

Práctica calificada 1 - CC312

Sergio Sebastian Pezo Jimenez - 20224087G

- Práctica calificada 1 - CC312
 - Parte 1: Packet Tracer - Comparamos el CLI y el SDN Controller NetworkManagement
 - Analizamos la topología de red
 - Utilizamos el CLI para recopilar información
 - Configuramos el controlador de PT
 - Configuramos la conectividad para nuestro PT-Controller0
 - Usamos un SDN controller para descubrir una topología
 - Usamos un SDN para recopilar información
 - Usamo un SDN controller para configurar la configuración de red
 - Parte 2: Construimos un CI/CD pipeline usando Jenkins
 - Inicializamos la VM de DEVASC.
 - Commiteamos el sample app a git
 - Modificamos los archivos para pushearlos de nuevo
 - Descargamos y corremos la imagen de Jenkins
 - Configuramos Jenkins
 - Usamos Jenkins para ejecutar un build de nuestra app
 - Usamos Jenkins para testear el Build
 - Creamos un pipeline job
 - Preguntas:
 - Pregunta 1
 - Pregunta 2
 - Pregunta 3
 - Pregunta 4
 - Pregunta 5

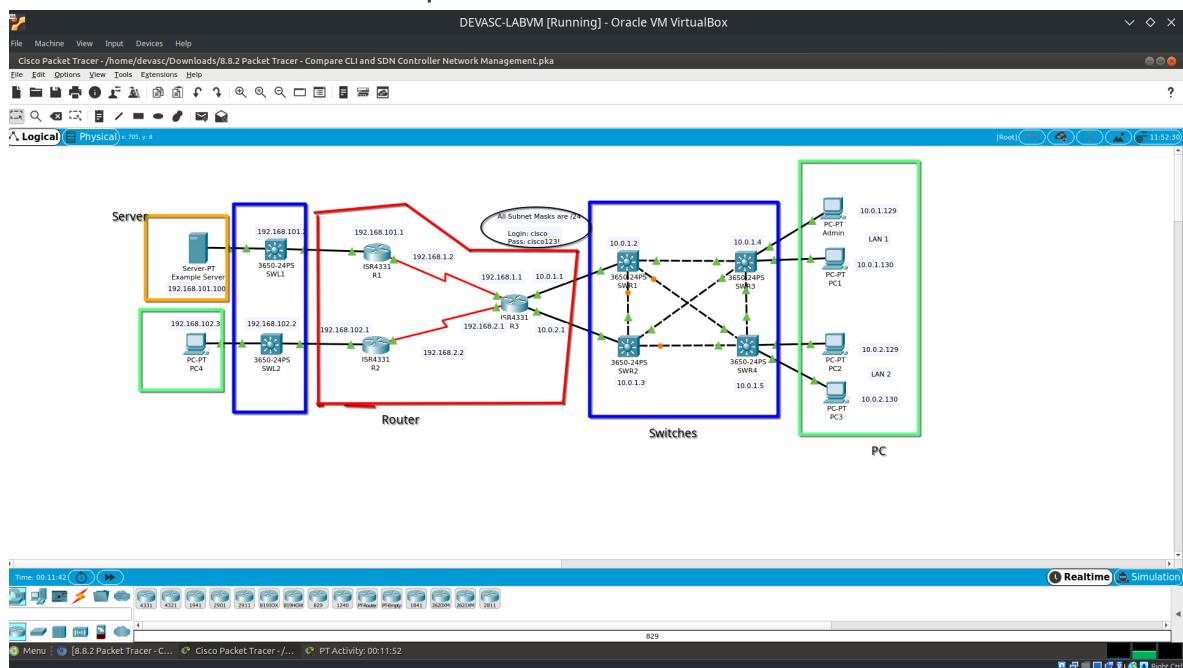
Parte 1: Packet Tracer - Comparamos el CLI y el SDN Controller NetworkManagement

En este apartado vamos a comparar las diferencias entre usar el CLI y un SDN para manejar la red.

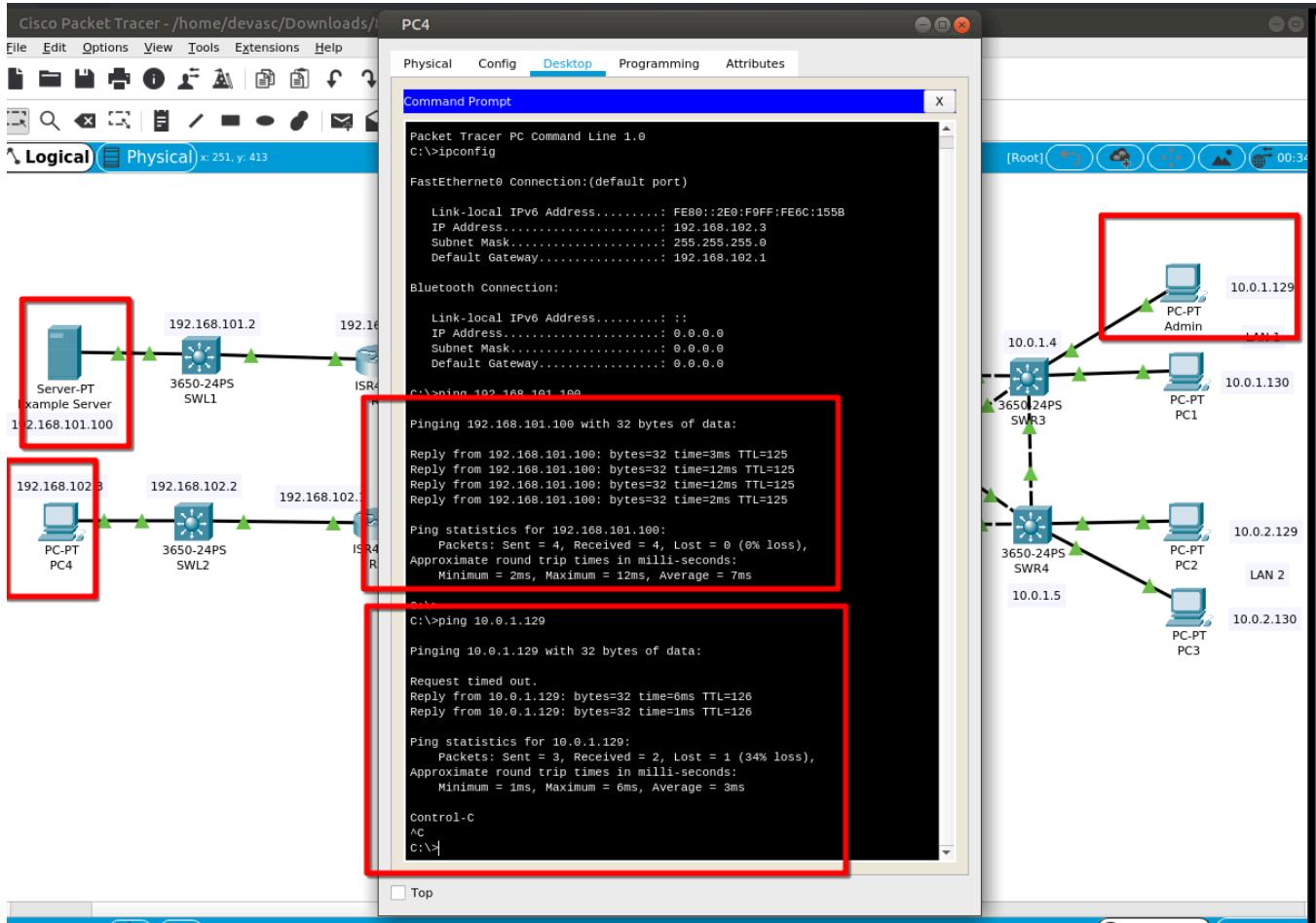
Analizamos la topología de red

La red está configurada de la siguiente manera:

- Los routers están ejecutando OSPFv2.
- SSH está habilitado en todos los dispositivos con el usuario "cisco" y la contraseña "cisco123!"
- R1 no tiene hosts.
- La red LAN de R2 tiene una configuración IPv4 estática.
- R3 es el servidor DHCPv4 para LAN1 y LAN2.
- Los switches son de Capa 2 (sin VLANs).
- Todos los switches SWR# pertenecen a LAN1.

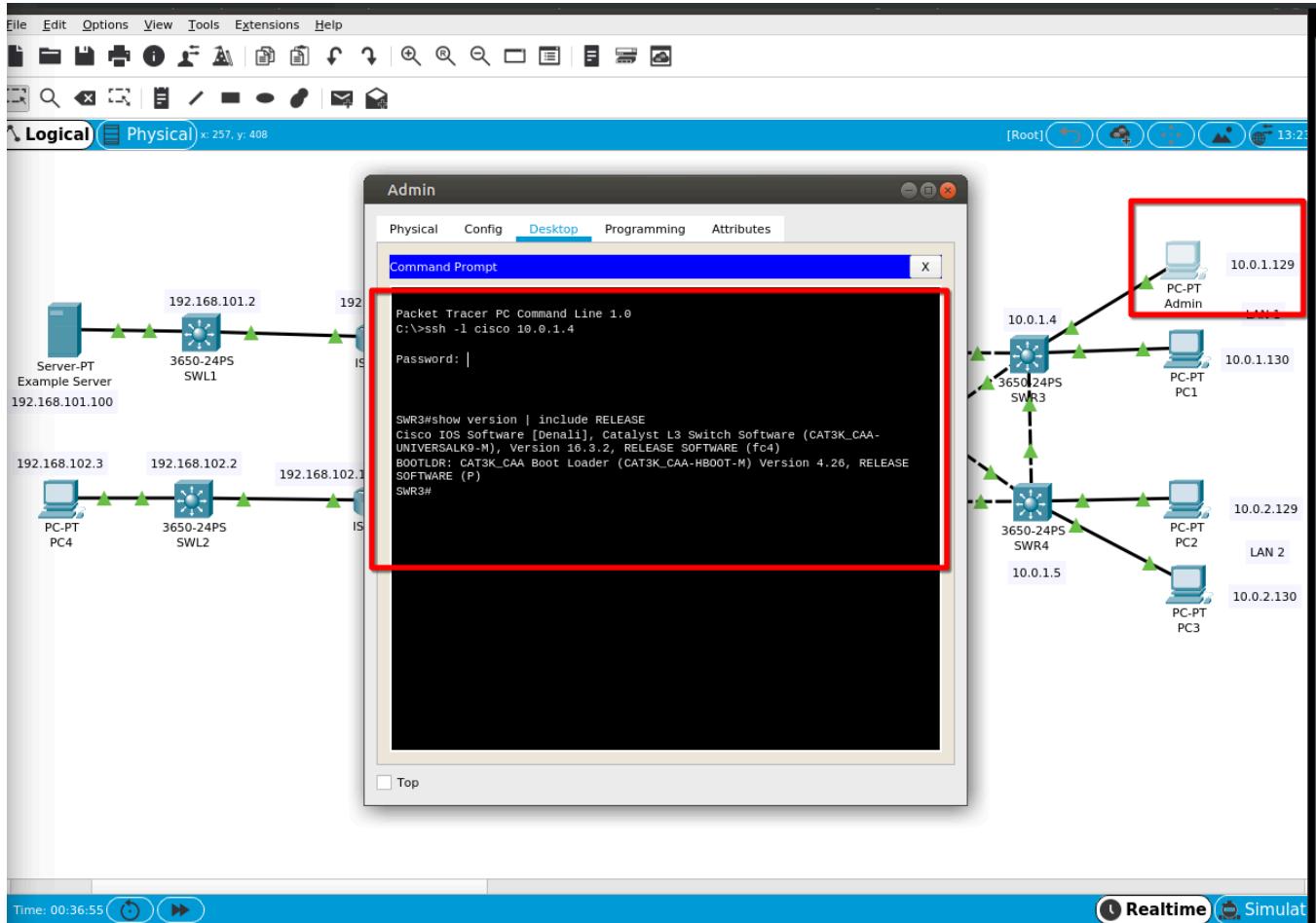


Hacemos ping desde la PC4 con el server con su ip y con la PC-Admin, de igual manera con los demás dispositivos.



Utilizamos el CLI para recopilar información

Desde la PC-Admin, accedemos a las SWR3 switch mediante ssh con nuestro usuario y contraseña mencionados en el inicio.



Con el command Prompt en SWR3, repetiremos el proceso para los dispositivos: SWL1, SWL2, SWR1, SWR2, SWR3, SWR4, R1, R2, y R3.

Empezamos con la primer red con:
SWR1, SWR2, SWR3, SWR4 y R3:

```

Password:01

SWR3#show version | include RELEASE
Cisco IOS Software [Denali], Catalyst L3 Switch Software (CAT3K_CAA-UNIVERSALK9-M), Version 16.3.2, RELEASE SOFTWARE (fc4)
BOOTLDR: CAT3K_CAA Boot Loader (CAT3K_CAA-HBOOT-M) Version 4.26, RELEASE SOFTWARE (P)
SWR3#xd
SWR3#ssh -l cisco 10.0.1.2
Password:

SWR1#show version | include RELEASE
Cisco IOS Software [Denali], Catalyst L3 Switch Software (CAT3K_CAA-UNIVERSALK9-M), Version 16.3.2, RELEASE SOFTWARE (fc4)
BOOTLDR: CAT3K_CAA Boot Loader (CAT3K_CAA-HBOOT-M) Version 4.26, RELEASE SOFTWARE (P)
SWR1#ssh -l cisco 10.0.1.3
Password:

SWR2#show version | include RELEASE
Cisco IOS Software [Denali], Catalyst L3 Switch Software (CAT3K_CAA-UNIVERSALK9-M), Version 16.3.2, RELEASE SOFTWARE (fc4)
BOOTLDR: CAT3K_CAA Boot Loader (CAT3K_CAA-HBOOT-M) Version 4.26, RELEASE SOFTWARE (P)
SWR2#
SWR2#ssh -l cisco 10.0.1.5
Password:

SWR4#ssh -l cisco 10.0.1.1
Password:

R3#
R3#show version | include RELEASE

```

Continuamos con la red de la izquierda:

```

File Edit Options View Tools Extensions Help
Admin
Physical Config Desktop Programming Attributes
Command Prompt
SWR3#show version | include RELEASE
Cisco IOS Software [Denali], Catalyst L3 Switch Software (CAT3K_CAA-UNIVERSALK9-M), Version 16.3.2, RELEASE SOFTWARE (fc4)
BOOTLDR: CAT3K_CAA Boot Loader (CAT3K_CAA-HBOOT-M) Version 4.26, RELEASE SOFTWARE (P)
SWR3#xd
SWR3#ssh -l cisco 10.0.1.2
Password:

SWR1#show version | include RELEASE
Cisco IOS Software [Denali], Catalyst L3 Switch Software (CAT3K_CAA-UNIVERSALK9-M), Version 16.3.2, RELEASE SOFTWARE (fc4)
BOOTLDR: CAT3K_CAA Boot Loader (CAT3K_CAA-HBOOT-M) Version 4.26, RELEASE SOFTWARE (P)
SWR1#ssh -l cisco 10.0.1.3
Password:

SWR2#show version | include RELEASE
Cisco IOS Software [Denali], Catalyst L3 Switch Software (CAT3K_CAA-UNIVERSALK9-M), Version 16.3.2, RELEASE SOFTWARE (fc4)
BOOTLDR: CAT3K_CAA Boot Loader (CAT3K_CAA-HBOOT-M) Version 4.26, RELEASE SOFTWARE (P)
SWR2#
SWR2#ssh -l cisco 10.0.1.5
Password:

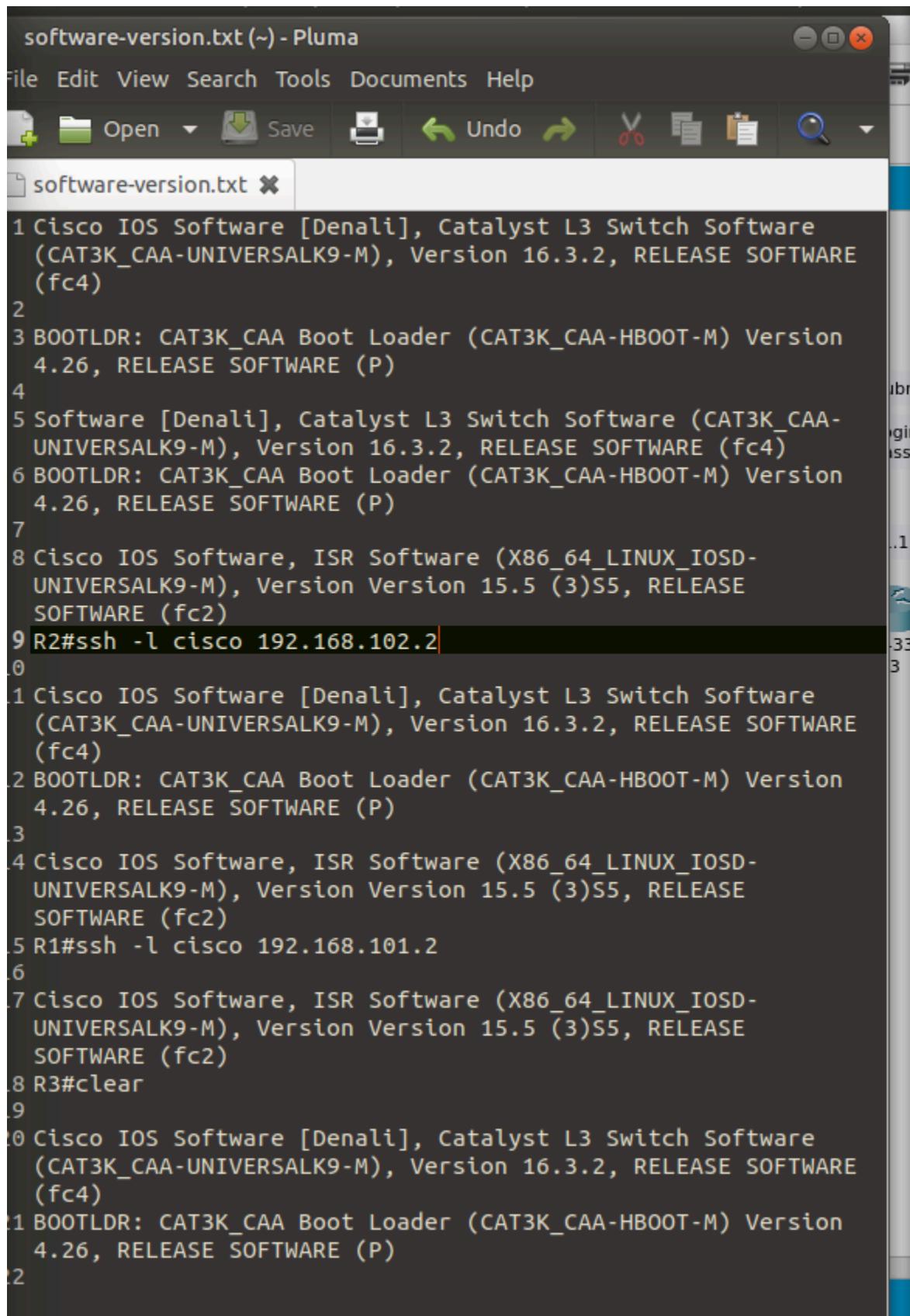
SWR4#ssh -l cisco 10.0.1.1
Password:

R3#
R3#show version | include RELEASE
Cisco IOS Software, ISR Software, X86_64_LINUX_IOSD-UNIVERSALK9-M,
Version Version 15.5 (3)S5, RELEASE SOFTWARE (fc2)
R3#clear
% Incomplete command.
R3#ssh -l cisco 192.168.101.1
Password:
% Login invalid

```

Time: 00:55:20

Gurdamos todos las versiones en un archivo `software-version.txt`



The screenshot shows a window titled "software-version.txt (~) - Pluma". The menu bar includes File, Edit, View, Search, Tools, Documents, and Help. The toolbar contains icons for Open, Save, Undo, Redo, Cut, Copy, Paste, and Find. The main text area displays a list of software versions from different Cisco devices:

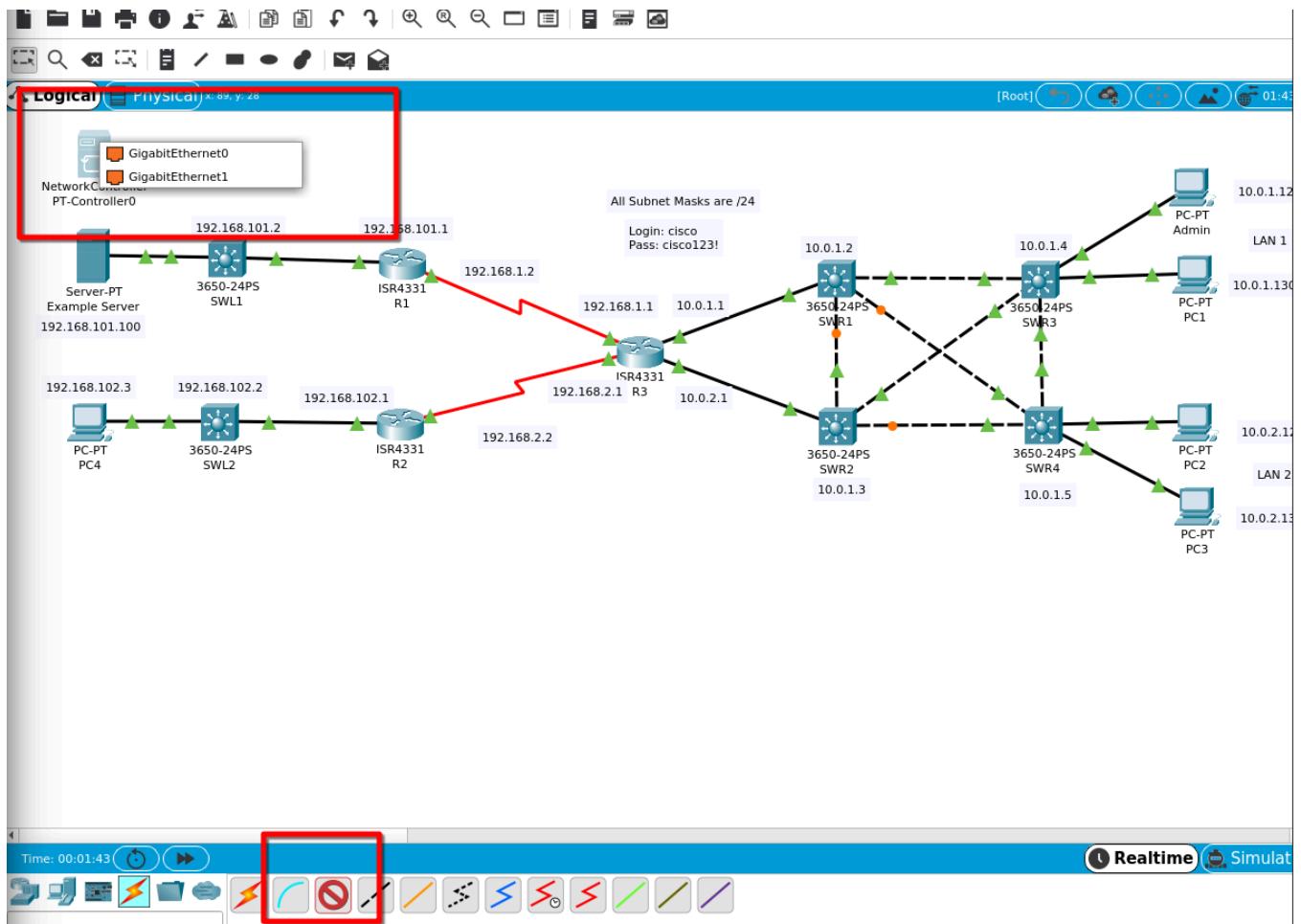
```
1 Cisco IOS Software [Denali], Catalyst L3 Switch Software  
  (CAT3K_CAA-UNIVERSALK9-M), Version 16.3.2, RELEASE SOFTWARE  
  (fc4)  
2  
3 BOOTLDR: CAT3K_CAA Boot Loader (CAT3K_CAA-HBOOT-M) Version  
  4.26, RELEASE SOFTWARE (P)  
4  
5 Software [Denali], Catalyst L3 Switch Software (CAT3K_CAA-  
  UNIVERSALK9-M), Version 16.3.2, RELEASE SOFTWARE (fc4)  
6 BOOTLDR: CAT3K_CAA Boot Loader (CAT3K_CAA-HBOOT-M) Version  
  4.26, RELEASE SOFTWARE (P)  
7  
8 Cisco IOS Software, ISR Software (X86_64_LINUX_IOSD-  
  UNIVERSALK9-M), Version Version 15.5 (3)S5, RELEASE  
  SOFTWARE (fc2)  
9 R2#ssh -l cisco 192.168.102.2  
0  
1 Cisco IOS Software [Denali], Catalyst L3 Switch Software  
  (CAT3K_CAA-UNIVERSALK9-M), Version 16.3.2, RELEASE SOFTWARE  
  (fc4)  
2 BOOTLDR: CAT3K_CAA Boot Loader (CAT3K_CAA-HBOOT-M) Version  
  4.26, RELEASE SOFTWARE (P)  
3  
4 Cisco IOS Software, ISR Software (X86_64_LINUX_IOSD-  
  UNIVERSALK9-M), Version Version 15.5 (3)S5, RELEASE  
  SOFTWARE (fc2)  
5 R1#ssh -l cisco 192.168.101.2  
6  
7 Cisco IOS Software, ISR Software (X86_64_LINUX_IOSD-  
  UNIVERSALK9-M), Version Version 15.5 (3)S5, RELEASE  
  SOFTWARE (fc2)  
8 R3#clear  
9  
0 Cisco IOS Software [Denali], Catalyst L3 Switch Software  
  (CAT3K_CAA-UNIVERSALK9-M), Version 16.3.2, RELEASE SOFTWARE  
  (fc4)  
1 BOOTLDR: CAT3K_CAA Boot Loader (CAT3K_CAA-HBOOT-M) Version  
  4.26, RELEASE SOFTWARE (P)  
2
```

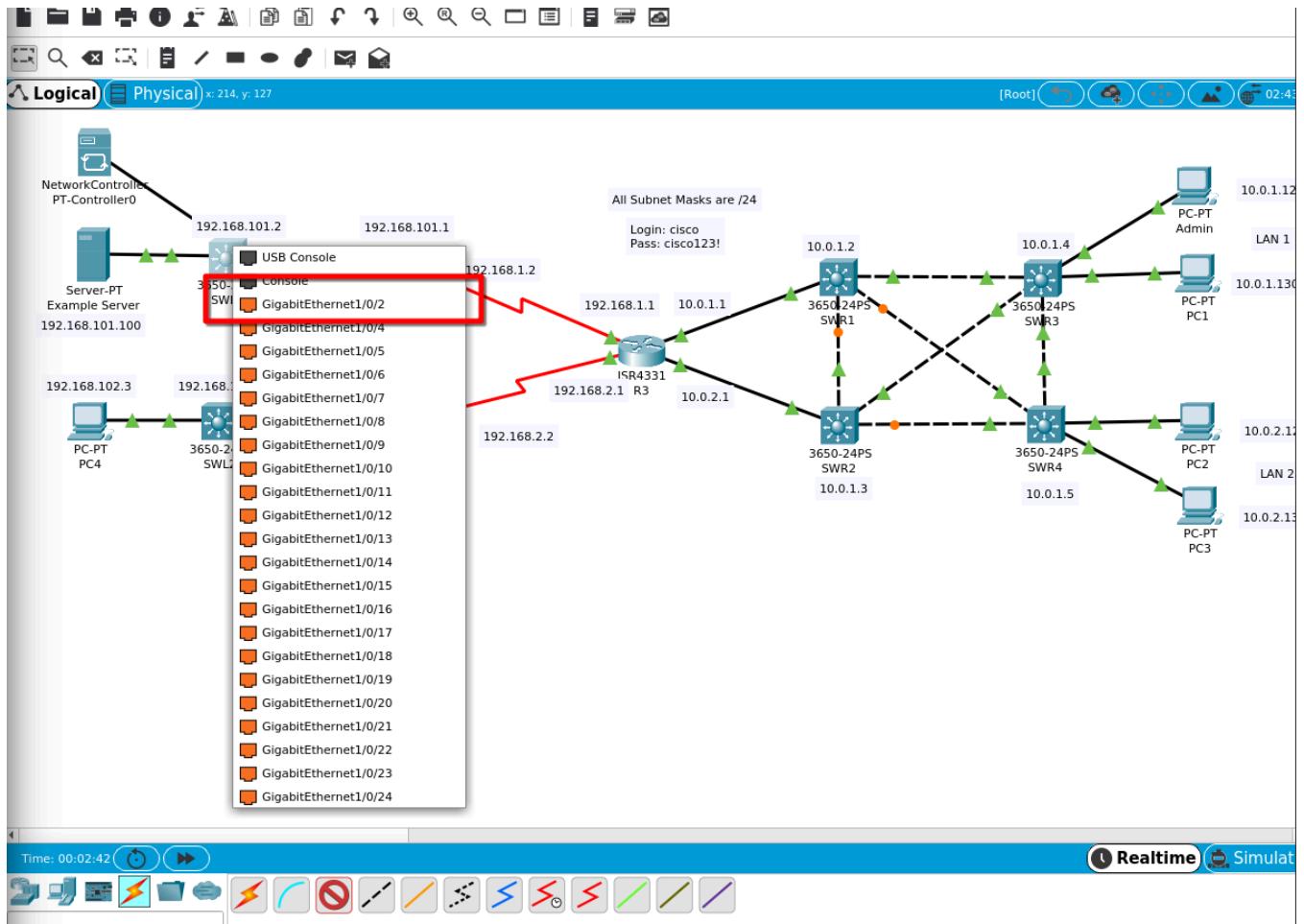
Finalmente salimos de cada máquina para volver a PC-Admin

```
SWL2#exit
[Connection to 192.168.102.2 closed by foreign host]
R2#exit
[Connection to 192.168.102.1 closed by foreign host]
SWL1#exit
[Connection to 192.168.101.2 closed by foreign host]
R1#exit
[Connection to 192.168.101.1 closed by foreign host]
R3#exit
[Connection to 10.0.1.1 closed by foreign host]
SWR4#exit
[Connection to 10.0.1.5 closed by foreign host]
SWR2#exit
[Connection to 10.0.1.3 closed by foreign host]
SWR1#exit
[Connection to 10.0.1.2 closed by foreign host]
SWR3#exit
[Connection to 10.0.1.4 closed by foreign host]
C:\>
```

Configuramos el controlador de PT

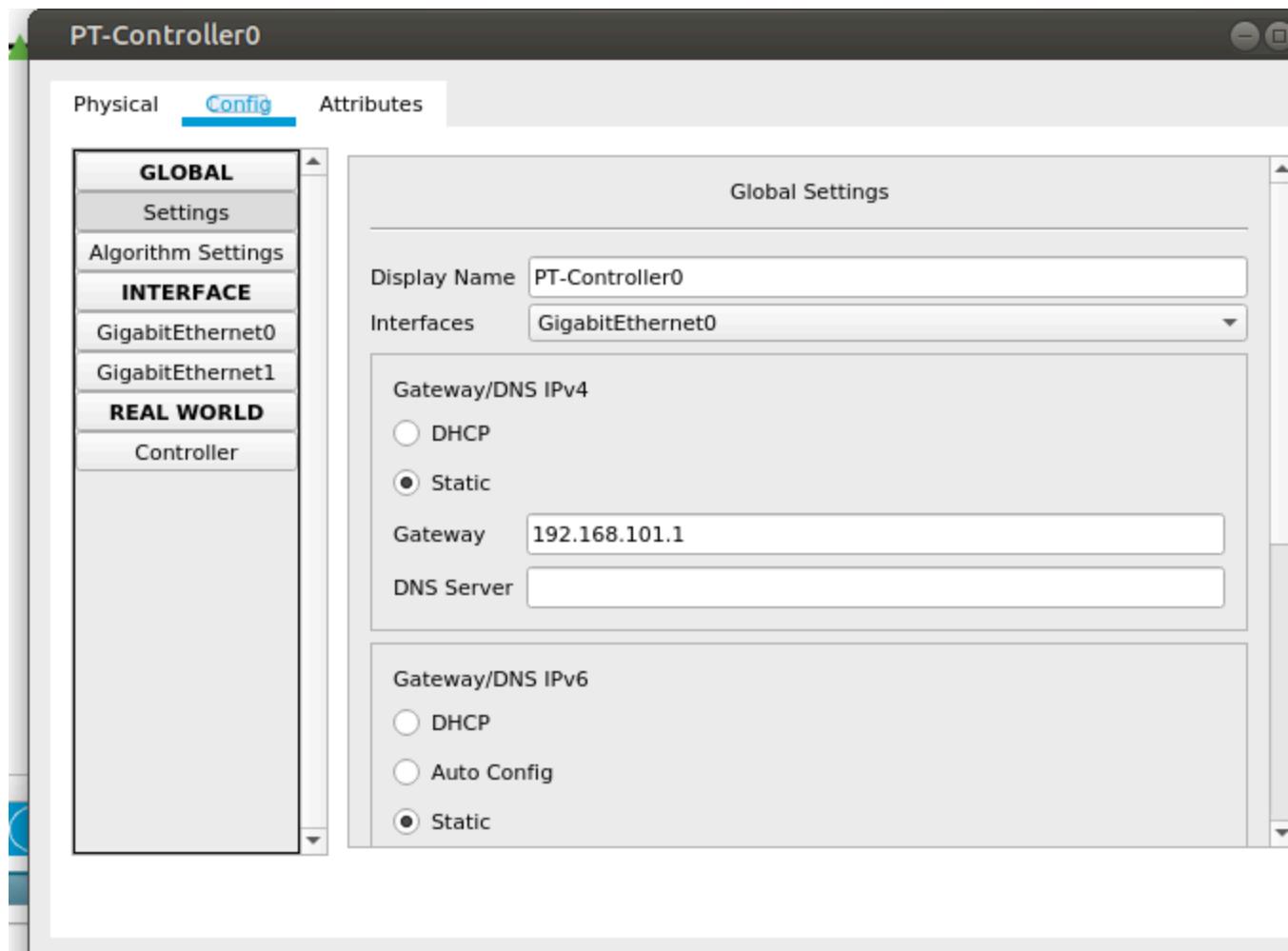
Agregamos un dispositivo NetworkController, y con un cable de cobre lo conectamos a la topología.

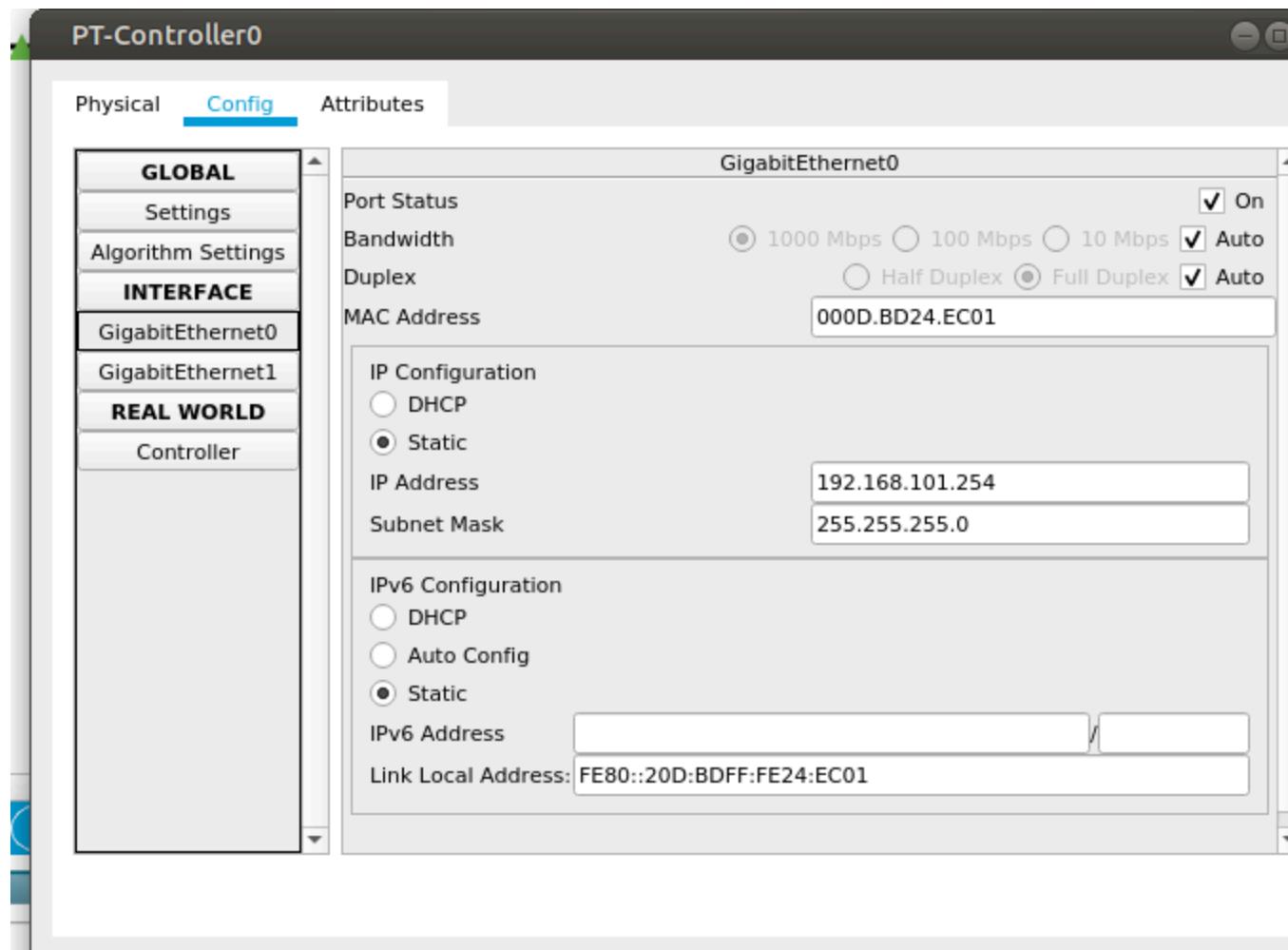




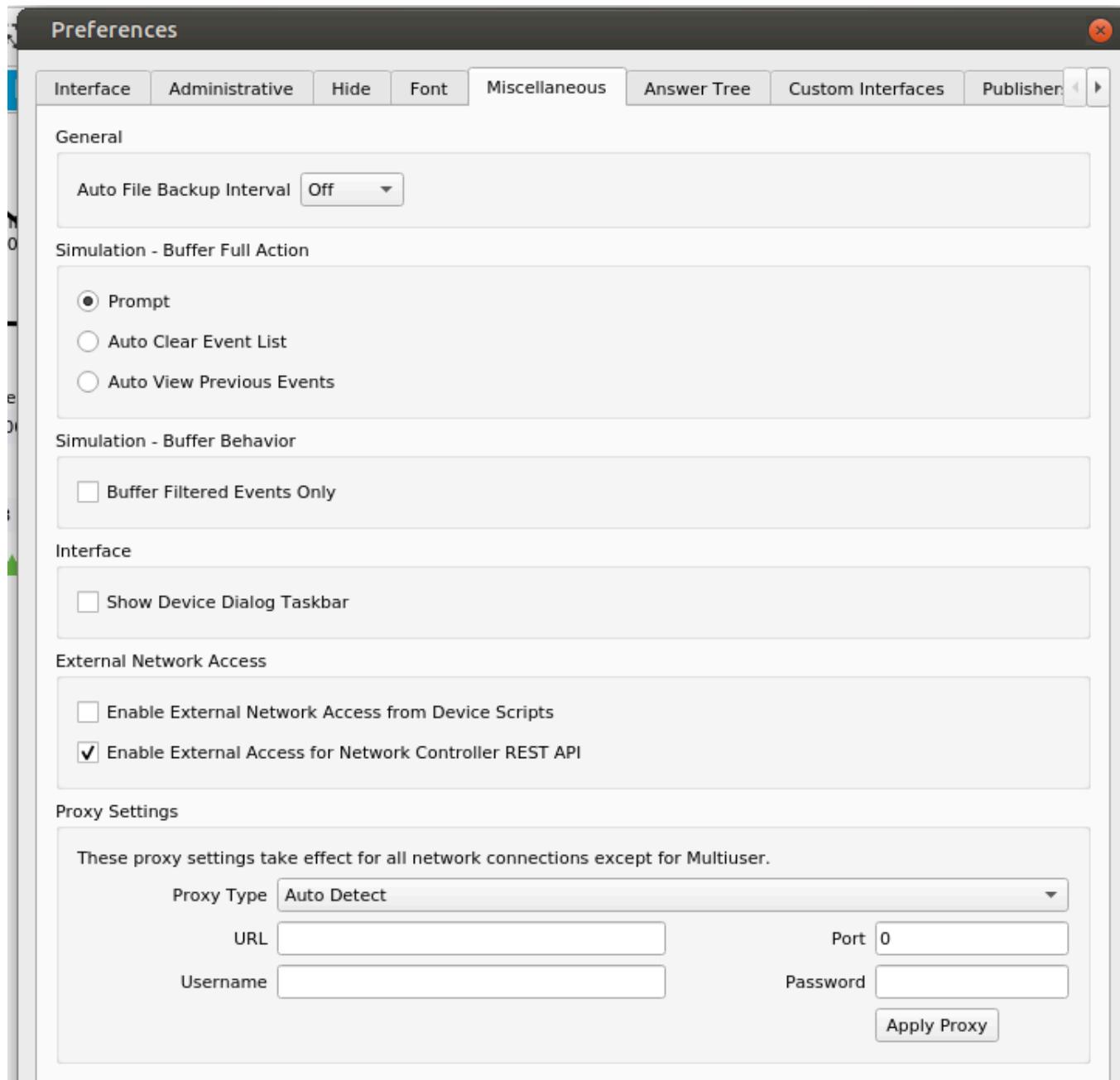
Configuramos la conectividad para nuestro PT-Controller0

Configuramos el gateway, para usar el router R1:

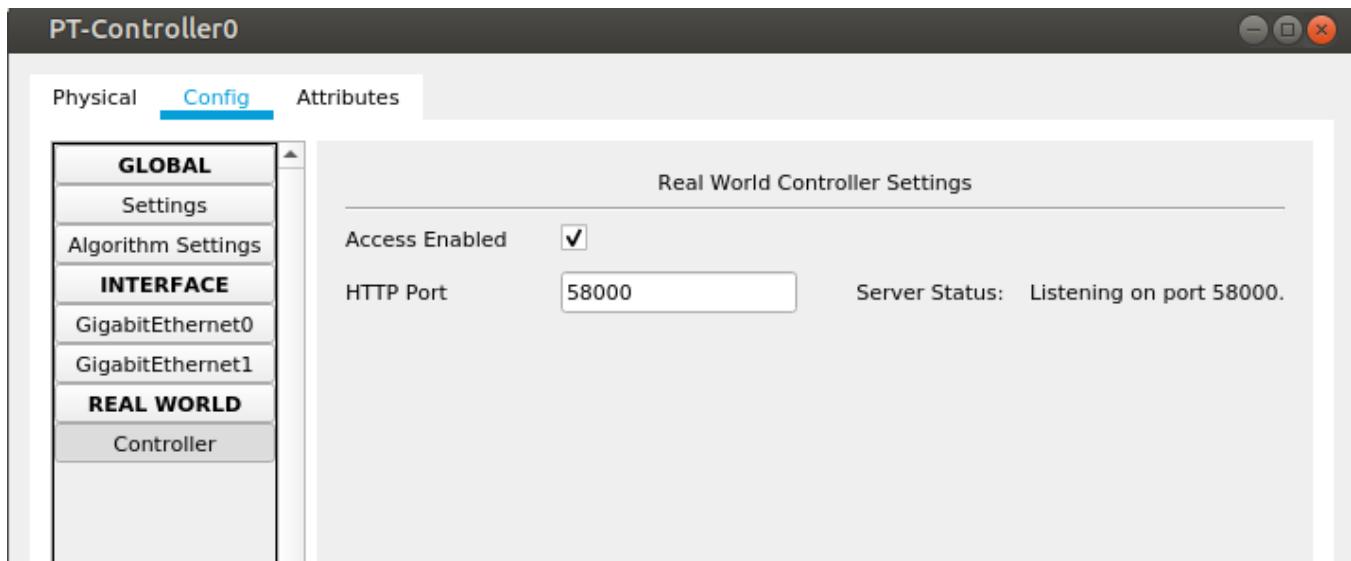




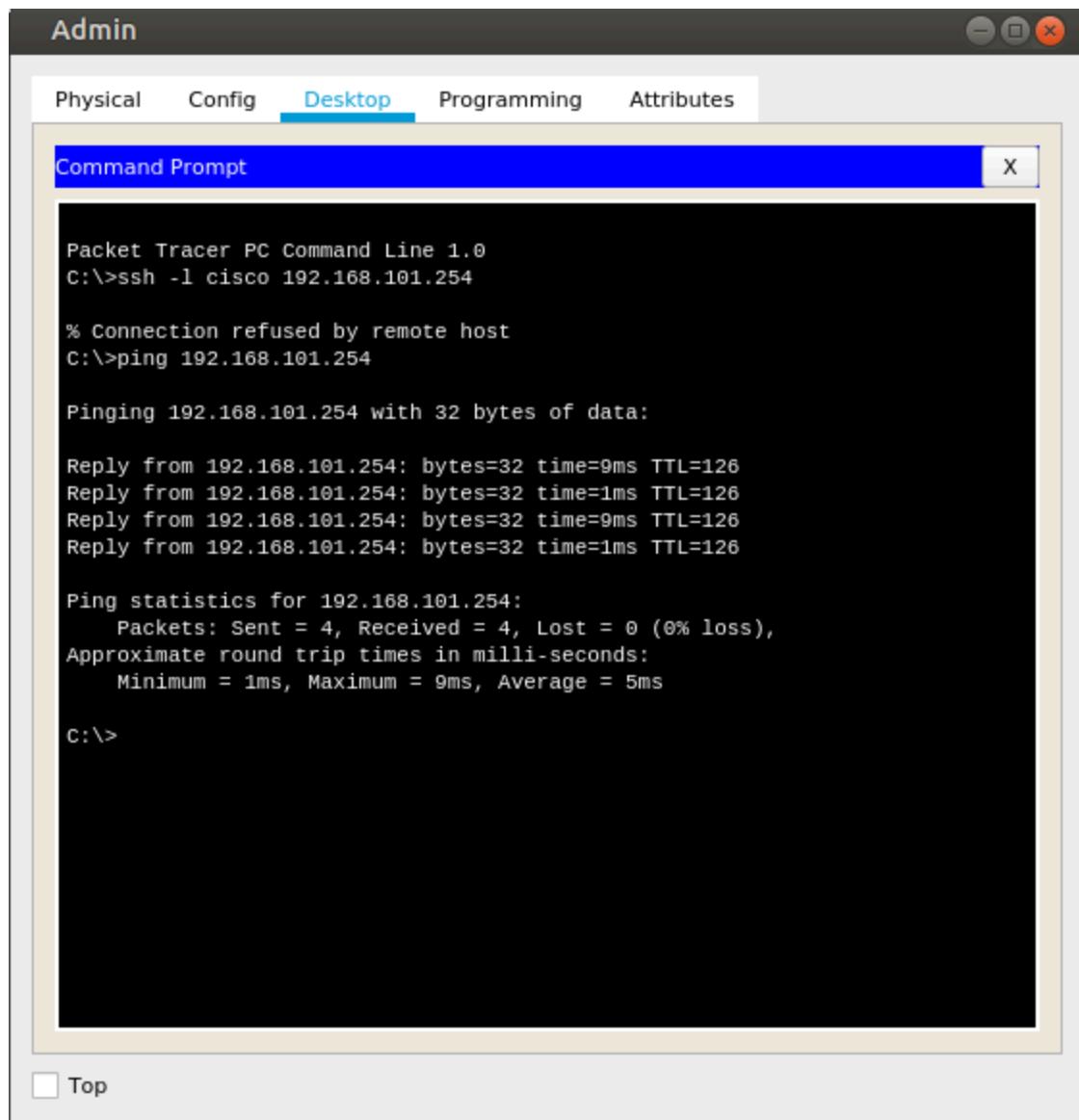
Verificamos



Habilitamos el acceso a nuestro network controller:



Desde Admin verificamos la conectividad a nuestro NetworkController



Nos registramos e iniciamos sesion dentro del PT-Controller O.

Admin

Physical Config Desktop Programming Attributes

Web Browser X

< > URL http://192.168.101.254 Go Stop

 Network Controller
User Setup

Register a new user account

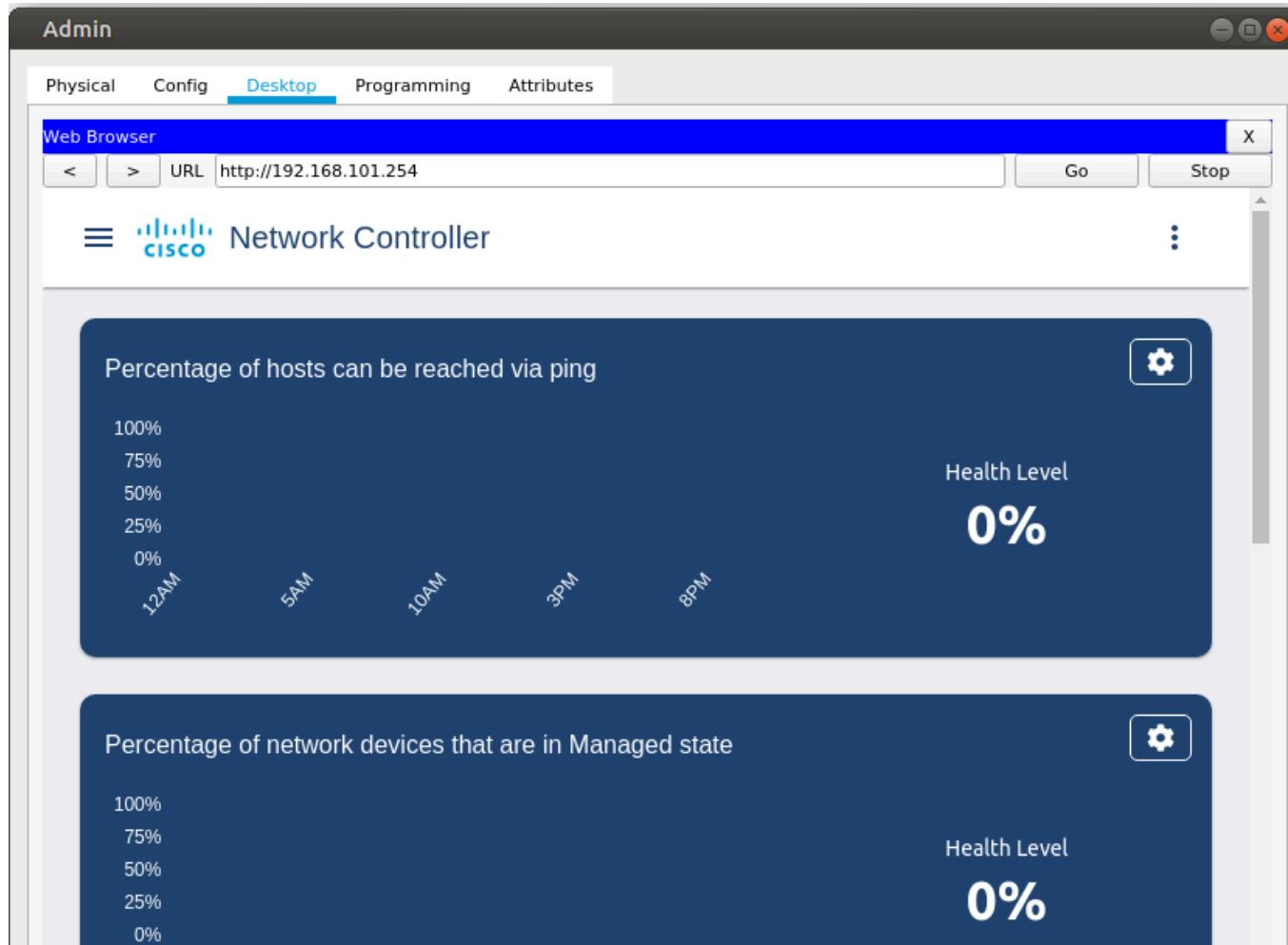
Username

Password
 

Confirm Password
 

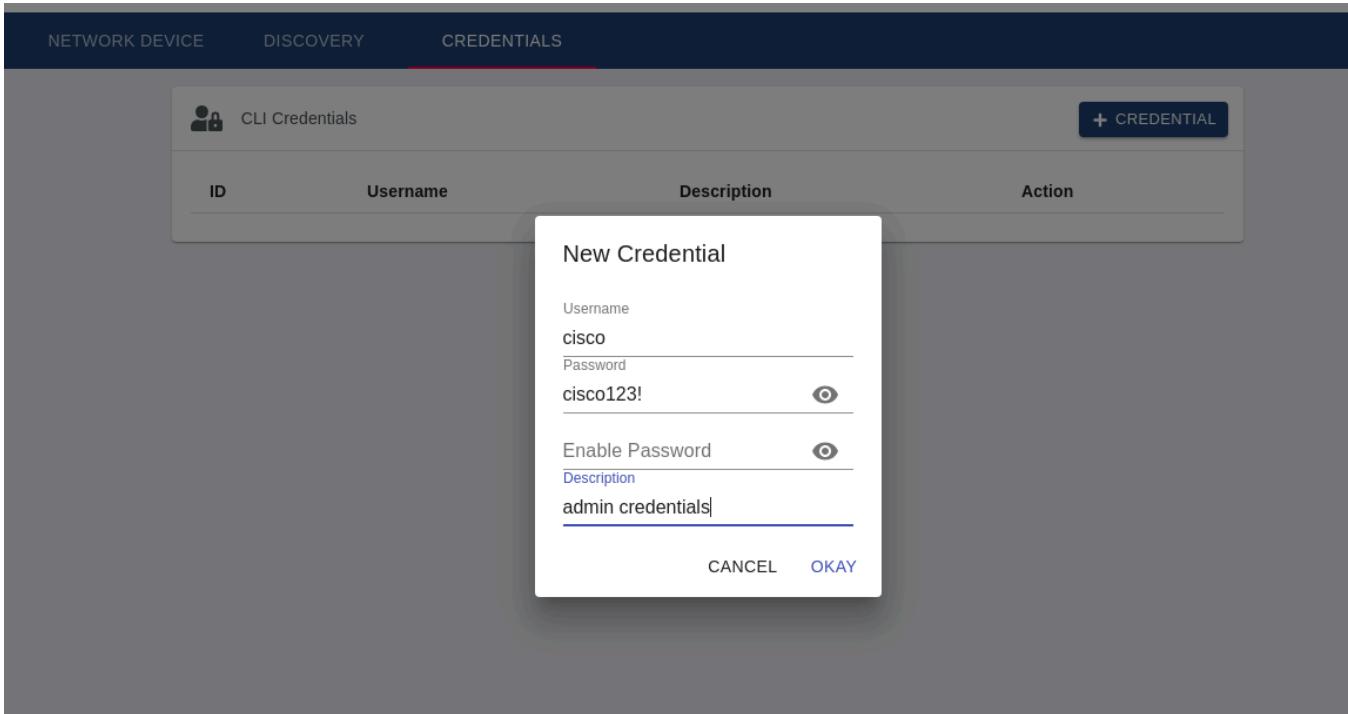
 + SETUP

Top



Usamos un SDN controller para descubrir una topología

Primero, agregamos credenciales de acceso a toda la red de dispositivos en la topología.



Usamos el CDP para decubrir los dispositivos

New Discovery

Discovery Type

CDP



Name

SWL1

IP Address

192.168.101.2

Timeout

5

Retry

3

CDP Level

No options

cisco - admin credentials x



CANCEL

ADD

URL: http://192.168.101.254

Status	Name
✓ Complete	SWL1
✓ Complete	new_device_detection

Condition: Complete ✓
Status: Inactive
Type: CDP
ID: 0

Discovery Details

CDP Level	Retry Count	TimeOut	IP Range
16	3	5	192.168.101.2

CLI Credentials

ID	Username	Description
0ed7057e-5d3b-4e3d-974a-1b272781e6bd	cisco	admin credentials

Discovered Devices

Hostname	Type	IP	Reachability Status
[Search]		0.0.0.0	Unreachable
[Search] R3	Router	10.0.1.1	Reachable
[Search] Admin	Pc	10.0.1.129	Reachable
[Search] PC1	Pc	10.0.1.130	Reachable
[Search] SWR1	MultiLayerSwitch	10.0.1.2	Reachable
[Search] SWR2	MultiLayerSwitch	10.0.1.3	Reachable

Usamos un SDN para recopilar información

Veamos todos los dispositivos de red descubiertos

Network Device

+ DEVICE

	Hostname	Type	IP	Up Time	Last Updated	Collection Status
[Settings]	SWL1	MultiLayerSwitch	192.168.101.2	1 minutes, 16 seconds	2024-09-28 18:25:03	Managed
[Settings]	R1	Router	192.168.101.1	1 minutes, 16 seconds	2024-09-28 18:25:03	Managed
[Settings]	R3	Router	192.168.2.1	1 minutes, 16 seconds	2024-09-28 18:25:03	Managed
[Settings]	R2	Router	192.168.2.2	1 minutes, 16 seconds	2024-09-28 18:25:03	Managed
[Settings]	SWR1	MultiLayerSwitch	10.0.1.2	1 minutes, 16 seconds	2024-09-28 18:25:03	Managed
[Settings]	SWR2	MultiLayerSwitch	10.0.1.3	1 minutes, 16 seconds	2024-09-28 18:25:03	Managed
[Settings]	SWL2	MultiLayerSwitch	192.168.102.2	1 minutes, 16 seconds	2024-09-28 18:25:03	Managed
[Settings]	SWR4	MultiLayerSwitch	10.0.1.5	1 minutes, 16 seconds	2024-09-28 18:25:03	Managed
[Settings]	SWR3	MultiLayerSwitch	10.0.1.4	1 minutes, 16 seconds	2024-09-28 18:25:03	Managed

Vemos que al dar click en el boton de la izquierda, recopilamos la ifnормacion de tal dispositivo

The screenshot shows a 'Network Controller' interface with a 'Update Network Device' title. It has two main sections: 'Device Detail' and 'Device Configuration'.

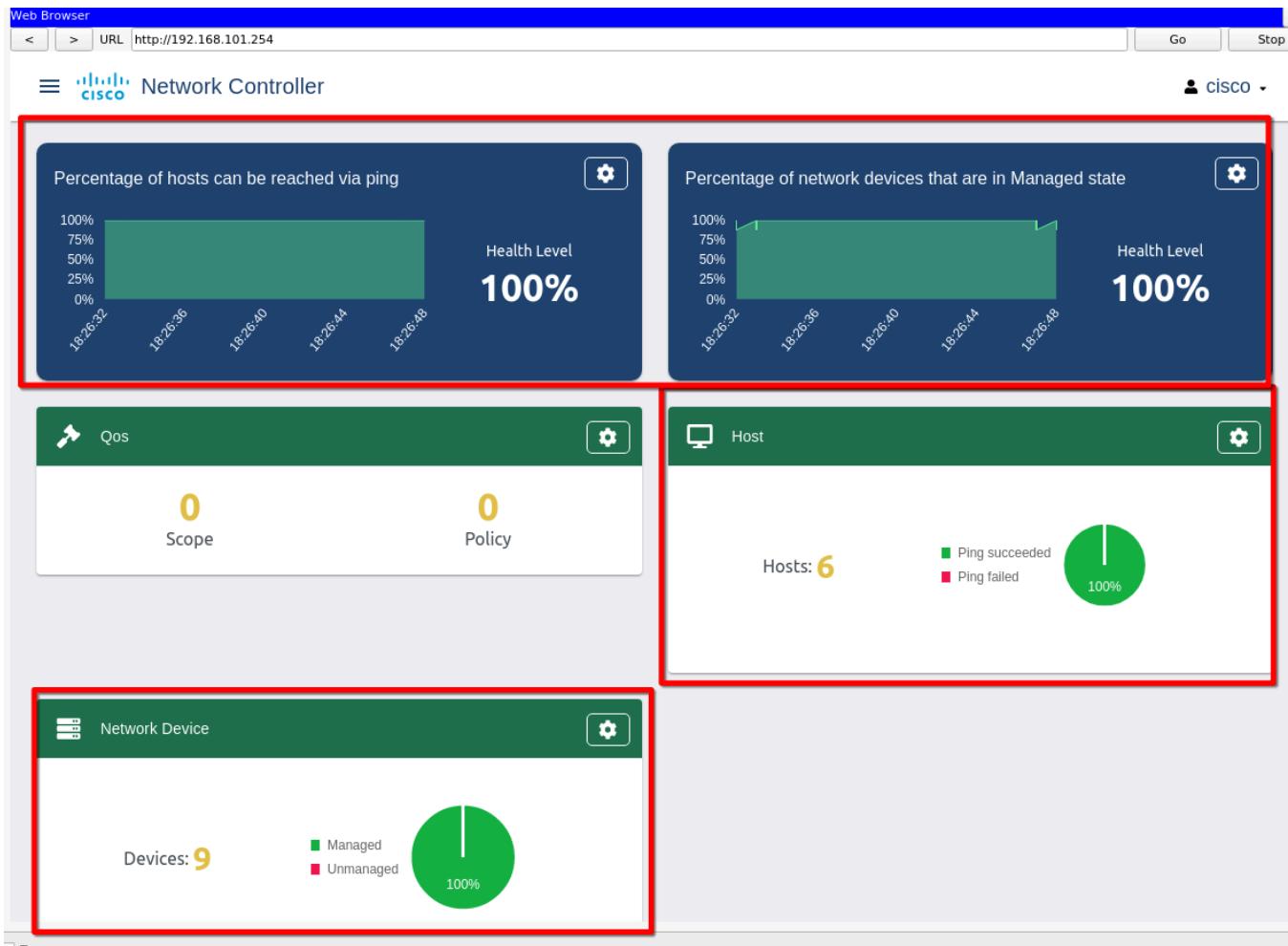
Device Detail:

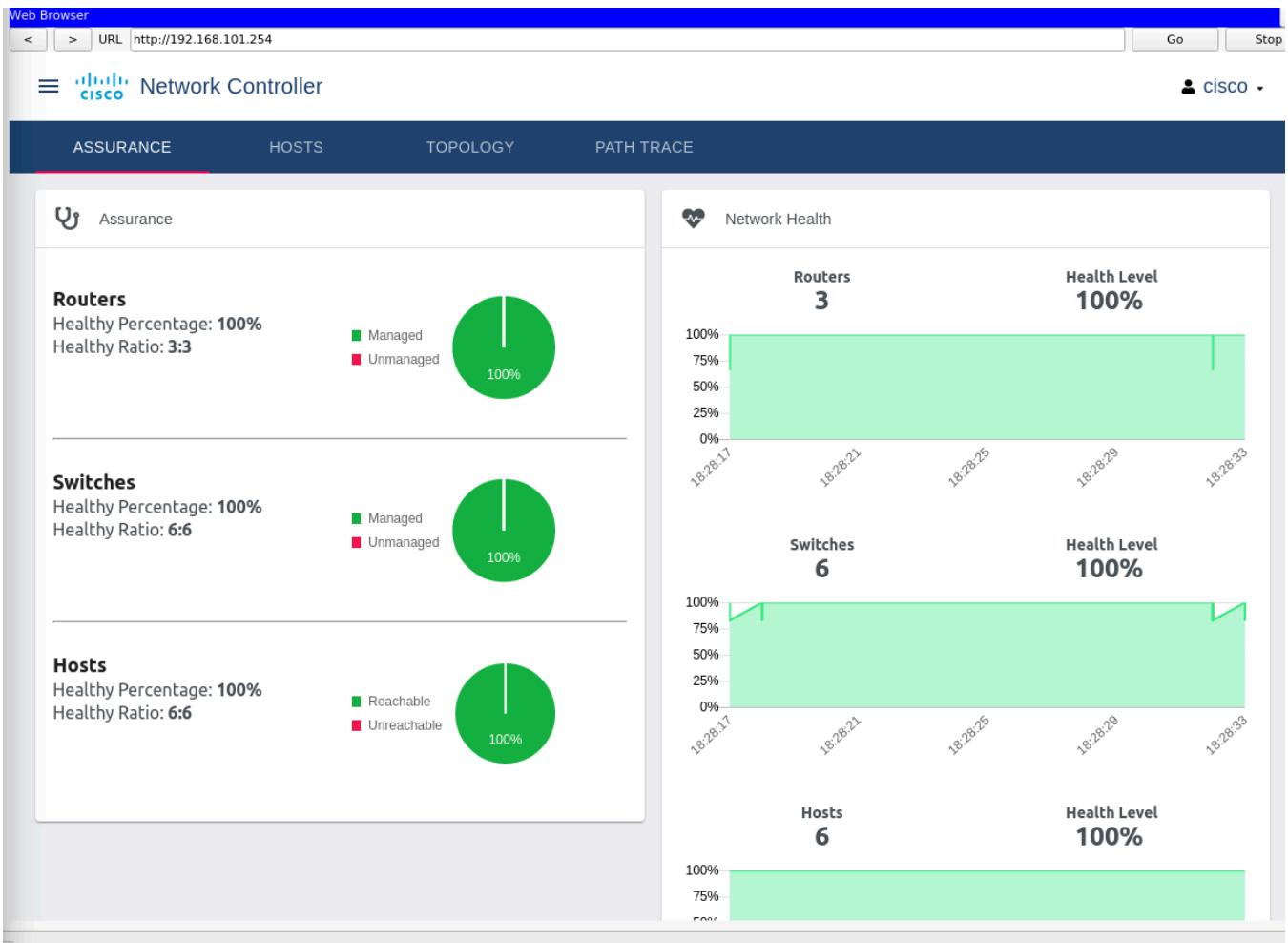
Hostname	SWL1
ID	CAT1010BT47-uuid
Interface Count	29
Software Version	16.3.2
MAC Address	000C.CF42.2B11
Management IP Address	192.168.101.2
Platform ID	3650
Product ID	3650-24PS
Serial Number	CAT1010BT47-
Type	MultiLayerSwitch
UpTime	2 minutes, 16 seconds

Device Configuration:

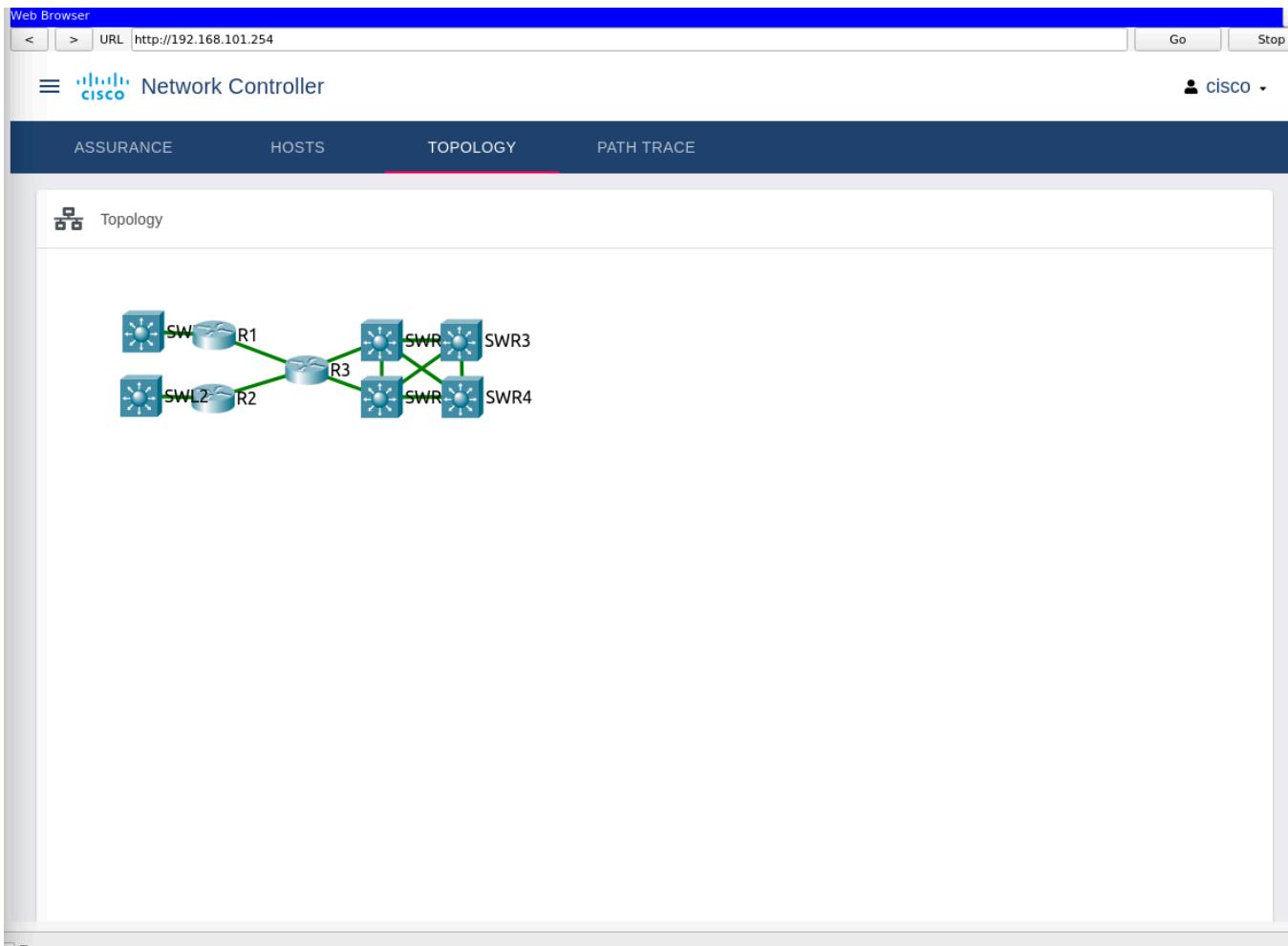
cisco - admin credentials	▼
---------------------------	---

Vamos al dashboard

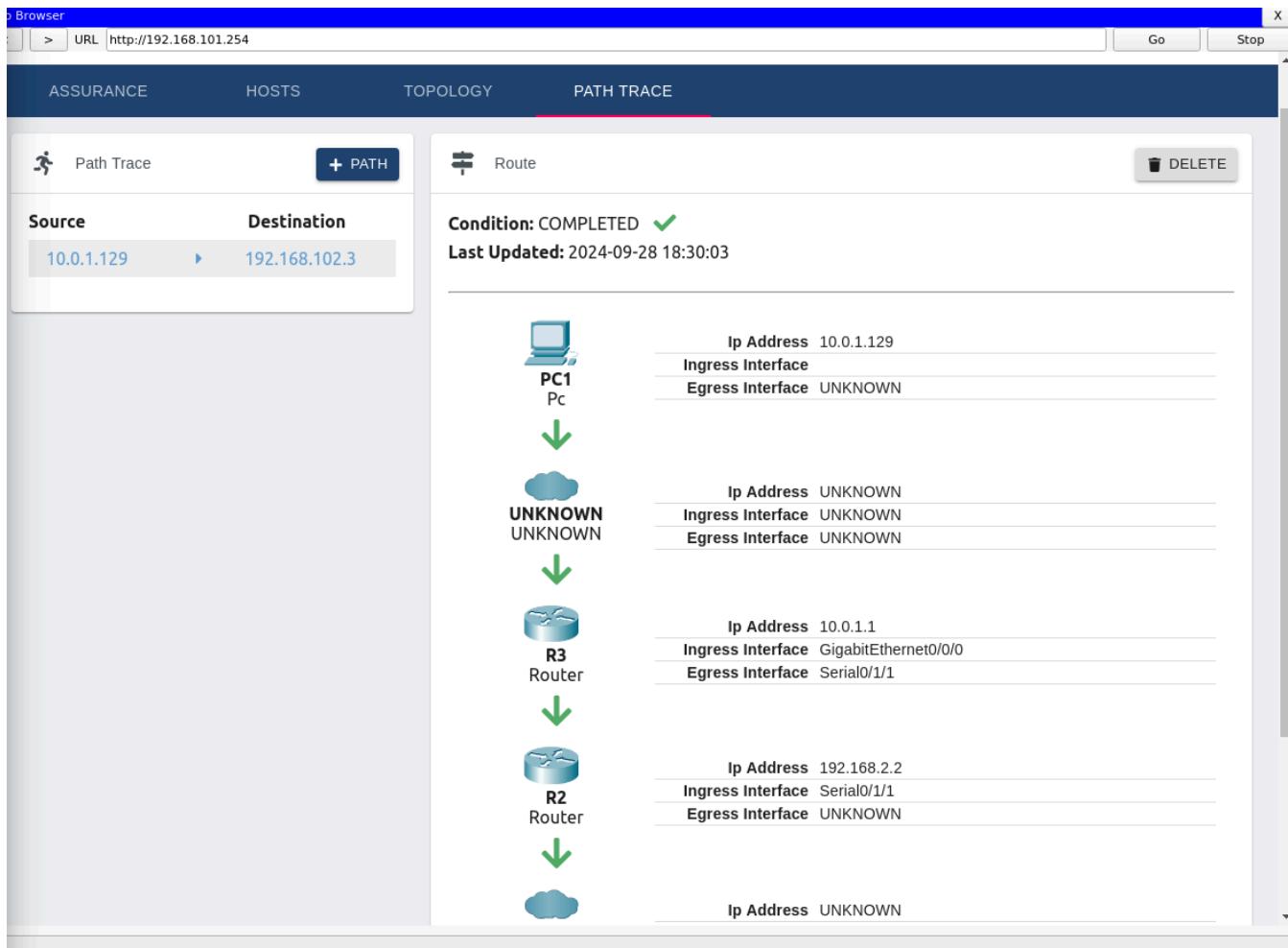




Vemos la topología creada por el Controlador

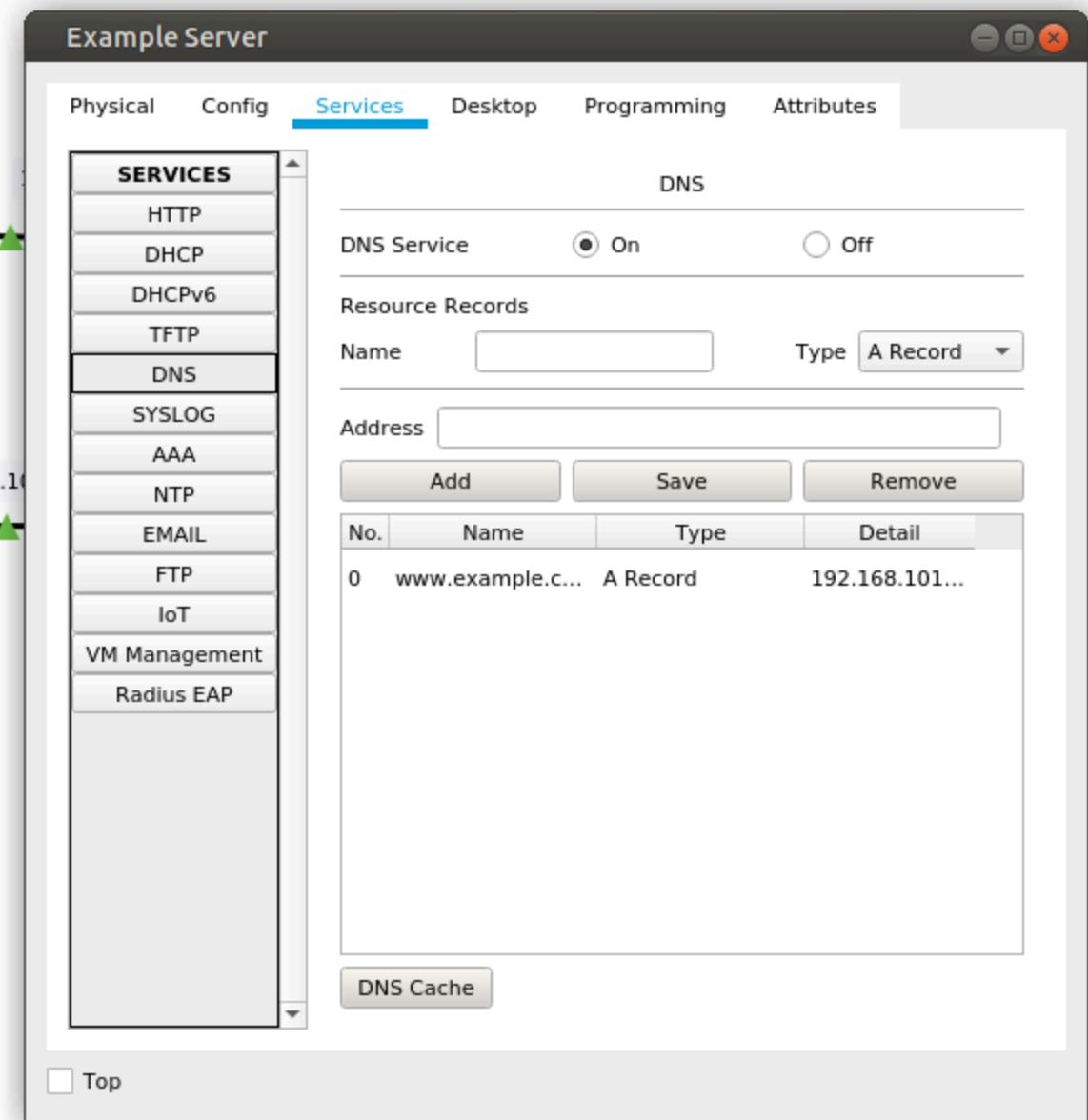


También podemos ver el PATH entre cualquier dispositivos por ejemplo entre PC-1 y PC-4



Usamo un SDN controller para configurar la configuración de red

Investigamos la configuración de Example server y vemos sus servicios habilitados



Configuramos una global policy para DNS, SYSLOG, y NTP.

QOS

NETWORK SETTINGS



Network Settings

PUSH CONFIG

AAA

Domain Name

example.com

DNS

Ip Address

192.168.101.100

NET FLOW

NTP

SYSLOG

SAVE

Note: This functionality is only available on

Saved Successfully

* and Switch 2960-24TT

Top

QOS

NETWORK SETTINGS



Network Settings



PUSH CONFIG

AAA

Server Ip

192.168.101.100

DNS

NET FLOW

SAVE

NTP

SYSLOG

Note: This functionality is only available on

Saved Successfully

* and Switch 2960-24TT

Top

QOS

NETWORK SETTINGS



Network Settings

PUSH CONFIG

AAA

- +

DNS

Server Ip (1)

192.168.101.100

NET FLOW

NTP

SAVE

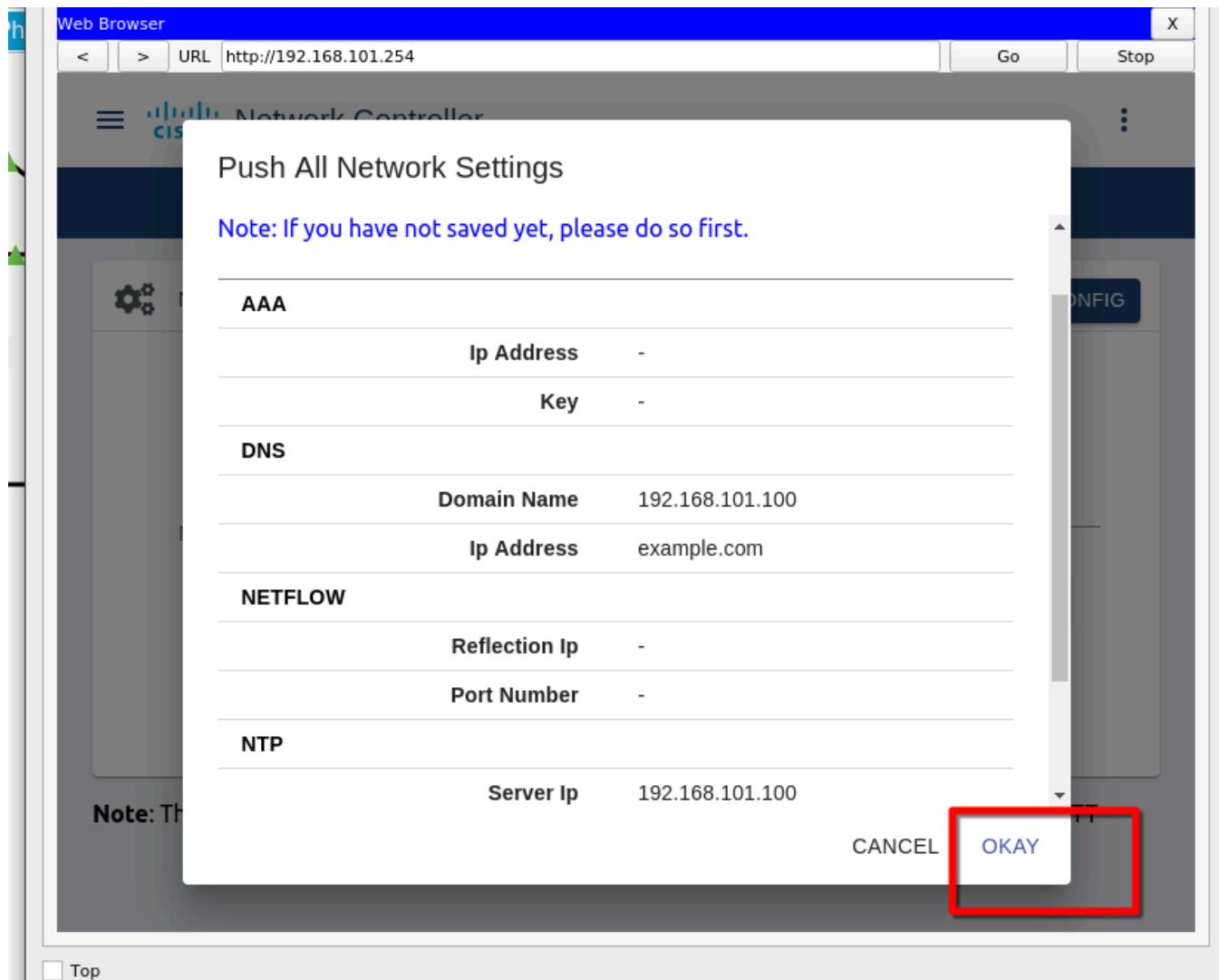
SYSLOG

Note: This functionality is only available on the

Saved Successfully

* and Switch 2960-24TT

Top



Verificamos y testeamos la configuración de red previamente actualizada con el Router 1.

Primer verificamos el DNS

```
R1>enable
R1#show run | begin ip domain
ip domain-name example.com
ip name-server 192.168.101.100
!
```

Continuamos con la config de NTP.

```
R1#show ntp associations
address          ref clock      st  when   poll   reach   delay
offset           disp
~192.168.101.100. INIT.        16   -     64     0     0.00
0.00            0.24
* sys.peer, # selected, + candidate, - outlyer, x falseticker, ~
configured
R1#show clock
*21:1:13.31 UTC Thu Jun 11 2020
R1#show run | include logging
```

Verificamos si el logging está configurado

```
R1#show run | include logging
logging 192.168.101.100
```

Finalmente testeamos este último.

```
-- 
R1#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
R1(config)#interface s0/1/0
R1(config-if)#shutdown

R1(config-if)#
%LINK-5-CHANGED: Interface Serial0/1/0, changed state to administratively
down

%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0/1/0, changed
state to down

21:01:52: %OSPF-5-ADJCHG: Process 1, Nbr 192.168.2.1 on Serial0/1/0 from
FULL to DOWN, Neighbor Down: Interface down or detached
```

```
R1(config-if)#no shutdown

R1(config-if)#
%LINK-5-CHANGED: Interface Serial0/1/0, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0/1/0, changed
state to up

R1(config-if)#
21:02:48: %OSPF-5-ADJCHG: Process 1, Nbr 192.168.2.1 on Serial0/1/0 from
LOADING to FULL, Loading Done

R1(config-if)#end
R1#
%SYS-5-CONFIG_I: Configured from console by console

R1#
```

Para finalizar verificamos que el example server tenga los mismos mensajes de syslog y este logeado al servidor.

Example Server

Physical Config Services Desktop Programming Attributes

SERVICES

- HTTP
- DHCP
- DHCPv6
- TFTP
- DNS
- SYSLOG**
- AAA
- NTP
- EMAIL
- FTP
- IoT

VM Management

Radius EAP

Syslog

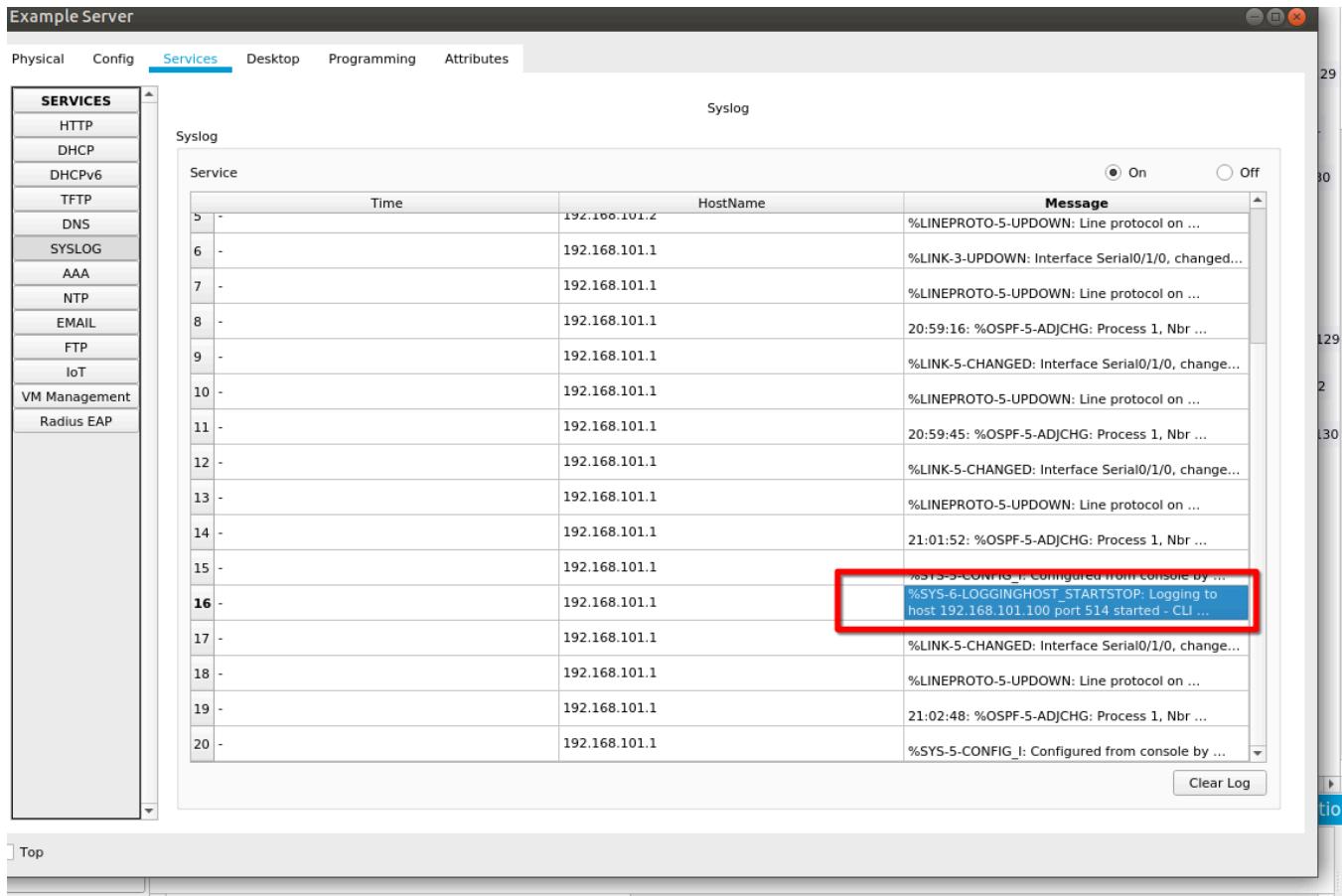
Service

Time	HostName	Message
1 -	192.168.101.1	*LINK-3-UPDOWN: Interface Serial0/1/0, changed...
2 -	192.168.101.1	*LINEPROTO-5-UPDOWN: Line protocol on ...
3 -	192.168.101.1	20:58:07: %OSPF-5-ADJCHG: Process 1, Nbr ...
4 -	192.168.101.2	*LINK-3-UPDOWN: Interface GigabitEthernet1/0/...
5 -	192.168.101.2	*LINEPROTO-5-UPDOWN: Line protocol on ...
6 -	192.168.101.1	*LINK-3-UPDOWN: Interface Serial0/1/0, changed...
7 -	192.168.101.1	*LINEPROTO-5-UPDOWN: Line protocol on ...
8 -	192.168.101.1	20:59:16: %OSPF-5-ADJCHG: Process 1, Nbr ...
9 -	192.168.101.1	*LINK-5-CHANGED: Interface Serial0/1/0, change...
10 -	192.168.101.1	*LINEPROTO-5-UPDOWN: Line protocol on ...
11 -	192.168.101.1	20:59:45: %OSPF-5-ADJCHG: Process 1, Nbr ...
12 -	192.168.101.1	*LINK-5-CHANGED: Interface Serial0/1/0, change...
13 -	192.168.101.1	*LINEPROTO-5-UPDOWN: Line protocol on ...
14 -	192.168.101.1	21:01:52: %OSPF-5-ADJCHG: Process 1, Nbr ...
15 -	192.168.101.1	*SYS-5-CONFIG_I: Configured from console by ...
16 -	192.168.101.1	*SYS-6-LOGGINGHOST_STARTSTOP: Logging to ...

On Off

Clear Log

Top



Parte 2: Construimos un CI/CD pipeline usando Jenkins

Inicializamos la VM de DEVASC.

Commiteamos el sample app a git

Creamos nuestro repositorio

File Machine View Input Devices Help

New repository

github.com/new

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner * / **Repository name ***
 sample-app is available.

Great repository names are short and memorable. Need inspiration? How about [literate-invention](#) ?

Description (optional)

 **Public**
Anyone on the internet can see this repository. You choose who can commit.

 **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

Add a README file
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

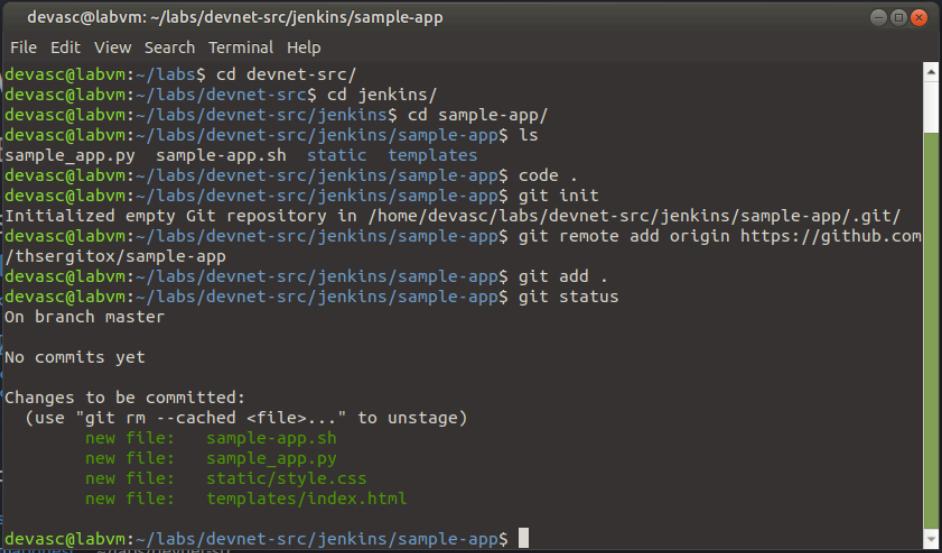
A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

 You are creating a private repository in your personal account.

Configuramos nuestras credenciales en la VM

```
devasc@labvm: ~
File Edit View Search Terminal Help
devasc@labvm:~$ git config --global user.name "thsergitox"
devasc@labvm:~$ git config --global user.name "sergiopezoj@gmail.com"
devasc@labvm:~$
```

Configuramos nuestro repositorio remoto y colocamos nuestros archivos al staged.



```
File Edit View Selection View Go Run Terminal Help
File Edit View Search Terminal Help
devasc@labvm:~/labs/devnet-src/jenkins/sample-app
devasc@labvm:~/labs/devnet-src/
devasc@labvm:~/labs/devnet-src$ cd jenkins/
devasc@labvm:~/labs/devnet-src/jenkins$ cd sample-app/
devasc@labvm:~/labs/devnet-src/jenkins/sample-app$ ls
sample_app.py sample-app.sh static templates
devasc@labvm:~/labs/devnet-src/jenkins/sample-app$ code .
devasc@labvm:~/labs/devnet-src/jenkins/sample-app$ git init
Initialized empty Git repository in /home/devasc/labs/devnet-src/jenkins/sample-app/.git/
devasc@labvm:~/labs/devnet-src/jenkins/sample-app$ git remote add origin https://github.com/thsergitox/sample-app
devasc@labvm:~/labs/devnet-src/jenkins/sample-app$ git add .
devasc@labvm:~/labs/devnet-src/jenkins/sample-app$ git status
On branch master

No commits yet

Changes to be committed:
(use "git rm --cached <file>..." to unstage)
    new file:   sample-app.sh
    new file:   sample_app.py
    new file:   static/style.css
    new file:   templates/index.html

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

Finalmente lo pusheamos

The screenshot shows a GitHub repository page for 'thsergitox/sample-app'. The repository has 1 branch and 0 tags. A recent commit was made by 'sergilopezj@gmail.com' at 4a5d901, 1 minute ago, committing sample-app files. The commit message is 'Committing sample-app files.' The commit details show changes to 'static', 'templates', 'sample-app.sh', and 'sample_app.py'. On the right side of the page, there are sections for 'About', 'Activity', 'Releases', 'Packages', 'Languages', and 'Suggested workflows'. A terminal window is overlaid on the page, showing the command-line output of a user named 'devasc@labvm' running 'git commit' and 'git push' to push the changes to the 'master' branch.

```
devasc@labvm:~/labs/devnet-src/jenkins/sample-app
File Edit View Search Terminal Help
new file: sample_app.py
new file: static/style.css
new file: templates/index.html

devasc@labvm:~/labs/devnet-src/jenkins/sample-app$ git commit -m "Committing sample-app files."
[master (root-commit) 4a5d901] Committing sample-app files.
 4 files changed, 45 insertions(+)
 create mode 100644 sample-app.sh
 create mode 100644 sample_app.py
 create mode 100644 static/style.css
 create mode 100644 templates/index.html
devasc@labvm:~/labs/devnet-src/jenkins/sample-app$ git push origin master
Username for 'https://github.com': thsergitox
Password for 'https://thsergitox@github.com':
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 2 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (8/8), 1.03 KiB | 1.03 MiB/s, done.
Total 8 (delta 0), reused 0 (delta 0)
To https://github.com/thsergitox/sample-app
 * [new branch] master -> master
devasc@labvm:~/labs/devnet-src/jenkins/sample-app$
```

Modificamos los archivos para pushearlos de nuevo

Cambiamos de puerto y versión de python

The screenshot shows a code editor with two panes displaying a file comparison. The top bar includes 'File', 'Edit', 'Selection', 'View', 'Go', 'Run', 'Terminal', and 'Help'.

The left pane shows the initial state of the files:

- sample_app.py**:

```
10 10 |     return render_template("index.html"
11 11 |
12 12 if __name__ == "__main__":
13 13+     sample.run(host="0.0.0.0", port=8080)
14 14 |
```
- sample-app.sh**:

```
8 8 cp -r templates/* tempdir/templates/
9 9 cp -r static/* tempdir/static/
10 10 |
11 11     -echo "FROM python" >> tempdir/Dockerfile
12 12+ echo "FROM python:3.9.20-slim" >> tempdir/Dockerfile
13 13 echo "RUN pip install flask" >> tempdir/Dockerfile
14 14 echo "COPY ./static /home/myapp/static" >> tempdir/Dockerfile
15 15 echo "COPY ./templates /home/myapp/templates" >> tempdir/Dockerfile
16 16+ echo "COPY sample_app.py /home/myapp/" >> tempdir/Dockerfile
17 17 echo "EXPOSE 8080" >> tempdir/Dockerfile
18 18 |
19 19 cd tempdir
20 20 docker build -t sampleapp .
21 21+ docker run -t -d -p 8080:8080 --name sampleapp
22 22 docker ps -a
23 23 |
```

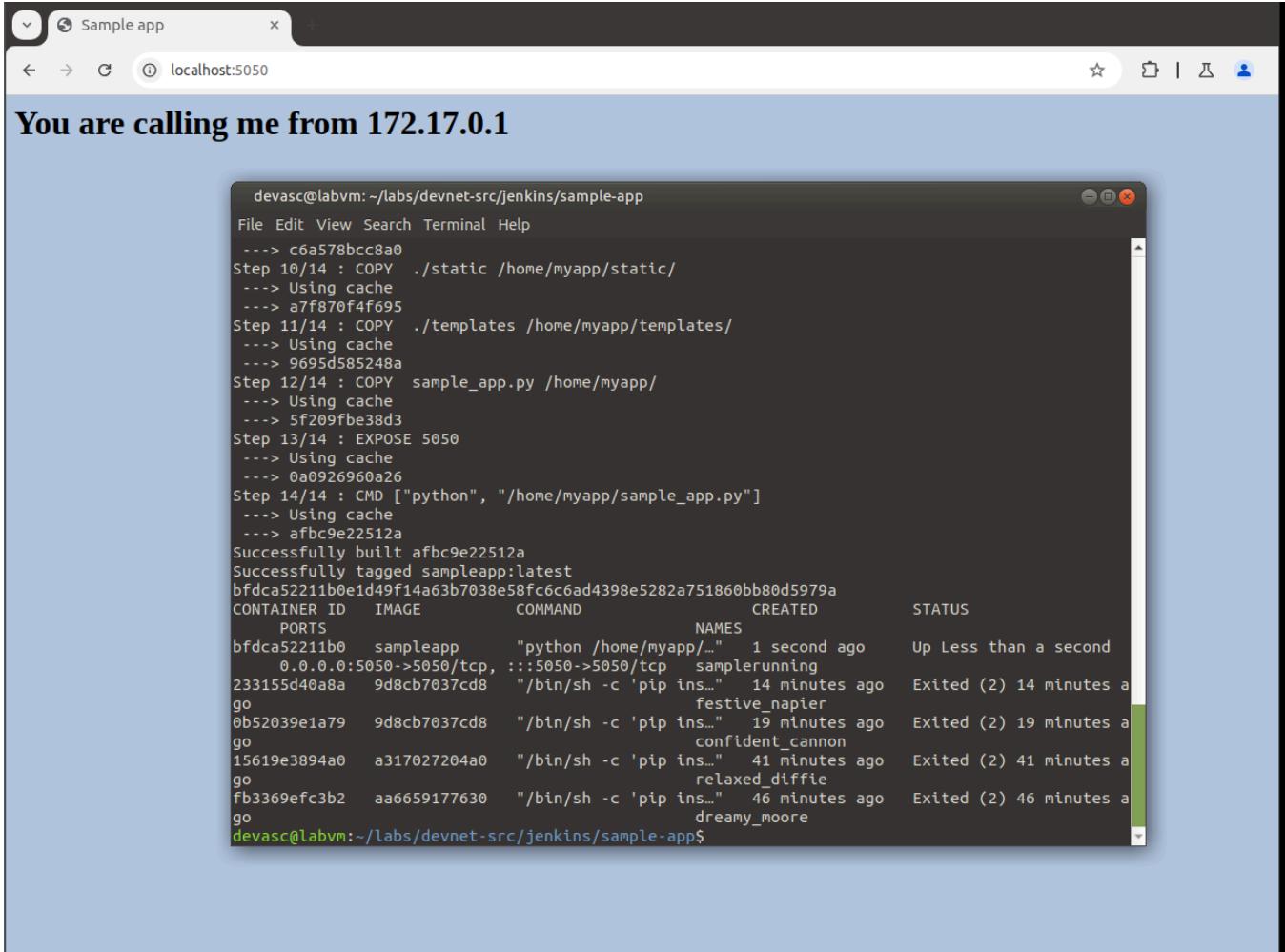
The right pane shows the updated state after changes:

- sample_app.py**:

```
10 10 |     return render_template("index.html"
11 11 |
12 12 if __name__ == "__main__":
13 13+     sample.run(host="0.0.0.0", port=5050)
14 14 |
```
- sample-app.sh**:

```
8 8 cp -r templates/* tempdir/templates/
9 9 cp -r static/* tempdir/static/
10 10 |
11 11     -echo "FROM python" >> tempdir/Dockerfile
12 12+ echo "FROM python:3.9.20-slim" >> tempdir/Dockerfile
13 13 echo "RUN pip install flask" >> tempdir/Dockerfile
14 14 echo "COPY ./static /home/myapp/static" >> tempdir/Dockerfile
15 15 echo "COPY ./templates /home/myapp/templates" >> tempdir/Dockerfile
16 16+ echo "COPY sample_app.py /home/myapp/" >> tempdir/Dockerfile
17 17 echo "EXPOSE 8080" >> tempdir/Dockerfile
18 18 |
19 19 cd tempdir
20 20 docker build -t sampleapp .
21 21+ docker run -t -d -p 8080:8080 --name sampleapp
22 22 docker ps -a
23 23 |
```

Construimos nuestra app con docker



Paramos el contenedor

```
go          dreamy_moore
devasc@labvm:~/labs/devnet-src/jenkins/sample-app$ docker stop samplerunning
samplerunning
```

Pusheamos todo

```

devasc@labvm:~/labs/devnet-src/jenkins/sample-app$ git add .
devasc@labvm:~/labs/devnet-src/jenkins/sample-app$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   requirements.txt
    modified:   sample-app.sh
    modified:   sample_app.py
    new file:   tempdir/Dockerfile
    new file:   tempdir/sample_app.py
    new file:   tempdir/static/style.css
    new file:   tempdir/templates/index.html

devasc@labvm:~/labs/devnet-src/jenkins/sample-app$ git commit -m "Changed port from 8080 to 5050."
[master 673831e] Changed port from 8080 to 5050.
 7 files changed, 42 insertions(+), 4 deletions(-)
create mode 100644 requirements.txt
create mode 100644 tempdir/Dockerfile
create mode 100644 tempdir/sample_app.py
create mode 100644 tempdir/static/style.css
create mode 100644 tempdir/templates/index.html
devasc@labvm:~/labs/devnet-src/jenkins/sample-app$ git push origin master
Username for 'https://github.com': thsergitox
Password for 'https://thsergitox@github.com':
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 2 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (7/7), 880 bytes | 880.00 KiB/s, done.
Total 7 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/thsergitox/sample-app
 4a5d901..673831e master -> master
devasc@labvm:~/labs/devnet-src/jenkins/sample-app$
```

Descargamos y corremos la imagen de Jenkins

Traemos la imagen de la Docker Hub y la iniciamos, en específico la versión 2.414.3-slim-jdk17, pues mi cuenta de GitHub estaba configurada con SSH, y el plugin para Jenkins estaba para esa versión y todo genial de ahí.

```

devasc@labvm:~/labs/devnet-src/jenkins/sample-app$ docker pull jenkins/jenkins:2.271
2.271: Pulling from jenkins/jenkins
3192219af0d4: Pull complete
17c160265e75: Pull complete
c44fe40d0e61: Pull complete
9d647f502a07: Pull complete
d108bb8c498aa: Pull complete
1bfe918b8aa5: Pull complete
dafa1a7c0751: Pull complete
a10933ea2f2f: Pull complete
dacec5718df4: Pull complete
0cd1192f374e: Pull complete
bac0875b818f: Pull complete
fc71206ecc5e0: Pull complete
65580027eff4: Pull complete
ea01a82194c6: Pull complete
be9da8492eeef: Pull complete
8351e2eec838: Pull complete
909eab257f21: Pull complete
9107ef57fb0b5: Pull complete
f925e67af94f: Pull complete
54b9422507ad: Pull complete
Digest: sha256:7648cdca09867d87462a82e6a2aac39942bd2c6deb687da2025cae3f928966cb
Status: Downloaded newer image for jenkins/jenkins:2.271
docker.io/jenkins/jenkins:2.271
devasc@labvm:~/labs/devnet-src/jenkins/sample-app$ docker run --rm -u root -p 8080:8080 -v jenkins-data:/var/jenkins_home -v $(which docker):/usr/bin/docker -v /var/run/docker.sock:/var/run/docker.sock -v "$HOME":/home --name jenkins_server jenkins/jenkins:2.271
Running from: /usr/share/jenkins/jenkins.war
webroot: EnvVars.masterEnvVars.get("JENKINS_HOME")
```

Verificamos

```
*****
Jenkins initial setup is required. An admin user has been created and a password generated.
Please use the following password to proceed to installation:

2d3e06f5022a46baaedaf4db02539bd4

This may also be found at: /var/jenkins_home/secrets/initialAdminPassword

*****
2024-09-28 20:55:26.363+0000 [id=26] INFO jenkins.InitReactorRunner$1#onAttained: Completed initialization
2024-09-28 20:55:26.378+0000 [id=20] INFO hudson.WebAppMain$3#run: Jenkins is fully up and running
2024-09-28 20:55:26.734+0000 [id=41] INFO h.m.DownloadService$Downloadable#load: Obtained the updated data file for hudson.task
aven.MavenInstaller
2024-09-28 20:55:26.734+0000 [id=41] INFO hudson.util.Retriger#start: Performed the action check updates server successfully at
attempt #1
2024-09-28 20:55:26.736+0000 [id=41] INFO hudson.model.AsyncPeriodicWork$lambda$doRun$0: Finished Download metadata. 20,295 ms
```

Configuramos Jenkins

The screenshot shows the Jenkins dashboard at `localhost:8080`. The top navigation bar includes links for 'Dashboard', 'New Item', 'Build History', 'Manage Jenkins', and 'My Views'. On the left, there are sections for 'Build Queue' (empty) and 'Build Executor Status' (2 Idle). The main content area features a 'Welcome to Jenkins!' message, a 'Start building your software project' button, and a 'Set up a distributed build' section with links for 'Set up an agent', 'Configure a cloud', and 'Learn more about distributed builds'.

Descargamos todos los plugins recomendados

The screenshot shows the Jenkins plugin manager interface at localhost:8080/pluginManager/updates. The left sidebar has a 'Download progress' section with 36 items listed, all marked as 'Pending'. The right panel shows a 'Preparation' section with two bullet points: 'Checking internet connectivity' and 'Checking update center connectivity'. The list of pending items includes Jenkins APIs like Ant, commons-text API, ASM API, SCM API, Pipeline: API, Plugin Utilities API, Font Awesome API, Bootstrap 5 API, Branch API, Build Timeout, Checks API, commons-text API, Credentials, Durable Task, JQuery3 API, ECharts API, Pipeline: Job, Email Extension, Folders, Font Awesome API, and Mina SSHD API :: Common.

Usamos Jenkins para ejecutar un build de nuestra app

Creamos un item llamado BuildAppJob

The screenshot shows the Jenkins dashboard at localhost:8080. A red box highlights the 'Build History' section, which lists a single job named 'BuildAppJob'. The table shows the last success was 4 min 29 sec ago (#4) and the last failure was 16 min ago (#3). The 'Last Duration' is 4.3 sec. Below the table, there are links for 'Icon legend', 'Atom feed for all', 'Atom feed for failures', and 'Atom feed for just latest builds'. On the left sidebar, there are links for 'New Item', 'People', 'Build History', 'Project Relationship', 'Check File Fingerprint', 'Manage Jenkins', and 'My Views'. At the bottom, there are sections for 'Build Queue' (No builds in the queue) and 'Build Executor Status' (1 Idle, 2 Idle).

Agregamos nuestras credenciales para conectarnos con GitHub y nuestro repositorio.

The screenshot shows the Jenkins job configuration page for 'BuildAppJob'. Under the 'Source Code Management' section, 'Git' is selected as the provider. A repository URL is set to 'https://github.com/thsergitox/sample-app' and a credential named 'ghcredentials' is chosen. The 'Save' button is highlighted in blue at the bottom.

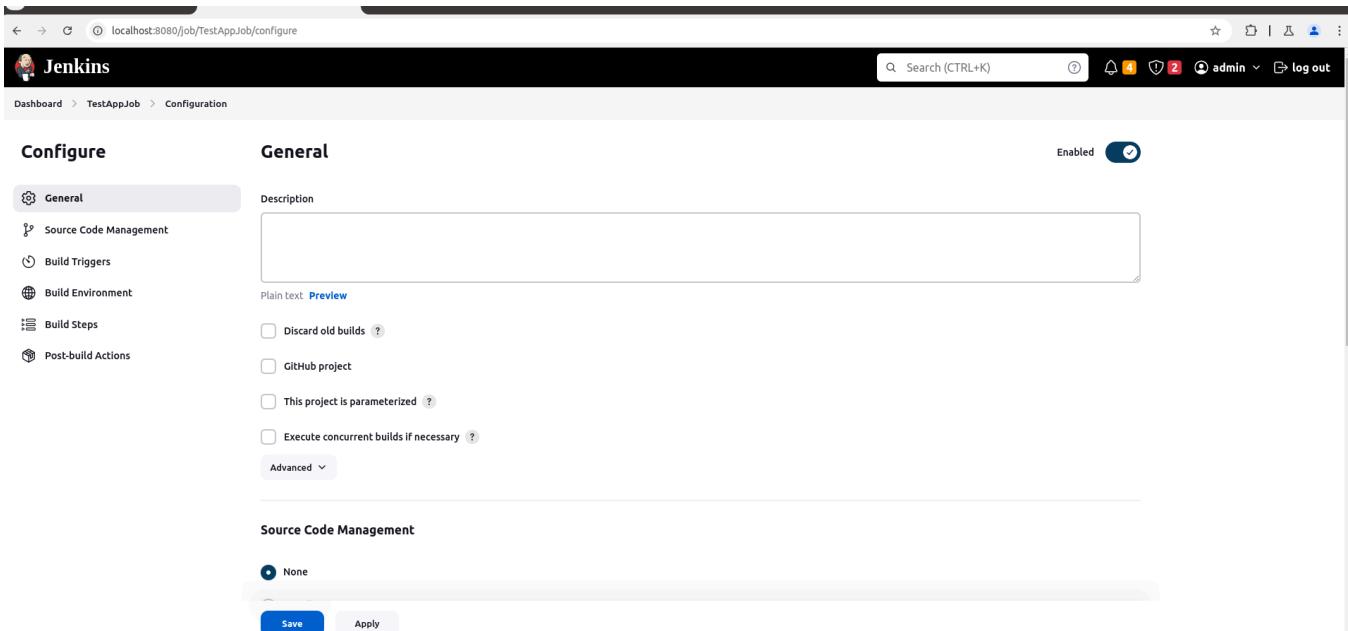
Agregamos un **build step**

The screenshot shows the Jenkins job configuration page for 'BuildAppJob'. Under the 'Build Steps' section, an 'Execute shell' step is added with the command 'bash ./sample-app.sh'. The 'Save' button is highlighted in blue at the bottom.

Realizamo un Build y lo ejecutó a la perfección

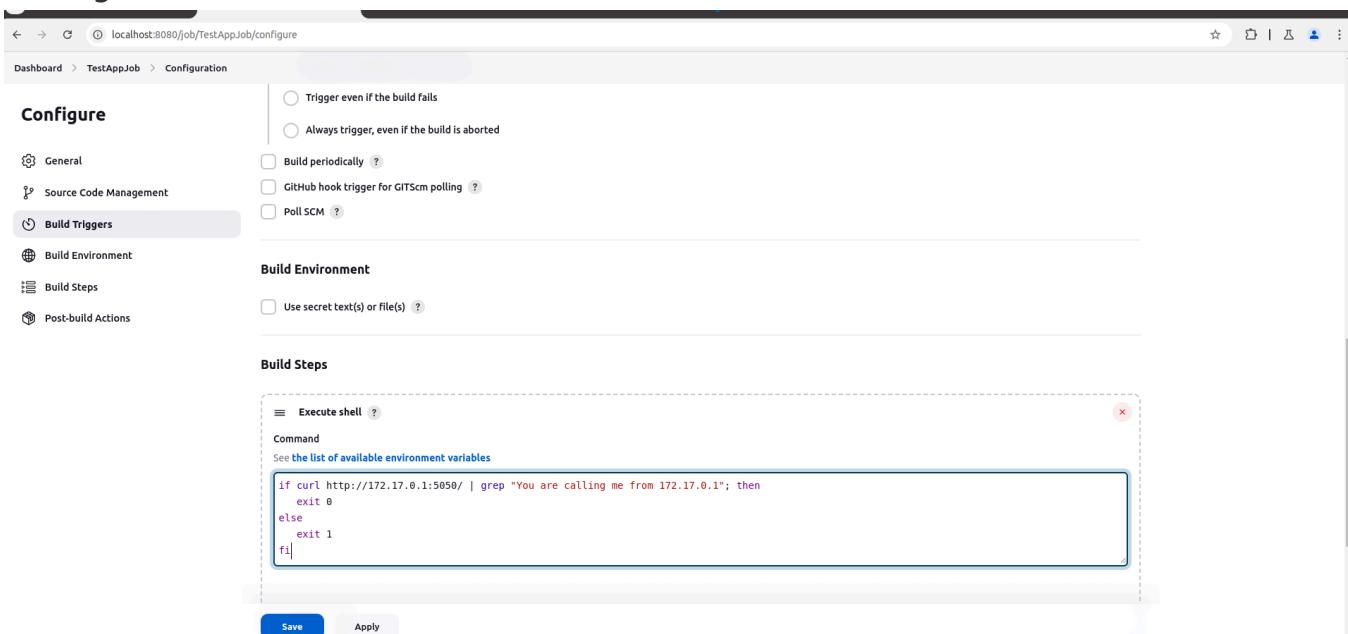
Usamos Jenkins para testear el Build

Creamos un nuevo item llamado **TestAppJob**



The screenshot shows the Jenkins configuration interface for a job named "TestAppJob". The "General" tab is selected. The "Enabled" checkbox is checked. The "Description" field is empty. Under "Source Code Management", the "None" option is selected. At the bottom are "Save" and "Apply" buttons.

Configuramos nuestro nuevo item.



The screenshot shows the Jenkins configuration interface for the "Build Triggers" section of the "TestAppJob" configuration. It includes options for triggering builds even if they fail or abort, periodic polling, GitHub hooks, and SCM polling. Under "Build Environment", there is a checkbox for using secret text or files. The "Build Steps" section contains a step to execute a shell command. The command is:

```
if curl http://172.17.0.1:5050/ | grep "You are calling me from 172.17.0.1"; then
    exit 0
else
    exit 1
fi
```

At the bottom are "Save" and "Apply" buttons.

Aqui tenemos los dos

The screenshot shows the Jenkins dashboard at localhost:8080. The main content area displays a table of jobs:

S	W	Name	Last Success	Last Failure	Last Duration
		BuildAppJob	12 min #8	1 hr 3 min #3	1.5 sec
		TestAppJob	N/A	N/A	N/A

Below the table, there are links for "Icon legend", "Atom feed for all", "Atom feed for failures", and "Atom feed for just latest builds". At the bottom right, it says "REST API" and "Jenkins 2.414.3".

Ejecutamos el build de BuilAppJob y se ejecutó con éxito, tanto ese como el test.

The screenshot shows the Jenkins console output for the TestAppJob build #1 at localhost:8080/job/TestAppJob/lastBuild/console. The left sidebar has a "Console Output" link highlighted with a red box. The main content area shows the console log:

```

Started by upstream project "BuildAppJob" build number 9
originally caused by:
Started by user admin
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/TestAppJob
[TestAppJob] $ /bin/sh -xe /tmp/jenkins1314031255895492401.sh
+ curl http://172.17.0.1:5650/
+ grep You are calling me from 172.17.0.1
      % Total    % Received % Xferd  Average Speed   Time   Time     Current
          0     0     0     0     0     0 --:--:-- --:--:-- --:--:--  0
          0    177  100  177    0     0  47149    0 --:--:-- --:--:--  59000
<h1>You are calling me from 172.17.0.1</h1>
+ exit 0
Finished: SUCCESS

```

Creamos un pipeline job

Creamos un nuevo Job de tipo pipeline, instalando el plugin Pipeline.

Dashboard > All >

Enter an item name

SamplePipelineJob > Required field

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.

Organization Folder
Creates a set of multibranch project subfolders by scanning for repositories.

If you want to create a new item from other existing, you can use this option:

OK **Copy From**

Agregamos el pipeline script

Dashboard > SamplePipelineJob > Configuration Advanced

Configure

General **Advanced Project Options** **Pipeline**

Pipeline

Definition Pipeline script

Script ?

```
1 node {
2     stage('Preparation') {
3         catchError(buildResult: 'SUCCESS') {
4             sh 'docker stop samplerunning'
5             sh 'docker rm samplerunning'
6         }
7     }
8     stage('Build') {
9         build 'BuildAppJob'
10    }
11    stage('Results') {
12        build 'TestAppJob'
13    }
14 }
```

try sample Pipeline... ▾

Use Groovy Sandbox ?

Pipeline Syntax

Save **Apply**

REST API Jenkins 2.414.3

Le damos click a Build Now y funciona

Jenkins

Dashboard > SamplePipelineJob >

Status

Pipeline SamplePipelineJob

Changes

Build Now

Configure

Delete Pipeline

Rename

Pipeline Syntax

Permalinks

- Last build (#1), 52 sec ago
- Last stable build (#1), 52 sec ago
- Last successful build (#1), 52 sec ago
- Last completed build (#1), 52 sec ago

Build History trend

#	Date	Status
1	Sep 28, 2024, 11:58 PM	Success

Atom feed for all Atom feed for failures

REST API Jenkins 2.414.3

Entramos al último Permalink y vemos como está el console uotput, y todo SUCCESS

Dashboard > SamplePipelineJob > #1

Replay

Pipeline Steps

Workspaces

```
[Pipeline] sh
+ docker stop samplerunning
samplerunning
[Pipeline] sh
+ docker rm samplerunning
samplerunning
[Pipeline]
[Pipeline] // catchError
[Pipeline]
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Build)
[Pipeline] build (Building BuildAppJob)
[Pipeline] }
Starting building: BuildAppJob #10
Build BuildAppJob #10 completed: SUCCESS
[Pipeline]
[Pipeline] // stage
[Pipeline] {
[Pipeline] { (Results)
[Pipeline] build (Building TestAppJob)
[Pipeline] }
Starting building: TestAppJob #2
Build TestAppJob #2 completed: SUCCESS
[Pipeline]
[Pipeline] // stage
[Pipeline] {
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

REST API Jenkins 2.414.3

Finalizado.

Preguntas:

Pregunta 1

Una pipeline de Jenkins es una serie automatizada de pasos que define el proceso de integración y entrega continua del software. Hay dos principales:

- Pipelines declarativas: Usan una estructura predefinida y son más fáciles de leer.
- Pipelines scripted: Ofrecen más flexibilidad pero pueden ser más complejas de escribir.

Pregunta 2

Los trabajos de Jenkins se pueden controlar programáticamente de varias formas, como el **CLI de Jenkins**, usado en el laboratorio de la pc1, jenkins tiene una interfaz de línea de comandos que puedes usar en scripts. Es útil si prefieres trabajar desde la terminal o necesitas integrar Jenkins en **scripts de shell**.

Además, tambien existen diversos **plugins y librerías específicas**, que te permiten programar trabajos de formas más avanzadas. Jenkins tiene una **API REST** bastante completa. Puedes usarla para activar, detener o controlar trabajos enviando solicitudes HTTP. Así como, también podemos configurar a Jenkins para que escuche WebHooks, que notofiquen ciertos eventos de otros sistemas.

Pregunta 3

La integración continua con Jenkins se logra automatizando el proceso de construcción y prueba del código cada vez que se realiza un cambio en nuestro repositorio localizado por ejemplo, en GitHub. Esto implica:

- Configurar Jenkins para monitorear el repositorio de código.
- Definir una pipeline que incluya pasos para compilar el código, ejecutar pruebas y generar informes con mayor agilidad y facilidad.
- Además permitiéndonos configurar notificaciones para que el equipo sepa rápidamente si hay problemas.

Pregunta 4

La aplicación de CI/CD a las redes implica crear un "gemelo digital" de la infraestructura existente para probar cambios antes de implementarlos en producción. Esto requiere modelar con precisión el hardware y software de la red, incluyendo componentes, interconexiones y configuraciones, lo cual se complica con tecnologías de virtualización como VRF y EVPN. El objetivo es poder simular cambios y validar la conectividad y funcionamiento sin afectar la red real,

empezando con pruebas básicas y evolucionando hacia verificaciones más complejas de seguridad y calidad de servicio.

La implementación de este enfoque debe ser gradual, comenzando con tareas de automatización simples y de bajo impacto, y avanzando progresivamente hacia aspectos más complejos de la configuración. Se recomienda mantener la consistencia en el diseño y configuración de la red en sitios similares, y adoptar el proceso de CI/CD como el único método para implementar cambios. Aunque inicialmente puede parecer un esfuerzo adicional, el objetivo final es reducir errores humanos, minimizar interrupciones y mejorar la estabilidad de la red mediante la adopción de las mejores prácticas de la industria del desarrollo de software.

Pregunta 5

Son tecnologías de virtualización de red:

- **Virtual Routing and Forwarding (VRF)**: permite crear múltiples instancias de tablas de enrutamiento en un mismo router, lo que posibilita la separación lógica del tráfico.
- **Ethernet VPN (EVPN)**: es una tecnología que permite extender la conectividad Ethernet a través de redes IP/MPLS, facilitando la creación de redes virtuales más flexibles y escalables.

Estas tecnologías añaden complejidad al modelado y gestión de redes, pero también ofrecen mayor flexibilidad y capacidades avanzadas de segmentación y aislamiento de tráfico.

Práctica Calificada 1 - Sergio Pezo.