ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELECTRÍCA Y ELECTRÓNICA

AUTOMATIZACIÓN DE REDES UTILIZADAS PARA EOT

AUTOMATIZACIÓN DE REDES CON NETMIKO

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN TECNOLOGÍAS DE LA INFORMACIÓN

JEAN CARLOS TANDAZO TANDAZO

jean.tandazo@epn.edu.ec

DIRECTOR: CARLOS ROBERTO EGAS ACOSTA

carlos.egas@epn.edu.ec

DMQ, Septiembre 2022

CERTIFICACIONES

Yo, JEAN CARLOS TANDAZO TANDAZO declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

JEAN CARLOS TANDAZO TANDAZO

Certifico que el presente trabajo de integración curricular fue desarrollado por JEAN CARLOS TANDAZO TANDAZO, bajo mi supervisión.

CARLOS ROBERTO EGAS ACOSTA
DIRECTOR

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

JEAN CARLOS TANDAZO TANDAZO

CARLOS ROBERTO EGAS ACOSTA

ÍNDICE DE CONTENIDOS

RESUMEN

El campo de la tecnología siempre se encuentra en constante evolución y crecimiento, y es por estas razones que los usuarios que nos desenvolvemos principalmente en el ámbito de las redes, también debemos irnos actualizando a la par de la tecnología.

Actualmente una de las tecnologías mayormente aplicadas en el campo de las TI, es la automatización de redes.

La automatización de redes consiste básicamente en la transición de procesos realizados en forma manual a procesos realizados por un software, esto con el firme objetivo de mejorar el desempeño de las redes.

El presente trabajo de integración curricular tiene como finalidad realizar la automatización de un prototipo de red mediante el uso de la biblioteca Netmiko, para lo cual en primera instancia se aborda toda la parte conceptual necesaria, y luego una vez teniendo muy claro todos estos conceptos, se procede a implementar la topología de red con la ayuda del software de emulación GNS3, posteriormente se procede a crear los scripts que contienen los comandos de configuración y visualización, cabe recalcar que dichos scripts son implementados en lenguaje de programación Python.

Finalmente se procede a realizar la ejecución de los scripts creados, lo cual nos permite apreciar con total claridad las virtudes y ventajas de realizar la automatización de una red.

PALABRAS CLAVE: Automatización, Redes, Programabilidad, Netmiko, Python.

ABSTRACT

The field of technology is always in constant evolution and growth, and it is for these reasons that users who work mainly in the field of networks must also update ourselves along with technology.

Currently one of the most widely applied technologies in the IT field is network automation.

Network automation basically consists of the transition from processes carried out manually to processes carried out by software, with the firm objective of improving network performance.

The purpose of this curricular integration work is to carry out the automation of a network prototype through the use of the Netmiko library, for which in the first instance all the necessary conceptual part is addressed, and then once all these concepts are very clear, we proceed to implement the network topology with the help of the GNS3 emulation software, then we proceed to create the scripts that contain the configuration and visualization commands, it should be noted that these scripts are implemented in the Python programming language.

Finally, we proceed to carry out the execution of the created scripts, which allows us to fully appreciate the virtues and advantages to do of automation a network.

KEYWORDS: Automation, Networks, Programmability, Netmiko, Python.

1. DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

La automatización de redes es el presente y sobre todo el futuro en el campo de las TI, por lo cual todas las organizaciones tanto académicas, pero sobre todo las empresariales, están realizando la transición de sus procesos manuales a procesos automatizados.

A medida que el tiempo va avanzado la complejidad en el manejo de la administración de las redes va aumentando, esto debido a que cada vez las redes están conformadas por una gran cantidad y variedad de dispositivos de red, como enrutadores o conmutadores, y es debido a este factor que se está volviendo casi imposible de manejarlos de manera individual por parte de los programadores o personal encargado de estas áreas cuando se presentan errores o se debe configurar gran cantidad de equipos en un corto tiempo, por lo que no le queda más opción a las empresas que contratar mayor personal humano para poder realizar estas actividades y no generar inconvenientes a sus usuarios finales, pero cabe recalcar que esto genera una gran inversión económica por parte de los dueños de las organizaciones y que no garantiza que no se cometan errores, ya que se debe tener muy presente que la configuración se realiza por el ser humano.

A raíz de todas las dificultades mencionadas anteriormente es que las organizaciones no están dudando en cambiar su forma de operar a un modelo automatizado, ya que la automatización de redes brinda la facilidad de poder configurar, administrar, supervisar, solucionar errores etc., de múltiples dispositivos de red al mismo tiempo, lo cual sin duda alguna genera un enorme ahorro económico para la empresa, pero sobre todo mejora sustancialmente el rendimiento y confiabilidad de la red.

En el presente trabajo de integración curricular en sus inicios se realiza un estudio teórico acerca de los conceptos más importantes de la automatización y programabilidad de las redes, así también como un estudio de algunas de las bibliotecas que se puede utilizar para implementar la automatización de redes en las organizaciones, para posteriormente proceder a realizar la automatización de un prototipo red con la ayuda del emulador GNS3 y la utilizando de la biblioteca Netmiko, combinándola con el lenguaje de programación Python, el cual es el lenguaje preferido en la actualidad para trabajar en el campo de la automatización de redes debido a sus múltiples virtudes.

1.1. Objetivo General

Realizar la automatización de un prototipo de red mediante la utilización de la herramienta Netmiko combinada con el lenguaje de programación Python.

1.2. Objetivos Específicos

- Comprender el estudio del lenguaje de programación Python en la programabilidad de redes.
- Conocer las características y ventajas que nos brinda el trabajar la herramienta Netmiko en el campo de la automatización de redes.
- Dominar los métodos de uso común de la herramienta Netmiko.
- Configurar y chequear automáticamente un prototipo de red utilizando la herramienta Netmiko.

1.3. Alcance

1.3.1. Fase de planteamiento

Dentro de esta fase se procederá a revisar la distinta documentación e información que nos permita conocer acerca del protocolo SSH, el software de emulación GNS3, la automatización de redes, las características de las redes programables, así también como las distintas particularidades que nos ofrecen las diversas herramientas existentes para la automatización de redes, y finalmente dentro de esta fase se procederá también a comprender el estudio de Python en la programabilidad de redes, para cumplir todas estas actividades mencionadas anteriormente se buscará la información tanto en libros, trabajos de titulación relacionados al tema, los cuales se encuentran almacenados dentro de los repositorios digitales de las distintas universidades y organizaciones de investigación como IEEE, adicionalmente también la información requerida será consultada en sitios web de alta confiabilidad.

Para poder tener muy claro las distintas cualidades y virtudes que presentan las diversas herramientas existentes para la automatización de redes se procederá a realizar un resumen con las características más importantes de las mismas, las cuales nos permitirán compararlas entre sí y de esta manera tener muy presente que herramienta es la mejor dentro del campo de la automatización de redes. Dentro de esta fase se procederá a estudiar más a profundidad a la herramienta Netmiko, debido

a que dicha herramienta es la seleccionada para la realización del presente trabajo de integración curricular.

Como se mencionó anteriormente la herramienta Netmiko, es el tema principal del presente trabajo, por lo cual también se investigará los diversos métodos de uso común de la misma con la finalidad de poder comprenderlos y así también poder dominarlos para ponerlos en práctica en las futuras etapas.

En cuanto al apartado acerca del estudio de Python en la programabilidad de redes lo que se hará a más de revisar la información necesaria en las fuentes mencionadas anteriormente, será buscar la documentación almacenada dentro del sitio oficial de Python relacionada al tema, así también se procederá a utilizar videos tutoriales en donde se nos explique de una manera más desarrollada y mediante ejemplos las virtudes de Python en el campo de la programabilidad de redes.

Finalmente, como resultados de esta fase se esperará poder tener un conocimiento teórico de todos los temas anteriormente citados lo suficientemente avanzados y claros que nos permitan poder seguir adelante con la fase de implementación del trabajo de integración curricular.

1.3.2. Fase de implementación

Dentro de esta fase se realizará la implementación de la topología de red utilizando el software GNS3, y posteriormente la configuración básica de todos los elementos pertenecientes al prototipo de red. La topología contendrá tres (3) dispositivos de red a ser configurados de manera automática, posteriormente se procederá a dos (2) scripts con distintos comandos de configuración y utilizando los métodos de uso común de la herramienta Netmiko estudiados en la fase anterior, y aplicando el lenguaje de programación Python.

Finalmente, como resultados de esta fase se esperará poder tener una topología de red funcional para realizar las pruebas necesarias, así también se espera tener los scripts generados libres de errores sintácticos y listos para ser ejecutados.

1.3.3. Fase de evaluación y análisis de resultados

Dentro de esta fase, se procederá a ejecutar los dos scripts creados anteriormente y se esperará obtener un resultado totalmente exitoso en todos los escenarios planteados.

Finalmente se procederá a realizar un análisis detallado de los resultados de cada uno de los escenarios con la finalidad de comprobar que se hayan cumplido con los objetivos planteados al inicio del trabajo de integración curricular.

1.4. Marco teórico

1.4.1. Protocolo SSH

El protocolo SSH (Secure Shell) surge con la intención de garantizar y sobre todo facilitar la comunicación de forma segura entre dos sistemas ya que, a diferencia de otros protocolos existentes como FTP o Telnet, SSH realiza una encriptación de la sesión de conexión, evitando así que cualquier entidad no autorizada tenga acceso a los paquetes de información. [6]

Una de las virtudes de este protocolo y que es fundamental dentro del campo de la automatización de redes es que les permite a las personas conectarse a uno o varios hosts de manera remota, algo muy importante que se debe mencionar sobre el protocolo SSH, es que este se encuentra basado en el funcionamiento de la arquitectura cliente – servidor.

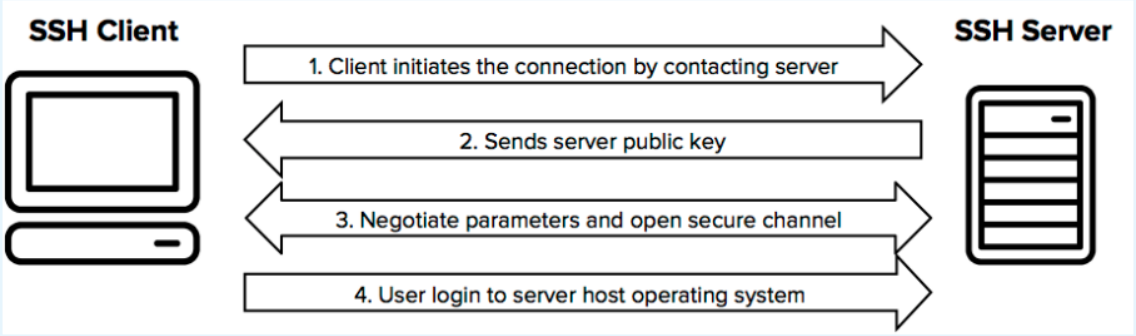


Figura 1. Establecimiento de la conexión entre Cliente SHH y Servidor SSH [17]

El procedimiento que emplea el protocolo SSH para establecer una conexión exitosa entre el sistema A, al cual llamaremos cliente SSH y el sistema B, al cual llamaremos servidor SSH, que se puede apreciar en la figura 1, es el siguiente:

En primer lugar, SSH procede a separar a la información por paquetes, dichos paquetes contienen los siguientes campos: la longitud del paquete, la cantidad de relleno, la carga útil, un relleno adicional y finalmente el código de autenticación de mensajes (MAC). Cabe recalcar que todos los campos del paquete con excepción de la longitud del paquete y el código de autenticación de mensajes (MAC), se encuentran encriptados. [19]

SSH con la intención de brindar una conexión totalmente segura entre el o los clientes con el servidor, implementa tres métodos para la manipulación de la información, los cuales son, el cifrado simétrico, el cifrado asimétrico y el hashing.

En primera instancia se utiliza la encriptación simétrica para cifrar toda la conexión, luego la clave secreta es generada mediante la aplicación del algoritmo conocido como Diffie Hellman [18], el cual es utilizado para realizar el intercambio de claves, posteriormente toda la información enviada por medio de SSH se cifra y descifra con la clave secreta generada anteriormente, a continuación, el servidor emplea la encriptación asimétrica para realizar la autenticación del cliente, a este proceso se lo conoce comúnmente como autenticación basada en clave SSH, por lo tanto el cliente procede a generar un par de claves e ingresa la clave pública del servidor al que desea acceder, una vez que se ha logrado establecer una conexión totalmente segura entre el cliente y el servidor, por medio del cifrado simétrico, el servidor procede a realizar la autenticación del cliente, enviando un mensaje de desafío, el cual se encuentra encriptado con la clave pública del cliente, y solo en caso de que dicho cliente pueda descifrar el mensaje de desafío lo que probaría que el cliente posee la clave privada asociada, se procede a realizar la autenticación del mismo.

Finalmente, el hashing es utilizado dentro de SSH para obtener el código de autenticación de mensajes (MAC), con la finalidad de comprobar que el mensaje obtenido no haya sufrido ningún daño o manipulación.[19]

La encuesta Netdevops realizada en [30] (última actualización 31 Octubre 2020), la cual es una encuesta destinada para usuarios que desempeñan actividades relacionadas al campo de la redes, nos permite obtener información estadística acerca de las herramientas o lenguajes de programación que estas personas están implementando para realizar la automatización de redes de sus empresas, dentro de la misma existe una pregunta que nos permite conocer que métodos de conexión o transporte utilizan los usuarios para implementar la automatización de redes dentro de las organizaciones en que laboran, arrojando los siguientes datos, los cuales se pueden observar en la figura 2.

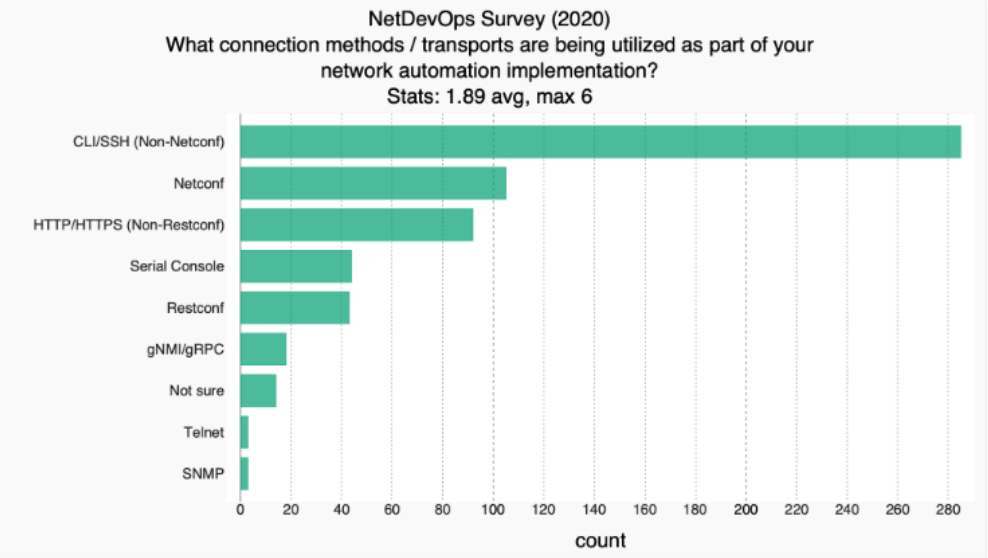


Figura 2. Métodos de conexión o transporte utilizado para la automatización de redes. [30]

En base a la figura 2, se puede confirmar que el método de conexión preferido por los encuestados para realizar las actividades relacionadas con la automatización de redes dentro de sus empresas es el protocolo SSH.

1.4.2. GNS3

El software GNS3 (Graphical Network Simulator-3), es una herramienta de simulación y virtualización gráfica de redes de código abierto, que se encuentra escrito en Python, puede ser usado tanto en el ámbito educativo como profesional, ya que permite a los usuarios trabajar con diversidad de escenarios de configuración de redes del mundo real, además brinda la facilidad de poder saber el comportamiento que tendrá una red real compleja, antes de su implementación física. [31],[32]

GNS3 es un software multiplataforma ya que se encuentra disponible para sistemas operativos Windows, Linux y MAC.

Una de las grandes virtudes que presenta GNS3, es que los dispositivos emulados en esta plataforma son capaces de interactuar con dispositivos de redes físicos, además permite a los usuarios monitorear el tráfico que se genera en la red, esto debido a que permite la utilización del analizador de paquetes Wireshark. [32]

Tomando como referencia la encuesta Netdevops ejecutada en [30], específicamente la pregunta acerca de la solución o software que los usuarios están utilizando para ejecutar dispositivos de red de manera virtual dentro de sus organizaciones, arrojan los siguientes resultados, los cuales se pueden apreciar en la figura 3.

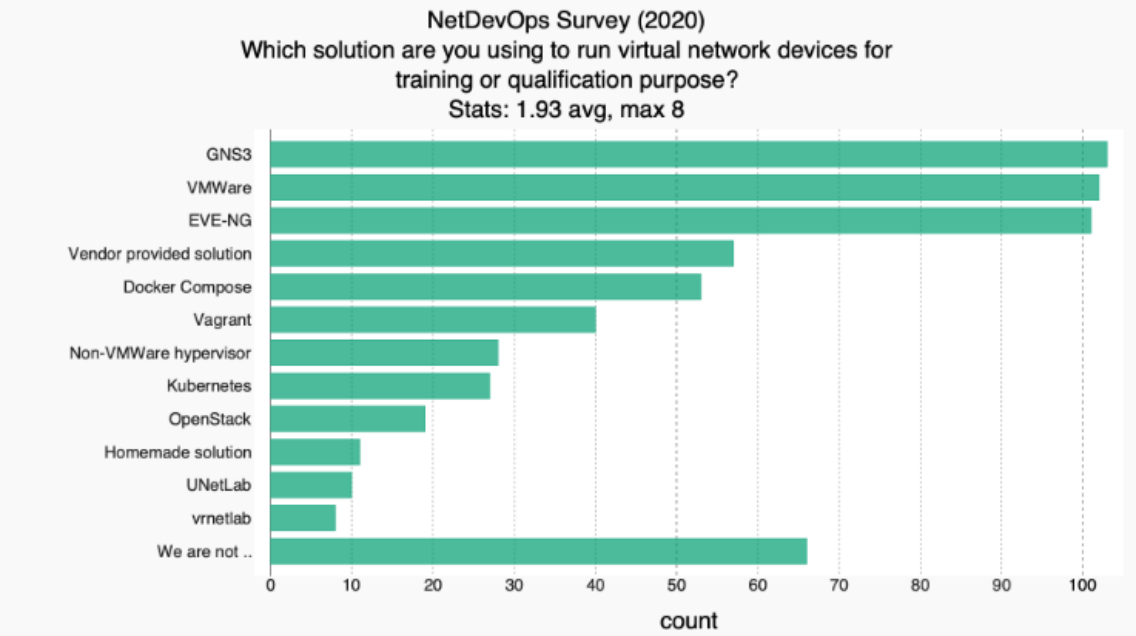


Figura 3. Soluciones para ejecutar dispositivos de red de manera virtual [30]

De la figura 3, podemos mencionar que el software o entorno preferido por los usuarios, al momento de trabajar con dispositivos de red virtuales es GNS3, el cual es el mismo programa que se empleará en el desarrollo práctico del presente trabajo de integración curricular.

1.4.3. Trabajos Relacionados

En la actualidad el tema de la automatización de redes es algo que cada vez se vuelve una necesidad el aplicarla tanto en el mundo académico, pero sobre todo en el mundo laboral, ya que le brinda una gran cantidad de ventajas a las empresas, y es por todo esto que actualmente existen múltiples estudios y trabajos que abarcan dicho tema, a continuación, se hace referencia a algunos de estos trabajos.

En [6] el autor, realiza un análisis del rendimiento de algunas bibliotecas Python que admiten el protocolo SSH y que son utilizadas para realizar automatización de redes, luego utilizando la librería timeit(), la cual nos permite conocer el tiempo que demora en ejecutarse un determinado código, realiza pruebas en dos escenarios, el primero con un solo dispositivo de red, mientras que en el segundo escenario se trabaja con múltiples dispositivos de red.

Finalmente, luego de haberse realizado las pruebas necesarias, se logra apreciar con total claridad que la biblioteca Netmiko, es la que brinda mejores resultados, ya que en

ambos escenarios fue la que mejor rendimiento brindó, debido a que fue la que menor tiempo tardó en ejecutar los comandos.

En [3] el autor, plantea la creación de una plataforma de gestión de red llamada (NMP), En primer lugar realiza un análisis teórico muy bien fundamentado acerca de la programabilidad y automatización de redes, en donde habla también de la importancia del lenguaje de programación Python dentro de estos campos, para a continuación, centrarse en la biblioteca Netmiko, de la cual explica sus virtudes, plataformas en las que opera y sus métodos de uso común, para finalmente con la ayuda del software GNS3, realizar la implementación de la red y probar los múltiples casos de uso que brindan las herramientas de automatización de redes.

En [15] los autores, realizan un análisis de rendimiento entre el tiempo que se emplea en realizar la configuración de dispositivos de red, aplicando la forma manual y la forma automatizada, es decir aplicando herramientas o bibliotecas de automatización, para lo cual utilizan una topología de red que contiene 36 dispositivos de la marca Cisco de diversas versiones de IOS, luego de realizar las respectivas pruebas se comprobó que en la forma manual se emplean 5797 segundos lo que representa un aproximado de 96,62 minutos, mientras que al aplicar las herramientas de automatización únicamente se emplea 120 segundos, es decir solamente 2 minutos, y lo mejor de todo es que la configuración fue libre de errores.

En [16] los autores, demostraron que el lenguaje de programación Python es el presente y futuro en el campo de la automatización de redes, ya que dicho lenguaje de programación combinado con herramientas o bibliotecas de automatización como son Paramiko, pero sobre todo Netmiko, simplifican y sobre todo mejoran la productividad de los programadores, ya que comprobaron que al utilizar en conjunto estas herramientas se reduce el tiempo de configuración, se simplifica el mantenimiento, se mejora la seguridad y la estabilidad de los dispositivos pertenecientes a la red.

2. METODOLOGÍA

La metodología a implementar está compuesta de tres fases, la primera fase donde se estudia todo lo relacionado con los conceptos y fundamentos teóricos referentes al tema de la automatización de redes con Netmiko en donde se abordan principalmente tópicos relacionados con la automatización y programabilidad de redes, el lenguaje de programación Python, un análisis comparativo de las distintas herramientas que se pueden aplicar en el campo de la automatización de redes y finalmente todo lo relacionado a la librería Netmiko, el cual es el tópico fundamental del trabajo.

La segunda fase comprende la construcción tanto del prototipo de red como de los scripts necesarios para configurar la topología planteada y finalmente en la tercera etapa se realiza la ejecución de los scripts mencionados anteriormente para comprobar la funcionalidad del prototipo de red y por ende cumplir con todos los objetivos planteados dentro del Trabajo de Integración curricular.

2.1. Automatización de Redes

2.1.1. Introducción

La automatización de redes es una metodología, que consiste en reemplazar los procesos manuales, es decir los desarrollados por el ser humano, por un software, donde dicho software se encarga de realizar procesos de configuración, administración, seguridad, pruebas, implementación y operaciones tanto de dispositivos físicos como virtuales pertenecientes a la red. [25], [27], [28]

La automatización de redes se puede implementar en cualquier tipo de red, como por ejemplo las redes de área local, redes de área extendida, redes de centros de datos, redes en la nube e inclusive en redes inalámbricas, es por estas razones que en la actualidad las grandes empresas, centros de datos, así como los diferentes proveedores de redes, implementan la automatización de redes dentro de sus organizaciones con la finalidad de poder gestionar y evitar procedimientos repetitivos, reducir gastos de operación, evitar errores humanos, pero sobre todo mejorar exponencialmente el desempeño, disponibilidad y confiabilidad de la red. [29]

Tomando como referencia nuevamente a la encuesta Netdevops realizada en [30], y centrándonos en la pregunta relacionada a conocer en que regiones del mundo se está utilizando la automatización de redes, se obtuvieron las siguientes estadísticas, las cuales se pueden observar en la figura 4.

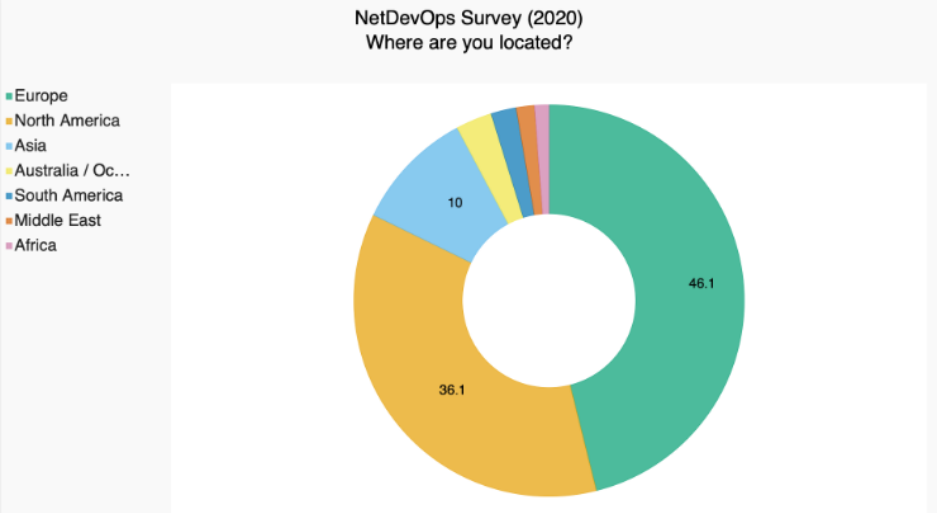


Figura 4. Automatización de redes las regiones del mundo [30]

De la figura 4, podemos mencionar que Sudamérica es una de las regiones en donde la automatización de redes, aún se encuentra en sus inicios, esto a pesar de que dicha tecnología lleva en el mundo cerca de 20 años.

2.1.2. Automatización de redes basada en scripts

La automatización de redes basada en secuencias de comandos o instrucciones (scripts), utiliza lenguajes de programación para realizar procesos de configuraciones o monitoreos, en donde a medida que aumenta la complejidad de las redes, los lenguajes de programación de código abierto van incrementando su popularidad y por ende su utilización, esto debido a que dichos lenguajes son de fácil implementación, destacando en la actualidad Python.

2.1.3. Beneficios de implementar la automatización de redes

- **Minimización de posibles errores humanos:** esto debido a que aplicando la automatización se disminuye sustancialmente la participación del ser humano en la realización de procedimientos que requieran configuraciones de red de alta complejidad, tratando de esta manera de evitar errores de configuración, pero sobre todo errores tipográficos, lo que a su vez produce un aumento sustancial de la confiabilidad y rendimiento de la red.
- **Optimización del rendimiento de la red:** la automatización de redes admite la inclusión de diversas herramientas de monitoreo, los usuarios administradores podrán recibir alertas acerca de errores que se produzcan en la red, altos niveles de uso de los recursos y sobre todo datos que ocasionen problemas de rendimiento, lo cual ayudará a solucionar dichos inconvenientes en el menor tiempo posible.

- **Simplificación de la administración de la red:** el automatizar la red permite reducir lo más posible el tiempo de inactividad o falla de la misma, ya que las distintas configuraciones se ejecutarán a todos los dispositivos requeridos de una manera consistente, rápida y sobre todo sencilla, mejorando exponencialmente la escalabilidad de la red, lo que a su vez representa una rentabilidad económica para la organización.

2.2. Redes programables

Las redes programables son aquellas redes que cuentan con herramientas prácticas de software como por ejemplo la virtualización de funciones de red. Estas herramientas facilitan al usuario la creación, despliegue, administración, configuración y ejecución de los diversos servicios y recursos de la red, por lo que en resumidas palabras permiten que las redes puedan ser modificadas de una manera dinámica y ágil sin la necesidad de preocuparse por el número de dispositivos que intervengan dentro de la misma.

Adicionalmente estas redes cuando presentan algún problema, brindan y facilitan la posibilidad de recolectar información de cualquier segmento de red y de ejecutar las acciones necesarias para corregir dichos problemas sobre cualquier dispositivo de red que sea necesario.

2.2.1. Redes definidas por Software (SDN)

Las Redes definidas por Software (SDN) también conocidas como redes programables y automatizadas, tienen sus orígenes en el año de 1990, y nacen con la finalidad de mejorar la eficiencia, compatibilidad y escalabilidad de las redes, reducir costos de operación, pero sobre todo facilitar la administración de las mismas.

En la actualidad existen tres tipos de implementación SDN, las cuales se mencionan a continuación:

- **SDN basada en dispositivos:** aplicaciones creadas por programadores en dispositivos, utilizando lenguajes de programación C, Python y Java.
- **SDN basada en controlador:** Un controlador centralizado realiza toda la gestión de los dispositivos de red.
- **SDN basada en políticas:** Apps de alto nivel que permiten la automatización del controlador basándose en las políticas mediante una interfaz gráfica amigable, por lo cual no se necesita conocimientos o habilidades de programación.

2.2.1.1. Características de las SDN

- Cuenta con una infraestructura programable.

- Permite una administración centralizada.
- Realiza abstracción de la red, y separa el plano de control del plano de datos.
- Brinda flexibilidad y permite actualizaciones en tiempo real.
- Implementa automatización.
- Es del tipo Open source, es decir se basa en código abierto, lo que permite la neutralidad del fabricante.

2.2.1.2. Arquitectura de las SDN

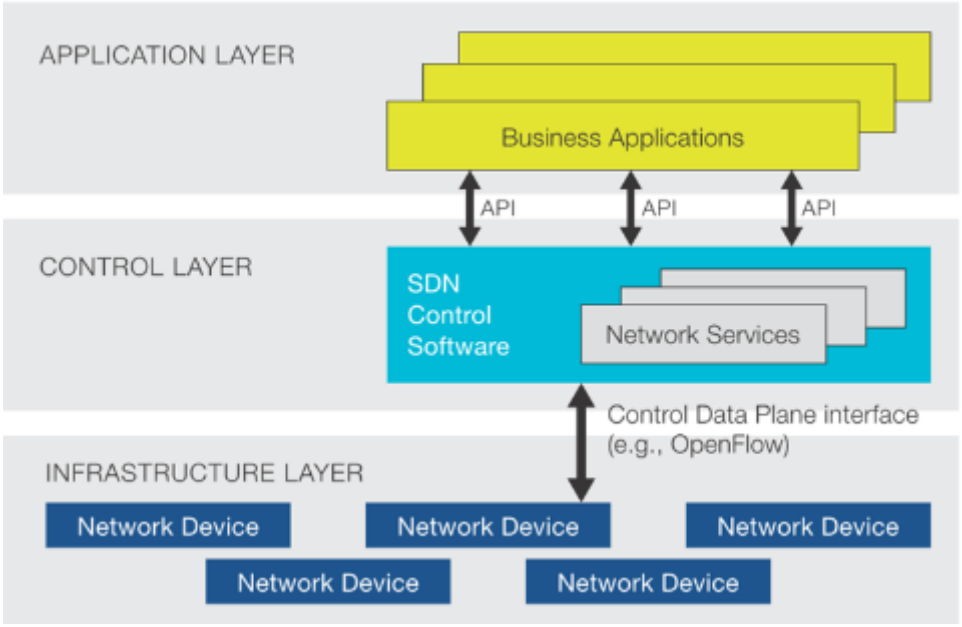


Figura 5. Arquitectura de una SDN [38]

La arquitectura de una SDN se compone de tres capas: capa aplicación, capa de control y capa de infraestructura.

- **Capa aplicación:** realiza la inclusión de aplicaciones y funciones de red, las cuales ayudan a mejorar el rendimiento de la red y la ejecución de manera automática de configuraciones y nuevos servicios de red. [38]
- **Capa de control:** se encarga de administrar las políticas y el flujo de los datos de la red. [38]
- **Capa de infraestructura:** está conformada por enrutadores y conmutadores físicos de la red, dichos dispositivos realizan la administración de las tablas de flujo. [38]

2.2.1.3. Ventajas de las SDN frente a las redes tradicionales

- Debido a su flexibilidad, las redes SDN tienen la facilidad de poder crecer con mayor rapidez y con una inversión económica mucho menor.

- Permiten el mejoramiento exponencial de la escalabilidad y seguridad de las redes.
- En las redes tradicionales los dispositivos de red utilizan softwares propietarios, mientras que SDN utiliza el protocolo OpenFlow.

2.3. Lenguaje de Programación Python

2.3.1. Introducción

Python es un poderoso, pero sobre todo versátil lenguaje de programación el cual tiene sus orígenes en el año de 1991 de la mano de Guido van Rossum. En la actualidad es uno de los lenguajes de programación mayormente utilizados debido a que es de tipo interpretado, orientado a objetos, pero sobre todo de alto nivel, por lo cual su sintaxis es bastante fácil y por ende es sumamente sencillo de comprender y aprender por parte de los usuarios. [22]

La popularidad en la actualidad de Python se debe a que permite desarrollar un sin número de aplicaciones con bastante rapidez, pero sobre todo su utilización en el campo de la automatización de redes se debe primordialmente a que cuenta con gran cantidad de módulos y bibliotecas que ayudan a ejecutar de una manera más rápida y eficiente las distintas tareas.[22]

2.3.2. Python dentro del campo de la automatización de redes

Python se ha vuelto el lenguaje de programación preferido al momento de trabajar con redes, esto es debido a su gran capacidad para adaptarse y trabajar con una amplia variedad de bibliotecas, las cuales facilitan a los usuarios la creación de scripts, los cuales por lo general son de pocas líneas de código, pero que su ejecución permite desarrollar tareas de bastante complejidad, además cabe mencionar que al trabajar con Python no es necesario definir ni variables, tampoco objetos antes de poder utilizarlos lo cual simplifica sustancialmente el proceso de inicio. [3]

Otra de las razones por la que Python es el lenguaje de moda en el campo de las redes, es debido a que tanto los proyectos de código abierto como los diversos proveedores de redes, cada vez siguen creando nuevas bibliotecas y herramientas, basándose en este lenguaje de programación, también se debe mencionar que la biblioteca estándar de Python brinda un soporte completo para los múltiples protocolos de red existentes. [24]

La utilización de Python dentro del campo de la automatización de redes, sin duda alguna que ha ayudado enormemente al desarrollo del mismo, puesto que su intervención ha permitido aumentar significativamente la eficiencia, la seguridad, la escalabilidad y también la confiabilidad de las redes. [23]

Nuevamente tomando como referencia la encuesta Netdevops realizada en [30], específicamente en la pregunta acerca de que lenguaje de programación utilizan mayormente los usuarios para realizar la automatización de redes dentro de sus empresas, arrojan los siguientes datos, los cuales se pueden observar en la figura#....

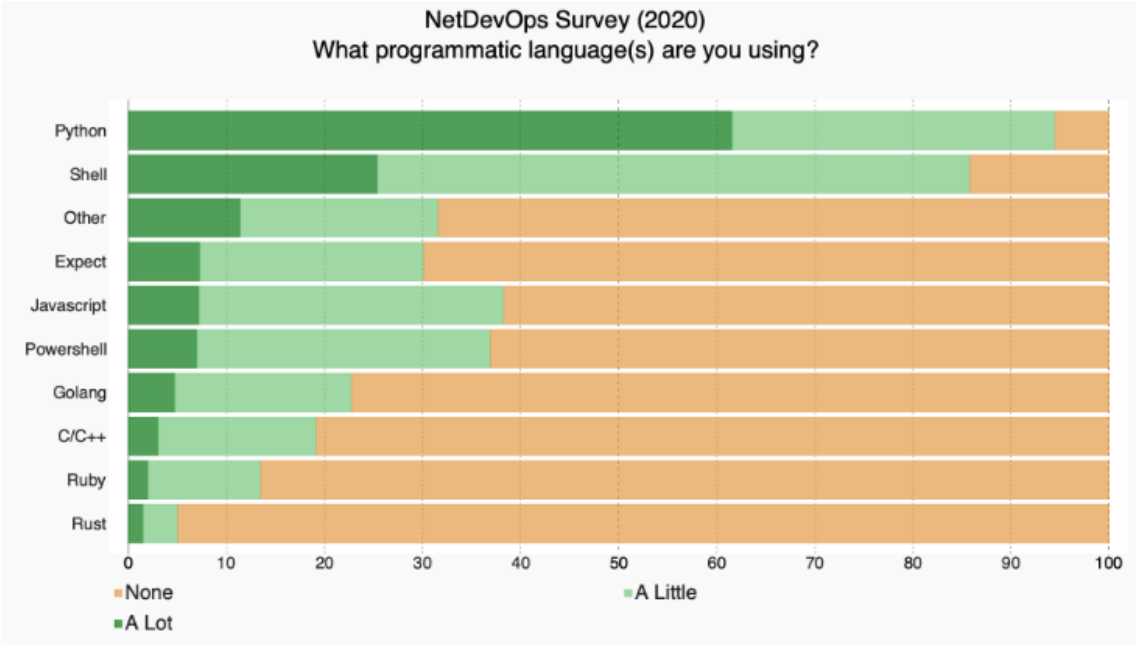


Figura 6. Lenguajes de programación más utilizados para la automatización de redes [30]

Como se puede apreciar en la figura 6, la comunidad contestó de una forma mayoritaria que utilizan el lenguaje de programación Python para realizar las actividades relacionadas con la automatización de redes dentro de sus organizaciones.

2.4. Netmiko

Netmiko es considerada una biblioteca de tipo Python que tuvo su origen a finales del año 2014 de la mano de Kirk Byers, esta biblioteca tiene la facilidad de poder operar para múltiples proveedores, además cabe mencionar que es de código abierto y que su principal función es el facilitar el proceso de creación y administración de los distintos dispositivos de una red a través del protocolo SSH. [1], [2]

Netmiko tiene sus bases en la biblioteca Paramiko SSH, la cual es la biblioteca SSH por defecto del lenguaje de programación Python. [1], [2]

2.4.1. Historia

Netmiko nace a raíz de que la creadora de esta biblioteca, se fijó en las dificultades que tenían una gran cantidad de personas al trabajar con Python-SSH junto con los distintos

dispositivos de red, en donde la resolución de dichos problemas tomaba una gran cantidad de tiempo, llegando inclusive a casos donde ni siquiera se podía lograr una solución, es por esto que Netmiko fue creado con el objetivo de facilitar la administración SSH de bajo nivel, con la visión de poder operar para una gran cantidad de plataformas y diversos proveedores de redes. [3]

2.4.2. Dispositivos compatibles

Netmiko al ser considerada una biblioteca multiplataforma tiene la versatilidad de poder operar dentro de una gran variedad de dispositivos de red, a los cuales se los ha clasificado en tres categorías (última actualización 20 de junio de 2022), las cuales se aprecian en la tabla 1, a continuación:

Tabla 1. Dispositivos compatibles con Netmiko divididos en tres categorías *[4]*.

Regularly tested	Limited testing	Experimental
• Arista vEOS • Cisco ASA • Cisco IOS • Cisco IOS-XE • Cisco IOS-XR • Cisco NX-OS • Cisco SG300 • HP ProCurve • Juniper Junos • Linux	• 6Wind • Adtran OS • Alcatel AOS6/AOS8 • Apresia Systems AEOS • Broadcom ICOS • Calix B6 • Centec Networks • Cisco AireOS (Wireless LAN Controllers) • CloudGenix ION • Dell OS9 (Force10) • Dell OS10 • Dell PowerConnect • Ericsson IPOS • Extreme ERS (Avaya) • Extreme MLX/NetIron (Brocade/Foundry) • Extreme TierraOS • Extreme VDX (Brocade) • Extreme VSP (Avaya) • HPE Comware7 • Huawei • Huawei OLT	• A10 • Accedian • Allied Telesis AlliedWare Plus • Aruba • Brocade Fabric OS • C-DOT CROS • Ciena SAOS • Citrix Netscaler • Cisco Telepresence • Cisco Viptela • Check Point GAiA • Coriant • Dell OS6 • Dell EMC Isilon • Eltex • Enterasys • Endace • Extreme EXOS • Extreme Wing • Extreme SLX (Brocade) • F5 TMSH • F5 Linux • Fortinet • MRV Communications OptiSwitch

	• Huawei SmartAX • IP Infusion OcNOS • Juniper ScreenOS • MikroTik RouterOS • MikroTik SwitchOS • NetApp cDOT • Netgear ProSafe • Nokia/Alcatel SR OS • Nokia SR Linux • NVIDIA-Mellanox • OneAccess • Palo Alto PAN-OS • Pluribus • Ruckus ICX/FastIron • Ruijie Networks • Supermicro SMIS • TPLink JetStream • Ubiquiti EdgeSwitch • Vyatta VyOS • Yamaha • ZTE ZXROS	• MRV LX • Nokia/Alcatel SR-OS • Nokia SR Linux • QuantaMesh • Rad ETX • Raisecom ROAP • Sophos SFOS • Ubiquiti Unifi Switch • Versa Networks FlexVNF • Watchguard Firebox • Zyxel NOS • 6WIND TurboRouter

2.4.3. Casos de Uso

A continuación, se menciona algunos de los múltiples casos de uso de la biblioteca Netmiko.

- **Copias o respaldos de seguridad de las configuraciones:** Nos permite automatizar la recuperación de los datos de salida de las diversas configuraciones que se encuentren ejecutándose, todo esto de una manera programada. [2]
- **Auditorías de seguridad:** Se puede ejecutar un comando que nos permita conocer si los dispositivos de la red están ejecutando una determinada versión de software vulnerable. [2]
- **Automatización de soluciones a diversos problemas:** Sin duda uno de los usos principales de la biblioteca Netmiko, ya que nos brinda la posibilidad de automatizar los procesos de ejecución de uno o varios comandos, con la finalidad de poder solucionar cualquier problema que se presente, de una manera rápida y eficiente. [2]

2.4.4. Métodos de uso común en Netmiko

En la tabla 2, se presenta los métodos mayormente utilizados para realizar scripts con la ayuda de la biblioteca Netmiko, con una breve descripción de su funcionalidad.

Tabla 2. Métodos de uso común en Netmiko [3].

Método	Descripción
net_connect.send_command()	Permite el envío de un comando a cualquier dispositivo de la red a través del canal. Ejemplo: output = net_connect.send_command('show ip int brief')
net_connect.send_config_set()	Permite el envío de los comandos de configuración a los dispositivos remotos.
net_connect.send_config_from_file()	Permite el envío de los comandos de configuración a los dispositivos remotos, cargados desde un archivo.
net_connect.save_config()	Permite guardar la configuración en ejecución en la configuración inicial.
net_connect.enable()	Permite ingresar al modo de habilitación.
net_connect.disconnect()	Realiza el cierre de la conexión.

Netmiko puede ser ejecutado en distintos softwares como por ejemplo Visual Studio Code, por lo cual para poder apreciar de una mejor manera el uso de algunos de los métodos mencionados en la Tabla 2, se procede a realizar un ejemplo práctico, para lo cual se utiliza el siguiente sandbox, cuyos datos se pueden apreciar a continuación.

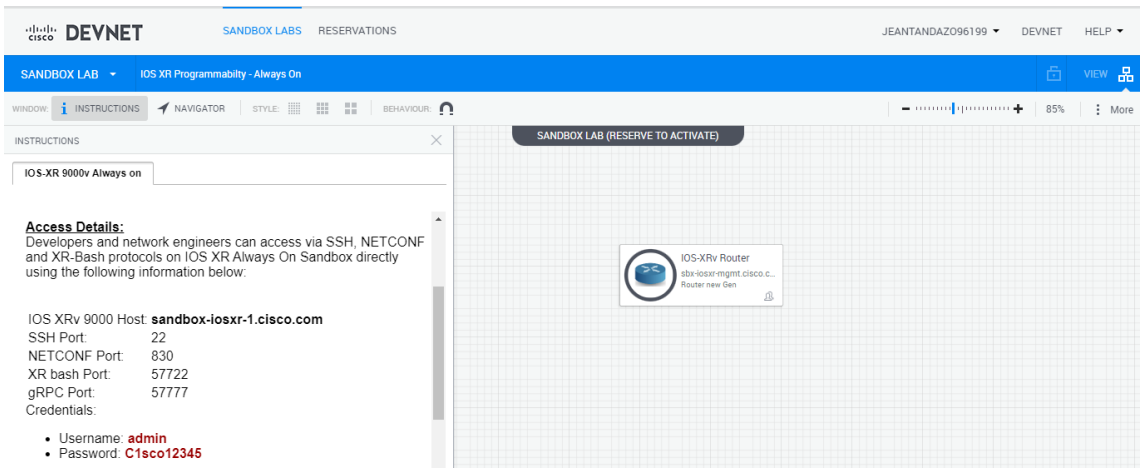


Figura 7. Sandbox IOS XRv 9000 [39]

Algo que se debe tener muy en cuenta antes de empezar a crear el respectivo script dentro del programa Visual Studio Code es que debemos comprobar que tengamos instalado Python dentro de nuestro computador, para comprobar lo mencionado podemos ejecutar dentro del cmd el siguiente comando:

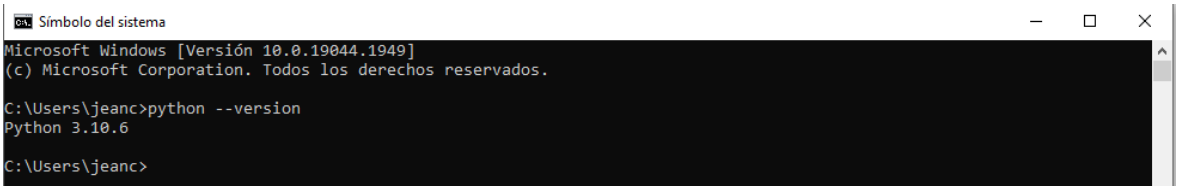


Figura 8. Versión Python instalada

Una vez comprobado lo anterior, procedemos a instalar la biblioteca netmiko, ejecutando el comando pip install netmiko, posteriormente obtendremos un resultado como el que se muestra en la siguiente imagen.

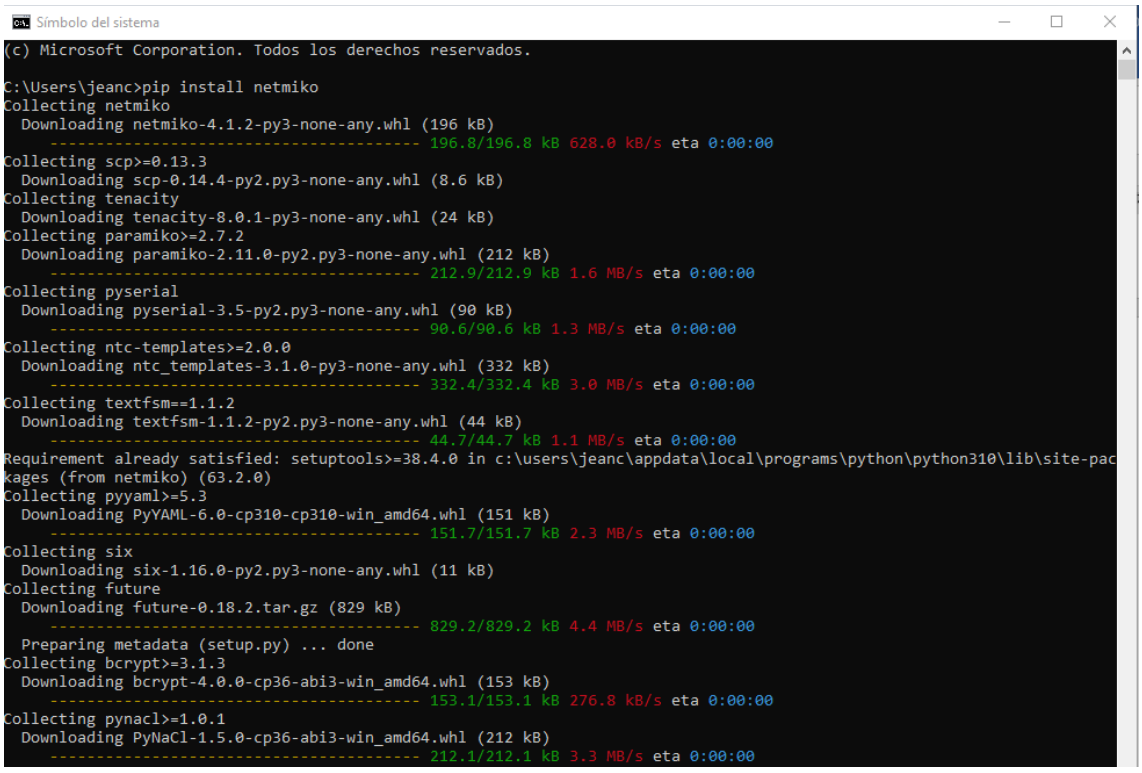


Figura 9. Instalación de la biblioteca Netmiko

Una vez que tengamos instalada la herramienta Netmiko, procedemos a crear el script, pero no sin antes mencionar que la página que nos proporciona el sandbox, nos solicita que no se realice ningún cambio en las configuraciones del dispositivo, por lo cual, para esta ocasión dentro del script, únicamente se ejecutarán comandos para chequeo o

visualización del estado del dispositivo, teniendo muy presente esta recomendación se realiza el script que se puede apreciar a continuación.

NetmikoTIC.py

C: > Users > jeanc > NetmikoTIC.py > ...

```
1
2   from netmiko import ConnectHandler
3
4   ios_XRv_9000 = {
5       'device_type': 'cisco_xr',
6       'host': 'sandbox-iosxr-1.cisco.com',
7       'username' : 'admin',
8       'password' : 'C1sco12345'
9   }
10  net_connect = ConnectHandler(**ios_XRv_9000)
11  output = net_connect.send_command('show ip int brief')
12  print (output)
13  output = net_connect.send_command('show ip route')
14  print (output)
15  net_connect.disconnect()
16
```

Figura 10. Script generado para el sandbox IOS XRv 900

Finalmente se muestra los resultados obtenidos, al mandar a ejecutar el script.

En la siguiente figura se aprecia los resultados generados del comando show ip int brief

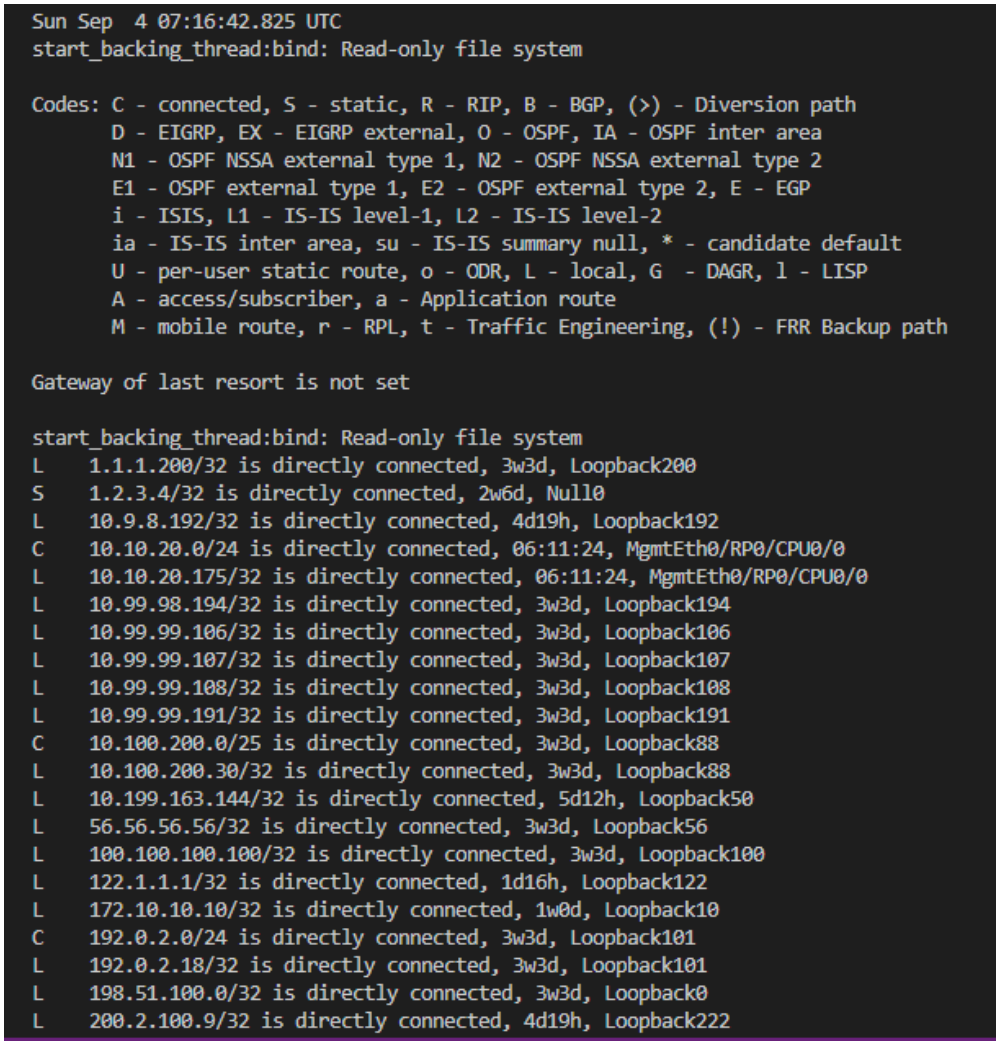
```
PS C:\Users\jeanc> & C:/Users/jeanc/AppData/Local/Programs/Python/Python310/python.exe c:/Users/jeanc/NetmikoTIC.py

Sun Sep  4 07:16:42.195 UTC
start_backing_thread:bind: Read-only file system
start_backing_thread:bind: Read-only file system

Interface                      IP-Address      Status          Protocol Vrf-Name
Bundle-Ether23                 unassigned      Down            Down     default
Bundle-Ether24                 unassigned      Down            Down     default
Bundle-Ether24.300             10.1.1.1        Down            Down     default
Loopback0                      198.51.100.0    Up              Up       default
Loopback10                     172.10.10.10    Up              Up       default
Loopback50                     10.199.163.144  Up              Up       default
Loopback56                     56.56.56.56     Up              Up       default
Loopback88                     10.100.200.30   Up              Up       default
Loopback99                     unassigned      Up              Up       default
Loopback100                    100.100.100.100 Up              Up       default
Loopback101                    192.0.2.18      Up              Up       default
Loopback104                    192.168.1.1     Shutdown        Down     default
Loopback106                    10.99.99.106    Up              Up       default
Loopback107                    10.99.99.107    Up              Up       default
Loopback108                    10.99.99.108    Up              Up       default
Loopback111                    unassigned      Up              Up       default
Loopback122                    122.1.1.1       Up              Up       default
Loopback191                    10.99.99.191    Up              Up       default
Loopback192                    10.9.8.192      Up              Up       default
Loopback194                    10.99.98.194    Up              Up       default
Loopback200                    1.1.1.200       Up              Up       default
Loopback222                    200.2.100.9     Up              Up       default
Loopback291                    10.99.99.191    Shutdown        Down     default
Loopback292                    unassigned      Up              Up       default
Loopback888                    unassigned      Up              Up       default
tunnel-ip10                    10.199.163.173  Down            Down     default
MgmtEth0/RP0/CPU0/0            10.10.20.175    Up              Up       default
GigabitEthernet0/0/0/0         172.31.1.1      Down            Down     default
GigabitEthernet0/0/0/0.12312   unassigned      Down            Down     default
GigabitEthernet0/0/0/1         unassigned      Down            Down     default
```

Figura 11. Resultados obtenidos del comando show ip int brief

A continuación, se observa los resultados obtenidos del comando show ip route.



```
Sun Sep  4 07:16:42.825 UTC
start_backing_thread:bind: Read-only file system

Codes: C - connected, S - static, R - RIP, B - BGP, (>) - Diversion path
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - ISIS, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, su - IS-IS summary null, * - candidate default
       U - per-user static route, o - ODR, L - local, G  - DAGR, l - LISP
       A - access/subscriber, a - Application route
       M - mobile route, r - RPL, t - Traffic Engineering, (!) - FRR Backup path

Gateway of last resort is not set

start_backing_thread:bind: Read-only file system
L    1.1.1.200/32 is directly connected, 3w3d, Loopback200
S    1.2.3.4/32 is directly connected, 2w6d, Null0
L    10.9.8.192/32 is directly connected, 4d19h, Loopback192
C    10.10.20.0/24 is directly connected, 06:11:24, MgmtEth0/RP0/CPU0/0
L    10.10.20.175/32 is directly connected, 06:11:24, MgmtEth0/RP0/CPU0/0
L    10.99.98.194/32 is directly connected, 3w3d, Loopback194
L    10.99.99.106/32 is directly connected, 3w3d, Loopback106
L    10.99.99.107/32 is directly connected, 3w3d, Loopback107
L    10.99.99.108/32 is directly connected, 3w3d, Loopback108
L    10.99.99.191/32 is directly connected, 3w3d, Loopback191
C    10.100.200.0/25 is directly connected, 3w3d, Loopback88
L    10.100.200.30/32 is directly connected, 3w3d, Loopback88
L    10.199.163.144/32 is directly connected, 5d12h, Loopback50
L    56.56.56.56/32 is directly connected, 3w3d, Loopback56
L    100.100.100.100/32 is directly connected, 3w3d, Loopback100
L    122.1.1.1/32 is directly connected, 1d16h, Loopback122
L    172.10.10.10/32 is directly connected, 1w0d, Loopback10
C    192.0.2.0/24 is directly connected, 3w3d, Loopback101
L    192.0.2.18/32 is directly connected, 3w3d, Loopback101
L    198.51.100.0/32 is directly connected, 3w3d, Loopback0
L    200.2.100.9/32 is directly connected, 4d19h, Loopback222
```

Figura 12. Resultados obtenidos del comando show ip route

Como se pudo apreciar, el script generado fue libre de errores y se obtuvo la información solicitada dentro del mismo, adicionalmente se pudo observar la utilización de algunos de los comandos de uso común de la biblioteca Netmiko que era el objetivo de este ejemplo, ya que más adelante, se procede a realizar una topología de red más completa donde intervienen todos los comandos mencionados.

2.4.5. Ventajas

El trabajar con la herramienta Netmiko, presenta grandes beneficios para las personas y por ende para las empresas, dichas virtudes se mencionan a continuación:

- Facilidad para lograr establecer exitosamente una conexión SSH con el dispositivo de red deseado. [1]

- Ayuda a simplificar la recuperación de los datos salientes, así como también facilita la ejecución de los diversos comandos show, [3]
- Permite simplificar la ejecución de los múltiples comandos de configuración existentes, incluyendo también las probables acciones de confirmación. [3]
- Versatilidad ya que puede operar dentro de una gran variedad de plataformas y diversos proveedores de redes. [1]
- Netmiko tiene también la gran la virtud de poder admitir a más de conexiones SSH, conexiones Telnet, conexiones en serie, e inclusive Secure Copy. [1]

2.5. Tipos de bibliotecas SSH para automatización de redes

Además de la biblioteca Netmiko, la cual fue analizada anteriormente, existen otras bibliotecas SSH que permiten al usuario realizar automatización de redes, a continuación, se nombra las más importantes, y luego se procede a realizar un análisis comparativo de dichas bibliotecas.

2.5.1. Paramiko

Paramiko SSH o simplemente Paramiko es la biblioteca SSH por defecto del lenguaje de programación Python, esta biblioteca ayuda a realizar la conexión de servidores remotos mediante el proceso de autenticación y encriptación, además es considera multiplataforma ya que puede operar para varios sistemas operativos como Linux, Solaris, BSD, MacOS X, Windows, entre otros, adicionalmente presenta compatibilidad con el modelo cliente-servidor SFTP, ya que al operar como cliente o usuario se puede iniciar sesión mediante el ingreso de una contraseña o una clave, en cambio al operar en modo servidor, permite realizar el monitoreo de la red con la finalidad de poder conocer cuales usuarios o clientes tienen acceso y cuál es la plataforma que pueden utilizar. [6], [7], [8], [9]

Paramiko presenta un poco de complejidad en su utilización sobre todo para las personas que recién están empezando a trabajar con el lenguaje Python, esto se debe a que en dicha biblioteca se tiene que manejar de una manera muy minuciosa cada uno de los detalles para administrar cada paso del proceso, habilitar el acceso, ingresar al modo de configuración, enviar y guardar los respectivos cambios realizados, entre otras acciones. [3]

2.5.2. SSH2-python

SSH2-python es una biblioteca perteneciente al protocolo SSH2 extremadamente rápida y ligera, además es también un enlace del lenguaje de programación Python para libssh2. [11]

Dicha biblioteca fue desarrollada con el firme objetivo de ser un contenedor ligero de libssh2, y por dicha razón fue implementada en Cython, ya que Cython es un compilador estático que ayuda a la optimización del lenguaje de programación Python [12], obteniendo de esta manera la menor sobrecarga posible, y al mismo tiempo el máximo rendimiento, además cabe recalcar que también es una biblioteca multiplataforma, ya que trabaja en sistemas operativos como Linux, OSX y Windows [10]

2.5.3. Scrapli

Scrapli es una biblioteca perteneciente al lenguaje de programación Python, que fue creada por Carl Montanari, con el objetivo de simplificar la conexión a dispositivos de red como enrutadores, conmutadores, firewalls, etc., esto ya sea a través de Telnet o SSH. [13]

Dentro de sus características, destacan su flexibilidad, ya que permite seleccionar tanto el tipo y protocolo de transporte con el que se desee trabajar, a continuación, se menciona las diversas opciones de transporte disponibles con las que Scrapli opera. [14]

- system (Open SSH)
- paramiko
- ssh2
- telnet (telnetlib)
- asyncssh
- asynctelnet [14]

Otra de las características de Scrapli es que es una biblioteca multiplataforma, ya que es compatible con las siguientes plataformas:

- Cisco IOS-XE
- Cisco NX-OS
- Juniper JunOS
- Cisco IOS-XR
- Arista EOS [13]

Como casos de uso de esta biblioteca se puede mencionar la realización de copias de seguridad de las configuraciones, el envío de nuevas configuraciones a los dispositivos de red, y posteriormente la validación de dichas configuraciones, mediante la ejecución remota de comandos show. [13]

2.6. Análisis de rendimiento de las bibliotecas de automatización de redes

Para realizar el análisis de rendimiento de las bibliotecas de automatización de redes estudiadas anteriormente, se utilizó la herramienta Python timeit(), el cual es un método que nos ayuda a calcular el tiempo que se demora en ejecutarse una determinada fracción de código. [21]

2.6.1. Pruebas de escenario

Como se menciona en [6], se realizaron las pruebas en dos escenarios distintos, el primer escenario se da en una topología de red que cuenta únicamente con un solo dispositivo de red a ser configurado, mientras que el segundo escenario se da en una topología de red con múltiples dispositivos de red conectados.

Para poder comparar el rendimiento de las distintas bibliotecas SSH analizadas, se toma como parámetro el tiempo de ejecución de cada una de ellas, para lo cual se utiliza el método timeit() dentro del script llamado test_time.py, el cual mandándolo a ejecutarse, nos arroja los siguientes resultados.

2.6.1.1. Primer escenario

```
-bash-4.2$ python test_time.py
Single Device

 scrapli: 14.62141122808592
 ssh2: 10.677067372016609
 paramiko: 11.580183147045318
 netmiko: 7.1105942610302
```

Figura 13. Resultados primer escenario [6]

Explicación de resultados: Como se puede apreciar, muy claramente en base a los datos de la figura 13, la biblioteca Netmiko es la que presenta el tiempo de ejecución más pequeño cuando se trabaja con un solo dispositivo de red.

2.6.1.2. Segundo escenario

```
Multiple Devices in one server

 scrapli: 207.880212849995587
 ssh2: 54.365094934997614
 paramiko: 62.12168894999195
 netmiko: 31.02830006496515
-bash-4.2$
```

Figura 14. Resultados segundo escenario [6]

Explicación de resultados: En base a los datos obtenidos en la figura 14, nuevamente se puede comprobar que la biblioteca Netmiko es la que brinda el menor tiempo de ejecución, ahora inclusive trabajando con múltiples dispositivos de red.

2.7. Desarrollo de la topología

La topología desarrollada está conformada por un Ubuntu Docker Guest, que es donde se ejecuta la instalación de los paquetes necesarios para que funcione la librería Netmiko así como los scripts desarrollados para ejecutar la automatización de todos los dispositivos de red que conforman al prototipo de red propuesta, en este caso un router Cisco c7200, dos switches tipo IOU, los cuales presentan las siguientes características: IOU1(Cisco IOU L2 15.1g) e IOU2(Cisco IOU L2 15.1a), también se cuenta con un conmutador ethernet que se encuentra conectado a una nube NAT que es la encargada de proporcionarnos el acceso a Internet.

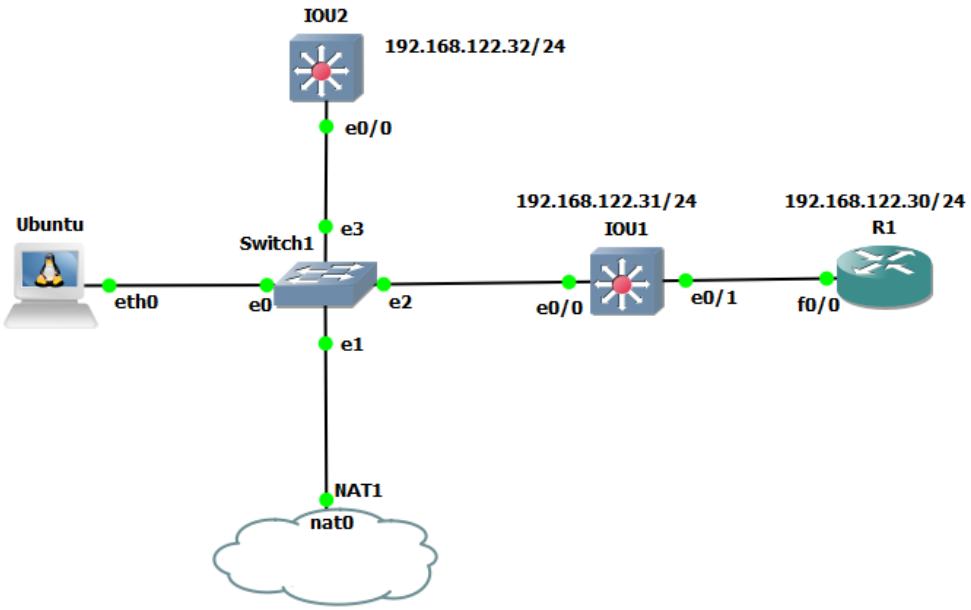


Figura 15. Topología de red implementada

Gracias a las configuraciones, realizadas en los dispositivos de red (R1, IOU1 y IOU2), adjuntas en el anexo I (Manual de usuario), podremos ahora conectarnos vía SSH a dichos dispositivos, cabe mencionar nuevamente que se habilita la conexión SSH, debido a que la biblioteca Netmiko, utiliza dicho protocolo con el objetivo de poder aplicar las distintas configuraciones de forma remota a todos los dispositivos conectados a la red, estos dispositivos presentan las siguientes direcciones IP:

- R1:192.168.122.30
- IOU1: 192.168.122.31
- IOU2: 192.168.122.32

2.7.1. Scripts desarrollados

2.7.1.1. Primer Script

Para este primer escenario, procederemos a trabajar con dos dispositivos de red, los cuales son IOU1 e IOU2, por lo cual en primera instancia procedemos a importar la biblioteca netmiko, luego creamos dos variables, es decir una por cada dispositivo que deseemos configurar, por lo cual creamos las variables llamadas iosIOU1 e iosIOU2, dichas variables almacenan la información del dispositivo que se desea configurar en este caso los dispositivos IOU1 y IOU2 respectivamente (tipo de dispositivo, dirección ip, nombre de usuario y contraseña), estos datos fueron configurados en el anexo I (Manual de usuario).

A continuación, procedemos a crear una nueva variable a la cual llamaremos all_devices, en la cual se especifica la lista de dispositivos que se van a configurar, luego utilizando un lazo for, se procede en primer lugar a crear una lista de comandos de configuración para una interfaz de loopback, para luego ser enviados a ambos dispositivos, y que sus resultados se muestren en pantalla, a continuación, y nuevamente aplicando un lazo for se procede a crear un rango de vlans determinado.

Finalmente se envían los comandos de visualización show ip int brief y show vlan brief, con el objetivo de poder observar que todas las configuraciones anteriores se hayan realizado exitosamente.

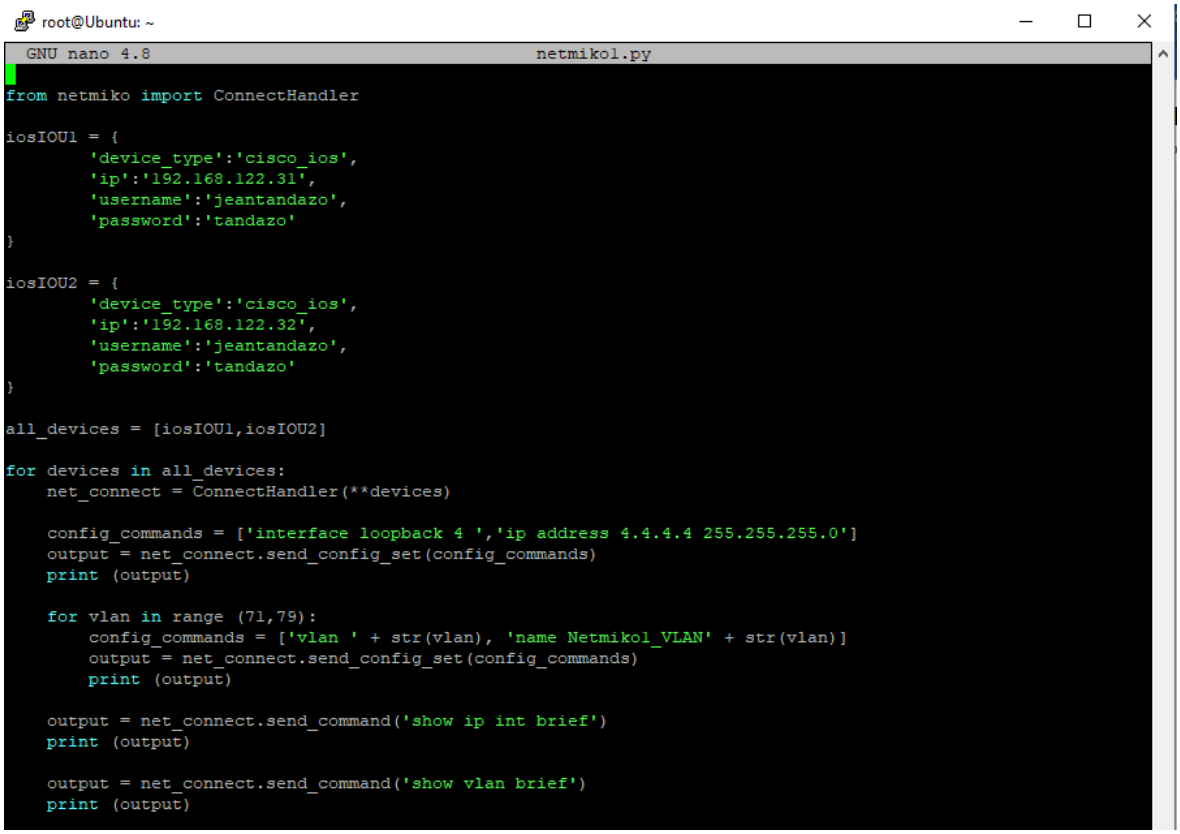


Figura 16. Script netmiko1.py

2.7.1.2. Segundo script

Para este segundo escenario, procedemos a trabajar con los tres dispositivos de red configurables existentes en la red, los cuales son: R1, IOU1 e IOU2, y al igual que en script anterior, en primera instancia procedemos a importar la biblioteca netmiko, y posteriormente creamos una variable por cada dispositivo que deseemos configurar, por lo cual en este caso creamos tres variables llamadas iosR1, iosIOU1 e iosIOU2, dichas variables almacenan la información del dispositivo que se desea configurar en este caso los dispositivos R1, IOU1 y IOU2 respectivamente.

A continuación, se procede dentro del script a añadir dos fragmentos de código que permiten abrir y por ende ejecutar archivos donde se almacenan distintos comandos de configuración, para este caso como tenemos dos tipos de dispositivos en primera instancia un Router(R1) y luego dos switches (IOU1 e IOU2), por lo cual se crean dos archivos de configuración distintos, el archivo de configuración para R1 se lo ha nombrado como configR1, mientras que al archivo de configuración para IOU1 e IOU2 es nombrado como configIOU, sus contenidos se los puede apreciar en la figura 18 y la figura 19,

respectivamente, cabe recalcar que el contenido de estos archivos pueden variar según la configuración que los usuarios deseen generarle a los dispositivos.

Finalmente se escriben diversos comandos show que nos permitirán visualizar o chequear el estado de nuestra red y comprobar que todas las configuraciones establecidas dentro del script creado, se hayan ejecutado correcta y exitosamente.

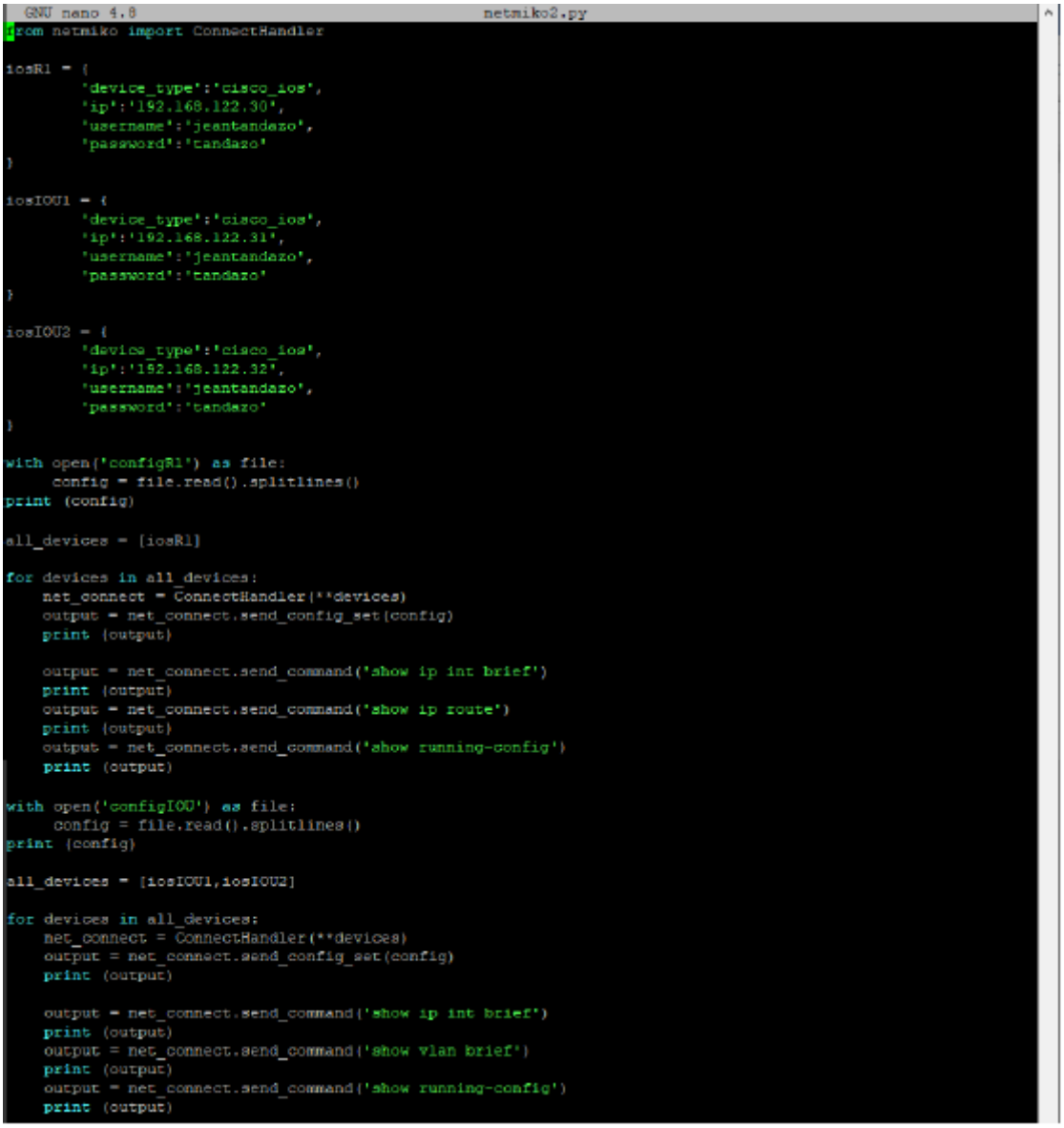


Figura 17. Script netmiko2.py

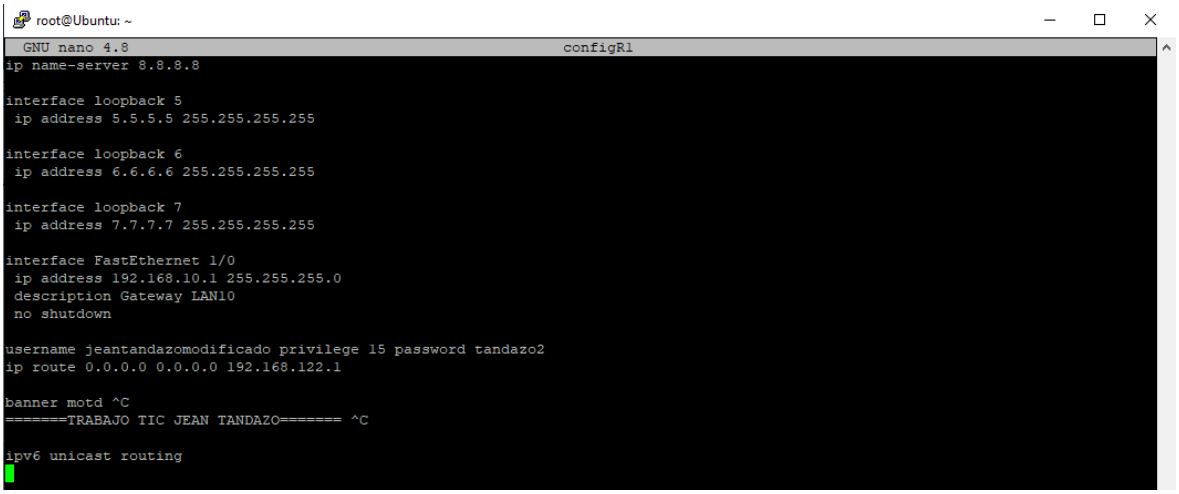


Figura 18. Archivo configR1 con los comandos de configuración para el dispositivo R1 para ser ejecutado dentro del Script netmiko2.py

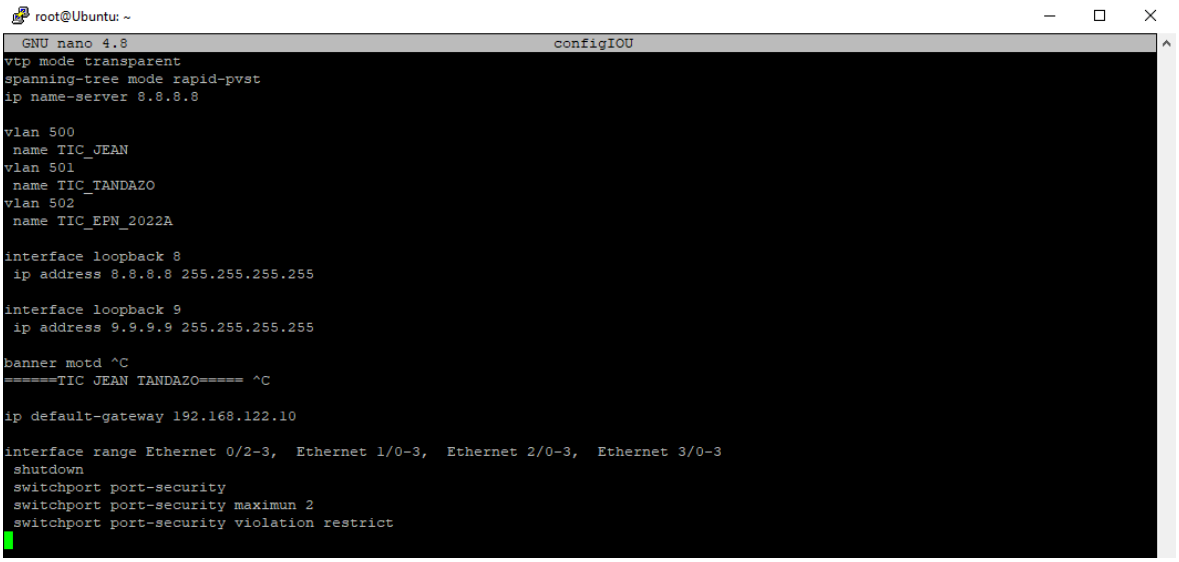


Figura 19. Archivo configIOU con los comandos de configuración para los dispositivos IOU1 e IOU2 para ser ejecutado dentro del Script netmiko2.py

3. RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

3.1. Resultados

3.1.1. Primer Script

Una vez ejecutado el Script netmiko1.py, procedemos a observar en primer lugar que en la figura 20, se nos muestra en pantalla la configuración especificada para la creación de la interfaz de loopback, posteriormente observamos la configuración y creación del rango de vlans solicitadas, las cuales se especifican dentro del lazo for del script.

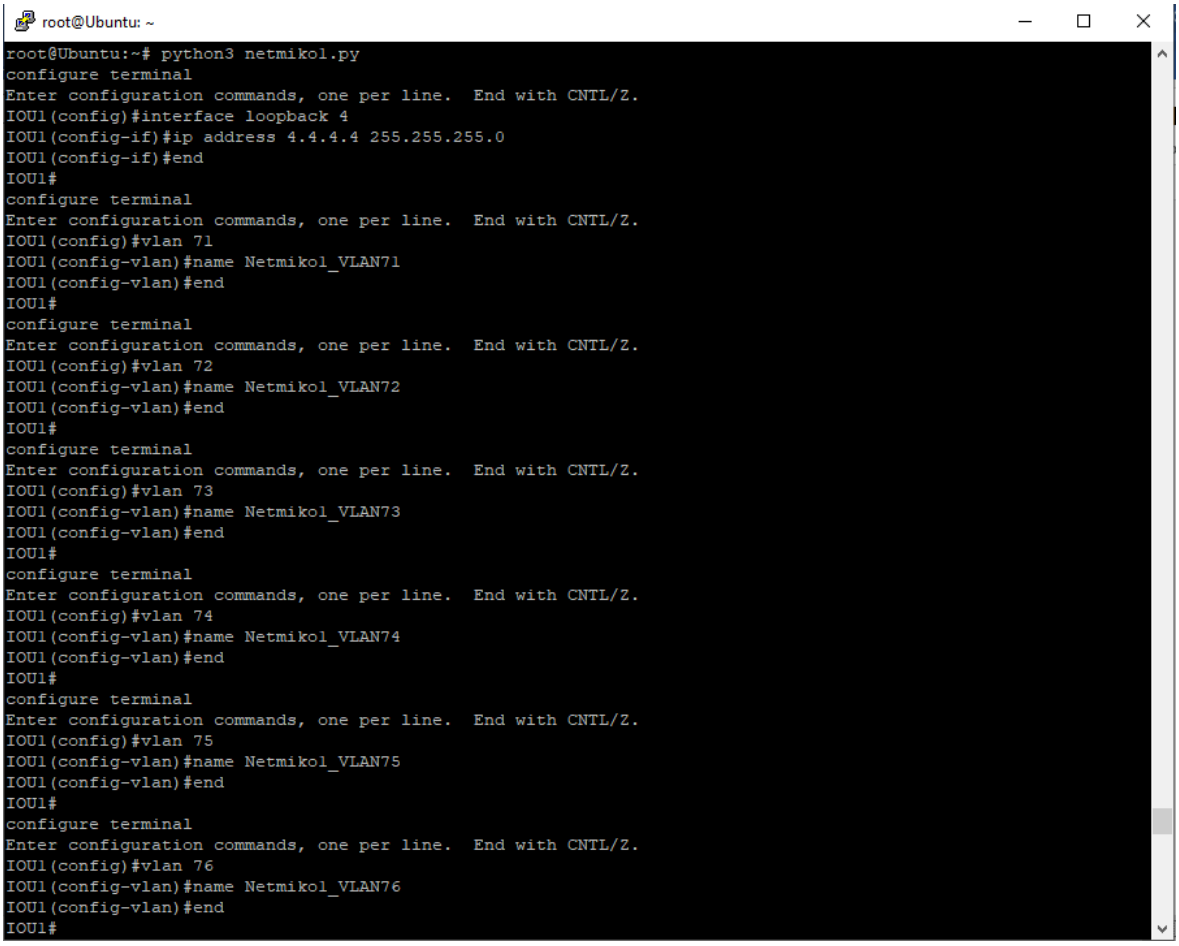


Figura 20. Configuración de la interfaz loopback y vlans solicitadas para el dispositivo IOU1

Luego de haberse creado todas las vlans solicitadas, en la figura 21 podemos observar en pantalla la información de las interfaces existentes en el dispositivo IOU1, esto debido a que dentro del script se programó la ejecución del comando “show ip int brief”, a continuación se puede apreciar que el rango de vlans solicitadas dentro del script se crearon exitosamente dentro del dispositivo IOU1, lo cual se puede apreciar debido a la ejecución del comando “show vlan brief” que igualmente se programó dentro del script.

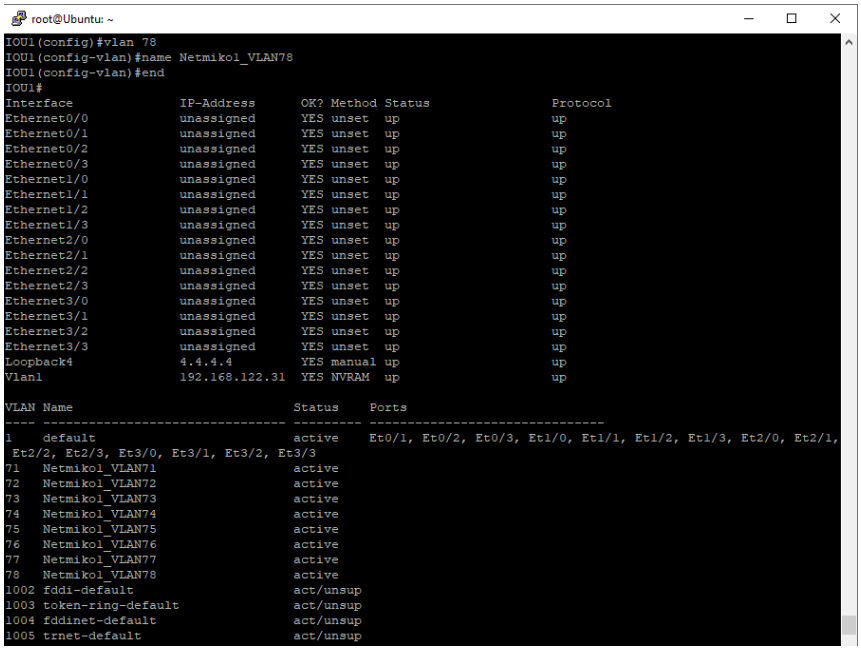


Figura 21. Resultados de los comandos "show ip int brief" y "show vlan brief" para el dispositivo IOU1

Una vez terminado todos los comandos programados dentro del script para el dispositivo IOU1, inmediatamente el programa procede con la configuración del dispositivo IOU2, obteniendo los mismos resultados que para el dispositivo IOU1, lo cual se puede apreciar en la figura 22 y figura 23, esto nos demuestra que el script generado fue ejecutado correctamente y libre de errores.

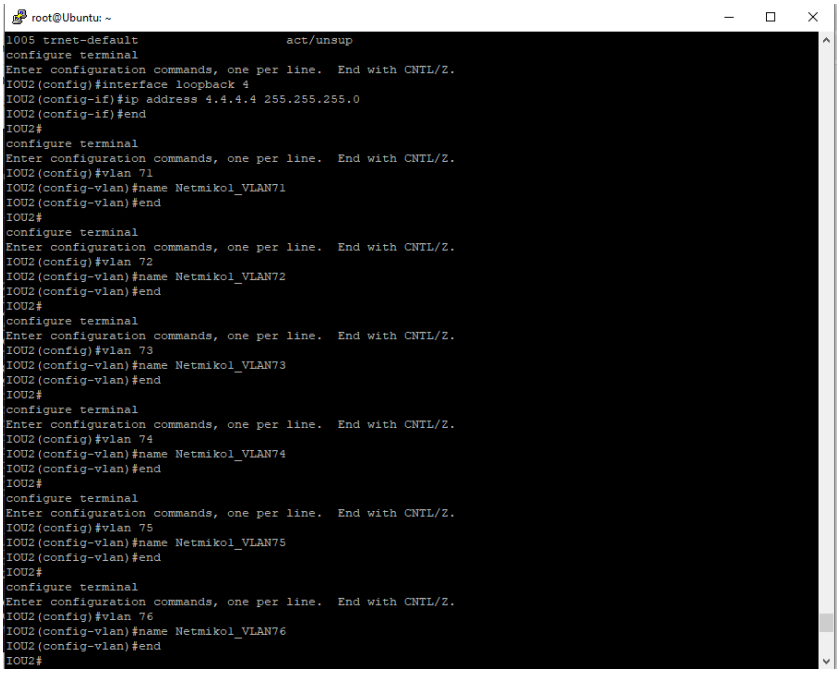


Figura 22. Configuración de la interfaz loopback y vlans solicitadas para el dispositivo IOU2

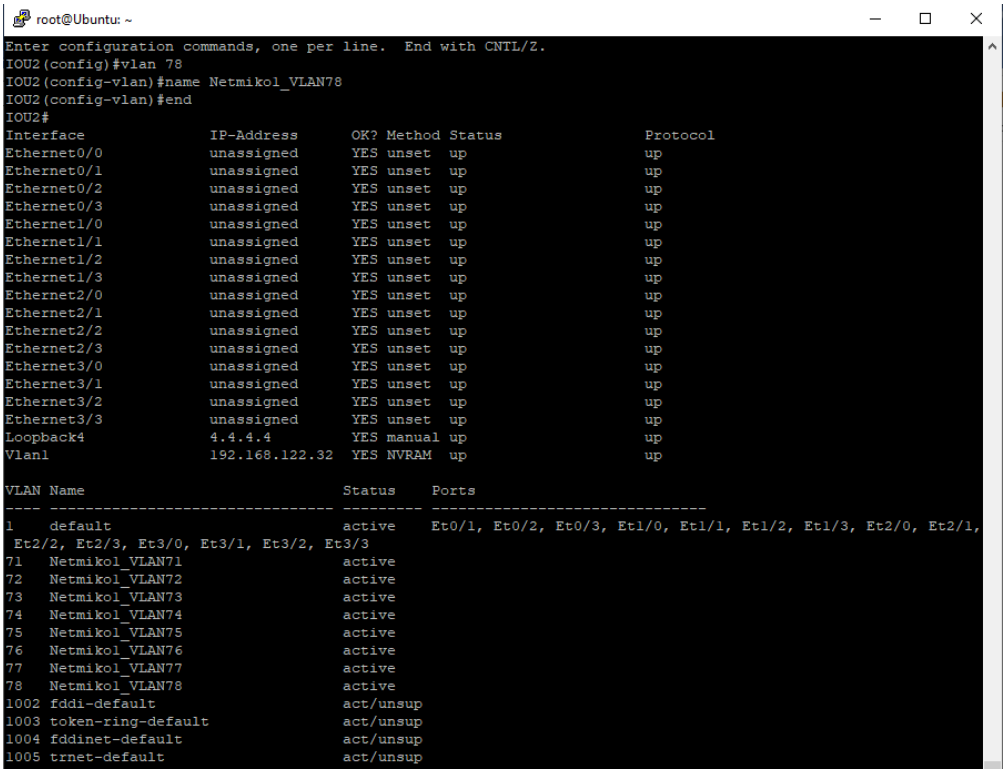


Figura 23. Resultados de los comandos "show ip int brief" y "show vlan brief" para el dispositivo IOU2

3.1.2. Segundo Script

Al mandar a ejecutar el Script netmiko2.py, podemos darnos cuenta que todos los comandos que fueron escritos dentro del archivo configR1**,** empiezan a colocarse como una lista o un arreglo, para a continuación proceder a aplicarlos al dispositivo R1, lo cual se puede observar en la figura 24. A continuación en la figura 25, podemos apreciar la continuación de los comandos de configuración escritos dentro del archivo configR1, y se empieza a visualizar el resultado de la ejecución de los comandos “show ip int brief y show ip route”, para finalmente en la figura 26 apreciar los resultados obtenidos del último comando show programado, el cual es “show running-config”.

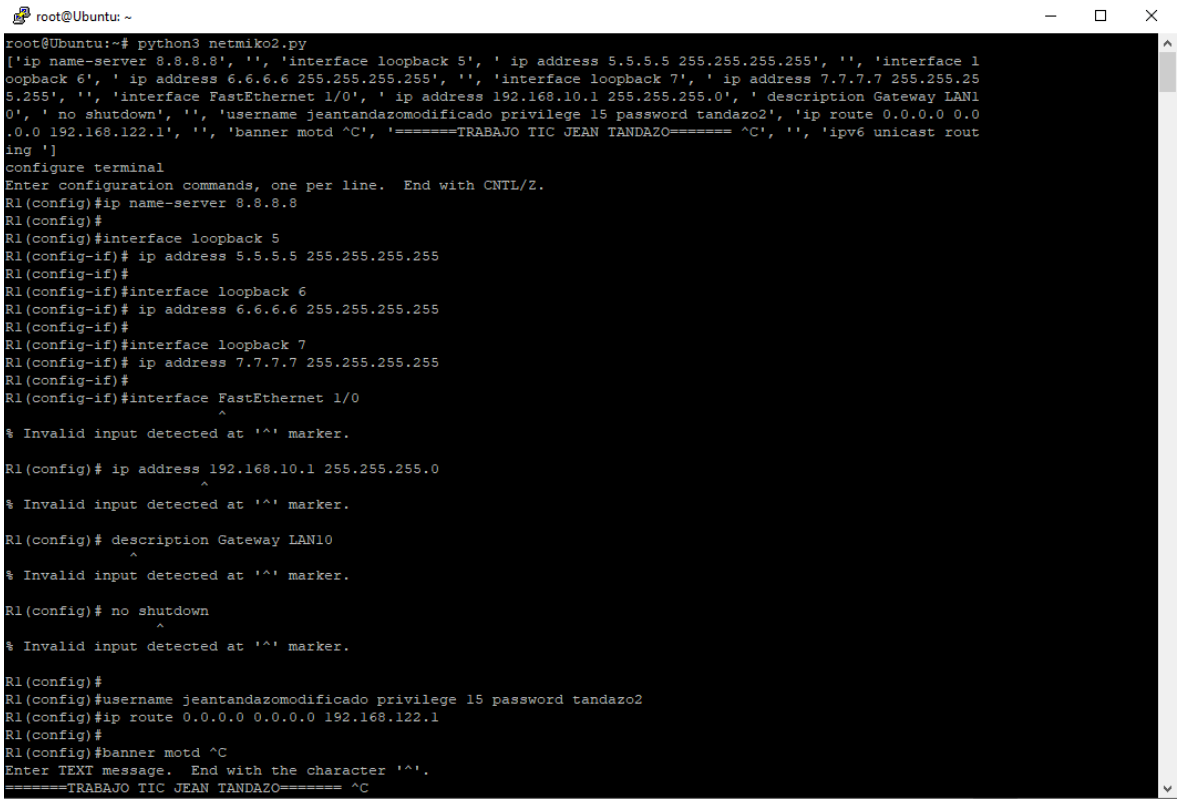


Figura 24. Inicio de la configuración del dispositivo R1, mediante el archivo configR1

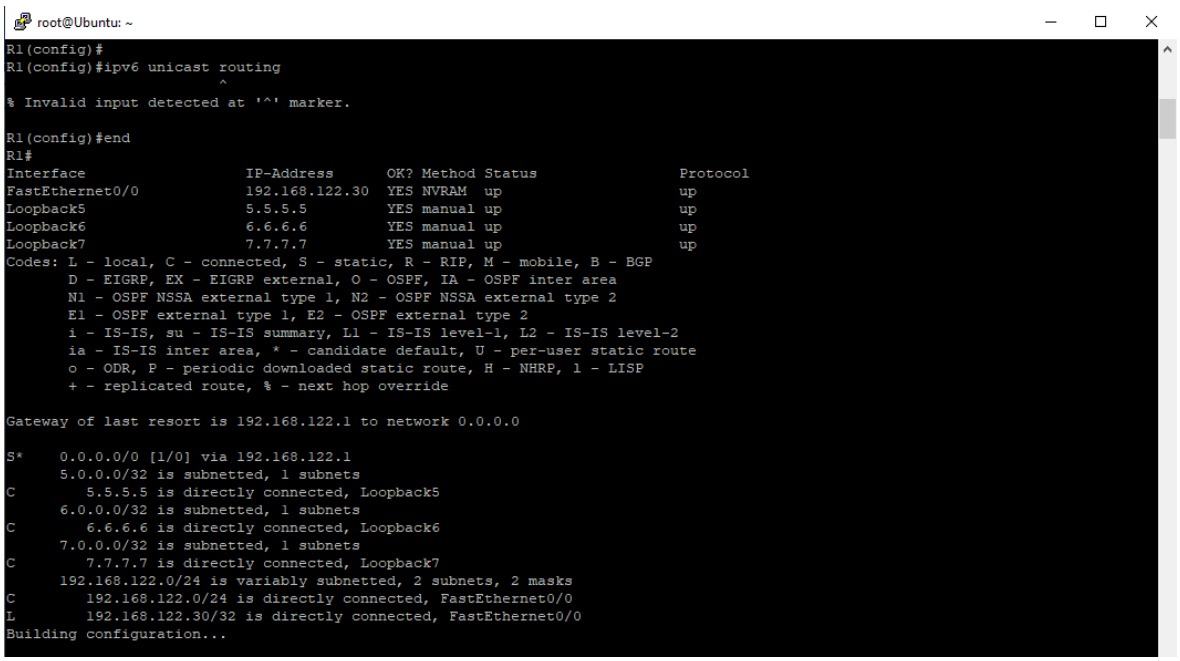


Figura 25. Resultados obtenidos al ejecutar los comandos de visualización “show ip int brief y show ip route”

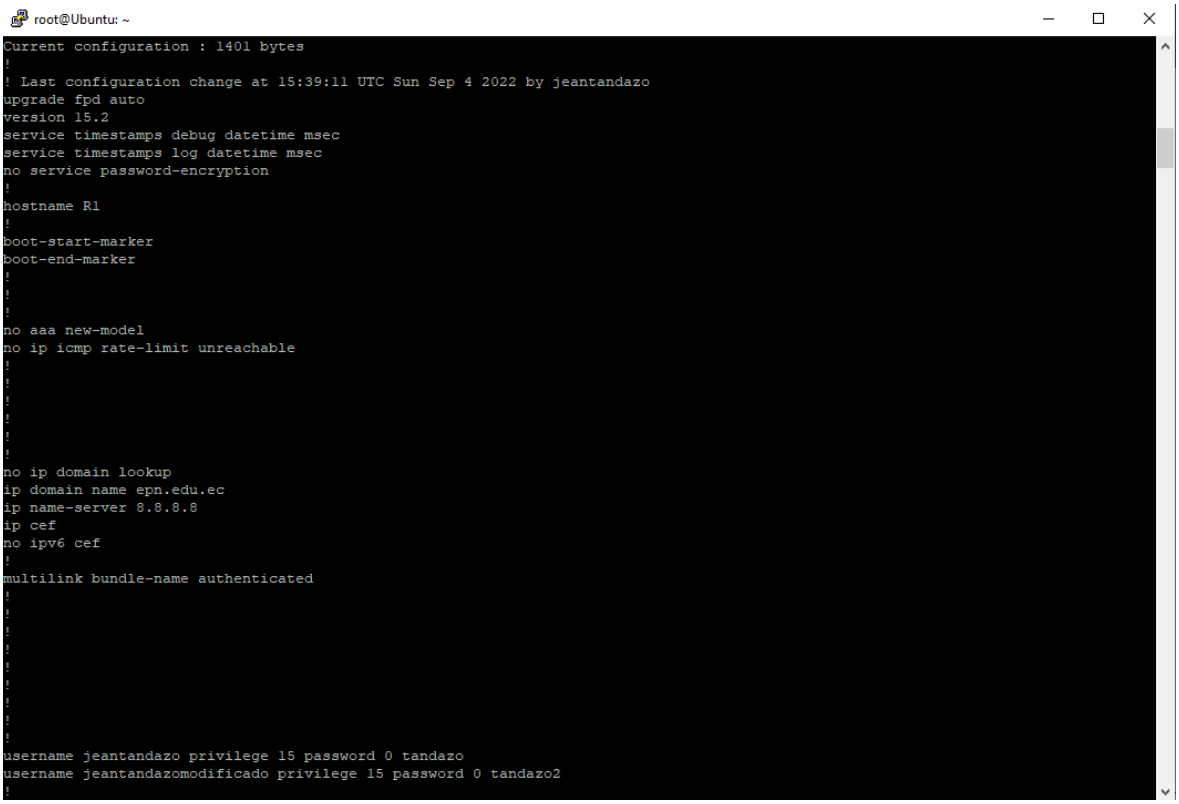


Figura 26. Resultados obtenidos al ejecutar el comando de visualización “show running-config"

Una vez terminadas todas las acciones programadas mediante el script netmiko2.py para el dispositivo R1, el programa empieza a leer el archivo configIOU, que contiene las configuraciones para los dispositivos IOU1 e IOU2, y esto se puede apreciar ya que las configuraciones escritas dentro del archivo configIOU empiezan a colocarse como una lista o un arreglo, y posteriormente empieza con la configuración de los dispositivos, en primer lugar el programa ejecuta las configuraciones respectivas para el dispositivo IOU1, lo cual se aprecia en la figura 27.

Una vez terminadas las configuraciones solicitadas y para comprobar que estas fueron exitosas dentro del script se programó la ejecución de diversos comandos show, los cuales son: show ip int brief, show vlan brief y finalmente show running-config.

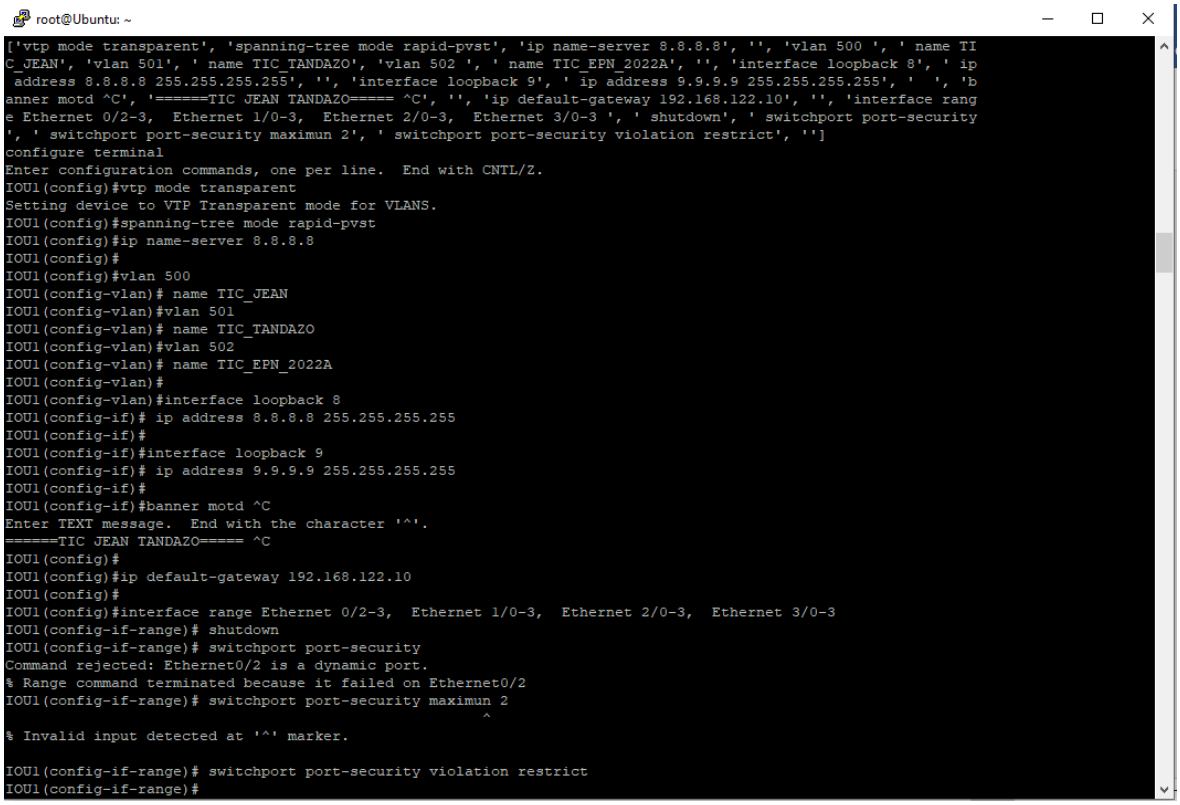


Figura 27. Configuración del dispositivo IOU1, mediante el archivo ConfigIOU

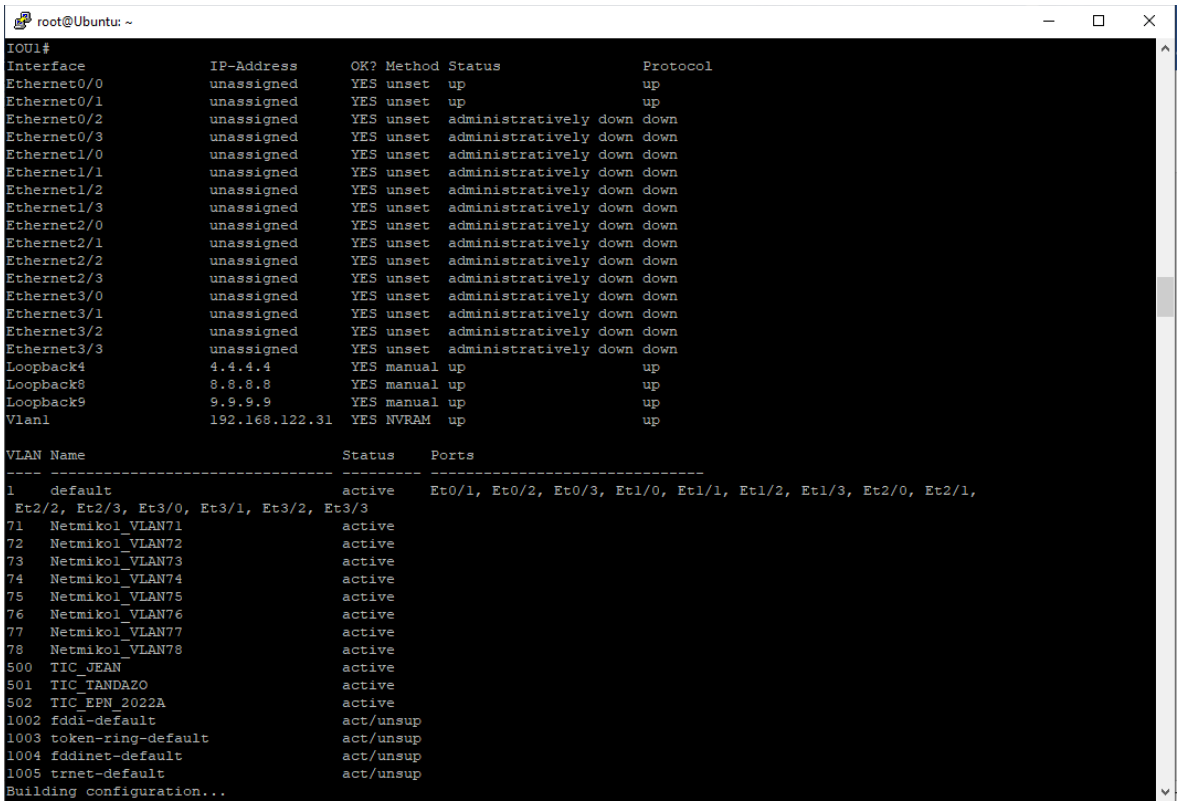


Figura 28. Ejecución de los comandos "show ip int brief" "show vlan brief" para IOU1

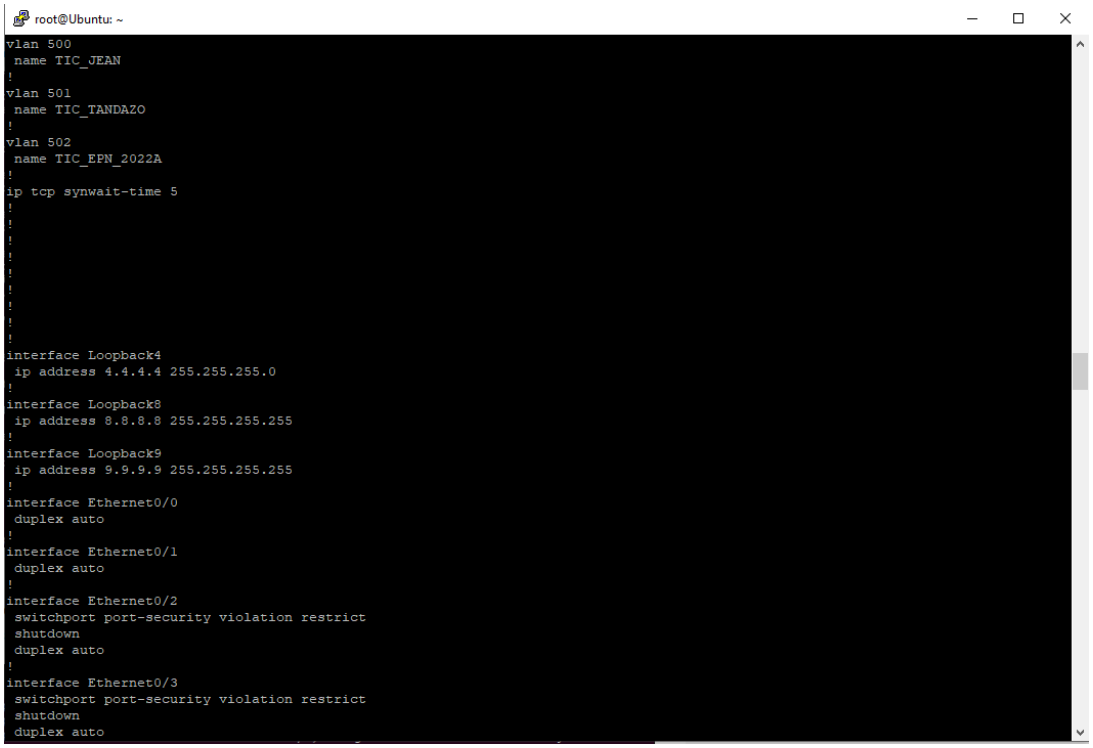


Figura 29. Ejecución del comando "show running-config" para IOU1

Una vez finalizadas las tareas para el dispositivo IOU1, empieza instantáneamente la configuración del dispositivo IOU2, lo cual se puede observar en la figura 30, y al igual que para el dispositivo IOU1, para IOU2 mediante la ayuda de los comandos show mencionados anteriormente, podemos chequear que todas las configuraciones solicitadas se efectuaron con éxito.

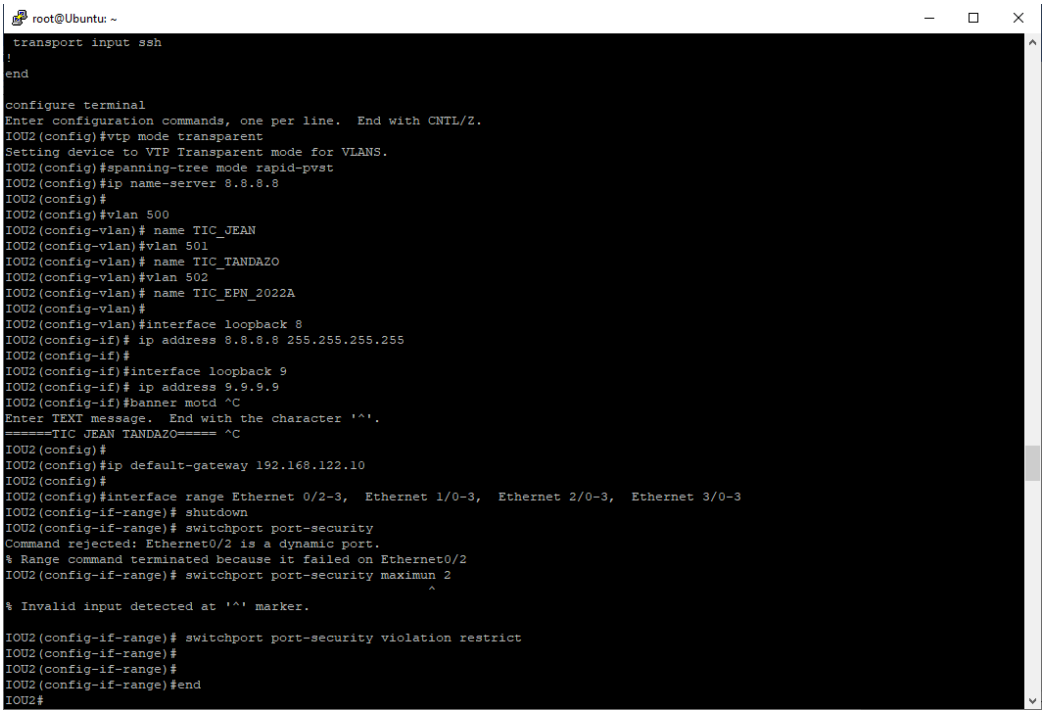


Figura 30. Configuración del dispositivo IOU2, mediante el archivo ConfigIOU

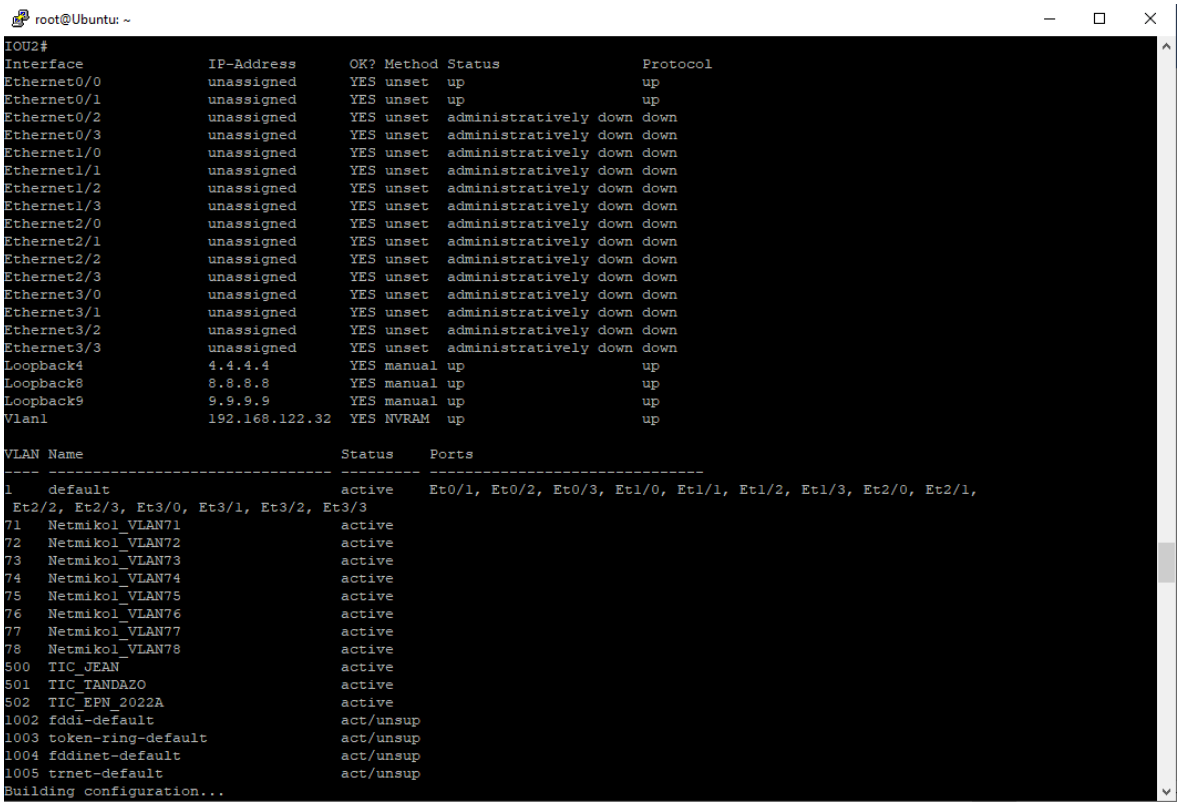


Figura 31. Ejecución de los comandos "show ip int brief" "show vlan brief" para IOU2

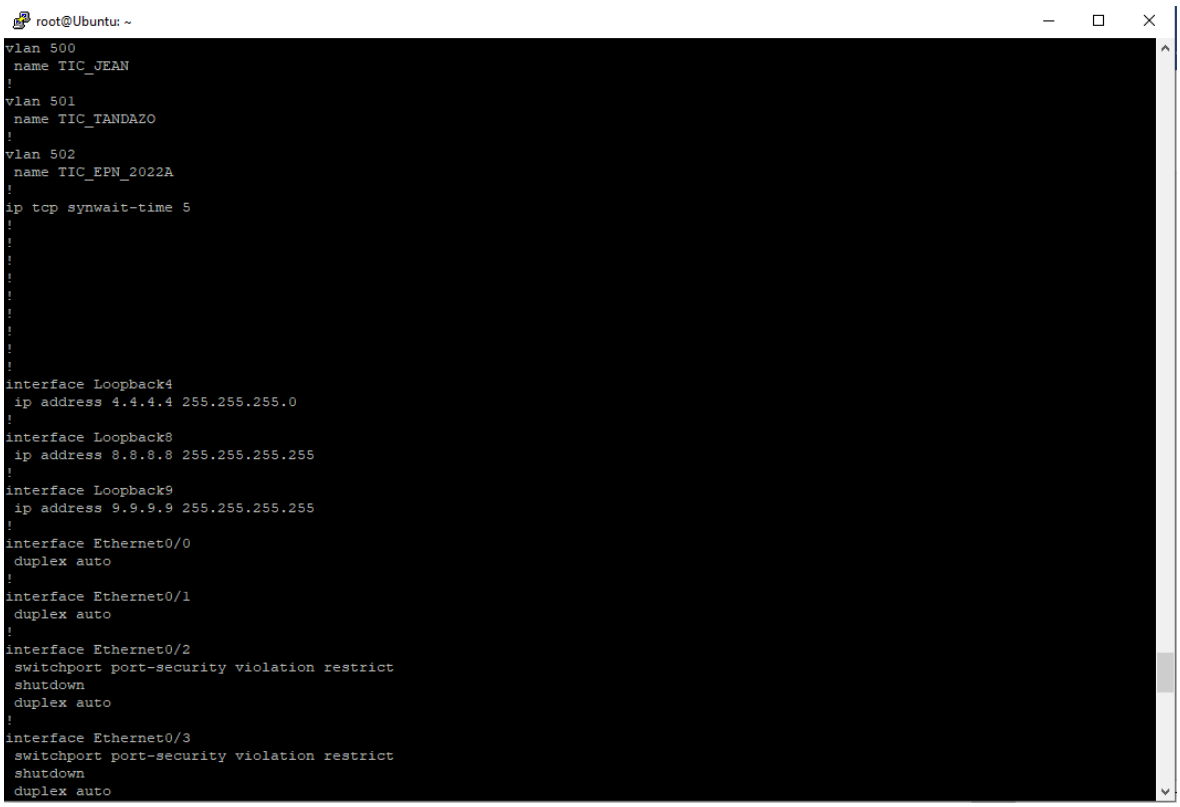


Figura 32. Ejecución del comando "show running-config" para IOU2

3.2. Conclusiones

- La automatización de redes es sin lugar a dudas el presente y futuro dentro del campo de las TI, ya que la misma puede implementarse en redes de área local, extendida, redes de centros de datos, redes en la nube e inclusive en redes inalámbricas, por lo cual su implementación en la actualidad dentro del ámbito educativo, pero sobre todo en el campo empresarial debe ser algo casi obligatorio, ya que como se pudo apreciar dentro del trabajo TIC, la automatización de la redes les brinda a las organizaciones un sin número de mejoras, lo cual se traduce principalmente en réditos económicos para las empresas.
- Existen múltiples bibliotecas SSH y lenguajes de programación que permiten a los usuarios ejecutar la automatización de redes, pero gracias a la realización del presente trabajo TIC, se pudo comprobar que la herramienta Netmiko presenta los mejores resultados trabajando con uno o varios dispositivos de red, y dicha herramienta de la mano del lenguaje de programación Python, son la mejor combinación para trabajar dentro de este campo, todo esto debido a que ambas herramientas presentan una gran versatilidad y múltiples ventajas frente a las demás opciones existentes.
- Como se pudo apreciar en la parte práctica más específicamente en el primer script, se pueden agregar dentro del mismo de manera directa los diversos comandos de configuración y visualización de todos los elementos de la red que se desee configurar o chequear su estado, pero lo ideal y lo que más se utiliza dentro del campo profesional es lo realizado dentro del segundo script, donde se crea diferentes archivos que contienen los diversos comandos de configuración y chequeo o visualización que se desee ejecutar en los dispositivos, y adicionalmente dentro del script principal se utiliza comandos que permitan la lectura y ejecución de los diferentes archivos, cabe mencionar que dichos archivos pueden contener la cantidad de comandos que el usuario crea conveniente.
- El software GNS3 es sin duda alguna una de las opciones preferidas no solo por los estudiantes sino también por los profesionales de TI al momento de trabajar en el campo de la automatización de redes, ya que esta herramienta brinda resultados bastante cercanos a lo que se obtendría si se trabajara con equipos físicos, y es por esta razón que se eligió este software para la realización del presente TIC, dentro de la cual se trabajó con equipos de la marca Cisco, pero cabe recalcar que tanto la herramienta Netmiko, el lenguaje de programación Python y el software GNS3, tienen la gran virtud de poder operar también con equipos de otros fabricantes.

3.3. Recomendaciones

- Se recomienda realizar la implementación de la automatización de redes tanto en las instituciones educativas como en el ámbito laboral ya que como se pudo observar durante la realización del trabajo TIC, la automatización de redes presenta múltiples ventajas en comparación al método manual que se utiliza actualmente.
- Como se mencionó durante la realización del TIC, existen varias bibliotecas SSH y lenguajes de programación para realizar la automatización de redes, pero se recomienda utilizar la biblioteca Netmiko, junto con el lenguaje de programación Python, debido a que estas herramientas en conjunto, permiten a los usuarios realizar un sin número de tareas de una manera rápida y sencilla.
- Se recomienda también tener instalado las versiones más actuales de los distintos softwares que se desee utilizar, para de esta manera poder obtener mejores resultados.
- Es recomendable en primer lugar familiarizarse y tener muy claro que función realizan los comandos más utilizados por la herramienta que se desee utilizar antes de empezar con la creación de los distintos scripts.
- Se recomienda también investigar la información necesaria en las páginas oficiales de las herramientas que se desee usar, las cuales por lo general al ser de código abierto tienen un repositorio en la plataforma GitHub, en donde se puede encontrar ejemplos y también solución a diversos errores que se pueden presentar durante la realización de las distintas actividades.
- Adicionalmente se recomienda también investigar las características de los equipos de red que se van a utilizar para la realización de la topología, ya que en muchas ocasiones algunos de estos equipos no son compatibles con varios de los protocolos que se desea implementar.

4. REFERENCIAS BIBLIOGRÁFICAS

[1] K. Byers, «Python For Network Engineers,» 30 Septiembre 2021. [En línea]. Available: https://pynet.twb-tech.com/blog/netmiko-python-library.html. [Último acceso: 01 Julio 2022].

[2] R. Donato, «Packet Cooders,» 2022. [En línea]. Available:]https://www.packetcoders.io/netmiko-the-what-and-the-why/. [Último acceso: 01 Julio 2022].

[3] A. Boumezrag, «A Proposed of Novel Network Management Platform for Network Automation and Programmability with Implementation on GNS30,» *University of Mohamed Khider Biskra0,* p. 105, 2020.

[4] K. Byers, «Github,» 09 Agosto 2022. [En línea]. Available: https://github.com/ktbyers/netmiko/blob/develop/PLATFORMS.md. [Último acceso: 01 Julio 2022].

[5] K. Byers, «Github,» 2022. [En línea]. Available: https://ktbyers.github.io/netmiko/docs/netmiko/base_connection.html. [Último acceso: 01 Julio 2022].

[6] K. Solani y V. Kiran, «Performance Comparison of SSH Libraries,» *University of Shanghai for Science and Technology,* vol. 23, 06 Junio 2021.

[7] Anonimo, «Programador Clic,» 2022. [En línea]. Available: https://programmerclick.com/article/52721567167/. [Último acceso: 04 Julio 2022].

[8] A. Lima, «PYTHON: INSTALE PARAMIKO EN WINDOWS Y LINUX,» 2022. [En línea]. Available: https://es.acervolima.com/python-instale-paramiko-en-windows-y-linux/. [Último acceso: 06 Julio 2022].

[9] J. Forcier, «Paramiko,» 2022. [En línea]. Available: https://www.paramiko.org/. [Último acceso: 08 Julio 2022].

[10] P. Kittenis, «ssh2-Python,» 2017. [En línea]. Available: https://ssh2-python.readthedocs.io/en/latest/introduction.html. [Último acceso: 08 Julio 2022].

[11] P. Kitteenis, «ssh2-python 1.0.0,» 31 Julio 2022. [En línea]. Available: https://pypi.org/project/ssh2-python/. [Último acceso: 15 Julio 2022].

[12] S. Behnel, «Cython C-Extensions for Python,» 31 Julio 2022. [En línea]. Available: https://cython.org/.

[13] C. Montanari, «Packet Coders,» 2022. [En línea]. Available: https://www.packetcoders.io/introduction-to-scrapli/. [Último acceso: 15 Julio 2022].

[14] N. Samoylenko, «Python for Network engineers,» 14 Julio 2022. [En línea]. Available: https://pyneng.readthedocs.io/en/latest/book/18_ssh_telnet/scrapli.html. [Último acceso: 25 Julio 2022].

[15] A. M. Mazin, R. A. Rahman, M. Kassim y A. R. Mahmud, «Performance Analysis on Network Automation Interaction with Network Devices Using Python,» *2021 IEEE 11th IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE),* pp. 360 - 366, 2021.

[16] P. MIHĂILĂ, T. BĂLAN, R. CURPEN y F. SANDU, «Network Automation and Abstraction using Python Programming Methods,» *MACRo 2017 - 6th International Conference on Recent Achievements,* nº 95-103, 19 Octubre 2017.

[17] Anonimo, «Wifi - Libre,» 26 Octubre 2017. [En línea]. Available: https://www.wifi-libre.com/topic-911-guia-basica-conexion-por-protocolo-ssh-servidor-cliente-llaves.html. [Último acceso: 18 Julio 2022].

[18] C. D. Colliard, «Universidad Tecnologica Nacional Facultad Regional Santa Fe,» *Protocolos Criptograficos basados en los algoritmos de DIFFIE-HELLMAN Y RSA,* pp. 1-174, 2019.

[19] X. Vasco, «MAZE: A Secure Cloud Storage Service Using Moving,» *University of Pittsburgh,* pp. 1-166, 2020.

[20] P. Tim, «Python Software Foundation,» Timeit — Measure execution time of small code snippets, 20 Julio 2022. [En línea]. Available: https://docs.python.org/3/library/timeit.html. [Último acceso: 01 Agosto 2022].

[21] E. Rico Schmidt, «Python3 - module,» 2019. [En línea]. Available: https://rico-schmidt.name/pymotw-3/timeit/. [Último acceso: 01 Agosto 2022].

[22] J. Larsson, «Network Automation in a Multi-vendor,» *Luleå tekniska universitet,* pp. 1-37, 2020.

[23] K. Cajas, «Redes de sensores inalambricos para IOT: Automatizacion de Redes Inalambricas de Sensores,» *Escuela Politecnica Nacional,* p. 74, Febrero 2022.

[24] S. Dif y F. Bakhti, «Study on Python for Network Automation,» *Ziane Achour university of Djelfa,* pp. 1-70, 2019.

[25] G. Milios, «Network Automation using Python,» *International Hellenic University,* pp. 1-70, Diciembre 2020.

[26] WMware, «What is network automation?,» 2022. [En línea]. Available: https://www.vmware.com/topics/glossary/content/network-automation.html#:~:text=Network%20automation%20is%20the%20process,in%20conjunction%20with%20network%20virtualization.. [Último acceso: 01 Agosto 2022].

[27] A. Froehlich y J. Scarpati, «TechTarget,» 2022. [En línea]. Available: https://www.techtarget.com/searchnetworking/definition/network-automation. [Último acceso: 01 Agosto 2022].

[28] SolarWinds, «SolarWinds Worldwide,» 2022. [En línea]. Available: https://www.solarwinds.com/resources/it-glossary/network-automation. [Último acceso: 01 Agosto 2022].

[29] F. Engineer, «Field Engineer,» 2022. [En línea]. Available: https://www.fieldengineer.com/skills/network-automation. [Último acceso: 01 Agosto 2022].

[30] D. Garros, «Github pages,» 31 Octubre 2020. [En línea]. Available: https://dgarros.github.io/netdevops-survey/reports/2020. [Último acceso: 01 Agosto 2022].

[31] E. R y A. M, «Configuración automática de red en un entorno virtualizado usando GNS3,» *10.ª Conferencia Internacional sobre Ciencias de la Computación y Educación (ICCSE),* pp. 25-30, 2015.

[32] D. Lal N, B. Ghorbani y S. Vaghri, «A Survey on the use of GNS3 for virtualizing computer networks,» *International Journal of Computer Science,* vol. 5, pp. 1-10, 01 Diciembre 2016.

[33] K. Byers, «Github,» 27 Junio 2021. [En línea]. Available: https://github.com/ktbyers/netmiko/blob/develop/EXAMPLES.md. [Último acceso: 01 Agosto 2022].

[34] S. Vinasiththamby, «Packetswitch,» 06 Mayo 2022. [En línea]. Available: https://www.packetswitch.co.uk/netmiko-intro/. [Último acceso: 01 Agosto 2022].

[35] S. Vinasiththamby, «Packetswitch,» 07 Mayo 2022. [En línea]. Available: https://www.packetswitch.co.uk/netmiko-par2/. [Último acceso: 01 Agosto 2022].

[36] A. García, C. Rodríguez, C. Calderón y F. Casmartiño, «Controladores SDN, elementos para su seleccion y evaluacion,» *Revista digital de las tecnologias de la informacion y las comunicaciones ,* vol. 13, pp. 10-20, Diciembre 2014.

[37] A. Conde, «Automatizacion de Redes Informaticas con Python,» 03 Enero 2020.

[38] M. Ochoa, «Características de las Redes Definidas por Software (SDN),» *Universidad Catolica de Santiago de Guayaquil,* pp. 1-152, 22 Enero 2018.

[39] S. Clark, «Cisco DEVNET,» [En línea]. Available: https://devnetsandbox.cisco.com/RM/Diagram/Index/e83cfd31-ade3-4e15-91d6-3118b867a0dd?diagramType=Topology. [Último acceso: 03 09 2022].

[40] M. Alvarez, «Coding Networks,» 18 Julio 2020. [En línea]. Available: https://codingnetworks.blog/es/la-programabilidad-de-redes-y-automatizacion/. [Último acceso: 01 Agosto 2022].

5. ANEXOS

5.1. ANEXO I. Manual de instalación de la herramienta Netmiko y configuración de los equipos para la realización del desarrollo práctico del Trabajo de Integración Curricular.

ANEXO I

Desarrollo

1. Ingresamos a nuestro servidor en este caso nuestro Ubuntu Docker Guest, y dentro del mismo procedemos a ingresar al archivo de configuración con el siguiente comando:

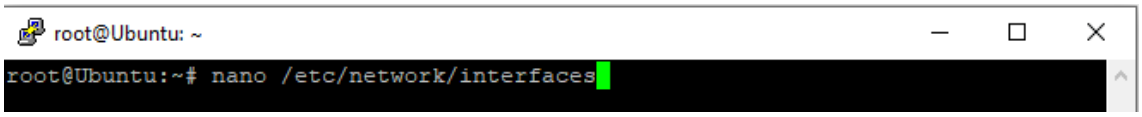


Figura 33. Comando de ingreso al archivo de configuración

2. Una vez dentro del archivo de configuración de Ubuntu Docker Guest, procedemos a configurar a la PC para que obtenga una dirección ip dinámica, descomentando las siguientes líneas de código como se puede observar en la figura 34.

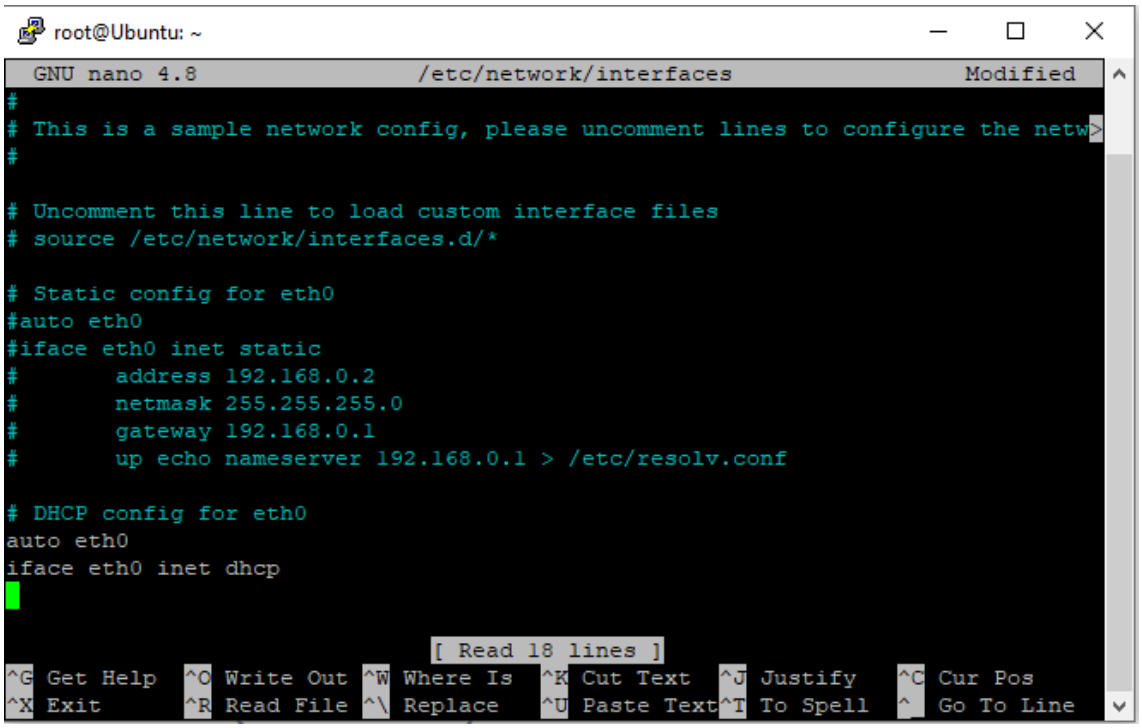


Figura 34. Configuración de ip dinámica de Ubuntu Docker Guest

3. Guardamos la configuración realizada, y para que dichos cambios se apliquen, procedemos a reiniciar el servidor, y posteriormente con la ayuda del comando ifconfig, en la figura 35, observamos lo siguiente:

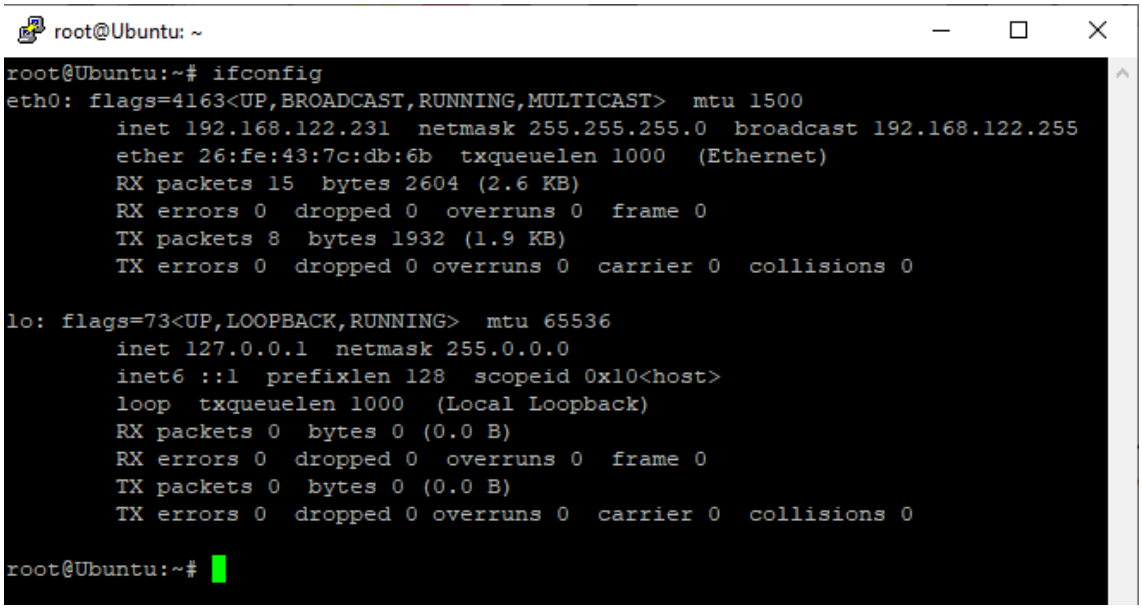


Figura 35. Dirección ip dinámica de Ubuntu Docker Guest

4. Antes de comenzar con la instalación de las librerías necesarias para realizar el trabajo TIC, debemos ejecutar los siguientes comandos.

Figura 36. Ejecución del comando apt-get update

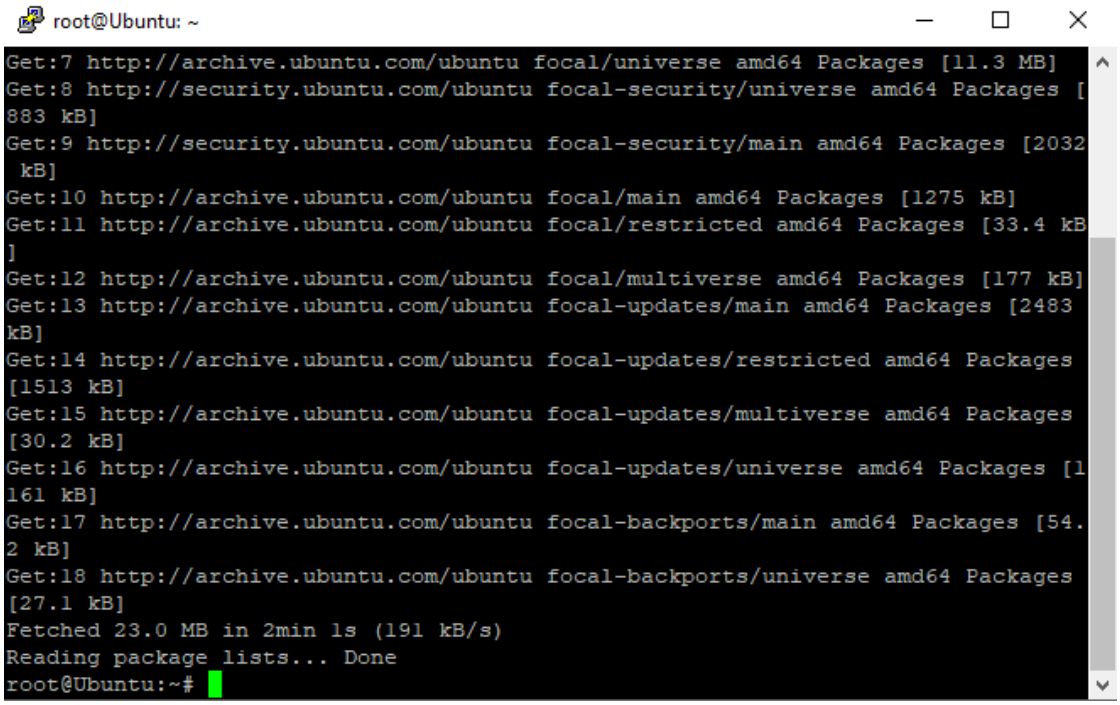


Figura 37. Instalación correcta del comando apt-get update

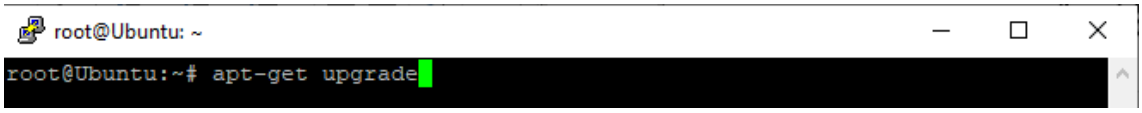


Figura 38. Ejecución del comando apt-get upgrade

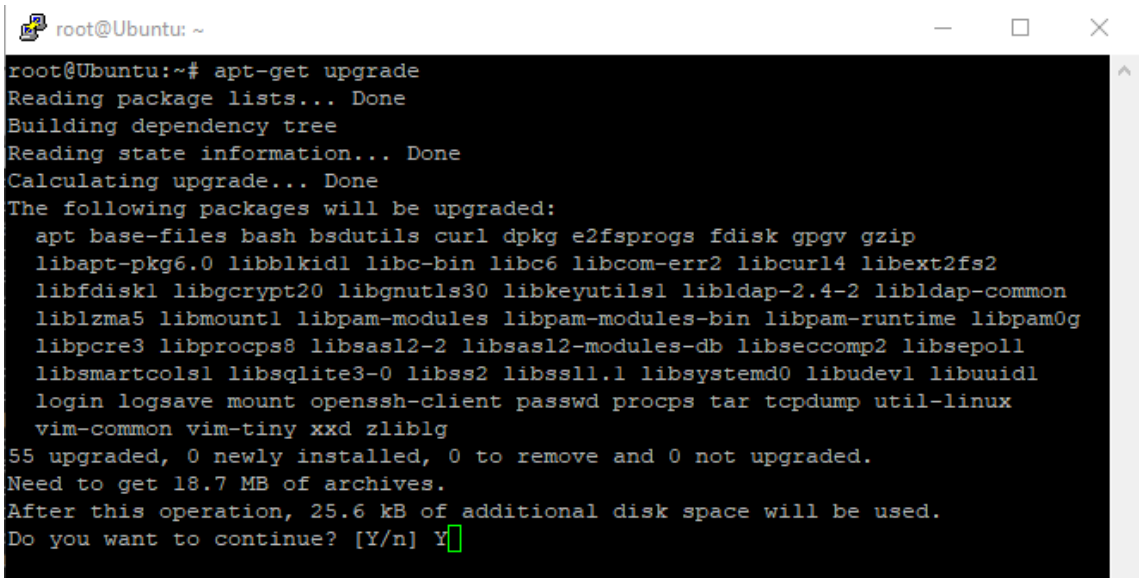


Figura 39. Selección de la opción Yes, para la correcta instalación del comando apt-get upgrade

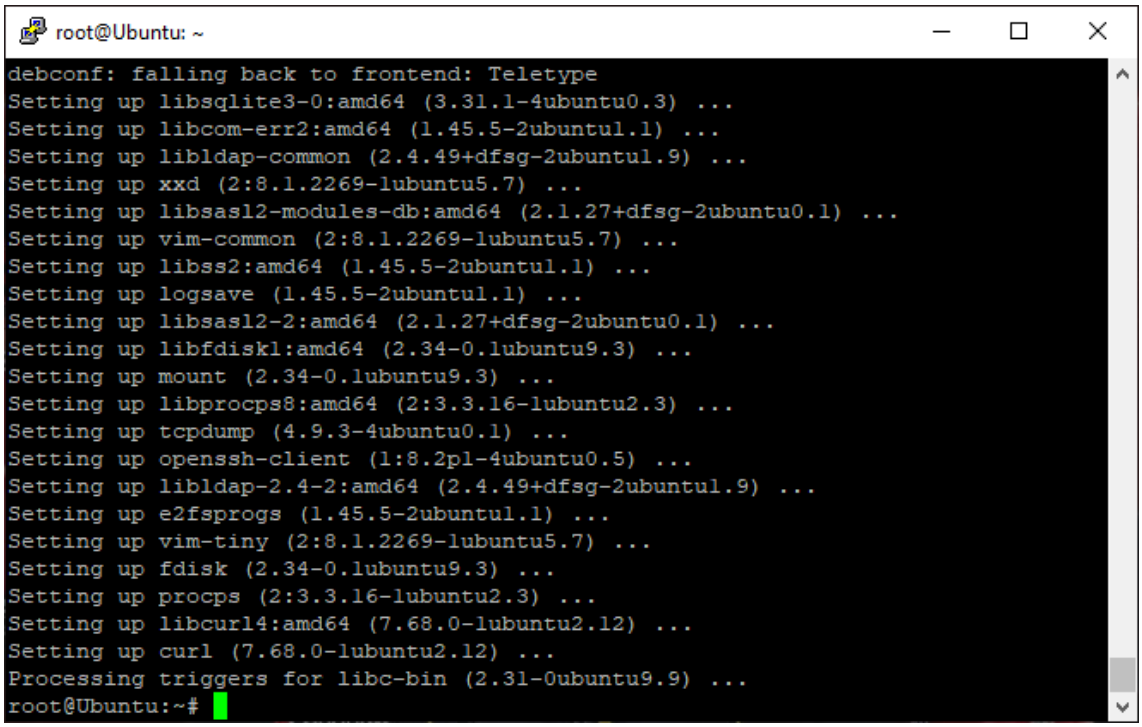


Figura 40. Instalación correcta del comando apt-get upgrade

5. Una vez instalados los paquetes anteriores, procedemos a instalar python3 y la librería Netmiko, dentro de nuestra máquina de Ubuntu Docker Guest, ejecutando los comandos que se muestran a continuación.

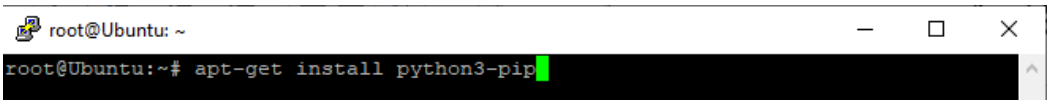


Figura 41. Ejecución del comando apt-get install python3-pip

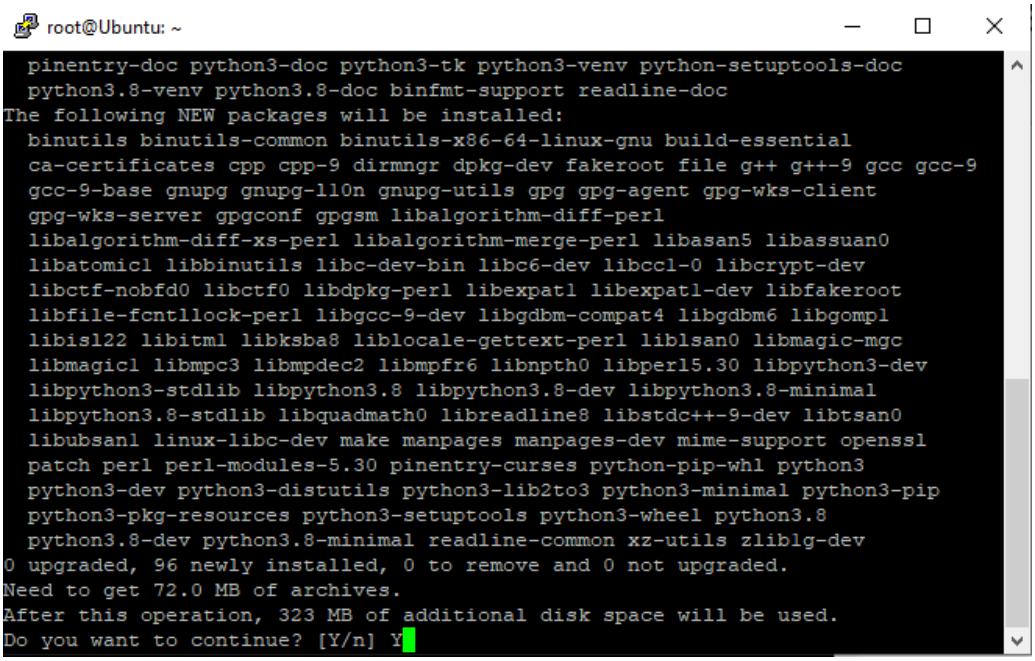


Figura 42. Selección de la opción Yes, para la correcta instalación del comando apt-get install python3-pip

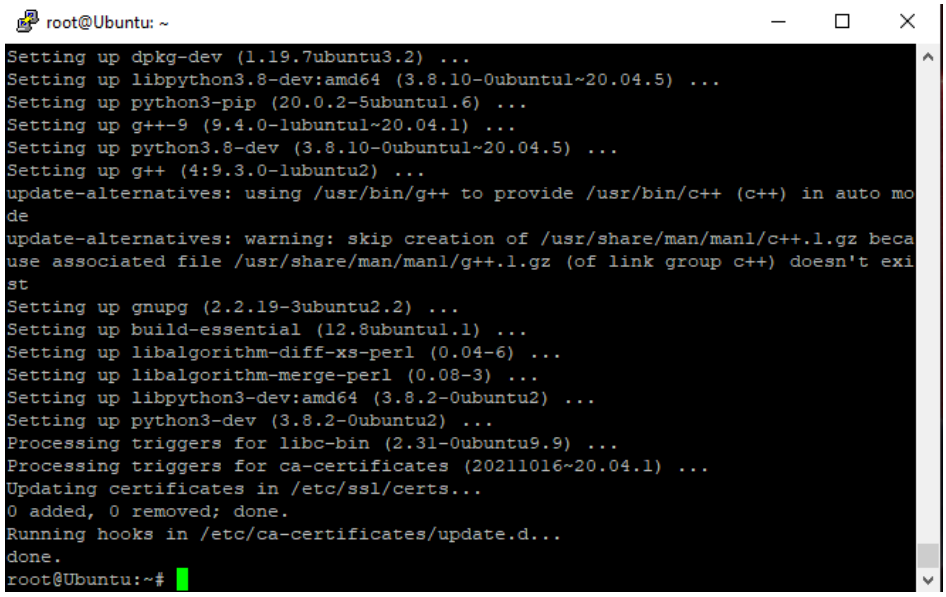


Figura 43. Instalación correcta del comando apt-get install python3-pip.

6. Ahora procedemos a instalar la biblioteca Netmiko, ejecutando el siguiente comando.

Figura 44. Ejecución del comando pip3 install -U netmiko

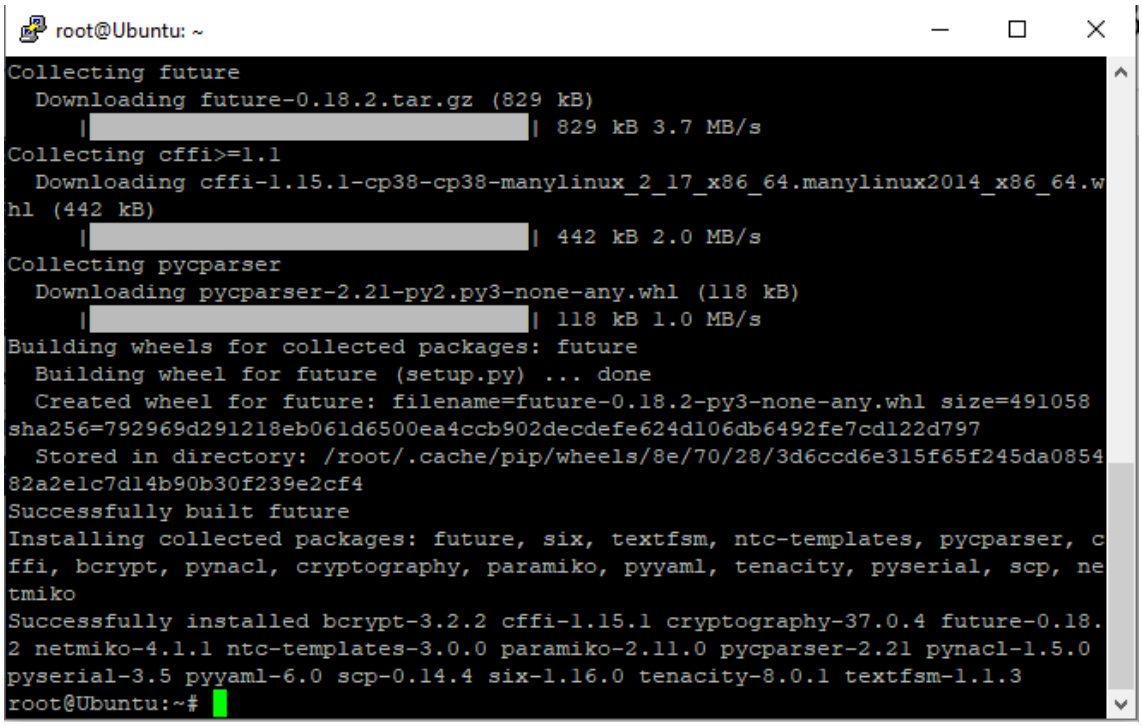


Figura 45. Instalación correcta del comando pip3 install -U netmiko

7. Una vez instalado de manera correcta la biblioteca Netmiko, se procede a ejecutar los siguientes comandos para conocer si los paquetes pip y setuptools, se encuentran actualizados.

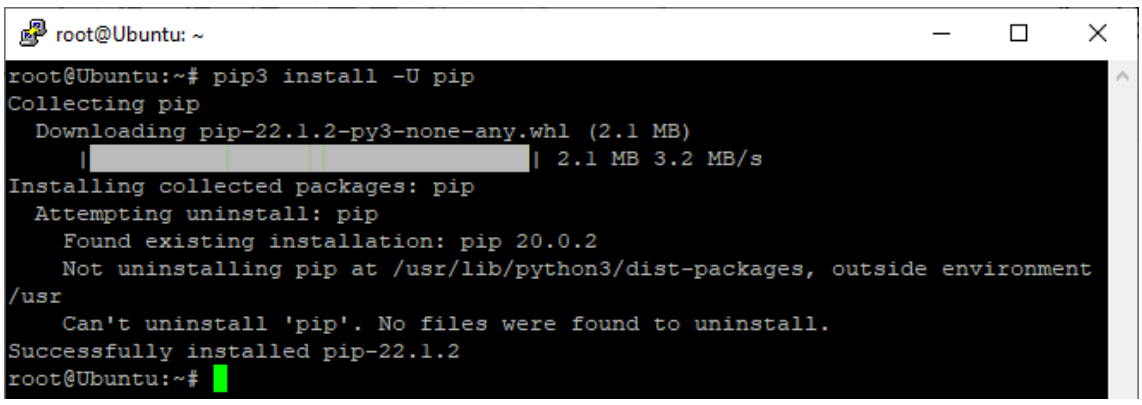


Figura 46. Ejecución e instalación correcta del comando pip3 install -U pip

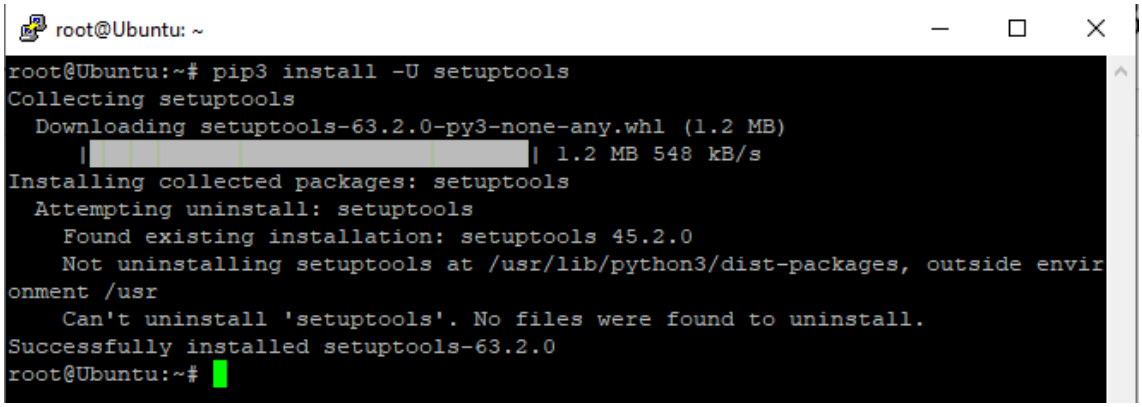


Figura 47. Ejecución e instalación correcta del comando pip3 install -U setuptools

Una vez finalizado el proceso de instalación de todos los comandos necesarios para trabajar con la biblioteca Netmiko, dentro de nuestra máquina de Ubuntu Docker Guest, procedemos a realizar la configuración básica de los dispositivos de red de la topología mostrada anteriormente en la figura 15, el objetivo fundamental de estas configuraciones es el configurar, las direcciones ip, las credenciales de los equipos para poder acceder de manera remota, y la habilitación del protocolo SSH.

8. Configuración del dispositivo R1 (Router R1)

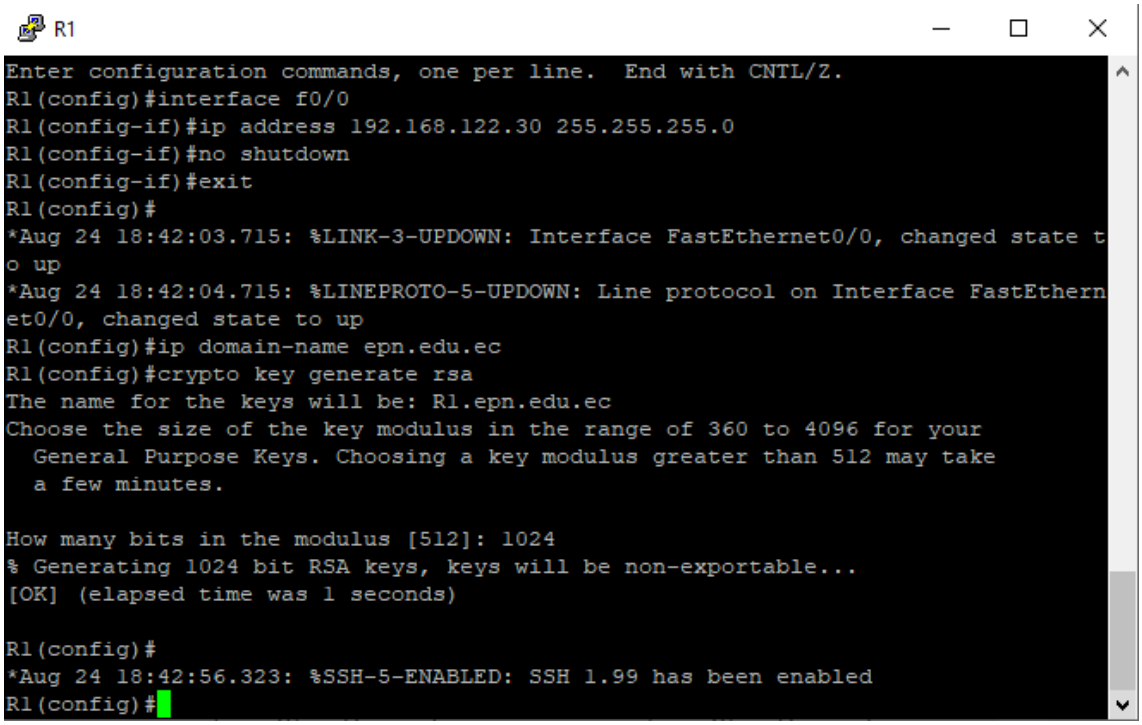


Figura 48. Configuración del dispositivo de red R1

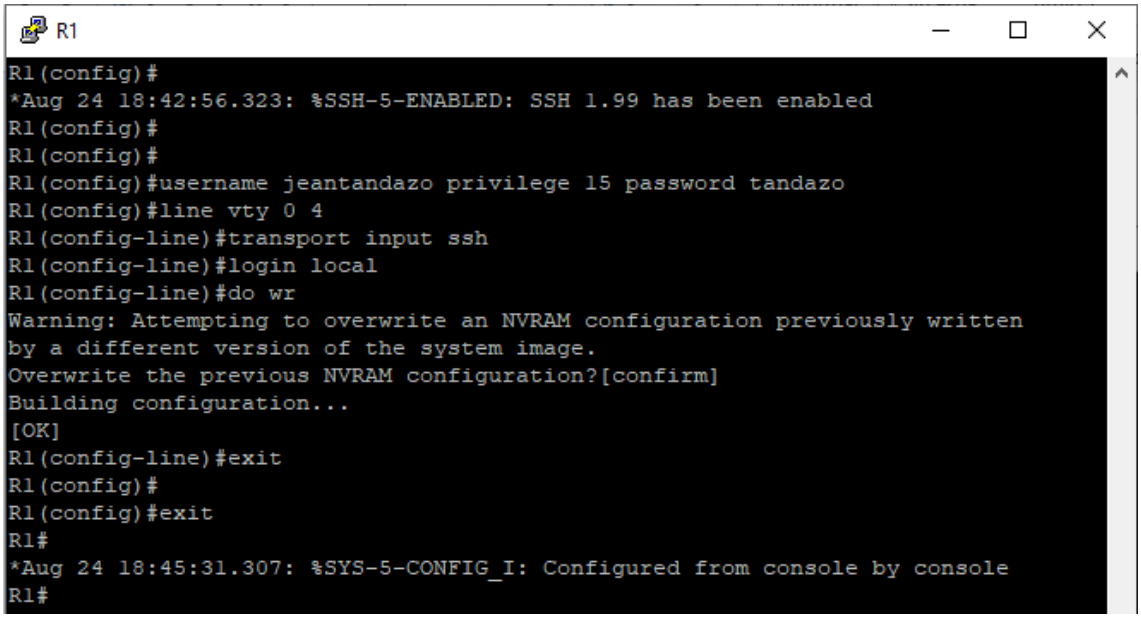


Figura 49. Continuación de la configuración del dispositivo de red R1

9. Configuración del dispositivo IOU1

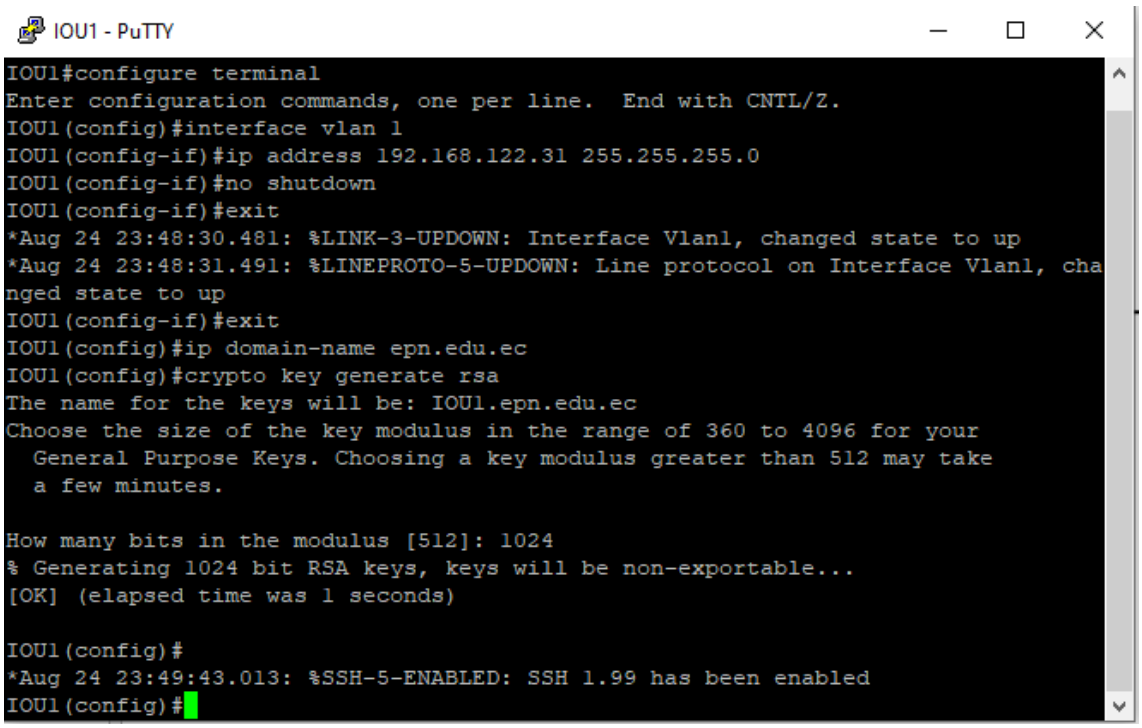


Figura 50. Configuración del dispositivo de red IOU1

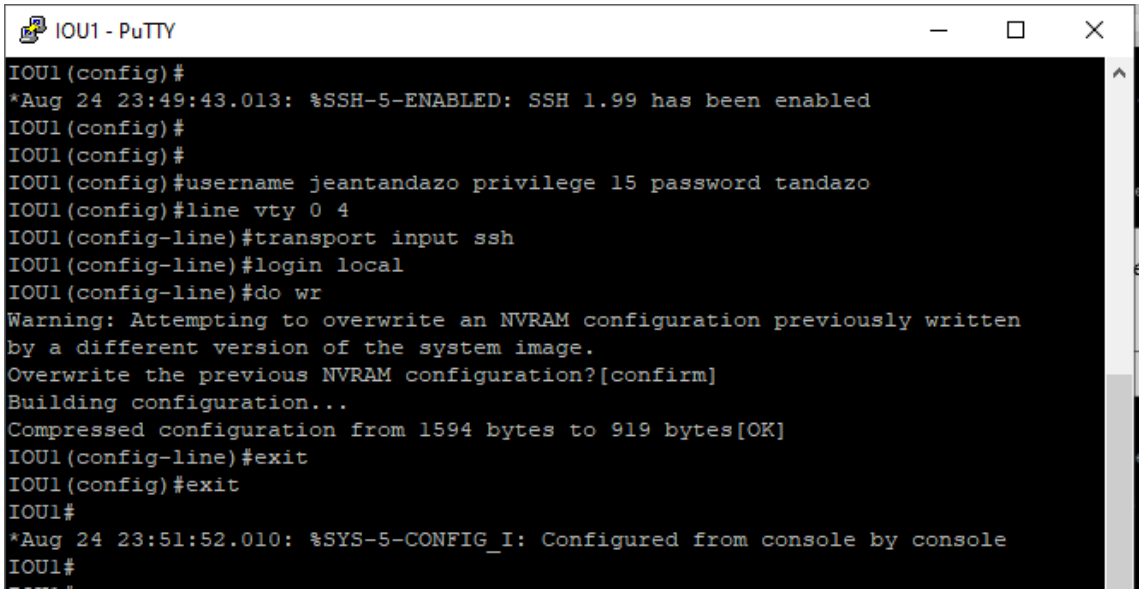


Figura 51. Continuación de la configuración del dispositivo de red IOU1

10. Configuración del dispositivo IOU2

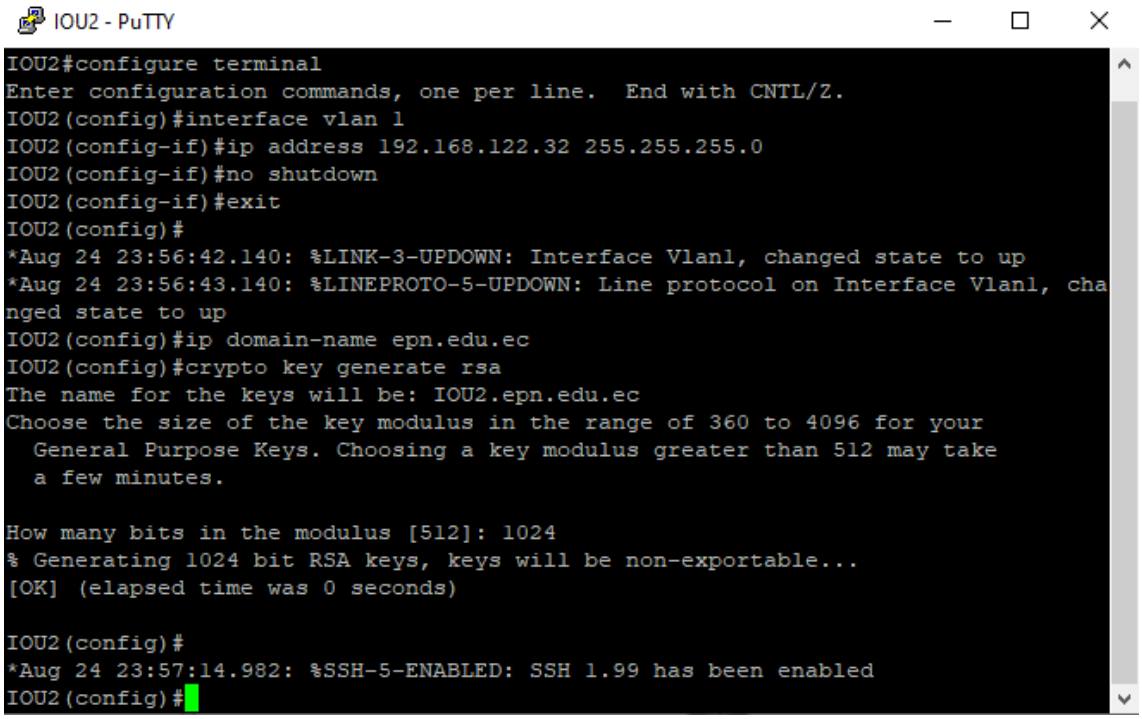


Figura 52. Configuración del dispositivo de red IOU2

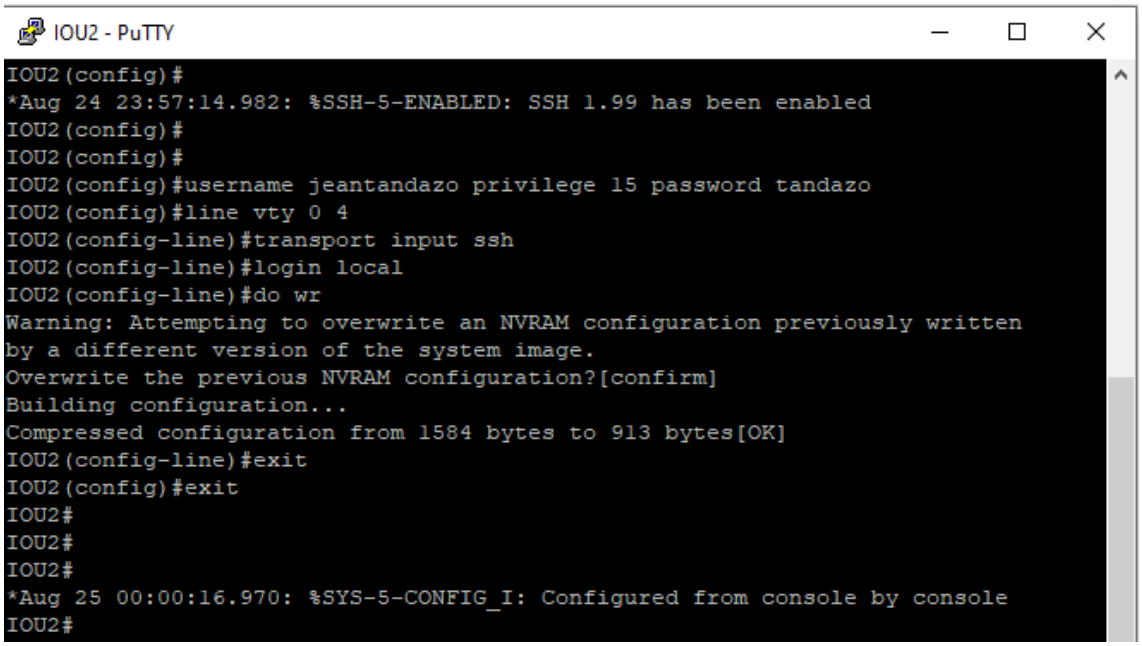


Figura 53. Continuación de la configuración del dispositivo de red IOU2