1. CLI

1. Instalación de Vagrant y arranque de una VM (vagrant_cli). Conexión con vagrant ssh

a. Descargaremos el software necesario adecuado al sistema operativo que usemos:
   i. https://www.virtualbox.org/wiki/Downloads
   ii. https://www.vagrantup.com/downloads
b. Instalaremos Git si no lo tenemos ya instalado
   i. Windows: https://git-scm.com/download/win
   ii. Ubuntu: apt install git
c. Clonaremos el repositorio del curso
   i. git clone https://github.com/KeepCodingCloudDevops5/sysadmin.git
   ii. Si tenéis clave rsa -> git clone git@github.com:KeepCodingCloudDevops5/sysadmin.git
d. Nos movemos al directorio vagrant_cli dentro del repositorio y ejecutamos **vagrant up**. Si todo ha ido bien nos arrancará una máquina virtual a la que nos conectaremos con **vagrant ssh**.

https://www.linode.com/lp/affiliate-referral/?irclickid=XiW3l9w9YxyITPtUqswjdSANUkG2suShGWtsQM0&irgwc=1&utm_source=impact
https://www.digitalocean.com/?refcode=8223070813dd&utm_campaign=Referral_Invite&utm_medium=Referral_Program&utm_source=CopyPaste

2. Movimiento por directorios

a. Abrir un terminal, teclear **cd**, a continuación **pwd** para ver el directorio de trabajo
b. Teclear **touch file1** para crear un fichero vacío. En el home se pueden crear todas los ficheros que uno quiera.
c. Teclear **cd /**. Cambiará el directorio actual a la raíz. Teclear **touch file2** y deberíamos tener un error de permisos denegados.
d. Teclear **cd /tmp** nos moverá al directorio temporal donde todos los usuarios tienen permisos. Teclea de nuevo **touch file2** y debería crearlo sin problema.
e. Teclea de nuevo **cd** para volver al home.
f. Creamos un directorio nuevo con **mkdir files**.
g. Creamos un directorio con la ruta absoluta con **mkdir /home/$USER/files**. Deberíamos tener un error de que el fichero ya existe.
h. Usaremos **rmdir files** para borrar este directorio.

3. Creación y copia de ficheros

a. Abrir un terminal
b. Teclear **pwd** para verificar que estamos en el home del usuario
c. Teclear **mkdir newfiles** y **mkdir oldfiles**. Con **ls** veremos los directorios que se acaban de crear.
d. Teclea **touch newfiles/.ocultos** y **touch newfiles/visibles**. Esto creará dos ficheros en el directorio newfiles.
e. Entramos en el directorio oldfiles con **cd oldfiles**.
f. Con **ls -la** veremos que hay dos ítems: . hace referencia al directorio actual y .. hace referencia al directorio padre.
g. Ejecuta **ls -la ../newfiles** para ver los ficheros del directorio newfiles. Notar que le estamos pasando una ruta relativa.
h. Copiaremos el directorio newfiles con **cp -a ../newfiles/ .**
i. Con **ls -a** veremos que se ha creado un subdirectorio newfiles en el directorio oldfiles
j. Borramos el directorio newfiles con **rm -rf newfiles**.
k. Usaremos el comando **cp -a ../newfiles/* .** y a continuación **ls -la** para ver que se ha copiado esta vez. Veremos que el fichero oculto no se ha copiado
l. Para copiar todos los ficheros ejecutaremos **cp -a ../newfiles/. .**
m. Verificamos con **ls -la** que se ha copiado todo esta vez

4. Enlaces

a. Abrir un terminal
b. En el directorio home ejecutar **ln /etc/passwd .**
c. Esta operación dará un error porque el usuario no es propietario del fichero
d. Ejecuta ahora **ln -s /etc/passwd .**
e. En este caso funciona porque es un enlace simbólico
f. Ejecuta **ln -s /etc/hosts** sin especificar el punto final también funciona
g. Crea un fichero nuevo con **touch newfile** y crea un enlace duro a ese fichero con **ln newfile linkedfile**.
h. Ejecuta **ls -l** y verifica el contador de enlaces es actualmente 2
i. Crea un enlace simbólico a newfile con **ln -s newfile symlinkfile**
j. Borra el fichero newfile con **rm newfile**
k. Intenta acceder al enlace simbólico con **cat symlinkedfile** y tendremos un error
l. Sin embargo acceder al enlace duro no dará problemas con **cat linkedfile**
m. Ejecuta un **ls -l** para verificar que está pasando
n. Ejecuta **ln linkedfile newfile** para recrear el enlace
o. Ejecuta **ls -l** otra vez para verificar si la situación se ha restaurado

5. Head y tail

a. Ejecuta **sudo tail -f /var/log/syslog** para ver las últimas líneas del fichero /var/log/syslog.
b. Ejecuta **Ctrl+C** para salir del comando.
c. Ejecuta **head -n 5 /etc/passwd** para ver las 5 primeras filas del fichero /etc/passwd.
d. Ejecuta **tail -n 2 /etc/passwd** para ver las últimas dos filas del fichero /etc/passwd.
e. Ejecuta **head -n 5 /etc/passwd | tail -n 1** para ver la quinta línea del fichero /etc/passwd.
f. Ejecuta **cut -f 1 -d : /etc/passwd** para obtener la primera columna con el listado de usuarios del sistema
g. Ejecuta **sort /etc/passwd** para obtener el contenido del fichero ordenado alfabéticamente
h. Ejecuta **ps aux | wc** para obtener el número de líneas, palabras y caracteres de la entrada

6. Grep

a. Ejecuta **grep '^#' /etc/ssh/sshd_config**. Mostrará que el fichero /etc/ssh/sshd_config contiene un gran número de líneas que empiezan por # y están comentadas.
b. Para ver las líneas del archivo de configuración que realmente importan ejecuta **grep -v '^#' /etc/ssh/sshd_config**. Esto mostrará las líneas que no empiezan por #.
c. Ahora ejecuta **grep -v '^#' /etc/ssh/sshd_config -B 5**. Esto mostrará líneas que no empiecen por # y además las cinco líneas previas a esta línea, esto es útil porque en esas líneas normalmente se encuentran comentarios
d. Ejecuta **grep -v -e '^#' -e '^$' /etc/ssh/sshd_config**. Excluye líneas en blanco y las que empiezan por #.

7. Expresiones regulares con grep

a. https://www.cyberciti.biz/faq/grep-regular-expressions/

8. Reemplazar texto con sed

a. https://www.cyberciti.biz/faq/how-to-use-sed-to-find-and-replace-text-in-files-in-linux-unix-shell/

9. Tuberías y salida estándar

a. Abre un terminal y ejecuta **cd** sin argumentos para asegurarnos de que estamos en el home del usuario. Ejecuta **pwd** para verificarlo
b. Ejecuta **ls**. Veras los resultados en la pantalla
c. Ejecuta **ls > /dev/null**. Esto redirecciona la salida al dispositivo null, haciendo que no se vea nada
d. Ejecuta **ls ilwehgi > /dev/null**. Este comando devolverá un error porque éste no se escribe en el STDOUT si no en el STDERR
e. Ejecuta **ls ilwehgi 2> /dev/null**. El error se redirecciona al dispositivo null
f. Crea el directorio Documents con **mkdir Documents**. Y crea varios ficheros dentro de él con **touch Documents/Doc1.txt Documents/Doc2.tx**t
g. Ejecuta **ls ilwehgi Documents 2> /dev/null**. Mostrará el nombre del directorio Documents ocultando el mensaje de error.
h. Ejecuta **ls ilwehgi Documents 2> /dev/null > output**. Seguiremos ocultando el mensaje de error redirigiendo el STDERR a /dev/null y además enviaremos la salida STDOUT a un fichero con nombre output.
i. Ejecuta **cat output** para ver el contenido del fichero.
j. Ejecuta **echo hello > output**. Sobreescribimos el contenido del fichero.
k. Ejecuta **ls >> output**. Añade al final del fichero la salida del comando ls.
l. Ejecuta **ls -R /**. Muestra la lista de ficheros del sistema, puede tardar mucho con lo que se puede parar con Ctrl+C.
m. Ejecuta **ls -R / | less**. Muestra el mismo resultado pero a través de less, donde se puede navegar hacia arriba y abajo con las teclas de flechas.
n. Pulsa **q** para cerrar less.
o. Ejecuta **ls > /dev/tty1**. Devuelve un error porque se está ejecutando el comando como un usuario normal, solo el usuario root tiene permisos de escritura en dispositivos de ficheros
p. Ejecuta **w** para saber en qué terminal estamos conectados.
q. Envía un string al terminal con **echo "hola mundo" > /dev/pts/0** (cambiar por el terminal que hemos obtenido antes)

10. Control de procesos

a. Abre un terminal y cambia al usuario root con **sudo su -**.
b. Ejecuta los siguientes comandos:
   sleep 3600 &
   dd if=/dev/zero of=/dev/null &
   sleep 7200
c. Como no se ha utilizado el caracter & después del último comando tendremos que esperar 1 hora antes de volver a tener el control de la shell. Teclea **Ctrl+Z** para pararlo
d. Ejecuta **jobs**. Se verá que están los 3 jobs que se acaban de arrancar. Los dos primeros en estado running y el último en stopped.

e. Ejecuta **bg 3** para reanudar la ejecución del job 3 en background. También se podría ejecutar directamente bg sin el 3 ya que se arrancó en último lugar.
f. Ejecuta **fg 1** para obtener el control del job 1.
g. Pulsa **Ctrl+C** para cancelar el job número 1 y usa jobs para confirmar que ya no existe.
h. Usa el mismo procedimiento para parar los jobs 2 y 3.
i. Abre un segundo terminal y ejecuta **dd if=/dev/zero of=/dev/null &**.
j. Ejecuta **exit** para salir de este segundo terminal.
k. Desde el primer terminal arranca **top**. Deberás ver que el proceso dd está todavía corriendo. En top, usa **k** para parar el job dd.

11. Nice y kill

a. Abre un terminal y cambia al usuario root con **sudo su -**.
b. Ejecuta **q** y repítelo tres veces.
c. Ejecuta **ps aux | grep dd**. En la salida nos filtrará por todas las líneas que tengan dd en ella, no solo la de nuestros procesos de dd.
d. Usa el PID de uno de los procesos de dd para ajustar la prioridad a la baja (nice) usando **renice -n 5 <PID>**.
e. Abriremos **top** y verificaremos que el proceso al que le hemos cambiado la prioridad tiene menos tiempo de CPU asignado. Salimos de top con **q**.
f. Ejecuta **ps fax | grep -B5 dd**. Veremos la relación jerárquica entre procesos, pudiendo encontrar fácilmente el PID desde el que todos los procesos dd se arrancaron.
g. Encuentra el PID de la shell y ejecuta **kill -9 <PID>** para matar todos los procesos de una vez. También matará la shell con lo que nos desconectará de la sesión.

12. Creación de usuarios

a. Abre un terminal y cambia al usuario root con **sudo su -**.
b. Ejecuta **vim /etc/login.defs** para abrir el fichero de configuración de logins y cambiar algunos parámetros antes de crear nuevos usuarios. Busca el parámetro **CREATE_HOME** y setealo a "**yes"** (si no existe se puede añadir al final del fichero). El parámetro **USERGROUPS_ENAB** lo cambiaremos a "**no**". Esto hará que los nuevos usuarios los añada al grupo por defecto con GID 100 o el especificado en /etc/default/useradd.
c. Con **cd /etc/skel** nos movemos al directorio /etc/skel. Ejecutaremos **mkdir Pictures** y **mkdir Documents** para añadir dos directorios por defecto al home de todos los nuevos usuarios. Además, cambiaremos el contenido del fichero **.bashrc** añadiendo la línea **export EDITOR=/usr/bin/vim** al final, esto hará que configuremos el editor por defecto usado a la hora de modificar ficheros de texto.

d. Ejecuta **useradd keepcoding** para crear una cuenta para el usuario keepcoding. Después, ejecuta **id keepcoding** para verificar que keepcoding es miembro del grupo users y de ningún otro grupo. También podemos verificar que se han creado los directorios Pictures and Documents en el home del usuario keepcoding.
e. Usa **passwd keepcoding** para configurar la contraseña del usuario que acabamos de crear.
f. Ejecuta **passwd -n 30 -w 3 -x 90 keepcoding** para cambiar las propiedades del password. Ahora la contraseña expirará tras 90 días (-x 90). Tres días antes de expirar, el usuario recibirá un aviso (-w 3), y la contraseña tiene que tener una antiguedad de al menos 30 dias antes de poder volver a ser cambiada.
g. Creamos algunos usuarios mas: keepadmin, keepbck, and keepdev, usando el siguiente bucle **for i in keepadmin keepbck keepdev; do useradd $i; done**.
h. Con **grep keepadmin /etc/passwd /etc/shadow /etc/group** verificamos que el ususario keepadmin se ha creado correctamente y se han seteado las entradas correspondientes en estos tres ficheros críticos.
i. Crearemos el grupo keepcoding con **groupadd keepcoding** y añadiremos el usuario keepcoding a dicho grupo con **usermod -a -G keepcoding keepcoding**

13. Modificar permisos

a. Abre un terminal y cambia al usuario root con **sudo su -**.
b. Ejecuta **mkdir -p /opt/keepcoding**.
c. Antes de cambiar los permisos, cambiar el owner de este directorio, usando: **chown keepcoding:keepcoding /opt/keepcoding.**
d. Configuraremos los permisos para habilitar al usuario y miembros del grupo a escribir ficheros en este directorio y denegar el acceso a todos los otros usuarios: **chmod 770 /opt/keepcoding**.
e. Usaremos **su - keepcoding** para cambiarnos al usuario keepcoding y nos moveremos al directorio /opt/keepcoding. Con **touch emptyfile** crearemos un fichero vacío. ¿Funciona?
f. Aún como keepcoding, usa **cd /opt** y volvemos a crear otro fichero vacío con **touch emptyfile**. ¿Funciona?

14. Programar tarea con cron

a. Open a root shell. Type **cat /etc/crontab** to get an impression of the contents of the /etc/crontab configuration file.

b. Type **crontab -e**. This opens an editor interface that by default uses vi as its editor. Add the following line:

    0 * * * * logger message from root

c. Use the vi command **:wq!** to close the editing session and write changes.
d. Use **cd /etc/cron.hourly**. In this directory, create a script file with the name **eachhour** that contains the following line:

    logger This message is written at $(date)

e. Use **chmod +x eachhour** to make the script executable; if you fail to make it executable, it will not work.
f. Now enter the directory /etc/crond.d and in this directory create a file with the name **eachhour**. Put the following contents in the file:

    11 * * * * root logger This message is written from /etc/cron.d

g. Save the modifications to the configuration file and go work on the next section. (For optimal effect, perform the last part of this exercise after a couple of hours.)
h. After a couple of hours, type **grep written /var/log/syslog** and read the messages that have been written which verifies correct cron operations.

15. Instalación de software en ubuntu

a. Abre un terminal
b. Actualizaremos la cache de apt con **sudo apt-get update**
c. Busca el nombre del metapaquete correspondiente al servidor web apache2 con **sudo apt-cache search ^apache2**.
d. Verifica si el paquete ya está instalado con **sudo dpkg -l | grep apache**
e. Instalamos el paquete ejecutando **sudo apt-get install apache2**
f. Verificamos con el comando dpkg que se ha instalado correctamente.
g. Con **sudo apt-get upgrade** podemos actualizar todos los paquetes instalados en el sistema a la última versión

16. Módulos del kernel

a. Type **lsmod | grep zstd_compress**.This module should be loaded, and it should also indicate that it is used by the btrfs module.
b. Type **modprobe -r zstd_compress**. This will not work because the module is in use by the btrfs module.
c. Type **modprobe -r btrfs; modprobe -r zstd_compress**. This will unload both modules.
d. Type **modinfo ip_tables**. This will show information about the ip_tables module including the parameters that it supports. One of these is the debug parameter, that supports a Boolean as its value.
e. Now use the command **modprobe libiscsi_tcp debug_libiscsi_tcp=1**. This will load the libiscsi_tcp module with the debug parameter set to on.
f. Type **dmesg**. For some kernel module, load information is written to the kernel ring

buffer which can be displayed using the **dmesg** command.

17. Módulos de kernel

a. Type **lsmod | head**. This shows all kernel modules currently loaded.
b. Type **modprobe xfs** to load the xfs kernel module. Verify that it is loaded, using the **lsmod** command again.
c. Type **modinfo xfs** to get information about the xfs kernel module. Notice that it does not have any parameters.
d. Type **modprobe -r xfs** to unload the xfs kernel module again.
e. Repeat the last steps with the ext4 module. What happended?.

18. Cambio parámetros del kernel

a. Verificar el máximo número de ficheros permitidos: **cat /proc/sys/fs/file-max**
b. Verificar el mismo valor con sysctl: **sysctl -a | grep file-max**
c. Cambio temporal del parámetro: **sysctl -w fs.file-max=100000**
d. Verificamos el cambio con el comando ejecutado en a o b.
e. Cambio permanente. Añadimos el fichero **/etc/sysctl.d/99-sysctl.conf** la línea **fs.file-max = 200000**

    echo fs.file-max = 200000 >> /etc/sysctl.d/99-sysctl.conf
f. Aplicamos los cambios con **sysctl -p**
g. Verificamos si el cambio ha tenido efecto.