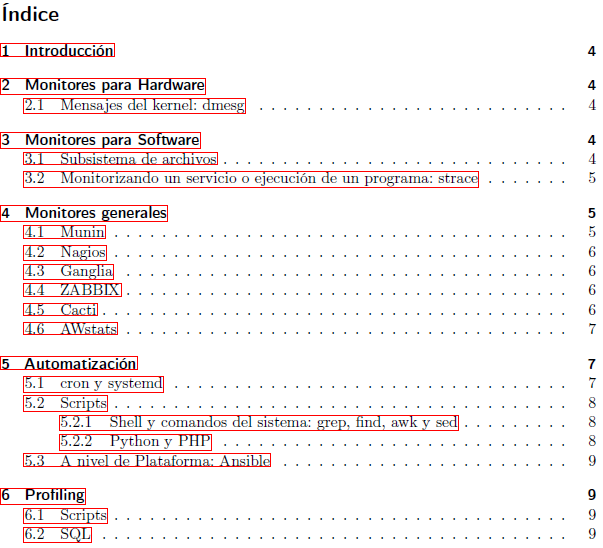
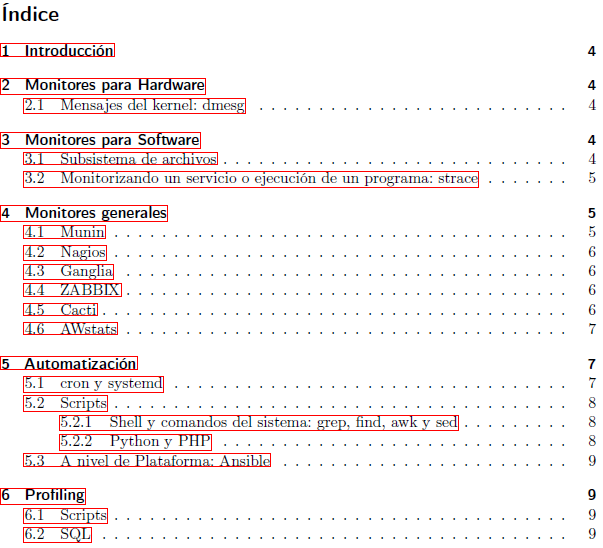
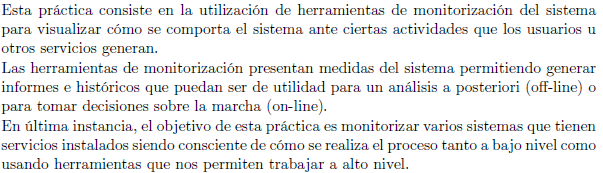
# PRÁCTICA 3: Monitorización y “Profiling”



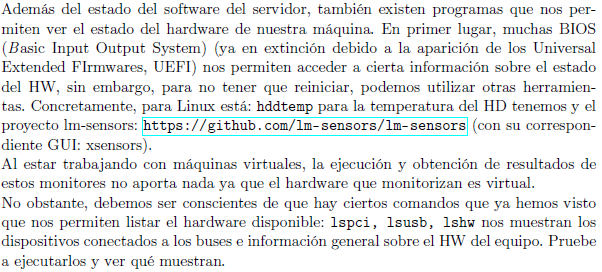
## LECCIÓN1: Monitorización del RAID1, Monitores y Automatización

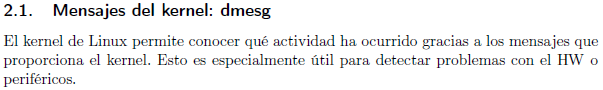


1. **Introducción**

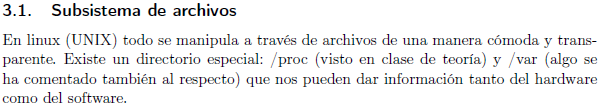


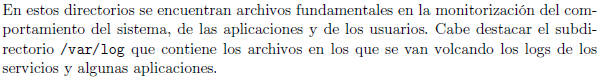
1. **Monitores para Hardware**

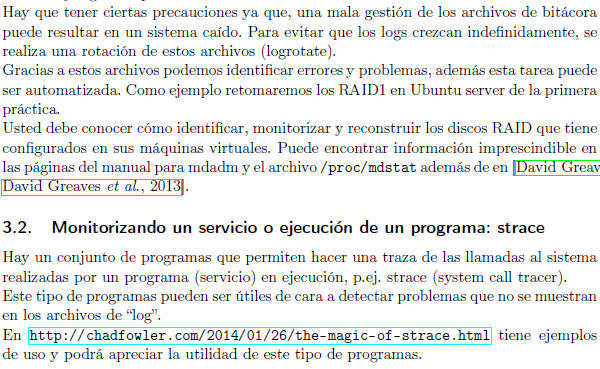




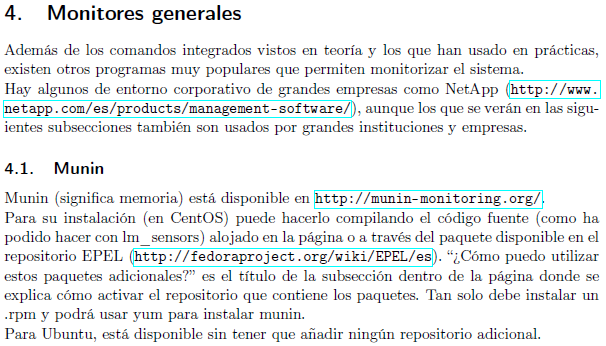
1. **Monitores para Software**

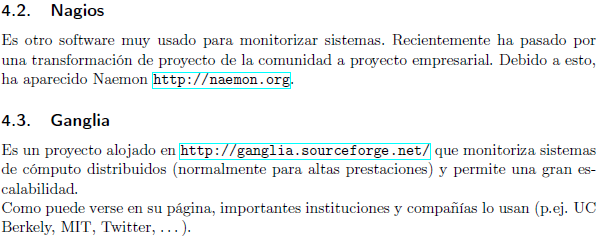


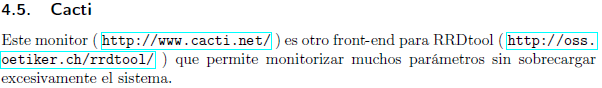


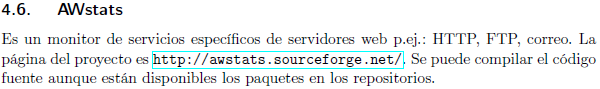


1. **Monitores generales**

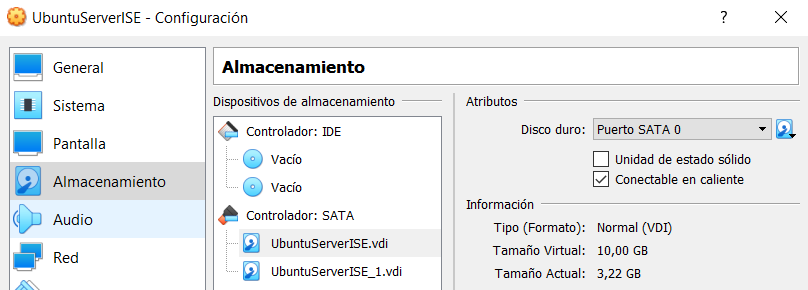




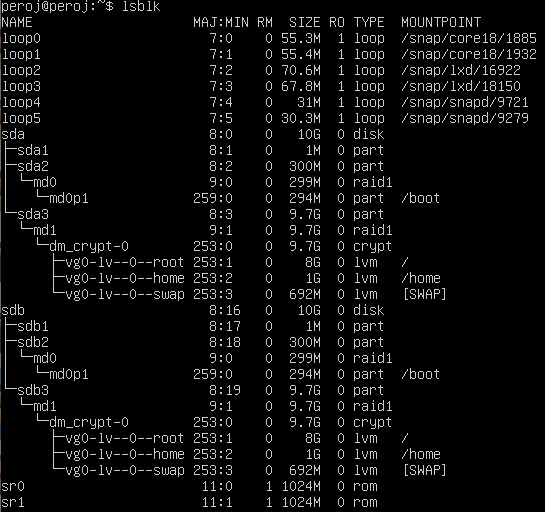




Primeramente, vamos a simular que mientras estamos trabajando se nos rompe uno de los discos duros del RAID de Ubuntu (y después veremos cómo recuperarlo). Para ello (y con la máquina apagada) debemos habilitar la opción de *Conectable en caliente* desde la configuración del almacenamiento de la máquina virtual para poder quitar el disco duro mientras la máquina está encendida.

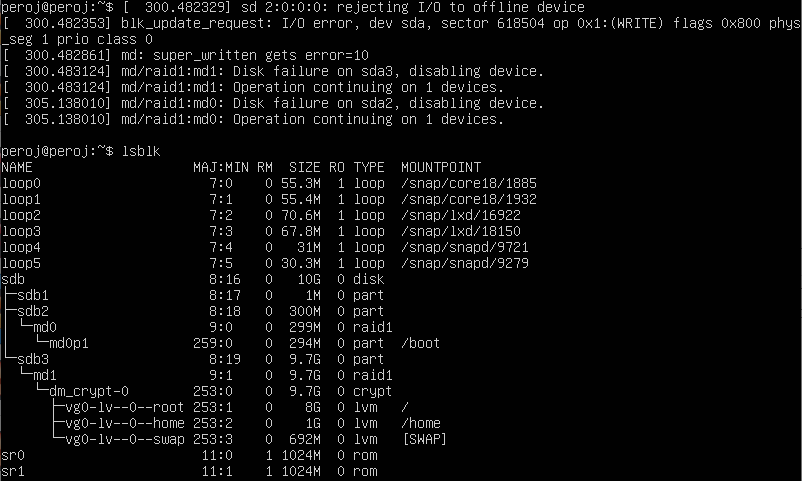


Configuración de la maquina antes de tocar nada.



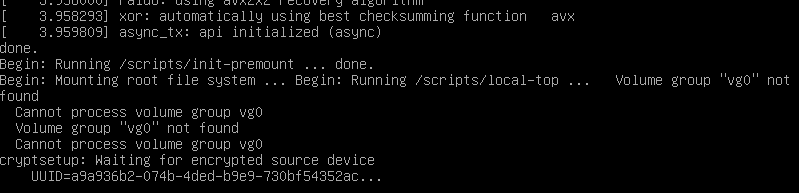
Iniciamos. Y una vez que se inicie, volvemos a la configuración y quitamos el disco duro (ControladorSATA).

Al quitarlo, nos da fallo en *sda3* y *sda2* (los que están en RAID) y nos dice que el sistema continúa únicamente con un disco duro (*sdb*).

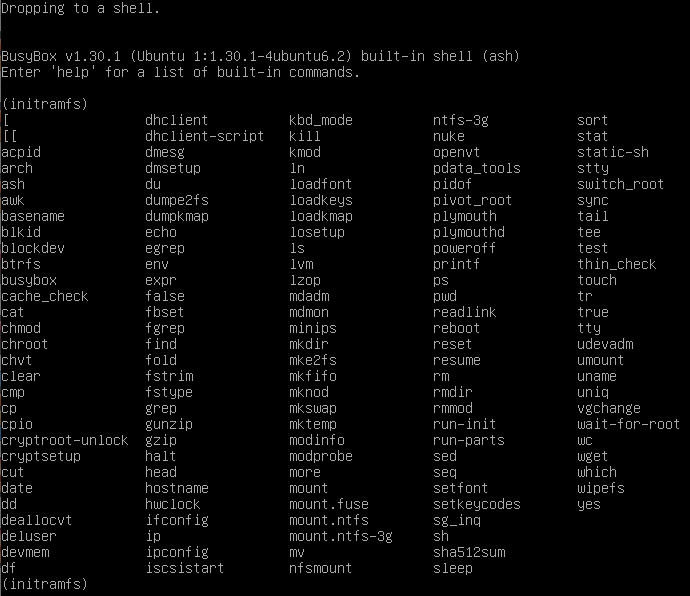


Como estamos en RAID (y tenemos lo mismo en el *sda* que en el *sdb*), si se nos rompe el disco duro a, el sistema sigue funcionando.

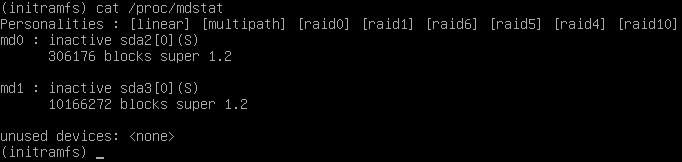
Pero, ahora bien, si restauramos a la instantánea anterior. Y en vez de quitarlo en caliente, lo quitamos ahora antes de iniciar. Y iniciamos el sistema con un disco duro. En este caso, no puede procesar el grupo de volúmenes y quedará esperando (intentando recuperar) hasta conseguir iniciar con un solo disco duro (tardará como mucho 5 minutos hasta que el sistema se da cuenta de que no puede recuperar la información).



En este caso, pasado el tiempo de espera, se iniciará por defecto una herramienta del sistema llamada ***initramfs***, que es un sistema de ficheros que se monta en memoria RAM para que se intente arreglar el error de inicio (dentro de esta herramienta se cambia la distribución de teclado, por ejemplo, el \_ es la /). Contiene las herramientas justas para gestionar el sistema para arreglar el sistema cuando resulta dañado (podemos verlas pulsando dos veces el *tab*). En este caso, sabemos que falla, pero en otros casos con esta herramienta no bastará.

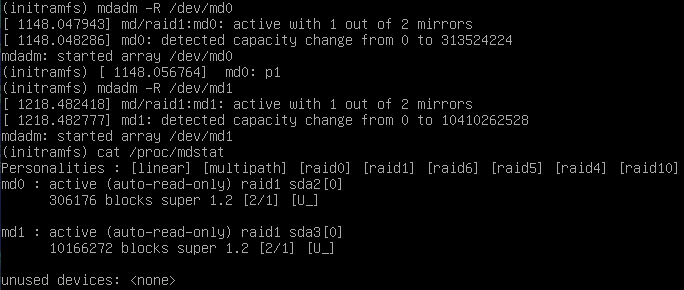


También podemos ver la función ***dmesg*** que nos da información sobre el sistema, sobre todo el sistema, a nivel de hardware también te da la información de todos los mensajes que el hardware devuelve.

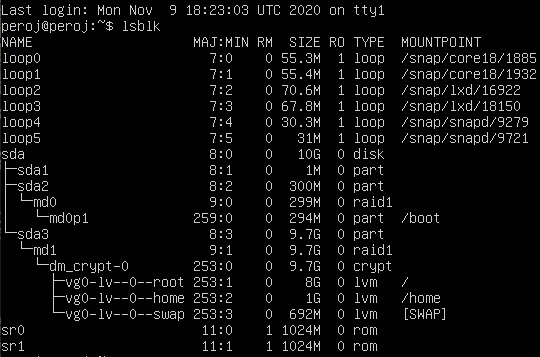
Como sabemos que lo que falla el RAID, lo primero que vamos a hacer es ver el fichero ***/proc/mdstat*** que contiene la información del RAID. Vemos que tanto md0 como md1 están inactivos porque no se han podido arrancar.

Como hemos visto antes, cuando hemos quitado el disco duro en caliente en Ubuntu, con un disco duro se puede arrancar el sistema y así el RAID continuar. Para forzar esta situación y que el sistema continúe arrancando con un único RAID usamos ***mdadm -R /dev/md0*** y ***mdadm -R /dev/md1*** (el - está en la ¿).

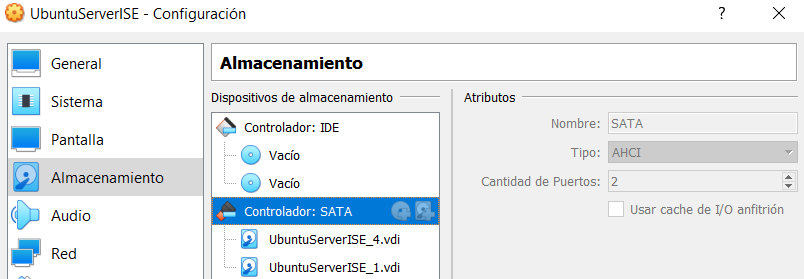
Y vemos que lo primero que nos dice es que el RAID1 md0 está activo con uno de los dos discos duros de espejo. Y volvemos a visualizar el fichero para ver qué se han hecho efectivos los cambios, ya que en este fichero es donde podemos ver cuántos discos duros están caídos en los corchetes **[*2/1]***, de dos está activo uno, y en los corchetes ***[\_U]*** donde la \_ indica que está caído el primero, es decir, *sda* y la U (up) indica que el segundo disco duro(*sdb*) está activo.



Hacemos ***exit*** para salirnos del *initramfs*. Y vemos que nos pide la contraseña del volumen encriptado, por lo que ya ha conseguido organizarse para iniciar con un disco duro. Una vez recuperado el sistema sin perder información debemos arreglar el RAID. Comprobamos cuando entramos al sistema con ***lsblk*** que ahora estamos con un disco duro y que *sdb* ha pasado a ser nuestro *sda*.

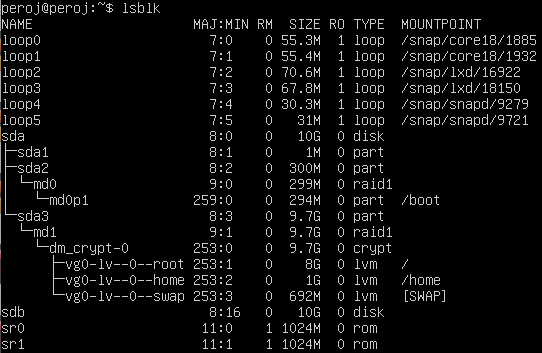


Ahora deberíamos conectar nuestro disco duro nuevo para reemplazar el estropeado. Esto lo vamos a simular insertando en caliente en el sistema un nuevo disco duro.



Apagamos a máquina por si hubiera problema por incompatibilidad en los puertos SATA de los discos duros. Debemos asegurarnos de que el disco duro que contiene la información está en el puerto SATA 0 para que el sistema lo pueda tomar de arranque (y el nuevo en el SATA 1). Iniciamos la máquina de nuevo y debería arrancar sin problemas con un único disco duro.

Con ***lsblk*** vemos que el disco duro nuevo lo tenemos vacío (con 10GB). Con ***mdadm --add /dev/md0 /dev/sdb*** podíamos añadir un RAID a un disco duro, pero el problema es que tenemos dos RAID y evidentemente no se puede hacer esto así, primero, porque el disco duro *sda* tiene 3 particiones (*sda1* 1MB para el grub del sistema*, sda2* 300MB para el RAID de *boot* y *sda3* 9’7GB para el RAID “gordo” del sistema). Por esto no podríamos usar ese comando para hacer esto porque le estaríamos diciendo que le añada a *md0* todo el disco duro *sdb*.

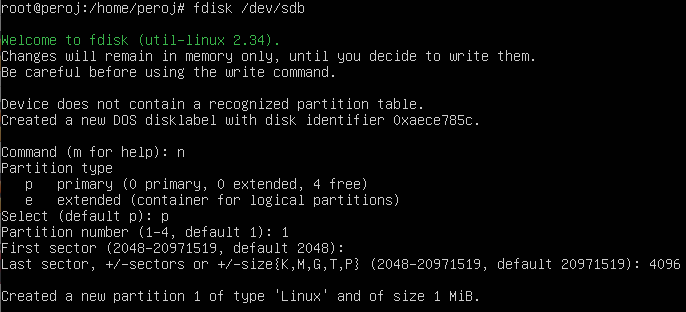


Entonces debemos de hacer las particiones de *sdb* a mano, hacer 3 particiones y después añadir a cada partición el RAID que les corresponde. Esto lo hacemos con la herramienta ***fdisk***.

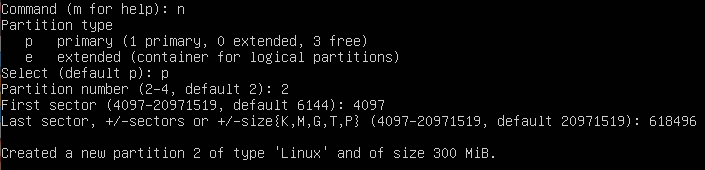
1º- Con ***fdisk /dev/sdb*** entramos a la consola de la herramienta para formatear *sdb*.

2º- Para añadir una nueva partición pulsamos ***n*.** Para que sea una partición primaria ***p***. Y establecemos el número de la partición ***1***.

3º- Para lo siguiente debemos usar una regla de tres para calcular los sectores (2048-20971519). Algo así como, si 20971519 total de sectores --> 10GB.... Entonces, empezando en 2048 debemos recordar que la primera partición tiene 1MB, que es más o menos 2048 sectores, entonces habría que pasar de 2048 a 4096 (ya que si empezamos en 2048 el último sector sería el 4096).

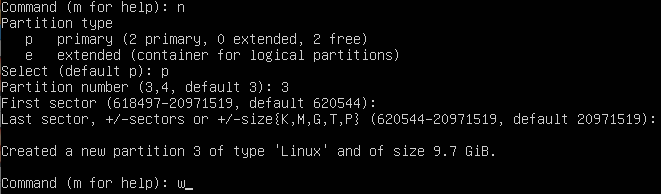


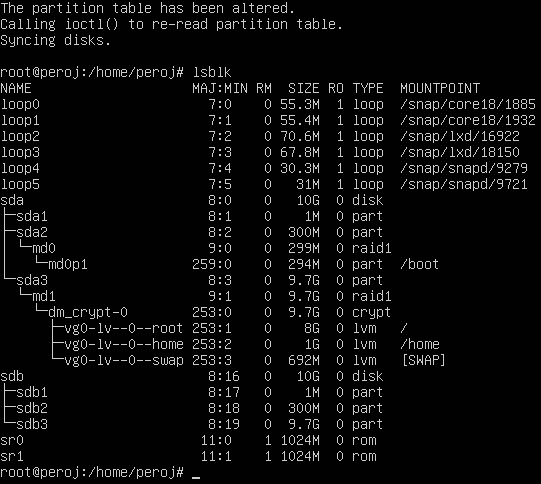
4º)- Si no fuésemos a hacer más particiones deberíamos guardar la creada hasta ahora pulsando ***w***. Pero como vamos a seguir, volvemos al paso 1º para crear *sdb2* y en este caso asignaremos el número de partición a ***2*** (que era de 300MB). Entonces como nos hemos quedado en el 4096, debemos empezar en el 4097 los sectores (eliminando así el mega que se van poniendo entre sector y sector por defecto).



5º) Del mismo modo creamos la partición ***3***. Dejando por defecto el apartado de los sectores, para así comprobar cuando se cree que tiene exactamente los 9’7GB que le corresponden.

6º) Pulsamos a ***w*** para crear las particiones a nivel físico y salir de ***fdisk***.

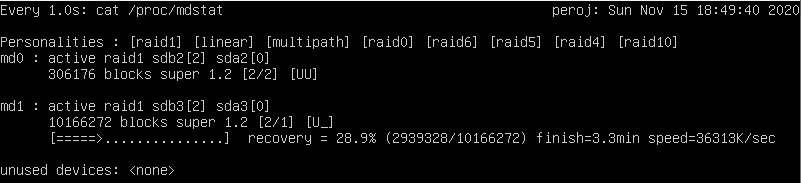




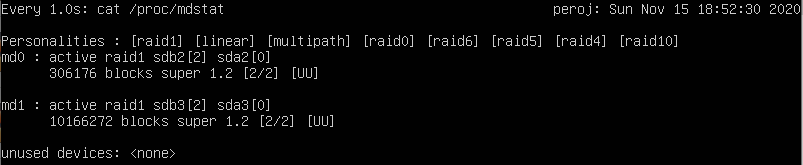
Ahora ya podemos sincronizar los RAID ya que ahora le indicamos a ***mdadm*** que a *md0* le añada *sdb2* y que a *md1* le añada el *sdb3*. Entonces para sincronizar el RAID de *boot* usamos ***mdadm --add /dev/md0 /dev/sdb2*** y para el RAID del sistema usamos ***mdadm --add /dev/md1 /dev/sdb3***.

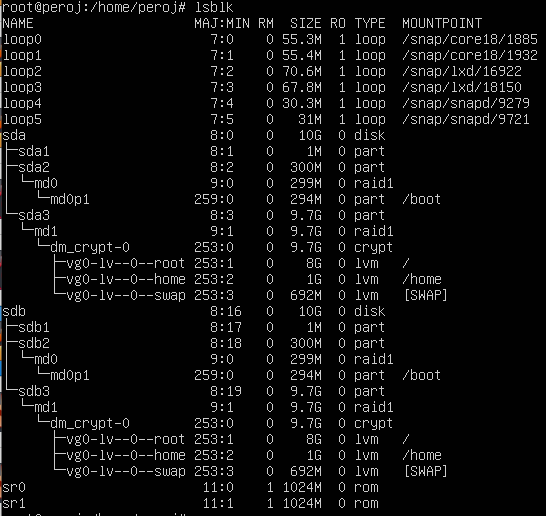


Una vez ejecutado esto, podemos ver cada segundo en el fichero de configuración de los RAID con ***watch -n 1 cat /proc/mdstat*** como se van restaurando el sistema y resincronizando los espejos.

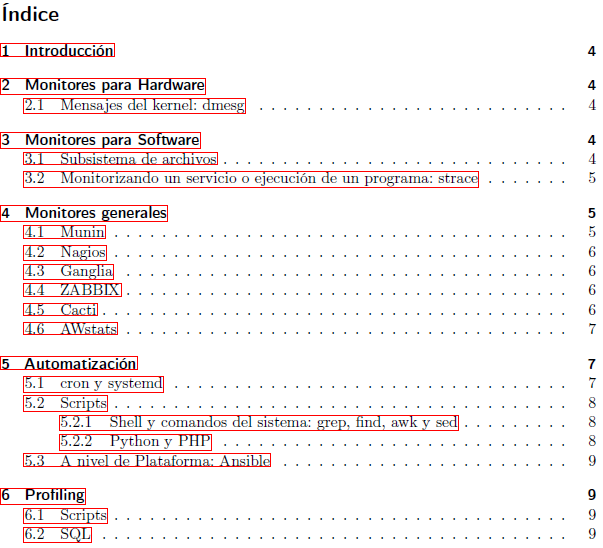


En cuanto acabe, ya tendríamos el sistema completamente recuperado con nuestro disco duro nuevo como si nada hubiese pasado.

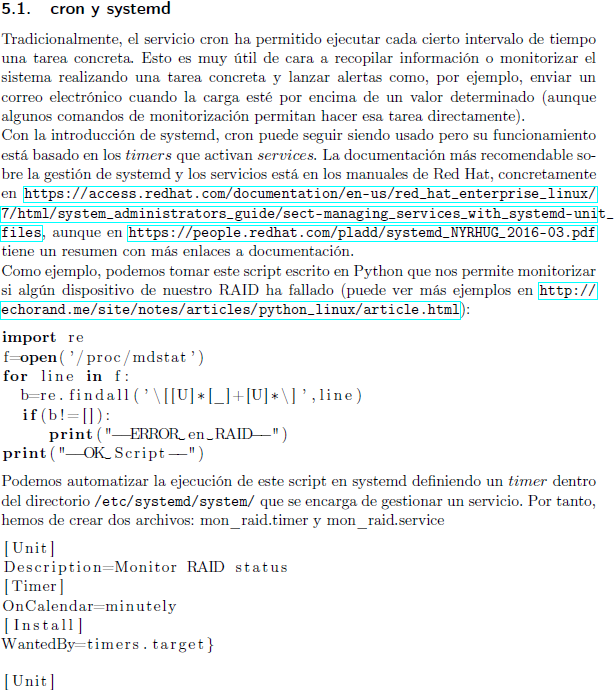


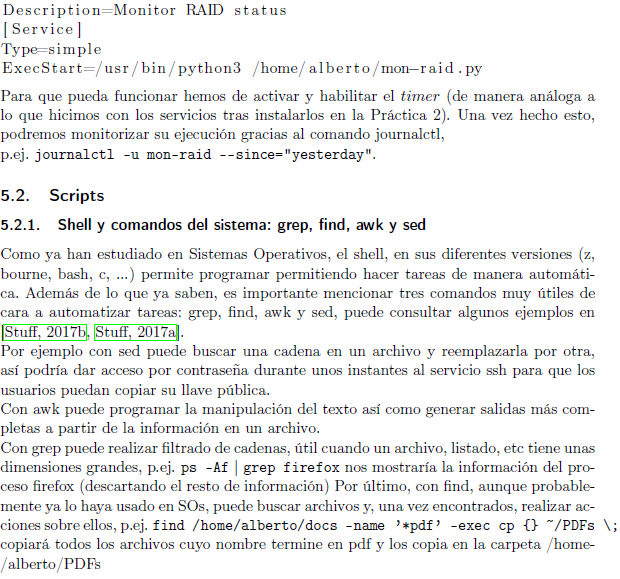


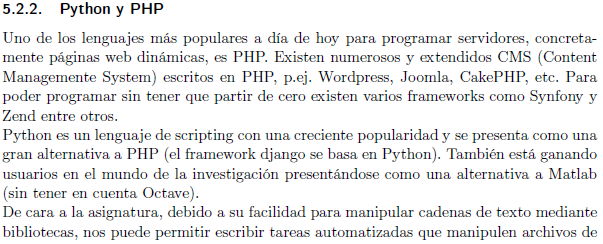
## LECCIÓN2:

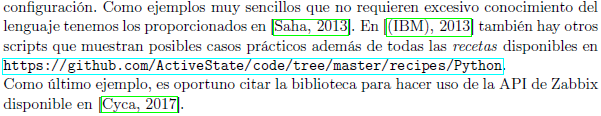


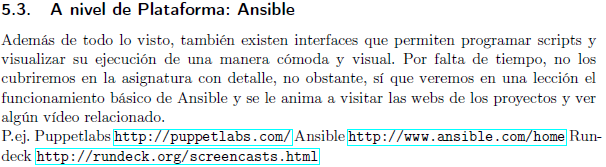
1. **Automatización**



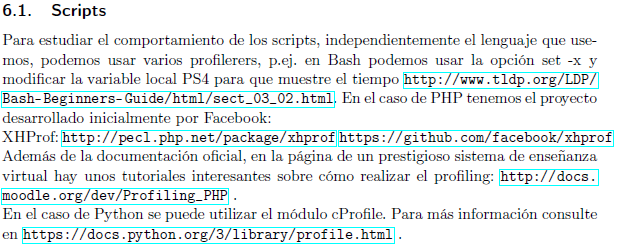


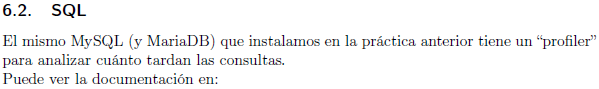


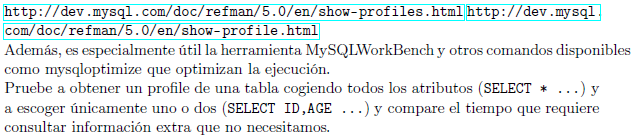




1. **Profiling**







En esta Lección vamos a ver cómo automatizar los procesos de la lección anterior creando scripts y servicios asociados a estos en Ubuntu.

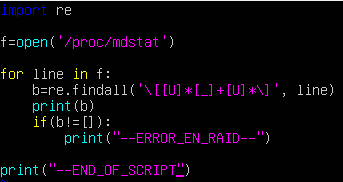
Como hemos visto, en el fichero ***/proc/mdstat*** está la información acerca de los RAID. Cuando uno de los discos duros del RAID está caido, aparece denotado por una ***\_*** y si están funcionando correctamente por una ***U***. Entonces vamos a ver cómo hacerle un “Profiling” (poder automatizar este tipo de cosas y perfilarlas) al proceso anterior, es decir, cómo mantener controlado en nuestro sistema el estado de los RAID (en este caso).

Primeramente, haremos un script en Python (mostrado en el apartado 5), donde importamos la librería para expresiones regulares para detectar la \_ en el contenido de ***/proc/mdstat***, lo que devolverá que existe un error en el RAID. Está opción es buena para usar puntualmente, pero si queremos automatizarlo tenemos dos opciones:

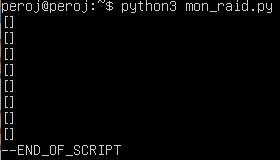
* o bien, modificando el fichero ***crontab*** donde podemos indicar líneas de ejecución, el periodo de ejecución y el usuario que lo ejecutará.
* o bien, crear un servicio que ejecute esto cada minuto, usando ***timer***. Tenemos que definir un *timer*dentro de la carpeta **/etc*/systemd/system/*** y debemos crear también un servicio asociado a este *timer* para que el sistema automáticamente cuando lea este *timer* busca el servicio análogo al mismo. Y una vez definido, damos de alta el servicio como siempre lo hemos hecho con ***systemctl enable*** y ***start timernuevo***. De esta forma, cada minuto vamos a saber cómo está el RAID y podemos monitorizar (o visualizar) su ejecución con el ***journalctl,*** que va mostrando todos los mensajes del sistema desde un momento determinado.

Antes de encender la máquina con UbuntuServer debemos cerciorarnos en *Configuracion/Almacenamiento* que el disco duro *sdb* sea *Conectable en caliente* (para hacer comprobaciones posteriores). También tomamos una instantánea del estado actual de la máquina.

Creamos el script con ***vi mon\_raid.py*** como hemos visto (guardar y salir ***:wq***). La expresión regular ‘\[[U]\*[\_]+[U]\*\]’ busca en el fichero si hay una \_ sola o detrás o delante o entre dos U, básicamente, si encuentra cualquier guion bajo.



Comprobamos que funciona correctamente ejecutándolo con ***python3 mon\_raid.py*** y vemos que como no ha encontrado nada, sale todo vacío.

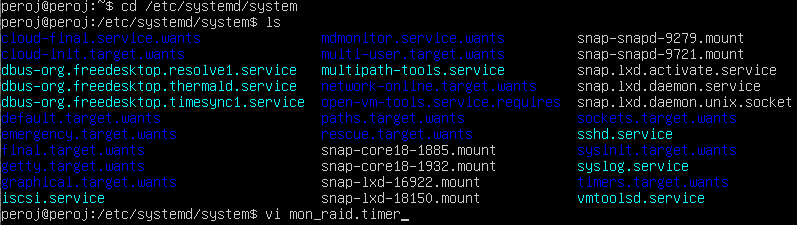


Ahora vamos a desconectar el segundo disco duro *sdb* y cuando la máquina lo haya detectado volvemos a ejecutar el script y vemos que encuentra dos errores, lógicamente, ya que tenemos dos RAID.

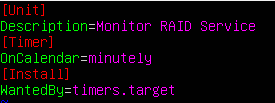


Una vez visto el funcionamiento, vamos a crear un ***timer*** que minuto a minuto ejecute el script para checkear los RAID. De esta forma, nosotros tenemos información detallada minuto a minuto de si ha pasado algo.

Ejecutamos ***sudo su*** y nos movemos a la carpeta ***/etc/systemd/system*** y crearemos dos ficheros, el ***.timer*** y el ***.service*** asociados a nuestro script.



Primero creamos el *timer* con ***vi mon\_raid.timer*** :

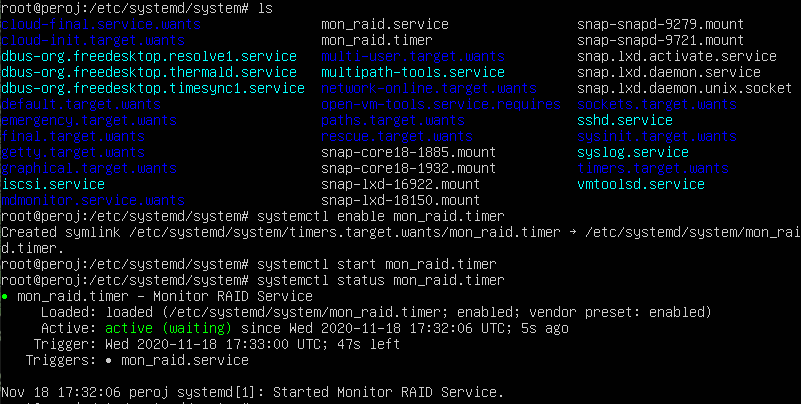


El *Unit* indica que vamos a crear una unidad, es decir, un servicio. Le damos la descripción. Le ponemos el target que es donde le decimos que se actualizará esto minuto a minuto y por último en el *Install*, le indicamos que lo instale como *Timer* y que esté pendiente minuto a minuto de esto.

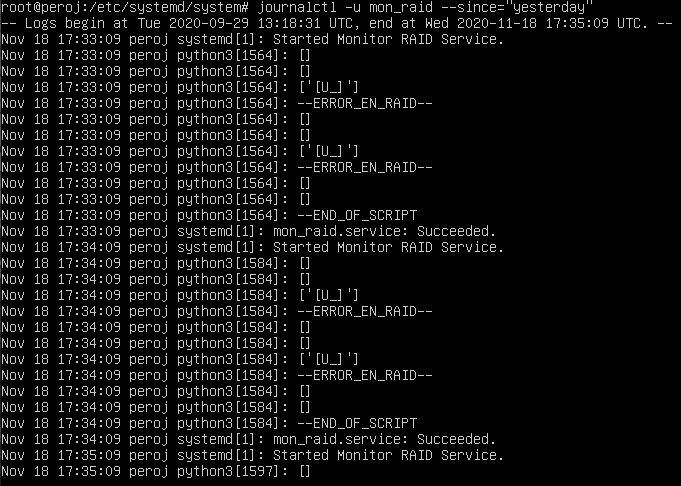
Después creamos el *servicio* con ***vi mon\_raid.service*** (con el mismo nombre que el *timer*):



Vemos que ya lo tenemos definido con ***ls*** y con ***systemctl enable mon\_raid.timer*** le decimos al sistema que habilite el *timer* (internamente el sistema buscará el *servicio*) y con ***systemctl start mon\_raid.timer*** que inicie el servicio. Comprobamos que está *running* con ***systemctl status mon\_raid.timer*** .

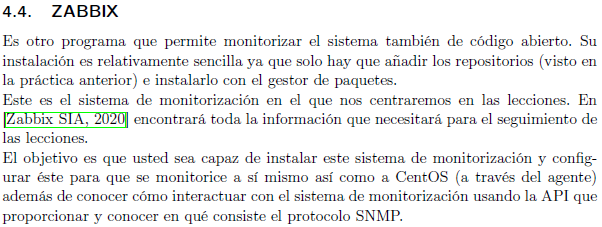


Para ver lo que está monitorizando el servicio usamos ***journalctl*** que recoge los mensajes del sistema (incluidos los mensajes lanzados por los servicios) pero deberemos de filtrarlo para ver concretamente el nuestro y por ejemplo desde ayer, con el comando siguiente ***journalctl -u mon\_raid --since=”yesterday”***.

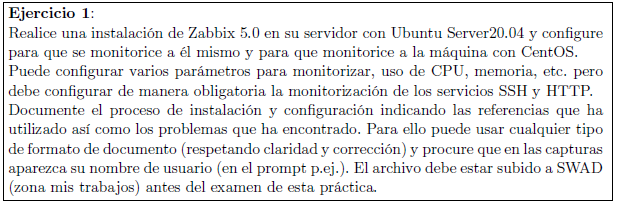


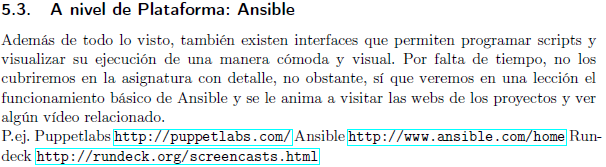
Como vemos el servicio está reconociendo correctamente el error en los RAID cada minuto.

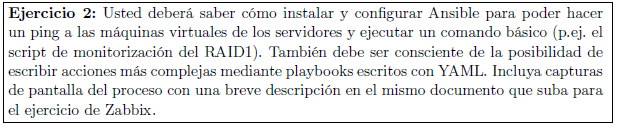
## MEMORIA: Zabbix



Software de monitorización que crea un frontend en el navegador que permite ver de forma visual los envíos de los agentes (la información que envían los servidores monitorizados).







Pasos:

1. Instalar y configurar Ansible en Ubuntu y añadimos las IPs de los servidores CentOS y Ubuntu al fichero de configuración (también podemos crear un grupo y en el momento de ejecutarlo en vez de ejecutar ***ansible*** con ***all*** ponemos el nombre del grupo).
2. Deberemos hacerlo con clave pública-privada y cambiar al puerto por defecto (22) de ssh. Deberemos regenerar las claves pública-privada, de Ubuntu a sí mismo y de Ubuntu a CentOS.
3. Cuando Ansible nos devuelva el ping correctamente a los servidores, deberemos crear en CentOS (o enviar por ssh) el fichero ***mon\_raid.py*** y simplemente si tenemos ping y el fichero está en CentOS, debería funcionar correctamente.

El software Ansible, permite automatizar el profiling anterior (la monitorización del RAID), porque a Ansible se le añade tantas máquinas como se tengan (en nuestro caso 2 IPs, Ubuntu y CentOS) y le dice a las máquinas con el script que hemos creado que devuelvan cómo están los RAID (en este caso) en cada máquina, sin necesidad de ir consultándolo servidor por servidor, directamente con Ansible, desde el servidor “maestro” puedes ver la información del resto de servidores.

Con Ansible podemos automatizar monitorizaciones a lo largo de multitud de servidores.

Teniendo Ansible instalado tenemos que configurar que monitorice (automatice el script) tanto Ubuntu como CentOS. Realmente es simplemente añadirle los host (tanto Ubuntu como CentOS) para que Ansible sea capaz de verlos y poder hacer este tipo de automatizaciones, Ansible, más que para monitonizar se utiliza para automatizar. Ansible en definitiva nos permite automatizar tareas en muchos servidores.

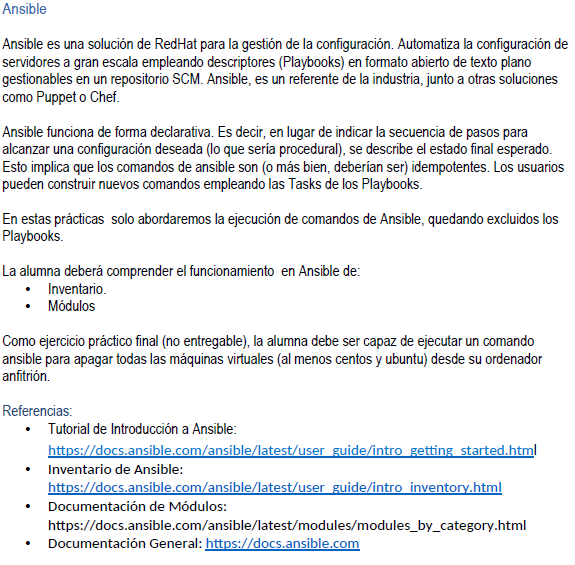
Una vez que tenemos los host añadidos, podemos comprobarlo desde Ubuntu con ***ansible all -m ping -u peroj*** . Como Ansible hace la conexión por ssh, debemos estar (dado que no tenemos acceso para root por ssh en las máquinas) y poner el mismo usuario en todos los servidores para hacerle ping, en nuestro caso tenemos el mismo tanto en CentOS como en Ubuntu (en un caso real deberemos tener un usuario para estas cosas). Comprobamos que se hace ping correctamente.

Ahora debemos crear el script ***mon\_raid.py*** tanto en CentOS como en Ubuntu (y en la misma ubicación). También podemos enviarlo por ssh desde Ubuntu para CentOS.

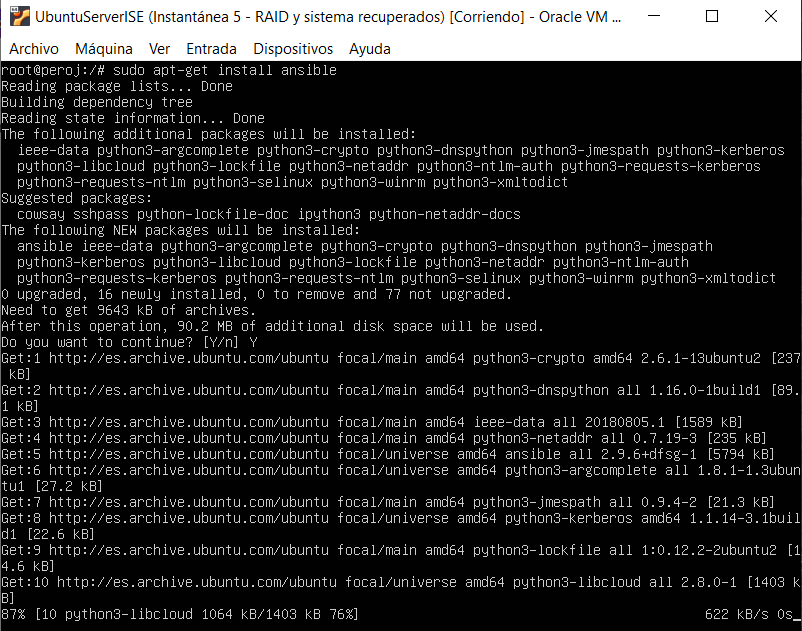
Ahora podemos ejecutar con Ansible en todos los servidores y recibir el resultado en nuestra máquina con ***ansible all -a “python3 /home/peroj/mon\_raid.py”*** (si el fichero a ejecutar no estuviera en la misma ubicación en las dos máquinas, esto deberíamos hacerlo con un script de configuración). Comprobamos y vemos como en Ubuntu nos devuelve el error en los RAID pero en CentOS no, dado que en CentOS funciona correctamente.

Ansible nos permite una automatización brutal si tenemos muchos servidores, incluso después de esto podríamos coger con otro script qué RAIDs están fallando y arreglarlo en todas las máquinas a la vez ejecutando dicho script (siempre y cuando fuese el mismo problema en todas y se pudiese subsanar así).

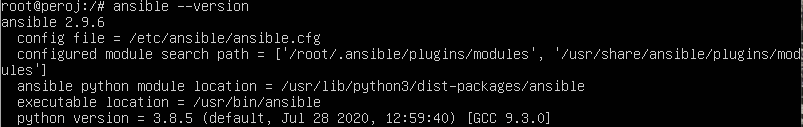
También debemos saber que se pueden hacer acciones más complejas mediante los ***playbooks*** (libros de juegos) que son los scripts de Ansible escritos en lenguaje YAML.



1º) Instalamos ***ansible*** en UbuntuServer con ***sudo apt-get install ansible***. Comprobamos la versión con ***ansible --version***.



Y vemos que usa la versión 3.8.5 de Python, es decir, el módulo *python3*.

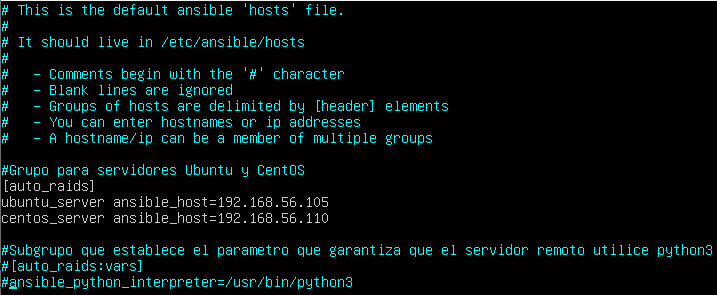


2º) Ahora deberemos irnos al archivo de inventario con ***vi /etc/ansible/hosts*** donde indicaremos los host con las IPs de los servidores con los que queremos tratar, en nuestro caso, tanto de Ubuntu como de CentOS (podemos ver varios ejemplos de configuración comentados en el archivo).

Este archivo se suele utilizar también para configurar variables que serán válidas sólo para hosts o grupos específicos, a fin de usarse dentro de los playbooks y las plantillas. Algunas variables también pueden afectar la forma en que se ejecuta un playbook, como la variable *ansible\_python\_interpreter* que veremos a continuación.



Creamos un grupo de servidores llamado *auto\_raids* donde incluiremos las IPs tanto de CentOS como de Ubuntu. También creamos (si es necesario) un subgrupo de este llamado *auto\_raids:vars* donde establecemos el parámetro ***ansible\_python\_interpreter=/usr/bin/python3*** (descomentar solo si es necesario).

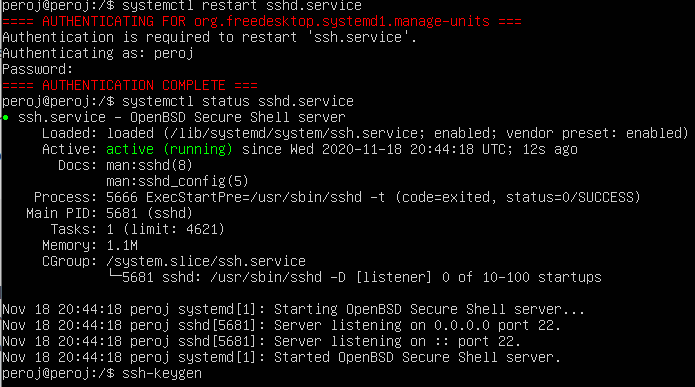


Comprobamos que se han añadido los host correctamente al inventario con ***ansible-inventory --list -y***.

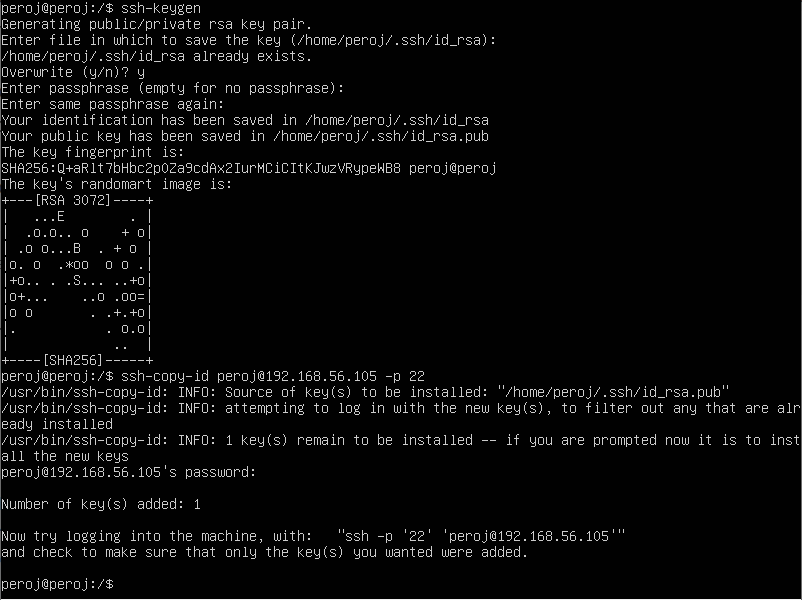


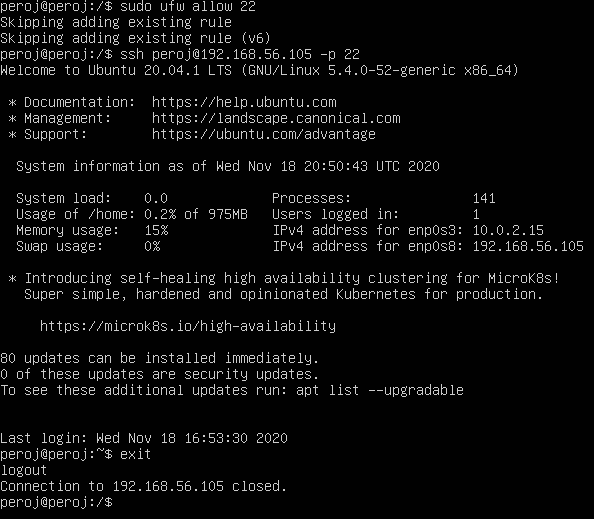
3º) Antes de probar conexión, debemos cambiar el puerto de las máquinas al puerto por defecto (22), con lo que esto conlleva, en Ubuntu deberemos simplemente avisar al firewall, pero en CentOS también deberemos actualizar el SELinux y ya después, volver a generar los pares de claves pública-privada, tanto de Ubuntu consigo mismo como con CentOS (ya que *ansible* se conecta por ssh).

-En Ubuntu: Abrimos ***/etc/ssh/sshd\_config*** y cambiamos el puerto al *Port 22* y el parámetro *Password Autentication yes* para poder generar las nuevas claves (después lo cambiamos otra vez)*.* Después de guardar el fichero, reiniciamos el servicio con ***systemctl restart sshd.service*** para aplicar los cambios y lo vemos con ***systemctl status sshd.service***. Nos aseguramos que el puerto 22 está abierto en el firewall ***ufw allow 22*** *(*que modifica las *iptables* para que nos permita comunicación por este puerto).



Primero generamos la clave en el cliente (Ubuntu) con ***ssh-keygen*** . Se la mandamos al servidor (Ubuntu) con ***ssh-copy-id*** [***peroj@192.168.56.105***](mailto:peroj@192.168.56.105) ***-p 22,*** metemos la contraseña de Ubuntu y vemos que se ha añadido una clave. Finalmente, comprobamos haciendo login por ***ssh*** [***peroj@192.168.56.105***](mailto:peroj@192.168.56.105) ***-p 22***, ya no nos pide la contraseña.

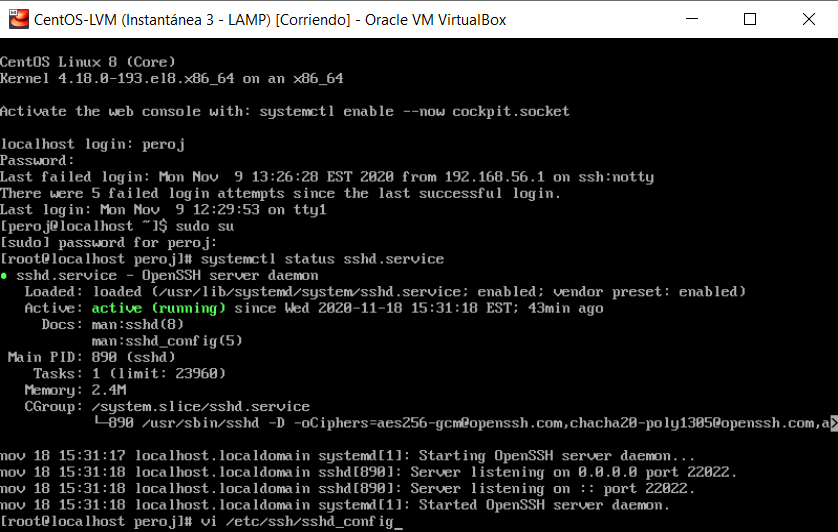




Ya tenemos conexión por ssh de Ubuntu consigo misma a través del puerto 22.

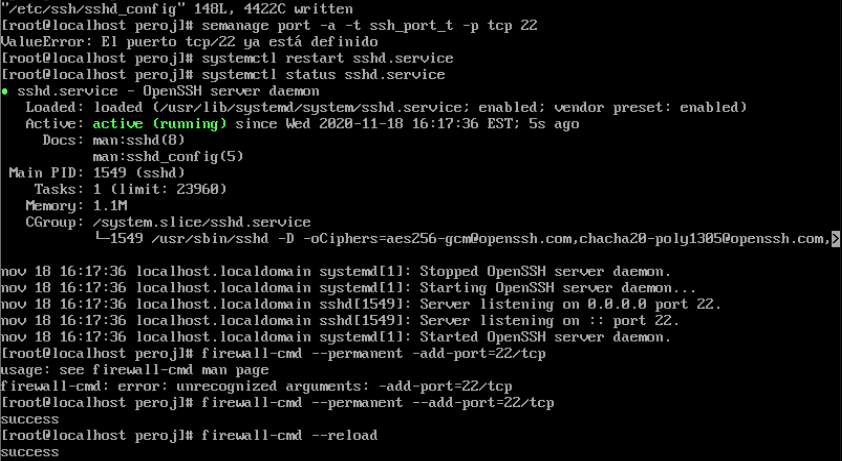
En Ubuntu debemos recordar poner el parámetro de ***/etc/ssh/sshd\_config*** *AutenticationPassword no*.

-En CentOS: Haremos “el mismo” proceso anterior para configurarle el ssh server y podernos conectar desde UbuntuServer. Comprobamos que esta dado de alta el servicio con ***sudo systemctl status sshd.service***. Vamos a cambiarle el puerto 22 que trae por defecto por el 22022 y el *PasswordAutentication* a *yes* modificando el fichero ***sudo vi /etc/ssh/sshd\_config.***



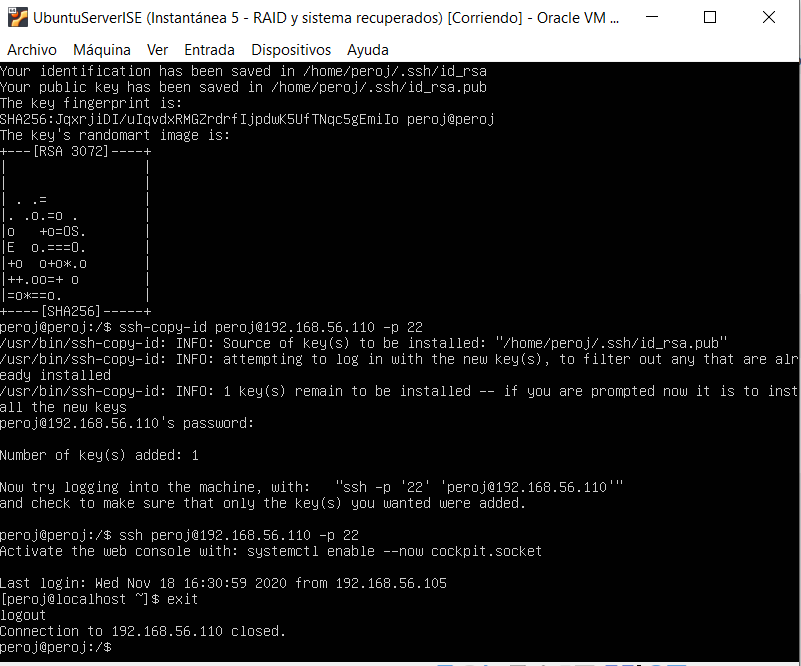
Una vez hecho esto, debemos avisar al sistema de seguridad de Linux SELinux de que vamos a cambiar el puerto con la orden ***semanage port -a -t ssh\_port\_t -p tcp 22*** y reiniciamos el servicio con ***sudo systemctl restart sshd.service***.

También avisamos al cortafuegos de que permita conexiones por ese puerto con ***sudo firewall-cmd --permanent --add-port=22022/tcp*** y recargamos las reglas con ***sudo firewall-cmd --reload***.

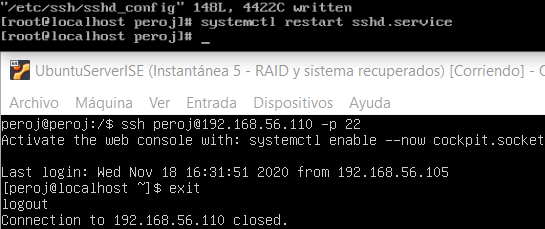


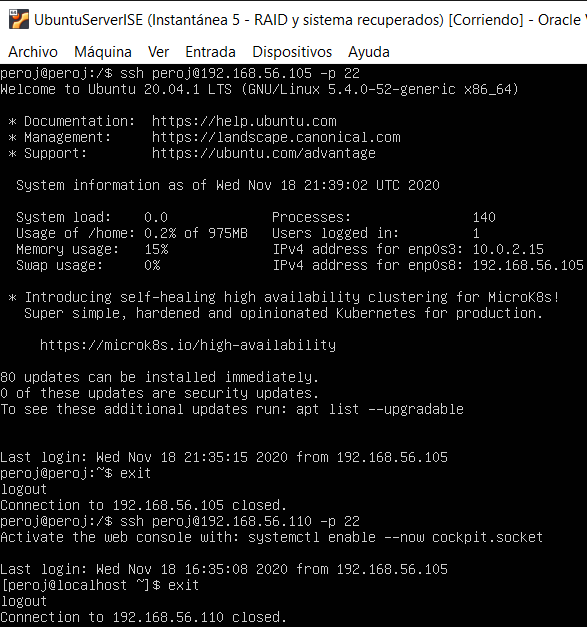
Ya si intentamos conectarnos desde Ubuntu a CentOS con ***ssh*** [***peroj@192.168.56.110***](mailto:peroj@192.168.56.110) ***-p 22*** ya podemos sin problema.

Ahora debemos enviar desde Ubuntu la clave pública-privada a CentOS para que no sea necesaria la contraseña.

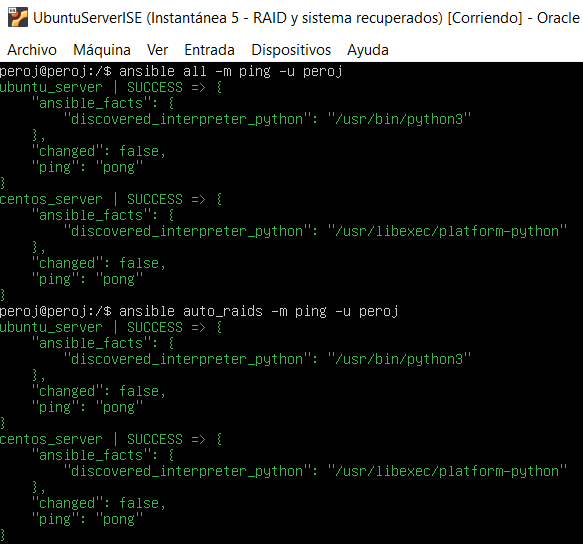


Y finalmente, le cambiamos del fichero de configuración de CentOS ***sudo vi /etc/ssh/sshd\_config*** el *PasswordAuthentication* a no*,* para que solo se pueda conectar un usuario que previamente esté vinculado con una clave pública-privada. Y reiniciamos el servicio con ***sudo systemctl restart sshd.service***.





4º) Una vez que tenemos correctamente configuradas las conexiones ssh procedemos a comprobar que Ansible nos devuelve el ping correctamente a los servidores con ***ansible all -m ping -u peroj*** o ***ansible auto\_raids -m ping -u peroj.***



5º) Ahora debemos crear el script ***mon\_raid.py*** en CentOS (y en la misma ubicación que en Ubuntu). También podemos enviarlo por ssh desde Ubuntu para CentOS.

Ahora podemos ejecutar con Ansible en todos los servidores y recibir el resultado en nuestra máquina con ***ansible all -a “python3 /home/peroj/mon\_raid.py”*** (si el fichero a ejecutar no estuviera en la misma ubicación en las dos máquinas, esto deberíamos hacerlo con un script de configuración o playbook). Comprobamos y vemos como en Ubuntu nos devuelve el error en los RAID pero en CentOS no, dado que en CentOS funciona correctamente.

