

Lección 1 - P3 ISE

Monitorización

Monitorizar significa vigilar la máquina mientras está en ejecución para poder analizarla. Es decir, vigilar los servicios que tiene activos.

Puede ser interna, si trata de vigilar desde la misma red. O externa si se usa una plataforma de un proveedor externo. La externa es mucho más fiable.

También existe la monitorización por hardware (ver características de dichos componentes) o la software.

Tipos de monitorización

Monitorización por logging = tiene que ver con la monitorización de los `logs` que estan en la carpeta `/var/log` (ELK STACK plataforma de las más usadas para este tipo de monitorización)

Monitorización de trazabilidad: siguen el movimiento del usuario dentro de las aplicaciones (ej: Google Analytics). De aquí se extraen datos importantes como por ejemplo: estadísticas de embudo → hasta donde llegan los usuarios (útil en marketing y para los ingenieros porque puedes ver si hay algún fallo para acceder a cierta sección de tu server)

Entre otros tipos de monitorización.

Profiling: consiste en análisis del rendimiento de una aplicación. Así podremos descubrir donde nuestro programa necesita optimización.

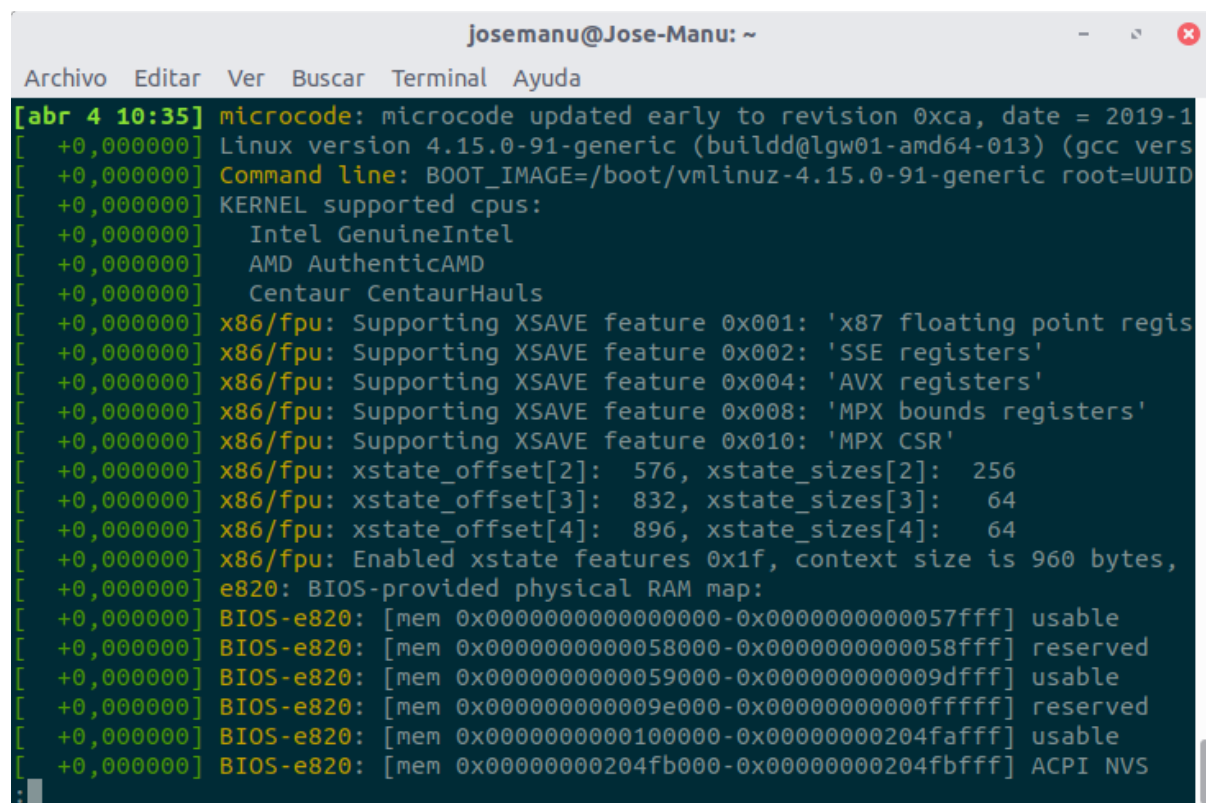
DMESG (*hardware*)

Comando para consultar los logs del kernel. Va a un buffer y hace volcados y borrados

En este ejemplo: lanzamos Ubuntu y hacemos ssh desde nuestra maquina (recomendado)

Ejecutamos `dmesg`, su uso es para depurar aspectos criticos del SO. Aquí vemos algunos errores que se pueden no mostrar durante el inicio de nuestra maquina

`-Hw` la H es para mostrar en horario humano y la w para que se quede activo en nuestro shell

A screenshot of a terminal window titled 'josemanu@Jose-Manu: ~'. The terminal shows the output of the 'dmesg' command with the '-Hw' flag. The output is color-coded: timestamps in green, kernel messages in yellow, and hardware-related messages in red. The messages include microcode updates, Linux version information (4.15.0-91-generic), command line details, kernel supported CPUs (Intel GenuineIntel, AMD AuthenticAMD, Centaur CentaurHauls), x86/fpu features (XSAVE, SSE, AVX, MPX), xstate offsets and sizes, and BIOS-e820 memory map details (usable, reserved, ACPI NVS).

ejemplo de ejecución

Con este comando corriendo en nuestra shell metemos un USB.



Puede ser que tengamos que seguir estos pasos en el caso de que nuestra VM no detecte el USB. 1) VM manger 2) configuración USB 3) añadir y te saldrá el nombre del USB que has conectado 4) click y confirmamos.

Veremos info importante sobre el USB que hemos introducido.

Logs de servicios y aplicaciones

El directorio `var` tiene información que varía en nuestro sistema Linux. Dentro encontramos `log` donde se guardan los documentos donde ver accesos, errores, etc.

`wtmp` → archivos binarios de seguridad

`syslog` → Son el lugar donde se registran propiamente los logs del sistema. Dentro podemos encontrar `rsyslog` que son los accesos remotos. Aquí echan los servicios sus logs.



Apache decidió tener sus propios logs. Otras aplicaciones o servicios pueden igualmente tomar esta alternativa

`syslog` es un servicio realmente (proceso) al que le mandan los servicios los logs y él los vuelca a memoria. Su configuración es compleja y en la actualidad se reemplaza por otros sistemas más dinámicos.

▼ Práctica habitual para estos archivos que son un chingo de grande (logs)

En linux `logrotate` coge los archivos y por fecha o tamaño entre otros, los rota. Rotarlo significa hacer una copia, ponerle un `.1` o `.fecha`. Si tienes muchos archivos el `logrotate` hasta te lo comprime en `.gz`

Config de logrotate → `/etc/logrotate.conf`

Rotación en resumen: una copia del archivo y empezar otro de nuevo.

Otros comandos útiles

`who` → es para ver los inicios de users en tu sistema

`last` → los últimos accesos y qué vainas hizo el usuario

`grep` → para buscar cosas en los archivos

Directorio /proc

Es un sistema de ficheros virtual volátil. Ahí se guardan los procesos que tenemos ejecutando. Ciertamente aquí no hay nada, son archivos y directorios que el sistema crea cuando tratamos de acceder a ellos.

`cpuinfo`: información de la CPU de nuestra máquina

`/proc/sys`: aquí hay directorios en los que guarda cosas



Ejemplo: accedemos dentro de este directorio a `/net/ipv4/ip_forward` (esto permite a tu máquina actuar como router). Cambiar en caliente ese documento con `echo 1>ip_forward`.



Otro ejemplo: `echo 'ise' > /proc/sys/kernel/hostname` esto es para apuntar un dominio a la ruta servidor que queramos (es como un DNS pero en local)



Podemos modificar cosas en caliente(máquina encendida), tocando documentos o con sysctl (ejemplo: `sysctl -w kernelctl.hostname=ise2`)

Si queremos estos cambios pa foreve recompilaremos `kernel` o en `etc/sysctl.conf`.



El arranque de linux se mete en este doc y se lo manda a sysctl para que lo active.

`watch` coge un documento que este cambiando, parametros `-n 2` (cada dos segundos) cat 'documento' y lo ejecuta cada numero de segundos que le pegues tras la n. (`watch -n 2 cat 'hola.txt'`)

`strace` (no entra)

Se encarga principalmente de ver las llamadas al sistema que realiza un programa o un servicio.

Monitores generales

Programas muy populares para monitorizar sistemas. Munin, Nagios, Ganglia, Cacti, AWstats y ZABBIX (que usaremos más adelante).

Tareas programadas con `cron`

Es un servicio estandar de linux usado para programar tareas que se van a ejecutar en un momento concreto. Común y muy utilizado.

Es un servicio que corre en background.

CentOS esta cambiando `cron` por `systemd/Timers`.

En `/etc/` podemos encontrar archivos relacionados con `crontab`. CentOS no lo hace así.

Dentro de `crontab` vemos la programación de cron (algo de esto hicimos en SO)

Para modificar `crontab` con algún editor de textos (nano) aunque usa `crontab -e` para hacer tareas a nivel de usuario (no `root`).



Ventajas `systemd/Timers`: granularidad y si la tarea no se puede ejecutar puedes decirle que otro comportamiento tomará



En [esta web](#) podemos hacer ejemplos de `crontab` de manera muy intuitiva para poder escribirlos directamente.

Monitorización del estado del Raid

▼ Ejercicio que consiste en modificar una de las maquinas y definir un Raid 1 y corromperlo. Desarrollaremos un monitor para ver que ocurre

Pinchamos un disco ms, `mdadm` creamos un RAID, despues de `/proc/` y con la info de `mdstat` que que almacena cosas del Raid como lo que vimos en la P1.

Programamos un script que compruebe `mdstat` y que si la vaina este bien nos Raid1 OK y Riad1 KO si nuestro Raid ha caido y lo enviamos al `syslog` del equipo con el comando `logger`.



`logger` envia a `syslog`

▼ Prueba rápida

`/etc` aqui vemos `nano -w crontab` (-w es para editar directamente) y añade una linea que cada minuto, hora, dia etc `root echo "HOLA" >&1 | logger -t isep311` (esto no lo hagamos nunca porque desde crontab es dificil depurar)

Entoncesse aconseja crear un archivo tipo bash shell y hacer

```
#!/usr/bin/env bash
echo "HOLA" >&1 | logger -t isep311
```

permisos de ejecucion (`chmod u+x 'archivo'`) y `tail -f syslog` en otra ventna del terminal.

ahora vamos y editamos el `crontab` desde el usuario nuestro (con `crontab -e`) esto es pa depurar los scripts fuera de cron, y despues lo metemos en cron.



Como simular que el RAID se ha corrompido????? → apagando pero no tenemos activo el `crontab` entonces usamos el comando `mdadm 'nombre del raid' --set-faulty 'disco que haremos que falle'` el cual sirve para marcar un Raid como fallido



`mdadm 'nombre del raid'--add 'nombre del disco a añadir'` pa añadir discooooooooo


```
josemanu@josemanuchido: ~
Archivo Editar Ver Buscar Terminal Ayuda
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab`
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
17 * * * * root    cd / && run-parts --report /etc/cron.hourly
25 6 * * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report
t /etc/cron.daily )
47 6 * * 7 root    test -x /usr/sbin/anacron || ( cd / && run-parts --report
t /etc/cron.weekly )
52 6 1 * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report
t /etc/cron.monthly )
* * * * * root    /home/josemanu/check_raid.sh

1,1 Todo
```

quedará algo así

6. Debería de estar funcionando, vamos a comprobarlo. De nuevo en otro terminal ejecutamos `tail -f /var/log/syslog`

```
Apr  5 18:42:01 josemanuchido CRON[3020]: (root) CMD (/home/josemanu/check_raid.sh)
Apr  5 18:42:01 josemanuchido ISEP3L1: Raid OK
Apr  5 18:43:01 josemanuchido CRON[3032]: (root) CMD (/home/josemanu/check_raid.sh)
Apr  5 18:43:01 josemanuchido ISEP3L1: Raid OK
Apr  5 18:44:01 josemanuchido CRON[3044]: (root) CMD (/home/josemanu/check_raid.sh)
Apr  5 18:44:01 josemanuchido ISEP3L1: Raid OK
```

vemos que el Raid está funcionando

7. Desactivamos uno de los discos de nuestra máquina → `mdadm /dev/md0 --set-faulty /dev/sdb1` (el disco que queremos desactivar en este caso)

8. Vamos a nuestro terminal con el comando `tail` en ejecución

```
Apr  5 18:49:01 josemanuchido CRON[3113]: (root) CMD (/home/josemanu/check_raid.sh)
Apr  5 18:49:01 josemanuchido ISEP3L1: Raid KO
Apr  5 18:50:01 josemanuchido CRON[3126]: (root) CMD (/home/josemanu/check_raid.sh)
Apr  5 18:50:01 josemanuchido ISEP3L1: Raid KO
```

ya no tenemos el Raid funcionando

Para recuperar el disco:

1. Lo quitamos → `mdadm /dev/md0 -r /dev/sdb1`
2. Y lo añadimos → `mdadm /dev/md0 --add /dev/sdb1`