INFORME PRÁCTICA 3

Julio César Velásquez Cárdenas 1397896
David Sánchez González 1401641
Asignatura: Gestió i Administració de Xarxes

Instalación de Apache

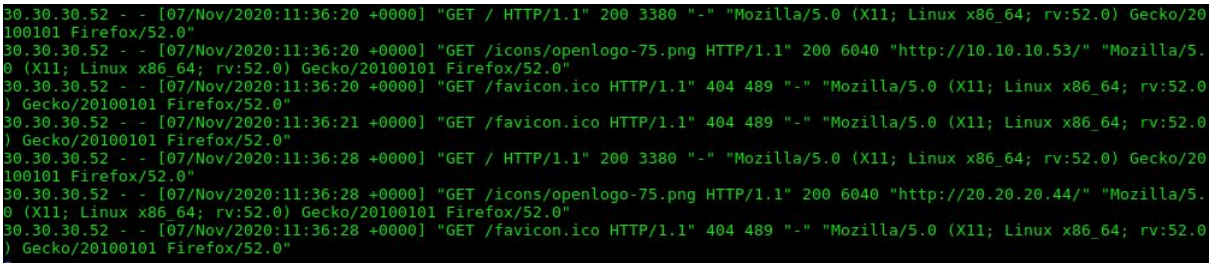
Para poder utilizar una máquina como servidor Apache necesitamos el paquete *apache2*. Una vez instalado, solo queda iniciar el servicio con *systemctl start apache2*. Lo que nos dará la siguiente página web de prueba en *localhost* para la que hace de server, en este caso A o introduciendo cualquiera de las dos ip de A desde las máquinas B y C o usar el dominio local establecido en la página anterior, en nuestro caso *masterI.gax.org* o *masterM.gax.org*.

Como se puede ver, ambas máquinas pueden acceder a la página web por defecto, por lo que podemos pasar al siguiente punto, modificar el HTML y comprobar que se actualiza.

Como se puede ver, hemos optado por hacer unos cambios sencillos, como cambiar el nombre de la pestaña y el título de la página por *Apache2 en la máquina A*.

Echando un vistazo en el fichero que controla los accesos al servidor (*access.log* en */var/log/apache2*), es posible ver como la conexión realizada ha sido correcta,

además de poder ver que la IP que ha realizado la conexión con el servidor es la de la máquina C (*30.30.30.52*).

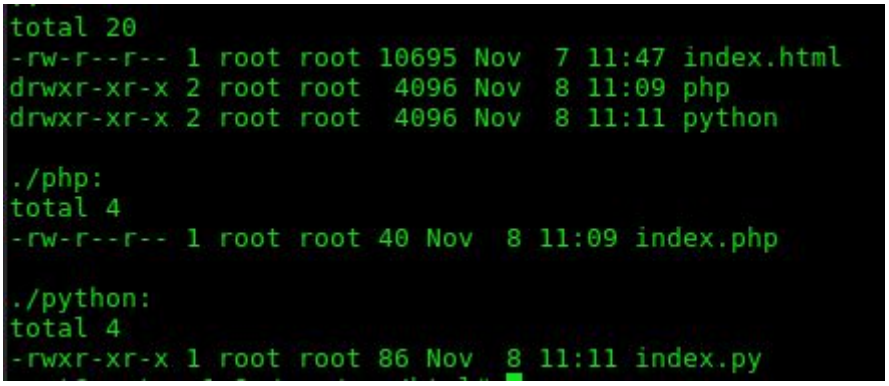


Comprobando a acceder desde la nueva máquina D, podemos ver que podemos acceder desde la IP de A en la interfaz de Internet, pero no desde la IP de la interfaz Middle.

Módulos de Apache (PHP y Python)

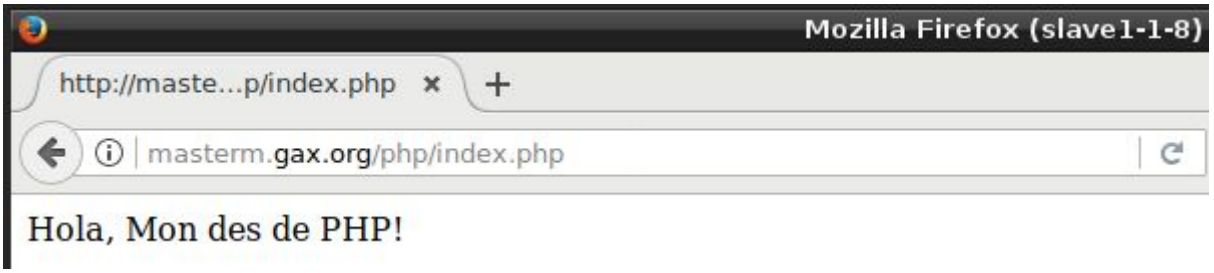
Antes de realizar la instalación de los módulos, debemos instalar primero los paquetes necesarios de ambos módulos, en el caso de Python viene instalado por defecto en los sistemas Unix. Para PHP deberemos hacer un *apt install php* para instalar lo necesario.

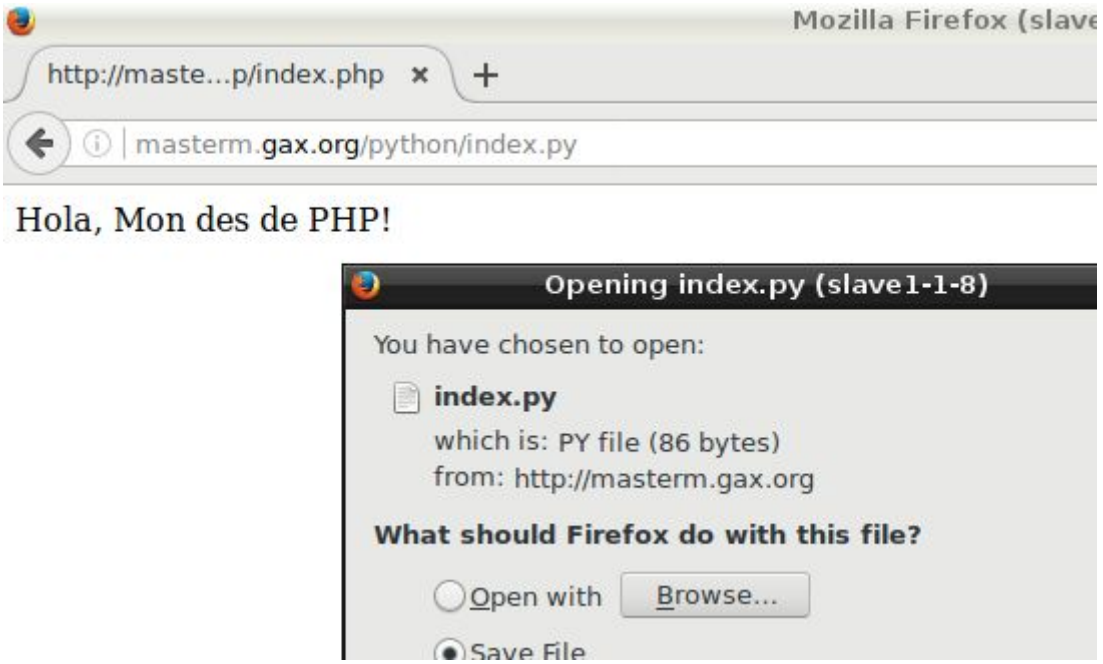
Una vez hecho, solo queda instalar los módulos de ambos para Apache. El nombre de los mismos son *libapache2-mod-python* y *libapache2-mod-python*. Una vez instalados, creamos los ficheros de ejemplo para Python y PHP especificados en el ejemplo.



En la captura de pantalla anterior puede verse como quedará dispuesto el directorio */var/www/html* después de la creación de los scripts, con *index.py* teniendo permisos de ejecución.

Hecho realizamos el comando *a2enmod cgi*, para que sea posible para el servidor Apache interpretar en este caso el script de Python. Reiniciamos el servidor para que se apliquen los cambios.





Como se puede ver, nuestro servidor de Apache puede procesar los ficheros de PHP pero no los de Python, ofreciendo solo la descarga de los mismos.

Hosts virtuales por IP

Para comenzar este apartado debemos modificar la configuración de puertos de Apache, para ello debemos hacer cambios en el fichero *ports.conf* que se encuentra en */etc/apache2*.

```
# If you just change the port or add more ports here, you will likely also
# have to change the VirtualHost statement in
# /etc/apache2/sites-enabled/000-default.conf

Listen 80

NameVirtualHost 10.10.10.53:80
NameVirtualHost 20.20.20.44:80
```

Como se puede ver hemos añadido las dos IP de las que dispone nuestro server (máquina A) como hosts virtuales.

También deberemos cambiar la configuración de forma individual para cada host, para ello duplicamos el archivo *000-default.conf* en */etc/apache2/sites-available*. Lo duplicaremos dos veces, uno en *outside.conf* y otro en *inside.conf*.

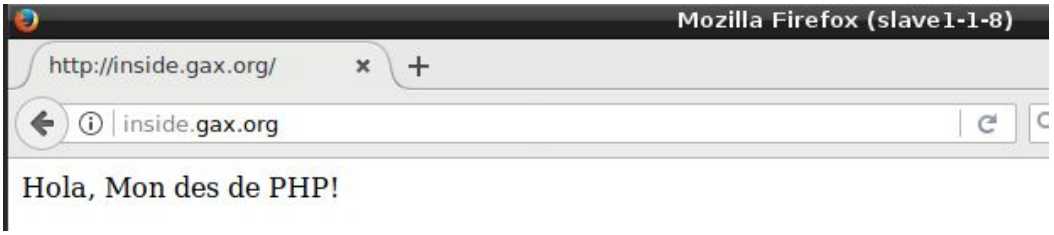
```
<VirtualHost 10.10.10.53:80>
        ServerName outside.gax.org

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html/python

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

        <Directory /var/www/html/python>
                DirectoryIndex index.py
                Options Indexes FollowSymLinks MultiViews ExecCGI
                AllowOverride None
                Order allow,deny
                allow from all
                AddHandler cgi-script .py
        </Directory>
</VirtualHost>
```

```
<VirtualHost 20.20.20.44:80>
        ServerName inside.gax.org

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html/php

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

        <Directory /var/www/html/php>
                DirectoryIndex index.php
                Options Indexes FollowSymLinks MultiViews ExecCGI
                AllowOverride None
                Order allow,deny
                allow from all
        </Directory>

</VirtualHost>
```

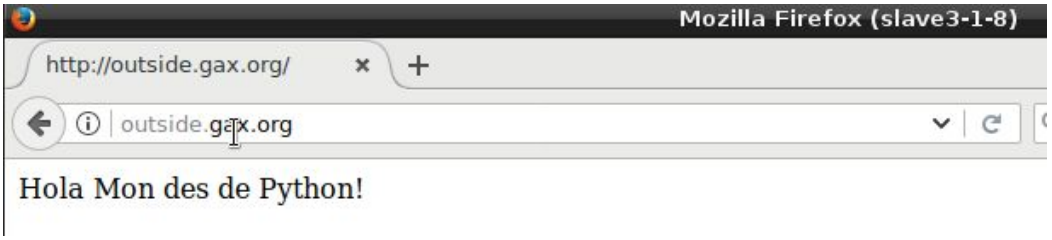
Quedando tal como se muestra en las capturas anteriores para Python y PHP respectivamente. Reiniciamos Apache para aplicar los cambios.

Deshabilitamos la configuración por defecto con *a2dissite 000-default.conf* en */etc/apache2/sites-available*, y habilitamos las nuevas configuraciones para los hosts virtuales.

En nuestro caso hemos empezado por activar *inside.conf*, correspondiente al sitio escrito en PHP, ejecutando el comando *a2ensite inside.conf*. Como se podrá ver en la siguiente captura, la conexión desde la máquina B es exitosa.



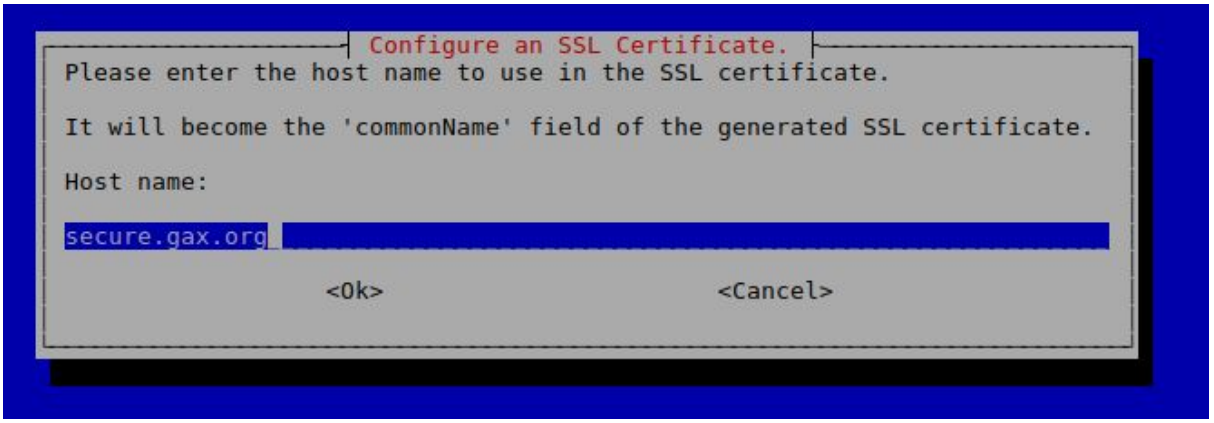
A continuación habilitaremos el sitio para Python, con *a2ensite outside.conf* y comprobaremos la conexión desde la máquina D a la interfaz Internet de la máquina A.



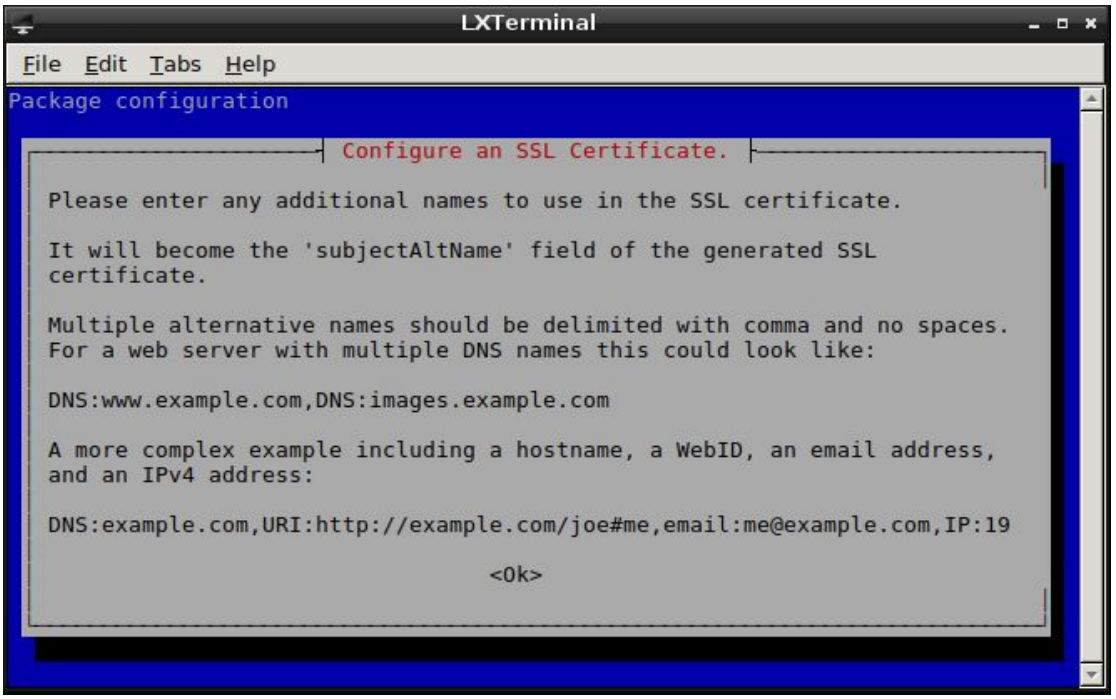
Con lo cual, gracias a los Virtual Hosts, una misma máquina está sirviendo dos páginas diferentes según la IP a la que se conecte otra máquina.

Certificado y HTTPS

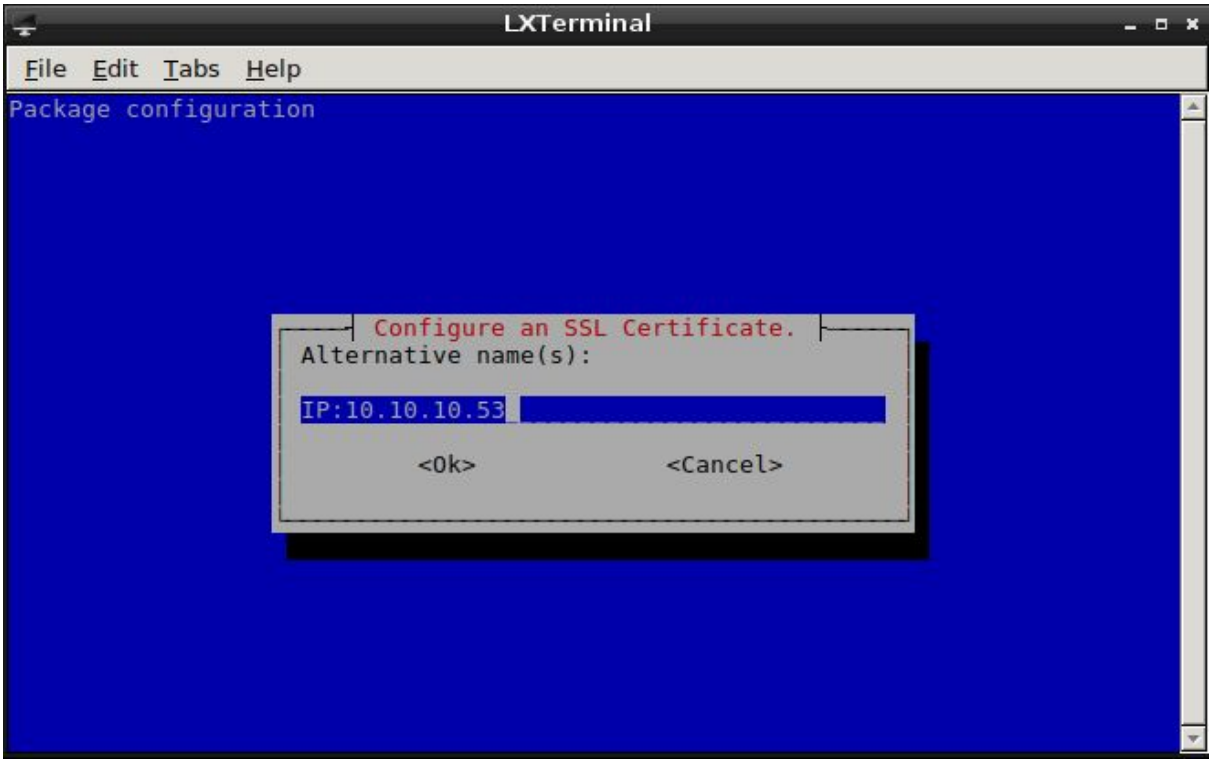
Para la creación del certificado, hemos de usar el comando *make-ssl-cert /usr/share/ssl-cert/ssleay.cnf /etc/ssl/private/secure.crt* donde *secure.crt* será el nombre de nuestro certificado una vez creado. Al ejecutar este comando, se abrirá una pantalla donde debemos introducir el dominio usado en el certificado.



Después se nos mostrará el siguiente aviso que nos indicará que podemos introducir un nombre alternativo.



Al cual haremos caso e introduciremos la dirección IP de la máquina.



Usamos el comando *a2enmod ssl* para habilitar el ssl en nuestro servidor apache y tal como nos indica, reiniciamos el apache.

```
root@master-1-8:/var/www/html# a2enmod ssl
Considering dependency setenvif for ssl:
Module setenvif already enabled
Considering dependency mime for ssl:
Module mime already enabled
Considering dependency socache_shmcb for ssl:
Enabling module socache_shmcb.
Enabling module ssl.
See /usr/share/doc/apache2/README.Debian.gz on how to configure SSL and create s
elf-signed certificates.
To activate the new configuration, you need to run:
  systemctl restart apache2
root@master-1-8:/var/www/html#
```

Después de reiniciar el apache, creamos un directorio nuevo en /var/www/html llamado secure y copiamos el archivo index.html dentro para asegurarnos que realmente está encargando el certificado seguro.

Modificamos el archivo default-ssl.conf en el directorio */etc/apache2/sites-available/* para indicar dónde está nuestro certificado y el directorio al cual apunta al entrar a esa dirección.

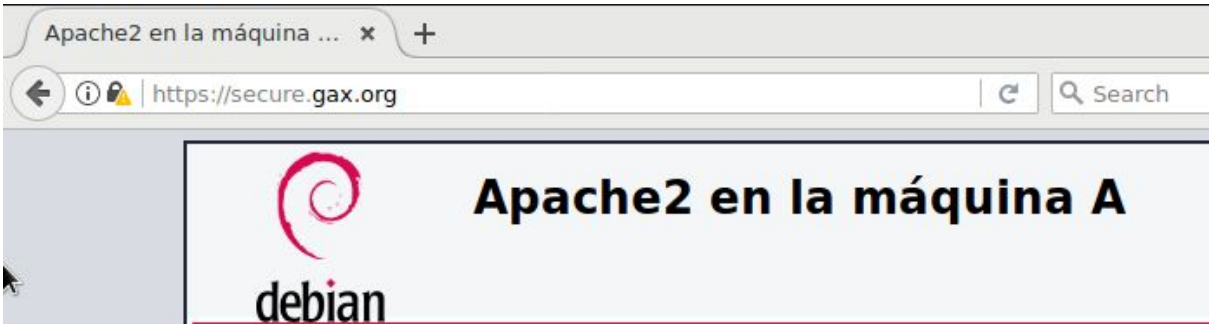
```
root@master-1-8:/var/www/html/secure# cd /etc/apache2/sites-available/
root@master-1-8:/etc/apache2/sites-available# ls
000-default.conf  default-ssl.conf  inside.conf  outside.conf
```

```
#   A self-signed (snakeoil) certificate can be created by installing
#   the ssl-cert package. See
#   /usr/share/doc/apache2/README.Debian.gz for more info.
#   If both key and certificate are stored in the same file, only the
#   SSLCertificateFile directive is needed.
SSLCertificateFile      /etc/ssl/private/secure.crt
#SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key
```

```
DocumentRoot /var/www/html/secure
```

Habilitamos el site con el comando *a2ensite default-ssl.conf* y recargamos Apache.

Entramos desde la máquina B a nuestra dirección o IP usada y obtenemos lo siguiente dándonos a entender que tenemos todo bien configurado.



DNAT y Proxy reverse

Para usar el DNAT, primero hemos de instalar Apache en la nuestra máquina B siguiendo los pasos anteriormente descritos en el primer apartado. Comprobamos que funcione cambiando el index.html que viene por defecto al instalar el Apache.

Después de esta comprobación, usamos el comando *iptables -t nat -A PREROUTING -i ens3 -p tcp --dport 80 -j DNAT --to 20.20.20.57:80* en la máquina A. Con esto haremos que todas las peticiones que lleguen a la máquina A sean atendidas por la máquina B.

```
root@master-1-8:/var/www/html# iptables -t nat -A PREROUTING -i ens3 -p tcp --dport 80 -j DNAT --to 20.20.20.57:80
root@master-1-8:/var/www/html#
```

Para comprobar que todo este funcione correctamente, nos conectamos a nuestra máquina D que está conectada a internet y nos conectamos a cualquier página del Apache de A. Como podemos ver, la página a la que nos dirige es el index.html que hemos modificado antes en B.

Para configurar y poder usar un reverse proxy, primero tenemos que instalar los módulos de Apache necesarios, es decir, los módulos proxy, proxy_html y proxy_http de la siguiente manera y reiniciamos el Apache.

```
root@master-1-8:/etc/apache2/mods-available# a2enmod proxy
Enabling module proxy.
To activate the new configuration, you need to run:
  systemctl restart apache2
root@master-1-8:/etc/apache2/mods-available# a2enmod proxy_html
Considering dependency proxy for proxy_html:
Module proxy already enabled
Considering dependency xml2enc for proxy_html:
Enabling module xml2enc.
Enabling module proxy_html.
To activate the new configuration, you need to run:
  systemctl restart apache2
root@master-1-8:/etc/apache2/mods-available# a2enmod proxy_http
Considering dependency proxy for proxy_http:
Module proxy already enabled
Enabling module proxy_http.
To activate the new configuration, you need to run:
  systemctl restart apache2
root@master-1-8:/etc/apache2/mods-available#
```

Creamos un fichero *reverse.conf* en el directorio /etc/apache2/sites-available y lo modificamos de la siguiente manera.

```
<VirtualHost 20.20.20.57:80>
        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html/

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

        Servername proxy.gax.org
        ProxyRequests Off
        ProxyPreserveHost On
        ProxyPass / http://20.20.20.57/
        ProxyPassReverse / http://client.gax.org

</VirtualHost>
```

Con esto haremos que todas las conexiones que entren por la dirección proxy.gax.org sean desviadas a la máquina B. Probamos desde la máquina D otra vez y observamos como nos da el resultado deseado.

Rendimiento

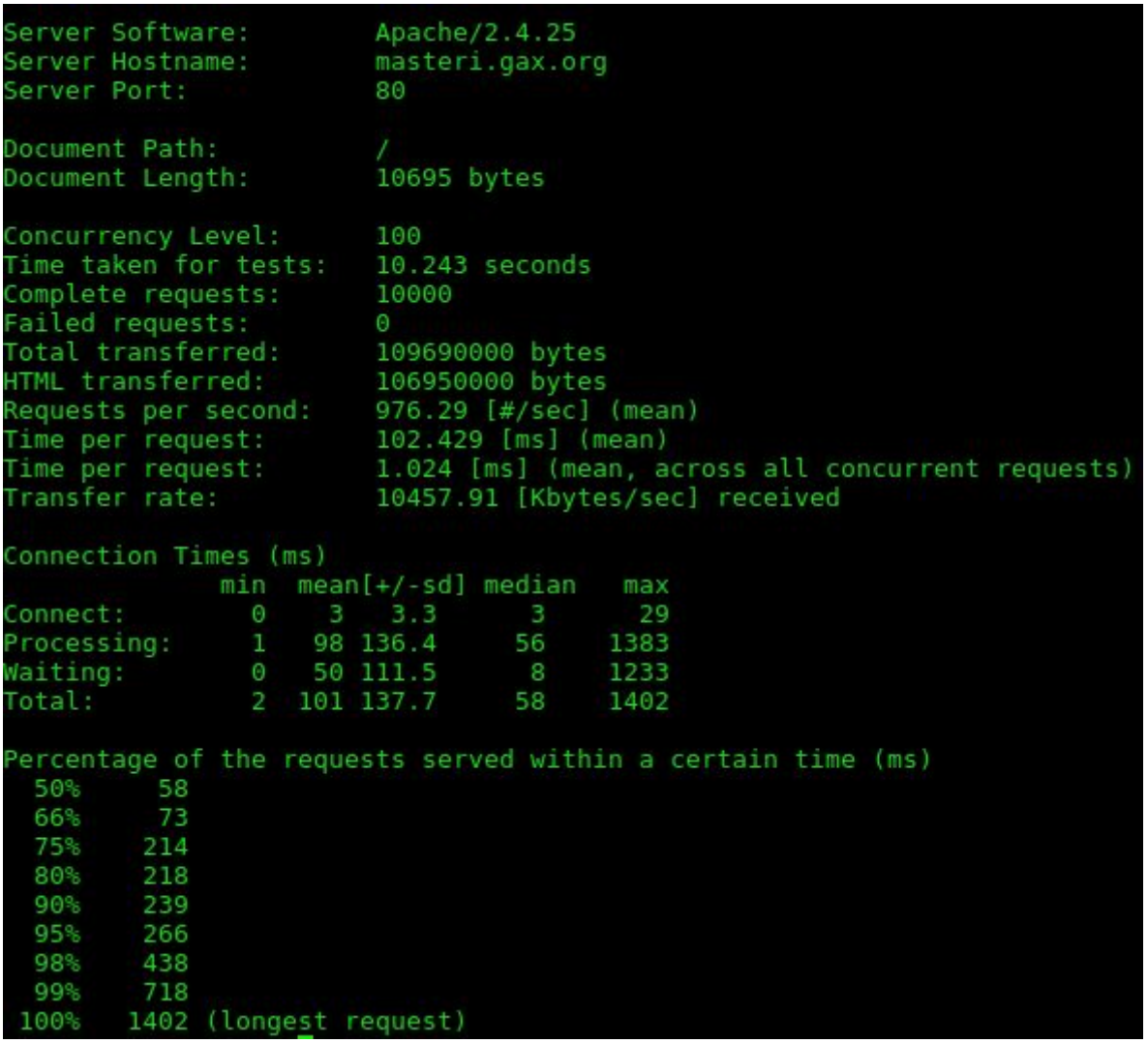
Para empezar, hemos de instalar el paquete apache2-utils en la máquina D que es de donde haremos las pruebas. Primero usamos el comando *apt update* y después usamos el comando *apt install apache2-utils*. Para comprobar que se ha instalado de forma correcta, usamos el comando *ab -V* que nos dará el siguiente resultado.

```
root@slave3-1-8:~# ab -V
This is ApacheBench, Version 2.3 <$Revision: 1757674 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/
```

Usamos el comando *ab -n 10000 -c 100* http://masteri.gax.org/ que enviara 10000 peticiones con una concurrencia de 100 peticiones cada vez a la dirección especificada que es nuestra dirección en el Apache de la máquina A. Obtendremos lo siguiente.

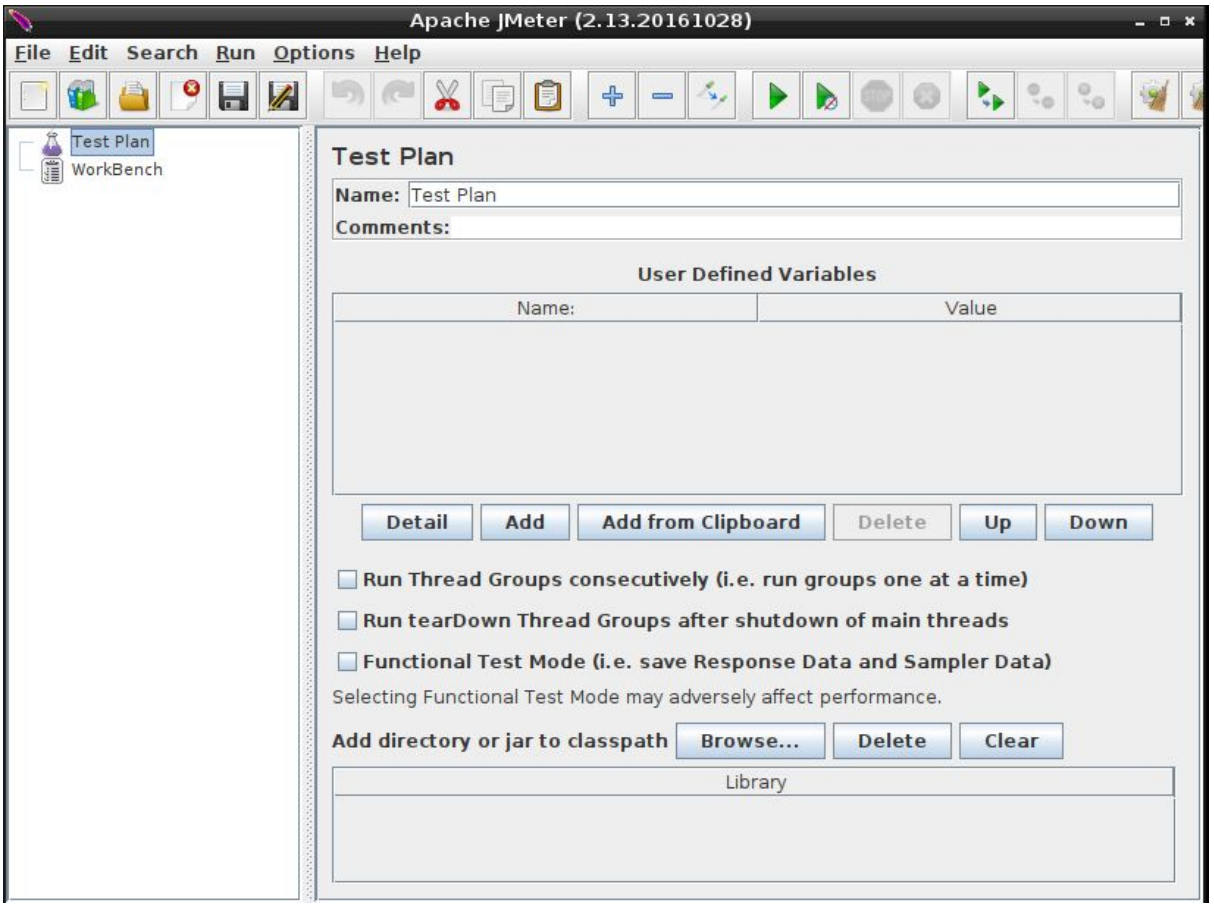
```
root@slave3-1-8:~# ab -n 10000 -c 100 http://masteri.gax.org/
This is ApacheBench, Version 2.3 <$Revision: 1757674 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking masteri.gax.org (be patient)
Completed 1000 requests
Completed 2000 requests
Completed 3000 requests
Completed 4000 requests
Completed 5000 requests
Completed 6000 requests
Completed 7000 requests
Completed 8000 requests
Completed 9000 requests
Completed 10000 requests
Finished 10000 requests
```

Y tras esperar unos segundos, este será el resultado.

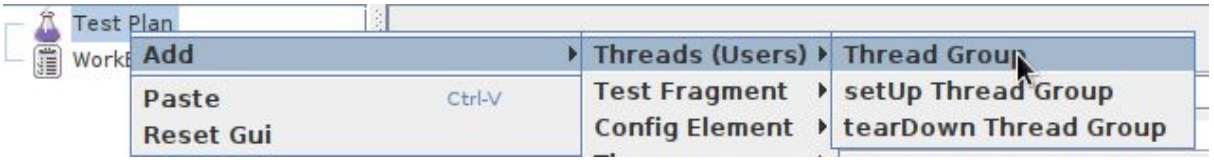


```
Server Software:        Apache/2.4.25
Server Hostname:        masteri.gax.org
Server Port:            80

Document Path:          /
Document Length:        10695 bytes

Concurrency Level:      100
Time taken for tests:   10.243 seconds
Complete requests:      10000
Failed requests:        0
Total transferred:      109690000 bytes
HTML transferred:       106950000 bytes
Requests per second:    976.29 [#/sec] (mean)
Time per request:       102.429 [ms] (mean)
Time per request:       1.024 [ms] (mean, across all concurrent requests)
Transfer rate:          10457.91 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    3   3.3      3      29
Processing:     1   98 136.4     56    1383
Waiting:        0   50 111.5      8    1233
Total:          2  101 137.7     58    1402

Percentage of the requests served within a certain time (ms)
  50%     58
  66%     73
  75%    214
  80%    218
  90%    239
  95%    266
  98%    438
  99%    718
 100%   1402 (longest request)
```

Para poder utilizar la herramienta Jmeter, tenemos que instalarla usando el comando *apt install jmeter*. Después de esto simplemente llamaremos a la aplicación y se nos abrirá la siguiente ventana.

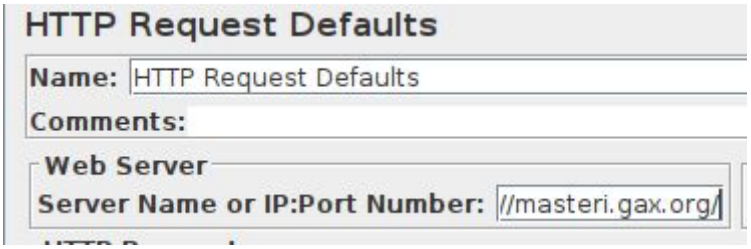


Añadimos un Thread Group en el que pondremos cuantos usuarios, cuantos threads y cada cuanto tiempo queremos que se intenten conectar a nuestra página.

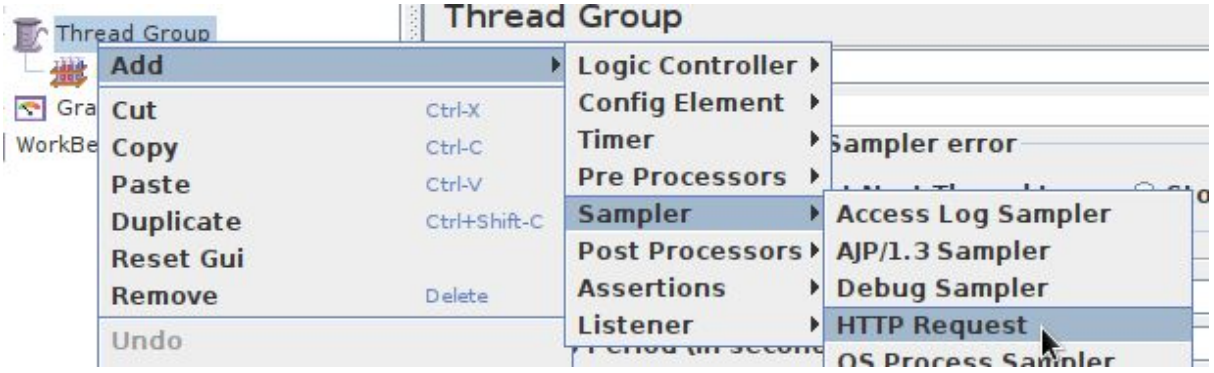


Contrario a la imagen, hemos puesto 100 threads cada 100 segundos, es decir un usuario cada segundo, y cada uno de estos intentara entrar 10 veces.

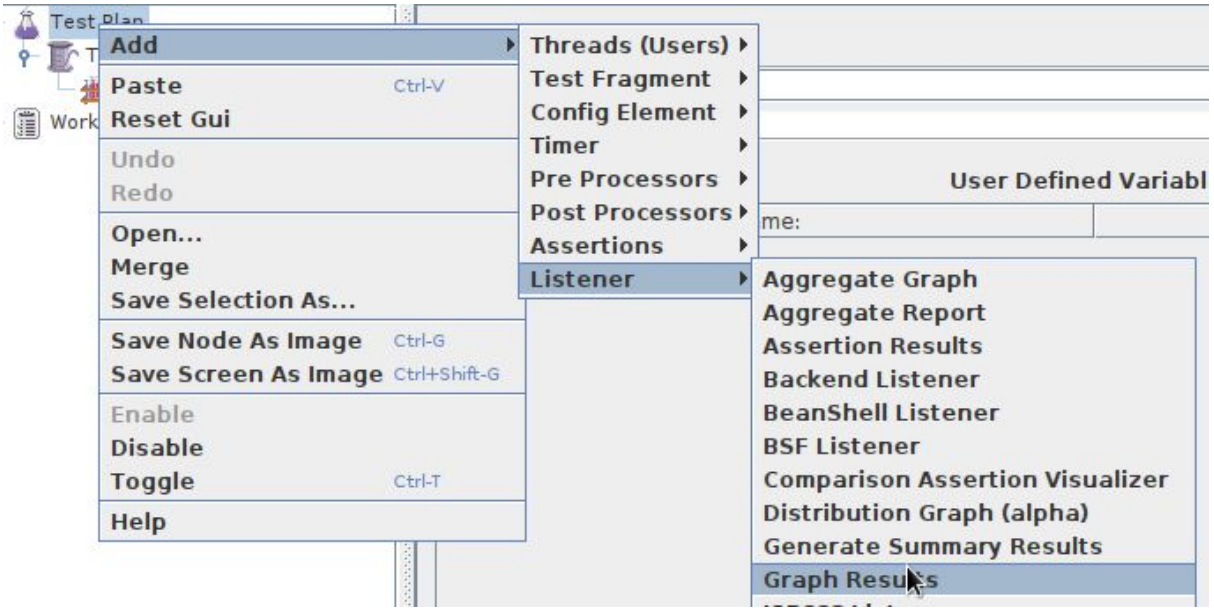
Después hemos de añadir el HTTP Request Defaults que indicará a la dirección que nos queremos conectar. Al igual que la anterior, en ves de usar la dirección en la imagen, nos hemos conectado a *master-1-8.gax.org*.



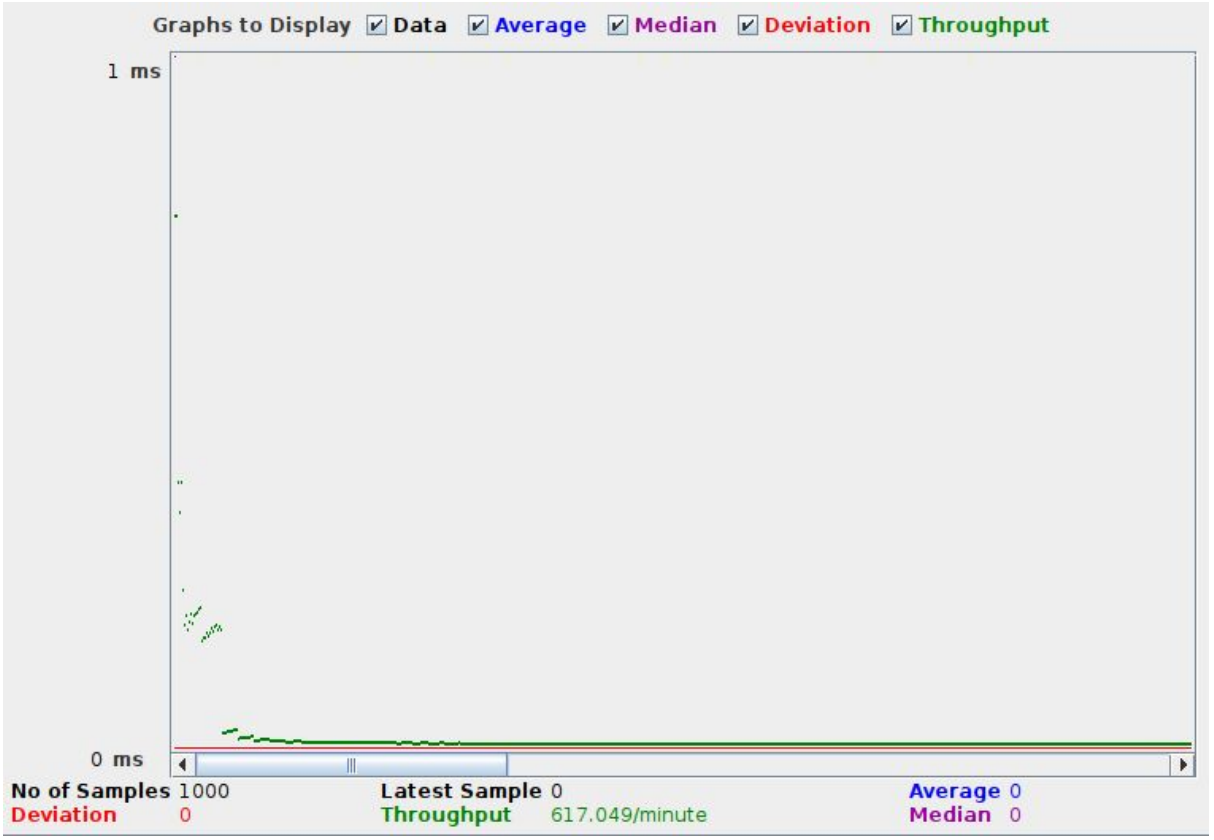
Agregamos un HTTP Request que dejamos en blanco que sera simplemente como el usuario intenta entrar a la página.



Y por último añadimos un Graph Results Listener en el que veremos los resultados.



Después de todo esto, ejecutamos la prueba y obtenemos el siguiente resultado.



Aún así, es bastante probable que la prueba no sea cien por cien del todo correcta y que incluso hayamos cometido errores, por eso preferimos usar los datos obtenidos por el Apache Utils que son más fiables.