

# AUTOMATIZACIÓN DE SERVICIOS EN RED CON ANSIBLE



## AUTOR

David Bañón Ortiz

## TUTOR

Jose Manuel Carreres Paredes

## TUTOR DE PROYECTO

Eduardo Tellechea Amesti

CICLO FORMATIVO DE GRADO SUPERIOR  
ADMINISTRACIÓN DE SISTEMAS INFORMÁTICOS EN RED



Esta obra está bajo una licencia Reconocimiento-Compartir bajo la misma licencia 3.0 España de Creative Commons. Para ver una copia de esta licencia, visite <http://creativecommons.org/licenses/by-sa/3.0/es/> o envíe una carta a Creative Commons, 171 Second Street, Suite 300, San Francisco, California 94105, USA.

## INDICE

1 INTRODUCCIÓN.....	3
2 INTRODUCCIÓN EN INGLÉS.....	4
3 ALCANCE DEL PROYECTO.....	5
4 ESTUDIO DE VIABILIDAD.....	6
4.1 Estado actual del sistema.....	6
4.2 Requisitos del cliente.....	6
4.3 Posibles soluciones.....	7
Terraform:.....	7
Ansible:.....	7
¿Que son tareas Ad hoc?.....	7
Ejemplos de tareas “ad hoc”:.....	8
4.4 Solución elegida.....	8
4.5 Planificación temporal de las tareas del proyecto.....	9
4.6 Planificación de los recursos a utilizar.....	10
5 ANÁLISIS.....	11
5.1 Requisitos funcionales.....	11
5.2 Requisitos no funcionales.....	11
6 PREVENCIÓN DE RIESGOS LABORALES EN INFORMÁTICA.....	12
7 DISEÑO.....	14
7.1 Disposición gráfica del proyecto.....	14
8 IMPLEMENTACIÓN DEL PROYECTO.....	15
8.1 Configuración de máquinas virtuales para la demostración.....	16
8.1.1 Configuración de interfaces.....	18
8.1.2 Configuración fichero sudoers.....	19
8.2 Configuración SSH entre nodos.....	20
8.3 Configuración de ansible y su inventario en un nodo cualquiera.....	23
8.4 Ejecución de playbook para instalar Ansible en el resto de nodos.....	25
8.5 Configuración del resto de inventarios.....	27
8.6 Playbook de instalación de paquetes sobre todos los nodos.....	28
8.7 Creación de scripts encargados de la automatización.....	30
8.7.1 Script de Nmap.....	30
8.7.2 Script DHCP.....	31
8.7.3 Scripts DNS.....	32
8.8 Creación de playbook de automatización de servicios en red.....	36
8.9 Creación de script de automatización de red constante.....	37
8.10 Comprobación de funcionamiento en nodo cualquiera.....	38
9 PROPUESTA DE MEJORA.....	43
10 CONCLUSIONES.....	44
11 FUENTES.....	45
12 ANEXOS.....	46

## 1 INTRODUCCIÓN

Actualmente, la administración de servicios en red es fundamental en cualquier empresa. A medida que las organizaciones se expanden y adoptan tecnologías avanzadas, la gestión manual de direcciones IP y nombres de dominio se vuelve insostenible y propensa a errores. Una gestión eficaz de estos servicios permite a las empresas implementar aplicaciones a nivel interno, manteniendo una gestión privada y segura de la información delicada de los clientes.

La automatización de estos servicios resuelve de manera inmediata los tiempos de espera que pueden surgir al gestionar el acceso de los empleados a las herramientas internas. Esto se logra al proporcionar una IP estática desde el primer momento con DHCP y una resolución de nombres mediante DNS, facilitando la comunicación entre el administrador de sistemas, el servidor y los empleados.

Este enfoque también mejora la escalabilidad de la empresa, asegurando que, independientemente del número de empleados, los servicios siempre estén operativos gracias a un sistema de alta disponibilidad. Esto garantiza una gestión adecuada y segura de cada equipo en la red.

Para alcanzar este objetivo, este proyecto se centra en desarrollar un sistema utilizando Ansible y un conjunto de herramientas, protocolos y scripts en lenguajes de programación como Bash, Python y YAML, para la correcta automatización y gestión de los servidores encargados de proporcionar los servicios en red, denominados nodos. Todos estos elementos se implementarán en diversos equipos con Ubuntu para garantizar que, en caso de fallo o avería de alguno, el servicio no quede inactivo. Además, no requiere de grandes recursos de hardware, lo que hace que el proyecto sea sostenible y asequible para cualquier empresa.

## **2 INTRODUCCIÓN EN INGLÉS**

Currently, network service management is fundamental in any company. As organizations expand and adopt advanced technologies, the manual management of IP addresses and domain names becomes unsustainable and prone to errors. Effective management of these services allows companies to implement applications internally, maintaining private and secure handling of sensitive client information.

The automation of these services immediately resolves the wait times that can arise when managing employee access to internal tools. This is achieved by providing a static IP address from the start and DNS name resolution, facilitating communication between the system administrator, the server, and the employees.

This approach also enhances the company's scalability, ensuring that, regardless of the number of employees, the services remain operational thanks to a high-availability system. This guarantees proper and secure management of each device on the network.

To achieve this goal, this project focuses on developing a system using Ansible and a set of tools, protocols, and scripts in programming languages such as Bash, Python, and YAML, for the correct automation and management of the servers providing network services, referred to as nodes.

All these elements will be implemented on various Ubuntu devices to ensure that, in the event of a failure or malfunction of any, the service does not become inactive. Moreover, it does not require significant hardware resources, making the project sustainable and affordable for any company.

### **3 ALCANCE DEL PROYECTO**

El propósito del proyecto es lograr una automatización de servicios mediante nodos de Ansible para garantizar el acceso a todos los usuarios de la red a los recursos y aplicaciones que se desarrollen a nivel interno por parte de una empresa. Esto se realizará instalando un sistema de alta disponibilidad en el que varios nodos, gracias a Ansible, al protocolo SSH, a la herramienta NMAP y a un conjunto de scripts desarrollados con bash y python, proporcionen de forma automatizada una reserva dinámica de los servicios DHCP y DNS.

A nivel de hardware, se utilizarán portátiles que se encargarán de actuar como nodos del servidor. A nivel de software, se utilizarán varios programas, que son:

- Ansible: Esta herramienta permitirá la ejecución de los scripts creados y la comprobación de paquetes en todos los nodos disponibles en el servidor.
- Nmap: Es una herramienta de escaneo de red, nos servirá para localizar todos los equipos de nuestra red y almacenarlos en un fichero para procesarlo posteriormente
- Servicios DHCP Y DNS: Utilizaremos ambos servicios para reservar tanto una dirección IP por equipo dentro del DHCP, como un nombre de resolución DNS.
- Scripts en Bash, Python y YAML: Gracias a la creación de scripts, realizaremos la automatización de los servicios DNS Y DHCP utilizando la lista de equipos extraída con Nmap.

## **4 ESTUDIO DE VIABILIDAD**

En esta fase, se llevará a cabo un análisis teniendo en cuenta tanto las circunstancias internas como externas de la empresa. Se analizarán las posibles soluciones, evaluando sus capacidades y los recursos disponibles. Para ello, se realizará una valoración del estado actual del sistema y de los requisitos del cliente.

Para realizar dicho proyecto, es necesario tener varios equipos disponibles que vayan a actuar de servidores, ya que serán los encargados de gestionar la automatización de servicios con Ansible. Será imprescindible disponer mínimo de dos equipos ya que la idea es gestionar un sistema de alta disponibilidad que permita la gestión constante de los equipos conectados a la red.

En la demostración práctica, se trabaja con un único equipo ya que sobre el mismo se virtualizan diferentes entornos para poder comprobar el correcto funcionamiento del proyecto.

### **4.1 Estado actual del sistema**

Actualmente, la mayoría de empresas utilizan un sistema híbrido entre manual y automático ya que se emplean una serie de scripts para dar de alta en el conjunto de herramientas y servicios a cada usuario de forma individual, esto hace los tiempo de espera más largos ya que en primer lugar se tiene que reconocer el equipo del empleado, posteriormente asignar un nombre de usuario y ya raíz de esos datos, Se tramitan los scripts mencionados. Además, los equipos nuevos, no suelen tener los programas instalados, lo que hace aún más lenta la integración de nuevos empleados.

## **4.2 Requisitos del cliente**

Cualquier empresa requiere de agilidad a la hora de gestionar el alta de los empleados en los sistemas, ya que, de esto depende que sus servicios se puedan proporcionar de forma eficiente.

Por esta misma razón, se trata de maximizar la automatización, dado que esta, facilitará la implementación de equipos y usuarios desde el primer día que se conecten a la red.

## **4.3 Posibles soluciones**

Hay múltiples programas que realizan funciones de automatización de servicios y coordinación entre nodos para disponer de alta disponibilidad. Algunas de estas herramientas están diseñadas para trabajar en la nube, mientras que otras pueden operar de forma local, permitiendo la conexión y gestión de equipos dentro de la misma infraestructura. Dos de las herramientas más destacadas en este campo son Terraform y Ansible.

### **Terraform:**

**Descripción:** Terraform, desarrollado por HashiCorp, es una herramienta que permite definir y aprovisionar infraestructura en múltiples proveedores de nube.

**Capacidades:** Con Terraform, puedes automatizar la creación y configuración de recursos en la nube, como servidores, bases de datos, redes y más. Es especialmente útil para gestionar entornos de infraestructura complejos y garantizar que todas las configuraciones sean consistentes y reproducibles.

### **Ansible:**

**Descripción:** Ansible es una herramienta de gestión de configuración y automatización que utiliza un enfoque sin agentes para gestionar nodos a través de SSH.

**Capacidades:** Ansible puede automatizar la configuración de sistemas, el despliegue de aplicaciones, la orquestación de servicios y la ejecución de tareas “ad hoc” en nodos locales o remotos. Utiliza archivos YAML llamados playbooks para definir las tareas de configuración y despliegue.

#### ***¿Que son tareas Ad hoc?***

Las tareas ad hoc en el contexto de Ansible y otras herramientas de automatización se refieren a aquellas tareas que se realizan de manera puntual y específica, en lugar de ser parte de un proceso automatizado recurrente o un conjunto de instrucciones predefinidas en un playbook o script. Estas tareas son ejecutadas directamente por el administrador cuando surge la necesidad de realizar una acción rápida y única.

#### ***Ejemplos de tareas “ad hoc”:***

**Reiniciar un servicio:** Si un servicio en un servidor se detiene inesperadamente, un administrador puede usar una tarea “ad hoc” para reiniciarlo.

**Actualizar un paquete:** Instalar o actualizar un paquete en uno o más servidores sin necesidad de escribir un playbook completo.

**Comprobación de estado:** Ejecutar comandos para verificar el estado de un sistema o servicio.

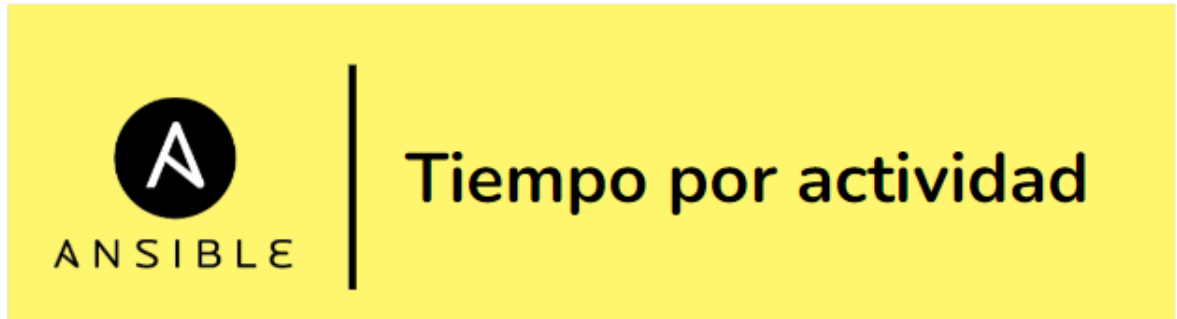
Ambas herramientas permiten a las organizaciones mejorar la eficiencia operativa y reducir errores humanos mediante la automatización de tareas repetitivas y complejas, ya sea en la nube o en infraestructuras locales.



#### **4.4 Solución elegida**

Para hacer más económica la solución, se realizará sobre Ansible ya que no se requiere de suscripciones a plataformas en la nube como Amazon Web Services. Además permite distribuir, tanto ejecuciones repetitivas gracias a los playbooks (como por ejemplo la detección de usuarios en la red combinando el uso de Nmap y scripts) como ejecuciones puntuales gracias a las tareas “ad hoc” (como por ejemplo la comprobación del estado de cada servidor). Gracias a Ansible es posible desarrollar este proyecto, llamado “Automatización de servicios en red con Ansible”.

#### 4.5 Planificación temporal de las tareas del proyecto



Tareas/Semana	1º Semana	2º Semana	3º Semana	4º Semana	5ºSemana
Configuración interfaces					
Configuración SSH					
Configuración de Ansible					
Creación de playbooks base					
Configuración de inventario por nodo					
Creación de scripts					
Comprobaciones de scripts					
Creación playbook final					
Creación script final					
Pruebas de funcionamiento					

Figura 1: Diagrama de Gantt.

#### **4.6 Planificación de los recursos a utilizar**

Para llevar a cabo el proyecto “Automatización de servicios en red con Ansible” se requiere de varios equipos con especificaciones adecuadas, pero no necesariamente de alta gama. El equipo que se utilice para la automatización de servicios, debe contar con un procesador Intel Core i5 o equivalente en AMD, un mínimo de 8GB de memoria RAM (se recomiendan 16GB) y 100 GB de almacenamiento.

Es importante que disponga de una tarjeta de red Ethernet de alta velocidad para garantizar una conexión estable y fiable. Además, si se quiere salir a internet por una interfaz diferente, se deberá tener en cuenta y añadirla.

Lo ideal, es que cada administrador de sistemas tenga un portátil con estas características en los que se desplieguen los servicios en red, así los administradores tendrán acceso completo a la gestión de los nodos y la red.

## Automatización de servicios en red con Ansible

Por ejemplo:

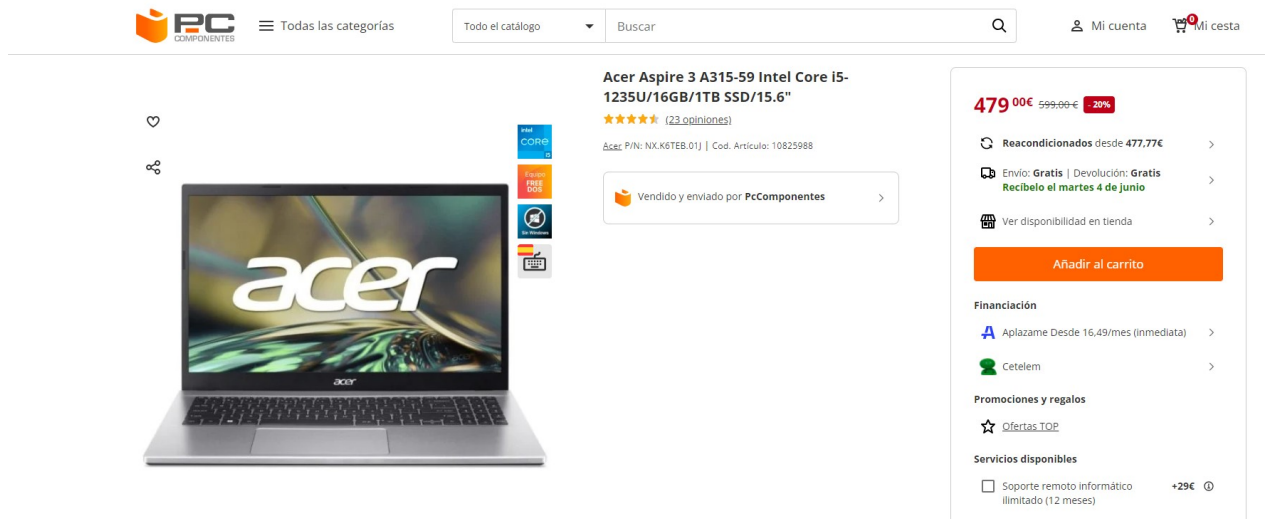


Figura 2: Ejemplo PC para administrar servidor.

Este ordenador cumple perfectamente con las características mencionadas ya que, además de tener un procesador potente para actuar de servidor, tiene 1TB SSD que permite desplegar los servicios en red y cualquier otro servicio que se quiera implementar (por ejemplo bases de datos).

## **5 ANÁLISIS**

### **5.1 Requisitos funcionales**

El principal requisito del proyecto es proporcionar servicios en red de forma controlada y gestionada automáticamente a cualquier cantidad de empleados de manera instantánea. Esto implica que todos los servicios necesarios para el funcionamiento diario de la empresa, como correo electrónico, acceso a bases de datos, y almacenamiento compartido, deben estar disponibles de manera rápida y confiable.

La automatización de estos servicios permite ahorros significativos en tiempo y recursos humanos. Los scripts y herramientas de automatización reducirán la necesidad de intervención manual, minimizando la posibilidad de errores humanos. Un administrador de sistemas solo necesitará asegurarse de que al menos un nodo esté funcionando correctamente para garantizar que los servicios en red estén disponibles para todos los empleados. Esto no solo simplifica la gestión del sistema, sino que también mejora la resiliencia del servicio, ya que el fallo de un nodo no afectará al resto del sistema.

### **5.2 Requisitos no funcionales**

Como se ha mencionado anteriormente, el objetivo del proyecto es proporcionar servicios de forma instantánea a todos los clientes. Esto no solo mejora significativamente el tiempo de respuesta respecto a un sistema no automatizado, sino que también contribuye a la eficiencia operativa de la organización.

Por otro lado, la definición de sentencias mediante scripts no solo facilita la automatización de tareas repetitivas, sino que también mejora la fiabilidad de los servicios. Los scripts permiten la ejecución de tareas complejas de manera rápida y precisa, independientemente del número de servidores (nodos) implicados. Esta característica es crucial para la escalabilidad del sistema, asegurando que el rendimiento no se degrade a medida que se añaden más nodos.

## **6 PREVENCIÓN DE RIESGOS LABORALES EN INFORMÁTICA**

La prevención de riesgos laborales en el ámbito de la informática es esencial para garantizar la seguridad y el bienestar de los trabajadores. A continuación, se detallan las principales áreas de riesgo y las medidas preventivas que deben implementarse.

### **1. Ergonomía y Postura**

#### **Riesgos:**

- Dolor de espalda, cuello y muñecas debido a posturas inadecuadas.
- Fatiga visual por uso prolongado de pantallas.

#### **Medidas Preventivas:**

- Ajustar la altura de la silla y la pantalla del ordenador para mantener una postura adecuada.
- Utilizar sillas ergonómicas que proporcionen soporte lumbar.
- Colocar el teclado y el ratón a una altura que permita mantener los brazos en un ángulo de 90 grados.
- Realizar pausas regulares para descansar los ojos y estirar el cuerpo.
- Implementar ejercicios de estiramiento y rotación para reducir la tensión muscular.

### **2. Iluminación y Ambiente**

#### **Riesgos:**

- Fatiga ocular y dolores de cabeza por iluminación inadecuada.
- Estrés térmico debido a una climatización insuficiente.

#### **Medidas Preventivas:**

- Asegurar una iluminación adecuada y uniforme en el espacio de trabajo, evitando reflejos en las pantallas.
- Utilizar lámparas de escritorio ajustables para proporcionar luz adicional donde sea necesario.
- Mantener una temperatura ambiente confortable y bien ventilada.
- Realizar ajustes regulares del brillo y contraste de las pantallas para minimizar la fatiga ocular.

### 3. Seguridad Eléctrica

#### Riesgos:

- Electrocución por equipos eléctricos defectuosos.
- Incendios causados por sobrecargas eléctricas.

#### Medidas Preventivas:

- Revisar y mantener regularmente los equipos eléctricos.
- No sobrecargar las tomas de corriente y utilizar regletas con protección contra sobrecargas.
- Asegurar que todos los equipos eléctricos estén correctamente conectados y que los cables no estén dañados.
- Desconectar equipos eléctricos no utilizados para evitar sobrecalentamientos.

### 4. Estrés y Salud Mental

#### Riesgos:

- Estrés laboral debido a plazos ajustados y altas demandas de trabajo.
- Burnout (síndrome de agotamiento profesional).

#### Medidas Preventivas:

- Fomentar una cultura de trabajo saludable que incluya la gestión del tiempo y la delegación de tareas.
- Proporcionar apoyo psicológico y acceso a recursos de salud mental para los empleados.
- Implementar horarios flexibles y políticas de teletrabajo para mejorar el equilibrio entre vida laboral y personal.
- Organizar actividades de team building y de relajación para reducir el estrés.

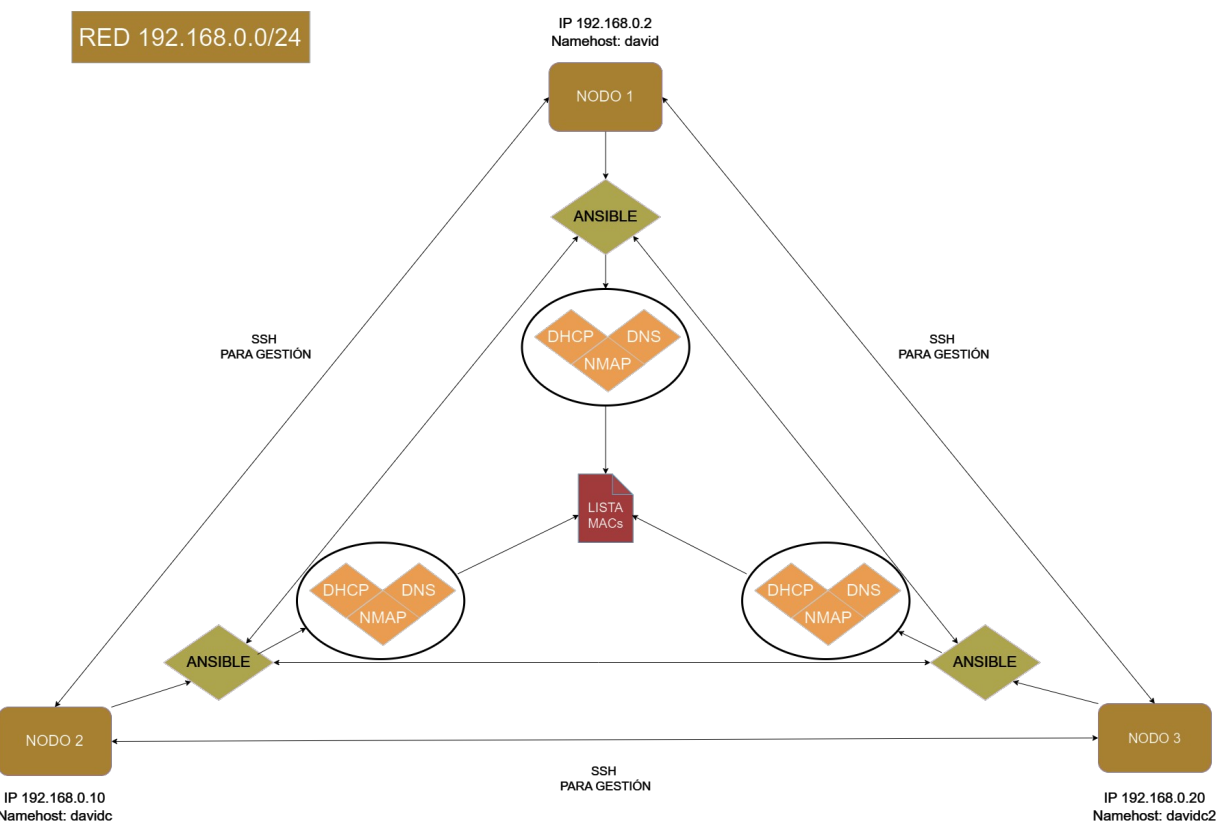


Figura 3: Riesgos laborales en informática

## 7 DISEÑO

Para la realización del proyecto, trabajaremos sobre tres nodos para comprobar que el sistema funciona, para ello configuraremos SSH entre máquinas importando las claves entre los nodos implicados. Así se evitará que cualquier dispositivo en la red pueda realizar solicitudes de conexión SSH ya que se denegarán automáticamente. Posteriormente, se configurará ansible y el conjunto de scripts necesarios para automatizar la red. Todos estos scripts dependerán de un fichero que contendrá la lista de MACs almacenadas.

### 7.1 Disposición gráfica del proyecto



*Figura 4: Esquema del proyecto*



## 8 IMPLEMENTACIÓN DEL PROYECTO

En este apartado se realizará una demostración sobre máquinas de Virtualbox con Ubuntu Desktop 22.04.1 de las diferentes configuraciones y scripts utilizados para la realización del proyecto. Para tener más detalles de la implementación está disponible en el anexo del documento un enlace a GitHub con los ficheros utilizados y un manual detallado de usuario. Los nodos de la demostración son los siguientes:

Hostname	IP interfaz enp0s3 (interfaz 1)
david	192.168.0.2
davidc	192.168.0.10
davidc2	192.168.0.20

Figura 5: Tabla de equipos en el proyecto

## 8.1 Configuración de máquinas virtuales para la demostración

Para la realización, deberemos configurar cada nodo con dos interfaces, ambas en modo puente.

La primera se encargará de la gestión de la red de nivel interno a automatizar y la segunda dará salida a internet. Replicaremos la configuración en todos los nodos.

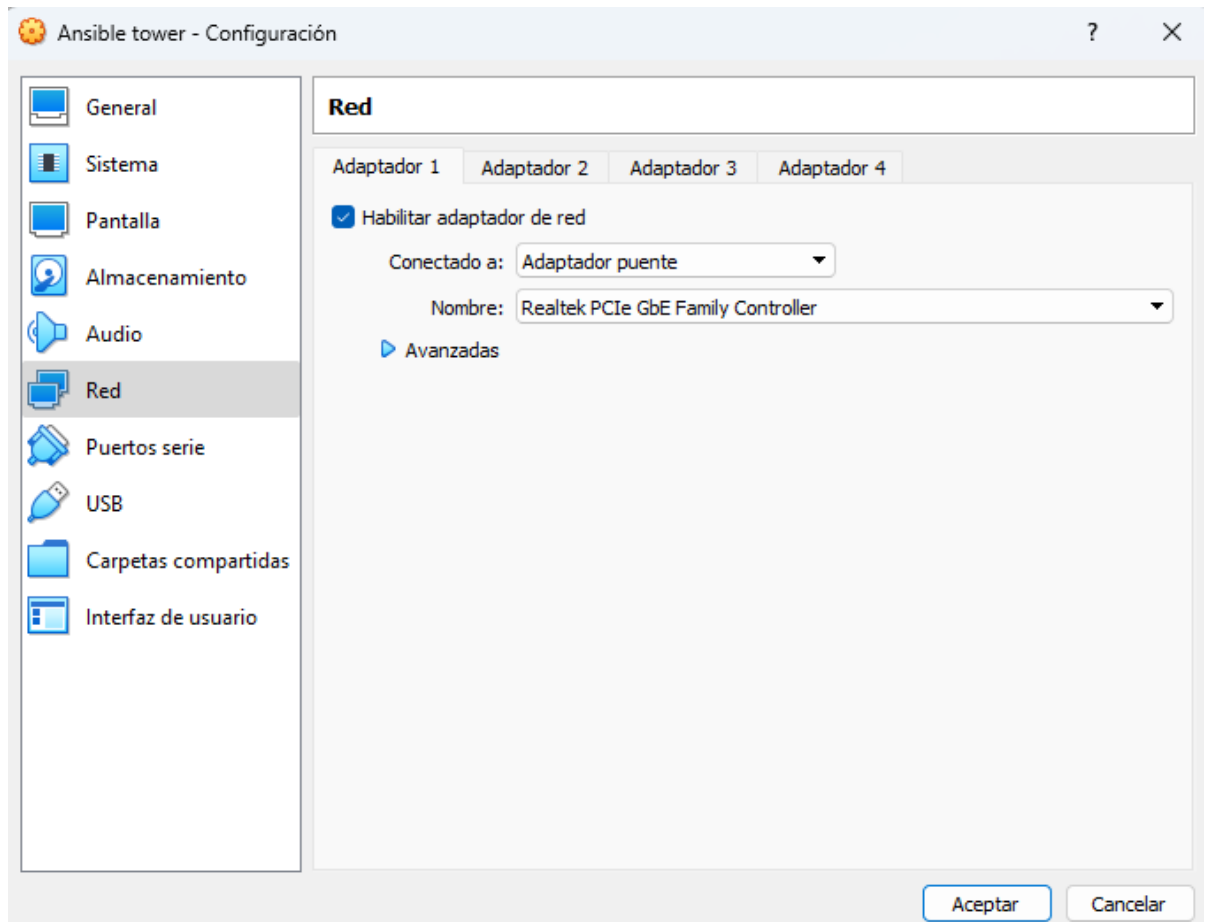


Figura 6: Primera interfaz de red

## Automatización de servicios en red con Ansible

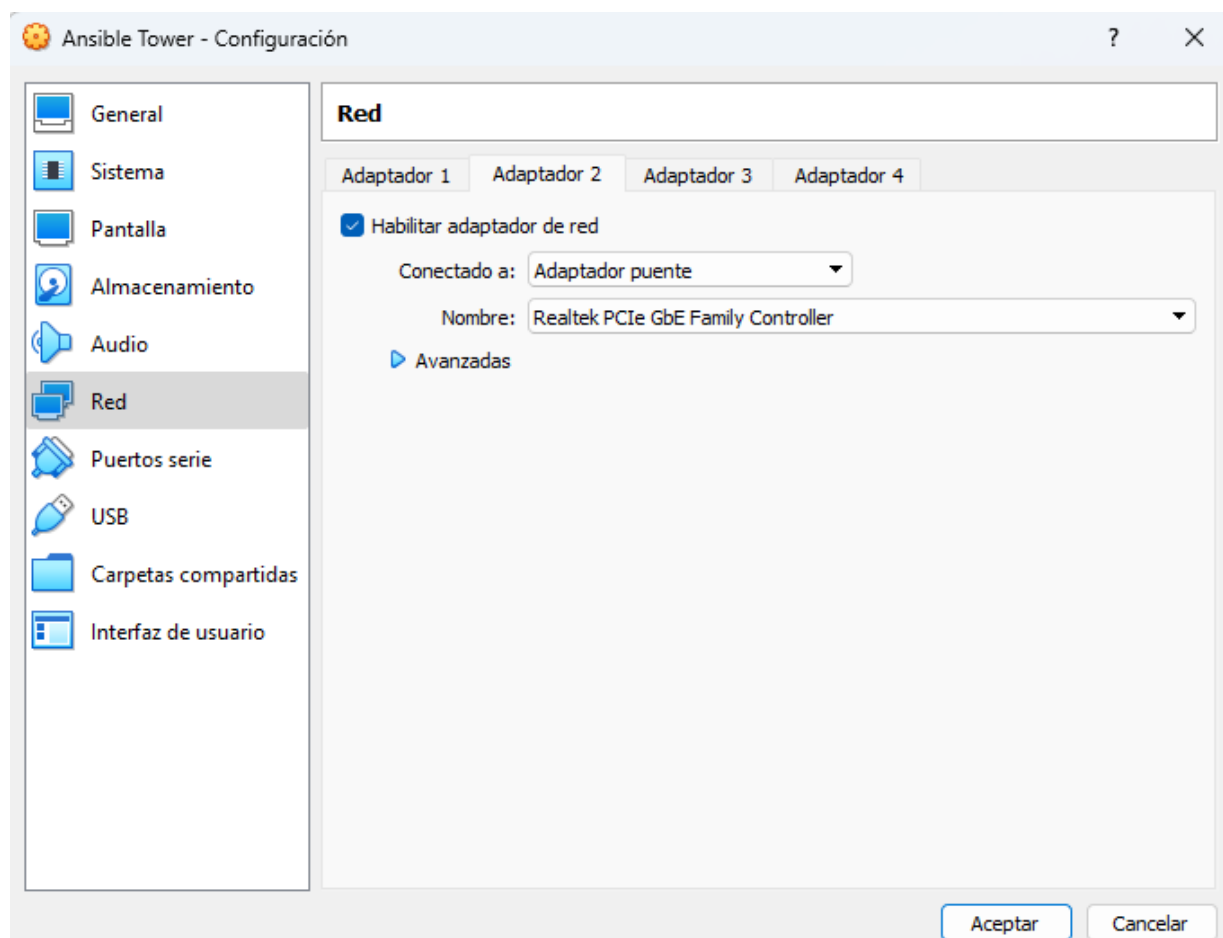


Figura 7: Segunda interfaz de red

### 8.1.1 Configuración de interfaces

Una vez creadas las máquinas con las interfaces necesarias, deberemos configurarlas.

En este caso, como estamos utilizando Ubuntu Desktop, deberemos acceder al fichero netplan

`sudo nano /etc/netplan/01-network-manager-all.yaml`



```

GNU nano 6.2 /etc/netplan/01-network-manager-all.yaml *
network:
  version: 2
  renderer: NetworkManager
  ethernets:
    enp0s3:
      addresses: [192.168.0.2/24]
      # gateway4: 192.168.0.1
      nameservers:
        addresses: [192.168.0.2, 192.168.0.10, 192.168.0.20]
    enp0s8:
      dhcp4: false
      addresses: [192.168.1.2/24] # Dirección IP estática en la red 192.168.1.0
      gateway4: 192.168.1.1 # Dirección del gateway en la red 192.168.1.0
      nameservers:
        addresses: [8.8.8.8, 8.8.4.4] # Servidores DNS de Google

```

Figura 8: Netplan del nodo david

Realizaremos lo mismo con el resto de nodos.



```

GNU nano 6.2 /etc/netplan/01-network-manager-all.yaml
network:
  version: 2
  renderer: NetworkManager
  ethernets:
    enp0s3:
      addresses: [192.168.0.10/24]
      # gateway4: 192.168.0.1
      nameservers:
        addresses: [192.168.0.2, 192.168.0.10, 192.168.0.20]
    enp0s8:
      dhcp4: false
      addresses: [192.168.1.10/24] # Dirección IP estática en la red 192.168.1.0
      gateway4: 192.168.1.1 # Dirección del gateway en la red 192.168.1.0
      nameservers:
        addresses: [8.8.8.8, 8.8.4.4] # Servidores DNS de Google

```

Figura 9: Netplan nodo davidc

## Automatización de servicios en red con Ansible



The screenshot shows a terminal window with the title bar "davidc2@davidc2: ~". The terminal is running GNU nano 6.2, editing the file /etc/netplan/01-network-manager-all.yaml. The content of the file is as follows:

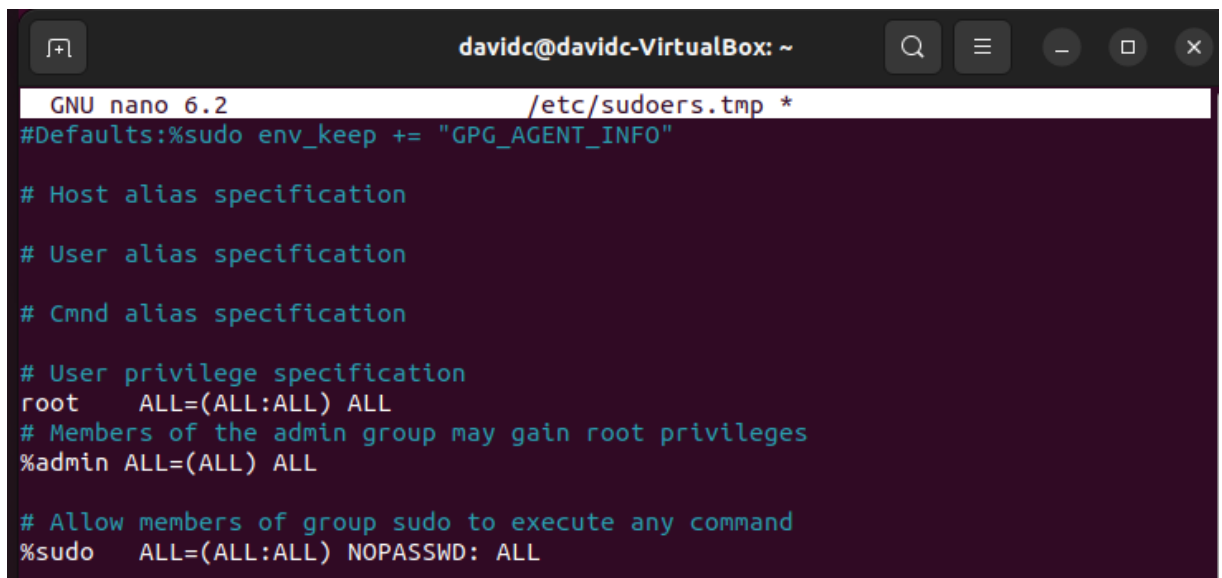
```
network:
  version: 2
  renderer: NetworkManager
  ethernets:
    enp0s3:
      addresses: [192.168.0.20/24]
      # gateway4: 192.168.0.1
      nameservers:
        addresses: [192.168.0.2, 192.168.0.10, 192.168.0.20]
    enp0s8:
      dhcp4: false
      addresses: [192.168.1.20/24] # Dirección IP estática en la red 192.168.1.0
      gateway4: 192.168.1.1 # Dirección del gateway en la red 192.168.1.0
      nameservers:
        addresses: [8.8.8.8, 8.8.4.4] # Servidores DNS de Google
```

Figura 10: Netplan nodo davidc2

### 8.1.2 Configuración fichero sudoers

Una vez configuradas las interfaces, se debe modificar el fichero sudoers de cada nodo para que cuando hagamos ejecuciones de comandos utilizando “sudo” no nos solicite la contraseña. De esta forma podremos permitir la gestión como administrador con Ansible.

*sudo visudo*



```
GNU nano 6.2 /etc/sudoers.tmp *
#Defaults:%sudo env_keep += "GPG_AGENT_INFO"

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL
# Members of the admin group may gain root privileges
%admin   ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) NOPASSWD: ALL
```

Figura 11: fichero sudoers

Se modificará la última línea de la imagen para que quede de la siguiente forma:

```
%sudo  ALL=(ALL:ALL) NOPASSWD: ALL
```

## 8.2 Configuración SSH entre nodos

Para realizar la configuración que permita la comunicación entre máquinas, se tendrá que configurar SSH ya que Ansible utiliza este protocolo para realizar la ejecución de los comandos en los diferentes nodos, ya sean en modo “ad hoc” o en playbooks.

**NOTA:** Estos pasos se deben realizar en todos los nodos implicados

Se instala openssh-server:

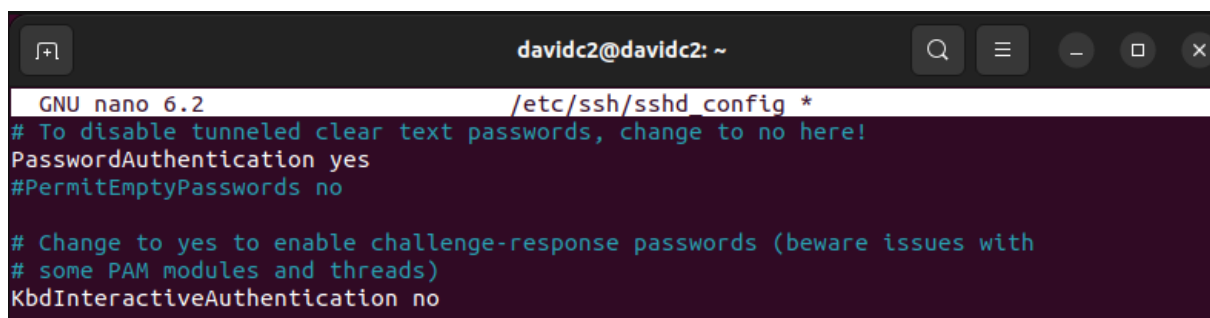
```
davidc@davidc-VirtualBox:~$ sudo apt install openssh-server
Leyendo lista de paquetes... Hecho
```

Figura 12: instalación de servidor ssh

A continuación, se va a modificar la configuración para permitir el acceso con clave, esto se realiza de forma temporal para importar las claves generadas. Más tarde se modificará para que únicamente puedan hacer solicitudes SSH los usuarios con claves importadas en los nodos.

Esto se realiza para añadir seguridad sobre los nodos servidores ya que todos los usuarios que van a estar en la red podrían tratar de acceder realizando solicitudes SSH para tratar de vulnerar el servidor.

Para permitir la autenticación de cualquier usuario (de forma temporal para añadir las claves generadas en cada nodo) se añade la siguiente línea: “PasswordAuthentication yes” dentro del fichero de configuración ubicado en “/etc/ssh/sshd\_config”



```
davidc2@davidc2: ~
GNU nano 6.2 /etc/ssh/sshd_config *
# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication yes
#PermitEmptyPasswords no

# Change to yes to enable challenge-response passwords (beware issues with
# some PAM modules and threads)
KbdInteractiveAuthentication no
```

Figura 13: Modificación temporal fichero configuración SSH

## Automatización de servicios en red con Ansible

Para aplicar los cambios se reinicia el servicio

```
davidc2@davidc2:~$ sudo systemctl restart ssh
```

Figura 14: Reinicio del servicio SSH

Una vez modificado, se genera la clave ssh del nodo para posteriormente importarla en el resto de nodos implicados.

```
davidc@davidc-VirtualBox:~$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/davidc/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/davidc/.ssh/id_ed25519
Your public key has been saved in /home/davidc/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:z4jsGefAav0aRbiGvSFRFLvjhKS8E1NKjp9tnAK6TXs davidc@davidc-VirtualBox
The key's randomart image is:
+--[ED25519 256]--+
|      .+.      |
|      . o      |
|    . + o .    |
|  = = = +      |
|o B + O S      |
|.o B O * +      |
|. B =.X o o     |
| + =Eo.B        |
|. oo. +oo       |
+-----[SHA256]-----+
```

Figura 15: Generar clave SSH

A continuación, se añade la clave generada al agente SSH. Esto se realiza para no tener que ingresar contraseña cada vez que se utilice el protocolo.

```
davidc@davidc-VirtualBox:~$ eval "$(ssh-agent -s)"
Agent pid 2862
davidc@davidc-VirtualBox:~$ ssh-add /home/davidc/.ssh/id_ed25519
Identity added: /home/davidc/.ssh/id_ed25519 (davidc@davidc-VirtualBox)
```

Figura 16: Añadir identidad SSH



## Automatización de servicios en red con Ansible

tras configurarlo, se accede a cada nodo que queramos importar la clave por SSH. Al realizar esto,

```
davidc@davidc-VirtualBox:~$ ssh -l davidc2 192.168.0.20
davidc2@192.168.0.20's password:
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 6.5.0-28-generic x86_64)
```

Figura 17: SSH a nodo davidc2

automáticamente nuestro nodo se añadirá en el nodo destino a una lista de hosts conocidos.

al realizar la conexión, solo quedará salir de la conexión SSH e importar la clave generada:

```
davidc@davidc-VirtualBox:~$ ssh-copy-id -i /home/davidc/.ssh/id_ed25519.pub davidc2@192.168.0.20
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/davidc/.ssh/id_ed25519.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
davidc2@192.168.0.20's password:

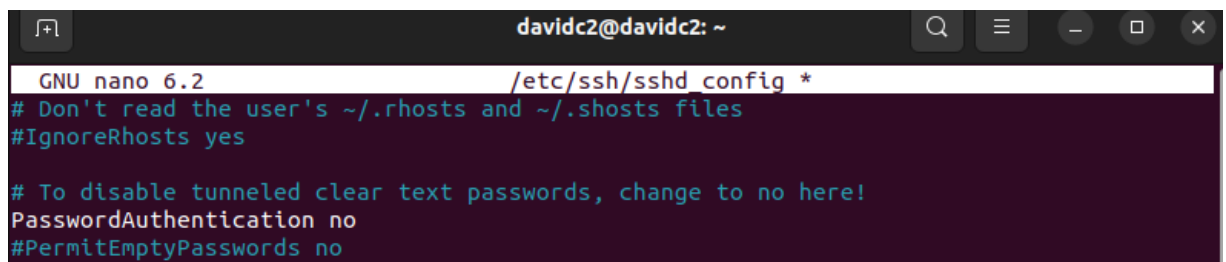
Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'davidc2@192.168.0.20'"
and check to make sure that only the key(s) you wanted were added.
```

Figura 18: importar clave generada en nodo davidc a davidc2

tras esto, se deshabilita la conexión por autenticación de contraseña en la configuración SSH

La línea debe quedar así: "PasswordAuthentication no"



```
davidc2@davidc2: ~
GNU nano 6.2 /etc/ssh/sshd_config *
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication no
#PermitEmptyPasswords no
```

Figura 19: Modificar fichero configuración SSH para denegar conexiones por contraseña

Para aplicar los cambios realizados se debe reiniciar el servicio

```
davidc2@davidc2:~$ sudo systemctl restart ssh
```

Figura 20: Reiniciar servicio ssh

**NOTA:** La configuración se debe realizar en ambos sentidos, es decir A debe importar la clave en B y viceversa. En caso de que no se realice, aparecerá un mensaje similar al siguiente

```
davidc2@davidc2:~$ ssh davidc@192.168.0.10  
davidc@192.168.0.10: Permission denied (publickey).
```

Figura 21: Ejemplo de permisos denegados

### **8.3 Configuración de ansible y su inventario en un nodo cualquiera**

Para realizar la automatización de la red sobre varios nodos, hay que conocer varios conceptos sobre Ansible:

**INVENTARIO:** Lista de equipos que vamos a gestionar con Ansible. Se pueden realizar en formato INI (formato del archivo predeterminado) o formato YAML. En este caso lo trabajaremos en formato INI.

**PLAYBOOK:** Conjunto de instrucciones que ejecuta Ansible sobre los Equipos inventariados. Se realizan en formato YAML.

Se empezará configurando Ansible en un nodo, para posteriormente, gracias a los playbooks que permite desplegar, configurar el resto de nodos todos a la vez.

**NOTA:** En este caso se trabajarán todos los ficheros creados en el directorio raíz de la máquina. Lo ideal es gestionarlo dentro de una carpeta con permisos específicos.

## Automatización de servicios en red con Ansible

Una vez vistos los conceptos básicos, se instala Ansible sobre un nodo cualquiera (en este caso sobre el nodo david).

```
david@david:~$ sudo apt-get install ansible
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Los paquetes indicados a continuación se instalaron de forma automática y ya no son necesarios.
  libflashrom1 libftdi1-2 liblvm13
Utilice «sudo apt autoremove» para eliminarlos.
Se instalarán los siguientes paquetes adicionales:
  ieee-data python-babel-localedata python3-argcomplete python3-babel python3-distutils
  python3-dnspython python3-jinja2 python3-jmespath python3-kerberos python3-libcloud python3-netaddr
  python3-ntlm-auth python3-packaging python3-pycryptodome python3-requests-kerberos
  python3-requests-ntlm python3-requests-toolbelt python3-selinux python3-simplejson python3-winrm
  python3-xlwtodict
Paquetes sugeridos:
  cowsay sshpass python3-sniffio python3-trio python-jinja2-doc ipython3 python-netaddr-docs
Se instalarán los siguientes paquetes NUEVOS:
  ansible ieee-data python-babel-localedata python3-argcomplete python3-babel python3-distutils
  python3-dnspython python3-jinja2 python3-jmespath python3-kerberos python3-libcloud python3-netaddr
  python3-ntlm-auth python3-packaging python3-pycryptodome python3-requests-kerberos
  python3-requests-ntlm python3-requests-toolbelt python3-selinux python3-simplejson python3-winrm
  python3-xlwtodict
0 actualizados, 22 nuevos se instalarán, 0 para eliminar y 8 no actualizados.
Se necesita descargar 28,2 MB de archivos.
Se utilizarán 272 MB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] s
Des:1 http://es.archive.ubuntu.com/ubuntu jammy/main amd64 python-babel-localedata all 2.8.0+dfsg.1-7 [4.9
82 kB]
```

Figura 22: Instalación de Ansible

Una vez termine la instalación, se puede comprobar con el siguiente comando:

```
david@david:~$ ansible --version
ansible 2.10.8
  config file = None
  configured module search path = ['/home/david/.ansible/plugins/modules', '/usr/share/ansible/plugins/mod
ules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  executable location = /usr/bin/ansible
  python version = 3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0]
```

Figura 23: Versión de Ansible instalada

A continuación, se creará un fichero llamado “hosts” con el inventario de los nodos servidores:

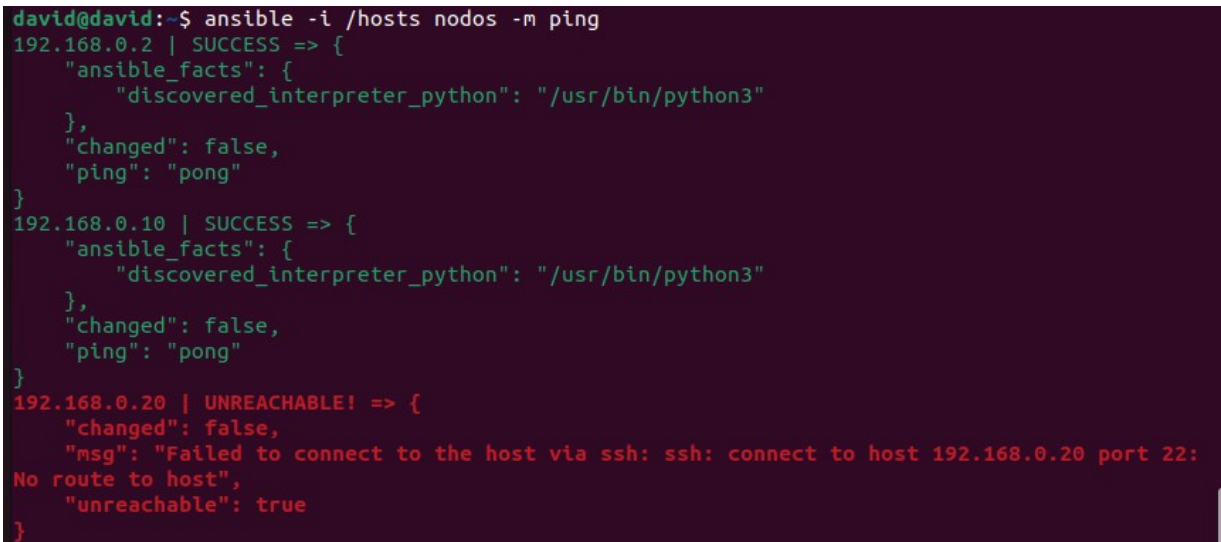
```
GNU nano 6.2 /hosts *
[+]
192.168.0.2 ansible_connection=local
192.168.0.10 ansible_user=davidc
192.168.0.20 ansible_user=davidc2
```

Figura 24: Definir inventario de nodo david

## Automatización de servicios en red con Ansible

Con esto definido, se pueden ejecutar comandos “Ad hoc” para comprobar si está correctamente configurado:

Por ejemplo, lanzando un ping sobre todos los nodos del inventario hosts para verificar si están activos o no.



```
david@david:~$ ansible -i /hosts nodos -m ping
192.168.0.2 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
192.168.0.10 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
192.168.0.20 | UNREACHABLE! => {
  "changed": false,
  "msg": "Failed to connect to the host via ssh: ssh: connect to host 192.168.0.20 port 22: No route to host",
  "unreachable": true
}
```

Figura 25: Ping a nodos del inventario

Como se observa en la imagen, el nodo con IP 192.168.0.20 (nodo davidc2) no está activo mientras que el resto si.

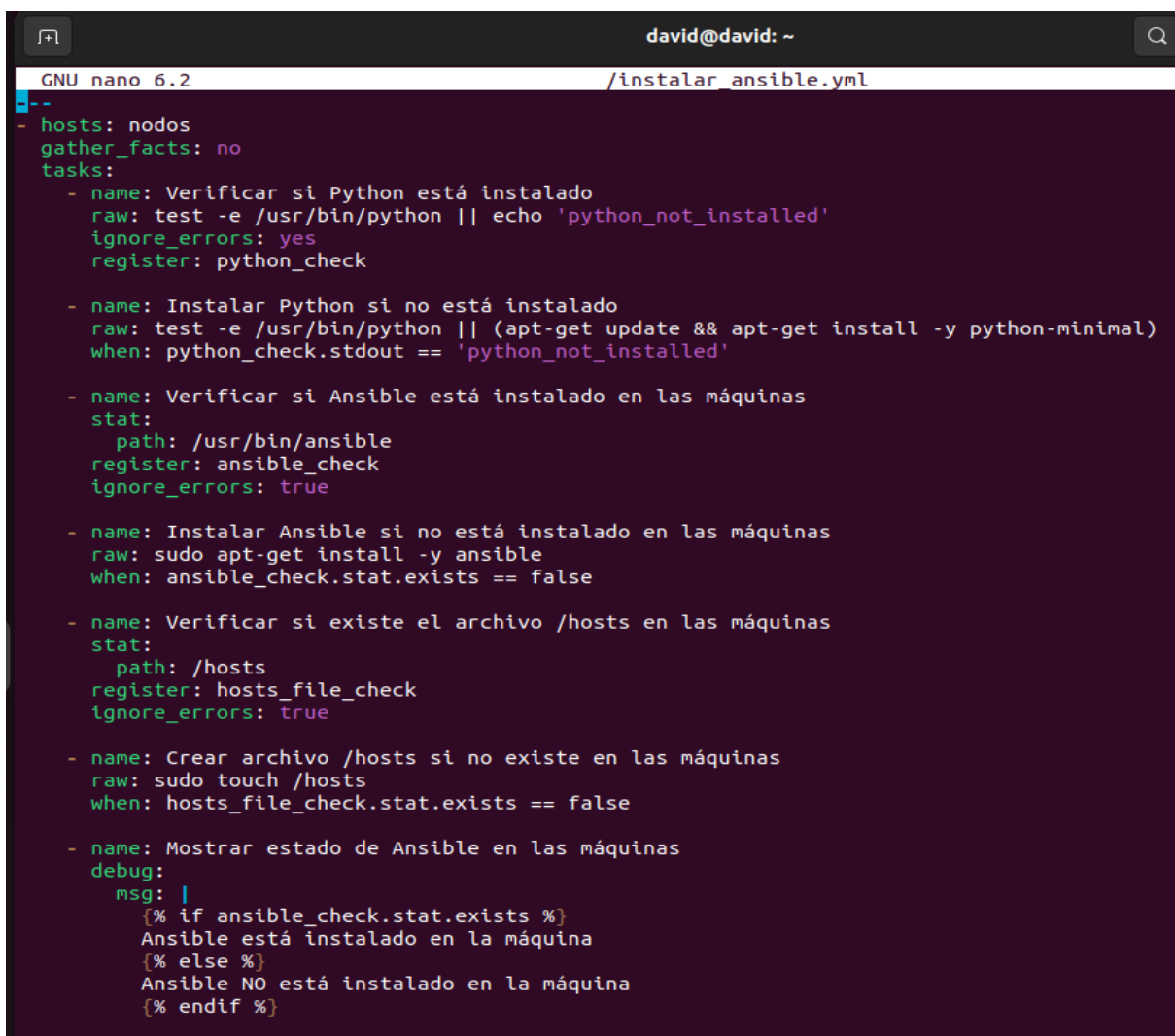
La prueba que realiza este comando, consiste en conectar por SSH al nodo destino y obtener una respuesta, en este caso “pong” al “ping” solicitado. Como se puede leer en el mensaje de error, no ha podido conectarse al nodo destino, por lo que asumimos que está apagado.

## 8.4 Ejecución de playbook para instalar Ansible en el resto de nodos

Una vez se ha configurado Ansible sobre un nodo, se realizarán varios playbooks para automatizar la instalación sobre el resto de nodos. Para ello, utilizaremos la función “Raw”.

Raw → permite ejecutar comandos “crudos” en los nodos, con lo que se consigue utilizar sentencias como “sudo” para aplicar configuraciones que sin dicho parámetro no funcionan correctamente.

Con esto, se realizará un primer playbook que verifique si está Python y Ansible instalado, Si no lo están, se instalarán. Además, se creará un fichero hosts vacío para configurar posteriormente el inventario de cada nodo.



```

GNU nano 6.2 /instalar_ansible.yml
---
- hosts: nodos
  gather_facts: no
  tasks:
    - name: Verificar si Python está instalado
      raw: test -e /usr/bin/python || echo 'python_not_installed'
      ignore_errors: yes
      register: python_check

    - name: Instalar Python si no está instalado
      raw: test -e /usr/bin/python || (apt-get update && apt-get install -y python-minimal)
      when: python_check.stdout == 'python_not_installed'

    - name: Verificar si Ansible está instalado en las máquinas
      stat:
        path: /usr/bin/ansible
        register: ansible_check
        ignore_errors: true

    - name: Instalar Ansible si no está instalado en las máquinas
      raw: sudo apt-get install -y ansible
      when: ansible_check.stat.exists == false

    - name: Verificar si existe el archivo /hosts en las máquinas
      stat:
        path: /hosts
        register: hosts_file_check
        ignore_errors: true

    - name: Crear archivo /hosts si no existe en las máquinas
      raw: sudo touch /hosts
      when: hosts_file_check.stat.exists == false

    - name: Mostrar estado de Ansible en las máquinas
      debug:
        msg: |
          {% if ansible_check.stat.exists %}
          Ansible está instalado en la máquina
          {% else %}
          Ansible NO está instalado en la máquina
          {% endif %}

```

Figura 26: Playbook instalar Ansible

## Automatización de servicios en red con Ansible

Al ejecutar el playbook, ejecutará una salida similar a la siguiente:

```
david@david: /
TASK [Instalar Ansible si no está instalado en las máquinas] *****
skipping: [192.168.0.2]
^C [ERROR]: User interrupted execution
david@david:/$ ansible-playbook /instalar_ansible.yml -i /hosts

PLAY [nodos] *****

TASK [Verificar si Python está instalado] *****
changed: [192.168.0.2]
changed: [192.168.0.10]
changed: [192.168.0.20]

TASK [Instalar Python si no está instalado] *****
skipping: [192.168.0.2]
skipping: [192.168.0.10]
skipping: [192.168.0.20]

TASK [Verificar si Ansible está instalado en las máquinas] *****
ok: [192.168.0.2]
ok: [192.168.0.20]
ok: [192.168.0.10]

TASK [Instalar Ansible si no está instalado en las máquinas] *****
skipping: [192.168.0.2]
changed: [192.168.0.20]
changed: [192.168.0.10]
```

Figura 27: Ejecución de playbook instalar Ansible

una vez finalice, se puede comprobar que todos los nodos tienen Ansible instalado:

```
davidc@davidc-VirtualBox:~$ ansible --version
ansible 2.10.8
  config file = None
  configured module search path = ['/home/davidc/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  executable location = /usr/bin/ansible
  python version = 3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0]
```

Figura 28: Comprobación en el resto de nodos 1

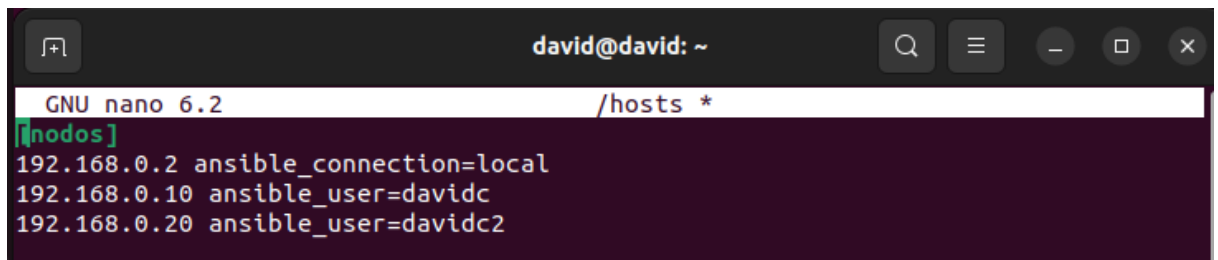
```
davidc@davidc-VirtualBox:~$ ls / | grep hosts
hosts
```

Figura 29: Comprobación en el resto de nodos 2

## 8.5 Configuración del resto de inventarios

Los inventarios, son un requisito fundamental para realizar la automatización de la red sobre varios nodos. Por ello, hay que configurar dicho inventario definiendo como “local” al nodo que va a ejecutar el playbook. Los inventarios de los nodos quedarán de la siguiente forma:

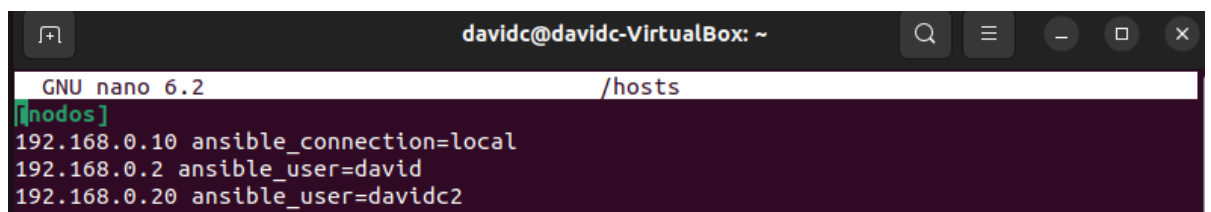
nodo david



```
GNU nano 6.2 /hosts *
[nodos]
192.168.0.2 ansible_connection=local
192.168.0.10 ansible_user=davidc
192.168.0.20 ansible_user=davidc2
```

Figura 30: Inventario nodo david

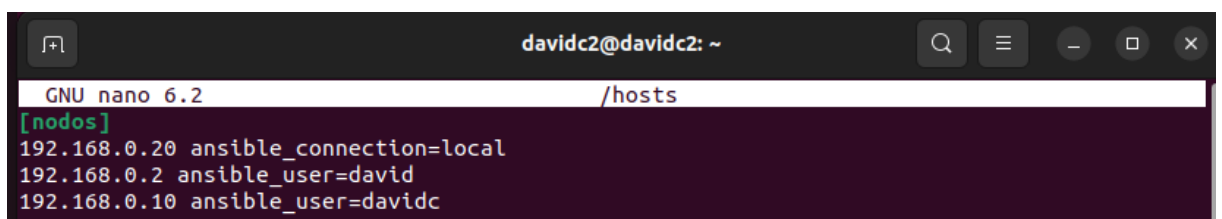
nodo davidc



```
GNU nano 6.2 /hosts
[nodos]
192.168.0.10 ansible_connection=local
192.168.0.2 ansible_user=david
192.168.0.20 ansible_user=davidc2
```

Figura 31: Inventario nodo davidc

nodo davidc2



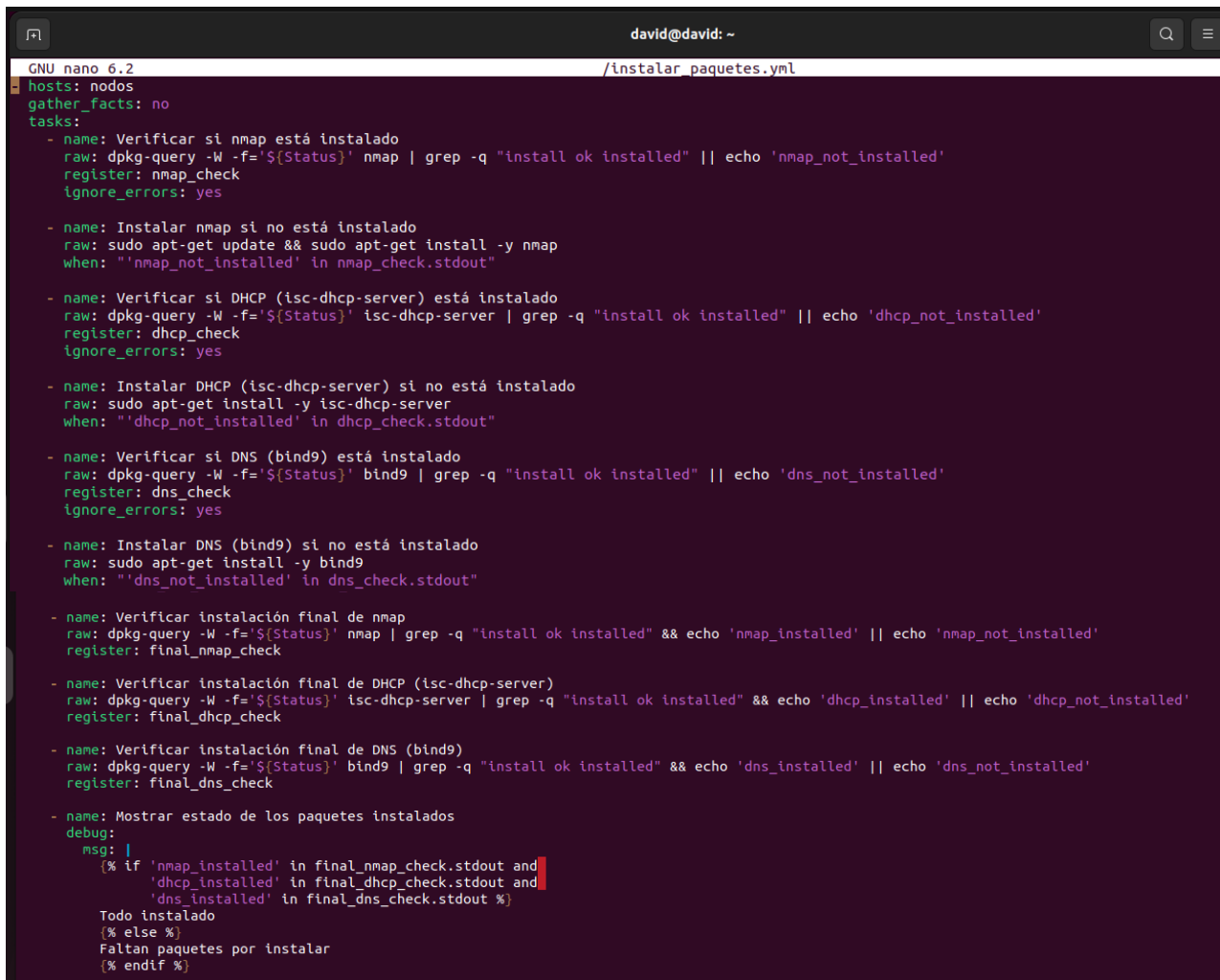
```
GNU nano 6.2 /hosts
[nodos]
192.168.0.20 ansible_connection=local
192.168.0.2 ansible_user=david
192.168.0.10 ansible_user=davidc
```

Figura 32: Inventario nodo davidc2



## 8.6 Playbook de instalación de paquetes sobre todos los nodos

Para realizar este proyecto, se requiere utilizar Nmap, DHCP y DNS. Por esta razón, se creará un playbook que instale todos los paquetes mencionados en cada nodo.



```

GNU nano 6.2 /instalar_paquetes.yml
hosts: nodos
gather_facts: no
tasks:
  - name: Verificar si nmap está instalado
    raw: dpkg-query -W -f='${Status}' nmap | grep -q "install ok installed" || echo 'nmap_not_installed'
    register: nmap_check
    ignore_errors: yes

  - name: Instalar nmap si no está instalado
    raw: sudo apt-get update && sudo apt-get install -y nmap
    when: "'nmap_not_installed' in nmap_check.stdout"

  - name: Verificar si DHCP (isc-dhcp-server) está instalado
    raw: dpkg-query -W -f='${Status}' isc-dhcp-server | grep -q "install ok installed" || echo 'dhcp_not_installed'
    register: dhcp_check
    ignore_errors: yes

  - name: Instalar DHCP (isc-dhcp-server) si no está instalado
    raw: sudo apt-get install -y isc-dhcp-server
    when: "'dhcp_not_installed' in dhcp_check.stdout"

  - name: Verificar si DNS (bind9) está instalado
    raw: dpkg-query -W -f='${Status}' bind9 | grep -q "install ok installed" || echo 'dns_not_installed'
    register: dns_check
    ignore_errors: yes

  - name: Instalar DNS (bind9) si no está instalado
    raw: sudo apt-get install -y bind9
    when: "'dns_not_installed' in dns_check.stdout"

  - name: Verificar instalación final de nmap
    raw: dpkg-query -W -f='${Status}' nmap | grep -q "install ok installed" && echo 'nmap_installed' || echo 'nmap_not_installed'
    register: final_nmap_check

  - name: Verificar instalación final de DHCP (isc-dhcp-server)
    raw: dpkg-query -W -f='${Status}' isc-dhcp-server | grep -q "install ok installed" && echo 'dhcp_installed' || echo 'dhcp_not_installed'
    register: final_dhcp_check

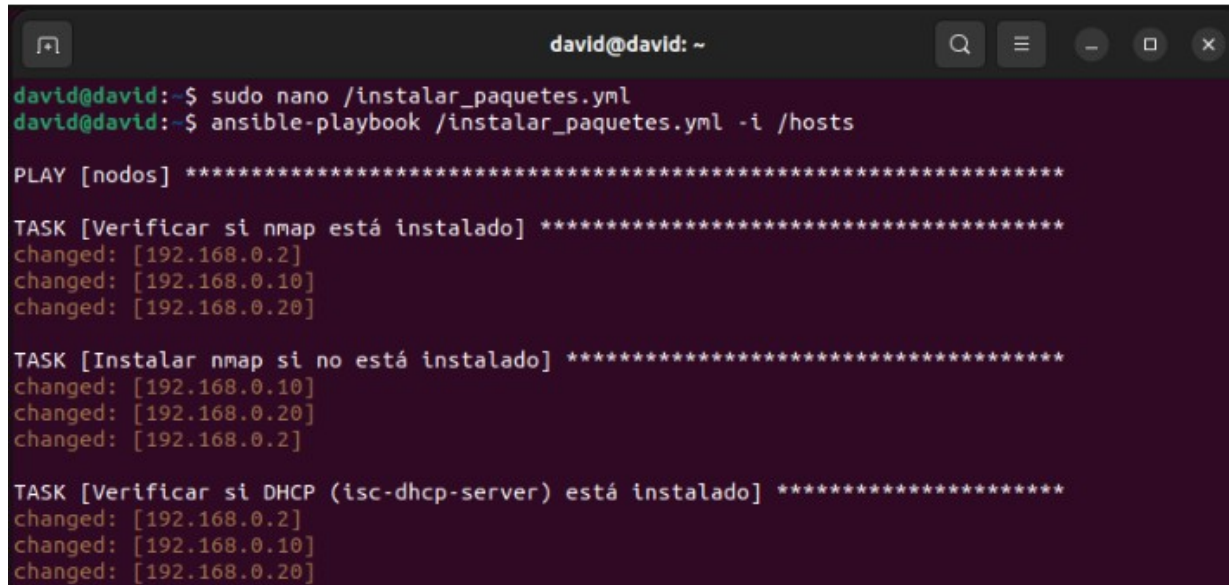
  - name: Verificar instalación final de DNS (bind9)
    raw: dpkg-query -W -f='${Status}' bind9 | grep -q "install ok installed" && echo 'dns_installed' || echo 'dns_not_installed'
    register: final_dns_check

  - name: Mostrar estado de los paquetes instalados
    debug:
      msg: |
        {% if 'nmap_installed' in final_nmap_check.stdout and
          'dhcp_installed' in final_dhcp_check.stdout and
          'dns_installed' in final_dns_check.stdout %}
          Todo instalado
        {% else %}
          Faltan paquetes por instalar
        {% endif %}
  
```

Figura 33: Playbook instalar paquetes

## Automatización de servicios en red con Ansible

Al ejecutarlo, este playbook proporcionará una salida similar a la siguiente imagen:

A terminal window titled 'david@david: ~' showing the execution of an Ansible playbook. The user runs 'sudo nano /instalar\_paquetes.yml' and then 'ansible-playbook /instalar\_paquetes.yml -i /hosts'. The output shows three tasks: 'Verificar si nmap está instalado', 'Instalar nmap si no está instalado', and 'Verificar si DHCP (isc-dhcp-server) está instalado'. Each task lists the IP addresses of the nodes where it was executed, with some showing 'changed' status.

```
david@david:~$ sudo nano /instalar_paquetes.yml
david@david:~$ ansible-playbook /instalar_paquetes.yml -i /hosts

PLAY [nodos] *****

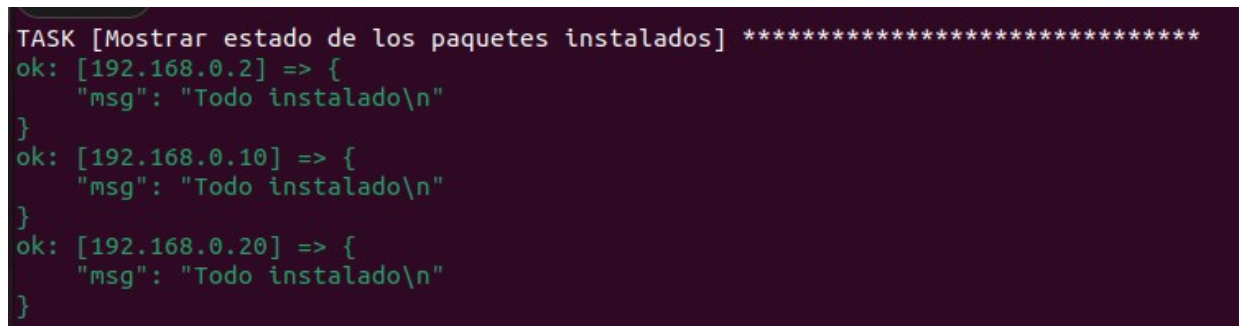
TASK [Verificar si nmap está instalado] *****
changed: [192.168.0.2]
changed: [192.168.0.10]
changed: [192.168.0.20]

TASK [Instalar nmap si no está instalado] *****
changed: [192.168.0.10]
changed: [192.168.0.20]
changed: [192.168.0.2]

TASK [Verificar si DHCP (isc-dhcp-server) está instalado] *****
changed: [192.168.0.2]
changed: [192.168.0.10]
changed: [192.168.0.20]
```

Figura 34: Ejecución de playbook instalar paquetes 1

Además, al finalizar indicará si se ha podido instalar todos los paquetes o no sobre cada nodo

A terminal window showing the final output of the Ansible playbook. The task 'Mostrar estado de los paquetes instalados' displays a JSON-like output for each node, indicating that all packages were successfully installed ('msg': 'Todo instalado\n').

```
TASK [Mostrar estado de los paquetes instalados] *****
ok: [192.168.0.2] => {
  "msg": "Todo instalado\n"
}
ok: [192.168.0.10] => {
  "msg": "Todo instalado\n"
}
ok: [192.168.0.20] => {
  "msg": "Todo instalado\n"
}
```

Figura 35: Ejecución de playbook instalar paquetes 2

Con esto se habrá completado la instalación de los servicios necesarios para automatizar una red con Ansible.

## **8.7 Creación de scripts encargados de la automatización**

El objetivo de la automatización de servicios en red con Ansible, es conseguir que cada equipo de la red tenga una reserva de IP dentro del DHCP así como una resolución por nombres gracias al DNS.

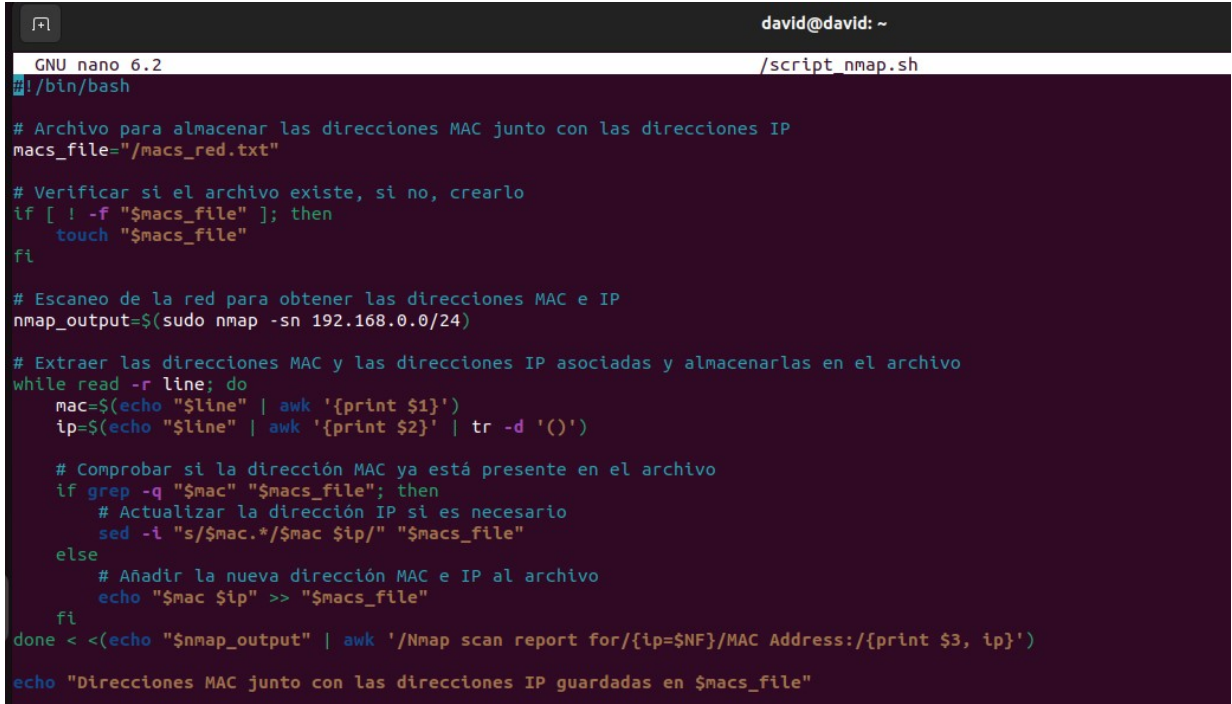
Para ello, se escaneará constantemente la red con Nmap y se guardarán los resultados en un fichero llamado "macs\_red.txt". A raíz de este fichero, se crearán scripts de automatización para cada servicio, con el fin de cumplir el objetivo del proyecto.

Los scripts que se van a realizar son:

- Script de nmap (script\_nmap.sh)
- Script DHCP (script\_dhcp.sh)
- Script base DNS (base\_dns.sh)
- Script DNS (script\_dns.sh)

### 8.7.1 Script de Nmap

Este script, como se ha mencionado en el apartado anterior, realizará un escaneo de la red y guardará la salida en un fichero llamado “macs\_red.txt”.



```

GNU nano 6.2 /script_nmap.sh
#!/bin/bash

# Archivo para almacenar las direcciones MAC junto con las direcciones IP
macs_file="macs_red.txt"

# Verificar si el archivo existe, si no, crearlo
if [ ! -f "$macs_file" ]; then
    touch "$macs_file"
fi

# Escaneo de la red para obtener las direcciones MAC e IP
nmap_output=$(sudo nmap -sn 192.168.0.0/24)

# Extraer las direcciones MAC y las direcciones IP asociadas y almacenarlas en el archivo
while read -r line; do
    mac=$(echo "$line" | awk '{print $1}')
    ip=$(echo "$line" | awk '{print $2}' | tr -d '()')

    # Comprobar si la dirección MAC ya está presente en el archivo
    if grep -q "$mac" "$macs_file"; then
        # Actualizar la dirección IP si es necesario
        sed -i "s/$mac.*/$mac $ip/" "$macs_file"
    else
        # Añadir la nueva dirección MAC e IP al archivo
        echo "$mac $ip" >> "$macs_file"
    fi
done < <(echo "$nmap_output" | awk '/Nmap scan report for/{ip=$NF}/MAC Address:/{print $3, ip}')

echo "Direcciones MAC junto con las direcciones IP guardadas en $macs_file"

```

Figura 36: Script Nmap

la salida del fichero “macs\_red.txt” tiene el siguiente formato:



```

GNU nano 6.2 /macs_red.txt
08:00:27:64:F9:A9 192.168.0.10
08:00:27:B3:D5:35 192.168.0.20

```

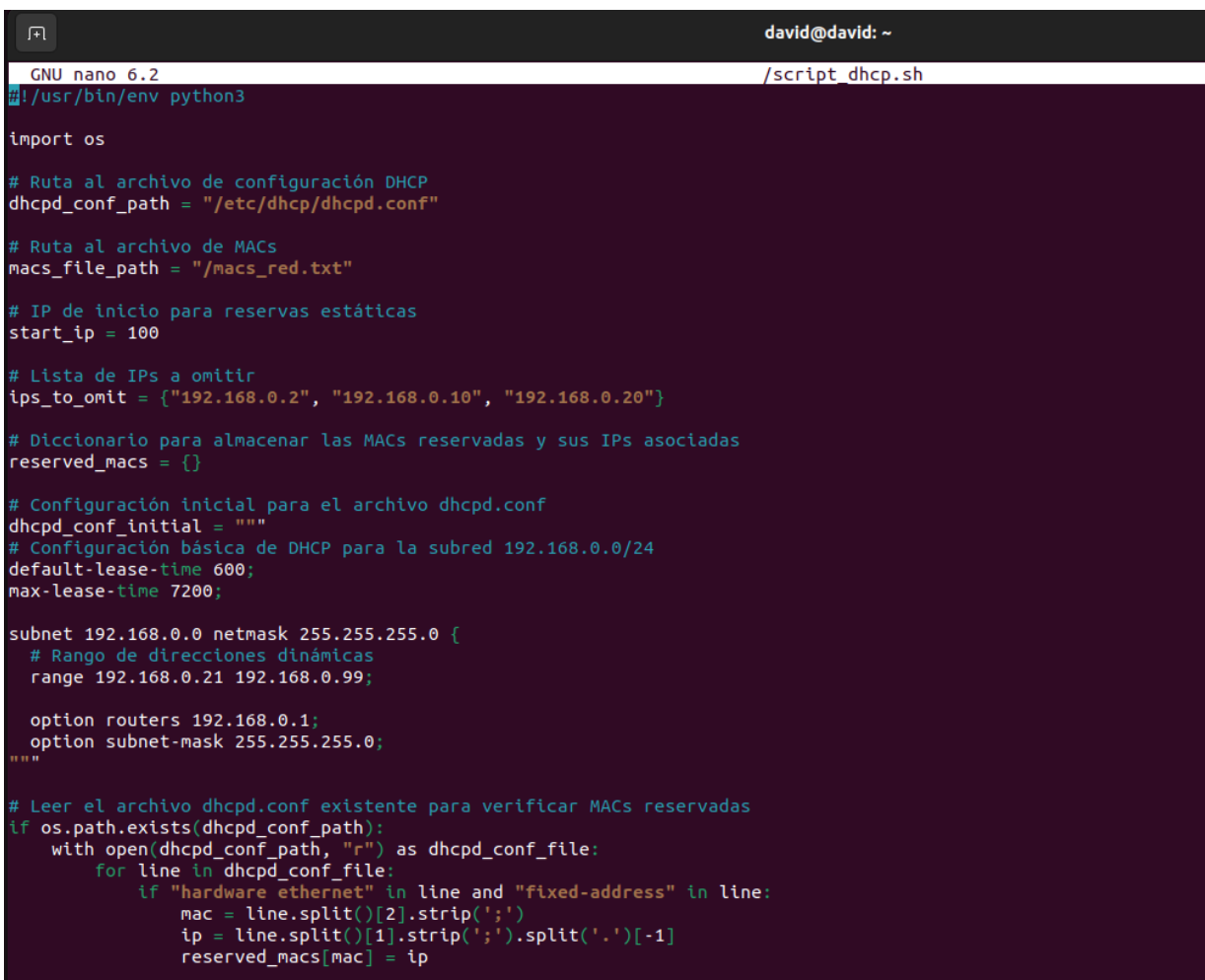
Figura 37: Salida fichero macs

### 8.7.2 Script DHCP

En este script, primero se definirán las rutas de los ficheros con los que trabajar. A continuación, se establecerá un contador de IP que empezará en 100.

Este contador, se ha declarado para indicar a partir de qué IP se van a reservar los equipos conectados. Las anteriores a 100 se dejarán libres para que cualquier usuario que haga una primera conexión a la red, tenga asignación antes de realizarse su reserva.

Además, se ha creado una lista con las IPs de los nodos para que se omitan en las reservas y otra con las MACs reservadas para evitar que una dirección MAC tenga varias IPs reservadas, es decir, para evitar duplicidad.



```

david@david: ~
GNU nano 6.2 /script_dhcp.sh
#!/usr/bin/env python3

import os

# Ruta al archivo de configuración DHCP
dhcpd_conf_path = "/etc/dhcp/dhcpd.conf"

# Ruta al archivo de MACs
macs_file_path = "/macs_red.txt"

# IP de inicio para reservas estáticas
start_ip = 100

# Lista de IPs a omitir
ips_to_omit = {"192.168.0.2", "192.168.0.10", "192.168.0.20"}

# Diccionario para almacenar las MACs reservadas y sus IPs asociadas
reserved_macs = {}

# Configuración inicial para el archivo dhcpd.conf
dhcpd_conf_initial = """
# Configuración básica de DHCP para la subred 192.168.0.0/24
default-lease-time 600;
max-lease-time 7200;

subnet 192.168.0.0 netmask 255.255.255.0 {
    # Rango de direcciones dinámicas
    range 192.168.0.21 192.168.0.99;

    option routers 192.168.0.1;
    option subnet-mask 255.255.255.0;
}
"""

# Leer el archivo dhcpd.conf existente para verificar MACs reservadas
if os.path.exists(dhcpd_conf_path):
    with open(dhcpd_conf_path, "r") as dhcpd_conf_file:
        for line in dhcpd_conf_file:
            if "hardware ethernet" in line and "fixed-address" in line:
                mac = line.split()[2].strip(';')
                ip = line.split()[1].strip(';').split('.')[1]
                reserved_macs[mac] = ip

```

Figura 38: Script DHCP

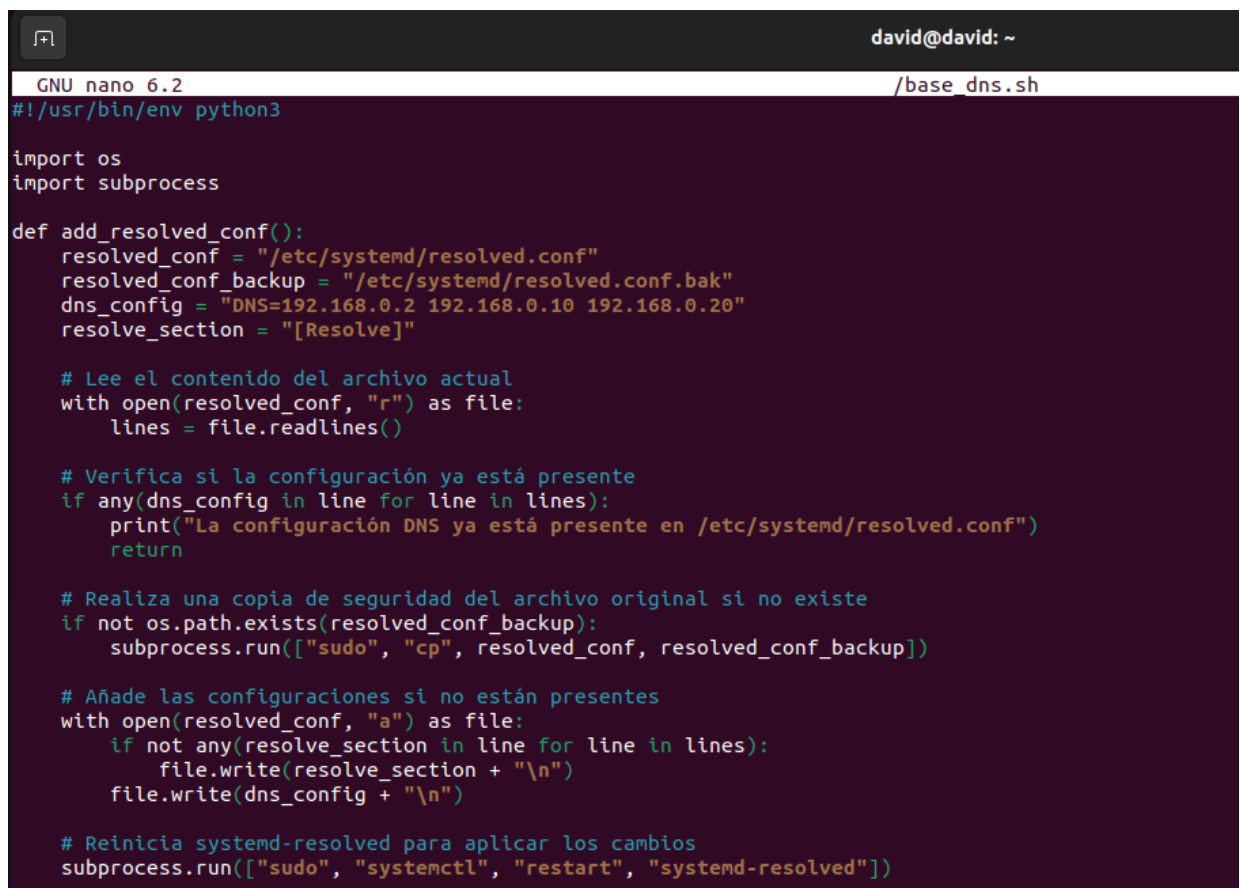
### 8.7.3 Scripts DNS

Para la gestión del servicio DNS, se realizarán dos scripts:

El primero de ellos, creará los ficheros estáticos y básicos configurados para el equipo donde se ejecuta. Recibirá el nombre de “base\_dns.sh”

El segundo, se encargará de la parte dinámica de la automatización, es decir, leerá el fichero “macs\_red.txt” y asignará una resolución tanto directa como inversa para cada dispositivo detectado. Además de la resolución, creará una entrada de tipo CNAME para identificar de una forma fácil los dispositivos registrados.

A continuación se configura el fichero “base\_dns.sh”



```

david@david: ~
GNU nano 6.2 /base_dns.sh
#!/usr/bin/env python3

import os
import subprocess

def add_resolved_conf():
    resolved_conf = "/etc/systemd/resolved.conf"
    resolved_conf_backup = "/etc/systemd/resolved.conf.bak"
    dns_config = "DNS=192.168.0.2 192.168.0.10 192.168.0.20"
    resolve_section = "[Resolve]"

    # Lee el contenido del archivo actual
    with open(resolved_conf, "r") as file:
        lines = file.readlines()

    # Verifica si la configuración ya está presente
    if any(dns_config in line for line in lines):
        print("La configuración DNS ya está presente en /etc/systemd/resolved.conf")
        return

    # Realiza una copia de seguridad del archivo original si no existe
    if not os.path.exists(resolved_conf_backup):
        subprocess.run(["sudo", "cp", resolved_conf, resolved_conf_backup])

    # Añade las configuraciones si no están presentes
    with open(resolved_conf, "a") as file:
        if not any(resolve_section in line for line in lines):
            file.write(resolve_section + "\n")
            file.write(dns_config + "\n")

    # Reinicia systemd-resolved para aplicar los cambios
    subprocess.run(["sudo", "systemctl", "restart", "systemd-resolved"])
  
```

Figura 39: Script base dns 1

```
def create_direct_zone():
    hostname = os.getlogin()
    zone_file = f"/var/lib/bind/ansible.hosts"

    if not os.path.exists(zone_file):
        os.makedirs(os.path.dirname(zone_file), exist_ok=True)
        with open(zone_file, "w") as file:
            file.write(f"""$TTL      86400

ansible.hosts.      IN      SOA      {hostname}.ansible.hosts. admin.ansible. (
                        2          ; Serial
                        604800     ; Refresh
                        86400     ; Retry
                        2419200    ; Expire
                        604800 )   ; Negative Cache TTL
;
ansible.hosts.      IN      NS      david.ansible.hosts.
ansible.hosts.      IN      NS      davidc.ansible.hosts.
ansible.hosts.      IN      NS      davidc2.ansible.hosts.

david.ansible.hosts. IN      A      192.168.0.2
davidc.ansible.hosts. IN      A      192.168.0.10
davidc2.ansible.hosts. IN      A      192.168.0.20
""")

def create_reverse_zone():
    hostname = os.getlogin()
    reverse_zone_file = f"/var/lib/bind/db.192.168.0"

    if not os.path.exists(reverse_zone_file):
        os.makedirs(os.path.dirname(reverse_zone_file), exist_ok=True)
        with open(reverse_zone_file, "w") as file:
            file.write(f"""$TTL      86400

0.168.192.in-addr.arpa.  IN      SOA      {hostname}.ansible.hosts. admin.ansible. (
                        2          ; Serial
                        604800     ; Refresh
                        86400     ; Retry
                        2419200    ; Expire
                        604800 )   ; Negative Cache TTL
;
0.168.192.in-addr.arpa.  IN      NS      david.ansible.hosts.
0.168.192.in-addr.arpa.  IN      NS      davidc.ansible.hosts.
0.168.192.in-addr.arpa.  IN      NS      davidc2.ansible.hosts.

2.0.168.192.in-addr.arpa. IN      PTR     david.ansible.hosts.
10.0.168.192.in-addr.arpa. IN      PTR     davidc.ansible.hosts.
20.0.168.192.in-addr.arpa. IN      PTR     davidc2.ansible.hosts.
""")

def update_named_conf_local():
    named_conf_local = "/etc/bind/named.conf.local"
    zone_direct = """
zone "ansible.hosts" {
    type master;
    file "/var/lib/bind/ansible.hosts";
};
"""
    zone_reverse = """
zone "0.168.192.in-addr.arpa" {
    type master;
    file "/var/lib/bind/db.192.168.0";
};
"""
```

Figura 40: Script base dns 2

## Automatización de servicios en red con Ansible

```
# Lee el contenido del archivo actual
with open(named_conf_local, "r") as file:
    content = file.read()

# Verifica si las configuraciones ya están presentes
if 'zone "ansible.hosts"' in content and 'zone "0.168.192.in-addr.arpa"' in content:
    print("Las configuraciones de zona ya están presentes en /etc/bind/named.conf.local")
    return

# Añade las configuraciones si no están presentes
with open(named_conf_local, "a") as file:
    if 'zone "ansible.hosts"' not in content:
        file.write(zone_direct)
    if 'zone "0.168.192.in-addr.arpa"' not in content:
        file.write(zone_reverse)

# Reinicia el servicio bind9 para aplicar los cambios
subprocess.run(["sudo", "systemctl", "restart", "bind9"])
subprocess.run(["sudo", "systemctl", "restart", "systemd-resolved"])

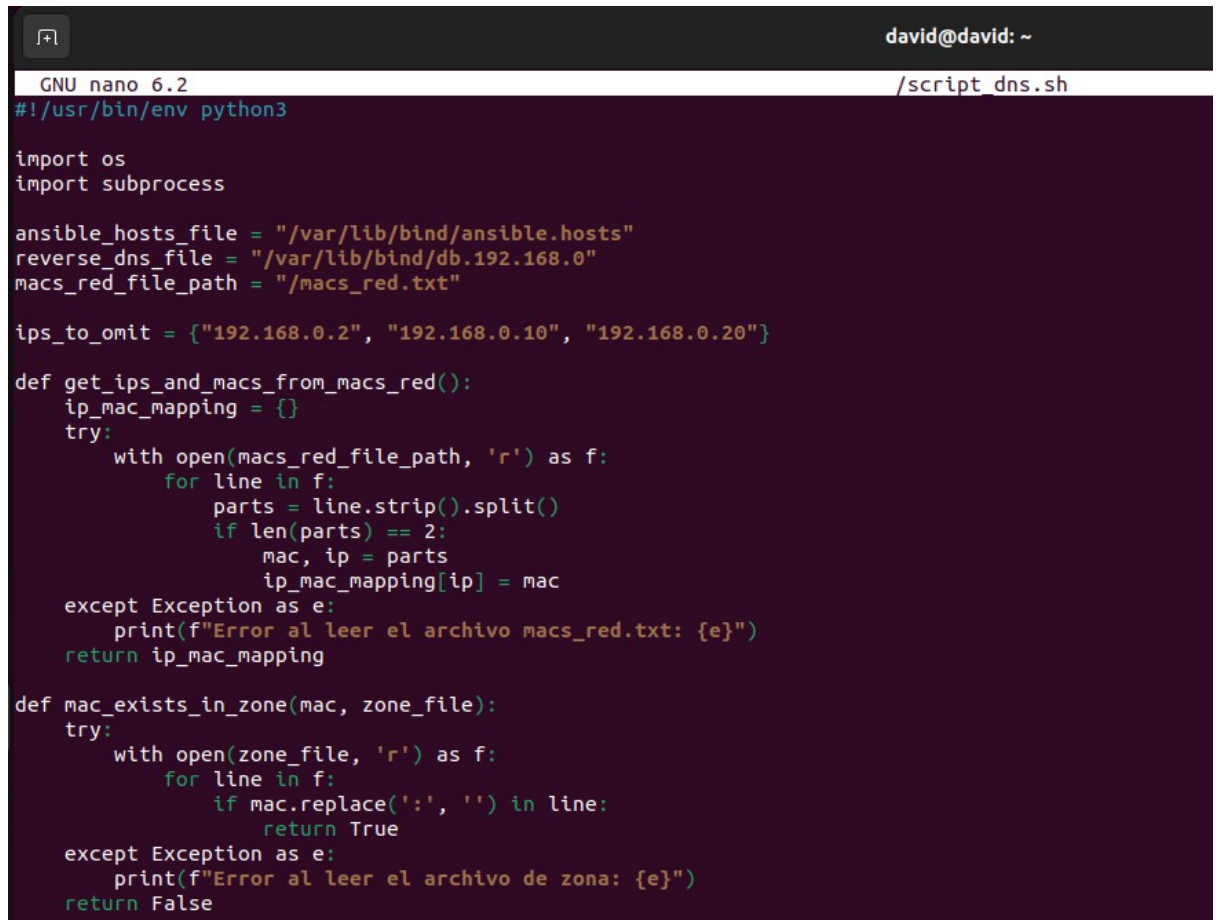
def main():
    add_resolved_conf()
    create_direct_zone()
    create_reverse_zone()
    update_named_conf_local()
    hostname = os.getlogin()
    print(f"Script completado, las configuraciones DNS han sido actualizadas para el usuario {os.getlogin()}.")

if __name__ == "__main__":
    main()
```

Figura 41: Script base dns 3

Se creará también el segundo script mencionado:





```
GNU nano 6.2 /script_dns.sh
#!/usr/bin/env python3

import os
import subprocess

ansible_hosts_file = "/var/lib/bind/ansible.hosts"
reverse_dns_file = "/var/lib/bind/db.192.168.0"
macs_red_file_path = "/macs_red.txt"

ips_to_omit = {"192.168.0.2", "192.168.0.10", "192.168.0.20"}

def get_ips_and_mac_from_mac_red():
    ip_mac_mapping = {}
    try:
        with open(macs_red_file_path, 'r') as f:
            for line in f:
                parts = line.strip().split()
                if len(parts) == 2:
                    mac, ip = parts
                    ip_mac_mapping[ip] = mac
    except Exception as e:
        print(f"Error al leer el archivo macs_red.txt: {e}")
    return ip_mac_mapping

def mac_exists_in_zone(mac, zone_file):
    try:
        with open(zone_file, 'r') as f:
            for line in f:
                if mac.replace(':', '') in line:
                    return True
    except Exception as e:
        print(f"Error al leer el archivo de zona: {e}")
    return False
```

Figura 42: Script dns 1

## Automatización de servicios en red con Ansible

```
def add_ip_to_zone(ip, mac):
    try:
        if ip in ips_to_omit:
            print(f"La IP {ip} está en la lista de IPs a omitir.")
            return

        if mac_exists_in_zone(mac, ansible_hosts_file):
            print(f"La MAC {mac} ya está en el archivo de zona directa. Actualizando la IP...")
            remove_mac_from_zone(mac, ip)

        with open(ansible_hosts_file, 'a') as f:
            f.write(f"{mac.replace(':', ' ').ansible.hosts.      IN      A      {ip}\n")

        reverse_octet = ip.split('.')[::-1]
        with open(reverse_dns_file, 'a') as f:
            f.write(f"{reverse_octet}.0.168.192.in-addr.arpa.      IN      PTR      {mac.replace(':', ' ').ansible.hosts.\n")

        print(f"IP {ip} añadida a los archivos de zona con el nombre de cliente siendo la MAC.")

        with open(ansible_hosts_file, 'a') as f:
            f.write(f"{reverse_octet}.ansible.hosts.      IN      CNAME      {mac.replace(':', ' ').ansible.hosts.\n")

        print(f"CNAME {reverse_octet}.ansible.hosts. añadido en el archivo de zona.")
    except Exception as e:
        print(f"Error al agregar la IP {ip} a los archivos de zona: {e}")

ip_mac_mapping = get_ips_and_macs_from_macs_red()

for ip, mac in ip_mac_mapping.items():
    add_ip_to_zone(ip, mac)

# Reiniciar los servicios systemd-resolved y bind9
subprocess.run(["sudo", "systemctl", "restart", "systemd-resolved"])
subprocess.run(["sudo", "systemctl", "restart", "bind9"])

print("DNS actualizado correctamente.")
```

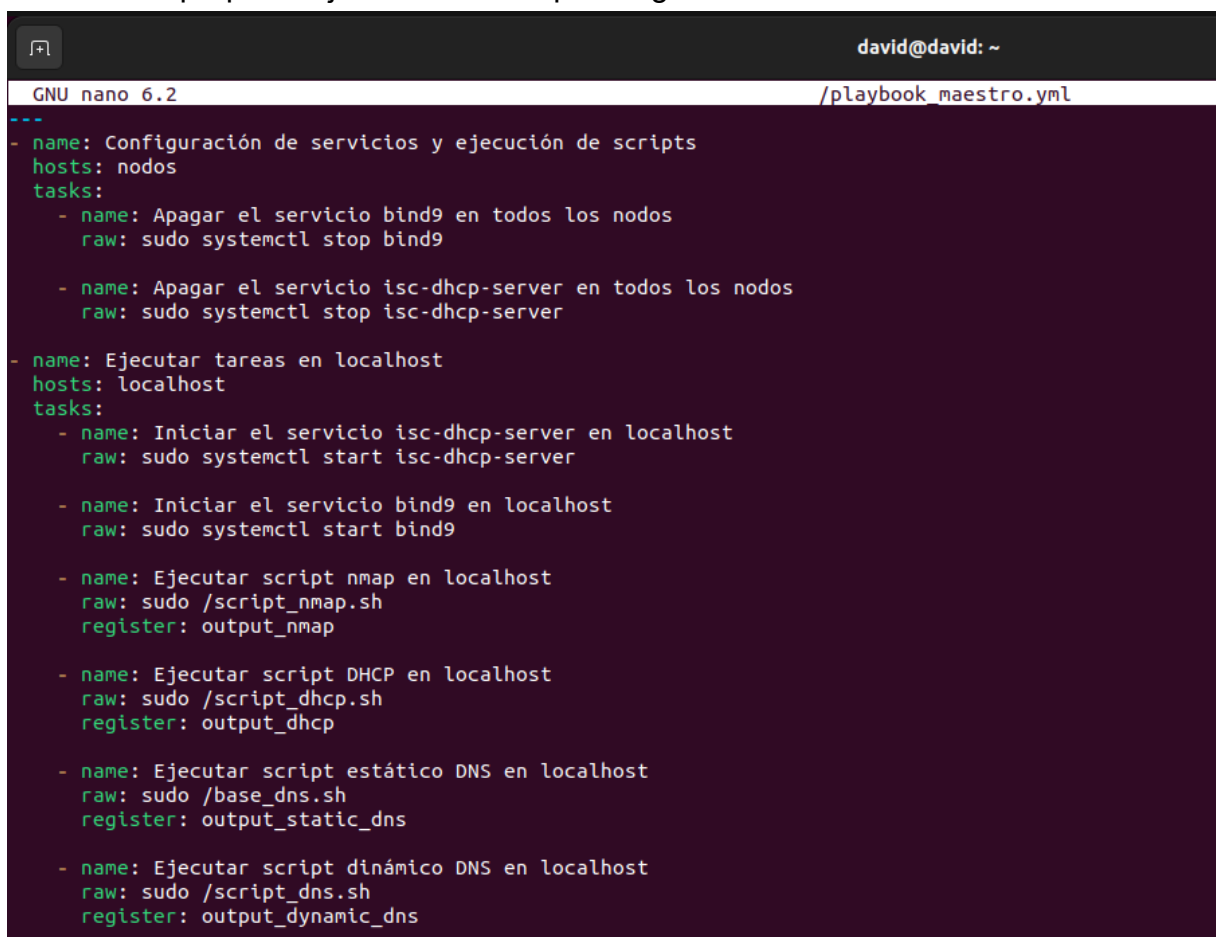
Figura 43: Script dns 2

## 8.8 Creación de playbook de automatización de servicios en red

Con todos los apartados anteriores configurados, nos faltará unificar cada script para hacer que funcionen en conjunto. Para ello se creará un playbook que los ejecute en orden.

Otro aspecto importante a tener en cuenta, es que, el playbook apagará los servicios DNS y DHCP en todos los nodos y lo encenderá únicamente en el nodo que esté ejecutando el playbook en local, así se evitarán conflictos.

Dicho playbook, se ejecutará una única vez, por lo que posteriormente se deberá crear un script que lo ejecute en bucle para lograr el funcionamiento deseado.



```

---
- name: Configuración de servicios y ejecución de scripts
  hosts: nodos
  tasks:
    - name: Apagar el servicio bind9 en todos los nodos
      raw: sudo systemctl stop bind9

    - name: Apagar el servicio isc-dhcp-server en todos los nodos
      raw: sudo systemctl stop isc-dhcp-server

- name: Ejecutar tareas en localhost
  hosts: localhost
  tasks:
    - name: Iniciar el servicio isc-dhcp-server en localhost
      raw: sudo systemctl start isc-dhcp-server

    - name: Iniciar el servicio bind9 en localhost
      raw: sudo systemctl start bind9

    - name: Ejecutar script nmap en localhost
      raw: sudo /script_nmap.sh
      register: output_nmap

    - name: Ejecutar script DHCP en localhost
      raw: sudo /script_dhcp.sh
      register: output_dhcp

    - name: Ejecutar script estático DNS en localhost
      raw: sudo /base_dns.sh
      register: output_static_dns

    - name: Ejecutar script dinámico DNS en localhost
      raw: sudo /script_dns.sh
      register: output_dynamic_dns
  
```

Figura 44: Playbook maestro

## 8.9 Creación de script de automatización de red constante

Para terminar con el proyecto, solo faltará crear un script que ejecute el playbook anterior constantemente:

```

david@david: ~
GNU nano 6.2 /script_maestro.sh
#!/bin/bash

# Función para imprimir una línea decorativa
print_line() {
    printf "%80s\n" | tr ' ' '='
}

# Función para imprimir un mensaje en una caja
print_message() {
    local message="$1"
    local len=${#message}
    local padding_left=$(( (80 - len) / 2 ))
    local padding_right=$(( 80 - len - padding_left ))
    printf "%${padding_left}s%${padding_right}s\n" "" "$message" ""
}

# Limpiar la pantalla
clear

# Imprimir el encabezado
print_line
print_message ";AUTOMATIZACIÓN DE LA RED EN EJECUCIÓN!"
print_message "Por favor, deje esta ventana de terminal abierta o ejecútela en segundo plano."
print_message "Para detener el script, pulse CTRL+C."
print_message "Se puede ejecutar también en segundo plano con '/script_maestro.sh &'"
print_line

# Obtener el ID del proceso del script
script_pid=$$

# Obtener el nombre de usuario actual
current_user=$SUDO_USER

# Si SUDO_USER está vacío, obtener el nombre de usuario actual usando whoami
if [ -z "$current_user" ]; then
    current_user=$(whoami)
fi

# Mostrar el comando para detener el script en segundo plano
echo "NOTA: Si se ejecuta el script en segundo plano, deberá detenerse con el siguiente comando: kill $script_pid"
print_line
echo ""

# Ciclo principal
while true; do
    # Ejecutar el playbook y redirigir la salida estándar y la salida de error a un archivo de registro
    sudo -u "$current_user" ansible-playbook -i /hosts /playbook_maestro.yml

    # Mostrar el ID del proceso generado
    echo "PROCESO DEL SCRIPT ---> $script_pid"
    print_line

    # Copiar fichero macs en el resto de nodos
    sudo -u "$current_user" scp /macs_red.txt davidc@192.168.0.10:/macs_red.txt
    sudo -u "$current_user" scp /macs_red.txt davidc2@192.168.0.20:/macs_red.txt
    # Esperar 30s antes de la próxima ejecución
    sleep 30
done

```

Figura 45: Script maestro

## 8.10 Comprobación de funcionamiento en nodo cualquiera

Son muchas las comprobaciones que se pueden realizar, en la guía de usuario, se prueban uno a uno todos los scripts ejecutados con más detalles de cada paso realizado. En esta memoria del proyecto, se va a comprobar una ejecución general del mismo, para verificar que se ha cumplido el objetivo:

```

=====
;AUTOMATIZACIÓN DE LA RED EN EJECUCIÓN!
Por favor, deje esta ventana de terminal abierta o ejecútela en segundo plano.
Para detener el script, pulse CTRL+C.
Se puede ejecutar también en segundo plano con '/script_maestro.sh &'
=====
NOTA: Si se ejecuta el script en segundo plano, deberá detenerse con el siguiente comando: kill 10833
=====

PLAY [Configuración de servicios y ejecución de scripts] *****

TASK [Gathering Facts] *****
ok: [192.168.0.2]
ok: [192.168.0.20]
ok: [192.168.0.10]

TASK [Apagar el servicio bind9 en todos los nodos] *****
changed: [192.168.0.10]
changed: [192.168.0.20]
changed: [192.168.0.2]

TASK [Apagar el servicio isc-dhcp-server en todos los nodos] *****
changed: [192.168.0.2]
changed: [192.168.0.10]
changed: [192.168.0.20]

PLAY [Ejecutar tareas en localhost] *****

TASK [Gathering Facts] *****
ok: [localhost]

TASK [Iniciar el servicio isc-dhcp-server en localhost] *****
changed: [localhost]

TASK [Iniciar el servicio bind9 en localhost] *****
changed: [localhost]

TASK [Ejecutar script nmap en localhost] *****
changed: [localhost]

TASK [Ejecutar script DHCP en localhost] *****
changed: [localhost]

TASK [Ejecutar script estático DNS en localhost] *****
changed: [localhost]

TASK [Ejecutar script dinámico DNS en localhost] *****

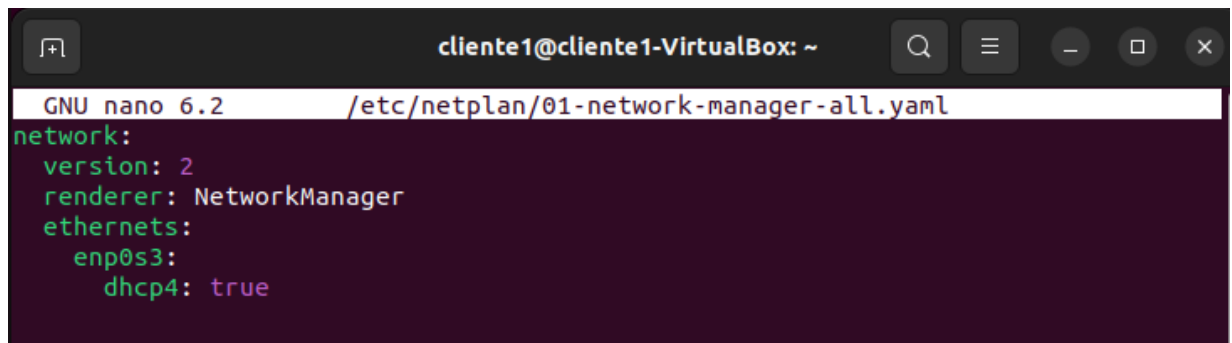
```

Figura 46: Pruebas ejecución 1

Como se puede observar, está ejecutándose correctamente.

A continuación, teniendo el script en bucle, se va a conectar un nuevo equipo a la red para verificar que se añade a los servicios DHCP y DNS:

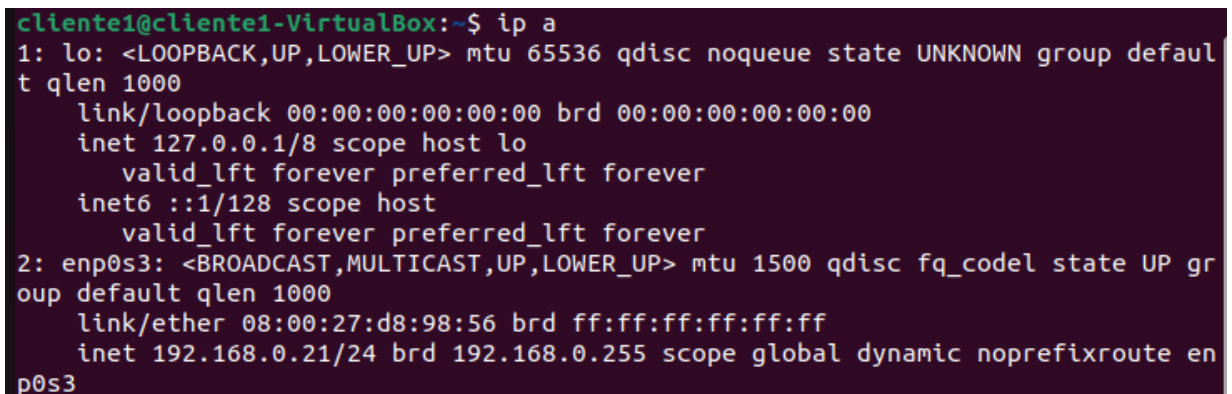
netplan del cliente nuevo:



```
cliente1@cliente1-VirtualBox: ~  
GNU nano 6.2 /etc/netplan/01-network-manager-all.yaml  
network:  
  version: 2  
  renderer: NetworkManager  
  ethernets:  
    enp0s3:  
      dhcp4: true
```

Figura 47: Pruebas ejecución 2

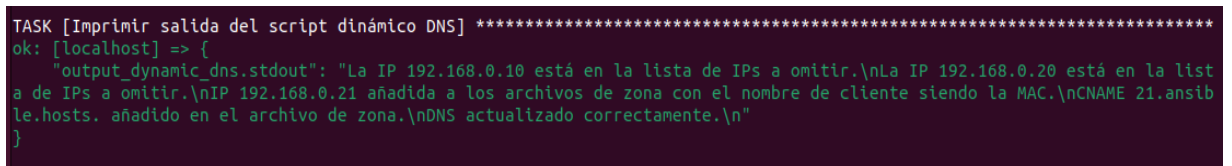
IP Asignada: 192.168.0.21



```
cliente1@cliente1-VirtualBox:~$ ip a  
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000  
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00  
    inet 127.0.0.1/8 scope host lo  
        valid_lft forever preferred_lft forever  
    inet6 ::1/128 scope host  
        valid_lft forever preferred_lft forever  
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000  
    link/ether 08:00:27:d8:98:56 brd ff:ff:ff:ff:ff:ff  
    inet 192.168.0.21/24 brd 192.168.0.255 scope global dynamic noprefixroute enp0s3
```

Figura 48: Pruebas ejecución 3

El script detectará el nuevo host. En la salida del script dinámico DNS, se puede comprobar que se ha añadido la IP 192.168.0.21



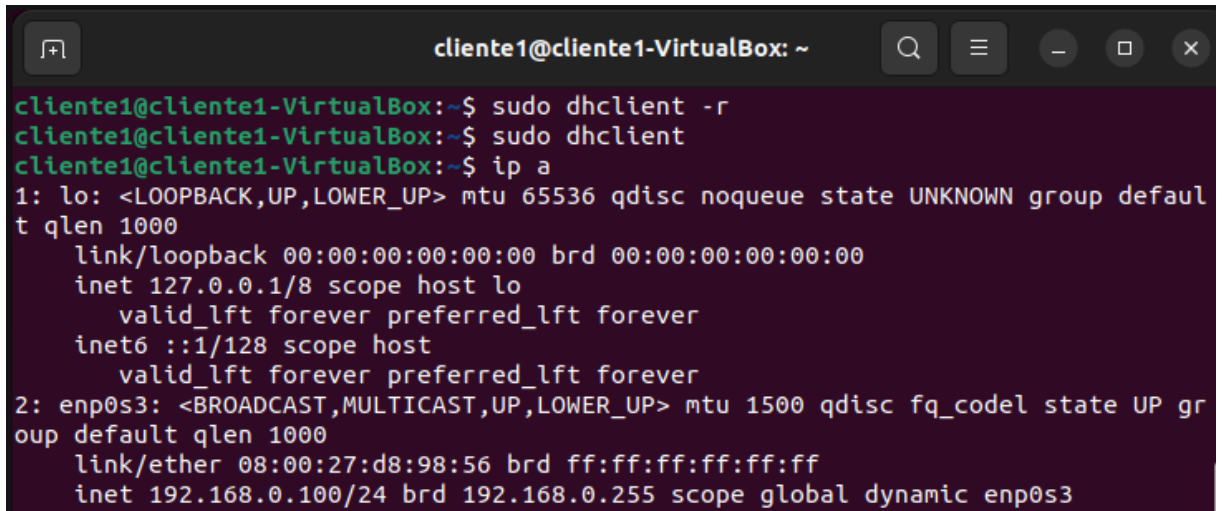
```
TASK [Imprimir salida del script dinámico DNS] *****  
ok: [localhost] => {  
  "output_dynamic_dns.stdout": "La IP 192.168.0.10 está en la lista de IPs a omitir.\nLa IP 192.168.0.20 está en la lista de IPs a omitir.\nIP 192.168.0.21 añadida a los archivos de zona con el nombre de cliente siendo la MAC.\n\nNAME 21.ansible.hosts. añadido en el archivo de zona.\n\nDNS actualizado correctamente.\n"}  
}
```

Figura 49: Pruebas ejecución 4



## Automatización de servicios en red con Ansible

ahora el cliente deberá hacer una renovación de IP:

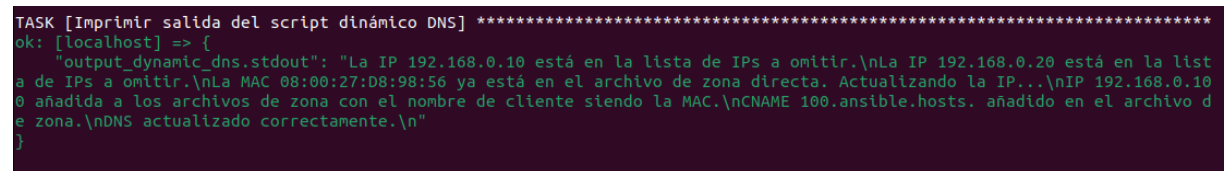


```
cliente1@cliente1-VirtualBox: ~  
cliente1@cliente1-VirtualBox:~$ sudo dhclient -r  
cliente1@cliente1-VirtualBox:~$ sudo dhclient  
cliente1@cliente1-VirtualBox:~$ ip a  
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000  
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00  
    inet 127.0.0.1/8 scope host lo  
        valid_lft forever preferred_lft forever  
    inet6 ::1/128 scope host  
        valid_lft forever preferred_lft forever  
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000  
    link/ether 08:00:27:d8:98:56 brd ff:ff:ff:ff:ff:ff  
    inet 192.168.0.100/24 brd 192.168.0.255 scope global dynamic enp0s3
```

Figura 50: Pruebas ejecución 5

Como podemos observar, tras renovarla, ya pertenece al rango de IPs reservadas (192.168.0.100).

Cuando el script vuelva a ejecutarse, actualizará la IP en el DNS:

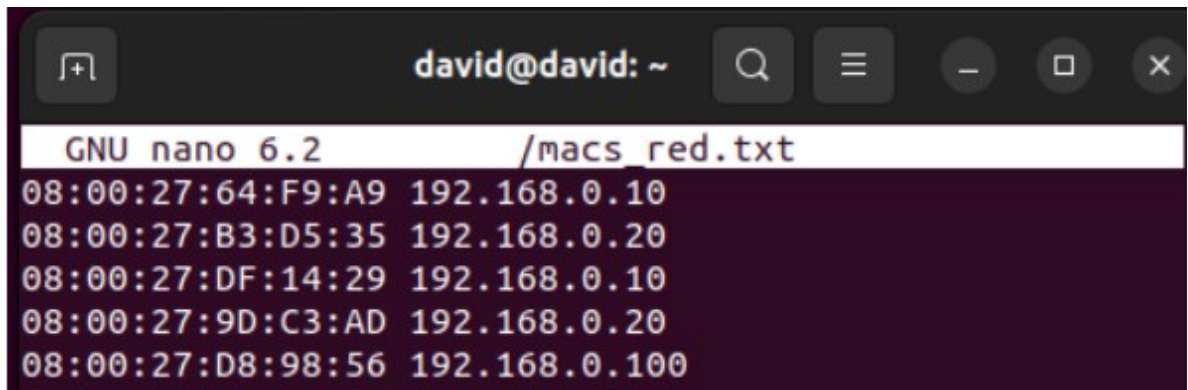


```
TASK [Imprimir salida del script dinámico DNS] *****  
ok: [localhost] => {  
    "output_dynamic_dns.stdout": "La IP 192.168.0.10 está en la lista de IPs a omitir.\nLa IP 192.168.0.20 está en la lista de IPs a omitir.\nLa MAC 08:00:27:D8:98:56 ya está en el archivo de zona directa. Actualizando la IP...\nIP 192.168.0.100 añadida a los archivos de zona con el nombre de cliente siendo la MAC.\nCNAME 100.ansible.hosts. añadido en el archivo de zona.\nDNS actualizado correctamente.\n"}
```

Figura 51: Pruebas ejecución 6

A continuación, se comprobará que no hay duplicidad en ningún fichero de los implicados:

FICHERO DE MACS:

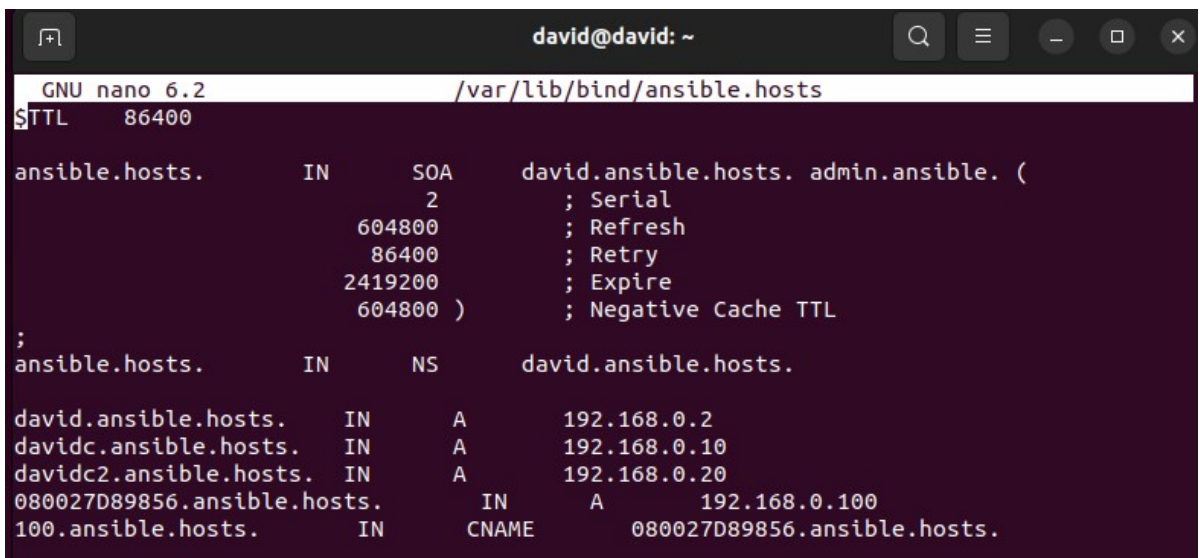


```
GNU nano 6.2 /macs_red.txt
08:00:27:64:F9:A9 192.168.0.10
08:00:27:B3:D5:35 192.168.0.20
08:00:27:DF:14:29 192.168.0.10
08:00:27:9D:C3:AD 192.168.0.20
08:00:27:D8:98:56 192.168.0.100
```

Figura 52: Pruebas ejecución 7

FICHEROS DNS:

Zona directa:



```
GNU nano 6.2 /var/lib/bind/ansible.hosts
$TTL      86400

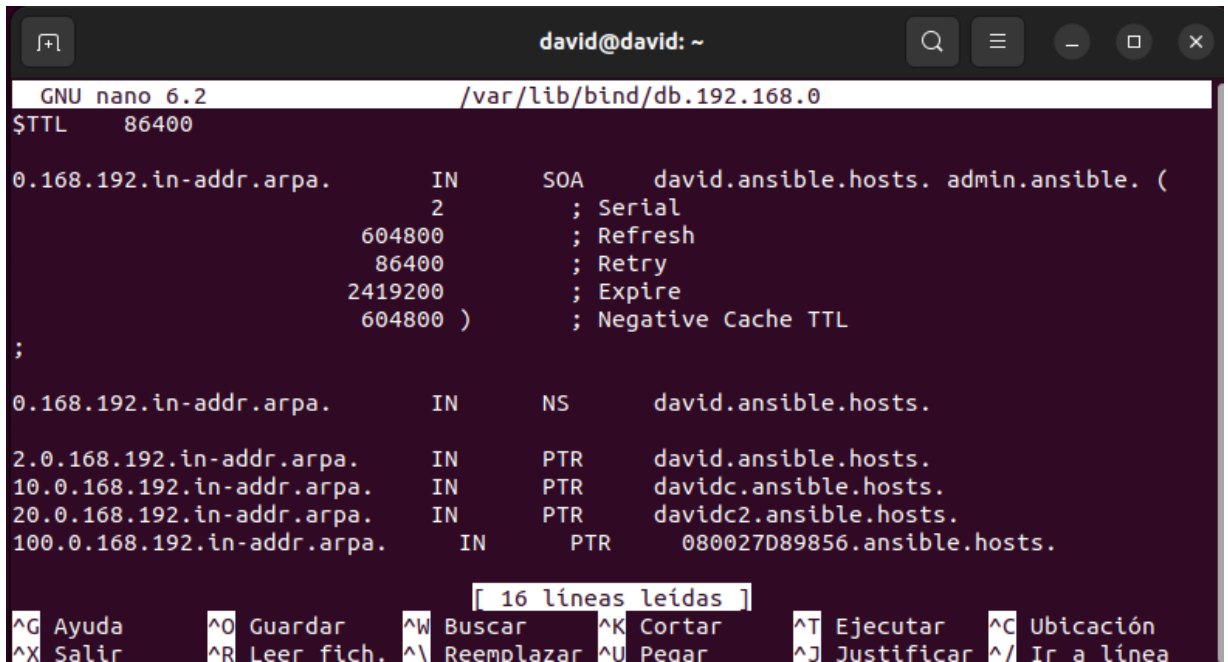
ansible.hosts.      IN      SOA      david.ansible.hosts. admin.ansible. (
                        2          ; Serial
                        604800     ; Refresh
                        86400     ; Retry
                        2419200    ; Expire
                        604800 )   ; Negative Cache TTL
;
ansible.hosts.      IN      NS       david.ansible.hosts.
david.ansible.hosts. IN      A        192.168.0.2
davidc.ansible.hosts. IN     A        192.168.0.10
davidc2.ansible.hosts. IN    A        192.168.0.20
080027D89856.ansible.hosts. IN  A        192.168.0.100
100.ansible.hosts.  IN      CNAME     080027D89856.ansible.hosts.
```

Figura 53: Pruebas ejecución 8



## Automatización de servicios en red con Ansible

Zona inversa:



```
GNU nano 6.2 /var/lib/bind/db.192.168.0
$TTL      86400

0.168.192.in-addr.arpa.    IN      SOA      david.ansible.hosts. admin.ansible. (
                           2          ; Serial
                           604800     ; Refresh
                           86400      ; Retry
                           2419200    ; Expire
                           604800 )   ; Negative Cache TTL
;


0.168.192.in-addr.arpa.    IN      NS       david.ansible.hosts.

2.0.168.192.in-addr.arpa. IN      PTR      david.ansible.hosts.
10.0.168.192.in-addr.arpa. IN     PTR      davidc.ansible.hosts.
20.0.168.192.in-addr.arpa. IN     PTR      davidc2.ansible.hosts.
100.0.168.192.in-addr.arpa. IN    PTR      080027D89856.ansible.hosts.

[ 16 líneas leídas ]
^G Ayuda      ^O Guardar    ^W Buscar     ^K Cortar     ^T Ejecutar   ^C Ubicación
^X Salir      ^R Leer fich. ^\ Reemplazar ^U Pegar      ^J Justificar ^_ Ir a línea
```

Figura 54: Pruebas ejecución 9

FICHERO DHCP:



```
GNU nano 6.2 /etc/dhcp/dhcpd.conf

# Configuración básica de DHCP para la subred 192.168.0.0/24
default-lease-time 600;
max-lease-time 7200;

subnet 192.168.0.0 netmask 255.255.255.0 {
    # Rango de direcciones dinámicas
    range 192.168.0.21 192.168.0.99;

    option routers 192.168.0.1;
    option subnet-mask 255.255.255.0;

    host 080027D89856 {
        hardware ethernet 08:00:27:D8:98:56;
        fixed-address 192.168.0.100;
    }
}
```

Figura 55: Pruebas ejecución 10

DIG DIRECTO AL NUEVO CLIENTE:

```
david@david:~$ dig 100.ansible.hosts

; <<>> DiG 9.18.18-0ubuntu0.22.04.2-Ubuntu <<>> 100.ansible.hosts
;; global options: +cmd
;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 33492
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;100.ansible.hosts.                IN      A

;; ANSWER SECTION:
100.ansible.hosts.      86400   IN      CNAME   080027D89856.ansible.hosts.
080027D89856.ansible.hosts. 86400   IN      A       192.168.0.100

;; Query time: 7 msec
;; SERVER: 127.0.0.53#53(127.0.0.53) (UDP)
;; WHEN: Mon Jun 10 18:31:59 CEST 2024
;; MSG SIZE rcvd: 89
```

Figura 56: dig a cliente con answer section

DIG INVERSO AL NUEVO CLIENTE:

```
david@david:~$ dig -x 192.168.0.100

; <<>> DiG 9.18.18-0ubuntu0.22.04.2-Ubuntu <<>> -x 192.168.0.100
;; global options: +cmd
;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 46048
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;100.0.168.192.in-addr.arpa.      IN      PTR

;; ANSWER SECTION:
100.0.168.192.in-addr.arpa. 86400   IN      PTR      080027D89856.ansible.hosts.

;; Query time: 4 msec
;; SERVER: 127.0.0.53#53(127.0.0.53) (UDP)
;; WHEN: Mon Jun 10 18:32:51 CEST 2024
;; MSG SIZE rcvd: 95
```

Figura 57: dig inverso a cliente con answer section

aclarar que será tarea del administrador verificar que se copia el fichero más reciente de macs al resto de nodos utilizando SSH. Esta tarea se ha añadido al script para que se realice automáticamente, pero será conveniente revisarlo para que los nodos sigan coordinados entre sí.

## **9 PROPUESTA DE MEJORA**

Este proyecto se puede mejorar de diversas formas. La primera mejora que debería considerarse es conseguir automatizar la configuración de los inventarios en todos los nodos modificando el playbook “/instalar\_ansible.yml”.

Otra tarea que había como objetivo principal pero se tuvo que adaptar por falta de tiempo, es la coordinación automática de los nodos por prioridades para que, en caso de que un nodo falle, el servicio siga proporcionándose por cualquier otro nodo disponible en la cola de prioridades. Con esto conseguiríamos mejorar el sistema de alta disponibilidad.

Como mejoras menos relevantes pero interesantes, podría considerarse implementar un firewall para añadir seguridad a los servidores filtrando el tráfico que reciben. Incluso se podría coordinar con una base de datos que sirva para asignar a cada equipo de la red con reserva, un usuario y contraseña para acceder a las aplicaciones de nivel interno de la empresa.

También podría añadirse una lista negra de macs excluidas y añadir todas las nuevas macs que se detecten automáticamente ahí para que, después de pasar un filtro, se decida que dispositivos están permitidos y cuales no. Añadiendo así una capa de seguridad al proyecto.

## **10 CONCLUSIONES**

Como se ha podido comprobar durante el proyecto, tiene mucho potencial y margen de mejora, lo que lo hace interesante. En el apartado de mejoras, se proponen dos ideas que estaban planteadas para ejecutarse en el proyecto, pero que, por falta de tiempo, no han podido llevarse a cabo y se han tenido que adaptar.

Personalmente me parece un proyecto complejo por la cantidad de detalles a tener en cuenta, pero, que hacen del proyecto algo diferente a lo que hay actualmente en cualquier empresa. Además, con la evolución que está teniendo la inteligencia artificial, cada día existen más herramientas para aprender y facilitar la creación de programas y scripts, lo que reduce los tiempos de depuración y de comprensión de nuevos lenguajes, como en este caso ha ocurrido con YAML.

## 11 FUENTES

ChatGPT:

<https://chatgpt.com/>

Curso de Ansible en castellano:

[https://youtube.com/playlist?  
list=PLd7FFr2YzghNETVzT99w0hiWUDRNsqkq6&si=X7KwEGqzgYU2Bqci](https://youtube.com/playlist?list=PLd7FFr2YzghNETVzT99w0hiWUDRNsqkq6&si=X7KwEGqzgYU2Bqci)

Guía explicativa sobre ansible:

<https://blog.invgate.com/es/ansible>

Guía oficial de Ansible playbooks:

[https://docs.ansible.com/ansible/latest/playbook\\_guide/playbooks\\_intro.html](https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_intro.html)

Guía oficial de comandos Ansible:

[https://docs.ansible.com/ansible/latest/command\\_guide/cheatsheet.html](https://docs.ansible.com/ansible/latest/command_guide/cheatsheet.html)

## **12 ANEXOS**

Proyecto, ficheros y manual de usuario:

<https://github.com/davidbanon/Automatizacion-de-servicios-en-red-con-Ansible>