UNIVERSIDAD NACIONAL DE CHIMBORAZO

FACULTAD DE INGENIERIA

CARRERA DE TELECOMUNICACIONES

Desarrollo de un sistema aprovisionador para reducir el tiempo de configuración de un servidor FTP, utilizando herramientas open source.

Trabajo de Titulación para optar al título de Ingeniero en Telecomunicaciones.

Autor:
Armijos Guillen, Darwin Francisco

Tutor:
PhD. Pedro Fernando Escudero Villa

Riobamba, Ecuador. 2025

DECLARATORIA DE AUTORÍA

Yo, **Darwin Francisco Armijos Guillen**, con cédula de ciudadanía **1721377420**, autor del trabajo de investigación titulado**: "Desarrollo de un sistema aprovisionador para reducir el tiempo de configuración de un servidor FTP, utilizando herramientas open source"**, certifico que la producción, ideas, opiniones, criterios, contenidos y conclusiones expuestas son de mí exclusiva responsabilidad.

Asimismo, cedo a la Universidad Nacional de Chimborazo, en forma no exclusiva, los derechos para su uso, comunicación pública, distribución, divulgación y/o reproducción total o parcial, por medio físico o digital; en esta cesión se entiende que el cesionario no podrá obtener beneficios económicos. La posible reclamación de terceros respecto de los derechos de autor de la obra referida será de mi entera responsabilidad; librando a la Universidad Nacional de Chimborazo de posibles obligaciones.

En Riobamba, 15 de enero 2025.

Darwin Francisco Armijos Guillen

C.I:172137742-0

DICTAMEN FAVORABLE DEL PROFESOR TUTOR

Quien suscribe, **Pedro Fernando Escudero Villa** catedrático adscrito a la Facultad de **Ingeniería**, por medio del presente documento certifico haber asesorado y revisado el desarrollo del trabajo de investigación titulado: **"Desarrollo de un sistema aprovisionador para reducir el tiempo de configuración de un servidor FTP, utilizando herramientas open source"**, bajo la autoría de **Darwin Francisco Armijos Guillen**; por lo que se autoriza ejecutar los trámites legales para su sustentación.

Es todo cuanto informar en honor a la verdad; en Riobamba, a los 19 días del mes de noviembre de 2024

PhD. Pedro Fernando Escudero Villa
C.I: 0603612524
TUTOR

CERTIFICADO DE LOS MIEMBROS DEL TRIBUNAL

Quienes suscribimos, catedráticos designados Miembros del Tribunal **de Grado** para la evaluación del trabajo de investigación **"DESARROLLO DE UN SISTEMA APROVISIONADOR PARA REDUCIR EL TIEMPO DE CONFURACIÓN DE UN SERVIDOR FTP, UTILIZANDO HERRAMIENTAS OPEN SOURCE",** presentado por **Darwin Francisco Armijos Guillen**, con cédula de identidad número **172137742-0**, bajo la tutoría de **PhD. Pedro Fernando Escudero Villa**; certificamos que recomendamos la **APROBACIÓN** de este con fines de titulación. Previamente se ha evaluado el trabajo de investigación y escuchada la sustentación por parte de su autor; no teniendo más nada que observar.

De conformidad a la normativa aplicable firmamos, en Riobamba 06 de febrero 2025.

PhD. Carlos Peñafiel Ojeda
PRESIDENTE DEL TRIBUNAL DE GRADO

Mgs. Klever Torres Rodríguez
MIEMBRO DEL TRIBUNAL DE GRADO

PhD. Daniel Santillán Haro
MIEMBRO DEL TRIBUNAL DE GRADO

C E R T I F I C A C I Ó N

Que, **ARMIJOS GUILLEN DARWIN FRANCISCO** con CC: **1721377420**, estudiante de la Carrera **TELECOMUNICACIONES**, Facultad de **INGENIERÍA**; ha trabajado bajo mi tutoría el trabajo de investigación titulado " **DESARROLLO DE UN SISTEMA APROVISIONADOR PARA REDUCIR EL TIEMPO DE CONFIGURACIÓN DE UN SERVIDOR FTP, UTILIZANDO HERRAMIENTAS OPEN SOURCE**", cumple con el 5 %, de acuerdo al reporte del sistema Anti plagio **Compilatio Magister+**, porcentaje aceptado de acuerdo a la reglamentación institucional, por consiguiente autorizo continuar con el proceso.

Riobamba, 20 de Enero de 2025

Firmado electrónicamente por:
PEDRO FERNANDO ESCUDERO VILLA

PhD. Pedro Fernando Escudero Villa
TUTOR(A)

DEDICATORIA

Dedico el trabajo de titulación a mis padres, Martha y Víctor, quienes son mi inspiración y apoyo a lo largo de este recorrido académico. Gracias por creer en mí, por brindarme la oportunidad de perseguir mis sueños, y por nunca dejar de alentarme a alcanzar nuevas metas. Su amor incondicional y sus sabios consejos han sido un pilar fundamental en mi crecimiento personal y profesional.

A mis maestros, aquellos hombres y mujeres que con dedicación y pasión compartieron conmigo sus conocimientos y experiencias en el campo de las telecomunicaciones. Gracias por despertar en mí la curiosidad y el deseo de explorar los límites de la tecnología. Sus enseñanzas no solo han enriquecido mi formación académica, sino que también han moldeado mi visión del mundo y mi compromiso con la innovación.

A mis colegas y amigos, quienes han recorrido conmigo este camino lleno de desafíos y alegrías. Gracias por brindarme su amistad, por ser un constante apoyo en los momentos más difíciles, y por celebrar conmigo cada uno de los logros alcanzados. Sus perspectivas únicas y su espíritu colaborativo han sido invaluables en la realización de este proyecto.

AGRADECIMIENTO

Quiero expresar mi más sincero agradecimiento a mi director de tesis, el PhD Pedro Escudero. Gracias por creer en este proyecto desde el principio, por guiarme con paciencia y sabiduría a lo largo de este proceso, y por empujarme constantemente a ir más allá de mis propios límites. Sus consejos que han sido fundamentales en el desarrollo de esta investigación.

De igual manera, extiendo mi gratitud a todo el equipo de docentes de la Carrera de Ingeniería en Telecomunicaciones de la Universidad Nacional de Chimborazo. Gracias por brindarme las herramientas y el soporte necesario para desarrollar este trabajo, por fomentar un ambiente de aprendizaje y por inspirarme a continuar explorando las fronteras de las telecomunicaciones.

Un agradecimiento especial a mis compañeros de estudio, quienes han sido un pilar fundamental durante esta jornada académica. Gracias por las largas jornadas de estudio compartidas y el compañerismo vivido durante estos años de preparación académica.
Finalmente, quiero expresar mi profunda gratitud a mi familia. A mis padres, Martha y Víctor, por brindarme su amor incondicional, por creer en mí, a mis hermanos por su constante motivación y por celebrar conmigo cada uno de los logros alcanzados.

ÍNDICE GENERAL

DECLARATORIA DE AUTORÍA
DICTAMEN FAVORABLE DEL PROFESOR TUTOR
CERTIFICADO DE LOS MIEMBROS DEL TRIBUNAL
CERTIFICADO ANTIPLAGIO
DEDICATORIA
AGRADECIMIENTO
ÍNDICE DE TABLAS
ÍNDICE DE FIGURAS
ABSTRACT

CAPÍTULO I 15
INTRODUCCIÓN 15
1.1 Antecedentes 15
1.2 Problema 17
1.3 Justificación 18
1.4 Objetivos 19
1.4.1 General 19
1.4.2 Específicos 19
CAPÍTULO II 20
MARCO TEÓRICO 20
2.1 Estado del arte 20
2.2 Marco teórico 21
2.2.1 Aprovisionador 21
2.2.2 Hipervisor VirtualBox 21
2.2.3 Alma Linux 21
2.2.4 Protocolo de trasferencia de archivos 22
2.2.5 Vftpd 22
2.2.6 Vagrant Hashicorp 22
2.2.7 Subsistema de Windows para Linux 22
2.2.8 Herramientas para el despliegue automatizado de servidores 23
2.2.9 Comparativa de software de automatización 28
2.2.10 Aprovisionamiento de máquinas virtuales con Vagrant 30
CAPÍTULO III 31
METODOLOGIA 31
3.1 Tipo de investigación 31
3.2 Técnica documental 31

3.3 Técnica de recolección de datos 31
3.4 Población de estudio y tamaño de la muestra 31
3.4.1 Operacionalización de variables 32
3.5 Hipótesis 32
3.6 Método de análisis y procesamiento de datos 32
3.7 Proceso de la metodología 33
3.8 Requisitos de configuración de un servidor de transferencia de archivos 34
3.8.1 Máquina virtual 34
3.8.2 Configuración del Servidor 35
3.8.3 Configuración del firewall 36
3.8.4 Habilitación de servicios 36
3.8.5 Creación de usuarios 36
3.8.6 Configuración de la máquina virtual 37
3.9 Aprovisionamiento Manual 37
3.10 Aprovisionamiento con Shell Script 39
3.11 Aprovisionamiento con Chef 43
3.12 Aprovisionamiento con Ansible 46
CAPÍTULO IV 53
RESULTADOS Y DISCUSIÓN 53
4.1 Resultados 53
4.1.1 Análisis descriptivo 53
4.1.2 Prueba de hipótesis 63
4.2 Discusión 66
CAPÍTULO V 67
CONCLUSIONES Y RECOMENDACIONES 67
5.1 Conclusiones 67
5.2 Recomendaciones 68
BIBLIOGRÁFIA 69
ANEXOS 72
Anexo1. Códigos extras 72
Anexo2. Datos recopilados 74
Anexo3. Tiempo de ejecución en la configuración 78

ÍNDICE DE TABLAS

Tabla 1. Herramientas de aprovisionamiento, Shell-Script, Ansible, Chef. 28
Tabla 2. Herramientas de aprovisionamiento, Puppet, SaltStack, Pulumi. 29
Tabla 3. Variable dependiente. 32
Tabla 4. Variables independientes. 32
Tabla 5. Requisitos de configuración básica de la máquina virtual. 34
Tabla 6. Paquete de instalación del servidor. 35
Tabla 7. Permisos requeridos en la instalación del servidor. 35
Tabla 8. Comandos usados en la configuración del firewall. 36
Tabla 9. Códigos de habilitación de los servicios configurados. 36
Tabla 10. Códigos usados en la creación de usuarios. 36
Tabla 11. Resumen de procesamiento de casos. 53
Tabla 12. Descriptivo de los estados de configuración. 54
Tabla 13. Resumen de procesamiento de casos. 56
Tabla 14. Descriptivo de los métodos de configuración. 57
Tabla 15. Anova. 60
Tabla 16. Prueba post hoc. 61
Tabla 17. Subconjuntos homogéneos. 61
Tabla 18. Pruebas de normalidad. 63
Tabla 19. Resumen de hipótesis. 64
Tabla 20. Test no paramétrico de Kruskal-Wallis. 64
Tabla 21. Comandos usados en la instalación del subsistema de Linux para Windows. 72
Tabla 22. Comandos de verificación de paquetes en Ubuntu. 72
Tabla 23. Comandos empleados en la instalación de Chef. 72
Tabla 24. Comandos de instalación de Vagrant y Ansible. 73
Tabla 25. Comandos relacionados con el archivo Bashrc. 73
Tabla 26. Registro del tiempo de ejecución del aprovisionador Shell-Script. 74
Tabla 27. Registro del tiempo de ejecución del aprovisionador Chef. 75
Tabla 28. Registro del tiempo de ejecución del aprovisionador Ansible. 76
Tabla 30. Registro del tiempo en la configuración Manual. 77

ÍNDICE DE FIGURAS

Figura 1. Protocolo de transferencia de archivos. 22
Figura 2. Arquitectura de Ansible. 24
Figura 3. Arquitectura de Chef. 25
Figura 4. Arquitectura de Puppet. 26
Figura 5. Arquitectura de Salt Stack. 27
Figura 6. Arquitectura de Pulumi. 28
Figura 7. Uso de herramientas de automatización. 29
Figura 8. Aprovisionamiento de máquinas virtuales con Vagrant. 30
Figura 9. Fases de trabajo. 33
Figura 10. Instalación del paquete vsftpd. 37
Figura 11. Lista de archivos del directorio vsftpd. 38
Figura 12. Creación de usuario y contraseña. 38
Figura 13. Configuración del firewall. 38
Figura 14. Prueba de transferencia de archivos cliente-servidor. 39
Figura 15. Lista de archivos y directorios del aprovisionador Shell Script. 39
Figura 16. Lista de archivos del directorio vsftpd. 40
Figura 17. Verificación del estado del servidor de transferencia de archivos. 41
Figura 18. Verificación del estado del firewall. 41
Figura 19. Transferencia de archivos cliente-servidor. 42
Figura 20. Lista de archivos y directorios del aprovisionador Chef. 43
Figura 21. Lista de archivos de configuración en la distribución Linux. 43
Figura 22. Lista de archivos del directorio vsftpd. 45
Figura 23. Transferencia de archivos cliente-servidor. 46
Figura 24. Lista de archivos y directorios del aprovisionador Ansible. 46
Figura 25. Transferencia de archivos cliente-servidor. 50
Figura 26. Tipos de Aprovisionamiento. 51
Figura 27. Distribución del tiempo de la configuración (Antes-Manual). 54
Figura 28. Distribución del tiempo de la configuración (Después). 55
Figura 29. Diagrama de cajas (Estado). 56
Figura 30. Distribución del tiempo de la configuración (Shell-Script). 57
Figura 31. Distribución del tiempo de la configuración (Chef). 58
Figura 32. Distribución del tiempo de la configuración (Ansible). 59

Figura 33. Diagrama de cajas (Método). 60
Figura 34. Gráfico de medias. 62
Figura 35. Muestras independientes. 65
Figura 36. Tiempos empleados en las configuraciones. 78
Figura 37. Servidores virtuales configurados. 78

RESUMEN

En el presente trabajo de investigación, se abordó la problemática relacionada con el aprovisionamiento de un servidor de transferencia de archivos. El aprovisionamiento se automatizó con la finalidad de reducir el tiempo de configuración. Por otro lado, el sistema fue desarrollado empleando herramientas de código abierto como Vagrant, Ansible, Chef y Shell Script. Además, el servidor fue desplegado en una máquina virtual y alojado en el hipervisor VirtualBox. El sistema operativo utilizado en el servidor fue AlmaLinux. La herramienta Vagrant se utilizó para gestionar el despliegue de la máquina virtual. En conjunto con las herramientas de aprovisionamiento Shell Scripts, Chef y Ansible, el despliegue de la máquina virtual incluye la configuración de la memoria RAM del servidor, la configuración de red, el número de núcleos y la imagen del sistema operativo. Además, mediante el aprovisionamiento con las herramientas utilizadas, se logró configurar el servidor FTP, establecer las reglas de firewall, habilitar los puertos TCP 20 y 21, asignar los permisos necesarios al servidor y crear los usuarios correspondientes. El acceso y uso del servidor se realiza a través de la línea de comandos. Cabe destacar que las tres herramientas probadas presentan diferentes tiempos de aprovisionamiento, los cuales varían entre 2.11 y 6.11 minutos. El tiempo de configuración se redujo significativamente, pasando de 34.56 minutos en una configuración manual a 2.11, 2.34 y 6.11 minutos, respectivamente, al utilizar cada herramienta. Esto representa una disminución del 93.89 % con Ansible, 93.22 % con Shell Script y 82.32 % con Chef. Además, los diferentes métodos de aprovisionamiento emplearon varios tipos de archivos de configuración: Shell Script utiliza archivos con extensión (.sh), Ansible trabaja con archivos (.yml), y Chef emplea archivos (.rb) y (.erb).

Palabras clave: Automatización de Aprovisionamiento, Virtualización, Herramientas de código abierto, Configuración de servidores FTP.

ABSTRACT

This research focused on addressing the challenges of providing a file transfer server. The provisioning process was automated to minimize configuration time. The system was developed using open-source tools such as Vagrant, Ansible, Chef, and Shell Script. The server was deployed on a virtual machine hosted on the VirtualBox hypervisor, and the operating system used for the server was AlmaLinux. Vagrant was utilized to manage the deployment of the virtual machine. During the deployment, Shell Scripts, Chef, and Ansible were employed to configure the server's RAM, network settings, CPU cores, and the operating system image. The FTP server was set up through this provisioning process, firewall rules were established, and TCP ports 20 and 21 were enabled. Necessary server permissions were assigned, and user accounts were created. Access to the server is conducted through the command line. The three tested tools demonstrated varying provisioning times, ranging from 2.11 to 6.11 minutes. The configuration time was significantly reduced, dropping from 34.56 minutes in a manual setup to 2.11, 2.34, and 6.11 minutes, respectively, with each tool. This reflects a reduction of 93.89% with Ansible, 93.22% with Shell Script, and 82.32% with Chef. Additionally, each provisioning method utilized different types of configuration files: Shell Script used files with a (.sh) extension, Ansible used (.yml) files, and Chef employed (.rb) and (.erb) files.

Keywords: Provisioning Automation, Virtualization, Open-Source Tools, FTP Server Configuration.

Reviewed by:

Lic. Raquel Verónica Abarca Sánchez. Msc.

ENGLISH PROFESSOR

c.c. 0606183804

CAPÍTULO I

INTRODUCCIÓN

En esta sección se presentan ideas relacionadas en la configuración y gestión de servidores, la utilidad de las herramientas existentes, y los resultados que se han obtenido al ser usadas.

1.1 Antecedentes

En el mundo actual, donde la tecnología se desarrolla a un ritmo acelerado, la eficiencia se ha convertido en pilares fundamentales en la sociedad. En el campo de las telecomunicaciones, el proceso de configuración de servidores puede resultar una tarea amplia y prolongada; por tanto, surge la necesidad de implementar soluciones capaces de reducir el tiempo de configuración mejorando la productividad. Sin embargo, en el ámbito tecnológico relacionado con los servidores, se está presenciando un acelerado avance en las formas de gestionar sus datos, donde las demandas de los recursos por parte de los usuarios exigen soluciones diversas como mejorar las políticas, aumento de recursos, entre otros [1].

El análisis realizado a las herramientas que despliegan infraestructura de código como Ansible, Puppet, Chef, Terraform y SaltStack, para fines de configuración de servicios en la nube en el 2023, según la tasa de uso fue, del 30 % para Terraform, 20 % Ansible,14 % Chef, 15 % Puppet,10 % SaltStack. Además, para identificar las herramientas más utilizadas, se realizaron búsquedas en la biblioteca digital, Clarivate Web of Science y Google Scholar, de estudios publicados en los últimos dos años utilizando las palabras clave Infrastructure as Code IaC, IaC Management e IaC Tools [2].

Por otra parte, en las filiales, Amsterdam en Europa, Bangalore en Asia y San Francisco en América del Norte, de la empresa Digital Ocean. Han instalado un sistema de registro automatizado para el análisis de ataques cibernéticos de una red con el fin de verificar la efectividad de los roles creados con Ansible. El análisis se enfocó en los servicios que se proporcionan de un servidor Honeypot. Todos los ataques registrados dieron evidencia de varias solicitudes realizadas a servicios inactivos, incluidos HTTP, HTTPS, SMTP y POP3, siendo inalcanzables. Esto se logró con redirección de puertos usando los códigos creados en Ansible [3].

Cabe considerar que la creciente demanda y requerimientos en los proyectos de software, los equipos necesitan automatizar sus tareas en la configuración de infraestructura, aunque este concepto es nuevo existen escasez de expertos para satisfacer esta demanda, por ende una investigación desarrollada sobre el uso de estas herramientas con la comparación de pares y literatura gris tomando como referencias a, Terraform con el 73.7 %, AWS 78.9 %, Azure 31.6 %, Ansible 26.3 %, Pulumi 21.1 %, Chef 5.3 %, Puppet 5.3 % y Salt Stack 5.3 %, obteniendo como resultados a AWS, Terraform y Azure las más utilizadas en configuraciones de servicios, aunque las herramientas de automatización no son difícil de aprender a usarlas, se requieren de inversión y tiempo [4].

Un aspecto significativo señalado en la literatura es el crecimiento de herramientas de software especializadas que han surgido como consecuencia de la adopción de infraestructura de código en entornos de servicios en la nube. Destaca particularmente Ansible, Chef y Puppet como ejemplos de estas soluciones, estas herramientas han demostrado ser fundamentales para facilitar la gestión de arquitecturas complejas, incluyendo clúster de servidores, contenedores, sistemas de almacenamiento y redes distribuidas de gran escala [5].

La comparación de Argon y Ansible en relación con su efectividad, eficiencia, facilidad de usos en el despliegue de infraestructura, involucró a 67 estudiantes, dónde se desarrollaron tres experimentos, que consistían en la evaluación de cada herramienta, por separado y una en conjunto, Argon obtuvo el 75 % de la eficiencia, mientras Ansible el 50 %. En la efectividad, el 75 % de ambas herramientas, en cuanto a la facilidad de uso, Argon resultó ser más fácil de usar en el despliegue de configuraciones [6].

Por consiguiente, se enfrentan numerosas dificultades al momento de implementar servicios y gestionar servidores debido a la creciente evolución de los sistemas operativos. En este contexto, la automatización en configuración de servicios es esencial para mejorar la respuesta de los administradores a los requerimientos de los usuarios. Además, existen herramientas de configuración y gestión como Chef, Puppet, Ansible, entre otras. Estas herramientas permiten configurar y administrar de manera centralizada los dispositivos de red. Particularmente, Ansible se distingue como una herramienta de orquestación y gestión de configuración de código abierto, capaz de automatizar la configuración de diversos dispositivos de telecomunicaciones [7].

Para dar una solución a la problemática en la configuración de servidores de trasferencia de archivos, el presente trabajo de titulación aborda la necesidad de la implementación de un sistema de aprovisionamiento. Por otra parte, de todas las herramientas listadas, en este trabajo se comparará 3, considerando las características como conjunto único en sus arquitecturas y el tipo de lenguaje que emplean. Este trabajo investigativo se centra en automatizar las tareas de configuración de un servidor, aprovechando las capacidades de automatización de las herramientas Shell, Ansible y Chef, buscando optimizar el proceso de configuración para minimizar la intervención manual. Esta propuesta está enfocada al ahorro de tiempo de las pequeñas empresas que requieran de estos servicios en la configuración de servidores de transferencia de archivos alojados en máquinas virtuales [8].

1.2 Problema

La creciente necesidad de transferencia de archivos entre ordenadores ha llevado al uso general del protocolo FTP (File Transfer Protocol), siendo una herramienta que permite a los usuarios enviar y recibir archivos de forma rápida y sencilla sin necesidad de usar medios físicos como un CD, un DVD o una memoria USB. Sin embargo, existe una alta demanda de estos servicios para diferentes propósitos, ya sea en empresas medianas o pequeñas; muchas de estas carecen de recursos o conocimientos técnicos en sistemas de transferencia de archivos, y estas carencias ocasionan ineficiencias operativas [9].

Además, la configuración manual de servidores requiere conocimientos técnicos y puede llevar varias horas o incluso días, dependiendo de la complejidad del entorno, por ende, estas configuraciones están quedando obsoletas. Por otra parte, estudios recientes han demostrado que aproximadamente el 60% de las brechas de seguridad en sistemas FTP se deben a errores de configuración. Estos factores subrayan la necesidad de un enfoque menos propenso a errores en la configuración de servidores FTP [10].

Entre los factores que influyen se encuentran la diversidad de sistemas operativos y versiones de software FTP, la falta de herramientas de automatización específicas para la configuración de servidores y recurso humano limitado. Además, la creciente adopción de arquitecturas ha aumentado la necesidad de desplegar y configurar nuevos servidores, sin dejar a un lado los errores de sintaxis que se pueden generar en los archivos de configuración, configuración incorrecta de los permisos, problemas de firewall; por consiguiente, estos procesos aumentan la carga laboral y la operatividad en los administradores de los servicios en telecomunicaciones y afectan una institución o empresa [11].

El aprovisionamiento permite preparar a un servidor o varios para su uso, donde incluye la instalación del sistema operativo, la configuración de red, la instalación de paquetes, entre otros. Además, el proceso de instalación se reduce, liberando recursos humanos para otras tareas específicas de administración de servicios en telecomunicaciones. Por otra parte, a medida que la infraestructura de los servidores crece, la gestión manual de la configuración requiere mayor tiempo y conocimiento, lo que dificulta la escalabilidad y el mantenimiento del sistema [12].

1.3 Justificación

La necesidad de abordar este problema se sitúa en la creciente demanda de sistemas de transferencia de archivos en diversos sectores, como el financiero, el sanitario y el gubernamental, entre otros. Según un estudio realizado por Red Hat, indica que antes de Ansible, las implementaciones de aplicaciones y el aprovisionamiento de servidores tomaban un promedio de 4 a 6 horas por instancia. Con la automatización de Ansible, estos procesos se simplificaron a un promedio de 30 a 45 minutos, lo que representa una reducción de más del 80 %. Además, la automatización de tareas y la reducción de errores generaron un ahorro del 30 % en costos laborales [13].

La viabilidad se sustenta mediante recursos como, Vagrant, que ha demostrado ser una solución en la gestión de entornos virtualizados. Ansible y Chef, son plataformas de automatización de infraestructura reconocidas, Shell-Script es una herramienta propia de los sistemas Linux. La combinación de estas herramientas con scripts personalizados ofrece un marco sólido para el desarrollo de un sistema aprovisionador adaptable a los requerimientos de los servidores FTP. Sin embargo, es importante señalar que la mayoría de las investigaciones se han centrado en servidores web o de bases de datos, dejando un vacío en lo que respecta a los servidores FTP [14].

La implementación de un aprovisionador incluye la instalación y configuración del software necesario, la asignación de permisos de acceso, configuración de la gestión de los archivos y directorios compartidos, entre otras; además, la configuración automatizada reduce los errores de configuraciones que podrían surgir si son realizadas por las personas; por otra parte, se pueden reasignar recursos humanos a proyectos estratégicos [15].

La presente investigación propone la implementación de un sistema aprovisionador que reduzca el tiempo de configuración; además, este servidor está alojado en una distribución Linux en ambientes de máquinas virtuales, basado en herramientas como Vagrant, Ansible, Chef y Shell-Script. La transferencia de archivos se realiza de manera remota en redes LAN; por otra parte, es necesario comprender el funcionamiento del protocolo de transferencia de archivos para determinar los requerimientos necesarios.

Este proyecto puede aportar como un caso de estudio en la aplicación de técnicas de aprovisionamiento automatizado en otros tipos de servidores y servicios en ambiente virtual. Se espera contribuir al conocimiento teórico en la automatización de infraestructuras de redes como la gestión de sistemas de transferencia de archivos. Entre los sectores favorecidos de este proyecto estarían los administradores de sistemas operativos y el equipo de soporte de redes. Las instituciones que dependen de la transferencia de archivos y los usuarios finales.

1.4 Objetivos

1.4.1 General

- Desarrollar un sistema aprovisionador para disminuir el tiempo de configuración de un servidor FTP, basado en herramientas open source.

1.4.2 Específicos

- Realizar un estudio comparativo de diferentes herramientas open source para automatizar la instalación de servidores FTP.
- Diseñar el flujo de trabajo de aprovisionamiento adaptado a los requisitos de configuración de un servidor FTP.
- Evaluar las ganancias de rendimiento logradas al implementar el sistema de aprovisionamiento en comparación con configuraciones manuales.

CAPÍTULO II

MARCO TEÓRICO

En esta sección se encuentra el análisis literario de artículos publicados, libros, textos académicos relacionados con el tema de investigación basado en los últimos 5 años.

2.1 Estado del arte

Pierre Talani analizó las configuraciones de 8 routers en ambientes virtuales (GNS3) utilizando Ansible y Netmiko, evaluó el rendimiento y la escalabilidad, se determinó que Ansible ayuda en entornos empresariales y en las configuraciones de varios dispositivos, a diferencia de Netmiko que tiene dificultades cuando aumenta el número de dispositivos, el rendimiento fue evaluado en la trasferencia de bits de los dispositivos que se configuraron con el uso de los scripts, y la escalabilidad en la capacidad de configuración de los dispositivos, Netmiko demoro 45 segundos y Ansible 35 segundos en configurar los dispositivos, esto representa que Ansible es aproximadamente un 50% más rápido que Netmiko al configurar 8 enrutadores, mientras que Netmiko es un 33% más rápido al configurar 1 enrutador [15].

Gleb Gurbatov evaluó dos herramientas, Ansible y Terraform, en la capacidad de uso de memoria y tiempo de configuración de máquinas virtuales en OpenStack, donde obtuvo resultados para la primera herramienta, el 20.91 % de uso de recursos del computador y se demoró 48.3 segundos, para la segunda se obtuvieron valores de 5.22 % de uso de recursos del computador y toda la configuración la realizó en 60.1 segundos. En general, los datos demuestran que Ansible utiliza más recursos de memoria, pero completa las tareas en menor tiempo que Terraform. Sin embargo, Terraform usa menos recursos, pero tarda más tiempo en completar las tareas [16].

Zargham Ahmad implementó una solución utilizando Vagrant, Ansible y Cyber-Sandbox-Creator para aprovisionar servidores en entornos de desarrollo. Crearon un Vagrantfile para definir la máquina virtual y scripts para configurar el servidor web. Lograron crear entornos de desarrollo consistentes y reproducibles, reduciendo el tiempo de configuración en un 70%. La principal ventaja fue la facilidad para replicar entornos idénticos, mientras que la desventaja fue el aprendizaje para dominar ambas herramientas [17].

Liliana Enciso describe que la implementación de redes mediante el uso de archivos yml de Ansible han mejorado los procesos de configuraciones de dispositivos de red; además logró optimizar la gestión y la disponibilidad de los costos de la red debido a la simultaneidad de Ansible, ya que permite configurar varios dispositivos al mismo tiempo. El protocolo implementado en la red fue OSPF (Open Shortest Path First). Por otra parte, la cantidad de dispositivos configurados fueron 5 switches de capa 3, 6 switches de capa 2 y 1 router; todos los dispositivos fueron configurados con todas las tareas programadas al 100 % [18].

Arfan Efendi y su equipo de trabajo detallan que mejoró la infraestructura de red, integrando la tecnología Vxlan con técnicas de automatización mediante Python, Ansible y Git. Se observaron mejoras en la escalabilidad de las redes. Además, el uso de Python en la automatización de las operaciones de red agilizó la configuración y gestión, mientras que el papel de Ansible ayudó en la gestión de las copias de seguridad. Estas fueron realizadas 4 veces al día en los intervalos de 00:00, 06:00, 12:00 y 18:00 horas, dando una efectividad del 100 %. Por otra parte, redujo el tiempo de configuración al 50 % [19].

Niko Kalliomaa comparó tres herramientas, Terraform, Pulumi y Ansible. Se descubrió que Terraform y Pulumi son adecuados para la gestión de la infraestructura con dependencias complejas, mientras que Ansible se adapta mejor a escenarios donde la infraestructura ya existe o es poco compleja. Los parámetros evaluados fueron, características y capacidades técnicas de las herramientas, facilidad de uso y curva de aprendizaje, donde Terraform y Pulumi ofrecen una documentación más completa y guías paso a paso, mientras que Ansible tiene un enfoque más conceptual [20].

2.2 Marco teórico

En este proyecto se abordarán componentes de un sistema aprovisionador, donde se considera el conjunto de software para implementar el sistema y la integración de las herramientas de automatización. Además, se analiza la capacidad de gestión y el tipo de lenguaje que usan para generar las tareas de configuración del servidor de transferencia de archivos en máquinas virtuales.

2.2.1 Aprovisionador

Es una herramienta o software que se utiliza para automatizar la creación y configuración de recursos informáticos, como redes y servidores. Su objetivo es simplificar y agilizar las tareas y configuraciones de los sistemas operativos [21].

2.2.2 Hipervisor VirtualBox

Permite crear y ejecutar máquinas virtuales en un ordenador anfitrión. Las máquinas virtuales son entornos aislados que pueden ejecutar sus propios sistemas operativos y aplicaciones. Esto permite a los usuarios ejecutar múltiples sistemas operativos en un solo ordenador. Puede ser útil para probar software, desarrollar aplicaciones o ejecutar aplicaciones que no son compatibles con el sistema operativo anfitrión [22].

2.2.3 Alma Linux

Es una distribución de Linux empresarial gratuita y de código abierto basada en Red Hat, y es compatible binariamente con RHEL. Esto significa que las aplicaciones y los paquetes que se ejecutan en RHEL también se ejecutarán en Alma Linux [23].

2.2.4 Protocolo de trasferencia de archivos

Es un protocolo de red estándar que permite a los usuarios transferir archivos entre computadoras a través de una red. Se utiliza principalmente para transferir archivos entre un cliente y un servidor, pero también se puede utilizar para transferir archivos entre dos servidores [24].

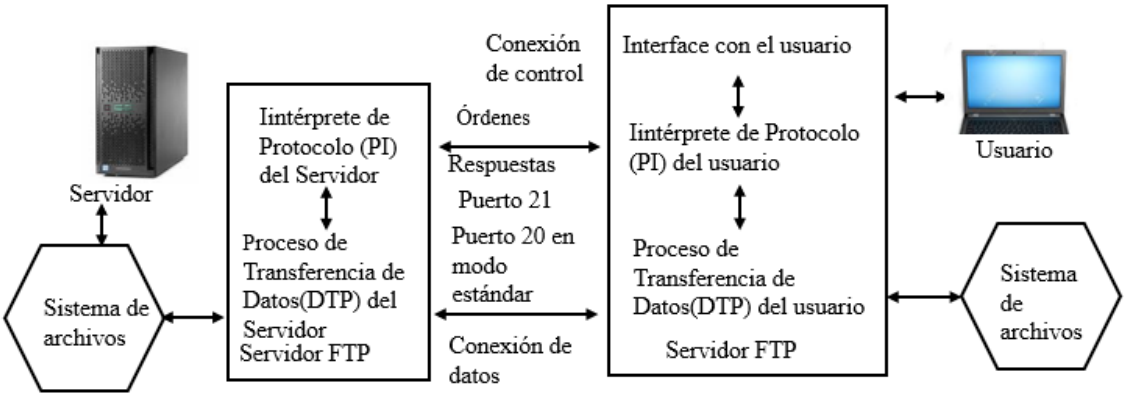

Figura 1. Protocolo de transferencia de archivos.

La Figura 1 incluye la funcionalidad del protocolo FTP, donde la transferencia de archivos comienza cuando el usuario inicia la conexión desde la laptop, comunicándose mediante la interfaz de usuario con el intérprete de protocolo, luego se establece la conexión de control por el puerto 21, este realiza las peticiones desde el protocolo de interfaz de usuario al protocolo de interfaz del servidor, ambos instruyen el proceso de trasferencia de datos a través del puerto 20, este proceso utiliza el protocolo TCP/IP, ya que es orientado a conexión, es decir, tanto el cliente como el servidor realizan la negociación de trasferencia de archivos.

2.2.5 Vftpd

Vsftpd (Very Secure FTP Daemon) es un servidor de transferencia de archivos gratuito para sistemas operativos tipo Unix. Es conocido por su seguridad y estabilidad [25].

2.2.6 Vagrant Hashicorp

Permite crear, configurar y administrar máquinas virtuales de manera automatizada y portable. Utiliza un archivo de configuración llamado vagrantfile, escrito en el lenguaje Ruby, que describe las características de la máquina virtual que se desea crear, como el sistema operativo, los recursos asignados y las aplicaciones que se deben instalar [26].

2.2.7 Subsistema de Windows para Linux

Es una funcionalidad integrada en Windows que posibilita a los usuarios instalar y ejecutar distribuciones de Linux directamente en su sistema operativo Windows de manera nativa. Esta característica proporciona un ambiente completo de comandos de línea compatible con Linux, además permite trabajar en entornos multiplataforma [27].

2.2.8 Herramientas para el despliegue automatizado de servidores

Existen varias herramientas de código abierto que pueden ayudar a automatizar la instalación y configuración de servidores de transferencia de archivo. Algunas de las más utilizadas son:

- Shell Script
- Ansible
- Chef
- Puppet
- SaltStack
- Pulumi

Shell Script

Se refiere a la práctica de escribir programas en lenguajes interpretados por la Shell del sistema operativo. Estos automatizan tareas y combinan comandos, programas y utilidades de Linux. Además, los scripts pueden realizar tareas sin intervención manual, lo que ahorra tiempo y reduce los errores. [28].

```
#/bin/bash
mkdirk my_new-directory
     my_new-directory"
cd my_new-directory
touch my_file.txt
inside the directory
```

Script 1. Ejemplo de creación de directorio.

Según el ejemplo del Script 1, contiene una tarea de automatización donde se crea un directorio llamado my_new_directory, y se cambia a dicho directorio con el comando cd; además, se crea un archivo .txt nombrado my_file.txt. Por otra parte, para poder ejecutar el script es necesario guardar el archivo y otorgar los permisos correspondientes para su ejecución.

Ventajas:

- **Automatización:** Tiene la capacidad de automatizar tareas repetitivas como, realizar copias de seguridad de los archivos, actualizar software o generar informes, y estas tareas las realiza sin intervención de las personas [29].
- **Flexibilidad:** Se pueden escribir para manejar amplias tareas, desde simples manipulaciones de archivos hasta operaciones complejas del sistema [29].
- **Portabilidad:** Se ejecutan en cualquier distribución de Linux con cambios mínimos, esto los hace portátiles entre diferentes sistemas operativos [30].
- **Eficiencia de recursos:** Tienden a ser livianos y consumen recursos mínimos del sistema. Esto es beneficioso para dispositivos antiguos con limitaciones [31].

Ansible

Tiene una sintaxis simple de interpretar, y para aprovisionar servidores se utilizan roles y libros de jugadas, estos últimos son archivos (.yml), mientras que los roles son estructuras organizadas que agrupan y encapsulan tareas. Además, emplea la multiplexación Secure Shell (SSH) para poder comunicarse con la máquina virtual que se desea configurar [32].

```
-hosts: webservers
 tasks:
 -name: instalar apache
   yum:
        name: httpd
         state: lastest
```

Script 2. Ejemplo de la instalación de Apache.

El ejemplo del Script 2 abarca el libro de jugadas y cómo se asignan las tareas de instalación del servidor y el paquete httpd; además, se instala la última versión.

Ventajas:

- **Integración con otras herramientas:** Se integra con herramientas de desarrollo como Vagrant, controlador de versiones, entre otras [33].
- **Reutilización de códigos:** Gracias a los roles se puede llegar a reutilizar los códigos de un libro de jugadas en otro y así evitar reescribir el mismo código [34].
- **No requiere de agentes:** Se conecta a la máquina virtual que se desea aprovisionar mediante conexión SSH, es decir, no requiere de otros programas adicionales para realizar la configuración requerida [35].

Arquitectura de Ansible

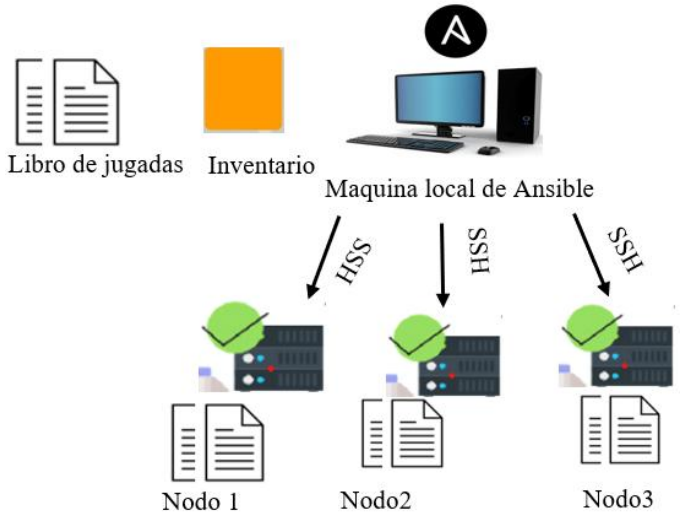

Figura 2. Arquitectura de Ansible.

Conforme a la Figura 2, en el centro se encuentra la máquina local; esta máquina utiliza un libro de jugadas (playbook) que contiene las instrucciones que son ejecutadas, y un

inventario que lista los nodos (servidores). Ansible se conecta a tres nodos utilizando el protocolo SSH y lee el playbook, consulta el inventario, y luego ejecuta las tareas definidas en cada nodo de forma remota. Esto permite una gestión centralizada de múltiples sistemas, automatizando tareas de configuración y despliegue [33].

Chef

Permite gestionar la configuración de máquinas virtuales, su modelo de trabajo es de cliente-servidor; además, sus archivos de configuración son escritos en el lenguaje Ruby; por otra parte, Chef utiliza arquitectura basada en agentes, es decir, necesita tener instalado un cliente Chef en cada dispositivo que se desea aprovisionar y un servidor Chef en la máquina principal [36].

Componentes principales de Chef

- **Servidor Chef:** Actúa como servidor común entre la estación de trabajo y los nodos clientes; aquí se encuentran las recetas (cookbooks), metadatos, y entonces los Chef-clientes realizan peticiones de configuración al Chef-servidor [37].
- **Estación Chef:** Es la estación de trabajo donde se crean los cookbooks o recetas; además, interactúa con el servidor y el cliente, ya que contiene los roles donde se administran los accesos a los nodos [37].
- **Nodos Chef:** Estos pueden ser un servidor virtual o físico que se administra desde el nodo central. Además, estos son llamados clientes. Por otra parte, son encargados de desplegar los cookbooks y usarlos en las configuraciones [38].

Arquitectura de Chef

La arquitectura de Chef es un sistema de gestión de configuración y automatización de infraestructura.

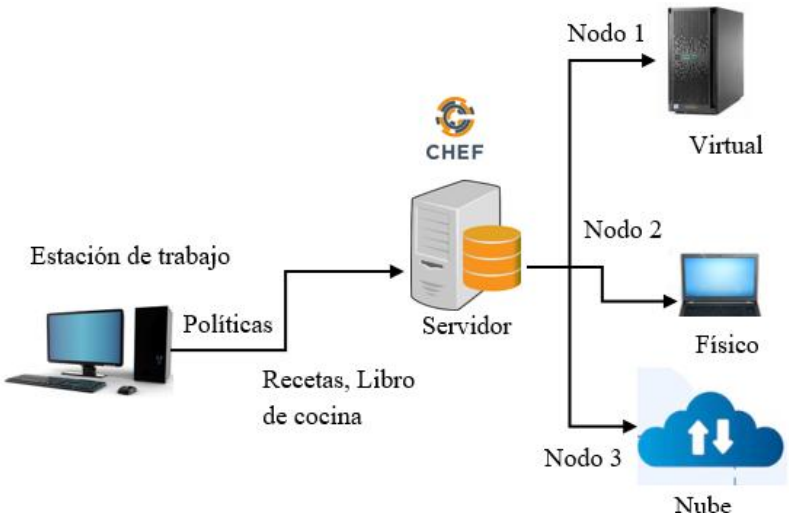

Figura 3. Arquitectura de Chef.

De acuerdo con la Figura 3, la estación de trabajo es el punto de inicio donde se crean y gestionan las políticas y las recetas. Estas se envían al servidor Chef, que actúa como central

de la infraestructura. El servidor Chef almacena las configuraciones y las distribuye a los nodos gestionados. El flujo de trabajo implica la creación de políticas en la estación de trabajo, su transferencia al servidor Chef, y luego la aplicación de estas configuraciones en los diferentes nodos.

Puppet

Permite la gestión y la configuración a través de código; además, agiliza el mantenimiento y la implementación de los servidores. Es utilizado en la automatización de las configuraciones de red. Su lenguaje de programación se basa en ser declarativo, es decir, solo se escriben los resultados sin utilizar una secuencia detallada de instrucciones [37].

Arquitectura de Puppet

Se divide en nodo principal y nodo agente. En el nodo principal se almacenan todos los archivos necesarios para la configuración, mientras que el nodo agente interactúa con el servidor para determinar que las configuraciones se realicen de manera correcta; además, el agente recopila información de los estados de las actualizaciones realizadas [39].

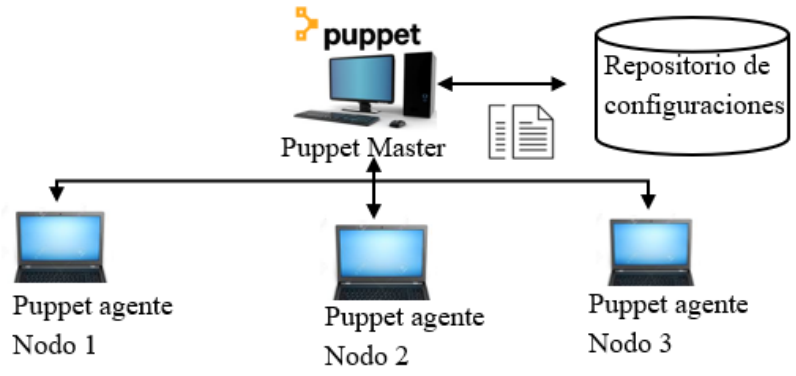

Figura 4. Arquitectura de Puppet.

Según se aprecia en la Figura 4, el Puppet Máster actúa como el servidor de control. Este se conecta a un repositorio de configuraciones, donde se almacenan las definiciones de estado deseado para los sistemas gestionados. El Puppet Máster interactúa con tres nodos. Las flechas bidireccionales indican una comunicación en ambos sentidos entre el Máster y los agentes. Implica que el Máster envía configuraciones a los agentes y estos reportan su estado de regreso.

Salt Stack

Se utiliza para configurar servidores, gestionar sistemas y desplegar aplicaciones de seguridad. Su arquitectura cliente-servidor emplea un modelo de difusión para ejecutar comandos SSH. Además, compite directamente con, Puppet, Chef y Ansible [40].

Arquitectura de Salt Stack

Trabaja en una arquitectura cliente-servidor, con un Salt máster que actúa como servidor central y Salt minions como los clientes. El Máster envía comandos y estados a los Minions, que ejecutan estas instrucciones y estos devuelven los resultados al Máster. La comunicación es bidireccional y se realiza a través de los puertos 4505 y 4506 (TCP) [41].

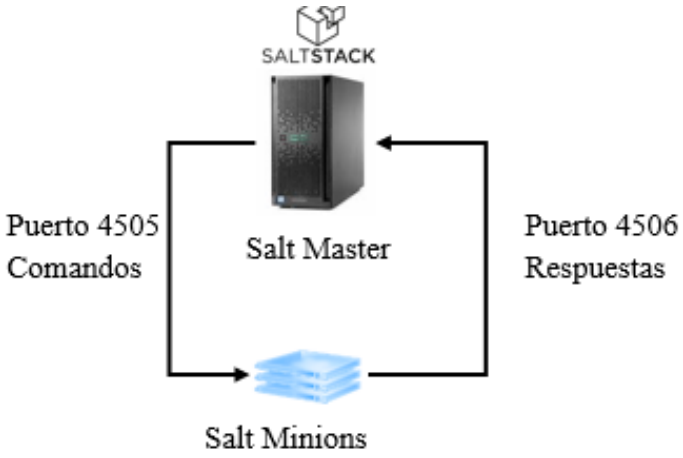

Figura 5. Arquitectura de Salt Stack.

Tal como se observa en la Figura 5, el Salt Máster actúa como el cerebro central del sistema, conectándose con los Salt Minions (agentes) a través de dos puertos, el puerto 4505, dedicado al envío de comandos y configuraciones desde el Máster hacia los Minions, y el puerto 4506, utilizado por los Minions para enviar respuestas y reportes de estado de vuelta al Máster. Esta comunicación bidireccional forma un ciclo completo de control y retroalimentación, donde el Máster puede ordenar cambios de configuración, actualizaciones o ejecutar comandos, mientras los Minions implementan estas instrucciones y reportan sus resultados.

Pulumi

Es compatible con JavaScript, Python, Go, Net, YALM. Pulumi ofrece una funcionalidad para la gestión de infraestructura de código y permite definir y desplegar recursos en la nube, lo que facilita la creación y mantenimiento de infraestructuras complejas. Su motor central orquesta el proceso de despliegue e interactúa con diversos servicios a través de complementos especializados [42].

Arquitectura de Pulumi

Da seguimiento de los recursos en las actualizaciones, proporciona capacidades de previsualización en los cambios. Esto ayuda a revisar y aprobar modificaciones antes de aplicarlas, por otra parte, fomenta la reutilización de código. Además, su integración con otras herramientas permite el soporte y control de versiones. Por otro lado, está diseñada para ser escalable y se adapta a las necesidades de proyectos de cualquier tamaño y complejidad [43].

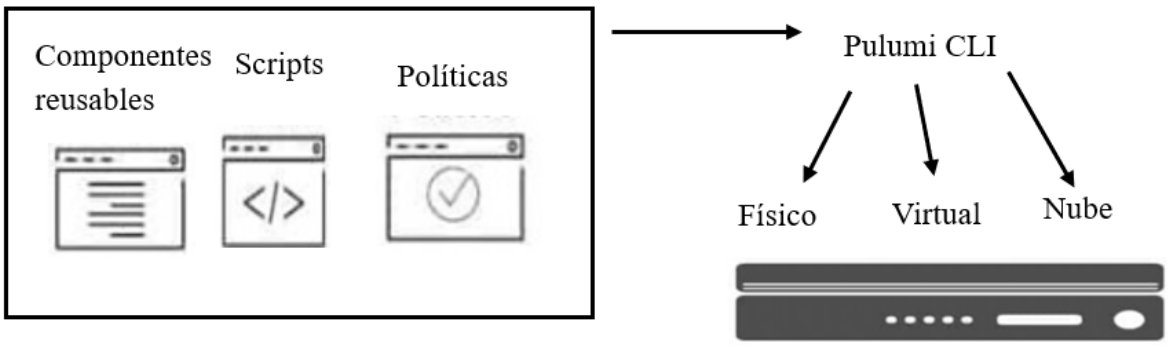

Figura 6. Arquitectura de Pulumi.

De acuerdo con la Figura 6, en el lado izquierdo, dentro del recuadro, se encuentran los tres elementos. Son los componentes que crean bloques de código reutilizables, los scripts contienen las instrucciones y automatizaciones, y las políticas definen las reglas y estándares de configuración. La flecha que conecta con Pulumi CLI representa el proceso de interpretación y ejecución, donde Pulumi toma estos elementos y los procesa. Desde el CLI, se ramifican tres flechas que indican los destinos de despliegue [44].

2.2.9 Comparativa de software de automatización

En este apartado se presenta la comparación de algunos softwares de automatización de servidores que han sido estudiados mediante recopilaciones bibliográficas, y 3 de estos fueron utilizados en el desarrollo del sistema aprovisionador del servidor FTP [45].

Tabla 1. Herramientas de aprovisionamiento, Shell-Script, Ansible, Chef.

Clasificación	**Shell Script**	**Chef**	**Ansible**
Arquitectura	N/A	Cliente-servidor	Solo cliente
Lenguaje	Bash, sh, etc	Ruby	YAML
Escalabilidad	Limitada	Alta	Alta
Administración	Manual	Centralizada	Centralizada
Compatibilidad	Linux, macOS, Windows	Linux, macOS, Windows, nubes públicas	Linux, macOS, Windows, dispositivos de red, nubes públicas
Uso típico	Tareas simples	Aprovisionamiento	Aprovisionamiento
Año	1995	2009	2012

Como se evidencia en Tabla 1**,** las herramientas que fueron utilizadas en el desarrollo del aprovisionador fueron Ansible, Shell Script, Chef; además, se encuentran las diferencias entre ellas en el tipo de lenguaje, la compatibilidad de los sistemas operativos y el uso que tienen.

Tabla 2. Herramientas de aprovisionamiento, Puppet, SaltStack, Pulumi.

Clasificación	**Puppet**	**SaltStack**	**Pulumi**
Arquitectura	Cliente-servidor	Cliente-servidor	Basado en la nube
Lenguaje	Ruby	YAML, Python	Python, TypeScript, Go, NET
Escalabilidad	Alta	Alta	Alta
Administración	Centralizada	Centralizada	Descentralizada
Compatibilidad	Linux, Windows, UNIX.	Linux, Windows, UNIX.	Multi-nube (AWS, Azure, GCP, etc.)
Uso típico	Aprovisionamiento	Aprovisionamiento en la nube	Aprovisionamiento multi-nube
Año	2005	2011	2018

En la Tabla 2 se observa la comparación de las herramientas Puppet, Salt Stack y Pulumi. Cada herramienta tiene características y su evolución según el año de lanzamiento.

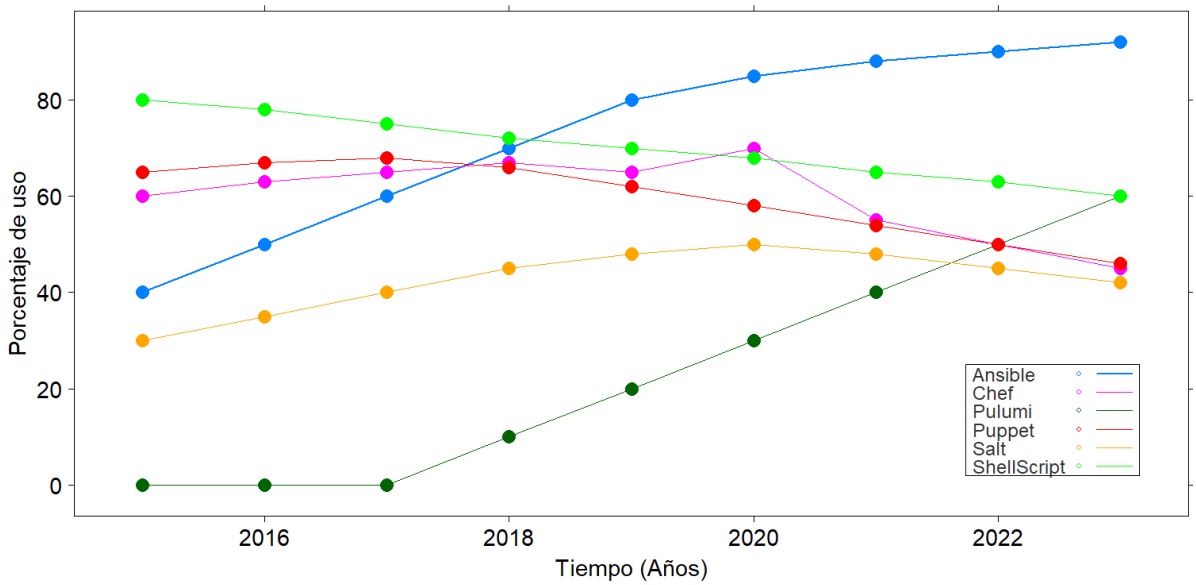

Figura 7. Uso de herramientas de automatización.

Según la Figura 7, el uso de las herramientas para la instalación de servidores entre el periodo 2015-2023, donde el eje Y muestra el porcentaje de uso (de 0 a 100%) y el eje X representa el tiempo en años, se pueden observar seis herramientas representadas por líneas de distintos colores. Ansible (azul) tiene un crecimiento constante desde aproximadamente 40 % hasta más del 80 %, convirtiéndose en la herramienta más utilizada. Salt (naranja) mantiene un crecimiento moderado entre 30 % y 50 %; Shell (verde) comienza con cerca del 80 % y muestra una tendencia descendente hasta aproximadamente 60 %; Puppet (rojo) inicia en 65 % y muestra una disminución hasta 45 %; Chef (violeta) mantiene una tendencia estable entre 60 % y 70 % antes de descender; y Pulumi (verde oscuro) comienza en 0 % y muestra un crecimiento hasta alcanzar aproximadamente el 60 %, siendo la herramienta que parte desde el punto más bajo.

2.2.10 Aprovisionamiento de máquinas virtuales con Vagrant

Permite configurar y preparar una máquina virtual después de su creación. Este proceso ayuda a instalar software, configurar servicios y realizar ajustes en el sistema operativo de forma automática [46].

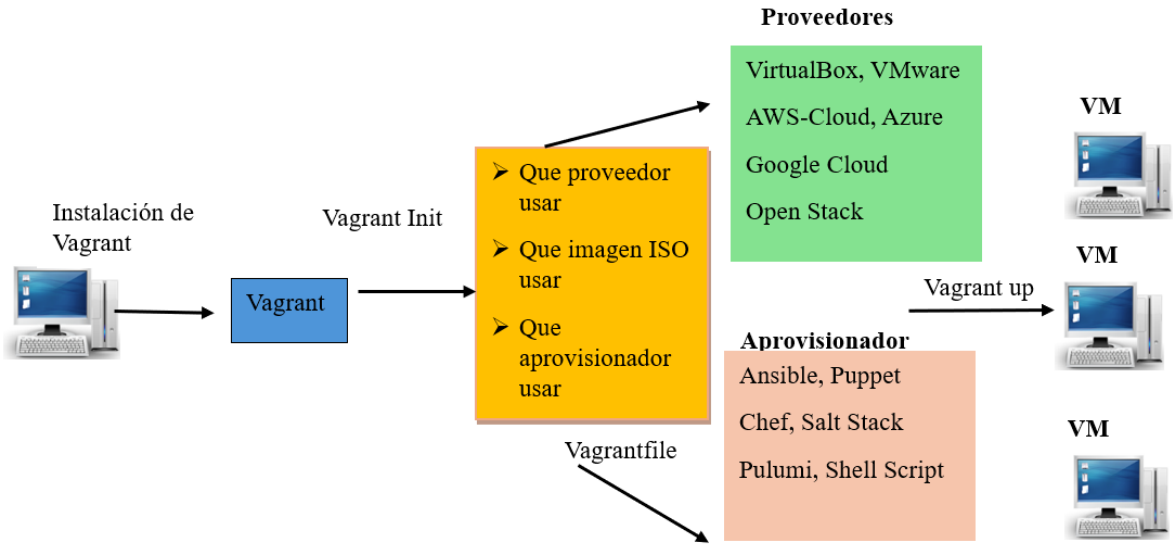

Figura 8. Aprovisionamiento de máquinas virtuales con Vagrant.

Mediante la Figura 8, el aprovisionamiento de máquinas virtuales comienza con la instalación de Vagrant en un dispositivo host, luego se da inicio al sistema con los comandos vagrant init, seguido se encuentra el archivo vagrantfile, el mismo que permite configurar las características de la máquina virtual, las opciones de qué proveedor usar y qué herramienta de aprovisionamiento, para trabajar en conjunto con Vagrant, y finalmente el despliegue se realiza cuando se ejecuta el comando vagrant up [47].

CAPÍTULO III

METODOLOGIA

En esta sección, una investigación sobre servidores y herramientas de aprovisionamiento existentes fue realizada inicialmente. Luego, el sistema fue abordado, siendo definidos los requisitos específicos y seleccionadas las tecnologías apropiadas. En la fase de desarrollo, un prototipo funcional del sistema aprovisionador fue creado. Posteriormente, las evaluaciones comparativas fueron llevadas a cabo para verificar la reducción de tiempos en la configuración del servidor de transferencia de archivos.

3.1 Tipo de investigación

La presente investigación fue de carácter experimental, a través del cual fue evaluado el aprovisionamiento de las herramientas, cuyos tiempos de configuración fueron comparados con las configuraciones manuales. Se empleó el enfoque cuantitativo para recopilar y analizar datos sobre el tiempo requerido en la configuración del servicio de transferencia de archivos [48].

3.2 Técnica documental

La información se recopiló de artículos científicos, libros, tesis, repositorios de investigación y documentos relacionados con el caso de estudio, dónde se clasificó para 2 fines, el primero para identificar los procesos de configuraciones y el segundo para determinar las de herramientas de software libre para automatizar servidores FTP [49]**.**

3.3 Técnica de recolección de datos

Se realizó mediante observación directa y medición sistemática, empleando dos métodos de control. Para la configuración automatizada, se implementaron logs que almacenaron los tiempos de ejecución del proceso de aprovisionamiento, y para la configuración manual, se empleó cronometraje controlado, documentando la duración de cada tarea de configuración.

3.4 Población de estudio y tamaño de la muestra

El universo poblacional de estudio en la presente investigación comprende los datos obtenidos del tiempo de configuración de un servidor FTP mediante el sistema aprovisionador y a través de la configuración manual; además, al utilizarse un entorno virtual, no se cuenta con un número definido de servidores, por tanto, se utilizó la fórmula para poblaciones infinitas. La muestra está definida por la siguiente fórmula estadística (1).

$$n = \frac{Z^2 * P * Q}{e^2} \quad (1)$$

$$n = \frac{1.88^2 * 0.5 * 0.5}{0.06^2} = 245$$

Donde:

n = Tamaño de la muestra.
Z = Nivel de confianza (NC), 94 %, equivalente a un Z = 1.88
p = Probabilidad de que ocurra el caso estudiado, 50 %.
q = Probabilidad de que no ocurra el caso estudiado, 50 %.
e = Error de estimación (Para este caso 6 % =0.06).
n = 245

3.4.1 Operacionalización de variables

Tabla 3. Variable dependiente.

Variable	Descripción	Método	Instrumento	Indicador
Tiempo de configuración del servidor	Tiempo que demora el aprovisionador en configurar el servidor FTP/configuración manual FTP.	Observación.	Cronómetro.	Segundos.

Tabla 4. Variables independientes.

Variable	Descripción	Método	Indicador
El tipo de configuración.	Forma que se configura el servidor, siendo a través del aprovisionador o configuración manual.	Observación directa.	**Tipo:1** Aprovisionador. **Tipo:2** Manual.

3.5 Hipótesis

H_0: Podrá un aprovisionador reducir el tiempo de configuración de un servidor FTP en comparación a una configuración manual**.**
Ha: No podrá un aprovisionador reducir el tiempo de configuración de un servidor FTP en comparación a una configuración manual.

3.6 Método de análisis y procesamiento de datos

El análisis y procesamiento de datos fue realizado sobre una muestra de 245 registros de los tiempos de configuración, que fueron obtenidos mediante la implementación del sistema aprovisionador y configuraciones manuales. SPSS fue utilizado como plataforma principal para el procesamiento estadístico de los datos. Por otra parte, se realizó un análisis descriptivo de los datos, incluyendo el cálculo de medidas de tendencia central como la media, así como medidas de dispersión, como la desviación estándar y el rango.

Adicionalmente, se llevaron a cabo pruebas de hipótesis para determinar la existencia de diferencias estadísticamente significativas entre los grupos de interés.

3.7 Proceso de la metodología

En esta sección se aborda la metodología y los pasos seguidos en el desarrollo del sistema aprovisionador.

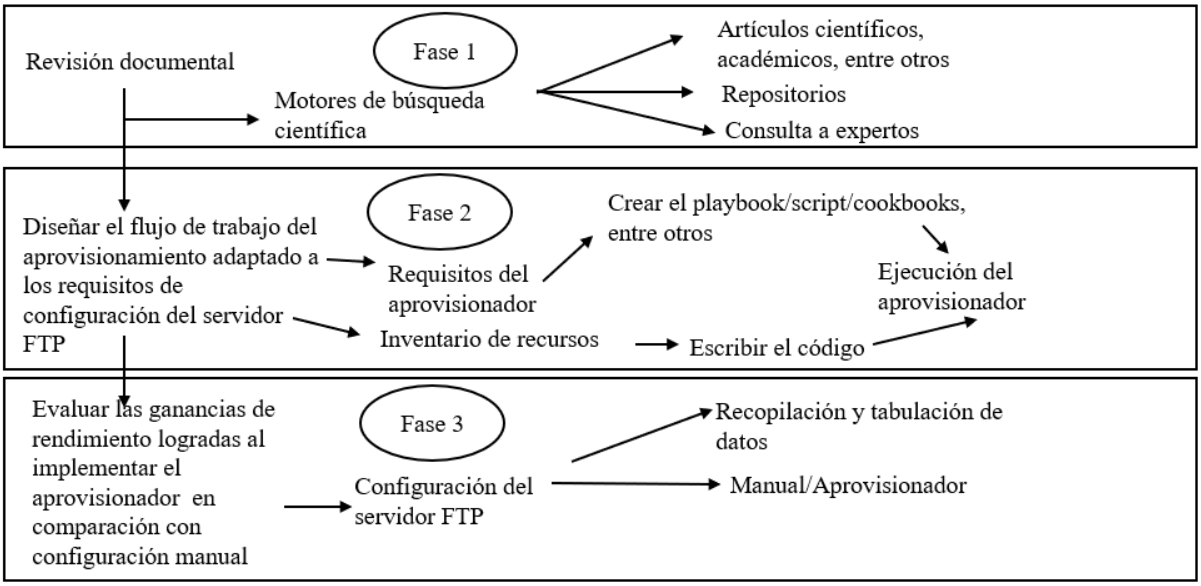

Figura 9. Fases de trabajo.

Fase 1: Revisión documental

La revisión documental se llevó a cabo utilizando motores de búsqueda científica como punto de partida para acceder a los recursos informativos. Comenzó con la identificación de conceptos y palabras claves como aprovisionamiento, virtualización y herramientas de código abierto relacionadas con la configuración de servidores. Estas palabras clave se utilizaron en la búsqueda para localizar artículos científicos, académicos, tesis.
La búsqueda se extendió a diversas bases de datos académicas, como IEEE, Google Scholar, Mendeley, entre otras.
Durante esta fase, se recopiló, organizó y analizó toda la información obtenida de 23 artículos, 22 tesis y 5 libros, a fin de ser gestionada, consultada y referenciada durante las fases siguientes de la investigación.

Fase 2: Diseñar el flujo de trabajo

Aquí se tradujo el conocimiento adquirido durante la revisión documental en un plan concreto para el desarrollo del sistema aprovisionador. El proceso comenzó con la selección de los requisitos de configuración del servidor, entre ellos esta, instalación de paquetes, configuración de firewall, habilitación de servicios y la creación de usuarios, entre otros.

Una vez establecidos los requisitos, se procede a la creación de los archivos de configuración, como el playbook, inventario, script y cookbook, que guiaron los procesos de configuración. La escritura de los códigos fue un proceso que implicó el uso de las estrategias de aprovisionamiento definidas por cada herramienta como Shell, Ansible y Chef. La ejecución de los códigos fue realizada desde el intérprete de comandos de líneas en Ubuntu con los comandos vagrant init para establecer el entorno virtualizado.

Fase 3: Evaluación de rendimiento

En la fase final del proyecto, se centró en la evaluación del sistema aprovisionador desarrollado, comparando su rendimiento con las configuraciones manuales del servidor. Esta etapa validó al sistema y cuantificó las mejoras en términos de tiempo.
El proceso de evaluación inició desde la configuración del despliegue de la máquina virtual hasta la ejecución de las tareas programadas; se establecieron tres instancias de servidores, cada una con diferente codificación correspondiente a cada herramienta.
Se emplearon técnicas estadísticas para analizar los resultados, como el análisis descriptivo y pruebas de hipótesis.

3.8 Requisitos de configuración de un servidor de transferencia de archivos

Los requisitos necesarios que fueron utilizados en la configuración del servidor se detallan a continuación como, el levantamiento de la máquina virtual, instalación de los paquetes del servidor de transferencia de archivos, la habilitación de los servicios, la creación de usuarios, entre otros.

3.8.1 Máquina virtual

La configuración de la máquina virtual se realizó siguiendo los requisitos especificados en la Tabla 5. Se utilizó VirtualBox como hipervisor principal en conjunto con Vagrant para automatizar el proceso de configuración de la máquina virtual. La imagen ISO seleccionada fue AlmaLinux. Los parámetros de configuración básica incluyeron la asignación de 2 GB de memoria RAM y 2 núcleos de CPU. La configuración de red se estableció utilizando NAT (Network Address Translation) con un adaptador puente; por tanto, permite una conectividad con la máquina host como con la red externa [50].

Tabla 5. Requisitos de configuración básica de la máquina virtual.

Hipervisor	VirtualBox
Imagen ISO	AlmaLinux
Memoria RAM	4 G
Número de núcleos de CPU	2
Configuración de red	NAT /Adaptador Puente.

Es importante haber configurado adecuadamente estos aspectos. De ello depende una instalación garantizada, segura y estable en el entorno virtualizado.

3.8.2 Configuración del Servidor

La configuración de Vsftpd (Very Secure FTP Daemon) implicó varios aspectos importantes para garantizar un servicio de transferencia de archivos seguro y funcional. A continuación, se observan puntos claves para configurar el servidor.

Tabla 6. Paquete de instalación del servidor.

dnf install vsftpd -y	Instala vsftpd y con el indicador '-y' se confirma de manera automática.
cd/etc/vsftpd	Ruta donde se encuentran los siguientes archivos de configuración ftpuser, user_list, vsftpd.conf, estos permiten la gestión del servidor ftp

Tabla 7. Permisos requeridos en la instalación del servidor.

anonymous_enable=YES	Habilita el acceso anónimo al servidor FTP.
no_anon_password=YES	Requiere que los usuarios anónimos no ingresen una contraseña para conectarse.
local_enable=YES	Habilita el acceso de usuarios locales del sistema al servidor
write_enable=YES	Permite que todos los usuarios (anónimos y locales) tengan permisos de escritura.
ftpd_banner=Bienvenido a ftp	Este es el mensaje de bienvenida que se muestra cuando un usuario se conecta al servidor FTP.
chroot_local_user=YES	Restringe a los usuarios locales a su directorio home y los subdirectorios debajo de él.
allow_writeable_chroot=YES	Permite que los usuarios locales puedan escribir (cargar, modificar, eliminar) dentro de su entorno chroot.
userlist_deny=NO	Indica que el archivo user_list contiene una lista de usuarios permitidos.

Con las configuraciones realizadas, el servidor de transferencia de archivos admitió conexiones de dos tipos de usuarios. La primera, los usuarios sin cuentas en el sistema pudieron acceder de forma anónima sin necesidad de credenciales; sin embargo, estos usuarios accedieron a la lectura de los documentos públicos. La segunda, los usuarios registrados localmente en la máquina virtual también tuvieron permiso para conectarse, utilizando sus nombres de usuario y contraseñas establecidas; por otra parte, fue restringido el acceso exclusivamente a sus respectivos directorios personales y subdirectorios sin poder navegar a otras áreas del sistema de archivos. A diferencia de los usuarios anónimos, sí podían realizar modificaciones a los archivos que se encuentran en sus respectivos directorios. Por otra parte, los ajustes se realizaron en el archivo vsftpd.conf.

3.8.3 Configuración del firewall

Tabla 8. Comandos usados en la configuración del firewall.

firewall-cmd --zone=public --permanent --add-service=ftp	Agrega el servicio FTP a la zona pública del firewall de manera permanente.
firewall-cmd --zone=public --permanent --add-port=20-21/tcp	Agrega explícitamente los puertos 20 y 21 para el tráfico TCP a la zona pública del firewall de manera permanente
firewall-cmd --reload	Este comando recarga la configuración del firewall para aplicar las nuevas reglas.
firewall-cmd --list-all	Muestra todas las reglas y configuraciones actuales del firewall.

Estos comandos abrieron los puertos necesarios y habilitaron el servicio FTP en la zona pública del firewall. Además, permitieron que las conexiones FTP entrantes desde redes no confiables o internet llegaran al servidor.

3.8.4 Habilitación de servicios

Tabla 9. Códigos de habilitación de los servicios configurados.

systemctl start vsftpd	Este comando inicia el servicio vsftpd inmediatamente.
systemctl enable vsftpd	Configura el servicio vsftpd para que se inicie automáticamente cada vez que el sistema arranca.
systemctl status vsftpd	El comando muestra el estado actual del servicio vsftpd.
dnf install ftp -y	Instala el cliente de transferencia de archivos en el servidor.

En conjunto, estos códigos iniciaron el servicio FTP automáticamente en el arranque del sistema y después verificaron el estado actual del servidor.

3.8.5 Creación de usuarios

Tabla 10. Códigos usados en la creación de usuarios.

useradd ftpdarwin	Crea un nuevo usuario en el sistema con el nombre "ftpdarwin".
passwd ftpdarwin	Este comando se utiliza para establecer la contraseña del usuario "ftpdarwin".

Conforme se evidencia, los códigos ayudaron a crear el usuario y deben ser ejecutados, con privilegios de superusuario, siendo necesarios para crear usuarios y establecer contraseñas.

3.8.6 Configuración de la máquina virtual

```
# -*- mode: ruby -*-
# vi: set ft=ruby:
Vagrant.configure("2") do |config|
  config.vm. define "nodo1" do |nodo1|
   nodo1.vm.box = "boxomatic/almalinux-9"
   nodo1.vm.hostname = 'nodo1'
   nodo1.vm.network: public_network,:bridge=>"eth0"
  end
  config.vm. provider "virtualbox" do |vb|
    vb.gui = true
    vb. memory = "4096"
    vb. cpus = 2
    vb. customize ["modifyvm", :id, "--usb", "on"]
    vb. customize ["modifyvm", :id, "--usbxhci", "on"]
    vb. customize ["modifyvm", :id, "--macaddress1", "auto"]
    vb. customize ["modifyvm", :id, "--clipboard", "bidirectional"]
    vb. customize ["modifyvm", :id, "--draganddrop", "bidirectional"]
    vb. customize ["modifyvm", :id, "--uart1", "0x3F8", "4"]
  end
```

Script 3. Códigos usados en la configuración de la máquina virtual.

Los códigos del Script 3 pertenecen al archivo Vagranfile, que contiene la configuración de la máquina virtual; el archivo fue escrito en el lenguaje Ruby, donde se asignó el nombre como nodo1, y la red se configuró en puente. Además, se especificaron las configuraciones para el proveedor VirtualBox; por otra parte, se asignaron 4096 MB de memoria RAM y se configuró el procesamiento en 2 CPUs, y, por último, se habilitaron varias características de VirtualBox como USB, USB 3.0, la compartición de carpetas en bidireccional, arrastrar y soltar.

3.9 Aprovisionamiento Manual

El aprovisionamiento manual comenzó desde la descarga del sistema operativo AlmaLinux, donde se realizaron las configuraciones respectivas en VirtualBox, instalación del sistema operativo en la máquina virtual, y después continuaron las configuraciones del servidor, donde se incluyó la creación de usuario y la configuración de los permisos del firewall.

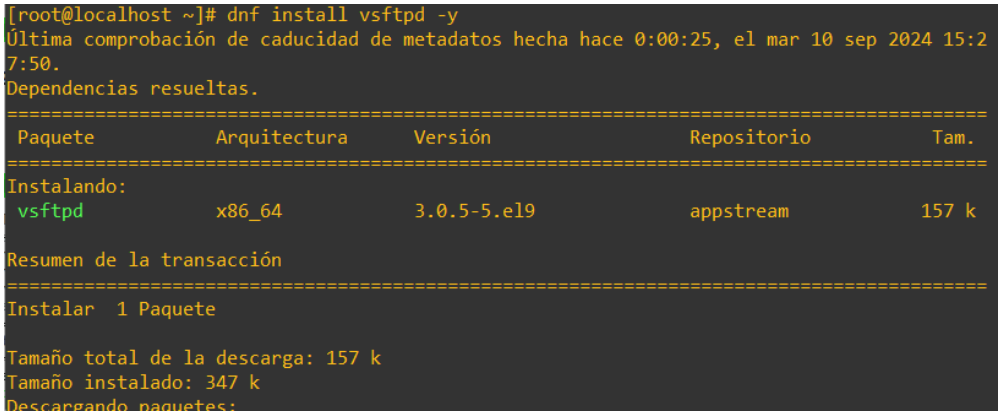

Figura 10. Instalación del paquete vsftpd.

Según la Figura 10, la instalación del paquete vsftpd se realizó con el comando (dnf install vsftpd -y). En el sistema operativo Alma Linux, esta instalación fue ejecutada como superusuario.

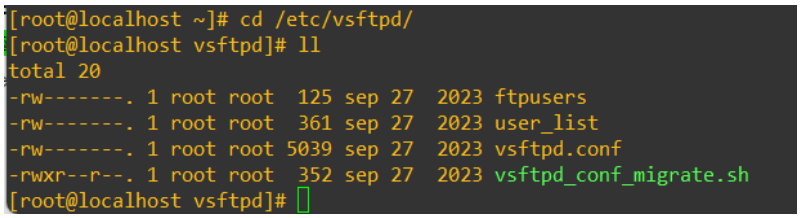

Figura 11. Lista de archivos del directorio vsftpd.

En la lista de archivos del directorio vsftpd se encuentran cuatro archivos; el de mayor importancia es el vsftpd.conf; en este archivo se han configurado los permisos correspondientes a la Tabla 7. Estos permisos fueron necesarios para lograr la transferencia de archivos.

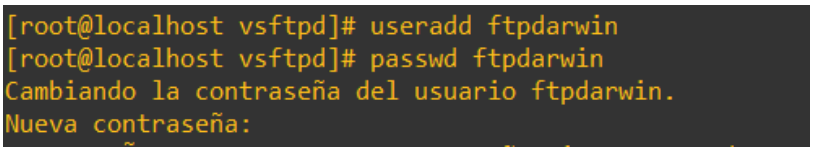

Figura 12. Creación de usuario y contraseña.

La creación del usuario se realizó con el siguiente comando (useradd ftpdarwin).
Luego, se ejecutó el comando passwd ftpdarwin para establecer la contraseña (12345). Por otra parte, el sistema respondió con un mensaje que indica el cambio de la contraseña respectiva.

Figura 13. Configuración del firewall.

La configuración del firewall presentada en la Figura 13 contiene las reglas activadas de la Tabla 8, necesarias en la habilitación de los puertos 20-21/TCP a la zona pública.

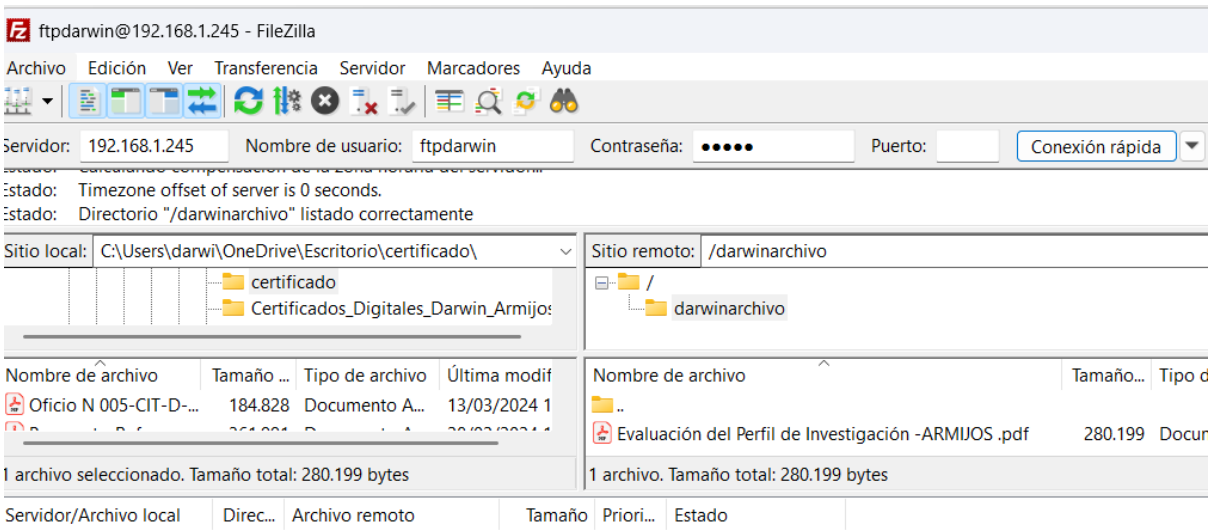

Figura 14. Prueba de transferencia de archivos cliente-servidor.

De acuerdo con la Figura 14, la trasferencia de archivos entre el cliente FileZilla y el servidor FTP configurado de manera manual, se logró transferir el archivo de prueba en el servidor; para esto se utilizó el usuario creado con su respectiva contraseña (ftpdarwin/12345) y su dirección IP (Internet Protocol) 192.168.1.245 asignada al servidor.

3.10 Aprovisionamiento con Shell Script

En este método fue utilizado el archivo (.sh) y la herramienta Vagrant; con la combinación de estas se logró la configuración del servidor alojado en la máquina virtual.

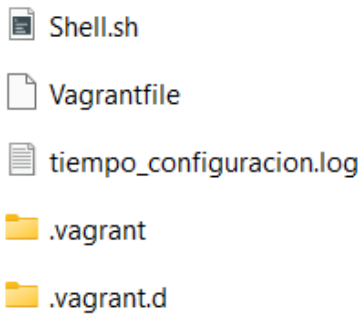

Figura 15. Lista de archivos y directorios del aprovisionador Shell Script.

Según la Figura 15, los archivos usados en el aprovisionamiento del servidor FTP han sido el Vagrantfile y el Shell.sh. El primero ayudó en la configuración de la máquina virtual y el segundo realizó el aprovisionamiento del servidor.

```
config.vm. provision "shell", path: "Shell.sh"
```

Script 4. Codificación de la ruta al script desde Vagrant.

En la línea de código del Script 4 fue configurado el archivo Shell.sh para ser ejecutado como aprovisionador desde el Vagranfile.

```
echo "actualiza el servidor e instala en el paquete vsftpd "
dnf update
dnf install -y vsftpd
```

Script 5. Código de instalación del paquete vsftpd en Alma Linux.

En el Script 5, se encuentra la codificación usada en la instalación del paquete del servidor y la actualización de los paquetes instalados.

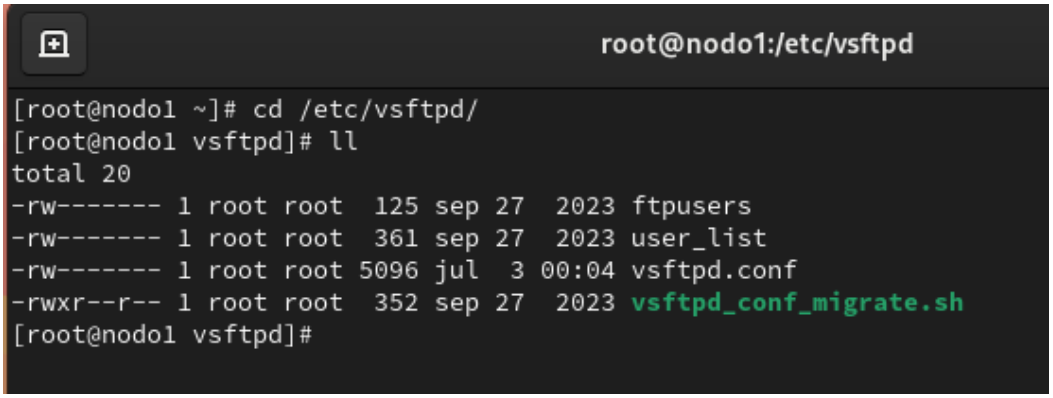

Figura 16. Lista de archivos del directorio vsftpd.

La lista de archivos que contiene la Figura 16 fueron instalados por el paquete vsftpd del servidor; además, estos archivos son ubicados en la ruta (cd/etc/vsftpd). Por otra parte, el archivo vsftpd.conf se configuró mediante el script de Shell.

```
# Configurar vsftpd en el vsftpd.config
sed -i 's/anonymous_enable=NO/anonymous_enable=YES/' /etc/vsftpd/vsftpd.conf
sed -i 's/#no_anon_password=YES/no_anon_password=YES/' /etc/vsftpd/vsftpd.conf
sed -i 's/local_enable=NO/local_enable=YES/' /etc/vsftpd/vsftpd.conf
sed -i 's/#write_enable=YES/write_enable=YES/' /etc/vsftpd/vsftpd.conf
echo 'ftpd_banner=Bienvenidos a ftp' >> /etc/vsftpd/vsftpd.conf
sed -i 's/#chroot_local_user=YES/chroot_local_user=YES/' /etc/vsftpd/vsftpd.conf
echo 'allow_writeable_chroot=YES' >> /etc/vsftpd/vsftpd.conf
sudo dnf install ftp -y
```

Script 6. Códigos de configuración del archivo vsftpd.conf.

Los permisos que fueron usados en la configuración del archivo vsftpd.conf se encuentran en el Script 6, cada comando sed (Stream editor) modificó una línea específica en el archivo de configuración, cambió las opciones o descomentó las líneas especificadas; además, los comandos (-i), opción que significa en el lugar, permitieron a sed modificar el archivo directamente. Por otra parte, los comandos (echo) añadieron nuevas líneas al archivo de configuración y los comandos (>>) indican el redireccionamiento de la ruta del archivo configurado.

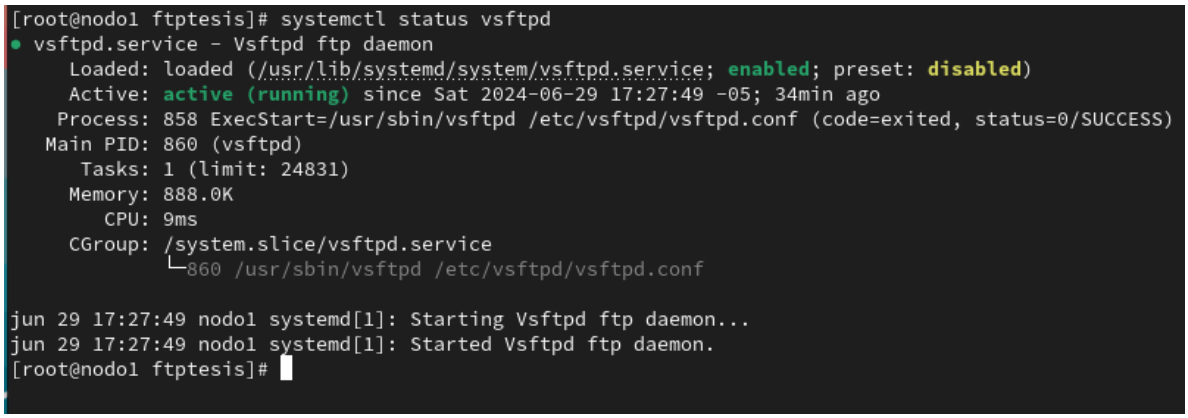

Figura 17. Verificación del estado del servidor de transferencia de archivos.

La Figura 17 presenta la verificación del estado del servidor realizada con los comandos (systemctl status vsftpd). El servidor estaba habilitado y activado.

```
echo "Configurar el firewall "
systemctl start firewalld
firewall-cmd --zone=public --permanent --add-service=ftp
firewall-cmd --zone=public --permanent --add-port=20-21/tcp
firewall-cmd --reload
firewall-cmd --list-all
```

Script 7. Códigos de configuración del firewall.

El Script 7, contiene las líneas de código que fueron escritas para configurar el firewall, donde se habilitaron los puertos 20-21 TCP; además, se permitió el tráfico FTP en la zona pública.

Por otra parte, el comando (firewall-cmd --list-all) fue utilizado para comprobar las configuraciones en la distribución AlmaLinux.

Figura 18. Verificación del estado del firewall.

En la Figura 18, se verificó la configuración del firewall con los comandos (firewall-cmd --list-all). Muestra que el firewall fue configurado para permitir los servicios como FTP, SSH, y los puertos asociados con FTP (20-21/TCP).

```
echo "Crear usuario FTP "
# Crear usuario FTP
useradd -m -s /bin/bash ftptesis
echo "ftptesis:12345" | chpasswd
# Configurar sudo para el nuevo usuario
echo "ftptesis ALL=(ALL) NOPASSWD: ALL" >> /etc/sudoers
```

Script 8. Códigos usados en la creación de usuarios.

En el Script 8, se realizó la creación de un usuario llamado ftptesis; posteriormente, fue añadido en el sistema en el directorio home y el shell bash como shell por defecto. Por otra parte, la contraseña (12345) fue asignada a este usuario mediante el comando (chpasswd). Finalmente, los privilegios de superusuario fueron otorgados con el propósito de establecer un nuevo usuario con los permisos en el sistema.

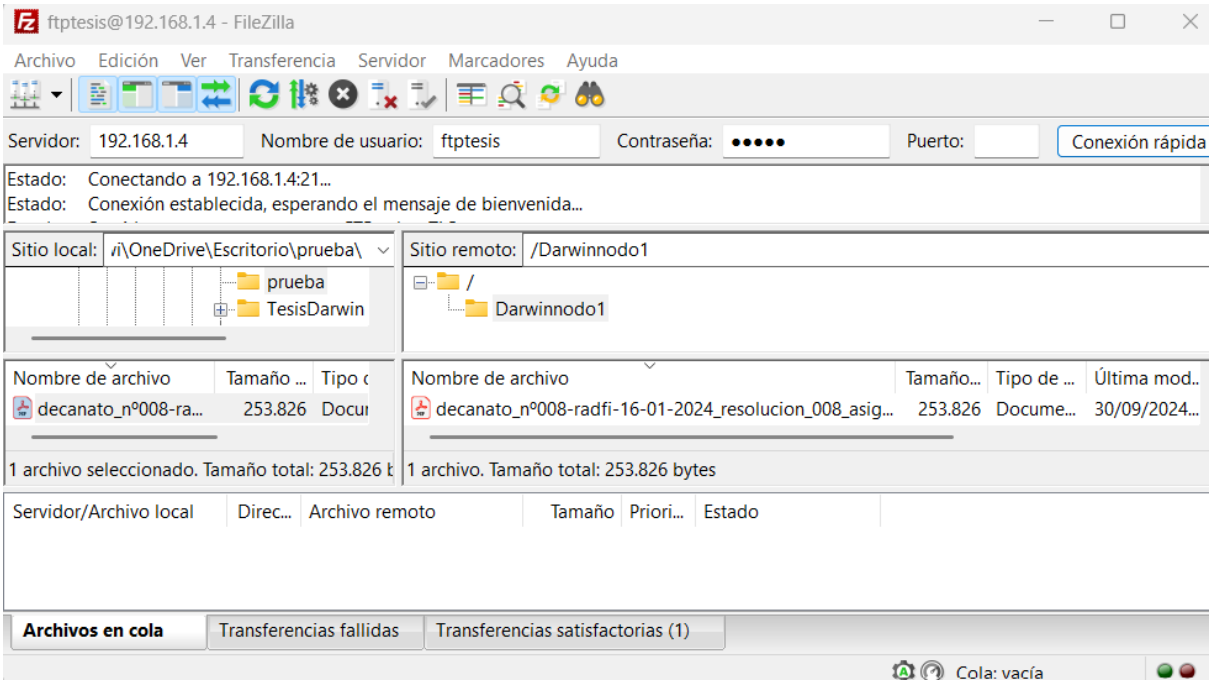

Figura 19. Transferencia de archivos cliente-servidor.

Como presenta la Figura 19, se realizó la transferencia de archivo, utilizando el usuario configurado denominado ftptesis con su respectiva contraseña 12345 en la red LAN con la dirección IP asignada al servidor 192.168.1.4. Se obtuvo el ingreso con éxito, de igual manera la transferencia del archivo pdf.

3.11 Aprovisionamiento con Chef

Para el aprovisionamiento con Chef fue necesaria la instalación de esta herramienta en Ubuntu, útil para poder desplegar el aprovisionamiento usando los cookbooks. Por otra parte, en el entorno de desarrollo se utilizó la estación de trabajo Chef y Vagrant.

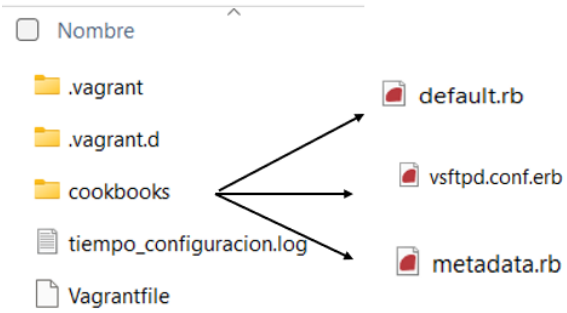

Figura 20. Lista de archivos y directorios del aprovisionador Chef.

La Figura 20, contiene los archivos que fueron utilizados para aprovisionar el servidor FTP con la herramienta Chef, los cuales fueron:
Vagrantfile: Estableció la configuración de la máquina virtual.
metadata.rb: contiene información especificada sobre los cookbooks de Chef.
default.rb (en recipes): Contribuyó con las instrucciones para configurar el servidor FTP.
vsftpd.conf.erb: Plantilla que generó el archivo de configuración del servidor FTP.

```
config.vm. provision "chef_solo" do |chef|
 chef. arguments = "--chef-license accept"
 chef.add_recipe "ftp-server"
 chef. cookbooks_path = ["cookbooks"]
 end
```

Script 9. Codificación de la ruta al cookbooks desde Vagrant.

Los códigos utilizados en el Script 9 ayudaron a establecer la ruta desde el Vagrantfile hacia los cookbooks y el recipe; además, se aceptó la licencia del servidor Chef_Solo. Por otra parte, la versión de Chef instalada en la distribución de Linux se revisó con el siguiente comando (chef-client –version).

```
[root@nodo2 ftp-server]# cd /var/chef/cache/cookbooks/ftp-server/
[root@nodo2 ftp-server]# ll
total 12
-rw------- 1 root root  242 jul 10 13:27 metadata.rb
drwxr-xr-x 2 root root 4096 jul 10 13:27 recipes
drwxr-xr-x 2 root root 4096 jul 10 13:27 templates
[root@nodo2 ftp-server]#
```

Figura 21. Lista de archivos de configuración en la distribución Linux.

A diferencia de los archivos de configuración que se encuentran en la Figura 20, los archivos que están en Alma Linux en la dirección (cd/var/chef/cache/cookbooks/ftp-server/) fueron copiados desde el directorio principal donde se encuentra el Vagranfile, ya que este proceso

se realizó automáticamente debido a que Chef usa agentes, es decir necesita tener los archivos en el servidor que se va a configurar para realizar el aprovisionamiento.

```
# Configuración básica del servidor FTP
anonymous_enable=YES
no_anon_password=YES
local_enable=YES
write_enable=YES
local_umask=022
dirmessage_enable=YES
xferlog_enable=YES
connect_from_port_20=YES
xferlog_std_format=YES
listen=YES
pam_service_name=vsftpd
userlist_enable=YES
ftpd_banner=Bienvenidos a ftp
chroot_local_user=YES
allow_writeable_chroot=YES
```

Script 10. Códigos de configuración del archivo vsftpd.conf.

De acuerdo con los códigos del Script 10, conformaron el archivo vsftpd.conf.erb, perteneciente a los cookbooks, y este archivo está ubicado en el directorio de plantillas (templates), donde estas configuraciones fueron copiadas a la distribución de Linux y alojadas en la ruta (cd/var/chef/cache/cookbooks/ftp-server/) para ser usadas desde esta ruta. Para considerar, este es un archivo erb (Embedded Ruby) que ayudó en la configuración del paquete vsftpd. Estos archivos (.erb) permitieron generar configuraciones dinámicas utilizando el lenguaje Ruby.

```
# Instalar y configurar vsftpd
dnf_package 'vsftpd' do
  action: install
end
template '/etc/vsftpd/vsftpd.conf' do
  source 'vsftpd.conf.erb'
  notifies: restart, 'service[vsftpd]',: delayed
end
```

Script 11. Estructura de código de ruta del archivo vsftpd.conf.erb.

El archivo default.rb fue configurado con los códigos del Script 11; además, se estableció la estructura para generar el enlace del archivo vsftpd.conf.erb. Este archivo fue la estructura del recipe de Chef, ya que se utilizó como punto de entrada principal para las acciones realizadas en el sistema cuando se utilizó el cookbook.

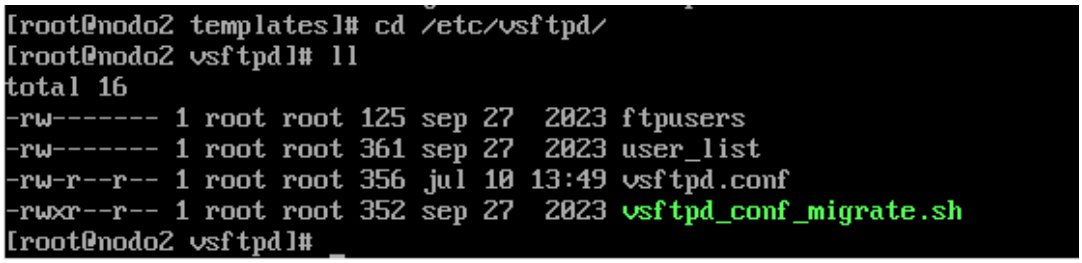

Figura 22. Lista de archivos del directorio vsftpd.

Revisando la ruta (cd/etc/vsftpd/) en la distribución de Linux, se observó que los mismos archivos que se presentaron en la Figura 16, son idénticos a los usados en Shell. Estos archivos fueron necesarios para la configuración del servidor mediante Chef.

```
# Configurar el firewall
service 'firewalld' do
  action [: enable, start]
end
execute 'firewall_ftp' do
  command 'firewall-cmd --zone=public --permanent --add-service=ftp'
  notifies: run, 'execute[firewall_reload]',: delayed
end
execute 'firewall_ports' do
  command 'firewall-cmd --zone=public --permanent --add-port=20-21/tcp'
  notifies: run, 'execute[firewall_reload]',: delayed
end
execute 'firewall_reload' do
  command 'firewall-cmd --reload'
  action: nothing
end
```

Script 12. Configuración del firewall mediante Chef.

El firewall fue habilitado con los códigos del Script 12, posteriormente, los comandos (firewall-cmd) fueron ejecutados para añadir el servicio FTP y los puertos 20-21/TCP a la zona pública. Cada uno de estos comandos fue configurado para notificar la ejecución del firewall; esta codificación fue escrita en el lenguaje Ruby y almacenada en el archivo default.rb.

```
username = 'ftptesis'
password = '12345'
user username do
  password "$1$JJsvHslasdfjVEroftprNn4JHtDi"
  manage_home true
  shell '/bin/bash'
  action: create
end
execute 'set_user_password' do
  command "echo '#{username}: #{password}' | chpasswd"
  action: run
end
```

Script 13. Códigos usados en la creación de usuario.

Una serie de acciones fueron ejecutadas para gestionar la creación y configuración de un usuario en el sistema. En el Script 13, un nuevo usuario denominado ftptesis fue creado en el directorio home gestionado automáticamente y el shell /bin/bash asignado como su intérprete de comandos predeterminado. Inicialmente, una contraseña codificada fue establecida para este usuario. Posteriormente, esta contraseña fue sobrescrita mediante un comando ejecutado (chpasswd) para establecer la contraseña (12345) de manera segura; por otra parte, estos códigos fueron escritos en el archivo default.rb debido a que conformaron el recipe de Chef.

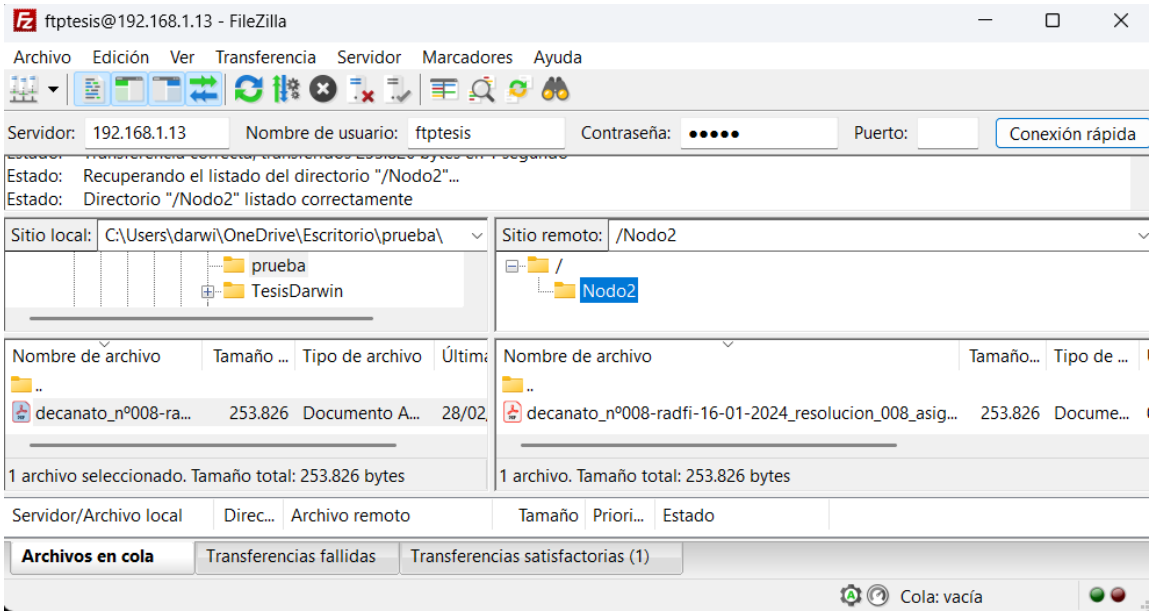

Figura 23. Transferencia de archivos cliente-servidor.

Con relación a la Figura 23, la trasferencia de archivo cliente-servidor fue realizada mediante el uso del usuario (ftptesis) y la contraseña respectiva (12345) en la dirección de red LAN con la IP asignada 192.168.1.13 en la interface del servidor.

3.12 Aprovisionamiento con Ansible

Para el aprovisionamiento realizado con esta herramienta fue necesario instalarla en Ubuntu, además para verificar la instalación se usaron los comandos (ansible - -version).

Por otra parte, es necesario tener en cuenta que la versión sea compatible con Vagrant, ya que el aprovisionamiento fue realizado con la integración de estas.

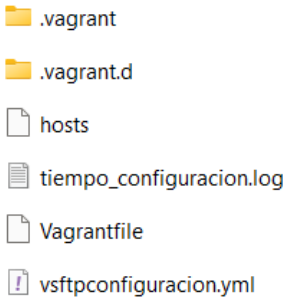

Figura 24. Lista de archivos y directorios del aprovisionador Ansible.

Como se evidencia en la Figura 24, los directorios vagrant y vagrant.d fueron creados de manera automática; el primero almacena los datos de estado y configuración de la máquina virtual, mientras que el segundo fue utilizado para el almacenamiento de los datos generales del sistema. El archivo hosts fue configurado para gestionar las asignaciones de nombres de host y direcciones IP. El Vagrantfile fue definido para contener la configuración principal de la máquina virtual, y, finalmente, el archivo vsftpconfiguracion.yml fue implementado para establecer los parámetros específicos del servidor FTP en formato yml.

```
config.vm. provision "ansible" do |a|
a. compatibility_mode = "2.0"
a. playbook = "vsftpconfiguracion.yml"
a. inventory_path = "hosts"
a. limit = "all"
end
```

Script 14. Codificación de la ruta al playbook desde Vagrant.

Los códigos ubicados en el Script 14, establecieron la ruta de conexión en el Vagranfile. El archivo vsftpconfiguracion.yml fue designado como el playbook principal que contiene todas las tareas que fueron realizadas en el servidor. El archivo hosts fue utilizado como el inventario, donde fue definido el servidor y host que fueron gestionados por Ansible. Finalmente, el parámetro limit fue configurado como all para asegurar un aprovisionamiento completo.

```
---
- hosts: all
  gather_facts: yes
  become: yes
  tasks:
    - name: Instala VSFFTP server
      yum:
        name: vsftpd
        state: present
      when: ssh_test is succeeded
    - name: inicio y habilito VSFFTP service
      service:
        name: vsftpd
        state: started
        enabled: yes
      when: ssh_test is succeeded
```

Script 15. Código de instalación del paquete vsftpd del servidor.

Los códigos utilizados en la instalación del paquete vsftpd mediante Ansible constan en el Script 15. Estos códigos conformaron parte del archivo vsftpconfiguracion.yml; el archivo está codificado para ejecutarse en el host que fue definido por el inventario; además, realizó la instalación del paquete vsftpd mediante el gestor de paquetes yum. Posteriormente, el servicio vsftpd fue iniciado y habilitado para arrancar automáticamente durante el inicio del sistema. Por otra parte, el comando (name) definió el nombre de la tarea; (service) indica que se usó el módulo de servicio de Ansible; (state y started) manifestaron que el servicio

esta inicializado, y, por último, (yes y enable) configuraron el servicio para que se inicie automáticamente.

```
- name: Configuración vsftpd server
  lineinfile:
    path: /etc/vsftpd/vsftpd.conf
    regexp: "{{item. regexp}} "
    line: "{{item. line}} "
  with_items:
    - {regexp: '^anonymous_enable=NO', line: 'anonymous_enable=YES'}
    - {regexp: '^#no_anon_password=YES', line: 'no_anon_password=YES'}
    - {regexp: '^local_enable=NO', line: 'local_enable=YES'}
    - {regexp: '^#write_enable=YES', line: 'write_enable=YES'
    - {regexp: '^#chroot_local_user=YES', line: 'chroot_local_user=YES'}
    - {regexp: '^#allow_writeable_chroot=YES', line: 'allow_writeable_chroot=YES'}
  notify:
    - restart vsftpd
  when: ssh_test is succeeded
- name: AGREGAR FTP banner
  lineinfile:
    path: /etc/vsftpd/vsftpd.conf
    line: 'ftpd_banner=Bienvenidos a ftp'
  when: ssh_test is succeeded
- name: Instala ftp client
  dnf:
    name:
      - ftp
    state: present
```

Script 16. Códigos de configuración del archivo vsftpd.conf.

Las líneas de código del Script 16 configuraron los permisos de la Tabla 7. Estos códigos conformaron el playbook de Ansible, donde fue habilitado el acceso a los usuarios anónimos, acceso local, permisos de lectura y escritura. Como parte final, los comandos, (when) estableció una condición para ejecutar la tarea, (lineinfile) utilizó el módulo lineinfile para modificar líneas, (path) especificó la ruta del archivo a modificar, (regexp y line) definieron el patrón a buscar y la línea a insertar/reemplazar, (with_items) permitió iterar sobre una lista de elementos y finalmente (notify) indicó que reinicia el servicio vsftpd si hay cambios.

```
- name: Configuracion de firewall
  block:
    - systemd:
        state: started
        enabled: yes
        name: firewalld
    - firewalld:
        zone: public
        permanent: yes
        service: ftp
        state: enabled
    - firewalld:
        zone: public
        permanent: yes
        port: 20-21/tcp
        state: enabled
  command: firewall-cmd --reload
```

Script 17. Configuración del firewall mediante Ansible.

Mediante los códigos del Script 17, la configuración del firewall y de los puertos 20-21/TCP fue establecida en varios pasos; primeramente, el servicio firewalld fue inicializado a través del comando (systemd), después se habilitó la zona pública del firewall con los comandos (zone public permanent), y como paso final, todas las reglas del firewall fueron recargadas mediante el comando (firewall-cmd - -reload) para que los cambios fueran aplicados en el sistema; por otra parte, estos códigos conformaron parte del archivo vsftpconfiguracion.yml.

```
- name: Crear usuario FTP
  user:
    name: ftptesis
    create_home: yes
    shell: /bin/bash
- name: Establecer contraseña para el usuario FTP
  user:
    name: ftptesis
    password: "{{'12345' | password_hash('sha512')}}"
- name: Configurar sudo para el nuevo usuario
  lineinfile:
    path: /etc/sudoers
    line: "ftptesis ALL=(ALL) NOPASSWD: ALL"
    validate: /usr/sbin/visudo -cf %s
```

Script 18. Códigos usados en la creación de usuario.

La estructura de código que contiene el Script 18 fue utilizada para la creación del usuario en el sistema bajo el nombre ftptesis con la contraseña (12345), donde fue asignado al directorio home y el shell bash como su intérprete; además, fueron configurados los permisos de superusuario. Por otra parte, los códigos más representativos fueron:
user: Indicó que se usa el módulo user de Ansible.
name: ftptesis: Especificó el nombre del usuario creado.
create_home: yes: Dio la instrucción de crear un directorio home para el usuario.
shell: /bin/bash: Estableció el shell por defecto del usuario como bash.
name: ftptesis: Especificó el usuario al que se le asignará la contraseña.

path: /etc/sudoers: Especificó el archivo a modificar.

```
[vagrant]
default ansible_host=172.17.112.1 ansible_port=2222 ansible_user=vagrant
ansible_private_key_file=~/. vagrant.d/insecure_private_key
```

Script 19. Codificación del inventario en Ansible.

Como muestra el Script 19, la configuración del inventario para Ansible fue establecida para el host denominado default bajo el grupo vagrant. Los parámetros de conexión fueron definidos de la siguiente manera; la dirección IP del host fue establecida como 172.17.112.1, el puerto SSH fue configurado en el número 2222, el usuario de acceso fue especificado como vagrant, y la autenticación fue configurada para utilizar una llave privada SSH que fue ubicada en la ruta (~/. vagrant.d/insecure_private_key). Todas estas configuraciones fueron establecidas para permitir que Ansible fuera capaz de conectarse y gestionar el host vagrant.

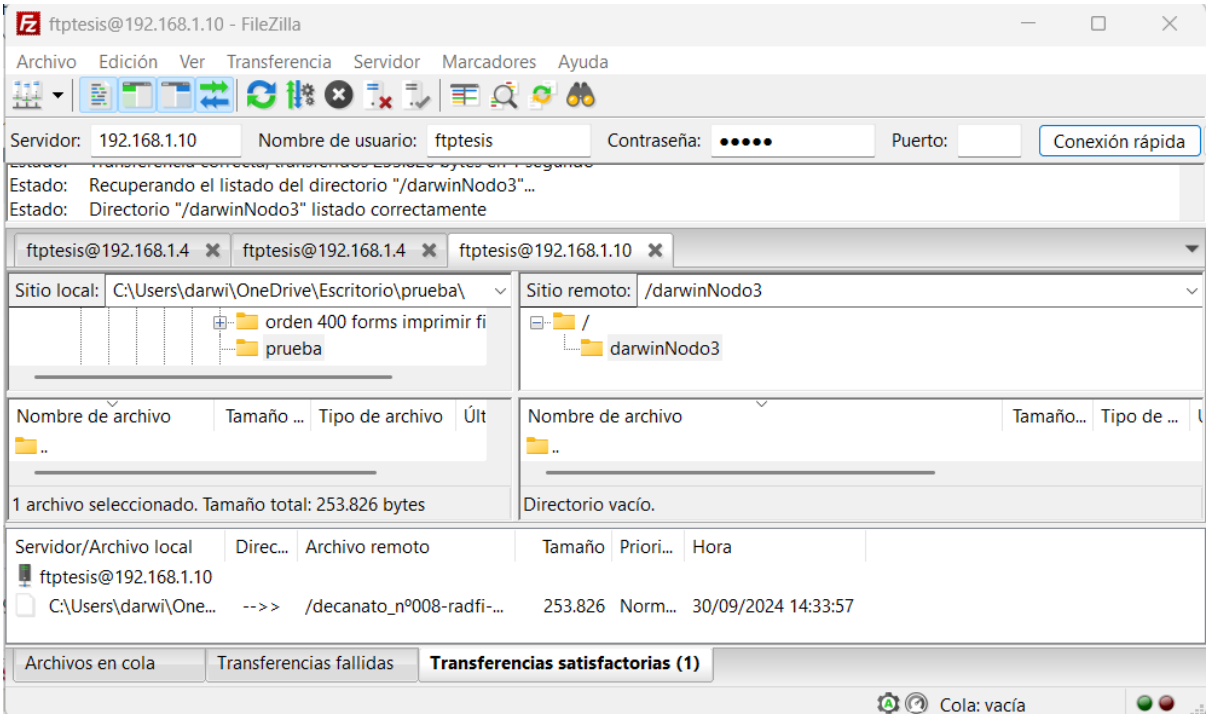

Figura 25. Transferencia de archivos cliente-servidor.

En la prueba de transferencias de archivos que se realizó en la Figura 25 mediante el servidor y el cliente filezilla usando el usuario ftptesis en la red LAN con IP del servidor 192.168.1.10, se logró transferir el archivo entre ellos de una manera intuitiva sin dificultad.

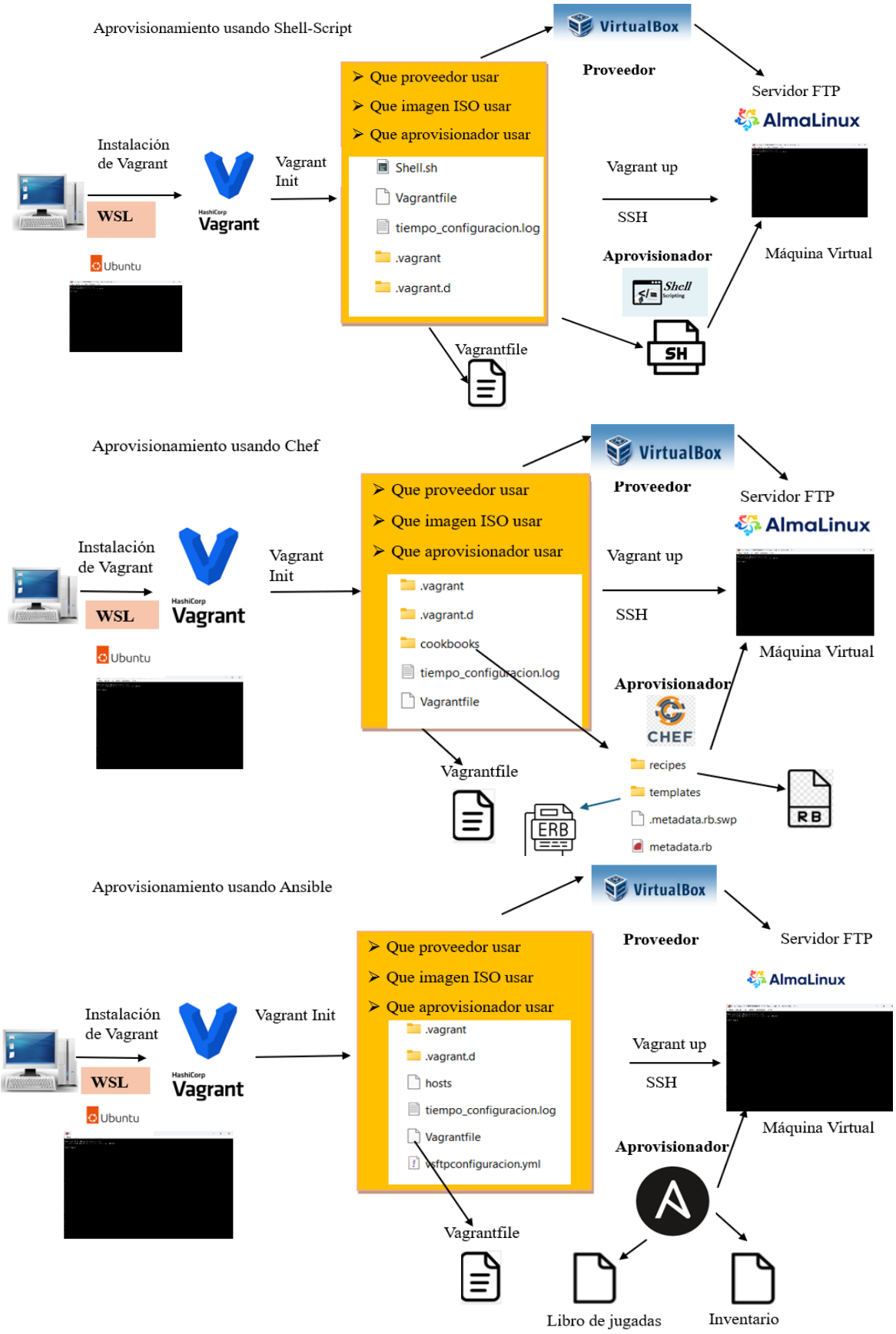

Figura 26. Tipos de Aprovisionamiento.

De acuerdo con la Figura 26, presenta cómo se utilizó Vagrant para aprovisionar un servidor FTP en una máquina virtual alojada en Virtualbox mediante el uso de las tecnologías como AlmaLinux, Ubuntu, Shell Script, Chef y Ansible. Los tipos de aprovisionamientos involucraron varias etapas, como el proceso general de la instalación de Vagrant en el sistema operativo de Ubuntu y Windows.

Una vez que Vagrant fue instalado e inicializado con el comando (vagrant init), fueron generados los archivos de configuración, como el Vagrantfile, Shell.sh para Shell, default.rb, vsftpd.conf.erb, metadata.rb para Chef, vsftpconfiguracion.yml y el host para Ansible.

El Vagrantfile fue configurado para utilizar VirtualBox como hipervisor y AlmaLinux como sistema operativo. Cuando los comandos (vagrant up) fueron ejecutados, una máquina virtual fue creada y aprovisionada mediante los archivos de configuración correspondientes para cada caso.

Primer caso (Shell); Shell.sh fue empleado como script donde los comandos de instalación y configuración del servidor FTP fueron escritos para ser ejecutados durante el aprovisionamiento.

En el segundo caso (Chef), default.rb fue utilizado como receta principal donde las instrucciones de instalación y configuración fueron codificadas y vsftpd.conf.erb fue empleado como plantilla donde la configuración personalizada del servidor FTP fue definida. Además, metadata.rb fue usado para declarar las dependencias, información y recursos necesarios para las recetas de Chef.

Tercer caso (Ansible), vsftpconfiguracion.yml fue implementado como playbook donde las tareas de instalación y configuración del servidor FTP fueron definidas, y finalmente el archivo hosts fue utilizado como inventario donde los servidores objetivo y sus parámetros de conexión fueron especificados.

Las secuencias de flechas en cada caso han sido usadas para indicar el flujo de trabajo donde, primero, el proceso fue iniciado desde subsistema de Linux para Windows, luego la configuración fue dirigida a través de Vagrant, posteriormente el aprovisionamiento fue conducido hacia VirtualBox, y finalmente la configuración fue aplicada en la máquina virtual con AlmaLinux donde el servidor de transferencia de archivos fue establecido.

CAPÍTULO IV

RESULTADOS Y DISCUSIÓN

En esta sección se presentan los datos obtenidos a través del análisis estadístico para mostrar la evidencia que respalda la investigación.

4.1 Resultados

En este estudio, tres enfoques diferentes de aprovisionamiento fueron implementados y analizados para la configuración de un servidor FTP en un entorno virtualizado. Los resultados han sido obtenidos mediante el uso de Shell, Chef y Ansible como herramientas de aprovisionamiento, donde cada método ha presentado sus resultados en términos de tiempo en comparativa a configuraciones manuales.

4.1.1 Análisis descriptivo

Permitió evaluar parámetros como el tiempo de configuración de del servidor, la eficiencia de las herramientas de software libre utilizadas, ayudando a comprender la estructura básica de los datos e identificar patrones [51].

Análisis descriptivo de la variable tiempo respecto al factor estado

Tabla 11. Resumen de procesamiento de casos.

		Casos					
		Válido		**Perdidos**		**Total**	
	Estado	**N**	**Porcentaje**	**N**	**Porcentaje**	**N**	**Porcentaje**
Tiempo (seg)	Antes	245	100 %	0	0 %	245	100 %
	Después	735	100 %	0	0 %	735	100 %

La Tabla 11. Presenta un resumen del procesamiento de casos para las diferentes configuraciones del servidor. En la primera configuración, identificada como (Antes), se tienen 245 datos válidos que representan el 100 % del total. Posteriormente, en la segunda configuración, identificada como (Después), se observa un total de 735 datos válidos. Esta variación se debe a que en la segunda configuración se utilizaron tres tipos diferentes de herramientas de aprovisionamiento (Shell, Chef y Ansible), lo cual permitió recopilar datos adicionales de 245 datos para cada uno de estos tipos, resultando en un total de 735 datos analizados. Es importante destacar que en ambas configuraciones no se registraron datos perdidos. favoreciendo la confiabilidad del análisis realizado.

Tabla 12. Descriptivo de los estados de configuración.

	Estado		Estadístico
Tiempo (segundo)	**Antes**	Media	2095.93
		Desviación estándar	4.229
		Mediana	2096
		Mínimo	2090
		Máximo	2106
		Rango	16
	Después	Media	216.64
		Desviación estándar	110.154
		Mediana	154
		Mínimo	114
		Máximo	394
		Rango	280

La Tabla 12. Proporciona un resumen de la sección (Antes) representa los datos de la configuración manual, sin utilizar estas herramientas automatizadas. El tiempo medio de configuración manual es de 2095.93 segundos (aproximadamente 35 minutos).La desviación estándar de 4.229 segundos indica que los tiempos de configuración manual tienen poca variabilidad. Esto depende de las habilidades y la experiencia del administrador que realiza manualmente cada paso de configuración. La sección (Después) representa los datos de la configuración automatizada utilizando Shell, Chef y Ansible. El tiempo medio de configuración automatizada es de 216.64 segundos (aproximadamente 3.37 minutos), una mejora en comparación con la configuración manual. La desviación estándar de 110.154 segundos muestra una mayor dispersión en los tiempos de configuración automatizada, lo cual es esperable dado que pueden influir factores como el proceso de codificado de los scripts, el rendimiento de la máquina y la disponibilidad de los recursos de la red.

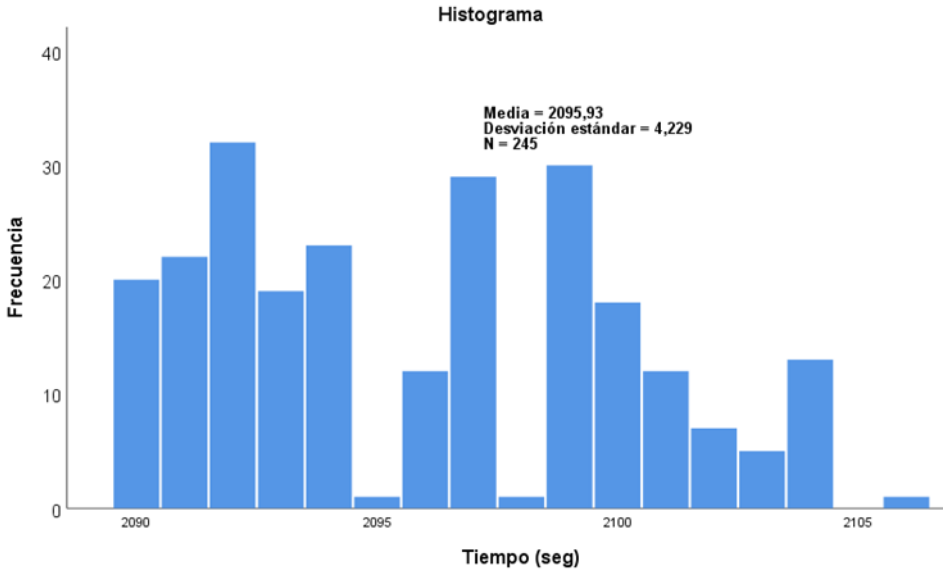

Figura 27. Distribución del tiempo de la configuración (Antes-Manual).

Cada barra corresponde a un rango de tiempo de configuración, y la altura de cada barra indica cuántas veces se observó ese rango de tiempo durante el proceso de configuración del servidor FTP.
La media de los tiempos de configuración es de 2095.93 segundos, con una desviación estándar de 4.229. Esto significa que la mayoría de los tiempos de configuración se agrupan alrededor de los 2095.93 segundos. Los valores más repetidos se concentran entre los 2092, 2097 y 2099 segundos, lo que indica que esos rangos de tiempo fueron los más comunes durante el proceso de configuración. Por otro lado. Los valores más bajos en términos de frecuencia se encuentran a la derecha del histograma, entre los 2100 y 2106 segundos aproximadamente. Aunque son menos frecuentes que los valores alrededor de la media, se manifiesta que es el tiempo cuando recién se empezó con las configuraciones manuales de aprovisionamiento, y los valores más comunes (2090 a 2094 segundos) indican que se obtuvo experiencia en la configuración del servidor por ser los valores más homogéneos.

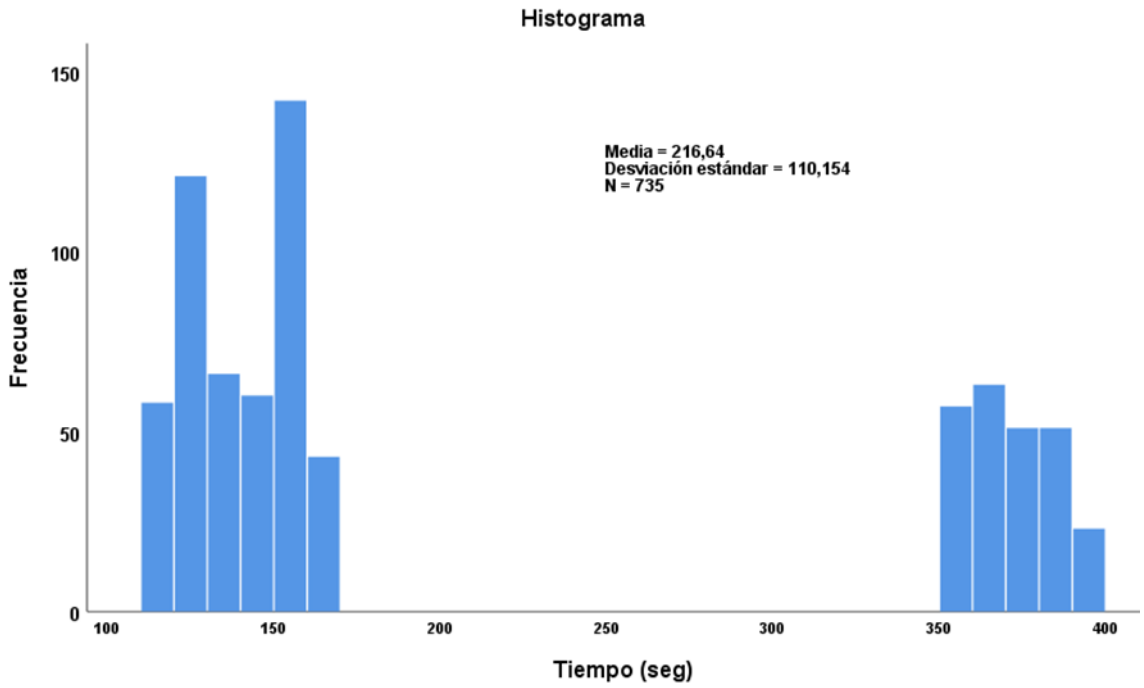

Figura 28. Distribución del tiempo de la configuración (Después).

El proceso de configuración se realizó de manera automatizada ,y la media de los tiempos de configuración es de 216.64 segundos, con una desviación estándar de 110.154.
Los valores frecuentes se encuentran entre los 120 y 160 segundos, lo que sugiere que estos rangos de tiempo fueron los más comunes durante el proceso de configuración automatizado. Por otra parte, el rango de 114 a 163 segundos corresponde a las herramientas Ansible y Shell, y el rango entre 350 a 394 segundos a Chef, además el rango de 164 a 349 segundos no se registró ninguna configuración.

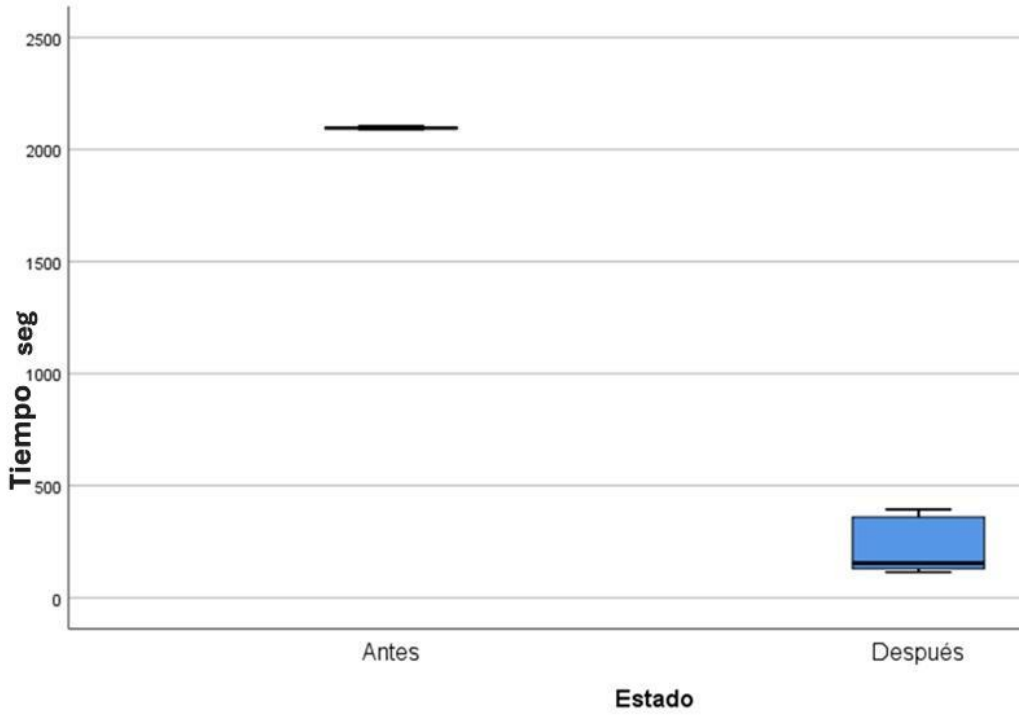

Figura 29. Diagrama de cajas (Estado).

Esta gráfica compara los tiempos de configuración manual y automatizada de un servidor FTP. En la sección (Antes), que representa la configuración manual, los tiempos se concentran alrededor de los 2095 segundos con poca variabilidad, reflejada en una desviación estándar de 4.229 segundos. Por el contrario, la sección (Después), que corresponde a la configuración automatizada con herramientas como Ansible, Shell y Chef, muestra una mayor dispersión con una desviación estándar de 110.154 segundos y un rango de 280 segundos. Esta mayor variabilidad en los tiempos automatizados se debe a que el proceso tiene condiciones cambiantes como la disponibilidad de los recursos de la red y la capacidad de respuesta de la máquina virtual.

Análisis descriptivo de la variable tiempo con respecto al factor método.

Tabla 13. Resumen de procesamiento de casos.

		Casos					
		Válido		Perdidos		Total	
	Método	N	Porcentaje	N	Porcentaje	N	Porcentaje
Tiempo (seg)	Shell	245	100 %	0	0 %	245	100 %
	Chef	245	100 %	0	0 %	245	100 %
	Ansible	245	100 %	0	0 %	245	100 %

Esta tabla resume el procesamiento de casos de una configuración de servidor FTP, mostrando los resultados para diferentes métodos utilizados: Shell, Chef y Ansible.
Donde indica que, en todos los casos, de los 245 datos procesados, el 100 % muestra una tasa de éxito y sin casos perdidos.

Tabla 14. Descriptivo de los métodos de configuración.

	Método		**Estadístico**
Tiempo (Segundos)	**Shell**	Media	153.91
		Desviación estándar	5.026
		Mínimo	145
		Máximo	163
		Rango	18
	Chef	Media	370.95
		Desviación estándar	12.586
		Mínimo	350
		Máximo	394
		Rango	44
	Ansible	Media	125.06
		Desviación estándar	5.959
		Mínimo	114
		Máximo	135
		Rango	21

Los resultados indican que los tiempos medios varían entre los diferentes métodos automatizados, Shell tiene un tiempo medio de 153.91 segundos (2.34 minutos), Chef presenta el tiempo medio más largo con 370.95 segundos (6.11 minutos), mientras que Ansible tiene un tiempo medio de 125.06 segundos (2.11 segundos).

La variabilidad también difiere, con Chef mostrando una desviación estándar más alta de 12.586 segundos, en comparación con Ansible, que tiene 5.959 segundos. Chef utiliza un modelo cliente-servidor, donde el nodo que se está configurando (el cliente) interactúa con el servidor de Chef para obtener y aplicar las configuraciones. Esto requiere tiempos adicionales de comunicación y procesamiento.

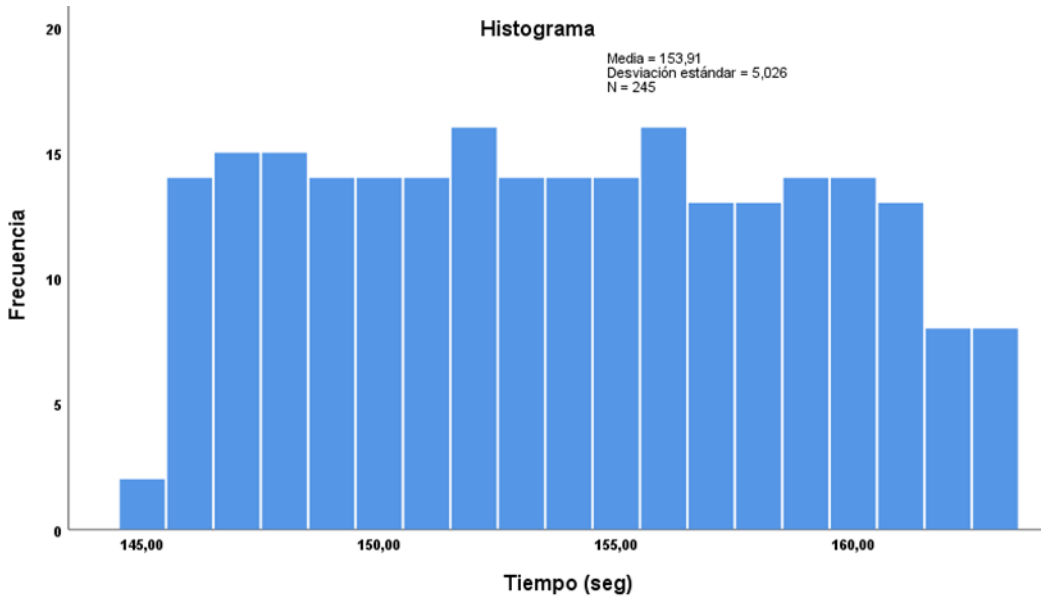

Figura 30. Distribución del tiempo de la configuración (Shell-Script).

La media de los tiempos de configuración es de 153.91 segundos, con una desviación estándar de 5.026 segundos. Esto indica que la mayoría de los tiempos de configuración se agrupan cerca de los 154 segundos y los valores más comunes se concentran entre los 152 y 156 segundos. Por otra parte, los datos analizados presentan una homogénea mayor que la configuración manual. Esto es debido a que, al ser una herramienta de línea de comandos integrada en el núcleo de Linux, Shell puede interactuar directamente con los componentes del sistema, facilitando la ejecución de scripts y comandos de manera fluida.

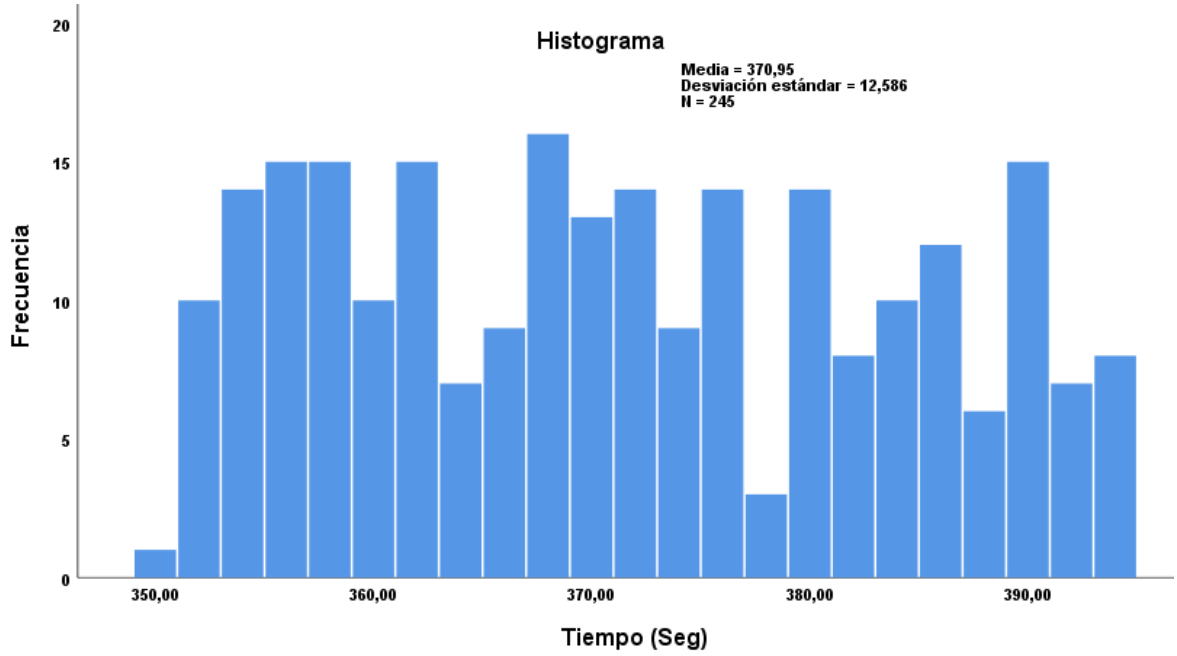

Figura 31. Distribución del tiempo de la configuración (Chef).

A diferencia de la distribución más homogénea observada con el uso de Shell, en este caso se aprecia una mayor variabilidad en los tiempos de configuración con Chef. Los valores comunes se concentran entre los 352 y 378 segundos.

Además, la necesidad de sincronizar y coordinar la comunicación entre el cliente y el servidor Chef puede introducir cierta variabilidad en los tiempos de configuración, en comparación con una herramienta más autosuficiente como Shell.

Chef funciona mediante una arquitectura cliente-servidor, donde el cliente (el nodo que se está configurando) se comunica con el servidor Chef para obtener la configuración y ejecutar las tareas correspondientes. Cuando la comunicación entre el cliente y el servidor Chef es rápida y eficiente, sin problemas de red u otros cuellos de botella, el proceso de configuración se puede completar de manera ágil, resultando en tiempos más cortos, y cuando existen problemas de la conectividad entre el cliente y el servidor Chef, pueden ralentizar la transferencia de datos y la ejecución de las tareas, alargando los tiempos de configuración.

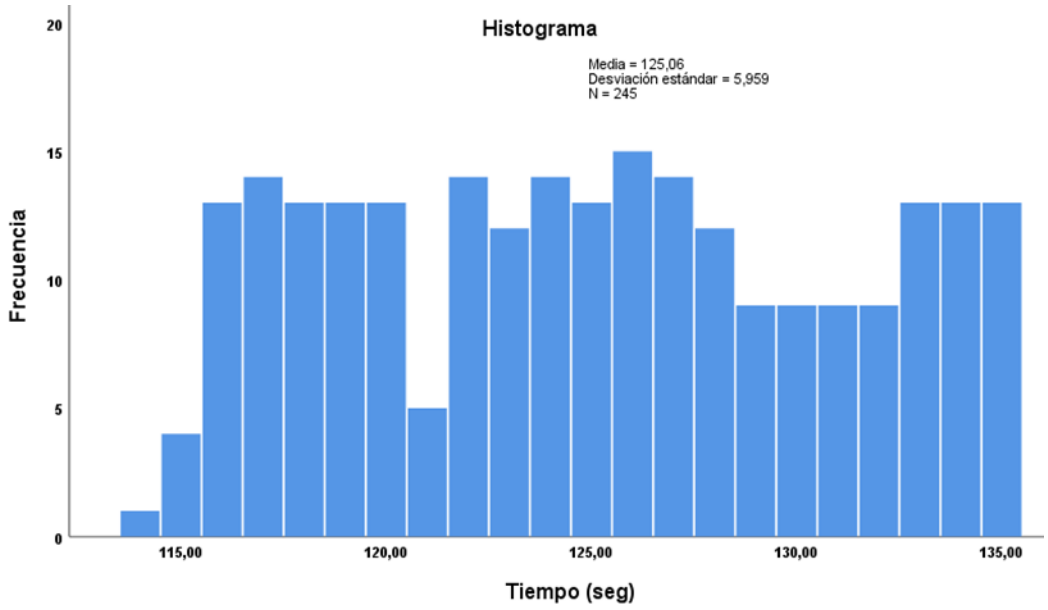

Figura 32. Distribución del tiempo de la configuración (Ansible).

La media está en 125.06 segundos, con una desviación estándar de 5.95 segundos. Esto indica que la mayoría de los tiempos de configuración se agrupan alrededor de los 125 segundos. A diferencia del caso anterior, donde se observó una mayor variabilidad en los tiempos, la distribución de frecuencias se ve más uniforme. Los valores comunes se concentran entre los 122 y 128 segundos; los valores por debajo de los 115 segundos reflejan situaciones en las que Ansible logró optimizar y acelerar el proceso de configuración.

La uniformidad de la distribución observada está relacionada con las características propias de Ansible. A diferencia de las arquitecturas cliente-servidor de herramientas como Chef, Ansible no requiere agentes instalados en los nodos a configurar, lo que puede contribuir a una mayor consistencia en los tiempos de ejecución al no tener que lidiar con problemas de comunicación o sincronización entre componentes.

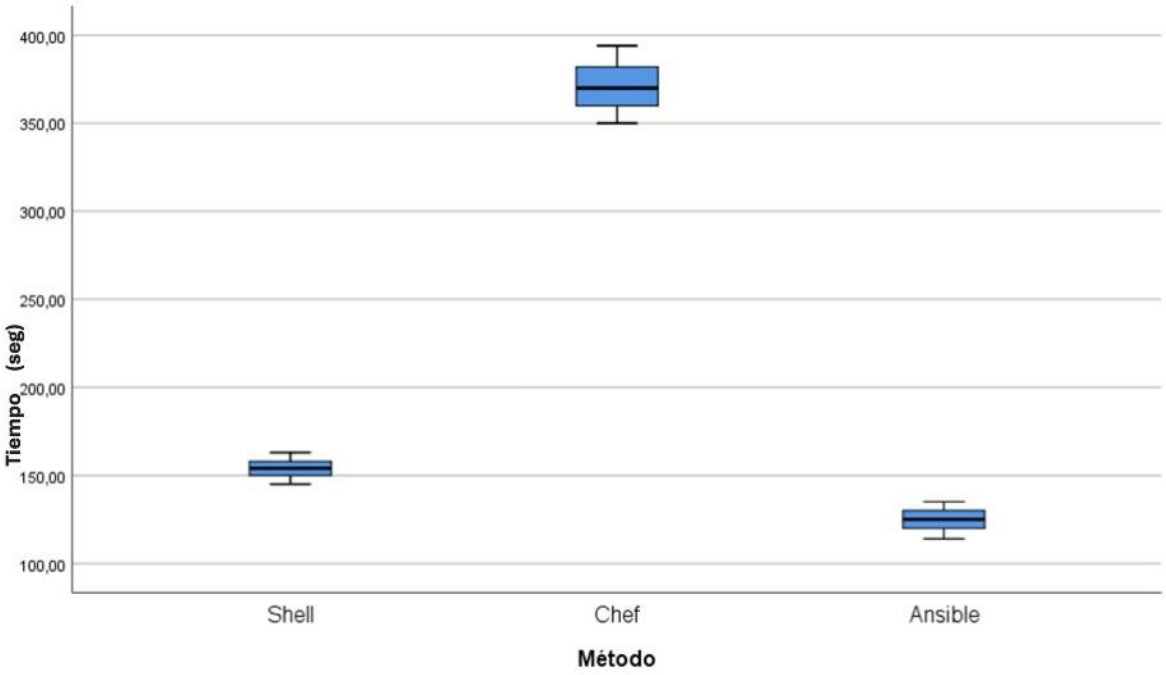

Figura 33. Diagrama de cajas (Método).

Acorde a la Figura 33, se muestran los tiempos de configuración en segundos para los métodos, Shell, Chef y Ansible.

Para el método Shell, se observa una media de 153.91 segundos, con una caja que indica la dispersión o variabilidad de los datos. La caja muestra que la mayoría de los valores se concentran en torno a la mediana, con algunas observaciones más alejadas.

En el caso de Chef, la media es alta, en 370.95 segundos. La caja también indica una mayor dispersión de los tiempos de configuración con esta herramienta. Por último, para Ansible, la media se encuentra en 125.06 segundos, siendo el método más rápido de los tres. La dispersión de los datos también es menor. Ansible, con un enfoque intermedio, logra un equilibrio entre eficiencia y consistencia.

Anova de un factor (Método)

Tabla 15. Anova.

Tiempo					
	Suma de cuadrados	**gl**	**Media cuadrática**	**F**	**Sig.**
Entre grupos	8852787.363	2	4426393.682	60585.855	0
Dentro de grupos	53479.812	732	73.060		0
Total	8906267.176	734			0

Los valores en la tabla muestran los resultados del análisis para los diferentes niveles de los factores (Entre-grupos/Dentro-de-grupos). Esto permite evaluar si existen diferencias significativas en los datos analizados.

La fila (Entre-grupos) muestra un valor de F = 60585.855 y un nivel de significancia (Sig.) de 0, esto indica que hay diferencias significativas entre los grupos comparados, los valores observados en los diferentes grupos son estadísticamente diferentes.
La fila (Dentro-de-grupos) muestra un valor de F = 73.060 y un nivel de significancia (Sig.) también de 0.
Esto significa que también hay diferencias estadísticamente significativas dentro de los grupos (Shell, Chef y Ansible).

Tabla 16. Prueba post hoc.

Comparaciones múltiples						
Variable dependiente: tiempo						
HSD Tukey						
					Intervalo de confianza al 95%	
(I) Método	**(J) Método**	**Diferencia de medias (I-J)**	**Desv. Error**	**Sig.**	**Límite inferior**	**Límite superior**
shell	Chef	-217.03673*	0.7723	0.0	-218.85	-215.22
	Ansible	28.85714*			27.044	30.67
Chef	shell	217.03673*			215.22	218.85
	Ansible	245.89388*			244.08	247.70
Ansible	shell	-28.85714*			-30.67	-27.04
	Chef	-245.89388*			-247.70	-244.08

Para determinar entre qué grupos específicos existen diferencias significativas, se realizó la prueba post-hoc HSD de Tukey para comparar el tiempo de ejecución entre los diferentes métodos. En primer lugar, se compara el método Shell con el método Chef y Ansible mostrando una diferencia de medias que es estadísticamente significativa ($p<0.000$).
Además, se observa un (*) esto permite identificar que existen diferencias significativas en todos los métodos de configuración, esto se contrasta con los intervalos de confianza, donde ambos limites tienen el mismo signo. En la diferencia de medias comparando Ansible con Shell y Chef, se identifica que los valores mostrados tienen el signo menos; esto quiere decir que Ansible es significativamente menor en su tiempo de configuración que los otros dos métodos.

Tabla 17. Subconjuntos homogéneos.

Tiempo (seg)				
HSD Tukey[a]				
		Subconjunto para alfa = 0.05		
Método	**N**	**1**	**2**	**3**
Ansible	245	125.06		
shell	245		153.91	
Chef	245			370.95
Sig.		1.0	1.0	1.0

En la determinación de qué pares de medias de grupos difieren de manera estadísticamente significativa de los tres métodos de configuración se analizaron 245 datos respectivamente donde se obtuvo valores con relación a la media, Ansible con 125.06 segundos (2.11 minutos), Shell con 153.91 segundos (2.34 minutos) y Chef con 370.95 segundos (6.11 minutos), donde el subconjunto par un alfa de 0.05 demuestra que si existe diferencia significativa entre ellos , y Ansible es en método con menor tiempo de configuración.

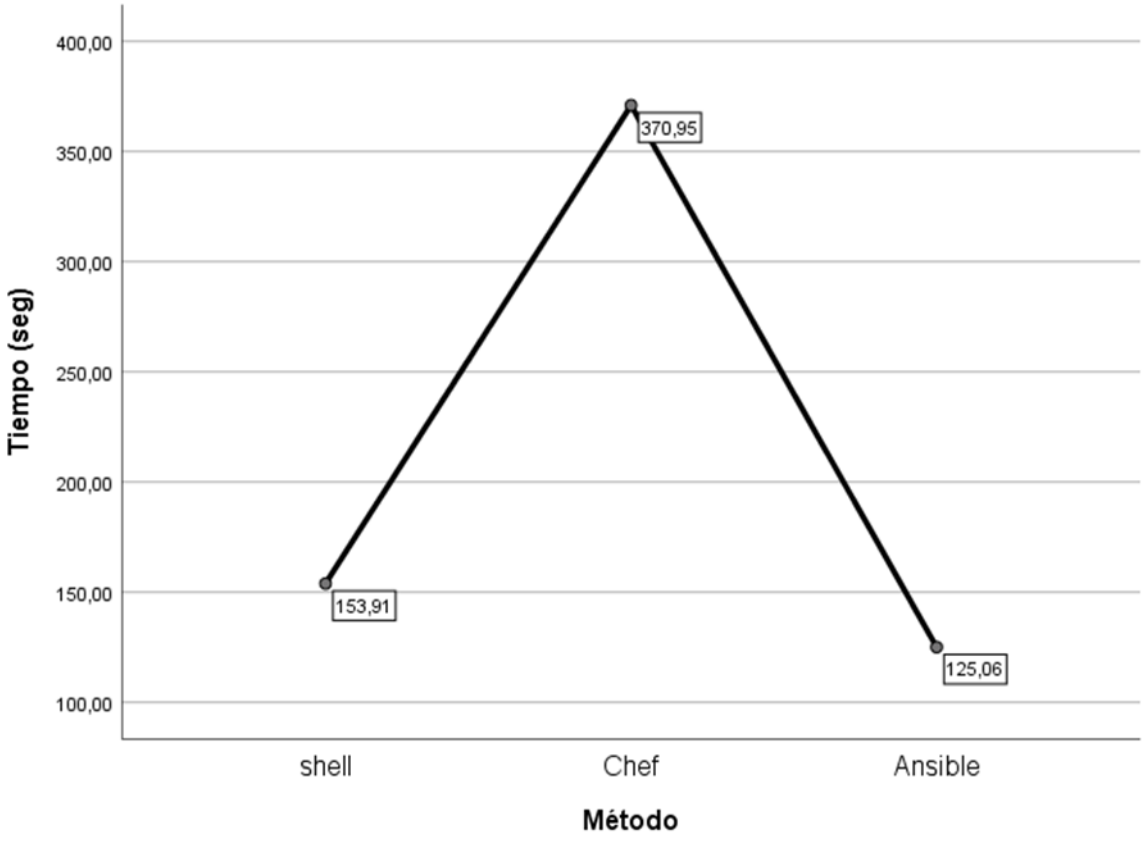

Figura 34. Gráfico de medias.

Como demuestra la Figura 34, la evaluación realizada de los tiempos de ejecución en segundos basados en las medias, si existen diferencias significativas entre los tres métodos comparados. Ansible demostró ser el método más eficiente con un tiempo de ejecución de 125.06 segundos, seguido por Shell-Script que requirió 153.91 segundos; Chef, por su parte, necesitó un tiempo mayor de 370.95 segundos. Esta comparación puso en manifiesto la clara ventaja en el proceso de aprovisionamiento, donde Ansible destacó como la solución más eficiente, siendo aproximadamente 3 veces más rápido que Chef, demostrando que el uso de herramientas de automatización puede resultar mejoras en la productividad y reducción de tiempos de operación.

4.1.2 Prueba de hipótesis

Prueba de normalidad

Esta prueba fue aplicada para evaluar la distribución estadística de los datos recopilados. Este análisis ayudó a definir si los datos siguen una distribución normal, y si no los cumplen, se deben utilizar métodos de análisis no paramétricos para garantizar la validez de los resultados.

Planteamiento de las hipótesis

H_0: Los datos recopilados del tiempo de aprovisionamiento del servidor FTP usando, Shell Script, Chef, Ansible y configuración manual tienen una distribución normal.
H1: Los datos recopilados del tiempo de aprovisionamiento del servidor FTP usando, Shell Script, Chef, Ansible y configuración manual no tienen una distribución normal.
Nivel de significancia (α), 0.05
Confianza =95 %
Si p-valor < α, se rechaza H_0 (los datos posiblemente no son normales).
Lilliefors (Kolmogorov-Smirnov), tamaño de la muestra 245.

Tabla 18. Pruebas de normalidad.

		Kolmogorov-Smirnov[a]		
	Método	**Estadístico**	**gl**	**Sig.**
Tiempo (Seg)	**Shell**	0.084	245	0.0
	Chef	0.088	245	0.0
	Ansible	0.092	245	0.0
	Manual	0.149	245	0.0

Interpretación:

- Se rechazó la hipótesis nula, y se aceptó la hipótesis alternativa. Los datos de las configuraciones utilizadas, Shell Script, Chef, Ansible y la configuración manual no se distribuyen normalmente al 95 % de confianza.
- Todos los p-valores son menores que el nivel de significancia de 0.05, lo que lleva a rechazar la hipótesis nula para todas las configuraciones.
- El método Manual tiene el estadístico más alto (0.149), lo que significa que es el que más se desvía de la normalidad.
- Shell tiene el estadístico más bajo (0.084), indicando que es el que más se acerca a una distribución normal entre las cuatro configuraciones del servidor FTP.

Prueba de Kruskal-Wallis

Comparación de todos los métodos (Manual, Ansible, Chef, Shell-Script).
La prueba de Kruskal-Wallis es una prueba estadística no paramétrica utilizada para comparar tres o más grupos, y permite determinar si existe diferencia significativa entre los grupos que se analizan [52].

Planteamiento de las hipótesis

H_0: No hay diferencia significativa en el tiempo de configuración de un servidor FTP entre el uso de un aprovisionador y configuración manual.
H1: Existe diferencia significativa en el tiempo de configuración de un servidor FTP entre el uso de un aprovisionador y configuración manual.
Si el valor $p < 0,05$, se rechaza H_0, y se concluye que existen diferencias significativas entre los grupos.

Tabla 19. Resumen de hipótesis.

Resumen de contrastes de hipótesis				
	Hipótesis nula	**Prueba**	**Sig.**	**Decisión**
1	La distribución de Tiempo es la misma entre categorías de Método.	Prueba de Kruskal-Wallis para muestras independientes	0.000	Rechace la hipótesis nula.
Se muestran significaciones asintóticas. El nivel de significación es de 0.050.				

Tabla 20. Test no paramétrico de Kruskal-Wallis.

Test Kruskal-Wallis					
Tipos de configuración	**Medianas (segundos)**	**Nivel de significancia**	**Estadístico de prueba**	**P-valor(Sig)**	**Grados de libertad**
Shell Script	154	α=0.05	chi-squared = 918.04	0.00	3
Chef	370				
Ansible	125				
Manual	2096				

Interpretación:

- Las medianas han mostrado que Ansible es el método más rápido.
- El p-valor 0 indica que hay diferencias estadísticamente significativas entre los tipos de configuración; por tanto, se rechazó la hipótesis nula, ya que p-valor < α, y se aceptó la hipótesis alternativa.
- Si existe diferencia significativa en el tiempo de configuración de un servidor FTP entre el uso de un aprovisionador y configuración manual al 95 % de confianza.

En resumen, la prueba Kruskal-Wallis proporcionó evidencia estadística de que existen diferencias significativas en los tiempos de ejecución en los cuatro métodos de configuración, con Ansible en primer lugar, seguido de Shell, Chef y finalmente la configuración manual.

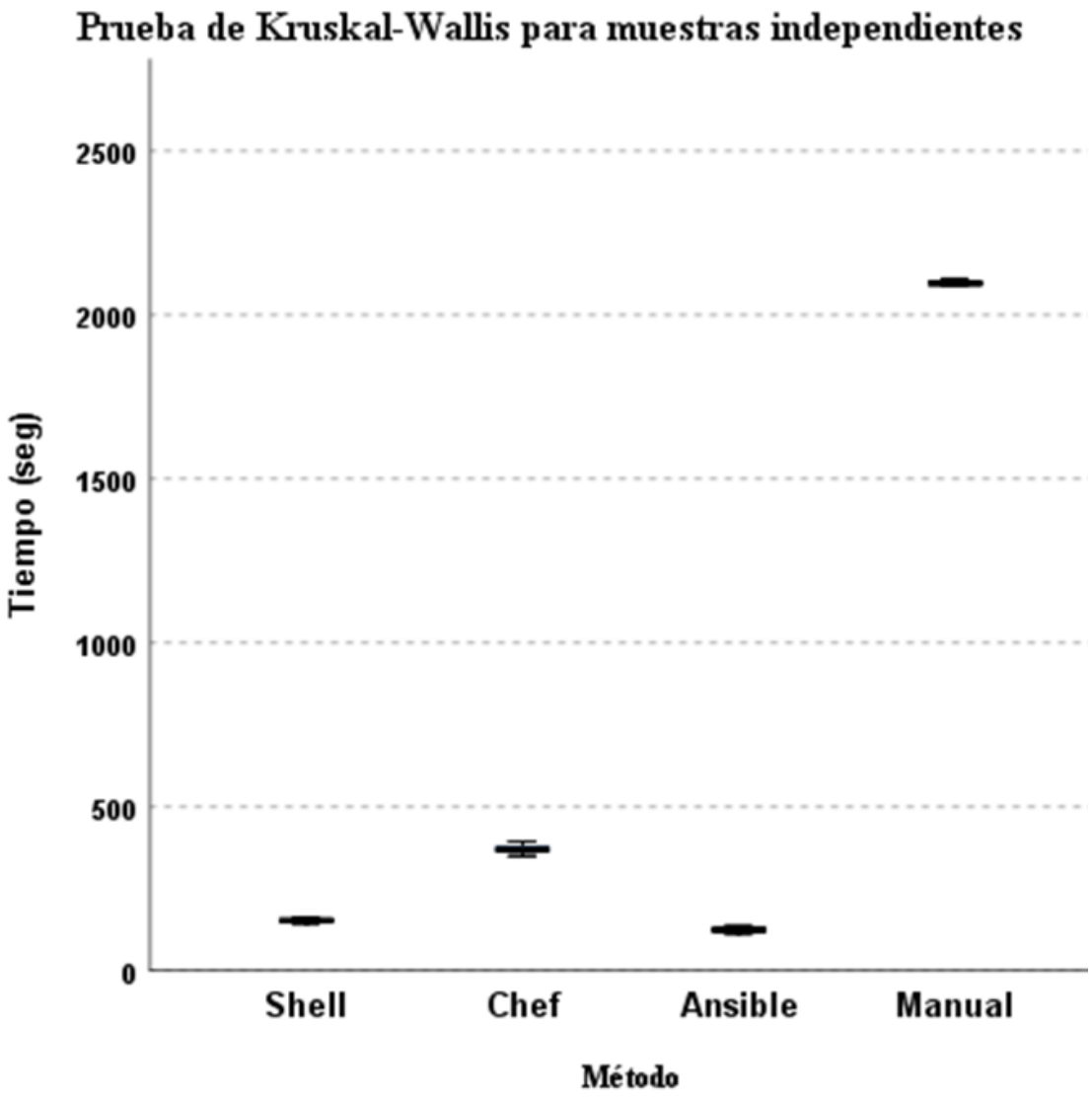

Figura 35. Muestras independientes.

En la comparación de los tiempos de ejecución de diferentes métodos estadísticos utilizados en la prueba de Kruskal-Wallis para muestras independientes, el método (Ansible) destaca por tener los valores de (Tiempo) más bajos, con una mediana de 125, lo que determina que es el método más rápido de los cuatro. En contraste, los métodos (Chef), (Shell y Manual) muestran distribuciones de tiempo más amplias, con medianas (370,154, 2096), lo que implica que estos tardan más tiempo en ejecutar la prueba. El método Ansible es el más recomendable si el objetivo es reducir el tiempo de configuración del servidor.

4.2 Discusión

La presente investigación buscaba un método de configuración de un servidor FTP en una distribución de Linux en una máquina virtual con el propósito de reducir el tiempo de configuración en comparativa con configuraciones manuales, obteniendo tres métodos de configuración, donde se usaron herramientas como Vagrant, Ansible, Shell-Script y Chef.
El estudio sobre el sistema aprovisionador dio los siguientes resultados. El tiempo de configuración con relación a las medianas se redujo de 34.56 minutos a 2.11, 2.34 y 6.11 minutos respectivamente, lo que representa una mejora del 93.89 % usando la herramienta Ansible, 93.22 % con Shell Script y, por último, 82.32 % con Chef. Por otra parte, los sistemas de aprovisionamiento tienen acceso en interfaz de consola. Además, los diferentes métodos de configuración tienen su variabilidad. Shell emplea archivos (sh), Ansible utiliza archivos (yml) y, finalmente Chef usa archivos (rb y erb).

Cada aprovisionador realiza las configuraciones del servidor, levantamiento del servicio, configuración de firewall, creación de usuarios, y al ser empleado Vagrant como gestor de las máquinas virtuales, se crean usuarios automáticamente con el nombre de usuario vagrant y contraseña vagrant. Por tanto, existe un riesgo de vulnerabilidad del sistema y para mitigar este riesgo se cambió la contraseña del usuario por defecto.

Los resultados superan los obtenidos por Pierre Talani, que fueron del 38.8% usando Ansible y Netmiko en la configuración de dispositivos de red (routers) utilizando técnicas de aprovisionamiento similares. Esta diferencia podría atribuirse a nuestro enfoque específico en servidores FTP y la optimización de procesos particulares de este tipo de servidor.
Cabe destacar que la configuración del servidor se realiza mediante SSH; para esta investigación, el aprovisionamiento mediante Ansible es el que realiza la configuración en menor tiempo. Esto se debe a que no utiliza agentes en la configuración, a diferencia de Chef que sí lo requiere.

Por otra parte, el proceso de implementación se fundamentó en un análisis detallado de los requisitos técnicos, como el paquete Vsftpd del servidor utilizado por su compatibilidad y soporte que existe en los sistemas Linux. La ejecución de los scripts automatizados desde la terminal de comandos de Ubuntu garantizó la coherencia de las configuraciones. Así mismo, la aplicación de pruebas de normalidad a los datos recopilados validó la pertinencia de los métodos estadísticos empleados en el análisis.

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

En esta sección se interpretan y sintetizan los resultados, proporcionando una guía para que otros investigadores puedan expandir la investigación en el futuro.

5.1 Conclusiones

En esta tesis se desarrolló un sistema aprovisionador para automatizar la configuración de un servidor FTP en máquinas virtuales. El estudio comparativo se realizó de herramientas de código abierto como Shell-Script, Ansible, Chef, Vagrant y otras. Evaluando sus capacidades de automatización y compatibilidad con servidores FTP, se diseñó un flujo de trabajo de aprovisionamiento, que incluyó la creación de playbooks, scripts y cookbooks. Finalmente, se llevó a cabo una evaluación comparando los tiempos de implementación entre el sistema automatizado y las configuraciones manuales; la validación del sistema fue del tiempo de implementación en escenarios de máquinas virtuales.

El análisis comparativo determinó que la integración de Vagrant y Ansible fueron las herramientas más adecuadas para este proyecto, debido a su arquitectura sin agentes y fácil sintaxis, en comparación con Shell-Script o Chef. Ansible requirió 2.11 minutos, a diferencia de Shell 2.34 minutos y Chef 6.11 en configurar el servidor. Por otra parte, la combinación de estas herramientas facilitó la gestión de configuraciones y permitió el despliegue en entornos virtualizados de las tareas de configuración. El flujo de trabajo diseñado demostró ser eficiente, logrando automatizar completamente el proceso de aprovisionamiento con las tres herramientas seleccionadas. Las pruebas de rendimiento demostraron una mejora en los tiempos de implementación; el sistema automatizado logró reducir el tiempo promedio de configuración de un servidor FTP de 34.56 minutos (configuración manual) a solo 2.11 minutos (Ansible), representando una mejora del 93.89 %.

Además, se han demostrado ventajas en términos de eficiencia y consistencia. Las principales ventajas incluyen la reducción en tiempos de configuración y la eliminación de errores en las configuraciones realizadas por las personas. En comparación con trabajos anteriores, como lo señala Zargham Ahmad, que reportaban reducciones de tiempo del 70 %, el sistema desarrollado logró una mejora del 93.89 %, superando las expectativas iniciales en condiciones similares.
Por otra parte, la principal desventaja identificada es el aprendizaje inicial en el uso de las herramientas de automatización, aunque esto se compensa con la reducción de tiempo ganada a largo plazo, aunque la documentación existente mitiga esta limitación.

El trabajo actual presenta ciertas limitaciones que abren camino a futuras mejoras; el sistema está realizado específicamente para entornos virtualizados con VirtualBox, limitando su aplicación en otros entornos de virtualización. Además, las configuraciones actuales están centradas en implementaciones básicas de un servidor FTP, sin abordar casos de uso más

especializados. Como trabajo futuro, se propone el estudio de compatibilidad con otros servicios y niveles de seguridad.

5.2 Recomendaciones

Basado en la investigación realizada se recomienda continuar con el estudio del aprovisionamiento de servidores y el levantamiento de máquinas virtuales, como la configuración de varias máquinas virtuales con cada herramienta y determinar si existe escalabilidad de servicios como servicios web, correo electrónico, VPN, entre otros, y así determinar si las herramientas utilizas permiten realizar una escalabilidad en los servidores, configuración paralela de servidores empleando cada herramienta, múltiples configuraciones de manera simultánea, explorar la integración de tecnologías con otros hipervisores o dispositivos reales, determinar si dichas configuraciones continúan siendo favorables en términos de tiempos, en servidores y dispositivos de redes.

Por la facilidad de uso, en esta investigación se utilizaron contraseñas simples; aunque cabe considerar que la implementación de medidas de seguridad en los servidores FTP constituye un elemento importante en la infraestructura de cualquier organización que maneje transferencias de archivos. La utilización de claves de acceso seguras representa una capa fundamental de protección. Esta vulnerabilidad inherente hace imperativo establecer contraseñas complejas que incorporen combinaciones de caracteres alfanuméricos y especiales, reduciendo la probabilidad de comprometer el sistema mediante ataques de fuerza bruta o técnicas de diccionario.

Por otra parte, la compatibilidad de versiones entre Vagrant (2.4.1), VirtualBox (7.0), Ansible (2.16.18) y Chef (18.4.12) es importante porque estas herramientas trabajan en conjunto para crear y gestionar entornos de desarrollo virtualizados. Vagrant actúa como orquestador y necesita comunicarse correctamente con VirtualBox (el hipervisor) para crear y administrar las máquinas virtuales, mientras que Ansible y Chef (como herramientas de automatización y configuración) deben poder interactuar con las máquinas virtuales. Si las versiones no son compatibles, pueden surgir problemas como fallos en la creación de máquinas virtuales, errores en la ejecución de scripts de aprovisionamiento, e incluso la imposibilidad de iniciar los entornos virtuales, lo que interrumpiría completamente el flujo de trabajo de desarrollo y pruebas.

En el contexto de la arquitectura del servidor, la configuración de una dirección IP estática es necesaria como un requisito técnico indispensable. Esta configuración garantiza la estabilidad en las comunicaciones, también facilita la gestión de servicios y permite que los clientes siempre puedan encontrar y conectarse al servidor usando la misma dirección.

BIBLIOGRÁFIA

[1] R. D. L. Ignacio, “Implementación automática de servidores DHCP, DNS y correo electrónico en Linux utilizando DevOps para intranets,” 2023.

[2] E. Özdoğan. and O. Ceran, “Systematic Analysis of Infrastructure as Code Technologies,” *Gazi Univ. J. Sci. Part A Eng. Innov.*, vol. 10, no. 4, pp. 452–471, 2023.

[3] D. Parente, P. Drss, J. Clare, and P. M. Loreti, “Master of Science in Computer Science Tear Down the Firewall !,” 2024.

[4] O. Murphy, “Adoption of Infrastructure as Code (IaC) in Real World Lessons and practices from industry,” no. December, p. 61, 2022.

[5] L. Dominguez-Quintero and M. Vargas-Lombardo, “Herramientas de infraestructura como código: Ansible, Terrafom, Chef, Puppet,” *I+D Tecnológico*, vol. 17, no. 2, pp. 25–29, 2021.

[6] J. Sandobalin, E. Insfran, and S. Abrahao, “On the effectiveness of tools to support infrastructure as code: Model-driven versus code-centric,” *IEEE Access*, vol. 8, pp. 17734–17761, 2020.

[7] B. Santoso and M. W. Sari, “Improvement of Setup Time on Server Infrastructure Automation Using Ansible Framework,” *J. Eng. Sci. Technol.*, vol. 17, no. 5, pp. 3660–3671, 2022.

[8] D. Jonathan and N. Coraizaca, “Automatización de redes,” 2023.

[9] H. J. R. Macias, J. F. R. Márquez, C. S. P. Cabrera, J. P. C. Ruperti, and R. S. J. Calero, “Análisis de intrusiones cibernéticas con el uso del Honeypots. Una revisión sistemática,” *Brazilian Appl. Sci. Rev.*, vol. 5, no. 6, pp. 2218–2248, 2021.

[10] S. Morgan, “2022 Official Cybercrime Report,” pp. 1–23, 2022.

[11] Lara, “administración de la infraestructura de TI en una Empresa Constructora en Lima - 2022,” no. 8.5.2017, pp. 2003–2005, 2022.

[12] C. M. Ojeda, J. Lopez, S.-M. Josep, and J. Esteve, “Automatización de Infraestructura IT con IaC,” 2020.

[13] R. Sivakolundhu and D. N. Yagamurthy, “Ansible-Powered Automation for Large-Scale Configuration Management and Rehydration: A Case Study of Enterprise Transformation,” *Artic. Int. J. Sci. Res.*, no. August, 2024.

[14] M. Alander, “Migration from data centre to AWS cloud using packer,” no. November, 2022.

[15] P. Talani and P. Talani, “A Comparison Between Ansible and Netmiko in a Virtualized Environment,” 2024.

[16] G. Gurbatov, “A comparison between Terraform and Ansible on their impact upon the lifecycle and security management for modifiable cloud infrastructures in OpenStack.,” no. May, 2022.

[17] ZarghamAhmad, “Emulation and Detection of Cyber Threat Scenarios,” 2024.

[18] L. Enciso and C. Morales-Iñiguez, “Ansible una estrategia de administración y configuración automatizada sobre GNS3 con OSPF para redes empresariales medianas,” *Rev. Ibérica Sist. e Tecnol. Informação*, pp. 239–252, 2021.

[19] A. Efendi, D. Husna, and I. G. D. Nugraha, “Advancing Network Infrastructure : Integrating VXLAN Technology with Automated Circuit Operations and NOS Configurations,” no. December, pp. 103–124, 2023.

[20] Niko Kalliomaa, “Choosing the Right IaC Tool for Building Reusable Cloud Infrastructure,” 2024.

[21] J. Sandobalin, E. Insfran, and S. Abrahao, “An Infrastructure Modelling Tool for Cloud Provisioning,” *Proc. - 2020 IEEE 14th Int. Conf. Serv. Comput. SCC 2020*, no.

November, pp. 354–361, 2020.
[22] V. P. Belusso, “implantação de nuvem privada,” 2020.
[23] “Implementación De Hardening En Sistemas Operativos De Servidor Implementación.” 2023.
[24] S. Khadafi, Y. D. Pratiwi, and E. Alfianto, “Keamanan Ftp Server Berbasiskan Ids Dan Ips Menggunakan Sistem Operasi Linux Ubuntu,” *Netw. Eng. Res. Oper.*, vol. 6, no. 1, p. 11, 2021.
[25] C. Jelesnianski, M. Ismail, Y. Jang, D. Williams, and C. Min, “Protect the System Call, Protect (Most of) the World with Bastion,” *Int. Conf. Archit. Support Program. Lang. Oper. Syst.*, vol. 3, pp. 528–541, 2023.
[26] K. Henttunen, “Automated hardening of Linux infrastructure,” p. 71, 2023.
[27] A. : Carlos, G. Pinto, M. Ángel, and O. Pérez, “Manuales de usuario para WSL, el subsistema de Windows para Linux,” 2023.
[28] C. Polanco, “Mastering Linux Through Scriptcraft : Scripting , Parallelization , And Virtualization,” no. November, 2023.
[29] G. K. Srivatsa, S. Mukhopadhyay, G. Katrapati, and M. Shrivastava, “A Survey of using Large Language Models for Generating Infrastructure as Code,” *Proc. 20th Int. Conf. Nat. Lang. Process.*, pp. 523–533, 2023.
[30] P. Stockle, B. Grobauer, and A. Pretschner, “Automated Implementation of Windows-related Security-Configuration Guides,” *Proc. - 2020 35th IEEE/ACM Int. Conf. Autom. Softw. Eng. ASE 2020*, pp. 598–610, 2020.
[31] R. Opdebeeck, A. Zerouali, and C. De Roover, “Control and Data Flow in Security Smell Detection for Infrastructure as Code: Is It Worth the Effort?,” *Proc. - 2023 IEEE/ACM 20th Int. Conf. Min. Softw. Repos. MSR 2023*, pp. 534–545, 2023.
[32] A. C. Management, B. Meijer, and L. Hochstein, “Ansible : Up and Running,” 2022.
[33] M. D. Elradi, “Ansible: A Reliable Tool for Automation,” *Electr. Comput. Eng. Stud.*, vol. 2, no. 1, pp. 1–11, 2023.
[34] M. Rastenis, “A study of bugs found in the Ansible configuration management system,” pp. 1–16, 2022.
[35] S. Kokuryo, M. Kondo, and O. Mizuno, “An Empirical Study of Utilization of Imperative Modules in Ansible,” *Proc. - 2020 IEEE 20th Int. Conf. Softw. Qual. Reliab. Secur. QRS 2020*, pp. 442–449, 2020.
[36] E. Luchian, C. Filip, A. B. Rus, I. A. Ivanciu, and V. Dobrota, “Automation of the infrastructure and services for an openstack deployment using chef tool,” *Netw. Educ. Res. RoEduNet Int. Conf. 15th Ed. RoEduNet 2020 - Proc.*, 2020.
[37] O. Kortelainen, “Olli Kortelainen Infrastucture Management In Multicloud Environments,” no. April, 2023.
[38] A. Pessa, “Comparative Study Of Infrastructure As Code Tools,” no. June, 2023.
[39] C. Science and K. G. Srivatsa, “Leveraging Large Language Models for Generating Infrastructure as Code : Open and Closed Source Models and Approaches,” no. June, 2024.
[40] S. Wågbrant and V. D. Radic, “Automated network configuration,” 2022.
[41] B. A. Dapshima and S. K. Ahmad, “Evaluation and Assessment of Software Security Risks and Vulnerabilities Within the Realm of Secure DevOps,” no. July, 2024.
[42] M. Valkinen, “Infrastructure management of multiple cloud platforms,” no. February, 2022.
[43] P. Arviointi, T. K. Systä, and D. Hästbacka, “Ville Penttinen Infrastruktuuri Koodina,” 2021.
[44] S. Achar, “Enterprise SaaS Workloads on New-Generation Infrastructure-as-Code (IaC) on Multi-Cloud Platforms,” *Glob. Discl. Econ. Bus.*, vol. 10, no. 2, pp. 55–74,

2021.
[45] S. R. Goniwada, “Infrastructure Automation,” *Cloud Nativ. Archit. Des.*, no. August, pp. 619–634, 2022.
[46] M. Letemple, G. Gain, S. Ben Mariem, L. Mathy, and B. Donnet, “vTNT: Unikernels for Efficient and Flexible Internet Probing,” *TMA 2024 - Proc. 8th Netw. Traffic Meas. Anal. Conf.*, pp. 1–4, 2024.
[47] M. A. Hernández Mayorga, “Implementación del ambiente de desarrollo de un sistema mediante máquinas virtuales,” *Cuad. Técnicos Univ. la DGTIC*, vol. 1, no. 1, pp. 9–16, 2023.
[48] C. Ramos-Galarza, “Editorial: Diseños de investigación experimental,” *CienciAmérica*, vol. 10, no. 1, pp. 1–7, 2021.
[49] J. I. Martínez-corona and G. E. Palacios-almón, “Desde El Enfoque Investigativo,” *Ra Ximhai*, vol. 19, no. 1, pp. 67–83, 2023.
[50] N. Smyth, *AlmaLinux 9 Essentials*, vol. 6, no. 1. 2023.
[51] H. A. Luzuriaga Jaramillo, C. A. Espinosa Pinos, A. F. Haro Sarango, and H. D. Ortiz Román, “Histograma y distribución normal: Shapiro-Wilk y Kolmogorov Smirnov aplicado en SPSS,” *Rev. Latinoam. Ciencias Soc. y Humanidades*, vol. 4, no. 4, pp. 596–607, 2023.
[52] V. Ruiz, N. Isela, and R. Manjarrez, “Manual de pruebas estadísticas . Cómo aplicarlas en SPSS,” no. April 2024, 2023.

ANEXOS

Anexo1. Códigos extras

En este apartado se encuentran todos los comandos relacionados con la instalación de las herramientas usadas en el aprovisionamiento del servidor.

Tabla 21. Comandos usados en la instalación del subsistema de Linux para Windows.

Comando	**Función**
wsl --install	Instalación del wsl
wsl --uninstall	Desinstalación del wsl
wsl --install -d Ubuntu-22.04	Instalación de Ubuntu
wsl --uninstall -d ubuntu	Desinstalación de Ubuntu
wsl --unregister ubuntu	Elimina completamente ubuntu
wsl.exe --status	Verifica el estado de wsl
wsl --list	Enlista en wsl las instalaciones
wsl --list --online	Verifica la instalación online

Los comandos presentados en la Tabla 21 permiten la instalación, eliminación, verificación de estado del subsistema de Linux mediante el cmd de Windows.

Tabla 22. Comandos de verificación de paquetes en Ubuntu.

Comando	**Función**
python3 --version	Verifica la versión de Python instalada
apt update && apt install traceroute mtr vim wget curl -y	actualizar la lista de paquetes y luego instalar varios paquetes
vagrant plugin install virtualbox_WSL2	Instala un plugin para vagrant Este complemento soluciona problemas relacionados con los comandos vagrant up y vagrant ssh cuando se ejecutan desde ubuntu, mejorando la compatibilidad y estabilidad del uso de vagrant y virtualbox

La Tabla 22 presenta los comandos que se utilizan en ubuntu, y permite la verificación de versión de Python y la instalación de los plugings de vagrant

Tabla 23. Comandos empleados en la instalación de Chef.

Comando	**Función**
sudo apt install wget curl	Instala las dependencias necesarias
curl -L https://omnitruck.chef.io/install.sh | sudo bash	Descarga el script de instalación de Chef
chef-client --version	Verifica la instalación

Estos comandos son necesarios en la instalación de Chef.

Tabla 24. Comandos de instalación de Vagrant y Ansible.

Comando	**Función**
wget -O- https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg	Descarga la clave GPG de HashiCorp (creadores de Vagrant) La decodifica y la guarda en el directorio de claves del sistema
echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(lsb_release -cs) main" | sudo tee /etc/apt/sources.list.d/hashicorp.list	Añade el repositorio de HashiCorp a las fuentes de software del sistema
sudo apt update && sudo apt install vagrant=versión que desea instalar	Actualiza la lista de paquetes Instala Vagrant en la versión especificada
sudo apt install python3-pip	Instala pip, el gestor de paquetes de Python
pip3 install ansible==2.9.27(Version compatible con vagrant)	Instala Ansible versión 2.9.27 usando pip
ansible --version	Verifica la instalación mostrando la versión de Ansible

Tabla 25. Comandos relacionados con el archivo Bashrc.

Comando	**Función**
export VAGRANT_WSL_ENABLE_WINDOWS_ACCESS="1" export PATH="$PATH:/mnt/c/Program Files/Oracle/VirtualBox" export VAGRANT_WSL_WINDOWS_ACCESS_USER_HOME_PATH="/mnt/c/Users/darwi/OneDrive/Escritorio/Proyecto"	Esta configuración es útil cuando se usa vagrant dentro del subsistema de Linux y se necesita acceder a archivos en el sistema de archivos de Windows. Permite a vagrant saber dónde encontrar ciertos archivos o directorios en el sistema Windows desde dentro del subsistema de Linux.
vagrant box list	Este comando te mostrará los boxes que tienes instalados localmente.
source ~/.bashrc	Se utiliza para cargar las configuraciones y variables de entorno definidas en el archivo (.bashrc) en la sesión de terminal actual.

Anexo2. Datos recopilados

En esta sección se encuentran los datos recopilados de cada aprovisionamiento con relación al tamaño de la muestra seleccionada. Los datos están expresados en segundos.

Tabla 26. Registro del tiempo de ejecución del aprovisionador Shell-Script.

#	Seg	#	Seg	#	Seg	#	Seg	#	Seg	#	Seg	#	Seg
1	145	37	152	73	149	109	153	145	160	181	159	217	159
2	158	38	156	74	158	110	147	146	154	182	151	218	151
3	152	39	149	75	153	111	160	147	159	183	146	219	146
4	162	40	158	76	147	112	154	148	151	184	157	220	157
5	149	41	153	77	160	113	159	149	146	185	150	221	150
6	156	42	147	78	154	114	151	150	157	186	163	222	163
7	153	43	160	79	159	115	146	151	150	187	155	223	155
8	147	44	154	80	151	116	157	152	163	188	148	224	148
9	160	45	159	81	146	117	150	153	155	189	161	225	161
10	154	46	151	82	157	118	163	154	148	190	152	226	152
11	159	47	146	83	150	119	155	155	161	191	156	227	156
12	151	48	157	84	163	120	148	156	152	192	152	228	145
13	146	49	150	85	155	121	161	157	156	193	156	229	148
14	157	50	163	86	148	122	152	158	149	194	149	230	152
15	150	51	155	87	161	123	156	159	158	195	158	231	162
16	163	52	148	88	152	124	149	160	153	196	153	232	149
17	155	53	161	89	156	125	158	161	147	197	147	233	156
18	148	54	152	90	149	126	153	162	160	198	160	234	153
19	161	55	156	91	158	127	147	163	154	199	154	235	147
20	152	56	149	92	153	128	160	164	159	200	159	236	160
21	156	57	158	93	147	129	154	165	151	201	151	237	154
22	149	58	153	94	160	130	159	166	146	202	146	238	159
23	158	59	147	95	154	131	151	167	157	203	147	239	151
24	153	60	160	96	159	132	146	168	150	204	150	240	146
25	147	61	154	97	151	133	157	169	162	205	162	241	157
26	160	62	159	98	146	134	150	170	155	206	155	242	150
27	154	63	151	99	157	135	162	171	148	207	148	243	163
28	159	64	146	100	150	136	155	172	161	208	161	244	155
29	151	65	157	101	162	137	148	173	152	209	152	245	148
30	146	66	150	102	155	138	161	174	156	210	156		
31	157	67	162	103	148	139	152	175	149	211	149		
32	150	68	155	104	161	140	156	176	158	212	158		
33	162	69	148	105	152	141	149	177	153	213	153		
34	155	70	161	106	156	142	158	178	147	214	147		
35	148	71	152	107	149	143	153	179	160	215	160		
36	161	72	156	108	158	144	147	180	154	216	154		

Tabla 27. Registro del tiempo de ejecución del aprovisionador Chef.

#	Seg	#	Seg	#	Seg	#	Seg	#	Seg	#	Seg	#	Seg
1	367	37	354	73	379	109	356	145	364	181	375	217	355
2	355	38	379	74	364	110	382	146	371	182	361	218	390
3	378	39	364	75	387	111	368	147	356	183	386	219	367
4	362	40	387	76	371	112	360	148	382	184	372	220	353
5	351	41	371	77	356	113	393	149	368	185	355	221	380
6	384	42	357	78	382	114	375	150	360	186	390	222	369
7	373	43	392	79	368	115	361	151	393	187	367	223	358
8	359	44	366	80	360	116	386	152	375	188	353	224	384
9	390	45	352	81	393	117	372	153	361	189	380	225	367
10	366	46	381	82	375	118	355	154	386	190	369	226	355
11	353	47	368	83	361	119	390	155	372	191	358	227	378
12	380	48	360	84	386	120	367	156	355	192	384	228	362
13	369	49	394	85	372	121	353	157	390	193	373	229	351
14	357	50	375	86	355	122	380	158	367	194	362	230	384
15	386	51	361	87	390	123	369	159	353	195	389	231	373
16	372	52	386	88	367	124	358	160	380	196	376	232	359
17	360	53	372	89	353	125	384	161	369	197	351	233	390
18	393	54	355	90	380	126	373	162	358	198	385	234	366
19	368	55	390	91	369	127	362	163	384	199	370	235	353
20	356	56	367	92	358	128	389	164	373	200	357	236	380
21	382	57	353	93	384	129	376	165	362	201	391	237	369
22	375	58	380	94	373	130	351	166	389	202	366	238	357
23	361	59	369	95	362	131	358	167	376	203	354	239	386
24	388	60	358	96	389	132	384	168	351	204	379	240	372
25	370	61	384	97	376	133	373	169	391	205	364	241	360
26	352	62	373	98	351	134	362	170	366	206	387	242	393
27	377	63	362	99	385	135	389	171	354	207	371	243	368
28	363	64	389	100	370	136	376	172	379	208	356	244	356
29	389	65	376	101	357	137	351	173	364	209	382	245	382
30	374	66	351	102	391	138	385	174	387	210	368		
31	358	67	385	103	366	139	370	175	371	211	360		
32	383	68	370	104	354	140	357	176	356	212	393		
33	365	69	357	105	379	141	391	177	382	213	375		
34	350	70	391	106	364	142	366	178	368	214	361		
35	391	71	366	107	387	143	354	179	360	215	386		
36	376	72	354	108	371	144	379	180	393	216	372		

Tabla 28. Registro del tiempo de ejecución del aprovisionador Ansible.

#	Seg	#	Seg	#	Seg	#	Seg	#	Seg	#	Seg	#	Seg
1	124	37	134	73	125	109	133	145	117	181	122	217	120
2	118	38	123	74	129	110	126	146	133	182	127	218	134
3	129	39	126	75	117	111	128	147	122	183	120	219	118
4	126	40	115	76	134	112	117	148	127	184	134	220	125
5	122	41	130	77	123	113	134	149	120	185	118	221	131
6	131	42	121	78	126	114	122	150	134	186	125	222	116
7	117	43	125	79	131	115	127	151	118	187	131	223	135
8	133	44	133	80	119	116	120	152	125	188	116	224	123
9	120	45	118	81	124	117	132	153	131	189	135	225	126
10	125	46	124	82	128	118	118	154	116	190	123	226	129
11	128	47	129	83	115	119	125	155	135	191	126	227	119
12	114	48	116	84	133	120	129	156	123	192	129	228	124
13	132	49	135	85	122	121	116	157	126	193	119	229	128
14	119	50	122	86	127	122	126	158	129	194	124	230	117
15	127	51	127	87	118	123	135	159	119	195	128	231	133
16	123	52	120	88	135	124	123	160	124	196	117	232	122
17	130	53	132	89	121	125	126	161	128	197	133	233	127
18	116	54	117	90	130	126	131	162	117	198	122	234	120
19	134	55	126	91	116	127	119	163	133	199	127	235	134
20	121	56	131	92	132	128	124	164	122	200	120	236	118
21	126	57	119	93	125	129	128	165	127	201	134	237	125
22	129	58	124	94	129	130	117	166	120	202	118	238	130
23	115	59	134	95	120	131	133	167	134	203	125	239	116
24	133	60	121	96	134	132	122	168	118	204	130	240	135
25	118	61	128	97	123	133	127	169	125	205	116	241	123
26	124	62	115	98	126	134	120	170	130	206	135	242	126
27	131	63	133	99	117	135	134	171	116	207	123	243	132
28	120	64	122	100	131	136	118	172	135	208	126	244	119
29	127	65	127	101	122	137	125	173	123	209	132	245	117
30	122	66	118	102	127	138	130	174	126	210	119		
31	135	67	135	103	119	139	116	175	132	211	124		
32	117	68	124	104	135	140	135	176	119	212	128		
33	132	69	130	105	124	141	123	177	124	213	117		
34	125	70	116	106	130	142	119	178	128	214	133		
35	128	71	132	107	116	143	124	179	117	215	122		
36	119	72	120	108	121	144	128	180	133	216	127		

Tabla 29. Registro del tiempo en la configuración Manual.

#	Seg	#	Seg	#	Seg	#	Seg	#	Seg	#	Seg	#	Seg
1	2100	37	2095	73	2099	109	2099	145	2091	181	2106	217	2100
2	2099	38	2090	74	2094	110	2104	146	2100	182	2092	218	2092
3	2096	39	2093	75	2093	111	2103	147	2096	183	2101	219	2091
4	2100	40	2097	76	2097	112	2097	148	2100	184	2099	220	2101
5	2094	41	2094	77	2099	113	2099	149	2092	185	2091	221	2099
6	2099	42	2099	78	2094	114	2094	150	2101	186	2100	222	2094
7	2092	43	2091	79	2092	115	2092	151	2094	187	2096	223	2093
8	2096	44	2092	80	2090	116	2090	152	2101	188	2100	224	2097
9	2093	45	2096	81	2092	117	2092	153	2104	189	2092	225	2100
10	2090	46	2097	82	2093	118	2093	154	2101	190	2091	226	2104
11	2092	47	2090	83	2090	119	2100	155	2094	191	2102	227	2092
12	2091	48	2099	84	2092	120	2102	156	2099	192	2091	228	2099
13	2097	49	2091	85	2091	121	2101	157	2102	193	2099	229	2097
14	2094	50	2090	86	2091	122	2101	158	2096	194	2104	230	2099
15	2092	51	2096	87	2099	123	2099	159	2103	195	2103	231	2097
16	2091	52	2090	88	2094	124	2104	160	2100	196	2097	232	2093
17	2099	53	2092	89	2093	125	2093	161	2092	197	2090	233	2101
18	2091	54	2091	90	2097	126	2097	162	2101	198	2104	234	2096
19	2090	55	2094	91	2090	127	2090	163	2097	199	2092	235	2094
20	2096	56	2091	92	2093	128	2094	164	2104	200	2099	236	2093
21	2090	57	2094	93	2092	129	2104	165	2092	201	2097	237	2100
22	2092	58	2093	94	2090	130	2103	166	2101	202	2104	238	2101
23	2091	59	2097	95	2092	131	2102	167	2099	203	2093	239	2100
24	2091	60	2099	96	2091	132	2091	168	2091	204	2097	240	2096
25	2099	61	2094	97	2091	133	2092	169	2090	205	2102	241	2090
26	2094	62	2092	98	2099	134	2097	170	2096	206	2091	242	2102
27	2093	63	2090	99	2094	135	2094	171	2100	207	2101	243	2096
28	2097	64	2092	100	2093	136	2093	172	2092	208	2099	244	2100
29	2090	65	2094	101	2097	137	2097	173	2091	209	2104	245	2092
30	2094	66	2093	102	2090	138	2100	174	2103	210	2093		
31	2092	67	2097	103	2094	139	2094	175	2097	211	2097		
32	2099	68	2090	104	2092	140	2092	176	2100	212	2100		
33	2097	69	2094	105	2099	141	2104	177	2104	213	2104		
34	2099	70	2092	106	2097	142	2102	178	2092	214	2092		
35	2097	71	2099	107	2099	143	2097	179	2099	215	2099		
36	2093	72	2097	108	2097	144	2098	180	2097	216	2100		

Anexo3. Tiempo de ejecución en la configuración

En esta sección se observa el tiempo de ejecución de cada aprovisionamiento y las máquinas virtuales configuradas.

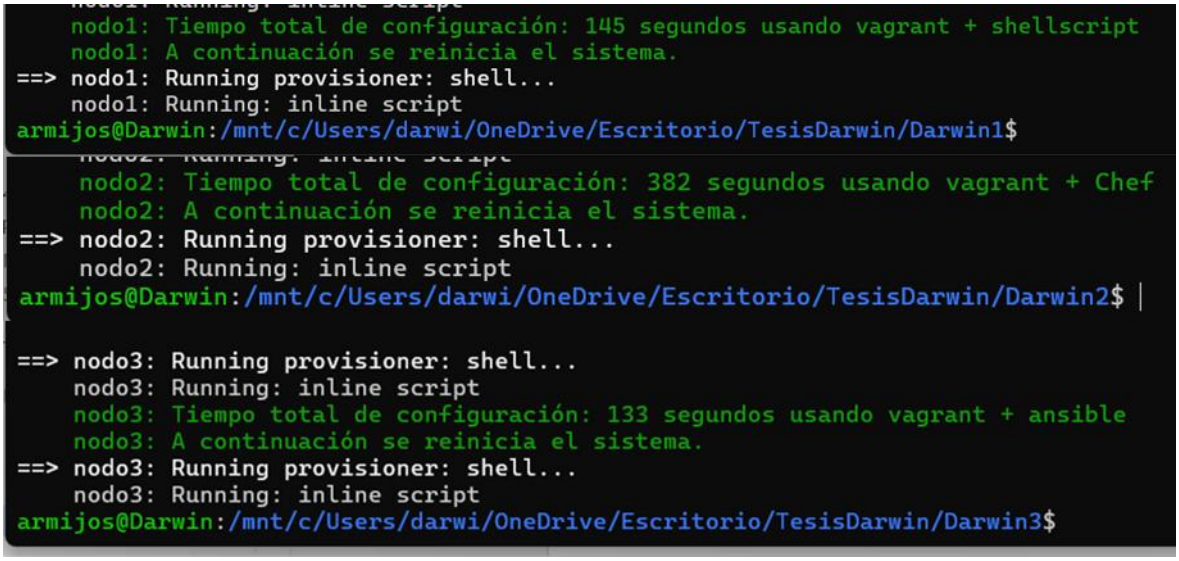

Figura 36. Tiempos empleados en las configuraciones.

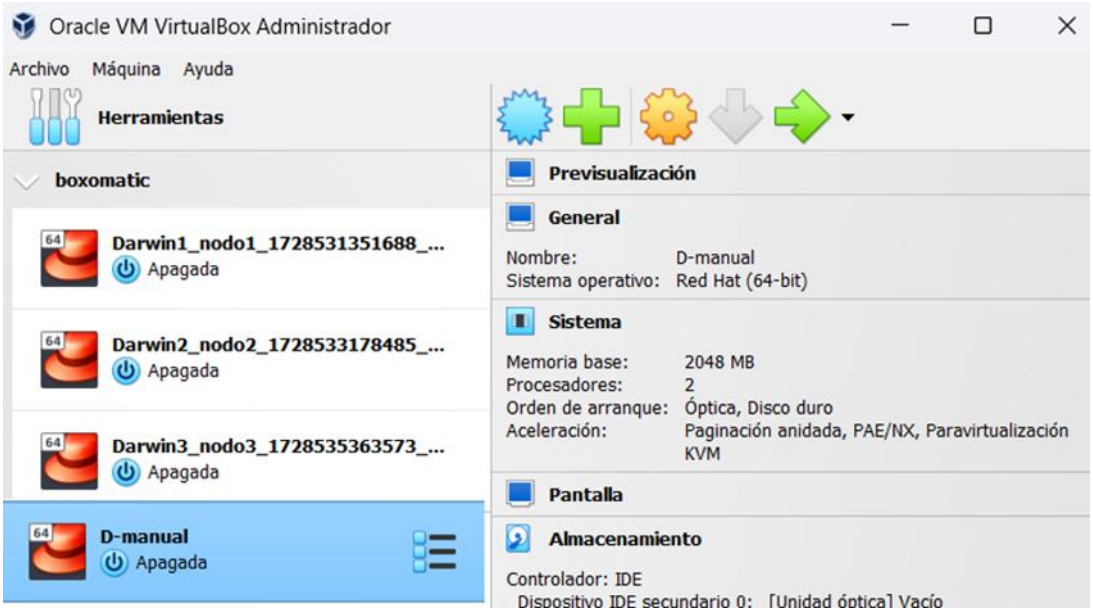

Figura 37. Servidores virtuales configurados.