ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

IMPLEMENTACIÓN DE ANSIBLE PARA LA CONFIGURACIÓN DE EQUIPOS DE *NETWORKING*

IMPLEMENTAR ANSIBLE PARA HABILITAR PROTOCOLOS DE ENRUTAMIENTO DINÁMICO IPV4

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO SUPERIOR EN REDES Y TELECOMUNICACIONES

JIMMY ADRIAN SARANGO VACA

DIRECTOR: ING. GABRIELA KATHERINE CEVALLOS SALAZAR, MSC.

DMQ, agosto 2022

CERTIFICACIONES

Yo, Jimmy Adrian Sarango Vaca 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.

Jimmy Adrian Sarango Vaca

jimmy.sarango@epn.edu.ec

jimmy_asv@yahoo.es

Certifico que el presente trabajo de integración curricular fue desarrollado por Jimmy Adrian Sarango Vaca, bajo mi supervisión.

ING. Gabriela Katherine Cevallos Salazar, MSC.

DIRECTOR

gabriela.cevalloss@epn.edu.ec

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.

JIMMY SARANGO

DEDICATORIA

El presente trabajo está dedicado a mis padres Vicenta y Miguel que gracias a su apoyo incondicional en todo este trayecto de formación han sabido proporcionarme sus consejos y sobre todo han depositado su confianza en mí para lograr mis sueños, gracias a mi madre que ha sabido orientarnos para tomar buenas decisiones, aunque a veces muy estricta pero ha logrado que sea una persona respetuosa y responsable, gracias a mi padre que a la distancia me ha apoyado incondicionalmente y por todo su cariño.

A mis hermanas Nataly y Janela que todos los días con sus ocurrencias han sabido alentarme a seguir esforzándome cada día.

A mi tía Patricia que en estos últimos meses que convivimos me enseñaste que lo más bonito que tenemos es la vida y me quedo muy contento de haber compartido contigo, ahora eres mi ángel en el cielo.

A la persona más importante de mi vida Jessenia, que ha estado en los momentos más difíciles apoyándome hasta donde tu podías, no fue fácil pero tampoco complicado ahora este logro es de los dos.

Jimmy

AGRADECIMIENTO

En primer lugar, quiero agradecer a Dios por darme salud para ver cumplidas mis metas en especial en esta etapa de formación.

A mis padres y mis hermanas por estar conmigo apoyándome en cada decisión que he tomado durante toda mi vida universitaria, alentándome a ser perseverante y seguir adelante.

A mis profesores, amigos y a la Escuela Politécnica Nacional, especialmente a la ESFOT por todas las oportunidades que se me brindaron, no fue sencillo el proceso, pero gracias a ello he podido culminar mi carrera y graduarme como un feliz profesional.

Finalmente, gracias a todas aquellas personas que de una u otra manera me apoyaron durante mi proceso formativo y la vida diaria, esta es una recompensa por tanto esfuerzo que me trajo hasta aquí afrontando muchas adversidades.

Jimmy

ÍNDICE DE CONTENIDOS

CERTIFICACIONES ..... **¡Error! Marcador no definido.**

DECLARACIÓN DE AUTORÍA ..... **¡Error! Marcador no definido.**

DEDICATORIA ..... **¡Error! Marcador no definido.**

AGRADECIMIENTO ..... **¡Error! Marcador no definido.**

ÍNDICE DE CONTENIDO ..... **¡Error! Marcador no definido.**

RESUMEN ..... **¡Error! Marcador no definido.**I

ABSTRACT ..... **¡Error! Marcador no definido.**I

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO 1

1.1 Objetivo general 1

1.2 Objetivos específicos 1

1.3 Alcance 1

1.4 Marco Teórico 1

Características y requerimientos de la herramienta Ansible 1

Software de simulación de redes 3

2 METODOLOGÍA 5

3 RESULTADOS 6

3.1 Instalación de la herramienta de Ansible 6

3.2 Establecimiento de las topologías de red 9

Creación de las topologías para RIP y OSPF 10

Configuración de IP estáticas en interfaces de *routers* y generación de claves para SSH 11

Configuración de IP estática en el Controlador 12

Generación de llaves SSH en el nodo controlador 14

3.3 Implementación de las configuraciones mediante Ansible 17

Configuración del archivo Inventario 17

Creación de *playbooks* para RIP 18

Creación de *playbooks* para OSPF 24

3.4 Pruebas de Funcionamiento y verificación 27
Configuración inicial de equipos para RIP 27
Ejecución del *Playbook* para la topología RIP 30
Verificación de los cambios en la topología RIP 31
Tabla de Enrutamiento IP Protocolo RIP 33
Pruebas de conectividad topología RIP 34
Configuración inicial de equipos para OSPF 36
Ejecución del *Playbook* para la topología OSPF 39
Verificación de los cambios topología OSPF 40
Tabla de Enrutamiento IP protocolo OSPF 43
Pruebas de conectividad topología OSPF 44
4 CONCLUSIONES 47
5 RECOMENDACIONES 48
6 REFERENCIAS BIBLIOGRÁFICAS 49
7 ANEXOS 52
ANEXO I: Certificado de Originalidad i
ANEXO II: Enlaces ii
ANEXO III: Códigos Fuente iii
Playbook RIP *router* 1 (ripv2Router1) iii
Playbook RIP *router* 2 (ripv2Router2) iv
Playbook OSPF *router* 1 (ospfRouter1) v
Playbook OSPF *router* 2 (ospfRouter2) vi

RESUMEN

El presente proyecto de investigación, IMPLEMENTAR ANSIBLE PARA HABILITAR PROTOCOLOS DE ENRUTAMIENTO DINÁMICO IPV4, permite crear *playbooks* para realizar configuraciones automáticas de diferentes protocolos de enrutamiento, a través de comunicación SSH desde el nodo controlador hacia los equipos *host.*

En la primera sección, se hallan los requerimientos mínimos y características de la herramienta de Ansible con la cual se trabajará, también se expone el objetivo general y los objetivos específicos los cuales son fundamentales para desarrollar la investigación.

La segunda sección, expone el tipo de investigación que se realizó para desarrollar el proyecto de titulación, también se presenta el procedimiento para dar cumplimiento a los objetivos específicos.

La tercera sección, presenta de forma detallada los pasos a seguir para realizar la instalación del nodo controlador, además de la implementación y despliegue de los *playbooks* en las topologías de red establecidas para los diferentes protocolos de enrutamiento. En esta sección también se encuentran las pruebas de funcionamiento del proyecto de investigación.

Finalmente, se expone las conclusiones y recomendaciones que surgieron durante el desarrollo del proyecto.

PALABRAS CLAVE: Ansible, Inventario, libro de jugadas, Estado de enlace, Vector distancia.

ABSTRACT

This present research project IMPLEMENT ANSIBLE TO ENABLE IPV4 DYNAMIC ROUTING PROTOCOLS, allows the creation of playbooks to perform automatic configurations of different routing protocols, through SSH communication from the controller node to host computers.

In the first section presents the minimum requirements and characteristics of Ansible tool with which we will work are found, the general objective and specific objectives which are fundamental to develop the research are also exposed.

The second section, exposes the type of research that was carried out to develop the degree project, also presents the procedure to comply with the specific objectives.

The third section presents in detail the steps to follow to perform the installation of the controller node, in addition to the implementation and deployment of the playbooks in the network topologies established for the different routing protocols. In this section you will also find the tests of the research project.

Finally, the conclusions and recommendations that emerged during the development of the project are presented.

KEYWORDS: *Ansible, Inventory, Playbooks, Link State, Vector Distance.*

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

El presente proyecto pretende desplegar una topología de red, donde se habilitarán protocolos de enrutamiento IPv4, automatizando la configuración de los equipos de *networking* mediante el uso de la herramienta de DevOps Ansible. Ansible permite automatizar tareas de administración de los dispositivos de red de manera remota; garantizando eficiencia, rapidez y fiabilidad; con menos esfuerzo y riesgo de errores humanos.

1.1 Objetivo general

Implementar Ansible para la configuración de equipos de *networking.*

1.2 Objetivos específicos

- Instalar la herramienta de Ansible en el nodo controlador.
- Montar los diferentes dispositivos de la red en un simulador de redes.
- Implementar las configuraciones mediante la herramienta Ansible.
- Realizar pruebas de funcionamiento y verificación de los resultados obtenidos.

1.3 Alcance

El presente proyecto conlleva la implementación de una topología de red, donde los dispositivos logren tener conectividad configurándolos de forma automatizada mediante la herramienta de Ansible. Para esto se debe establecer una comunicación SSH entre el nodo controlador, máquina principal donde se instala Ansible, y los diferentes dispositivos de red. Con esto se despliega de manera remota las configuraciones necesarias para la implementación de cada topología de red establecida.

1.4 Marco Teórico

Características y requerimientos de la herramienta Ansible

Las herramientas de automatización para desarrollo y operaciones dentro de las TI tienen la capacidad de realizar tareas de administración, actualización y despliegue de tareas en los dispositivos de una red [1] [2]. Ansible puede realizar las tareas de gestión que un administrador de red realizaría; el uso de módulos en Ansible lleva consigo un beneficio de ahorro de tiempo y eficiencia en las tareas simples, complejas y repetitivas, además que excluye el error humano en las configuraciones [1].

Ansible se instala en un ordenador que es una máquina nodo controlador, en esta se crean los *playbooks*. El nodo controlador al estar conectado a los nodos controlados o *host* despliega los *playbooks* automatizando tareas en los mismos [2].

Ansible para que se ejecute como nodo controlador necesita un sistema basado en Linux/UNIX [3], compatible con *Python* incluidos *Red Hat* y MacOS; también se puede gestionar en equipos *Windows*, aunque no compatible con el nodo controlador [4].

Un sistema operativo óptimo para su instalación es Ubuntu que en su versión de servidor no requiere gran cantidad de recursos, debido que mínimo necesita de 384 (MB) de RAM, 1.5 (GB) de Disco y tarjeta de video con resolución de 640 x 480 [5]; apto y muy conocido en diferentes plataformas virtuales y físicas para la automatización con Ansible [6].

La herramienta de Ansible funcionando sobre Ubuntu puede realizar tres clases de automatización:

- Aprovisionamiento: Puede agregar más dispositivos a la red según lo que la infraestructura necesite [7].
- Gestión de Configuración: Ejecuta diversos cambios de configuración en los dispositivos de la red, manteniendo el rendimiento de los equipos [7].
- Implementación de Aplicaciones: Favorece al desarrollo y operación en la puesta en marcha de la automatización de las aplicaciones desarrolladas en el entorno de Ansible [7].

Los nodos controlados o *host* para ser automatizados no requieren de la instalación de un *software* especial o agente, requiere de conexión SSH y que el equipo pueda ejecutar *Python* 2 o *Python* 3 [4]. Previo a la configuración que será desplegada mediante Ansible, todos los dispositivos que conforman la infraestructura deben estar en red y tener conectividad. Ansible automatiza varios equipos simultáneamente, para lo cual debe conocer a los nodos *host* a través de un inventario o lista de *hosts*; esta lista permite que Ansible ejecute el *playbook* sobre un *host* o grupo de *hosts* [8] [9].

Los *playbooks* son un conjunto de órdenes con las cuales se automatiza a los dispositivos remotos o *host*, este archivo de configuración ejecuta varias tareas repetitivas secuencialmente en varias máquinas definidas por un orden. El *playbook* está realizado en formato YAML el cual es amigable e intuitivo para el administrador [10].

Ansible en su versión 2.9.27 requiere de un mínimo en espacio de almacenamiento, lo cual es de 20 (MB) ya que Ansible es una versión no gráfica; para la instalación se descarga paquetes del repositorio de Ansible mismos que están construidos en la arquitectura AMD64 [11].

En este contexto, Ansible es una herramienta que se puede ejecutar en cualquier ambiente de TI, facilitando enormemente la labor de los administradores de redes, ya que es una herramienta fácil de aprender con lenguaje intuitivo, logrando realizar tareas complejas con menos desgaste y ocupando los recursos necesarios.

Software de simulación de redes

La simulación en el área de las redes y de las telecomunicaciones es muy útil, ya que se puede determinar el funcionamiento real de los equipos y a partir de ello examinar resultados, previo a la instalación de toda la red con el equipamiento físico. Evitando, además, averiar uno o varios equipos ya que el costo por la pérdida de los mismos es elevado. Algunos de los simuladores permiten emular redes, lo que ayuda a conectarlos a una red real y obtener datos reales de tráfico permitiendo un análisis más detallado [12].

GNS3

Graphical Network Simulator GNS3 es un *software* que simula las funciones y características de un dispositivo, emula o imita el *hardware* de un dispositivo real en el entorno virtual, ayuda a probar y verificar las implementaciones en el mundo real [13]. GNS3 usa módulos de *Dynamips*, Quemu y VirtualBox que ayuda a las simulaciones acercarse mucho a la realidad; este *software* consta de dos partes: la Interfaz Gráfica (GUI) todo en uno disponible para Windows, Linux y MAC; y la opción de máquina virtual (VM) que trabaja en conjunto con la interfaz gráfica en la cual se alojan y ejecutan los dispositivos en un proceso de servidor ya sea servidor local, máquina virtual local y remota que pueden ejecutarse incluso en la nube [13] [14] [15].

GNS3 es compatible con dispositivos como Cisco, Cisco ASA, conmutadores Cumulus Linux, diversas versiones de Linux Docker; en versiones de Cisco IOS es compatible con IOS 12.X y actualmente con IOS 15.X, se recomienda usar imágenes de Cisco VIRL [13].

GNS3 tiene grandes ventajas ya que no tiene una limitación en cuanto a dispositivos compatibles, trabaja con hipervisores o sin ellos los mismos que pueden ser de paga o gratuitos, tiene diversas opciones de conmutación, descargas gratuitas de dispositivos para una mejor implementación, admite entornos y *software* de varios proveedores [13].

La gran desventaja es que requiere de recursos tanto en Memoria y CPU del equipo donde se instala, las imágenes para la implementación deben ser facilitadas por el administrador [13]. Sin embargo, GNS3 ofrece una instalación sencilla que se reduce a un instalador el cual puede ejecutarse en cualquier entorno [16] [17].

EVE-NG

Emulated Virtual Environment Next Generation es un *software* de emulación de redes de diferentes proveedores, el cual está realizado para redes empresariales de gran tamaño y complejas. Se basa en módulos estructurales *Dynamips* y Quemu, el entorno de trabajo es muy intuitivo con configuraciones variables ya que trabaja con varios fabricantes como CheckPoint, F5, PaloAlto entro otros [14] [18].

EVE-NG disponible en *Community Edition* tiene aceleración KVM por *hardware*, configuraciones para importar, exportar imágenes y mapas, se puede realizar pruebas pasando de una red real a una virtual, utiliza interfaz de usuario HTML5 para la optimización de memoria, posee compatibilidad de 63 nodos por implementación. EVE-NG en la versión PRO tiene mejoras como compatibilidad con Docker, nube NAT con DHCP de EVE y compatibilidad hasta 1024 nodos por implementación [14] [18].

Las ventajas de EVE-NG es que no sobrecarga los recursos del computador, ya que la interfaz de usuario es a través de una página *Web*; permite la creación de topologías totalmente configurables, admite múltiples conexiones simultáneamente. La gran desventaja es que el usuario debe facilitar la imagen del IOS de los equipos a usar, no ofrece imágenes de equipos precargadas [15].

Los entornos de virtualización tanto GNS3 como EVE-NG están basados en los mismos módulos, y cada uno tiene características que los hacen únicos en cuanto a simulación y emulación. Para el desarrollo del presente proyecto se tomó la decisión de usar GNS3 debido a que la instalación se realiza a través de instalación de paquetes, además los requerimientos son mínimos como un sistema operativo Windows 7 o posterior, RAM de 4 (GB), 1 (GB) de disco duro para almacenamiento y procesador de 2 núcleos lógicos o más [17]. En cambio, EVE-NG la instalación se realiza a través de una máquina virtual o en un *hardware* físico usando una imagen ISO llamada también instalación *bare metal*, además los requisitos mínimos son RAM de 8 (GB), 40 (GB) de disco duro para almacenamiento y procesador Intel I7 de 4 núcleos lógicos [19].

2 METODOLOGÍA

En el presente trabajo de integración curricular se realizó una investigación de tipo exploratoria, la cual parte desde la búsqueda de información sobre el funcionamiento, características, requerimientos y ventajas de la herramienta Ansible para desplegar una automatización de configuración sobre una red.

Con la investigación realizada se despliega Ansible para configurar equipos de *networking*, lo cual permitió configurar interfaces de red y protocolos de enrutamiento logrando la optimización de la configuración, ahorro de tiempo y reducción de errores humanos. La metodología que se sigue a lo largo del proyecto comprende lo siguiente:

Se creó una máquina virtual utilizando el hipervisor VirtualBox, se utilizó el sistema operativo Ubuntu *Server* 18.04, durante la instalación del sistema como característica adicional se coloca OpenSSH. En esta máquina virtual se instaló la herramienta de Ansible para conformar el nodo controlador.

Mediante GNS3 se procedió a colocar los equipos de *networking,* dispositivos finales y el nodo controlador. Se crean dos topologías de red, una para realizar la ejecución del enrutamiento mediante RIP y la otra mediante OSPF. Se estableció el direccionamiento IPv4 en el nodo controlador y los dispositivos de *networking* para establecer la comunicación inicial. En esta sección también se generan los usuarios y contraseñas en los *routers* para tener comunicación mediante SSH. En el nodo controlador se generan las claves y se agregan los *hosts.*

Previo a realizar los *playbooks*, se creó un archivo inventario para establecer las características de cada *host*. Se creó y ejecutó el *playbook* para que se configure RIP en dos *routers,* el direccionamiento IPv4 para la comunicación entre ellos y el direccionamiento IPv4 para su interfaz Ethernet. De igual forma se creó y ejecutó el *playbook* para que se configure OSPF en dos *routers,* el direccionamiento IPv4 para la comunicación entre ellos y el direccionamiento IPv4 para su interfaz Ethernet.

Finalmente se realizó las pruebas de funcionamiento verificando que los equipos de *networking* se hayan configurado exitosamente mediante Ansible. Además, se verificó la conectividad entre diferentes redes implementadas con los protocolos de enrutamiento RIP y OSPF.

3 RESULTADOS

En esta sección se desarrolla la habilitación de protocolos de enrutamiento dinámico RIP y OSPF a través de la herramienta Ansible. En primera instancia se instala Ansible en una máquina virtual con Ubuntu *Server* para establecer la comunicación SSH con los *routers* de la topología implementada en GNS3. Además, se detalla la creación y ejecución de los *playbooks* para cada topología de red creada, esto permitirá una fácil comprensión de la herramienta Ansible y su uso en el despliegue de configuraciones automáticas en equipos de red.

3.1 Instalación de la herramienta de Ansible

Para instalar la herramienta Ansible se requiere un servidor que trabaje como nodo controlador, el mismo que se encuentra virtualizado en el *software* VirtualBox; para ello la máquina virtual trabaja con el sistema operativo Ubuntu *Server* 18.04. La máquina virtual posee características como se observa en la Figura 3.1 con 2 (GB) de RAM, 20 (GB) de almacenamiento interno y 16 (MB) para memoria de video ya que no se requiere gran cantidad de recursos, además durante la instalación se descargó e instaló los paquetes para servidor OpenSSH como se observa en la Figura 3.2.

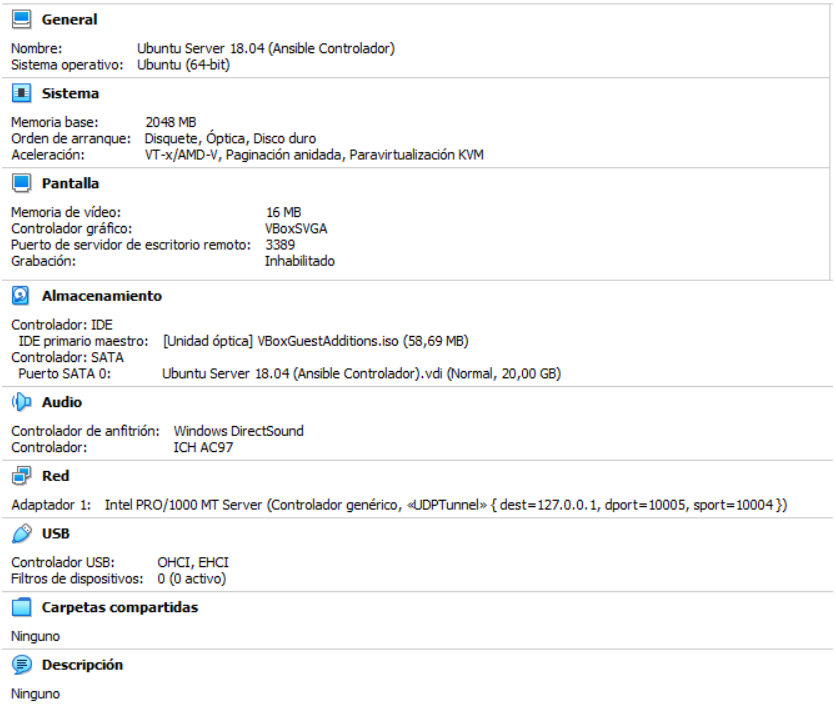

Figura 3.1 Máquina Virtual para nodo controlador

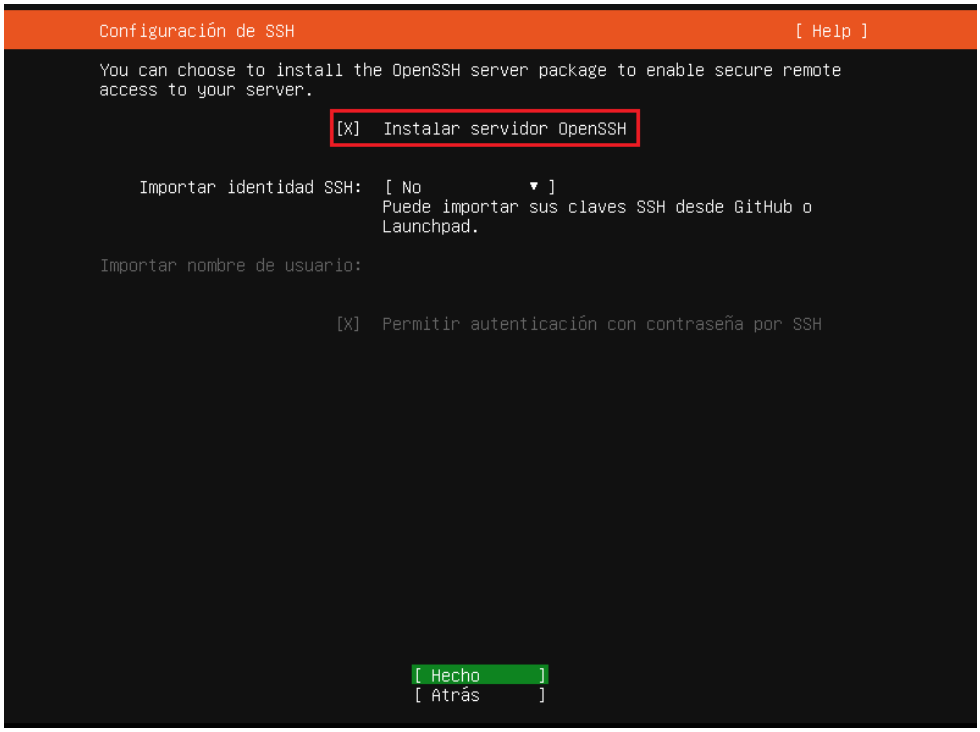

Figura 3.2 Servidor OpenSSH

Iniciada la máquina virtual con el Ubuntu *Server* y antes de instalar Ansible se debe actualizar los paquetes del sistema con los comandos ***sudo apt-get upgrade*** y ***sudo apt-get update***, para evitar inconvenientes con la instalación de diferentes paquetes.

Luego se ejecuta el comando ***sudo apt install software-properties-common*** que facilita *scripts* para agregar y eliminar repositorios, ver Figura 3.3.

```
controlador@srv-controlador:~$ sudo apt install software-properties-common
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
software-properties-common ya está en su versión más reciente (0.96.24.32.18).
fijado software-properties-common como instalado manualmente.
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
controlador@srv-controlador:~$
```

Figura 3.3 Comando para manejo de repositorios

Seguidamente se agregó el repositorio PPA de Ansible a través del comando ***sudo apt-add-repository ppa:ansible/ansible***, ver Figura 3.4.

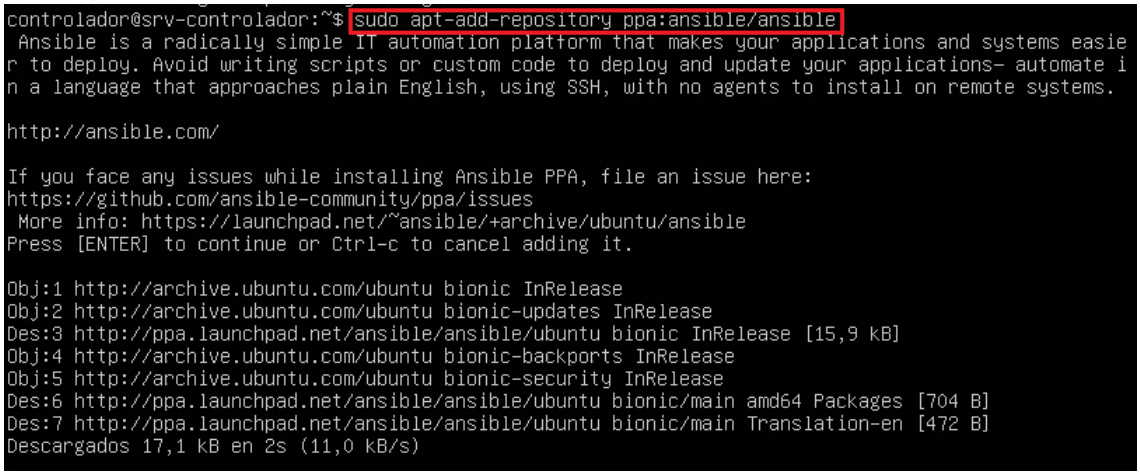

Figura 3.4 Repositorio de Ansible

Después se instaló Ansible a través del comando ***sudo apt install ansible***, para que la herramienta Ansible esté lista para administrar los dispositivos *host*, esto se observa en la Figura 3.5.

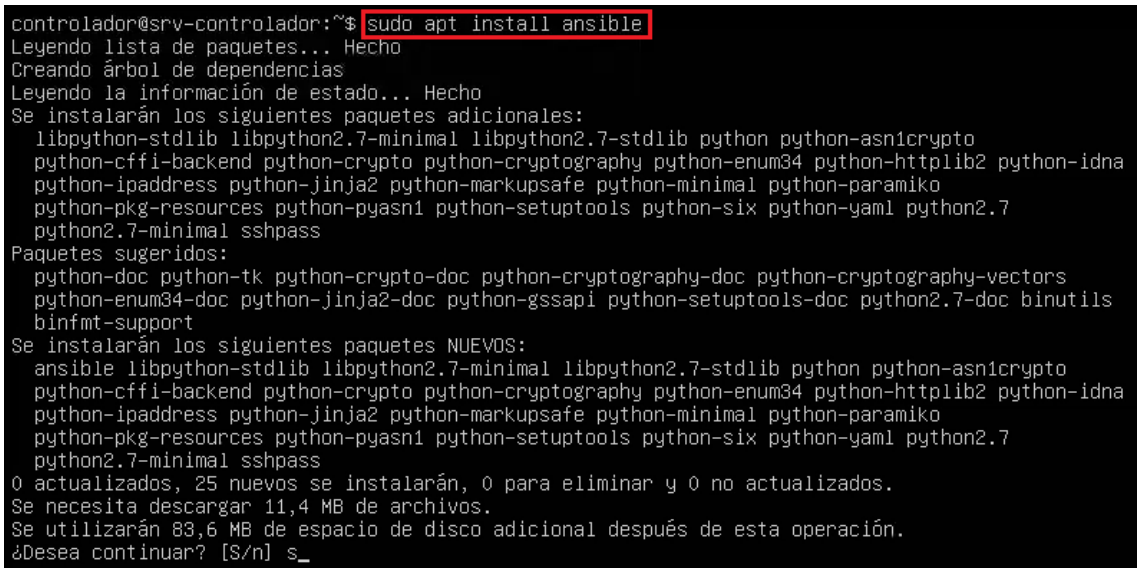

Figura 3.5 Instalación de Ansible

Finalmente se verificó la versión de Ansible instalada, a través de la ejecución del comando ***ansible --versión***, así mismo se puede ver en la Figura 3.6 que se instala Python que sirve de interprete entre el nodo controlador y *host*.

Figura 3.6 Versión de Ansible

De acuerdo con los equipos que se usan en la topología, en Ansible se puede descargar la colección de dispositivos de red, en este caso se usó dispositivos de la marca Cisco por lo tanto se instaló la colección ***cisco.ios*** usando el comando ***ansible-galaxy collection install cisco.ios,*** ver Figura 3.7.

Figura 3.7 Colección de dispositivos Cisco

3.2 Establecimiento de las topologías de red

A la máquina virtual controlador de Ansible se le cambió la preferencia de red, de NAT a Red interna, esto en Virtual Box. Al conectar el nodo controlador con GNS3 el tipo de adaptador de red se lo colocó como controlador genérico UDP Túnel en el modelo Intel PRO / 1000 MT *Server* (82545EM), como se observa en la Figura 3.8. Esto se ejecutó con el fin de realizar la comunicación con los *hosts* usando conexión remota a través de SSH.

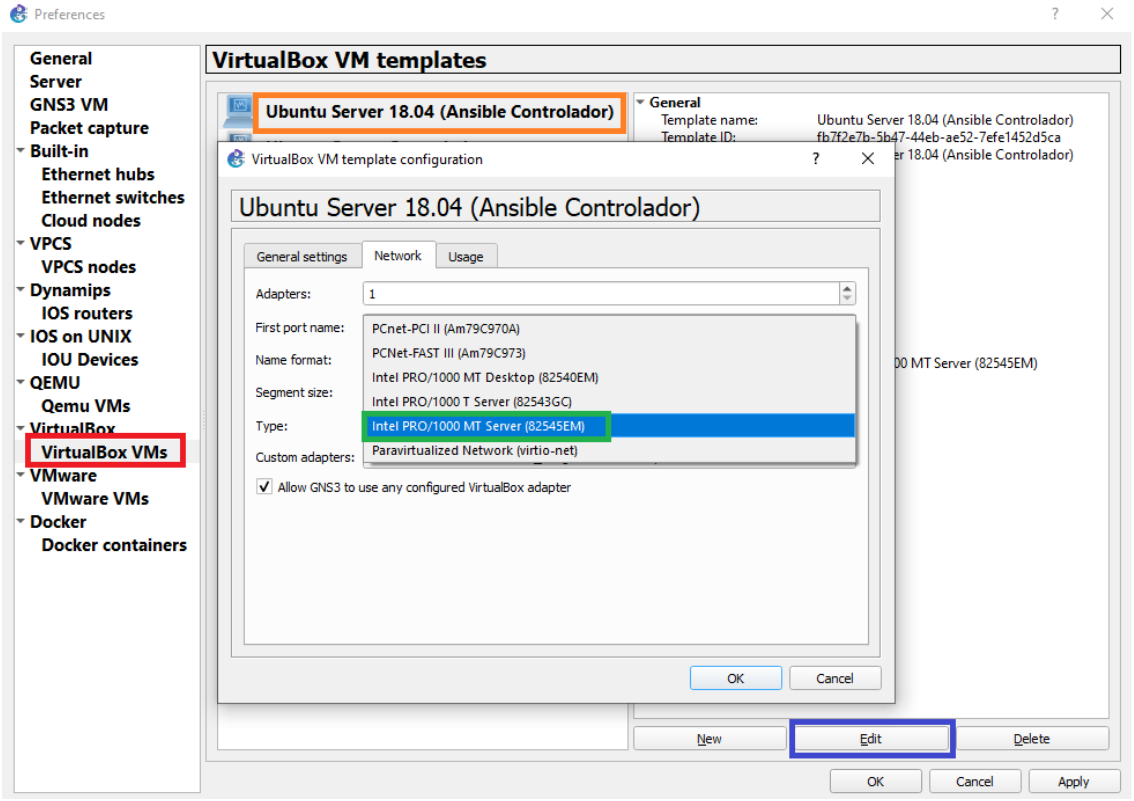

Figura 3.8 Adaptador de red virtual

Creación de las topologías para RIP y OSPF

Con la ayuda de GNS3 se creó las topologías para ejecutar el proyecto de titulación, que contendrán el protocolo de enrutamiento vector distancia RIP como el protocolo de enrutamiento *link state* OSPF. En primer lugar, se agregó la imagen del *router* Cisco 3725 124-25 T14 a GNS3, luego se conectó dos *routers* a través de la interfaz serial; a cada *router* se lo enlazó a un computador, que simula a la red LAN. Se estableció la dirección IP estática a cada *router* en una de sus interfaces FastEthernet, para la comunicación con el nodo controlador, esto se observa en la Figura 3.9 y la Figura 3.10.

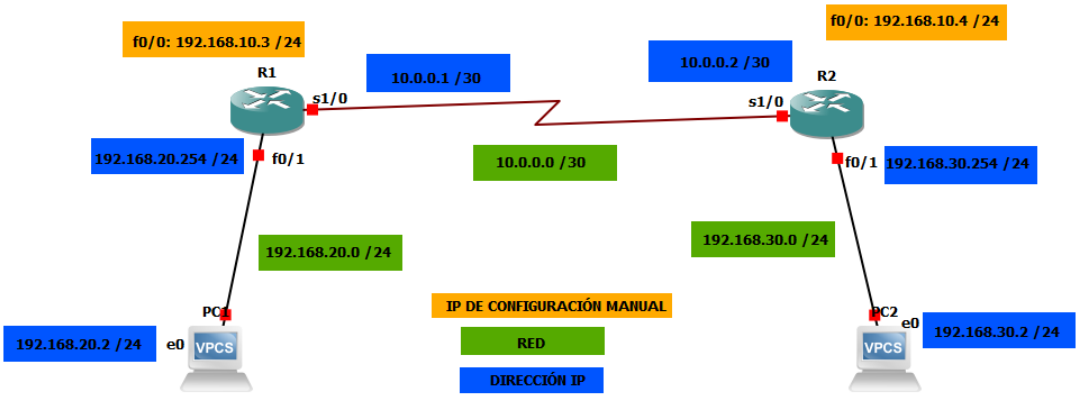

Figura 3.9 Topología RIP

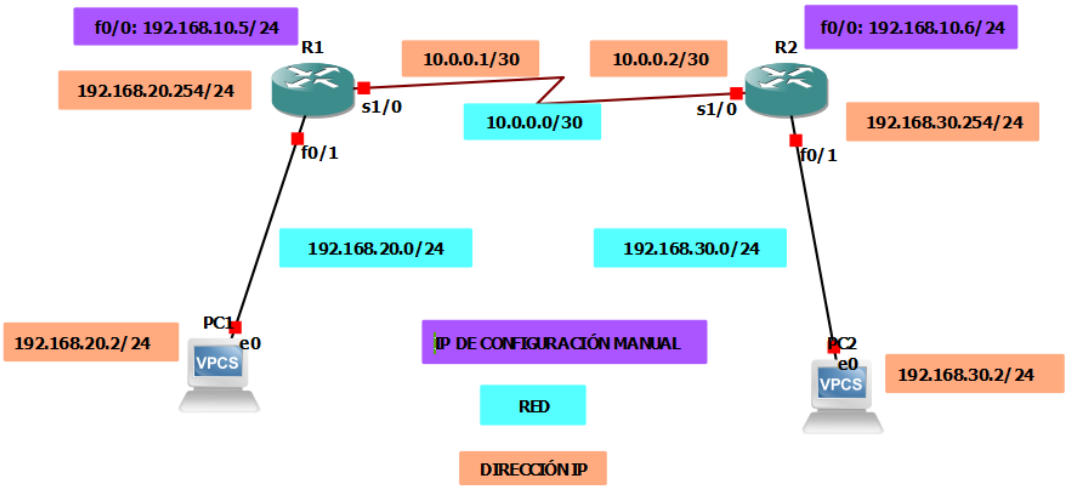

Figura 3.10 Topología OSPF

Configuración de IP estáticas en interfaces de *routers* y generación de claves para SSH

Una vez creadas las topologías se ingresó a cada uno de los *routers* y se establece una dirección IP estática en la interfaz FastEthernet0/0 como se observa en la Figura 3.9 y Figura 3.10, además de establecer un *hostname*, nombre de dominio, creación de un usuario y contraseña a cada *router*; esto con el fin de tener un enlace para el controlador Ansible y establecer la conexión a través de SSH, ver Figura 3.11.

Luego se generan las claves criptográficas RSA para la emisión de certificados en una conexión, se habilita la versión 2 de SSH ya que es más segura y eficiente, además se habilita el transporte de SSH para la terminal de tipo virtual o VTY. La dirección estática de uno de los *routers* de las topologías y la configuración de la generación de claves se puede ver en la Figura 3.11.

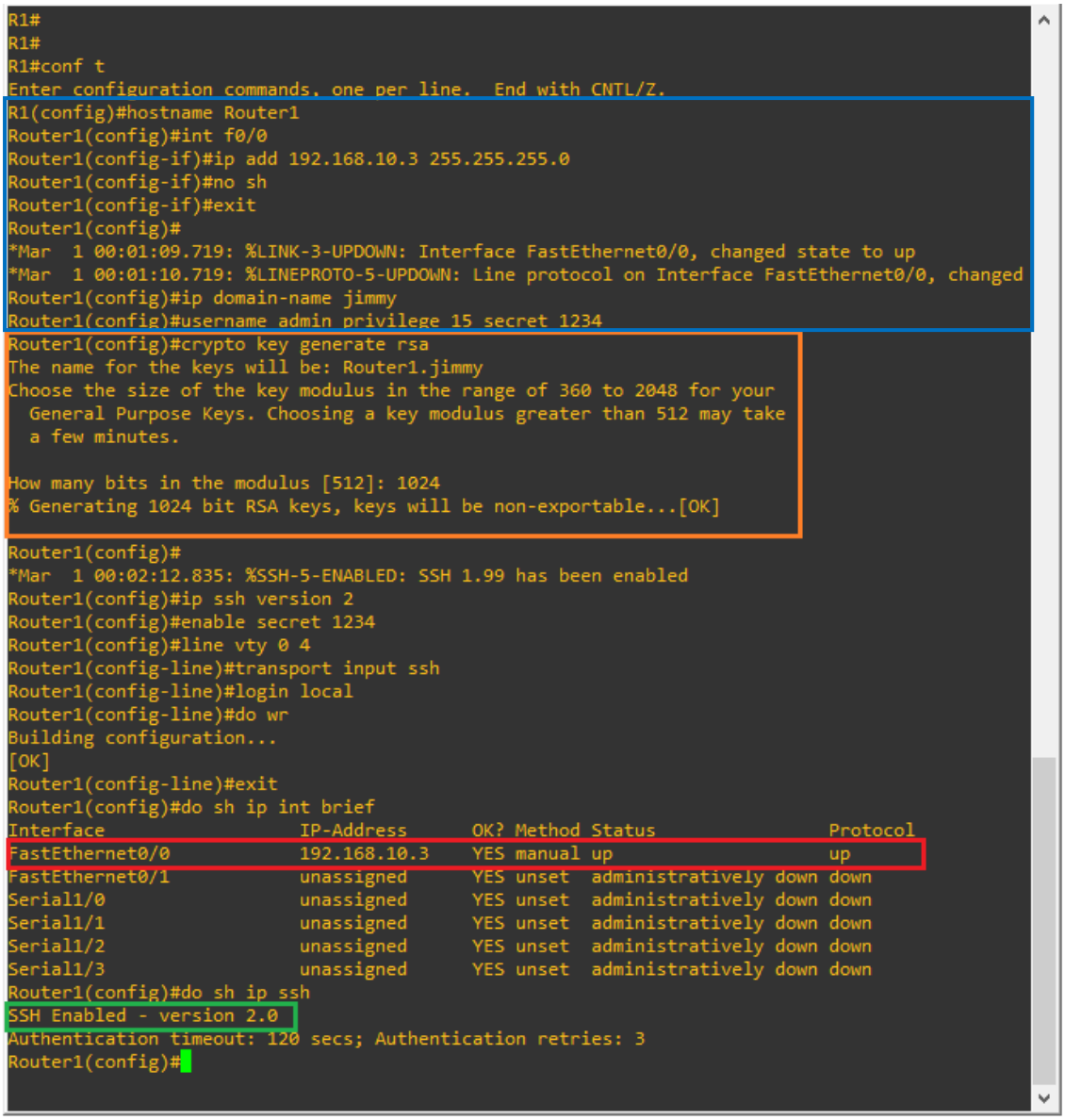

Figura 3.11 Configuraciones del *router* 1

Configuración de IP estática en el Controlador

El nodo controlador agregado a GNS3 y configurado el adaptador de red, se lo añade en la simulación como se observa en la Figura 3.12, se inició los dispositivos y en consecuencia de ello el nodo controlador también se inicia.

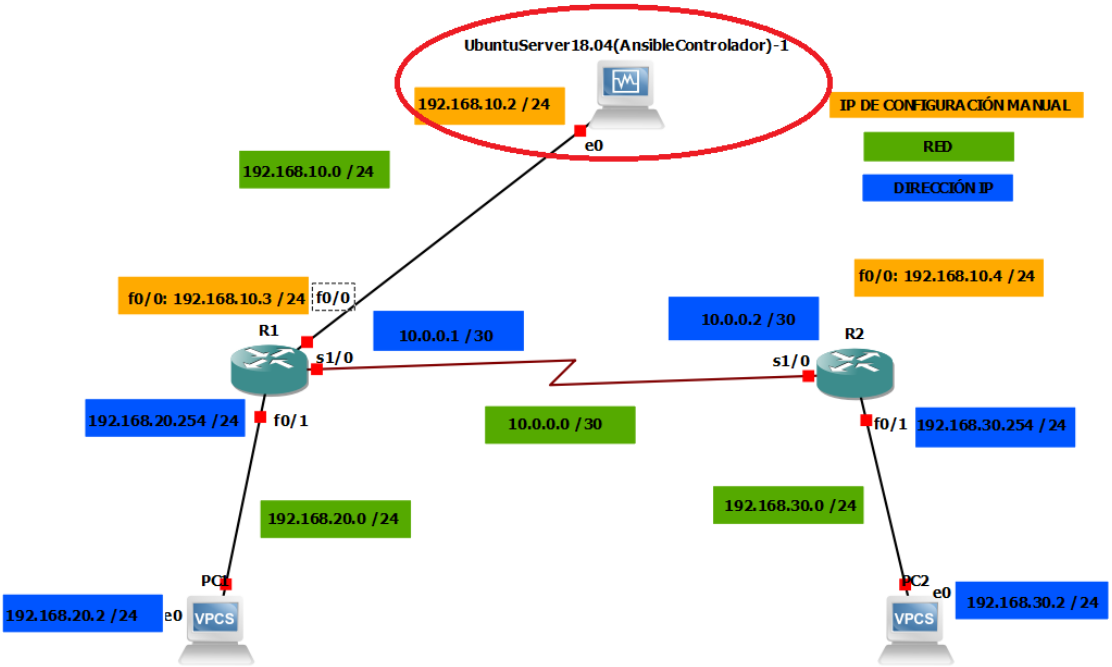

Figura 3.12 Topología RIP con controlador Ansible

Iniciado el nodo controlador, se ejecuta ***ip link show*** para mostrar los atributos de las interfaces de red, ver Figura 3.13.

```
controlador@srv-controlador:~$ ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen
1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp0s17: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 100
0
    link/ether 08:00:27:9e:18:2f brd ff:ff:ff:ff:ff:ff
controlador@srv-controlador:~$
```

Figura 3.13 Atributos de Interfaz de red en el nodo controlador

Luego se realizó el cambio de directorio a netplan para fijar una IP estática en el archivo que contiene el mismo, para ello se ejecutó el comando ***cd /etc/netplan***, con un ***ls*** muestra el archivo ***00-installer-config.yaml*** como se observa en la Figura 3.14.

```
controlador@srv-controlador:~$ cd /etc/netplan
controlador@srv-controlador:/etc/netplan$ ls
00-installer-config.yaml
controlador@srv-controlador:/etc/netplan$
```

Figura 3.14 Archivo de configuración Netplan

En el archivo de netplan se escribe la dirección IP estática 192.168.10.2 /24, Gateway 192.168.10.1, DNS de servidores 8.8.8.8, 8.8.4.4, y se desactivó el DHCP de la interfaz de red, ver la Figura 3.15.

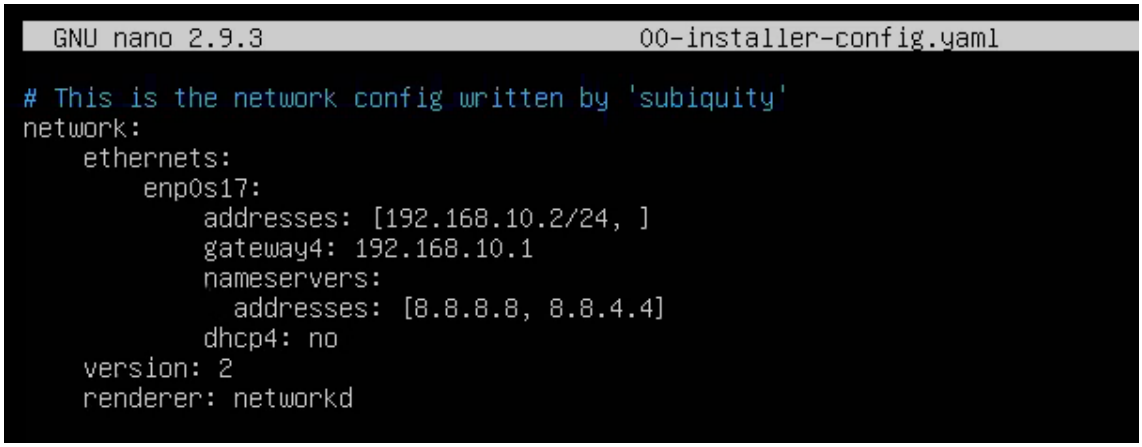

Figura 3.15 Archivo Netplan para configuración de IP estática

Para aplicar los cambios a la interfaz de red se ejecuta el comando ***sudo netplan apply*** y a través del comando ***ifconfig*** se verifican los cambios realizados a la interfaz de red, ver Figura 3.16.

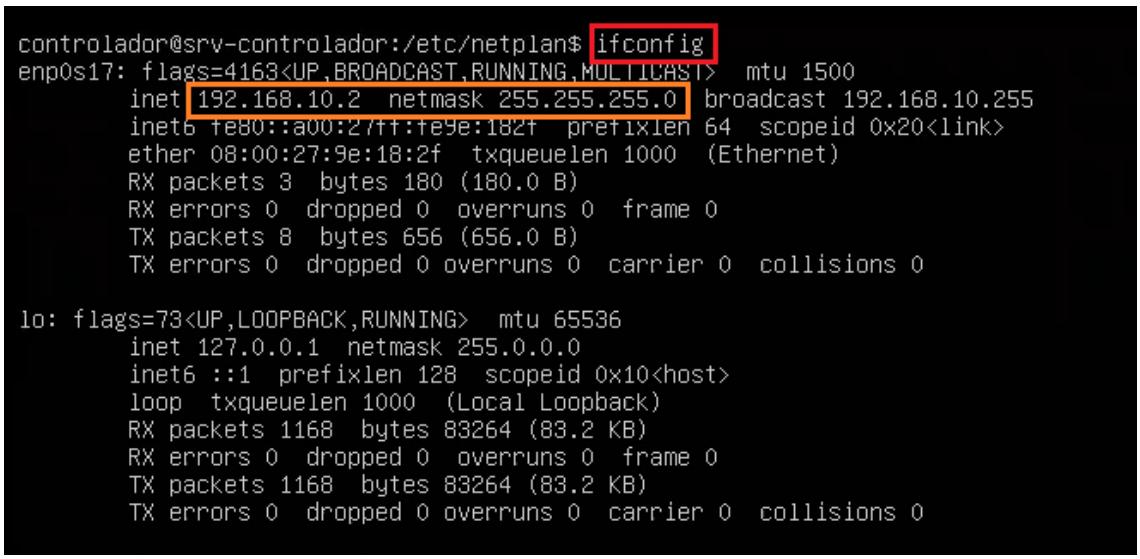

Figura 3.16 Interfaz de red con IP estática

Generación de llaves SSH en el nodo controlador

Después de haber establecido la IP estática en el nodo controlador, se realizó una configuración en el archivo ***ssh_config*** ubicado en la ruta ***/etc/ssh.*** A este archivo se le agregó la línea ***KexAlgorithms +diffie-hellman-group1-sha1*** la misma que permite el intercambio de claves en SSH, además se quita el comentario de la línea ***Ciphers aes128-ctr,aes192-ctr,aes256-ctr,aes128-cbc,3des-cbc*** la cual permite el método de encriptación de la conexión SSH, se puede ver los cambios en la Figura 3.17.

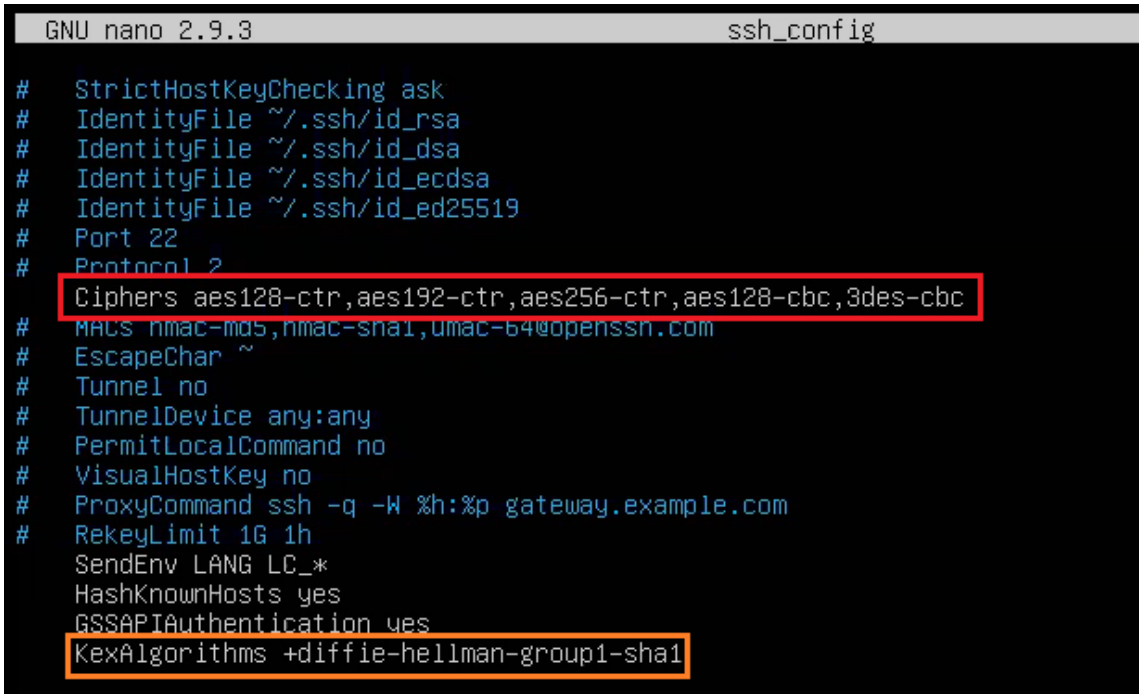

Figura 3.17 Configuración de ssh_config

Una vez hecho los cambios para el intercambio de llaves, se generó las llaves SSH para establecer la conexión entre el *host* y nodo controlador; la creación de las llaves se logra ejecutando el comando ***ssh-keygen***, el cual crea un archivo oculto para guardar las llaves, el mismo que se deja sin frase de seguridad para facilitar la conexión. Esto se visualiza en la Figura 3.18.

Figura 3.18 Generación de llaves SSH en el nodo controlador

Seguidamente se copió las llaves de los *hosts* al controlador con el comando ***ssh-copy-id username@ip-address*** como se observa en la Figura 3.19, se visualiza el reemplazo del usuario y la dirección IP de cada *host.*

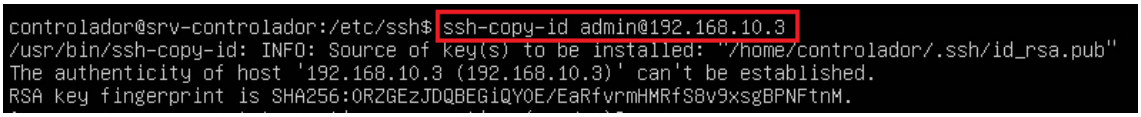

Figura 3.19 Copia de llaves del *host*

Una vez copiadas las llaves del *host* en el nodo controlador, se mantiene la conexión para verificar si ya están alojadas las llaves en el nodo controlador; si no lo están se pide que se ingrese la contraseña del nuevo *host* para almacenarla como se muestra en la Figura 3.20.

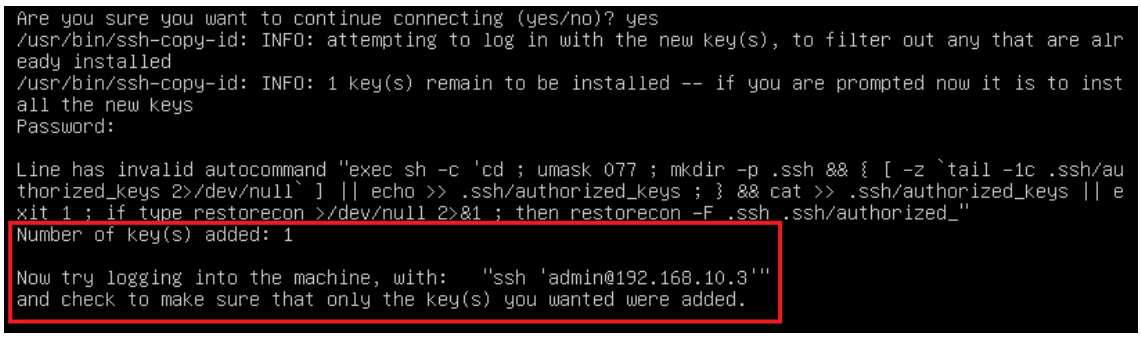

Figura 3.20 Almacenado de llaves SSH en el nodo controlador

La copia de las llaves es similar para cada uno de los *hosts* que se conectan al nodo controlador, como se muestra en la Figura 3.21, lo que cambia es el usuario y la dirección IP del *host.*

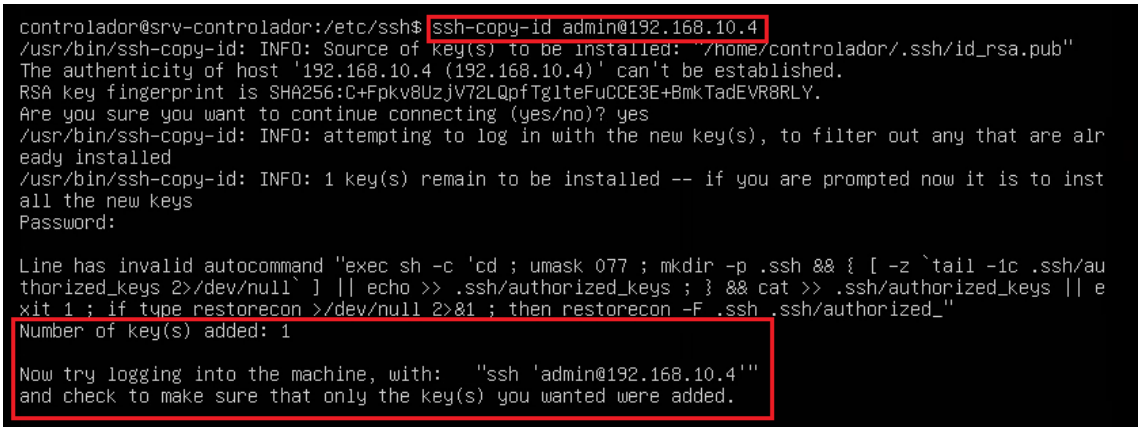

Figura 3.21 Copia de llaves del segundo *host*

3.3 Implementación de las configuraciones mediante Ansible

Configuración del archivo Inventario

Con las llaves SSH en el controlador, se crea el archivo inventario en la ruta **/etc/ansible** a través del comando ***sudo nano host-Ansible,*** como se observa en la Figura 3.22. En ***host-Ansible*** se agrega el *hostname* y la dirección IP de cada uno de los *hosts* de las topologías, los usuarios, las contraseñas, interfaz de usuario de conexión, sistema operativo de *router* y el intérprete para la comunicación entre controlador y *routers*, lo cual se hace para cada uno de los *hosts,* como se puede ver en la Figura 3.23

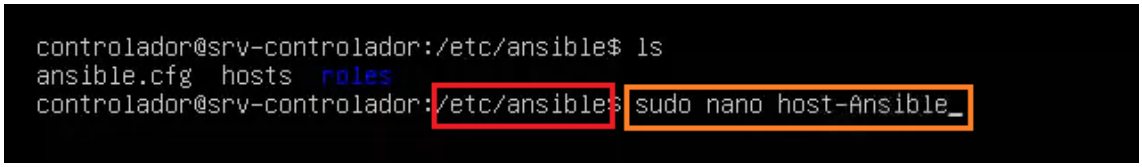

Figura 3.22 Comando para crear inventario y ruta específica de creación

Figura 3.23 Archivo Inventario de Ansible

Una vez creado el inventario, se configura el archivo de **ansible.cfg** ubicado en la ruta ***/etc/*ansible**, y a través del comando ***sudo nano ansible.cfg,*** se accede al archivo de configuración como se muestra en la Figura 3.24, se elimina el carácter de comentario y se cambia la ruta del inventario, remplazando ***hosts*** por el nombre del *inventario* ***host-Ansible***, como se muestra en la Figura 3.25.

Figura 3.24 Comando para editar ancible.cfg

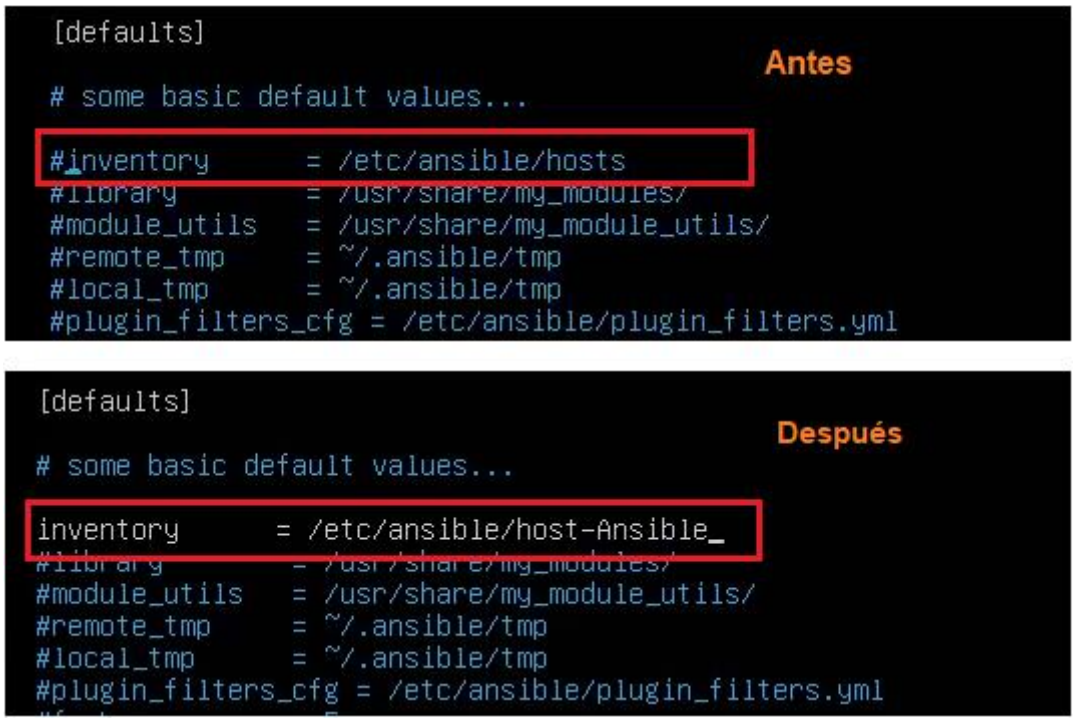

Figura 3.25 Antes y después de la configuración del Inventario

Creación de *playbooks* para RIP

Una vez elaborado el inventario, se realizaron los *playbooks* para el *Router1* como para el *Router2* con el fin de establecer el protocolo de enrutamiento RIP. En primer lugar, se debe ubicar en el directorio que se desee guardar el *playbook,* en este caso se guarda en la ruta ***/etc/ansible,*** y usando el comando ***sudo nano ripv2Router1.yml*** (el nombre del *playbook* para *Router1* es ripv2Router1) se crea el archivo yml y se abre para realizar el *playbook* con las instrucciones requeridas, como se muestra en la Figura 3.26.

Figura 3.26 Comando para crear el *playbook* para *Router1*

El *playbook* inicia con tres guiones (-), indicador de inicio de un *playbook*, es importante el espaciado usando la tecla espaciadora y no el tabulador, ya que la jerarquía de cada instrucción depende del número de espacios.

En primer lugar, se tiene la línea ***- hosts: "nombre del host"*** que indica a qué equipo de la lista de inventario se va a configurar, bien puede ser un solo equipo o un grupo de equipos, ver Figura 3.23. En la siguiente línea ***gater_facts: false,*** el *gater facts* recopila información de los *hosts* remotos y se puede configurar de dos formas: *true* que es la configuración determinada y *false* para negar la recopilación. Luego está la línea ***remote_user: "usuario remoto"*** indica a qué usuario remoto se va a conectar, por defecto se conecta a todos los usuarios, por lo que se debe especificar a qué usuario se conecta. Seguidamente, está la línea ***tasks*** indica las acciones o tareas que se realizan sobre el equipo mencionado en la primera línea *hosts,* estas primeras líneas del *playbook* se pueden observar en la Figura 3.27.

Luego se tienen las tareas que se realizarán de forma secuencial al ejecutar el *playbook*, la primera tarea está colocada a cuatro espacios del nivel de *hosts* para establecer la jerarquía, y es la configuración de interfaces, ***- name: Configuración de Interfaces,*** *name* permite identificar la tarea que se realiza, nombre con el cual se identifica al proceso durante la ejecución del *playbook*, esta identificación funciona bajo la jerarquía de *tasks.*

La siguiente línea ***cisco.ios.ios_l3_interfaces:*** es el módulo de recursos de Cisco para la administración de interfaces en la capa 3, la línea ***config;*** ubicada a ocho espacios del nivel de *hosts* y cuatro del nivel de *name,* siendo de menor jerarquía, es el diccionario de opciones de interfaz de capa 3.

La línea ***– name: FastEthernet0/1,*** inicia a nivel jerárquico de *config* con guion, espacio y seguido de *name* que a diferencia de *tasks* se usa para colocar el nombre completo de la interfaz en este caso FastEthernet e identifica a qué interfaz se le realiza cambios con las líneas posteriores.

La línea ***ipv4***: es para establecer el tipo de direccionamiento que se le asigna a la interfaz que se colocó en *name*, la línea ***– address: "ipv4",*** ubicada en la jerarquía de la línea *ipv4* se coloca la dirección IP y máscara que se asigna a la interfaz, en este caso 192.168.10.2/24. Las siguientes tres líneas asignan dirección IP y máscara a una interfaz Serial, en la cual la asignación es similar que la realizada en la interfaz FastEthernet. La primera tarea culmina con la línea ***state: merged,*** o estado en el que se debe dejar la configuración, el estado puede ser *merged, replaced, overridden,*

deleted, rendered, gathered, parsed. El estado *merged* es predeterminado en el módulo de interfaces de capa 3. La primera tarea se puede observar en la Figura 3.27, cuadro color verde.

La segunda tarea en el *playbook,* cuadro color naranja de la Figura 3.27, es la Habilitación de Interfaces en donde ***- name: Configuración de Interfaces*** es el nombre de la tarea. La siguiente línea ***cisco.ios.ios_interfaces*** es el módulo para administrar los atributos de las interfaces en Cisco, la línea ***config*** posee el diccionario de opciones de la interfaz, la línea ***– name: FastEthernet0/1*** es para nombrar la interfaz, la línea ***enabled*** es el estado de la interfaz que puede ser *true* para interfaz administrativa o *false* para deshabilitar la interfaz administrativa, la línea ***description*** se usa para agregar una descripción a la interfaz configurada y poder ofrecer mayor información al personal de TI. Las siguientes tres líneas habilitan y agregan descripción a la interfaz Serial siguiendo la misma estructura para la interfaz FastEthernet y la segunda tarea culmina con la línea ***state: merged*** que es el estado de la interfaz; *merged* es el valor predeterminado del módulo interfaces.

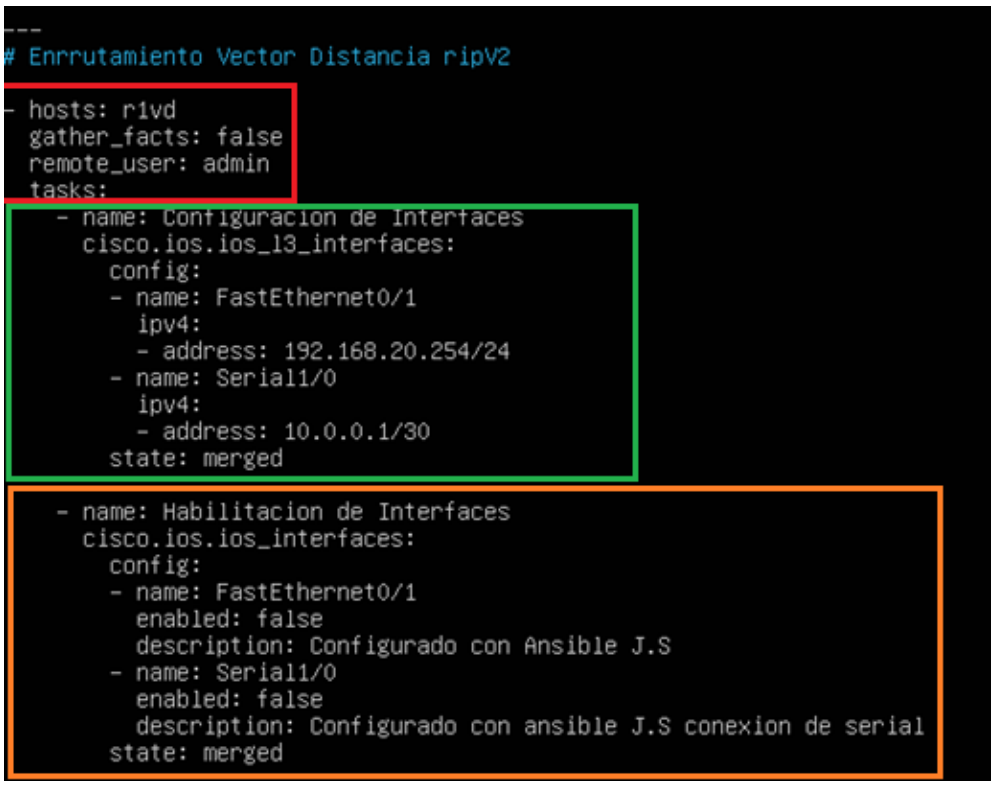

Figura 3.27 *Playbook* RIP *router* 1 primera parte

La tercera tarea, cuadro color azul de la Figura 3.28, es el *Clock rate* para el sincronismo de la conexión serial, la primera línea de esta tarea es ***- name: Clock rate*** nombre de la misma. La línea ***cisco.ios.ios_config*** es el módulo para administrar las secciones de

configuración con los comandos propios de IOS, la línea ***lines*** abre el conjunto de comandos que deben estar ordenados para la configuración, los comandos deben ser exactamente los mismos con los que se realiza la configuración en el dispositivo, la línea ***– interface Serial 1/0*** se ubica en la interfaz Serial para realizar cambios, la línea ***-clock rate 64000*** establece la tasa de transmisión de datos en bits por segundo entre los extremos de la interfaz Serial.

La línea ***– do wr*** guarda las configuraciones realizadas en la interfaz serial y la línea final de esta tarea es ***save_when: always*** que pregunta cómo guarda los cambios realizados en el dispositivo, se tiene cuatro opciones: *always, never, modified, changed; never* es la opción por defecto y no guardará las configuraciones de ejecución o *running-config* en las configuraciones de inicio o *startup-config*; con la opción *always* las configuraciones de ejecución siempre se copiarán en las configuraciones de inicio del dispositivo, y es la que se utilizó.

La cuarta tarea Borrado de Protocolo, ver cuadro amarillo en la Figura 3.28, es para realizar un borrado del protocolo RIP por seguridad, esto en el caso que el dispositivo a configurar ya haya sido configurado con anterioridad. La tarea inicia con ***- name: Borrado de Protocolo*** es el nombre de la tarea que se mostrará en el resumen de ejecución, ***cisco.ios.ios_config*** es el módulo para administrar las secciones de configuración con los comandos propios de IOS, la línea ***lines*** abre el conjunto de comandos que deben estar ordenados para la configuración, la línea ***– no router rip*** deshabilita RIP y elimina todas las declaraciones del protocolo RIP, la línea ***- do wr*** guarda las configuraciones realizadas en el dispositivo ***save_when: always*** guarda las configuraciones de ejecución o *running-config* en las configuraciones de inicio o *startup-config*.

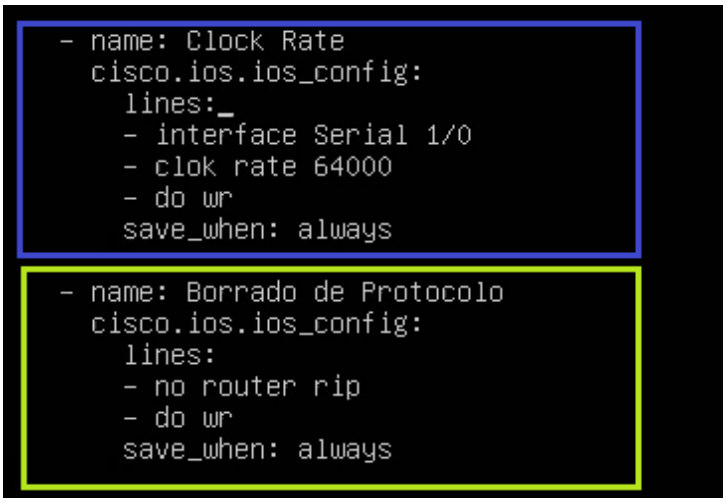

Figura 3.28 *Playbook* RIP *router* 1 segunda parte

La tarea final Enrutamiento ripv2 es para configurar el protocolo de enrutamiento Vector distancia RIP versión 2, la primera línea de esta tarea es ***name: Enrutamiento ripv2*** es el nombre de la tarea que se mostrará en el resumen de ejecución, ***cisco.ios.ios_config*** es el módulo para administrar las secciones de configuración con los comandos propios de IOS, la línea ***lines*** abre el conjunto de comandos que deben estar ordenados para la configuración.

La línea ***– router rip*** habilita el protocolo de enrutamiento RIP, la línea ***- versión 2*** define la versión del protocolo RIP, la línea ***- no auto-summary*** se usa para anunciar subredes, caso contrario entre las clases de redes se anunciará la red principal resumida, la línea ***- network 10.0.0.0*** agrega las dirección de red de la serial con la cual se va a intercambiar información del enrutamiento, con ***- network 192.168.20.0*** se agrega la dirección de red de la LAN para que sea notificada al resto de *routers.*

Con la línea ***- do wr*** se guarda las configuraciones realizadas en el dispositivo, ***save_when: always*** guarda las configuraciones de ejecución o *running-config* en las configuraciones de inicio o *startup-config*, la tarea final Enrutamiento ripv2 se observa en la Figura 3.29.

```
- name: Enrutamiento ripV2
  cisco.ios.ios_config:
    lines:
    - router rip
    - version 2
    - no auto-summary
    - network 10.0.0.0
    - network 192.168.20.0
    - do wr
    save_when: always
```

Figura 3.29 *Playbook* RIP *router* 1 tercera parte

Una vez creado el *playbook* del *Router 1* se procede a crear el *playbook* del *Router* 2 denominado ***ripv2Router2.yml***, para lo cual se hace una copia usando el comando ***sudo cp ripv2Router1.yml /etc/ansible/ripv2Router2.yml,*** como se muestra en la Figura 3.30.

```
controlador@srv-controlador:/etc/ansible$ ls
ansible.cfg  host-Ansible  hosts  ripv2Router1.yml  roles
controlador@srv-controlador:/etc/ansible$ sudo cp ripv2Router1.yml /etc/ansible/ripv2Router2.yml
controlador@srv-controlador:/etc/ansible$ ls
ansible.cfg  host-Ansible  hosts  ripv2Router1.yml  ripv2Router2.yml  roles
controlador@srv-controlador:/etc/ansible$ 7
```

Figura 3.30 Comando para copiar *playbook*

En el *playbook* del *Router 2* se debe hacer ciertos cambios, el primer cambio y el más importante es el *host* a r2vd. En la primera tarea Configuración de Interfaces se cambia las direcciones IP de la FastEthernet a 192.168.30.254/24 y de la Serial a 10.0.0.2/30, en la segunda tarea Habilitación de Interfaces no se realiza cambios ya que las interfaces tanto Serial como FastEthernet tienen la misma numeración, como se puede ver en la topología Figura 3.12. Los cambios realizados en este *playbook* se pueden ver en la Figura 3.31.

```
---
# Enrrutamiento Vector Distancia ripV2

- hosts: r2vd
  gather_facts: false
  remote_user: admin
  tasks:
    - name: Configuracion de Interfaces
      cisco.ios.ios_l3_interfaces:
        config:
        - name: FastEthernet0/1
          ipv4:
          - address: 192.168.30.254/24
        - name: Serial1/0
          ipv4:
          - address: 10.0.0.2/30
        state: merged

    - name: Habilitacion de Interfaces
      cisco.ios.ios_interfaces:
        config:
        - name: FastEthernet0/1
          enabled: false
          description: Configurado con Ansible J.S
        - name: Serial1/0
          enabled: false
          description: Configurado con ansible J.S conexion de serial
        state: merged
```

Figura 3.31 *Playbook* RIP *router* 2 primera parte

La tercera tarea Borrado de Protocolo no se realiza ningún cambio ya que son comandos de configuración propios de IOS para RIP; en la tarea de *Clock Rate* se elimina totalmente a causa de que en el *Router 1* ya se configuró el *clock rate.* En la cuarta y última tarea Enrutamiento ripv2 se debe cambiar las direcciones de red LAN, en la Figura 3.32 se observan los cambios realizados.

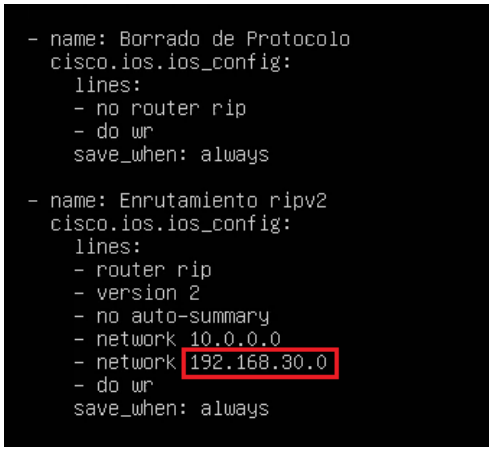

Figura 3.32 *Playbook* RIP *router* 2 segunda parte

Creación de *playbooks* para OSPF

Como se observa en la Figura 3.9 y Figura 3.10 las topologías tanto RIP y OSPF tienen las mismas direcciones IP, por lo tanto los *playbooks* para los dos *routers* de OSPF están estructurados de la misma forma que los *playbooks* de RIP, de modo que las configuraciones de inicio son las mismas a excepción de los *hosts* que determina a qué equipo del inventario configurar, ver Figura 3.23.

La tarea de Configuración de Interfaces es la misma, se toma en cuenta las IPs tanto para la interfaz serial como para la interfaz FastEthernet. La tarea de Habilitación de Interfaces especifica qué interfaces se habilitó, las cuales ya poseen una IP gracias a la tarea anterior. En la Figura 3.33 se observan las primeras tareas para configurar el *router* 1 con el protocolo de enrutamiento *link state* OSPF.

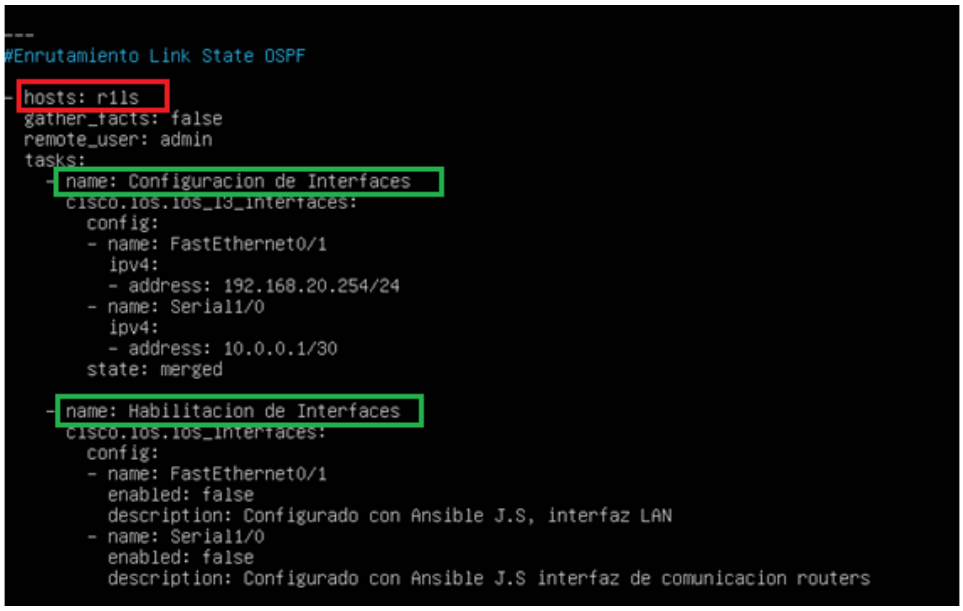

Figura 3.33 *Playbook* OSPF *router* 1 primera parte

En la Figura 3.34 se observan tres tareas, primero la de *clock rate* que es para establecer la velocidad de transmisión de datos a través de la serial, la segunda tarea es ***- name: Delet Protocol*** es el nombre de la tarea que se mostrará en el resumen de ejecución, ***cisco.ios.ios_ospfv2:*** es el módulo para administrar OSPF propios de IOS, la línea ***state: deleted*** indica que todos los procesos de OSPF que estén iniciados se los eliminará.

La tercera tarea Enrutamiento OSPF inicia con la línea ***- name: Enrutamiento ripv2***, luego la linea ***cisco.ios.ios_config:*** es el módulo para administrar las secciones de configuración con los comandos propios de IOS, la línea ***lines:*** abre el conjunto de comandos que deben estar ordenados para la configuración. La línea ***– router ospf 1*** habilita el protocolo de enrutamiento OSPF y el número 1 es el identificador de proceso de OSPF, la línea ***- network 192.168.20.0 0.0.0.255 area 0*** agrega la dirección de red de la LAN para la actualización de rutas, con la respectiva *wildcard* y se especifica el área de OSPF la cual es 0 ya que es de área única, ***- network 10.0.0.0 0.0.0.3 area 0*** agrega la dirección de red de la serial para actualizar rutas, agrega la *wilcard* y especifica el área que también es de área única.

Se coloca ***- do wr*** para guardar las configuraciones realizadas en el dispositivo, ***save_when: always*** guarda las configuraciones de ejecución o *running-config* en las configuraciones de inicio o *startup-config*.

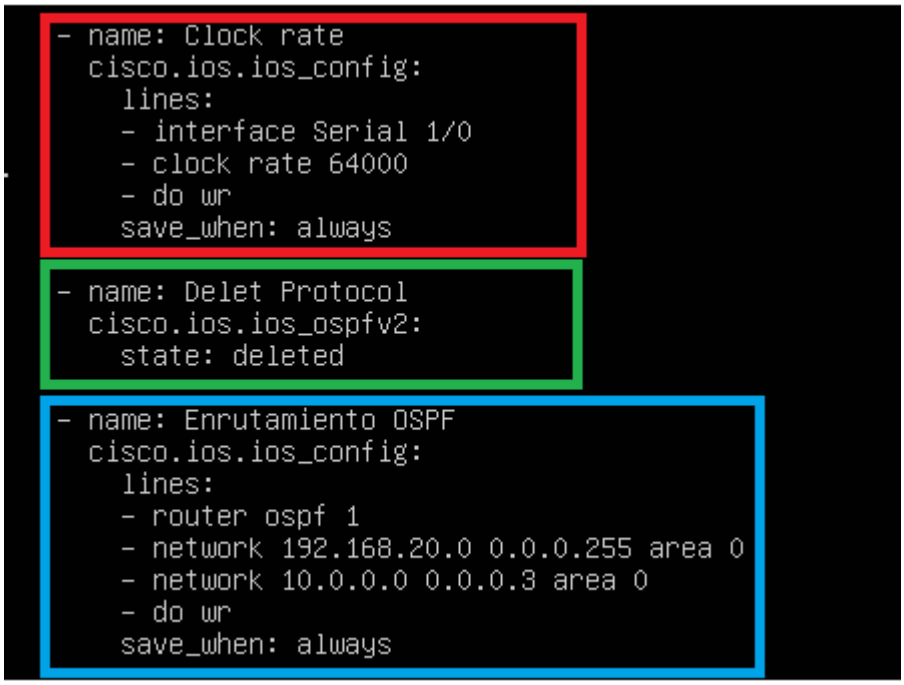

Figura 3.34 *Playbook* OSPF *router* 1 segunda parte

Una vez creado el *playbook* del *Router 1,* se procede a crear el *playbook* del *Router* 2 denominado ***ospfRouter2.yml***, para lo cual se hace una copia usando el comando ***sudo cp ospfRouter1.yml ospfRouter2.yml,*** como se observa en la Figura 3.35. En la Figura 3.36 se observa el *playbook*.

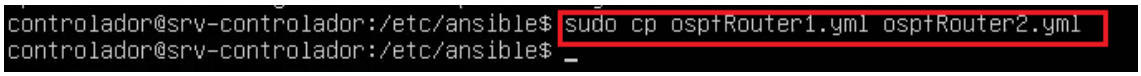

Figura 3.35 Comando para hacer copia de *playbook*

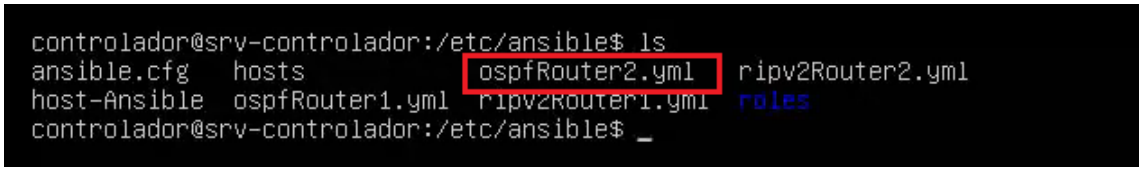

Figura 3.36 *Playbook* de *router* 2 OSPF

En el *playbook* del *router* 2 se debe cambiar el *host*, especificando el nombre del router2, establecido en el inventario como r2ls, ver Figura 3.23. En la tarea Configuración de Interfaces se cambia las direcciones IP de la FastEthernet a 192.168.30.254/24 y de la Serial a 10.0.0.2/30. Para la segunda tarea Habilitación de Interfaces no se realiza cambios ya que las interfaces a los que están conectados los *routers* tiene la misma numeración, como se puede ver en la Figura 3.10, los cambios realizados en el *playbook* se observan en la Figura 3.37.

Figura 3.37 *Playbook* OSPF de *router* 2 primera parte

En la tercera tarea *Delet Protocol* no se realiza ningún cambio ya que el módulo de administración es propio de la IOS y los comandos son los mismos. La tarea de *Clock Rate* se elimina totalmente, a causa de que en el *Router 1* ya se configuró el *clock rate.* En la cuarta y última tarea de Enrutamiento OSPF se deben cambiar las direcciones de red LAN, en la Figura 3.38 se pueden observar los cambios realizados.

```
- name: Delet Protocol
  cisco.ios.ios_ospfv2:
    state: deleted

- name: Enrutamiento OSPF
  cisco.ios.ios_config:
    lines:
    - router ospf 1
    - network 192.168.30.0 0.0.0.255 area 0
    - network 10.0.0.0 0.0.0.3 area 0
    - do wr
    save_when: always
```

Figura 3.38 *Playbook* OSPF de *router 2* segunda parte

3.4 Pruebas de Funcionamiento y verificación

Configuración inicial de equipos para RIP

Previo a ejecutar los *playbooks,* en la topología RIP se les establece una dirección IP a cada uno de los *routers* en la interfaz FastEthernet con la cual se van a conectar con el nodo controlador. En la Figura 3.39, se hace uso del comando ***show ip interface brief*** donde se observa que el *router 1* tiene una dirección IP en la FastEthernet0/0. En la Figura 3.40 también se observa la dirección IP del *router 2* en la FastEthernet0/0; las direcciones IPs de *routers* y controlador están en la red 192.168.10.0/24.

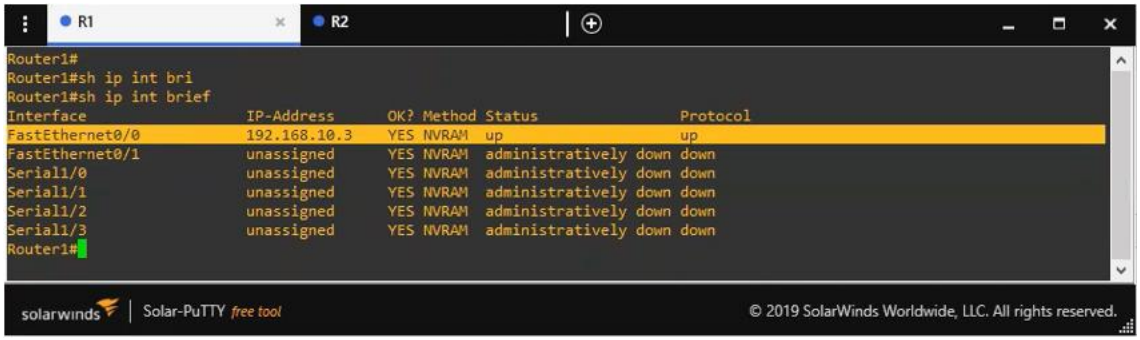

Figura 3.39 Información inicial de *router 1* RIP

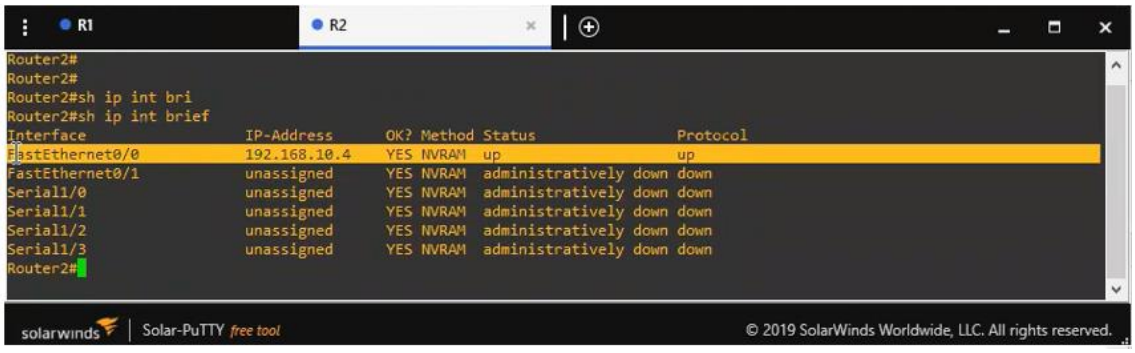

Figura 3.40 Información Inicial de *router 2* RIP

También usando el comando ***show running-config*** se observa el estado de las configuraciones del *router.* En la Figura 3.41 se observa que el *router 1* de la topología RIP tiene un *hostname* y está creado un *userrname.* En la Figura 3.42 se observa la interfaz FastEthernet0/0 la cual es la única que tiene una dirección IP, tiene creado un usuario, las demás interfaces tanto FastEthernet como Serial están vacías, así mismo no posee protocolo de enrutamiento y la conexión para comunicación a través de SSH está establecida.

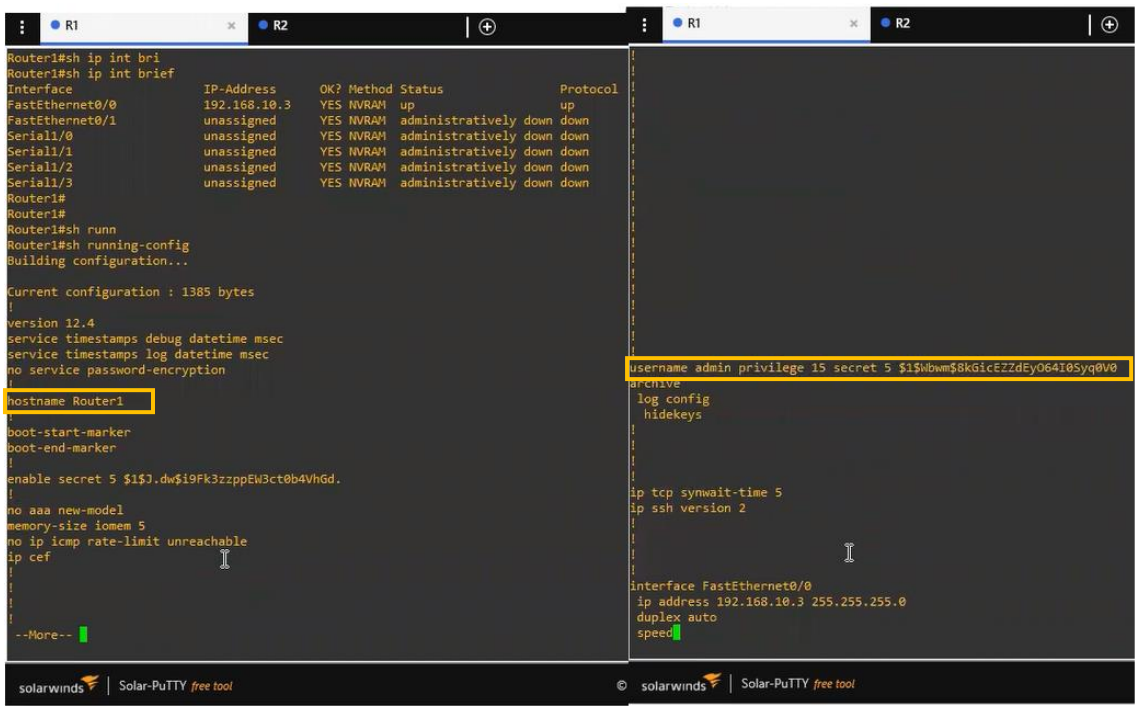

Figura 3.41 Información inicial de *running-config router 1* RIP primera parte

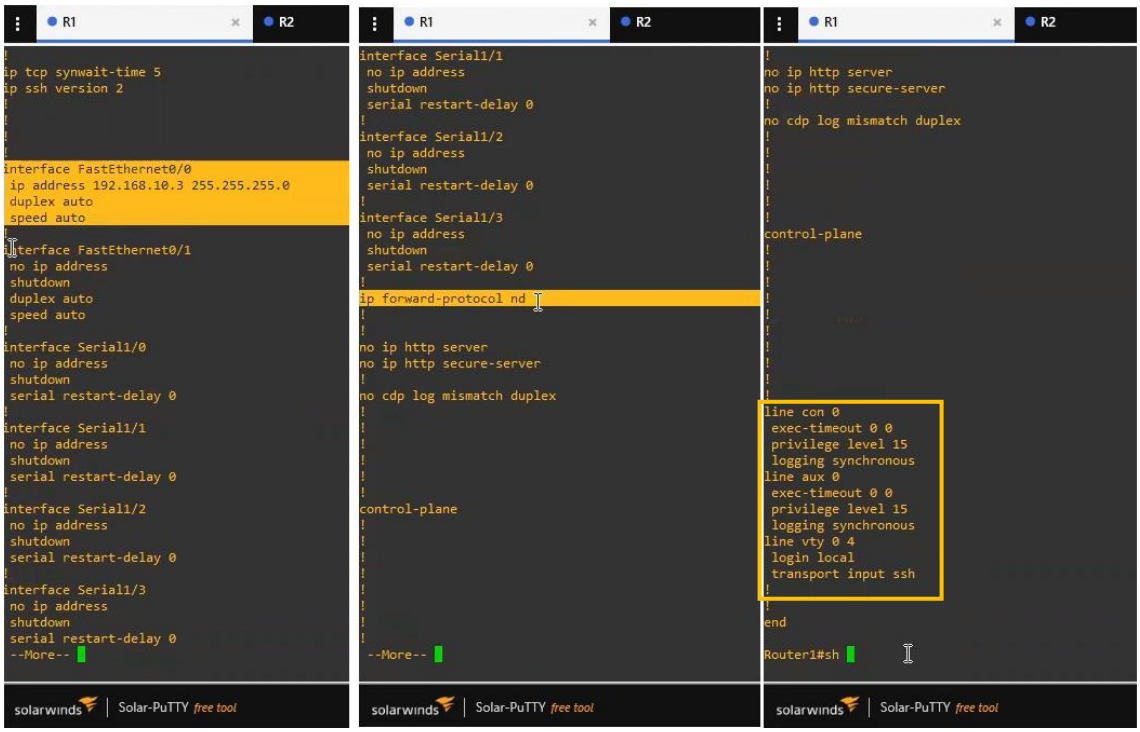

Figura 3.42 Información inicial de *running-config router 1* RIP segunda parte

De la misma forma en la Figura 3.43 y Figura 3.44 se observan las configuraciones del ***show running-config*** del *router* 2 de la topología RIP.

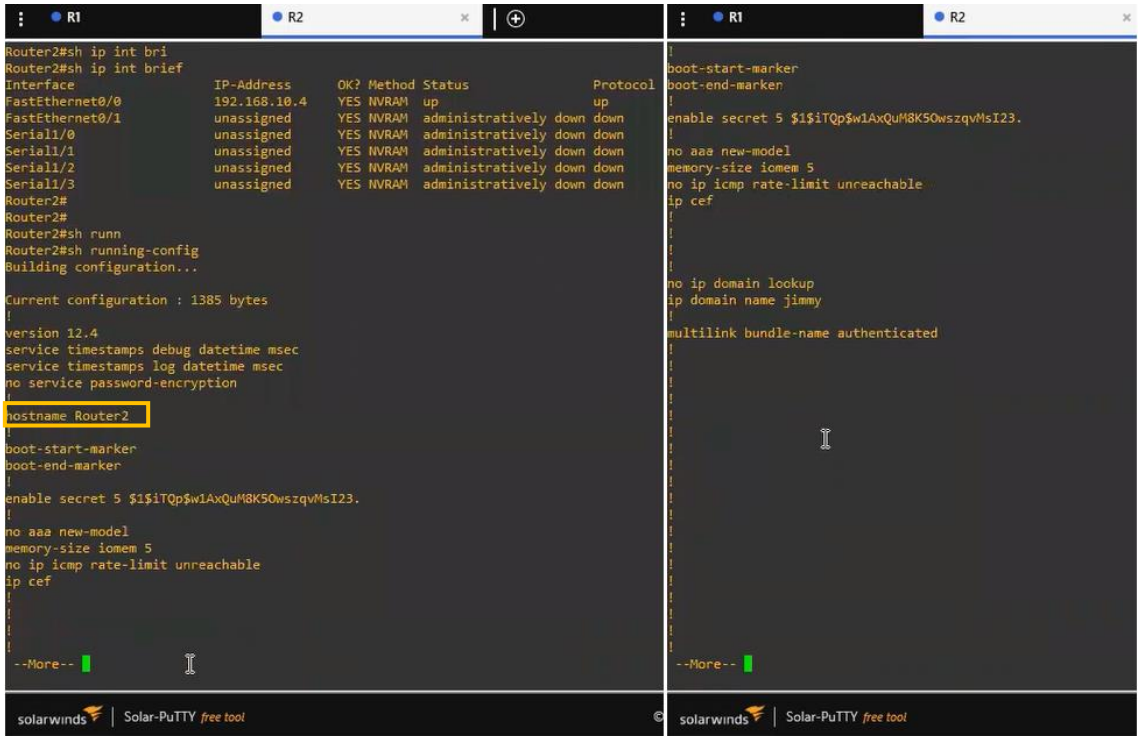

Figura 3.43 Información inicial de *running-config router* 2 RIP primera parte

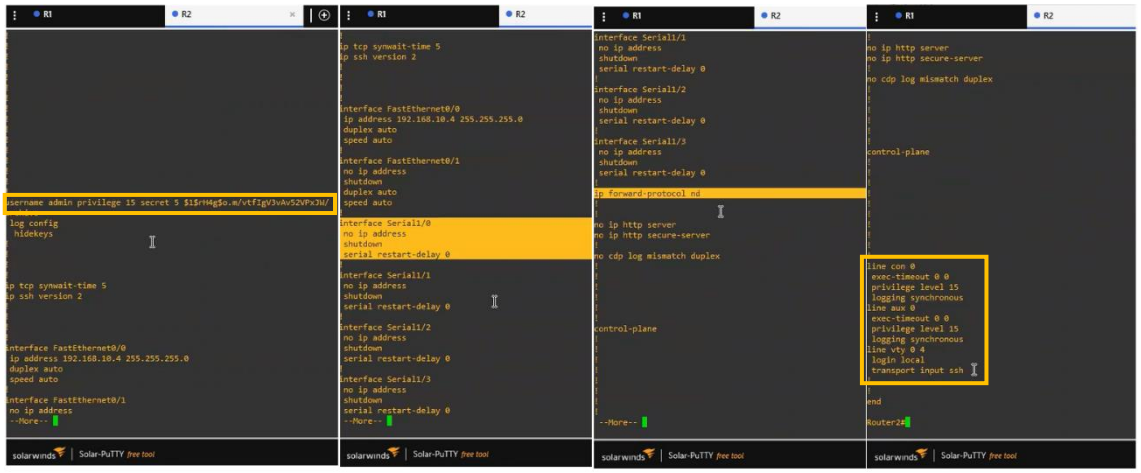

Figura 3.44 Información inicial de *running-config router* 2 RIP segunda parte

Ejecución del *Playbook* para la topología RIP

Una vez creado el *playbook* se debe ejecutar usando el comando ***playbook-ansible name.yml*** el *name* es el nombre del *playbook* con el que se ha creado, como se puede observar en la Figura 3.45 el primer *playbook* a ejecutar es el de ripv2Router1.yml. La ejecución del *playbook* se puede observar en la Figura 3.45 el cual ejecuta cinco tareas en las cuales cada tarea realiza un cambio al equipo, al final se muestra un resumen del número de tareas y cambios realizados.

```
controlador@srv-controlador:/etc/ansible$ ansible-playbook ripv2Router1.yml

PLAY [r1vd] ********************************************************************************************************

TASK [Configuracion de Interfaces] *********************************************************************************
changed: [Router1]

TASK [Habilitacion de Interfaces] **********************************************************************************
changed: [Router1]

TASK [Clock Rate] **************************************************************************************************
[WARNING]: To ensure idempotency and correct diff the input configuration lines should be similar
to how they appear if present in the running configuration on device
changed: [Router1]

TASK [Borrado de Protocolo] ****************************************************************************************
changed: [Router1]

TASK [Enrutamiento ripv2] ******************************************************************************************
changed: [Router1]

PLAY RECAP *********************************************************************************************************
Router1                    : ok=5    changed=5    unreachable=0    failed=0    skipped=0    rescued=
0    ignored=0
```

Figura 3.45 Ejecución de RIP *router* 1

Para el ejecutar el *playbook* en el *router*2 se cambia la conexión y se usa el comando ***ansible-playbook ripv2Router2.yml*** el cual realiza cuatro tareas de cambios en el equipo, esto se puede observar en la Figura 3.46 mostrando las tareas realizadas.

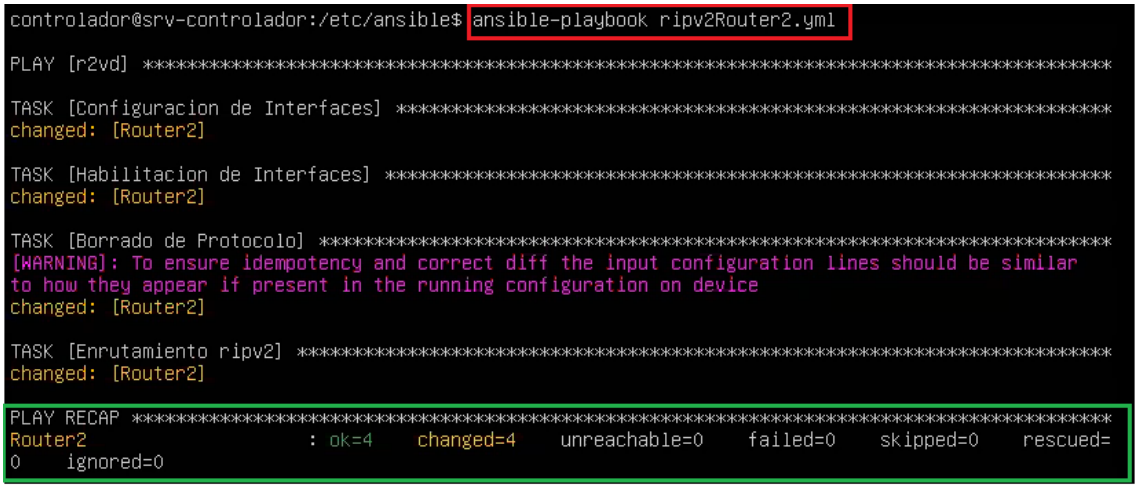

Figura 3.46 Ejecución *Playbook* RIP *router2*

Verificación de los cambios en la topología RIP

Para verificar los cambios realizados en cada *router,* desde el controlador se ingresa a la consola para desde allí ejecutar los comandos como el ***show ip interface brief*** para verificar las direcciones IPs asignadas a las interfaces de cada uno de los *routers.* En la Figura 3.47 se observa la configuración de las interfaces del *router*1 y en la Figura 3.48 se observa la configuración del *router*2.

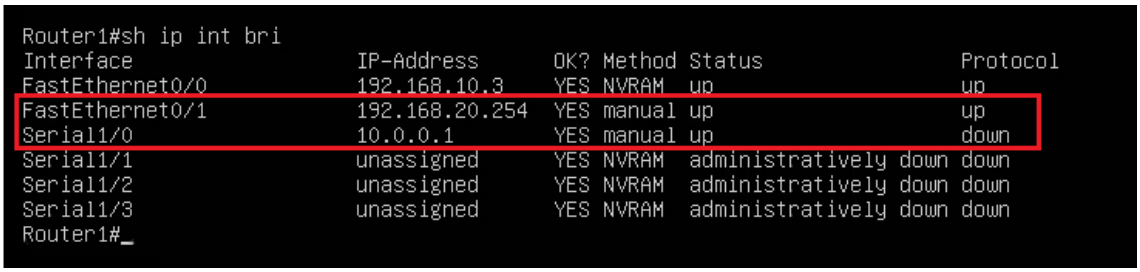

Figura 3.47 Información de configuración de interfaces en *router* 1 de RIP

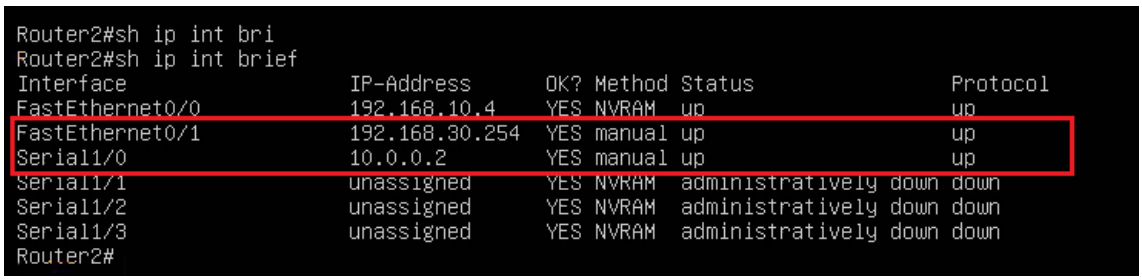

Figura 3.48 Información de configuración de interfaces en *router* 2 de RIP

Para verificar y obtener información más detallada de la configuración, se hace uso del comando ***show running-config*** en cada *router*. En la Figura 3.49 se observa información del *router 1*; la interfaz serial posee una descripción, *clock rate* y dirección IP, todo esto agregado con Ansible a través del *playbook*, además se observa el protocolo de enrutamiento RIP en la versión 2. En la Figura 3.50 se visualizan los cambios realizados en el *router 2*, a excepción del *clock rate.*

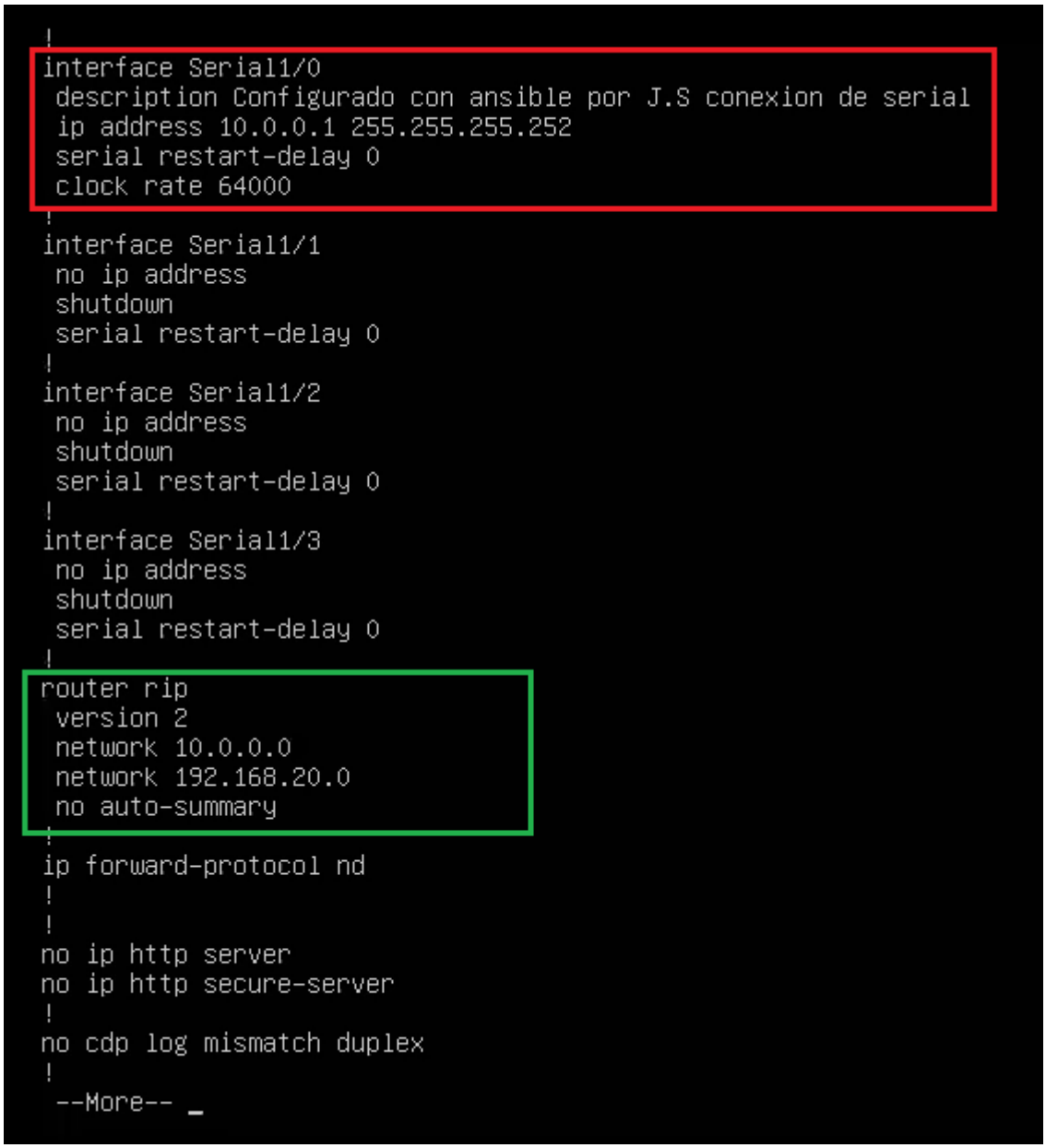

Figura 3.49 Información de running-config *router* 1 RIP

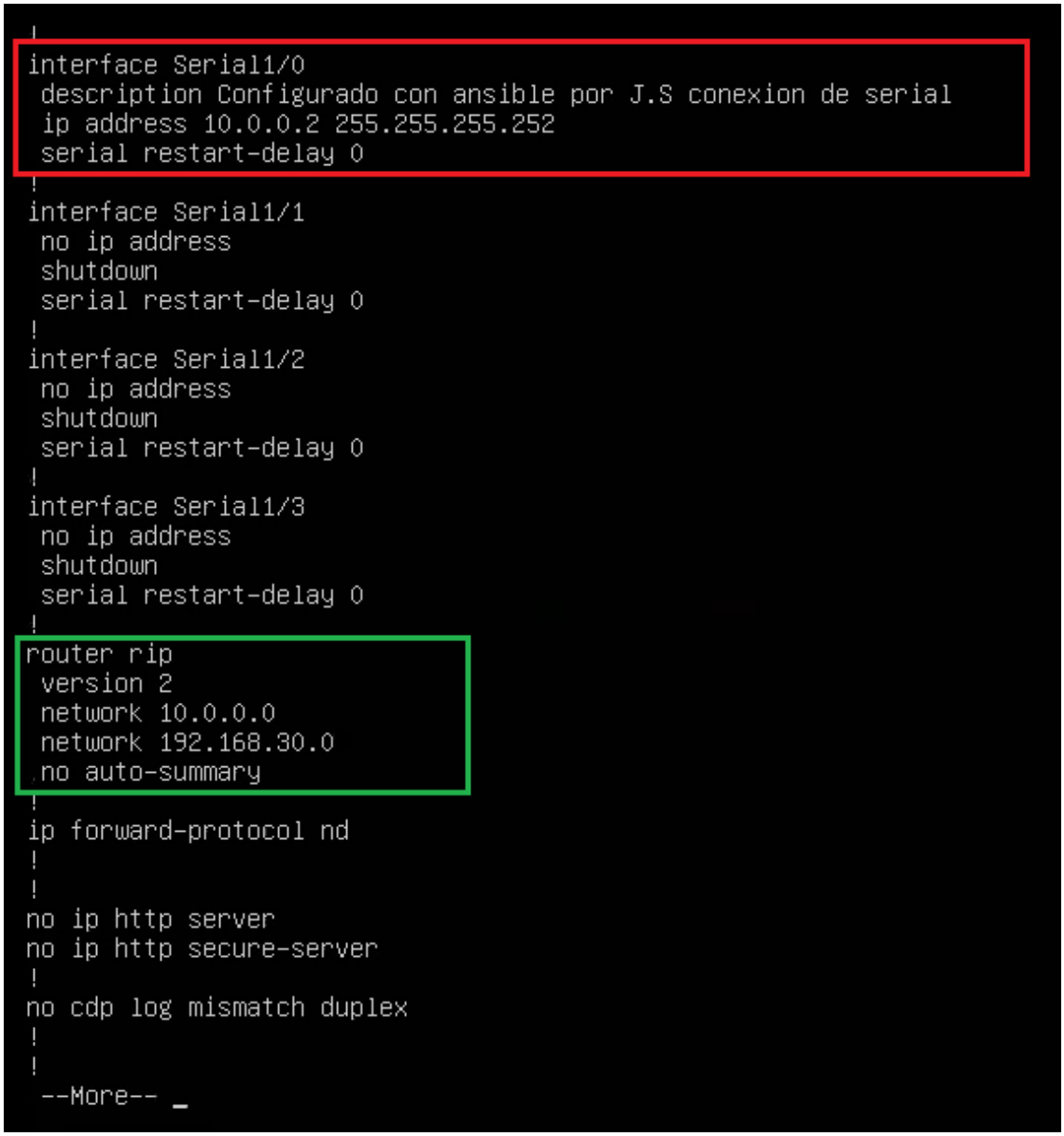

Figura 3.50 Información de running-config *router* 2 RIP

Tabla de Enrutamiento IP Protocolo RIP

Para verificar el protocolo RIP se hace uso del comando ***Show ip route*** en el cual se observa en la Figura 3.51 para el *router 1* y la Figura 3.52 para el *router2*

Figura 3.51 Tabla de enrutamiento RIP *router* 1

Figura 3.52 Tabla de enrutamiento RIP *router* 2

Pruebas de conectividad topología RIP

Una vez comprobados los cambios en las interfaces y verificado el protocolo de enrutamiento en cada *router* atraves de las tablas de enrutamiento, se realiza una comprobación entre los equipos configurados y las redes LAN. Refiriéndose a la Figura 3.9 que describe la topología, en la Figura 3.53 se observa el *ping* exitoso realizado desde el *router 1* hacia la interfaz FastEthernet del *router 2*, interfaz con dirección IP 192.168.30.254. En la Figura 3.54 se observa el *ping* exitoso realizado desde el *router 2* hacia la interfaz FastEthernet del *router 1*, interfaz con dirección IP 192.168.20.254.

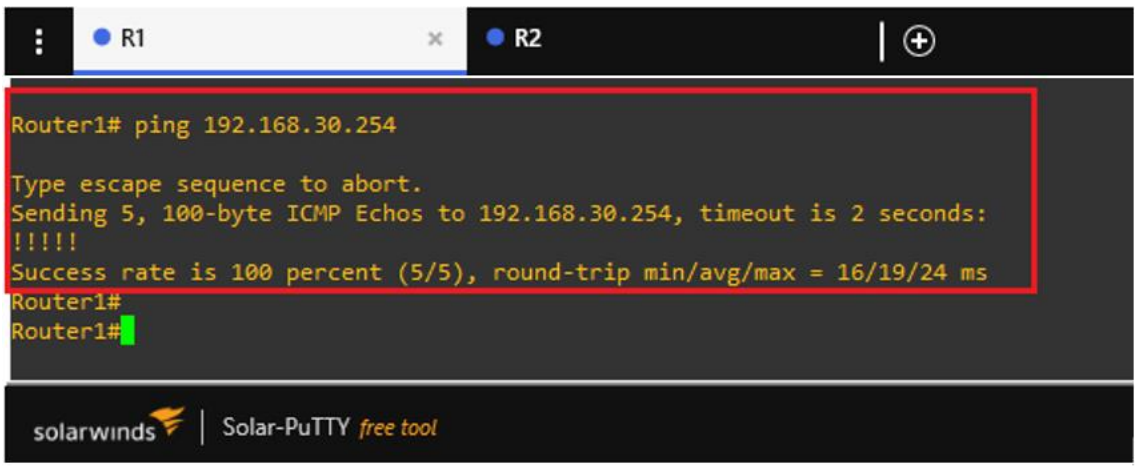

Figura 3.53 *Ping* del *router 1* a la interfaz LAN del *router 2,* topología RIP

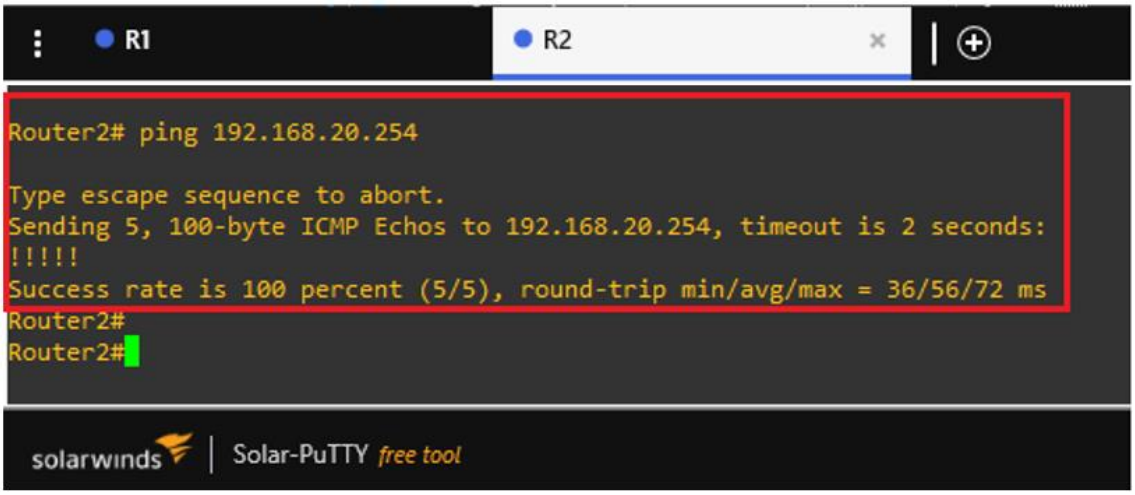

Figura 3.54 *Ping* del *router 2* a la interfaz LAN del *router 1,* topología RIP

Así mismo se observa en la Figura 3.55 un *ping* exitoso desde la PC1 de la red LAN del *router 1* con dirección IP 192.168.20.2 hacia la PC2 de la red LAN del *router 2* con dirección IP 192.168.30.2. Además, en la Figura 3.56 se observa el *ping* realizado desde la PC2 hacia la PC1, ambos *pings* realizados exitosamente, comprobando así que el enrutamiento RIP fue establecido a través de Ansible.

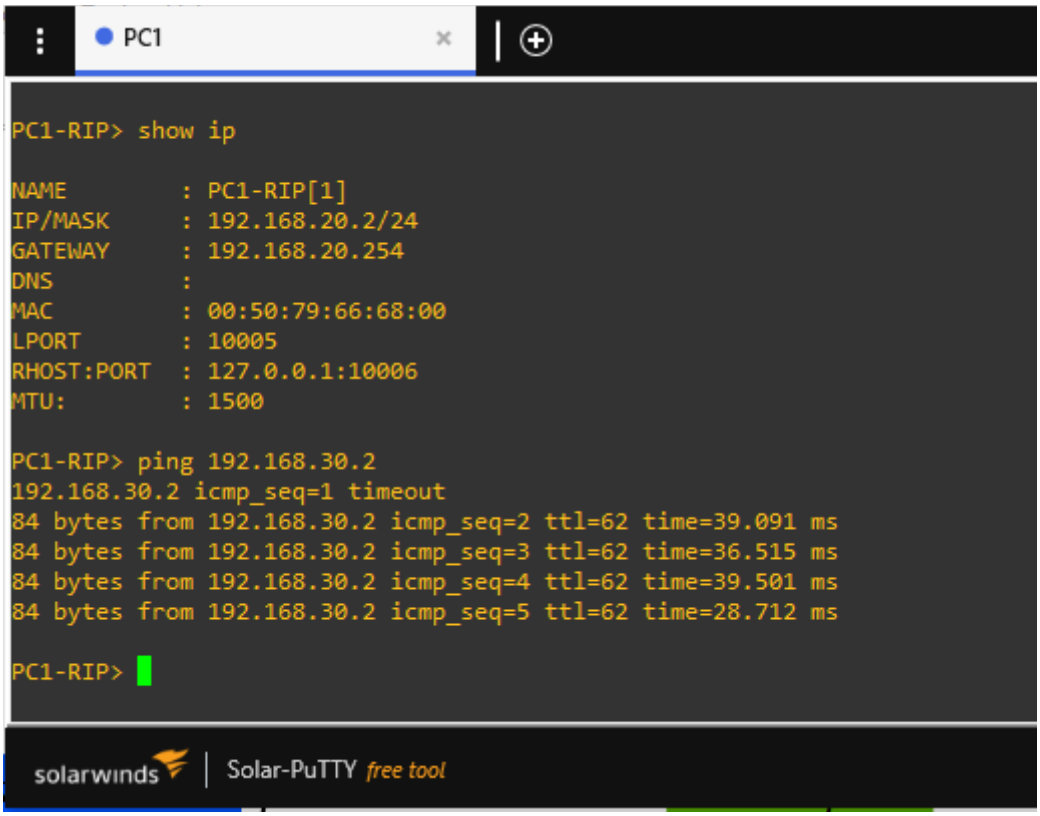

Figura 3.55 *Ping* desde PC1 hacia PC2 topología RIP

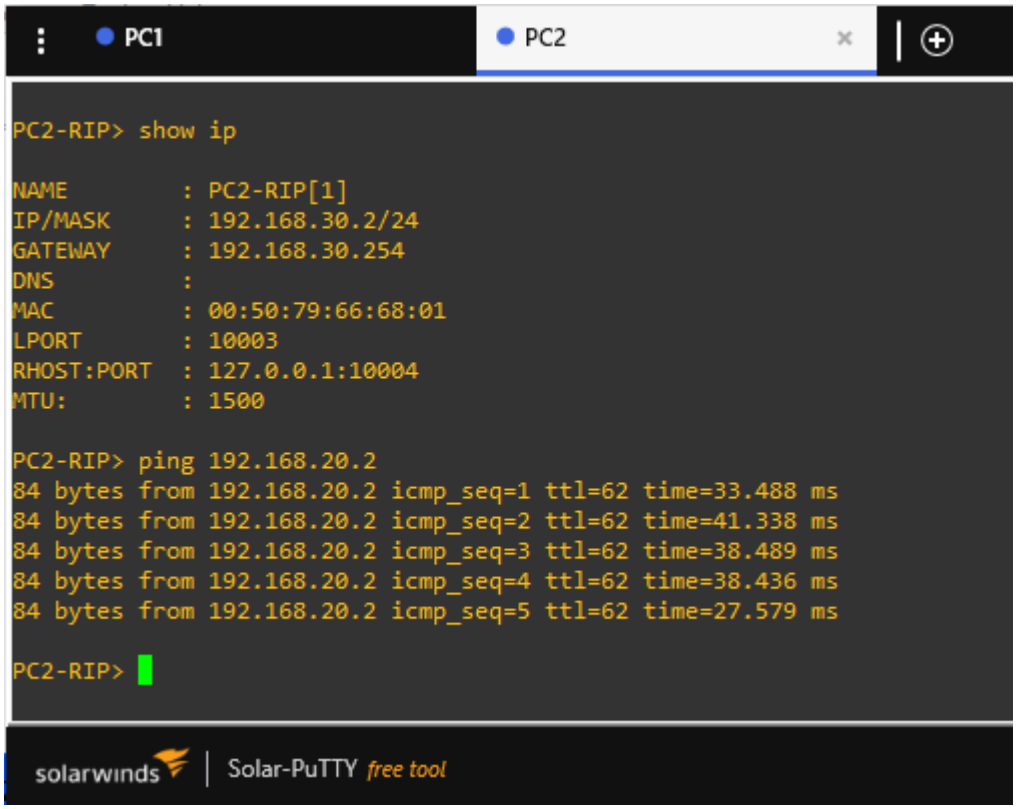

Figura 3.56 *Ping* desde PC2 hacia PC1 topología RIP

Configuración inicial de equipos para OSPF

En la topología OSPF los *routers* también se les ha colocado una dirección IP para comunicarse con el nodo controlador. En la interfaz FastEthernet, como se observa en la Figura 3.57, se hace uso del comando ***show ip interface brief*** donde se observa que el *router 1* tiene la dirección IP 192.168.10.5 en la FastEthernet0/0/, lo cual permite estar en red con el nodo controlador. Así mismo en la Figura 3.58 se observa la dirección IP del *router 2.*

Figura 3.57 Información Inicial de *router 1* OSPF

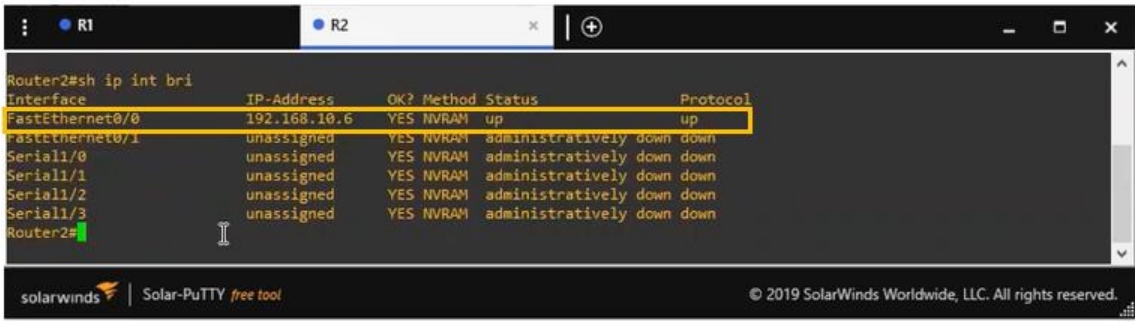

Figura 3.58 Información Inicial de *router 2* OSPF

En la Figura 3.60 y Figura 3.60 se observa el estado de las configuraciones del *router 1* mostradas con el comando ***show running-config.***

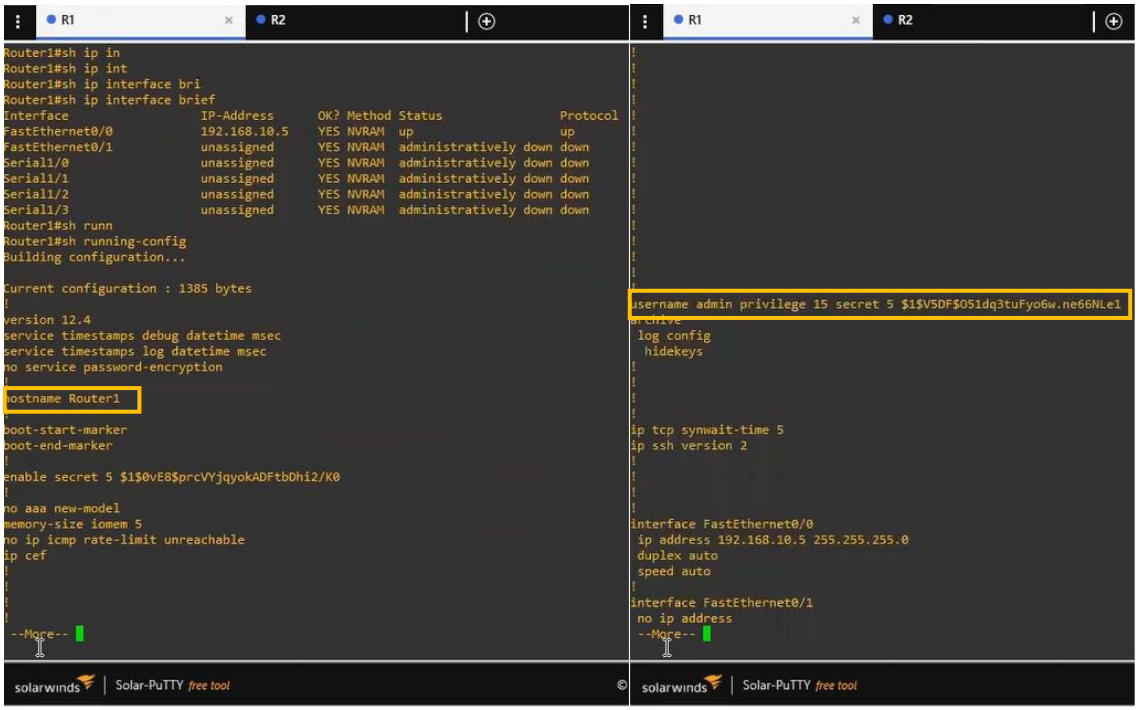

Figura 3.59 Información inicial de *running-config router 1* OSPF primera parte

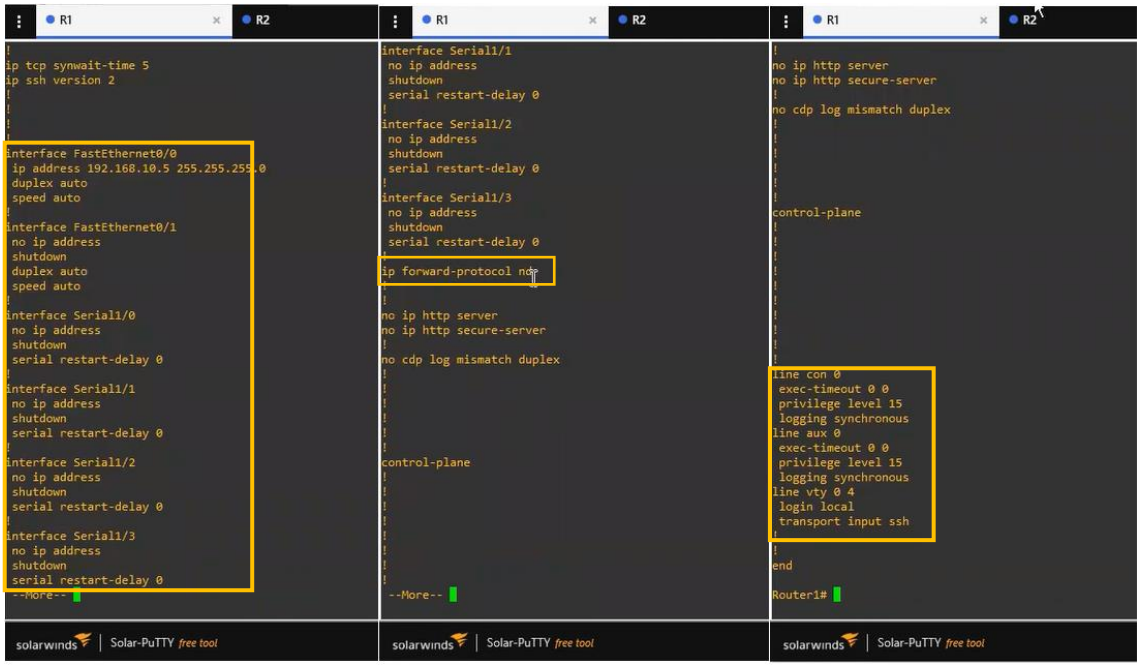

Figura 3.60 Información inicial de *running-config router 1* OSPF segunda parte

En la Figura 3.62 y Figura 3.62 se observa el estado de las configuraciones del *router 2* mostradas con el comando ***show running-config.***

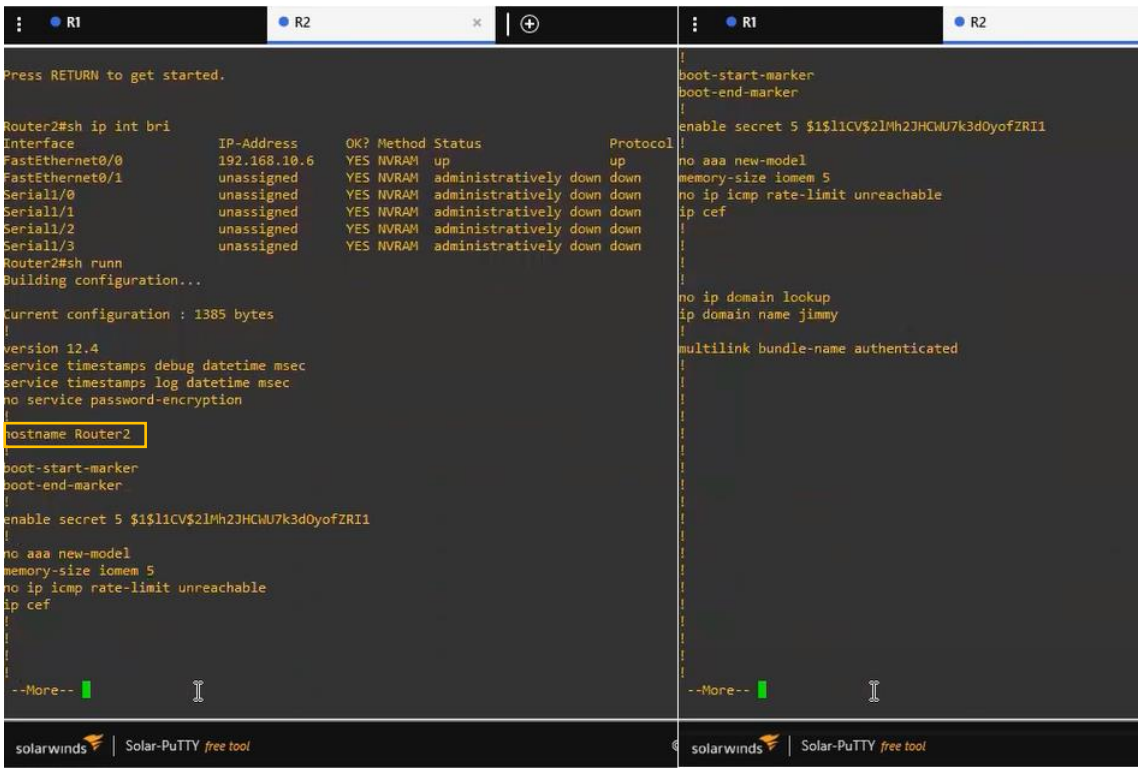

Figura 3.61 Información inicial de *running-config router 2* OSPF primera parte

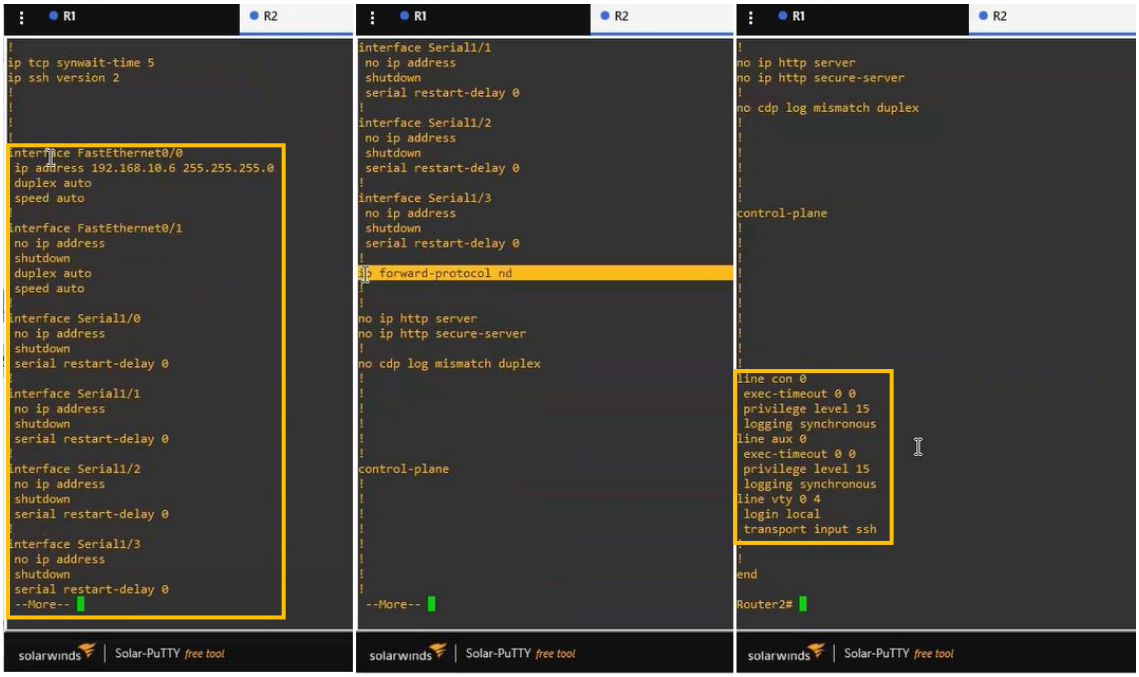

Figura 3.62 Información inicial de *running-config router 2* OSPF segunda parte

Ejecución del *Playbook* para la topología OSPF

Una vez conectado el controlador a la topología para OSPF con el comando ***ansible-playbook ospfRouter1.yml*** se ejecuta el *playbook* con el cual se realizan cinco tareas para configurar el *router 1*, estas corresponden a la configuración de interfaces, habilitación de interfaces, *clock rate, delet protocol* y enrutamiento. La ejecución se verifica en la Figura 3.63 en el cual se observa el resumen de las cambios y tareas realizadas. El mensaje mostrado en color morado son advertencias de cómo se debe escribir las líneas de comando dentro del módulo de configuración.

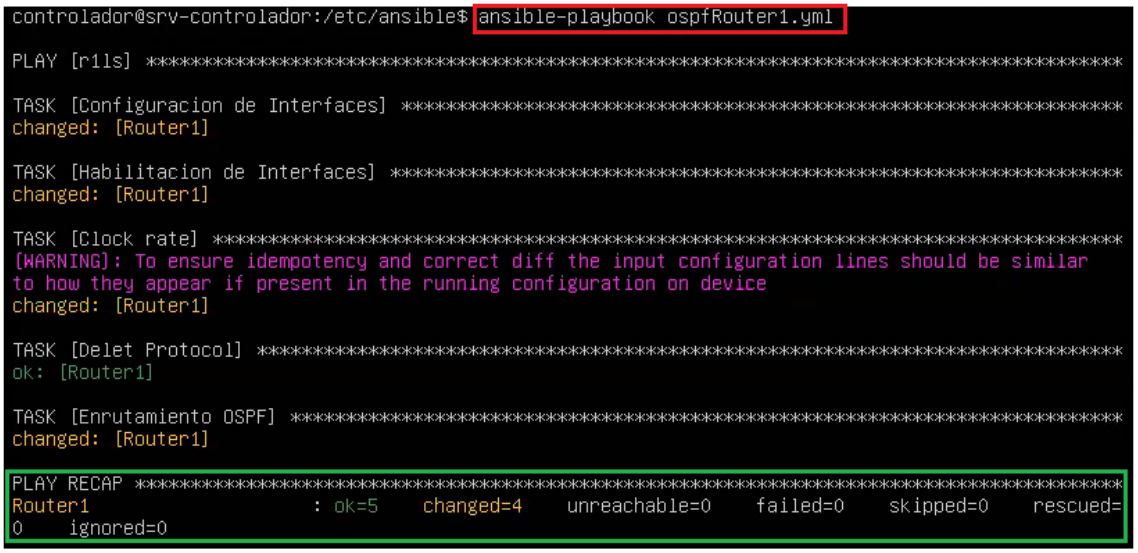

Figura 3.63 Ejecución de *Playbook* OSPF *router* 1

Luego se cambia la conexión al *router* 2 y con el comando ***ansible-playbook ospfRouter2.yml*** se ejecuta el ***playbook,*** el cual realiza cuatro tareas de cambios en el equipo, recordando que aquí no se ejecuta la tarea de *clock rate*, se muestra en la Figura 3.64 las tareas realizadas.

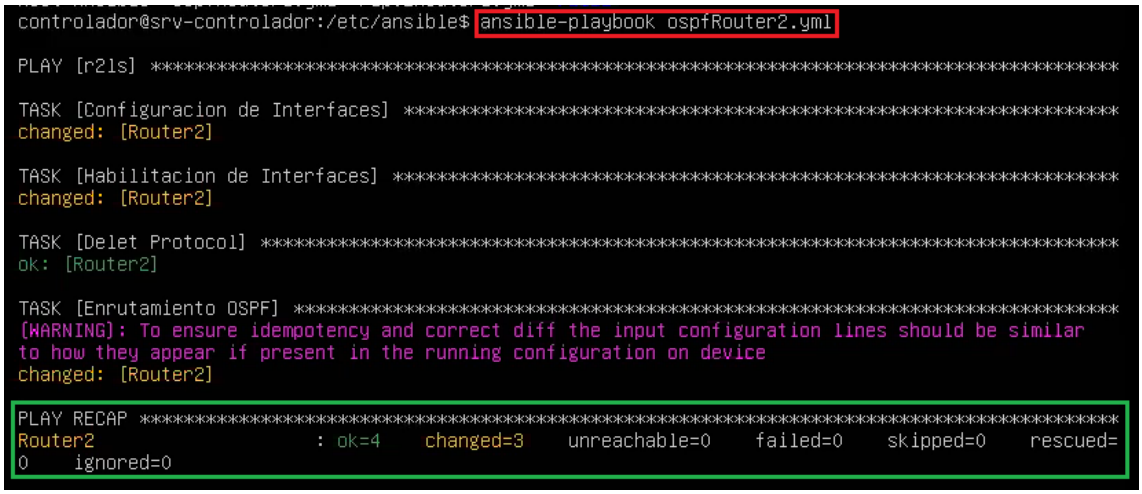

Figura 3.64 Ejecución de *Playbook* OSPF *router* 2

Verificación de los cambios topología OSPF

Una vez ejecutados los *playbooks* OSPF y a través de SSH se ingresa a la consola para desde allí ejecutar ***show ip interface brief*** para verificar las direcciones IPs asignadas a las interfaces de cada uno de los *routers.* En la Figura 3.65 se observan las configuraciones de las interfaces del *router1* y en la Figura 3.66 se observan las configuraciones de las interfaces del *router*2.

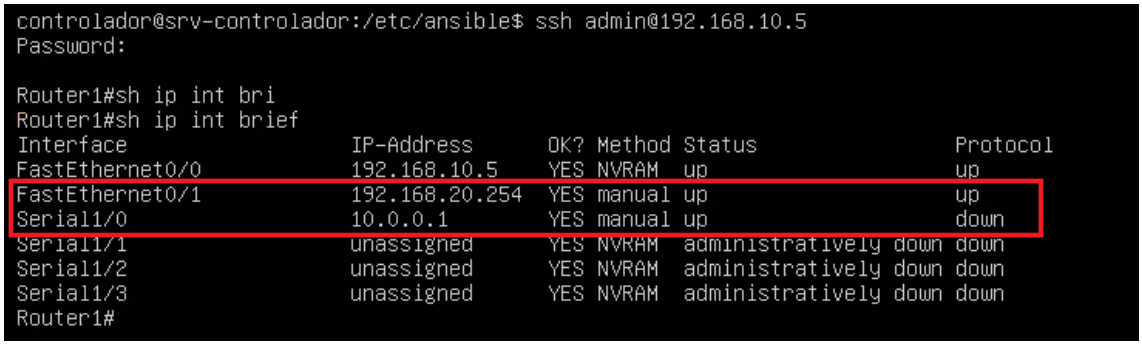

Figura 3.65 Información de configuración de interfaces en *router* 1 de OSPF

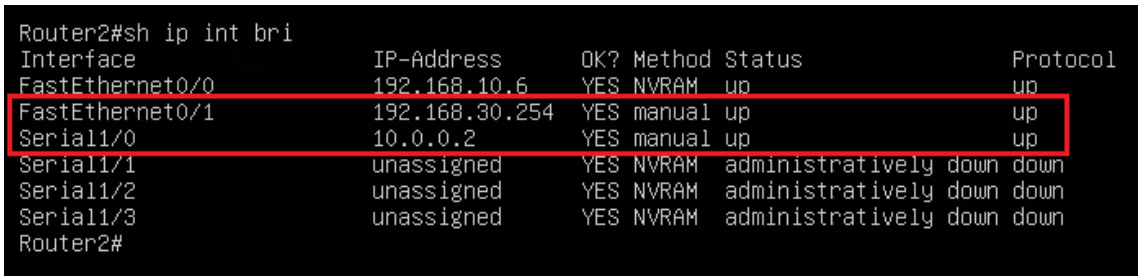

Figura 3.66 Información de configuración de interfaces en *router* 2 de OSPF

Y para verificar con más detalle las configuraciones se hace uso del comando ***show running-config*** en cada *router*. En la Figura 3.67 se observa la información del *router 1,* la interfaz serial posee una descripción, *clock rate*, dirección IP y enrutamiento OSPF, todo esto agregado con Ansible a través del *playbook*. En la Figura 3.68 se visualizan los cambios realizados en el *router 2* a excepción del *clock rate.*

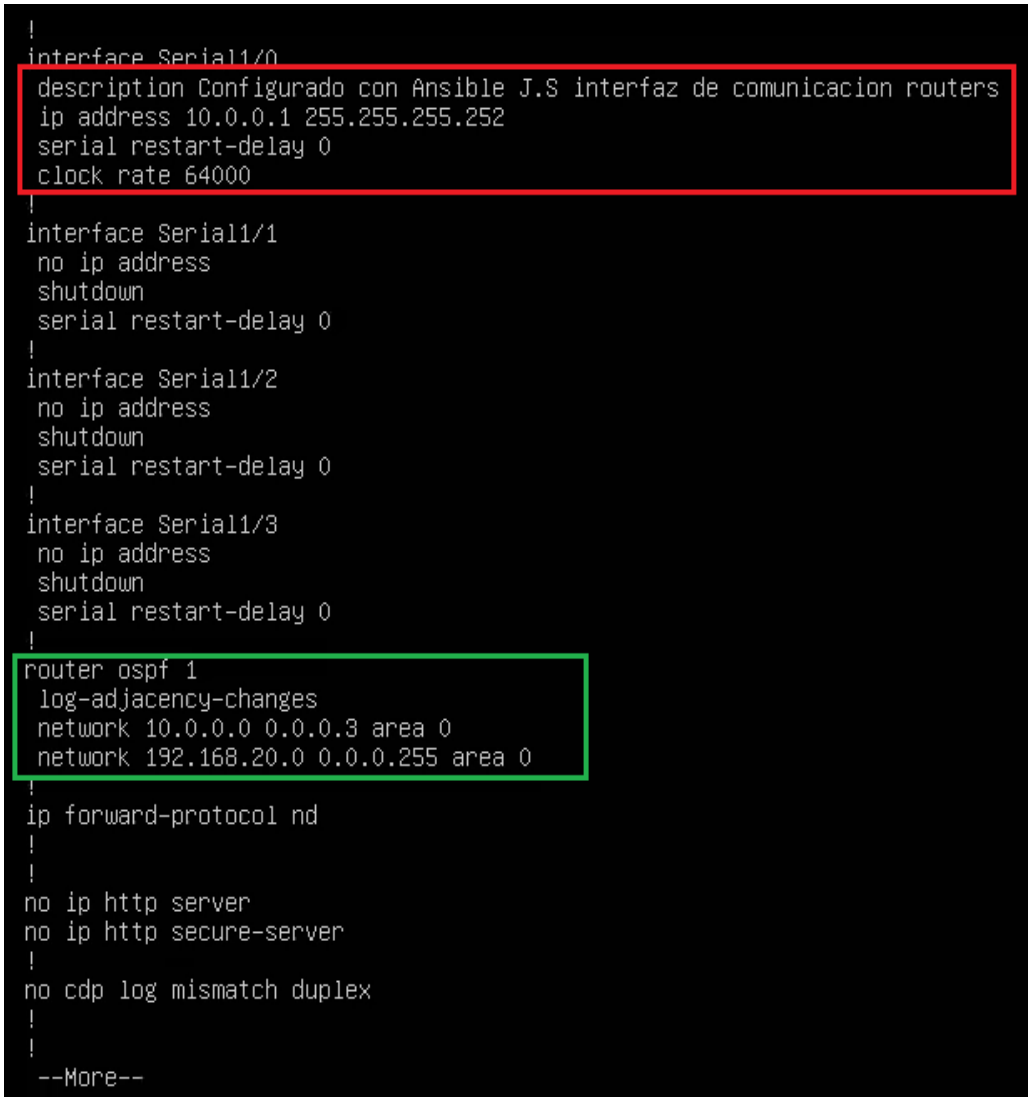

Figura 3.67 Información de running-config *router* 1 OSPF

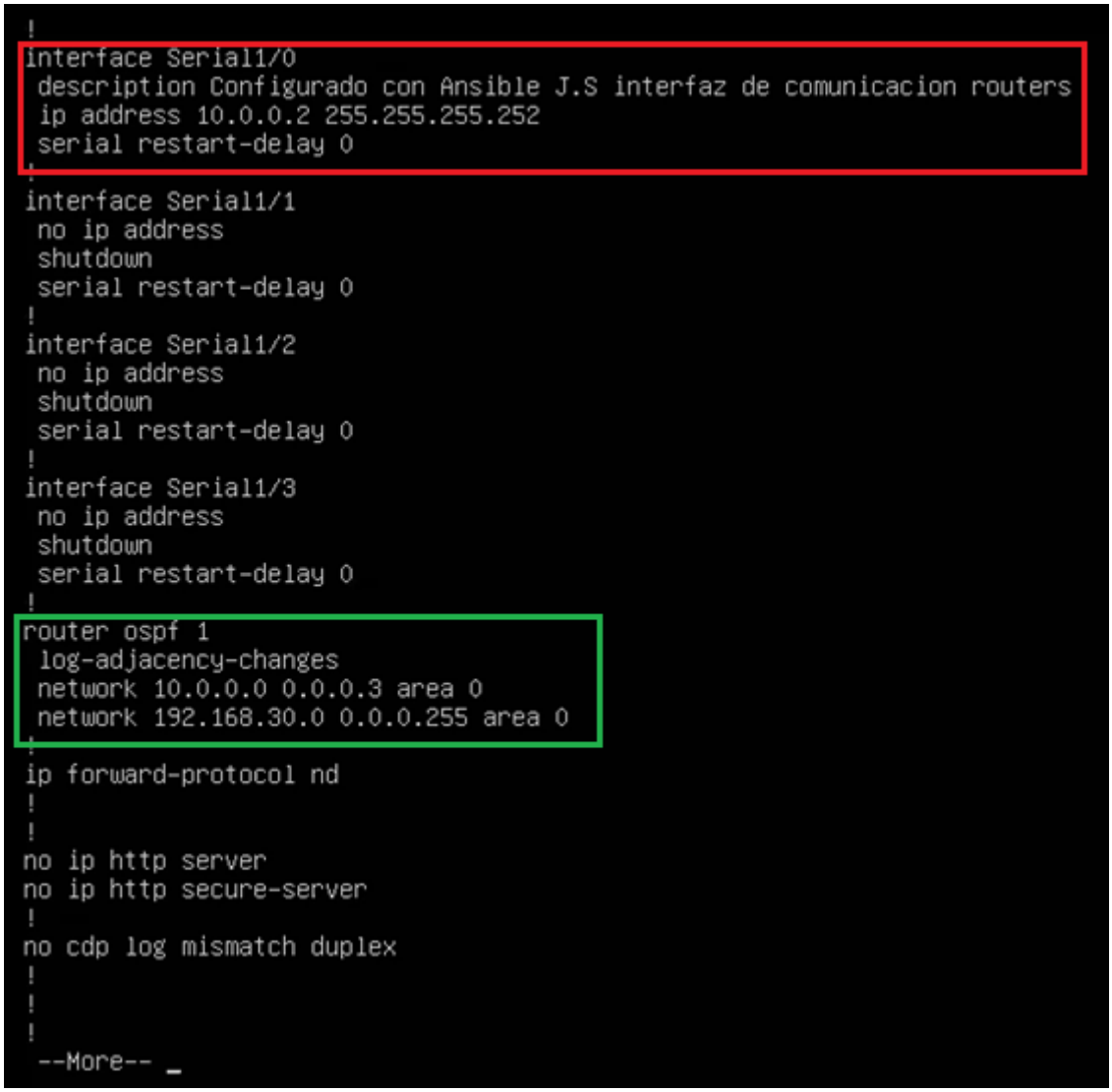

Figura 3.68 Información de running-config *router* 2 OSPF

Tabla de Enrutamiento IP protocolo OSPF

En la Figura 3.69 y Figura 3.70 se usa el comando ***show ip route*** con el cual se muestra el protocolo de enrutamiento OSPF de cada uno de los *routers*.

Figura 3.69 Tabla de enrutamiento OSPF *router* 1

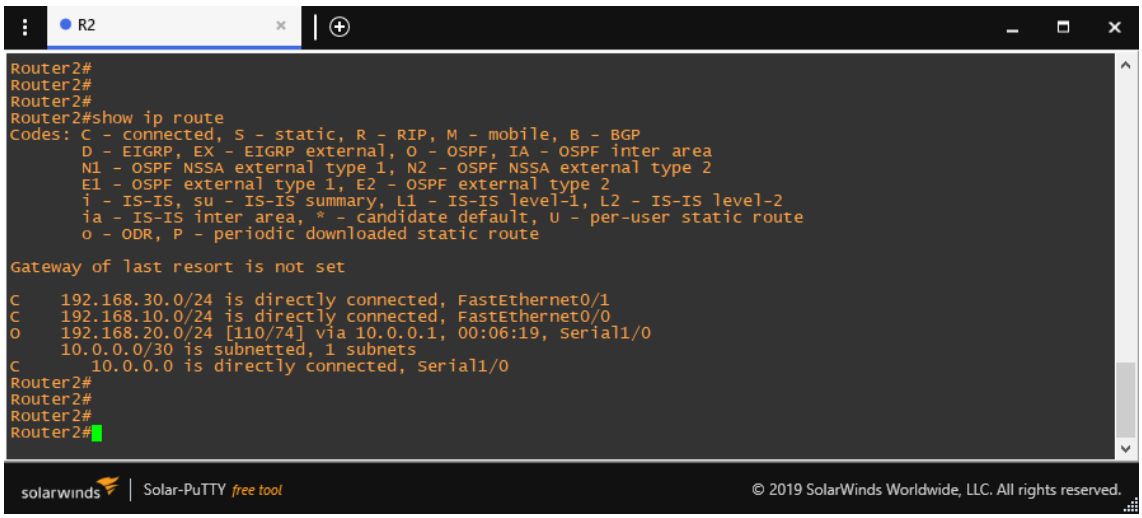

Figura 3.70 Tabla de enrutamiento OSPF *router* 2

Pruebas de conectividad topología OSPF

Una vez comprobados los cambios en las interfaces y verificado el protocolo de enrutamiento en cada *router*, se comprueba la conectividad entre los equipos configurados y las redes LAN. Refiriéndose a la Figura 3.10 que describe la topología, en la Figura 3.71 se puede observar el *ping* exitoso realizado desde el *router 1* hacia la interfaz FastEthernet del *router* 2, interfaz con dirección IP 192.168.30.254. En la Figura 3.72 se observa el *ping* exitoso desde el *router* 2 hacia la interfaz FastEthernet del *router* 1, interfaz con dirección IP 192.168.20.254.

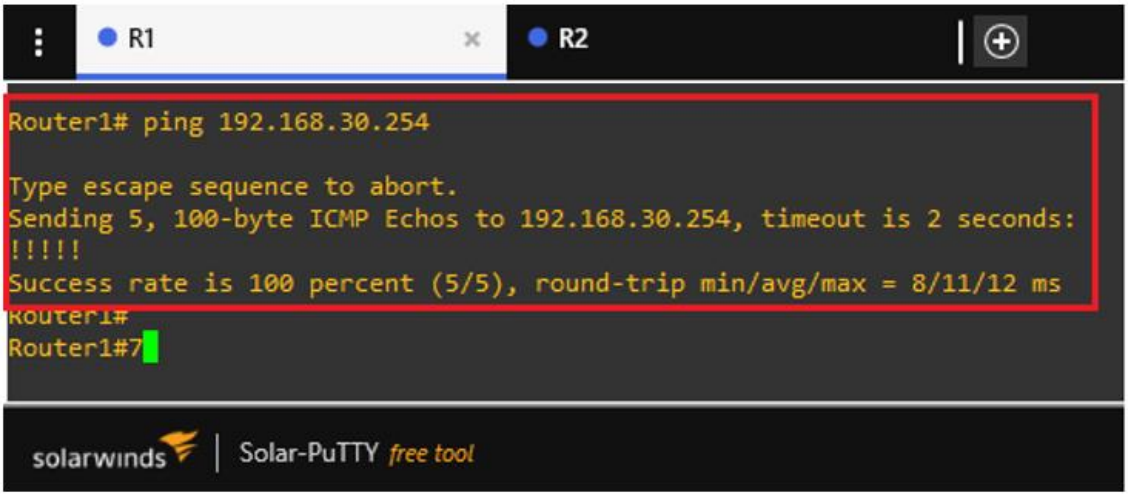

Figura 3.71 *Ping* del *router 1* a la interfaz LAN del *router 2, topología OSPF*

Figura 3.72 *Ping* del *router 2* a la interfaz LAN del *router 1,* topología OSPF

Así mismo en la Figura 3.73 se observa un *ping* exitoso desde la PC1 con dirección IP 192.168.20.2 de la red LAN del *router 1* hacia la PC2 con dirección IP 192.168.30.2 de la red LAN del *router 2.* Además, en la Figura 3.74 se observa el *ping* realizado desde la PC2 hacia la PC1, ambos *pings* realizados exitosamente, comprobando así que el enrutamiento OSPF fue establecido a través de Ansible.

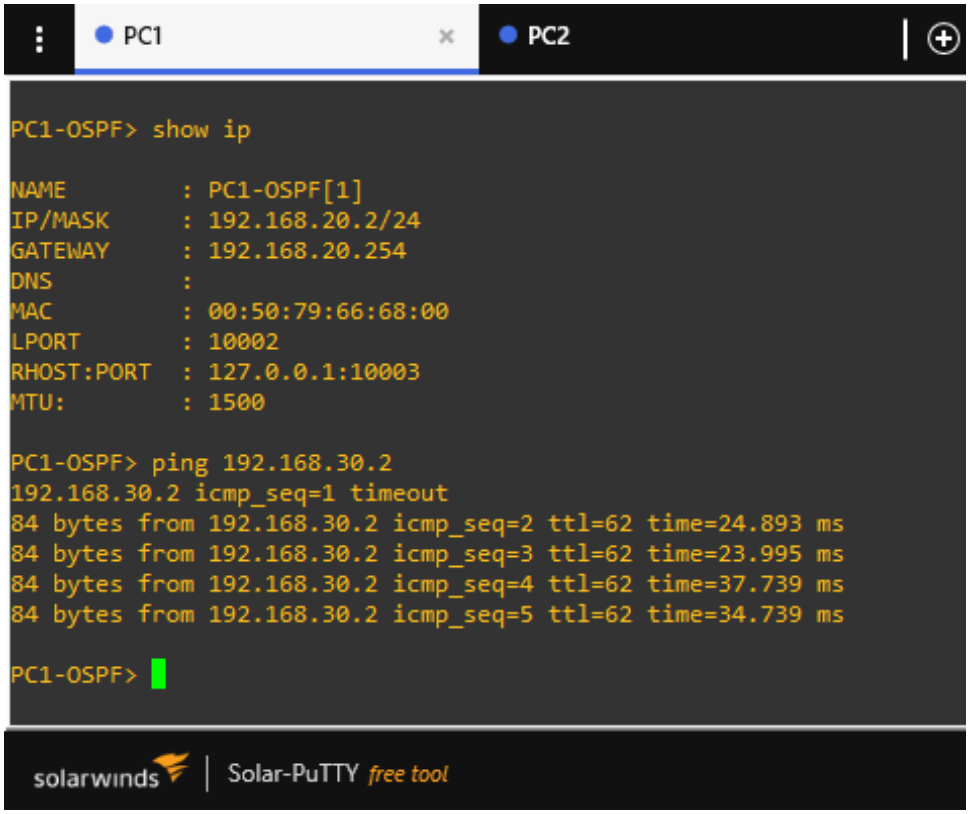

Figura 3.73 *Ping* desde PC1 hacia PC2 topología OSPF

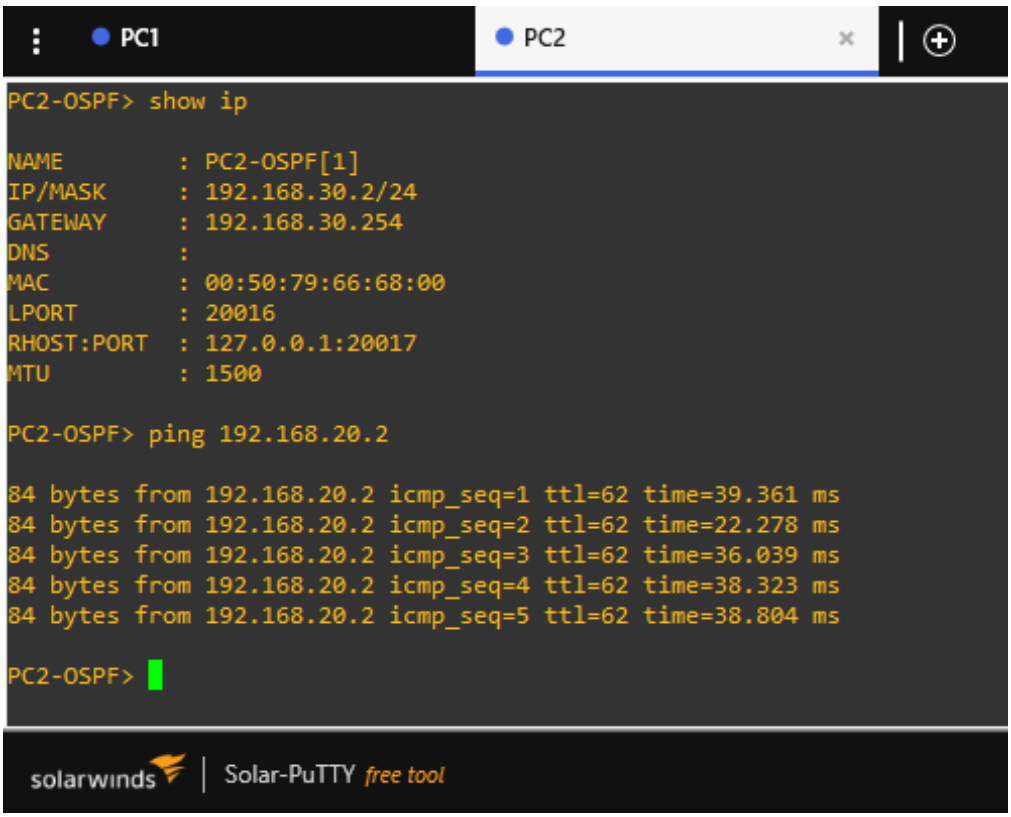

Figura 3.74 *Ping* desde PC2 hacia PC1 topología OSPF

4 CONCLUSIONES

- Para llevar a cabo la virtualización de máquinas se utilizó el *software* VirtualBox el cual posee una interfaz de usuario amigable para crear, configurar y ejecutar máquinas con requisitos específicos propios de un sistema operativo. Dentro de este hipervisor se instaló el sistema operativo Ubuntu *Server* 18.04 mismo que posee características adicionales como el OpenSSH, indispensable para la conexión entre nodo controlador y *host.*
- Ansible durante la instalación también instala Python, el cual es indispensable para la comunicación del nodo controlador y el *host* a través del protocolo SSH, ya que Ansible requiere de un intermediario adecuado de Python para cada *host* el cual se verifica en la primera conexión.
- Para crear la topología se utilizó el *software* GNS3, donde se agrega la máquina virtual del controlador desde un hipervisor. El adaptador de red de la máquina virtual fue establecido como red interna y GNS3 lo detecta como un adaptador de red virtual de tipo Intel PRO/1000 MT *Desktop* el cual se cambia a adaptador Intel PRO/1000 MT *Server*. Esto con el fin de tener mayor confiabilidad en la transmisión de datos y además que el nodo controlador sea un servidor.
- El inventario permite tener el control para gestionar equipos de la red con los que se trabaja, siendo un directorio muy importante debido que allí se guarda información de cada equipo o *host,* la creación debe ser ordenada, permitiendo agruparlos y definiendo rangos y variables para facilitar la ejecución del *playbook.*
- La creación del *playbook* se lo puede utilizar para configurar otro equipo cambiando solo información específica, se puede reutilizar la estructura del mismo para realizar configuraciones remotas a diferentes equipos haciendo la administración de red mucho más sencilla y simplificando las tareas redundantes.
- Con las pruebas de funcionamiento se verificó que las configuraciones establecidas en los *playbooks* ejecutaron cambios en los *routers.* A través de comandos de información se observó en cada uno de los *routers* las direcciones IPs y protocolos de enrutamiento establecidos para cada topología.
- La conectividad de la red se la demostró mediante un *ping,* realizado entre las dos redes LAN, la comunicación es satisfactoria por el correcto envío de paquetes de prueba.

5 RECOMENDACIONES

- Para realizar una simulación se debe tomar en cuenta los equipos a los que se les realizará la configuración, debido a que, si uno de los equipos no guarda los cambios realizados, como en el caso del *router* 3660 que no guarda la configuración de SSH, se debe cambiar por un *router* más estable como el Cisco 3725, el cual si guarda los cambios de configuración como SSH.
- En el archivo inventario es conveniente agregar el intérprete Python para que el *playbook* pueda realizar cambios dentro del sistema operativo de un equipo host, o también se lo puede agregar en el inicio del *playbook;* el agregado y ubicación del interprete queda a gusto del administrador.
- Antes de crear los *playbooks* es importante verificar los equipos de la topología con la cual se va a trabajar, resulta más eficiente descargar los módulos para equipos como Cisco, Fortinet, Juniper y entornos en la nube como Amazon aws, Azure, Google, entre otros.
- El comando *KexAlgoritms +diffie-hellman-group1-sha1* se agrega al final del archivo *ssh_config*, con el fin de establecer una conexión SSH entre el nodo controlador y los *hosts* sin errores, caso contrario el intercambio de claves no será posible, además el mismo SSH lo muestra como solución en caso de no estar agregado en el archivo de configuración.
- Al realizar los *playbooks* se debe tomar muy en cuenta los espacios, con la barra de espacio no tabulador, además de la jerarquía de cada línea de instrucción ya que al ejecutar el *playbook* la tarea no se realizará y quedará inconclusa la ejecución del *playbook*.

6 REFERENCIAS BIBLIOGRÁFICAS

[1] I. Lemus, «¿Qué es ANSIBLE?,» Conocimiento Libre, 05 Noviembre 2019. [En línea]. Available: https://conocimientolibre.mx/que-es-ansible/. [Último acceso: 25 mayo 2022].

[2] «Uso de Ansible para administrar maquinas remotas - Acervo Lima,» Acervolima.com, 2021. [En línea]. Available: https://es.acervolima.com/uso-de-ansible-para-administrar-maquinas-remotas/. [Último acceso: 07 Junio 2022].

[3] Keepcoding Tech School, «¿Qué es Ansible? Cómo iniciarte en esta herramienta DevOps,» Keepcoding, 31 Mayo 2021. [En línea]. Available: https://keepcoding.io/blog/que-es-ansible/. [Último acceso: 25 Mayo 2022].

[4] Ansible, «Installing Ansible - Ansible Documentation,» Ansible Project Contributors, 2021. [En línea]. Available: https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html?extIdCarryOver=true&sc_cid=701f2000001OH6fAAG#managed-node-requirements. [Último acceso: 25 Mayo 2022].

[5] DriveMeca, «Como instalar y configurar Ubuntu Server 18.04 LTS - DriveMeca,» GeneratePress, 28 Abril 2018. [En línea]. Available: https://www.drivemeca.com/como-ubuntu-server-18-04-lts/. [Último acceso: 07 Junio 2022].

[6] Red Hat Ansible, «Infrastructure Automation with Ansible,» Red Hat, Inc., 2020. [En línea]. Available: https://www.ansible.com/integrations/infrastructure?hsLang=en-us. [Último acceso: 07 Junio 2022].

[7] P. Kapgate, «Installing and Configuring Ansible on Ubuntu 20.04,» CloudSigma AG, 10 Enero 2022. [En línea]. Available: https://www.cloudsigma.com/installing-and-configuring-ansible-on-ubuntu-20-04/. [Último acceso: 07 Junio 2022].

[8] Ansible , «User Guide - Ansible Documentation,» Ansible Project Contributors, 2021. [En línea]. Available: https://docs.ansible.com/ansible/latest/user_guide/. [Último acceso: 07 Junio 2022].

[9] Red Hat, «What is Ansible?,» Red Hat, Inc., 15 Septiembre 2021. [En línea]. Available: https://www.redhat.com/en/technologies/management/ansible/what-is-ansible. [Último acceso: 02 Junio 2022].

[10] Ansible, «Intro to playbooks - Ansible Documentation,» Ansible Project Contributors, 2022. [En línea]. Available: https://docs.ansible.com/ansible/latest/user_guide/playbooks_intro.html#about-playbooks. [Último acceso: 02 Junio 2022].

[11] Canonical Ltd., «Packages in “ansible-2.9”,» “Ansible” team, 11 Octubre 2021. [En línea]. Available: https://launchpad.net/~ansible/+archive/ubuntu/ansible-2.9/+packages. [Último acceso: 08 Junio 2022].

[12] Mundowin, «5 mejores simuladores de red para imitar una red de ordenadores en vivo en un PC - Mundowin,» Mundowin, 22 Agosto 2019. [En línea]. Available: https://mundowin.com/5-mejores-simuladores-de-red-para-imitar-una-red-de-ordenadores-en-vivo-en-un-pc/. [Último acceso: 02 Junio 2022].

[13] Galaxy Technologies LLC, «Getting Started with GNS3 - GNS3 Documentation,» Galaxy Technologies LLC, 2012. [En línea]. Available: https://docs.gns3.com/docs/. [Último acceso: 02 Junio 2022].

[14] J. Jimenez, «Simuladores para virtualizar redes y aprender routing y switching,» RedesZone.net, 11 Marzo 2022. [En línea]. Available: https://www.redeszone.net/tutoriales/redes-cable/programas-simular-red/. [Último acceso: 08 Junio 2022].

[15] C. Cabrera, «Qué es EVE-NG? el «nuevo» emulador para redes - Info++,» Tema Astra para WordPress, 06 Mayo 2018. [En línea]. Available: https://cesarcabrera.info/que-es-eve-ng-un-nuevo-emulador-para-redes/. [Último acceso: 09 Junio 2022].

[16] Ronald F. Clayton, «Simuladores de Red,» VSIP.INFO, 2019. [En línea]. Available: https://vsip.info/simuladores-de-red-pdf-free.html. [Último acceso: 10 Julio 2022].

[17] Galaxy Technologies LLC., «“GNS3 Windows Install | GNS3 Documentation,”,» Galaxy Technologies LLC., 2012. [En línea]. Available: https://docs.gns3.com/docs/getting-started/installation/windows/. [Último acceso: 10 Julio 2022].

[18] EVE-NG Ltd, «EVE - The Emulated Virtual Environment For Network, Security and DevOps Professionals,» EVE-NG Ltd, 2022. [En línea]. Available: https://www.eve-ng.net/. [Último acceso: 09 Junio 2022].

[19] U. Dzerkals, M. Doe y C. Lim, «EVE-NG | EVE-NG Community CookBook Version 5.1 | EVE-NG documentation,» 24 Mayo 2022. [En línea]. Available: https://www.eve-ng.net/index.php/documentation/community-cookbook/. [Último acceso: 10 Julio 2022].

7 ANEXOS

ANEXO I: Certificado de Originalidad

CERTIFICADO DE ORIGINALIDAD

Quito, D.M. 29 de agosto de 2022

De mi consideración:

Yo, GABRIELA KATHERINE CEVALLOS SALAZAR, en calidad de Director del Trabajo de Integración Curricular titulado IMPLEMENTAR ANSIBLE PARA HABILITAR PROTOCOLOS DE ENRUTAMIENTO DINÁMICO IPV4 elaborado por el estudiante JIMMY ADRIAN SARANGO VACA de la carrera en TECNÓLOGÍA SUPERIOR EN REDES Y TELECOMUNICACIONES, certifico que he empleado la herramienta Turnitin para la revisión de originalidad del documento escrito completo, producto del Trabajo de Integración Curricular indicado.

El documento escrito tiene un índice de similitud del 12%.

Es todo cuanto puedo certificar en honor a la verdad, pudiendo el interesado hacer uso del presente documento para los trámites de titulación.

NOTA: Se adjunta el link del informe generado por la herramienta Turnitin.

https://epnecuador-my.sharepoint.com/:f:/g/personal/gabriela_cevalloss_epn_edu_ec/Eoxt2S_vvP9Nv7583DQj030Bc8noOP9DfFazJy99rVZbzQ?e=V1ivLs

Atentamente,

Gabriela Cevallos

Docente

Escuela de Formación de Tecnólogos

ANEXO II: ENLACES

Enlace YouTube: https://youtu.be/v_KQo45-pBY

Anexo II.I Código QR de la implementación y pruebas de funcionamiento

ANEXO III: Códigos Fuente

Playbook RIP *router* 1 (ripv2Router1)

```
---
# Enrrutamiento Vector Distancia ripV2

- hosts: r1vd
  gather_facts: false
  remote_user: admin
  tasks:
    - name: Configuracion de Interfaces
      cisco.ios.ios_l3_interfaces:
        config:
        - name: FastEthernet0/1
          ipv4:
          - address: 192.168.20.254/24
        - name: Serial1/0
          ipv4:
          - address: 10.0.0.1/30
        state: merged

    - name: Habilitacion de Interfaces
      cisco.ios.ios_interfaces:
        config:
        - name: FastEthernet0/1
          enabled: false
          description: Configurado con Ansible J.S
        - name: Serial1/0
          enabled: false
          description: Configurado con ansible J.S conexion de serial
        state: merged

    - name: Clock Rate
      cisco.ios.ios_config:
        lines:
        - interface Serial 1/0
        - clock rate 64000
        - do wr
        save_when: always

    - name: Borrado de Protocolo
      cisco.ios.ios_config:
        lines:
        - no router rip
        - do wr
        save_when: always

    - name: Enrutamiento ripV2
      cisco.ios.ios_config:
        lines:
        - router rip
        - version 2
        - no auto-summary
        - network 10.0.0.0
```

```
    - network 192.168.20.0
    - do wr
    save_when: always
```

Playbook RIP *router* 2 (ripv2Router2)

```
---
# Enrrutamiento Vector Distancia ripV2

- hosts: r2vd
  gather_facts: false
  remote_user: admin
  tasks:
    - name: Configuracion de Interfaces
      cisco.ios.ios_l3_interfaces:
        config:
        - name: FastEthernet0/1
          ipv4:
          - address: 192.168.30.254/24
        - name: Serial1/0
          ipv4:
          - address: 10.0.0.2/30
        state: merged

    - name: Habilitacion de Interfaces
      cisco.ios.ios_interfaces:
        config:
        - name: FastEthernet0/1
          enabled: false
          description: Configurado con Ansible J.S
        - name: Serial1/0
          enabled: false
          description: Configurado con ansible J.S conexion de serial
        state: merged

    - name: Borrado de Protocolo
      cisco.ios.ios_config:
        lines:
        - no router rip
        - do wr
        save_when: always

    - name: Enrutamiento ripV2
      cisco.ios.ios_config:
        lines:
        - router rip
        - version 2
        - no auto-summary
        - network 10.0.0.0
        - network 192.168.30.0
        - do wr
        save_when: always
```

Playbook OSPF *router* 1 (ospfRouter1)

```
---
#Enrutamiento Link State OSPF

- hosts: r1ls
  gather_facts: false
  remote_user: admin
  tasks:
    - name: Configuracion de Interfaces
      cisco.ios.ios_l3_interfaces:
        config:
        - name: FastEthernet0/1
          ipv4:
          - address: 192.168.20.254/24
        - name: Serial1/0
          ipv4:
          - address: 10.0.0.1/30
        state: merged

    - name: Habilitacion de Interfaces
      cisco.ios.ios_interfaces:
        config:
        - name: FastEthernet0/1
          enabled: false
          description: Configurado con Ansible J.S, interfaz LAN
        - name: Serial1/0
          enabled: false
          description: Configurado con Ansible J.S interfaz de comunicacion
routers

    - name: Clock rate
      cisco.ios.ios_config:
        lines:
        - interface Serial 1/0
        - clock rate 64000
        - do wr
        save_when: always

    - name: Delet Protocol
      cisco.ios.ios_ospfv2:
        state: deleted

    - name: Enrutamiento OSPF
      cisco.ios.ios_config:
        lines:
        - router ospf 1
        - network 192.168.20.0 0.0.0.255 area 0
        - network 10.0.0.0 0.0.0.3 area 0
        - do wr
        save_when: always
```

Playbook OSPF *router* 2 (ospfRouter2)

```
---
#Enrutamiento Link State OSPF

- hosts: r2ls
  gather_facts: false
  remote_user: admin
  tasks:
    - name: Configuracion de Interfaces
      cisco.ios.ios_l3_interfaces:
        config:
        - name: FastEthernet0/1
          ipv4:
          - address: 192.168.30.254/24
        - name: Serial1/0
          ipv4:
          - address: 10.0.0.2/30
        state: merged

    - name: Habilitacion de Interfaces
      cisco.ios.ios_interfaces:
        config:
        - name: FastEthernet0/1
          enabled: false
          description: Configurado con Ansible J.S, interfaz LAN
        - name: Serial1/0
          enabled: false
          description: Configurado con Ansible J.S interfaz de comunicacion
routers

    - name: Delet Protocol
      cisco.ios.ios_ospfv2:
        state: deleted

    - name: Enrutamiento OSPF
      cisco.ios.ios_config:
        lines:
        - router ospf 1
        - network 192.168.30.0 0.0.0.255 area 0
        - network 10.0.0.0 0.0.0.3 area 0
        - do wr
        save_when: always
```