MONITORATGE AMB LINUX I DOCKER

Iker López
ITB

ÍNDEX

1. **Introducció** ..... 3 1.1. Justificació del projecte ..... 3 1.2. Avantatges del Programari Lliure 3
2. **Entorn Tecnològic** ..... 4 2.1. Ubuntu Server 22.04 LTS ..... 4 2.2. Tecnologia de Contenidors: Docker 5
3. **Desplegament de la Infraestructura** ..... 6 3.1. Preparació del Sistema Operatiu ..... 6 3.2. Instal·lació del Motor Docker i Docker Compose 7
4. **Implementació de Zabbix (Backend)** ..... 9 4.1. Definició de l'arxiu docker-compose.yml ..... 9 4.2. Desplegament de contenidors i BBDD 11
5. **Configuració del Monitoratge** ..... 12 5.1. Instal·lació d'Agents Zabbix en clients ..... 12 5.2. Configuració de Hosts i Templates ..... 13 5.3. Creació de Triggers (Alertes) 14
6. **Visualització de Dades amb Grafana** ..... 15 6.1. Connexió de Grafana amb Zabbix API ..... 15 6.2. Disseny de Dashboards interactius 16
7. **Automatització i Scripting** ..... 17 7.1. Script d'Auto-reparació de serveis ..... 17 7.2. Backup automatitzat de configuracions 18
8. **Conclusions** 20

1. INTRODUCCIÓ

1.1. Justificació del projecte

En l'administració de sistemes moderna, la reactivitat ja no és suficient. Els administradors no poden esperar que un usuari reporti una fallada; han de saber-ho abans que passi. Aquest projecte proposa la implementació d'un sistema **SIEM (Security Information and Event Management)** i monitoratge de rendiment utilitzant eines Open Source.

L'objectiu és tenir un control total ("Ulls que tot ho veuen") sobre l'estat de CPU, RAM, Discos i Xarxa de tots els servidors de l'organització, centralitzant les dades en un panell visual comprensible.

1.2. Avantatges del Programari Lliure

S'ha optat per Zabbix i Grafana davant de solucions propietàries per:

- **Cost zero de llicències:** Permet escalar a milers de dispositius sense cost addicional.
- **Comunitat:** Ampli suport i plantilles predefinides.
- **Flexibilitat:** Capacitat d'adaptar-se a qualsevol entorn mitjançant scripts personalitzats.

2. ENTORN TECNOLÒGIC

2.1. Ubuntu Server 22.04 LTS

Com a base del sistema s'utilitzarà la distribució Linux Ubuntu Server en la seva versió Long Term Support (Jammy Jellyfish). Aquesta elecció es basa en la seva estabilitat, seguretat i ampli suport per a tecnologies de contenidors. El servidor s'ha configurat sense entorn gràfic (headless) per optimitzar recursos.

2.2. Tecnologia de Contenidors: Docker

En lloc d'una instal·lació "bare-metal" tradicional (instal·lar Apache, PHP i MySQL directament al SO), utilitzarem **Docker**.

Docker permet empaquetar l'aplicació i les seves dependències en un "contenidor" aïllat. Això ofereix avantatges crítics:

1. **Portabilitat:** El sistema funciona igual al meu portàtil, al servidor de proves i a producció.
2. **Neteja:** No "embrutem" el sistema operatiu base amb llibreries conflictives.
3. **Manteniment:** Actualitzar Zabbix és tan fàcil com canviar una línia de versió en un fitxer de text i reiniciar el contenidor.

3.1. Preparació del Sistema Operatiu

Comencem actualitzant els repositoris del sistema i aplicant els pegats de seguretat.

```
sudo apt update && sudo apt upgrade -y
sudo apt install ca-certificates curl gnupg lsb-release
```

Configurem una IP estàtica al servidor de monitoratge editant el fitxer `/etc/netplan/00-installer-config.yaml`:

```
network:
  ethernets:
    ens33:
      addresses: [192.168.1.50/24]
      gateway4: 192.168.1.1
      nameservers:
        addresses: [8.8.8.8, 1.1.1.1]
  version: 2
```

3.2. Instal·lació del Motor Docker i Docker Compose

Instal·lem Docker des del repositori oficial per obtenir l'última versió.

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
/usr/share/keyrings/docker-archive-keyring.gpg
sudo apt install docker-ce docker-ce-cli containerd.io docker-compose-plugin
```

Verifiquem la instal·lació i habilitem el servei perquè arrenqui a l'inici:

```
sudo systemctl enable docker
sudo systemctl start docker
docker --version
```

4. IMPLEMENTACIÓ DE ZABBIX (BACKEND)

4.1. Definició de l'arxiu docker-compose.yml

Creem un directori `/opt/monitoratge` i a dins generem l'arxiu `docker-compose.yml`. Aquest arxiu orquestra tots els serveis necessaris: Base de dades, Servidor Zabbix, Interfície Web i Grafana.

```
version: '3.5'
services:
 # Servidor de Base de Dades MySQL
  mysql-server:
    image: mysql:8.0
    command: --default-authentication-plugin=mysql_native_password
    environment:
      - MYSQL_DATABASE=zabbix
      - MYSQL_USER=zabbix
      - MYSQL_PASSWORD=zabbix_pass
      - MYSQL_ROOT_PASSWORD=root_pass
    volumes:
      - ./mysql_data:/var/lib/mysql

 # Servidor Java Gateway (Per monitorar aplicacions Java)
  zabbix-java-gateway:
    image: zabbix/zabbix-java-gateway:latest
    ports:
      - "10052:10052"

 # Servidor Principal de Zabbix
  zabbix-server:
    image: zabbix/zabbix-server-mysql:latest
    ports:
      - "10051:10051"
    volumes:
      - /etc/localtime:/etc/localtime:ro
    links:
      - mysql-server:mysql-server
      - zabbix-java-gateway:zabbix-java-gateway
    environment:
      - DB_SERVER_HOST=mysql-server
      - MYSQL_DATABASE=zabbix
      - MYSQL_USER=zabbix
      - MYSQL_PASSWORD=zabbix_pass
```

```
# Interfície Web (Frontend)
 zabbix-web-nginx-mysql:
   image: zabbix/zabbix-web-nginx-mysql:latest
   ports:
     - "8080:8080"
   links:
     - mysql-server:mysql-server
     - zabbix-server:zabbix-server
   environment:
     - PHP_TZ=Europe/Madrid
     - MYSQL_SERVER_HOST=mysql-server
     - MYSQL_DATABASE=zabbix
     - MYSQL_USER=zabbix
     - MYSQL_PASSWORD=zabbix_pass
     - ZBX_SERVER_HOST=zabbix-server
     - ZBX_SERVER_PORT=10051

# Grafana per a visualització
 grafana:
   image: grafana/grafana:latest
   ports:
     - "3000:3000"
   volumes:
     - grafana-storage:/var/lib/grafana
   depends_on:
     - zabbix-server
```

4.2. Desplegament de contenidors

Un cop definit l'arxiu, llancem la pila de serveis amb una sola comanda. Docker descarregarà les imatges i aixecarà l'entorn.

```
cd /opt/monitoratge
sudo docker compose up -d
```

Comprovem que els contenidors estan corrent amb `docker ps`.

Accedim via navegador a `http://192.168.1.50:8080` i completem l'assistent d'instal·lació web de Zabbix.

5. CONFIGURACIÓ DEL MONITORATGE

5.1. Instal·lació d'Agents Zabbix

Zabbix monitora equips remots instal·lant un petit programari anomenat "Zabbix Agent". En un servidor client (ex. un servidor Web Apache), executem:

```
wget
https://repo.zabbix.com/zabbix/6.0/ubuntu/pool/main/z/zabbix-release/zabbix-release_6.0-4+ubuntu22.04_all.deb
sudo dpkg -i zabbix-release_6.0-4+ubuntu22.04_all.deb
sudo apt update
sudo apt install zabbix-agent
```

Editem `/etc/zabbix/zabbix_agentd.conf` per apuntar al nostre servidor:

```
Server=192.168.1.50
ServerActive=192.168.1.50
Hostname=ServidorWeb01
```

5.2. Configuració de Hosts i Templates

Des de la interfície web de Zabbix, anem a *Configuration* > *Hosts* > *Create Host*.

Assignem el nom `ServidorWeb01` i la IP de l'agent. El més important és vincular una **Template (Plantilla)**. Seleccionem "Linux by Zabbix Agent".

Aquesta plantilla predefinida crea automàticament centenars de mètriques:

- Ús de CPU per nucli.
- Memòria lliure i Swap.
- Trànsit d'entrada/sortida en targetes de xarxa.
- Espai lliure a `/`, `/home`, `/var`.

5.3. Creació de Triggers (Alertes)

Un sistema de monitoratge és inútil si no avisa. Configurem un "Trigger" personalitzat per detectar si el servidor web cau.

Expressió del Trigger: `last(/ServidorWeb01/net.tcp.service[http])=0`

- **Severitat:** Desastre.
- **Acció:** Enviar correu electrònic a l'administrador.

Hem provat el trigger parant manualment el servei Apache al client (`systemctl stop apache2`). Als 30 segons, el dashboard de Zabbix ha parpellejat en vermell i s'ha generat una incidència.

6. VISUALITZACIÓ DE DADES AMB GRAFANA

6.1. Connexió de Grafana amb Zabbix API

Tot i que Zabbix té gràfics, Grafana és superior estèticament i permet combinar fonts de dades. Accedim a `http://192.168.1.50:3000`.

1. Instal·lem el plugin de Zabbix per a Grafana.
2. A *Data Sources*, afegim Zabbix.
3. URL: `http://zabbix-web-nginx-mysql:8080/api_jsonrpc.php`.
4. Usuari i contrasenya de Zabbix.

6.2. Disseny de Dashboards interactius

Creem un panell de control "Executiu" que mostra l'estat de salut de la infraestructura d'un cop d'ull.

Components afegits al Dashboard:

- **Gràfic de sèrie temporal:** Càrrega de CPU de tots els servidors superposada.
- **Mesuradors (Gauges):** RAM usada actualment.
- **Semàfors:** Estat Online/Offline dels servidors (Ping).
- **Mapa de calor:** Latència de la xarxa.

El resultat és un panell professional en mode fosc que s'actualitza cada 5 segons.

7. AUTOMATITZACIÓ I SCRIPTING

7.1. Script d'Auto-reparació (Bash)

Per anar un pas més enllà del monitoratge, hem creat un script als clients que verifica si un procés crític ha mort i tracta d'aixecar-lo automàticament, registrant l'esdeveniment.

Script: `/usr/local/bin/autoheal.sh`

```
#!/bin/bash
SERVEI="nginx"
LOG="/var/log/autoheal.log"
DATA=$(date '+%Y-%m-%d %H:%M:%S')

# Comprovar si el servei està corrent
if systemctl is-active --quiet $SERVEI
then
   echo "[$DATA] $SERVEI està OK."
else
   echo "[$DATA] ALERTA: $SERVEI detingut. Intentant reiniciar..." >> $LOG
   systemctl start $SERVEI

   # Comprovar si ha reviscut
   if systemctl is-active --quiet $SERVEI
   then
      echo "[$DATA] ÈXIT: $SERVEI reiniciat correctament." >> $LOG
   else
      echo "[$DATA] ERROR CRÍTIC: No s'ha pogut iniciar $SERVEI." >> $LOG
   fi
fi
```

7.2. Backup automatitzat de configuracions

Atès que utilitzem Docker, el backup és senzill. Creem un script que exporta la base de dades MySQL del contenidor.

```
#!/bin/bash
# Backup diari de Zabbix DB
BACKUP_DIR="/mnt/backup/zabbix"
FILENAME="zabbix_db_$(date +%F).sql.gz"

mkdir -p $BACKUP_DIR
```

```
# Executar mysqldump dins del contenidor
docker exec zabbix-mysql-server mysqldump -u zabbix -pzabbix_pass zabbix | gzip
> $BACKUP_DIR/$FILENAME

# Esborrar backups antics (+30 dies)
find $BACKUP_DIR -type f -mtime +30 -name "*.gz" -delete
```

8. PROVES D'ESTRÈS

Per verificar la robustesa del sistema, es va utilitzar l'eina `stress-ng` en un dels clients per simular una càrrega del 100% de CPU.

Comanda executada: `stress-ng --cpu 4 --timeout 60s`

Resultats observats:

1. Als 5 segons, l'ús de CPU va pujar al gràfic de Grafana.
2. Als 10 segons, Zabbix va detectar "High CPU Load".
3. Als 15 segons, es va rebre un correu electrònic d'alerta.
4. En finalitzar l'estrès, el sistema va tornar a verd automàticament ("Problem Resolved").

9. CONCLUSIONS

El desplegament d'aquest sistema de monitoratge compleix amb tots els objectius plantejats. La integració de **Docker** ha simplificat enormement la instal·lació i gestió de dependències, permetent desplegar el servidor Zabbix en menys de 10 minuts.

La combinació de la potència de recol·lecció de dades de **Zabbix** amb la capacitat visual de **Grafana** ofereix una eina professional comparable a solucions de pagament, però basada enterament en estàndards oberts. La infraestructura és ara observable, permetent prendre decisions basades en dades reals d'ús i rendiment.

Bibliografia i Recursos:

- Documentació Oficial de Zabbix 6.0 LTS.
- Documentació de Docker i Docker Compose.
- Grafana Labs: "Building Dashboards with Zabbix Plugin".