Práctica 3

Parte Teórica:

1. Monitores Hardware

lspci -> indica los periféricos hardware conectados.
lshw -> muestra una descripción detallada de todo el hardware que hay en el ordenador
sensors -> muestra todos los sensores que pueda tener el equipo (En Alma)
dmesg -> muestra los mensajes que ha ido almacenando el kernel

2. Monitores Software

cat /proc/cpuinfo -> muestra información detallada sobre el procesador del sistema

cat /proc/meminfo -> muestra información detallada sobre la memoria del sistema

cat /proc/version -> muestra información detallada sobre la version del sistema

cat /proc/uptime -> muestra el tiempo que lleva el ordenador encendido

cat /proc/loadavg -> muestra la carga promedio de la cpu en diferentes tiempos

cat /proc/mdstat -> muestra la información sobre los RAID activos o configurados en el sistema

3. Monitores Generales

Existen varios como Munin, Nagios, Ganglia, etc. Pero nosotros trabajaremos con ZABBIX.

4. Automatización

cron -> Usado historicamente

systemd -> Es el estándar actualmente, y se divide en dos partes, en una describimos la tarea que debe ejecutar (script) y la segunda parte define cuándo se debe ejecutar.

Para los scripts comandos útiles podrían ser:

- grep -> Para filtrar resultados
- find -> Para realizar búsquedas
- awk / sed -> Para procesamiento de texto avanzado

Podemos usar para escribir los scripts PHP o Python.

5. Profiling

Se mide la eficiencia, velocidad y correcto funcionamiento para optimizarlo.

Parte Práctica:

En Debian:

En VirtualBox, y con la máquina apagada, activamos la opción “conectar en caliente” para los discos duros que tengamos en el apartado **Almacenamiento.**

Iniciamos la máquina y una vez dentro, vamos a simular que desconectamos uno de los discos (quitándolo desde VirtualBox).
Nos saldrá un mensaje diciéndonos que ha fallado uno de los discos pero el sistema seguirá funcionando.

Ahora vamos a simular lo mismo pero en frío, es decir, desconectamos el disco con la máquina apagada. Esta vez, vemos que al iniciar la máquina nos sale un error, sin embargo, nos pedirá la clave y funcionará con normalidad.

Todo esto es debido a que tenemos un RAID 1.

Ahora vamos a conectar un nuevo disco duro distinto al que habíamos desconectado, y vamos a intentar restaurar todo a como lo teníamos con el disco anterior, para ello deberemos usar fdisk para hacer las particiones necesarias:

- sudo fdisk /dev/sdb

Como usamos **MBR**, no tenemos que hacer una nueva tabla de particiones. Le damos a **n**, dejamos las opciones por defecto menos una que tenemos que ponerlo a ***+tamañoquesea*** y esta será la boot (para el tamaño nos hemos tenido que fijar en sda para ponerlo igual). Volvemos a usar **n**, para hacer otra partición pero esta vez lo dejamos todo por defecto para coger todo el espacio libre. Ahora con **t**, cambiamos cada partición a raid. Por último usamos **w** para guardar las particiones y salimos.

Para recuperar el RAID vamos a usar el comando **sudo mdadm –add /dev/md0 /dev/sdb1** y también lo hacemos para md1, **sudo mdadm –add /dev/md1 /dev/sdb2** . (Hacer una screenshot antes de ejecutar el comando)

Para la monitorización usamos **watch -n 1 cat /proc/mdstat** (-n nº se refiere a cada cuanto se actualiza, en este caso cada 1 segundo).

Con **sudi mdadm –detail /dev/md0**, podemos ver más detalles sobre el RAID md0.

El script que vamos a utilizar es:

```
import re
f=open( ' / p roc /mdstat ' )
for line in f :
        b=re.findall( r' \ [ [U] ∗ [_]+[U] ∗ \ ] ' , l i n e )
        i f ( b ! = [ ] ) :
                print ( "−−ERROR␣ en␣RAID−−" )
print ( "−−OK␣ S c ri p t −−" )
```

El script debemos guardarlo en el home, **touch mon-raid.py**, y copiamos el script en mon-raid.py.

Ahora en /etc/systemd/system **nano mon-raid.timer** y ponemos:

```
[ Unit ]
D e s c ri p ti o n=Monitor RAID s t a t u s
[ Timer ]
OnCalendar=minu tely
[ I n s t a l l ]
WantedBy=tim e r s . t a r g e t }
```

También tendremos que hacer **nano mon-raid.service** y ponemos:

[Unit] D e s c ri p ti o n=Monitor RAID s t a t u s
[S e r v i c e]
Type=sim pl e
E xecS ta r t=/u s r / bin / python3 /home/jpeiper/mon−r ai d . py
Ejecutamos el script con **python3 /home/jpeiper/mon-raid.py**, y nos debería devolver –OK, Script.
Debemos habilitar el timer con **sudo systemctl enable mon-raid.timer.**
Una vez habilitado podremos monitorizarlo con **journalctl -u mon-raid –since=”yesterday”**.

sudo mdadm –manage –set-faulty /dev/md0 /dev/sdb1, para ponerlo en fallo.