

Informe examen AS

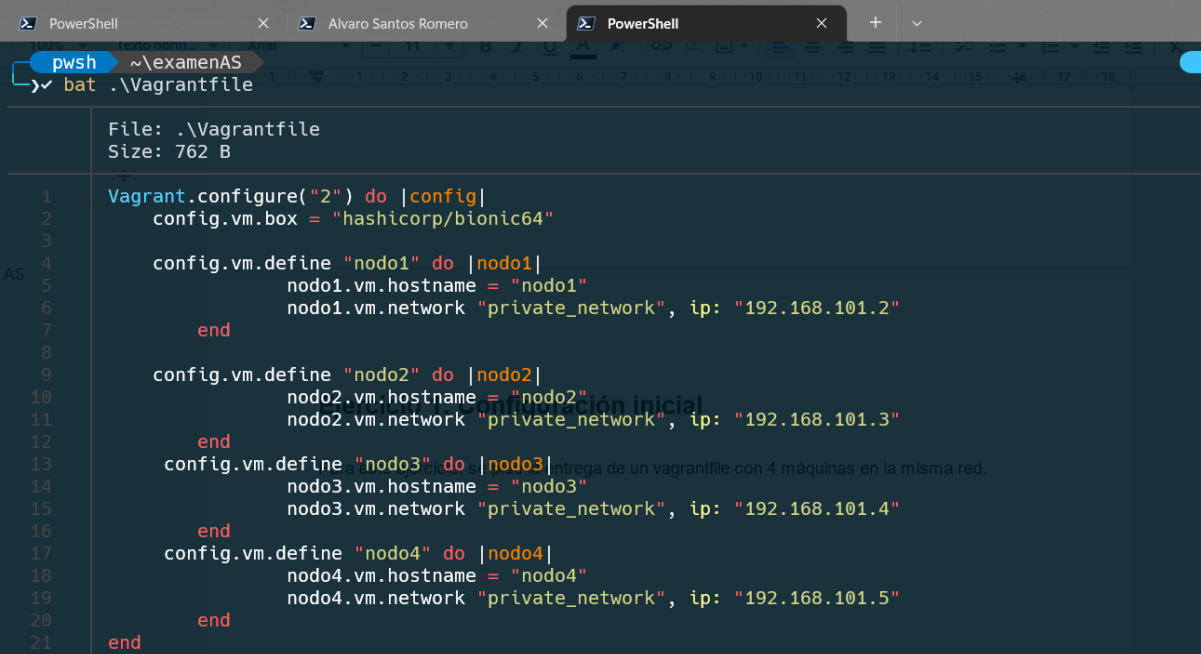
Administración de Servidores
Grado en Ingeniería Informática



Alvaro Santos Romero

Ejercicio 1: Configuración inicial

Para este ejercicio, se pide la entrega de un vagrantfile con 4 máquinas en la misma red.



```
File: .\Vagrantfile
Size: 762 B

1 Vagrant.configure("2") do |config|
2   config.vm.box = "hashicorp/bionic64"
3
4   config.vm.define "nodo1" do |nodo1|
5     nodo1.vm.hostname = "nodo1"
6     nodo1.vm.network "private_network", ip: "192.168.101.2"
7   end
8
9   config.vm.define "nodo2" do |nodo2|
10    nodo2.vm.hostname = "nodo2"
11    nodo2.vm.network "private_network", ip: "192.168.101.3"
12  end
13  config.vm.define "nodo3" do |nodo3|
14    nodo3.vm.hostname = "nodo3"
15    nodo3.vm.network "private_network", ip: "192.168.101.4"
16  end
17  config.vm.define "nodo4" do |nodo4|
18    nodo4.vm.hostname = "nodo4"
19    nodo4.vm.network "private_network", ip: "192.168.101.5"
20  end
21 end
```

Ejercicio 2: Cortafuegos

apartado 1: Restringir las peticiones de entrada aceptando puertos HTTP, HTTPS y SSH.

`sudo iptables -P INPUT DROP` ->rechazar por defecto los paquetes de entrada

`sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT` ->aceptar peticiones de entrada HTTP

`sudo iptables -A INPUT -p tcp --dport 443 -j ACCEPT` ->aceptar peticiones de entrada HTTPS

`sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT` ->aceptar peticiones de entrada SSH

```
PowerShell PowerShell Alvaro Santos Romero
vagrant@nodo1:~$ sudo iptables -L -n
Chain INPUT (policy DROP)
target    prot opt source                destination              tcp dpt:80
ACCEPT    tcp  --  0.0.0.0/0              0.0.0.0/0                tcp dpt:443
ACCEPT    tcp  --  0.0.0.0/0              0.0.0.0/0                tcp dpt:22
```

explicación:

-P : política por defecto de la cadena, es decir, si el paquete no es filtrado por ninguna regla de esa cadena, se le aplica la regla por defecto.

-A : append, añadir a la cadena (de arriba a abajo).

-p : protocolo al que dirigimos la regla (HTTP, HTTPS y SSH utilizan el protocolo tcp).

--dport: puerto destino de la regla.

apartado 2: permitir conexiones ya establecidas y relacionadas.

`sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED -p all -j ACCEPT`

`sudo iptables -A INPUT -m conntrack --ctstate RELATED -p all -j ACCEPT`

```
PowerShell PowerShell Alvaro Santos Romero
vagrant@nodo1:~$ sudo iptables -L -n
Chain INPUT (policy DROP)
target    prot opt source                destination              tcp dpt:80
ACCEPT    tcp  --  0.0.0.0/0              0.0.0.0/0                tcp dpt:443
ACCEPT    tcp  --  0.0.0.0/0              0.0.0.0/0                tcp dpt:22
ACCEPT    all  --  0.0.0.0/0              0.0.0.0/0                ctstate ESTABLISHED
ACCEPT    all  --  0.0.0.0/0              0.0.0.0/0                ctstate RELATED
```

explicación:

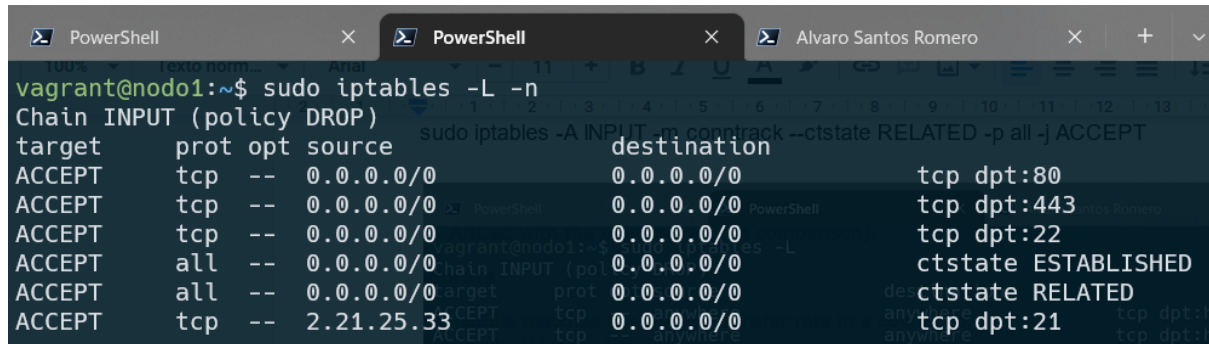
-m conntrack: módulo que nos permite controlar el estado de las conexiones.

-established: conexiones ya establecidas.

-related: conexiones relacionadas.

apartado 3: permitir conexión por FTP a un servidor externo ip 2.21.25.33

```
sudo iptables -A INPUT -p tcp --dport 21 -s 2.21.25.33 -j ACCEPT
```

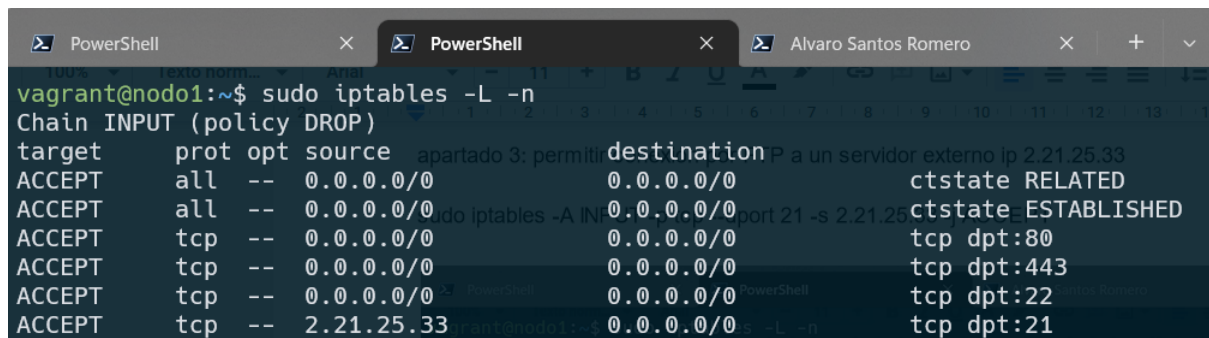


```
PowerShell PowerShell Alvaro Santos Romero
vagrant@nodo1:~$ sudo iptables -L -n
Chain INPUT (policy DROP)
target    prot opt source                destination           tcp dpt:80
ACCEPT    tcp  --  0.0.0.0/0              0.0.0.0/0             tcp dpt:443
ACCEPT    tcp  --  0.0.0.0/0              0.0.0.0/0             tcp dpt:22
ACCEPT    all  --  0.0.0.0/0              0.0.0.0/0             ctstate ESTABLISHED
ACCEPT    all  --  0.0.0.0/0              0.0.0.0/0             ctstate RELATED
ACCEPT    tcp  --  2.21.25.33             0.0.0.0/0             tcp dpt:21
```

explicación:

-s source: Origenario de la conexión.

IMPORTANTE: El orden correcto para que tengan sentido las reglas es el siguiente:



```
PowerShell PowerShell Alvaro Santos Romero
vagrant@nodo1:~$ sudo iptables -L -n
Chain INPUT (policy DROP)
target    prot opt source                destination           tcp dpt:80
ACCEPT    all  --  0.0.0.0/0              0.0.0.0/0             ctstate RELATED
ACCEPT    all  --  0.0.0.0/0              0.0.0.0/0             ctstate ESTABLISHED
ACCEPT    tcp  --  0.0.0.0/0              0.0.0.0/0             tcp dpt:80
ACCEPT    tcp  --  0.0.0.0/0              0.0.0.0/0             tcp dpt:443
ACCEPT    tcp  --  0.0.0.0/0              0.0.0.0/0             tcp dpt:22
ACCEPT    tcp  --  2.21.25.33             0.0.0.0/0             tcp dpt:21
```

Explicaciones:

- orden de las reglas:

Normalmente el orden de las reglas suelen ir de **más específicas a menos específicas**, es decir, más concretas a más genéricas.

Por ello, antes de nada, debemos de tratar las reglas establecidas y relacionadas, ya que por defecto rechazamos la entrada de paquetes a los puertos distintos de 80,43,22.

Si tuviéramos un proceso escuchando por el puerto 3306 (mysql por ejemplo), nos lo cargaremos si no tuviéramos una regla que aceptara conexiones ya establecidas/relacionadas como primera comprobación de la cadena.

Finalmente, añadimos las reglas de aceptación de paquetes por puertos específicos.

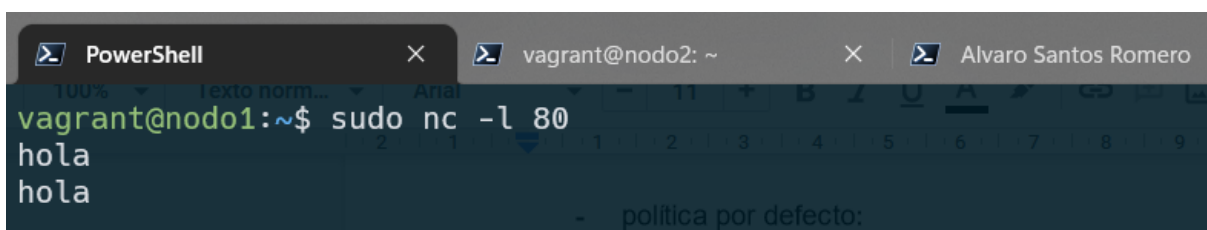
- política por defecto:

En este caso, la política por defecto es de **rechazo**, puesto que nos pide que las peticiones de entrada estarán restringidas.

- Comprobar el funcionamiento de las reglas:

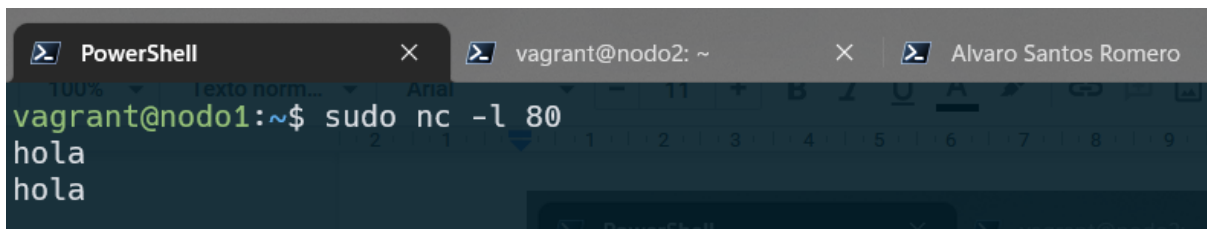
Para ello, vamos a usar NC ,TELNET, y NMAP.

Nos ponemos en escucha por el puerto deseado:



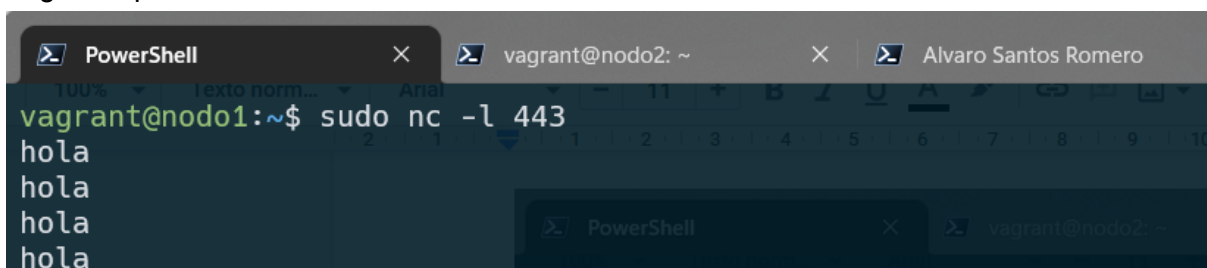
```
PowerShell x vagrant@nodo2: ~ x Alvaro Santos Romero
vagrant@nodo1:~$ sudo nc -l 80
hola
hola
```

Regla aceptar HTTP:



```
PowerShell x vagrant@nodo2: ~ x Alvaro Santos Romero
vagrant@nodo1:~$ sudo nc -l 80
hola
hola
```

Regla aceptar HTTPS:



```
PowerShell x vagrant@nodo2: ~ x Alvaro Santos Romero
vagrant@nodo1:~$ sudo nc -l 443
hola
hola
hola
hola
```

Regla aceptar SSH (conexión por ssh desde nodo2):

```
PowerShell x vagrant@nodo1: ~ x Alvaro Santos Romero x +
vagrant@nodo2:~$ ssh 192.168.101.2
The authenticity of host '192.168.101.2 (192.168.101.2)' can't be established.
ECDSA key fingerprint is SHA256:uY6GIjFdI9qTC4QYb980QRk+WblJF9cd5glr3SmmL+w.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.101.2' (ECDSA) to the list of known hosts.
vagrant@192.168.101.2's password:
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-58-generic x86_64)

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

System information as of Wed May 25 14:54:36 UTC 2022

System load:  0.0
Usage of /:   2.5% of 61.80GB
Memory usage: 12%
Swap usage:   0%

Processes: 89
Users logged in: 0
IP address for eth0: 10.0.2.15
IP address for eth1: 192.168.101.2

* 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

0 packages can be updated.
0 updates are security updates.

Last login: Wed May 25 14:10:07 2022 from 10.0.2.2
vagrant@nodo1:~$ |
```

-Reglas established y related:

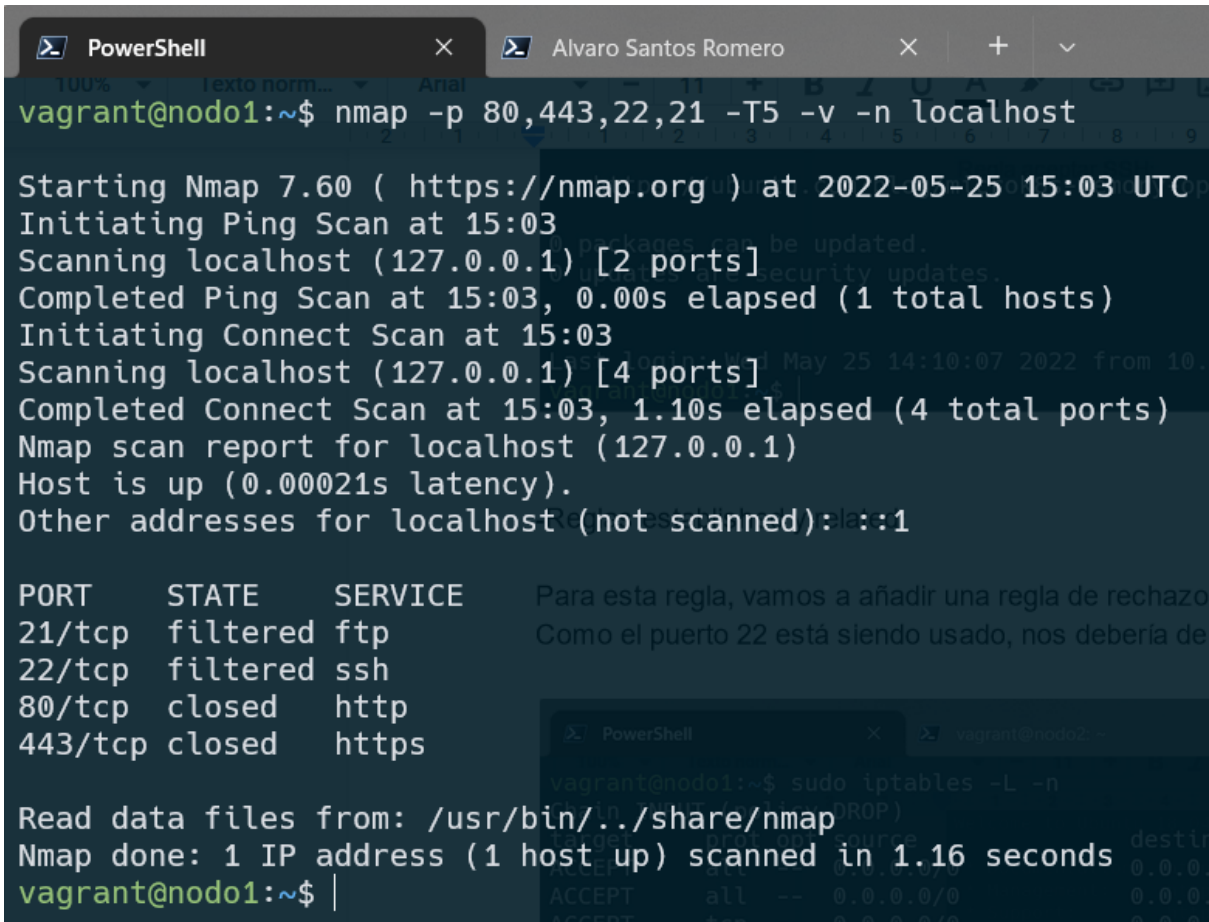
Para esta regla, vamos a añadir una regla de rechazo en el puerto 22.

Como el puerto 22 está siendo usado, nos debería de dejar usarlo igualmente.

```
PowerShell x vagrant@nodo2: ~ x Alvaro Santos Romero x +
vagrant@nodo1:~$ sudo iptables -L -n
Chain INPUT (policy DROP)
target    prot opt source                destination
ACCEPT    all  --  0.0.0.0/0              0.0.0.0/0             ctstate RELATED
ACCEPT    all  --  0.0.0.0/0              0.0.0.0/0             ctstate ESTABLISHED
ACCEPT    tcp  --  0.0.0.0/0              0.0.0.0/0             tcp dpt:80
ACCEPT    tcp  --  0.0.0.0/0              0.0.0.0/0             tcp dpt:443
ACCEPT    tcp  --  2.21.25.33             0.0.0.0/0             tcp dpt:21
DROP      tcp  --  0.0.0.0/0              0.0.0.0/0             tcp dpt:22
```

Vemos que aunque rechazamos la conexión por SSH, podemos seguir usando la máquina puesto que la conexión ya estaba establecida.

Comprobación con NMAP de los puertos implicados en las reglas:



```
vagrant@nodo1:~$ nmap -p 80,443,22,21 -T5 -v -n localhost

Starting Nmap 7.60 ( https://nmap.org ) at 2022-05-25 15:03 UTC
Initiating Ping Scan at 15:03
Scanning localhost (127.0.0.1) [2 ports]
Completed Ping Scan at 15:03, 0.00s elapsed (1 total hosts)
Initiating Connect Scan at 15:03
Scanning localhost (127.0.0.1) [4 ports]
Completed Connect Scan at 15:03, 1.10s elapsed (4 total ports)
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00021s latency).
Other addresses for localhost (not scanned): ::1

PORT      STATE      SERVICE
21/tcp    filtered  ftp
22/tcp    filtered  ssh
80/tcp    closed    http
443/tcp   closed    https

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 1.16 seconds
vagrant@nodo1:~$
```

Sólo aparecen **filtered** los puertos 22 y 21, puesto que NMAP no puede determinar su estado (el puerto 22 tiene rechazo de cualquier paquete de entrada y el puerto 21 sólo acepta paquetes del servidor 2.21.25.33).

Ejercicio 3: Apache

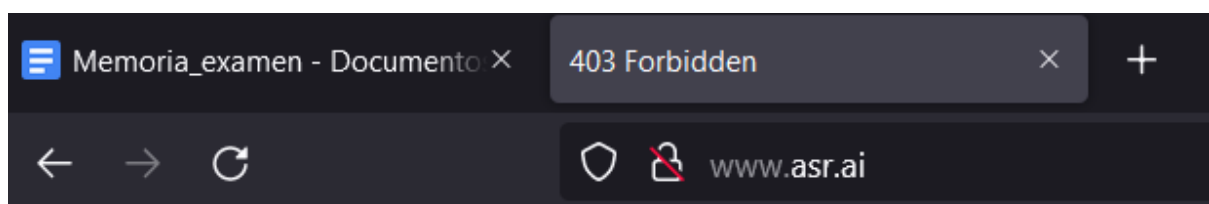
Configuración básica.

- Crearemos una carpeta (en mi caso asr) para albergar el sitio web en **/var/www**.
- Crearemos nuevas configuraciones de sitio en **/etc/apache2/sites-available**, con nombres (**www.asr.ai**), (**pruebas.asr.ai**) y (**admin.asr.ai**).

Apartado 1:

```
<Directory "/var/www/asr">
    Options -Indexes
</Directory>
<Directory "/var/www/asr/src">
    Options +Indexes
</Directory>
```

Desactivamos la opción de indexado para todas las carpetas excepto para la carpeta **src**.

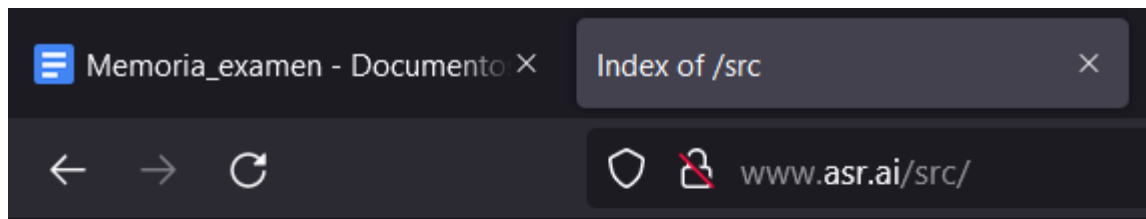


Forbidden



You don't have permission to access this resource.

Apache/2.4.29 (Ubuntu) Server at www.asr.ai Port 80

Vemos como el dominio principal tiene desactivado la indexación.



Index of /src

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 Parent Directory		-	
 codigomaligno.py	2022-05-25 15:26	0	

Apache/2.4.29 (Ubuntu) Server at www.asr.ai Port 80

Sin embargo, la carpeta src tiene el índice activado.

Apartado 2:

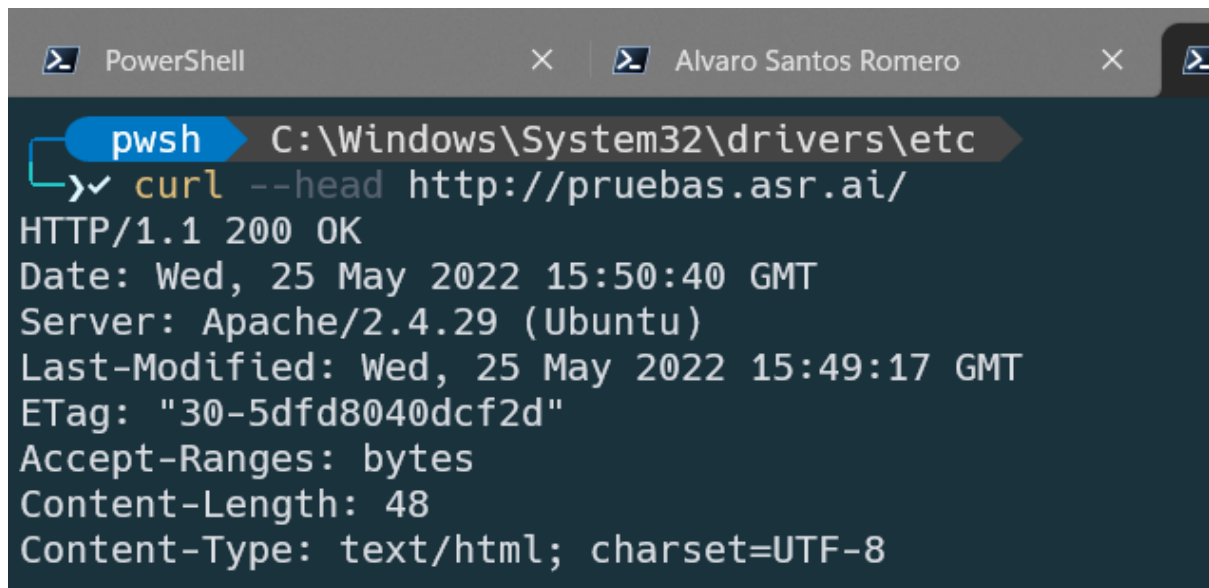
- Para ejecutar PHP, instalaremos el módulo que incorpora php a apache, llamado **libapache2-mod-php**.
- Para utilizar caracteres especiales, editaremos el fichero charset.conf en /etc/apache2/conf-available.

```
PowerShell | Alvaro Santos Romero
# Read the documentation before enabling AddDefaultCharset
# In general, it is only a good idea if you know that
# have this encoding. It will override any encoding g
# in meta http-equiv or xml encoding tags.

AddDefaultCharset UTF-8

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

De esta manera, ya estaría activado, ya que apache cargará las configuraciones activadas.

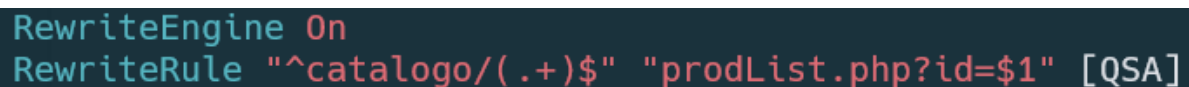


```
PowerShell C:\Windows\System32\drivers\etc
> curl --head http://pruebas.asr.ai/
HTTP/1.1 200 OK
Date: Wed, 25 May 2022 15:50:40 GMT
Server: Apache/2.4.29 (Ubuntu)
Last-Modified: Wed, 25 May 2022 15:49:17 GMT
ETag: "30-5dfd8040dcf2d"
Accept-Ranges: bytes
Content-Length: 48
Content-Type: text/html; charset=UTF-8
```

Apartado 3:

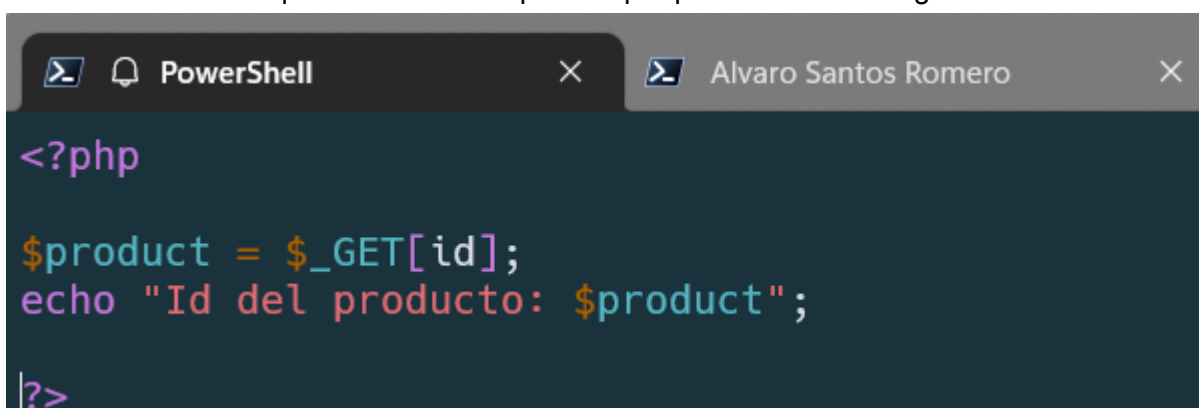
Antes de nada, hay que activar el módulo de reescritura.

- Conversión de url.



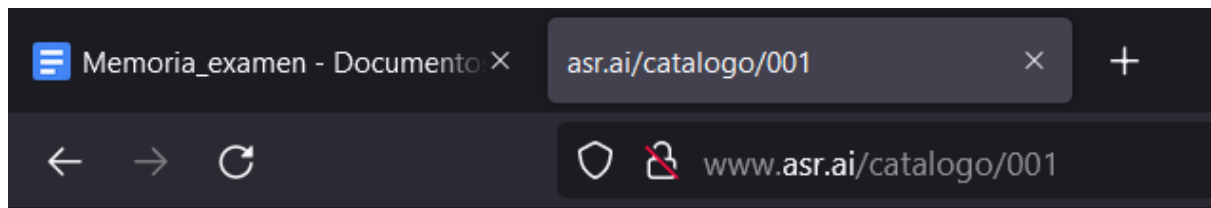
```
RewriteEngine On
RewriteRule "^catalogo/(.+) $" "prodList.php?id=$1" [QSA]
```

- Fichero PHP que recibe el valor pasado por parámetro en la regla.



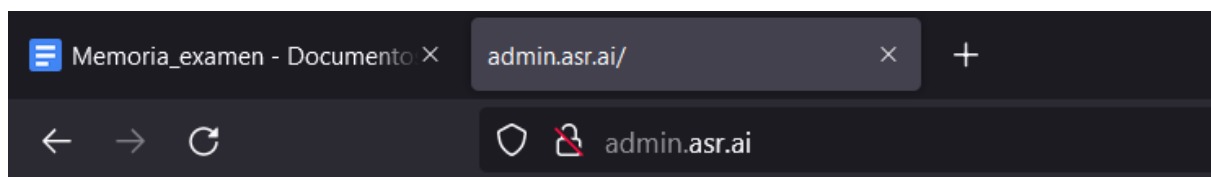
```
<?php
$product = $_GET[id];
echo "Id del producto: $product";
?>
```

- Salida de la URL.

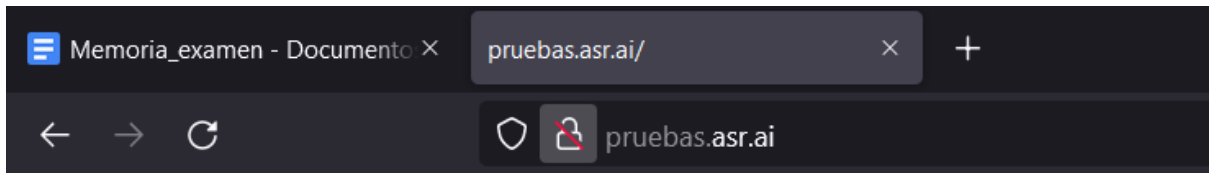


Id del producto: 001

Subdominios:



Página de administrador del sitio asr.ai



Página de pruebas del dominio asr.ai

Capturas de los ficheros conf de las zonas:

Zona www.asr.ai

```
PowerShell x Alvaro Santos Romero x PowerShell x
100% Texto normal Anal
<VirtualHost *:80>
  ServerName www.asr.ai
  ServerAdmin webmaster@localhost
  DocumentRoot /var/www/asr
  ErrorLog ${APACHE_LOG_DIR}/error.asr.log
  CustomLog ${APACHE_LOG_DIR}/access.asr.log combined

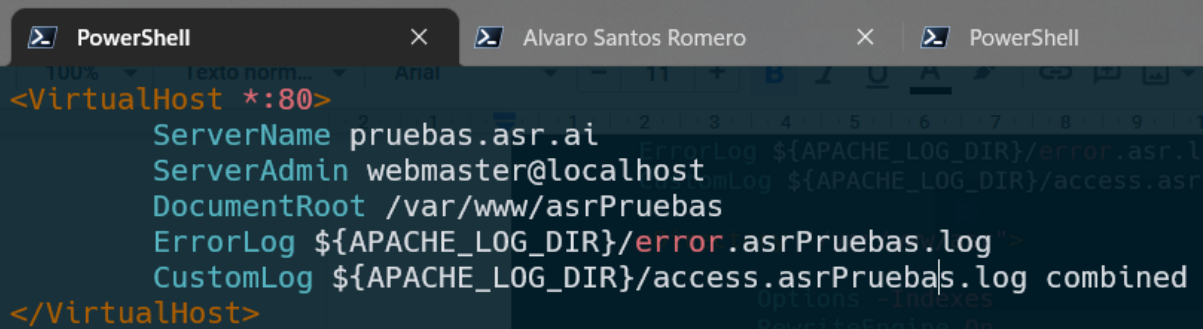
  <Directory "/var/www/asr">
    Options -Indexes
    RewriteEngine On
    RewriteRule "^catalogo/(.+) $" "prodList.php?id=$1" [QSA]
  </Directory>

  <Directory "/var/www/asr/src">
    Options +Indexes
  </Directory>
</VirtualHost>
```

Zona admin.asr.ai

```
PowerShell x Alvaro Santos Romero x PowerShell x
100% Texto normal Anal
<VirtualHost *:80>
  ServerName admin.asr.ai
  ServerAdmin webmaster@localhost
  DocumentRoot /var/www/asrAdmin
  ErrorLog ${APACHE_LOG_DIR}/error.asrAdmin.log
  CustomLog ${APACHE_LOG_DIR}/access.asrAdmin.log combined
</VirtualHost>
```

Zona pruebas.asr.ai



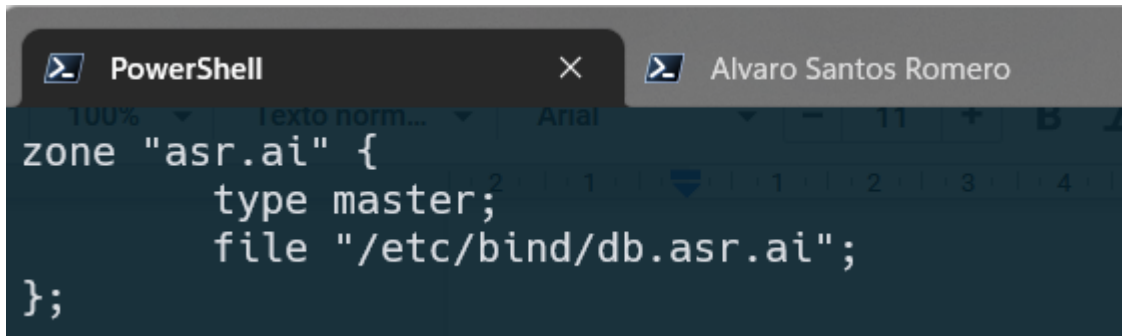
```
<VirtualHost *:80>
    ServerName pruebas.asr.ai
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/asrPruebas
    ErrorLog ${APACHE_LOG_DIR}/error.asrPruebas.log
    CustomLog ${APACHE_LOG_DIR}/access.asrPruebas.log combined
</VirtualHost>
```

Ejercicio 4: Configuración DNS

Para ello, instalamos Bind9.

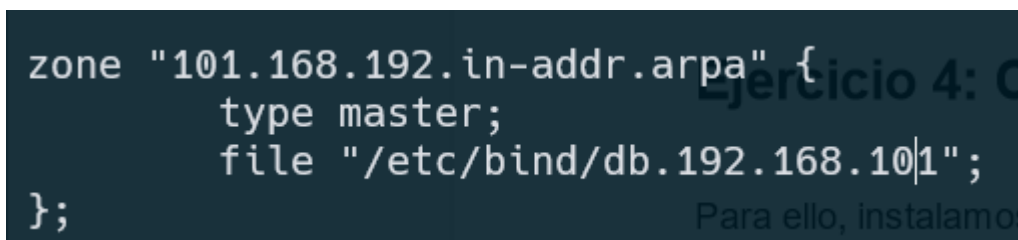
Apartado 1: Zonas directas.

- Crearemos en el fichero named.conf.local las zonas directas deseadas.



```
zone "asr.ai" {  
    type master;  
    file "/etc/bind/db.asr.ai";  
};
```

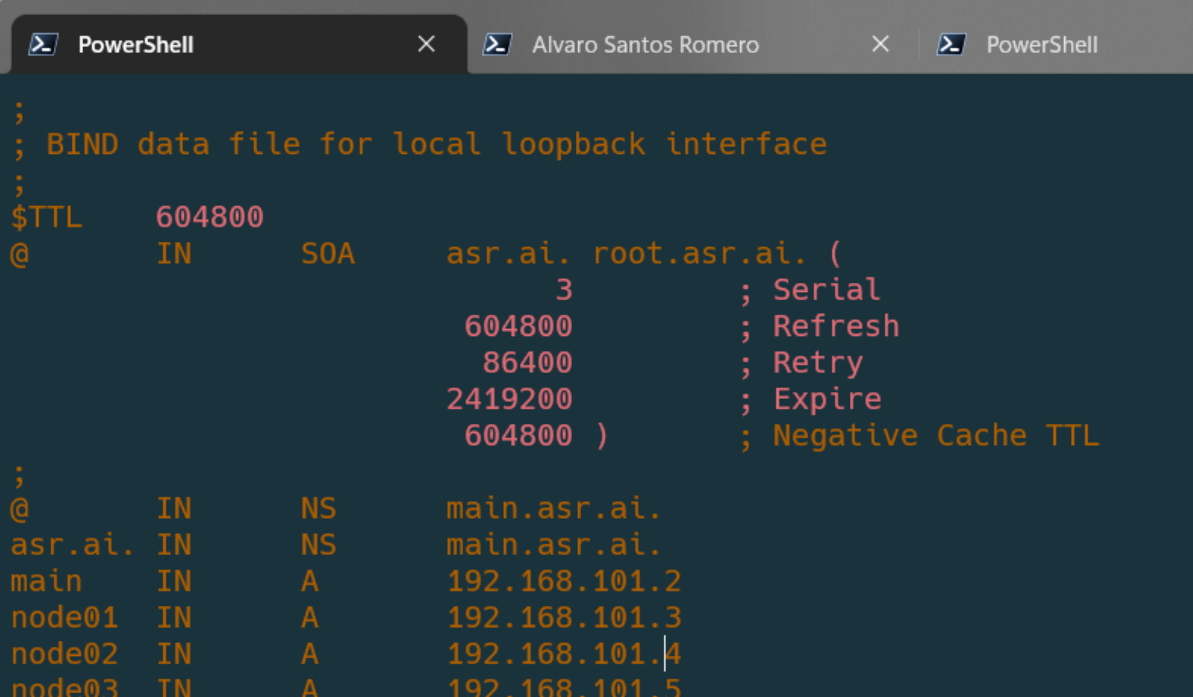
- Crearemos en el fichero named.conf.local las zonas inversas deseadas.



```
zone "101.168.192.in-addr.arpa" {  
    type master;  
    file "/etc/bind/db.192.168.101";  
};
```

Ahora, editaremos los ficheros de DB para cada zona.

- Zona directa:



```
;
; BIND data file for local loopback interface
;
$TTL      604800
@         IN      SOA      asr.ai. root.asr.ai. (
                        3      ; Serial
                        604800 ; Refresh
                        86400  ; Retry
                        2419200 ; Expire
                        604800 ) ; Negative Cache TTL
;
@         IN      NS       main.asr.ai.
asr.ai.   IN      NS       main.asr.ai.
main      IN      A        192.168.101.2
node01    IN      A        192.168.101.3
node02    IN      A        192.168.101.4
node03    IN      A        192.168.101.5
```

Este fichero recoge los dominios y las ips correspondientes a cada uno, de tal manera que cuando nos pregunten por una ip de una zona determinada, si la tenemos contemplada en el fichero named.conf.local, este mirará los ficheros (db) asociados a la zona y buscará la IP del dominio.

- Zona inversa:

```
PowerShell
Alvaro Santos Romero
PowerShell

;
; BIND reverse data file for local loopback interface
;
$TTL 604800
@ IN SOA main.asr.ai. root.asr.ai. (
    5 ; Serial
    604800 ; Refresh
    86400 ; Retry
    2419200 ; Expire
    604800 ) ; Negative Cache TTL
;
@ IN NS main.
2 IN PTR main.asr.ai.
3 IN PTR node01.asr.ai.
4 IN PTR node02.asr.ai.
5 IN PTR node03.asr.ai.
```

Para comprobar que están cargadas correctamente, utilizaremos el siguiente comando:

```
PowerShell
Alvaro Santos Romero
PowerShell

vagrant@nodo1:/etc/bind$ sudo named-checkconf -z named.conf.local
zone asr.ai/IN: loaded serial 3
zone 101.168.192.in-addr.arpa/IN: loaded serial 24800
vagrant@nodo1:/etc/bind$
```

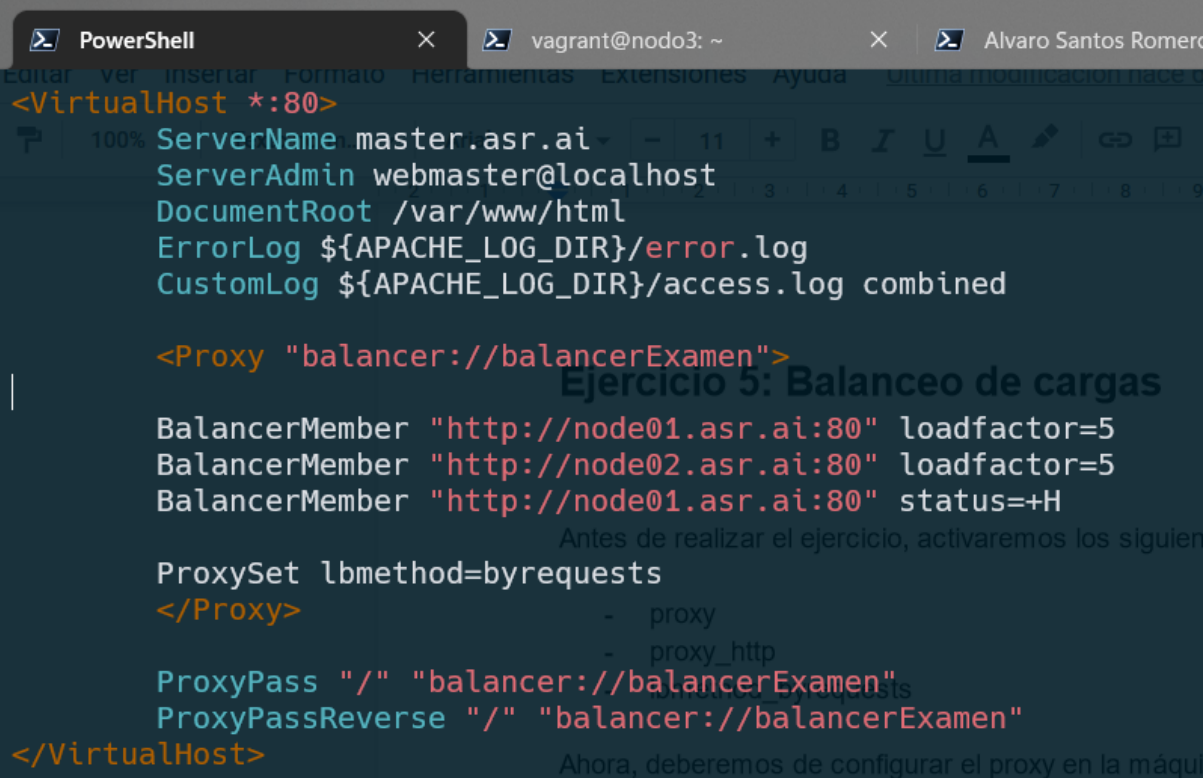

Ejercicio 5: Balanceo de cargas

Antes de realizar el ejercicio, activaremos los siguientes mods:

- proxy
- proxy_http
- lbmethod_byrequests

Ahora, deberemos de configurar el proxy en la máquina donde tenemos instalado el servidor DNS.

La configuración del proxy la haremos en el fichero 000-default.conf, ya que no es necesario crear un sitio nuevo.

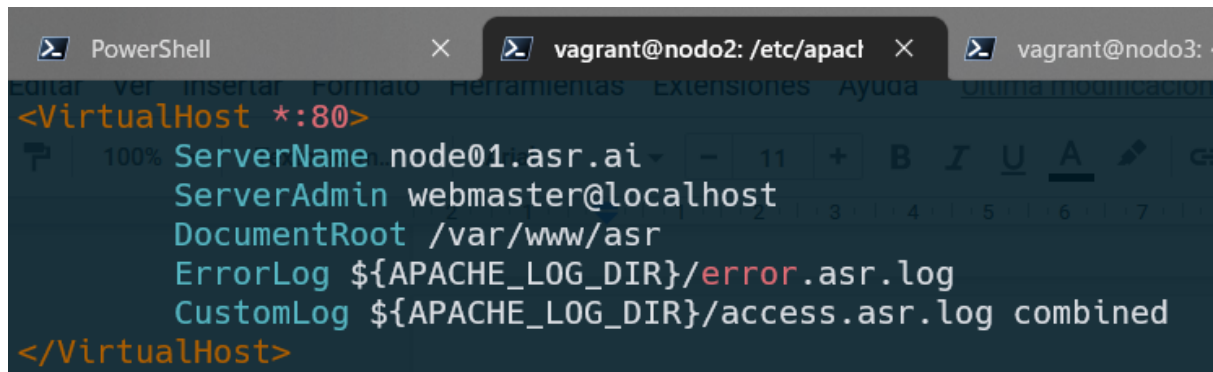


```
<VirtualHost *:80>
    ServerName master.asr.ai
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/html
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined

    <Proxy "balancer://balancerExamen">
        BalancerMember "http://node01.asr.ai:80" loadfactor=5
        BalancerMember "http://node02.asr.ai:80" loadfactor=5
        BalancerMember "http://node01.asr.ai:80" status=+H
        ProxySet lbmethod=byrequests
    </Proxy>
    ProxyPass "/" "balancer://balancerExamen"
    ProxyPassReverse "/" "balancer://balancerExamen"
</VirtualHost>
```

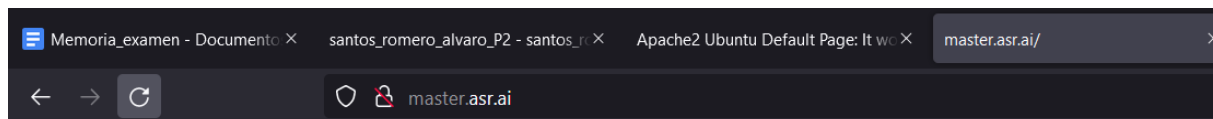
- El proxy tiene como miembros de este los nodos 1,2,3.
- Los nodos 1 y 2, responden la mitad de las peticiones cada uno.
- El nodo 3 es un nodo de respaldo por si los dos nodos anteriores fallan.
- El método de carga del proxy es por peticiones.
- El proxyPass y proxyPassReverse nos sirven para obtener y enviar la información a los nodos.

Y ahora, duplicaremos un servidor en estos 3 nodos:



```
<VirtualHost *:80>
  ServerName node01.asr.ai
  ServerAdmin webmaster@localhost
  DocumentRoot /var/www/asr
  ErrorLog ${APACHE_LOG_DIR}/error.asr.log
  CustomLog ${APACHE_LOG_DIR}/access.asr.log combined
</VirtualHost>
```

Resultado de entrar al servidor master.asr.ai:



Respuesta desde el nodo 2

Si vamos recargando, responden el nodo 1 y el nodo2.

Sin embargo, el nodo3 sólo responderá si el nodo1 y nodo2 están caídos (o inaccesibles).

Para comprobar que está usando el DNS, adjunto capturas de pantalla del fichero hosts y del fichero resolv.conf:

```
PowerShell PowerShell Alvaro San
editar ver insertar Formato Herramientas Extensiones Ayuda Última modificac
127.0.0.1 localhost
127.0.1.1 vagrant.vagrant.vagrant + B I U A
# The following lines are desirable for IPv6 capable hosts
::1 localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
127.0.2.1 nodo1 nodo1
```

```
nameserver 192.168.101.2
nameserver 127.0.0.53
options edns0
search uca.es
```