ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

IMPLEMENTACIÓN DE SERVICIOS DE RED CON HERRAMIENTAS DE DEVOPS

DESPLIEGUE DE VRF MEDIANTE DEVOPS

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

CHRISTIAN DAVID ÑACATO ANDRANGO

DIRECTOR: FERNANDO VINICIO BECERRA CAMACHO

DMQ, MARZO 2023

CERTIFICACIONES

Yo, CHRISTIAN DAVID ÑACATO ANDRANGO 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.

Christian David Ñacato Andrango

Christian.nacato01@epn.edu.ec

Christian.nacato01@gmail.com

Certifico que el presente trabajo de integración curricular fue desarrollado por CHRISTIAN DAVID ÑACATO ANDRANGO, bajo mi supervisión.

Fernando Vinicio Becerra CamachoDIRECTOR

Fernando.becerra@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.

CHRISTIAN DAVID ÑACATO ANDRANGO

DEDICATORIA

El presente trabajo de titulación, le dedico a mi familia por todo el apoyo brindado en esta etapa de mi vida. En especial a mis padres Manuel y Susana que son el motor de mi vida, gracias a ellos he logrado esta meta propuesta.

A mis abuelitos, que fueron como mis segundos padres, que me enseñaron valores para la vida. En especial me enseñaron a perseverar y luchar para lograr las metas propuestas en la vida.

A mi hermana Alexandra, por apoyarme y creer en mí, a pesar de la distancia.

A mi hermana Marlene, por apoyarme en momentos difíciles que he pasado a lo largo de la vida.

A mi hermano Luis, por siempre contar con él cuando lo he necesitado.

Madre, déjame decirte que te admiro y te amo, por nunca rendirte y ser un ejemplo para nosotros tus hijos.

Christian D. Ñacato A.

AGRADECIMIENTO

Agradezco a Dios por darme la oportunidad de lograr culminar la carrera universitaria y sobre todo la valentía para llegar a culminar con una de las metas propuestas en mi vida. También agradezco por contar con mis seres queridos, en los buenos y malos momentos.

A mi familia, por la comprensión y sobre todo por la tolerancia, que tuvieron para apoyarme a lo largo de mi carrera universitaria.

A la ESFOT, por las enseñanzas que he adquirido a lo largo de la carrera y sobre todo por las experiencias que me enseñaron a formar mi carácter como profesional y sobre todo como persona.

A las personas que hicieron lo posible por lograr esta meta propuesta en la vida.

Christian D. Ñacato A.

ÍNDICE DE CONTENIDOS

CERTIFICACIONES I

Fernando Vinicio Becerra CamachoDIRECTOR I

DECLARACIÓN DE AUTORÍA II

DEDICATORIA III

AGRADECIMIENTO IV

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO 9

1.1 Objetivo general 9

1.2 Objetivos específicos 9

1.3 Alcance 9

1.4 Marco Teórico 9

DevOps 9

Ansible 10

Inventario 10

Playbook 10

Nodo de Control 11

GNS3 11

VRF 11

2 METODOLOGÍA 12

3 RESULTADOS 13

3.1 Analizar el DevOps y sus características 14

3.2 Diseñar el algoritmo para implementar VRF 14

Configuraciones de VRF 14

Configuraciones de EIGRP 15

Configuraciones de OSPF 15

Configuraciones de Cisco NAT 16

3.3 Implementar el algoritmo con VRF mediante la herramienta de DevOps 17

Instalación del nodo de control 17

Implementación de la topología 21
Conexión SSH y configuración de claves 22
Configuración de SSH 23
Generación de llaves SSH en Ansible 24
Construcción del Inventario 26
Creación del primer *playbook* 28
Creación del segundo *playbook* 33
Creación del tercer *playbook* 36
Creación del cuarto *playbook* 39
3.4 Verificar el funcionamiento del programa 44
Ejecución del primer *playbook* 44
Ejecución del segundo *playbook* 46
Ejecución del tercer *playbook* 48
Ejecución del cuarto *playbook* 50
Revisión de las tablas de enrutamiento 52
Pruebas de conectividad 55
4 CONCLUSIONES 57
5 RECOMENDACIONES 58
6 Referencias bibliográficas 59
7 ANEXOS 62
ANEXO I: Certificado de Originalidad i
ANEXO II: Enlaces ii
ANEXO III: Códigos Fuente iii
Inventario (hosts-vrf) iii
Primer *playbook* (Router1VRF) iv
Segundo *playbook* (Router2VRF) vii
Tercer *playbook* (Router3VRF) x
Cuarto *playbook* (Router4VRF) xv

RESUMEN

El presente proyecto de trabajo de integración curricular, DESPLIEGUE DE VRF MEDIANTE DEVOPS, se implementa un servicio de red mediante la tecnología *Virtual Routing and Forwarding*, el cual es desplegado por Ansible como una de las herramientas de DevOps. Ansible permite la configuración dinámica de los equipos de *networking*, a través de un nodo de control que se comunica mediante el protocolo SSH, a los nodos administrados.

La sección uno, describen los objetivos del proyecto de titulación, como también se muestra el alcance y el marco teórico que se realizó para implementar la tecnología VRF mediante una de las herramientas de DevOps.

La sección dos, se presenta la metodología utilizada para implementar la tecnología VRF. Además, expone los pasos a seguir para la instalación de la herramienta Ansible para la configuración dinámica de los equipos de *networking*.

La sección tres, presenta los resultados, donde se detalla los pasos para realizar las configuraciones de VRF, protocolos de enrutamiento y Cisco NAT. También detalla los pasos para la instalación del nodo de control, establecer la comunicación remota, crear el inventario de los hosts, realizar los *playbooks* para cada equipo de *networking* y verificar el funcionamiento de la simulación.

En la última sección, corresponde a las conclusiones y recomendaciones obtenidas durante el desarrollo del proyecto de titulación.

PALABRAS CLAVE: VRF, DevOps, Ansible, Inventario, Libro de jugadas y Networking.

ABSTRACT

The present project of curricular integration work, DEVELOPMENT OF VRF THROUGH DEVOPS, implements a network service through Virtual Routing and Forwarding technology, which is deployed by Ansible as one of the DevOps tools. Ansible allows the dynamic configuration of networking equipment through a control node that communicates via SSH protocol to the managed nodes.

Section one describes the objectives of the degree project, as well as the scope and the theoretical framework that was performed to implement VRF technology through one of the DevOps tools.

Section two presents the methodology used to implement VRF technology. In addition, it exposes the steps to follow for the installation of the Ansible tool for the dynamic configuration of the networking equipment.

Section three presents the results, detailing the steps to perform the VRF configurations, routing protocols and Cisco NAT. It also details the steps to install the control node, establish remote communication, create the hosts inventory, create the playbooks for each networking equipment and verify the operation of the simulation.

The last section corresponds to the conclusions and recommendations obtained during the development of the degree project.

KEYWORDS: *VRF, DevOps, Ansible, Inventory, Playbook y Networking.*

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

El presente proyecto de titulación se desea implementar el servicio de red, el cual será desplegado mediante DevOps. La herramienta de DevOps permitirá la configuración dinámica del servicio de red en cada elemento de *networking* además se realizarán pruebas de funcionamiento mediante simuladores de red.

1.1 Objetivo general

Implementar la tecnología VRF por medio de una herramienta de DevOps.

1.2 Objetivos específicos

- Analizar el DevOps y sus características.
- Diseñar el algoritmo para implementar VRF.
- Implementar el algoritmo con VRF mediante herramientas de DevOps.
- Verificar el funcionamiento del programa.

1.3 Alcance

En el presente proyecto permite el manejo de una herramienta de DevOps y aplicarlo a los equipos de *networking*, para ello se va a necesitar virtualizar los elementos de red y para poderlos simularlos. En este proyecto se van a implementar el servicio de red VRF mediante una herramienta de DevOps para automatizar su despliegue.

1.4 Marco Teórico

DevOps

Actualmente, DevOps es desplegada por empresas de desarrollo y despliegue de software, lo cual involucra a profesionales de TI y usuarios, con el propósito de tener una forma eficiente y estratégica de trabajar frente a la brecha que existe entre los procesos de desarrollo y operaciones [1].

Para poder definir DevOps es complicado, ya que se encuentran diferentes definiciones en Internet, como por ejemplo la de Google Cloud, que la define como un movimiento cultural, con intención de acelerar la comercialización del software, a fin de mejorar la fiabilidad del servicio [2]. Por otra parte, Microsoft define como la unión entre el desarrollo de software y las operaciones de TI, con el objetivo de acoplar a diferentes profesionales de desarrollo de software, TI, procesos, tecnologías y seguridad, a lo cual conlleva adoptar una cultura de conocimientos y herramientas ofrecidas por DevOps,

aumentando la confianza en las aplicaciones, por parte de los usuarios y la velocidad de los despliegues de las soluciones [3].

Ansible

Una de las principales herramientas de gestión de la configuración y administración, que aligera las iniciativas de DevOps es Ansible. Ansible es una herramienta de automatización de redes, que permite la interconectividad de equipos de red. Además, la herramienta Ansible es de código abierto, desarrollado en el lenguaje de programación Python, que administra a los dispositivos de red mediante la conexión remota, por medio del protocolo *Secure Shell* (SSH). Ansible es utilizado para administrar, servidores, sistemas en la nube, equipos de *networking y* entre otros, permitiendo la gestión y configuración de dichos equipos. Con esto se obtiene una mejor eficiencia del grupo de TI, eliminando la ejecución de labores repetitivas mediante plantillas escritas en el lenguaje de serialización de datos llamado *Yet Another Markup Language* (YAML) [4] [5].

Ansible cuenta con módulos integrados, que se utilizan para la automatización de procesos repetitivos o se puede realizar configuraciones especiales instalando nuevos módulos, siempre y cuando se establezcan en formato *JavaScript Object Notation* (JSON), para que estos archivos sean compatibles con el lenguaje YAML [6].

Inventario

El inventario es un archivo que contiene y describe tanto a los hosts como a los grupos de hosts que serán administrados, gestionado y configurado por Ansible. El inventario enumera los nombres o alias de cada host como también las direcciones IP de cada uno de estos. Ansible utiliza el inventario para poder administrar y saber cómo conectarse con cada uno de los hosts que lo conforman. Por lo general, el archivo del inventario se encuentra en la ruta */etc/ansible/hosts*, pero también se puede crear un nuevo inventario y especificar la dirección del nuevo inventario con la opción -i, al momento de ejecutar el comando *ansible-playbook* [7].

Playbook

Los *playbooks* o libros de jugadas, es un archivo en formato YAML que contiene una lista de una o más jugadas. Cada jugada en un *playbook* es asignada a un conjunto de tareas que serán ejecutas en los hosts que se encuentran dentro del inventario. Estas tareas se pueden definir mediante módulos proporcionados por Ansible, que son pequeños programas previamente escritos, a fin de realizar tareas específicas, como son: configurar, instalar software y copiar archivos [8] [9].

Nodo de Control

El nodo de control es el equipo en el cual se instala y ejecuta la herramienta Ansible, para lo cual es necesario contar con las siguientes características: que posea un sistema operativo equivalente a UNIX y a la vez que cuente con una versión instalada igual o superior a Python 3.9. Los sistemas operativos que cuentan con estas características son: Ubuntu 20.04 LTS, Debian 11.5, Red Hat 9.1, macOS 13.0 y BSD 13.1. Mientras que para el sistema operativo Windows es necesario contar con un subsistema de Windows, que permita ejecutar un entorno GNU/Linux [10] [11] [12].

GNS3

Graphic Network Simulator-3 (GNS3) es un simulador gráfico de *networking* gratuito y sobre todo de código abierto, utilizado por miles de profesionales de redes con diferentes propósitos como emular, configurar, experimentar y brindar soluciones tanto a redes virtuales como reales. GNS3 permite simular pequeñas y grandes topologías, ya que se puede agregar equipos de red básicos como: máquinas virtuales, switches, routers y servidores. Además, GNS3 admite equipos de red Cisco, como también se puede trabajar con más equipos comerciales, por ejemplo, Huawei y Juniper [13].

GNS3 se puede instalar en varios sistemas operativos como: Windows, Windows server, Mac Os X Mavericks y Linux. Una limitación que tiene GNS3 es el *Hardware*, por cuál el simulador dependerá de ciertas características mínimas de CPU y memoria, como poseer un procesador de dos o más núcleos, memoria RAN de 4 (GB) o más y en su instalación disponer de al menos 1 (GB) de almacenamiento [14].

VRF

Virtual Routing and Forwarding (VRF) es una tecnología de redes IP que poseen los equipos de *networking* como: routers y switches de capa 3, que permiten crear varias redes virtuales, para tener múltiples instancias de la tabla de enrutamiento y estas puedan coexistir de forma simultánea dentro de un mismo *Hardware*, a fin de aislar el tráfico de diferentes clientes como se visualiza en la **Figura 1.1**. Las instancias son independientes, ya que se puede asignar una VRF a una o varias interfaces de red, pero ninguna de las VRFs comparten rutas entre sí. Los paquetes solo se envían a otras interfaces que fueron asignadas a la misma VRF, porque cada red virtual contiene su propia tabla de rutas. Por lo tanto, es posible asignar direcciones IP superpuestas o iguales, sin que estas entren en conflicto entre sí, a las diferentes interfaces de red [15] [16] [17].

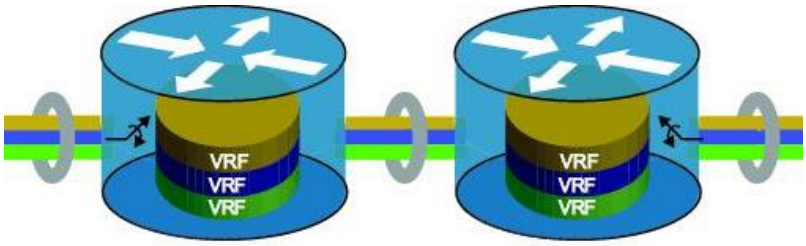

Figura 1.1 Ejemplo de diferentes instancias de la tabla de enrutamiento [17].

Existen dos modos de configuración de una VRF que son: Full VRF y VRF-Lite.

- FULL VRF es su forma completa, necesariamente requiere del transporte de datos mediante *Multiprotocolo Label Switching* (MPLS), para etiquetar el tráfico correspondiente a la capa 3 del modelo OSI. Esta configuración comúnmente es implementada por los ISP o proveedores de servicio de Internet [18].
- VRF-Lite es un subconjunto de Full VRF, por el hecho de que no trabaja con MPLS. VRF-Lite por lo general es implementado en entornos empresariales [18].

2 METODOLOGÍA

El presente proyecto de trabajo de integración curricular se llevó a cabo por medio de la investigación exploratoria. En primera instancia se realizó la indagación de la información acerca de la definición, funcionamiento e implantación de DevOps, Ansible y VRF. Con la investigación exploratoria, referente a DevOps se define como una cultura que conlleva a prácticas y el uso de herramientas que permitan automatizar los procesos de una organización.

Ansible permite realizar la gestión y configuración de los equipos de *networking* que conforman la topología de red, con la finalidad de beneficiarse de la automatización de las redes. En este proyecto de titulación se utilizó Ansible como herramienta de DevOps para configurar de forma automática los routers con la tecnología *Virtual Routing and Forwarding.* La herramienta Ansible permitirá agilizar cambios en la gestión y configuración de los equipos de red, minimizando los errores producidos por los humanos al momento de ejecutar tareas repetitivas.

Mientras que para la configuración de la tecnología VRF en los enrutadores se creara dos instancias, es decir, una tabla de rutas para cada VRF. Cada una de las tablas de las VRFs son independientes de la tabla de enrutamiento global del router, con el propósito de aislar el tráfico de los clientes por medio de las instancias creadas, ya que cuentan con una tabla de rutas exclusiva para cada grupo de clientes.

Una vez realizada la investigación de los tres puntos anteriormente mencionados y cumplir con el objetivo general del proyecto, se procedió a Instalar el simulador gráfico de red GNS3, por medio del software de virtualización VirtualBox, para montar la máquina virtual de GNS3. Una vez instalado el software de simulación se agregó los equipos de *networking* a utilizar en la topología de red.

Para agregar el enrutador, se procedió a descargar la imagen IOS del equipo Cisco c3725, mientras tanto para el nodo de control se descargó directamente de la página oficial de GNS3, el dispositivo *Network Automation,* el cual contiene la herramienta Ansible para la automatización de *networking*.

Posteriormente, se realizó la comunicación remota entre el nodo de control y los enrutadores, para lo cual se configuró dentro de cada router la comunicación SSH. Además, dentro de Ansible se generó y copio las llaves SSH, para establecer la conexión remota.

Como penúltimo paso se procedió a crear el inventario de los hosts que conforman la topología de red. Además, se creó cada *playbook*, para la configuración dinámica de los equipos de *networking*. Los playbooks contiene configuración como: ingresar el nombre del equipo, crear una o dos VRFs, asignar la dirección IP a cada interfaz de red y habilitar los protocolos de enrutamiento. Para realizar las configuraciones en los equipos de *networking*, el usuario ingresa información como: nombre del equipo, nombre de la VRF, distintivo de ruta, asignación de VRF y dirección IP a las diferentes interfaces de red.

Finalmente, se realiza pruebas de funcionamiento, ejecutando los *playbooks, para* verificar los cambios efectuados en la configuración de cada router, mediante la herramienta Ansible. Además, se procedió a la revisión de las tablas de rutas correspondientes a cada instancia de las VRFs, así como también se verificó el estado de la conexión de los hosts locales, mediante pruebas de conectividad.

3 RESULTADOS

En la presente sección se despliega, los pasos a seguir para la implementación de la tecnología *Virtual Routing and Forwarding* mediante una herramienta de DevOps. En primera instancia se procedió con la instalación del nodo de control que contiene la herramienta Ansible, con la finalidad de establecer la comunicación remota mediante el protocolo SSH con cada uno de los routers.

Una vez realizada la comunicación remota con cada router, se procedió a crear el inventario de los hosts. Además, se creó un *playbook* para cada router, a fin de realizar

configuraciones dinámicas para dichos equipos. Las tareas que contienen los *playbooks* son las siguientes: modificar el nombre del equipo, configuración de las interfaces de red, creación de dos instancias de enrutamiento en toda la red, asignación de la instancia, configuración de los protocolos dinámicos EIGRP y OSPF. Finalmente, se agrega la configuración de NAT a los equipos Cisco.

3.1 Analizar el DevOps y sus características

DevOps, se basa en una metodología de desarrollo de software que tiene el propósito de mejorar la colaboración y comunicación entre los diferentes profesionales de desarrollo y operaciones de TI. Con la finalidad de entregar el software de forma rápida y eficiente [19]. Algunas características claves de DevOps son las siguientes:

- Requiere adoptar un cambio de cultura a base de la colaboración y comunicación entre profesionales de desarrollo y operaciones de TI, dentro de una organización.
- Crear un ciclo de vida para el desarrollo de software más rápido y eficiente.
- Adquirir la práctica de administrar la infraestructura del software.
- Recopilar información de los sistemas, para identificar problemas y mejorar el rendimiento.
- Implementar automáticamente los cambios en un código, mediante la automatización.
- Uso de herramientas y tecnologías, para respaldar las diferentes prácticas y procesos de la metodología de DevOps.

La información adicional se encuentra desarrollado en el marco teórico del trabajo de titulación.

3.2 Diseñar el algoritmo para implementar VRF

Para iniciar el diseño del algoritmo de la tecnología de VRF, se realizó la indagación de los comandos Cisco de las siguientes configuraciones como: VRF Lite, protocolos de enrutamiento dinámicos tal como OSPF y EIGRP, además la traducción de direcciones de red (NAT).

Configuraciones de VRF

En la **Tabla 3.1**, se puede visualizar los comandos básicos utilizados, para la configuración de VRF Lite en un router Cisco.

Tabla 3.1 Comandos para configuración VRF Lite [20].

Comando	Descripción
ip vrf *name-vrf*	Ingresa a las configuraciones de la VRF y asigna el nombre a la VRF.
rd *ruta-distinguir*	Crea la tabla de VRF y agrega el distintivo de ruta. Este distintivo puede ser un número arbitrario o dirección IP.
interfaz *id-interfaz*	Ingresar al modo de configuraciones de la interfaz de red.
ip vrf forwarding vrf-name	Relaciona una VRF a una Interfaz de red.
show ip route vrf	Muestra la tabla de rutas de la VRF.

Configuraciones de EIGRP

Para habilitar el protocolo EIGRP en router Cisco, se hace uso de los siguientes comandos, que se muestra en la **Tabla 3.2**

Tabla 3.2 Comandos del protocolo EIGRP [21].

Comando	Descripción
router eigrp *autonomous-system-number*	Habilitar enrutamiento EIGRP.
address-family ipv4 vrf *vrf-name*	Ingresar al modo de configuraciones de la familia de direcciones para una instancia.
no auto-summary	Obtener rutas resumidas para otros routers EIGRP.
network *network-number*	Ingresar dirección de red para el enrutamiento EIGRP.
autonomous-system *autonomous-system-number*	Ingresar el número del sistema autónomo.

Configuraciones de OSPF

Para habilitar el protocolo OSPF en equipos Cisco, se utiliza los siguientes comandos que se muestra en la **Tabla 3.3**

Tabla 3.3 Comandos para el protocolo OSPF [22].

Comando	Descripción
router ospf *process-id* ***vrf*** *vrf-name*	Habilitar el enrutamiento OSPF y especificar una tabla de rutas de VRF.
capability vrf-lite	Suprimir comprobaciones del router proveedor perimetral.
router-id *id-value*	Configura el router ID.
network *network-number* ***area*** *area-id*	Ingreso de dirección de red e ID del área de OSPF.

Configuraciones de Cisco NAT

Tabla 3.4 En la **Tabla 3.4** se muestra los comandos de Cisco, para la configuración de la traducción de direcciones de red o sus siglas NAT.

Tabla 3.4 Comandos para la configuración Cisco NAT [23].

Comando	Descripción
Interface loopback *number*	Habilitar y asignar una interfaz de loopback.
ip vrf forwarding *vrf-name*	Asignar VRF a una interfaz.
ip address	Ingresar dirección IP y mascara.
ip nat inside	Traducción de direcciones IP desde el interior hacia el exterior.
interface serial	Ingresar al modo de configuraciones de la interfaz.
ip nat outside	Traducción de direcciones IP desde el exterior hacia el interior.
access-list *IP extended access list* ***permit ip host*** *Source address* ***any***	Determinar los permisos de la lista de control de acceso y a quien se le permite el tráfico.
ip nat inside source list *number* ***interface serial*** *interface number* ***vrf*** *vrf name* ***overload***	Establece una traducción del tráfico local al dispositivo de borde que ejecuta NAT.

A continuación, se describe los pasos para diseñar el algoritmo de VRF:

- Se instalará un software de simulación de redes.
- Se descargará las imágenes ISO de los equipos de networking.
- Se instalará un nodo de control que contendrá la herramienta Ansible, para la configuración de los equipos de *networking.*
- Se creará una topología de red con cuatro routers Cisco c3725, dos VPCS y dos Ethernet Switches.
- Se realizará la configuración de la comunicación SSH en los routers.
- Se realizará la conexión remota entre el nodo de control y routers.
- Se creará dos instancias de la tabla de enrutamiento, mediante los comandos para la configuración de VRF.
- Se habilitará dos protocolos de enrutamiento dinámico en toda la red mediante los de EIGRP y OSPF.
- Se agregará la configuración de Cisco NAT mediante los comandos investigados anteriormente.

Ya que se tiene claro el panorama de qué se va a implementar, se procede a crear la topología de red, la cual contará con el servicio de VRF. Todos estos pasos se detallarán en la implementación del algoritmo con VRF mediante la herramienta de DevOps.

3.3 Implementar el algoritmo con VRF mediante la herramienta de DevOps

Para implementar el algoritmo de VRF, es necesario realizar los siguientes pasos: instalación de la herramienta Ansible, implementación de la topología de red para VRF, establecer la comunicación remota entre un nodo de control y los equipos de *networking* para realizar una comunicación SSH en cada uno de los routers.

Instalación del nodo de control

Para utilizar la herramienta de Ansible en el software de simulación GNS3, se requiere agregar un dispositivo preconfigurado. Se ingresa a la página oficial de GNS3 en la pestaña *Marketplace* opción *Appliances,* para descarga el archivo que contiene el dispositivo *Network Automation* como se puede apreciar en la **Figura 3.1**. El dispositivo contiene herramientas para la automatización de redes.

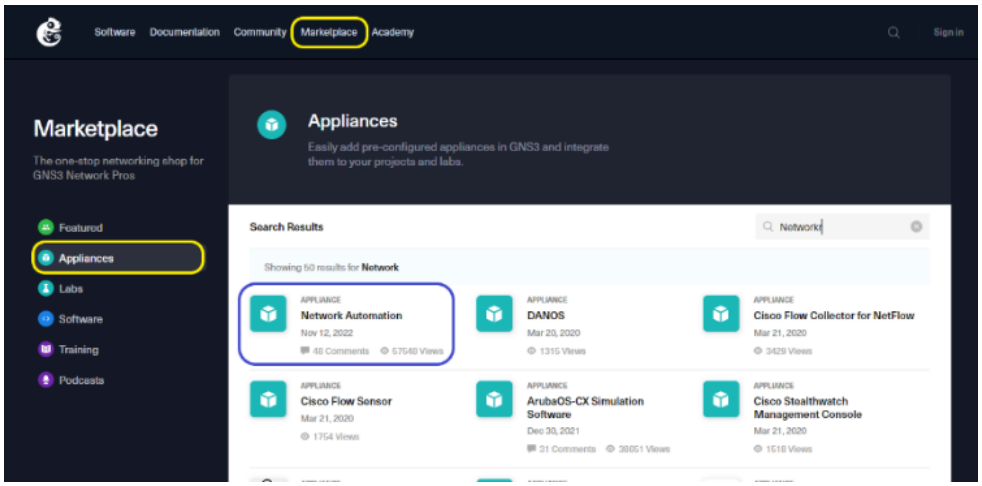

Figura 3.1 Sitio web de descarga de los *Appliances*.

Una vez descargado el archivo *Network Automation*, en la interfaz gráfica del usuario de GNS3, se dirige a la pestaña *File*, sección *Import appliance*, como se puede apreciar en la **Figura 3.2**.

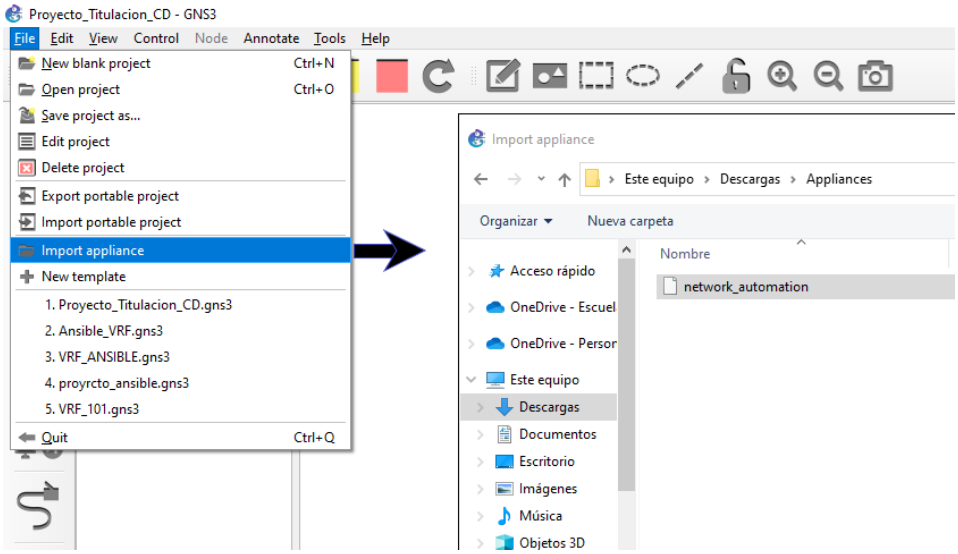

Figura 3.2 Interfaz gráfica de GNS3 para agregar *Appliance.*

Se despliega la ventana *Server*, donde se presenta una única opción para instalar el dispositivo *Network Automation*. Este dispositivo requiere la máquina virtual GNS3 (GNS3 VM) para ser ejecutado, debido a lo cual las otras opciones no se pueden seleccionar, ya que se encuentran atenuadas como se muestra en la **Figura 3.3**.

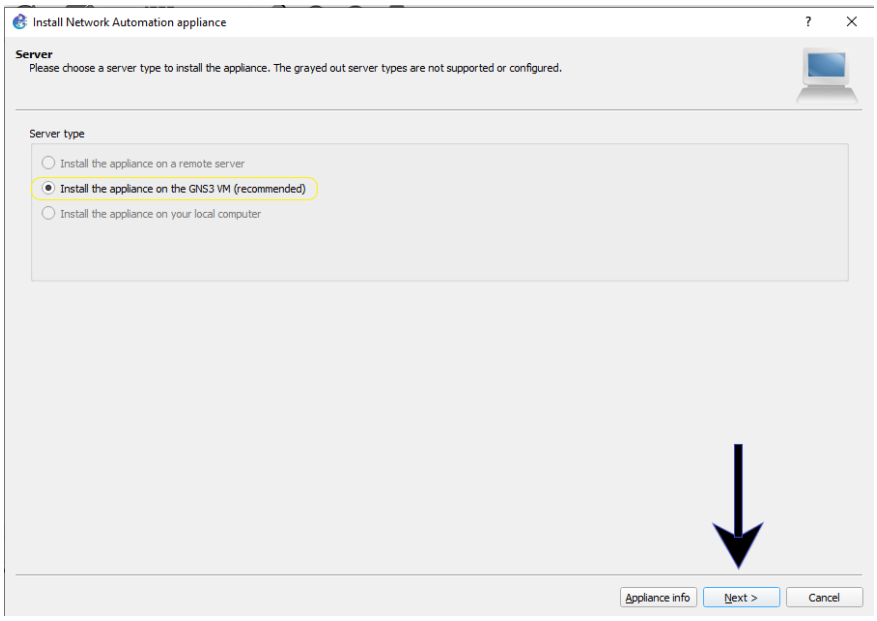

Figura 3.3 Ventana *Server* para instalar el *Appliance*.

Finalmente, se muestra la ventana *Usage*, para completar la instalación del dispositivo *Network Automation,* se selecciona la opción *Finish*. El dispositivo se agrega correctamente a GNS3 y muestra una ventana con el mensaje de instalación exitosa, como se visualiza en la **Figura 3.4**.

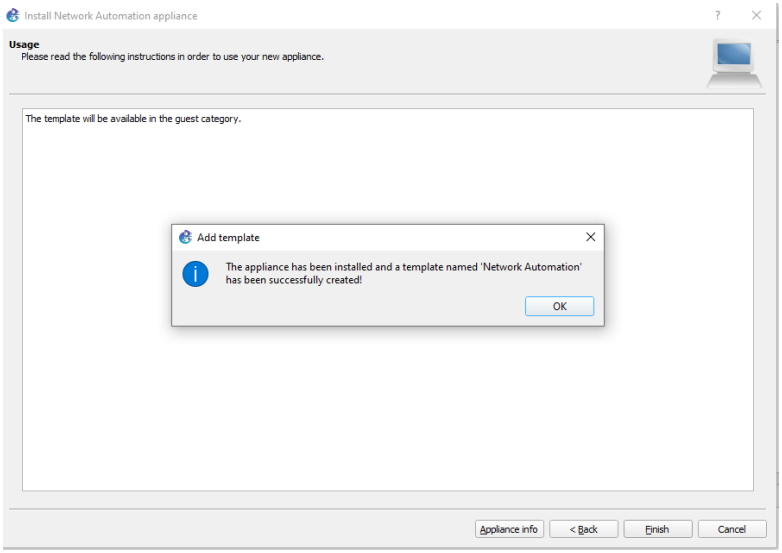

Figura 3.4 Ventana *Usage*.

Se procede a verificar la versión de Ansible preinstalada en el dispositivo agregado, mediante el comando mostrado en recuadro verde de la **Figura 3.5**, como también en el recuadro rojo que muestra la versión Ansible 2.9.6.

```
root@NetworkAutomation-1:~#
root@NetworkAutomation-1:~# ansible --version
ansible 2.9.6
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/root/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  executable location = /usr/bin/ansible
  python version = 3.8.5 (default, Jan 27 2021, 15:41:15) [GCC 9.3.0]
root@NetworkAutomation-1:~#
```

Figura 3.5 Consola de Ansible con su versión.

Para utilizar la herramienta Ansible para este proyecto de titulación, se debe descargar la colección Cisco IOS. Para lo cual es necesario conectar el dispositivo a la red, con ayuda de la nube que simula un NAT, como se muestra en la **Figura 3.6**.

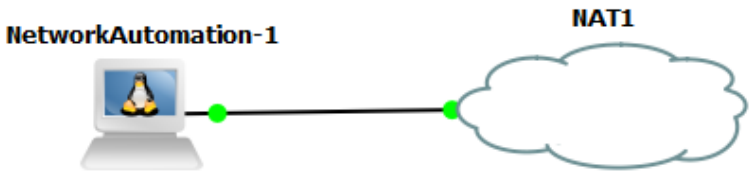

Figura 3.6 Nodo controlador conectado a la red.

Al ingresar a las configuraciones del dispositivo *Network Automation*, donde se despliega la ventana *Node properties*, que se visualiza en la **Figura 3.7**. En la pestaña *General settings* opción *Network configuration*, se selecciona *Edit*, a fin de editar las configuraciones de la red.

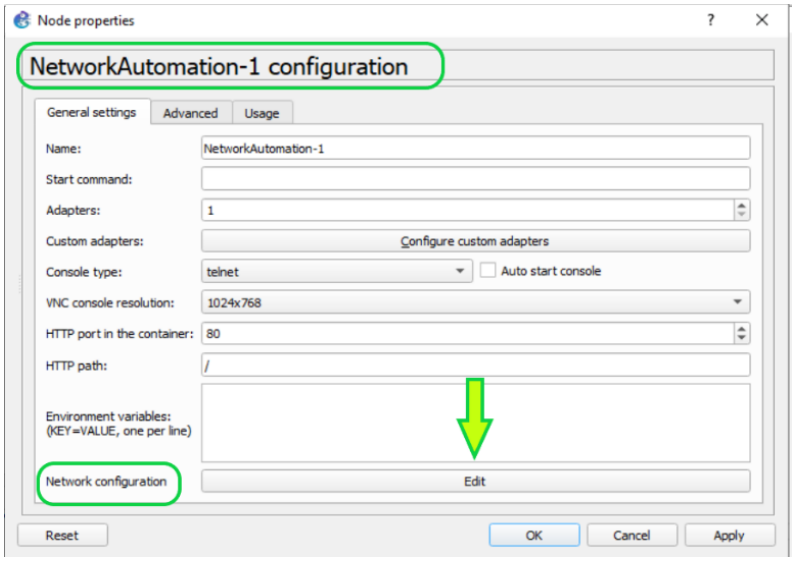

Figura 3.7 Ventana *Node properties*.

Se procede a eliminar los comentarios de las líneas de direccionamiento IP por DHCP y se guardan los cambios, como se muestra en el recuadro verde de la **Figura 3.8**.

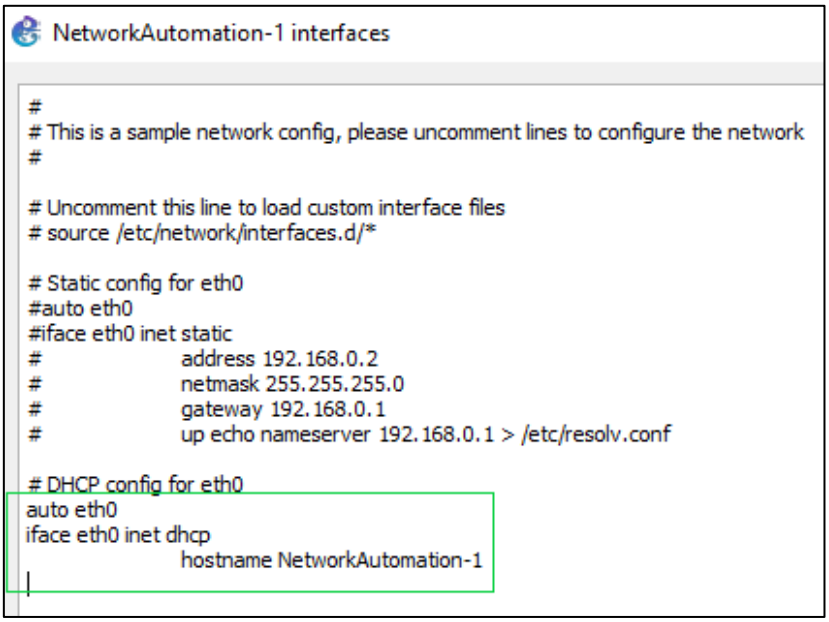

Figura 3.8 Editor de configuraciones de red.

En la consola del dispositivo *Network Automation*, se procede a descargar la colección Cisco IOS, para utilizar los módulos de Ansible, mediante el comando que se muestra en el recuadro verde de la **Figura 3.9**.

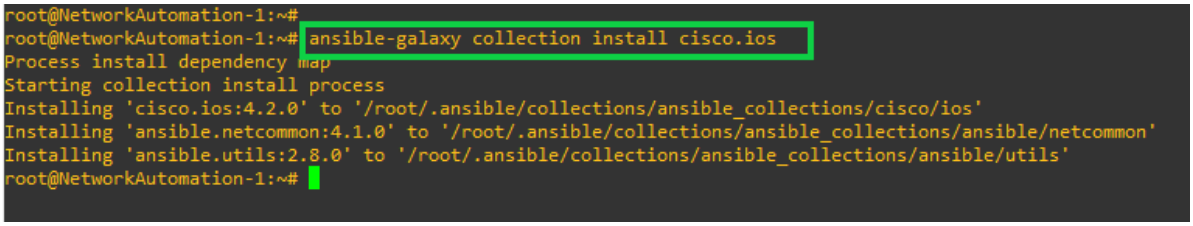

Figura 3.9 Comando para instalar colección Cisco IOS.

Implementación de la topología

Una vez agregado el nodo de control con la herramienta Ansible, se procede a agregar los cuatro routers Cisco c3725, así como también se agrega los equipos VPCS y Ethernet Switches, que conforman parte de la red. A continuación, se realiza las conexiones respectivas entre los equipos de red, como se puede observar en la **Figura 3.10**. La topología consta de cuatro routers donde, en toda la red, se creará dos VRFs para aislar el tráfico entre cada una de las tablas de rutas, también se habilitarán dos protocolos de enrutamiento dinámico y se agregara la configuración NAT.

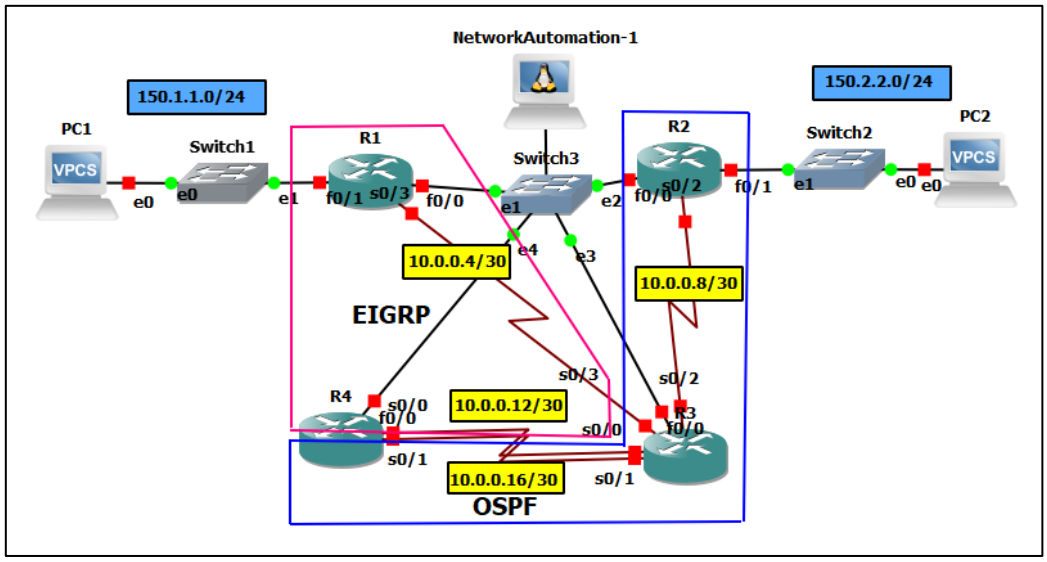

Figura 3.10 Topología de la red.

Conexión SSH y configuración de claves

Para realizar la conexión remota entre los routers y el nodo de control, como primer paso se asigna una dirección IP estática al dispositivo *Network Automation*. Se debe editar el archivo **interfaces**, que se encuentra en la ruta **/etc/network**, con la finalidad de modificar y asignar una dirección IP estática, como se muestra el recuadro amarillo de la **Figura 3.11**.

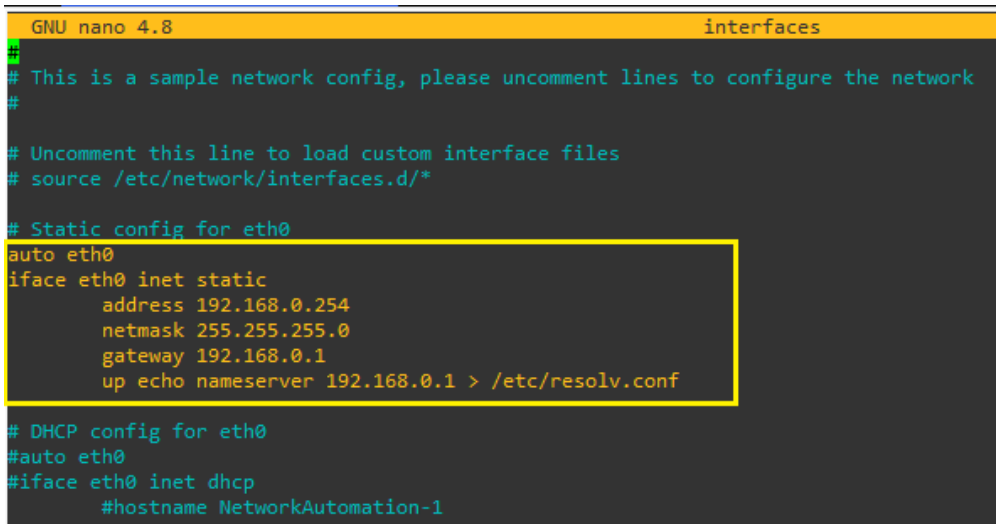

Figura 3.11 Interfaz de configuraciones de red.

Se verifica los cambios en la consola del dispositivo, efectuando el comando mostrado en el recuadro verde de la **Figura 3.12**, así también se observa en el recuadro rojo la dirección IP y máscara de red, anteriormente ingresados.

```
root@NetworkAutomation-1:~#
root@NetworkAutomation-1:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.254  netmask 255.255.255.0  broadcast 0.0.0.0
        inet6 fe80::2845:a9ff:fef8:79d9  prefixlen 64  scopeid 0x20<link>
        ether 2a:45:a9:f8:79:d9  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figura 3.12 Dirección IP estática del dispositivo *Network Automation*.

Configuración de SSH

Para realizar la configuración del equipo Cisco se ingresa a la consola de cada router, para proceder a la configuración de la comunicación SSH. A continuación, se habilita y agrega una dirección IP a la interfaz FastEthernet0/0, con los siguientes comandos mostrados en la **Figura 3.13**.

```
R1#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#interface FastEthernet0/0
R1(config-if)#ip address 192.168.0.10 255.255.255.0
R1(config-if)#no shutdow
R1(config-if)#
*Mar  1 00:02:01.591: %LINK-3-UPDOWN: Interface FastEthernet0/0, changed state to up
*Mar  1 00:02:02.591: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0, changed state to up
R1(config-if)#
```

Figura 3.13 Configuración de interfaz FastEthernet0/0.

Una vez configurado la interfaz de red, por la cual se conecta al nodo de control, se habilita y configura el protocolo SSH en el router Cisco, donde se utiliza los siguientes comandos: definir un nombre de dominio, creación de las credenciales de usuario y contraseña, generar las llaves criptográficas RSA, seleccionar la versión 2 del protocolo SSH, establecer la contraseña para la conexión de consola, habilitar los puertos virtuales para la comunicación SSH, limitar la comunicación a conexiones SSH y autentificación del área local de los usuarios, como se observa en la **Figura 3.14**.

```
*Mar  1 00:02:54.479: %SYS-5-CONFIG_I: Configured from console by console
R1#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#ip domain-name christian
R1(config)#username david privilege 15 secret david123
R1(config)#crypto key generate rsa
The name for the keys will be: R1.christian
Choose the size of the key modulus in the range of 360 to 2048 for your
  General Purpose Keys. Choosing a key modulus greater than 512 may take
  a few minutes.

How many bits in the modulus [512]: 1024
% Generating 1024 bit RSA keys, keys will be non-exportable...[OK]

R1(config)#
*Mar  1 00:03:46.135: %SSH-5-ENABLED: SSH 1.99 has been enabled
R1(config)#ip ssh version 2
R1(config)#enable secret 1234
R1(config)#line vty 0 4
R1(config-line)#transport input ssh
R1(config-line)#login local
R1(config-line)#do wr
Building configuration...
[OK]
R1(config-line)#
```

Figura 3.14 Configuraciones para habilitar conexión SSH.

Generación de llaves SSH en Ansible

Se debe generar las llaves SSH, en la herramienta Ansible, por lo cual se realiza modificaciones en el archivo de configuraciones de SSH, este archivo se encuentra dentro de la ruta **/etc/ssh/ssh_config**. Una vez que se ingresa al archivo **ssh_config**, se elimina el comentario de la línea que se muestra en el recuadro rojo de la **Figura 3.15**, que sirve para negociar el cifrado AES 128 o 256. Además, se agrega dentro del archivo de configuraciones de SSH, la línea que se muestra en el recuadro verde, que ayuda al intercambio de llaves SSH.

```
#   IdentityFile ~/.ssh/id_dsa
#   IdentityFile ~/.ssh/id_ecdsa
#   IdentityFile ~/.ssh/id_ed25519
#   Port 22
    Ciphers aes128-ctr,aes192-ctr,aes256-ctr,aes128-cbc,3des-cbc
#   MACs hmac-md5,hmac-sha1,umac-64@openssh.com
#   EscapeChar ~
#   Tunnel no
#   TunnelDevice any:any
#   PermitLocalCommand no
#   VisualHostKey no
#   ProxyCommand ssh -q -W %h:%p gateway.example.com
#   RekeyLimit 1G 1h
    SendEnv LANG LC_*
    HashKnownHosts yes
    GSSAPIAuthentication yes
    KexAlgorithms +diffie-hellman-group1-sha1
```

Figura 3.15 Archivo de configuraciones SSH.

Luego de realizar los cambios en el archivo SSH, se procede a crear la llave pública y privada que posibilita la autentificación para la conexión remota mediante el protocolo SSH, a fin de establecer la comunicación entre el nodo de control y cada uno de los routers. Para generar las llaves de SSH, se ingresa el comando, que se muestra en el recuadro azul de la **Figura 3.16**.

```
root@NetworkAutomation-1:/#
root@NetworkAutomation-1:/# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:Mbiw8hfiu3KrE75bjmeWqBM4gpU59Rrptx6XnyPYFik root@NetworkAutomation-1
The key's randomart image is:
+---[RSA 3072]----+
|                 |
|     .  .        |
|   +.o. o        |
|  = oo.. o       |
|o..oooo S.       |
|* .+o.E.o.       |
|.+ o+oo=o.       |
|. =+*ooo+...     |
|.o=@=+.. .o.     |
+----[SHA256]-----+
root@NetworkAutomation-1:/#
```

Figura 3.16 Comando para generar llaves SSH.

Ya generada las llaves SSH, se debe copiar dichas llaves, entre el nodo de control y el router, con la finalidad de facilitar la conexión sin necesidad de la contraseña en cada inicio de sesión, gracias a que copia la llave pública en el servidor para otorgar el acceso. Esto se realiza mediante el comando que se muestra en el recuadro verde de la **Figura 3.17**, seguido del nombre del usuario y la dirección IP de la interfaz de red, por la cual se conectará el nodo de control.

```
root@NetworkAutomation-1:/# ssh-copy-id david@192.168.0.10
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/root/.ssh/id_rsa.pub"
The authenticity of host '192.168.0.10 (192.168.0.10)' can't be established.
RSA key fingerprint is SHA256:XkL5z5/GdjkHb7dhYLkjaLBjSVr81a/nOss3C38YeyQ.
```

Figura 3.17 Comando para copiar llaves SSH.

Luego de copiar las llaves de SSH, se debe iniciar sesión, para verificar y asegurar que se agregaron correctamente las llaves, mediante el comando mostrado en el recuadro verde de la **Figura 3.18**.

```
root@NetworkAutomation-1:/# ssh david@192.168.0.10
Password:

R1#
```

Figura 3.18 Comando para iniciar sesión SSH.

Construcción del Inventario

Como primer paso, para realizar las configuraciones por medio de Ansible, se debe crear el archivo del inventario, el cual se utiliza para organizar los hosts que posteriormente van a ser gestionados y configurados. Por defecto, Ansible cuenta con un archivo de inventario, llamado hosts. Para este proyecto de titulación se creó un nuevo inventario en la ruta **/etc/ansible/hosts-vrf**, mediante el comando mostrado en el recuadro verde de la **Figura 3.19**.

```
root@NetworkAutomation-1:/# cd /etc/ansible
root@NetworkAutomation-1:/etc/ansible# nano hosts-vrf
```

Figura 3.19 Comando para crear el archivo del inventario.

Inmediatamente creado el archivo, se procede a definir los hosts y los grupos de una manera organizada. Con el inventario, se simplifica el proceso de gestión y configuración de los equipos de *networking*. El inventario consta de diferentes secciones como se observa en la **Figura 3.20**, a continuación, se describe las diferentes secciones del inventario como:

- Sección uno tiene como encabezado entre corchetes ***[divice],*** el nombre del equipo o alias, además se define la dirección IP con el comando ***ansible_host="ip-address".***
- Sección dos se agrega el grupo de variables, con el encabezado entre corchetes ***[divice:vars].*** El grupo de variables contiene el nombre del usuario y contraseña, que se utiliza para conectarse al host. Estos datos se definen mediante los comandos ***ansible_user="user"*** y ***ansible_password="password"***, respectivamente.
- Sección tres se agrega un grupo de variables globales ***[divice_group:vars]***, que contendrá las siguientes líneas de comandos como: establece la conexión para ejecutar comandos en los equipos de *networking* ***ansible_connection=network_cli***, reconocer el sistema operativo de los equipos de *networking* ***ansible_network_os=ios*** y especificar la versión de

Python, con la que se usará automáticamente para trabajar ***ansible_python_interpreter=/usr/bin/python3.***

- Sección cuatro se agrega el grupo de los subgrupos de hosts, con el siguiente encabezado ***[divice_group:children],*** que contiene el nombre o alias de cada uno de los equipos de *networking*, a fin de simplificar el proceso de gestión y configuración.

```
  GNU nano 4.8                                          hosts-vrf
[r1a]
Router1 ansible_host=192.168.0.10
[r1a:vars]
ansible_user=david
ansible_password=david123

[r2b]
Router2 ansible_host=192.168.0.20
[r2b:vars]
ansible_user=david
ansible_password=david123

[r3ab]
Router3 ansible_host=192.168.0.30
[r3ab:vars]
ansible_user=david
ansible_password=david123

[r4ab]
Router4 ansible_host=192.168.0.40
[r4ab:vars]
ansible_user=david
ansible_password=david123

[routers:vars]
ansible_connection=network_cli
ansible_network_os=ios
ansible_python_interpreter=/usr/bin/python3

[routers:children]
r1a
r2b
r3ab
r4ab
```

1 2 3 4

Figura 3.20 Archivo del inventario.

Para seleccionar la ubicación del inventario creado y no utilizar el archivo predeterminado de Ansible, es necesario ingresar al archivo de configuraciones de Ansible, dentro de la ubicación ***/etc/ansible*** y editar el archivo ***ansible.cfg***, con el siguiente comando que se muestra en el recuadro verde de la **Figura 3.21**.

```
root@NetworkAutomation-1:/# cd /etc/ansible
root@NetworkAutomation-1:/etc/ansible# ls
ansible.cfg  hosts  hosts-vrf
root@NetworkAutomation-1:/etc/ansible# nano ansible.cfg
```

Figura 3.21 Archivo de configuraciones de Ansible.

Dentro del archivo se procede a borrar el comentario de la línea ***inventory***, a fin de cambiar la ubicación ***/etc/ansible/hosts*** por ***/etc/ansible/hosts-vrf***, el cual contiene la ubicación del nuevo inventario, como se muestra en el recuadro rojo de la **Figura 3.22**.

```
  GNU nano 4.8                                                ansible.cfg
# config file for ansible -- https://ansible.com/
# ===============================================

# nearly all parameters can be overridden in ansible-playbook
# or with command line flags. ansible will read ANSIBLE_CONFIG,
# ansible.cfg in the current working directory, .ansible.cfg in
# the home directory or /etc/ansible/ansible.cfg, whichever it
# finds first

[defaults]

# some basic default values...

 inventory      = /etc/ansible/hosts-vrf
#library        = /usr/share/my_modules/
#module_utils   = /usr/share/my_module_utils/
#remote_tmp     = ~/.ansible/tmp
#local_tmp      = ~/.ansible/tmp
#plugin_filters_cfg = /etc/ansible/plugin_filters.yml
#forks          = 5
```

Figura 3.22 Ruta del nuevo inventario.

Creación del primer *playbook*

Para el primer *playbooks* **Router1VRF.yml** tendrá las siguientes configuraciones que se detallan a continuación:

- Cambio de nombre del router, el cual será ingresado por el usuario.
- Crear una VRF, donde el nombre será asignado por el usuario a igual que el distintivo de ruta.
- Configuración de la interfaz FastEthernet0/1 y serial0/3, donde las direcciones IP de cada interfaz será ingresado por el usuario.
- Asignación de la primera VRF a cada una de las interfaces.
- Habilitación del protocolo EIGRP, donde el usuario tendrá que ingresar los datos como: el número del sistema autónomo e ingresar la dirección de red de la interfaz FastEthernet0/1 y serial0/3 para participar del proceso del enrutamiento dinámico,

Para configurar los equipos de *networking*, mediante Ansible, es necesario crear *playbosks* o libros de jugadas, que contengan la lista de tareas que posteriormente se ejecutaran en los hosts que conforman el inventario, para lo cual es necesario crear un archivo en formato YAML. Como primer paso se crea el archivo **Router1VRF.yml** (nombre del *playbook* para router uno) en la ubicación **/etc/ansible**, mediante el comando que se muestra en el recuadro verde de la **Figura 3.23**.

```
root@NetworkAutomation-1:~# cd /etc/ansible/
root@NetworkAutomation-1:/etc/ansible# nano Router1VRF.yml
```

Figura 3.23 Archivo del *playbook* Router1VRF.

Para empezar a desarrollar la lista de jugadas o tareas de los playbooks, es necesario tener en cuenta los siguientes aspectos como:

- Los playbooks se debe expresar en el formato de YAML.
- Un archivo en formato YAML, debe iniciar con tres guiones (- - -) y finalizar con tres puntos (…), ya que la sintaxis del formato YAML, con dichos caracteres, indica el inicio y el final de la lista ordenada de tareas del *playbook*.
- No es admitido las tabulaciones, se debe utilizar la tecla espaciadora para denotar la estructura del código.
- Para una secuencia de listas se utiliza un guion medio (-) y un espacio.
- Al momento de utilizar dos puntos (:), debe ir seguido de un espacio en blanco.
- Se utiliza el símbolo numeral (#), para referirse a un comentario dentro del *playbook*.

En la **Figura 3.24** se muestra la cabecera del libro de jugadas, donde se denota la línea ***– hosts: "alias del equipo de red"***, que se encuentra dentro del inventario de los hosts. La línea ***gather_facts: false***, ayuda a la selección de las variables, para la recopilación de información sobre los nodos administrados. Además, la línea ***remote_user: david***, indica, el nombre del usuario, al cual se establecerá la comunicación remota.

```
  GNU nano 4.8
---
# Configuracion Router 1 VRF ONE

- hosts: r1a
  gather_facts: false
  remote_user: david
```

Figura 3.24 Cabecera del *playbook* Router1VRF.yml.

Uno de los bloques que se encuentra conformado el libro de jugadas es ***vars_prompt***, el cual utiliza el libro de jugadas para solicitar al usuario datos como: nombres, identificadores, direcciones IP entre otros. En la **Figura 3.25** se observa tres secciones como: ***name:*** se ingresa el nombre de la variable, ***prompt:*** es una descripción de la variable y ***private:*** si los datos ingresados por el usuario son visibles.

```
GNU nano 4.8
vars_prompt:

  - name: Name_Router
    prompt: Nombre del dispositivo de Red
    private: no
```

Figura 3.25 Estructura de ingreso de datos.

En la **Figura 3.26** se muestra múltiples datos solicitados por el libro de jugadas como: nombre del router, nombre de la VRF, distintivo de ruta de la VRF creada, dirección IP para cada interfaz de red, número del sistema autónomo de EIGRP y direcciones de las redes a enrutar.

```
- name: Name_Router
  prompt: Nombre del dispositivo de Red
  private: no

- name: Name_VRF
  prompt: Ingrese nombre VRF
  private: no

- name: R_D
  prompt: Ingrese distintivo de ruta ejemplo = "100:100"
  private: no

- name: Interface_f01
  prompt: Ingrese la direccion IP FastEthernet0/1 ejemplo = "150.1.1.1 255.255.255.0" Asignada a VRF
  private: no

- name: Interface_s03
  prompt: Ingrese la direccion IP Serial0/3 ejemplo = "10.0.0.5 255.255.255.252" Asignada a VRF
  private: no

- name: System_Number
  prompt: Ingrese el sistema autonomo de EIGRP "<1-65535>"
  private: no

- name: IP_RED1
  prompt: Ingrese direccion de la red 1 para el enrutamiento EIGRP ejemplo = "150.1.1.1"
  private: no

- name: IP_RED2
  prompt: Ingrese la direccion de la red 2 para el enrutamiento EIGRP ejemplo = "10.0.0.5"
  private: no
```

Figura 3.26 Datos solicitados por el *playbook* Router1VRF.yml.

Otra sección del libro de jugadas es las tareas, donde se identifica mediante la línea ***tasks.*** A continuación, se puede observar en la **Figura 3.27** una de las tareas que contiene el libro de jugadas. Donde se utiliza el módulo ***cisco.ios.ios_config*** a fin de segmentar en secciones la configuración de Cisco IOS. Para cambiar el nombre del host o el equipo de red, se utiliza el comando ***hostname*** seguidamente de la variable solicitada al usuario ***"{{ Name_Router }}".***

```
tasks:

#Nombre Router A

  - name: Nombre Router
    cisco.ios.ios_hostname:
      config:
        hostname: "{{ Name_Router }}"
      state: merged
```

Figura 3.27 Sección tareas del *playbook* Router1VRF.yml.

Con el módulo de configuraciones de Cisco IOS, se crea la VRF también se solicita el nombre de dicha VRF que se almacena en la variable **{{ Name_VRF }}**, donde el nombre de la VRF será asignado por el usuario. En la línea ***name*** se tiene una breve descripción de la tarea a realizar, la línea ***cisco.ios.ios_config*** específica el módulo a utilizar en el libro de jugadas, dentro de ***lines:*** se especificará una lista ordenada de comandos para la configuración del equipo de red, con el comando ***rd*** se agrega un distintivo para la ruta de la VRF, finalmente con ***parents:*** expresa el comando global o de nivel superior de los demás comandos dentro del módulo. La configuración antes descrita se observa en la **Figura 3.28**.

```
#Creando VRF ONE

  - name: VRF ONE
    cisco.ios.ios_config:
      lines:
      - rd {{ R_D }}
      parents: ip vrf {{ Name_VRF }}
```

Figura 3.28 Configuraciones de VRF.

Para la asignación de la primera VRF a las interfaces FastEthernet0/1 y Serial0/3, se utilizó el módulo de configuraciones de Cisco. Los comandos que se puede observar en la **Figura 3.29** son los siguientes:

- Comando global ***interface*** tipo-y-número, para ingresar al modo de configuraciones de la interfaz.
- ***ip vrf forwardinf*** *{{ Name_VRF }}*, tiene la finalidad de asignar la VRF creada a la interfaz de red.
- ***ip address*** *{{ Interface_f01 }},* se utiliza para asignar la dirección IP a la interfaz de red.
- ***no shutdow***, se utiliza para habilitar la interfaz.

```
    - name: Asignacion de la primera VRF a F0/1
      cisco.ios.ios_config:
        lines:
        - ip vrf forwarding {{ Name_VRF }}
        - ip address {{ Interface_f01 }}
        - no shutdow
        parents: interface FastEthernet0/1

# Asignacion de VRF Uno a int S0/3

    - name: Asignacion de primera VRF  a s0/3
      cisco.ios.ios_config:
        lines:
        - ip vrf forwarding {{ Name_VRF }}
        - ip address {{ Interface_s03 }}
        - no shutdown
        parents: interface serial0/3
```

Figura 3.29 Configuraciones de las interfaces de red.

De la misma forma se utilizó el módulo de configuraciones de Cisco IOS, para habilitar el protocolo de enrutamiento EIGRP, como se aprecia en la **Figura 3.30**, también se puede visualizar las siguientes líneas de comandos:

- Comando global ***router eigrp*** *{{ System_Number }}*, para habilitar el protocolo de enrutamiento con el número de sistema autónomo ingresado por el usuario.
- ***address-family ipv4 vrf*** *{{ Name_VRF }},* con la finalidad de ingresar al modo de configuraciones de la familia de direcciones IPv4 para la VRF.
- ***network {{ IP_RED1 }} 0.0.0.0***, con la finalidad de ingresar la IP de red, para el enrutamiento dinámico. La dirección 0.0.0.0 especifica que la dirección IPv4, se encuentra dentro del equipo local, también especifica que no se solicita la máscara de la red.
- ***no auto-summary***, a fin de tener rutas resumidas para otros routers con el mismo protocolo de enrutamiento.
- ***autonomous-system*** *autonomous-system-number*, para ingresar el número del sistema autónomo.

```
    - name: Configuracion de Eigrp
      cisco.ios.ios_config:
        lines:
        - address-family ipv4 vrf {{ Name_VRF }}
        - network {{ IP_RED1 }} 0.0.0.0
        - network {{ IP_RED2 }} 0.0.0.0
        - no auto-summary
        - autonomous-system 254
        parents: router eigrp {{ System_Number }}
```

Figura 3.30 Habilitación del protocolo EIGRP.

Creación del segundo *playbook*

Las configuraciones que tiene el *playbook* **Router2VRF.yml** son las siguientes:

- Cambiar el nombre del router, donde el nombre será ingresado por el usuario.
- Crear la segunda VRF, donde el nombre será asignado por el usuario al igual que el identificador de ruta.
- Configuración de la interfaz FastEthernet0/1 y serial0/2, donde la dirección IP será ingresado por el usuario para cada una de las interfaces.
- Asignación de la segunda VRF a las interfaces anteriormente mencionas.
- Habilitar el protocolo de enrutamiento OSPF, donde el usuario tendrá que ingresar datos como: ID del proceso, la dirección de router ID y las direcciones de red de las interfaces que participarán en el enrutamiento.

En la **Figura 3.31**, se observa múltiples líneas del *playbook* ***Router2.VRF.yml*** tales como:

- ***- hosts: r2b***, se refiere al host o al nodo administrado, que se encuentra dentro del inventario, al cual se va a configurar.
- La línea ***gater_facts: false***, se utiliza para recopilar información de los nodos administrados u hosts.
- ***remote_user: david***, especifica al usuario remoto, al cual se establecerá la comunicación.
- La sección ***vars_prompt***, se utiliza para solicitar al usuario el ingreso de datos o información.
- La línea ***– name: "variable-name"***, se refiere al nombre de la variable donde se almacenará la información ingresada por el usuario.
- La línea ***prompt: "description"***, se utiliza para la descripción de la variable.
- La línea ***private: no***, quiere decir que la información del usuario no será oculta. La variable ***Name_Router***, almacena el nombre del segundo router.
- La variable ***Name_VRF***, almacena el nombre de la segunda VRF.
- La variable ***R_D***, almacena el distintivo de ruta de la VRF.
- La variable ***Interface_f01***, almacena la dirección IP de la interfaz FastEthernet0/1.
- La variable ***Interface_s02***, almacena la dirección IP de la interfaz serial0/2.

```
GNU nano 4.8                                    Router2VRF.yml
# Configuracion

- hosts: r2b
  gather_facts: false
  remote_user: david

  vars_prompt:

    - name: Name_Router
      prompt: Nombre del dispositivo de Red
      private: no

    - name: Name_VRF
      prompt: Ingrese nombre la segunda VRF
      private: no

    - name: R_D
      prompt: Ingrese distintivo de ruta de la segunda VRF  ejemplo = "200:200"
      private: no

    - name: Interface_f01
      prompt: Ingrese la direccion IP FastEthernet0/1 ejemplo = "150.2.2.2 255.255.255.0" Asignada a la segunda VRF
      private: no

    - name: Interface_s02
      prompt: Ingrese la direccion IP Serial0/2 ejemplo = "10.0.0.9 255.255.255.252" Asignada a la segunda VRF
      private: no
```

Figura 3.31 Datos solicitados por el *playbook* Router2VRF.yml.

En la **Figura 3.32** se puede apreciar en los recuadros diferentes tareas para realizar la configuración del segundo router como:

- Primer recuadro, se utiliza el módulo ***cisco.ios.ios_hostname***, para modificar el nombre del router. Donde el nombre del router se encuentra almacenada en la variable ***{{ Name_Router }}.***
- Segundo recuadro, muestra la configuración para la creación de la segunda VRF, tal que el nombre de la VRF se encuentra almacenado en la variable ***{{ Name_VRF }}*** y en la variable ***{{ R_D }}***, se encuentra el distintivo de ruta.
- Tercer recuadro, muestra la asignación de la segunda VRF a la interfaz FastEthernet0/1, donde se tiene las variables ***{{ Name_VRF }}*** que almacena el nombre de la segunda VRF y la variable ***{{ Interface_f01 }}*** que almacena la dirección IP para dicha interfaz.
- Cuarto recuadro, de la misma forma, muestra la asignación de la segunda VRF, para la interfaz serial0/2, donde la dirección IP de la interfaz se almacena en la variable ***{{ Interface_s02 }}***.

```
tasks:

#Nombre Router A

  - name: Nombre Router
    cisco.ios.ios_hostname:
      config:
        hostname: "{{ Name_Router }}"
      state: merged
```

1

```
#Creando VRF DOS

  - name: VRF DOS
    cisco.ios.ios_config:
      lines:
      - rd {{ R_D }}
      parents: ip vrf {{ Name_VRF }}
```

2

```
#Asignacion de VRF DOS

  - name: f0/1
    cisco.ios.ios_config:
      lines:
      - ip vrf forwarding {{ Name_VRF }}
      - ip address {{ Interface_f01 }}
      - no shutdown
      parents: interface FastEthernet0/1
```

3

```
#Asignacion de VRF DOS

  - name: configuracion S0/2
    cisco.ios.ios_config:
      lines:
      - ip vrf forwarding {{ Name_VRF }}
      - ip address {{ Interface_s02 }}
      - no shutdown
      parents: interface serial0/2
```

4

Figura 3.32 Tareas del *playbook* Router2VRF.yml.

Para habilitar el protocolo de enrutamiento OSPF, se utilizó el módulo de configuraciones Cisco IOS, como se observa en la **Figura 3.33**, además se utilizó los comandos de Cisco como:

- La línea ***parents: router ospf*** *{{ process-ID }}* ***vrf*** *{{ Name_VRF }},* es el comando de mayor jerarquía que habilita el enrutamiento OSPF y especifica una tabla de rutas de VRF. Donde el proceso ID y el nombre de la VRF, es ingresado por el usuario que se almacenan en las variables respectivas.
- ***capability vrf-lite***, para suprimir comprobaciones del router proveedor perimetral.
- ***router-id {{*** *router_id }}*, para configurar el router ID, que se solicita al usuario para la configuración router.

  network *{{ IP_RED1 }} 0.0.0.0.0* ***area {{*** *Area-id }}*, para el ingreso de la dirección de red e ID del área de OSPF, que se solicita al usuario para la configuración del router. Además, la máscara wildcard 0.0.0.0 especifica que es una dirección de interfaz.

```
#Protocolo de enrutamiento OSPF

    - name: ospf 2
      cisco.ios.ios_config:
        lines:
        - capability vrf-lite
        - router-id {{ router_id }}
        - network {{ IP_RED1 }} 0.0.0.0 area {{ Area_ID }}
        - network {{ IP_RED2 }} 0.0.0.0 area {{ Area_ID }}
        parents: router ospf {{ Process_ID }} vrf {{ Name_VRF }}

...
```

Figura 3.33 Habilitación del protocolo OSPF.

Creación del tercer *playbook*

Para el *playbook* **Router3VRF.yml** contienen las siguientes configuraciones que se detallan a continuación:

- Cambio del nombre del router, donde el nombre será ingresado por el usuario.
- Crear la primera VRF, donde el nombre será asignado por el usuario, como también ingresará el distintivo de ruta.
- Crear la segunda VRF, donde el nombre será asignado por el usuario, como también ingresará el distintivo de ruta.
- Configuración de las interfaces serial0/0, serial0/1, serial0/2 y serial0/3, donde el usuario ingresará la dirección IP para cada una de las interfaces.
- Asignación de la primera VRF a las interfaces serial0/0 y serial0/3.
- Asignación de la segunda VRF a las interfaces serial0/1 y serial0/2.
- Habilitación del protocolo EIGRP para las rutas de la primera VRF, donde el usuario ingresará los siguientes datos como: número del sistema autónomo y el ingreso de dos direcciones de red que participan del enrutamiento.
- Habilitación del protocolo OSPF para las rutas de la segunda VRF, donde el usuario ingresará los siguientes datos como: número del proceso ID, dirección del router ID y el ingreso de dos direcciones de red para que participen del enrutamiento.

El *playbook* **Router3VRF.yml**, se puede observar en la **Figura 3.34** algunos de los datos solicitados al usuario, para la configuración del router como:

- La variable **Name_Router**, almacena el nombre del router ingresado por el usuario.

- La variable **Name_VRFONE**, almacena el nombre de la primera VRF, que será ingresado por el usuario.
- La variable **RD_ONE**, almacena el identificador de ruta ingresado por el usuario.
- La variable **Name_VRFTWO**, almacena el nombre de la segunda VRF, que será ingresado por el usuario.
- La variable **RD_TWO**, almacena el identificador de ruta ingresado por el usuario.
- La variable **Interface_s00**, almacena la dirección IP de la interfaz serial0/1.
- La variable **Interface_s03**, almacena la dirección IP de la interfaz serial0/3.

```
vars_prompt:

  - name: Name_Router
    prompt: Nombre del dispositivo de Red
    private: no

  - name: Name_VRFONE
    prompt: Ingrese nombre de la primera VRF
    private: no

  - name: RD_ONE
    prompt: Ingrese distintivo de ruta de la primera VRF ejemplo = "100:100"
    private: no

  - name: Name_VRFTWO
    prompt: Ingrese nombre la segunda VRF
    private: no

  - name: RD_TWO
    prompt: Ingrese distintivo de la ruta de la segunda VRF ejemplo = "200:200"
    private: no

  - name: Interface_s00
    prompt: Ingrese la direccion IP Serial0/0 ejemplo = "10.0.0.13 255.255.255.252" Asignada a la primera VRF
    private: no

  - name: Interface_s03
    prompt: Ingrese la direccion IP Serial0/3 ejemplo = "10.0.0.6 255.255.255.252" Asignada a la primera VRF
    private: no
```

Figura 3.34 Datos solicitados por el *playbook* Router3VRF.yml

Se puede apreciar en los recuadros de la **Figura 3.35** las diferentes tareas del *playbook* como:

- Asignar el nombre del equipo, que ingresará el usuario y se almacenará en la variable ***{{ Name_Router }}.***
- Crear la primera VRF, donde el nombre de la VRF será asignado por el usuario y se almacenara en la variable ***{{ Name_VRFONE }}***.
- Crear la segunda VRF, donde el nombre de la VRF será asignado por el usuario y se almacenara en la variable ***{{ Name_VRFTWO }}***.
- Asignar la primera VRF, a la interfaz serial0/0 e ingresar una dirección IP para dicha interfaz.

```
tasks:

#Nombre Router tres

    - name: Nombre Router
      cisco.ios.ios_hostname:
        config:
          hostname: "{{ Name_Router }}"
        state: merged

#Creando VRF ONE

    - name: VRF TWO
      cisco.ios.ios_config:
        lines:
        - rd {{ RD_ONE }}
        parents: ip vrf {{ Name_VRFONE }}

#Creando VRF TWO

    - name: VRF TWO
      cisco.ios.ios_config:
        lines:
        - rd {{ RD_TWO }}
        parents: ip vrf {{ Name_VRFTWO }}

#Asignacion VRF ONE S0/0

    - name: Asignacion VRF ONE S0/0
      cisco.ios.ios_config:
        lines:
        - ip vrf forwarding {{ Name_VRFONE }
        - ip address {{ Interface_s00 }}
        - no shutdown
        parents: interface serial0/0
```

Figura 3.35 Tareas del *playbook* Router3VRF.yml

Para el tercer *playbook*, se utilizó los protocolos de enrutamiento EIGRP y OSPF, mediante el módulo de configuración Cisco IOS, como se puede apreciar en los recuadros de la **Figura 3.36**.

- Para habilitar el protocolo EIGRP, se tiene una lista de tareas dentro del módulo de configuraciones, como las líneas: ***parents: router eigrp {{ System_Number }}*** este es el comando de mayor jerarquía que habilita el protocolo de enrutamiento, ***address-famili ipv4 vrf {{ Name_VRFONE}}*** ingresa al modo de configuraciones de la familia de direcciones IPv4 para la VRF, **no *auto-summary*** se tiene rutas resumidas y ***network*** para definir las redes que participan del enrutamiento.
- Para habilitar el protocolo OSPF, se tiene una lista de tareas dentro del módulo de configuraciones como las líneas: ***parents: router ospf {{ Process_ID }} vrf {{ Name_VRFTWO }}*** habilita el protocolo de enrutamiento, **capability vrf-lite** suprime comprobaciones del router proveedor perimetral, **no *auto-summary*** se tiene rutas resumidas y ***network*** para definir las redes que participan del enrutamiento.

```
#Protocolo EIGRP

    - name: eigrp
      cisco.ios.ios_config:
        lines:
        - address-family ipv4 vrf {{ Name_VRFONE }}
        - no auto-summary
        - network {{ IP_RED1_EIGRP }} 0.0.0.0
        - network {{ IP_RED2_EIGRP }} 0.0.0.0
        - autonomous-system 254
        parents: router eigrp {{ System_Number }}

#Protocolo OSPF

    - name: ospf 3
      cisco.ios.ios_config:
        lines:
        - capability vrf-lite
        - router-id {{ router_id }}
        - network {{ IP_RED1_OSPF }} 0.0.0.0 area {{ Area_ID }}
        - network {{ IP_RED2_OSPF }} 0.0.0.0 area {{ Area_ID }}
        parents: router ospf {{ Process_ID }} vrf {{ Name_VRFTWO }}
...
```

Figura 3.36 Habilitación de los protocolos EIGRP y OSPF.

Creación del cuarto *playbook*

Para el *playbook* **Router4VRF.yml** contiene las siguientes configuraciones que se detallan a continuación:

- Cambiar el nombre del router que será asignado por el usuario.
- Crear la primera VRF, donde el nombre será asignado por el usuario como también el distintivo de ruta.
- Crear la segunda VRF, donde el nombre será asignado por el usuario como también el distintivo de ruta.
- Configuración de la interfaz serial0/0 y serial0/1, donde la dirección IP será ingresado por el usuario para cada una de las interfaces.
- Asignación de la primera VRF a la interfaz serial0/0.
- Asignación de la segunda VRF a la interfaz serial0/1.
- Habilitación del protocolo EIGRP para las rutas de la primera VRF, donde el usuario ingresará los siguientes datos como: número del sistema autónomo y el ingreso de una dirección de red.
- Habilitación del protocolo OSPF para las rutas de la segunda VRF, donde el usuario ingresará los siguientes datos como: número del proceso ID, dirección del router ID y el ingreso de una dirección de red.
- Habilitar la configuración de la traducción de direcciones de red.

El *playbook* **Router4VRF.yml**, se puede visualizar en la **Figura 3.37**, en la línea ***– hosts: r4ab***, especifica el alias del cuarto router, al cual se establecerá la conexión. La

línea ***gather_facts: false*** se utiliza para la recopilación de variables de los hosts remotos, en este caso no es necesario, por lo que está en estado ***false***. Además, la línea ***remote_user: david*** específica al usuario remoto que se establecerá la conexión.

```
---
# Configuracion

- hosts: r4ab
  gather_facts: false
  remote_user: david
```

Figura 3.37 Cabecera *playbook* Router4VRF.yml.

En la **Figura 3.38** se observa, los diferentes datos solicitados al usuario al momento de ejecutar el *playbook* **Router4.VRF.yml** como:

- Nombre del router, que se almacenará en la variable ***Name_Router***.
- Nombre de la primera VRF, que se almacenará en la variable ***Name_VRFONE***, como también ***RD_ONE*** contendrá el distintivo de ruta de la VRF.
- Nombre de la segunda VRF, que se almacenará en la variable ***Name_VRFTWO***, como también ***RD_TWO*** contendrá el distintivo de ruta de la VRF.
- Ingreso de la dirección IP para las interfaces serial0/0 y serial0/1, que se almacenará en las variables ***Interface_serial00*** e ***Interface_serial01*** respectivamente.
- Ingreso del número del sistema autónomo del protocolo EIGRP, que se almacenará en la variable ***System_Number***.

```
- name: Name_Router
  prompt: Nombre del dispositivo de Red
  private: no

- name: Name_VRFONE
  prompt: Ingrese nombre la primera VRF
  private: no

- name: RD_ONE
  prompt: Ingrese distintivo de ruta de la primera VRF ejemplo = "100:100"
  private: no

- name: Name_VRFTWO
  prompt: Ingrese nombre de la segunda VRF
  private: no

- name: RD_TWO
  prompt: Ingrese distintivo de ruta de la segunda VRF ejemplo = "200:200"
  private: no

- name: Interface_s00
  prompt: Ingrese la direccion IP Serial0/0 ejemplo = "10.0.0.14 255.255.255.252" Asignada a la primera VRF
  private: no

- name: Interface_s01
  prompt: Ingrese la direccion IP Serial0/1 ejemplo = "10.0.0.18 255.255.255.252" Asignada a la segunda VRF
  private: no

- name: System_Number
  prompt: Ingrese el sistema autonomo de EIGRP "<1-65535>"
  private: no
```

Figura 3.38 Datos solicitados al usuario por el *playbook* Router4VRF.yml**.**

En la **Figura 3.39** se visualiza los demás datos solicitados por el *playbook* **Router4VRF.yml** como:

- El ingreso de la dirección de red, para el enrutamiento EIGRP, que se almacenará en la variable ***IP_RED1_EIGRP***.
- El número del identificador del proceso ID de OSPF, que se almacenará en la variable ***Process_ID***.
- El ingreso de la dirección del router ID, que se almacenará en la variable ***router_id***.
- El ingreso de la dirección de red, para el enrutamiento OSPF, que se almacenará en la variable ***IP_RED1_OSPF***.
- Ingreso del área de OSPF, que se almacenará en la variable ***Area_ID***.
- Ingreso del número de cada una de las interfaces de loopback, que se almacenarán en las variables ***Loopback_Number_One*** y ***Loopback_Number_Two*** respectivamente.
- El ingreso de la dirección IP para las loopbacks, que se almacenará en la variable ***IP_Loopback***.

```
- name: IP_RED1_EIGRP
  prompt: Ingrese direccion de la red 1 para el enrutamiento EIGRP ejemplo = "10.0.0.14"
  private: no

- name: Process_ID
  prompt: Ingrese identificacion del proceso de OSPF "<1-65535>"
  private: no

- name: router_id
  prompt: Ingrese Router-ID ejemplo = "4.4.4.4"
  private: no

- name: IP_RED1_OSPF
  prompt: Ingrese la direccion de la red 1 para el enrutamiento OSPF ejemplo = "10.0.0.18"
  private: no

- name: Area_ID
  prompt: Ingrese Id del area de OSPF
  private: no

- name: Loopback_Number_One
  prompt: Ingrese numero de interfaz de loopback 1
  private: no

- name: Loopback_Number_Two
  prompt: Ingrese el numero de interfaz de loopback 2
  private: no

- name: IP_Loopback
  prompt: Ingrese direccion IP para las Loopback  ejemplo = "100.4.4.4 255.255.255.255"
  private: no
```

Figura 3.39 Datos solicitados por el *playbook* Router4VRF.yml.

En la **Figura 3.40** se puede observar cuatro tareas que contiene el *playbook* **Router4VRF.yml** son las siguientes:

- Cambio del nombre del router.
- Crear y asignar el nombre de la primera y segunda VRF, con sus respectos distintivos de rutas.
- Asignación de la primera VRF a la interfaz serial0/0 y levantamiento de la interfaz.

```
#Nombre Router
    - name: Nombre Router
      cisco.ios.ios_hostname:
        config:
          hostname: "{{ Name_Router }}"
        state: merged
#Creando VRF ONE
    - name: VRF TWO
      cisco.ios.ios_config:
        lines:
        - rd {{ RD_ONE }}
        parents: ip vrf {{ Name_VRFONE }}
#Creando VRF TWO
    - name: VRF TWO
      cisco.ios.ios_config:
        lines:
        - rd {{ RD_TWO }}
        parents: ip vrf {{ Name_VRFTWO }}
#Asignacion VRF ONE S0/0
    - name: Asignacion VRF ONE S0/0
      cisco.ios.ios_config:
        lines:
        - ip vrf forwarding {{ Name_VRFONE }}
        - ip address {{ Interface_s00 }}
        - no shutdown
        parents: interface serial0/0
```

Figura 3.40 Tareas del *playbook* Router4VRF.yml.

En la **Figura 3.41** se visualiza tres tareas que contiene el *playbook* Router4VRF.yml que se describen a continuación:

- Asignación de la segunda VRF a la interfaz serial0/1 y levantamiento de la interfaz.
- Habilitación del protocolo EIGRP, para la tabla de rutas de la primera VRF.
- Habilitación del protocolo OSPF, para la tabla de rutas de la segunda VRF.

```
#Asignacion VRF TWO S0/1
    - name: configuracion S0/1
      cisco.ios.ios_config:
        lines:
        - ip vrf forwarding {{ Name_VRFTWO }}
        - ip address {{ Interface_s01 }}
        - no shutdow
        parents: interface serial0/1
#Protocolo EIGRP
    - name: eigrp
      cisco.ios.ios_config:
        lines:
        - address-family ipv4 vrf {{ Name_VRFONE }}
        - no auto-summary
        - network {{ IP_RED1_EIGRP }} 0.0.0.0
        - autonomous-system 254
        parents: router eigrp {{ System_Number }}
#Protocolo OSPF
    - name: ospf 4
      cisco.ios.ios_config:
        lines:
        - capability vrf-lite
        - router-id {{ router_id }}
        - network {{ IP_RED1_OSPF }} 0.0.0.0 area {{ Area_ID }}
        parents: router ospf {{ Process_ID }} vrf {{ Name_VRFTWO }}
```

Figura 3.41 Tareas del *playbook* Router4VRF.yml.

Para el cuarto *playbook*, se implementó la configuración de Cisco IOS NAT. Las configuraciones del *playbook* **Router4VRF.yml**, se pueden apreciar en la **Figura 3.42**, a continuación, se detalla la lista de tareas:

- Para la configuración de las loopbacks se utilizó los siguientes comandos: ***Interface loopback*** para habilitar y asignar una interfaz de loopback, ***ip vrf forwarding*** para asignar una VRF a cada una de las interfaces de loopback***, ip address*** para ingresar la dirección IP e ***ip nat inside*** para la traducción de direcciones IP desde el interior hacia el exterior.
- Para la configuración de las interfaces seriales se utilizó los siguientes comandos: ***interface serial*** ingresa al modo de configuraciones de la interfaz e ***ip nat outside*** para la traducción de direcciones IP desde el exterior hacia el interior.
- Para determinar los permisos de la lista de control de acceso y a quien se le permite el tráfico se utilizó los comandos: ***access-list*** IP extended access list ***permit ip host*** Source address ***any*** e ***ip nat inside source list*** number interface serial ***interface serial*** number ***vrf*** vrf name ***overload*** para establecer una traducción del tráfico local al dispositivo de borde que ejecuta NAT.

```
#loopback
    - name: loopback 100
      cisco.ios.ios_config:
        lines:
        - ip vrf forwarding {{ Name_VRFONE }}
        - ip address {{ IP_Loopback }}
        - ip nat inside
        parents: interface loopback {{ Loopback_Number_One }}
#loopack
    - name: loopback 200
      cisco.ios.ios_config:
        lines:
        - ip vrf forwarding {{ Name_VRFTWO }}
        - ip address {{ IP_Loopback }}
        - ip nat inside
        parents: interface loopback {{ Loopback_Number_Two }}
    - name: Nat outside s0/0
      cisco.ios.ios_config:
        lines:
        - ip nat outside
        parents: interface serial0/0
    - name: Nat outside s0/1
      cisco.ios.ios_config:
        lines:
        - ip nat outside
        parents: interface serial0/1
#NAT
    - name: NAT
      cisco.ios.ios_config:
        lines:
        - access-list 100 permit ip host {{ IP_Host }} any
        - ip nat inside source list 100 interface serial 0/0 vrf {{ Name_VRFONE }} overload
        - ip nat inside source list 100 interface serial 0/1 vrf {{ Name_VRFTWO }} overload
...
```

Figura 3.42 Configuración de Cisco NAT.

3.4 Verificar el funcionamiento del programa

Ejecución del primer *playbook*

Para ejecutar el *playbook*, se utiliza el comando ***ansible-playbook "playbook.yml"***, en el recuadro de la **Figura 3.43** se puede apreciar el comando para ejecutar el *playbook* del primer router.

```
root@NetworkAutomation-1:/etc/ansible# ansible-playbook Router1VRF.yml
```

Figura 3.43 Comando para ejecutar el *playbook*.

Las configuraciones de cada router, se realiza mediante la ejecución del *playbook*, donde se solicita al usuario el ingreso de los datos específicos, para continuar con la configuración. En la **Figura 3.44** se aprecia los datos solicitados e ingresados por parte del usuario como, por ejemplo:

- Se ingresó el nombre del primer router como **Quito**.

- Se ingresó el nombre de la primera VRF como **VRFUNO** y el distintivo de ruta **100:100**.
- Se asignó la dirección IP **150.1.1.1/24** para la interfaz FastEthernet0/1 y **10.0.0.5/30** para la interfaz serial0/3.
- Para el número del sistema autónomo de EIGRP se ingresó el número **1**.
- Las direcciones de red que participarán del enrutamiento EIGRP son: **150.1.1.1** y **10.0.0.5**.

```
Nombre del dispositivo de Red: Quito
Ingrese nombre la primera VRF: VRFUNO
Ingrese distintivo de ruta de la primera VRF  ejemplo = "100:100": 100:100
Ingrese la direccion IP FastEthernet0/1 ejemplo = "150.1.1.1 255.255.255.0" Asignada a la primera VRF: 150.1.1.1 255.255.255.0
Ingrese la direccion IP Serial0/3 ejemplo = "10.0.0.5 255.255.255.252" Asignada a la primera VRF: 10.0.0.5 255.255.255.252
Ingrese el sistema autonomo de EIGRP "<1-65535>": 1
Ingrese direccion de la red 1 para el enrutamiento EIGRP ejemplo = "150.1.1.1": 150.1.1.1
Ingrese la direccion de la red 2 para el enrutamiento EIGRP ejemplo = "10.0.0.5": 10.0.0.5
```

Figura 3.44 Datos para *playbook* Router1VRF.yml.

Una vez ingresado los datos por parte del usuario, continua con la ejecución del *playbook*. En la **Figura 3.45** se puede observar, cinco tareas ejecutadas, dichas tareas realizan cambios en las configuraciones del equipo de *networking*, además se visualiza la correcta ejecución de cada tarea.

- La primera tarea se refiere al cambio del nombre del router.
- La segunda tarea se refiere a la creación y configuración de la primera VRF.
- La tercera y cuarta tarea asigna la VRF a las interfaces FastEhetnet0/1 y serial0/3, como también realiza la configuración de dichas interfaces.
- La quinta tarea habilita y configura el protocolo de enrutamiento EIGRP.

```
PLAY [r1a] ***********************************************************************************************

TASK [Nombre Router] *************************************************************************************
changed: [Router1]

TASK [VRF UNO] *******************************************************************************************
[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 [Asignacion de la primera VRF a F0/1] ***************************************************************
changed: [Router1]

TASK [Asignacion de primera VRF  a s0/3] *****************************************************************
changed: [Router1]

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

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

Figura 3.45 Ejecución de las tareas del *playbook* Router1VRF.yml.

Se procede a ingresar a la consola del router para verificar las configuraciones efectuadas por la ejecución del *playbook*. Para visualizar las configuraciones del equipo,

se hace uso del comando ***show running-config*** mostrado en el recuadro la **Figura 3.46**.

```
Quito#show running-config
Building configuration...
```

Figura 3.46 Comando para mostrar las configuraciones.

En la **Figura 3.47** se muestra en los recuadros de color azul, los cinco cambios efectuados en el router uno, mediante la herramienta Ansible como:

- Modificación del nombre del router.
- Nombre de la primera VRF, con su distintivo de ruta.
- Configuración de las interfaces FastEthernet0/1 y serial0/3, además, la asignación de la VRF a cada una de las interfaces.
- Configuración del protocolo de enrutamiento EIGRP.

```
!
hostname Quito
!
!
ip vrf VRFUNO
 rd 100:100
!
!
interface FastEthernet0/1
 ip vrf forwarding VRFUNO
 ip address 150.1.1.1 255.255.255.0
!
interface Serial0/3
 ip vrf forwarding VRFUNO
 ip address 10.0.0.5 255.255.255.252
 clock rate 2000000
!
```

```
!
router eigrp 1
 auto-summary
 !
 address-family ipv4 vrf VRFUNO
  network 10.0.0.5 0.0.0.0
  network 150.1.1.1 0.0.0.0
  no auto-summary
  autonomous-system 254
 exit-address-family
!
ip forward-protocol nd
!
!
no ip http server
no ip http secure-server
!
no cdp log mismatch duplex
```

Figura 3.47 Configuraciones del router uno.

Ejecución del segundo *playbook*

En la **Figura 3.48** se muestran los datos ingresados por parte del usuario, al momento de ejecutar el *playbook* **Router2VRF.yml** como, por ejemplo:

- Se ingresó el nombre del segundo router como **Sangolqui**.
- Se ingresó el nombre de la segunda VRF como **VRFDOS** y el distintivo de ruta **200:200**.
- Se asignó la dirección IP **150.2.2.2/24** para la interfaz FastEthernet0/1, **10.0.0.9/30** para la interfaz serial0/2.
- Para el número de identificación del proceso de OSPF se ingresó el número **2**.

- Se ingresó la dirección del router ID **2.2.2.2**.
- Las direcciones de red que participan del enrutamiento OSPF son: **150.2.2.2** y **10.0.0.9**.

```
root@NetworkAutomation-1:/etc/ansible# ansible-playbook Router2VRF.yml
Nombre del dispositivo de Red: Sangolqui
Ingrese nombre la segunda VRF: VFRDOS
Ingrese distintivo de ruta de la segunda VRF  ejemplo = "200:200": 200:200
Ingrese la direccion IP FastEthernet0/1 ejemplo = "150.2.2.2 255.255.255.0" Asignada a la segunda VRF: 150.2.2.2 255.255.255.0
Ingrese la direccion IP Serial0/2 ejemplo = "10.0.0.9 255.255.255.252" Asignada a la segunda VRF: 10.0.0.9 255.255.255.252
Ingrese identificacion del proceso de OSPF "<1-65535>": 2
Ingrese Router-ID ejemplo = "2.2.2.2": 2.2.2.2
Ingrese la direccion de la red 1 para el enrutamiento OSPF ejemplo = "150.2.2.2": 150.2.2.2
Ingrese la direccion de la red 2 para el enrutamiento OSPF ejemplo = "10.0.0.9": 10.0.0.9
Ingrese Id del area de OSPF: 0
```

Figura 3.48 Datos para *playbook* Router2VRF.yml**.**

Luego de ingresar los datos solicitados, para la configuración del equipo de red, el *playbook* realiza los cambios en las configuraciones en el segundo router. En la **Figura 3.49** se puede observar cada tarea ejecutada correctamente.

- Primera tarea, modifica el nombre del router.
- Segunda tarea, crea y configura la segunda VRF.
- Tercera y cuarta tarea, configura las interfaces FastEthernet0/1 y serial0/2, como también asigna la segunda VRF a las interfaces.
- Quinta tarea, realiza la configuración del protocolo de enrutamiento OSPF.

```
PLAY [r2b] *********************************************************************************************

TASK [Nombre Router] ************************************************************************************
changed: [Router2]

TASK [VRF DOS] ******************************************************************************************
[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: [Router2]

TASK [f0/1] *********************************************************************************************
changed: [Router2]

TASK [configuracion S0/2] *******************************************************************************
changed: [Router2]

TASK [ospf 2] *******************************************************************************************
changed: [Router2]

PLAY RECAP **********************************************************************************************
Router2                    : ok=5    changed=5    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Figura 3.49 Ejecución de tareas del *playbook* Router2VRF.yml.

Finalmente, se revisa las configuraciones realizadas mediante Ansible, para el segundo router. En la **Figura 3.50** se muestran el detalle de las configuraciones del equipo, como:

- Cambio del nombre del router por **Sangolqui**.
- Creación y configuración de la segunda VRF.
- Configuración de las interfaces FastEthernet0/1 y serial0/2, como también asignación de la VRF a cada interfaz.

- Configuración del protocolo de enrutamiento OSPF, para la tabla de rutas de la segunda VRF.

```
!
hostname Sangolqui
!
!
ip vrf VFRDOS
 rd 200:200
!
!
interface FastEthernet0/1
 ip vrf forwarding VFRDOS
 ip address 150.2.2.2 255.255.255.0
!
interface Serial0/2
 ip vrf forwarding VFRDOS
 ip address 10.0.0.9 255.255.255.252
 clock rate 2000000
!
 speed auto
!
router ospf 2 vrf VFRDOS
 router-id 2.2.2.2
 log-adjacency-changes
 capability vrf-lite
 network 10.0.0.9 0.0.0.0 area 0
 network 150.2.2.2 0.0.0.0 area 0
!
ip forward-protocol nd
!
!
no ip http server
no ip http secure-server
!
no cdp log mismatch duplex
!
!
```

Figura 3.50 Cambios efectuados por el *playbook* Router2VRF.yml.

Ejecución del tercer *playbook*

Para el cuarto *playbook* **Router4VRF.yml**, de igual forma el usuario tendrá que ingresar los datos solicitados, al momento de la ejecución del *playbook*. En la **Figura 3.51** se observa todos los datos ingresados por el usuario como, por ejemplo:

- Se ingresó el nombre del segundo router como **Ambato**.
- Se ingresó el nombre de la primera VRF como **VRFUNO** y el distintivo de ruta **100:100**.
- Se ingresó el nombre de la segunda VRF como **VRFDOS** y el distintivo de ruta **200:200**.
- Se ingresó la dirección IP **10.0.0.13/30** para la interfaz serial0/0, **10.0.0.17/30** para la interfaz serial0/1, **10.0.0.10/30** para la interfaz serial0/2 y **10.0.0.6/30** para la interfaz serial0/3.
- Para el número del sistema autónomo de EIGRP se ingresó el número **1**.
- Las direcciones de red que participan del enrutamiento EIGRP son: **10.0.0.6** y **10.0.0.13**.
- Para el número de identificación del proceso de OSPF se ingresó el número **3**.
- Se ingresó la dirección del router ID **3.3.3.3**.
- Las direcciones de red que participan del enrutamiento OSPF son: **10.0.0.10** y **10.0.0.17**.

```
root@NetworkAutomation-1:/etc/ansible# ansible-playbook Router3VRF.yml
Nombre del dispositivo de Red: Ambato
Ingrese nombre de la primera VRF: VRFUNO
Ingrese distintivo de ruta de la primera VRF ejemplo = "100:100": 100:100
Ingrese nombre la segunda VRF: VRFDOS
Ingrese distintivo de la ruta de la segunda VRF ejemplo = "200:200": 200:200
Ingrese la direccion IP Serial0/0 ejemplo = "10.0.0.13 255.255.255.252" Asignada a la primera VRF: 10.0.0.13 255.255.255.252
Ingrese la direccion IP Serial0/3 ejemplo = "10.0.0.6 255.255.255.252" Asignada a la primera VRF: 10.0.0.6 255.255.255.252
Ingrese la direccion IP Serial0/1 ejemplo = "10.0.0.17 255.255.255.252" Asignada a la segunda VRF: 10.0.0.17 255.255.255.252
Ingrese la direccion IP Serial0/2 ejemplo = "10.0.0.10 255.255.255.252" Asignada a la segunda VRF: 10.0.0.10 255.255.255.252
Ingrese el sistema autonomo de EIGRP "<1-65535>": 1
Ingrese direccion de la red 1 para el enrutamiento EIGRP ejemplo = "10.0.0.6": 10.0.0.6
Ingrese la direccion de la red 2 para el enrutamiento EIGRP ejemplo = "10.0.0.13": 10.0.0.13
Ingrese identificacion del proceso de OSPF "<1-65535>": 3
Ingrese Router-ID ejemplo = "3.3.3.3": 3.3.3.3
Ingrese la direccion de la red 1 para el enrutamiento OSPF ejemplo = "10.0.0.10": 10.0.0.10
Ingrese la direccion de la red 2 para el enrutamiento OSPF ejemplo = "10.0.0.17": 10.0.0.17
Ingrese Id del area de OSPF: 0
```

Figura 3.51 Datos solicitados por *playbook* Router3VRF.yml.

En la **Figura 3.52** se muestra las nueve tareas realizadas por el *playbook,* como también en la parte inferior, muestra un detalle del número de cambios correctamente efectuados, al router tres.

- Primera tarea, modifica el nombre del router.
- Segunda y tercera tarea, crea y configura la primera y segunda VRF respectivamente.
- Cuarta y quinta tarea, configura las interfaces serial0/0y serial0/3, como también asigna la primera VRF a las interfaces.
- Cuarta y quinta tarea, configura las interfaces serial0/0y serial0/3, como también asigna la primera VRF a las interfaces.
- Sexta y séptima tarea, configura las interfaces serial0/1 y serial0/2, como también asigna la segunda VRF a las interfaces.
- Octava tarea, realiza la configuración del protocolo de enrutamiento EIGRP.
- Novena tarea, realiza la configuración del protocolo de enrutamiento OSPF.

```
PLAY [r3ab] ***********************************************************************************************

TASK [Nombre Router] **************************************************************************************
changed: [Router3]

TASK [VRF TWO] ********************************************************************************************
[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: [Router3]

TASK [VRF TWO] ********************************************************************************************
changed: [Router3]

TASK [Asignacion VRF ONE S0/0] ****************************************************************************
changed: [Router3]

TASK [Asignacion VRFONE SO/3] *****************************************************************************
changed: [Router3]

TASK [Asignacion VRF TWO S0/1] ****************************************************************************
changed: [Router3]

TASK [Asignacion VRF TWO s0/2] ****************************************************************************
changed: [Router3]

TASK [eigrp] **********************************************************************************************
changed: [Router3]

TASK [ospf 3] *********************************************************************************************
changed: [Router3]

PLAY RECAP ************************************************************************************************
Router3                    : ok=9    changed=9    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Figura 3.52 Tareas ejecutadas por *playbook* Router3VRF.

En los recuadros de la **Figura 3.53** se puede evidenciar los cambios, realizados en la configuración del router, como:

- Nombre del tercer router, asignado como **Ambato**.
- Creación y configuración de la primera y segunda VRF.
- Configuración de las interfaces serial0/0, serial0/1, serial0/2 y serial0/3, como también asignación de una de las VRF a cada interfaz.
- Configuración del protocolo de enrutamiento EIGRP, para la tabla de rutas de la primera VRF.
- Configuración del protocolo de enrutamiento OSPF, para la tabla de rutas de la segunda VRF.

```
!
hostname Ambato
!
!
ip vrf VRFDOS
 rd 200:200
!
ip vrf VRFUNO
 rd 100:100
!
!
interface Serial0/1
 ip vrf forwarding VRFDOS
 ip address 10.0.0.17 255.255.255.252
 clock rate 2000000
!
interface Serial0/2
 ip vrf forwarding VRFDOS
 ip address 10.0.0.10 255.255.255.252
 clock rate 2000000
!
!
interface Serial0/0
 ip vrf forwarding VRFUNO
 ip address 10.0.0.13 255.255.255.252
 clock rate 2000000
!
```

```
!
interface Serial0/3
 ip vrf forwarding VRFUNO
 ip address 10.0.0.6 255.255.255.252
 clock rate 2000000
!
!
router eigrp 1
 auto-summary
 !
 address-family ipv4 vrf VRFUNO
  network 10.0.0.6 0.0.0.0
  network 10.0.0.13 0.0.0.0
  no auto-summary
  autonomous-system 254
 exit-address-family
!
router ospf 3 vrf VRFDOS
 router-id 3.3.3.3
 log-adjacency-changes
 capability vrf-lite
 network 10.0.0.10 0.0.0.0 area 0
 network 10.0.0.17 0.0.0.0 area 0
!
ip forward-protocol nd
!
!
```

Figura 3.53 Cambios efectuados por el *playbook* Router3VRF.yml.

Ejecución del cuarto *playbook*

Para realizar las configuraciones mediante el *playbook* **Router4.VRF.yml**, se debe ingresar los datos solicitados. En la **Figura 3.54** se puede observar los datos ingresados por el usuario.

- Se ingresó el nombre del segundo router como **Cuenca**.
- Se ingresó el nombre de la primera VRF como **VRFUNO** y el distintivo de ruta **100:100**.
- Se ingresó el nombre de la segunda VRF como **VRFDOS** y el distintivo de ruta **200:200**.
- Se ingresó la dirección IP **10.0.0.14/30** y para la interfaz serial0/0, **10.0.0.18/30**.

- Para el número del sistema autónomo de EIGRP se ingresó el número **1**.
- La dirección de red que participa del enrutamiento EIGRP es **10.0.0.14.**
- Para el número de identificación del proceso de OSPF se ingresó el número **4**.
- Se ingresó la dirección del router ID **4.4.4.4**.
- La dirección de red que participa del enrutamiento OSPF es **10.0.0.18**.
- Se ingresó los números para las interfaces de loopback **100** y **200** respectivamente.
- Se ingresó la dirección IP para las interfaces de loopback **100.4.4.4 255.255.255.255**.
- Se ingresó la dirección de host de origen **100.4.4.4**.

```
root@NetworkAutomation-1:/etc/ansible# ansible-playbook Router4VRF.yml
Nombre del dispositivo de Red: Cuenca
Ingrese nombre la primera VRF: VRFUNO
Ingrese distintivo de ruta de la primera VRF ejemplo = "100:100": 100:100
Ingrese nombre de la segunda VRF: VRFDOS
Ingrese distintivo de ruta de la segunda VRF ejemplo = "200:200": 200:200
Ingrese la direccion IP Serial0/0 ejemplo = "10.0.0.14 255.255.255.252" Asignada a la primera VRF: 10.0.0.14 255.255.255.252
Ingrese la direccion IP Serial0/1 ejemplo = "10.0.0.18 255.255.255.252" Asignada a la segunda VRF: 10.0.0.18 255.255.255.252
Ingrese el sistema autonomo de EIGRP "<1-65535>": 1
Ingrese direccion de la red 1 para el enrutamiento EIGRP ejemplo = "10.0.0.14": 10.0.0.14
Ingrese identificacion del proceso de OSPF "<1-65535>": 4
Ingrese Router-ID ejemplo = "4.4.4.4": 4.4.4.4
Ingrese la direccion de la red 1 para el enrutamiento OSPF ejemplo = "10.0.0.18": 10.0.0.18
Ingrese Id del area de OSPF: 0
Ingrese numero de interfaz de loopback 1: 100
Ingrese el numero de interfaz de loopback 2: 200
Ingrese direccion IP para las Loopback  ejemplo = "100.4.4.4 255.255.255.255": 100.4.4.4 255.255.255.255
Ingrese direccion del host de origen ejemplo = "100.4.4.4": 100.4.4.4
```

Figura 3.54 Datos solicitados por *playbook* Router4VRF.yml.

En la **Figura 3.55** se puede visualizar, todas las tareas ejecutadas por el *playbook*, como también un resumen del número de cambios realizados en las configuraciones del cuarto router.

- Primera tarea, modifica el nombre del router.
- Segunda y tercera tarea, crea y configura la primera y segunda VRF respectivamente.
- Cuarta tarea, configura la interfaz serial0/0, como también asigna la primera VRF a la interfaz.
- Quinta tarea, configura la interfaz serial0/1, como también asigna la segunda VRF a la interfaz.
- Sexta tarea, realiza la configuración del protocolo de enrutamiento EIGRP.
- Séptima tarea, realiza la configuración del protocolo de enrutamiento OSPF.
- Octava y novena tarea, configuración de las interfaces de loopback.
- Décima y décima primera tarea, identificación de las interfaces serial0/0 y serial0/1 externas.

- Décima segunda tarea, determina los permisos de la lista de control de acceso para permitir el tráfico a las interfaces.

```
PLAY [r4ab] ***********************************************************************************************************************

TASK [Nombre Router] **************************************************************************************************************
changed: [Router4]

TASK [VRF TWO] ********************************************************************************************************************
[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: [Router4]

TASK [VRF TWO] ********************************************************************************************************************
changed: [Router4]

TASK [Asignacion VRF ONE S0/0] ****************************************************************************************************
changed: [Router4]

TASK [configuracion S0/1] *********************************************************************************************************
changed: [Router4]

TASK [eigrp] **********************************************************************************************************************
changed: [Router4]

TASK [ospf 4] *********************************************************************************************************************
changed: [Router4]

TASK [loopback 100] ***************************************************************************************************************
changed: [Router4]

TASK [loopback 200] ***************************************************************************************************************
changed: [Router4]

TASK [Nat outside s0/0] ***********************************************************************************************************
changed: [Router4]

TASK [Nat outside s0/1] ***********************************************************************************************************
changed: [Router4]

TASK [NAT] ************************************************************************************************************************
changed: [Router4]

PLAY RECAP ************************************************************************************************************************
Router4                    : ok=12   changed=12   unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Figura 3.55 Tareas ejecutadas por *playbook* Router3VRF.

Revisión de las tablas de enrutamiento

Para desplegar la tabla de enrutamiento, se hace uso del comando s***how ip route***. Como se observa en la **Figura 3.56.** En la figura entes mencionada se evidencia que las tablas de enrutamiento de cada uno de los cuatro routers, la única ruta que se conoce es la red directamente conectada.

```
Quito#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       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
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

C    192.168.0.0/24 is directly connected, FastEthernet0/0
Quito#
```

```
Ambato#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       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
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

C    192.168.0.0/24 is directly connected, FastEthernet0/0
Ambato#
```

R2

```
Sangolqui#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       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
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

C    192.168.0.0/24 is directly connected, FastEthernet0/0
Sangolqui#
```

R4

```
Cuenca#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       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
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

C    192.168.0.0/24 is directly connected, FastEthernet0/0
Cuenca#
```

Figura 3.56 Tablas de enrutamiento de los routers.

Mediante el comando ***show ip route vrf*** *name-vrf,* muestra la tabla de las rutas, del VRFUNO, como se puede apreciar en la **Figura 3.57**. En la tabla de rutas del primer router que se le asignó con el nombre **Quito**, muestra la red **10.0.0.12** conocida

mediante el protocolo EIGRP, como también las redes directamente conectadas como la **10.0.0.4** y **150.1.1.0**.

```
Quito#show ip route vrf VRFUNO

Routing Table: VRFUNO
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       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
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     10.0.0.0/30 is subnetted, 2 subnets
D       10.0.0.12 [90/2681856] via 10.0.0.6, 00:07:37, Serial0/3
C       10.0.0.4 is directly connected, Serial0/3
     150.1.0.0/24 is subnetted, 1 subnets
C       150.1.1.0 is directly connected, FastEthernet0/1
Quito#
```

Figura 3.57 Tabla de VRFUNO del primer router.

Para el segundo router que se le denominó con el nombre **Sangolqui,** muestra la tabla de rutas de la VRFDOS, como se observa en la **Figura 3.58**. En la tabla de rutas, se tiene la red **10.0.0.10** conocida mediante el protocolo OSPF, como también dos redes directamente conectadas como la **10.0.0.8** y **150.2.2.0**.

```
Sangolqui#
Sangolqui#show ip route vrf VRFDOS

Routing Table: VRFDOS
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       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
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     10.0.0.0/30 is subnetted, 2 subnets
C       10.0.0.8 is directly connected, Serial0/2
O       10.0.0.16 [110/128] via 10.0.0.10, 00:02:26, Serial0/2
     150.2.0.0/24 is subnetted, 1 subnets
C       150.2.2.0 is directly connected, FastEthernet0/1
Sangolqui#
```

Figura 3.58 Tabla de VRFDOS del segundo router.

El tercer router que se le asignó el nombre de **Ambato**, tiene la tabla de rutas, tanto para la VRFUNO como para la VRFDOS, donde se evidencia, que estas tablas de rutas, no comparte ningún camino. En la **Figura 3.59** se tiene la tabla de rutas para VRFUNO, donde se observa la red **150.1.1.0** conocida por el protocolo EIGRP y las redes

directamente conectadas **10.0.0.4** y **10.0.0.12**. Mientras la tabla de rutas para VRFDOS se observa en la **Figura 3.60**, donde la red **150.2.2.0** es conocida por el protocolo OSPF y las redes directamente conectadas son la **10.0.0.8** y **10.0.0.16**.

```
Ambato#show ip route vrf VRFUNO

Routing Table: VRFUNO
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       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
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     10.0.0.0/30 is subnetted, 2 subnets
C       10.0.0.12 is directly connected, Serial0/0
C       10.0.0.4 is directly connected, Serial0/3
     150.1.0.0/24 is subnetted, 1 subnets
D       150.1.1.0 [90/2195456] via 10.0.0.5, 00:57:37, Serial0/3
Ambato#
```

Figura 3.59 Tabla de VRFUNO del tercer router.

```
Ambato#show ip route vrf VRFDOS

Routing Table: VRFDOS
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       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
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     10.0.0.0/30 is subnetted, 2 subnets
C       10.0.0.8 is directly connected, Serial0/2
C       10.0.0.16 is directly connected, Serial0/1
     150.2.0.0/24 is subnetted, 1 subnets
O       150.2.2.0 [110/74] via 10.0.0.9, 00:06:37, Serial0/2
Ambato#
```

Figura 3.60 Tabla de VRFDOS del tercer router.

Para el cuarto router que se le asignó el nombre de **Cuenca**, tiene la tabla de rutas VRFUNO, que se visualiza en la **Figura 3.61**. La tabla de rutas muestra dos redes conocidas por el protocolo EIGRP la **10.0.0.4** y **150.1.1.0**, Además, muestra las redes directamente conectas.

```
Cuenca#show ip route vrf VRFUNO

Routing Table: VRFUNO
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       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
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     100.0.0.0/32 is subnetted, 1 subnets
C       100.4.4.4 is directly connected, Loopback100
     10.0.0.0/30 is subnetted, 2 subnets
C       10.0.0.12 is directly connected, Serial0/0
D       10.0.0.4 [90/2681856] via 10.0.0.13, 00:25:44, Serial0/0
     150.1.0.0/24 is subnetted, 1 subnets
D       150.1.1.0 [90/2707456] via 10.0.0.13, 00:25:44, Serial0/0
Cuenca#
```

Figura 3.61 Tabla de VRFUNO del cuarto router.

También el cuarto router, tiene la tabla de rutas VRFDOS, que se observa en la **Figura 3.62**. Donde se tiene dos redes conocidas por el protocolo OSPF la **10.0.0.8** y **150.2.2.0** y las redes directamente conectadas.

```
Cuenca#show ip route vrf VRFDOS

Routing Table: VRFDOS
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       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
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     100.0.0.0/32 is subnetted, 1 subnets
C       100.4.4.4 is directly connected, Loopback200
     10.0.0.0/30 is subnetted, 2 subnets
O       10.0.0.8 [110/128] via 10.0.0.17, 00:09:38, Serial0/1
C       10.0.0.16 is directly connected, Serial0/1
     150.2.0.0/24 is subnetted, 1 subnets
O       150.2.2.0 [110/138] via 10.0.0.17, 00:09:28, Serial0/1
Cuenca#
```

Figura 3.62 Tabla de VRFDOS de router cuatro.

Pruebas de conectividad

Se realiza una prueba de conectividad, para la VRFUNO NAT, mediante un ping a la red **150.1.1.1** desde la loopback **100**. En la **Figura 3.63** se observa que se obtuvo una tasa de envío de los paquetes del 100 por ciento.

```
Cuenca#ping vrf VRFUNO 150.1.1.1 source loopback 100

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 150.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 100.4.4.4
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/4/4 ms
Cuenca#
```

Figura 3.63 Prueba de conexión a la red 150.1.1.1.

Se realiza la verificación de las traducciones NAT de la VRFUNO, usando el comando ***show ip nat translatios***. En la **Figura 3.64** se observa, las direcciones locales internas son traducidas a direcciones IP locales externas.

```
Cuenca#show ip nat translations vrf VRFUNO verbose

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 150.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 100.4.4.4
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms
Cuenca#show ip nat translations vrf VRFUNO verbose
Pro Inside global      Inside local       Outside local      Outside global
icmp 10.0.0.14:1       100.4.4.4:1        150.1.1.1:1        150.1.1.1:1
    create 00:00:03, use 00:00:03 timeout:60000, left 00:00:56, Map-Id(In): 1,
    flags:
extended, use_count: 0, VRF : VRFUNO, entry-id: 2, lc_entries: 0
Cuenca#
```

Figura 3.64 Comando para verificar traducciones.

Como se observa en la **Figura 3.65**, se realiza una prueba de conexión, tanto a la dirección **150.1.1.1** como a la **150.2.2.2**, como resultado se tiene una tasa de éxito del 0 por ciento. Esto se debe a que dichas direcciones IP no se encuentran dentro de la tabla de enrutamiento global del router. Para poder realizar un ping a las rutas basadas en la tabla de enrutamiento de la VRF, se debe especificar la dirección y el nombre de la VRF a la que pertenece dicha red, utilizado el comando ***ping vrf*** *name-vrf IP-address*.

```
Cuenca#ping 150.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 150.1.1.1, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
Cuenca#
Cuenca#ping vrf VRFUNO 150.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 150.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
Cuenca#
```

```
Cuenca#ping 150.2.2.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 150.2.2.2, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
Cuenca#ping vrf VRFDOS 150.2.2.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 150.2.2.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/4/12 ms
Cuenca#
```

Figura 3.65 Ping a redes de diferentes instancias.

4 CONCLUSIONES

- DevOps no es una tecnología o un software, tampoco es un simple conjunto de prácticas y herramientas, al contrario, engloba toda una cultura que abarca ciertos puntos claves como: colaboración, automatización, integración, monitoreo y mejora continua.
- Ansible es una herramienta de automatización de redes simple y fácil de usar que permite automatizar tareas y configuraciones repetitivas de diferentes equipos de *networking*. Además, tiene una arquitectura sin agentes, por lo que no es necesario instalar ningún software adicional en los equipos de *networking*, que van a ser administrados por el nodo de control.
- Ansible cuenta con una gran comunidad activa, que crean y comparten *playbooks* y módulos. Esto facilita la búsqueda y el uso de soluciones prediseñadas para tareas comunes, ya que los *playbooks* de Ansible son una herramienta poderosa para automatizar tareas y configuraciones, de los equipos de *networking*, también se puede reutilizar y ejecutar varias veces el *playbook*.
- La sección ***vars_prompt*** del *playbook*, solicita al usuario cierta entrada de datos, con la finalidad que las configuraciones sean dinámicas y exista la posibilidad de tener variaciones en la asignación del direccionamiento IP.
- La tecnología VRF permite crear múltiples tablas de enrutamiento virtual en un solo enrutador físico, lo que permite aislar el tráfico de un grupo de usuarios y brindar conectividad segura entre diferentes redes. Esto proporciona un nivel más de seguridad y privacidad en la red.
- VRF es flexible en cuanto a la configuración de diferentes protocolos de enrutamiento para cada tabla de enrutamiento virtual. Lo que permite adaptar el enrutamiento de las redes, a diferentes necesidades que se requieran en la red. Por lo tanto, puede ayudar a disminuir costos de *Hardware* y mantenimiento en la adquisición de equipos de red, ya que permite tener múltiples redes en un solo enrutador.
- Se verificó, que la tabla de enrutamiento global de cada router, no comparten rutas, entre las tablas de cada VRF, por el hecho de que cada tabla de enrutamiento virtual es independiente. Por lo cual se puede aislar el tráfico de los diferentes clientes que se encuentren en diferentes instancias de la tabla de enrutamiento.

5 RECOMENDACIONES

- Para el montar el nodo de control se puede crear una máquina virtual con el sistema operativo Ubuntu Server o descargar el dispositivo *Network Automation*, que contiene herramientas preinstaladas de automatización de redes.
- Para no tener complicaciones al momento de realizar múltiples configuraciones en el router c3725, se requiere modificar la memoria RAM del router u optar por un router de mejores características.
- Para utilizar los módulos en los playbooks se requiere, instalar la versión actual de la colección del servidor ***ansible-galaxy collection install cisco.ios***.
- Al momento de realizar la comunicación por el protocolo SSH, es necesario editar el archivo de configuraciones SSH, para habilitar la línea ***Ciphers aes128-ctr, aes192-ctr, aes256-ctr, aes128-cbc, 3des-cbc***, para la encriptación al momento de la comunicación remota.
- Mediante el comando ***ansible all -m ping***, se verifica la correcta comunicación entre el nodo de control y los hosts.
- Usar el comando ***ansible-playbook "name-playbook" -vvv***, si se tiene un error al momento de ejecutar el *playbook*, para realizar un rastreo de la ejecución de las tareas.
- Al momento de ejecutar un *playbook*, puede generar un error de ejecución debido al tiempo de persistencia de la conexión y al tiempo de espera del comando, ya que el valor por defecto de estos tiempos tiene una duración de 30 segundos. Si se cuenta con una topología grande, es necesario incrementar estos tiempos para no perder la conexión entre el nodo de control y los hosts administrados. Para editar el valor de cada tiempo se ingresa al archivo ***ansible.cfg***.

6 REFERENCIAS BIBLIOGRÁFICAS

[1] L. D. P. R. J. C. G. S. Zoraida Mamani Rodríguez, «Arquitectura basada en Microservicios y DevOps para una ingeniería de software continua,» [En línea]. Available: https://www.redalyc.org/articulo.oa?id=81665362014. [Último acceso: 4 1 2023].

[2] G. Cloud, «DevOps,» [En línea]. Available: https://cloud.google.com/devops?hl=es#section-1. [Último acceso: 4 1 2023].

[3] Microsoft, «¿Qué es DevOps?,» [En línea]. [Último acceso: 4 1 2023].

[4] R. H. Ansible, «Cómo funciona Ansible,» [En línea]. Available: https://www.ansible.com/overview/how-ansible-works. [Último acceso: 4 1 2023].

[5] R. H. Ansible, «Automatización para operaciones confiables, consistentes y conformes,» [En línea]. Available: https://www.ansible.com/use-cases/configuration-management. [Último acceso: 4 1 2023].

[6] R. Hat, «¿Qué es YAML?,» [En línea]. Available: https://www.redhat.com/es/topics/automation/what-is-yaml. [Último acceso: 4 1 2023].

[7] D. Ansible, «Cómo construir tu inventario,» [En línea]. Available: https://docs.ansible.com/ansible/latest/inventory_guide/intro_inventory.html. [Último acceso: 5 1 2023].

[8] R. Hat, «¿Qué es un playbook de Ansible?,» [En línea]. Available: https://www.redhat.com/es/topics/automation/what-is-an-ansible-playbook. [Último acceso: 4 1 2023].

[9] D. Ansible, «Manuales de Ansible,» [En línea]. Available: https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_intro.html. [Último acceso: 5 1 2023].

[10] Ansible, «Instalación de Ansible,» [En línea]. Available: https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html#control-node-requirements. [Último acceso: 5 1 2023].

[11] D. Microsoft, «¿Qué es WSL?,» [En línea]. Available: https://learn.microsoft.com/en-us/windows/wsl/about. [Último acceso: 4 1 2023].

[12] Blogger, «¿Por que no hay controlador Ansible para Windows?,» [En línea]. Available: http://blog.rolpdog.com/2020/03/why-no-ansible-controller-for-windows.html. [Último acceso: 5 1 2023].

[13] GNS3, «Primeros pasos con GNS3,» [En línea]. Available: https://docs.gns3.com/docs/. [Último acceso: 5 1 2023].

[14] GNS3, «GNS3 Software,» [En línea]. Available: https://www.gns3.com/software. [Último acceso: 5 1 2023].

[15] A. Networks, «Virtual Routing and Forwarding,» [En línea]. Available: https://web.archive.org/web/20210824232159/https://avinetworks.com/glossary/virtual-routing-and-forwarding-vrf/. [Último acceso: 12 1 2023].

[16] MEDIACLOUD, «VRF: qué es y las ventajas de un enrutamiento virtual,» [En línea]. Available: https://blog.mdcloud.es/vrf-que-es-y-las-ventajas-de-un-enrutamiento-virtual/. [Último acceso: 12 1 2023].

[17] CISCO, «Guía de diseño de reenvío de ruta virtual,» [En línea]. Available: https://www.cisco.com/c/en/us/td/docs/voice_ip_comm/cucme/vrf/design/guide/vrfDesignGuide.html. [Último acceso: 12 1 2023].

[18] IPWITHEASE, «VRF vs VRF Lite,» [En línea]. Available: https://ipwithease.com/vrf-vs-vrf-lite/#:~:text=VRF%2Dlite%20is%20a%20subset,MBGP%20VPNv4%20or%20MPLS%20required. [Último acceso: 12 1 2023].

[19] R. Hat, «DevOps,» [En línea]. Available: https://www.redhat.com/es/topics/devops#:~:text=DevOps%20es%20un%20modo%20de,de%20servicios%20de%20alta%20calidad.. [Último acceso: 15 2 2023].

[20] Cisco, «Configuring VRF Lite,» [En línea]. Available: https://www.cisco.com/c/en/us/td/docs/optical/15000r8_0/ethernet/454/guide/d80ether/r8vrf.pdf. [Último acceso: 15 2 2023].

[21] Cisco, «Cisco IOS IP Routing: EIGRP Command Reference,» [En línea]. Available: https://www.cisco.com/c/en/us/td/docs/ios/iproute_eigrp/command/reference/ire_book/ire_a1.html. [Último acceso: 15 2 2023].

[22] Cisco, «Configuring a VPN Routing Session,» [En línea]. Available: https://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst4500/15-1/XE_330SG/configuration/guide/config/vrf.html. [Último acceso: 15 2 2023].

[23] Cisco, «IP Addressing: NAT Configuration Guide, Cisco IOS Release 15M&T,» [En línea]. Available: https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipaddr_nat/configuration/15-mt/nat-15-mt-book/iadnat-addr-consv.html. [Último acceso: 15 2 2023].

7 ANEXOS

ANEXO I: Certificado de Originalidad

CERTIFICADO DE ORIGINALIDAD

Quito, D.M. 2 de marzo de 2023

De mi consideración:

Yo, FERNANDO VINICIO BECERRA CAMACHO, en calidad de Director del Trabajo de Integración Curricular titulado DESPLIEGUE DE VRF MEDIANTE DEVOPS elaborado por el estudiante CHRISTIAN DAVID ÑACATO ANDRANGO 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 10%.

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.

LINK

https://epnecuador-my.sharepoint.com/:b:/g/personal/fernando_becerrac_epn_edu_ec/EfNKebBfRZtFm54ebmAjwcgB1VT50e848BfIsD6IOFDlMg?e=9swneO

Atentamente,

Fernando Vinicio Becerra Camacho

Docente

Escuela de Formación de Tecnólogos

ANEXO II: Enlaces

El video del funcionamiento del proyecto de titulación se lo puede encontrar en el enlace https://youtu.be/TyPa9Vzu9-c o mediante el código QR.

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

ANEXO III: Códigos Fuente

Inventario (hosts-vrf)

```
[r1a]

Router1 ansible_host=192.168.0.10

[r1a:vars]

ansible_user=david

ansible_password=david123


[r2b]

Router2 ansible_host=192.168.0.20

[r2b:vars]

ansible_user=david

ansible_password=david123


[r3ab]

Router3 ansible_host=192.168.0.30

[r3ab:vars]

ansible_user=david

ansible_password=david123


[r4ab]

Router4 ansible_host=192.168.0.40

[r4ab:vars]

ansible_user=david

ansible_password=david123
```

```
[routers:vars]
ansible_connection=network_cli
ansible_network_os=ios
ansible_python_interpreter=/usr/bin/python3

[routers:children]
r1a
r2b
r3ab
r4ab
```

Primer *playbook* (Router1VRF)

```
---
- hosts: r1a
  gather_facts: false
  remote_user: david

  vars_prompt:
    - name: Name_Router
      prompt: Nombre del dispositivo de Red
      private: no
    - name: Name_VRF
      prompt: Ingrese nombre la primera VRF
      private: no
    - name: R_D
      prompt: Ingrese distintivo de ruta de la primera VRF  ejemplo = "100:100"
```

```
    private: no
  - name: Interface_f01
    prompt: Ingrese la direccion IP FastEthernet0/1 ejemplo = "150.1.1.1
255.255.255.0" Asignada a la primera VRF
    private: no
  - name: Interface_s03
    prompt: Ingrese la direccion IP Serial0/3 ejemplo = "10.0.0.5 255.255.255.252"
Asignada a la primera VRF
    private: no
  - name: System_Number
    prompt: Ingrese el sistema autonomo de EIGRP "<1-65535>"
    private: no
  - name: IP_RED1
    prompt: Ingrese direccion de la red 1 para el enrutamiento EIGRP ejemplo =
"150.1.1.1"
    private: no
  - name: IP_RED2
    prompt: Ingrese la direccion de la red 2 para el enrutamiento EIGRP ejemplo =
"10.0.0.5"
    private: no


 tasks:
  - name: Nombre Router
    cisco.ios.ios_hostname:
      config:
        hostname: "{{ Name_Router }}"
      state: merged
```

```
- name: VRF UNO
  cisco.ios.ios_config:
    lines:
    - rd {{ R_D }}
    parents: ip vrf {{ Name_VRF }}

- name: Asignacion de la primera VRF a F0/1
  cisco.ios.ios_config:
    lines:
    - ip vrf forwarding {{ Name_VRF }}
    - ip address {{ Interface_f01 }}
    - no shutdow
    parents: interface FastEthernet0/1

- name: Asignacion de primera VRF  a s0/3
  cisco.ios.ios_config:
    lines:
    - ip vrf forwarding {{ Name_VRF }}
    - ip address {{ Interface_s03 }}
    - no shutdown
    parents: interface serial0/3

- name: Configuracion de Eigrp
  cisco.ios.ios_config:
    lines:
    - address-family ipv4 vrf {{ Name_VRF }}
```

```
        - network {{ IP_RED1 }} 0.0.0.0
        - network {{ IP_RED2 }} 0.0.0.0
        - no auto-summary
        - autonomous-system 254
        parents: router eigrp {{ System_Number }}
...
```

Segundo *playbook* (Router2VRF)

```
---
- hosts: r2b
  gather_facts: false
  remote_user: david


  vars_prompt:
    - name: Name_Router
      prompt: Nombre del dispositivo de Red
      private: no
    - name: Name_VRF
      prompt: Ingrese nombre la segunda VRF
      private: n
    - name: R_D
      prompt: Ingrese distintivo de ruta de la segunda VRF  ejemplo = "200:200"
      private: no
    - name: Interface_f01
      prompt: Ingrese la direccion IP FastEthernet0/1 ejemplo = "150.2.2.2
255.255.255.0" Asignada a la segunda VRF
      private: no
```

```
  - name: Interface_s02
    prompt: Ingrese la direccion IP Serial0/2 ejemplo = "10.0.0.9 255.255.255.252"
Asignada a la segunda VRF
    private: no
  - name: Process_ID
    prompt: Ingrese identificacion del proceso de OSPF "<1-65535>"
    private: no
  - name: router_id
    prompt: Ingrese Router-ID ejemplo = "2.2.2.2"
    private: no
  - name: IP_RED1
    prompt: Ingrese la direccion de la red 1 para el enrutamiento OSPF ejemplo =
"150.2.2.2"
    private: no
  - name: IP_RED2
    prompt: Ingrese la direccion de la red 2 para el enrutamiento OSPF ejemplo =
"10.0.0.9"
    private: no
  - name: Area_ID
    prompt: Ingrese Id del area de OSPF
    private: no


 tasks:
  - name: Nombre Router
    cisco.ios.ios_hostname:
      config:
```

```
      hostname: "{{ Name_Router }}"
    state: merged

- name: VRF DOS
  cisco.ios.ios_config:
    lines:
    - rd {{ R_D }}
    parents: ip vrf {{ Name_VRF }}

- name: f0/1
  cisco.ios.ios_config:
    lines:
    - ip vrf forwarding {{ Name_VRF }}
    - ip address {{ Interface_f01 }}
    - no shutdown
    parents: interface FastEthernet0/1

- name: configuracion S0/2
  cisco.ios.ios_config:
    lines:
    - ip vrf forwarding {{ Name_VRF }}
    - ip address {{ Interface_s02 }}
    - no shutdown
    parents: interface serial0/2

- name: ospf 2
```

```
    cisco.ios.ios_config:
      lines:
      - capability vrf-lite
      - router-id {{ router_id }}
      - network {{ IP_RED1 }} 0.0.0.0 area {{ Area_ID }}
      - network {{ IP_RED2 }} 0.0.0.0 area {{ Area_ID }}
      parents: router ospf {{ Process_ID }} vrf {{ Name_VRF }}
...
```

Tercer *playbook* (Router3VRF)

```
---
- hosts: r3ab
  gather_facts: false
  remote_user: david


  vars_prompt:
    - name: Name_Router
      prompt: Nombre del dispositivo de Red
      private: no
    - name: Name_VRFONE
      prompt: Ingrese nombre de la primera VRF
      private: no
    - name: RD_ONE
      prompt: Ingrese distintivo de ruta de la primera VRF ejemplo = "100:100"
      private: no
    - name: Name_VRFTWO
      prompt: Ingrese nombre la segunda VRF
```

private: no

- name: RD_TWO

prompt: Ingrese distintivo de la ruta de la segunda VRF ejemplo = "200:200"

private: no

- name: Interface_s00

prompt: Ingrese la direccion IP Serial0/0 ejemplo = "10.0.0.13 255.255.255.252" Asignada a la primera VRF

private: no

- name: Interface_s03

prompt: Ingrese la direccion IP Serial0/3 ejemplo = "10.0.0.6 255.255.255.252" Asignada a la primera VRF

private: no

- name: Interface_s01

prompt: Ingrese la direccion IP Serial0/1 ejemplo = "10.0.0.17 255.255.255.252" Asignada a la segunda VRF

private: no

- name: Interface_s02

prompt: Ingrese la direccion IP Serial0/2 ejemplo = "10.0.0.10 255.255.255.252" Asignada a la segunda VRF

private: no

- name: System_Number

prompt: Ingrese el sistema autonomo de EIGRP "<1-65535>"

private: no

- name: IP_RED1_EIGRP

prompt: Ingrese direccion de la red 1 para el enrutamiento EIGRP ejemplo = "10.0.0.6"

```
      private: n
    - name: IP_RED2_EIGRP
      prompt: Ingrese la direccion de la red 2 para el enrutamiento EIGRP ejemplo =
"10.0.0.13"
      private: n
    - name: Process_ID
      prompt: Ingrese identificacion del proceso de OSPF "<1-65535>"
      private: no
    - name: router_id
      prompt: Ingrese Router-ID ejemplo = "3.3.3.3"
      private: no
    - name: IP_RED1_OSPF
      prompt: Ingrese la direccion de la red 1 para el enrutamiento OSPF ejemplo =
"10.0.0.10"
      private: no
    - name: IP_RED2_OSPF
      prompt: Ingrese la direccion de la red 2 para el enrutamiento OSPF ejemplo =
"10.0.0.17"
      private: no
    - name: Area_ID
      prompt: Ingrese Id del area de OSPF
      private: no


  tasks:
    - name: Nombre Router
      cisco.ios.ios_hostname:
```

```
    config:
      hostname: "{{ Name_Router }}"
    state: merged

- name: VRF TWO
  cisco.ios.ios_config:
    lines:
    - rd {{ RD_ONE }}
    parents: ip vrf {{ Name_VRFONE }}

- name: VRF TWO
  cisco.ios.ios_config:
    lines:
    - rd {{ RD_TWO }}
    parents: ip vrf {{ Name_VRFTWO }}

- name: Asignacion VRF ONE S0/0
  cisco.ios.ios_config:
    lines:
    - ip vrf forwarding {{ Name_VRFONE }}
    - ip address {{ Interface_s00 }}
    - no shutdown
    parents: interface serial0/0

- name: Asignacion VRFONE SO/3
  cisco.ios.ios_config:
```

```
    lines:
    - ip vrf forwarding {{ Name_VRFONE }}
    - ip address {{ Interface_s03 }}
    - no shutdown
    parents: interface serial0/3

- name: Asignacion VRF TWO S0/1
  cisco.ios.ios_config:
    lines:
    - ip vrf forwarding {{ Name_VRFTWO }}
    - ip address {{ Interface_s01 }}
    - no shutdown
    parents: interface Serial0/1

- name: Asignacion VRF TWO s0/2
  cisco.ios.ios_config:
    lines:
    - ip vrf forwarding {{ Name_VRFTWO }}
    - ip address {{ Interface_s02 }}
    - no shutdown
    parents: interface serial0/2

- name: eigrp
  cisco.ios.ios_config:
    lines:
    - address-family ipv4 vrf {{ Name_VRFONE }}
```

```
        - no auto-summary
        - network {{ IP_RED1_EIGRP }} 0.0.0.0
        - network {{ IP_RED2_EIGRP }} 0.0.0.0
        - autonomous-system 254
        parents: router eigrp {{ System_Number }}

    - name: ospf 3
      cisco.ios.ios_config:
        lines:
        - capability vrf-lite
        - router-id {{ router_id }}
        - network {{ IP_RED1_OSPF }} 0.0.0.0 area {{ Area_ID }}
        - network {{ IP_RED2_OSPF }} 0.0.0.0 area {{ Area_ID }}
        parents: router ospf {{ Process_ID }} vrf {{ Name_VRFTWO }}
...
```

Cuarto *playbook* (Router4VRF)

```
--
- hosts: r4ab
  gather_facts: false
  remote_user: david

  vars_prompt:
    - name: Name_Router
      prompt: Nombre del dispositivo de Red
      private: no
    - name: Name_VRFONE
```

prompt: Ingrese nombre la primera VRF

private: no

- name: RD_ONE

prompt: Ingrese distintivo de ruta de la primera VRF ejemplo = "100:100"

private: no

- name: Name_VRFTWO

prompt: Ingrese nombre de la segunda VRF

private: no

- name: RD_TWO

prompt: Ingrese distintivo de ruta de la segunda VRF ejemplo = "200:200"

private: no

- name: Interface_s00

prompt: Ingrese la direccion IP Serial0/0 ejemplo = "10.0.0.14 255.255.255.252" Asignada a la primera VRF

private: no

- name: Interface_s01

prompt: Ingrese la direccion IP Serial0/1 ejemplo = "10.0.0.18 255.255.255.252" Asignada a la segunda VRF

private: no

- name: System_Number

prompt: Ingrese el sistema autonomo de EIGRP "<1-65535>"

private: no

- name: IP_RED1_EIGRP

prompt: Ingrese direccion de la red 1 para el enrutamiento EIGRP ejemplo = "10.0.0.14"

private: no

```
  - name: Process_ID
    prompt: Ingrese identificacion del proceso de OSPF "<1-65535>"
    private: no
  - name: router_id
    prompt: Ingrese Router-ID ejemplo = "4.4.4.4"
    private: no
  - name: IP_RED1_OSPF
    prompt: Ingrese la direccion de la red 1 para el enrutamiento OSPF ejemplo =
"10.0.0.18"
    private: no
  - name: Area_ID
    prompt: Ingrese Id del area de OSPF
    private: no
  - name: Loopback_Number_One
    prompt: Ingrese numero de interfaz de loopback 1
    private: no
  - name: Loopback_Number_Two
    prompt: Ingrese el numero de interfaz de loopback 2
    private: no
  - name: IP_Loopback
    prompt: Ingrese direccion IP para las Loopback  ejemplo = "100.4.4.4
255.255.255.255"
    private: no
  - name: IP_Host
    prompt: Ingrese direccion del host de origen ejemplo = "100.4.4.4"
    private: no
```

```
tasks:
  - name: Nombre Router
    cisco.ios.ios_hostname:
      config:
        hostname: "{{ Name_Router }}"
      state: merged

  - name: VRF TWO
    cisco.ios.ios_config:
      lines:
      - rd {{ RD_ONE }}
      parents: ip vrf {{ Name_VRFONE }}

  - name: VRF TWO
    cisco.ios.ios_config:
      lines:
      - rd {{ RD_TWO }}
      parents: ip vrf {{ Name_VRFTWO }}

  - name: Asignacion VRF ONE S0/0
    cisco.ios.ios_config:
      lines:
      - ip vrf forwarding {{ Name_VRFONE }}
      - ip address {{ Interface_s00 }}
      - no shutdown
```

```
    parents: interface serial0/0

- name: configuracion S0/1
  cisco.ios.ios_config:
    lines:
    - ip vrf forwarding {{ Name_VRFTWO }}
    - ip address {{ Interface_s01 }}
    - no shutdow
    parents: interface serial0/1

- name: eigrp
  cisco.ios.ios_config:
    lines:
    - address-family ipv4 vrf {{ Name_VRFONE }}
    - no auto-summary
    - network {{ IP_RED1_EIGRP }} 0.0.0.0
    - autonomous-system 254
    parents: router eigrp {{ System_Number }}

- name: ospf 4
  cisco.ios.ios_config:
    lines:
    - capability vrf-lite
    - router-id {{ router_id }}
    - network {{ IP_RED1_OSPF }} 0.0.0.0 area {{ Area_ID }}
    parents: router ospf {{ Process_ID }} vrf {{ Name_VRFTWO }}
```

```
- name: loopback 100
  cisco.ios.ios_config:
    lines:
    - ip vrf forwarding {{ Name_VRFONE }}
    - ip address {{ IP_Loopback }}
    - ip nat inside
    parents: interface loopback {{ Loopback_Number_One }}

- name: loopback 200
  cisco.ios.ios_config:
    lines:
    - ip vrf forwarding {{ Name_VRFTWO }}
    - ip address {{ IP_Loopback }}
    - ip nat inside
    parents: interface loopback {{ Loopback_Number_Two }}

- name: Nat outside s0/0
  cisco.ios.ios_config:
    lines:
    - ip nat outside
    parents: interface serial0/0

- name: Nat outside s0/1
  cisco.ios.ios_config:
    lines:
```

```
      - ip nat outside
      parents: interface serial0/1

  - name: NAT
    cisco.ios.ios_config:
      lines:
      - access-list 100 permit ip host {{ IP_Host }} any
      - ip nat inside source list 100 interface serial 0/0 vrf {{ Name_VRFONE }} overload
      - ip nat inside source list 100 interface serial 0/1 vrf {{ Name_VRFTWO }} overload
...
```