Practica-2.pdf

fer__luque

Ingeniería de Servidores

3º Grado en Ingeniería Informática

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
Universidad de Granada

Práctica 2: Instalación de servicios

Sesión 1: SSH

Ubuntu

Instalación SSH

El primer paso es instalar SSH, del que hay que diferenciar entre cliente y servidor.

En nuestro caso, debemos instalar tanto cliente como servidor (al igual en centOS).

Para instalarlos, se puede hacer con el gestor de paquetes (*openssh-client* y *openssh-server*), pero en nuestra ocasión lo vamos a instalar con tasksel (que habrá que instalar previamente). Sólo debemos usar tasksel con el sistema "limpio"

```
# Instalamos tasksel
sudo apt install tasksel
# Ejecutamos tasksel
tasksel
# Instalamos la opción de openssh server, el client viene ya instalado en Ubuntu
```

Lo instalamos con el gestor porque da error el tasksel

```
sudo apt-get install openssh-server
sudo apt-get install openssh-client # Debería estar instalado
```

Una vez instalado, debemos comprobar que se ha instalado. Para cualquier servicio, debemos revisarlo con systemctl

```
# Revisamos si está activo
sudo systemctl status ssh # sshd en centOS
# Si no lo está activamos
sudo systemctl start ssh # O restart
```

En general para los servicios, si no se arrancan al inicio del sistema, debemos hacerles un enable

```
sudo systemctl enable <servicio>
```

Configuración SSH

Modificamos el fichero de configuración de ssh

```
sudo vi /etc/ssh/sshd_config
```

Descomentamos la línea que indica el puerto 22 y le damos uno de los puertos que esté libre (22022). Todo esto se hace para ir "complicando" los ataques

```
Port 22022
```

Descomentamos el PermitRootLogin y no le permitimos acceder:

```
PermitRootLogin no
```

Reseteamos el servicio para que se vuelva a leer la configuración

```
sudo systemctl restart ssh
```

Comprobamos ahora si el servicio está activo (desde windows por ejemplo). Le tenemos que especificar el puerto que hemos cambiado con -p, porque por defecto intenta el puerto 22

```
ssh ludef@192.192.56.105 -p 22022
```

CentOS

Instalación SSH

En centOS tenemos ssh instalado, tanto el servidor como el cliente, comprobamos con systemctl

```
sudo systemctl status sshd
```

Configuración SSH

Vemos que el puerto actual es el 22, del mismo modo modificaremos ahora las mismas configuraciones que en ubuntu

```
Port 22022
PermitRootLogin no
```

Al intentar reiniciar el servicio con

```
sudo systemctl restart sshd
```

nos da un error. Podemos revisar el error con

```
journalctl -xe
```

SELinux - semanage

Vemos que hay un error al enlazar el puerto solicitado porque no tenemos permiso. Esto es debido a SELinux, al que hay que "pedirle permiso" para cambiar el puerto. Para ello utilizaremos la herramienta **semanage** que hay que instalar

```
sudo dnf install semanage
```

lo que da error porque no encuentra el paquete. Para encontrar en que paquete se encuentra ejecutamos

```
sudo dnf provides semanage
```

instalamos el paquete que nos pide

KEEP CALM AND ESTUDIA UN POQUITO

WUOLAH + #QuédateEnCasa

```
sudo dnf install policycoreutils-python-utils
```

y ahora ya podemos instalar semanage

```
sudo dnf install semanage
```

Consultamos ahora la lista de puertos y la filtramos con grep por ssh

```
sudo semanage port -l | grep ssh
```

Ahora modificamos el puerto que nos indica

```
sudo semanage port -a (añadir) -t ssh_port_t -p tcp 22022
```

Podemos volver a consultar los puertos y vemos que se mantendrá el puerto 22, ya que SELinux no nos deja eliminar el puerto por defecto

Ahora sí, reiniciamos el servicio

```
sudo systemctl restart sshd
```

Apertura de puertos

Sin embargo ahora si intentamos acceder a él (desde windows por ejemplo)

```
ssh ludef@192.168.56.110 -p 22022
```

no nos deja porque hay que abrir el puerto previamente.

Para que nos deje, debemos abrirlo con firewall

```
sudo firewall-cmd --add-port=22022/tcp # Esto lo abre automáticamente hasta que reinicie la máquina
sudo firewall-cmd --permanent --add-port=22022/tcp # Esta opción lo deja abierto
```

Sin embargo, con el --permanent no se abre automáticamente, podemos reiniciar el sistema o recargar el firewall

```
sudo firewall-cmd --reload
```

Y finalmente podemos acceder desde Windows

```
ssh ludef@192.168.56.110 -p 22022
```

Común

Generación de claves

IP Ubuntu: 105

IP CentOS: 110

IMPORTANTE: **No hacer estos pasos con sudo/root**

Para que no nos pidan la contraseña cada vez que entramos, se genera una contraseña. Desde **Ubuntu**

```
ssh-keygen
```

Y se la pasamos al servidor

```
ssh-copy-id ludef@192.168.56.110 -p 22022
```

Comprobamos que ha recibido la contraseña haciendo un ssh a centOS, si no nos pide la contraseña ha funcionado

```
ssh ludef@192.168.56.110 -p 22022
```

Del mismo modo lo hacemos desde **centOS** para acceder a Ubuntu.

Prohibir contraseñas

Para evitar la entrada con contraseña, modificamos el fichero de configuración de ssh y descomentamos la línea de PasswordAuthentication

```
PasswordAuthentication no
```

De esta forma, sólo se pueden entrar con las claves (keygen)

Sesión 2: Copias de seguridad y control de versiones

Copias de seguridad

Por copia de seguridad entendemos cualquier redundancia que le demos a uno o varios archivos.

Copia binaria

Se puede copiar de un disco a otro

```
dd if=/dev/sda of=/dev/sdb
```

Copiar archivos y empaquetar

Copia clásica

```
cp;
```

Copia archivos a y desde (archivos)

```
ls | cpio
```

Empaquetado

```
tar
```

Sincronizar

```
rsync;
```

Sincroniza una fuente y un destino (incluso a través de SSH):

- Copia enlaces, dispositivos, propietarios, grupos y permisos
- Permite excluir archivos
- Transfiere los bloques modificados de un archivo
- Puede agrupar todos los cambios de todos los archivos en un único archivo
- Puede borrar archivos

Para hacer una copia con rsync en un mismo dispositivo:

```
rsync prueba/* pruebaBackup/
```

Sin embargo, hacer una copia en el mismo servidor no tiene sentido. Lo más lógico sería hacerlo en otro dispositivo

Como ejemplo, hacemos una copia de una carpeta de Ubuntu en CentOS. Para ello, en Ubuntu:

```
rsync -avize "ssh -p 22022" /home/ludef/prueba/*
ludef@192.168.56.110:/home/ludef/pruebaBackupCentOS
# Nos da información sobre la operación, vemos que se envían más bytes de los que
se reciben porque comprime
```

Otra opción es rnapshot

```
rsnapshot; # Revisar en Swad
```

Control de versiones - Git

Para evitar posibles errores y eliminaciones de copias de seguridad y datos, una herramienta muy interesante es el control de versiones

Cómo funciona

- **Directorio de trabajo**: lo que tenemos en desarrollo
- **Stage**: Zona donde se registran los campbios. Ahí incluimos los cambios que queremos seguir
- **Commit**: Zona donde los cambios se convierten en permanentes (podemos hacer commits en diferentes ramas "branches" y ponerles etiquetas "tags").
- El repositorio remoto (como GitHub) sería la cuarta zona

La evolución del trabajo puede mostrarse como un grafo dirigido acíclico

Cada vez que se hace un commit (nueva versión) se genera un hash que lo identifica (usando los primeros 7 suele ser suficiente).

El último commit es el HEAD y podemos desplazarnos a partir del identificador con ~ y ^:

- HEAD~3 == HEAD\^\^\^ == HEAD^3

Objetos de git:

- *Blobs*: archivos (inodos y datos)
- *Trees*: directorios
- *Commits*

Iniciando un repositorio

Instalamos git

Nos situamos en el directorio donde queramos trabajar

Escribimos:

```
git init
```

Ya hemos tomado la instantánea ¿de qué?, para revisarlo:

```
git log; git status; git branch;
```

Todo la información se almacena en `$PWD/.git`, si lo borramos, perderemos el control de cambios

Una vez iniciado, debemos indicar quienes somos, editando `~/.gitconfig`:

```
[user]
    name = Alum Uno
    email = auno@correo.ugr.es
```

o con `git config`

```
git config --global user.name "Alumn Dos"
git config --global user.email ados@correo.ugr.es
```

En lugar de iniciar un repositorio en un directorio, vamos a crear un repositorio en GitHub:

Entramos en GitHub y creamos un repositorio:

- README file: Descripción del respositorio
- .gitignore: para evitar el seguimiento de algunos archivos
- License: La licencia de distribución

Una vez creado, copiamos su URL y hacemos un git clone donde queramos meter el repositorio

```
git clone https://...
```

Creamos algunos archivos para hacer pruebas en nuestro repositorio clonado.

Una vez creados, revisamos con

```
git status
```

los archivos que no están registrados en el control de versiones. Nos dice todos aquellos que no están trackeados. Los ficheros solo tienen que ser añadidos una vez, ya que cuando git los conozca no tenemos por qué volver a añadirlos.Para añadirlos:

```
git add * # Para trackearlos todos
```

Con el add hemos pasado del **Directorio de trabajo** al **Stage**. Para que este cambio pase también al **Commit**, debemos de hacer:

```
git commit -m "Incluido prueba.txt" # -m para indicar el nombre del commit
```

Para ver los diferentes commit que hemos hecho usamos:

```
git log
```

Modificamos el archivo y hacemos commit para ver más. Hacemos git log y ahora encontramos los dos commit que hemos hecho, solo que ahora, el HEAD es el ultimo commit.

Para ver exactamente qué cambios se han hecho dentro de cada commit, copiamos el hash y hacemos:

```
git show 7e1be05e801c33ad1480eebe891c5f2f4a63296d
```

Revertir cambios

Introducimos algunos cambios (modificar prueba.txt) y les hacemos commit

En caso de que quisiéramos revertir un commit, cogemos el hash de la versión a la que volver y hacer un git reset:

```
git reset 4f075fcb70236b62a541f0821dbaf0c8353f8f0e
```

El reset no elimina los cambios, sino que devuelve las versiones antiguas al directorio de trabajo, pero los almacena en el commit, es decir, hemos hecho que cambie el puntero de HEAD. Para eliminar esa versión hacemos:

```
git restore [archivos]
```

Si no queremos eliminarla pero sí queremos volver a un commit antiguo sin eliminar los nuevos:

```
git revert 7e1be05e801c33ad1480eebe891c5f2f4a63296d
```

Pero tenemos algunos conflictos que hay que modificar a mano. Una vez modificados:

```
git revert --continue
```

Ahora ya nos queda la versión antigua en el tope de la pila pero hemos mantenido los que tenemos encima.

Ver modificaciones

Para ver las modificaciones que se han realizado entre el directorio de trabajo, el stage y el repositorio:

Cuando modifiquemos algo:

- Al hacer `git diff` sin `git add`, encontrará diferencias
- Una vez hagamos add, `git diff` no las encontrará, pero sí lo hará `git diff --staged` y el `git diff HEAD`

- Si hacemos commit, ninguna de las 3 las encontrará

Repositorio remoto

Para subir todos los cambios a nuestro repositorio remoto, debemos hacer push:

```
git push origin master
```

Para sincronizar nuestro repositorio local con el remoto, hacemos pull

```
git pull origin master
```

Si añadimos algo desde GitHub se hace un commit automáticamente, para sincronizarlo con nuestro repositorio local hacemos git pull

Creando ramas

Por defecto se encuentra la rama master, para crear nuevas ramas :

```
git branch <nombre_rama>
```

Para ver en qué rama estamos

```
git branch
```

Para cambiar de rama:

```
git checkout <nombre_rama>
```

Para poder salvar el directorio actual sin tener que hacer un commit (así podremos cambiar de rama):

```
git stash
```

Así podemos dejar trabajo a medias sin hacer commit y sin perderlo. Para aplicar el último stash creado

```
git stash apply
git stash apply <stash> # Para aplicar uno en concreto
```

Para mostrar los stash disponibles

```
git stash list
```

Para mezclar ramas:

```
git merge
```

Estos merge dependen de la política que se tengan pero normalmente se realiza por medio de merge pull requests que tienen que ser aceptados.

Sesión 3: Instalación Sevidor Web

Una vez instalado SSH, nos falta la instalación del propio servicio. Para ello, llevaremos a cabo la instalación de lo que se conoce como pila LAMPP (Linux, Apache, MySQL (o MariaDB), PHP y Python).

Ubuntu

Tasksel

La pila LAMPP se puede instalar directamente con la herramienta tasksel (LAMP server).

```
sudo tasksel
# Opción LAMP server
```

Sin embargo, es muy destructivo porque elimina algunas cosas como ssh, por lo que solo conviene hacerlo justo tras instalar Ubuntu

Apache

Instalamos apache

```
sudo apt-get install apache2
# Si da error hacemos sudo apt-get update
# Puede dar error de dpkg (hacemos lo que dice)
```

Comprobamos que se ha instalado

```
sudo systemctl status apache2
# Si está inactivo hacemos start o restart
sudo systemctl start apache2
```

Otra forma de comprobar si está activo, es comprobar la ip de ubuntu en el navegador.

Puede ser que no permita acceder porque ubuntu haya cerrado el puerto 80. Lo abrimos

```
sudo ufw allow 80
```

MySQL

Lo instalamos, tanto el cliente como el servidor

```
sudo apt-get install mysql-server mysql-client
```

Comprobamos el estado y si no activamos

```
sudo systemctl status mysql
```

PHP

Instalamos

```
sudo apt-get install php
```

Comprobamos que interprete con un hello world.

```
php -a # Abrir intérprete
> echo('Hello world');
```

Y con esto ya estaría todo instalado en Ubuntu

CentOS

Por si no está automatizado el arranque de internet ejecutamos

```
sudo ifpu enp0s3
```

Apache

Instalamos (con yum) el paquete de apache (que en CentOS se llama httpd)

```
sudo yum install httpd
```

Comprobamos y activamos

```
sudo systemctl status httpd
sudo systemctl start httpd
```

Podemos comprobar en el navegador. Vemos que no funciona porque centOS sí que trae el firewall cerrado. Tenemos que abrir el puerto 80 igual que lo hicimos con ssh, pero podemos indicarle ahora el servicio en lugar del puerto. Hacemos un reload para que se abra el puerto porque el --permanent no lo hace en el momento

```
sudo firewall-cmd --permanent --add-service=http
sudo firewall-cmd --reload
```

MariaDB

Instalamos

```
sudo yum install mariadb mariadb-server
```

Comprobamos y activamos

```
sudo systemctl status mariadb
sudo systemctl start/enable mariadb
```

Debemos ejecutar un script de configuración para "personalizar" la seguridad básica del mariadb

```
sudo mysql_secure_installation
```

Seguimos los pasos del script:

- Contraseña de root de MariaDB (en este caso no hemos configurado, metemos en blanco)

- Metemos una nueva contraseña de root (ISE)
- Eliminar los usuarios anónimos (pensados para tests)
- Deshabilitamos el login remoto de root
- Eliminamos la base de datos para tests
- Recargamos los privilegios de las tablas

Probamos mysql y vemos si sirve la contraseña

```
mysql -u root -p
> ISE
```

PHP

Instalamos

```
sudo yum install php
```

Comprobamos

```
php -a
> echo('Hello world');
```

Ya está el LAMPP instalado en CentOS

Creación de página dinámica con PHP en CentOS

Librerías PHP

Para conectarnos a las bases de datos de mysql

```
sudo yum install php-mysqlnd.x86_64 # tabulamos
```

Vamos a añadir la página de index donde nos indica Apache HTTP Server en su página por defecto:

```
/var/www/html/
```

Creamos el fichero

```
sudo vi /var/www/html/index.php
```

Introducimos el script php en este fichero y lo probamos. Primero con php "en local":

```
php /var/www/html/index.php
```

Si da mucha información está bien, si da error puede haber algún fallo de sintaxis o de conexión.

Ahora desde el navegador vemos que no nos lo muestra. Esto ocurre ya que por defecto apache busca un index.**html** y no un php, por lo que tenemos que cambiarlo en el fichero de configuración

```
sudo vi /etc/httpd/conf/httpd.conf
# En ubuntu no se llama httpd sino apache2
```

Una vez dentro tenemos que encontrar (vi con shift+7 (barra)) la palabra index.html.

Aquí se indica el directorio donde busca http, simplemente añadimos el index.php con una coma detrás.

Reiniciamos httpd

```
sudo systemctl restart httpd
```

Ahora, intentamos desde el navegador y vemos que interpreta bien, pero no se puede conectar con la base de datos.

Este problema viene de las relaciones que permite SELinux tener entre servicios. En este caso, no está permitiendo que http se conecte a la base de datos. Por tanto, hay que indicarle que lo permita.

Para encontrar estos flags

```
sudo getsebool -a | grep httpd
```

En concreto, se trata del flag `httpd_can_network_connect_db` que está en off. Para cambiarlo

```
sudo setsebool -P httpd_can_network_connect_db on
```

Una vez cambiemos el flag, ya debería funcionar desde el navegador.

Seguridad en SSH

`fail2ban`

Instalamos pero primero hay que instalar una dependencia

```
sudo yum install epel-release
sudo yum install fail2ban
```

Funciona como un servicio, así que actuamos como siempre

```
sudo systemctl status fail2ban
sudo systemctl enable fail2ban
sudo systemctl start fail2ban
```

Para ver información sobre fail2ban

```
sudo fail2ban-client status
```

Funciona por cárceles, vamos a configurar la cárcel de ssh. Como todo servicio en la carpeta /etc

```
sudo vi /etc/fail2ban/jail.conf
```

Tal y como nos dice el fichero, no hacemos los cambios en jail.conf, sino en un fichero local jail.local

```
sudo cp -a /etc/fail2ban/jail.conf /etc/fail2ban/jail.local
```

Empezamos a configurar el jail.local. Para ello nos vamos al apartado de jails, donde tenemos de SSH, de HTTP...

Para dar de alta la cárcel de ssh

```
enabled = true
port = 22022
```

Reiniciamos el servicio

```
sudo systemctl restart fail2ban
```

Con

```
sudo fail2ban-client status sshd
```

podemos ver las estadísticas de esta cárcel.

Para probarlo, tenemos que empezar a fallar las contraseñas. Ahora mismo las contraseñas de ssh están desactivadas, por lo que tenemos que activarlas:

```
sudo vi /etc/ssh/sshd_config
# Modificar línea PasswordAuthentication yes
```

Reiniciar el sshd.

Desde windows intentamos entrar con contraseñas erróneas. Podemos revisar el estado de la cárcel. Una vez fallemos 5 veces vemos que nuestra IP del anfitrión está baneada.

Para desbanear

```
sudo fail2ban-client set sshd unbanip 192.168.56.1
```

Y ya podríamos entrar desde windows.