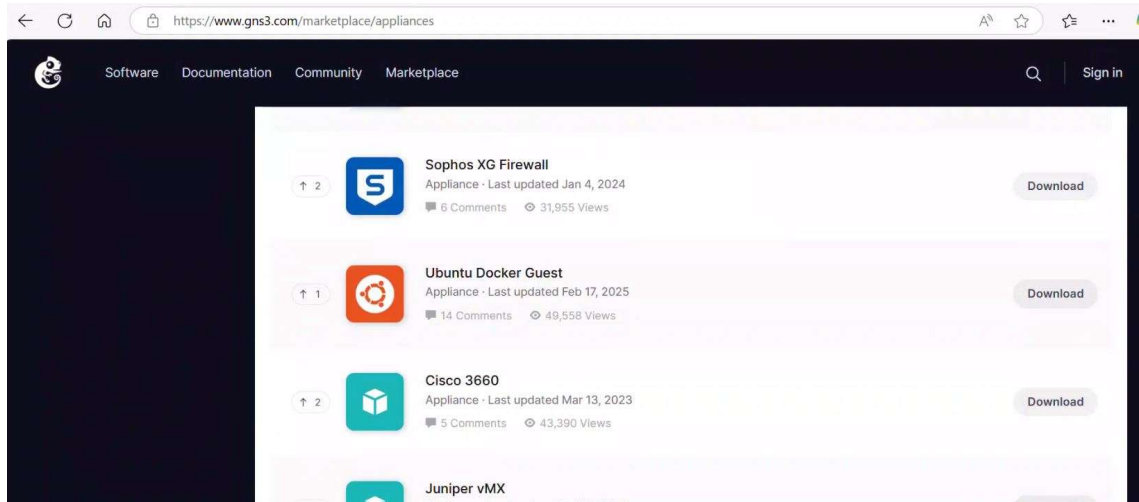


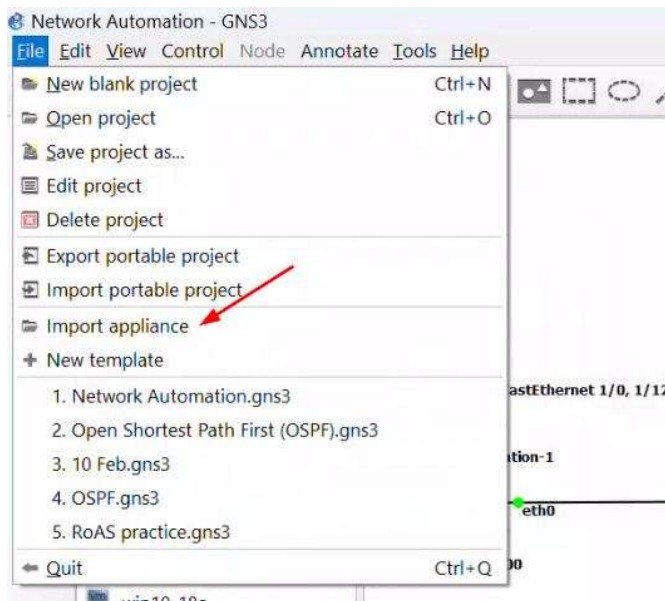
PYTHON TO AUTOMATE CONFIG ON NETWORK DEVICES

1. Download ubuntu docker for GNS3 from GNS3 website.

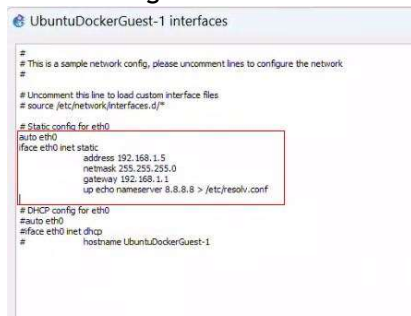
<https://www.gns3.com/marketplace/featured>



2. Add the downloaded ubuntu docker containers.



3. Drag and drop the Ubuntu machine to GNS3 workspace. Right click and select configure. Under General Settings, edit the Network Configuration and enable the static configuration of IP address, and assign appropriate IP settings.



4. Check connectivity.

```
root@UbuntuDockerGuest-1:~# ping google.com
PING google.com (142.250.69.78) 56(84) bytes of data.
64 bytes from tzyula-aa-in-f14.1e100.net (142.250.69.78): icmp_seq=1 ttl=115 time=42.2 ms
64 bytes from tzyula-aa-in-f14.1e100.net (142.250.69.78): icmp_seq=2 ttl=115 time=43.0 ms
^C
--- google.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1003ms
rtt min/avg/max/mdev = 42.153/42.594/43.035/0.441 ms
root@UbuntuDockerGuest-1:~#
```

5. Update the system.

```
root@UbuntuDockerGuest-1:~# apt-get update
Get:1 http://archive.ubuntu.com/ubuntu noble InRelease [256 kB]
Get:2 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:3 http://archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:4 http://archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:5 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Packages [916 kB]
Get:6 http://archive.ubuntu.com/ubuntu noble/universe amd64 Packages [19.3 MB]
Get:7 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [1057 kB]
```

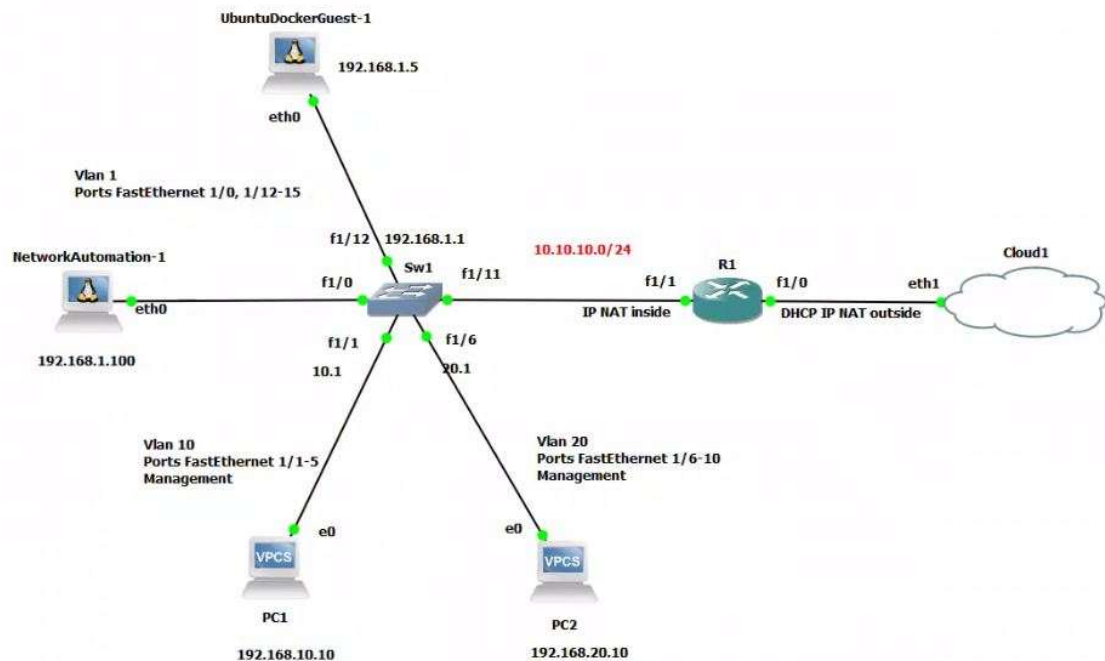
6. Install python.

```
root@UbuntuDockerGuest-1:~# apt-get install python-is-python3
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  ca-certificates libexpat1 libpython3-stdlib libpython3.12-minimal
  libpython3.12-stdlib libreadline8t64 libsqlite3-0 libssl3t64 media-types
  openssl python3 python3-minimal python3.12 python3.12-minimal
  readline-common tzdata
Suggested packages:
  python3-doc python3-tk python3-venv python3.12-venv python3.12-doc binutils
  binfmt-support readline-doc
The following NEW packages will be installed:
  ca-certificates libexpat1 libpython3-stdlib libpython3.12-minimal
  libpython3.12-stdlib libreadline8t64 libsqlite3-0 media-types openssl
  python-is-python3 python3 python3-minimal python3.12 python3.12-minimal
  readline-common tzdata
```

7. Verify python version by typing **python** and **quit()** to exit python.

```
root@UbuntuDockerGuest-1:~# python
Python 3.12.3 (main, Feb 4 2025, 14:48:35) [GCC 13.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> quit()
```

Configure loopback interfaces on R1



1. Configure login and telnet remote access for R1.

```
R1(config)#hostname R1
R1(config)#enable password cisco
R1(config)#username cisco password cisco
R1(config)#line vty 0 4
R1(config-line)#login local
R1(config-line)#transport input all
R1(config-line)#exit
```

2. Before running the script, verify that you have telnet access to router.

```
root@UbuntuDockerGuest-1:~# telnet 10.10.10.1
Trying 10.10.10.1...
Connected to 10.10.10.1.
Escape character is '^]'.

User Access Verification

Username: cisco
Password:
R1>exit
Connection closed by foreign host.
```

3. Create a file to input our script. We will name it pythonR1script1.
4. Input the following script and provide the needed information to access and configure the router:

```
import getpass
import telnetlib
```

```
HOST = "10.10.10.1"
user = input("Enter your telnet username: ")
password = getpass.getpass()
```

```
tn = telnetlib.Telnet(HOST)
```

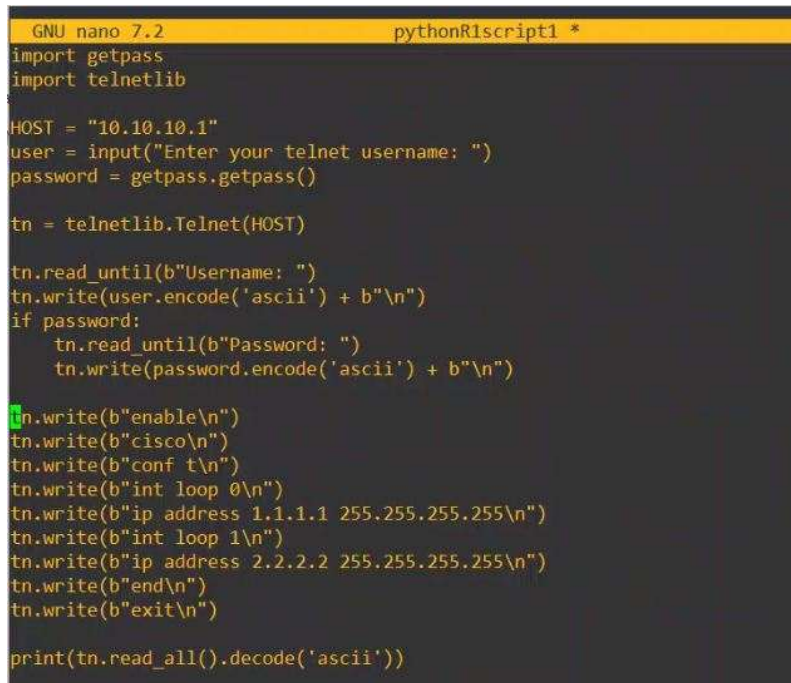
```

tn.read_until(b"Username: ")
tn.write(user.encode('ascii') + b"\n")
if password:
    tn.read_until(b"Password: ")
    tn.write(password.encode('ascii') + b"\n")

tn.write(b"enable\n")
tn.write(b"cisco\n")
tn.write(b"conf t\n")
tn.write(b"int loop 0\n")
tn.write(b"ip address 1.1.1.1 255.255.255.255\n")
tn.write(b"int loop 1\n")
tn.write(b"ip address 2.2.2.2 255.255.255.255\n")
tn.write(b"end\n")
tn.write(b"exit\n")

print(tn.read_all().decode('ascii'))

```



```

GNU nano 7.2 pythonR1script1 *
import getpass
import telnetlib

HOST = "10.10.10.1"
user = input("Enter your telnet username: ")
password = getpass.getpass()

tn = telnetlib.Telnet(HOST)

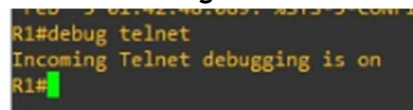
tn.read_until(b"Username: ")
tn.write(user.encode('ascii') + b"\n")
if password:
    tn.read_until(b"Password: ")
    tn.write(password.encode('ascii') + b"\n")

tn.write(b"enable\n")
tn.write(b"cisco\n")
tn.write(b"conf t\n")
tn.write(b"int loop 0\n")
tn.write(b"ip address 1.1.1.1 255.255.255.255\n")
tn.write(b"int loop 1\n")
tn.write(b"ip address 2.2.2.2 255.255.255.255\n")
tn.write(b"end\n")
tn.write(b"exit\n")

print(tn.read_all().decode('ascii'))

```

5. Save the file and exit.
6. If you want to see live the telnet access of the script to the router, use the command **debug telnet**.



```

R1#debug telnet
Incoming Telnet debugging is on
R1#

```

7. Run your script in ubuntu. It will prompt for username and password for R1.

```
root@UbuntuDockerGuest-1:~# python pythonR1script1
/root/pythonR1script1:2: DeprecationWarning: 'telnetlib' is deprecated and slated for removal in Python 3.13
  import telnetlib
Enter your telnet username: cisco
Password:
```

8. Verify the creation of loopback interfaces on R1.

```
Serial2/0/0/0 unassigned YES NVRAM administratively down down
Loopback0 1.1.1.1 YES manual up up
Loopback1 2.2.2.2 YES manual up up
R1#
```


Configure VLANs on a switch

1. Configure login and telnet remote access for the switch.

```
Sw1(config)#hostname Sw1
Sw1(config)#enable password cisco
Sw1(config)#username cisco password cisco
Sw1(config)#line vty 0 4
Sw1(config-line)#login local
Sw1(config-line)#transport input all
Sw1(config-line)#exit
```

2. Before running the script, verify that you have telnet access to router.

```
Sw1#telnet 192.168.1.1
Trying 192.168.1.1 ... Open

User Access Verification

Username: cisco
Password:
Sw1>exit

[Connection to 192.168.1.1 closed by foreign host]
```

3. Copy the previous script pythonR1script1 as pythonSw1script1 and modify it.

```
root@UbuntuDockerGuest-1:~# cp pythonR1script1 pythonSw1script1
root@UbuntuDockerGuest-1:~# nano pythonSw1script1
```

```
GNU nano 7.2 pythonSw1script1 *
import getpass
import telnetlib

HOST = "192.168.1.1"
user = input("Enter your telnet username: ")
password = getpass.getpass()

tn = telnetlib.Telnet(HOST)

tn.read_until(b"Username: ")
tn.write(user.encode('ascii') + b"\n")
if password:
    tn.read_until(b"Password: ")
    tn.write(password.encode('ascii') + b"\n")

tn.write(b"enable\n")
tn.write(b"cisco\n")
tn.write(b"vlan dat\n")
tn.write(b"vlan 2\n")
tn.write(b"vlan 2 name PYTHON- VLAN-2\n")
tn.write(b"vlan 3\n")
tn.write(b"vlan 3 name PYTHON-VLAN-3\n")
tn.write(b"vlan 4\n")
tn.write(b"vlan 4 name PYTHON-VLAN-4\n")
tn.write(b"vlan 5\n")
tn.write(b"vlan 5 name PYTHON-VLAN-5\n")
tn.write(b"exit\n")
tn.write(b"exit\n")

print(tn.read_all().decode('ascii'))
```

4. Save the script and exit. Then verify the existing VLANs on the switch.

```
Sw1#sh vlan-switch
```

VLAN	Name	Status	Ports
1	default	active	Fa1/0, Fa1/12, Fa1/13, Fa1/14 Fa1/15
10	VLAN0010	active	Fa1/1, Fa1/2, Fa1/3, Fa1/4 Fa1/5
20	VLAN0020	active	Fa1/6, Fa1/7, Fa1/8, Fa1/9 Fa1/10
1002	fddi-default	active	
1003	token-ring-default	active	
1004	fddinet-default	active	
1005	trnet-default	active	

5. Run the script.

```
root@UbuntuDockerGuest-1:~# python pythonSw1script1
/root/pythonSw1script1:2: DeprecationWarning: 'telnetlib' is deprecated and slated for removal in Python 3.13
import telnetlib
Enter your telnet username: cisco
Password:

Sw1>enable
Password:
Sw1#vlan dat
Sw1(vlan)#vlan 2
VLAN 2 added:
  Name: VLAN0002
Sw1(vlan)#vlan 2 name PYTHON- ^
% Invalid input detected at '^' marker.

Sw1(vlan)#vlan 3
VLAN 3 added:
  Name: VLAN0003
Sw1(vlan)#vlan 3 name PYTHON-VLAN-3
VLAN 3 modified:
  Name: PYTHON-VLAN-3
Sw1(vlan)#vlan 4
VLAN 4 added:
  Name: VLAN0004
Sw1(vlan)#vlan 4 name PYTHON-VLAN-4
VLAN 4 modified:
  Name: PYTHON-VLAN-4
Sw1(vlan)#vlan 5
VLAN 5 added:
  Name: VLAN0005
Sw1(vlan)#vlan 5 name PYTHON-VLAN-5
VLAN 5 modified:
  Name: PYTHON-VLAN-5
Sw1(vlan)#exit
APPLY completed.
Exiting....
Sw1#exit
```

6. Verify the creation of VLANs on the switch.

```
Sw1#sh vlan-switch
```

VLAN	Name	Status	Ports
1	default	active	Fa1/0, Fa1/12, Fa1/13, Fa1/14 Fa1/15
2	VLAN0002	active	
3	PYTHON-VLAN-3	active	
4	PYTHON-VLAN-4	active	
5	PYTHON-VLAN-5	active	
10	VLAN0010	active	Fa1/1, Fa1/2, Fa1/3, Fa1/4 Fa1/5
20	VLAN0020	active	Fa1/6, Fa1/7, Fa1/8, Fa1/9 Fa1/10
1002	fddi-default	active	

Create and run an executable file

1. Copy the previous script pythonSw1script1 as Sw1script2.py and modify it.

```
root@UbuntuDockerGuest-1:~# cp pythonSw1script1 Sw1script2.py
root@UbuntuDockerGuest-1:~# nano Sw1script2.py
```

2. At the beginning of the script, add a reference to the interpreter to be used.

```
GNU nano 7.2 Sw1script2.py *
#!/usr/bin/env python3

import getpass
import telnetlib

HOST = "192.168.1.1"
user = input("Enter your telnet username: ")
password = getpass.getpass()
```

3. Edit the script to add another VLAN (VLAN 6) to the switch. Then save and exit.

```
tn.write(b"vlan dat\n")
tn.write(b"vlan 2\n")
tn.write(b"vlan 2 name PYTHON-VLAN-2\n")
tn.write(b"vlan 3\n")
tn.write(b"vlan 3 name PYTHON-VLAN-3\n")
tn.write(b"vlan 4\n")
tn.write(b"vlan 4 name PYTHON-VLAN-4\n")
tn.write(b"vlan 5\n")
tn.write(b"vlan 5 name PYTHON-VLAN-5\n")
tn.write(b"vlan 6\n")
tn.write(b"vlan 6 name PYTHON-VLAN-6\n")
tn.write(b"exit\n")
tn.write(b"exit\n")
```

4. Change the file permissions to make it executable.

```
root@UbuntuDockerGuest-1:~# ls
Sw1script2.py pythonR1script1 pythonSw1script1
root@UbuntuDockerGuest-1:~# chmod +x ./Sw1script2.py
root@UbuntuDockerGuest-1:~# ls
Sw1script2.py pythonR1script1 pythonSw1script1
```

5. Verify the existing VLANs on the switch.

```
Sw1#sh vlan-switch
```

VLAN	Name	Status	Ports
1	default	active	Fa1/0, Fa1/12, Fa1/13, Fa1/14 Fa1/15
2	VLAN0002	active	
3	PYTHON-VLAN-3	active	
4	PYTHON-VLAN-4	active	
5	PYTHON-VLAN-5	active	
10	VLAN0010	active	Fa1/1, Fa1/2, Fa1/3, Fa1/4 Fa1/5
20	VLAN0020	active	Fa1/6, Fa1/7, Fa1/8, Fa1/9 Fa1/10
1002	fdi-default	active	

6. Run the script.

```
root@UbuntuDockerGuest-1:~# ./Sw1script2.py
/root/./Sw1script2.py:4: DeprecationWarning: 'telnetlib' is deprecated and slated for removal in Python 3.13
  import telnetlib
Enter your telnet username: cisco
Password:

Sw1>enable
Password:
Sw1#vlan dat
Sw1(vlan)#vlan 2
VLAN 2 modified:
Sw1(vlan)#vlan 2 name PYTHON-VLAN-2
VLAN 2 modified:
```

7. Verify the creation of new VLAN on the switch.

```
Sw1#sh vlan-switch
```

VLAN	Name	Status	Ports
1	default	active	Fa1/0, Fa1/12, Fa1/13, Fa1/14 Fa1/15
2	PYTHON-VLAN-2	active	
3	PYTHON-VLAN-3	active	
4	PYTHON-VLAN-4	active	
5	PYTHON-VLAN-5	active	
6	PYTHON-VLAN-6	active	
10	VLAN0010	active	Fa1/1, Fa1/2, Fa1/3, Fa1/4 Fa1/5
20	VLAN0020	active	Fa1/6, Fa1/7, Fa1/8, Fa1/9 Fa1/10

Remove passwords and improve scripts

1. Change the user privileges on the Switch to land directly in privileged-exec mode.

```
Sw1(config)#user cisco privilege 15
```

2. Verify that the user lands directly in privileged-exec mode using telnet.

```
Sw1#telnet 192.168.1.1
Trying 192.168.1.1 ... Open
```

```
User Access Verification
```

```
Username: cisco
```

```
Password:
```

```
Sw1#
```

3. Create a backup file of the previous script Sw1script2.py.

```
root@UbuntuDockerGuest-1:~# cp Sw1script2.py Sw1script2.bk
root@UbuntuDockerGuest-1:~# ls
Sw1script2.bk Sw1script2.py pythonR1script1 pythonSw1script1
```

4. Edit the script Sw1script2.py to remove the lines with enable command and enable password.

```
tn.write(b"enable\n")
tn.write(b"cisco\n")
tn.write(b"vlan dat\n")
```

5. Add another VLAN (VLAN 7) to the script. Then save and exit.

```
tn.write(b"vlan 5 name PYTHON-VLAN-5\n")
tn.write(b"vlan 6\n")
tn.write(b"vlan 6 name PYTHON-VLAN-6\n")
tn.write(b"vlan 7\n")
tn.write(b"vlan 7 name PYTHON-VLAN-7\n")
```

6. Verify the existing VLANs on the switch.

```
Sw1#sh vlan-switch
```

VLAN	Name	Status	Ports
1	default	active	Fa1/0, Fa1/12, Fa1/13, Fa1/14 Fa1/15
2	PYTHON-VLAN-2	active	
3	PYTHON-VLAN-3	active	
4	PYTHON-VLAN-4	active	
5	PYTHON-VLAN-5	active	
6	PYTHON-VLAN-6	active	
10	VLAN0010	active	Fa1/1, Fa1/2, Fa1/3, Fa1/4 Fa1/5
20	VLAN0020	active	Fa1/6, Fa1/7, Fa1/8, Fa1/9 Fa1/10

7. Run the script.

```
root@UbuntuDockerGuest-1:~# ./Sw1script2.py
/root/./Sw1script2.py:4: DeprecationWarning: 'telnetlib' is deprecated and slated for removal in Python 3.13
  import telnetlib
Enter your telnet username: cisco
Password:

Sw1#vlan dat
Sw1(vlan)#vlan 2
VLAN 2 modified:
Sw1(vlan)#vlan 2 name PYTHON-VLAN-2
VLAN 2 modified:
      Name: PYTHON-VLAN-2
Sw1(vlan)#vlan 3
VLAN 3 modified:
```

8. Verify that the new VLAN has been created on the switch.

```
Sw1#sh vlan-switch
```

VLAN	Name	Status	Ports
1	default	active	Fa1/0, Fa1/12, Fa1/13, Fa1/14, Fa1/15
2	PYTHON-VLAN-2	active	
3	PYTHON-VLAN-3	active	
4	PYTHON-VLAN-4	active	
5	PYTHON-VLAN-5	active	
6	PYTHON-VLAN-6	active	
7	PYTHON-VLAN-7	active	
10	VLAN0010	active	Fa1/1, Fa1/2, Fa1/3, Fa1/4, Fa1/5
20	VLAN0020	active	Fa1/6, Fa1/7, Fa1/8, Fa1/9, Fa1/10
1002	fddi-default	active	

Using a loop to create VLANs

1. Edit the Sw1script2.py script used above and replace the lines for creation of VLANs with a FOR loop.

```
GNU nano 7.2                               ./Sw1scr
#!/usr/bin/env python3

import getpass
import telnetlib

HOST = "192.168.1.1"
user = input("Enter your telnet username: ")
password = getpass.getpass()

tn = telnetlib.Telnet(HOST)

tn.read_until(b"Username: ")
tn.write(user.encode('ascii') + b"\n")
if password:
    tn.read_until(b"Password: ")
    tn.write(password.encode('ascii') + b"\n")

tn.write(b"vlan dat\n")

for n in range(2,11):
    lan = str(n)
    tn.write(b"vlan " + str(n).encode() + b"\n")
    tn.write(b"vlan " + str(n).encode() + b" name PYTHON-VLAN-" + str(n).encode() + b"\n")

tn.write(b"exit\n")
tn.write(b"exit\n")

print(tn.read_all().decode('ascii'))
```

2. Verify the existing VLANs on the switch.

```
Sw1#sh vlan-switch
```

VLAN	Name	Status	Ports
1	default	active	Fa1/0, Fa1/12, Fa1/13, Fa1/14, Fa1/15
2	PYTHON-VLAN-2	active	
3	PYTHON-VLAN-3	active	
4	PYTHON-VLAN-4	active	
5	PYTHON-VLAN-5	active	
6	PYTHON-VLAN-6	active	
7	PYTHON-VLAN-7	active	
10	VLAN0010	active	Fa1/1, Fa1/2, Fa1/3, Fa1/4, Fa1/5
20	VLAN0020	active	Fa1/6, Fa1/7, Fa1/8, Fa1/9, Fa1/10
1002	fddi-default	active	

3. Run the script.

```
root@UbuntuDockerGuest-1:~# ./Sw1script2.py
/root/./Sw1script2.py:4: DeprecationWarning: 'telnetlib' is deprecated
  import telnetlib
Enter your telnet username: cisco
Password:

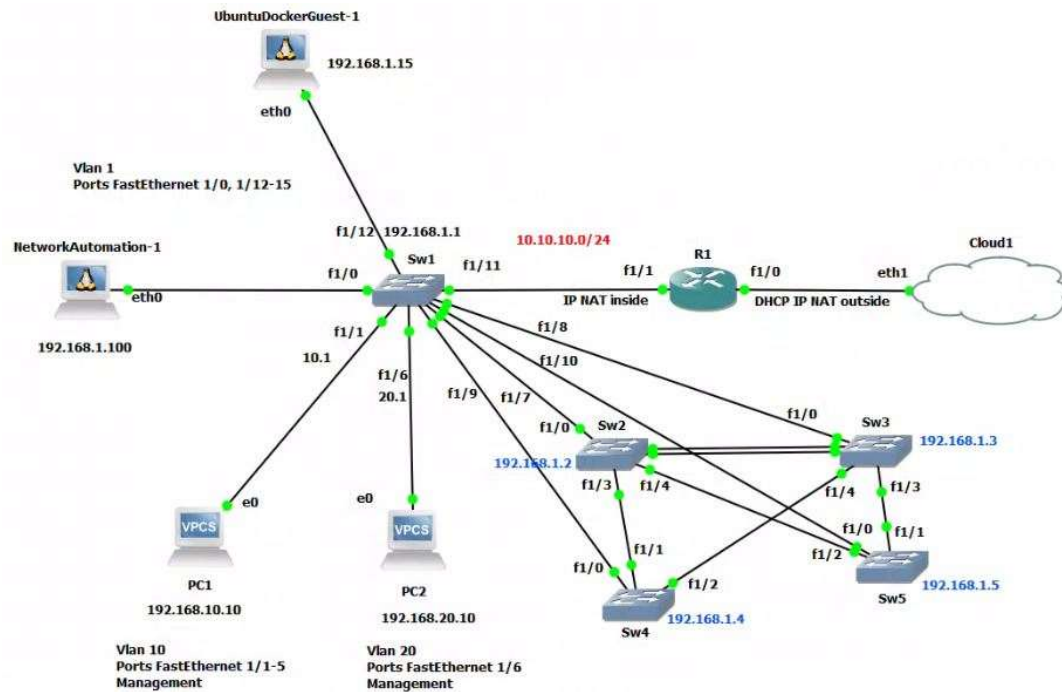
Sw1#vlan dat
Sw1(vlan)#vlan 2
VLAN 2 modified:
Sw1(vlan)#vlan 2 name PYTHON-VLAN-2
VLAN 2 modified:
  Name: PYTHON-VLAN-2
Sw1(vlan)#vlan 3
VLAN 3 modified:
Sw1(vlan)#vlan 3 name PYTHON-VLAN-3
VLAN 3 modified:
  Name: PYTHON-VLAN-3
```

4. Verify that the new VLAN has been created on the switch.

```
Sw1#sh vlan-switch
```

VLAN	Name	Status	Ports
1	default	active	Fa1/0, Fa1/12, Fa1/13, Fa1/14, Fa1/15
2	PYTHON-VLAN-2	active	
3	PYTHON-VLAN-3	active	
4	PYTHON-VLAN-4	active	
5	PYTHON-VLAN-5	active	
6	PYTHON-VLAN-6	active	
7	PYTHON-VLAN-7	active	
8	PYTHON-VLAN-8	active	
9	PYTHON-VLAN-9	active	
10	PYTHON-VLAN-10	active	Fa1/1, Fa1/2, Fa1/3, Fa1/4, Fa1/5
20	VLAN0020	active	Fa1/6, Fa1/7, Fa1/8, Fa1/9, Fa1/10
1002	fddi-default	active	

Using a loop within a loop to create VLANs on multiple switches (OOBM)



1. Add four switches to the topology to be configured using a python script. Perform basic telnet configuration on each of the switches.

```
Sw2#conf t
Enter configuration commands, one per line. End with CNTL/Z.
Sw2(config)#enable password cisco
Sw2(config)#username cisco privilege 15 password 0 cisco
Sw2(config)#line vty 0 4
Sw2(config-line)#login local
Sw2(config-line)#transport input all
Sw2(config-line)#exit
Sw2(config)#spanning-tree vlan 1 root primary
Sw2(config)#int vlan 1
Sw2(config-if)#ip add 192.168.1.2 255.255.255.0
Sw2(config-if)#no shut
```

```
Sw3#conf t
Enter configuration commands, one per line. End with CNTL/Z.
Sw3(config)#enable password cisco
Sw3(config)#username cisco privilege 15 password 0 cisco
Sw3(config)#line vty 0 4
Sw3(config-line)#login local
Sw3(config-line)#transport input all
Sw3(config-line)#exit
Sw3(config)#int vlan 1
Sw3(config-if)#ip add 192.168.1.3 255.255.255.0
Sw3(config-if)#no shut
```

2. Verify telnet access to all the four switches.

```
Sw1#telnet 192.168.1.4
Trying 192.168.1.4 ... Open

User Access Verification

Username: cisco
Password:
Sw4#exit
```

3. Copy the previous script Sw1script2.py as Sw1script3.py and modify it.

```
root@UbuntuDockerGuest-1:~# cp ./Sw1script2.py ./Sw1script3.py
root@UbuntuDockerGuest-1:~# ls
Sw1script2.bk Sw1script2.py Sw1script3.py pythonR1script1 pythonSw1script1
root@UbuntuDockerGuest-1:~# nano ./Sw1script3.py
```

4. Modify the script as below:

```
#!/usr/bin/env python3

import getpass
import telnetlib

user = input("Enter your telnet username: ")
password = getpass.getpass()

for n in range (2,6):
    HOST = "192.168.1." + str(n)

    tn = telnetlib.Telnet(HOST)

    tn.read_until(b"Username: ")
    tn.write(user.encode('ascii') + b"\n")
    if password:
        tn.read_until(b"Password: ")
        tn.write(password.encode('ascii') + b"\n")

    tn.write(b"vlan dat\n")

    for n in range (2,12):
        tn.write(b"vlan " + str(n).encode() + b"\n")
        tn.write(b"vlan " + str(n).encode() + b" name PYTHON-VLAN-" + str(n).encode() + b"\n")

    tn.write(b"exit\n")
    tn.write(b"exit\n")

    print(tn.read_all().decode('ascii'))
```

5. Run the script.

```
root@UbuntuDockerGuest-1:~# ./Sw1script3.py
/root/./Sw1script3.py:4: DeprecationWarning: 'telnetlib' is deprecated and slated
for removal in Python 3.13
  import telnetlib
Enter your telnet username: cisco
Password:

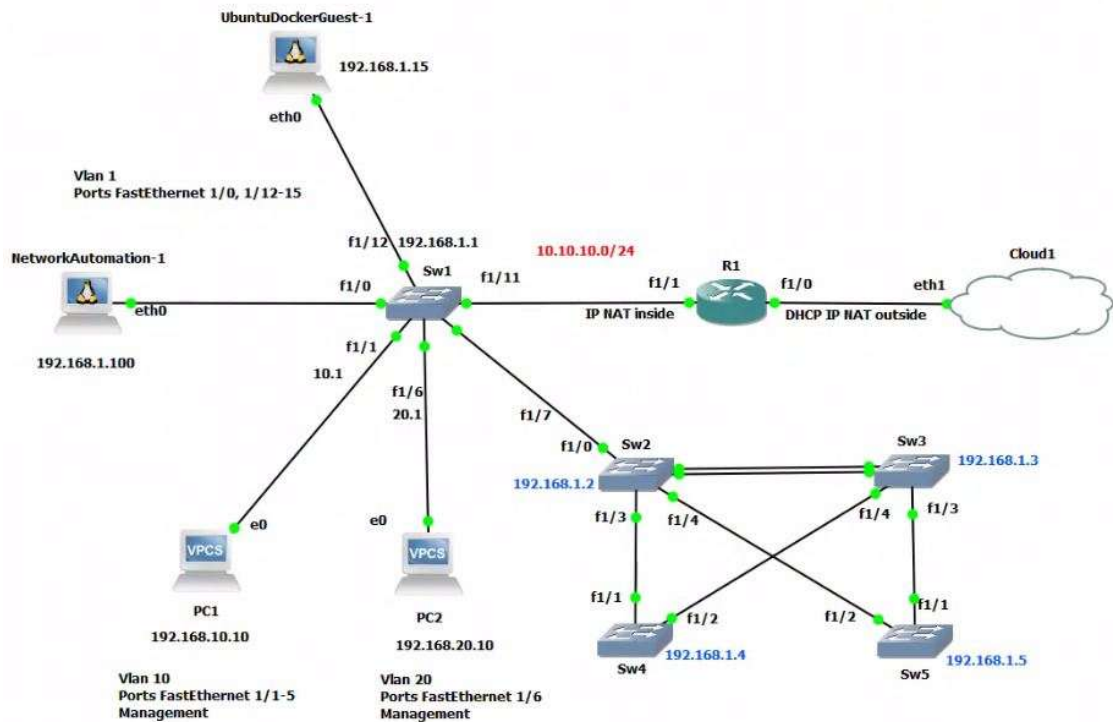
Sw2#vlan dat
Sw2(vlan)#vlan 2
VLAN 2 added:
  Name: VLAN0002
Sw2(vlan)#vlan 2 name PYTHON-VLAN-2
VLAN 2 modified:
  Name: PYTHON-VLAN-2
Sw2(vlan)#vlan 3
VLAN 3 added:
```

6. Verify that the new VLAN has been created on the switches.

```
Sw4#show vlan-switch br
```

VLAN	Name	Status	Ports
1	default	active	Fa1/0, Fa1/1, Fa1/2, Fa1/3 Fa1/4, Fa1/5, Fa1/6, Fa1/7 Fa1/8, Fa1/9, Fa1/10, Fa1/11 Fa1/12, Fa1/13, Fa1/14, Fa1/15
2	PYTHON-VLAN-2	active	
3	PYTHON-VLAN-3	active	
4	PYTHON-VLAN-4	active	
5	PYTHON-VLAN-5	active	
6	PYTHON-VLAN-6	active	
7	PYTHON-VLAN-7	active	
8	PYTHON-VLAN-8	active	
9	PYTHON-VLAN-9	active	
10	PYTHON-VLAN-10	active	
11	PYTHON-VLAN-11	active	
1002	fddi-default	active	
1003	token-ring-default	active	

Create VLANs on multiple switches (In-band Management)



1. The previous script Sw1script3.py could also be used for in-band management and it'll work if there is connectivity to the destination switch. However, here we modified the script to add VLAN 12 and save the configuration to the switch.

```

GNU nano 7.2                                                                    ./Sw1script3.py
#!/usr/bin/env python3

import getpass
import telnetlib

user = input("Enter your telnet username: ")
password = getpass.getpass()

for n in range (2,6):
    HOST = "192.168.1." + str(n)

    tn = telnetlib.Telnet(HOST)

    tn.read_until(b"Username: ")
    tn.write(user.encode('ascii') + b"\n")
    if password:
        tn.read_until(b"Password: ")
        tn.write(password.encode('ascii') + b"\n")

    tn.write(b"vlan dat\n")

    for n in range (2,13):
        tn.write(b"vlan " + str(n).encode() + b"\n")
        tn.write(b"vlan " + str(n).encode() + b" name PYTHON-VLAN-" + str(n).encode() + b"\n")

    tn.write(b"exit\n")
    tn.write(b"copy run start\n")
    tn.write(b"\n")
    tn.write(b"exit\n")

    print(tn.read_all().decode('ascii'))

```

2. Run the script.

```
root@UbuntuDockerGuest-1:~# ./Sw1script3.py
/root/./Sw1script3.py:4: DeprecationWarning: 'telnetlib' is deprecated and slated
for removal in Python 3.13
  import telnetlib
Enter your telnet username: cisco
Password:

Sw2#vlan dat
Sw2(vlan)#vlan 2
VLAN 2 modified:
Sw2(vlan)#vlan 2 name PYTHON-VLAN-2
VLAN 2 modified:
      Name: PYTHON-VLAN-2
```

3. The configuration messages show that the new vlan has been created and the switch configuration has been saved to NVRAM.

```
Sw5(vlan)#vlan 12
VLAN 12 added:
      Name: VLAN0012
Sw5(vlan)#vlan 12 name PYTHON-VLAN-12
VLAN 12 modified:
      Name: PYTHON-VLAN-12
Sw5(vlan)#exit
APPLY completed.
Exiting...
Sw5#copy run start
Destination filename [startup-config]?
Building configuration...
[OK]
Sw5#exit
```