SSH

SSH = Secure Shell (Autenticidad, encriptación, integridad)

Importante: Diferenciar entre ssh y sshd
Archivos: /etc/ssh/ssh_config y /etc/ssh/sshd_config

Diferencias Ubuntu/Rocky:

Ubuntu	Rocky
- Nomenclatura: para el servidor se le puede llamar ssh o sshd.	- Nomenclatura: sólo sshd.
- Hay que instalar ssh.	- SSH ya viene instalado por defecto.
- Permite al root conectarse por ssh con contraseña por defecto.	- Por defecto no se permite al root conectarse con contraseña.
- Firewall ufw desactivado	- Firewall firewall-cmd activado

CONFIGURACIÓN SSH EN UBUNTU:

Usamos el comando apt para conectarnos con los servidores de paquetes. Buscamos ssh:
apt search ssh

Instalamos openssh:
sudo apt update
sudo apt install openssh-server

Comprobamos si el instalador ya ha iniciado el servicio:
ps -Af | grep sshd : Supuestamente debería salir que lo ha iniciado.
Si no lo ha iniciado podemos hacer:
sudo /etc/init.d/ssh start o sudo systemctl start sshd

Probamos invocar al cliente y conectarnos con el servidor:
ssh localhost : > yes, > contraseña
Ctrl^D : salimos

En home/ tenemos un nuevo directorio home/user/.ssh/. Nos movemos a él:
cd .ssh/
ls : vemos un archivo known_hosts que contiene el fingerprint del servicio
cat known_hosts
Por tanto, si reinstaláramos el sistema y el fingerprint cambiase, daría una alerta y no nos dejaría conectarnos.

Sabemos que el usuario que está en todas las máquinas es root, por tanto debemos impedir el acceso del root modificando un archivo. Si quisiéramos acceder al root, tendríamos que usar un usuario como pasarela que cambiará de usuario y pasará a tener privilegios.
sudo vi /etc/ssh/sshd_config
Aquí, buscamos #PermitRootLogin y ponemos no (por defecto está prohibit-password: podemos acceder a root con cualquier mecanismo distinto a la contraseña).

systemctl restart sshd : Reiniciamos el servicio cuando cambiamos la configuración

Ahora vemos si podemos acceder como root desde otra terminal de otra máquina.
ssh -l root 192.168.56.105 o ssh root@192.168.56.105 : (la IP es la de la MV de ubuntu) vemos que no tenemos permiso.

ssh -l root 192.168.56.105 -v : -v nos sirve para depurar, nos da todos los detalles.

CONFIGURACIÓN SSH EN ROCKY:

El servicio ssh ya viene instalado por defecto. Veamos si está arrancado:
ps -Af | grep sshd o systemctl status sshd : Sí está running

ssh localhost : probamos conexión desde dentro → funciona correctamente

Desde otra terminal de fuera probamos la conexión:
ssh elsarm@192.168.56.106 : Nos deja acceder bien
ssh root@192.168.56.106 : También nos deja acceder con contraseña!!

Editamos el archivo de configuración para impedir acceso al root:
sudo vi /etc/ssh/sshd_config : Buscamos #PermitRootLogin y ponemos no
systemctl restart sshd

Desde otra MV comprobamos si ya podemos acceder como root con contraseña y vemos que no nos deja:
ssh root@192.168.56.106

YA TENEMOS CONFIGURADO SSH EN UBUNTU Y ROCKY Y TENEMOS DESHABILITADO EL ACCESO COMO ROOT. (Es lo mínimo que hay que hacer cuando usamos sshd)

El puerto de SSH es el 22. Ahora queremos cambiarlo para que no sea trivial intentar acceder.

CAMBIO A PUERTO 22022 EN ROCKY:

Para ello modificamos el archivo de configuración usando el comando sed (stream editor). Este comando nos permite sustituir una cadena por otra en todo el documento.
sed s/'#Port 22'/'Port 22022'/ -i /etc/ssh/sshd_config : Cuidado, hay que descomentar
systemctl restart sshd : Nos da un error al iniciar el servicio
journalctl -xe : Nos indica que hay un error al intentar enlazar a ssh el 22022 por permiso denegado.

Editamos el archivo de configuración con vi y vemos que en la cabecera hay una línea que nos dice que si queremos cambiar el puerto en un sistema que ejecuta SELinux, hay que informar a SELinux sobre este cambio. Esto lo hacemos con el comando semanage.

dnf provides semanage (si no lo tenemos instalado) : Instalamos el paquete policycoreutils-python … con dnf

semanage port -l | grep -i sshd : listar los puertos relacionados con ssh
semanage port -a -t ssh_port_t -p tcp 22022 : Añadir puerto 22022 de tipo ssh con protocolo tcp.
systemctl restart sshd o systemctl reload sshd : Reiniciamos el servicio
Es mejor usar reload: "Reload will tell the service to reload its configuration files, but keep the same process running. Restart tells it to shut down entirely, then restart."

Vemos si nos podemos conectar por ssh desde dentro usando el nuevo puerto desde la MV Rocky:
ssh localhost -p 22022
Ctrl^D

Vemos si nos podemos conectar desde fuera desde otra máquina:
ssh elsarm@192.168.56.106 -p 22022 -v : Error: no route to host

El problema es el cortafuegos, hay que configurarlo.
Rocky → firewall-cmd

En la máquina de Rocky añadimos el puerto:
sudo firewall-cmd --state : Vemos que está running
sudo firewall-cmd --list-all : lista todas las reglas que están aplicándose. Podemos ver los puertos y los servicios.
Para añadir el puerto y guardar los cambios tenemos varias opciones:
1. sudo firewall-cmd --add-port 22022/tcp --permanent : Si añadimos el puerto de forma permanente, al recargar el firewall o reiniciar máquina, el puerto se abrirá, pero en el momento en el que ejecutamos este comando no se abre el puerto. Por ello, podemos abrirlo ejecutando:
sudo firewall-cmd --add-port 22022/tcp (al ejecutarlo abre el puerto pero al recargar el comando se cierra, pero como hemos ejecutado lo anterior con --permanent, al recargar sí que se abre)
o directamente recargando el comando:
sudo firewall-cmd --reload
2. sudo firewall-cmd --add-port 22022/tcp : Añadir el puerto y abrirlo directamente:
sudo firewall-cmd --runtime-to-permanent : Guardar los cambios de forma permanente

Si ahora volvemos a listar los puertos, vemos que se ha añadido el 22022:
sudo firewall-cmd --list-all

Si ahora nos intentamos conectar desde otra máquina de fuera, sí que nos deja:
ssh elsarm@192.168.56.106 -p 22022
Ctrl^D

También podemos añadir/borrar un servicio:
sudo firewall-cmd --add-service=mysql : análogo con delete
sudo firewall-cmd --list-services

CAMBIO A PUERTO 22022 EN UBUNTU:

Modificamos el valor del puerto en el archivo de config:
sudo vi /etc/ssh/sshd_config : Descomentamos la linea de Port 22 y cambiamos 22 por 22022.
systemctl restart sshd

Comprobamos si tenemos acceso desde otra máquina:

ssh elsarm@192.168.56.105 -p 22022 : Funciona bien, nos deja conectarnos
Ctrl^D

Esto ocurre porque el firewall de Ubuntu (ufw) aunque está instalado, por defecto está desactivado luego tenemos que activarlo.
sudo ufw status : está inactivo.
sudo ufw status verbose : me da más info
sudo ufw enable : ya está activo.

Comprobamos si tenemos acceso desde fuera:
ssh elsarm@192.168.56.105 -p 22022 : no nos deja conectarnos

sudo ufw allow 22022 : para que permita el tráfico con el puerto 22022.

Comprobamos si tenemos acceso desde fuera:
ssh elsarm@192.168.56.105 -p 22022 : Funciona bien, nos deja conectarnos
Ctrl^D

Podemos borrar puertos con delete:
sudo ufw delete PORT_NUMBER

PERMITIR ACCESO SIN CONTRASEÑA:

Para ello, como la comunicación por ssh entre dos máquinas va cifrada simétricamente, nosotros vamos a usar un cifrado asimétrico en el que el cliente tiene la llave privada y la llave pública la copiamos en nuestro servidor → AUTENTICACIÓN.
(La parte de encriptación se hace con cifrado simétrico por eficiencia)

EN ROCKY:

Primero generamos las llaves privadas y públicas. Para ello usamos el siguiente comando:
ssh-keygen : Dejamos los valores por defecto. Esto hace que en el usuario elsarm guarde la llave privada en la carpeta /home/elsarm/.ssh/.
ls /.ssh/ : El archivo id_rsa es la llave privada y id_rsa.pub es la llave pública que vamos a ir distribuyendo.

Ahora queremos copiar la llave pública en el servidor de ubuntu (la otra MV):
ssh-copy-id elsarm@192.168.56.105 -p 22022 (la ip es la de ubuntu) : > yes, > nos pide autenticarnos en la máquina de ubuntu, pero una vez que ya tengamos acceso sin contraseña podremos desactivarlo.

En el servidor de ubuntu aparece el archivo:
more .ssh/authorized_keys

Comprobamos:
ssh elsarm@192.168.56.105 -p 22022 : Hemos iniciado sesión dentro de ubuntuserver y no nos ha pedido contraseña (porque se ha autenticado con la llave privada).
Ctrl^D

Ahora pasamos a desactivar el acceso por contraseña para que solo se pueda autenticar con el cifrado asimétrico. Esto lo hacemos en el servidor de ubuntu.

EN UBUNTU:
sudo vi /etc/ssh/sshd_config : Descomentamos la línea PasswordAuthentication y cambiamos ‘yes’ por ‘no’.
systemctl restart sshd :reiniciar

EN ROCKY:
ssh elsarm@192.168.56.105 -p 22022 : Seguimos pudiendo conectarnos
Ctrl^D

DESDE OTRA TERMINAL DE FUERA (anfitrión):
ssh elsarm@192.168.56.105 -p 22022 : No nos deja (permission denied).

Ahora queremos copiar la clave pública en el anfitrión pero, como no podemos conectarnos con ssh, no podemos hacer ssh_copy_id. Para poder hacerlo:
1. revertimos los cambios en el archivo /etc/ssh/sshd_config para permitir acceso con contraseña
2. reiniciamos el servicio: systemctl restart sshd
3. hacemos ssh-copy-id elsarm@192.168.56.105 -p 22022 en el anfitrión
4. volvemos a deshabilitar el acceso con contraseña en el archivo de configuración
Ya sí podemos acceder desde el anfitrión:
ssh elsarm@192.168.56.105 -p 22022
Ctrl^D

Ahora vamos a hacer que solo se permita a unos usuarios concretos editando el archivo de configuración.

Primero creamos un nuevo usuario en la máquina de Rocky y de Ubuntu (hay que hacerlo en las dos para que funcione):
sudo adduser -m nico
sudo passwd nico : ponemos contraseña

En Rocky:
ssh 192.168.56.105 -l nico -p 22022 : no nos deja porque no tenemos activado el acceso por contraseña y no tenemos copiada en ubuntuserver la clave pública de nico.
1. revertimos los cambios en el archivo /etc/ssh/sshd_config de servidor de Ubuntu para permitir acceso con contraseña
2. reiniciamos el servicio: systemctl restart sshd

En Rocky:
ssh 192.168.56.105 -l nico -p 22022 : ya sí nos deja y nos pide la contraseña.

En Ubuntu:
Ahora vamos a decirle que solo permita entrar al usuario elsarm:
sudo vi /etc/ssh/sshd_config : Insertamos la línea ‘AllowUsers elsarm’
systemctl restart sshd

En Rocky:
Comprobamos:
ssh 192.168.56.105 -l nico -p 22022 : no nos deja
ssh 192.168.56.105 -l elsarm -p 22022 : sí nos deja

COPIAS DE SEGURIDAD

COMANDO ‘dd’
(Data Duplicator)

Se usa para copiar dispositivos a nivel de byte (p.ej. Clonado de discos, copiar pendrive, backups, etc).

Sintaxis: `sudo dd if=origen of=destino`

Ejemplos:
dd if=/dev/sda of=/dev/sdb : copia de sda a sdb
dd if=/dev/sda of=~/hdadisk.img : imagen del disco en vez de dispositivo
dd if=hdadisk.img of=/dev/sdb : recuperas de la imagen a sdb
dd if=/dev/sda of=./sda_mbr.bin bs=512 count=1 : bs = número de bytes que serán leidos y escritos en una vez (por defecto 512), count = número de bloques que serán volcados
dd if=/dev/zero of=./muchos_zeros.raw bs=1024 count=512

COMANDO ‘cp’
Programa de copia a nivel de sistema de ficheros.

COMANDO ‘cpio’
(Copy In Out)

Toma la estructura del directorio y lo comprime dentro de un archivo.

Ejemplos:
ls | cpio -ov > /tmp/object.cpio : -o = copy out → lee una lista de nombres de archivo (en este caso de ls) y crea un archivo conteniendo estos archivos y su estado. -v = verbose
cpio -idv < /tmp/object.cpio : -i = extrae de un archivo (que se supone que previamente se ha comprimido con -o). -d = crea los directorios que hagan falta

cd /etc/
find .iname ‘*.conf’ | cpio -ov > /tmp/etc_conf_backup.cpio : comprimir
cpio -iduv < /tmp/etc_conf_backup.cpio : extrae. -u para reemplazar todos los archivos incondicionalmente.

---- ??
otra terminal de ubuntu, desde root:
nano /etc/pam.conf
desde la terminal primera:
cpio -iduv "pam.*" < /tmp/etc_conf_backup.cpio

COMANDO ‘tar’

Muy parecido a cpio. También comprime y descomprime.

Ejemplos:

Creamos unas carpetas con archivos random:
sudo apt install tree
cp hosts trabajo/
mv hosts trabajo/ugr/
mv sda_mbr.bin trabajo/personal/
mv muchos_zeros.raw trabajo/
tree : muestra de manera gráfica la jerarquía de directorios de nuestro SO.

Comprimimos:
tar czf trabajo.tgz trabajo/ : -c crea un nuevo archivo trabajo.tgz que contiene el directorio trabajo/ y todos sus archivos. -z filtrar el archivo con gzip. -f archive file.
ls : tenemos el nuevo archivo trabajo.tgz
rm -rf trabajo : forzamos borrado recursivo del directorio trabajo/ y todos sus archivos
tar xzf trabajo.tgz : extraer el directorio.
ls : ahora vuelve a aparecer el directorio trabajo/

COMANDO 'rsync'
(Remote Syncronization)

Sirve para sincronizar el contenido de dos directorios en el sistema de ficheros: sincroniza un directorio origen en un directorio destino (sólo actualiza lo que se ha modificado).

También sirve en remoto (a través de ssh) : esto se usa mucho ya que reduce la exposición de un servidor puesto que ssh es un programa muy seguro.
Cuando lo usamos con SSH, rsync debe estar instalado en los dos equipos.

Ejemplos usando SSH: (lo hacemos desde Rocky)

export RSYNC_RSH=ssh : indicamos que use ssh como programa de shell remota (rsh) para comunicarse entre las copias de archivos locales y remotos de rsync. Esto la modifica temporalmente, si queremos que se guarde permanentemente hay que modificar el .bash como hacemos con el history)

rsync -av --delete trabajo/ elsarm@192.168.56.105:/home/elsarm : copiamos el contenido del directorio trabajo/ de Rocky en el /home/elsarm/trabajo/ de la máquina de Ubuntu. -v = verbose. --delete hace que elimine archivos estraños

rsync -av --delete elsarm@192.168.56.106:/home/elsarm/trabajo/ . : Trae el directorio local de la máquina de Rocky a donde estoy ahora.

Observando esto, vemos que surge la necesidad de un sistema de control de versiones (Git).

GIT

- Directorio: el directorio donde hacemos lo que tengamos que hacer
- Stage: Zona intermedia donde se registran cambios e incluimos los cambios a seguir
- Commit: Zona donde se registran ya los cambios en la BD. Tras cada commit, se genera un hash de 32 bits que lo identifica. El último commit es el HEAD.

Todos los objetos de git tienen un hash para identificarlos (commits, trees: directorios y blobs: archivos - aunque solo 1 hash para todas las versiones del archivo).

Resumen comandos git importantes:

- init
- add
- commit [--amend]
- status
- log
- show
- clone
- push
- pull
- branch
- merge
- tag

git init proyecto : crea un repositorio con un subdirectorio oculto: /proyecto/.git
cd ./proyecto
nano -w README.md : Creamos el README y podemos añadir algo (“De momento, vacío”)

git status

git add README.md

git status

git commit : “Toma una fotografía de la copia actual del repositorio y la almacena”. Con la opción -a evitamos decir archivo por archivo cuál queremos commitear. Opción -m para decir un mensaje

git commit -a -m “Inicializo el repositorio” (Poner comentarios significativos en el examen)

Ahora nos pide identificarnos porque no lo hemos hecho y nos da dos posibles comandos para ello. Puede ocurrir que no nos lo pida y lo coja por defecto pero tenemos que modificarlo.

git config --global user.email “elsarodriguezm@correo.ugr.es”
git config --global user.name “Elsa”

Para modificar esto de otra forma podemos modificar el archivo:

nano -w .gitconfig

git commit -a -m “Inicializo el repositorio”

git log

nano -w README.md : Modificamos (“Ya no, ya hay algo”)

La primera línea del README.md es importante porque muchas veces aparece solo esa línea en el git log.

git log --oneline : Me dice cual es la rama máster y la cabecera

El último commit siempre lo podemos modificar, pero sólo el último:

git commit -a --amend : guarda lo último que hemos añadido en el README.md ya que no lo habíamos guardado. Nos puede servir por si nos equivocamos en el examen

.gitignore : Serie de templates para archivos que git no va a considerar.

Clonar un repositorio:
Meternos en gitHub, meternos en el repositorio que queremos clonar, darle al botón verde <Code> y elegir la opción de SSH, copiamos la URL, y en la terminal hacer:

git clone <URL> : Nos pregunta algo de la clave y decimos que sí.
cd <carpeta de la url>
git log

nano -w .gitignore : Aparecen archivos ejecutables, librerías, etc
git status : ahora nos dice que hemos modificado gitignore y que tenemos que subirlo
git add .gitignore
git commit -a : Escribimos “Añadimos los arhivos modificados y además…”
git push : Subimos los archivos a GitHub (podemos verlo en la página)

Si ahora queremos clonar un repositorio que tenemos en local, tenemos que indicar la ruta completa:
git clone ./proyecto/.git copiaproyecto/ : Copiar el repositorio proyecto en copiaProyecto
cd ./copiaProyecto
git remote
git remote show origin :

Crear una clave pública privada:
ssh -keygen
Luego tenemos que vincularla en nuestra cuenta de git (lo hacemos a través de la página)

EJERCICIO P2
git clone git@github.com:davidPalomar-ugr/ejercicioIseP2.git
cd ./repoViernes
git branch elsaRodriguez
git checkout elsaRodriguez : para cambiarme de rama a esa

nano -w alumnos.csv : Introducir lo siguiente: DNI, Apellidos, Nombre, emailUGR (nos podemos inventar el DNI)
git add alumnos.csv
git commit -a -m “Añado a la alumna Elsa Rodriguez”
git status
git push
git log
git push -u origin elsaRodriguez : Para indicar que lo estamos pusheando en nuestra rama
Hacer una mezcla (merge). Para ello tenemos que irnos a la rama principal y traernos nuestra rama. Hacemos primero git pull para asegurarnos de que estamos trabajando con el repo actualizado:
git checkout main
git pull
git merge elsaRodriguez
Si nos dice que hay un conflicto, tenemos que resolverlo. Abrimos el archivo
nano -w alumnos.csv
git status
Hacemos una de las opciones que nos dice (echarnos atrás o seguir hacia delante)
git add alumnos.csv
git status
git commit
git log
git push

Importante: que un merge no de conflicto no significa que sea correcto.

Otro aspecto: lo más común para crear ramas y movernos a ella directamente:
git checkout -b nuevaRama (con -b hacemos que se cree)

Para justificar lo que hemos hecho en el examen:
git log --decorate --graph --oneline --all

Para el examen podemos traer definido el usuario, correo y todo eso.

git tag v1.0.0 <checksum del commit> : crear una etiqueta simbólica a un commit
git push --tags : para que suba los tags al repositorio

git rebase -i HEAD 4 : si queremos que nos revise todos los últimos 4 commits. Esto nos puede ser útil cuando queramos juntar dos commits en uno.

git rebase --continue : No sé para qué sirve, supongo que por si lo anterior da conflicto

LAMP

En Rocky (más difícil que Ubuntu)
LAMP = Linux, Apache, Mysql o MariaDB, Python o PHP u otro intérprete

1. INSTALAR APACHE

dnf search apache : vemos que lo tiene httpd
sudo dnf install httpd
sudo systemctl status httpd : vemos si se ha arrancado al instalar y nos dice que no está arrancado
sudo systemctl enable httpd : lo habilitamos (pero sigue inactivo)
sudo systemctl start httpd
sudo systemctl status httpd : ya está arrancado

2. INSTALAR PHP:

dnf search php
sudo dnf install php : php es un traductor intérprete, no un servicio por lo que no hay que comprobar si se ha arrancado.
php -a : vemos que funciona
vi prueba.php : Escribir:

```
<?php
echo “Hola”
?>
```

Se ha creado un nuevo archivo de configuración /etc/php/php.ini. Podemos modificarlo para que apache coja php por defecto en todos los archivos terminados en .php (no sé cómo, el archivo es larguísimo.

phpinfo() : nos dice si todo está funcionando correctamente (útil para el examen). Para ejecutarlo desde la línea de comandos:
php -r “phpinfo();”
php -i : también da información pero no es lo mismo

3. INSTALAR MARIADB

sudo dnf install mariadb
Vemos si se ha iniciado el servicio:
systemctl status mariadb : Nos dice que no se encontró el servicio!!! Esto es porque hemos instalado el cliente, no el servicio. Instalamos el servidor:
sudo dnf install mariadb-server
systemctl status mariadb-server : Está desactivado y deshabilitado

sudo systemctl enable mariadb
sudo systemctl start mariadb
systemctl status mariadb

Ahora queremos ejecutar mysql.
mysql : nos da un error de acceso denegado para nuestro usuario. Probamos con root
mysql -u root : al profe le funciona pero a mí me sigue denegando el acceso. Con sudo mysql sí que me deja pero me pide contraseña.

Script para ajustar parámetros que vienen por defecto:
mysql_secure_installation : Ahora al pedir la contraseña vuelve a decirme acceso denegado para el root… Lo único que me ha funcionado ha sido seguir el tutorial https://linuxhint.com/change-mysql-root-password-ubuntu/

mysql_secure_installation : eliminamos usuarios anónimos, deshabilitaamos el acceso de root remoto, eliminamos bd de prueba y recargamos la tabla de privilegios

mysql -u root -p: Ahora esto nos pide la contraseña y entramos.

Si en el examen tenemos que hacer algo de una base de datos, tenemos que comprobar que el usuario tiene acceso a la base de datos:
mysql --uadmin -pise2023 ise2023 : ise2023 es la base de datos y -pise2023 es la contraseña

4. CONECTARNOS CON APACHE A MYSQL

Comando curl (Client URL): sirve para verificar la conectividad a una URL y para transferir datos.

Ahora comprobamos que funciona con el comando curl:
curl -v http://localhost : Me sale algo como el contenido de una pag web (pinta bien)
ip addr : Cogemos la ip de enp0s8 y la copiamos en el navegador. Nos da un error
Otra forma es hacer desde una terminal:
curl 192.168.56.106 (ip de mi maquina Rocky) : Da un error de conexión. Sin embargo, sí que funciona ping.

Entonces, el problema es el firewall de rocky ya que desde dentro podemos conectarnos pero desde fuera no.
sudo firewall-cmd --state : Vemos que está running (activo)
sudo firewall-cmd --list-all : Listar todos los puertos activos. Vemos que el puerto activo es 22022/tcp

Queremos añadir el servicio HTTP que nos permita conectarnos a la URL. Para ello podemos: o bien añadir el servicio http o bien añadir el puerto 80 (que es el de http). Las dos formas son:
1. sudo firewall-cmd --add-port=80/tcp --permanent
2. sudo firewall-cmd --add-service=http --permanent

Ya debería funcionar.
sudo firewall-cmd --list-all : Sigue saliendo lo mismo que antes…
Para que se actualice, tenemos que recargar:

sudo firewall-cmd --reload
sudo firewall-cmd --list-all : Ya sí vemos los cambios

Buscarmos ahora en el navegador http://IP y ya debería funcionar. También funciona:
curl IP : ya devuelve algo como el contenido de una web

sudo firewall-cmd --runtime-to-permanent : para guardarlo permanentemente

5. INTEGRAR SCRIPT DE PHP QUE SE CONECTA A UNA BD Y EL RESULTADO LO MUESTRA A TRAVÉS DE UN DOCUMENTO HTML

a) Conectamos PHP a MySQL:

Desde la terminal de nuestro ordenador nos conectamos con ssh a la máquina de Rocky y lo hacemos desde ahí.
¿Dónde creamos el archivo html? → ejecutamos: less /etc/httpd/conf/httpd.conf y vemos que el archivo que nos indica DocumentRoot es "/var/www/html"
cd /var/www/html/
sudo vi index.php : Copiar el script de https://www.php.net/manual/es/function.mysqli-connect.php modificando el usuario a root y la contraseña. La bd la dejamos porque la vamos a crear ahora.
Nos conectamos a mysql:
mysql -u root -p
> CREATE DATABASE mi_bd;

Veamos si esto ha funcionado. Buscamos en nuestro navegador: http://IP/index.php
- Al profe le sale el script php tal cual. Él lo que hace es modificar el fichero
sudo vi /etc/httpd/conf/httpd.conf : En la línea de DirectoryIndex añade *.php para que pueda cualquier fichero php. Luego reinicia el servicio: systemctl restart httpd. Al buscar de nuevo en el navegador le da un error 500 de HTTP.
- A mí me sale vacío.

Ambos casos ocurren porque al ejecutar index.php da un error con mysqli (no lo tengo instalado). Si nos queremos conectar a una base desde un lenguaje de alto nivel (en nuestro caso php) tenemos que instalarnos un driver:
sudo dnf install php-mysqlnd o sudo yum install php-mysqli
systemctl restart httpd : Reinciar

Ahora, si ejecutamos php index.php obtiene éxito. Sin embargo, al ejecutarlo desde el navegador no se puede conectar correctamente a la bd. Esto es un problema de permisos.
sudo less /var/log/audit/audit.log : Buscamos la palabra 'httpd' y vemos que no se le da permiso a httpd para que conecte con mysqld.
Otra forma de verlo es ejecutar
getsebool -a | grep httpd : el booleano httpd_can_network_connect_db está a false.
sudo setsebool -P httpd_can_network_connect_db=on : lo modificamos de forma permanente.
getsebool -a | grep httpd : Comprobamos que ya lo tenemos a on

AHORA AL BUSCAR EN EL NAVEGADOR http://IP/index.php YA FUNCIONA!!!

OBSERVACIÓN: El profe en clase creó un archivo .html en vez de .php:
nano -w index.html : Introducir cosas con comandos de html

Para el examen, nos dejará hacerlo como root.