



Práctica 3: Monitorización y "Profiling" Ingeniería de Servidores

Raúl Durán Racero

27 de noviembre de 2021



Índice

1. Ejercicio 1	3
1.1. Instalación de Zabbix en UbuntuServer	3
1.2. Configuración	6
2. Ejercicio 2	9

1. Ejercicio 1

Realice una instalación de Zabbix 5.0 en su servidor con **Ubuntu Server 20.04** y configure para que se monitorice 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**.

1.1. Instalación de Zabbix en UbuntuServer

Por comodidad a la hora de instalar, me conectaré a la máquina virtual de UbuntuServer a través de SSH, para poder copiar algunos comandos que son muy largos.

Lo primero que tenemos que hacer es instalar Zabbix en Ubuntu. Para ello, descargamos su paquete:

```
durar@durar:~$ wget https://repo.zabbix.com/zabbix/5.0/ubuntu/
pool/main/z/zabbix-release/zabbix-release_5.0-1+focal_all.
deb
```

Una vez descargado, lo instalamos con `dpkg -i` y obtenemos las actualizaciones con `apt update` (es necesario tener permisos de superusuario):

```
durar@durar:~$ sudo dpkg -i zabbix-release_5.0-1+focal_all.deb
durar@durar:~$ sudo apt update
```

Nos falta instalar el servidor, la interfaz y el agente de Zabbix:

```
durar@durar:~$ sudo apt install zabbix-server-mysql zabbix-
frontend-php zabbix-apache-conf zabbix-agent
```

Ya tenemos todo debidamente instalado, por lo que pasaremos a crear una base de datos inicial y un usuario, dándole todos los permisos (entraré en mysql como superusuario ya que no me permite entrar con `mysql -u root -p`).

```
durar@durar:~$ sudo mysql
mysql> create database zabbix character set utf8 collate
utf8_bin;
mysql> create user zabbix@localhost identified by 'ISE';
mysql> grant all privileges on zabbix.* to zabbix@localhost;
mysql> quit;
```

A continuación tenemos que ejecutar el script que establece el esquema por defecto de la base de datos.

```
durar@durar:~$ zcat /usr/share/doc/zabbix-server-mysql*/create.
sql.gz | mysql -uzabbix -p zabbix
```

Esto puede tardar un tiempo. Podemos comprobar que se está ejecutando con `htop` desde la máquina de UbuntuServer:

Puede verse que efectivamente mysql está en ejecución.
Ya tenemos la base de datos creada, así que ahora toca configurarla. Primero, editamos el archivo `/etc/zabbix/zabbix_server.conf`:

```
durar@durar:~$ sudo vi /etc/zabbix/zabbix_server.conf
```

Y ponemos nuestra contraseña en la línea donde está **DBPassword**:

```
DBPassword=ISE
```

Para buscar la línea, se puede usar **Shift+7** en vi para buscar DBPassword más fácilmente.

A continuación, configuramos PHP para la interfaz de Zabbix, editando el archivo `/etc/zabbix/apache.conf`, descomentando y cambiando la zona horaria:

```
durar@durar:~$ sudo vi /etc/zabbix/apache.conf
> php_value date.timezone Europe/Madrid
```

Además de esto, hay que cambiar también el archivo `/etc/php/7.4/apache2/php.ini`, ya que si no lo hacemos nos dará un error más tarde, así que lo hacemos ahora:

```
durar@durar:~$ sudo vi /etc/php/7.4/apache2/php.ini
> date.timezone = UTC
```

Iniciamos los procesos del agente y del servidor de Zabbix y los configuramos para que se inicien a la par que el sistema (*enable*):

```
durar@durar:~$ systemctl restart zabbix-server zabbix-agent
apache2
durar@durar:~$ systemctl enable zabbix-server zabbix-agent
apache2
```

Por último, nos conectamos a nuestra interfaz Zabbix desde el navegador para instalar el *frontend*: `192.168.56.105/zabbix`

Nos deberá aparecer una interfaz de instalación de Zabbix, donde seguiremos los pasos siguientes:

Welcome: Next Step

Check of pre-requisites: Comprobamos que todo esté correcto ->Next Step

Configure DB connection: Introducimos la contraseña de la BD ->Next Step

Zabbix server details: Next Step

Preinstallation summary: Next Step

install: Finish

Nos notificará de que lo hemos instalado con éxito, y podremos logearnos:

User: Admin

Password: zabbix

Instalación en CentOS

La instalación en CentOS es similar a la de UbuntuServer, incluso más sencilla, ya que solo hay que instalar el agente: Empezamos instalando el repositorio de Zabbix y el paquete del agente:

```
[durar@localhost ~]$ sudo rpm -Uvh https://repo.zabbix.com/
zabbix/5.0/rhel/8/x86_64/zabbix-release-5.0-1.el8.noarch.
rpm
[durar@localhost ~]$ dnf clean all
[durar@localhost ~]$ sudo dnf install zabbix-agent
```

Hay que cambiar el archivo de configuración en `/etc/zabbix/zabbix_agentd.conf`, modificando los valores que se muestran resaltados en las siguientes capturas:

```
durar@localhost:~$ cat /etc/zabbix/zabbix_agentd.conf
# Enable logging of executed shell commands as warnings.
# 0 - disabled
# 1 - enabled
#
# Mandatory: no
# Default:
# LogRemoteCommands=0

##### Passive checks related

### Option: Server
# List of comma delimited IP addresses, optionally in CIDR notation, or DNS names of
# Zabbix servers and Zabbix proxies.
# Incoming connections will be accepted only from the hosts listed here.
# If IPv6 support is enabled then '127.0.0.1', '::127.0.0.1', '::ffff:127.0.0.1' are
# treated equally
# and '::/0' will allow any IPv4 or IPv6 address.
# '0.0.0.0/0' can be used to allow any IPv4 address.
# Example: Server=127.0.0.1,192.168.1.0/24,::1,2001:db8::/32,zabbix.example.com
#
# Mandatory: yes, if StartAgents is not explicitly set to 0
# Default:
# Server=

Server=192.168.56.105

### Option: ListenPort
# Agent will listen on this port for connections from the server.
#
-- INSERT --

durar@localhost:~$ cat /etc/zabbix/zabbix_agentd.conf
### Option: ServerActive
# List of comma delimited IP:port (or DNS name:port) pairs of Zabbix servers and Zabbix
# proxies for active checks.
# If port is not specified, default port is used.
# IPv6 addresses must be enclosed in square brackets if port for that host is specified.
# If port is not specified, square brackets for IPv6 addresses are optional.
# If this parameter is not specified, active checks are disabled.
# Example: ServerActive=127.0.0.1:20051,zabbix.domain,[:1]:30051,::1,[12fc::1]
#
# Mandatory: no
# Default:
# ServerActive=

ServerActive=192.168.56.105

### Option: Hostname
# Unique, case sensitive hostname.
# Required for active checks and must match hostname as configured on the server.
# Value is acquired from HostnameItem if undefined.
#
# Mandatory: no
# Default:
# Hostname=

Hostname=Zabbix server

### Option: HostnameItem
-- INSERT --
```

Iniciamos y establecemos el proceso del agente de Zabbix:

```
[durar@localhost ~]$ systemctl enable zabbix-agent
[durar@localhost ~]$ systemctl start zabbix-agent
```

1.2. Configuración

Ya tenemos el frontend de Zabbix en el navegador, así que ahora añadimos las máquinas que hay que monitorizar. El host de UbuntuServer viene por defecto con la instalación que hicimos previamente. Este host se encarga de monitorizar el propio servidor. Nos falta el de CentOS, así que lo creamos: Configuration -> Hosts -> Create Host y añadir los datos siguientes:

Host configuration page in Zabbix. Fields include Host name (CentOS), Visible name, Groups (Linux servers), and Interfaces. The interface table has columns: Type, IP address, DNS name, Connect to, Port, and Default. One interface is listed: Agent, 192.168.56.110, IP, 10050. A red arrow points to the port '10050'.

Nos daremos cuenta de que el puerto que se utiliza es el 10050, como señala la flecha, así que tendremos que abrir dicho puerto en CentOS.

```
[ durar@localhost ~]$ sudo firewall-cmd --add-port=10050/tcp --permanent
[ durar@localhost ~]$ sudo firewall-cmd --add-port=10050/tcp
[ durar@localhost ~]$ sudo firewall-cmd --reload
```

Ya tenemos el puerto abierto y el host de CentOS, así que ya podemos monitorizar ambas máquinas. Nos piden que monitoricemos los servicios SSH y HTTP en ambas máquinas. Para ello, tenemos que añadir un nuevo ítem para cada servicio en las dos máquinas. Así que nos dirigimos a Configuration -> Hosts -> Zabbix server -> Items, le damos a *Create Item* e introducimos los datos que se muestran en las figuras. Repetimos como corresponda para el resto de ítems.

Figura 1: SSH de UbuntuServer

Item configuration page in Zabbix. Fields include Name (Monitor SSH), Type (Zabbix agent), Key (net.tcp.service[ssh,22022]), Host interface (127.0.0.1:10050), Type of information (Numeric (unsigned)), Units, Update interval (1m), Custom intervals (Flexible Scheduling, 50s, 1-7,00:00-24:00), History storage period (90d), and Trend storage period (365d).

Figura 2: SSH de CentOS

The screenshot shows the Zabbix web interface for configuring an item. The left sidebar contains navigation links: Monitoring, Inventory, Reports, Configuration (expanded), and Administration. The main content area is titled 'Items' and shows the configuration for an item named 'Monitor SSH'. The configuration includes:

- Name:** Monitor SSH
- Type:** Simple check
- Key:** net.tcp.service[ssh,,22022]
- Host interface:** 192.168.56.110 : 10050
- User name:** (empty field)
- Password:** (empty field)
- Type of information:** Numeric (unsigned)
- Units:** (empty field)
- Update interval:** 1m
- Custom intervals:** A table with columns Type, Interval, Period, and Action. It shows a 'Flexible' interval of 50s and a 'Scheduling' interval of 1-7,00:00-24:00.
- History storage period:** Do not keep history
- Trend storage period:** Do not keep trends
- Show value:** As is
- New application:** (empty field)
- Applications:** None

Para monitorizar el servicio HTTP en UbuntuServer, queremos comprobar si funciona el servicio de apache. Zabbix ya nos ofrece un *Template* configurado para Apache, así que lo utilizaremos:

Figura 3: HTTP para UbuntuServer

The screenshot shows the Zabbix web interface for configuring a host. The left sidebar contains navigation links: Monitoring, Inventory, Reports, Configuration (expanded), and Administration. The main content area is titled 'Hosts' and shows the configuration for a host named 'Zabbix server'. The configuration includes:

- Host:** Zabbix server
- Enabled:** Yes
- Link new templates:** A search bar with the text 'Template App Apache by HTTP' and a 'Select' button.
- Update:** A button to update the host configuration.
- Clone:** A button to clone the host configuration.
- Full clone:** A button to create a full clone of the host configuration.
- Delete:** A button to delete the host configuration.
- Cancel:** A button to cancel the configuration.

Figura 4: HTTP para CentOS

El tipo que hemos escogido en los monitores es *entering*, ya que lo que queremos es que simplemente que compruebe si el servicio está activo o no. La *key* la podemos seleccionar en el botón *Select*. Además, podemos modificar el intervalo con el que se actualizan los datos monitorizados (*Update Interval*).

2. 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.

Se nos pide instalar y configurar Ansible para poder hacer ping a nuestras máquinas virtuales de los servidores, nos hace falta otra máquina que sea la que hace ping. Por eso, clonaremos una de las dos (en mi caso lo haré en CentOS, por simpleza en la instalación), le cambiaremos su IP para que no coincida con la original e instalaremos en ella Ansible. Lo primero será, una vez clonada la máquina, cambiar su IP como hicimos en la práctica 1:

```
[durar@localhost ~]$ sudo vi /etc/sysconfig/network-scripts/
ifcfg-enp0s8
```

La Ip de la nueva máquina será **192.168.56.111**. Reiniciamos para aplicar los cambios y comprobamos la IP con *ifconfig*. Ahora toca instalar Ansible:

```
[durar@localhost ~]$ sudo yum install epel-release
[durar@localhost ~]$ sudo yum install ansible
```

Una vez, instalado, lo configuramos editando el archivo */etc/ansible/hosts* y añadimos las IPs de nuestras máquinas, con el puerto 22022, junto a una etiqueta que forme el grupo.

```

durar@localhost:~
## [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]
##
## db01.intranet.mydomain.net
## db02.intranet.mydomain.net
## 10.25.1.56
## 10.25.1.57

# Here's another example of host ranges, this time there are no
# leading 0s:

## db-[99:101]-node.example.com

[maquinasservers]
192.168.56.110:22022
192.168.56.105:22022
-- INSERT --

```

Ansible se comunica con las máquinas remotas mediante el protocolo SSH, así que al igual que en la práctica anterior, crearemos y añadiremos la clave pública a las máquinas:

```

[durar@localhost ~]$ ssh-keygen
[durar@localhost ~]$ ssh-copy-id 192.168.56.110 -p 22022
[durar@localhost ~]$ ssh-copy-id 192.168.56.105 -p 22022

```

Comprobamos con ssh que se han copiado bien, y hacemos ping:

```

[durar@localhost ~]$ ansible -m ping "maquinasservers"

```

Nos dará una salida como esta:

```

durar@localhost:~
[durar@localhost ~]$ [durar@localhost ~]$
[durar@localhost ~]$ ansible -m ping "maquinasservers"
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"
}
[durar@localhost ~]$

```

También podemos ejecutar el script de monitorización del RAID1 que vimos en clase:

```
[durar@localhost ~]$ ansible "maquinasservers" -a "python3 mon-raid.py"
```

Ansible nos permite escribir acciones más complejas mediante unos documentos YAML, los *playbooks*. Crearemos uno que cree un archivo temporal y nos muestre el estado del servicio SSH:

```
durar@localhost:~  
- -  
- name: My Playbook  
  hosts: maquinasservers  
  tasks:  
    - name: Leaving a mark  
      command: "touch /tmp/ansible_was_here"  
    - name: Comprobar que ssh se esta ejecutando  
      service:  
        name: sshd  
        state: started  
~  
~  
~
```

Para ejecutarlo:

```
[durar@localhost ~]$ ansible-playbook myplaybook.yaml
```

Nos dará la siguiente salida:

```
durar@localhost:~  
[durar@localhost ~]$ [durar@localhost ~]$  
[durar@localhost ~]$ ansible-playbook myplaybook.yaml  
PLAY [My Playbook] *****  
TASK [Gathering Facts] *****  
ok: [192.168.56.105]  
ok: [192.168.56.110]  
TASK [Leaving a mark] *****  
[WARNING]: Consider using the file module with state=touch rather than running 'touch'. If you need  
to use command because file is insufficient you can add 'warn: false' to this command task or set  
'command_warnings=False' in ansible.cfg to get rid of this message.  
changed: [192.168.56.105]  
changed: [192.168.56.110]  
TASK [Comprobar que ssh se esta ejecutando] *****  
ok: [192.168.56.105]  
ok: [192.168.56.110]  
PLAY RECAP *****  
192.168.56.105 : ok=3 changed=1 unreachable=0 failed=0 skipped=0 rescued=0  
ignored=0  
192.168.56.110 : ok=3 changed=1 unreachable=0 failed=0 skipped=0 rescued=0  
ignored=0  
[durar@localhost ~]$
```

Vemos que nos muestra un warning por el touch, y nos dice como solucionarlo o ignorarlo, pero acepta la otra tarea, que se ejecuta correctamente.