ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

COMUNICACIONES HABILITADAS POR DRONES: REQUISITOS, TECNOLOGÍAS Y AUTOMATIZACIÓN

AUTOMATIZACIÓN DE RED CON HERRAMIENTAS DE CÓDIGO ABIERTO NETMIKO Y ANSIBLE

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

JUAN ESTEBAN QUIJIA QUIJIA

juan.quijia@epn.edu.ec

DIRECTOR: ING. CHRISTIAN JOSÉ TIPANTUÑA TENELEMA, M.Sc.

christian.tipantuna@epn.edu.ec

DMQ, abril 2023

CERTIFICACIONES

Yo, Juan Esteban Quijia Quijia, declaro bajo juramento que el trabajo 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. A través de la presente declaración dejo constancia de que la Escuela Politécnica Nacional podrá hacer uso del presente trabajo según los términos estipulados en la Ley, Reglamentos y Normas vigentes.

JUAN ESTEBAN QUIJIA QUIJIA

Certifico que el presente trabajo de integración curricular fue desarrollado por Juan Esteban Quijia Quijia, bajo mi supervisión.

ING. CHRISTIAN JOSÉ TIPANTUÑA TENELEMA, M.Sc.

DIRECTOR DE PROYECTO

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el producto resultante 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.

JUAN ESTEBAN QUIJIA QUIJIA

M.Sc. CHRISTIAN JOSE TIPANTUÑA TENELEMA

DEDICATORIA

Este trabajo va dedicado para las personas que me supieron ayudar de alguna manera y en especial a mis padres, que a pesar de las adversidades confiaron en que yo podía lograr esto y mucho más. A mi hermana quien estuvo junto a mí en todo momento. Y para el resto de mi familia y amigos quienes siempre me supieron apoyar. Es a todos quien dedico este logro que estoy por conseguir.

AGRADECIMIENTO

Agradezco a mi Padre Juan y a mi Madre Rosa por sus consejos, su cariño, su sacrificio, su apoyo económico, su dedicación, y las enseñanzas las cuales me dieron el ánimo para seguir y nunca rendirme. A mi hermana Irina gracias por ayudarme en los momentos que la necesitaba. Gracias a mis amigos Dannes y Paulina por el apoyo, los consejos y la ayuda que recibí en el transcurro de esta etapa. A mi tía Paola por siempre darme consejos y ánimos para poder culminar esta etapa, gracias. Al resto de mi familia gracias por todas sus palabras de ánimo. Un agradecimiento a los buenos amigos que conocí al realizar mis prácticas pre-profesionales, a Gabriel, Vanessa y Jessy, por sus consejos, la confianza, el apoyo en todo momento durante la realización del trabajo y por cariño hacia mi persona. Por la paciencia, los consejos, los sermones que me ayudaron y toda la colaboración recibida durante la realización de este trabajo Dr. Christian Tipantuña le doy las gracias. Y a todas las personas que me han ayudado y me acompañaron a lo largo de todo este trayecto para alcanzar la meta, doy gracias.

ÍNDICE DE CONTENIDO

RESUMEN

El presente trabajo de integración curricular tiene como objetivo la implementación del aprovisionamiento automático de las distintas configuraciones de red o también denominado como network automation, que se realiza en infraestructuras de comunicación con Ansible. Para ello se utiliza el programa de emulación GNS3, para implementar las distintas topologías en este trabajo. La utilización de GNS3 se llevará a cabo a través de una máquina virtual empleando para este propósito el hypervisor Virtual Box. La máquina virtual desplegada se utilizará para instalar herramientas de automatización y los equipos terminales y de red.

En el trabajo se realizó la simulación en el programa GNS3, se realizó una revisión que consideran algunas topologías cada una con la configuración distinta como BGP, VLANs, MPLS en los equipos de red que se usarán, estos pueden ser Routers, Switches y Firewall, estos equipos son considerados dentro del nodo secundario (los playbook son redactados en YAML y las plantillas son redactadas en JINJA2). Al trabajar en el nodo central, mediante Ansible se hace cargo de que las configuraciones se apliquen a los Equipos de red, mediante los diferentes directorios, ficheros, playbook y plantillas. Cada uno cumple un rol específico para ejecutar las tareas de configuración en todos los nodos cliente.

Ansible es una herramienta de automatización que no usa operador que ayuden a su ejecución, ni base de datos, ni dominios; lo que evita tener ninguna vulnerabilidad en los equipos, así se usará de manera sencilla la herramienta en los equipos de red. Por ende serán usadas en las topologías ya que cada una tiene varios dispositivos de red, y mediante una documentación estructurada y ordenada de cada playbook y plantilla.

El resultado de lo implementado se mostrará mediante la ejecución de los distintos playbooks, en cada una de las topologías de red. Para luego mostrar estos resultados en la documentación realizada.

PALABRAS CLAVE: redes, automatización de redes, GNS3, Ansible, playbooks, plantillas, máquina virtual.

ABSTRACT

The objective of this curricular integration work is to implement the automatic provisioning of the different network configurations or also known as network automation, which is carried out in communication infrastructures with Ansible. For this, the GNS3 emulation program is used to implement the different topologies in this work. The use of GNS3 will be carried out through a virtual machine using the Virtual Box hypervisor for this purpose. The deployed virtual machine will be used to install automation tools and network and terminal equipment.

At work, the simulation was carried out in the GNS3 program, a review was carried out that considers some topologies each with different configuration such as BGP, VLANs, MPLS in the network equipment that will be used, these can be Routers, Switches and Firewall, these teams are considered within the child node (playbooks are written in YAML and templates are written in JINJA2). When working in the central node, through Ansible it takes care that the configurations are applied to the network equipment, through the different directories, files, playbook and templates. Each one fulfills a specific role to execute the configuration tasks in all the client nodes.

Ansible is an automation tool that does not use operators to help its execution, or database, or domains; which avoids having any vulnerability in the equipment, thus the tool will be used in a simple way in the network equipment. Therefore they will be used in the topologies since each one has several network devices, and through a structured and ordered documentation of each playbook and template.

The result of what has been implemented will be shown by executing the different playbooks, in each of the network topologies. To later show these results in the documentation made.

KEY WORDS: networks, network automation, GNS3, Ansible, playbooks, templates, virtual machine.

1 INTRODUCCIÓN

Para que haya una conexión se necesita dos elementos: los factores físicos como hardware y las conexiones (cables) entre computadoras, dentro de eso están los componentes lógicos, estos son protocolos que se usa para la comunicación y el software que permite la comunicación. La cantidad de equipos de la red y las prestaciones que se ofrece, influyen de manera directa en estos elementos ya que pueden aumentar la cantidad y complejidad [7].

En la configuración para cualquier dispositivo se usa una interfaz física de línea de comandos (CLI: Command-line Interface) esto permite a los gestionadores de red realizar configuraciones mediante las distintas líneas de texto y con la interfaz gráfica de usuario (GUI: Graphical User Interface) que utiliza un gran consumo de recursos computacionales. El CLI y GUI son hechas por los gestionadores de red que se enfrentan a un problema de actualización en los equipos de red, ya que hay muchos equipos de diferentes marcas y cada uno con diferentes protocolos de enrutamiento. La actualización de cada equipo ocasiona pérdidas que son significativas para cualquier empresa lo que impacta en la eficiencia operativa, la gestión y la prestación de sus servicios [16].

El trabajo a continuación realizara una configuración automática mediante el uso de scripts y guardarlos en el nodo mediante una herramienta de emulación, que permitirá efectuar las distintas tareas de automatización para los equipos de red mediante la ejecución de dichos scripts y luego mostrar los resultados obtenidos en cada tarea [17].

La Network Automation efectúa una configuración automatizada en la red para las plataformas físicas y virtuales reduce el tiempo en que se realiza los procesos de configuración. En el ámbito de configuración de red hay varias herramientas que ofertan la tecnología como: Chef, Netmiko, Puppet, Salt Stack y Ansible [1].

1.1 OBJETIVOS

1.1.1 OBJETIVO GENERAL

Ejecutar el proceso de configuración y gestión de manera automática en infraestructuras de red usando herramientas Open Source como Netmiko o Ansible.

1.1.2 OBJETIVOS ESPECÍFICOS

- ❒ Estudiar dos distintas alternativas de herramientas Open source (Ansible y Netmiko) para la automatización de una infraestructura de red.
- ❒ Verificar el correcto funcionamiento de la herramienta Open source (Network Automation) mediante la herramienta de simulación (GNS3).
- ❒ Diseñar las 4 redes a usar considerando las infraestructuras (topologías) idóneas para su ejecución, usando la herramienta de simulación (GNS3).
- ❒ Implementar Network Automation en la herramienta de simulación (GNS3) en las 4 diferentes infraestructuras de red.
- ❒ Mostrar los resultados obtenidos a través de la ejecución de Network Automation en cada una de las infraestructuras de red revisadas.

1.2 ALCANCE

En el presente trabajo se realiza el análisis de los sistemas open source como Netmiko y Ansible, que realizaran el proceso de automatización de una red. Para tal propósito se ha seleccionado un sistema de red para entrega y recepción de información, que será eficiente e idóneo para ejecutar los sistemas open source. El trabajo parte de una revisión general de conceptos relacionados con la programación de red, generación de código y scripts para el proceso de automatización. Para poder realizar la verificación del proceso de automatización se utilizará un emulador de red, en específico GNS3 en que se mostrará los procedimientos y resultados.

Una vez seleccionado el sistema open source o herramienta de automatización, se probarán algunos códigos o scripts a usar en los distintos equipos de red en los que se usará la herramienta de automatización. La gestión automática de redes será testeada en 4 diferentes topologías.

Se realiza la emulación de las 4 topologías requeridas como en todos los equipos dentro de la red a usarse, así proceder a ejecutar las pruebas necesarias, lo que garantice el funcionamiento de cada red, como su automatización.

En todas las empresas un administrador de red debe realizar la configuración y gestión de servicios, de las herramientas que se necesiten para que toda la red tenga un buen funcionamiento; esto se piensa en la disponibilidad, escalabilidad y seguridad, todo aquello se tomará en cuenta al realizar la simulación de la Network Automation en este trabajo.

El trabajo pretende ejecutar la configuración automática mediante el uso de scripts y así subirlas al nodo principal para que mediante la herramienta de emulación permita ejecutar y visualizar las tareas. La automatización de equipos de red ejecuta un conjunto de acciones de manera ordenada y correcta.

1.3 MARCO TEÓRICO

1.3.1 AUTOMATIZACIÓN DE UNA RED (NETWORK AUTOMATION)

La Automatización de red son pasos continuos que permiten la creación y ejecución de cambios en la configuración, administración y operación dentro de varios equipos de red. Estos cambios permiten realizar la configuración más eficientemente para un grupo considerable de equipos de red, lo que también es considerado para redes pequeñas. Lo importante al administrar y configurar las redes de manera automática, es para dar entendimiento con otros equipos de red y reducir el error humano. La automatización de red trabaja en un área muy amplia, incluyendo la recolección de datos de los dispositivos de red; lo que conlleva a la solución de problemas como cruce de direcciones IP, actualización de equipos, daño en la comunicación entre equipos, etc, resolviéndose de manera automática. Esta red automatizada debe ser lo bastante deductiva para poder arreglar los problemas ella misma, para ello depende de los factores internos o externos de la red [26].

La automatización se la aplica en distintos tipos de red. Tanto redes de área local (LAN: Local Area Network), redes de área extendida (WAN: Wide Area Network), redes de centros de datos, y redes inalámbricas. Se deduce que la automatización en cualquier equipo de red es manipulado a través de la CLI o una interfaz de programación (API: Application Programming Interfaces) para lograr automatizar la red [9].

A continuación se explican dos maneras de automatización, estas son la automatización basada en scripts y la automatización basada en software[9].

1.3.1.1 Automatización Basada en Scripts

En la Figura 1.1 se muestra un modelo de automatización basada en scripts, eso se especifica en algunos lenguajes de codificación al ejecutar las tareas, y hace que los procedimientos sean

consistentes con lo escrito en el código [20]. Existen algunos lenguajes que prevalecen en la automatización de redes así se puede hablar de, Perl y Tcl, debido a su familiaridad. Como las redes se vuelven cada vez más complicadas, aparecen otros lenguajes de programación que son de código abierto, como Python, que en el último tiempo ha ganado renombre por su facilidad al momento de usar y la flexibilidad [5].

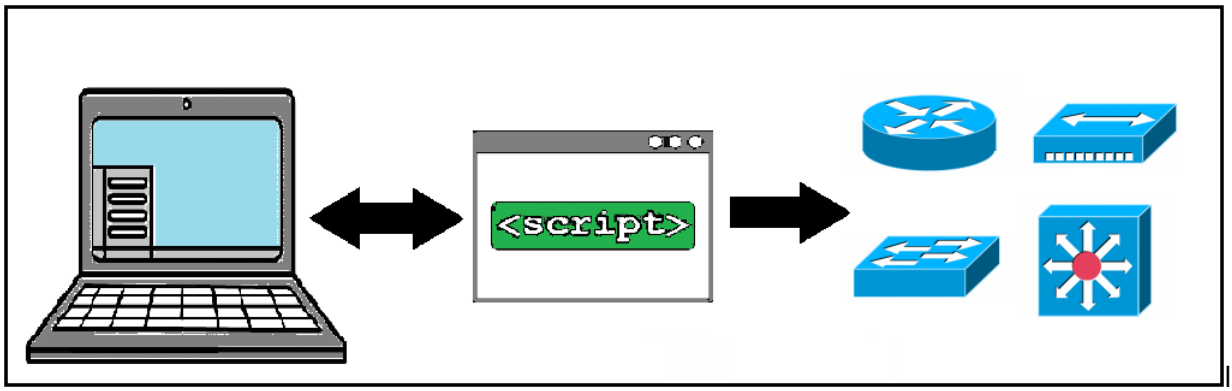

Figura 1.1: Automatización basada en scripts para equipos de red.

1.3.1.2 Automatización Basada en Software

Nombrada como la automatización inteligente, esta se maneja por medio de un nodo central que elimina la necesidad de ejecutar los scripts de manera manual e individual. Estos nodos proporcionan listas para crear y ejecutar tareas establecidas en las políticas de configuración de red, mediante un lenguaje sencillo como el de Python [5].

1.3.2 BENEFICIOS DE LA AUTOMATIZACIÓN DE RED

La automatización de red brinda algunos beneficios dentro de la industria de TI [23]:

- **Agilidad:** Lo que significa rapidez en el despliegue de la automatización en una topología de TI. Productividad y flexibilidad es la ventaja típica de la automatización, lo que beneficia a la red y justifica la situación actual.
- **Escalabilidad:** Se aprovecha de buena manera este beneficio que toma en cuenta los recursos de los equipos de red, para compensar la demanda. Para satisfacer el mejoramiento de la red referente a las nuevas tecnologías, se emplean códigos simples; lo que permite construir, reconstruir, configurar y avanzar sobre la infraestructura de red, esto es posible lograrlo en poco tiempo según las necesidades del entorno.
- **Eficiencia:** Servirá para manejar varias de las tareas y en distintos equipos dentro de una red extensa con mucha facilidad, esto aumentará la agilidad y la eficiencia de la automatización

en un grupo que se pueda desplegar.

- **Gestión de recursos:** Colabora para mantener un diseño de infraestructura que sea coherente. Enfocado en un diseño de red, es basado en el código, esto lleva a una forma moldeable y manipulable de gestionar los equipos de una red.
- **Eficacia en la implementación:** El contenido para una prueba existe en el entorno de implementación, aquí la disciplina es esencial para aplicar la secuencia de comandos exacta y así ejecutar los cambios del entorno, luego para realizar otros cambios se hará de manera rápida.

1.3.3 HERRAMIENTAS DE NETWORK AUTOMATION

Existen algunos tipos de herramientas para la automatización de red, en cuales los protocolos que son usados para ejecutar la automatización son basados en scripts. El CLI es la manera habitual que se usa para implementar la automatización. Como se puede apreciar para este ámbito existen algunas herramientas que se utilizan en los entornos de automatización de redes tal como se muestra en la Figura 1.2 la Network Automation puede usar cualquier herramienta de automatización. Estas herramientas se ejecutan a través de comandos especificados en cada una, para realizar un trabajo que podrá repetirse de manera simple [21].

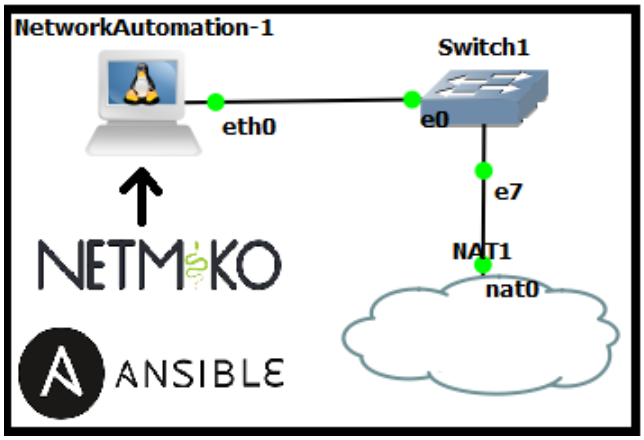

Figura 1.2: Dos Herramientas de Network Automation: Ansible y Netmiko.

1.3.3.1 Ansible

Ansible es una herramienta para la automatización de TI, esta proporciona la ejecución de aplicaciones en sistemas de red. Esta plataforma es de fácil entendimiento para cualquiera que lo necesite, incluyendo a los principiantes, se debe al uso de SSH (Secure SHell) para establecer conexión con varios equipos de una red y lograr ejecutar de manera remota las tareas de configuración en los dispositivos, por ello resulta una herramienta de fácil ejecución debido al uso de scripts. Y esta es usada de realizar la configuración de red [12].

En Ansible, las tareas pueden ser repetidas el número de veces que sea necesario, todo esto sin codificación extra. La aplicación de los scripts se basa en eventos, que incluyen el sistema y la información del resto de equipos para que pueda ejecutar las tareas. Dichas tareas hacen uso de los recursos de la herramienta para la verificación del estado y lograr determinar si necesita realizar un cambio para poder obtener el resultado esperado [12].

Las características más relevantes de Ansible son [23]:

- ❒ Es de código libre que se sirve para administrar y gestionar las tareas de configuración y ejecución de sistemas de red y conectividad.
- ❒ Usa SSH para administrar sistemas de red, esto sin la necesidad de instalar ningún agente de software [1].
- ❒ Es eficiente y simple para su instalación ya que no se necesita ningún archivo de configuración.
- ❒ Configura las tareas a través de playbook (script propio de Ansible) el que usa un lenguaje simple, que se basa en YAML (Yet Another Markup Language).

1.3.3.2 Arquitectura de Ansible

Para describir la arquitectura que usa Ansible, se separan en dos grupos de equipos que son conocidos como, nodo de control y hosts administrados [15].

La herramienta Ansible se instala en el nodo de control o principal y dentro de dicho nodo se mantienen los componentes. Para los hosts administrados se deben enumerar en un inventario de host, éste se encuentra en un archivo de texto dentro del nodo de control que incluye una lista de nombres de los host, sus direcciones IP, inventario de equipos, complementos, redes vecinas, estos componentes se observan en la Figura 1.3 [3].

Para administrar los diferentes sistemas de red debe iniciar sesión en el nodo de control lo que se refiere a correr la herramienta Ansible. En el nodo de control se debe proporcionar uno o varios playbooks y el hosts o el grupo de hosts a los que se deben llegar para lograr administrarlos. Ansible usa SSH para comunicarse con todos los hosts administrados. Hay módulos que se nombran respecto al proceso que realizan, los que se especifica dentro del playbook y se copian dentro del grupo de los hosts administrados que se encuentran en el nodo de control. Luego de ejecutarse, con todos los argumentos que se han especificado en los playbooks que se realizan, cada usuario puede escribir sus propios módulos de manera personalizada, si es necesario. Con Ansible se puede ejecutar la mayoría de las tareas (asignar direcciones IP, nombres de hosts, redes vecinas, etc.) de

[1] Los agentes de software son una herramienta importante para contra restar lo llamado “sobrecarga de información”. Esta tecnología permite que haya aplicaciones concretas funcionando en organizaciones, como en el escritorio de cualquier usuario [25]

administración del sistema de red [3].

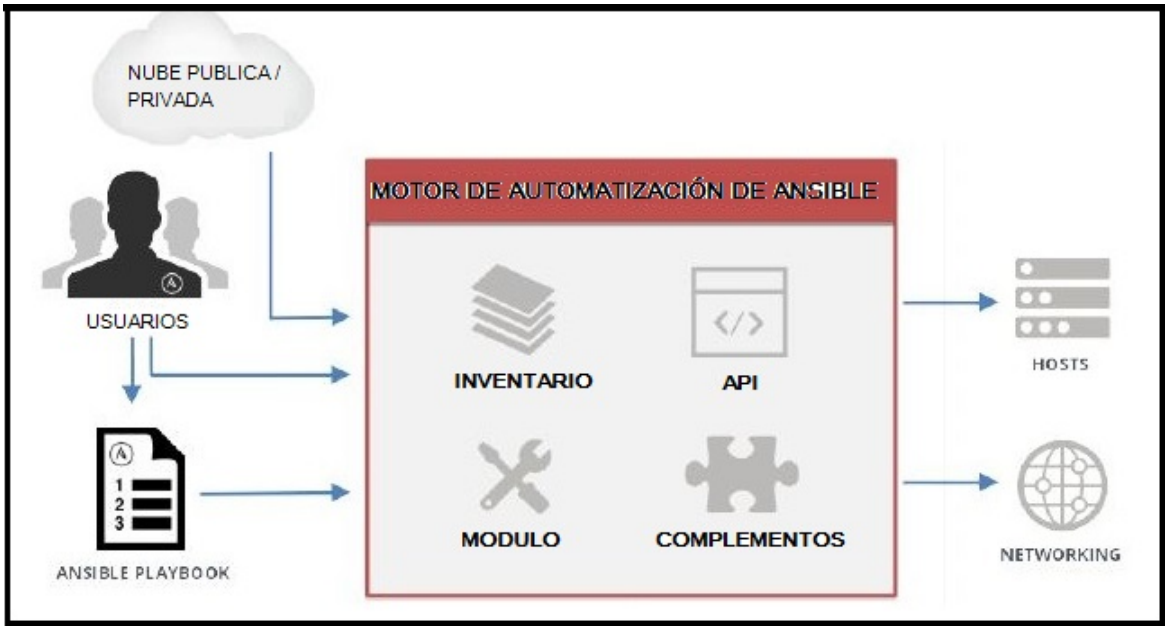

Figura 1.3: Arquitectura de la herramienta de automatización Ansible [3].

Existen varios elementos de Ansible que se mantienen en el nodo de control [3]:

- **Ansible configuración:** Ansible tiene una configuración que muestra la manera de cómo se comporta. Esta configuración incluye a un usuario remoto el que ejecutará comandos y contraseñas, se necesita proporcionar comandos remotos con sudo. En la configuración se observa que los valores de configuración predeterminados (Dirección IP de Ansible) pueden ser reemplazados por valores definidos.
- **Host Inventory:** Se refiere al inventario de todos los host Ansible, que define los equipos y grupos de configuración de la red, a los que pertenecen cada hosts.
- **Core modules:** Son módulos que ya vienen incluidos en la configuración de Ansible los que asignan direcciones IP específicas, para ello existen más de 400 módulos (asignación de direcciones IPv4 e IPv6, generación de direcciones estáticas o por DHCP: Dynamic Host Configuration Protocol) principales.
- **Custom modules:** Se puede ampliar la funcionalidad dentro de Ansible lo que se realiza redactando módulos propios, para luego agregarlos a su biblioteca. Estos módulos habitualmente son escritos en Python (un playbook creado para una red específica), pero pueden ser escritos en otro lenguaje de programación, que sea compatible con Ansible.
- **Playbooks:** Son archivos realizados en sintaxis YAML que representan a los módulos, con argumentos, para ejecutar en los nodos administrados.
- **Connection Plugins:** Son los elementos extra que permiten la comunicación con los hosts gestionados. Se incluye al SSH nativo y local.

- **Plugins:** Son partes de código que se añaden al código usado, estos incrementan la funcionalidad de Ansible.

1.3.3.3 Conceptos de Ansible

INVENTARIO

El inventario define que hosts puede manejar Ansible, ya que pertenecen a agrupaciones que usan para localizar la lista de todos los hosts que se encuentran en el centro de datos (nodo central). Se afirma que un host puede pertenecer a más de un grupo. Existen dos maneras en las que se define los inventarios de los hosts, estos son [13]:

- **Inventario de host estático:** Se define como un grupo de hosts, entonces la sección empieza con un nombre de grupo, que se lo expresa entre corchetes (ver Figura 1.4). Se deben listar las entradas de cada host gestionado que se encuentra en el grupo, cada uno trabaja una sola línea, y para ello deben tener los nombres o las direcciones IP de los hosts que administran dentro del grupo [3].

```
[dist]
D1  ansible_hostname=D1
D2  ansible_hostname=D2

[acc]
A1  ansible_hostname=A1
A2  ansible_hostname=A2

[switches]
dist
acc
```

Figura 1.4: Anatomía de un Playbook [3].

- **Inventario de host dinámico:** La información de las fuentes de inventario dinámico se encuentran en proveedores de una nube pública o privada. Entonces Ansible incluye scripts que maneja la información de manera dinámica en los hosts, grupos y las variables de los proveedores [3].

ESTRUCTURA DE UN PLAYBOOK

El playbook es un comando de nivel superior que se usa para la automatización de los dispositivos de red. El playbook usa YAML para lograr explicar el conjunto de tareas para automatizar, y cada

uno se compone de uno o más plays estos van identificados al inicio de cada play. [7].

En la Figura 1.5 se observa que se asignó números a distintas secciones de un playbook. A continuación describiremos la parte del playbook de la que se trata con respecto a estos números:

- **1- Plays o Jugadas** existe varias dentro de cada playbook de Ansible. Cada playbook consta de partes importantes que vienen en manera de pregunta [22]:
  - La primera es, ¿Qué configurar?, se configura uno o un grupo de hosts para que se reproduzca la jugada. Se incluye información de conexión, de a qué usuario se esta conectando, etc.;
  - La segunda es, ¿Qué ejecutar?, se especifican las tareas que se ejecutan, así también se incluye los componentes del sistema a modificar y el estado en el que están.
- **2- Tasks o tareas** son la última parte de cada play. Es un listado de acciones a efectuar dentro de Ansible y todo eso de manera ordenada según la lista. Existen algunos estilos (command, shell, dest, etc) para expresar los argumentos de los módulos [12].
  Estas tareas que se expresan dependiendo del uso, usan la identificación por nombre de igual manera que los plays. Aquí el texto depende de su uso y se lo muestra cuando de ejecuta el playbook, esto mejora la legibilidad durante su ejecución, este es un parámetro que se usa o no dentro de cada tarea [7].
- **3- Módulos** son scripts que vienen encapsulados dentro Ansible y ejecutan una acción en el host. Existe una gran variedad en los módulos dentro de Ansible [14]. Cualquier lenguaje para la programación es usado para escribir los módulos, pero más a menudo se escriben en Python [7].
- **4- Variables** son una manera eficaz de administrar valores dinámicos dentro del entorno de un proyecto de Ansible. Los nombres de las variables deben usar letras, número y guión bajo; siempre debe empezar por una letra. Y cada administrador usa sus propias variables.[3].

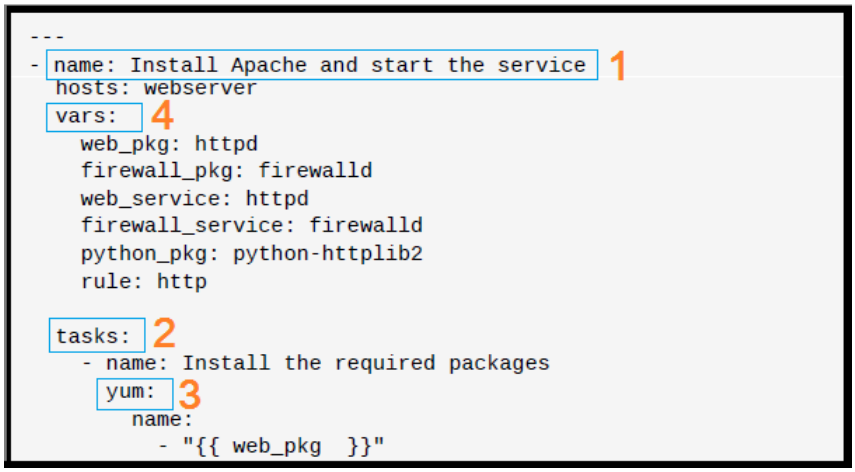

Figura 1.5: Anatomía de un Playbook [3].

1.3.3.4 Netmiko

Esta herramienta que para la automatización usa la biblioteca SSH Python, misma que trata de múltiples proveedores, esto simplifica el proceso de conexión. Netmiko utiliza la lógica específica del proveedor Paramiko. Para realizar la conexión hay que configurar el SSH, para así enviar un comando a un dispositivo de red y obtener lo requerido. El proceso de conexión trabaja con el manejo de la cadena de comandos necesarios para saber cuándo el dispositivo ha enviado su respuesta [7].

Netmiko resuelve muchas de las complejidades, proporcionadas por una biblioteca estándar para las conexiones SSH Python con un conjunto de métodos sencillos de usar. A continuación, se muestran algunos de los procedimientos utilizados por la herramienta [4]:

- ❒ Realiza copias de seguridad, de la configuración de forma programada según se requieran los casos que necesite la red.
- ❒ Ejecuta auditorías de seguridad, para saber si un comando dentro del dispositivo de red está siendo ejecutando de manera correcta.
- ❒ Automatiza la resolución de problemas dentro de la red trabajada, la herramienta ejecuta varios comandos para solucionar un problema.

Las características más relevantes de Netmiko son [4]:

- ❒ Analiza la red a través de las bibliotecas NTC TextFSM y Genie, los que contienen algunos analizadores para una variedad proveedores y tipos de plataformas.
- ❒ Admite una gran variedad de equipos de red, que pueden ser de distintos proveedores.
- ❒ Proporciona maneras para aplicar la configuración en la red, que ocurre a partir de una cadena de comandos o un archivo de comandos.
- ❒ Proporciona varias maneras para leer u observar la codificación empleada para cada equipo de red.
- ❒ Garantiza la estabilidad para dispositivos de red que, trabajan con retardo y que transportan los comandos por la red.

1.3.4 INTRODUCCIÓN A YAML

YAML es un acrónimo que significa “YAML Ain’t Markup Language”, que se afirma como una forma de escritura de datos. Es un lenguaje serial de lectura, diseñado específicamente para facilitar la lectura y edición. Teniendo una manera de escritura simple y también un proceso sobre estas tres estructuras (mapeo, arrays y cadenas) más comunes usadas en el lenguaje serial (siendo un lenguaje secuencial), siendo un lenguaje fácil de usar [8].

Los archivos YAML inician de tres guiones de marcador de documento y de manera opcional se terminan con un marcador de fin de archivo de tres puntos. Entre los marcadores de documento inicial y final como se muestra en la Figura 1.6, existen estructuras de datos que se visualiza mediante un formato de esquema, usando caracteres espaciales para la sangría.

No hay un requisito especifico con respecto al número de caracteres de espacios que se usan para la sangría, además de que los elementos con menor relevancia deben tener más sangría esto indica relaciones anidadas [3].

Los archivos YAML usan etiquetas principalmente para asociar los datos a los nodos secundarios, por ejemplo, en YAML se escoge con qué tipo de usuario trabajar y al objeto que representa un nodo. Los tipos de datos son [8]:

- ❒ Números (hexadecimales / octales, enteros, números de coma flotante).
- ❒ Cadenas (con soporte Unicode).
- ❒ Afirmaciones (verdadero / falso).
- ❒ Marcas de tiempo.
- ❒ Matrices (matrices asociativas / objetos con pares clave-valor).
- ❒ Secuencias (matrices, listas ordenadas).
- ❒ Valor nulo.

1.3.4.1 Reglas de Yaml

- ❒ **Regla 1:** Usa el espaciado fijo como esquema para representar relaciones entre cada capa de datos creada; y aquí no se usa tabulación [26].
- ❒ **Regla 2:** El símbolo de dos puntos (:) se usa en la representación de matrices asociativas; que son asignaciones entre una clave/valor. Esta regla se usa a un nivel superior y ahí se notará el uso de la sangría [26].
- ❒ **Regla 3:** También el símbolo de guion (-) se usa para representar un elemento de la lista dentro de la estructura del documento [26].

1.3.4.2 Usando Yaml en Playbook

Observado en los subcapítulos anteriores todos los playbooks de Ansible son creados en un formato de lista; y dichos elementos de la lista son clave/valor. Y su composición requiere un conocimiento inicial en la sintaxis de YAML que se mira en la Figura 1.6, esto será detallado a continuación [3]:

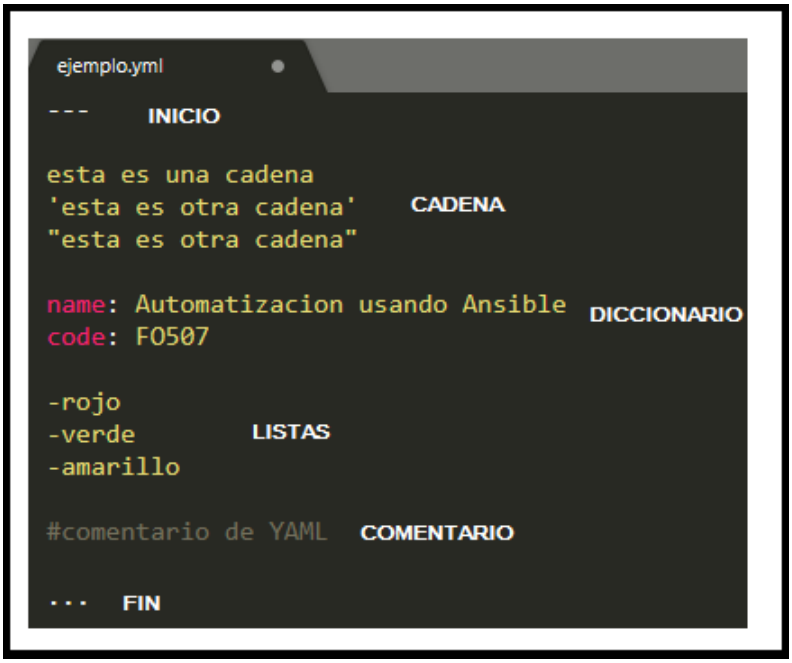

Figura 1.6: YAML en Playbook.

- ❒ **Inicio y final:** Todos los archivos YAML empiezan con un marcador de inicio de archivo necesario, se representa con tres guiones y terminan con un marcador de documento opcional, se representa con tres puntos.
- ❒ **Cadenas:** Se escriben con estilo estándar en línea, se representa con comillas o sin comillas (simples o dobles), se refiere a la notación de bloque.
- ❒ **Diccionarios:** También denominadas matrices asociativas son datos clave / valor usados en YAML. Las claves se diferencian de los valores usando un delimitador que son los dos puntos y un espacio.
- ❒ **Listas de YAML:** Sirven para representar elementos, para poder asignar un elemento en la lista. Se usa un guión para agregar un elemento a la lista de elementos.
- ❒ **Comentarios:** Son utilizados para ayudar al entendimiento de lo que se quiere ejecutar. Aquí los comentarios se representan con el símbolo de numeral al inicio del comentario; y se los puede agregar al final de cualquier comando.

1.3.5 INTRODUCCIÓN BREVE A JINJA2

Jinja se originó de la palabra japonesa templo, Es similar en fonética a la plantilla de la palabra. Jinja es un motor de plantillas, expresivo y extensible. En Jinja2 se tiene unas características importantes que son [22]:

- ❒ Es rápido y compila simultáneamente el código de byte de Python.
- ❒ Esta cuenta con un entorno opcional que cuenta con un espacio aislado.

- ❒ Es fácil de depurar.
- ❒ Es compatible con los playbooks.

Las plantillas construyen dinámicamente al playbook con otros datos que se relacionan, como variables y documento de datos, de manera que permite a Ansible utilizar el lenguaje Jinja2. Y los elementos creados se registran con la extensión ".j2", el nombre antes del punto, se lo registrará con cualquier nombre escogido por el administrador [17].

Los filtros de Jinja2 son simples funciones de Python que seleccionan los argumentos para procesarlos y devolver los resultados. Por ejemplo, considere el comando: jeq | filter ya que, jeq se representa como una variable; Ansible filtrará jeq en Jinja2 como argumento. Este filtro procesará y lo regresará como dato resultante [17].

Generalmente, no se debe modificar los archivos de configuración de Jinja2, mediante la lógica en las plantillas, estas pueden ser de utilidad en los sistemas necesitan las versiones sin modificar el archivo [3].

1.3.5.1 Reglas de Jinja2

- ❒ **Regla 1:** Se usan el símbolo de agrupación (llaves dobles), lo que significa que se realiza una sustitución de la variable por su valor asignado, como se observa en la Figura 1.7.

```
{% for item in access_ports -%}

{% for access in item.ports %}
{% if "-" in access|string %}
interface range {{ access }}
{% else %}
interface {{ access }}
```

Figura 1.7: Reglas de Jinja2.

- ❒ **Regla 2:** Hay uno o varios condicionales dentro de las plantillas este crea una ruta de decisión; esto se escogerá entre algunos bloques de código. Para dicho condicional siempre hay un mínimo de dos caminos, la una es cuando cumple la condición y otra cuando no cumple [15].
- ❒ **Regla 3:** El bucle crea secciones de manera dinámica en los archivos de plantilla, y es de utilidad para cuando necesitamos operar una gran cantidad de elementos desconocidos de la misma manera [15].

1.3.6 GNS3

GNS3 usa librerías de Dynagen para crear una interfaz gráfica (GUI). Las principales funciones es editar el documento de texto y realizar las operaciones del CLI hechas por Dynagen y Dynamips.

En adición se incorpora el poder de simular PCs. GNS3 es un "front end grafic de Dynamips y Dynagen"; que es una herramienta que permiten la emulación de IOS variados. Dynagen brinda una interfaz de comandos más simple a Dynamips, esta es el último punto y tiene la responsabilidad de la emulación de una IOS (Operating System). El usuario realizara su propio fichero para la topología de red para ejecutarlo por Dynagen. GNS3 ayuda con el proceso, creando una interfaz gráfica sencilla como se muestra en la Figura 1.8 que muestra al usuario los detalles de cada configuración que ocurre dentro del escenario de red [19].

La herramienta GNS3 se considera como el lugar donde se junta una variedad de emuladores de distintos sistemas operativos. El importante de estos es Dynamips. Este permite la emulación de enrutadores de Cisco y cuenta con una lista de dispositivos e interfaces genéricos. Hay diferentes emuladores que son compatibles con GNS3, y estos son [27]:

- ❒ Qemu: Emula dispositivos ASA (Cisco Adaptive Security Appliance) de Cisco, enrutadores Juniper, enrutadores Vyatta y hosts Linux.
- ❒ Pemu: Usada específicamente para emular cortafuegos Cisco PIX (Private Internet EXchange).
- ❒ VirtualBox: Emula enrutadores Juniper, enrutadores Vyatta, hosts Linux y hosts de Windows.

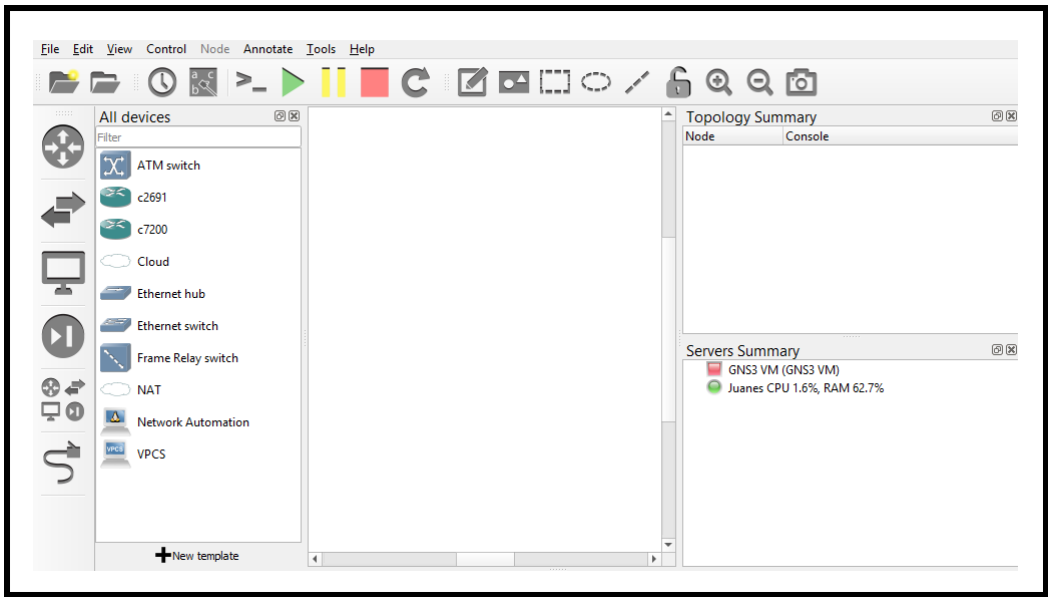

Figura 1.8: Interface Gráfica de GNS3.

Cada procedimiento de un enrutador o cualquier elemento que se efectúe generará una copia de

su sistema operativo que entrará a la pelea por los ciclos de la memoria RAM (Random Access Memory). Se considera que todos los equipos, como routers, switches y cortafuegos estos requieren de una terminal para poder acceder y configurar, tal es el caso de Gnome Terminal, iTerm2, Konsole, PuTTY, SecureCRT, SuperPutty, TeraTerm, Windows Telnet client o incluso Xterm [27].

Existen dos diferentes aplicaciones que complementan pero que no son muy necesarias, aunque frecuentemente trabajan de manera conjunta con GNS3, estas son [27]:

- ❒ Wireshark: herramienta que captura de paquetes de código abierto, para poder analizarlos y así diagnosticar problemas de red.
- ❒ Virtual PC Simulator (VPCS): Simula una gran cantidad (hasta nueve) de PCs que se usan para hacer ping, traceroute y más.

2 METODOLOGÍA

Este capítulo define la manera en que se va a desarrollar el trabajo de integración. Donde se especifican las técnicas e instrumentos a usar, con lo que definirán las diferentes etapas o fases metodológicas que permitirán el desarrollo del trabajo.

Se realiza la justificación sobre la selección de la herramienta de automatización, así como la plataforma de networking y los protocolos a usar en cada topología para realizar el trabajo.

Se aborda diferentes campos de estudio dentro de este capitulo, como el de la elección de tecnologías, elección de la herramienta de automatización, y el software de la herramienta de emulación. También se seleccionan los protocolos a usar en el emulador; en concordancia dentro del emulador se plantean diferentes topologías de manera que sirvan para verificar el correcto funcionamiento de los protocolos escogidos. Posteriormente se verifican los resultados de la emulación.

Conforme lo mencionado, se ha dedicido utilizar 4 topologías enumeradas secuencialmente del 1 al 4. Las topologías usan equipos Cisco, y cada una de ella usan distintas configuraciones como VLANS, BGP y MPLS respectivamente. En la cuarta topología se trabaja con un Firewall (ASA: Adaptive Security Appliance).

2.1 REQUERIMIENTOS PARA LA SIMULACIÓN

Para realizar las simulaciones respecto a automatización de redes se considera el proceso expuesto en la Figura 2.1. Los requerimientos generales para llevar a cabo la simulación se indican a continuación:

- ❒ El equipo donde se realizará la simulación debe cumplir varios requisitos, como tener una RAM de 8 GB, una maquina virtual de GNS3, el sistema operativo Windows 10 de 64 bits, procesador mínimo de Core i7.
- ❒ El programa de emulación trabajará en su versión actual, dicho programa debe trabajar de manera conjunta con la máquina virtual de GNS3.
- ❒ Los equipos como switches (capa 2 y 3), routers, y otros deben tener unas características físicas determinadas (por ejemplo, número de puertos) para trabajar en cada topología que se

presenta en detalle en la siguientes secciones.

- ❒ La herramienta de automatización es esencial para cumplir con el trabajo. La siguiente sección abordara detalles de esta herramienta.

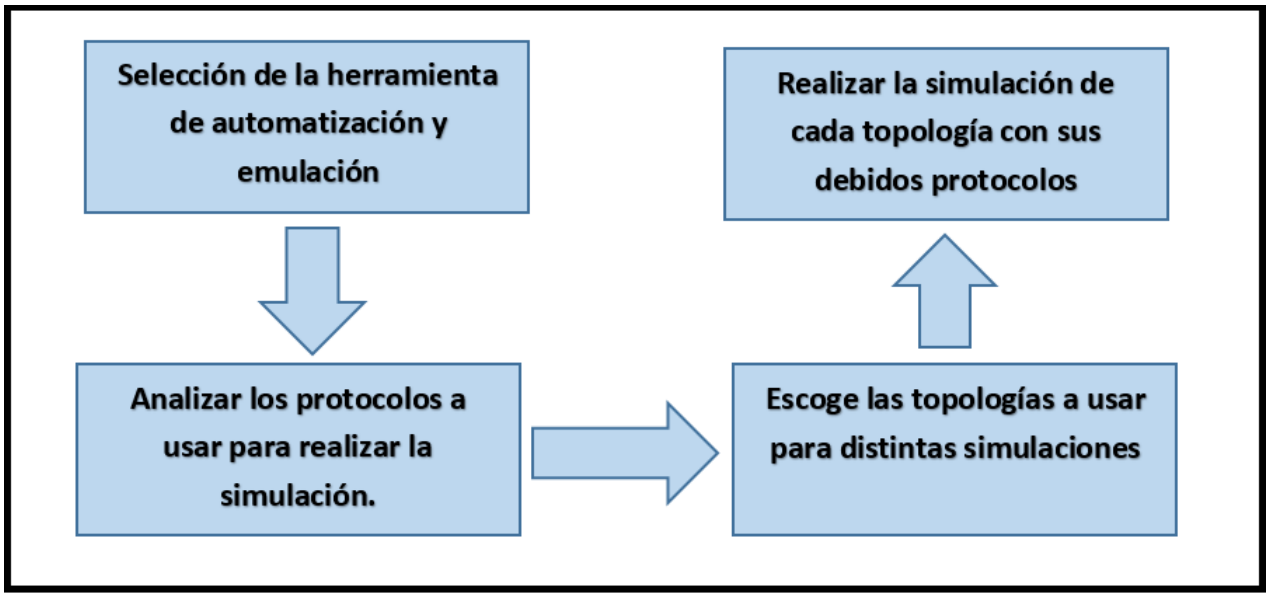

Figura 2.1: Proceso a Realizar para Simulacion.

2.2 HERRAMIENTA DE AUTOMATIZACIÓN

Anteriormente se familiarizó con las diferentes características que tienen las herramientas de automatización que se encuentran a disposición en el mercado, con esas características se realizará una comparación de los parámetros y así determinar cuál de ellas ofrece mejores prestaciones al instante de realizar el presente trabajo de integración curricular.

Para realizar el proceso de comparar las herramientas de automatización, se ha usado la escala de Likert [1], para poder medir variables cualitativas en forma cuantitativa, y lograr altos niveles de confiabilidad. Esta comparación se basará en la información expuesta en la Tabla 2.1, se procederá a la elección de la herramienta de automatización adecuada para el trabajo de integración mediante la asignación de valores de 1 a 3 (3: excelente, 2: normal y 1: deficiente). Asignando los valores a las variables cualitativas dentro de las herramientas de automatización que se están comparando[10].

[1] Se usa este tipo de escala que mide actitudes, es decir, que se emplea para medir el grado en que se da una actitud o disposición de lo encuestado. El objetivo es agrupar numéricamente los datos que se expresen en forma verbal, para poder operar con ellos, como si se tratara de datos cuantitativos para poder analizarlos correctamente. [18]

Tabla 2.1: Características de dos Herramientas de Automatización, basado en [10].

PARÁMETROS	ANSIBLE	NETMIKO
Escalabilidad	Pequeños-medios-grandes (3)	Pequeños-medios-grandes (3)
Facilidad en la Configuración	Sin agentes (SSH) (3)	Sin agentes (SSH)(3)
Disponibilidad	Instancia primaria y secundaria (3)	Gran disponibilidad (3)
Interoperabilidad	Ansible también admite máquinas Windows, pero el servidor Ansible esta en la máquina Linux /Unix (3)	Netmiko se admite en Windows mediante una subrutina, e incluir una maquina virual Linux (3)
Tipo de Lenguaje	Declarativo (3)	Declarativo (3)
Agentes de Traducción	Push (3)	Push (3)
Documentación	documentación, referencias, tutoriales, presentaciones, vídeos (3)	documentación, referencias, tutoriales, presentaciones, vídeos (3)
Lenguaje de Desarrollo	Python (3)	Python (3)
Comunicación	SSH (3)	SSH (3)
Repositorio de la Comunidad	Ansible en Github (3)	Ansible en Developer (3)
Interfaces de Control	Playbooks: archivos de texto JSON (JavaScript Object Notation), Jinja2, YAML (3)	Playbooks: archivos de texto Jinja, YAML (2)

La Tabla 2.1 muestra el análisis referente a las características de cada una de las herramientas de automatización mencionadas; determinando que ANSIBLE brinda una variada gama de parámetros, que son semejantes a NETMIKO, por ello cualquiera de las dos herramientas se usará, esto influye de manera directa al realizar la emulación de manera satisfactoria.

Se puede observar ya que los playbooks se basan en YAML (usados por Netmiko y Ansible), ya que son simples de leer, entender y mantener; lo que da por resultado un excelente aprendizaje [6].

2.3 HERRAMIENTA DE EMULACIÓN

Para la herramienta de emulación, en este caso se escogió la mencionada en el Capítulo 1. En la tabla 2.2 se presentan las características mas relevantes de esta herramienta.

Tabla 2.2: Características de GNS3 [27].

PARÁMETROS	GNS3
Escalabilidad	alto
Facilidad de Instalación	fácil
Procesador	2 núcleos
Memoria	4 GB u 8 GB de RAM
Almacenamiento	1 GB
Red	LAN, WAN
Sistema Operativo	Windows 8, 10
Tamaño de Aplicación	377.5 MB (Máquina virtual)
Licencia	Gratis
Interoperabilidad	Alta
Administración	Interfaz amigable al usuario

GNS3 ofrece un consumo de hardware mínimo referente a la Memoria RAM y un mejor procesamiento por parte de la herramienta de emulación. Una de las características más relevantes que influyó al escoger GNS3 es por que, la herramienta ha sido utilizada en la carrera universitaria y se está habituado a interactuar con la misma.

2.4 PROTOCOLOS A USAR EN LA SIMULACION

Para las 4 topologías que se usarán en las simulaciones, que se tienen planteadas, se usarán diferentes protocolos como se los muestra:

- ❒ Virtual Local Area Network
- ❒ Border Gateway Protocol
- ❒ Multiprotocol Label Switching

2.4.1 VLAN (Virtual Local Area Network)

Para una topología se usará VLANs, debido a su fácil administración, seguridad y confidencialidad. Un usuario será limitado al uso que no se ah autorizado, no permitiendo ingresar a la información, recursos y documentos que no sean de su propiedad. Las VLANs son bastantes usadas en redes institucionales y corporativas, en el aspecto público y privado, ya que trabajan con jerarquías; lo que se puede dividir en varios grupos de trabajo. Las VLANs son importantes para poder relacionarlas

con las redes MPLS. Por todo lo mencionado se escogió a las VLANS como el protocolo a usar[24].

2.4.2 BGP (Border Gateway Protocol)

BGP es otro protocolo a usar en las simulaciones, este es complejo, se usa por medio del Internet y dentro de varias empresas. La función del protocolo es para comunicar varios sistemas autónomos (AS)[2], este protocolo fue escogido debido a que existen en Ecuador varios proveedores de Internet que usan el protocolo para la comunicación entre sí [24].

2.4.3 MPLS (Multiprotocol Label Switching)

En otra topología se utilizará MPLS, que posibilita a los operadores ofrecer múltiples servicios, y las Redes Privadas Virtuales (MPLS/VPN), es un método eficaz para enrutar distintos tipos de tráfico. Se seleccionó el protocolo debido a que los SP (Service Providers) del Ecuador lo usan para la trasmisión de datos [24].

2.5 CONFIGURACIONES PARA LAS DIFERENTES TOPOLOGÍAS

2.5.1 TOPOLOGÍA DE UNA RED

En esta sección se describen los elementos principales de cada una de las topologías de red a usar con la herramienta de automatización, tal como se muestra en la Figura 2.2.

En un nodo central se encuentran todos los documentos de configuración escritos en YAML y Jinja2. Además, en este nodo se encuentra anidado Ansible que se encarga de automatizar toda la red, también hay un servidor (representado por la nube) donde se guardarán todos los archivos que se necesiten ejecutar para automatización de red.

Dentro del nodo secundario se encuentran varios equipos de red que van a tomar las configuraciones del nodo central. Los dispositivos a usar son routers, switches (capa 2 y 3) y firewall.

[2] Un sistema autónomo es una red o grupo de redes muy grande con una política de enrutamiento. A cada AS se le asigna un único ASN (número que identifica al AS) [5]

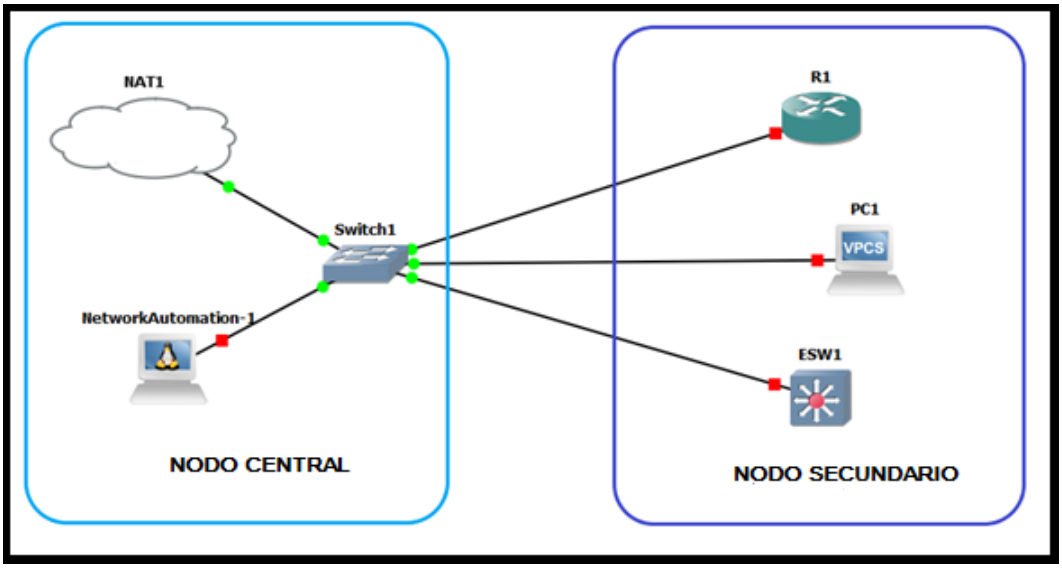

Figura 2.2: Topología de Red general con Ansible.

2.5.2 COMANDOS USADOS EN EL NODO CENTRAL

Para la automatización de red se utiliza ansible, que pertenece a la distribución de Linux Fedora, heredada de Red Hat. Utiliza distintos comandos para crear, eliminar y visualizar los ficheros, directorios y scripts (playbooks y plantillas), mismos que se describe a continuación[15].

- **mkdir:** Viene de make directory (crear directorio), el comando crea un directorio teniendo en cuenta la ubicación actual del directorio.
- **nano:** Es el más básico editor de texto, con un uso sencillo. El comando es nativo en algunos sistemas como Ubuntu, teniendo a otros editores de texto como emacs o vi.
- **cat:** Es referente a concatenar (mostrar), siendo de mucha utilidad para visualizar el contenido de un documento de texto realizado con el comando anterior.
- **rm:** Significa remover, el comando es necesario para borrar los archivos (.j2 y .yml) o un directorio.
- **cd:** Referido a change directory o cambio directorio, este comando es necesario para acceder a un directorio distinto, y trabajar en el directorio. Este comando tiene distintas opciones:
  - cd .. (con dos puntos) para ir un directorio hacia arriba.
  - cd para ir directamente a la carpeta de inicio.
  - cd- (con un guión) para ir al directorio anterior.
- **ls:** Este comando se usa para listar el contenido de un directorio, para encontrar un fichero, archivo o directorio.

2.5.3 CONFIGURACIÓN EN EL NODO CENTRAL

Para la configuración que se realizará en el nodo principal se ejecutará los comandos que se expondrán a continuación. La primera configuración que se observará es el comando nano /etc/network/interfaces, sirve para activar la interfaz eth0 de Ansible (nodo central o principal) para obtener una dirección IP. Al editar el script, se habilitará las siguientes lineas: auto eth0 y iface eth0 inet dhcp, quitando el símbolo numeral (comentario), como se muestra en la Figura 2.3.

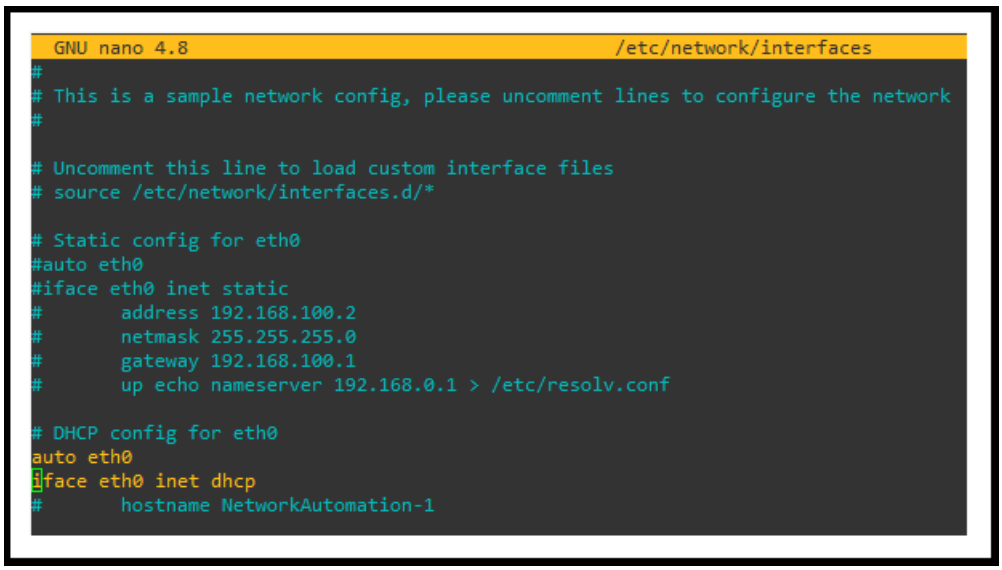

Figura 2.3: Configuración de la Maquina Central de Ansible con Direccionamiento IP.

Para que la configuración surja efecto se debe apagar y luego encender la máquina central que tiene Ansible. Al habilitar el DHCP en la interfaz eth0 a través de NAT (Network Address Translation) se asigna una dirección IP de manera automática como se muestra en la Figura 2.4. Cuando se le asigna una nueva dirección IP a cualquier dispositivo del nodo secundario se deberá reiniciarla la máquina para que no exista ningún problema al instante de ejecutar los playbooks.

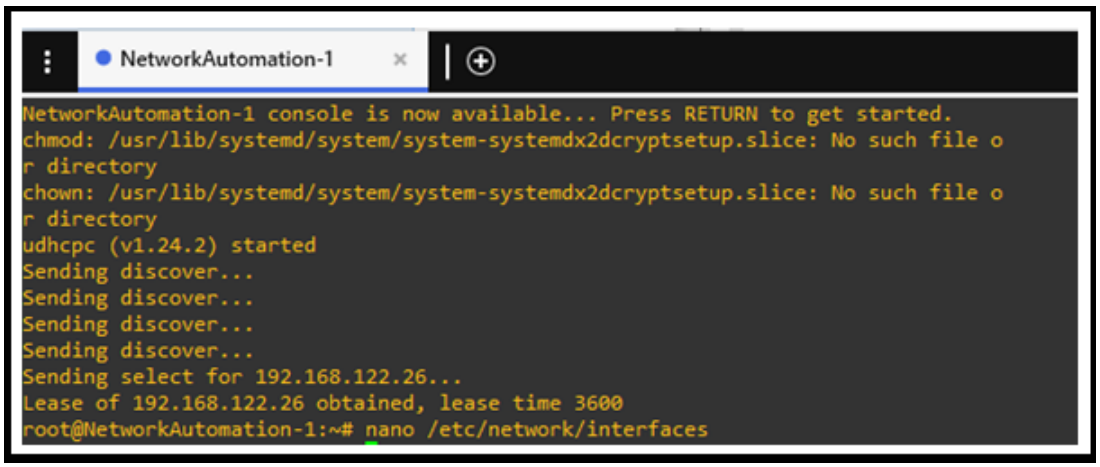

Figura 2.4: Inicialización de la Maquina Central y su Dirección IP.

Para la segunda configuración, se agregan los nombres de los dispositivos de red y también las direcciones IP, que se encuentran en el nodo secundario, para ello se usa el la linea de comando nano /etc/hosts, como se visualiza en la Figura 2.5. Este procedimiento sirve para modificar el fichero hosts en la maquina central (Ansible). En este fichero se anclarán las direcciones IP que se necesiten en el nodo secundario.

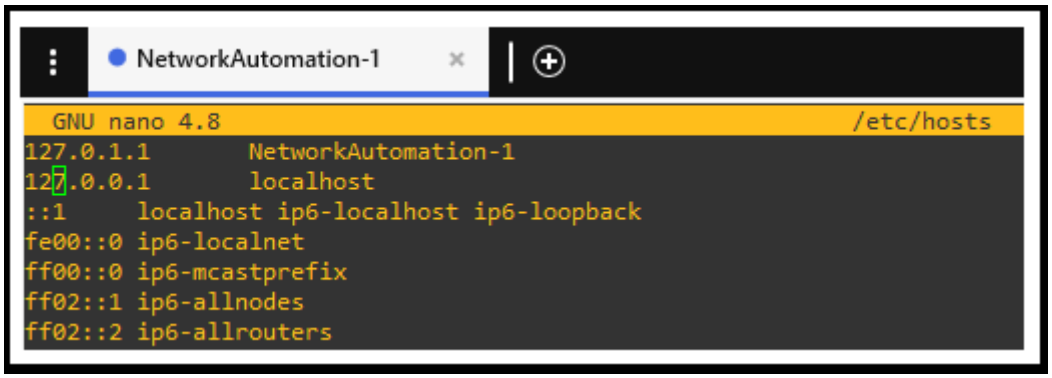

Figura 2.5: Listado de Direcciones IP de los Dispositivos de red, dentro del nodo central.

2.5.4 CONFIGURACIÓN EN EL NODO SECUNDARIO

En la Figura 2.6 se muestra varios comandos iniciales, en la línea 3 se asigna un nombre de usuario y habilita la contraseña, de la linea 4 a la 11, se ejecutan los pasos necesarios para la activación de SSH en un equipo de red dentro del nodo secundario, de la linea 13 en adelante se activan las interfaces que use el equipo de red; estas configuraciones se usan en un equipo Cisco.

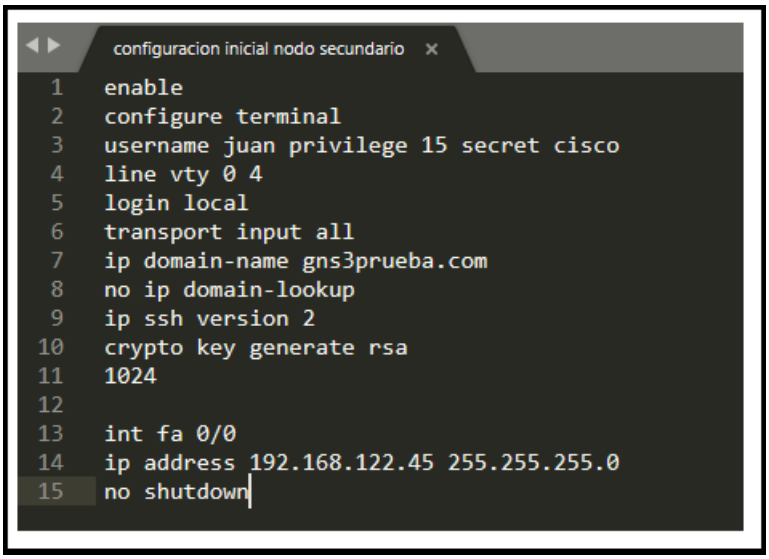

Figura 2.6: Configuración Inicial para Equipos CISCO.

El nodo secundario, se conoce también como nodo de clientes, se toma en cuenta por ejecutar y recibir las configuraciones realizadas en el nodo central. Existen condiciones necesarias para que

funcione de manera correcta el nodo. Estas configuraciones iniciales corresponden a una dirección IP y habilitación de SSH asociado a un usuario y una contraseña. En el nodo secundario es posible aumentar el número de usuarios a los dispositivos que se encuentren dentro del nodo, para fines prácticos del presente trabajo se considera a un solo usuario. Los códigos usados en la configuración inicial varían, teniendo en cuenta la marca de los equipos de red.

2.6 IMPLEMENTACIÓN DE 4 DIFERENTES TOPOLOGÍAS

En la presente sección se explica cómo se realizará la implementación en las distintas topologías a considerarse en el Trabajo de Integración Curricular.

2.7 PRIMERA TOPOLOGÍA - LAN Y VLAN

Para esta la primera topología se usa un switch de capa 3 de marca Cisco que posee distintos adaptadores como: Ethernet, FastEthernet y GigaEthernet, tal como se muestra en la Figura 2.7 se trabajará con los FastEthernet.

Para esta topología se ha habilitado el SSH para agregar el usuario y contraseña. Se ha asignado dirección IP en la interfaz de cada uno de los switches capa 3: D1, D2, A1 y A2 que son capa 3; para lograr configurar todos los equipos.

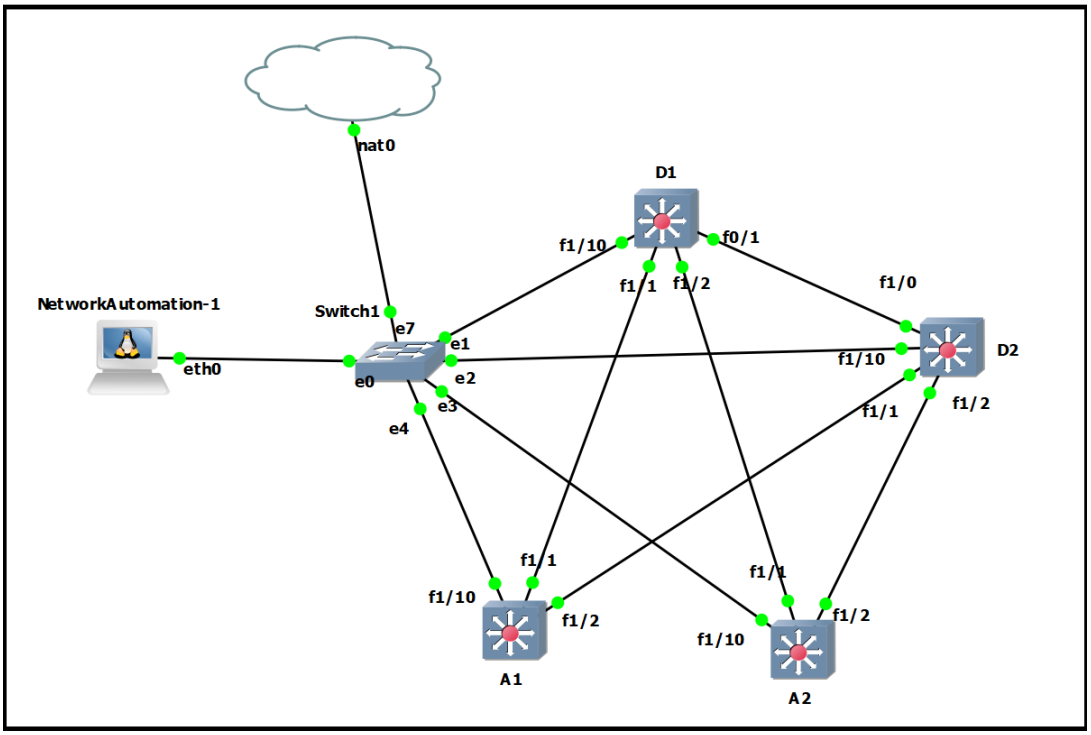

Figura 2.7: Topología seleccionada para el Protocolo LAN y VLAN.

Para la topología vista en la Figura 2.7, se generan las distintas VLAN, también se asignan direcciones IP con nombres claves. De igual manera se crean los playbooks que están en el servidor de network automation (nodo central), dentro del que se crean archivos Jinja2 y se añadirá en la lista de hosts las direcciones IP de cada switch de capa 3 . Estos archivos serán usados para la ejecución de los playbooks antes mencionados, todo ello se los visualizara en la Figura 2.8.

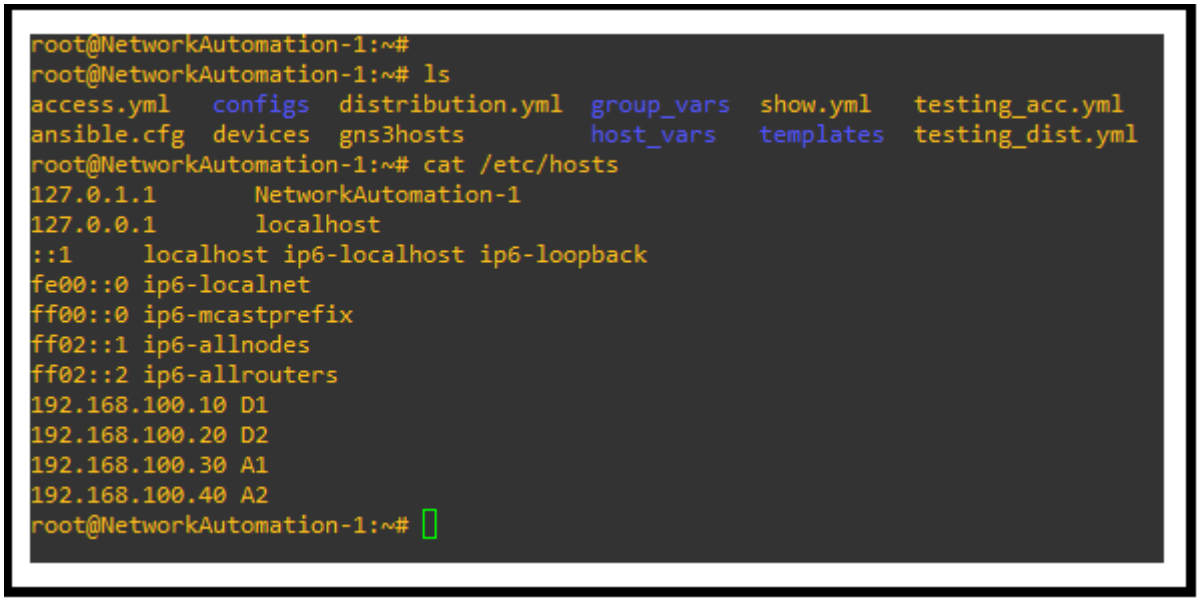

Figura 2.8: Listado de Playbooks de la Primera Topología.

En la Figura 2.9 se observa como se compone el playbook referente al Switch de Acceso 1 (A1), y este es semejante al playbook del Switch de Acceso 2.

Se observa que de la línea 2 a la 7 se realiza la asignación del nombre, número de identificación y la dirección de red a la que pertenece cada VLAN que se relaciona con el switch. De la línea 10 a la 19 se realiza la activación de los puertos de acceso, con su respectivo nombre (depende del administrador) y la interfaz del puerto en el Switch. El resto de las líneas del código, son para activar los puertos troncales que van dirigidos a los switches de distribución (D1 Y D2), de igual manera como lo anterior asignando interfaces a los puertos en el Switch.

Lo mismo ocurrirá con Switch de Acceso 2, ya que cada uno tiene distintos nombres, número de identificación, VLANs, y sus direcciones IP; las que se agregan sin que haya redundancias.

```
---
vlan_01:
  accounting: { id: '30', subnet: 10.1.30.0/24, root: D1_dist }
  management: { id: '31',  subnet: 10.1.31.0/24, root: D1_dist }
  voice01: { id: '10', subnet: 10.1.10.0/24, root: D1_dist }

vlans: "{{ vlan_01|combine(vlan_02) }}"


access_ports:
  - ports:
      - 'Gi1/0 - 1'
      - 'Gi2/0'
    data: accounting
    voice: voice01
  - ports:
      - 'Gi1/2 - 3'
    data: management
    voice: voice01

trunk_ports:
  - { group: 1, ports: ['Gi0/0 - 1'], description: -> D1_dist }
  - { group: 2, ports: ['Gi0/2 - 3'], description: -> D2_dist }

allowed_vlans: "{{ vlans.values()|map(attribute='id')|list|join(',') }}"
```

Figura 2.9: Switch de Acceso de la Primera Topología.

La Figura 2.10 muestra el playbook a usar para la configuración del Switch de Distribución 1, de igual manera se lo realiza el mismo procedimiento con el Switch de Distribución 2.

Se muestra que de la línea 3 a la 7, se habilita los puertos troncales, mediante las interfaces que se usan en el código van dirigidas a los Switches de Acceso (A1 Y A2). En el resto de las líneas de código se muestra como se asigna una dirección IP a las interfaces, y también la descripción de la mismas.

```
---

trunk_ports:
  - { group:1, ports: ['f1/1'], description: -> A1 }
  - { group:2, ports: ['f1/2'], description: -> A2 }

svi_id: 1

interfaces:
    Po40: { ip: 172.168.100.1/30, routed: true, ports: [f0/1], description: ptp link to D2 }

loopbacks:
lo0: {ip: 10.250.0.20/32 }

```

Figura 2.10: Switch de Distribución de la Primera Topología.

La Figura 2.12 se trata del playbook a usar para la configuración del switch de acceso. Tal como se observa en las primeras líneas de código se asignan nombres y referencias del tipo de conexión que se ejecuta.

A partir de la línea 7 del código, el nombre de todo el procedimiento es designado como tasks(tareas). Para el siguiente procedimiento se trabaja con distintos archivos .j2 (Jinja), cada uno ejecuta distin-

tos procesos de configuración, estos son representados por nombres como se muestra en la Figura 2.12.

Estos procesos que se encuentran dentro del directorio templates, y son: Vlans config, Spanning-tree config, Default interface config, Access port config, Trunk port config; como se visualiza en la Figura 2.11.

```
~# cd templates/
~/templates# ls
~/templates# nano access-ports.j2
~/templates# nano default-interface.j2
~/templates# nano hsrp.j2
~/templates# nano interface.j2
~/templates# nano stp.j2
~/templates# nano svi.j2
~/templates# nano trunk-ports.j2
~/templates# nano vlan.j2
```

Figura 2.11: Creación del directorio Templates y archivos Jinja

```
1  ---
2  - name: access switch configs
3    hosts: acc
4    gather_facts: no
5    connection: local
6
7    tasks:
8
9      - name: vlans config
10       ios_config:
11         src: templates/vlan.j2
12       tags: vlan
13
14     - name: spanning-tree config
15       ios_config:
16         src: stp.j2
17       tags: stp
```

Figura 2.12: Configuración Switch de Acceso de la Primera Topología.

Al igual que con los switch de acceso, el siguiente playbook se muestra en la Figura 2.13 y hace referencia a la configuración del switch de distribución. Al igual que en el anterior playbook las primeras líneas de código son usadas para nombrarlo y especificar la forma de conexión a ejecutarse.

```
1  ---
2  - name: distribution switch configs
3    hosts: dist
4    gather_facts: no
5    connection: local
6    serial: 1
7
```

Figura 2.13: Configuración Switch de Distribución 1 de la Primera Topología.

La Figura 2.14 muestra una parte del playbook, ahí se procede a modificar lo referente a LAN virtual, se realiza distintos procesos de configuración.

```
10  # lan switching
11      - name: vlans config
12        ios_config:
13          src: templates/vlan.j2
14        tags: vlan
15
16      - name: spanning-tree config
17        ios_config:
18          src: templates/stp.j2
19        tags: stp
```

Figura 2.14: Configuración Switch de Distribución 2 de la Primera topología.

Para la siguiente parte del playbook mostrado en la Figura 2.15, se realiza la configuración de las interfaces físicas y virtuales. Como en los anteriores párrafos la configuración se realiza a través de los archivos con extensión .j2.

```
31  # interfaces
32      - name: switch vlan interface config
33        ios_config:
34          src: templates/svi.j2
35        tags: svi
36
37      - name: interface config
38        ios_config:
39          src: templates/interface.j2
40        tags: interface
```

Figura 2.15: Configuración Switch de Distribución 3 de la Primera Topología.

Para terminar se agrega un playbook que se observa en la Figura 2.16. En este playbook se ejecutan varios comandos para visualizar las interfaces físicas y virtuales, también se especifica como acceder a los equipos por medio de SSH.

```
7   tasks:
8     - name: Ejecutar multiples comandos IOS
9       ios_command:
10        commands:
11          - show vlan brief
12          - show interface
13      register: print_output
14    - debug: var=print_output.stdout_lines
15
16  vars:
17    ansible_user: esteban
18    ansible_ssh_pass: este123
19    authorize: yes
20    auth_pass: este123
```

Figura 2.16: Ejecución de Comandos en la Primera Topología.

2.8 SEGUNDA TOPOLOGÍA: BGP

Para la siguiente topología se utilizan 4 routers R1, R2, R3 y R4. En estos dispositivos se realiza la configuración inicial considerando la conexión SSH, el establecimiento de usuario y contraseña, las direcciones IP y la asignación de interfaces. Como se observa en la Figura 2.17, se utiliza 4 routers CISCO.

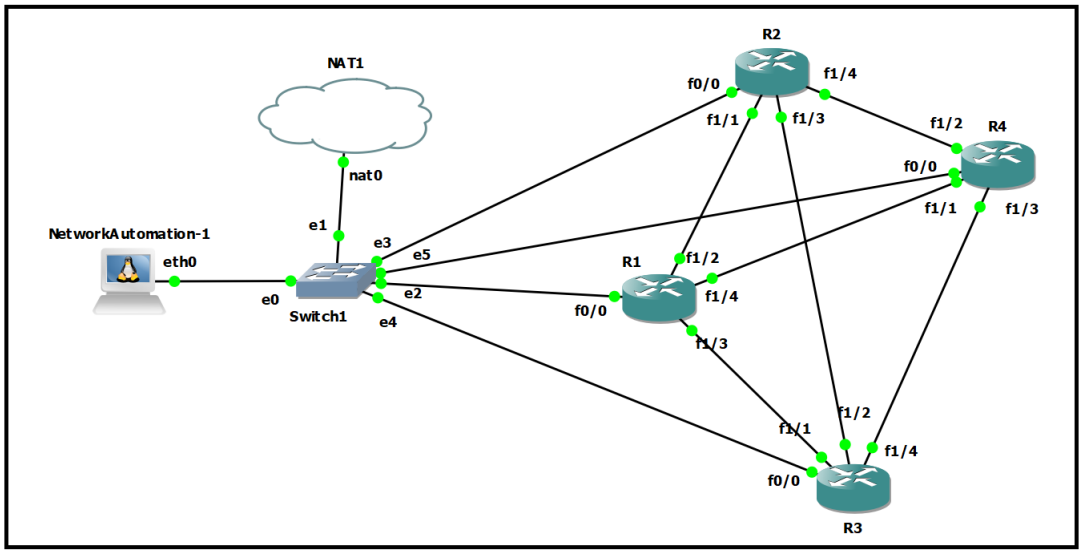

Figura 2.17: Topología seleccionada para el protocolo BGP.

En el nodo central se crean directorios, ficheros, playbook y plantillas; lo necesario para trabajar en la topología. Se toma en cuenta el playbook de los routers para configurar las interfaces (de la linea 6 a la 20)y el protocolo de enrutamiento a usar. Se asignan direcciones IP para las interfaces que se conectan a cada router, y el enrutamiento (de la linea 21 en adelante) funciona a partir de la asignación de direcciones IP, como se visualiza en la Figura 2.18 y Figura 2.19. Este procedimiento

se repite en cada router cambiando los parámetros correspondientes.

```
---

hostname: R1
domain_name: gns3prueba.com

loopback:
  address: 10.255.0.1
  mask: 255.255.255.255
```

Figura 2.18: Configuración de Routers de la Segunda Topología.

```
interfaces:
  1/2:
    alias: connection R2
    address: 10.0.255.1
    mask: 255.255.255.252

  1/3:
    alias: connection R3
    address: 10.0.255.5
    mask: 255.255.255.252

bgp:
  asn: 65001
  neighbor:
    - {address: 10.0.255.2, remote_as: 65000}
    - {address: 10.0.255.6, remote_as: 65000}
  networks:
    - {network: 10.0.255.0, mask: 255.255.255.252}
    - {network: 10.0.255.4, mask: 255.255.255.252}
    - {network: 10.255.0.1, mask: 255.255.255.255}
  maxpath: 2
```

Figura 2.19: Configuración de Routers de la Segunda Topología.

En esta topología se trabaja con varios playbooks. En la Figura 2.20 se muestra la designación de hostname y el dominio, esto varía dependiendo del router en el que se aplique.

```
1  ---
2
3  - name: set dns and hostname
4    ios_system:
5      hostname: "{{ hostname }}"
6      domain_name: "{{ domain_name }}"
7      lookup_enabled: 'no'
```

Figura 2.20: Configuración DNS y Hostname de la Segunda Topología.

Para la configuración de las interfaces, el playbook trabaja con un archivo Jinja (de la linea 4 a la 6), y de igual manera activando las interfaces que se asignan a cada router (de la linea 8 en adelante), como se visualiza en la Figura 2.21.

```
1  ---
2
3  - name: write interfaces config
4    ios_config:
5      src: "templates/interfaces.j2"
6    register: result
7
8  - name: enable interfaces
9    ios_config:
10     parents: "interface Ethernet{{ item.0 }}"
11     lines: "no shutdown"
12     match: none
13   when: result.changed
14   with_items:
15     - "{{ interfaces.items() }}"
```

Figura 2.21: Configuración de las Interfaces en la Segunda Topología.

En el playbook nombrado site.yml se trabaja en la configuración de ruteo que de la misma manera que el playbook anterior es un archivo de extensión .j2. Lo mismo ocurre en la configuración de SNMP [3] e ICMP [4]. También se activan las interfaces asignadas a cada router. Estos 3 playbooks (check icmp, site y check snmp) son semejantes al visto en la Figura 2.21, con distintos parámetros en cada playbook.

2.9 TERCERA TOPOLOGÍA: MPLS

Para la topología se usa routers de marca Cisco. Se configuran 10 routers dentro de la topología de red (4 routers de borde proveedor, 4 routers de borde cliente y 2 routers proveedor), esto se observa en la Figura 2.22.

[3] SNMP (Simple Network Management Protocol) es un protocolo del nivel de aplicación que proporciona un formato de mensajes para el intercambio de información entre gestores y agentes; que facilita el intercambio de información de administración entre dispositivos de red [11].

[4] ICMP (Internet Control Message Protocol) es un protocolo autónomo aun cuando los diferentes mensajes están incluidos en paquetes IP tradicionales [2].

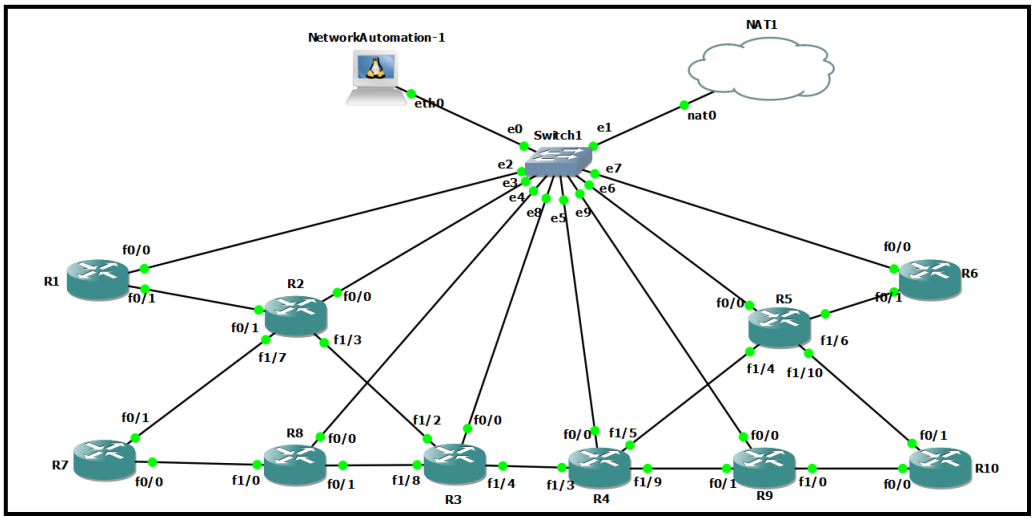

Figura 2.22: Topología seleccionada para el protocolo MPLS.

Conforme lo mencionado, se deben configurar los distintos routers de la topología. El playbook que se realiza en el router ce (ce: borde de cliente) se muestra en la Figura 2.23. Un playbook similar se encuentra en cada uno de los 4 routes de borde de cliente. Como se indica en el playbook de la Figura 2.23, se asigna la dirección IP al loopback (de la linea 6 a la 8), se asigna una dirección IP y una VLAN a la interfaz (de la linea 11 a la 16) que se conecta al router pe (pe: borde de proveedor). Al trabajar con la configuración de protocolo BGP se deben asignar vecinos que en este caso serian los routers de proveedor (P1 y P2) con los que se comunica (de la linea 18 en adelante).

```
1  ---
2
3  hostname: ce_banco
4  domain_name: gns3prueba.com
5
6  celoopback:
7    address: 192.255.0.1
8    mask: 255.255.255.255
9
10
11 interfaces:
12   1/1.200:
13     alias: connection pe_paris
14     vlan: 200
15     address: 192.168.200.2
16     mask: 255.255.255.252
17
18 bgp:
19   asn: 65113
20   neighbor:
21     - {address: 192.168.200.1, remote_as: 1}
```

Figura 2.23: Configuración Router de Borde de Cliente en la Tercera Topología.

En la Figura 2.24, se muestra un playbook que representa a los routers pe. La configuración de este playbook es similar en los otros 3 routers pe, y también es semejante al del router ce. Como en los

casos anteriores se debe configurar la dirección IP del loopback y IP de los clientes. Otra parte del código se refiere a la configuración de las interfaces, en donde se asigna una dirección IP y una VLAN a cada interfaz que se este usando en el router.

En la Figura 2.26 y la Figura 2.25 se muestra el playbook de los routers de borde (P1 y P2), que son semejantes a los playbooks de los routers pe y ce, con la diferencia de que se añade la configuración del OSPF (Open Shortest Path First). También se configura lo referente a BGP, añadiendo las direcciones IP de todos los vecinos al que se conectan tanto el router P1 y P2.

```
1  ---
2
3  hostname: pe_londres
4  domain_name: gns3prueba.com
5
6  peloopback:
7    address: 10.255.0.3
8    mask: 255.255.255.255
9
10 customers:
11   acme:
12     rd: 10.255.0.3:10
13     rt: "10:10"
14
28 interfaces:
29   1/1.78:
30     alias: connection p2
31     vlan: 78
32     address: 10.0.253.1
33     mask: 255.255.255.252
```

Figura 2.24: Configuración Router de Borde de Proveedor en la Tercera Topología.

```
1  ---
2
3  hostname: P1
4  domain_name: gns3prueba.com
5
6  peloopback:
7    address: 10.255.0.100
8    mask: 255.255.255.255
9
10
11 interfaces:
12   1/1.122:
13     alias: connection p2
14     vlan: 122
15     address: 10.0.122.1
16     mask: 255.255.255.252
17
```

Figura 2.25: Configuración Router de Borde de Proveedor en la Tercera Topología.

```
34  ospf:
35    proc: 1
36    ldp: mpls ldp autoconfig
37    networks:
38     - {network: 0.0.0.0, mask: 255.255.255.255, area: 0}
39
40
41  bgp:
42    asn: 1
43    neighbor:
44      - {address: 10.255.0.1, remote_as: 1}
45      - {address: 10.255.0.2, remote_as: 1}
46      - {address: 10.255.0.3, remote_as: 1}
47      - {address: 10.255.0.4, remote_as: 1}
48    ups:
49      - {address: 10.255.0.1, loopback: Loopback0}
50      - {address: 10.255.0.2, loopback: Loopback0}
51      - {address: 10.255.0.3, loopback: Loopback0}
52      - {address: 10.255.0.4, loopback: Loopback0}
```

Figura 2.26: Configuración de OSPF en la Tercera Topología.

2.10 CUARTA TOPOLOGÍA: FIREWALL

Para la ultima topología a tratar se usan dos routers y un firewall de marca CISCO, estos equipos se conectan entre sí como se indica en la Figura 2.27.

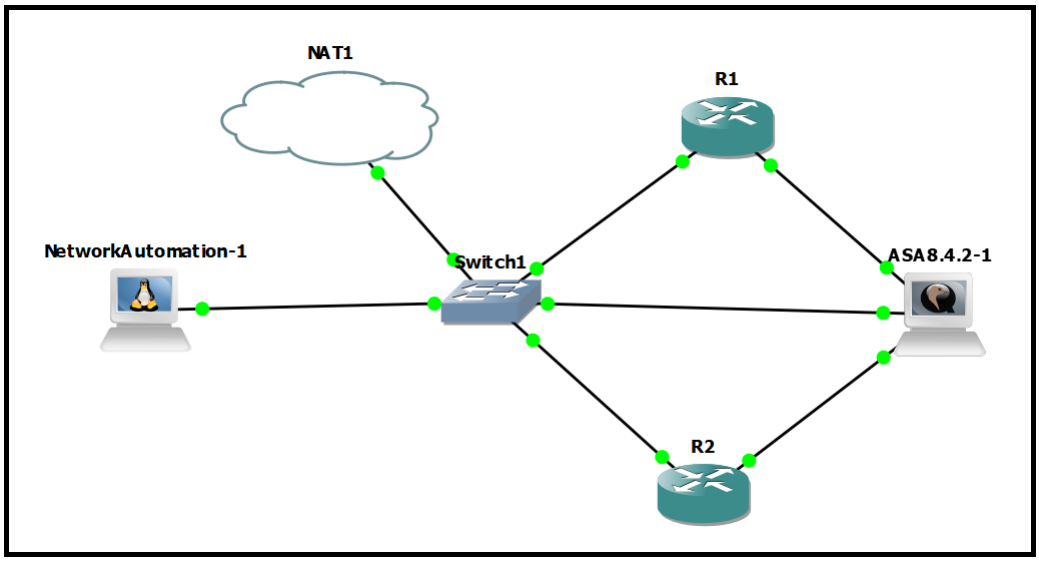

Figura 2.27: Topología seleccionada para la ejecución del Firewall.

Al igual que en las otras topologías se añaden los equipos del nodo secundario a los hosts que se encuentran dentro de la Network Automation o nodo central mediante la siguiente linea de comando: cat gns3hosts /etc/hosts. La ejecución de este comando se muestran Figura 2.28.

```
root@NetworkAutomation-1:~# cat /etc/hosts
127.0.1.1       NetworkAutomation-1
127.0.0.1       localhost
::1     localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
192.168.122.41 asav1
192.168.122.42 rtr-1
192.168.122.43 rtr-2
```

Figura 2.28: Hosts del Nodo Secundario.

Luego de agregar los hosts, se crean de los ficheros, directorios, playbooks y plantillas necesarios en el nodo primario (Network Automation). Estos archivos son necesarios para la ejecución del network automation y se crea el playbook para el firewall. En este playbook se activan las interfaces, agregando direcciones IP, el nivel de seguridad y a donde va dirigida cada interfaz, tal como se observa en la Figura 2.29.

```
root@NetworkAutomation-1:~# ls
ansible.cfg           config_firewall.yml  devices    group_vars  roles
asav1_check_icmp.yml  config_routers.yml   gns3hosts  host_vars
root@NetworkAutomation-1:~# cd host_vars/
root@NetworkAutomation-1:~/host_vars# cat asav1.yml
---

hostname: asav1
domain_name: gns3prueba.com

interfaces:
  0/0:
    alias: connection rtr-1 inside
    nameif: inside
    security_level: 100
    address: 10.0.255.1
    mask: 255.255.255.0

  0/1:
    alias: connection rtr-2 outside
    nameif: outside
    security_level: 0
    address: 217.100.100.1
    mask: 255.255.255.0

routes:
  - route outside 0.0.0.0 0.0.0.0 217.100.100.254 1
```

Figura 2.29: Plantillas, Directorios y Playbooks.

De igual manera, como se creó el playbook para el firewall se deben crear distintos playbooks como: objetos, listas de acceso y políticas de servicio, se ubican en el directorio group vars mostrado en la Figura 2.30. También mostrando el playbook denominado access list creado para asignar las listas

de acceso que se van a usar en el firewall. En el playbook denominado access list se muestra que existen dos elementos en la lista de acceso, cada uno dando permiso de entrada y salida mediante tcp e icmp. El resto de playbooks que se encuentran en el directorio group vars realizan distintos procesos. El playbook object groups puede crear una lista de servidores y subredes, si se las necesita. El playbook objects asigna direcciones IP a cada servidor y subred creado por el playbook object group. El playbook policy framework crea políticas a cumplir como: un tamaño máximo de mensajes y designación del tamaño del trafico(alto, medio, bajo).

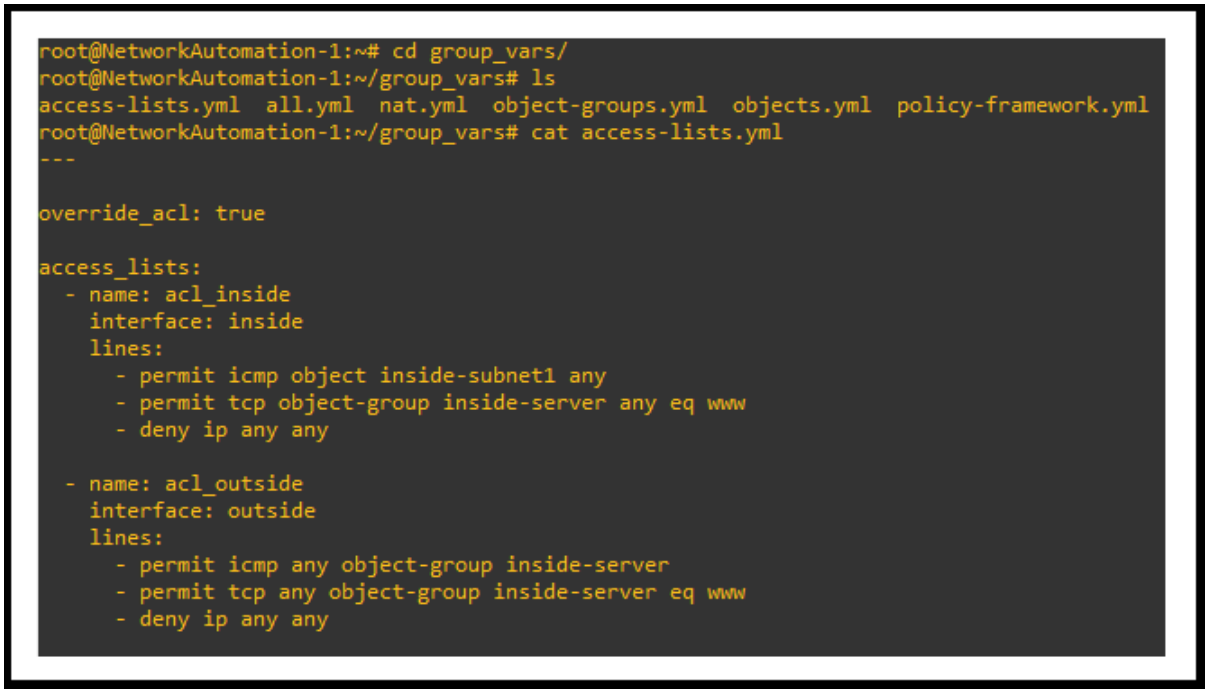

Figura 2.30: Plantillas, Directorios y Playbooks 2.

3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

3.1 OBSERVACIÓN DE LOS RESULTADOS

3.1.1 EN LA PRIMERA TOPOLOGÍA

Para mostrar el resultado de la primera topología se emplea el playbook denominado testing, el cual se observa en la Figura 3.1. En este playbook se muestran las configuraciones aplicadas a los switches capa 3.

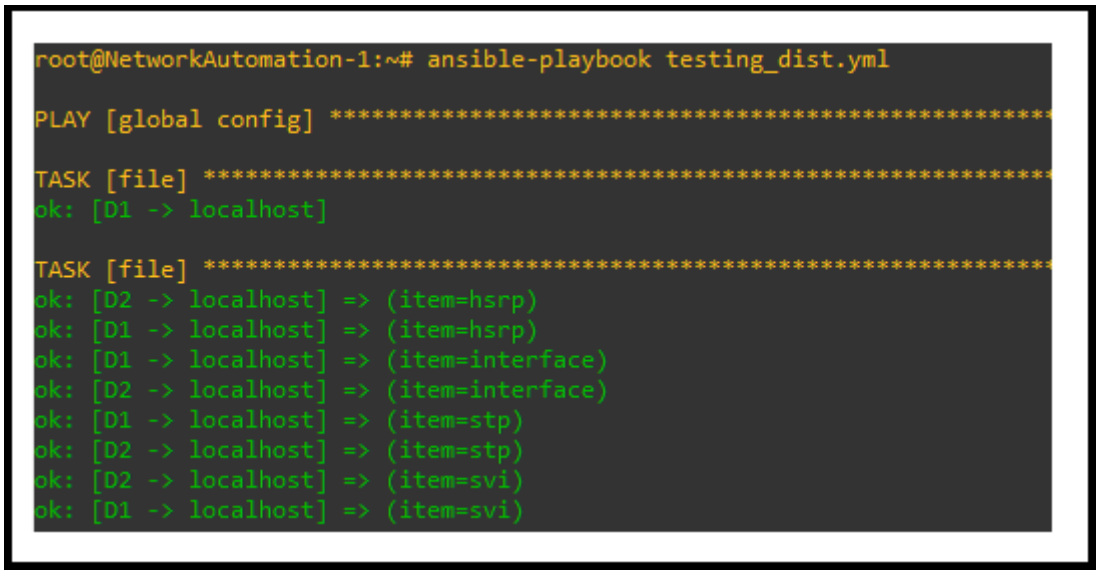

Figura 3.1: Ejecución de Playbook en la Primera Topología.

En la Figura 3.2 se observa que existe comunicación entre dos switch capa 3 usados en la simulación, lo mismo ocurre entre los otros switches capa 3 de la topología. Además, se observa que cada VLAN tiene su nombre y su propia dirección IP que son accesibles dentro de cada switch. En cada uno de los equipos de red se puede observar algo similar cuando los cambios son efectuados.

Para la topología con el protocolo VLAN y LAN, se toma en cuenta de mayor manera la conexión entre cada uno de sus equipos además de la creación y el correcto funcionamiento de cada VLAN. Para ello se confirma la conexión con cada equipo mediante el comando ping, a pesar de ser un

comando simple es muy efectivo para verificar conexión.

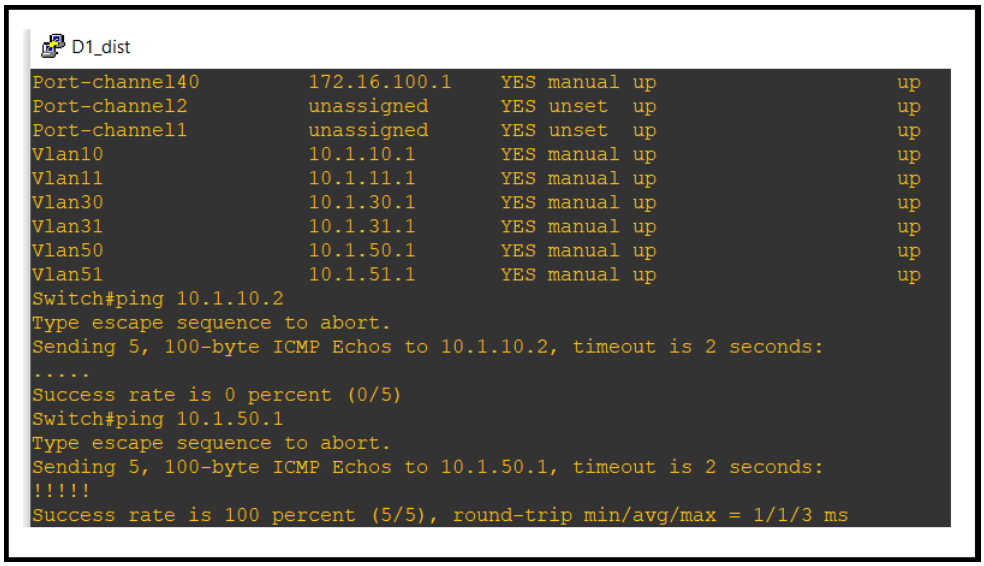

Figura 3.2: Puertos VLANs y Ejecución algunos Pings.

3.1.2 EN LA SEGUNDA TOPOLOGÍA

En esta topología se revisa la configuración aplicada a los routers: R1, R2, R3 y R4. Al ejecutar el playbook de un router denominado R1.yml, se muestra que el router R1 se conecta de manera satisfactoria con los otros tres routers, lo mismo ocurre con el router R2 y el resto; también se muestra el camino que siguió por medio de que dirección IP se dirigió. Para mostrar estos resultados se debe usar el playbook check para ICMP que verifica la configuración por medio del comando ping entre los distintos routers, tal como se muestra en la Figura 3.3.

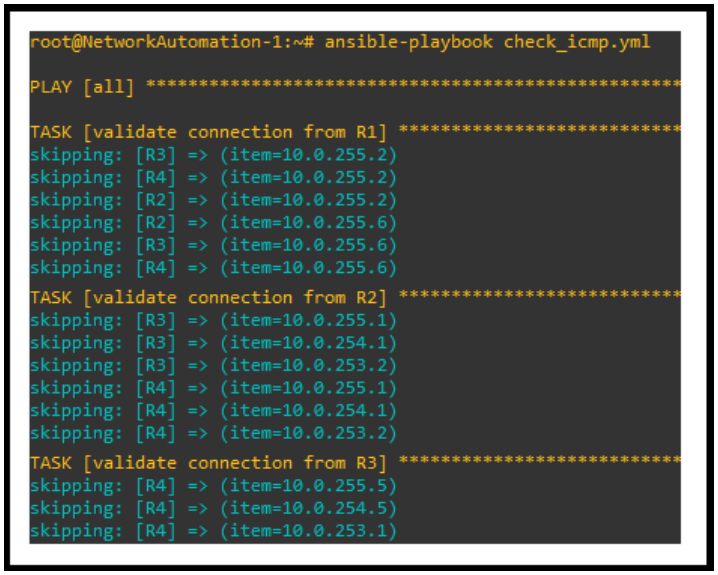

Figura 3.3: Ejecución de un Playbook de la Segunda Topología.

Para verificar conectividad, se accede a cualquiera de los 4 routers, y en estos routers se procede a realizar pings a los distintos vecinos, como se muestra en la Figura 3.4.

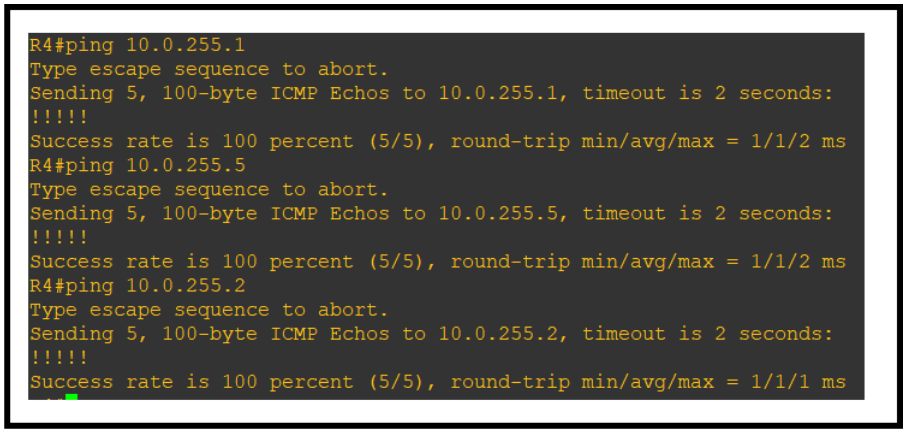

Figura 3.4: Verificación de conexión entre routers mediante el comando ping.

3.1.3 RESULTADOS DE LA TERCERA TOPOLOGÍA

Para visualizar los resultados se realiza la ejecución de un playbook denominado check.yml, que se muestra en la Figura 3.5. Este playbook contiene información de todos los equipos usados y que aplican las tareas de configuración que se realizaron.

Se afirma que cuando un bloque de tarea falla, los bloques posteriores no se ejecutan en los hosts. Para que las tareas no fallen se debe encontrar y corregir el error; para volver a ejecutar el playbook check.yml así se ejecutarán las tareas nuevamente. La ejecución exitosa de los otros playbooks llamados deploy pe como en ce (con extensión .yml), indica que no hubo error.

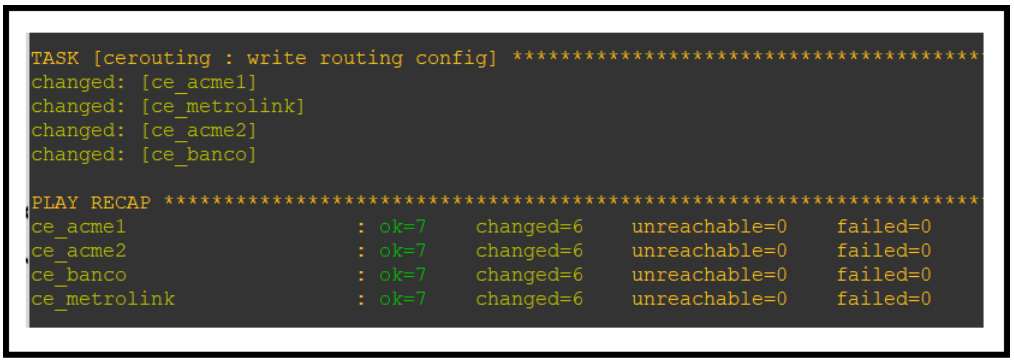

Figura 3.5: Ejecución de un Playbook de la Tercera Topología.

El siguiente playbook ejecutado llamado check.yml se muestra en la Figura 3.6. El que valida la conexión entre cada uno de los 10 routers, y verificando por cual dirección IP se comunican con cada router, esto se muestra como resultado de la ejecución de dicho playbook, y no genera ningún error.

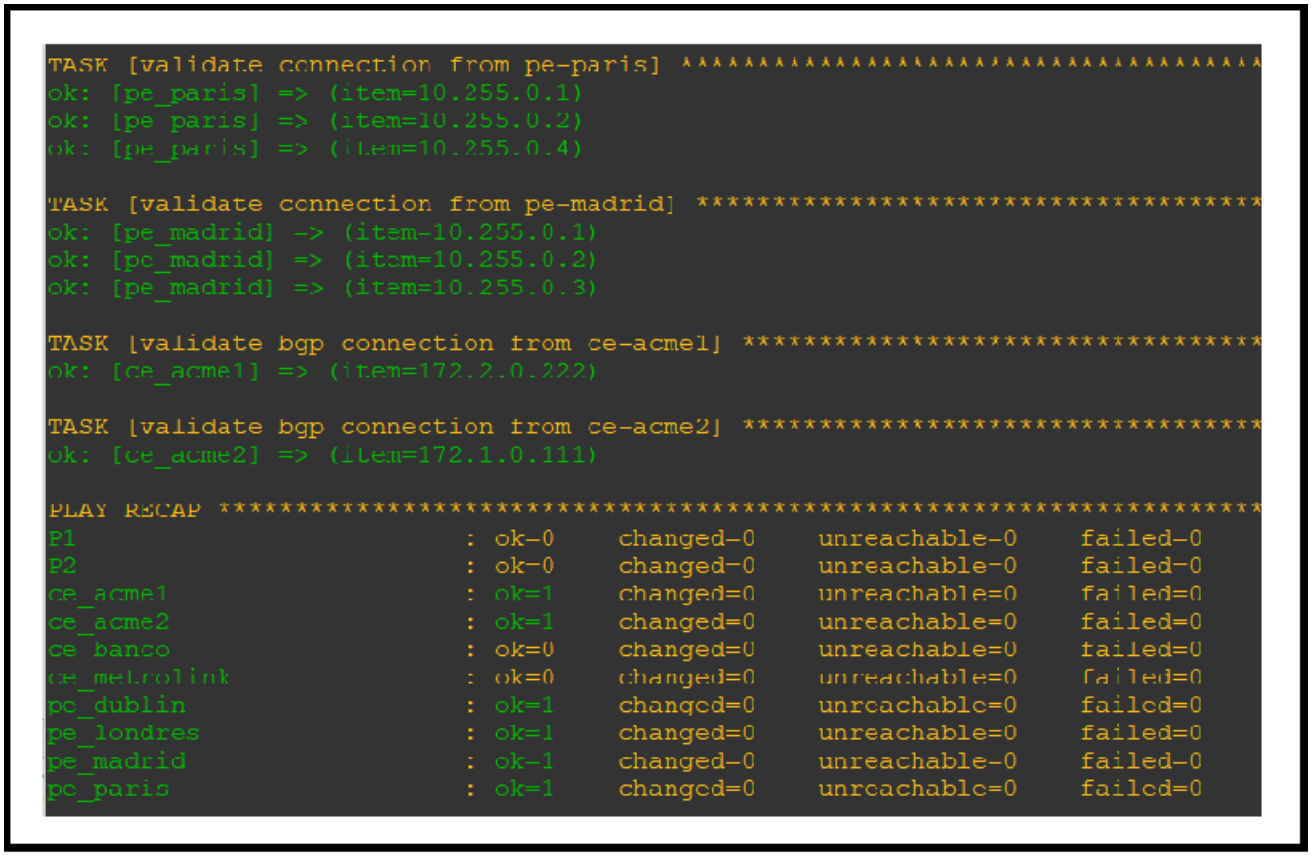

Figura 3.6: Ejecución de un Playbook de la Tercera Topología 2.

De igual manera que en las anteriores topologías en la Figura 3.7 se efectúa la verificación de conectividad por medio del comando ping desde un router dirigido al resto de routers dentro de la topología.

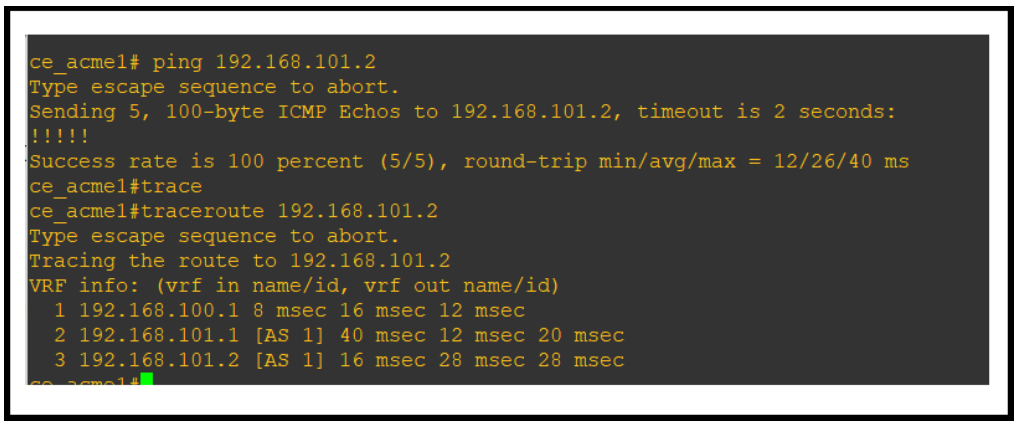

Figura 3.7: Ejecución del Comando Ping para Verificar Conexión.

3.1.4 RESULTADOS DE LA CUARTA TOPOLOGÍA

Al verificar los resultados en esta topología se comprobó la conexión entre los routers, esto mediante la ejecución del playbook asav1 ejecutada en la Figura 3.8, lo que se muestra que la conectividad falló, debido a que existió un numero de saltos nulo, por que el destino no se encontraba asignado. Sin embargo, en la Figura 3.9 se aprecia su correcto funcionamiento ya que se le agregó el destino que faltaba, al ejecutar nuevamente el playbook.

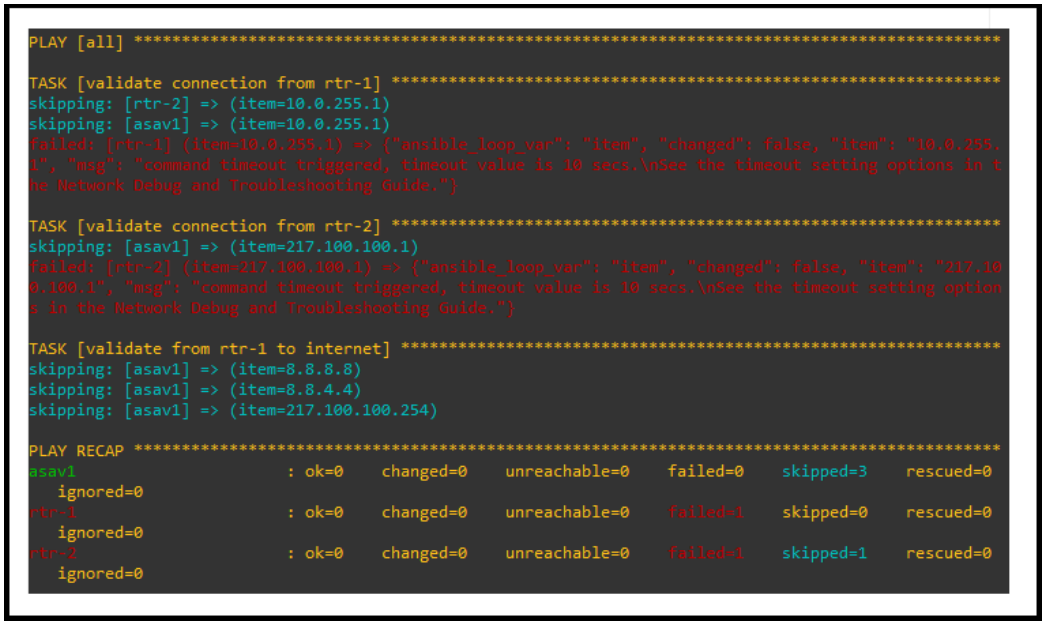

Figura 3.8: Ejecución del Comando Fallido.

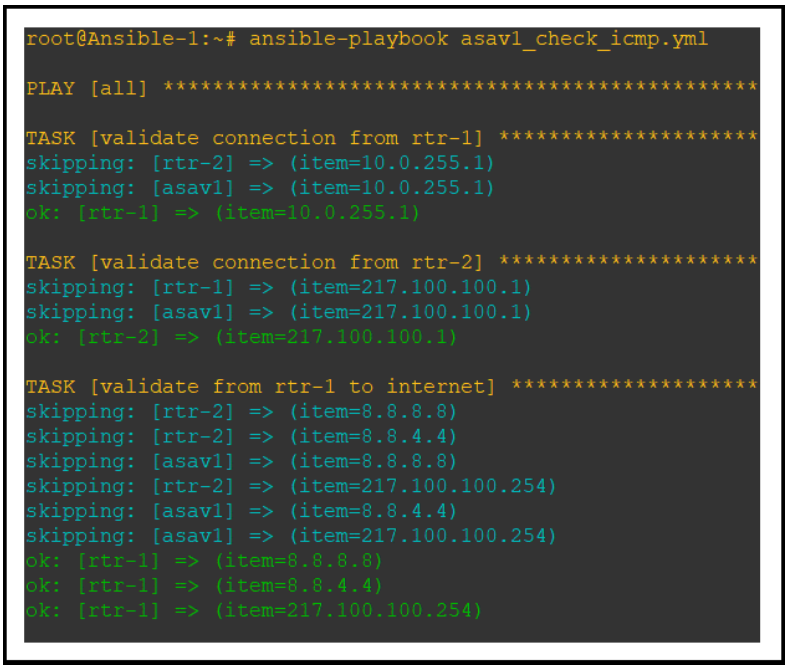

Figura 3.9: Ejecución del Comando Funcional.

3.2 CONCLUSIONES

- Se obtuvo una comprensión más detallada del funcionamiento de las herramientas open source usadas en el Network Automation, debido a que cada una de las herramientas estudiadas servirán para la ejecución de este trabajo, ya que al estudiar dichas herramientas se observó tienen características, procedimientos similares; y se ejecutan de la misma manera para el funcionamiento dentro de la network automation.
- Se tiene que una vez realizado el estudio de las dos herramientas de Network Automation, el cual determinó mediante distintos parámetros tales como: escalabilidad, disponibilidad, interoperabilidad, lenguaje y si es óptimo para realizar la automatización; de acuerdo a un análisis efectuado la herramienta idónea es Ansible, no obstante Netmiko es muy similar en esos aspectos. Se designó una para realizar el proceso de automatización de una red.
- Se observó que en la implementación de la Network Automation realiza procesos automáticos, dicho proceso automático proporciona un lenguaje único dentro de la topología de red, se da mediante el uso de scripts llamados playbooks y plantillas, al ejecutar los scripts estos generan un ahorro de tiempo ya que optimiza la gestión de cada equipo de red y la disponibilidad del administrador de red. Concluyendo que al usar Network Automation se optimiza el tiempo de trabajo, mejorándolo con respecto al proceso que actualmente es manual .
- Para la verificación de la Network Automation se usa una topología que consta de un nodo central y un nodo secundario, esta nos permite utilizar distintos protocolos; que al simularlos en la herramienta GNS3 nos permite configurar varios equipos de red simultáneamente lo que disminuye considerablemente el tiempo de configuración en cada equipo, esto mejora el rendimiento en la gestión.
- Se aplicaría varios tipos de configuración en los equipos de red, tales como router, switches y firewall mediante la network automation. Para este propósito se ha utilizado unos scripts llamados playbooks que son encargados de almacenar la configuración de cada equipo en el nodo principal y otros scripts llamados plantillas que se encargan de cómo se van a aplicar dichas configuraciones, y el requisito primordial es que la plataforma de networking automation trabaje conjuntamente con la herramienta de automatización.
- Se implementó la tecnología denominada Network Automation mediante las distintas configuraciones entre ellas se ha considerado la creación de VLANs y el manejo de BGP y MPLS en los equipos de red. Se generó en los equipos de red la configuraron respecto a los protocolos usados, implementando en los scripts las reglas y métodos usados visto en la parte teórica.
- Se observó que al ejecutar los playbooks adecuados en cada topología simulada, todos los playbooks creados en el nodo central de cada topología se ejecutarán de manera concatenada, ya que alguno de los playbooks dependen de una o más plantillas (estas generan bucles). Al ejecutar dicho proceso ayuda en la eficiencia de la red y en el óptimo desarrollo de la Network Automation.

- ❒ Se indica que para usar el lenguaje dentro del Network Automation existe una facilidad de aprendizaje de este lenguaje, para YAML que trabaja de manera lineal y por bloques; para JINJA2 que trabaja con bucles de decisión. Esto determina la compatibilidad entre las herramientas.

3.3 RECOMENDACIONES

- ❒ Al realizar una simulación o una implementación de la herramienta de Network Automation en las variadas topologías de redes que se necesite, existirán muchos equipos de red y se llevará una documentación detallada y en orden los playbooks y plantillas que se realizan para toda la red. Para todo eso se deben usar herramientas de editor de texto o editor de código fuente para guardar los códigos que se van a utilizar.
- ❒ Al tener que implementar Ansible en la herramienta de emulación GNS3, se recomienda usar un equipo con características funcionales mínimas. Por ejemplo se recomienda tener una memoria RAM de 16 GB para una máquina física o 8 GB para una máquina virtual, estos requisitos influyen en la cantidad de equipos que vayan a usarse.
- ❒ Hay que considerar las distintas características que tiene una red, a la que se vaya aplicar el aprovisionamiento automático esto debido a que las características influyen en el desempeño y rendimiento de la Network Automation. El rendimiento también depende de la herramienta escogida para llevar a cabo la automatización.
- ❒ Es necesario administrar eficientemente los directorio, hosts, direcciones ip y scripts de configuración de la herramienta de automatización. Hay que recordar que por lo general en el nodo central siempre trabaja con un sistema que se basa en Linux.
- ❒ Hay que realizar nuevos estudios para que se puedan implementar la herramienta de automatización en escenarios reales, pues este caso se lo realizó en una simulación.
- ❒ Es recomendable asignar las direcciones IP de manera estática en Ansible para realizar una gestión adecuada de la red. Además, se sugiere que las direcciones y nombres sigan cierto patrón asociado a la funcionalidad de cada equipo.
- ❒ Cuando se usa el GNS3 se debe tomar en cuenta un cierre brusco de esta herramienta, ya que si no se encuentra detenida la simulación los equipos de emulación se verían afectados o también afectarían a la máquina virtual de la herramienta de emulación.

4 REFERENCIAS BIBLIOGRÁFICAS

[1] *Automatización de la Red*. 2020. URL: https://www.efficientip.com/es/soluciones/automatizacion/.

[2] Fernando Boronat Segui. «Funcionalidades del protocolo ICMP». En: (2012).

[3] Razique Mahroua Adolfo Vazquez Snehangshu Karmakar Chen Chang George Hacker. *Automation with Ansible*. Edition 1. Red Hat. 2016.

[4] Brendan Choi. «Python Network Automation Labs: SSH paramiko and netmiko». En: *Introduction to Python Network Automation: The First Journey*. Springer, 2021, págs. 583-628.

[5] *Configuraciones de una Automatización*. URL: https://www.netapp.com/devops-solutions/configuration-management/.

[6] Rishabh Das. *Extending Ansible*. Packt Publishing, 2016.

[7] Jason Edelman. *Network Automation with Ansible*. O'Reilly Media, Incorporated, 2016.

[8] Malin Eriksson y Victor Hallberg. «Comparison between JSON and YAML for data serialization». En: *The School of Computer Science and Engineering Royal Institute of Technology* (2011), págs. 1-25.

[9] Andrew Froehlich y Jessica Scarpati. *What is network automation? definition from searchnetworking*. 2022. URL: https://www.techtarget.com/searchnetworking/definition/network-automation.

[10] Ana María García Pérez y Enrique Carbonell Muela. «Comparación de herramientas de gestión de la configuración». En: 2016.

[11] Juan Carlos Gutiérrez Chávez. «Monitoreo de una red de cómputo a través de SNMP». En: ().

[12] Daniel Hall. *Ansible Configuration Management*. Packt Publishing, 2015.

[13] Michael Heap. *Ansible: from beginner to pro*. Springer, 2016.

[14] Lorin Hochstein y Rene Moser. *Ansible: Up and Running: Automating configuration management and deployment the easy way*. .º'Reilly Media, Inc.", 2017.

[15] Jesse Keating. *Mastering Ansible*. Packt Publishing Ltd, 2015.

[16] JF Martínez. «¿ Qué es la automatización». En: *Blog SEAS. Recuperado el* 18 (2017).

[17] Madhurranjan Mohaan y Ramesh Raithatha. *Learning Ansible*. Packt Publishing, 2014.

[18] Natalia Morales et al. «Escala de Likert una herramienta económica». En: *Revista PDF* 6 (2016).

[19] Jason C Neumann. *The book of GNS3: build virtual network labs using Cisco, Juniper, and more*. No Starch Press, 2015.

[20] *Qué es la Automatización | Blog SEAS — seas.es*. https://www.seas.es/blog/automatizacion/que-es-la-automatizacion/.. [Accessed 04-Mar-2023].

[21] Danish Rafique y Luis Velasco. «Machine learning for network automation: overview, architecture, and applications [Invited Tutorial]». En: *Journal of Optical Communications and Networking* 10.10 (2018), págs. D126-D143.

[22] Gourav Shah. *Ansible Playbook Essentials*. Packt Publishing Ltd, 2015.

[23] Rishabh Sharma y Mitesh Soni. *Learning Chef*. Packt Publishing Ltd, 2015.

[24] William Stallings et al. *Comunicaciones y Redes de Computadores, 6aedición*. Prentice-Hall, 2000.

[25] Gabriel Hernan Tolosa y Fernando Raul Alfredo Bordignon. «Revisión: tecnologia de agentes de software». En: *Ciência da Informação* 28.3 (1999).

[26] Mircea Ulinic y Seth House. *Network automation at scale*. O'Reilly Media, 2017.

[27] Chris Welsh. *GNS3 network simulation guide*. Packt Publ., 2013.