

# Memoria de la Práctica 3

## Ingeniería de Servidores

### Pablo Olivares

Antes de comenzar con la resolución del ejercicio, aclararé el estado inicial de las máquinas. Trabajaremos sobre dos servidores Linux, Ubuntu Server 20.04 y CentOS 8, ambos activos en una red local con las IPs 192.168.56.105 y 192.168.56.110 respectivamente. Tendremos instalados y activos los servicios de Apache y MySQL. Por defecto tendremos configurado los servidores SSH con el puerto 22022 abierto para dicho servicio. Además, se han realizado las configuraciones previas en CentOS de SELinux y el cortafuegos para permitir dicha conexión SSH. De la misma forma haremos con HTTP en su puerto por defecto. Dicho esto, procedo con la resolución.

### Ejercicio 1.

*Realice una instalación de Zabbix 5.0 en su servidor con Ubuntu Server 20.04 y configure para que se monitoree a él mismo y para que monitorice a la máquina con CentOS. Puede configurar varios parámetros para monitorizar, uso de CPU, memoria, etc. pero debe configurar de manera obligatoria la monitorización de los servicios SSH y HTTP. Documente el proceso de instalación y configuración indicando las referencias que ha utilizado así como los problemas que ha encontrado. Para ello puede usar cualquier tipo de formato de documento (respetando claridad y corrección) y procure que en las capturas parezca su nombre de usuario (en el prompt p.ej.). El archivo debe estar subido a SWAD (zona mis trabajos) antes del examen de esta práctica.*

### Paso 1. Instalación del Servidor Zabbix en Ubuntu Server

Comenzaremos con la instalación de Zabbix 5.0 en Ubuntu Server. Para ello, accederemos a su página web en la sección de instalación. Dicha página es [Download and install Zabbix](#). Seleccionamos las opciones 5.0 LTS, Ubuntu, 20.04 (Focal) y Apache y seguimos todos los pasos de instalación.

[PRODUCT](#)
[SOLUTIONS](#)
[SUPPORT & SERVICES](#)
[TRAINING](#)
[PARTNERS](#)
[COMMUNITY](#)
[ABOUT US](#)

[DOWNLOAD](#)

[Home](#) / [Product](#) /

# Download and install Zabbix

Zabbix Packages

Zabbix Cloud Images

Zabbix Containers

Zabbix Appliance

Zabbix Sources

Zabbix Agents

## 1 Choose your platform

ZABBIX VERSION	OS DISTRIBUTION	OS VERSION	DATABASE	WEB SERVER
6.0 LTS	Red Hat Enterprise Linux	20.04 (Focal)	MySQL	Apache
5.0 LTS	CentOS	18.04 (Bionic)	PostgreSQL	NGINX
4.0 LTS	Oracle Linux	16.04 (Xenial)		
6.2 PRE-RELEASE	Ubuntu	14.04 (Trusty)		
	Debian			
	SUSE Linux Enterprise Server			
	Raspberry Pi OS			
	Ubuntu (arm64)			

Concretamente, estos son los comandos a introducir (todo en usuario root):

```
wget https://repo.zabbix.com/zabbix/5.0/ubuntu/pool/main/z/zabbix-release/zabbix-release_5.0-1+focal_all.deb
dpkg -i zabbix-release_5.0-1+focal_all.deb
apt update

apt install zabbix-server-mysql zabbix-frontend-php zabbix-apache-conf zabbix-agent
mysql -uroot -p
practicas,ise
```

Una vez en mysql, creamos la base de datos que usará Zabbix para almacenar la monitorización:

```
create database zabbix character set utf8 collate utf8_bin;
create user zabbix@localhost identified by 'practicas,ise';
grant all privileges on zabbix.* to zabbix@localhost;
quit;
```

A continuación, importamos los datos iniciales del host en Zabbix:

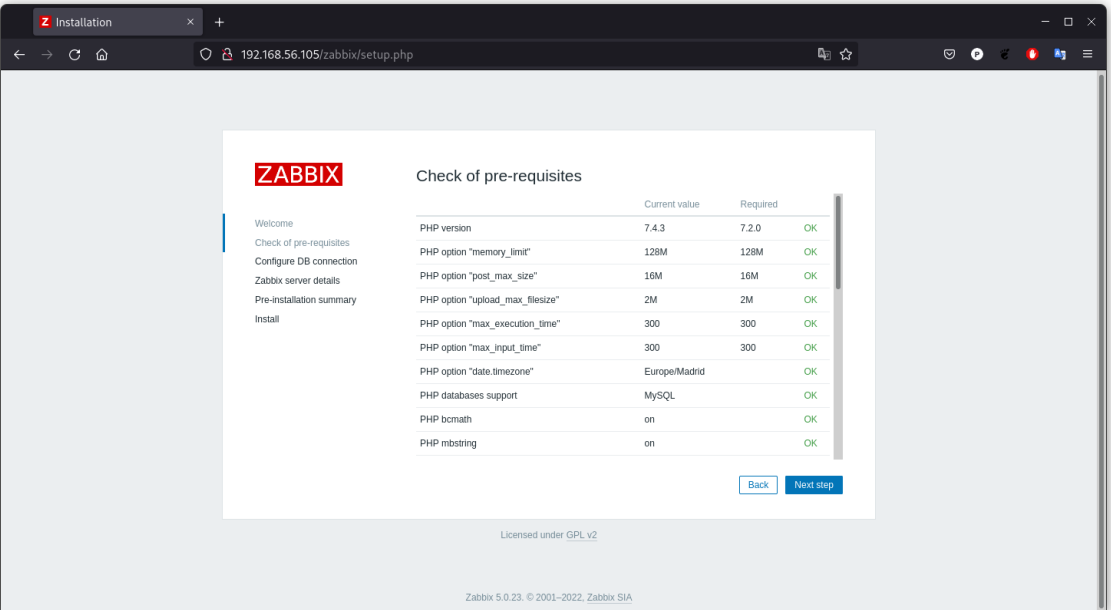
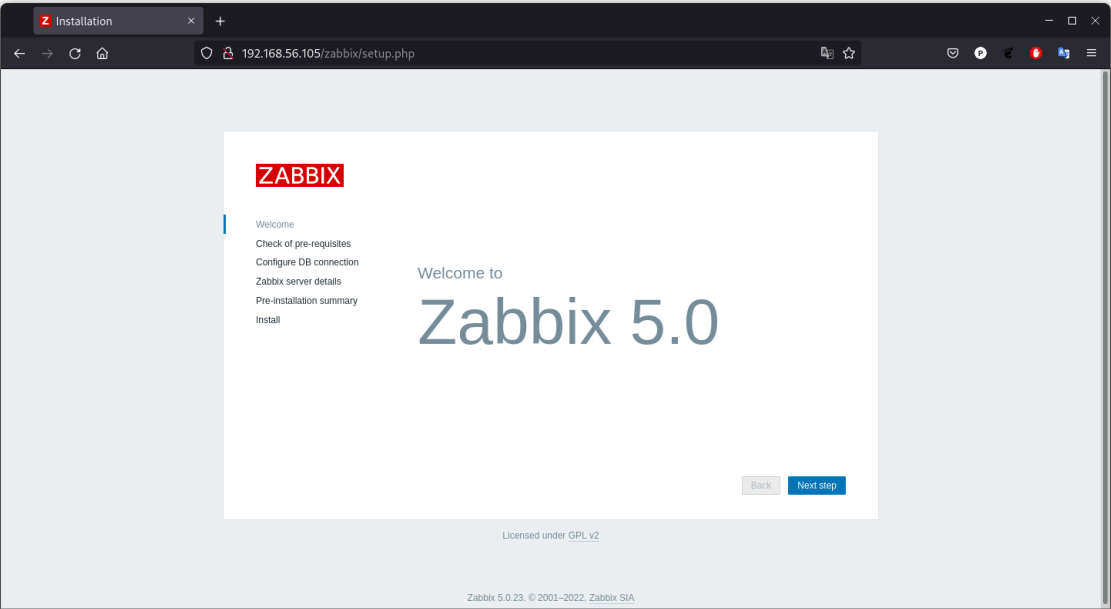
```
zcat /usr/share/doc/zabbix-server-mysql*/create.sql.gz | mysql -uzabbix -p zabbix
```

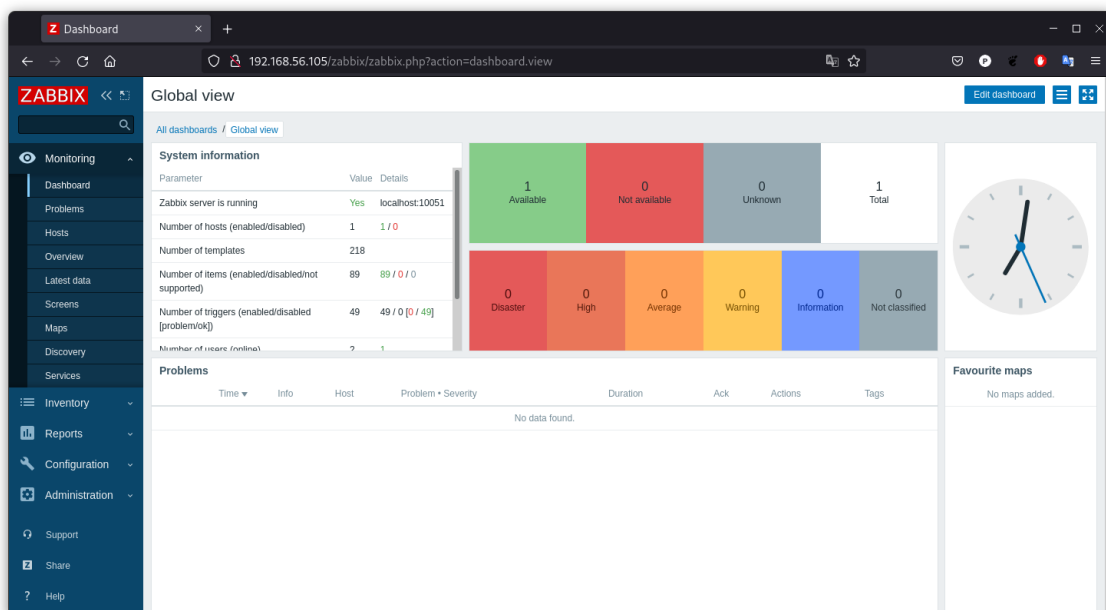
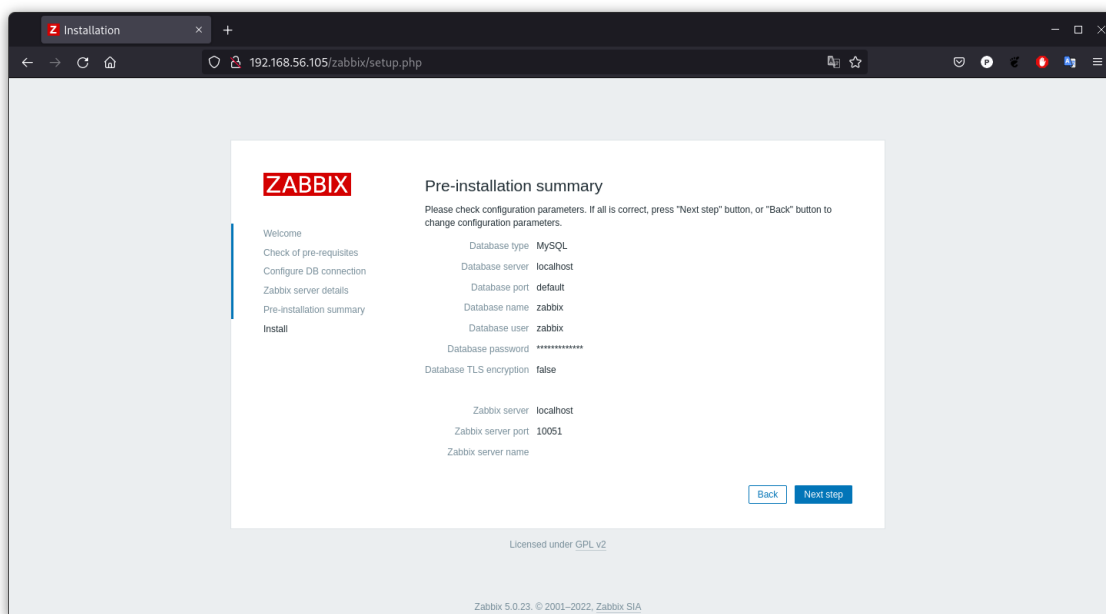
Finalmente, ponemos la contraseña establecida en el servidor de mysql en el archivo de configuración del servidor Zabbix, esto es, `DBPassword=practicais` en `/etc/zabbix/zabbix_server.conf` y modificamos la región en `/etc/zabbix/apache.conf` tal que `php_value date.timezone Europe/Madrid`.

Reiniciamos los servicios:

```
systemctl restart zabbix-server zabbix-agent apache2  
systemctl enable zabbix-server zabbix-agent apache2
```

Una vez realizada la instalación, accedemos al frontend de Zabbix desde nuestra máquina anfitrión a través de `http://192.168.56.105/zabbix` para terminar la configuración inicial.





En principio hemos dejado todo por defecto. Iniciamos sesión con el usuario por defecto, Admin con contraseña zabbix . Como se puede ver, Zabbix está corriendo y funcionando correctamente. Ahora procederemos a instalar el agente de Zabbix en CentOS para gestionarlo desde nuestro servidor principal, es decir, Ubuntu Server. Para ello, accederemos a su web de instalación, [Download Zabbix agents](#), y seguiremos todos los pasos.

## Paso 2. Instalación del Agente Zabbix en CentOS

Lo primero que haremos será desactivar SELinux para evitar problemas durante la configuración de CentOS. Para ello accedemos al fichero `/etc/selinux/config` y ponemos `SELINUX=disabled` . A continuación, en usuario root, instalamos el agente:

```
rpm -Uvh https://repo.zabbix.com/zabbix/5.0/rhel/8/x86_64/zabbix-release-5.0-1.el8.noarch.rpm
dnf clean all
dnf install zabbix-agent
systemctl enable --now zabbix-agent
```

Una vez instalado y activo, configuramos el cliente para que se conecte con nuestro servidor Zabbix en Ubuntu Server. En el archivo `/etc/zabbix/zabbix_agentd.conf` establecemos las siguientes líneas del fichero con los datos correspondientes a nuestro servidor:

```
Server=192.168.56.105
ServerActive=192.168.56.105
Hostname=CentOS
```

Es decir, hemos establecido el servidor Ubuntu tanto para las comprobaciones activas como las pasivas (más información en [Zabbix Agent: Active vs. Passive - Zabbix Blog](#)) y le hemos puesto de nombre a nuestro agente Zabbix CentOS, para así identificar más fácilmente nuestra máquina. Tal vez no estaría de más añadir al localhost en la lista de comprobaciones, así en caso de que haya algún problema con nuestro Servidor Zabbix, permitir las consultas desde la propia máquina. Tan solo había que añadir `127.0.0.1` de la siguiente forma:

```
Server=192.168.56.105, 127.0.0.1
ServerActive=192.168.56.105, 127.0.0.1
```

Sin embargo, como la práctica no lo pide, obviaremos este último paso. Finalmente, configuraremos el cortafuegos para abrir el puerto utilizado por Zabbix y permitir la conexión. Si quisiéramos reactivar SELinux, tendríamos que añadir el puerto a la lista blanca de SELinux para que permitiese el tráfico a través de él.

```
firewall-cmd --permanent --add-port=10050/tcp
firewall-cmd --reload
```

Por último reiniciamos el agente:

```
systemctl enable zabbix-agent
systemctl restart zabbix-agent
```

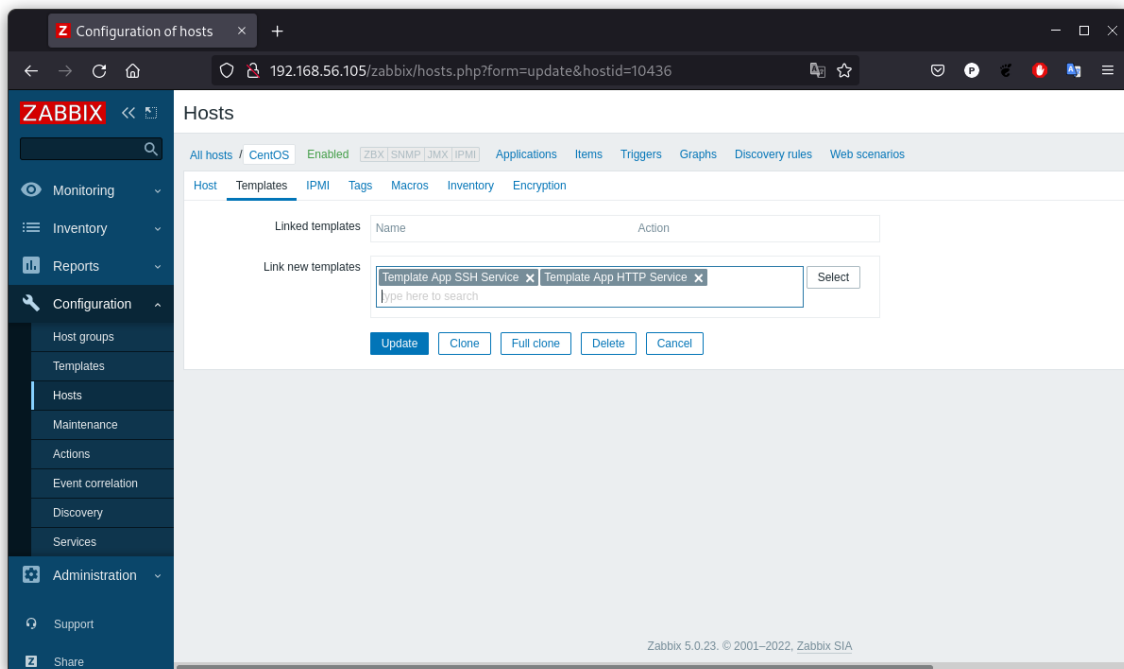
### Paso 3. Creación de Hosts en el Servidor Zabbix

La siguiente tarea a realizar será crear un host para CentOS y otro para Ubuntu Server. Si nos fijamos un poco en la primera captura del frontend de Zabbix, en el *Dashboard*, veremos que pone que tenemos 1 host activo. Esto se debe a que al instalar Zabbix en nuestro servidor, éste vino con él un host por defecto que monitoriza muchos aspectos distintos de nuestro sistema. Sin embargo, a nosotros tan sólo nos interesa monitorizar HTTP y SSH, por eso, a parte de crear un host para CentOS, crearemos otro para Ubuntu Server.

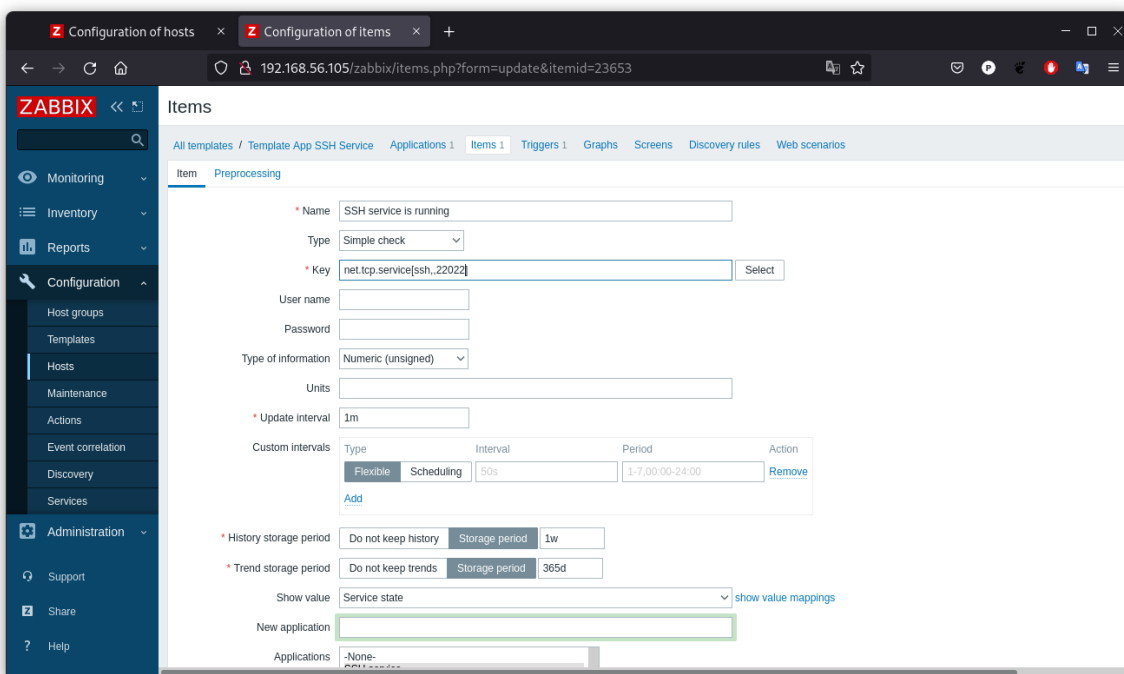
Para ello, accederemos al frontend de Zabbix y en el apartado de **Configuration** -> **Hosts**, le daremos a *Create host*. Una vez ahí, rellenamos los campos como veremos en la siguiente imagen. También es obligatorio establecer el grupo, no obstante, la práctica no especifica el grupo al que debemos añadirlo, así por defecto pondré el grupo *Linux Server*.

The screenshot shows the Zabbix web interface for creating a new host. The browser address bar shows the URL `192.168.56.105/zabbix/hosts.php?form=create`. The left sidebar contains the Zabbix logo and a navigation menu with categories: Monitoring, Inventory, Reports, Configuration, Administration, Support, and Share. The 'Configuration' menu is expanded, showing sub-items like Host groups, Templates, Hosts, Maintenance, Actions, Event correlation, Discovery, and Services. The 'Hosts' sub-item is selected. The main content area is titled 'Hosts' and has tabs for Host, Templates, IPMI, Tags, Macros, Inventory, and Encryption. The 'Host' tab is active, displaying a form with the following fields: 'Host name' (CentOS), 'Visible name' (CentOS), 'Groups' (a search box with a 'Select' button), 'Interfaces' (a table with one row: Agent, 192.168.56.110, empty DNS name, IP, 10050, and a 'Remove' button), 'Description' (a text area containing 'Agente de CentOS'), 'Monitored by proxy' (a dropdown menu set to '(no proxy)'), and 'Enabled' (a checked checkbox). At the bottom of the form are 'Add' and 'Cancel' buttons.

Ahora en *Templates*, pondremos que el host de Zabbix deba monitorizar SSH y HTTP, estableciendo las plantillas predeterminadas para ello.

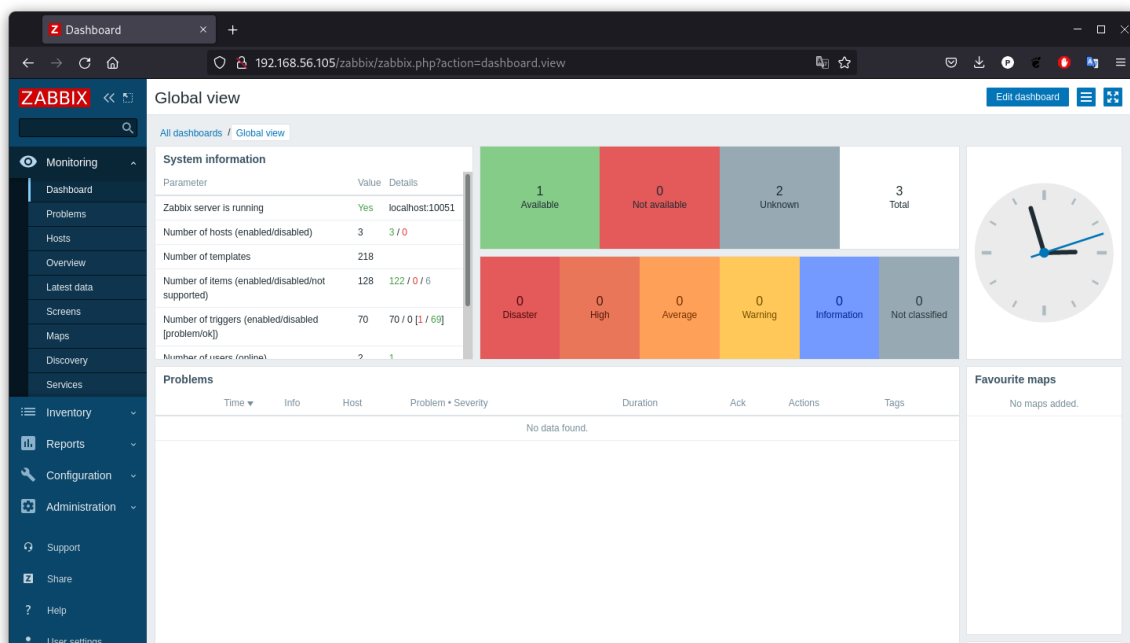
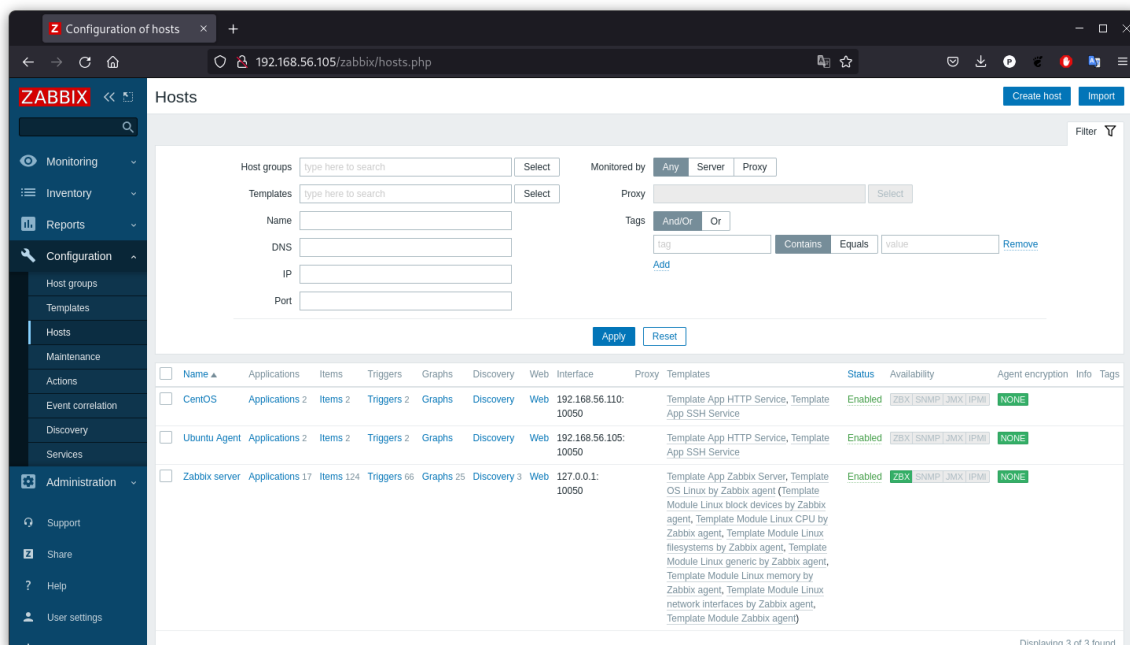


Como tenemos configurado el servicio SSH en el puerto 22022 en ambas máquinas, debemos modificar dichas plantillas. Para ello, accedemos al **Hosts -> CentOS -> Items** y desde ahí, pinchamos sobre *SSH service is running* para acceder al ítem asociado a SSH. Desde ahí, modificamos **Key** para que tenga en cuenta el puerto 22022 en vez del puerto por defecto y guardamos los cambios.



Por último, seguimos los mismos pasos para crear otro host para Ubuntu Server. Quedaría de esta forma.



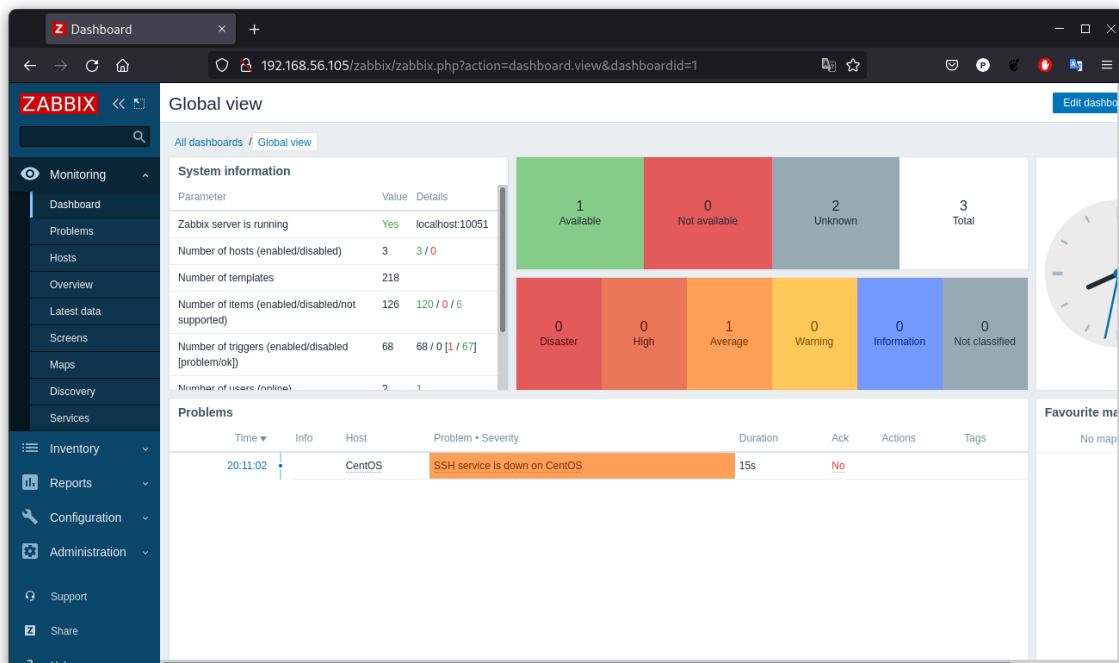


## Paso 4. Prueba de funcionamiento de Zabbix

Ya como parte final de la práctica, comprobaremos que Zabbix funciona correctamente. Para dicho fin, desconectaremos el servidor SSH en CentOS y veamos qué sucede.

```
CentOS (VNet) [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
[root@localhost pabloom]# systemctl stop sshd
[root@localhost pabloom]# systemctl status sshd
• sshd.service - OpenSSH server daemon
  Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)
  Active: inactive (dead) since Fri 2022-05-06 14:00:23 EDT; 10s ago
    Docs: man:sshd(8)
          man:sshd_config(5)
  Process: 855 ExecStart=/usr/sbin/sshd -D $OPTIONS $CRYPTO_POLICY (code=exited, status=0/SUCCESS)
 Main PID: 855 (code=exited, status=0/SUCCESS)

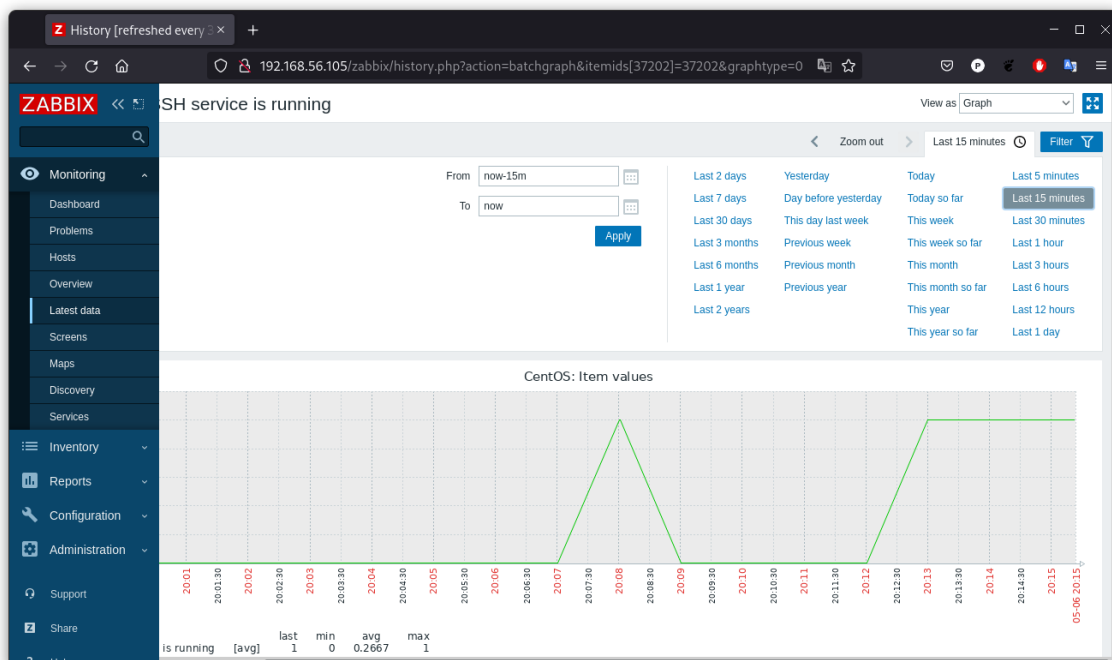
may 06 13:40:42 localhost.localdomain systemd[1]: Starting OpenSSH server daemon...
may 06 13:40:42 localhost.localdomain sshd[855]: Server listening on 0.0.0.0 port 22022.
may 06 13:40:42 localhost.localdomain sshd[855]: Server listening on :: port 22022.
may 06 13:40:42 localhost.localdomain systemd[1]: Started OpenSSH server daemon.
may 06 14:00:23 localhost.localdomain systemd[1]: Stopping OpenSSH server daemon...
may 06 14:00:23 localhost.localdomain systemd[1]: Stopped OpenSSH server daemon.
[root@localhost pabloom]# _
```



Como vemos, Zabbix nos notifica que el servidor SSH en CentOS está caído. Ahora, si reactivamos el servicio, vemos que nos notifica que el problema se ha resuelto. Podemos ver un gráfico del estado de actividad en **Monitoring -> Latest Data** seleccionando el host y servicio del que queremos ver la información y abajo del todo, seleccionando *Display graph*.

```
CentOS (VNet) [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
[root@localhost pabloom]# systemctl stop sshd
[root@localhost pabloom]# systemctl start sshd
[root@localhost pabloom]# systemctl status sshd
• sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2022-05-06 14:12:03 EDT; 3s ago
     Docs: man:sshd(8)
           man:sshd_config(5)
   Main PID: 2046 (sshd)
     Tasks: 1 (limit: 5019)
    Memory: 1.1M
   CGroup: /system.slice/sshd.service
           └─2046 /usr/sbin/sshd -D -oCiphers=aes256-gcm@openssh.com,chacha20-poly1305@openssh.com,

may 06 14:12:03 localhost.localdomain systemd[1]: Starting OpenSSH server daemon...
may 06 14:12:03 localhost.localdomain sshd[2046]: Server listening on 0.0.0.0 port 22022.
may 06 14:12:03 localhost.localdomain sshd[2046]: Server listening on :: port 22022.
may 06 14:12:03 localhost.localdomain systemd[1]: Started OpenSSH server daemon.
lines 1-15/15 (END)
```



La comprobación de HTTP y en Ubuntu es análoga a la realizada en CentOS para SSH.

## Ejercicio 2.

*Usted deberá saber cómo instalar y configurar Ansible para poder hacer un ping a las máquinas virtuales de los servidores y ejecutar un comando básico (p.ej. el script de monitorización del RAID1). También debe ser consciente de la posibilidad de escribir acciones más complejas mediante playbooks escritos con YAML. Incluya capturas de pantalla del proceso con una breve descripción en el mismo documento que suba para el ejercicio de Zabbix.*

## Paso 1. Configuración previa de Ansible

Este ejercicio nos pide que configuremos Ansible para realizar una automatización de la monitorización de los servidores de nuestras máquinas virtuales. Para la realización de dicho proceso, comenzaremos instalando Ansible en nuestro anfitrión (uso `dnf` porque tengo Fedora).

```
sudo dnf install ansible
```

Una vez instalado, accederemos al archivo de configuración de los hosts `/etc/ansible/hosts` y añadiremos un grupo con las IPs de nuestras máquinas virtuales.

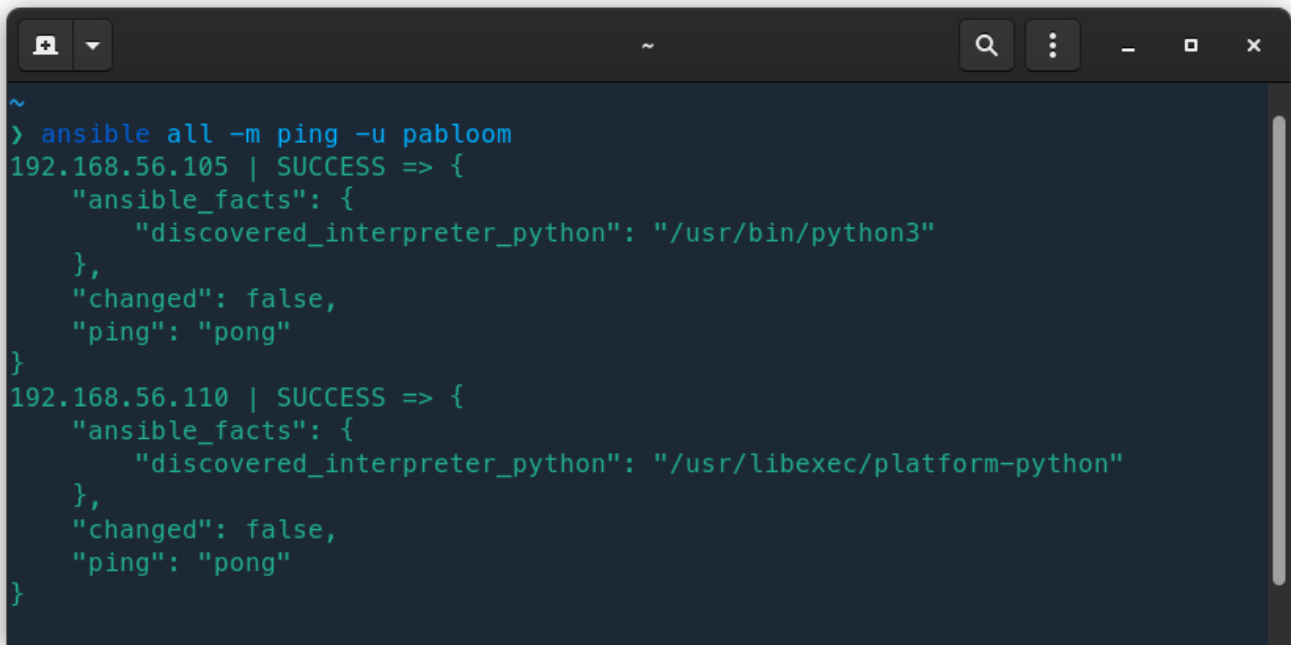
```
[ansibleise]
192.168.56.110
192.168.56.105
```

Antes de continuar, conviene que establezcamos el acceso sin contraseña por SSH a nuestros servidores. De esta forma, podremos automatizar de forma real la monitorización, ya que si cada vez que se ejecutase tuviésemos que introducir la contraseña de nuevo, la automatización no sería productiva.

```
ssh-copy-id -p 22022 pabloom@192.168.56.105
ssh-copy-id -p 22022 pabloom@192.168.56.110
```

Y en los archivos de configuración SSH de cada máquina, haber activado el acceso sin contraseña. Esto es, en `/etc/ssh/sshd_config` poner `PubkeyAuthentication yes` y ya de paso, por seguridad `PermitRootLogin no`.

Una vez realizados estos pasos previos, comprobamos que tenemos conexión SSH a través de Ansible con las máquinas virtuales y debe devolver esto.

A terminal window with a dark background and light-colored text. The window has standard Linux window controls at the top. The text shows the execution of an Ansible command to ping two hosts. The output for the first host (192.168.56.105) shows a successful ping with the response 'pong' and the Python interpreter path '/usr/bin/python3'. The output for the second host (192.168.56.110) also shows a successful ping with the response 'pong' but with a different Python interpreter path '/usr/libexec/platform-python'.

```
~
> ansible all -m ping -u pabloom
192.168.56.105 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
192.168.56.110 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/libexec/platform-python"
  },
  "changed": false,
  "ping": "pong"
}
```

Aquí habría que tener en cuenta dos cosas que podrían dar error: una es que no hemos dicho a Ansible en que puerto está configurado el SSH. Para resolverlo, podemos hacerlo de dos formas:

- La primera, sería accediendo al fichero `/etc/ansible/ansible.cfg` y en `remote_port`, indicar que se trata del puerto 22022.
- La segunda, pensada a que tengamos distintos puertos configurados en cada máquina, podemos indicarlo en el fichero `/etc/ansible/hosts` junto a las IPs que hemos declarado de la siguiente forma:

```
[ansibleise]
192.168.56.110 ansible_port=22022
192.168.56.105 ansible_port=22022
```

El segundo error podría darse que una de las dos máquinas no funcionase el ping. Esto sería probablemente por no haber configurado el acceso sin contraseña en dicha máquina previamente.

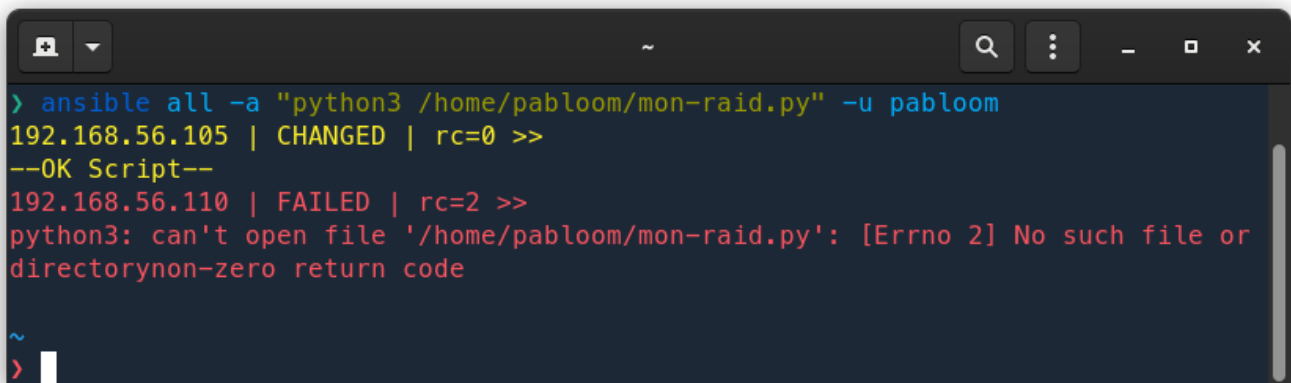
## Paso 2. Automatización de monitorización del RAID

Comencemos ahora con el proceso de monitorización. Para ello, crearemos un script de python que va a monitorizar el estado del RAID realizado en la práctica 1. El script es el siguiente:

```
import re
f = open('/proc/mdstat')
for line in f:
    b = re.findall('\[[U]*[_]+[U]*\]', line)
    if(b!=[]):
        print("--ERROR en RAID--")
print("--OK Script--")
```

A este script le pondremos el nombre que queramos, en mi caso le llamaré `mon-raid.py`. En mi caso, lo crearé en la máquina de Ubuntu Server en `/home/pabloom/mon-raid.py` y lo copiaré a través de SSH a la máquina CentOS.

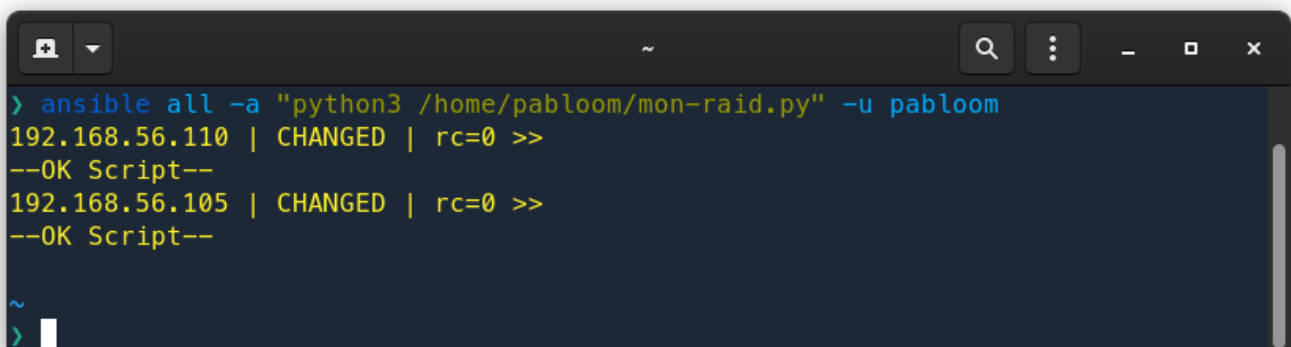
Para ejecutar el servicio a través de Ansible, escribiremos el siguiente comando: `ansible all -a "python3 /home/pabloom/mon-raid.py" -u pabloom`. Si lo ejecutásemos antes de copiarlo a CentOS, nos saldría el siguiente error:



```
> ansible all -a "python3 /home/pabloom/mon-raid.py" -u pabloom
192.168.56.105 | CHANGED | rc=0 >>
--OK Script--
192.168.56.110 | FAILED | rc=2 >>
python3: can't open file '/home/pabloom/mon-raid.py': [Errno 2] No such file or
directorynon-zero return code

~
>
```

Por tanto, lo copiamos desde ubuntu con el siguiente comando: `scp -P 22022 /home/pabloom/mon-raid.py pabloom@192.168.56.110:/home/pabloom/mon-raid.py`. Volveríamos a ejecutar el comando y listo.



```
> ansible all -a "python3 /home/pabloom/mon-raid.py" -u pabloom
192.168.56.110 | CHANGED | rc=0 >>
--OK Script--
192.168.56.105 | CHANGED | rc=0 >>
--OK Script--

~
>
```