

# Informe Examen Técnico



Álvaro Santos Romero

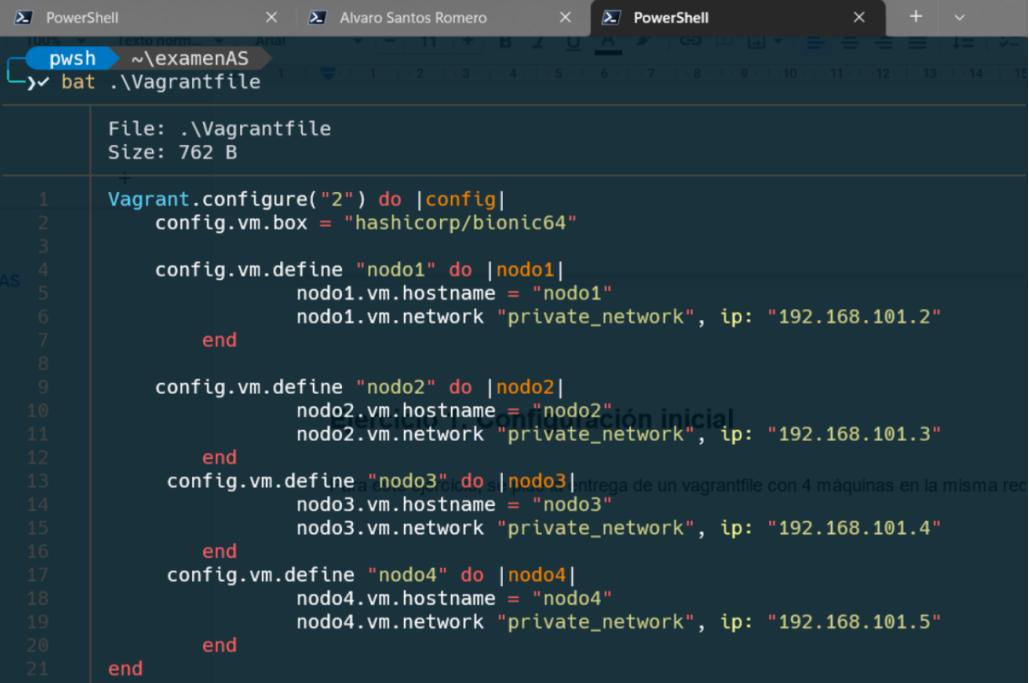
Mayo del 2022

## Índice

<b>1. Exercise 1: Setup</b>	<b>2</b>
<b>2. Exercise 2: Firewall</b>	<b>3</b>
<b>3. Exercise 3: Apache</b>	<b>7</b>
3.1. Indexes . . . . .	7
3.2. UTF-8 . . . . .	8
3.3. Apache Engine . . . . .	9
<b>4. Exercise 4: DNS config</b>	<b>12</b>
4.1. Direct Zones . . . . .	12
4.2. Inverse Zones . . . . .	13
<b>5. Load Balancer</b>	<b>14</b>
5.1. Load Balancer with DNS . . . . .	16

## 1. Exercise 1: Setup

Vagrantfile setting up four machines in the same net.



The screenshot shows a terminal window with two tabs open. The active tab is titled 'PowerShell' and contains the command 'pwsh ~\examenAS >> bat .\Vagrantfile'. Below this, the output of the Vagrantfile is displayed. The Vagrantfile defines four machines ('nodo1', 'nodo2', 'nodo3', 'nodo4') in a private network with specific IP addresses. The code is color-coded for syntax highlighting.

```
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
```

Figura 1: Vagrantfile

In this exercise, we are defining four machines in the same net, as we want them to be visible from each others.

## 2. Exercise 2: Firewall

We will use **iptables** to configure a simple firewall.

### 1. Restrict input requests from HTTP, HTTPS and SSH

- `iptables -P input DROP` — default deny input requests rule
- `iptables -A INPUT tcp -dport 80 -j ACCEPT` — accept input requests from port 80
- `iptables -A INPUT tcp -dport 443 -j ACCEPT` — accept input requests from port 443
- `iptables -A INPUT tcp -dport 22 -j ACCEPT` — accept input requests from port 22

```
vagrant@nodo1:~$ sudo iptables -L -n
Chain INPUT (policy DROP)
target     prot opt source               destination
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
```

Figura 2: iptables

### 2. Allow established and related connections

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

In this case, we are using a module called **conntrack** which is used to manage connections.

### 3. Allow FTP connection to an external server (2.21.25.33)

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

```
ACCEPT      tcp  --  2.21.25.33          0.0.0.0/0           tcp dpt:21
```

Figura 3: ftp

### !Caution!

Iptables rules are followed in order, so the rules must be set in this order to work properly.

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

Figura 4: iptables rules

### Rules order

Usually, the rules are order from most to less specific.

As we discussed earlier, iptables follows a secuencial order to execute the rules.

First of all, we should set the **established** and **related** ones first, as we dont want them to be overrided by the **restricted** ones.

The last rules should always be the most specific, such as acepting requests from a particular port or similar.

## 4. Testing iptables rules

We will use [NetCat](#), [Telnet](#) and [Nmap](#).

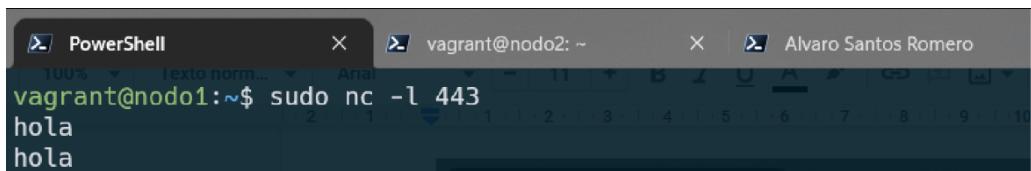
- Listening port 80



```
vagrant@nodo1:~$ sudo nc -l 80
hola
hola
```

Figura 5: HTTP connection

- Listening port 443



```
vagrant@nodo1:~$ sudo nc -l 443
hola
hola
```

Figura 6: HTTPS connection

- SSH connections

The screenshot shows a terminal window with three tabs: "PowerShell", "vagrant@nodo1: ~", and "Alvaro Santos Romero". The "vagrant@nodo1: ~" tab is active and displays the following output:

```
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          Processes:      89
 Usage of /:   2.5% of 61.80GB  Users logged in:  0
 Memory usage: 12%
 Swap usage:   0%
 * 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:~$ |
```

Figura 7: SSH connection

In this case, even if we decided to create a new rule denying connections from SSH, we will not be affected as we have a ESTABLISHED rule affecting this protocol.

- Nmap scan

The screenshot shows a terminal window titled "PowerShell" with the command "nmap -p 80,443,22,21 -T5 -v -n localhost" running. The output details a ping scan, a connect scan, and a service scan for ports 80, 443, 22, and 21. It notes that port 21 is filtered and port 22 is closed. Below the main output, there is a smaller window showing the result of the command "sudo iptables -L -n", which lists several rules, including one for port 21.

```
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          Description
21/tcp    filtered  ftp              Para esta regla, vamos a añadir una regla de rechazo
22/tcp    filtered  ssh              Como el puerto 22 está siendo usado, nos debería de
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
```

Figura 8: Nmap scan

Ports **22** and **21** are tagged as **filtered** as Nmap can not determine its states. (port 21 denies all input requests and port 22 only accepts from 2.21.25.33).

### 3. Exercise 3: Apache

#### Setup

- New folder to contain the website at **var/www**
- Create site configurations for in **/etc/apache2/sites-available**

#### 3.1. Indexes

We disable **indexes** option for all folder except for src.

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

Figura 9: Site config

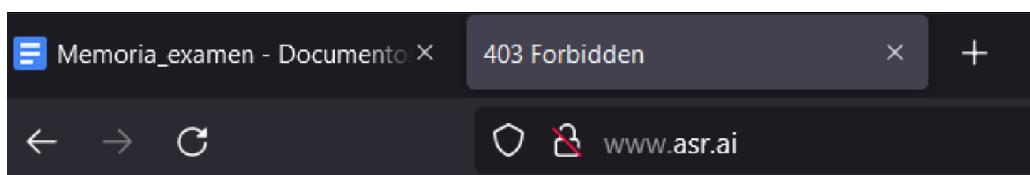
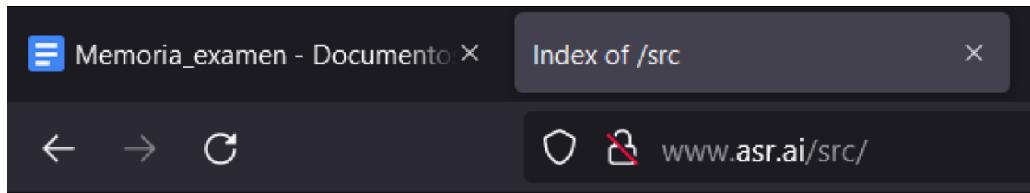


Figura 10: home

As we see, we can not access the content in the main domain.



## Index of /src

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

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

Figura 11: src route

However, we can access the content of src as we enabled the indexes option.

### 3.2. UTF-8

First of all, we need to import the module **libapache2-mod-php** to be able to use PHP in Apache. In order to use special characters, we need to edit the file **charset.conf** in **/etc/apache2/conf-available**.

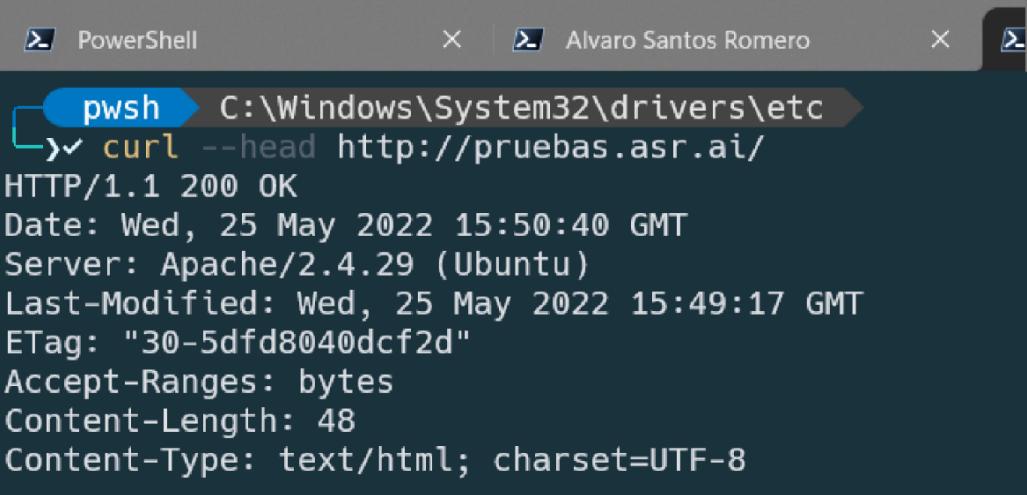
 A screenshot of a terminal window titled "PowerShell" and "Alvaro Santos Romero". The window contains the Apache configuration file "charset.conf" with the following content:
 

```
# 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 given
# in meta http-equiv or xml encoding tags.

AddDefaultCharset UTF-8
# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

Figura 12: enabling UTF-8

We can check if the configuration is working by using the tool **curl**.



```
pwsh 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
```

Figura 13: curl

### 3.3. Apache Engine

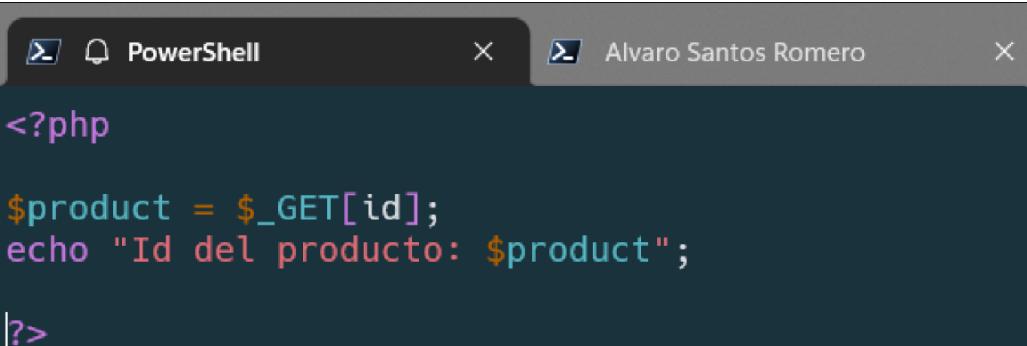
In order to rewrite rules, we need to enable the rewrite module.

- url conversion

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

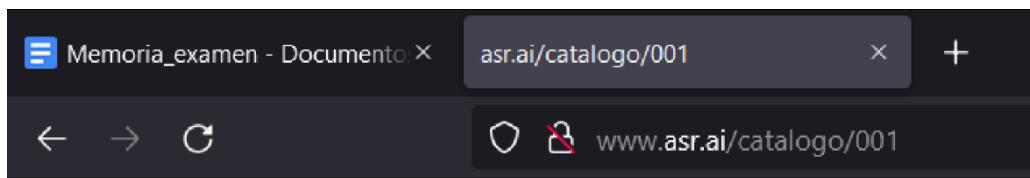
Figura 14: url conversion

- PHP file to get the variable



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

Figura 15: php variable



**Id del producto: 001**

Figura 16: php file output

- Subdomains

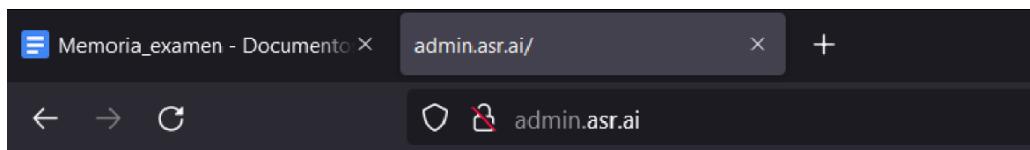


Figura 17: admin page

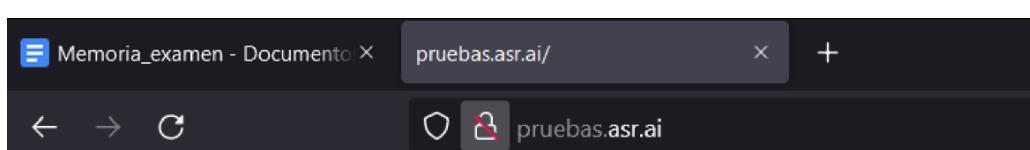
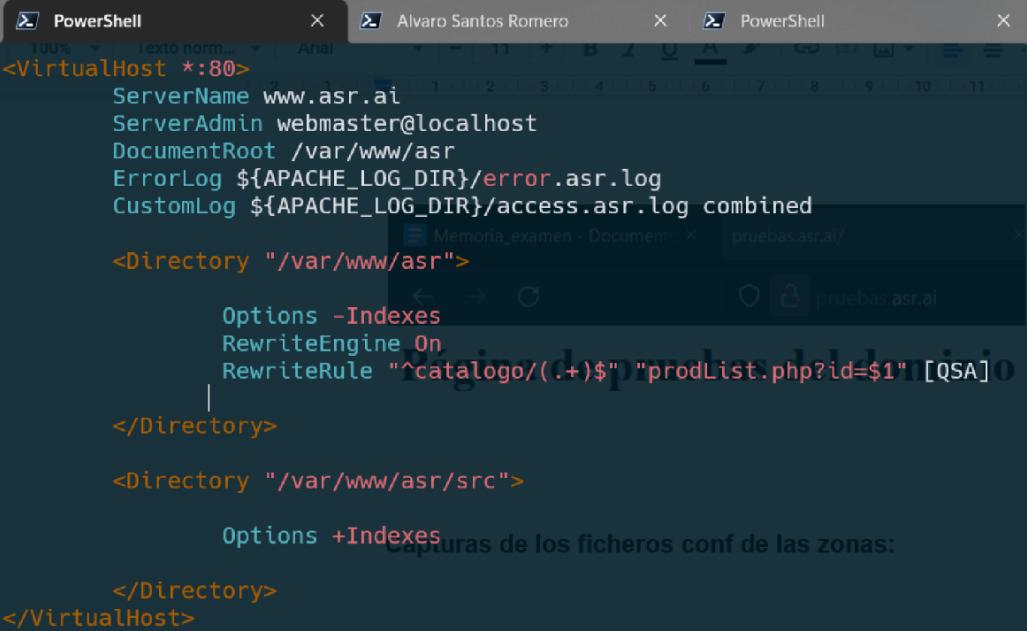


Figura 18: pruebas page

- Zones



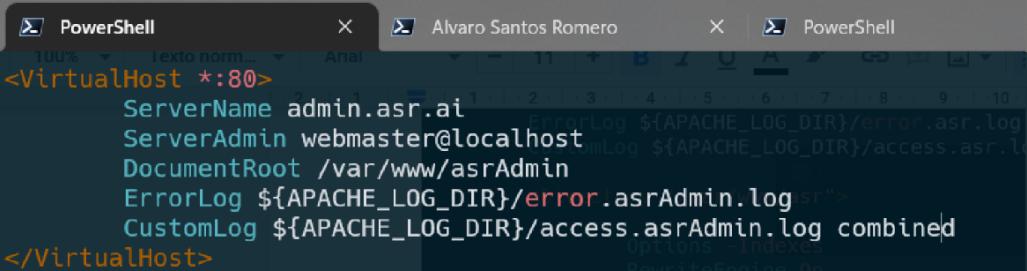
```
<VirtualHost *:80>
    ServerName www.asr.ai
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/asr
    ErrorLog ${APACHE_LOG_DIR}/error.asr.log
    CustomLog ${APACHE_LOG_DIR}/access.asr.log combined

    <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>
```

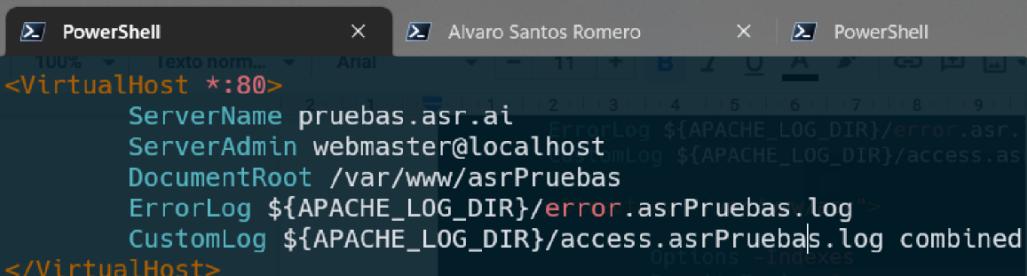
Capturas de los ficheros conf de las zonas:

Figura 19: www.asr.ai



```
<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>
```

Figura 20: admin.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>
```

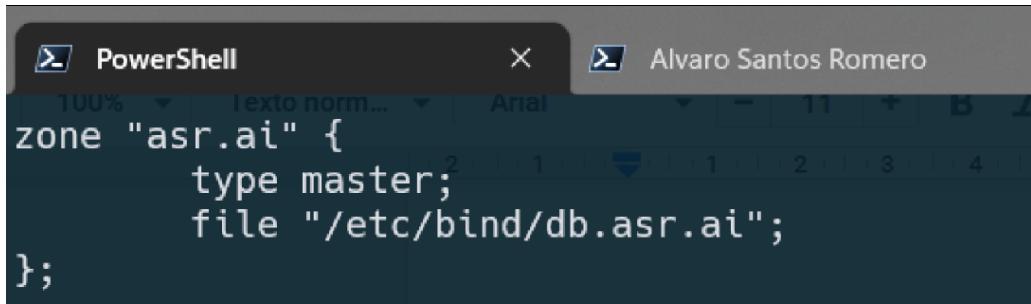
Figura 21: pruebas.asr.ai

## 4. Exercise 4: DNS config

Before hand, we will install **Bind9** in order to use the DNS.

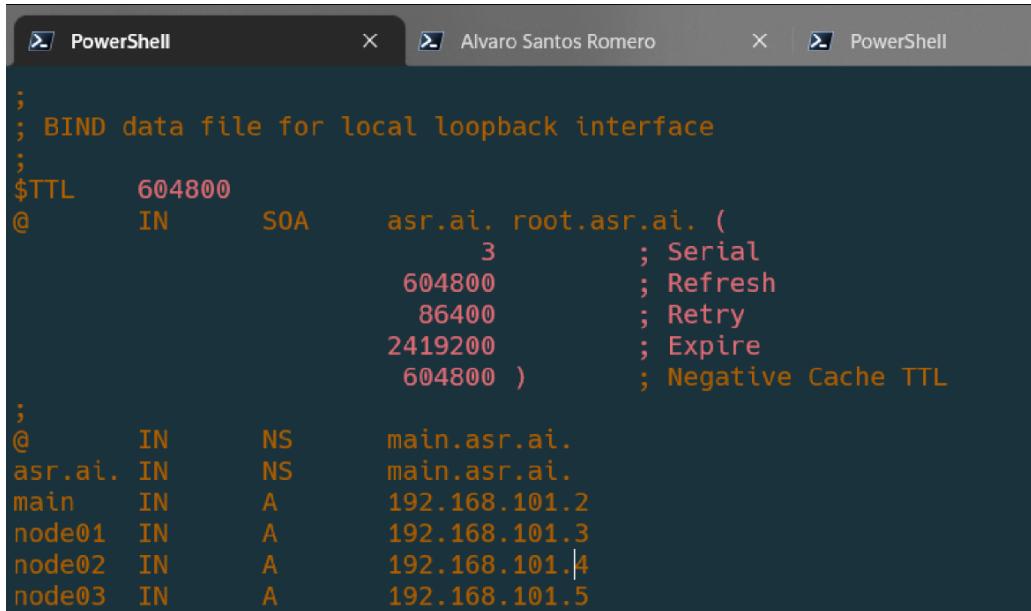
### 4.1. Direct Zones

We can create the zones in **named.conf.local**.



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

Figura 22: Direct zone asr.ai



```
PowerShell Alvaro Santos Romero PowerShell
;
; 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
```

Figura 23: DB file asr.ai

This file is used to store the domains and IPs of the different sites, so the DNS can associate the domain with the IP.

## 4.2. Inverse Zones

As we did with the direct zones, we can create them in `named.conf.local`.

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

Para ello, instalamos

Figura 24: Inverse zone 101.168.192

```
; BIND reverse data file for local loopback interface A
;
$TTL    604800
@       IN      SOA     main.asr.ai. zoroot.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.
```

Figura 25: db file 101.168.192

We can check if the zones are loaded by using:

```
name-checkconf -z named.conf.local
```

```
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 604800
vagrant@nodo1:/etc/bind$ |
```

Figura 26: zones check

## 5. Load Balancer

Before hand, we have to load these modules:

- proxy
- proxy\_http
- lbmethod\_byrequests

We will need to have an operative DNS installed in order to use the proxy.

We will use the default config file **000-default.conf** as we will not include any additional features to the proxy.

```
<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://node03.asr.ai:80" status=+H
    
```

Ejercicio 5: Balanceo de cargas

```
        ProxySet lbmethod=byrequests
    </Proxy>
        - proxy
        - proxy_http
    
```

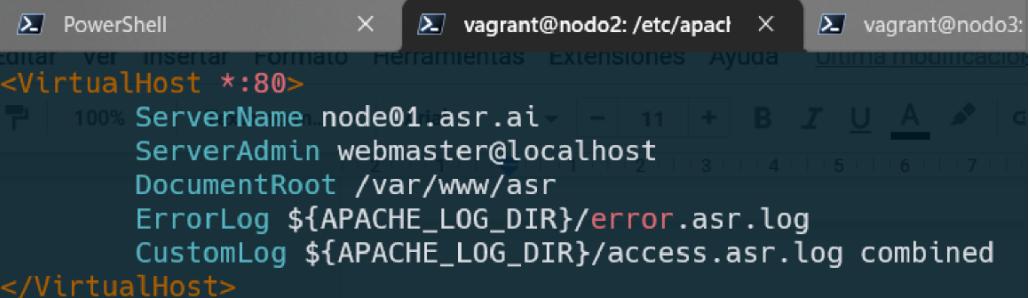
```
    ProxyPass "/" "balancer://balancerExamen"
    ProxyPassReverse "/" "balancer://balancerExamen"
</VirtualHost>
```

Ahora, deberemos de configurar el proxy en la máquina

Figura 27: 000-default.conf

- Balance Members: node01, node02 and node03.
- Nodes 01 and 02 respond to half of the requests.
- Node03 is acting as a backup in case any of the other nodes fails.
- Load method is by requests.
- ProxyPass and ProxyPassReverse is used to get and send the information between the nodes.

Now, we will duplicate the nodes in order to have the "same" server accepting the client's requests.



```
<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>
```

Figura 28: node server

Now, whenever a client sends a request to the server, one of the two nodes will answer.

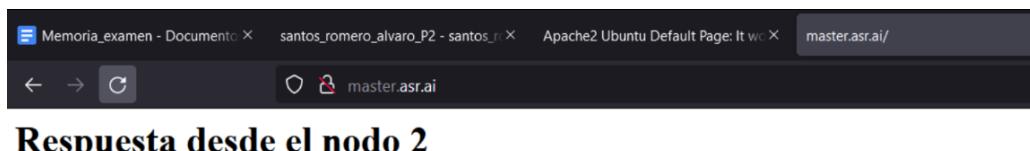


Figura 29: balancer working

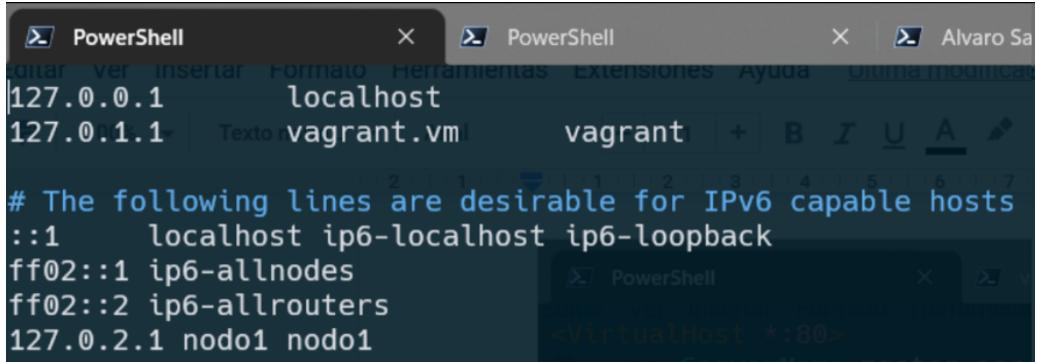
If we reload the page, we will see that the answers would come from node01 and node02. Node03 will answer only if both node01 and node02 are down.

### 5.1. Load Balancer with DNS

In general, **proxy\_mod** works with IPs by default.

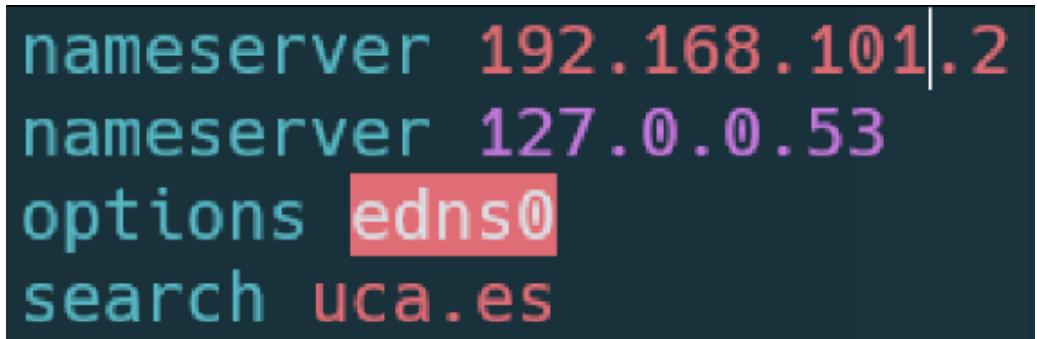
However, we can provide a DNS to be able to use the domain names instead of the IPs.

In order to use our DNS, we have to edit both the **hosts** and **resolv.conf** files.



```
127.0.0.1      localhost
127.0.1.1      vagrant.vm      vagrant
# 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
```

Figura 30: hosts file



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

Figura 31: resolv.conf file