
```

```
import paramiko
import time
import getpass
ssh_client = paramiko.SSHClient() # creating an ssh client object
ssh_client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
password = getpass.getpass('Enter password:')
router = {'hostname': ‘192.168.122.10', 'port': '22', 'username’:’cicso', 'password’:password}
ssh_client.connect(**router, look_for_keys=False, allow_agent=False)
print(f’Connecting to {router[“hostname”]}’)
shell = ssh_client.invoke_shell()
shell.send('show version\n’)
shell.send('terminal length 0\n’)
time.sleep(1)
output = shell.recv(10000)
output = output.decode('utf-8’)
print(output)
if  print(ssh_client.get_transport().is_active()) == True:
    print('Closing connection’)
    ssh_client.close()
```

get pass from console

```
Enter password:
```

OSPF Area 0


R1
10.0.0.1
s2/1
192.168.122.10
e0/1
10.0.0.2 s2/1
R2
e0/2
192.168.122.20
e0/1
SW
e0/2
e0/3
e3/3
R3
e0/3
192.168.122.30
Python_Station
192.168.122.1

OSPF

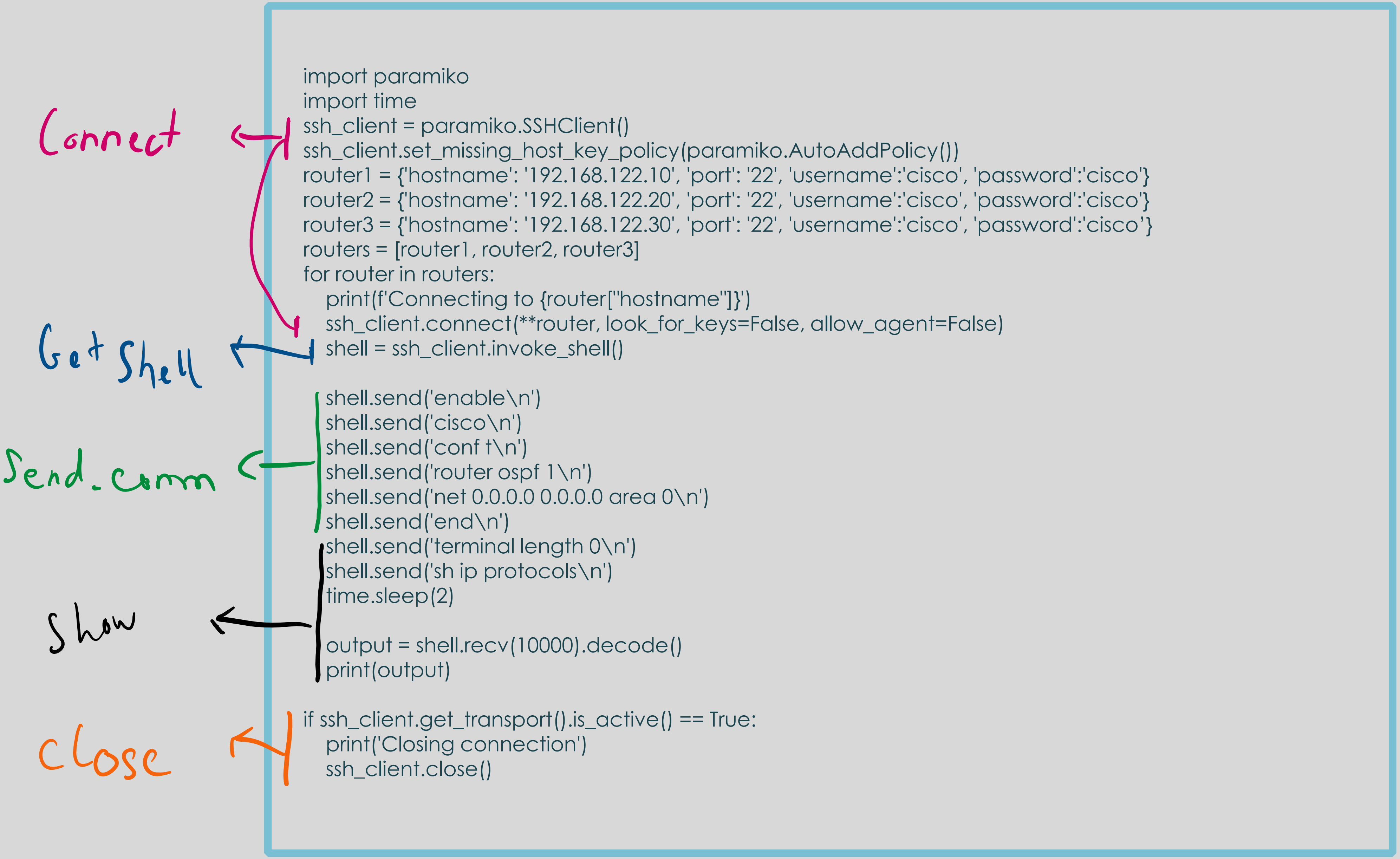
List ←

```
import paramiko
import time
ssh_client = paramiko.SSHClient()
ssh_client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
router1 = {'hostname': '192.168.122.10', 'port': '22', 'username':'cisco', 'password':'cisco'}
router2 = {'hostname': '192.168.122.20', 'port': '22', 'username':'cisco', 'password':'cisco'}
router3 = {'hostname': '192.168.122.30', 'port': '22', 'username':'cisco', 'password':'cisco'}
routers = [router1, router2, router3]
for router in routers:
    print(f'Connecting to {router["hostname"]}')
    ssh_client.connect(**router, look_for_keys=False, allow_agent=False)
    shell = ssh_client.invoke_shell()

    shell.send('enable\n')
    shell.send('cisco\n')
    shell.send('conf t\n')
    shell.send('router ospf 1\n')
    shell.send('net 0.0.0.0 0.0.0.0 area 0\n')
    shell.send('end\n')
    shell.send('terminal length 0\n')
    shell.send('sh ip protocols\n')
    time.sleep(2)

    output = shell.recv(10000).decode()
    print(output)

if ssh_client.get_transport().is_active() == True:
    print('Closing connection')
    ssh_client.close()
```

→ enable pass

enable ospf on all ints

→ check ospf



```
import paramiko
import time
ssh_client = paramiko.SSHClient()
ssh_client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
router1 = {'hostname': '192.168.122.10', 'port': '22', 'username':'cisco', 'password':'cisco'}
router2 = {'hostname': '192.168.122.20', 'port': '22', 'username':'cisco', 'password':'cisco'}
router3 = {'hostname': '192.168.122.30', 'port': '22', 'username':'cisco', 'password':'cisco’}
routers = [router1, router2, router3]
for router in routers:
    print(f'Connecting to {router["hostname"]}')
    ssh_client.connect(**router, look_for_keys=False, allow_agent=False)
    shell = ssh_client.invoke_shell()

    shell.send('enable\n')
    shell.send('cisco\n')
    shell.send('conf t\n')
    shell.send('router ospf 1\n')
    shell.send('net 0.0.0.0 0.0.0.0 area 0\n')
    shell.send('end\n')
    shell.send('terminal length 0\n')
    shell.send('sh ip protocols\n')
    time.sleep(2)

    output = shell.recv(10000).decode()
    print(output)

if ssh_client.get_transport().is_active() == True:
    print('Closing connection')
    ssh_client.close()
```

Function

```python
import paramiko
import time

def connect(server_ip, server_port, user, passwd):
    ssh_client = paramiko.SSHClient()
    ssh_client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    print(f'Connecting to {server_ip}')
    ssh_client.connect(hostname=server_ip, port=server_port, username=user, password=passwd,
                       look_for_keys=False, allow_agent=False)
    return ssh_client

def get_shell(ssh_client):
    shell = ssh_client.invoke_shell()
    return shell

def send_command(shell, command):
    print(f'Sending command: {command}')
    shell.send(command + '\n')
    #time.sleep(timeout)

def show(shell, command, n=10000, timeout = 1):
    print(f'Sending command: {command}')
    shell.send('terminal length 0\n')
    shell.send(command + '\n')
    time.sleep(timeout)
    output = shell.recv(n)
    output = output.decode()
    print (output)
    return output

def close(ssh_client):
    if ssh_client.get_transport().is_active() == True:
        print('Closing connection')
        ssh_client.close()
```

Save to myParamiko.py

```
import myParamiko as m

router1 = {'server_ip': '192.168.122.10', 'server_port': '22', 'user':'cisco', 'passwd':'cisco'}
router2 = {'server_ip': '192.168.122.20', 'server_port': '22', 'user':'cisco', 'passwd':'cisco'}
router3 = {'server_ip': '192.168.122.30', 'server_port': '22', 'user':'cisco', 'passwd':'cisco'}

routers = [router1, router2, router3]

for router in routers:
   print(f'Connecting to {router["server_ip"]}’)
   ssh_client = m.connect(**router)
   shell = m.get_shell(ssh_client)

   m.send_command(shell,'enable')
   m.send_command(shell,'cisco')
   m.send_command(shell, 'conf t')
   m.send_command(shell,'router ospf 1')
   m.send_command(shell,'net 0.0.0.0 0.0.0.0 area 0')
   m.send_command(shell,'end’)
   m.show(shell,'show ip protocols’)
   m.close(ssh_client)
```

call connect

Call get shell

Call Send_command

call show

Call close

Save to routers.txt

```
[{'server_ip': '192.168.122.10', 'server_port': '22', 'user':'cisco', 'passwd':'cisco'},
{'server_ip': '192.168.122.20', 'server_port': '22', 'user':'cisco', 'passwd':'cisco'},
{'server_ip': '192.168.122.30', 'server_port': '22', 'user':'cisco', 'passwd':'cisco'}]
```

Dictionary

List

List [Dictionaries]

Read file

turn to list

```
def get_list_from_file(filename):
    with open(filename) as f:
        data = ast.literal_eval(f.read())
        f.close()
        return data
```

Return list

Add this function to myParamiko.py

Routers list

```
import myNewParamiko as m

routers = m.get_list_from_file('routers.txt')

for router in routers:
    print(f'Connecting to {router["server_ip"]}')
    ssh_client = m.connect(**router)
    shell = m.get_shell(ssh_client)

    m.send_command(shell,'enable')
    m.send_command(shell,'cisco')
    m.send_command(shell, 'conf t')
    m.send_command(shell,'router ospf 1')
    m.send_command(shell,'net 0.0.0.0 0.0.0.0 area 0')
    m.send_command(shell,'end’)

    m.show(shell,'show ip protocols’)

    m.close(ssh_client)
```

Backup Config

1st router ←

```
import myNewParamiko as m

routers = m.get_list_from_file('routers.txt')
router = routers[0]

print(f'Connecting to {router["server_ip"]}')
ssh_client = m.connect(**router)
shell = m.get_shell(ssh_client)
m.send_command(shell, 'terminal length 0')
m.send_command(shell, 'enable')
m.send_command(shell, 'cisco')
output = m.show(shell, 'show run')
```

Building configuration...

~~Current configuration : 2106 bytes~~
~~!~~
! ~~Last configuration change at 04:13:40 UTC Thu Oct 1 2020 by cisco~~
version 15.2
service timestamps debug datetime msec
……….
end
~~#R1~~

```
import myNewParamiko as m

routers = m.get_list_from_file('routers.txt')
router = routers[0]

print(f'Connecting to {router["server_ip"]}')
ssh_client = m.connect(**router)
shell = m.get_shell(ssh_client)
m.send_command(shell, 'terminal length 0')
m.send_command(shell, 'enable')
m.send_command(shell, 'cisco')
output = m.show(shell, 'show run’)
output_list = output.splitlines()
print(output_list)
```

split to lines
add each line to list

0 1 2 3 4 5 6 7 8 9 10 11

```
['', 'R1>terminal length 0', 'R1>enable', 'Password: ', 'R1#terminal length 0', 'R1#show
run', 'Building configuration...', '', 'Current configuration : 2106 bytes', '!', '! Last
configuration change at 04:13:40 UTC Thu Oct 1 2020 by cisco', 'version 15.2', 'service
timestamps debug datetime msec’ ……….. ‘#R1’]
```

-1

slice [11:-1]

```
import myNewParamiko as m

routers = m.get_list_from_file('routers.txt')
router = routers[0]

print(f'Connecting to {router["server_ip"]}')
ssh_client = m.connect(**router)
shell = m.get_shell(ssh_client)
m.send_command(shell, 'terminal length 0')
m.send_command(shell, 'enable')
m.send_command(shell, 'cisco')
output = m.show(shell, 'show run’)
output_list = output.splitlines()
output_list = output_list[11:-1]
print(output_list)
```

Slicing

```
['version 15.2', 'service timestamps debug datetime msec', 'service timestamps log
datetime msec', 'no service password-encryption', '!', 'hostname R1', '!', 'boot-start-
marker', 'boot-end-marker', '!'.......... '!', 'end', '']
```

need to join

```
import myNewParamiko as m

routers = m.get_list_from_file('routers.txt')
router = routers[0]
print(f'Connecting to {router["server_ip"]}')
ssh_client = m.connect(**router)
shell = m.get_shell(ssh_client)
m.send_command(shell, 'terminal length 0')
m.send_command(shell, 'enable')
m.send_command(shell, 'cisco')
output = m.show(shell, 'show run’)
output_list = output.splitlines()
output_list = output_list[11:-1]
output = '\n'.join(output_list)
print(output_list)
```

join each element with newline

```
version 15.2
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
........
!
end
```

✓

let's save to file

```
import myNewParamiko as m

routers = m.get_list_from_file('routers.txt')
router = routers[0]
print(f'Connecting to {router["server_ip"]}')
ssh_client = m.connect(**router)
shell = m.get_shell(ssh_client)
m.send_command(shell, 'terminal length 0')
m.send_command(shell, 'enable')
m.send_command(shell, 'cisco')
output = m.show(shell, 'show run’)
output_list = output.splitlines()
output_list = output_list[11:-1]
output = '\n'.join(output_list)
print(output_list)

with open('backupR1.txt', 'w') as f:
    f.write(output)

m.close(ssh_client)
```

→ write output to backupR1.txt

But how about version control !!!

```
import myNewParamiko as m

routers = m.get_list_from_file('routers.txt')
router = routers[0]
print(f'Connecting to {router["server_ip"]}')
ssh_client = m.connect(**router)
shell = m.get_shell(ssh_client)
m.send_command(shell, 'terminal length 0')
m.send_command(shell, 'enable')
m.send_command(shell, 'cisco')
output = m.show(shell, 'show run’)
output_list = output.splitlines()
output_list = output_list[11:-1]
output = '\n'.join(output_list)

from datetime import datetime
now = datetime.now()
year = now.year
month = now.month
day = now.day
hour = now.hour
minute = now.minute

file_name = f'{router["server_ip"]}_{year}-{month}-{day}.txt’

with open(file_name, 'w') as f:
    f.write(output)
```

How about multi router ???

get time

192.168.122.10_2020_9_30.txt

```
import myNewParamiko as m

def backup(router):
    print(f'Connecting to {router["server_ip"]}')
    ssh_client = m.connect(**router)
    shell = m.get_shell(ssh_client)
    m.send_command(shell, 'terminal length 0')
    m.send_command(shell, 'enable')
    m.send_command(shell, 'cisco')
    output = m.show(shell, 'show run’)

    output_list = output.splitlines()
    output_list = output_list[11:-1]
    output = '\n'.join(output_list)

    from datetime import datetime
    now = datetime.now()
    year = now.year
    month = now.month
    day = now.day
    hour = now.hour
    minute = now.minute

    file_name = f'{router["server_ip"]}_{year}-{month}-{day}.txt'
    print(file_name)

    with open(file_name, 'w') as f:
        f.write(output)

    m.close(ssh_client)
```

convert to function

then save to myBackupRouter.py

```
import myNewParamiko as m
import myBackupRouter as bk

routers = m.get_list_from_file('routers.txt')

for router in routers:
    bk.backup(router)
```

→ Backup module

But! it's too slow

```
import myNewParamiko as m
import myBackupRouter as bk
import threading

routers = m.get_list_from_file('routers.txt')
threads = list()

for router in routers:
th = threading.Thread(target=bk.backup, args=(router,))
   threads.append(th)  # appending the thread to the list

for th in threads:
   th.start()

for th in threads:
   th.join()
```

Thread list

Backup function

list of routers

start Threads

wait for threads to finish