

Ingeniería de Servidores (2021-2022)
GRADO EN INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE GRANADA

Memoria Práctica 3

José María Ramírez González

28 de diciembre de 2021

Índice

1	Instalación y monitorización con Zabbix	3
1.1	Instalación de Zabbix en Ubuntu Server	3
1.2	Configuración de la base de datos	4
1.3	Configuración de PHP para el frontend de Zabbix	6
1.4	Configuración del frontend de Zabbix	6
1.5	Configuración del host de CentOS	8
1.6	Configuración de monitorización	10
2	Instalación y uso de Ansible	13
2.1	Instalación de Ansible en Ubuntu Server	13
2.2	Configuración de Ansible	14
2.3	Creando un inventario de Ansible	14
2.4	Ejecución de comandos y creación de <i>Playbooks</i>	15
2.5	Automatización de tareas con <i>Playbooks</i>	16
3	Conclusión	17

Índice de figuras

1.1	Resultado de ejecutar las órdenes mencionadas	3
1.2	Instalación de dependencias para el correcto funcionamiento de Zabbix	4
1.3	Inicialización de la base de datos	5
1.4	Creación del usuario y la tabla en la BD	5
1.5	Aporctación de la contraseña de la BD a Zabbix	6
1.6	Establecimiento de la zona horaria	6
1.7	Primer acceso al frontend	6
1.8	Configuración desde el frontend	7
1.9	Página de inicio de Zabbix, sin configuración previa	8
1.10	Configuración de SELinux	8
1.11	Servicio del agente de Zabbix en CentOS	9
1.12	Configuración de parámetros de agente de Zabbix en CentOS	9
1.13	Creación del host de CentOS en la web de Zabbix	10
1.14	Configuración de los items para el agente de CentOS	11
1.15	Visualización de las primeras monitorizaciones	11
1.16	Gráficas generadas a partir de las monitorizaciones	12
2.1	Instalación de Python y paquete auxiliar	13
2.2	Comprobación de la instalación de ansible	14
2.3	Comprobación de la instalación de ansible	15
2.4	Ping de prueba simple con ansible	15
2.5	Script básico para borrar el contenido de la carpeta temporal	16
2.6	Ejecución de Playbook sencillo	16

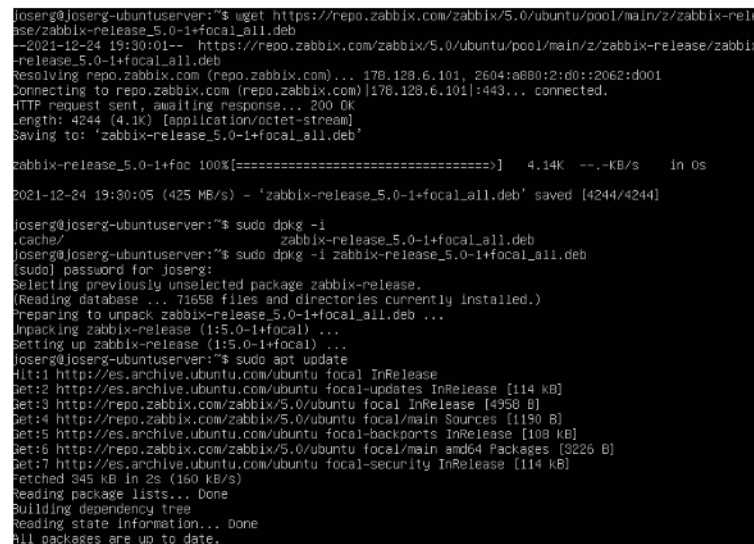
1. Instalación y monitorización con Zabbix

En este primer apartado, vamos a ver paso a paso cómo instalar, configurar y utilizar Zabbix sobre *Ubuntu Server* para monitorizar varios aspectos de (en nuestro caso) una máquina virtual con *CentOs8* y de sí mismo.

1.1. Instalación de Zabbix en Ubuntu Server

Para realizar la instalación de Zabbix, vamos a remitirnos a su documentación y vamos a ir siguiendo los pasos que allí se indican. Comenzaremos por descargarnos el paquete de zabbix para nuestra versión de Ubuntu server e instalar los repositorios, ejecutando los siguientes comandos y obteniendo el resultado que vemos en la figura 1.1

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



```
joserg@joserg-ubuntu-server:~$ wget https://repo.zabbix.com/zabbix/5.0/ubuntu/pool/main/z/zabbix-release/zabbix-release_5.0-1+focal_all.deb
--2021-12-24 19:30:01-- https://repo.zabbix.com/zabbix/5.0/ubuntu/pool/main/z/zabbix-release/zabbix-release_5.0-1+focal_all.deb
Resolving repo.zabbix.com (repo.zabbix.com)... 178.128.6.101, 2604:a880:2:d0::2062:d001
Connecting to repo.zabbix.com (repo.zabbix.com)|178.128.6.101|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4244 (4.1K) [application/octet-stream]
Saving to: 'zabbix-release_5.0-1+focal_all.deb'

zabbix-release_5.0-1+foc 100%[=====] 4.14K --.-KB/s in 0s

2021-12-24 19:30:05 (425 MB/s) - 'zabbix-release_5.0-1+focal_all.deb' saved [4244/4244]

joserg@joserg-ubuntu-server:~$ sudo dpkg -i
./cache/ zabbix-release_5.0-1+focal_all.deb
joserg@joserg-ubuntu-server:~$ sudo dpkg -i zabbix-release_5.0-1+focal_all.deb
[sudo] password for joserg:
Selecting previously unselected package zabbix-release.
(Reading database ... 71658 files and directories currently installed.)
Preparing to unpack zabbix-release_5.0-1+focal_all.deb ...
Unpacking zabbix-release (1:5.0-1+focal) ...
Setting up zabbix-release (1:5.0-1+focal) ...
joserg@joserg-ubuntu-server:~$ sudo apt update
Hit:1 http://es.archive.ubuntu.com/ubuntu focal InRelease
Get:2 http://es.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:3 http://repo.zabbix.com/zabbix/5.0/ubuntu focal InRelease [4958 B]
Get:4 http://repo.zabbix.com/zabbix/5.0/ubuntu focal/main Sources [1190 B]
Get:5 http://es.archive.ubuntu.com/ubuntu focal-backports InRelease [108 kB]
Get:6 http://repo.zabbix.com/zabbix/5.0/ubuntu focal/main amd64 Packages [3226 B]
Get:7 http://es.archive.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Fetched 345 kB in 2s (160 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
All packages are up to date.
```

Figura 1.1: Resultado de ejecutar las órdenes mencionadas

Ahora bien, para poder ejecutar zabbix correctamente, necesitamos software que no tenemos instalado por defecto. En concreto, necesitamos una pila LAMP. En nuestro caso, vamos a optar por prescindir de MySQL y hacer uso de MariaDB.

En cualquier caso, tendremos que instalar los paquetes que aparecen en la figura 1.2.

```
josemg@josemg-ubuntu-server:~$ sudo apt install mariadb-client apache2 php php-gd libxslt2 php-mysql
[sudo] password for josemg:
Reading package lists... Done
Building dependency tree
Reading state information... Done
libxslt2 is already the newest version (2.9.10+dfsg-Subuntu0.20.04.1).
libxslt2 set to manually installed.
The following additional packages will be installed:
  apache2-bin apache2-data apache2-utils fontconfig-config fonts-dejavu-core libapache2-mod-php7.4
  libapr1 libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap libconfig-inifiles-perl
  libdbd-mysql-perl libdbi-perl libfontconfig1 libgd3 libjansson4 libjpeg8 libjpeg-turbo8 libjpeg9
  liblua5.2-0 libmysqlclient21 libsnappy1v5 libterm-readkey-perl libtiff5 libwebp6 libxpm4
  mariadb-client-10.3 mariadb-client-core-10.3 mariadb-common mysql-common php-common php7.4
  php7.4-cli php7.4-common php7.4-gd php7.4-json php7.4-mysql php7.4-opcache php7.4-readline
  ssl-cert
Suggested packages:
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom www-browser php-pear libclone-perl
  libmldbm-perl libnet-daemon-perl libsql-statement-perl libgd-tools openssl-blacklist
The following NEW packages will be installed:
  apache2 apache2-bin apache2-data apache2-utils fontconfig-config fonts-dejavu-core
  libapache2-mod-php7.4 libapr1 libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap
  libconfig-inifiles-perl libdbd-mysql-perl libdbi-perl libfontconfig1 libgd3 libjansson4 libjpeg8
  libjpeg-turbo8 libjpeg9 liblua5.2-0 libmysqlclient21 libsnappy1v5 libterm-readkey-perl libtiff5
  libwebp6 libxpm4 mariadb-client mariadb-client-core-10.3 mariadb-client-common
  mariadb-common mysql-common php php-common php-gd php-mysql php7.4 php7.4-cli php7.4-common php7.4-gd
  php7.4-json php7.4-mysql php7.4-opcache php7.4-readline ssl-cert
```

Figura 1.2: Instalación de dependencias para el correcto funcionamiento de Zabbix

Acto seguido, vamos a instalar el resto de recursos de zabbix:

```
$ sudo apt install zabbix-server-mysql zabbix-frontend-php
zabbix-apache-conf zabbix-agent
```

1.2. Configuración de la base de datos

Ahora toca configurar la base de datos¹, para ello vamos a seguir el tutorial al pie de la letra, obviando la seguridad y usando las contraseñas genéricas que nos proporcionan, ya que esto no va a estar realmente expuesto a internet.

Lo primero es crear las tablas del sistema para que comience a funcionar (recordemos que hay que tener instalado, en nuestro caso, mariadb-server). Para ello, usamos el siguiente comando:

```
$ sudo mysql_install_db
```

Así inicializamos la base de datos con las tablas que le permitirán llevar los permisos y registros pertinentes de las demás tablas y usuarios, luego iniciamos el servicio y realizamos una instalación segura, creando el usuario root 1.3.

¹Nos vamos a referir a la BD como MariaDB o MySQL indistintivamente a lo largo de todo el apartado, ya que muchos comandos de mariadb usan mysql en la sintaxis.

```
joserg@joserg-ubuntu:~$ sudo mysql_secure_installation
NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB
SERVERS IN PRODUCTION USE! PLEASE READ EACH STEP CAREFULLY!

In order to log into MariaDB to secure it, we'll need the current
password for the root user. If you've just installed MariaDB, and
you haven't set the root password yet, the password will be blank,
so you should just press enter here.

Enter current password for root (enter for none):
OK, successfully used password, moving on...

Setting the root password ensures that nobody can log into the MariaDB
root user without the proper authorisation.

Set root password? [Y/n] Y
New password:
Re-enter new password:
Password updated successfully!
Reloading privilege tables..
... Success!

joserg@joserg-ubuntu:~$ sudo systemctl enable mariadb
joserg@joserg-ubuntu:~$ sudo systemctl start mariadb
```

(a) Activamos el servicio de mariadb

(b) Uso de mysql_secure_installation

Figura 1.3: Inicialización de la base de datos

Ahora creamos el usuario y la tabla para zabbix en la BD como se muestra en 1.4.

```
joserg@joserg-ubuntu:~$ sudo mysql -uroot -p
[sudo] password for joserg:
Enter password:
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 67
Server version: 10.3.32-MariaDB-0ubuntu0.20.04.1 Ubuntu 20.04

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

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

MariaDB [(none)]> create database zabbix character set utf8 collate utf8_bin;
Query OK, 1 row affected (0.001 sec)

MariaDB [(none)]> create user zabbix@localhost identified by 'password'
-> ;
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> grant all privileges on zabbix.* to zabbix@localhost;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that
MariaDB server version for the right syntax to use near 'privileges on zabbix.'
at line 1
MariaDB [(none)]> grant all privileges on zabbix.* to zabbix@localhost;
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> quit
Bye
```

Figura 1.4: Creación del usuario y la tabla en la BD

Ahora, si seguimos la documentación de Zabbix, nos indica que tenemos que inicializar la tabla de zabbix. Eso se consigue ejecutando el comando que viene en el propio tutorial.²

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

Esta orden, tediosa cuanto menos por el tiempo que requiere para completarse, no se especifica que sea estrictamente necesaria, por lo que recomendaría probar a seguir el tutorial sin incluir este paso y, en caso de que no funcionara, repetirlo incluyéndolo ya que, indudablemente, es el paso que más tiempo requiere de toda la instalación de Zabbix debido a la ingente cantidad de datos que necesita escribir, cosa que se ralentiza aún más al estar en una MV. Tras unas cuantas horas de espera, y al ver que no ha finalizado la

²El asterisco indica que la carpeta varía en función de la BD utilizada, en nuestro caso, usamos MariaDB/MySQL, por lo que dejamos el que se indica.

operación, personalmente voy a optar por cancelar con Ctrl+C y continuar como si esto no hubiera pasado, pero con la tranquilidad de que lo he intentado.

En caso de tener un error más adelante, siempre podremos repetir la orden y, si la habíamos cancelado, tan solo tendremos que añadir un -f al final de la misma.

Ya solo quedaría añadir la contraseña de la BD a los archivos de configuración de Zabbix para que pueda acceder a la misma. El archivo en cuestión que tenemos que modificar es el siguiente: `/etc/zabbix/zabbix_server.conf`, en concreto, tendremos que añadir el parámetro que vemos en la figura 1.5.

```
### Option: DBPassword
# Database password.
# Comment this line if no password is used.
#
# Mandatory: no
# Default:
DBPassword=password_
```

Figura 1.5: Aportación de la contraseña de la BD a Zabbix

Una vez aportada la contraseña, el apartado de la base de datos debería estar finalizado.

1.3. Configuración de PHP para el frontend de Zabbix

Esta subsección resulta ser bastante corta, ya que tan solo necesitamos editar un archivo de configuración (`/etc/zabbix/apache.conf`) para establecer la zona horaria, como vemos en la figura 1.6.

```
php_value date.timezone Europe/Madrid_
```

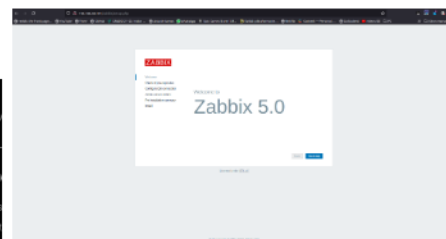
Figura 1.6: Establecimiento de la zona horaria

1.4. Configuración del frontend de Zabbix

Una vez que hemos realizado todos estos pasos vamos a, antes de nada, reiniciar los servicios como se ve en la figura 1.7a y acceder al frontend desde nuestra máquina host, viendo lo que se observa en la figura 1.7b.

```
osergiloseg@ubuntu-server:~$ sudo systemctl restart zabbix-server zabbix-agent apache2
[sudo] password for osergiloseg:
osergiloseg@ubuntu-server:~$ sudo systemctl restart zabbix-server zabbix-agent apache2
Synchronizing state of zabbix-server.service with SysV service script with /lib/systemd/systemd-sysd-install.
Executing: /lib/systemd/systemd-sysd-install
Synchronizing state of zabbix-agent.service with SysV service script with /lib/systemd/systemd-sysd-install.
Executing: /lib/systemd/systemd-sysd-install
Synchronizing state of apache2.service with SysV service script with /lib/systemd/systemd-sysd-install.
Executing: /lib/systemd/systemd-sysd-install
Created symlink /etc/systemd/system/multi-user.target.wants/zabbix-server.service → /lib/systemd/system/zabbix-server.service.
Created symlink /etc/systemd/system/multi-user.target.wants/zabbix-agent.service → /lib/systemd/system/zabbix-agent.service.
```

(a) Reinicio de servicios



(b) Visualización del frontend

Figura 1.7: Primer acceso al frontend

Ahora si vamos pasando por los distintos apartados que nos ofrece la configuración del frontend, iremos visualizando el correcto funcionamiento e instalación, teniendo que

especificar algunos valores como la contraseña de la BD y verificar el funcionamiento de PHP como vemos en la figura 1.8.

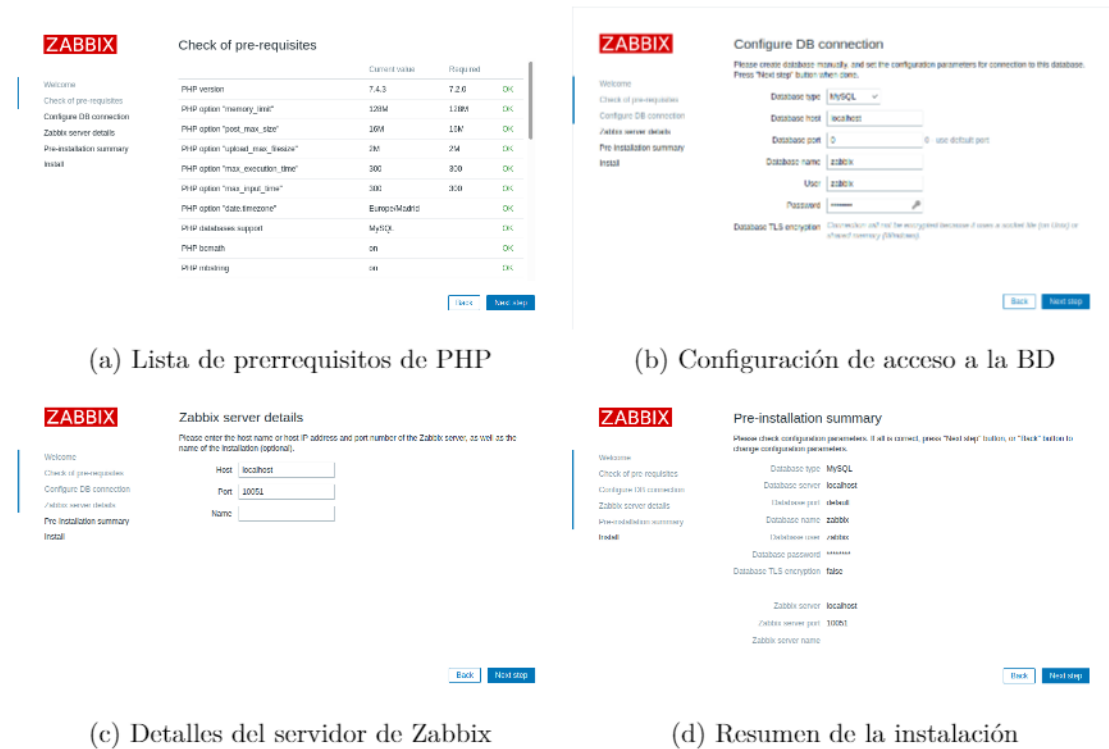


Figura 1.8: Configuración desde el frontend

Si seguimos la documentación, nuestro nuevo usuario de acceso al frontend de Zabbix será **Admin** con contraseña **zabbix**. Con esto, nos loguearemos como superusuarios en Zabbix, donde tendremos acceso a los menús de configuración y administración. Aquí encontraremos algo similar a lo que vemos en la figura 1.9.

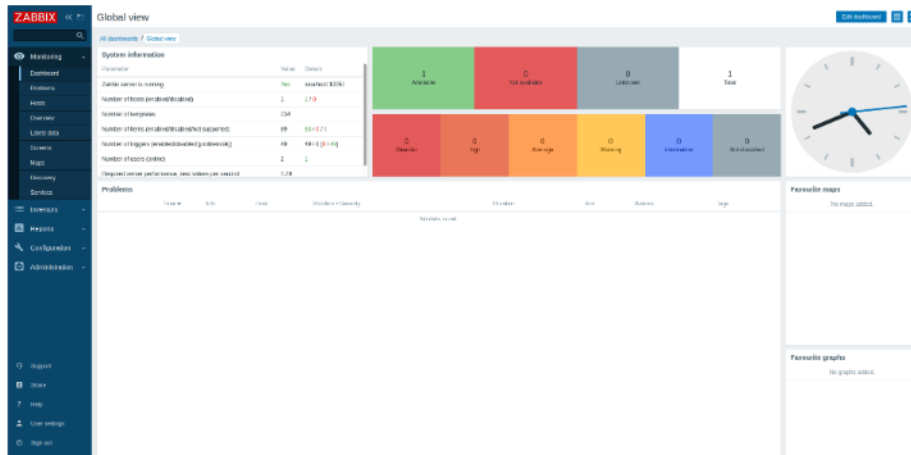


Figura 1.9: Página de inicio de Zabbix, sin configuración previa

En esta última figura podemos observar ya cierta información, sobre, por ejemplo, donde está corriendo zabbix, si está o no funcionando, los hosts, los templates...

Lo óptimo en una situación real sería crear un usuario nuevo y cambiar la contraseña por defecto, ya que aportaría una capa extra de seguridad, pese a que Zabbix tiene ya procedimientos para evitar ataques de diccionario y por fuerza bruta. Aún así, vamos a optar por no crearlo, ya que para el motivo de este informe, no es totalmente determinante y, como hemos dicho antes, este equipo no va a estar realmente expuesto a internet.

1.5. Configuración del host de CentOS

Vamos a instalar un agente en nuestra MV con CentOS para monitorizarlo también. Para ello, una vez que tengamos la máquina funcionando, tenemos que desactivar SELinux. Esto es totalmente recomendable si queremos ahorrarnos dolores de cabeza a la hora de configurar el agente de Zabbix. Para desactivar SELinux, editamos el fichero de configuración `/etc/selinux/config` y marcamos que esté deshabilitado por defecto, tal y como vemos en la figura 1.10.

```
SELINUX=disabled
```

Figura 1.10: Configuración de SELinux

Una vez desactivado SELinux, tendremos que descargar y configurar el agente en la máquina de CentOS.

El primer paso es descargar el agente, pero antes tendremos que añadir el repositorio.

```
$ sudo rpm -Uvh https://repo.zabbix.com/zabbix/5.0/rhel/8/
x86_64/zabbix-release-5.0-1.el8.noarch.rpm
$ sudo dnf clean all
```

Una vez hecho esto, ya podremos descargarlo:


```
$ sudo dnf install zabbix-agent
```

E iniciar el servicio:

```
$ sudo systemctl enable --now zabbix-agent
```

Si comprobamos el estado del servicio, debería aparecer algo parecido a lo que vemos en la figura 1.11.

```
[joseg@localhost ~]$ sudo systemctl status zabbix-agent
zabbix-agent.service - Zabbix Agent
Loaded: loaded (/usr/lib/systemd/system/zabbix-agent.service; enabled; vendor preset: disabled)
Active: active (running) since Sat 2021-12-25 11:15:16 EST; 8s ago
Process: 1841 ExecStart=/usr/sbin/zabbix_agentd -c $CONFFILE (code=exited, status=0/SUCCESS)
Main PID: 1843 (zabbix_agentd)
Tasks: 6 (limit: 4963)
Memory: 3.7M
CGroup: /system.slice/zabbix-agent.service
└─1843 /usr/sbin/zabbix_agentd -c /etc/zabbix/zabbix_agentd.conf
  └─1844 /usr/sbin/zabbix_agentd: collector [idle 1 sec]
    └─1845 /usr/sbin/zabbix_agentd: listener #1 [waiting for connection]
      └─1846 /usr/sbin/zabbix_agentd: listener #2 [waiting for connection]
        └─1847 /usr/sbin/zabbix_agentd: listener #3 [waiting for connection]
          └─1848 /usr/sbin/zabbix_agentd: active checks #1 [idle 1 sec]

dic 25 11:15:16 localhost.localdomain systemd[1]: Starting Zabbix Agent...
dic 25 11:15:16 localhost.localdomain systemd[1]: zabbix-agent.service: Can't open PID file /run/zabbix-agent.pid: Permission denied
dic 25 11:15:16 localhost.localdomain systemd[1]: Started Zabbix Agent.
lines 1-18/18 (END)
```

Figura 1.11: Servicio del agente de Zabbix en CentOS

una vez instalado el agente, tenemos que configurar sus parámetros para que proporcionen la información a la dirección correcta. Esto se consigue editando el archivo `/etc/zabbix/zabbix_agentd.conf`. En la figura 1.12 podemos ver los parámetros de configuración, donde *Server* y *ServerActive* llevan la dirección de la máquina donde está corriendo el servidor de Zabbix (la de Ubuntu Server en nuestro caso) y *Hostname* es el nombre que aparecerá en la web de monitorización para referirnos a este sistema.

```
Server=192.168.56.101 ServerActive=192.168.56.101 Hostname=CentOs8_
```

Figura 1.12: Configuración de parámetros de agente de Zabbix en CentOS

Ahora recargamos el agente con

```
$ sudo systemctl restart zabbix-agent
```

y vamos a configurar el firewall para permitir la transmisión de datos en los puertos y servicios que usamos³.

```
$ sudo firewall-cmd --add-service={http,https} --permanent
$ sudo firewall-cmd --add-port={10051/tcp,10050/tcp} --permanent
$ sudo firewall-cmd --reload
```

Una vez hecho esto, en la configuración de la web de Zabbix, tendremos que añadir que se monitorice este nuevo host.

³Nosotros estamos usando los puertos por defecto, que son el 10051 y el 10050

Para ello nos vamos al menú de *Configuration*→*Hosts* y en la esquina superior derecha pulsamos sobre *Create host* y rellenamos los valores pertinentes correctamente, obteniendo, en nuestro caso, lo que vemos en la figura 1.13.

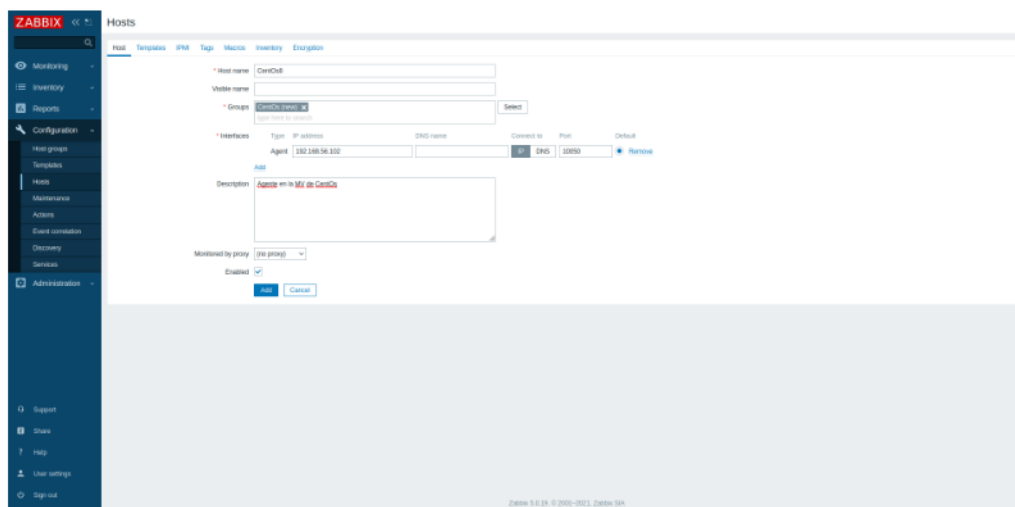


Figura 1.13: Creación del host de CentOs en la web de Zabbix

1.6. Configuración de monitorización

Una vez añadidos los hosts que queramos, vamos a indicarle a Zabbix qué es aquello que tiene que monitorizar. Vamos a empezar por echarle un ligero vistazo a los items predefinidos de zabbix para ver si con ellos tenemos suficiente.

Personalmente, nos interesa medir la carga de CPU, la carga de disco y monitorizar los servicios SSH y HTTP.

Por lo que parece al buscar en la lista de items predefinidos, podemos realizar nuestra monitorización sin necesidad de declarar nuevos items, ya que existe:

- *net.tcp.service.perf[service,<ip>,<port>]*, donde especificando el servicio podemos monitorizar SSH y HTTP.
- *system.cpu.load[<cpu>,<mode>]*, para medir la carga de la CPU.
- *vfs.dev.read[<device>,<type>,<mode>]* y *vfs.dev.write[<device>,<type>,<mode>]*, para medir la lectura y escritura en disco.

Vamos a irnos al apartado *Configuration*→*Hosts* y vamos a crear nuevos *items* para CentOs (los que hemos indicado en la lista de arriba). En la figura 1.14 podemos observar como creamos los distintos items y en la figura 1.15 podemos ver el resultado de cómo se empiezan a monitorizar automáticamente los items en cuestión.



Figura 1.14: Configuración de los items para el agente de CentOs

Item	Host	Application	Last check	Last value	Change
CPU load	CentOs	CPU load	2021-12-25 18:21:05	0.11	+0.06
Disk read	CentOs	Disk read	2021-12-25 18:21:07	0	-0
Disk write	CentOs	Disk write	2021-12-25 18:21:06	3.5833	-15.1167
HTTP monitoring	CentOs	HTTP monitoring	2021-12-25 18:21:09	0	-0
SSH monitoring	CentOs	SSH monitoring	2021-12-25 18:21:08	0.906483	-0.000017

Figura 1.15: Visualización de las primeras monitorizaciones

Una vez hayamos estado monitorizando durante cierto tiempo, podemos pedirle a Zabbix que nos muestre gráficas. Esto se hace seleccionando todos los items y clickando en *Display graph*. Seleccionamos nuestro intervalo de tiempo favorito y podremos disfrutar

de una gráfica similar a la de la figura 1.16.

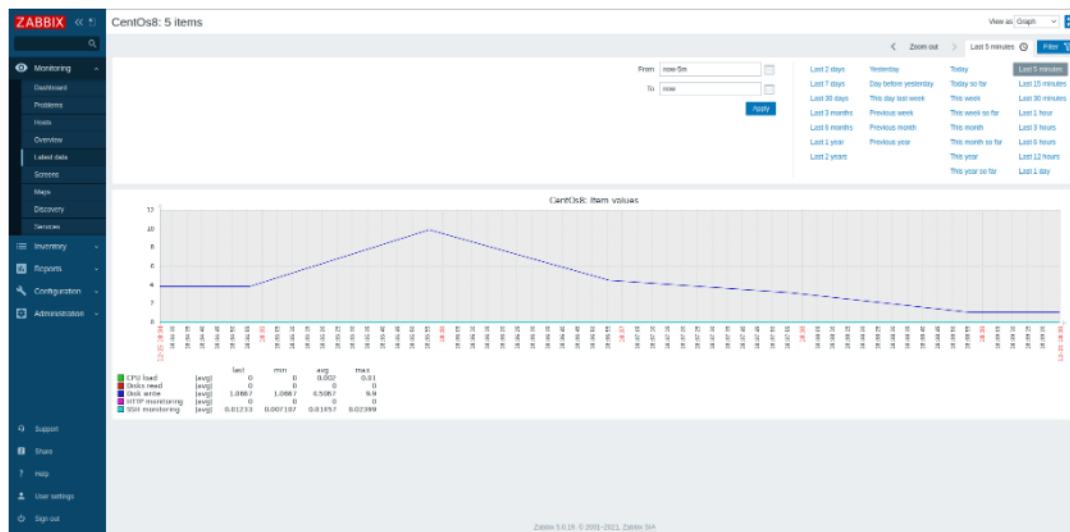


Figura 1.16: Gráficas generadas a partir de las monitorizaciones

Con esto, ya podemos monitorizar servicios con Zabbix. Para más comodidad, se podría incluir en el inicio las gráficas y mediciones que sean más importantes o que nos resulten de mayor interés para evitar el desplazarnos por los menús.

2. Instalación y uso de Ansible

Vamos a realizar la instalación de Ansible sobre Ubuntu Server.

Al igual que con Zabbix, seguiremos la documentación para guiarnos.

Antes de comenzar, quiero citar la propia web de Ansible, donde se comenta lo siguiente:

Ansible is simple, agentless automation that anyone can use

Esto por un lado nos indica que no necesita agentes en el lado del cliente, es decir, tan solo vamos a necesitar configurar el servidor. En el lado del cliente solo necesitaremos abrir los puertos necesarios a lo sumo.

Por otro lado, nos indica que el proceso de instalación y uso va a ser sencillo, no va a ser complicado.

Dicho esto, vamos a comenzar con la instalación de Ansible.

2.1. Instalación de Ansible en Ubuntu Server

En la instalación de Ansible, se habla de *nodos de control* y *nodos controlados*, siendo los primeros aquellos que controlan y los segundos aquellos a los que les mandamos órdenes. Ahora mismo nos interesan los nodos de control, que va a ser nuestra MV con Ubuntu Server. El único requerimiento para un nodo de control es que tenga Python 3.8 instalado. Esto se comprueba ejecutando `sudo apt install python3` en la máquina de control, así, en caso de no tenerlo, lo descargaremos.

A su vez, instalaremos el paquete `python-is-python3` para agilizar la sintaxis.

```
joseg@joseg-ubuntu-server:~$ sudo apt install python3 python-is-python3
[sudo] password for joseg:
Reading package lists... Done
Building dependency tree
Reading state information... Done
python3 is already the newest version (3.8.2-0ubuntu2).
python3 set to manually installed.
The following NEW packages will be installed:
  python-is-python3
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 2364 B of archives.
After this operation, 10.2 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://es.archive.ubuntu.com/ubuntu focal/main amd64 python-is-python3 all 3.8.2-4 [2364 B]
Fetched 2364 B in 1s (4503 B/s)
Selecting previously unselected package python-is-python3.
(Reading database ... 75159 files and directories currently installed.)
Preparing to unpack .../python-is-python3_3.8.2-4_all.deb ...
Unpacking python-is-python3 (3.8.2-4) ...
Setting up python-is-python3 (3.8.2-4) ...
```

Figura 2.1: Instalación de Python y paquete auxiliar

Como vemos en la figura 2.1, ya teníamos *Python3.8* instalado, pero hemos instalado el paquete `python-is-python3`, lo que nos facilitará la ejecución de scripts en caso de ser necesarios.

Si seguimos leyendo la documentación, nos vamos a dar cuenta de que existen 2 versiones distintas de Ansible para instalar disponibles en los repositorios: *ansible* y *ansible-core*⁴. *ansible-core* es una versión minimalista del propio *ansible*, ya que este último es una versión con módulos y colecciones preinstaladas.

⁴ *ansible-base* en la versión 2.10

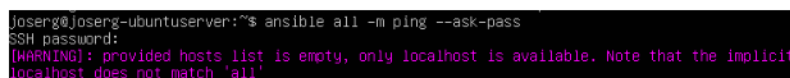
Vamos a optar por instalar el paquete normal, *ansible*, ya que no tenemos problemas de espacio y la velocidad de ejecución tampoco es un problema que nos preocupe ahora mismo al no tratarse de un escenario real.

Podemos instalar con el gestor de paquetes del sistema *apt* o con el gestor de paquetes de Python *pip*. En nuestro caso instalamos con el gestor de paquetes del sistema con la siguiente orden:

```
$ sudo apt update
$ sudo apt install software-properties-common
$ sudo add-apt-repository --yes --update ppa:ansible/ansible
$ sudo apt install ansible
```

Con esto se instalan también las posibles dependencias que nos falten.

Una vez instalado, podemos confirmar que todo está correcto realizando lo que observamos en la figura 2.2, en la que obtenemos el funcionamiento esperado.



```
joserg@joserg-ubuntuserver:~$ ansible all -m ping --ask-pass
SSH password:
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit
localhost does not match 'all'
```

Figura 2.2: Comprobación de la instalación de ansible

2.2. Configuración de Ansible

Desde la versión 2.4 de Ansible podemos usar la utilidad *ansible-config* para modificar el comportamiento de Ansible.

Esta utilidad modifica y lee del archivo de configuración de Ansible */etc/ansible/ansible.cfg*. La configuración que indica este archivo se aplica a todos los usuarios, pero cada usuario puede sobrescribir parámetros de la misma en el archivo *~/ansible.cfg*.

También se pueden establecer parámetros de configuración con variables de entorno, que sobrescribirán los ajustes de los archivos de configuración.

Y por si fuera poco, también están los parámetros de ejecución del propio Ansible, que sobrescriben a todos los anteriores.

En nuestro caso, de momento dejaremos la configuración por defecto.

2.3. Creando un inventario de Ansible

Lo que se conoce como un inventario de Ansible es un conjunto de máquinas que queremos controlar. En nuestro caso vamos a crear uno muy simple, con tan solo la IP de la máquina de CentOS (192.168.56.102) y la IP local (127.0.0.1). Los hemos agrupado bajo el alias de *prueba*. Esto hará que cuando queramos ejecutar comandos sobre estos dos servidores, en lugar de llamarlos uno a uno, podremos llamarlos de la forma *ansible prueba ...*⁵.

En la figura 2.3 podemos observar cómo quedaría en el archivo de configuración (*/etc/ansible/hosts*).

⁵Aunque ya que solo tenemos dos máquinas configuradas, si no pensamos llamarlas por separado podríamos hacer *ansible all ...*, que tendría el mismo efecto

```
[prueba]
192.168.56.102
127.0.0.1
```

Figura 2.3: Comprobación de la instalación de ansible

Vamos a hacer un ping para comprobar que todo se haya realizado correctamente. Es importante que a la hora de llamar a la orden de ansible desde la máquina de control indiquemos la opción `-ask-pass` para que nos pida la contraseña de ssh de las máquinas, puesto que no hemos realizado, ni vamos a realizar, un `ssh-copy-id`, que nos permitiría conectarnos sin contraseña.

Vamos a comparar los resultados con la opción `-ask-pass` y sin ella en la figura 2.4.



Figura 2.4: Ping de prueba simple con ansible

Como vemos en la figura 2.4a, al introducir la contraseña de nuestras máquinas, funciona el ping mientras que en la 2.4b falla.

2.4. Ejecución de comandos y creación de *Playbooks*

Antes de nada, vamos a hablar de cómo escalar privilegios, cosa que puede ser de ayuda si necesitamos ejecutar alguna orden con permisos en otra máquina. Para ello, basta con añadir a la orden de Ansible el parámetro `-become`. Este parámetro provoca que al conectar con la máquina se ejecute la orden como `sudo`.

También podemos especificar que se realice como otro usuario con `-become -become-user NombreUsuario`.

Ahora, vamos a crear un script sencillo, el de la figura 2.5, para borrar los elementos de la carpeta temporal.

Vamos a ver cómo podríamos crear una tarea con Ansible para automatizar este proceso y ejecutarlo en todas las máquinas con una sola orden.⁶

⁶Esto es simplemente para ilustrar que se puede programar una tarea con Ansible de manera sencilla utilizando *Playbooks*, la tarea a implementar ya depende del lector


```
#!/bin/bash
rm -rf /tmp/*
echo "Eliminado contenido de la carpeta temporal"
```

Figura 2.5: Script básico para borrar el contenido de la carpeta temporal

Al crear un *Playbook* como el de la de figura 2.6a y ejecutarlo puntualmente, obtenemos el resultado que se puede ver en la otra figura, la 2.6b.

```

- name: Delete elements in /tmp
  hosts: prueba
  remote_user: joserg
  tasks:
    - name: Copy and execute the script
      script: /home/joserg/simpleScript.sh

```

```

joserg@osmpedun01:~/server7$ ansible-playbook playbook.yml --ask-pass
SSH password:
PLAY [Delete elements in /tmp] *************************************
TASK [Gathering Facts] *********************************************
ok: [192.168.55.101]
TASK [Copy and execute the script] *****************************
changed: [192.168.55.101]
PLAY RECAP *********************************************************
192.168.55.101 : ok=2  changed=1  unreachable=0  failed=0  skipped=0  rescued=0

```

(a) Contenido del *Playbook*

(b) Resultado de ejecución del *Playbook*

Figura 2.6: Ejecución de *Playbook* sencillo

2.5. Automatización de tareas con *Playbooks*

Partiendo del *Playbook*⁷ que aparece en la figura 2.6a, vamos a automatizar el proceso para que se ejecute de forma periódica sin nuestra intervención.

Para ello, tan solo necesitamos hacer uso del *demonio cron*, ya que por defecto y al contrario que Puppet, Ansible no admite este tipo de configuración. Lo que tendríamos que hacer sería una configuración sencilla de cron, indicándole que ejecutara la orden que queremos, en este caso:

```
$ ansible-playbook /home/joserg/playbook.yml
```

Cada cierto intervalo de tiempo.

Un tema que tratamos previamente fue el de tener que introducir la contraseña manualmente cada vez que queremos lanzar una orden con Ansible.

Pues bien, si queremos automatizar la tarea con *cron*, necesariamente tenemos que prescindir de la contraseña, por lo que será necesario hacer un *ssh-copy-id* a todos las máquinas que queramos controlar antes de automatizar la tarea, prescindiendo así de la contraseña.

Otra opción, si no queremos usar cron, aunque mucho menos útil, sería usar la orden *at* para programar el lanzamiento del *Playbook*.

Esto no es una solución real, puesto que no estamos programando nada realmente, ya que la orden solo se ejecutaría una vez, por lo que no serviría para el supuesto planteado.

⁷Puede encontrar más información sobre los *Playbooks* y YAML en el siguiente enlace

3. Conclusión

A lo largo de estas páginas, hemos comprendido el funcionamiento básico y el potencial de algunas herramientas como son Ansible y Zabbix.

Respecto a Zabbix, hemos aprendido a monitorizar una o varias máquinas, al igual que a realizar una instalación básica de su servidor y ponerlo a funcionar de manera correcta, al igual que hemos investigado sus diferentes capacidades, como es el caso de los items que vienen predefinidos, aunque no hayamos implementado la gran mayoría.

Respecto a Ansible, hemos aprendido a crear un nodo de control, a ejecutar órdenes sencillas desde él y a gestionar un inventario con distintos *aliases* y grupos, al igual que a crear *Playbooks* a menor y mayor complejidad y visualizar de forma generalizada la funcionalidad de los módulos incluidos en el paquete de Ansible.

A su vez, hemos descubierto nuevas herramientas, como sería *Puppet*, y hemos practicado con tecnologías con las que ya estamos familiarizados en mayor o menor grado, como *apache* o los propios SOs con los que hemos trabajado.