**PRACTICA 3 - MONITORIZACIÓN, AUTOMATIZACIÓN Y PROFILING**

**Monitores para Hardware**

**dmesg:** muestra los mensajes del kernel. Sirve para poder detectar problemas conn el HW o periféricos. Se puede utilizar dmesg con cauce a grep para buscar la información que queramos saber (memoría física, dispositivos USB, advertencias, etc).

**Otros comandos útiles:** lspci, lsusb, lshw, que listan el hardware disponible y conectado.

**Monitores para Software**

**Subsistema de archivos:** en /proc y /var se encuentra información tanto del hardware como del software así como archivos fundamentales para la monitorización del sistema, de las aplicaciones y de los usuarios. */var/log* contiene los archivos en los que se van volcando los logs de los servicios y algunas aplicaciones.

**Identificar, monitorizar y reconstruir los discos de un RAID 1:**

**Force-fail by Software:**

*mdadm --manage --set-faulty /dev/md1 /dev/sdc2* (suficiente para eliminar el disco /edv/sdc2 del array /dev/md1). Sale que hay un error que el disco se ha ido blabla.

*mdadm --detail /dev/md1* (miramos que se ha ido el sdc2)

cat /proc/mdstat también muestra la información del fallo.

*mdadm /dev/md1 -r /dev/sdc2* (aquí eliminamos el failed disk de md1, este comando solo se puede utilizar cuando el disco está caído, evidentemente)

*mdadm /dev/md1 -a /dev/sdc2* (ahora volvemos a añadir el disco al array si queremos)

**Force-fail by Hardware:**

**Monitorizando un servicio o ejecución de un programa: strace**

Hay un conjunto de programas que permiten hacer una traza de las llamadas al sistema realizadas por un programa (servicio) en ejecución, p.ej. strace (system call tracer). Este tipo de programas pueden ser útiles de cara a detectar problemas que no se muestran en los archivos de “log”.

**Monitores generales**

* **Munin:** usa RRDTool1. Tiene una arquitectura maestro/esclavo (cliente/servidor) en la que el servidor conecta a todos los clientes en intervalos de tiempo (Round Robin) y les pide los datos. Luego los almacena en archivos RRD y actualiza los graficos. RRDtool(round robin database tool) es una herramienta que trabaja con una base de datos que maneja planificación según Round Robin. El funcionamiento es el siguiente: se trata la base de datos como si fuese un círculo, sobrescribiendo los datos almacenados con anterioridad una vez alcanzada la capacidad máxima de la misma. Esta capacidad máxima dependerá de la cantidad de información que se quiera conservar como historial. Su finalidad principal es el tratamiento de datos temporales y datos seriales como temperaturas, transferencias en redes, cargas del procesador, etc.
* **Naemon y Nagios:**
* **Ganglia:**
* **Cacti:**
* **AW stats:** todos los de arriba monitorizan el sistema.

**Zabbix (ver puidge final doc):**

**Ejercicio 1:** Realice una instalación de Zabbix 5.0 en su servidor con Ubuntu Server20.04 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.

**UbuntuServer:** Instalar el repositorio de zabbix. Para ello nos vamos a https://www.zabbix.com/download?zabbix=5.0&os\_distribution=ubuntu&os\_version=20.04\_focal&db=mysql y seguimos los pasos. Debajo, nos aparecen los comandos a seguir para realizar la instalación los cuales voy a listar:

errores:

dkpg frontend lock is locked by another process

Solución: ps -ef | grep apt para ver procesos que estan usando dkpg y matarlos o esperar

# sudo 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

Instalamos ahora el servidor, el front-end y el agente de zabbix.

sudo apt install -y zabbix-server-mysql zabbix-frontend-php zabbix-apache-conf zabbix-agent (-y sirve para decir que si a todo)

Tenemos ahora que habilitar apache. sudo systemctl start/enable/reload apache

Configuramos ahora la base de datos:

* La creamos:

(Seguir los pasos de la pagina siguiente para centOS)

https://www.zabbix.com/download?zabbix=5.0&os\_distribution=centos&os\_version=8&db=mysql&ws=apache

**Automatización**

**cron y systemd**

El servicio cron ha permitido ejecutar cada cierto intervalo de tiempo una tarea concreta. Esto es muy útil de cara a recopilar información o monitorizar el sistema realizando una tarea concreta y lanzar alertas como, por ejemplo, enviar un correo electrónico cuando la carga esté por encima de un valor determinado. cron es un servicio que corre en background. CentOS está reemplazando esto por systemd timers, porque cron tiene una versatilidad muy limitada, pero son complejos de configurar.

En /etc/cron tenemos una definición de directorios (en Ubuntu tienen una forma específica).

En /etc/crontab establecemos las reglas.

Como usuario no privilegiado también se pueden programar tareas periódicas con crontab -e, a nivel de usuario.

Importante: Para facilitarnos trabajar con expresiones de cron hay muchas páginas online, como crontab.guru.com

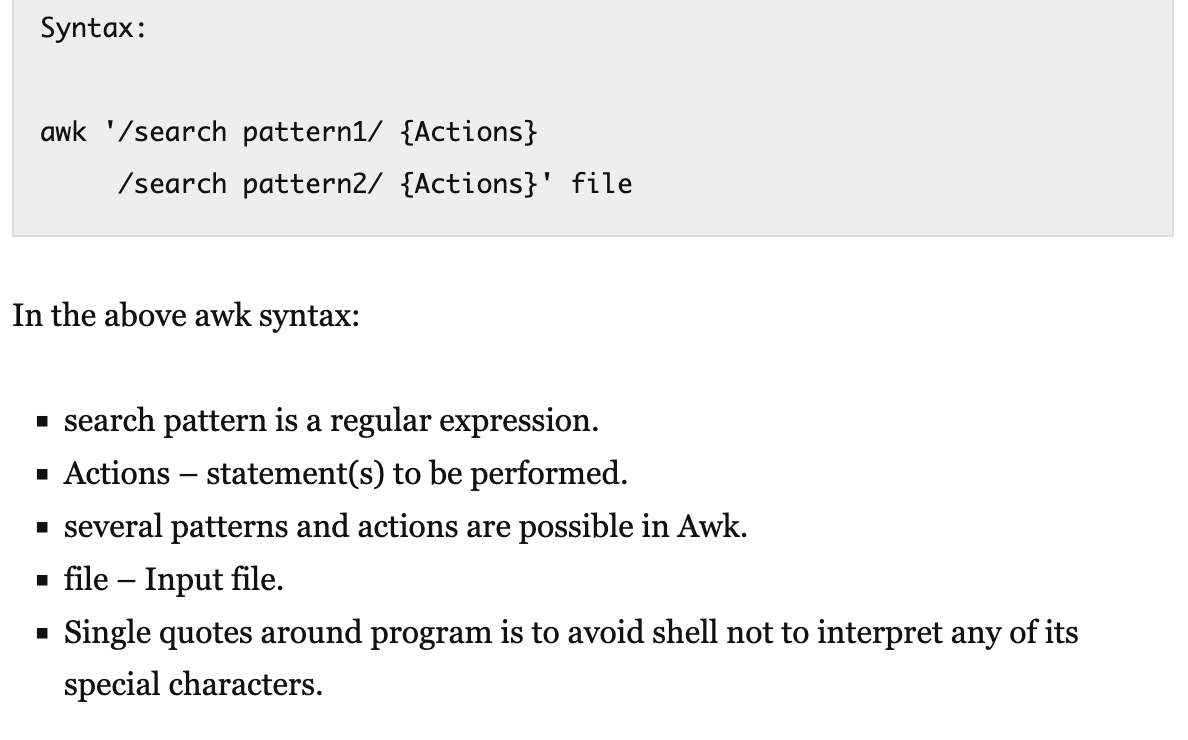
Para automatizar la ejecución de un script con systemd tenemos que definir un timer dentro del directorio /etc/systemd/system/ que se encarga de gestionar un servicio. Debemos crear dos archivos: uno archivo.timer y otro archivo.service. Para que pueda funcionar dicho timer, hay que activarlo de la siguiente forma: *systemctl enable archivo.timer* y *systemctl start archivo.timer.* Por último, ejecutamos el comando *journalctl -u archivo --since="yesterday".* (No tiene por qué ser yesterday, puede ser today o lo que sea).

**Scripts**

***Shell y comandos del sistema: grep, find, awk, sed.***

**Sed:** con sed puede buscar una cadena en un archivo y reemplazarla por otra, así podría dar acceso por contraseña durante unos instantes al servicio ssh para que los usuarios puedan copiar su llave pública. Ejemplos de uso: sed 's/Linux/Linux-Unix/' thegeekstuff.txt (s es para reemplazar)

**Awk:** puede programar la manipulación del texto así como generar salidas más completas a partir de la información en un archivo.



**Grep:** puede realizar filtrado de cadenas, útil cuando un archivo, listado, etc tiene unas dimensiones grandes, p.ej. ps -Af | grep firefox nos mostraría la información del pro- ceso firefox.

**Find**: puede buscar archivos y, una vez encontrados, realizar acciones sobre ellos, p.ej. *find /home/alberto/docs -name ’\*pdf’ -exec cp {} ~/PDFs \;* copiará todos los archivos cuyo nombre termine en pdf y los copia en la carpeta /home- /alberto/PDFs

***Python y php -> sí***

**A nivel de plataforma: Ansible (ver puidge final doc)**

**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.