# ESCUELA POLITÉCNICA NACIONAL

## ESCUELA DE FORMACIÓN DE TECNÓLOGOS

## IMPLEMENTACIÓN DE UN *FIREWALL* MEDIANTE LA HERRAMIENTA ANSIBLE

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

**LILIANA ALEXANDRA ACHIG TIPÁN**

**DIRECTOR: ING. GABRIELA KATHERINE CEVALLOS SALAZAR, MSC.**

**DMQ, febrero 2022**

# CERTIFICACIONES

Yo, Liliana Alexandra Achig Tipán 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.

**Liliana Alexandra Achig Tipán**

**liliana.achig@epn.edu.ec**

**lili_ale97@hotmail.com**

Certifico que el presente trabajo de integración curricular fue desarrollado por Liliana Alexandra Achig Tipán, bajo mi supervisión.

**ING. Gabriela Katherine Cevallos Salazar, MSC.**

**DIRECTOR**

**gabriela.cevalloss@epn.edu.ec**

# DECLARACIÓN DE AUTORÍA

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

LILIANA ACHIG

# DEDICATORIA

Mi tesis está dedicada con mucho cariño a mis padres Martha y Juan, por siempre estar a mi lado en todo momento, puesto que, sin su apoyo incondicional no estaría en esta instancia final, a mi madre por su cariño incondicional que siempre vela por mi bienestar y salud, dándome ánimos en todo momento, a mi padre por siempre estar pendiente de mí, a veces a pesar de la distancia.

A mi hermano Bryan, por estar presente en todo momento y apoyarnos el uno al otro en cualquier situación que se afronte.

Todo esto es por ustedes, por todo su esfuerzo que han hecho en todo este tiempo, sacrificando muchas cosas por mí, es por esto que les doy las gracias.

A mi padrino César, que se encuentra en el cielo y sé que desde allá está guiando todos mis pasos e intercediendo por mí.

A mi mejor amiga Paola, por estar siempre pendiente de mí, dándome su apoyo incondicional en los buenos y malos momentos que he pasado en el transcurso de mi carrera.

Por último, a mí misma, por demostrarme que el esfuerzo constante a pesar de las dificultades al final da frutos, sin importar las adversidades que se hayan puesto en frente.

**Liliana**

# AGRADECIMIENTO

Quiero agradecer en primer lugar a Dios por darme su bendición y darme las fuerzas necesarias para lograr cumplir este objetivo y esta etapa de mi vida se vea cumplida.

A mis padres y hermano por su apoyo, amor que me han brindado a lo largo de mi carrera universitaria, por siempre estar para mí en cada decisión.

A mis mejores amigos Adrián, Carlos y Kevin por brindarme su amistad y apoyo incondicional desde el inicio de la carrera universitaria hasta finalizarla, por enfrentar cada problema que se nos ha presentado.

Agradezco a la EPN ESFOT por todos los conocimientos adquiridos a lo largo de mi carrera universitaria, por los conocimientos obtenidos que me han forjado para poder ser una gran profesional.

Un agradecimiento a mi tutora de tesis Ing. Gabriela Cevallos quien me ha guiado desde el inicio hasta el final, ayudándome y apoyándome en todo el proceso de la realización de mi trabajo de integración curricular.

A cada uno de los docentes que he conocido en el transcurso de mi formación académica, ya que han sido pilares fundamentales para guiar, ayudar y apoyar en el trascurso de la carrera.

**Liliana**

# ÍNDICE DE CONTENIDOS

CERTIFICACIONES ..... I
DECLARACIÓN DE AUTORÍA ..... II
DEDICATORIA ..... III
AGRADECIMIENTO ..... IV
ÍNDICE DE CONTENIDO ..... V
RESUMEN ..... VII
*ABSTRACT* ..... VIII
1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO ..... 1
1.1 Objetivo general ..... 1
1.2 Objetivos específicos ..... 1
1.3 Alcance ..... 2
1.4 Marco Teórico ..... 2
Infraestructura como código ..... 2
Características y requerimientos de la herramienta Ansible ..... 4
Servidores que se pueden ejecutar en Ansible ..... 6
2 METODOLOGÍA ..... 8
Objetivo 1 ..... 9
Objetivo 2 ..... 9
Objetivo 3 ..... 9
Objetivo 4 ..... 9
Objetivo 5 ..... 10
3 RESULTADOS ..... 10
3.1 Crear la máquina virtual con el nodo ..... 10
3.2 Instalación de máquina virtual para el controlador ..... 13
3.3 Instalación de la herramienta de Ansible ..... 15
Generación de llaves SSH en el controlador ..... 17
Autorización de llaves SSH en el nodo *host* ..... 18

Configuración del archivo inventario ..... 18
3.4 Implementación de un *firewall* mediante la herramienta de Ansible ..... 20
Despliegue del *firewall* en el nodo *host* ..... 23
3.5 Pruebas de funcionamiento ..... 25
Pruebas de funcionamiento de la herramienta de Ansible ..... 25
Pruebas de funcionamiento del *firewall* implementado ..... 26
4 CONCLUSIONES ..... 35
5 RECOMENDACIONES ..... 36
6 REFERENCIAS BIBLIOGRÁFICAS ..... 37
7 ANEXOS ..... 40
ANEXO I: Certificado de Originalidad ..... i
ANEXO II: Video de implementación y pruebas de funcionamiento ..... ii

# RESUMEN

El presente proyecto de investigación, IMPLEMENTANCIÓN DE UN *FIREWALL* MEDIANTE LA HERRAMIENTA ANSIBLE, permite desplegar el *firewall* y bloquear o abrir puertos que se requieran en un cliente, esto mediante el manejo remoto con SSH desde una máquina central hacia la máquina *host*.

En la primera sección, se muestra la descripción del proyecto, la cual está conformada por el análisis y características que requiere la herramienta Ansible para su ejecución, de esta manera se obtiene el conocimiento necesario para el correcto funcionamiento de Ansible, además se definen tanto el objetivo general, como los objetivos específicos que son guía para el desarrollo del proyecto.

En la segunda sección se describe la metodología utilizada para la implementación del proyecto, mostrando las instrucciones y parámetros que permiten el desarrollo de cada uno de los objetivos.

En la tercera sección se detalla el procedimiento para tener listo el nodo controlador y el nodo *host*, y la implementación y despliegue del *firewall* mediante la herramienta Ansible, además en esta sección se encuentran las pruebas de funcionamiento correcto del proyecto.

La cuarta y quinta sección contienen conclusiones y recomendaciones obtenidas en el desarrollo del proyecto.

**PALABRAS CLAVE:** Ansible, Orquestación, Inventario, protocolo SSH.

# ABSTRACT

*The present research project, IMPLEMENTATION OF A FIREWALL THROUGH THE* ANSIBLE *TOOL, allows to deploy the firewall and block or open ports that are required in a client, this by means of remote management with* SSH *from a central machine to the host machine.*

*In the first section, the description of the project is shown, which is conformed by the analysis and characteristics that the* Ansible *tool requires for its execution, in this way to obtain the necessary knowledge for the correct operation of* Ansible*, in addition the general objective is defined, as well as the specific objectives that are guide for the development of the project.*

*The second section describes the methodology used for the implementation of the project, showing the instructions and parameters that allow the development of each of the objectives.*

*The third section details the procedure to have the controller node and the host node ready, and the implementation and deployment of the firewall using the* Ansible *tool, also in this section are the tests of correct operation of the project.*

*The fourth and fifth sections contain conclusions and recommendations obtained in the development of the project.*

***KEYWORDS:*** Ansible*, Orchestration, Inventory,* SSH *protocol.*

# 1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

Debido al avance de la tecnología, también se han generado varias amenazas que se propagan sin que el usuario lo perciba en su equipo; es decir si el equipo tiene un nivel bajo de configuración en cuando a seguridad o vulnerabilidades sin parches, es más fácil una infiltración.

Por tal motivo es importante tener un *firewall* que permita identificar y bloquear el tráfico que no es deseado o bloquear usuarios no autorizados. Su función principal es relevante, ya que, de no ser por el *firewall*, las redes podrían ser atacadas con mayor frecuencia [1].

Ansible es una herramienta potente en cuanto a la automatización de un código abierto, ya que no requiere de un *software* o paquetes de manera adicional para su ejecución, facilitando su implementación. Define tareas realmente necesarias para que se ejecute de forma remota en el sistema de destino mediante un *playbook.*

El presente proyecto pretende desplegar un *firewall* mediante la herramienta de Ansible, con el fin de que el usuario tenga un corta fuegos para configurar reglas de *firewall* que permitan bloquear o autorizar puertos y protocolos que necesite la máquina. Todo esto de manera remota; con el fin de garantizar la eficiencia, rapidez y fiabilidad, con menos esfuerzo y riesgo de errores humanos o vulnerabilidades de seguridad.

## 1.1 Objetivo general

Implementar un *firewall* mediante la herramienta Ansible.

## 1.2 Objetivos específicos

- Crear la máquina virtual con el nodo.
- Instalar la máquina virtual para el controlador.
- Instalar la herramienta de Ansible.
- Implementar un *firewall* mediante la herramienta de Ansible en el nodo.
- Realizar las pruebas de funcionamiento.

## 1.3 Alcance

El presente proyecto conlleva el despliegue de un *firewall* mediante la herramienta Ansible, por medio de la creación de máquinas virtuales las cuales son: un controlador que es la máquina principal encargada de comenzar la orquestación donde se instala Ansible y un nodo el cual se maneja mediante el controlador a través de SSH; ayudando así de manera remota a realizar las configuraciones necesarias para la implementación del *firewall* requerido.

## 1.4 Marco Teórico

### Infraestructura como código

La infraestructura como código (IaC) tiene la capacidad de administrar y operar en componentes de infraestructura mediante código; es decir automatizar un servidor, una red privada o una infraestructura en general mediante código. Está basada en la programación de estructuras de *hardware* con tres principales apoyos tecnológicos, los cuales son: servidores, almacenamiento y redes [2] [3].

Las características más relevantes de IaC son las siguientes:

- Se utiliza código para describir la infraestructura: Es decir no se utiliza una interfaz de usuario para iniciar la máquina virtual, sino que se describe las características deseadas en código y se utiliza una herramienta para que lo ejecute.
- La utilización de un mismo código probado en las operaciones de administración de infraestructura: Se debe confiar en las funciones de código probadas que se utilice en el equipo para las operaciones comunes de administración, garantizando así que no se cree su propia implementación de manera repetitiva.
- Las operaciones automatizadas: No se necesita de comandos para iniciar y configurar el sistema, sino que se requiere de una sintaxis de configuración proporcionada por la herramienta IaC para indicar lo que se debe realizar.
- Aplicación de prácticas de desarrollo de *software* a la infraestructura: Para el desarrollo de *software* se debe mantener el código de control de fuente o revisiones por pares muy comunes; haciendo un desarrollo fiable y un trabajo en equipo.

En concreto, IaC brinda la posibilidad de automatizar procesos en los cuales casi siempre van a tener un cierto grado de complejidad y que se podría repetir de manera frecuente.

Entre los ejemplos de herramientas IaC se encuentran Terraform, *Puppet,* Ansible, entre otras [4].

### Terraform

Es una herramienta que se maneja por medio de código abierto que ha sido desarrollado por HashiCorp, ayudando así al aumento de la automatización en cuanto a la infraestructura como código, mediante la creación y modificación de recursos y la destrucción de los que ya no sean necesarios [5].

Terraform tiene un dato interesante, el cual es que se puede comprobar la configuración antes de aplicarla, de esta manera se puede corregir detalles antes de que se ejecuten los cambios, para al final validar lo que se tiene definido en el código y ejecutarlo. Otro dato importante es que Terraform se puede usar con varios proveedores de nube, ya sean públicos o privados, entre ellos: Azure, Google *Cloud Platform*, OpenStack*,* entre otros [6].

Las principales características que posee Terraform se tiene:

- Infraestructura como código: Se encarga de guardar ficheros de la configuración de proveedores de infraestructuras.
- Planes de ejecución: Es donde se informa lo que se va a realizar de manera exacta, esto antes de ejecutar el plan, por ejemplo: máquinas virtuales que se vayan a construir o destruir.
- Grafo de recursos: Se puede visualizar los distintos componentes de manera dependiente, ya sean máquinas, bases de datos, etc.
- Cambios automatizados: Se lo hace de manera programada, para ejecutar *test*, despliegues a producción automática, entre otros.

### *Puppet*

Es otra de las herramientas de IaC en la administración de configuración, al utilizar *Puppet* se puede definir el estado final en el que se encuentra, para así saber si es el deseado en la infraestructura y realiza de manera exacta lo requerido; tiene la función de automáticamente arreglar algún cambio erróneo, para así reforzar el estado deseado [7].

La herramienta *Puppet* es gratuita, siempre y cuando se utilice menos de 10 nodos, caso contrario se convierte en un modelo empresarial con una forma de pago; esta herramienta es la preferida en compañías como Google o Dell*,* ya que su infraestructura es consistente y maximiza su productividad [8].

### Ansible

Ansible se caracteriza por ser una herramienta libre, desarrollada por RedHat, la cual permite la automatización de varios equipos que se requiera administrar; siendo un motor en la automatización de infraestructura simple, administración de configuración, despliegue de aplicaciones, orquestación dentro del servicio, entre otras.

Mediante el análisis realizado de diferentes herramientas de IaC, se determina que todas las herramientas son potentes y fáciles de aprender, dependiendo de la necesidad y requerimiento del usuario.

En este contexto, Terraform permite ser independiente de la nube, admitiendo varios proveedores, puesto que los usuarios se valen de la administración de diferentes entornos de nube u ofertas que utilizan el mismo lenguaje de programación. Por otro lado, *Puppe*t está basado en un modelo servidor – cliente, actualizando al cliente a través de un catálogo; es difícil desplegarlo desde cero, ya que la complejidad de su implementación conlleva a tener un análisis de despliegue más arduo. Finalmente, Ansible es una herramienta poderosa ya que se la utiliza para transportar servicios y servidores a un estado deseado, maneja una gran diversidad de métodos y clases de configuración. Ansible puede conectarse a diferentes proveedores mediante módulos para configurar recursos.

La mayoría de usuarios prefieren usar Ansible por ser liviano en cuanto a lo que es codificación, dando como resultado la capacidad de una implementación eficaz y rápida; permite ocultar un diseño final en la solución, esto a partir del fichero de inventario; tiene la capacidad de instalar y administrar *software* o la configuración de servidores existentes, como también la distribución de infraestructura.

El presente proyecto se lo implementará sobre Ansible, herramienta potente, fácil de aprender y amigable con el usuario. Se puede realizar todas las tareas sobre los servidores necesarios de forma automática, de esta manera se aumenta la productividad y se minimiza los errores humanos que se producen en instalaciones y configuraciones manuales.

## Características y requerimientos de la herramienta Ansible

Ansible es una herramienta de automatización que no requiere de agentes, ni de infraestructura de seguridad adicional que sea personalizada, de esta forma se hace más fácil de implementar. Utiliza el lenguaje YAML *(Ain't Markup Language)* el cual permite describir los trabajos de automatización de una forma que se asemeja a un inglés simple [3]. Puntualizando en sus características se tienen las siguientes:

- Sin agente: No existe *software* o agente que deba ser instalado en el cliente para comunicarse con el servidor y únicamente utiliza OpenSSH (*Open Secure Shell).*
- Idempotente: No importa el número de veces que sea llamada la operación, el resultado siempre debe ser el mismo.
- Simple y extensible: La herramienta Ansible se encuentra escrita en Python y usa YAML como un leguaje de *playbooks* o denominado libro de jugadas.

La herramienta Ansible como requerimiento principal utiliza SSH para poder establecer la comunicación entre los *hosts* remotos que se encuentran administrados y no se necesita ningún agente en ellos o *software* especial, lo único que se necesita es Python, en el caso que sea de GNU/Linux. Ansible se encuentra distribuida en Fedora, RedHat Ubuntu, *Enterprise* Linux, CentOS y *Scientific* Linux [9] [10].

La orquestación en Ansible involucra diferentes elementos para que se dé una operación completa en la ejecución, gestión de configuración y despliegue de aplicaciones. Para ello se necesita de los siguientes componentes:

**Nodo o máquina de control**

En el nodo de control se instala y configura Ansible, se puede tener algunos nodos de control y se configura sobre cualquier sistema operativo que esté basado en Linux o Unix. Por ahora, Ansible no se puede instalar sobre *Windows*, ya que resulta complejo [11], pero una de las soluciones es utilizar algún contenedor, como por ejemplo Docker [12]. La máquina de control conoce a los nodos administrados o *host* a través de un inventario.

**Nodo administrado o *host***

Los nodos *hosts* son manejados desde la máquina de control mediante SSH y que a su vez tengan instalado Python 2 (versión 2.6 o superior) o Python 3 (versión 3.5 o superior). Ansible permite algunos sistemas operativos como nodos administrados, incluso *host* con sistema operativo *Windows* [11].

**Inventario**

Existe un kit para la automatización de configuración de Ansible, esto mediante un archivo denominado inventario y la ejecución de comandos para automatizar por completo la configuración del servidor LEMP (**L**inux, **E** Nginx, **M**ariaDB y **P**HP-FPM) remoto.

Los archivos de inventario son la lista de *hosts* que será administrado mediante Ansible, en el cual se enumeran en el archivo simple, conjuntamente con las direcciones IP,

servidores, bases de datos, entre otros. Una vez registrado los cambios en el inventario, se pueden asignar variables a cualquiera de los *hosts* utilizando dicho archivo [13].

#### *Playbooks*

Los *playbooks* constan de una lista sistemática de tareas y alguna otra directiva que indica los *hosts* que se tiene como objetivo para la automatización, tomando en cuenta si se debe utilizar o no un sistema de privilegios que se requiera para ejecutar las tareas, definir variables o incluir archivos [11].

Ansible se encarga de ejecutar tareas de manera secuencial, y la ejecución completa del *playbook* es denominada *play*, cada uno está compuesto por una o varias jugadas; estos archivos simples están escritos en YAML [13].

#### Controladores

Se utilizan los controladores para realizar acciones en un servicio, como reiniciar o detener un servicio que se encuentre ejecutándose de manera activa en el sistema del nodo administrado. Estos controladores se activan mediante las tareas y ejecución que se realice al final de un *playbook*, dando como resultado el reinicio del servicio y al final se observa lo ejecutado [11].

#### Funciones

Las funciones son un conjunto de *playbooks* y sus archivos están relacionados a una estructura predestinada a que Ansible la reconozca. Estas funciones ayudan en la reutilización y reasignación de *playbooks* en paquetes compartidos, que tienen como objetivo la automatización, como podría ser la configuración de un servidor web, en un entorno PHP o de un servidor MySQL [11].

Debido a que la mayoría de módulos de Ansible se encuentran escritos en Linux para su ejecución y administración, en los denominados nodos remotos, los módulos están escritos en Python y la mayoría de ellos no tienen funciones en *Windows*. Es por ello que se necesita la *PowerShell*. Con esto se puede concluir que para no tener dificultades en cuanto a las conexiones entre la máquina central y el nodo *hosts* o administrativo el mejor sistema operativo que se puede utilizar es GNU Linux. Se realizará el proyecto sobre Ubuntu que tiene varias características, las más relevantes son: el fácil manejo, instalación robusta de programas, libertad de uso y distribución.

### Servidores que se pueden ejecutar en Ansible

Ansible se ha utilizado para la automatización de varios servidores, puesto que es una solución que tiene escalabilidad en cuanto a los desafíos que tienen varias empresas,

es por ello que Ansible toma en cuenta que al momento de ejecutar un servidor este sea lo más seguro posible, ya que un servidor inseguro es un servidor comprometido [14]. A continuación, se listan algunos ejemplos de servidores que se pueden ejecutar conjuntamente con Ansible.

**Nginx**

Servidor *Web* implementado para aminorar el uso de memoria, mediante la función de no crear nuevos procesos para cada solicitud *Web* que se reciba, sino que su enfoque asincrónico atienda las solicitudes en un solo hilo, esto quiere decir que cada que aparezca un usuario realizando una petición al servidor, solo entrando al dominio se generará un nuevo hilo. La característica principal de Nginx es su capacidad por lidiar con varias conexiones y su rápida velocidad [15].

**Jenkins**

Es una de las herramientas de automatización de tareas en procesos de integración continua (CI), dicha herramienta es un servidor de *Open Source* que ayuda con la aceleración de un proceso de desarrollo y entrega de *software* mediante la automatización [16]. Las características principales son:

- Automatizar: Compilación y el testeo del *software.*
- Notificar: En los equipos pertinentes a detección de errores.
- Desplegar cambios: En el código que se haya desplegado.
- Realizar seguimientos: La calidad de código y cobertura deber tener pruebas de funcionamiento.
- Generar documentación del proyecto.

**UFW – *Firewall***

Es un paquete que está revestido de *IPTables,* pero UFW *(Uncomplicated Firewall)* es mucho más fácil, tomando en cuenta todas las funcionalidades del mismo, ya que está instalado por defecto en Ubuntu a partir de la versión 8.x de su distribución.

Se encuentra escrito en Python y se lo publica bajo la licencia de GNU, no necesita una instalación, pero debe estar actualizado [17]. Para la configuración se debe tener acceso *root* o capacidades con el comando sudo, entre otras prioridades, es decir el puerto 22 debe conectarse sin estar abierto mediante SSH. Consta de cuatro etapas que son: instalación, reglas predeterminadas, reglas que se pasan a una lista de reglas adicionales, seguido de la habilitación de UFW [18].

***FirewallD* (*Firewall* Daemon)**

Es un servicio dinámico de un cortafuegos, tiene el propósito de entregar un *firewall* administrado de manera dinámica, el cual cuente con soporte para zonas de red en las que se define el nivel de confianza de las conexiones o interfaces que se utiliza [19].

*FirewallD* ofrece una interfaz para servicios o aplicaciones, esto con el fin de agregar reglas de *firewall,* facilitando así de manera directa las tareas de control. La principal ventaja de usar *FirewallD* es que los cambios que se realicen, se pueden hacer en tiempo real sobre el entorno de ejecución sin que se reinicie de manera obligatoria el servicio o *Daemon* como suele suceder con otras utilidades [20].

Es un módulo que se lo utiliza para la actualización de reglas de *firewall* en *hosts* remotos, los puertos pueden ser tanto TCP como UDP, que se pueden deshabilitar o habilitar. De la misma manera, los servicios que se pueden permitir o bloquear [20].

Para trabajar con *FirewallD* en Ansible se debe tomar en cuenta que consta de tres capas, las cuales son:

- Capa central: Es la encargada de manejar la configuración y *backends*
- Interfaz D-Bus: Es el medio primordial para el cambio y creación de la configuración del *firewall.*
- *Backends:* Sirve para interactuar con *netfliter* (es un módulo nativo de *kernel*, que se lo utiliza para el *firewall*) [21].
- *FirewallD* acepta dos tipos de entornos de configuración: Configuración de tiempo de ejecución que es efectiva solo hasta que se reinicia el servicio de *FirewallD. Y* Configuración permanente que se guarda y funciona de manera persistente [21].

# 2 METODOLOGÍA

Según el desarrollo del presente proyecto se utilizó una metodología exploratoria, debido a que se partió de la investigación de diferentes herramientas IaC, y se analizó las características y ventajas más relevantes de Ansible. La información que se obtuvo ayudó a entender cómo funciona Ansible conjuntamente con sus características y componentes principales.

Luego se implementó la metodología aplicada, debido a que se aplica esta potente herramienta desplegando de forma automática un *firewall*, el cual tiene como objetivo permitir o denegar puertos a una máquina. Mediante esto Ansible satisface la efectividad

y simplificación de tareas complejas al momento de su ejecución, puesto que, al automatizar tareas, se eliminan en un gran porcentaje los errores humanos en las operaciones de TI, ahorrando tiempo de implementación.

### Objetivo 1

Se instaló la máquina virtual para el nodo *host*, la cual cumple con los requerimientos necesarios, como lo es la instalación de OpenSSH para que pueda trabajar de manera conjunta con la máquina central.

Así como también se agrega a la máquina virtual la preferencia de red para obtener direccionamiento lo cual ayudará al proceso de conexión mediante SSH con la máquina central.

### Objetivo 2

Se instaló la máquina virtual correspondiente a la máquina central o controlador de Ansible, cumpliendo con cada uno de los requerimientos para la instalación correcta de la herramienta Ansible, entre ellos la instalación de OpenSSH, que es primordial para la conexión remota SSH entre el nodo *host* y controlador.

De igual forma que con la máquina del nodo, se agrega a la máquina virtual la preferencia de red para obtener direccionamiento IP.

### Objetivo 3

Se procedió con la instalación de la herramienta Ansible, la cual debe contener Python y una versión actualizada para trabajar sin ninguna complicación, también se verificó que todas las actualizaciones de Linux se encuentren al día.

Además, se generaron llaves SSH para obtener una conexión remota con el nodo *host*; así como también se ejecutó un inventario, el cual está encargado de guardar la información necesaria y de esta manera se llevó a cabo la conexión SSH entre en el nodo *host* y el controlador.

### Objetivo 4

Se realizó la creación y ejecución del *playbook,* de esta manera se logra el despliegue correcto del *firewall* mediante tareas que se encuentran especificadas en el *playbook*, siendo así el bloqueo o apertura de puertos de protocolos en el nodo.

### Objetivo 5

Se realizó finalmente, las pruebas necesarias para comprobar el funcionamiento, tanto de la herramienta Ansible, así como también de la implementación del *firewall,* para estar seguros que no existe errores en cuanto a la ejecución del proyecto.

# 3 RESULTADOS

En esta sección se tiene el procedimiento de la implementación de un *firewall* mediante la herramienta Ansible. En primera instancia se da a conocer la creación de una máquina denominada nodo *host* y de la máquina controlador, estableciendo la comunicación entre ellas mediante SSH. Además, se detalla el procedimiento de la ejecución de Ansible y el despliegue del *firewall* en el nodo *host*; esto facilitará al lector plasmarlo en algún momento y aprender cómo funciona la herramienta Ansible.

## 3.1 Crear la máquina virtual con el nodo

Se utilizó como *software* de virtualización a VirtualBox, dentro de este virtualizador se crea una nueva máquina virtual, la cual se le denomina Ubuntu Nodo Ansible, ver Figura 3.1. Sobre esta máquina virtual correrá el sistema operativo Ubuntu *Server* 18.04, observar Figura 3.2, para que funcione como el nodo *host* de Ansible. Se creó la máquina con los parámetros de 2048 (MB) de RAM y 20 (GB) de disco, una vez completada la configuración requerida se inició la instalación de Ubuntu *Server*.

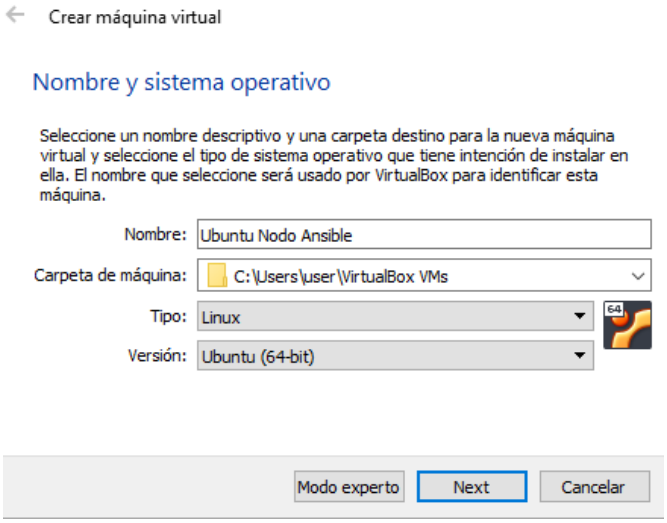


**Figura 3.1** Creación de la máquina virtual del nodo *host*

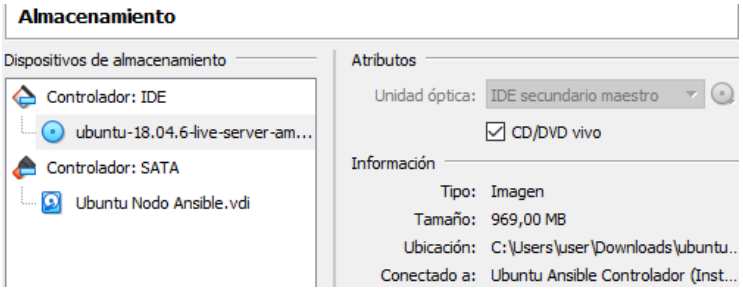


**Figura 3.2** Instalación de la ISO de Ubuntu *Server*

Después de haber terminado con la instalación, se procedió a correr el sistema operativo y dentro de Ubuntu *Server* se configuran los datos verificados en la Figura 3.3. Es importante considerar que durante la instalación se pregunta si desea instalar OpenSSH, como se observa en la Figura 3.4, se confirma dicha instalación, ya que se requiere que en la máquina Ubuntu Nodo *host* esté instalado SSH para lograr así la comunicación remota con la máquina controlador.


**Figura 3.3** Configuración del perfil de Ubuntu Nodo *host*


**Figura 3.4** Instalación de OpenSSH en Ubuntu Nodo *host*

Para finalizar, se obtiene la instalación correcta de Ubuntu *Server* para la máquina del nodo *host*, el cual para acceder a su terminal pide el usuario y contraseña, ver Figura 3.5.

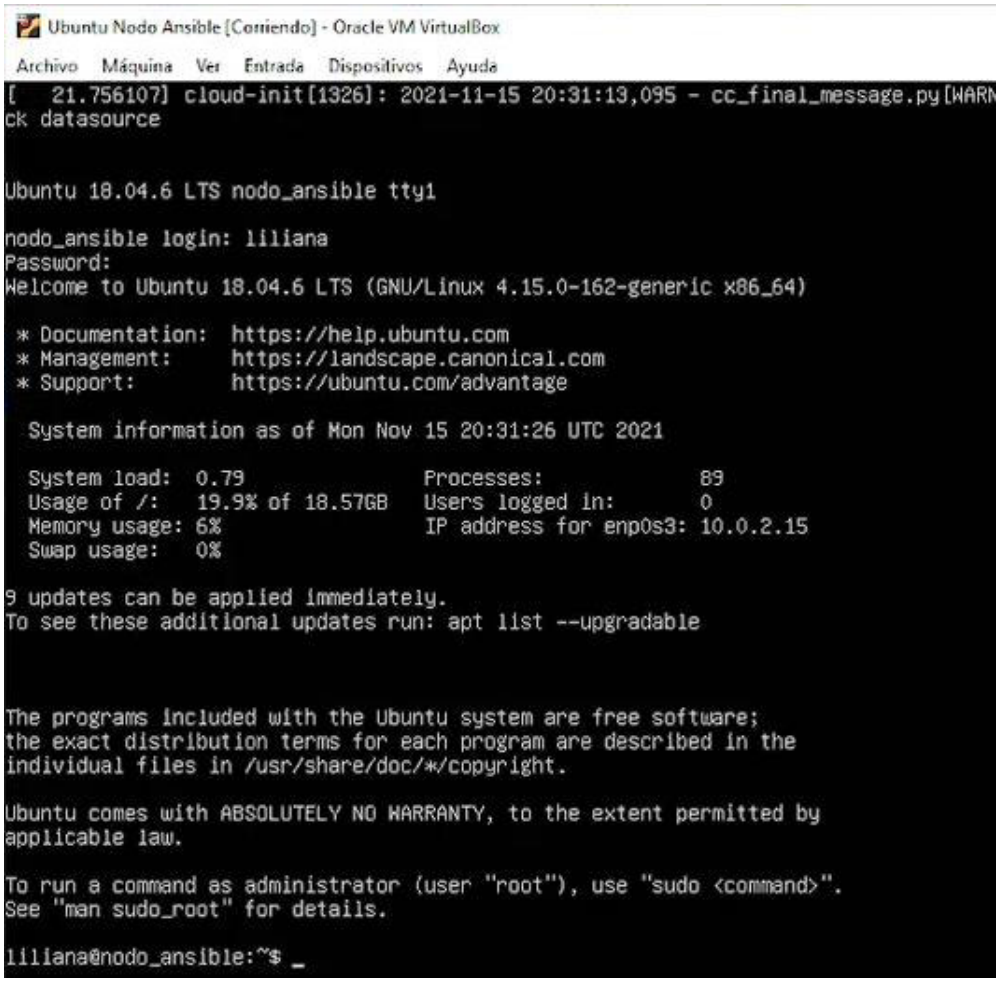


**Figura 3.5** Instalación completa de Ubuntu Server – Nodo *host*

Una vez finalizada la instalación del nodo *host,* como preferencia de red en VirtualBox se coloca en Adaptador Puente, esto con el fin de obtener una dirección IP del servidor DHCP (*Dynamic Host Configuration Protocol*) del hogar, con esto se tiene acceso a Internet. Se procedió a realizar un ***ifconfig***, el cual ayudó a obtener el direccionamiento IP del nodo *host*, en este caso es 192.168.56.101/24, como se observa en la Figura 3.6; dicha dirección será usada en la generación de claves SSH e inventario en la máquina central.

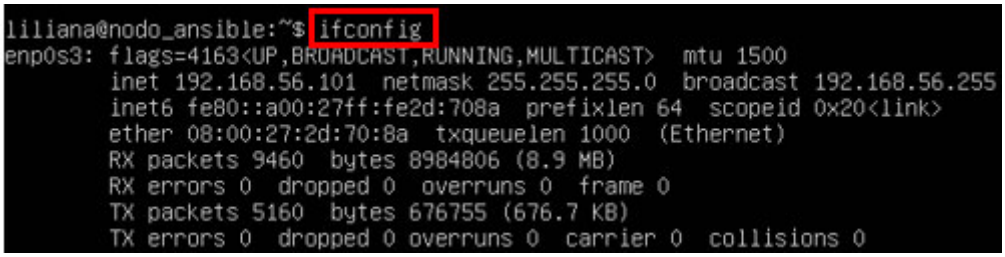


**Figura 3.6** Direccionamiento IP del nodo *host*

## 3.2 Instalación de máquina virtual para el controlador

Al igual que la máquina virtual del nodo, se utilizó como *software* de virtualización a VirtualBox, dentro de este virtualizador se crea una nueva máquina virtual, la cual se le denomina Ubuntu Ansible Controlador, ver Figura 3.7. Sobre esta máquina virtual correrá el sistema operativo Ubuntu *Server* 18.04 como se observa en la Figura 3.8, para que funcione como controlador o máquina central de Ansible. Se creó la máquina con los parámetros de 2048 (MB) de RAM y 40 (GB) de disco, esto debido a que para la herramienta Ansible se necesitó guardar ciertos cambios realizados; una vez completada la configuración requerida por el mismo, se inició la instalación de Ubuntu *Server* 18.04.

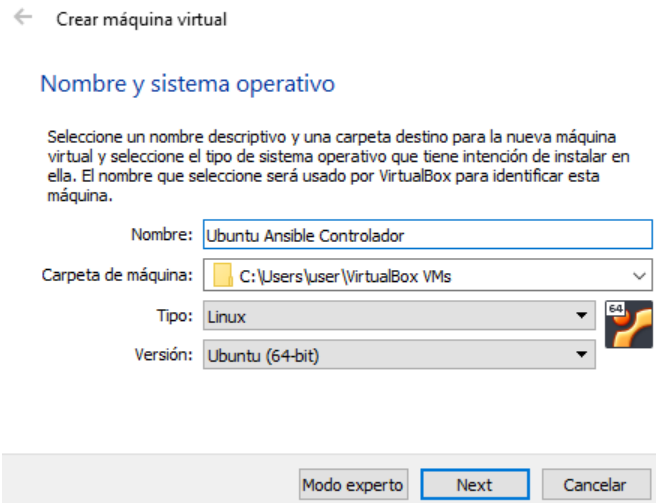


**Figura 3.7** Creación de la máquina virtual – Ubuntu Ansible Controlador

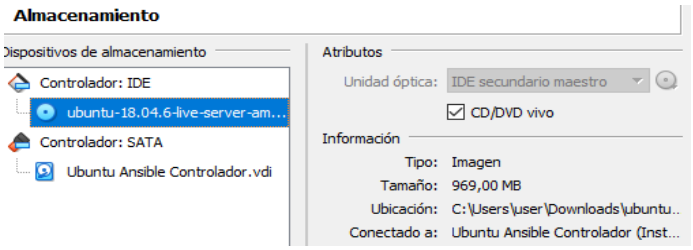


**Figura 3.8** Instalación del sistema operativo Ubuntu *Server*

Dentro de la instalación de Ubuntu *Server*, se colocan los datos verificados en la Figura 3.9, igualmente que con la instalación del nodo *host* se instala OpenSSH, como se observa en la Figura 3.10, esto ayudará a que posteriormente se conecten de manera

remota tanto la máquina del controlador o máquina central con la máquina del nodo *host.*

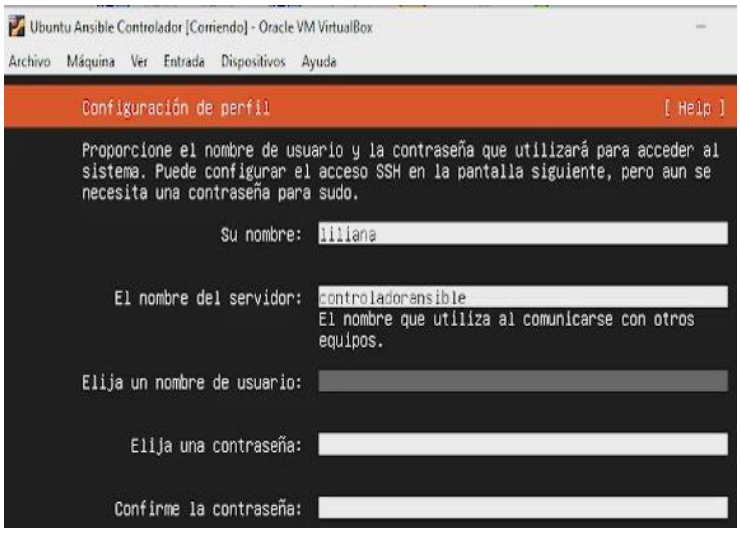


**Figura 3.9** Configuración del perfil para la máquina central Ansible

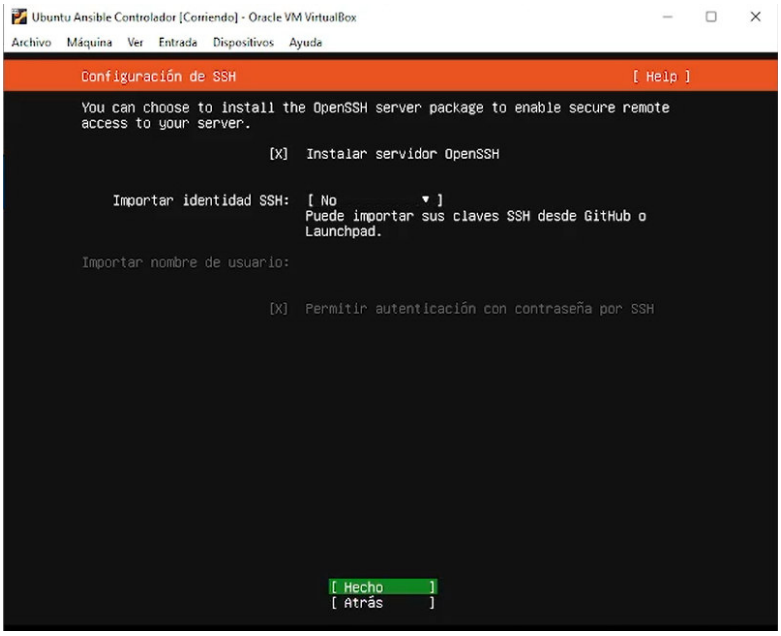


**Figura 3.10** Instalación de OpenSSH

Finalmente se obtiene la instalación correcta de Ubuntu *Server* para la máquina central o controlador, el cual para acceder a su terminal solicita el usuario y contraseña, ver Figura 3.11.

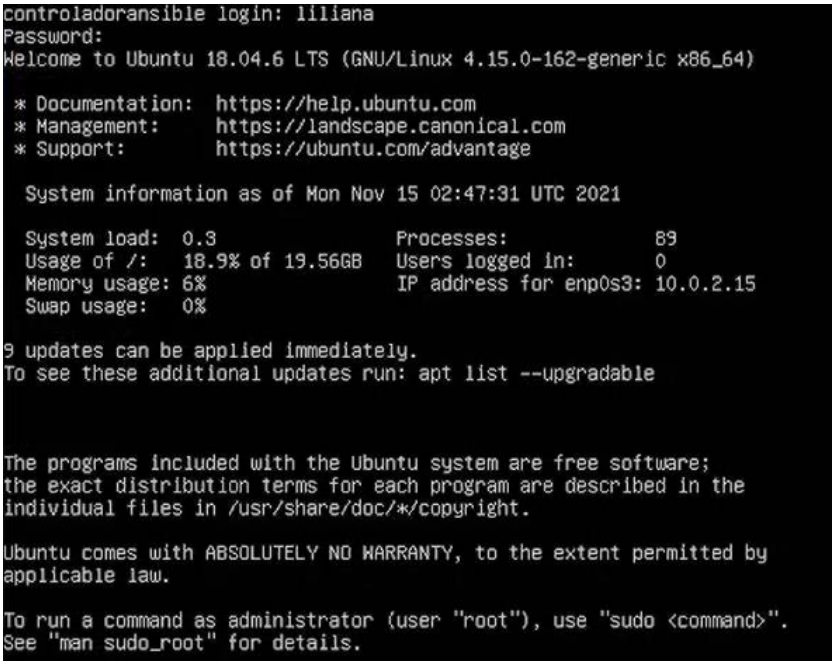


**Figura 3.11** Instalación correcta de Ubuntu *Server*

Una vez finalizada la instalación del sistema operativo para el controlador, como preferencia de red en VirtualBox se coloca en Adaptador Puente, esto con el fin de obtener una dirección IP del servidor DHCP del hogar, con esto se tiene acceso a Internet que es necesario para la instalación de Ansible. A continuación, se realiza un ***ifconfig***, para obtener la información acerca del direccionamiento IP, que en este caso posee la dirección 192.168.56.104/24, observar Figura 3.12.

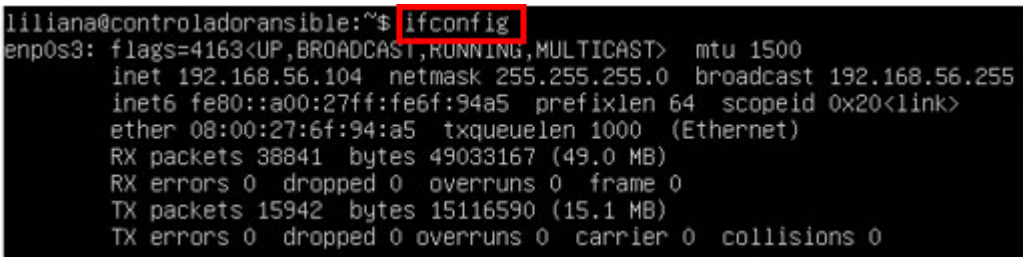


**Figura 3.12** Direccionamiento IP del controlador

## 3.3 Instalación de la herramienta de Ansible

Antes de realizar la instalación del Ansible en la máquina que funciona como la máquina central o controlador de Ansible, se ejecutó la actualización de la aplicación sudo para obtener todos los permisos correctos con el comando ***sudo apt update*** y así evitar problemas cuando se instalen diferentes paquetes, se sugiere también actualizarlo con ***sudo apt upgrade***.

Para instalar la herramienta se ejecuta el comando ***sudo apt install software-properties-common***, tal como se observa en la Figura 3.13.

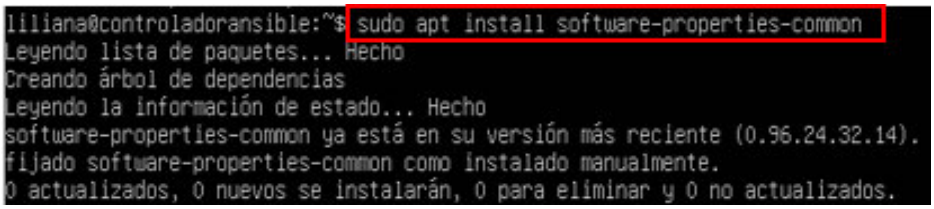


**Figura 3.13** Instalación de Ansible en el nodo controlador

Además se ejecutó el siguiente comando ***sudo add-apt-repository --yes --update ppa:ansible/ansible*** para incluir los paquetes personales de Ansible (PPA), en la lista de fuentes del sistema, como se observa en la Figura 3.14.

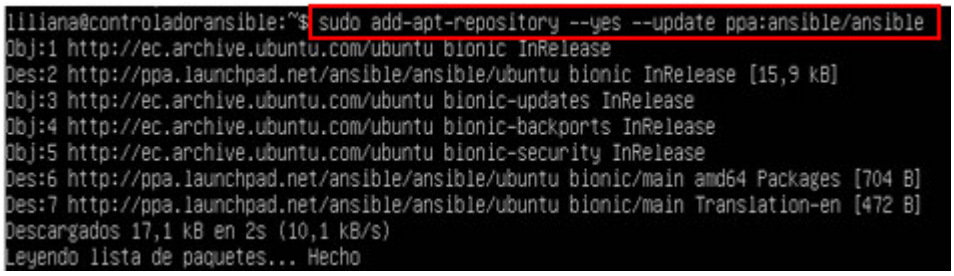


**Figura 3.14** Instalación de PPA en el controlador

Luego se ejecutó el comando ***sudo apt install ansible***, para instalar el *software* de la herramienta Ansible y de esta manera se cuenta con todo el *software* necesario para administrar al nodo *host,* ver Figura 3.15.


**Figura 3.15** Instalación del *software* Ansible

Con el fin de verificar que se encuentra instalado correctamente la herramienta Ansible, se procedió a ejecutar el siguiente comando ***ansible --version***, la respuesta permite conocer si Ansible se encuentra instalada correctamente, puesto que se verifica la versión, así como la instalación de Python, como se puede observar en Figura 3.16.

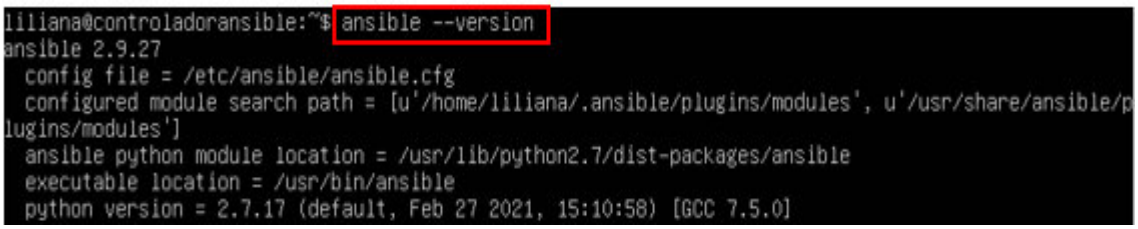


**Figura 3.16** Comprobación de instalación correcta para Ansible

### Generación de llaves SSH en el controlador

Después de tener la instalación completa de Ansible, se generó las llaves SSH para el nuevo usuario y establecer la conexión. Para ello se necesita el comando que se muestra en la Figura 3.17, de esta manera se conoce dónde se creó, es decir en el *home,* que es un directorio oculto de SSH, el cual se lo deja sin frase para así conectarse de una manera más fácil.

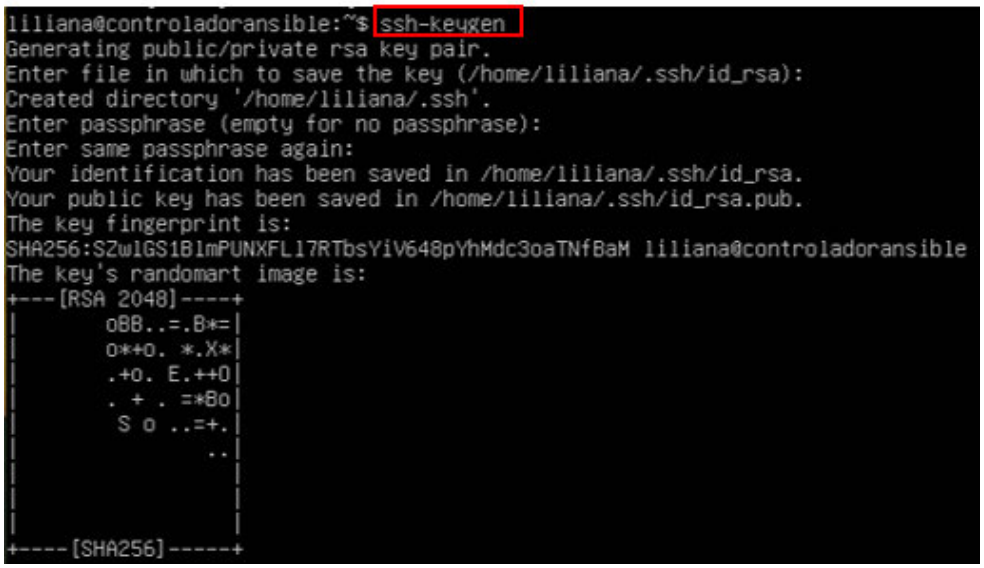


**Figura 3.17** Generación de claves SSH

Luego se copió las llaves SSH ID del nodo *host,* con el comando ***ssh-copy-id username @ip-address*** que se lo reemplaza por el nombre del usuario y dirección IP del nodo *host*, como se puede evidenciar en la Figura 3.18.

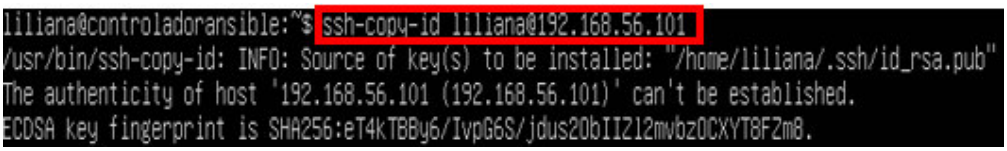


**Figura 3.18** Copia de llaves SSH ID

Una vez autentificada la IP del nodo *host*, se confirma que se quiere continuar con la conexión, mediante la palabra ***yes***, luego se introduce la contraseña que se configuró en la instalación de la máquina nodo *host*; finalmente se obtiene el número de llaves añadidas mediante SSH, en este caso es solamente una y se observa en la Figura 3.19.

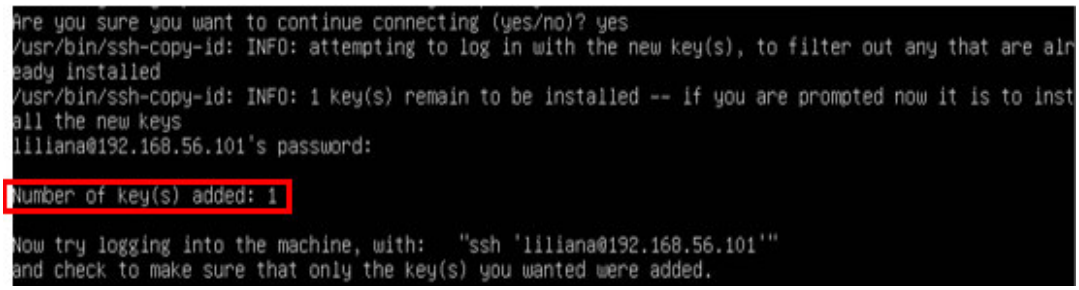


**Figura 3.19** Llave añadida mediante SSH

### Autorización de llaves SSH en el nodo *host*

Finalmente, para obtener la conexión remota SSH, es importante crear un directorio SSH en el nodo *host* después de haber generado las llaves SSH en la máquina central o controlador. Esto se realiza mediante el comando ***ssh username@ip-address mkdir –p .ssh,*** el cual será reemplazado con los datos necesarios, como se puede evidenciar en la Figura 3.20.

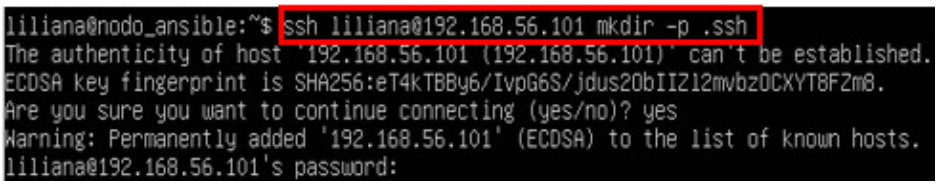


**Figura 3.20** Creación del directorio SSH

Una vez creado el directorio, se procede a verificar en el controlador mediante el comando ***cd .ssh***, el cual ayuda a ingresar al directorio de SSH, para después mediante ***ls*** verificar mediante el listado de contenido si se añadió el archivo *autorized_keys,* el cual se encarga de resguardar la llave pública, que es utilizada en OpenSSH para la autenticación. Ver Figura 3.21.

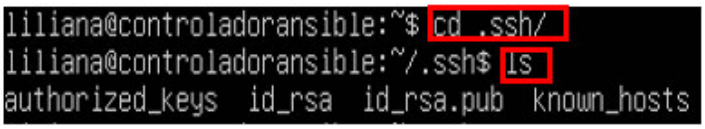


**Figura 3.21** Listado del directorio SSH

### Configuración del archivo inventario

El archivo inventario contiene la información acerca del *host* que se administra con Ansible. Este archivo de inventario se utiliza también para la configuración de *playbooks* y plantillas.

Para su edición, se abrió el archivo inventario, lo cual se lo hizo a través del comando ***sudo nano /etc/ansible/hosts***, una vez dentro se definió el grupo en el cual estará el

nodo *host* personalizado: nodo1, con su respectiva IP, como se evidencia en la Figura 3.22.

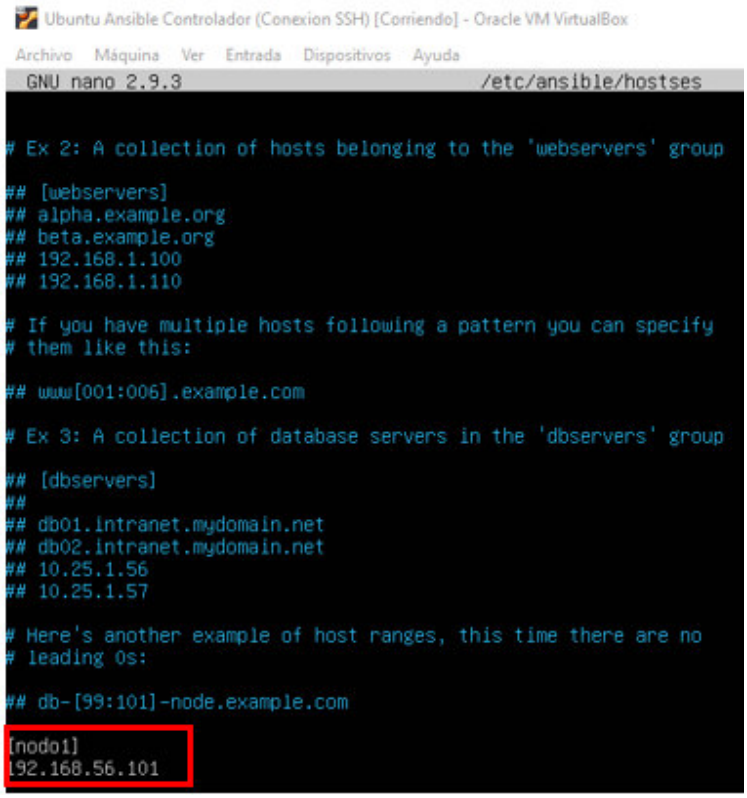


**Figura 3.22** Creación del inventario

Una vez que se estableció en el inventario el nodo *host,* se guardó las modificaciones realizadas con CTRL+X, luego ***yes*** para confirmarlo y dando Enter. Para consultar el inventario se lo realizó de la siguiente manera, ver Figura 3.23, donde se verifican los nodos que se tiene como *hosts,* con su respectiva IP y el nombre que se le asignó.

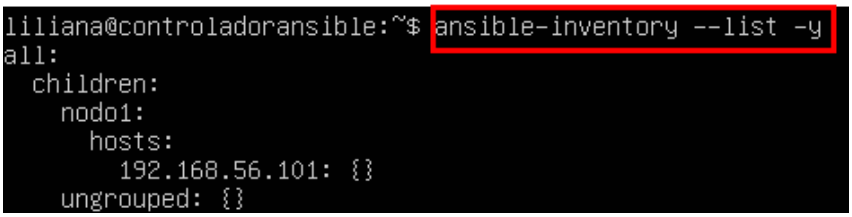


**Figura 3.23** Lista de inventario

Con estas configuraciones se procedió a probar conectividad con el nodo *host*, realizando *ping* desde el controlador al nodo y viceversa, como se evidencia en la Figura 3.24 y Figura 3.25.

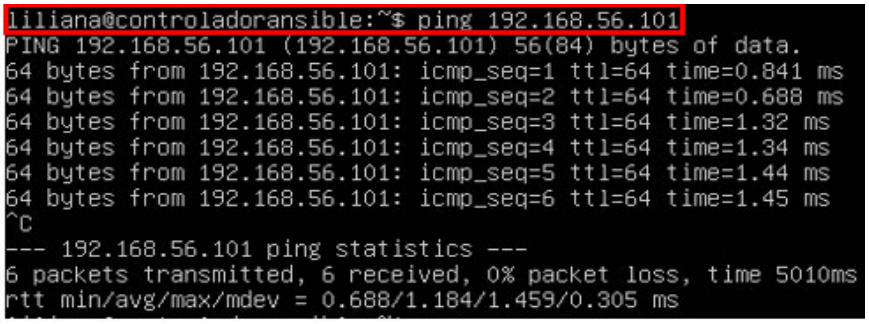


**Figura 3.24** *Ping* desde el controlador hacia el nodo *host*

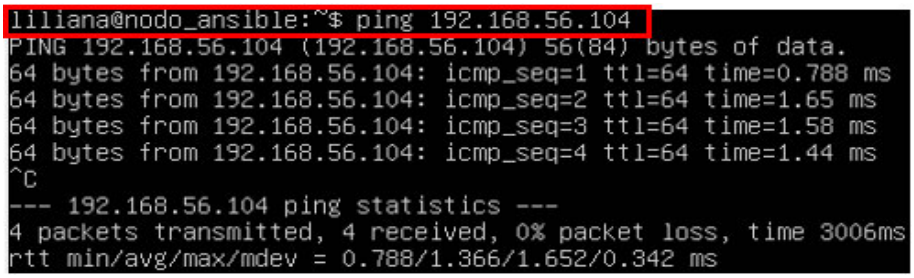


**Figura 3.25** *Ping* desde el nodo *host* hacia el controlador

Así mismo se verifica la conexión SSH entre el controlador y el nodo, mediante el comando ***ansible all –m ping***, el cual probó que sí es posible el acceso al nodo *host,* las credenciales de SSH son válidas, de esta manera el nodo *host* puede ejecutar los módulos de Ansible que utiliza Python.

Al conectarse al nodo *host* se recibe la respuesta *"pong"*, como se ilustra en la Figura 3.26; *Pong* significa que se está listo para la ejecución de comandos y *playbooks* de Ansible en el servidor [22].

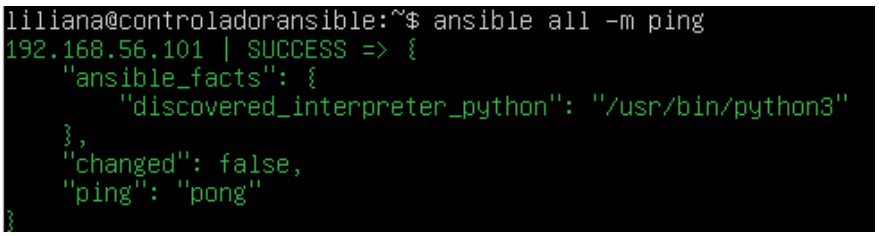


**Figura 3.26** Conexión SSH entre el controlador y nodo

## 3.4 Implementación de un *firewall* mediante la herramienta de Ansible

Para la implementación de un *firewall* se requiere realizar en primera instancia el *playbook,* el cual contendrá varias tareas definidas; esto se realizó en el controlador y mediante la conexión SSH se ejecuta el *firewall* en el nodo *host,* logrando así el despliegue del mismo.

Para establecer el *playbook* se crea un archivo *nano* mediante el comando ***sudo nano*** seguido del nombre o ruta que se le dé al archivo; al cual se lo denomina ***playbook-firewall-apache,*** como se observa en la Figura 3.27. Se lo realizó con el lenguaje YAML; dicho formato describe las acciones que se realizan y configuraciones que se propagan en el nodo *host.*

```
liliana@controladoransible:~$ sudo nano playbook-firewall-apache.yaml
```

**Figura 3.27** Creación del *playbook*

A continuación, la Figura 3.28 muestra el *playbook* creado*,* dentro de este archivo se procedió a detallar las tareas que se requieren ejecutar en el nodo *host.* Además, en la Tabla 3.1 se detalla la función de cada línea del *playbook.* Cabe recalcar que en la primera línea del *playbook* se debe comenzar con tres guiones medios (- - -) que indica el inicio de un documento YAML.

```
 GNU nano 2.9.3                    playbook-firewall-apache.yaml

---
- hosts: all
  name: Install
  become: yes
  tasks:
          - name: Update apt packages
            apt:
              state: latest

          - name: Install UFW Firewall
            apt:
              state: latest

          - name: Allow SSH ufw
            ufw:
              state: enabled
              rule: allow
              port: "{{ item }}"
              proto: tcp
            with_items:
               - "22"

          - name: Allow HTTP ufw
            ufw:
              state: enabled
              rule: allow
              port: "{{ item }}"
              proto: tcp
            with_items:
               - "80"
               - "443"

          - name: Allow SMTP ufw
            ufw:
              state: enabled
              rule: allow
              port: "{{ item }}"
              proto: tcp
            with_items:
               - "25"

          - name: Allow FTP ufw
            ufw:
              state: enabled
              rule: allow
              port: "{{ item }}"
              proto: tcp
            with_items:
               - "20"
               - "21"
```

**Figura 3.28** Tareas del *playbook*

**Tabla 3.1** Funciones de las tareas del *playbook*

| **Parámetros** | | **Descripción** | |
|---|---|---|---|
| *hosts* | *all* | Indica el grupo de *hosts* en el que se va a ejecutar el *playbook,* en este caso se tiene solo uno; se coloca *all* para no hacer ningún cambio en esta sección si se agrega más nodos *hosts.* | |
| *name* | *install* | Se indica el nombre de la jugada o (*play).* | |
| *become* | *yes* | Ayuda a la ejecución de todos los comandos como un usuario *sudo.* | |
| *tasks* | | Son las tareas que se ejecutan; en este caso son seis tareas, las cuales están definidas mediante un guión medio. | |
| | Nombre de la tarea 1<br>*- name: Update apt packages,* | Sirve para la actualización del caché del repositorio.<br>*state: latest,* se activa mediante la tarea y se ejecuta al final de la misma, es decir notifica al servicio para que se inicie una vez que se instale. | |
| | Nombre de la tarea 2<br>*- name: Install UFW Firewall* | Se instala el corta fuegos UFW<br>*state: latest,* sirve para notificar al servicio para que se inicie UFW una vez que se instale. | |
| | Nombre de la tarea 3<br>*- name: Allow SSH ufw* | Se definen las reglas para el *firewall* SSH<br>*ufw:* | |
| | | *state: enabled* | Cadena de estado, recarga y habilita el *firewall* en el arranque. |
| | | *rule: allow* | Cadena de regla para permitir la apertura del puerto. |
| | | *port: "{{ item }}"* cadena de puerto | Lista de elementos en bucle del puerto. |
| | | *proto:* tcp | Cadena de protocolo. |
| | | *with_items:*<br>- "22” | Bucle con la lista de puertos a utilizarse. |
| | Nombre de la tarea 4<br>*- name: Allow* HTTP *ufw* | Se definen las reglas para el *firewall* HTTP<br>*ufw:* | |
| | | *state: enabled* | Cadena de estado, recarga y habilita el *firewall* en el arranque. |
| | | *rule: allow* | Cadena de regla para permitir la apertura del puerto. |
| | | *port: "{{ item }}"* cadena de puerto | Lista de elementos en bucle del puerto. |
| | | *proto:* tcp | Cadena de protocolo. |
| | | *with_items:*<br>- "80”<br>- "443” | Bucle con la lista de puertos a utilizarse. |
| | Nombre de la tarea 5<br>*- name: Allow* SMTP *ufw* | Se definen las reglas para el *firewall* SMTP<br>*ufw:* | |
| | | *state: enabled* | Cadena de estado, recarga y habilita el *firewall* en el arranque. |

| | | | |
|---|---|---|---|
| | | *rule: allow* | Cadena de regla para permitir la apertura del puerto. |
| | | *port: "{{ item }}"* cadena de puerto | Lista de elementos en bucle del puerto. |
| | | *proto:* tcp | Cadena de protocolo. |
| | | *with_items:*<br>- "25" | Bucle con la lista de puertos a utilizarse. |
| | Nombre de la tarea 6<br>*- name: Allow* FTP *ufw* | Se definen las reglas para el *firewall* FTP<br>*ufw:* | |
| | | *state: enabled* | Cadena de estado, recarga y habilita el *firewall* en el arranque. |
| | | *rule: allow* | Cadena de regla para permitir la apertura del puerto. |
| | | *port: "{{ item }}"* cadena de puerto | Lista de elementos en bucle del puerto. |
| | | *proto:* tcp | Cadena de protocolo. |
| | | *with_items:*<br>- "20"<br>- "21" | Bucle con la lista de puertos a utilizarse. |

### Despliegue del *firewall* en el nodo *host*

Como primer paso se verifica que no está funcionando UFW en el nodo *host*, para ello se coloca el comando ***sudo ufw status,*** y se comprueba que UFW está inactivo, esto se observa en la Figura 3.29.

```
liliana@nodo_ansible:~$ sudo ufw status
Status: inactive
```

**Figura 3.29** UFW inactivo en el nodo *host*

Luego de esta verificación, y antes de implementar el *firewall* se debe probar la conexión SSH entre el controlador y el nodo *host,* para no tener problemas con la autenticación de las llaves SSH. Esto mediante el comando que se observa en la Figura 3.30.

```
liliana@controladoransible:~$ ansible all -m ping_
```

**Figura 3.30** Conexión SSH entre el controlador y nodo *host*

Para desplegar el *firewall,* se ejecutó el comando ***sudo ansible-playbook*** seguido del nombre del archivo *nano,* especificado anteriormente, en el cual se encuentra el código a ejecutar, ver Figura 3.31. Se coloca al final –K para solicitar la contraseña e ingresar como usuario administrador de Ansible y ejecutar el *playbook* [23].

```
liliana@controladoransible:~$ sudo ansible-playbook playbook-firewall-apache.yaml -K
BECOME password:
```

**Figura 3.31** Ejecución del *playbook*

De esta manera, se despliegan los cambios que se pueden observar en la Figura 3.32, se instala y ejecuta cada una de las tareas descritas en el *playbook,* dando como resultado los cambios en color verde en cuanto a la instalación para la actualización del caché del repositorio, la instalación de UFW y se ejecuta de manera predeterminada la tarea *Gathering Facts* (esta tarea ayuda a Ansible con la recopilación informativa acerca del *host* remoto, como lo es la dirección IP). En color amarillo se muestran los cambios en cuanto a los puertos que fueron habilitados en el *playbook.*

En cuanto al *Play Recap* se tiene un resumen de las modificaciones que se realizaron en el nodo *host*, lo cual significa que:

- *ok*: Se ejecutó 7 tareas sin errores.
- *changed:* Se ejecutó 4 tareas y se realizó los cambios en el nodo *host.*

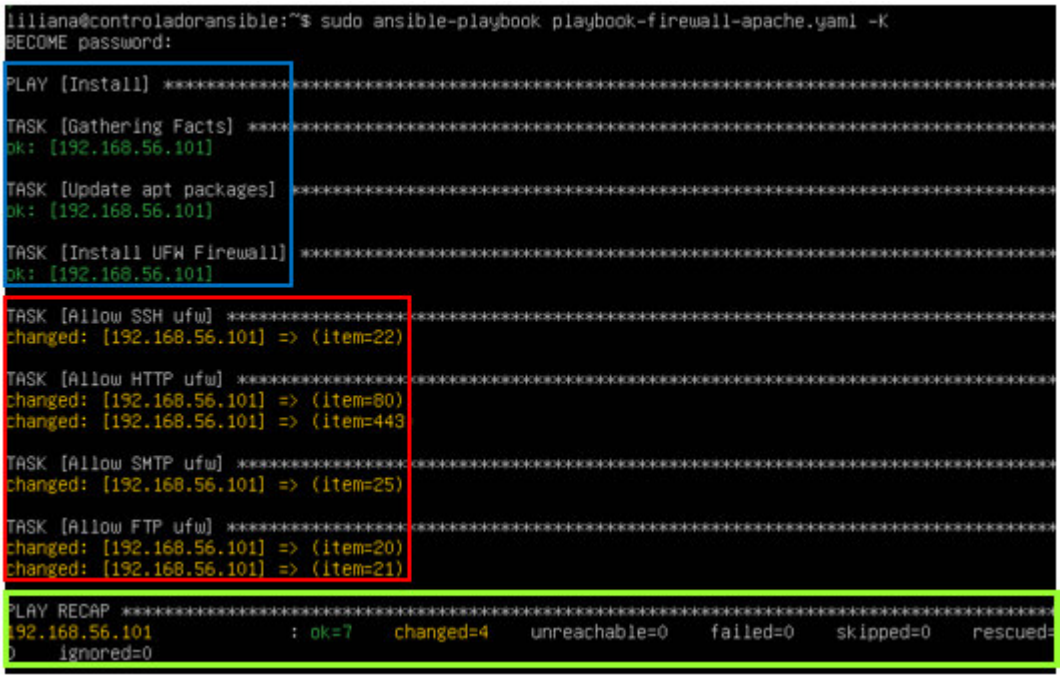


**Figura 3.32** Implementación del *firewall* mediante el *playbook*

Una vez realizada la implementación del *firewall* mediante el *playbook,* se comprueba el despliegue correcto en el nodo *host,* esto se lo realizó con el comando ***sudo ufw status***, el cual ayudó a verificar el estado del UFW y a su vez el despliegue del *playbook* con cada una de sus tareas, como se visualiza en la Figura 3.33.

En la primera columna se observa los puertos que se aplicaron en el *playbook,* tanto en IPv4 e IPv6, en la segunda columna se tiene la acción, es decir que permiten el acceso a servicios que ofrecen dichos puertos, y en la tercera columna se observa *Anywhere* lo cual significa que estas reglas se aplicarán a cualquier *host* donde se ejecute este *playbook*, ya que se colocó en el mismo *hosts: all*.

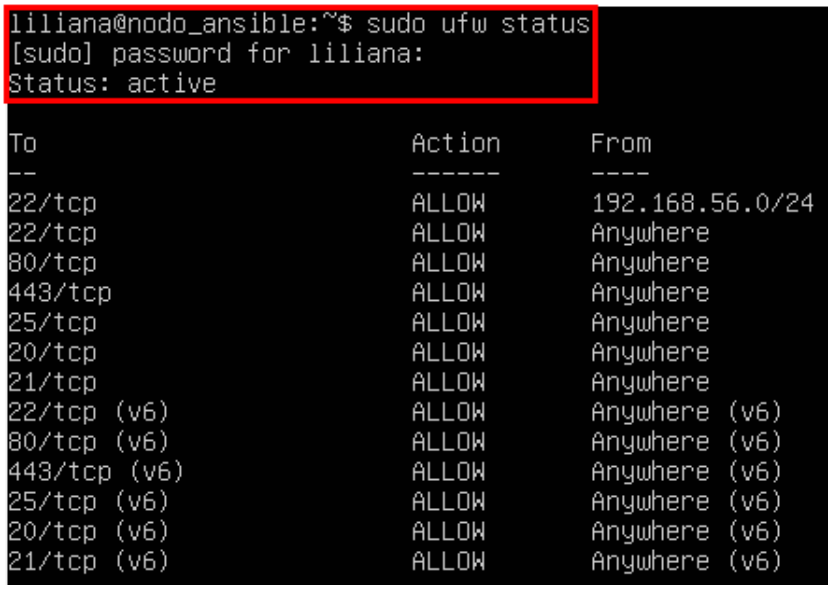


**Figura 3.33** Despliegue del *firewall* en el nodo *host*

## 3.5 Pruebas de funcionamiento

### Pruebas de funcionamiento de la herramienta de Ansible

En la máquina del controlador, se ejecuta el comando ***ansible --version*** con el fin de observar que Ansible está activado y la versión que tiene el mismo, así como también el módulo de Python, ver Figura 3.34.

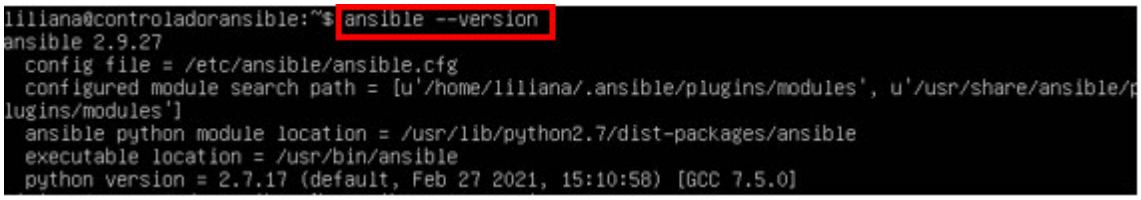


**Figura 3.34** Prueba de funcionamiento de Ansible

Se ejecuta igualmente el comando ***ansible all –m ping***, para observar que la comunicación SSH está activa con el nodo *host,* el cual tiene la dirección IP 192.168.56.101, se confirma con éxito y se evidencia en la Figura 3.35.


**Figura 3.35** Prueba de funcionamiento de conexión SSH

Verificado esto se puede realizar el despliegue del *firewall* y la verificación de su funcionamiento.

### Pruebas de funcionamiento del *firewall* implementado

Para las pruebas de funcionamiento y análisis de los resultados, del despliegue del *firewall* mediante la herramienta de Ansible, se procede a modificar el *playbook* creado anteriormente en la Figura 3.28; esto con el fin de visualizar los cambios ejecutados y nuevos resultados. Ver la Figura 3.36. Se modificó la tarea cuatro, cinco y seis donde se deniega el acceso a los puertos designados mediante la regla *deny* (cerrado).


**Figura 3.36** Modificación del *playbook*

Después de guardar los cambios realizados en el *playbook,* se procede a ejecutarlo en la máquina central o controlador con total éxito, desplegando el *firewall* en el nodo *host*, ver Figura 3.37.

De esta manera, se instala el *play,* como se indicó anteriormente se establece de forma predeterminada la tarea *Gathering Facts*, la cual muestra la dirección IP 192.168.56.101 perteneciente al nodo *host*. A continuación, se ejecutan cada una de las tareas descritas en el *playbook;* en color verde se tiene la instalación para actualizar el caché del repositorio, la instalación de UFW, la tarea *Gathering Facts* y finalmente el puerto SSH habilitado. En color amarillo se puede apreciar los cambios que se realizaron en las tareas del *playbook,* respecto a los puertos denegados.

El resumen del despliegue se muestra en el *Play Recap*, lo cual significa que:

- *ok*: Se ejecutó 7 tareas sin errores.
- *changed:* Se ejecutó 3 tareas y se realizó los cambios en el nodo *host.*
  Es decir, se obtiene un cambio menos que en la Figura 3.32, ya que el puerto 22 quedó habilitado sin modificación y los demás puertos se denegaron.

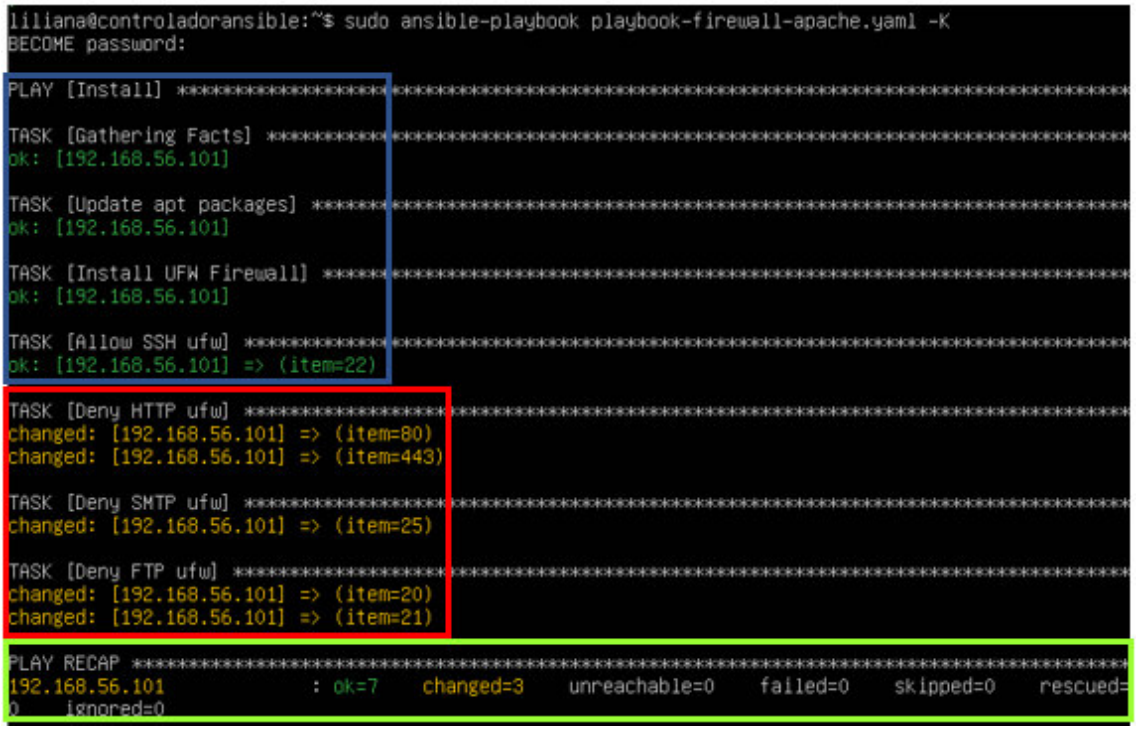


**Figura 3.37** Ejecución del *playbook* modificado

A su vez se ejecutó el comando de ***sudo ufw status*** en la máquina nodo, con el fin de comprobar el estado del mismo en el nodo *host*, ver Figura 3.38.

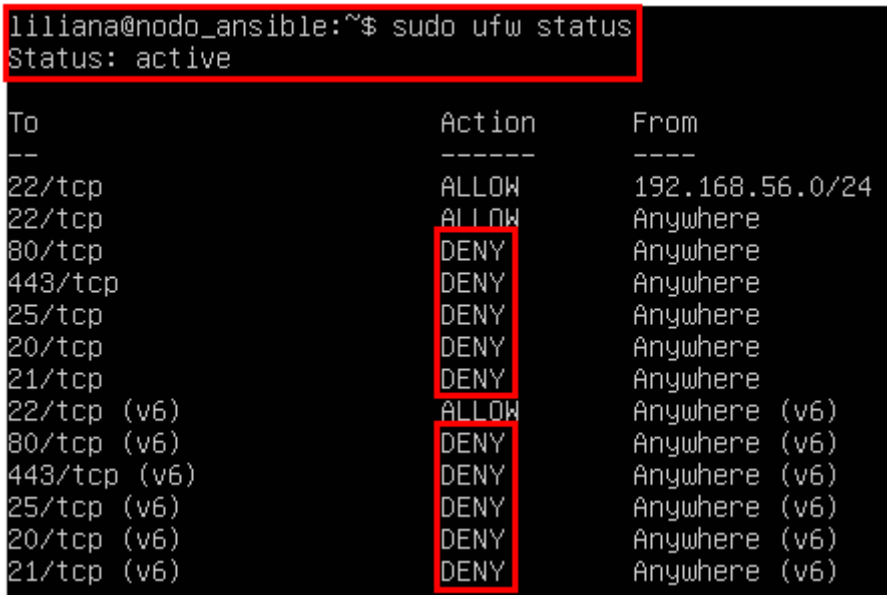


**Figura 3.38** Estado del UFW con las modificaciones respectivas

De igual forma mediante Netcat se prueba el funcionamiento correcto del *firewall* implementado, puesto que sirve para detectar los errores o problemas que afectan al funcionamiento y seguridad óptima de una red.

El comando a ser utilizado es ***nv –zv (IP o Dominio) (Puerto)***, *se coloca* **v** para observar si el puerto está abierto o no, mientras tanto la **z** ayuda a cerrar la conexión una vez realizada la consulta ejecutada por Netcat. De esta manera se puede observar en la Figura 3.39, que el puerto 22 de SSH se encuentra abierto, obteniendo una respuesta de *succeeded*, que significa que se logró la conexión.

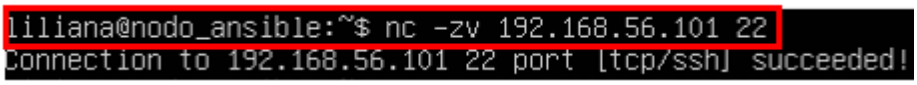


**Figura 3.39** Prueba de funcionamiento mediante Netcat – Puerto 22

De igual forma se procede con el puerto 25 de SMTP *(Simple Mail Transfer Protocol),* el puerto 80 de HTTP (*Hypertext Transfer Protocol*), puerto 443 de HTTPS *(HyperText Transfer Protocol Secure)*, puerto 20 de FTP *Data (File Transfer Protocol Data)* y el puerto 21 FTP *Control (File Transfer Protocol Control*). Se verifica que se encuentran bloqueados ya que se obtiene como respuesta *Connection refused* lo cual significa conexión denegada, ver Figura 3.40.

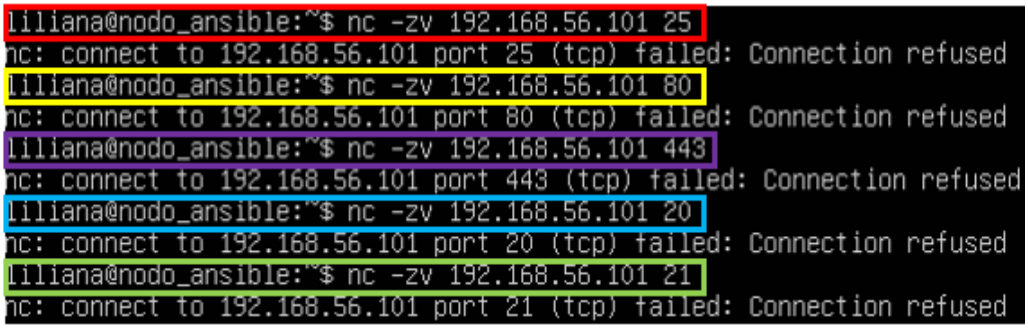


**Figura 3.40** Prueba de funcionamiento mediante Netcat – Conexión denegada

Otra prueba se la realizó mediante Telnet, la cual se encarga de determinar si algún puerto está habilitado o deshabilitado; mediante el comando ***telnet (IP o Dominio) (Puerto)*** que ayuda a probar la conectividad del servidor remoto o un puerto determinado. De lo cual se evidencia en la Figura 3.41 que el puerto 25, 80, 443, 20 y 21 se encuentra bloqueado mediante la respuesta *Connection refused.*

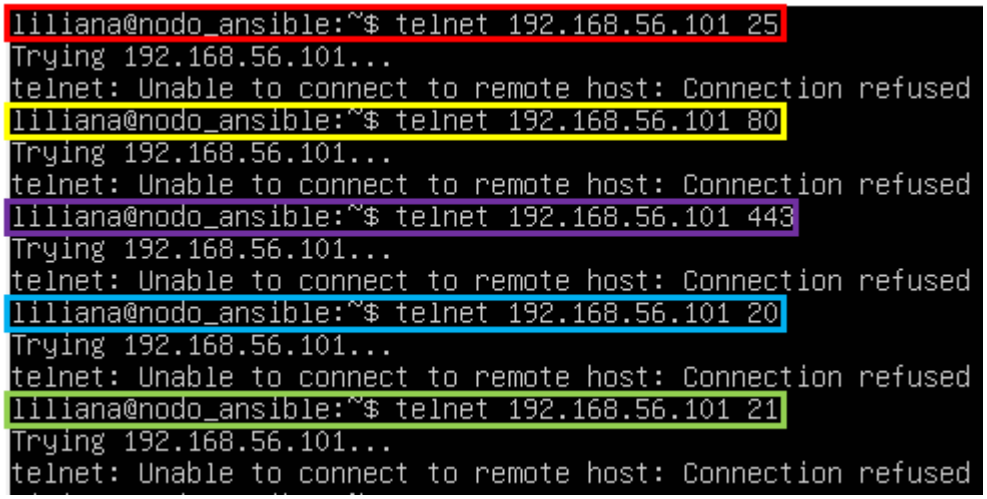


**Figura 3.41** Prueba de funcionamiento con Telnet – Denegación

Mientras tanto que para el puerto 22, que corresponde al protocolo SSH, se obtiene que no tiene ningún problema al conectarse, mediante el mensaje que se observa en la Figura 3.42.

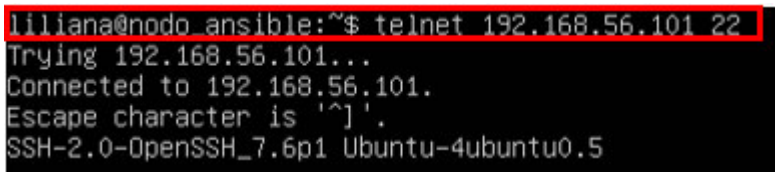


**Figura 3.42** Prueba de funcionamiento con Telnet – Puerto 22

Se realizó una segunda prueba de funcionamiento modificando el *playbook,* como se muestra en la Figura 3.43. En el cual se mantienen las tres primeras tareas, en la tarea cuatro se agregó y denegó el puerto de MSP *(Message Send Protocol)*, en cuanto a las tareas cinco y seis fueron eliminadas con el fin de observar los cambios que se dan después de ejecutar el *playbook*.

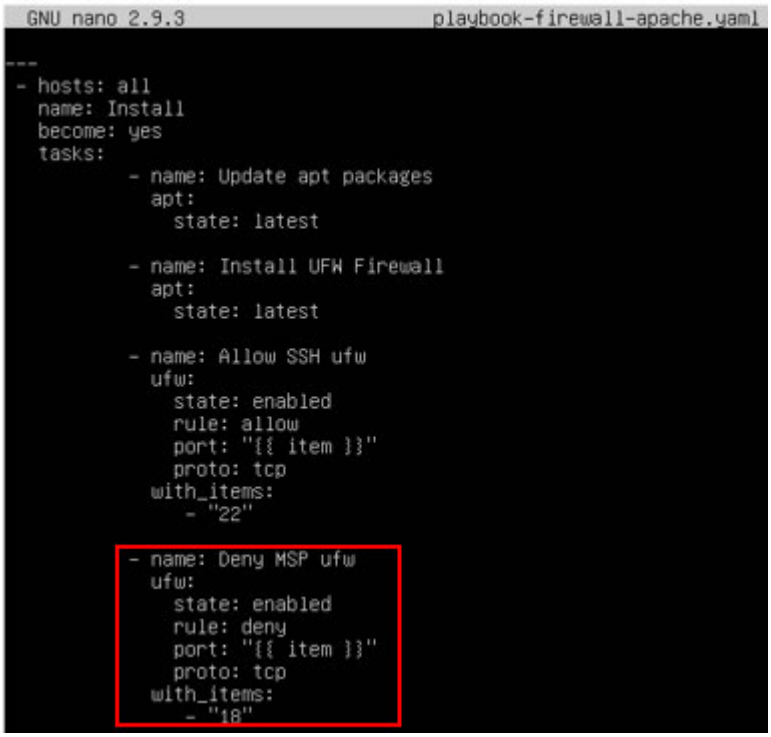


**Figura 3.43** Segunda modificación de *playbook*

Luego de guardar los cambios realizados en el *playbook,* se lo ejecutó con total éxito, esto se lo visualiza en la Figura 3.44. De este modo, se instala el *play,* se ejecutan cada una de las tareas descritas en el *playbook;* el cambio en color amarillo se puede apreciar tanto en el *task* como en el *Play Recap*, lo cual consta de:

- *ok*: Se ejecutó 5 tareas sin errores.
- *changed:* Se ejecutó 1 tarea y se realizó los cambios en el nodo *host.*

Es decir, se tiene un solo cambio (denegación del protocolo MSP); los puertos que anteriormente se denegaron se quedan denegados, esto se lo comprobó aplicando ***sudo ufw status*** en el nodo host, como se puede apreciar en la Figura 3.45.

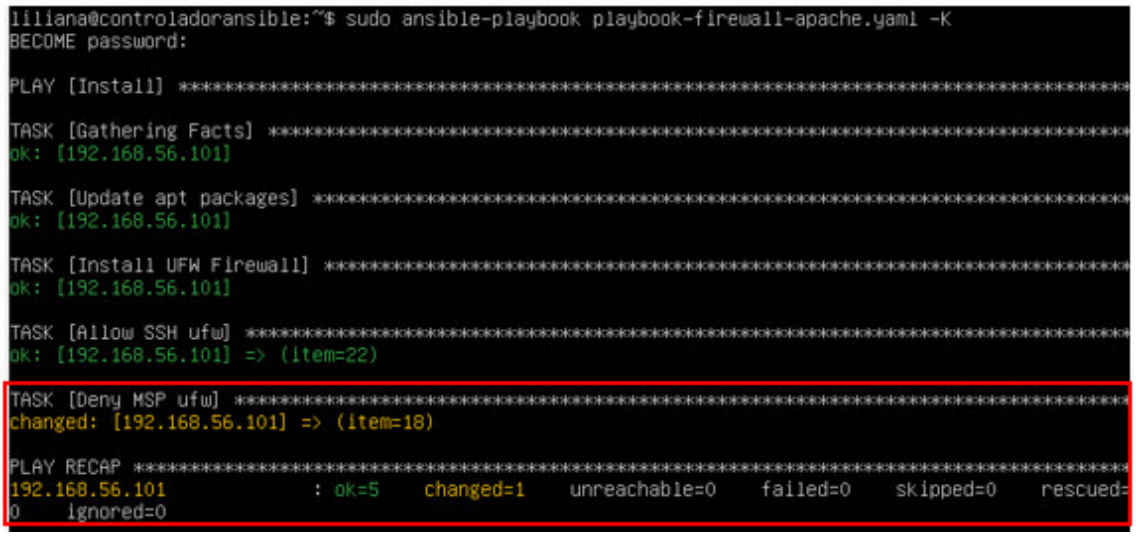


**Figura 3.44** Ejecución del *playbook* modificado, según la prueba dos

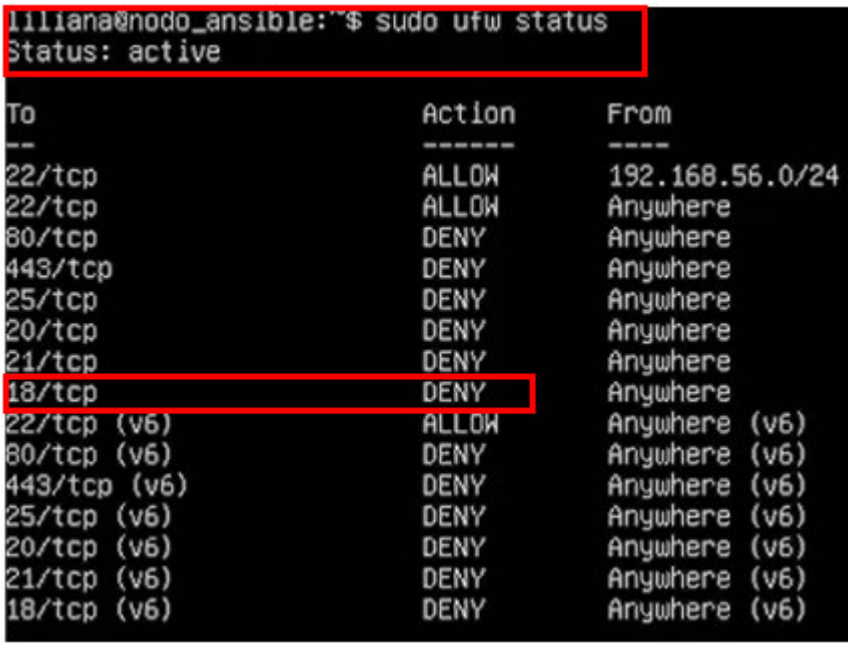


**Figura 3.45** Estado del UFW modificado, según la prueba dos

De la misma manera mediante Netcat se prueba el funcionamiento correcto del *firewall* desplegado, como se puede observar en la Figura 3.46, en la que se comprueba que el puerto 22 está habilitado, al contrario que el puerto 18 de MSP que se encuentra denegado.

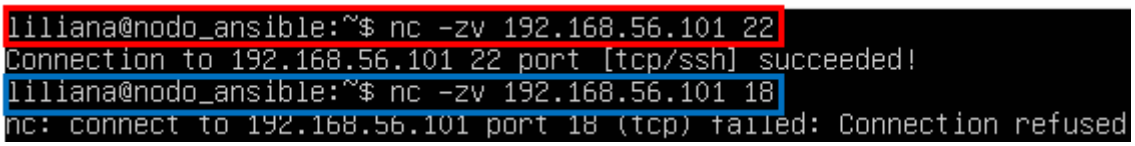


**Figura 3.46** Segunda prueba de funcionamiento mediante Netcat

Así mismo mediante Telnet se realiza la verificación, en la Figura 3.47 se puede confirmar que el puerto 22 está habilitado, mientras que el puerto 18 se encuentra bloqueado.

```
liliana@nodo_ansible:~$ telnet 192.168.56.101 22
Trying 192.168.56.101...
Connected to 192.168.56.101.
Escape character is '^]'.
SSH-2.0-OpenSSH_7.6p1 Ubuntu-4ubuntu0.5
^C
Connection closed by foreign host.
liliana@nodo_ansible:~$ telnet 192.168.56.101 18
Trying 192.168.56.101...
telnet: Unable to connect to remote host: Connection refused
```

**Figura 3.47** Segunda prueba de funcionamiento mediante Telnet

Finalmente, con una última prueba de funcionamiento se agregó la tarea cinco, en la cual se denegó el puerto 143 de IMAP *(Internet Message Access Protocol)*, ver Figura 3.48.

```
 GNU nano 2.9.3                    playbook-firewall-apache.yaml

---
- hosts: all
  name: Install
  become: yes
  tasks:
          - name: Update apt packages
            apt:
              state: latest

          - name: Install UFW Firewall
            apt:
              state: latest

          - name: Allow SSH ufw
            ufw:
              state: enabled
              rule: allow
              port: "{{ item }}"
              proto: tcp
            with_items:
               - "22"

          - name: Deny MSP ufw
            ufw:
              state: enabled
              rule: deny
              port: "{{ item }}"
              proto: tcp
            with_items:
               - "18"

          - name: Deny IMAP ufw
            ufw:
              state: enabled
              rule: deny
              port: "{{ item }}"
              proto: tcp
            with_items:
               - "143"
```

**Figura 3.48** Tercera modificación de *playbook*

Al guardar las modificaciones realizadas en el *playbook* se lo aplica de manera exitosa y se lo visualiza en la Figura 3.49. Así mismo en este despliegue se ejecutan cada una de las tareas descritas y modificadas en el *playbook,* los cambios se visualizan en color amarillo. En el *Play Recap* se obtiene lo siguiente:

- *ok*: Se ejecutó 6 tareas sin errores.
- *changed:* Se ejecutó 1 tarea y se realizó los cambios en el nodo *host.*

Dicho de otro modo, se tiene como resultado un solo cambio ya que solo se agregó la denegación del puerto 143 (IMAP), pero en este caso son seis tareas las que se ejecutaron en total. Los puertos anteriores que se implementaron y denegaron en la primera prueba de funcionamiento, como lo son: 20, 21, 25, 80 y 443, se quedaron guardados en su estado inicial, esto se puede observar en la Figura 3.50 luego de aplicar el comando ***sudo ufw status*** en el nodo *host.*

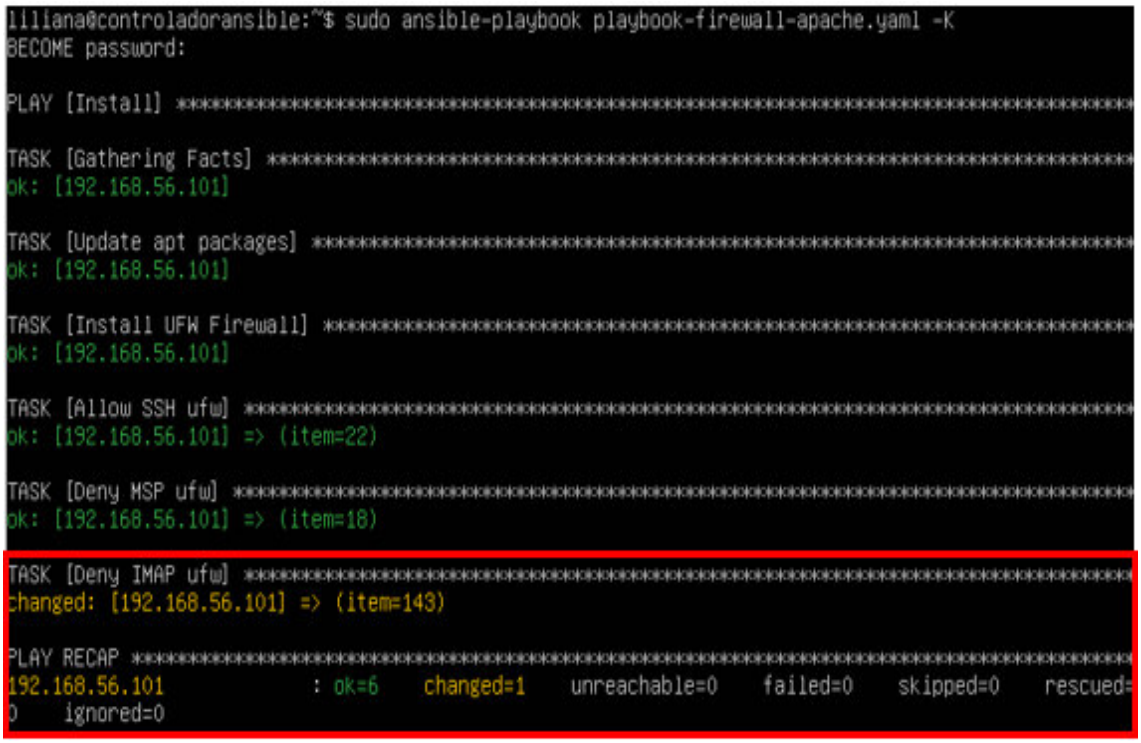


**Figura 3.49** Ejecución del *playbook* modificado, según la prueba tres

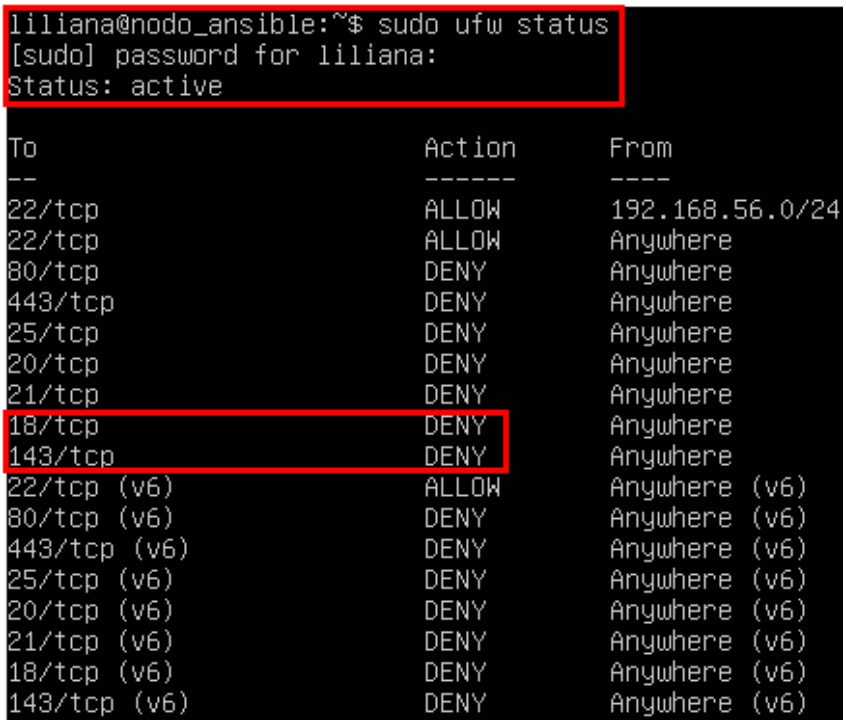


**Figura 3.50** Estado del UFW modificado, según la prueba tres

Para comprobar el funcionamiento correcto del *firewall* implementado se lo realizó mediante Netcat, se comprueba que el puerto 22 está habilitado, como se observa en la Figura 3.51, al contrario que el puerto 18 de MSP y 143 de IMAP que se encuentran denegados, como se comprueba en la Figura 3.52.

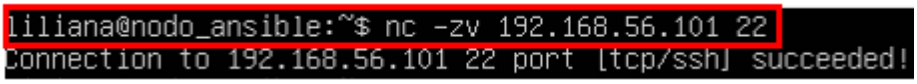


**Figura 3.51** Tercera prueba de funcionamiento mediante Netcat – Puerto 22

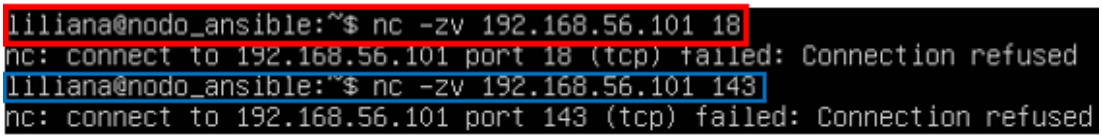


**Figura 3.52** Tercera prueba de funcionamiento mediante Netcat – Denegación

Se utilizó también Telnet, en la Figura 3.53 se puede confirmar que el puerto 22 está habilitado, mientras que el puerto 18 y 143 se encuentran denegados, ver Figura 3.54.

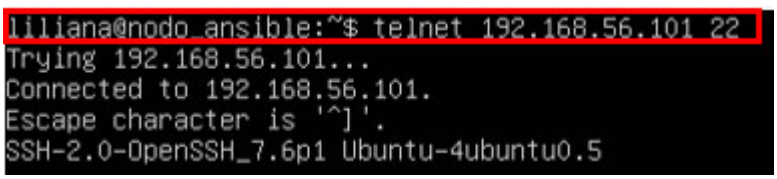


**Figura 3.53** Tercera prueba de funcionamiento mediante Telnet – Puerto 22

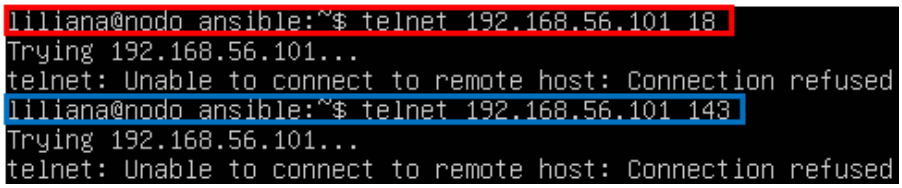


**Figura 3.54** Tercera prueba de funcionamiento mediante Telnet – Denegación

A través del código QR mostrado en el Anexo II, se puede evidenciar la implementación y las pruebas de funcionamiento del presente proyecto de titulación.

# 4 CONCLUSIONES

- Para la creación de las máquinas virtuales para el nodo *host* y el controlador, en el presente proyecto se utilizó la distribución Ubuntu *Server* 18.04, ya que tiene variedades propias para servidores, incluye paquetes como lo es OpenSSH que permitió la conexión remota entre la máquina del nodo *host* y el controlador.
- Para la comunicación en red de la máquina controlador y el nodo *host* es necesario la configuración de las Preferencias de red en VirtualBox, en las dos máquinas debe ser la misma con el fin de que se encuentren en red.
- Para llevar a cabo la instalación de Ansible, no es necesario instalarlo en todas las máquinas; Ansible no requiere de agentes, por lo cual solo se lo instala en una máquina (controlador) que es la encargada de manejar a las demás.
- Ansible busca cifrar la información y tener un protocolo de comunicación seguro, por lo que SSH es la mejor opción para la comunicación entre la máquina controlador y el nodo *host*, ya que es seguro y trabaja con distintos tipos de encriptación.
- Para la instalación de Ansible, es necesario verificar que Python también se encuentre instalado, puesto que la herramienta Ansible está desarrollada en base a Python; siendo así un punto clave para la ejecución correcta de la misma.
- La herramienta Ansible es fundamental en cuanto a la automatización de seguridad, puesto que se disminuye la intervención humana, agilizando así los procesos para identificar o validar las amenazas de manera rápida y sin necesidad de alguna intervención manual.
- Para ejecutar el *playbook* es necesario colocar en el comando ***–K*** esto es importante puesto que ayuda a tener acceso de usuario sudo mediante la

contraseña que se configuró en el perfil de usuario de la máquina del nodo *host;* sin dicho comando no se puede ejecutar el *playbook*.

- Los *playbooks* ayudan a simplificar el trabajo mediante la creación de una propia lista de tareas, de esta manera se logra la automatización de manera remota mediante roles y tareas específicas.
- El *firewall* UFW ayuda a generar eficientes reglas de un *firewall* de una manera eficiente; existe dos maneras diferentes para su instalación, en este caso se utilizó Ansible *playbook* donde en una de las tareas se ejecutó su instalación.
- Se realizó también la ejecución de *FirewallD*, sin tener un buen resultado debido a que al momento de visualizar el estado de los puertos mediante *iptables,* no se obtuvo éxito, puesto que no se generó la lista completa para proceder a bloquear los puertos que se eligieron para el presente proyecto.
- A partir de las pruebas de funcionamiento realizadas, se establece el correcto despliegue del *firewall,* así como también la verificación del mismo, comprobado mediante el cierre o apertura de puertos o protocolos especificados dentro de la tarea del *playbook.*
- Para la verificación del correcto despliegue del *firewall*, como primera instancia y siendo parte de UFW se lo hizo en el nodo *host* mediante el comando *sudo ufw status,* esto con el fin de observar el estado de los puertos que se ejecutaron en el *playbook*; otra manera de verificación fue mediante Netcat y Telnet, ya que estos se encargan de diagnosticar el estado y funcionalidad de la red.

# 5 RECOMENDACIONES

- Es recomendable que las máquinas virtuales de Linux se mantengan actualizadas, ya que permite conservar los paquetes y últimas mejoras, esto con el fin de evitar errores o mensajes de error al momento de instalar cualquier otra herramienta dentro de Linux.
- Se recomienda verificar la conexión SSH entre la máquina central o controlador y el nodo *host* antes de ejecutar el *playbook,* ya que de lo contrario no se obtendrá las llaves, ni tampoco la autentificación de las mismas, dando un resultado fallido.

- Para efectuar el despliegue correcto del *firewall,* se recomienda seguir los pasos de forma ordenada en cuanto a las tareas que se encuentran dentro del *playbook,* así como también verificar que cada línea se encuentre ubicada correctamente, caso contrario se evidenciara errores al tratar de correr el *playbook.*

- Al instalar la herramienta Ansible, se recomienda verificar tanto su versión, conjuntamente con el módulo de Python, ya que necesita de dicho módulo para funcionar correctamente.

- Antes de ejecutar el *playbook* se debe verificar que el estado de UFW en el nodo *host* se encuentre inactivo*,* de esta manera se determina en primera instancia que no se tenía configurado un *firewall* ni reglas de seguridad; y al ejecutar el *playbook* corroborar el despliegue exitoso del *firewall* y las reglas configuradas dentro del mismo.

# 6 REFERENCIAS BIBLIOGRÁFICAS


[1] M. Tibor, SoftwareLab.org, 2014. [En línea]. Available: https://softwarelab.org/es/que-es-un-firewall/. [Último acceso: 27 Diciembre 2021].

[2] P. Kochavara, F5 GLOSSARY, 13 Julio 2021. [En línea]. Available: https://www.f5.com/es_es/services/resources/glossary/what-is-infrastructure-as-code. [Último acceso: 12 Noviembre 2021].

[3] R. Ahumada, Netmind, 15 Mayo 2019. [En línea]. Available: https://netmind.net/es/infraestructura-iac-y-ansible/. [Último acceso: 11 Noviembre 2021].

[4] atSistemas, [En línea]. Available: https://www.atsistemas.com/es/blog/seguridad-iac-cac. [Último acceso: 12 Noviembre 2021].

[5] F. Flores, OpenWebinars, 31 Mayo 2021. [En línea]. Available: https://openwebinars.net/blog/infraestructura-como-codigo-que-es-y-herramientas/. [Último acceso: 23 Noviembre 2021].

[6] E. Novoseltseva, Apiumhub, 20 Octubre 2020. [En línea]. Available: https://apiumhub.com/es/tech-blog-barcelona/beneficios-y-casos-de-uso-de-terraform/. [Último acceso: 23 Noviembre 2021].

[7] E. Novoseltseva, Apiumhub, 3 Noviembre 2020. [En línea]. Available: https://apiumhub.com/es/tech-blog-barcelona/infraestructura-como-codigo-beneficios-y-herramientas/. [Último acceso: 23 Noviembre 2021].

[8] R. d. Juana, Ipnet, 3 Diciembre 2019. [En línea]. Available: https://www.muycomputerpro.com/2019/12/03/cinco-herramientas-para-automatizar-tu-infraestructura-it. [Último acceso: 23 Noviembre 2021].

[9] V. Carceler, Institut Puig Castellar, [En línea]. Available: https://elpuig.xeill.net/Members/vcarceler/articulos/ansible. [Último acceso: 13 Noviembre 2021].

[10] M. Peréz, OpenWebinars, 20 Junio 2017. [En línea]. Available: https://openwebinars.net/blog/que-es-ansible/. [Último acceso: 13 Noviembre 2021].

[11] E. Heidi, DigitalOcean Community, 11 Junio 2020. [En línea]. Available: https://www.digitalocean.com/community/conceptual_articles/an-introduction-to-configuration-management-with-ansible-es. [Último acceso: 18 Noviembre 2021].

[12] K. Gauza, it-swarm-es.com, 16 Enero 2019. [En línea]. Available: https://www.it-swarm-es.com/es/python/instalacion-del-paquete-ansible-python-en-windows/806388595/. [Último acceso: 26 Noviembre 2021].

[13] A. Sánchez, «Informatica Paralela,» 4 Junio 2020. [En línea]. Available: https://finanpers.com.mx/2020/06/04/ansible-para-principiantes/. [Último acceso: 18 Noviembre 2021].

[14] R. Marti, A Cloud Guru A Pluralsight Company, [En línea]. Available: https://acloudguru.com/course/introduction-to-ansible. [Último acceso: 30 Noviembre 2021].

[15] Vadavo.com, [En línea]. Available: https://www.vadavo.com/blog/que-es-nginx-como-funciona/#%C2%BFQue_es_Nginx_y_Como_Funciona. [Último acceso: 27 Noviembre 2021].

[16] Sentrio, Sentrio, 16 Septiembre 2021. [En línea]. Available: https://sentrio.io/blog/que-es-jenkins/. [Último acceso: 27 Noviembre 2021].

[17] M. León, «arsys,» 21 Diciembre 2020. [En línea]. Available: https://www.arsys.es/blog/ansible/#Conclusion. [Último acceso: 18 Noviembre 2021].

[18] J. Tegnér, Jite.eu, [En línea]. Available: https://jite.eu/2019/2/25/ansible-firewall/. [Último acceso: 20 Noviembre 2021].

[19] P. Pedamkar, EDUCBA, [En línea]. Available: https://www.educba.com/ansible-firewalld/. [Último acceso: 30 Noviembre 2021].

[20] S. Sistemas, solvetic.com, 16 Noviembre 2018. [En línea]. Available: https://www.solvetic.com/tutoriales/article/6414-como-instalar-y-configurar-firewalld-en-centos-y-ubuntu/. [Último acceso: 30 Noviembre 2021].

[21] Linux-Console.net, [En línea]. Available: https://es.linux-console.net/?p=2306. [Último acceso: 30 Noviembre 2021].

[22] S. Rees, D. Mark y E. HeidI, «DigitalOcean Community,» 7 November 2019. [En línea]. Available: https://www.digitalocean.com/community/tutorials/como-instalar-y-configurar-ansible-en-ubuntu-18-04-es. [Último acceso: 12 Enero 2022].

[23] B. Jatri, «bidhankhatri.com.np,» 19 Mayo 2021. [En línea]. Available: https://bidhankhatri.com.np/automation/ansible-playbook-to-deploy-user-with-ssh-key/. [Último acceso: 13 Enero 2022].

# 7 ANEXOS

# ANEXO I: Certificado de Originalidad

**CERTIFICADO DE ORIGINALIDAD**

Quito, D.M. 17 de Febrero de 2022

De mi consideración:

Yo, GABRIELA KATHERINE CEVALLOS SALAZAR, en calidad de Director del Trabajo de Integración Curricular titulado IMPLEMENTACIÓN DE UN FIREWALL MEDIANTE LA HERRAMIENTA ANSIBLE asociado a la IMPLEMENTACIÓN DE UN FIREWALL MEDIANTE LA HERRAMIENTA ANSIBLE elaborado por la estudiante LILIANA ALEXANDRA ACHIG TIPÁN 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 11%.

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

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

https://epnecuador-my.sharepoint.com/:b:/g/personal/gabriela_cevalloss_epn_edu_ec/EepqVlrL8QVEsuy0EZplQ0EBcmhSUJs-ATUe2LORLyXYUQ?e=0gaUe6

Atentamente,

Gabriela Cevallos

Docente

Escuela de Formación de Tecnólogos

## ANEXO II: Video de implementación y pruebas de funcionamiento

**Anexo II.1** Código QR de la implementación y pruebas de funcionamiento