# ESCUELA POLITÉCNICA NACIONAL

## ESCUELA DE FORMACIÓN DE TECNÓLOGOS

### IMPLEMENTACION DE CONTENEDORES CON SERVIDORES MEDIANTE DEVOPS

### IMPLEMENTACIÓN DE SERVIDORES MEDIANTE *PODMAN*

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

**ANDRÉS ISAAC GUALOTUÑA LLUMIQUINGA**

**DIRECTOR: ING. JAVIER ALEJANDRO ARMAS NAVARRETE**

**DMQ, septiembre 2022**

# CERTIFICACIONES

Yo, ANDRÉS ISAAC GUALOTUÑA LLUMIQUINGA declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

**ANDRÉS ISAAC GUALOTUÑA LLUMIQUINGA**

**andres.gualotuna@epn.edu.ec**

**andresisaac5@gmail.com**

Certifico que el presente trabajo de integración curricular fue desarrollado por ANDRÉS ISAAC GUALOTUÑA LLUMIQUINGA, bajo mi supervisión.

**ING. JAVIER ALEJANDRO ARMAS NAVARRETE**

**DIRECTOR**

**javier.armas@epn.edu.ec**

# DECLARACIÓN DE AUTORÍA

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

ANDRÉS ISAAC GUALOTUÑA LLUMIQUINGA

# DEDICATORIA

A mis padres Rosa y Patricio por siempre brindarme la guía, la paciencia, el apoyo emocional y económico necesario, además de sus valiosas enseñanzas que han inculcado en mí los valores de la perseverancia, el esfuerzo y el valor del trabajo duro para lograr esta gran meta en mi vida.

A mis hermanos Beatriz y Nicolas por acompañarme durante todo este viaje y brindarme su cariño incondicional.

A todos mis amigos, por acompañarme y motivarme a seguir mis metas personales y profesionales.

# AGRADECIMIENTO

Un agradecimiento especial a todos mis docentes a lo largo de la carrera de Redes y Telecomunicaciones por bridarme sus valiosos conocimientos con paciencia y claridad.

A las autoridades de la Escuela de Formación de Tecnólogos que siempre brindan la ayuda necesaria a los estudiantes durante todo el proceso educativo.

# ÍNDICE DE CONTENIDOS

# RESUMEN

El presente trabajo de titulación tiene como finalidad la implementación, configuración y despliegue de servidores DHCP, DNS y WEB montados dentro de contenedores, usando la herramienta DevOps de creación y administración de contenedores *Podman*. Por lo que se hará uso de máquinas virtuales creadas en VirtualBox que están ejecutando Ubuntu.

En la primera sección se analizan los conceptos más importantes acerca del uso de contenedores y de la herramienta *Podman*, así como sus ventajas y desventajas además de una rápida comparación con la popular herramienta para administrar contenedores *Docker*. También se tratan las definiciones de servidores DNS, DHCP y WEB y sus características más importantes.

En la segunda parte del documento se explica detalladamente el proceso para la implementación de cada uno de los contenedores con su respectivo servicio y las herramientas que se emplearon.

En la tercera sección se realiza el proceso de implementación de cada uno de los servicios, se muestran las configuraciones necesarias, se verifica su funcionalidad y su comportamiento en el esquema cliente-servidor.

Finalmente, en la cuarta sección se realiza pruebas de funcionamiento con diferentes niveles de conexión entre el servidor y los clientes internos o externos. Se verifica su funcionalidad y aplicabilidad práctica dentro de un entorno controlado.

**PALABRAS CLAVE:** Contenedor, *Podman*, DHCP, DNS, WEB.

# *ABSTRACT*

*The purpose of this degree work is the implementation, configuration and deployment of DHCP, DNS and WEB servers mounted inside containers, using the DevOps tool for creating and managing containers named Podman. Therefore, virtual machines created in VirtualBox and running Ubuntu will be used.*

*The first section discusses the most important concepts about using containers and the Podman tool, as well as its advantages and disadvantages, plus a quick comparison with the popular Docker container management tool. The definitions of DNS, DHCP and WEB servers and their most important characteristics are also covered.*

*The second part of the document explains in detail the process for the implementation of each of the containers with their respective service and the necessary tools that were used.*

*In the third section, the implementation process of each of the services is carried out, the necessary configurations are shown, their functionality and their behavior in the client-server scheme are verified.*

*Finally, in the fourth section, performance tests are carried out with different levels of connection between the server and internal or external clients. Its functionality and practical applicability are verified within a controlled environment.*

***KEY WORDS:*** *Container, Podman, DHCP, DNS, WEB.*

# 1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

El presente proyecto de titulación permite mostrar las capacidades y la flexibilidad que ofrecen los contenedores para trabajar en un entorno de DevOps usando un motor de contenedores relativamente nuevo llamado *Podman*. Que se presenta como una alternativa a *Docker*.

Usando *Podman* se procede a crear contenedores, a los cuales se les asigna las características necesarias para el despliegue de servidores y su respectiva comunicación con los clientes con el fin de obtener servicios fácilmente manejables y con un uso eficiente de recursos de *software*.

Los servicios a ser desplegados en los contenedores son DNS, DHCP y *WEB*. Para cada uno de ellos se procede a mostrar las configuraciones necesarias en cada caso y demostrar su funcionamiento tanto a nivel de la maquina *host* en los que son creados y con máquinas clientes externas.

## 1.1 Objetivo general

Implementar servidores con herramientas de DevOps.

## 1.2 Objetivos específicos

- Analizar el contenedor *Podman* y sus características.
- Diseñar tres contenedores con DNS, DHCP y WEB.
- Implementar los tres contenedores diseñados.
- Verificar el funcionamiento de los contenedores.

## 1.3 Alcance

El presente trabajo de investigación permite conocer y aprender el manejo de contenedores mediante DevOps. Esto es importante debido a que en la actualidad es una herramienta usada en gran parte de las implementaciones de desarrollo. En tal sentido este proyecto se plantea ejecutar un total de tres servidores, los cuales son: DNS, DHCP y WEB, a estos tres servidores se los pondrá en marcha en una máquina virtual para de esta manera observar su funcionamiento.

## 1.4 Marco Teórico

### *VirtualBox*

Es un programa dedicado a la virtualización del tipo x86 para las tecnologías de Intel64 y AMD64, de uso doméstico y empresarial. La principal ventaja con la cuenta *VirtualBox* es que se trata de un programa de código abierto regido bajo las condiciones de Licencia Pública General (GNU)[1].

Tiene compatibilidad con diversos sistemas operativos entre los que se encuentran *Macintosh*, *Windows*, *Linux* y *Solaris*, además, soporta gran cantidad de sistemas operativos invitados como *Windows* desde su versión NT 4.0 hasta *Windows 10*, *DOS*, *Linux* y sus diversas distros, *Solaris*, *OpenSolaris*, *OpenBSD*, *OS/2*, entre otros[1].

Al ser de código abierto recibe gran aportación de la comunidad, pero con el respaldo de una empresa dedicada (*Oracle*), como resultado se obtiene actualizaciones constantes y soporte para cada vez más sistemas operativos invitados.

### Contenedor Linux

Los contenedores de Linux son herramientas de *software* que posibilitan aislar y empaquetar aplicaciones incluyendo su código y todo su entorno de ejecución, es decir, junto con todos los archivos necesarios para su puesta en marcha [2]. Esto permite una facilidad de movimiento de la aplicación entre los diferentes en entornos de desarrollo y operaciones sin que pierda ninguna de sus características.

La ventaja de manejar contenedores es que su estructura está aislada de su entorno lo que dinamiza el desarrollo de aplicaciones, debido a que permite una mejor portabilidad, aislamiento y configuración de los programas, es decir, el contenedor corre siempre de la misma forma independientemente de la infraestructura[3], [4].

El funcionamiento de los contenedores suele comparase con el uso de máquinas virtuales, aunque tienen algo en común, su funcionamiento es totalmente diferente. Mientras que una máquina Virtual necesita como base una infraestructura, es controlada por un Hipervisor y necesita de un sistema operativo invitado para poder desarrollar sus tareas. Los contenedores Necesitan únicamente de una infraestructura y un sistema operativo anfitrión donde el contenedor puede desplegarse[5]. La comparación puede ser observada de manera más clara la Figura 1.1.

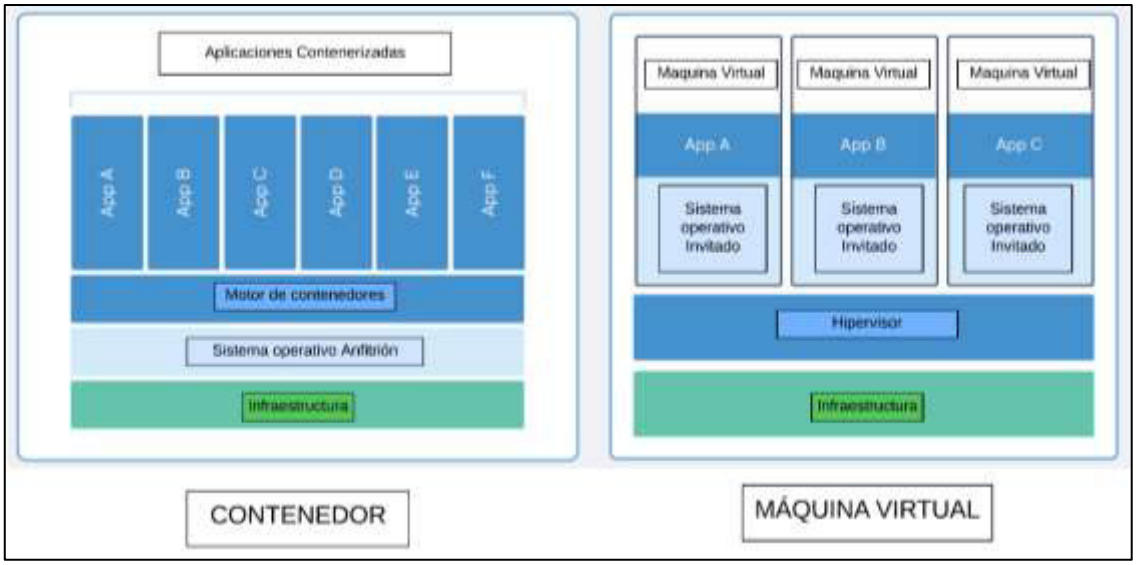


**Figura 1.1** Estructura de contenedores vs. Máquinas virtuales[5]

### *Podman*

Es un motor de contenedores que se utiliza para administrar, desarrollar y ejecutar contenedores e imágenes OCI (*Open Container Initiative*) en un sistema Linux. A diferencia de otros motores de contenedores como *Docker*, *Podman* puede ser ejecutado desde la raíz del sistema o desde un usuario sin privilegios [6].

*Libpod* es la biblioteca de *Podman* que administra todo el entorno de los contenedores, en la cual se administra todo el ecosistema. Esto incluye los *pods*, volúmenes e imágenes de contenedores [7].

*Podman* está especializado en todas la funciones y comandos, tales como, etiquetar y extraer, que ayudan a modificar y mantener imágenes de contenedores OCI, además, permite ejecutar los contenedores creados a partir de imágenes dentro de un entorno de producción [7].

El servicio *Podman* se ejecuta únicamente en plataformas Linux, aunque dentro de las plataformas MAC y Windows se lo puede ejecutar con la ayuda de un cliente *API REST* (Interfaz de programación de aplicaciones) mediante protocolo de cápsula segura (SSH) en una máquina virtual de Linux [6].

### *POD*

El termino *pod* fue en primera instancia establecido por Kubernetes y se define como un grupo de uno o varios contenedores con recursos como almacenamiento y red compartidos, además, de especificaciones para ejecutar los contenedores. La necesidad de agrupar contenedores en *pods* se da para aumentar la eficiencia al momento de compartir recursos, de tal manera que, se pueden combinar gran variedad de aplicaciones virtualizadas [8].

Un *pod* tiene como núcleo principal "contenedores infra" que son responsables de gestionar y asegurar recursos como CPU, puertos de red, memoria, espacios de nombre, entre otros [9]. En la Figura 1.2 se muestra la estructura de un *pod*.

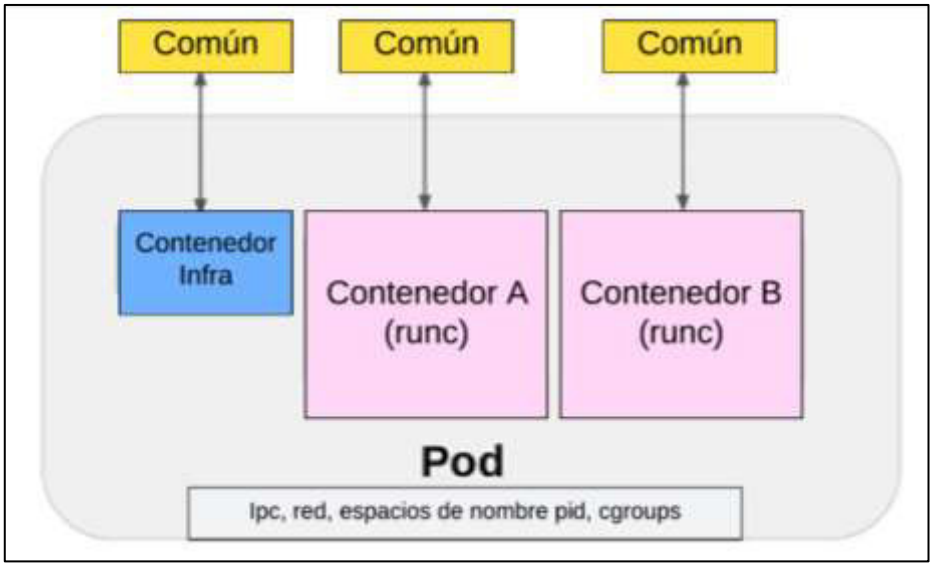


**Figura 1.2** Esquema de estructura de un *pod*[9]

### Servidor DHCP

DHCP es el protocolo de configuración dinámica de host, está basado en el modelo cliente-servidor, en donde, un dispositivo fijo o móvil que se conecta a una red hace una solicitud al servidor de una dirección IP por los puertos UDP 67 y 68 en el caso de IPv4 y los puertos 546 y 547 en el caso de IPv6, el servidor DHCP consulta parámetros de red libres en su base de datos antes de asignar direcciones IP y una vez los parámetros son confirmados se los envía al cliente. Los parámetros que recibe el cliente son puertas de enlace predeterminadas, dirección IP única, servidores DNS, mascara de subred, entre otros[10].

El proceso automático de asignación consiste en 4 etapas.

- **Broadcast o difusión amplia:** El cliente envía un paquete de descubrimiento desde la dirección 0.0.0.0 a la dirección de broadcast 255.255.255.255 para entablar comunicación con todos los entes de la red con el objetivo de localizar al servidor DHCP[11].
- **Oferta:** El servidor DHCP de la red escucha las peticiones por el puerto 67 y cuando detecta una, envía un paquete de ofrecimiento que contienen la dirección MAC del cliente, una dirección IP libre, la máscara de subred, el ID y dirección IP del servidor[11].
- **Solicitud:** El cliente recibe el paquete con los parámetros asignados, y realiza una confirmación[11].
- **Confirmación:** El servidor recibe la confirmación del cliente establece los parámetros TCP/IP para el mismo y guarda los datos la dirección MAC del cliente junto con sus parámetros asignados anteriormente en su base de datos[11].

### Servidor DNS

Los navegadores web se comunican mediante direcciones IP, pero no resulta una forma eficiente de navegar para las personas, dado que, que asocian de mejor manera nombres de dominio que direcciones IP. El Servidor de Nombres de Dominio interpreta los nombres de dominio a direcciones IP para que los navegadores puedan cargar recursos de la red y las personas puedan navegar de forma más fácil. Los servidores DNS eluden la necesidad de memorizar direcciones IP tales como 172.31.110.20 en IPv4 o direcciones IP con combinaciones alfanuméricas mucho más complicadas, así como fe80::38c7:6210:3e12:12b7%7 (en IPv6) [12].

### Servidor WEB

Un servidor web es la combinación de hardware y software que, mediante protocolos como el Protocolo de Transferencia de Hipertexto (HTTP), Protocolo de Transferencia simple de correo (SMTP), Protocolo de transferencia de archivos (FTP), entre otros, es capaz de responder las peticiones realizadas por clientes en la Red mundial extendida (WWW). Su principal función es mostrar el contenido de un sitio *web* luego de procesar, almacenar y entregar las páginas *web* a los usuarios [13].

La parte física del servidor *web (hardware)* está conectada a internet y facilita el intercambio de datos entre diferentes dispositivos conectados, en cambio, la parte lógica *(software)* del servidor maneja el modo en que los usuarios acceden a los datos. Los datos para sitios *web*, aplicaciones web o aplicaciones basadas en web son albergados en alojamientos web *(web hosting)*[13].

# 2 METODOLOGÍA

En primera instancia se realizó una investigación sobre el manejo y administración de contenedores, centrándose particularmente en *Podman* que fue establecida como la principal herramienta para el despliegue del presente proyecto de titulación, a diferencia de *Docker* que es el programa de administración de contenedores más conocido en el medio, *Podman* presenta características más eficientes en este campo, entre las cuales destacan la compatibilidad total con todo el software desarrollado para el manejo de contenedores, la no necesidad de otorgar privilegios de *root* y no requerir de demonios para crear imágenes y ejecutar contenedores[14].

Una vez aclarado el panorama sobre como implementar y administrar contenedores se procedió con el diseño para la implementación y despliegue de tres servidores: DNS, DHCP y WEB. Para esto, se investigó cuáles son las plataformas más amigables y con mayor soporte para la implementación de servidores, además, tomando en cuenta que dichas plataformas puedan ser implementadas a su vez dentro del entorno de un contenedor.

Los servidores DNS, DHCP y WEB fueron implementados en tres contenedores por separado pero manejados dentro de un *pod*, lo que implica que los tres contenedores son capaces de compartir diferentes características como por ejemplo estar dentro de una misma red, esto permite manejarlos dentro de un ambiente controlado y que facilita las pruebas de funcionamiento [15].

Finalmente se realizó la comprobación del funcionamiento de cada uno de los tres servidores en un ambiente simulado de un esquema cliente-servidor, haciendo uso de una máquina virtual principal en la cual corren los tres contenedores con sus respectivos servicios y máquinas cliente, las cuales están en una misma red.

Se verificó que haya una correcta comunicación entre las máquinas, que cada máquina sea capaz de reconocer los servicios dentro de los contenedores, y que cada servidor

implementado cumpla con su respectiva tarea. En el caso del servidor DHCP, que asigne direcciones IP dentro del rango establecido[10]. Por su parte, el servidor DNS debe ser capaz de realizar la resolución de nombres de dominio [12], mientras que el servidor web debe desplegar páginas web sin ningún inconveniente [13].

# 3 RESULTADOS

El presente proyecto tiene como objetivo desplegar tres servidores DNS, DHCP y WEB, cada uno de estos servidores será implementado dentro de un contenedor creado, manejado y administrado específicamente por el motor de contenedores *Podman* y, posteriormente, se probará su funcionamiento en un entorno controlado con la ayuda de máquinas virtuales[6].

## 3.1 Análisis del Contenedor *Podman* y sus características.

*Podman* es un programa de código abierto que está disponible para gran parte de las plataformas Linux y que reside en GitHub. Es un motor de contendores cuya principal característica es que deja de lado el uso de demonios, a diferencia de *Docker*. Además, administra y ejecuta todo el ecosistema de contenedores incluyendo imágenes, *pods* y volúmenes [6].

### Alcance y características de *Podman*

- Afinidad con varios formatos de imágenes para contenedores, incluidas imágenes *Docker* y OCI[16].
- Administración completa de tales imágenes, incluyendo la extracción de varias fuentes de verificación y confianza, la creación (las imágenes se construyen a través de *Dockerfile* o *Containerfile* o *committed from a container*) y el envío a registros y otros *backends* de almacenamiento[16].
- Administración total del ciclo de vida del contenedor, incluyendo la creación (tanto a partir de un sistema de archivos raíz ampliado o de una imagen), la creación de puntos de control, la ejecución, la restauración utilizando la tecnología de Punto de Control/Restauración en el Espacio de Usuario (CRIU) y la eliminación[17].
- Gestión completa de redes de contenedores, utilizando la Interfaz de Red de Contenedores (CNI), Netavark que es una tecnología de para la configuración

de redes de contenedores y slirp4netns que proporciona redes en modo usuario (*slirps*) para trabajar en contenedores sin privilegios *root*[18], [19].

- Compatibilidad con *pods*, grupos de contenedores que comparten recursos y se administran juntos[16].
- Soporte para ejecutar contenedores y *pods* sin privilegios *root* u otros privilegios elevados[16].
- Aislamiento de recursos de contenedores y *pods*[16].
- Compatibilidad con una interfaz CLI compatible con *Docker*[16].
- Sin demonio administrador, para mejorar la seguridad y reducir la utilización de recursos en reposo[16].

### ***Podman* vs. *Docker*.**

*Docker* es un gestor estándar de contenedores y el mayor referente respecto al desarrollo de contenedores dentro de la industria debido a su gran versatilidad, puesto que, se desarrolló como una herramienta independiente y autosuficiente capaz de cubrir las necesidades de los desarrolladores a medida que la complejidad del manejo de contenedores aumentaba [14].

Sin embargo, con el pasar del tiempo tal autosuficiencia presento defectos, ya que, aparecieron diferentes herramientas y plataformas más livianas que presentan dificultades para interactuar con *Docker* [14].

Hay múltiples diferencias entre *Podman* y *Docker*, pero entre las más importantes se tiene:

- En su arquitectura, *Docker* utiliza demonios (programas en segundo plano) para crear imágenes y ejecutar contenedores, es decir, una lógica cliente-servidor con el demonio como intermediario. *Podman*, por otro lado, tiene una arquitectura libre de demonios, lo que le permite ejecutar contenedores bajo el mismo usuario que inició el contenedor, es decir, no se necesita intermediario.[14].
- *Podman* puede trabajar sin necesidad de otorgar privilegios de usuario raíz (*root*) a sus contenedores lo que adiciona una barrera natural de seguridad entre los niveles raíz y usuario. *Docker* por defecto otorga privilegios de raíz a sus contenedores provocando que sea una puerta de enlace para posibles atacantes [14].

- Como una herramienta autosuficiente *Docker* es capaz de crear imágenes por sí solo, por otro lado, *Podman* necesita de una herramienta especializada llamada Buildah (creador de contenedores) [14].
- Finalmente, la diferencia crucial entre estas dos tecnologías es que *Docker* tiene un enfoque todo en uno, mientas que, *Podman* cuenta con un enfoque modular que se basa en herramientas especializadas para tareas específicas[14].

## 3.2 Diseño de contenedores con DHCP, DNS y WEB.

### Instalación de *Podman*

Una vez instalado *Ubuntu Linux* se puede proceder con la instalación de la herramienta principal de manejo de contendedores para este proyecto como lo es *Podman*.

En primer lugar, hay que abrir una terminal en Ubuntu y acceder al modo de super usuario o *root* con el comando **sudo su**. Inmediatamente el sistema pide la contraseña de usuario, la cual es la misma configurada para el inicio de sesión. Una vez ingresada la contraseña, la terminal ya se encuentra en modo de super usuario como se ve en la Figura 3.21.

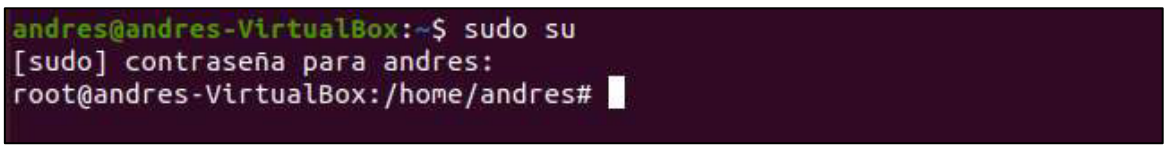


**Figura 3.1** Comando **sudo su** en terminal *Ubuntu*

El siguiente paso es importante para evitar errores de compatibilidad con cualquier programa dentro de Ubuntu incluido *Podman.* Se debe actualizar todos los paquetes del sistema con dos comandos, el primero es **apt update -y** y el segundo es **apt upgrade -y**, en la terminal se puede resolver los dos comandos al mismo tiempo con la siguiente estructura: **apt update -y && apt upgrade -y** como se ve en la Figura 3.2. El parámetro **-y** es para que el proceso no solicite una confirmación.

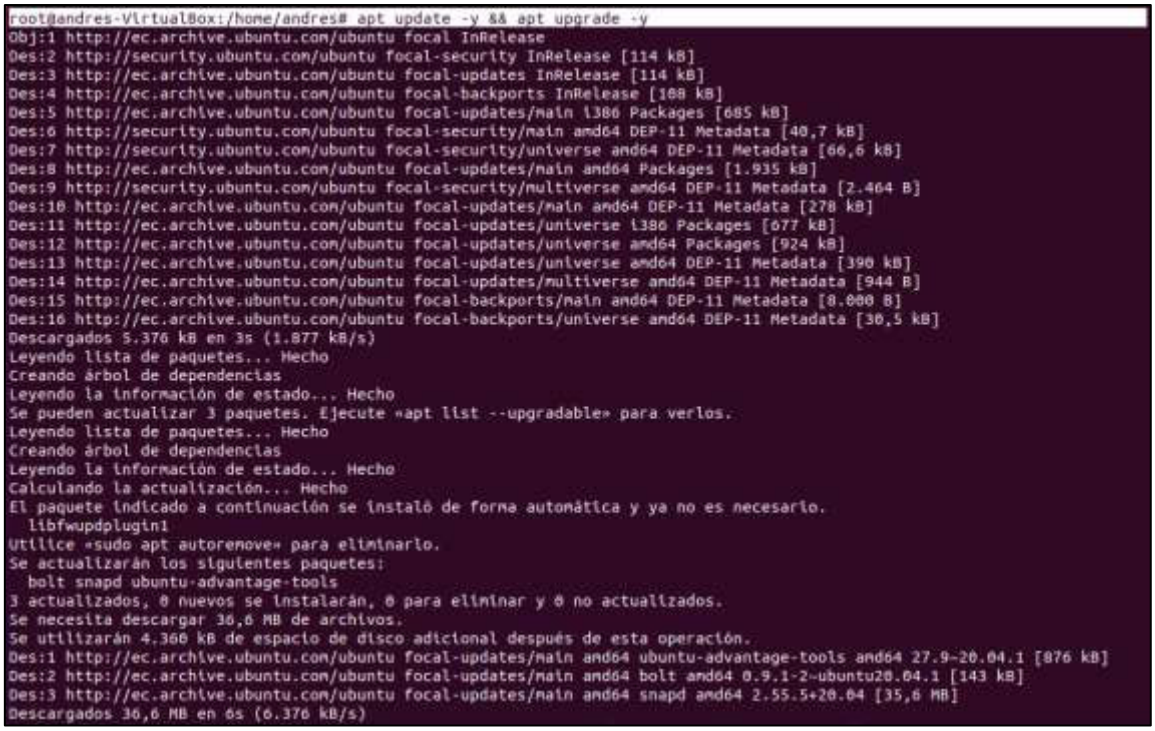


**Figura 3.2** Comandos **apt update -y && apt update -y** para actualizar paquetes

En Ubuntu 20.04, para instalar *Podman* es necesario instalar primero ciertas dependencias y agregar el repositorio que lo contiene, ya que no se encuentra en los repositorios por defecto.

Las dependencias se instalan con el comando **apt-get install curl wget gnupg2 -y** como se puede apreciar en la Figura 3.3 [20].

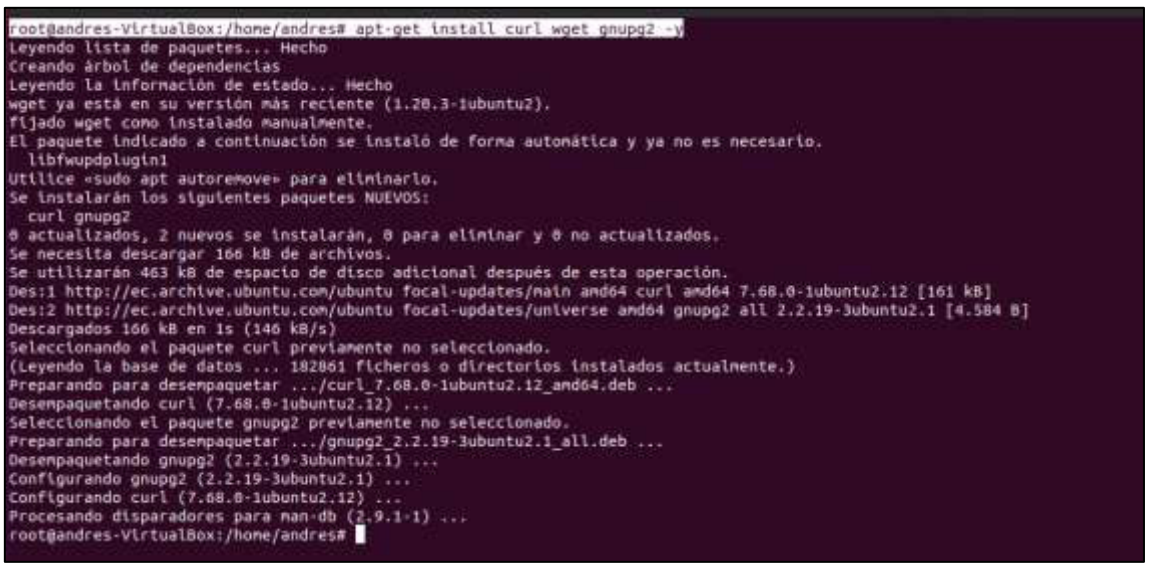


**Figura 3.3** Instalación de dependencias para *Podman*

Para agregar el repositorio se procede con el comando **echo "deb https://download.opensuse.org/repositories/devel:/kubic:/libcontainers:/stable/xUbuntu_${VERSION_ID}/ /" | sudo tee /etc/apt/sources.list.d/devel:kubic:libcontainers:stable.list** como se ve en la Figura 3.4 [20].


**Figura 3.4** Actualización de repositorio para *Podman*

Lo siguiente es descargar y agregar una llave de Protección de Privacidad de GNU (*GPG*), que permite firmar y encriptar los datos por temas de seguridad [21]. Esto se realiza con el comando **wget -nv https://download.opensuse.org/repositories/devel:kubic:libcontainers:stable/xUbuntu_${VERSION_ID}/Release.key -O-** | **apt-key add –** ,como se ven la Figura 3.5[20].


**Figura 3.5** Descarga de llave GPG

El último paso es actualizar el repositorio e instalar *Podman* con el comando **apt-get update -qq -y && apt-get -qq --yes install Podman** como se ve en la Figura 3.6 [20].

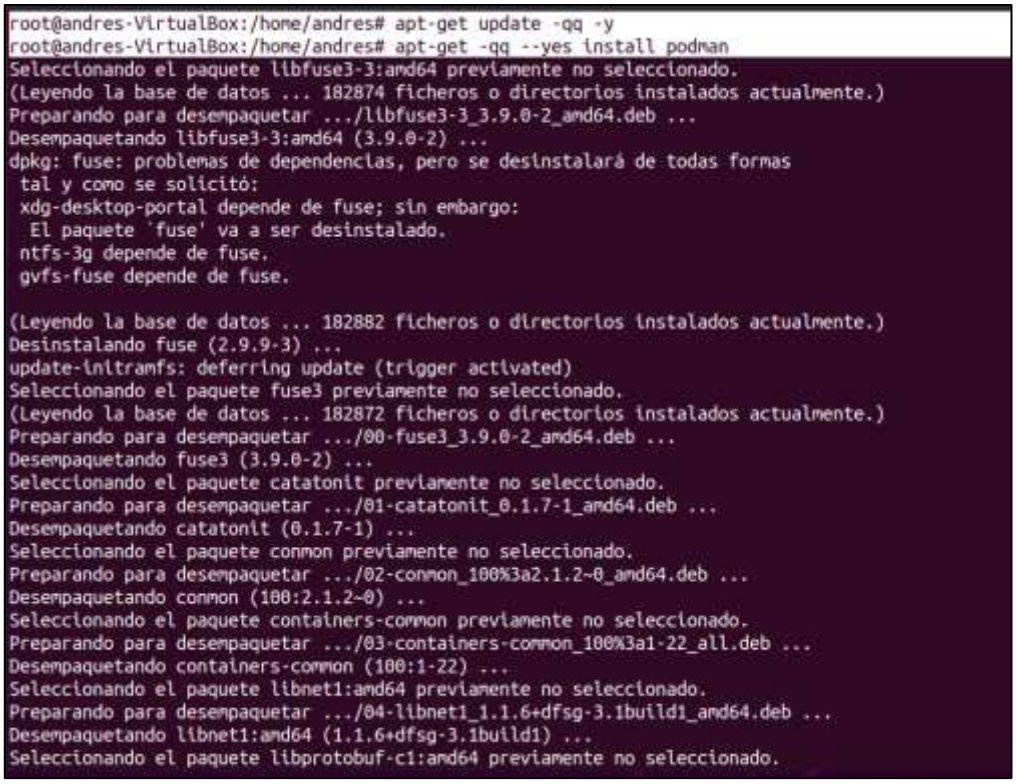


**Figura 3.6** Actualización de repositorio e instalación de *Podman*

Una vez realizado el proceso de instalación de *Podman*, se puede verificar la versión instalada con el comando **podman --versión**, como se aprecia en la figura 3.7 [20].

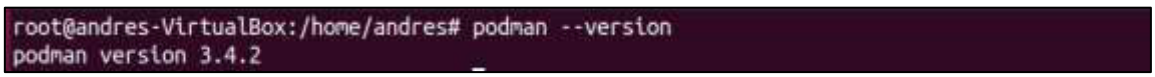


**Figura 3.7** Versión de *Podman* instalada en el sistema

Si se requiere ver más datos acerca *Podman*, se puede utilizar el comando **podman info** y se despliega toda la información referente al programa, como se ve en la figura 3.8 [20].


**Figura 3.8** Información completa de *Podman*

### Diseño de contendores base

Se diseñó un tipo de contenedor base que maneja las características necesarias para la implementación de cada servidor.

Para montar el contenedor base se necesita la imagen de un sistema operativo que presente facilidades para la implementación de servidores, en este caso se utilizó nuevamente *Ubuntu Linux* en su versión 20.04, la diferencia entre la imagen normal diseñada para ordenadores o máquinas virtuales es que, la imagen para contenedores solo cuenta con lo mínimo para que el sistema funcione y en el proceso de implementación se instalan únicamente las herramientas necesarias para el propósito deseado.

Las imágenes diseñadas para contenedores se pueden obtener de varios repositorios oficiales como *GitHub* o *DockerHub,* éste último maneja un amplio catálogo de plantillas

de imágenes de sistemas operativos y programas diseñados para contenedores[22]. Dentro de *DockerHub* se encuentra la imagen de Ubuntu, tal como se aprecia en la Figura 3.9, la página indica el comando para la descarga directo desde *Linux*.

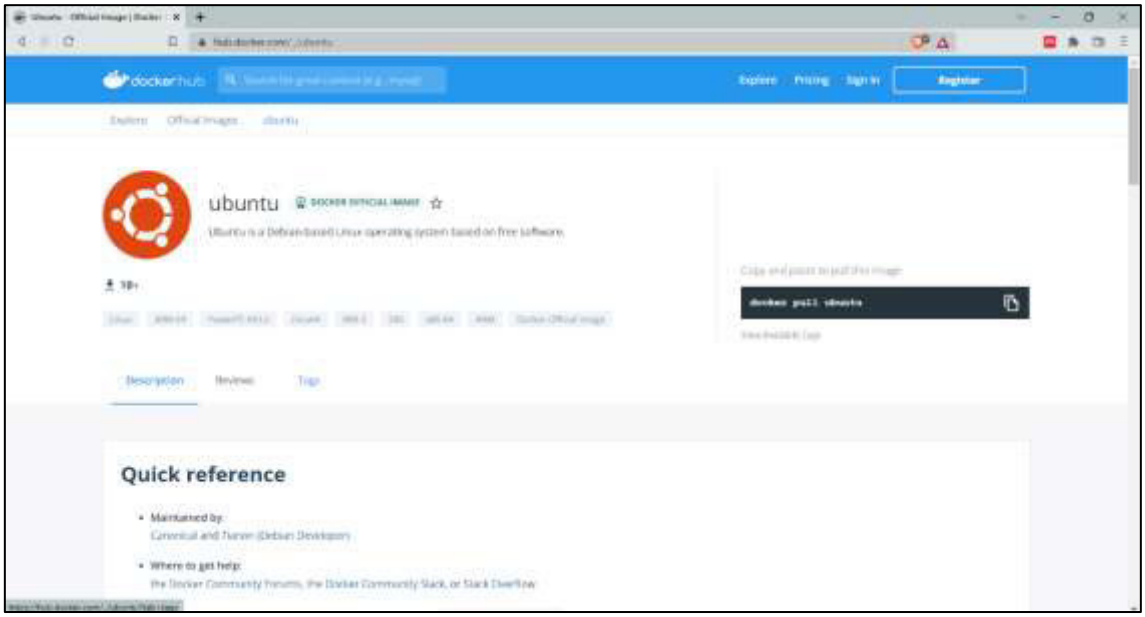


**Figura 3.9** Imagen de *Ubuntu* en el repositorio *Docker Hub* [23]

Para descargar la imagen directamente desde la terminal se procede con el comando **Podman pull Ubuntu:20.04** como se ve la Figura 3.10, si no se especifica la versión, *Podman* descarga la última versión disponible.

```
root@andres-VirtualBox:/home/andres# podman pull ubuntu:20.04
Resolved "ubuntu" as an alias (/etc/containers/registries.conf.d/000-shortnames.conf)
Trying to pull docker.io/library/ubuntu:20.04...
Getting image source signatures
Copying blob d7bfe07ed847 done
Copying config 20fffa419e done
Writing manifest to image destination
Storing signatures
20fffa419e3aaca519dad480b21e86d9bcfbc7795abf9419adc5adce1198c95e
```

**Figura 3.10** Descarga de la imagen de Ubuntu con *Podman*

Para comprobar que la imagen ha sido descargada se puede verificar con el comando **podman image list –all,** que despliega una lista de todas las imágenes guardadas localmente. En la Figura 3.11 se puede apreciar que la imagen de Ubuntu ya se encuentra descargada.

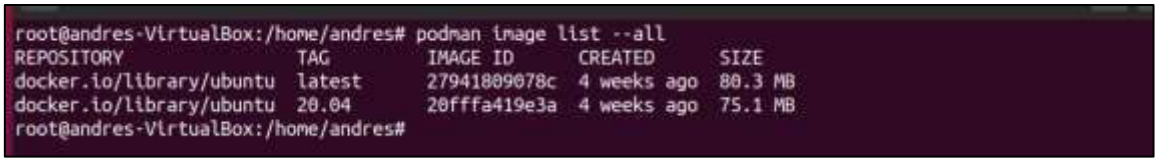


**Figura 3.11** Lista de imágenes descargas y disponibles localmente

Para crear los contenedores base que sirven para la implementación de cada servidor se procede con el comando **podman run**, acompañado de varios parámetros que son iguales para cada uno de los contenedores y difiere únicamente al nombrar cada contenedor. Así por ejemplo, para el contenedor del servidor DNS se procede con el comando **--intreactive --tty --network host --name  dns-server ubuntu:20.04**, como se ve en Figura 3.12. En el caso del contenedor para el servidor web se procede con el comando **--intreactive --tty --network host --name  web-server ubuntu:20.04**, como se aprecia en la Figura 3.13.


**Figura 3.12** comando para crear contenedor de servidor DNS

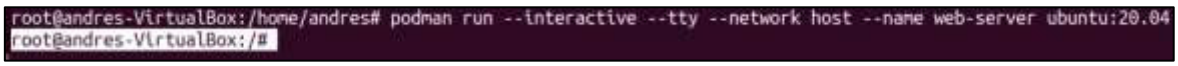


**Figura 3.13** comando para crear contenedor de servidor WEB

En el caso del servidor DHCP, su contenedor necesita dos características especiales para funcionar de forma correcta. La primera es que necesita una imagen de Ubuntu que cuente con un administrador de sistemas y servicios que se ejecute como PID 1, por ejemplo, *systemd*. La principal razón de esto es porque así se pueden visualizar y comprobar los procesos y, en caso de tener errores, encontrarlos de una manera más precisa. Por lo expuesto anteriormente, se procede a descargar la imagen *jrei/systemd-ubuntu* con el comando **podman pull *Docker*.io/jrei/systemd.ubuntu** [24].

La segunda característica especial es que es necesario agregar el parámetro **--privileged** en el comando **podman run**. Este parámetro permite que el contenedor tenga permisos *root,* algo absolutamente para el servidor DHCP, caso contrario el servidor no se levantará. [25].

Por lo tanto, el comando para crear el contenedor DHCP es: **podman run -d --privileged --interactive --network host --name dhcp-server jrei/systemd-ubuntu,** como se observa en la Figura 3.14.


**Figura 3.14** comando para crear contenedor de servidor DHCP

Los parámetros utilizados en el comando se explican a continuación, en la Tabla 3.1.

**Tabla 3.1** Parámetros del comando **podman run**[25]

| Parámetro | Descripción |
|---|---|
| -d | detach, contenedor en segundo plano |
| --interactive | Permite que el contenedor sea interactivo |
| --tty | Ejecuta un *shell* interactivo desechable. El valor predeterminado es falso. |
| --network host | Asigna una la configuración de red de la maquina *host* |
| --name | Asigna un nombre especifico al contenedor |
| --privileged | Asigna permiso *root* al contenedor |
| ubuntu:20.04 | Imagen a ser montada en el contenedor |
| jrei/systemd-ubuntu | Imagen de Ubuntu con systemd |

Al crear cada contenedor, automáticamente se despliega una terminal de la imagen de Ubuntu montada, por lo que se procede en primer lugar a realizar un proceso de actualización e instalación de las herramientas requeridas para la implementación de los servidores, el proceso se repite en cada contenedor creado.

Para hacer una actualización de los paquetes del sistema, se utiliza el comando **apt update -y && apt upgrade -y**, como se observa en la Figura 3.15.

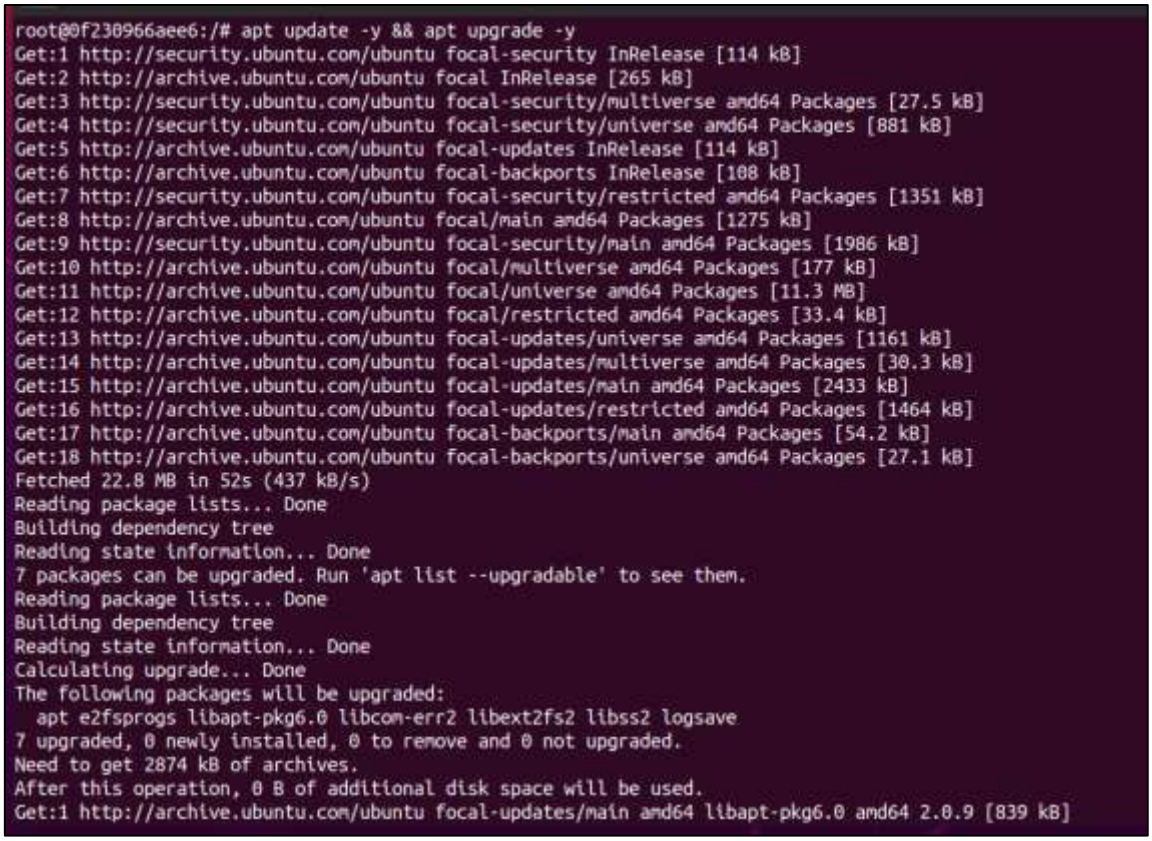


**Figura 3.15** Actualización de los paquetes de Ubuntu dentro del contenedor

Una vez actualizados los paquetes se procede a instalar las herramientas requeridas para la implementación de los servidores. Inicialmente, se necesita de un editor de texto para modificar los archivos de configuración, en este caso se hará uso de *nano*. Para la instalación se utiliza el comando **apt-get install nano**, como se ve en la Figura 3.16.

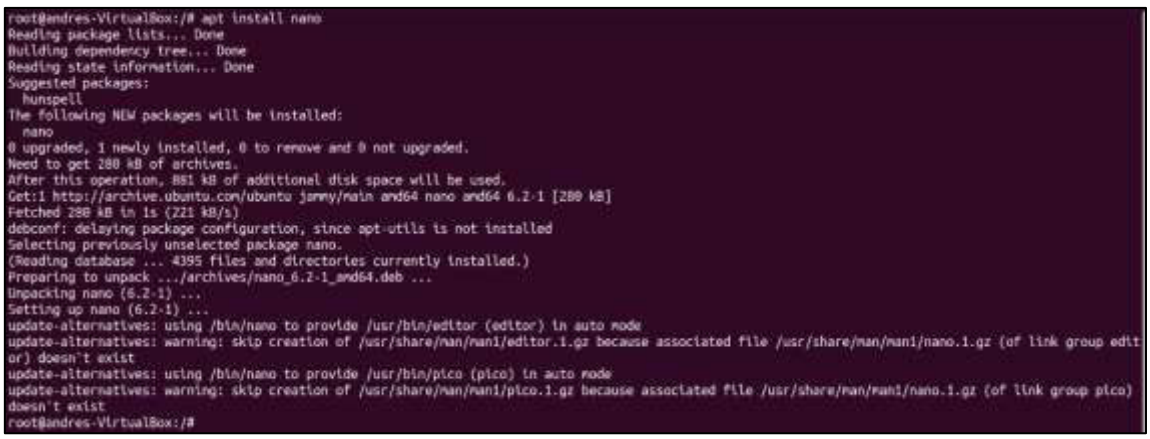


**Figura 3.16** Instalación del editor de texto *nano*

La segunda herramienta son las *net-tools,* necesarias para manejar los parámetros de red dentro del contenedor. Para instalarlas, se procede con el comando **apt-get install net-tools**, como se aprecia en la Figura 3.17.

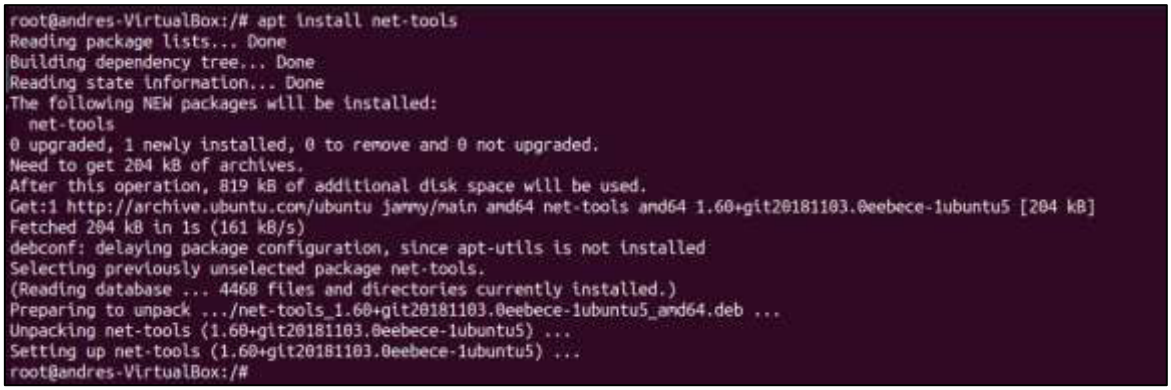


**Figura 3.17** Instalación de las *net-tools*

## 3.3 Implementación de los tres contenedores diseñados

Una vez diseñados los contenedores base, es momento de implementar cada servidor. Debido a que cada uno de los servidores requiere procesos diferentes, se procederá a explicar la implementación de cada uno de ellos por separado.

El comando para visualizar todos los contenedores creados localmente es **podman container list --all.** Como se ve en la Figura 3.18, el comando permite enlistar los contenedores ya creados anteriormente.

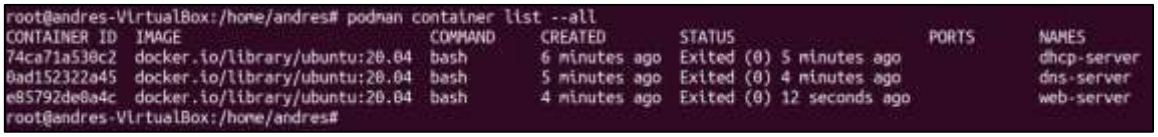


**Figura 3.18** Comando para mostrar todos los contenedores creados localmente.

### Implementación del servidor DHCP

Para implementar el servidor DHCP se seguirá con el proceso especificado en el diagrama de Flujo mostrado en la Figura 3.19.

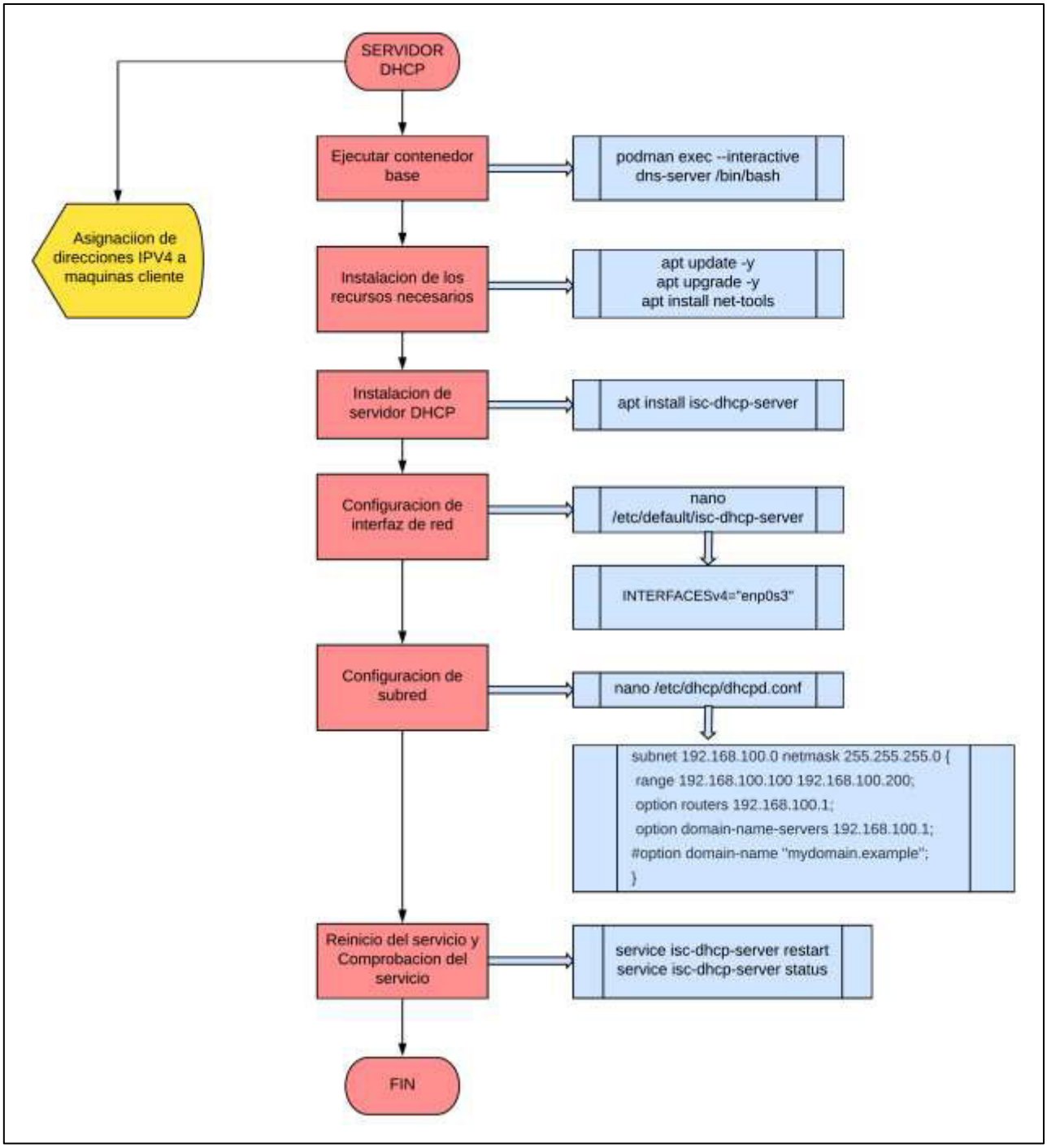


**Figura 3.19** Diagrama de Flujo de la implementación del servidor DHCP.

En primer lugar, se debe abrir el contenedor base diseñado para este servidor, para lo cual se usa el comando **podman exec -it dhcp-server bash,** como se ve en la Figura 3.20, e inmediatamente se abre una terminal dentro del contenedor.

```
root@andres-VirtualBox:/home/andres# podman exec -it dhcp-server bash
root@andres-VirtualBox:/#
```

**Figura 3.20** Ejecución del contenedor DHCP.

Los parámetros del comando se explican a continuación, en la Tabla 3.2.

**Tabla 3.2** Parámetros del comando ***Podman* exec** [26]

| Parámetro | Descripción |
|---|---|
| -it | Permite que el contenedor sea interactivo |
| dhcp-server | Nombre del contenedor que se desea ejecutar |
| bash | Ejecuta una terminal *bash* |

Luego, se procede a instalar la herramienta de servidor DHCP llamada *isc-dhcp-server* con el comando **apt-get install isc-dhcp-server**, y se procede a descargar todos los paquetes y archivos de configuración necesarios para el servidor, como se ve en la figura 3.21 [27].

```
root@andres-VirtualBox:/# apt-get install isc-dhcp-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  isc-dhcp-common libdns-export1109 libirs-export161 libisc-export1105 libisccfg-export163
Suggested packages:
  policykit-1 isc-dhcp-server-ldap policycoreutils
The following NEW packages will be installed:
  isc-dhcp-common isc-dhcp-server libdns-export1109 libirs-export161 libisc-export1105 libisccfg-export163
0 upgraded, 6 newly installed, 0 to remove and 0 not upgraded.
Need to get 1503 kB of archives.
After this operation, 4913 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 isc-dhcp-common amd64 4.4.1-2.1ubuntu5.20.04.3 [44.9 kB]
Get:2 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 libisc-export1105 amd64 1:9.11.16+dfsg-3~ubuntu1 [175 kB]
Get:3 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 libdns-export1109 amd64 1:9.11.16+dfsg-3~ubuntu1 [765 kB]
Get:4 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 libisccfg-export163 amd64 1:9.11.16+dfsg-3~ubuntu1 [45.9 kB]
Get:5 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 libirs-export161 amd64 1:9.11.16+dfsg-3~ubuntu1 [18.6 kB]
Get:6 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 isc-dhcp-server amd64 4.4.1-2.1ubuntu5.20.04.3 [453 kB]
Fetched 1503 kB in 1s (1388 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
```

**Figura 3.21** Instalación de la herramienta *isc-dhcp-server*

Lo siguiente es configurar el archivo *isc-dhcp-server,* para lo cual es necesario en primer lugar conocer las interfaces de red que se están manejando, las interfaces disponibles se las puede ver con el comando **ifconfig** en una terminal externa, como se puede observar en la Figura 3.22.


**Figura 3.22** Interfaces de red disponibles.

Una vez se conocen las interfaces disponibles se procede a abrir el archivo que se encuentra en el directorio /**etc**/**default**, como se ve en la Figura 3.23. Dentro del archivo se configura la interfaz de red que va a usarse para el servidor.


**Figura 3.23** Acceso al archivo de configuración **isc-dhcp-server** con *nano*

La interfaz a utilizar es la **enp0s3**, de tal forma que el archivo de configuración se modifica como se observa en la Figura 3.24[27]. La línea modificada se resalta en color blanco.


**Figura 3.24** Configuración de interfaz de red para servidor DHCP.

Posteriormente, se debe modificar el archivo **dhcpd.conf** que se ubica en el directorio /**etc**/**dhcp**, como se ve en la Figura 3.25.


**Figura 3.25** Acceso al archivo **dhcpd.conf** con *nano*

Dentro del archivo se procede a agregar una configuración básica, en la cual se encuentra la subred, el rango requerido, la dirección del *router* y la dirección de servidores DNS locales y opcionales para direcciones externas. De tal manera que el archivo debe quedar como se observa en la Figura 3.26. Las configuraciones añadidas se resaltan en color blanco[27].

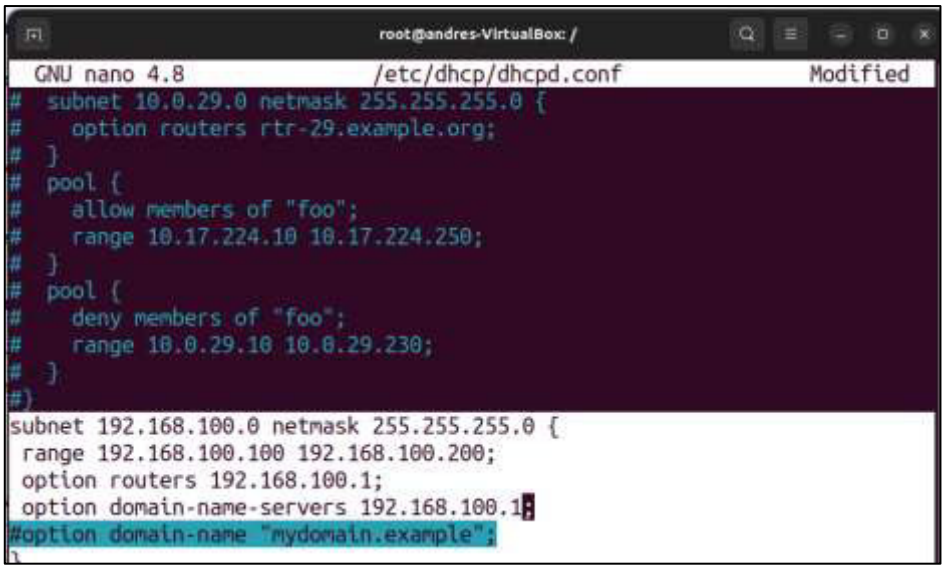


**Figura 3.26** configuración básica de subred para servidor DHCP

Finalmente, se debe reiniciar el servicio con el comando **systemctl restart isc-dhcp-serve**. Para comprobar que el servicio está activo se utiliza el comando **systemctl status isc-dhcp-server** [27]. En la Figura 3.27 se puede observar que el servicio está levantado y no presenta errores.

```
root@andres-VirtualBox:/# systemctl restart isc-dhcp-server
root@andres-VirtualBox:/# systemctl status isc-dhcp-server
● isc-dhcp-server.service - ISC DHCP IPv4 server
     Loaded: loaded (/lib/systemd/system/isc-dhcp-server.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2022-08-21 09:20:22 UTC; 8s ago
       Docs: man:dhcpd(8)
   Main PID: 742 (dhcpd)
      Tasks: 4 (limit: 387)
     Memory: 4.4M
        CPU: 7ms
     CGroup: /system.slice/isc-dhcp-server.service
             └─742 dhcpd -user dhcpd -group dhcpd -f -4 -pf /run/dhcp-server/dhcpd.pid -cf /etc/dhcp/dhcpd.conf enp0
s3

Aug 21 09:20:22 andres-VirtualBox sh[742]: PID file: /run/dhcp-server/dhcpd.pid
Aug 21 09:20:22 andres-VirtualBox dhcpd[742]: Wrote 0 leases to leases file.
Aug 21 09:20:22 andres-VirtualBox sh[742]: Wrote 0 leases to leases file.
Aug 21 09:20:22 andres-VirtualBox dhcpd[742]: Listening on LPF/enp0s3/08:00:27:4d:7d:99/192.168.100.0/24
Aug 21 09:20:22 andres-VirtualBox sh[742]: Listening on LPF/enp0s3/08:00:27:4d:7d:99/192.168.100.0/24
Aug 21 09:20:22 andres-VirtualBox dhcpd[742]: Sending on   LPF/enp0s3/08:00:27:4d:7d:99/192.168.100.0/24
Aug 21 09:20:22 andres-VirtualBox sh[742]: Sending on   LPF/enp0s3/08:00:27:4d:7d:99/192.168.100.0/24
Aug 21 09:20:22 andres-VirtualBox dhcpd[742]: Sending on   Socket/fallback/fallback-net
Aug 21 09:20:22 andres-VirtualBox sh[742]: Sending on   Socket/fallback/fallback-net
Aug 21 09:20:22 andres-VirtualBox dhcpd[742]: Server starting service.
```

**Figura 3.27** Reinicio y comprobación del servidor DHCP

### Implementación del servidor DNS

El proceso de implementación se guía por el diagrama mostrado en la Figura 3.28.

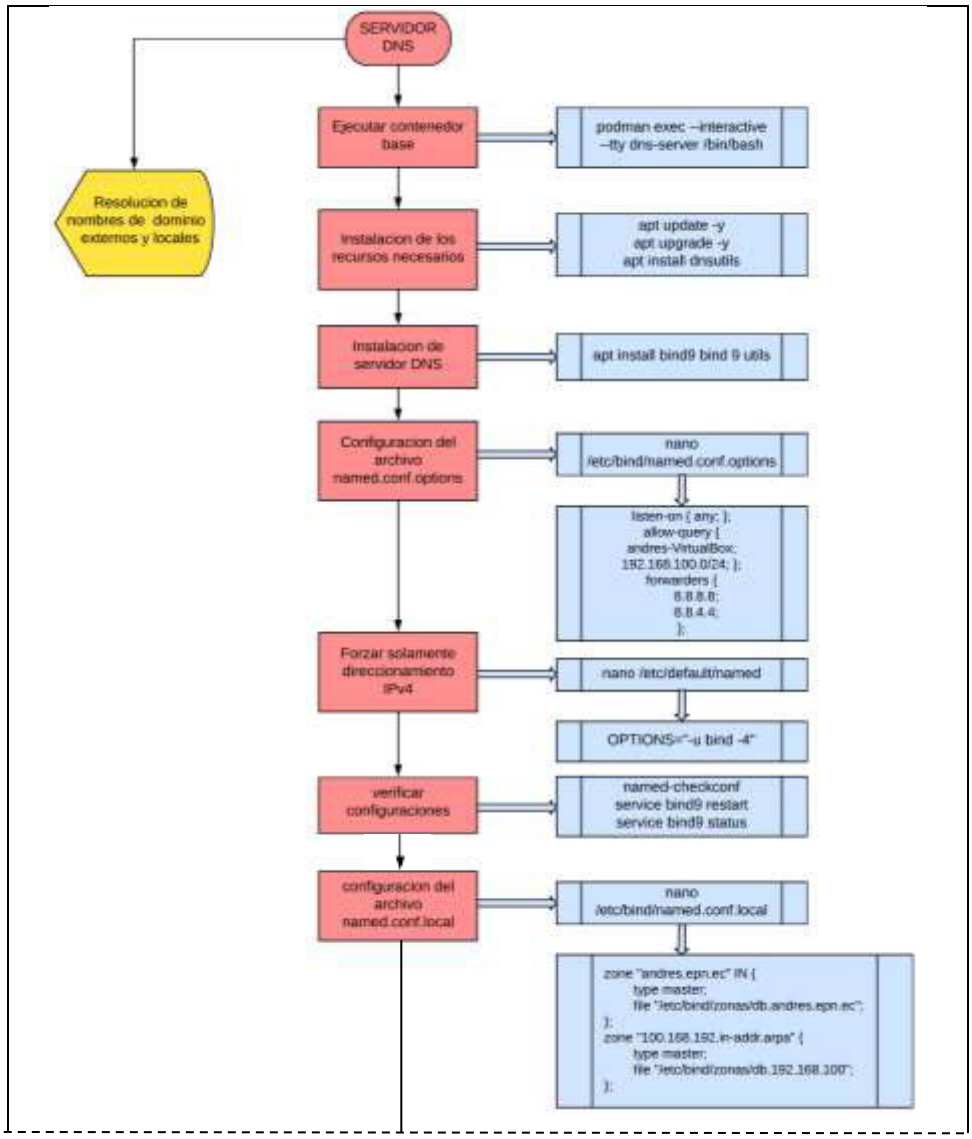


**Figura 3.28** Diagrama de Flujo para implementación del servidor DNS parte 1

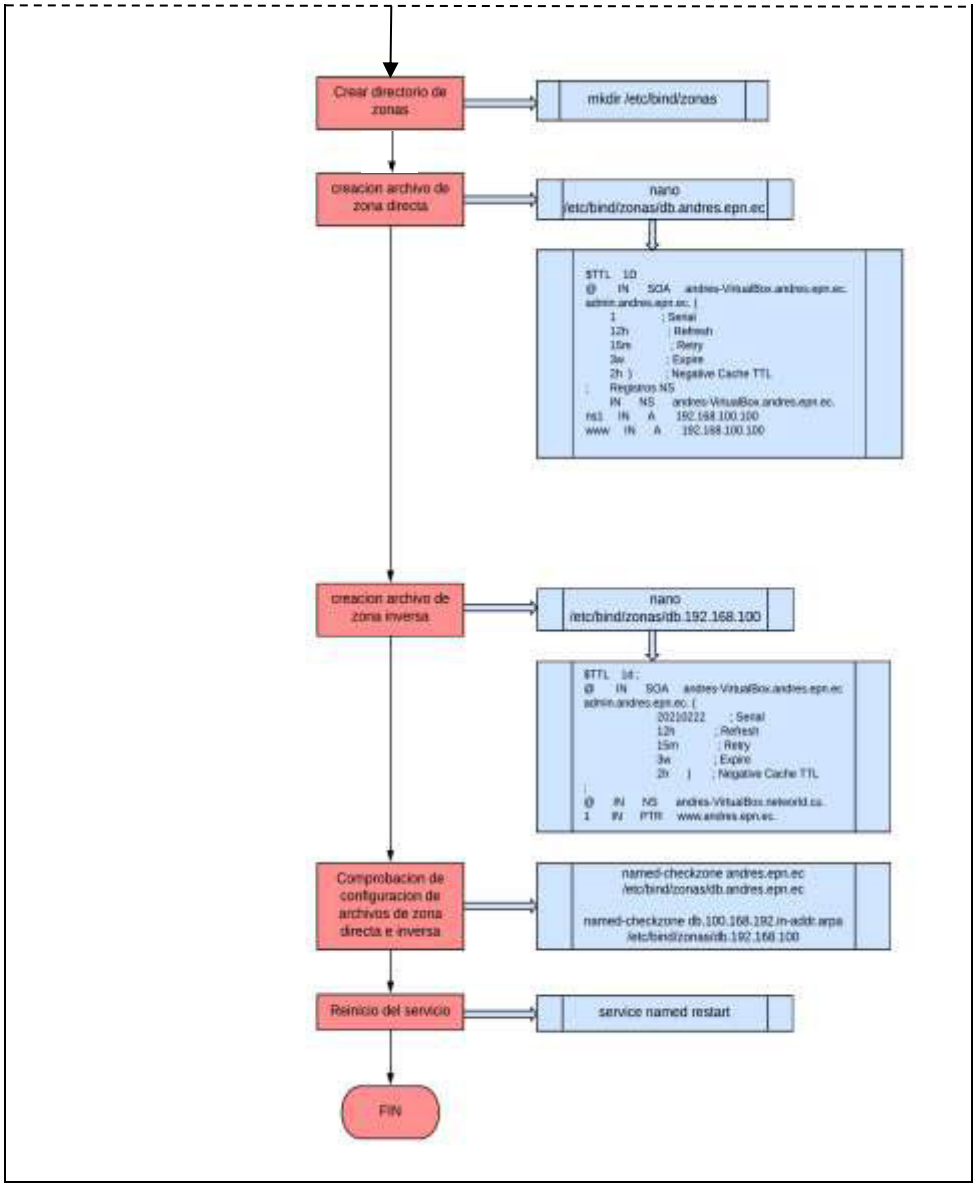


**Figura 3.29** Diagrama de Flujo para implementación del servidor DNS parte 2

En primer lugar, se ejecuta el contenedor creado para el servidor DNS, para lo cual se usa el comando **podman start dns-server** y, luego, el comando **podman exec –interactive –tty dns-server /binbash**, como se muestra en la Figura 3.30. Los parámetros son los mismos utilizados en el servidor DHCP y son explicados en la Tabla 3.2.

```
root@andres-VirtualBox:/home/andres# podman start dns-server
root@andres-VirtualBox:/home/andres# podman exec --interactive --tty dns-server /bin/bash
root@andres-VirtualBox:/#
```

**Figura 3.30** Ejecución del contenedor para servidor DNS

Posteriormente, se procede a instalar el servidor DNS llamado *BIND9* y sus respectivas utilidades con el comando **apt-get install bind9 bind9-utils**, como se ve en la Figura 3.31. Durante la instalación se pedirán datos de ubicación y zona horaria que se llenarán según el caso. [28]


**Figura 3.31** Instalación del servidor DNS *BIND9*

Una vez instalado el servidor, se puede realizar una configuración mínima para poder demostrar el funcionamiento, por lo que se procede a modificar el archivo **named.conf.options** ubicado en el directorio /**etc/bind** utilizando el editor de texto *nano,* el comando se muestra en la Figura 3.32 [28].


**Figura 3.32** Acceso al archivo **named.conf.options** con *nano*

Una vez abierto el archivo se necesita configurar los siguientes parámetros:

1. Definir la dirección donde estará escuchando *Bind9*, en este caso no escucha una dirección especifica por lo que se usa el comando **listen-on { any; };** [28].

2. Definir desde que dirección IP o red es posible realizar consultas. Por lo general es en la misma red donde está servidor DNS, por lo que se usa el comando **allow-query { andres-VirtualBox; 192.168.100.0/24; };** [28].
3. Establecer los servidores DNS a los que *Bind9* redirigirá las peticiones que no sea capaz de resolver. Se utiliza el comando **forwarders { 8.8.8.8; 8.8.4.4; };** [28].

Con las configuraciones utilizadas, el archivo queda como se observa en la Figura 3.33. Las líneas de comando agregadas se resaltan en color blanco. Se procede a guardar la configuración y cerrar el archivo.

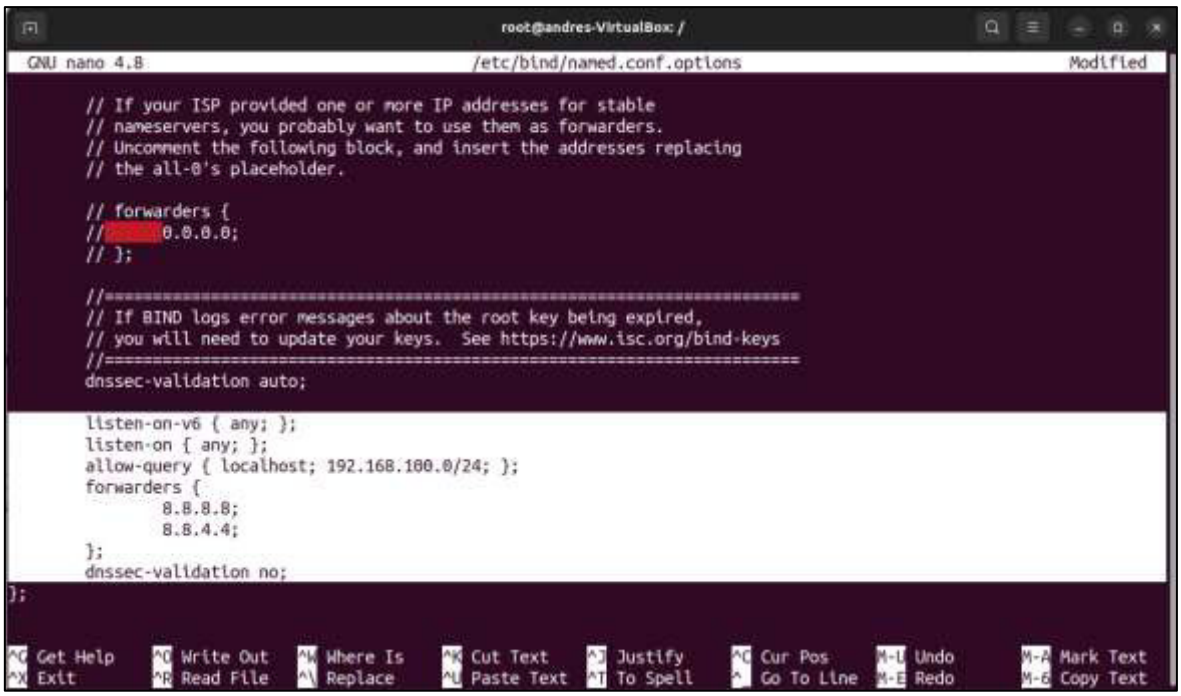


**Figura 3.33** Líneas de comando agregadas al archivo **named.conf.options**

Lo siguiente es restringir al servidor para que únicamente use el direccionamiento IPv4. Para esto, se debe acceder al archivo **named** ubicado en el directorio /**etc**/**default**, como se ve en la Figura 3.34.

```
root@andres-VirtualBox:/# nano /etc/default/named
```

**Figura 3.34** Acceso al archivo **named** con *nano*

Dentro del archivo **named,** en la última línea, se debe reemplazar **OPTIONS="-u bind"** por **OPTIONS="-u bind -4"**, el resultado es el archivo como se observa en la Figura 3.35 [28]. Guardar y salir.

Para comprobar si la configuración es correcta, se usa el comando **named-checkconf.** Si la salida del comando no muestra ningún error, entonces la configuración es correcta [28].


**Figura 3.35** Configuración del archivo **named**

Ahora, se debe agregar la configuración de zonas en el archivo **named.conf.local**, que se encuentra en el directorio /**etc**/**bind**. Para editarlo, se procede como en la Figura 3.36

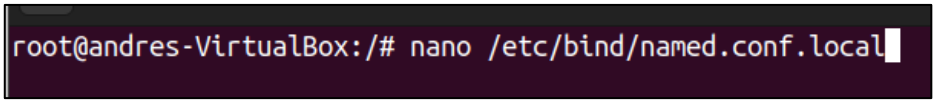


**Figura 3.36** Acceso al archivo **named.conf.local**

Se debe configurar tanto la zona directa como la zona inversa. Para la zona directa se debe agregar el nombre de dominio a utilizarse, en este caso, el dominio de ejemplo es **andres.epn.ec.** También se agrega el tipo, que, en este caso en **maestro,** y la ubicación del archivo de configuración de zona directa, que se lo encontrará en "**/etc/bind/zonas/db.andres.epn.ec"** [28].

Para la zona inversa se debe agregar la dirección IP en sentido de la siguiente forma: **"100.168.192.in-addr.arpa".** También se debe agregar el tipo, que en este caso es **maestro** y la ubicación del archivo de configuración de la zona inversa, que en este caso es **"/etc/bind/zonas/db.192.168.100"** [28].

El archivo configurado debe ser igual a como se muestra en la Figura 3.37. Las líneas agregadas resaltan en color blanco.

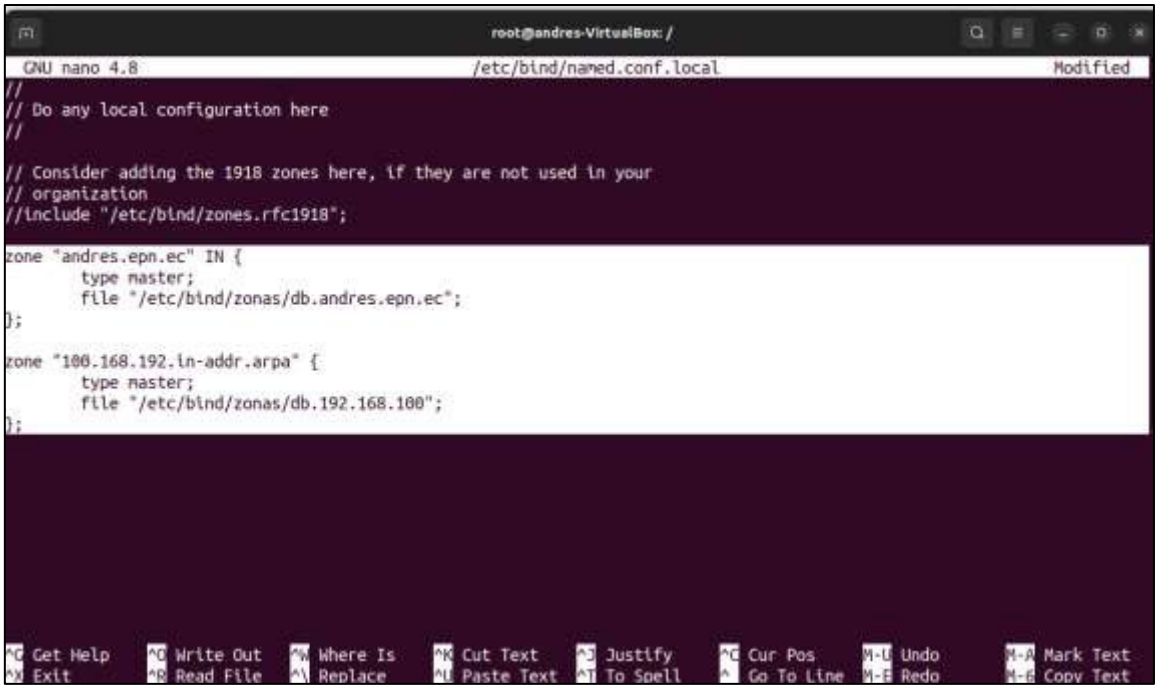


**Figura 3.37** Configuración del archivo **named.conf.local**

Es momento de crear el directorio donde se guardarán los archivos de zonas. Para esto se usa el comando **mkdir /etc/bind/zonas**, como se ve en la Figura 3.38.

```
root@andres-VirtualBox:/# mkdir /etc/bind/zonas
```

**Figura 3.38** Creación del directorio **zonas**

Dentro del directorio se crea el archivo de zona directa nombrado **db.andres.epn.ec**, como se indica en la figura 3.39. Posteriormente, al archivo se le añaden las líneas de comando que se muestran en la Figura 3.40. Dentro del archivo es importante destacar que las direcciones están compuestas por el nombre del host y el dominio utilizado para el ejemplo. Los tiempos de refresco son importantes para el correcto funcionamiento del servidor. Estos datos deben escribirse de forma correcta para que el servidor se levante sin ningún inconveniente [28].

```
root@andres-VirtualBox:/# nano /etc/bind/zonas/db.andres.epn.ec
```

**Figura 3.39** Creación del archivo de zona directa

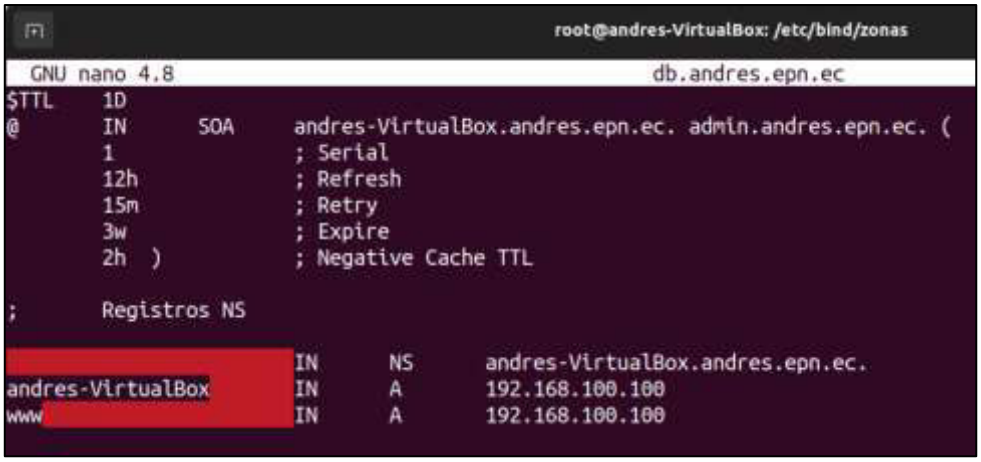


**Figura 3.40** Configuración del archivo de zona directa [28]

De igual forma, dentro del mismo directorio, se crea el archivo de zona inversa nombrado **db.192.168.100**, como se ve en la Figura 3.41. Posteriormente, se añaden las líneas de comando mostradas en la Figura 3.42, donde nuevamente se configuran los dominios, los tiempos de refresco y las direcciones necesarias para el ejemplo [28].


**Figura 3.41** Creación del archivo de zona inversa

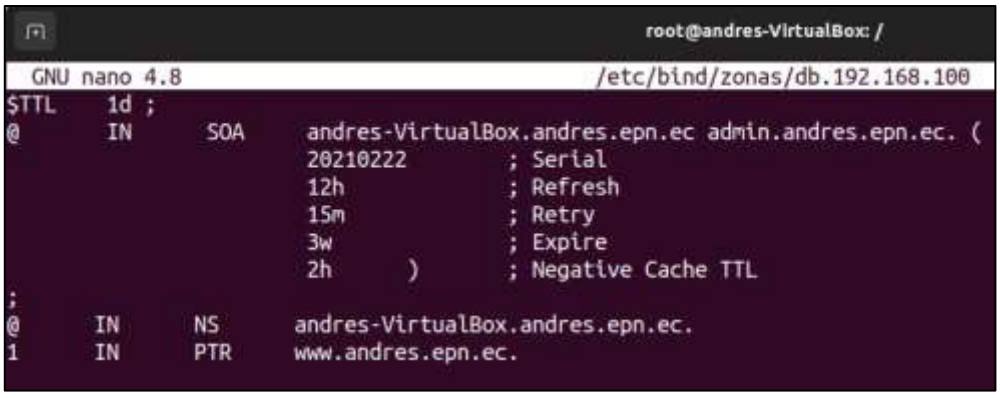


**Figura 3.42** Configuración del archivo de zona inversa[28]

Es importante aclarar que, cualquier signo, letra o palabra mal escrita, provocará que el servidor no arranque. Por esa razón, las configuraciones de cada archivo deben verificarse con dos comandos:

**named-checkzone andres.epn.ec /etc/bind/zonas/db.andres.epn.ec** para el archivo de zona directa y **named-checkzone db.100.168.192.in-addr.arpa /etc/bind/zonas/db.192.168.100** para el archivo de zona inversa. Cuando los comandos se resuelven se muestra un **OK** si el archivo está bien configurado, como se observa en la Figura 3.43. Si el archivo no estuviera bien configurado se mostraría un error [28].

```
root@andres-VirtualBox:/etc/bind/zonas# named-checkzone andres.epn.ec /etc/bind/zonas/db.andres.epn.ec
zone andres.epn.ec/IN: loaded serial 1
OK
root@andres-VirtualBox:/etc/bind/zonas# named-checkzone db.100.168.192.in-addr.arpa /etc/bind/zonas/db.192.168.100
zone db.100.168.192.in-addr.arpa/IN: loaded serial 20210222
OK
root@andres-VirtualBox:/etc/bind/zonas#
```

**Figura 3.43** Verificación de archivos de zona directa e inversa

Una vez comprobadas las configuraciones y sin tener errores, se procede a reiniciar el servicio DNS con el comando **service named restart** y comprobar el estado del servicio con el comando **service named status**, como se observa en la Figura 3.44.

```
root@andres-VirtualBox:/# service named restart
 * Stopping domain name service... named
waiting for pid 1115 to die
                                                                    [ OK ]
 * Starting domain name service... named                           [ OK ]
root@andres-VirtualBox:/# service named status
 * bind is running
root@andres-VirtualBox:/#
```

**Figura 3.44** Reinicio y verificación del servicio DNS

Con el servicio levantado, se pueden realizar pruebas de varias formas. La primera es mediante el comando **nslookup** más una dirección IP o dominio, como se muestra en la Figura 3.45, donde se usa el dominio de Google y el servidor resuelve sus respectivas direcciones IP.

```
root@andres-VirtualBox:/etc/bind/zonas# nslookup google.com
Server:         192.168.100.1
Address:        192.168.100.1#53

Non-authoritative answer:
Name:   google.com
Address: 142.250.78.142
Name:   google.com
Address: 2800:3f0:4005:407::200e
```

**Figura 3.45** Comprobación del servicio DNS con nslookup a google.com

### Implementación del servidor WEB

El proceso de implementación se guía por el diagrama descrito en la Figura 3.46.

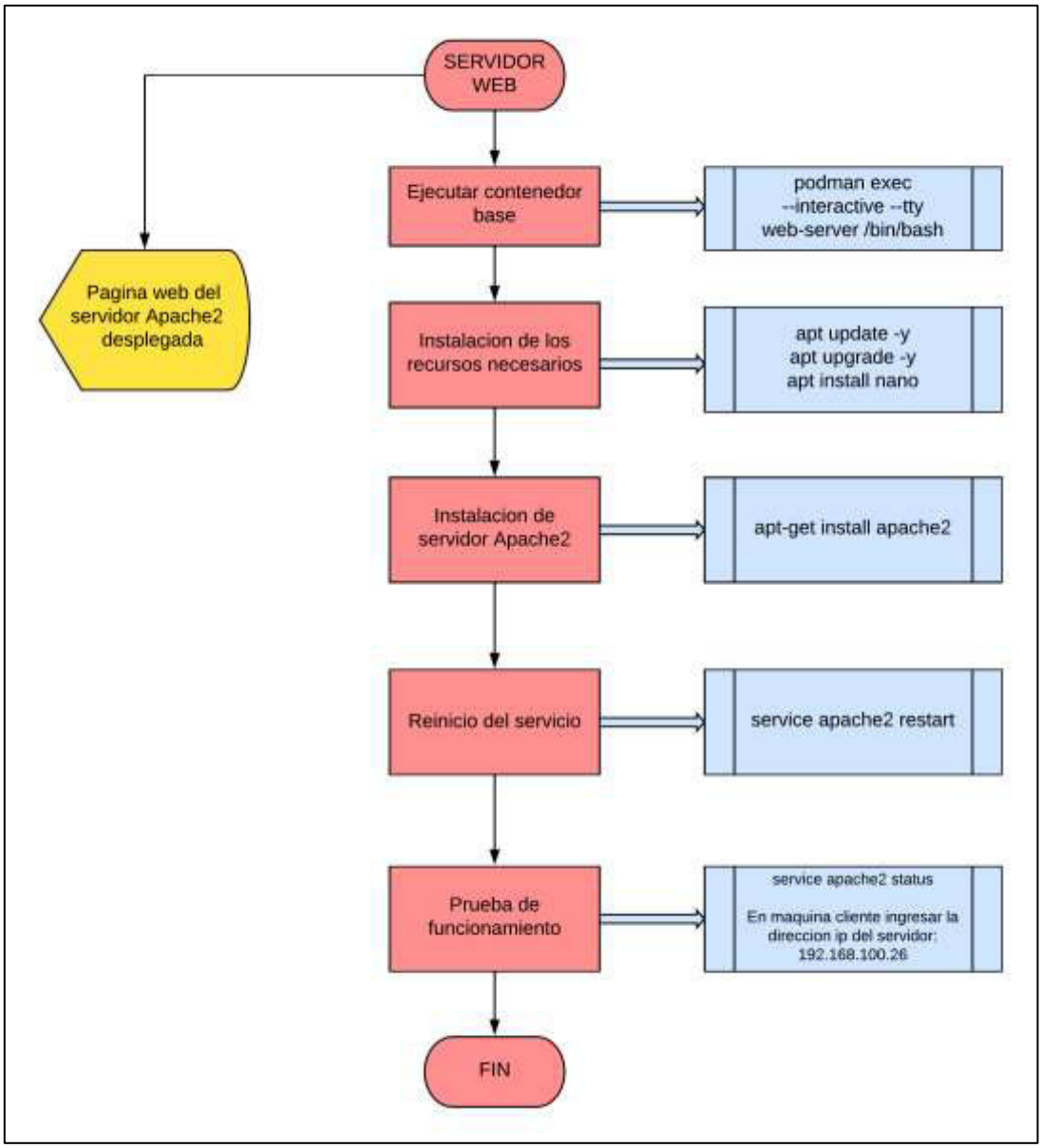


**Figura 3.46** Diagrama de Flujo para la implementación del servidor WEB

En primer lugar, se ejecuta el contenedor creado para el servidor WEB. Para esto se usa el comando **podman start web-server** y luego el comando **podman exec –interactive –tty web-server /binbash**, como se observa en la Figura 3.47. Los parámetros son los mismos utilizados en los anteriores servidores y son explicados en la Tabla 3.2.

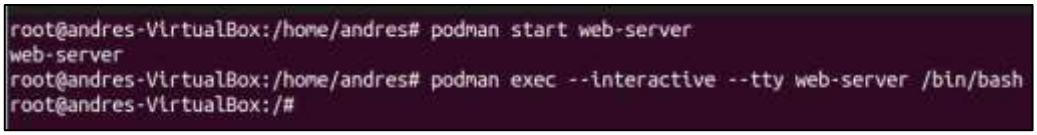


**Figura 3.47** Ejecución del contenedor de servidor WEB

Luego, se instala el servidor web *Apache2* con el comando **apt-get install apache2**, como se ve en la Figura 3.48. Durante la instalación se pedirán datos de ubicación y zona horaria, que se llenan según el caso [29].

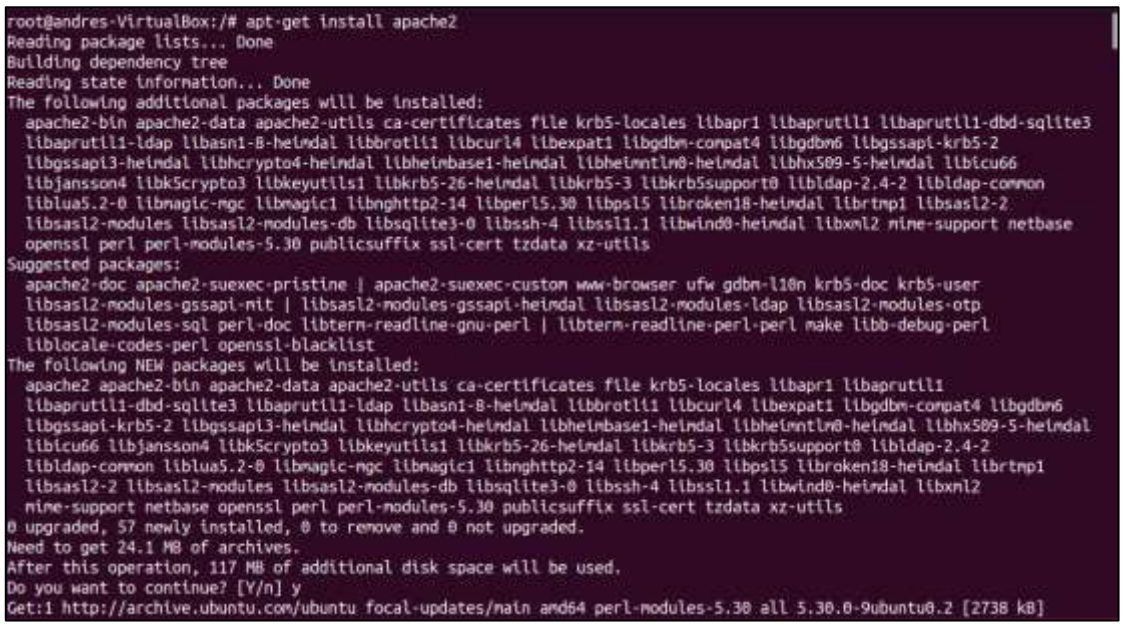


**Figura 3.48** Instalación del servidor web *Apahe2* [29]

Una vez que el servidor ha sido instalado, se debe hacer un reinicio del servicio y comprobar su estado con los comandos **service apche2 restart** y **service apache2 status,** como se observa en la Figura 3.49 [29].


**Figura 3.49** Reinicio y verificación del servicio WEB

Una vez levantado el servicio, se puede comprobar su funcionamiento en la máquina local ingresando al navegador y, en la barra de búsqueda, escribir *localhost,* como se muestra en la Figura 3.50. En una máquina cliente se lo puede hacer accediendo mediante la dirección IP del servidor [29].


**Figura 3.50** Comprobación el servicio WEB

## 3.4 Verificación del funcionamiento de los tres contenedores

En la presente sección se realiza la comprobación del funcionamiento de todos los servidores anteriormente levantados. Para validar el funcionamiento de cada uno de los servidores se tendrá en cuenta el funcionamiento en la máquina *host* del contenedor y, adicionalmente, en una máquina cliente con similares características configurada en la misma red de la máquina servidor.

La red de la máquina virtual *host* y la cliente están configuradas como adaptador tipo puente (*bridge*) en *VirtualBox,* tal como se observa en la Figura 3.51.

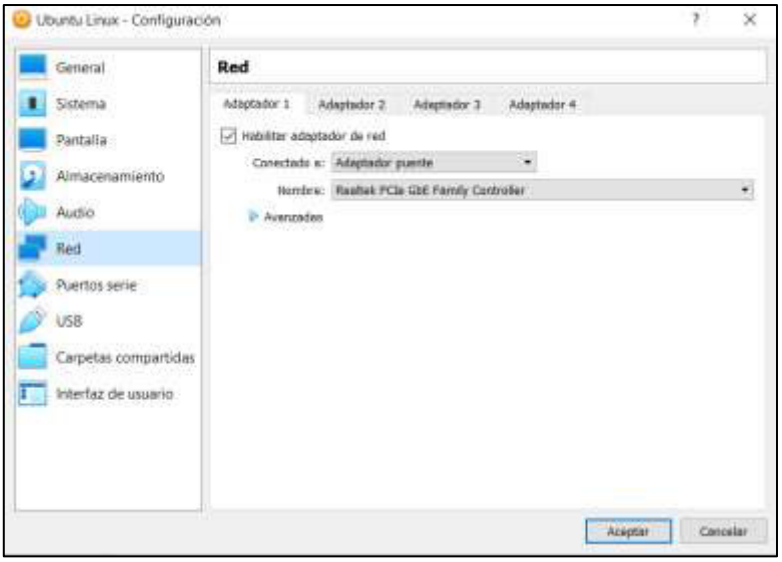


**Figura 3.51** Configuración de red de las máquinas virtuales

Todos los contenedores han sido implementados en una misma máquina *host* y comparten las características de red de la misma, por ende, cada servicio comparte la misma dirección IP, la cual se observa en la Figura 3.52.

```
root@andres-VirtualBox:/home/andres# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:4d:7d:99 brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.100/24 brd 192.168.100.255 scope global noprefixroute enp0s3
       valid_lft forever preferred_lft forever
    inet6 2800:bf0:3:342:5b44:2401:30f3:de4c/64 scope global temporary dynamic
       valid_lft 577061sec preferred_lft 58355sec
    inet6 2800:bf0:3:342:198d:75f8:13b6:48b/64 scope global dynamic mngtmpaddr noprefixroute
       valid_lft 1209196sec preferred_lft 604396sec
    inet6 fe80::b794:3fe0:600e:6398/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
root@andres-VirtualBox:/home/andres#
```

**Figura 3.52** Configuración de red de la máquina *host* de todos los contenedores

### Servidor DHCP

Con el servicio DHCP levantado y activo, se procede a ejecutar una máquina virtual cliente. Para la presente comprobación se hace uso de una máquina virtual Ubuntu 16, en la cual, en primer lugar, se debe cambiar la configuración de red para obtener direcciones mediante DHCP como se puede apreciar en la Figura 3.53.


**Figura 3.53** Configuración del método de conexión automático DHCP

Una vez configurado el método de conexión automático DHCP se procede a abrir una terminal y digitar el comando **dhclient -r -v.** El comando **dhclient** facilita un medio para configurar uno o más enlaces de red utilizando el protocolo de configuración de huésped dinámico. Además, los parámetros utilizados son:

- **-r:** Obliga al sistema desvincularse del servidor en uso y buscar uno nuevo.
- **-v:** Habilita los logs que permiten visualizar las peticiones del cliente.

Con el comando ejecutado, se obtiene un resultado como el de la Figura 3.54, que muestra la solicitud realizada al servidor de la interfaz **enp0s3**.

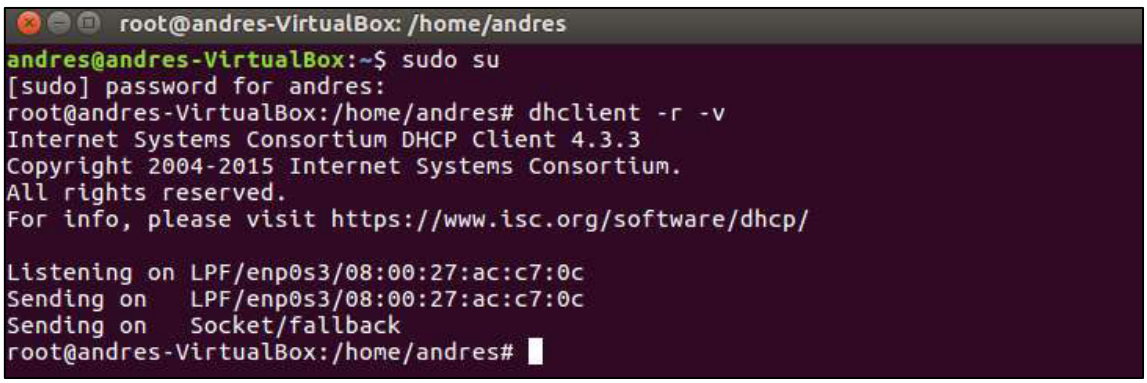


**Figura 3.54** Ejecución del comando **dhclient**

Ahora, se puede comprobar la dirección asignada con el comando **ifconfig** o **ip a**, como se puede ver en la Figura 3.55. La dirección IP asignada se resalta en color blanco.

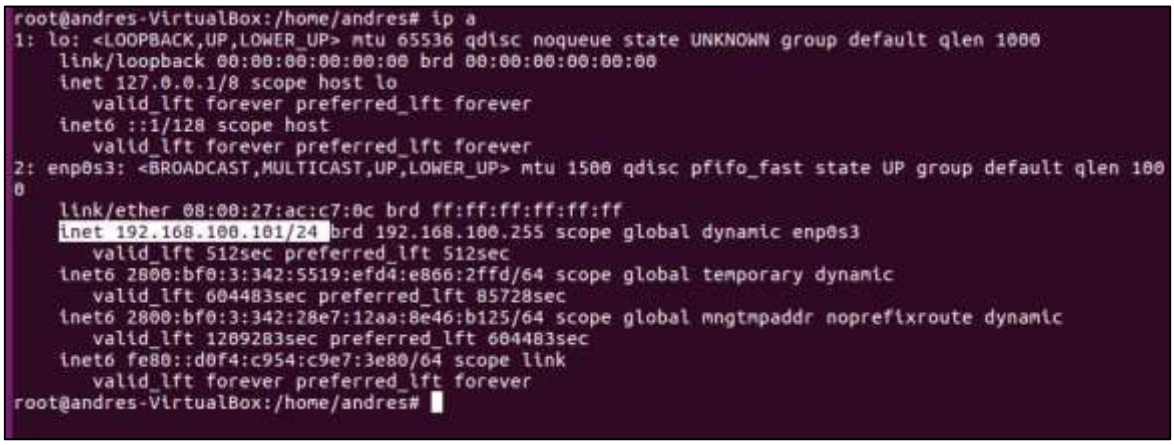


**Figura 3.55** Comprobación de la dirección IP asignada en el cliente

La interfaz del servidor es la **enp0s3**, además, el rango de direcciones configurado fue desde la 192.168.100.100 hasta la 192.168.100.200. En este caso se asignó la dirección 192.168.100.101.

Para comprobar que el servidor dentro del contenedor es el que está proporcionando las direcciones, se procede a consultar el estado del servidor y, como se puede apreciar en la Figura 3.56, se despliega el proceso de solicitud del cliente y la asignación de la dirección que proporciona el servidor.

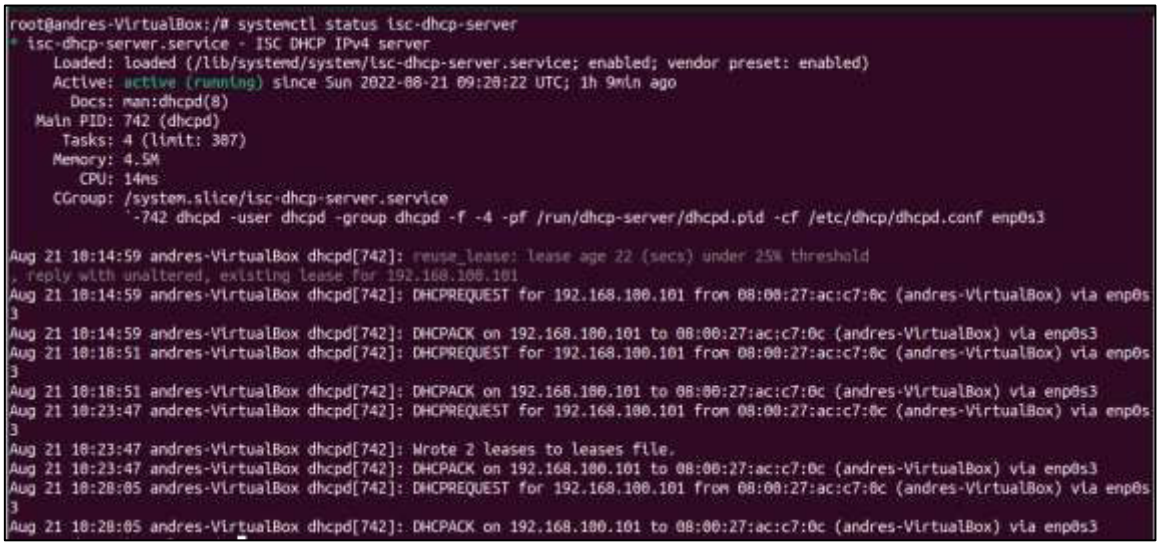


**Figura 3.56** Comprobación de la dirección IP asignada en el servidor

### Servidor DNS

Antes de las pruebas de funcionamiento, en todas las máquinas clientes del servidor y en el mismo contenedor se debe configurar el archivo /**etc/resolv.conf**, en el que se debe agregar la línea de comando **nameserver 192.168.100.100**, como se aprecia en la Figura 3.57, la línea agregada se resalta en color blanco. El comando permite que los clientes puedan direccionarse al servidor creado.

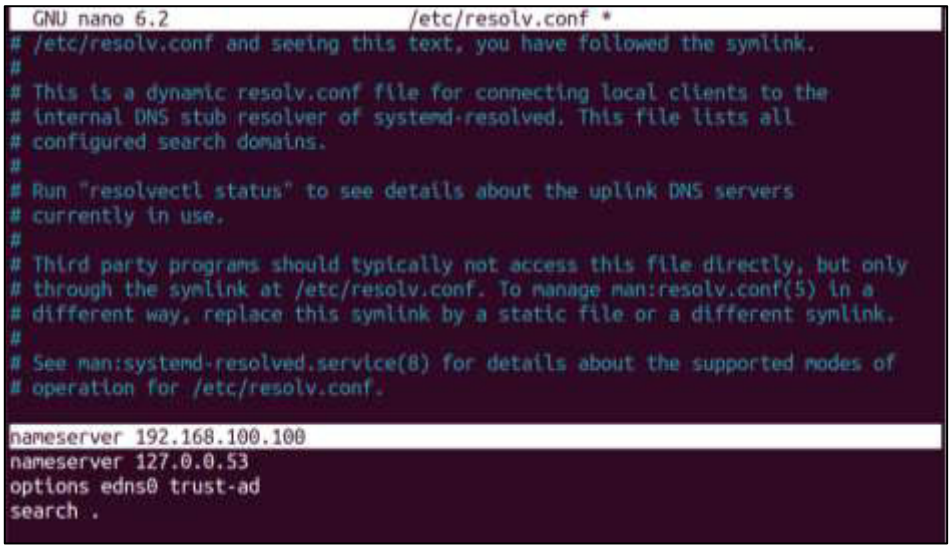


**Figura 3.57** Configuración del archivo **resolv.conf**

Se comprobará el funcionamiento del servidor con el ejemplo creado. En primer lugar, se procede con el comando **nslookup** y la dirección antes configurada y, como se ve en la Figura 3.58, al buscar mediante el dominio **www.andres.epn.ec** se despliega la dirección IP correspondiente.

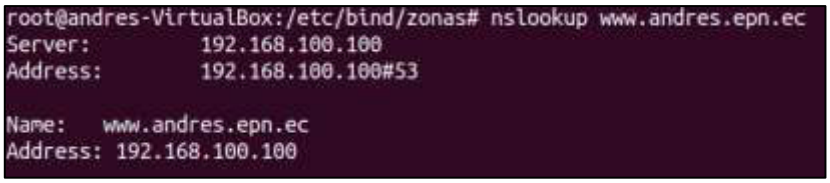


**Figura 3.58** Comprobación del servicio DNS con nslookup a **www.andres.epn.ec**

La segunda comprobación del servicio fuera del entorno del contenedor es abriendo una terminal extra y hacer un ping al dominio creado como ejemplo. Esto se puede ver en la Figura 3.59, si se obtiene una respuesta, significa que el servidor trabaja correctamente.

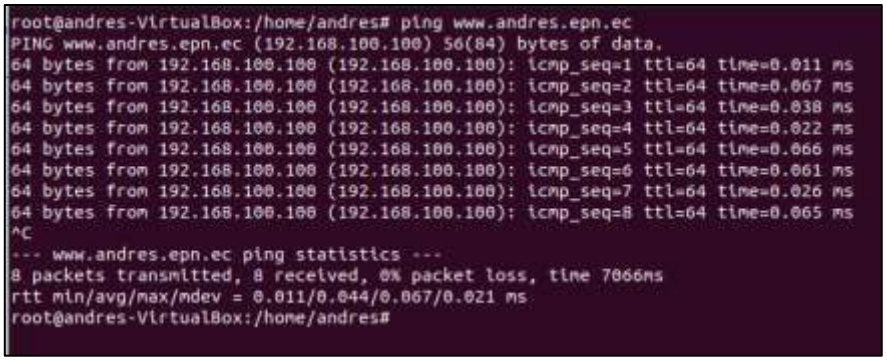


**Figura 3.59** Comprobación del servicio fuera del contenedor con un ping

Finalmente, se hace la comprobación con una máquina cliente. Para esto se usa una máquina virtual con *Ubuntu Linux* 16.04 con características similares a la máquina *host,* como se ve en la Figura 3.60.

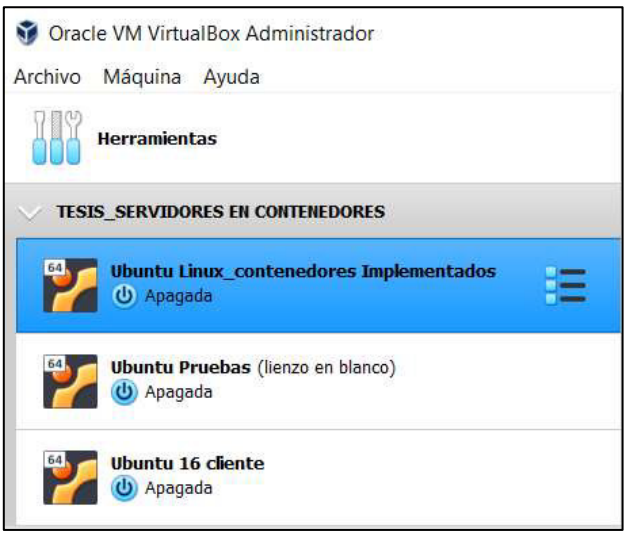


**Figura 3.60** Máquinas virtuales disponibles para pruebas

En la máquina cliente se realiza un ping a la dirección **www.andres.epn.ec**, desde la maquina cliente con dirección **192.168.100.150** dado que se da una respuesta desde el servidor con dirección **192.168.100.100** como se muestra en la Figura 3.61, se puede comprobar que el servidor está funcionando correctamente.

```
root@andres-VirtualBox:/home/andres# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:ac:c7:0c brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.150/24 brd 192.168.100.255 scope global enp0s3
       valid_lft forever preferred_lft forever
    inet6 2800:bf0:3:342:28e7:12aa:8e46:b125/64 scope global noprefixroute dynamic
       valid_lft 1209430sec preferred_lft 604630sec
    inet6 fe80::d0f4:c954:c9e7:3e80/64 scope link
       valid_lft forever preferred_lft forever
root@andres-VirtualBox:/home/andres# ping www.andres.epn.ec
PING www.andres.epn.ec (192.168.100.100) 56(84) bytes of data.
64 bytes from 192.168.100.100: icmp_seq=1 ttl=64 time=0.285 ms
64 bytes from 192.168.100.100: icmp_seq=2 ttl=64 time=0.341 ms
64 bytes from 192.168.100.100: icmp_seq=3 ttl=64 time=0.331 ms
64 bytes from 192.168.100.100: icmp_seq=4 ttl=64 time=0.336 ms
64 bytes from 192.168.100.100: icmp_seq=5 ttl=64 time=0.225 ms
64 bytes from 192.168.100.100: icmp_seq=6 ttl=64 time=0.334 ms
64 bytes from 192.168.100.100: icmp_seq=7 ttl=64 time=0.346 ms
^C
--- www.andres.epn.ec ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6127ms
rtt min/avg/max/mdev = 0.225/0.314/0.346/0.040 ms
root@andres-VirtualBox:/home/andres#
```

**Figura 3.61** Comprobación de servicio en maquina cliente

### Servidor WEB

Para comprobar el funcionamiento del servidor web se procede de dos formas, la primera es de manera local, abriendo el navegador de la maquina *host* del contenedor y digitando *localhost* en el buscador, como puede apreciar en la Figura 3.62.

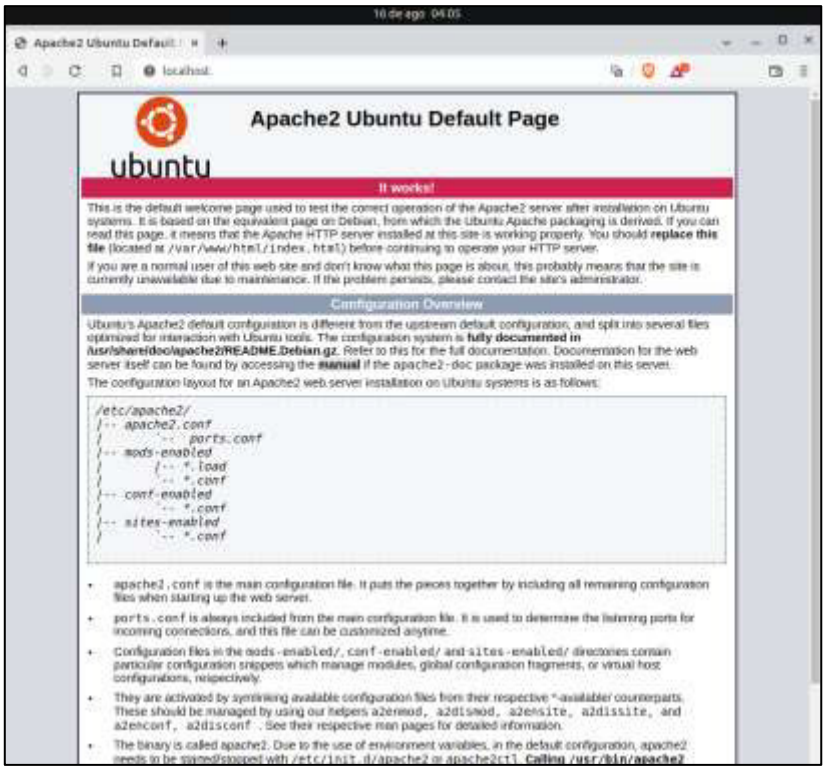


**Figura 3.62** Comprobación de servicio web en host

La segunda forma de comprobar el servidor web es con la máquina cliente *Ubuntu Linux* 16.04. Igualmente abriendo el navegador y digitando la dirección IP del servidor en el buscador, como se ve en la figura 3.63.

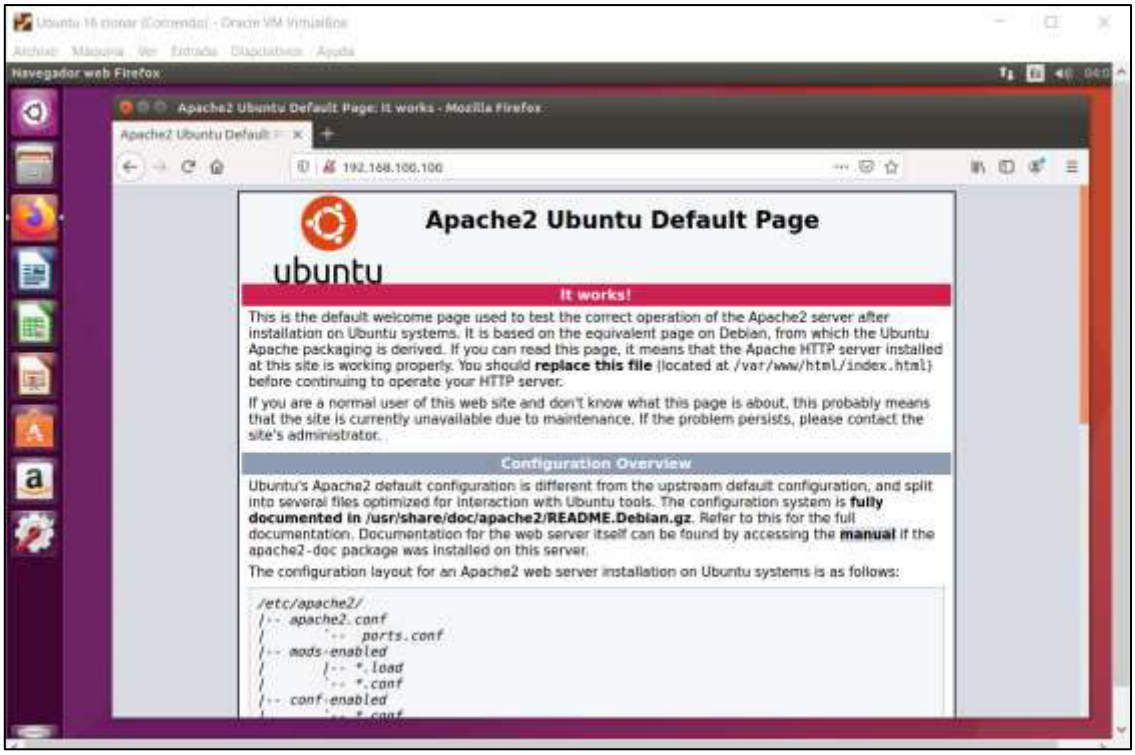


**Figura 3.63** Comprobación de servicio WEB en máquina cliente

### Uso de recurso de los contenedores

Finalmente, se puede analizar el tamaño en memoria y el uso de procesador que tiene cada uno de los contenedores con sus servicios levantados y funcionando. En comparación a la máquina virtual donde se realizó toda la implementación donde el uso de procesador en la maquina *host* con una carga de trabajo mínima es de entre 10% al 14%, el uso de memoria RAM es de entre 65 MB a 70 MB además de que ocupa un espacio en el disco duro de 14.37 GB de los 20GB reservados desde un principio, cómo se puede ver en la Figuras 3.64 y 3.65.

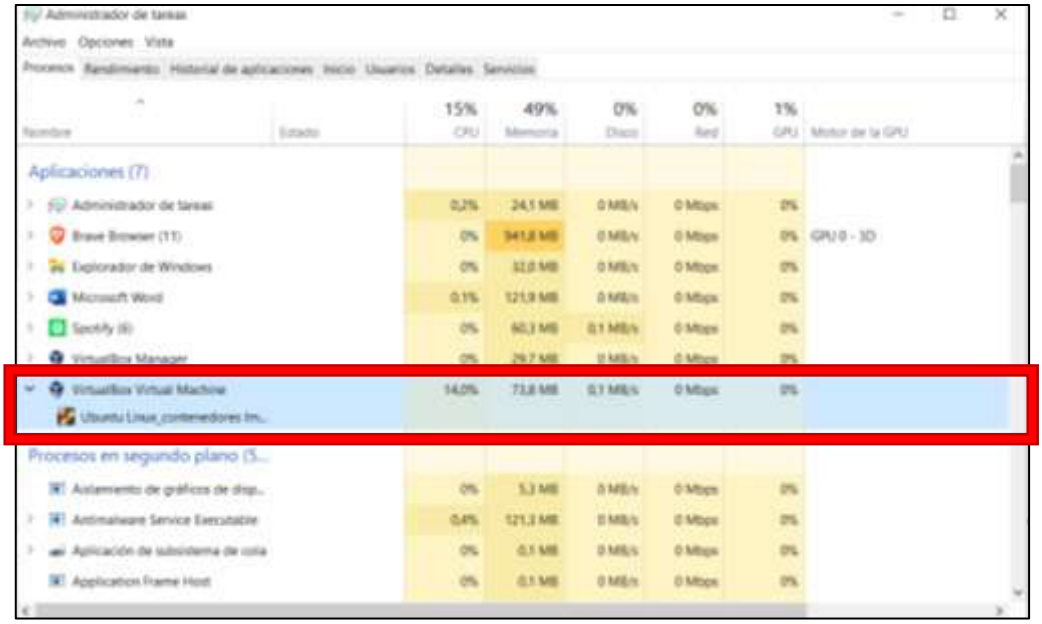

**Figura 3.64** Uso de recursos de la máquina virtual usada para la implementación

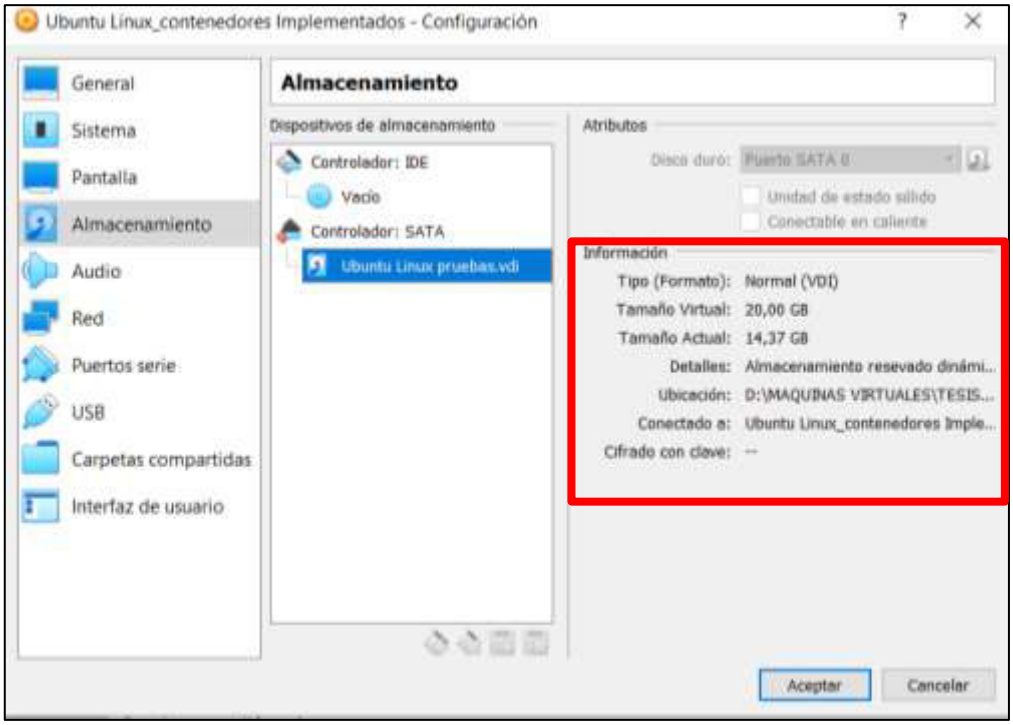


**Figura 3.65** Tamaño en memoria de la máquina virtual usada para la implementación

Los contenedores presentan un uso más eficiente de recursos como se puede ver en las Figuras 3.66, 3.67 y 3.68, el tamaño de todos los contenedores no sobrepasa los 85 Mb, y tampoco superan el 5% de uso del procesador, lo que ratifica que trabajar con contenedores supone un ahorro en recursos de *software* y *hardware*.


**Figura 3.66** Uso de recursos de contenedor DHCP


**Figura 3.67** Uso de recursos de contenedor DNS


**Figura 3.68** Uso de recursos de contenedor WEB

# 4 CONCLUSIONES

- En el presente trabajo cada uno de los servicios implementados dentro de los contenedores fue armado y estructurado desde cero, con el fin de mostrar el paso a paso de las configuraciones necesarias y pertinentes en cada servidor. Dentro del entorno y la comunidad de desarrollo de herramientas de DevOps ya se encuentran imágenes enfocadas al uso en contenedores y que ofrecen múltiples servicios. Estos suelen encontrarse ya montados y desarrollados a medida para necesidades específicas. Incluso los servidores montados en el presente trabajo se pueden encontrar de esta forma, sin embargo, es importante conocer las estructuras teóricas de cada herramienta de desarrollo que se utiliza. Por otro lado, se debe señalar que aprovechar las herramientas desarrolladas por la comunidad pude ser de gran utilidad debido a que son soluciones rápidas y eficientes en el entorno actual de tecnologías de desarrollo.
- El ejercicio de montar servidores con Podman permitió comprobar una de las grandes ventajas que tiene este motor de contenedores. Esta radica principalmente en que no es necesario dar permisos *root* a todos los contenedores creados. Desde esta perspectiva se encuentra ya un atributo respecto a lo que a seguridad se refiere. Sin embargo, existen casos en los que se debe necesariamente dar permisos *root,* esto se pudo apreciar específicamente en el servidor de DHCP, el cual ni siquiera llega a levantarse si no tiene tales permisos. En estas situaciones se debe tomar en cuenta que existe un hueco de seguridad con dicho servidor, y buscar alternativas para solventar el inconveniente.
- El tamaño en memoria que tienen los contenedores con los servidores ya implementados demuestra otro de los puntos fuertes por lo que esta tecnología se popularizó y es que resulta sumamente eficiente trabajar con aplicaciones finales que pesan unos cuantos megabytes y tienen una carga mínima de procesador. Haciendo una comparación rápida, la máquina virtual en la cual se realizó toda la implementación pesa alrededor de 10GB mientras que el contenedor más pesado tiene un máximo de 100MB. Esto considerando que se usa la misma lógica de trabajo, es decir, un sistema Ubuntu Linux corriendo tanto en la maquina como en el contenedor, lo que ejemplifica claramente el nivel eficiencia que se consigue.
- Aunque se mencionó que *Podman* presenta una gran compatibilidad con *Docker*, es importante tener en cuenta que cuando se presenta un problema específico

de *Podman*, las fuentes de información para solucionarlo aún son limitadas, y mayormente se encuentran en otros idiomas, lo que para usuarios novatos o en un nivel básico puede resultar una barrera para su correcto aprendizaje.

# 5 RECOMENDACIONES

- El campo de desarrollo de contenedores está sumamente avanzado hoy en día, teniendo herramientas y tecnologías ya establecidas en este campo como *Docker* es importante aprovechar toda la información obtenida a lo largo de los años. Ya que como se pudo observar a lo largo del desarrollo del presente trabajo muchas de las herramientas como las imágenes usadas para montar los servidores, fueron originalmente desarrolladas para Docker, y como éstas existen un sin fin de aplicaciones más. Teniendo en cuenta que Podman es compatible con comandos *Docker* se puede tomar ventaja de ambas herramientas por igual.
- *Podman* al ser una herramienta relativamente nueva suele presentar problemas en ambientes diferentes de *Linux*, un ejemplo claro fue que al usar la herramienta de *Linux* para Windows que se encuentra disponible en la *Microsoft Store*, *Podman* presento varios problemas de incompatibilidad con el *kernel* de esa aplicación. Por tanto, no es una opción viable si se necesita montar servicios importantes, otro ejemplo se puede observar cuando se usan distribuciones de *Linux* no estables en este caso también se presentan problemas de incompatibilidad. La mejor forma de trabajar con Podman es usando entornos en los que se tenga un soporte total a la aplicación.
- La implementación realizada en el presente trabajo de titulación requirió el uso de máquinas virtuales con sistemas operativos invitados para realizar el despliegue de los contenedores, con el objetivo de presentar y demostrar el trabajo de una forma más práctica, pero, no es una forma eficiente de trabajar con este tipo de tecnología ya que el objetivo de la misma es evitar el uso innecesario de recursos. Para un uso más práctico de este tipo de *software* dentro de un sistema DevOps es recomendable usar plataformas que permitan desplegar contenedores de forma nativa.
- Los contenedores aun presentan ciertos aspectos que no están del todo solucionados en cuanto a servidores se refiere, en el caso de los servidores WEB

y DNS no se presenta mayor problema incluso tienen imágenes listas para su despliegue y configuración, pero en el caso del servidor DHCP aún se requiere proporcionar accesos privilegiados tanto a la red como con el sistema operativo para que funcione de forma correcta, por lo tanto, en el caso de este último servidor sigue siendo recomendable su despliegue de formas más tradicionales.

# 6 REFERENCIAS BIBLIOGRÁFICAS


[1] “Oracle VM VirtualBox.” https://www.virtualbox.org/ (accessed Sep. 08, 2022).

[2] “El concepto de los contenedores.” https://www.redhat.com/es/topics/containers (accessed May 14, 2022).

[3] “LXC: ¿qué son los contenedores de Linux y cómo funcionan? - IONOS.” https://www.ionos.es/digitalguide/servidores/know-how/que-es-lxc/ (accessed Sep. 08, 2022).

[4] “Contenedores Linux.” https://www.teamnet.com.mx/blog/contenedores-linux (accessed Sep. 08, 2022).

[5] “What is a Container? - Docker.” https://www.docker.com/resources/what-container/ (accessed Sep. 08, 2022).

[6] “What is podman?” https://podman.io/whatis.html (accessed May 14, 2022).

[7] “GitHub - containers/podman: Podman: A tool for managing OCI containers and pods.” https://github.com/containers/podman (accessed May 13, 2022).

[8] “¿Qué es un pod de Kubernetes?” https://www.redhat.com/es/topics/containers/what-is-kubernetes-pod (accessed May 20, 2022).

[9] “Podman: Managing pods and containers in a local container runtime | Red Hat Developer.” https://developers.redhat.com/blog/2019/01/15/podman-managing-containers-pods#podman_pods__what_you_need_to_know (accessed May 14, 2022).

[10] “What is a DHCP Server? | Learn What They Are & How They Work | Infoblox.” https://www.infoblox.com/glossary/dhcp-server/ (accessed May 14, 2022).

[11] “Qué es un Servidor DHCP | OpenWebinars.” https://openwebinars.net/blog/que-es-un-servidor-dhcp/ (accessed Sep. 08, 2022).

[12] “¿Qué es DNS? | Nociones básicas | Cloudflare | Cloudflare.” https://www.cloudflare.com/es-es/learning/dns/what-is-dns/ (accessed May 14, 2022).

[13] “¿Qué es Servidor web? - Definición en WhatIs.com.” https://www.computerweekly.com/es/definicion/Servidor-web (accessed May 14, 2022).

[14] “Podman vs Docker: What are the differences?” https://www.imaginarycloud.com/blog/podman-vs-docker/ (accessed May 14, 2022).

[15] “GitHub - containers/podman: Podman: A tool for managing OCI containers and pods.” https://github.com/containers/podman (accessed May 14, 2022).

[16] “What is Podman? Docker’s competitor, current container master.” https://pandorafms.com/blog/what-is-podman/ (accessed Sep. 07, 2022).

[17] “CRIU.” https://criu.org/Main_Page (accessed Sep. 07, 2022).

[18] “slirp4netns — How does it work. slirp4netns provides user-mode… | by M Castelino | Medium.” https://mcastelino.medium.com/slirp4netns-how-does-it-work-5c0bd31200ce (accessed Sep. 07, 2022).

[19] “GitHub - containers/netavark: Container network stack.” https://github.com/containers/netavark (accessed Sep. 07, 2022).

[20] “How to Install and Use Podman on Ubuntu 20.04 | Atlantic.Net.” https://www.atlantic.net/dedicated-server-hosting/how-to-install-and-use-podman-on-ubuntu-20-04/ (accessed Jun. 28, 2022).

[21] “The GNU Privacy Guard.” https://www.gnupg.org/ (accessed Jun. 28, 2022).

[22] “Qué es Docker y para qué sirven los contenedores.” https://www.ondho.com/que-es-docker-para-que-sirve/ (accessed Jul. 05, 2022).

[23] “Ubuntu - Official Image | Docker Hub.” https://hub.docker.com/_/ubuntu (accessed Jul. 05, 2022).

[24] “systemd (Español) - ArchWiki.” https://wiki.archlinux.org/title/systemd_(Espa%C3%B1ol) (accessed Aug. 20, 2022).

[25] “podman-run — Podman documentation.” https://docs.podman.io/en/latest/markdown/podman-run.1.html (accessed Aug. 20, 2022).

[26] “podman-exec — Podman documentation.” https://docs.podman.io/en/latest/markdown/podman-exec.1.html (accessed Aug. 20, 2022).

[27] “A Step-by-Step Guide to Set up a DHCP Server on Ubuntu - LinuxForDevices.” https://www.linuxfordevices.com/tutorials/ubuntu/dhcp-server-on-ubuntu (accessed Aug. 20, 2022).

[28] “Instalar y configurar servidor DNS con Bind9 en Ubuntu 20.04 – NETWORLD.” https://www.netntw.com/archivos/533?utm_source=NETWORLD&utm_medium=articulo&utm_campaign=aprendizaje (accessed Aug. 15, 2022).

[29] “Cómo instalar el servidor web Apache en Ubuntu 20.04 | DigitalOcean.” https://www.digitalocean.com/community/tutorials/how-to-install-the-apache-web-server-on-ubuntu-20-04-es (accessed Aug. 15, 2022).

# 7 ANEXOS

## ANEXO I: Certificado de Originalidad

**CERTIFICADO DE ORIGINALIDAD**

Quito, D.M. 13 de septiembre de 2022

De mi consideración:

Yo, **JAVIER ALEJANDRO ARMAS NAVARRETE**, en calidad de Director del Trabajo de Integración Curricular titulado **IMPLEMENTACIÓN DE SERVIDORES MEDIANTE PODMAN** asociado al proyecto **IMPLEMENTACIÓN DE CONTENEDORES CON SERVIDORES MEDIANTE DEVOPS**, elaborado por el estudiante **ANDRÉS ISAAC GUALOTUÑA LLUMIQUINGA** de la carrera **TECNOLOGÍA SUPERIOR EN REDES Y TELECOMUNICACIONES**, certifico que he empleado la herramienta Turnitin para la revisión de originalidad del documento escrito completo, producto del Trabajo de Integración Curricular indicado.

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

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

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

LINK: Informe Turnitin Completo - TIC - Andrés Gualotuña.pdf

Atentamente,

JAVIER ALEJANDRO ARMAS NAVARRETE

Docente

Escuela de Formación de Tecnólogos

# ANEXO II: Enlaces

**Anexo II.I** Código QR - Video Implementación Completa

Código QR - Video Pruebas de Funcionamiento

# ANEXO III:  Archivos de Configuración

**Anexo IIII.I** Código QR de los archivos de configuración para los servidores DHCP y DNS