

Arquitectura de SO

Clase 1

Teoría

https://www.utnfravirtual.org.ar/pluginfile.php/523117/mod_resource/content/3/Clase%201-parte%201.pdf

https://www.utnfravirtual.org.ar/pluginfile.php/523177/mod_resource/content/4/Clase%201-parte%202.pdf

- Diferencia entre dato e información

Un dato es un dato, pero la información son datos ya procesados. La compu se encarga de, dado un dato que obtiene mediante un prompt, dar un resultado.

- Leer la diferencia entre algoritmo, instrucciones y programa.

Saber lo que es la **RAM**. Random access memory. Acá se almacenan datos de los programas temporalmente.

ROM, read only memory. La BIOS es un ejemplo de esto. Hoy en día se usa un tipo de ROM llamado EEPROM (memoria borrable eléctricamente).

Investigar cómo persiste la información en la computadora cuando la apago. Es decir, cómo funciona un SSD y HDD.

CPU.

Es el microprocesador.

Este contiene la unidad de control, y la unidad aritmética lógica.

Unidad aritmética lógica: hace operaciones aritméticas

Unidad de control controla todo. Revisar la ppt.

Arquitectura de computadoras.

Software:

Diferencia entre lenguajes de bajo y alto nivel.

La diferencia es el nivel de abstracción del hardware.

El primer sistema de procesamiento de datos fue una calculadora, diseñada por DaVinci, e implementada por Pascal.

Comparar la Máquina Analítica de Babbage con el funcionamiento actual de un CPU.

La Mark I, sólo hacía operaciones matemáticas.

1er computadora: ENIAC.

Hacía 5k operaciones por segundo.

Usaban tubos de vacío.

Pesaba 30 toneladas.

Importante!!

Revisar la ppt sobre las **generaciones de computadoras**.

1era generación usaban tubos de vacío.

La 2da generación de computadoras implica el pasaje al uso de **transistores**.

La 3er generación implica el uso de **circuitos integrados**. Multiprogramación.

4ta generación. Nacen los **microprocesadores**.

5ta generación. **IA**. Mejoras de hardware y software. Esto incluye la salida de laptops, smartphones.

Cuadro Comparativo Sobre las Generaciones del Computador						
	1RA GENERACION	2DA GENERACION	3RA GENERACION	4TA GENERACION	5TA GENERACION	6TA GENERACION
PERIODO	1946-1958	1958-1964	1964-1971	1971-1983	1984-1999	2000 a la actualidad
FUNCIONAMIENTO Y TECNOLOGIA	Estaban construidos con electronica de valvulas, usaban tarjetas perforadas para entrar los datos y los programas La tecnologia era a base de bulbos o tubos de vacio	Reemplazo las valvulas de vacio por los transistores para procesar informacion Los equipos estaban construido con la electronica de transistores	Se utilizan los circuitos integrados (abaratia costos y aumentaba la capacidad de procesamiento) Desarrollo de circuitos integrados en los que se les colocaba miles de componentes electronicos en una integracion en miniatura	La integracion se realiza sobre los componentes electronicos. Lo que propicio la aparicion del microprocesador Esta generacion es el producto de la miniaturizacion de los circuitos electronicos	Se crea el multiprocesador La inteligencia artificial (los sistemas expertos, el lenguaje natural, la robtica y el reconocimiento de voz)	Las computadoras de esta generacion cuentan con arquitecturas combinadas paralelo/vectorial, con cientos de microprocesadores vectoriales trabajando trabajando al mismo tiempo
ALMACENAMIENTO	Utilizaban cilindros magneticos para almacenar informacion e instrucciones internas	Uzaban pequeños anillos magneticos para almacenar informacion e instrucciones	Se desarrollo la memoria virtual	Se reemplaza la memoria de anillos magneticos por la memoria de chips de silicio	Se aumenta la capacidad de memoria	La capacidad de memoria es superior a la generacion anterior

Cuestionario en clase:

Test 2 Generaciones y Clasificaciones de Computadoras

Cuestión: 1/20	Aciertos: 1	Fallos: 0	Puntos: 1
----------------	-------------	-----------	-----------

La siguiente imagen representa a una computadora, ¿de qué tipo?

supercomputadora
 macrocomputadora
✓ microcomputadora



Preguntas sobre que usa cada generación de computadoras.

Test 2 Generaciones y Clasificaciones de Computadoras

Cuestión: 5/20	Aciertos: 4	Fallos: 1	Puntos: 4
----------------	-------------	-----------	-----------

Se programaban en lenguaje máquina:

✓ las computadoras de la primera generación
✗ las computadoras de la tercera generación
 las computadoras de la segunda generación

Un ejemplo es los cables que se enchufan y desenchufan

Los lenguajes de alto nivel nacen en la 3er generación.

1eros lenguaje de alto nivel: COBOL, BASIC, PASCAL, FORTRAN, PL-1.

En la rama de la Computación, las siglas PC ¿qué significan?

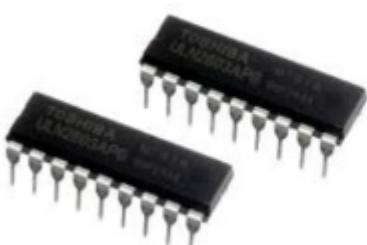
Piece of Cake
✓ Personal Computer
 Piece of Computer

La imagen corresponde al nombre de:



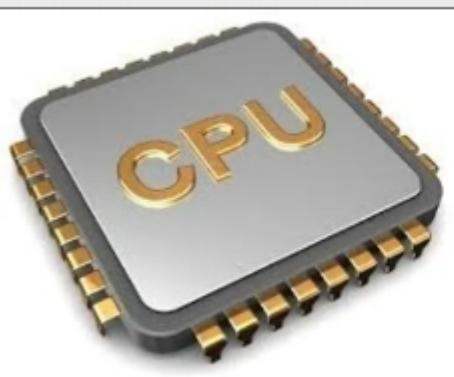
- transistor
- tubos de vacío
- circuito integrado

La imagen corresponde a un:



- circuito integrado
- tubo de vacío
- transistor

La imagen, ¿corresponde al nombre de?

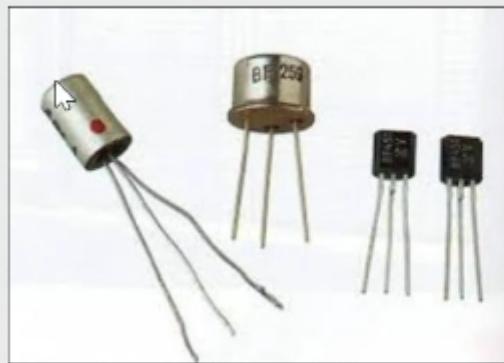


- microporcesador
- transistor
- circuito integrado

Otra forma de llamarle a los circuitos integrados es:

- conjunto de circuitos integrados madre
- gabinete
- chip

La imagen corresponde, ¿a qué nombre?



- transistor
- circuito integrado
- tubo de vacío

Leer la PPT de microprocesadores.

https://www.utnfravirtual.org.ar/pluginfile.php/523179/mod_resource/content/5/CLASE%20de%20microprocesadores.pdf

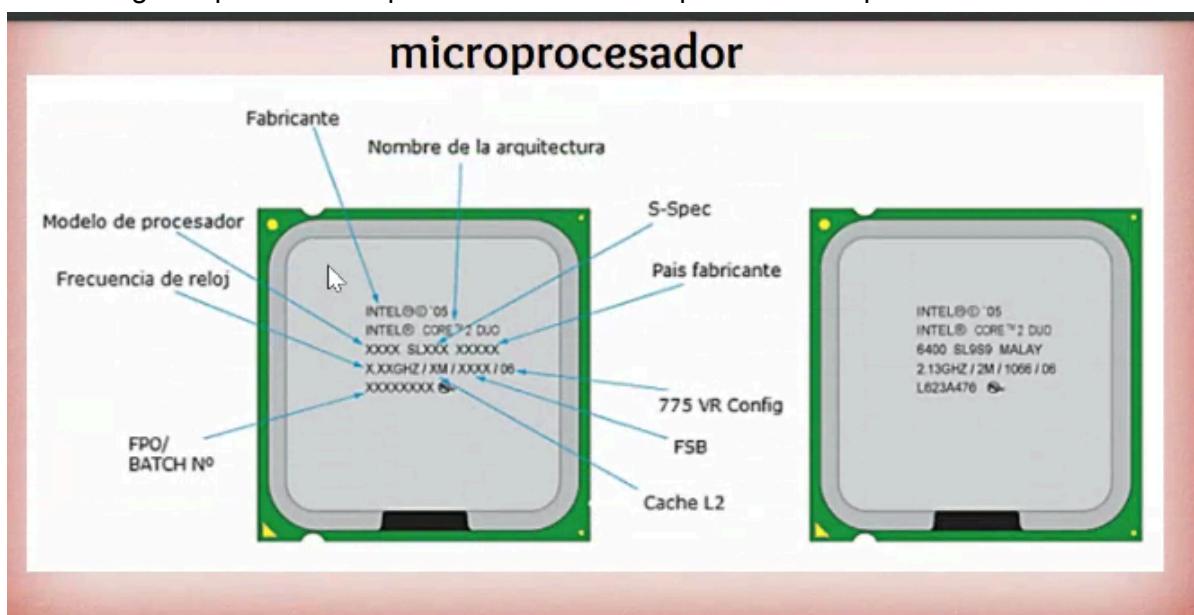
https://www.utnfravirtual.org.ar/pluginfile.php/523181/mod_resource/content/7/Frecuencia%20del%20procesador%20%281%29.pdf

Diferencia entre Multiprogramación y Multiprocesamiento.

Multiprogramación. Esa instrucción se va turnando. Estaba un ratito, se procesaba, salía, venía otra instrucción.

Multiprocesamiento. Procesamiento simultáneo. Es que realmente haya 2 o más instrucciones trabajando en un núcleo.

Esto se logra duplicando componentes dentro de la pastilla del chip.



Tenemos 1 microprocesador con muchos **cores**.

Por cada core vamos a tener 2 hilos.

História de las computadoras (Clase 1)

- Evolución de las computadoras:
 - Desde los primeros dispositivos manuales como el ábaco, pasando por las primeras calculadoras mecánicas (Pascalina, Leibniz), hasta las primeras computadoras electrónicas como el ENIAC y la arquitectura de Von Neumann.
 - Generaciones de computadoras:
 - Primera generación: Válvulas de vacío.
 - Segunda generación: Transistores.
 - Tercera generación: Circuitos integrados.
 - Cuarta generación: Microprocesadores.
 - Quinta generación: Inteligencia artificial.
- Arquitectura moderna:
 - Componentes de la arquitectura de Von Neumann: Unidad de control (UC), Unidad aritmético-lógica (UAL), memoria (RAM y ROM), dispositivos de entrada y salida.
 - Procesadores multinúcleo: Cómo funcionan los núcleos y su impacto en el rendimiento.

Teoría sobre software y sistemas operativos (Clase 2)

- Software:
 - Clasificación por tipo de distribución:
 - Software libre: Definido por las 4 libertades (ejecutar, estudiar, redistribuir, y mejorar).
 - Software propietario: Derechos restringidos sobre su uso, modificación y distribución.
 - Ventajas y desventajas de ambos tipos de software.
- Sistemas operativos:
 - Clasificación por administración:
 - Administración de tareas: Monotarea vs multitarea.
 - Administración de usuarios: Monousuario vs multiusuario.
 - Administración de recursos: Centralizados vs descentralizados.
 - Kernel:
 - Funciones principales: gestión de memoria, procesos, dispositivos, seguridad, etc.
 - Tipos de kernel: Microkernel, monolítico, híbrido.
- Sistema de archivos: Cómo se almacenan y recuperan datos.
 - Diferencias entre los sistemas de archivos de Linux (jerárquico) y Windows (por unidades).
- Interfaz gráfica:
 - Ventajas: Simplicidad, facilidad de uso.
 - Desventajas: Menor velocidad en comparación con las interfaces de línea de comandos.

Teoría Linux (Clase 3)

1. **GNU/Linux y Unix:**
 - GNU/Linux es un sistema operativo libre de tipo Unix, con un núcleo desarrollado por Linus Torvalds en 1991. Es multitarea, multiusuario y de código abierto. Destacan las distribuciones como Debian, Ubuntu y Red Hat.
2. **Arquitectura del núcleo de Linux:**
 - El kernel de Linux es monolítico, modular y multitarea. Los controladores de dispositivos pueden cargarse dinámicamente mientras el sistema está en ejecución, lo que mejora la gestión de hardware y multiprocesamiento.
3. **Sistemas de archivos en Linux:**
 - Clasificación de los directorios en Linux: Estáticos (e.g., `/bin`), dinámicos (e.g., `/var`), compartidos, y restringidos (e.g., `/etc`). Cada uno cumple funciones específicas en la administración y operación del sistema.
4. **Componentes del sistema Linux:**
 - Incluyen el gestor de arranque (GNU GRUB), programa de inicio (`init`), bibliotecas de software (como glibc), y comandos básicos del sistema.
5. **Entornos gráficos y aplicaciones:**
 - Los entornos gráficos como GNOME y KDE permiten una interacción gráfica con el usuario. Las distribuciones incluyen repositorios de software.

Teoría Linux (Clase 4)

1. **Usuarios en Linux:**
 - Linux es multiusuario, y se administra a través de UID y GID. Existen tres tipos de usuarios: root (superusuario), usuarios especiales (sin inicio de sesión), y usuarios normales.
2. **Archivo `/etc/passwd` y `/etc/shadow`:**
 - Estos archivos son fundamentales para gestionar cuentas de usuarios y contraseñas. `/etc/passwd` contiene información básica, mientras que `/etc/shadow` almacena contraseñas cifradas.
3. **Permisos en Linux:**
 - Los permisos de archivos están estructurados en tres niveles: dueño, grupo y otros. Los permisos se representan en formato octal (e.g., `rwx = 7`). Se gestionan con el comando `chmod`.

Teoría Discos (Clase 5)

1. **Particiones de disco:**
 - Las particiones dividen un disco físico en lógicos. Tipos de particiones: primarias, extendidas y lógicas. Las particiones encapsulan datos y pueden usar diferentes sistemas de archivos para mejorar el rendimiento.
2. **Sistemas de archivos:**
 - Estructura jerárquica de directorios, con directorios claves como `/bin` (ejecutables), `/boot` (arranque), `/dev` (dispositivos), `/etc` (configuración), `/home` (usuarios), y `/var` (logs y caché).
3. **RAID:**

- Tecnología que combina múltiples discos duros para mejorar la seguridad y rendimiento. RAID 0 mejora la velocidad, RAID 1 asegura integridad de datos, y RAID 5 equilibra ambas cosas.

4. LVM (Administración de Volúmenes Lógicos):

- LVM permite agrupar dispositivos de almacenamiento y redimensionar volúmenes de forma flexible. Facilita la gestión de volúmenes y es compatible con RAID.

Repaso Particiones

Muestro:

sudo fdisk -l

(El comando `fdisk` se utiliza para manipular la tabla de particiones de un disco.)

Particiones:

sudo fdisk /dev/vdc

NO TE OLVIDES DE GUARDAR CON W

Formatear:

sudo mkfs.ext4 /dev/vdc1

Montar:

Abre el archivo `/etc/fstab`:

sudo vim /etc/fstab

Agrega la siguiente línea al final del archivo, asegurándose de reemplazar los valores según tu configuración:

```
/dev/vdc1      /Examenes-UTN/alumno_1/parcial_1      ext4      defaults      0
0
```

Guarda el archivo y cierra el editor.

Monta las particiones usando:

sudo mount -a

Mostrar resultado:

lsblk -f \$DISK

para listar la estructura del disco y las particiones de los dispositivos de almacenamiento.

Si tengo que deshacer todo:

Desmontar particiones:

sudo umount /dev/vdc1

Eliminar las entradas del archivo `/etc/fstab`

Abre el archivo **/etc/fstab** y elimina o comenta (agregando un **#** al inicio de cada línea) las líneas que corresponden a tu disco.

```
sudo vim /etc/fstab
```

Tecla Backspace o Delete, o me posiciono en la linea y oprimo **dd**

Guarda los cambios **Ctrl + O** y luego **Enter**. y cierra el editor presionando **Ctrl + X**.

Otra opción es el comando :**wq**.

Borrar particiones:

```
sudo fdisk /dev/vdc
p
d
w
```

Limpiar el disco:

```
sudo wipefs -a /dev/vdb
```

Verificar:

```
lsblk
```

Crear un **usuario** y asignarle un **grupo** en una sola línea:

```
sudo useradd -m -g p1c2_2024_gAlumno p1c2_2024_A1
filtrar el nombre del grupo
grep -i "nombre_grupo" /etc/group | awk -F: '{print $1}'
```

grep: filtra líneas

awk: filtra columnas

Creo los usuarios

```
sudo useradd -m -s /bin/bash -p "$PASSWORD" -G p1c2_2024_gAlumno p1c2_2024_A1
```

Creo los grupos

```
sudo groupadd p1c2_2024_gAlumno
```

Seteo los dueños de los directorios

```
sudo chown p1c2_2024_A1:p1c2_2024_A1 /Examenes-UTN/alumno_1
```

Seteo los permisos

```
sudo chmod -R 750 /Examenes-UTN/alumno_1
```

Validaciones

```
sudo su -c "whoami > /Examenes-UTN/alumno_1/validar.txt" p1c2_2024_A1
```

Crear usuarios y grupos: Crea un nuevo usuario y grupo en una sola línea y asigna el usuario a ese grupo.

```
sudo groupadd nombre_grupo && sudo useradd -m -G nombre_grupo  
nombre_usuario
```

Listar usuarios: Puedes ver todos los usuarios en el sistema consultando el archivo

/etc/passwd:

```
cat /etc/passwd
```

Listar grupos: Para ver todos los grupos, consulta el archivo **/etc/group:**

```
cat /etc/group
```

Eliminar el usuario y el grupo:

```
sudo userdel -r mi_usuario
```

```
sudo groupdel mi_grupo
```

Práctica

Instalamos una iso de linux en una VM.

Clase 2

Creamos una **VM usando Vagrant**.

Para eso, clonamos el siguiente repo.

https://github.com/upszot/UTN-FRA_SO_Vagrant

En el directorio **VagrantFiles/1_equipo/VagrantFiles**, abrimos ese file en la terminal, y con el comando **vagrant up** la instalo.

Una vez instalada, cuando la quiera volver a levantar tengo que volver a tirar el mismo comando.

Si tiro un **vagrant ssh**, me va a conectar a esa VM definida en ese archivo.

Concepto de Vagrant:

Es para levantar una VM sin la necesidad de configurar nada a mano.

Para destruir la VM:

```
vagrant destroy -f (la f es de force)
```

Con este comando borro la VM. Si quiero tener persistencia de datos, no la destruyo, simplemente la apago.

Para apagarla, tiramos un **vagrant halt**.

SSH: Security shell, se usa en linux para conectarse entre diferentes computadoras.

Comandos:

ls: lista los directorios

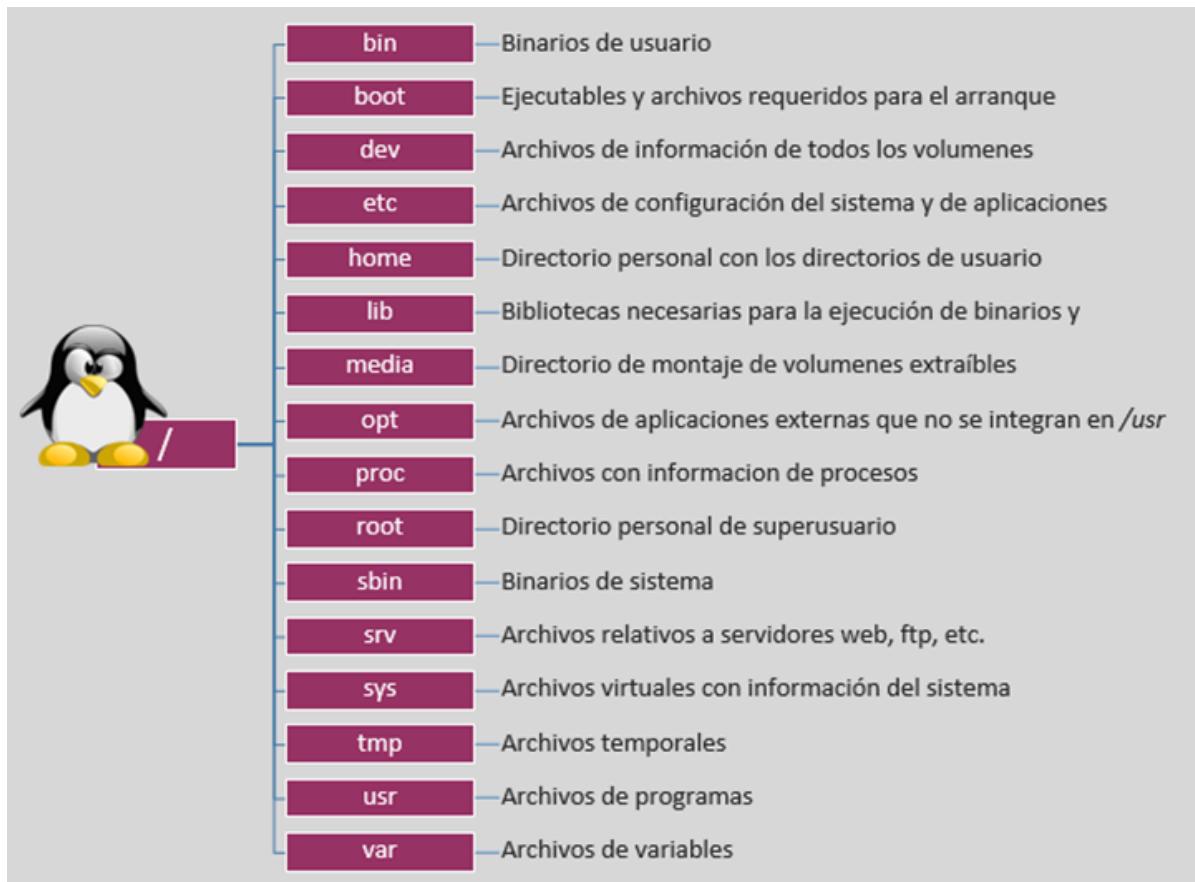
cd: change directory

Para conectarme a la VM en vagrant, luego de haberla creado, levantado y todo, tiro un vagrant ssh.

Una vez conectado, voy a ver lo siguiente en la terminal

```
vagrant@VMPruebas: ~  
vagrant@VMPruebas :~$
```

Estructura de directorios en Linux.



Tenemos que saber qué cosas están en cada directorio.

en /dev tenemos los dispositivos (mouse, discos, webcam, etc)

Es importante destacar que, el home, no es sólo el **/home**, si no, que es **/home/username**

Mostrar núcleos de la computadora:

```
[upszot@halley ~]$ grep -i proce /proc/cpuinfo
processor : 0
processor : 1
processor : 2
processor : 3
processor : 4
processor : 5
processor : 6
processor : 7
```

El less sirve para mostrar datos de un archivo. En linux todo es un archivo.

Si tiro un **less /proc/cpuinfo**, me va a tirar datos del procesador.

```
processor : 0
vendor_id : AuthenticAMD
cpu family : 25
model : 80
model name : AMD Ryzen 5 5600H with Radeon Graphics ←
stepping : 0
cpu MHz : 3293.808
cache size : 512 KB
physical id : 0
siblings : 2
core id : 0
cpu cores : 2
apicid : 0
initial apicid : 0
fpu : yes
fpu_exception : yes
cpuid level : 16
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat p
se36 clflush mmx fxsr sse sse2 ht syscall nx mmxext fxsr_opt rdtscp lm constant_tsc r
ep_good nop1 nonstop_tsc cpuid extd_apicid tsc_known_freq pni pclmulqdq ssse3 cx16 ss
e4_1 sse4_2 x2apic movbe popcnt aes xsave avx rdrand hypervisorlahf_lm cmp_legacy cr
8_legacy abm sse4a misalignsse 3dnowprefetch vmmcall fsgsbase bmi1 avx2 bmi2 invpcid
rdseed clflushopt arat
bugs : fxsave_leak sysret_ss_attrs null_seg spectre_v1 spectre_v2 srso
bogomips : 6587.61
TLB size : 2560 4K pages
clflush size : 64
cache_alignment : 64
address sizes : 48 bits physical, 48 bits virtual
power management:

processor : 1
vendor_id : AuthenticAMD
cpu family : 25
model : 80
model name : AMD Ryzen 5 5600H with Radeon Graphics
stepping : 0
cpu MHz : 3293.808
cache size : 512 KB
physical id : 0
siblings : 2
core id : 1
cpu cores : 2
apicid : 1
initial apicid : 1
```

Clase 3

Vimos comandos de linux:

man: muestra un manual del comando

Para buscar data dentro del man, puedo tocar la barra “/” y buscar texto

Con la letra n voy saltando entre las coincidencias.

Todos estos atajos son los mismos que usa vim.

Revisar las diferencias entre man 1, man 2, etc

Pq en la clase usamos mucho el man 5

useradd: para agregar un usuario

passwd: para actualizar la pw de un user

Si no soy admin del sistema, solo puedo cambiar mi propia password.

Si soy root, puedo cambiar cualquier pw de cualquier usuario.

Para ejecutar el comando como root, tengo que tirar un sudo.

sudo passwd pepe -> con esto le cambio la pw al user pepe

estas claves seteadas se guardan en /etc/passwd

para ver esto, puedo tirar un **man 5 passwd**

Este archivo lo puede leer cualquier persona, por eso no se muestra la pw.

Con ese comando averiguo la password del user llamado “vagrant”

grep vagrant /etc/passwd

Me tira el siguiente output:

vagrant:x:1000:1000:,,,:/home/vagrant:/bin/bash

La pass real la vamos a poder en /etc/shadow.

Esta no la vamos a poder leer

Crear una carpeta que tenga otra carpeta dentro

mkdir -p papa/hijo

el -p es para poder crear un parent. ya que el directorio /papa no existe, y también lo estoy creando

lsb_release -a

muestra informacion de la distribución de linux que estoy utilizando

sudo grep pepe /etc/shadow

suponiendo que pepe es mi username
me devuelve el value de mi password encriptada

si tiro un:

sudo grep \$(whoami)/etc/shadow

me tira lo mismo, siempre y cuando whoami sea pepe.
Es decir, le agrego el valor de lo que devuelve el comando

el retorno es el siguiente:

```
nano:$y$j9T$c.Tia2G.av4WSZdbNNmoL0$M.nON2UQVY6GEJyAVFXzrLJIHxwvIFOrtjmw0xmXgAB:19  
822:0:99999:7:::
```

Tenemos que aprender a armar un script para listar los usuarios, y que tomemos la clave de otro usuario.

A revisar cómo hacer esto en proximas clases

Guardar el retorno de un comando en un archivo:

comando > archivo

Entonces, se me guarda el retorno del comando en el archivo que llamé “archivo”

El mayor “pisa” lo anterior

Si quiero conservar lo anterior, agregando algo nuevo, tiro un **>>**

comando >> archivo

Esto se llama direccionamiento a la derecha

También tengo direccionamiento a la izquierda, con el operador **<<**

Conocer cuantos procesadores tiene mi compu

/proc/cpuinfo

acá adentro está toda la info del cpu

si tiro un

grep proce /proc/cpuinfo

me tira la cantidad de procesadores que tengo

si tiro un **tail**, nos muestra lo último del archivo

por default muestra las ultimas 20 lineas

si tiro un **tail -n1 fileName**, me devuelve la última line del archivo llamado fileName

lo contrario de esto, es un **head**.

muestra las primeras 20 lineas

comando **wc**, cuenta las líneas de un archivo

por ejemplo:

echo "hola mundo" | wc

si retorna **1 2 11**, significa lo siguiente

1 linea

2 palabras

11 caracteres

bc << 5+2

bc no es un “comando” como tal, es un programa, como una calculadora.

wc -l << EOF

el comando wc -l me mostró la cantidad de lineas de lo que escribí luego de este comando

```
cat << EOF
```

el comando cat me mostró lo que escribí luego de meter este comando

```
cat << FIN >> archivo
```

podemos escribir sobre un archivo sin usar un editor de texto, sólo usando redireccionamiento

Crear directorios dentro de otros.

En el parcial vamos a tener que armar una estructura de directorios como esa importante no dejar espacios entre los caracteres

```
mkdir -p ejercicio1/{papa,mama}/hijos_{1..5}
```

```
nano@nahuel-ubuntu:~$ mkdir -p ejercicio1/{papa,mama}/hijos_{1..5}
nano@nahuel-ubuntu:~$ tree ejercicio1/
ejercicio1/
└── mama
    ├── hijos_1
    ├── hijos_2
    ├── hijos_3
    ├── hijos_4
    └── hijos_5
└── papa
    ├── hijos_1
    ├── hijos_2
    ├── hijos_3
    ├── hijos_4
    └── hijos_5

12 directories, 0 files
```

```
mkdir -p ejercicio2/{papa/{hijos_{1..5}},amante},mama/hijos_{1..5}}
```

```
nano@nahuel-ubuntu:~$ mkdir -p ejercicio2/{papa/{hijos_{1..5}},amante},mama/hijos_{1..5}
nano@nahuel-ubuntu:~$ tree ejercicio2
ejercicio2
├── mama
│   ├── hijos_1
│   ├── hijos_2
│   ├── hijos_3
│   ├── hijos_4
│   └── hijos_5
└── papa
    ├── amante
    ├── hijos_1
    ├── hijos_2
    ├── hijos_3
    ├── hijos_4
    └── hijos_5

13 directories, 0 files
```

lo que pongo dentro de llaves, separado por comas, son hermanos.

```
nano@nahuel-ubuntu:~$ grep -i proc -B2 -A1 /proc/cpuinfo
processor      : 0
vendor_id      : AuthenticAMD
--
power management:

processor      : 1
vendor_id      : AuthenticAMD
--
power management:

processor      : 2
vendor_id      : AuthenticAMD
--
power management:

processor      : 3
vendor_id      : AuthenticAMD
```

Muestro los procesadores (o los nucleos de mi procesador)

Para visualizar el archivo, puedo tirar un **less**, o un **more**

Comando para ver lo que está ocurriendo en el kernel. Lo obtiene desde un log

dmesg

Clase 4

Comandos de linux + particionamiento

Un compa presentó pantalla para instalar fedora 39
Instalar esta version para el parcial

lsb_release -a

para saber en qué distribución estoy

Creación de usuarios

Tenemos 2 comandos para hacerlo:

useradd y **adduser**

adduser es interactivo.

Es decir, te va haciendo preguntas y vos vas contestando.

Los resultados son los mismos.

el useradd es más rápido pq le pasa por params toda la info y listo.

Agregamos un usuario llamado prueba01

sudo useradd prueba01

para buscarlo, tiro un **id prueba01**

En debian y derivados no crea la carpeta /home/prueba01 del usuario recien creado

En cambio, en Fedora si funciona.

siendo sudo, si tiro el comando **su prueba01**, me convierto en ese usuario sin necesidad de poner una password.

Si tiro un **su - prueba01**, intenta setearme ese usuario y moverme a la home de ese usuario.

En caso de que no exista, es pq no puse los parámetros para que cree el directorio home.

Con el comando **env** puedo ver las variables de entorno.

SHELL=/bin/sh es el intérprete de comando que tiene asignado el usuario.

USER=prueba01 es el nombre de usuario

PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/usr/games:/usr/local/games:/snap/bin esta me indica dónde voy a ir a buscar todo los comandos

LANG=en_US.UTF-8 me indica el lenguaje

Comandos para agregar un usuario correctamente

params:

- c para tirar un comment asociado al usuario
- d para cambiar la ubicación de la home del usuario a crear
- g para el grupo primario
- G para los grupos secundarios
- m para crear el directorio home del usuario
- p para pasarle el hash de la clave
- s setea el tipo de shell, nosotros vamos a usar /bin/bash

Entonces, para crear un usuario correctamente, lo hacemos así:

El orden de los params no importa, excepto en el caso del comment. Es decir, lo que venga despues del -c, debe ser el string con el comentario

useradd -c "usuario bien creado" -m -s /bin/bash prueba02

Y listo.

Luego, para chequear que esté todo ok, puedo tirar esto: **grep prueba02 /etc/passwd**

Y me retorna lo siguiente:

prueba02:x:1002:1002:usuario bien creado:/home/prueba02:/bin/bash

Revisar que es cada campo, en el retorno de ese grep.

Crear un usuario y meterlo dentro de un grupo

Para esto, primero tengo que crear el grupo.

groupadd es para crear grupos.

Para crear un grupo, tiro lo siguiente:

```
groupadd grupoprueba
```

Ahora, creo al usuario en su grupo

```
useradd -c "usuario bien creado" -m -s /bin/bash -g grupoprueba prueba03
```

```
root@nahuel-ubuntu:/home/nano# groupadd grupoprueba
root@nahuel-ubuntu:/home/nano# useradd -c "usuario bien creado" -m -s /bin/bash -g grupoprueba prueba03
root@nahuel-ubuntu:/home/nano# ls -l /home/
total 12
drwxr-x--- 18 nano      nano      4096 abr 29 19:25 nano
drwxr-x---  2 prueba02   prueba02  4096 abr 29 19:52 prueba02
drwxr-x---  2 prueba03   prueba03  4096 abr 29 19:53 prueba03
```

Eso marcados en rojo son los grupos.

```
root@nahuel-ubuntu:/home/nano# id prueba03
uid=1003(prueba03) gid=1003(grupoprueba) groups=1003(grupoprueba)
```

El usuario 1003 pertenece al grupo 1003, llamado grupoprueba

Ahora vamos a crear un nuevo usuario y meterlo en un grupo secundario

```
useradd -m -s /bin/bash -c "usuario prueba04" -G grupoprueba prueba04
```

Tiro un grep para mostrar la palabra prueba en este archivo:

/etc/group

```
root@nahuel-ubuntu:/home/nano# grep prueba /etc/group
prueba01:x:1001:
prueba02:x:1002:
grupoprueba:x:1003:prueba04 ←
prueba04:x:1004:
```

Agregarle un grupo a un usuario ya existente

```
usermod -G grupoprueba -s /bin/bash -a prueba01
```

```
root@nahuel-ubuntu:/home/nano# usermod -G grupoprueba -s /bin/bash -a prueba01
root@nahuel-ubuntu:/home/nano# grep prueba /etc/group
prueba01:x:1001:
prueba02:x:1002:
grupoprueba:x:1003:prueba04,prueba01 ←
prueba04:x:1004:
```

Y ahora, si tiro ese grep, veo que el grupoprueba (linea 4) tiene 2 usuarios, el prueba04 y el prueba01

Cambiarle la password a un usuario ya existente

passwd a secas sin pasarle params, estoy cambiando la clave con el user que estoy logueado en ese momento.

Si soy user vagrant, y tiro un passwd, me voy a setear “vagrant” de pw. Es decir, mi nombre de usuario.

Si le quiero cambiar la password al usuario prueba02, tiro lo siguiente:

passwd prueba02

```
root@nahuel-ubuntu:/home/nano# passwd prueba02
New password:
BAD PASSWORD: The password contains the user name in some form
Retype new password:
passwd: password updated successfully
```

El usuario root puede cambiarle la password y setear cualquier cosa.

Si lo hago desde otro usuario, y pongo una clave débil, no me deja.

Logearse con otro usuario.

Logearse con otro usuario.

Si estoy en mi user nano, y quiero logearse con el usuario prueba02, tiro lo siguiente:

su - prueba02

Ingreso la password, y listo.

Que no me pida que ingrese la password cada vez que tiro un sudo su.

tirar el comando **visudo**

```
root@nahuel-ubuntu:/home/nano 104x40
GNU nano 6.2                               /etc/sudoers.tmp *

# Completely harmless preservation of a user preference.
Defaults: %sudo env_keep += "GREP_COLOR"

# While you shouldn't normally run git as root, you need to with etckeeper
Defaults: %sudo env_keep += "GIT_AUTHOR_* GIT_COMMITTER_"

# Per-user preferences; root won't have sensible values for them.
Defaults: %sudo env_keep += "EMAIL DEBEMAIL DEBFULLNAME"

# "sudo scp" or "sudo rsync" should be able to use your SSH agent.
Defaults: %sudo env_keep += "SSH_AGENT_PID SSH_AUTH_SOCK"

# Ditto for GPG agent
Defaults: %sudo env_keep += "GPG_AGENT_INFO"

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL

# Members of the admin group may gain root privileges
%admin  ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
nano    ALL=(ALL) NOPASSWD: ALL ←

# See sudoers(5) for more information on "@include" directives:

@include /etc/sudoers.d
```

Y agregamos esa linea

Ver la password hasheada de un usuario

Tiramos un **sudo grep username /etc/shadow**

```
shadow  shadow  shadow
nano@nahuel-ubuntu:~$ sudo grep prueba02 /etc/shadow
prueba02:$y$9T$tLcUD7grbiWCU9KtaLUTF.$1RlZsKvxxAwCR1xRhIbkCzFJ07KggygI/LvMQhrSUPKA:19842:0:99999:7:::
```

lo rojo es la 1era columna

lo verde es la 2da columna. Esta es la password hasheada
al resto de columnas no le voy a dar bola

Necesito obtener el valor de la columna verde con un comando
Lo puedo hacer con este comando:

```
sudo grep prueba02 /etc/shadow | awk -F ":" '{print $2}'
```

awk -F ':' es para que el delimitador, en vez del espacio, sea el signo de dos puntos (:)
Y luego lo printeo

```
nano@nahuel-ubuntu:~$ sudo grep prueba02 /etc/shadow | awk -F ":" '{print $2}'  
$y$j9T$tLcUD7grbiWCU9KtaLUTF.$1RiZsKvxxAwCR1xRhIbkCzFJ07KgygI/LvMQhrSUpKA
```

Y ahora, para crear un usuario llamado prueba05, y setearle esa pass, puedo hacer esto:

```
[root@halley plex]# grep root /etc/shadow  
root:$y$j9T$tLcUD7grbiWCU9KtaLUTF.$1RiZsKvxxAwCR1xRhIbkCzFJ07KgygI/LvMQhrSUpKA:19842:0:99999:7:::  
[root@halley plex]# grep root /etc/shadow | awk -F ":" '{print "useradd -p \"$2\" \"pepe\""}'  
useradd -p "$y$j9T$tLcUD7grbiWCU9KtaLUTF.$1RiZsKvxxAwCR1xRhIbkCzFJ07KgygI/LvMQhrSUpKA" pepe  
[root@halley plex]#
```

Ver todos los usuarios que contienen la palabra prueba:

```
nano@nahuel-ubuntu:~$ sudo grep prueba /etc/shadow  
prueba01!:19842:0:99999:7:::  
prueba02:$y$j9T$tLcUD7grbiWCU9KtaLUTF.$1RiZsKvxxAwCR1xRhIbkCzFJ07KgygI/LvMQhrSUpKA:19842:0:99999:7:::  
prueba03!:19842:0:99999:7:::  
prueba04!:19842:0:99999:7:::  
prueba05:.RiZsKvxxAwCR1xRhIbkCzFJ07KgygI/LvMQhrSUpKA:19842:0:99999:7:::
```

Setearle una password ya existente a un usuario ya existente

```
sudo usermod -p $(bash Scripts/obtener_password.sh userDelQueQuieroLaPw)  
userAlQueLeQuieroActualizarLaPw
```

Permisos en linux

Estas 3 líneas verdes de la imagen representan los permisos

Dueño -> 3 primeros caracteres

Grupo -> 3 segundos caracteres

Otros -> 3 últimos caracteres

```
[upszot@halley tmp]$ ls -l script1.sh  
-rwxr--r--. 1 upszot upszot 7 abr 29 21:30 script1.sh  
[upszot@halley tmp]$ chmod
```

Número	Binario	Lectura (r)	Escritura (w)	Ejecución (x)
0	000	✗	✗	✗
1	001	✗	✗	✓
2	010	✗	✓	✗
3	011	✗	✓	✓
4	100	✓	✗	✗
5	101	✓	✗	✓
6	110	✓	✓	✗
7	111	✓	✓	✓

Cambiar los permisos
chmod 765 script1.sh

permisos iniciales:

rw rw r

cat si

ls no

wx wx r

334

Esto me deja ls pero cat no

14

ese comando significa lo siguiente:

Para el archivo script1.sh,

7 representa que el dueño tendrá permisos para todo

6 representa que el grupo tendrá permisos de lectura y escritura

5 representa que otros sólo tendrán permisos de lectura

Tal como dice la tabla:

Sólo Lectura = 4

Sólo Ejecución = 1

Sólo Escritura = 2

Normalmente vamos a encontrar las siguientes combinaciones de permisos **644, 755**

Posibles preguntas de parcial relacionadas a esto:

Quiero que tal usuario pueda escribir.

Quiero que tal otro usuario no pueda.

Etc

Exámenes:

https://github.com/upszot/UTN-FRA_SO_Examenes/tree/master/20230521

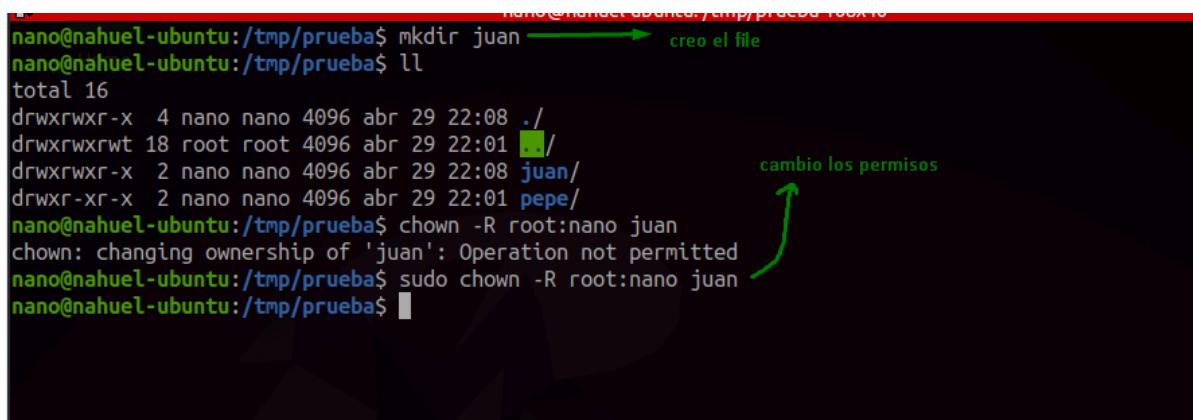
Revisar cómo hacer el formateo de particiones de forma automatizada en el siguiente file:
https://github.com/upszot/UTN-FRA_SO_Examenes/blob/master/20230521/resolucion.md

Cambiar el owner al file

chown (change owner)

Si le quiero cambiar el owner al directorio llamado **juan**, hago lo siguiente

sudo chown -R root:nano juan



```
nano@nahuel:~/tmp/prueba$ mkdir juan → creo el file
nano@nahuel:~/tmp/prueba$ ll
total 16
drwxrwxr-x  4 nano nano 4096 abr 29 22:08 ./
drwxrwxrwt 18 root root 4096 abr 29 22:01 ../
drwxrwxr-x  2 nano nano 4096 abr 29 22:08 juan/ ← cambio los permisos
drwxr-xr-x  2 nano nano 4096 abr 29 22:01 pepe/
nano@nahuel:~/tmp/prueba$ chown -R root:nano juan
chown: changing ownership of 'juan': Operation not permitted
nano@nahuel:~/tmp/prueba$ sudo chown -R root:nano juan
nano@nahuel:~/tmp/prueba$
```

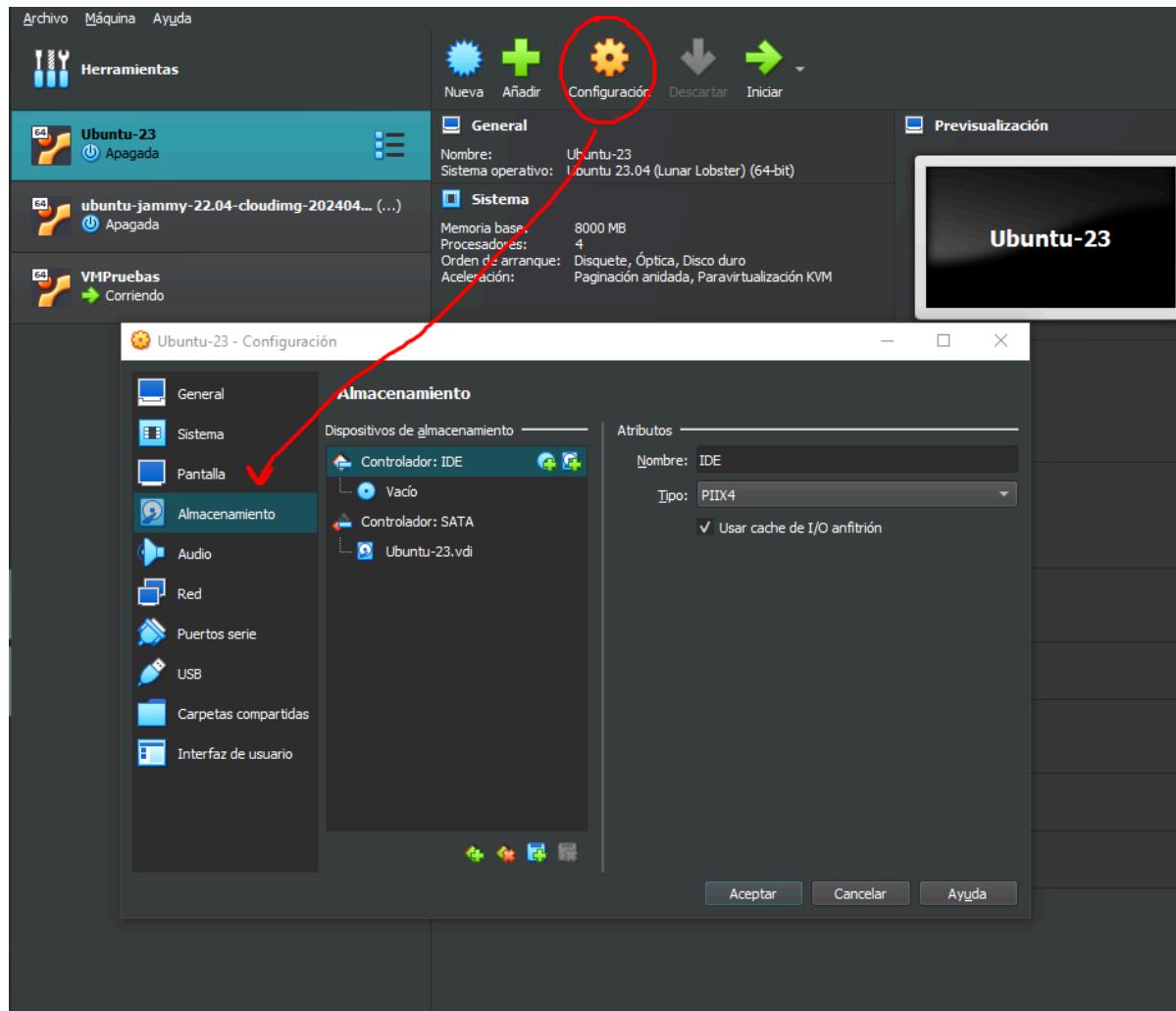
-R es para cambiar el dueño de esa carpeta y de todos los directorios hijos
root:nano significa que, a partir de ahora, root es el dueño del archivo, y nano es el grupo del archivo

Clase 5

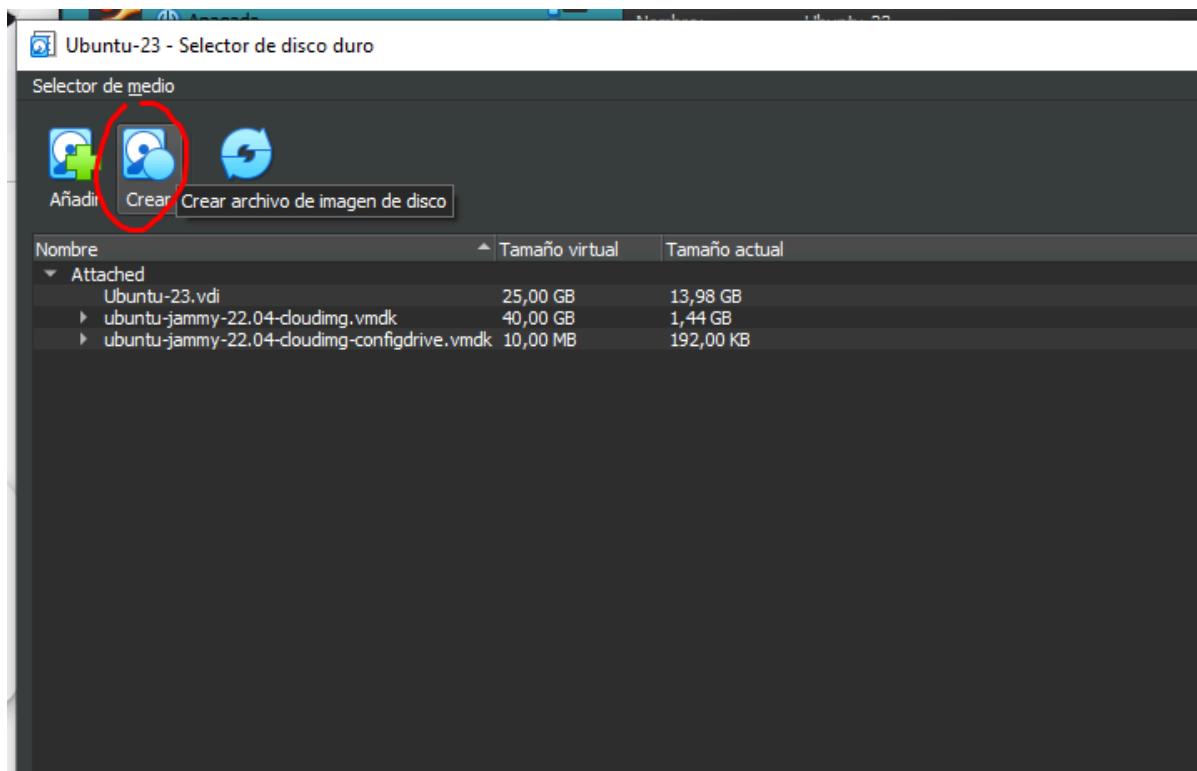
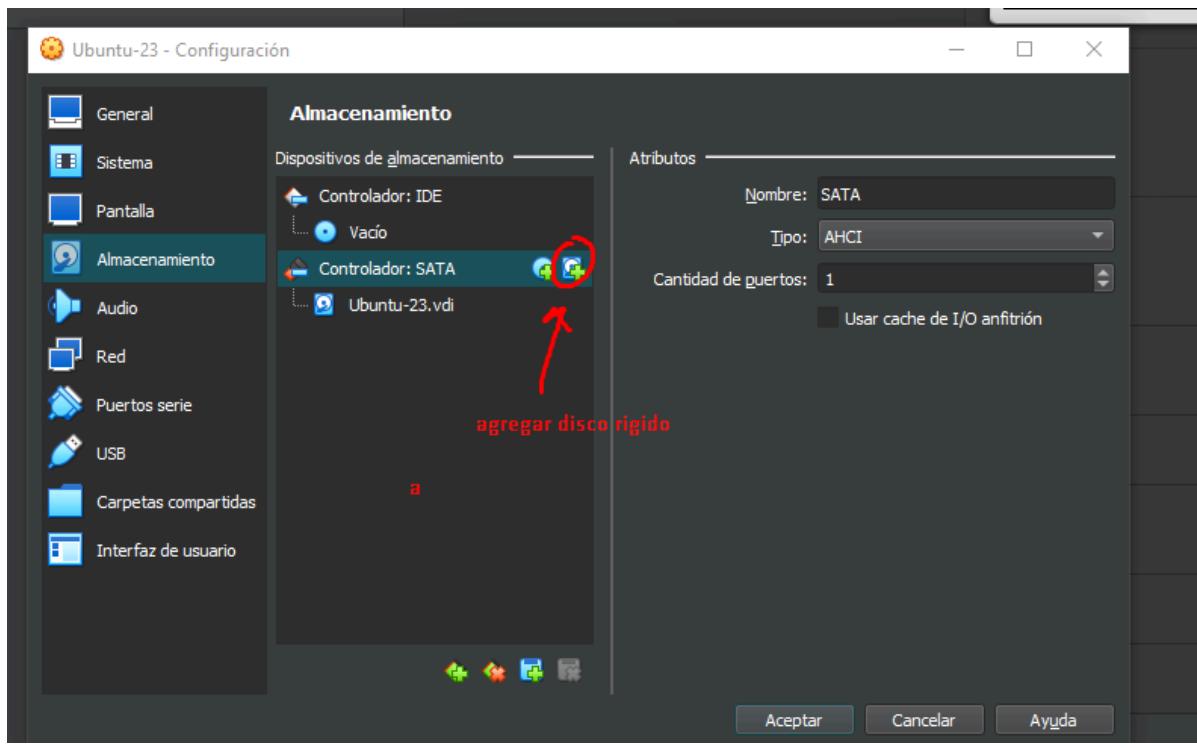
tipos de archivos del sistema.

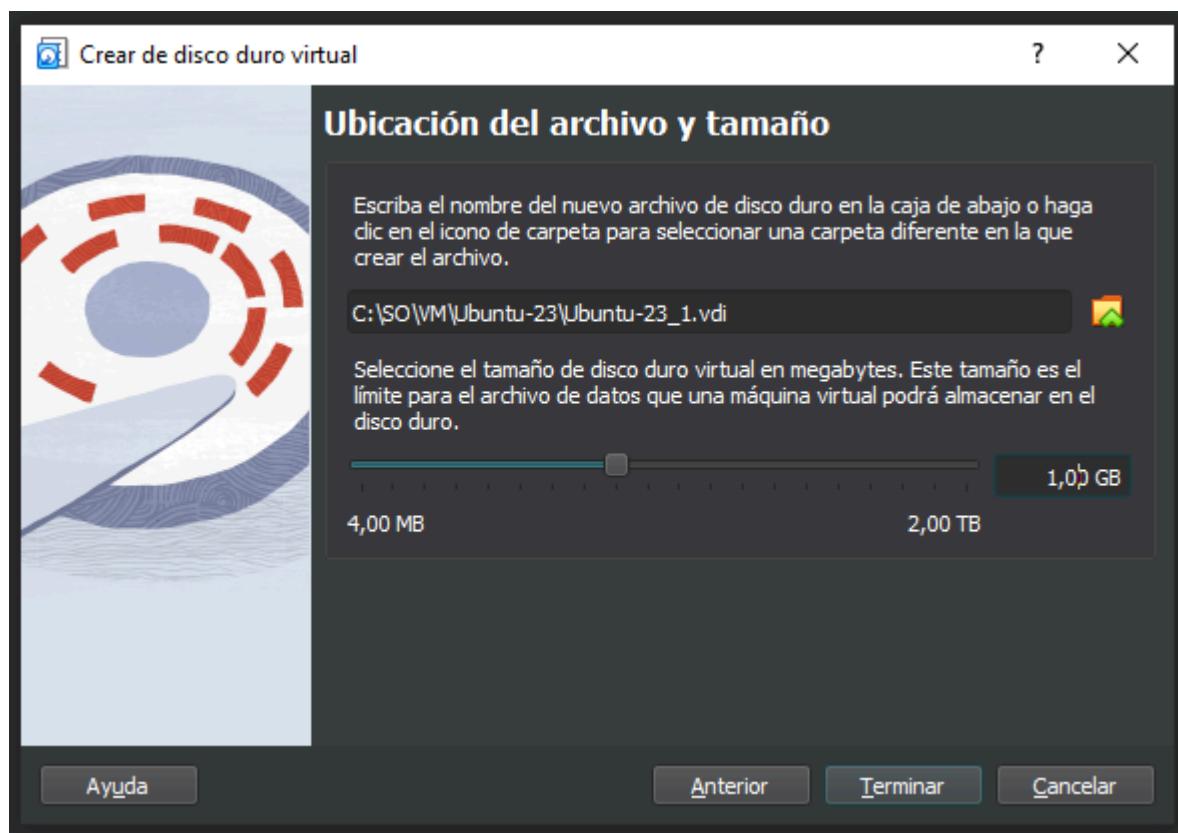
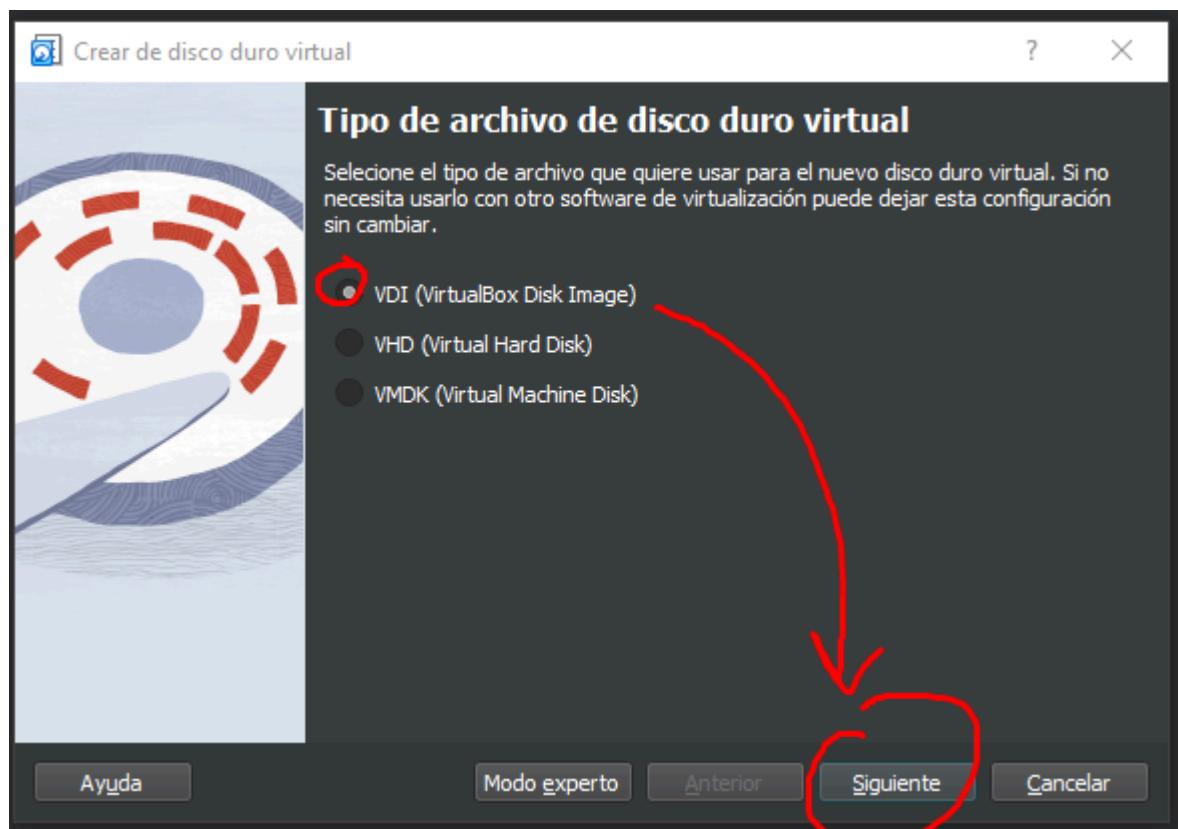
https://github.com/upszot/UTN-FRA_SO_onBording/blob/master/Teoria/FileSistem.md

Agregar discos a una VM

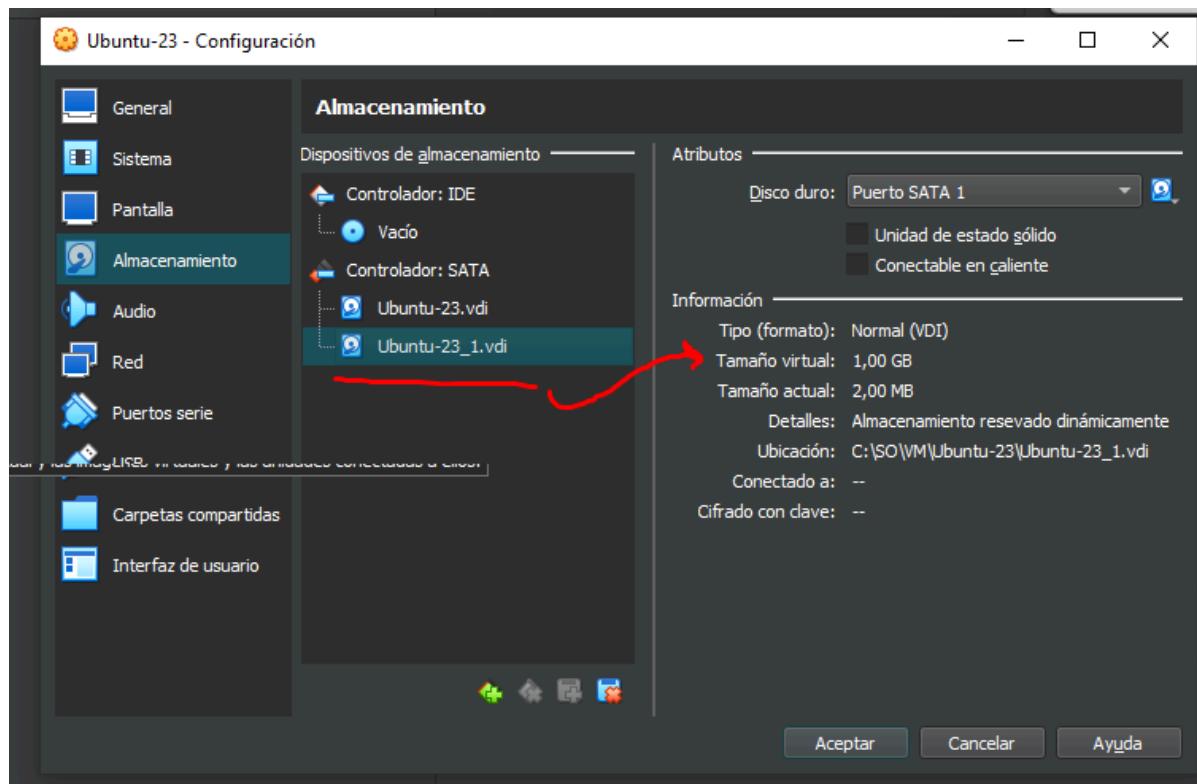


Entro a Controlador:SATA





Y de esa forma agregamos un disco de 1GB.



Y ahí se agregó.

Para el parcial, la idea es arrancar con una VM en limpio, a menos que usemos vagrant. Ya que voy a necesitar usar la VM sin ningún disco inicial.

dmesg

sudo dmesg | grep -i attached

```
nano@nahuel-ubuntu:~$ sudo dmesg | grep -i attached
[    0.711326] sr 1:0:0:0: Attached scsi CD-ROM sr0
[    0.711468] sr 1:0:0:0: Attached scsi generic sg0 type 5
[   1.570758] sd 2:0:0:0: Attached scsi generic sg1 type 0
[   1.577129] sd 2:0:0:0: [sda] Attached SCSI disk
[   1.942427] sd 3:0:0:0: Attached scsi generic sg2 type 0
[   1.944656] sd 3:0:0:0: [sdb] Attached SCSI disk
```

Esto es para conocer los discos que tengo en mi computadora.
En este caso, tengo 2 discos.

sudo ls -l /var/log/dmesg

En este archivo se guardan los logs del kernel.
Este comando va para ubuntu.

Otra forma de ver lo que vimos arriba es la siguiente.

```
nano@nahuel-ubuntu:~$ sudo grep -i attach /var/log/dmesg
[    0.711326] kernel: sr 1:0:0:0: Attached scsi CD-ROM sr0
[    0.711468] kernel: sr 1:0:0:0: Attached scsi generic sg0 type 5
[   1.570758] kernel: sd 2:0:0:0: Attached scsi generic sg1 type 0
[   1.577129] kernel: sd 2:0:0:0: [sda] Attached SCSI disk
[   1.942427] kernel: sd 3:0:0:0: Attached scsi generic sg2 type 0
[   1.944656] kernel: sd 3:0:0:0: [sdb] Attached SCSI disk
```

journalctl -k

journalctl - -dmesg

Estos 2 comandos hacen lo mismo, muestra solamente la información de los mensajes del kernel.

Esto se suele usar cuando no reconoce dispositivos, o hay problemas con el booteo de la máquina

Otros comandos:

journalctl -f

journalctl - -follow

fdisk -l

Este comando me muestra los discos que tengo en la compu

Me muestra el tamaño, la cantidad de sectores del disco, etc.

```
Disk /dev/sda: 25 GiB, 26843545600 bytes, 52428800 sectors
Disk model: VBOX HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: 42080E5D-5E7E-493D-9E2D-ECCA18885BD1

Device      Start    End  Sectors  Size Type
/dev/sda1     2048   4095    2048   1M BIOS boot
/dev/sda2    4096 1054719 1050624 513M EFI System
/dev/sda3 1054720 52426751 51372032 24,5G Linux filesystem
```



sda1 el type es BIOS boot. Acá está el sector de arranque, donde están guardado los scripts de booteo del linux.

Esto me trae el nuevo disco de 1GB que creamos, pq estamos filtrando por sdb

el anterior creado es sda

```
root@nahuel-ubuntu:/home/nano# fdisk /dev/sdb -l
Disk /dev/sdb: 1 GiB, 1073741824 bytes, 2097152 sectors
Disk model: VBOX HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

Formatear y particionar este disco de 1GB que creamos recién:

fdisk /dev/sdb

(/dev/sdb pq el disco se llama así, también podría ser sdc, sde, etc)

```
root@nahuel-ubuntu:/home/nano# fdisk /dev/sdb

Welcome to fdisk (util-linux 2.37.2).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table.
Created a new DOS disklabel with disk identifier 0x788ae00a.

Command (m for help):
```

Tocamos la m para ver la ayuda:

```
Command (m for help): m

Help:

DOS (MBR)
  a  toggle a bootable flag
  b  edit nested BSD disklabel
  c  toggle the dos compatibility flag

Generic
  d  delete a partition
  F  list free unpartitioned space
  l  list known partition types
  n  add a new partition
  p  print the partition table
  t  change a partition type
  v  verify the partition table
  i  print information about a partition

Misc
  m  print this menu
  u  change display/entry units
  x  extra functionality (experts only)

Script
  I  load disk layout from sfdisk script file
  O  dump disk layout to sfdisk script file

Save & Exit
  w  write table to disk and exit
  q  quit without saving changes

Create a new label
  g  create a new empty GPT partition table
  G  create a new empty SGI (IRIX) partition table
  o  create a new empty DOS partition table
  s  create a new empty Sun partition table
```

Las que más vamos a usar son:

w para guardar cambios,

q salir sin grabar

p para imprimir la tabla de partición

l (ele) para listar las particiones

n para crear nuevas particiones

Ahora agregamos partición, tocamos la letra **n**

```
Command (m for help): n
Partition type
  p  primary (0 primary, 0 extended, 4 free)
  e  extended (container for logical partitions)
Select (default p):
```

Tenemos los 2 siguientes tipos de particiones:

- **p (primaria)**

- **e (extendida)**

dentro de **p**, tenemos 0 primarias, 0 extendidas, 4 libres

dentro de **e**, podemos hacer hasta 256 particiones lógicas.

Esto indica que nuestro disco sólo acepta 4 particiones primarias.

Tocamos **p** pq vamos a crear 1 particion primaria

luego **enter**, pq vamos a seleccionar la particion por default (es la 1)

```
Select (default p): p
Partition number (1-4, default 1):
First sector (2048-2097151, default 2048):
```

Luego toco **enter**, y en la parte de Last sector, escribo **+200M**, para crear una partición de 200MB

```
Command (m for help): n
Partition type
  p  primary (0 primary, 0 extended, 4 free)
  e  extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1):
First sector (2048-2097151, default 2048): +200M
Value out of range.
First sector (2048-2097151, default 2048):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-2097151, default 2097151): +200M ←
Created a new partition 1 of type 'Linux' and of size 200 MiB.
```

Ahora, creamos más particiones, con el mismo proceso.

Creamos las 3 primeras de 200Mb

Y la 4ta de 100mb

Es decir, un total de 700MB

```

Command (m for help): p
Disk /dev/sdb: 1 GiB, 1073741824 bytes, 2097152 sectors
Disk model: VBOX HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x788ae00a

Device      Boot   Start     End Sectors  Size Id Type
/dev/sdb1        2048  411647  409600 200M 83 Linux
/dev/sdb2    411648  821247  409600 200M 83 Linux
/dev/sdb3    821248 1230847  409600 200M 83 Linux
/dev/sdb4  1230848 1435647  204800 100M 83 Linux

```

creamos 4 particiones primarias
de un total de 700MB

Ahora tengo que eliminar la última partición, pq es primaria, y no es “extendida”

con la **d** elimino particiones

```

Command (m for help): d ←
Partition number (1-4, default 4): 4 ←

Partition 4 has been deleted.

```

Ahora, voy a crear otra partición con el resto del tamaño que me queda:

```

Command (m for help): n
Partition type
  p  primary (3 primary, 0 extended, 1 free)
  e  extended (container for logical partitions)
Select (default e): e ←

Selected partition 4 ←
First sector (1230848-2097151, default 1230848):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (1230848-2097151, default 2097151):

Created a new partition 4 of type 'Extended' and of size 423 MiB.

```

Y me quedó así:

Device	Boot	Start	End	Sectors	Size	Id	Type
/dev/sdb1		2048	411647	409600	200M	83	Linux
/dev/sdb2		411648	821247	409600	200M	83	Linux
/dev/sdb3		821248	1230847	409600	200M	83	Linux
/dev/sdb4		1230848	2097151	866304	423M	5	Extended

Y luego puedo agregar más particiones, que se crean a partir de la **sdb4**, que es la última partición de tipo extendida

Y ahora puedo crear más particiones a partir de esa extendida

```

Command (m for help): n
All primary partitions are in use.
Adding logical partition 5
First sector (1232896-2097151, default 1232896):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (1232896-2097151, default 2097151): +100M
Created a new partition 5 of type 'Linux' and of size 100 MiB.

Command (m for help): n
All primary partitions are in use.
Adding logical partition 6
First sector (1439744-2097151, default 1439744):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (1439744-2097151, default 2097151):
Created a new partition 6 of type 'Linux' and of size 321 MiB.

Command (m for help): p
Disk /dev/sdb: 1 GiB, 1073741824 bytes, 2097152 sectors
Disk model: VBOX HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x788ae00a

Device      Boot   Start     End Sectors  Size Id Type
/dev/sdb1        2048  411647  409600 200M 83 Linux
/dev/sdb2    411648  821247  409600 200M 83 Linux
/dev/sdb3    821248 1230847  409600 200M 83 Linux
/dev/sdb4   1230848 2097151  866304 423M  5 Extended
/dev/sdb5   1232896 1437695 204800 100M 83 Linux
/dev/sdb6   1439744 2097151 321M 83 Linux

```

Si el **id** de la partición es un **5**, es pq es **extendida**

Importante: Si creo una partición extendida, tengo que usar todo el espacio restante en el disco, ya que si no lo hago, el resto del espacio queda inutilizado.

Cambiar el type a una partición

t para cambiar el type de la partición
 elijo la partición **6**
 y puedo tocar **L** para ver los types de particiones existentes

```

Command (m for help): t
Partition number (1-6, default 6): 6
Hex code or alias (type L to list all): L

```

00 Empty	24 NEC DOS	81 Minix / old Lin	bf Solaris
01 FAT12	27 Hidden NTFS Win	82 Linux swap / So	c1 DRDOS/sec (FAT-
02 XENIX root	39 Plan 9	83 Linux	c4 DRDOS/sec (FAT-
03 XENIX usr	3c PartitionMagic	84 OS/2 hidden or	c6 DRDOS/sec (FAT-
04 FAT16 <32M	40 Venix 80286	85 Linux extended	c7 Syrinx
05 Extended	41 PPC PReP Boot	86 NTFS volume set	da Non-FS data
06 FAT16	42 SFS	87 NTFS volume set	db CP/M / CTOS / .
07 HPFS/NTFS/exFAT	4d QNX4.x	88 Linux plaintext	de Dell Utility
08 AIX	4e QNX4.x 2nd part	8e Linux LVM	df BootIt
09 AIX bootable	4f QNX4.x 3rd part	93 Amoeba	e1 DOS access
0a OS/2 Boot Manag	50 OnTrack DM	94 Amoeba BBT	e3 DOS R/O
0b W95 FAT32	51 OnTrack DM6 Aux	9f BSD/OS	e4 SpeedStor
0c W95 FAT32 (LBA)	52 CP/M	a0 IBM Thinkpad hi	ea Linux extended
0e W95 FAT16 (LBA)	53 OnTrack DM6 Aux	a5 FreeBSD	eb BeOS fs
0f W95 Ext'd (LBA)	54 OnTrackDM6	a6 OpenBSD	ee GPT
10 OPUS	55 EZ-Drive	a7 NeXTSTEP	ef EFI (FAT-12/16/
11 Hidden FAT12	56 Golden Bow	a8 Darwin UFS	f0 Linux/PA-RISC b
12 Compaq diagnost	5c Priam Edisk	a9 NetBSD	f1 SpeedStor
14 Hidden FAT16 <3	61 SpeedStor	ab Darwin boot	f4 SpeedStor
16 Hidden FAT16	63 GNU HURD or Sys	af HFS / HFS+	f2 DOS secondary
17 Hidden HPFS/NTF	64 Novell Netware	b7 BSDI fs	fb VMware VMFS
18 AST SmartSleep	65 Novell Netware	b8 BSDI swap	fc VMware VMKCORE
1b Hidden W95 FAT3	70 DiskSecure Mult	bb Boot Wizard hid	fd Linux raid auto
1c Hidden W95 FAT3	75 PC/IX	bc Acronis FAT32 L	fe LANstep
1e Hidden W95 FAT1	80 Old Minix	be Solaris boot	ff BBT

Aliases:

linux	- 83
swap	- 82
extended	- 05
uefi	- EF
raid	- FD
lvm	- 8E
linuxex	- 85

```

Hex code or alias (type L to list all): 

```

Elijo la **82**, pq quiero hacer una memoria swap.

```

17 Hidden HPFS/NTF  64 Novell Netware  b7 BSDI fs      fb VMware VMFS
18 AST SmartSleep  65 Novell Netware  b8 BSDI swap    fc VMware VMKCORE
1b Hidden W95 FAT3  70 DiskSecure Mult  bb Boot Wizard hid fd Linux raid auto
1c Hidden W95 FAT3  75 PC/IX       bc Acronis FAT32 L fe LANstep
1e Hidden W95 FAT1  80 Old Minix     be Solaris boot ff BBT

Aliases:
  linux      - 83
  swap       - 82
  extended   - 05
  uefi       - EF
  raid        - FD
  lvm         - 8E
  linuxex    - 85

Hex code or alias (type L to list all): 82
Changed type of partition 'Linux' to 'Linux swap / Solaris'. ↗

Command (m for help): p
Disk /dev/sdb: 1 GiB, 1073741824 bytes, 2097152 sectors
Disk model: VBOX HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x788ae00a

Device   Boot   Start   End Sectors  Size Id Type
/dev/sdb1        2048  411647  409600  200M 83 Linux
/dev/sdb2      411648  821247  409600  200M 83 Linux
/dev/sdb3      821248 1230847  409600  200M 83 Linux
/dev/sdb4    1230848 2097151  866304  423M  5 Extended
/dev/sdb5    1232896 1437695  204800  100M 83 Linux
/dev/sdb6    1439744 2097151  657408  321M 82 Linux swap / Solaris ↙

```

Y ahora, para guardar y salir, tengo que usar la **w**

mkfs

significa make file system

El disco rígido es como una torta, que la cortamos en pedazos para hacerle particiones.
Ahora queremos formatear una de esas particiones, la 1er partición

mkfs -t ext4 /dev/sdb1

```
root@nahuel-ubuntu:~# mkfs -t ext4 /dev/sdb1
mke2fs 1.46.5 (30-Dec-2021)
Creating filesystem with 51200 4k blocks and 51200 inodes
Filesystem UUID: ac9ee6d6-d2eb-49fb-824d-1194b24ecb44
Superblock backups stored on blocks:
      32768

Allocating group tables: done
Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done
```

```
root@nahuel-ubuntu:~# mkfs -t ext4 /dev/sdb1
mke2fs 1.46.5 (30-Dec-2021)
Creating filesystem with 51200 4k blocks and 51200 inodes
Filesystem UUID: ac9ee6d6-d2eb-49fb-824d-1194b24ecb44
Superblock backups stored on blocks:
      32768

Allocating group tables: done
Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done
root@nahuel-ubuntu:~#
```

disco que quiero formatear

formato nuevo a setear

Listar los discos que tenemos actualmente:

fdisk /dev/sdb -l

```
root@nahuel-ubuntu:~# fdisk /dev/sdb -l
Disk /dev/sdb: 1 GiB, 1073741824 bytes, 2097152 sectors
Disk model: VBOX HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x788ae00a

Device      Boot   Start     End Sectors  Size Id Type
/dev/sdb1            2048  411647  409600  200M 83 Linux
/dev/sdb2            411648  821247  409600  200M 83 Linux
/dev/sdb3            821248 1230847  409600  200M 83 Linux
/dev/sdb4            1230848 2097151  866304  423M  5 Extended
/dev/sdb5            1232896 1437695  204800  100M 83 Linux
/dev/sdb6            1439744 2097151  657408  321M 82 Linux swap / Solaris
```

Otra forma de hacer el formateo es la siguiente:

mkfs.ext4 /dev/sdb1

Este comando hace exactamente lo mismo que este **mkfs -t ext4 /dev/sdb1**

Ver el tipo de los discos:

Validar que los discos estén formateados:

Ver en qué formato tengo mis particiones:

lsblk -f

NAME	FSTYPE	FSVER	LABEL	UUID	FSAVAIL	FSUSE%	MOUNTPOINTS
loop0	squashfs	4.0			0	100%	/snap/bare/5
loop1	squashfs	4.0			0	100%	/snap/core/16928
loop2	squashfs	4.0			0	100%	/snap/firefox/3836
loop3	squashfs	4.0			0	100%	/snap/core22/1122
loop4	squashfs	4.0			0	100%	/snap/gnome-42-2204/176
loop5	squashfs	4.0			0	100%	/snap/gnome-42-2204/141
loop6	squashfs	4.0			0	100%	/snap/snap-store/959
loop7	squashfs	4.0			0	100%	/snap/gtk-common-themes/1535
loop8	squashfs	4.0			0	100%	/snap/snapd/20671
loop9	squashfs	4.0			0	100%	/snap/snapd/21184
loop10	squashfs	4.0			0	100%	/snap/snapd-desktop-integration/157
loop11	squashfs	4.0			0	100%	/snap/snapd-desktop-integration/83
loop12	squashfs	4.0			0	100%	/snap/tree/18
sda							
└─sda1							
└─sda2	vfat						
└─sda3	ext4	1.0	5BB5-76E5 b103bba2-e33c-447c-87c4-173312633a1f		505,9M 11,8G	1% 45%	/boot/efi /var/snap/firefox/common/host-hunspell
sdb							
└─sdb1	ext4	1.0	ac9ee6d6-d2eb-49fb-824d-1194b24ecb44				
└─sdb2							
└─sdb3							
└─sdb4							
└─sdb5	ext4	1.0	98a39b05-fe80-4364-9704-b47ce4d9fdb1				
└─sdb6							
sr0							

Esa imagen me indica que, dentro de mi disco sdb, que tiene 6 particiones, la sdb1 y la sdb5 con de type ext4

Además de esa data, también puedo ver en dónde está montado mi disco. Eso lo explico abajo.

Montar un disco

El comando es el siguiente:

`mount -t type queQueremosMontar enDondeQueremosMontarlo`

Entonces, terminaría quedando así, pq quiero montar el disco /dev/sdb1 en el directorio /mnt
mount -t ext4 /dev/sdb1 /mnt

Si no retorna nada, significa que está todo bien.

Ahora, puedo tirar un **lsblk -f** para ver el status

```
root@nahuel-ubuntu:~# mount -t ext4 /dev/sdb1 /mnt
root@nahuel-ubuntu:~# lsblk -f
NAME   FSTYPE  FSVER LABEL UUID                                     FSUSE% MOUNTPOINTS
loop0  squashfs 4.0
loop1  squashfs 4.0
loop2  squashfs 4.0
loop3  squashfs 4.0
loop4  squashfs 4.0
loop5  squashfs 4.0
loop6  squashfs 4.0
loop7  squashfs 4.0
loop8  squashfs 4.0
loop9  squashfs 4.0
loop10 squashfs 4.0
loop11 squashfs 4.0
loop12 squashfs 4.0
sda
└─sda1
└─sda2 vfat      FAT32      5BB5-76E5
└─sda3 ext4      1.0        b103bba2-e33c-447c-87c4-173312633a1f  45% /var/snap/firefox/common/host-hunspell
                           /
sdb
└─sdb1 ext4      1.0        ac9ee6d6-d2eb-49fb-824d-1194b24ecb44  0% /mnt
└─sdb2
└─sdb3
└─sdb4
└─sdb5 ext4      1.0        98a39b05-fe80-4364-9704-b47ce4d9fb1
└─sdb6
sr0
```

Y vemos que ese disco está montado, y tiene un 0% de FSUSE, pq no le cargamos nada todavía a ese disco.

Partición de type Swap

En español la llamamos “memoria de intercambio”

¿Qué es?

La memoria swap es un área de almacenamiento en el disco duro de una computadora que se utiliza para almacenar temporalmente datos cuando la memoria RAM física de la computadora está llena.

Tirando un **free -h** puedo ver la memoria que tengo, y cuánta memoria tengo libre

```
nano@nahuel-ubuntu:~$ free -h
              total        used        free      shared  buff/cache   available
Mem:       7,6Gi       865Mi      5,5Gi        33Mi       1,2Gi       6,4Gi
Swap:      2,0Gi          0B      2,0Gi
```

Aca vemos la información para saber de dónde se consume esa memoria swap

```
nano@nahuel-ubuntu:~$ swapon -s
Filename                Type      Size    Used  Priority
/swappfile               file    2097148     0      -2
```

Vamos a montar una particion para usar como swap

Para eso, hay que formatear una partición para que sea de tipo swap.

ASI HAY QUE HACER EL FORMATEO PARA QUE NO FALLE
mkswap discoAFormatear

Quedaría así:

mkswap /dev/sdb6, porque quiero formatear esa partición

```
nano@nahuel-ubuntu:~$ sudo mkswap /dev/sdb6
Setting up swapspace version 1, size = 321 MiB (336588800 bytes)
no label, UUID=0d8616da-3fd4-4223-8f80-f73db8bfedb2
nano@nahuel-ubuntu:~$ sudo fdisk /dev/sdb -l
Disk /dev/sdb: 1 GiB, 1073741824 bytes, 2097152 sectors
Disk model: VBOX HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x788ae00a

Device      Boot   Start     End  Sectors  Size Id Type
/dev/sdb1            2048  411647  409600  200M 83 Linux
/dev/sdb2            411648  821247  409600  200M 83 Linux
/dev/sdb3            821248 1230847  409600  200M 83 Linux
/dev/sdb4            1230848 2097151  866304  423M  5 Extended
/dev/sdb5            1232896 1437695  204800  100M 83 Linux
/dev/sdb6            1439744 2097151  657408  321M 82 Linux swap / Solaris ↙
```

Ver los discos, y conocer en donde están montados:

```
nano@nahuel-ubuntu:~$ sudo lsblk -l /dev/sdb
NAME MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sdb    8:16   0   1G  0 disk
sdb1   8:17   0 200M  0 part /home/nano/peliculas
sdb2   8:18   0 200M  0 part
sdb3   8:19   0 200M  0 part
sdb4   8:20   0    1K  0 part
sdb5   8:21   0 100M  0 part
sdb6   8:22   0 321M  0 part
```

Ahora, para montar esa partición como memoria swap, tiro lo siguiente:

swapon /dev/sdb6

```
nano@nahuel-ubuntu:~$ sudo swapon /dev/sdb6
nano@nahuel-ubuntu:~$ sudo swapon -s
Filename                Type      Size        Used       Priority
/swapfile               file     2097148      0          -2
/dev/sdb6                partition 328700      0          -3
```

Y listo, ahí vemos que ahora /dev/sdb6 está montada como memoria swap.

Sacar la partición como memoria swap:

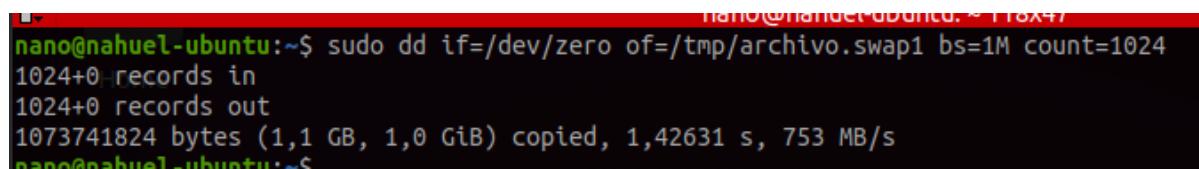
sudo swapoff /dev/sdb6

Hacer una memoria swap en archivos

if = input file

of = output file (nos escriba un archivo con ceros en ese lugar)

sudo dd if=/dev/zero of=/tmp/archivo.swap1 bs=1M count=1024



```
nano@nahuel-ubuntu:~$ sudo dd if=/dev/zero of=/tmp/archivo.swap1 bs=1M count=1024
1024+0 records in
1024+0 records out
1073741824 bytes (1,1 GB, 1,0 GiB) copied, 1,42631 s, 753 MB/s
nano@nahuel-ubuntu:~$
```

Velocidad de escritura 753MB

Si el if es un pendrive, o un CD, y el of es un cd.iso, puedo generar una imagen iso del cd. Esto lo puedo usar para copiar todo el contenido de un lugar en otro.

Formatear este archivo como si fuese una memoria swap

sudo mkswap /tmp/archivo.swap1

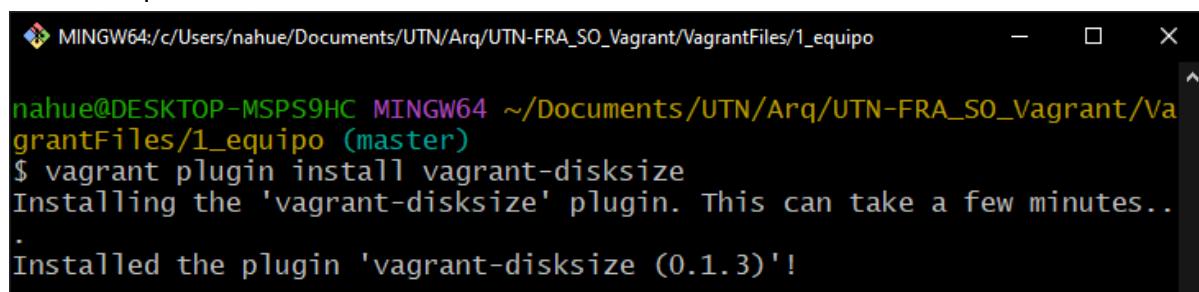
y ahora lo puedo setear como memoria swap, tirando un
swapon /tmp/archivo.swap1

Crear un disco nuevo en la VM con vagrant

Primero, tenemos que instalar el siguiente plugin:

vagrant plugin install vagrant-disksize

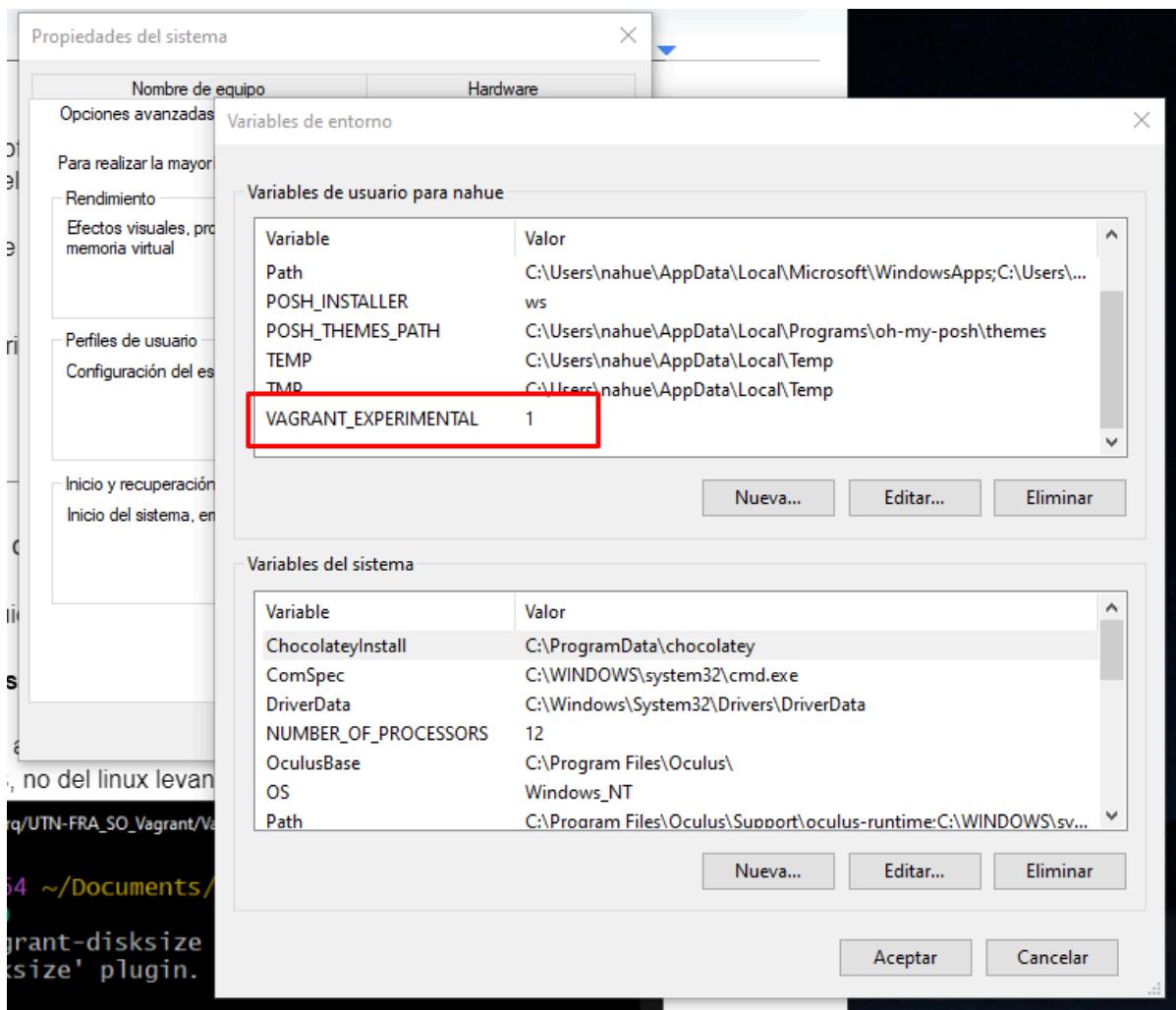
Esto lo tengo que tirar con le vagrant apagado, por fuera del vagrant. Es decir, es un comando que se tira en la terminal de windows, no del linux levantado.



```
MINGW64:/c/Users/nahue/Documents/UTN/Arq/UTN-FRA_SO_Vagrant/VagrantFiles/1_equipo
nahue@DESKTOP-MSPS9HC MINGW64 ~/Documents/UTN/Arq/UTN-FRA_SO_Vagrant/VagrantFiles/1_equipo (master)
$ vagrant plugin install vagrant-disksize
Installing the 'vagrant-disksize' plugin. This can take a few minutes..
.
Installed the plugin 'vagrant-disksize (0.1.3)'!
```

Ahora, hay que setear una variable de entorno en windows
podemos abrir la terminal de windows y tirar
setx VAGRANT_EXPERIMENTAL 1

Y vemos que ahí se creó



En el repo del profe está este directorio /1_equipo_con_mas_discos
El vagrantFile tiene que tener esa linea, que indica que tenemos más discos

Si nosotros queremos agregar más discos en un vagrant, tenemos que editar ese archivo

Por ahora vamos a trabajar con el directorio /1_equipo_con_mas_discos, para no editar ningún archivo.

```
MINGW64:/c/Users/nahue/Documents/UTN/Arq/UTN-FRA_SO_Vagrant/VagrantFiles/1_equipo_con_mas... — □ ×
nahue@DESKTOP-MSPS9HC MINGW64 ~/Documents/UTN/Arq/UTN-FRA_SO_Vagrant/VagrantFiles/1_equipo (master)
$ cd ..

nahue@DESKTOP-MSPS9HC MINGW64 ~/Documents/UTN/Arq/UTN-FRA_SO_Vagrant/VagrantFiles (master)
$ ls
1_equipo/           1_equipo_con_GUI/      2_equipos/
1_equipo_Multiples_GUI/ 1_equipo_con_mas_discos/

nahue@DESKTOP-MSPS9HC MINGW64 ~/Documents/UTN/Arq/UTN-FRA_SO_Vagrant/VagrantFiles/1_equipo_con_mas_discos (master)
$ cd 1_equipo_con_mas_discos/

nahue@DESKTOP-MSPS9HC MINGW64 ~/Documents/UTN/Arq/UTN-FRA_SO_Vagrant/VagrantFiles/1_equipo_con_mas_discos (master)
$ cat Vagrantfile
# -*- mode: ruby -*-
# vi: set ft=ruby :

Vagrant.configure("2") do |config|
  config.ssh.insert_key = false

  #config.vm.box = "generic/rhel7"
  #config.vm.box = "generic/rhel8"
  #config.vm.box = "generic/rhel9"
  config.vm.box = "ubuntu/jammy64"

  config.vm.synced_folder '.', '/home/vagrant/foo'
  config.vm.hostname = "VMPruebas"
  config.vm.define :VMPruebas do |t|
    end

  config.vm.network "private_network", :name => '', ip: "192.168.56.3"

  # Agrego un nuevo disco (Ver Readme.md)
  config.vm.disk :disk, size: "5GB", name: "extra_storage"
  config.vm.disk :disk, size: "2GB", name: "extra_storage2" ←

  config.vm.provider "virtualbox" do |vb|
    vb.memory = "2048"
    vb.name = "VMPruebas"
    vb.cpus = 2
    vb.linked_clone = true
  end

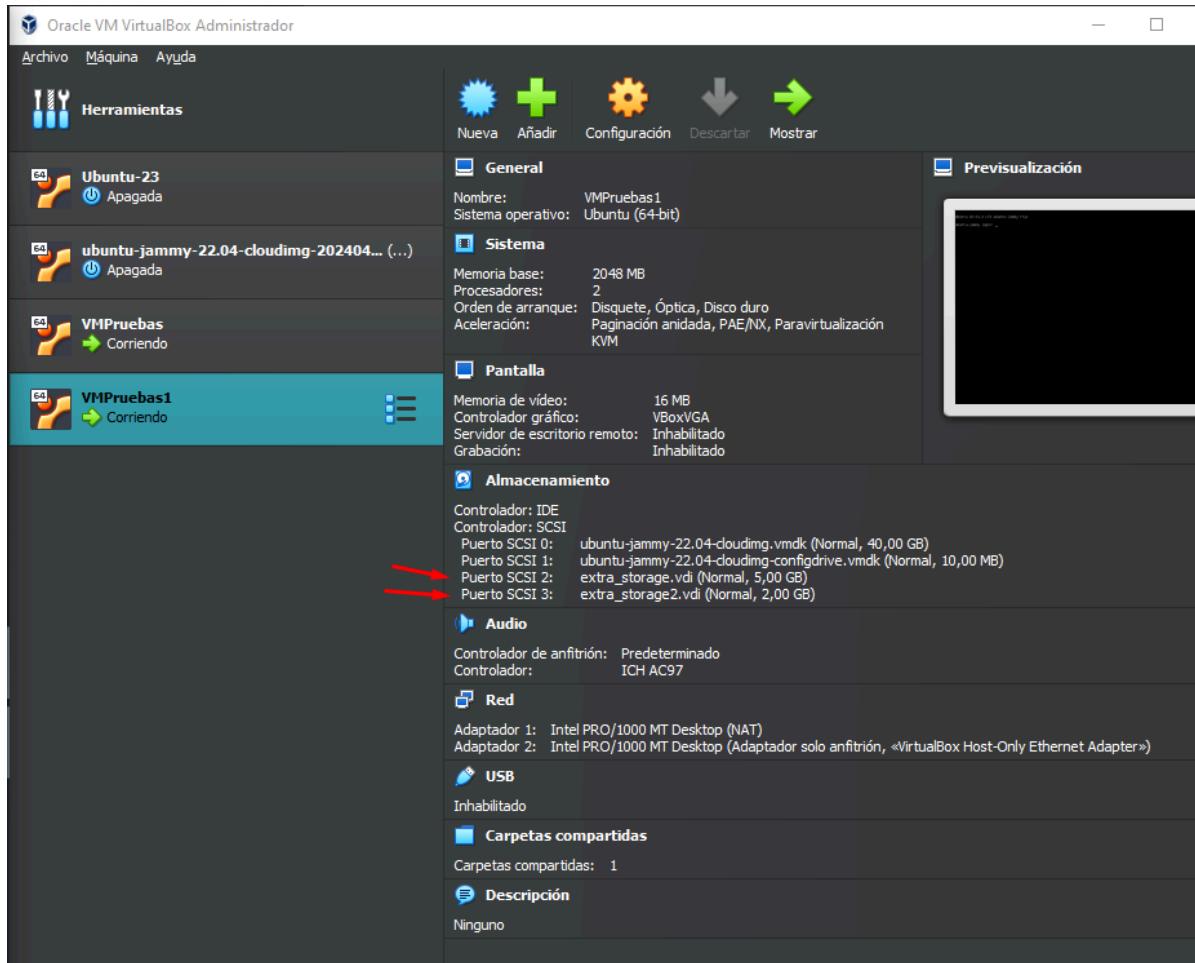
  # Puedo Ejecutar un script que esta en un archivo
  config.vm.provision "shell", path: "script_Enable_ssh_password.sh"

  # Agrega la key Privada de ssh en .vagrant/machines/default/virtualbox/private_key
  config.ssh.insert_key = true
end

nahue@DESKTOP-MSPS9HC MINGW64 ~/Documents/UTN/Arq/UTN-FRA_SO_Vagrant/VagrantFiles/1_equipo_con_mas_discos (master)
$
```

Y ahora podemos tirar un vagrant up para levantar la VM

Ahora vemos que la VM tiene 2 discos más



Resumen:

instalamos los plugins

ejecutamos esa linea para habilitar vagrant experimental en las envvar de windows

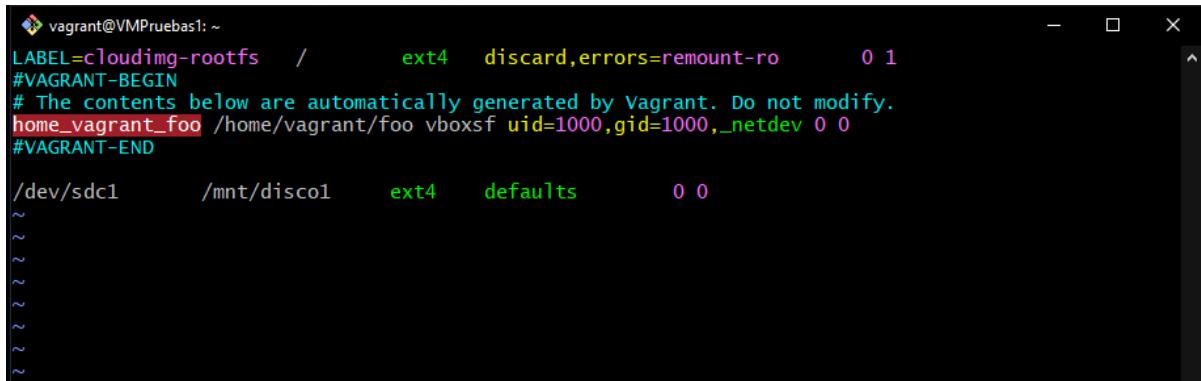
Persistir el disco montado

(ver grabación a partir del horario 22:20 aprox)

Tenemos que escribir en el archivo **/etc/fstab**

Tenemos que insertar lo siguiente:
(los espacios son tabs)

```
/dev/sdc1      /mnt/disco1    ext4    defaults          0 0
```



```
vagrant@VMPruebas1: ~
LABEL=cloudimg-rootfs  /      ext4  discard,errors=remount-ro  0 1
#VAGRANT-BEGIN
# The contents below are automatically generated by Vagrant. Do not modify.
home_vagrant_foo /home/vagrant/foo vboxsf uid=1000,gid=1000,_netdev 0 0
#VAGRANT-END
/dev/sdc1      /mnt/disco1    ext4  defaults        0 0
~
```

Y ahora persistimos la data de que la partición **/dev/sdc1** esté montado en **/mnt/disco1**, en formato **ext4**

Una vez que escribimos ese file, tenemos que tirar un **mount -a**.

En caso de que falle ese comando, no reiniciar la VM, pq se rompe todo.

Si falla, borramos esta linea y revisamos qué falló, lo arreglamos, y vamos a poder reiniciar la VM cuando el **mount -a** tire un retorno correcto

Importante:

Tenemos que tirar un **mount -a** para que lo monte, de lo contrario, va a tratar de montar algo en un lugar que no existe, y no voy a poder reiniciar la máquina virtual con vagrant.

Teoria: https://github.com/upszot/UTN-FRA_SO_onBording/tree/master/Teoria

Clase 6

Comando **diff**

Para obtener las diferencias entre diferentes archivos.

diff < archivo1.txt archivo2.txt

Eso me va a devolver qué diferencias existen entre el **archivo1** y el **archivo2**.

```
nano@nahuel-ubuntu:~$ diff <(echo "hola") <(echo "hola2")
1c1
< hola
---
> hola2
```

En esa imagen, en vez de tirarle archivos para que compare, le tiré 2 echo que devuelven textos diferentes.

Teoria que se toma en el parcial:

En el parcial se va a tomar la siguiente teoría:

https://github.com/upszot/UTN-FRA_SO_onBording/tree/master/Teoria

- 1 - Saber la historia
 - 2 - Saber diferencias entre software libre y privado, y las legislaciones del software libre.
Que sea libre no significa que sea gratis. Un ejemplo de esto es RedHat. Podemos usar hasta 16 servidores sin que nos cobren la licencia.
 - 3 - Saber que es la virtualización de tipo1 y tipo2. hipervisor de tipo1 y tipo2.
-

Simulacro de parcial.

Importante, antes de empezar el parcial, tener una VM con vagrant instalado, y hacer un **sudo apt update**
y luego un **sudo apt install tree**

Formatear muchas particiones juntas

```
sudo fdisk -l /dev/sdc | grep dev | grep -viE 'disk|Ext' | awk '{print "sudo mkfs.ext4 \"\$1\""}' | /bin/bash
```

Eso que pinté es el disco del que quiero formatear sus particiones.

El resto, son comandos para extraer la cantidad de particiones existentes, sacando la partición extendida, ya que esa no se formatea

Y le indico que lo voy a formatear en formato ext4.

Otra forma de hacerlo

```
sudo fdisk -l /dev/sdc | grep dev | grep -viE 'disk|Ext' | awk '{print $1}' | xargs -I DISCO  
echo "sudo mkfs.ext4 DISCO"
```

Si tiro ese comando, voy a ir viendo lo que estoy escribiendo.

Ahora, para efectivamente ejecutarlo, tengo que borrarle ese **echo** y las comillas que engloban el comando a tirar.

Quedaría así:

```
sudo fdisk -l /dev/sdc | grep dev | grep -viE 'disk|Ext' | awk '{print $1}' | xargs -I DISCO  
sudo mkfs.ext4 DISCO
```

Revisar cómo hacer un for para montar todos los discos en cada directorio.

Quedé terminando el video part3

Revisar cómo persistir los discos para que queden montados aunque apague la máquina.

Ver archivos ocultos

ls -la

Crear un script

Ahí creé un script, que es un archivo con una extensión .sh

Y arriba de todo le puse ese comentario para que, cuando se ejecute, se ejecute con bash

El contenido del script es el siguiente:

```
#!/bin/bash  
  
pwd  
ls -l  
whoami  
mkdir -p pepe/juan  
echo "hola" > pepe/juan/archivo
```

Examen:

Ver cómo traer información del disco

Particionar

Formatear

Montar

Crear struct de directorios

Crear x usuarios y grupos con diferentes permisos

- Informacion que hay que saber obtener:

- Cuanta CPU tiene
- A qué frecuencia trabaja el microprocesador

Saber escribir un comando, y guardar ese comando en un archivo

history | grep 39 | grep -v history

```
45 ls -l
46 ls -la
47 ll
48 ls -la
49 vim .bashrc
50 cat 1_script.sh
51 ./1_script.sh
52 vim 1_script.sh
53 cat 1_script.sh
54 ./1_script.sh
55 vim 1_script.sh
56 ./1_script.sh param1 param2 param3 param4
57 ll
58 ll param1/param2/
59 cat 1_script.sh
60 cat pepe/juan/archivo_param4.log
61 history
vagrant@VMPruebas:~$ history |grep 39
39 alias saludo='echo "Hola $(whoami)"'
62 history |grep 39
vagrant@VMPruebas:~$ history |grep 39 |grep -v history
39 alias saludo='echo "Hola $(whoami)"'
vagrant@VMPruebas:~$ history |grep -E '39|40' |grep -v history
39 alias saludo='echo "Hola $(whoami)"'
40 alias
vagrant@VMPruebas:~$ history |grep -E '39|40' |grep -v history > punto_3.txt
```

obtengo la línea 39 del history

obtengo la 39 y 40 del history

Así puedo obtener un comando que tiré, filtrando con un grep.

Y con el > puedo far esa línea a un archivo, como se ve en la línea de abajo de todo, que estoy guardando esas líneas en un archivo llamado punto_3.txt

Obtener varias líneas del history:

history | sed -n '19,30p'

Eso me obtiene desde la línea 19 hasta la 30 inclusive

Saber en que directorio estan los archivos de log
los archivos de configuracion

Revisar como crear un usuario y asignarle una clave

Obtener la memoria ram total

free -h | grep Mem | awk -F " " '{print \$2}'

Obtener el peso de una carpeta:

sudo du -sh /etc

En ese caso estoy obteniendo el peso de la carpeta /etc

Obtener el modelo del procesador

less /proc/cpuinfo | grep 'model name' | head -n 1

Obtener el tamaño de la home del usuario

sudo du -sh /home/\$(whoami)

Que quede sólo con el valor:

echo "Tamaño de la home del usuario: \$(sudo du -sh /home/\$(whoami) | awk '{print \$1}')"

sudo su -c “comando a tirar” username

Con ese comando puedo tirar un comando como si fuese el usuario username, sin necesidad de hacer todo el login completo con ese usuario.

Esto sirve para probar el tema de qué permisos tiene cada usuario.

Ver si existe:

```
sudo su -c "ls -l /home/nano/Programa1/Produccion" Tester
```

Ver contenido de un archivo:

```
sudo su -c "cat /home/nano/Programa1/Produccion/file.txt" Tester
```

```
#Asignar permisos de escritura al grupo en la carpeta otros y todo su contenido
#sudo chmod -R g+w /PuntoB/otro/
sudo chmod -R 774 /PuntoB/otro
sudo ls -l /PuntoB/otro

#Realizar las modificaciones necesarias para que el usuario "parc1_user3"
#pueda conocer la existencia de un archivo en el directorio Grupo2, pero no
#pueda ver su contenido.
sudo su -c "ls -l /PuntoB/Grupo2/" parc1_user3
sudo su -c "cat /PuntoB/Grupo2/existe.txt" parc1_user3
sudo chmod 774 /PuntoB/Grupo2/

#Validar
sudo su -c "whoami >> /PuntoB/otro/validar.txt" parc1_user1
sudo su -c "whoami >> /PuntoB/otro/validar.txt" parc1_user2
sudo cat /PuntoB/otro/validar.txt

sudo su -c "whoami > /PuntoB/Grupo2/existe.txt" parc1_user2
[REDACTED]
sudo su -c "ls -l /PuntoB/Grupo2/" parc1_user3
sudo su -c "cat /PuntoB/Grupo2/existe.txt" parc1_user3
```

Crear estructura de archivos (no de directorios)

```
for i in {4..9}; do
    touch Punto_${i}.txt
done
```

ver password hasheada

```
sudo grep $USER /etc/shadow
```

Conocer la cantidad de procesadores que tengo

```
nproc
```

Crear directorio con la fecha de hoy

```
mkdir -p /home/$(whoami)/RTA_examen_$(date +%Y-%m-%d)
```

dps busco como quedo el path y

```
touch /home/$(whoami)/RTA_examen_2024-05-20/puntoB.txt
```

Agregar varios usuarios de una a un mismo grupo

```
sudo groupadd -U user1,user2 groupName
```

Hacer que `history` guarde automáticamente tras cada comando:

```
export PROMPT_COMMAND="history -a; $PROMPT_COMMAND"
```

Esto hará que `bash` escriba los comandos nuevos en el archivo `.bash_history` automáticamente después de ejecutar cada comando. De este modo, si se trabaja la VM, no perderás el historial hasta ese momento.

Después de agregar esta línea, puedes aplicar el cambio inmediatamente con:

```
source ~/.bashrc
```

Obtener una linea del history sin el numero de linea

```
history | sed -n '245p' | awk '{$1=""; print $0}' | sed 's/^ //'
```

Obtener varias líneas del history sin los números de linea

```
history | sed -n '240,245p' | awk '{$1=""; print $0}' | sed 's/^ //'
```

token:

```
ghp_yPv2DGLkQLI7psOGhIN4k6j593yTRe1PeouD
```

Clase 7 (A partir de acá son temas del 2do parcial)

<https://www.youtube.com/watch?v=mXV6Wabl9Ws&list=PLjf9eZZkd6L-4eUkElISd7uzdZ3ifkzmQ&index=10>

La clase empezó a la hora 20:10 (minuto 1:34:00 del video de arriba)

Repositorios en linux

Se explica cómo “descargar” algo mediante la terminal, con apt.

En este archivo se encuentran los repositorios: **/etc/apt/sources.list.d**

Instalamos docker

Se hace a la hora 21:10 del video.

sudo apt install docker

Con este comando podremos eliminar versiones viejas de docker

for pkg in docker.io docker-doc docker-compose docker-compose-v2 podman-docker containerd runc; do sudo apt-get remove \$pkg; done

Ese comando lo encontramos aca:

<https://docs.docker.com/engine/install/ubuntu/>

Para instalar docker, tiramos todo eso

Install using the apt repository

Before you install Docker Engine for the first time on a new host machine, you need to set up the Docker repository. Afterward, you can install and update Docker from the repository.

1. Set up Docker's apt repository.

```
# Add Docker's official GPG key:  
sudo apt-get update  
sudo apt-get install ca-certificates curl  
sudo install -m 0755 -d /etc/apt/keyrings  
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/  
sudo chmod a+r /etc/apt/keyrings/docker.asc  
  
# Add the repository to Apt sources:  
echo \  
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/d  
$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \  
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null  
sudo apt-get update
```

Give feedback

Finalmente, se terminó instalando docker 26.1.3

```
sudo docker version  
Client: Docker Engine - Community  
Version: 26.1.3
```

Una vez instalado docker correctamente, ya no tendremos que recordar esto, ni hacerlo otra vez.

LVM (Logical Volume Manager)

Hora 21:30

Es para administrar las particiones y su tamaño sin necesidad de reiniciar el sistema, o desmontar lo que esté montado.

https://github.com/upszot/UTN-FRA_SO_onBording/blob/master/LVM/lvm.md

PV: Physical Volume. Un disco físico que se utiliza en LVM.

VG: Volume Group. Grupo de volúmenes que actúan como única unidad de almacenamiento

LV: Logical Volume. Volumen lógico que se crea dentro de un VG.



El siguiente comando me sirve para “limpiar” desde cero el disco.

Obviamente que si se lo tiro a un disco que ya tenía particiones, esas particiones se me van a borrar.

wipefs -a /dev/sdb

Lo ideal es tirar ese comando ni bien agrego los discos, cosa de empezar el parcial con los dicos limpios.

Práctica LVM

Para hacer el proceso, podemos usar este machete como guía:

https://github.com/upszot/UTN-FRA_SO_onBording/blob/master/LVM/Machete_LVM.md

Si no tengo LVM instalado, lo puedo instalar con este comando:

sudo apt install lvm2

Primero que nada, vamos a iniciar nuestra máquina con 2 discos. Uno de 5GB y otro de 2GB.

Al primer disco (el de 5GB), le creamos una partición primaria de 2GB

Ahora, vamos a cambiarle el type de partición.

La formateamos con el type **Linux LVM (8e)**

(para formatear con este formato, lo hacemos desde el fdisk)

Y luego, le creamos otra partición del total restante del disco, y le pusimos que sea de type **Linux swap / Solaris (82)**

```
nano@nahuel-ubuntu:~$ sudo fdisk -l /dev/sdb
Disk /dev/sdb: 5 GiB, 5368709120 bytes, 10485760 sectors
Disk model: VBOX HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xe9b64f30

Device      Boot   Start     End Sectors Size Id Type
/dev/sdb1        2048 4196351 4194304   2G 8e Linux LVM ←
/dev/sdb2        4196352 10485759 6289408   3G 82 Linux swap / Solaris ←
```

Así debería quedar el disco de 5GB.

Ahora vamos a crear el PV (Physical Volume)

sudo pvcreate /dev/sdb1

```
nano@nahuel-ubuntu:~$ sudo pvcreate /dev/sdb1
  Physical volume "/dev/sdb1" successfully created.
```

Le pasamos la partición de type Linux LVM que creamos (NO EL DISCO ENTERO, sólo la partición)

Mostrar datos de LVM en forma de resumen:

sudo pvs (muestra los PV)

```
nano@nahuel-ubuntu:~$ sudo pvs
  PV          VG Fmt Attr PSize PFree
  /dev/sdb1    lvm2 --- 2,00g 2,00g
```

sudo vgs (muestra los VG)

sudo lvs (muestra los LV)

Crear el VG

El VG es un contenedor de PVs.

sudo vgcreate vg_datos /dev/sdb1

`sudo vgcreate nombreDelVg nombreDelPv`

El PV puede ser uno solo o varios.

```
nano@nahuel-ubuntu:~$ sudo vgcreate vg_datos /dev/sdb1
```

Nombre del VG

PV que va a estar dentro del VG llamado vg_datos

Lo creamos:

```
nano@nahuel-ubuntu:~$ sudo vgcreate vg_datos /dev/sdb1
Volume group "vg_datos" successfully created
```

```
nano@nahuel-ubuntu:~$ sudo pvs
PV          VG      Fmt Attr PSize  PFree
/dev/sdb1   vg_datos lvm2 a--  <2,00g <2,00g
nano@nahuel-ubuntu:~$ sudo vgs
VG          #PV #LV #SN Attr   VSize  VFree
vg_datos    1   0   0 wz--n- <2,00g <2,00g
```

De la imagen de arriba, vemos que ahora el pv /dev/sdb1 está dentro del VG vg_datos

Crear el LV

```
sudo lvcreate -l 50%FREE nombreDelVg -n nombreDelLv
```

```
nano@nahuel-ubuntu:~$ sudo lvcreate -l +50%FREE vg_datos -n lv_peliculas
Logical volume "lv_peliculas" created.
nano@nahuel-ubuntu:~$ sudo lvcreate -l +50%FREE vg_datos -n lv_series
Logical volume "lv_series" created.
nano@nahuel-ubuntu:~$
```

Espacio del LV.
Le estoy diciendo que agarre un 50% del espacio existente dentro del VG llamado vg_datos

Nombre del LV que estoy creando

En ese caso le estoy pasando un -l, lo que indica que le voy a pasar un tamaño en porcentajes

Si le paso un -L, le podría tirar un tamaño específico, por ejemplo:

```
lvcreate -L +1G nombreDelVg -n nombreDelLv
```

Y finalmente así vemos cómo se crearon los LV

sudo lvs

```
nano@nahuel-ubuntu:~$ sudo lvs
  LV   Home   VG     Attr       LSize   Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  lv_peliculas  vg_datos -wi-a---- 1020,00m
  lv_series      vg_datos -wi-a---- 512,00m
```

Formatear los LV

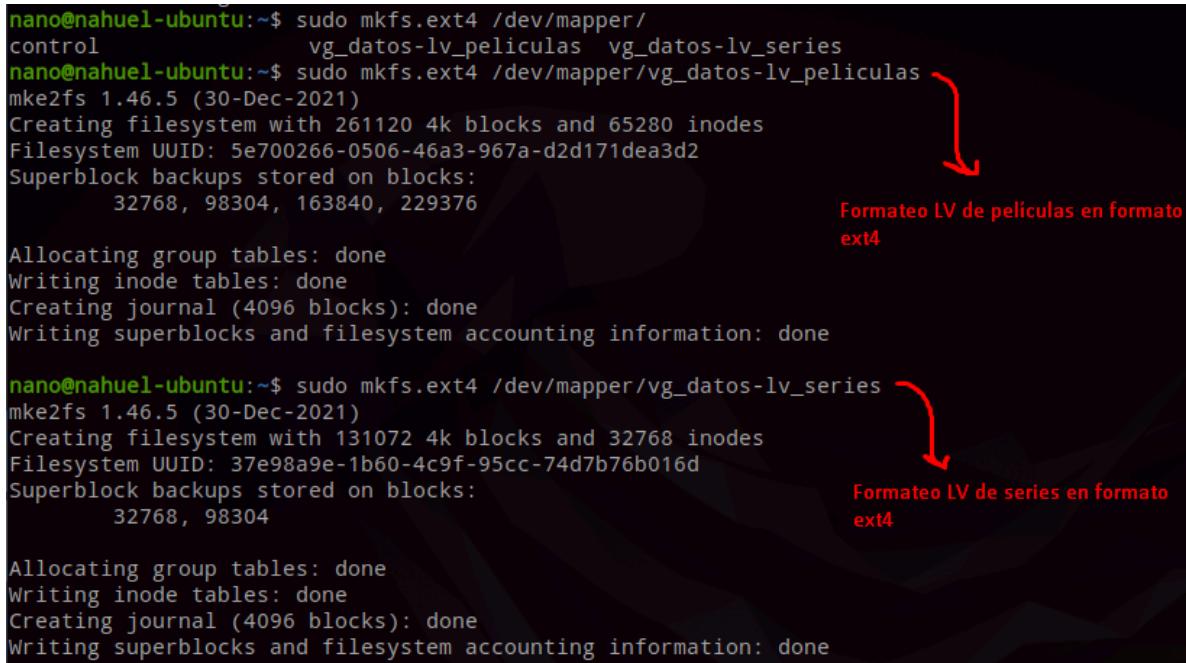
sudo mkfs.ext4 /dev/mapper/nombreDelLV

OJO, porque acá ese nombreDelLV no es realmente el que escribí yo, es uno que se me crea en base al VG y al LV.

Darle TAB para ver las sugerencias, y elegir el correcto.

```
nano@nahuel-ubuntu:~$ sudo mkfs.ext4 /dev/mapper/
control          vg_datos-lv_peliculas  vg_datos-lv_series
nano@nahuel-ubuntu:~$ sudo mkfs.ext4 /dev/mapper/vg_datos-lv_peliculas
mke2fs 1.46.5 (30-Dec-2021)
Creating filesystem with 261120 4k blocks and 65280 inodes
Filesystem UUID: 5e700266-0506-46a3-967a-d2d171dea3d2
Superblock backups stored on blocks:
      32768, 98304, 163840, 229376
Allocating group tables: done
Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done

nano@nahuel-ubuntu:~$ sudo mkfs.ext4 /dev/mapper/vg_datos-lv_series
mke2fs 1.46.5 (30-Dec-2021)
Creating filesystem with 131072 4k blocks and 32768 inodes
Filesystem UUID: 37e98a9e-1b60-4c9f-95cc-74d7b76b016d
Superblock backups stored on blocks:
      32768, 98304
Allocating group tables: done
Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done
```



Crear los directorios en donde quiero montar mis LV.

sudo mkdir -p /datos/{peliculas,series}

Y ahora los monto

sudo mount nombreDelLv nombreDirectorio

```
nano@nahuel-ubuntu:~$ sudo mount /dev/mapper/vg_datos-lv_peliculas /datos/peliculas/
nano@nahuel-ubuntu:~$ sudo mount /dev/mapper/vg_datos-lv_series /datos/series/
```

Ojo acá también, no es el nombreDelLv directo, es el nombre que existe dentro de **/dev/mapper/**

Revisar que se haya montado correctamente:

`df -h`

Filesystem	Size	Used	Avail	Use%	Mounted on
tmpfs	775M	1,6M	773M	1%	/run
/dev/sda3	24G	16G	6,9G	70%	/
tmpfs	3,8G	0	3,8G	0%	/dev/shm
tmpfs	5,0M	4,0K	5,0M	1%	/run/lock
/dev/sda2	512M	6,1M	506M	2%	/boot/efi
tmpfs	775M	184K	775M	1%	/run/user/1000
/dev/mapper/vg_datos-lv_peliculas	986M	24K	919M	1%	/datos/peliculas
/dev/mapper/vg_datos-lv_series	488M	24K	452M	1%	/datos/series

Vemos que los 2 últimos son mis LV montadas en mis directorios

Tiramos un comando para escribir un nuevo archivo dentro de /datos/peliculas

`sudo dd if=/dev/zero of=/datos/peliculas/peli1.avi bs=1M count=900`

```
nano@nahuel-ubuntu:~$ sudo dd if=/dev/zero of=/datos/peliculas/peli1.avi bs=1M count=900
900+0 records in
900+0 records out
943718400 bytes (944 MB, 900 MiB) copied, 2,37981 s, 397 MB/s
```

Esto lo hacemos con el objetivo de crear un archivo para que ocupe espacio.

<code>ls -l /datos/peliculas/ -h</code>	Ahora tengo un archivo de 900M dentro de /datos/peliculas
<code>total 901M</code>	
<code>drwx----- 2 root root 16K jun 19 11:20 lost+found</code>	

<code>df -h</code>	Aca podemos ver que tengo ocupado el 90% del espacio
<code>Filesystem</code>	
<code>tmpfs</code>	
<code>/dev/sda3</code>	
<code>tmpfs</code>	
<code>tmpfs</code>	
<code>/dev/sda2</code>	
<code>tmpfs</code>	
<code>/dev/mapper/vg_datos-lv_peliculas</code>	
<code>/dev/mapper/vg_datos-lv_series</code>	

Extender LV

Al principio nosotros usamos el disco de 5GB para trabajar, pero también teníamos otro de 2GB sin uso.

Vamos a usar ese disco de 2GB.

Sobre ese disco, usando el comando fdisk, vamos a crear unas nuevas particiones, de type

Linux LVM.

No es necesario que sean varias, puede ser 1 sola.

```
nano@nahuel-ubuntu:~$ sudo fdisk -l /dev/sdc
Disk /dev/sdc: 2 GiB, 2147483648 bytes, 4194304 sectors
Disk model: VBOX HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x739ab89b

Device      Boot   Start     End Sectors  Size Id Type
/dev/sdc1          2048 2099199 2097152    1G 8e Linux LVM
/dev/sdc2      2099200 4194303 2095104 1023M 8e Linux LVM }
```

Creamos 2 particiones pq pintó, pero podría crear 1 sola, o las que quiera. Lo importante es que sean de type Linux LVM (8e)

Una vez que tengo eso, creo los PV para esas 2 particiones

```
nano@nahuel-ubuntu:~$ sudo pvcreate /dev/sdc1 /dev/sdc2
Physical volume "/dev/sdc1" successfully created.
Physical volume "/dev/sdc2" successfully created.
```

Para verlas tiro un sudo pvs

```
nano@nahuel-ubuntu:~$ sudo pvs
PV   Home   VG       Fmt  Attr PSize   PFree
/dev/sdb1  vg_datos lvm2 a--    <2,00g  512,00m
/dev/sdc1           lvm2 ---    1,00g   1,00g
/dev/sdc2           lvm2 --- 1023,00m 1023,00m } Estas son las que creé recien
```

Vemos que los 2 últimos PV que agregué recien, no están asociados a ningún VG.

sudo vgextend nombreDelVG nombreDelPV

```
nano@nahuel-ubuntu:~$ sudo vgextend vg_datos /dev/sdc1 → extiendo vg_datos con el PV llamado /dev/sdc1
Volume group "vg_datos" successfully extended
nano@nahuel-ubuntu:~$ sudo pvs
PV   VG       Fmt  Attr PSize   PFree
/dev/sdb1  vg_datos lvm2 a--    <2,00g  512,00m
/dev/sdc1  vg_datos lvm2 a-- 1020,00m 1020,00m
/dev/sdc2           lvm2 --- 1023,00m 1023,00m
nano@nahuel-ubuntu:~$ sudo vgs
VG       #PV #LV #SN Attr   VSize VFree } Ahora veo que mi VG ahora tiene 2 PV, y que es más grande
```

Antes el size del VG era 2,00g

Ahora es de 2,99g, pq le agregué más espacio

Ahora, si quiero tener más espacio, voy a tener que extender mi LV.

sudo lvextend -L +1G /dev/mapper/vg_datos-lv_peliculas

```
nano@nahuel-ubuntu:~$ sudo lvextend -L +1G /dev/mapper/vg_datos-lv_peliculas
  Size of logical volume vg_datos/lv_peliculas changed from 1020,00 MiB (255 extents) to <2,00 GiB (511 extents).
  Logical volume vg_datos/lv_peliculas successfully resized.
nano@nahuel-ubuntu:~$ sudo lvs
  LV       VG       Attr     LSize   Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  lv_peliculas  vg_datos -wi-ao--- <2,00g } Ahora le agregué 1G
  lv_series    vg_datos -wi-ao--- 512,00m
```

Una vez extendido el LV, tengo que cambiarle el size al fs (file system)

sudo resize2fs /dev/mapper/vg_datos-lv_peliculas

```
nano@nahuel-ubuntu:~$ sudo resize2fs /dev/mapper/vg_datos-lv_peliculas → Hago el resize
resize2fs 1.46.5 (30-Dec-2021)
Filesystem at /dev/mapper/vg_datos-lv_peliculas is mounted on /datos/peliculas; on-line resizing require
d
old_desc_blocks = 1, new_desc_blocks = 1
The filesystem on /dev/mapper/vg_datos-lv_peliculas is now 523264 (4k) blocks long.

nano@nahuel-ubuntu:~$ df -h /datos/peliculas/
Filesystem           Size   Used  Avail Use% Mounted on
/dev/mapper/vg_datos-lv_peliculas  2,0G  901M  986M  48% /datos/peliculas
```

Y ahora tengo en uso el 48%

En el parcial vamos a tener que agregar 2 discos rígidos y generar unos LV y luego agrandarlos.

Y montarlos.

Tener en cuenta que para agrandar, no tenemos que desmontar.

Clase 8

Bash Scripting

Lo ideal sería llegar al parcial con un script ya hecho que me permita crear usuarios, grupos, contraseñas, etc.

El script debe permitirte:

- Crear un grupo
- Crear un usuario
- Poder setearle a ese usuario una pass de otro usuario ya existente. (Para esto podemos ver cómo obtener la pass hasheada de un usuario)

Repo con algunos scripts que armó el profe:

https://github.com/upszot/UTN-FRA_SO_Bash

Ansible Teoría

Empezamos a ver esto a la hora 7:45pm de la clase

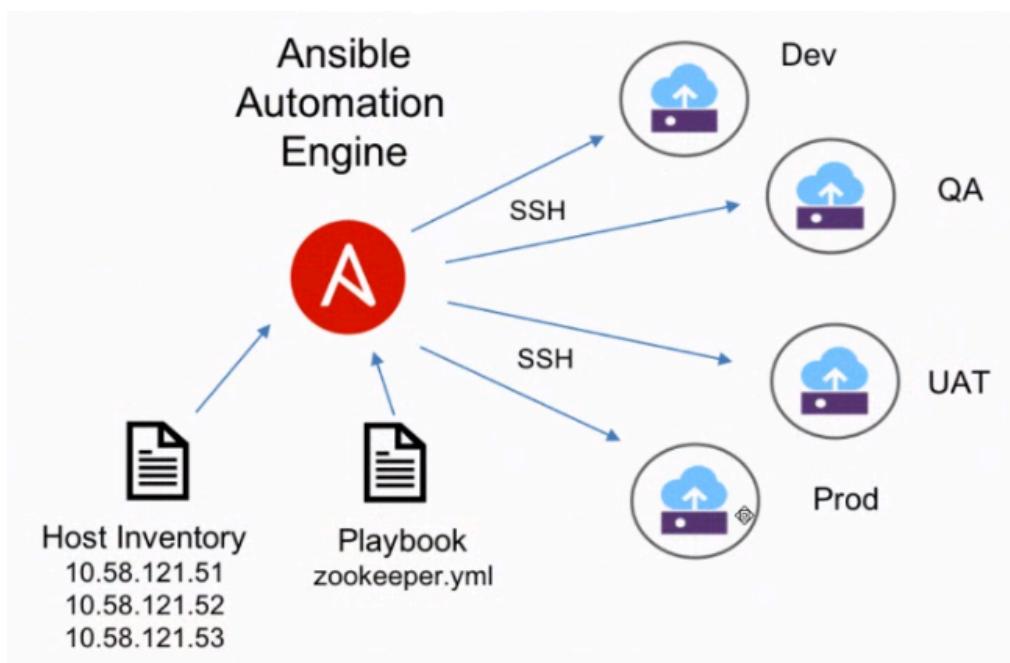
Herramienta de RedHat para automatizar tareas y abstraernos de en qué distribución de linux estamos.

Incluso con Ansible le podemos pegar a un windows.

Para ejecutarlo si o si tenemos que tener linux.

Repo con ejemplos del profe:

https://github.com/upszot/UTN-FRA_SO_Ansible



Infraestructura por código: Programar como queremos nuestra computadora, que programas queremos que tengan etc. Entonces nosotros ejecutamos este script para diferentes computadoras y nos quedan todas igual

Ansible Práctica

Empezamos a ver esto a la hora 8:10pm de la clase

Clonamos el siguiente repo: https://github.com/upszot/UTN-FRA_SO_Ansible

Instalamos ansible:

sudo apt update

sudo apt install ansible

Validar que el servicio del servidor de ssh esté corriendo.

Levantar el sistema de ssh para que quede corriendo y quede en enabled.

Cosa de no tener que levantarla cada vez que prendemos la compu

En ubuntu primero tengo que instalar el openssh-server:

sudo apt install openssh-server

Y después miro el status de la siguiente manera:

sudo systemctl status sshd

Y lo habilito con el siguiente comando:

sudo systemctl enable --now sshd

El servidor ssh me sirve para permitir que otras personas se conecten a mi equipo

```
nano@nahuel-ubuntu:~$ sudo systemctl status sshd
● ssh.service - OpenBSD Secure Shell server
  Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
  Active: active (running) since Mon 2024-06-03 20:22:06 -03; 20s ago
    Docs: man:sshd(8)
          man:sshd_config(5)
   Main PID: 8272 (sshd)
      Tasks: 1 (limit: 9133)
     Memory: 1.7M
        CPU: 19ms
      CGroup: /system.slice/ssh.service
              └─8272 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

jun 03 20:22:06 nahuel-ubuntu systemd[1]: Starting OpenBSD Secure Shell server...
jun 03 20:22:06 nahuel-ubuntu sshd[8272]: Server listening on 0.0.0.0 port 22.
jun 03 20:22:06 nahuel-ubuntu sshd[8272]: Server listening on :: port 22.
jun 03 20:22:06 nahuel-ubuntu systemd[1]: Started OpenBSD Secure Shell server.
```

ahí veo que está enabled y running

Ahora vamos a hacer que otras personas se conecten a la computadora

En este ejemplo nos vamos a conectar a nuestra propia computadora virtual, para no levantar otra VM

ssh localhost

Ver quienes están conectados a esta computadora

who

```
nano@nahuel-ubuntu:~$ who
nano      tty2          2024-06-03 18:30 (tty2)
nano      pts/1          2024-06-03 20:29 (127.0.0.1)
```

También puedo usar la w

w

```
nano@nahuel-ubuntu:~$ w
20:30:45 up  2:01,  2 users,  load average: 0,13, 0,08, 0,05
USER     TTY     FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
nano     tty2     tty2            18:30    1:55m  0.02s  0.02s /usr/libexec/
nano     pts/1    127.0.0.1       20:29    5.00s  0.02s  0.00s w
```

Generar una key

ssh-keygen

Y le damos todo enter

Y nos genera el .ssh

```
nano@nahuel-ubuntu:~$ ls -la .ssh/
total 24
drwx----- 2 nano nano 4096 jun  3 20:29 .
drwxr-x--- 20 nano nano 4096 jun  3 20:17 ..
-rw----- 1 nano nano  419 may 19 13:11 id_ed25519
-rw-r--r-- 1 nano nano  104 may 19 13:11 id_ed25519.pub
-rw----- 1 nano nano 1956 jun  3 20:29 known_hosts
-rw----- 1 nano nano 1120 jun  3 20:29 known_hosts.old
```

Ahora tenemos que cruzar esa clave pública. (el archivo llamado id_ed25519.pub)

ssh-copy-id localhost

```
nano@nahuel-ubuntu:~$ ssh-copy-id localhost
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompt
ed now it is to install the new keys
nano@localhost's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'localhost'"
and check to make sure that only the key(s) you wanted were added.
```

Con eso cruzamos la clave

```
nano@nahuel-ubuntu:~$ ls -la .ssh/
total 28
drwx----- 2 nano nano 4096 jun  3 20:37 .
drwxr-x--- 20 nano nano 4096 jun  3 20:17 ..
-rw----- 1 nano nano  104 jun  3 20:37 authorized_keys
-rw----- 1 nano nano  419 may 19 13:11 id_ed25519
-rw-r--r-- 1 nano nano  104 may 19 13:11 id_ed25519.pub
-rw----- 1 nano nano 1956 jun  3 20:29 known_hosts
-rw----- 1 nano nano 1120 jun  3 20:29 known_hosts.old
```

aca se creó la clave publica

Ahora, el archivo **authorized_keys** va a ser igual al **id_ed25519.pub**

Para validar esto, puedo tirar el siguiente comando:

```
diff .ssh/authorized_keys .ssh/id_ed25519.pub
```

Finalmente, cuando tiremos un ssh localhost, ya no nos va a pedir la clave. Me logueo de una.

También puedo conectarme al localhost mediante la ip en vez de por el nombre
ssh 127.0.0.1

Ip de localhost: 127.0.0.1

Ejemplos de ansible:

Ejemplo 1

Me cloné el repo de práctica de ansible y tiré el siguiente comando dentro del directorio ejemplo_01

ansible-playbook -i inventory playbook.yml

```
nano@nahuel-ubuntu:~/UTN-FRA_S0_Ansible/ejemplo_01$ ansible-playbook -i inventory playbook.yml
[WARNING]: A duplicate localhost-like entry was found (localhost). First found
localhost was 127.0.0.1

PLAY [testing,desarrollo,produccion] ****
TASK [Gathering Facts] ****
ok: [localhost]
ok: [127.0.0.1]
fatal: [no-existe.com]: UNREACHABLE! => {"changed": false, "msg": "Failed to connect to the host via ssh: ssh: connect to host no-existe.com port 22: Connection refused", "unreachable": true}

TASK [Mensaje 1] ****
ok: [127.0.0.1] => {
    "msg": "Ejemplo de playbook - 01 "
}
ok: [localhost] => {
    "msg": "Ejemplo de playbook - 01 "
}

TASK [Mensaje 2] ****
ok: [127.0.0.1] => {
    "msg": [
        "Soy: nahuel-ubuntu ",
        "Estoy en el grupo: ['testing']"
    ]
}
ok: [localhost] => {
    "msg": [
        "Soy: nahuel-ubuntu ",
        "Estoy en el grupo: ['produccion']"
    ]
}

PLAY RECAP ****
127.0.0.1      : ok=3    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
localhost       : ok=3    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
no-existe.com   : ok=0    changed=0    unreachable=1   failed=0    skipped=0    rescued=0    ignored=0
```

Me quise conectar a 3 computadoras, pude a las 2 primeras pq existen. A la 3era no, porque no existe.

Ejemplo 3:

dentro del directorio /UTN-FRA_SO_Ansible/ejemplo_03/role_ejemplo1

Si nosotros tenemos que hacer muchas cosas (cambiar nombre del equipo, asignar una ip, configurar una BBDD, etc), nos conviene separar en roles.

Cada rol va a hacer una tarea específica.

Para generar el rol, voy a tener que tirar el siguiente comando:

ansible-galaxy role init rol_ejemplo1

De todas formas, por ahora no lo vamos a tirar, ya que esto lo tendremos que hacer cuando estemos practicando. En el caso de este ejemplo 3, el rol ya está creado.

Nosotros vamos a tener que hacer todos los roles por separado, y después crear un playbook que llame a todos los roles.

```
nano@nahuel-ubuntu:~/UTN-FRA_SO_Ansible/ejemplo_03/role_ejemplo1$ ll
total 48
drwxrwxr-x 9 nano nano 4096 jun  3 20:11 .
drwxrwxr-x 3 nano nano 4096 jun  3 20:11 ..
-rw-rw-r-- 1 nano nano  281 jun  3 20:11 ansible.cfg
drwxrwxr-x 2 nano nano 4096 jun  3 20:11 defaults/
drwxrwxr-x 2 nano nano 4096 jun  3 20:11 handlers/
drwxrwxr-x 2 nano nano 4096 jun  3 20:11 logs/
drwxrwxr-x 2 nano nano 4096 jun  3 20:11 meta/
-rw-rw-r-- 1 nano nano 1328 jun  3 20:11 README.md
drwxrwxr-x 2 nano nano 4096 jun  3 20:11 tasks/
drwxrwxr-x 2 nano nano 4096 jun  3 20:11 tests/
-rw-rw-r-- 1 nano nano  539 jun  3 20:11 .travis.yml
drwxrwxr-x 2 nano nano 4096 jun  3 20:11 vars/
```

Esta estructura es una “buena práctica”. Poner cada cosa en cada lugar. Se genera automáticamente con el comando.

Una vez generados los roles, en el ansible.cfg tengo que indicarle a dónde tiene que ir a buscar los roles.

```
nano@nahuel-ubuntu:~/UTN-FRA_SO_Ansible/ejemplo_03/role_ejemplo1$ cat ansible.cfg
[defaults]
host_key_checking = False
roles_path = /tmp/.../Le indico donde tiene que ir a buscar los roles
;ask_vault_pass = True
log_path = ./logs/ansible_output.log
nocows=1

[paramiko_connection]
record_host_keys = False

[ssh_connection]
ssh_args = -o ControlMaster=auto -o ControlPersist=60s -o UserKnownHostsFile=/dev/null
```

Ejecutamos el playbook que está dentro de la carpeta test, en el inventario que está dentro de la carpeta test

```
ansible-playbook -i tests/inventory tests/test.yml
```

```
nano@nahuel-ubuntu:~/UTN-FRA_SO_Ansible/ejemplo_03/role_ejemplo1$ ansible-playbook -i tests/inventory tests/test.yml
PLAY [test] ****
TASK [Gathering Facts] ****
ok: [localhost]
TASK [role_ejemplo1 : mensaje] ****
ok: [localhost] => {
    "msg": "Ejemplo 03 - role ansible"
}
PLAY RECAP ****
localhost                  : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Este comando indica que estamos ejecutando el playbook que está en la carpeta test (llamado test.yml), con el inventario que también está en la carpeta test (llamando inventory)

Crear un nuevo rol de Ansible

```
ansible-galaxy role init role_ejemplo2
```

```
nano@nahuel-ubuntu:~/UTN-FRA_SO_Ansible/ejemplo_03$ ansible-galaxy role init role_ejemplo2
- Role role_ejemplo2 was created successfully
nano@nahuel-ubuntu:~/UTN-FRA_SO_Ansible/ejemplo_03$ ll
total 20
drwxrwxr-x 4 nano nano 4096 jun  3 21:45 .
drwxrwxr-x 8 nano nano 4096 jun  3 20:11 ..
-rw-rw-r-- 1 nano nano 1563 jun  3 20:11 README.md
drwxrwxr-x 9 nano nano 4096 jun  3 20:11 role_ejemplo1/
drwxrwxr-x 8 nano nano 4096 jun  3 21:45 role_ejemplo2/
nano@nahuel-ubuntu:~/UTN-FRA_SO_Ansible/ejemplo_03$ cd role_ejemplo2/
nano@nahuel-ubuntu:~/UTN-FRA_SO_Ansible/ejemplo_03/role_ejemplo2$ ls -l
total 28
drwxrwxr-x 2 nano nano 4096 jun  3 21:45 defaults
drwxrwxr-x 2 nano nano 4096 jun  3 21:45 handlers
drwxrwxr-x 2 nano nano 4096 jun  3 21:45 meta
-rw-rw-r-- 1 nano nano 1328 jun  3 21:45 README.md
drwxrwxr-x 2 nano nano 4096 jun  3 21:45 tasks
drwxrwxr-x 2 nano nano 4096 jun  3 21:45 tests
drwxrwxr-x 2 nano nano 4096 jun  3 21:45 vars
nano@nahuel-ubuntu:~/UTN-FRA_SO_Ansible/ejemplo_03/role_ejemplo2$ cat tests/
inventory  test.yml
nano@nahuel-ubuntu:~/UTN-FRA_SO_Ansible/ejemplo_03/role_ejemplo2$ cat tests/test.yml
---
- hosts: localhost
  remote_user: root
  roles:
    - role_ejemplo2
nano@nahuel-ubuntu:~/UTN-FRA_SO_Ansible/ejemplo_03/role_ejemplo2$
```

Y se crea el directorio con la estructura correcta del nuevo rol.

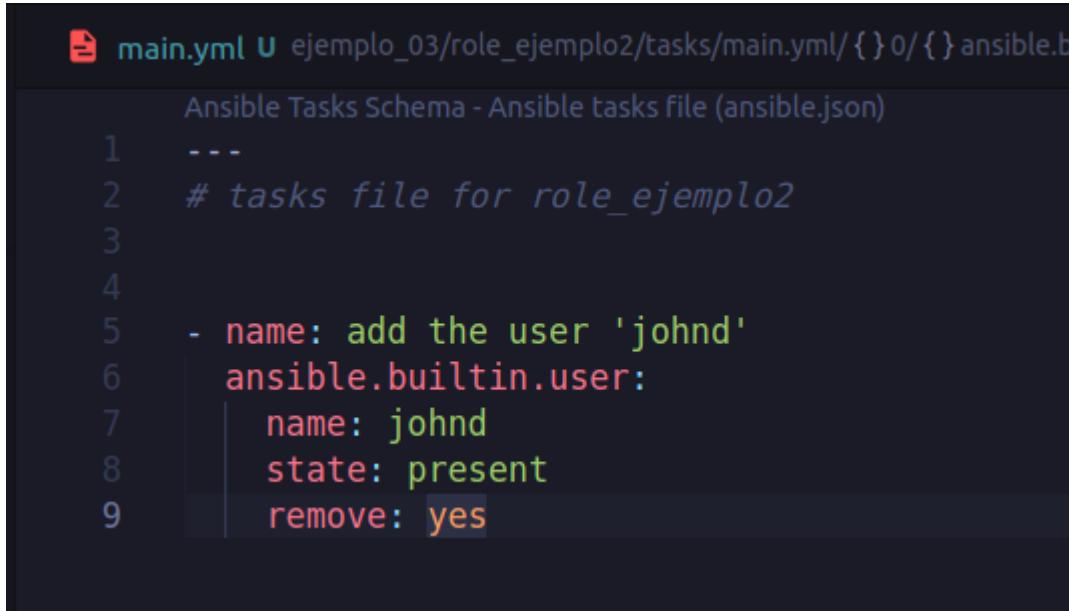
Nosotros vamos a tener que crear roles que hagan cosas distintas

En este archivo vamos a escribir las instrucciones de la tarea

```
nano@nahuel-ubuntu:~/UTN-FRA_SO_Ansible/ejemplo_03/role_ejemplo2$ cat tasks/main.yml
---
# tasks file for role ejemplo2
```

Si editamos ese file, ahí podemos hacer la tarea que queramos. Por ejemplo, crear un usuario.

[De acá](#) podemos sacar ejemplos de cómo crear usuarios por ejemplo



```
main.yml
Ansible Tasks Schema - Ansible tasks file (ansible.json)
1  ---
2  # tasks file for role_ejemplo2
3
4
5  - name: add the user 'johnd'
6    ansible.builtin.user:
7      name: johnd
8      state: present
9      remove: yes
```

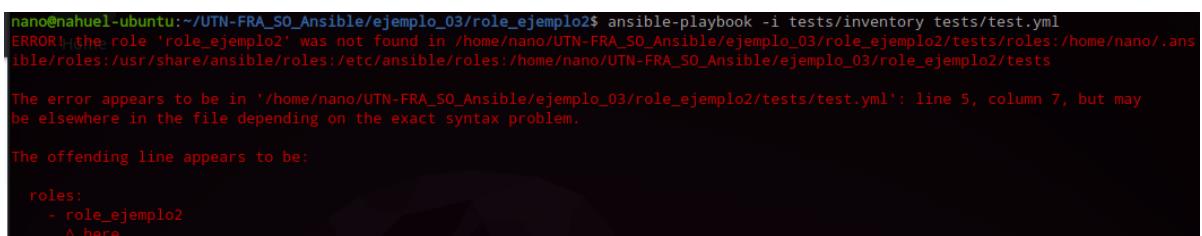
Así quedaría nuestro .yml

El name y el nombre del módulo están al mismo nivel de indentación
Los otros 3 estan mas adentro

Si no hacemos esto con la identación nos va a tirar error

Y ahora lo ejecutamos con el siguiente comando:

ansible-playbook -i tests/inventory tests/test.yml



```
nano@nahuel-ubuntu:~/UTN-FRA_SO_Ansible/ejemplo_03/role_ejemplo2$ ansible-playbook -i tests/inventory tests/test.yml
ERROR! the role 'role_ejemplo2' was not found in /home/nano/UTN-FRA_SO_Ansible/ejemplo_03/role_ejemplo2/tests/roles:/home/nano/.ansible/roles:/usr/share/ansible/roles:/etc/ansible/roles:/home/nano/UTN-FRA_SO_Ansible/ejemplo_03/role_ejemplo2/tests

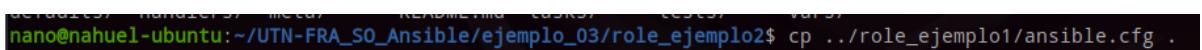
The error appears to be in '/home/nano/UTN-FRA_SO_Ansible/ejemplo_03/role_ejemplo2/tests/test.yml': line 5, column 7, but may
be elsewhere in the file depending on the exact syntax problem.

The offending line appears to be:

  roles:
    - role_ejemplo2
      ^ here
```

Ese error es pq no encontró el rol. Esto pasa pq lo va a buscar a un lugar donde no está el rol.

Para cambiar esto, Tengo que copiarme el ansible.cfg que me está faltando



```
nano@nahuel-ubuntu:~/UTN-FRA_SO_Ansible/ejemplo_03/role_ejemplo2$ cp ../role_ejemplo1/ansible.cfg .
```

copié el ansible.cfg de role_ejemplo1/ dentro del directorio en el que estoy parado.

Así me quedó

```
ansible.cfg u ejemplo_03/role_ejemplo2/ansible.cfg
1 [defaults]
2 host_key_checking = False
3 roles_path = /tmp:../
4 ;ask_vault_pass = True
5 log_path = ./logs/ansible_output.log
6 nocows=1
7
8
9 [paramiko_connection]
10 record_host_keys = False
11
12 [ssh_connection]
13 ssh_args = -o ControlMaster=auto -o ControlPersist=60s -o UserKnownHostsFile=/dev/null
```

Ahora, ejecuto devuelta

```
nano@nahuel-ubuntu:~/UTN-FRA_SO_Ansible/ejemplo_03/role_ejemplo2$ ansible-playbook -i tests/inventory tests/test.yml
[WARNING]: log file at /home/nano/UTN-FRA_SO_Ansible/ejemplo_03/role_ejemplo2/logs/ansible_output.log is not writeable and we cannot create it, aborting

PLAY [localhost] ****
TASK [Gathering Facts] ****
fatal: [localhost]: UNREACHABLE! => {"changed": false, "msg": "Failed to connect to the host via ssh: Warning: Permanently added 'localhost' (ED25519) to the list of known hosts.\r\nroot@localhost: Permission denied (publickey,password).", "unreachable": true}

PLAY RECAP ****
localhost                  : ok=0      changed=0      unreachable=1      failed=0      skipped=0      rescued=0      ignored=0
```

Me tiró ese error que dice “Permission denied”

Eso pasó pq estoy queriendo agregar usuarios con mi usuario, en vez de hacerlo con el usuario root.

Para fixarlo, tengo que editar el yml, para que quede así:
(tuve que agregar el become: yes)

```
---
# tasks file for role_ejemplo2

- name: add the user 'johnd'
  become: yes
  ansible.builtin.user:
    name: johnd
    state: present
    remove: yes
```

Otra cosa que puedo hacer en vez de pasarle el become: yes, es tirar lo siguiente:
ansible-playbook -i tests/inventory tests/test.yml -u nano -k

El próximo paso que tengo que hacer es tirar un **mkdir logs** para generar la carpeta logs

Si tiro un **cat test/inventory** me tira localhost

```
nano@nahuel-ubuntu:~/UTN-FRA_SO_Ansible/ejemplo_03/role_ejemplo2$ cat tests/inventory  
localhost → antes  
nano@nahuel-ubuntu:~/UTN-FRA_SO_Ansible/ejemplo_03/role_ejemplo2$ cd ..  
nano@nahuel-ubuntu:~/UTN-FRA_SO_Ansible/ejemplo_03$ ls  
README.md  role_ejemplo1  role_ejemplo2  
nano@nahuel-ubuntu:~/UTN-FRA_SO_Ansible/ejemplo_03$ cd role_ejemplo2/  
nano@nahuel-ubuntu:~/UTN-FRA_SO_Ansible/ejemplo_03/role_ejemplo2$ cat ../role_ejemplo1/tests/inventory  
[test]  
localhost ansible_connection=local  
  
nano@nahuel-ubuntu:~/UTN-FRA_SO_Ansible/ejemplo_03/role_ejemplo2$ vim tests/inventory  
nano@nahuel-ubuntu:~/UTN-FRA_SO_Ansible/ejemplo_03/role_ejemplo2$ cat tests/inventory  
localhost ansible_connection=local → ahora  
nano@nahuel-ubuntu:~/UTN-FRA_SO_Ansible/ejemplo_03/role_ejemplo2$
```

Lo modifco para que quede así:

"localhost ansible_connection=local"

Ese no es un comando, es el texto que tiene que quedar en el inventory

Vuelvo a ejecutar:

```
nano@nahuel-ubuntu:~/UTN-FRA_SO_Ansible/ejemplo_03/role_ejemplo2$ ansible-playbook -i tests/inventory tests/test.yml  
[WARNING]: log file at /home/nano/UTN-FRA_SO_Ansible/ejemplo_03/role_ejemplo2/logs/ansible_output.log is not writeable  
and we cannot create it, aborting  
  
PLAY [localhost] *****  
TASK [Gathering Facts] *****  
ok: [localhost]  
  
TASK [role_ejemplo2 : add the user 'johnd'] *****  
changed: [localhost]  
  
PLAY RECAP *****  
localhost : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Finalmente anduve

En amarillo dice “changed”, por lo que entendemos que realizó una modificación. Es decir, creó el usuario.

Para validarla, podemos tirar un **id johnd** y vemos que efectivamente lo creó

```
localhost ansible_connection=local  
nano@nahuel-ubuntu:~/UTN-FRA_SO_Ansible/ejemplo_03/role_ejemplo2$ ansible-playbook -i tests/inventory tests/test.yml  
[WARNING]: log file at /home/nano/UTN-FRA_SO_Ansible/ejemplo_03/role_ejemplo2/logs/ansible_output.log is not writeable  
and we cannot create it, aborting  
  
PLAY [localhost] *****  
TASK [Gathering Facts] *****  
ok: [localhost]  
  
TASK [role_ejemplo2 : add the user 'johnd'] *****  
changed: [localhost]  
  
PLAY RECAP *****  
localhost : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0  
  
nano@nahuel-ubuntu:~/UTN-FRA_SO_Ansible/ejemplo_03/role_ejemplo2$ id johnd  
uid=1001(johnd) gid=1001(johnd) groups=1001(johnd) → creó el usuario
```

Finalmente, así es cómo tiene que quedar la estructura del rol:

```
└─ role_ejemplo2
    ├─ defaults
    ├─ handlers
    ├─ log
    ├─ meta
    ├─ tasks
    └─ tests
        ├─ inventory
        ├─ test.yml
        └─ vars
            ├─ ansible.cfg
            └─ README.md
```

Ejecutar el playbook como si estuviese logueado con usuario

Me voy al directorio **/UTN-FRA_SO_Ansible/playbook_pruebas**

Dentro de este directorio, el playbook.yml tiene un prompt, que es para pedirle al usuario que ingrese un dato.

```
nano@nahuel-ubuntu:~/UTN-FRA_SO_Ansible/playbook_pruebas$ ll
total 28
drwxrwxr-x 4 nano nano 4096 jun  3 20:11 .
drwxrwxr-x 8 nano nano 4096 jun  3 20:11 ..
-rw-rw-r-- 1 nano nano 307 jun  3 20:11 ansible.cfg
drwxrwxr-x 4 nano nano 4096 jun  3 20:11 inventory/
-rw-rw-r-- 1 nano nano 282 jun  3 20:11 playbook.yml
-rw-rw-r-- 1 nano nano 1203 jun  3 20:11 README.md
drwxrwxr-x 3 nano nano 4096 jun  3 20:11 roles/
nano@nahuel-ubuntu:~/UTN-FRA_SO_Ansible/playbook_pruebas$ ll roles/
total 12
drwxrwxr-x 3 nano nano 4096 jun  3 20:11 .
drwxrwxr-x 4 nano nano 4096 jun  3 20:11 ..
drwxrwxr-x 9 nano nano 4096 jun  3 20:11 multi_Pruebas/
nano@nahuel-ubuntu:~/UTN-FRA_SO_Ansible/playbook_pruebas$ cat playbook.yml
---
- hosts: all

  vars_prompt:
    - name: "NroPrueba"
      prompt: "Ingrese Nro de Prueba:"
      default: "1"
      private: no

  tasks:
    - include_role:
        name: multi_Pruebas

    - name: "Otra tarea"
      debug:
        msg: "Despues de la ejecucion del rol"
```

Pedimos un prompt al usuario

```
nano@nahuel-ubuntu:~/UTN-FRA_SO_Ansible/playbook_pruebas$ ll roles/multi_Pruebas/tasks/
total 40
drwxrwxr-x 3 nano nano 4096 jun  3 20:11 .
drwxrwxr-x 9 nano nano 4096 jun  3 20:11 ..
-rw-rw-r-- 1 nano nano 305 jun  3 20:11 main.yml
drwxrwxr-x 2 nano nano 4096 jun  3 20:11 _otros/
-rw-rw-r-- 1 nano nano 128 jun  3 20:11 prueba_1.yml
-rw-rw-r-- 1 nano nano 345 jun  3 20:11 prueba_2.yml
-rw-rw-r-- 1 nano nano 389 jun  3 20:11 prueba_3.yml
-rw-rw-r-- 1 nano nano 116 jun  3 20:11 prueba_4.yml
-rw-rw-r-- 1 nano nano 1343 jun  3 20:11 prueba_5.yml
-rw-rw-r-- 1 nano nano 1539 jun  3 20:11 prueba_useradd.yml
```

Nosotros vamos a clonar un directorio así y vamos a hacer ciertas pruebas, teniendo que crear un yml para el parcial, que ejecute las cosas que necesitemos.

Vemos que en la imagen hay varias tasks, que pueden ejecutar varios yml.

Ahora, para ejecutarlo, me paro en el directorio playbook_pruebas y tiro un **ansible-playbook -i inventory/hosts playbook.yml**

Y listo, ya debería andar.

Segun la version de ansible que tenga, en el /tasks/main.yml, este param se puede llamar **include** o **include_tasks**

```
msg: "Dentro del rol"
tags:
  - cartel
  Carpeta personal de
    - luca
  - name: "Mostrando facts - No incluir"
    include_tasks: _otros/otros.yml
    tags:
      - apelera
      - never
      - otros

  - debug:
    msg: "archivo a incluir Nro: {{ NroPrueba }}"

  - name: "Incluyo archivo prueba_NroPrueba"
    include_tasks: prueba_{{ NroPrueba }}.yml
```

Ejemplo de otra prueba, la 4

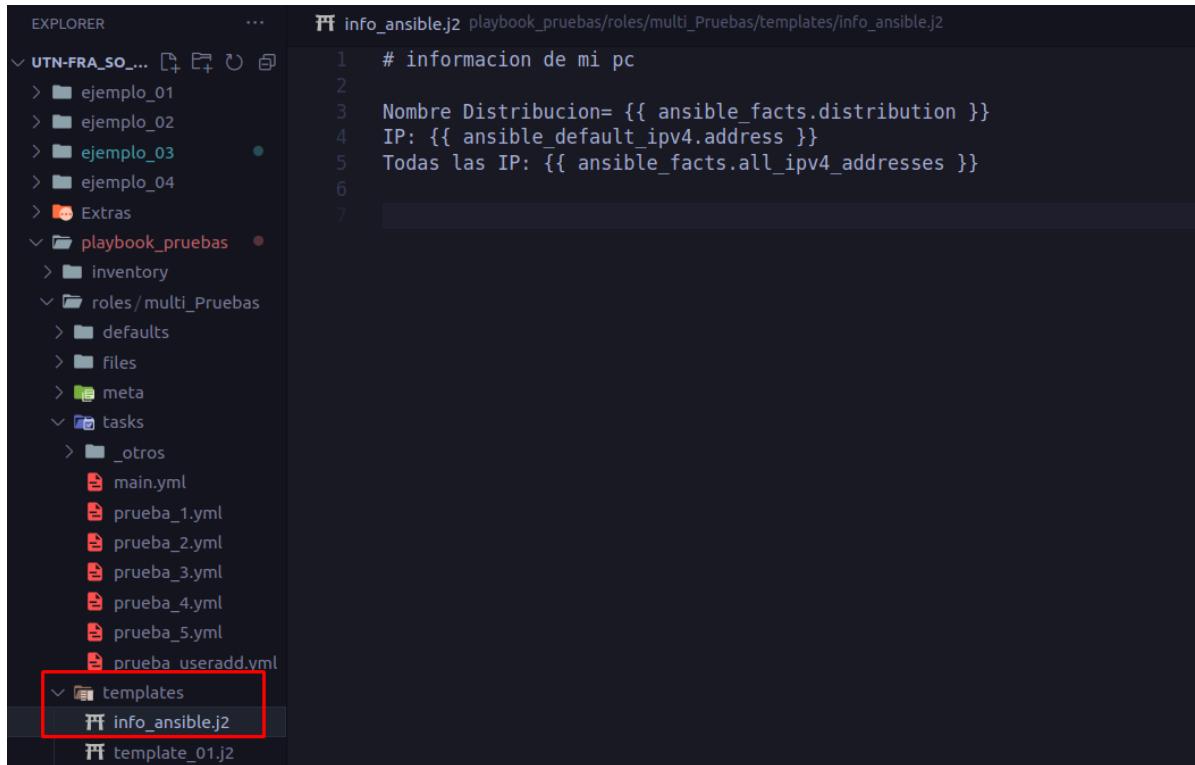
https://github.com/upszot/UTN-FRA_SO_Ansible/blob/main/playbook_pruebas/roles/multi_P_ruebas/tasks/prueba_4.yml

```
Code Blame

1  ---
2
3  - name: "Agrego archivo desde template"
4    template:
5      src: info_ansible.j2
6      dest: /tmp/info_ansible.txt
```

Esa task ejecuta ese yml, que lo que hace es copiar el archivo **info_ansible.j2**, y pegarlo en **/tmp/info_ansible.txt**

El contenido de info_ansible.j2 es el siguiente:



```
EXPLORER ... info_ansi... playbook_pruebas/roles/multi_Pru.../templates/info_ansi...  
UTN-FRA_SO... ejemp... Nombre Distribucion= {{ ansible_facts.distribution }}  
> ejemplo_01 IP: {{ ansible_default_ipv4.address }}  
> ejemplo_02 Todas las IP: {{ ansible_facts.all_ipv4_addresses }}  
> ejemplo_03  
> ejemplo_04  
> Extras  
playbook_pruebas  
> inventory  
roles/multi_Pru...  
> defaults  
> files  
> meta  
> tasks  
> _otros  
main.yml  
prueba_1.yml  
prueba_2.yml  
prueba_3.yml  
prueba_4.yml  
prueba_5.yml  
prueba_useradd.yml  
templates  
info_ansi...  
template_01.j2
```

En esa variable **ansible_facts** tengo la data de mi disco rígido, cpu, frecuencia, etc..

Yo puedo ir modificando este archivo e ir consumiendo cosas de esa variable, y cuando lo ejecute, va a quedar el archivo con esa data

Entonces, cuando ejecute ese playbook con esa task, va a realizar esa tarea, y en /tmp/info_ansi... se va a completar eso:

```
nano@nahuel-ubuntu:~/UTN-FRA_SO_Ansible/playbook_pruebas$ cat /tmp/info_ansi...  
# informacion de mi pc  
  
Nombre Distribucion= Ubuntu  
IP: 10.0.2.15  
Todas las IP: ['10.0.2.15', '172.17.0.1']
```

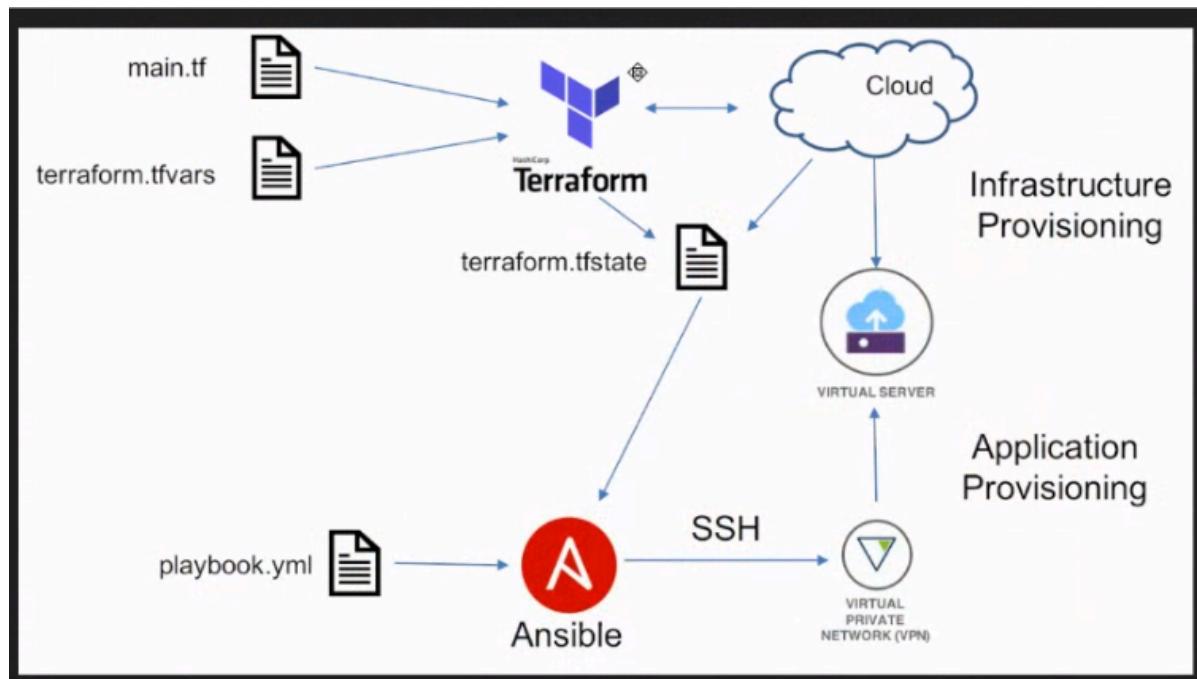
Ejemplo de consigna: Usa un módulo de template, y deja el archivo en /tmp/info...

Para el parcial vamos a necesitar saber cómo correr un archivo de ansible, y conocer las variables, y donde las tenemos que poner para ver como van asumiendo los valores. Vamos a tener que escribir un archivo con cierta info (pc, distro, etc), tomando la data de la variable **ansible_fact**

Clase 9

Ver la clase desde que arranca hasta las 19:10

Docker



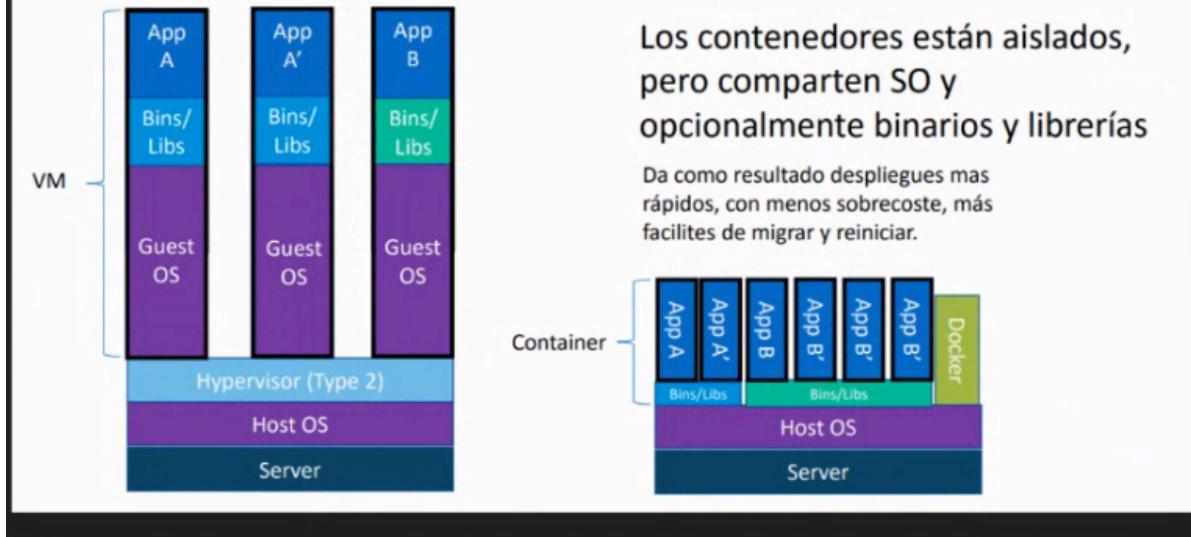
El objetivo de docker es empaquetar y distribuir aplicaciones

Actualmente estamos parados en la imagen de la izquierda.

Estamos en una computadora con Windows, que dentro tiene una VM con un SO completo.
Si ahí quiero levantar algo, voy a tener que instalarla dentro de ese SO.

Tengo 3 partes, simboliza 3 VM diferentes dentro de la misma computadora real.

Contenedores vs Maquinas Virtuales



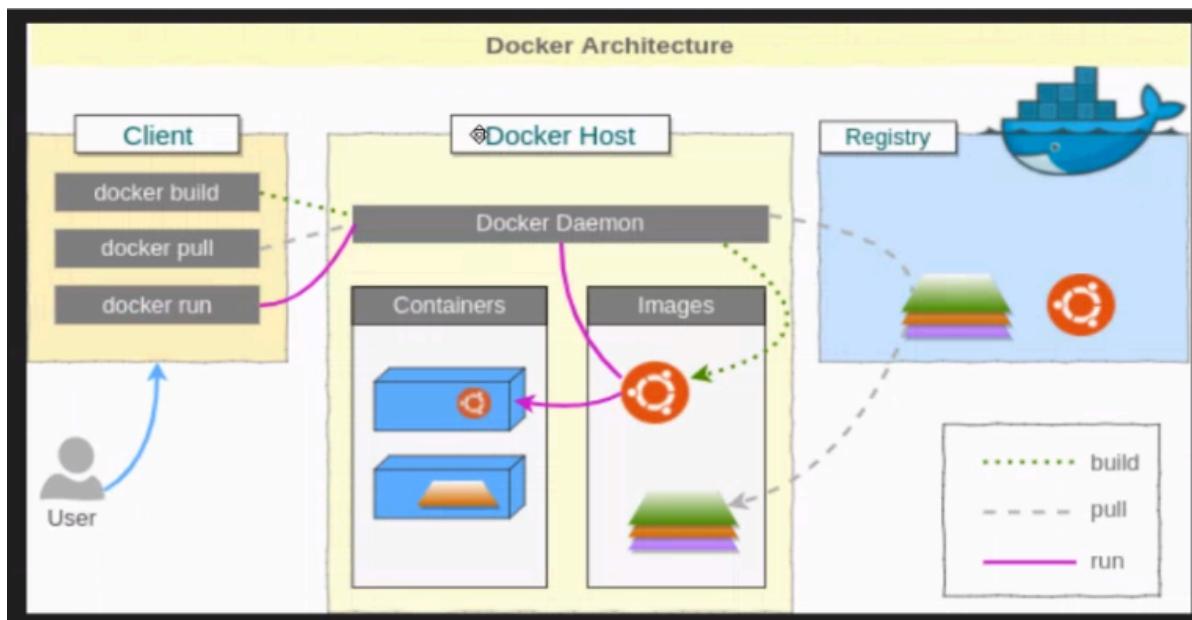
En la img de la derecha, nosotros tenemos nuestra computadora real, y tenemos a docker que gestiona las apps y las librerías que esa app necesita.

La app con la librería es un contenedor, que va a estar usando el mismo kernel que el SO base.

El kernel de mi SO se comparte con los contenedores que van corriendo.

Docker desktop puede levantar una VM con linux y te permite correr una app de linux.

Arq de Docker



Tengo un cliente que tiene un usuario con ciertos privilegios.

El "demonio" de Docker, a menudo referido simplemente como "Docker daemon" en inglés, es un componente fundamental de la arquitectura de Docker. Es un servicio que se ejecuta

en segundo plano en el sistema operativo y es responsable de administrar los contenedores de Docker.

Los comandos de la izq de todo en la imagen sirven para actualizar la “imagen” de docker, correrla, etc.

Aca puedo buscar imagenes de docker, por ejemplo, bases de datos:

<https://hub.docker.com/>

Cheatset de docker:

Cheatsheet for Docker CLI			
Run a new Container	Manage Containers	Manage Images	Info & Stats
<pre>Start a new Container from an Image docker run IMAGE docker run nginx ...and assign it a name docker run --name CONTAINER IMAGE docker run --name web nginx ...and map a port docker run -p HOSTPORT:CONTAINERPORT IMAGE docker run -p 8000:80 nginx ...and map all ports docker run -P IMAGE docker run -P nginx ...and start container in background docker run -d IMAGE docker run -d nginx ...and assign it a hostname docker run --hostname HOSTNAME IMAGE docker run --hostname srv nginx ...and add a dns entry docker run --add-host HOSTNAME:IP IMAGE ...and map a local directory into the container docker run -v HOSTDIR:TARGETDIR IMAGE docker run -v ~/usr/share/nginx/html nginx ...but change the entrypoint docker run -it --entrypoint EXECUTABLE IMAGE docker run -it --entrypoint bash nginx</pre>	<pre>Show a list of running containers docker ps Show a list of all containers docker ps -a Delete a container docker rm CONTAINER docker rm web Delete a running container docker rm -f CONTAINER docker rm -f web Delete stopped containers docker container prune Stop a running container docker stop CONTAINER docker stop web Start a stopped container docker start CONTAINER docker start web Copy a file from a container to the host docker cp CONTAINER:SOURCE TARGET docker cp web:/index.html index.html Copy a file from the host to a container docker cp TARGET CONTAINER:SOURCE docker cp index.html web:/index.html Start a shell inside a running container docker exec -it CONTAINER EXECUTABLE docker exec -it web bash Rename a container docker rename OLD_NAME NEW_NAME docker rename 096 web Create an image out of container docker commit CONTAINER docker commit web</pre>	<pre>Download an image docker pull IMAGE[:TAG] docker pull nginx Upload an image to a repository docker push IMAGE docker push myimage:1.0 Delete an image docker rmi IMAGE Show a list of all images docker images Delete dangling images docker image prune Delete all unused images docker image prune -a Build an image from a Dockerfile docker build DIRECTORY docker build . Tag an image docker tag IMAGE NEWIMAGE docker tag ubuntu ubuntu:10.04 Build and tag an image from a Dockerfile docker build -t IMAGE DIRECTORY docker build -t myimage . Save an image to tar file docker save IMAGE > FILE docker save nginx > nginx.tar Load an image from a tar file docker load -i TARFILE docker load -i nginx.tar</pre>	<pre>Show the logs of a container docker logs CONTAINER docker logs web Show stats of running containers docker stats Show processes of container docker top CONTAINER docker top web Show installed docker version docker version Get detailed info about an object docker inspect NAME docker inspect nginx Show all modified files in container docker diff CONTAINER docker diff web Show mapped ports of a container docker port CONTAINER docker port web</pre>

Docker-CheatSheet

Ejemplo de un dockerfile

Glosario Docker: Dockerfile

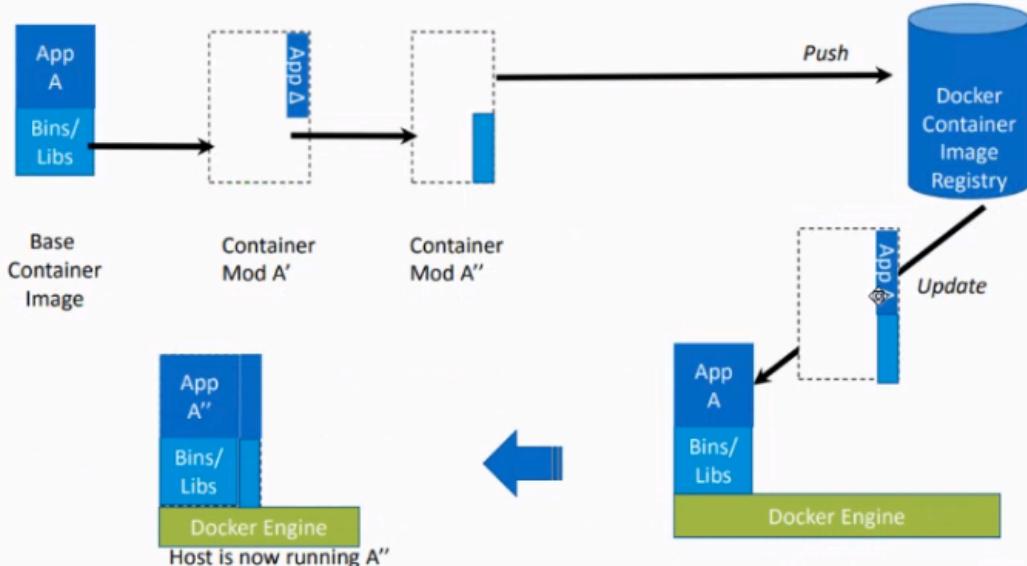
```
# A basic apache server.  
  
FROM ubuntu:14.04  
  
MAINTAINER Kimbro Staken version: 0.1  
  
RUN apt-get update && apt-get install -y apache2  
  
ENV APACHE_LOG_DIR /var/log/apache2  
  
EXPOSE 80  
  
CMD ["/usr/sbin/apache2", "-D", "FOREGROUND"]
```



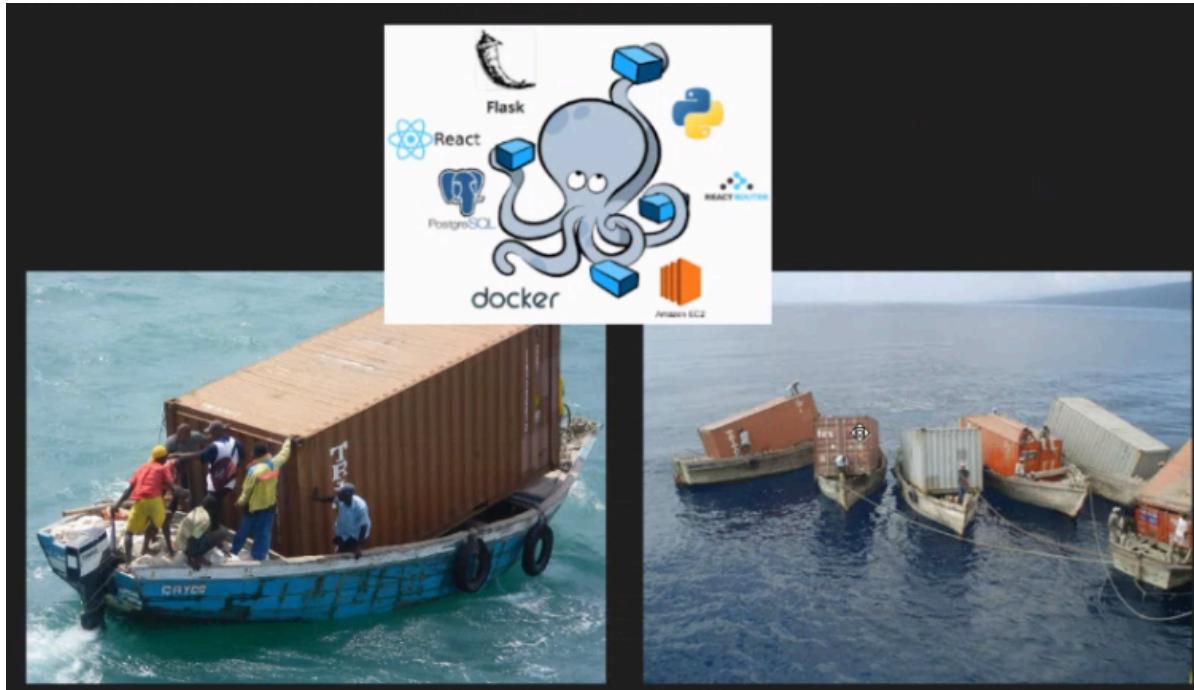
Esas letras es un script que se va ejecutando con las configuraciones de docker. Son los mismos comandos que venimos usando hasta ahora.

Como vemos en el esquema, una **registry** de docker es similar al comportamiento de git. Es decir, nosotros le podemos ir pusheando cosas a esa imagen. No vamos a pushear toda la imagen entera nuevamente, solo va a ir pusheando los cambios que hagamos.

Cambios y actualizaciones



Docker corriendo con distintas apps



Docker compose

```
version: '2'
services:
  web:
    build: .
    depends_on:
      - db
      - redis
  redis:
    image: redis
  db:
    image: postgres
```

```
graph LR
    web[web] --> redis[redis]
    web --> db[db]
```

Reference: <https://docs.docker.com/compose/compose-file>

Defino un servicio web con ciertas dependencias

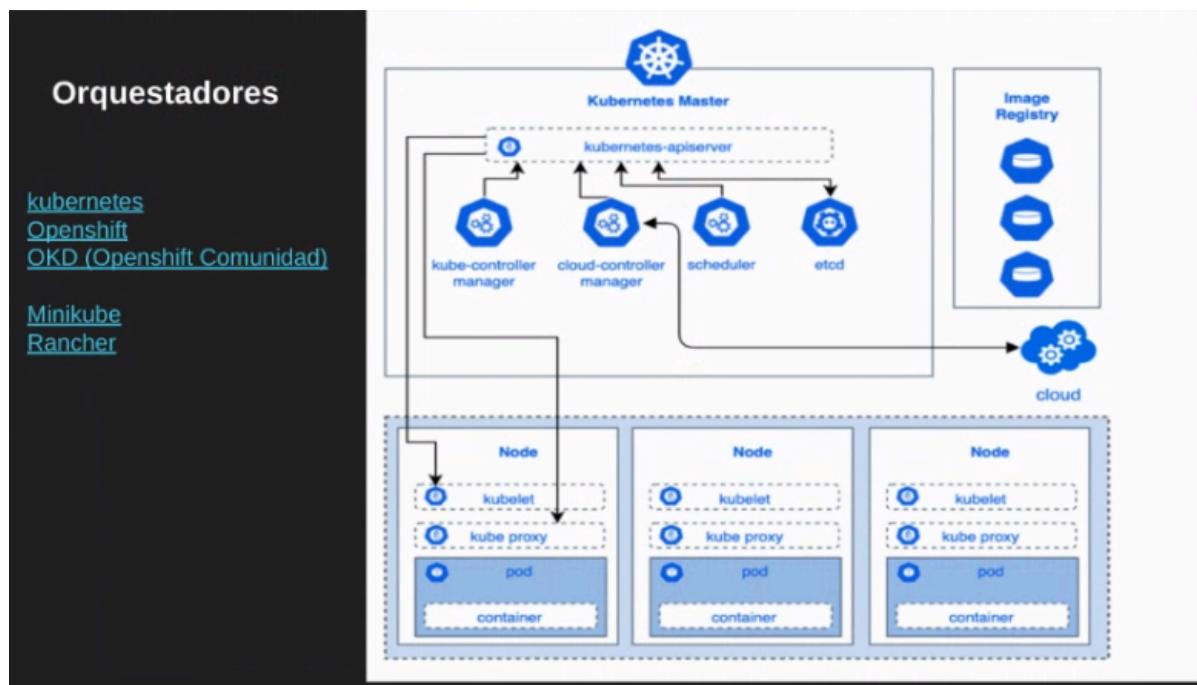
Y también defino esas dependencias en el mismo yml de configuración.

En caso de que se caiga algo, podemos hacer alguna config para que se levante automáticamente. Cosa de no tener que levantar uno por uno, incluso cuando apago y prendo la pc (o el server)

En casos de empresas grandes, donde hay muchas imágenes, ya no vamos a usar docker compose.

En esos casos vamos a usar kubernetes.

Kubernetes es un orquestador de contenedores.



Esto es lo mismo, pero ahora tenemos un cluster de servidores donde vamos a tener algunos que tengan la función de servidores **master**, que hablan con otros llamados "**nodos**". En los nodos se ejecutan las instancias de los contenedores.

Además de contenedores, se pueden llamar "pod". Pueden ser uno o más contenedores. En buenas prácticas, cada pod debería ser un servidor virtual.

Si se cae un pod de un nodo, se puede ir a otro nodo, entonces no tenemos que ir manualmente levantando cosas caídas. No dejaríamos de brindar servicios.

Lo que está a la derecha de la imagen son posibles orquestadores. Una empresa puede contratar servidores virtuales, e instalarle alguno de estos orquestadores con servicios de la empresa.

Configuración inicial (8:26 hs)

Link al repo de docker

https://github.com/upszot/UTN-FRA_SO_Docker

Ver el status de docker en mi máquina:

sudo systemctl status docker

```
nano@nahuel-ubuntu:~/UTN-FRA_SO_Docker$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
     Active: active (running) since Mon 2024-06-10 19:06:59 -03; 46min ago
TriggeredBy: ● docker.socket
   Docs: https://docs.docker.com
      Main PID: 1084 (dockerd)
        Tasks: 11
       Memory: 106.3M
          CPU: 806ms
         CGroup: /system.slice/docker.service
                   └─1084 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

jun 10 19:06:59 nahuel-ubuntu dockerd[1084]: time="2024-06-10T19:06:59.031871115-03:00" level>
jun 10 19:06:59 nahuel-ubuntu dockerd[1084]: time="2024-06-10T19:06:59.037243996-03:00" level>
jun 10 19:06:59 nahuel-ubuntu dockerd[1084]: time="2024-06-10T19:06:59.198769600-03:00" level>
jun 10 19:06:59 nahuel-ubuntu dockerd[1084]: time="2024-06-10T19:06:59.207822351-03:00" level>
jun 10 19:06:59 nahuel-ubuntu dockerd[1084]: time="2024-06-10T19:06:59.581185510-03:00" level>
jun 10 19:06:59 nahuel-ubuntu dockerd[1084]: time="2024-06-10T19:06:59.634178160-03:00" level>
jun 10 19:06:59 nahuel-ubuntu dockerd[1084]: time="2024-06-10T19:06:59.667304511-03:00" level>
jun 10 19:06:59 nahuel-ubuntu dockerd[1084]: time="2024-06-10T19:06:59.667794801-03:00" level>
jun 10 19:06:59 nahuel-ubuntu dockerd[1084]: time="2024-06-10T19:06:59.711959553-03:00" level>
jun 10 19:06:59 nahuel-ubuntu systemd[1]: Started Docker Application Container Engine.
```

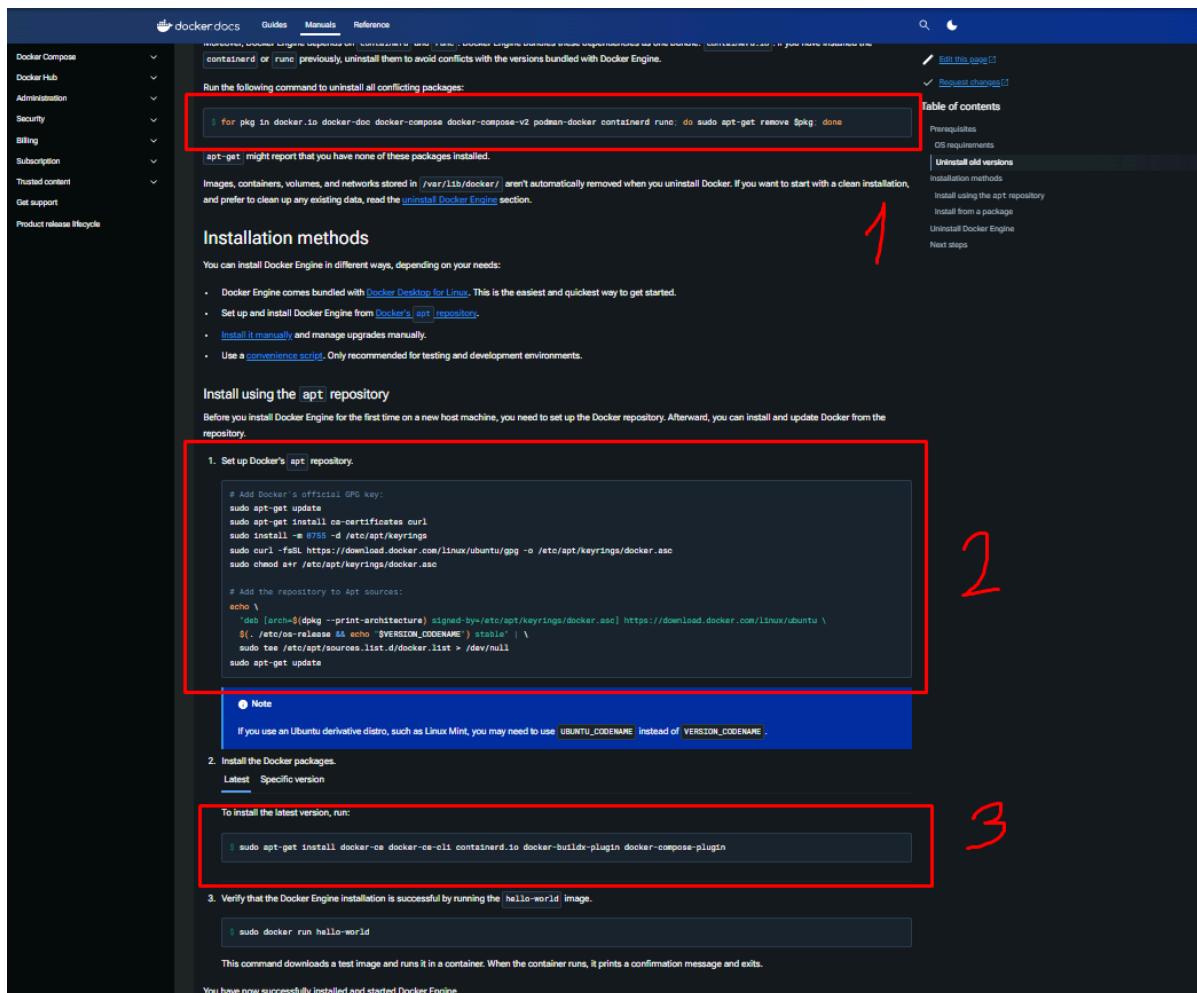
En la parte de Loaded (2da linea) dice que el service está enabled.

En caso de que no esté enabled, podemos tirar este comando:

sudo systemctl enable --now docker

Web para instalar el engine de docker en mi ubuntu:

<https://docs.docker.com/engine/install/ubuntu/>



getenforce
sudo setenforce 0

En caso de tener ubuntu, esto no es necesario. Ya que ubuntu no tiene selinux.

```
DMateo05@fedora:~/Escritorio$ id
uid=1000(DMateo05) gid=1000(DMateo05) grupos=1000(DMateo05),10
DMateo05@fedora:~/Escritorio$ sudo usermod -a -G docker $(whoami)
DMateo05@fedora:~/Escritorio$ id
uid=1000(DMateo05) gid=1000(DMateo05) grupos=1000(DMateo05),10
DMateo05@fedora:~/Escritorio$ getenforce
Enforcing
DMateo05@fedora:~/Escritorio$ sudo setenforce 0
DMateo05@fedora:~/Escritorio$ getenforce
Permissive
DMateo05@fedora:~/Escritorio$ sudo vim /etc/
```

```
vim /etc/selinux/config
```

el comando **getenforce** debe devolver Permissive

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/using_selinux/changing-selinux-states-and-modes_using-selinux

Parte práctica (8:26 hs)

Nos metemos dentro del ejemplo 0, del repo que clonamos
/UTN-FRA_SO_Docker/ejemplo0

```
nano@nahuel-ubuntu:~/UTN-FRA_SO_Docker/ejemplo0$ ll
total 16
drwxrwxr-x 2 nano nano 4096 jun 10 19:50 .
drwxrwxr-x 9 nano nano 4096 jun 10 19:50 ../
-rwxrwxr-x 1 nano nano 98 jun 10 19:50 01_run.sh*
-rwxrwxr-x 1 nano nano 431 jun 10 19:50 02_metodos_protocolo-HTTP.sh*
nano@nahuel-ubuntu:~/UTN-FRA_SO_Docker/ejemplo0$ cat 01_run.sh
#!/bin/bash

# Corremos el contenedor en background
docker run -d -p 80:80 kennethreitz/httpbin
```

Tiramos:

```
docker run -d -p 80:80 kennethreitz/httpbin
```

-d para que se corra en 2do plano

-p para indicar los puertos

Ver si tenemos algo corriendo (alguna imagen de docker):

```
docker ps
```

```
nano@nahuel-ubuntu:~/UTN-FRA_SO_Docker/ejemplo0$ docker run -d -p 80:80 kennethreitz/httpbin
Unable to find image 'kennethreitz/httpbin:latest' locally
latest: Pulling from kennethreitz/httpbin
473ede7ed136: Pull complete
c46b5fa4ad940: Pull complete
93ae3df89c92: Pull complete
6b1eed27cade: Pull complete
0373952b589d: Pull complete
7b82cd0ee527: Pull complete
a36b2d884a89: Pull complete
Digest: sha256:599fe5e5073102dbb0ee3dbb65f049dab44fa9fc251f6835c9990f8fb196a72b
Status: Downloaded newer image for kennethreitz/httpbin:latest
2f4bb4d9d617b6bb506aa63b33a3375caa30d1bce42ae7a38508cdfee6c73c9
nano@nahuel-ubuntu:~/UTN-FRA_SO_Docker/ejemplo0$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
2f4bb4d9d617 kennethreitz/httpbin "gunicorn -b 0.0.0.0..." 28 seconds ago Up 27 seconds 0.0.0.0:80->80/tcp, :::80->80/tcp great_ride
```

Ahí abajo vemos que tenemos corriendo esa imagen que acabamos de darle run, en el puerto 80

Comando **ip a**

Nos sirve para mostrar las interfaces de red

Otro que también nos sirve para averiguar nuestra ip es **ifconfig**

El comando ip es más poderoso

Mi ip es esta: **10.0.2.15**

Y si la abro en el navegador, me debería abrir la web que se levantó en local.

http://10.0.2.15:80

Ejemplo 1:

En este ejemplo NO GENERAMOS una imagen. Simplemente usamos docker para levantar algo que tenemos en nuestro entorno local.

```
nano@nahuel-ubuntu:~/UTN-FRA_SO_Docker/ejemplo1$ ll
total 16
drwxrwxr-x 3 nano nano 4096 jun 21 11:53 .
drwxrwxr-x 9 nano nano 4096 jun 10 19:50 ../
-rwxrwxr-x 1 nano nano 90 jun 10 19:50 run.sh*
drwxrwxr-x 3 nano nano 4096 jun 10 19:50 share/
nano@nahuel-ubuntu:~/UTN-FRA_SO_Docker/ejemplo1$ tree share/
share/
└── html
    └── index.html

1 directory, 1 file
nano@nahuel-ubuntu:~/UTN-FRA_SO_Docker/ejemplo1$ cat run.sh
#!/bin/bash

docker run -d -p 80:80 -v "$PWD"/share/html:/usr/share/nginx/html nginx
```

Tenemos ese run que levanta la web (index.html) que está dentro de la carpeta share

Vamos a tirar este comando:

```
docker run -d -p 8081:80 -v "$PWD"/share/html:/usr/share/nginx/html nginx
```

8081: es el puerto de la izquierda, representa a mi puerto local

80: el de la derecha, representa el puerto del contenedor de docker

-v "\$PWD"/share/html:/usr/share/nginx/html => significa que vamos a montar el html pintado de amarillo, en el directorio pintado de verde que está DENTRO del docker.

Izq es mi computadora, y lo verde es dentro de docker.

nginx: Nombre de la imagen del contenedor que queremos levantar

```
nano@nahuel-ubuntu:~/UTN-FRA_SO_Docker/ejemplo1$ docker run -d -p 8081:80 -v "$PWD"/share/html:/usr/share/nginx/html nginx
Home
```

Puerto local de mi compu Puerto del contenedor

Ahora, si tiro un docker ps, tengo 2 imagenes corriendo

```
nano@nahuel-ubuntu:~/UTN-FRA_SO_Docker/ejemplo1$ docker run -d -p 8081:80 -v "$PWD"/share/html:/usr/share/nginx/html nginx
8829ed57174045cba3c48fc3a3658a10b0/Zb98C7455f00771defbd3ab975e7
nano@nahuel-ubuntu:~/UTN-FRA_SO_Docker/ejemplo1$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
8829ed571740 nginx "/docker-entrypoint..." About a minute ago Up About a minute 0.0.0.0:8081->80/tcp, :::8081->80/tcp agitated_sammet
2f4bb4d9d617 kennethreitz/httpbin "gunicorn -b 0.0.0.0..." 27 minutes ago Up 27 minutes 0.0.0.0:80->80/tcp, :::80->80/tcp great_ride
```

Si yo edito el directorio local donde tengo mi web, se modifica, ya que eso es lo que estoy levantando. Nunca generé una imagen.

La imagen se genera en el ejemplo siguiente

Ejemplo 2:

Este ejemplo usa ese dockerfile para armar una imagen con una web, y luego usamos esa imagen creada para levantar la web.

El código de esa web se encuentra dentro de la carpeta static-html-directory/

```
nano@nahuel-ubuntu:~/UTN-FRA_SO_Docker/ejemplo2$ ll
total 24
drwxrwxr-x 3 nano nano 4096 jun 10 19:50 .
drwxrwxr-x 9 nano nano 4096 jun 10 19:50 ../
-rwxrwxr-x 1 nano nano 80 jun 10 19:50 01_make-build.sh*
-rwxrwxr-x 1 nano nano 162 jun 10 19:50 02_run.sh*
-rw-rw-r-- 1 nano nano 61 jun 10 19:50 dockerfile
drwxrwxr-x 2 nano nano 4096 jun 10 19:50 static-html-directory/
nano@nahuel-ubuntu:~/UTN-FRA_SO_Docker/ejemplo2$ cat 02_run.sh
#!/bin/bash
#Corro la imagen generada anteriormente
docker run -d -p 8080:80 some-content-nginx

echo This site can't be reached
echo "Entre en: http://localhost:8080"
echo 10.0.2.15 refused to connect.
docker ps
```

Generar una imagen de docker:

En este ejemplo, tenemos el script make-build con el siguiente contenido:
docker build -t some-content-nginx .

-t: es para pasarle el nombre. En este caso, el nombre de la imagen será “some-content-nginx”
el punto del final hace referencia al directorio en el que estoy parado.

Vamos a tener que ejecutar ese script, o tirar ese comando para generar la imagen

```
nano@nahuel-ubuntu:~/UTN-FRA_S0_Docker/ejemplo2$ cat 01_make-build.sh
#!/bin/bash

#Haciendo build de la imagen
docker build -t some-content-nginx .
```

Como yo estoy parado en un directorio que tiene un archivo dockerfile, el make build va a usar eso para armar la imagen.

```
nano@nahuel-ubuntu:~/UTN-FRA_S0_Docker/ejemplo2$ cat dockerfile
FROM nginx
COPY static-html-directory /usr/share/nginx/html
```

static-html-directory es la carpeta que tenemos en local, dentro del directorio ejemplo2

```
nanonahuel-ubuntu:~/UTN-FRA_S0_Docker/ejemplo2$ docker build -t some-content-nginx . → ejecuto el comando para generar la imagen
[+] Building 0.2s (7/7) FINISHED
=> [internal] load build definition from dockerfile
=> => transferring dockerfile: 98B
=> [internal] load metadata for docker.io/library/nginx:latest
=> [internal] load .dockerrcignore
=> => transferring context: 2B
=> [internal] load build context
=> => transferring context: 131B
=> [1/2] FROM docker.io/library/nginx:latest
=> [2/2] COPY static-html-directory /usr/share/nginx/html
=> exporting to image
=> => exporting layers
=> => writing image sha256:cc470f7176ddc73e65fc25aa2bc84503c48228c887dabfd9d2a4e55217a2c54
=> => naming to docker.io/library/some-content-nginx
nanonahuel-ubuntu:~/UTN-FRA_S0_Docker/ejemplo2$ docker image ls → imagen generada con el comando de arriba
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
some-content-nginx  latest   cc470f7176dd  6 seconds ago  188MB
nginx               latest   4f67c83422ec  3 weeks ago   188MB
hello-world         latest   d2c94e258dc9  13 months ago  13.3kB
kennethreitz/httpbin  latest   b138b9264903  5 years ago   534MB
```

Puedo usar el comando **docker image ls** para ver qué imágenes generé

Nosotros sólo generamos la imagen, no la subimos a ningún lado. Y tampoco está corriendo en ningún lado.

Para el parcial vamos a tener que subir una imagen a **docker-hub**.

Ejecutar una imagen/contenedor de docker:

Ahora vamos a ejecutar una imagen ya existente.

```
docker run -d -p 8080:80 some-content-nginx
```

Ese comando lo podemos tirar desde cualquier lado, ya que estamos llamando al NOMBRE de la imagen. No necesito pararme en cierto directorio específico.

Ahora si tiro un docker ps, veo que tengo 3 cosas corriendo

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
ead5b5d650e8	some-content-nginx	"/docker-entrypoint..."	About a minute ago	Up About a minute	0.0.0.0:8080->80/tcp, :::8080->80/tcp	distracted_shockley
8829ed571740	nginx	"/docker-entrypoint..."	17 minutes ago	Up 17 minutes	0.0.0.0:8081->80/tcp, :::8081->80/tcp	agitated_sammet
2f4bb4d9d617	kennethreitz/httpbin	"unicorn -b 0.0.0.0..."	43 minutes ago	Up 43 minutes	0.0.0.0:80->80/tcp, :::80->80/tcp	great_ride

Si yo quiero modificar lo que tengo levantado, tengo que GENERAR OTRA IMAGEN NUEVA con el código modificado, y volver a subir esa nueva imagen.

Modificar una imagen ya existente y buildear:

Primero modifco lo que quiera subir.

```
nano@nahuel-ubuntu:~/UTN-FRA_SO_Docker/ejemplo2$ echo "nueva linea agregada build 2" >> static-html-directory/index.html
nano@nahuel-ubuntu:~/UTN-FRA_SO_Docker/ejemplo2$ cat static-html-directory/index.html
hola mundo - dockerfile 01
nueva linea agregada build 2
```

Ahora tiro:

```
docker build -t some-content-nginx:v1 .
```

(el punto final es parte del comando, hace referencia al directorio en el que estoy parado)

Al nombre de la imagen “some-content-nginx” le agregué un :v1
ese :v1 representa el tag de la imagen

```
nano@nahuel-ubuntu:~/UTN-FRA_SO_Docker/ejemplo2$ docker build -t some-content-nginx:v1 .
[+] Building 0.1s (7/7) FINISHED
=> [internal] load build definition from dockerfile
=> => transferring dockerfile: 98B
=> [internal] load metadata for docker.io/library/nginx:latest
=> [internal] load .dockerrignore
=> => transferring context: 2B
=> [internal] load build context
=> => transferring context: 160B
=> CACHED [1/2] FROM docker.io/library/nginx:latest
=> [2/2] COPY static-html-directory /usr/share/nginx/html
=> exporting to image
=> => exporting layers
=> => writing image sha256:1b173c3d9d559c4256433e465721daf6e1b1f0028c539438de64b19ce8194960
=> => naming to docker.io/library/some-content-nginx:v1
```

Y eso me generó una imagen nueva, con su respectivo tag:

```
nano@nahuel-ubuntu:~/UTN-FRA_SO_Docker/ejemplo2$ docker image ls
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
some-content-nginx  v1      1b173c3d9d55  48 seconds ago  188MB
some-content-nginx  latest   cc470f7176dd  18 minutes ago  188MB
nginx               latest   4f67c83422ec  3 weeks ago   188MB
hello-world         latest   d2c94e258dc9  13 months ago  13.3kB
kennethreitz/httpbin  latest   b138b9264903  5 years ago   534MB
```

Nueva imagen con su tag v1

Ahora cuando lo quiera levantar, voy a tener que tirar **docker run -d -p 8080:80 some-content-nginx:v1**

```
nano@nahuel-ubuntu:~/UTN-FRA_SO_Docker/ejemplo2$ docker run -d -p 8080:80 some-content-nginx:v1 → Corro la imagen nueva, pero va a fallar
23e33e0983299b2317558f102082e7d1990dcbb42dfd777d85effb5192a453e
docker: Error response from daemon: driver failed programming external connectivity on endpoint reverent_dirac (012fdef06e17a59ec8a3ab1c7e4f1c
01626315d88ee4b3a9d8397e4e43a753a8): Bind for 0.0.0.0:8080 failed: port is already allocated.
```

Está bien que falle, pq en el puerto 8080 ya tengo otra cosa levantada. Tengo una imagen vieja

Esto lo puedo ver tirando un **docker ps**

```
nano@nahuel-ubuntu:~/UTN-FRA_SO_Docker/ejemplo2$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
ead5b5d650e8 some-content-nginx "/docker-entrypoint..." 17 minutes ago Up 17 minutes 0.0.0.0:8080->80/tcp, :::8080->80/tcp, 0.0.0.0:8081->80/tcp, :::8081->80/tcp, 0.0.0.0:80->80/tcp, ::::80->80/tcp quédó levantada la imagen vieja hace 17 minutos
8829ed571740 nginx "/docker-entrypoint..." 33 minutes ago Up 33 minutes agitated_sammet
2f4bb4d9d617 kennethreitz/httpbin "gunicorn -b 0.0.0.0..." About an hour ago Up About an hour great_ride
```

Ahora entonces tengo que matar esa imagen vieja que está corriendo

docker stop containerId

```
nano@nahuel-ubuntu:~/UTN-FRA_SO_Docker/ejemplo2$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
ead5b5d650e8 some-content-nginx "/docker-entrypoint..." 17 minutes ago Up 17 minutes 0.0.0.0:8080->80/tcp, :::8080->80/tcp, 0.0.0.0:8081->80/tcp, :::8081->80/tcp, 0.0.0.0:80->80/tcp, ::::80->80/tcp distracted_shockley
8829ed571740 nginx "/docker-entrypoint..." 33 minutes ago Up 33 minutes agitated_sammet
2f4bb4d9d617 kennethreitz/httpbin "gunicorn -b 0.0.0.0..." About an hour ago Up About an hour great_ride
nano@nahuel-ubuntu:~/UTN-FRA_SO_Docker/ejemplo2$ docker stop ead5b5d650e8^C
nano@nahuel-ubuntu:~/UTN-FRA_SO_Docker/ejemplo2$ docker stop ead5b5d650e8
ead5b5d650e8
```

Y me retorna el id del container que maté.

Y ahora si, puedo correr la nueva imagen en el mismo puerto:

docker run -d -p 8080:80 some-content-nginx:v1

```
nano@nahuel-ubuntu:~/UTN-FRA_SO_Docker/ejemplo2$ docker run -d -p 8080:80 some-content-nginx:v1 → Ahora si corro la imagen nueva
5dc9a7455ce9fdb9db46b390331f34f5668f23552b86fea1f30598a7fa684da
nano@nahuel-ubuntu:~/UTN-FRA_SO_Docker/ejemplo2$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
5dc9a7455ce9 some-content-nginx:v1 "/docker-entrypoint..." 5 seconds ago Up 5 seconds 0.0.0.0:8080->80/tcp, :::8080->80/tcp infallible_edison
8829ed571740 nginx "/docker-entrypoint..." 2 hours ago Up 2 hours 0.0.0.0:8081->80/tcp, :::8081->80/tcp agitated_sammet
2f4bb4d9d617 kennethreitz/httpbin "gunicorn -b 0.0.0.0..." 2 hours ago Up 2 hours 0.0.0.0:80->80/tcp, ::::80->80/tcp great_ride
```

Ejemplo 3: (22:18hs)

En este ejemplo vamos a ver cómo levantar más de una “cosa”, usando docker compose.
(creo que esto no entra en el parcial).

En el parcial vamos a tener que hacer algo igual a lo que se hace en el ejercicio 2

```

nano@nahuel-ubuntu:~/UTN-FRA_SO_Docker/ejemplo3$ ll
total 24
drwxrwxr-x 3 nano nano 4096 jun 20 12:39 .
drwxrwxr-x 9 nano nano 4096 jun 10 19:50 ..
drwxrwxr-x 2 nano nano 4096 jun 10 22:39 code/
-rw-rw-r-- 1 nano nano 419 jun 20 12:39 docker-compose.yml
-rw-rw-r-- 1 nano nano 571 jun 10 19:50 README.md
-rw-rw-r-- 1 nano nano 517 jun 10 22:28 site.conf
nano@nahuel-ubuntu:~/UTN-FRA_SO_Docker/ejemplo3$ cat docker-compose.yml
version: '2'

services:
  web:
    image: nginx:latest
    ports:
      - "8080:80"
    volumes:
      - ./code:/code
      - ./site.conf:/etc/nginx/conf.d/default.conf
    networks:
      - code-network
  php:
    image: php:fpm
    volumes:
      - ./code:/code
    networks:
      - code-network

networks:
  code-network:
    driver: bridge

```

Estructura del ejemplo 3

Docker compose es para levantar varias cosas a la vez

Levanta una web y un php a la vez

El primero, la web, va a levantar la imagen de nginx

El otro, php, levanta una imagen de php.

Eso lo hace ya que así está configurado el docker-compose.yml

Para ejecutar esto, tengo que correr

docker compose up -d

```

nano@nahuel-ubuntu:~/UTN-FRA_SO_Docker/ejemplo3$ docker compose up -d
[WARN] [0000] /home/nano/UTN-FRA_SO_Docker/ejemplo3/docker-compose.yml: `version` is obsolete
[*] Running 1/1
  ✓ php Pulled
    ✓ 2cc3ae149d28 Pull complete
    ✓ 01c187ab622c Pull complete
    ✓ 4382a8029fff Pull complete
    ✓ 43046b340e34 Pull complete
    ✓ 41722365abab Pull complete
    ✓ a52941633aa9 Pull complete
    ✓ 930f8db3b95e Pull complete
    ✓ f32aedd4fa72d Pull complete
    ✓ 499ff39c692f7 Pull complete
    ✓ add8a6605e0d Pull complete
[*] Running 3/3
  ✓ Network ejemplo3_code-network Created
  ✓ Container ejemplo3-php-1 Started
  ✓ Container ejemplo3-web-1 Started
nano@nahuel-ubuntu:~/UTN-FRA_SO_Docker/ejemplo3$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
dd6b69c73a0b php:fpm "docker-php-entrypoi..." 3 seconds ago Up 2 seconds 9000/tcp ejemplo3-php-1
d83ec6315d74 nginx:latest "/docker-entrypoint..." 3 seconds ago Up 2 seconds 0.0.0.0:8080->80/tcp, :::8080->80/tcp exemplo3-web-1

```

Y vemos que levantó 2 imágenes con 1 sólo comando. La de la web y la de php.

Una vez que terminé, puedo tirar un **docker compose down** para matar todo lo que tengo levantado

Ver los logs de un contenedor:

docker logs containerId

Tenemos que reemplazar containerId por el id del contenedor, lo obtengo tirando un docker ps

Ver que tengo corriendo en los puertos

sudo netstat -putona

Active Internet connections (servers and established)						
Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State	PID/Program name
tcp	0	0	0.0.0.0:22	0.0.0.0:*	LISTEN	935/sshd: /usr/sbin off (0.00/0/0)
tcp	0	0	127.0.0.53:53	0.0.0.0:*	LISTEN	730/systemd-resolve off (0.00/0/0)
tcp	0	0	127.0.0.1:631	0.0.0.0:*	LISTEN	913/cupsd off (0.00/0/0)
tcp6	0	0	:::22	:::*	LISTEN	935/sshd: /usr/sbin off (0.00/0/0)
tcp6	0	0	:::631	:::*	LISTEN	913/cupsd off (0.00/0/0)
udp	0	0	0.0.0.0:631	0.0.0.0:*		980/cups-browsed off (0.00/0/0)
udp	0	0	0.0.0.0:5353	0.0.0.0:*		827/avahi-daemon: r off (0.00/0/0)
udp	0	0	0.0.0.0:36570	0.0.0.0:*		827/avahi-daemon: r off (0.00/0/0)
udp	0	0	127.0.0.53:53	0.0.0.0:*		730/systemd-resolve off (0.00/0/0)
udp	0	0	10.0.2.15:68	10.0.2.2:67	ESTABLISHED	832/NetworkManager off (0.00/0/0)
udp6	0	0	:::45499	:::*		827/avahi-daemon: r off (0.00/0/0)
udp6	0	0	:::5353	:::*		827/avahi-daemon: r off (0.00/0/0)

Parciales:

https://github.com/upszot/UTN-FRA_SO_Examenes/blob/master/202311/202311_2doParcial.pdf

https://github.com/upszot/UTN-FRA_SO_Examenes/blob/master/20231211/20231211_Recu2doParcial.pdf

https://github.com/upszot/UTN-FRA_SO_Examenes/blob/master/20231204/20231204_Recu2doParcial.pdf

https://github.com/upszot/UTN-FRA_SO_Examenes/blob/master/20231211/20231211_Recu2doParcial.pdf

Clase 10 (Repaso antes del parcial - Jueves 20/06)

Hicimos un modelo de parcial.

LVM.

PV -> VG -> LV

No puedo meter el mismo PV en 2 VG distintos. Ya que un PV es una partición.
Si nos pide 2 VG distintos, vamos a necesitar hacer 2 particiones diferentes, pq es un PV para cada VG.

Si queremos bajar una imagen para levantar un nginx, si tenemos un LV de poco espacio (por ejemplo 20MB), lo vamos a tener que ampliar, por más que en el parcial no lo diga.

No podemos crear el PV sobre el disco, tenemos que hacerlo sobre la partición, ya que la partición debe ser de type 8e (Linux LVM)

Si hay un LV que se llama “swap”, no tenemos que formatearlo como ext4, vamos a tener que formatearlo como memoria swap, usando el comando mkswap.
Recordemos que en este caso lo que se formatea es el LV, no la partición.

Antes de montar un LV en un directorio, tengo que crear ese directorio con un mkdir

Docker parte 1 (7:17hs)

Validamos que docker esté levantado: **sudo systemctl status docker**
Tiene que estar en enabled y en running.

Sino, **sudo systemctl enable –now docker**

Agregar al grupo docker mi usuario. (ya lo tengo hecho)

Lo que hicimos fue extender el LV para poder descargar la imagen de docker, pero no resolvimos todavía el punto de Docker

Bash scripting (7:50hs)

Vamos a necesitar un script que haga un bucle y vaya agregando usuarios.
Ponerle lógica para que no cree grupos si ya existen, etc.

El profe nos va a dar una lista, Así que necesitamos que el script reciba por params esa lista

Tirar echos indicando que se fue haciendo

grep desa /etc/group

Si ese comando me trae esa palabra “desa”, es pq el grupo existe. Entonces ahí no creo ese grupo.

Tengo que meter un if con eso.

Extension de VSCode => linter para los .sh

<https://marketplace.visualstudio.com/items?itemName=timonwong.shellcheck>

Docker parte 2 (8:12hs)

El profe nos pasa un file.

Le tenemos que cargar los valores a un yml

Eso hay que subirlo a docker-hub

<https://www.returngis.net/2019/02/publicar-tu-imagen-en-docker-hub/>

Obviamente me tengo que crear un usuario antes

```
docker tag nodejs-webapp 0gis0/nodejs-webapp:v1
```

Ese comando me lo puedo ahorrar si hago un docker build

Del repo que nos pasó el profe:

Ejemplo 0 - como corro una imagen

Ejemplo 1 - cómo corro una imagen montando un directorio

Ejemplo 2 - make build de la imagen y armar un dockerfile

Para el parcial voy a tener que buildear un tag local, puedo ver cómo está el ejemplo 2 y copiarlo

Y luego voy a tener que subir esa imagen al docker-hub, y ejecutarla en mi máquina local.

Finalmente le puedo hacer un curl para validar que esté todo ok

Ansible (8:33hs)

Parcial:

Docker

1. Crear el file dockerfile con minusculas.

2. Al dockerfile hay que ponerle este contenido:

```
FROM nginx:latest
COPY index.html /usr/share/nginx/html
```

3. buildearlo

```
docker build -t 2do-parcial-lalin-test UTN-FRA_SO_Examenes/202311/docker/
```

4. ver que exista la imagen generada

```
docker build -t 2do-parcial-lalin-test UTN-FRA_SO_Examenes/202311/docker/
```

5.

Script para correr

```
#!/bin/bash
docker run -p 8081:8081 -v
/home/nano/UTN-FRA_SO_Examenes/202311/docker/share:/mnt/share
nahuelglalin/test-modelo-examen
```

Ansible instalar programas

```
- name: "Instalar net-tools, htop y bpytop"
become: true
ansible.builtin.package:
  name: net-tools
  state: present
```

Máquina donde hice el modelo:

Ubuntu-23-2do-parcial-ultima-clase-v2