

Práctica 3: Apache



Álvaro Santos Romero

Índice

1. Intalación de la máquina	2
1.1. Vagrantfile	2
1.2. Script de aprovisionamiento	2
1.3. Fichero hosts máquina host	2
2. Configuración básica de apache	3
2.1. Configuración del sitio	3
2.2. Generar fichero de configuración del sitio	3
2.3. Restringir acceso a usuarios	4
2.4. Vista de los directorios	5
2.5. Redirección	6
2.6. Grupos Laboratorios	7
2.7. Creación de una Web sencilla	8
3. Caché	9
4. Redirección	11
4.1. Cambio de extensión	11
4.2. Cambio de extensión (modificado)	11
4.3. Página personal usuarios	13
5. Balanceo de cargas	14
5.1. Pasos previos	14
5.2. Configuración del balanceador	15
5.3. Benchmarking	16
5.4. Conclusiones	18

1. Intalación de la máquina

Creación del entorno de trabajo para la realización de la práctica.

1.1. Vagrantfile

Vagrantfile utilizado para el correcto funcionamiento de la práctica.

```
Vagrant.configure("2") do |config|
  config.vm.box = "hashicorp/bionic64"
  config.vm.provision :shell, path: "bootstrap.sh"
  config.vm.network "private_network", ip: "192.168.100.10"
end
```

Código 1: Vagrantfile del entorno

1.2. Script de aprovisionamiento

Script de aprovisionamiento para la instalación de **apache** y **php**.

```
#!/bin/bash
apt-get update
apt-get install apache2 -y
apt-get install libapache2-mod-php -y
```

Código 2: Aprovisionamiento de la máquina

1.3. Fichero hosts máquina host

Antes de comenzar con la práctica, es conveniente añadir la IP de la máquina invitada a nuestro equipo host.

Fichero hosts en máquinas Windows

Este fichero se encuentra en *C:\Windows\System32\drivers\etc*.

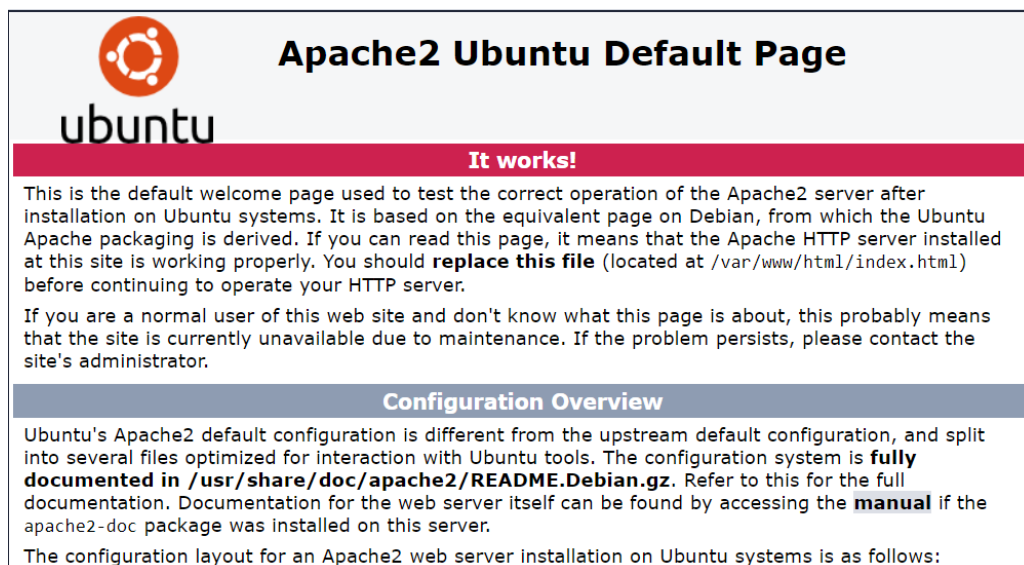


Figura 1: Vista desde equipo host.

2. Configuración básica de apache

Configuración básica de [apache](#).

2.1. Configuración del sitio

La página principal estará bajo el dominio **as.uca.es**.

2.2. Generar fichero de configuración del sitio

Por comodidad, podemos copiar el contenido de *000-default.conf* en nuestro fichero de configuración *as.uca.es.conf*.

```
<VirtualHost *:80>
    ServerName as.uca.es
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/as
    ErrorLog ${APACHE_LOG_DIR}/error.as.log
    CustomLog ${APACHE_LOG_DIR}/access.as.log combined
</VirtualHost>
```

Código 3: Creación de la configuración del sitio as.uca.es

Ahora, vamos a activar nuestro sitio:

```
vagrant@vagrant:/etc/apache2/sites-available$ sudo a2ensite as.uca.es.conf
Enabling site as.uca.es.
To activate the new configuration, you need to run:
    systemctl reload apache2
vagrant@vagrant:/etc/apache2/sites-available$
```

Figura 2: Activación de la configuración del sitio.

Hay que asegurarse de crear también una carpeta para albergar los ficheros de nuestro sitio en */var/www*.

```
chown: changing ownership of 'as': Operation not permitted
vagrant@vagrant:/var/www$ sudo chown www-data as
vagrant@vagrant:/var/www$ sudo chgrp www-data as
vagrant@vagrant:/var/www$ ls
as  html
vagrant@vagrant:/var/www$ ls -l
total 8
drwxr-xr-x 2 www-data www-data 4096 Apr 28 13:36 as
drwxr-xr-x 2 root      root      4096 Apr 27 18:46 html
vagrant@vagrant:/var/www$ sudo systemctl restart apache2
```

Figura 3: Directorio de sitio y configuración de permisos.

2.3. Restringir acceso a usuarios

Para restringir el acceso, utilizaremos el mod `auth_basic`.

Normalmente, al instalar apache, se nos autocarga el módulo, pero si no, lo cargaremos utilizando `a2enmod auth.basic.load`.

Para conseguir esta restricción, vamos a hacer lo siguiente en el fichero de configuración de nuestro sitio:

```
<LocationMatch "^/docs">
    AuthType basic
    AuthName "Autenticacion requerida"
    AuthBasicProvider file
    AuthUserFile /home/vagrant/AuthUserAsdb
    require user admin manolo
</LocationMatch>
```

Código 4: Configuración de autenticación para /docs

También debemos de generar el fichero de autenticación, de la siguiente manera:

```
root@vagrant:/home/vagrant# htpasswd -c AuthUserAsdb admin
New password:
Re-type new password:
Adding password for user admin
root@vagrant:/home/vagrant# htpasswd AuthUserAsdb manolo
New password:
Re-type new password:
Adding password for user manolo
root@vagrant:/home/vagrant# cat AuthUserAsdb
admin:$apr1$SWrerzkg$zTKtZKroqIUo2Kz2avQw31
manolo:$apr1$DBg2dqzX$0PX3NcC.6/B74HCCIThxY0
```

Figura 4: Creación de fichero de autenticación.

Si nos vamos a `as.uca.es/docs`, nos pedirá autenticación:

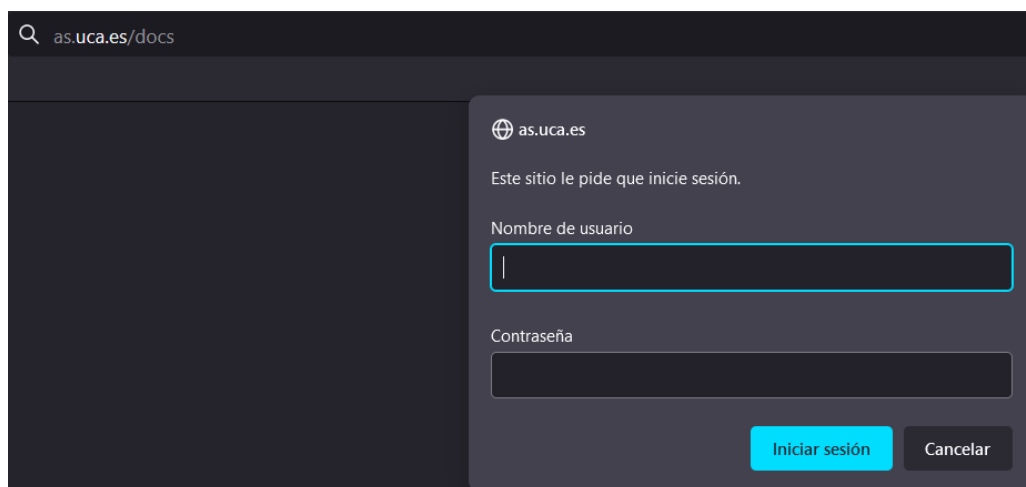


Figura 5: Vista de la página con la restricción de usuarios.



Figura 6: Contenido de la página.

2.4. Vista de los directorios

Para conseguir que no se muestren los directorios, tendremos que añadir en el fichero de configuración lo siguiente en cada ruta a que queremos aplicar esta configuración:

```
<Directory "/ruta">
  Options -Indexes
  ...
</Directory>
```

Código 5: Desactivar Indexes de un directorio

Ahora, para que sólo admin pueda acceder a **software**:

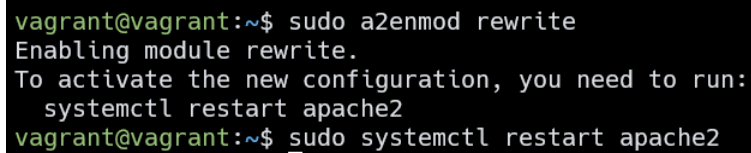
```
<LocationMatch "^/software">
  AuthType basic
  AuthName "Autenticacion requerida"
  AuthBasicProvider file
  AuthUserFile /home/vagrant/AuthUserAsdb
  require user admin
</LocationMatch>
```

Código 6: Restricción de acceso a /software

2.5. Redirección

En este apartado, vamos a hacer que la ruta `/sci2s` apunte a [SCI2S](https://sci2s.ugr.es/).

Para ello, vamos a activar el módulo de reescritura:



```
vagrant@vagrant:~$ sudo a2enmod rewrite
Enabling module rewrite.
To activate the new configuration, you need to run:
  systemctl restart apache2
vagrant@vagrant:~$ sudo systemctl restart apache2
```

Figura 7: Activación del mod rewrite.

Para habilitar esta redirección, escribiremos lo siguiente en el fichero de configuración de nuestro sitio:

```
<Directory "/var/www/as">
  RewriteEngine on
  RewriteRule "^sci2s$" "https://sci2s.ugr.es/" [R]
</Directory>
```

Código 7: Regla de reescritura para redireccionar a la página deseada

Ahora, al entrar a as.uca.es/sci2s, nos debería de redireccionar a la página mencionada anteriormente.

2.6. Grupos Laboratorios

Para hacer este ejercicio, vamos a crear dos nuevos ficheros de configuración para albergar las configuraciones.

```
<VirtualHost *:80>
    ServerName lab1.as.uca.es
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/lab1
    ErrorLog ${APACHE_LOG_DIR}/error.lab1.log
    CustomLog ${APACHE_LOG_DIR}/access.lab1.log combined
</VirtualHost>
```

Código 8: Configuración del sitio Lab1.as.uca.es

```
<VirtualHost *:80>
    ServerName lab2.as.uca.es
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/lab2
    ErrorLog ${APACHE_LOG_DIR}/error.lab2.log
    CustomLog ${APACHE_LOG_DIR}/access.lab2.log combined
</VirtualHost>
```

Código 9: Configuración del sitio Lab2.as.uca.es

Y ahora, si entramos en los dominios:

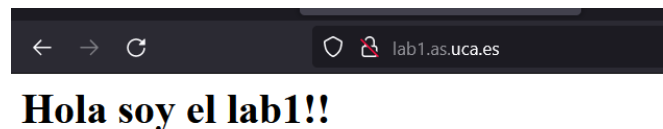


Figura 8: Vista del sitio web Lab1.as.uca.es

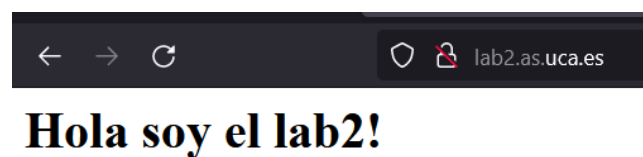


Figura 9: Vista del sitio web Lab2.as.uca.es

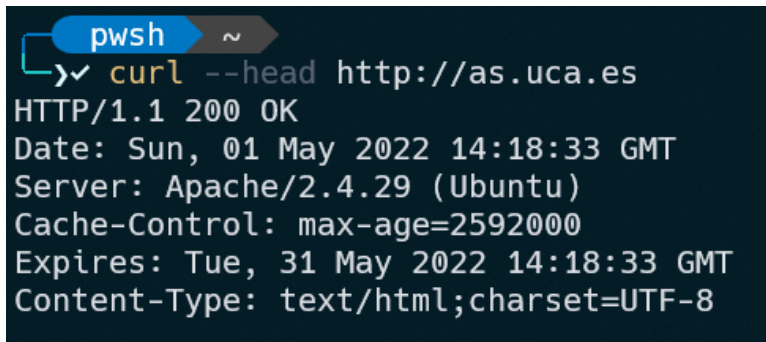
2.7. Creación de una Web sencilla

Para indicar a *apache* que queremos utilizar la codificación *utf8*, tendremos que editar el archivo */etc/apache2/conf-available/charset.conf* y descomentar la siguiente línea:

```
AddDefaultCharset utf-8
```

Código 10: Fichero de configuración *charset.conf*

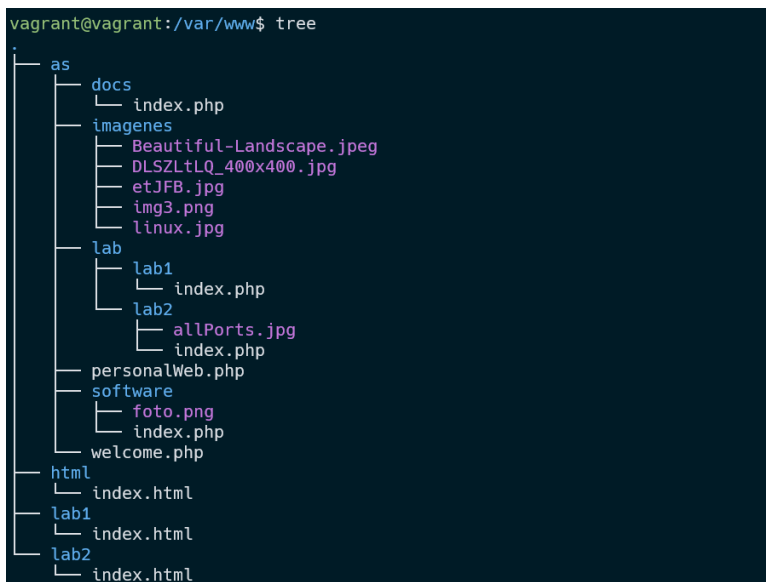
Al activar esta opción, ya podemos utilizar tildes y caracteres especiales:



```
pws@ ~  
$ curl --head http://as.uca.es  
HTTP/1.1 200 OK  
Date: Sun, 01 May 2022 14:18:33 GMT  
Server: Apache/2.4.29 (Ubuntu)  
Cache-Control: max-age=2592000  
Expires: Tue, 31 May 2022 14:18:33 GMT  
Content-Type: text/html; charset=UTF-8
```

Figura 10: Caracteres especiales.

Para la creación de una web sencilla, se nos pide que cada subcarpeta contenga al menos un fichero *"holamundo"*, por lo que, así quedaría la estructura de la web:



```
vagrant@vagrant:/var/www$ tree  
.  
├── as  
│   ├── docs  
│   │   ├── index.php  
│   │   └── imagenes  
│   │       ├── Beautiful-Landscape.jpeg  
│   │       ├── DLSZLtLQ_400x400.jpg  
│   │       ├── etJFB.jpg  
│   │       ├── img3.png  
│   │       └── linux.jpg  
│   ├── lab  
│   │   ├── lab1  
│   │   │   └── index.php  
│   │   └── lab2  
│   │       ├── allPorts.jpg  
│   │       └── index.php  
│   ├── personalWeb.php  
│   ├── software  
│   │   ├── foto.png  
│   │   ├── index.php  
│   │   └── welcome.php  
├── html  
│   └── index.html  
├── lab1  
│   └── index.html  
└── lab2  
    └── index.html
```

Figura 11: Estructura de la carpeta.

importante

Al crear los *index.html*, al entrar en la ruta raíz de cada sección, nos redirige a esta directamente. También se ha creado una regla de reescritura en la ruta *docs* para que cuando no encuentre un fichero o directorio a partir de este, lo redirija a *welcome.php*.

3. Caché

Para este ejercicio, utilizaremos el módulo **expires**. Este módulo se encarga de controlar la cabecera **Expires** y de controlar la directiva **max-age** de la cabecera **cache-control**.

No obstante, necesitamos habilitar estos módulos adicionalmente:

```
modules _____  
expires_mod  
headers_mod  
cache_mod
```

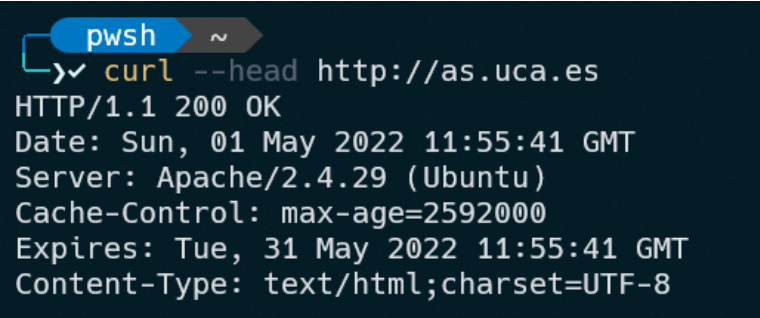
Una vez habilitados, vamos a crearnos un archivo en */etc/apache2/conf-available* llamado **expires.conf**, donde configuraremos esta funcionalidad:

```
<IfModule mod_expires.c>  
    # Enable expirations  
    ExpiresActive On  
  
    # Default directive  
    ExpiresDefault "access plus 1 month"  
  
    # Images  
    ExpiresByType image/png "access plus 2 month"  
    ExpiresByType image/jpg "access plus 2 month"  
  
</IfModule>
```

Código 11: Fichero expires.conf para activar las directivas de caché

Posteriormente, lo activamos con **a2enconf expires**.

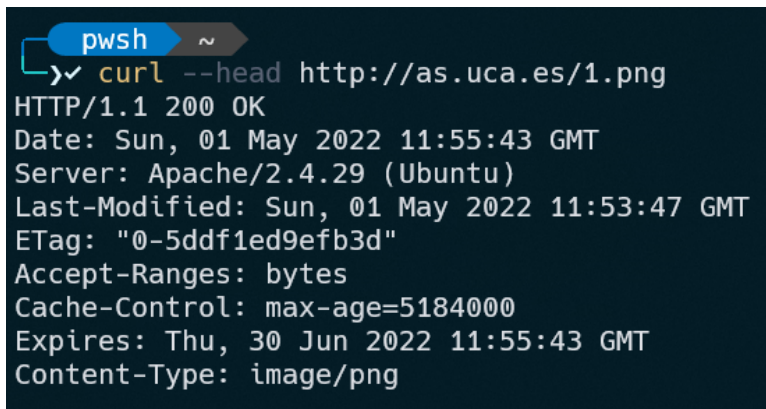
Ahora, utilizando la herramienta **CURL**, podremos verificar si el módulo está funcionando correctamente:



```
pwsh ~  
> curl --head http://as.uca.es  
HTTP/1.1 200 OK  
Date: Sun, 01 May 2022 11:55:41 GMT  
Server: Apache/2.4.29 (Ubuntu)  
Cache-Control: max-age=2592000  
Expires: Tue, 31 May 2022 11:55:41 GMT  
Content-Type: text/html; charset=UTF-8
```

Figura 12: caché por defecto.

No obstante, para comprobar si realmente se está aplicando el módulo a las imágenes, haremos un **curl** a una imagen:



```
pwwsh ~  
✓ curl --head http://as.uca.es/1.png  
HTTP/1.1 200 OK  
Date: Sun, 01 May 2022 11:55:43 GMT  
Server: Apache/2.4.29 (Ubuntu)  
Last-Modified: Sun, 01 May 2022 11:53:47 GMT  
ETag: "0-5ddf1ed9efb3d"  
Accept-Ranges: bytes  
Cache-Control: max-age=5184000  
Expires: Thu, 30 Jun 2022 11:55:43 GMT  
Content-Type: image/png
```

Figura 13: caché para imagen png.

4. Redirección

4.1. Cambio de extensión

Para alcanzar el objetivo de este ejercicio, tenemos que crear primero la ruta **imagenes** en **var/www/as**.

Crearemos una serie de archivos para comprobar el funcionamiento de la regla de reescritura una vez instalada.

```
vagrant@vagrant:/var/www/as/imagenes$ tree
.
├── img1.jpg
├── img2.jpg
├── img3.png
├── img4.jpg
└── index.php
```

Figura 14: Estructura de la carpeta **imagenes**.

Para activar esta regla, haremos lo siguiente:

```
<Directory "/var/www/as/imagenes">
  RewriteEngine on
  RewriteRule "^(.+)\.jpeg$" "/imagenes/$1.jpg"
</Directory>
```

Código 12: Regla para la búsqueda con cambio de extensión.

4.2. Cambio de extensión (modificado)

En este caso, sólo deberemos de cambiarle la extensión a los ficheros existentes con la extensión errónea.

```
<Directory "/var/www/as/imagenes">
  RewriteEngine on
  # Si no existe el fichero, entonces se realiza la reescritura
  RewriteCond %{REQUEST_FILENAME} !-f
  RewriteRule "^(.+)\.jpeg$" "/imagenes/$1.jpg" [R]
</Directory>
```

Código 13: Regla para la búsqueda con cambio de extensión modificada

Con esta regla, permitimos que si el fichero existe, no se cambie su extensión.

Funcionamiento de esta:

`url = as.uca.es/imagenes/linux.jpeg`.

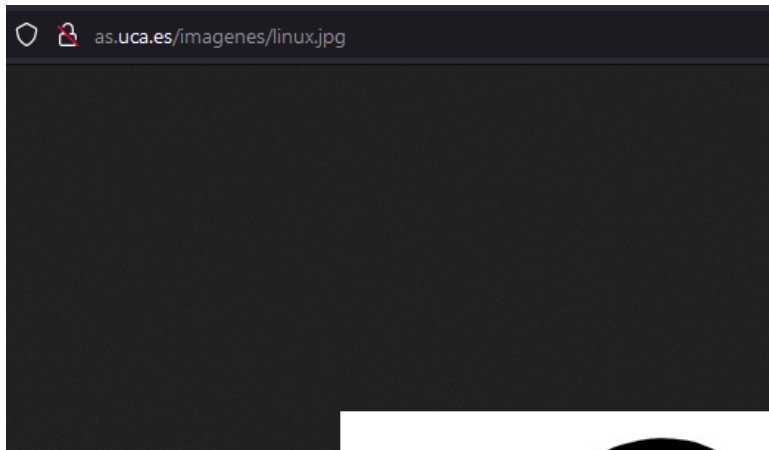


Figura 15: Cambio de extensión si existe archivo .jpg con mismo nombre.

Ahora, probaremos a introducir el nombre de un archivo jpeg existente:

`url = as.uca.es/imagenes/Beautiful-Landscape.jpeg`.

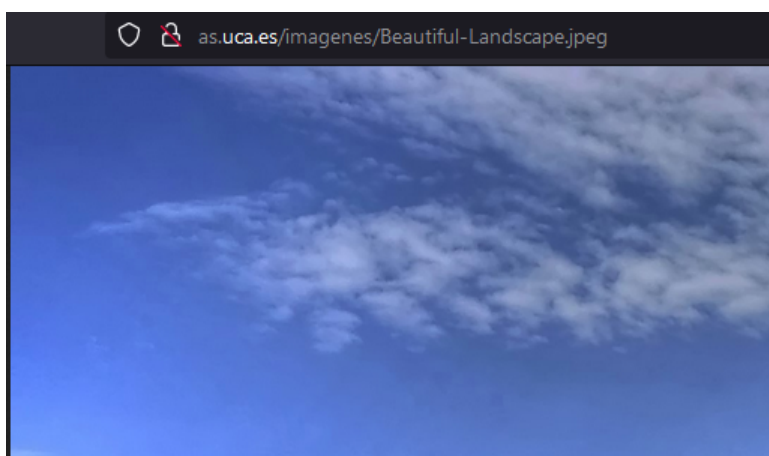


Figura 16: Imagen con extensión .jpeg.

Como se observa, no cambia la extensión del fichero puesto que la imagen existe en el directorio.

4.3. Página personal usuarios

En este caso, tenemos que transformar una URL *"fea"* en una URL *"friendly"*.

Para ello, vamos a utilizar de nuevo las reglas de reescritura:

```
<Directory "/var/www/as">
  Options +Indexes
  RewriteEngine on
  RewriteRule "^sci2s" "https://sci2s.ugr.es/" [R,L]
  RewriteRule "^team/(.*)$" "/personalWeb.php?user=$1" [NC,QSA,L]
</Directory>
```

Código 14: Regla de reescritura para transformar URLs

importante

Utilizar la flag **L (LAST)** para que en el momento en que se ejecute una regla, las demás no lo hagan.

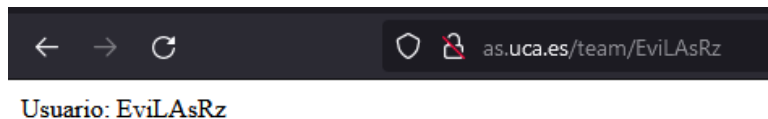


Figura 17: Salida de la ruta /team/EviLAsRz

```
<?php
$res = $_GET[user];
echo "Usuario: $res";

?>
```

Código 15: Código PHP para mostrar la "información" del usuario

5. Balanceo de cargas

5.1. Pasos previos

Para este ejercicio, vamos a crear un entorno multimáquina para albergar el balanceador y 3 nodos:

```
Vagrant.configure("2") do |config|
  config.vm.box = "hashicorp/bionic64"

  config.vm.define "router" do |master|
    master.vm.hostname = "master"
    config.vm.provision :shell, path: "provision.sh"
    master.vm.network "private_network", ip: "192.168.101.2"
  end

  config.vm.define "nodo1" do |nodo1|
    nodo1.vm.hostname = "nodo1"
    config.vm.provision :shell, path: "provision.sh"
    nodo1.vm.network "private_network", ip: "192.168.101.3"
  end

  config.vm.define "nodo2" do |nodo2|
    nodo2.vm.hostname = "nodo2"
    config.vm.provision :shell, path: "provision.sh"
    nodo2.vm.network "private_network", ip: "192.168.101.4"
  end

  config.vm.define "nodo3" do |nodo3|
    nodo3.vm.hostname = "nodo3"
    config.vm.provision :shell, path: "provision.sh"
    nodo3.vm.network "private_network", ip: "192.168.101.5"
  end
end
```

Código 16: Vagrantfile para el entorno del Balanceador de Cargas

En este entorno, utilizaremos el mismo aprovisionamiento para las máquinas.

```
#!/bin/bash
apt-get update
apt-get install apache2 -y
apt-get install libapache2-mod-php -y
```

Código 17: Aprovisionamiento de la máquina

5.2. Configuración del balanceador

Antes de nada, tenemos que activar los siguientes **módulos** para que funcione el proxy en nuestro contexto:

```
modules _____  
proxy_mod  
proxy_balancer_mod  
proxy_http_mod
```

Ahora, crearemos un pequeño servidor replicado en cada nodo para observar el comportamiento del balanceador.

```
<VirtualHost *:80>  
    ServerName nodo01.as.uca.es  
    ServerAdmin webmaster@localhost  
    DocumentRoot /var/www/as  
    ErrorLog ${APACHE_LOG_DIR}/error.as.log  
    CustomLog ${APACHE_LOG_DIR}/access.as.log combined  
</VirtualHost>
```

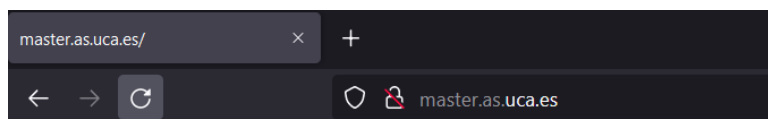
Código 18: Servidores réplica

Y, para poder configurar el balanceador, debemos de indicarlo de la siguiente manera:

```
<Proxy "balancer://balancerAS">  
  
    BalancerMember "http://nodo01.as.uca.es:80" loadfactor=6  
    BalancerMember "http://nodo02.as.uca.es:80" loadfactor=3  
    BalancerMember "http://nodo03.as.uca.es:80" status=+H  
  
    ProxySet lbmethod=byrequests  
  
</Proxy>  
  
ProxyPass "/" "balancer://balancerAS"  
ProxyPassReverse "/" "balancer://balancerAS"
```

Código 19: Configuración del proxy en el balanceador

Observamos que funciona correctamente:



Hola soy una respuesta desde nodo2!

Figura 18: Balanceo de la carga

5.3. Benchmarking

Para poder comprobar el número de peticiones por segundo de nuestro servidor, vamos a utilizar una herramienta proporcionada por apache. Esta herramienta se llama [AB](#).

Para ello, vamos a realizar un test para determinar cómo se comporta nuestro servidor de dos maneras:

- Servidor sin balanceo de cargas.

```
vagrant@vagrant:~$ ab -n 1000 -c 100 http://nodo01.as.uca.es/
This is ApacheBench, Version 2.3 <$Revision: 1807734 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking nodo01.as.uca.es (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests

Server Software:      Apache/2.4.29
Server Hostname:      nodo01.as.uca.es
Server Port:          80

Document Path:        /
Document Length:      48 bytes

Concurrency Level:    100
Time taken for tests:  1.441 seconds
Complete requests:    1000
Failed requests:       0
Total transferred:    294000 bytes
HTML transferred:     48000 bytes
Requests per second:  693.92 [#/sec] (mean)
```

Figura 19: Resultados del test sin balanceo de cargas.

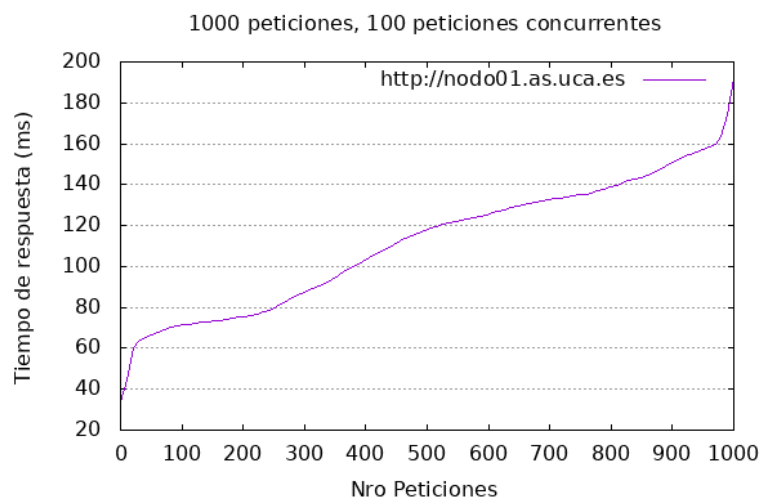


Figura 20: Resultados del test representados gráficamente.

- Servidor con balanceo de cargas.

```
vagrant@vagrant:~$ ab -n 1000 -c 100 http://master.as.uca.es/
This is ApacheBench, Version 2.3 <$Revision: 1807734 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking master.as.uca.es (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests

Server Software:      Apache/2.4.29
Server Hostname:      master.as.uca.es
Server Port:          80

Document Path:        /
Document Length:      45 bytes

Concurrency Level:    100
Time taken for tests:  1.306 seconds
Complete requests:    1000
Failed requests:       666
  (Connect: 0, Receive: 0, Length: 666, Exceptions: 0)
Total transferred:    292998 bytes
HTML transferred:     46998 bytes
Requests per second:  765.76 [#/sec] (mean)
```

Figura 21: Resultados del test con balanceo de cargas.

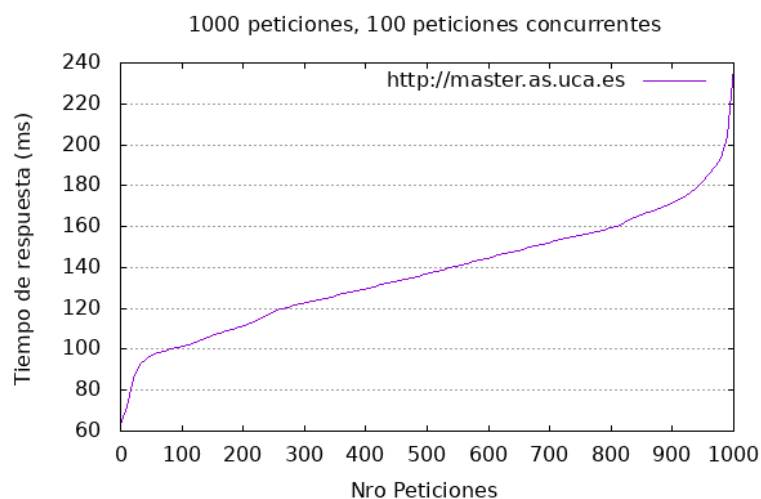


Figura 22: Resultados del test representados gráficamente.

5.4. Conclusiones

Como se ha podido observar en los gráficos, la mejora en eficiencia aumenta en un **6.5 %**, lo cual es moderadamente significativo, además de la ventaja de repartir la carga entre varios servidores, disminuyendo la carga de trabajo para cada servidor y así, aumentando la eficiencia en los puntos críticos (con mayor peticiones).