Diagrama

El contenido generado por IA puede ser incorrecto.d

I.E.S. “ÁGORA”

Dpto.: Informática

Ciclo Formativo de G. Superior: Administración de Sistemas Informáticos y Redes

Automatización de la gestión de entornos virtuales: Implantación de infraestructura con Ansible y Vagrant

Imagen que contiene Icono

El contenido generado por IA puede ser incorrecto.

Autor o autores: Carla Galindo Corchero

Tutor: Maria Francisca Roncero Holgado

Cáceres, a junio de 2025

**ÍNDICE:**

**1.** **FICHA DE PROYECTO** 3

**2.** **INTRODUCCIÓN** 4

**3.** **ESTUDIO DE VIABILIDAD** 4

**3.1** **ANÁLISIS DAFO** 4

**3.2** **ANÁLISIS CAME** 5

**4.** **DESARROLLO TÉCNICO DEL PROYECTO** 6

**4.1 ANÁLISIS DE REQUISITOS Y DISEÑO** 7

**4.1.1** **REQUISITOS NECESARIOS PARA QUE SE PUEDA IMPLANTAR** 7

**4.1.2** **DISEÑO DE RED Y DE INFRAESTRUCTURA:** 8

**4.1.3** **SOLUCIONES PARA SU SEGURIDAD:** 9

**4.2 TECNOLOGÍAS A USAR** 9

**4.2.1 ANSIBLE** 11

**4.2.2 VAGRANT** 12

**4.2.3 VIRTUALBOX** 14

**4.2.4 COMBINACIÓN DE ANSIBLE, VAGRANT Y VIRTUALBOX** 14

**4.3 IMPLEMENTACIÓN** 16

**4.4 DOCUMENTACIÓN** 17

**4.5 ADMINISTRACIÓN** 17

**4.5.1 ADMINISTRACIÓN EN LA PREPARACIÓN** 18

**4.5.2 ADMINISTRACIÓN EN EL DESARROLLO** 18

**4.5.3 ADMINISTRACIÓN DE MANTENIMIENTO** 19

**4.6 DESPLIEGUE** 20

**6. CONCLUSIÓN** 22

**7. BIBLIOGRAFÍA** 22

**8. ANEXOS** 24

1. **FICHA DE PROYECTO**

La ficha del proyecto presentada para la propuesta del proyecto es la siguiente y consta de las siguientes partes:

* **TÍTULO DEL PROYECTO**: Automatización de la gestión de entornos virtuales: Implementación de infraestructura con Ansible y Vagrant usando VirtualBox.
* **DESCRIPCIÓN GENERAL**: Consiste en crear, configurar y gestionar máquinas virtuales alojadas en VirtualBox de forma automática con las herramientas de Ansible y Vagrant.
* **OBJETIVO:** El objetivo principal del proyecto será la investigación y una pequeña puesta en marcha del proceso de cómo sería crear máquinas virtuales con la herramienta de Vagrant en VirtualBox y después usar la herramienta de Ansible para automatizar la configuración y gestión de dichas máquinas.
* **JUSTIFICACIÓN:** Cada vez las empresas necesitan más máquinas virtuales para realizar pruebas de simulación a la hora de lanzar alguna aplicación o configuraciones nuevas y no quieren correr el riesgo de estropear la máquina anfitriona, pero cada vez que se quiere hacer eso se tiene que crear y configurar una máquina virtual y puede ser un proceso monótono, repetitivo y expuesto a más errores. Con este proyecto lo que se quiere conseguir es que se pueda optimizar todo esto y asegurar estabilidad y menos número de errores de manera eficiente y con gestión mejor de los recursos.
* **ASPECTOS PRINCIPALES A ABORDAR**: Uso de Vagrant para crear y gestionar las máquinas virtuales de VirtualBox, Uso de Ansible para configurar estas máquinas virtuales de VirtualBox, Simulación para las pruebas con VirtualBox, Optimización del tiempo y de los recursos que se vayan a usar, Configuración de accesos y permisos necesarios en las máquinas virtuales, Estrategias para mejorar esta seguridad y estabilidad de las máquinas virtuales.
* **MEDIOS QUE SE UTILIZAN**: Hardware: un portátil en el que simule la infraestructura cliente-servidor, Software: La virtualización de los entornos virtuales (VirtualBox), el programa para la gestión y automatización de las máquinas virtuales (Vagrant), el programa para la configuración de las máquinas virtuales (Ansible), Esquema de la topología de red e información de las conexiones usadas en formato simulado, el lenguaje de programación con el que trabaja Ansible (Phyton), software para controlar las versiones de los ficheros de configuración de Ansible y Vagrant y sistema de almacenamiento de información y enlaces con respecto a la investigación del proyecto (GitHub), redacción de documentación por programa adecuado para ello (Microsoft Word ), más herramientas que se usarán en profundidad...,Sistemas Operativos: Se usará Ubuntu Server 22.04 y clientes Ubuntu Xenial 16.04.
* **ÁREAS DE TRABAJO Y OTROS ELEMENTOS:** Creación e Implementación de servidores en máquinas virtuales y automatización de las configuraciones de Ubuntu, Configuración de redes y conexión de las máquinas virtuales, Instalación y configuración de los servicios de forma automática y su configuración de accesos y permisos, Control del rendimiento e implementación de configuraciones seguras.

1. **INTRODUCCIÓN**

Las empresas, hoy en día, cuentan con un parque informático muy numeroso y complejo de administrar, por ello, el departamento de informática tiene un peso muy importante y relevante dentro de las organizaciones empresariales.

Entre los miembros del departamento, los administradores de sistemas son los encargados de llevar a cabo las labores de gestión y administración de todo el equipamiento informático y, junto a ellos, los programas necesarios para facilitar esta tarea. Pero al usar estos programas o procesos un número elevado de veces, nos podemos encontrar con el problema de que la tarea para el administrador puede resultar un tanto monótona, con mayor indicio de errores, menos eficiente…

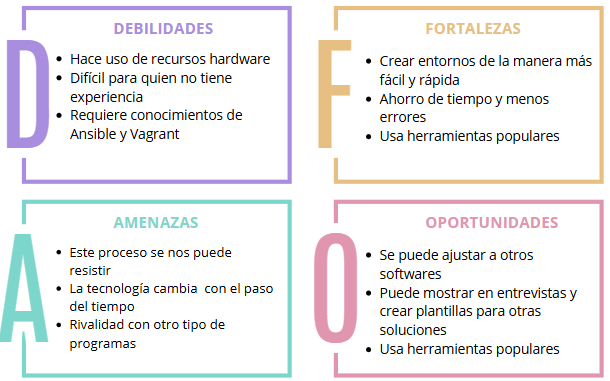
Para evitar esos problemas mencionados anteriormente, los administradores se apoyan en herramientas y software como son los mencionados como objeto de este proyecto: (Ansible y Vagrant junto a VirtualBox), los cuales ahora comentaremos ahora en brevedad para saber de lo que estamos hablando. Considero que es muy importante el uso de ellos para mejorar en dichos problemas.

1. **ESTUDIO DE VIABILIDAD**

En primer lugar, se realiza un estudio a nivel de viabilidad para comprobar si el proyecto y la automatización de tareas junto a la creación de entornos virtuales es una idea que merece la pena de realizar o, por el contrario, no resulta viable. Claramente, se puede dirigir a un público el cual necesite una gran cantidad de entornos virtuales de manera fácil y rápida o para automatizar otro tipo de tareas.

## **ANÁLISIS DAFO**

Para deducir su viabilidad, primero vamos a realizar un análisis DAFO. En un análisis DAFO se plantean 4 campos para comprobar todo esto: las fortalezas (son los puntos fuertes que diferencian el proyecto entre otros proyectos), las debilidades (son los puntos que se necesitan mejorar y que actúan como puntos de desventaja en comparación con otros proyectos), las oportunidades (son puntos que se pueden aprovechar en el proyecto para que tenga más posibilidades de que sea más viable), y las amenazas (son los requisitos externos que pueden afectar al proyecto en el presente o en un futuro). Su esquema gráfico con algunas ideas es el siguiente:

****

## **ANÁLISIS CAME**

Frente a este análisis DAFO, también podemos trabajar con las ideas planteadas para unificarlo a un análisis CAME. En este análisis CAME trabajaremos con los campos anteriores para mejorar su presencia corrigiendo las debilidades, afrontando las amenazas, manteniendo las fortalezas, y explotando las nuevas oportunidades. El esquema se muestra a continuación:

Escala de tiempo

El contenido generado por IA puede ser incorrecto.

Esta sería la idea del proyecto desde un enfoque más cuadrado y visual. Con respecto a los recursos hardware y software, hablaremos de ellos más adelante en el punto siguiente.

1. **DESARROLLO TÉCNICO DEL PROYECTO**

En este apartado se hablará de los diferentes puntos que se llevarán a cabo para entender el proceso de implantación técnica de esta automatización y creación de entornos virtuales. Se desarrollarán temas desde: los requisitos que se utilizarán, el diseño de la infraestructura de la red, las herramientas necesarias, el proceso de cómo se implementará todo, la documentación necesaria, las dichas tareas de administración, y el proceso de despliegue.

Todo este proceso es fundamental para asegurar el funcionamiento del proyecto y cumplir con los objetivos que se quieren alcanzar.

**4.1 ANÁLISIS DE REQUISITOS Y DISEÑO**

En este punto del proyecto, vamos a desarrollar diferentes subapartados para comprender mejor los requisitos que usa y el diseño creado.

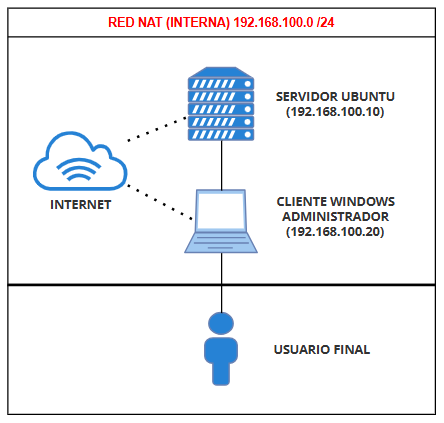
### **REQUISITOS NECESARIOS PARA QUE SE PUEDA IMPLANTAR**

Para que se pueda implantar este proyecto, necesitamos cumplir con una serie de requisitos para asegurarse el éxito a la hora de implantarlo y evitar sorpresas a posteriori. Algunos de los requisitos necesarios son:

* **Saber que entornos virtuales vamos a lanzar de manera automática**: en este proyecto se quieren lanzar entornos de Ubuntu, porque únicamente se va a trabajar con ese Sistema Operativo. No obstante, se podrían hacer de la misma manera con Windows sin ningún problema adicional.
* **Una vez lanzados, el proceso de configuración**: para configurar estos entornos, nos basaremos en el proceso práctico explicado en los anexos. El software necesario para la configuración va a ser Ansible junto a su automatización y para crearlos previamente se va a usar Vagrant. Las versiones de los sistemas operativos que van a tener las máquinas virtuales dependerán de la disponibilidad de recursos hardware de la máquina anfitriona, pero se intentará que sean la última versión. Las dependencias para poder llevar a cabo la automatización de tareas y la creación de los entornos virtuales serán paquetes de software como Python, también los servicios como SSH para realizar la conexión entre ambas máquinas, los roles para la realización de las tareas con Ansible (si son necesarios y se tengan que utilizar), los archivos de scripts y las conexiones de red con playbooks.
* **Que sea seguro, escalable y con buen rendimiento**: para que podamos gestionar el número de entornos virtuales generados y que no se desmorone, que su tiempo de creación sea el adecuado y no se alargue el tiempo, y que a la hora de realizar todo esto se haga de una manera segura para proteger todo en caso de intrusos y amenazas.
* **La compatibilidad de todas las herramientas**: asegurar que todas las herramientas, software, hardware, dependencias, etc… sean compatibles entre sí para que no haya ningún fallo y se pueda realizar el proyecto de manera segura.

### **DISEÑO DE RED Y DE INFRAESTRUCTURA:**

Para comprender un poco mejor la idea de la topología de red y de la infraestructura del proyecto, voy a presentar de manera gráfica ambas partes con su respectiva explicación para una mejor comprensión.

Este esquema representa la topología de red que se ha seguido. Consta de una Red NAT personalizada en VirtualBox (haciendo este de router) con la dirección de red 192.168.100.0 /24 y dándole acceso a internet a la máquina controladora, gracias a la conexión de la máquina anfitriona.

Un servidor con sistema operativo Ubuntu a la que se le ha asignado una dirección IP de manera manual con Netplan 192.168.100.10 /24. En esta se usará la herramienta de Ansible para hacer de núcleo controlador.

Y los usuarios finales, que va a ser el que use las máquinas virtuales generadas con Vagrant y las cuales cada una constan de su propia IP manual.

Ahora se mostrará un esquema de la infraestructura que sigue el proyecto en cuanto a su funcionalidad:

Diagrama

El contenido generado por IA puede ser incorrecto. En la máquina virtual se usará principalmente Vagrant para la automatización y creación de los entornos virtuales en VirtualBox.

Dentro tendremos los diferentes paquetes de software que usaremos, los cuales instalará Ansible de forma automática a través de los módulos.

Para hacer eso necesitaremos una serie de servicios disponibles para que se puedan ejecutar las tareas que se creen. Por ejemplo, si necesitamos realizar una conexión remota con SSH, para que se pueda hacer esta tarea automática tendremos que tener el servicio activo y disponible. Ansible comprobará que dichos servicios estén disponibles y accesibles.

Se hará mediante roles y módulos, que podremos reutilizar para hacer la misma tarea u otra similar en otro momento y que resulte eficiente. Sirven para automatizar las configuraciones que se creen.

Estos módulos y roles se realizarán mediante configuraciones y scripts que la máquina virtual necesita tener implementados para configurar todo correctamente. Por ejemplo, se pueden usar scripts o archivos de configuración que, en el futuro, usará Ansible.

Todas estas dependencias son obligatorias para que se automatice todo con el sistema de Vagrant y Ansible.

### **SOLUCIONES PARA SU SEGURIDAD:**

Necesitamos tener algo de seguridad en este proyecto ya que, si no, podrían entrar intrusos o personal no autorizado a las configuraciones de Ansible y podrían poner en peligro todo el trabajo realizado. También hay que asegurar el tráfico de la red y la protección de los datos, ya sea por archivos o scripts, o permisos en la red y roles…

Para todo esto, se va a configurar la autenticación y el control de acceso de SSH para el acceso a los entornos virtuales usando claves SSH públicas, privadas o de sesión, según se necesiten.

En cuanto a los datos, van a estar cifrados a través de Ansible. Se trata de configurar Ansible de manera que pueda automatizar una tarea para aplicar una configuración que permita cifrar la información de los servicios que esté usando.

Por último, se usará el firewall de Ubuntu (UFW) para vigilar y permitir o denegar el tráfico de la red y garantizar que todos los puertos que se tengan que usar estén abiertos para los servicios. Todo esto se hará a modo de tarea automatizada con Ansible para las máquinas generadas.

## **4.2 TECNOLOGÍAS A USAR**

Las tecnologías que se han usado para realizar el proyecto son las siguientes. Las explicaré primero de manera resumida:

* Ansible, que es la que se encarga de la gestión y automatización de la estructura que vamos a crear de los entornos virtuales a través de configuraciones, instalación de software, programación de tareas…
* VirtualBox, software necesario para la virtualización de las máquinas virtuales y la ejecución de varios sistemas operativos que usaremos en el proyecto.
* Vagrant, que utilizaremos para la creación de los entornos virtuales aplicándole el sistema operativo y las características que queremos que tengan mediante una interfaz gráfica. Se usará en compañía de los dos programas mencionados anteriormente.
* Phyton, Bash…, que son los lenguajes que se van a usar para la creación de los documentos y archivos de configuración. Esto se usará mediante sus respectivos módulos y extensiones, por ejemplo: extensiones, los apt, nano...
* Microsoft Word y GitHub, que son los programas que se usan en la máquina anfitriona para la redacción de la documentación y el guardado de información con respecto al código y el proyecto en sí.
* Ubuntu, que es el sistema operativo que vamos a usar para generar los entornos virtuales. Existen varias versiones, pero en este caso vamos a usar las que hemos mencionado en el apartado de arriba. Está basado en Linux. Se crearán en archivos de configuración, de imágenes ISO y/o imágenes base.

Todas estas herramientas han sido seleccionadas por medio de recomendación de la tutora de PFC y también por investigación de multitud de herramientas para creación de entornos virtuales y gestión y automatización de las tareas. He considerado y visto que son fáciles de usar por cualquier persona con conocimientos informáticos y con labores de administración de sistemas, se complementan muy bien y cumplen las funciones que se necesitan para llevar a cabo el proyecto.

Aún así, se ha hecho una pequeña búsqueda de otras alternativas a las recomendadas e investigadas por mí. Como resultado de este sondeo, a continuación, se muestra un esquema de las alternativas que se han podido utilizar y su justificación del por qué no se han utilizado:

Texto

El contenido generado por IA puede ser incorrecto.

A partir de ahora, vamos a describir las tecnologías elegidas.

### **4.2.1 ANSIBLE**

Cuando estamos en una empresa, es muy importante el sector de la informática para gestionar y administrar todo lo relacionado con la tecnología e informática. Para ello, los administradores de sistemas son los encargados de llevar a cabo este proceso y, junto a ellos, los programas necesarios para facilitar esta tarea. Pero al usar estos programas o procesos un número elevado de veces, nos podemos encontrar el problema de que la tarea para el administrador puede resultar un tanto monótona, con mayor indicio de errores, menos eficiente…

Para solucionar esto, podemos usar Ansible, que es una herramienta gratuita y de código abierto que sirve para automatizar estas tareas a la hora de administrar una gran cantidad de equipos de manera repetitiva, para evitar este gran número de errores, y para trabajar de una manera más óptima. Su conexión con otros equipos se hace de manera remota a través del protocolo SSH.

Logotipo

El contenido generado por IA puede ser incorrecto.

Este programa se divide en una máquina de control, en módulos y en playbooks:

La **máquina de control** es la que se encarga de mandarle a los demás equipos las tareas para que las realicen de manera automática a través de un código que se crea para esa misma función. Este código se almacena en un inventario, donde aparecen las máquinas que hay que gestionar. El lenguaje principal usado para este código es Python, aunque se puedan usar otros.

Los **módulos** son los programas que contienen ese código del que hemos estado hablando anteriormente junto a metadatos que dicen donde y cuando se realiza la tarea, creados en la máquina de control y mandados a los demás host para que puedan realizar las tareas automatizadas. Hay muchos tipos de módulos que podemos utilizar para la creación de la tarea: crear usuarios, gestionar bases de datos, gestionar la red…

Los **playbooks** son los archivos de texto que indican el resultado final de la tarea que queremos obtener para aplicarlos a los equipos que estamos administrando. Estos se usan para administrar las configuraciones de los servidores, existen numerosas posibilidades de uso y configuraciones. Se crean o expresan desde el lenguaje YAML, que es un lenguaje de declaración de datos que se usa, sobre todo, en archivos de configuración.

El **inventory** es el archivo usado para, digamos, registrar información sobre las máquinas virtuales generadas. Un listado que incluye ubicación, ip del host, nombre…

El proceso de ejecución de Ansible con las partes mencionadas anteriormente se haría de la siguiente manera:

Primero creas un playbook con el resultado final que quieres obtener y en qué lugar lo quieres obtener (se crearía en la máquina de control). Luego la máquina de control lee ese playbook y selecciona el o los módulos correspondientes para ejecutar la tarea solicitada. Cuando selecciona los módulos que va a necesitar, envía esos módulos a los equipos en los que se quiere automatizar la o las tareas y los ejecuta allí.

Para verlo más claro pongo un ejemplo. Imagina que quieres crear una tarea para que se instale un paquete. Para ello primero deberías crear un playbook diciéndole que dicho paquete esté instalado, luego Ansible accederá al playbook y usará un módulo apt que se requiere para instalar el paquete, y finalmente la máquina de control manda ese módulo con ese apt a los equipos donde se quiera instalar el paquete y los pone en marcha allí.

Las ventajas que se pueden brindar a la hora de usar Ansible son las siguientes:

* No se necesitan otros programas específicos para realizar una tarea, ya que con los módulos de Ansible se puede hacer todo desde este único programa
* Para crear las tareas y que se automaticen, solo se necesita Ansible y que esté instalado Phyton en la máquina de control
* Hace uso de módulos, que amplia mucho su funcionalidad, sus opciones y su facilidad de uso

### **4.2.2 VAGRANT**

Cuando queremos automatizar estas tareas como hemos explicado anteriormente, deberemos hacerlas sobre una máquina que incluya un sistema operativo. También se da el problema en las empresas y lugares con muchos equipos que usan entornos virtuales la mítica frase “en mi máquina funciona”, queriendo dar lugar a posibles configuraciones diferentes en un entorno virtual por equivocación o por necesidad de modificar la configuración de alguna. No van a la par. Lo que se tardaría en crear una máquina virtual de manera normal (horas, días…) con este software la puedes tener en 5 minutos.

Imagen que contiene Forma

El contenido generado por IA puede ser incorrecto.Para solucionar este error de rendimiento o eficiencia, podremos acudir también a Vagrant. Es un software libre que se utiliza tanto de forma individual como de forma conjunta (como vamos a mostrar en este caso) para la creación de entornos virtuales a través de scripts en un archivo de configuración que crearemos con los requisitos necesarios y básicos.

Lo que hace este software es crear una capa por encima de la virtualización para poder realizar las acciones necesarias.

También tiene una serie de configuraciones, herramientas, y comandos principales que hacen posible este proceso de creación de entornos virtuales, son los siguientes: box, vagrantfile, provisiones,y providers. Adicional (PuPHPet)

Los **box o boxes** son las imágenes que generamos para usarlas en las máquinas virtuales con un sistema operativo. Pero claro, para esto es necesario tener un programa para ejecutar esas ISO, se utilizará el software de VirtualBox. Con estas ISO que creamos, podremos compartirlas a toda la planta empresarial para que no de problemas de diferenciación de entornos virtuales.

Un **vagrantfile**, como su propio nombre indica, se trata del fichero de configuración del entorno virtual de este software. En él podremos crear el código o script sobre las características que van a usar nuestras máquinas virtuales y que sean compatibles con la máquina anfitriona. Por ejemplo: indicarle las especificaciones con respecto al hardware virtual, número de puertos, las tarjetas de red, indicarle la ISO a usar, decirle con qué más herramientas tiene que trabajar (en este caso Ansible), controlar su rendimiento…

Los **provisioners** son las herramientas que hacen posible que el usuario final pueda cambiar la configuración de la máquina virtual a gusto y necesidad sin que se produzca ningún fallo. También hace que la instalación de los software se realice de manera automatizada (también lo podría hacer Ansible)

Y los **providers** son los diferentes servicios que va a llegar a usar el programa para que se puedan crear entornos virtuales en la máquina física o anfitriona. Digamos que sirve para que el software de Vagrant pueda colaborar con el programa de virtualización de los entornos virtuales. Por ejemplo, el proveedor que vamos a usar de VirtualBox

Como herramienta adicional que nos puede servir de ayuda, de manera opcional, es “PuPHPet”. Este es un software online que hace de asistente a la hora de crear el archivo de configuración de vagrantfile. Nos ayuda, por ejemplo, a definir qué características queremos que tenga nuestra máquina virtual, que ISO es recomendable que use, las bases de datos y los lenguajes de programación disponibles, servicios que se pueden instalar en el entorno… Cuando tengamos toda esta información completada, nos descargará el archivo de configuración de vagrantfile y también nos dirá qué pasos hay que seguir para ponerlo en funcionamiento.

Algunas de las ventajas que tiene el uso de Vagrant son las siguientes:

* Es fácil de configurar, ya que puedes usar un asistente de ayuda y no requiere muchos conocimientos en profundidad
* No necesita nada más, es decir, suele ser más que suficiente el uso de esta herramienta
* Es bastante seguro, porque solo se exponen a los puertos reenviados en la máquina virtual

### **4.2.3 VIRTUALBOX**

Para poder usar estos programas mencionados anteriormente, necesitamos un lugar donde se puedan ejecutar estos entornos virtuales y una o varias máquinas invitadas que cojan recursos de la máquina anfitriona para poder instalar en ellas las herramientas necesarias para poder crear nuestros entornos virtualizados. En este caso he elegido el software de VirtualBox.

VirtualBox es un software libre dentro de la empresa de Oracle que se usa para la virtualización de sistemas operativos dentro de nuestro ordenador físico, de manera aislada para que ese ordenador físico no se encuentre en riesgo. También se le conoce hipervisor de tipo 2, que quiere decir que para su funcionamiento requiere un sistema operativo almacenado en una imagen ISO.

Imagen que contiene Icono

El contenido generado por IA puede ser incorrecto.

Para su funcionamiento es necesario repartirle los recursos adecuado acorde a nuestra máquina anfitriona con respecto al hardware para que ambas puedan funcionar de manera correcta. Es compatible con muchas versiones de muchos sistemas operativos, como, por ejemplo: Windows, Linux, MacOS…

### **4.2.4 COMBINACIÓN DE ANSIBLE, VAGRANT Y VIRTUALBOX**

Ahora vamos a comentar un escenario en el que se pueden implementar estas 3 herramientas y comentar su flujo de trabajo.

* Se instalará en la máquina anfitriona el software de VirtualBox, que nos permitirá crear los entornos virtuales sobre los que trabajaremos. A este software se le añadirá la descarga de las imágenes ISO, que contendrán los sistemas operativos necesarios.
* Luego procederemos a la instalación de los esas imágenes ISO introduciéndolas en VirtualBox, añadiendo los requisitos con respecto al hardware que tendrán, y su instalación una vez dentro de la máquina. Los requisitos que he especificado para cada una de ellas se muestran a continuación:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **NOMBRE MÁQUINA** | **ESPACIO DE ALMACENAMIENTO** | **CPU** | **RAM** | **GUEST**  **ADITTIONS** |
| UBUNTU\_SERVER\_24.04 | 75 GB | 4 NUCLEOS | 5GB | SI |

* Cuando tengamos disponibles las máquinas virtuales con las que vamos a trabajar, conectadas a la red y entre ellas, tendremos que instalar las herramientas o paquetes que se necesiten. Primero comenzaremos instalando el software de Ansible (porque después él se instalará las demás herramientas)
* Dentro de Ansible, crearemos las primeras instrucciones de configuración en formato .YAML, es decir, los Playbooks de Ansible.
* Después de crear esas instrucciones, Ansible se conectará a nuestra máquina virtual con la que estamos trabajando y las iniciará. Obtendremos herramientas como Vagrant, que usaremos después.
* Con las instrucciones creadas en el programa de Ansible, Vagrant empezará a crear los entornos virtuales con una configuración que le hemos dado en esos ficheros de configuración anteriormente.
* Luego se realizará la conexión entre Vagrant y Ansible con la máquina virtual para que se pueda configurar correctamente acorde a los requisitos que queremos.
* Y para instalar y configurar todo lo necesario que queramos en esa máquina virtual, Ansible lo que hará será entrar el su playbook para acceder a todo este contenido en cada máquina virtual.

## **4.3 IMPLEMENTACIÓN**

Con la realización del proyecto, se han incluido unos puntos necesarios que se siguen a la hora de la implantación de la idea de proyecto y del proyecto en sí. Abarcamos desde puntos referidos al hardware, software, virtualización, automatización, seguridad…

A continuación se va a explicar más al detalle para que se entienda mejor cada uno de estos puntos de implementación

1. **Organización y estructurado del proyecto:**

Se ha seguido la estructura de Ansible y Vagrant, tanto a nivel de árbol de directorios, como de investigación, y como de proceso de ejecución. Todo se realiza de esta forma para que sea fácil de usar y efectiva.

1. **Creación de entornos virtuales a través de Vagrant y VirtualBox**

Hemos cumplido el objetivo principal de generar las máquinas virtuales con un archivo vagrantfile, aparte de configurarlas cada una con sus requisitos hardware y de red, y al terminarlas de configurar levantarlas con su correspondiente comando.

1. **Desarrollo del fichero de configuración playbook**

Otro de los objetivos cumplidos ha sido la creación de este archivo en el que se engloban los roles o tareas automatizadas que hemos ideado para que puedan usar las máquinas.

1. **Planificación, creación y ejecución de varios roles con Ansible**

De los roles mencionados anteriormente, se han creado con el objetivo de que cada uno cumpla con una función en concreto:

* Rol Usuarios: creado para que la máquina genere nuevos usuarios en cada inicio de sesión y eliminar usuarios antiguos para la máquina no se sature.
* Rol Primer Mensaje: una opción muy creativa si lo que se quiere es dar más contenido visual a los entornos virtuales a través de un mensaje personalizado de bienvenida
* Rol Firewall: la seguridad es muy importante para todo, por eso se ha querido incluir en este proyecto. La función principal de este rol es de instalar UFW para imponer las reglas de habilitar ciertos puertos y activar el firewall de forma segura.
* Rol Base de Datos: rol diseñado para la instalación del servidor de MariaDB, también configurar dicho servidor para que cree un usuario con todos los permisos y una base de datos plantilla para que se pueda crear en cada máquina.

1. **Pruebas de toda la configuración en las máquinas generadas**

Para comprobar que todo se ha realizado correctamente, hemos entrado a cada uno de los entornos generados y dentro de cada una se han comprobado todos los roles creados.

1. **Preparación de un manual de administrador y un manual de usuario**

Todo el proceso llevado a cabo en este proyecto se ha documentados en dos anexos independientes para explicar cómo se crea desde 0 y como se acceden a esas configuraciones: Manual de Administrador y Manual de Usuario.

## **4.4 DOCUMENTACIÓN**

En este apartado se recogen los siguientes anexos. Luego se adjuntarán junto con todo el proyecto para que la persona lo puedo llevar a cabo desde el principio:

* **ANEXO I: Manual de Administrador** : Contendrá el proceso de cómo crear los entornos virtuales y como gestionarlos y automatizarlos. Todo esto con VirtualBox, Ansible y Vagrant.
* **ANEXO ll: Manual de Usuario:** Mostrará cómo lanzar los entornos creados en el ANEXO I desde el punto de vista del usuario.

## **4.5 ADMINISTRACIÓN**

La administración es un punto muy importante para este proyecto y para la informática en general. Sin la administración no podríamos dar inicio a este proyecto, ni que el usuario pudiera consumir los entornos virtuales que este genera junto a sus tareas automatizadas internas.

Por eso, se considera que es importante ver, planificar y abarcar este punto en el proyecto para crear y mantener un buen espacio de trabajo con dicha administración. También para tener un plan seguro en el caso de que se dieran futuros fallos.

Vamos a dividir este apartado en tres subapartados que harán mejor el entendimiento de la administración del sistema del proyecto si se quisiera implantar en algún futuro en empresas, administraciones públicas…

Se dividirá en: Administración en la Preparación, Administración en el Desarrollo, Administración de Mantenimiento. Digamos que cada una se refiere a un periodo de tiempo en el proyecto (el pasado, el presente, y el futuro)

### **4.5.1 ADMINISTRACIÓN EN LA PREPARACIÓN**

Diagrama

El contenido generado por IA puede ser incorrecto.Primero llevaremos a cabo la idea de administración cuando preparamos el proyecto para después empezar a ejecutarlo. Sí, también en este proceso del proyecto es importante la administración, porque sin esto no sería posible avanzar.

Tendremos que pensar cómo vamos a plantearlo y las ideas de administración para cada punto de planteamiento.

Diseñaremos la arquitectura virtual mediante el programa de Vagrant con las máquinas que hemos creado. El administrador tiene que tener bien claro este primer paso para que a la hora de realizar el proceso tenga en cuenta la estructura y como lo va a aplicar. Para verlo de una forma más visual

Cuando el administrador sepa más o menos cómo va a organizarlo todo, se instalarán y configurarán las herramientas necesarias. A la hora de hacer esto, el administrador tiene que asegurarse que instala todo correctamente y configura los ficheros de cada programa acorde a su previa planificación. Es muy importante esta parte porque es la que más puede dar problemas a la hora de probarlo.

También hay que gestionar los usuarios iniciales que tendrán acceso a las máquinas generadas para que las puedan usar después, esto mediante Vagrant. Cuando lo vaya a realizar el administrador, tiene que tener en cuenta y debería anotar: cuáles son los nombres de usuario y contraseñas de cada máquina, los permisos básicos que se le asignan a estos usuarios, y como debe realizar la conexión mediante SSH

### **4.5.2 ADMINISTRACIÓN EN EL DESARROLLO**

Continuaremos con el tema de la administración, pero esta vez a la hora de ir creando lo que se vendría diciendo tronco del proyecto. Es de los puntos más importantes a la hora de administrarlo, ya que si falla algo en esta parte, costaría el tiempo de revisar qué puede ser lo que haga que no funcionen las cosas.

Para ello, tiene que tener en cuenta los siguientes puntos:

A la hora de trabajar en el entorno de la máquina controladora, el administrador deberá hacer una actualización seguida del sistema operativo para que se actualicen los paquetes y repositorios que se vayan agregando. Lo hará de manera manual en la máquina controladora, pero en los entornos virtuales se realizará de manera automática gracias al rol de sistema.

También deberá de llevar a cabo la gestión de los puertos que vaya a usar, permitiendo el paso de los que se usen (como por ejemplo es de SSH) y denegar los puertos que no se usen. Se hará mediante la herramienta UFW en este caso. Es bueno que el administrador lo lleve tramitado para que no de problemas más adelante.

En cuanto a Ansible, a la hora de ejecutar los roles, el administrador deberá de realizar una previa comprobación simulada de estos para ver si da error y resolver los problemas que se puedan dar. Sería también recomendable que después de eso, los compruebe de manera conjunta con la prueba real. Todo esto realizado desde la máquina controladora.

Llevar un mantenimiento de todos los ficheros y archivos que permiten que se realice el proyecto, como por ejemplo: vagrantfile, inventory, playbooks, main.yml… Para que el administrador pueda corregir errores detectados (ya sea por cosas mal escritas, permisos, dependencias…)

Todo esto lo deberá de tener registrado el administrador en uno o varios documentos en formato de manual, para que otra personas o administradores puedan implantar la idea de proyecto y no tener tantos conocimientos de informática para llevarlo a cabo.

### **4.5.3 ADMINISTRACIÓN DE MANTENIMIENTO**

Por último, después de realizar el proyecto y haberlo implantado en un cierto entorno, tendremos que realizar un mantenimiento recomendado cada cierto tiempo para verificar que todo está instalado con la última versión, que todo funciona correctamente y que nada se ha desorganizado, no borrado, ni otros posibles problemas.

Si fuera necesario sería recomendable que cada cierto tiempo el administrador implantara y actualizara unas medidas de seguridad para proteger claves privadas de conexión SSH ( por ejemplo con el uso de *ansible-vault* ) o de los usuarios. También se podría deshabilitar el acceso de manera remota con root y sustituirlo por el uso de ciertos usuarios específicos con *become*.

El administrador, si lo viera susceptible, podría realizar copias de seguridad cada cierto tiempo implantando un sistema de copias de seguridad en sitios como: bases de datos o ficheros de configuración.

Podría en un futuro ampliarse el proyecto, si el consumidor lo precisara, para añadir más mas máquinas virtuales o más máquinas controladoras con más máquinas virtuales para los clientes si se quisiera aumentar el número de estas. Para ello, el administrador debería repetir el proceso de este proyecto tantas veces como fueran necesarias.

A esto se le podría aplicar también el añadir nuevos roles para tener nuevas tareas automatizadas de lo que precise el cliente (servidores web, DNS…)

Y para terminar, no viene mal que el administrador realizara una revisión cada cierto tiempo sobre los usuarios que están activos en cada entorno virtual generado y/o eliminar los servicios que a la larga ya no sean necesarios, aparte de supervisiones del estado de los servicios que sí se usen, estado de firewall…

## **4.6 DESPLIEGUE**

Para realizar el despliegue de este proyecto se ha creado un manual de usuario, para que la persona que vaya a usar estos entornos virtuales junto a sus tareas automatizadas pueda empezar a usar esta idea. La idea principal de su despliegue se basa en poner en marcha las máquinas virtuales creadas desde la máquina controladora usando las herramientas de Vagrant, Ansible y VirtualBox, donde en ellas se puedan usar las tareas automáticas implantadas. Se puede realizar de manera local para realizar las pruebas o en un futuro para un lugar que precise de muchas máquinas virtuales

Para iniciar el despliegue, deberemos de cumplir con unos requisitos antes. Algunos de ellos serían los siguientes:

* Tener una máquina controladora (en este caso de Ubuntu Server) preparada
* También tener los programas de software que vamos a utilizar disponibles en dicha máquina (Ansible, Vagrant y VirtualBox)
* Contar con una estructura de árbol de directorios, para que la información de todo el proyecto quede bien estructurada y también funcione correctamente a la hora de probarlo.

Con Vagrant tendremos el archivo Vagrantfile, para definir las máquinas. Con Ansible contaremos con un archivo Inventario para registrar los hosts que usarás las tareas, los Playbooks para mencionar los roles a crear, y Roles para crear y configurar las tareas para las máquinas.

* Comprobar que funciona toda la red NAT que hemos creado en VirtualBox de la máquina anfitriona, para poder realizar en las máquinas las correspondientes instalaciones y actualizaciones

Una vez cumplidos estos requisitos previos podremos desplegarlo siguiendo, como se ha mencionado antes, el manual de usuario que se ha creado para esto respectivamente.

Primero encenderemos la máquina controladora para poder iniciar los entornos virtuales mediante Vagrant. Una vez tengamos la máquina controladora lista, nos ubicaremos en el directorio de Vagrant (en el árbol de directorios de nuestro proyecto) para lanzarlos. Los podrá poner en marcha y configurarlos gracias a que se encuentran registrados en el fichero de configuración Vagrantfile.

Cuando se inician, se les aplican automáticamente la configuración que les hemos asignado. Entre ellas reciben las IP privadas que se les van a asigna, las claves SSH correspondientes a cada máquina, los usuarios y las contraseñas, los requisitos en cuanto al hardware…

Las claves privadas se almacenan en el archivo Inventory de Ansible. Las configuraciones se pueden modificar una vez creadas las máquinas virtuales, pero lo recomendable es que no se hagan y se dejen las que reciben por defecto (porque puede dar lugar a errores en el futuro).

Una vez iniciadas ya podremos acceder a ellas y ver que se han creado correctamente. El acceso a ellas se puede realizar de dos maneras: mediante conexión SSH y mediante modo gráfico. Por SSH usa las rutas a las claves privadas generadas por Vagrant. Mediante el entorno gráfico se accede por VirtualBox.

Ya teniendo las máquinas iniciadas podremos aplicar los roles que hemos creado antes con Ansible. Simplemente nos dirigiremos al directorio donde tenemos todos los archivos de Ansible y ejecutaremos el comando para comprobar y lanzar los roles creados. En el caso de este proyecto, los roles que se han creado para las tareas automáticas son: creación y eliminación de usuarios, configuración del firewall, configuración del primer mensaje, instalación y uso de base de datos con María DB.

Por última parte del despliegue, verificaremos si los roles creados se han configurado de manera correcta y se han aplicado a cada máquina virtual generada. Primero accederemos a los entornos virtuales y haremos la ejecución de los roles para ver si son efectivos.

Esta manera de ver el proyecto permite que todo se despliegue de manera ordenada y escalable, facilitando la administración el varias máquinas virtuales.

**6.** **CONCLUSIÓN**

Con la realización de este proyecto, se ha querido descubrir una forma de mejorar la eficiencia de la creación y el uso de máquinas virtuales que podemos utilizar para implantar en diversos lugares donde se precisen. Todo esto mediante el uso de ciertas herramientas que, de manera escalonada, se han programado para que sea posible desplegar y configurar todo desde una máquina controladora.

El objetivo es que se pueda realizar todo este proceso sin necesidad de tener que intervenir en todas las configuraciones desde cero cada vez que se quiera crear una máquina virtual o cada vez que se quiera implantar una tarea automatizada en las máquinas cliente.

El hecho de que se hagan uso de módulos y roles de manera automatizada hace que se reduzcan bastantes los tiempos de configuración, que sea coherente y que sea fácil de administrar por parte del administrador.

Vamos a analizar la eficacia de las herramientas que se han implantado y la conclusión de cada una de ellas.

La herramienta de Vagrant ha sido útil para definir o crear las máquinas virtuales plasmadas en VirtualBox que vamos a generar para aplicarle las tareas automatizadas. Con Ansible hemos conseguido crear tareas automatizadas mediante unos roles diseñados para tareas administrativas que aplicaremos a los entornos virtuales para que cuando se acceda a la máquina, el cliente pueda realizar acciones más ágilmente y más fácilmente.

Su uso en conjunto ha sido capaz de demostrar que este proyecto puede ser viable, escalable y aplicable a proyectos reales

Un resumen de los resultados que hemos conseguido con este proyecto es el siguiente:

* Hemos creado una máquina controladora en nuestra máquina anfitriona para que así podamos generar dos entornos virtuales dentro de ella mediante Vagrant.
* Se han diseñado una serie de roles para poner en función las tareas automatizadas que son las siguientes: crear y eliminar usuarios, actualizar el sistema de la máquina, configurar los puertos del firewall, instalar y usar un servidor de base de datos con MariaDB
* También se han comprobado los roles creados en las máquinas clientes de manera manual
* Los objetivos principales que queríamos cumplir se han cumplido, es decir, hemos conseguido crear un proyecto escalable, capaz de ser reproducible y se puede mantener de manera fácil.

Pero como hay resultados obtenidos, también se han observado que este proyecto puede mejorarse y ampliarse aplicando una serie de configuraciones adicionales. Son ideas que se pueden llevar a cabo para implantarlo más adelante en futuras versiones o cuando se quiera expandir.

* Una posible idea sería añadir más seguridad al proyecto mediante la protección de contraseñas y archivos delicados con ansible-vault (que es una función que viene incluida con la herramienta de Ansible de forma complementaria)
* Añadir nuevos roles que el cliente solicite y para que pueda realizar determinadas tareas específicas, por ejemplo, servidor DNS, realizar copias de seguridad periódicas…
* Hacer un uso de control de versiones de las herramientas con Git, para mantener un historial actualizado del proyecto

Como conclusión, con este proyecto he aprendido un mundo muy efectivo de la virtualización con la creación, gestión y administración de entornos virtuales junto con la inclusión de tareas automatizadas que hacen posible las administración en las máquinas para los clientes. Pensaba que sería una tarea más compleja por el hecho de mantener la máquina y todo lo que en su interior lleva.

También ha hecho que adquiera experiencia con el uso de estas herramientas y aplicarlas a un entorno Linux. Otro punto en el que puedo reflexionar es la capacidad de afrontar los problemas que han ido surgiendo con las configuraciones y los ficheros.

Para terminar, todo esto me ha mostrado la importancia que tiene organizar todo esto y plasmarlo en una documentación a nivel de administración, útil para futuros administradores o usuarios que quieran implantar y usar esta idea de proyecto. Con esto se mejora la calidad del resultado final, la facilidad de su mantenimiento y/o ampliación en un futuro

**7. BIBLIOGRAFÍA**

Todo esto ha sido posible gracias a la investigación de una serie de apartados que se han ido cuestionando para seguir la elaboración del proyecto. Entre ellos se encuentran: qué son y cómo se usan cada una de las herramientas (tanto individualmente como en conjunto), enlaces de almacenamiento del proyecto, descarga de repositorios necesarios, algunos tutoriales de apoyo…

Se incluyen a continuación en una lista:

* GitHub. (2025) *CarlaGalindoCorchero / PROYECTO\_TFG* <https://github.com/CarlaGalindoCorchero/PROYECTO_TFG>. *Contenido para descargar archivos necesarios para la elaboración del proyecto.*
* Oracle. (2025) *Oracle VirtualBox*. <https://www.oracle.com/es/virtualization/technologies/vm/downloads/virtualbox-downloads.html>. *Utilizado para descargar el programa de VirtualBox.*
* Ualmtorres. (2019) *Curso de iniciación a Ansible para la automatización de operaciones* <https://ualmtorres.github.io/CursoAnsible/tutorial/>. *Información sobre qué es Ansible.*
* Red Hat. (2022) *Ansible ¿Qué es y cómo funciona?* <https://www.redhat.com/es/topics/automation/learning-ansible-tutorial>. *Para saber qué son y cómo funcionan los módulos de Ansible*
* Adictos al trabajo. (2024) *Creación de entornos de integración con Ansible y Vagrant* <https://adictosaltrabajo.com/2015/09/04/creacion-de-entornos-de-integracion-con-ansible-y-vagrant/>. *Comprender el funcionamiento de la combinación de Ansible y Vagrant.*
* Arsys. (2024) *¿Qué es Ansible?* <https://www.arsys.es/blog/ansible>. *Para saber qué son y cómo funciona el fichero de configuración Playbooks*
* Arsys. (2024) *Qué es Vagrant y cómo configurar un entorno de desarrollo virtualizado* <https://www.arsys.es/blog/entorno-desarrollo-vagrant>. *Información sobre qué es Vagrant*
* KeepCoding. (2024) *¿Qué es Vagrant y para qué sirve?* <https://keepcoding.io/blog/que-es-vagrant-y-para-que-sirve/>. *Para saber qué son y cómo funcionan los provisioners y los providers de Vagrant*
* VagrantAndDocker. (2015) *Vagrant & Docker* <https://vagrantanddocker.blogspot.com/2015/06/cuales-son-sus-ventajas-y-desventajas.html>. *Ventajas de uso de Vagrant*
* Geeknetic. (2020) *VirtualBox: ¿Qué es y para qué sirve?* <https://www.geeknetic.es/VirtualBox/que-es-y-para-que-sirve>. *Página que contiene información acerca de VirtualBox*
* SiteGroung. (2025) *¿Qué es SSH y cómo lo uso?* <https://www.siteground.es/kb/que-es-ssh/>*. Idea para implementar el uso de SSH con respecto a sus claves*
* IMB. (2024) *¿Qué es Docker?* <https://www.ibm.com/es-es/think/topics/docker>. *Información sobre qué es Docker*
* Pure Storage.(2025) *¿Qué es VMWare?* <https://www.purestorage.com/la/knowledge/what-is-vmware.html#:~:text=VMware%20es%20una%20empresa%20de,restricciones%20de%20su%20hardware%20f%C3%ADsico>. *Información acerca de qué es VMWare*
* Ubunlog. (s.f) *Multipass, el proyecto de Canonical para implementar máquinas virtuales* <https://ubunlog.com/multipass-el-proyecto-de-canonical-para-implementar-maquinas-virtuales/>. *Información con respecto a qué es la herramienta de Multipass*
* CodigoElectronica. (2022) *¿Qué es Puppet*? <http://codigoelectronica.com/blog/que-es-puppet>. Para saber qué es Puppet
* Somos Libres. (2023) *Como crear un árbol de directorios con un solo comando: Paso a paso* <https://www.somoslibres.org/index.php/76-aprenda-linux/12186-como-crear-un-arbol-de-directorios-con-un-solo-comando-paso-a-paso#google_vignette>. Para refrescar la memoria acerca de la creación de una estructura de directorios en Ubuntu
* Red Hat. (2023) *YAML: qué es y usos* <https://www.redhat.com/es/topics/automation/what-is-yaml>. *Información acerca de qué es un archivo .YAML y cómo funciona*
* Ansible Galaxy. (s.f) *Roles* <https://galaxy.ansible.com/ui/standalone/roles/>. *Página donde encontrar códigos para la automatización de tareas que los usuarios publican*
* HashiCorp Cloud Platform. (s.f) *Descubre los Box de Vagrant* <https://portal.cloud.hashicorp.com/vagrant/discover>. *Descargar los boxes / imágenes base para elegir las características de la máquina que generemos en Vagrant*
* Instalar VirtualBox y Vagrant en Ubuntu. (2015) <https://www.youtube.com/watch?v=yGviTwD3hWM>. *Proceso de instalación del software de VirtualBox y Vagrant dentro del S.O de Ubuntu*

**8.** **ANEXOS**

A partir de aquí se adjuntan los manuales necesarios para poder poner en práctica el proyecto., tanto el manual de administrador como el manual de usuario.

Diagrama

El contenido generado por IA puede ser incorrecto.

I.E.S. “ÁGORA”

Dpto.: Informática

Ciclo Formativo de G. Superior: Administración de Sistemas Informáticos y Redes

ANEXO I : Manual de Administrador

Imagen que contiene Icono

El contenido generado por IA puede ser incorrecto.

Autor o autores: Carla Galindo Corchero

Tutor: Maria Francisca Roncero Holgado

Cáceres, a junio de 2025

**INDICE**

[**1.** **REQUISITOS PREVIOS** 27](#_Toc197287418)

[**2.** **INSTALACIÓN DEL SOFTWARE PRINCIPAL** 31](#_Toc197287426)

[**2.1 INSTALACIÓN DE VAGRANT** 32](#_Toc197287427)

[**2.2 INSTALACIÓN DE ANSIBLE** 33](#_Toc197287428)

[**3.** **CREACIÓN DE LA ESTRUCTURA DE DIRECTORIOS** 34](#_Toc197287429)

[**4.** **CREACIÓN DEL ARCHIVO DE CONFIGURACIÓN *VAGRANTFILE*** 35](#_Toc197287430)

[**5.** **LANZAMIENTO Y COMUNICACIÓN DE ENTORNOS CREADOS** 39](#_Toc197287431)

[**6.** **CREACIÓN DEL ARCHIVO DE INVENTARIO *INVENTORY* PARA *HOST*** 42](#_Toc197287432)

[**7.** **CREACIÓN DEL ARCHIVO DE CONFIGURACIÓN PLAYBOOK**](#_Toc197287433) 44

[**8.** **CREACIÓN Y APLICACIÓN DE ROLES** 46](#_Toc197287434)

[**8.1** **ROL DE USUARIOS** 48](#_Toc197287437)

[**8.2** **ROL DE PRIMER MENSAJE** 49](#_Toc197287439)

[**8.3** **ROL DE FIREWALL** 50](#_Toc197287440)

[**8.4** **ROL DE BASE DE DATOS** 51](#_Toc197287441)

[**8.5 COMPROBACIÓN DE ROLES CREADOS** 53](#_Toc197287442)

[**9.** **LANZAMIENTO DE CONFIGURACIONES DE ANSIBLE** 54](#_Toc197287443)

1. **REQUISITOS PREVIOS**

Antes de nada, deberemos de realizar unas configuraciones previas para preparar el entorno de trabajo donde se va a llevar a cabo el proyecto.

Lo primero será preparar las herramientas que se van a ir usando. Como primera herramienta instalamos VirtualBox (para emular los entornos virtuales) en nuestra máquina anfitriona. Adjunto el enlace de descarga [AQUÍ](https://www.oracle.com/es/virtualization/technologies/vm/downloads/virtualbox-downloads.html)

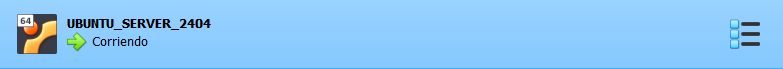
Logotipo

El contenido generado por IA puede ser incorrecto.

Cuando VirtualBox esté instalado y configurado correctamente, se prepararán las imágenes ISO que necesitamos para poner en funcionamiento las máquinas virtuales que vamos a usar. He adjuntado las imágenes ISO en el siguiente enlace (GitHub) de Ubuntu Server [AQUÍ](https://github.com/CarlaGalindoCorchero/PROYECTO_TFG)

Con esas imágenes ISO, crearemos la máquina virtual principal con las siguientes especificaciones: Ubuntu Server 22.04 LTS con entorno gráfico (el cual instalaremos más adelante).

**REQUISITOS Y CONFIGURACION DEL SERVIDOR EN VIRTUALBOX**:



**Tabla

El contenido generado por IA puede ser incorrecto.**

Arrancamos la máquina virtual y completamos la instalación de esta. Ahora vamos a instalarle el entorno gráfico a Ubuntu Server para que podamos trabajar de manera más cómoda en el proyecto y podamos tener acceso a más funcionalidades de manera gráfica.

Iniciaremos nuestra máquina virtual de Ubuntu y escribiremos el siguiente comando para buscar las actualizaciones y para que esas actualizaciones buscadas se actualicen (la captura la muestro desde un entorno gráfico porque lo estoy explicando después de haber realizado las pruebas)

|  |
| --- |
| adminserver@adminserver:~$ **sudo apt update && upgrade -y** |

Después pondremos el comando que hará que se instale el paquete para poder desbloquear el menú de instalación del entorno gráfico. Se muestra a continuación:

|  |
| --- |
| adminserver@adminserver:~$ **sudo apt install tasksel** |

Cuando se haya instalado el paquete, ya podremos ejecutar Tasksel (lo necesario para tener entorno gráfico en nuestro Ubuntu Server). Añadiremos el siguiente comando al terminal.

|  |
| --- |
| adminserver@adminserver:~$ **sudo tasksel** |

Se nos abrirá un menú como el que se muestra a continuación y seleccionaremos el entorno que más nos guste o el que sea más conveniente según vayamos a usarlo moviéndonos por el menú con las teclas de dirección. En este caso he usado MATE porque veo que es de los más fáciles de usar y entender. Cuando lo seleccionemos, pulsaremos la barra espaciadora y se comenzará a descargar.

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

Texto

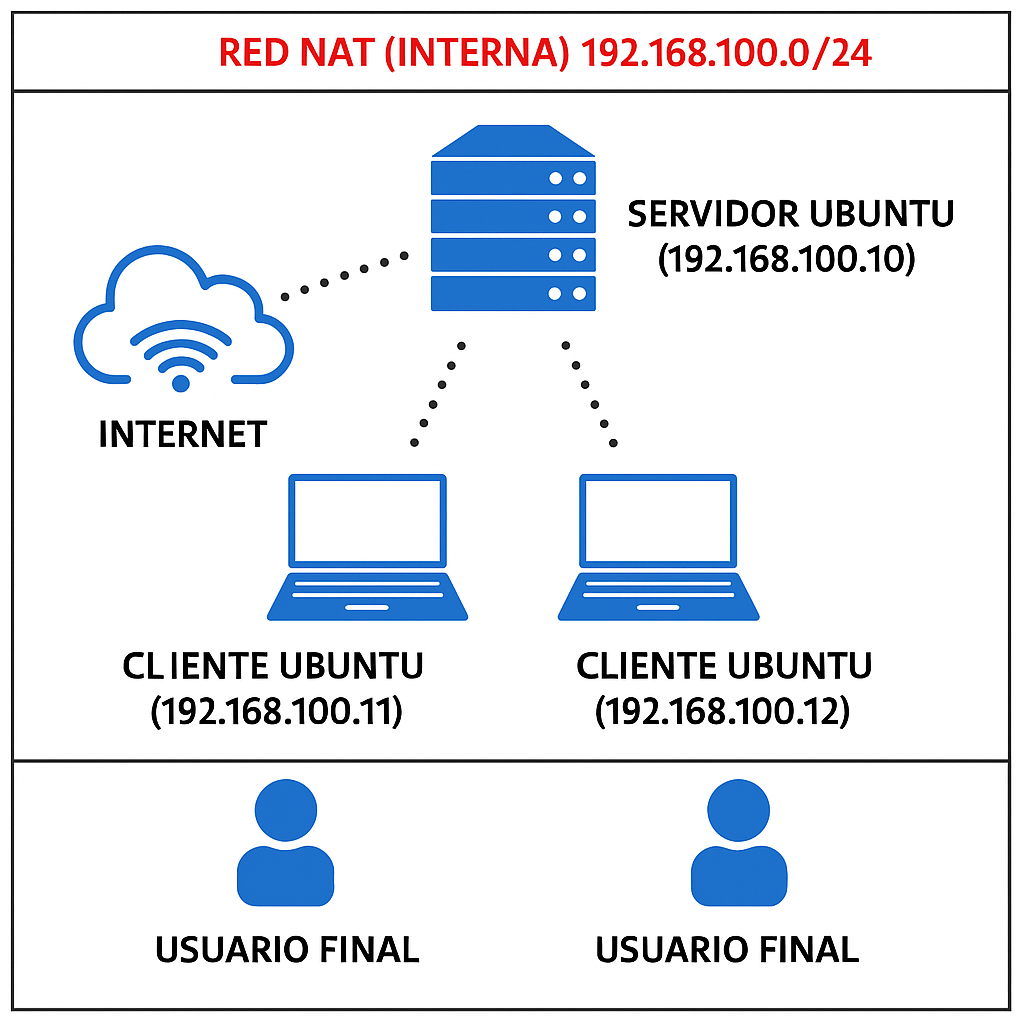
El contenido generado por IA puede ser incorrecto.

Cuando se haya completado la descarga, solo habrá que reiniciar el sistema y ya tendríamos aplicado el entorno gráfico

El siguiente requisito previo para cumplir será estructurar la topología de red que vamos a implantar. En este caso se ha decidido crear una red virtual interna, con IP de red: (192.168.100.0 /24), donde conectaremos nuestro dispositivo dentro de esa red mediante RED NAT de VirtualBox.

Hay que aclarar que la red virtual tendrá una funcionalidad: proporcionar conexión a internet en estas (siempre que la máquina física /anfitriona esté conectado a internet.

Muestro un esquema de como quedaría este escenario:



* El Servidor Ubuntu va a actuar como máquina controladora. Se va a encargar de tanto de crear los entornos virtuales (con Vagrant) como de configurarlos para que actúen de forma automática según le indiquemos en las tareas (con Ansible).
* Los Clientes Ubuntu son los que va a hacer de clientes en el proyecto. Se encargarán de recibir las tareas automatizadas para que se puedan ofrecer estos entornos a los usuarios y puedan trabajar con todas las herramientas implantadas.
* El Usuario final se encargará de recibir el resultado final realizado por el administrador en el servidor e la máquina controladora. El usuario final solo tendrá acceso a las máquinas generadas, mientras que el servidor será administrador por un administrador que las creará y configurará.

En VirtualBox, los adaptadores de la topología de red explicada anteriormente se quedarían de la siguiente manera.

Imagen que contiene Tabla

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Deberemos de entrar en la máquina controladora de Ubuntu Server para terminar de configurar la red. Se hará a través del archivo de configuración de “Netplan”, para asignar la IP de manera manual.

## 

## **SERVIDOR**:



Texto

El contenido generado por IA puede ser incorrecto.



Para comprobar que tenía conexión a internet, realicé un ping hacia internet (8.8.8.8). Vi que no se podía realizar la conexión porque a la máquina virtual no le llegaba internet aún. Revisé primero la RED NAT que había creado y vi que tenía activado DHCP (cuando debería de estar desactivado porque estoy creando una conexión en la que le asigno IP de forma manual). Lo volví a intentar y el mismo fallo seguía. También comprobé el tema de adaptadores, posibles incompatibilidades…

La solución finalmente era actualizar la versión de VirtualBox a la última versión, así solo se arregló el fallo.

1. **INSTALACIÓN DEL SOFTWARE PRINCIPAL**

Para poder empezar a crear y gestionar los entornos virtuales, tenemos que instalar el software necesario. Estas serán instaladas en la máquina controladora de Ubuntu Server, en este caso. También será instalado Ansible para demostrar que se pueden gestionar máquinas virtuales que no estén únicamente creadas con Vagrant. Con esto se consigue que la máquina controladora de Ubuntu Server pueda gestionar varios tipos de infraestructuras.

Lo dividiremos en 2 apartados: en la instalación de Vagrant, y la instalación de Ansible.

## **2.1 INSTALACIÓN DE VAGRANT**

El primero programa que instalaremos en Ubuntu será Vagrant, para poder crear los entornos de virtualización. He visto que hay dos métodos de instalación: a través del navegador web por la página oficial de Vagrant (requiere de Ubuntu con entorno gráfico), o a través del terminal por la biblioteca de repositorios. Se hará de la segunda manera.

Para obtener Vagrant, lo haremos desde la siguiente página de descarga [AQUÍ](https://developer.hashicorp.com/vagrant/install). El enlace te redirigirá a una web donde explica la instalación de Vagrant según el sistema operativo que vayamos a usar. Como se va a usar Ubuntu, pues me dirijo al apartado de “Linux” y “Ubuntu/Debian”.

Texto

El contenido generado por IA puede ser incorrecto.

Tendremos en cuenta esos tres comandos y los copiaremos de uno en uno para pegarlo en el terminal de la máquina y poder instalar la herramienta.

|  |
| --- |
| adminserver@adminserver:~$ **wget -O –** [**https://apt.releases.hashicorp.com/gpg**](https://apt.releases.hashicorp.com/gpg) **| sudo gpg—dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg** |

*(En este primer comando se instala la clave pública de HashiCorp, para que podamos después instalar Vagrant en formato “APT” sin ningún problema)*

|  |
| --- |
| adminserver@adminserver:~$ **echo “deb [arch=$(dpkg –print-architecture) signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg]** [**https://apt.releases.hashicorp.com**](https://apt.releases.hashicorp.com) **$(lsb\_release -cs) main” | sudo tee /etc/apt/source.list.d/hashicorp.list** |

*(Con este segundo comando, coge la clave pública que hemos instalado antes y busca los repositorios a descargar para poder instalarlos. Los instala correctamente)*

|  |
| --- |
| adminserver@adminserver:~$ **sudo apt update && sudo apt install Vagrant** |

*(Y con este último comando, lo que hacemos es actualizar los repositorios de nuestra máquina para poder instalar Vagrant sin problema)*

Una vez realizado todo esto, para comprobar que está todo debidamente instalado, escribiremos lo siguiente. Deberá de aparecer la versión que estamos usando de Vagrant:

|  |
| --- |
| adminserver@adminserver:~$ **vagrant—version** |

## 

## **2.2 INSTALACIÓN DE ANSIBLE**

Para instalar Ansible, hay que tener antes instalado el repositorio de Python. Este repositorio es necesario porque algunas de las funciones que se ejecutan en los playbooks de Ansible hacen uso de este lenguaje para que se puedan entender y ejecutar correctamente las tareas.

Para instalar el PIP (Administrador de Paquetes) de Phyton, abriremos el terminal en nuestra máquina controladora y ejecutaremos el siguiente comando:

|  |
| --- |
| adminserver@adminserver:~$ **sudo apt install pip** |

Para comprobar que todo ha salido satisfactoriamente, mostraremos la versión de Python. Si aparece, significa que lo hemos hecho correctamente. Se hace mediante el siguiente comando:

|  |
| --- |
| adminserver@adminserver:~$ **pip—version** |

Cuando ya se haya instalado Phyton, ya se podrá buscar en los repositorios de Ubuntu para descargar Ansible.

|  |
| --- |
| adminserver@adminserver:~$ **sudo pip install ansible** |

Pero al introducir este comando me decía que era peligroso realizar este paso tan agresivo porque se pueden poner en peligro y romper dependencias del propio sistema. Para esto le asigné la siguiente solución y me dejó.

|  |
| --- |
| adminserver@adminserver:~$ **sudo apt install ansible** |

Por último, para comprobar que Ansible ha sido instalado correctamente, mostraremos la versión con el siguiente comando:

|  |
| --- |
| adminserver@adminserver:~$ **ansible—version** |

1. **CREACIÓN DE LA ESTRUCTURA DE DIRECTORIOS**

Para seguir con nuestro proyecto, deberemos guardar todo el contenido referente a archivos, directorios, paquetes instalados, archivos de configuración, ficheros de vagrant y ansible… en un sitio para que sea más fácil de encontrar y gestionar de manera separada.

Se hará mediante un árbol de directorios el cual se explicará a continuación para que sea más fácil entender dónde está cada cosa antes de ponerse a su creación. Puede haber más niveles:

* **Raíz:** Llamada “proyecto”. Será la raíz de nuestro árbol de directorios y contendrá todos los archivos de los niveles inferiores.
* **Primer Nivel:** Se divide en dos subdirectorios “vagrant” y “ansible”. Cada uno de ellos contendrá sus archivos correspondientes para crear y configurar los entornos virtuales y las tareas automatizadas.
* **Segundo Nivel:** Se divide en tres subdirectorios dentro de su directorio padre “ansible”. Uno será para los “inventario” los cuales queremos que tengan su automatización, los “playbooks” donde guardaremos las ordenes o tareas que vamos a asignar, y los “roles” que crearemos.
* **Tercer Nivel:** En este nivel tendremos la información respecto a los roles que creemos en el directorio padre “roles” y demás archivos necesarios.
* **Posibles Niveles creados durante el proyecto (posterior):** Se podrían crear adicionalmente un directorio en el primer nivel para scripts o archivos de texto.

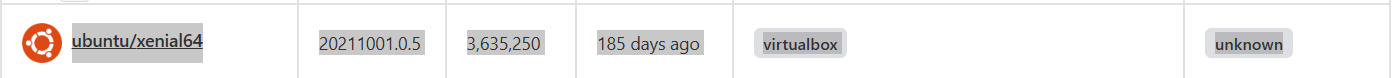
Ahora crearemos esta estructura para poder guardarlo todo de manera estructurada y fácil de encontrar. Quedaría de la siguiente manera:

Texto

El contenido generado por IA puede ser incorrecto.

1. **CREACIÓN DEL ARCHIVO DE CONFIGURACIÓN *VAGRANTFILE***

Crearemos un archivo de configuración de Vagrant (vagrantfile) donde definiremos los entornos virtuales a crear en la máquina controladora de Ubuntu Server. Pero antes de eso nos tendremos que elegir el Box a usar con el que queremos que nuestra máquina se construya diciéndoselo a Vagrant (digamos que elegiremos la imagen ISO que queremos que tengan nuestra máquinas virtuales). En este caso he decidido que sea un S.O Ubuntu Server llamado en la web “ubuntu/xenial64”. Nos iremos a la web, buscaremos el box con el que queremos trabajar y descargamos la última versión. (he elegido ese concretamente porque mi pc no soporta más nivel para virtualizar en este proyecto)



Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Luego nos dirigiremos al directorio que hemos creado anteriormente para trabajar allí, será en este caso “vagrant”. Pero para que funcione, Vagrant lo gestionará dentro de sus directorios de su software.

Con el siguiente comando se añadirá el box que hemos elegido en el siguiente enlace [AQUÍ](https://portal.cloud.hashicorp.com/vagrant/discover):

|  |
| --- |
| adminserver@adminserver:/proyecto/vagrant$ **vagrant box add ubuntu/xenial64** |

No tendrá que salir algo parecido a esto:

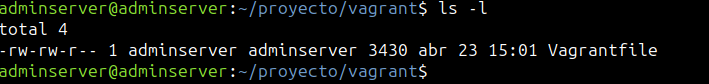
Texto

El contenido generado por IA puede ser incorrecto.

El siguiente paso será ya crear el archivo de configuración de la máquina virtual de Ubuntu vagrantfile. Para ello, pondremos el siguiente comando, que hará la creación de este fichero que es el que se va a ejecutar cada vez que iniciemos el entorno virtual creado con sus configuraciones asignadas junto con el box que acabamos de instalar.

|  |
| --- |
| adminserver@adminserver:/proyecto/vagrant$ **vagrant init ubuntu/xenial64 –box- version 20211001.0.5** |

El resultado será confirmado mostrando un ls en el directorio actual donde hemos creado el fichero y mostrándolo con **ls -l**



Abriremos ese archivo con el editor de “nano” y lo empezaremos a rellenar con la configuración que nosotros queramos asignarle.

|  |
| --- |
| adminserver@adminserver:/proyecto/vagrant$ **sudo nano vagrantfile** |

Se adjuntará el archivo de configuración completo con las configuraciones que se han elegido y luego se explicará que significa cada parte:

|  |
| --- |
| Vagrant.configure(“2”) do |config|  config.vm.define “cliente1” do |cliente1|  cliente1.vm.box = “ubuntu/xenial64”  cliente1.vm.box\_version = “20211001.1.0”  cliente1.vm.hostname = “cliente1”  cliente1.vm.network “private\_network”, ip: “192.168.56.10”  cliente1.vm.provider “virtualbox” do |vb|  vb.memory = 1024  vb.cpus = 1  end  cliente1.vm.provision “shell”, inline: <<-SHELL  apt-get update  apt-get install -y openssh-server python3  SHELL  end  config.vm.define “ubuntu\_cliente2” do |cliente2|  cliente2.vm.box = “ubuntu/jammy64”  cliente2.vm.box\_version = “20240315.1.0”  cliente2.vm.hostname = “cliente2”  cliente2.vm.network “private\_network”, ip: “192.168.56.20”  cliente2.vm.provider “virtualbox” do |vb|  vb.memory = 1024  vb.cpus = 1  end  cliente2.vm.provision “shell”, inline: <<-SHELL  apt-get update  apt-get install -y openssh-server python3  SHELL  end  end |

Con esta configuración se crearán dos máquinas virtuales clientes: el “cliente1” y el “cliente2”. En la línea 1, comenzamos indicando que se usa la versión 2 de Vagrant en cuanto a su configuración.

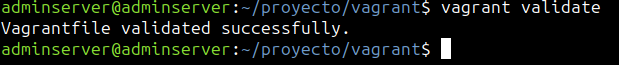
La siguiente línea define el primer entorno virtual a crear llamado “cliente1”. Este contendrá algunas características que se han definido: el box y la versión del box, el nombre que tendrá nuestro entorno virtual, la red privada en la que lo queremos incluir con una ip fija dentro del rango de nuestra red interna, el proveedor o herramienta que va a actuar a la hora de su ejecución (es VirtualBox) con alguna característica hardware, y un script que se ejecutará la primera vez que se inicie el entorno virtual (con los paquetes de SSH y Python que usaremos más adelante)

En la otra parte restante del código se define la segunda máquina que vamos a generar. Se podría seguir así con todas las máquinas que se quisieran generar, siempre que se le aporte su configuración correspondiente.

Para asegurarnos de que hemos escrito todo esto adecuadamente, lo guardamos y escribimos el siguiente comando de verificación de sintaxis.

|  |
| --- |
| adminserver@adminserver:/proyecto/vagrant$ **vagrant validate** |

El resultado de ese comando tiene que ser el siguiente (correcto):



1. **LANZAMIENTO Y COMUNICACIÓN DE ENTORNOS CREADOS**

Cuando tengamos listo el Vagrantfile, tendremos que levantar las máquinas automáticamente y también comprobar que contamos con conexión desde la máquina controladora de Ubuntu Server y entre sí. Para ello, ejecutaremos el siguiente comando para levantarlas:

|  |
| --- |
| adminserver@adminserver:/proyecto/vagrant$ **vagrant up** |

A la hora de ejecutarlo me daba un error que mencionaba que las ip que he usado (192.168.100.30 y 192.168.100.40) o están dentro del rango permitido de IP privada. Entonces he modificado el Vagrantfile para añadir nuevas IP que estén dentro del rango de IP privada. Lo bueno de esto es que no va a afectar a acceso a internet en las máquinas a crear. Solucionado el problema, las levantamos y deberá de aparecer como la siguiente foto:

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

Para comprobar el estado de dichas máquinas creadas, lo veremos con el siguiente comando:

|  |
| --- |
| adminserver@adminserver:/proyecto/vagrant$ **vagrant status** |

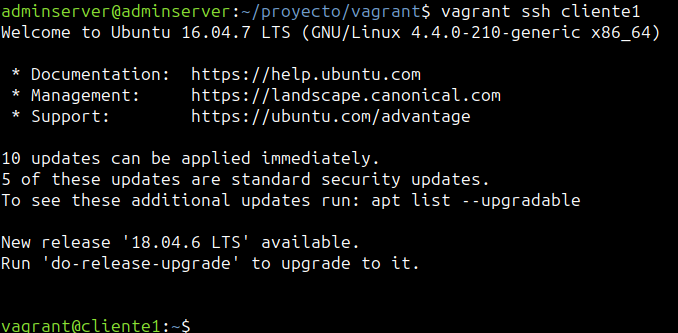
Texto

El contenido generado por IA puede ser incorrecto.

Ya tendríamos nuestras máquinas levantadas y funcionales, pero queremos comprobar si se pueden comunicar entre ellas para facilitar todas las futuras tareas. Esto se hará a **través de SSH**, accediendo a cada máquina y comprobando si hay comunicación entre ellas. SSH funcionará con un método de acceso sin contraseña que se basa en el uso de la clave pública para identificar a usuarios.

Se pondrá el siguiente comando por cada máquina a la que queramos conectarnos con SSH y entraremos (en este caso solo voy a poner el ejemplo de conexión de una de ellas):

|  |
| --- |
| adminserver@adminserver:/proyecto/vagrant$ **vagrant ssh cliente1** |



Por último, en este apartado, también será necesario configurar las claves SSH para que Ansible después pueda usar los entornos **sin necesidad de contraseña**. Puede parecer un proceso parecido al anterior, pero la diferencia entre ellos dos es que uno sirve para verificar que la máquina controladora tiene acceso a los entornos generados. Esta que vamos a realizar, es para que Ansible pueda conectarse a las máquinas virtuales sin contraseña y pueda aplicar las tareas que acabamos de automatizar.

Primero generaremos una clave pública y otra clave privada y omitiremos la información que nos pida pulsando intro (incluso si nos pide password)

|  |
| --- |
| adminserver@adminserver:/proyecto/vagrant$ **ssh-keygen** |

Con esta clave que acabamos de generar, la copiaremos y la pegaremos a cada máquina virtual que hemos creado. Hay que saber la ip de cada máquina para hacer esto. Si nos pidiera información con respecto al usuario o la contraseña pondremos la que suele venir por defecto al crear entornos virtuales con Vagrant (“vagrant”) y aceptamos la primera vez la huella digital. A continuación se muestra el comando que hay que poner para cada máquina virtual:

|  |
| --- |
| adminserver@adminserver:/proyecto/vagrant$ **ssh-copy-id vagrant@192.168.56.10** |

Probamos a acceder y, si podemos hacerlo sin medio de usar la contraseña, significa que estará todo bien configurado:

|  |
| --- |
| adminserver@adminserver:/proyecto/vagrant$ **ssh vagrant@192.168.56.10** |

No me dejaba entrar porque Vagrant solo permite el acceso por clave privada. Entonces busqué cual era la clave privada de Vagrant y la puse a prueba. Con este método si nos deja acceder a la máquina por SSH con Vagrant.

|  |
| --- |
| adminserver@adminserver:/proyecto/vagrant$ **ssh -p 2222 -i /home/adminserver/proyecto/vagrant/.vagrant/machines/cliente1/virtualbox/private\_key vagrant@192.168.56.10** |

Como podemos ver, se puede acceder de las dos maneras por SSH.

1. **CREACIÓN DEL ARCHIVO DE INVENTARIO *INVENTORY* PARA *HOST***

El siguiente punto de este proyecto se basa en la creación del archivo de inventario (llámalo “inventory” o llámalo “host”).

La función principal de este archivo es la de añadir las máquinas que generamos con Vagrant, para que Ansible sepa a que entornos virtuales debe de conectarse. Otras funciones que cumple este archivo son: decir que máquinas se conectan, como se llaman los grupos a los que pertenecen dichas máquinas, los usuarios que se tienen que usar, otras variables…

Se puede crear este archivo de dos formas: de forma manual o de forma automática. En este Caso usaremos la forma manual, para ir añadiendo a mano las máquinas a este archivo.

El primer paso es crear el archivo host, donde guardaremos toda la información para la recogida de información y configuración de los entornos virtuales creados. Se hará en el directorio de “ansible” y lo crearemos con el editor de nano. Se usa el siguiente comando para crearlo:

|  |
| --- |
| adminserver@adminserver:/proyecto/ansible$ **sudo nano hosts** |

Luego dentro, pondremos la siguiente información para que las máquinas virtuales queden registradas. Se explica a continuación el contenido del fichero “hosts”

|  |
| --- |
| [clientes]  192.168.56.10 ansible\_user=vagrant ansible\_ssh\_private\_key\_file=/home/adminserver/proyecto/vagrant/.vagrant/machines/cliente1/virtualbox/private\_key  192.168.56.20 ansible\_user=vagrant ansible\_ssh\_private\_key\_file=/home/adminserver/proyecto/vagrant/.vagrant/machines/ubuntu\_cliente2/virtualbox/private\_key |

Primero crea un grupo al que le hemos llamado “clientes” con las direcciones IP de las máquinas con las que estamos trabajando. También indica el usuario que tienen las máquinas, la clave privada de SSH para realizar la conexión son necesidad de contraseña, y por último dice que Ansible tiene que usar Python3.

Una vez explicado todo esto, guardamos los cambios y comprobamos que funciona y que Ansible es capaz de conectarse con las máquinas virtuales. Realizaremos un ping con Ansible para conectar con las máquinas virtuales de la siguiente manera:

|  |
| --- |
| adminserver@adminserver:/proyecto/ansible$ **ansible -i hosts clientes -m ping** |

*NOTA: -i hace referencia a inventory -m hace referencia a module*

Si nos aparece un mensaje como en el de la siguiente captura, será que hemos creado el archivo correctamente y ya permite almacenar las máquinas virtuales de forma manual para Ansible a través de su fichero de configuración y que el ping que acabamos de hacer se ha realizado de manera correcta.

Texto

El contenido generado por IA puede ser incorrecto.

Los WARNINGS simplemente se tratan de la detección de versiones modernas de Python em los entornos virtuales y Ansible los está reconociendo. No es nada importante ni que afectará a lo largo de la realización del proyecto ni de la automatización de las tareas.

1. **CREACIÓN DEL ARCHIVO DE CONFIGURACIÓN PLAYBOOK**

Después de verificar que las máquinas virtuales están correctamente instaladas con Vagrant y las hemos añadido al archivo de configuración host para que Ansible las reconozca y pueda trabajar con ellas, ahora crearemos el archivo de configuración Playbook.

El playbook en este proyecto se encargará de registrar las tareas que queremos que sean automatizadas para nuestros entornos virtuales a través de un archivo de configuración que crearemos y ajustaremos manualmente. Este archivo tendrá como extensión “YAML”, ya que lo utilizaremos para el diseño de archivo de configuración, para que sepa con el archivo hosts a que máquinas debe conectarse, con que permisos lo debe de hacer... Se crearán ese conjunto de reglas que van a ir ejecutándose de una en una en orden descendente a través de la creación de unos roles. A continuación, explicaremos como se crea este archivo de configuración y qué significa cada uno de estos componentes.

Para realizar esto, nos situaremos en la carpeta de Ansible (que es donde tenemos todo guardado con respecto a esta herramienta) dentro del directorio del proyecto. Nos ubicaremos como en la imagen a continuación:

Texto

El contenido generado por IA puede ser incorrecto.

Dentro del directorio de Ansible crearemos el fichero Playbook en formato YAML añadiéndole la extensión con el siguiente comando:

|  |
| --- |
| adminserver@adminserver:/proyecto/ansible$ **sudo nano playbook.yml** |

Dentro de él escribiremos las siguientes órdenes para la automatización de las tareas, donde definiremos la información básica del playbook de Ansible:

|  |
| --- |
| # automatizacion simple  - name: Automatización de Roles  hosts: clientes  become: true  gather\_facts: true  roles:  - usuarios  - primermensaje  - firewall  - basededatos |

Ahora se explicará que forma cada parte de este archivo de configuración.

Primeramente, indicamos el nombre que queremos que tenga este primer playbook para que podamos identificarlo en un futuro con “name”. Luego indicamos el nombre que le asignamos a los entornos virtuales en el archivo “hosts”, con “become” le indicamos que se realizará todo mediante usuario con privilegios, y la última “gather\_facts” hace que se recoja la información necesaria de los entornos virtuales por defecto antes de ejecutar las tareas automatizadas. En roles tenemos los roles de las tareas que queremos automatizar y se irán ejecutando de arriba abajo. Una página donde se podrían sacar los códigos de algunos roles sería la siguiente: [AQUÍ](https://galaxy.ansible.com/ui/standalone/roles/). También se puede seguir el enlace: <https://galaxy.ansible.com/ui/standalone/roles/>. Los roles los crearemos posteriormente.

Para verificar que todo está escrito correctamente, pondremos el siguiente comando de verificación de sintaxis:

|  |
| --- |
| adminserver@adminserver:/proyecto/ansible$ **ansible-playbook playbook.yml --syntax-check -i hosts** |

Es normal que de error nada más crear el archivo, porque los roles están solo definidos, pero no creados de manera manual con su configuración. Pero lo haremos y lo probaremos después.

1. **CREACIÓN Y APLICACIÓN DE ROLES**

Una vez tenemos tanto el “inventory”, como el “playbook”, como los entornos virtuales generados con Vagrant… es hora de crear las tareas que queremos que se automaticen y apliquen a cada máquina virtual. Para ello recurriremos a los roles, que nos ayudarán a organizar esas tareas mencionadas antes.

Los roles que se van a crear en este proyecto son los siguientes:

* **Rol de Usuario:** En este rol se definirá tareas de creación y eliminación de usuarios
* **Rol de Primer Mensaje:** En este rol se mostrará un mensaje de bienvenida cada vez que el usuario acceda a una máquina virtual generada
* **Rol de Firewall:** En este rol se habilitarán y deshabilitarán ciertos puertos que se indiquen con UFW.
* **Rol de Base de Datos:** En este rol se instalará automáticamente el servicio de MaríaDB (MySQL) para crear automáticamente una base de datos desde cero y también creará dentro de esa base de datos un usuario con permisos sobre esa base de datos.

Para la creación de estos roles, los guardaremos en el directorio llamado “roles” dentro del directorio padre de Ansible. Cada archivo guardado en esa carpeta, vamos a crear 4 carpetas para cada rol y dentro de cada carpeta para cada rol estará los archivos de configuración donde se almacenarán la o las tareas automatizadas que configuraremos a continuación.

Para crear los 4 roles, nos situaremos en el directorio de “ansible” y desde ahí pondremos lo siguiente:

|  |
| --- |
| adminserver@adminserver:/proyecto/ansible$ **ansible-galaxy init roles/usuarios** |

|  |
| --- |
| adminserver@adminserver:/proyecto/ansible$ **ansible-galaxy init roles/primermensaje** |

|  |
| --- |
| adminserver@adminserver:/proyecto/ansible$ **ansible-galaxy init roles/firewal** |

|  |
| --- |
| adminserver@adminserver:/proyecto/ansible$ **ansible-galaxy init roles/basededatos** |

El árbol de directorios donde guardamos nuestro proyectos con estos cambios quedaría de la siguiente manera:

Texto

El contenido generado por IA puede ser incorrecto.Imagen que contiene Diagrama

El contenido generado por IA puede ser incorrecto.

**Texto

El contenido generado por IA puede ser incorrecto.Texto

El contenido generado por IA puede ser incorrecto.**

* 1. **ROL DE USUARIOS**

Este rol lo vamos a configurar para que cree nuevos usuarios en los entornos virtuales cada vez que se quiera y que también pueda eliminarlos cuando no los necesite.

Para plantear la idea: vamos a crear dos usuarios (usuario1 y usuario2) y le diremos que esos usuarios los cree en una tarea varias veces, pero con diferentes datos. También crearemos otra regla para que elimine esos mismos usuarios si fuera necesario. No se han añadido contraseñas para que el proyecto sea mas llevadero, pero se podrían añadir en el bucle “loop” después del nombre de usuario.

Abriremos su correspondiente fichero de configuración en el directorio donde se encuentra el rol y pondremos la siguiente información (abrir el archivo .yml dentro del directorio task en el rol):

|  |
| --- |
| adminserver@adminserver:/proyecto/ansible...$ **sudo nano main.yml** |

El resultado de este archivo debe de quedar de la siguiente manera:

|  |
| --- |
| - name: Crear usuarios de prueba  user:  name: "{{ item.name }}"  state: present  shell: /bin/bash  loop:  - { name: 'usuario1' }  - { name: 'usuario2' }  - name: Eliminar usuarios de prueba  user:  name: "{{ item }}"  state: absent  loop:  - prueba1  - prueba2 |

Este fichero lo que hace, básicamente es: utilizar la variable “user” para gestionar las cuentas de usuario a la hora de crearlos y eliminarlos, también el bucle “loop” para que la tarea se haga más de una vez con los datos que se precisen, la variable “shell /bin/bash” menciona en qué tipo de terminal que va a tener el usuario. Para el borrado de usuarios se usa “state: absent”.

Guardaremos el archivo y lo comprobaremos cuando estén todos los roles creados para evitar fallos a la hora de su lanzamiento con los entornos virtuales de Vagrant.

* 1. **ROL DE PRIMER MENSAJE**

Continuamos con el siguiente rol, el de primer mensaje. Su funcionalidad se basará en la creación de un mensaje personalizado por el administrador que se muestre al iniciar sesión en la máquina virtual generada. El cliente cuando quiera acceder a esa máquina verá dicho mensaje. Su código contendrá el mensaje que querrémos mostrar, que en este caso dirá: “¡Bienvenido/a a la máquina generada por Vagrant. Espero que la disfrutes!"

Para crear y configurar este rol haremos como hemos hecho antes con el rol de usuarios, es decir, abriremos su fichero de configuración dentro del directorio de task (el archivo con extensión .yml) y dentro añadiremos la siguiente configuración (omito el comando para abrir el fichero porque ya aparece en el rol de arriba):

|  |
| --- |
| - name: añadir el primer mensaje de bienvenida a la máquina cliente  copy:  content: "¡Bienvenido/a a la máquina generada por Vagrant. Espero que la disfrutes!"  dest: /etc/motd |

Ahora se explicará lo que hemos añadido en el archivo de configuración: con “content” Indicaremos el mensaje que queremos que se muestre en la máquina. Con “dest” le indicaremos la ubicación donde se va a situar el mensaje dentro del sistema.

Después de tenerlo todo, lo guardaremos y más tarde lo comprobaremos junto a los demás roles. Podemos continuar con el proyecto.

* 1. **ROL DE FIREWALL**

En el tercer rol que vamos a crear y añadir, habilitaremos y/o deshabilitaremos puertos donde pasan o no conexiones de red, servicios… a través de la herramienta de UFW (un firewall o cortafuegos) específico para Ubuntu.

Vamos a poner un ejemplo para la automatización de esta tarea: habilitaremos el puerto TCP (3306) para que en el siguiente rol pueda permitir las conexiones a MariaDB y MySQL y poder automatizar la creación de una base de datos con Ansible, y el de SSH (22) que será el que se encargue de permitir las conexiones remotas que ya usamos (para verlo a modo ejemplo).

Comenzaremos abriendo el fichero de configuración “main.yml” dentro del directorio “task” y escribiremos y aplicaremos la siguiente configuración:

|  |
| --- |
| - name: instalación de UFW (por si no estuviera)  apt:  name: ufw  state: present  update\_cache: true  - name: habilitación de puertos TCP (3306) y 22 (SSH)  ufw:  rule: allow  port: "{{ item }}"  loop:  - 22  - 80  - name: Denegar el puerto FTP (21/TCP)  ufw:  rule: deny  port: "{{ item }}"  proto: tcp  loop:  - 21  - name: activar UFW  ufw:  state: enabled |

En este archivo indicamos una serie de pasos para una tarea específica: el primer párrafo comprueba si el paquete en Ubuntu de UFW está instalado. Si está instalado menciona que ya está instalado, pero si no está instalado lo instala automáticamente en la máquina.

El segundo párrafo del fichero permite, en este caso, que con “allow” que se abran los puertos 3306 y 22 a través de un bucle “loop”.

El tercer párrafo hace que se bloquee todo el acceso al puerto de transporte de mensajes FTP. No podrá haber tráfico en dicho puerto.

Y el cuarto y último párrafo activa las reglas que crea UFW para que se habiliten o deshabiliten los puertos que indicamos. Si se quiera denegar algún puerto en específico, habría que crear otra regla con la misma estructura de “allow”, pero indicándole “deny”.

Haremos como hemos estado haciendo todo este tiempo, guardaremos el archivo y más tarde realizaremos la comprobación junto a los demás roles.

* 1. **ROL DE BASE DE DATOS**

Llegamos al último rol, el llamado base de datos. Con este rol conseguiremos que en cada máquina virtual se creen base de datos automáticamente donde podremos trabajar directamente sobre la base que configuraremos y ahorrarnos unos pasos gracias a esta tarea automatizada.

Su tarea será instalar y configurar el servicio de MariaDB (con uso de MySQL) para su posterior creación de base de datos (pondremos una de demostración), y también tocaremos el tema de usuarios y permiso para esa respectiva base de datos que habremos creado.

Antes de usar este rol, deberemos de tener instalado el paquete de “python3-pymysql”.

El archivo de configuración contendrá la siguiente información:

|  |
| --- |
| - name: Instalación de MariaDB  apt:  name: mariadb-server  state: present  update\_cache: yes  - name: Iniciar el servicio MariaDB  service:  name: mariadb  state: started  enabled: yes  become: yes  - name: Crear usuario administrador en MariaDB  community.mysql.mysql\_user:  name: administrador  password: adminpass123  priv: '\*.\*:ALL'  host: localhost  state: present  login\_user: root  login\_password: unodostrescuatro  - name: Crear la base de datos  community.mysql.mysql\_db:  name: plantilla  state: present  login\_user: administrador  login\_password: adminpass123  become: yes  - name: Asignar privilegios al usuario sobre la base de datos  community.mysql.mysql\_user:  name: administrador  password: adminpass123  priv: plantilla.\*:ALL  state: present  login\_user: root  login\_password: unodostrescuatro |

La configuración que hemos creado es la siguiente: en la primera parte instalaremos el servicio de MariaDB en todas las máquinas virtuales (pero si está ya instalado lo omitirá e indicará que ya está instalado con “update\_cache” : yes”.

En la segunda parte iniciaremos el servicio de MariaDB (el de la base de datos) y le indicaremos para que se inicie siempre que se enciendan nuestros entornos virtuales con “state : started” y “enabled : true”.

En la tercera parte, crearemos un usuario llamado “administrador” con su contraseña “unodostrescuatro”. A este usuario de plantilla le aplicaremos todos los permisos para que pueda usar libremente la base de datos plantilla (cuando se esté manipulando esa base de datos se podrán crear más usuarios y gestionarlos al gusto o necesidad). También podrá conectarse desde cualquier IP a la base de datos generada automáticamente con “host: %”.

En el paso número cuatro del fichero de configuración crearemos la base de datos de plantilla con la que podremos trabajar posteriormente cada vez que queramos crear una nueva base de datos. En este caso le pondremos un nombre genérico para saber que es la plantilla base y podremos editar después cuando estemos trabajando sobre ella con el nombre “plantilla”.

Y por último, le daremos los privilegios que sean necesarios para que pueda trabajar sobre la plantilla de esa base de datos.

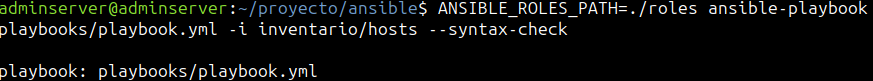
Cuando hayamos realizado todo, guardaremos el archivo y a continuación comprobaremos todos los roles.

## **8.5 COMPROBACIÓN DE ROLES CREADOS**

Ya tenemos todos los roles creados, pero hay que verificar si esos roles creados están bien creados y configurados en nuestra máquina controladora. Este comando es para la sintaxis principalmente. Para ello nos situaremos en el árbol de directorios de nuestro proyecto al nivel del directorio llamado “ansible” y ejecutaremos el siguiente comando en el terminal:

|  |
| --- |
| adminserver@adminserver:/proyecto/ansible$ **ANSIBLE\_ROLES\_PATH=./roles ansible-playbook playbooks/playbook.yml -i inventario/hosts --syntax-check** |

Y nos tendrá que aparecer de resultado algo como en la siguiente imagen:

****

1. **LANZAMIENTO DE CONFIGURACIONES DE ANSIBLE**

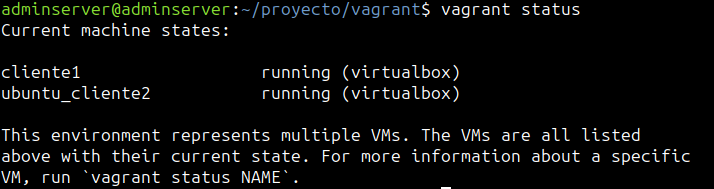
Cuando ya tenemos todas las máquinas generados junto con sus archivos y junto a sus roles o tareas automatizadas, se tienen que aplicar estas tareas o roles a los entornos virtuales para prepararlos correctamente y que puedan realizar esas tareas. Este paso es muy importante si de verdad queremos terminar el tutorial del proyecto de manera efectiva y sin fallos.

Para empezar a lanzar las configuraciones, evidentemente tienen que estar las máquinas virtuales en funcionamiento, ya que si no pudiéramos acceder a ellas no nos podríamos conectar y revisar y aplicar la configuración creada en la máquina controladora. Nos dirigimos al directorio de “vagrant” para iniciarlas y cuando ya estemos allí situados, pondremos el siguiente comando para iniciar las máquinas:

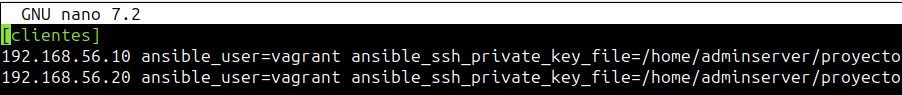
|  |
| --- |
| adminserver@adminserver:/proyecto/vagrant$ **vagrant up** |

Este proceso no es inmediato y tardará un par de minutos, pero podremos asegurar el encendido de las máquinas clientes hecho a través de un terminal. Para verificar su encendido ejecutaremos este comando. Nos deberá aparecer algo como la siguiente imagen:

|  |
| --- |
| adminserver@adminserver:/proyecto/vagrant$ **vagrant status** |



Continuaremos con el fichero de configuración *inventory* o inventario o *host*. En este paso comprobaremos que los entornos virtuales generados y encendidos aparecen y están bien clasificados en este archivo. Para ello nos dirigiremos al directorio de ansible y al subdirectorio de inventario. Después abriremos el archivo con el editor nano y tendremos que tener un resultado como:



Texto

El contenido generado por IA puede ser incorrecto.

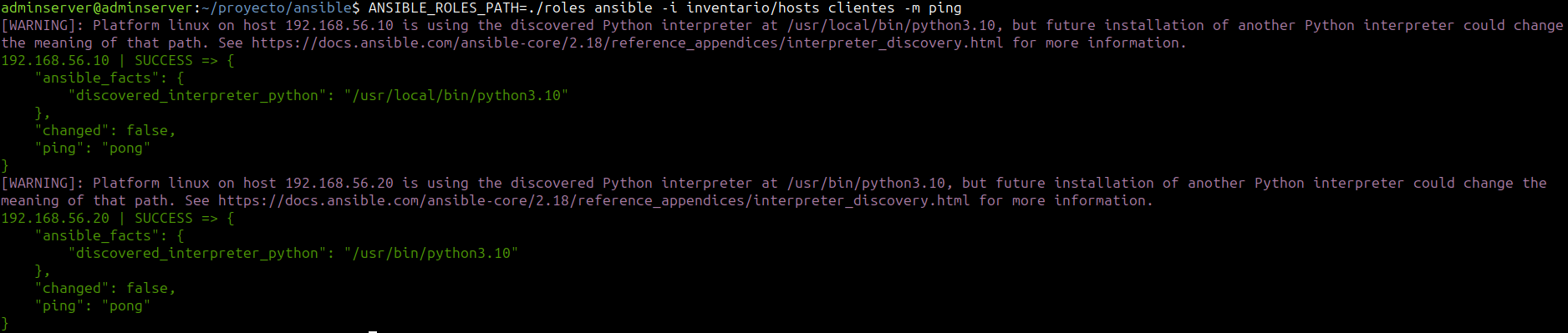
*(las imágenes han tenido que ser capturadas así ya que si realizo la captura completa no se visualiza el comando)*

Si tenemos todo tal cual así, será que lo estamos haciendo bien. ¡Enhorabuena!

Ahora comprobaremos si nos podemos conectar a los entornos creados de manera remota por a través de las claves que contiene ese fichero y también por SSH. Para eso saldremos de este fichero anterior y nos dirigiremos al directorio de Ansible para ejecutar el siguiente comando.

|  |
| --- |
| adminserver@adminserver:/proyecto/ansible$ ANSIBLE\_ROLES\_PATH=./roles ansible -i inventario/hosts clientes -m ping |

Este comando recorrerá los archivos y verá si puede realizar la conectividad con las máquinas a través de ansible. Si funciona todo deberá de aparecerte algo así (resumiendo “ping : pong):



Y como paso final, ejecutaremos el listado de reglas de juego o *playbooks* de dos maneras:

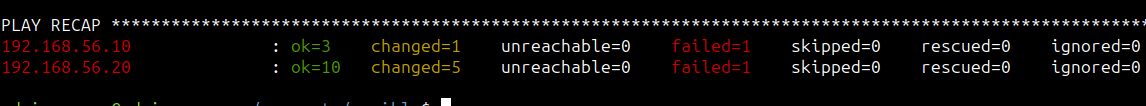
* La primera vez será una simulación, para no liarla en el paso final y no arriesgar.
* La segunda vez se hará ya de manera real, para terminar de aplicar todas las configuraciones a las máquinas.

En el modo simulación, ejecutaremos en el directorio actual donde estamos (ansible) este comando, que hará una comprobación general de todo, pero sin aplicar los cambios:

|  |
| --- |
| adminserver@adminserver:/proyecto/ansible$**ANSIBLE\_ROLES\_PATH=./roles ansible-playbook playbooks/playbook.yml -i inventario/hosts --check** |

Puede resultar muy familiar el comando porque es muy parecido a uno que hemos usado anteriormente en el proyecto para verificar la sintaxis del fichero playbook, pero no es del todo igual.

El resultado de este comando simulado deberá ser el siguiente (una comprobación de los roles básicamente):



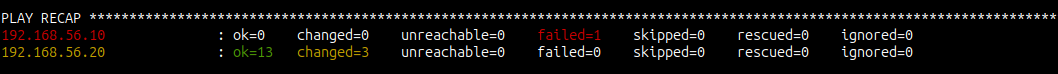
El error que aparece es debido a que al realizar la comprobación de roles en modo simulado aparece un error de permisos en la base de datos y en el firewall. Esto se solucionará al aplicar el playbook en la segunda opción (no la simulada, la real)

Ahora veremos que estos errores desaparecen al ejecutarlo en modo real digamos.

En el modo real, al igual que en simulado, se inicia un comando diferente con la diferencia que se aplicarán los roles de forma real a los entornos virtuales con vagrant. Para poder hacer, escribiremos en el terminal el siguiente comando y lo ejecutaremos:

|  |
| --- |
| adminserver@adminserver:/proyecto/ansible$ **ANSIBLE\_ROLES\_PATH=./roles ansible-playbook playbooks/playbook.yml -i inventario/hosts** |

Como resultado final e indicando que hemos realizado todo el proceso correctamente, el resultado final se deberá ver como en la siguiente captura:



Ese error es el mismo que nos salía anteriormente por las versiones de Python, al incompatibilizarse o dar problemas con las antiguas versiones de las máquinas generadas. Ese error no afectará al funcionamiento de los roles o tareas automatizadas creadas.

Ahora en el “Anexo \_II\_Manual\_de\_Usuario” veremos cómo acceder a estas configuraciones impartidas en las máquinas generadas y como usarlas.

Diagrama

El contenido generado por IA puede ser incorrecto.

I.E.S. “ÁGORA”

Dpto.: Informática

Ciclo Formativo de G. Superior: Administración de Sistemas Informáticos y Redes

ANEXO II : Manual de Usuario

Imagen que contiene Icono

El contenido generado por IA puede ser incorrecto.

Autor o autores: Carla Galindo Corchero

Tutor: Maria Francisca Roncero Holgado

Cáceres, a 7 de abril 2025

**ÍNDICE**

[**1.** **INTRODUCCIÓN** …………………………………………………………………………………………………………….60](#_Toc198282857)

[**2.** **REQUISITOS NECESARIOS** ……………………………………………………………………………………………… 60](#_Toc198282858)

[**3.** **ESTRUCTURA DEL ARBOL DE DIRECTORIOS** *…..*](#_Toc198282859) …………………………………………………………….. 61

[**4.** **LEVANTAR ENTORNOS VIRTUALES GENERADAS *………………………………………………………….*** 63](#_Toc198282860)

[**4.1** **MEDIANTE EL TERMINAL**……](#_Toc198282861)…………………………………………………………………………..64

[**4.2** **MEDIANTE VIRTUALBOX**….…………………………………………………………………………….66](#_Toc198282862)

[**5.** **PRUEBA DE LOS ROLES (TAREAS AUTOMATIZADAS)…………………………………………………….** 70](#_Toc198282863)

1. **INTRODUCCIÓN**

En este manual se va a explicar el procedimiento que tiene que llevar a cabo el usuario para usar las máquinas virtuales que genera el administrador con Vagrant y podrá hacer uso de las tareas automatizadas con Ansible directamente dentro de ellas.

Se explicarán también las dos formas en las que nos podemos conectar a estas máquinas y usar sus funcionalidades. El usuario podrá conectarse a estos entornos virtuales mediante el programa de VirtualBox o mediante el terminal de la propia máquina controladora. Una vez dentro de los clientes, comprobar de forma manual la funcionalidad de las tareas automatizadas.

Para tener estos entornos disponibles y continuar con este manual, se tendrán que tener todos los pasos realizados anteriormente en el “ANEXO\_I\_Manual\_de\_Administrador” y tener todas las configuraciones correctas.

1. **REQUISITOS NECESARIOS**

Necesitaremos unos requisitos para poder ejecutar el resultado final del proyecto. Entre ellos se encuentran los siguientes:

* **Sistema Operativo**: Necesitaremos disponer de un sistema operativo que actúe como máquina controladora donde se realizará todo el proceso del proyecto y, con el cual, el administrador se encargará de crear todas las máquinas y configuraciones para las máquinas clientes que usaran los usuarios finales.

En el caso de este proyecto se usará Ubuntu Server 22.04, pero se puede usar cualquier otra versión que sea compatible con las herramientas que se han utilizado, e incluso usar otro tipo de sistema operativo como, por ejemplo, Windows.

* **Herramientas de trabajo:** Las herramientas principales que tenemos que tener instaladas son las siguientes:
* **VirtualBox:** Será la plataforma de virtualización que hemos elegido y que se usará para ejecutar tanto la máquinas controladora, como las máquinas generadas por Vagrant para los usuarios finales.
* **Vagrant:** Esta herramienta será necesaria para la generación de los entornos virtuales que usarán los usuarios finales posteriormente. Se encargará tanto de la creación, configuración y gestión de dichos entornos.
* **Ansible:** Por último tenemos este software, que automatizará las tareas que añadamos a las máquinas clientes para que los usuarios tengan un espacio onde trabajar de manera más efectiva y eficiente. Conseguiremos mas facilidad, flexibilidad y comodidad a la hora de usar las máquinas por las configuraciones que le apliquemos desde la máquina controladora
* **Conexión a internet:** Es fundamental contar con conexión a internet para poder llevar a cabo todo el proceso de la realización del proyecto, sobre todo para la instalación de herramientas, paquetes, y dependencias necesarias.

Vagrant usará la conexión a internet para la descargar de las imágenes base, que contendrán los sistemas operativos. Ansible también hará uso de la conexión a internet para instalar los paquetes y dependencias necesarias mencionadas anteriormente

1. **ESTRUCTURA DEL ARBOL DE DIRECTORIOS**

Para entender mejor la configuración interna de las máquinas clientes con las que va a trabajar el usuario final, se explicará la estructura del árbol de directorios ubicada en la máquina controladora

Este árbol está organizado en directorios que separan las diferentes herramientas que hemos comentado antes para tener una mejor visión de cómo funciona todo el proceso.

A continuación se mostrará a modo esquema dicho árbol y el contenido de cada uno de los directorios que están desglosados por niveles, desde el directorio raíz hasta las hojas del último nivel:

Texto

El contenido generado por IA puede ser incorrecto.

El directorio raíz donde se aloja todo este árbol de directorios se le ha llamado “proyecto”, un nombre simple y entendible para lo que vamos a hacer. Este guarda toda la información con respecto a ficheros, configuraciones, paquetes… de sus nodos hijos.

Dentro de ese directorio raíz nos dividimos en tres subdirectorios: ansible, vagrant y scripts.

Comenzando por ansible, almacena todo o relacionado con la automatización de tareas mediante los ficheros como “inventory” (para digamos denominar las máquinas generadas por vagrant), el archivo de configuración “playbook” (que se encarga de resumir que tareas se van a ejecutar en las máquinas de manera automática), y el archivo de “roles” (donde se almacenan diferentes subdirectorios dependiendo del número de tareas que se le quieran aplicar a los entornos virtuales). Cada rol contiene diferentes ficheros para configurar las tareas con diferentes opciones.

Continuamos con vagrant, en el cual se almacenan dos únicas cosas: el archivo “vagrantfile” y la imagen base. Cuando hablamos de “vagrantfile” hablamos del fichero de configuración que se usa para especificar las características que queremos que contengas las máquinas virtuales a generar (por ejemplo: opciones del hardware, nombre del equipo, redes, puertos, sistema operativo…).

La imagen base que también entra en este directorio es la imagen del sistema operativo que vamos a importar al archivo que hemos explicado antes. Define el sistema operativo que queremos que contengan los entornos virtuales. Podemos crear todas con la misma imagen base o podemos definir cada máquina con una imagen base diferente.

Para finalizar esta parte, tenemos un directorio llamado scripts. En el caso de este proyecto no se ha llegado a usar porque a la hora de realizar las configuraciones de roles y máquinas se ha utilizado Bash de Linux. Ese directorio está disponible por si se quieren programar o definir todas estas configuraciones mediante PowerShell, Python…

1. **LEVANTAR ENTORNOS VIRTUALES GENERADAS**

A partir de este punto, el usuario comenzará ya a consumir esta idea de proyecto, probar las máquinas virtuales y comprobar si estas funcionan correctamente y con las tareas que se le han aplicado desde la máquina controladora.

El proceso de levantar las máquinas las podremos llevar a cabo mediante dos formas diferentes: Mediante el terminal y mediante el software de VirtualBox. Ambas opciones se deben hacer en este caso desde la máquina controladora, porque en este caso se ha decidido planificar de esa manera.

Ahora vamos a explicar cómo se haría cada una de estas opciones mencionadas antes y su proceso:

* 1. **MEDIANTE EL TERMINAL**

Esta es una de las posibilidades que tenemos para iniciar o arrancar las máquinas. Para esto únicamente necesitaremos el terminal de la máquina controladora. A diferencia con el otro método, es que este requiere de comandos y de la conexión SSH para poder realizar la conexión a los entornos.

Para arrancar las máquinas virtuales desde el terminal, primero deberemos de iniciar la máquina controladora donde tenemos todo nuestro proyecto guardado o todas nuestras configuraciones aplicadas y abrir su terminal.

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

Ya teniendo todo esto listo, nos dirigiremos al árbol de directorios de nuestro proyecto y nos iremos, específicamente, al directorio donde tenemos guardado todo lo relacionado con “vagrant”. Una vez dentro simplemente tendremos que arrancar las máquinas clientes de manera general con el comando que se muestra a continuación en la imagen:

Texto

El contenido generado por IA puede ser incorrecto.

Al aplicar ese comando, todas las máquinas comenzarán a encenderse de manera automática. Este proceso no es inmediato y puede llevar alrededor de 3 a 5 minutos de duración que se enciendan completamente, ya que la máquina controladora tiene que encender más de un entorno y cada uno conlleva la preparación de sus configuraciones individuales o conjuntas.

Cuando se carguen todas las máquinas, comprobaremos si se han encendido de manera correcta. La manera para ver esto es escribir en el terminal el siguiente comando:

Texto

El contenido generado por IA puede ser incorrecto.

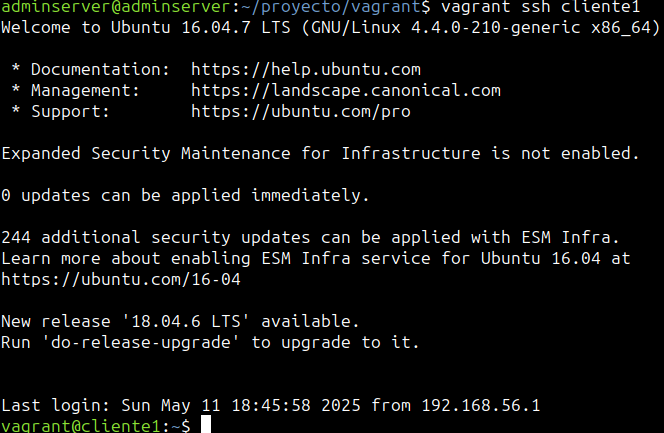
Como podemos observar, las dos máquinas que se han creado para este proyecto están en funcionamiento. También nos indica el nombre que les hemos añadido a los host en el archivo “vagrantfile” y el proveedor de virtualización que las ejecuta (que en este caso se trata de VirtualBox).

Ya que tenemos los entornos virtuales encendidos podremos acceder a ellos. Solo se podrá acceder si el administrador ha realizado las configuraciones de manera correcta. La peculiaridad de este tipo de acceso es que trabajamos con el método SSH para conectarnos a cada máquina, la cual resulta fácil cambiar de una máquina a otra, porque solo bastará con cerrar la conexión con una e iniciarla con otra.

Para conectarnos a una máquina, deberemos estar en el directorio en el que estamos actualmente para encender las máquinas. Cada conexión sigue el mismo patrón. El patrón es el siguiente:

|  |
| --- |
| vagrant ssh “nombre de la máquina cliente” |

Para que la comprensión de este comando se vea de una manera más clara, vamos a poner un ejemplo: nos vamos a conectar a una de las máquinas llamada “cliente1”, ver cómo se conecta y ver como realiza esa conexión mediante SSH. Se muestra la imagen de ejemplo abajo:



Y para salir la sesión, simplemente se sale con exit, apareciendo el mensaje de confirmación de salida “logout”:

Imagen que contiene objeto, reloj, conjunto, medidor

El contenido generado por IA puede ser incorrecto.

* 1. **MEDIANTE VIRTUALBOX**

El otro método con el que se puede acceder a las máquinas virtuales sin tener que usar el terminal del sistema es mediante el entorno de virtualización de VirtualBox. Con este método se pueden acceder a las máquinas generadas por Vagrant a través de este software para acceder a su entorno gráfico, cosa que con el método anterior no se puede hacer.

Esto nos ofrece bastantes ventajas con respecto al método de arranque anterior: accesibilidad a herramientas gráficas, múltiples usos de configuraciones del sistema, mayor facilidad de uso…

Para poder llevar a cabo este método, seguiremos los siguientes pasos:

Primeramente encenderemos, como es evidente, la máquina controladora donde tenemos todo el contenido y configuraciones del proyecto y después de eso abriremos el terminal.

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

Una vez dentro de todo esto, nos dirigiremos al directorio de vagrant en el árbol de directorios de proyecto para levantar las máquinas virtuales todas al mismo tiempo. Se hará como se muestra en la imagen:

Texto

El contenido generado por IA puede ser incorrecto.

Para comprobar que hemos hecho todo esto de manera correcta y verificar que los entornos virtuales están encendidos y corriendo, comprobaremos su estado con el siguiente comando:

Texto

El contenido generado por IA puede ser incorrecto.

Cuando nos aparece esto, quiere decir que las máquinas están encendidas, en funcionamiento, y ya están disponibles para que las puedan utilizar los clientes. Nos aparece el nombre de cada host que aparece en el archivo “Vagrantfile”, su estado actual, y el proveedor de virtualización que lo lleva a cabo (que en el caso de este proyecto es VirtualBox)

Después de realizar todos estos pasos, podremos minimizar el terminal y abrir el software de VirtualBox. Nos dirigiremos al inicio, y en sistema elegimos VirtualBox:

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.

Veremos que al iniciar el programa, ya están las máquinas encendidas y en ejecución. Esto comprueba que lo que hemos realizado anteriormente ha funcionado. Deberá de verse algo como lo siguiente:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Ahora nos entramos en cualquiera de ellas para asegurarnos de que funcionan tal y como las hemos configurado dándole a la flecha verde donde indica “mostrar”. Una vez dado al botón, simplemente esperaremos a que la máquina nos lleve al apartado de login.

Al crear los entornos virtuales con Vagrant, el usuario y la contraseña de las máquinas por defecto será “vagrant”, pero se puede modificar como lo haríamos de forma normal una vez dentro de la máquina virtual para que cada cliente posea su usuario y su contraseña.

Texto

El contenido generado por IA puede ser incorrecto.

1. **PRUEBA DE LOS ROLES (TAREAS AUTOMATIZADAS)**

Al tener las máquinas activas y ver que todas las configuraciones de Vagrant han surgido efecto, procederemos a probar los roles creados con Ansible para ver si se han aplicado correctamente y ver si la idea es factible.

Gracias a hacer esto, podremos ver si los roles funcionan y ver cómo se comportan de manera individual y corregir los posibles errores que puedan surgir

Vamos a realizar la prueba de los roles introduciendo los comandos correspondientes en el terminal asociados a cada rol y ver si, según las reglas de Ansible, funcionan como se espera.

En este caso vamos a hacer la comprobación de 2 roles de 4 que hemos aplicado: rol de “usuarios” y rol de “base de datos”.

El primer rol para comprobar será el de “usuarios”. En la tarea de la máquina controladora le dijimos que creara dos usuarios a cada máquina virtual, llamados “usuario1” y “usuario2”. Para ver si se han creado correctamente vamos a hacer una búsqueda y ver si aparecen. Se harán como en la captura:

Texto

El contenido generado por IA puede ser incorrecto.

Como segundo rol a comprobar, tenemos el de base de datos. Esta tarea automatizada permitía iniciar el servicio de MariaDB, crear los usuarios y bases de datos adecuados, y darle los permisos que les corresponda.

Primero iniciamos el servicio de MariaDB y vemos si está activo. Si lo está lo dejamos así, pero si no lo está se enciende con:

|  |
| --- |
| **systemctl start mariadb.service** |

El servicio activo se ve de la siguiente manera:

Texto

El contenido generado por IA puede ser incorrecto.

Cuando está todo activo, procederemos a crear un usuario con el que trabajaremos en la base de datos. Después de crearlo le daremos todos los permisos necesarios para que pueda acceder a todas las funcionalidades posibles y tenga privilegios sobre todas las opciones y las configuraciones (todo esto se hará dentro del usuario root.

Hay que destacar que el usuario y la contraseña que le apliquemos a este usuario va a contener la misma información que añadimos a la tarea automatizada en el rol, para que todo funcione.

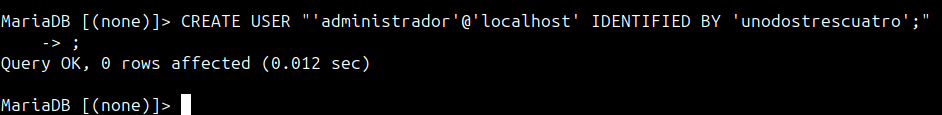
Entramos en MariaDB y añadimos los siguientes comandos:

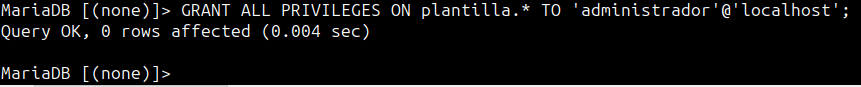
1. Nos entramos a MariaDB mediante root:

Texto

El contenido generado por IA puede ser incorrecto.

1. Una vez dentro, procederemos a crear el usuario administrador, con todos los privilegios sobre la base de datos ya creada por defecto en los roles.





Después de crear el usuario y darle los privilegios correspondientes, comprobaremos que la base de datos y el usuario que acabamos de crear están disponibles y realizan su función. Para ello:

1. Mostramos las bases de datos que tenemos disponibles.

Texto

El contenido generado por IA puede ser incorrecto.

1. Ahora mostramos los usuarios que existen en el sistema. Como podemos ver, ya había un usuario administrador creador de manera local porque es la que se a creado con la tarea automatizada de Ansible.

Imagen que contiene Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

1. Verificamos los permisos del usuario que acabamos de crear, para ver si tiene todos los permisos.

Imagen que contiene computadora, teclado, computer, grande

El contenido generado por IA puede ser incorrecto.

Ya hemos comprobado que el rol de base de datos ha creado correctamente la base de datos plantilla (donde el usuario de la máquina virtual podrá trabajar desde cero si lo desea) y junto a esto el usuario administrador en localhost con todos los privilegios.

Con esto, ya tendríamos todo lo necesario para poder implantar esta idea de proyecto en espacios donde se precisen muchas máquinas virtuales, o donde se quiera implantar.