Práctica 1: Preparación de las herramientas

01/03/2022

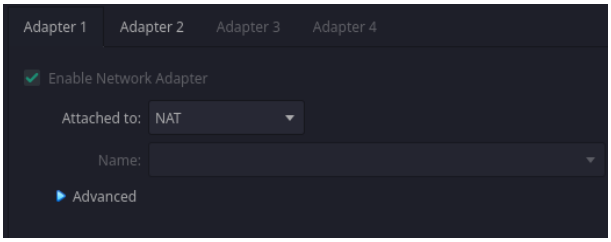
—

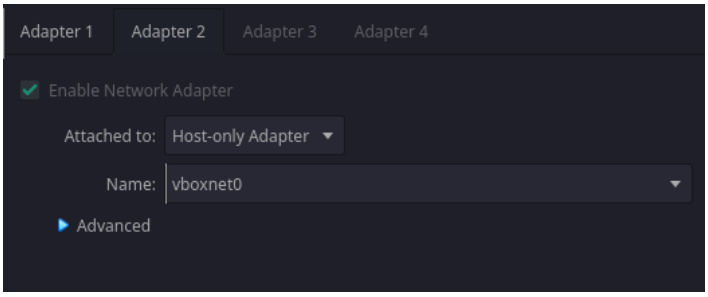
Adrián Acosa Sánchez
3º Grado en Ingeniería Informática (Tecnologías de la Información)

Índice

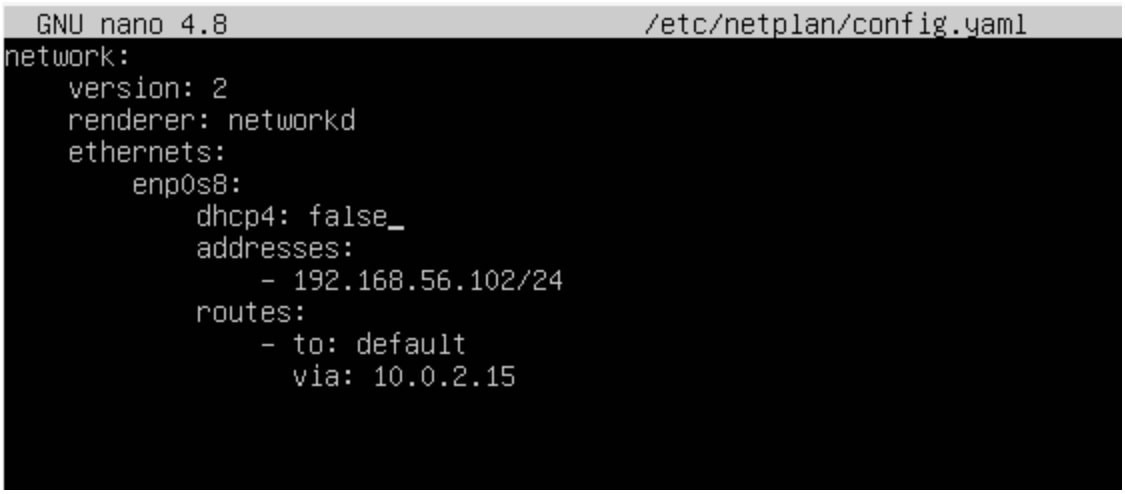
Configuración de las máquinas virtuales

Para la práctica he usado la imagen de Ubuntu 20.04.4, configurando todo tal como pone en el guión. En este caso mi "hostname" es "m1-adrianacosa" o "m2-adrianacosa" en función de la máquina en la que estemos, mi usuario es "adrianacosa" y la contraseña de ambas máquinas es "Swap1234". Lo primero que he hecho con ambas máquinas es habilitar la interfaz de red "host-only" para que se puedan comunicar entre ambas ellas y el anfitrión. Para ello he configurado en virtualbox lo siguiente en ambas máquinas:





Una vez tenemos las interfaces configuradas, el siguiente paso será configurar el netplan en cada una de las máquinas para que cada vez que las iniciemos se le asigne una IP y una puerta de enlace para poder conectarnos a internet en caso de que no sea una IP de la red de "host-only" (de igual manera en M2):



En este caso he asignado las direcciones estáticas 192.168.56.102 con máscara /24 (podría usar una máscara más pequeña, ya que sólo necesitaremos como mucho 4 IPs, pero quiero darle direcciones de sobra ya que es una red solo al host y no me importa reservar en la máscara tantas IPs). También he desactivado la asignación dinámica de IPs mediante

dhcp ya que quiero que siempre voy a tener asignadas las mismas IP a cada una de las máquinas. Por último, he configurado la puerta de enlace a la dirección del adaptador NAT, ya que si una petición no se hace con destino a alguna de las IP del adaptador de "host-only", esa petición se redirigirá o bien a una dirección del rango de direcciones del adaptador NAT o bien a internet mediante éste último.

He configurado la contraseña de root para cuando tengamos que manejar archivos de configuración no tengamos que estar poniendo sudo delante de cada comando. En este caso, mediante la orden "sudo passwd root", establecemos la contraseña de superusuario la cual he decidido que sea la misma que para el usuario "adrianacosa" ("Swap1234").

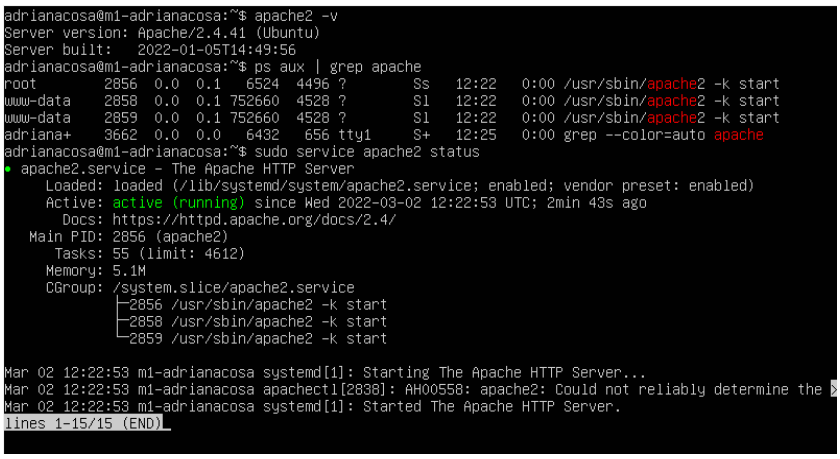
Antes de pasar a realizar las tareas de la práctica, procederemos a instalar todos los programas necesarios para ésta. En este caso tenemos que instalar la pila "LAMP", es decir, apache y mysql como se especifica en el guión. La instalación ha sido realizada como aparece en la siguiente imagen:

```
adrianacosa@m2-adrianacosa:~$ sudo apt-get install apache2 mysql-server mysql-client
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  apache2-bin apache2-data apache2-utils libapr1 libaprutil1 libaprutil1-dbd-sqlite3
  libaprutil1-ldap libcgi-fast-perl libcgi-pm-perl libencode-locale-perl libevent-core-2.1-7
  libevent-pthreads-2.1-7 libfcgi-perl libhtml-parser-perl libhtml-tagset-perl
  libhtml-template-perl libhttp-date-perl libhttp-message-perl libio-html-perl libjansson4
  liblua5.2-0 liblwp-mediatypes-perl libmecab2 libtimedate-perl liburi-perl mecab-ipadic
  mecab-ipadic-utf8 mecab-utils mysql-client-8.0 mysql-client-core-8.0 mysql-common
  mysql-server-8.0 mysql-server-core-8.0 ssl-cert
Suggested packages:
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom www-browser libdata-dump-perl
  libipc-sharedcache-perl libwww-perl mailx tinyca openssl-blacklist
The following NEW packages will be installed:
  apache2 apache2-bin apache2-data apache2-utils libapr1 libaprutil1 libaprutil1-dbd-sqlite3
  libaprutil1-ldap libcgi-fast-perl libcgi-pm-perl libencode-locale-perl libevent-core-2.1-7
  libevent-pthreads-2.1-7 libfcgi-perl libhtml-parser-perl libhtml-tagset-perl
  libhtml-template-perl libhttp-date-perl libhttp-message-perl libio-html-perl libjansson4
  liblua5.2-0 liblwp-mediatypes-perl libmecab2 libtimedate-perl liburi-perl mecab-ipadic
  mecab-ipadic-utf8 mecab-utils mysql-client mysql-client-8.0 mysql-client-core-8.0 mysql-common
  mysql-server mysql-server-8.0 mysql-server-core-8.0 ssl-cert
0 upgraded, 37 newly installed, 0 to remove and 1 not upgraded.
Need to get 33.3 MB of archives.
After this operation, 270 MB of additional disk space will be used.
Do you want to continue? [Y/n] _
```

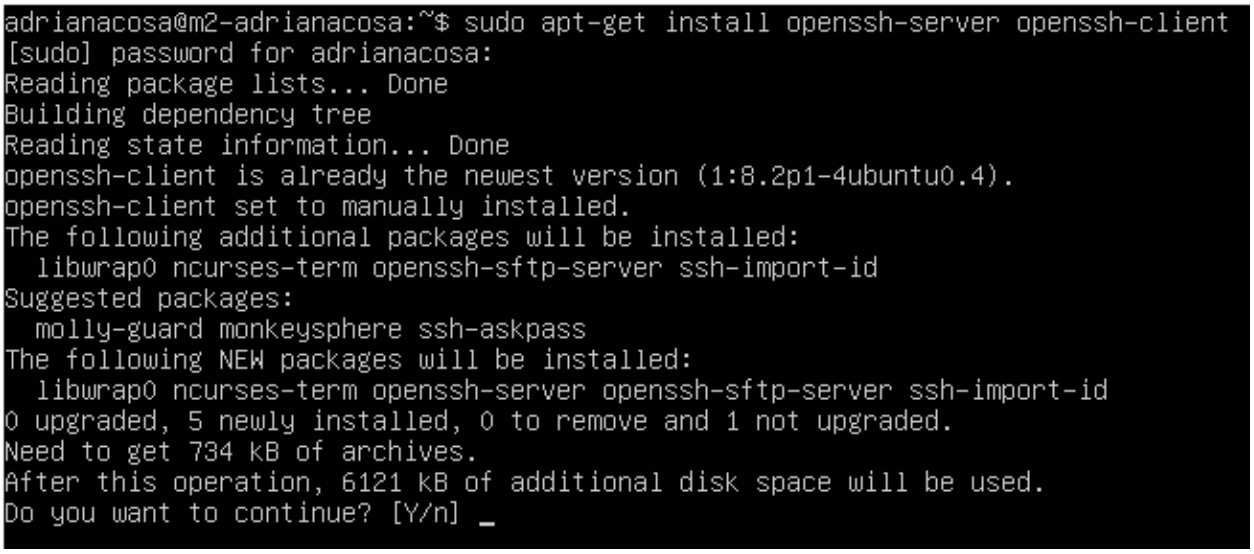
Tras la instalación, comprobamos la versión que hemos instalado del servidor con el comando "apache2 -v" que proporciona la siguiente salida:

```
adrianacosa@m1-adrianacosa:~$ apache2 -v
Server version: Apache/2.4.41 (Ubuntu)
Server built:   2022-01-05T14:49:56
adrianacosa@m1-adrianacosa:~$
```

También, para asegurarnos completamente de que el servicio está funcionando correctamente, podemos comprobarlo de la siguiente manera:



También tenemos que instalar en ambas máquinas el servicio de SSH, que se instala con los paquetes “openssh-server” y “openssh-client”:



Y debido a que es una buena práctica comprobar el estado de estos servicios una vez instalados, procedo a comprobarlo:

```
adrianacosa@m1-adrianacosa:~$ sudo service sshd status
● ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
     Active: active (running) since Wed 2022-03-02 15:57:13 UTC; 10min ago
       Docs: man:sshd(8)
             man:sshd_config(5)
   Main PID: 1627 (sshd)
      Tasks: 1 (limit: 1066)
     Memory: 2.1M
     CGroup: /system.slice/ssh.service
             └─1627 sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups

Mar 02 15:57:13 m1-adrianacosa systemd[1]: Starting OpenBSD Secure Shell server...
Mar 02 15:57:13 m1-adrianacosa sshd[1627]: Server listening on 0.0.0.0 port 22.
Mar 02 15:57:13 m1-adrianacosa sshd[1627]: Server listening on :: port 22.
Mar 02 15:57:13 m1-adrianacosa systemd[1]: Started OpenBSD Secure Shell server.
adrianacosa@m1-adrianacosa:~$ _
```

Como último paso de la configuración de los servidores web, lo que haré será crear un archivo swap.html en la carpeta /var/www/html/ para comprobar que haciendo uso del comando curl desde la otra máquina es posible la obtención de la página html. Para ello voy a seguir el archivo html que se usa de ejemplo en el guión pero en este caso con mis datos:

```
<html>
        <head>
                <title>MAQUINA 1</title>
        </head>
        <body>
                <h1>MAQUINA 1</h1>
                <p>Web de ejemplo de adrianacosa para SWAP<br>
                Email: adrianacosa@correo.ugr.es</p>
        </body>
</html>
```

Y haciendo uso de curl podemos comprobar el correcto funcionamiento accediendo desde la máquina 2 al swap.html de la máquina 1 como aparece en la siguiente imagen:

```
<html>
        <head>
                <title>MAQUINA 2<title>
        </head>
        <body>
                <h1>MAQUINA 2</h1>
                <p>Web de ejemplo de adrianacosa para SWAP<br>
                Email: adrianacosa@correo.ugr.es</p>
        </body>
</html>
```

Ahora ya podemos empezar con los requerimientos de la práctica.

Instalación y configuración de SSH Server y SSH Cliente

La instalación de SSH, tanto de server como de cliente, lo tengo explicado en la configuración de las máquinas. Luego en este apartado me centraré en la configuración del servicio.
Antes de configurar nada de SSH, lo primero que voy a hacer es comprobar que el servicio funciona como debería accediendo desde el equipo anfitrión como se muestra en la imagen:

```
~ ssh adrianacosa@192.168.56.101
The authenticity of host '192.168.56.101 (192.168.56.101)' can't be established.
ED25519 key fingerprint is SHA256:Ey0oH9DGtH4Bxuv8i3wdEUOemuPapbt+ltumYYVkwOY.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.56.101' (ED25519) to the list of known hosts.
adrianacosa@192.168.56.101's password:
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.4.0-100-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Wed 02 Mar 2022 04:41:34 PM UTC

  System load:  0.0               Processes:               111
  Usage of /:   53.2% of 8.90GB   Users logged in:         1
  Memory usage: 59%               IPv4 address for enp0s3: 10.0.2.15
  Swap usage:   0%                IPv4 address for enp0s8: 192.168.56.101

 * Super-optimized for small spaces - read how we shrank the memory
   footprint of MicroK8s to make it the smallest full K8s around.

   https://ubuntu.com/blog/microk8s-memory-optimisation

1 update can be applied immediately.
To see these additional updates run: apt list --upgradable


Last login: Wed Mar  2 15:53:52 2022
adrianacosa@m1-adrianacosa:~$
```

Una vez comprobado esto, ya podemos comenzar con las configuraciones. Para ello vamos a su archivo de configuración situado en el directorio /etc/ssh/sshd_config. En este fichero es donde se encuentran todas las configuraciones de SSH del sistema. Luego vamos a empezar a cambiar parámetros según nuestro criterio.

Configuración de ssh

En primer lugar vamos a configurar el puerto de escucha del servicio SSH del 22 al 22022. También vamos a activar el acceso mediante clave pública. Y por último vamos a activar activar el PasswordAuthentication para que cuando enviemos la clave pública desde el cliente SSH, tengamos que logearnos para comprobar que somos nosotros. Todas estas configuraciones aparecen en la siguiente imagen con colores distintos al cian:

```
Include /etc/ssh/sshd_config.d/*.conf

Port 22022
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::

#HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_ecdsa_key
#HostKey /etc/ssh/ssh_host_ed25519_key

# Ciphers and keying
#RekeyLimit default none

# Logging
#SyslogFacility AUTH
#LogLevel INFO

# Authentication:

#LoginGraceTime 2m
#PermitRootLogin prohibit-password
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

PubkeyAuthentication yes
```

```
# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication yes
#PermitEmptyPasswords no
```

Para aplicar estas configuraciones tendremos que reiniciar el servicio con la orden “systemctl restart sshd.service”.

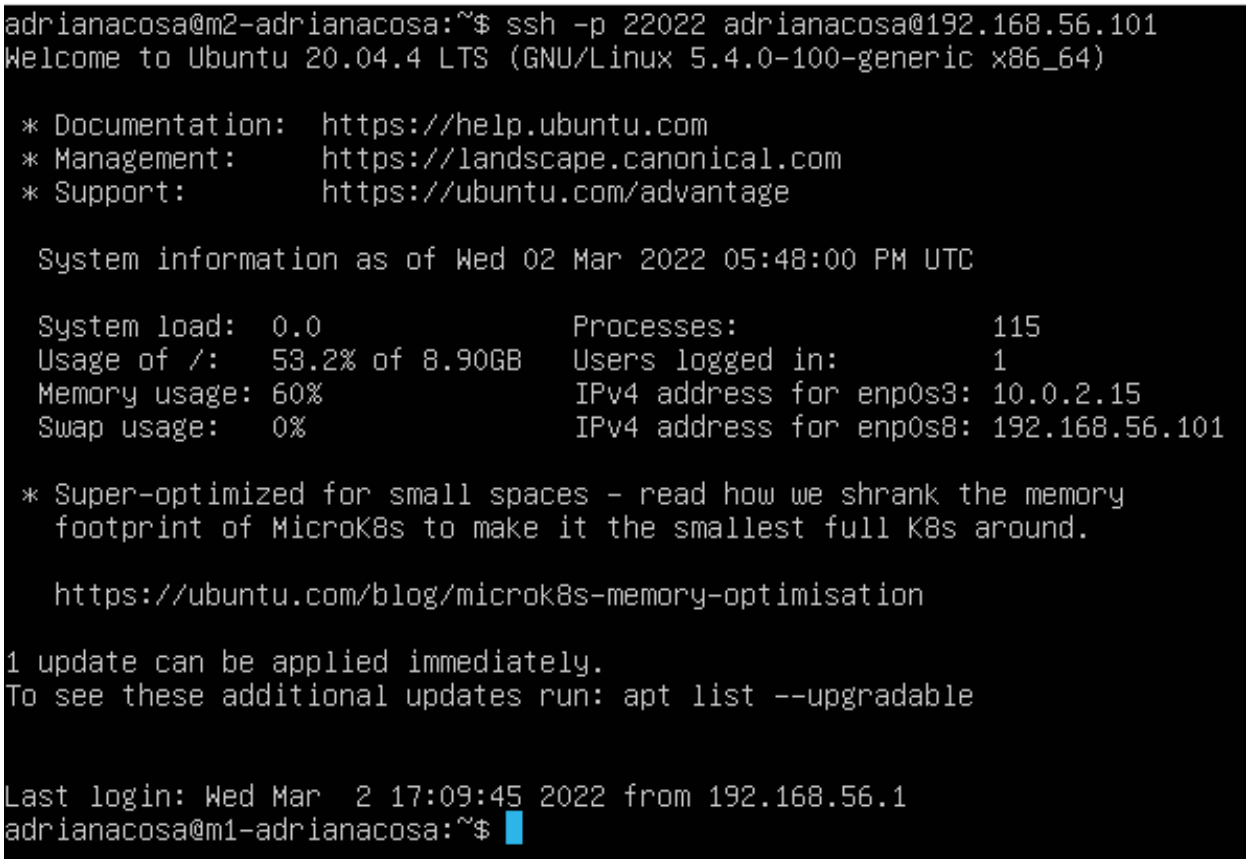
Inicio de sesión sin contraseña (clave pública)

Ahora vamos a pasar a los clientes SSH, que serán la máquina 2 y la máquina anfitrión. Lo primero que tenemos que hacer es generar una clave SSH en la máquina dos que será copiada en la máquina 1 para que quede registrada como conocida y así no necesitar contraseña para entrar. Para ello tenemos que seguir los siguientes pasos que aparecen en las imágenes siguientes (el primer comando es ssh-keygen, pero sale cortado en la máquina virtual):

```
Generating public/private rsa key pair.
Enter file in which to save the key (/home/adrianacosa/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/adrianacosa/.ssh/id_rsa
Your public key has been saved in /home/adrianacosa/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:bm6p4VE1WyOM121tMseXjC6vokY6DJ7MDWSLtfieFOk adrianacosa@m2-adrianacosa
The key's randomart image is:
+---[RSA 3072]----+
|                 |
|       o o o o . |
|      . * O * +  |
|  +.   . B B .   |
| *oo    S . .    |
|o.=.  .o   o     |
| =E* oo o.  .    |
| .*.=..=+  .     |
| .o  o++...      |
+----[SHA256]-----+
adrianacosa@m2-adrianacosa:~$ ssh-copy-id -p 22022 adrianacosa@192.168.56.101
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/adrianacosa/.ssh/id_rsa.pub"
The authenticity of host '[192.168.56.101]:22022 ([192.168.56.101]:22022)' can't be established.
ECDSA key fingerprint is SHA256:hw395z2wyksJCEJYOhurY/lqaN4eaUAkYwuvkwAE+Nw.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are alr
eady installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to inst
all the new keys
adrianacosa@192.168.56.101's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh -p '22022' 'adrianacosa@192.168.56.101'"
and check to make sure that only the key(s) you wanted were added.

adrianacosa@m2-adrianacosa:~$
```

Para poder copiar nuestra clave pública en el servidor SSH lo hacemos mediante el comando “ssh-copy-id”, a la que le indicamos el usuario y la IP del servidor al que queremos mandar nuestra clave pública y éste nos pedirá que iniciemos sesión para comprobar que verdaderamente somos nosotros los que le estamos mandando la clave.

Una vez tenemos esto hecho, probamos como dice al final a conectarnos por SSH a la máquina 1 y vemos si pide o no la contraseña:



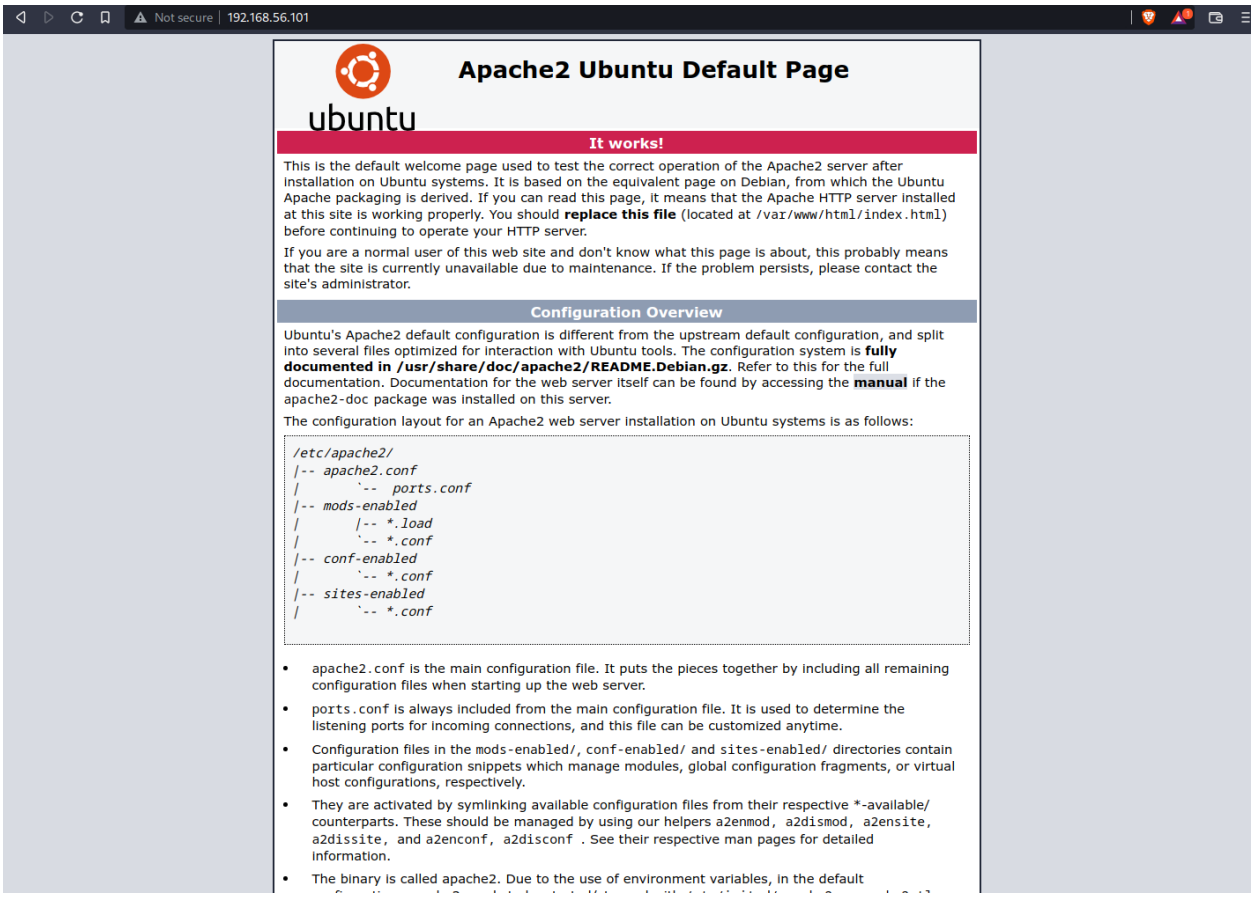
Y como vemos no la pide.

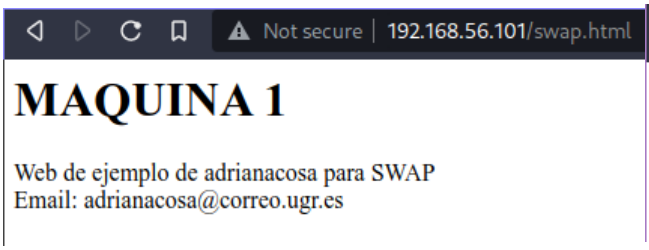
Con esto hemos terminado de configurar por completo SSH. Resumiendo: hemos configurado para SSH un puerto distinto al 22 (el que viene por defecto), hemos habilitado el acceso por clave pública y hemos habilitado la autenticación por contraseña para poder confirmar nuestra identidad a la hora de conectarnos al servidor SSH.

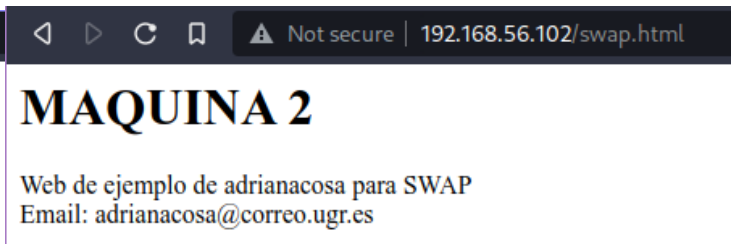
Instalar y configurar servicios Apache

Al igual que dije en el servicio SSH, la instalación de apache aparece en el apartado de configuración de las máquinas virtuales. Luego comenzaremos directamente a configurar lo que se pide en la práctica.

Como podemos comprobar en la imagen inferior, tenemos en funcionamiento el servicio de apache con dos archivos: index.html y swap.html (al que le he hecho unos arreglos para que sea más legible)







Una vez comprobado que todo está en orden, vamos a tocar algunas configuraciones.

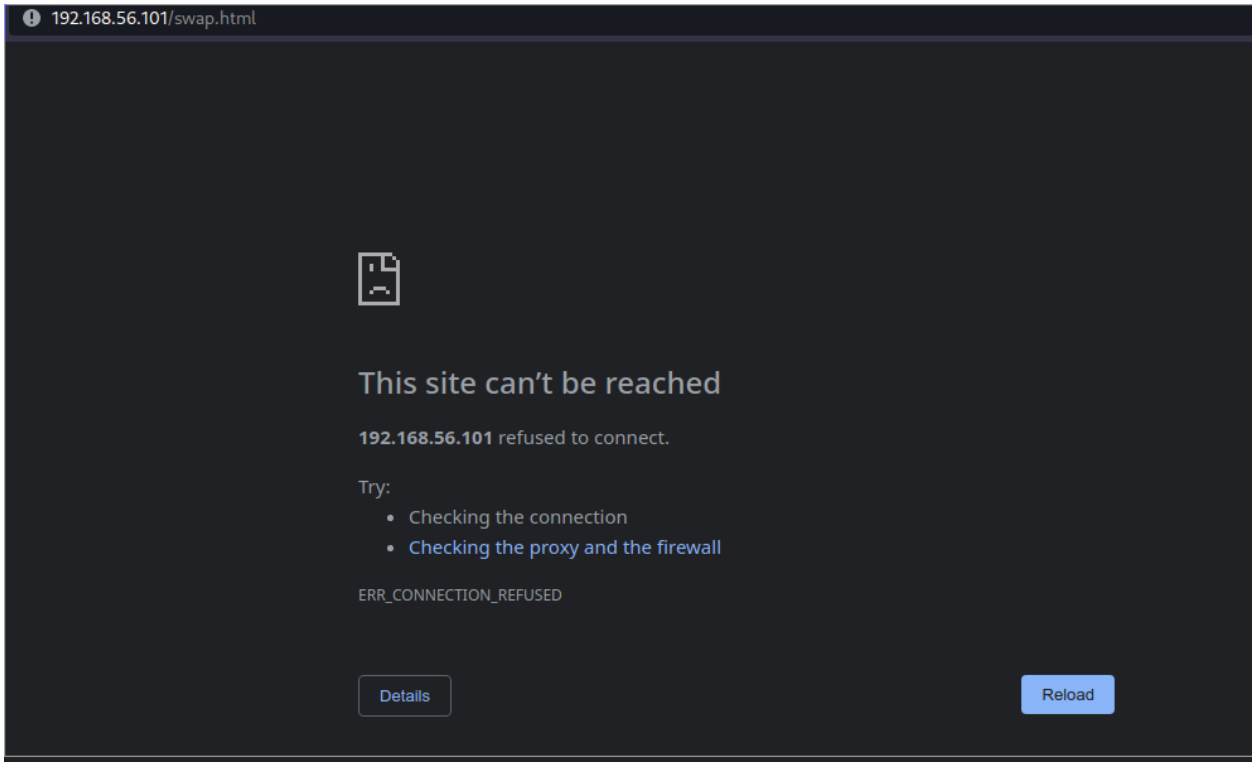
Cambio de puerto de apache

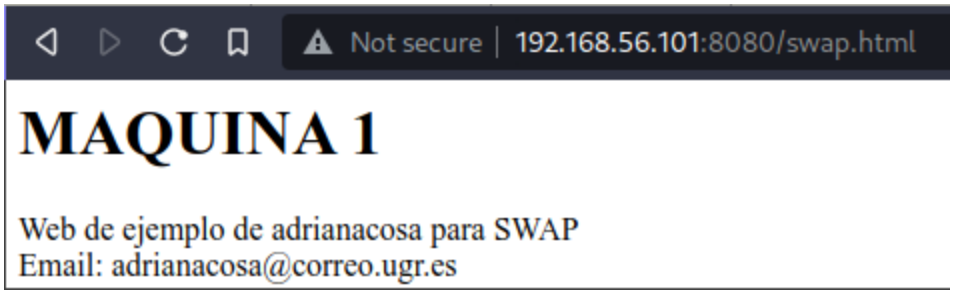
La primera configuración que vamos a editar va a ser la de los puertos. Éstos se editan en el archivo de configuración de apache llamado ports.conf situado en el directorio /etc/apache2/. En este archivo vamos a cambiar la línea en la que pone “Listen 80” por “Listen 8080”.

Tras esto no hemos acabado, tenemos que cambiar la configuración del host virtual en el archivo /etc/apache2/sites-enabled/000-default.conf cambiando también el valor 80 por 8080, quedando de la siguiente manera:

```
<VirtualHost *:8080>
```

Una vez tenemos estos valores cambiados, sólo queda reiniciar el servicio de apache y ya sólo podremos acceder a nuestro servidor web mediante el puerto 8080:





Configuración de hosts virtuales

Ahora vamos a pasar a crear los host virtuales para nuestro servidor web. En este caso voy a crear dos directorios distintos para dos hosts virtuales. Los voy a llamar virtualhost1 y virtualhost2, en los cuales dentro voy a tener una carpeta más llamada public_html donde guardaré los html. El árbol de directorios partiendo del directorio /var/www/html es el siguiente (usando la orden "tree" que acabo de instalar):

```
adrianacosa@m1-adrianacosa:/var/www/html$ sudo mkdir -p virtualhost1/public_html
adrianacosa@m1-adrianacosa:/var/www/html$ sudo mkdir -p virtualhost2/public_html
adrianacosa@m1-adrianacosa:/var/www/html$ tree
.
├── index.html
├── swap.html
├── virtualhost1
│   └── public_html
└── virtualhost2
    └── public_html

4 directories, 2 files
```

Como dentro de este directorio sólo tiene permisos el usuario root, voy a darle permisos a mi usuario (el cual es el administrador del sistema) para que pueda gestionar los directorios virtuales libremente. Esto se hace con la orden "chown" de la siguiente manera:

```
adrianacosa@m1-adrianacosa:/var/www/html$ sudo chown -R $USER:$USER /var/www/html/virtualhost1/publi
c_html
adrianacosa@m1-adrianacosa:/var/www/html$ sudo chown -R $USER:$USER /var/www/html/virtualhost2/publi
c_html
adrianacosa@m1-adrianacosa:/var/www/html$
```

Aclarar que no me pide la contraseña de superusuario debido a que justo antes acabo de instalar el paquete tree, pero debería pedirla. También aclarar que le doy permisos a $USER, que se refiere al usuario que tiene la sesión iniciada en ese momento (en mi caso "adrianacosa").

Pero esto no es todo con respecto a los permisos. También necesitamos los permisos de lectura en el directorio raíz del servicio apache, es decir, /var/www/html para que cualquier usuario pueda leer pero no escribir en dicho directorio. Para ello ejecutamos el siguiente comando:

```
adrianacosa@m1-adrianacosa:/var/www/html$ sudo chmod -R 755 /var/www/html/
```

Y por lo que respecta a los permisos ya habríamos terminado. Ahora ya podemos crear el contenido que mostrarán los hosts virtuales. Vamos a hacer unos ficheros html muy simples para que cuando entremos desde el navegador veamos claramente que estamos en un host virtual en concreto como los siguientes:

```
adrianacosa@m1-adrianacosa:/var/www/html$ cat virtualhost1/public_html/index.html
<html>
        <head>
                <title>HOST VIRTUAL NUMERO 1</title>
        </head>
        <body>
                <h1>Este es el host virtual numero uno</h1>
        </body>
</html>
adrianacosa@m1-adrianacosa:/var/www/html$ cat
cat        catchsegv  catman
adrianacosa@m1-adrianacosa:/var/www/html$ cat virtualhost2/public_html/index.html
<html>
        <head>
                <title>HOST VIRTUAL NUMERO 2</title>
        </head>
        <body>
                <h1>Este es el host virtual numero dos</h1>
        </body>
</html>
adrianacosa@m1-adrianacosa:/var/www/html$
```

El siguiente paso es copiar el archivo de configuración del host virtual por defecto, archivo llamado 000-default.conf, para configurar ambos hosts virtuales. Dentro de cada uno de los archivos de configuración cambiamos los parámetros ServerAdmin, ServerName, ServerAlias y DocumentRoot en función del host que estamos configurando. Por ejemplo, el archivo de configuración del virtualhost1 quedaría de la siguiente manera:

```
<VirtualHost *:8080>
        # The ServerName directive sets the request scheme, hostname and port that
        # the server uses to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the ServerName
        # specifies what hostname must appear in the request's Host: header to
        # match this virtual host. For the default virtual host (this file) this
        # value is not decisive as it is used as a last resort host regardless.
        # However, you must set it for any further virtual host explicitly.
        #ServerName www.example.com

        ServerAdmin webmaster@virtualhost1
        ServerName virtualhost1
        ServerAlias www.virtualhost1.com
        DocumentRoot /var/www/html/virtualhost1/public_html

        # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
        # error, crit, alert, emerg.
        # It is also possible to configure the loglevel for particular
        # modules, e.g.
        #LogLevel info ssl:warn

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

        # For most configuration files from conf-available/, which are
        # enabled or disabled at a global level, it is possible to
        # include a line for only one particular virtual host. For example the
        # following line enables the CGI configuration for this host only
        # after it has been globally disabled with "a2disconf".
        #Include conf-available/serve-cgi-bin.conf
</VirtualHost>
```

Y de igual manera para el virtualhost2 (igual pero cambiando 1 por 2). Si nos fijamos en la configuración hemos asignado un nombre a nuestro directorio y un alias para poder acceder por nombre al host. También hemos ajustado la ruta donde se encuentran cada uno de los archivos html que compondrán el directorio.

Lo siguiente que deberíamos hacer es deshabilitar el host virtual por defecto, que en este caso accedería al archivo de configuración por defecto anteriormente nombrado, y en su lugar habilitar ambos hosts virtuales para poder acceder a cada uno de ellos por separado. Para ello tenemos que ejecutar las siguientes órdenes:

```
adrianacosa@m1-adrianacosa:~$ sudo a2dissite 000-default.conf
Site 000-default disabled.
To activate the new configuration, you need to run:
  systemctl reload apache2
adrianacosa@m1-adrianacosa:~$ sudo a2ensite virtualhost1.conf
Enabling site virtualhost1.
To activate the new configuration, you need to run:
  systemctl reload apache2
adrianacosa@m1-adrianacosa:~$ sudo a2ensite virtualhost2.conf
Enabling site virtualhost2.
To activate the new configuration, you need to run:
  systemctl reload apache2
adrianacosa@m1-adrianacosa:~$ sudo systemctl reload apache2
adrianacosa@m1-adrianacosa:~$
```

Lo último que tendríamos que hacer antes de probar a conectarnos al servicio desde nuestro host es añadir al archivo /etc/hosts de éste último la resolución de los nombres que le hemos asignado a cada uno de los hosts virtuales. En este caso tenemos que añadir lo que sale a continuación en la siguiente foto:

```
192.168.56.102  www.virtualhost1.com
192.168.56.102  www.virtualhost2.com
```

Una vez hecho esto ya podemos acceder desde nuestro navegador:

Configuración de un directorio virtual mediante Alias

Ahora vamos a crear un directorio virtual. En primer lugar vamos a crear una carpeta para el directorio virtual dentro de mi home, es decir, de /home/adrianacosa. En este caso, al estar haciendo una configuración de prueba, voy a llamarlo directorio1.

Una vez tenemos el directorio vamos a crear un fichero html simple en él. En este caso voy a poner dentro del directorio 1 mi primer apellido:

```
adrianacosa@m1-adrianacosa:~$ cat directorio1/acosa.html
<html>
        <head>
                <title>Primer apellido</title>
        </head>
        <body>
                <h1>Mi primer apellido es Acosa</h1>
        </body>
</html>
adrianacosa@m1-adrianacosa:~$
```

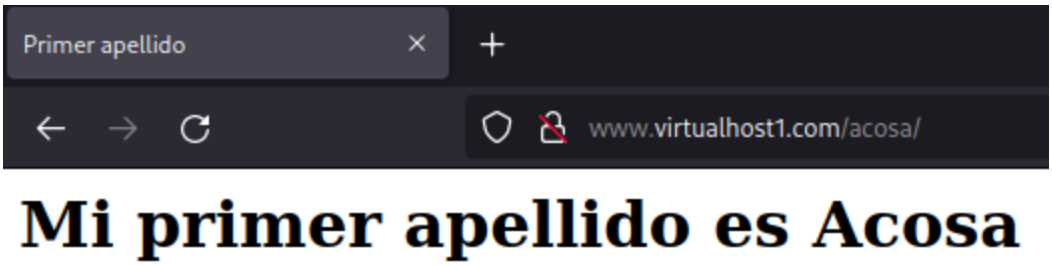
También necesitamos tener los permisos necesarios sobre el archivo que acabamos de crear. En este caso las órdenes que hay que ejecutar son muy parecidas a las ejecutadas anteriormente en el directorio /var/www/html:

```
adrianacosa@m1-adrianacosa:~$ sudo chmod -R 755 /home/adrianacosa
adrianacosa@m1-adrianacosa:~$ sudo chmod -R 644 /home/adrianacosa/directorio1/acosa.html
```

Ahora ya podemos pasar a configurar los directorios virtuales en apache. Para ello lo primero que tenemos que hacer es ir al archivo de configuración de uno de nuestros hosts virtuales, el cual en este caso por simplificar sólo voy a hacerlo en virtualhost1.conf. Aquí debemos establecer un alias con el nombre que le queramos dar y asignarle a dicho alias la ruta del archivo al que queremos acceder, el cual en nuestro caso será el directorio /home/adrianacosa/directorio1. Luego el archivo de configuración de virtualhost1 debería quedar de la siguiente manera:

```
<VirtualHost *:80>
        # The ServerName directive sets the request scheme, hostname and port that
        # the server uses to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the ServerName
        # specifies what hostname must appear in the request's Host: header to
        # match this virtual host. For the default virtual host (this file) this
        # value is not decisive as it is used as a last resort host regardless.
        # However, you must set it for any further virtual host explicitly.
        #ServerName www.example.com

        ServerAdmin webmaster@virtualhost1
        ServerName virtualhost1.com
        ServerAlias www.virtualhost1.com
        DocumentRoot /var/www/html/virtualhost1/public_html

        Alias /acosa "/home/adrianacosa/directorio1"
        <Directory /home/adrianacosa/directorio1>
                DirectoryIndex acosa.html
                Require all granted
        </Directory>
```

Donde especificamos que vamos a asignarle a /acosa la ruta que aparece a continuación entre comillas. A ese directorio tenemos que decirle que su índice es acosa.html con la directiva DirectoryIndex y también para que no haya ningún problema de accesos he añadido la línea de “Require all granted” para que pueda acceder quien quiera. El resultado tras reiniciar el servicio es el siguiente:



La ruta /acosa nos devuelve el archivo /home/adrianacosa/directorio1/acosa.html.

Instalación y uso de cURL

CURL es un paquete que ya viene instalado por defecto en los servidores Ubuntu Server 20.04. Su instalación, en el caso de que no estuviese instalado, sería mediante el comando "apt-get install curl". Para comprobar que efectivamente lo tenemos instalado, basta con ejecutar el comando "curl --version", el cual al tenerlo instalado nos proporciona la siguiente salida:

```
adrianacosa@m1-adrianacosa:~$ curl --version
curl 7.68.0 (x86_64-pc-linux-gnu) libcurl/7.68.0 OpenSSL/1.1.1f zlib/1.2.11 brotli/1.0.7 libidn2/2.2
.0 libpsl/0.21.0 (+libidn2/2.2.0) libssh/0.9.3/openssl/zlib nghttp2/1.40.0 librtmp/2.3
Release-Date: 2020-01-08
Protocols: dict file ftp ftps gopher http https imap imaps ldap ldaps pop3 pop3s rtmp rtsp scp sftp
smb smbs smtp smtps telnet tftp
Features: AsynchDNS brotli GSS-API HTTP2 HTTPS-proxy IDN IPv6 Kerberos Largefile libz NTLM NTLM_WB P
SL SPNEGO SSL TLS-SRP UnixSockets
adrianacosa@m1-adrianacosa:~$
```

Salida que nos dice que tenemos instalada la versión 7.68.0 del programa.

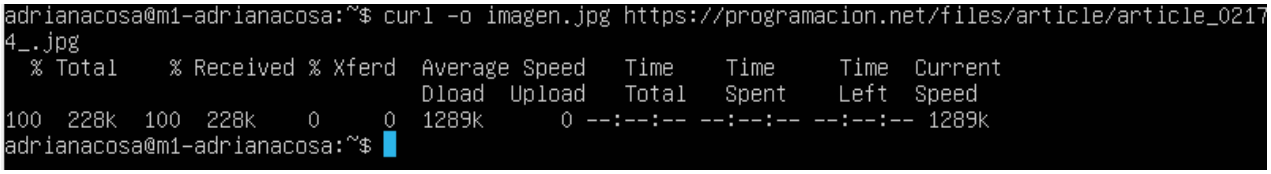
Su uso es muy sencillo, simplemente si queremos ver el contenido de una página web basta con pasarle la URL de dicha web y nos devolverá el html de dicha página. Por ejemplo, si le damos la URL del host virtual creado anteriormente nos devolverá por la salida estándar:

```
adrianacosa@m1-adrianacosa:~$ curl www.virtualhost1.com
<html>
        <head>
                <title>HOST VIRTUAL NUMERO 1</title>
        </head>
        <body>
                <h1>Este es el host virtual numero uno</h1>
        </body>
</html>
adrianacosa@m1-adrianacosa:~$
```

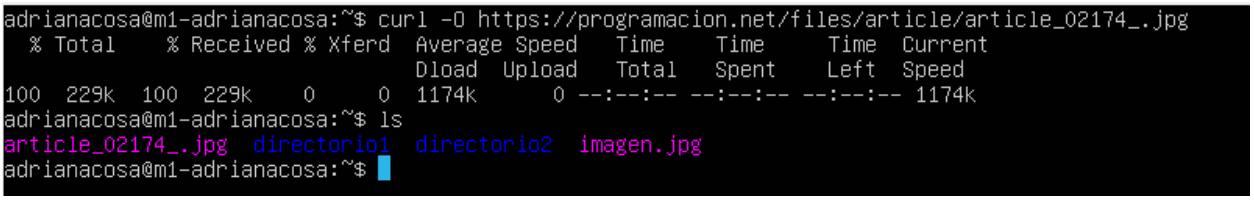
La salida que nos proporciona es el html en bruto que proporciona la url www.virtualhost1.com.

Descarga de archivos con cURL

Existen también algunas opciones que nos permiten manejar hacia dónde queremos que vaya dicha salida. Una opción es la opción -o, a la que se le indica dónde queremos que nos guarde esa salida. De ésta manera podemos, por ejemplo, descargarnos una imagen de internet si le proporcionamos una URL de ella como se puede ver en la siguiente captura:



Luego nos encontramos con la opción -O, que hace algo parecido a -o, y es que descarga un archivo directamente sin especificarle la ruta donde lo queramos guardar:



Visualización de peticiones HTTP con cURL

Podemos ver información sobre el protocolo HTTP con el uso de curl. Para poder ver la información del protocolo podemos hacer uso de la opción -v o --verbose:

```
adrianacosa@m1-adrianacosa:~$ curl -v www.virtualhost1.com
*   Trying 192.168.56.102:80...
* TCP_NODELAY set
* Connected to www.virtualhost1.com (192.168.56.102) port 80 (#0)
> GET / HTTP/1.1
> Host: www.virtualhost1.com
> User-Agent: curl/7.68.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Sat, 05 Mar 2022 18:29:12 GMT
< Server: Apache/2.4.41 (Ubuntu)
< Last-Modified: Sat, 05 Mar 2022 15:23:04 GMT
< ETag: "86-5d97a352bdd22"
< Accept-Ranges: bytes
< Content-Length: 134
< Vary: Accept-Encoding
< Content-Type: text/html
<
<html>
        <head>
                <title>HOST VIRTUAL NUMERO 1</title>
        </head>
        <body>
                <h1>Este es el host virtual numero uno</h1>
        </body>
</html>
* Connection #0 to host www.virtualhost1.com left intact
adrianacosa@m1-adrianacosa:~$
```

Visualización de cookies con cURL

Una función muy interesante de curl es que podemos obtener las cookies al acceder a una URL con la opción -c o --cookie-jar. Un ejemplo de su uso es el siguiente aplicado a google.com:

Bibliografía

https://serverpilot.io/docs/how-to-use-ssh-public-key-authentication/

https://ostechnix.com/configure-apache-virtual-hosts-ubuntu-part-1/

https://profesorcyber.blogspot.com/2017/03/directorios-virtuales-en-apache.html

https://linux.die.net/man/1/curl