Práctica 3

Monitorización, automatización y "profiling"

Esta práctica consiste en la utilización de herramientas de monitorización del sistema para visualizar cómo se comporta el sistema ante ciertas actividades que los usuarios u otros servicios generan. Estas herramientas presentan medidas del sistema permitiendo generar informes e históricos que puedan ser de utilidad para un análisis a posteriori o para tomar decisiones sobre la marcha.

Monitores para Hardware

Existen programas que nos permiten ver el estado hardware de la máquina.

1. En primer lugar, muchas BIOS (Basic Input Output System) nos permiten acceder a cierta información sobre el del HW (aunque ya están en extinción debido a UEFI, Universal Extended Firmwares).
2. Para no tener que reiniciar al querer usar la BIOS, tenemos otras herramientas como **hddtemp** y el proyecto **lm-sensors** (con su correspondiente GUI xsensors).
3. **lspci, lsusb, lshw** para listar HW de nuestro equipo.

El kernel de Linux permite conocer qué actividad ha ocurrido gracias a los mensajes que proporciona el kernel. Los podemos ver con **dmesg**.

Monitores para Software

Subsistema de archivos

En linux (UNIX), todo se manipula a través de archivos. Los directorios /proc y /var contienen archivos con información tanto del software como del hardware. Se encuentran archivos fundamentales en la monitorización del comportamiento del sistema, de las aplicaciones y de los usuarios. El subdirectorio /var/log/ contiene los archivos en los que se vuelcan los logs de los servicios y algunas aplicaciones.

Una mala gestión de estos archivos puede derivar en un sistema caído. La rotación de los archivos de log (logrotate) se hace para evitar que los logs crezcan indefinidamente.

En esta web se profundiza más sobre los logs y logrotate: LOGS Y LOGROTATE

Estos archivos nos ayudan a identificar errores y problemas, además esta tarea puede ser automatizada.

Con **mdadm y /proc/mdstat** podemos identificar, monitorizar y reconstruir los discos RAID que tenemos en nuestras máquinas virtuales.

/proc/mdstat

A continuación se muestra un ejemplo de un mensaje de estado cuando ambos discos están disponibles y montados correctamente:

```
[root@host ~]# cat /proc/mdstat
```

```
Personalities : [raid1]
read_ahead 1024 sectors
md2 : active raid1 sda3[1] sdb3[0]
262016 blocks [2/2] [UU]

md1 : active raid1 sda2[1] sdb2[0]
119684160 blocks [2/2] [UU]

md0 : active raid1 sda1[1] sdb1[0]
102208 blocks [2/2] [UU]

unused devices: <none>
```

El ejemplo muestra tres arrays RAID (md0, md1, md2). En la segunda línea de cada array se muestra el estado de las particiones individuales entre corchetes. Una **U** significa que el dispositivo respectivo funciona correctamente (up).

En el caso de que falte un disco (en este ejemplo, sdb), se muestra un mensaje como el siguiente:

```
cat /proc/mdstat
```

```
Personalities : [raid1]
read_ahead 1024 sectors
md0 : active raid1 sda1[1]
102208 blocks [2/1] [_U]

md2 : active raid1 sda3[1]
262016 blocks [2/1] [_U]

md1 : active raid1 sda2[1]
119684160 blocks [2/1] [_U]

unused devices: <none>
```

La combinación **[_U]** significa que algo no está funcionando correctamente.

mdadm

El comando **mdadm -D /dev/md1** proporciona información detallada sobre un dispositivo RAID.

A continuación se muestra un ejemplo de un RAID en funcionamiento:

```
[root@host ~]# mdadm -D /dev/md1
```

```
/dev/md1:
Version : 0.90
Creation Time : Tue Sep 11 21:33:24 2012
Raid Level : raid1
Array Size : 4194240 (4.00 GiB 4.29 GB)
Used Dev Size : 4194240 (4.00 GiB 4.29 GB)
Raid Devices : 2
Total Devices : 2
Preferred Minor : 1
Persistence : Superblock is persistent

Update Time : Wed Jun 10 11:11:05 2015
State : clean
Active Devices : 2
Working Devices : 2
Failed Devices : 0
Spare Devices : 0

UUID : 2db35cef:bde058e8:1f51fb89:78ee93fe
Events : 0.251

Number Major Minor RaidDevice State
0 8 1 0 active sync /dev/sda1
1 8 17 1 active sync /dev/sdb1
```

Si falta un disco, se muestra una información similar a la siguiente:

```
[root@host ~]# mdadm -D /dev/md0
```

```
/dev/md0:
Version : 00.90.00
Creation Time : Thu Aug 21 12:22:43 2003
Raid Level : raid1
Array Size : 102208 (99.81 MiB 104.66 MB)
Device Size : 102208 (99.81 MiB 104.66 MB)
Raid Devices : 2
Total Devices : 1
Preferred Minor : 0
Persistence : Superblock is persistent

Update Time : Fri Oct 15 06:25:45 2004
State : dirty, no-errors
Active Devices : 1
Working Devices : 1
Failed Devices : 0
Spare Devices : 0


Number Major Minor RaidDevice State
0 0 0 0 faulty removed
1 3 1 1 active sync /dev/sda1
```

Recursos interesantes

- Detecting, quering and testing
- /proc/mdstat

Monitorizando un servicio o ejecución de un programa: strace

Para detectar problemas que no se muestran en los archivos de log.

Estos programas permiten hacer un traza de las llamadas al sistema realizadas por un programa (servicio) en ejecución, como es el caso de **strace**.

Ejemplos de uso **strace**

Monitores generales

Munin

En Ubuntu está disponible sin añadir ningún repositorio adicional. En CentOS, hay un paquete disponible en el repositorio EPEL(click para ver como activar el repositorio), después podremos usar dnf para instalar Munin.

Aquí tenemos una demo de Munin corriendo.

Nagios y Naemon

Nagios ha pasado a ser proyecto empresarial, por eso ha surgido Naemon (demo, uso de la interfaz Thruk).

Ganglia

Monitoriza sistemas de cómputo distribuidos (normalmente para altas prestaciones) y permite una gran escalabilidad.

Zabbix

Nos centraremos en él en las lecciones. Para su instalación solo hay que añadir los repositorios e instalarlo con el correspondiente gestor de paquetes. Aquí encontramos toda la información para el seguimiento de las lecciones.

El objetivo es que seamos capaces de instalar este sistema de monitorización y configurar este para que se monitorice a sí mismo así como a CentOS (a través del agente) además de conocer cómo interactuar con el sistema de monitorización usando la API que proporcionar y conocer en qué consiste el protocolo SNMP.

Cacti

Cacti es otro front-end para RDRtool, que permite monitorizar muchos parámetros sin sobrecargar excesivamente el sistema.

AWstats

Monitor de servicios específicos de servidores web HTTP, FTP, correo, etc. Se puede compilar el código fuente aunque están disponibles los paquetes en los repos.

Automatización

cron y systemd

El servicio cron ha permitido ejecutar cada cierto intervalo de tiempo una tarea concreta. Con la introducción de systemd, cron puede seguir siendo usado pero su funcionamiento está basado en los timers que acrtivan services. Documentación de RedHat.

Podemos automatizar la ejecución de este script (ejemplo del pdf de SWAD) en systemd definiendo un timer dentro del directorio /etc/systemd/system/ que se encarga de gestionar un servicio. Por tanto, hemos de crear dos archivos: mon_raid.timer y mon_raid.service
[Unit]
D e s c r i p t i o n=Monitor RAID s t a t u s

[Timer]
OnCalendar=m i n u t e l y

[Install]
WantedBy=t i m e r s . t a r g e t }

[Unit]
D e s c r i p t i o n=Monitor RAID s t a t u s

[Service]
Type=s i m p l e
E x e c S t a r t=/u s r / b i n / python3 /home/ a l b e r t o /mon−r a i d . py

Para que pueda funcionar hemos de activar y habilitar el timer (de manera análoga a lo que hicimos con los servicios tras instalarlos en la Práctica 2). Una vez hecho esto, podremos monitorizar su ejecución gracias al comando journalctl, p.ej. journalctl -u mon-raid --since="yesterday".

Scripts

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

Existen cuatro comandos muy útiles (ejemplos) para la automatización de tareas: grep, find, awk y sed.

Con **sed** podemos buscar una cadena en un archiva y reemplazarla por otra.

Con **awk** podemos programar la manipulación de texto así como generar salidas más completas a partir de la información en un archivo.

Con **grep** podemos realizar filtrado de cadenas (ps -Af | grep firefox).

Con **find** podemos buscar archivos y una vez encontrados realizar acciones sobre ellos (find /home/alberto/docs -name '*pdf' -exec cp {} ~/PDFs \;).

Python y PHP

Mirar PDF prácticas.

A nivel de Plataforma: Ansible

Existen interfaces que permiten programar scripts y visualizar su ejecución de una manera cómoda y visual. Veremos una lección con el funcionamiento básico de Ansible

Profiling

Scripts

Para estudiar el comportamiento de los scripts, independientemente del lenguaje que usemos, podemos usar varios profilers por ejemplo, en Bash, podemos usar la opción set -x y modificar la variable local PS4 para que muestre el tiempo.

En el caso de PHP tenemos XHProf (aquí tutorial de cómo realizar el profiling).

Por último, en el caso de Python, se pude utilizar el módulo eProfile.

SQL

El MySQL y MariaDB que instalamos en la práctica anterior tienen un "profiler" para analizar cuánto tardan las consultas. Son útiles también las herramientas **MySQLWorkBench** y comandos como **mysqloptimize**, que optimizan la ejecución.