Práctica calificada 1 - CC312

Sergio Sebastian Pezo Jimenez - 20224087G

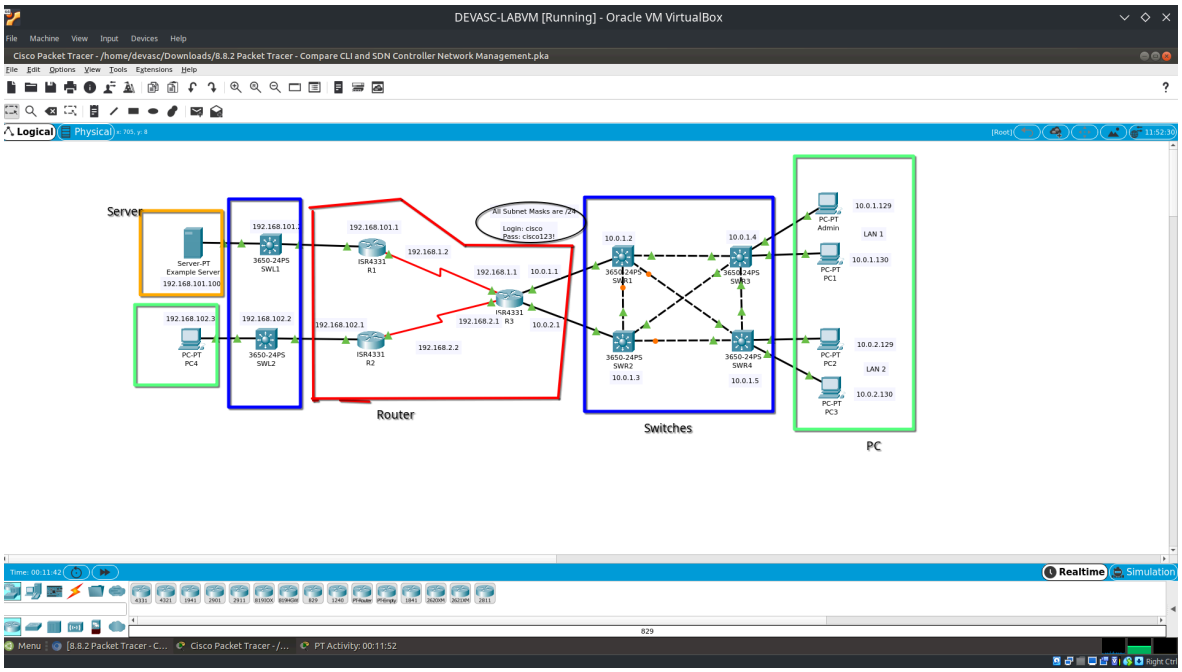
Parte 1: Packet Tracer - Comparamos el CLI y el SDN Controller NetworkManagement

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

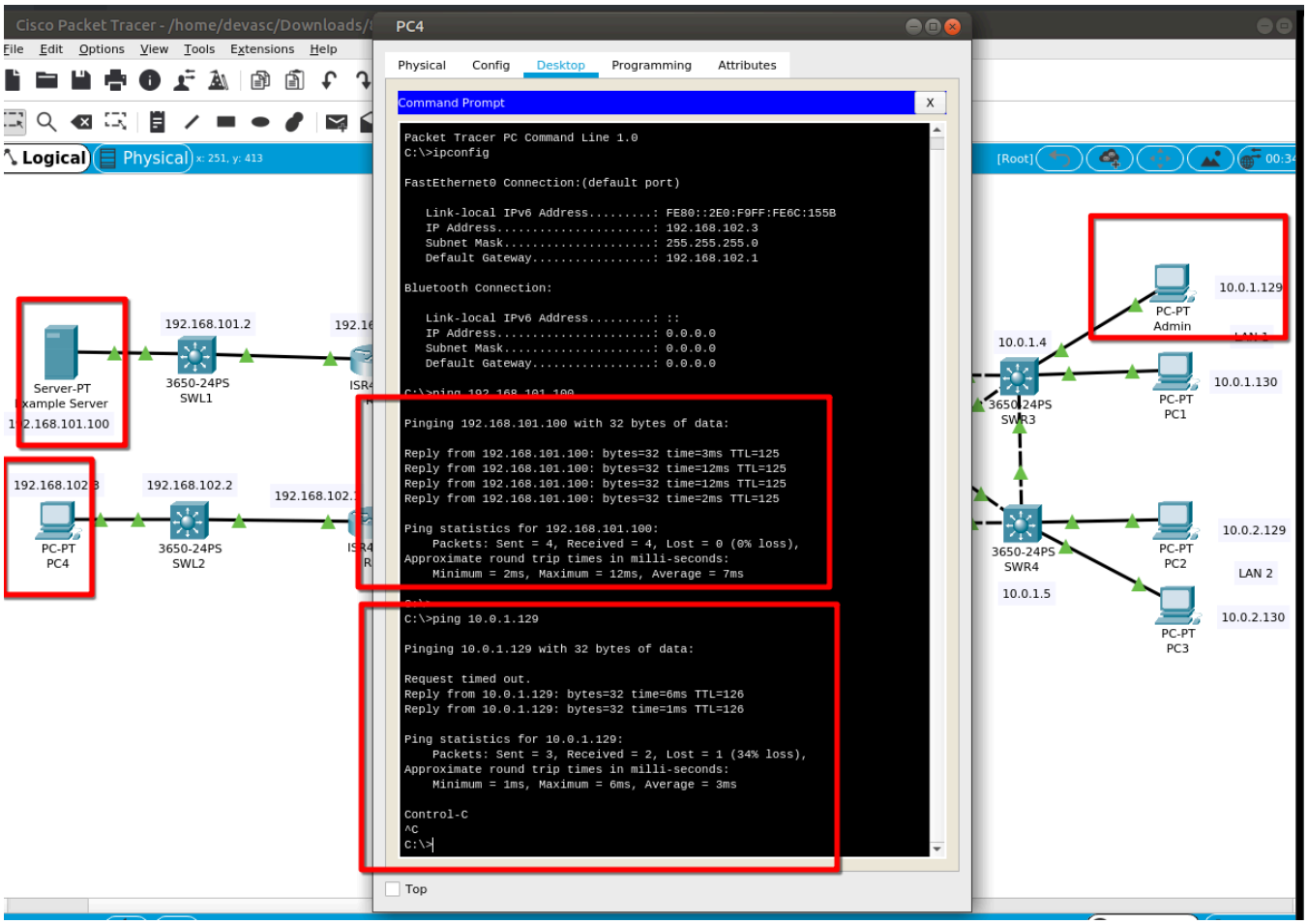
Analizamos la topología de red

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.

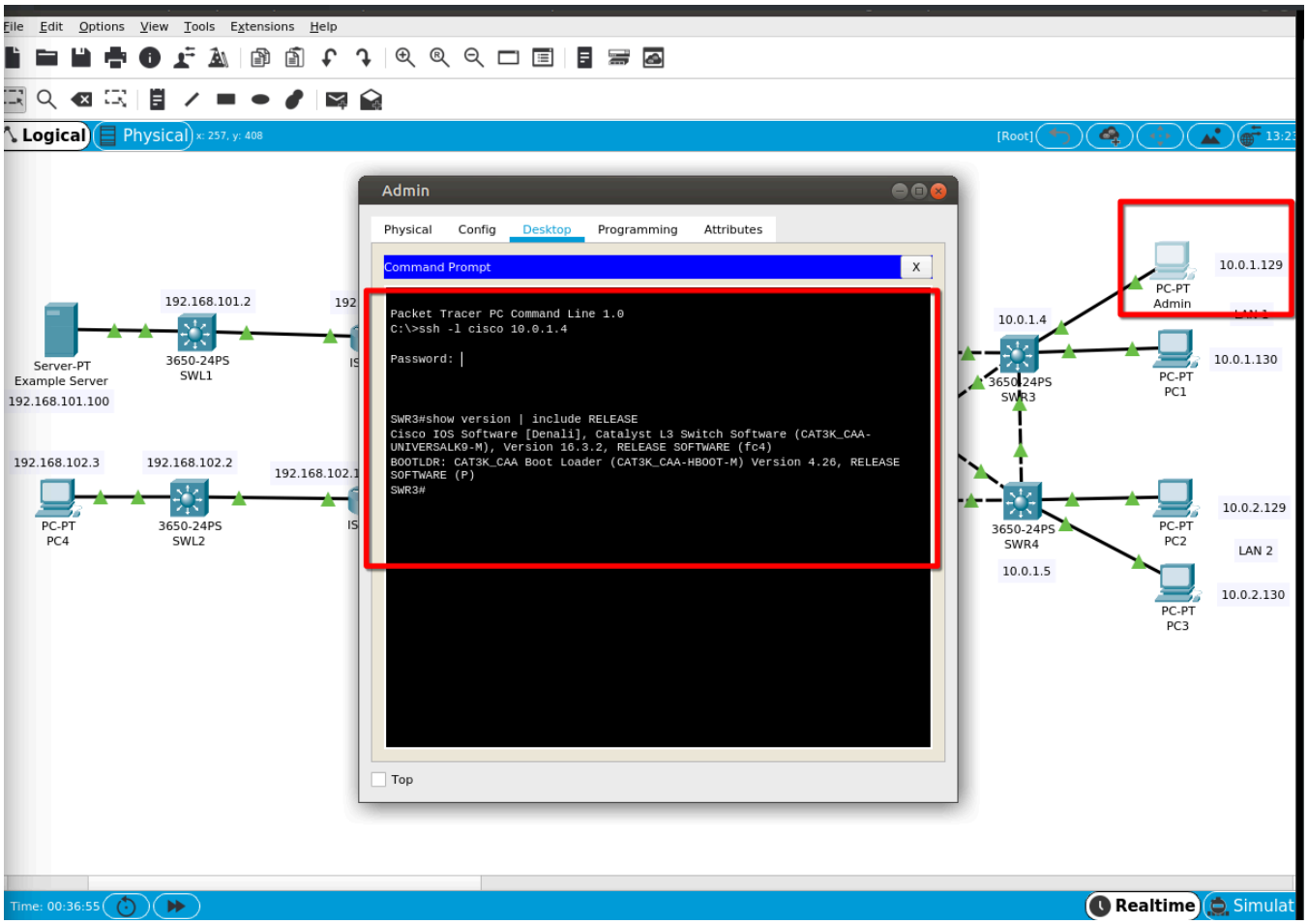


Hacemos ping desde la PC4 con el server con su ip y con la PC-Admin, de igual manera con los demás dispositivos.



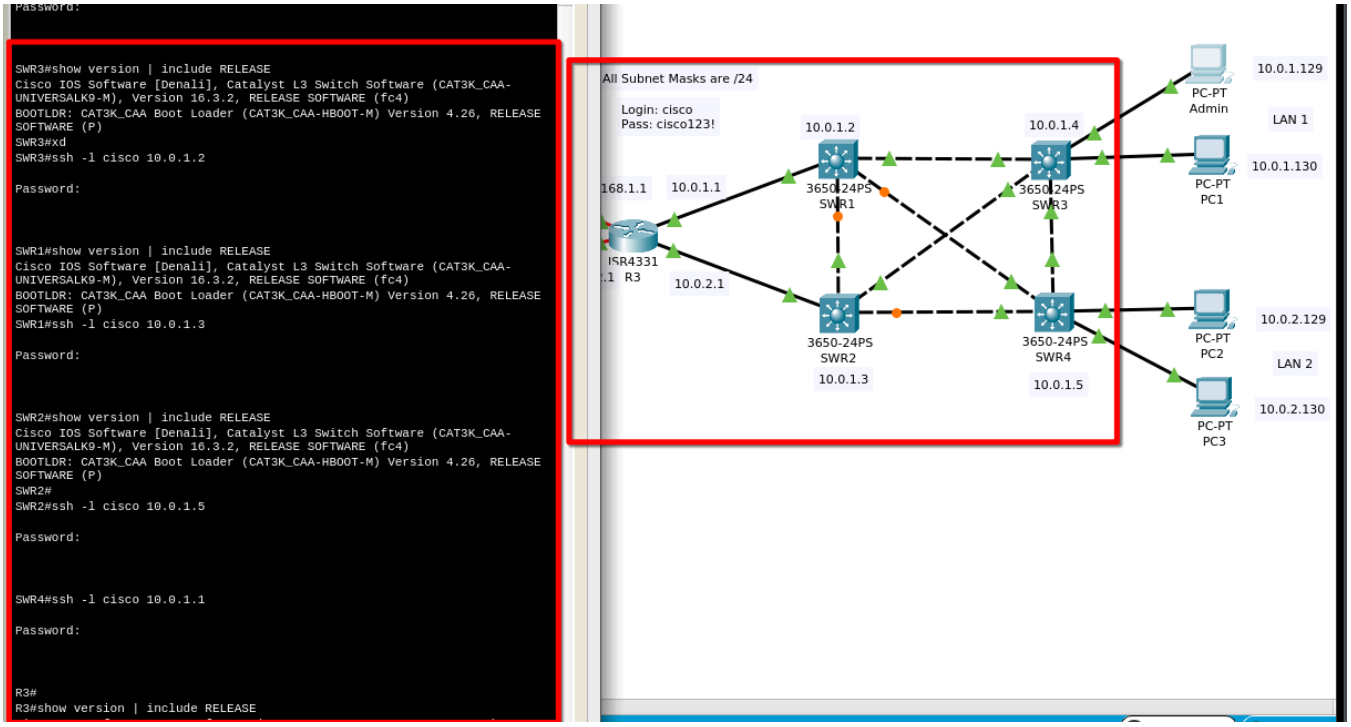
Utilizamos el CLI para recopilar información

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

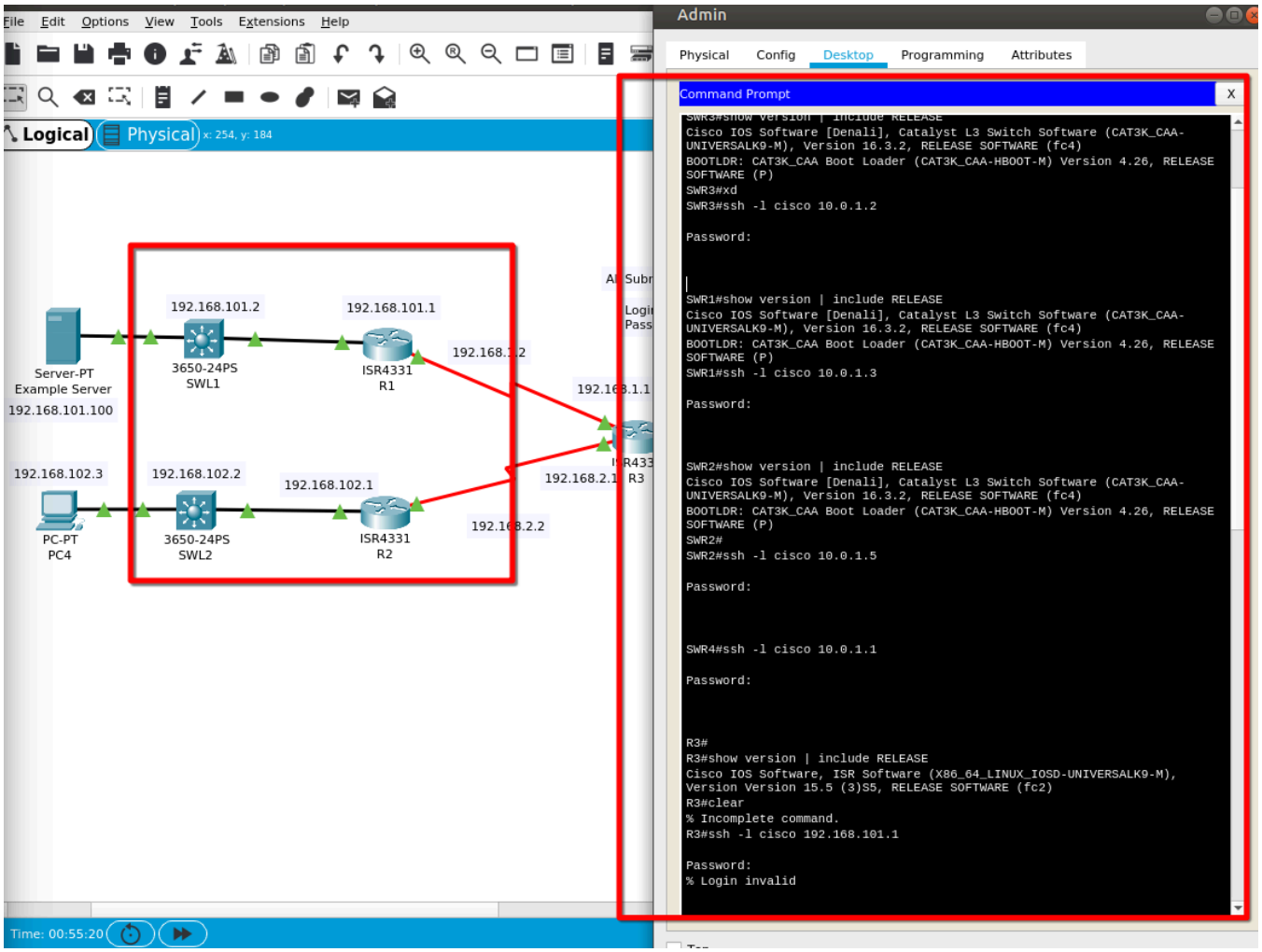


Con el command Prompt en SWR3, repetiremos el proceso para los dispositivos: SWL1, SWL2, SWR1, SWR2, SWR3, SWR4, R1, R2, y R3.

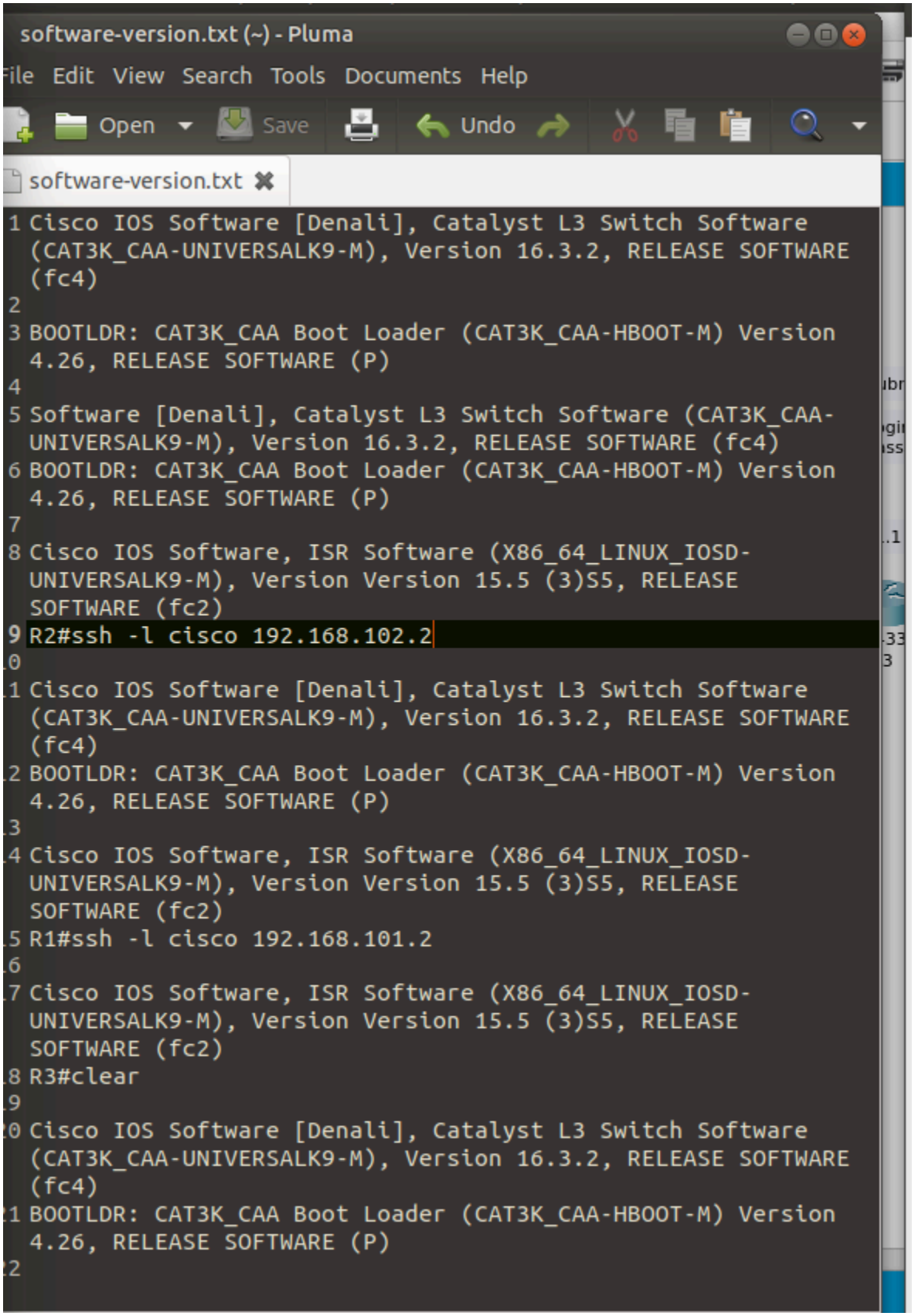
Empezamos con la primer red con:
SWR1, SWR2, SWR3, SWR4 y R3:



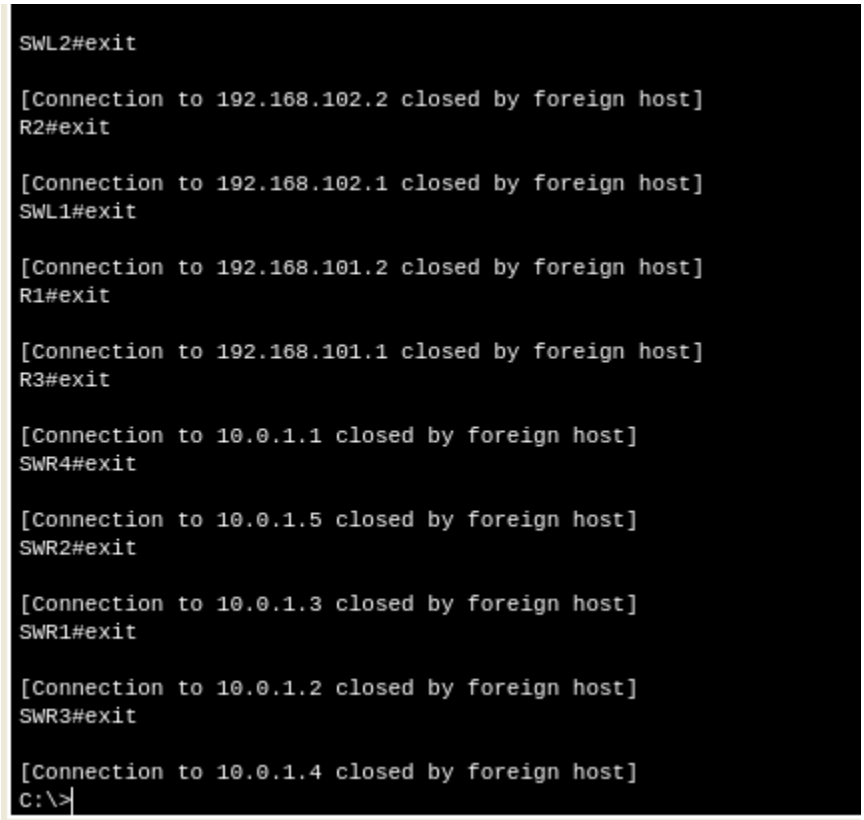
Conituamos con la red de la izquierda:



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

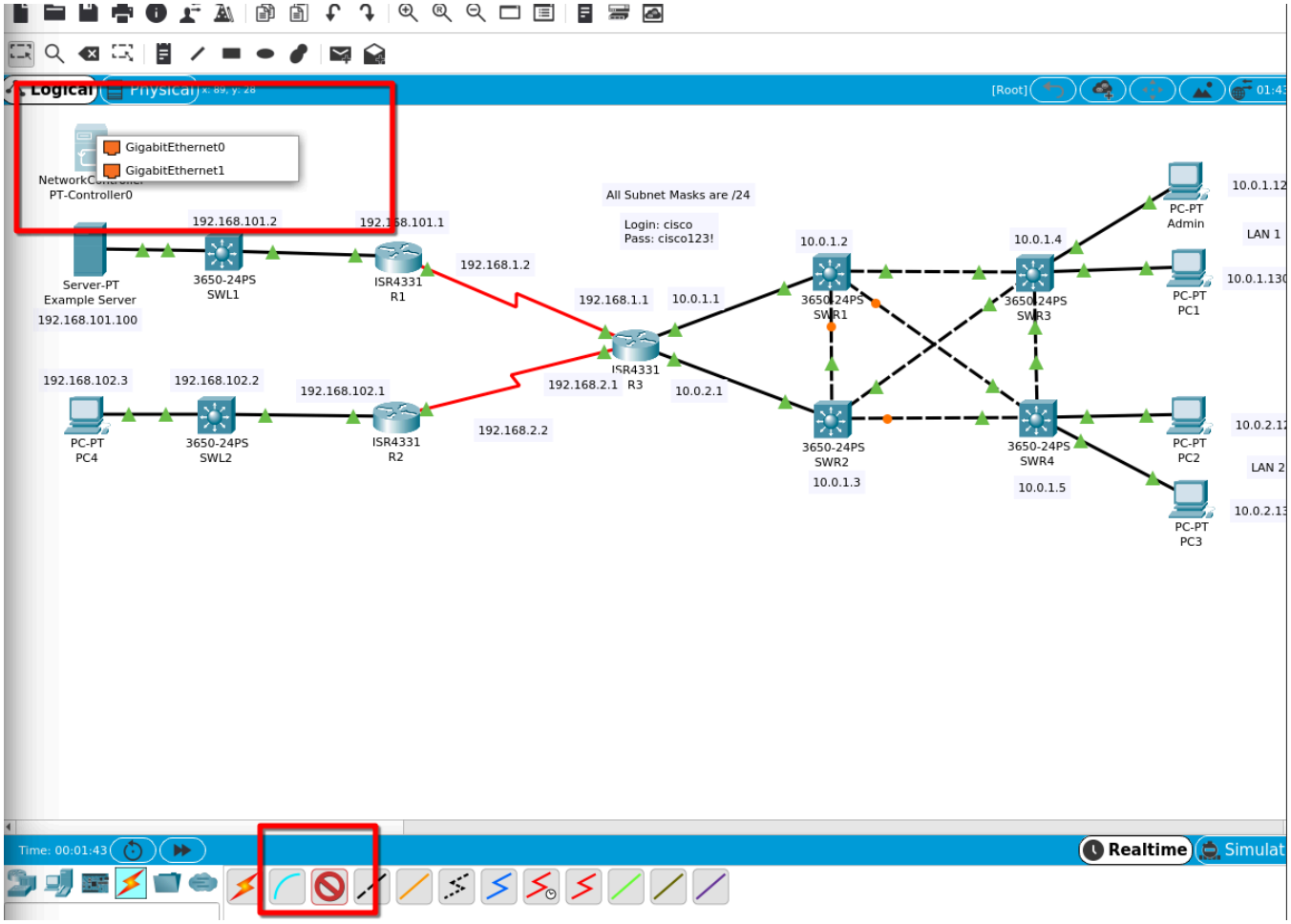


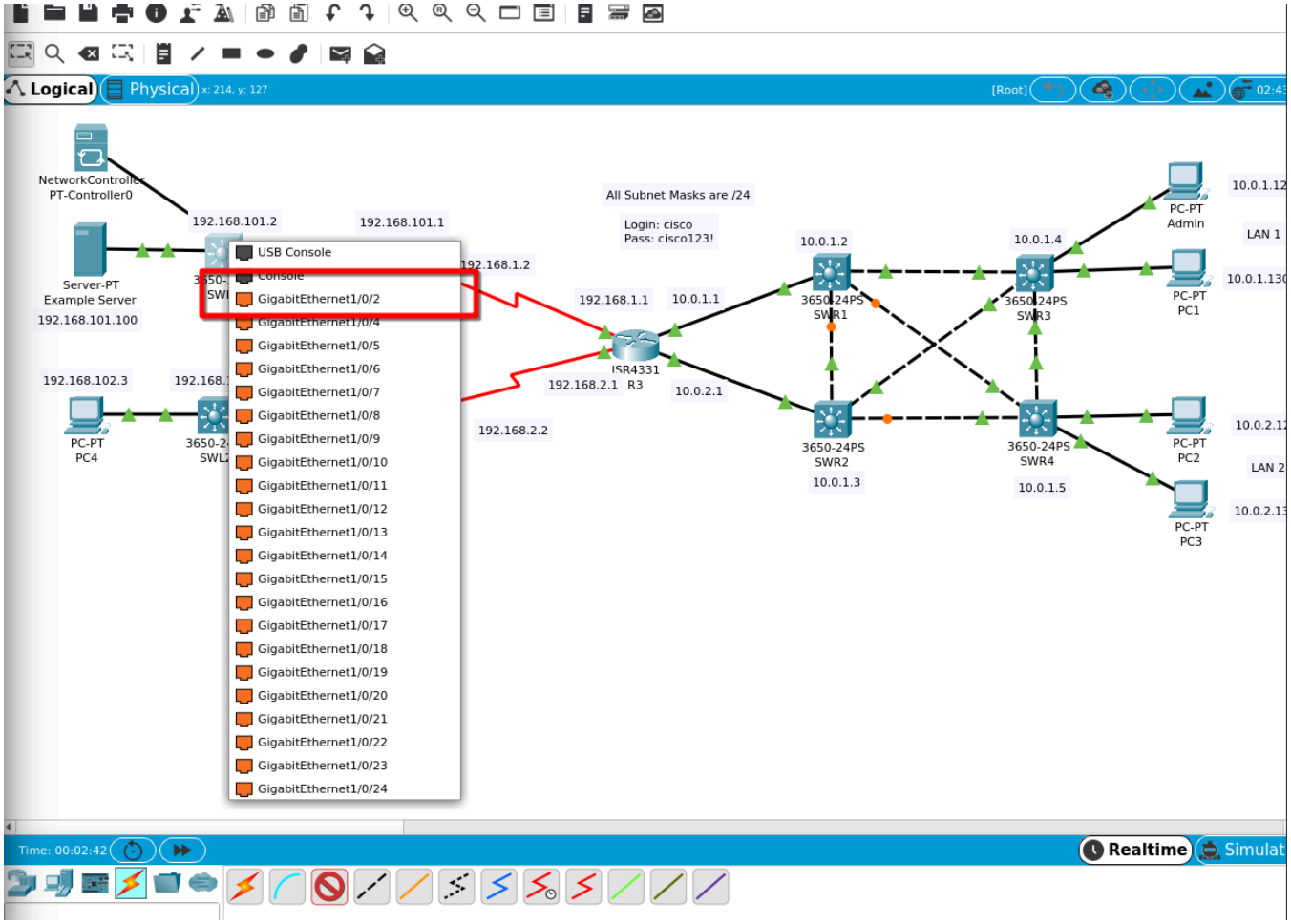
Finalmente salimos de cada máquina para volver a PC-Admin



Configuramos el controlador de PT

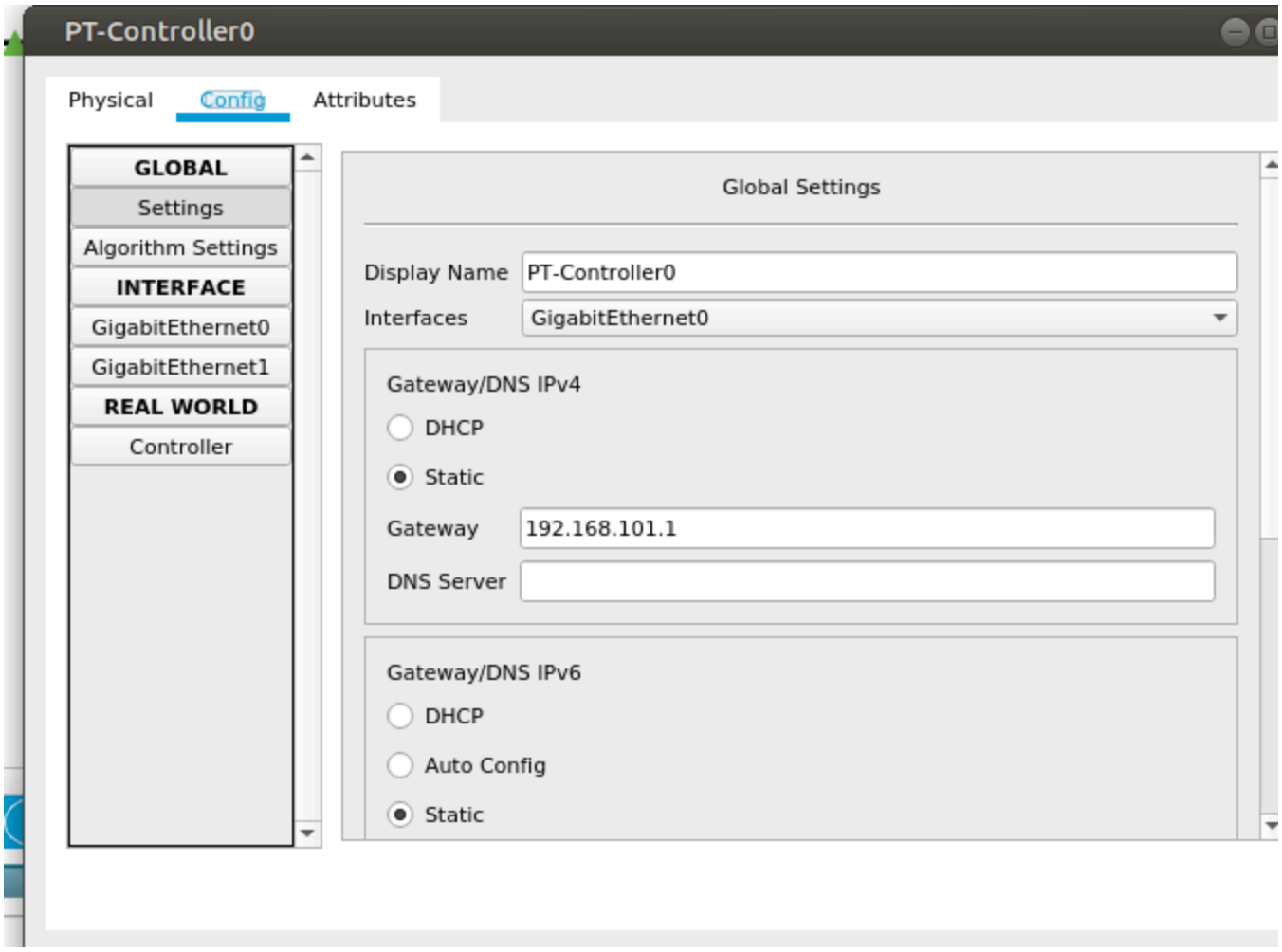
Agregamos un dispotivo NetworkController, y con un cable de cobre lo conectamos a la topología.

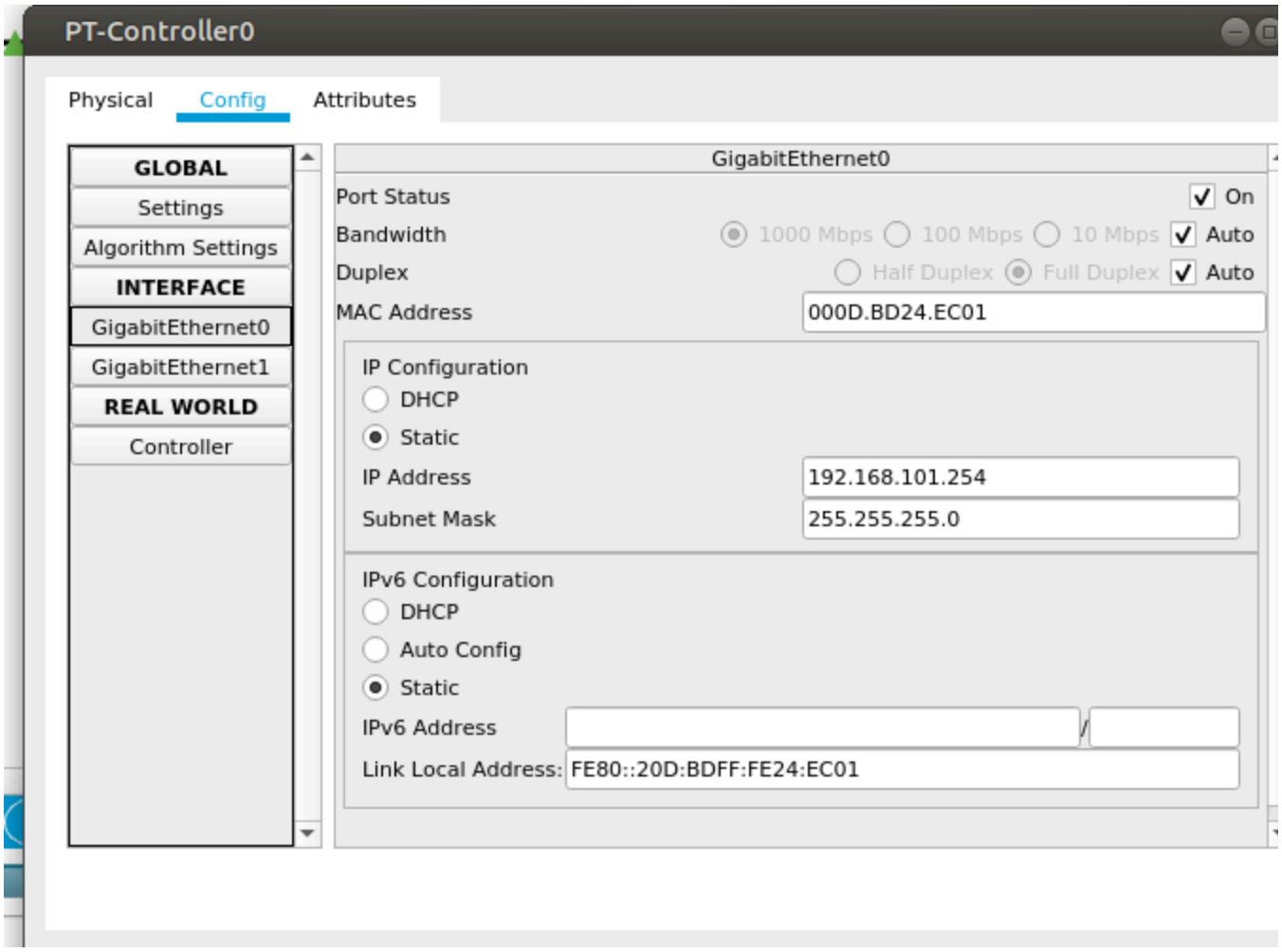

Logical
Physical
[Root]
GigabitEthernet0
GigabitEthernet1
PT-Controller0
All Subnet Masks are /24
Login: cisco
Pass: cisco123!
192.168.101.2
192.168.101.1
Server-PT
Example Server
192.168.101.100
3650-24PS
SWL1
ISR4331
R1
192.168.1.2
192.168.1.1
10.0.1.1
ISR4331
R3
192.168.2.1
10.0.2.1
10.0.1.2
3650-24PS
SWR1
10.0.1.4
3650-24PS
SWR3
PC-PT
Admin
PC-PT
PC1
LAN 1
10.0.1.12
10.0.1.130
192.168.102.3
192.168.102.2
192.168.102.1
PC-PT
PC4
3650-24PS
SWL2
ISR4331
R2
192.168.2.2
3650-24PS
SWR2
10.0.1.3
3650-24PS
SWR4
10.0.1.5
PC-PT
PC2
PC-PT
PC3
LAN 2
10.0.2.12
10.0.2.13
Time: 00:01:43
Realtime
Simulat



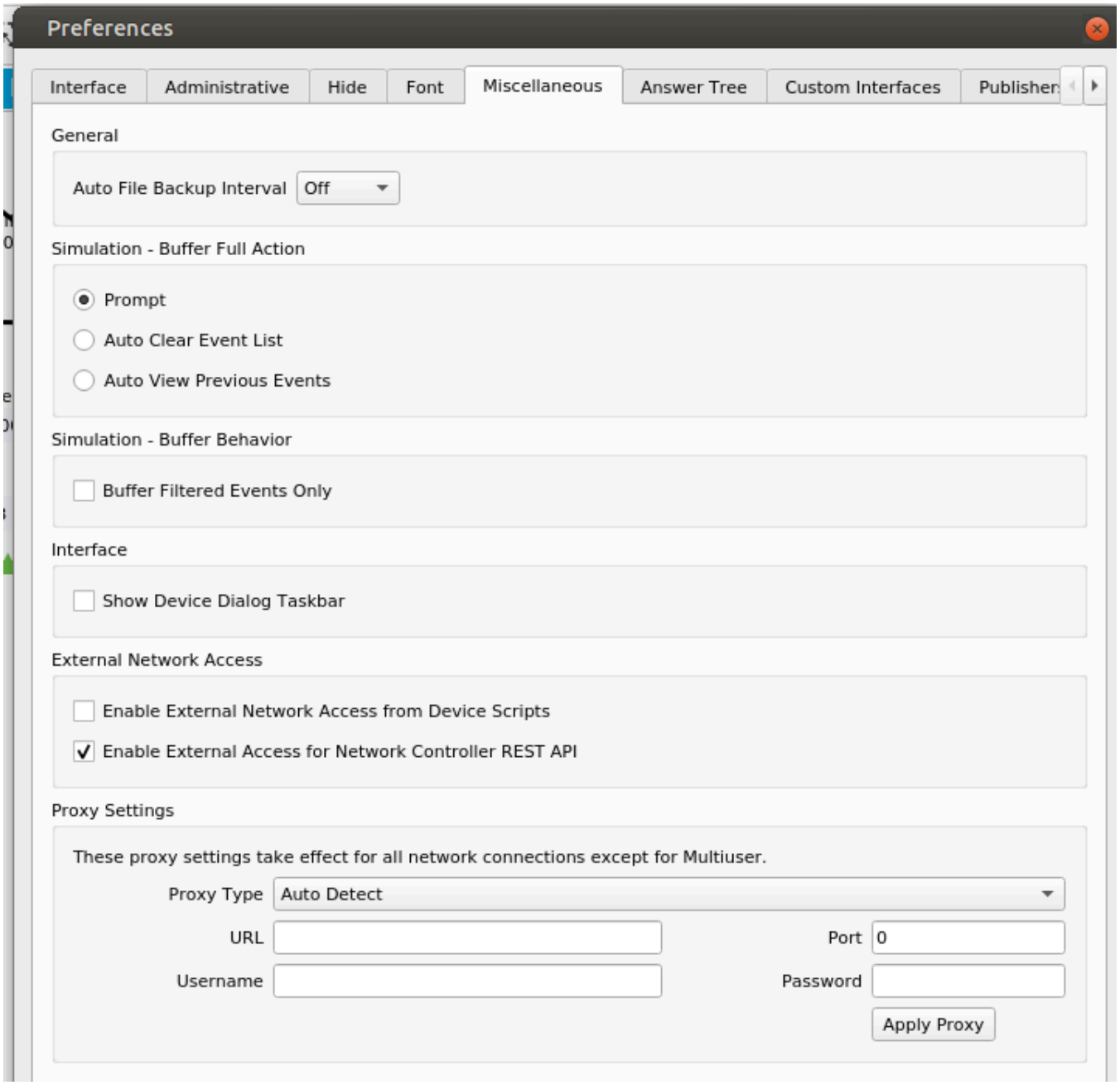
Configuramos la conectividad para nuestro PT-Controller0

Configuramos el gateway, para usar el router R1:

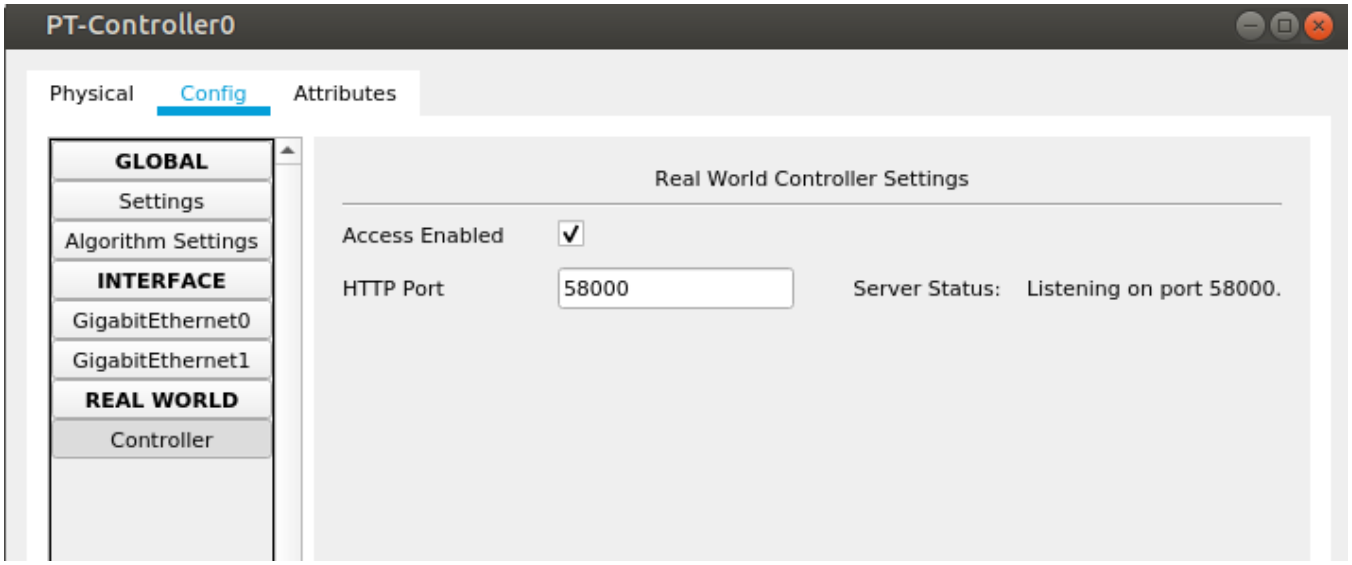

PT-Controller0
Physical
Config
Attributes
GLOBAL
Settings
Algorithm Settings
INTERFACE
GigabitEthernet0
GigabitEthernet1
REAL WORLD
Controller
Global Settings
Display Name
PT-Controller0
Interfaces
GigabitEthernet0
Gateway/DNS IPv4
DHCP
Static
Gateway
192.168.101.1
DNS Server
Gateway/DNS IPv6
DHCP
Auto Config
Static


PT-Controller0
Physical
Config
Attributes
GLOBAL
Settings
Algorithm Settings
INTERFACE
GigabitEthernet0
GigabitEthernet1
REAL WORLD
Controller
GigabitEthernet0
Port Status
On
Bandwidth
1000 Mbps
100 Mbps
10 Mbps
Auto
Duplex
Half Duplex
Full Duplex
Auto
MAC Address
000D.BD24.EC01
IP Configuration
DHCP
Static
IP Address
192.168.101.254
Subnet Mask
255.255.255.0
IPv6 Configuration
DHCP
Auto Config
Static
IPv6 Address
Link Local Address:
FE80::20D:BDFF:FE24:EC01

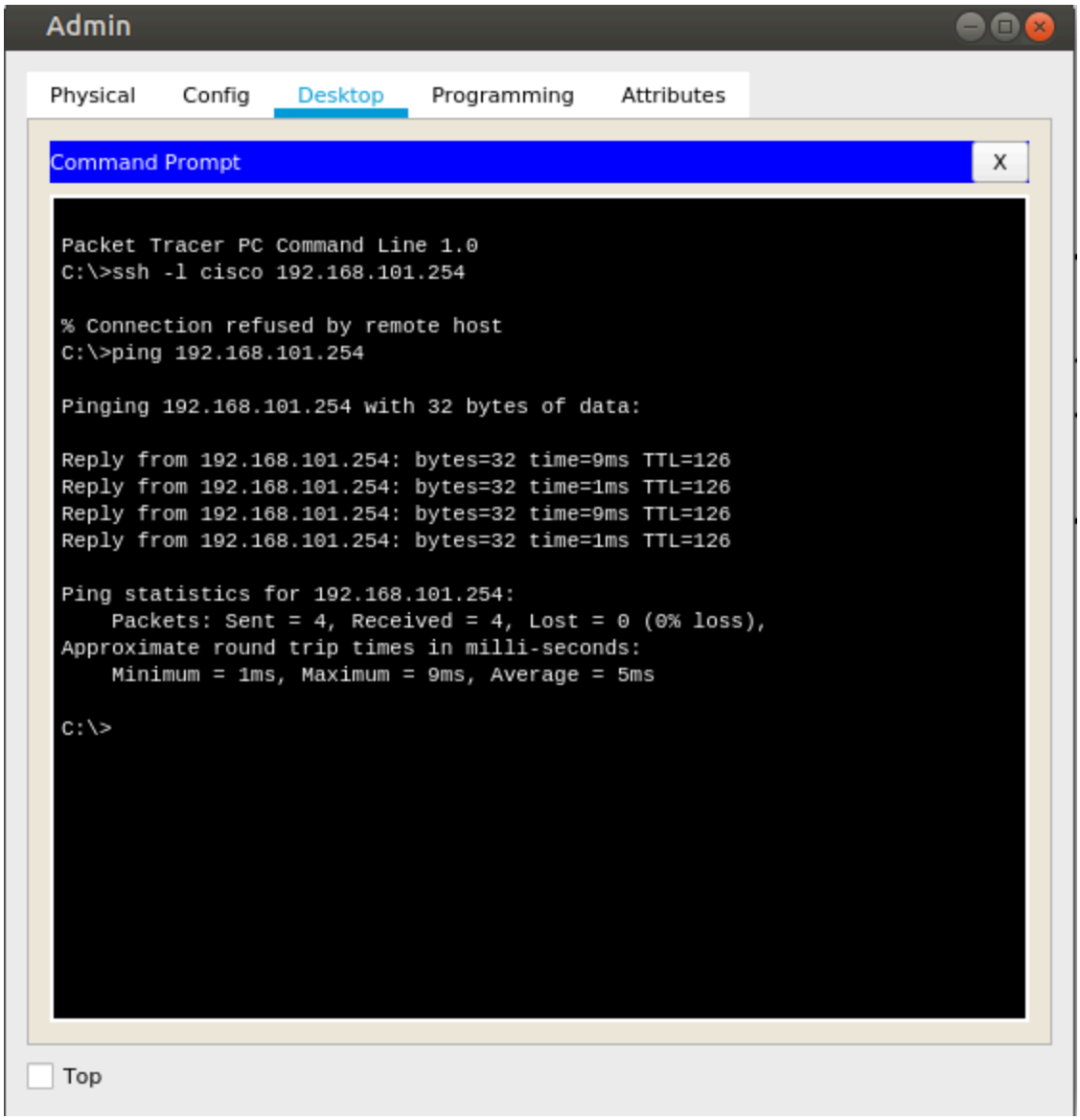
Verficamos



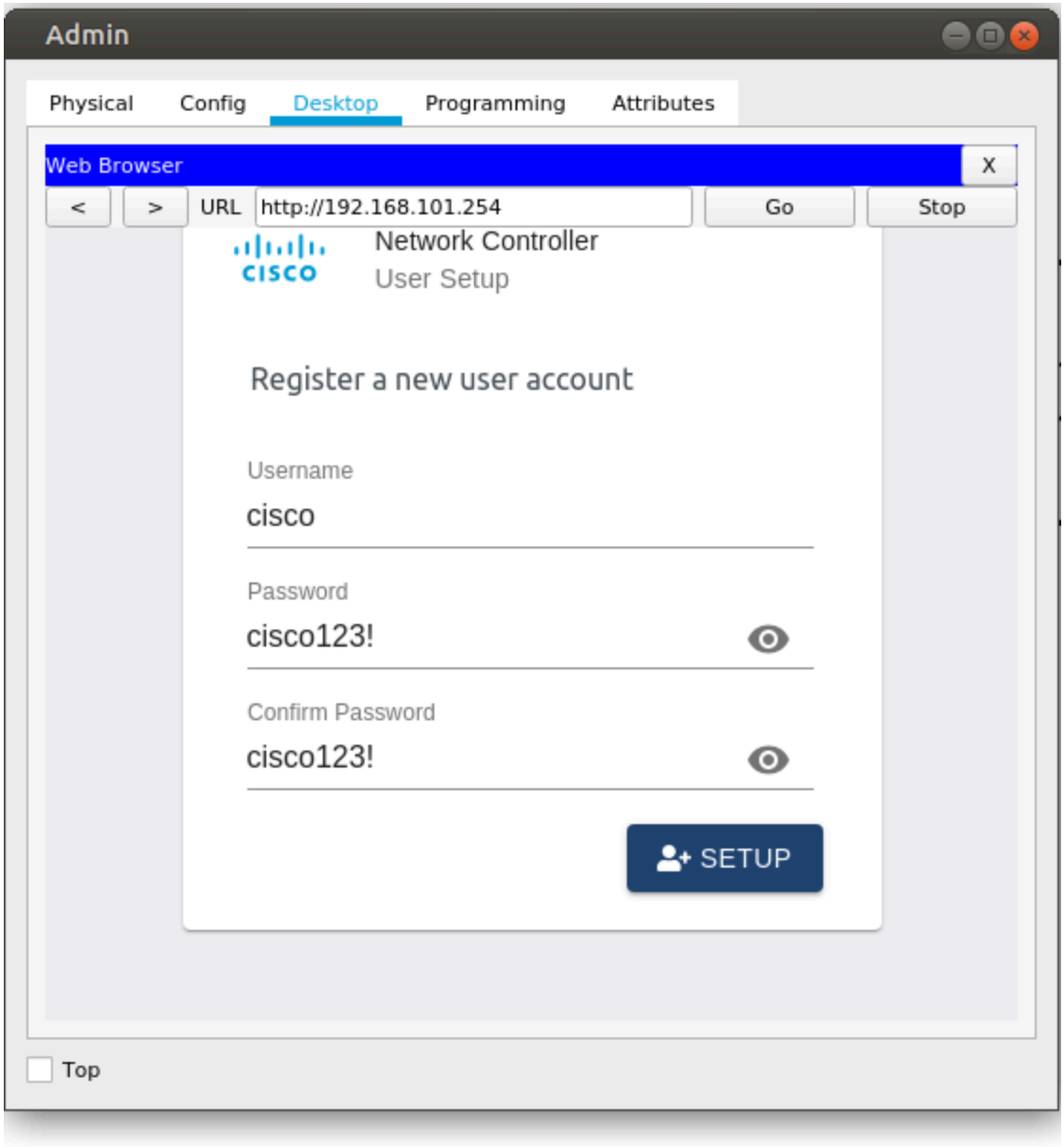
HAbilitamos el acceso a nuestro network controller:

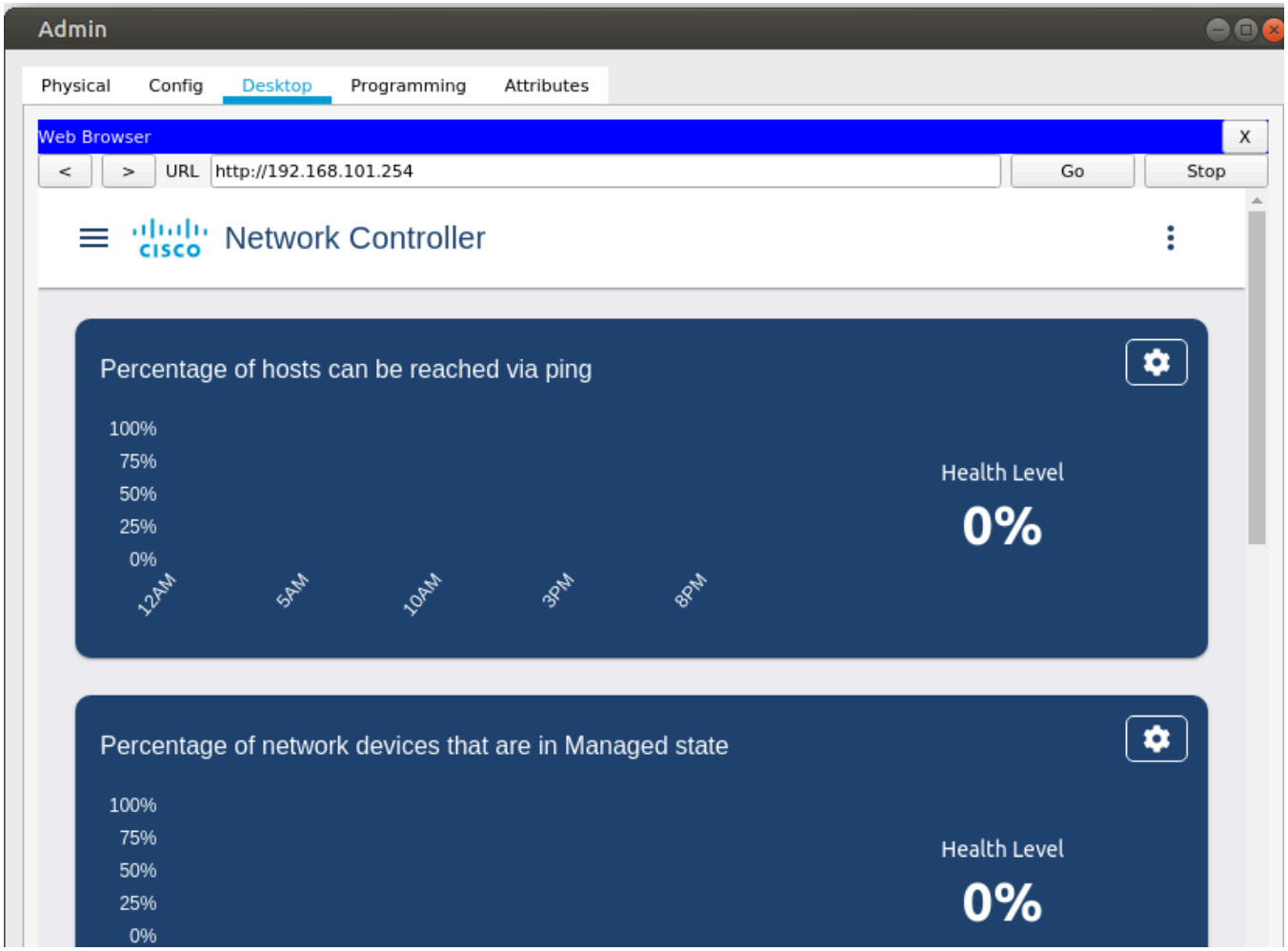


Desde Admin verificamos la conectividad a nuestro NetworkController



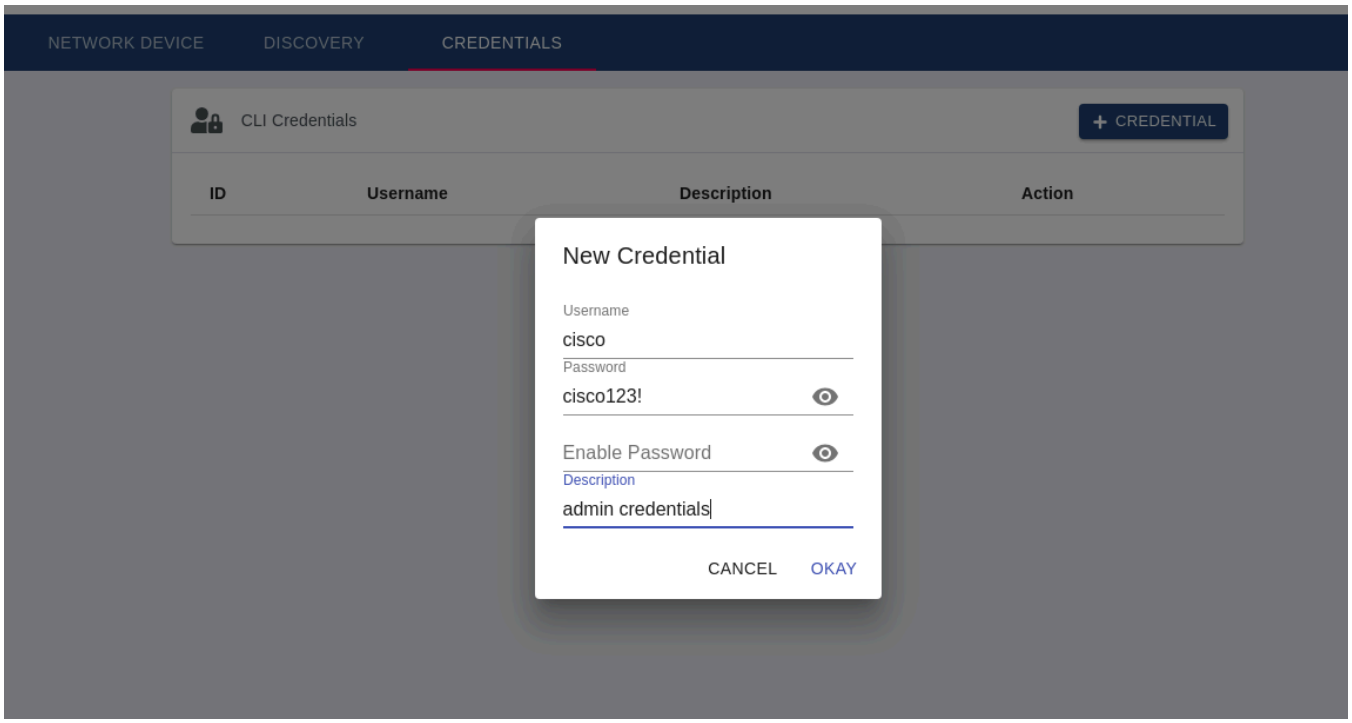
Nos registramos e iniciamos sesion dentro del PT-Controller O.


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
cisco
Password
cisco123!
Confirm Password
cisco123!
SETUP
Top

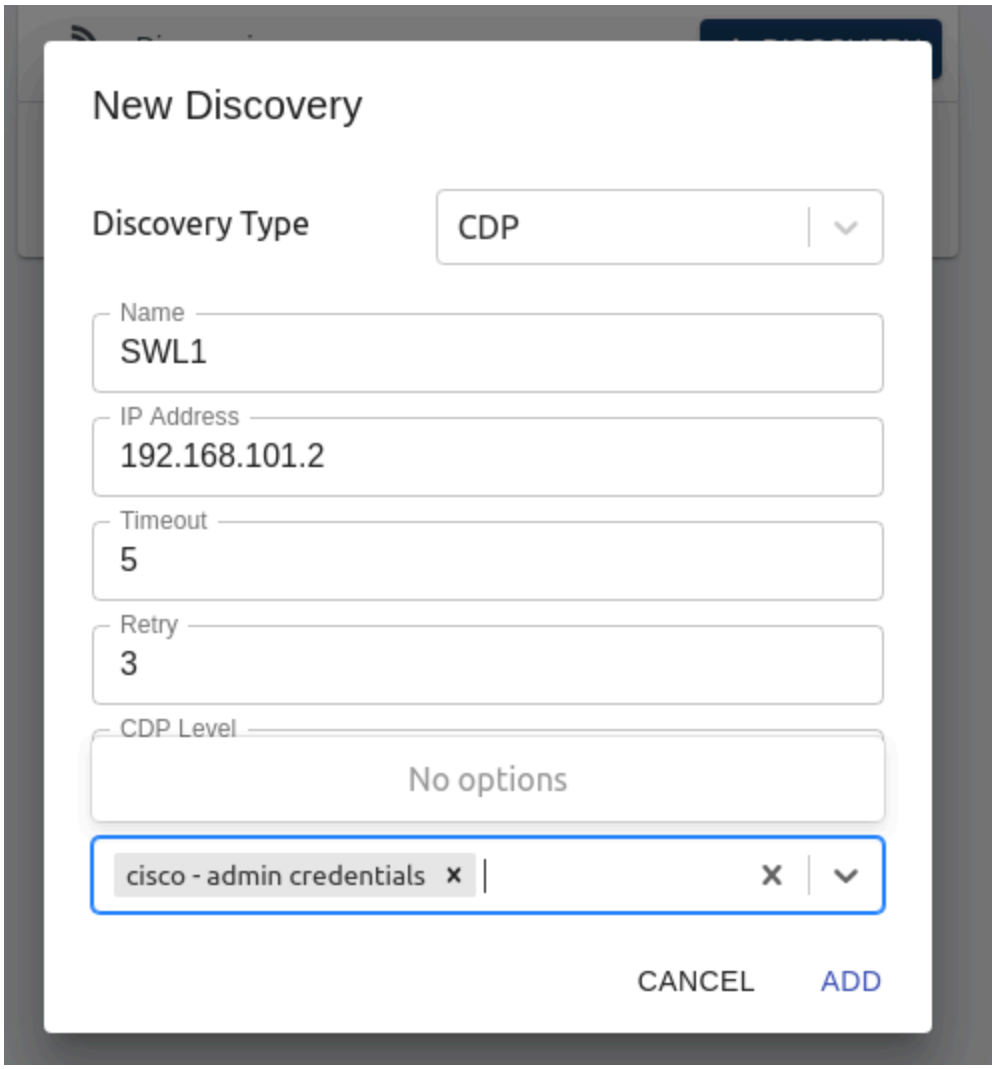


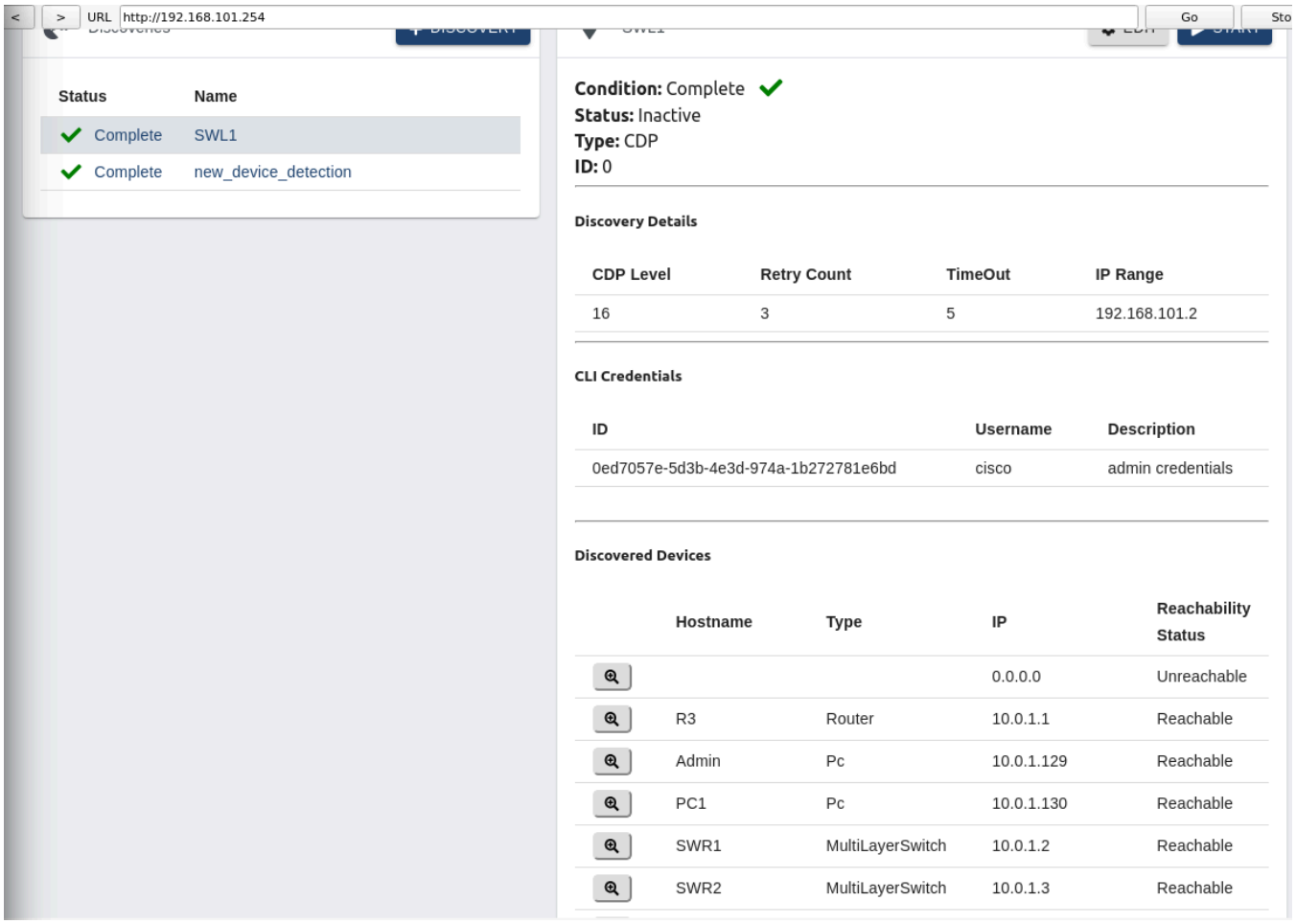
Usamos un SDN controller para descubrir una topología

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



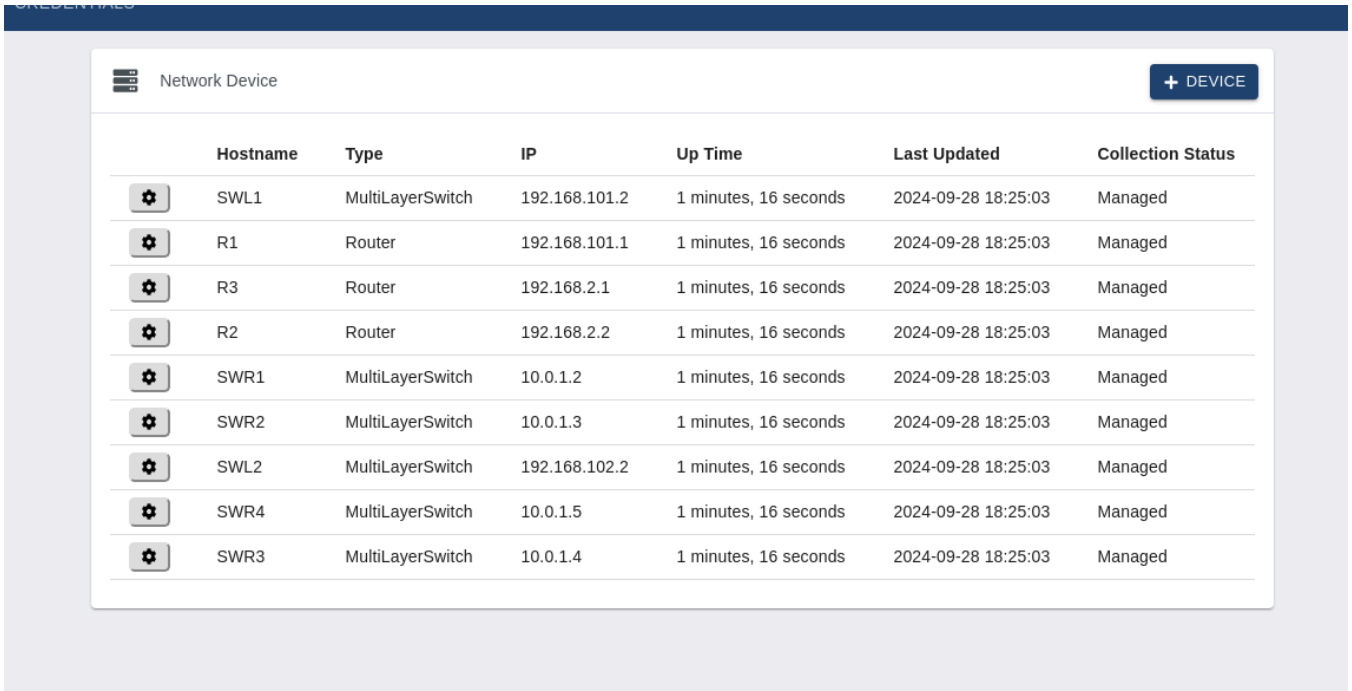
Usamos el CDP para decubrir los dispositivos


New Discovery
Discovery Type
CDP
Name
SWL1
IP Address
192.168.101.2
Timeout
5
Retry
3
CDP Level
No options
cisco - admin credentials
CANCEL
ADD

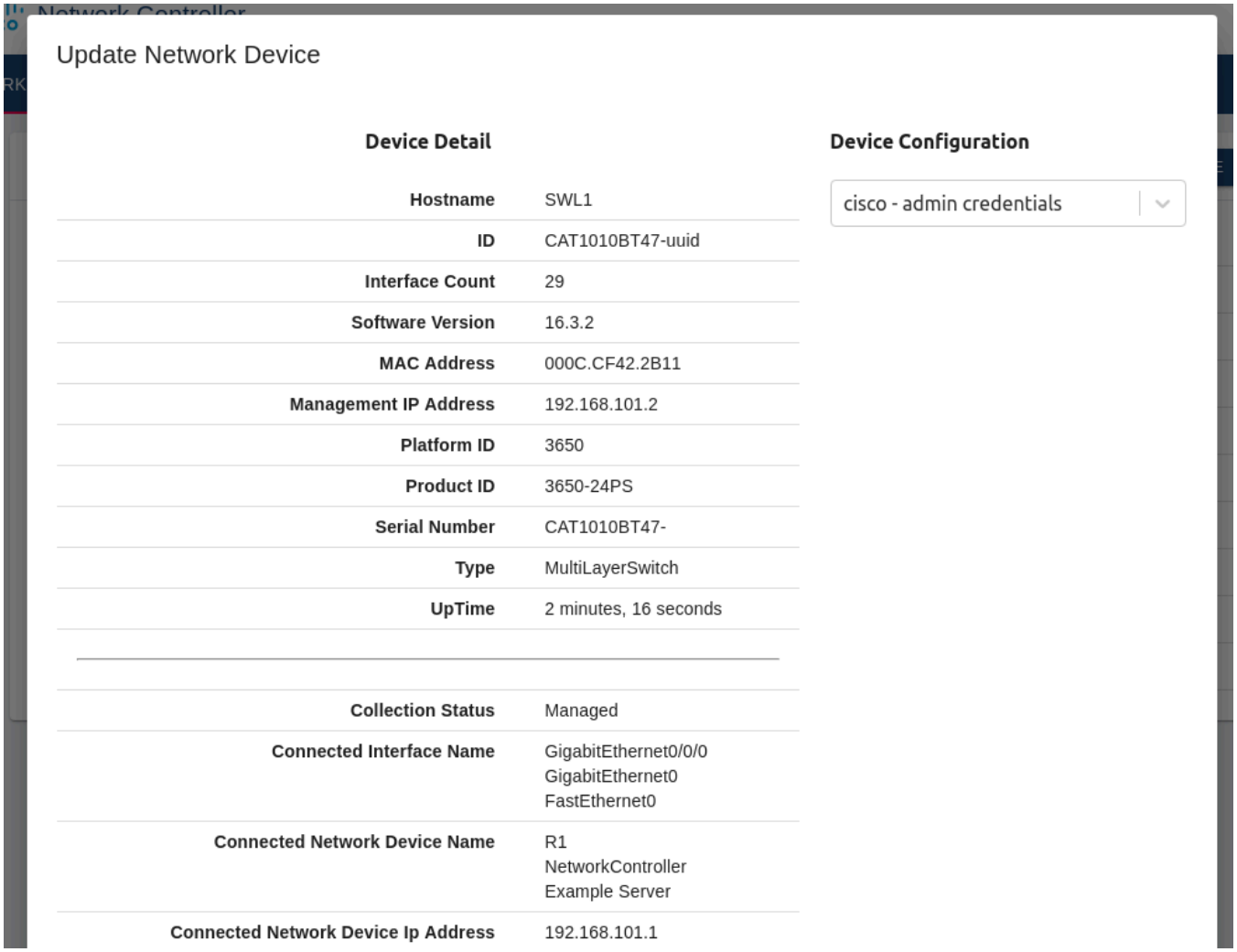


Usamos un SDN para recopilar información

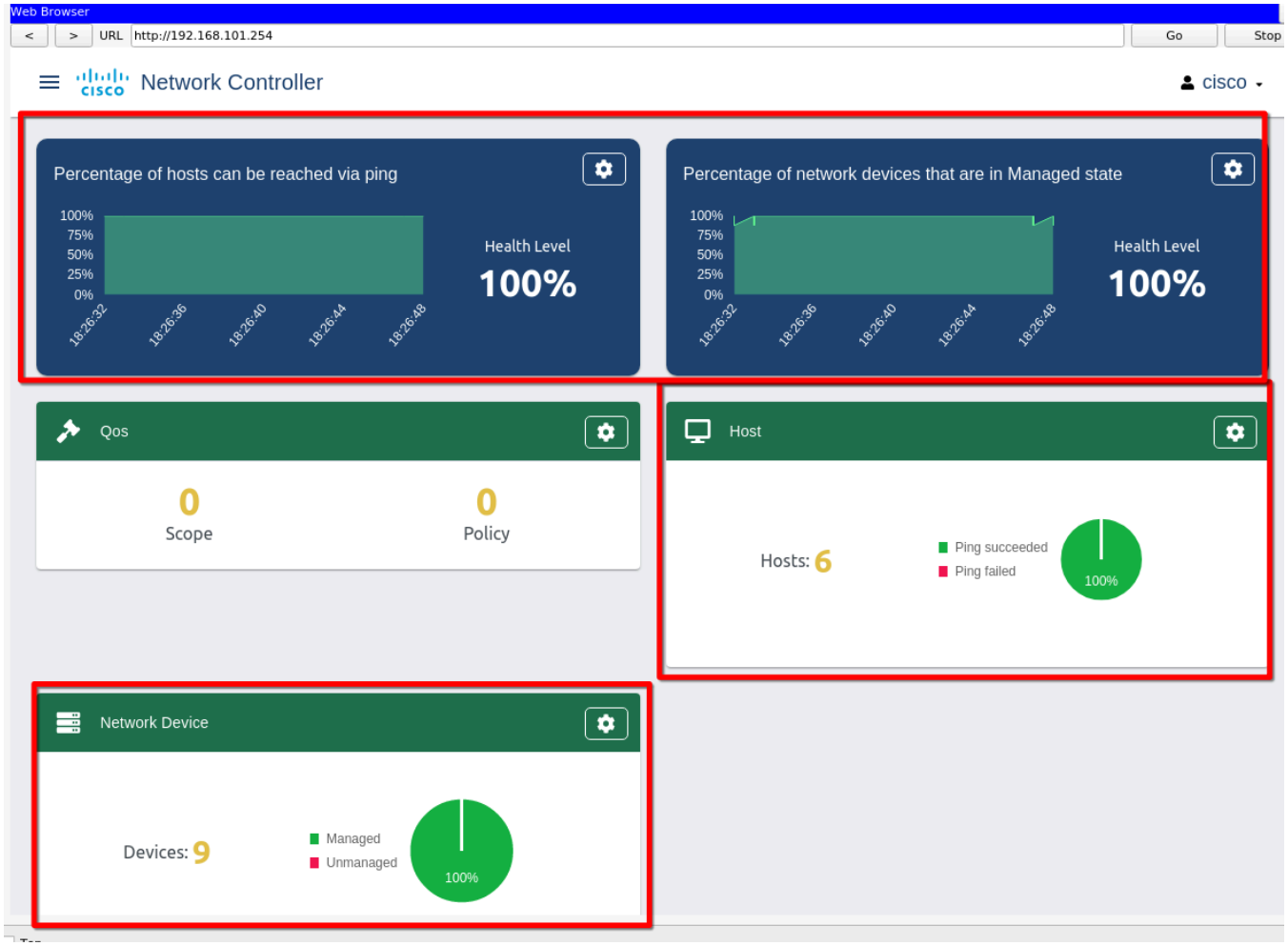
Veamos todos los dispotivos de red descubiertos



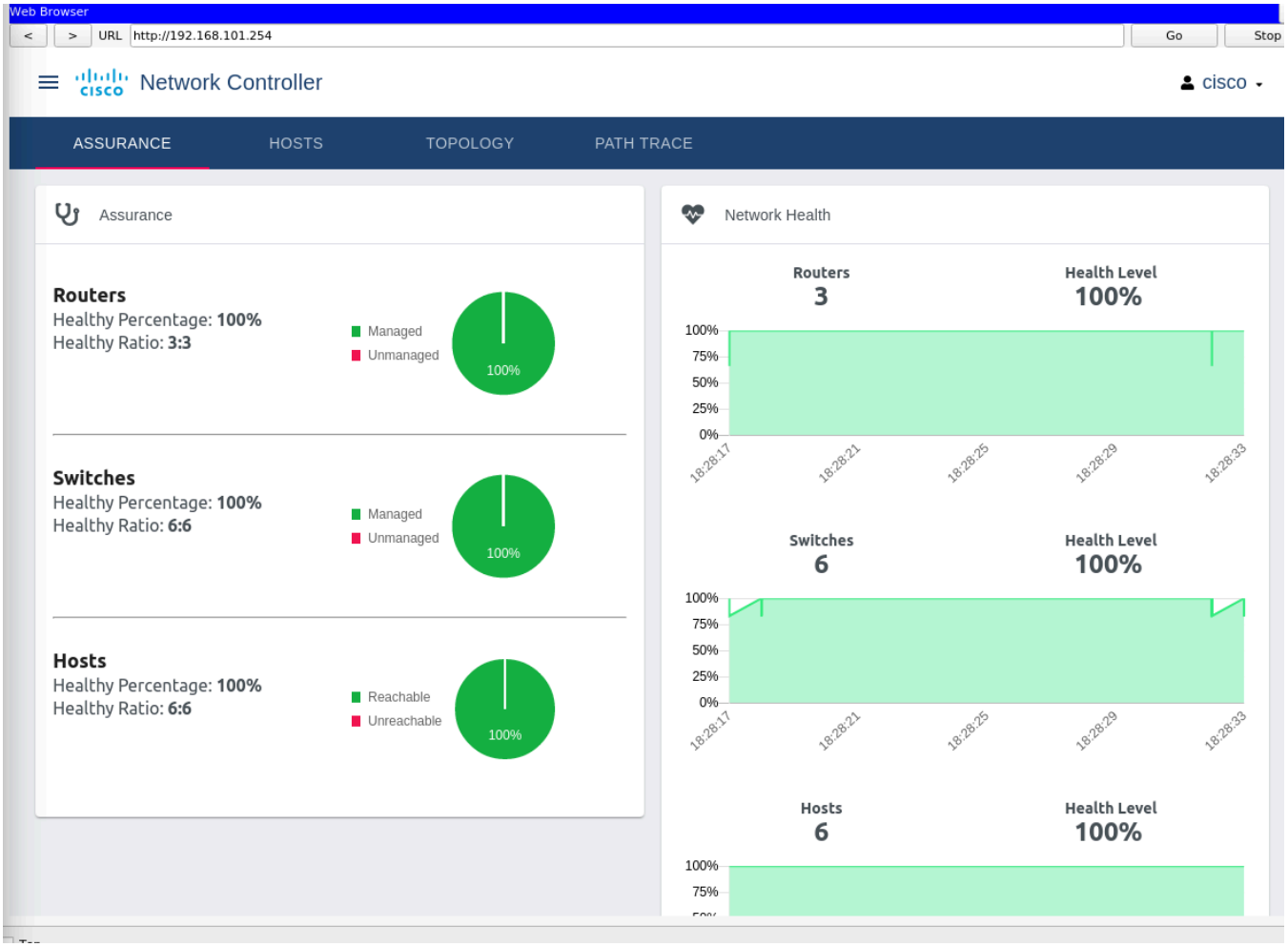
Vemos que al dar click en el boton de la izquierda, recopilamos la ifnormacion de tal dispositivo



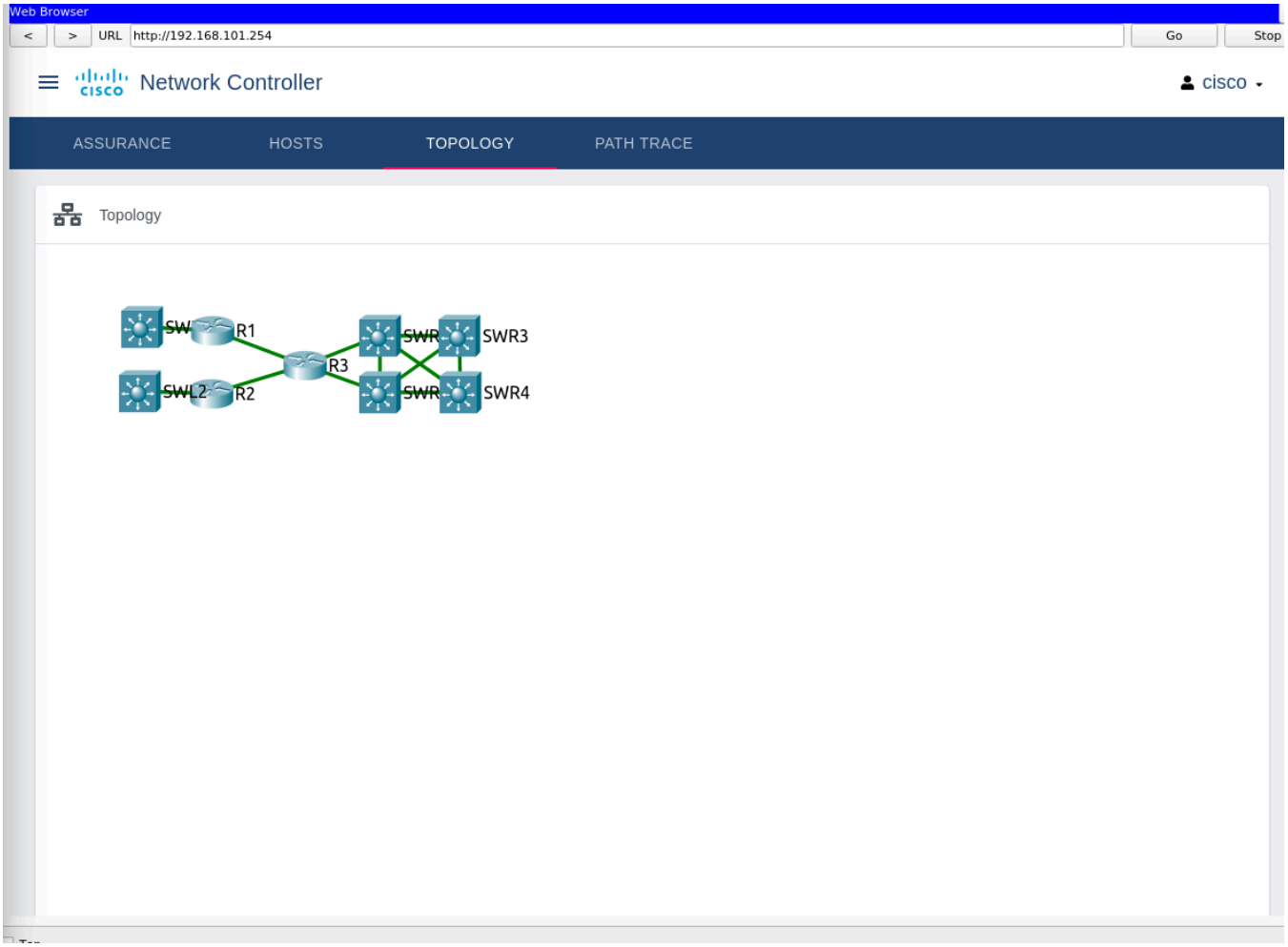
Vamos al dashboard



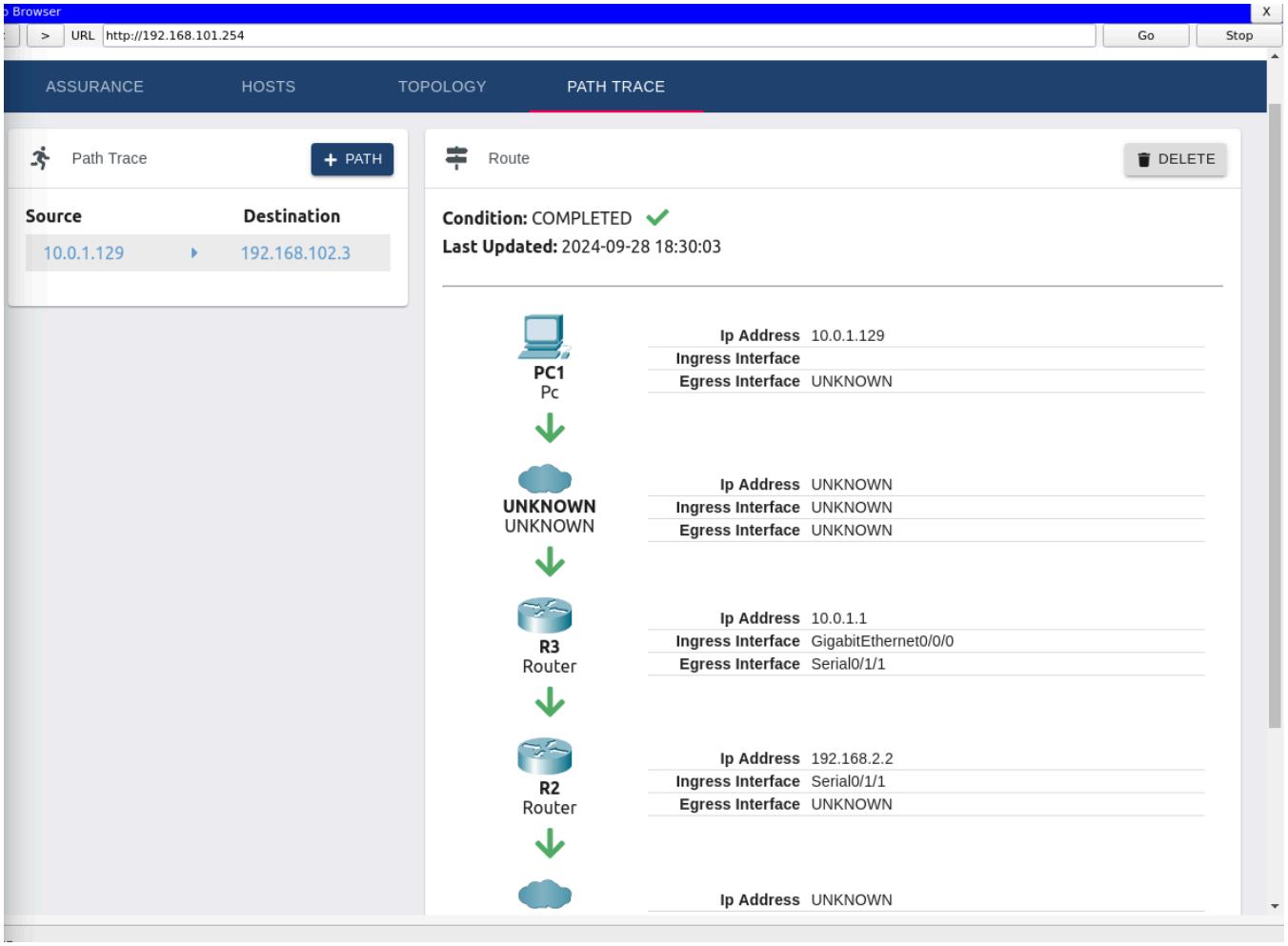
Al Assurance



Vemos la toppología cread por el Controlador

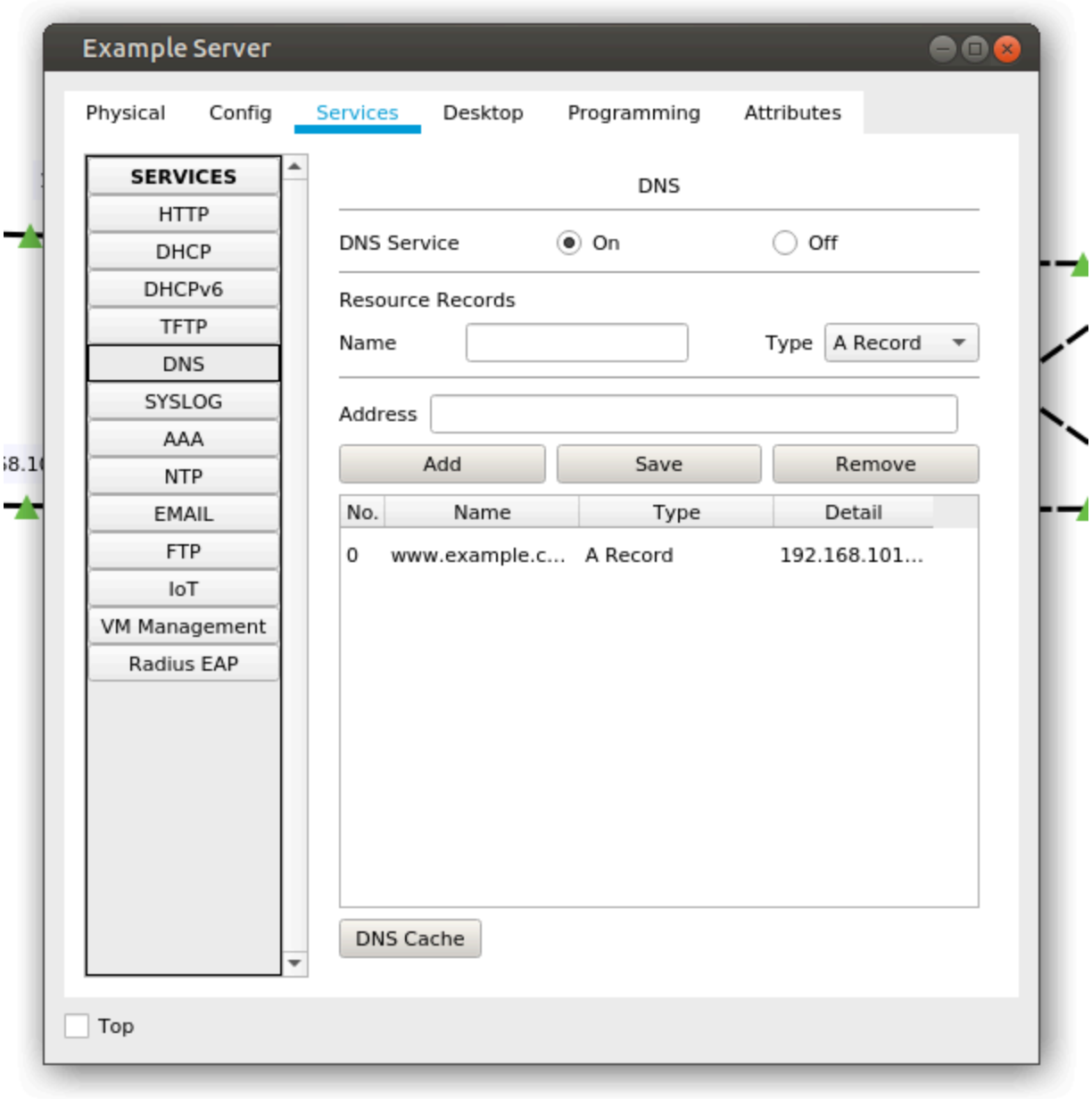


También podemos ver el PATH entre cualquier dispotivos por ejemplo entre PC-1 y PC-4

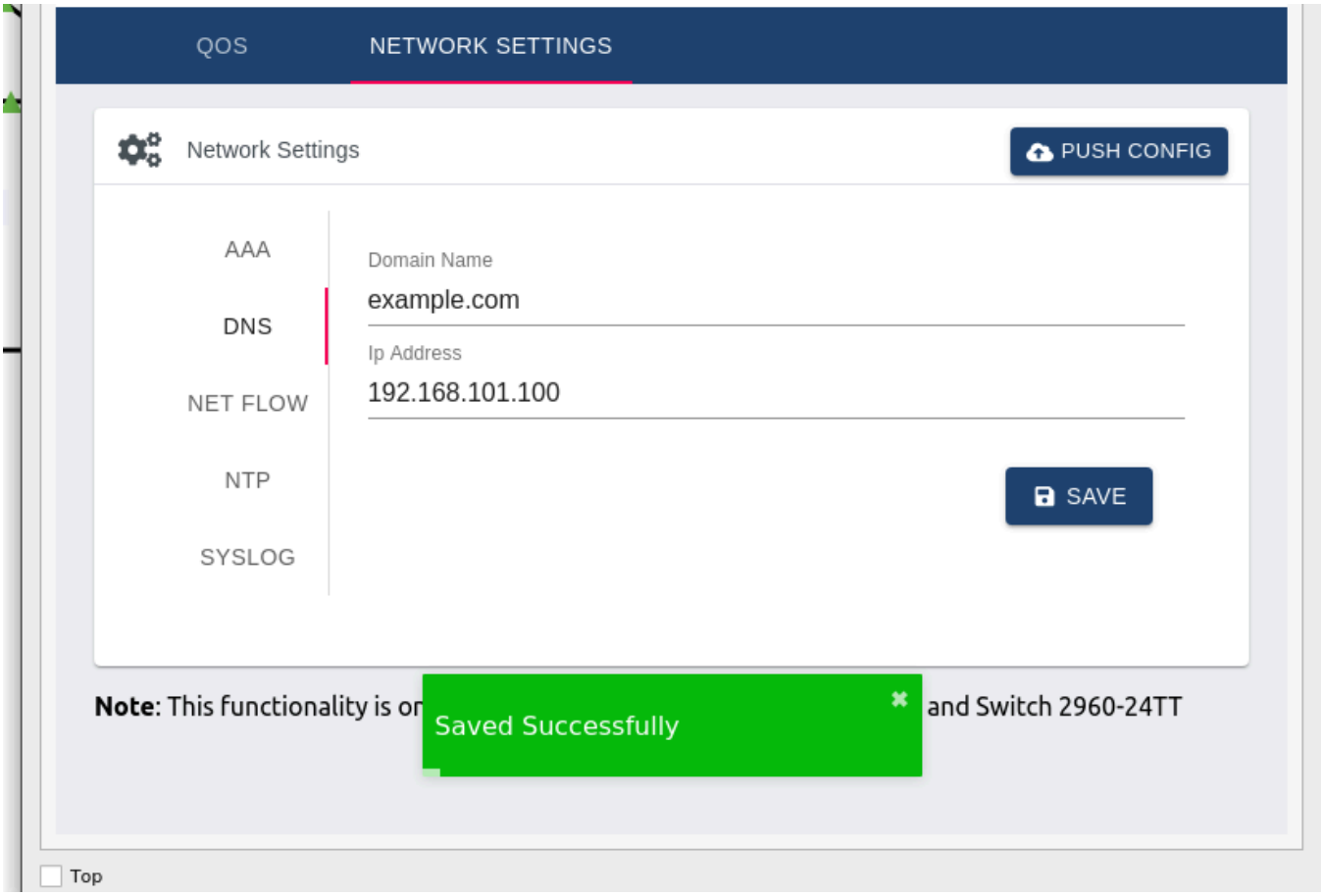


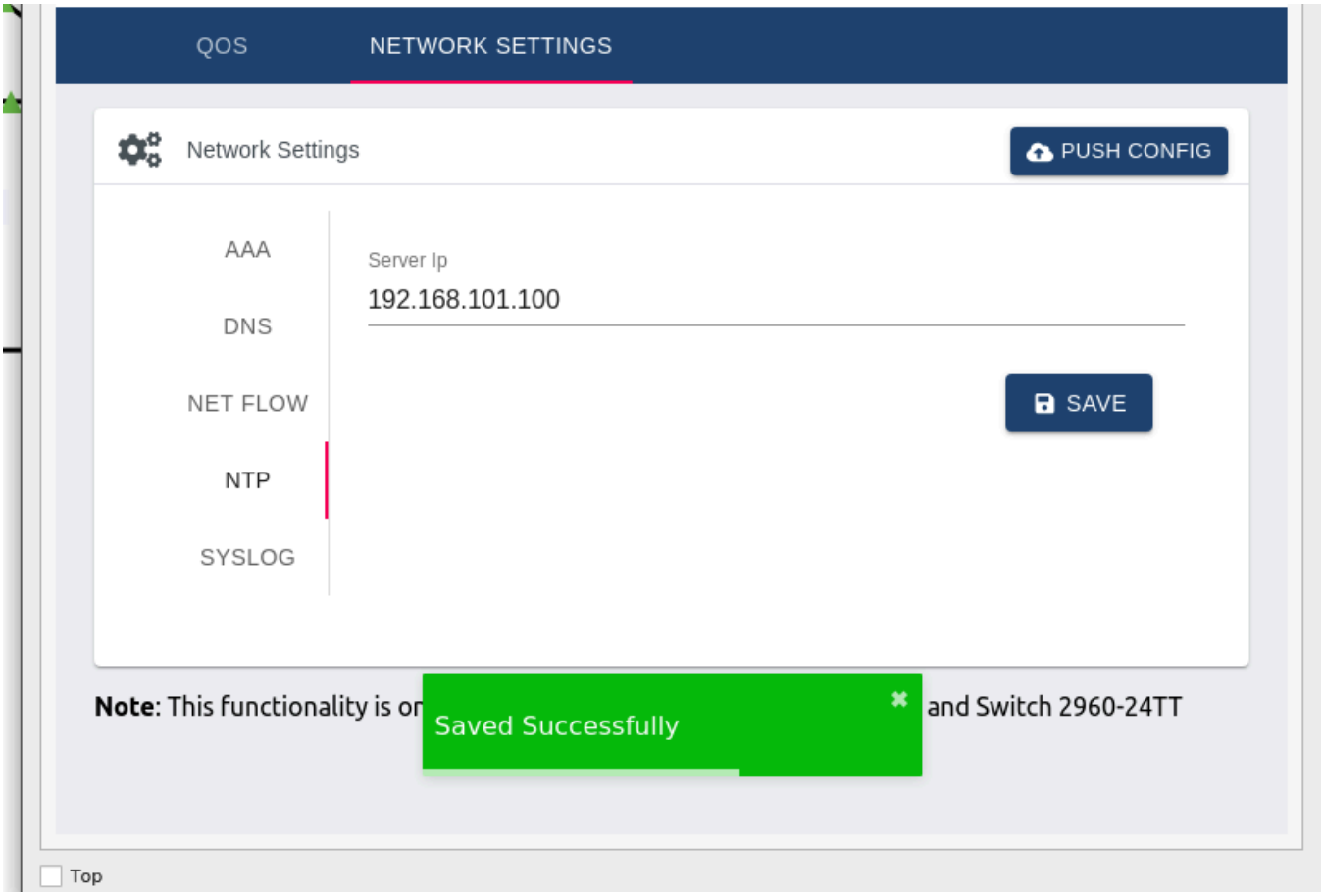
Usamo un SDN controller para configurar la configuración de red

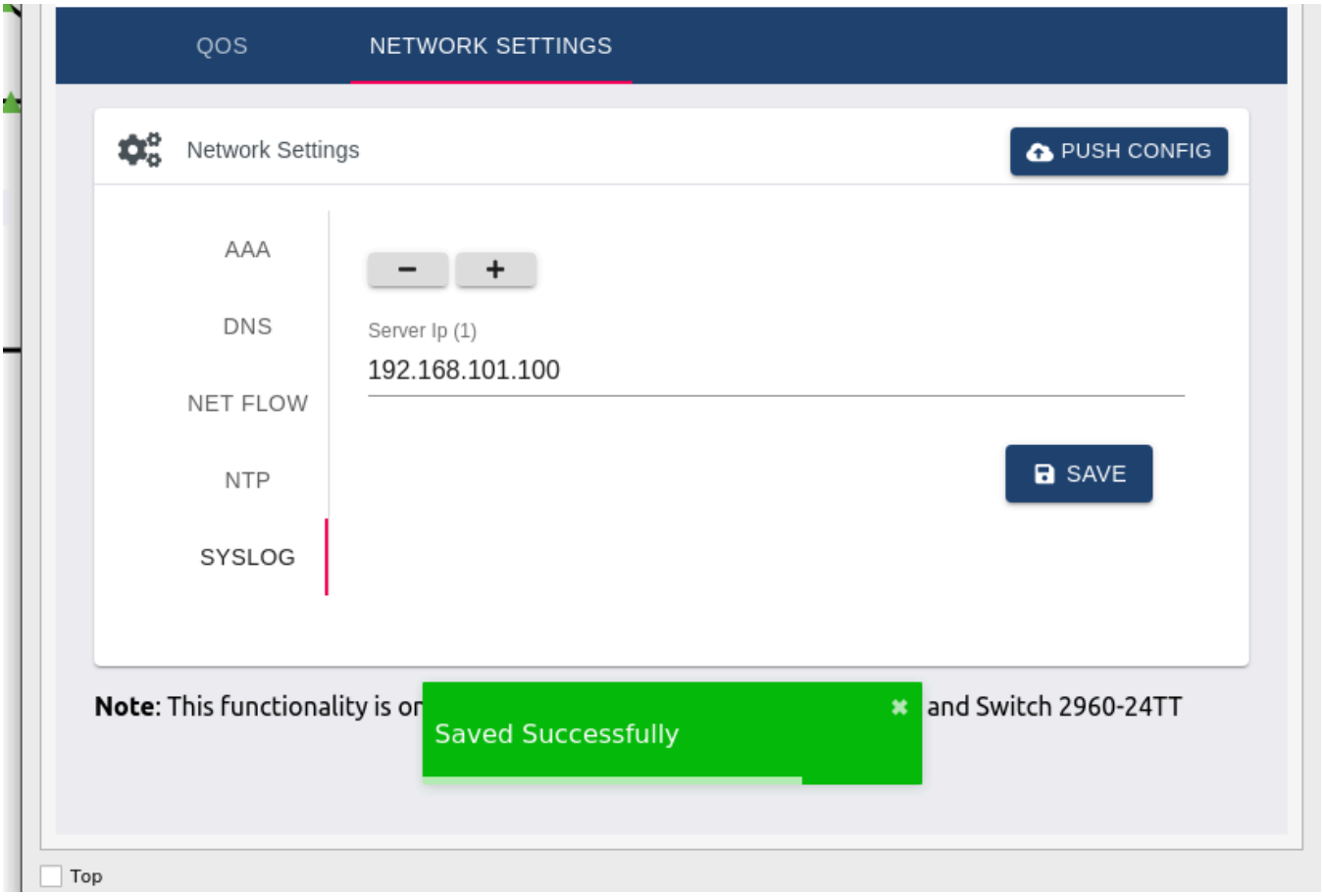
Ivestiagmos la configuración de Example server y vemos sus servicios habilitados

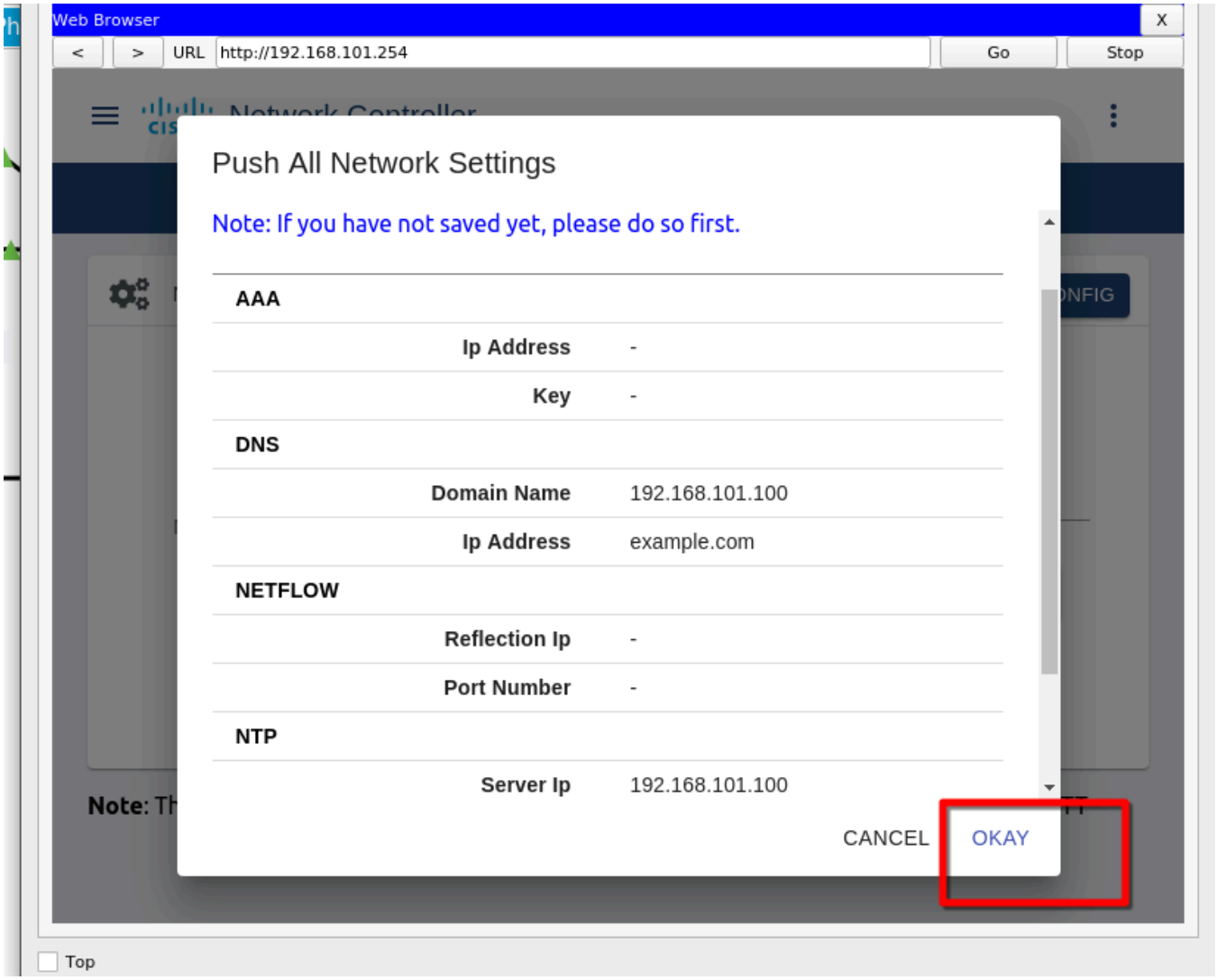


Configuramosa una global policy para DNS, SYSLOG, y NTP.


QOS
NETWORK SETTINGS
Network Settings
PUSH CONFIG
AAA
DNS
NET FLOW
NTP
SYSLOG
Domain Name
example.com
Ip Address
192.168.101.100
SAVE
Note: This functionality is or
Saved Successfully
and Switch 2960-24TT
Top


QOS
NETWORK SETTINGS
Network Settings
PUSH CONFIG
AAA
DNS
NET FLOW
NTP
SYSLOG
Server Ip
192.168.101.100
SAVE
Note: This functionality is or
Saved Successfully
and Switch 2960-24TT
Top


QOS
NETWORK SETTINGS
Network Settings
PUSH CONFIG
AAA
DNS
NET FLOW
NTP
SYSLOG
Server Ip (1)
192.168.101.100
SAVE
Note: This functionality is or
Saved Successfully
and Switch 2960-24TT
Top



Verficamos 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

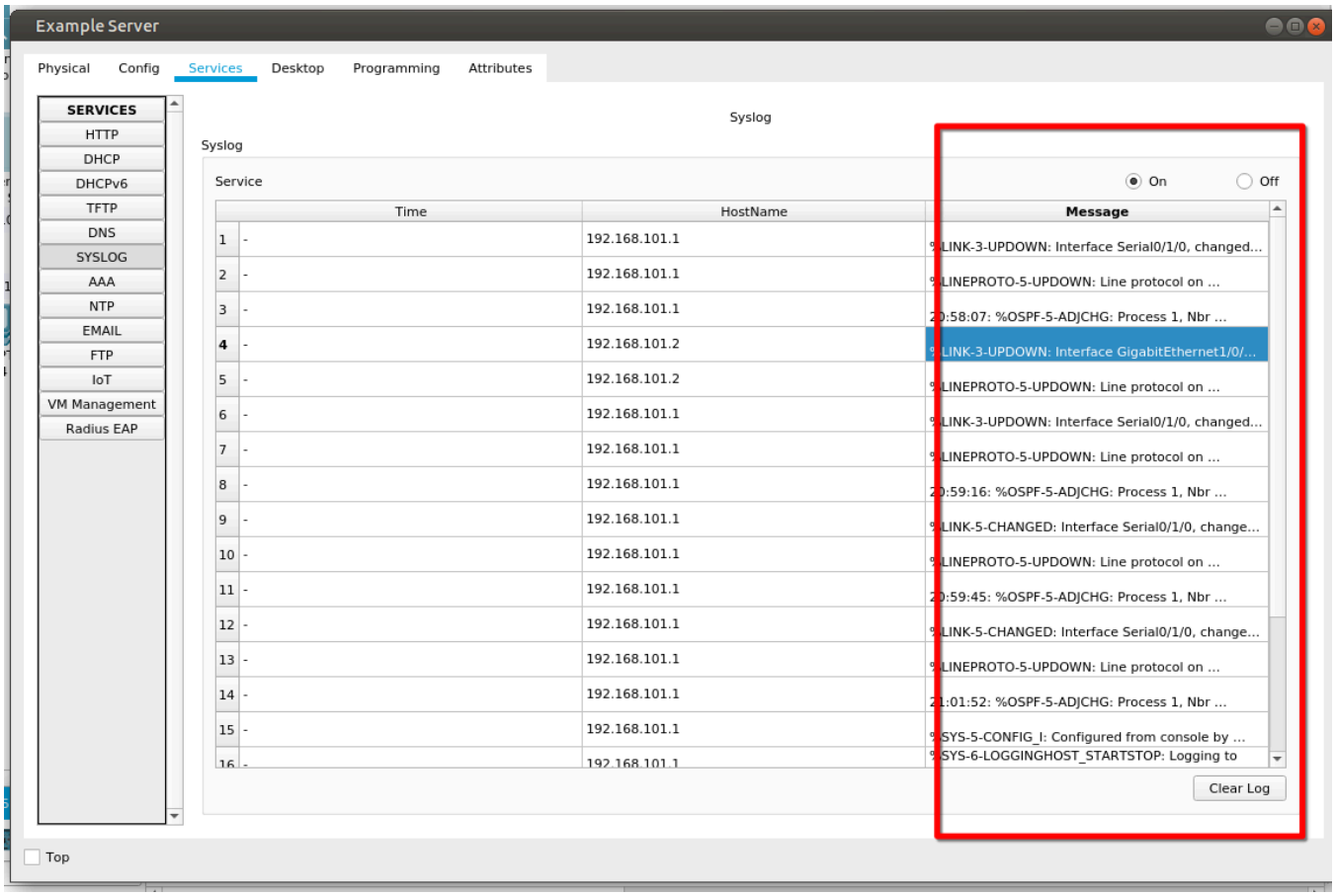
```
R1#show run | include logging
logging 192.168.101.100
```

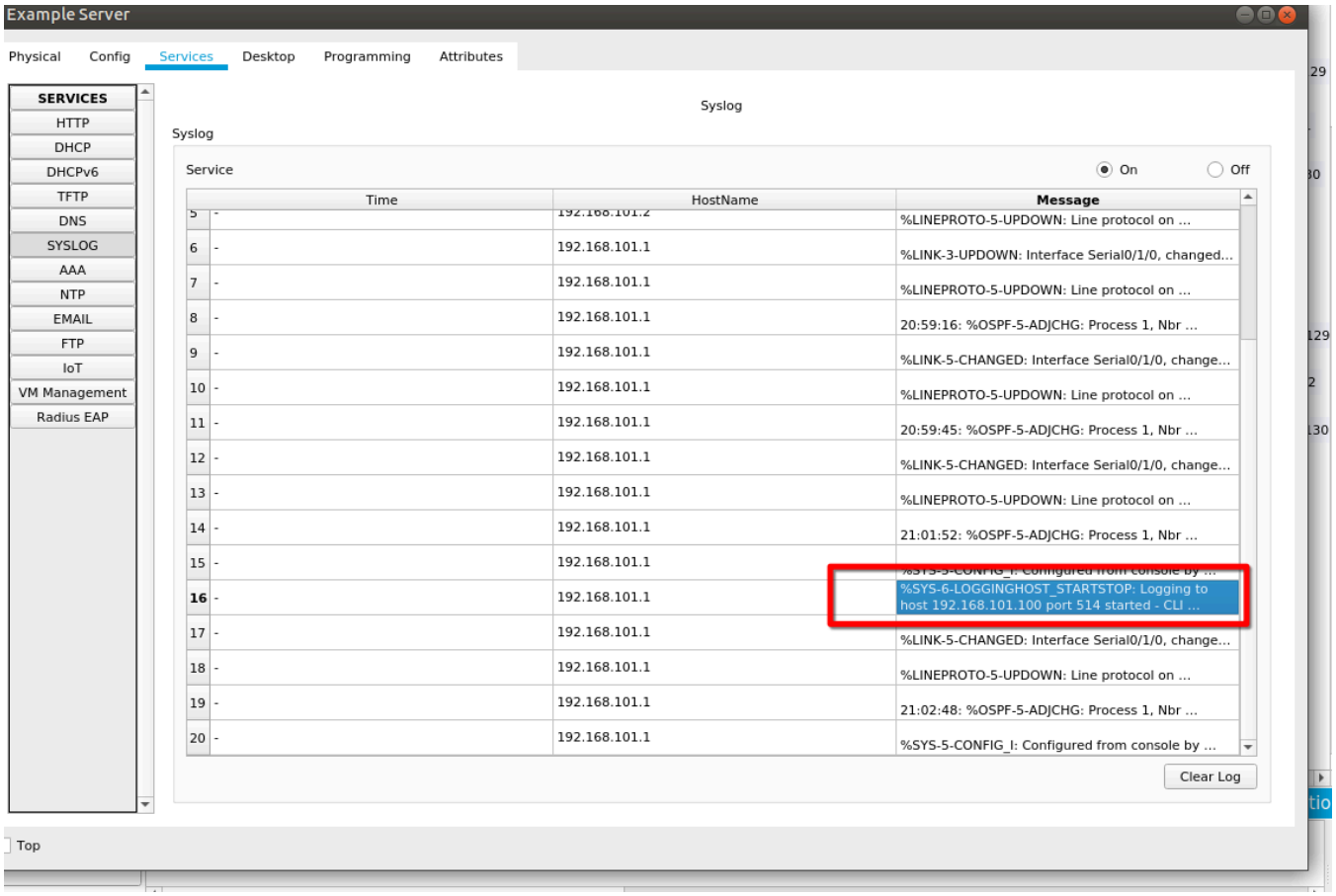
Finalmente testeamos este último.

```
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 administrativel
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
Syslog
Service
On
Off
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
Clear Log
Top

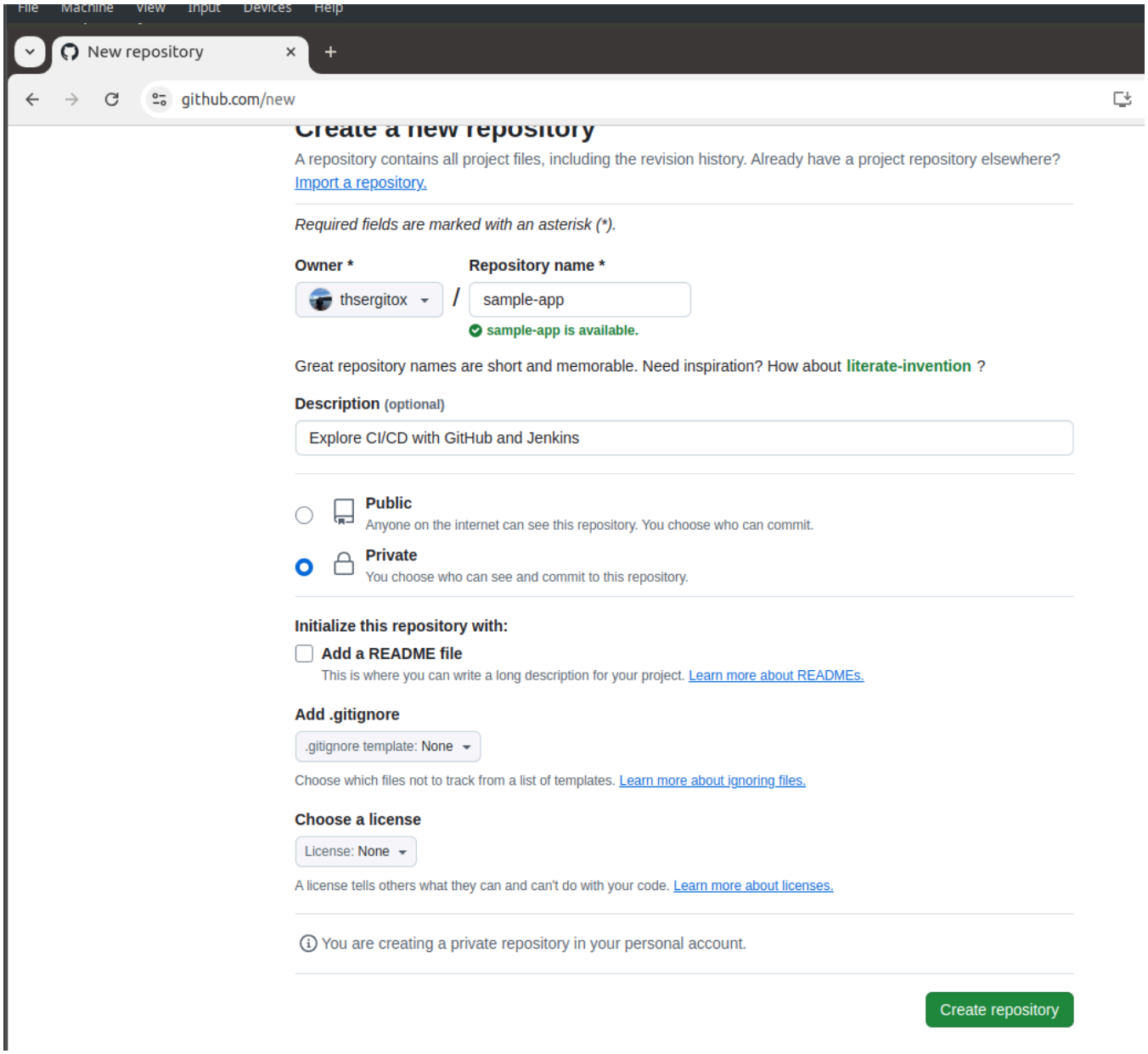


Parte 2: Construimos un CI/CD pipeline usando Jenkins

Inicializamos la VM de DEVASC.

Commiteamos el sample app a git

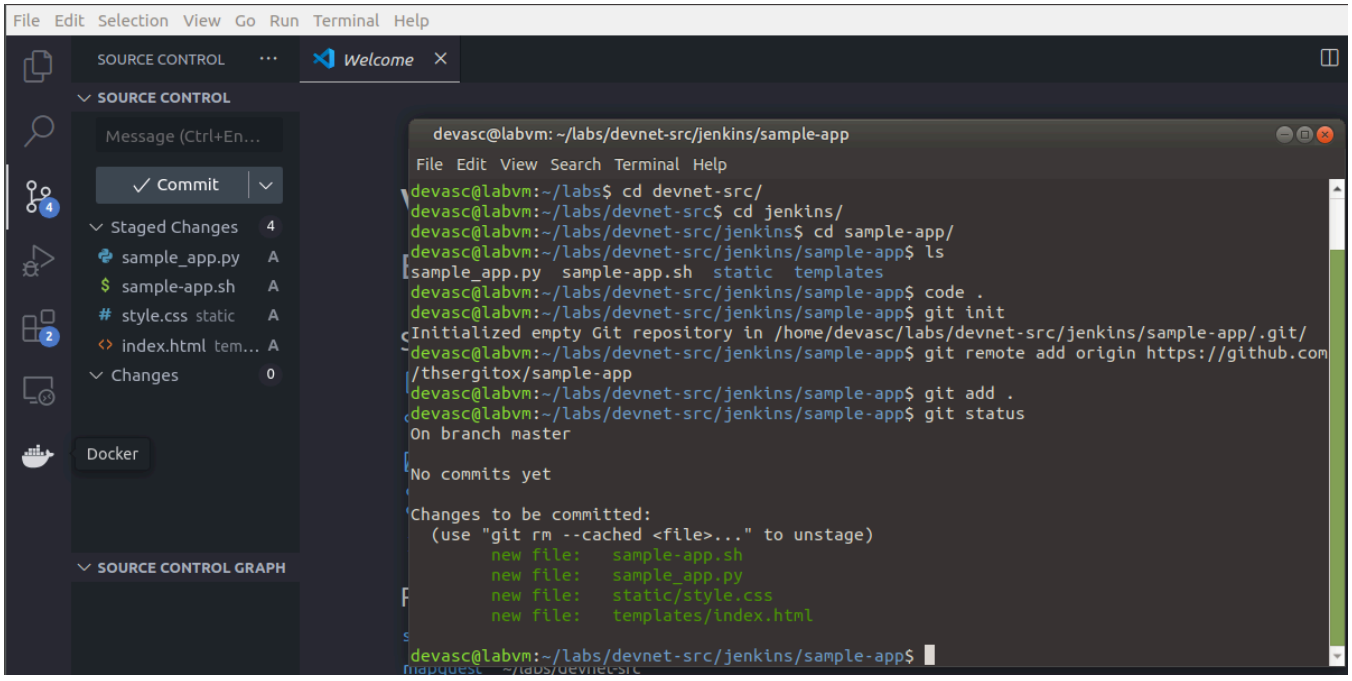
Creamos nuestro repositorio



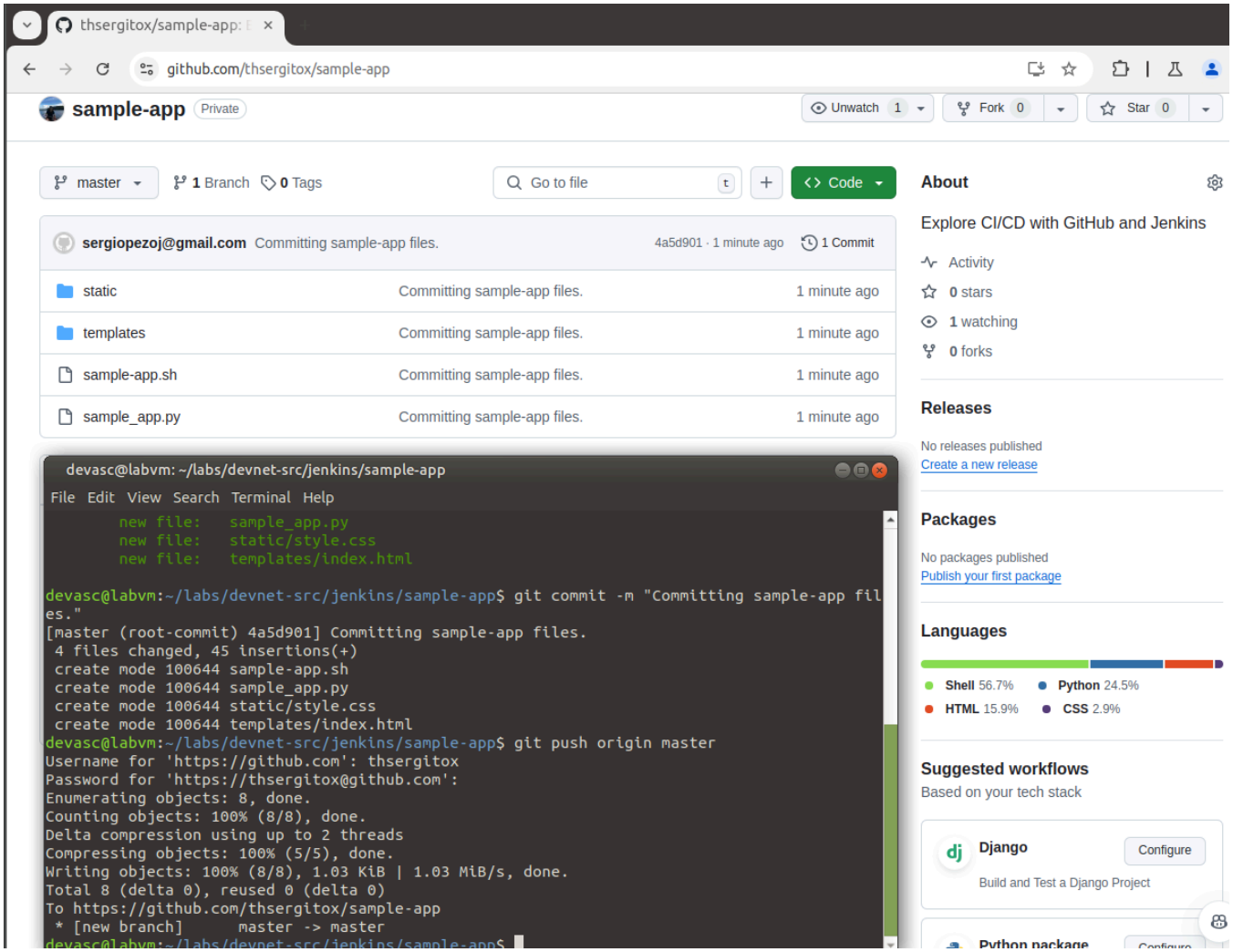
Configuramos nuestras credenciales en la VM

```
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 nuestro archivos al staged.

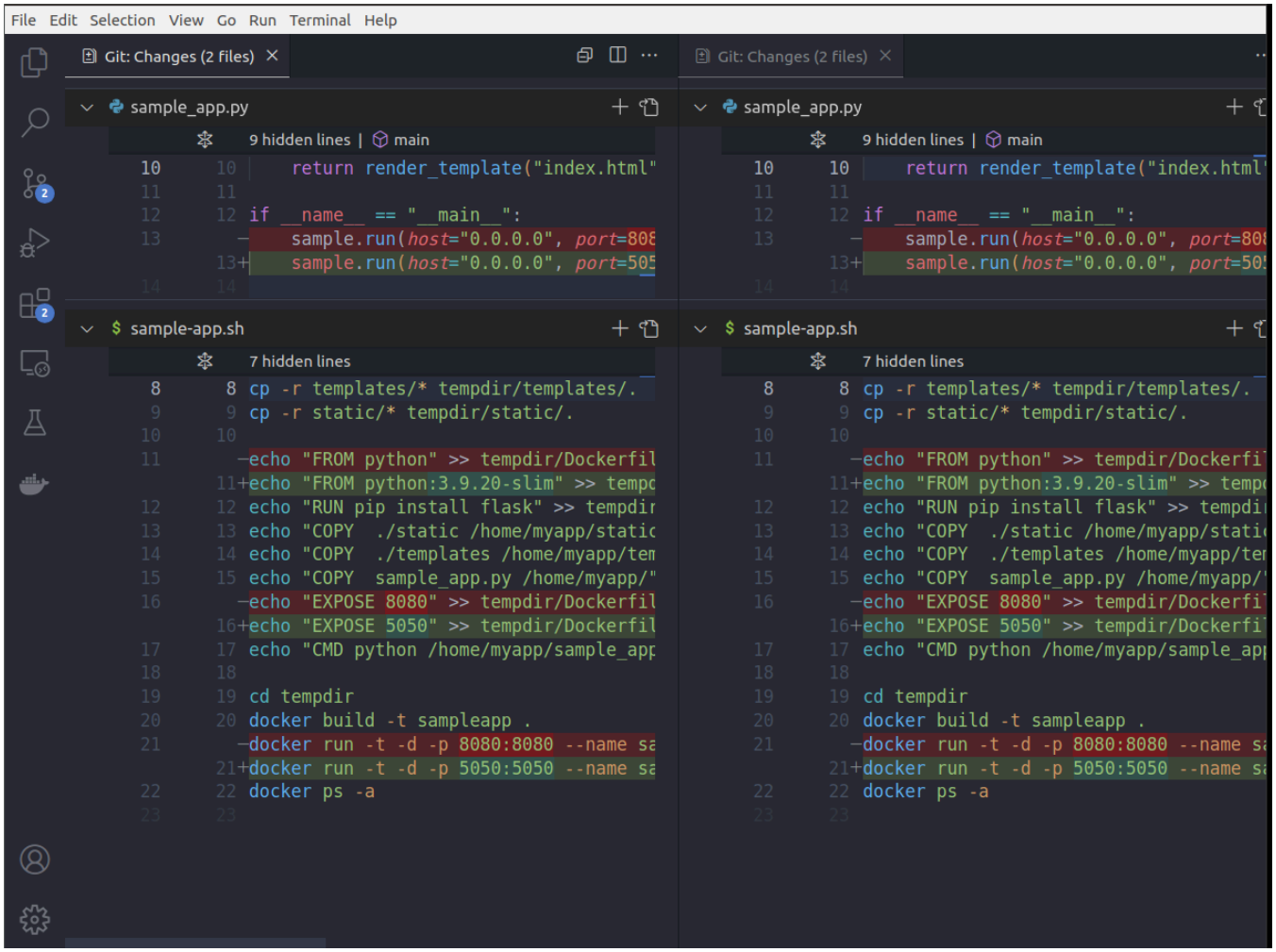


Finalmete lo pusheamos

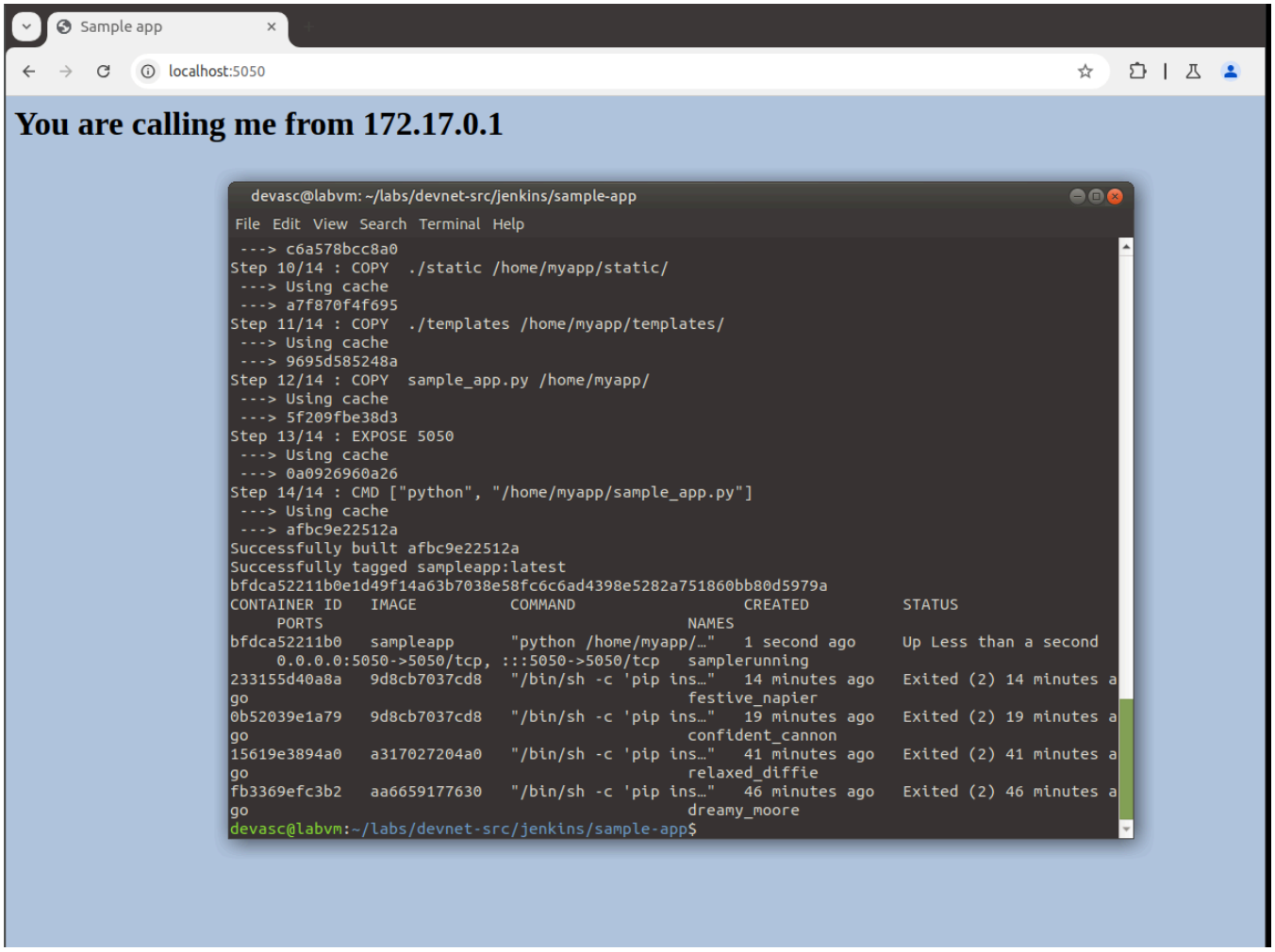


Modificamos los archivos para pushearlos de nuevo

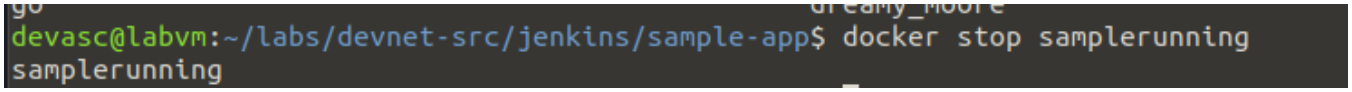
Cambiamos de puerto y versión de python



Construimos nuesra app con docker



Paramos el contenedor



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

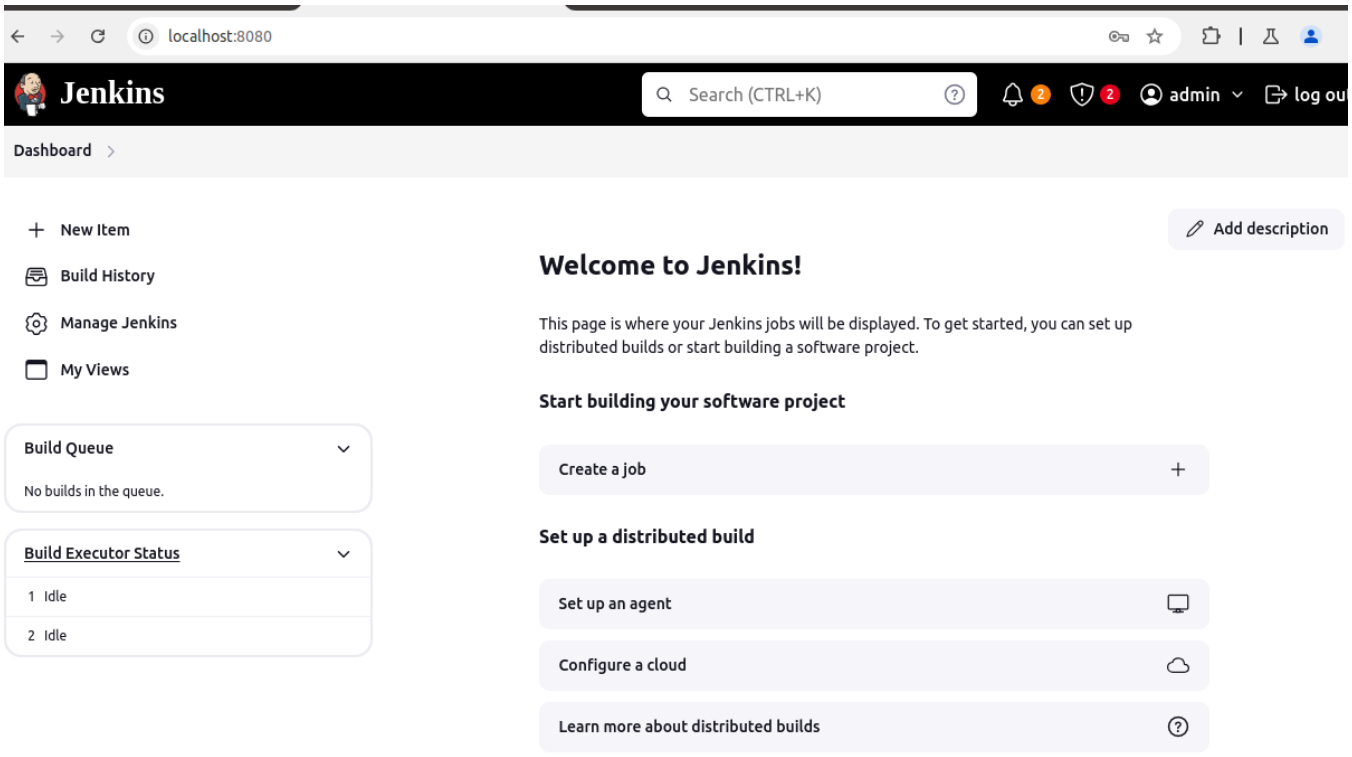
Traemos la imagen de la Docker Hub y la iniciamos, en específico la versión 2.414.3-slim-jdk17, pues mi cuenta de GitHub estaba configurada con SSH, y el plugin para Jenkins estaba para esa versión y todo genial de ahí.

```
devasc@labvm:~/labs/devnet-src/jenkins/sample-app$ docker pull jenkins/jenkins:2.271
2.271: Pulling from jenkins/jenkins
3192219afd04: Pull complete
17c160265e75: Pull complete
cc4fe40d0e61: Pull complete
9d647f502a07: Pull complete
d108b8c498aa: Pull complete
1bfe918b8aa5: Pull complete
dafa1a7c0751: Pull complete
a10933ea2f2f: Pull complete
dacec5718df4: Pull complete
0cd1192f374e: Pull complete
bac0875b818f: Pull complete
fc7126ecc5e0: Pull complete
65580027eff4: Pull complete
ea01a82194c6: Pull complete
be9da8492eef: Pull complete
8351e2eec838: Pull complete
909eab257f21: Pull complete
9107ef57f0b5: 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 $(which do
ker):/usr/bin/docker -v /var/run/docker.sock:/var/run/docker.sock -v "$HOME":/home --name jenkins_server jenkins/jenkins:2.271
Running from: /usr/share/jenkins/jenkins.war
webroot: EnvVars.masterEnvVars.get("JENKINS_HOME")
```

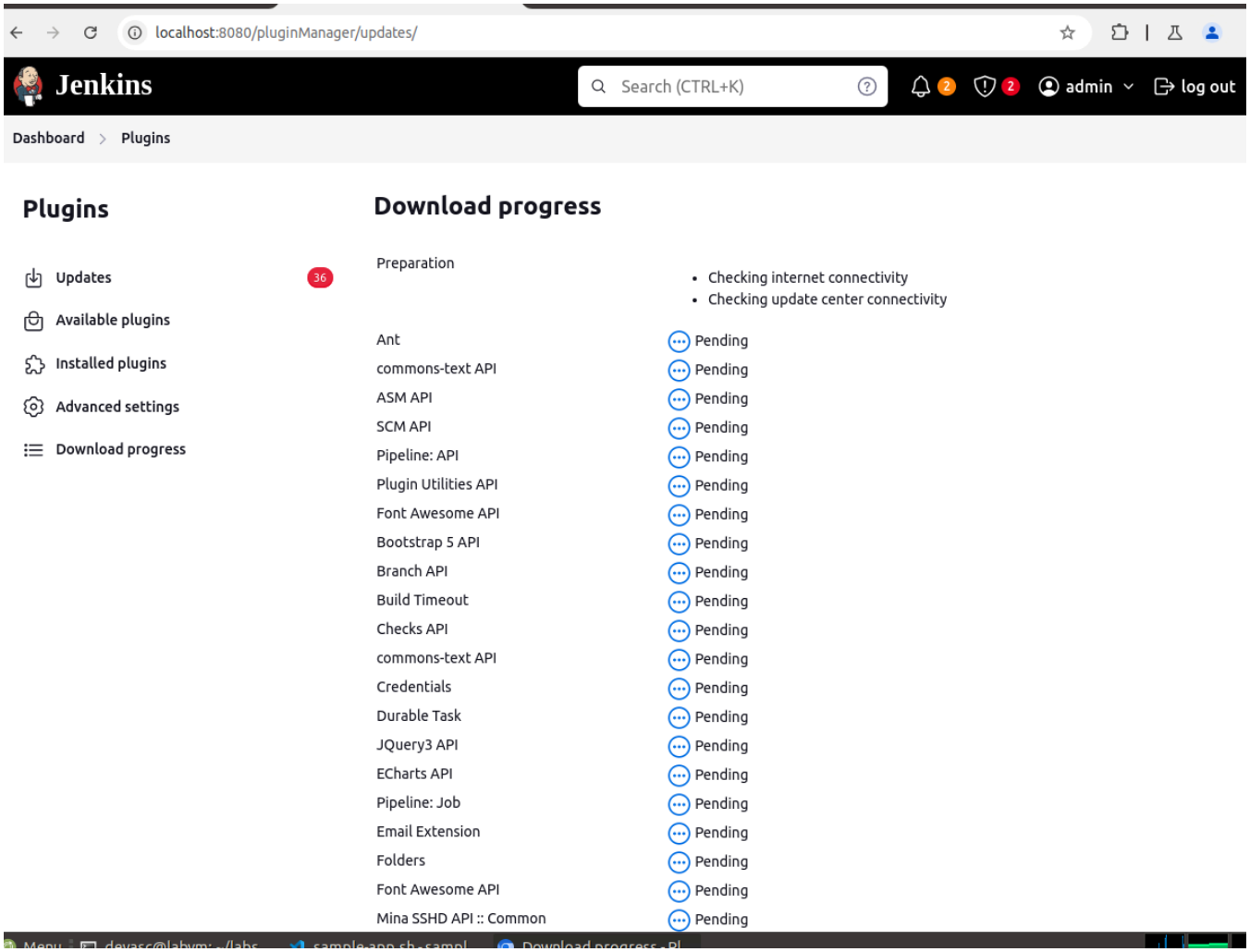
Verificamos

```
*************************************************************
*************************************************************

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.Retrier#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

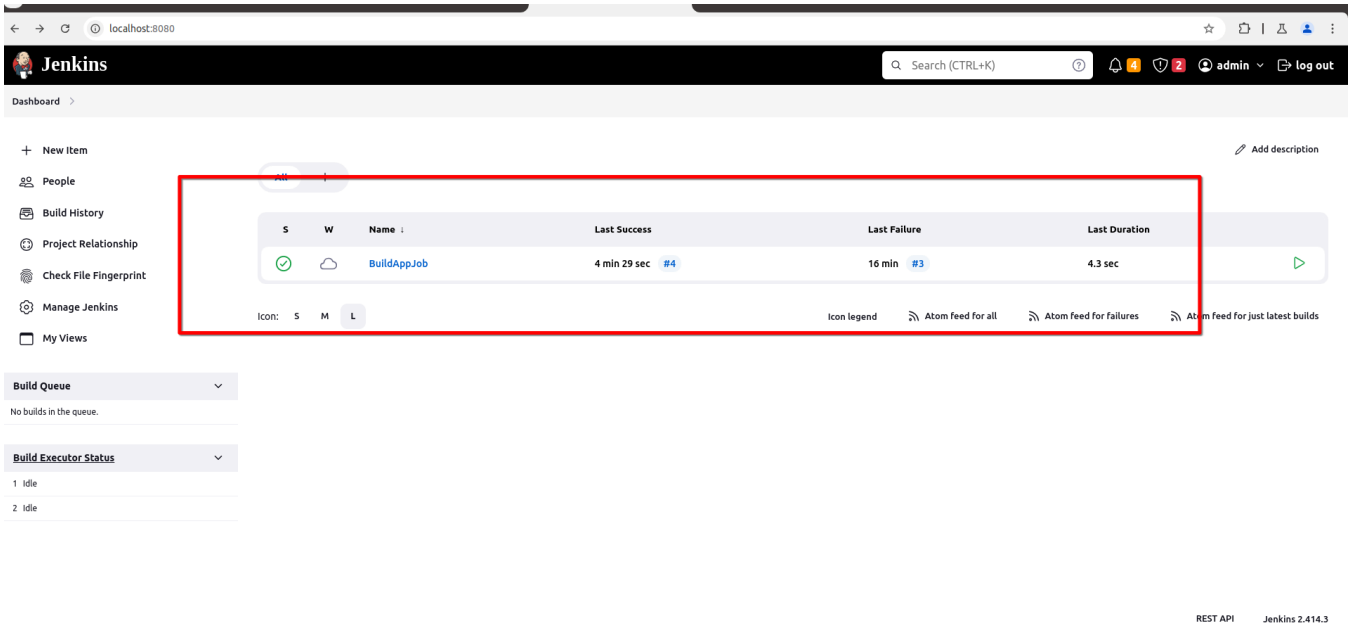


Descargamos todos los plugins recomendados

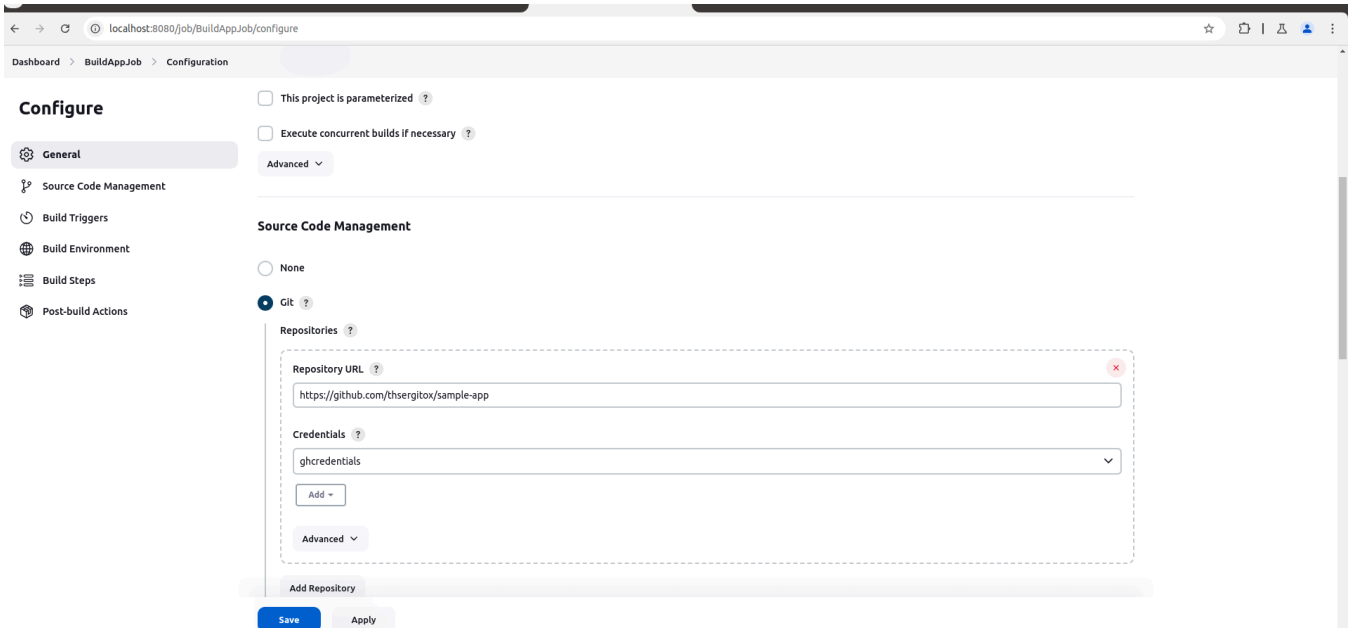


Usamos Jenkins para ejecutar un build de nuestra app

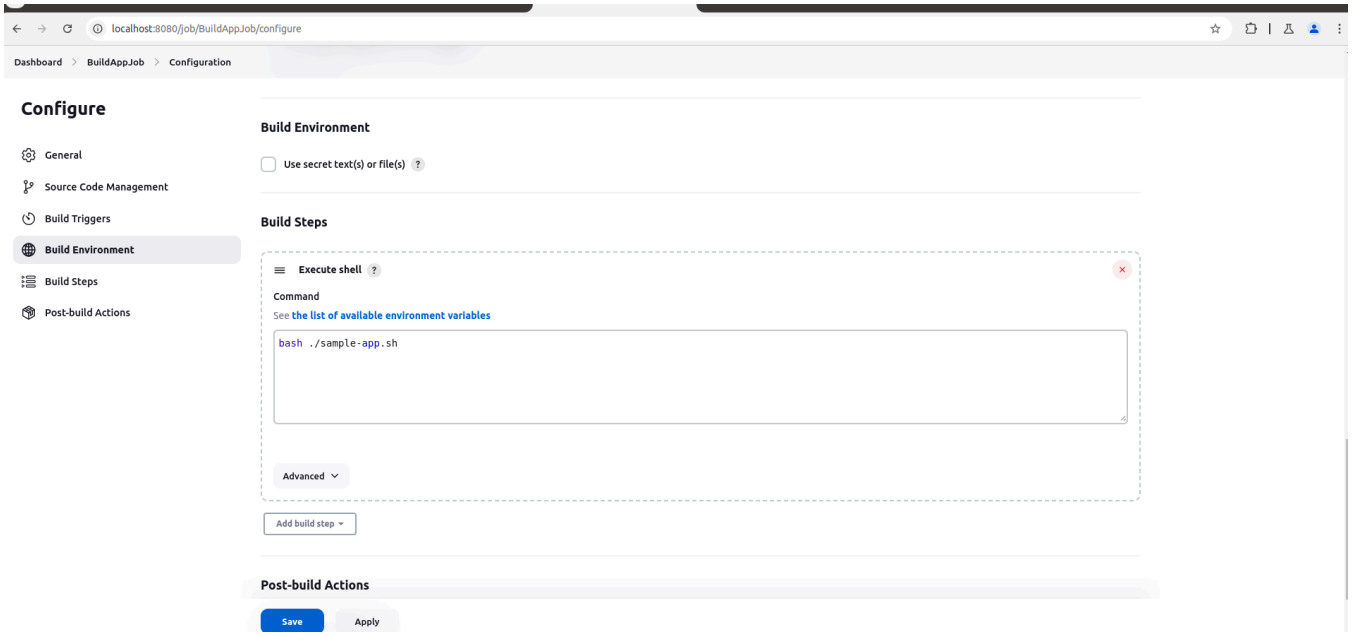
Creamos un item llamado BuildAppJob



Agregamos nuestras credenciales para conectarnos con GitHub y nuestro repositorio.



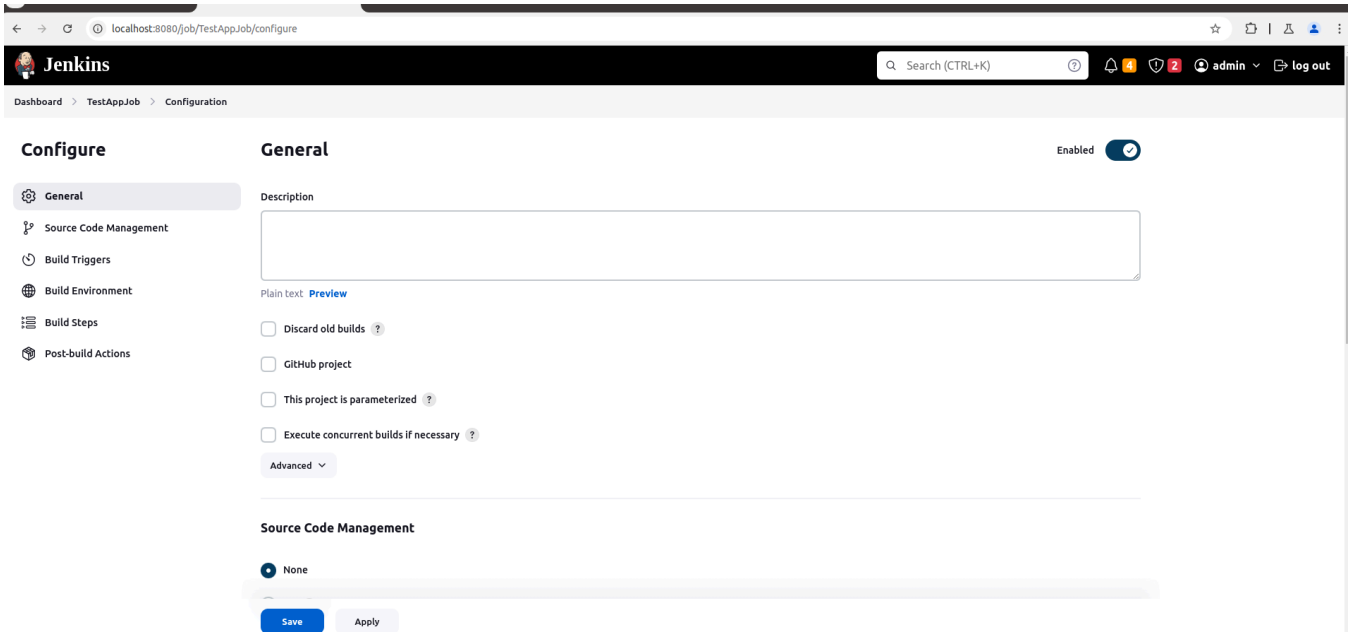
Agregamos un **build step**



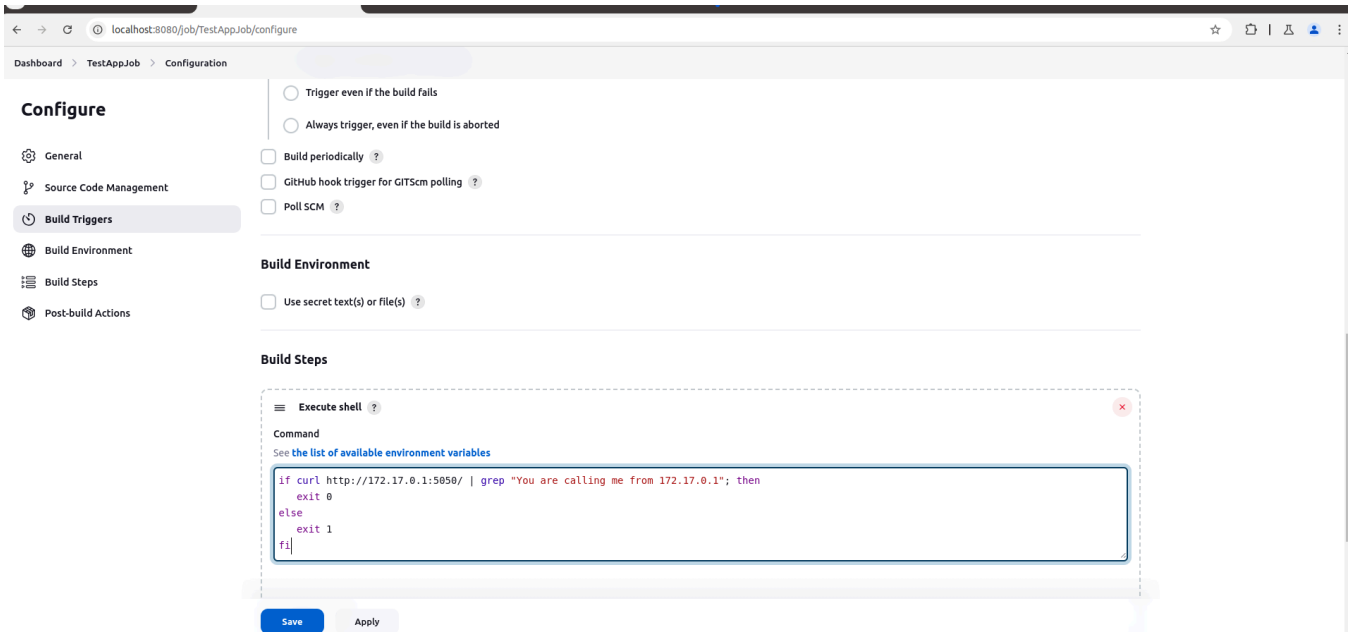
Realizamo un Build y lo ejecutó a la perfección

Usamos Jenkins para testear el Build

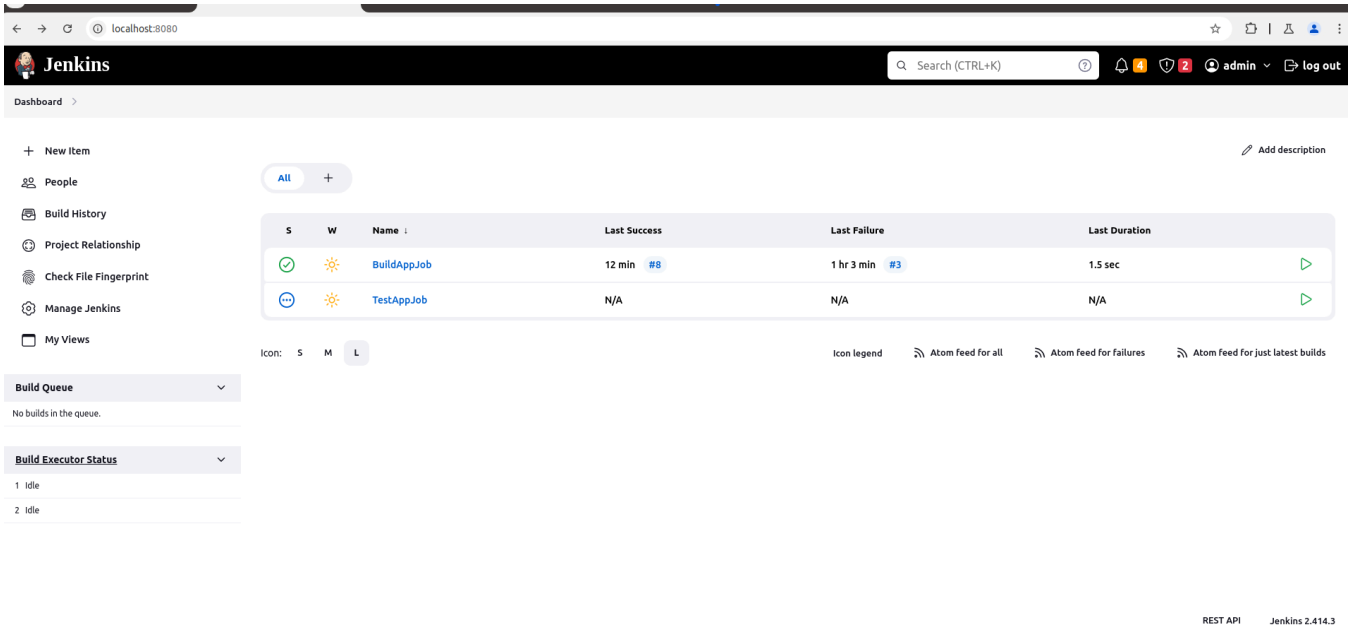
Creamos un nuevo item llamado **TestAppJob**



COnfiguramos nuestro nuevo item.

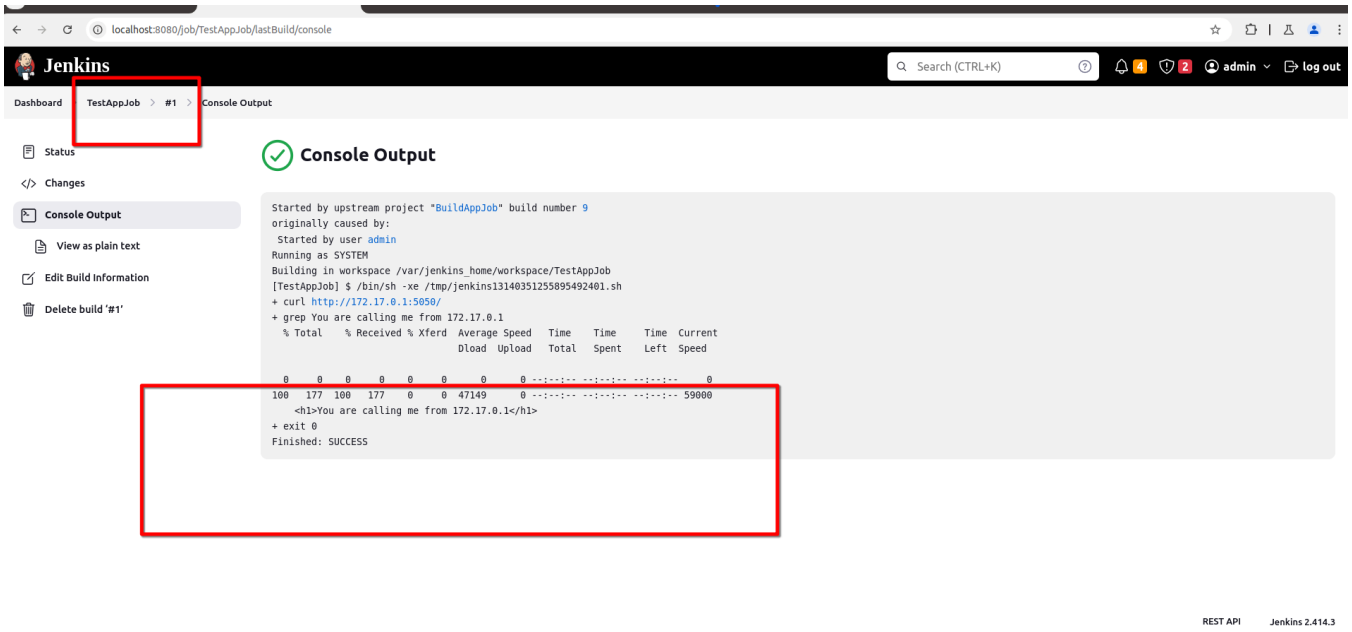


Aqui tenemos los dos



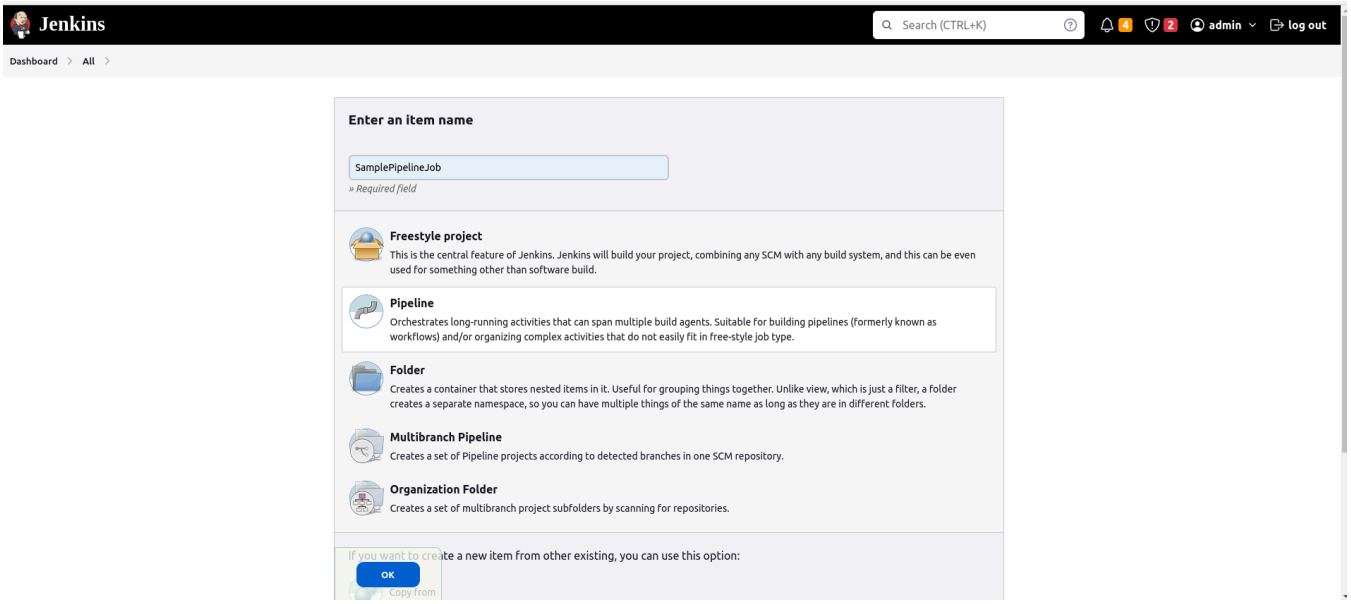
Ejecutamos el build de BuilAppJob y se ejecutó con éxito, tanto ese como el test.

S	W	Name ↓	Last Success	Last Failure	Last Duration
✓	☀	BuildAppJob	2 min 28 sec #9	1 hr 7 min #3	1.9 sec
✓	☀	TestAppJob	2 min 17 sec #1	N/A	34 ms

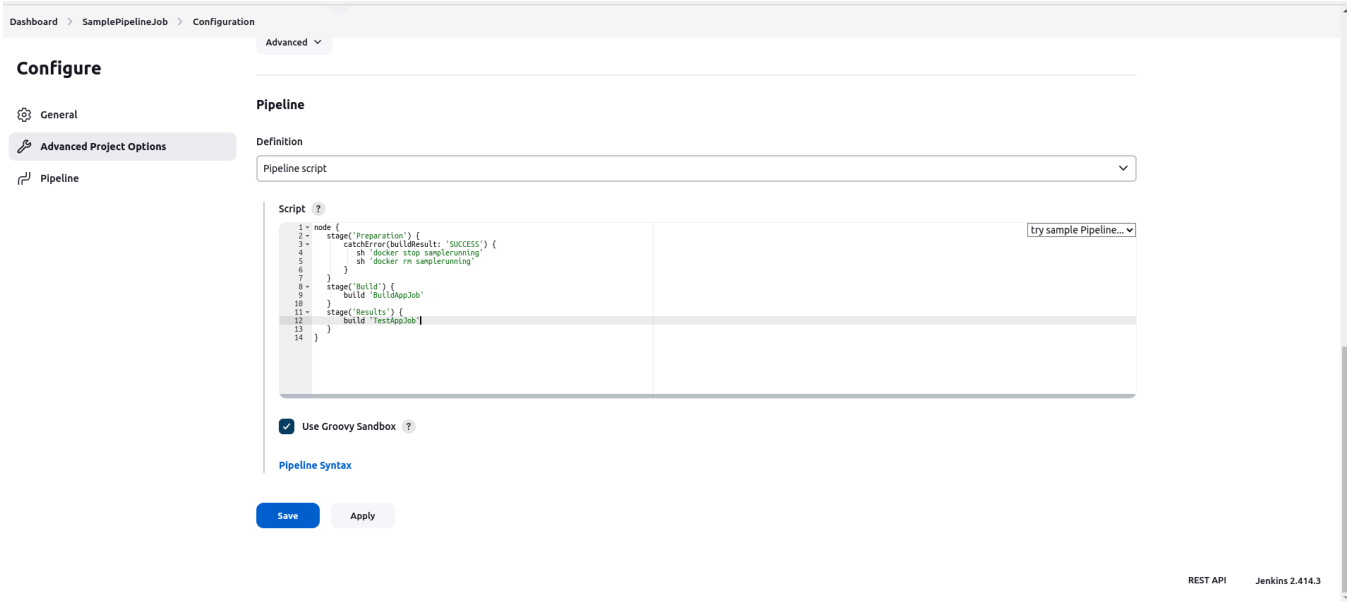


Creamos un pipeline job

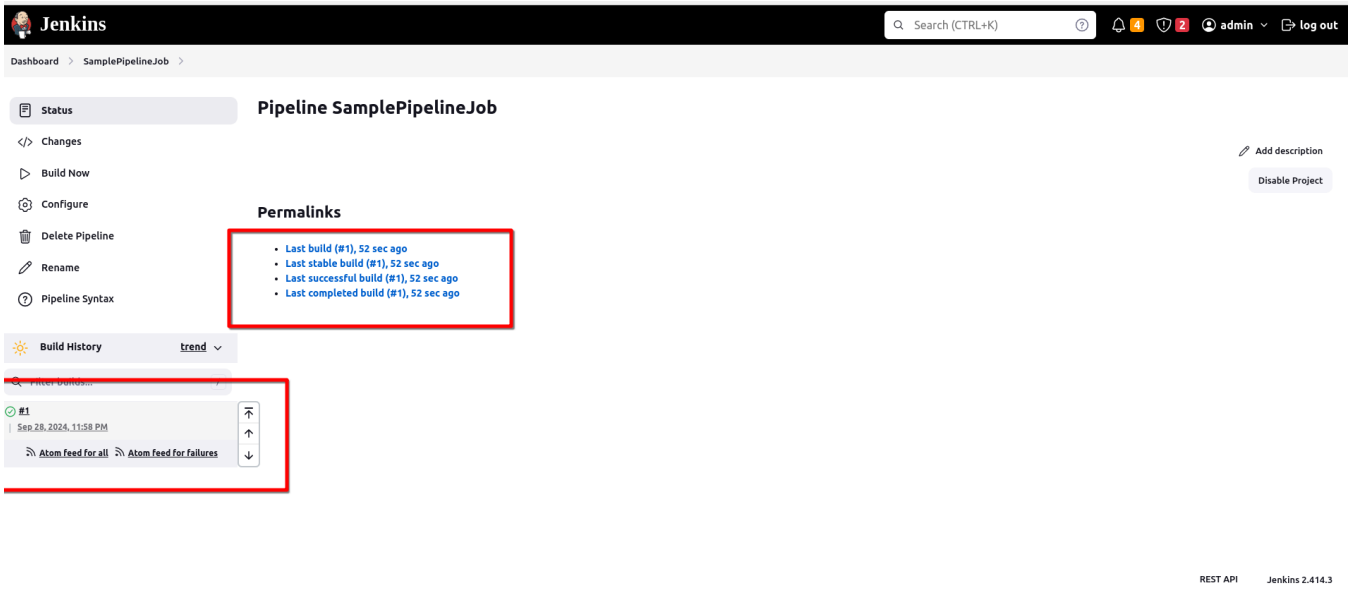
Creamos un nuevo Job de tipo pipeline, instalando el plugin Pipeline.



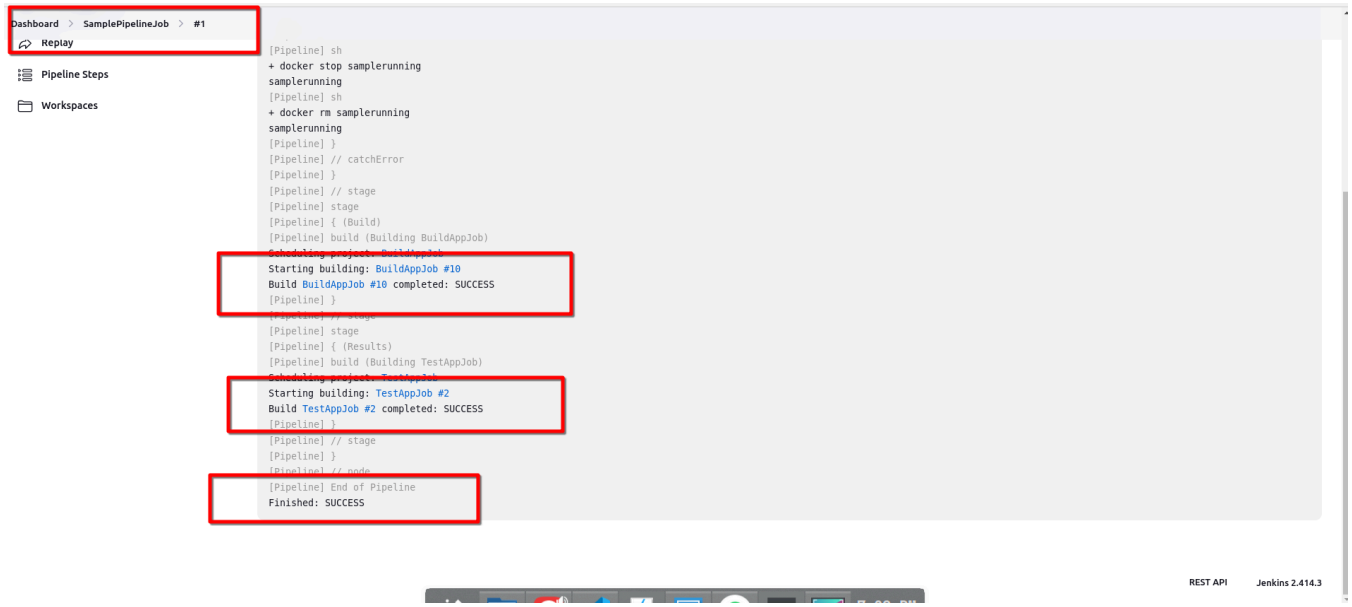
Agregamos el pipeline script



Le damos click a Build Now y funciona



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



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.

Prática Calificada 1 - Sergio Pezo.