

# Administración Linux

# Administración de Sistemas

Unai Lopez Novoa  
unai.lopez@ehu.eus



Universidad  
del País Vasco Euskal Herriko  
Unibertsitatea

# Intérprete de comandos (Shell)

- Interfaz del sistema entre usuario/aplicaciones y las llamadas al sistema operativo
- Es un programa del sistema operativo con los mismos privilegios que cualquier otro
- Símbolo de uso:
  - \$                    usuario normal
  - #                    superusuario / administrador / root
- Tipos de acceso:
  - Local:            Terminales de texto (Ctrl+Alt+F1...6)  
                      y una terminal gráfica (Ctrl+Alt+F7)
  - Remoto:            A través de la red: telnet, rlogin, ssh, ...



# Intérprete de comandos (Shell)

- Múltiples Shell en los sistemas Unix/Linux, p.e.:

bash Bourne Again Shell,

csh C Shell

ksh KornShell

tcsh Tenex Csh, tiene mejoras sobre csh

zsh Extensión de bash con características de ksh y tcsh  
*zsh es la shell por defecto desde macOS 10.15*

- Es importante conocer cuál estamos usando



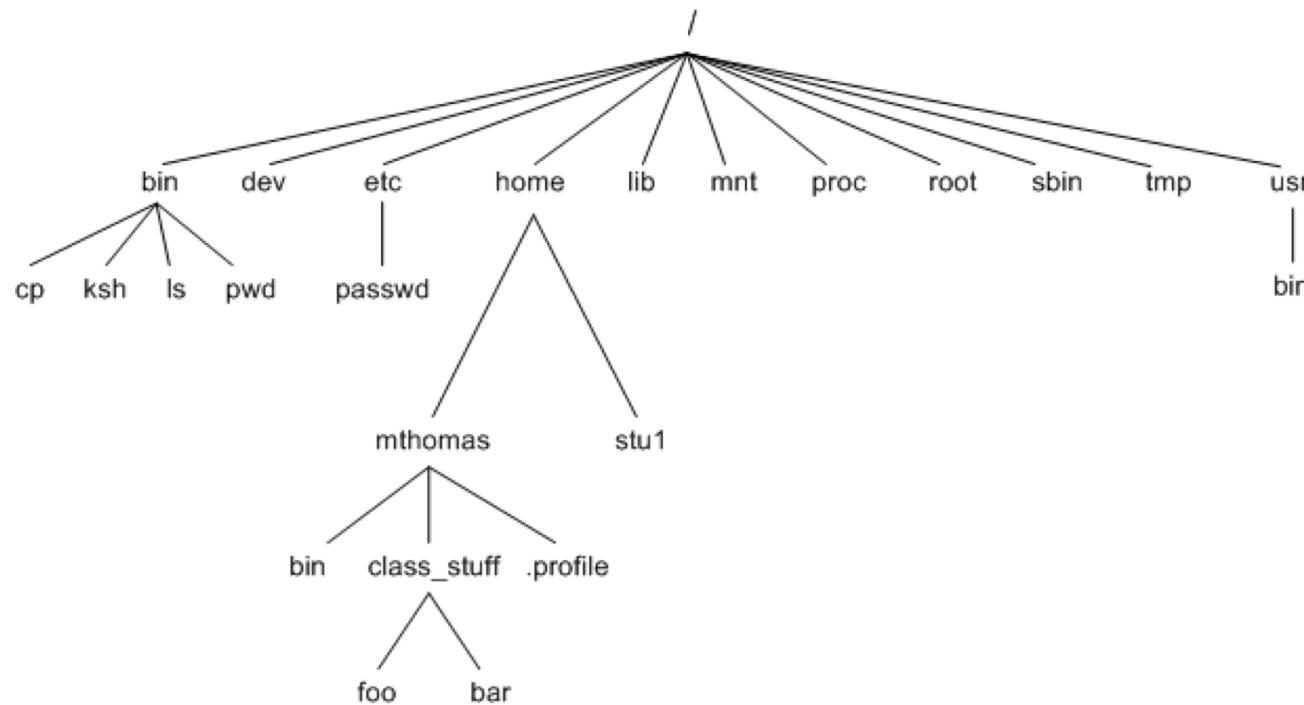
# Intérprete de comandos

- Comando **man**
  - Debería ser el primer comando a aprender
  - Muestra información sobre un comando, función, ...
- Sintaxis: **man <comando>**
- Los manuales se encuentran en **/usr/share/man**
  - Normalmente se busca alfabéticamente y muestra la 1<sup>a</sup> entrada que coincide
- Configuración en **/etc/manpath.config**



# Sistema de ficheros

- Jerarquía habitual en sistemas Unix o Linux



# Sistema de ficheros

- Estructura de árbol
- Formas de acceso:
  - Absolutas: /home/unai/miArchivo.txt
  - Relativas a la ruta actual: ../../miCarpeta/miArchivo.txt
- Los archivos ocultos comienzan su nombre con “.”
- Unidades externas (CDs, HDDs externos, ...):
  - Se pueden asociar a cualquier posición de la jerarquía
  - Un mismo programa puede tratar ficheros internos y dispositivos externos de manera indiferente



# Sistema de ficheros

/	Directorio raíz.
/bin	Comandos básicos del sistema operativo.
/boot	Kernel y archivos necesarios para su carga.
/dev	Dispositivos: discos, impresoras, ....
/etc	Archivos para inicio y configuración del sistema.
/mnt	Puntos de montaje temporales.
/lib	Librerías compartidas.
/home	Directorios raíz por defecto de los usuarios.
/opt	Paquetes de software opcional (no siempre)
/root	Directorio raíz para el superusuario.
/sbin	Comandos críticos del sistema operativos.
/proc	Información de procesos en ejecución.
/tmp	Archivos temporales.
/usr	Recursos y software de los usuarios.
/usr/local	Software instalado por los usuarios.
/var	Datos específicos y configuración del sistema.



# Sistema de ficheros

- Comando ls: listar archivos
  - Uno de los más utilizados
  - Algunos parámetros (combinables entre sí)
    - l      Mostrar información adicional de cada archivo  
(fecha, tamaño, permisos, ...)
    - a      Mostrar todos los archivos, incluidos los ocultos
    - r      Listar en orden inverso
    - t      Listar en orden cronológico ascendente
    - h      Mostrar tamaños de ficheros en KB o MB en vez de bytes
- P.e. ls -larth muestra toda la información de los archivos de un directorio ordenados de manera cronológica y con tamaños en KB o MB



# Sistema de ficheros

- Comandos para navegar por el sistema de ficheros:
  - `pwd` Mostrar directorio actual.
  - `cd` Cambiar directorio.
  - `mkdir` Crear directorio.
- Manipulación de archivos
  - `cp` Copiar archivos
  - `mv` Mover archivos (o renombrar)  
*Ejemplo: mv <ruta-fichero-origen> <destino>*
  - `rm` Borrar archivos (o directorios con el parámetro `-r`)
  - `ln` Crear enlaces a archivos (parámetro `-s` para simbólico)  
*Ejemplo: ln -s <ruta-fichero-origen> <ruta-enlace>*
  - `whereis` Ubicación de un comando
  - `find` Buscar archivos  
*Ejemplo: find <carpeta-base> -name <nombre-archivo>*



# Sistema de ficheros

- Contenido de archivos
  - cat/more/less Mostrar contenidos de un archivo
  - wc Contar palabras (o líneas con el parámetro -l)
  - head/tail Mostrar N primeras/últimas líneas (utilizando -n)
  - grep Buscar patrón de texto en una archivo  
*Ejemplo: grep casa miArchivo.txt*
  - cut Muestra secciones concretas de un archivo  
*Ejemplo: cut -d " " -f2 a.txt*  
*Lee a.txt, separado por " ", y devuelve la 2ª columna*
  - tar Comprimir/descomprimir archivos/carpetas  
*Comprimir: tar cfvz carpeta.tgz miCarpeta*  
*Descomprimir: tar xfvz carpeta.tgz*
  - sort Ordena las líneas de un archivo alfabéticamente



# Atajos

- Autocompletado de Shell
  - Utilizar la tecla tabulador para autocompletar nombres/rutas
  - Si hay varias posibles opciones, las muestra antes de completar
- Expresiones regulares con caracteres comodín (wildcards)
  - Algunos caracteres se usan con fines especiales:
    - \* Reemplazar todos los caracteres
    - ? Reemplazar un único carácter
    - [] Reemplazar por un rango numérico, p.e., [12]
    - ~ Atajo para el directorio raíz del usuario (\$HOME)
  - Si hiciera falta buscar un carácter de estos en un archivo, escribirlo precedido por \ o rodeado por ""



# Atajos: historia de comandos

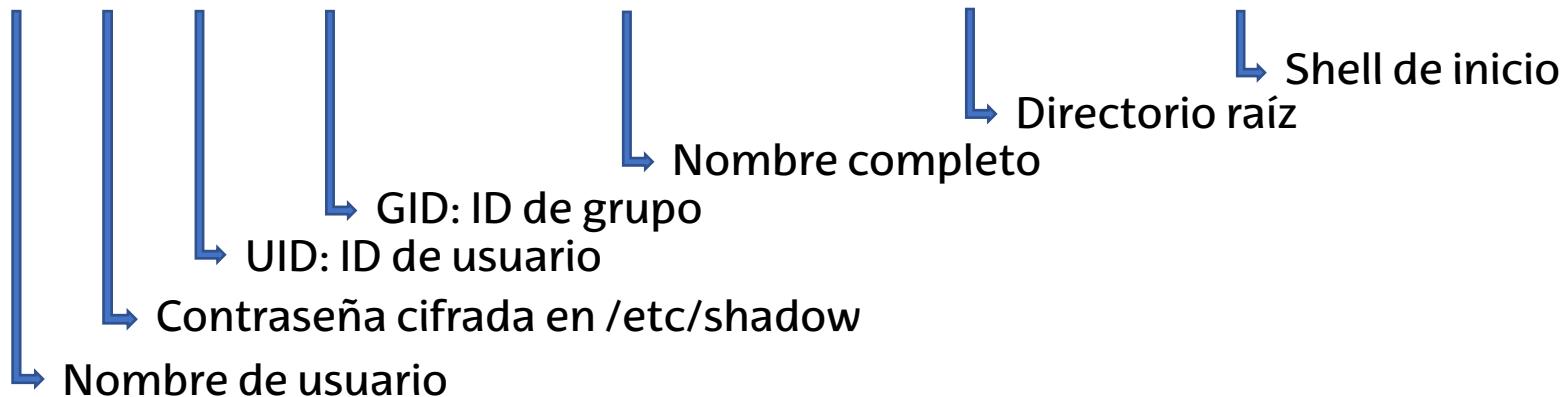
- La Shell guarda el histórico de comandos que hemos tecleado, se puede visualizar con **history**
  - Una vez mostrado el histórico, se puede volver a ejecutar un comando escribiendo !<nº de comando>
  - P.e. si el nº 20 ha sido **ls**, escribir **!20** ejecutará **ls** de nuevo
  - **!!** ejecuta de nuevo el último comando
- Con los cursores arriba/abajo se navega por la historia más reciente
- Pulsar **Ctrl+r** y empezar a escribir para que la Shell autocomplete comandos históricos
  - Utilizar **Ctrl+r** en este modo para navegar por las opciones



# Usuarios

- En Unix, los usuarios están organizados en grupos
- Los ficheros `/etc/passwd` y `/etc/group` contienen la información sobre los usuarios y grupos del sistema
  - Cada línea tiene diferentes campos separados por el carácter :
  - Ejemplo de línea:

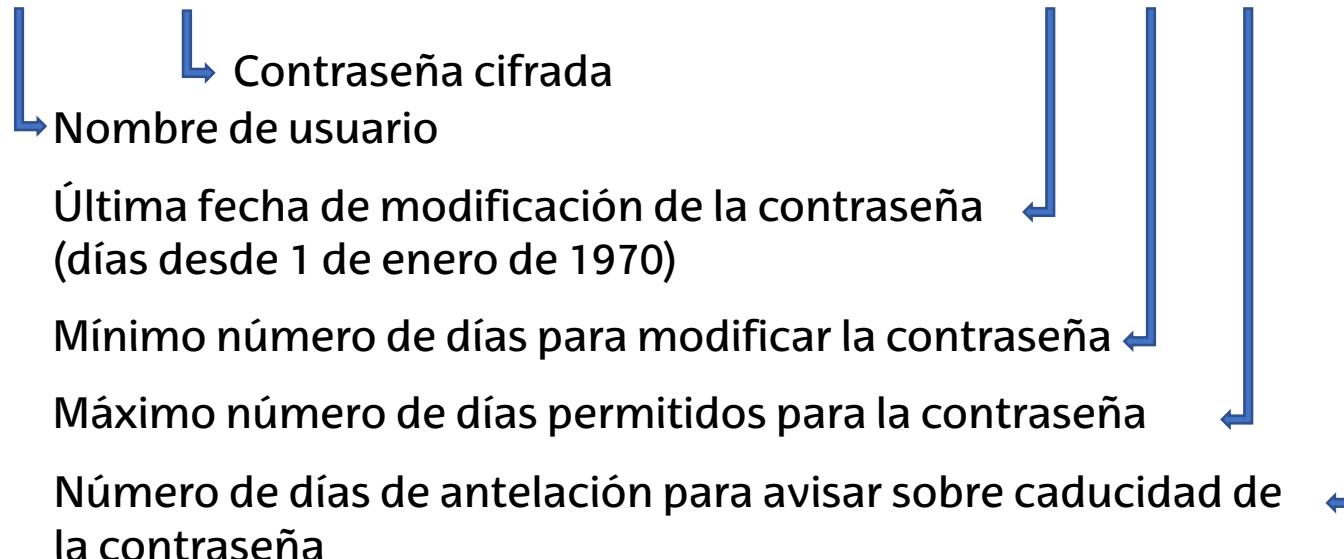
```
unai:x:1011:1011:Unai Lopez Novoa:/home/unai:/bin/bash
```



# Usuarios

- El fichero `/etc/shadow` contiene contraseñas del sistema
  - Hace falta permisos root para leerlo
- Cada línea tiene la siguiente forma:

unai:\$MZdY3ECAD6tFmEtYAIOSdnqidnqwdibb1:18148:0:99999:7::::



# Usuarios

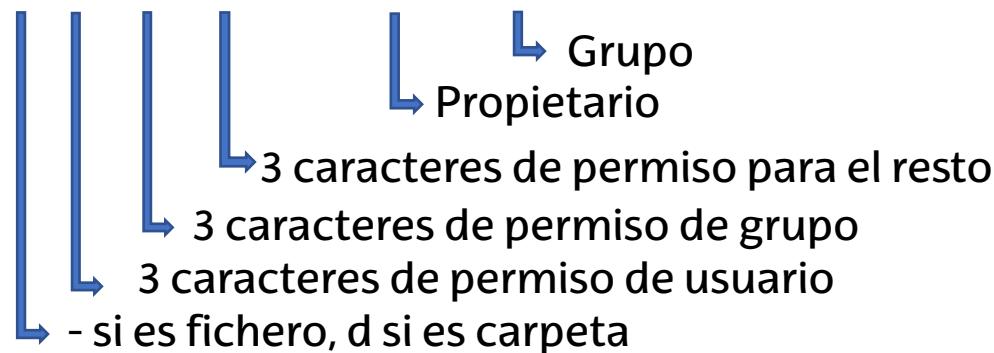
- Unix gestiona la seguridad del sistema de ficheros con usuarios, grupos y permisos para cada uno.
- Cada fichero (y carpeta) tiene un único propietario y unos permisos de acceso.
- Los permisos son:
  - r              Lectura: permite leer el contenido
  - w              Escritura: permite modificar el contenido
  - x              Ejecución: permite la ejecución de un fichero
- Cada permiso se configura para:
  - usuario: El propietario del fichero
  - grupo: Usuarios del grupo al que pertenezca el propietario
  - resto: Resto de usuarios del sistema



# Usuarios

- Generalmente, los usuarios sólo tienen permisos de escritura en su directorio raíz (p.e. /home/<usuario>)
  - Y en algunos directorios temporales como /tmp
- El super-usuario (o sysadmin) tiene acceso todo el sistema de archivos
- Ejemplo de permisos de un archivo (con ls -l):

```
-rw-rw-r-- 1 unai unai 21 jul 3 12:59 a.txt
```



# Usuarios

- Comandos básicos de gestión de usuarios

whoami	Muestra el nombre del usuario actual
who	Muestra los usuarios conectados al sistema
w	Equivalente a who, algo más de información
passwd	Cambiar contraseña del usuario actual
write	Escribir un mensaje otro usuario
useradd	Crear usuario en el sistema
adduser	Script “asistente” para crear un usuario

- Gestión de permisos

chmod	Modificar permisos de un fichero o carpeta
chown/chgrp	Modificar UID/GID de un fichero



# Variables de entorno

- Sirven para guardar valores en una sesión de Shell
- Para leer el contenido: echo \$VARIABLE
  - P.e. podéis probar a ejecutar echo \$HOME
- Hay de 2 tipos:
  - Del usuario:
    - Se mantienen durante la sesión activa, después se eliminan
    - Se listan con el comando **env**
    - P.e. \$HOME, \$LD\_LIBRARY\_PATH
  - Del sistema:
    - Están en todas las sesiones del sistema
    - Se listan con el comando **set**
    - P.e., \$BASH, \$HOSTNAME



# Variables de entorno

- Depende de la Shell, las variables de entorno se crean de manera diferente
- En una Shell tipo sh
  - P.e. Bash, ksh
  - Comando **export**, p.e. export MIVARIABLE=4
- En una Shell tipo csh
  - P.e. tcsh, zsh
  - Comando **setenv**, p.e. setenv MIVARIABLE 4
- Estas variables desaparecen al finalizar la sesión



# Variables de entorno

- Algunas variables de entorno comunes

\$PATH	Listado de directorios donde están los binarios de los comandos. Al ejecutar un comando (p.e. ls), se busca aquí.
\$HOME	Directorio raíz del usuario actual.
\$TERM	Tipo de terminal utilizado para conectar al sistema.
\$SHELL	Tipo de Shell de la sesión, p.e. Bash.
\$PWD	Directorio actual



# Variables de entorno

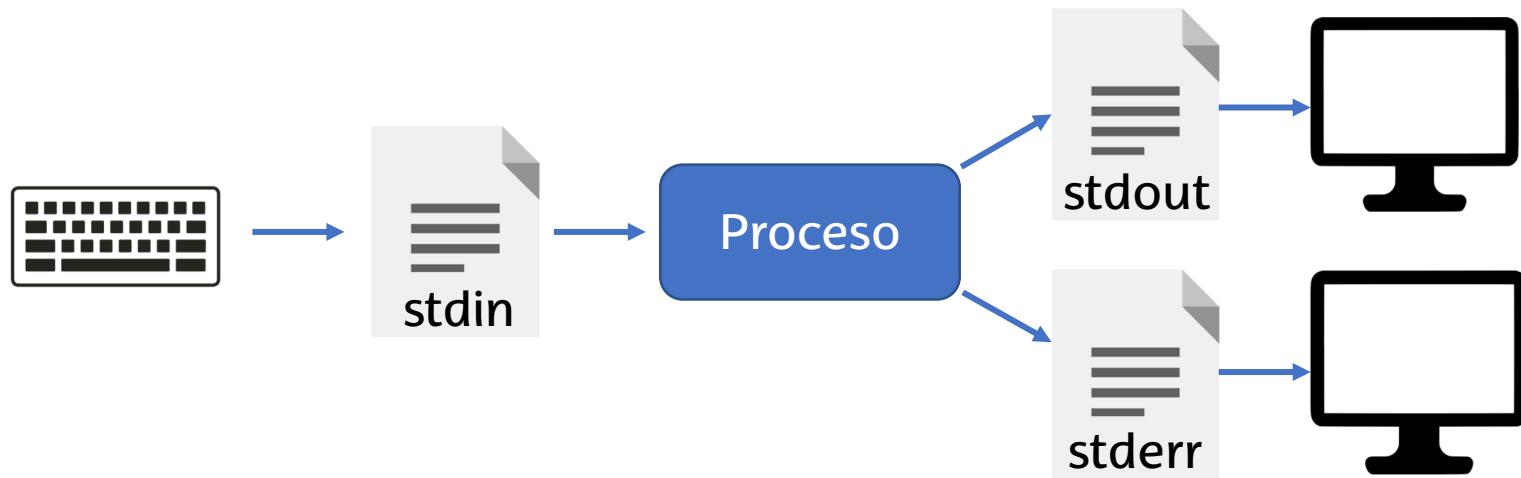
- Hay unos ficheros concretos en el sistema que permiten configurar la Shell
- Podemos definir variables de entorno en ellos para que estén siempre disponibles en nuestras sesiones
- En Bash hay varios, en orden de relevancia:  
`/etc/bashrc → /etc/profile → $HOME/.bashrc → $HOME/.bash_profile`
- Igualmente en zsh:  
`/etc/zshrc → /etc/profile → $HOME/.zshrc → $HOME/.zprofile`



# Redirección y Pipes

En Unix, cada proceso tiene asociado por defecto 3 ficheros que sirven para gestionar entrada y salida:

- `stdin` Teclado (entrada estándar)
- `stdout` Pantalla (salida estándar)
- `stderr` Pantalla (salida de errores)



# Redirección y Pipes

- **Redirección:** utilizar un fichero en lugar del teclado/pantalla para la entrada/salida
- Redirección de **stdin**: utilizar un fichero en lugar del teclado para la entrada de datos  
Ejemplo: sort < a.txt



# Redirección y Pipes

- Redirección de **stdout**:

- `cat a.txt > b.txt` Abre/Borra b.txt y escribe la salida de cat
- `cat a.txt >> b.txt` Añade la salida de cat al final de b.txt

- Redirección de **stderr**:

- `cat a.txt 2> b.txt` Escribe en b.txt los errores que genera cat

- Redirección de **stdout** y **stderr**:

- `cat a.txt &> b.txt` Escribe en b.txt la salida de cat y sus errores



# Redirección y Pipes

- Un **Pipe** (“tubería”) permite redirigir la salida de un comando a la entrada de otro
  - Ejemplo: cat a.txt | grep casa | wc -l
  - Contaría el número de líneas de a.txt que contienen “casa”
- Además, en Bash se puede **concatenar** la ejecución de varios comandos seguidos
  - Ejemplo: ls -l; cd ..; ls -l
  - Alternativamente: ls -l && cd .. && ls -l



# Ejercicios 1

- Mostrar los nombres de los usuarios del sistema
  - Pista: 1<sup>a</sup> columna del fichero /etc/passwd
- Mostrar la información de los usuarios que tengan bash como Shell de inicio
  - Pista: esta información también está en /etc/passwd
- Crear un enlace simbólico a /bin/bash en /tmp llamado "miBash"
- Contar el número de líneas del fichero .bashrc en vuestro directorio raíz de usuario



# Ejercicios 2

- Mostrar el usuario del sistema que cambió la contraseña hace más tiempo
  - Pista: la información está en /etc/shadow
- Crear una variable de entorno CIUDAD persistente con el nombre de una ciudad.
  - Cerrar y volver a abrir sesión en el sistema, comprobar que existe.
- Guardar en un fichero las 4 últimas columnas resultantes de llamar al comando “ping www.ehu.eus”
  - Utilizar el parámetro “-c” de ping para limitar el número de iteraciones.



# Shell scripting

- Un guion de Shell (Shell script) es un conjunto de comandos que se ejecutan con un objetivo concreto.
- Su forma más simple: un fichero de texto con un comando por línea.

- Ejemplo:

```
#!/bin/bash  
echo "Hola"  
echo "Mundo"
```

La 1<sup>a</sup> línea se llama Shebang,  
indica la ruta del Shell a usar

- Para ejecutar un script:

- Si el script no tiene permisos:     *bash miScript*
- Alternativamente, modificar los permisos del script y ejecutarlo como un binario:     *chmod +x miScript; ./miScript*



# Shell scripting

- Entrada/salida en un script

`read` Para leer del teclado

`echo/printf` Para escribir por pantalla (stdout)

- Un script puede recibir parámetros

- Los parámetros que reciba se convierten en variables:

`$1 .. 9` Los parámetros, el número indica su posición

`$0` Contiene el nombre del script

`$#` Número de parámetros recibidos

`$?` El PID del proceso que está ejecutando el script

`$*` Contiene todos los parámetros ( $\$1 + \$2 + \dots$ )



# Shell scripting

- Control de flujo
  - Sentencia if - else
    - en /bin/bash              if [[ <condición> && <condición> ]]; then
    - en /bin/sh                if [ <condición> ] && [ <condición> ]; then
  - Ejemplo

```
#!/bin/bash
if [ `whoami` = "unai" ]; then
    echo "El usuario actual es unai"
else
    echo "El usuario actual no es unai"
fi
```

Comillas `` para  
ejecutar un comando

Atención al "fi" para  
cerrar el bloque "if"



# Shell scripting

- Control de flujo
  - Operadores de comparación

Strings	Numéricos	Verdadero si
x = y	x -eq y	x es igual a y
x != y	x -ne y	x es diferente a y
x < y	x -lt y *	x es menor que y
	*: -gt, -le, -ge	equivalentes: >, <=, >=

Operador	Verdadero si
-d carpeta	Carpeta existe
-e fichero	Fichero existe
-r fichero -w fichero	Usuario tiene permiso de lectura / escritura en el fichero
-s fichero	Fichero existe y no está vacío

- Sentencia case

```
usu=$1
case $usu in
    "unai") echo "El usuario es Unai";;
    "mikel") echo "El usuario es Mikel";;
    "jon")   echo "El usuario es Jon";;
esac
```



# Shell scripting

- **Bucles**

- Se puede utilizar el bucle **for** para iterar sobre (entre otros):

- Archivos:

```
for archivo in `ls`; do  
    echo $archivo;  
done
```

- Número o elementos concretos:

```
for i in 1 3 5 9 11; do  
    echo $i;  
done
```

- Secuencias de números:

```
for i in {1..8};do  
    echo $i;  
done
```



# Shell scripting

- Variables

- Todas las variables en Bash se consideran Strings
- Se puede hacer aritmética con variables Bash

- Utilizar `(( ))`
- Ejemplo:

```
a=1  
b=3  
c=$a+$b  
d=$(( $a+$b ))  
echo $c $d
```

Concatenación de strings

Suma aritmética

- Se pueden crear listas de variables

- Utilizar `list`
- Ejemplo:

```
aa=11  
bb=22  
cc=33  
list=(aa bb cc)  
  
echo ${list[@]}  
echo ${list[0]}
```

Muestra todos los elementos

Muestra el 1er elemento



# Shell scripting

- **Expresiones regulares**

- Utilizadas para buscar texto que corresponda a un patrón
- Un patrón se construye con literales y caracteres especiales
- Ejemplos de patrones:

- “casa”
  - Encontraría “casa”
- “c([a-z])sa”
  - “casa”, “cbsa”, “ccsa”,...
- “c([a-z]+)sa”
  - “casa”, “caasa”, “cxyzsa”
- “c(\d\*)sa”
  - “csa”, “c101sa”, ...

Símbolo	Significado
.	Cualquier carácter
[]	Caracteres definidos entre []
\d	Cualquier dígito
*	Una, ninguna o más del elemento precedido
+	Uno o más del elemento precedido

- Un listado exhaustivo de caracteres especiales:

<https://cheatography.com/davechild/cheat-sheets/regular-expressions/>



# Shell scripting

- Funciones

- Se pueden crear funciones para ordenar el código
- Se le pueden pasar parámetros
  - Se recogen en la función como \$1, \$2, ... en base al orden
- Para devolver un valor de la función, utilizar **echo** y no **return**
  - return se interpreta como el comando de finalizar el programa

- Ejemplo:

```
function suma ()  
{  
    OP1=$1  
    OP2=$2  
    echo $((OP1+OP2))  
}  
  
A=33  
B=44  
res=$(suma $A $B)  
echo "La suma de $A y $B es $res"
```



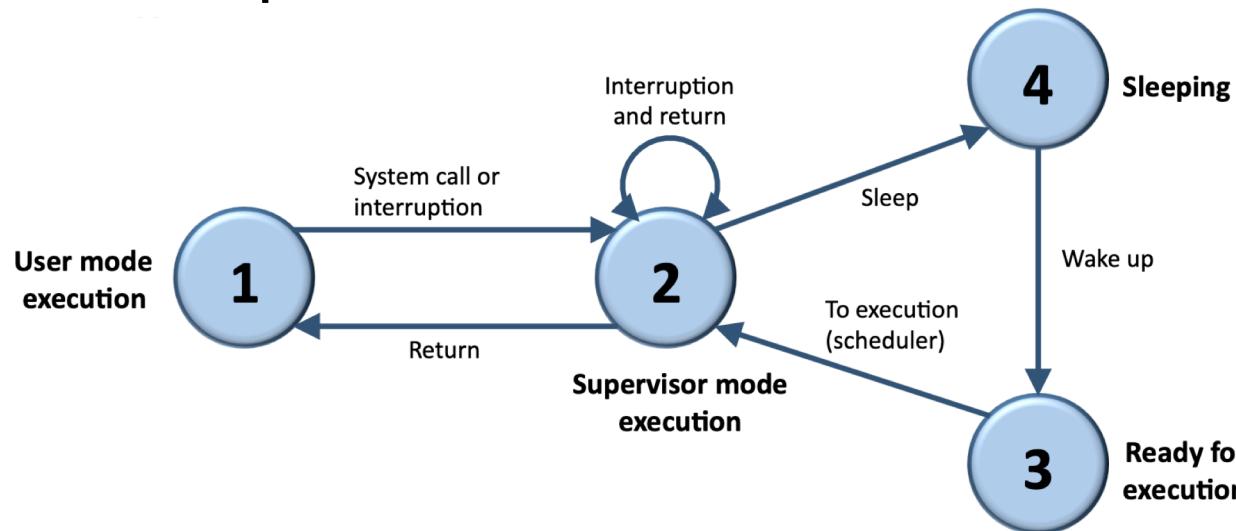
# Ejercicios 3

- Hacer un script que muestre los nombres de todos los archivos en /tmp usando un bucle
- Extender el script anterior haciendo que, si /tmp está vacío, cree un archivo llamado vacio.txt en /tmp
- Hacer un script que implemente una calculadora
  - El script debe recibir 2 parámetros numéricos
  - El script debe mostrar una lista de las siguientes operaciones y pedir al usuario que introduzca la operación deseada
    - Suma, resta, multiplicación, división
  - Cada operación debe implementarse como una función



# Gestión de procesos

- **Proceso:** secuencia de instrucciones y datos almacenados en memoria
- Se identifican con un número único: PID
- Estados de un proceso:



# Gestión de procesos

- En Unix, los procesos se organizan en árbol
- El proceso raíz es **init**
  - Cada proceso (excepto init) tiene un padre
  - El kernel tiene control de todos los procesos del sistema
- Para cada proceso, su PID se identifica con el UID del usuario al que le pertenece
- En cada sesión, podemos lanzar procesos en primer y segundo plano
  - Primer plano: la terminal se bloquea hasta que el proceso termina
  - Segundo plano: la terminal devuelve el control inmediatamente



# Gestión de procesos

- Hasta ahora, hemos lanzado todo en primer plano:
  - Ejemplo: `./miScriptLargo`
- Para lanzar un proceso en 2º plano, añadir & al final
  - Ejemplo: `./miScriptLargo &`
- Por otra parte, el comando **nohup** mantiene un proceso vivo aunque se cierre la sesión
  - Nohup significa *No Hang Up*
- Al combinar nohup y &, podemos dejar un proceso en ejecución y cerrar nuestra sesión de Shell
  - Ejemplo: `nohup ./miScriptLargo &`



# Gestión de procesos

- En la carpeta /proc se encuentra la información asociada a los procesos, que es usada por el kernel
  - Cada proceso tiene una carpeta, por ejemplo:

```
unai@unai-server:/proc$ ls
1 13 1742 205 2365 252 310 43 821 953 99 ipmi sched_debug
10 1300 18 20537 2372 2530 32 44 822 957 acpi irq schedstat
```

- Cada carpeta (o proceso) contiene:

```
root@unai-server:/proc/2554# ls
attr coredump_filter gid_map mountinfo oom_score sched stat uid_map
autogroup cpuset io mounts oom_score_adj schedstat statm wchan
auxv cwd limits mountstats pagemap sessionid status
cgroup environ loginuid net patch_state setgroups syscall
clear_refs exe map_files ns personality smaps task
cmdline fd maps numa_maps projid_map smaps_rollup timers
comm fdinfo mem oom_adj root stack timerslack_ns
```

- fd Listado de ficheros abiertos por el proceso
- stat Estado del proceso: PID, PPID, utime, ...



# Gestión de procesos

- Comandos para gestionar procesos
    - top Muestra el estado del sistema (carga de CPU, memoria,...)  
Tiene comandos propios controlar su uso (q para salir)  
*P.e, Shift+M* ordenar por consumo de RAM
    - ps Muestra información sobre los procesos activos  
Tiene parámetros para recuperar información  
*P.e. ps aux* muestra estadísticas por proceso
    - kill Enviar una señal a un proceso  
*P.e. kill -9 miProceso* fuerza su terminación
    - pstree Muestra el árbol de procesos



# Otros comandos útiles

- **sed (Stream Editor)**
  - Comando para modificar un fichero dado como entrada
  - La modificación se hace línea a línea
  - Sintaxis: `sed <opciones> <instrucciones> <fichero>`
    - Opciones:
      - `-i` El fichero de entrada es sobrescrito, en lugar de usar stdout
    - Instrucciones:
      - `i` Insertar línea antes de la actual
      - `p` Mostrar línea actual en salida estándar
      - `s` Reemplazar patrón por otro patrón en línea actual
  - Ejemplos:
    - `sed -i 's/casa/coche/g' a.txt` Reemplazar “casa” por “coche” en a.txt
    - `sed -i '/cadena/d' archivo` Eliminar toda ocurrencia de cadena en a.txt
    - `sed '2,3 p'` \* Mostrar líneas 2 y 3 de todos los ficheros



# Otros comandos útiles

- Awk

- Es un lenguaje de programación diseñado para procesado de texto
- Su nombre deriva de los apellidos de sus creadores
  - Alfred Aho, Peter Weinberger, Brian Kernighan.
- Sintaxis general de un programa awk: patrón { acción }
- El patrón determina cuando aplicar la acción
- Procesa el texto línea a línea
- Si el patrón es cierto, se aplica la acción para esa línea
- Si no se proporciona patrón, se aplica la acción a todas las líneas

- Ejemplos:

```
awk '{print $1,$4}' data.txt      Muestra la 1a y 4a columna de data.txt  
awk '{print $1,$NF}' data.txt    Muestra la 1a y última columna de data.txt  
awk 'END { print NR }' data.txt Cuenta el nº de líneas de data.txt
```



# Bibliografía

- Este tema está basado en:
- Pablo Abad Fidalgo, José Ángel Herrero Velasco. Advanced Linux System Administration, Topic 2: Command Line (Shell). OCW UNICAN 2018.
  - Publicado bajo licencia Creative Commons BY-NC-SA 4.0
  - <https://ocw.unican.es/course/view.php?id=241>
- Consultado en junio 2020



# Sistemas de ficheros

# Administración de Sistemas

Unai Lopez Novoa  
unai.lopez@ehu.eus



Universidad  
del País Vasco Euskal Herriko  
Unibertsitatea

# Contenido

1. Introducción
2. Comandos de administración
3. LVM
4. RAID
5. Copias de seguridad



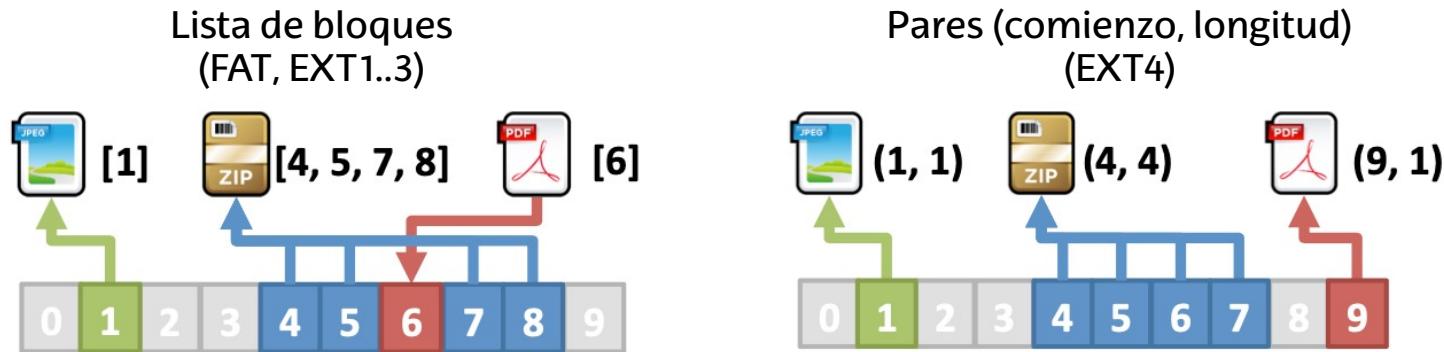
# El sistema de ficheros

- Parte del Sistema Operativo que administra la memoria de los dispositivos y unidades
- Características principales:
  - Ficheros identificados por un nombre
  - Meta-information para cada fichero
    - Fecha de creación, permisos ...
  - Organización como una jerarquía tipo árbol
- Implementación:
  - Internamente los ficheros se almacenan en bloques secuenciales
  - La jerarquía no se tiene en cuenta a nivel interno



# El sistema de ficheros

- Componentes de un fichero:
  - Datos
    - Uno o más bloques del disco con información binaria
  - Meta-information
    - Nombre, tamaño, permisos, correspondencia con los bloques de disco
- Un fichero se guarda en al menos 1 bloque del disco



# Sistema FAT

- Creado en 1977 y utilizado por MS-DOS
- *File Allocation Table* - Tabla de reserva de ficheros
  - Se basa en una lista enlazada que contiene la información de los bloques ocupados por cada archivo y los que están libres.
  - Actualmente en desuso en sistemas de escritorio
- Variantes en uso a día de hoy:
  - exFAT:
    - Orientado a tarjetas flash (p.e. SDXC) y memorias USB
  - FATX:
    - Utilizado en las videoconsolas Xbox



# Sistema NTFS

- Creado en 1993 y utilizado en Microsoft Windows
- *New Technology File System*
  - Introduce *journaling*: sistema de diario.
- Versión actual: v3.1<sup>1</sup>
  - No es tan compatible como FAT
- En proceso de ser sustituido por ReFS<sup>2</sup>
  - Disponible en Windows Server desde 2012
  - Windows 11 puede leer discos ReFS – pero no crearlos

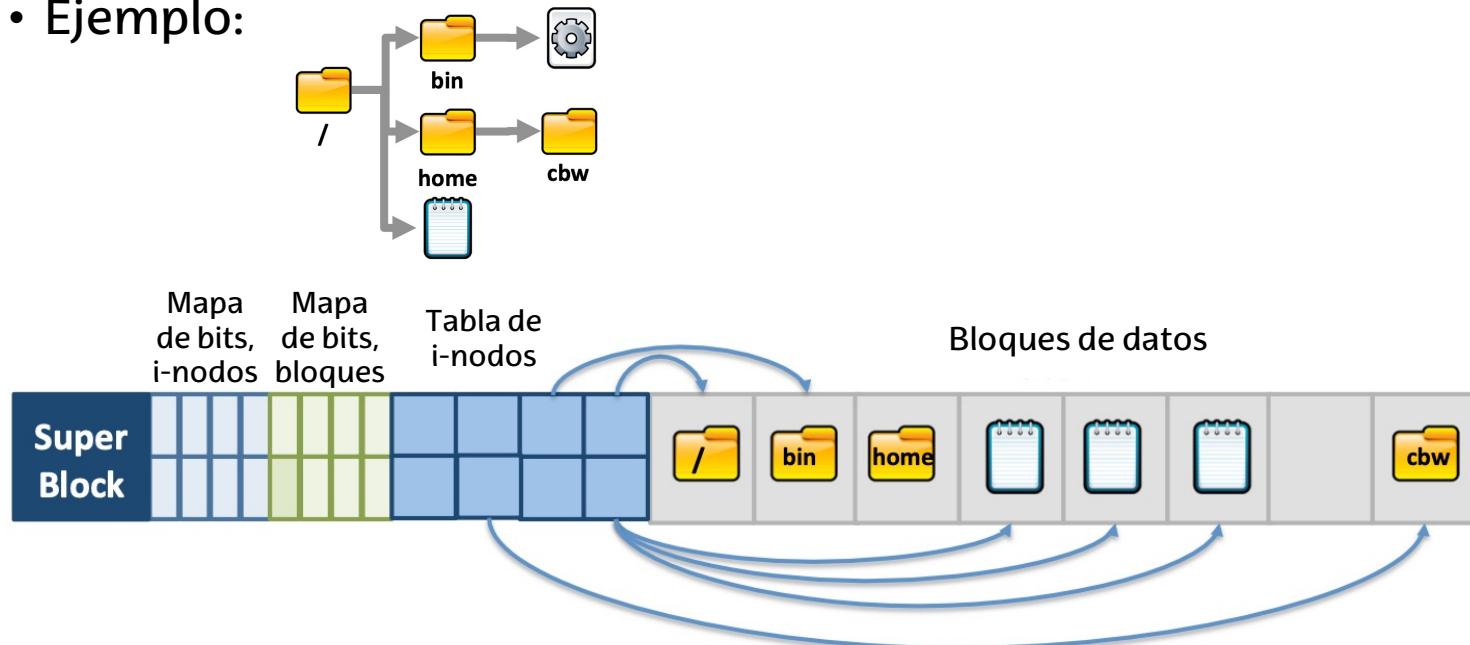


<sup>1</sup>NTFS, Microsoft: <https://docs.microsoft.com/es-es/windows-server/storage/file-server/ntfs-overview>

<sup>2</sup>ReFS, Microsoft: <https://docs.microsoft.com/en-us/windows-server/storage/refsv2/refsv2-overview>

# Sistema EXT

- Creado en 1992, el primer sistema de ficheros para Linux
- Su nombre viene de **Extended File System**
- Organiza los datos en base a i-nodos
  - Ejemplo:



# Sistema EXT

- Versión actual: Ext4
  - Introducido en 2006 y en uso estable desde 2008
- Mejoras notables sobre ext3:
  - Uso de Extents
    - Su uso reduce la fragmentación del sistema
  - Directorios organizados en árboles tipo B-Tree
    - Permite una búsqueda más rápida de directorios
  - Compatibilidad hacia delante y hacia atrás
    - Un sistema ext3 puede ser montado como ext4 sin cambios





# Btrfs

- Creado en 2009, parte de Linux desde 2013
- Características relevantes
  - Gestor de volúmenes integrado
  - Capacidad de snapshots
- Recomendado para grandes volúmenes de datos<sup>1</sup>
  - El mismo Btrfs puede expandirse a varios discos
- Posible sucesor de Ext4
  - No parece que vaya a haber Ext5<sup>2</sup>



<sup>1</sup>"Btrfs vs Ext4 - Functionalities, Strengths, and Weaknesses", LinOxide, <https://linoxide.com/btrfs-vs-ext4/>

<sup>2</sup>"Talk Of "EXT5" File-System; Should EXT4 Be Frozen?", Phoronix: [https://www.phoronix.com/scan.php?page=news\\_item&px=MTIxNTE](https://www.phoronix.com/scan.php?page=news_item&px=MTIxNTE)

# ZFS

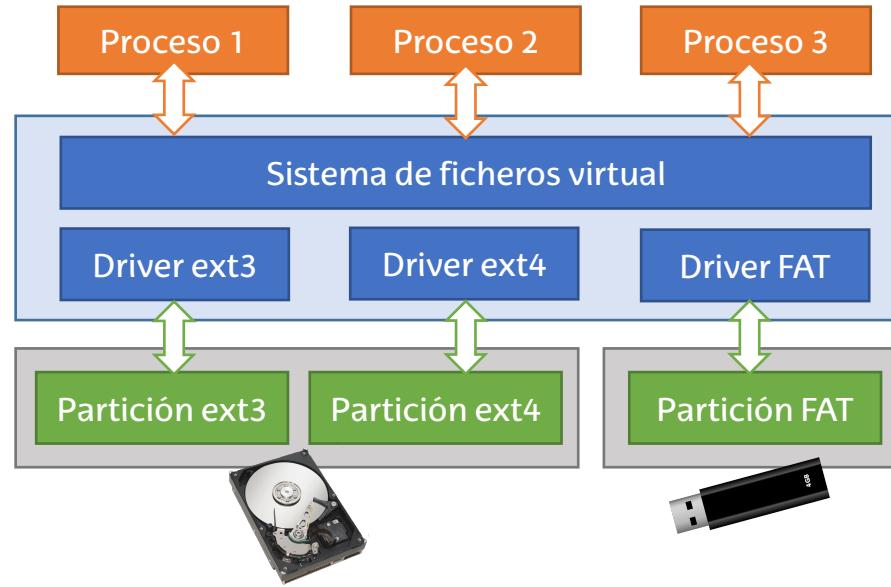
- Creado en 2001 por Sun Microsystems, ahora Oracle
- Es un sistema de ficheros y gestor de volúmenes
- Muy estable
- Licencia privativa
  - Linus Torvalds ha creado polémica al respecto<sup>1</sup>
- Variante libre: OpenZFS
  - Creado en 2013
  - Utilizado en entornos Linux

<sup>1</sup>: <https://arstechnica.com/gadgets/2020/01/linus-torvalds-zfs-statements-arent-right-heres-the-straight-dope/>



# Sistema de ficheros virtual

- Es una interfaz de Linux que:
  - Expone una API POSIX a los procesos
  - Envía las peticiones concretas al Driver que corresponda
- Aunque el SSOO monte particiones de diferentes tipos, su uso es transparente a los procesos



# Administración

- Respecto a los sistemas de ficheros, el *sysadmin* debe:
  - Garantizar que los procesos de los usuarios pueden acceder a los sistemas de ficheros locales y remotos
  - Supervisión y gestión de la capacidad de almacenamiento
  - Gestionar copias de seguridad para evitar:
    - Corrupción de los datos
    - Errores hardware
    - Errores de usuario
  - Garantizar la confidencialidad de los datos
  - Conectar y configurar nuevos discos



# Administración

- Fichero del dispositivo
  - Fichero del SSOO que posibilita que las aplicaciones accedan a un dispositivo (a través del kernel), p.e.:

cat /dev/dsp	Acceso a un DSP
cat /dev/input/mouse	Acceso al ratón
  - Todos los dispositivos se encuentran en /dev:

Dispositivos SATA:	/sdXX
Dispositivos RAID:	/mdX
Especiales, p.e.	/null (nulo)      /urandom (números aleatorios)
- Driver del dispositivo
  - Rutinas del kernel que definen cómo se comunica el SSOO con el dispositivo: Interrupciones, DMA, ...



# Administración

- Partición:
  - Unidad de almacenamiento lógico que permite tratar un único dispositivo físico como varios.
    - Permite tener un sistema de ficheros diferente en cada uno.
- Utilidad:
  - Impide que ciertos directorios crezcan indefinidamente, p.e.:

/var/spool	Para aplicaciones de colas (correo, impresión, ...)
/tmp	Para archivos temporales
  - Permite dividir el espacio para software y para archivos de usuarios



# Administración

- Partición:
  - En sistemas Unix/Linux:
    - Se encuentran en /dev, con un número adjunto al nombre del disco.
    - Ejemplo: El 2º disco SATA de un sistema Linux con 2 particiones sería:
      - /dev/sdb Disco
      - /dev/sdb1 Partición 1 del disco
      - /dev/sdb2 Partición 2 del disco
  - Con Kernels recientes, el sistema crea un alias para cada partición:
    - Se puede usar cada vez que sea necesario.
    - Evita tener que comprobar nombres después de cada reinicio.
    - P.e.: /dev/disk/by-uuid/{UUID}       enlaza al /dev/sdXX correspondiente
      - Listar UUID de cada partición: comando **blkid**



# Administración

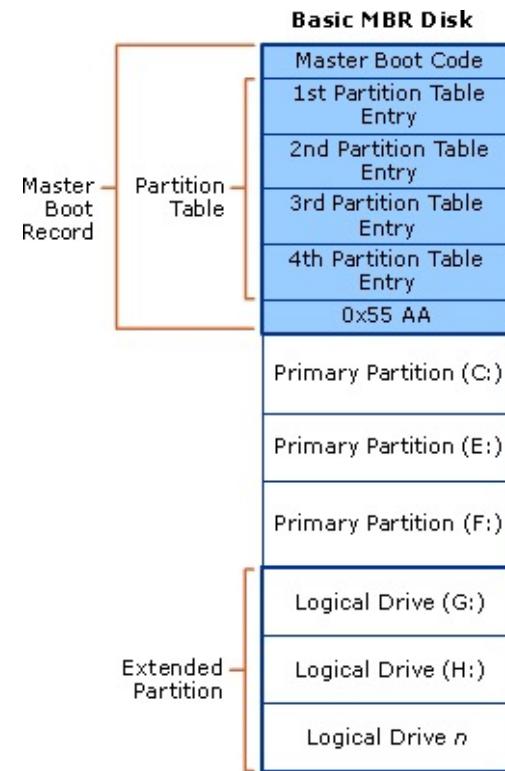
- Tabla de particiones

- Esquema que indica cómo se organizan las particiones del disco

- Generalmente es MBR o GPT

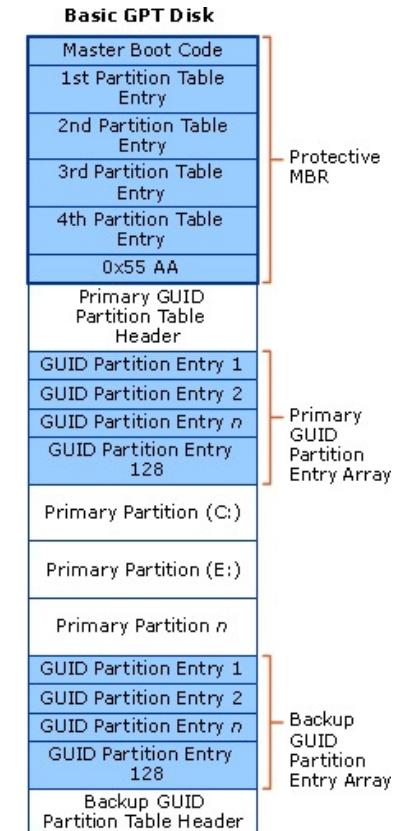
- Master Boot Record (MBR)

- A veces mostrado como DOS (por MS-DOS)
  - Introducido en 1983
  - Permite dividir 1 disco en 4 particiones primarias
  - Para crear más particiones:
    - Convertir 1 partición primaria en lógica
    - Crear particiones extendidas dentro la lógica
    - Límite: 23 particiones extendidas
  - Almacena la meta información al comienzo del disco
  - Límite: 2 TB por disco



# Administración

- Tabla de particiones
  - Esquema que indica cómo se organizan las particiones del disco
    - Generalmente es MBR o GPT
  - GUID Partition Table (GPT)
    - Introducido entre 1990~2000
    - Permite realizar hasta 128 particiones en 1 disco.
    - Almacena la meta-information distribuida por el disco.
    - Límite: 9.7 zabytes por disco
    - No es tan compatible como MBR
      - Para poder usar un disco GPT para arranque, el sistema debe ser BIOS UEFI.



# Tareas de administración

- Manipular la tabla de particiones:

- Comando **fdisk**

- Sintaxis:      **fdisk <fichero-de-dispositivo>**
    - P.e.            **fdisk /dev/sda**
    - Algunas opciones:
      - p**              Ver tabla de particiones de disco
      - n**              Nueva partición
      - w**              Escribir nueva tabla de particiones
      - q**              Salir

- Comando **cfdisk**

- Variante visual de fdisk
    - Más fácil de utilizar
    - No tiene todas las funciones de fdisk.

```
Disk: /dev/sdc
      Size: 1 GiB, 1073741824 bytes, 2097152 sectors
      Label: gpt, identifier: 7326CA46-6866-1B4B-8CF3-A1B9325C1A49

      Device           Start         End       Sectors   Size Type
>> /dev/sdc1        2048     1050623    1048576  512M Linux filesystem
      Free space     1050624     2097118    1046495  511M

Partition UUID: 0F803D31-4B5C-C647-86BC-F4F96A3C28CB
Partition type: Linux filesystem (0FC63DAF-8483-4772-8E79-3D69D8477DE4)
Filesystem UUID: a0cc5ef4-f5c1-4e1e-a3af-00912a409159
Filesystem: ext4

[ Delete ] [ Resize ] [ Quit ] [ Type ] [ Help ] [ Write ] [ Dump ]

Write partition table to disk (this might destroy data)
```



# Tareas de administración

- Formatear una partición:
  - Tras crear una partición, crear un sistema de ficheros en ella
  - Tipos de sistema soportados en /proc/filesystems
- Comando **mkfs**: crea un sistema de ficheros en una partición
  - Sintaxis:      `mkfs.<tipo-de-sistema> <partición>`
  - P.e.            `mkfs.ext4 /dev/sda3`
  - Una forma alternativa es:    `mkfs [-V -t tipo-de-sistema] <partición>`
    - P.e.            `mkfs -t ext4 /dev/sda3`
    - Se desaconseja el uso de esta forma.



# Tareas de administración

- Montar una partición
  - Habilitar el acceso al dispositivo desde el sistema de ficheros (usando el fichero de dispositivo)
  - Por defecto, comando **mount**
    - Sintaxis:      `mount <opciones> [fichero-disp] [punto-montaje]`
    - Algunas opciones:      `-t` Tipo de sistema      `-r` Montar en sólo lectura
    - Ejemplo:            `mount -t ext3 /dev/sdc1 /home/unai/miDisco`
- Desmontar una partición
  - Por defecto, comando **umount**
    - Sintaxis:      `umount [punto-montaje]`
    - Requiere que ningún proceso esté usando la partición
      - Se puede usar el comando **lsof** para mostrar qué procesos la están usando



# Tareas de administración

- Montaje automático

- El fichero /etc/fstab define los dispositivos a montar automáticamente en el arranque del sistema
- Columnas:

<file sys>	Dispositivo
<mount point>	Punto de montaje (directorio)
<type>	Tipo de partición: ext3, ext4, swap, ...
<dump>	Frecuencia de backup, no se usa en la actualidad
<pass>	Flag: ejecutar fsck al siguiente arranque

- Ejemplo:

```
# /etc/fstab: static file system information.
#
#<file sys>  <mount point>   <type>        <options>      <dump>
<pass>
proc          /proc           proc         defaults      0      0
/dev/sda1     /               ext3         errors=remount-ro 0      1
/dev/sda5     none            swap          sw            0      0
/dev/hdc      /media/cdrom0  udf,iso9660  user,noauto   0      0
/dev/fd0      /media/floppy0 auto           rw,user,noauto 0      0
```



# Tareas de administración

- Explorar las particiones del sistema
  - Comando **lsblk**
    - Lista el hardware de almacenamiento y particiones
    - Parámetro “-e7” para ocultar particiones Snap en Ubuntu
  - Comando **df**
    - Lista particiones y puntos de montajes
    - Parámetro “-h” para mostrar tamaños en formato “humano”
    - Parámetro “-t” para mostrar los tipos de sistemas de ficheros
  - Comando **mount**
    - Muestra las particiones montadas
    - Parámetro “-l” para listar
    - Parámetro “-t” para indicar tipo de sistema, p.e. “-t ext4”



# Tareas de administración

- Comprobar el sistema de ficheros
  - Comando **fsck**
    - Detección y corrección (no siempre) de problemas de corrupción en el sistema de ficheros
    - Compara la lista de bloques libres con las direcciones en los i-nodos
    - Verifica la lista de i-nodos libres con los i-nodos de los directorios
    - No es muy efectivo para detectar ficheros corruptos
  - Comando **badblocks**
    - Detecta y excluye sectores inválidos del disco
  - Funciones SMART de un disco duro
    - Self Monitoring Analysis and Reporting Technology
    - Herramientas para acceder a la información de estado del disco
    - Software y funcionalidades dependen del fabricante



# Tareas de administración

- Redimensionar el sistema de ficheros
  - Comando **resize2fs**
    - Requiere una versión del kernel  $\geq 2.6$
    - Tiene que haber espacio suficiente para poder redimensionar
  - Es conveniente hacer una copia de seguridad de la tabla de particiones:
    - Utilizando dd:      `dd if=/dev/sdc of=part.bkp count=1 bs=1`
- Comando **parted**
  - Sintaxis:      `parted /dev/sdX`
  - Permite copiar, mover, cambiar sistemas de ficheros



# Tareas de administración

- Redimensionar el sistema de ficheros
  - Comando **gparted**
    - Versión visual del comando parted

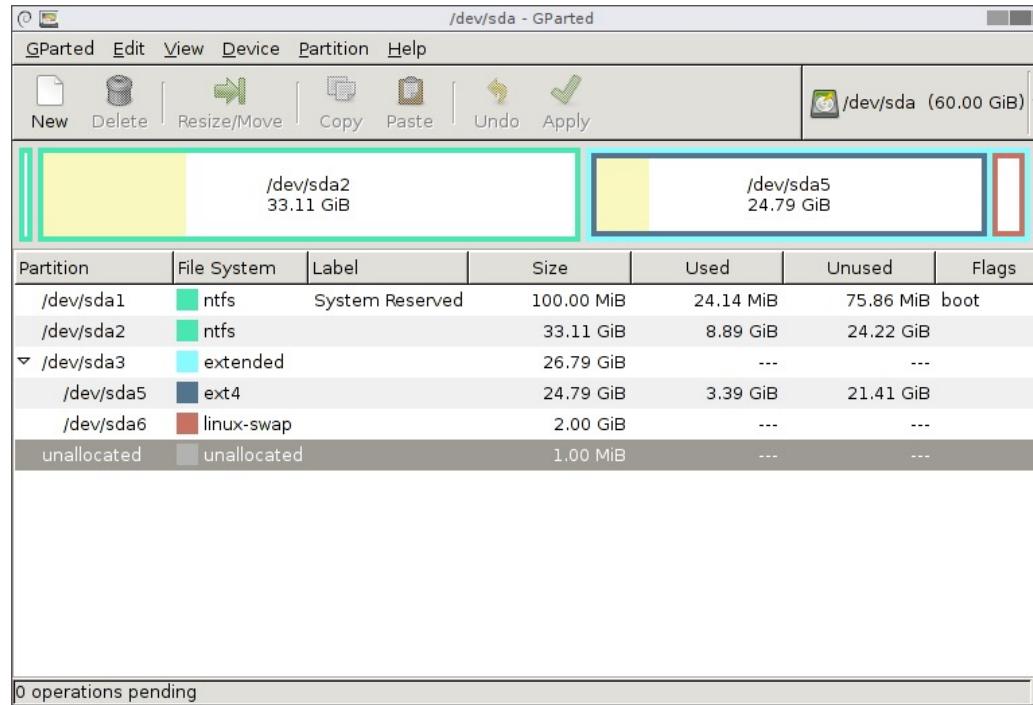
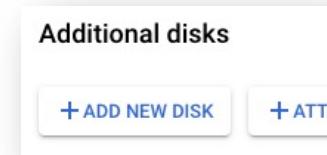


Imagen: <https://gparted.org/screens/gparted-main-window.png>

# Discos en GCP

- Añadir un disco a una instancia
  - Editar la configuración de la MV
  - Buscar la sección “Additional disks”
    - Seleccionar “Add new disk”
  - Configurar el nuevo disco, entre otras:
    - Nombre y tamaño en GB
    - Origen: “blank disk” para un disco en blanco
    - Tipo: ver siguiente diapositiva
  - Crear disco y guardar cambios en la instancia
- Administrar discos
  - Apartado “Discos” en la sección “Almacenamiento” de Compute Engine.



# Discos en GCP

- Tipos de discos<sup>1</sup>:
  - El coste mensual depende de la región<sup>2</sup>
    - Precios para la región europe-north1 a fecha 15/09/2022

Tipo	Descripción	Coste (\$/mes)
pd-standard	Discos duros tradicionales (los más lentos)	0.044 / GB
pd-balanced	Discos sólidos (SSD) configurados para ser competitivos en coste	0.110 / GB
pd-ssd	Discos sólidos (SSD)	0.187 / GB
pd-extreme	Discos sólidos (SSD) configurados para máximo rendimiento	0.137 / GB

- Ejemplo: un disco pd-balanced de 100GB en europe-north1 cuesta 11\$/mes



<sup>1</sup>GCP – Disk types: <https://cloud.google.com/compute/docs/disks#disk-types>

<sup>2</sup>GCP – Disk pricing: <https://cloud.google.com/compute/disks-image-pricing#disk>

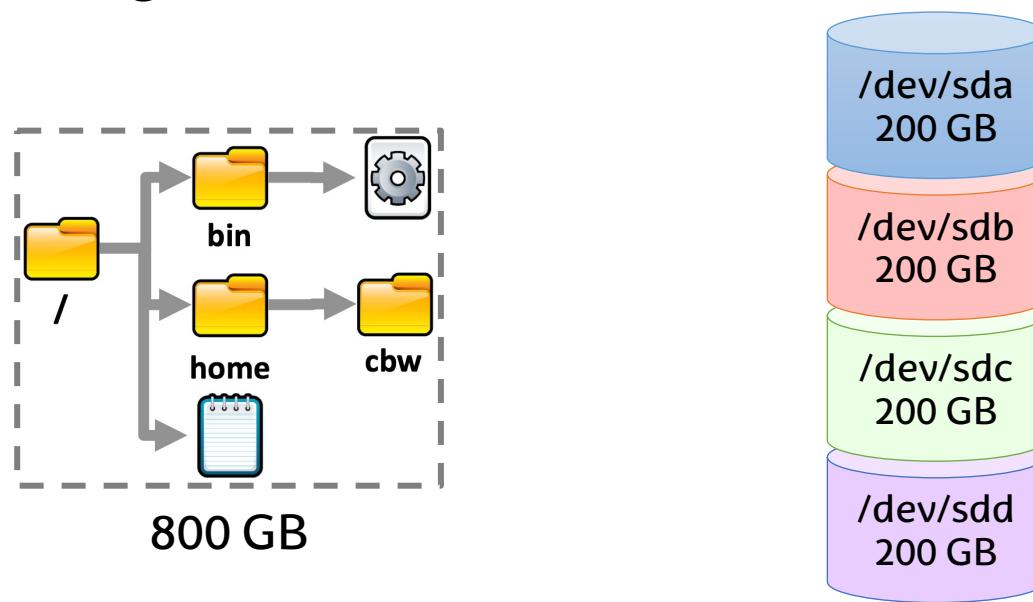
# Ejercicio 1

- Añadir un disco duro pd-balanced de 10 GB a la maquina virtual.
- Crear una partición de 10 GB en el nuevo disco y formatearla como *ext4*.
- Montar la partición en /miDisco y copiar el contenido de /var/log.
- Borrar la partición *ext4* y crear 2 particiones de 5 GB
  - Formatear una de ellas como *ext4* y la otra *btrfs*.
- Montar la partición *btrfs* en /miDisco2 y copiar el contenido de /var/log/apt.



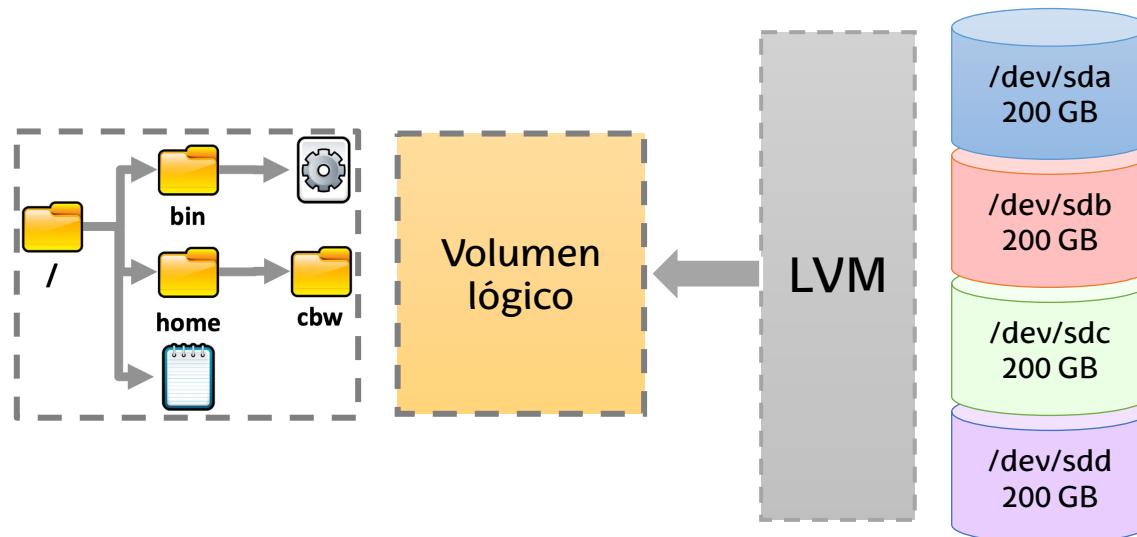
# LVM

- ¿Qué pasa si mi sistema de ficheros ocupa 800 GB pero sólo tengo discos de 200 GB ?



# LVM

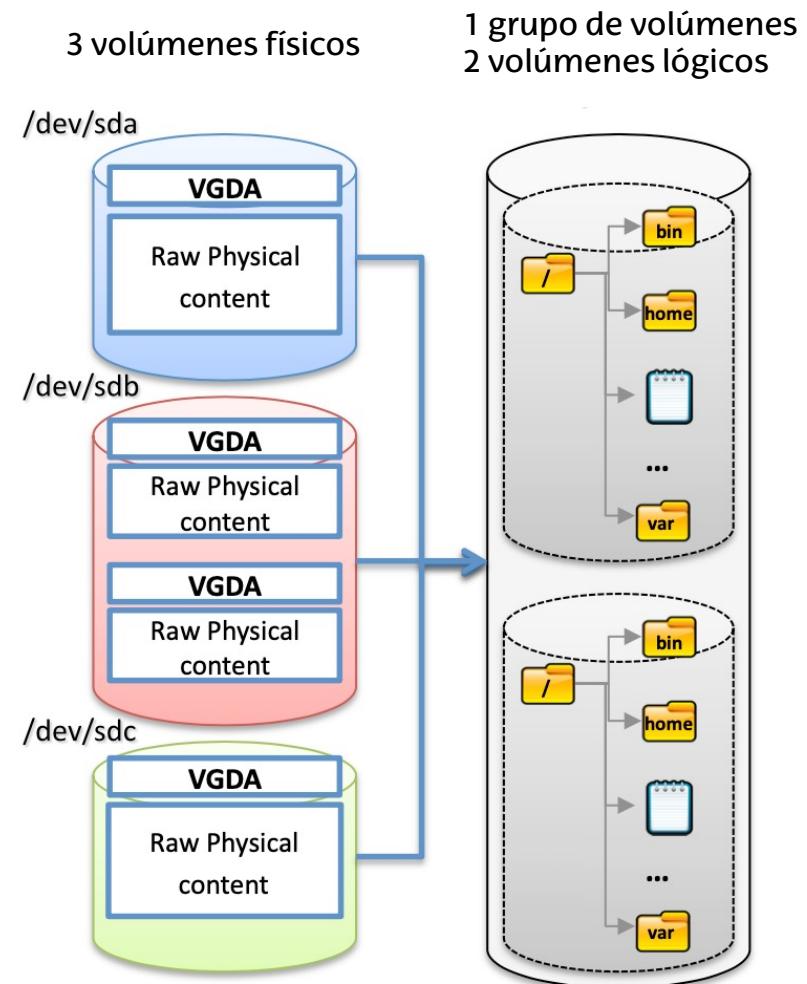
- Logical Volume Manager (LVM) crea una capa de abstracción sobre el almacenamiento físico
- Permite crear volúmenes lógicos que “escondan” el hardware real



# LVM

- Jerarquía de LVM:

- Volúmenes físicos
  - Partición completa
  - Contiene el VGDA
    - Volume Group Descriptor Area
  - Contiene los datos físicos
- Grupos de volúmenes
  - Equivalente a “super-discos”
- Volúmenes lógicos
  - Equivalente a “super-particiones”
  - Albergan los sistemas de ficheros



# LVM

- Ventajas de LVM
  - Gestión flexible del almacenamiento en disco
    - Elimina los límites del espacio físico
  - Almacenamiento redimensionable
    - Los volúmenes se pueden agrandar/reducir de forma simple
    - Algunas operaciones no requieren desmontar el sistema de ficheros
  - Traslado de datos en caliente
    - Los datos se pueden mover entre discos aunque estén en uso
    - Se puede reemplazar un disco sin interrumpir el servicio
  - Captura de instantáneas
    - Simplifica las copias de seguridad



# LVM

- Administración de LVM

- Comando **pvcreate**

- Crear un volumen físico
    - Sintaxis:      `pvcreate [partición]`
      - Es necesario crear antes la partición (p.e. con cfdisk)

- Comando **vgcreate**

- Crear un grupo de volúmenes con varios volúmenes físicos
    - Sintaxis:      `vgcreate [nombre-vol] [vols-físicos]`
      - Ejemplo: `vgcreate grupovol /dev/sdb1 /dev/sdc1`

- Comando **lvcreate**

- Creación de un volumen lógico
    - Sintaxis:      `lvcreate [nombre-grupo] -l [tamaño] -n [nombre-vol-log]`
      - Ejemplo: `lvcreate grupovol -l 100%FREE -n miVolumen`

Crea el volumen en `/dev/grupovol/miVolumen`



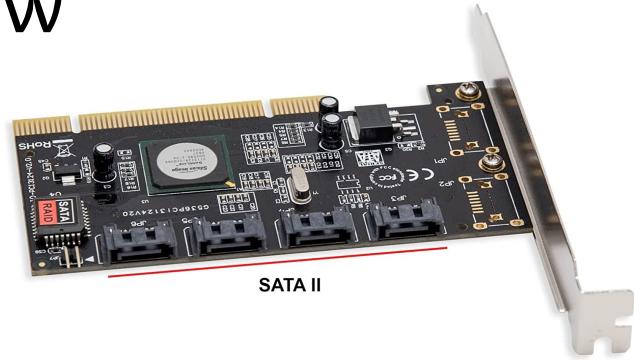
# LVM

- Administración de LVM
  - Comando **vgextend**
    - Añadir un nuevo volumen físico al grupo de volúmenes
  - Comando **lvextend**
    - Extender un volumen lógico a un grupo de volúmenes mas grande
  - Se puede redimensionar el sistema de ficheros
    - Utilizar **resize2fs**
  - Para reducir el tamaño de los volúmenes
    - Comandos **vgreduce** (grupo de volúmenes) y **lvreduce** (volumen lógico)
  - Se mostrar el estado del volumen con **lvdisplay**



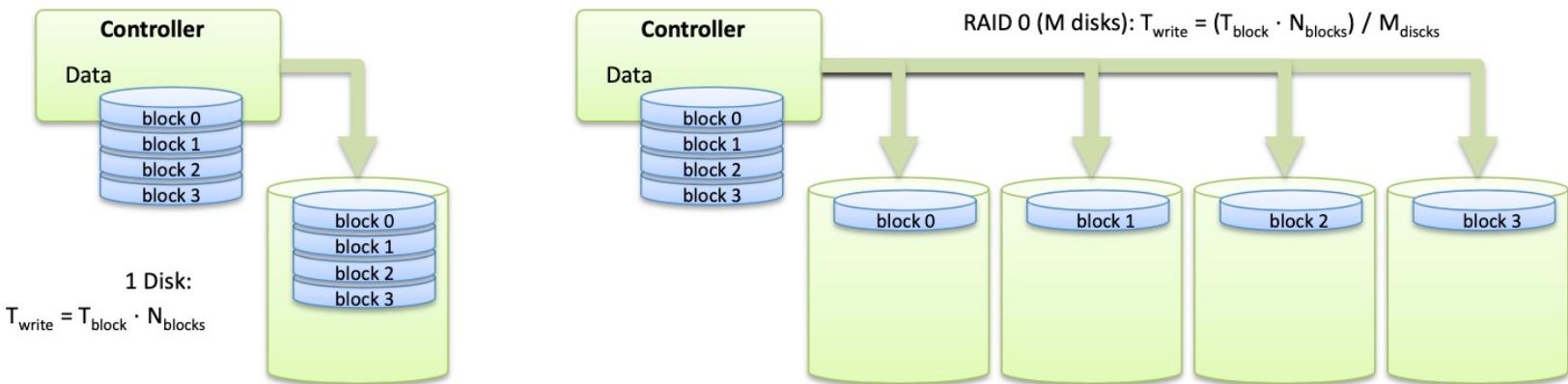
# RAID

- *Redundant Array of Independent Disks*
- Es una técnica de almacenamiento en la que los datos se distribuyen o replican entre varios discos
  - Es transparente para el usuario y para el SSOO
- Diferentes opciones de configuración (niveles)
  - Según necesidades de fiabilidad, rendimiento y capacidad
- Se puede implementar a nivel HW o SW
  - Hardware: más eficiente pero más caro
    - Imagen: controladora PCI para RAID 0, 1, 5 y 10
  - Software: apropiado para RAID 0 y 1



# RAID

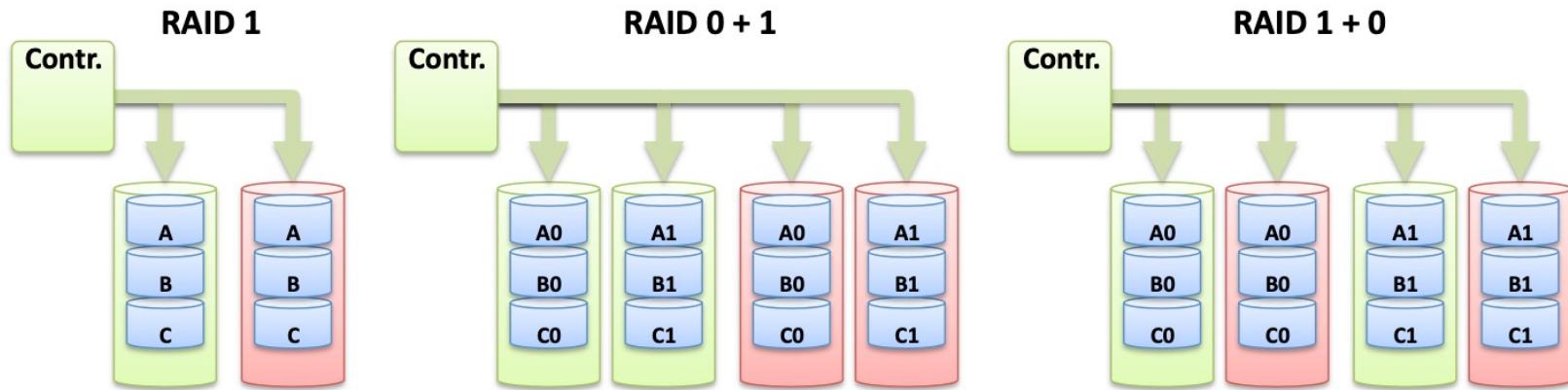
- **RAID 0: Striping (Volumen dividido)**
  - Los datos se dividen en segmentos y se distribuyen entre los discos
- **Rendimiento:** Bueno, acceso paralelo a los discos
  - Cuantos más discos, más velocidad
- **Fiabilidad:** No hay tolerancia a fallos
- **Capacidad:** 100% de uso (0 redundancia)



# RAID

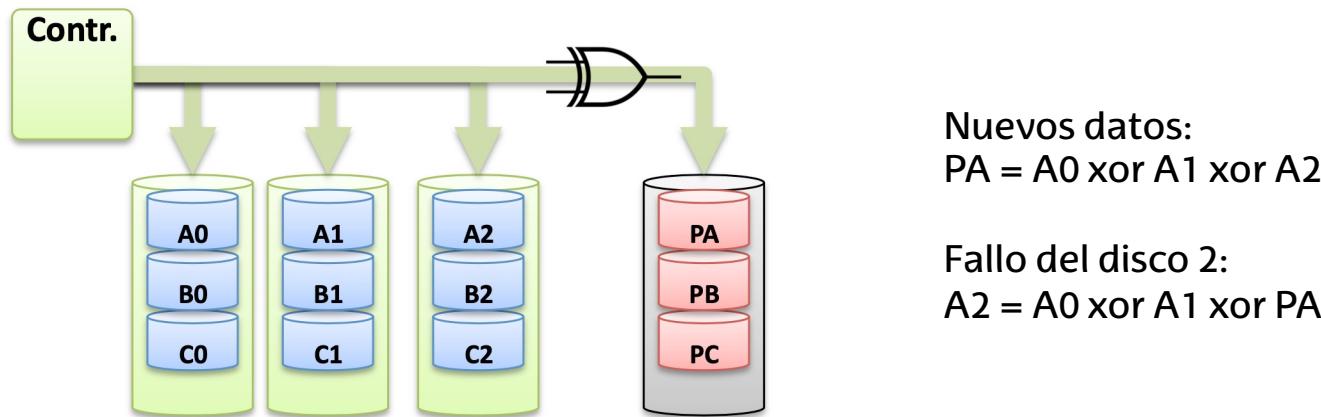
- **RAID 1: Espejo**

- Utilizar un disco secundario para copiar todos los datos
- **Rendimiento:** Bajo, debido al exceso de escrituras
- **Fiabilidad:** Alta por la alta redundancia
- **Capacidad:** 50% de la disponible



# RAID

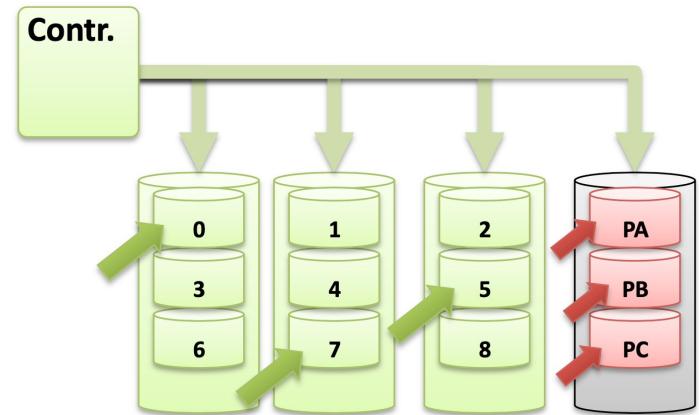
- **RAID 4: Striping + paridad**
  - Un disco almacena información de paridad sobre el resto
  - **Rendimiento:** Bueno en lectura, malo en escritura
  - **Fiabilidad:** Tolerancia al fallo de 1 disco
  - **Capacidad:** 1 disco dedicado exclusivamente a redundancia



# RAID

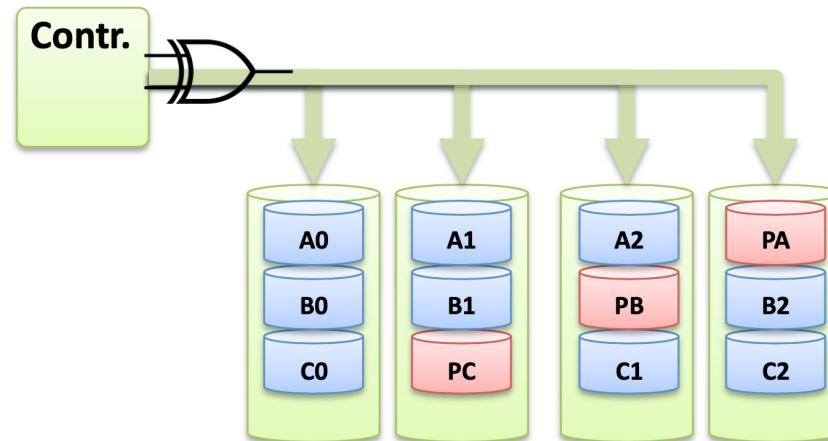
- **RAID 4: Striping + paridad**

- El mayor problema en RAID 4 son las escrituras serializadas en el mismo disco
- Ejemplo: actualizar las posiciones 0, 5 y 7
  - 1) Leer bloques 0, 5 y 7 y PA, PB y PC
  - 2) Calcular el nuevo valor de PA, PB y PC
  - 3) Escribir los nuevos bloques de datos
  - 4) Escribir los nuevos bloques de paridad
    - Este último paso implica escrituras serializadas
    - **Bajo rendimiento**



# RAID

- RAID 5: *Striping + paridad distribuida*
  - La información de paridad se distribuye por todos los discos
  - **Rendimiento:** Mejor que RAID 4, elimina la escritura serializada
  - **Fiabilidad:** Tolerancia al fallo de 1 disco
  - **Capacidad:** Se dedica a redundancia el equivalente a 1 disco



# RAID

Otros niveles RAID:

- **RAID 2, RAID 3**
  - Paridad a nivel de bit (RAID2) o byte (RAID3), en lugar de bloque.
  - No es muy utilizado
- **RAID 6: Striping + Doble paridad**
  - RAID 4 pero usando el doble de espacio para paridad
  - Tolerante al fallo de 2 discos
- **RAID anidados: jerarquías en árbol**
  - P.e, RAID 0+1, RAID 1+ 0 (10), ...



# RAID

- Administración RAID, se utiliza el comando **mdadm**:
  - Creación de un dispositivo RAID
    - Para crear el dispositivo /dev/md0:
      - `mdadm --create /dev/md0 --verbose --level=0 --raid-devices=2 /dev/sdb /dev/sdc2`
      - Los discos tienen que haber sido previamente particionados (p.e. con cfdisk)
      - El proceso de creación se puede monitorizar:
        - `cat /proc/mdstat`
    - Monitorizar el sistema RAID
      - `mdadm --monitor [opciones] /dev/md0`
    - Eliminar (desactivar) RAID:
      - Parar el dispositivo:           `mdadm --stop /dev/md0`
      - Limpiar información:       `mdadm --zero-superblock /dev/sdX`
        - Limpia la información existente de un dispositivo RAID parado



# RAID

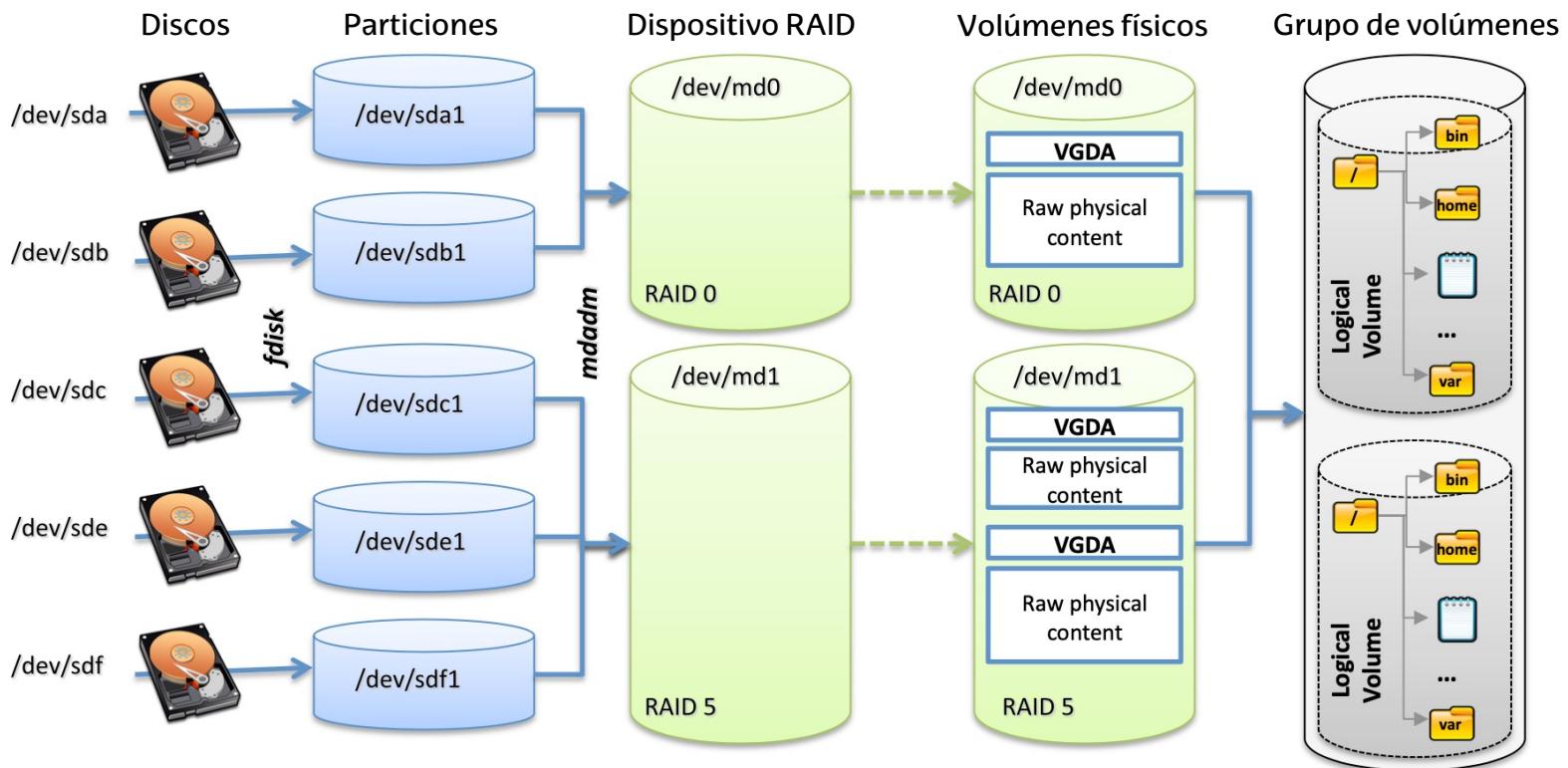
- En caso de fallo de 1 disco
  - Asumiendo un sistema RAID 5
  - El disco roto se puede recuperar automáticamente:
    - Eliminar el disco roto del RAID:
      - mdadm /dev/md0 -r /dev/sdc1
    - Reemplazar el disco físico por otro (debe ser idéntico)
    - Crear particiones como en el original:
      - fdisk /dev/sdc.
    - Añadir al dispositivo RAID:
      - mdadm /dev/md0 -a /dev/sdc1
    - Monitorizar el proceso de reconstrucción:
      - cat /proc/mdstat
  - Se puede simular el fallo de un disco:
    - Utilizar: mdadm /dev/md0 -f /dev/sdc1
    - Toda la información en: /var/log/syslog



# RAID

- Combinando RAID y LVM

- LVM se debe implementar **sobre** RAID



# Ejercicio 2

- *Utilizar el disco virtual creado en el ejercicio 1.*
- Borrar las particiones existentes y crear 3 particiones de 2 GB.
- Juntar las 3 particiones como un único volumen LVM llamado *volumenEj2*.
- Formatear *volumenEj2* como *ext4* y montarlo en */mivol*
  - Verificar el tamaño del volumen resultante



# Backups

- RAID + Journaling no es suficiente para tener una disponibilidad del 100%
- Tener copias de seguridad es esencial
  - Solución para eventos inesperados, tanto HW como SW
  - Evita potenciales problemas de los usuarios
- Implica dedicar recursos exclusivos
  - Recursos físicos
    - Discos dedicados exclusivamente a copias, Servidores SAN, ...
    - Cintas: LTO (LinearTape-Open), SAIT, AIT
  - Almacenamiento en la nube



# Backups

- Informes de BackBlaze
  - Consultora dedicada al almacenamiento en la nube
    - A fecha de 30 de junio 2022 gestionan 219.444 discos duros en 4 *datacenters*
  - Cada trimestre publica un informe detallando:
    - Ratio de fallos en sus discos duros
    - Comparativas de rendimiento
    - Datos históricos
  - Último informe: Abril – Junio 2022
    - <https://www.backblaze.com/blog/backblaze-drive-stats-for-q2-2022/>



# Backups

- Informe trimestral de BackBlaze

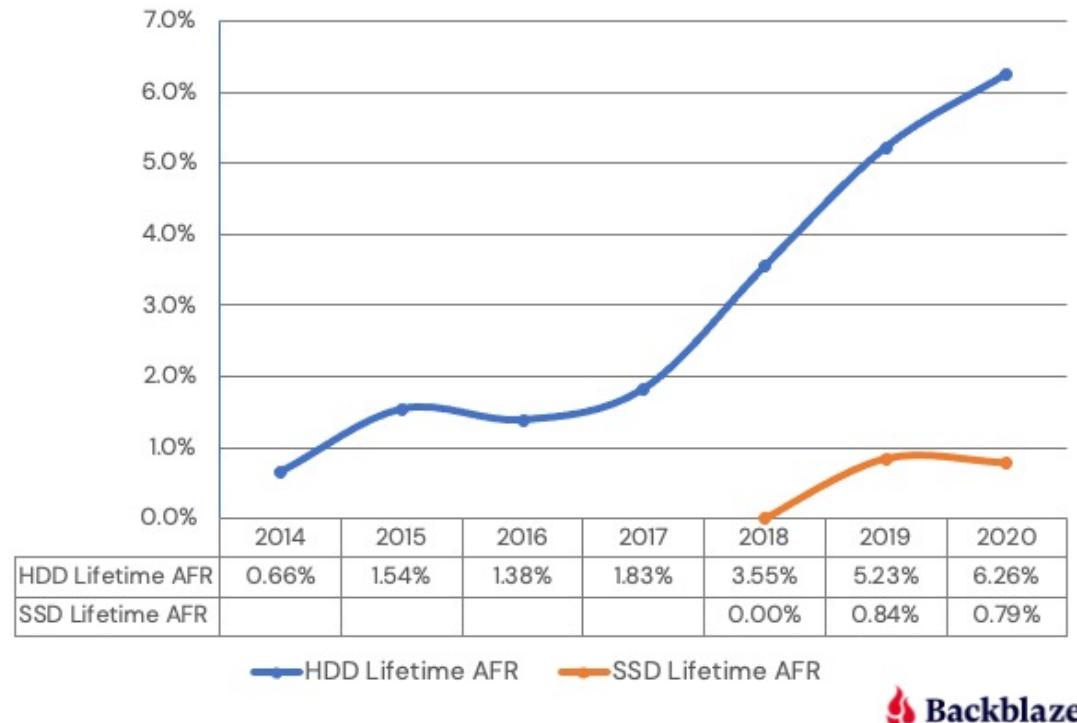
- Abril – Junio 2022,  
Ratio de fallos en  
discos HDD y SSD:
- La columna AFR  
(*Annualized Failure Rate*) mide el nº de  
fallos por año en un  
grupo de discos.

MFG	Model	Drive Size	Drive Count	Avg. Age (months)	Drive Days	Drive Failures	AFR
HGST	HMS5C4040ALE640	4TB	3,664	71.4	325,695	13	1.46%
HGST	HMS5C4040BLE640	4TB	12,728	68.2	1,154,012	18	0.57%
HGST	HUH728080ALE600	8TB	1,126	50.5	101,318	3	1.08%
HGST	HUH728080ALE604	8TB	76	61.9	6,846	–	0.00%
HGST	HUH721212ALE600	12TB	2,605	32.9	234,867	2	0.31%
HGST	HUH721212ALE604	12TB	13,138	15.3	1,186,624	19	0.58%
HGST	HUH721212ALN604	12TB	10,802	38.9	972,567	16	0.60%
Seagate	ST4000DM000	4TB	18,359	80.3	1,662,649	156	3.42%
Seagate	ST6000DX000	6TB	886	86.7	80,626	2	0.91%
Seagate	ST8000DM002	8TB	9,630	68.6	872,039	48	2.01%
Seagate	ST8000NM0055	8TB	14,370	58.1	1,294,335	79	2.23%
Seagate	ST10000NM0086	10TB	1,180	55.8	106,596	9	3.08%
Seagate	ST12000NM0007	12TB	1,288	31.7	116,642	16	5.01%
Seagate	ST12000NM0008	12TB	20,033	27.0	1,812,740	108	2.17%
Seagate	ST12000NM001G	12TB	12,389	19.4	1,111,321	35	1.15%
Seagate	ST14000NM001G	14TB	10,728	17.0	970,470	28	1.05%
Seagate	ST14000NM0138	14TB	1,571	18.8	142,330	22	5.64%
Seagate	ST16000NM001G	16TB	16,860	8.9	1,393,804	44	1.15%
Toshiba	MDO4ABA400V	4TB	97	85.3	8,730	–	0.00%
Toshiba	MG07ACA14TA	14TB	38,205	20.1	3,455,150	97	1.02%
Toshiba	MG07ACA14TEY	14TB	501	16.6	44,174	–	0.00%
Toshiba	MG08ACA16TA	16TB	2,488	2.0	32,064	–	0.00%
Toshiba	MG08ACA16TE	16TB	5,946	8.7	536,234	41	2.79%
Toshiba	MG08ACA16TEY	16TB	4,032	9.4	299,701	2	0.24%
WDC	WUH721414ALE6L4	14TB	8,408	37.4	761,275	2	0.10%
WDC	WUH721816ALE6LO	16TB	2,702	17.5	242,077	2	0.30%
WDC	WUH721816ALE6L4	16TB	1,199	9.0	107,979	1	0.34%
			215,011		19,032,865	763	1.46%



# Backups

- Informe trimestral de BackBlaze
  - *Annualized failure rate (AFR)* Abril 2013 – Diciembre 2020



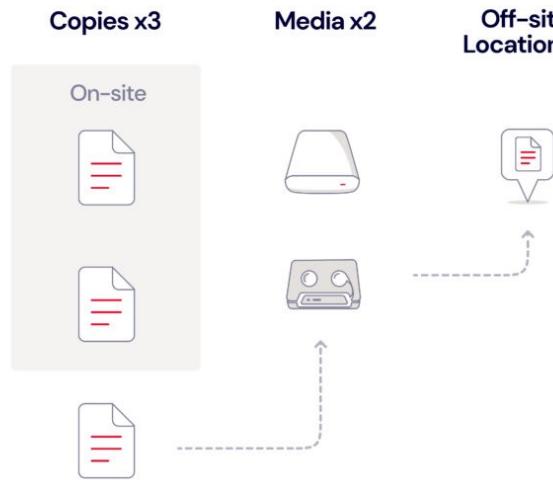
# Backups

- La política de copias debe ir acorde a nuestros requisitos
  - ¿Qué es necesario guardar?
    - Datos de usuarios / aplicaciones / sistema
    - Las partes críticas del sistema
  - ¿Cuándo queremos hacer las copias?
    - No recargar el sistema en momentos críticos
    - Dependerá del nivel de uso y la parte del sistema de ficheros
    - Automatizar las copias (usando p.e. cron)
  - ¿Dónde queremos hacer las copias?
    - Balance entre copias locales y en ubicaciones remotas

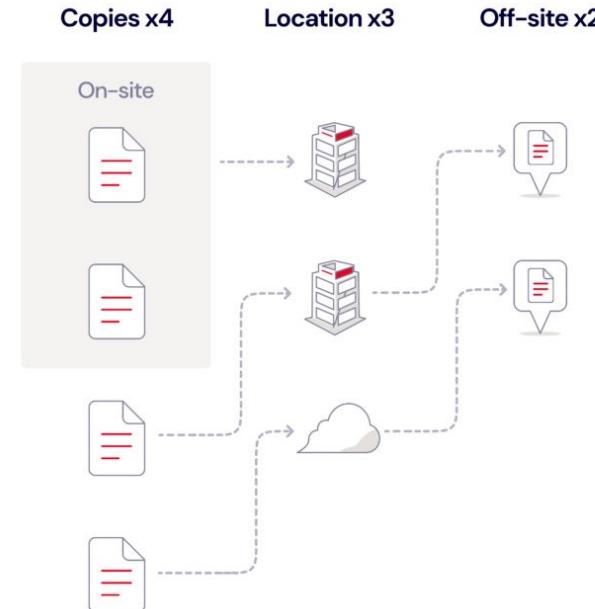


# Backups

- Estrategias para el almacenamiento de Backups
  - Ejemplos:



Estrategia 3-2-1



Estrategia 4-3-2



# Backups

- Comando **rsync**
  - Herramienta GNU para backups
  - Web oficial: <https://rsync.samba.org/>
    - Documentación, FAQ, Novedades
  - Forma de uso más simple:
    - rsync [opciones] <origen> <destino>
    - Opciones:
      - v      Modo verboso
      - a      Mantiene usuarios,
      - z      Comprime antes de copiar
      - h      Mostrar tasas de transferencia y tamaños en formato legible (MB/s en vez de bytes/s)
    - Ejemplo:    rsync -vazh /home /dev/sdc
  - Se suele utilizar para copias remotas por red



# Backups

- Comando **rsnapshot**
  - Herramienta basada en **rsync** para realizar copias incrementales, gestionando un histórico de las mismas con rotación
    - Web: <https://rsnapshot.org/>
    - Documentación: <https://wiki.archlinux.org/title/Rsnapshot>
  - No viene instalada en Ubuntu Server por defecto
    - Instalar con “apt install rsnapshot”
  - Configuración: /etc/rsnapshot.conf
  - Uso:
    - **rsnapshot configtest** Verifica que la configuración es correcta
    - **rsnapshot <TAG>** Realiza una copia del tipo <TAG>, p.e. “daily”
    - **rsnapshot-diff** Compara 2 copias hechas en instantes diferentes



# Backups

- Alternativas más rudimentarias:
  - Comando **tar**
    - Combinándolo con herramientas de compresión (bzip, zip)
  - Comando **dd**
    - dd if=/dev/sda2 of=/dev/tape
  - Comando **cp -a**
    - Para replicar contenido de disco a nivel de fichero
- Alternativas comerciales:
  - HP Data Protector
  - IBM Spectrum Protect (Tivoli Storage Manager)
  - ...



# Bibliografía

- Pablo Abad Fidalgo, José Ángel Herrero Velasco. "Advanced Linux System Administration", OCW UNICAN, 2018<sup>1</sup>:
  - Topic 6: File systems fundamentals
  - Topic 7: File systems, advanced management
  - Publicado bajo licencia Creative Commons BY-NC-SA 4.0
  - <https://ocw.unican.es/course/view.php?id=241>
- Alberto González, "¿Que es Logical Volume Manager o LVM?", Octubre 2015<sup>1</sup>:
  - <https://nebul4ck.wordpress.com/2015/10/06/que-es-logical-volume-manager-o-lvm/>
- GitLab Docs, "File system performance benchmarking"<sup>2</sup>:
  - [https://docs.gitlab.com/ee/administration/operations/filesystem\\_benchmarking.html](https://docs.gitlab.com/ee/administration/operations/filesystem_benchmarking.html)
- Consultados en julio 2020<sup>1</sup> y septiembre 2021<sup>2</sup>



# Monitorización

# Administración de Sistemas

Unai Lopez Novoa  
unai.lopez@ehu.eus



Universidad  
del País Vasco Euskal Herriko  
Unibertsitatea

# Contenido

1. Gestión de recursos
  - CPU, memoria y disco
2. Registros del sistema (logs)
3. Monitorización en GCP
4. Rendimiento



# Introducción

- Gestionar correctamente el uso de los recursos hardware es crítico para asegurar un mínimo nivel de servicio.
- Tareas asociadas
  - Monitorizar y supervisar el estado del sistema
  - Gestionar las prioridades de los procesos
  - Hacer uso de los recursos CPU en horas de baja utilización
    - P.e., programar análisis o copias de seguridad durante la noche



# Monitorización de CPU

- Comando **top**
  - Uso de recursos del sistema en tiempo real
- Comando **ps**
  - Listado de procesos y su uso de recursos
- Comando **pstree**
  - Árbol de procesos del sistema
- *Estos 3 están descritos en el Tema 1, Diapositiva 42*



# Gestión de CPU

- Prioridades de los procesos
  - El planificador del SSOO asigna intervalos de tiempo a los procesos según su prioridad.
  - Esto se controla según 2 valores:
    - Prioridad (**PR**): puede tomar valores en el rango -100 a 39.
    - Valor "nice" (**NI**): puede tomar valores en el rango -20 a 19.
    - *Columnas PR y NI en el comando Top.*
    - *Para ambos, cuanto más negativo el valor, mayor prioridad.*
  - En Linux, los procesos se consideran de 2 tipos:
    - Procesos normales (la mayoría de los lanzados por usuarios).
    - Procesos de tiempo real (generalmente, los esenciales para el SSOO).



# Gestión de CPU

- Prioridades de los procesos normales
  - Se calcula:  $\text{PR} = 20 + \text{NI}$ 
    - P.e. si el valor NI de un proceso es -20, su prioridad es 0
    - P.e. si el valor NI es 19, su prioridad es 39.
  - Los procesos normales ocupan el rango 0-39 de prioridades.
  - Por defecto, el valor “nice” (NI) es 0.
  - Un usuario normal puede modificar el valor NI entre 0 y 19.
    - Puede reducir la prioridad sobre el resto de procesos del sistema
  - El usuario *root* puede modificar el valor NI -20 y 19.



# Gestión de CPU

- Prioridades de los procesos normales
  - Comando **nice**
    - Lanza un comando con un valor Nice concreto
    - Sintaxis:      nice -n valor comando
      - Valor es relativo (define cuánto más o menos)
    - Ejemplo:      nice -n 10 ./miScript
  - Comando **renice**
    - Cambia el valor Nice de un proceso (o grupo) en ejecución
    - Un usuario normal (no root) sólo puede incrementar el valor Nice.
      - Y cada cambio que haga es irreversible
    - Sintaxis:      renice -n valor -p PID [-g grupo]
      - Valor es absoluto
    - Ejemplo:      renice -n 15 -p 7552



# Gestión de CPU

- Prioridades de los procesos en tiempo real
  - Se calcula:  $\text{PR} = -1 - \text{prioridad\_tiempo\_real}$ .
  - El valor *prioridad\_tiempo\_real* toma valores entre 1 y 99.
    - P.e. si *prioridad\_tiempo\_real* es 50, PR vale -51.
    - El valor "nice" no se tiene cuenta.
    - En el comando top, si PR=-100, se muestra como 'rt' (real time).
- Comando **chrt**
  - Lanza un proceso con una prioridad de tiempo real
  - Sintaxis:      chrt --rr <prioridad\_tiempo\_real> <programa>
  - Ejemplo:      chrt --rr 20 ./miPrograma
    - Lanzaría ./miPrograma con PR=-21



# Gestión de CPU

- Prioridades de los procesos: comando **ps**
  - El comando **ps** puede mostrar datos en diferentes formatos
    - Cada formato puede mostrar una misma prioridad con diferentes valores.
- Formato BSD:
  - Comando: ps al

```
F UID  PID  PPID PRI  NI  VSZ   RSS   WCHAN STAT TTY TIME COMMAND
...
0 1000 2033 1104 21  1  7208 2708 - SN+.  pts/0  0:00 ping www.ehu.eus
```

- Formato Unix:
  - Comando: ps -u unai -o pid,user,pri,nice,args

```
PID  USER PRI NI COMMAND
...
2033  unai 18  1  ping www.ehu.eus
```

Se muestra un valor de prioridad (PRI) diferente para un mismo proceso (PID=2033).



# Gestión de CPU

- Comando **kill**
  - Envía señales a procesos (no sólo matarlos)
  - Sintaxis:      **kill <opciones> PID**
    - Opciones:    -l       Mostrar las señales disponibles  
                  -señal   Mandar una señal al proceso
    - Hay 3 formas de indicar una señal:
      - Con su número    -19
      - Con el prefijo SIG -SIGSTOP
      - Sin el prefijo SIG -STOP
    - Señales:       -STOP   Parar el proceso  
                  -CONT   Reanudar el proceso (parado con STOP)  
                  -KILL   Matar el proceso
  - ...



# Gestión de CPU

- Comando **ulimit**
  - Limitar el uso de recursos
  - Los límites sirven para la Shell en uso
  - Sintaxis:      **ulimit -<opción> [límite]**
    - Opciones:    -a      Lista los límites establecidos
    - f      Máximo número de ficheros creados por la Shell
    - m      Máxima memoria disponible
    - t      Máximo tiempo de CPU (segundos)
- Ejemplo:

```
unai@unai-server:~$ ulimit -a
core file size          (blocks, -c) 0
scheduling priority      (-e) 0
file size                (blocks, -f) unlimited
pending signals           (-i) 31543
max memory size          (kbytes, -m) unlimited
open files                 (-n) 1024
POSIX message queues     (bytes, -q) 819200
real-time priority        (-r) 0
stack size                  (kbytes, -s) 8192
cpu time                   (seconds, -t) unlimited
```



# Gestión de CPU

- Fichero `/etc/security/limits.conf`
  - Permite hacer una configuración permanente de límites
  - Cada línea tiene el siguiente formato:  
`usuario/grupo tipo-de-límite ítem valor`
  - Donde:
    - `usuario/grupo` Nombre del usuario o grupo (comienza con @)
    - `tipo-de-límite` soft/hard
    - `ítem` Puede ser: cpu, nproc, maxlogins, fsize, ...
    - `valor` Valor para el ítem definido
  - Ejemplos:

```
@student hard nproc 20
@faculty soft nproc 20
@faculty hard nproc 50
ftp      hard nproc 0
```
  - Mas información: `man limits.conf`



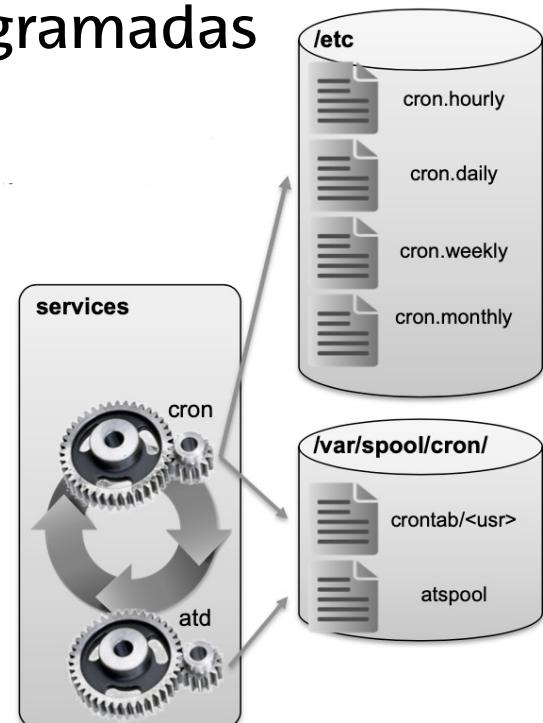
# Gestión de CPU

- Comando **cpulimit**
  - Permite limitar el % de uso constante de CPU de un proceso
    - ulimit y limits.conf sólo permiten limitar el tiempo total de uso CPU
    - nice y renice permiten reducir la prioridad pero no fijar un umbral
  - Está en los repositorios Debian
  - Uso: **cpulimit --pid PID --limit <límite>**
    - Donde <límite> es el límite de % CPU máximo que queremos permitir
  - Más información:
    - <https://www.tecmint.com/limit-cpu-usage-of-a-process-in-linux-with-cpulimit-tool/>



# Planificación de tareas

- Se pueden programar tareas para que se ejecuten periódicamente (**cron**) o una única vez (**atd**)
- **cron** y **atd** leen periódicamente sus ficheros de configuración para ejecutar tareas programadas
  - Por defecto, cada minuto
- Algunas tareas programables:
  - Rotación de logs
  - Borrar la carpeta /tmp
  - Copias de seguridad
  - Actualizar una BBDD

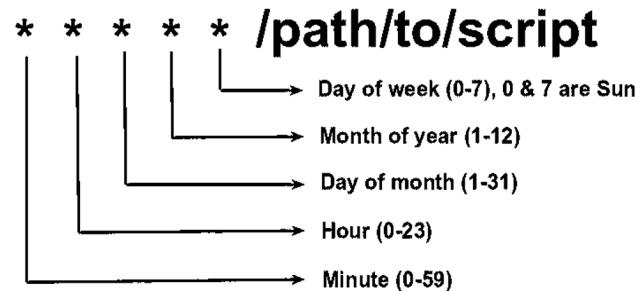


# Planificación de tareas

- Comando **crontab**

- Una línea por tarea programada
- Sintaxis:      crontab <opciones>
  - Opciones:    -l      Mostrar las tareas programadas
  - e      Editar las tareas programadas
  - r      Elimina las tareas programadas

- Cada entrada de cron es una línea sigue la estructura:



# Planificación de tareas

- Comando **crontab**

- Ejemplos:

- |                                |  |
|--------------------------------|--|
| • 6 17 * * * /scripts/copia.sh | Ejecuta copia.sh los sábados a las 17:00   |
| • * * * * /scripts/miScript.sh | Ejecuta miScript.sh cada minuto            |
| • * 5,17 * * * /scripts/api.sh | Ejecuta api.sh a diario a las 5:00 y 17:00 |
| • * */10 * * * /scripts/mon.sh | Ejecuta mon.sh a diario cada 10 minutos    |

- Más ejemplos en:

- <https://tecadmin.net/crontab-in-linux-with-20-examples-of-cron-schedule/>



# Planificación de tareas

- Comando **crontab**
  - <https://crontab.guru/>
  - Editor online de entradas cron

**crontab guru**

The quick and simple editor for cron schedule expressions by [Cronitor](#)

*“At 14:15 on day-of-month 1.”*

[next](#) at 2020-10-01 14:15:00      [random](#)

15 14 1 \* \*

minute	hour	day (month)	month	day (week)
*				any value
,				value list separator
-				range of values
/				step values



# Planificación de tareas

- Comando at
  - Controla las tareas a ejecutar por atd
  - Para programar una tarea
    - Desde Shell: at HORA (donde HORA es una hora en formato HH:MM)
    - Se abre el Shell de at, escribir el/los comando(s) deseado(s):
      - P.e. ls /home/unai -l
    - Cerrar la Shell de at (Ctrl + D)
    - La salida estándar (stdout) se envía por mail usando sendmail
      - Conviene revisar /var/spool/mail/<usuario>
  - Otras opciones:
    - Desde Shell: at -l Listado de tareas pendientes
    - Desde Shell: at -d <ID> Eliminar tarea (obtener ID con -l)



# Gestión de memoria

- Monitorizar la memoria

- Comando **top**

- Utilizar Shift+m para ordenar por consumo descendente de memoria

- Comando **vmstat**

- Campos relativos a la memoria:

procs	memoria				swap		io		sistema				cpu			
r	b	swpd	libre	búfer	caché	si	so	bi	bo	in	cs	us	sy	id	wa	st
1	0	8972	1097844	776348	5687216	0	0	0	3	1	2	0	0	100	0	0
0	0	8972	1097712	776348	5687216	0	0	0	0	42	48	0	0	100	0	0
0	0	8972	1097712	776348	5687216	0	0	0	0	39	48	0	0	100	0	0

**swpd:** Memoria swap en uso

**libre:** Memoria libre

**buff:** Memoria usada como buffer

**cache:** Memoria usada como cache

**si:** Memoria swap retirada de disco(Swap In)  
**so:** Memoria swap llevada a disco (swap out)



# Gestión de discos

- Monitorización
  - Comando **df**
    - Listado de particiones del sistema de ficheros y espacio disponible
    - Sintaxis: df <opciones>
    - Ejemplo: df -h
      - Muestra los tamaños en kB, MB, ... en lugar de en bytes (-h para Human readable)
  - Comando **du**
    - Tamaño de una rama del sistema de ficheros (p.e., de un directorio)
    - Sintaxis: du <opciones> directorio
    - Ejemplos: du -sh /home
      - Muestra el tamaño total del directorio /home sin listar todo su contenido



# Gestión de discos

- Cuotas de uso
  - Se puede controlar la cantidad de datos y ficheros para cada usuario/grupo.
    - Los usuarios son responsables de respetar sus límites.
  - Se pueden establecer dos tipos de límites:
    - *Soft*: se puede exceder por un periodo limitado de tiempo.
    - *Hard*: no se puede exceder.
  - ¿Qué límites debo establecer?
    - Dependerá del tipo de usuario y aplicaciones que usen, y del espacio de disco disponible.
    - Mejor políticas conservadoras: es más fácil expandir que reducir.



# Gestión de discos

- Cuotas de uso
  - Requiere:
    - Soporte de cuotas en el kernel del SSOO
    - Configurar el sistema de ficheros
      - Modificar /etc/fstab con usrquota y grpquota (después, remontar con mount)
    - Ejemplo:
      - Editar /etc/fstab para añadir cuotas a la partición montada en /myFs

```
# /etc/fstab: static file system information.  
...  
/dev/sdb1  /myFs   ext4   defaults,usrquota,grpquota  0  0
```

- Volver a montar:

```
mount -o remount /myFs
```

- Verificar que se ha montado con soporte para cuotas:

```
root@unai-server:/myFs# mount | grep /dev/sdb  
/dev/sdb1 on /myFs type ext4 (rw,relatime,quota,usrquota,grpquota)
```



# Gestión de discos

- Cuotas de uso
  - Comando **edquota**

- Sintaxis:      **edquota <opciones> [-u usuario] [-g grupo]**
- Inicia un editor de texto para añadir cuotas
- Los límites se definen en bloques de 1 KB e i-nodos (null = ilimitado)
- Ejemplo:

```
Disk quotas for user mikel (uid 1001):
Filesystem    blocks      soft      hard      inodes      soft      hard
/dev/sdb1        4          8          9          1          0          0
```

- Opciones:
  - t      Cambiar el tiempo de exceso en una quota tipo Soft
  - p      Copia valores entre usuarios  
P.e. **edquota -p usuario1 usuario2**



# Gestión de discos

- Cuotas de uso
  - Comando **quotacheck**
    - Verificar la integridad del sistema de cuotas.
    - Es necesario ejecutarlo antes de usar **edquota** o **quotaon**.
  - Comando **quotaon/quotaoff**
    - Activar/desactivar sistema de cuotas.
  - Comando **repquota**
    - Reporta el estados de las cuotas en un sistema de ficheros.
  - Comando **quota**
    - Ver estado y quota de usuario:        `quota -u <usuario>`
- En Ubuntu, instalar el paquete **quota**.



# Ejercicio 1

- Instalar el paquete stress-ng
  - Suite de programas para evaluar el rendimiento
- Ejecutar stress-ng durante 2 minutos con 2 hilos CPU. Mientras está en ejecución, cambiar la prioridad de uno de sus procesos a la mínima posible.
  - ¿Qué sucede?
  - Probar a cambiar la prioridad del mismo a la máxima posible.
- Ejecutar stress-ng durante 3 minutos con 1 hilo CPU. Mientras está en ejecución, limitar su uso de CPU al 50%.
  - Verificar con top.



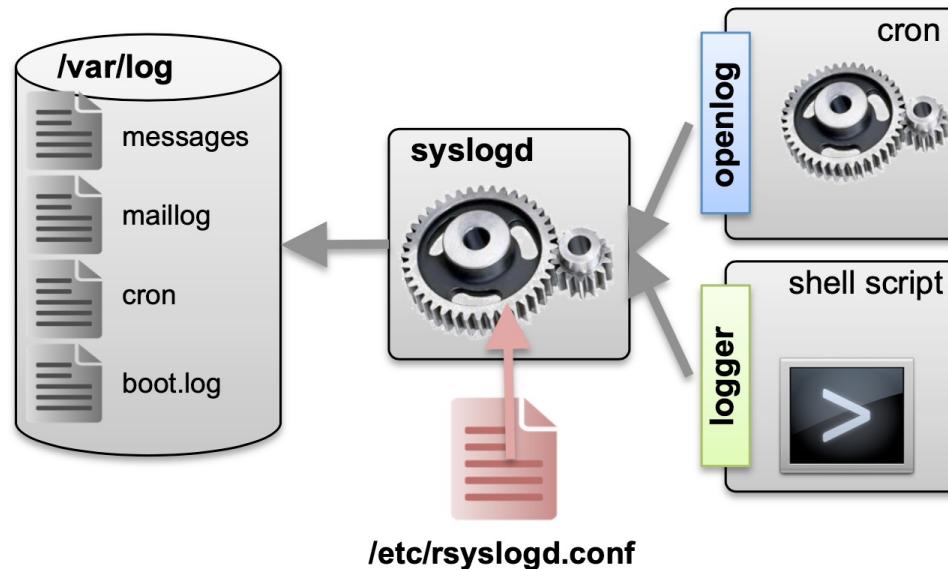
# Logs

- El kernel de Linux, los servicios y las aplicaciones generan eventos constantemente:
  - Información sobre su estado
  - Información sobre fallos/anomalías
  - Errores de arranque
  - Acceso a información (seguridad)
- Una gestión adecuada de esta información es esencial para descubrir y solucionar problemas
- Todos estos eventos suelen estar gestionados por un único servicio
  - En Unix/Linux es syslog



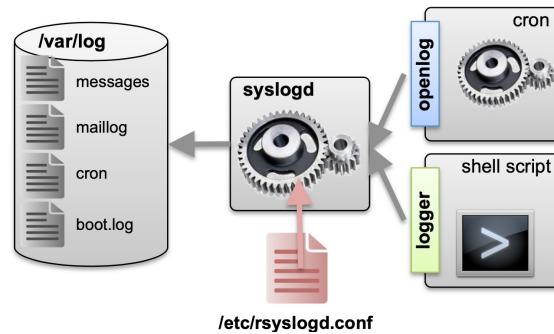
# Syslog

- Es el recolector de eventos empleado por el kernel, servicios y aplicaciones
- Flexible, seguro y fácil de usar
- Está compuesto por los siguientes elementos:



# Syslog

- Partes de syslog:
  - **syslogd**: Servicio del sistema. Recibe los mensajes del resto de servicios y aplicaciones y los añade al registro.
  - **openlog**: Librerías para usar syslog desde una aplicación.
    - P.e., openlog (C/C++), sys::syslog(openlog(),syslog()) (Perl)
  - **logger**: Comando del sistema para enviar mensajes a syslog.
  - **rsyslogd.conf**: Fichero de configuración.
    - Ver siguiente transparencia.



# Syslog

- **rsyslogd.conf**
  - Listado de acciones a realizar en función de los mensajes recibidos.
  - Tiene una línea por acción, con el formato:
    - entidad.nivel acción
  - **Entidad:** lista de valores definidos por el sistema
    - P.e.: Kern, user, daemon (otro servicio), auth (login, su, ssh), mail, cron, ...
  - **Nivel:** tipo de notificación
    - emerg, alert, crit, err, warning, notice, info, debug, \* (todos los niveles)
  - **Acción:**
    - <nombre-de-fichero> Escribir el mensaje a ese fichero.
    - <nombre-dominio>/<IP> Enviar el mensaje al syslogd del nodo indicado.
    - <nombre-usuario> Enviar mensaje al usuario, si está conectado.
    - \* Enviar mensaje a todo usuario conectado.



# Syslog

- **rsyslogd.conf**

- Ejemplo:

```
# First some standard log files.  Log by facility.  
#  
auth,authpriv.* /var/log/auth.log  
*.*;auth,authpriv.none -/var/log/syslog  
#cron.* /var/log/cron.log  
mail.* -/var/log/mail.log  
#user.* -/var/log/user.log  
  
# Logging for the mail system.  
#  
#mail.info -/var/log/mail.info  
#mail.warn -/var/log/mail.warn  
mail.err /var/log/mail.err  
...
```

- En Ubuntu, por defecto es: /etc/rsyslog.d/50-default.conf



# Syslog

- Syslog escribe en ficheros en /var/log:
  - /syslog Eventos generales, ni críticos ni de depuración
  - /maillog Información de e-mails
  - /cron Registros del proceso cron
  - /boot.log Mensajes e información de inicio del sistema
- Hay ficheros en /var/log/ que **no** gestiona syslog
  - /wtmp Registra accesos de los usuarios y reinicios
    - Está en formato binario
    - Es utilizado por los comandos last y uptime
  - /lastlog Contiene el último acceso de cada usuario
  - /dmesg Eventos del inicio del sistema
    - Es utilizado por el kernel y proceso init



# Gestión de logs

- Los logs son una herramienta fundamental para el control y reparación del sistema.

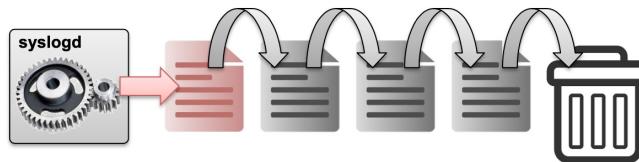
*Pero...*

- Cuanta más información de logs, mayor uso de disco.
  - Los logs pueden llegar a consumir un espacio significativo.
  - Puede ser costoso buscar información/datos concretos entre miles de líneas.



# Gestión de logs

- Rotación de logs
  - Periódicamente cambiar el fichero donde se escriben los logs, cambiando a escribir en uno nuevo y borrando el más antiguo.



- Se puede hacer de manera manual con un script.
  - Ejemplo:

```
#!/bin/bash
cd /var/log/
mv messages.2 messages.3
mv messages.1 messages.2
mv messages messages.1
cat /dev/null > messages
chmod 600 messages

#Reiniciar syslog
service restart rsyslog
```



# Gestión de logs

- Rotación de logs
  - Se puede implementar la rotación con un servicio del sistema.
    - Evita errores humanos al crear scripts.
  - Servicio **logrotate**
  - Se configura con los siguientes ficheros:

*/etc/logrotate.conf*

*Por defecto, para todos los servicios*

```
# rotate log files weekly, monthly
weekly
# keep 4 weeks worth of backlogs
rotate 4
# send errors to root
errors root
# compressed log files
compress
...
...
```

*/etc/logrotate.d/*

*Sobrescribe logrotate.conf  
para un servicio concreto*

```
/var/log/dpkg.log {
    monthly
    rotate 12
    compress
    notifempty
    create 0664 root
    adm
}
```



# Gestión de logs

- Analizando logs

- Para depuración:

- Útil para obtener más información cuando algo va mal
    - Activar modo verbose de las aplicación
      - P.e. activar flag -d, /etc/init.d/ssh sshd -d
    - Importante: desactivar el modo verbose al volver a producción

- Para monitorización:

- Problema: abundante información, de la cual mucha puede no ser útil
    - Utilizar herramientas para buscar los mensajes relevantes, p.e.:
      - Swatch: Programa Perl que busca patrones en los logs<sup>1</sup>
      - LogWatch: Genera resúmenes para su envío por e-mail.
      - Soluciones más complejas, p.e., pila ELK.



<sup>1</sup>: Swatch: The Simple Log Watcher: <https://www.linuxtoday.com/blog/swatch-the-simple-log-watcher.html>

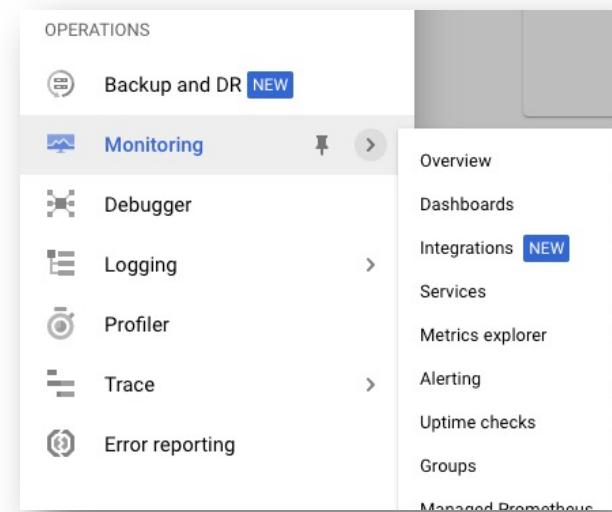
# Ejercicio 2

- Enviar un mensaje al log del sistema desde el terminal y comprobar que se ha añadido correctamente.
- Añadir una regla a syslog para que los mensajes de usuario de tipo "debug" se escriban en /var/log/user\_debug.log
- Enviar un mensaje de tipo debug y comprobar que se escribe en un /var/log/user\_debug.log
- Devolver syslog a su situación anterior
  - Eliminar la regla recién creada



# Monitorización en GCP

- GCP provee un sistema de monitorización para los servicios en uso
  - Se accede en la sección “Operaciones”
- Permite obtener métricas en diferentes resoluciones
  - Minutos, horas, días, ...
  - La mayor resolución es 1 minuto.
- Tiene un coste asociado
  - Existe una capa gratuita que incluye monitorización básica<sup>1</sup>



<sup>1</sup>Precios de la suite de operaciones en GCP: <https://cloud.google.com/stackdriver/pricing?hl=es>

# Monitorización en GCP

- Organiza la información en Dashboards de control
  - Por defecto, incluye varios para los recursos más comunes
    - Sección Dashboards de “Monitoring”:

The screenshot shows a list of pre-built dashboards under the 'Monitoring' section. The columns are 'Name' and 'Icon'. The dashboards listed are: Disks, Firewalls, Infrastructure summary, and VM Instances.

Name	Icon
Disks	Disk icon
Firewalls	Shield icon
Infrastructure summary	Bar chart icon
VM Instances	Virtual machine icon

- Permite crear Dashboard personalizados
  - Con diferentes tipos de gráficos:



# Monitorización en GCP

- Para monitorizar una MV<sup>1</sup>:
  - Se debe instalar un agente en la MV a monitorizar
    - Es un proceso que envía datos de monitorización a Google Cloud

```
curl -sSO https://dl.google.com/cloudagents/add-google-cloud-ops-agent-repo.sh  
sudo bash add-google-cloud-ops-agent-repo.sh --also-install
```

- Verificar que GCP ha detectado la instalación correctamente
  - Puede tardar unos pocos minutos.
  - Desde el Dashboard “VM Instances”:

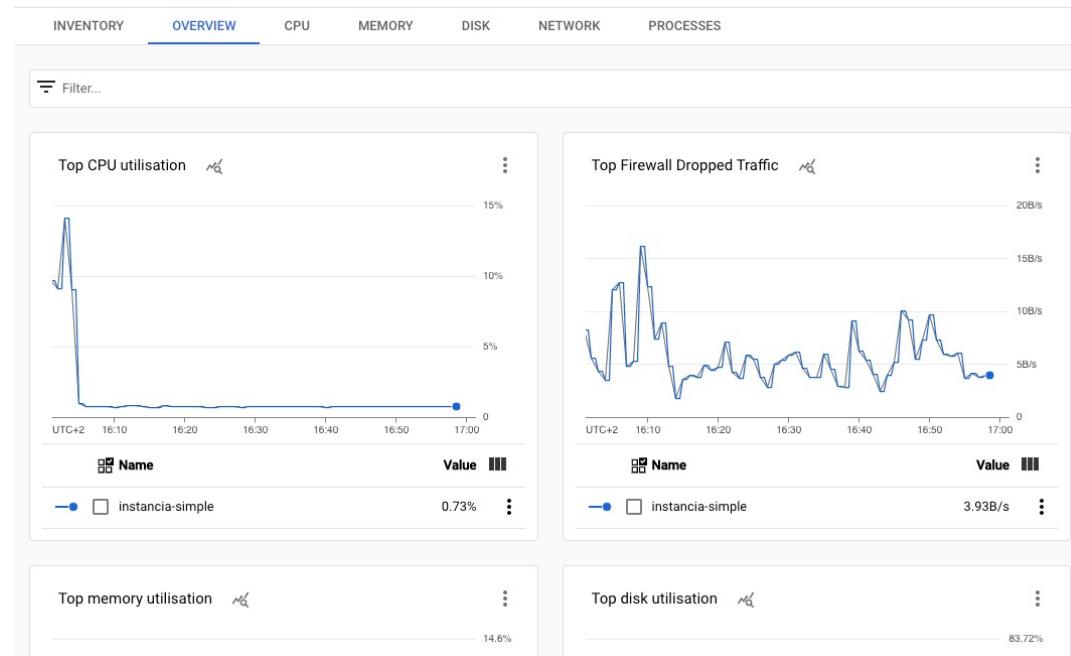
Instances		 INSTALL/UPDATE AGENTS	▼
 Filter Enter property name or value			
<input type="checkbox"/>	Name	Agent	↓ Active alerts
<input type="checkbox"/>	<a href="#">instancia-simple</a>	 Ops Agent	Ops Agent version: 2.20.0



<sup>1</sup>Supervisa una MV de Compute Engine: <https://cloud.google.com/monitoring/monitor-compute-engine-virtual-machine?hl=es>

# Monitorización en GCP

- Métricas de una VM
  - Por defecto, se obtienen métricas relativas a uso de recursos y red
  - Se visualizan en forma de gráfica:



# Monitorización en GCP

- Se pueden crear alertas que envíen notificaciones cuando sucedan eventos concretos.
  - P.e. cuando una métrica supere un umbral, al detectar un fallo, etc.
  - Se gestionan desde la sección “Alertas”
- Para crear alertas personalizadas:
  - Crear una política
    - Elegir la métrica a monitorizar
    - Elegir el umbral de la métrica
    - Elegir el canal de notificación (p.e. e-mail)
    - Revisar y guardar



# Ejercicio 3

- Activar la monitorización para una MV de Compute Engine.
- Ejecutar stress-ng durante 2 minutos
  - Verificar su ejecución con el comando **top**
- Verificar que el incremento de carga de CPU se refleja en los Dashboards de monitorización.



# Rendimiento

- Depende del entorno del trabajo, los usuarios van a utilizar una colección de aplicaciones más o menos variada.
- Es útil caracterizar las máquinas de nuestro entorno para saber cómo van a responder con estas aplicaciones.



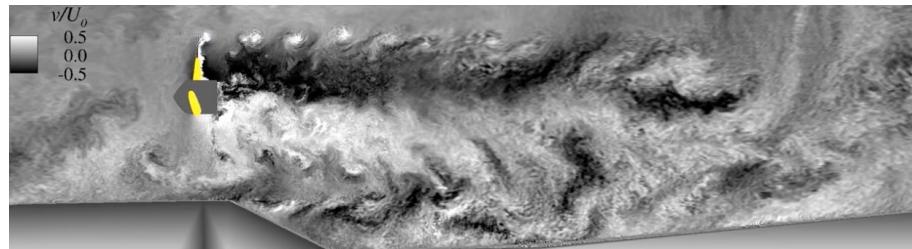
# Rendimiento

- Generalmente, una aplicación está limitada por uno de los siguientes:
  - Cómputo
    - Operaciones a realizar en la CPU > operaciones con la memoria
    - Ejemplos de aplicaciones:
      - Renderizado de gráficos/video, simulaciones químicas, ...
  - Memoria
    - Datos a transferir > capacidad de procesado de la CPU
    - Ejemplos de aplicaciones:
      - Análisis de datos, simulaciones de dinámica de fluidos, ...



# Rendimiento

- Ejemplo de aplicación limitada por memoria:
  - Simulación de dinámica de fluidos



- Video: [https://youtu.be/7zeoOv\\_wPxM](https://youtu.be/7zeoOv_wPxM)
- Software: Hydro3D<sup>1</sup>
- Tiempo utilizado para la simulación del video: ~3 meses en 456 cores CPU
- Conocer el rendimiento de la máquina es de ayuda para estimar el tiempo de ejecución.



<sup>1</sup>: Hydro3D - <https://github.com/OuroPablo/Hydro3D>

# Benchmarks

- Son aplicaciones cuyo objetivo es comprobar el rendimiento de un factor concreto, p.e.
  - CPU, memoria, discos, red, librerías, BBDDs, ...
- Generalmente ejecutan una operación (o pocas) de manera repetida y miden el tiempo necesario
  - En ocasiones el objetivo es validar que un software es estable
- Características que debe un benchmark cumplir:
  - Relevante y representativo
  - Repetible
  - Escalable



# Benchmarks

- SPEC
  - Standard Performance Evaluation Corporation (SPEC) es un consorcio americano dedicado a desarrollar benchmarks
    - Web oficial: <https://www.spec.org/benchmarks.html>
  - SPEC CPU
    - Software comercial (1.210 US\$), última versión: 2017 (anterior 2006)
    - Diferentes versiones: *speed, integer, floating point, ...*
  - SPEC Cloud
    - Software comercial (2.420 US\$), última versión: 2018 (anterior 2016)
  - Otros: SPEC ACCEL, SPEC MPI,...



# Benchmarks

- Firestarter
  - Benchmark *open-source* de stress CPU.
  - Crea y ejecuta diferentes patrones de carga en intervalos configurables por el usuario.
  - Implementación específica para diferentes arquitecturas:
    - Intel: Sandy Bridge, Broadwell, Skylake, Knights Landing, ...
    - AMD: Zen, Zen+
- Web oficial: <http://tu-dresden.de/zih/firestarter/>
  - Última versión: 2.0 (Enero 2021)
  - Ejercicio: descargar Firestarter y ejecutar en PC o máquina virtual



# Benchmarks

- Firestarter
  - Algunos resultados para comparar:

Modelo de CPU	Arquitectura	Lanzamiento	Capacidad	GFLOP/s
Intel Xeon E5-2620	Sandy Bridge	2012	6 cores @ 2.0 GHz	33.9
Intel Xeon E5-2695 v4	Broadwell	2016	18 cores @ 2.1 GHz	397.5
Intel Xeon Gold 6148	Skylake	2017	20 cores @ 2.4 GHz	1000.9
Intel Xeon Silver 4216	Cascade Lake	2019	16 cores @ 2.1 GHz	398.3

- Se utiliza el tipo de instrucción más apropiado para cada chip.

Resultados de: U. Lopez-Novoa. *Exploring Performance and Energy Consumption Differences between Recent Intel Processors* (2019) <https://ieeexplore.ieee.org/abstract/document/9060093>



# Benchmarks

- STREAM
  - Benchmark de evaluación de memoria muy popular
  - Realiza diferentes operaciones con 2 vectores:
    - Copy               $a(i) = b(i)$
    - Scale              $a(i) = s*b(i)$
    - Add                 $a(i) = b(i)+c(i)$
    - Triad               $a(i) = b(i)+s*c(i)$
  - El tamaño de los vectores lo define el usuario.
  - STREAM mide el tiempo en realizar estas operaciones.
  - Es importante compilar y configurar STREAM correctamente para obtener resultados fiables.
- Web oficial: <https://www.cs.virginia.edu/stream/>



# Benchmarks

- STREAM
  - Algunos resultados para comparar:

Modelo de CPU	Memoria	GB/s	GFLOP/s
Intel Xeon E5-2620	32 GB DDR3 @ 1333 MHz	23.61	33.9
Intel Xeon E5-2695 v4	128 GB DDR4 @ 2400 MHz	48.10	397.5
Intel Xeon Gold 6148	384 GB DDR4 @ 2666 MHz	74.90	1000.9

- Resultados de la prueba Triad con tamaño de array  $2^{25}$ .
- Compilado con GCC v7.3 y flags `-O3` y `-march`

Resultados de: U. Lopez-Novoa. *Exploring Performance and Energy Consumption Differences between Recent Intel Processors* (2019) <https://ieeexplore.ieee.org/abstract/document/9060093>



# Rendimiento

- ¿Cómo determinar si nuestra aplicación está limitada por cómputo o memoria?
- Utilizar:
  - Herramientas de profiling
    - Comerciales: Intel Vtune, ARM MAP, Cray PAT
    - Libres: Linux perf, Linux Trace Toolkit, Valgrind, gprof
  - Modelos de rendimiento
    - Descripción matemática que representa una interacción hardware-software
    - Generalmente son específicos a una máquina o aplicación
    - Requieren de un trabajo de configuración
      - A veces, utilizar benchmarks para obtener información del hardware



# Consumo energético

- Cada vez se da más importancia al consumo energético de servidores y centros de proceso de datos.
- El consumo energético está subiendo cada vez más:
  - El supercomputador más potente del mundo en junio 2012 consumía 7.9 MW<sup>1</sup>.
  - El más potente a junio 2022 consume 21.1 MW<sup>1</sup>.
    - Equivale ~3670 domicilios españoles<sup>2</sup>.
- Se puede analizar con diferentes herramientas.
  - Profilers, Librerías como Performance API

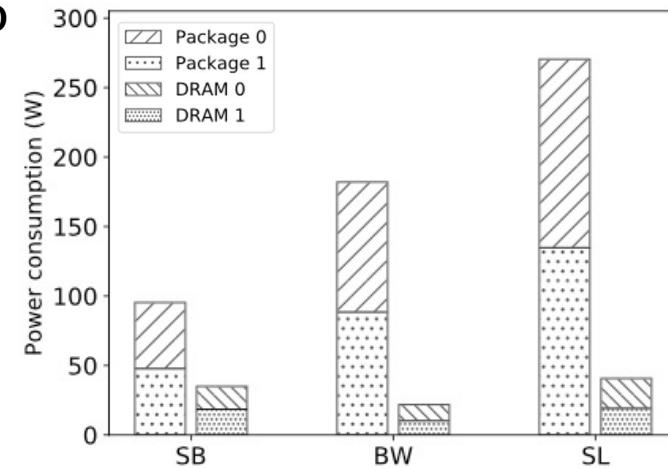


<sup>1</sup>: Top500 - <https://www.top500.org>

<sup>2</sup>: Asumiendo 5.75kW/casa: <https://www.certicalia.com/blog/cuenta-electricidad-consume-una-casa>

# Consumo energético

- Ejemplo de análisis del consumo energético
  - Utilizando 5 aplicaciones como benchmarks
  - Servidores con 2 CPUs, modelos de las tablas anteriores
    - Sandy Bridge (SB), Broadwell (BW) y Skylake (SL)
    - Consumo de CPU (Package) y DRAM
- Promedio de consumo energético al ejecutar los benchmarks:



Resultados de: U. Lopez-Novoa. *Exploring Performance and Energy Consumption Differences between Recent Intel Processors* (2019) <https://ieeexplore.ieee.org/abstract/document/9060093>



# Bibliografía

Este tema está parcialmente basado en:

- Pablo Abad Fidalgo, José Ángel Herrero Velasco.  
"Advanced Linux System Administration", OCW UNICAN,  
2018:
  - Topic 8: Resource management
  - Topic 9: Logging
  - Publicado bajo licencia Creative Commons BY-NC-SA 4.0
  - <https://ocw.unican.es/course/view.php?id=241>
- Consultados en julio 2020



# Servicios en red

# Administración de Sistemas

Unai Lopez Novoa  
unai.lopez@ehu.eus



Universidad  
del País Vasco Euskal Herriko  
Unibertsitatea

# Contenido

1. Intercambio de ficheros en red
  - NFS
2. Intercambio de mensajes en red
  - MQTT
  - Kafka



# Introducción

- Los servicios en red son necesarios para habilitar la comunicación entre aplicaciones que funcionen en diferentes equipos o máquinas virtuales.
- En este tema nos vamos a enfocar en servicios para intercambiar ficheros y mensajes por red.



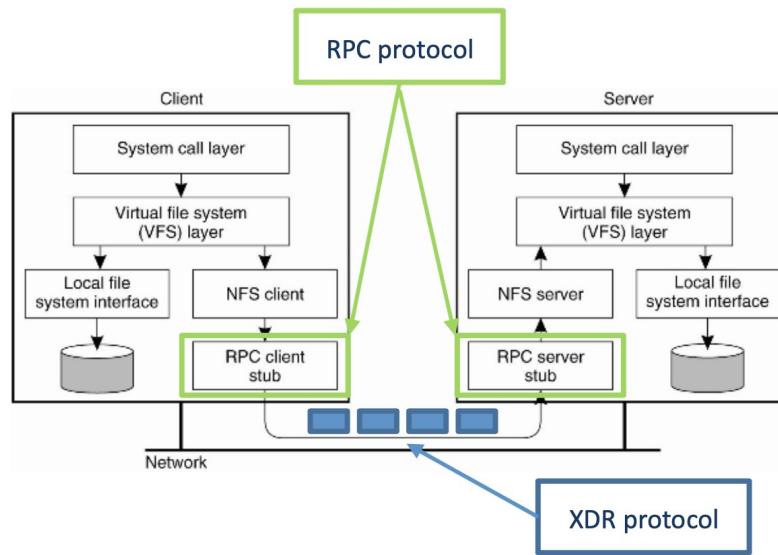
# Sistemas de ficheros distribuidos

- Permiten acceder a datos en equipos remotos de manera transparente.
  - Se encargan de almacenar, recuperar, listar, compartir y asegurar la seguridad de los datos.
- Clasificación (*y ejemplos*):
  - DFS y sistemas Cluster: *GlusterFS, HDFS, ...*
  - Sistemas en la nube: *Dropbox, Oracle Cloud, ...*
  - Sistemas de ficheros paralelos: *PVFS2, orangeFS, ...*
  - Sistemas de ficheros en red: *NFS, pNFS, ...*



# NFS

- Network File System (NFS) es un protocolo utilizado para sistemas de ficheros distribuidos.
  - La primera versión fue desarrollada en 1984.
  - La última versión es la v4.
    - v4.2 publicada en 2016: [RFC 7862](#)
- Esquema:



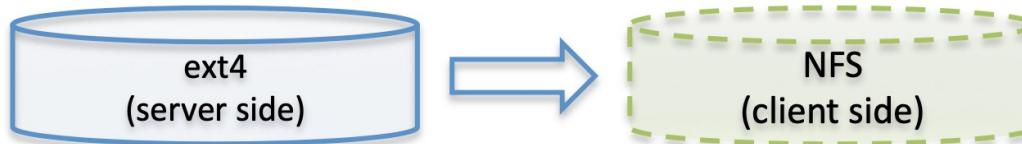
# NFS

- El protocolo define una serie de reglas que permiten compartir ficheros sobre TCP/IP de una manera transparente.
- Es un protocolo sin estado.
  - Cada llamada tiene toda la información necesaria para completar la tarea.
  - El servidor no guarda información entre llamadas.
- Interoperabilidad entre NFS v2, v3 y v4.



# NFS v4

- Punto de vista del usuario:
  - No hay diferencias entre el sistema de ficheros local y remoto.



- Transparencia en la usabilidad.
  - En los clientes, los datos se acceden a través de un punto de montaje.
- La diferencia de rendimiento en el acceso dependerá del rendimiento de los discos y de la red
  - También del número de clientes que acceden al servidor NFS.
- Visión uniforme de los ficheros.
  - Los ficheros del usuario están disponibles desde cualquier cliente de la red.

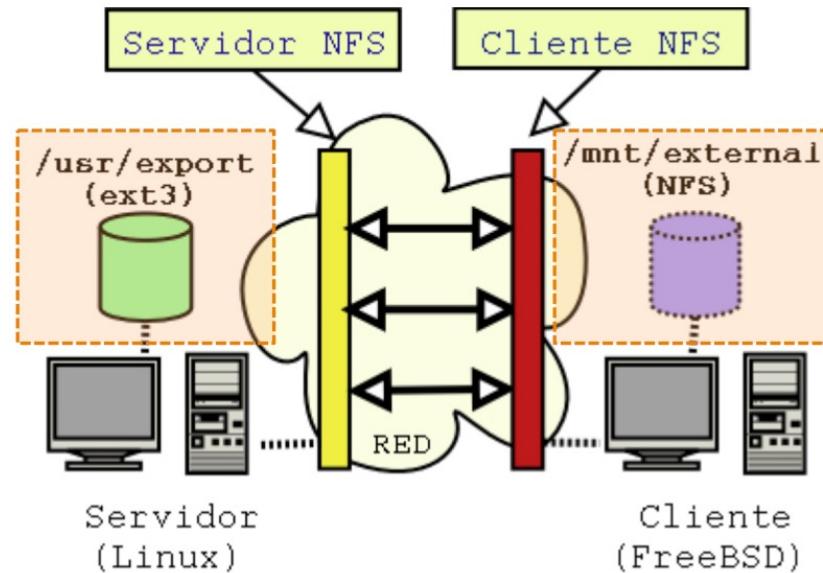
# NFS v4

- Punto de vista del administrador del sistema:
  - Los datos están centralizados.
    - *Pro*: único punto gestión (actualizaciones, backups, ...)
    - *Con*: único punto de fallo
  - La optimización es mas sencilla.
    - Único servidor en el que implementar RAID/LVM.
  - Aislamiento.
    - Los discos están en servidores no accesibles por los usuarios.
  - Tolerante a fallos (parcialmente).
    - *Pro*: los fallos críticos en los clientes no implican pérdida de datos.
    - *Con*: si el servidor se cae, todos los usuarios pierden acceso a sus ficheros.
  - Único puerto: TCP 2049
    - Desde v4.



# NFS v4

- Arquitectura interna
  - Cliente: fichero de configuración de montaje
  - Servidor: Exports con el fichero /etc/exportfs



# NFS v4

- Instalación del servicio.
  - Ejemplos para Debian/Ubuntu.

- En el cliente:

```
apt update  
apt install nfs-common
```

- En el servidor:

```
apt update  
apt install nfs-kernel-server nfs-common
```



# NFS v4

- Configuración del servidor: fichero /etc(exports
  - Controla qué sistemas de ficheros se exportan a las máquinas remotas (y cómo).
  - Cada línea tiene la siguiente estructura:

directorio      cliente (opción1, opción2, ...)  
*donde:*  
    directorio      Es el directorio del servidor a compartir  
    cliente          IP o nombre de dominio, se pueden utilizar wildcards (\*, ?)  
    opciones        Listado de 0 o más opciones
- Por ejemplo:

```
/var/nfs/general  client_ip(rw, sync, no_subtree_check)
/home              client_ip(rw, sync, no_root_squash, no_subtree_check)
```

- Las líneas en blanco se omiten y los comentarios se hacen con #



# NFS v4

- Configuración del servidor: fichero /etc(exports
  - Opciones:

ro	Acceso sólo lectura
rw	Acceso de lectura y escritura
sync	Contestar a peticiones sólo cuando los cambios se hayan escrito a disco.
wdelay	Retrasa la escritura a disco si prevé que otra escritura a disco es inminente (opuesto a sync).
no_subtree_check	Impide la lectura de subdirectorios.
root_squash	Impide que usuarios conectados como "root" en los clientes tengan permisos root en el servidor y les asigna el usuario NFS nobody:nogroup
no_root_squash	Permite privilegios "root" (opuesto a root_squash).
acl	Activa el uso de listas de control de acceso (ACLs).



# NFS v4

- Configuración del servidor: fichero /etc(exports
  - Si no se proporcionan opciones, NFS toma por defecto:
    - ro, wdelay, root\_squash
  - El resto de las opciones se pueden consultar en:
    - <https://linux.die.net/man/5/exports>



# NFS v4

- Configuración del servidor: permisos
  - Hay que tener en cuenta los permisos de las carpetas a compartir
  - Si el UID de un usuario en el servidor coincide con el UID de un usuario en el cliente, los permisos se mantienen.
  - Si root\_squash está activo, las operaciones de root en cliente se registrarán como operaciones de nobody:nogroup en el servidor
    - Si hay alguna carpeta compartida en el servidor que pertenezca a root, es conveniente cambiar los permisos a nobody:

```
sudo chown nobody:nogroup /nfs/general
```



# NFS v4

- Configuración del servidor
  - Comprobar los sistemas exportados por NFS
    - Definidos en /etc/exports

```
exportfs -v
```

- Aplicar la configuración de NFS

```
exportfs -ra
```

- Reiniciar el servicio NFS

```
service nfs-kernel-server restart
```



# NFS v4

- Cliente: Crear un punto de montaje

- Elegir un directorio para el montaje
  - Alternativamente, crear uno:

```
mkdir -p /tmp/nfs
```

- Montar el directorio remoto usando la IP o nombre de dominio

```
sudo mount -t nfs 192.168.0.8:/var/nfs/general /tmp/nfs
```

- Verificar que el montaje se ha hecho correctamente

```
df -h
```

Filesystem	Size	Used	Avail	Use%	Mounted on
...					
192.168.0.8:/var/nfs/general	25G	1.8G	23G	8%	/tmp/nfs



# NFS v4

- Cliente: Montaje permanente

- Fichero /etc/fstab

- Añadir una entrada para cada export NFS:

<servidor-NFS>:<directorio> <directorio-local>

nfs

(opciones) 0 0

dump

fsck

- Opciones:

ro/rw Montar como sólo lectura/lectura y escritura

hard En caso de desconexión del servidor NFS, las aplicaciones usando NFS esperan hasta re-conexión y no se pueden matar.

soft En caso de desconexión del servidor NFS, las aplicaciones esperan un tiempo determinado y después lanzan un error.

intr Junto con la opción *hard*, permite que las aplicaciones esperando por la re-conexión del servidor NFS se puedan matar.

timeo Junto con la opción *soft*, define el tiempo a esperar.

noexec Impide la ejecución de binarios o scripts en un directorio NFS.



# Ejercicio 1

- *Este ejercicio se realizará en parejas*

- Un miembro de la pareja será **C**, el otro **S**. Cada uno usa su MV.
  - **C** será el cliente NFS, **S** el servidor NFS

Puede ser necesario  
abrir los puertos TCP  
y UDP 2049 en GCP

- 1) **S** comparte por NFS una carpeta */compartir*

- La carpeta debe contener un fichero *texto.txt* con texto aleatorio.
- Compartir de forma que **C** sólo pueda leer, no modificar.

- 2) **C** monta la carpeta en */tmp/compartido*

- Verificar que el fichero no se puede modificar.

- 3) **S** modifica las opciones NFS para permitir cambios en los ficheros de la carpeta.

- **C** realiza algún cambio en el fichero, **S** lo verifica.



# MQTT

- Intercambiar ficheros es fundamental para compartir datos y configuraciones en sistemas distribuidos.
- *Pero...*
- En muchas situaciones es mucho más ágil intercambiar mensajes.
  - Evita las complejidades de manipular ficheros.



# MQTT

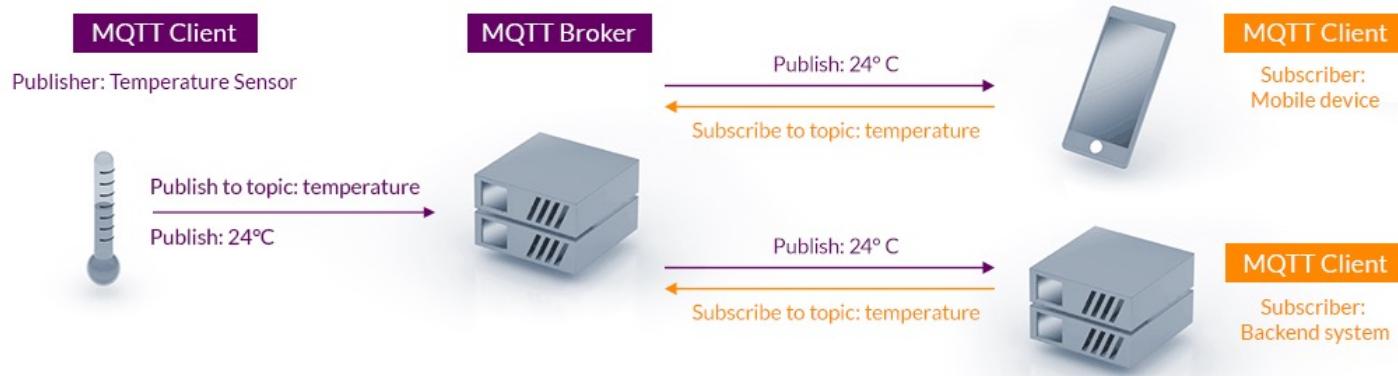


- Message Queue Telemetry Transport
- Es un protocolo orientado a paso de mensajes en formato publicar-suscribir
  - En contraposición al cliente-servidor.
  - Separa al cliente que envía mensajes del que recibe mensajes.
- Web: <https://mqtt.org/>



# MQTT

- Arquitectura:



- Partes:

- **Publishers:** Envían mensajes.
- **Broker:** Recibe mensajes y los distribuye a los suscriptores.
- **Subscribers:** Reciben mensajes enviados por los Publisher.

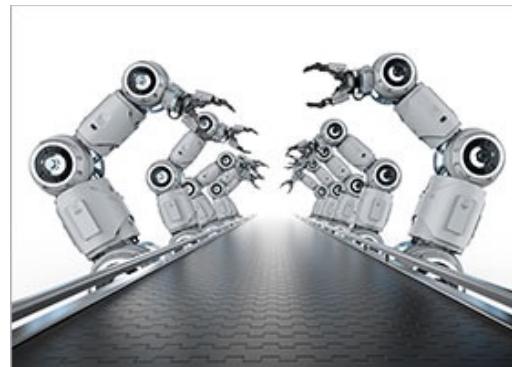


# MQTT

- Es muy usado en entornos Internet of Things (IoT)
  - Permite gestionar cientos/miles de clientes ligeros.
- Por ejemplo:



Logística



Industria



Domótica

# MQTT: Topics

- El proceso Broker filtra los mensajes en base a *topics*.
  - Un *topic* es un String de texto que identifica mensajes de una temática concreta, por ejemplo:  
**edificio/planta 1/sala7/temperatura**
  - Los *topic*:
    - Pueden ser multi-nivel, se separan mediante /
    - Son sensibles a mayúsculas-minusculas y pueden contener espacios
    - No deben comenzar por \$
- También se pueden filtrar por contenido o tipo.
  - Pero no son las formas más usadas.



# MQTT: Topics

- Se pueden utilizar *wildcards* al dirigirse a *topics*:

Un sólo nivel: +



- ✓ myhome / groundfloor / livingroom / temperature
- ✓ myhome / groundfloor / kitchen / temperature
- ✗ myhome / groundfloor / kitchen / brightness
- ✗ myhome / firstfloor / kitchen / temperature
- ✗ myhome / groundfloor / kitchen / fridge / temperature

Multi-nivel: #



- ✓ myhome / groundfloor / livingroom / temperature
- ✓ myhome / groundfloor / kitchen / temperature
- ✓ myhome / groundfloor / kitchen / brightness
- ✗ myhome / firstfloor / kitchen / temperature



# Mosquitto



- Framework que implementa MQTT
  - Open source
  - Web: <https://mosquitto.org/>
- Es muy utilizado en entornos IoT
  - Es ligero, configurable y extensible mediante plug-ins
- En estas diapositivas veremos su uso básico.



# Mosquitto

- Instalación
  - En Debian/Ubuntu:

```
apt update  
apt install mosquitto mosquitto-clients
```

- Verificar que el servicio está en marcha

```
service mosquitto status
```



# Mosquitto

- Clientes de línea de comando
  - Publicar un mensaje en un *topic*

```
mosquitto_pub -h <host> -t <topic> -m <mensaje> <opc>
```

- donde:
  - host: IP del Broker
  - topic: String de texto que indica el topic
  - mensaje: String de texto con el mensaje a publicar
  - opciones: P.e. -d para activar el modo *debug* y mostrar más información
- Ejemplo:

```
mosquitto_pub -h 127.0.0.1 -t "miTopic" -m "Hola" -d
```



# Mosquitto

- Clientes de línea de comando
  - Suscribirse a los mensajes de un *topic*

```
mosquitto_sub -h <host> -t <topic> <opc>
```

- donde:
  - host: IP del Broker
  - topic: String de texto que indica el topic
  - opciones: P.e. *-d* activa el modo *debug*, *-v* activa el modo *verbose*
- Ejemplo:

```
mosquitto_sub -h 127.0.0.1 -t "miTopic" -v -d
```



# Mosquitto

- Configuración del servicio:
  - Fichero /etc/mosquitto/mosquitto.conf

```
# Place your local configuration in /etc/mosquitto/conf.d/
...
log_dest file /var/log/mosquitto/mosquitto.log
include_dir /etc/mosquitto/conf.d
```

- Añadir las siguientes líneas para:
  - Permitir conexiones al puerto 1883 desde fuera del servidor

```
listener 1883 0.0.0.0
```

- Permitir conexiones anónimas (sin usar usuario/contraseña)

```
allow_anonymous true
```



# Mosquitto

- Por defecto, MQTT sólo entrega mensajes a los suscriptores de un *topic* que estén conectados.
  - Un suscriptor no recibirá mensajes que se publiquen mientras esté desconectado del Broker.
- Se puede hacer que los suscriptores reciban el último mensaje que se envió al *topic*.
  - Se denominan *mensajes retenidos*
  - Se puede publicar un mensaje retenido con el parámetro “-r”
    - Ejemplo:

```
mosquitto_pub -h ... -t ... -m "Mi nuevo mensaje" -r
```



# Mosquitto

- Existen librerías de Mosquitto para múltiples lenguajes de programación:
  - P.e. Paho para Python: <https://pypi.org/project/paho-mqtt/>
- Se puede configurar un nivel QoS en los envíos:
  - +info: <https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels/>
- Se pueden configurar un sistema de autenticación:
  - +info: <https://mosquitto.org/documentation/authentication-methods/>



# Ejercicio 2

- *Este ejercicio se realizará en parejas (esta vez, seréis A y B)*
  - A será un Publisher MQTT, B será un Subscriber MQTT
  - B ejecutará el servidor y Broker Mosquitto
- B se suscribe a un *topic* “ciudades/bizkaia”
- A publica un mensaje “Bilbao” en “ciudades/bizkaia”
  - B debe verificar que lo recibe
- B se suscribe a un *topic* “ciudades” y todos sus *sub-topics*.
- A publica un mensaje “Donostia” en “ciudades/gipuzkoa”
  - B debe verificar que lo recibe
- B se desconecta del Broker Mosquitto
- A envía un mensaje retenido “Gasteiz” en “ciudades/araba”
- B se suscribe a “ciudades/#”. Debe recibir el mensaje “Gasteiz”.

Puede ser necesario  
abrir el puerto TCP  
1883 en GCP



# Apache Kafka

- En muchos entornos necesitamos que los mensajes enviados por un cliente no se pierdan.
  - Aunque no haya “suscriptores” conectados.
- En estos entornos se debe utilizar un sistema que gestione la persistencia.
  - MQTT no es un sistema de colas.
    - A pesar de la Q en su nombre.
  - Apache Kafka es uno de los más utilizados.



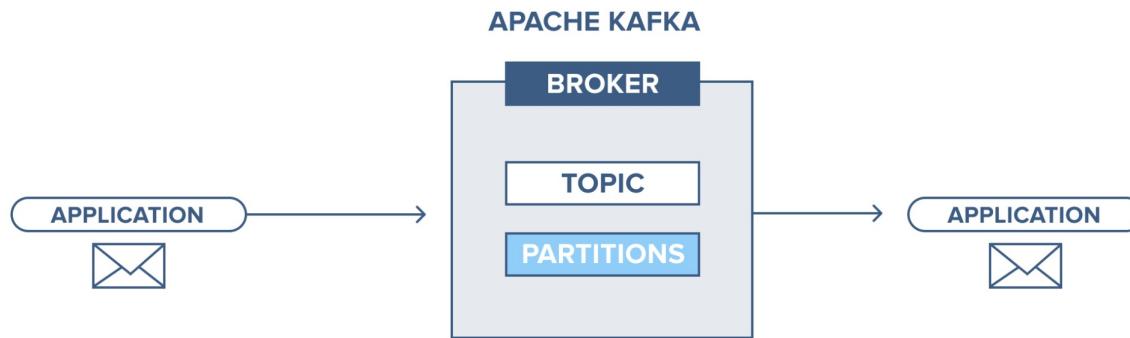
# Apache Kafka



- Sistema distribuido de gestión de mensajes
  - Open Source
  - Web: <https://kafka.apache.org/>
- Escalable, robusto y confiable
  - Más complejo que Mosquitto
  - En estas diapositivas veremos su uso básico.
- Muy utilizado en entornos Big Data, p.e.:
  - Industria 4.0, para monitorizar máquinas de fabricación
  - Sector financiero, para recopilar datos de diferentes fuentes

# Apache Kafka

- Arquitectura
  - De forma general, un sistema de comunicación entre aplicaciones

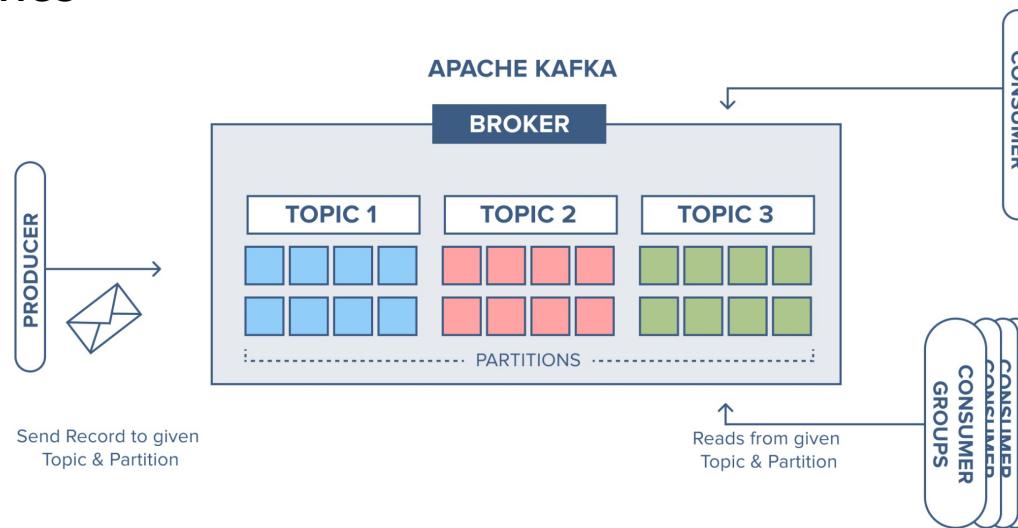


- Broker: Servicio gestor de los mensajes
- Topic: Categoría que sirve para ordenar mensajes
- Particiones: Almacenan los mensajes de los *topic*



# Apache Kafka

- Arquitectura
  - Provee persistencia y replicación a nivel de *topic* usando particiones



- Productor: equivalente a *Publisher* en MQTT
- Consumidor: equivalente a un *Subscriber* en MQTT



# Apache Kafka

- Instalación en un sistema Ubuntu
  - Kafka funciona sobre Java

```
wget https://downloads.apache.org/kafka/3.2.3/kafka_2.12-3.2.3.tgz  
tar xvfz kafka_2.12-3.2.3.tgz  
sudo apt install default-jre
```

Necesario sólo si Java no está instalado

- Iniciar un Broker
  - Requiere iniciar ZooKeeper, un servicio coordinador
  - Estos comandos cargan la configuración por defecto
    - Deben usarse desde la carpeta en la que se ha descomprimido Kafka

```
bin/zookeeper-server-start.sh config/zookeeper.properties  
bin/kafka-server-start.sh config/server.properties
```



# Apache Kafka

- En Kafka, los *topic* deben crearse de antemano
  - A diferencia de MQTT
- Los mensajes de un *topic* se almacenan de forma ordenada y persistente.
- Desde línea de comandos:
  - Ejemplo para crear un *topic* “miTopic” en un Broker local.

```
bin/kafka-topics.sh --create --topic miTopic --bootstrap-server  
localhost:9092
```



# Apache Kafka

- Herramientas de línea de comando para uso básico:

- Consumir eventos de un *topic*

- El parámetro *--from-beginning* recupera todos los eventos del topic.
    - El último parámetro indica la IP y puerto del Broker.
    - Ejemplo:

```
bin/kafka-console-consumer.sh --topic miTopic --from-beginning  
--bootstrap-server localhost:9092
```

- Enviar eventos a un *topic*

- Cada nueva línea se envía como un evento.
    - El último parámetro indica la IP y puerto del Broker.
    - Ejemplo:

```
bin/kafka-console-producer.sh --topic miTopic --bootstrap-server  
localhost:9092
```



# Apache Kafka

- Configuración de Kafka
  - Fichero config/server.properties
  - Contiene la configuración del Broker en el nodo local
- Para permitir conexiones desde cualquier IP, añadir las siguientes líneas<sup>1</sup>:
  - Incluir la IP pública de vuestra máquina en la 2<sup>a</sup> línea

```
##### Socket Server Settings #####
...
listeners=PLAINTEXT://:9092
advertised.listeners=PLAINTEXT://<IP-publica>:9092
```



# Apache Kafka

- Existen librerías de Kafka para múltiples lenguajes de programación.
  - P.e. en Java: <https://learn.microsoft.com/es-es/azure/hdinsight/kafka/apache-kafka-producer-consumer-api>
- Se pueden configurar réplicas para mayor tolerancia:
  - +info: [https://medium.com/@\\_amanarora/replication-in-kafka-58b39e91b64e](https://medium.com/@_amanarora/replication-in-kafka-58b39e91b64e)
- Incluye múltiples mecanismos de seguridad:
  - +info: <https://kafka.apache.org/documentation/#security>



# Ejercicio 3

- *Este ejercicio se realizará en parejas (seréis A y B)*
  - A será productor de mensajes
  - B será consumidor y ejecutará el Broker Kafka
- B crea un *topic* llamado “colores” y se suscribe a él.
- A envía mensajes “verde” y “azul” al *topic*.
- B se desconecta del *topic*.
- A envía mensajes “amarillo” y “naranja” al *topic*.
- B se suscribe al *topic*. Debe recibir todos los mensajes.

Puede ser necesario  
abrir el puerto TCP  
9092 en GCP



# Bibliografía I

- V. Puente Varona, J.A. Herrero Velasco, P. Abad Fidalgo. "Computer System Design and Administration", OCW UNICAN, 2018:
  - Topic 7: Network file system service: NFSv4.
  - Publicado bajo licencia Creative Commons BY-NC-SA 4.0
  - <https://ocw.unican.es/course/view.php?id=245>
- Brian Boucheron "How To Set Up an NFS Mount on Ubuntu 20.04", Digital Ocean, 2020:
  - <https://www.digitalocean.com/community/tutorials/how-to-set-up-an-nfs-mount-on-ubuntu-20-04>
- Consultados en julio 2020



# Bibliografía II

- Mosquitto Documentation, 2022.
  - <https://mosquitto.org/documentation/>
- L. Llamas, "Cómo instalar Mosquitto", luisllamas.es, 2020.
  - <https://www.luisllamas.es/como-instalar-mosquitto-el-broker-mqtt/>
- Apache Kafka Documentation, "Getting Started", 2022.
  - <https://kafka.apache.org/documentation/>
- L. Johansson, "Apache Kafka for beginners: What is Apache Kafka?", CloudKarafka, 2020.
  - <https://www.cloudkarafka.com/blog/part1-kafka-for-beginners-what-is-apache-kafka.html>
- Consultados en septiembre 2022

