
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.

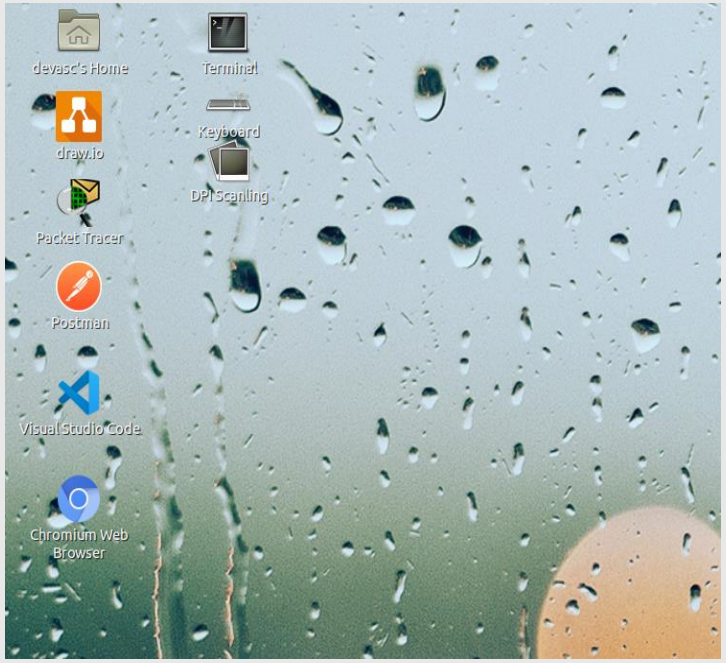
It is a 7.5GB linux Virtual Machine.

With Python IDE is Visual Studio Code.

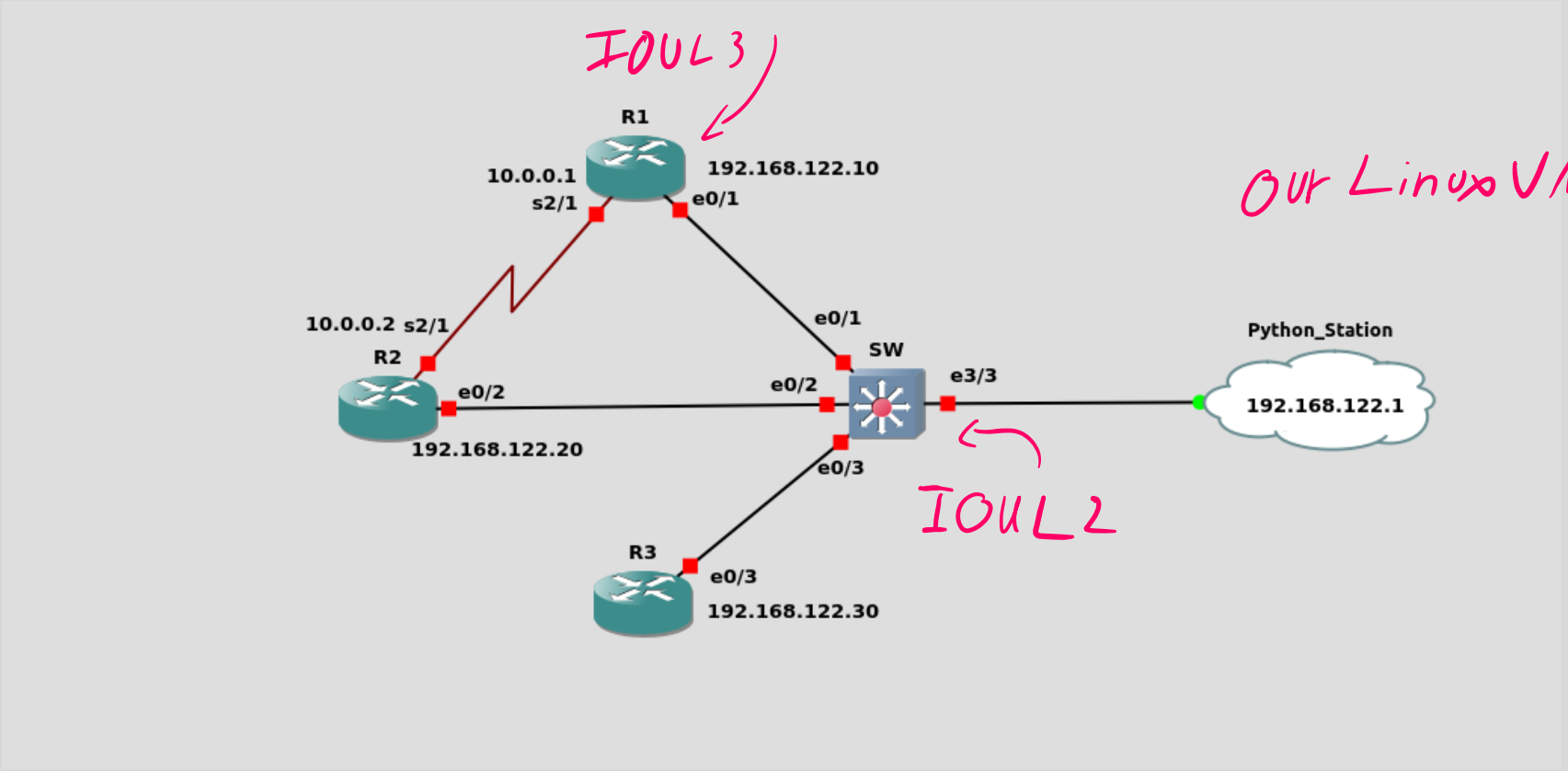
GNS3 Simulation for Routers and Switches.

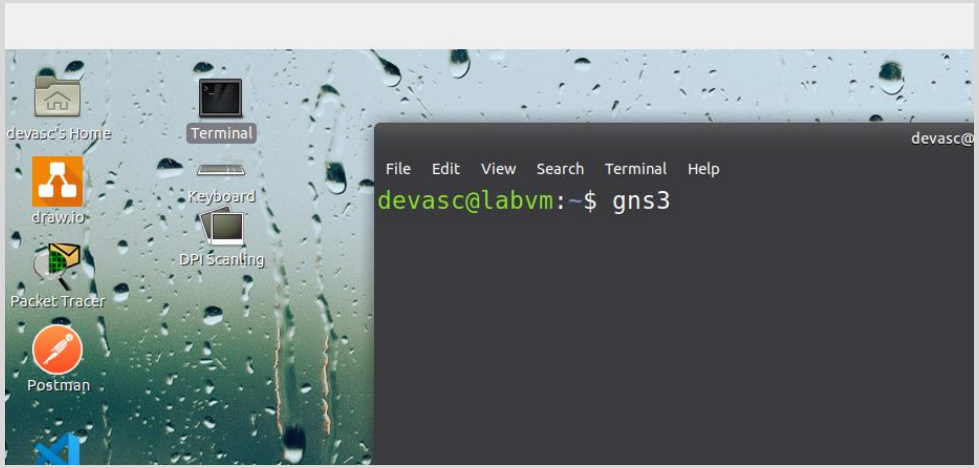
Installed Library are: Paramiko, Netmiko, Ansible, NetConfig, Yang and more.

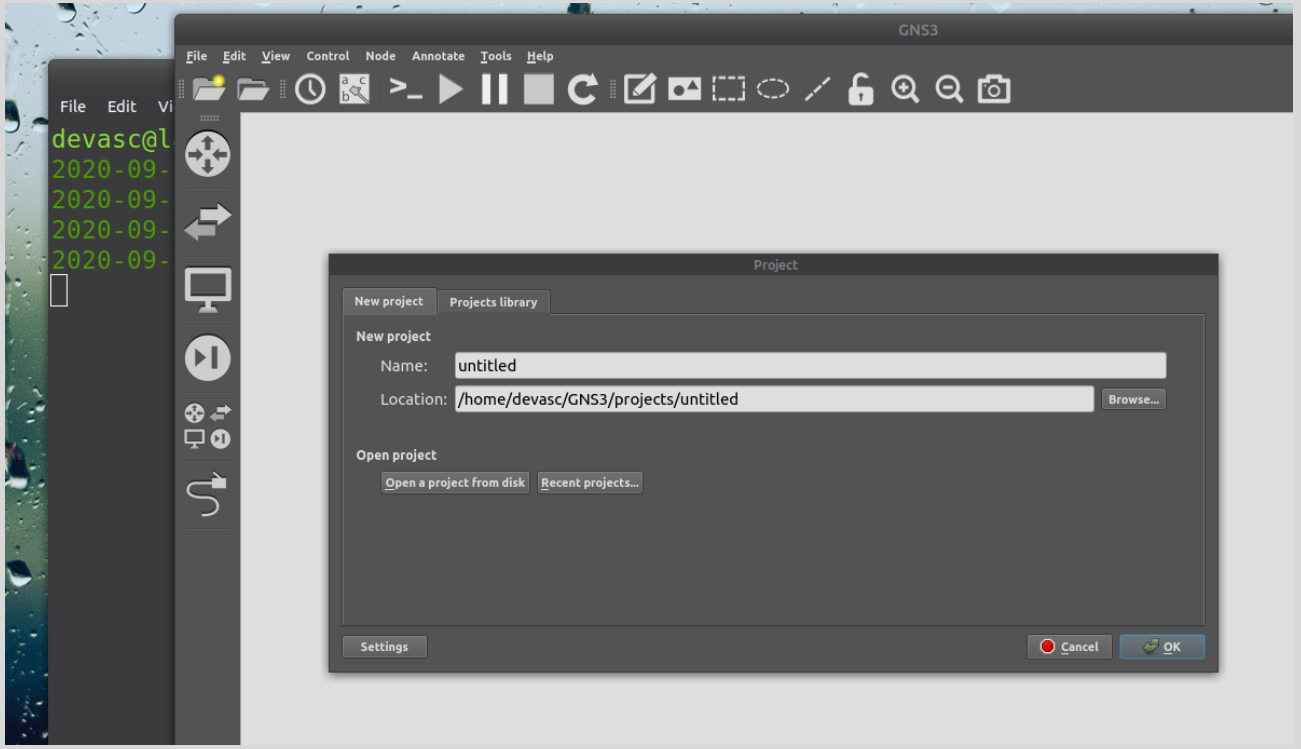
Your Lab Environment

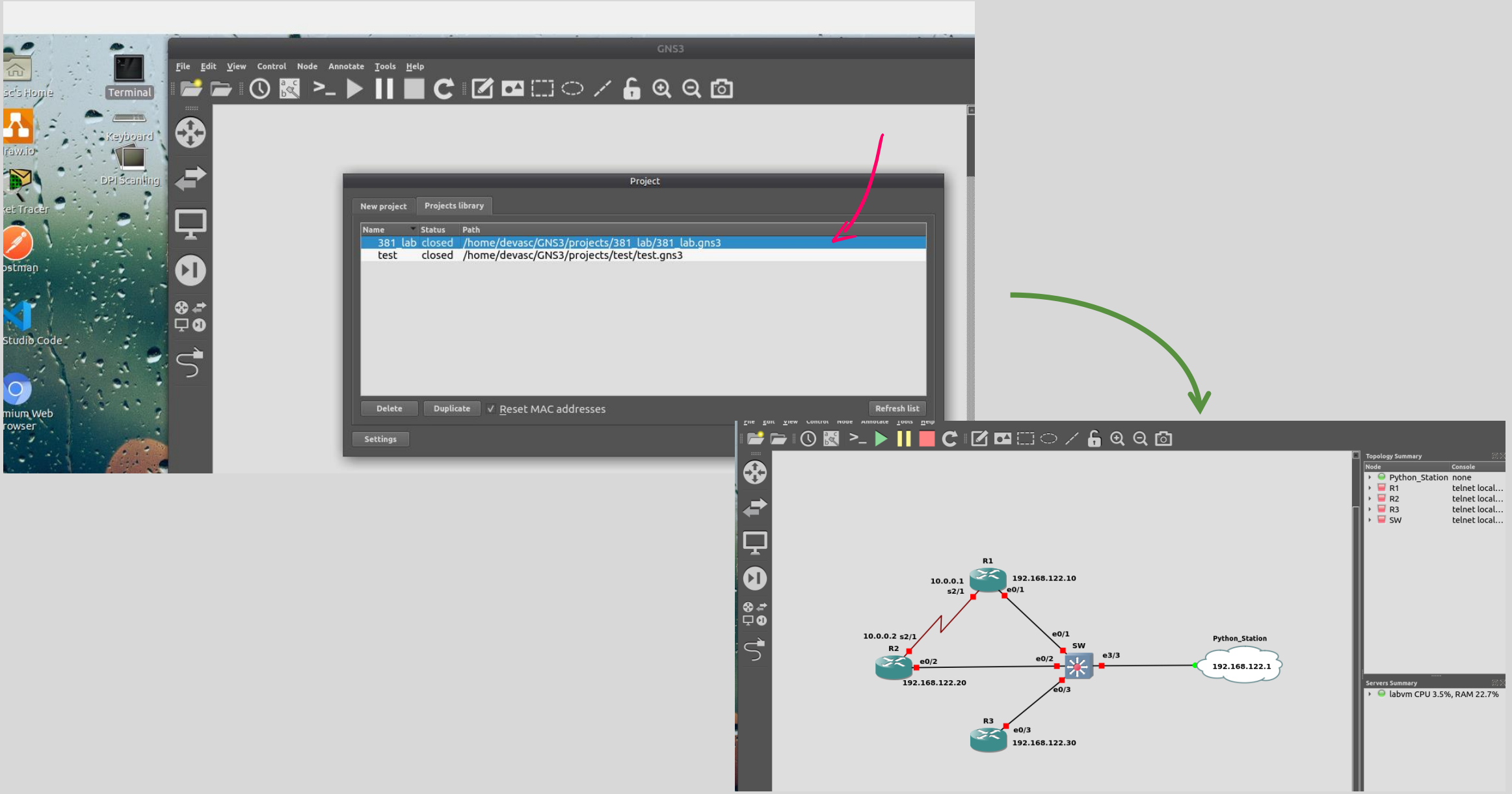


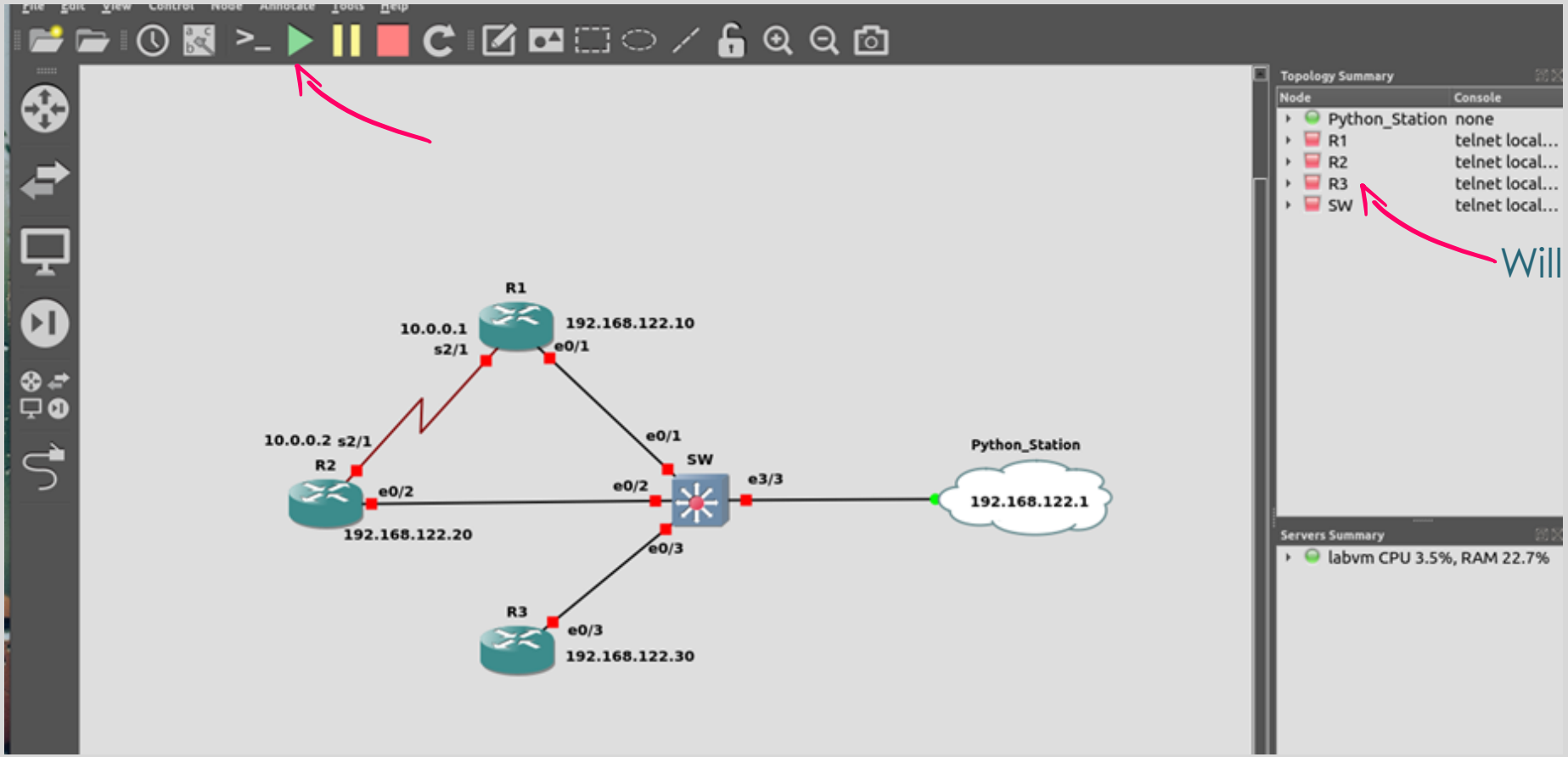
GNS3 Project



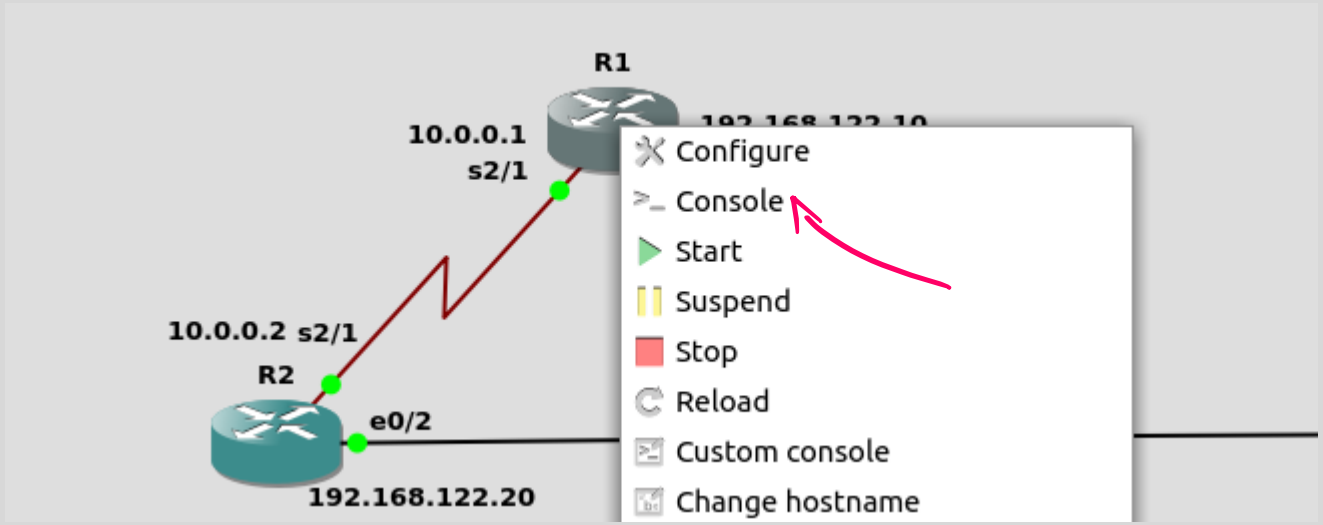

devasc's Home
Terminal
draw.io
Keyboard
DPI Scaling
Packet Tracer
Postman
devasc@
File Edit View Search Terminal Help
devasc@labvm:~$ gns3


GNS3
File Edit View Control Node Annotate Tools Help
File Edit Vi
devasc@l
2020-09-
2020-09-
2020-09-
2020-09-
Project
New project
Projects library
New project
Name: untitled
Location: /home/devasc/GNS3/projects/untitled
Browse...
Open project
Open a project from disk
Recent projects...
Settings
Cancel
OK


GNS3
File Edit View Control Node Annotate Tools Help
Terminal
Keyboard
DPI Scanling
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
Delete
Duplicate
Reset MAC addresses
Refresh list
Settings
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%
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
e3/3
Python_Station
192.168.122.1
e0/3
R3
e0/3
192.168.122.30



Will go Green




```
R1
File  Edit  View  Search  Terminal  Help
*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

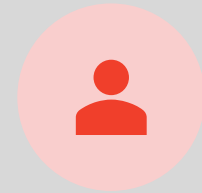
NEED A HOSTNAME.

IP DOMAIN-NAME

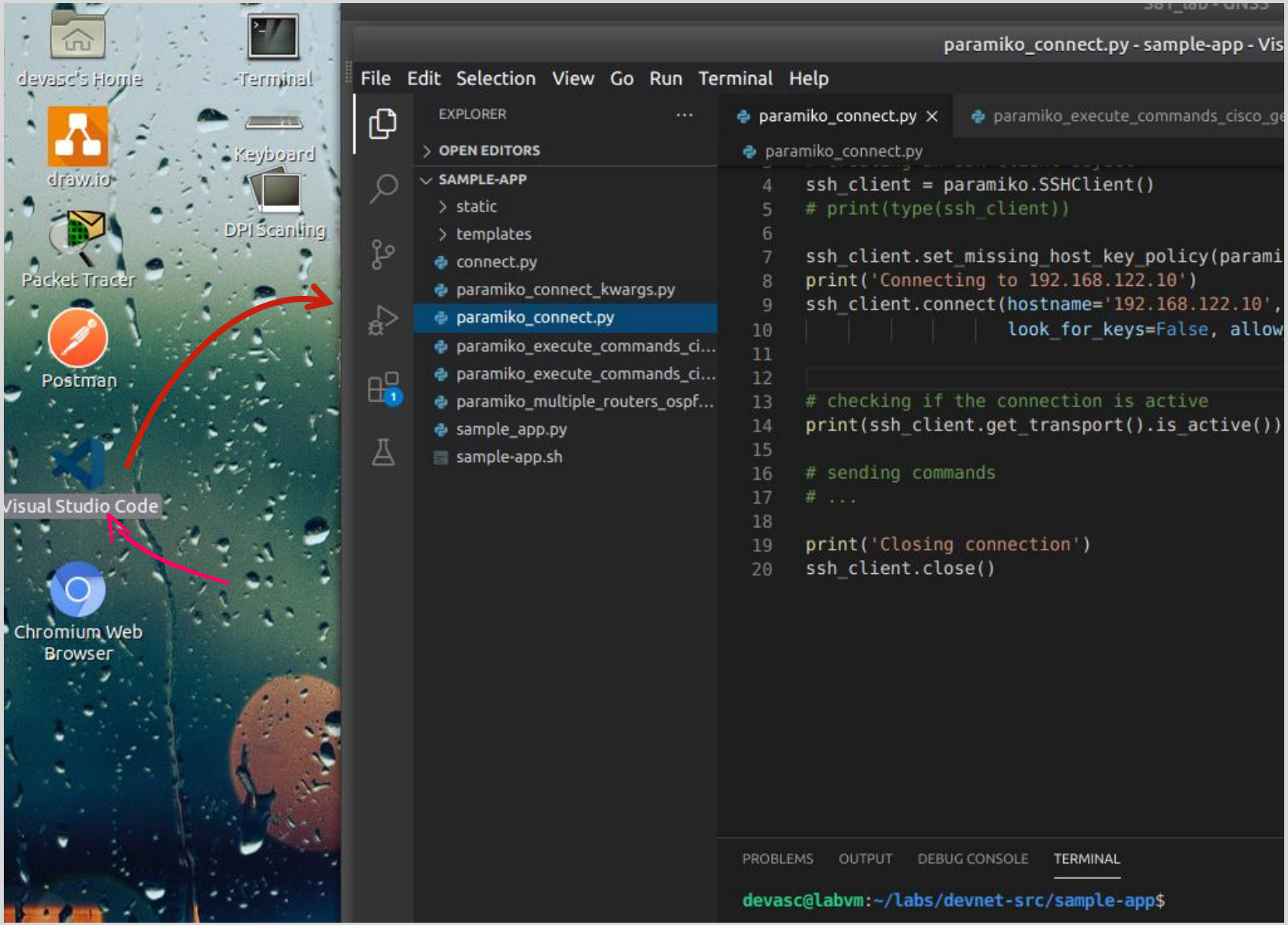
CRYPTO KEY RSA 2048 BITS

IP SSH VERSION 2

ENABLE IN VTY 0 4



LOGIN LOCAL


devasc's Home
Terminal
draw.io
Keyboard
DPI Scanling
Packet Tracer
Postman
Visual Studio Code
Chromium Web Browser
paramiko_connect.py - sample-app - Vis
File Edit Selection View Go Run Terminal Help
EXPLORER
OPEN EDITORS
SAMPLE-APP
static
templates
connect.py
paramiko_connect_kwargs.py
paramiko_connect.py
paramiko_execute_commands_ci...
paramiko_execute_commands_ci...
paramiko_multiple_routers_ospf...
sample_app.py
sample-app.sh
paramiko_connect.py
paramiko_execute_commands_cisco_ge
paramiko_connect.py
4 ssh_client = paramiko.SSHClient()
5 # print(type(ssh_client))
6
7 ssh_client.set_missing_host_key_policy(parami
8 print('Connecting to 192.168.122.10')
9 ssh_client.connect(hostname='192.168.122.10',
10 look_for_keys=False, allow
11
12
13 # checking if the connection is active
14 print(ssh_client.get_transport().is_active())
15
16 # sending commands
17 # ...
18
19 print('Closing connection')
20 ssh_client.close()
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
devasc@labvm:~/labs/devnet-src/sample-app$

Variable

module

object

Att

Methods

```
import paramiko

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

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

```
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)
```

Method

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'}

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

print(f'Connecting to {router["hostname"]}')
print(ssh_client.get_transport().is_active()) # checking if  the connection is active
print('Closing connection')
ssh_client.close()
```

key Value

format

key

```
Connecting to 192.168.122.10
True
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':'cisco', '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)
print(output)

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

for sleep

shell object

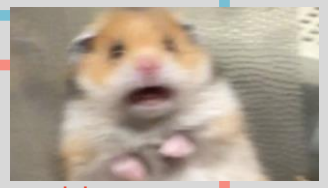
send commands

wait a bit

get output in bytes

close if active

```
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

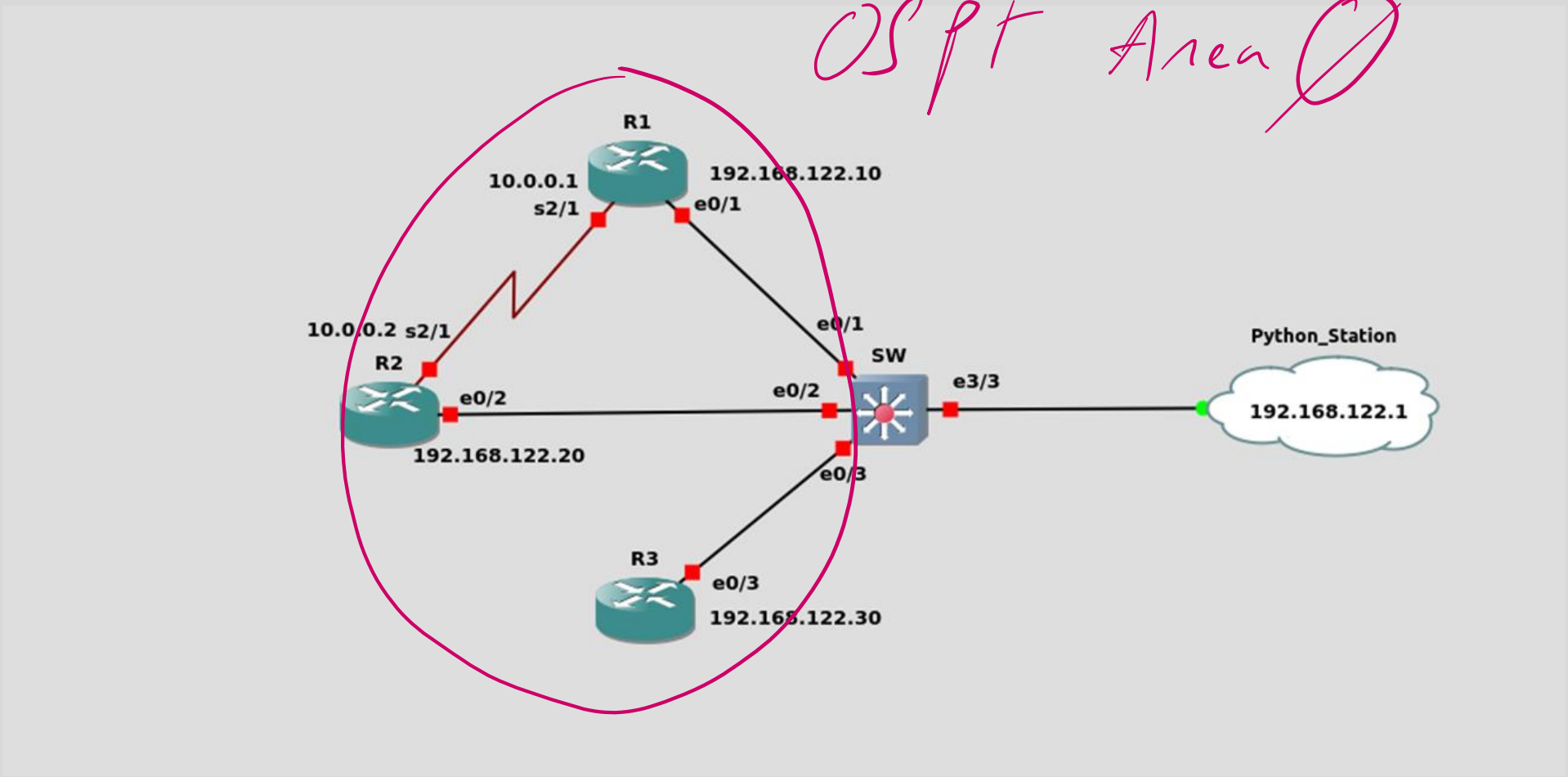
```
.....
Configuration register is 0x0

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

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