Ingeniería de Servidores

grado en ingeniería informática

Guión de prácticas resueltas

Autor
Carlos Sánchez Páez

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Curso 2019-2020

Índice

Índice de figuras

1. Práctica 1: Virtualización e instalación de Sistemas Operativos

1.1. Sesión 1

1.1.1. Tipos de arquitecturas

Un servidor es una máquina que se dedica a resolver peticiones. Hay varios tipos:

- **Hosting dedicado**. Alguien monta su propio servidor y únicamente lo utiliza él.
- **VPS (*Virtual Private Server*)**. Se utiliza la virtualización para proporcionar recursos dedicados (privados) al cliente a partir de un servidor con múltiples usuarios.
- ***Serverless***. Se necesita un proveedor cloud (*AWS, Azure, Google Cloud...*), que gestiona dinámicamente los recursos. Las aplicaciones se ejecutan al detectarse determinados eventos. Se cobra por ancho de banda utilizado, capacidad de disco duro, etc. La principal ventaja de esta arquitectura es la escalabilidad.

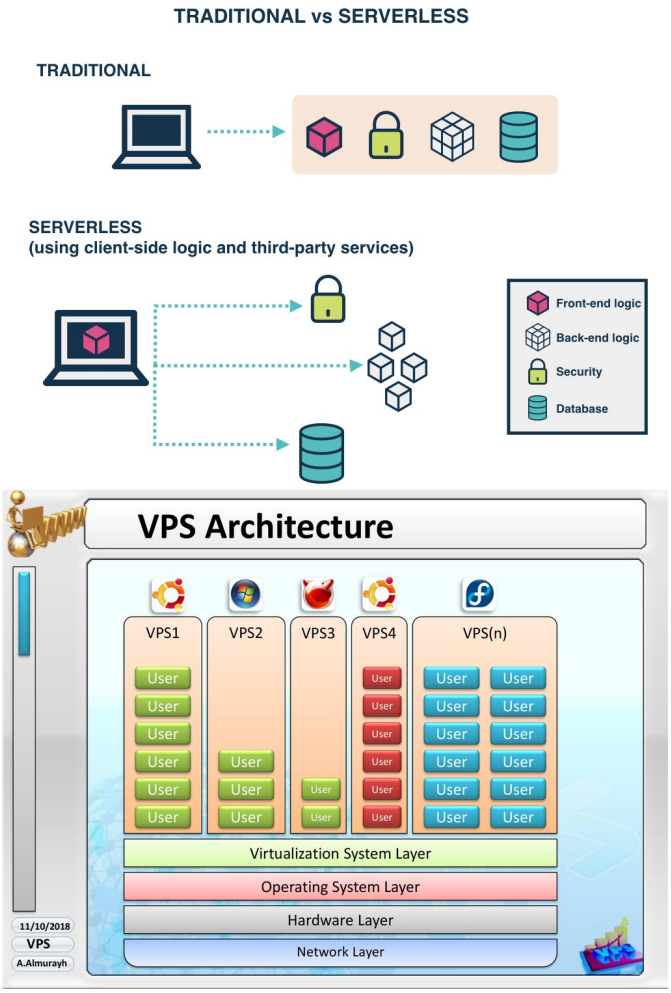


Figura 1: Tipos de arquitecturas de servidor

Una **máquina virtual** es aquella en la que todo su *hardware* está virtualizado, es decir, compartido con el host. Las principales ventajas de las MV son precio, encapsulamiento y flexibilidad. Realmente no son más que archivos.

Un **contenedor** empaqueta una aplicación con sus correspondientes dependencias, consiguiendo así la máxima *portabilidad*. Lo único que se necesita tener instalado en una máquina para ejecutar la aplicación es el hipervisor adecuado (por ejemplo, *Docker*).

Un **hipervisor** es un motor que se encarga de traducir las instrucciones de una máquina virtual (o contenedor) a llamadas al sistema. Algunos ejemplos de hipervisor son *VirtualBox* o *VMWare*.

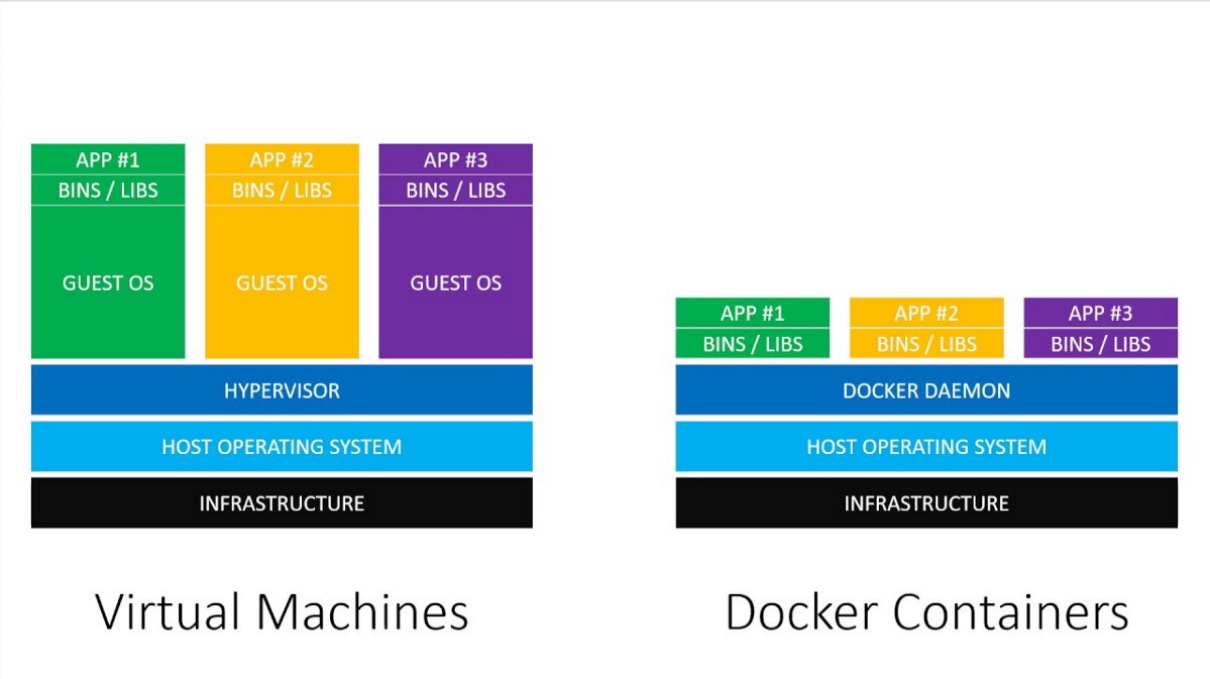


Figura 2: Diferencias entre contenedor y máquina virtual

Podríamos realizar una analogía diciendo que las máquinas virtuales son procesos (encapsulados, no pueden acceder entre ellos) y los contenedores son hebras (sí que pueden comunicarse).

1.1.2. RAID

RAID (***R**edundant **A**rray of **I**ndependent/**I**nexpensive* **D**isks) es una tecnología que utiliza varias unidades de almacenamiento entre las que se replican los datos. Existen varios tipos de RAID:

- **RAID0**. En este modelo no hay réplica. Los datos se distribuyen equitativamente entre ambos volúmenes.
- **RAID1**. Los datos se copian en el otro disco (espejo).
- **RAID2,3,4** no se usan en la actualidad.

- **RAID5**. Implementa bloques de paridad como medida de redundancia. Puede fallar un disco como máximo.
- **RAID6**. Implementa doble paridad. Pueden fallar dos discos como máximo.
- **RAID0+1,RAID1+0**. Se anidan ambos tipos de RAID.

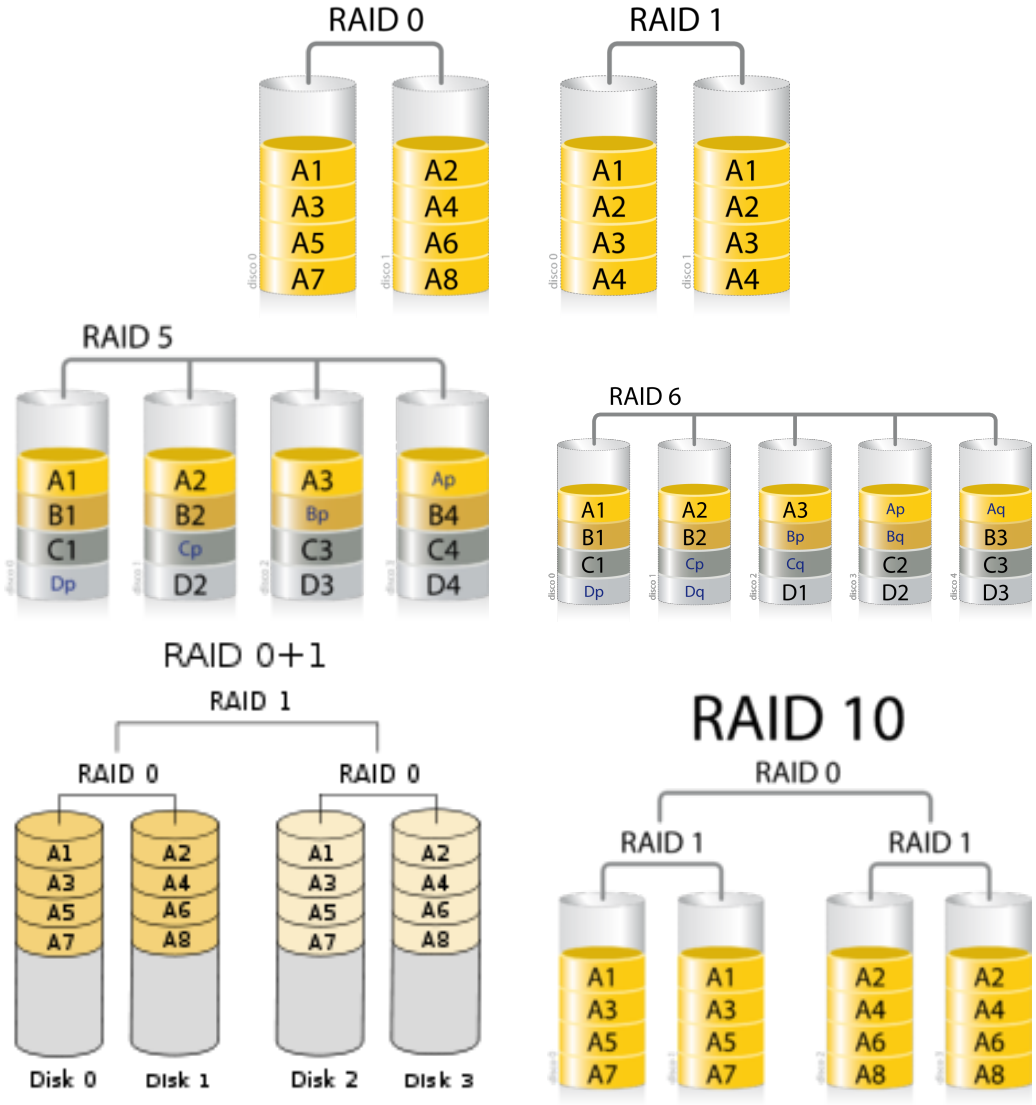


Figura 3: Tipos de RAID

1.1.3. LVM

LVM (***L**ogical **V**olume **M**anager*) provee abstracción sobre el almacenamiento físico y el sistema de ficheros. Sus mayor ventaja es la flexibilidad: podemos incorporar nuevas unidades de almacenamiento *en caliente* (sin parar el sistema) de forma sencilla. LVM está compuesto por:

- Volumen físico (*Physical Volume*, **PV**). Es un dispositivo de almacenamiento (HDD, partición, RAID...).
- Grupo de volúmenes (*Volume Group*, **VG**). Es el centro de LVM. Está formado por uno o más PV. Para aumentar su espacio sólo hay que añadir más volúmenes físicos, siendo esto transparente para los sistemas de archivos, procesos o usuarios.
- Volumen lógico (*Logical Volume*, **LV**). Es el "producto final", es decir, dispositivos que usaremos para crear sistemas de ficheros. Podríamos decir que son las particiones.

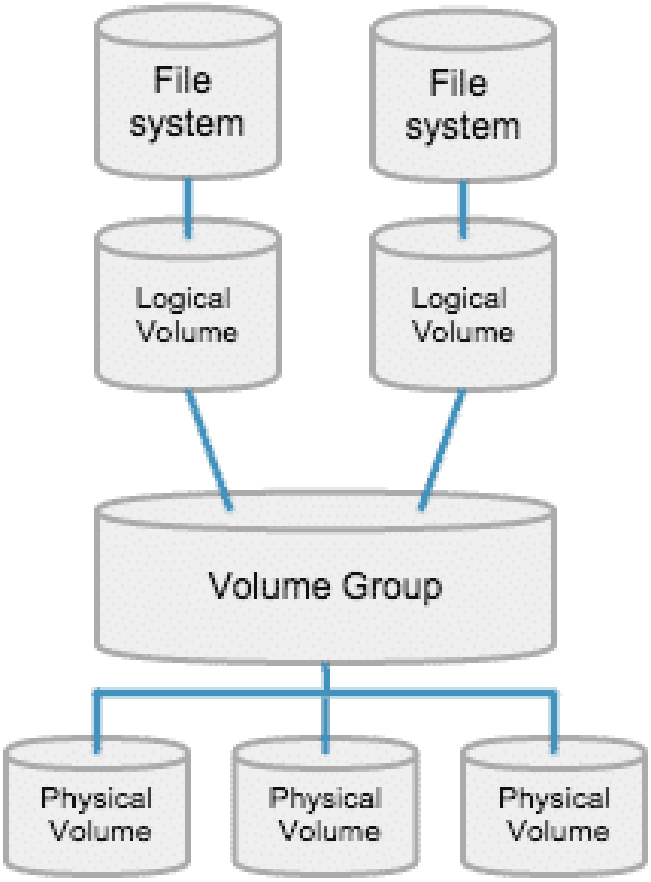
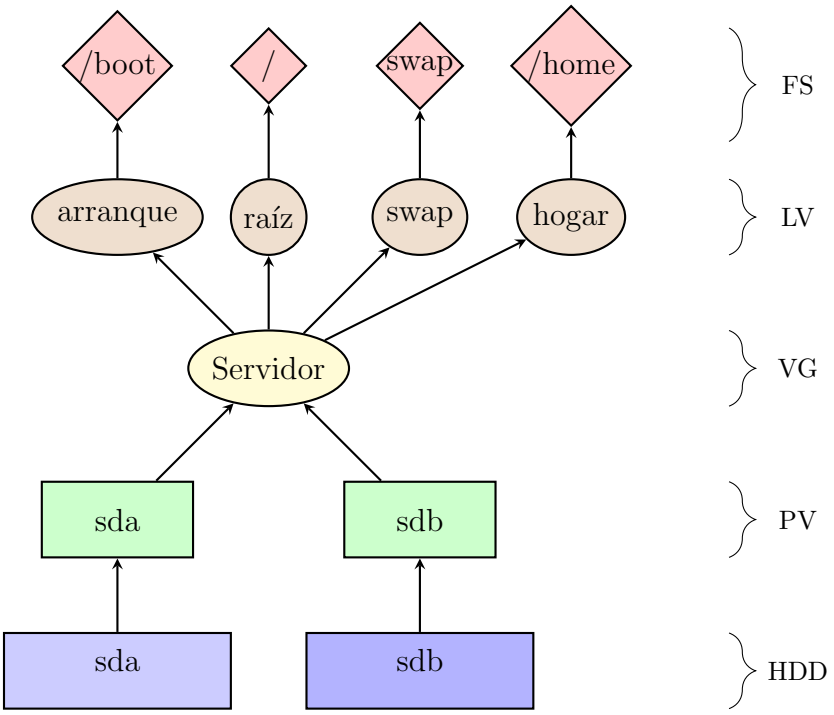


Figura 4: Arquitectura LVM

La estructura a crear es la siguiente:



1.1.4. Instalación de Ubuntu Server con RAID1

1. Descargamos la ISO desde aquí.
2. Creamos una máquina virtual con 1024MB RAM y dos discos duros VDI reservados dinámicamente de 10GB cada uno. Por último, montamos la ISO en el controlador IDE.
3. Arrancamos la máquina, marcamos idioma español y procedemos a la instalación.
4. Elegimos España y la distribución de teclado Spanish/Spanish.
5. Dejamos el nombre de la máquina en "ubuntu" y establecemos el nombre de usuario con nuestras iniciales. La clave será *practicas,ISE*.
6. No ciframos la carpeta personal, ya que usaremos FDE (*Full Disk Encryption*).
7. Aceptamos la zona horaria propuesta.
8. Elegimos particionado manual.
9. Damos Enter en ambos discos para crear las tablas de particiones.
10. Comenzamos configurando RAID (*Configurar RAID por software*).
11. Creamos dispositivo MD (*Multiple Devices*), elegimos RAID1, establecemos 2 discos en uso y 0 vacíos, los seleccionamos y damos a terminar.
12. Pasamos a configurar LVM (*Configurar el Gestor de Volúmenes Lógicos (LVM)*).
13. Creamos el grupo de volúmenes, con nombre *Servidor* y elegimos */dev/md0*.

14. Creamos ahora los volúmenes lógicos sobre Servidor (*swap* (1024MB),*arranque* (200MB),*hogar* (800MB) y *raíz* (resto)). Por último, damos a terminar.

15. Pasamos a la configuración del cifrado (*Configurar los volúmenes cifrados*).

16. Damos a *Create encrypted volumes* y seleccionamos todos menos *arranque* (si lo encriptamos no podremos arrancar el sistema).

17. Mantenemos los parámetros predeterminados, por lo que elegimos *Se ha terminado de definir la partición* en todos los casos.

18. Damos *Sí* a mantener la distribución existente, *Finish* y establecemos la clave *practicas,ISE* en todos los volúmenes.

19. Por último, vamos a formatear los volúmenes y asignar los puntos de montaje. Para ello, elegimos las particiones (marcadas con el símbolo #) *arranque*, *hogar* y *raíz*. Seleccionamos *Utilizar como: sistema de ficheros ext4 transaccional* y asignamos los puntos de montaje */boot*, */home* y */* respectivamente.

20. Para la partición *swap*, elegimos *Utilizar como: área de intercambio*.

21. Finalizamos el particionado y esperamos.

22. Dejamos en blanco el campo de proxy y elegimos la opción *Sin actualizaciones automáticas*, ya que queremos mantener el control total sobre el sistema.

23. Dejamos seleccionado *Standard system utilities* y pulsamos Enter.

24. Instalamos en cargador de arranque en */dev/sda*.

25. Una vez que el sistema esté instalado, iniciamos sesión y desbloqueamos los volúmenes cifrados.

26. Pasamos ahora a instalar *GRUB* en el otro disco. Para ello ejecutamos:

```
$> sudo grub install /dev/sdb
```

27. Para finalizar, tomamos una instantánea de la máquina para poder volver a este estado (Instantánea $\Longrightarrow$ Tomar)

1.1.5. Configuración de red de Ubuntu Server

Necesitamos configurar la red de forma que las máquinas puedan comunicarse entre sí,con el *host* y con el exterior. Para ello, seguiremos los siguientes pasos:

1. Crear una red sólo-anfitrión.

   a) En VirtualBox (¡no en la máquina!) damos a Archivo-Administrador de red anfitrión.

   b) Seleccionamos Crear y comprobamos que la IPv4a sea 192.168.56.1. En caso contrario, la modificamos.

   c) En la configuración de la máquina virtual Ubuntu, habilitamos el adaptador 2 (conectado a adaptador sólo-anfitrión)

2. Añadir la interfaz y activarla.

   a) Arrancamos la máquina.

   b) Editamos el archivo de interfaces mediante *sudo nano /etc/network/interfaces*.

   c) Añadimos el siguiente código:

```
# Host-only interface
auto enp0s8
iface enp0s8 inet static
address 192.168.56.105
```

   d) Salimos y guardamos.

   e) Activamos la interfaz con *sudo ifup enp0s8*.

   f) Ejecutamos *ip addr* y comprobamos que la interfaz tiene asignada la IP que configuramos (192.168.56.105).

3. Comprobar que funciona.

   a) Desde nuestro host, abrimos una terminal y ejecutamos *ping 192.168.56.105 -c 5*. Deberíamos recibir las 5 respuestas.

   b) Desde la máquina virtual ejecutamos *ping 192.168.56.1 -c 5*. Deberíamos recibir las 5 respuestas.

1.2. Sesión 2

1.2.1. Instalación de CentOS

1. Descargamos la ISO desde este enlace.

2. Creamos una máquina virtual con 1024MB de RAM y un disco duro de 10GB (elegimos la opción *Fedora 64bits*). Por último, montamos la ISO.

3. Seleccionamos *Install CentOS Linux*.

4. Elegimos Español (España).

5. En *Destino de la instalación* elegimos el único disco disponible y damos a *Listo* y a *Empezar instalación*.

6. Creamos el usuario (iniciales como usuario y *practicas,ISE* como clave). Establecemos también la contraseña del *root* (*practicas,ISE*).

7. Tomamos una instantánea de la máquina.

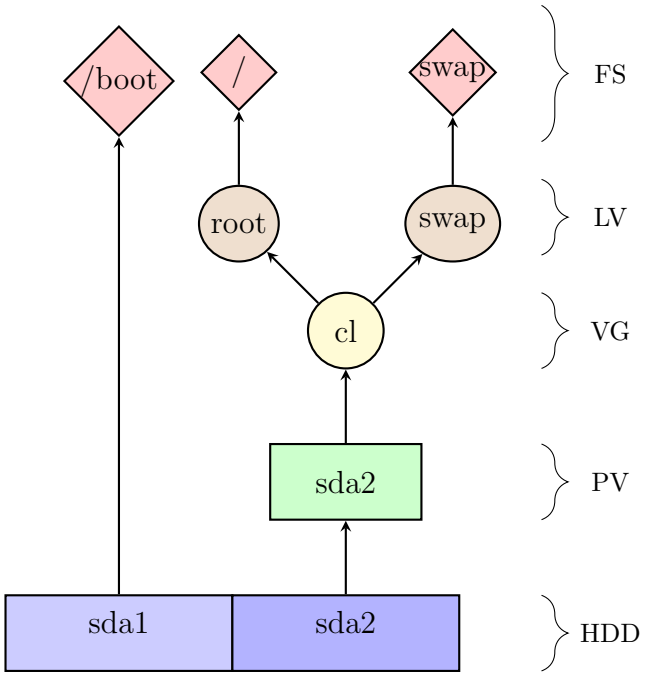
La temática de esta sesión es la siguiente: necesitamos espacio en */var* pero no tenemos suficiente. Para dar solución a ello, tendremos que seguir estos pasos:

1. Añadir el disco a la máquina.

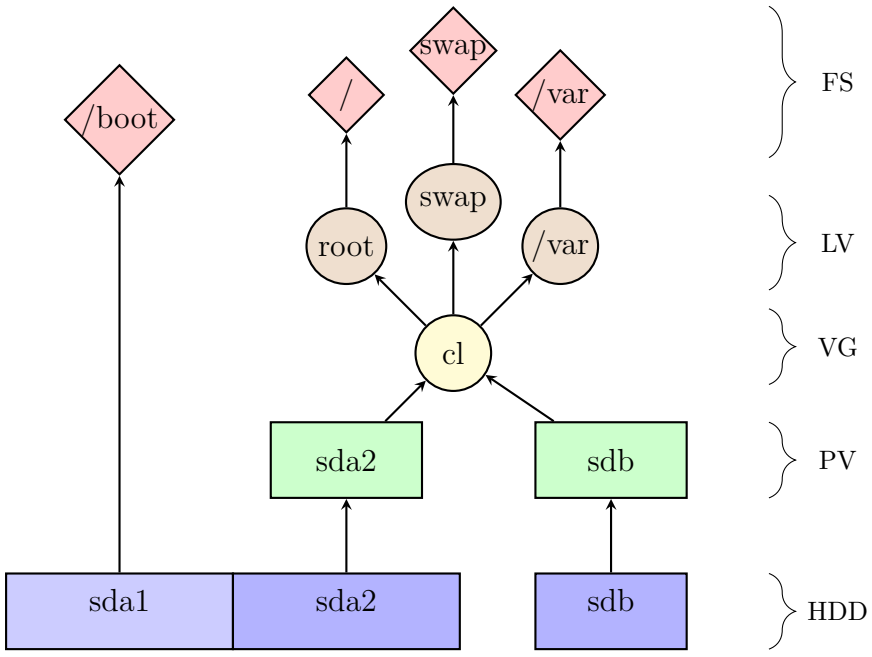
2. Configurar el disco mediante *LVM* y darle formato.

3. Copiar los datos de */var* al nuevo volumen.
4. Indicar al SO que monte el nuevo volumen en */var*.
5. Borrar los datos antiguos de */var* (en la vida real ésto se haría un tiempo prudencial después para poder disponer de un backup.)

Básicamente, tenemos que pasar del esquema actual:



Al siguiente:



Añadir el disco a la máquina Mediante el administrador de VirtualBox añadimos un nuevo disco (VDI, 10GB, reserva dinámica) a nuestra instancia.

Configurar el disco mediante *LVM* y darle formato

1. Arrancamos la máquina.
2. Comprobamos que el disco efectivamente ha sido detectado mediante *lsblk*(**LiS**t **BL**oc**K** devices). Deberíamos ver el nuevo dispositivo */dev/sdb*. Podemos apreciar también aquí que *CentOS* usa *LVM* de forma nativa, bajo un grupo de volúmenes llamado *cl*. También podemos ver como */boot* está sobredimensionado. Esto se hace para poder tener un backup del kernel antiguo en caso de que deba actualizarse.
3. Como vamos a modificar elementos del sistema, nos logueamos como superusuario:
   ```
   $> su
   ```
4. Podemos ver los detalles de *LVM* mediante los comandos *lvdisplay* (***L**ogical **V**olume Display*), *vgdisplay* (***V**olume **G**roup Display*) o *pvdisplay* (***P**hysical **V**olume Display*).
5. Creamos el volumen físico:
   ```
   #> pvcreate /dev/sdb
   ```
   Podemos leer en el *man* que podemos ejecutar el comando tanto en discos duros como en particiones. Comprobamos que se ha creado mediante *pvdisplay*.
6. Ahora debemos añadir el volumen físico al grupo de volúmenes existente (*cl*). Para ello, ejecutamos
   ```
   #> vgextend cl /dev/sdb
   ```
   Comprobamos que ha ido bien mediante *vgdisplay*.
7. Por último, debemos crear el volumen lógico. Para ello, ejecutamos el siguiente comando:
   ```
   #> lvcreate -L 5G -n newvar cl
   ```
   Donde:
   - -L indica el tamaño del volumen.
   - -n indica el nombre del volumen.

   Para finalizar, comprobamos que ha ido bien mediante *lvdisplay*.

En este momento *LVM* ha sido configurado completamente.

Dar formato al nuevo volumen

1. Debemos comenzar creando un sistema de archivos en nuestro volumen lógico. Pero primero debemos ver cuál es el punto de montaje del mismo mediante *lvdisplay* (campo *LV Path*).
2. Ahora ejecutamos el comando para crear el sistema de archivos:

```
#> mkfs -t ext4 /dev/cl/newvar
```

Copiar los datos de */var* al nuevo volumen

1. Lo primero que debemos hacer es montar el volumen. Para ello ejecutamos

   ```
   #> mkdir /mnt/newvar
   #> mount /dev/cl/newvar /mnt/newvar
   ```

   Comprobamos con *mount*.

2. Ahora debemos plantearnos lo siguiente: la copia debe realizarse de forma *atómica*, es decir, nadie puede modificar ningún archivo, ya que esto generaría inconsistencias. Para ello, accedemos al modo de mantenimiento:

   ```
   #> systemctl isolate runlevel1.target
   ```

   Introducimos de nuevo la clave del *root*.

3. Realizamos la copia de los ficheros:

   ```
   #> cp -a /var/. /mnt/newvar
   ```

   Donde:

   - -a indica que se deben copiar los archivos recursivamente, manteniendo enlaces y preservando el contexto. El contexto está compuesto por políticas de seguridad de *SELinux* (***S**ecurity **E**nhanced Linux*) que controlan el acceso a recursos de los procesos.
   - /var/. indica todos los archivos (incluidos los ocultos).

   Comprobamos mediante *ls -ahZ* sobre */var* y *mnt/newvar*

Indicarle al SO que monte el nuevo volumen en */var*

1. Abrimos el editor de texto sobre */etc/fstab*

   ```
   #> vi /etc/fstab
   ```

2. Pulsamos *i* para acceder al modo de inserción y añadimos la siguiente línea:

   ```
   /dev/mapper/cl-newvar /var ext4 defaults 0 0
   ```

3. Salimos de vi (ESC, escribimos *:wq* y Enter).

4. Desmontamos el nuevo volumen:

```
#> umount /mnt/newvar
```

5. Montamos los sistemas de archivos de */etc/fstab*:

```
#> mount -a
```

   Comprobamos ejecutando *mount*.

Borrar los datos antiguos de */var* Actualmente no tenemos acceso al antiguo */var* (el que tenemos montado actualmente es el nuevo).

1. Comenzamos desmontando */var*:

```
#> umount /dev/mapper/cl-newvar
```

2. Cambiamos el nombre de */var*:

```
#> mv /var /var_old
```

3. Creamos */var* para que se pueda montar:

```
#> mkdir /var
```

4. Restauramos el contexto de */var*:

```
#> restorecon /var
```

5. Montamos la nueva partición */var*:

```
#> mount -a
```

6. Salimos del modo de seguridad:

```
#> systemctl isolate default
```

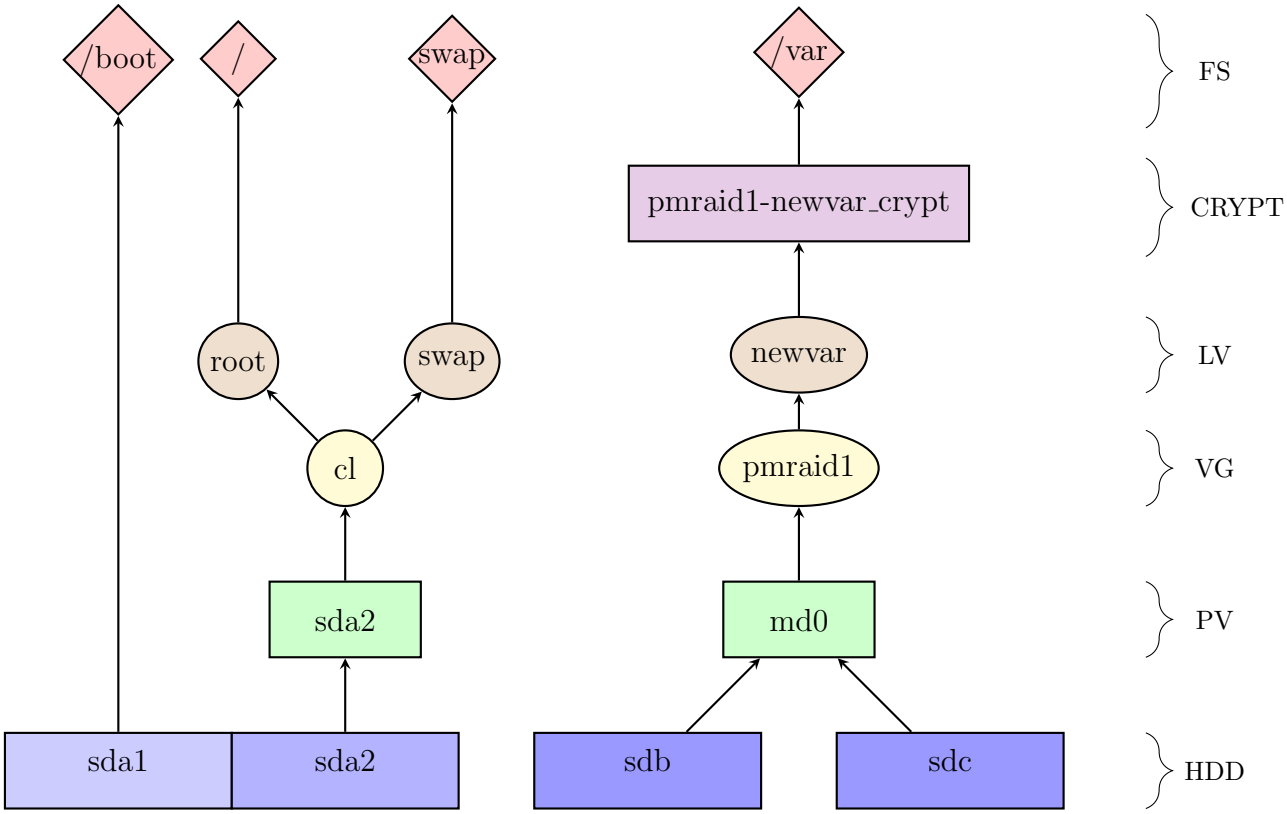
7. Eliminamos los archivos antiguos:

```
#> rm -rf /var_old
```

8. Como de costumbre, tomamos la instantánea.

1.3. Sesión 3

Deberemos partir de una instalación fresca de CentOS (restauramos la instantánea correspondiente). El objetivo es configurar un RAID1 con cifrado. El esquema sería el siguiente:



Los pasos a seguir son los siguientes:

1. Creación de RAID1.
2. Montar pila LVM añadiendo el cifrado.
3. Copiar los archivos (igual que en la sesión anterior)

Creación de RAID1

1. Mediante el administrador de VirtualBox añadimos dos discos VDI de 2GB cada uno.
2. Iniciamos la máquina y vemos que se hayan añadido correctamente mediante *lsblk*. Deberíamos ver *sdb* y *sdc*.
3. Nos logueamos como superusuarios con *su*.
4. Debemos instalar la herramienta *mdadm* para gestionar la creación de un RAID software. Para ello, la instalamos:

```
#> yum install -y mdadm
```

Al ejecutar el comando vemos que tenemos un error: no hay conexión a Internet.

5. Ejecutamos *ip addr* para ver información sobre las interfaces. Vemos que no tenemos IP asignada. Levantamos la intefaz mediante:

```
#> ifup enp0s3
```

   Comprobamos otra vez con *ip addr* que tenemos IP e instalamos *mdadm*.

6. Creamos el RAID1 mediante la herramienta que acabamos de instalar. Mediante *man* podemos ver que su sintaxis es:

```
#> mdadm <modo> <dispositivo a crear> <tipo de RAID>
   ↪  <num. de dispositivos> <dispositivos>
```

   En nuestro caso:

```
#> mdadm --create /dev/md0 --level=1 --raid-devices
   ↪ =2 /dev/sdb /dev/sdc
```

   Comprobamos mediante *lsblk* que la creación se ha realizado.

Pila LVM, cifrado y copia

1. Comenzamos creando el PV:

```
#> pvcreate /dev/md0
```

   Comprobamos con *pvdisplay*

2. Seguimos con el VG. Creamos uno nuevo porque queremos que los datos se guarden exclusivamente en el RAID1. Si añadiéramos los dispositivos al VG anterior (*cl*), los datos podrían estar en cualquier disco.

```
#> vgcreate pmraid1 /dev/md0
```

   Comprobamos con *vgdisplay*

3. Pasamos al LV:

```
#> lvcreate -L 1G -n newvar pmraid1
```

   Comprobamos con *lvdisplay*

4. Ahora vamos al cifrado. Para ello utilizaremos *LUKS,* ***L****inux* ***U****nified* ***K****ey* ***S****etup* mediante la herramienta *cryptsetup*.

   a) Comenzamos instalando el programa:

```
#> yum install -y cryptsetup
```

b) *LUKS* toma un volumen desencriptado y lo divide en dos partes: la cabecera de *LUKS* y el sistema de archivos encriptado:

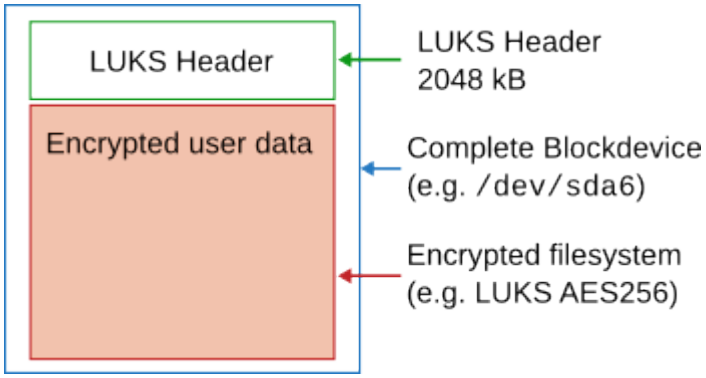


Figura 5: Esquema de *LUKS*

Por tanto, deberemos crear ambos elementos manualmente.

c) Lo primero que debemos hacer es entrar en modo de mantenimiento:

```
#> systemctl isolate runlevel1.target
```

d) Después creamos la cabecera dando el formato *LUKS*. La sintaxis es la siguiente:

```
#> cryptsetup luksFormat <dispositivo>
```

En nuestro caso:

```
#> cryptsetup luksFormat /dev/pmraid1/newvar
```

Introducimos *YES* (en mayúsculas) y establecemos la clave *practicas,ISE*. Al ejecutar este comando borramos todos los datos del volumen.

e) Ahora creamos el sistema de archivos encriptado:

```
#> cryptsetup luksOpen <dispositivo> <nombre del SA>
```

Que en nuestro caso sería:

```
#> cryptsetup luksOpen /dev/pmraid1/newvar pmraid1-
   ↪ newvar_crypt
```

Introducimos la clave anteriormente definida y comprobamos que se ha creado mediante *lsblk*.

f) Por último, modificamos el archivo */etc/crypttab*, que se encarga de asociar los volúmenes con formato *LUKS* con sus respectivos sistemas de archivos cifrados. Su estructura es la siguiente:

```
<nombre del SA> UUID=<UUID del volumen LUKS> TYPE=
    ↪ none
```

El *UUID, **U**niversal **U**nique **D**evice **ID**entifier* de un dispositivo es una forma de identificarlo unívocamente. Se obtiene mediante *blkid*. Para no tener que copiarlo a mano, redirigimos la salida al archivo:

```
#> blkid | grep crypto >> /etc/crypttab
```

g) Ahora editamos el archivo mediante *vi* y lo adaptamos a su formato. Debemos quitar las comillas del UUID. Debería quedar algo así:

```
pmraid1-newvar_crypt UUID=<UUID obtenido con grep
    ↪ sin comillas> TYPE=none
```

5. Continuamos como en la práctica 2. Damos formato al volumen encriptado:

```
#> mkfs -t ext4 /dev/mapper/pmraid1-newvar_crypt
```

6. Montamos el SA:

```
#> mkdir /mnt/varCifr
#> mount /dev/mapper/pmraid1-newvar_crypt /mnt/
    ↪ varCifr
```

Comprobamos con *mount*

7. Copiamos los archivos:

```
#> cp -a /var/. /mnt/varCifr
```

8. Editamos */etc/fstab* añadiendo la siguiente línea:

```
/dev/mapper/pmraid1-newvar_crypt /var ext4 defaults
    ↪ 0 0
```

9. Desmontamos el volumen:

```
#> umount /mnt/varCifr
```

Comprobamos con *mount*.

10. Borramos los datos antiguos:

```
#> mv /var /varOld
```

En un tiempo prudencial se eliminaría el directorio */varOld*.

11. Creamos */var* de nuevo:

```
#> mkdir /var
```

12. Restauramos el contexto:

```
#> restorecon /var
```

13. Montamos los sistemas de archivos de */etc/fstab*:

```
#> mount -a
```

    Comprobamos con *mount*.

14. Para finalizar, podemos ejecutar *lsblk* para confirmar que todo está conforme al esquema.

15. Reiniciamos la máquina y debería pedirnos la clave del cifrado.

16. Tomamos una instantánea de la máquina.

1.3.1. Configuración de red de CentOS

1. Comenzamos añadiendo la interfaz de sólo anfitrión en el adaptador 2 de la máquina mediante VirtualBox (igual que en la sesión 1).

2. Buscamos en qué interfaz se encuentra la nueva red. Para ello ejecutamos:

```
$> ip addr
```

    y vemos que la nueva es *enp0s8*.

3. Nos logueamos con permisos *root* mediante *su*

4. Creamos su script de configuración:

```
#> vi /etc/sysconfig/network-scripts/ifcfg-enp0s8
```

    y añadimos el siguiente contenido:

```
TYPE=Ethernet
BOOTPROTO=none
NAME=enp0s8
DEVICE=enp0s8
ONBOOT=yes
IPADDR=192.168.56.110
NETMASK=255.255.255.0
```

5. Levantamos la interfaz:

```
#> ifup enp0s8
```

6. Podemos editar el archivo *ifcfg-enp0s3* para activar el flag *ONBOOT* y que la interfaz que nos conecta con Internet se active al iniciar:

```
#> vi /etc/sysconfig/network-scripts/ifcfg-enp0s3
```

   Cambiamos *ONBOOT=no* por *ONBOOT=yes*.

Ahora comprobamos que toda la conectividad funciona. Para ello arrancamos ambas máquinas, CentOS y Ubuntu Server. Realizaremos las siguientes pruebas:

- Ping entre las máquinas:

```
Ubuntu Server $> ping 192.168.56.110 -c 5
CentOS $> ping 192.168.56.105 -c 5
```

- Ping de las máquinas al host (actúa como router):

```
 $> ping 192.168.56.1 -c 5
```

- Ping desde el anfitrión a las máquinas:

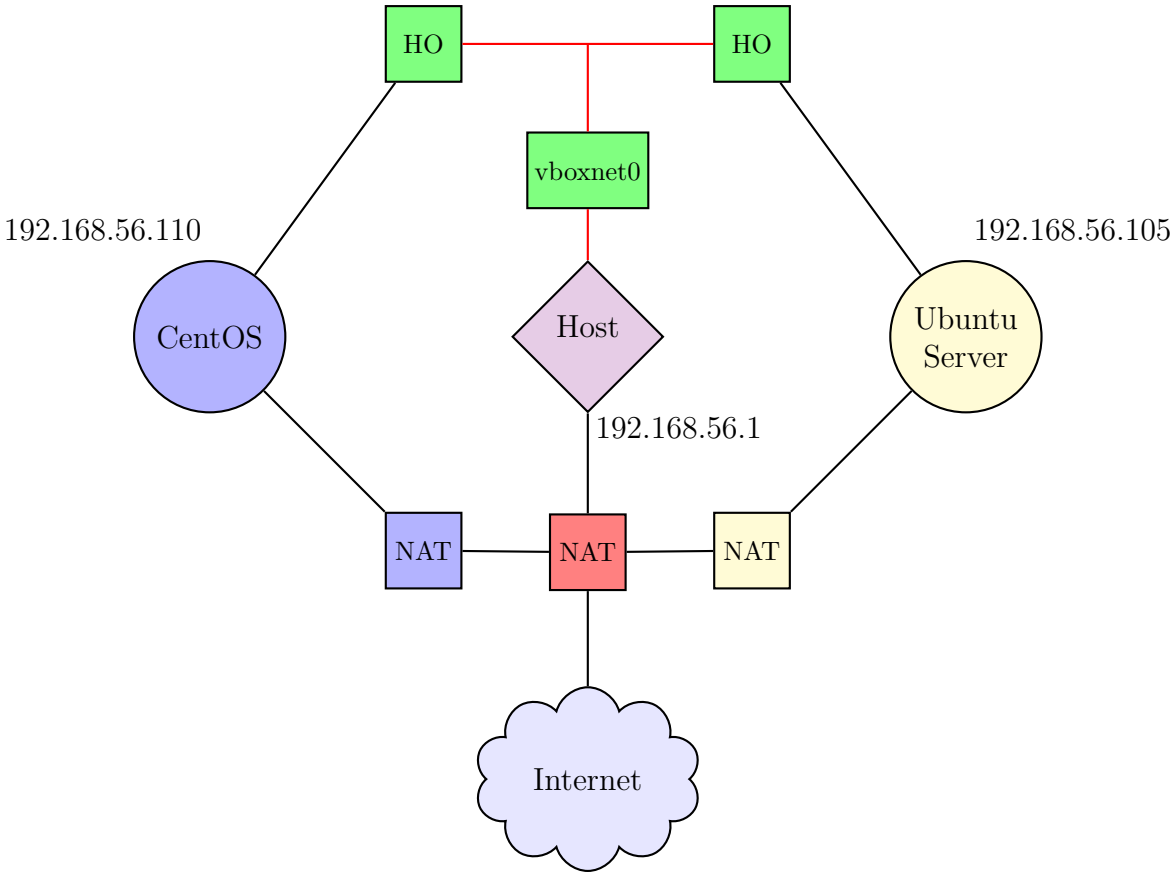
```
$> ping 192.168.56.105 -c 5
$> ping 192.168.56.110 -c 5
```

- Ping entre las máquinas e Internet:

```
$> ping google.es -c 5
```

En todos los casos las 5 respuestas deberían ser recibidas.

El esquema de red resultante es el siguiente:



Añadimos la interfaz sólo anfitrión porque las máquinas no pueden comunicarse entre ellas mediante NAT.

2. Práctica 2: Instalación y configuración de servicios

2.1. Sesión 1

En esta sesión instalaremos el servicio *ssh* en Ubuntu. SSH es un protocolo de administración remota que permite que otro se usuario se conecte a nuestro servidor y lo maneje (y viceversa). Consta de dos elementos:

- ***ssh***. Es el cliente. Sirve para conectarnos a otra máquina.
- ***sshd*** *(SSH Daemon)*. Es el servidor, sirve para que otras máquinas se conecten a la nuestra.

1. Comenzamos arrancando la máquina en la instantánea de la instalación limpia con la red configurada.
2. Instalamos el paquete mediante:

```
$> sudo apt install openssh-server
```

3. Si ejecutamos *systemctl status ssh* podemos ver que el servicio está activo y que su PID cuelga del servidor (*sshd*). Es decir, Ubuntu, a diferencia de CentOS, no diferencia entre *ssh* y *sshd*.

4. Para poder conectarnos por SSH a una máquina debemos ejecutar el comando:

```
$> ssh IP -l usuario (-p puerto)
```

   El usuario por defecto es el de la sesión abierta en ese momento y el puerto el 22. Probamos a ejecutarlo desde el host:

```
Host $> ssh 192.168.56.105 -l csp
```

   Si en alguna ocasión nos hemos conectado a esa máquina mediante SSH, tendremos que eliminar el correspondiente contenido de *~/.ssh/known_hosts* mediante

```
Host $> ssh-keygen -f "~/.ssh/known_hosts" -R
   ↪ "192.168.56.105"
```

5. Vamos a cambiar varios parámetros del **servidor**. Para ello editamos su archivo de configuración:

```
$> sudo vi /etc/ssh/sshd_config
```

   - Cambiamos el puerto al 22022 cambiando la línea

     ```
     PORT 22
     ```

     por

     ```
     PORT 22022
     ```

   - Desactivamos el acceso root cambiando la línea

     ```
     PermitRootLogin prohibit-password
     ```

     por

     ```
     PermitRootLogin no
     ```

   Para que los cambios hagan efecto reiniciamos el servicio mediante

```
$> sudo systemctl restart sshd
```

   Podemos comprobar que los cambios han funcionado conectándonos desde el host con el nuevo puerto o mediante *systemctl status sshd*.

6. Pasamos a crear la clave público privada. Se trata de un paradigma en el que el cliente genera una llave (clave privada) y un candado (clave pública), que es entregado al servidor. Para autenticarse, el cliente enviará su llave y si encaja con el candado, recibirá acceso. En nuestro caso, el cliente será CentOS y el servidor Ubuntu.

   a) Comenzamos creando llave y candado en CentOS:

```
CentOS $> ssh-keygen
```

Pulsamos Enter tres veces para elegir el destino por defecto y no establecer contraseña. Vemos que se crean dos archivos: *id_rsa* (clave privada, llave) e *id_rsa.pub* (clave pública, candado).

b) Entregamos la clave pública al servidor mediante:

```
CentOS $> ssh-copy-id 192.168.56.105 -p 22022
```

Introducimos la clave y pulsamos Enter.

c) Comprobamos que todo ha ido bien conectándonos:

```
CentOS $> ssh 192.168.56.105 -p 22022
```

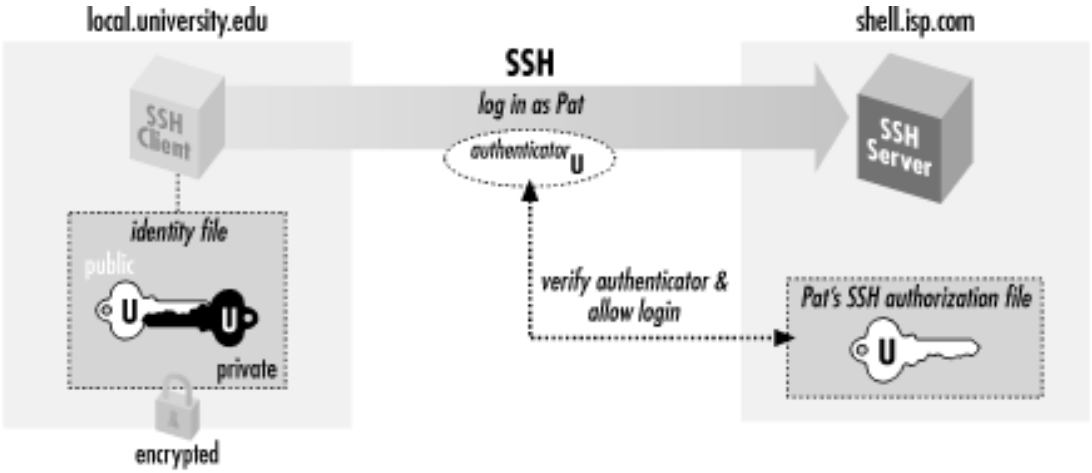


Figura 6: Esquema clave público-privada en SSH

2.2. Sesión 2

En esta sesión instalaremos el servicio SSH en CentOS y veremos las diferencias que hay con respecto a Ubuntu.

1. Primero comprobamos si el servicio está instalado:

```
$> systemctl status sshd
```

Podemos ver que sí que está activo. Sin embargo, no se nos muestra el puerto porque no somos usuarios root. Nos logueamos mediante *su* y ejecutamos de nuevo el comando anterior. Veremos que el puerto es el predeterminado (22).

2. Accedemos a la cofiguración para editar tanto el acceso root como el puerto:

```
#> vi /etc/ssh/sshd_config
```

Vemos que todo el archivo está comentado. Para modificar las reglas, descomentamos y editamos las líneas correspondientes:

```
Port 22022
PermitRootLogin no
```

Vemos que encima de la regla del puerto se nos especifica que tenemos que informar a SELinux del cambio que hemos realizado mediante *semanage*, que no está instalado por defecto.

3. Buscamos el paquete que incluye la herramienta *semanage*:

```
#> yum provides semanage
```

Instalamos el paquete que recibimos como respuesta:

```
#> yum install -y policycoreutils-python-2.5-33.el7.
   ↪ x86_64
```

4. Configuramos el puerto:

```
#> semanage port -a -t ssh_port_t -p tcp 22022
```

Comprobamos que la regla se añadió mediante:

```
#> semanage port -l | grep ssh
```

5. Ya tenemos *sshd* configurado y se puede acceder desde el equipo local. Sin embargo, debemos añadir una regla al firewall que permita las conexiones entrantes para el servicio (por defecto las bloquea).

```
#> firewall-cmd --permanent --add-port=22022/tcp
```

6. Finalmente, reiniciamos los servicios:

```
#> firewall-cmd --reload
#> systemctl restart sshd
```

En Ubuntu el firewall se denomina *ufw*. Sus comandos principales son:

```
$> sudo ufw enable       # Activar
$> sudo ufw disable      # Desactivar
$> sudo ufw allow <port> # Nueva regla
```

Podemos sintetizar las diferencias entre Ubuntu y CentOS en la siguiente tabla:

SO	Activo por defecto	Nombre del servidor	Archivo de configuración	Acceso root por defecto	Firewall
Ubuntu	×	SSH,SSHD	Sin comentar	× (prohibit-password)	UFW. Desactivado por defecto
CentOS	✓	SSHD	Comentado	✓	firewall-cmd. Activado por defecto.

2.3. Sesión 3

En esta sesión instalaremos la pila LAMP (Linux, Apache, MySQL, PHP) en Ubuntu y CentOS. Comencemos con Ubuntu:

1. Abrimos el instalador gráfico de paquetes:

```
$> sudo tasksel
```

2. Seleccionamos *LAMP Server* con la barra espaciadora y pulsamos Enter.

3. Establecemos la contraseña root de MySQL (*practicas,ISE*).

4. Esperamos a que termine de instalar y probamos los componentes uno a uno:

   a) Apache: ejecutamos *curl localhost* o visitamos *192.168.56.105* desde el navegador del host y confirmamos que podemos ver la página por defecto de Apache.

   b) MySQL: ejecutamos *mysql -u root -p*, introducimos la clave del root y comprobamos que tenemos acceso al servidor mediante el cliente (la consola).

   c) PHP: ejecutamos *php -a* y accedemos a la consola. Podemos comprobar que todo va bien ejecutando *echo("Prueba");*

Ya estaría terminada la instalación en Ubuntu. Vamos ahora a CentOS:

1. Nos logueamos como superusuario mediante *su*.

2. Comenzamos instalando Apache, que en CentOS se llama *httpd*:

```
#> yum install -y httpd
```

3. Vemos si el servicio está activado e iniciado:

```
#> systemctl status httpd
```

4. Vemos que tenemos que hacer ambas cosas:

```
#> systemctl enable httpd
#> systemctl start httpd
```

5. Comprobamos que el servicio funciona:

```
#> curl localhost
```

6. Tenemos el servicio en ejecución y funcionando, pero no podemos acceder desde otra máquina. Añadimos la correspondiente regla permanente al firewall y lo reiniciamos para que se aplique:

```
#> firewall-cmd --permanent --add-port=80/tcp
#> firewall-cmd --reload
```

   Por último, comprobamos que ya podemos acceder desde el host.

7. Pasamos a instalar MySQL. En CentOS instalaremos la versión gratuita (MariaDB).

```
#> yum install -y mariadb-server
```

8. Comprobamos el estado del servicio:

```
#> systemctl status mariadb
```

   y lo habilitamos e iniciamos:

```
#> systemctl enable mariadb
#> systemctl start mariadb
```

9. Configuramos el usuario root mediante el comando *mysql_secure_installation* (damos YES a todo).

10. Accedemos a la consola con *mysql -u root -p* y creamos una nueva base de datos.

```
MariaDB > create database mi_bd;
```

11. Pasamos a PHP. Lo instalamos:

```
#> yum install php -y
```

    y probamos que funciona (igual que en Ubuntu).

12. Ya tenemos la pila LAMP instalada. Ahora vamos a comprobar si podemos enlazar los componentes entre sí. La idea es montar un script PHP que acceda a la base de datos *mi_bd* y ejecutarlo desde el servidor web. Comenzamos instalando el conector:

```
#> yum install -y php-mysql
```

13. Ahora creamos el script:

```
#> vi /var/www/html/miscript.php
```

14. Accedemos a esta web y copiamos el ejemplo en vi. Modificamos el usuario y la contraseña con *root* y *practicas,ISE* respectivamente.

15. Ejecutamos el script para ver que funciona:

```
#> php /var/www/html/miscript.php
```

16. Ahora intentamos acceder a la ruta del archivo (*192.168.56.110/miscript.php*) desde el host y vemos que el script no se ha ejecutado, sino que estamos viendo el contenido. Accedemos a la configuración del servidor para arreglar esto:

```
#> vi /etc/httpd/conf/httpd.conf
```

Cambiamos el texto

```
<IfModule dir_module>
  DirectoryIndex index.html
</IfModule>
```

por

```
<IfModule dir_module>
  DirectoryIndex index.html miscript.php
</IfModule>
```

17. Reiniciamos el servicio y comprobamos si funciona:

```
#> systemctl restart httpd
```

18. Vemos que ya si que se nos ejecuta, pero la conexión no se realiza. Debemos autorizarla a través de SELinux:

```
#> setsebool -P httpd_can_network_connect_db 1
```

19. Probamos ahora y vemos que funciona correctamente.

2.3.1. Screen

Screen es una utilidad que permite crear sesiones de shell que podemos "descolgar"de la shell principal. Esto es muy útil si dejamos algún proceso ejecutando y la sesión se cierra.
El comando para instalar la herramienta es:

```
#> yum install -y screen
```

Los comandos más útiles son:

1. screen: crea una nueva sesión. Para descolgarla debemos pulsar Ctrl+A+D.
2. screen -list: ver una lista de las sesiones abiertas junto con su PID.
3. screen -r $< pid >$: permite recuperar la sesión asociada a $< pid >$.
4. exit: permite cerrar una sesión.

Vamos a ver dos herramientas interesantes: *Fail2Ban* y *RKHunter*. Ambas herramientas se encuentran en la release *EPEL (Extra Packages Enterprise Linux)*, por lo que deberemos instalar primero la release con:

```
#> yum install -y epel-release
```

2.3.2. Fail2Ban

Fail2Ban es una herramienta que se encarga de "banear"las IPs que intenten acceder a nuestra máquina y fallen más de n veces (por defecto 5).

1. Comenzamos instalando la utilidad:

   ```
   #> yum install -y fail2ban
   ```

2. Activamos e iniciamos el servicio:

   ```
   #> systemctl enable fail2ban
   #> systemctl start fail2ban
   ```

3. Podemos ver el estado del servicio mediante

   ```
   #> fail2ban-client status
   ```

4. Vamos a configurar una *jail* que baneará a los usuarios que introduzcan erróneamente sus credenciales en el servicio SSH. Comenzamos creando una copia del archivo de configuración (fail2ban toma los archivos *.local* como configuración, no el *.conf*).

   ```
   #> cd /etc/fail2ban
   #> cp jail.conf jail.local
   ```

5. Editamos el archivo de configuración:

   ```
   #> vi jail.local
   ```

6. Vamos a la jail de SSH:

```
# JAILS
[sshd]
```

y añadimos lo siguiente bajo *[sshd]*

```
# JAILS
[sshd]
enabled=true
port=22022
```

7. Reiniciamos el servicio y comprobamos que la cárcel funciona intentando acceder a ssh desde el host y fallando 5 veces.

8. Desbaneamos al host mediante:

```
#> fail2ban-client set sshd unban 192.168.56.1
```

2.3.3. RKHunter

RKHunter (*RootKit Hunter*) es una herramienta que escanea nuestro sistema en busca de amenazas como rootkits, troyanos, etc. Se instala mediante

```
#> yum install -y rkhunter
```

y se ejecuta un análisis con

```
#> rkhunter -c
```

3. Práctica 3: Monitorización y profiling

3.1. Sesión 1

En esta sesión comprobaremos el funcionamiento de RAID en Ubuntu. Quitaremos uno de los discos de la máquina tanto en frío como en caliente y veremos su respuesta.

1. Comenzamos restaurando la instantánea de la instalación de Ubuntu con red.

2. Vamos a los ajustes de almacenamiento, segundo disco y marcamos la opción de *conectable en caliente*.

3. Arrancamos la máquina y eliminamos el disco accediento a los ajustes de almacenamiento.

4. Vemos que Ubuntu nos avisa de que se ha perdido la conexión al disco. Si ejecutamos *lsblk* veremos que sólo hay un disco. Hemos comprobado que el sistema continúa funcional tras eliminar uno de los discos (como cabía esperar).

5. Ahora eliminaremos un disco en frío. Comenzamos restaurando la instantánea de red y eliminando el disco.

6. Arrancamos la máquina y vemos que no arranca, se queda bloqueada al intentar leer los volúmenes físicos.

7. Tendremos que esperar 300 segundos para que cargue *initramfs*, un shell mínimo que permite realizar labores de mantenimiento.

8. Podemos ver información de nuestro sistema RAID mediante

```
(initramfs)> cat /proc/mdstat
```

9. Podemos ver que RAID está inactivo. El campo *Personalities* nos dice los tipos de RAID que acepta nuestra máquina.

10. También podemos ver el diario del sistema (*journalctl*) mediante

```
(initramfs)> dmesg
```

11. Deducimos que debemos activar RAID. Para ello ejecutamos el siguiente comando (extraído del manual):

```
(initramfs)> mdadm --run /dev/md0
```

12. Podemos ver mediante

```
(initramfs)> cat /proc/mdstat
```

    que el RAID ya está activo.

13. Pulsamos Ctrl+D para arrancar el sistema y vemos que podemos iniciar sesión.

14. Ahora toca añadir un nuevo disco al RAID. Comenzamos añadiéndolo en las opciones de almacenamiento.

15. Comprobamos mediante *lsblk* que se ha añadido el nuevo disco a la máquina.

16. Ahora lo añadimos al RAID:

```
$> sudo mdadm --add /dev/md0 /dev/sdb
```

17. Podemos ver el proceso de clonado de los discos mediante:

```
$> watch cat /proc/mdadm
```

18. Por último, cuando termine el proceso podemos comprobar que todo está configurado correctamente mediante *lsblk*.

3.2. Instalación de Zabbix

En esta sesión instalaremos Zabbix en Ubuntu y lo configuraremos para que se monitorice a sí mismo y para que monitorice a CentOS. La documentación que usaremos será la siguiente.

La idea es instalar *server, frontend* y *agent* en Ubuntu y *agent* en CentOS. Veamos primero qué es cada componente:

- **Server**. Es el proceso central del monitor. Se encarga de capturar información y enviarla a los usuarios.
- **Agent**. Monitoriza activamente los recursos y aplicaciones locales. Toma estos datos y los envía al servidor para que los procese.
- **Frontend**. Muestra los datos que se han tomado de forma gráfica.

3.2.1. Instalación en Ubuntu

1. Arrancamos Ubuntu.
2. Descargamos e instalamos el repositorio de Zabbix para nuestra versión de Ubuntu (Xenial):

```
$> wget https://repo.zabbix.com/zabbix/3.4/ubuntu/pool/main/z/
   ↪ zabbix-release/zabbix-release_3.4-1+xenial_all.deb
$> sudo dpkg -i zabbix-release_3.4-1+xenial_all.deb
$> sudo apt update
```

3. Instalamos el servidor que funciona con MySQL y el frontend:

```
$> sudo apt install -y zabbix-server-mysql zabbix-frontend-php
```

4. Pasamos ahora a la configuración de la base de datos. Comenzamos accediendo a MySQL y creando una nueva base de datos:

```
$> mysql -u root -p
mysql> create database zabbix character set utf8 collate utf8_bin
   ↪ ;
mysql> grant all privileges on zabbix.* to zabbix@localhost
   ↪ identified by 'practicas,ISE';
mysql> exit;
```

5. Importamos el esquema inicial (el proceso tarda mucho, alrededor de media hora):

```
$> zcat /usr/share/doc/zabbix-server-mysql/create.sql.gz | mysql
   ↪ -u zabbix -p zabbix
```

6. Accedemos al fichero de configuración:

```
$> sudo vi /etc/zabbix/zabbix_server.conf
```

y establecemos la clave (el resto de campos vienen por defecto):

```
DBPassword=practicas,ISE
```

7. Iniciamos y activamos el servicio:

```
$> sudo systemctl enable zabbix-server
$> sudo systemctl start zabbix-server
```

8. Configuramos la zona horaria de PHP:

```
$> sudo vi /etc/php/7.0/apache2/php.ini
```

Buscamos el texto [Date] y lo modificamos:

```
[Date]
date.timezone="Europe/Madrid"
```

9. Reiniciamos el servidor web:

```
$> sudo systemctl restart apache2
```

10. Añadimos la correspondiente regla al firewall:

```
$> sudo ufw allow 80/tcp
```

11. Desde el host, accedemos a *192.168.56.105/zabbix* para iniciar el proceso de instalación. Pulsamos en *Next* y deberíamos ver todos los flags en OK.
12. Pasamos al siguiente paso e introducimos la contraseña de la base de datos en el campo *Password*.
13. Damos un nombre al servidor, por ejemplo, *ZabbixUbuntu*.
14. Revisamos todos los cambios y los aceptamos.
15. Accedemos a la interfaz del frontend con las credenciales por defecto: Admin/zabbix.
16. Por último, instalamos, activamos y arrancamos el agente:

```
$> sudo apt install -y zabbix-agent
$> sudo systemctl enable zabbix-agent
$> sudo systemctl start zabbix-agent
```

3.2.2. Instalación en CentOS

Ahora instalaremos el agente en CentOS.

1. Comenzamos logueándonos como root y añadiendo el repositorio:

```
#> rpm -ivh https://repo.zabbix.com/zabbix/3.4/rhel/7/x86_64/
   ↪ zabbix-release-3.4-2.el7.noarch.rpm
```

2. Instalamos el agente:

```
#> yum install -y zabbix-agent
```

3. Activamos y arrancamos el servicio:

```
#> systemctl enable zabbix-agent
#> systemctl start zabbix-agent
```

4. Vemos que al iniciar el servicio obtenemos un error relacionado con un límite de recursos. Para solucionarlo hacemos lo siguiente:

```
#> cat /var/log/audit/audit.log | grep zabbix_agentd | grep
   ↪ denied | audit2allow -M zabbix_agent_setrlimit
#> semodule -i zabbix_agent_setrlimit.pp
```

5. Añadimos la IP del servidor de Zabbix (Ubuntu) al archivo de configuración:

```
#> vi /etc/zabbix/zabbix_agentd.conf
```

```
Server=192.168.56.105
ServerActive=192.168.56.105
```

6. Vemos que el puerto en el que escucha el agente es el 10050, por lo que debemos abrirlo tanto en CentOS como en Ubuntu:

```
Ubuntu $> sudo ufw allow 10050/tcp
CentOS #> firewall-cmd --zone=public --add-port=10050/tcp --
   ↪ permanent
CentOS #> firewall-cmd --reload
```

7. Por último reiniciamos el servicio:

```
#> systemctl restart zabbix-agent
```

3.2.3. Añadir CentOS al frontend

Ahora añadiremos el host CentOS al frontend. Para ello, accedemos a la web, Configuration, Hosts, Create Host. Establecemos su nombre e IP:

```
Host name: CentOS
New group: ISE
IP address: 192.168.56.110
```

Para finalizar, activamos la monitorización de Ubuntu haciendo click en su estado (*Disabled*).

3.2.4. Monitorización de SSH y HTTP

1. Hacemos click en el host de CentOS que acabamos de crear.
2. Vamos a la pestaña *Templates*.
3. Añadimos las siguientes plantillas:
   - Template App HTTP Service
   - Template App SSH Service
4. Pulsamos en *Add* y luego en *Update*.
5. Debemos informar a la plantilla de SSH que servimos en el puerto 22022. Para ello, vamos a Configuration-Templates-Template App SSH Service.
6. Vamos a Items-SSH service is running.
7. En el campo *Key* cambiamos

   ```
   net.tcp.service[ssh]
   ```

   por

   ```
   net.tcp.service[ssh,,22022]
   ```

8. Hacemos click en *Update* y reabrimos el frontend. Vemos que se nos marca el problema de conexión a SSH como resuelto en la pestaña *Dashboard*.

Configuración de gráficas

1. En el frontend, accedemos a *Configuration-Hosts*.
2. Entramos en la sección *Graph* del host para el que vayamos a configurar la gráfica.
3. Damos en *Create Graph*.
4. Le damos el nombre *HTTP and SSH services* y añadimos ambos ítems.
5. En *Dashboard-Graphs* podremos ver las gráficas creadas.

3.2.5. Instalación de *zabbix_get*

Esta utilidad nos permite realizar peticiones de datos de monitorización desde el servidor al agente. Para instalarlo, ejecutamos la siguiente orden:

```
Ubuntu $> sudo apt install -y zabbix-get
```

Podemos usar *zabbix_get* para comprobar, por ejemplo, si los servicios SSHD y HTTPD están activos:

```
Ubuntu $> zabbix_get -s 192.168.56.110 -k net.tcp.service[ssh,,22022]
Ubuntu $> zabbix_get -s 192.168.56.110 -k net.tcp.service[http]
```

3.3. Sesión 2

En esta sesión trabajaremos con *Ansible*. *Ansible* es una plataforma que permite administrar computadoras distribuyendo una orden a un grupo de ordenadores que está construida sobre *Python*. Por ejemplo, podríamos ejecutar un script de configuración en varias máquinas con una sola línea. Ansible es *agentless*, es decir, no precisa la instalación de un agente en el cliente. Tan solo es necesario disponer del servicio SSH (y Python) y un par de claves público-privadas.

1. Para comenzar, instalamos *ansible* en el host:

   ```
   Host $> sudo apt install -y ansible
   ```

2. Como Python no está instalado en Ubuntu, lo instalamos:

   ```
   Ubuntu $> sudo apt install -y python
   ```

3. Debemos crear dos grupos de máquinas. Para ello editamos el archivo */etc/ansible/hosts*:

   ```
   [LosUbuntus]
   192.168.56.105:22022
   [LosCentOS]
   192.168.56.110:22022
   ```

4. Podemos ver que se han añadido los clientes mediante:

   ```
   Host $> ansible all --list-hosts
   ```

5. Creamos las claves RSA y copiamos la pública a los clientes:

   ```
   Host $> ssh-keygen
   Host $> ssh-copy-id csp@192.168.56.110 -p 22022
   Host $> ssh-copy-id csp@192.168.56.105 -p 22022
   ```

6. Ejecutemos un *ping* para ver que todo se ha instalado correctamente. La sintaxis de *ansible* es la siguiente:

   ```
   Host $> ansible <group> -m <command>
   ```

   en nuestro caso:

```
Host $> ansible LosUbuntus -m ping
Host $> ansible LosCentOS -m ping
Host $> ansible all -m ping
```

7. Creemos un directorio en ambos clientes para asegurar su funcionamiento:

```
Host $> ansible all -m command -a "mkdir /home/csp/test"
```

8. Si ejecutamos *ls* en ambos clientes veremos que el directorio ha sido creado.

4. Práctica 4: Benchmarking y Ajuste del Sistema

4.1. Sesión 1

En esta sesión instalaremos la suite de benchmarking *phoronix* y ejecutaremos dos tests sobre Ubuntu y CentOS para comparar su rendimiento. También ejecutaremos *ab* (Apache Benchmark) desde el host para evaluar el rendimiento de las máquinas al responder peticiones HTTP.

1. Comenzamos instalando la suite en las máquinas:

```
CentOS #> yum install -y phoronix-test-suite
Ubuntu $> sudo apt install -y phoronix-test-suite php-zip
```

2. Podemos ver los tests disponibles mediante:

```
$> phoronix-test-suite list-available-tests
```

3. Elegimos dos y los ejecutamos mediante:

```
$> phoronix-test-suite benchmark <nombre del test>
```

   En mi caso:

```
$> phoronix-test-suite benchmark pts/ramspeed
$> phoronix-test-suite benchmark pts/sudokut
```

4. Se nos pedirá nuestra clave para instalar las dependencias. La introducimos y pulsamos Enter. No se mostrará ninguna salida hasta que no se instalen todas.

5. Elegimos las opciones deseadas del test y le damos nombre, ID y descripción.

6. El test comenzará a ejecutarse. Cuando termine, podremos ver los resultados. En mi caso, Ubuntu saca mejor puntuación en ambos.

Pasemos ahora a Apache Benchmark:

1. Comenzamos instalando *apache2* en nuestro host:

```
Host $> sudo apt install -y apache2
```

2. La sintaxis de la herramienta es la siguiente:

```
Host $> ab -n <peticiones> -c <peticiones concurrentes> <host>
```

   Por ejemplo:

```
Host $> ab -n 1000 -c 100 192.168.56.105/
Host $> ab -n 1000 -c 100 192.168.56.110/
```

Si no añadimos la barra al final del host el benchmark no se ejecutará.

3. Cuando termine, podremos ver varias métricas. La que nos interesa es el tiempo medio de respuesta, *Time per request (mean)*. En mi caso, Ubuntu es más rápido que CentOS.

También podemos ejecutar los tests desde un contenedor de Docker:

1. Comenzamos instalando Docker:

```
$> curl -fsSL https://download.docker.com/linux/ubuntu/gpg |
   ↪ sudo apt-key add -
$> sudo add-apt-repository"deb [arch=amd64] https://download.
   ↪ docker.com/linux/ubuntu$(lsb_release -cs) stable"
$> sudo apt update
$> sudo apt install docker-ce
$> sudo usermod -aG $USER
```

2. Cerramos sesión y volvemos a loguearnos.

3. Descargamos y arrancamos el contenedor de phoronix:

```
$> docker run phoronix/pts
```

4. Ejecutamos los tests:

4.2. Sesión 2

En esta sesión usaremos jMeter para hacer pruebas de estrés contra una API REST (https://github.com/davidPalomar-ugr/iseP4JMeter).

1. Comenzamos descargando la utilidad desde aquí en nuestro host. La descomprimimos y la abrimos:

```
Host $> java -jar <path de la app>/bin/ApacheJMeter.jar
```

2. Instalamos Docker en Ubuntu.

3. Descargamos la aplicación en Ubuntu:

```
Ubuntu $> git clone https://github.com/davidPalomar-ugr/
   ↪ iseP4JMeter.git
```

4. Nos situamos en la carpeta del proyecto y levantamos los contenedores:

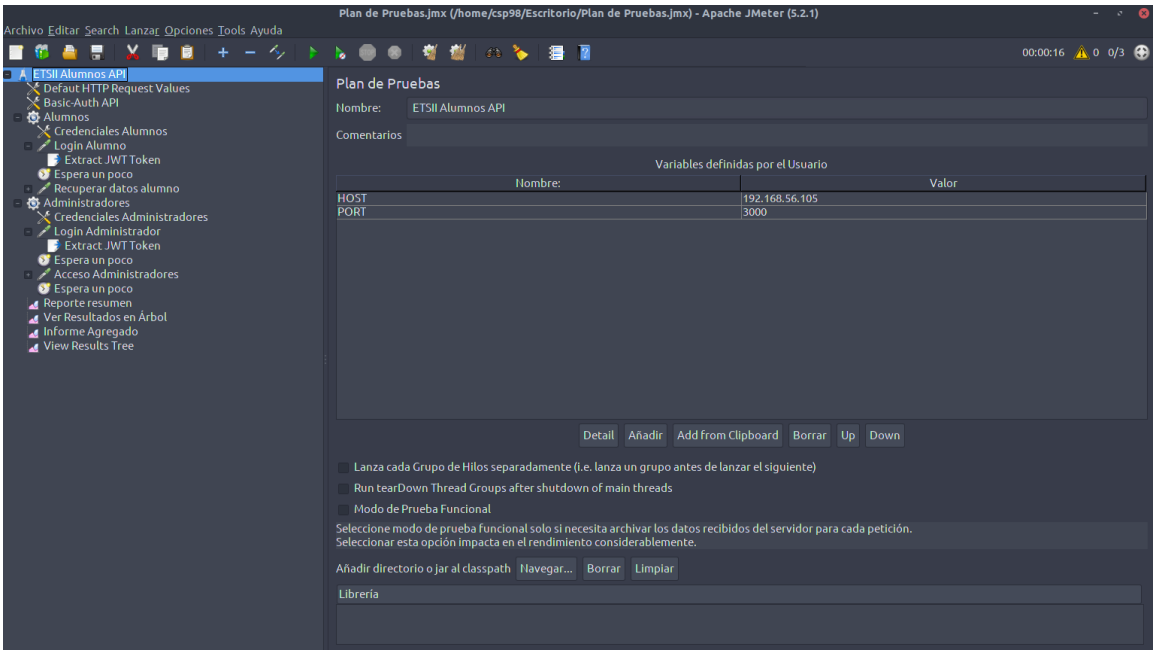
```
Ubuntu $> cd iseP4JMeter
Ubuntu $> docker-compose up -d
```

5. Comprobamos que la app funciona ejecutando el script *pruebaEntorno.sh*.

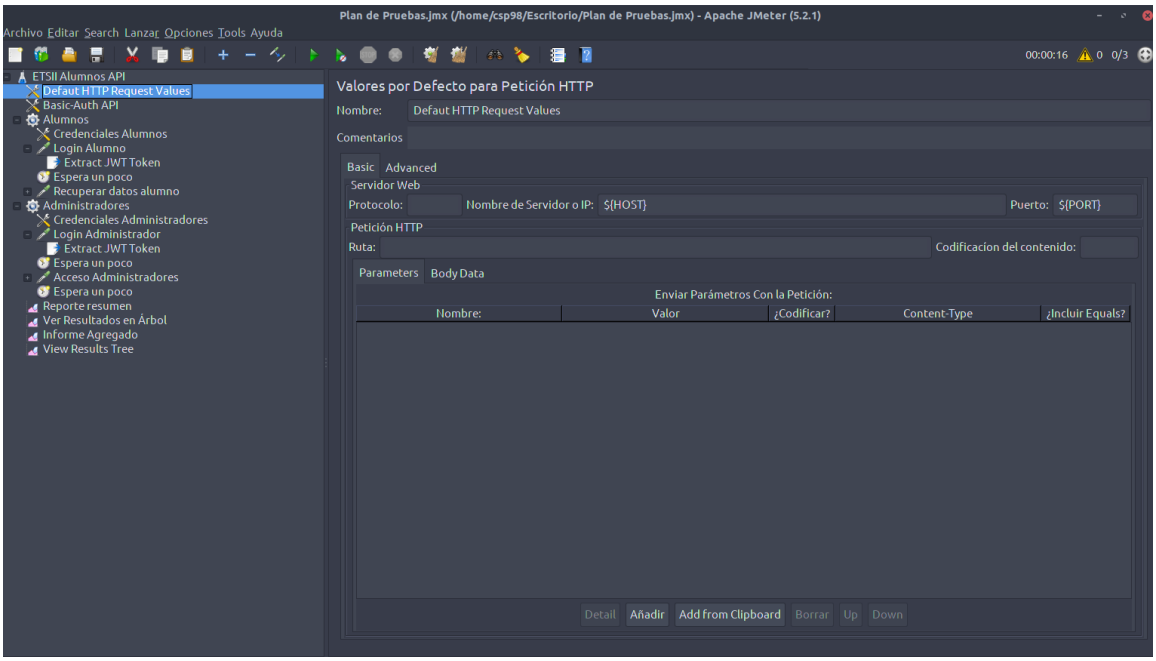
```
Ubuntu $> sh pruebaEntorno.sh
```

6. Comenzamos a crear el plan de pruebas en JMeter:

7. Establecemos las variables HOST y PORT en el plan de pruebas:

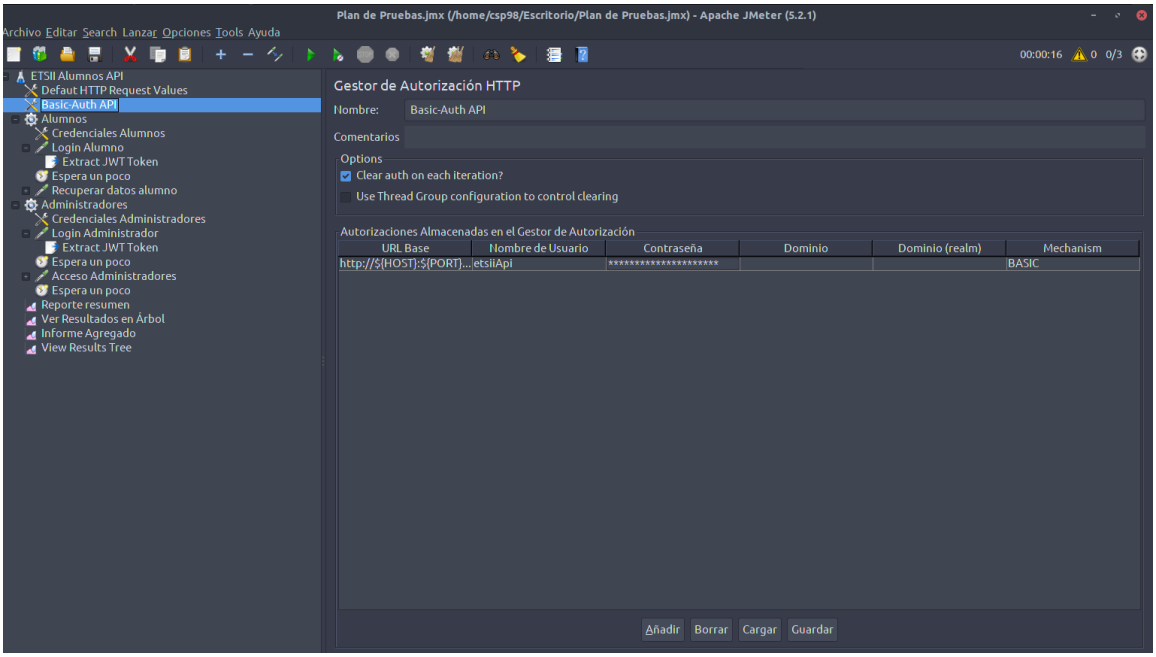


8. Creamos unos valores por defecto para la petición HTTP: botón derecho en Plan de Pruebas, Añadir, Elemento de Configuración, Valores por defecto para petición HTTP. Configuramos el host y el puerto referenciando a las variables ya definidas.

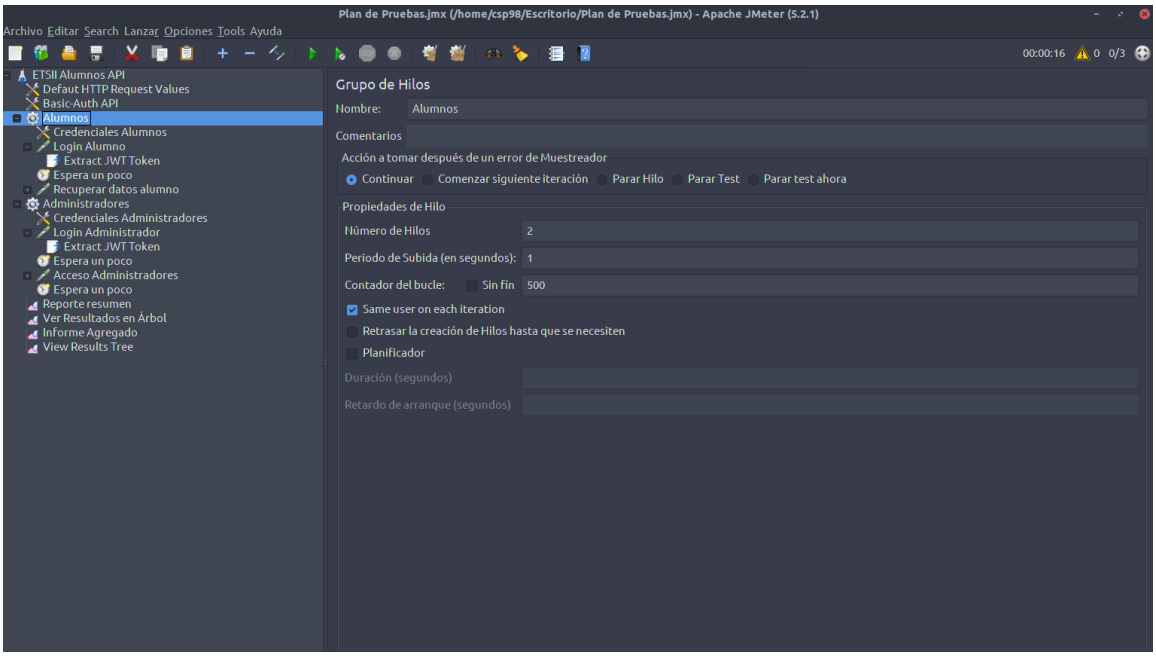


9. Creamos la autenticación para la API: Plan de Pruebas, Añadir, Elemento de Configuración, Gestor de Autorización HTTP. Configuramos:

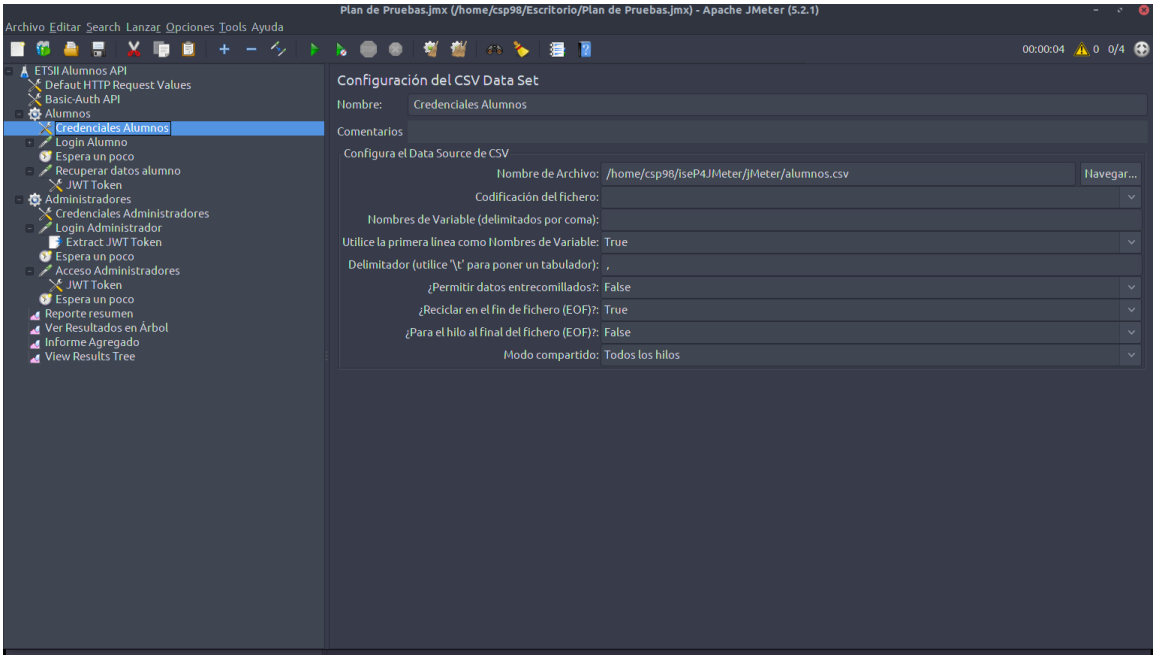
   - URL base: http://${HOST}:${PORT}/api/v1/auth/login
   - Nombre de usuario: etsiiApi
   - Contraseña: laApiDeLaETSIIDaLache



10. Creamos el grupo de hilos de Alumnos: Plan de Pruebas, Añadir, Hilos (usuarios), Grupo de Hilos. Establecemos dos hilos.

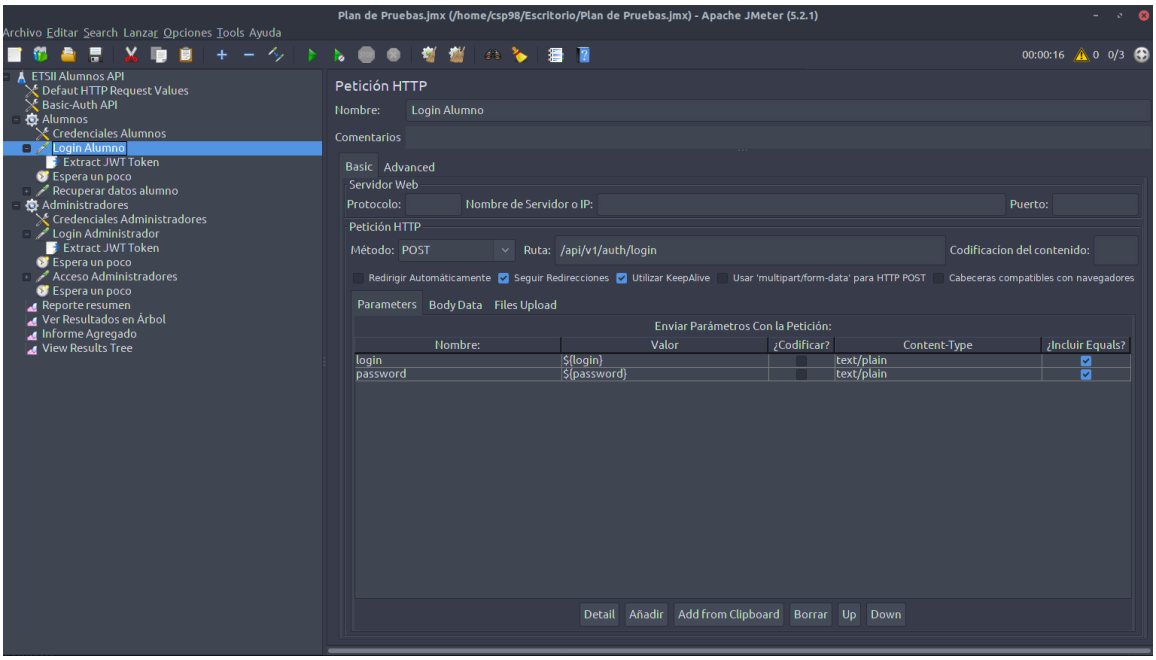


11. Añadimos el acceso a los credenciales de alumnos almacenados en el archivo CSV. Alumnos, Añadir, Elemento de Configuración, Configuración del CSV Data Set. Establecemos la ruta del archivo CSV.



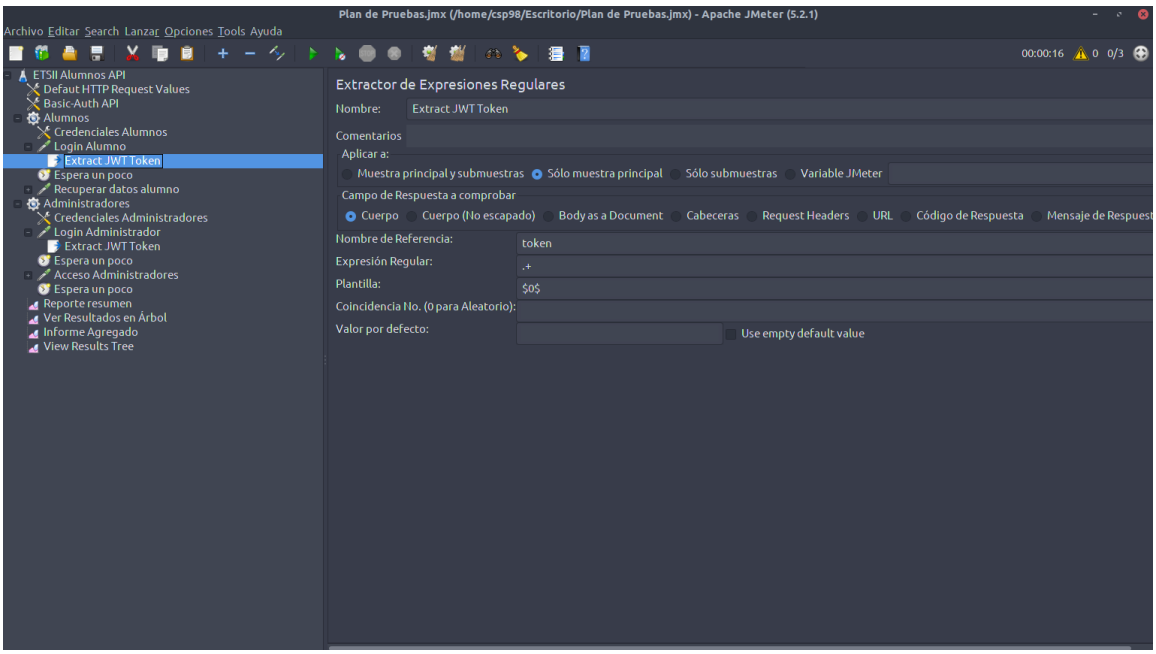
12. Creamos la petición de login del alumno. Alumnos, Añadir, Muestreador, Petición HTTP. Configuramos los siguientes campos:

    - Método: POST
    - Ruta: /api/v1/auth/login
    - Parámetros para la petición.

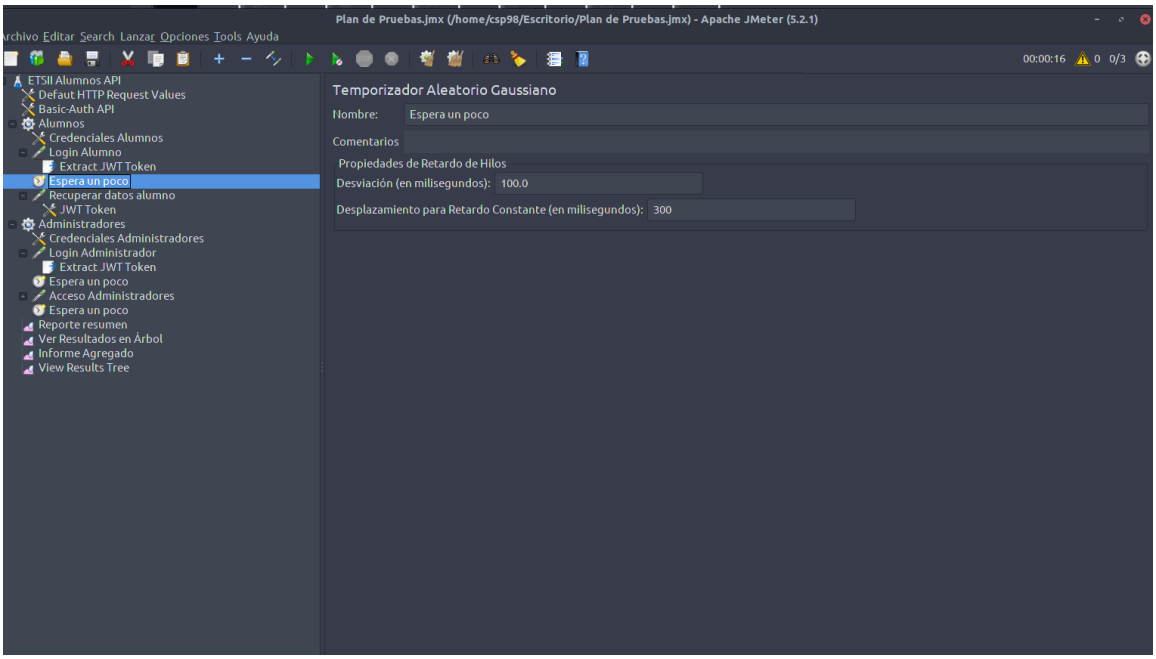


13. Ahora debemos extraer el token JWT. Login Alumno, Añadir, Post Procesadores, Extractor de expresiones regulares. Configuramos los siguientes campos:

    - Nombre de referencia: token
    - Expresión regular: .+
    - Plantilla: 0

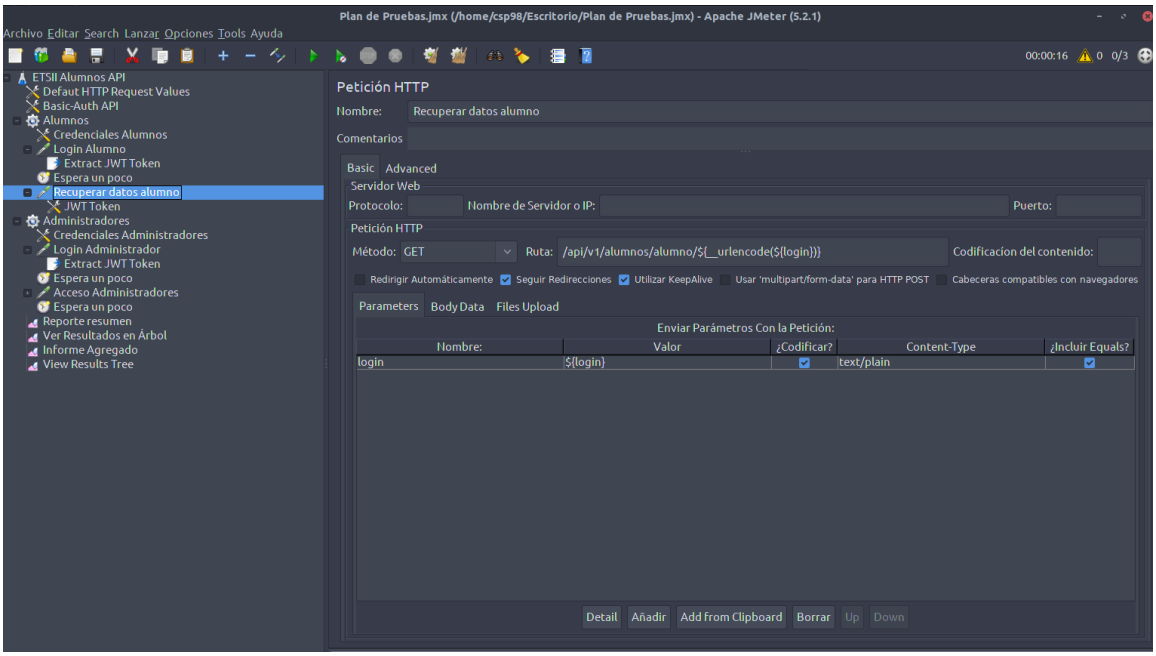


14. Esperamos a la extracción del token: Alumnos, Añadir, Temporizador, Temporizador Aleatorio Gaussiano.

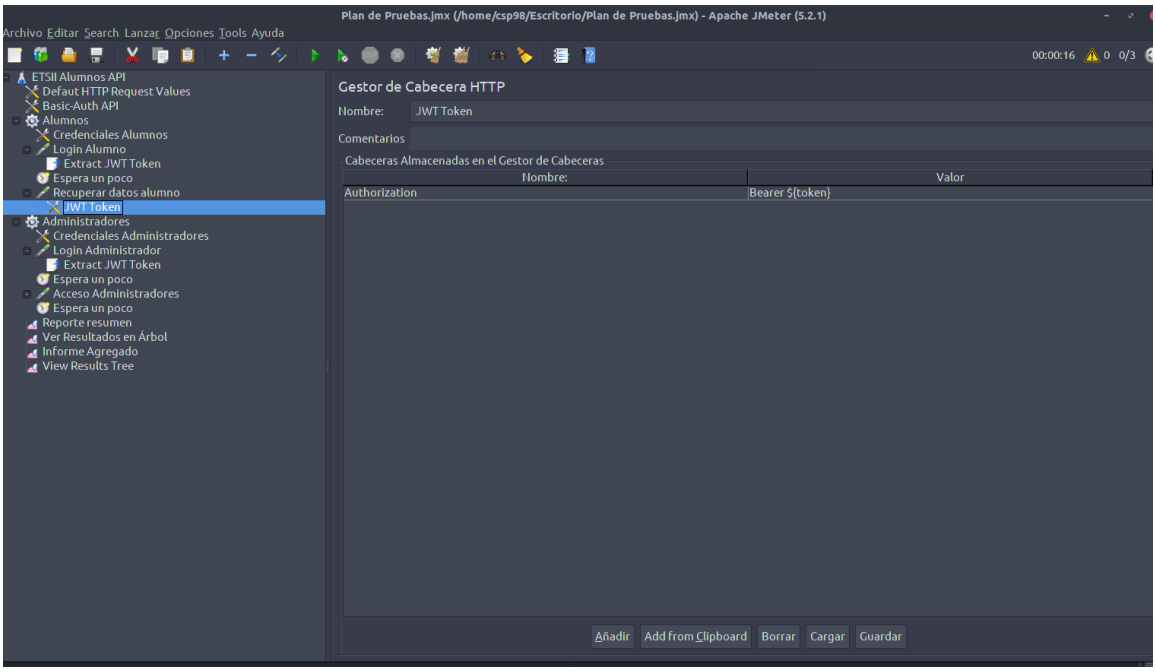


15. Añadimos la petición para recuperar los datos del alumno. Alumnos, Añadir, Muestreador, Petición HTTP. Configuramos los siguientes campos:

    - Método: GET
    - Ruta: /api/v1/alumnos/alumno/${__urlencode(${login})} . La función *_urlencode* se utiliza para traducir caracteres como la @ a un formato aceptado por HTTP.
    - Parámetro *login* para la petición.



16. Añadimos un gestor de cabecera para el token: Alumnos, Añadir, Elemento de Configuración, Gestor de Cabecera HTTP.



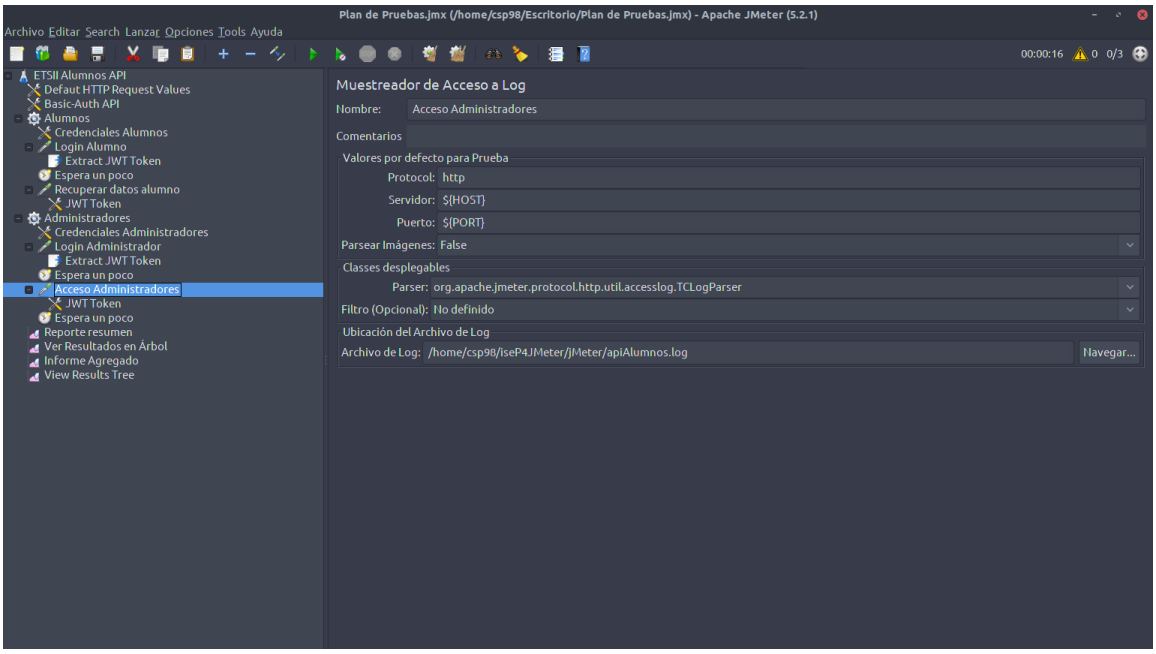
17. Creamos ahora el hilo de administradores igual que el de alumnos.

18. Añadimos la configuración del dataset con los credenciales de administrador seleccionando el CSV correspondiente.

19. Creamos la petición de login para el administrador, la extracción del token JWT y la espera igual que en alumnos.

20. Creamos la petición para el logueo del acceso de los administradores. Administradores, Añadir, Muestreador, Muestreador de Acceso a Log. Establecemos los siguientes campos:

    - Protocolo: http
    - Servidor: ${HOST}
    - Puerto: ${PORT }
    - Archivo de log.



21. Añadimos el gestor de cabecera HTTP al acceso de administradores (igual que en alumnos).

22. Añadimos un temporizador aleatorio gaussiano al grupo de hilos de administradores.

23. Añadimos los gráficos. Plan de Pruebas, Añadir, Receptor:

    - Reporte resumen.
    - Ver resultados en árbol.
    - Informe agregado.
    - Árbol de resultados.

24. Damos al icono de Play y tras 10 segundos a STOP. En el reporte resumen tendremos que tener todos los errores al 0 %.

5. Preguntas de examen

5.1. Práctica 1

1. Si monto un RAID0 y un RAID1 con 2 HDD de 10GB, ¿cuánto espacio tengo disponible?

   Solución

   - RAID0: 20GB, ya que no hay redundancia.
   - RAID1: 10GB, ya que los datos se replican.

   ¿Y si uso un HDD de 10GB y otro de 50GB?

   Solución

- RAID0: 60GB.
- RAID1: 10GB. Si tomáramos más espacio no podría replicarse.

2. Disponemos de un sistema con particiones *home*, *swap*, *boot* y *root*. Sólo podemos encriptar una de ellas. ¿Cuál elegirías?

   Solución

   SWAP, ya que contiene datos de la RAM, como claves, archivos de las aplicaciones, etc.

3. ¿Qué diferencias hay entre *EXT2* y *EXT4*?

   Solución

   EXT4, a diferencia de *EXT2*, admite *journaling*, es decir, un diario que permite restablecer los datos anteriores a una transacción en caso de que ésta falle. Esta técnica permite al sistema de archivos volver a un estado coherente si se produce un corte de electricidad o cualquier otro fallo durante una transacción. Otra característica muy interesante de *EXT4* es el asignamiento multi-bloque, que permite asignar varios bloques en una misma llamada.

4. ¿Qué diferencias hay entre:

   - *cp -a /var/ /newvar*
   - *cp -a /var/* /newvar*
   - *cp -a /var/. /newvar*

   ?

   Solución

   - *cp -a /var /newvar* copia la carpeta, no el contenido. Es decir, quedaría */newvar/var/archivos*
   - *cp -a /var/* /newvar* no copia los archivos ocultos.
   - *cp -a /var/. /newvar* copia todos los archivos (visibles y ocultos).

5. ¿Qué tipos de *LUKS* hay? ¿Cuál usamos nosotros en la sesión 3 de la práctica 1?

   Solución

   - *LUKS* on *LVM*. Consiste en montar la pila LVM y después cifrar mediante *LUKS* los LV deseados. Es más eficiente y permite tener LV cifrados y LV no cifrados. Es el óptimo cuando no se quiere cifrar todo el disco.
   - *LVM* on *LUKS*. Ciframos todo el disco con *LUKS* y después montamos LVM encima. Es más seguro pero conlleva un peor rendimiento.

   En la sesión 3 de la práctica 1 utilizamos *LUKS* on *LVM*.

5.2. Práctica 2

1. Tras instalar el servicio *ssh* en una máquina, ¿está activo?

   Solución

   Sí, está activo con los parámetros por defecto, por ejemplo, el puerto 22.

2. ¿Cómo diferencia Ubuntu entre el cliente y el servidor SSH?

   Solución

   No lo hace, para él ambos son el mismo servicio (el PID de *ssh* cuelga del de *sshd*). Lo que sí debemos tener en cuenta es el archivo de configuración que editamos (*/etc/ssh/sshd_config* o */etc/ssh/ssh_config*).

3. ¿Hay algún problema si consiguen robarme la clave pública (candado)?

   Solución

   No, ya que solo la clave privada asociada (llave) es capaz de abrirlo. Lo que sí que sería un problema es que alguien obtuviera la clave privada, ya que podría acceder a la máquina con facilidad.