

INGENIERÍA DE SERVIDORES (2021-2022)  
GRADO EN INGENIERÍA INFORMÁTICA  
UNIVERSIDAD DE GRANADA

---

## Memoria Práctica 3

---

Adrián Acosa Sánchez

2 de diciembre de 2021

# Índice

<b>1</b>	<b>Identificación, monitorización y recuperación de RAID</b>	<b>3</b>
1.1	Recuperación del RAID . . . . .	4
<b>2</b>	<b>Instalación y configuración de Zabbix 5.0</b>	<b>5</b>
2.1	Instalación de Zabbix . . . . .	5
2.2	Configuración de Zabbix . . . . .	7
2.3	Monitorización de HTTP con Zabbix . . . . .	12
2.4	Monitorización de SSH con Zabbix . . . . .	13
2.5	Monitorización de otro servidor con Zabbix . . . . .	14
<b>3</b>	<b>Instalación y configuración de Ansible</b>	<b>15</b>
3.1	Instalación de Ansible . . . . .	15
3.2	Configuración de Ansible . . . . .	17

# 1. Identificación, monitorización y recuperación de RAID

Para empezar con el ejercicio, partimos de una máquina virtual con un RAID 5 de 4 discos virtuales. Para hacer que falle, quitamos uno de los discos y añadimos otro vacío. Para ver el estado de nuestros discos hacemos uso de la opción -D de mdadm de la siguiente manera

```
mdadm -D [nom_dispositivo]
```

En este caso tenemos en nuestra máquina virtual lo siguiente:

```
adrianas@localhost ~]$ sudo mdadm -D /dev/md0
[sudo] password for adrianas:
/dev/md0:
  Version : 1.2
  Creation Time : Sat Nov 27 07:36:29 2021
  Raid Level : raid5
  Array Size : 25145856 (23.98 GiB 25.75 GB)
  Used Dev Size : 8381952 (7.99 GiB 8.58 GB)
  Raid Devices : 4
  Total Devices : 3
  Persistence : Superblock is persistent

  Update Time : Tue Nov 30 12:14:30 2021
  State : clean, degraded
  Active Devices : 3
  Working Devices : 3
  Failed Devices : 0
  Spare Devices : 0

  Layout : left-symmetric
  Chunk Size : 512K

Consistency Policy : resync

           Name : localhost.localdomain:0 (local to host localhost.localdomain)
           UUID : 4e8d3438:08350372:0ac2afa9:63bd0ebe
           Events : 22

Number   Major   Minor   RaidDevice State
  0         8       17         0   active sync  /dev/sdb1
  1         8       49         1   active sync  /dev/sdd1
  2         8       65         2   active sync  /dev/sde1
  -         0         0         3   removed
```

Figura 1.1: Se puede apreciar que hay un disco del RAID que aparece como eliminado, aunque el sistema sigue en funcionamiento al no suponer un error grave.

Haciendo uso del comando journalctl y buscando como palabra clave el dispositivo md0 en este caso, vemos que no supone un error grave el haber quitado uno de los discos que componían el RAID.

El comando a utilizar para ver esto es el siguiente:

```
journalctl -k -f | grep md0
```

```
adrianas@localhost ~]$ sudo journalctl -k -f | grep md0
Nov 30 12:14:29 localhost.localdomain kernel: XFS (md0): Mounting U5 Filesystem
Nov 30 12:14:29 localhost.localdomain kernel: XFS (md0): Ending clean mount
```

Figura 1.2: Podemos comprobar en la imagen que no supone un problema grave para el sistema.

Sea como sea, la forma de actuar ante estos casos es crear un log para poder informar al administrador del sistema y configurar un servicio de correo para que le notifique en caso de fallo de alguno de los dispositivos.

En primer lugar creamos un archivo de configuración al que le redirigimos el resultado de la monitorización del dispositivo afectado con el siguiente comando:

```
mdadm --detail --scan >> /etc/mdadm.conf
```

Tras crear este archivo, añadimos la variable MAILADDR con la dirección de correo del administrador del sistema para que le notifique en caso de fallo. Lo haríamos de la siguiente manera:

```
MAILADDR adrianacosa@correo.ugr.es
```

### 1.1. Recuperación del RAID

Para la recuperación de nuestro RAID, tenemos que seguir los siguientes pasos:

1. Añadimos el nuevo disco vacío al RAID
2. Marcamos el disco como fallo
3. Comprobamos si el disco ha sido marcado como fallo correctamente
4. Retiramos el disco fallido
5. Comprobamos los detalles del RAID

Para ello ejecutamos los siguientes comandos:

```
mdadm --manage /dev/md0 --add /dev/sdi  
mdadm --manage /dev/md0 --fail /dev/sdb  
mdadm --manage /dev/md0 --remove /dev/sdb  
mdadm --detail /dev/md0
```

Después de ejecutar cada uno de los comandos, tendríamos el disco nuevo añadido a nuestro RAID de manera correcta.

## 2. Instalación y configuración de Zabbix 5.0

### 2.1. Instalación de Zabbix

Antes de empezar a instalar Zabbix como tal, tenemos que instalar sus dependencias. Los paquetes necesarios para poder usar Zabbix son:

1. MySQL / MariaDB
2. Servidor web Apache
3. PHP con las extensiones requeridas

Primero tenemos que añadir los repositorios de zabbix a nuestro sistema:

```
[adrianas@localhost ~]$ rpm -Uvh https://repo.zabbix.com/zabbix/5.0/rhel/8/x86_64/zabbix-release-5.0-1.el8.noarch.rpm
Retrieving https://repo.zabbix.com/zabbix/5.0/rhel/8/x86_64/zabbix-release-5.0-1.el8.noarch.rpm
warning: /var/tmp/rpm-tmp.zfM018: Header V4 RSA/SHA512 Signature, key ID a14fe591: NOKEY
error: can't create transaction lock on /var/lib/rpm/.rpm.lock (Permission denied)
[adrianas@localhost ~]$ sudo rpm -Uvh https://repo.zabbix.com/zabbix/5.0/rhel/8/x86_64/zabbix-release-5.0-1.el8.noarch.rpm
[sudo] password for adrianas:
Retrieving https://repo.zabbix.com/zabbix/5.0/rhel/8/x86_64/zabbix-release-5.0-1.el8.noarch.rpm
warning: /var/tmp/rpm-tmp.SPwbz5: Header V4 RSA/SHA512 Signature, key ID a14fe591: NOKEY
Verifying...
Preparing...
Updating / installing...
 1:zabbix-release-5.0-1.el8
[adrianas@localhost ~]$ dnf clean all
0 files removed
```

Figura 2.1: Proceso de obtención del repositorio de Zabbix

Tras esto, procedemos a instalar el servidor de Zabbix, su frontend y su agent:

```
dnf install zabbix-server-mysql
dnf install zabbix-web-mysql
dnf install zabbix-apache-conf
dnf install zabbix-agent
```

El siguiente paso a realizar es crear la base de datos necesaria para nuestro servidor de Zabbix. Para ello ejecutamos los siguientes pasos dentro de mysql:

```
CREATE DATABASE zabbix CHARACTER SET utf8 collate utf8_bin;
CREATE USER zabbix@localhost IDENTIFIED BY 'password';
GRANT ALL PRIVILEGES ON zabbix.* TO zabbix@localhost;
```

```

ladrianas@localhost ~]$ sudo mysql -uroot
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 8.0.26 Source distribution

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database zabbix character set utf8 collate utf8_bin;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your
MySQL server version for the right syntax to use near 'collate utf8_bin' at line 1
mysql> create database zabbix character set utf8 collate utf8_bin;
Query OK, 1 row affected, 2 warnings (0.01 sec)

mysql> create user zabbix@localhost identified by 'password';
Query OK, 0 rows affected (0.00 sec)

mysql> grant all privileges on zabbix.* to zabbix@localhost;
Query OK, 0 rows affected (0.00 sec)

mysql> quit;
Bye

```

Figura 2.2: Puesta a punto de la base de datos de Zabbix

Ahora tenemos que importar los datos iniciales para el servidor. Esto se realiza con el siguiente comando:

```

zcat /usr/share/doc/zabbix-server-mysql*/create.sql.gz \
| mysql -uzabbix -p zabbix

```

Una vez hecho esto tendremos que realizar los siguientes pasos:

1. Descomentamos la línea `DBPassword=password` del archivo `/etc/zabbix/zabbix_server.conf`.
2. Descomentamos la línea `php_value[date.timezone] = Europe/Riga` del archivo `/etc/php-fpm.d/zabbix.conf`.
3. Activamos el servidor Zabbix (zabbix-server, zabbix-agent, httpd y php-fpm) con "systemctl" para que se inicie al iniciar el sistema.
4. Configuramos el frontend de Zabbix desde la página <http://127.0.0.1/zabbix>

Una cosa a tener en cuenta antes de acabar la instalación es que hay que configurar SELinux para poder iniciar los servicios de Zabbix. En concreto, hay que activar los booleanos de `httpd_can_connect_zabbix` y `httpd_can_network_connect_db`.

Durante la puesta en marcha del servidor, he tenido algunos problemas ya que no me dejaba reiniciar los servicios necesarios para activar Zabbix. En concreto he tenido que ejecutar `journalctl -xe` y viendo el error he visto que era necesario ejecutar un comando:

```

semodule -X 300 -i my-zabbixserver.pp

```

Tras esto, todo ha funcionado correctamente y tendríamos toda la instalación completa.

## 2.2. Configuración de Zabbix

Ahora vamos a pasar a configurar SELinux para que permita que el backend de zabbix se pueda conectar con su frontend. Para ello ejecutamos el siguiente comando:

```
sudo setenforce 0
```

También tenemos que añadir los puertos de zabbix y de httpd al firewall para que nos permita conectarnos:

```
sudo firewall-cmd \
--add-port={80,10051,10050}/tcp --permanent
sudo firewall-cmd --reload
```

Una vez hecho esto, ya podremos configurar la página web que nos proporciona Zabbix. Nada más entrar en la web nos salta una página de puesta a punto para que configuremos la interfaz.

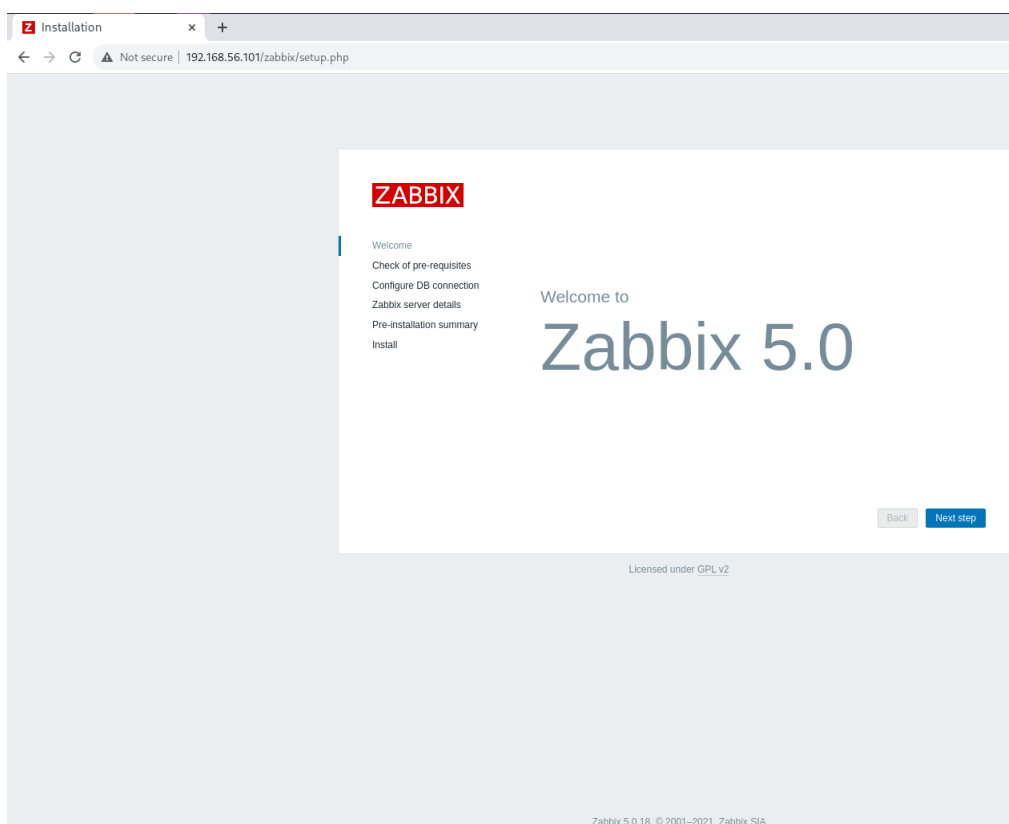


Figura 2.3: Página inicial de "setup" de Zabbix

Si le damos a "Next step" nos lleva a la siguiente página donde podremos comprobar si tenemos todos los requisitos para poder configurar la interfaz web:

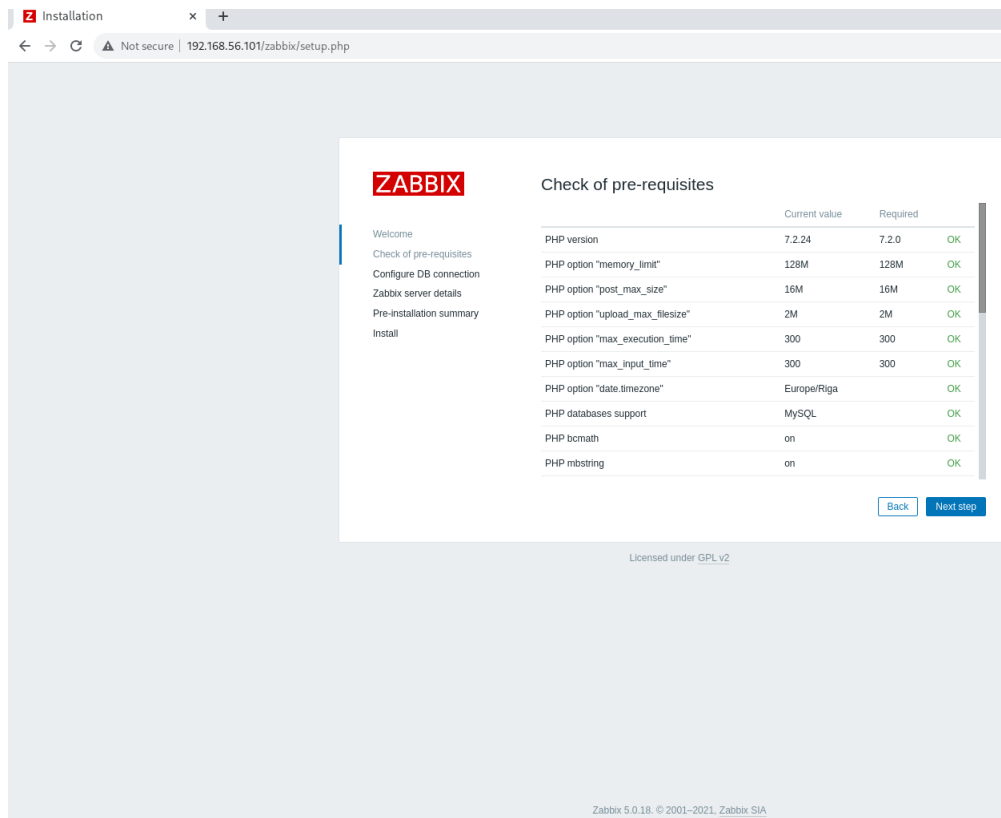


Figura 2.4: Página de comprobación de prerequisites

Una vez hemos comprobado que todos los requisitos se cumplen, pasamos al siguiente paso que consiste en configurar la conexión a la base de datos. Para ello tenemos que recordar los datos que pusimos en la configuración de MySQL.



**ZABBIX**

Configure DB connection

Please create database manually, and set the configuration parameters for connection to this database. Press "Next step" button when done.

Database type: MySQL

Database host: localhost

Database port: 0 0 - use default port

Database name: zabbix

User: zabbix

Password: .....

Database TLS encryption: Connection will not be encrypted because it uses a socket file (on Unix) or shared memory (Windows).

Back Next step

Figura 2.5: Rellenamos los datos de acuerdo a las credenciales de MySQL

En la siguiente página nos encontramos con la información relativa a donde se va a alojar el servidor y qué nombre queremos ponerle al servicio.

**ZABBIX**

Zabbix server details

Please enter the host name or host IP address and port number of the Zabbix server, as well as the name of the installation (optional).

Host: localhost

Port: 10051

Name: zabbixrockylinux.com

Back Next step

Figura 2.6: En este caso le ponemos el nombre zabbixrockylinux.com

Ahora en las dos siguientes capturas podemos ver un sumario de todos los datos que hemos introducido para poder asegurarnos de si está todo bien, y un mensaje de éxito en la configuracion.

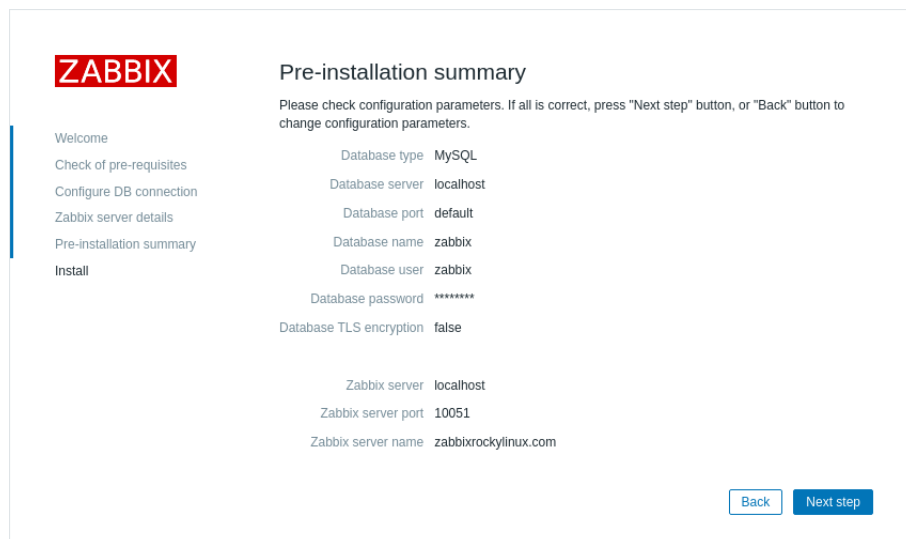


Figura 2.7: Resumen de la instalación

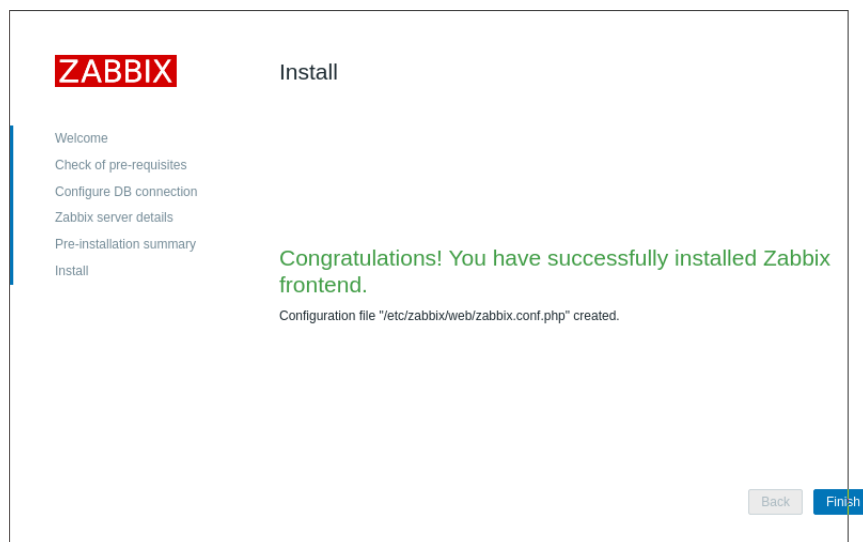


Figura 2.8: Mensaje de éxito en la instalación del frontend de zabbix

Tras esto ya tendríamos Zabbix configurado. Lo último que tenemos que hacer antes de entrar al servicio es conocer las credenciales por defecto, que en este caso son:

**Username:** Admin  
**Password:** zabbix

**ZABBIX**

Username

Admin

Password

\*\*\*\*\*

☒ Remember me for 30 days

Sign in

Figura 2.9: Página de inicio de sesión

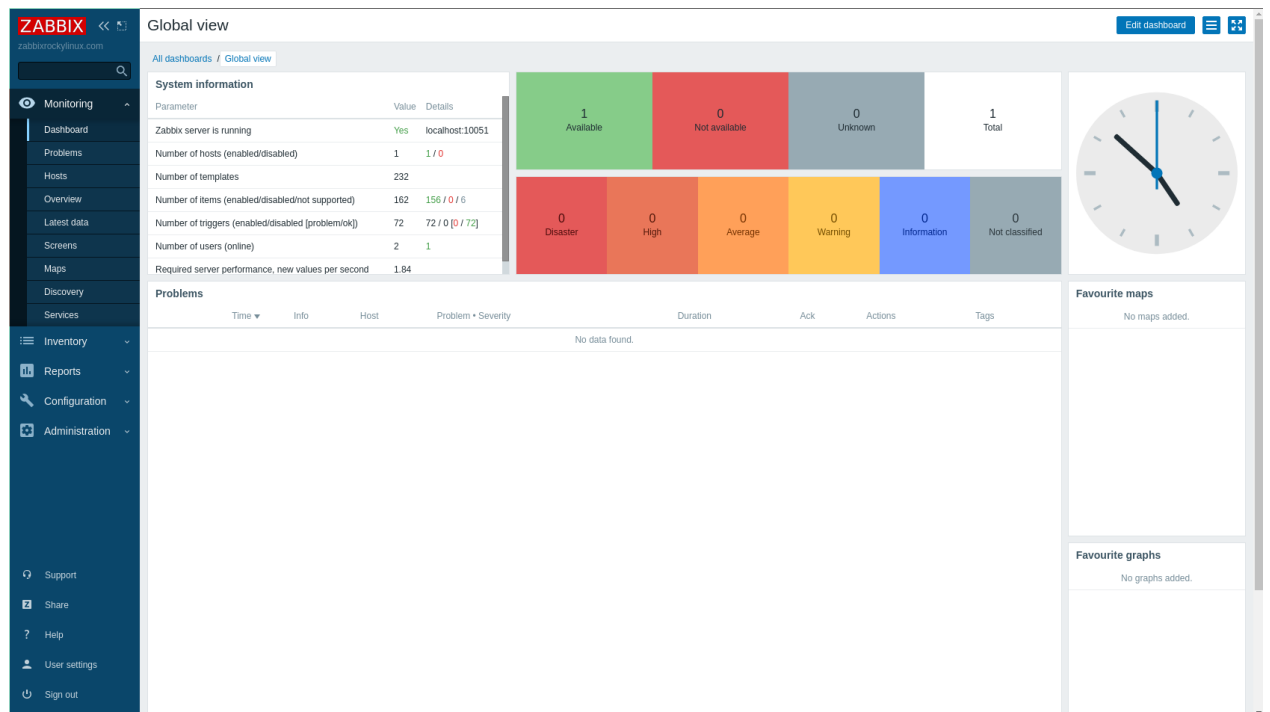


Figura 2.10: Dashboard de zabbix

### 2.3. Monitorización de HTTP con Zabbix

## 2.4. Monitorización de SSH con Zabbix

## 2.5. Monitorización de otro servidor con Zabbix

### 3. Instalación y configuración de Ansible

Ansible es un motor opensource que automatiza los procesos para preparar la infraestructura, gestionar la configuración, implementar aplicaciones y organizar los sistemas, entre otros procedimientos de TI.

Ansible permite instalar sistemas de software, automatizar las tareas diarias, preparar la infraestructura, mejorar la seguridad y el cumplimiento, ejecutar parches en los sistemas y compartir la automatización en toda una empresa.

#### 3.1. Instalación de Ansible

Para empezar, vamos a ver como se instala ansible en nuestro sistema. Lo primero que haremos sera instalar los paquetes adicionales para linux empresarial (EPEL) a partir del siguiente comando:

```
sudo dnf install epel-release
```

```
[adrianas@localhost ~]$ sudo dnf install epel-release
[sudo] password for adrianas:
Last metadata expiration check: 1 day, 23:33:23 ago on Tue 30 Nov 2021 01:18:56 PM EST.
Dependencies resolved.
=====
Package                        Architecture    Version         Repository      Size
=====
Installing:
epel-release                   noarch         8-13.el8        extras          23 k
=====
Transaction Summary
=====
Install 1 Package

Total download size: 23 k
Installed size: 35 k
Is this ok [y/N]:
```

Figura 3.1: Instalación del repositorio de EPEL

Tras esto ya tendremos acceso al paquete de ansible en nuestro sistema. Para instalarlo tenemos que ejecutar:

```
sudo dnf install ansible
```

```
Dependencies resolved.
```

Package	Arch	Version	Repository	Size
Installing:				
ansible	noarch	2.9.27-1.el8	epel	17 M
Upgrading:				
chkconfig	x86_64	1.19.1-1.el8	baseos	197 k
platform-python-pip	noarch	9.0.3-20.el8.rocky.0	baseos	1.7 M
Installing dependencies:				
libsodium	x86_64	1.0.18-2.el8	epel	162 k
python3-babel	noarch	2.5.1-7.el8	appstream	4.0 M
python3-bcrypt	x86_64	3.1.6-2.el8.1	epel	44 k
python3-cffi	x86_64	1.11.5-5.el8	baseos	237 k
python3-cryptography	x86_64	3.2.1-5.el8	baseos	558 k
python3-jinja2	noarch	2.10.1-3.el8	appstream	537 k
python3-jmespath	noarch	0.9.0-11.el8	appstream	44 k
python3-markupsafe	x86_64	0.23-19.el8	appstream	38 k
python3-pip	noarch	9.0.3-20.el8.rocky.0	appstream	19 k
python3-pyasn1	noarch	0.3.7-6.el8	appstream	125 k
python3-pycparser	noarch	2.14-14.el8	baseos	100 k
python3-pynacl	x86_64	1.3.0-5.el8	epel	100 k
python3-pytz	noarch	2017.2-9.el8	appstream	53 k
python36	x86_64	3.6.8-38.module+el8.5.0+671+195e4563	appstream	18 k
sshpas	x86_64	1.06-9.el8	epel	27 k
Installing weak dependencies:				
python3-paramiko	noarch	2.4.3-1.el8	epel	289 k
Enabling module streams:				
python36		3.6		
Transaction Summary				
=====				
Install	17 Packages			
Upgrade	2 Packages			
Total download size: 26 M				
Is this ok [y/N]: _				

Figura 3.2: Instalación del paquete de ansible

Una vez hecho esto ya tendremos disponible ansible en nuestro sistema y podemos comprobarlo mediante la ejecución del comando:

```
ansible --version
```

```
[adrianas@localhost ~]$ ansible --version
ansible 2.9.27
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/home/adrianas/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3.6/site-packages/ansible
  executable location = /usr/bin/ansible
  python version = 3.6.8 (default, May 19 2021, 03:00:47) [GCC 8.4.1 20200928 (Red Hat 8.4.1-1)]
[adrianas@localhost ~]$
```

Figura 3.3: Versión de ansible para comprobar que está correctamente instalado



### 3.2. Configuración de Ansible

Antes de configurar nada, tenemos que saber qué máquinas queremos gestionar a través del uso de Ansible. Para ello lo primero que tenemos que conocer es la ip de cada uno ya que nos conectaremos por ssh a cada una de las máquinas.

Las máquinas de las que dispongo son 2:

1. CentOS8 con IP 192.168.56.102
2. Otra CentOS8 pero con IP 192.168.56.110

Lo siguiente que haremos será configurar el servicio ssh, como hemos visto en la Práctica 2, en nuestra máquina con ansible y copiaremos la clave pública ssh a cada una de las dos máquinas.

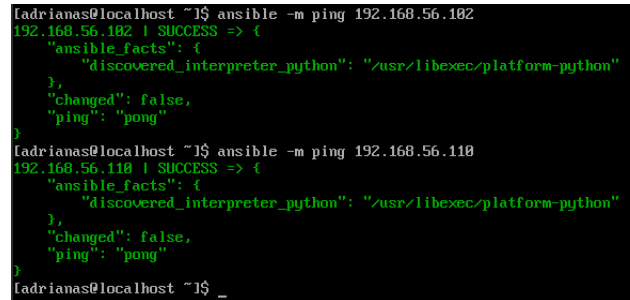
Una vez tengamos configurados los servicios ssh en cada una de las máquinas, editaremos el archivo `/etc/ansible/hosts` introduciendo al principio del archivo las IP de los dos servidores que queremos gestionar. Si quisiéramos gestionar servidores de apache, tendríamos que meter sus direcciones IP debajo de la etiqueta `[webservers]`.

```
#
# It should live in /etc/ansible/hosts
#
# - Comments begin with the '#' character
# - Blank lines are ignored
# - Groups of hosts are delimited by [header] elements
# - You can enter hostnames or ip addresses
# - A hostname/ip can be a member of multiple groups
#
# Ex 1: Ungrouped hosts, specify before any group headers.
## green.example.com
## blue.example.com
## 192.168.100.1
## 192.168.100.10
192.168.56.102
192.168.56.110
#
# Ex 2: A collection of hosts belonging to the 'webservers' group
## [webservers]
## alpha.example.org
## beta.example.org
## 192.168.1.100
## 192.168.1.110
#
# If you have multiple hosts following a pattern you can specify
# them like this:
## www[001:006].example.com
#
# Ex 3: A collection of database servers in the 'dbservers' group
## [dbservers]
"/etc/ansible/hosts" 47L, 1047C written
ladrianas@localhost ~1$ _
```

Figura 3.4: Configuración del archivo `/etc/ansible/hosts`. Añado las dos IP de las dos máquinas CentOS.

Ahora que ya tenemos todas las configuraciones correctas, es hora de probar a hacer un ping mediante ansible a cada una de las máquinas o a todas a la vez con las siguientes órdenes:

```
ansible -m ping 192.168.56.102
ansible -m ping 192.168.56.110
ansible -m ping all
```



```
[adrianas@localhost ~]$ ansible -m ping 192.168.56.102
192.168.56.102 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/libexec/platform-python"
  },
  "changed": false,
  "ping": "pong"
}
[adrianas@localhost ~]$ ansible -m ping 192.168.56.110
192.168.56.110 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/libexec/platform-python"
  },
  "changed": false,
  "ping": "pong"
}
[adrianas@localhost ~]$ _
```

Figura 3.5: Prueba de ping a ambas máquinas CentOS

La opción `-m` indica el uso de un módulo específico, en este caso `ping`. Para poder ejecutar comandos de shell tendríamos que hacer lo mismo pero en vez de usar la orden `ping`, tendríamos que usar `shell -a '<comando-de-shell>'` como aparece en la siguiente imagen. Antes de probar algún comando de shell, en la máquina en la que vayamos a ejecutarlo tenemos que ejecutar el siguiente comando para que no pida la contraseña de superusuario:

```
echo "$(whoami) ALL=(ALL) NOPASSWD:ALL" | sudo tee /etc/sudoers.d/$(whoami)
```

A continuación muestro un ejemplo de la orden:

```
ansible -b --become-method=sudo \
-m shell -a 'dnf update' 192.168.56.102
```



```
[adrianas@localhost ~]$ ansible -b --become-method=sudo -m shell -a 'dnf update' 192.168.56.102
```

Figura 3.6: Prueba de ejecución de un comando shell

Para comprobar que realmente funciona:

```
[adrianas@localhost ~]# echo "$(whoami) ALL=(ALL) NOPASSWD:ALL" | sudo tee /etc/sudoers.d/$(whoami)
[sudo] password for adrianas:
adrianas ALL=(ALL) NOPASSWD:ALL
[adrianas@localhost ~]# ps
  PID TTY          TIME CMD
 1394 tty1      00:00:00 bash
 2018 tty1      00:00:00 ps
[adrianas@localhost ~]# ps -a
  PID TTY          TIME CMD
 2013 pts/0      00:00:00 sudo
 2015 pts/0      00:00:00 sh
 2016 pts/0      00:00:00 platform-python
 2017 pts/0      00:00:00 dnf
 2019 tty1      00:00:00 ps
[adrianas@localhost ~]# ps -a
  PID TTY          TIME CMD
 2013 pts/0      00:00:00 sudo
 2015 pts/0      00:00:00 sh
 2016 pts/0      00:00:00 platform-python
 2017 pts/0      00:00:00 dnf
 2020 tty1      00:00:00 ps
[adrianas@localhost ~]# ps -a
  PID TTY          TIME CMD
 2021 tty1      00:00:00 ps
[adrianas@localhost ~]# ps -a
  PID TTY          TIME CMD
 2022 tty1      00:00:00 ps
[adrianas@localhost ~]# _
```

Figura 3.7: Se puede observar que se esta ejecutando dnf y las dependencias de ansible de forma remota, hasta que hacemos control+c.

## Referencias