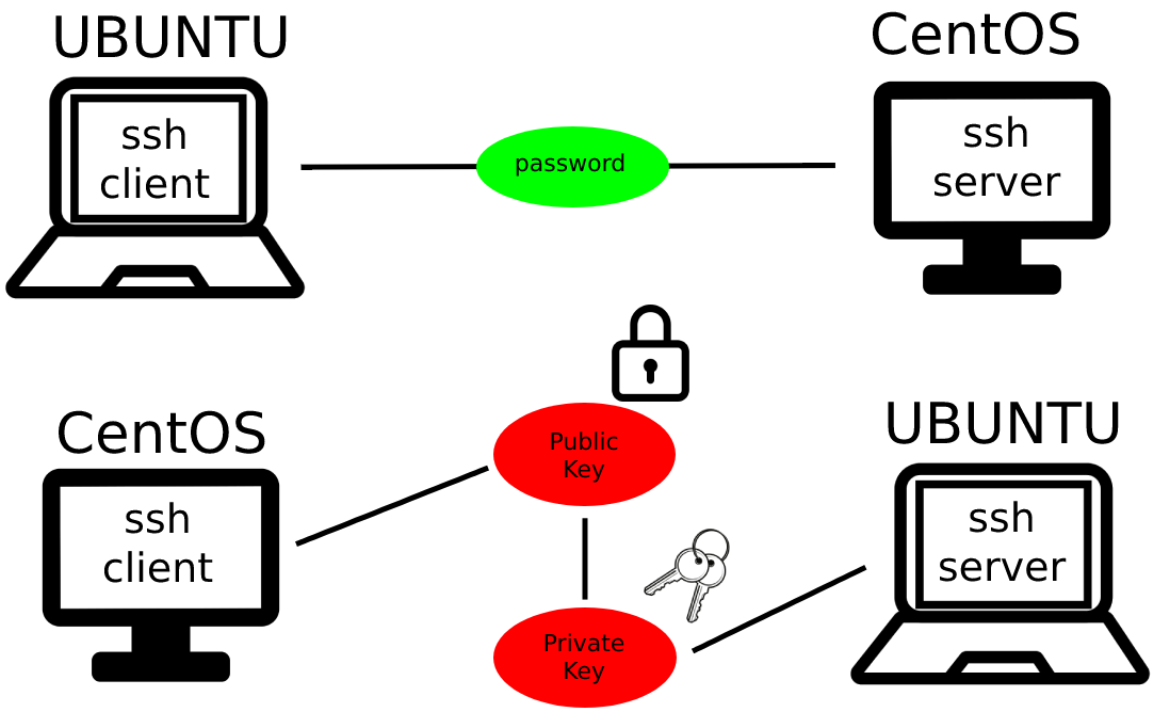
Guión práctica 2 de ISE

Nota: Donde aparezca alejandrorm (mi usuario) escribir el tuyo propio.

Lección 1

Para la primnera parte vamos a configurar ssh tanto en Ubuntu como en CentOS. En CentOS hay más mecanismos de seguridad así que tendremos que decirle al sistema que el puerto que usemos (cambariemos el por defecto) va a ser un puerto ssh. Además vamos a configurar que a CentOS solo se pueda conectar con clave pública, no se puede autenticicar con contraseña.



Entonces los pasos a seguir para la conexión Ubuntu -> CentOS son:

1. Primero instalamos en ubuntu openssh-server con la orden `sudo apt install openssh-server` (el paquete openssh-client ya está tanto en ubuntu como en centos).

2. Vamos a iniciar el demonio que controla ssh, el sshd, con el comando `sudo systemctl start sshd`.

3. Para comprobar que vamos bien podemos probar a hacer `ssh alejandrorm@192.168.56.105` desde CentOS y ver si iniciamos sesión en Ubuntu. (¡¡¡OJO!!! Todo lo que hagas ahora se hace en Ubuntu. Para salir tan sólo usa la orden `exit`).

4. Para cambiar el puerto por el que nos conectamos (22 por defecto) para mayor seguridad, pues sino basta con escuchar ese puerto y ya saben todo lo que hacemos, hacemos `sudo nano /etc/ssh/sshd_config` y descomentamos la línea `Port 22` y lo cambiamos por 22022. Para que no pueda entrar nadie siendo super usuario en nuestra máquina descomentamos también `PermitRootLogin yes` y lo cambiamos a no.

5. Recargamos el servicio desde Ubuntu con `sudo systemctl restart sshd` habiéndonos salido previamente en CentOS.

6. Ahora nos conectamos desde centos con la orden `ssh alejandrorm@192.168.56.105 -p 22022`. Notamos como ahora especificamos el puerto.

Ahora vamos a realizar la parte de CentOS, que es un poco más compleja:

1. En CentOS ya viene instalado el ssh por defecto. Vamos a cambiar también el puerto con `sudo nano /etc/shh/sshd_config` y descomentando el `Port 22` y cambiándolo acto seguido por 22022. Hacemos lo mismo con el `PermitRootLogin` tal y como en lo anterior.

2. Si intentamos resetear el demonio sshd, vamos a obtener un fallo porque no le hemos dicho a CentOS que el puerto 22022 será para ssh.

3. Para solucionar esto vamos a instalar el paquete `sumanage`. Para ello primero necesitamos configurar la red. Para ello hay que seguir una serie de pasos:

    1. Vamos a actualizar los mirrors. Para ello ejecutamos:

```
cd /etc/yum.repos.d/

sed -i 's/mirrorlist/#mirrorlist/g' /etc/yum.repos.d/CentOS-*

sed -i
's|#baseurl=http://mirror.centos.org|baseurl=http://vault.centos.org|g'
/etc/yum.repos.d/CentOS-*
```

    2. Ahora levantamos la interfaz de red. Para ello tan sólo hacemos `ifup enp0s8`

    Hay que notar que para que esto funcione hay que tener hecha la práctica anterior.

4. El paquete semanage no se puede instalar directamente. Vamos a instalar un paquete que lo contiene. Si hacemos `sudo yum provides semanage` obtendremos el nombre del paquete que lo contiene. Lo instalamos con `sudo yum install policycoreutils-python-utils`. Importante quitar el .noarch del final.

5. Ahora ya podemos hacer `sudo semanage port -l | grep ssh` para ver que puertos están configurados para ssh y entonces hacemos `sudo semanage port -a -t ssh_port_t -p tcp 22022` para añadir el 22022. Podemos comprobar que así ha sido otra vez con la orden `sudo semanage port -l | grep ssh`.

6. Lo que hay que hacer ahora es abrir el cortafuegos en ese puerto para paquetes tcp, que son los que utiliza ssh. Para ello usamos:

   `sudo firewall-cmd --permanent --add-port=22022/tcp`.

   Y recargamos el cortafuegos con `sudo firewall-cmd --reload`.

7. Ya podemos desde otra máquina hacer `ssh alejandrorm@192.168.56.110 -p 22022` y conectarnos a nuestro CentOS. La orden para salir sigue siendo `exit`.

8. Vamos a configurar la clave pública para poder conectarnos sin necesidad de autenticiarnos con contraseña desde Ubuntu. Importante SIN SER ROOT hacemos `ssh-keygen` y le damos a enter para aplicar todas las configuraciones por defecto (que son guardar la clave en el archivo ~/.ssh/id_rsa.pub y no establecer contraseña a la clave). Podemos comprobar nuestra clave con `cat .ssh/id_rsa.pub`.

9. Ahora hacemos `ssh-copy-id alejandrorm@192.168.56.110 -p 22022` también desde Ubuntu y ya nos podemos conectar desde Ubuntu a CentOS sin que nos pida la contraseña.

10. Para eliminar la posibilidad en CentOS de conectarnos con contraseña desde CentOS hacemos `sudo nano /etc/ssh/sshd_config`. Vamos a descomentar `PasswordAuthetication yes` y ponerlo a no. Tan sólo falta reiniciar el demonio sshd con `sudo systemctl restart sshd` y ya podemos comprobar desde Ubuntu como podemos conectarnos a CentOS sin que nos pida contraseña.

Lección 2

En esta lección vamos a aprender a utilizar git mediante la plataforma GitHub de Microsoft. Esto nos permite tener un control sobre las versiones de nuestros proyectos. Se puede realizar esta práctica desde nuestra máquina propia.

Empezamos por instalar git en el caso de que no lo tuviéramos.

Para ello empezamos creando una cuenta en GitHub (con la cuenta de la Universidad tienes GitHub Pro gratuito). Una vez que la tengamos nos vamos a nuestro perfil y creamos un nuevo repositorio. En la pantalla que aparece podemos darle un nombre al repositorio, en mi caso, pruebaISE; una descripción, por ejemplo "Un repositorio de prueba para Ingeniería de

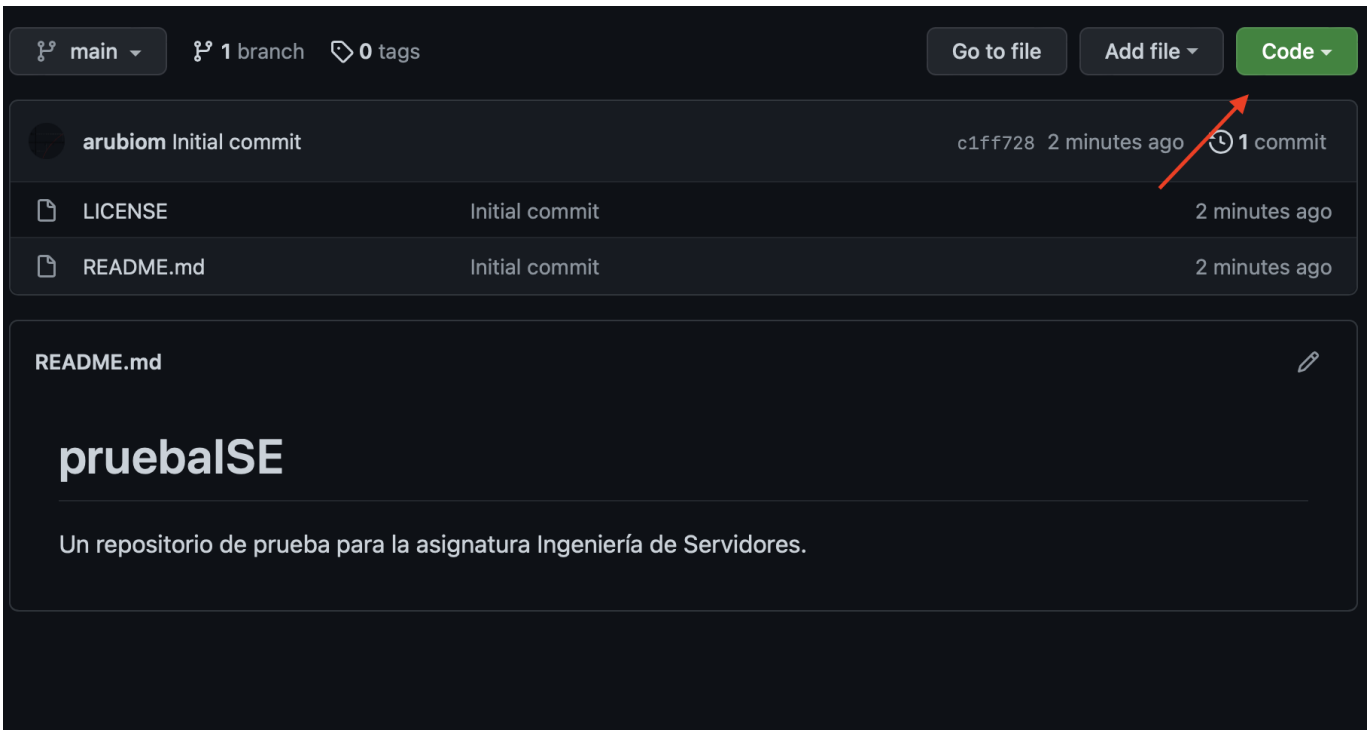
Servidores"; la visibilidad, que vamos a seleccionar privado; y los archivos que queremos incluir en nuestro repositorio desde el principio. Estos son:

- **.gitignore** Este fichero incluye los archivos de los que no queremos llevar una actualización.
- **README.md** Este fichero es una descripción en lenguaje de etiquetado MarkDown sobre nuestro repositorio. En GitHub se visualizará su versión ya "compilada".
- **license** Esta será la licencia que queremos ponerle a nuestro repositorio.

En este caso vamos a seleccionar empezar con un README y una licencia de tipo Apache License 2.0 por ningún motivo en especial. Ya podemos crear nuestro repositorio.

Podemos ver la previsualización de nuestro readme con el nombre del repositorio y la descripción de este así como los dos ficheros LICENSE y README.md.

Vamos a clonar nuestro repositorio en nuestra máquina para poder trabajar sobre él. Para ello nos vamos al botón Code:



Ahora vemos un menú flotante con un enlace. Podemos elegir que el enlace sea de https o de ssh. Para empezar vamos a elegir https.

Ahora nos vamos al lugar donde queremos guardar nuestro repositorio localmente. Abrimos una terminal en ese lugar y hacemos `git clone <enlace>` donde el enlace puede ser cualquiera de los dos. Vamos a modificar nuestro README y lo cambiamos por el siguiente (o por lo que se desee):

```
# Probando Git
Esto es una prueba de `Git`
```

Vamos a ver ahora como guardar los cambios. Desde nuestra terminal podemos usar `git status` para comprobar los cambios que hemos hechos y si ya los hemos guardado o no. Podemos ver como nos dice que README.md ha sido modificado.

Para guardar los cambios primero tenemos que añadir estos al "árbol". Para ello usamos `git add README.md`, o lo que es más común `git add .` que añadirá todo menos aquello que se encuentre en el .gitignore.

Ahora si hacemos `git status` podemos ver que ya registra el cambio, pero todavía nos pide que lo confirmemos. Esto quiere decir que hay que hacerle "commit", que según la definición dada por GitKraken, es una instantánea de tu repositorio en un instante de tiempo específico. Para hacerlo usamos `git commit -m "Modificado el README"`. Podemos poner el mensaje que deseemos.

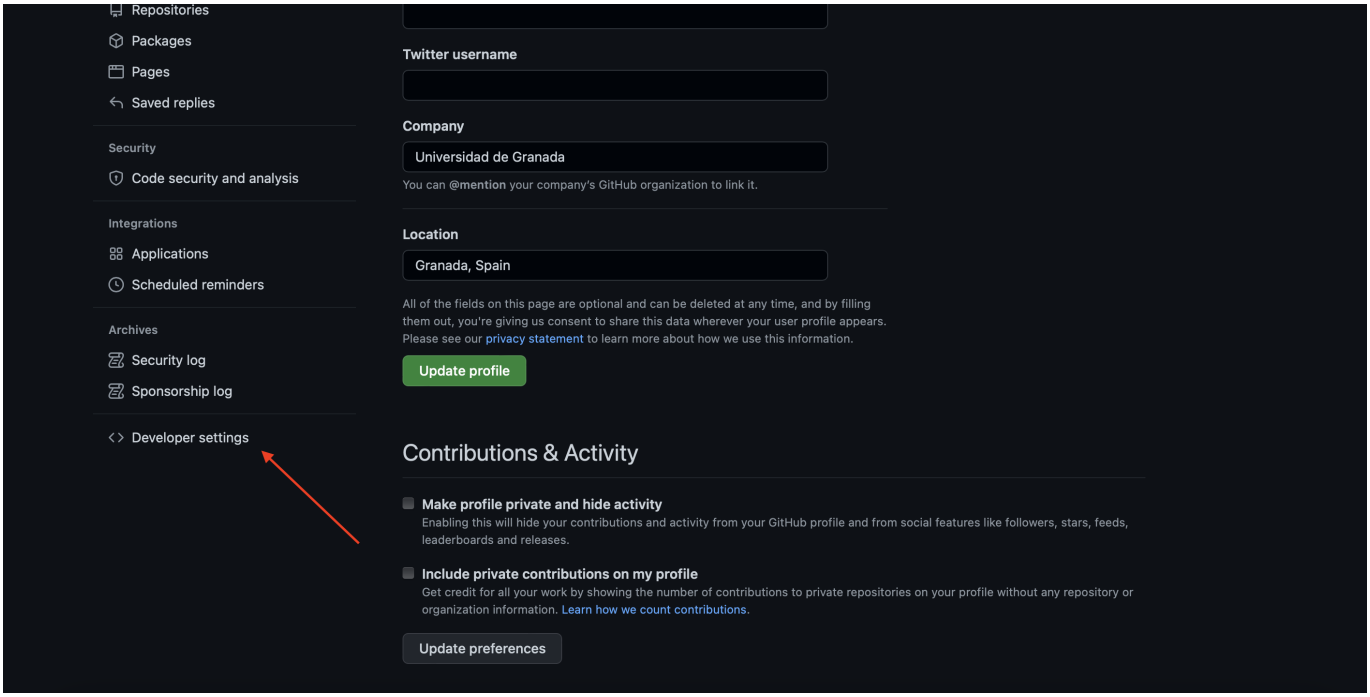
Ahora al usar `giot status` nos informa que nuestra rama (ya veremos que es eso) está adelantada por 1 commit. Esto quiere decir que si nos vamos a nuestro repositorio de GitHub y recargamos todavía no veremos los cambios que hemos realizado. Antes de nada necesitamos añadir nuestro nombre de ususario y nuestro correo a git. Para ello hacemos:

`git config --global user.email "tu email"`

`git config --global user.name "tu usuario"`

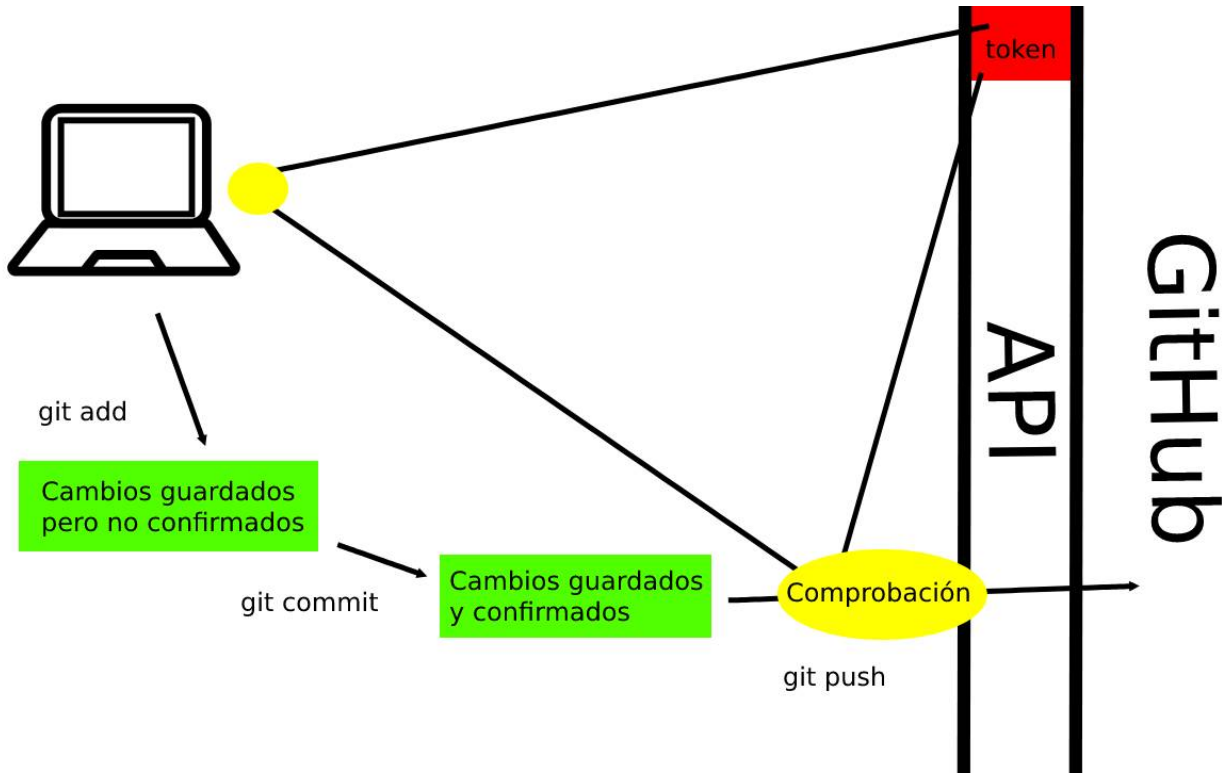
Ahora podemos probar a hacer `git push origin main` (origin es como nos referimos a nuestro repositorio local y main la rama a la que estamos subiendo los cambios) pero al pedirnos la contraseña obtendremos un error. Esto se debe porque desde 2020 GitHub obliga a usar tokens cuando queremos usar el protocolo https para subir los archivos.

Para crear un token nos vamos a nuestro perfil de GitHub -> Configuración y en la barra de la izquierda bajamos hasta encontrar las opciones para desarrollador:



Una vez dentro nos vamos a Generar nuevo token de acceso personal y le damos un nombre, seleccionamos cuanto tiempo queremos que dure y los permisos que le damos. Marcar el primero (repo) es suficiente para lo que necesitamos. Le damos a generar token y nos aparecerá nuestro token. Ojo, hay que guradarlo en algún fichero de nuestra máquina pues GitHub no nos lo volverá a enseñar más.

Ahora ya podemos ir a nuestra terminal de nuevo ubicada en la carpeta que tenemos el repositorio (si hacemos `ls -all` podemos ver que hay una carpeta oculta llamada .git) y hacer un `git push origin main`. Ahora cuando nos pida la contraseña pegamos nuestro token y ya podemos entrar en GitHub, recargar la página y ver que los cambios están subidos.

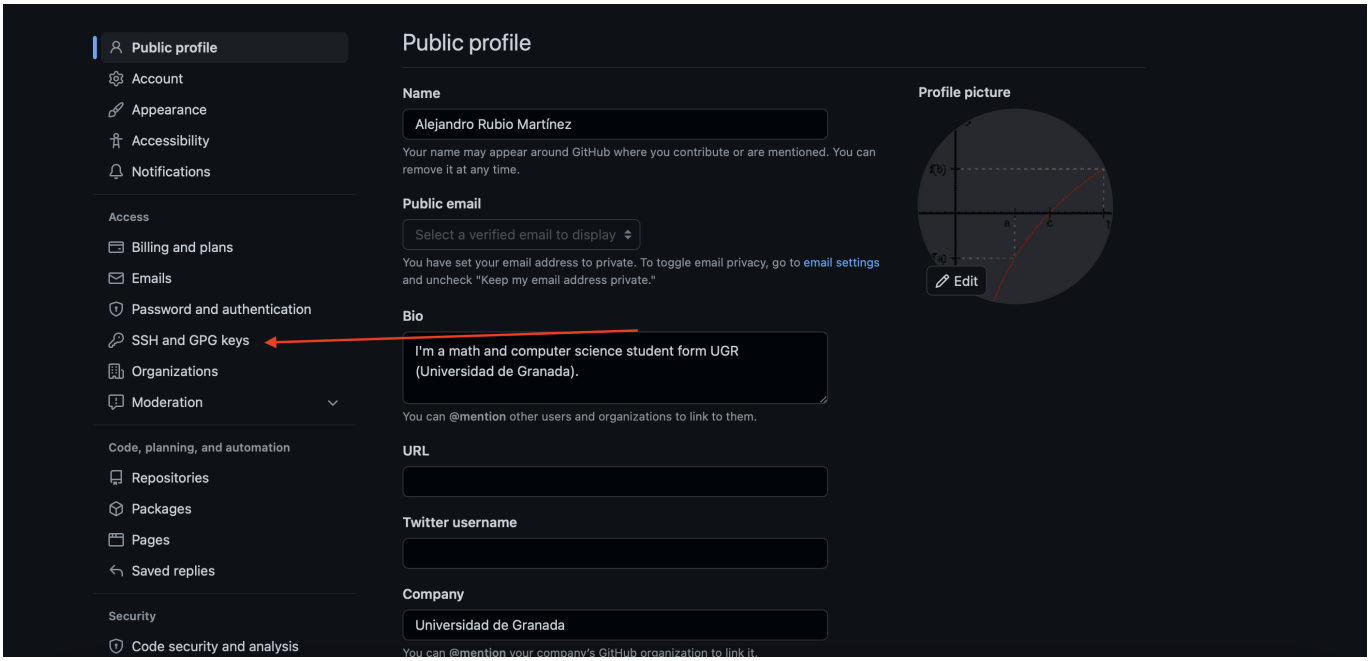


SSH

Vamos a hacer ahora que todo sea más fácil usando ssh. Para ello vamos a generar una clave ssh y subirla a GitHub. Con esto no tendremos que autenticarnos cada vez que queramos pushear (así no usamos los molestos tokens).

Para empezar generamos una clave ssh (de la misma forma que en la lección anterior) usando `ssh-keygen` y enter para todo por defecto. Esta clave se ha guardado en ~/.ssh por defecto. Imprimimos la clave con el comando `cat <nombre del fichero .pub>` y nos vamos a GitHub.

Ahora entramos en nuestro perfil -> Configuración y buscamos ahora en la barra izquierda Claves SSH y GPG.



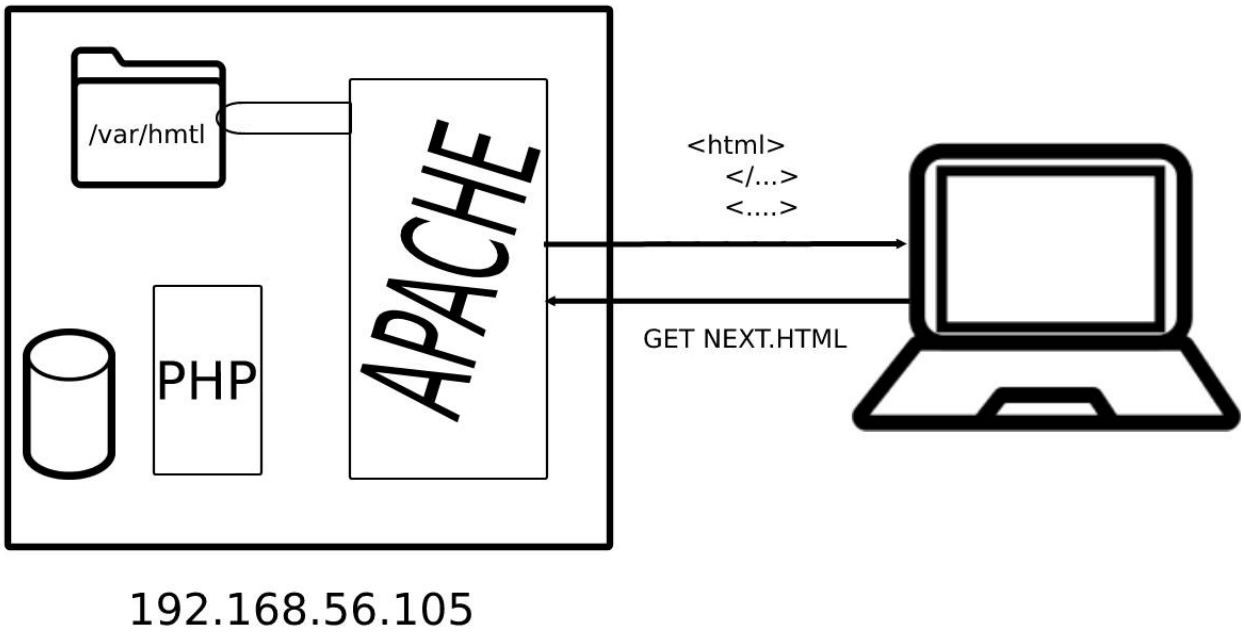
Una vez dentro le damos a añadir clave ssh y le damos un nombre y copiamos la clave.

Podemos probar la conexión por ssh con el comando `ssh -T git@github.com`. Si hay algún problema es posible que se necesite seguir la documentación oficial de GitHub para conectar con SSH.

Lección 3

Nota: Esta práctica puede romper ubuntu completamente así que es recomendable clonarla en una máquina aparte.

En esta práctcia vamos a configurar LAMP (Linux, Apache, Mysql, PHP) en nuestros servers, tanto Ubuntu como CentOS.



Vamos a empezar con Ubuntu:

1. Instalamos tasksel con `sudo apt install tasksel` y lo ejecutamos con `sudo tasksel`. Desmarcamos todas las opciones y marcamos LAMP server (al ok se va con el tabulador). Si da un error de apt-get update mientras se instala hacemos `sudo apt update` y repetimos. En el mensaje de abort Kernel le damos a sí.
2. Comprobamos los estados de los demonios de apache y mysql con `sudo systemctl status apache2` y `sudo systemctl status mysql`. En el caso de que no estuvieran iniciados usamos `systemctl start`. Podemos comprobar php usando php -a lo cual nos lleva a una pequeña consola de php.
3. Nos conectamos desde un navegador externo a ubuntu buscando http://192.168.56.105 Si queremos modificar la página que estamos viendo modificamos el fichero /var/www/html/index.html con nano. Por ejemplo probar a escribir algo debajo del del body.

Vamos ahora con la parte de CentOS:

1. Vamos a empezar con istalar el demonio de http con `sudo yum install httpd`. Además vamos a iniciarlo con `sudo systemctl start httpd`.

2. Vamos ahora a abrir el cortafuegos. Para ello hacemos `sudo firewall-cmd --permanent --add-service=http` y lo recargamos con `sudo firewall-cmd --reload`. Ahora nos conectamos desde un navegador externo a http://192.168.56.110].

3. Vamos ahora con la parte de mariadb (alternativa a mysql). Lo instalamos con `sudo yum install mariadb` y `sudo yum install mariadb-server`. Iniciamos el demonio con `systemctl start mariadb`. Ahora vamos a ponerle usuario y contraseña. Para ello hacemos `mysql_secure_installation`, y cuando nos pida la contraseña de mariadb, como actualmente no tiene ninguna, le damos al intro. En el momento que nos pregunte si queremos establecer una nueva contraseña escribimos yes. Le decimos a todo yes.

4. Vamos a instalar php con `sudo yum install php`. Podemos probar que funciona con `php -a`. Además vamos a instalarnos unas funciones de php con `sudo yum install php-mysqli`. Hacemos `sudo nano /var/www/html/index.php` y escribimos el siguiente script:

```
<?

echo("Holi");

$link = mysqli_connect('127.0.0.1::3306','root','practicas,ise');

if (!$link)
    die("No se puede conectar a la db");

echo("Conectado a la db");

?>
```

   Para ejecutarlo hacemos `sudo php /var/www/html/index.php` y probamos ahora a conectarnos desde un navegador externo a http://192.168.56.110/index.php

5. Ahora hacemos `sudo nano /etc/httpd/conf/httpd.conf` y nos vamos hasta:

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

   Y lo cambiamos por:

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

   Reiniciamos http con `sudo systemctl restart httpd` e intentamos conectarnos desde un navegador externo a http://192.168.56.110/index.php. Si no aparece nada podemos hacer click derecho -> Ver código fuente y vemos que no se ha conectado.

6. Vamos a cambiar una variable del sistema. Para ello usamos la orden `getsebool -a` pero aparecerán demasiadas. Vamos a filtar usando `getsebool -a | grep httpd`. Podemos fijarnos en la variable httpd_can_networfk_connect_db que está en off. Para cambiarla tan solo hacemos `sudo setsebool -P httpd_can_network_connect_db on`. Probamos ahora a recargar el navegador externo.

Vamos a seguir con CentOS pero ahora vamos a instalar un programa que nos banea si fallamos demasiadas veces en conectarnos por ssh mediante contraseña.

1. Empezamos con permitir la conexión mediante contraseña. Para ello hacemos `sudo nano /etc/ssh/shhd_config` y ponemos PasswordAuthetication a yes. Reiniciamos el demonio de ssh con `sudo systemctl restart sshd`.
2. Instalamos ahora los programas que vamos a usar con `sudo yum install fail2ban epel-release` e iniciamos el demonio de fail2ban con `sudo systemctl start fail2ban`. Para ejecutarlo hacemos `sudo fail2ban-client` y para comprobar el estado de las "cárceles" hacemos `sudo fail2ban-client status`.
3. Vamos a modificar el archivo de configuración de fail2ban. Para ello hacemos `sudo nano /etc/fail2ban/jail.conf`. Al abrir el archivo vemos como nos pide que no lo modifiquemos. Vamos a hacerle caso y copiarlo en otro sitio. Usamos `cp -a /etc/fail2ban/jail.conf /etc/fail2ban/jail.local`. Vamos a modificar este archivo con nano. Usamos la combinación de teclas ctrl+w para buscar la frase # SSH servers donde vamos a añadir enabled = true. Si buscamos #bantime.maxtime podemos desbanear automáticamente pasada cierta cantidad de tiempo pero ahora mismo no nos interesa. Reseteamos fail2ban con `sudo systemctl restart fail2ban`.
4. Ahora desde otra máquina nos conectamos por ssh con `ssh alejandrorm@192.168.56.110 -p 22022` y nos equivocamos intencionadamente unas cuantas veces hasta que nos banee.
5. Desde CentOS podemos comprobarlo con `sudo fail2ban-client status` y nos saldrá una cárcel que llamaremos X. Para ver la información de sólo esa cárcel podemos hacer `sudo fail2ban-client status X`. Para desbanearnos hacemos `sudo fail2ban-client set X unbanip <nuestra ip>`.