

# SERVIDORES WEB DE ALTAS PRESTACIONES

## PRÁCTICA 4: Asegurar la granja web

### Autor

Miguel Ángel Pérez Díaz



Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

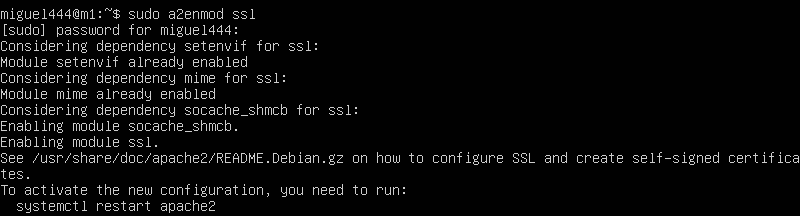
—

Granada,2020

1. ***Crear e instalar en la máquina M1 un certificado SSL autofirmado para configurar el acceso HTTPS al servidor. Se debe comprobar que el servidor acepta tanto el tráfico HTTP como el HTTPS.***

Para llevar a cabo la tarea propuesta debemos seguir una serie de pasos que voy a detallar a continuación:

* **Activamos el módulo SSL de Apache con el comando**:

*sudo a2enmod ssl*

* Como bien indica la salida de ejecutar el comando anterior: **reiniciamos Apache:**

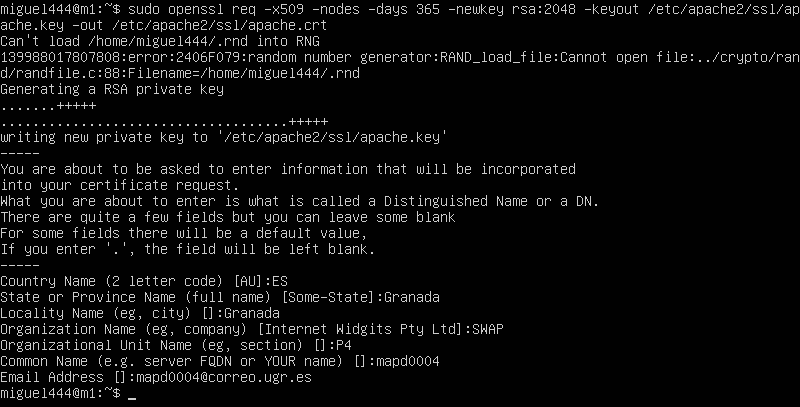
sudo service apache2 restart

* **Creamos el directorio donde vamos a generar el correspondiente certificado SSL**:

sudo mkdir /etc/apache2/ssl

* Finalmente **generamos el certificado SSL** indicándole los siguientes parámetros:

sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/apache2/ssl/apache.key -out /etc/apache2/ssl/apache.crt

* **openssl**: es la herramienta de línea de comandos básica para crear y administrar certificados, claves, y otros archivos de OpenSSL.
* **req**: este subcomando especifica que deseamos usar la administración de la solicitud de firma de certificados (CSR) X.509. El “X.509” es un estándar de infraestructura de claves públicas al que se adecuan SSL y TLS para la administración de claves y certificados a través de él. Queremos crear un nuevo certificado X.509, por lo que usaremos este subcomando.
* **-x509**: modifica aún más el subcomando anterior al indicar a la utilidad que deseamos crear un certificado autofirmado en lugar de generar una solicitud de firma de certificados, como normalmente sucede.
* **-nodes**: indica a OpenSSL que omita la opción para proteger nuestro certificado con una frase de contraseña. Necesitamos que Apache pueda leer el archivo, sin intervención del usuario, cuando se inicie el servidor. Una frase de contraseña evitaría que esto suceda porque tendríamos que ingresarla tras cada reinicio.
* **-days 365**: esta opción establece el tiempo durante el cual el certificado se considerará válido. En este caso, lo configuramos por un año.
* **-newkey rsa:2048**: especifica que deseamos generar un nuevo certificado y una nueva clave al mismo tiempo. No creamos la clave que se requiere para firmar el certificado en un paso anterior, por lo que debemos crearla junto con el certificado. La parte rsa:2048 le indica que cree una clave RSA de 2048 bits de extensión.
* **-keyout**: esta línea indica a OpenSSL dónde colocar el archivo de clave privada generado que estamos creando.
* **-out:** indica a OpenSSL dónde colocar el certificado que creamos.

Se puede observar que se me ha pedido los datos citados en el guion de prácticas para configurar el certificado del dominio y se han rellenado con mis pertinentes datos:

* Nombre de país: ES
* Provincia: Granada
* Localidad: Granada
* Organización: SWAP
* Organización sección: P4
* Nombre: mapd0004
* Email: mapd0004@correo.ugr.es
* El siguiente paso sería m**odificar el fichero** */etc/apache2/sites-available/default-ssl.conf***y agregar la ruta de los certificados anteriormente generados**, para ello se añade:

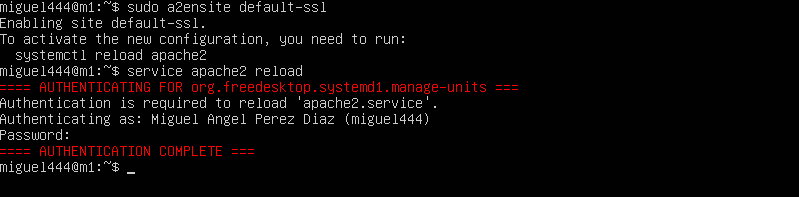
SSLCertificateFile /etc/apache2/ssl/apache.crt

SSLCertificateKeyFile /etc/apache2/ssl/apache.key

Se puede observar que ha sido necesario comentar las líneas de más abajo donde se especificaba otra ruta para los certificados (También podría haberse modificado la ruta existente por la nueva).

* Por último, solamente nos **faltaría activar el servicio** *default-ssl* **y reiniciar Apache**:

sudo a2ensite default-ssl

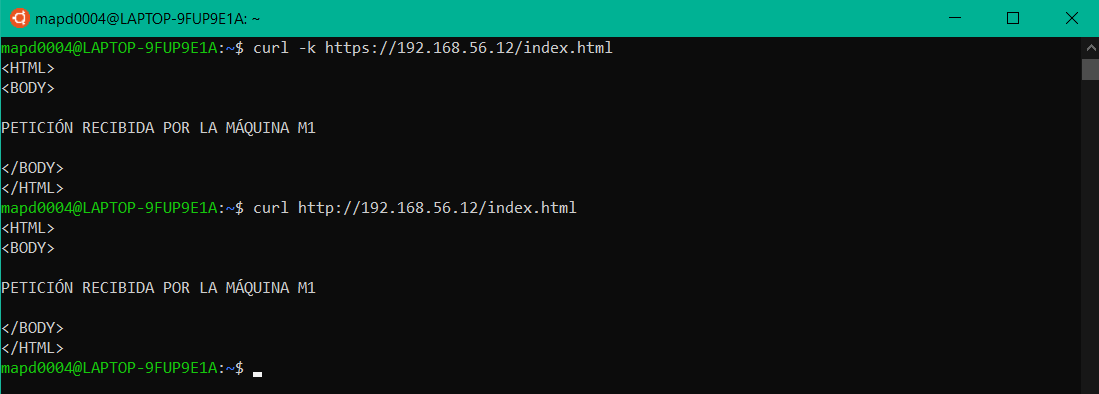
service apache2 reload

Con esto último ya tendríamos todo lo necesario para hacer peticiones a nuestro servidor tanto por HTTP como por HTTPS, tras haber generado nuestro certificado autofirmado.

Para ello vamos a realizar peticiones a la máquina M1 desde M4, siendo esta última mi máquina anfitriona. Para llevar a cabo las peticiones por HTTPS es necesario añadir el parámetro *-k* a la llamada de *cURL* como se observa en la siguiente imagen:

curl -k https://192.168.56.12/index.html

curl http://192.168.56.12/index.html



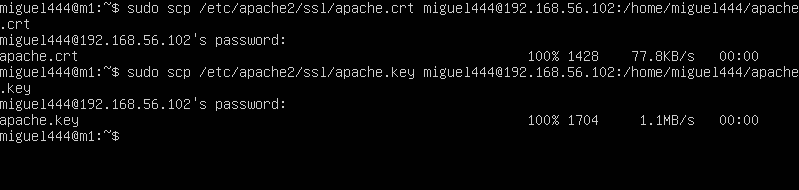
Por seguir con la dinámica actual se va a continuar con el primero de los ejercicios opcionales propuestos, por tanto para este apartado debemos copiar el certificado generado tanto a la máquina M2 como al balanceador M3 y comprobar que M3 puede balancear correctamente tanto el tráfico HTTP como HTTPS.

Para copiar el certificado generado a las dos máquinas vamos a utilizar el comando *scp* desde la máquina M1,por lo que bastaría con:

*Para la máquina M2*

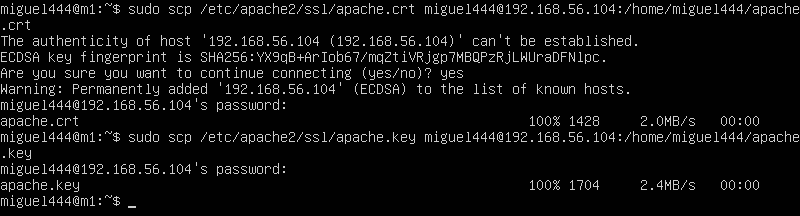
*sudo scp /home/miguel444/apache.crt miguel444@192.168.56.102:/home/miguel444/apache.crt*

*sudo scp /home/miguel444/apache.key miguel444@192.168.56.102:/home/miguel444/apache.key*

**

*Para el balanceador M3*

*sudo scp /home/miguel444/apache.crt miguel444@192.168.56.104:/home/miguel444/apache.crt*

*sudo scp /home/miguel444/apache.key miguel444@192.168.56.104:/home/miguel444/apache.key*

Tras copiar el certificado tanto a la máquina M2 como al balanceador M3 debemos seguir lo mismos pasos que para M1 en M2:

sudo mkdir /etc/apache2/ssl

sudo mv /home/miguel444/apache\* /etc/apache2/ssl

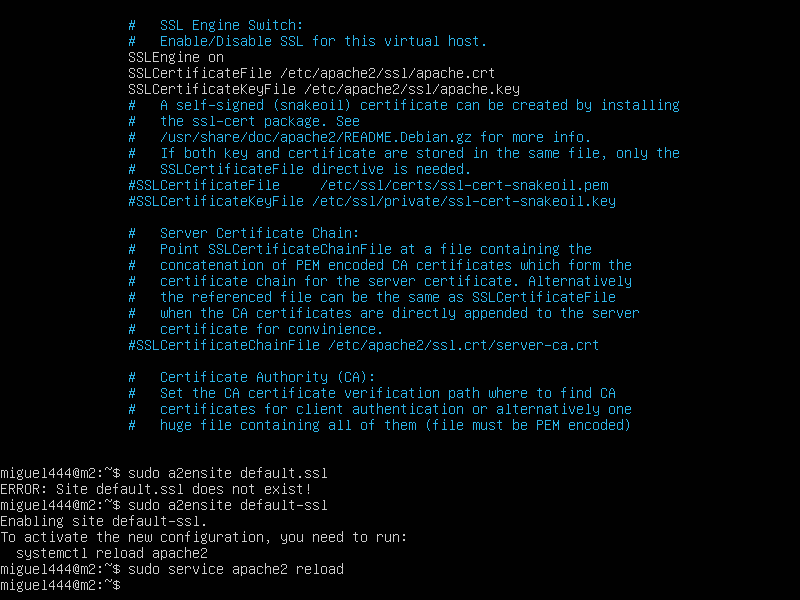
sudo a2enmod ssl & sudo service apache2 restart

sudo nano /etc/apache2/sites-available/default-ssl.conf

SSLCertificateFile /etc/apache2/ssl/apache.crt SSLCertificateKeyFile/etc/apache2/ssl/apache.key

sudo a2ensite default.ssl

Sudo service apache2 reload

**Máquina M2:**

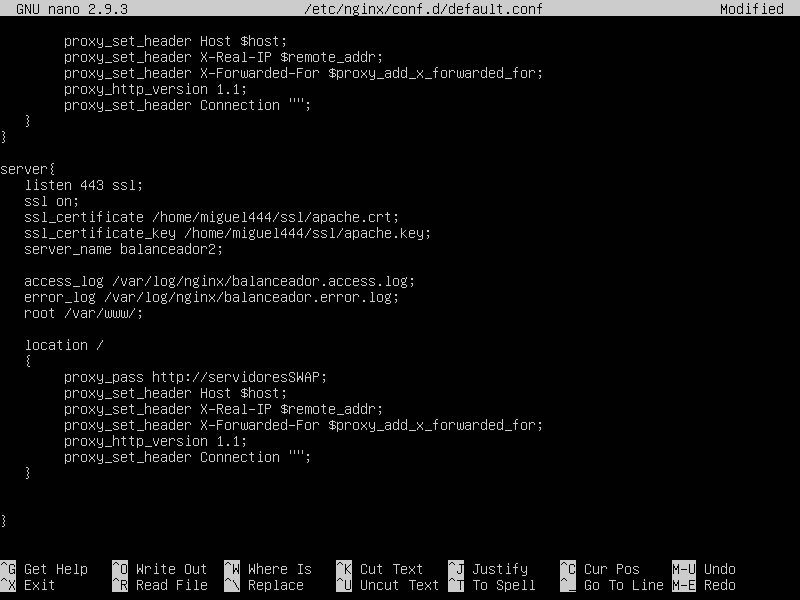
Además, para el balanceador debemos crear un nuevo ‘server’ en el archivo de configuración de Nginx pero en este caso debemos introducir los datos correspondientes a HTTPS y el certificado:

listen 443 ssl;

ssl on;

ssl\_certificate /home/usuario/ssl/apache.crt;

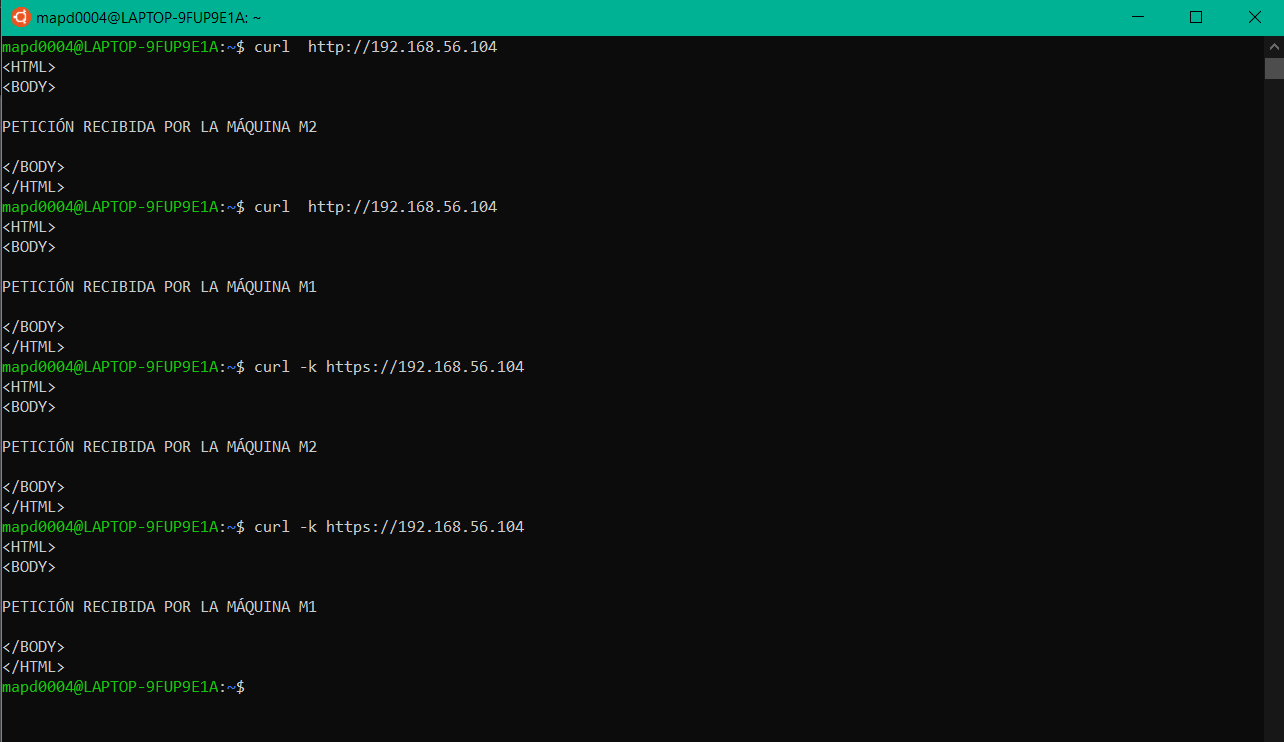
ssl\_certificate\_key /home/usuario/ssl/apache.key;



Una vez disponemos del certificado autofirmado tanto en los servidores web (M1 y M2) como en la máquina balanceadora (M3) vamos a comprobar el correcto funcionamiento del balanceador haciéndole peticiones tanto HTTP como HTTPS y observar si se obtienen las respuestas deseadas a dichas peticiones.

Para ello como se indicó anteriormente utilizaremos la herramienta cURL pero en este caso hacia el balanceador desde nuestra máquina anfitrión:

curl -k https://192.168.56.104

**curl http://192.168.56.104

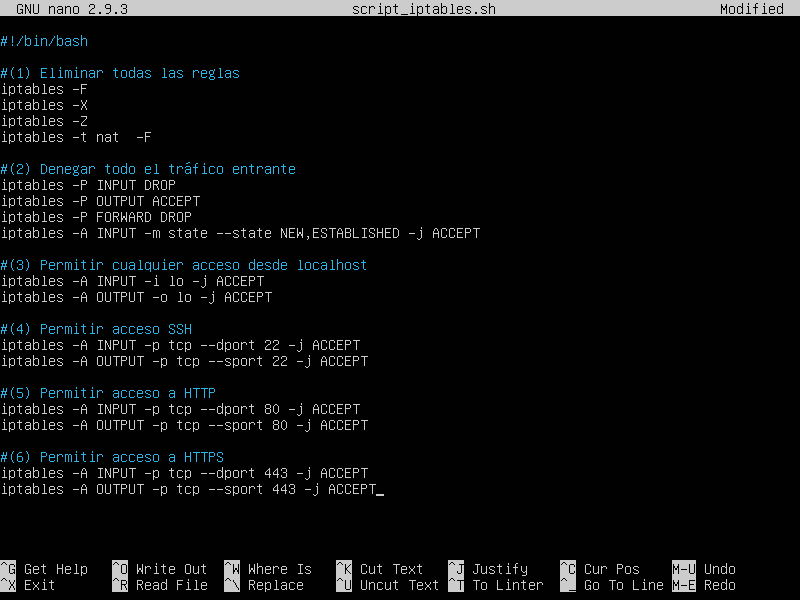
1. ***Configurar las reglas del cortafuegos con IPTABLES en uno de los servidores web finales (M1 o M2) para asegurarlo, permitiendo el acceso por los puertos de HTTP y HTTPS a dicho servidor. Como se indica, esta configuración se hará en una de las máquinas servidoras finales (p.ej. en la máquina M1), y se debe poner en un script con las reglas del cortafuegos.***

En esta sección debemos crear un script con reglas IPTABLES para permitir el tráfico HTTP y HTTPS en nuestra máquina M1. A continuación voy a detallar un poco las reglas utilizadas en el script para después comprobar su funcionamiento y ver si se obtiene una respuesta correcta del cortafuegos.

Para crear el script se ha generado un fichero en formato .sh y se ha declarado en la primera línea que sea ejecutable en el terminal:

sudo nano script\_iptables.sh

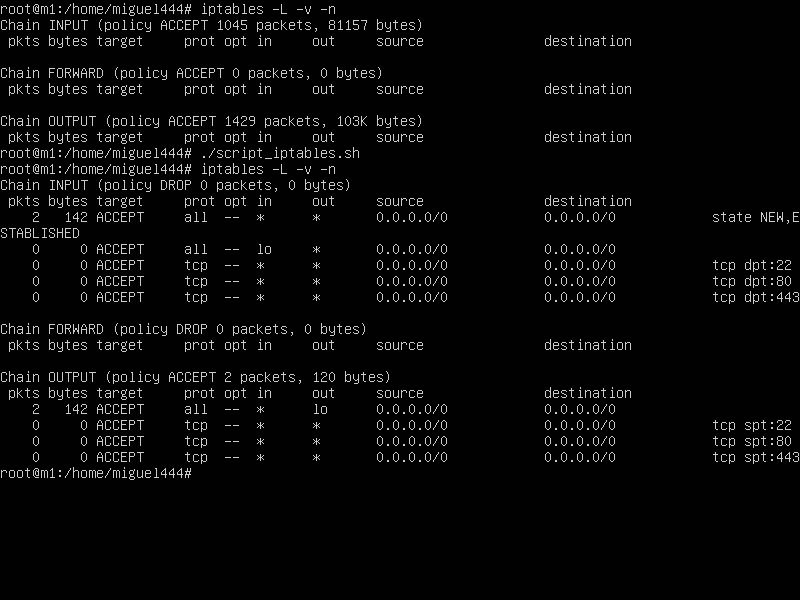
#!/bin/bash

Las reglas utilizadas en el script son las siguientes:

Damos permisos de ejecución al script con *chmod a+x script\_iptables.sh* y ejecutamos el script con *sudo ./script\_iptables.sh.*

Vamos a comprobar si el script creado aplica realmente las correspondientes reglas del cortafuegos, para ello podemos usar el siguiente comando para ver las reglas actuales:

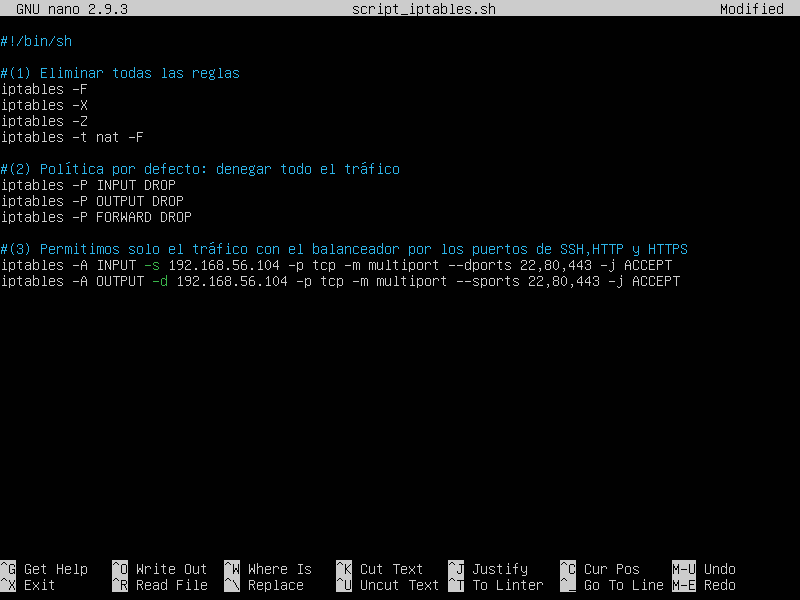
iptables -L -n -v

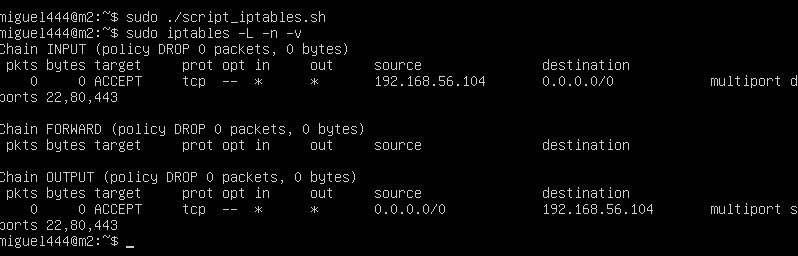


Tras finalizar la tarea vamos a continuar con el ejercicio opcional correspondiente a esta parte del cortafuegos en la que debemos configurar M1 y M2 para que solo acepte peticiones del balanceador M3 y que el balanceador solo reciba peticiones HTTP y HTTPS. Además habrá que guardar dicha configuración del cortafuegos para que se ejecute en cada máquina durante el arranque del sistema.

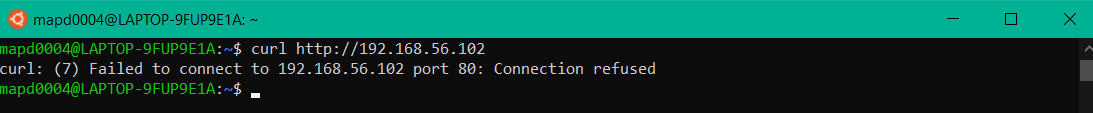
Inicialmente comenzaremos configurando tanto M1 como M2 para que solo acepten peticiones provenientes de M3, para ello debemos modificar el script generado para M1 y crear uno nuevo para M2 en donde definiremos un parámetro indicando la dirección IP del balanceador M3.

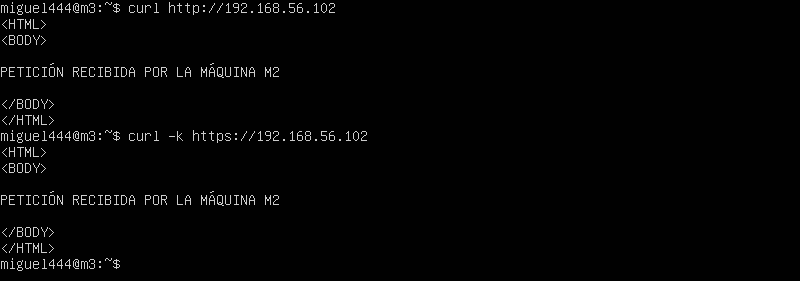
En este nuevo script hay que especificar tanto la IP de origen como de destino, en nuestro caso la del balanceador. Para especificar la dirección IP se añade un nuevo parámetro a las reglas anteriormente vistas: -s (IP origen) y -d (IP destino).



Al igual que antes daremos permisos de ejecución al script y lo ejecutaremos para después comprobar si se han añadido las reglas correctamente.

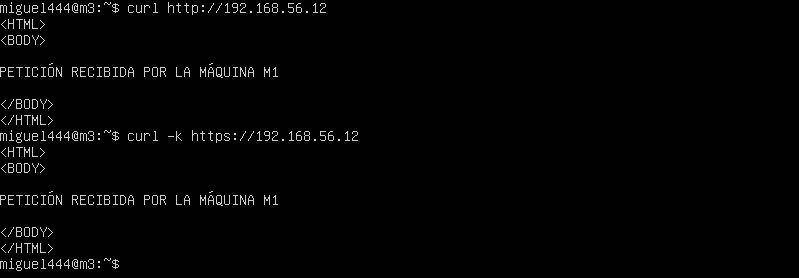
Vamos a comprobar si la máquina M2 solo acepta realmente peticiones desde M3 y deniega peticiones desde mi máquina anfitrión.

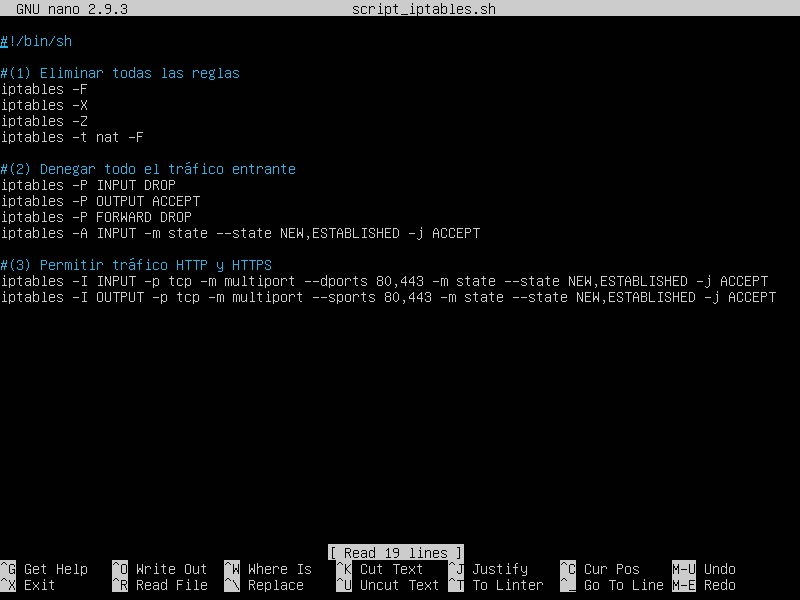
*MÁQUINA ANFITRIÓN*

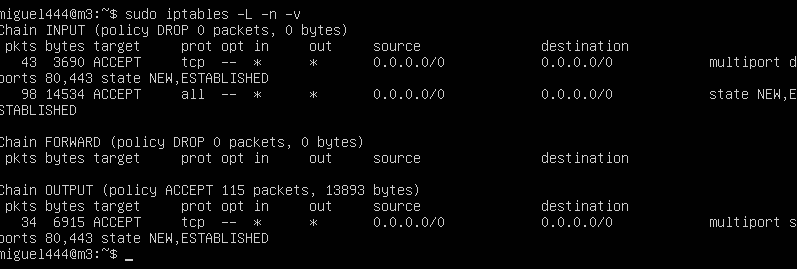
*MÁQUINA BALANCEADORA M3*

Una vez se ha comprobado que funciona correctamente para M2 podemos aplicar el mismo script para M1 y añadir las mismas reglas, así solo aceptarán peticiones del balanceador M3.

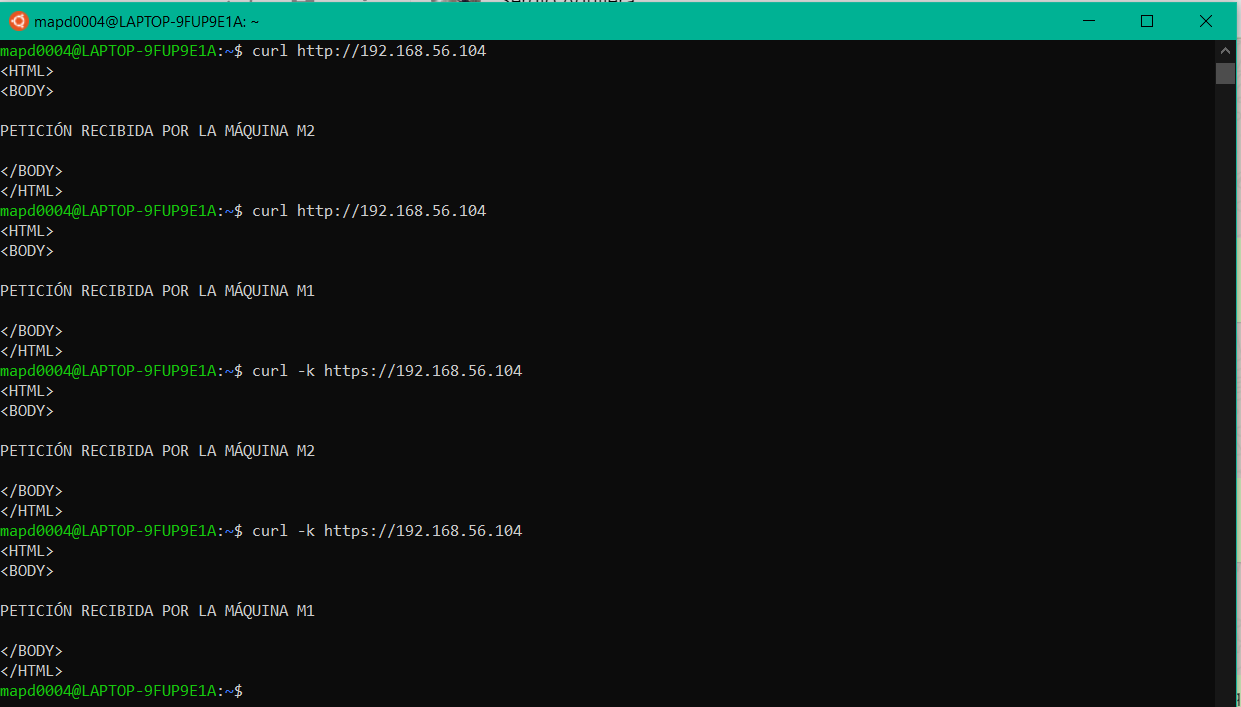




Terminada esta parte vamos a proceder a añadir las correspondientes reglas de cortafuegos en el balanceador para que solo acepte peticiones HTTP y HTTPS. Al igual que para los servidores de nuestra granja web, debemos crear un script con las reglas a añadir para luego ejecutarlo:

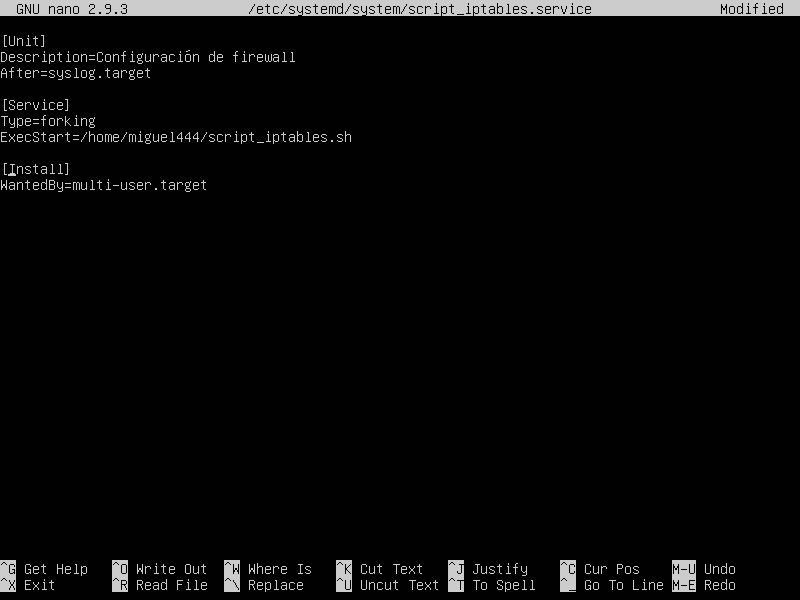
Posteriormente le damos permisos de ejecución y lo ejecutamos para observar si se han aplicado las reglas correctamente:

A continuación, veremos si el balanceador realiza su función correctamente, para ello realizaremos peticiones a él por HTTP y HTTPS.



Para finalizar la tarea solo faltaría configurar las máquinas para que el correspondiente script se ejecute durante el arranque del sistema. Cabe mencionar que se va a describir el proceso para la máquina M1, este mismo proceso se ha repetido para el resto de máquinas: M2 y M3.

Para ello inicialmente debemos crear un nuevo servicio en */etc/systemd/system/* llamado *script\_iptables.service* en mi caso :



En la sección ExecStart=/home/miguel44/script\_iptables.sh se da la ruta en dónde realmente está el script a ejecutar.  Y el tipo de demonio que va a ser en:  
  
[Service]  
Type=forking  
  
Por último, la sección [Install] nos dice en qué runlevel se ejecutará. En el modo multiusuario o en el clásico sysv, que sería el runlevel 3.  
  
[Install]  
WantedBy=multi-user.target  
  
Con esto ya tenemos todo listo para agregar nuestro script al inicio del sistema.

<https://www.ubuntuleon.com/2016/10/cargar-un-script-al-inicio-del-sistema.html>

Una vez hemos declarado el servicio, debemos darle los permisos necesarios para su ejecución: **rwxr-xr-x**

sudo chmod 744 script\_iptables.sh

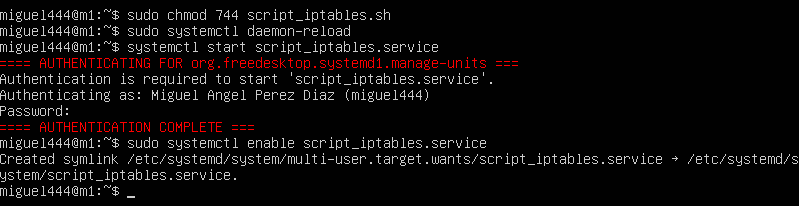
Una vez configurado todo, antes de ejecutar la nueva configuración, hay que refrescar systemd para que lea de nuevo los servicios:

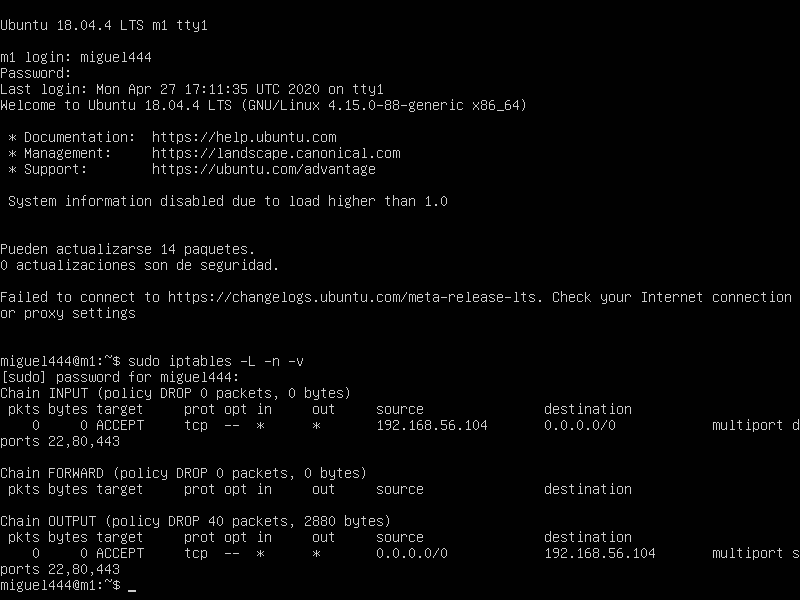
sudo systemctl daemon-reload

Levantamos el servicio:

sudo systemctl start script\_iptables.service

Habilitamos el servicio para que inicie durante el arranque del sistema:

sudo systemctl enable script\_iptables.service

Finalmente comprobamos que está activo al iniciar la máquina virtual: